INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Índice de Credibilidade de Conteúdos Noticiosos em Língua Portuguesa para Uso em Ambiente Escolar

MÁRCIA RAQUEL PINTO TEIXEIRA
Junho de 2021

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Índice de Credibilidade de Conteúdos Noticiosos em Língua Portuguesa para Uso em Ambiente Escolar

Márcia Raquel Pinto Teixeira

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Sistemas de Informação e Conhecimento

Orientador: Goreti Marreiros

Co-orientador: Ivone Amorim

Porto, Junho 2021

“The secret of change is to focus all of your energy, not on fighting the old, but on building the new.”

Socrates

Resumo

Nos dias de hoje, a desinformação e as notícias falsas são fenómenos cada vez mais frequentes. O rápido crescimento tecnológico permitiu um aumento exponencial deste tipo de eventos que, infelizmente, resultaram numa sociedade com níveis de desinformação alarmantes. As consequências que daqui advêm são especialmente preocupantes nas crianças e jovens. Este grupo etário é particularmente vulnerável pois, por um lado, têm contacto diário com diferentes plataformas digitais, onde se apresenta uma maior divulgação de informação falsa, e, por outro lado, têm ainda o seu pensamento crítico em desenvolvimento, possuindo desta forma menos defesas perante o fenómeno de desinformação.

Surge, neste contexto, o presente trabalho de dissertação intitulado "Índice de Credibilidade de Conteúdos Noticiosos em Língua Portuguesa para Uso em Ambiente Escolar", desenvolvido na organização MOG Technologies, no âmbito da unidade curricular de Tese / Dissertação / Estágio, do Mestrado em Engenharia Informática (TMDEI), do Departamento de Engenharia Informática (DEI) do Instituto Superior de Engenharia do Porto (ISEP). Em relação ao estudo efetuado sobre os vários trabalhos existentes relativos a este tema, verificou-se a existência de um maior número de projetos em língua inglesa comparativamente a apenas um único projeto desenvolvido em língua portuguesa, este com uma classificação binária. Durante a realização deste projeto desenvolveu-se um *dataset* de raiz de forma semi-automática, exclusivamente em língua portuguesa, com base em sites de notícias. O mesmo foi anotado em multiclasses de forma manual. De seguida, aplicaram-se diferentes técnicas de *Text Mining* aos dados presentes no mesmo e foram realizadas diversas experiências com o objetivo de encontrar o melhor algoritmo para classificar notícias. Foram efetuadas duas abordagens, a primeira com uma anotação multiclasses e a segunda em binário. Em todas as abordagens foram testados diferentes parâmetros de entrada e também vários algoritmos de classificação, tais como: *Multinomial Naive Bayes*, Máquina de Vetores de Suporte, *Gradient Boosting Classifier*, *XGBoosting Classifier*, *Logistic Regression, K-Nearest Neighbors Classifier*, *Decision Trees*, *Random Forest Classifier* e *Long Short-Term Memory*. Na abordagem binária, uma das experiências baseou-se numa arquitetura hierárquica e inovadora composta por três modelos. O primeiro modelo efetua uma análise textual da notícia, o segundo analisa os metadados presentes na mesma e o terceiro destina-se a efetuar uma ponderação entre os dois modelos descritos anteriormente para otimizar os resultados. O presente projeto conta com a apresentação de dois índices de credibilidade referentes ao conteúdo e aos metadados de uma determinada notícia e ainda um detetor de notícias falsas com 92 % de *performance*, sendo que o melhor algoritmo foi o Máquina de Vetores de Suporte. Por fim, estes modelos foram integrados num *web service* com o objetivo de validar toda a arquitetura implementada e, no futuro, espera-se que o mesmo possa ser integrado num motor de busca.

Palavras-chave: Notícias Falsas, *Dataset*, Língua Portuguesa, *Text Mining*, Índice de Credibilidade, Detetor de Notícias Falsas, *Web service*

Abstract

Nowadays, disinformation and fake news are phenomenons that are more often than ever before. The big and fast technological growth enabled a huge increase in the number of this kind of events which, unfortunately, resulted in a society with alarming levels of disinformation. The consequences of this are especially serious on the younger and older generations, which are the groups more vulnerable since, on the one hand, they have a great exposure to several digital platforms and, on the other hand, they have a natural lack of critical thinking skills.

In that way, the present dissertation, entitled "Índice de Credibilidade de Conteúdos Noticiosos em Língua Portuguesa para Uso em Ambiente Escolar", developed at the company MOG Technologies for the subject Thesis/Dissertation/Internship of the Master Degree in Informatic Engineering (TMDEI) for the Informatic Engineering Department (DEI) at Instituto Superior de Engenharia do Porto (ISEP).

In relation to the study made over several existent works about this theme, it was verified the existence of a greater number of projects in the English language comparatively to only one project developed in the Portuguese language, which has a binary classification. During the realization of the present project, it was needed the development of a dataset from scratch and exclusively in the Portuguese Language based on news websites where the same was manually noted in a multiclass way. Then, an application of different Text Mining techniques into the present data took place and several experiments were realized in order to find the best news classifier algorithm. Two approaches took place with the first one being based on multiclass notes and the second in binary. Not only the whole different entry parameters were tested in all approaches but also several classification algorithms such as: Multinomial Naive Bayes, Support Vector Machine, Gradient Boosting Classifier, XGBoosting Classifier, Logistic Regression, K-Nearest Neighbors Classifier, Decision Trees, Random Forest Classifier and Long Short-Term Memory. With the binary approach, one of the experiments provided an innovative hierarchic structure composed by three models. The first model executes a text analysis of the news, while the second one analysis the metadata contained on the news and the third one focus on the ponderation between the two previous models in order to optimize the results. The current project presents two results for the credibility indexes to the content and metadata of a certain news as well as a fake news detector with a performance of 92% with the Support Vector Machine being the best algorithm. Lastly, it was proposed a creation of a web service where the reached model is evaluated so, in the future, there are hopes of it being inserted on a search engine.

Keywords: Fake News, Dataset, Portuguese Language, Text Mining, Credibility Index, Fake News Detector, Web service

Agradecimentos

Neste ponto são bastantes as pessoas a quem gostaria de agradecer por terem cooperado para a conclusão deste projeto. Começo por agradecer às seguintes pessoas:
À instituição Instituto Superior Engenharia do Porto (ISEP) e em especial ao departamento (DEI) por todo o conhecimento que foi adquirido ao longo do mestrado, o que proporcionou a realização deste projeto.
À Empresa MOG Technologies, por me ter concedido a oportunidade de desenvolver um projeto enriquecedor e inovador.
À minha orientadora do ISEP, Doutora Goreti Marreiros e à minha co-orientadora, Doutora Ivone Amorim, por todas as recomendações, disponibilidade e suporte durante o decorrer deste projeto.
Ao meu supervisor da MOG Technologies Alexandre Ulisses, e em especial à colaboradora da empresa Inês Rito Lima, por todo o apoio, disponibilidade, orientação e confiança depositada em mim ao longo do desenvolvimento do projeto.
Aos meus amigos, em especial aos seguintes: Gonçalo Rodrigues, Rui Costa, Sofia Neves, Joel Peixoto, Nelson Palmas, Pedro Sousa, Joana Oliveira e Hugo Bento, por toda a compreensão e suporte ao longo deste período.
A todas as pessoas que conheci na MOG Technologies, em particular à Eduarda Fernandes, Matilde Valente, Luísa Lino e João Santos, por toda a motivação e companheirismo.
Por fim, deixo um agradecimento especial à minha família, pelo suporte e por me ter fornecido todas as condições para lutar e atingir todos os meus objetivos ao longo destes dois anos.

Índice

Lista de Figuras

Lista de Tabelas

Acrónimos e Símbolos

Lista de Acrónimos

AA	Aprendizagem Automática
AD	Árvore de Decisão
AHP	*Analythic Hierarchy Process*
API	*Application Programming Interface*
BPMN	*Business Process Model and Notation*
BoW	*Bag-of-Words*
CNN	*Convolutional Neural Network*
DL	*Deep Learning*
EDA	*Exploratory Data Analysis*
ET	*Extra Trees*
EUA	Estados Unidos da América
FAST	*Function Analysis System Technique*
GBC	*Gradient Boosting Classifier*
IA	Inteligência Artificial
INM	*Inference Network Model*
KNN	*K-Nearest Neighbors*

LIME *Local Interpretable Model-Agnostic Explanations*

LSI *Latent Semantic Indexing*

LSTM *Long short-term memory*

LSVM *Linear Support Vector Machine*

MVS Máquina de Vectores de Suporte

MNB *Multinomial Naïve Bayes*

NN *Neural Network*

PLN Processamento de Linguagem Natural

PLSI *Probabilistic Latent Semantic Indexing*

PSVM Polinomial Máquinas de Vetores de Suporte

QEF *Quality Evaluation Framework*

REST *Representational State Transfer*

RI Recuperação da Informação

RL Regressão Linear

RL Regressão Logística

RNN *Recurrent Neural Network*

RF *Random Forest*

RMVC Radial Máquinas de Vetores de Suporte

RSS *Really Simple Syndication*

SGD *Stochastic Gradient Descent*

TF-IDF *Term Frequency-Inverse Document Frequency*

VSM *Vector Space Model*

WS *Web Service*

WWW *World Wide Web*

XML *Extensible Markup Language*

1 Introdução

No presente capítulo, descreve-se o contexto da dissertação, o problema que motiva o desenvolvimento da mesma, os objetivos que se pretendem atingir, o processo necessário para a resolução do problema e, por último, uma apresentação sucinta de toda a estrutura do documento.

1.1 Contexto

As notícias falsas não são de todo uma novidade e as eleições presidenciais nos Estados Unidos da América (EUA), no ano de 2016, foram a rampa de lançamento para este fenómeno. O historiador Jacob Soll afirma que descobriu a origem do conceito muito antes, na invenção da impressora de Johanner Gutenberg's no ano de 1439. Com a implementação da impressão, o fenómeno das notícias falsas cresceu, dando lugar à criação de histórias como a existência de monstros marinhos, bruxas e a rumores sobre os pescadores serem os culpados pelos desastres naturais (Kalsnes, 2018).

Por outro lado, a evolução da Internet ofereceu um leque variado de tecnologias para a publicação de informação, tendo esta passado por três fases principais:

- Num primeiro momento, no ano de 1990, começaram a surgir vários *websites* e páginas *web*;
- De seguida apareceram vários serviços que permitiram às pessoas sem formação jornalística publicar informações;
- Por último, estes serviços evoluíram para plataformas digitais que permitem uma ligação direta entre os autores e os consumidores dos diferentes conteúdos (Turk, 2018).

Nos dias de hoje, a sociedade está cada vez mais suscetível à desinformação devido à facilidade com que a informação falsa é disseminada por qualquer pessoa. Associado a este facto está também o uso de múltiplas fontes digitais de informação e a evolução tecnológica que permite, por um lado, redes de banda larga extremamente rápidas e, por outro, o uso generalizado de dispositivos para o acesso à informação, tais como os *smartphones*.

Atualmente, as redes sociais são consideradas um meio de disseminação eficiente de notícias falsas e, infelizmente, cerca de 62% das pessoas consideram que as plataformas digitais como o Facebook, Twitter, Youtube, Instagram, entre outras, são uma fonte de informação segura (Elhadad, Li and Gebali, 2019). Neste contexto, em que as ideias, pontos de vista ou até ações das pessoas são facilmente

alterados, a verificação da veracidade de uma notícia passa a ser uma tarefa crucial (Shu *et al.*, 2020). Além de que, no caso das crianças e jovens, que estão cada vez mais próximas das tecnologias, este fenómeno é ainda mais preocupante. Isto porque, nestas idades, muitos processos cognitivos ainda não estão totalmente desenvolvidos, o que os torna mais vulneráveis.

Desta forma, torna-se essencial incutir nas pessoas, desde cedo, uma atitude crítica perante os *media*.

1.2 Problema

Conforme foi descrito na Secção 1.1, as notícias falsas e o nível da desinformação presente na sociedade não é um assunto novo mas é um problema sem solução imediata e única (Hofseth, 2017). Todavia, o fácil acesso à informação potencia a própria desinformação, o que resulta também de uma falta de espírito crítico em relação a todo o conteúdo noticioso divulgado.

Este fenómeno afeta a sociedade no seu todo, porém as crianças e jovens são o grupo mais vulnerável, onde as notícias falsas podem ter consequências dramáticas. Neste contexto, torna-se necessário desenvolver ferramentas que permitam a estes grupos de indivíduos desenvolver o seu espírito crítico e, assim, levá-los a avaliar a veracidade das notícias que os rodeiam.

O jornal Público, a Universidade de Aveiro e a MOG Technologies, conscientes deste problema, juntaram-se para o desenvolvimento de um projeto, designado TRUE, que tem como objetivo principal dotar as escolas e, consequentemente, os seus alunos, de ferramentas que permitam aumentar a literacia. Neste contexto, a MOG, sendo o único parceiro tecnológico do projeto, pretende desenvolver um ecossistema tecnológico que permita, entre outras coisas, a escrita de textos noticiosos para jornais escolares, a contextualização automática do conteúdo desses textos e o acesso a um motor de busca de notícias em língua portuguesa que classifique as mesmas em termos de credibilidade.
Surge, assim, a necessidade de se desenvolverem métodos que permitam a classificação automática de notícias (em língua portuguesa) no que diz respeito à credibilidade do seu conteúdo. Este era, precisamente, o objetivo principal desta dissertação, tal como vai ser descrito na próxima secção.

1.3 Objetivos

O principal propósito do trabalho descrito na presente dissertação consistiu no desenvolvimento de um modelo que permitisse classificar de forma automática as notícias no que diz respeito à sua credibilidade e a sua disponibilização através de um *web service*, para futuramente ser integrado num motor de busca de notícias. Este motor de busca não foi definido como um dos objetivos desta dissertação.

Para se atingir os objetivos definidos anteriormente, o projeto abrangeu as seguintes quatro etapas:

- Identificação de métricas e conceção de um índice de credibilidade de notícias;
- Construção um *dataset* de notícias;
- Desenvolvimento e treino de diferentes algoritmos de aprendizagem automática para a análise da credibilidade das notícias;

- Desenvolvimento de um *web service* que devolve o índice de credibilidade das notícias, que recorre ao modelo desenvolvido.

1.4 Abordagem Preconizada

Esta dissertação centrou-se na construção de um *dataset* e na sua anotação de forma manual, tendo em conta quatro critérios. Desenvolveu-se uma arquitetura composta por três modelos, sendo que dois deles remetem ao índice de credibilidade e o outro a um detetor de notícias falsas. E ainda, foi desenvolvido um *web service* onde os modelos foram integrados.

Na primeira fase do projeto foram executadas as seguintes tarefas: analisaram-se vários artigos científicos sobre o tema; procedeu-se à investigação das diferentes ferramentas de verificação manual de factos; verificaram-se os agregadores de notícias e as caraterísticas das mesmas necessárias para a construção do *dataset*; procedeu-se à análise de trabalhos relacionados com esta área e tecnologias envolventes e, por último, construiu-se uma possível arquitetura para toda a solução.

Na segunda fase do projeto procedeu-se à sua implementação, conforme se pode verificar nas seguintes etapas: criou-se um *dataset* de raiz e anotou-se o mesmo em várias classes; desenvolveu-se uma arquitetura hierárquica e inovadora composta por três modelos de aprendizagem automática e ainda um *web service* onde se integra os mesmos. Por último, elaborou-se o *design* da solução final.

Na Figura 1 apresenta-se todo o plano de trabalho desenvolvido, que foi devidamente acompanhado pelos orientadores da empresa através de reuniões semanais.

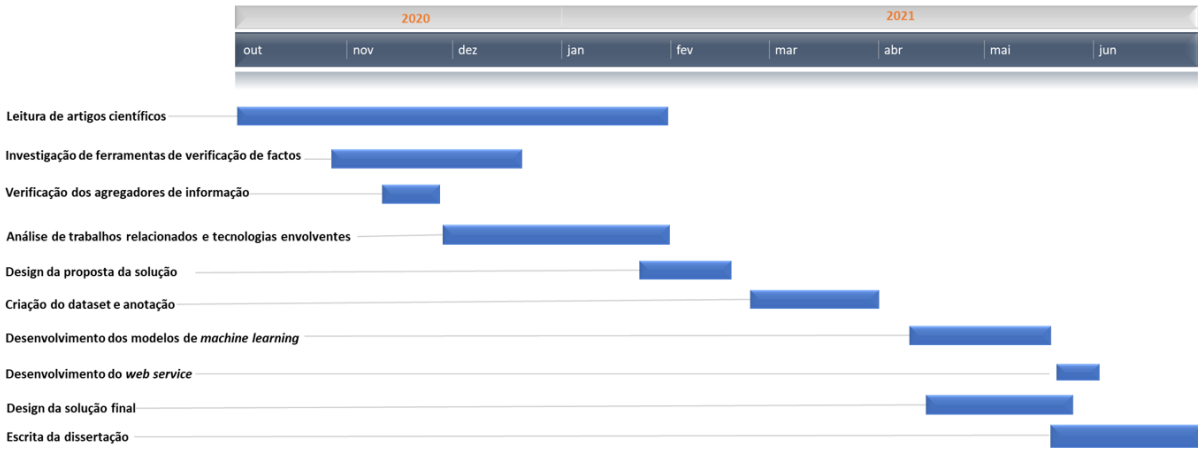


Figura 1 Plano de Trabalho

1.5 Contribuições

O presente trabalho teve como contribuições principais: a construção de um *dataset* exclusivamente em língua portuguesa baseado em *sites* de notícias fidedignos e não fidedignos; o desenvolvimento de uma arquitetura hierárquica e inovadora composta por três modelos; extração de dois índices de

credibilidade (conteúdo e metadados da notícia) e, por último, o desenvolvimento de um detetor de notícias falsas para notícias em língua portuguesa com elevada *performance*. Esta informação encontra-se sistematizada na publicação submetida na Conferência denominada *IEEE ISTAS21 (Engineering and Corporate Social Responsibility)* com o título *"A fake news detection and credibility ranking platform for Portuguese online news analysis"* (Pinto, M., Rito Lima, I., Amorim, I., Ulisses, A., Marreiros, G., 2021).

1.6 Estrutura do Documento

O presente documento encontra-se organizado nos seguintes oito capítulos: Introdução, Preliminares, Estado da Arte, Análise de Valor, Análise de Requisitos e Design, Implementação da solução, Experimentação e Avaliação, e por último, Conclusões.

O capítulo Introdução apresenta de uma perspetiva geral o contexto onde o projeto se insere, o problema que se pretendia resolver, os objetivos do mesmo e, por último, a abordagem aplicada.

O segundo capítulo, Preliminares, apresenta um contexto histórico sobre os conceitos de notícias falsas e desinformação, ações que se podem tomar para identificar se uma notícia é verdadeira ou falsa, bem como os *websites* existentes para a verificação manual de factos e uma exposição dos diferentes agregadores de notícias.

O terceiro capítulo, Estado da Arte, apresenta os trabalhos principais relacionados com o tema e uma análise das tecnologias que podem ser integradas na implementação da solução final.

O quarto capítulo, Análise de Valor, refere-se à análise de valor do projeto que consiste na identificação e análise da oportunidade, proposta de valor e, por último, o modelo *Function Analysis System Technique* (FAST) que expõe todas as funções do projeto.

O quinto capítulo, Análise de Requisitos e Design, aborda os seguintes pontos: os *stakeholders* do projeto e a análise de requisitos funcionais e não funcionais. No capítulo referente ao Design, apresenta-se numa primeira fase a proposta da solução que se pretende implementar expondo duas abordagens para o que se pretende implementar. E de seguida, o *design* de toda a solução desenvolvida, apresentando sempre duas soluções para a mesma solução.

No sexto capítulo, Implementação da solução, apresenta-se toda a solução técnica desenvolvida, a análise exploratória dos dados, a preparação dos mesmos para desenvolver os modelos de classificação, as várias experiências concretizadas durante todo o projeto e uma descrição do *web service* desenvolvido.

O sétimo capítulo, Experimentação e Avaliação, diz respeito à descrição do processo de avaliação e análise da solução, que englobou os seguintes passos: identificação da hipótese que se pretendia estudar, grandezas utilizadas para avaliar a solução, metodologia de avaliação da solução e *framework* de avaliação da qualidade.

No último capítulo, Conclusões, é efetuado um resumo de todos os objetivos atingidos, bem como as limitações identificadas, o trabalho que pode ser explorado no futuro e ainda as contribuições de todo o projeto desenvolvido.

2 Preliminares

Em meados de 1990, o rápido avanço no desenvolvimento *World Wide Web* (WWW), permitiu uma melhoria exponencial no que toca à comunicação a nível global. O aparecimento das redes sociais fez com que a divulgação de um grande volume de notícias se propagasse de forma mais rápida, e não validada, o que permitiu o surgimento de um elevado número de notícias falsas.

O presente capítulo tem como foco principal a contextualização do trabalho nesta dissertação, sendo que este se encontra dividido em seis secções. Na Secção 2.1, é introduzido o conceito teórico das notícias falsas. Na Secção 2.2, são apresentadas as diferentes categorias das notícias falsas e os motivos que proporcionam a divulgação das mesmas. Na Secção 2.3, são apresentadas as ações que qualquer cidadão comum pode concretizar para identificar se uma notícia é verdadeira ou falsa. Na Secção 2.4, são apresentados diferentes *websites* de verificação de factos que são considerados como a forma mais acessível de verificar se uma notícia é falsa. Na Secção 2.5, são descritos os diferentes agregadores de notícias e, por último, na Secção 2.6, é feito um resumo de todo o capítulo.

2.1 Conceito de Notícias Falsas

O termo "notícia falsa" está associado à disseminação de informação não verdadeira de forma intencional ou não intencional (Allcott and Gentzkow, 2017). Este é um problema antigo que inicialmente afetou a imprensa em papel, passando para a rádio e televisão, até chegar ao mundo digital onde a disseminação deste tipo de informação pode ser facilmente feita por qualquer pessoa, nomeadamente através das redes sociais (Shabani and Sokhn, 2018).

Um exemplo amplamente conhecido de uma série de notícias falsas que originou um grande impacto na sociedade é o chamado *Great Moon Hoax,* do ano de 1835, que se refere à publicação de um conjunto de seis artigos, num jornal Nova Iorquino, que revelavam diversas descobertas fantásticas sobre a Lua, umas atrás das outras, nomeadamente a existência de vida inteligente no único satélite natural da Terra (Shabani and Sokhn, 2018). István Vida, que fez um estudo detalhado sobre este tema (Vida, 2012), refere mesmo que o autor destes artigos noticiosos, Richard Adams Locke, conseguiu, através da sua imaginação, uma história de ficção científica tão perfeita que convenceu a generalidade das pessoas (incluindo cientistas) da veracidade das notícias. Quem não acreditava, sem conseguir justificar, era considerado alguém que não percebia nada de Astronomia. O sucesso alcançado por

estas notícias tornaram aquele que era um pequeno jornal de bolso num dos jornais mais vendidos do mundo e este era precisamente o objetivo inicial desta série de artigos.

Um exemplo mais recente refere-se às eleições presidenciais dos Estados Unidos da América (EUA) realizadas no ano de 2016. Alguns comentadores alegaram que as notícias falsas que foram disseminadas na altura, relativamente aos dois candidatos principais, tornaram possível a eleição de Donald Trump como presidente, em detrimento da candidata Hilary Clinton. De acordo com Allcott e Gentzknow (Allcott and Gentzkow, 2017), houve efetivamente um grande número de notícias falsas que foram partilhadas nas redes sociais e estas favoreciam maioritariamente o candidato Donald Trump. No entanto, o estudo apresentado por estes autores vai no sentido de mostrar que o impacto dessas notícias falsas, partilhadas nas redes sociais, não foi significativo ao ponto de ser um aspeto chave nos resultados eleitorais.

Os dois exemplos aqui apresentados mostram que o conceito de "notícia falsa" é muito abrangente. No caso específico do Great Moon Hoax, o autor criou uma história fictícia, extremamente sensacional, de forma a atrair a curiosidade dos leitores e, assim, aumentar as vendas do jornal para onde escrevia. No caso das eleições presidenciais de 2016 nos EUA, foram disseminadas informações falsas com o objetivo de favorecer um candidato presidencial em detrimento dos restantes.

A dificuldade em definir univocamente o conceito de "notícia falsa" tem levado a comunidade científica a adotar diferentes definições ou interpretações deste conceito, que por vezes estão relacionadas com o próprio contexto em que o termo é utilizado. Wardle (Wardle, 2018) define notícia falsa como "[..] informação fabricada que imita o conteúdo das notícias na sua forma, mas não no processo organizado ou intencional." Por sua vez, Parikh e P. K. Atrey (Parikh and Atrey, 2018) descrevem as notícias falsas como sendo histórias fabricadas que são apresentadas como se fossem de fontes legítimas com a intuito de enganar os leitores. Estes autores chamam também à atenção para o facto de a propagação deste tipo de notícias estar a ficar cada vez mais severa. Uma interpretação análoga é a apresentada por Mandical et al. (Mandical et al., 2020) que veem uma notícia falsa como uma história não verdadeira que é apresentada como sendo uma notícia genuína, com a intenção de enganar o público.

2.2 Categorização de Notícias Falsas

As notícias falsas podem ser construídas tanto por máquinas como por pessoas (Zhang and Ghorbani, 2020). Por norma, as máquinas são bots construídos com o propósito de divulgar rumores, spam, informação falsa e calúnias de forma automatizada, com estes a serem capazes de interagir com as pessoas a partir de plataformas digitais (Zhang and Ghorbani, 2020). No entanto, as notícias falsas produzidas por estes bots podem também ser divulgadas por seres humanos, por exemplo, através da sua partilha nas redes sociais. Uma vez que parte destes conteúdos são criados para muitas ocasiões específicas através de um simples registo online, torna-se desta forma difícil distinguir as informações verdadeiras daquelas que são falsas apenas pelo seu conteúdo e pela própria análise linguística das mesmas (Zhang and Ghorbani, 2020).

Na literatura alguns autores, nomeadamente Wardle em (Wardle, 2018), distinguem três tipos de notícias falsas: aquelas em que a informação está errada sem, no entanto, haver uma intenção

declarada do autor (*misinformation*), aquelas em que existe um propósito deliberado do autor na criação das mesmas (*disinformation*) e, por último, aquelas em que a informação, apesar de não ser falsa, é disponibilizada com o intuito de prejudicar alguém (*mal-information).*

Com base nesta distinção inicial, Wardle categorizou as notícias falsas em sete grandes grupos, comecemos então por descrever cada um dos grupos descritos anteriormente:

- "Satire or parody"

O propósito da sátira, que pode ser considerada como uma arte, é enfatizar uma opinião ou um ponto de vista sobre um assunto recorrendo ao humor. Este tipo de conteúdo humorístico não tem intenção de causar danos, mas é considerado uma fonte potencial de engano (Wardle, 2018).

Um dos *sites* mais conhecidos no mundo de publicação de notícias do tipo "satire" é o *The Onion*. Este parece um *site* de notícias verdadeiras, mas, na realidade só publica conteúdo deste género. A imagem apresentada na Figura 2 representa o início da pandemia em 2020 na qual houve uma procura louca por papel higiénico. Esta imagem foi divulgada no Diário de Notícias pelo criador de várias personagens de banda desenhada, Luís Louro, que pretendeu ilustrar a quarentena através da banda desenhada.

Figura 2 Cartoon que exemplifica a temática da sátira nas notícias[1]

- "False connection"

Esta categoria refere-se a notícias em que as capas das mesmas (legenda ou imagem) não apresentam qualquer relação com o conteúdo. O exemplo mais comum deste tipo de conteúdo é o *clickbait* (atrair as pessoas a partir de títulos atrativos) que é empregue em plataformas digitais, como o Facebook e o Twitter, com o objetivo de atrair a atenção dos utilizadores e assim adquirir um maior número de visualizações (Wardle, 2018). O título que se apresenta na Figura 3 - "Um homem tenta abraçar um leão selvagem, não vais acreditar no que acontece a seguir!" - representa um claro exemplo de uma falsa conexão.

[1] https://www.dn.pt/edicao-do-dia/22-abr-2020/covidiotas-o-humor-na-banda-desenhada-ja-descobriu-a-vacina-contra-a-covid-19-12090791.html

Figura 3 Conexão Falsa[2]

Com esta imagem pretende-se um aumento de pessoas a visualizarem a publicação e, cada vez mais, o *clickbait* é utilizado para atrair cliques mesmo que, ao ler a notícia, as pessoas se sintam enganadas.

- "Misleading content"

É um conceito usado para descrever o título ou toda a notícia criada para desinformar ou enganar os leitores através de conteúdo enganoso. Na maior parte das notícias com conteúdo enganoso, o objetivo principal é que o conteúdo das mesmas consiga influenciar o ponto de vista do leitor de maneira que este forme uma opinião, fique confuso ou discorde sobre um determinado assunto (Wardle, 2018) (Ireton, Posetti, and UNESCO, 2018).

- "False context"

O contexto falso é utilizado quando conteúdos genuínos são reorganizados com o propósito de atrair visualizações (Wardle, 2018) (Ireton, Posetti, and UNESCO, 2018). A Figura 4 apresenta um exemplo de um contexto falso pois no verão do ano de 2018, na Internet, circulou uma imagem sobre uma criança que se encontrava dentro de uma jaula. Apesar de esta imagem ser verdadeira, a mesma foi publicada num contexto completamente diferente dando lugar a uma mera encenação como parte de um protesto contra as políticas de imigração. No entanto, o autor, quando divulgou a mesma, não tinha noção que esta fazia parte de um protesto.

Figura 4 Conteúdo Falso[3]

[2] http://ethicalmediatraining.eu/training/activities/media-disinformation/

[3] https://firstdraftnews.org/long-form-article/understanding-information-disorder/

- "Imposter content"

Esta categoria refere-se a conteúdos falsos que utilizam fontes verdadeiras, nas quais estes são manipulados por fontes falsas ou conteúdos inventados (Scholz, 2019) (Ireton, Posetti, and UNESCO, 2018). A Figura 5 apresenta um vídeo criado sobre vítimas de violação de Bill Clinton durante as eleições presidenciais dos Estados Unidos da América no ano de 2016. Este vídeo circulou, na altura, com a autoria da empresa "NowThis" e esta acabou por ser obrigada a desmentir o vídeo publicado.

Figura 5 Conteúdo Impostor[4]

- "Manipulated content"

A categoria de conteúdo manipulado diz respeito ao conteúdo de uma notícia que é modificado com o propósito de enganar. Esta encontra-se muitas vezes em fotografias ou vídeos (Wardle, 2018) (Ireton, Posetti, and UNESCO, 2018). A Figura 6 ilustra um exemplo desse tipo de manipulação de conteúdo. No lado esquerdo da figura, encontra-se a fotografia verdadeira e do lado direito encontra-se uma imagem manipulada usando a ferramenta *Photoshop* com o intuito de fazer parecer que o presidente russo, Vladimir Puttin, estava a agarrar a gravata do Ex-Presidente dos Estados Unidos da América, Barack Obama.

Figura 6 Manipulação de conteúdo[5]

[4] https://firstdraftnews.org/long-form-article/understanding-information-disorder/
[5] https://guides.lib.uiowa.edu/c.php?g=849536&p=6077637

- "Fabricated content"

O conteúdo fabricado é 100% falso e tem como propósito enganar ou causar dano (Wardle, 2018) (Ireton, Posetti, and UNESCO, 2018). Na Figura 7 apresenta-se uma imagem que representa um exemplo deste tipo de conteúdo e que tem como protagonistas o Papa Francisco e Donald Trump.

Figura 7 Conteúdo Fabricado

Na figura anterior, podemos verificar uma afirmação do Papa Francisco, que foi manipulada com o objetivo de parecer um apoio a Donald Trump que era candidato às eleições dos Estados Unidos da América no ano de 2016. Este título foi divulgado pela Internet assim como outras diferentes notícias falsas antes das eleições presidenciais.

A Figura 8 apresenta um esquema, da autoria de Wardle, onde as sete categorias anteriormente nomeadas se encontram representadas e identificadas com os pontos principais que as distinguem e caracterizam numa escala que ele designa de "Intenção de Prejudicar".

Figura 8 Sete categorias de *Misinformation* e *Disinformation*

É importante salientar que a categorização das notícias falsas pode variar dependendo do autor. No entanto, um ponto fundamental a considerar-se sempre é qual a motivação subjacente à criação e publicação dessas notícias, uma vez que isto representa um fator muito importante no que diz respeito à sinalização destes eventos. Ou seja, quando se publica uma notícia, uma questão que deve sempre ser colocada é: Qual a consequência de publicar esta notícia/uma, notícia falsa?

Caso sejam publicadas notícias verdadeiras, existem duas partes que beneficiam: o leitor e o sujeito que a publica. Isto porque no caso de a notícia ser verdadeira, o leitor vai provavelmente passar a

confiar mais na fonte desta informação, revertendo-se isto em quem publica a notícia, que consegue um nível de confiança mais elevado por parte dos leitores nas suas publicações. Caso as notícias publicadas sejam falsas, existe um decréscimo na credibilidade da respetiva fonte de informação que, consequentemente, pode levar à perda de leitores e, em casos mais extremos, levar à falência da instituição a que essa pessoa pertence.

Numa perspetiva jornalística, se é necessário desenvolver soluções para estes problemas de *disinformation* e *misinformation*, também se torna relevante pensar em quem está a desenvolver a notícia e qual o motivo. Wardle (Wardle, 2018) apresentou oito motivos para a criação destes tipos de conteúdos, tendo como base quatro motivos, estes identificados por Eliot Higgins, e que apresentam algumas semelhanças com os trabalhos apresentados anteriormente. Estes motivos são: "*poor jounalism*"; "*to parody*"; "to *provoke or to 'punk'*"; "*passion*"; "*partisanship*"; "*profit*", "*political influence*" e "propaganda". Na Tabela 1 apresenta-se uma matriz que relaciona as motivações anteriormente mencionadas com os diferentes tipos de informação falsa.

Tabela 1 Motivos e as categorias de notícias falsas

Categorias / Motivações	Sátira ou Paródia	Conteúdo Enganador	Conteúdo Imposto	Conteúdo Fabricado	Falsa Conexão	Falso Contexto	Conteúdo Manipulado
Pobre Jornalismo		✓			✓	✓	
Paródia	✓		✓	✓			✓
Para Provocar ou 'mau jornalismo'			✓	✓			
Paixão						✓	
Partidário		✓				✓	
Lucro			✓	✓	✓		
Influência Política		✓		✓		✓	✓
Propaganda		✓	✓	✓		✓	✓

Finalmente, Wardle (Wardle, 2018) destaca a importância de refletir sobre a forma como o conteúdo noticioso está a ser divulgado, evidenciando três mecanismos de disseminação da informação. O primeiro consiste numa parte da informação ser divulgada involuntariamente pelas pessoas através das redes sociais, recorrendo ao uso da partilha sem verificar o seu conteúdo. O segundo identifica os jornalistas como uma parte capaz de relatar, com precisão e em tempo real, a informação emergente nas redes sociais. O terceiro refere-se à informação que é divulgada por grupos que estão deliberadamente a tentar influenciar a opinião pública (Wardle, 2018).

2.3 Identificação de Notícias Falsas

Conforme foi referido na Secção 2.2, as notícias falsas são difundidas nas plataformas digitais (redes sociais) como se fossem notícias verdadeiras. Estas notícias são criadas de forma apelativa e construídas com o propósito de obter a atenção dos leitores, garantindo assim que as mesmas

atingem uma grande audiência. Com o objetivo de diminuir a propagação de notícias falsas e, assim, reduzir o impacto negativo que as mesmas possam ter, têm vindo a ser desenvolvidas estratégias que permitam identificar este tipo de notícias.

Apresentam-se, de seguida, dezanove ações que podem permitir identificar se uma notícia é falsa ou verdadeira:

1. Consultar pessoas especializadas na área (jornalistas e autores) – quando não se tem a certeza da veracidade de uma notícia, podem-se procurar jornalistas ou autores que permitam esclarecer a autenticidade da mesma (Hewitt, 2017).
2. Verificar se a notícia tem identificado o órgão de comunicação social – quando uma notícia verdadeira, o órgão de comunicação social aparece sempre associado à mesma (Pena, 2019).
3. Verificar se a notícia já foi publicada noutro *site* – a notícia pode ser publicada em várias plataformas digitais, sendo que fazer uma pesquisa pela mesma pode permitir encontrar notícias análogas e identificar a informação falsa (Sharma and Sharma, 2019a).
4. Verificar se a notícia tem identificado o autor e/ou referências – as notícias verdadeiras tem sempre associado o autor e as referências, pelo que caso essa informação não seja disponibilizada, pode-se assumir que a notícia é falsa (Jaroucheh *et al.*, 2020) (IFLA, 2021) (Hewitt, 2017).
5. Verificar a data de publicação da notícia – por vezes é utilizada uma notícia que já foi publicada há muito tempo atrás (Sharma and Sharma, 2019a).
6. Ler a notícia de forma completa (título, conteúdo e referências) – algumas notícias surgem com um título convidativo para obter mais visualizações, no entanto o título pode não corresponder ao conteúdo, o que indica que a notícia é falsa (Sharma and Sharma, 2019a).
7. Verificar a extensão da fonte da notícia – um dos indícios nas notícias não serem fidedignas é o URL terminar com ". com.co" em vez de ".com", ou seja, "abcnews.com" é um *site* fidedigno, mas, no entanto, existe um "abcnews.com.co" que imita o *site* anterior (Thomas, 2020) (IFLA, 2021).
8. Verificar se a imagem não foi adulterada – algumas notícias contêm imagens que são falsas para obter mais atenção dos utilizadores (Thomas, 2020).
9. Verificar se o título da notícia não se encontra em maiúsculas – uma notícia com o título em maiúsculas é mais apelativa para os leitores mas, por norma, estas são notícias falsas (Thomas, 2020).
10. Verificar o URL (se existe uma ligação segura HTTPS/HTTP) – se uma notícia não contiver uma ligação segura no URL pode indicar que não é de uma fonte confiável e por essa razão pode ser considerada uma notícia falsa (Sharma and Sharma, 2019a) (Davis, 2016).
11. Verificar se o conteúdo se trata de uma piada – grande parte das notícias falsas são construídas com base em publicações de *sites* humorísticos/satíricos (Sharma and Sharma, 2019a) (IFLA, 2021).
12. Verificar se a página da fonte da notícia contém uma Secção do tipo "Sobre nós" – procurar mais informações na página inicial da Secção "sobre nós", para verificar onde está a ser publicado o artigo (Thomas, 2020) (Davis, 2016).
13. Verificar os tempos verbais – algumas notícias falsas contêm vários tempos verbais diferentes, assim como vários verbos auxiliares (Afroz, Brennan and Greenstadt, 2012) (Fuller, Biros and Wilson, 2009).

14. Verificar a utilização da 1º pessoa do singular – os criadores de notícias falsas por vezes escrevem o seu conteúdo na primeira pessoa no singular para expressarem a sua opinião (Afroz, Brennan and Greenstadt, 2012).
15. Verificar a utilização da 1º pessoa do plural - os criadores de notícias falsas por vezes escrevem o seu conteúdo na primeira pessoa do plural (Afroz, Brennan and Greenstadt, 2012).
16. Recurso à voz passiva na notícia toda – as notícias verdadeiras são normalmente escritas em voz passiva (Fuller, Biros and Wilson, 2009).
17. Verificar a frequência das palavras numa notícia – uma notícia falsa contém várias palavras para enfatizar uma ideia (Fuller, Biros and Wilson, 2009).
18. Verificar se a notícia tem erros ortográficos – uma notícia verdadeira normalmente não contém erros ortográficos (Hewitt, 2017) (Pena, 2019).
19. Verificar a quantidade de parágrafos numa notícia – as notícias falsas por vezes não efetuam mudanças de parágrafos no texto (Zhou et al., 2019).

De seguida, apresentam-se alguns exemplos práticos de como três das ações anteriormente referidas permitem identificar as notícias falsas, nomeadamente, as ações relacionadas com: a alteração da imagem original, com a verificação da extensão da fonte e, por último, com a leitura da notícia completa.

A Figura 9 apresenta duas imagens semelhantes, onde um piloto é visto a tirar uma fotografia com uma janela do avião aberta. A imagem apresentada do lado direito, pode ser considerada verdadeira, uma vez que o avião se encontra na pista de aterragem. No entanto, a imagem do lado esquerdo é falsa pois, tal como está assinalado com o círculo a vermelho, o avião encontra-se acima das nuvens e sabe-se que não é possível abrir a janela de um avião quando o mesmo se encontra a uma certa altitude, pois a abertura de compartimentos para o exterior não é segura nessas circunstâncias.

Figura 9 Exemplo duma imagem modificada[6]

Este é um exemplo de uma notícia falsa resultante da alteração da imagem original. A ferramenta disponibilizada pela Google, denominada de *Google Images*[7], pode ser uma ajuda fundamental neste tipo de verificação uma vez que permite pesquisar se existem fotos iguais publicadas noutros locais, num outro contexto ou noutra data.

De seguida, na Figura 10, apresentam-se os URL's para dois *websites* de notícias. Tal como se pode verificar, o endereço do *website* apresentado do lado esquerdo termina em ".com.co" e o do lado

[6] https://www.boredpanda.com/fake-news-photos-viral-photoshop/?utm_source=google&utm_medium=organic&utm_campaign=organic
[7] https://www.google.com/imghp?hl=pt-pt

direito termina em ".com". Pode-se então concluir que o *website* correto é o do lado direito, sendo o outro é uma versão falsa do *website* da ABC News.

Figura 10 Exemplo de uma notícia publicada com um endereço web semelhante a uma fonte verdadeira[8]

Por último, na Figura 11, apresenta-se o título de uma notícia (abaixo) e o conteúdo da mesma (acima). Tal como se pode verificar, o título da notícia está descontextualizado da frase mencionada pelo Ex-Presidente dos Estados Unidos da América, Donald Trump. Este exemplo mostra como a leitura completa da notícia pode permitir distinguir uma notícia falsa de uma verdadeira.

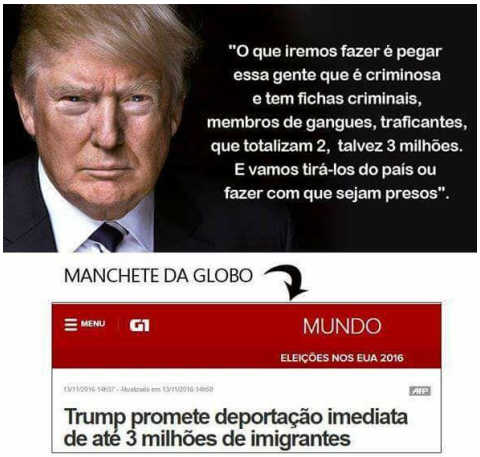


Figura 11 Exemplo de que se deve ler sempre toda a notícia[9]

2.4 Websites de Verificação de Factos

A forma mais tradicional de verificar se uma notícia é falsa foca-se essencialmente num processo de verificação manual de factos. De facto, este tipo de verificação está presente no âmbito jornalístico desde que este existe, contudo, a sua aplicabilidade tem sido mais notória com o grande desenvolvimento tecnológico e a recente utilização das redes sociais.

A verificação de factos tem como principal objetivo combater a desinformação, sendo que a forma mais comum de o fazer passa pela revisão manual de textos, vídeos, gravações e outras formas de conteúdo. Esta verificação pode hoje ser realizada por diversos tipos de utilizadores, sejam eles

[8] https://abc7.com/fake-news-facebook-snopes-fact-check/1621221/

[9] https://veja.abril.com.br/blog/felipe-moura-brasil/os-autoanistiadores-politicos-e-jornalistas-tentam-varrer-suas-culpas-para-baixo-do-tapete/

jornalistas ou cidadãos comuns numa lógica de *crowdsourcing*[10]. Alguns exemplos desta verificação são, por exemplo, a pesquisa de um jornalista pela veracidade de uma notícia, um utilizador de uma rede social confrontar amigos sobre uma dada informação que foi partilhada na rede, ou através da utilização de *websites* especializados que se dedicam unicamente à verificação de factos.

De facto, os *websites* especializados na verificação de factos podem ser uma ajuda fundamental para o cidadão comum verificar a veracidade de uma notícia antes de a publicar numa rede social ou de a partilhar através de qualquer outro meio.

Atualmente, por todo o mundo, existem vários projetos desenvolvidos nesta área de combate à desinformação, sendo notório o trabalho desenvolvido nos Estados Unidos da América, Argentina e Brasil, respetivamente o PolitiFact, o Chequeado e a Agência Lupa. No nosso país existem três ferramentas direcionadas para a verificação de factos, nomeadamente o Polígrafo, o Fact-Check Observador e, por último, a Prova dos Factos. Em destaque surge a ferramenta Polígrafo pois consiste num projeto desenvolvido exclusivamente em português, sendo disponibilizado em dois formatos (formato televisivo (SIC) e digital), e que classifica o grau de veracidade de uma informação em função de uma escala de sete níveis.

No resto desta Secção 2.4 são apresentadas algumas plataformas de verificação de factos com base em investigação jornalista e com o propósito de combater a desinformação, assim como uma análise comparativa das mesmas.

2.4.1 Polígrafo

O Polígrafo [11] foi o primeiro projeto português capaz de verificar factos e direciona-se para a verificação de rumores virais, que foram expostos nas redes sociais, sobre temas como o ambiente, a saúde, a economia e a tecnologia.

A metodologia implementada para verificar a informação, consiste em cinco passos: consultar a fonte de uma notícia, contextualizar a informação, consultar fontes de informação para complementar o processo de verificação, procurar a explicação dos autores da afirmação e avaliar a informação numa escala de avaliação. Esta escala de avaliação é feita em sete níveis (“Verdadeiro”, “Verdadeiro, mas...”, “Impreciso”, “Descontextualizado”, “Manipulado”, “Falso” e “Pimenta na língua”). A Figura 12 apresenta alguns exemplos de notícias classificadas pelo Polígrafo.

[10] Pesquisa em grupo

[11] https://poligrafo.sapo.pt/fact-checks

Figura 12 Notícias classificadas na ferramenta Polígrafo

2.4.2 Fact-Check Observador

O Fact-Check Observador[12] é um programa de verificação de factos para combater a desinformação do Jornal Observador. A sua classificação é feita em seis níveis, designadamente: "Verdadeiro"; "Praticamente verdadeiro"; "Inclusivo"; "Esticado"; "Enganador" e "Falso". A Figura 13 apresenta seis notícias classificadas de acordo com o programa Fact-Check do jornal Observador.

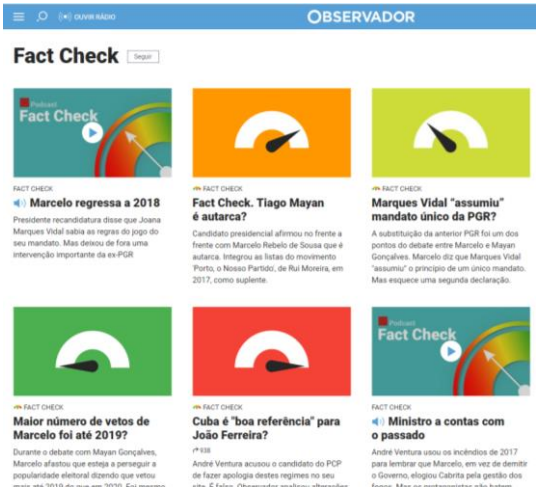


Figura 13 Notícias classificadas no programa Fact-Check Observador

2.4.3 Prova dos Factos

O jornal Público construiu um espaço denominado de "Prova dos Factos"[13] e o seu propósito destina-se a desconstruir a desinformação. Na página inicial deste *site* encontra-se o seguinte *slogan* "Há quem fale, fale – e às vezes minta. Numa era de ruído e desinformação, escrutinamos o que dizem os protagonistas do espaço público". Este classifica as suas notícias em quatro níveis como: "Verdadeiro",

[12] https://observador.pt/seccao/observador/fact-check/
[13] https://www.publico.pt/prova-dos-factos

"Parcialmente Verdadeiro", "Parcialmente Falso" e "Falso". A Figura 14 ilustra quatro notícias avaliadas por este *site*.

Figura 14 Análise de algumas notícias efetuadas pela A Prova dos Factos

2.4.4 FactCheck.org

O FactCheck.org[14] foi um projeto que começou em 2003 nos Estados Unidos da América (EUA) e tem como objetivo principal confirmar a veracidade das declarações dos vários líderes políticos, focando a sua atenção nas campanhas eleitorais do seu país. A metodologia utilizada por este projeto passa pelo contacto direto com o indivíduo/organização responsável para solicitar dados/fontes originais. No entanto, recorrem também a especialistas na área para ajudar a interpretar os dados obtidos.

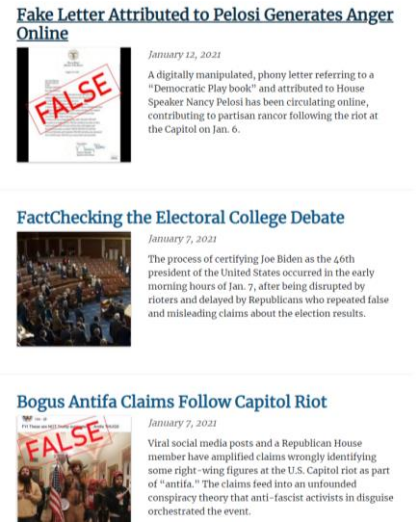


Figura 15 Análise de algumas notícias efetuadas pelo Fact Check

[14] https://www.factcheck.org/

2.4.5 PolitiFact

O PolitiFact[15] é um *website* que foi desenvolvido no ano 2007 nos Estados Unidos da América (EUA) que se dedica a confirmar a veracidade das afirmações de políticos e analistas, medindo-as através de um leque de classificações, denominado de *Truth-O-Meter.* Este classifica as notícias nas seguintes categorias: "Verdadeiro", "Na sua maior parte é verdade", "Meia Verdade", "Na sua maior parte é falso", "Falso" e "Calças em Chamas". A metodologia utilizada por este *website* é similar à da ferramenta FactCheck. A Figura 16 apresenta um exemplo de três afirmações ditas por Donald Trump e Mike Gallagher, estando estas classificadas em "Falso", "Na maior parte é falso" e "Verdadeiro".

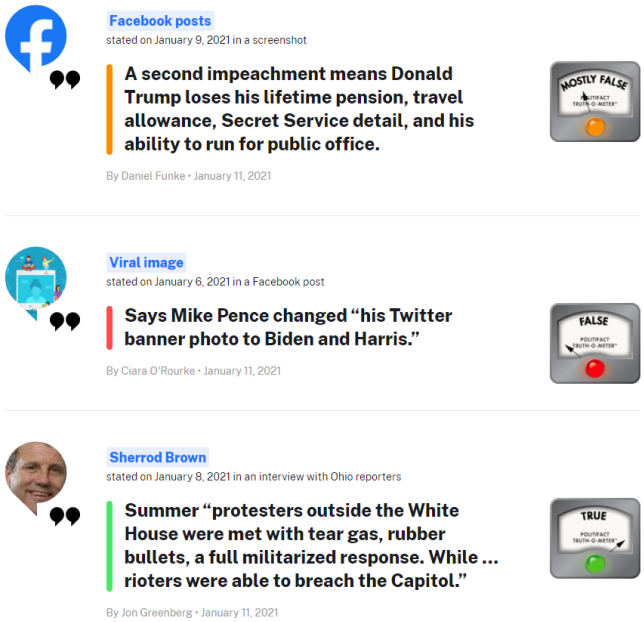


Figura 16 Notícias classificadas na ferramenta Politifact

2.4.6 Snopes

A ferramenta Snopes[16] apareceu no ano de 1994 com o propósito de investigar rumores e lendas urbanas. A metodologia utilizada nesta ferramenta, consiste em estabelecer contacto com a fonte, mas também procurar os indivíduos e organizações que tenham conhecimento sobre o assunto. Esta ferramenta classifica as notícias como, "Verdadeira"; "A maior parte é verdade"; "A maior parte é falso" e "Falso". Na Figura 17 apresenta-se uma lista de notícias classificadas de acordo com estes níveis.

[15] https://www.politifact.com/factchecks/list/?ruling=mostly-true
[16] https://www.snopes.com/fact-check/

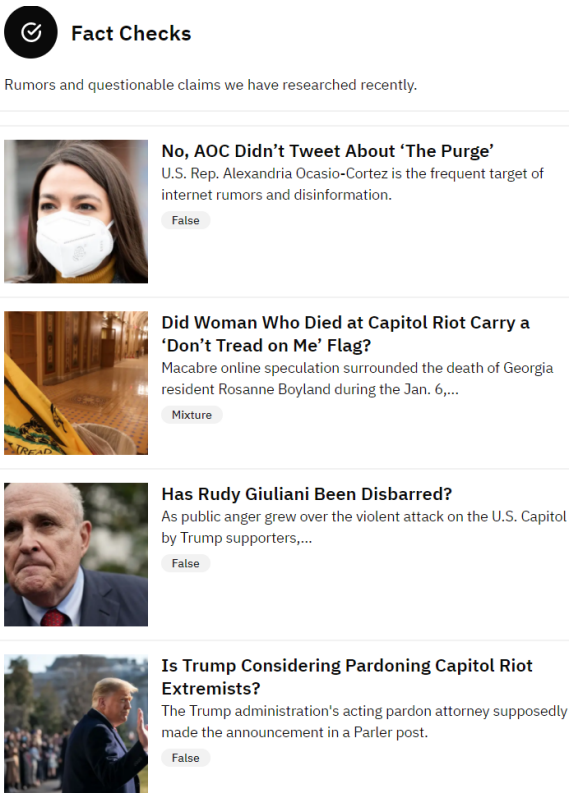
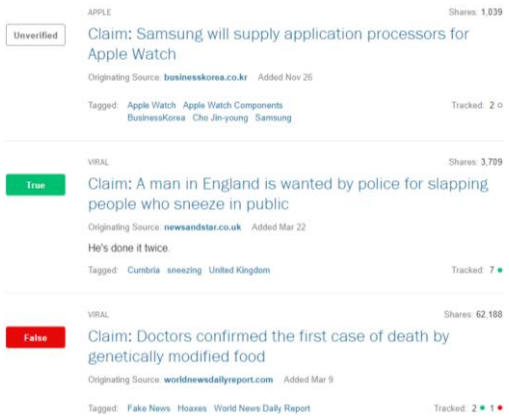


Figura 17 Notícias classificadas na ferramenta Snopes

2.4.7 Emergent

A ferramenta Emergent[17] rastreia, verifica ou desmascara rumores e conspirações em tempo real, faz parte de um projeto de pesquisa do centro digital de jornalismo da Universidade da Colômbia que se foca nas informações e rumores que não estão descritos nos *media*. Conforme se pode ver na Figura 18, na página inicial do seu *website* encontra-se uma lista de rumores que foi investigada, tendo cada rumor sido classificado com uma das seguintes opções: "Verdadeiro", "Falso" ou "Não Verificado".



[17] http://www.emergent.info/

Figura 18 Notícias classificadas na ferramenta Emergent

2.4.8 Chequeado

O Chequeado[18] é um *site* argentino que se dedica a fazer verificações diárias nos discursos de políticos, economistas, empresários, pessoas conhecidas, *media* e conteúdo que se apresenta viral nas redes sociais. Estes conteúdos são classificados de acordo com a consistência dos factos e dos dados. A metodologia deste *site* baseia-se em seis passos: selecionar uma frase mencionada publicamente; ponderar a sua relevância; consultar a fonte original e alternativas; verificar o contexto; confirmar, relativizar ou negar a declaração e por último, classificar. A classificação dos factos é feita em quatro níveis, nomeadamente, "Verdadeiro"; "Enganador"; "Insustentável" e "Falso". A Figura 19 apresenta um excerto de algumas notícias que já foram classificadas pelo *website*.

Figura 19 Notícias classificadas na ferramenta Chequeado

2.4.9 Agência Lupa

A Agência Lupa[19] foi a primeira empresa especializada na verificação de factos do Brasil. Esta empresa baseou-se nas plataformas descritas anteriormente nomeadamente, Chequeado e PolitiFact. O principal objetivo desta empresa é analisar afirmações feitas por políticos, líderes sociais e celebridades, pelos jornais, revistas, rádios, programas de televisão e Internet.

A metodologia utilizada pela Agência Lupa consiste na seleção da frase que pretende trabalhar, seguida de três critérios de relevância. Analisam preferencialmente afirmações feitas por personalidades de destaque nacional, assuntos de interesse público e/ou que tenham ganho destaque na imprensa ou na Internet. O *website* que esta empresa providencia classifica as notícias em nove níveis, sendo estes, "Verdadeiro", "Verdadeiro, mas", "Ainda é cedo para dizer", "Exagerado", "Contraditório", "Subestimado", "Insustentável", "Falso" e "De olho". A Figura 20 apresenta duas notícias classificadas por esta agência.

[18] https://chequeado.com/tag/falso-en-las-redes/
[19] https://piaui.folha.uol.com.br/lupa/tag/fact-checking/

Figura 20 Notícias e respetivas classificações efetuadas pela Agência Lupa

2.4.10 Truco

O Truco[20] foi um projeto de verificação de factos desenvolvido no Brasil, entre o ano de 2014 e 2018. Este tem como objetivo analisar o discurso de várias pessoas (políticos e pessoas conhecidas) mas também verificar as informações que circulam na Internet ou nas redes sociais. A metodologia utilizada neste *website* consiste na seleção de uma frase para ser verificada, e declarações analisadas são apenas escolhidas as que têm relevância eleitoral. Se possível entram em contacto com a campanha do candidato e solicitam a fonte de informação, em caso de necessidade também entram em contacto com especialistas. Posteriormente, comparam as informações que têm com as fornecidas e concluem se a afirmação é "Falsa", "Discutível", "Sem contexto", "Exagerado", "Subestimado", "Impossível de provar" ou "Verdadeira". A Figura 21 representa as sete classificações para as informações verificadas.

Figura 21 Classificação no projeto Truco

[20] https://apublica.org/checagem/

2.4.11 Truth-Or-Fiction

O Truth-Or-Fiction[21] é um *site* desenvolvido desde 1999 que também tem como objetivo identificar lendas urbanas e rumores na Internet. Este classifica as notícias em "Verdadeira" ou "Falsa", conforme se pode verificar na Figura 22.

Figura 22 Classificação da notícia "Facial recognition firm claims Antifa infiltrated Trump protesters who stormed Capitol"

2.4.12 GossipCop

O GossipCop[22] é um *site* que verifica factos de celebridades desenvolvido em Nova Iorque. Este investiga histórias de entretenimento divulgadas em revistas e jornais, mas também *online*, que classifica as notícias como sendo "Verdadeiras" ou "Falsas", conforme se pode verificar na Figura 23.

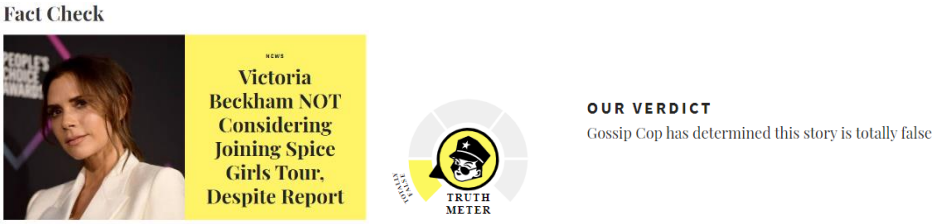


Figura 23 Notícia sobre Victoria Beckham classificada como totalmente falsa pelo GossipCop

2.4.13 Fiskkit

O Fiskkit[23] é um *site* onde cada utilizador pode carregar as suas notícias e classificá-las como sendo falsas ou verdadeiras (Hassan and Meziane, 2019) (Torabi Asr and Taboada, 2019). A Figura 24 apresenta a página inicial da ferramenta onde aparece o pedido de autenticação para carregar as notícias e dar a sua classificação.

[21] https://www.truthorfiction.com/category/fact-checks/
[22] https://www.gossipcop.com/
[23] https://fiskkit.com/

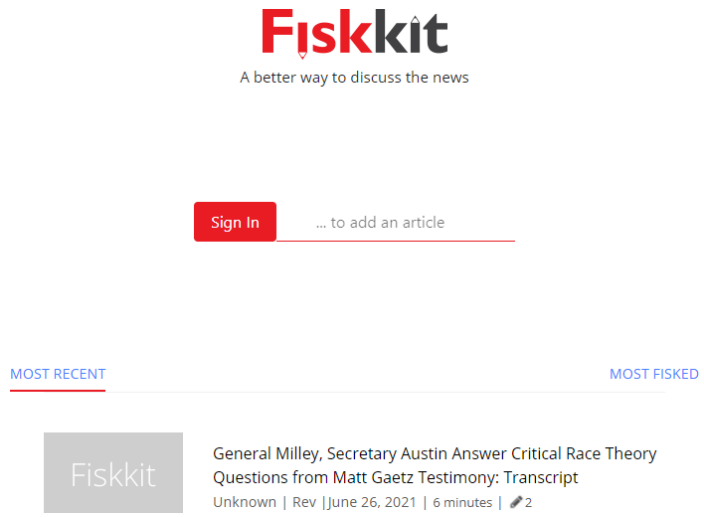


Figura 24 Página oficial do Fiskkit

2.4.14 Análise Comparativas dos Websites

Tal como foi apresentado anteriormente, existem várias ferramentas de verificação de factos *online* capazes de analisar várias fontes de informação. Algumas destas ferramentas especializaram-se na verificação de factos em temáticas específicas, nomeadamente em declarações políticas, histórias divulgadas nas revistas/jornais e notícias relativas a celebridades. Genericamente, os temas cobertos por estas ferramentas dirigem-se para o ramo da política, o da economia e o ramo social. Na Tabela 2, apresenta-se uma análise comparativa das diversas plataformas referidas anteriormente.

Tabela 2 Comparação dos websites de verificação de factos

Websites de verificação de factos	Temas			Fontes de Informação	Suporte de Linguagem	Níveis de Classificação de Credibilidade
	Política	Economia	Social			
FactCheck.org[24]	✓			Reclamações/ declarações políticas	Inglês	Verdadeiro; Não há evidências; Falso
Snopes[25]	✓	✓	✓	Várias fontes de dados	Inglês	Verdadeiro; Maior Parte Verdadeiro; Maior Parte Falso; Falso
PolitiFact[26]	✓	✓	✓	Reclamações/ declarações políticas	Inglês	Verdadeiro; Maior Parte Verdadeiro ; Metade Verdadeiro; Maior Parte Falso; Falso; Calças em fogo
Emergent[27]	✓	✓	✓	-	Inglês	Verdadeiro; Não Verificado; Falso
GossipCop[28]			✓	Investiga histórias de entretenimento publicadas em revistas e jornais	Inglês	Escala entre valores de 0-10
Polígrafo[29]	✓	✓	✓	Várias fontes de dados	Português de Portugal	Verdadeiro; Verdadeiro Mas; Impreciso; Descontextualizado; Manipulado; Falso; Pimenta na Língua
Prova dos Factos[30]	✓	✓	✓	Discurso/Declarações	Português de Portugal	Verdadeiro; Parcialmente Verdadeiro; Parcialmente Falso; Falso
Truth or Fiction[31]	✓			E-mail	Inglês	Verdadeiro; Ficção (Falso)
Chequeado[32]	✓	✓	✓	Várias fontes de dados	Espanhol	Verdadeiro; Enganador; Insustentável; Falso
Fact-Check Observador[33]	✓	✓	✓	Várias fontes de dados	Português de Portugal	Verdadeiro; Praticamente verdadeiro; Inclusivo; Esticado; Enganador; Falso
Truco[34]	✓	✓		Discurso, informações na internet e redes sociais	Português do Brasil	Verdadeiro; Sem contexto; Discutível; Exagerado; Subestimado; Impossível Provar; Falso
Agência Lupa[35]	✓	✓	✓	Jornais, revistas, rádios, *sites*, redes sociais	Português do Brasil	Verdadeiro; Verdadeiro, Mas; Ainda é cedo para dizer; Exagerado; Contraditório; Subestimado; Insustentável; Falso; De Olho

[24]https://www.factcheck.org/
[25] https://www.snopes.com/fact-check/
[26] https://www.politifact.com/
[27] http://www.emergent.info/
[28] https://www.gossipcop.com/page/3/
[29] https://poligrafo.sapo.pt/
[30] https://www.publico.pt/prova-dos-factos
[31] https://www.truthorfiction.com/
[32] https://chequeado.com/metodo/
[33] https://observador.pt/seccao/observador/fact-check/
[34] https://apublica.org/checagem/
[35] https://piaui.folha.uol.com.br/lupa/tag/fact-checking/

Da análise da Tabela 2 podemos concluir que os dados que constam na maior parte destas ferramentas manuais provêm de diversas fontes de informação. O tema mais abordado é a política e a maior parte destas ferramentas estão mais direcionadas para o conteúdo noticioso em inglês. De todos os *websites* de verificação de factos os que analisam notícias em português europeu são o Polígrafo, o Fact-Check do Observador e a Prova dos Factos, sendo que a única diferença entre eles é a escala de credibilidade que utilizam para classificar as notícias. Numa análise comparativa, o Polígrafo usa uma escala de sete níveis de credibilidade, o Fact-Check utiliza seis níveis e, por fim, a Prova dos Factos utiliza uma escala de quatro níveis.

2.5 Agregadores de Notícias

Nos últimos anos, o crescimento da Internet e a transmissão de dados *online* é cada vez mais rápida, levando assim a que o jornalismo se tornasse cada vez mais digitalizado. Esta digitalização proporcionou uma nova forma de consumir a informação, fazendo com que os jornais em papel se tornassem menos adequados e mais desinteressantes. Desta forma, e com o objetivo de armazenar as notícias que eram divulgadas *online* ao minuto, começaram a ser desenvolvidos vários agregadores de notícias. Nestes agregadores encontram-se dispostos os jornais de notícias *online* que contêm os seus próprios agregadores, nomeadamente Feed RSS e Newsletter, mas também existem outros, que contam com API's que reúnem uma maior quantidade de informação de diversas fontes de informação.

2.5.1 Feed RSS e Newsletter

A maior parte dos *sites* de notícias em língua portuguesa permitem a obtenção de notícias de forma atualizada e diária de duas formas distintas: através de um Feed RSS ou através da receção de uma Newsletter. O Feed RSS é uma tecnologia de acesso às notícias atualizadas, desenvolvido em linguagem XML (Zheng and Zhang, 2012). Este é um formato de dados textual utilizado para especificar regras, para codificar documentos e informações utilizando *tags*, elementos e atributos na forma semiestruturada (Finkelstein, 2005).

Na Tabela 3, mostram-se os elementos principais (título, descrição, autor, URL, *guid*, data de publicação e a categoria) do Feed RSS e as suas descrições.

Tabela 3 Elementos e descrição do Feed RSS

Elementos	Descrição
<title>	Título da notícia
<description>	Descrição sucinta da notícia
<author>	Autor da notícia
<link>	Url de todo o conteúdo do site original
<guid>	Identificador único universal
<pubDate>	Data da publicação do item
<category>	Categoria da notícia atribuída pelo autor

A Figura 25 ilustra os elementos abordados na tabela anterior relativamente ao Feed RSS da fonte de notícias Correio da Manhã.

```
<item>
▼<title>
   <![CDATA[ Parlamento aprova renovação do Estado de Emergência até dia 14 de fevereiro ]]>
  </title>
  <link>https://www.cmjornal.pt/multimedia/videos/detalhe/parlamento-aprova-renovacao-do-estado-de-emergencia-ate-dia-14-de-fevereiro</link>
  <guid>https://www.cmjornal.pt/multimedia/videos/detalhe/parlamento-aprova-renovacao-do-estado-de-emergencia-ate-dia-14-de-fevereiro</guid>
▼<description>
   <![CDATA[ Atual estado de emergência, o nono aprovado, termina já no próximo sábado às 23h59. ]]>
  </description>
▼<author>
   <![CDATA[ portal@cmjornal.pt (Correio da Manhã) ]]>
  </author>
▼<category>
   <![CDATA[ Vídeos ]]>
  </category>
  <enclosure type="image/jpeg" url="https://cdn.cmjornal.pt/images/2021-01/img_100x100$2021_01_29_12_15_47_1009336.jpg" length="0"/>
  <pubDate>Fri, 29 Jan 2021 12:20:36 +0000</pubDate>
</item>
```

Figura 25 Exemplo de uma Feed RSS do Correio da Manhã

Como se pode verificar pela Figura 25, o título da notícia indica que a mesma é sobre a aprovação do Estado de Emergência, até dia 14 de Fevereiro, feita pelo Parlamento. A descrição da notícia é sobre o atual estado de emergência e o autor desta notícia não se encontra identificado. Esta notícia foi inserida na categoria "Vídeos" e foi publicada no dia 29 de Janeiro de 2021, ao meio-dia e vinte.

Em relação ao outro agregador, designado NewsLetter, é uma subscrição que qualquer pessoa pode fazer na página do mesmo de forma gratuita. Este agregador permite que as pessoas recebam todos os dias um *e-mail* com as notícias da atualidade. Na Tabela 4, apresenta-se uma lista de *websites* de notícias portuguesas, assim como o formato dos dados.

Tabela 4 Sites de notícias, Feed RSS e Newsletter

Sites de Notícias	Feed RSS	Newsletter	Formatos dos Dados
Público	✓	✓	XML
TSF	✓	✓	XML
SIC Notícias	✓		XML
Correio da manhã (cm)	✓		XML
Jornal de Notícias	✓		XML
Diário de Notícias		✓	XML
RTP	✓	✓	XML
Observador	✓	✓	XML
Sábado	✓		XML
Notícias ao Minuto	✓	✓	XML
Record	✓		XML

Podemos concluir que todos os *sites* de notícias, exceto o Diário de Notícias, contêm Feed RSS e o formato dos seus dados é semiestruturado, ou seja, em XML. Por outro lado, apenas seis *websites*, de um total de onze, disponibilizam o serviço de envio de *Newsletters*.

2.5.2 API's

Conforme foi descrito anteriormente, a crescente evolução do jornalismo e a utilização de plataformas *online* para propagar as mesmas, fez com que alguns jornais oferecessem API's (*Application Programming Interface*) para fornecerem acesso ao seu conteúdo noticioso. Assim, destacam-se cinco API's relevantes:

- O Google News API (Ng, 2019) nasceu em 2002, é um serviço desenvolvido pela Google que agrega um conjunto de notícias em várias línguas.
- O New York Times API (Ng, 2019) é um jornal americano prestigiado, mas também é uma fonte segura para encontrar notícias, multimédia, opiniões e outras informações.
- O Bing News API (Ng, 2019) é um serviço de agregação de notícias da Microsoft que exibe notícias mais recentes na *web*.
- The Guardian API (Ng, 2019) é um jornal conhecido por oferecer notícias atualizadas, informações de desporto, comentários e opiniões de todo o mundo.
- Arquivo.pt[36] é uma infraestrutura de investigação que permite pesquisar e aceder às páginas da *web* que foram arquivadas desde 1996 até à data atual.

Na Tabela 5 apresenta-se uma síntese comparativa das API's acima referidas tendo por base os seguintes critérios: preço, formato de dados e notícias em português.

Tabela 5 APIs de notícias e caraterísticas (Ng, 2019)

Caraterísticas / API's	Preço	Formato dos Dados	Notícias em Português
Google News API	Gratuito	JSON	Yes
News York Times API	Gratuito	JSON	-
Bing News API		JSON	-
The Guardian API	Gratuito	JSON	-
Arquivo.pt	Gratuito	JSON	Yes

Tal como se pode observar na tabela anterior, as API's que contêm notícias de língua portuguesa são: Google News API e Arquivo.pt. Estas encontram-se divididas por subcategorias como: "*business*"; "entertainment"; "*health*"; "*science*"; "*sports*" e "*technology*".

Neste contexto uma notícia é caracterizada tendo em conta o seu conteúdo físico e não físico. Por conteúdo físico entende-se o conjunto de parâmetros da notícia, nomeadamente o título, o corpo principal, as imagens, os vídeos, etc. (Zhang and Ghorbani, 2020). Por sua vez, o conteúdo não físico está relacionado com o objetivo subjacente à notícia, assim como os sentimentos transmitidos pela mesma.

[36] https://arquivo.pt/image/search?hitsPerPage=10&query=covid&l=pt&ion-dt-0=1996-01-01&ion-dt-1=2021-01-30&dateStart=01%2F01%2F1996&dateEnd=30%2F01%2F2021

Atualmente, e tal como foi descrito anteriormente, o conteúdo físico encontra-se em plataformas *online* disponíveis através de API's e Feed RSS. Neste caso, cada notícia tem o seu *Uniform Resource Locator* (URL) para uma determinada página *web*, imagem ou até vídeo. Pelo que, estes são alguns dos parâmetros que se devem considerar na análise de notícias.

Com base nos agregadores mencionados anteriormente sistematizou-se, na Tabela 6, os parâmetros que se podem extrair e uma descrição sucinta dos mesmos.

Tabela 6 Parâmetros pedidos e descrição

Parâmetros Pedidos	**Descrição dos Parâmetros**
Autor	Autor da notícia
URL	Fonte da notícia
Guid	Identificador único universal
Descrição	Descrição sucinta da notícia
Categoria	Categoria da notícia (exemplo: fashion, desporto)
Data de Publicação	Data de publicação da notícia
Título	Título da notícia
Conteúdo	Conteúdo de toda a notícia
Url da Imagem	Fonte da imagem
FeedBurner	Fonte original da notícia

Com base na tabela anterior, apresenta-se uma nova tabela, Tabela 7, que identifica os parâmetros que se podem extrair dos agregadores de notícias abordados anteriormente (API's e Feed RSS).

Tabela 7 Parâmetros presentes nos agregadores de notícias

			Parâmetros Pedidos									
			Autor	URL	Guid	Descrição	Categoria	Data de Publicação	Título	Conteúdo	Url da Imagem	FeedBurner
Agregadores de Notícias	**APIs**	Google News API	✓	✓		✓		✓	✓	✓	✓	
		News York Times API	✓	✓	✓	✓	✓	✓	✓			
		Bing News API		✓		✓	✓				✓	
		Guardian API		✓			✓	✓	✓			
		Arquivo.pt		✓				✓	✓			✓
	Feed RSS	Correio da Manhã(CM)	✓	✓	✓	✓	✓	✓	✓		✓	
		Jornal Notícias (JN)		✓		✓	✓	✓	✓			✓
		Público	✓	✓	✓	✓	✓	✓	✓			
		Sábado	✓	✓	✓	✓	✓	✓	✓		✓	
		Record	✓	✓	✓	✓	✓	✓	✓		✓	
		Observador	✓	✓	✓	✓	✓	✓	✓		✓	
		TSF	✓	✓	✓	✓	✓	✓	✓			✓

Relativamente à tabela anterior podemos verificar que nas API's os parâmetros que podem ser extraídos são: URL, descrição, categoria, data de publicação, título e autor. No Feed RSS, apenas são disponibilizados os seguintes parâmetros: autor, URL, identificador único, descrição, categoria, data de publicação e título. Note-se que o mesmo não disponibiliza o conteúdo da notícia o que se torna

numa desvantagem relevante dos Feed RSS. No que diz respeito às API's, o Google News API's e o News York Times API são os que permitem extrair mais informações. Em relação, aos Feed RSS analisados são o Correio da Manhã, o Público, a Sábado, o Observador, a TSF e o Record que contêm mais parâmetros para ser extraídos.

2.6 Sumário

Neste capítulo apresentou-se uma contextualização sobre o tema de notícias falsas, a sua categorização e os motivos que possibilitam a criação de notícias falsas na perspetiva da autora Wardle (Wardle, 2018), bem como a exposição de diferentes ações que podem ser tomadas por qualquer pessoa para identificar se uma notícia é verdadeira/falsa e os *websites* especializados na área de verificação da veracidade de uma notícia. Estes *websites* são responsáveis pela verificação de factos desenvolvidos por jornalistas ou pessoas especializadas na área, de seguida procedeu-se a uma análise comparativa dos mesmos. Por último, analisaram-se os diferentes agregadores de notícias, nomeadamente, Feed RSS e API's.

Em suma, o leitor deve ter uma visão mais detalhada do conceito, das ações que pode tomar para identificar se uma notícia é verdadeira ou falsa, os diferentes *websites* existentes de verificação de factos manuais e, por último, os diferentes agregadores de notícias presentes em Portugal e noutros países. Estes agregadores de notícias foram investigados com o propósito de perceber que parâmetros podem ser extraídos e em que formato se encontram. Estes dados são relevantes para posteriormente se proceder à construção do *dataset* pretendido na fase de implementação da solução.

3 Estado da Arte

Com o aparecimento de um grande número de notícias falsas na Internet, o ser humano deixa de ter capacidade de resposta para verificá-las de forma manual e intuitiva, o que torna o método de verificação de factos impraticável. Neste contexto, surge a necessidade de se desenvolverem sistemas automáticos que sejam capazes de dar resposta ao grande volume de dados, que é agora mais necessário de se analisar que nunca. A maioria destes sistemas desenvolvidos concentra-se no discurso utilizado pelos autores das notícias falsas.

Tendo em conta que para a deteção das notícias falsas é importante compreender o texto, assim torna-se relevante abordar o processo de *text mining*. Neste é utilizado o processamento de linguagem natural (PLN) na fase de pré-processamento com o propósito de transformar os dados não estruturados em dados estruturados adequados para numa fase posterior se aplicarem os algoritmos de aprendizagem automática (Kao and Poteet, 2007). Feldman (Feldman, 1999) considera que o processamento de linguagem natural pressupõe que "[...] se os seres humanos conseguem definir padrões e descrevê-los a um computador, então a máquina consegue aprender sobre a forma como falamos e nos entendemos uns com os outros". Desta forma, o significado de um texto exibe sete formas: fonético, morfológico, lexical, sintático, semântico, pragmático e de discurso (Feldman, 1999). Esta área pode ser utilizada em vários casos como por exemplo, na classificação do texto em categorias, tradução automática, compreensão da linguagem, extração da informação, aquisição de conhecimento, entre outras.

Neste contexto, o presente capítulo encontra-se dividido em cinco secções. Na Secção 3.1 é introduzido o conceito teórico de *text mining* e as etapas do seu processo. Na Secção 3.2 são descritos os diferentes algoritmos de aprendizagem automática relativos à classificação de texto. Na Secção 3.3, apresentam-se os projetos relacionados sobre a deteção de notícias falsas tendo em conta o processo de *text mining.* Na Secção 3.4 são consideradas as tecnologias existentes para o desenvolvimento da solução e por último, na Secção 3.5 é feito um resumo de todo o capítulo.

3.1 Text Mining

O *Text mining* é um processo de extração de conhecimento proveniente do texto semiestruturado (como por exemplo XML e JSON) e não estruturado (documentos Word, vídeos e imagens) (Feldman

and Dagan, 1995). A Figura 26 representa todas as fases deste conceito, sendo estas: o pré-processamento; a transformação do texto; a seleção de caraterísticas; os métodos de *text mining* e por último, a avaliação dos resultados.

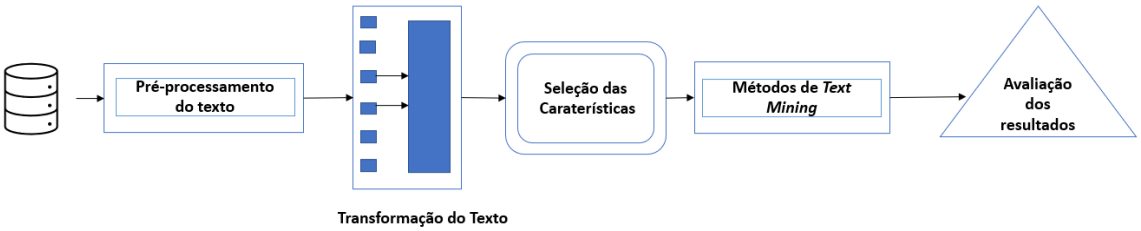


Figura 26 Fases do *Text Mining*

3.1.1 Pré-processamento

O pré-processamento no *text mining* permite a extração do conhecimento a partir de dados não estruturados. Nesta etapa, o texto é composto por um conjunto de elementos (parágrafos, palavras, pontuações, pronomes, espaços, determinantes, verbos, entre outros) que não são de todo relevantes para uma análise mais aprofundada. Neste sentido, torna-se fundamental efetuar o processamento do mesmo, para isso aplica-se um conjunto de técnicas, nomeadamente, tokenização, *stop-words removal, lemmatization* e *stemming* (Mohan, 2015) (Allahyari *et al.*, 2017).

Tendo em conta as técnicas referidas anteriormente, comecemos por descrever a técnica de tokenização do texto. Nesta o texto é divido em pequenas partes denominadas de *tokens*, seguida da remoção dos espaços e pontuações que se encontram presentes no mesmo (Verma, Renu and Gaur, 2014). Na Figura 27, apresenta-se um exemplo desta técnica na frase "Adoramos passar aqui as férias de Verão.".

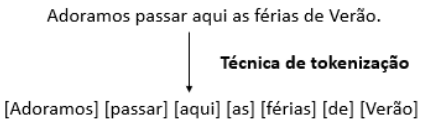


Figura 27 Técnica de tokenização

Conforme podemos ver a frase inicial era: "Adoramos passar aqui as férias de Verão.". Aplicando a técnica de tokenização com remoção de sinais de pontuação, as palavras da frase ficaram agrupadas em sete *tokens* - [Adoramos] [passar] [aqui] [as] [férias] [de] [Verão].

Por outro lado, a técnica de *stop-words removal* é responsável por remover as palavras que se encontram presentes no texto e que não acrescentam qualquer valor ao conteúdo do mesmo (pronomes, preposições, determinantes, conjunções), como é o caso das palavras que aparecem recorrentemente ou raramente (Müller and Guido, 2016). Esta técnica é capaz de reduzir os dados do texto e melhorar o desempenho do mesmo (Kannan and Gurusamy, 2014). Na Figura 28, apresenta-se um exemplo onde a técnica de *stop-words removal* é utilizada.

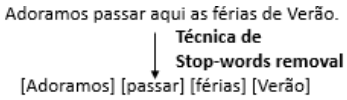


Figura 28 Técnica de stop-words removal

Por último, as técnicas de *lemmatization* e *stemming* são responsáveis pela normalização das palavras na área de processamento de linguagem natural. Jabeen (Jabeen, 2018) descreve que estas são utilizadas para preparar o texto, as palavras e os documentos para um processamento posterior. Ambas pretendem reduzir a palavra à sua raiz, porém têm abordagens distintas.

Lemmatization é uma técnica que analisa a morfologia das palavras, ou seja, identifica a palavra na sua forma canónica. Esta é aplicada nos verbos colocando os mesmos no infinitivo e os substantivos/adjetivos são dispostos no modo masculino e singular (Balakrishnan and Ethel, 2014).

Stemming é a técnica mais simples das duas abordagens, mas encontra-se sempre limitada no corte da palavra em *stem*[37] (Heidenreich, 2018) (Müller and Guido, 2016). Nesta técnica são removidos os sufixos ou prefixos de uma determinada palavra e consequentemente esta é reduzida ao seu radical (Cambridge University Press, 2008) (Lovins, 1968). No entanto, esta técnica sofre com dois problemas: *overstemming* [38] e *understemming* [39] *,* quando são retirados os sufixos/prefixos e as palavras resultantes não constam no dicionário. Atualmente, existem três bibliotecas disponíveis para esta técnica: *Porter Stemmer*, *Snowball Stemmer* e o *Lancaster Stemmer* (Heidenreich, 2018).

3.1.2 Transformação do Texto

No *text mining*, a transformação do texto é um processo bastante complexo, pelo que é necessário utilizar diferentes técnicas capazes de estruturar os dados textuais, em dados capazes de serem utilizados para uma análise mais profunda dos documentos. Sendo que os algoritmos só tem a capacidade de utilizar dados numéricos como entrada para os mesmos (Allahyari et al., 2017).

Uma técnica muito conhecida para representar um documento denomina-se *Bag-of-Words*[40] (BoW). Nesta, considera-se o número de vezes que cada palavra/frase aparece, sem ter em conta a ordem onde a mesma se apresenta no texto (Lee, Song and Kim, 2010). Desta forma, esta representação proporciona uma representação vetorial que pode ser analisada através dos vários métodos de redução de dimensão (*Latent Semantic Indexing* (*LSI*), *Probabilistic Latent Semantic Indexing* (*PLSA*) e Modelos de Tópicos) que têm como base um conjunto de algoritmos de aprendizagem automática e modelos estatísticos (Allahyari et al., 2017).

Por outro lado, na Recuperação de Informação (RI), os documentos precisam de ser classificados para se obter uma recuperação mais eficaz. Com o propósito de definir o peso que cada palavra tem no documento, os mesmos são representados como vetores, sendo que cada palavra descrita no texto é

[37] Palavra núcleo
[38] Corte exagerado da palavra
[39] Corte ligeiro da palavra
[40] Saco de palavras

representada por um valor numérico. Desta forma, são aplicados três modelos nomeadamente: o *Vector Space Model* (VSM), os modelos probabilísticos e, por último, *Inference Network Model* (INM) (Allahyari *et al.*, 2017).

O VSM é considerado como um dos melhores métodos para transformar um documento num vetor numérico (Li-Ping Jing, Hou-Kuan Huang, and Hong-Bo Shi, 2002) (Allahyari *et al.*, 2017). Neste, os documentos são representados como vetores e o seu sucesso tem como base a ponderação de cada termo (palavras, frases, ou outras unidades usadas para identificar o conteúdo do texto) (Premalatha and Srinivasan, 2014).

Esta ponderação é um aspeto importante a considerar nos sistemas modernos de recuperação de texto, sendo que para a atribuição do peso existem dois modelos capazes de o fazer, nomeadamente o Modelo Booleano e o *Term Frequency-Inverse Document Frequency* (TF-IDF) (Allahyari *et al.*, 2017).

Nestes modelos são usadas as seguintes variáveis:

- *D* representa o conjunto dos documentos, que é dada por $D = \{d_1, d_2, d_3, \dots, d_D\}$;
- *V* representa o vocabulário que é composto por um conjunto de palavras ou termos diferentes no conjunto dos documentos, que é dada por $V = \{w_1, w_2, w_3, \dots, w_v\}$;
- Frequência de uma palavra $w \in V$ no documento $d \in D$ é representado por $f_d(w)$;
- Número de documentos com a palavra w é representada por $f_D(w)$.

Posta esta representação, no Modelo Booleano, cada palavra que se encontre presente no documento tem atribuído um peso. Para uma palavra w_i, num documento d_j representa-se este peso por ω_{ij}. No caso de a palavra aparecer uma ou mais vezes no documento é atribuído um valor superior a zero *($\omega_{ij} > 0$)*, se a palavra que não aparecer, o peso que lhe é atribuído corresponde a zero ($\omega_{ij} = 0$).

Por outro lado, no modelo TF-IDF, representa-se por $q(w)$ o peso da palavra $w \in d$ e é dado pela seguinte equação:

$$q(w) = f_d(w) * \log \frac{|D|}{f_D(w)} \quad (1)$$

Em relação ao TF-IDF, a frequência da palavra é normalizada através da Frequência do Documento Inverso (FDI). Esta normalização leva à diminuição do peso das palavras mais frequentes que se encontram no conjunto dos documentos. Com base no esquema de ponderação dos termos, cada documento é representado por um vetor de pesos de palavras, em que *$\omega(d) = (\omega(d, w_1), \omega(d, w_2), \dots, \omega(d, w_v)$* (Allahyari et al., 2017). Quando se pretende verificar a similaridade de dois documentos d_1 e d_2, pode utilizar-se o modelo de *Cosine Similarity*, que é dado pela seguinte equação:

$$S(d_1, d_2) = \cos\theta = \frac{d_1 * d_2}{\sqrt{\sum_{i=1}^{v} w_{2i}^2} * \sqrt{\sum_{i=1}^{v} w_{1i}^2}} \quad (2)$$

3.1.3 Seleção de Caraterísticas

Os modelos não funcionam bem com uma elevada quantidade de atributos/caraterísticas (Beniwal and Arora, 2012). Assim, torna-se necessário remover todos os que não são relevantes para a entrada no modelo. Ora, para se resolver este problema aplicam-se diferentes técnicas de seleção de caraterísticas, tendo como principais objetivos: evitar o *overfitting*[41], melhorar a *performance* [42] dos modelos e proporcionar modelos mais rápidos e económicos (Beniwal and Arora, 2012). No contexto da classificação, as técnicas de seleção de caraterísticas são agrupadas em três grupos, filtragem, *wrapper* e *embedded.* No entanto, estas são diferentes umas das outras na forma como procedem à seleção das caraterísticas para o modelo, conforme ilustra a Figura 29 (Zheng and Casari, 2018) (Kuhn and Johnson, 2019).

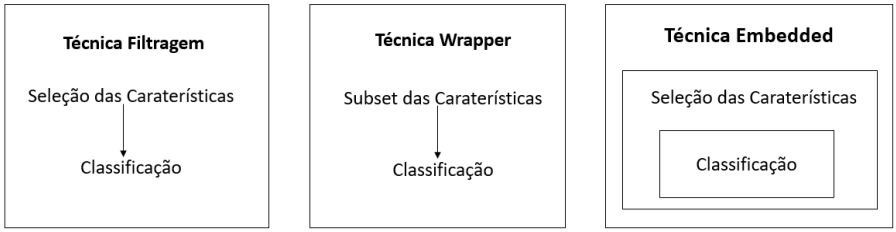


Figura 29 Técnicas de seleção de caraterísticas

3.1.3.1 Técnica de Filtragem

A Técnica de Filtragem seleciona um subconjunto de caraterísticas independentemente do método de aprendizagem que vai ser aplicado sobre o conjunto de exemplos representados, recorrendo a um subconjunto de caraterísticas selecionadas (Saunders and Grobelnik, 2005). Estas utilizam métodos estatísticos tais como: o coeficiente de correlação e a informação mútua (Zheng and Casari, 2018).

Esta técnica apresenta como vantagens o facto de ser computacionalmente simples e rápida, bem como serem independente do algoritmo de classificação. Desta forma, a seleção das caraterísticas apenas é feita uma única vez e de seguida os diferentes classificadores podem ser avaliados. Como desvantagem foi verificada a falta de interação com o classificador, isto leva a que cada recurso seja considerado como um só, ignorando as dependências existentes com os outros, assim o desempenho do algoritmo de classificação pode obter resultados pouco fiáveis (Saeys, Inza and Larrañaga, 2007).

3.1.3.2 Técnica Wrapper

A Técnica *Wrapper* consiste em selecionar um subconjunto de caraterísticas recorrendo a uma função de avaliação com base no algoritmo de aprendizagem que vai ser utilizado posteriormente, assim as caraterísticas não podem simplesmente ser eliminadas sem existir uma determinada razão (Saunders and Grobelnik, 2005) (Saeys, Inza and Larrañaga, 2007).

O modelo nesta técnica é tratado como se fosse uma "caixa negra" e a sua *performance* é vista como uma função de custo para avaliar os subconjuntos (Zheng and Casari, 2018). Esta avaliação é obtida com o treino e teste de um determinado modelo de classificação.

[41] Sob reajustado
[42] Desempenho

Uma vantagem desta técnica diz respeito à interação entre a pesquisa do subconjunto de caraterísticas e a seleção do modelo, mas também à capacidade de levar em consideração as dependências dos recursos. Como desvantagens, apresenta um maior risco no *overfitting* dos dados, em termos computacionais é mais lenta e, por último, a construção do classificador requer um alto custo computacional (Saeys, Inza and Larrañaga, 2007).

3.1.3.3 Técnica Embedded

Na Técnica *Embedded*, a seleção de caraterísticas e a aprendizagem são dependentes entre si. Um exemplo desta é mesmo a indução das Árvores de Decisão (AD) já que em cada etapa de treino do modelo executa-se recorrentemente a seleção das caraterísticas (Saunders and Grobelnik, 2005).

Tal como as abordagens *Wrapper*, as técnicas *Embedded* são específicas para um determinado algoritmo de aprendizagem. No entanto, as últimas apresentam como vantagens a interação com o modelo de classificação, mas também o facto de terem melhores custos computacionais do que os métodos *Wrapper* (Saeys, Inza and Larrañaga, 2007).

3.1.3.4 Comparação das Técnicas

Face ao exposto, a Tabela 8 foi construída com o propósito de se comparar as diferentes técnicas tendo em conta as seguintes caraterísticas: em que consiste a técnica, velocidade computacional, *overfitting*, desempenho e os métodos que são aplicados em cada técnica de seleção de caraterísticas.

Tabela 8 Comparação das técnicas de Filtragem, *Wrapper* e *Embedded* (Bhat, 2019)

Técnicas / **Caraterísticas**	**Filtragem**	***Wrapper***	***Embedded***
Abordagem	Seleção de caraterísticas independentemente do algoritmo de classificação	Organização de um subconjunto de caraterísticas, consoante a função de avaliação presente no modelo	Integração de caraterísticas na fase de construção do modelo
Velocidade Computacional	Rápido	Muito lento	Médio
Overfitting	Pouco Propício	Propício	Pouco Propício
Desempenho	Mau	Excelente	Bom
Métodos	*Correlation, Chi-Square Test, Information Gain*	*Forward Selection, Backward elimination, Stepwise selection*	*LASSO, Ridge Regression, Elastic Net*

Em suma, quando existe um elevado número de caraterísticas, a Técnica de Filtragem a nível computacional é mais rápida e útil; a Técnica *Wrapper* apresenta um melhor desempenho, mas a nível de velocidade computacional é muito lenta e, por último, a *Embedded* tem um bom desempenho, mas em termos de velocidade computacional apresenta uma *performance* mediana.

3.1.4 Métodos de Text Mining

Os diferentes métodos de *text mining* podem ser divididos em quatro grupos principais, nomeadamente: classificação, agrupamento, *topic modeling* e sumarização (Gohil, 2015).

O método de classificação insere-se na aprendizagem supervisionada. Nesta, o classificador aprende tendo em conta um conjunto de dados previamente etiquetados, pelo que os dados de saída devem ser semelhantes aos dados de entrada (Sukanya and Biruntha, 2012). Allahyari et al. (Allahyari *et al.*, 2017) apresentam quatro algoritmos que permitem classificar o texto, designadamente *Naive Bayes Classifier*, *Nearest Neighbours Classifier,* Árvores de Decisão de Classificação e Máquinas de Vetores de Suporte. As métricas utilizadas para avaliação e classificação dos resultados obtidos pelos algoritmos são a *precision*, o *recall* e o *f1-score*.

O método de agrupamento enquadra-se numa aprendizagem não supervisionada. Neste, os documentos são agrupados de acordo com as semelhanças entre si através de *clusters* (Sukanya and Biruntha, 2012). Este é muitas vezes utilizado na classificação, visualização e organização dos documentos. Os algoritmos de agrupamento que se utilizam para a classificação do texto são *Hierachical Clustering*, *K-Means* e *Probabilistic Clustering and Topic* (Allahyari *et al.*, 2017).

O *topic modeling* é um dos algoritmos de aprendizagem não supervisionada que trata os documentos como se fossem tópicos e para cada tópico é feita uma distribuição de probabilidade sobre as palavras presentes no mesmo. Com este algoritmo pretende-se criar um modelo genérico para o corpo de um determinado documento (Allahyari *et al.*, 2017).

O método de sumarização do texto tem como objetivo reduzir o tamanho e os pormenores do documento mantendo sempre presente o significado do mesmo. Este método é utilizado para resumir o grande volume de documentos presentes nas organizações de investigação (VijayGaikwad, Chaugule and Patil, 2014). Os métodos de sumarização de texto são classificados em sumarização extrativa e abstrativa. A sumarização extrativa consiste em selecionar as frases importantes, parágrafos, entre outros, do documento original. A importância das frases é decidida com base nas caraterísticas estatísticas e linguísticas. Por outro lado, na sumarização abstrativa tenta-se desenvolver uma compreensão dos conceitos principais num determinado documento e depois expressar esses conceitos na linguagem natural. Para isso usam-se métodos linguísticos para examinar e interpretar o texto e posteriormente encontrar os novos conceitos e expressões para melhor o descrever, criando um novo texto mais curto que transmite a informação mais importante do texto original (Patel and Soni, 2012).

3.1.5 Avaliação dos Resultados

Por último, torna-se importante fazer uma avaliação e interpretação dos resultados obtidos dos modelos mencionados anteriormente, tendo em conta as seguintes métricas: *accuracy, recall,* precisão e *f1-measure.* Estas métricas são obtidas a partir dos quatro resultados que se apresentam naquilo que é chamado de matriz confusão (ilustrada na Tabela 9). Esta matriz permite avaliar a *performance* de um determinado algoritmo de aprendizagem automática.

Tabela 9 Matriz confusão (Narkhede, 2021)

		Classe Prevista	
		Classe Verdadeira	**Classe Falsa**
Classe Real	**Classe Verdadeira**	Positivos Verdadeiros (TP)	Falsos Negativos (FN)
	Classe Falsa	Falsos Positivos (FP)	Negativos Verdadeiros (TN)

O autor Narkhede (Narkhede, 2021) descreve da seguinte forma as quatro entradas da matriz confusão:

- "Positivos Verdadeiros (TP) – número de casos positivos que foram classificados de forma correta";
- "Falsos Positivos (FP) – número de casos negativos que foram incorretamente classificados como positivos";
- "Negativos Verdadeiros (TN) – número de casos negativos que foram classificados de forma correta";
- "Falsos Negativos (FN) – número de casos positivos que foram incorretamente classificados como negativos".

3.1.5.1 Accuracy

A *accuracy* representa a taxa de acerto de todo o classificador, ou seja, a razão entre a soma dos acertos das classes a prever e o número total de instâncias (Shung, 2020). Esta é representada pelo número de Positivos Verdadeiros (TP) mais o número de Negativos Verdadeiros (TN) sobre o número de Positivos Verdadeiros mais os Negativos Verdadeiros (TN), Falsos Positivos (FP) e Falsos Negativos (FN) conforme representado na equação:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{3}$$

3.1.5.2 Recall

O *recall* representa o número total de classificações positivas fora da classe verdadeira. O *recall* é o coeficiente entre número de Verdadeiros Positivos (TP) e a soma entre o número de Verdadeiros Positivos (TP) e o número de Falsos Negativos (FN) (Shung, 2020).

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4}$$

3.1.5.3 Precisão

A precisão é uma medida da exatidão do modelo, caso esta seja alta significa que o classificador é considerado bom. A Equação (5) define a precisão como sendo o coeficiente entre o número de Verdadeiros Positivos (TP) sobre o número de Verdadeiros Positivos (TP) mais o número de Falsos Positivos (FP) (Shung, 2020).

$$Precision = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{5}$$

3.1.5.4 F1-Measure

A pontuação *f1-measure* é uma métrica que leva em consideração a *precision* e *recall*, e definida pela Equação (6):

$$f1\text{-}measure = 2 * \frac{(\mathrm{Precision} * \mathrm{Recall})}{(\mathrm{Precision} + \mathrm{Recall})} \tag{6}$$

Esta pode ser usada para obter o equilíbrio entre as métricas de *precision* e de *recall* ao mesmo tempo (Shung, 2020).

3.1.6 Aplicações do Text Mining

Hoje em dia, a área de *text mining* encontra-se aplicada em diversos domínios. Existem seis sectores principais nos quais a mesma é aplicada, sendo os mesmos: a publicação e meios de comunicação social; os bancos, os seguros e os mercados financeiros; as telecomunicações, a energia e outros serviços; a tecnologia da informação e a internet; as empresas farmacêuticas, de investigação e os cuidados de saúde e, por último, as instituições políticas e judiciais, analistas políticos e a administração pública (Kacprzyk, 2005) (Gupta and Lehal, 2009).

Tendo em conta os sectores descritos é possível identificar algumas especificações sectoriais com recurso ao *text mining*. No caso do sector da publicação e meios de comunicação social, aplica-se o processo de extração, transformação e carregamento nos catálogos, na produção e na otimização no processo de recuperação da informação.

No sector dos bancos e dos seguros é usado com o propósito de melhorar a gestão da comunicação com os clientes através de sistemas automáticos de reencaminhamento de mensagens e com aplicações de apoio aos motores de busca que colocam questões na linguagem por definição.

No sector da saúde utilizam-se aplicações do tipo *Competitive Intelligence and Technology Watch* para analisar, classificar e extrair a informação dos artigos, sejam estes resumos científicos ou patentes.

Por último, no sector das telecomunicações, este método é determinante para a obtenção de respostas desde a análise do mercado até à gestão dos recursos humanos, mas também desde a correção ortográfica até ao inquérito a um determinado cliente (Gupta and Lehal, 2009).

3.2 Algoritmos de Aprendizagem Automática

O autor Ayodele (Ayodele, 2010) identifica os algoritmos de aprendizagem supervisionada de modo a resolver problemas de classificação. Sendo que os algoritmos que melhor se adequam à classificação são: *Linear Classifiers*, Regressão Logística, *Naive Bayes*, *Perceptron*, Máquinas de Vetores de Suporte, *K-Means Clustering*, *Boosting*, Árvores de Decisão, *Random Forest* e Redes Neuronais (Akinsola, 2017).

De seguida, são apresentados alguns destes algoritmos de classificação e a descrição de uma técnica para avaliar a capacidade de robustez de um determinado modelo.

3.2.1 Algoritmos de Classificação

- *Naive Bayes*

O algoritmo *Naive Bayes* é um algoritmo probabilístico de aprendizagem supervisionada que tem como base o Teorema de *Bayes*. É um algoritmo simples de aprender desde que sejam assumidas as caraterísticas como independentes de uma determinada classe. O objetivo principal deste algoritmo é compreender quais as caraterísticas que afetam o desempenho do mesmo. Comparado com outros algoritmos mais sofisticados de classificação, com este método obtém-se um melhor desempenho e alguns exemplos da sua aplicabilidade são: classificação do texto, diagnósticos médicos, recuperação de informação e até na gestão do desempenho dos sistemas (I Rish, 2001).

Para além disso, o mesmo pode lidar com um grande número de variáveis e grandes conjuntos de dados, e trata o atributo como variáveis contínuas (Webb and Yu, 2004). Para se lidar com as variáveis contínuas, este algoritmo conta com três métodos principais, nomeadamente: normal, *kernel* e discretização. O método normal é o mais conhecido de todos e aproxima-se da distribuição da variável contínua recorrendo ao uso de uma distribuição parametrizada denominada de *Gaussian*. O método *kernel* utiliza uma aproximação não parametrizada. Por último, discretização, primeiro discretizam as variáveis contínuas em variáveis discretas, deixando um problema mais simples sem qualquer variável. Destes três métodos descritos anteriormente, o que obtém piores resultados é o método normal (Webb and Yu, 2004).

- *K-Nearest Neighbors*

O algoritmo *K-Nearest Neighbors* é um algoritmo de aprendizagem supervisionada, conhecido por resolver problemas tanto de classificação como de regressão, e apresenta como vantagens o facto de ser bastante simples e fácil de implementar. Este é construído com base na distância Euclidiana entre as amostras de teste e treino. Porém, quando o número de dados de entrada aumentam, o algoritmo tendencionalmente fica mais lento (Peterson, 2009) (Harrison, 2019).

- Máquina de Vetores de Suporte

O autor Vapnik desenvolveu o algoritmo da Máquina de Vetores de Suporte (MVS) com o intuito de resolver problemas de classificação de duas classes (Webb and Yu, 2004). O algoritmo Máquina de Vetores de Suporte baseia-se na Teoria de Aprendizagem Estatística e é um algoritmo de aprendizagem supervisionada, tal como o *K-Nearest Neighbors*, capaz de resolver problemas tanto de classificação como de regressão. Além disso também é simples em termos de implementação e apresenta boas precisões com um baixo custo computacional (Hearst et al., 1998) (Gandhi, 2018b).

O objetivo deste é descobrir um hiperplano (considerado um limite de decisão que pretende ajudar a classificar os dados de entrada), num espaço de dimensão N – número de atributos dados como valores de entrada. A MVS utiliza propriedades geométricas para calcular exatamente a ótica que separa o hiperplano diretamente do seu conjunto de dados. Este algoritmo pode resolver eficientemente o problema de aprendizagem, com os pontos fortes de uma boa classificação (Webb and Yu, 2004).

- Regressão Linear

O algoritmo Regressão Linear é um algoritmo de aprendizagem supervisionada direcionado para resolver problemas de regressão, mas também pode ser usada para prever uma variável dependente contínua, tendo em conta outras variáveis independentes do conjunto de dados. Este algoritmo é simples de implementar, os seus resultados são fáceis de interpretar e é utilizado com o propósito de fazer previsões e descobrir a relação dessa previsão com as suas variáveis (Lever, Krzywinski and Altman, 2016).

- Regressão Logística

O algoritmo Regressão Logística é um algoritmo direcionado para resolver problemas de classificação, também é considerado um algoritmo de análise preditiva e foi construído com base na Probabilidade. Este algoritmo é muito similar ao de Regressão Linear, porém contém uma função de custo um pouco mais complexa. A análise deste algoritmo consiste numa técnica estatística que descreve a relação entre duas variáveis, nomeadamente a variável independente que pode ou não ser contínua e uma variável dependente com apenas dois valores possíveis (Tripepi et al., 2008).

Este algoritmo pode resolver tanto problemas binários como de multiclasses. Relativamente ao algoritmo binário, um exemplo claro da sua aplicabilidade é: determinar se um e-mail é *spam* ou não. Por outro lado, um exemplo da aplicabilidade do mesmo multiclasses é: quando se pretende saber em que classe um animal se enquadra, pois este pode ser a ovelha, um cão, um gato, entre outros (Lever, Krzywinski and Altman, 2016) (Pant, 2019).

- *Linear Support Vector Machine*

O algoritmo *Linear Support Vector Machine* foi construído com o propósito de efetuar uma classificação binária, atualmente é considerado um dos algoritmos mais utilizados para resolver este problema de classificação. O mesmo pode conter um elevado conjunto de dados que continua a ser rápido e simples na sua execução, porém em termos de eficiência computacional e de memória não tem uma *performance* tão boa comparado com outros algoritmos de classificação (Ladicky and Torr, 2011).

- Árvores de Decisão

O algoritmo Árvores de Decisão é também um algoritmo de aprendizagem supervisionada para resolver problemas de classificação e regressão. Este algoritmo expressa os vários processos de forma visual até alcançar as decisões tomadas e implementado para ajudar a tomar decisões (Gupta, 2017). As árvores de decisão de classificação podem ser construídas com base num conjunto de instâncias através de uma estratégia de dividir e conquistar. Se todas as instâncias tiverem a mesma classe, a árvore vai corresponder a uma folha rotulada com essa classe. Senão, opta por uma instância que

contenha uma saída diferente para que pelo menos duas das instâncias sejam divididas de acordo com o mesmo resultado (Quinlan, 1996).

- *Multinomial Naive Bayes*

O algoritmo *Multinomial Naive Bayes* é um dos algoritmos de *Naive Bayes* utilizado especificamente para resolver os problemas de classificação de documentos. Este algoritmo pode alcançar boas *performances* usando uma aprendizagem localmente ponderada (Webb and Yu, 2004). Um exemplo da aplicabilidade deste algoritmo: Um documento pode ser classificado tendo em conta uma das seguintes categorias: política, desporto, entre outros (Gandhi, 2018a).

- *Random Forest*

O algoritmo *Random Forest* é considerado como um algoritmo eficiente em termos de classificação e pode ser descrito como um conjunto de árvores de classificação estruturadas (Akar and Gungor, 2012). Este algoritmo combina árvores de previsões como se cada árvore dependesse dos valores de um vetor aleatório amostrado independentemente e com a mesma distribuição para todas as árvores da floresta (Breiman and Cutler, 2004). Breiman et al. (Breiman and Cutler, 2004) descrevem este algoritmo como rápido, robusto, apenas necessita de dois parâmetros de entrada e permite que o utilizador construa sempre as *N* árvores que pretende.

- *Stochastic Gradient Descent*

O algoritmo *Stochastic Gradient Descent* é um dos algoritmos mais utilizados, considerado como a base das Redes Neuronais. Este foi proposto pelo autor Friedman que descrevia o funcionamento deste algoritmo como uma recolha de amostras uniformes, sem substituir o conjunto de dados antes de estimar a expetativa. Este passo proporcionou o aumento e melhoramento da sua *performance* (Ridgeway, 1999). Em cada iteração do algoritmo, os dados são escolhidos de forma completamente aleatória (Srinivasan, 2019). Este algoritmo é adequado para um grande volume de dados e a nível computacionalmente é bastante rápido.

- *Gradient Boosting Classifier*

O algoritmo *Gradient Boosting* é um algoritmo de aprendizagem supervisionada capaz de resolver problemas de modelação preditiva de classificação e regressão. Este treina de forma gradual e sequencial os vários modelos e utiliza uma função denominada de "perda" que determina se os coeficientes do modelo são bons no *fitting*[43] dos dados (Singh, 2018).

- *Convolutional Neural Network*

Convolutional Neural Network é um algoritmo de *Deep Learning* composto por várias camadas diferentes nomeadamente, *Convolution*, *Batch Normalization*, *Activation* e *Pooling*. A camada *Batch Normalization* normaliza os *layers* de entrada tendo em conta a sua média e variância. A camada *Activation* é uma função não-linear em relação aos seus elementos. Por último, a camada *Pooling* aplica um tipo de agrupamento no *layer* que entrou como parâmetro de entrada (Lin, Zhao and Pan, 2017).

Um exemplo da aplicabilidade deste algoritmo é quando recebe por exemplo como dado de entrada uma imagem atribuindo a cada caraterística da mesma um peso, sendo capaz de diferenciá-las umas das outras (Saha, 2018). De acordo com o autor Lin et al. (Lin, Zhao and Pan, 2017), alcançaram bons

[43] Encaixar

resultados em várias experiências no mundo real, como no caso da classificação de imagens e deteção de objetos.

- *Recurrent Neural Network*

O algoritmo *Recurrent Neural Network* é um modelo de sequência neuronal que não utiliza um tamanho limitado no seu contexto, resolve tarefas que incluem a modelação da linguagem, reconhecimento da falsa e a tradução automática (Zaremba, Sutskever and Vinyals, 2015). Este apresenta sobretudo bons resultados a fazer previsões e armazena a informação de forma recorrente sobre os seus dados de entrada (Roell, 2017).

- *Bi-Directional Long Short-Term Memory*

O algoritmo *Bi-Directional Long Short-Term Memory* é um algoritmo de aprendizagem supervisionada direcionado para processar um conjunto de dados de forma sequencial. De acordo com o autor Zhang et al. (Zhang et al., 2018) este algoritmo é aplicado para caraterizar o comportamento de degradação do sistema, conta com uma caraterística de dependência a longo prazo embutida na estrutura da rede *Long Short-Term Memory* (LSTM). Cada sequência deste algoritmo apresenta para a frente e para trás duas LSTM's separadas, permitindo assim o acesso à informação de forma completa antes e depois de cada passo de cada sequência. Este tem como objetivo separar cada sequência de treino em duas redes recorrentes, ou seja, neste algoritmo a rede têm sempre informações completas sobre todos os pontos de entrada e saída (Zhang et al., 2015).

- *AdaBoost*

O algoritmo *AdaBoost* é uma abreviação de "Adaptive Boosting" foi o primeiro algoritmo *boosting* desenvolvido por Freund e Schapire (Schapire, 2013). Este pode ser usado para classificação ou regressão e utiliza como base o algoritmo de Árvores de Decisão de apenas um nível para fazer previsões sobre o conjunto de dados de treino. Para além disso, mantém um conjunto de pesos sobre o conjunto de dados de treino. Em cada iteração, o algoritmo é chamado para minimizar o erro ponderado no conjunto de treino. Os pesos das instâncias de treino classificadas de forma incorreta aumentam e o segundo classificador é treinado e os pesos atualizados são reconhecidos. No fim de cada previsão, o algoritmo aumenta os pesos das instâncias classificadas de forma incorreta para que o próximo modelo alcance uma melhor *performance*. Concluindo, o *AdaBoost* utiliza de forma complementar os modelos mais fracos e corrige as previsões alcançadas com a adição dos modelos considerados mais fracos (Kittler and Roli, 2000).

- *Bagging*

Os três principais métodos para combinar os algoritmos de aprendizagem automática são *bagging*, *boosting* e *stacking*. O método de *Bagging é* também denominado de *Bootstrap Aggregation* e conta com o algoritmo de classificação mais baixo para melhorar a precisão pretendida. Este método permite ajustar várias modelos independentes, é implementado tanto em algoritmos de classificação como regressão. Esta função é definida em relação ao conjunto de dados de treino, recebe um parâmetro de entrada, executa a média das suas previsões para obter um modelo com uma variância menor e devolve uma saída. Esta abordagem é a mais conhecida no mundo científico e tenta produzir um modelo mais robusto do que os modelos individuais que o compõem. Uma vantagem deste método é o baixo tempo de construção do modelo (Gaikwad and Thool, 2015).

- *Extra Trees*

O algoritmo denominado de *Extra Trees (Extremely Randomised Trees)* é uma técnica de aprendizagem automática bastante recente e utiliza o mesmo princípio da *Random Forest*. Este utiliza um subconjunto aleatório de caraterísticas para treinar cada estimador, seleciona de forma aleatória a melhor caraterística juntamente com o valor correspondente para dividir o nó. Para além disso, utiliza todo o conjunto de dados de treino para treinar cada árvore de regressão. Este algoritmo permite estimar a importância de cada caraterística no modelo (Ahmad, Reynolds and Rezgui, 2018).

3.2.2 Validação dos Modelos

Para avaliar o desempenho de um algoritmo de classificação existem duas abordagens nomeadamente, *K-Fold Cross-Validation* e *Leave-One-Out Cross-Validation*. De seguida, descreve-se uma das técnicas mais conhecidas para validar os modelos, *K-Fold Cross-Validation*. Esta técnica tem como propósito obter a precisão do modelo induzida a partir de um determinado algoritmo de classificação, pois as precisões obtidas nos dados de treino permitem alcançar excelentes resultados no que diz respeito à robustez dos modelos (Wong, 2015).

Na Figura 30, representa-se o processo de *Cross-Validation,* considerando a divisão do conjunto de dados em 5 *Folds*. Esta técnica conta com um parâmetro K que representa em quantas vezes o conjunto de dados deve ser dividido. Em cada iteração o conjunto de dados *K-Folds - 1 é* usado para treino e o último para teste do modelo, sendo este um processo de K etapas, com um conjunto de dados a variar (conforme ilustrado na Figura 30) (Raschka, 2020).

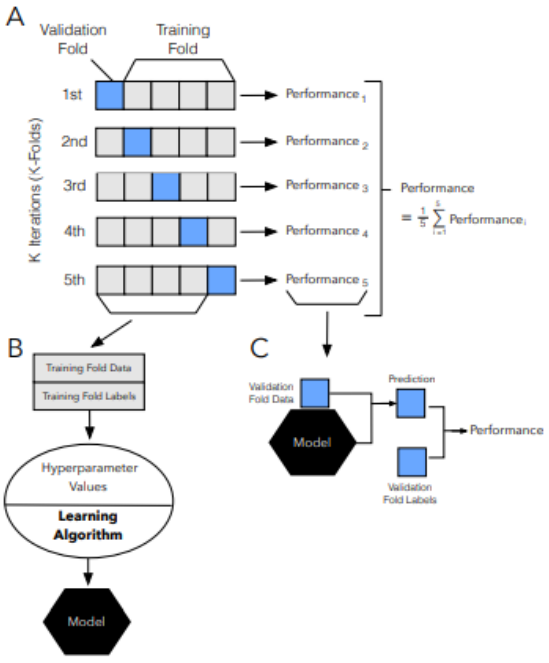


Figura 30 Cross-Validation com K-Folds = 5 (Raschka, 2020)

A utilização da *Cross-Validation* com *K-Folds* = 5 corresponde a cinco modelos diferentes que foram ajustados. Os modelos são ajustados tendo em conta um conjunto de treino distinto, mas parcialmente são sobrepostos e avaliados com base nos conjuntos de validação não sobrepostos.

Assim, a utilização desta técnica permite avaliar se os modelos entraram em *overfitting* ou se são capazes de generalizar perante novos dados de entrada (prova de conceito com base no bloco de teste de cada iteração). A consistência das *performances* obtidas nas várias iterações permitem avaliar a robustez dos modelos e consequentemente o grau de grandeza de *performance* (*accuracy*) do modelo implementado.

3.3 Projetos Relacionados

Nesta Secção 3.3, descrevem-se diferentes abordagens ao problema de deteção de notícias falsas com recurso ao *text mining*. De seguida, apresentam-se onze abordagens relativas à classificação em binário e em mais do que duas classes.

3.3.1 Classificação Binária

Nesta subsecção apresentam-se várias abordagens ao problema de classificação de notícias falsas em apenas duas classes (verdadeiro ou falso).

No estudo feito pelos autores Gaonkar et al. (Gaonkar et al., 2019) é proposto um modelo que pretende prever se uma dada notícia é verdadeira ou falsa, usando técnicas de processamento de linguagem natural e aprendizagem automática, conforme apresenta a Figura 31.

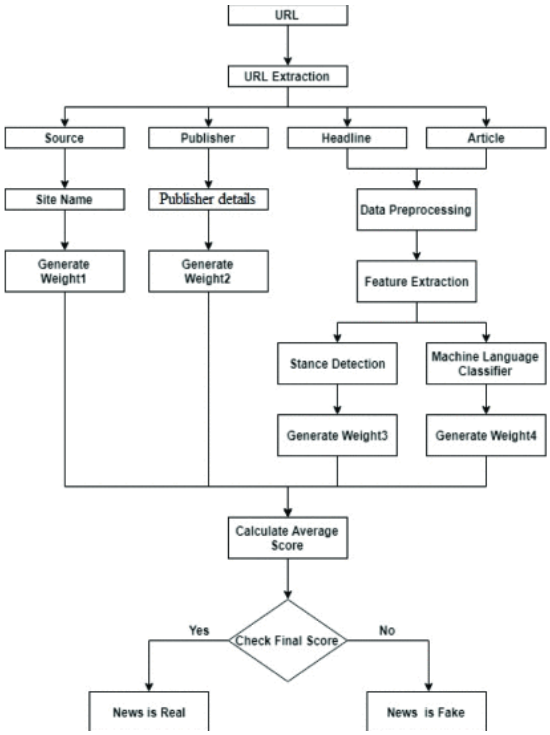


Figura 31 Abordagem proposta por Gaonkar et al.
(Gaonkar et al., 2019)

Tal como pode ser observado na figura anterior, foi utilizado o URL como parâmetro de entrada no modelo que contém vários metadados tais como: a fonte, o autor, o título e o artigo. Estes são

analisados um a um, sendo que de seguida são aplicadas algumas técnicas de pré-processamento, como por exemplo *stop-words removal*, tokenização e *stemming*. Para se extraírem as caraterísticas relevantes foram usadas as técnicas *TF-IDF e Probabilistic Context-Free Grammar*. Numa fase posterior, os resultados das fases anteriores são submetidos a um conjunto de algoritmos de classificação de aprendizagem automática e a cada metadado é atribuído uma pontuação. Para se proceder à classificação foram utilizados os seguintes algoritmos de classificação: *Naive Bayes*, *KNN*, Máquinas de Vetores de Suporte, *Linear Support Vector Machine*, Regressão Linear e por último, a Regressão Logística. Após se obterem todas as pontuações calcula-se a média com o propósito de construir uma pontuação final que determina se uma notícia é verdadeira ou falsa.

Numa outra abordagem, os autores Ahmad et al. (Ahmad et al., 2020) propuseram uma *framework* conforme é visível na Figura 32, que consiste na aplicação de um conjunto de técnicas linguísticas e no método de *ensemble* (uma técnica de aprendizagem máquina/automática que combina vários modelos para produzir um modelo preditivo ideal) com o propósito de classificar as várias notícias em verdadeiras ou falsas. Esta proposta aborda caraterísticas como o *Linguistic Inquiry and Word Count* (LIWC) e com o método *ensemble*.

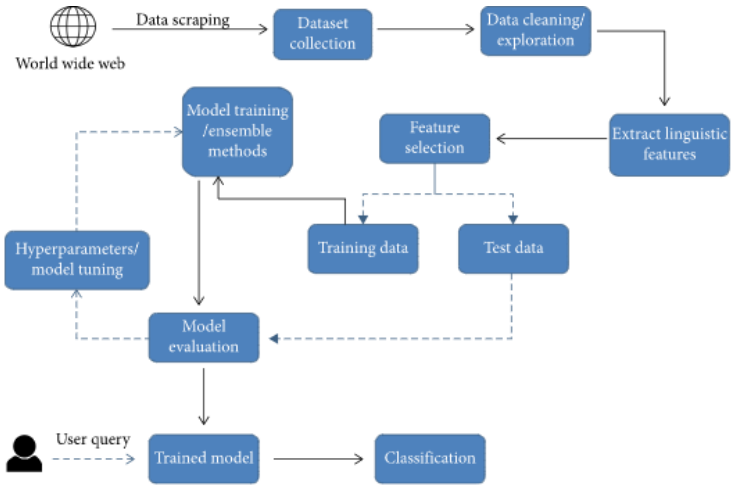


Figura 32 Abordagem proposta por Ahmad et al.
(Ahmad et al., 2020)

Os conjuntos de dados utilizados encontram-se disponíveis *online* e abordam vários domínios (tais como, política, entretenimento, tecnologia e desporto). Inicialmente, extraem-se os dados através de *data scraping*[44] (técnica que extrai os dados de forma legível para as pessoas), de seguida é feito um pré-processamento dos mesmos, que é responsável por eliminar todas as variáveis indesejadas tais como: os autores, a data de publicação, o URL, a categoria das notícias e as notícias que não contenham pelo menos 20 carateres no corpo das mesmas. Posteriormente, é feita uma extração das caraterísticas linguísticas que se encontram englobadas nas caraterísticas textuais (percentagem de palavras que classificam as mesmas em emoções negativas e positivas; pontuação; linguagem informal; percentagem de determinadas classes gramaticais como verbos, adjetivos, preposições, entre outros).

[44] https://academic.oup.com/bib/article/15/5/788/2422275?login=true

Estas precisam de ser transformadas em caraterísticas numéricas para serem utilizadas como entrada no modelo. Para extraírem as mesmas recorreram à ferramenta LIWC2015 que consegue dividir o texto em diferentes variáveis. Cada conjunto de dados é dividido em treino (70%) e teste (30%) e os algoritmos de aprendizagem são treinados com diferentes parâmetros para alcançar a máxima precisão possível, tendo em conta um determinado conjunto de dados. Cada modelo é treinado *n* vezes com diferentes parâmetros e recorreram ao *Grid Search* (método tradicional para otimizar os hiperparâmetros) para se obter os melhores parâmetros e o melhor resultado possível.

Para realizarem a experiência, recorreram a três conjuntos de dados existentes, sendo que o primeiro, o DS1, denomina-se de "ISOT Fake News Dataset" e contém cerca de 44.898 notícias falsas e verdadeiras extraídas *online*. As notícias falsas foram extraídas de várias fontes de informação principalmente de *sites* sinalizados pelo Politifact.com, as notícias verdadeiras foram extraídas do *site* da agência "reuters.com".

O segundo conjunto de dados, DS2, é extraído a partir de várias fontes de dados da Internet e é representado pelo *id* da notícia, título, autor, texto da notícia e uma *label* com valores em {0,1}.

O terceiro conjunto de dados, DS3, inclui 3.352 notícias falsas e verdadeiras, sendo que as notícias verdadeiras foram extraídas a partir dos *sites* da *CNN*, *Reuters*, *New York Times* e as notícias falsas são provenientes de *sites* de notícias que contém notícias que não são credíveis. Este conjunto de dados é composto pelo URL, título, texto da notícia e uma *label* com valores de zero e um. O último conjunto de dados, DS4, relaciona os três conjuntos de dados descritos anteriormente (DS1, D2, DS3), com o objetivo de avaliar o desempenho dos algoritmos.

Portanto, para cada conjunto de dados são testados vários algoritmos, o primeiro conjunto de dados DS1 atinge 99% de precisão com o algoritmo *Random Forest* e Perez-*Linear Support Vector Machine*. Sendo que também foram testados, o *Linear Support Vector Machine*, *Perceptron Multicamadas*, *Classifiers Bagging* e *Boosting* e alcançaram uma precisão de 98%. No DS2, o classificador *bagging* (Árvores de Decisão) e o *boosting* (XGBoost) são os que obtiveram melhor desempenho com uma precisão de 94%, porém os algoritmos *Linear Support Vector Machine*, *Random Forest* e Perez-*Linear Support Vector Machine* tiveram um mau desempenho. No DS3, o melhor algoritmo com uma precisão de 96% foi o Perez-*Linear Support Vector Machine*. E por fim, foi testado o conjunto de dados DS4 que combina os *datasets* anteriores e o algoritmo *Random Forest* foi o que apresentou melhor desempenho (91%) (Ahmad *et al.*, 2020).

No caso do trabalho desenvolvido pelos autores Ahmed et al. (Ahmed, Traore and Saad, 2017), estes propõem uma classificação dividida por cinco etapas conforme ilustra a Figura 33. Essas fases passam pela recolha dos dados, pré-processamento dos mesmos através de tokenização, *stop-words removal*, *stemming*, entre outros. De seguida, as caraterísticas importantes para o classificador são extraídas pelas técnicas de extração, nomeadamente, *TF* e *TF-IDF*. A última fase resume-se ao treino de seis classificadores nomeadamente, *Stochastic Gradient Descent* (SGD), Máquinas de Vetores de Suporte (MVS), *Linear Support Vector Machine* (LSVM), *K-Nearest Neighbors* (KNN) e Árvores de Decisão (DT).

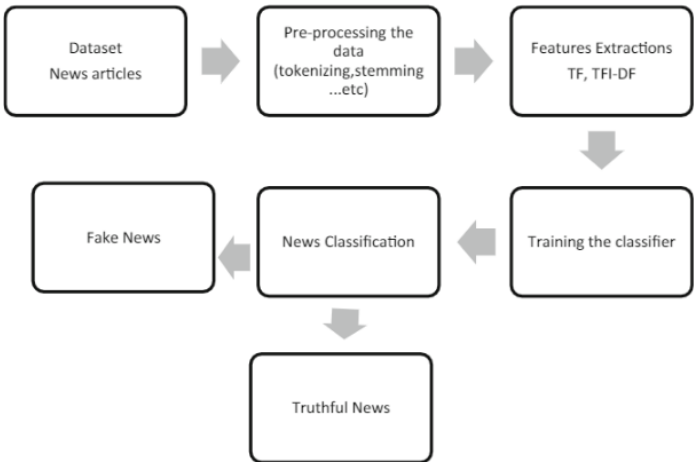


Figura 33 Classificação de notícias Ahmed et.al
(Ahmed, Traore and Saad, 2017)

Ahmed et al. (Ahmed, Traore and Saad, 2017) pretendiam classificar um conjunto de notícias em verdadeiro ou falso, para isso recorreram a um conjunto de dados recolhido a partir de fontes de notícias verdadeiras. Este conjunto conta com 25.200 notícias falsas e verdadeiras em contexto político, relativas ao ano de 2016. Cada notícia disponibiliza a seguinte informação: o texto da mesma, a categoria, a classe (verdadeiro ou falso), o título e a data de publicação. As experiências feitas começaram com a verificação do tamanho (*n*) dos *n-grams* no desempenho dos algoritmos de aprendizagem automática. Neste processo de verificação, iniciaram o processo com *uni-grams* (*n* = 1) até *four-grams* (*n* = 4) e cada valor de *n* foi testado com um número de caraterísticas que varia entre valores de 1.000 a 50.000. Estas experiências foram realizadas com uma validação cruzada de cinco vezes, para cada validação o conjunto de dados é dividido em 80% de treino e 20% em teste. Após diversas experiências, perceberam que os algoritmos lineares (*Linear Support Vector Machine, Stochastic Gradient Descent* e Regressão Logística) alcançam melhores resultados do que os não lineares. Sendo que o *Linear Support Vector Machine* alcançou 92% precisão e obteve bons desempenhos independentemente do número de caraterísticas utilizadas. Em suma, o modelo proposto por eles atinge a sua maior precisão recorrendo a caraterísticas *uni-grams* e com o algoritmo *Linear Support Vector Machine*, sendo esta de 92%.

Relativamente ao trabalho desenvolvido por Álvaro Figueira, Nuno Guimarães e José Pinto (Figueira, Guimarães and Pinto, 2019), foi construído um sistema de deteção baseado num *ensemble classifier* com a capacidade de prever se um determinado *post* é relevante ou não na perspetiva jornalística. A Figura 34 representa a arquitetura do sistema desenvolvido pelos mesmos e, tal como se pode verificar, depende de quatro módulos diferentes: *crawler,* que é responsável por extrair os dados da API do Twitter e do Facebook, limpeza dos dados, a extração das caraterísticas e, por último, um motor de análise para produzir uma decisão sobre se uma notícia (relevante ou não relevante).

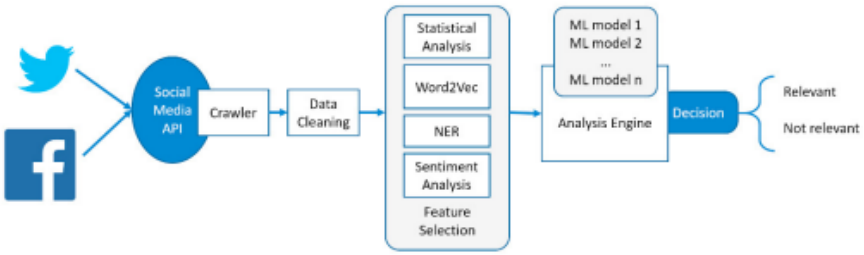


Figura 34 Arquitetura do sistema proposto por Figueira et. al.
(Figueira, Guimarães and Pinto, 2019)

Neste momento, o *crawler* apenas extrai *tweets* da API do Twitter pois a API do Facebook já não se encontra disponível. Tendo em conta a API do Twitter, foram extraídos os dados a partir da mesma de forma automática e direcionados para duas categorias, nomeadamente notícias e não notícias. Estas são usadas para treinar os modelos supervisionados. Estes dados são recolhidos tendo em conta um conjunto de contas previamente selecionadas no Twitter.

Os *tweets* que foram extraídos passam por um processo de limpeza e remoção de determinados carateres/símbolos específicos. Após os mesmos estarem processados, o sistema é capaz de identificar todas as caraterísticas que são dadas como entrada em cada conjunto de dados.

Posteriormente, ao processo de extração e seleção das caraterísticas, o sistema direciona-se para a última etapa que diz respeito ao processo de análise. Este é separado em conjunto de treino (66.6%) e teste (33.3%), onde foram aplicados *ensemble methods* supervisionados com o objetivo de obter o melhor modelo nomeadamente, *Linear Support Vector Machine* (LSVM), Radial Máquinas de Vetores de Suporte (RSVM), Polinomial Máquinas de Vetores de Suporte (PSVM), Árvores de Decisão (RF) e *Naive Baye*s (NB). Estes modelos são testados com o objetivo de verificar a relevância de determinadas notícias.

Em suma, utilizaram a técnica *Word2vec* para o *corpus* extraído, este é composto por 50000 notícias extraídas dos *websites OpenSources* e 50000 notícias extraídas dos vários Feeds RSS (The Guardian, CNN, BBC, The Economist, Washington Post, ABC, CBS News, Wall Street Journal, NBC News, Reuters, Sky News e Fox News). No total da experiência consideraram 100 caraterísticas máximas no modelo e para avaliar os modelos fizeram uma divisão do conjunto de dados em 90% de treino e 10% para teste. Os três algoritmos que foram selecionados foram treinados utilizando uma *K-Cross Fold Validation* de 10 *Folds* com 5 repetições. Segundo a avaliação dos próprios, verificou-se que o algoritmo MVS atingiu um melhor e mais alto desempenho com 87% *F1-Score*, 85% de precisão e 89% de *recall*.

Relativamente ao trabalho desenvolvido pelos autores Posadas Durán et al. (Posadas Durán *et al.*, 2019) contaram com um *dataset* denominado "Spanish Fake News Corpus" que contém notícias de vários jornais, empresas de comunicação social e *websites* de verificação de notícias. As notícias falsas foram retiradas através de *websites* analisados por jornalistas. As notícias são relativas ao ano 2018, escritas em espanhol e só o *corpus* da notícia foi anotado em duas classes (Verdadeiro/Falso).

Este trabalho teve como principal propósito identificar automaticamente as notícias falsas. Assim, o conjunto de dados foi dividido em 30% de teste e 70% de treino, de seguida foram utilizadas três representações de caraterísticas (BoW, *N-grams* e *POS N-grams)*. Avaliaram o desempenho do tipo de palavras e carateres com *N-grams* ao incluir e excluir *stop-words* para entrada no modelo. O pré-

processamento do texto passou pela eliminação de *stop-words* e dos sinais de pontuação. O desempenho de cada um dos conjuntos de caraterísticas foi avaliado de forma separada e através de diferentes combinações.

Ao longo das várias experiências foram utilizados os seguintes quatro algoritmos: Máquina de Vetores de Suporte, Regressão Logística, *Random Forest* e *Boosting* sobre o conjunto de caraterísticas descrito anteriormente e a métrica de avaliação do modelo foi a *accuracy*. Os algoritmos foram treinados com tamanho dos *N-grams* entre 3 e 5, sendo que os melhores resultados no conjunto de teste obtidos foram com N-grams = 4, sem remover as palavras denominadas *stop-words*. Segundo as várias experiências, o conjunto de caraterísticas com melhor *performance* foi *BoW + POS* e o algoritmo *Random Forest* foi o que alcançou uma *accuracy* de 76.94%.

De acordo com o trabalho desenvolvido pelos autores Silva et al. (Silva et al., 2020), o *dataset* que foi desenvolvido pelos mesmos, "Fake.Br Corpus", é composto por 7200 notícias recolhidas de forma manual sendo que 3600 notícias correspondem à classificação falsa e as outras 3600 à classificação verdadeira, do ano de 2016 ao ano 2018. As notícias foram classificadas como falsas, tendo em conta diferentes critérios, tais como: o autor da notícia não estar citado; o título da notícia ser apelativo para o utilizador clicar no mesmo; a notícia conter erros gramaticais; a notícia no seu todo conter muitas letras maiúsculas, com vários pontos de exclamação/interrogação; a notícia não conter data/ fontes/referências; o site da notícia não apresentar as pessoas responsáveis pelo mesmo e por último o *site* conter um *layout* duvidoso. Posteriormente, utilizaram vários *websites* de verificação de factos de português do Brasil, nomeadamente, Agência Lupa, Fato ou Fake, Aos Factos e Boatos.org para verificar a credibilidade de cada notícia. As notícias verdadeiras foram recolhidas através de um *web crawler*. As notícias recolhidas foram associadas de forma manual a uma das seguintes categorias: economia, ciência e tecnologia, sociedade e notícias diárias, política, religião, TV e celebridades.

Posteriormente, foram efetuadas diferentes experiências que tiveram como base a componente linguística, várias técnicas de representação de texto (*BoW*, *Word2Vec* e *FastText*) e os vários métodos de classificação: Regressão Logística, Máquina de Vetores de Suporte, Árvores de Decisão, *Random Forest*, *Bootstrap Aggregating* (*Bagging*) e por último, o *Adaptive Boosting* (*AdaBoost*). As medidas que foram utilizadas para comparar o desempenho de cada algoritmo foram: taxa de falsos positivos (LBR); *recall* (FCR); taxa de precisão de notícias falsas (FPR) e a *F-Measure*.

A primeira experiência contou com os algoritmos descritos anteriormente e com recurso à base linguística. O melhor algoritmo foi sem dúvida o *Random Forest* com um LBR de 0.060, FCR de 0.94, FPR de 0.94 e por último o *F-Measure* de 0.94. O que teve um pior desempenho foi a Árvore de Decisão com um LBR de 0.099, FCR de 0.90, FPR de 0.90 e por último a métrica *F-Measure* de 0.90. Em suma, desta experiência todos os algoritmos alcançaram uma *F-Measure* acima dos 90% o que indica que a componente de recursos com base linguística tem um peso bastante significativo para detetar mais de 90% das notícias falsas (FCR).

A segunda experiência contou com os mesmos algoritmos, mas com os recursos gerados pelas técnicas de representação de texto (*BoW, Word2Vec* e *FastText*). Na primeira fase, o texto não passou pela etapa de pré-processamento de dados, ou seja, não foram removidas as *stop-words* e não foi aplicado o processo de *lemmatization*. Nesta, utilizaram como técnica o *BoW* e o melhor algoritmo foi a Regressão Logística com uma métrica de *F-Measure* correspondente a 98% notícias falsas. Na segunda fase foram removidas apenas as palavras irrelevantes (*stop-words*) e utilizaram a técnica *BoW*,

o melhor algoritmo novamente foi a Regressão Logística com uma *F-Measure* de 96% notícias falsas. Na terceira fase, removeram as *stop-words*, aplicaram a técnica *lemmatization* ao texto e a técnica *BoW* foi novamente aplicada. Neste caso, o algoritmo com a *performance* de 96% a nível da métrica *F-Measure* foi a Regressão Logística. Na quarta fase, foram removidas as palavras irrelevantes e a técnica *BoW* contou com as 1.000 melhores *features* para entrada do modelo e o algoritmo com melhor *F-Measure* foi a Regressão Logística com 95%. De seguida, existiram outras experiências com outras técnicas, sendo que a melhor métrica *F-Measure* foi de 89%, (uma diferença de mais de 7%). Em suma, a técnica *BoW* foi a que obteve melhores resultados.

Em relação, ao trabalho desenvolvido pelos autores Zervopoulos et al. (Zervopoulos et al., 2020) contaram com um conjunto de dados provenientes do Twitter com um total de 9298 *tweets,* correspondente a 3910 *tweets* de notícias falsas e 5388 *tweets* que corresponde às notícias verdadeiras.

Os algoritmos utilizados ao longo da fase de experimentação foram o *Naive Bayes*, Máquina de Vetores de Suporte, C4.5 e o *Random Forest*. De todo o conjunto de dados foram selecionados diferentes *features*, tendo sido empregues as 10 primeiras *features* com um peso mais significativo, nomeadamente: comprimento do *tweet*, TTR, número de sinais de pontuação, número médio de carateres por frase, número de advérbios, número de palavras “to”, número de verbos, entidades, períodos e a entropia do *tweet*. Tendo em conta que a maior parte das *features* são de caráter linguístico, o objetivo foi identificar quais as palavras que correspondem às notícias falsas e às verdadeiras.

Em toda a fase de experimentação foi utilizada a *K-Cross Validation* com k = 5 para aumentar a confiabilidade dos resultados. As métricas que foram utilizadas na avaliação dos diferentes modelos foram precisão, *recall* e *F1-Score*. Todos os algoritmos descritos anteriormente tiveram um desempenho razoável. No entanto, em relação à métrica de *F1-Score* sem dúvida o *Random Forest* foi o que apresentou uma melhor *performance* e o que alcançou uma pontuação de 92%.

Tendo em conta que o melhor algoritmo foi a Árvore de Decisão, os autores tentaram compreender qual a *feature* que contribuía significativamente para os resultados alcançados e foi claro que a entropia do *tweet* teve um contributo bastante significativo na classificação. Segundo os autores, os modelos podem ter obtido *performances* mais elevadas, pois os *tweets* que se encontravam em chinês foram traduzidos através do Google Tradutor para inglês.

Por último, no projeto de dissertação de Rodrigues (Rodrigues, 2020) foi criado um *dataset* denominado *FakePT* através de *web scraping*. Este conjunto conta com notícias em português europeu com datas compreendidas entre Junho de 2018 e Agosto de 2020, composto por 3764 notícias com as seguintes caraterísticas: *title, source, summary, category* e *tag.* Estes registos foram classificados através de três *websites* de verificação de factos, nomeadamente: Polígrafo, Observador e Corona Verificado.

Tendo em conta que as notícias extraídas foram de diferentes *websites*, houve uma necessidade de transformarem todas as classificações em apenas duas classes (Verdadeiro/Falso). Assim, reagruparam as notícias Verdadeiras da seguinte forma: notícia classificada como totalmente verdadeira; se não houver provas de que a notícia foi manipulada/distorcida; se a notícia tiver uma estrutura duma notícia Verdadeira mesmo sem contexto. As notícias que foram classificadas como

Falsas seguiram os seguintes pontos: presença de elementos que podem distorcer a realidade; notícia classificada como totalmente falsa; sátiras.

De seguida, construiu uma pipeline de classificação de notícias falsas com as seguintes etapas: pré-processamento dos dados, seleção dos dados e classificação dos mesmos. Na primeira e na segunda fase, foram testadas diferentes técnicas de NLP (remoção da pontuação, *lemmatization*, remoção das *stop-words*, transformação das palavras em minúsculas, *Part-Of-Speech Tagging (*POS-Tagging*)*, W*ord2vec*, *Bow + TF-IDF)* nas variáveis textuais (*Title*, *Summary* e *Title + Summary*), bem como as técnicas de transformação de texto com maior impacto para o modelo final. Após a limpeza dos dados, os mesmos foram divididos em dois blocos, com valores entre 70-80% dos dados para treino e 30-20% para teste. Com o objetivo de determinarem a robustez do modelo final utilizaram duas técnicas semelhantes de *Cross-Validation*: *K-Fold* e *Stratified K-Fold.* A última fase da *pipeline* consistiu na aplicação dos vários algoritmos de classificação tanto de aprendizagem automática como de *Deep Learning*, nomeadamente *Multinomial Naive Bayes, Random Forest*, Regressão Logística, *Extra Trees*, CNN, LSTM, *K-Nearest Neighbors, Linear Support Vector Machine*, Máquina de Vetores de Suporte e *Gradient Boosting*.

Na fase de experiências e avaliação da solução, foram avaliados os dez algoritmos tendo em conta as várias métricas de avaliação. Os mesmos, são comparados com o propósito de se identificar quais os parâmetros, o pré-processamento, técnicas de representação de texto e os algoritmos que levam à melhor pontuação de avaliação.

Conforme foi mencionado anteriormente, este projeto contou com duas fases. A primeira fase consistiu na determinação do melhor conjunto de técnicas de pré-processamento dos dados e qual a melhor técnica para representar as variáveis textuais, e a segunda fase contou com dois conjuntos principais de experiências: uma com um conjunto de dados equilibrados e outra com o conjunto de dados desequilibrados.

Esta segunda fase teve como objetivo encontrar a melhor combinação a nível das caraterísticas, mas também alcançar o algoritmo com melhor desempenho, consoante o conjunto de dados testados. No caso do conjunto de dados equilibrados, a métrica que avalia o desempenho do algoritmo foi a *accuracy* e no caso do conjunto de dados desequilibrados foi utilizada a métrica *f1-score*.

Por um lado, no conjunto de dados equilibrados (50% notícias falsas e verdadeiras), foram testadas as três dimensões (*Title*, *Summary* e *Title+Summary*) com a técnica de representação denominada BoW com TF-IDF. No caso da dimensão título, o melhor algoritmo foi a Máquina de Vetores de Suporte com uma *accuracy* de 66%. Na dimensão resumo, o algoritmo com melhor performance foi *Extra Trees* com um valor de 74%. E por último, na dimensão título e resumo, o algoritmo Máquina de Vetores de Suporte apresentou uma *accuracy* de 70%. Tendo em conta os resultados expostos, a melhor dimensão foi o resumo e de seguida a mesma foi combinada com as diferentes dimensões categóricas sob diferentes técnicas de representação de texto. Das experiências realizadas, a que obteve melhor pontuação foi BoW com TF-IDF, as melhores dimensões foram (resumo e fonte de informação) com o algoritmo *Extra Trees* com 74%. Por outro lado, no conjunto de dados desequilibrados (77% notícias falsas e 23% de notícias verdadeiras), foram testadas as três dimensões (*Title*, *Summary* e *Title+ Summary*) com a técnica de representação denominada BoW com TF-IDF. No caso da dimensão (título e resumo), o melhor algoritmo foi a *Linear Support Vector Machine*, com um desempenho de 0.87 na classe Falso e 0.45 na classe Verdadeiro. Na dimensão resumo, o melhor algoritmo foi a *Linear Support*

Vector Machine, com um desempenho de 0.86 na classe Falso e 0.45 na classe Verdadeiro. E por último, na dimensão título, o melhor algoritmo foi a *Linear Support Vector Machine*, com um desempenho de 0.87 na classe Falso e 0.43 na classe Verdadeiro. Tendo em conta os resultados expostos, a melhor dimensão foi o (título e resumo). De seguida, a mesma foi combinada com as diferentes dimensões categóricas sob diferentes técnicas de representação de texto. Das experiências realizadas, a que obteve melhor pontuação foi BoW com TF-IDF, *One Hot Encoding*, as melhores dimensões foram título, resumo e categoria com o *Multinomial Naive Bayes* com um desempenho de 0.86 na classe Falsa e 0.60 na classe Verdadeiro.

Em suma, no conjunto de dados equilibrados, a dimensão resumo com técnica de representação de texto BoW com TF-IDF, alcançou uma *performance* de 74% com o algoritmo *Extra Trees*. No conjunto de dados desequilibrados, a dimensão (título, resumo, categoria), em conjunto com as técnicas Bow com TF-IDF e *One Hot Encoding,* obteve uma *f1-score* de 86% na classe Falso e 60% na classe Verdadeira (Rodrigues, 2020).

3.3.2 Classificação em Várias Classes

Para a classificação em várias classes foram identificados três projetos distintos. O trabalho documentado por Kaliyar et al. (Kaliyar, Goswami and Narang, 2019) consistiu na proposta de um método, avaliado com base no *dataset* "Fake News Challenge" (FNC-1) que contém os seguintes dados: o título, o corpo da notícia e uma *label* com quatro opções de classificação {"*unrelated"*, "*discuss"*, "*agree"* e "*disagree"*}. De seguida, implementaram nas variáveis textuais as técnicas de TF-IDF, *Cosine Similarity, Hand Selected, Word Overlap, Polarity* e por último, *Refuting* para obter as caraterísticas de entrada para o modelo. Com o propósito de selecionar os parâmetros foram testados os seguintes algoritmos: Classificador da Árvore Aleatória, *Multinomial NB*, *Gradient Boosting Classifier*, *Linear Support Vector Machine*, Classificador da Árvore de Decisão e a Regressão Logística. O recurso ao *Gradient Boosting* feito utilizando um algoritmo que atinge altos níveis de desempenho. As experiências efetuadas foram conduzidas com base nos desempenhos dos algoritmos de classificação abordados anteriormente. No que diz respeito à precisão, o algoritmo *Gradient Boosting Classifier* apresenta um valor de 86%. Também, foi calculada a pontuação de confiança atribuída a cada entrada, que indica o grau de confiança do algoritmo ao prever essa classe como saída. Kaliyar et al. (Kaliyar, Goswami and Narang, 2019) obtiveram, com a sua investigação, um total de 11651 instâncias previstas. Na pontuação de confiança verificou-se que o *Gradient Boosting* funciona bem com o modelo TF-IDF (para extração de caraterísticas). Em suma, o algoritmo *Gradient Boosting* apresenta uma grande aptidão para resolver problemas de classificação textual em diferentes classes. Os autores sugerem um modelo híbrido para se obter melhores classificações de notícias usando o seu conteúdo como a informação textual das mesmas.

O segundo projeto em multiclasses analisado diz respeito ao trabalho desenvolvido por Roy et al. (Roy et al., 2018), o conjunto de dados encontra-se disponível na plataforma LIAR e é composto por 12.800 afirmações curtas anotadas conjuntamente com as várias informações do orador. As anotações são feitas em seis classes: {*"Pants-Fire"*, *"False"*, *"Barely-True"*, *"Half-True"*, *"Mostly-True"* e *"True"*}. Cada linha do conjunto apresenta uma afirmação curta, uma classe da afirmação {"*Pants fire count*", "*False count*", "*Barely true count*", "*Half false count*" e "*Mostly true count*"} e onze colunas que dizem respeito às informações sobre o orador. Este *dataset* é composto por três conjuntos, nomeadamente

um conjunto com 10.269 afirmações, um conjunto de validação com 1.284 afirmações e, por último, um conjunto de testes com 1.266 afirmações. Todas as experiências foram realizadas na linguagem Python e com o recurso às seguintes bibliotecas: Keras, NLTK, Numpy, Pandas e Sklearn. O sistema foi avaliado com base nas métricas de *precision*, *recall* e *f1-score*. O modelo proposto por eles consistiu na combinação de dois algoritmos de redes neuronais (*Recurrent Neural Network* (RNN) e *Convolutional Neural Network (*CNN)) e apresentou um desempenho de 45%, melhor do que outros modelos propostos. Também foram avaliados os resultados dos seguintes algoritmos em termos de precisão, (CNN) (48% precisão), Redes Neuronais Recorrentes Bidirecionais (Bi-LSTM) (53% precisão) e da combinação do modelo CNN com BI-LSTM (55% precisão).

Relativamente ao último projeto analisado, Rasool et al. (Rasool et al., 2019) propuseram um método de deteção de notícias falsas em vários níveis com base em algoritmos de aprendizagem supervisionada. Estes abordaram a classificação das mesmas em três etapas, que consistem na extração de caraterísticas, re-etiquetagem e aprendizagem do modelo, conforme ilustra a Figura 35.

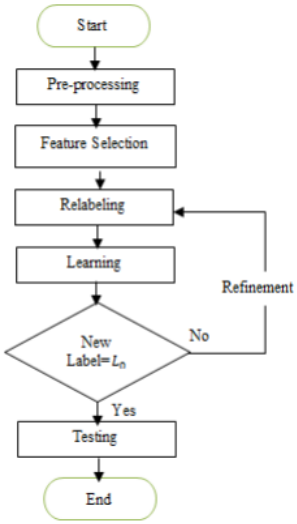


Figura 35 Abordagem de classificação das notícias em várias classes
(Rasool et al., 2019)

Na abordagem dos autores, os dados das notícias são pré-processados manualmente antes de serem utilizados para a construção do modelo. Esta consiste na limpeza manual das notícias e na conversão dos dados textuais em dados numéricos, de seguida procederam à seleção de caraterísticas para garantir que são excluídas as menos relevantes e que podem contribuir para a diminuição do desempenho do algoritmo. Na etapa de re-etiquetagem, consideraram duas classes sendo estas, 1-3 e 4-5, posteriormente resolveram o problema de multiclasses como se fosse uma classificação binária normal (verdadeiro ou falso). Tendo em conta estes fatores, as classes construídas foram treinadas em diferentes classificadores.

A solução proposta foi avaliada com base no conjunto de dados que se encontra disponível na ferramenta LIAR, esta conta com 12.836 dados que contém o texto da notícia, tópicos e algumas caraterísticas relacionadas com base no perfil do autor, mas também estão classificadas em seis classes {*"Pants-On-Fire"*, *"False"*, *"Barely false"*, *"Half-true"*, *"Mostly-true"* e *"True"*}. Após se ter feito a limpeza dos dados obtiveram-se 10.235 dados para se fazerem diferentes experiências.

Para a extração de caraterísticas, os autores usaram a ferramenta Weka, o que resultou na extração de cinco caraterísticas principais {*"Barely-true"*, *"False", "Half-true"*, *"Mostly true"* e *"Pants-on-Fire"*}. O conjunto de dados foi pré-anotado e as *labels* "Pants-on-Fire" e falso são considerados falsos, como verdadeiras são classificadas quase verdade, perto da verdade, praticamente verdade e verdade. Após *n* treinos, o conjunto de dados considerado sofre uma nova ramificação e fica classificado como {*"Barely-true"*, *"Half true"*} e {*"Mostly-true"*, *"True"*}. Os modelos multiníveis são treinados com algoritmos de *machine learning* que são testados com Máquinas de Vetores de Suporte e Árvores de Decisão. Estas avaliações foram feitas com *hold-out*, teste de conjuntos e abordagens de validação cruzada. Em suma, os algoritmos testados são algoritmos de classificação Máquinas de Vetores de Suporte e Árvore de Decisão. Os resultados são comparados com técnicas de referência e os resultados demonstraram uma boa *performance* em termos de precisão. O método proposto tem um desempenho superior ao de referência com uma precisão de 39,5%.

3.3.3 Conclusões

Tendo em conta a descrição feita de cada um dos projetos analisados anteriormente, pode-se afirmar que se analisou um maior número de projetos que permitem uma classificação binária do que aqueles que permitem uma classificação em multiclasses. A Tabela 10 construída com o propósito de resumir os trabalhos apresentados no que diz respeito ao método de recolha e transformação do conjunto de dados, o nome do *dataset* utilizado, o conjunto de dados ou parâmetros de entrada para o modelo e por fim a língua descrita no conjunto de dados.

Tabela 10 Comparação dos vários projetos estudados

Caraterísticas / **Artigos**	**Método de construção do *dataset***	**Nome do *dataset***	**Conjunto de dados/ Parâmetros de entrada**	**Língua**
Gaonkar et al. (Gaonkar *et al.*, 2019)	-	-	URL (Fonte; Autor; Título; Conteúdo)	Inglês
Ahmad et al. (Ahmad *et al.*, 2020)	*Data Scraping*	ISOT Fake News Dataset	DS4	Inglês
Ahmed et al. (Ahmed, Traore and Saad, 2017)	Manual	-	Texto, Tipo, *Label*, Título, Data de Publicação	Inglês
Figueira et al. (Figueira, Guimarães and Pinto, 2019)	*Crowdsourcing*	-	*Post*	Inglês
Posadas Durán et al. (Posadas Durán *et al.*, 2019)	Manual	Spanish Fake News Corpus	*Corpus*	Espanhol
Silva et al. (Silva *et al.*, 2020)	Semi-automático	Fake.Br	*Link*, Data, Número de *links*	Português do Brasil
Zervopoulos et al. (Zervopoulos *et al.*, 2020)	Manual	Twitter	*Tweets*	Inglês e Chinês
Rodrigues (Rodrigues, 2020)	*Web Scraping*	FakePT	*Title, Summary, Category, Source* e *Tag*	Português Europeu
Kaliyar et al. (Kaliyar, Goswami and Narang, 2019)	Manual	FNC-1	-	Inglês
Roy et al. (Roy et al., 2018)	Manual	Liar	-	Inglês
Rasool et al. (Rasool *et al.*, 2019)	Manual	Liar	Texto, Dados do Orador, Seis *Labels*	Inglês

Dos projetos estudados verifica-se que a maior parte constrói o *dataset* de forma manual e que a linguagem mais utilizada foi, sem dúvida, o inglês. No entanto, existem outros projetos que foram desenvolvidos noutras línguas, nomeadamente em espanhol, português europeu, português do brasil e chinês.

A Tabela 11 representa o conjunto de algoritmos utilizados nos projetos estudados, o índice de credibilidade de duas ou mais classes, bem como a métrica utilizada para avaliar os resultados dos modelos, a percentagem da mesma e o melhor algoritmo correspondente a cada estudo.

Tabela 11 Comparação dos projetos estudados em relação aos algoritmos utilizados

Caraterísticas / Artigos	*Naïve Bayes*	*K-Nearest Neighbors*	Máquinas de Vetores de Suporte	Regressão Linear	Regressão Logística	*Linear Support Vector Machine*	Árvore de Decisão	*Multinomial Naïve Bayes*	*Random Forest*	*Stochastic Gradient Descendent*	*Gradient Boosting Classifier*	*Convolutional Neural Network*	*Recurrent Neural Network*	*AdaBoost*	*Bagging*	*Bi-directional Long Short-Term Memory*	*Extra Trees*	Binário	Multiclasses	Métrica e *Performance*
Gaonkar et al. (Gaonkar et al., 2019)	✓	✓	✓	✓	✓	✓												✓		Pontuação Final
Ahmad et al. (Ahmad et al., 2020)		✓	✓				✓											✓		Accuracy (91%) RF
Ahmed et al. (Ahmed, Traore and Saad, 2017)		✓	✓			✓	✓			✓								✓		Accuracy (92%) LSVM
Figueira et al. (Figueira, Guimarães and Pinto, 2019)	✓					✓			✓									✓		F1-Score (87%) MVS
Posadas Durán et al. (Posadas Durán et al., 2019)			✓		✓				✓									✓		Accuracy (76%) RF
Silva et al. (Silva et al., 2020)	✓		✓		✓		✓		✓					✓	✓			✓		F-Measure (95%) RL
Zervopoulos et al. (Zervopoulos et al., 2020)	✓		✓						✓									✓		F1-Score (92%) AD
Rodrigues (Rodrigues, 2020)		✓	✓		✓	✓		✓	✓		✓	✓	✓				✓	✓		Accuracy (74%) ET
Kaliyar et al. (Kaliyar, Goswami and Narang, 2019)					✓	✓	✓	✓	✓		✓								✓	Accuracy (86%) GBC
Roy et al. (Roy et al., 2018)												✓	✓			✓			✓	Accuracy (55%) CNN + BI-LSTM
Rasool et al. (Rasool et al., 2019)			✓				✓												✓	Accuracy (39.7%) DT

Na Tabela 11, cada algoritmo é acompanhado da enumeração dos artigos, até à data analisados, que a este fazem referência. Relativamente aos algoritmos mais utilizados para o problema de classificação de notícias em binário destacam-se os seguintes: Máquina de Vetores de Suporte e *Random Forest.* Para o caso de classificação em multiclasses foi considerado a Árvore de Decisão.

Em suma, analisaram-se oito projetos que utilizam uma abordagem binária para a classificação e três projetos que classificam em multiclasses. Relativamente à abordagem binária, no projeto desenvolvido por Ahmad et al. (Ahmad et al., 2020) o melhor algoritmo foi o *Random Forest* com uma *accuracy* de 91%. No caso dos autores Ahmed et al. (Ahmed, Traore and Saad, 2017) o melhor algoritmo foi o *Linear Support Vector Machine* com uma *accuracy* de 92%. Os autores Posadas Durán et al. (Posadas Durán et al., 2019) conseguiram que o seu modelo *Random Forest* alcançasse uma *accuracy* de 76.94%. A métrica utilizada pelos autores Silva et al. (Silva et al., 2020) para avaliar a solução foi *F1-Measure* com 95% e o melhor algoritmo foi a Regressão Logística. O projeto Zervopoulos et al. (Zervopoulos et al., 2020) alcançou uma métrica *F1-Score* de 92% e o melhor algoritmo foi Árvore de Decisão. O autor Rodrigues (Rodrigues, 2020) alcançou um modelo com uma *accuracy* de 74% e o melhor algoritmo foi *Extra Trees.* Em relação à classificação multiclasses, a solução implementada por Kaliyar et al. (Kaliyar, Goswami and Narang, 2019) alcançou uma *accuracy* de 86% e o melhor algoritmo foi GBC. O último projeto desenvolvido pelos autores Roy et al. (Roy et al., 2018), o modelo CNN + BI-LSTM, contou com uma *accuracy* de 55%.

3.4 Tecnologias Existentes

Na presente Secção 3.4 descrevem-se as diferentes tecnologias/ferramentas que podem ser consideradas nas várias etapas do projeto. De seguida, apresentam-se três diferentes tecnologias que pode ser relevante para cada fase do sistema envolvente (aprendizagem automática, processamento de dados e desenvolvimento *web*).

3.4.1 Pré-processamento do Texto

De seguida, são apresentadas três bibliotecas disponíveis na linguagem Python para concretizar o pré-processamento do texto nomeadamente, Spacy, Scikit-Learn e NLTK.

- Spacy

Spacy[45] é uma biblioteca gratuita que se encontra disponível na linguagem Python, tem como objetivo processar um texto e transformá-lo num documento, mas também proporciona o desenvolvimento de sistemas capazes de extrair a informação. Esta biblioteca conta algumas caraterísticas: disponibiliza mais de sessenta e nove línguas, apresenta um vetor de palavras que já foi pré-treinado, apresenta diferentes componentes para o reconhecimento de entidades (*part-of-speech tagging*, classificação de textos, *lemmatization*, morfológica, entre outras) e tem uma precisão robusta.

[45] https://spacy.io/

- Scikit-learn

Scikit-learn (Sklearn)[46] é uma *framework open-source* de aprendizagem máquina/automática para a linguagem de programação Python. Esta foi desenvolvida em 2007 pela Google e atualmente considera-se como uma das melhores bibliotecas e mais eficiente para a aplicação de *data mining* e análise de dados. Esta biblioteca oferece um vasto leque de algoritmos de aprendizagem máquina/automática supervisionados (Máquinas de Vetores de Suporte*, Stochastic Gradient Descent*, *Naive Bayes*, Árvores de Decisão, *Ensemble*) e não supervisionados (Agrupamento, Estimação da Densidade e Modelos de Redes Neuronais). Esta é baseada num conjunto de outras bibliotecas, como o *Numpy*, *Scipy*, *Matplotlib*, *Pandas*, entre outras.

- NLTK

Natural Language ToolKit (NLTK)[47] foi lançado em 2001 por Steven Bird, conta com um conjunto de bibliotecas e programas direcionadas para o processamento de linguagem natural e análise do texto, desenvolvido em Python. O objetivo principal desta biblioteca é apoiar a pesquisa e o ensino do processamento de linguagem natural, incluindo a inteligência artificial, recuperação de informação, entre outros. O NLTK dispõe das seguintes funcionalidades: processamento de texto para classificação, tokenização, *stemming*, *tagging*, *parsing* e entre outras.

- Comparação

Para se proceder à comparação das bibliotecas anteriormente expostas e determinar qual ou quais se adequam melhor ao desenvolvimento deste projeto, procedeu-se à construção da Tabela 12.

Tabela 12 Comparação das bibliotecas para *Machine Learning* (Bobriakov, 2018)

Atributos / Bibliotecas	Ferramenta	*Open-Source* e gratuita	Algoritmos	Funcionalidades	Outros
Spacy	Muito rápida; Fácil de aprender e de utilizar		Recorre às redes neuronais para treinar o modelo	Processa objetos orientados (um conjunto de objetos que interagem entre si) ; Recorre às redes neuronais para treinar o modelo ; Oferece uma biblioteca com uma forte componente relativa ao processamento de linguagem natural	Separação da frase em *tokens* é lenta; Suporta língua portuguesa
NLTK	Lenta e difícil de usar	✓		Contém um conjunto de pacotes de bibliotecas de pré-processamento	Separação das palavras por *tokens* é um processo rápido; Suporta língua portuguesa
Scikit-learn	Eficiente e simples para a análise preditiva dos dados	✓	Classificação, Regressão	Usado para classificação, regressão, pré-processamento, redução da dimensionalidade dos dados, seleção dos dados, pré-processamento	Apresentam uma boa documentação e classes intuitivas

Desta tabela, pode-se concluir que todas as ferramentas são gratuitas e *open-source*, porém a biblioteca NLTK torna-se pouco relevante pois apresenta uma biblioteca bastante complexa e em termos usabilidade é lenta. As outras duas bibliotecas podem ser utilizadas no desenvolvimento do projeto.

[46] https://scikit-learn.org/stable/
[47] https://www.nltk.org/

3.4.2 Bibliotecas para Machine Learning

Nesta Secção 3.4.2 descrevem-se as diferentes bibliotecas, TensorFlow, Keras e Scikit-learn, que disponibilizam um conjunto de algoritmos de aprendizagem automática. A biblioteca Scikit-learn foi descrita anteriormente, mas também foi considerada pois apresenta um conjunto de algoritmos que pode ser pertinente para o decorrer do trabalho.

- TensorFlow

TensorFlow[48] é uma biblioteca *open-source* desenvolvida pela *Google Brain Team*, dedicada ao desenvolvimento de modelos de aprendizagem automática. Este permite treinar e implantar os modelos de uma forma simples, independentemente da linguagem ou da plataforma que se utiliza. Esta biblioteca pode ser executada em diversos CPU's e GPU's, bem como diversos sistemas operativos e dispositivos móveis. Algumas das caraterísticas presentes nesta são a depuração mais rápida com ferramentas Python, estar bem documentado e permite treinar e implementar o modelo rapidamente, entre outros.

- Keras

A biblioteca Keras[49] começou a ser desenvolvida como uma parte do esforço da pesquisa de um projeto chamado ONEIROS (*Open-Ended Neuro-Electronic Intelligent Robot Operating System*). Neste momento, o Keras é uma API de redes neuronais de alto nível desenvolvida em Python, executada na plataforma TensorFlow. Esta é desenvolvida com o foco em permitir uma experimentação rápida uma parte do ecossistema de TensorFlow que cobre todas as etapas do fluxo de trabalho de aprendizagem automática, desde a gestão dos dados até ao treino dos hiperparâmetros e soluções de implementação.

- Comparação

Para se comparar as ferramentas mencionadas anteriormente construiu-se a Tabela 13 de acordo com as seguintes caraterísticas, linguagem de programação, custo da utilização e funcionalidades implementadas.

Tabela 13 Comparação de *frameworks*

Caraterísticas / Frameworks	Linguagem de Programação	Custo Gratuito	Funcionalidades
Scikit-learn	Python, C, C++	✓	Pré-processamento, Classificação, Regressão, Agrupamento, Seleção do modelo, Reduzir dimensionalidade
TensorFlow	Python, C++	✓	Redes Neuronais e outras tarefas com aprendizagem automática, Criar um *dataset*
Keras.io	Python	✓	API para redes neuronais

Uma diferença notória nestas ferramentas diz respeito às funcionalidades. No caso do *Scikit-learn* apresenta como funcionalidades: o pré-processamento de dados, algoritmos de aprendizagem automática de classificação / regressão / agrupamento, selecionar o modelo e reduzir a

[48] https://www.tensorflow.org/
[49] https://keras.io/

dimensionalidade. Keras.io é uma biblioteca que se foca no desenvolvimento de redes neuronais, assim como Tensorflow, que para além do desenvolvimento de redes neuronais, também realiza outras tarefas de aprendizagem automática.

Tendo em conta a tabela anterior, a *framework* Keras.io pode ser considerada como uma biblioteca irrelevante para o desenvolvimento do projeto, pois apenas se direciona para a construção de redes neuronais.

3.4.3 Framework Web

Por último, descrevem-se três *frameworks* desenvolvidas em Python para construir *web services*, nomeadamente: Django, Flask e CherryPy.

- Django

O Django é uma *framework web* alto nível gratuita e *open-source* desenvolvida em Python, tem como principais vantagens o facto de ser conhecida, fácil e rápida de utilizar e não requer quaisquer conhecimentos de aplicação. Para além disso, contém uma forte documentação de forma organizada e ainda fornece diferentes tutorais *web* para iniciantes (Foundation, 2005).

- Flask

O Flask é uma *microframework web* gratuita e *open-source* desenvolvido com base nas bibliotecas WSGI Werkzeug e Jinja2. Esta *framework* contém menos funcionalidades incorporadas mas também é fácil e rápida de desenvolver, como também apresenta um tutorial elaborado de como desenvolver estas aplicações *web* (Flask, 2010).

- CherryPy

O CherryPy é uma *framework web* orientada a objetos desenvolvido na linguagem Python, isto é, o código base é menos desenvolvido, mas também requer um menor tempo de aplicação. Com a evolução esta *framework*, atualmente é muito rápida, estável e aplicada na construção de diferentes sites desde os mais simples aos mais complexos (CherryPy, 2001).

- Comparação

Para determinar a ferramenta que pode melhor se adequa ao projeto, construiu-se a Tabela 14 de acordo com as seguintes caraterísticas: desempenho, boa documentação, fácil aprendizagem, REST API e conhecimento.

Tabela 14 Comparação de *frameworks web*

Caraterísticas / Frameworks	Desempenho	Boa Documentação	Fácil Aprendizagem	REST API	Conhecimento
Django	Bom	✓	✓		
Flask	Excelente	✓	✓	✓	✓
CherryPy	Razoável			✓	

Das três ferramentas apresentadas o Flask é o mais adequado para projetos pequenos e bastante eficiente, mas também é uma ferramenta que contém uma *REST API* integrada, excelente

documentação, um desempenho mais rápido e com algum conhecimento adquirido ao longo do mestrado.

3.5 Sumário

Neste capítulo apresentou-se o processo de *text mining* que consiste nas etapas de pré-processamento, transformação do texto, seleção de caraterísticas, métodos de *text mining* e avaliação dos resultados obtidos. Para além da descrição do processo, foram apresentadas diferentes abordagens para a deteção de notícias falsas, tendo em conta os dados de entrada, os algoritmos de aprendizagem e o índice de credibilidade obtido. Estes foram divididos em dois grupos, sendo estes, o grupo de métodos e processos que classificam as notícias em duas classes e aqueles que fazem o mesmo num maior número de classes. Por fim, foram apresentadas as várias tecnologias existentes e uma comparação das mesmas, para averiguar quais se adequam ao desenvolvimento do projeto.

4 Análise de Valor

A análise de valor é um processo sistemático, formal e organizado de análise e avaliação, tem como objetivo obter um conjunto de funções necessárias num projeto com o menor custo possível (Rich and Holweg, 2000). Assim, pode-se afirmar que a análise de valor consiste num processo de gestão que requer um planeamento, controlo e coordenação. Esta análise é feita a um produto/serviço para atender aos requisitos requeridos pelos clientes. Este requisito funcional deve passar por um processo de revisão que deve incluir uma compreensão da finalidade do produto (Rich and Holweg, 2000).

Conforme está ilustrado na Figura 36, o processo da análise de valor é constituído por cinco fases, sendo estas, orientação, identificação e análise da funcionalidade, criar alternativas, análise e avaliação, e por último, a implementação.

Figura 36 Processo da Análise de Valor

Para cada uma das fases anteriormente descritas pode ser aplicado diferentes técnicas, porém no desenvolvimento deste projeto apenas foram consideradas as etapas de orientação, identificação e análise da funcionalidade e a implementação.

4.1 Modelo NCD

Com o propósito de conceber um novo produto é necessário seguir algumas fases para o processo de inovação e de desenvolvimento. Assim surge o modelo *News Concept Development* (NCD) com o objetivo de cultivar ideias e conceitos. Conforme ilustra a Figura 37, este modelo encontra-se dividido

em três áreas principais: o motor, os cincos elementos-chave e os fatores que podem afetar todo o processo.

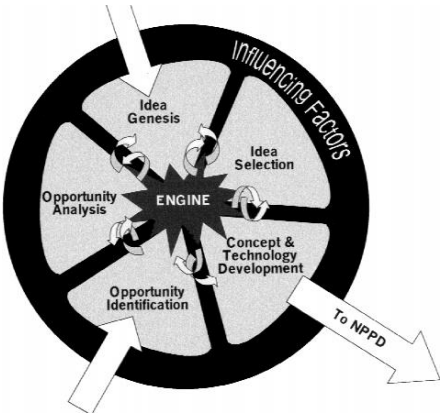


Figura 37 Modelo "The new concept development model"
(Koen et al., 2016)

Com base na figura anterior podemos dizer que o modelo contém cinco elementos sendo estes: a identificação da oportunidade, análise da oportunidade, geração e aperfeiçoamento de ideias, seleção de ideias, desenvolvimento do conceito e da tecnologia.

Este modelo inicia-se com a identificação de uma ou várias oportunidades ou então pela geração de uma ideia. De seguida, esta oportunidade ou ideias podem interagir com os outros elementos do modelo, interagindo também com os fatores de influência e impulsionados pelo motor, por último com este modelo obtém-se *New Product and Process Development (*NPPD), ou seja, Novo Produto e Desenvolvimento de Processos (Teza, Miguez, et al., 2013). De seguida, o foco é dirigido para os aspetos de identificação e análise de oportunidade.

4.1.1 Identificação de Oportunidade

Em conformidade com a Secção 2.2, o crescente desenvolvimento tecnológico e o aparecimento das redes sociais, têm proporcionado uma grande e rápida divulgação de notícias falsas, mas também a forma como a sociedade têm acesso a todo o tipo de informação. Hoje em dia, os jovens têm um contacto direto com as mais variadas plataformas digitais, sendo que existe uma preocupação maior tendo em conta obtém grande parte da informação através das mesmas e a maior parte das vezes de forma completamente descontextualizada.

Segundo um estudo feito pela Comissão Europeia, no ano de 2016, apenas 57% dos utilizadores, recorreram aos agregadores de notícias, redes sociais e motores de busca para ler notícias (*European Data Portal | European Data Portal*, 2016). No que diz respeito, à sociedade mais jovem, um terço dos inquiridos com idades entre os 18-24 anos afirmam que a sua principal fonte de notícias são as redes sociais (*European Data Portal | European Data Portal*, 2016). Assim, de acordo com este estudo pode-se depreender que o uso das redes sociais como fonte de informação está em crescente expansão e torna-se necessário incutir aos mesmos uma atenção sobre a informação que é partilhada na mesma. Neste sentido, é necessário criarem-se ferramentas capazes de ajudar os mais novos a avaliar a veracidade das notícias que lêem. Estas devem fazer uma contextualização das mesmas e de uma

avaliação correta das fontes que estão associadas, bem como das próprias notícias provenientes destas fontes.

Tendo em conta todo o estudo efetuado no estado de arte, verificou-se a existência de um projeto que desenvolveu um modelo para a verificação de uma notícia em português europeu. No entanto, este modelo apresenta algumas limitações no que diz respeito às fontes de informação que utiliza. Neste contexto, é relevante o desenvolvimento de uma ferramenta mais completa que permita, por um lado, determinar o índice de credibilidade de uma determinada notícia utilizando fontes de informação diversas e, por outro lado, garantir um aumento de expressividade utilizando uma classificação não binária.

4.1.2 Análise de Oportunidade

Com este projeto, em termos de oportunidades, a MOG pretende ampliar as suas competências na área de *machine learning*, de *data science*, da criação de plataformas de gestão de *media*, bem como aumentar a sua visibilidade no setor da educação.

Na ótica do consumidor final, apesar de existirem um conjunto de projetos e soluções dedicadas à análise de credibilidade, estes contêm dois problemas: ou não suportam notícias em língua portuguesa ou então não utilizam inteligência artificial. Neste contexto, pretende-se oferecer ao cliente final um sistema capaz de avaliar a confiabilidade de um *site* e do seu conteúdo noticioso com o enfoque na língua portuguesa. Este sistema deve aplicar algoritmos de *machine learning* para avaliar o grau de credibilidade dos mesmos em diferentes níveis.

4.2 Proposta de Valor

A proposta de valor pode ser construída na perspetiva do produto, na do produtor e, por último, na do investidor. Esta consiste numa afirmação através da qual se pretende demonstrar a correspondência entre as necessidades dos consumidores e os benefícios que resultam de utilização de um determinado produto (Twin, 2020).

Neste caso, a solução proposta deve atribuir um índice de credibilidade de conteúdos noticiosos em língua portuguesa para âmbito escolar. Assim, este projeto destina-se a alunos com diferentes níveis de ensino, mas também pode ser utilizado por professores e outras pessoas da escola. Com este trabalho, pretende-se dar uma forte contribuição para que os jovens escrevam textos noticiosos em português, de forma simples e contextualizada, com base numa análise da credibilidade das fontes de informação e respetivas nas notícias. O *Elevator Pitch* foi construído com o propósito de representar os valores e benefícios deste projeto, conforme se pode ver na Figura 38.

For target customer Jovens Estudantes
who statement of the need or opportunity Precisam de escrever textos noticiosos de qualidade
the product or service Serviço web de "Índice de credibilidade de conteúdos notícios em âmbito escolar"
is a product or service category Um serviço de avaliação da credibilidade
that quantified statement of most compelling benefit (not features) provided to the customer Pode ser integrada posteriormente num motor de busca
unlike competitors and competitive alternatives statement of primary differentiation of the product or service Automação do índice de credibilidade
our statement of primary differentiation of the product or service Apresenta a notícia e o seu respetivo índice de credibilidade através da Inteligência Artificial

Figura 38 *Elevator Pitch*

4.3 Diagrama FAST

O desenvolvimento do diagrama *Function Analysis System Technique* (FAST) é um processo de pensamento criativo que apoia a comunicação entre os membros de uma equipa. Com este diagrama pretende-se que exista: uma compreensão do desenvolvimento do projeto; identificação de funções em falta; definição, simplificação e clareza no problema; organizar e compreender as relações entre as funções; identificação da função básica do projeto/produto; melhorar a comunicação e o consenso; mas também estimular a criatividade (Dannana, 2020).

Este diagrama deve responder às seguintes questões: Como é que se consegue esta função?, Porque se faz esta função?, Quando se faz esta função, que outras funções se devem fazer? (Dannana, 2020). Desta forma, o diagrama é composto por uma função que é expandida num conjunto de duas instruções, "como" (no sentido da esquerda para a direita) e "porque" (no sentido da direita para a esquerda) para alcançar o produto final.
Este é também composto por vários passos, nomeadamente: Expandir as funções nas direções de "Como" e "Porquê"; Desenvolver ao longo da direção "Como", da esquerda para a direita, questionando sempre "Como é que esta função é alcançada?"; Testar a lógica na direção do "Porque", da direita para a esquerda, fazendo a seguinte questão: "Porque é que esta função está a ser desempenhada?"; Quando a lógica não funciona, devem ser identificadas funções que se encontrem em falta ou redundantes ou até alterar a ordem; Quando se encontrar funções que acontecem ao mesmo tempo, a pergunta que deve surgir deve ser: "Quando é que a função está concluída?"; "As funções que se encontram do lado esquerdo do diagrama o que está a ser "; as funções da direita descrevem como estão a ser realizadas (Dannana, 2020).

Como o projeto desta dissertação se concentra no desenvolvimento de um sistema, tornou-se relevante desenvolver este diagrama FAST, que descrever todas as funções do mesmo, conforme ilustra a Figura 39.

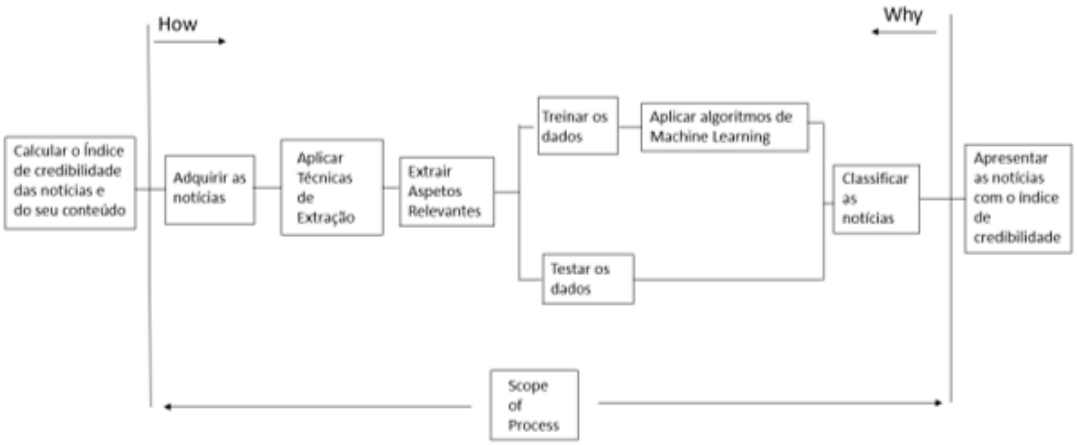


Figura 39 Diagrama FAST

Com base na figura anterior, podemos verificar que do lado esquerdo, encontra-se exposta a função que está a ser feita e é considerada a de ordem superior - “Calcular o índice de credibilidade das notícias e do seu conteúdo”, a função básica - “Adquirir as notícias” e como funções secundárias foram identificadas: “Aplicar técnicas de extração”; “Extrair aspetos relevantes”, “Treinar os dados”, “Testar os dados”, “Aplicar algoritmos de machine learning” e “Classificar as notícias”. A “função assumida” diz respeito ao “Apresentar as notícias com o índice de credibilidade”.

4.4 Sumário

Em suma, este capítulo resume-se a uma descrição de todo o processo de análise de valor com base na identificação e análise da oportunidade do desenvolvimento do sistema, mas também na apresentação da proposta de valor que é responsável pela principal diferença no sistema e ainda na descrição do modelo FAST para definir, analisar e compreender as funções do sistema.

5 Análise de Requisitos e Design

Neste capítulo é feita uma análise dos requisitos e a apresentação do *design*. Na Secção 5.1 são apresentados os principais interessados no desenvolvimento desta solução. Na Secção 5.2 e Secção 5.3 descrevem-se os requisitos funcionais e os requisitos não funcionais do sistema a desenvolver. Na Secção 5.4 é abordada a arquitetura de alto-nível da solução, um diagrama representativo do *deploy* da solução, o diagrama de componentes, o diagrama BPMN e um diagrama de sequência. Por último a Secção 5.5 descreve um resumo de todo o capítulo abordado.

5.1 Stakeholders

Os *stakeholders*, ou partes interessadas, são entidades que influenciam e podem ser afetadas pelo sucesso do projeto, direta ou indiretamente (Sommerville, 2005). No decorrer da análise do projeto foram identificados os seguintes *stakeholders*: toda comunidade escolar (professores, alunos, entre outros). No entanto, esta ferramenta também pode ser implementada para os jornalistas, cidadãos comuns e *social media*.

5.2 Requisitos Funcionais

Os requisitos funcionais definem um conjunto de funções que um sistema de *software* (tarefas ou serviços) deve ser capaz de realizar. Estas funções contam com um conjunto de entradas/parâmetros, aplicação das tarefas neste conjunto, e o resultado obtido da função executada como parâmetro de saída (Sommerville, 2005).

Nesta Subsecção 5.2 são descritos os requisitos funcionais do sistema a desenvolver. Posto isto, foi proposto pela empresa o desenvolvimento de um modelo de aprendizagem automática que permitisse classificar uma notícia em língua portuguesa no que diz respeito à sua credibilidade. Posteriormente, esse modelo deve ser integrado num *web service*. Assim, em termos de requisitos funcionais o sistema deve permitir que o utilizador insira uma notícia e, a partir desta, seja capaz de devolver o respetivo índice de credibilidade, com uma escala de classificação não binária.

5.3 Requisitos Não Funcionais

Os requisitos não funcionais representam as restrições que não são especificamente fornecidas pelo sistema ao utilizador. Estes podem estar relacionados com as propriedades, restrições na implementação e outras caraterísticas do sistema (Eeles, 2004).

Assim, Robert Grady da Hewlett Packard construiu um sistema FURPS + que classifica os requisitos em categorias. Este acrónimo é dividido em dois, sendo que existe o FURPS que identifica várias categorias, como, a Funcionalidade, Usabilidade, Confiabilidade, Desempenho e Suporte. O "+" representa os outros como os requisitos, restrições de *design* e implementação (Design, Implementação, Interface e Físico).

- **Desempenho**

Os requisitos de desempenho envolvem o tempo de resposta ao sistema, tempo de recuperação ou o tempo que demora de iniciar o mesmo (Eeles, 2004). Neste projeto foram identificados:

  - O tempo de processamento de cada notícia deve ser inferior ou igual a 2 minutos;
  - O tempo de atribuição do índice de credibilidade deve ser inferior a 1 minuto.

- **Suporte**

Os requisitos de suporte é onde se especifica uma série de outros requisitos tais como testabilidade, adaptabilidade, manutenção, instalação, escalabilidade, localização e entre outros (Eeles, 2004). Como os requisitos de suporte foram identificados:

  - O sistema deve ser fácil de manter;
  - O sistema deve suportar várias fontes de notícias *online* e em língua portuguesa;
  - O sistema deve suportar diferentes formatos de estruturação da própria fonte da notícia.

5.4 Design

O *design* consiste na base do processo de desenvolvimento de *software* (Pressman, 2015). Nesta secção são apresentadas duas propostas de arquitetura de alto-nível da solução, um diagrama de componentes e, por último, um diagrama de BPMN.

5.4.1 Proposta de Design

Nesta Subsecção descrevem-se duas arquiteturas e um diagrama BPMN da solução que se pretende implementar.

5.4.1.1 Arquitetura de Alto-Nível

Com o propósito de clarificar a solução que se pretende desenvolver, foi desenhada uma arquitetura de alto-nível, tendo em conta os componentes principais do mesmo e as iterações entre eles. De seguida, apresenta-se a arquitetura representada por duas alternativas, conforme ilustra a Figura 40 e Figura 41.

✓ **Primeira Alternativa**

Na primeira solução, Figura 40, as notícias encontram-se agregadas em Feed RSS e API's e são extraídas de forma manual. Com os dados das mesmas é construído um *dataset* que vai ser anotado manualmente e de seguida são testados vários algoritmos de aprendizagem automática que são responsáveis pela criação do modelo ideal. Este modelo vai ser utilizado pelo *web service* para apresentar a notícia e o respetivo índice de credibilidade.

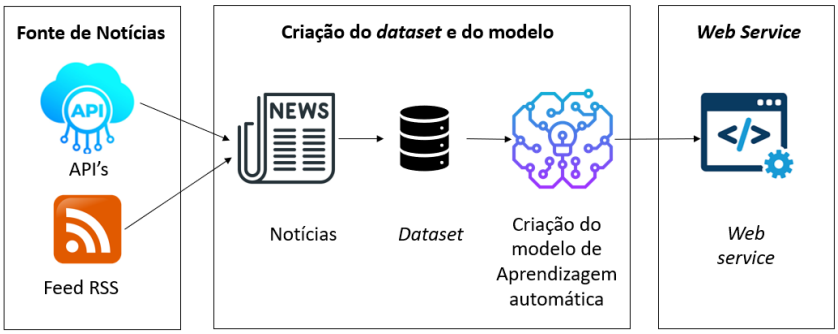


Figura 40 Arquitetura de Alto-Nível

✓ **Segunda Alternativa**

Uma alternativa para a arquitetura da solução apresentada anteriormente pode-se verificar na Figura 41. As notícias que se encontram agregadas nas Feed RSS e nas API's são extraídas com o recurso a um *web crawler (*é um *bot* que extrai informação dos diferentes *websites)*. Este é responsável por extrair a informação que vai ser utilizada na construção do *dataset,* de seguida são testados vários algoritmos de aprendizagem automática que são responsáveis pela criação do modelo ideal. Este modelo vai ser utilizado pelo *web service* para apresentar a notícia e o respetivo índice de credibilidade.

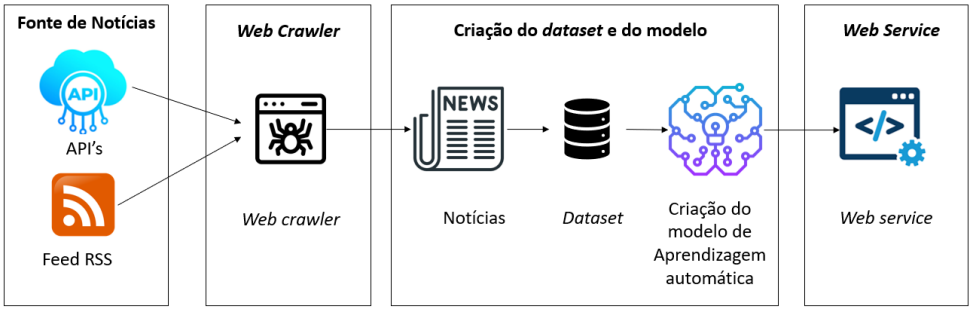


Figura 41 Arquitetura de Alto-Nível

✓ **Comparação**

A diferença visível entre a primeira arquitetura, Figura 40, e a segunda arquitetura, Figura 41, diz respeito à forma como as notícias são extraídas dos diferentes agregadores. Na primeira aquitetura as mesmas são extraídas de forma manual, já na segunda são extraídas de forma automática.

Tendo em conta que a arquitetura disposta anteriormente ainda não é da solução final e como a empresa pretende que a extração das notícias seja feita de forma manual, a arquitetura que se encontra representada na Figura 40 é a que melhor se adequa ao desenvolvimento da solução.

5.4.1.2 Diagrama Business Process Model and Notation

A construção do diagrama *Business Process Model and Notation* (BPMN) tornou-se relevante para apresentar as diferentes fases do projeto. A Figura 42 ilustra todo o processo de negócio do sistema e encontra-se dividido em duas *pools*, nomeadamente, agregadores de notícias e a empresa "MOG". Na *pool* representada pelo nome da empresa encontram-se duas *lanes* que dizem respeito à "Programadora" que é a pessoa responsável pelo desenvolvimento da solução e por essa razão todas as tarefas são consideradas manuais e a *lane* do "Sistema" que é responsável pela execução dos modelos de aprendizagem automática.

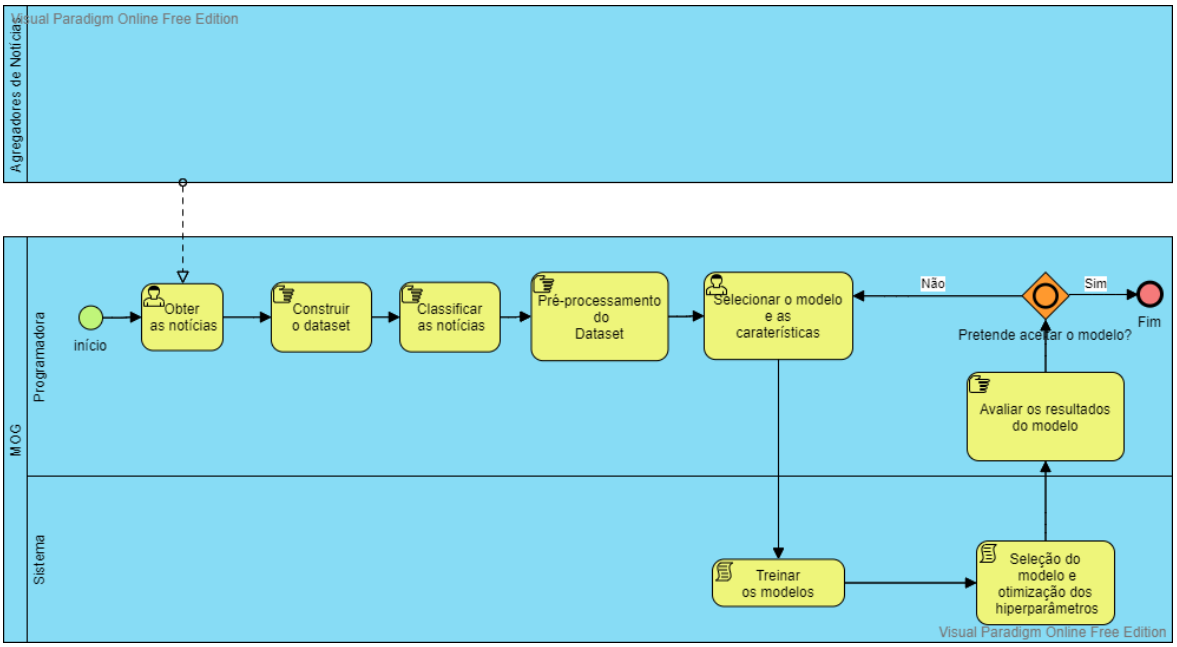


Figura 42 Diagrama do processo

As notícias que se encontram nos agregadores de notícias são extraídas pela *lane* "Programadora" onde se vai obter um conjunto com as notícias provenientes das várias fontes de informação. Posteriormente, as mesmas são classificadas em várias classes de veracidade, construindo um ficheiro CSV onde as notícias se encontram agrupadas, com várias colunas que indicam os seus parâmetros. Após essa tarefa, é feito um pré-processamento de todos os dados obtidos (limpeza, seleção de caraterísticas e entre outros). De seguida, as caraterísticas que são alimentadas no modelo e procura-se otimizar as mesmas e o modelo treinando os mesmos no sistema, quando os resultados do modelo chegam à avaliação do mesmo existem duas direções possíveis: se o modelo escolhido apresentar um bom resultado o processo é terminado; se não for volta-se ao processo de seleção do modelo e das características. Em suma, neste diagrama de processos encontram-se dispostas um conjunto de tarefas sequenciais que dependem umas das outras.

5.4.2 Design Final

Esta Subsecção é relativa à solução desenvolvida. De seguida, apresenta-se o fluxograma de todo o processo desenvolvido composto por dois diagramas de componentes e um diagrama de sequência que descreve a comunicação do *web service* com os modelos implementados, descritos no Capítulo 6.

5.4.2.1 Deploy da solução

A Figura 43 representa o diagrama de todo o processo desenvolvido, sendo que o treino do modelo foi construído na ferramenta Jupyter Notebook e o *deploy* do modelo no serviço *web* Flask foi implementado na ferramenta Microsoft Visual Studio 2019.

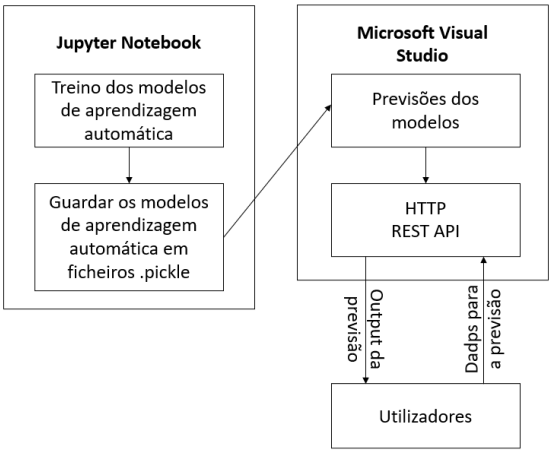


Figura 43 Fluxograma do processo de *deploy*

O utilizador introduz os dados das notícias solicitados na página *web* para o servidor utilizando um pedido HTTP POST. Os dados são recebidos em JSON e guardados no servidor, os mesmos são passados por dois modelos que foram armazenados treinados no ficheiro pickle. Após se determinarem as probabilidades das notícias dos mesmos, as mesmas são guardadas em ficheiros CSV para serem posteriormente corridas no terceiro modelo, onde se vai obter o resultado de cada previsão ao utilizador sob a forma de texto, mas também uma imagem de um semáforo com as cores respetivas com o texto.

5.4.2.2 Diagrama de Componentes

O diagrama de componentes mostra a relação entre os diferentes componentes de um sistema (Bell, 2004). De seguida, apresenta-se o diagrama de componentes representada por duas alternativas, conforme ilustra a Figura 44 e Figura 45.

- ✓ **Primeira Alternativa**

A Figura 44 representa o diagrama de componentes de toda a solução numa perspetiva de alto-nível, sendo que este diagrama tem como base três componentes principais: motor de busca; *web service.*

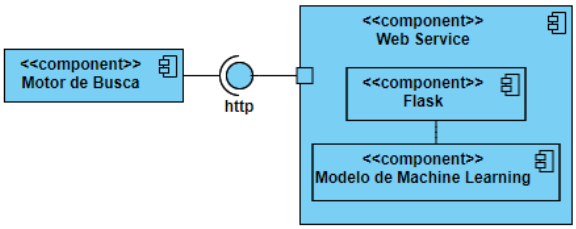


Figura 44 Diagrama de Componentes

O componente "Web service" contém dois componentes: o “Flask” e o “Modelo de Machine Learning”. O componente "Modelo de Machine Learning" é responsável pela construção do modelo de aprendizagem automática que vai é usado pelo componente "Web service" para validar toda a

implementação desenvolvida. Este é carrega o modelo de *machine learning* do componente mencionado anteriormente, disponibiliza uma API REST capaz de responder a pedidos HTTP em formato JSON, e também uma simples *interface* gráfica *web* para facilitar os testes ao modelo. A partir desta *interface web*, são então apresentadas as notícias e os respetivos índices de credibilidade a partir do modelo desenvolvido.

No futuro, tenciona-se que o componente "Motor de Busca" seja alimentado com os resultados que se apresentam no *web service* através de pedidos HTTP.

- ✓ **Segunda Alternativa**

A Figura 45 representa uma alternativa ao diagrama de componentes expostos anteriormente, este tem como base dois componentes principais: motor de busca e o *web service.* Porém, a presente dissertação vai se focar no desenvolvimento dos seguintes componentes: Flask e modelo de *machine learning*.

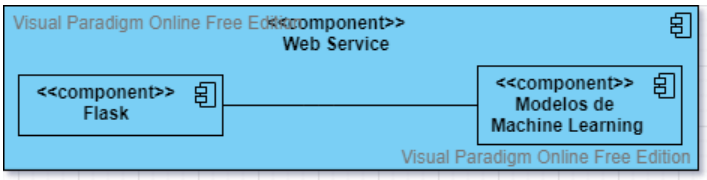


Figura 45 Diagrama de Componentes

O componente "Web service" contém dois componentes, nomeadamente o componente “Flask” e o componente "Modelo de Machine Learning". O componente "Modelo de Machine Learning" é responsável pela construção do modelo de aprendizagem automática que vai ser posteriormente usado pelo componente "Web service" para validar toda a implementação desenvolvida. Este é carrega o modelo de *machine learning* do componente mencionado anteriormente, disponibiliza uma API REST capaz de responder a pedidos HTTP em formato JSON, e também uma simples *interface* gráfica *web* para facilitar os testes ao modelo. A partir desta *interface web*, o Flask, são então apresentadas as notícias e os respetivos índices de credibilidade a partir do modelo desenvolvido.

- ✓ **Comparação**

Dos dois diagramas de componentes apresentados anteriormente, a Figura 45, é o que melhor se adequa à solução final. Tendo em conta que o modelo de aprendizagem automática, desenvolvido foi carregado no componente "Web service" através da biblioteca *pickle*. No entanto, a Figura 44 conta com a representação do “Motor de Busca” que futuramente pode vai necessário integrar.

5.4.2.3 Diagrama de Sequência

A Figura 46 representa o diagrama de sequência sobre a interação do utilizador com o *web service* construído no serviço *web* Python com base no modelo final alcançado.

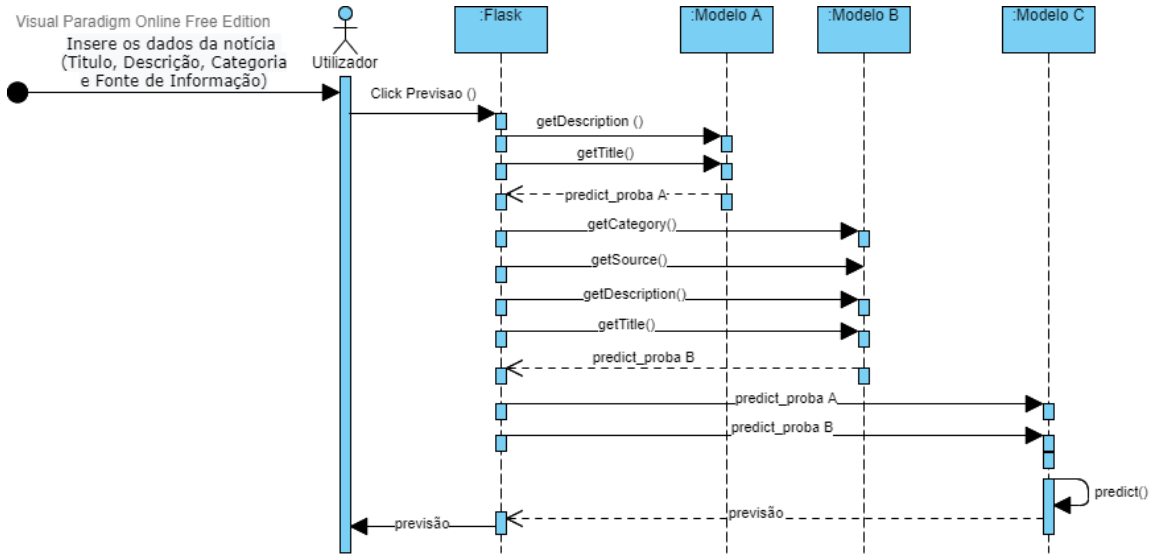


Figura 46 Diagrama de Sequência de interação entre o utilizador e o *web service*

O utilizador insere os dados da notícia nomeadamente, o título, a descrição, categoria e por último a fonte de informação. Após o utilizador clicar no botão "Verificar", o pedido é processado no modelo A, B e a previsão determinada no modelo C. Todos os modelos foram treinados anteriormente com os dados existentes no *dataset* e armazenados num ficheiro, esses modelos foram carregados no *web service*. O pedido feito ao serviço Flask vai enviar para o modelo A, a descrição e o título inseridos pelo utilizador necessário para determinar a previsão de cada classe do modelo A que é armazenada num ficheiro predict_probaA.csv. De seguida, todos os dados inseridos são utilizados para o modelo B efetuar a previsão de cada classe tendo em conta o modelo que já treinado com os outros dados, que depois foi armazenado num ficheiro predict_probaB.csv. Por último, os dois ficheiros com a probabilidade da classe 1 foram convergidos num só ficheiro, onde foi empregue o modelo C. Neste modelo é feita a previsão que futuramente é enviada ao utilizador final.

5.5 Sumário

Neste capítulo, apresentou-se a análise dos requisitos funcionais e não funcionais e a identificação das partes interessadas no desenvolvimento desta solução. Para além disto, foram expostos e descritos um conjunto de diagramas que apresenta a proposta de solução final e um conjunto de diagramas que descreve toda a implementação da solução.

6 Implementação da Solução

Neste capítulo expõem-se toda a abordagem para a resolução dos objetivos expostos na Secção 1.3. O mesmo apresenta todo o processo de construção e anotação do *dataset,* a análise exploratória dos dados e a *pipeline* implementada para o pré-processamento efetuado aos dados. Numa primeira abordagem, o *dataset* foi anotado em quatro classes, os resultados dos algoritmos não foram de todo significativos pois os critérios utilizados eram subjetivos. Neste sentido construiu-se a segunda abordagem que contou com a reanotação do mesmo em duas classes. Com o objetivo de validar o modelo final, implementou-se um *web service*. Toda a implementação foi realizada na linguagem de programação Python e as plataformas de suporte ao desenvolvimento da mesma foram o Jupyter Notebook e o Microsoft Visual Studio.

6.1 Dataset

No presente estado de arte, verificou-se a existência de um *dataset* em língua portuguesa construído com base em três *websites* de verificação de factos. Porém, o mesmo não foi utilizado na presente dissertação.

Como um dos objetivos desta dissertação era o desenvolvimento de um *dataset* em língua portuguesa com recurso a *sites* de notícias fidedignos e não fidedignos, tornou-se assim uma necessidade de construir o mesmo de raiz, tendo em conta os vários *sites* de notícias descritos anteriormente. Esta recolha de informação foi realizada de forma semi-automática e armazenada num *dataset* com vários parâmetros de entrada. De seguida, o mesmo foi anotado de forma manual em multiclasses.

6.1.1 Construção do Dataset

Para extrair a informação que se encontrava disponível nestes *sites,* utilizou-se a biblioteca *Beautiful Soup* e obteve-se desta forma o texto que se encontrava disposto nos elementos do HTML dos vários *sites* de notícias. O resultado de cada uma dessas extrações foi guardado num ficheiro CSV, com os seguintes parâmetros: título, URL, descrição, conteúdo, data de publicação, hora de publicação, autor,

categoria, ano, fonte de informação, proporcionando assim, um fácil acesso de consulta e alteração dos dados presentes no mesmo. Este *dataset* é composto por 708 notícias desde o ano 2017 ao ano 2021 presentes em diversos *sites,* fidedignos e não fidedignos. Em relação aos *sites* não fidedignos utilizou-se os que se encontram expostos no Anexo A - Figura 78.

6.1.2 Anotação do Dataset

No *dataset* foi adicionada uma coluna *Classification* que corresponde à anotação realizada de forma manual e relativa aos quatro critérios definidos. Estes foram assentes, segundo as ações descritas na Secção 2.3, que permitiram identificar se uma notícia é falsa ou verdadeira. Esta anotação foi feita em multiclasses, designadamente: 1 - Falso; 2 - Parcialmente Falso; 3 - Parcialmente Verdadeiro e 4 – Verdadeiro. A Tabela 15 relaciona a anotação do *dataset* em quatro classes com os respetivos critérios.

Tabela 15 Critérios e anotação do *dataset* em quatro classes

Critérios / **Anotação**	**Verificar o autor**	**Verificar no motor de busca**	**Verificar órgão de comunicação social**	**Websites de verificação de factos**
Falso			✓	✓
Parcialmente Falso	✓		✓	
Parcialmente Verdadeiro	✓	✓		
Verdadeiro	✓	✓	✓	✓

Na Tabela 15, a anotação da notícia em Falso corresponde ao facto de a notícia aparecer num único órgão de comunicação social, não apresentar autor e não aparecer no motor de busca, ou ter sido classificado por algum *website* de verificação de factos. A anotação da notícia em Parcialmente Falso diz respeito à notícia aparecer num único órgão de comunicação social, bem como apresentar o autor associado à notícia. A anotação da notícia em Parcialmente Verdadeiro corresponde à mesma ter associado um autor e aparecer nas pesquisas do motor de busca. Por último, a anotação da notícia em Verdadeiro tem de apresentar um autor considerado fidedigno, aparecer em diferentes meios de comunicação social, apresentar um órgão de comunicação social associado e, por fim estar presente nos *websites* de verificação de factos em Portugal (Polígrafo e Fact-Check Observador).

6.2 Análise Exploratória dos Dados

Posteriormente à fase de construção do *dataset*, partiu-se para uma análise exploratória das duas anotações (multiclasses e binária). Esta análise teve como objetivo resumir as caraterísticas do mesmo conjunto através de vários gráficos, visando assim, perceber a relação entre as variáveis e as palavras mais frequentes.

6.2.1 Classificação Multiclasses

Assim sendo, fez-se uma análise detalhada de todo o conjunto de dados que é composto por 708 notícias e 11 colunas, cada uma das variáveis do conjunto de dados foi analisada utilizando o *Exploratory Data Analysis* (EDA). Destas colunas, contando com a variável dependente do tipo ordinal denominada *Classification*, a Figura 79 presente no Anexo B expõe o conjunto de dados e as colunas representativas do mesmo.

As colunas correspondem ao título de uma notícia, ao URL da mesma, descrição, categoria, data de publicação, fonte de informação, ano, conteúdo, hora de publicação e, por último, a classificação. Destes 708 registos utilizou-se a função info() para verificar a existência de valores nulos.

```
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Title            708 non-null    object
 1   URL              708 non-null    object
 2   Description      704 non-null    object
 3   Category         707 non-null    object
 4   Publishing Date  704 non-null    object
 5   Source           708 non-null    object
 6   Year             703 non-null    float64
 7   Author           658 non-null    object
 8   Content          708 non-null    object
 9   Published Time   613 non-null    object
 10  Classification   708 non-null    int64
dtypes: float64(1), int64(1), object(9)
```

Figura 47 Função info ()

Da Figura 47 verificou-se a existência de 11 colunas, sem valores a nulo. Deste conjunto de dados, existem 9 variáveis que são do tipo objeto, uma de *float e* outra de inteiros. Após esta análise, utilizou-se a função head () para verificar a composição das 5 primeiras notícias e a função tail () que representa as 5 últimas notícias conforme é visível na Figura 48.

Title	URL	Description	Category	Publishing Date	Source	Year	Author	Content	Published Time	Classification
Fezes de morcego ajudam a decifrar o passado d...	https://zap.aeiou.pt/fezes-morcego-decifrar-pa...	Uma equipa de investigadores australianos usou...	Ciência e Saúde	14/04/2021	ZAP AEIOU	2021.0	Daniel Costa	Para perceber melhor como é que artefactos ant...	NaN	3
"É como fogo". Dezenas de pessoas morreram na ...	https://zap.aeiou.pt/pessoas-morreram-ingestao...	Depois de dezenas de pessoas morreram por into...	Mundo	14/04/2021	ZAP AEIOU	2021.0	Ana Isabel Moura	As lojas de bebidas clandestinas situavam-se s...	NaN	1
O custo da paz mundial é bastante inferior ao ...	https://zap.aeiou.pt/custo-paz-inferior-custo-...	Um recente relatório do Instituto de Economia ...	Economia	14/04/2021	ZAP AEIOU	2021.0	Daniel Costa	O Instituto de Economia e Paz estima que, em 2...	21:30	1
Vaticano vai discutir o celibato dos padres em...	https://zap.aeiou.pt/vaticano-discutir-celibat...	Será já no próximo ano que o Vaticano deverá d...	Mundo	14/04/2021	ZAP AEIOU	2021.0	ZAP	Sob o mote "Rumo a uma Teologia Fundamental do...	NaN	3
Para prevenir acidentes, China inaugura o prim...	https://zap.aeiou.pt/china-inaugura-o-primeiro...	Autoridades do norte da China inauguraram este...	Mundo	13/04/2021	ZAP AEIOU	2021.0	ZAP	O semáforo para camelos entrou em operação no ...	NaN	2

Figura 48 Cinco últimas notícias presentes no *dataset*

Desta análise, procedeu-se à construção de vários gráficos tendo em conta as anotações efetuadas ao *dataset*. Numa primeira fase, analisaram-se os dados que foram anotados em multiclasses e partiu-se para a exploração dos mesmos com diferentes gráficos. Assim, utilizou-se a biblioteca Seaborn que é

construída no matplotlib. Começou-se por visualizar a distribuição da variável contínua denominada ano conforme é visível na Figura 49.

Figura 49 Frequência de notícias por ano

Na Figura 49 apresentam-se os anos relativos às notícias. As mesmas variam entre o ano 2017 e o ano 2021, porém, também existiram notícias classificadas no ano 0, ou seja, notícias que não tinham nenhum ano associado. Portanto, deste gráfico pode-se concluir que a maior parte das notícias expostas no *dataset* são do ano 2021, com uma frequência de 644 notícias. O ano 2020 contou com 34 notícias, o ano 2019 conteve 14, o ano 0 conta com 12, o ano 2017 teve 3 e, por último, o ano 2018 com apenas 1.

Relativamente às duas variáveis categóricas presentes no conjunto de dados utilizado, *Category* e *Source*, tomou-se como um dos principais objetivos efetuar o EDA sobre este tipo de atributos para a identificação de valores únicos e a respetiva contagem. Na Figura 50 representam-se as 35 categorias possíveis para a variável que faz referência às categorias, *Category*.

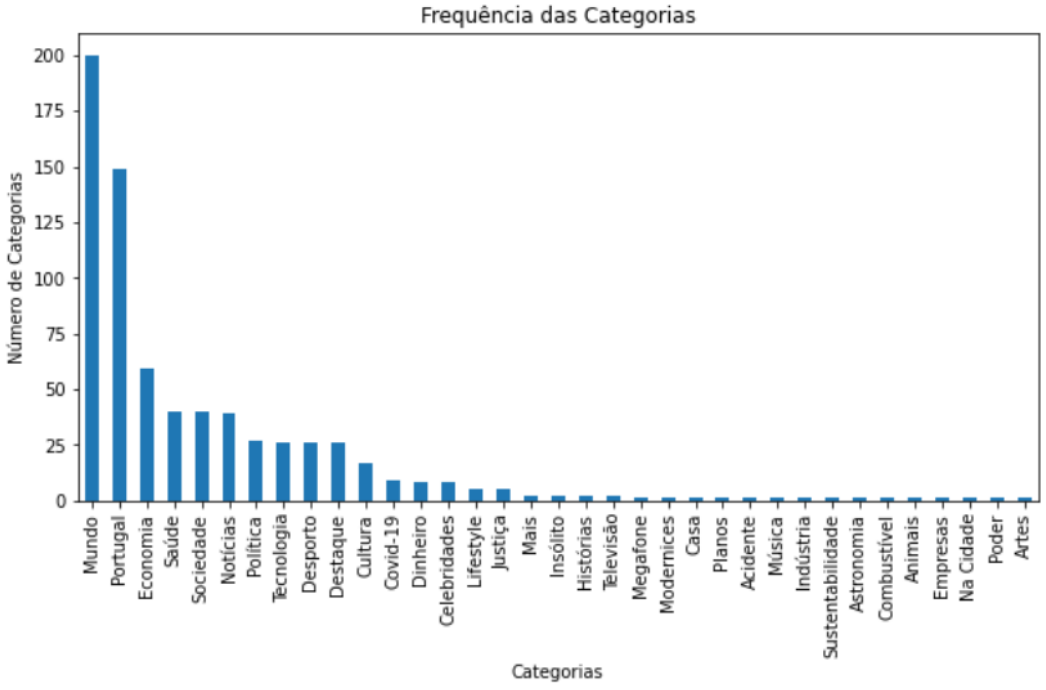


Figura 50 Frequência das Categorias

Segundo a Figura 50, foi possível verificar a elevada quantidade de amostras para a condição: “Mundo” – 200 e “Portugal” – 149 e uma quantidade residual de amostras para as categorias (“Mais”, “Na Cidade”, “Insólito”, “Televisão”, “Histórias”, “Casa”, “Artes”, “Megafone”, “Sustentabilidade”, “Planos”, “Poder”, “Acidente”, “Música”, “Modernices”, “Astronomia”, “Animais”, “Empresas”, “Indústria”, “Opinião” e “Combustível”) que apresentam apenas e só 1 notícia.

Das categorias expostas anteriormente, realizou-se uma análise mais detalhada sobre a relação das mesmas com a anotação feita em multiclasses. Na Figura 51 apresentam-se três das categorias (“Mundo”, “Nacional” e “Ciência e Saúde”), mencionadas anteriormente.

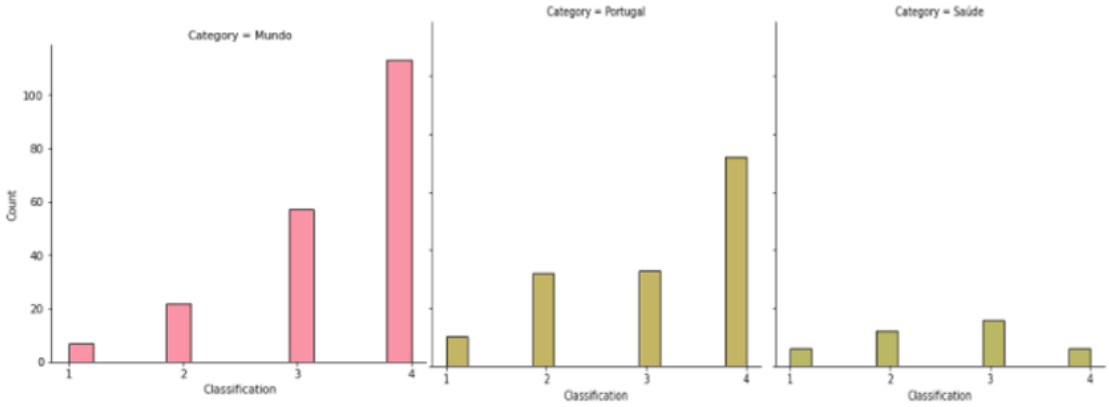


Figura 51 Relação das categorias e da classificação multiclasses

Relativamente à Figura 51, na categoria “Mundo” verificou-se que existe um maior número de notícias anotadas na classe 4 – Verdadeiro, contando com um valor superior a 100 notícias. Em relação à categoria “Portugal”, também se verificou a existência de um maior número de notícias classificadas na classe 4 – Verdadeiro. Por último, na categoria “Saúde”, as notícias foram maioritariamente anotadas nas classes 2 e 3, Parcialmente Falso e Parcialmente Verdadeiro, respetivamente.

No que diz respeito à variável *Source,* construiu-se o gráfico presente na Figura 52 para verificar o número de notícias associado a cada uma das 35 fontes de informação.

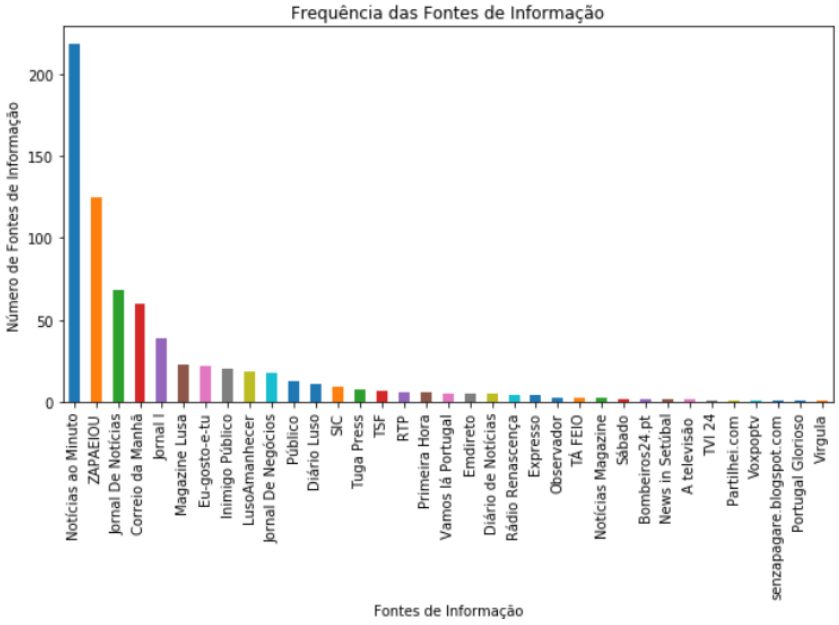


Figura 52 Frequência das Fontes de Informação

As quatro fontes de informação com maior número de notícias associados são: as "Notícias ao Minuto" – 218, "ZAPAEIOU" – 125, "Jornal de Notícias" – 68 e o "Correio da Manhã" – 60. Outro aspeto relevante de se verificar é a relação entre as várias fontes de informação e a anotação do conjunto de dados em multiclasses. Na Figura 53 pode-se verificar um excerto das análises efetuadas, nomeadamente da relação entre as fontes de informação denominadas "Eu-gosto-e-tu" e a "ZAP AEIOU".

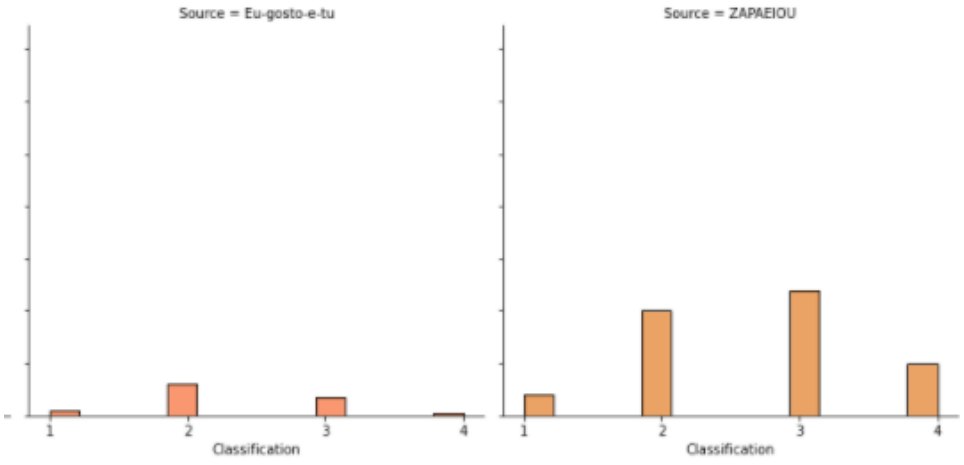


Figura 53 Relação das fontes de informação e da classificação multiclasses

Da Figura 53 confere-se que a fonte de informação denominada "Eu-gosto-e-tu" contém um maior número de notícias anotadas na classe 2 – Parcialmente Falso. Já a fonte de informação que se encontra no gráfico do lado direito designada "ZAP AEIOU", a maior parte das notícias estão anotadas entre a classe 2 – Parcialmente Falso (40) e 3 – Parcialmente Verdadeiro (48), ainda que contendo valores significativos para as restantes classes.

Relativamente às variáveis textuais presentes no conjunto de dados utilizado, *Title*, *Description* e *Content*, um dos principais objetivos de efetuar a EDA sobre este tipo de atributos foi identificar as diferentes palavras e a respetiva contagem. Assim, utilizou-se a biblioteca nltk.probability com o intuito de obter as palavras mais comuns. Na Figura 54 e Figura 55, observam-se dois gráficos com as 40 palavras mais frequentes presentes na variável "conteúdo" das notícias.

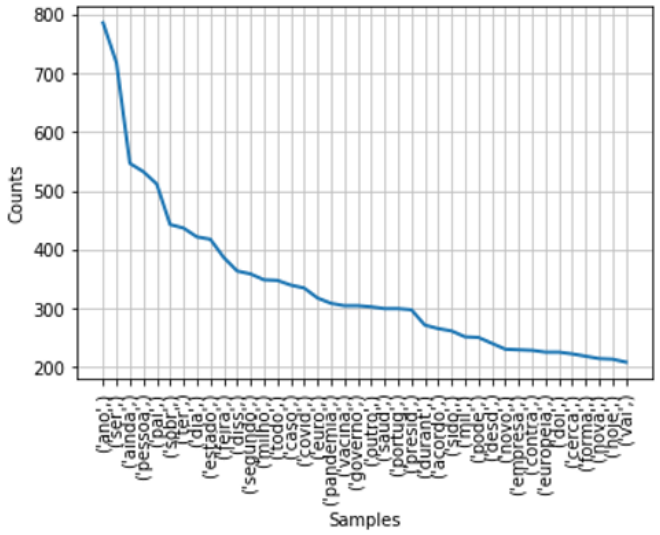


Figura 54 Frequência dos 40 *tokens* mais comuns no *Content*

Na Figura 54 apresenta-se apenas uma palavra (*Uni-grams*), sendo que a palavra mais comum do *conteúdo* é o "ano" com uma frequência de 786 e a palavra "vai" com uma frequência de 200. De seguida, avançou-se na complexidade da análise e averiguou-se quais eram as palavras mais frequentes com o tamanho de duas palavras (*Bi-grams*).

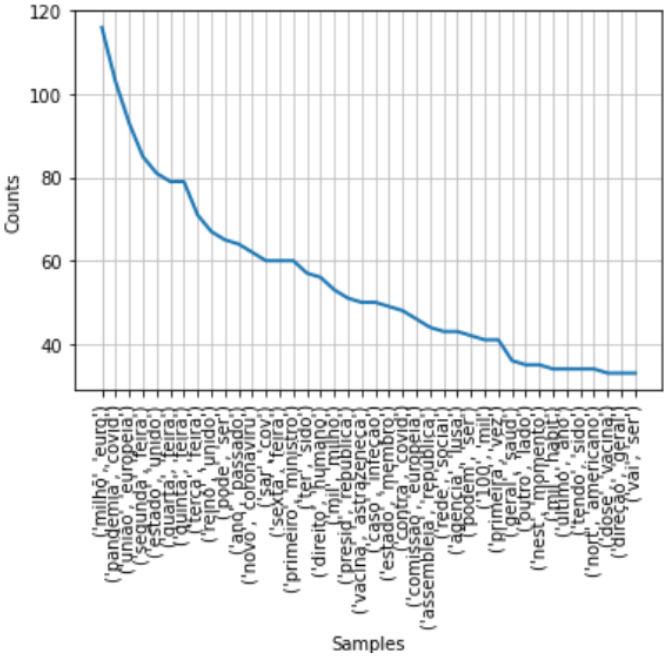


Figura 55 Frequência de *Bi-grams* da variável *Content*

Como é possível verificar na Figura 55 as palavras [“milhõ”, “euro”] tomam lugar em 116 notícias e as palavras [“vai”, “ser”] em 33 notícias. Em relação à variável *título* procedeu-se à mesma análise, a Figura 56 representa as 40 palavras mais comuns nessa variável.

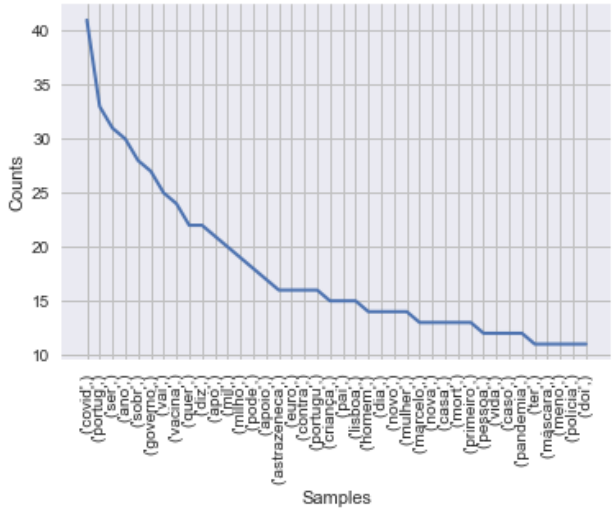


Figura 56 Frequência dos 40 *tokens* mais comuns no título (*Uni-grams)*

A palavra mais comum do *título* é “covid” com uma frequência de 41 notícias em contraste com a palavra “doi”, com uma frequência de 11 notícias. De seguida, analisou-se o tamanho de duas palavras da variável denominada *título*.

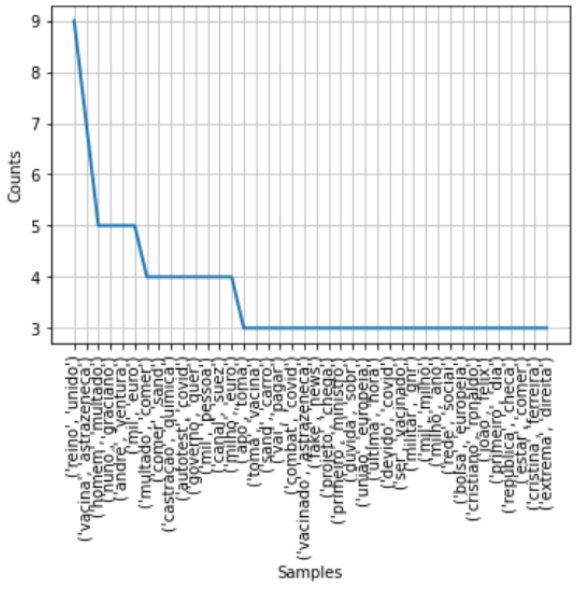


Figura 57 Frequência dos 40 *tokens* mais comuns no título (*Bi-grams)*

Na Figura 57 verificou-se que as duas palavras mais comuns nessa variável são o [“reino”, “unido”] com uma frequência de 9 notícias.

6.2.2 Classificação Binária

Nesta Secção 6.2.2 aborda-se uma análise exploratória da classificação do *dataset* em apenas duas classes: 1 – Falso e 4 – Verdadeiro. Tal como na abordagem multiclasses, primeiro foram usadas as funções head() e tail() para verificar as 5 notícias no início e no fim do *dataset*.

Tendo em conta que o número de notícias que apresentam os seguintes atributos: *ano*, *categoria* e *fonte de informação* é igual na abordagem multiclasses e na abordagem binária, procedeu-se para a utilização, novamente, da biblioteca Seaborn na elaboração da Figura 58 e Figura 59. A Figura 58 ilustra a relação entre as categorias "Mundo", "Notícias", "Saúde" e a classificação binária.

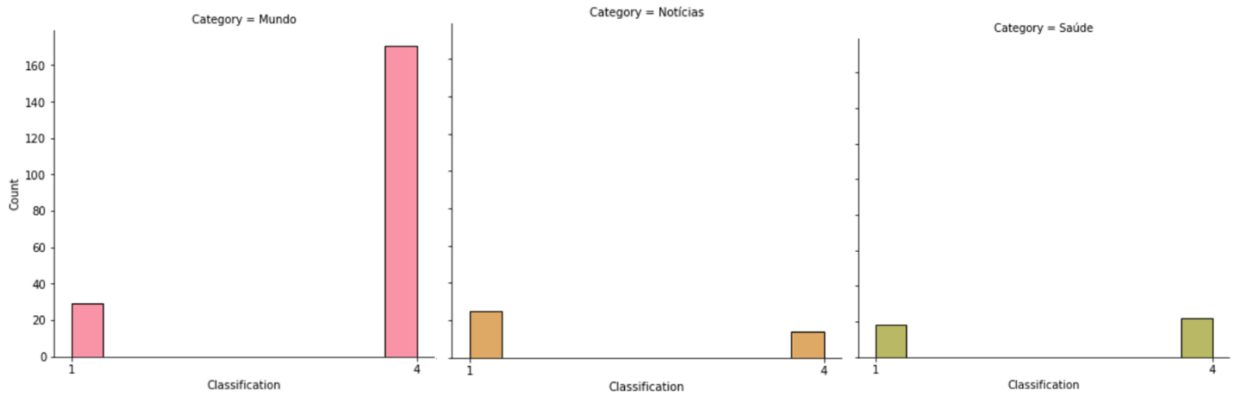


Figura 58 Relação entre as categorias e classificação binária

Na categoria "Mundo" apresentam-se 175 notícias classificadas como Verdadeiro e cerca de 30 anotadas como Falso. Na categoria "Notícias" mostram-se 25 notícias classificadas como Falso e abaixo das 25 notícias classificadas como Verdadeiro. Por último, a categoria "Saúde" contou com 20 notícias classificadas tanto como Verdadeiro, como Falso. Na Figura 59 pode-se verificar um excerto das análises efetuadas, nomeadamente da relação entre as fontes de informação denominadas "Eu-gosto-e-tu" e a "ZAP AEIOU".

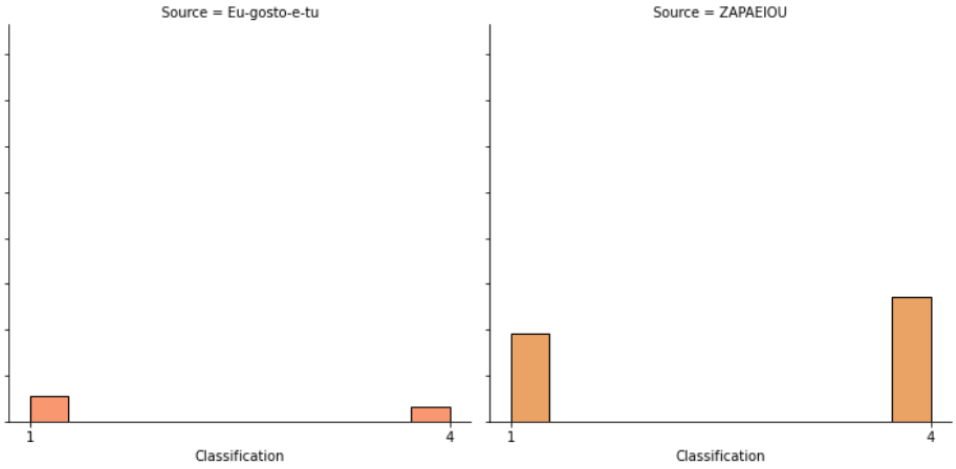


Figura 59 Relação das fontes de informação e a classificação binária

Após esta análise, avançou-se com uma apreciação às variáveis textuais que se apresentam ao longo do conjunto de dados, nomeadamente, *Title*, *Description*, *Content* e *Title+Description*. Assim, analisou-

se o número de palavras presentes em cada variável no *dataFrame* Verdadeiro e Falso, conforme ilustra a Tabela 16.

Tabela 16 Análise das variáveis textuais

Variável	Total de Palavras	Média	Classificação
Content	103256	189.80	Verdadeira
Title	3956	7.27	Verdadeira
Description	7575	13.92	Verdadeira
Title + Description	11531	21.19	Verdadeira
Content	28852	165.81	Falsa
Title	1383	7.94	Falsa
Description	2791	16.04	Falsa
Title + Description	4174	23.98	Falsa

Numa análise mais pormenorizada, averiguou-se quais são as palavras mais frequentes nas notícias classificadas como Falsas e Verdadeiras da variável *Title + Description*. Neste caso, a Figura 60 contém duas formas de representar as 40 palavras mais frequentes com o tamanho de uma só palavra (*Uni-grams*) anotadas como Falsa.

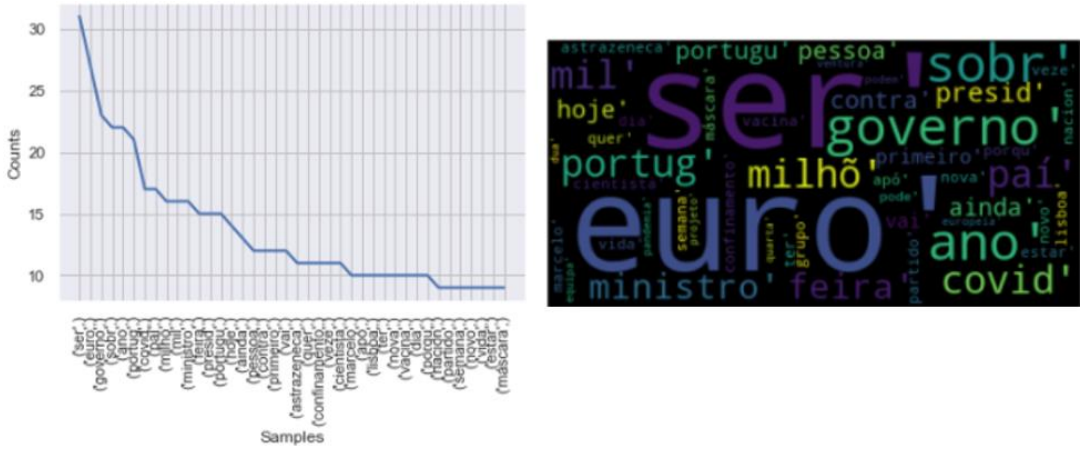


Figura 60 Frequência dos 40 *tokens* mais comuns no *dataFrame* falso

Relativamente ao *dataFrame* Falso, as palavras “ser”,” euro” e ”governo” são as palavras com uma presença mais frequente, sendo que a contagem das mesmas corresponde aos seguintes valores: 31, 27 e 23, respetivamente. Por outro lado, a Figura 61 apresenta novamente duas formas de representar as 40 palavras mais frequentes da variável *Title + Description* com o tamanho de uma só palavra (*Uni-grams*) das notícias que se apresentam como Verdadeiras.

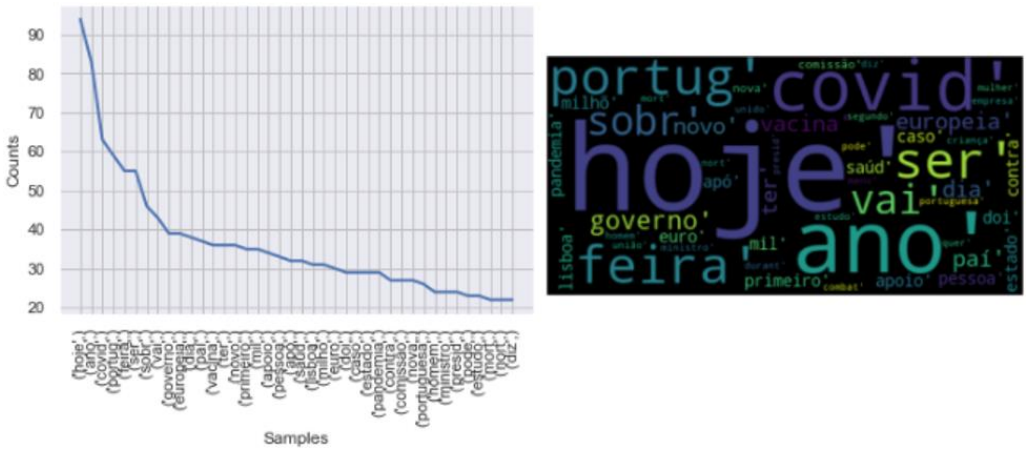


Figura 61 Frequência dos 40 *tokens* mais comuns no *dataFrame* verdadeiro

No *dataFrame* Verdadeiro, as palavras "hoje", "ano" e "covid" são as palavras com uma presença mais frequente, sendo a contagem das mesmas correspondentemente aos seguintes valores: 94, 84 e 63.

6.3 Pré-processamento dos Dados

Nesta etapa de pré-processamento dos parâmetros presentes no conjunto de dados desenvolvido na Secção 6.1. foi utilizada a biblioteca NLTK. Inicialmente, aplicaram-se aos parâmetros textuais: título, conteúdo e descrição da notícia em língua portuguesa, as seguintes técnicas de *Text Mining*:

- Remoção de todos os sinais de pontuação;
- Transformação de todo o texto presente nas variáveis em minúsculas;
- Transformação do texto em *tokens (Uni-grams* e *Bi-grams)*;
- Procedeu-se à verificação e remoção das *stop-words* em língua portuguesa;
- Transformação das palavras na sua raiz com a biblioteca *Porter Stemmer* para eliminar a redundância de uma determinada palavra.

Relativamente aos parâmetros categóricos, nomeadamente: *categoria* e *fonte de informação* utilizou-se o *Label Encoder* e *One-Hot Encoding*. O *Label Encoder* atribui a cada *categoria* um determinado número e o *One-Hot Encoding* transforma as *fontes de informação* numa matriz, onde são atribuídos valores binários. Com o processo de pré-processamento concluído, utilizou-se um conjunto de técnicas responsáveis pela transformação do texto em valores numéricos, sendo estas nomeadamente o TF-IDF e o *CountVectorizer*, conforme foi descrito no estado de arte Secção 3.1.2. De seguida, apresentam-se as duas abordagens e as suas experiências efetuadas ao longo de todo o projeto.

6.4 Abordagem Multiclasses

Na abordagem multiclasses considerou-se um conjunto de 708 notícias anotadas em quatro classes conforme foi descrito na Secção 6.1 . Na Figura 62 verifica-se que a classe 1 - Falso tem atribuídas 55

notícias; a classe 2 – Parcialmente Falso conta com 113 notícias; 198 notícias na classe 3 - Parcialmente Verdadeiro e, por último, a classe 4 – Verdadeiro conta com 342 notícias.

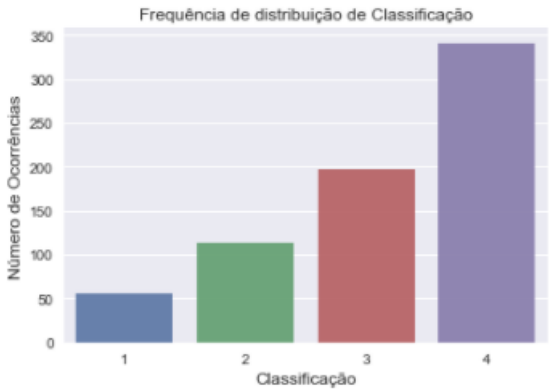


Figura 62 Frequência de distribuição de classificação das notícias

Na Figura 62 pode-se verificar que existe um desequilíbrio nítido em relação à classificação das notícias, ou seja, existe um maior número de notícias classificadas na classe 4 – Verdadeiro comparado com a classe 1 – Falso. Deste ponto de vista, o modelo pode ter tido um desempenho melhor para a classe 4 comparativamente com a classe 1.

6.4.1 Experiência 1

Numa primeira experiência, utilizou-se como parâmetro de entrada para o modelo a variável do *conteúdo*, sendo que esta passou pelas várias etapas de pré-processamento. Foi aplicada a técnica de transformação do texto *Count-Vectorizer* com diferentes *features* com um único *token* (*Uni-grams*) associado. Foi assim construída uma matriz cujas colunas são representadas por uma única palavra.

O conjunto de dados foi dividido em treino e teste através da função *train_test_split()*. Consideraram-se 80% dos dados para treino e o 20% dos mesmos para teste, o que corresponde a 566 amostras para treino e 142 amostras para teste. Esta divisão proporcionou o preenchimento das seguintes variáveis: *x_train* e *x_test* com o conteúdo do texto (variáveis independentes), *y_train* e *y_test* com as *labels* de classificação (variáveis dependentes). O treino dos modelos contou com três algoritmos de classificação, designadamente, o *Multinomial Naive Bayes*, o *Random Forest Classifier* e o *Gradient Boosting Classifier*.

A Tabela 17 apresenta uma comparação dos três algoritmos, sendo que esta reflete uma matriz composta por 186 colunas. Utilizou-se a função *classification_report* da biblioteca sklearn para expor as principais métricas de avaliação de cada algoritmo.

Tabela 17 Comparação dos algoritmos com 186 colunas

Features	**Algoritmo**	***Accuracy***	***Precision* Classe = 1**	***Precision* Classe = 2**	***Precision* Classe = 3**	***Precision* Classe = 4**
186	*Multinomial Naive Bayes*	42%	0%	14%	20%	51%
	Gradient Boosting Classifier	37%	0%	21%	25%	47%
	Random Forest Classifier	37%	0%	21%	25%	47%

Para avaliar este conjunto foram analisadas as métricas de *accuracy* e *precision* de cada classe. Em relação à métrica *accuracy,* o melhor algoritmo foi o *Multinomial Naive Bayes* com uma *performance* de 42%. Este processo foi repetido e o número de caraterísticas foi alterado para 467 colunas, com o intuito de verificar se a *accuracy* ou a precisão de cada classe sofria alguma alteração.

Tabela 18 Comparação dos algoritmos com 467 colunas

Features	**Algoritmo**	***Accuracy***	***Precision* Classe = 1**	***Precision* Classe = 2**	***Precision* Classe = 3**	***Precision* Classe = 4**
467	*Multinomial Naive Bayes*	36%	0%	5%	29%	55%
	Gradient Boosting Classifier	40%	0%	14%	29%	48%
	Random Forest Classifier	44%	0%	40%	18%	48%

Com a Tabela 19 percebeu-se que o aumento do número de colunas teve um impacto negativo nas classificações, sendo que o algoritmo *Multinomial Naive Bayes* foi o que apresentou uma pior *performance* e a melhor *performance* pertenceu ao *Random Forest Classifier* com 44% de *accuracy*. Por último, nesta experiência, aumentou-se o número de colunas resultando num valor total de 931 colunas.

Tabela 19 Comparação dos algoritmos com 931 colunas

Features	**Algoritmo**	***Accuracy***	***Precision* Classe = 1**	***Precision* Classe = 2**	***Precision* Classe = 3**	***Precision* Classe = 4**
931	*Multinomial Naive Bayes*	36%	0%	5%	29%	55%
	Gradient Boosting Classifier	40%	0%	14%	29%	48%
	Random Forest Classifier	44%	0%	40%	18%	48%

Da Tabela 19, verifica-se que a *performance* do algoritmo é exatamente igual à obtida com 467 colunas, ou seja, o mesmo não conseguiu aprender mais conforme era expetável.

Em suma, pode-se verificar que os resultados presentes nesta experiência conseguiram alcançar uma *accuracy* superior a 44% e os mesmos não conseguiram atribuir notícias à classe 1. Sendo assim, partiu-se para uma segunda experiência, com os seguintes parâmetros de entrada para o modelo: o *título+descrição.*

6.4.2 Experiência 2

Numa segunda experiência, juntou-se o atributo título com a descrição e contou-se como parâmetro de entrada para o modelo esse mesmo conjunto. De seguida, o mesmo passou pelas etapas de pré-processamento descritas anteriormente, Secção 6.3. Nesta experiência, utilizou-se a técnica de transformação do texto *Count-Vectorizer* com um único *token* (*Uni-grams*) de entrada para o modelo. A Tabela 20 representa o número de colunas e a frequência de cada palavra com um peso superior ou igual a 2, 3, 4 e 5.

Tabela 20 Frequência de palavras e colunas da técnica *Count-Vectorizer*

Frequência de uma palavra	**Colunas**
X>=2	2556
X>=3	1490
X>=4	1031
X>=5	750

Ao longo desta experiência foram testadas as diferentes colunas, tendo sempre em conta a divisão do conjunto de dados em dois conjuntos de dados, treino e teste, através da função *train_test_split()*. A representação do conjunto de dados de treino 80% (566 notícias) e 20% de teste (142 notícias). Esta divisão proporcionou o preenchimento das variáveis: *x_train* e *x_test* com o título e a descrição (variáveis independentes), *y_train* e *y_test* com as *labels* de classificação (variáveis dependentes). O conjunto de algoritmos de classificação utilizados foram o *Multinomial Naive Bayes*, *Random Forest Classifier* e o *Gradient Boosting Classifier*.

Esta fase começou com as palavras que aparecem mais do que duas vezes numa notícia o que resultou em 2556 colunas, conforme ilustra a Tabela 21.

Tabela 21 Comparação dos modelos com 2556 colunas

Features	**Algoritmo**	***Accuracy***	***Precision* Classe = 1**	***Precision* Classe = 2**	***Precision* Classe = 3**	***Precision* Classe = 4**
2556	*Multinomial Naive Bayes*	39%	14%	11%	34%	61%
	Gradient Boosting Classifier	43%	67%	9%	24%	51%
	Random Forest Classifier	43%	0%	25%	21%	50%

Em conformidade com a Tabela 21, a maioria das avaliações comparativas realizadas após o treino dos dados focou-se nas seguintes métricas de avaliação: *accuracy*, *precision* da classe = 1 e da classe = 4. Diante isto, o algoritmo *Gradient Boosting Classifier* demonstrou ser o melhor em termos de *accuracy*, pois alcançou um valor de 43%, bem como uma *precision* da classe = 1 de 67% e, apesar da *precision* da classe = 4 não apresentar uma boa *performance*, continua a ser o melhor.

A segunda fase da experiência corresponde à Tabela 22 que apresenta as palavras que aparecem mais do que três numa notícia, o que resulta em 1490 colunas e as métricas de avaliação para cada um dos algoritmos utilizados.

Tabela 22 Comparação dos modelos com 1490 colunas

Features	**Algoritmo**	***Accuracy***	***Precision* Classe = 1**	***Precision* Classe = 2**	***Precision* Classe = 3**	***Precision* Classe = 4**
1490	*Multinomial Naive Bayes*	43%	15%	17%	33%	66%
	Gradient Boosting Classifier	42%	10%	8%	29%	57%
	Random Forest Classifier	45%	0%	30%	26%	55%

De acordo a Tabela 22, a maioria das avaliações comparativas realizadas após o treino dos dados focou-se nas seguintes métricas de avaliação: *accuracy*, *precision* da classe = 1 e *precision* da classe = 4. Por um lado, verificou-se que o algoritmo *Random Forest Classifier* é o melhor em termos de *accuracy*, pois alcança um valor de 45%. Por outro lado, se observarmos o algoritmo *Multinomial Naive Bayes* apesar da *accuracy* ser de 43%, ou seja, inferior à do algoritmo *Random Forest Classifier*, a *precision* da classe = 1 consegue alcançar os 15% e em termos da *precision* da classe = 4 também atinge um valor melhor do que o exposto anteriormente de 66%.

A terceira fase é composta pelas palavras que aparecem mais do que quatro vezes, resulta em 1031 colunas como entrada para o modelo e as métricas de avaliação a ter em conta para cada um dos algoritmos utilizados, conforme ilustra a Tabela 23.

Tabela 23 Comparação dos modelos com 1031 colunas

Features	**Algoritmo**	***Accuracy***	***Precision* Classe = 1**	***Precision* Classe = 2**	***Precision* Classe = 3**	***Precision* Classe = 4**
1031	*Multinomial Naive Bayes*	44%	10%	22%	32%	63%
	Gradient Boosting Classifier	44%	17%	18%	29%	59%
	Random Forest Classifier	44%	0%	33%	23%	55%

A avaliação dos resultados presentes na Tabela 23, focou-se nas seguintes métricas de avaliação: *accuracy*, *precision* da classe = 1 e da classe = 4. Assim sendo, todos os algoritmos relevam uma *accuracy* de 44%, apesar disso o algoritmo *Gradient Boosting Classifier* é relativamente melhor tendo em conta que se evidência na *precision* da classe 1 que atinge um valor de 17% e a *precision* da classe 4 é de 59%.

A quarta fase corresponde às palavras que aparecem mais do que cinco vezes que transformou numa matriz com 750 colunas de entrada para o modelo e as métricas de avaliação a ter em conta para cada um dos algoritmos utilizados, exibe-se na Tabela 24.

Tabela 24 Comparação dos modelos com 750 colunas

Features	**Algoritmo**	***Accuracy***	***Precision* Classe = 1**	***Precision* Classe = 2**	***Precision* Classe = 3**	***Precision* Classe = 4**
750	*Multinomial Naive Bayes*	47%	14%	23%	37%	65%
	Gradient Boosting Classifier	38%	22%	0%	29%	49%
	Random Forest Classifier	37%	7%	24%	31%	51%

De acordo com a Tabela 24, verificou-se que esta experiência resultou numa % superior a 0% na *precision* na classe = 1. O algoritmo *Multinomial Naive Bayes* foi o melhor com uma *accuracy* de 47%, a *precision* da classe 1 foi de 14%, a *precision* da classe 2 foi de 23%, a *precision* da classe 3 foi de 37% e, por último, a *precision* da classe 4 foi de 65%. Desta experiência pode-se depreender que a redução do número de colunas para a entrada para o modelo tem um peso é significativo nas diferentes métricas de avaliação dos algoritmos.

6.4.3 Experiência 3

Na terceira experiência, contou-se como parâmetro de entrada para o modelo as seguintes variáveis: o *título+descrição*, mais a *fonte* de informação. Para a variável *fonte* de informação utilizou-se o *One Hot Encoding* e para variável textual foi aplicada a técnica de transformação do texto, *Count-Vectorizer* com apenas uma palavra (*Uni-grams*).

Nesta experiência utilizaram-se as colunas apresentadas na Tabela 20 mais a fonte de informação, assim as fases desta experiência contaram com os seguintes números de colunas: 2557, 1491, 1032 e 751. O conjunto de dados foi dividido em 80% para treino (566 notícias) e 20% para teste (142 notícias), através da função *train_test_split()*. Esta divisão resultou no preenchimento das variáveis: *x_train* e *x_test* com *título+descrição* e *fonte* de informação (variáveis independentes). Nas variáveis *y_train* e y*_test* associam-se as *labels* de classificação (variáveis dependentes). Os três algoritmos de classificação utilizados foram o *Multinomial Naive Bayes*, o *Random Forest Classifier* e o *Gradient Boosting Classifier*.

A primeira fase começou com as palavras que aparecem mais do que duas vezes numa notícia e a fonte de informação associada, o que resultou em 2557 colunas, conforme ilustra a Tabela 25.

Tabela 25 Comparação dos modelos com 2557 colunas

Features	**Algoritmo**	***Accuracy***	***Precision* Classe = 1**	***Precision* Classe = 2**	***Precision* Classe = 3**	***Precision* Classe = 4**
2557	*Multinomial Naive Bayes*	43%	12%	7%	33%	62%
	Gradient Boosting Classifier	48%	50%	11%	35%	58%
	Random Forest Classifier	46%	67%	0%	23%	54%

Face aos resultados expostos na Tabela 25, o algoritmo *Gradient Boosting Classifier* foi o melhor algoritmo em termos de *accuracy* com 48%, em relação à *precision*: a classe 1 conseguiu atingir 50%, a classe 2 foi de 11%, a classe 3 foi de 35% e, por último, a classe 4 com uma *precision* de 58%.

A segunda fase contou com as palavras que aparecem mais do que três vezes e a *fonte* de informação, assim o número de colunas resultou em 1491, as métricas de avaliação e os algoritmos utilizados foram os mesmos descritos anteriormente, conforme é possível observar na Tabela 26.

Tabela 26 Comparação dos modelos com 1491 colunas

Features	**Algoritmo**	***Accuracy***	***Precision* Classe = 1**	***Precision* Classe = 2**	***Precision* Classe = 3**	***Precision* Classe = 4**
1491	*Multinomial Naive Bayes*	45%	22%	14%	34%	67%
	Gradient Boosting Classifier	42%	13%	30%	26%	59%
	Random Forest Classifier	45%	0%	0%	30%	57%

Na Tabela 26, o algoritmo *Multinomial Naive Bayes* é o que apresenta melhores resultados em termos de *accuracy* com 45% e em relação às precisões de cada classe foi o que obteve melhores classificações (classe 1 = 22%, classe 2 = 14%, classe 3 = 34% e classe 4 = 67%).

A terceira fase exibe-se na Tabela 27 com as palavras que aparecem mais do que quatro vezes e a fonte de informação, o que resultou numa matriz com 1032 colunas como entrada para o modelo.

Tabela 27 Comparação dos modelos com 1032 colunas

Features	**Algoritmo**	***Accuracy***	***Precision* Classe = 1**	***Precision* Classe = 2**	***Precision* Classe = 3**	***Precision* Classe = 4**
1032	*Multinomial Naive Bayes*	46%	18%	19%	36%	66%
	Gradient Boosting Classifier	48%	11%	32%	34%	61%
	Random Forest Classifier	46%	0%	27%	34%	56%

Desta etapa verificou-se que o algoritmo *Multinomial Naive Bayes* foi o que apresentou melhores resultados, com 46% ao nível da *accuracy* e a precisão de cada uma das classes foi de 18%, 19%, 36% e 66%.

Na última fase, foram consideradas as palavras que aparecem mais do que cinco vezes e as fontes de informação, o que resultou numa matriz com 751 colunas como entrada para o modelo e as métricas de avaliação a ter em conta, exibem-se na Tabela 28.

Tabela 28 Comparação dos modelos com 751 colunas

Features	**Algoritmo**	***Accuracy***	***Precision* Classe = 1**	***Precision* Classe = 2**	***Precision* Classe = 3**	***Precision* Classe = 4**
751	*Multinomial Naive Bayes*	47%	36%	13%	33%	66%
	Gradient Boosting Classifier	45%	11%	20%	38%	57%
	Random Forest Classifier	39%	0%	41%	16%	50%

De acordo com a Tabela 28, verificou-se que algoritmo *Multinomial Naive Bayes* foi novamente o melhor, com uma *accuracy* de 47% a *precision* da classe 1 foi de 11%, a *precision* da classe 2 foi de 20%, a *precision* da classe 3 foi de 38% e, por último, a *precision* da classe 4 foi de 66%.

6.4.4 Experiência 4

Numa última experiência desta abordagem, utilizou-se como parâmetro de entrada para o modelo as variáveis do *título + descrição*, mais as fontes de informação. A variável textual passou pelas várias etapas de pré-processamento descritas anteriormente e a variável que contém a fonte de informação passou pela função replace() para obter as 35 fontes de informação, demonstrada na Secção 6.2. A variável textual passou pela técnica de transformação de texto TF-IDF com apenas um único *token* associado (*Uni-grams*) e a variável *fonte* passou pelo *Label Encoder.*

A Tabela 29 expõe as várias fases desta experiência, sendo visível o número de colunas e a frequência de cada palavra com um peso superior ou igual a 3, 4, 6 e 7.

Tabela 29 Frequência e as colunas da técnica TF-IDF

Frequência de uma palavra	**Colunas**
X>=3	82
X>=4	19
X>=6	14
X>=7	9

Ao longo das várias experiências foram utilizadas as várias colunas presentes na Tabela 29. O conjunto de dados foi dividido em 80% para treino (566 notícias) e 20% para teste (142 notícias), através da função *train_test_split()*. Com esta divisão, as variáveis: *x_train* e *x_test* são preenchidas com o título, a descrição e as fontes de informação (variáveis independentes), *y_train* e *y_test* são ocupadas com as *labels* de classificação (variáveis dependentes). Nesta experiência utilizaram-se os seguintes três algoritmos: o *Multinomial Naive Bayes*, *Random Forest Classifier* e o *Gradient Boosting Classifier*.

Na primeira etapa foram utilizadas 83 colunas (82 colunas da variável textual e 1 coluna da variável fontes de informação) e quatro métricas para avaliar a *performance* dos algoritmos expostos anteriormente, conforme ilustra a Tabela 30.

Tabela 30 Comparação dos modelos com 83 colunas

Features	**Algoritmo**	***Accuracy***	***Precision* Classe = 1**	***Precision* Classe = 2**	***Precision* Classe = 3**	***Precision* Classe = 4**
83	*Multinomial Naive Bayes*	47%	0%	0%	0%	48%
	Gradient Boosting Classifier	46%	100%	39%	28%	57%
	Random Forest Classifier	40%	40%	23%	25%	55%

Em conformidade com a Tabela 30, o melhor algoritmo desta fase foi o *Gradient Boosting Classifier* com uma *accuracy* de 46%, em termos de *precision* da classe 1 alcançou o valor de 100% e a classe 4 obteve uma *precision* de 57%.

Na Tabela 31 é possível verificar a experiência com 20 colunas (19 da variável textual e 1 coluna da variável da fonte de informação) e os respetivos resultados dos três algoritmos descritos anteriormente.

Tabela 31 Comparação dos modelos com 20 colunas

Features	**Algoritmo**	***Accuracy***	***Precision* Classe = 1**	***Precision* Classe = 2**	***Precision* Classe = 3**	***Precision* Classe = 4**
20	*Multinomial Naive Bayes*	47%	0%	0%	0%	48%
	Gradient Boosting Classifier	44%	45%	40%	26%	56%
	Random Forest Classifier	44%	22%	29%	33%	55%

Neste caso, o melhor algoritmo foi o *Gradient Boosting Classifier* com um *accuracy* de 44% e a precisão para cada classe, também foi o que apresentou melhores resultados. Sendo que a classe 1 apresentou uma precisão de 44%, a classe 2 com 40%, a classe 3 com 26% e a classe 4 com 56%.

A Tabela 32 expõe a terceira etapa desta experiência, onde foram utilizadas 15 colunas (14 colunas da variável textual e 1 coluna da variável fonte de informação) e expostas as diferentes métricas de avaliação dos diferentes algoritmos abordados anteriormente.

Tabela 32 Comparação dos modelos com 15 colunas

Features	**Algoritmo**	***Accuracy***	***Precision* Classe = 1**	***Precision* Classe = 2**	***Precision* Classe = 3**	***Precision* Classe = 4**
15	*Multinomial Naive Bayes*	47%	0%	0%	0%	48%
	Gradient Boosting Classifier	53%	56%	47%	34%	63%
	Random Forest Classifier	56%	43%	52%	40%	67%

De acordo com a Tabela 32, o melhor algoritmo foi o *Random Forest* com uma *accuracy* de 56% e as métricas de precisão de todas as classes obteve resultados medianos. A precisão da classe 1 contou com 43%, a precisão da classe 2 com 52%, a precisão da classe 3 com 40% e a precisão da classe 4 com 67%.

Por último, nesta experiência foram utilizadas as palavras que aparecem mais do que 7 vezes numa notícia, o que corresponde a 9 colunas, e ainda a fonte de informação como é visível na Tabela 33.

Tabela 33 Comparação dos modelos com 10 colunas

Features	**Algoritmo**	***Accuracy***	***Precision* Classe = 1**	***Precision* Classe = 2**	***Precision* Classe = 3**	***Precision* Classe = 4**
10	*Multinomial Naive Bayes*	47%	0%	0%	0%	48%
	Gradient Boosting Classifier	52%	60%	41%	30%	62%
	Random Forest Classifier	54%	67%	42%	37%	65%

Desta iteração da experiência, verificou-se que o melhor algoritmo foi o *Random Forest Classifier* com uma *accuracy* de 54% e as suas previsões tanto da classe 1 como da 4 conseguiram atingir valores superiores a 50%, o que correspondeu a uma melhor previsão da classificação das notícias.

Face ao exposto, os resultados apresentados com a função TF-IDF apresentou melhores resultados do que a função *Count Vectorizer*. O TF-IDF fornece a importância de cada palavra num determinado texto permitindo a remoção das palavras que são menos relevantes para a análise proporcionando assim, uma construção do modelo menos complexa.

Com esta abordagem verificou-se que apesar de o modelo conseguir adquirir conhecimento, este era de uma forma incorreta por causa dos critérios utilizados para a anotação do *dataset*. Neste sentido, proporcionou-se a reanotação do mesmo em duas classes e assim uma nova abordagem.

6.5 Abordagem Binária

Em conformidade com os resultados apresentados na abordagem multiclasses, verificou-se um problema de subjetividade dos critérios em relação à anotação do *dataset* abordado anteriormente. Nesse sentido, procedeu-se a uma reanotação do mesmo de forma manual em apenas duas classes 1 – Falsa e 4 – Verdadeira, conforme se pode verificar na Figura 63.

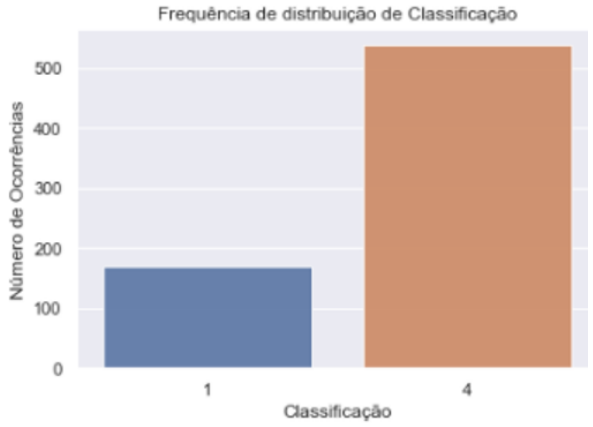


Figura 63 Frequência de distribuição de classificação das notícias em duas classes

Da Figura 63, 170 notícias pertencem à classe Falsa que é uma junção das notícias que se encontram dispostas na classe 1 e 2 da abordagem multiclasses e 538 notícias correspondem à classe Verdadeira e corresponde a uma junção das classes 3 e 4. De seguida, apresentam-se as diferentes experiências elaboradas nesta abordagem.

6.5.1 Experiência 1

Na primeira experiência, consideraram-se como parâmetros de entrada para o modelo as seguintes variáveis: *título + descrição* e a *fonte* de informação, foram pré-processadas através das etapas de pré-processamento descritas anteriormente. De seguida, o *título + descrição* passou pela técnica de transformação de texto TF-IDF e a fonte de informação contou com o *One-Hot Encoding.*

Esta experiência contou com duas fases, a primeira com 767 colunas e a segunda com 598 colunas relativas ao conteúdo textual. O conjunto de dados foi dividido em 80% para treino (566 notícias) e 20% para teste (142), consoante a função *train_test_split()*. Nesta divisão, foram preenchidas as seguintes variáveis: *x_train* e *x_test* com as variáveis definidas anteriormente (variáveis independentes) e *y_train* e *y_test* com as *labels* de classificação (variáveis dependentes). A Tabela 34 apresenta os dois algoritmos utilizados, bem como as métricas de avaliação dos mesmos.

Tabela 34 Comparação dos algoritmos de máximo de caraterísticas textuais 767

Max Features	**Algoritmo**	***Accuracy***	***Precision* Classe = 1**	***Precision* Classe = 4**	**ROC AUC = 1**	**ROC AUC = 4**	**Matriz Confusão**	
767	*Multinomial Naive Bayes*	83%	76%	85%	-	16%	16	18
							2	102
	Gradient Boosting Classifier	67%	20%	75%	-	14%	19	15
							6	102

Nesta fase, o melhor algoritmo foi *Multinomial Naive Bayes* com 83% de *accuracy*. No entanto, na matriz confusão dos dois algoritmos verificou-se que ambos falharam completamente nas classificações realizadas. Em relação à métrica ROC AUC verificou-se que o algoritmo apenas conseguiu aprender sobre a classe 4 – Verdadeiro.

A segunda fase contou exatamente com os mesmos dois algoritmos e com as métricas expostas anteriormente, mas com menos caraterísticas textuais, mais precisamente 598, conforme se verifica na Tabela 35.

Tabela 35 Comparação dos algoritmos de máximo de caraterísticas textuais 598

Max Features	**Algoritmo**	***Accuracy***	***Precision* Classe = 1**	***Precision* Classe = 4**	**ROC AUC = 1**	**ROC AUC = 4**	**Matriz Confusão**	
598	*Multinomial Naive Bayes*	83%	76%	85%	-	16%	16	18
							5	102
	Gradient Boosting Classifier	84%	77%	86%	-	14%	17	17
							5	102

Neste caso, o algoritmo *Gradient Boosting Classifier* foi o melhor em termos de *accuracy* com um valor de 84%. Porém ambas as matrizes confusões demonstram que ambos os algoritmos não conseguem classificar de forma correta as notícias classificadas como Falsas. Em relação à métrica ROC AUC, ambos os algoritmos só conseguem aprender sobre a classificação 4 – Verdadeiro.

6.5.2 Experiência 2

Numa segunda experiência, utilizou-se como parâmetro de entrada para o modelo a variável *título + descrição* como uma única variável. A mesma passou pelas várias etapas de pré-processamento descritas anteriormente. Após o pré-processamento, as notícias foram agrupadas pelas classes Verdadeiro e Falso, tendo sido utilizada a técnica de transformação de texto TF-IDF para cada uma dessas classes. Os mesmos convergiram novamente para um único *dataFrame* com 114 colunas de 1 único *token* (*Uni-grams*) que, posteriormente, foi dividido em dois conjuntos de dados: 80% para treino (566 notícias) e 20% para teste (142 notícias), segundo a função *train_test_split()*. Nesta divisão foram preenchidas as seguintes variáveis: *x_train* e *x_test* com a variável textual definida anteriormente (variáveis independentes) e *y_train* e *y_test* com as *labels* de classificação (variáveis dependentes).

De seguida, apresenta-se a Tabela 36 com os quatro algoritmos utilizados (*Multinomial Naive Bayes*, *Gradient Boosting Classifier*, *Random Forest Classifier* e o *XGBoosting Classifier*) nesta experiência.

Tabela 36 Comparação dos algoritmos de máximo de caraterísticas 114

Max Features	**Algoritmo**	***Accuracy***	***Precision* Classe = 1**	***Precision* Classe = 4**	**ROC AUC**	**Matriz Confusão**	
114	*Multinomial Naive Bayes*	76%	0%	76%	65%	0	34
						0	108
	Gradient Boosting Classifier	67%	20%	75%	82%	4	30
						16	92
	Random Forest Classifier	73%	38%	78%	92%	5	29
						8	100
	XGBoosting Classifier	70%	29%	77%	90%	6	28
						15	93

Nesta experiência, o melhor algoritmo foi o *XGBoosting* com 70% de *accuracy*, no entanto, na matriz confusão verifica-se que o algoritmo falhou completamente nas classificações realizadas. A mesma contou com apenas 6 notícias classificadas de forma correta na classe 1 e 28 notícias foram classificadas de forma incorreta na classe 4 quando deviam ter sido categorizadas na classe 1. Assim como, na classe 4, 15 notícias foram identificadas de forma errada e apenas acertou em 93 notícias.

6.5.3 Experiência 3

Numa terceira experiência, utilizou-se como parâmetro de entrada o *título + descrição.* A mesma passou pelas várias etapas de pré-processamento. Agrupou-se o conjunto de notícias através da variável classificação, Verdadeira e Falsa, e para cada conjunto utilizou-se a transformação de texto, TF-IDF. De seguida, dividiu-se o conjunto de dados em dois conjuntos de dados: 80% para treino (566

notícias) e 20% para teste (142 notícias), tendo em conta a função *train_test_split()*. Esta divisão resultou no preenchimento das variáveis: *x_train* e *x_test* com a variável definida anteriormente (variáveis independentes), e *y_train* e *y_test* com as *labels* de classificação (variáveis dependentes). Para realizar o treino foram utilizados os seguintes algoritmos: o *Multinomial Naive Bayes*, o *Random Forest Classifier*, o *Gradient Boosting Classifier*, o *XGBoosting Classifier* e a Regressão Logística. Ao longo desta experiência e das posteriores, foi utilizada a função *Grid Search* para cada um dos algoritmos com o objetivo de encontrar os parâmetros que obtêm uma melhor *performance* na solução final.

A primeira fase exibe-se na Tabela 37, sendo que o *max_features* definido na função TF-IDF de cada uma das classes foi de 70 colunas.

Tabela 37 Comparação dos algoritmos de máximo de caraterísticas 70

Max Features	**Algoritmo**	***Accuracy***	***Precision* Classe = 1**	***Precision* Classe = 4**	**ROC AUC**	**Matriz Confusão**		***Cross-Validation* (K=5)**
70	*Multinomial Naive Bayes*	88%	100%	86%	95%	17 0	17 108	[0.8661971 0.88732394 0.91549296 0.88652482 0.90780142]
	Multinomial Naive Bayes Grid Search	90%	100%	89%	96%	21 0	13 108	[0.88732394 0.94366197 0.93661972 0.90780142 0.91489362]
	Gradient Boosting Classifier	88%	100%	86%	82%	17 0	17 108	[0.81690141 0.88732394 0.84507042 0.87234043 0.85815603]
	Random Forest Classifier	90%	95%	89%	97%	21 1	13 107	[0.87323944 0.92957746 0.91549296 0.89361702 0.88652482]
	XGBoosting Classifier	84%	92%	83%	96%	12 1	22 107	[0.76760563 0.8028169 0.80985915 0.78723404 0.78014184]
	Regressão Logística	83%	100%	82%	95%	10 0	24 108	[0.80985915 0.83098592 0.81690141 0.82978723 0.82978723]

Dos resultados expostos, as melhores *performances* foram do algoritmo *Multinomial Naives Bayes Grid Search* e o *Random Forest Classifier* com um valor de 90%. Em relação à matriz confusão, verificou-se que 21 notícias classificadas de forma correta na classe 1 e 107 notícias classificadas de forma correta na classe 4. Ainda assim, o *Multinomial Naives Bayes Grid Search* foi sem dúvida o

melhor no que diz respeito à *precision* da classe 1 com um valor de 100%. Em relação à métrica *K-Cross Validation* a média de todos os *scores* deste modelo foi de 89%.

A segunda fase concentrou-se num aumento do número de colunas no *max_features* da função TF-IDF de cada uma das classes, sendo que essa variável tomou o valor de 100 colunas. A Tabela 38 apresenta uma comparação dos algoritmos e as seguintes métricas de avaliação dos mesmos: a *accuracy*, a *precision* da classe 1, a *precision* da classe 4, a ROC AUC, *K-Cross Validation* com 5 *Folds* e a respetiva matriz confusão para cada algoritmo.

Tabela 38 Comparação dos algoritmos de máximo de caraterísticas 100

Max Features	**Algoritmo**	***Accuracy***	***Precision* Classe = 1**	***Precision* Classe = 4**	**ROC AUC**	**Matriz Confusão**		***Cross-Validation* (K=5)**
100	*Multinomial Naive Bayes*	88%	100%	87%	96%	18	16	[0.8591549 0.8943662 0.9225352 0.9007092 0.8936170]
						0	108	
	Multinomial Naive Bayes Grid Search	93%	100%	92%	97%	25	9	[0.9295774 0.9366197 0.9577464 0.9219858 0.9290780]
						0	108	
	Gradient Boosting Classifier	84%	93%	84%	81%	13	21	[0.8661971 0.8732394 0.7887323 0.8652482 0.8439716]
						1	107	
	Random Forest Classifier	90%	92%	91%	98%	23	11	[0.9084507 0.9014084 0.8873239 0.9007092 0.8865248]
						2	106	
	XGBoosting Classifier	82%	83%	82%	96%	10	24	[0.7816901 0.8098591 0.8309859 0.8156028 0.7588652]
						2	106	
	Regressão Logística	82%	100%	81%	97%	9	25	[0.8028169 0.8521126 0.8098591 0.8085106 0.8226950]
						0	108	

Nesta tabela verificou-se que o aumento do número de colunas trouxe uma melhor *performance* e classificação. Neste caso, o algoritmo *Multinomial Naive Bayes Grid Search* obteve uma *accuracy* de 93% com uma matriz confusão de 25 notícias acertaram na classe 1 (Falso) e 9 falhadas na classificação (Verdadeiro). Relativamente à classe 4 (Verdadeiro), 110 notícias foram classificadas de forma correta. Em relação à métrica *K-Cross Validation* com 5 *Folds*, verificou-se que o modelo se encontrava robusto e consistente com uma *performance* média de 93%.

Na terceira fase, a função TF-IDF de cada uma das classes contou com 150 colunas máximas. A Tabela 39 apresenta uma comparação dos algoritmos e as métricas de avaliação dos mesmos: a *accuracy*, a *precision* da classe 1, a *precision* da classe 4, a ROC AUC, *K-Cross Validation* com 5 *Folds* e a respetiva matriz confusão para cada algoritmo.

Tabela 39 Comparação dos algoritmos de máximo de caraterísticas 150

Max Features	**Algoritmo**	***Accuracy***	***Precision* Classe = 1**	***Precision* Classe = 4**	**ROC AUC**	**Matriz Confusão**		***Cross-Validation* (K = 5)**
150	*Multinomial Naive Bayes*	86%	100%	85%	97%	15	19	[0.8521126 0.8802816 0.9084507 0.8510638 0.8723404]
						0	108	
	Multinomial Grid Search	92%	100%	91%	98%	23	11	[0.8943662 0.9366197 0.9718309 0.9007092 0.9007092]
						0	108	
	Gradient Boosting Classifier	88%	90%	88%	92%	19	15	[0.8732394 0.8591549 0.8591549 0.8936170 0.8368794]
						2	106	
	Random Forest Classifier	90%	95%	89%	98%	21	13	[0.8591549 0.8802816 0.9014084 0.8794326 0.8865248]
						1	107	
	XGBoosting Classifier	81%	82%	81%	95%	9	25	[0.7676056 0.7816901 0.8098591 0.7659574 0.7163120]
						2	106	
	Regressão Logística	78%	100%	78%	97%	3	31	[0.7816901 0.8098591 0.7746478 0.7943262 0.8014184]
						0	108	

Desta forma, os algoritmos *Multinomial Naive Bayes Grid Search* e *Random Forest Classifier* foram os que obtiveram uma melhor *accuracy*, 92% e 90% respetivamente. No entanto, o algoritmo *Multinomial Naive Bayes Grid Search* obteve uma matriz confusão relativamente melhor que o *Random Forest Classifier*. Na matriz confusão deste algoritmo, verifica-se que a classe 1 atribuiu 23 notícias à mesma e 11 classificadas de forma incorreta na classe 4. Em relação à métrica *K-Cross Validation* a média de todos os *scores* deste modelo foi de 87%.

Na última fase, a função TF-IDF contou com 200 colunas máximas de entrada na mesma para cada classe. A Tabela 40 apresenta uma comparação dos algoritmos e as seguintes métricas de avaliação dos mesmos: a *accuracy*, a *precision* da classe 1, a *precision* da classe 4, a ROC AUC, *K-Cross Validation* com 5 *Folds* e a respetiva matriz confusão para cada algoritmo.

Tabela 40 Comparação dos algoritmos de máximo de caraterísticas 200

Max Features	**Algoritmo**	***Accuracy***	***Precision* Classe = 1**	***Precision* Classe = 4**	**ROC AUC**	**Matriz Confusão**		***Cross-Validation* (K = 5)**
200	*Multinomial Naive Bayes*	85%	100%	84%	98%	14	20	[0.8450704 0.8521126 0.8802816 0.8581560 0.8368794]
						0	108	
	Multinomial Grid Search	92%	100%	92%	99%	24	10	[0.9154929 0.9295774 0.9577464 0.9219858 0.8865248]
						0	108	
	Gradient Boosting Classifier	85%	94%	85%	91%	15	19	[0.8028169 0.8239436 0.8943662 0.8297872 0.8368794]
						1	107	
	Random Forest Classifier	89%	100%	88%	99%	19	15	[0.8450704 0.8732394 0.8661971 0.8510638 0.8652482]
						0	108	
	XGBoosting Classifier	80%	78%	80%	95%	7	27	[0.7535211 0.7746478 0.7676056 0.7517730 0.7730496]
						2	106	
	Regressão Logística	77%	100%	77%	97%	2	32	[0.7676056 0.7816901 0.7676056 0.7801418 0.7730496]
						0	108	

Deste modo, na Tabela 40 averiguou-se que o algoritmo *Multinomial Naive Bayes Grid Search* foi o melhor com uma *accuracy* de 92%. No caso da matriz confusão verificou-se o mesmo obteve melhores resultados tanto na classe 1 como na classe 4. Na classe 1, analisou-se que 24 notícias foram bem classificadas na mesma classe e 10 falharam. Na classe 4, apurou-se que o mesmo acertou em 110 notícias de forma correta. Em relação à métrica *K-Cross Validation* a média de todos os *scores* deste modelo foi de 92%.

Em suma, pode-se verificar que de todas as fases expostas nesta experiência o modelo que continha 100 colunas como valor nas *max_features* da função TF-IDF de cada uma das classes foi o que adquiriu melhores resultados e o melhor algoritmo sem dúvidas que foi o *Multinomial Naive Bayes Grid Search*.

6.5.4 Experiência 4

Numa quarta experiência utilizou-se como parâmetro de entrada para o modelo o *título + descrição* como uma única variável que passou pela fase de pré-processamento. Essa variável contou com a

função TF-IDF com o objetivo de criar diferentes matrizes de caraterísticas com as palavras mais frequentes numa determinada notícia e os respetivos algoritmos. No TF-IDF foi definida na função *n-gram* de (1,2) o que corresponde a uma junção de *Uni-Grams* e *Bi-Grams*.

O conjunto de dados foi dividido em 90% para treino (637 notícias) e 10% para teste (71 notícias), com base na função *train_test_split()*. Esta mudança do conjunto de dados para treino foi realizada tendo em conta o reduzido número de dados presentes no *dataset, mas* também para verificar o impacto desta mudança no modelo final. Este treino resultou no preenchimento das variáveis: *x_train* e *x_test* com o conteúdo do texto (variáveis independentes) e *y_train* e *y_test* com as *labels* de classificação (variáveis dependentes). Nesta experiência utilizou-se os seguintes algoritmos: *Multinomial Naive Bayes*, *Gradient Boosting Classifier*, *Random Forest Classifier*, *XGBoosting Classifier* e Regressão Logística.

A primeira fase, exposta na Tabela 41, apresenta as *max_features* iguais a 70, ou seja, criou-se uma matriz de caraterísticas com as 70 palavras mais frequentes numa determinada notícia. A avaliação dos resultados teve em conta métricas de *accuracy*, *precision*, ROC AUC de cada um dos seis algoritmos testados, *K-Cross Validation* com 5 *Folds* e a sua respetiva matriz confusão.

Tabela 41 Comparação dos algoritmos de máximo de caraterísticas 70

Max Features	**Algoritmo**	***Accuracy***	***Precision* Classe = 1**	***Precision* Classe = 4**	**ROC AUC**	**Matriz Confusão**		***Cross-Validation* (K = 5)**
70	*Multinomial Naive Bayes*	90%	100%	89%	95%	10	7	[0.8661971 0.8873239 0.9154929 0.8865248 0.9078014]
						0	54	
	Multinomial Grid Search	92%	100%	92%	95%	13	4	[0.9295774 0.9507042 0.9577464 0.9148936 0.9219858]
						0	54	
	Gradient Boosting Classifier	91%	100%	90%	82%	11	6	[0.8169014 0.8873239 0.8450704 0.8723404 0.8581560]
						0	54	
	Random Forest Classifier	94%	100%	93%	98%	13	4	[0.8732394 0.9295774 0.9154929 0.8936170 0.8865248]
						0	54	
	XGBoosting Classifier	83%	100%	82%	97%	5	12	[0.7676056 0.8028169 0.8098591 0.7872340 0.7801418]
						0	54	
	Regressão Logística	86%	100%	84%	96%	7	10	[0.8098591 0.8309859 0.8169014 0.8297872 0.8297872]
						0	54	

Da Tabela 41 foram identificados quatro algoritmos com uma *acurracy* acima de 90%, sendo que o melhor algoritmo foi o *Random Forest* com uma *accuracy* de 94% e a sua matriz confusão expôs um bom resultado com 13 notícias bem classificadas na classe 1 e 4 notícias mal classificadas. Relativamente à classe 4, o algoritmo determinou de forma convicta que 54 notícias foram bem rotuladas. Em relação à métrica *K-Cross Validation* a média de todos os *scores* deste modelo foi de 89%.

Na segunda fase, criou-se uma matriz de caraterísticas com as 100 palavras mais frequentes numa determinada notícia. A Tabela 42 apresenta uma comparação dos algoritmos e as seguintes métricas de avaliação dos mesmos: a *accuracy*, a *precision* da classe 1, a *precision* da classe 4, a ROC AUC, *K-Cross Validation* com 5 *Folds* e a respetiva matriz confusão para cada algoritmo.

Tabela 42 Comparação dos algoritmos de máximo de caraterísticas 100

Max Features	**Algoritmo**	***Accuracy***	***Precision* Classe = 1**	***Precision* Classe = 4**	**ROC AUC**	**Matriz Confusão**		***Cross-Validation* (K = 5)**
100	*Multinomial Naive Bayes*	91%	100%	90%	98%	11	6	[0.8591549 0.8943662 0.9225352 0.9007092 0.8936170]
						0	54	
	Multinomial Grid Search	94%	100%	93%	98%	13	4	[0.9647887 0.9436619 0.9647887 0.9436241 0.9503546]
						0	54	
	Gradient Boosting Classifier	91%	100%	90%	88%	11	6	[0.8661971 0.8732394 0.7887323 0.8652482 0.8439716]
						0	54	
	Random Forest Classifier	92%	92%	91%	99%	12	5	[0.9084507 0.9014084 0.8873239 0.9007092 0.8865248]
						1	53	
	XGBoosting Classifier	85%	100%	83%	97%	6	11	[0.7816901 0.8098591 0.8309859 0.8156028 0.7588652]
						0	54	
	Regressão Logística	87%	100%	86%	98%	8	9	[0.8028169 0.8521126 0.8095859 0.8085106 0.8226950]
						0	54	

Nesta tabela verificou-se que o melhor algoritmo foi *Multinomial Naive Bayes Grid Search* obteve uma *accuracy* de 94% e a sua matriz confusão foi a melhor de todas, pois 13 notícias acertaram na classe 1 (Falso) e 4 falharam na classificação. Relativamente à classe 4 (Verdadeiro) 54 notícias foram classificadas de forma correta na classe 4. Em relação à métrica *K-Cross Validation* a média de todos os *scores* deste modelo foi de 95%.

Na terceira fase criou-se uma matriz de caraterísticas com as 150 palavras mais frequentes numa determinada notícia. Bem como as métricas de *accuracy, precision*, ROC AUC de cada um dos seis algoritmos testados, *K-Cross Validation* com 5 *Folds* e a sua respetiva matriz confusão, conforme é visível na Tabela 43.

Tabela 43 Comparação dos algoritmos de máximo de caraterísticas 150

Max Features	**Algoritmo**	***Accuracy***	***Precision* Classe = 1**	***Precision* Classe = 4**	**ROC AUC**	**Matriz Confusão**		***Cross-Validation* (K = 5)**
150	*Multinomial Naive Bayes*	87%	100%	86%	99%	8	9	[0.8521126 0.8802816 0.9084507 0.8510638 0.8723404]
						0	54	
	Multinomial Grid Search	91%	100%	90%	99%	13	4	[0.9718309 0.9647887 0.9718309 0.9290780 0.9432624]
						0	54	
	Gradient Boosting Classifier	84%	88%	84%	93%	7	10	[0.8732394 0.8591549 0.8591549 0.8936170 0.8368794]
						1	53	
	Random Forest Classifier	90%	100%	89%	99%	10	7	[0.8591549 0.8802816 0.9014084 0.8794326 0.8865248]
						0	54	
	XGBoosting Classifier	83%	100%	82%	97%	5	12	[0.7676056 0.7816901 0.8098591 0.7659574 0.7163120]
						0	54	
	Regressão Logística	79%	100%	78%	98%	2	15	[0.7816901 0.8098591 0.7746478 0.7943262 0.8014184]
						0	54	

Da Tabela 43, o melhor algoritmo é o *Multinomial Grid Search* com uma *accuracy* de 91% e a matriz confusão apresenta 13 notícias bem classificadas na classe 1 e 4 notícias que foram mal classificadas. Na classe 4 foram rotuladas de forma correta 54 notícias. Em relação à métrica *K-Cross Validation* a média de todos os *scores* deste modelo foi de 95%.

Na última fase criou-se uma matriz de caraterísticas com as 200 palavras mais frequentes numa determinada notícia. Na Tabela 44 encontram-se expostas as métricas de *accuracy*, *precision*, ROC AUC de cada um dos seis algoritmos testados, *K-Cross Validation* com 5 *Folds* e a sua respetiva matriz confusão.

Tabela 44 Comparação dos algoritmos de máximo de caraterísticas 200

Max Features	**Algoritmo**	***Accuracy***	***Precision* Classe = 1**	***Precision* Classe = 4**	**ROC AUC**	**Matriz Confusão**		***Cross-Validation* (K = 5)**
200	*Multinomial Naive Bayes*	87%	100%	86%	99%	8	9	[0.8450704 0.8521126 0.8802816 0.8581560 0.8368794]
						0	54	
	Multinomial Grid Search	91%	100%	90%	99%	11	6	[0.9647887 0.9647887 0.9647887 0.9503546 0.9219858]
						0	54	
	Gradient Boosting Classifier	84%	75%	86%	95%	9	8	[0.8028169 0.8239436 0.8943662 0.8297872 0.8368794]
						3	51	
	Random Forest Classifier	87%	100%	86%	99%	8	9	[0.8450704 0.8732394 0.8661971 0.8510638 0.8652482]
						0	54	
	XGBoosting Classifier	79%	67%	80%	97%	4	13	[0.7535211 0.7746478 0.7676056 0.7517730 0.7730496]
						2	52	
	Regressão Logística	79%	100%	78%	99%	2	15	[0.7676056 0.7816901 0.7676056 0.7801418 0.7730496]
						0	54	

Na Tabela 44 encontra-se representado o melhor algoritmo desta fase, *Multinomial Naive Bayes Grid Search,* com uma *accuracy* de 91%. Em relação à matriz confusão pode-se dizer que o modelo acertou de forma convicta em 11 notícias na classe 1 – Falso e falhou em 6 notícias na mesma classe. Na classe 4 – Verdadeiro, o modelo acertou de forma convicta em 54 notícias. Em relação à métrica *K-Cross Validation* a média de todos os *scores* deste modelo foi de 95%.

6.5.5 Experiência 5

Nesta quinta experiência, o parâmetro de entrada para o modelo foi a fonte de informação e usou-se a técnica *Label Encoder* para construir uma matriz com as colunas representadas pelas fontes de informação e as linhas associadas às diferentes notícias.

Nesta experiência, o conjunto de dados foi dividido em treino e teste através da função *train_test_split()*. Consideraram-se 90% para treino e 10% para teste, o que corresponde a 637 amostras para treino e 71 amostras para teste. Esta divisão resultou no preenchimento das variáveis: *x_train* e *x_test* com a fonte de informação (variáveis independentes), e *y_train* e *y_test* com as *labels* de classificação (variáveis dependentes).

A Tabela 45 representa o conjunto de algoritmos de classificação utilizados (*Multinomial Naive Bayes*, o *Random Forest Classifier*, o *Gradient Boosting Classifier*, *XGBoosting Classifier* e Regressão Logística) e ainda as métricas de avaliação dos mesmos.

Tabela 45 Comparação dos algoritmos com as fontes de informação

Algoritmo	***Accuracy***	***Precision* Classe = 1**	***Precision* Classe = 4**	**ROC AUC**	**Matriz Confusão**	
Multinomial Naive Bayes	83%	73%	85%	88%	8	9
					3	51
Multinomial Naive Bayes Grid Search	83%	73%	85%	88%	8	8
					3	51
Gradient Boosting Classifier	83%	73%	85%	98%	8	9
					3	51
Random Forest Classifier	83%	73%	85%	89%	8	9
					3	51
XGBoosting Classifier	82%	70%	84%	86%	7	10
					3	51
Regressão Logística	83%	73%	85%	87%	8	9
					3	51

Apesar de todos os algoritmos apresentarem uma *accuracy* com um valor aproximadamente de 83%, os mesmos não conseguem classificar de forma correta nenhuma das classes apresentadas, conforme se pode verificar na matriz confusão apresentada anteriormente.

6.5.6 Experiência 6

Tendo em conta que, o número reduzido de notícias presentes no *dataset* é uma limitação no projeto e atendendo ao facto de haver uma necessidade de convergir para uma solução final, mas também por o processo de recolha/anotação das mesmas ser feito de forma relativamente demorada, apenas foram recolhidas 10 notícias novas. Sendo que na fase de implementação desta experiência foram adicionadas ao *dataset*, o qual regista assim 718 notícias. As mesmas foram consideradas nesta experiência.

Ao longo das experiências, verificou-se que as variáveis *título + descrição* tiveram um impacto significativo no modelo, bem como as fontes de informação e as categorias presentes no conjunto de dados. Assim, esta última experiência incluiu 90% para treino (646 notícias), 10% para teste (72 notícias) e uma junção de *Uni-grams* e *Bi-grams*. Esta mudança fez com que o modelo alcançasse melhores resultados.

Esta experiência focou-se na implementação de uma arquitetura hierárquica e inovadora composta por três modelos diferentes, conforme se pode verificar na Figura 64. Os dados encontram-se a ser processados nos mesmos, sendo que dois deles funcionam em paralelo nomeadamente, o modelo A – conteúdo textual das notícias e o modelo B – metadados das notícias. Por último, o modelo C é uma otimização da relação entre os modelos expostos (A e B).

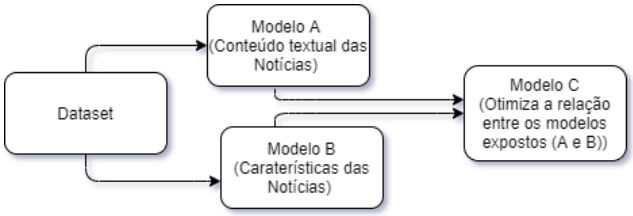


Figura 64 Arquitetura hierárquica inovadora

Começou-se por contruir a *pipeline* relativa ao modelo A que se foca numa análise textual do título e da descrição de uma notícia, conforme ilustra a Figura 65. Este modelo, contou com uma leitura do *dataset*, seleção do conteúdo textual da notícia e pré-processamento do mesmo, conforme foi abordado na Secção 6.3. Posteriormente, selecionou-se e extraíram-se 100 palavras dos dois *dataFrames*, 4 - Verdadeiro e 1 - Falso, com o TF-IDF, o que corresponde a 152 colunas para entrada no modelo. Os modelos foram treinados de forma iterativa e os seus hiperparâmetros foram otimizados. Neste caso, o melhor modelo foi o *Multinomial Naive Bayes*, de seguida extraiu-se o *predict_proba ()* das duas *labels* que permitiu determinar a probabilidade da classe 1 e 4, sendo que a classe 1 foi extraída e armazenada num novo ficheiro denominado de predict_proba_modeloA.csv.

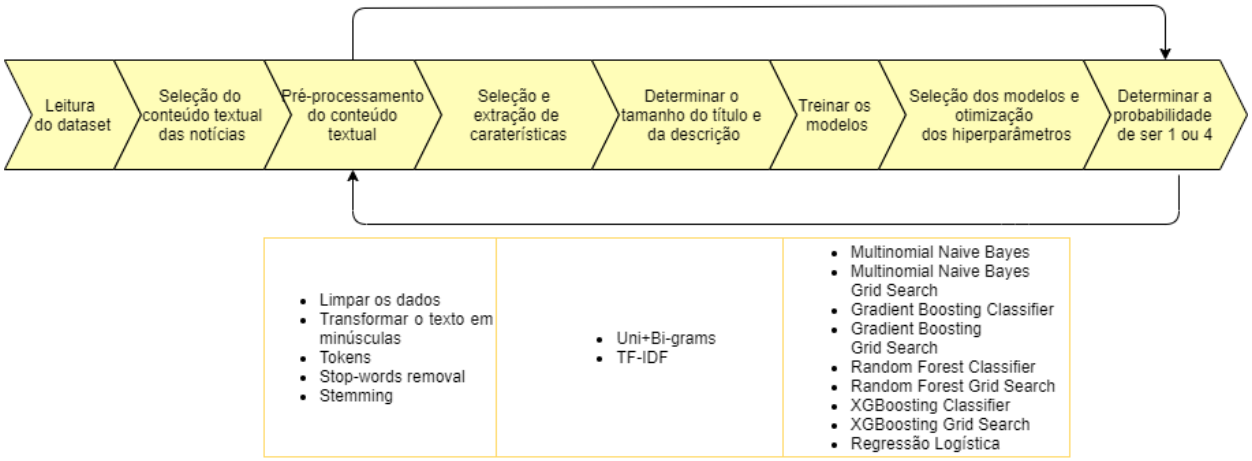


Figura 65 Modelo A

Dos algoritmos testados, o algoritmo *Multinomial Naive Bayes* foi o que apresentou melhores resultados com uma *accuracy* de 95%. A Figura 66 ilustra a matriz confusão do respetivo modelo.

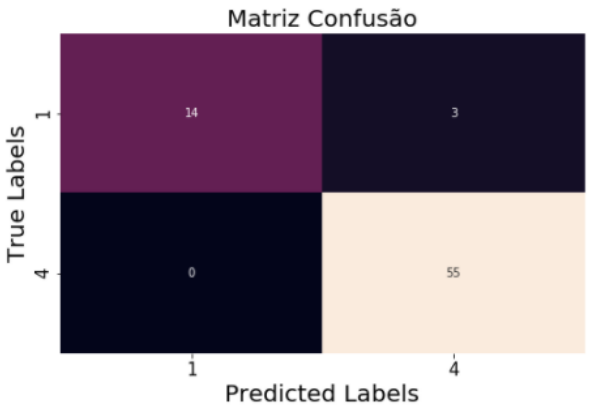


Figura 66 Matriz confusão do *Multinomial Naive Bayes*

Na Figura 66 percebe-se que o algoritmo do modelo A conseguiu atribuir de forma correta 14 notícias à classe 1 e classificar 55 notícias de forma acertada à classe 4.

Relativamente à *pipeline* do modelo B, Figura 67, procedeu-se à leitura do *dataset*, selecionou-se os metadados de uma determinada notícia (categorias e fontes de informação) e posteriormente realizou-se o replace () [50]. Como os algoritmos de aprendizagem automática não tem a capacidade de utilizar valores textuais como é o caso dos metadados, existiu uma necessidade de converter as mesmas em variáveis numéricas, conforme foi descrito na Secção 6.3. Após esta conversão, determinou-se o tamanho do título e da descrição de cada notícia. Assim, estas transformações proporcionaram a construção de um novo *dataframe* com as seguintes caraterísticas: categorias, fontes de informação, tamanho do título e da descrição. Este *dataframe* contou com 38 colunas como parâmetro de entrada para treinar os modelos e otimizar os hiperparâmetros dos mesmos. Após a determinação do melhor modelo, neste caso o *Random Forest Classifier*, extraiu-se o *predict_proba ()* das duas classes, ou seja, a probabilidade da classe 1 e da 4, sendo que a classe 1 também foi extraída e armazenada num novo ficheiro denominado de predict_proba_modeloB.csv.

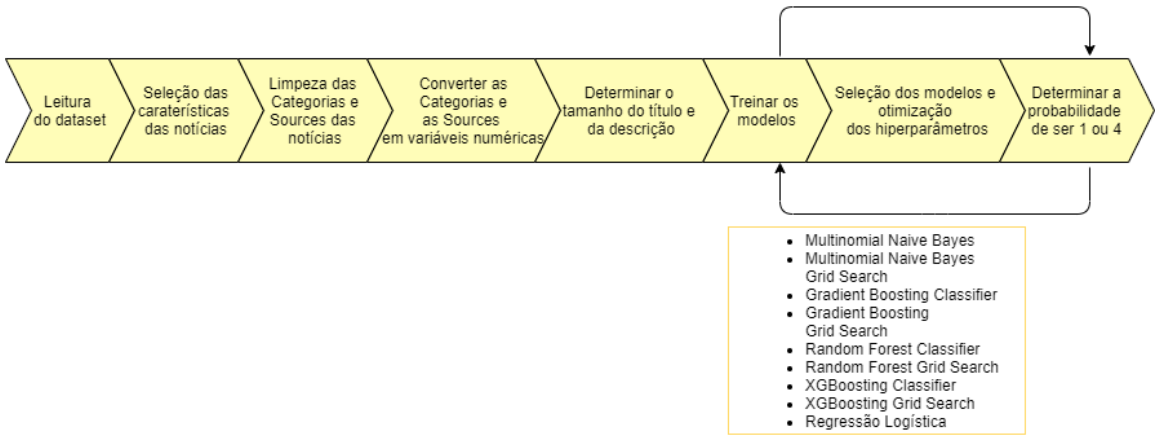


Figura 67 Modelo B

[50] Método implementado para substituir uma categoria por outra

O algoritmo *Random Forest Classifier* foi o que obteve a melhor *performance*, 87%, no modelo B. Na Figura 68 apresenta-se uma matriz confusão com as notícias previstas de forma correta e de forma incorreta numa determinada classe.

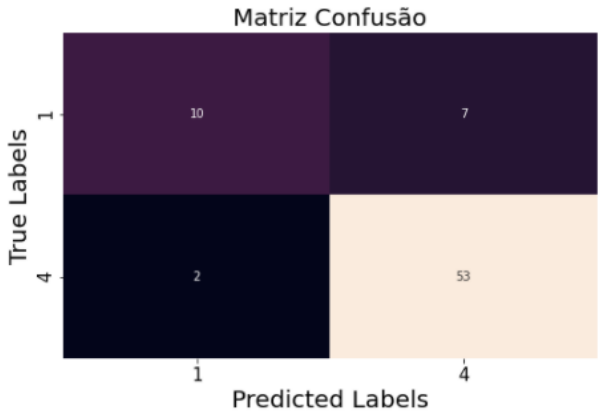


Figura 68 Matriz confusão do *Random Forest Classifier*

Neste caso, o algoritmo *Random Forest Classifier* na previsão da classe 1 acertou de forma convicta em 10 notícias e falhou em 7 pois classificou-as na classe 4. Na previsão da classe 4, 53 notícias foram classificadas de forma convicta na classe 4 e 2 das notícias na classe 1.

Por último, a *pipeline* do modelo C, Figura 69, formou-se pela leitura das probabilidades das classes 1 extraídas do modelo A e do B. Seguidamente, criou-se um novo *dataFrame* composto pelas duas probabilidades e efetuou-se o treino dos modelos. Por último, otimizou-se os hiperparâmetros dos mesmos. Este modelo permitiu alcançar as previsões das classes e as classificações reais.

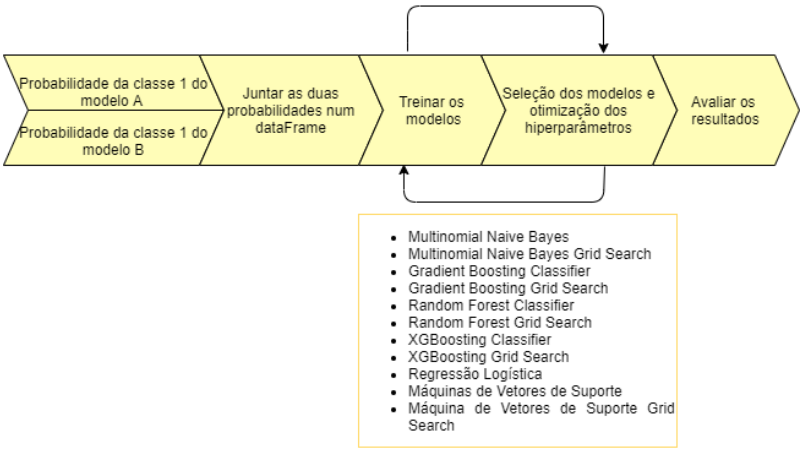


Figura 69 Modelo C

O melhor algoritmo do modelo C com uma *accuracy* de 92% foi a Máquinas de Vectores de Suporte e a precisão de cada classe corresponde a 92%. Na Figura 70 apresenta-se a matriz confusão que foi uti lizada para avaliar os resultados.

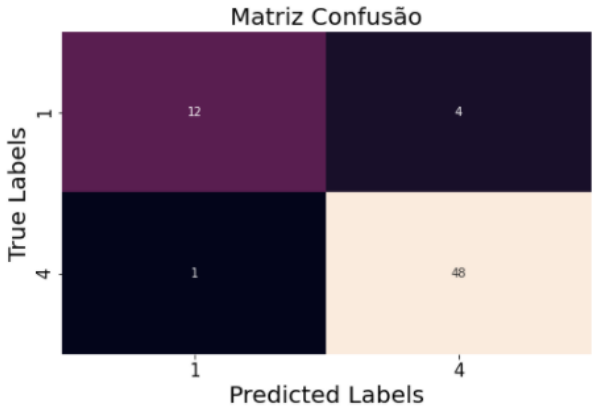


Figura 70 Matriz confusão do algoritmo Máquina Vetores de Suporte

Na Figura 70 encontra-se a matriz confusão do algoritmo Máquina Vetores de Suporte com 12 notícias classificadas na classe 1 e 4 notícias classificadas de forma errada na classe 1. Mas também é possível se verificar que o modelo para a classe 4 acertou de forma convicta em 48 e classificou mal apenas 1 na classe 1.

Em suma, esta arquitetura ABC proporcionou uma extração de dois índices de credibilidade associados à probabilidade de cada classe, bem como a resolução do problema de subjetividade com base nos critérios que foram selecionados na abordagem multiclasses e ainda a minimização da entropia da informação alimentada aos modelos.

6.5.6.1 Interpretação dos resultados

Numa análise mais complementar utilizou-se a biblioteca LIME (*Local Interpretable Model-Agnostic Explanations*) tratando-se o mesmo de um explicador local com base em *features importances*, ou seja, permite explicar a classificação de uma amostra com base na importância de cada uma das suas *features* (Ribeiro, Singh and Guestrin, 2016). Esta biblioteca utiliza um conjunto de dados de treino como uma matriz, bem como as caraterísticas que correspondem à lista de colunas desses dados de treino. Por fim, as classes, que neste caso alternam entre 1 (Falso) e 4 (Verdadeiro).

De seguida, apresenta-se um exemplo para cada modelo descrito anteriormente relativo à notícia número 1 do *dataset*. A Figura 71, Figura 72 e Figura 73 ilustram uma análise mais detalhada do modelo A, B e C, respetivamente. A primeira coluna refere a probabilidade de ambas as classes, a segunda coluna apresenta os contributos individuais de cada palavra em relação a cada uma das classes, e por fim a terceira coluna expõe o valor real de cada uma *features* na amostra.

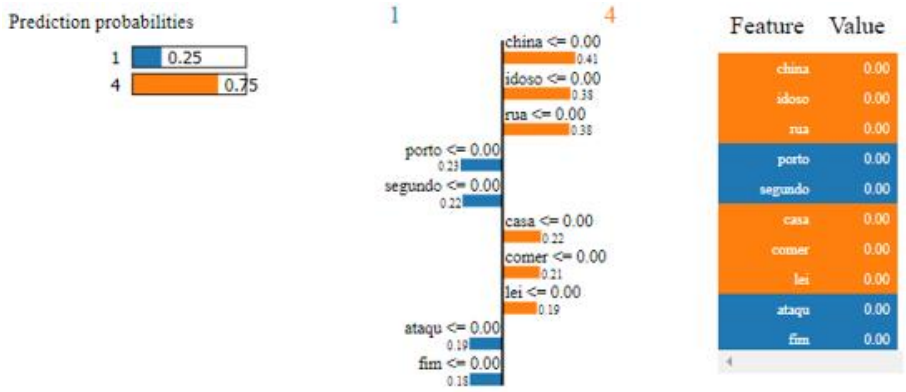


Figura 71 Análise *Explainable* do Modelo A

Relativamente ao modelo A, Figura 71, a primeira coluna apresenta a probabilidade da classe 1 igual a 25% e a da classe 4 igual a 75%. Na segunda e terceira colunas, as palavras que se encontram a azul são consideradas de classe 1 e a laranja de classe 4. Por exemplo, a palavra "China" toma o valor de 0.00 na amostra (terceira coluna), e contribuiu para a classificação da classe 4 com o valor de 41, tendo cumprido a regra <= 0 nesse sentido (segunda coluna). Outro exemplo, a palavra "porto" toma o valor de 0.00 na amostra (terceira coluna), e tem um contributo para a classificação da classe 1 com o valor de 23, sendo que cumpriu a regra <= 0 nesse sentido (segunda coluna).

O modelo B, Figura 72, refere-se à probabilidade da classe 1 que corresponde a 91% e à da classe 4 que contou com uma probabilidade de 0.09%. Na segunda coluna contou-se com as categorias, fontes de informação e o tamanho da descrição que correspondem à classe 1, encontradas a azul. Neste caso, o exemplo da "category_new" tomou o valor de 6.00 conforme ilustra a terceira coluna, e contribui para a classe 1 com o valor de 15, tendo cumprido a regra <= 6.00. Na terceira coluna representaram-se apenas as *features* que contribuem para a classe 1.

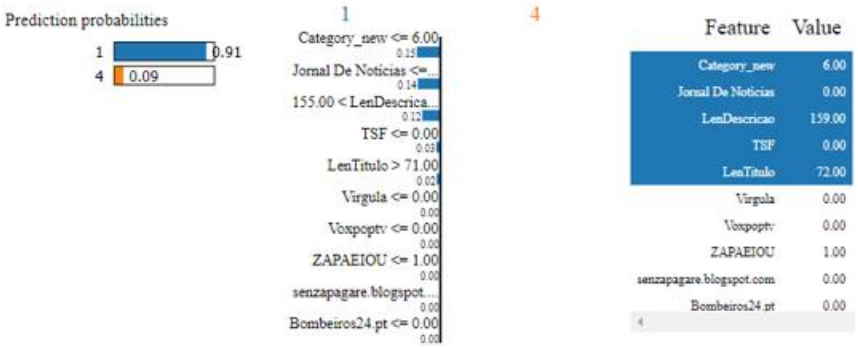


Figura 72 Análise *Explainable* do Modelo B

Por último o modelo C, Figura 73, na primeira coluna encontra-se a probabilidade da classe 1 que tomou um valor percentual de 92% e a probabilidade da classe 4 que teve uma atribuição de 0.08%. Na segunda coluna expôs-se as caraterísticas contribuintes para a classe 1 que toma o valor de 1.00 na amostra (terceira coluna) e com um contributo para a classificação da classe 1 com um valor de 53. Relativamente, à classe 4 contou apenas com análise textual, tendo esta tomado o valor de 0.25, com um contributo para a classe 4 de valor 0.02, tendo assim cumprido a regra < 0.25 (segunda coluna). Na última coluna, descreveu-se a *feature* "Caraterísticas" que contribuíram 1.00 para a classe 1, sendo que a *feature* "NLP" contribuiu 0.25 para a classe 4.

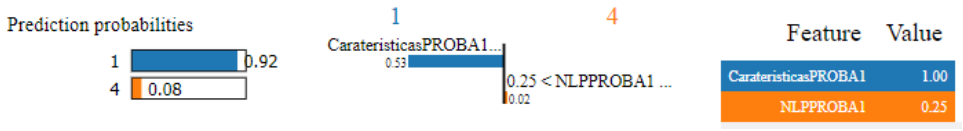


Figura 73 Análise *Explainable* do Modelo C

6.6 Web Service

Na última fase do projeto, após se ter desenvolvido o modelo ideal utilizou-se a biblioteca *pickle* que tem como objetivo transformar o modelo numa sequência de *bytes*. Esta biblioteca permite armazenar os três modelos (A, B e C) já treinados no disco. Estes ficheiros foram usados pelo servidor e carregados novamente pelo método pickle.load() que carrega os modelos e obtém os diferentes resultados expostos com o modelo.predict(). Para alcançar as probabilidades que aparecem no modelo A e B, utilizou-se o modelo.predict_proba() que expõe as probabilidades de cada classe.

Para validação e utilização do modelo abordado anteriormente, desenvolveu-se um *web service* e uma REST API com base na *microframework* Flask utilizando a linguagem Python. Esta API contém um *endpoint* onde o modelo foi testado. Esta aplicação *web* permite que qualquer utilizador insira todos os parâmetros relevantes para serem utilizados no modelo, com base no treino do mesmo. O modelo C permite detetar a veracidade de uma determinada notícia conforme os dados que foram inseridos.

O *web service* armazenou-se num servidor local na porta 5000. Inicialmente, implementou-se uma @app.route('/') que retorna a página html inicial do projeto denominada de "inicio.html" que representa a conexão feita ao servidor *Flask* em http://127.0.0.1:5000/ conforme é possível verificar na Figura 74.

Figura 74 Página Inicial do *web service*

Ao clicar no "Verificar?" é aberta uma nova página HTML, Figura 75, onde qualquer utilizador pode inserir os dados necessários para detetar se uma notícia é Verdadeira, Duvidosa ou Falsa. De seguida, desenvolveu-se uma API que permite detetar a classificação de cada notícia. Efetuou-se uma nova @app.route('/modelofinal') que recebe os dados dos parâmetros por meio da *interface* "detetor.html"

e, ao clicar no botão "Verificar" devolve o resultado na mesma página *web* utilizando o render_template().

Figura 75 Página do Detetor de notícias

Quando se clica no botão "Verificar" em detetor.html, o sistema envia os valores inseridos pelo utilizador usando um pedido *POST* (4 entradas – título, descrição, categoria e fontes de informação), passa a previsão obtida pelo modelo e devolve ao ficheiro detetor.html através da função render_template() o resultado obtido. A Figura 76 representa os dados inseridos pelo utilizador a serem transformados num dicionário para um *dataFrame* que são passados pelos modelos A, B e C. No fim deste método é devolvido a previsão de classificação da notícia inserida.

```
@app.route('/modelofinal', methods = ['GET','POST'])
def modelofinal():

        request_body = get_request_body(request)
        news_title = request_body['Title']
        news_description = request_body['Description']
        valor =  pd.DataFrame.from_dict(request_body, orient='index').T
        print(valor)
        valor = valor.rename(columns = {'Source[]': 'Source', 'Category[]': 'Category'}, inplace = False)
        print(valor)
```

Figura 76 Modelo Final

Relativamente à Figura 75, se o utilizador pretender apagar os dados inseridos e as previsões alcançadas pelo modelo deve clicar no botão "Limpar os dados". Se o utilizador pretender voltar à página inicial do *web service* deve clicar no *link* "Pretendes voltar para a página inicial?".

A classificação de uma determinada notícia realizou-se com base nos valores alcançados pela função predict_proba() da classe 4, conforme ilustra a Tabela 46.

Tabela 46 Classificação das notícias

Classificação	Valores do predict_proba ()
1 – Falsa	< 0.30
3 - Duvidosa	>= 0.30 & < 0.70
4 - Verdadeira	>= 0.70

Numa fase de teste ao *web service,* utilizou-se três exemplos de notícias extraídas (conforme se pode verificar na Figura 77 e nas Figura 80 e Figura 81 presentes no Anexo A) de três fontes de informação e duas categorias diferentes. A Figura 77 representa o exemplo de uma notícia classificada como Falsa da fonte de informação "Notícias ao Minuto" e com a categoria "Dinheiro".

Figura 77 Notícia classificada como Falsa

As experiências relativas à notícia ser classificada como Verdadeira e Duvidosa encontram-se expostas no Anexo A, Figura 80 e Figura 81, respetivamente. A Figura 80 ilustra uma notícia classificada como Verdadeira, da fonte de informação "ZAP AEIOU" e com a categoria "Portugal". A Figura 81 que representa uma notícia classificada como Duvidosa, proveniente da fonte de informação "Correio da Manhã" e com a categoria "Portugal".

Conforme foi demonstrado, o *web service* permite testar a veracidade de diferentes notícias, tendo em conta os diferentes parâmetros anteriormente expostos, sendo que as categorias e as fontes de informação se encontram limitadas às opções atualmente disponíveis no mesmo.

6.7 Análise Comparativa

Tendo em conta toda a solução desenvolvida e o estudo efetuado no estado de arte, decidiu-se comparar a solução final com o único projeto desenvolvido em português europeu pelo autor Rodrigues (Rodrigues, 2020). Ambos os projetos consistem num modelo de notícias falsas em português e contam com um *dataset* anotado em duas classes (Falso e Verdadeiro). No sentido de se

comparar as duas abordagens, construiu-se a Tabela 47 com o propósito de verificar a fonte de informação utilizada em cada projeto; o processo de construção do dataset; o número de notícias presente em ambos os conjuntos de dados; os parâmetros utilizados em cada modelo; técnicas de processamento das variáveis textuais; algoritmos utilizados e por último qual a métrica de avaliação da solução, a melhor *performance* e o melhor algoritmo em cada caso.

Tabela 47 Análise Comparativa

Autores / **Caraterísticas**	**Rodrigues (Rodrigues, 2020)**[51]	**Projeto de Dissertação**
Fonte de Informação	3 *Websites* de verificação de factos	Sites fidedignos e não fidedignos
Construção do *Dataset*	Automática	Semi-automática
Número de dados	3764 (872 verdadeiras e 2889 falsas)	718 (543 verdadeiras e 175 falsas)
Dados Desequilibrados	✓	✓
Conjunto de parâmetros para o modelo	Modelo composto pelo Título + Descrição + Categoria	Modelo A – "Título e Descrição" Modelo B – Fontes de informação, Categorias e Tamanho do título e descrição Modelo C – Probabilidade modelo A e B
Técnicas de Transformação do texto	TF-IDF e *One-Hot Encoding*	TF-IDF*, Label Encoder* e *One-Hot Encoding*
Algoritmos utilizados	*Multinomial Naive Bayes, Random Forest,* Regressão Logística, *Extra Trees,* CNN, LSTM, *K-Nearest Neighbors, Linear Support Vector Machine*, Máquina Vetores de Suporte*, Gradient Boosting*	*Multinomial Naive Bayes, Multinomial Grid Search, Gradient Boosting Classifier, Random Forest Classifier, XGBoosting Classifier,* Regressão Logística*, K-Nearest Neighbors Classifier,* Máquina Vetores de Suporte*, Decision Trees, LSTM*
Melhor Algoritmo e *Performance*	*Multinomial Naïve Bayes* (*F1-Score:* 86% - Falso e 60% - Verdadeiro)	Modelo A - *Multinomial Naive Bayes- Accuracy:* 95% Modelo B – *Random Forest Classifier- Accuracy:* 87% Modelo C – *Support Vector Machine- Accuracy:* 92%
Métrica	*Precision* de cada classe e *F1-Score*	*Accuracy, Precision* de cada classe e Matriz confusão

Na fase de experimentação, o autor Rodrigues (Rodrigues, 2020) efetuou duas experiências com dados equilibrados e desequilibrados. No entanto, na Tabela 47 encontra-se exposta apenas uma análise à experiência com os dados desequilibrados. Tendo em conta que o presente projeto de dissertação também se encontra desenvolvido com dados desequilibrados, verificou-se que o modelo com este conjunto de dados, apresenta uma *performance* relativamente superior à dos dados equilibrados, conforme se pode verificar na Secção 3.3.1. Dessa forma, pode-se dizer que a literatura suportou esta fase da implementação.

[51] Relativamente à outra experiência deste trabalho, com dados equilibrados, encontra-se descrito na Subsecção 3.3.1.

Apesar de ambos os projetos serem em língua portuguesa e apresentarem uma metodologia similar tanto a nível de pré-processamento de dados, como de técnica de representação das variáveis textuais, o presente projeto de dissertação conta com três vantagens nomeadamente, o facto de o *dataset* ter sido construído com base em *sites*, apresentar dois índices de credibilidade referentes à análise linguística e dos metadados de uma notícia e ainda um detetor de notícias falsas não binário.

Relativamente ao *dataset,* ainda que as notícias presentes no trabalho desenvolvido por Rodrigues (Rodrigues, 2020) tenham sido anotadas por jornalistas, apenas permite uma análise daquele conjunto de fontes de informação. No entanto, a presente dissertação contou com uma lista de *sites* considerados fidedignos e não, permitindo assim uma análise mais aprofundada, bem como uma análise das suas categorias e do seu conteúdo. Em comparação com o projeto desenvolvido pelo autor Rodrigues (Rodrigues, 2020), este só permite uma análise em relação às categorias e à sua análise textual.

Apesar do número de dados do projeto Rodrigues (Rodrigues, 2020) ser bastante superior ao desta dissertação, a mesma apresenta significativamente melhores resultados nas várias métricas. Em relação ao conjunto de dados, o autor Rodrigues (Rodrigues, 2020) contou com um maior número de notícias falsas do que verdadeiras em contraste com o presente projeto, o que apresenta o inverso. Em relação aos seus resultados pode-se afirmar que o modelo desenvolvido alcança melhores resultados no que diz respeito à classe Falsa.

No que diz respeito às métricas, pode-se dizer que a métrica precisão de cada classe do modelo final presente na mesma, apresenta um valor percentual de 92% em ambas as classes. O outro projeto, apresenta para a classe Falsa uma precisão de 87% e para a classe Verdadeira uma precisão de 68%. No que diz respeito, à métrica *accuracy*, o projeto de dissertação apresenta um valor de 92% e a mesma métrica referente ao outro projeto apresentou um valor de 80%.

Em suma, o resultado do modelo desenvolvido na presente dissertação retorna uma classificação em três classes (1 – Falsa, 3 – Duvidosa e 4 – Verdadeira), sendo que o projeto desenvolvido por Rodrigues (Rodrigues, 2020) apenas permite classificar uma notícia como Verdadeira ou Falsa.

6.8 Sumário

Neste capítulo descreveu-se toda a implementação da solução. Criou-se um *dataset* composto por 708 notícias de diferentes *sites* de notícias, exploraram-se as duas anotações (quatro classes e em duas classes), para se perceber quais as variáveis que teriam um impacto significativo no modelo final.

Na etapa seguinte, construiu-se uma *pipeline* de pré-processamento dos dados tendo por base várias técnicas de *Text Mining*. De seguida, apresentaram-se duas abordagens, a primeira relativa à anotação em multiclasses e a segunda em relação à anotação binária. Relativamente à segunda abordagem, anotação binária, testou-se um conjunto de parâmetros até serem encontrados os melhores e um conjunto de algoritmos. Entretanto, numa das experiências foi implementada a arquitetura ABC que é construída com base num *dataset* binário, que permite assim obter dois índices de credibilidade

(modelo A e B) e um detetor de notícias falsas com base em três níveis de credibilidade (Falsa, Duvidosa e Verdadeira). Com o objetivo de validar todo o modelo implementado, desenvolveu-se um *web service* que permitiu testar o detetor de notícias falsas em língua portuguesa. Por fim, efetuou-se uma análise que compara o projeto realizado com um projeto descrito no estado de arte.

7 Experimentação e Avaliação

Neste capítulo pretende-se abordar a Experimentação e Avaliação da solução, este encontra-se dividido em seis secções. Na Secção 7.1 expõem-se a hipótese que se pretende estudar no fim da concretização do projeto. Na Secção 7.2 descrevem-se as grandezas desta solução. Na Secção 7.3 identificam-se os indicadores e as fontes de informação. Na Secção 7.4 é descrita a *framework* de avaliação da qualidade e os diferentes indicadores considerados. Na Secção 7.5 expõem-se a metodologia de avaliação para avaliar a solução exposta. Por último, na Secção 7.6 descreve-se de forma sucinta um resumo de todo o capítulo abordado.

7.1 Hipótese

Com base no problema, tornou-se fulcral a formalização de uma hipótese para a avaliação da solução, assim, esta pode ser testada no fim da realização do projeto. Desta forma, foi considerada como hipótese:

- É possível construir um índice de credibilidade das notícias a partir de algoritmos de aprendizagem automática.

Tendo em conta, toda a solução implementada pode se considerar que a hipótese que se pretendia testar foi verificada através das grandezas abordadas e das duas metodologias de avaliação implementadas conforme se pode verificar de seguida.

7.2 Grandezas

Tendo em conta que o projeto desenvolvido se foca na determinação da classificação de forma acertada de uma determinada notícia, é necessário que os modelos presentes na arquitetura desenvolvida (ABC) apresentem uma exatidão na sua *performance*, mas também que na precisão em

cada classe prevista tenha notícias classificadas de forma correta. Nesse sentido, consideraram-se como grandezas: as taxas de acerto (matriz confusão) e a precisão alcançada de cada classe. Por muito que o modelo alcance excelentes resultados em termos de exatidão, a notícia pode ser classificada de forma errada, assim torna-se necessário analisar de forma mais complexa a matriz confusão e as respetivas precisões.

7.3 Identificação dos Indicadores e Fontes de informação

Os dados obtidos para o projeto foram alcançados a partir de vários sites de notícias considerados fidedignos e não fidedignos. Inicialmente, o *dataset* foi anotado em multiclasses (1 – Falso, 2 – Parcialmente Falso, 3 – Parcialmente Verdadeiro e 4 – Verdadeiro), verificou-se que os resultados não eram significativos, consoante os critérios subjetivos utilizados para a anotação. Dessa forma, procedeu-se a uma reanotação do mesmo numa classificação binária (4 - Verdadeiro e 1- Falso). Ao longo do projeto foram utilizados vários algoritmos de classificação e três indicadores para determinar o melhor algoritmo para cada modelo, nomeadamente: *accuracy*, precisão e a taxa de acerto (matriz confusão).

7.4 Framework de Avaliação da Qualidade

A *Framework* de Avaliação da Qualidade (QEF) é utilizada para avaliar a qualidade de um produto, sendo necessário identificar uma medida que quantifique o grau de alcance de uma caraterística de qualidade. A qualidade de um determinado sistema é definida no espaço cartesiano tridimensional (Escudeiro and Escudeiro, 2012).

Cada dimensão pode agregar um conjunto de fatores. Um fator é um componente que representa o desempenho do sistema de um determinado ponto de vista. As dimensões do projeto no espaço cartesiano de qualidade são: Desempenho (D), Fiabilidade (F) e Usabilidade (U). As dimensões de Desempenho e Fiabilidade são avaliadas de forma objetiva em relação aos algoritmos de aprendizagem automática incorporados na solução. Por sua vez, a dimensão Usabilidade é avaliada de forma subjetiva através de questionários sobre a interação dos utilizadores com o modelo.

7.5 Metodologia de Avaliação

A metodologia de avaliação da solução consiste em diferentes tipos de técnicas de avaliação nomeadamente, questionários, entrevistas, testes, entre outros. Neste caso pretende-se utilizar duas metodologias de avaliação: um conjunto de testes de *performance* capazes de avaliar a solução relativos ao desenvolvimento do modelo e um questionário para avaliar a usabilidade do *web service* desenvolvido.

7.5.1 Avaliar a Solução Relativa ao Modelo

Relativamente à avaliação do modelo desenvolvido foram utilizadas três métricas nomeadamente a *accuracy*, precisão e a taxa de acerto (matriz confusão). A primeira métrica, no modelo A obteve uma *accuracy* de 95%, no modelo B uma *accuracy* de 87% e no modelo C uma *accuracy* de 92%. A segunda métrica descreve a precisão de cada uma das classes. No modelo A, a precisão da classe 1 tomou o valor percentual de 100 e da classe 4 toma o valor percentual de 92. A precisão da classe 1 no modelo B toma o valor de 83%, a classe 4 toma o valor de 88%. No modelo C, a precisão de ambas as classes são de 92%.

Adicionalmente e com o intuito de validar a robustez da solução proposta as mesmas métricas foram consideradas num ambiente de *K-Cross Validation*, conforme se pode verificar na Tabela 48. Nesta técnica o *dataset* foi dividido em 5 *Folds* (K=5) e em cada iteração um *Fold* é reservado para teste enquanto os outros são usados para treino do modelo. Para cada iteração foi extraída a métrica *accuracy*.

Tabela 48 Avaliação da métrica *K-Cross Validation*

Modelos	***K-Cross Validation* (K=5)**
A	[0.9153 0.8992 0.9534 0.9379 0.9379]
B	[0.7692 0.8062 0.7829 0.8217 0.7906]
C	[0.9487 0.8620 0.9655 0.9482 0.9655]

Da Tabela 48 percebe-se o *score* de cada subdivisão do modelo e verifica-se que o *score* dos três modelos mantém a robustez. A média de todos os *scores* do modelo A, B e C foi de 0.92, 0.79 e 0.93, respetivamente.

Em relação à última métrica, taxa de acerto, verificou-se para cada modelo quantas notícias de teste foram classificadas de forma correta e incorreta na classe 1 e 4.

Tabela 49 Avaliação da métrica taxa de acerto

Modelos	**Matriz confusão**	
A	14	3
	0	55
B	10	7
	2	53
C	12	4
	1	48

Da Tabela 49, o modelo A conta com um acerto de 14 notícias classificadas na classe 1 e 3 notícias mal previstas classificadas na classe 4. Verifica-se que 55 notícias foram bem classificadas na classe 4. O modelo B conta com 10 notícias bem classificadas na classe 1 e 7 que deviam ter sido classificadas na

classe 1 foram atribuídas à classe 4. A classe 4 do modelo B conta com 2 notícias mal classificadas na classe 1 e 53 bem classificadas como 4. Por último, o modelo C tem atribuído 12 notícias classificadas como 1 e 4 notícias classificadas na classe 4. Em relação, à classificação das notícias na classe 4 acerta em 48 e atribuí apenas 1 à classe 1.

7.5.2 Avaliar a Usabilidade do *Web Service*

Por outro lado, foi desenvolvido um questionário que avalia a usabilidade do *web service*. O objetivo deste passou por efetuar várias questões a diferentes utilizadores permitindo assim extrair informações relevantes sobre o que foi desenvolvido.

Este questionário contou com uma amostra de 10 utilizadores. O número de amostras é limitado, tendo em conta que o *web service* encontra-se armazenado num servidor local. O questionário é composto por cinco questões conforme se pode verificar no Anexo B - Tabela 50.

Tendo em conta, as questões descritas na Tabela 50 é possível verificar-se as respostas mais frequentes de cada questão do questionário realizado, no Anexo B (Figura 82, Figura 83, Figura 84, Figura 85 e Figura 86). Quando questionados sobre a intuição de utilização do detetor, 100% concorda que é muito intuitivo, conforme ilustra a Figura 82. Em relação à utilidade do detetor 80% dos inquiridos responderam que o mesmo era muito útil, 10% responderam que era inútil e mediano, como se pode verificar na Figura 83. Quando questionados sobre a rapidez deste detetor, Figura 84, 90% dos inquiridos responderam que era mediano e 10% considerou que o mesmo era rápido. Relativamente à questão sobre a facilidade de interpretação dos dados, Figura 85, 80% dos inquiridos respondeu que era fácil de interpretar os mesmos e 20% considerou que era mediana. Em relação à última questão, questionados sobre a recomendação do detetor a outras pessoas, Figura 86, 90% dos inquiridos respondeu que recomendava.

Em suma, apesar do *web service* desenvolvido ser simples, numa visão geral pode-se dizer que o detetor é intuitivo, útil, medianamente rápido, fácil de interpretar e a maior parte destes utilizadores recomendaria este detetor a outras pessoas.

7.6 Sumário

Em suma, neste capítulo abordou-se a hipótese da solução e como foi avaliada, apresentaram-se as grandezas: as taxas de acerto (matriz confusão) e a precisão alcançada de cada classe, identificaram-se as fontes de informação e os indicadores. Para além disso, abordou-se as dimensões do projeto em relação à *framework* de avaliação da qualidade e por último, apresentaram-se as duas metodologias de avaliação, nomeadamente *K-Cross Validation* e o questionário desenvolvido.

8 Conclusões

Neste capítulo descrevem-se todos os objetivos atingidos tendo em conta os que foram propostos no início de todo o projeto. De seguida, descrevem-se as limitações encontradas no decorrer do desenvolvimento do mesmo e quais as melhorias que podem ser introduzidas no trabalho futuro de forma a valorizar todo o trabalho implementado. Por último, apresentam-se as contribuições deste projeto e o resumo de todo o capítulo.

8.1 Resultados Alcançados

Relativamente ao projeto apresentado, considera-se que o mesmo cumpriu todos os objetivos a que se propunha, tendo sido desenvolvido e anotado um *dataset* de raiz, um detetor de notícias falsas orientado para a língua portuguesa com três níveis de credibilidade (Falsa, Duvidosa e Verdadeira). Destaca-se ainda que as abordagens implementadas abordaram o problema de forma abrangente, tendo-se convergido para uma solução inovadora e de elevada fiabilidade.

Conforme foi referido anteriormente, recolheu-se um conjunto de dados provenientes de diferentes sites de notícias possibilitando a construção de um dataset composto por 708 notícias e composto por onze parâmetros de entrada, nomeadamente: *Title, Description, URL, Publishing Date, Publishing Time, Category, Source, Content, Author, Year* e *Classification*.

Na primeira fase do projeto, as notícias foram anotadas em quatro classes, com base em diferentes critérios, conforme foi descrito na Secção 6.1.2. De seguida, procedeu-se ao pré-processamento dos dados com técnicas de *Text Mining*. Nesta abordagem, foram desenvolvidas diferentes experiências tendo em conta duas metodologias, 80% de treino e 20% de teste e 90% de treino e 10% de teste. Ao longo das experiências foram testados diversos parâmetros de entrada, diferentes algoritmos de classificação e a otimização dos hiperparâmetros. Na análise e testes efetuados verificou-se que a categorização em 4 classes (1 – Falso, 2 – Parcialmente Falso, 3 – Parcialmente Verdeiro e 4 – Verdadeiro) não era de todo eficaz, uma vez que os critérios de segmentação eram subjetivos. Neste

sentido, reanotou-se o mesmo em 2 classes (1 – Falso e 4 – Verdadeiro), sendo esta é a abordagem que mais se aproxima de todo o estudo realizado no estado de arte. Com esta segunda abordagem, na experiência 6 (Subsecção 6.5.6), implementou-se uma arquitetura constituída por três modelos, nomeadamente: o modelo A, B e C. O modelo A foca-se numa análise do conteúdo textual de uma determinada notícia e o seu melhor algoritmo foi *Multinomial Naive Bayes* com uma *accuracy* de 95%. O modelo B diz respeito aos metadados presentes numa notícia e o melhor algoritmo foi o *Random Forest Classifier* com uma *accuracy* de 87%. Por último, o modelo C, tem como propósito fazer uma ponderação entre os dois modelos (A e B) para otimizar os resultados e o algoritmo que teve uma melhor performance foi Máquina de Vetores de Suporte com uma *accuracy* de 92%.

A arquitetura implementada trouxe as seguintes vantagens: minimização da entropia nos modelos; permitiu alcançar dois índices de credibilidade (modelo A e B) e um detetor de notícias falsas com três índices de credibilidade (modelo C).

A determinação dos melhores modelos consistiu na leitura das várias matrizes de confusão, tal como da análise da precisão para cada classe anotada. Posteriormente a se encontrar os melhores modelos, construiu-se um *web service* que serviu para validar o detetor de notícias falsas em língua portuguesa que corresponde ao modelo C.

8.2 Contribuições Principais

Nos dias de hoje, as notícias são facilmente divulgadas de forma rápida e desorganizada pelo mundo através dos vários meios de comunicação social *online*. As notícias falsas são cada vez mais um problema que envolve a sociedade no seu todo e não existe uma resposta imediata e eficaz para combater o mesmo.

No que se refere ao estudo do estado da arte relativo a ferramentas de deteção de notícias falsas verificou-se que existe um número maior de projetos para uso em língua inglesa. No entanto, encontrou-se apenas um projeto desenvolvido para a deteção de notícias falsas em português europeu.

Relativamente ao presente trabalho, construiu-se um *dataset* de raiz de forma semi-automática, exclusivamente em língua portuguesa com base em sites de notícias fidedignos e não fidedignos. Este foi anotado de forma manual em quatro classes. Ao conjunto de dados foram aplicadas diferentes técnicas de *Text Mining* e exploraram-se duas abordagens (multiclasses e binária). Os resultados presentes na abordagem multiclasses refletiu uma subjetividade nos critérios utilizados e assim implementou-se uma nova reanotação do *dataset* em apenas duas classes. Uma das experiências presentes na abordagem binária, baseou-se numa arquitetura hierárquica e inovadora composta por três modelos. O primeiro modelo efetua uma análise textual da notícia, o segundo analisa os metadados presentes na mesma e o terceiro destina-se a efetuar uma ponderação entre os dois modelos descritos anteriormente para otimizar os resultados. O projeto desenvolvido tem como contribuições principais: a apresentação de dois índices de credibilidade referentes ao conteúdo e aos metadados de uma determinada notícia e ainda um detetor de notícias falsas com 92 % de *performance* em língua portuguesa. Esta informação encontra-se sistematizada na publicação submetida na Conferência denominada *IEEE ISTAS21 (Engineering and Corporate Social Responsibility)*

com o título *"A fake news detection and credibility ranking platform for Portuguese online news analysis"* (Pinto, M., Rito Lima, I., Amorim, I., Ulisses, A., Marreiros, G., 2021).

8.3 Limitações e Trabalho Futuro

No entanto, apesar de os resultados de toda a solução serem bastante positivos e a proposta ser inovadora em vários sentidos, é possível identificar algumas limitações, tanto no projeto como nos resultados obtidos. Salienta-se, principalmente, a limitação associada ao número de notícias utilizadas para construir o *dataset*. O processo de extração e anotação das notícias foi bastante demorado e trabalhoso tendo em conta as várias fontes de informação existentes em Portugal. Consequentemente, o dataset desenvolvido no âmbito deste projeto compreende 708 notícias, sendo este valor relativamente inferior a outros *datasets* disponíveis, e fator de potencial enviesamento dos resultados. Considera-se que a amostra é representativa no sentido de abranger diversas fontes e níveis de credibilidade, podendo, portanto, o aumento do conjunto de dados usados para treino dos modelos ser determinante para a robustez e fiabilidade a longo prazo dos mesmos.

Ainda relativamente ao *dataset* construído, crê-se que o processo de anotação manual é, de igual forma, uma limitação do projeto, sofrendo de problemas de subjetividade inerentes à não regularização e estandardização na área relativa aos critérios de credibilidade de notícias em diversas línguas.

A outra limitação que existe diz respeito ao *web service*, pois este apenas permite validar os modelos implementados, nomeadamente, A, B e C, mas ainda assim realizar a previsão de classificação de uma determinada notícia tendo em conta as categorias e as fontes de informação presentes na página *web* desenvolvida. Ou seja, se um utilizador pretender testar uma nova notícia com uma categoria/fonte de informação diferente às que estão expostas, o modelo final não vai conseguir classificar a notícia nas três categorias anteriormente descritas (Falsa, Duvidosa e Verdadeira). É necessário implementar no modelo B um método capaz de inserir e processar novas categorias/fontes de informação.

Como trabalho futuro destacam-se dois pontos, nomeadamente: a introdução de um maior número de notícias no conjunto de dados e o desenvolvimento de um *web crawler*. No que diz respeito ao primeiro ponto, devem ser introduzidas um número equivalente de notícias falsas, proporcionando assim um equílibrio entre as duas anotações que permita alcançar um conjunto de dados mais robusto para treino dos vários algoritmos alcançados. Em relação ao segundo ponto, pode ser desenvolvido um *web crawler* que recolha as notícias de forma completamente automática de modo a que as mesmas sejam armazenadas no *dataset* de treino reduzindo dessa forma as tarefas realizadas pelo ser humano.

Referências

Afroz, S., Brennan, M. and Greenstadt, R. (2012) 'Detecting Hoaxes, Frauds, and Deception in Writing Style Online', in *2012 IEEE Symposium on Security and Privacy*. *2012 IEEE Symposium on Security and Privacy (SP) Conference dates subject to change*, San Francisco, CA, USA: IEEE, pp. 461–475. doi: 10.1109/SP.2012.34.

Agarwal, A. and Dixit, A. (2020) 'Fake News Detection: An Ensemble Learning Approach', in. *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*, pp. 1178–1183. doi: 10.1109/ICICCS48265.2020.9121030.

Ahmad, I. *et al.* (2020) 'Fake News Detection Using Machine Learning Ensemble Methods', *Complexity*. Edited by M. I. Uddin, 2020, pp. 1–11. doi: 10.1155/2020/8885861.

Ahmad, M., Reynolds, J. and Rezgui, Y. (2018) 'Predictive modelling for solar thermal energy systems: A comparison of support vector regression, random forest, extra trees and regression trees | Elsevier Enhanced Reader', pp. 1–12. doi: https://doi.org/10.1016/j.clepro.201808.207.

Ahmed, H., Traore, I. and Saad, S. (2017) *Detection of Online Fake News Using N-Gram Analysis and Machine Learning Techniques*, p. 138. doi: 10.1007/978-3-319-69155-8_9.

Ahn, Y. and Jeong, C. (2019) 'Natural Language Contents Evaluation System for Detecting Fake News using Deep Learning', in. *2019 16th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, Chonburi, Thailand: IEEE, pp. 289–292. doi: 10.1109/JCSSE.2019.8864171.

Akar, Ö. and Gungor, O. (2012) 'Classification of Multispectral Images Using Random Forest Algorithm', *Journal of Geodesy and Geoinformation*, 1, pp. 105–112. doi: 10.9733/jgg.241212.1.

Akinsola, J. E. T. (2017) 'Supervised Machine Learning Algorithms: Classification and Comparison', *International Journal of Computer Trends and Technology (IJCTT)*, 48, pp. 128–138. doi: 10.14445/22312803/IJCTT-V48P126.

Allahyari, M. *et al.* (2017) 'A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques', *arXiv:1707.02919 [cs]*. Available at: http://arxiv.org/abs/1707.02919 (Accessed: 29 January 2021).

Allcott, H. and Gentzkow, M. (2017) 'Social Media and Fake News in the 2016 Election', *Journal of Economic Perspectives*, 31(2), pp. 211–236. doi: 10.1257/jep.31.2.211.

Ayodele, T. O. (2010) *Types of Machine Learning Algorithms, New Advances in Machine Learning*. IntechOpen. doi: 10.5772/9385.

Balakrishnan, V. and Ethel, L.-Y. (2014) 'Stemming and Lemmatization: A Comparison of Retrieval Performances', *Lecture Notes on Software Engineering*, 2(3), pp. 262–267. doi: 10.7763/LNSE.2014.V2.134.

Bell, D. (2004) 'UML basics: The component diagram', p. 7.

Beniwal, S. and Arora, J. (2012) ‘Classification and feature selection techniques in data mining’, *International Journal of Engineering Research and Technology*, 1(6), pp. 1–6.

Bhat, S. (2019) ‘Feature Selection: Filter method, Wrapper method and Embedded method’, *Data Science Machine Learning*. Available at: http://www.datasciencesmachinelearning.com/2019/10/feature-selection-filter-method-wrapper.html (Accessed: 18 February 2021).

Bhoir, S. V. (2020) ‘An Efficient FAKE NEWS DETECTOR’, in. *2020 International Conference on Computer Communication and Informatics (ICCCI)*, India, pp. 1–9. doi: 10.1109/ICCCI48352.2020.9104177.

Bobriakov, I. (2018) *Comparison of Top 6 Python NLP Libraries, Medium*. Available at: https://medium.com/activewizards-machine-learning-company/comparison-of-top-6-python-nlp-libraries-c4ce160237eb (Accessed: 3 February 2021).

Bondielli, A. and Marcelloni, F. (2019) ‘A survey on fake news and rumour detection techniques’, *Information Sciences*, 497, pp. 38–55. doi: 10.1016/j.ins.2019.05.035.

Breiman, L. and Cutler, A. (2004) *Random forests - classification description*. Available at: https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm (Accessed: 27 February 2021).

Burgoon, J. K. *et al.* (2003) ‘Detecting Deception through Linguistic Analysis’, in Chen, H. et al. (eds) *Intelligence and Security Informatics*. Berlin, Heidelberg: Springer (Lecture Notes in Computer Science), pp. 91–101. doi: https://doi.org/10.1007/3-540-44853-5_7.

Cambridge University Press (2008) *Stemming and lemmatization*. Available at: https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html (Accessed: 17 February 2021).

Chen, Y., Conroy, N. J. and Rubin, V. L. (2015) ‘Misleading Online Content: Recognizing Clickbait as “False News”’, in *Proceedings of the 2015 ACM on Workshop on Multimodal Deception Detection*. New York, NY, USA: Association for Computing Machinery (WMDD ’15), pp. 15–19. doi: 10.1145/2823465.2823467.

CherryPy, T. (2001) *CherryPy — A Minimalist Python Web Framework*. Available at: https://cherrypy.org/ (Accessed: 8 June 2021).

Correia, C. (2020) *Apanhado na Fake - Apanhados na fake*. Available at: https://www.apanhadonafake.pt/recursos/apanhados-na-fake (Accessed: 3 November 2020).

Dannana, S. (2020) ‘Function Analysis and System Technique - FAST diagram’, *ExtruDesign*, 19 February. Available at: https://extrudesign.com/function-analysis-and-system-technique-fast-diagram/ (Accessed: 12 February 2021).

Davis, W. (2016) *Fake Or Real? How To Self-Check The News And Get The Facts, NPR.org*. Available at: https://www.npr.org/sections/alltechconsidered/2016/12/05/503581220/fake-or-real-how-to-self-check-the-news-and-get-the-facts (Accessed: 25 February 2021).

DeLand, S. (2019) *When to use Machine Learning or Deep Learning?* Available at: https://www.embedded-computing.com/guest-blogs/when-to-use-machine-learning-or-deep-learning (Accessed: 2 November 2020).

Developer, G. (2018) *What is Clustering? | Clustering in Machine Learning, Clustering in Machine Learning*. Available at: https://developers.google.com/machine-learning/clustering/overview (Accessed: 2 November 2020).

Eeles, P. (2004) *What, no supplementary specification?* Available at: http://www.ibm.com/developerworks/rational/library/3975.html (Accessed: 2 February 2021).

Egelhofer, J. L. and Lecheler, S. (2019) 'Fake news as a two-dimensional phenomenon: a framework and research agenda', *Annals of the International Communication Association*, 43(2), pp. 97–116. doi: 10.1080/23808985.2019.1602782.

Escudeiro, P. and Escudeiro, N. (2012) 'Evaluation of Serious Games in Mobile Platforms with QEF: QEF (Quantitative Evaluation Framework)', in. *Mobile and Ubiquitous Technology in Education 2012 IEEE Seventh International Conference on Wireless*, Takamatsu, Japan: IEEE, pp. 268–271. doi: 10.1109/WMUTE.2012.65.

European Data Portal | European Data Portal (2016). Available at: https://www.europeandataportal.eu/en/about/european-data-portal (Accessed: 16 November 2020).

Feldman, R. and Dagan, I. (1995) 'Knowledge Discovery in Textual Databases (KDT)', *KDT*, pp. 112–117.

Feldman, S. (1999) 'NLP Meets the Jabberwocky: Natural Language Processing in Information Retrieval', *Proceedings of the Fifth Hong Kong Web Symposium*, 23, pp. 62–73. doi: 10.1.1.708.5482.

Figueira, Á., Guimarães, N. and Pinto, J. (2019) 'A System to Automatically Predict Relevance in Social Media', *Procedia Computer Science*, 164, pp. 105–112. doi: 10.1016/j.procs.2019.12.160.

Figueira, Á., Guimarães, N. and Torgo, L. (2019) 'A Brief Overview on the Strategies to Fight Back the Spread of False Information', *Journal of Web Engineering (JWE)*, 18(Combined Issue 4, 5 & 6), pp. 319–352. doi: 10.13052/jwe1540-9589.18463.

Finkelstein, E. (2005) *Syndicating web sites with RSS feeds for dummies*. Hoboken, N.J: John Wiley (--For dummies). Available at: https://books.google.pt/books?hl=pt-PT&lr=&id=047J5iZDBtMC&oi=fnd&pg=PR5&dq=Syndicating+web+sites+with+RSS+feeds+for+dummies&ots=2Y089cG6Nk&sig=bfB-h-OMSgBasaQjTyu4pizDBR4&redir_esc=y#v=onepage&q=Syndicating%20web%20sites%20with%20RSS%20feeds%20for%20dummies&f=false (Accessed: 29 November 2020).

Flask (2010) *Welcome to Flask — Flask Documentation (2.0.x)*. Available at: https://flask.palletsprojects.com/en/2.0.x/ (Accessed: 8 June 2021).

Foundation, D. S. (2005) *Django overview | Django*. Available at: https://www.djangoproject.com/start/overview/ (Accessed: 8 June 2021).

Fuller, C. M., Biros, D. P. and Wilson, R. L. (2009) ‘Decision support for determining veracity via linguistic-based cues’, *Decision Support Systems*, 46(3), pp. 695–703. doi: 10.1016/j.dss.2008.11.001.

Gaikwad, D. and Thool, R. (2015) ‘Intrusion Detection System Using Bagging Ensemble Method of Machine Learning’, pp. 291–295. doi: 10.1109/ICCUBEA.2015.61.

Gandhi, R. (2018a) *Naive Bayes Classifier*, *Medium*. Available at: https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c (Accessed: 27 February 2021).

Gandhi, R. (2018b) *Support Vector Machine — Introduction to Machine Learning Algorithms*, *Medium*. Available at: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47 (Accessed: 27 February 2021).

Gaonkar, S. *et al.* (2019) ‘Detection Of Online Fake News:A Survey’, in. *2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN)*, Vellore, India: IEEE. doi: 10.1109/ViTECoN.2019.8899556.

Gohil, L. (2015) ‘Text Mining: Process and Techniques’, 3(3), pp. 70–72.

Granik, M. and Mesyura, V. (2017) ‘Fake news detection using naive Bayes classifier’, in. *2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON)*, Kyiv, UKraine: IEEE, pp. 900–903. doi: 10.1109/UKRCON.2017.8100379.

Gravanis, G. *et al.* (2019) ‘Behind the cues: A benchmarking study for fake news detection’, 128, pp. 201–213. doi: 10.1016/j.eswa.2019.03.036.

Gupta, A. and Kaushal, R. (2015) ‘Improving spam detection in Online Social Networks’, in. *Proceedings - 2015 International Conference on Cognitive Computing and Information Processing, CCIP 2015*, Noida, India: IEEE. doi: 10.1109/CCIP.2015.7100738.

Gupta, P. (2017) *Decision Trees in Machine Learning*, *Medium*. Available at: https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052 (Accessed: 27 February 2021).

Gupta, V. and Lehal, G. S. (2009) ‘A Survey of Text Mining Techniques and Applications’, *Journal of Emerging Technologies in Web Intelligence*, 1(1), pp. 60–76. doi: 10.4304/jetwi.1.1.60-76.

Harrison, O. (2019) *Machine Learning Basics with the K-Nearest Neighbors Algorithm*, *Medium*. Available at: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761 (Accessed: 27 February 2021).

Hassan, E. A. and Meziane, F. (2019) ‘A Survey on Automatic Fake News Identification Techniques for Online and Socially Produced Data’, in. *2019 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)*, Khartoum, Sudan: IEEE, pp. 1–6. doi: 10.1109/ICCCEEE46830.2019.9070857.

Hearst, M. A. *et al.* (1998) ‘Support vector machines’, *IEEE Intelligent Systems and their Applications*, 13(4), pp. 18–28. doi: 10.1109/5254.708428.

Heidenreich, H. (2018) *Stemming? Lemmatization? What?*, *Medium*. Available at: https://towardsdatascience.com/stemming-lemmatization-what-ba782b7c0bd8 (Accessed: 17 February 2021).

Hewitt, B. (2017) *How to spot fake news – an expert's guide for young people, The Conversation*. Available at: http://theconversation.com/how-to-spot-fake-news-an-experts-guide-for-young-people-88887 (Accessed: 2 November 2020).

Hofseth, A. (2017) *Fake News, Propaganda, and Influence Operations – a guide to journalism in a new, and more chaotic media environment, Reuters Institute for the Study of Journalism*. Available at: https://reutersinstitute.politics.ox.ac.uk/risj-review/fake-news-propaganda-and-influence-operations-guide-journalism-new-and-more-chaotic (Accessed: 1 March 2021).

IFLA (2021) *IFLA -- How To Spot Fake News*. Available at: https://www.ifla.org/publications/node/11174 (Accessed: 22 February 2021).

Ireton, C., Posetti, J., and UNESCO (2018) *Journalism, 'fake news' et disinformation: handbook for journalism education and training*. Available at: http://unesdoc.unesco.org/images/0026/002655/265552E.pdf (Accessed: 31 January 2021).

Jabeen, H. (2018) *Stemming and Lemmatization in Python*. Available at: https://www.datacamp.com/community/tutorials/stemming-lemmatization-python (Accessed: 17 February 2021).

Jaroucheh, Z. *et al.* (2020) 'TRUSTD: Combat Fake Content using Blockchain and Collective Signature Technologies', in. *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*, Toronto, Canada: Willieam Buchanan, pp. 1235–1240. doi: 10.1109/COMPSAC48688.2020.00-87.

Kacprzyk, J. (2005) 'Studies in Fuzziness and Soft Computing, Volume 185', *05/2005*, 185, p. 285.

Kaliyar, R. K., Goswami, A. and Narang, P. (2019) 'Multiclass Fake News Detection using ensemble machine learning', in. *2019 IEEE 9th International Conference on Advanced Computing (IACC)*, Tiruchirappalli, India: IEEE, pp. 103–107. doi: 10.1109/IACC48062.2019.8971579.

Kalsnes, B. (2018) 'Fake News', in. Oxford Research Encyclopedia of Communication. doi: https://doi.org/10.1093/acrefore/9780190228613.013.809.

Kannan, D. S. and Gurusamy, V. (2014) 'Preprocessing Techniques for Text Mining', 5(1), pp. 7–16.

Kao, A. and Poteet, S. R. (eds) (2007) *Natural language processing and text mining*. London: Springer Science \& Business Media.

Kittler, J. and Roli, F. (eds) (2000) *Multiple classifier systems: first international workshop, MCS 2000, Cagliari, Italy, June 21-23, 2000: proceedings*. *International Workshop on Multiple Classifier Systems*, Berlin ; New York: Springer (Lecture notes in computer science, 1857).

Koen, P. *et al.* (2016) 'Providing Clarity and A Common Language to the "Fuzzy Front End"', *Research-Technology Management*, 44(2), pp. 46–55. doi: 10.1080/08956308.2001.11671418.

Kong, S. H. *et al.* (2020) 'Fake News Detection using Deep Learning', in. *2020 IEEE 10th Symposium on Computer Applications Industrial Electronics (ISCAIE)*, Malaysia: IEEE, pp. 102–107. doi: 10.1109/ISCAIE47305.2020.9108841.

Kuhn, M. and Johnson, K. (2019) *Feature Engineering and Selection: A Practical Approach for Predictive Models*. CRC Press. Available at: 12/02/2021.

Ladicky, L. and Torr, P. (2011) 'Linear Support Vector Machines', in *Proceedings of the 28th International Conference on Machine Learning*. *ICML 2011*, Washington, USA, p. 8. Available at: https://www.researchgate.net/publication/221345963_Linear_Support_Vector_Machines (Accessed: 26 February 2021).

Lee, S., Song, J. and Kim, Y. (2010) 'An Empirical Comparison of Four Text Mining Methods', *Journal of Computer Information Systems*, p. 11.

Lever, J., Krzywinski, M. and Altman, N. (2016) 'Logistic regression', *Nature Methods*, 13(7), pp. 541–542. doi: 10.1038/nmeth.3904.

Lin, X., Zhao, C. and Pan, W. (2017) 'Towards Accurate Binary Convolutional Neural Network', *arXiv:1711.11294 [cs, stat]*. Available at: http://arxiv.org/abs/1711.11294 (Accessed: 15 June 2021).

Li-Ping Jing, Hou-Kuan Huang, and Hong-Bo Shi (2002) 'Improved feature selection approach TFIDF in text mining', in. *International Conference on Machine Learning and Cybernetics*, Beijing, China: IEEE, pp. 944–946 vol.2. doi: 10.1109/ICMLC.2002.1174522.

Lovins, J. B. (1968) 'Development of a stemming algorithm', 11, p. 10.

Mahid, Z. I., Manickam, S. and Karuppayah, S. (2018) 'Fake News on Social Media: Brief Review on Detection Techniques', in. *2018 Fourth International Conference on Advances in Computing, Communication Automation (ICACCA)*, Subang Jaya, Malaysia: IEEE, pp. 1–5. doi: 10.1109/ICACCAF.2018.8776689.

Mandical, R. R. *et al.* (2020) 'Identification of Fake News Using Machine Learning', in. *2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, Bangalore, India: IEEE, pp. 1–6. doi: 10.1109/CONECCT50063.2020.9198610.

Markines, B., Cattuto, C. and Menczer, F. (2009) 'Social spam detection', in *Proceedings of the 5th International Workshop on Adversarial Information Retrieval on the Web*. New York, NY, USA: Association for Computing Machinery (AIRWeb '09), pp. 41–48. doi: 10.1145/1531914.1531924.

Menezes, B. (2020) *O consumo de informação na era das fake news, MindMiners Blog*. Available at: https://mindminers.com/blog/fake-news/ (Accessed: 4 November 2020).

Mohan, V. (2015) ‘Preprocessing Techniques for Text Mining - An Overview’, *International Journal of Computer Science & Communication Networks*, 5, pp. 7–16.

Müller, A. C. and Guido, S. (2016) *Introduction to Machine Learning with Python: A Guide for Data Scientists*. 1st edn. Edited by S. Dawn. O’Reilly Media, Inc.

Narkhede, S. (2021) *Understanding Confusion Matrix, Medium*. Available at: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62 (Accessed: 6 February 2021).

Ng, J. (2019) *Top 10 Best News APIs: Google News, Bloomberg, BING News and more, Medium*. Available at: https://medium.com/rakuten-rapidapi/top-10-best-news-apis-google-news-bloomberg-bing-news-and-more-bbf3e6e46af6 (Accessed: 16 November 2020).

Nielsen, J. (2013) *Website Reading: It (Sometimes) Does Happen, Nielsen Norman Group*. Available at: https://www.nngroup.com/articles/website-reading/ (Accessed: 2 November 2020).

Pant, A. (2019) *Introduction to Logistic Regression, Medium*. Available at: https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148 (Accessed: 27 February 2021).

Parikh, S. B. and Atrey, P. K. (2018) ‘Media-Rich Fake News Detection: A Survey’, in. *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, Miami, FL, USA: IEEE, pp. 436–441. doi: 10.1109/MIPR.2018.00093.

Patel, F. N. and Soni, N. R. (2012) ‘Text mining: A Brief survey’, 2(6), p. 4. doi: 10.1.1.300.9006.

Pena, P. (2019) *Nove passos para distinguir informação de fake news - DN*. Available at: https://www.dn.pt/edicao-do-dia/05-out-2019/nove-passos-para-distinguir-informacao-de-fake-news-11371288.html (Accessed: 2 November 2020).

Pessanha, C. P. (2019) *Random Forest: como funciona um dos algoritmos mais populares de ML | by Cínthia Pessanha | cinthiabpessanha | Medium*. Available at: https://medium.com/cinthiabpessanha/random-forest-como-funciona-um-dos-algoritmos-mais-populares-de-ml-cc1b8a58b3b4 (Accessed: 2 November 2020).

Peterson, L. E. (2009) ‘K-nearest neighbor’, *Scholarpedia*, 4(2), p. 1883. doi: 10.4249/scholarpedia.1883.

Pinto, M., Rito Lima, I., Amorim, I., Ulisses, A., Marreiros, G. (2021) ‘A fake news detection and credibility ranking platform for Portuguese online news analysis’, in. *IEEE ISTAS21 (Engineering and Corporate Social Responsibility)*, IEEE (under revision).

Posadas Durán, J. *et al.* (2019) ‘Detection of fake news in a new corpus for the Spanish language’, *Journal of Intelligent & Fuzzy Systems*, 36, pp. 4869–4876. doi: 10.3233/JIFS-179034.

Premalatha, R. and Srinivasan, S. (2014) ‘Text processing in information retrieval system using vector space model’, in. *International Conference on Information Communication and*

Embedded Systems (ICICES2014), Chennai, India: IEEE, pp. 1–6. doi: 10.1109/ICICES.2014.7033837.

Pressman, R. S. (2015) *Software engineering: a practitioner's approach*. Eighth edition. New York, NY: McGraw-Hill Education. Available at: https://books.google.pt/books?hl=pt-PT&lr=&id=bL7QZHtWvaUC&oi=fnd&pg=PR30&dq=Software+engineering:+a+practitioner%E2%80%99s+approach.&ots=O8uc5RrMaj&sig=MZjTeZSpEIi4xjKhf_DyErqMq8k&redir_esc=y#v=onepage&q=Software%20engineering%3A%20a%20practitioner%E2%80%99s%20approach.&f=false.

Quinlan, J. R. (1996) 'Learning decision tree classifiers', *ACM Computing Surveys*, 28(1), pp. 71–72. doi: 10.1145/234313.234346.

Raschka, S. (2020) 'Model Evaluation, Model Selection, and Algorithm Selection in Machine Learning', *arXiv:1811.12808 [cs, stat]*. Available at: http://arxiv.org/abs/1811.12808 (Accessed: 9 June 2021).

Rasool, T. *et al.* (2019) 'Multi-Label Fake News Detection using Multi-layered Supervised Learning', in. *the 2019 11th International Conference*, Perth, WN, Australia: ACM Press, pp. 73–77. doi: 10.1145/3313991.3314008.

Ribeiro, M. T., Singh, S. and Guestrin, C. (2016) '"Why Should I Trust You?": Explaining the Predictions of Any Classifier', in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: Association for Computing Machinery (KDD '16), pp. 1135–1144. doi: 10.1145/2939672.2939778.

Rich, N. and Holweg, M. (2000) *Value Analysis*. Cardiff, United Kingdom: Lean Enterprise Research Centre, pp. 1–32. Available at: https://www.urenio.org/tools/en/value_analysis.pdf (Accessed: 24 December 2020).

Ridgeway, G. (1999) *The State of Boosting*.

Rish, I. (2001) 'An empirical study of the naive bayes classifier', *IJCAI 2001 workshop on empirical methods in artificial intelligence*, pp. 41–46.

Rodrigues, J. F. C. (2020) *Fake News Classification in European Portuguese Language*. Available at: https://repositorio.iscte-iul.pt/bitstream/10071/22194/1/master_joao_carrico_rodrigues.pdf.

Roell, J. (2017) *Understanding Recurrent Neural Networks: The Preferred Neural Network for Time-Series Data | by Jason Roell | Towards Data Science*. Available at: https://towardsdatascience.com/understanding-recurrent-neural-networks-the-prefered-neural-network-for-time-series-data-7d856c21b759 (Accessed: 27 February 2021).

Rout, A. (2020) 'Advantages and Disadvantages of Logistic Regression', *GeeksforGeeks*, 25 August. Available at: https://www.geeksforgeeks.org/advantages-and-disadvantages-of-logistic-regression/ (Accessed: 2 November 2020).

Roy, A. *et al.* (2018) 'A Deep Ensemble Framework for Fake News Detection and Classification', *Fake News Detection using Deep Learning*, 12 November. Available at: http://arxiv.org/abs/1811.04670 (Accessed: 1 February 2021).

Saeys, Y., Inza, I. and Larrañaga, P. (2007) ‘A review of feature selection techniques in bioinformatics’, *Bioinformatics*, 23(19), pp. 2507–2517. doi: 10.1093/bioinformatics/btm344.

Saha, S. (2018) *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*, *Medium*. Available at: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (Accessed: 27 February 2021).

Saini, N. *et al.* (2020) ‘Multimodal, Semi-supervised and Unsupervised web content credibility analysis Frameworks’, in. *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*, Madurai, India: IEE, pp. 948–955. doi: 10.1109/ICICCS48265.2020.9121005.

Sakuma, A. and Saliba, E. (2016) *Would you believe the pope endorsed Trump? Five tips for spotting fake news*, *NBC News*. Available at: https://www.nbcnews.com/news/us-news/five-tips-how-spot-fake-news-online-n687226 (Accessed: 4 November 2020).

Saunders, C. and Grobelnik, M. (2005) *Subspace, Latent Structure and Feature Selection*. Slovenia. Available at: https://link.springer.com/content/pdf/10.1007/11752790.pdf.

Schapire, R. E. (2013) ‘Explaining AdaBoost’, pp. 1–16.

Scholz, J. (2019) *LibGuides: Fake News: Types of Fake News*. Available at: https://cameron.libguides.com/FakeNews/TypesOf (Accessed: 30 January 2021).

Shabani, S. and Sokhn, M. (2018) ‘Hybrid Machine-Crowd Approach for Fake News Detection’, in. *2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)*, Philadelphia, PA, USA: IEEE, pp. 299–306. doi: 10.1109/CIC.2018.00048.

Sharma, S. and Sharma, D. K. (2019b) ‘Fake News Detection: A long way to go’, in. *2019 4th International Conference on Information Systems and Computer Networks (ISCON)*, Mathura, India: IEEE, pp. 816–821. doi: 10.1109/ISCON47742.2019.9036221.

Shu, K. *et al.* (2019) ‘dEFEND: Explainable Fake News Detection’, in. *KDD ’19: The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, Anchorage AK USA: ACM, pp. 395–405. doi: 10.1145/3292500.3330935.

Shu, K. *et al.* (2020) ‘Combating disinformation in a social media age’, *WIREs Data Mining and Knowledge Discovery*, 10(6), p. e1385. doi: 10.1002/widm.1385.

Shung, K. P. (2020) *Accuracy, Precision, Recall or F1?*, *Medium*. Available at: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9 (Accessed: 25 February 2021).

Siering, M., Koch, J.-A. and Deokar, A. V. (2016) ‘Detecting Fraudulent Behavior on Crowdfunding Platforms: The Role of Linguistic and Content-Based Cues in Static and Dynamic Contexts’, *Journal of Management Information Systems*, 33(2), pp. 421–455. doi: 10.1080/07421222.2016.1205930.

Silva, R. M. *et al.* (2020) ‘Towards automatically filtering fake news in Portuguese’, *Expert Systems with Applications*, 146, p. 113199. doi: 10.1016/j.eswa.2020.113199.

Singh, H. (2018) *Understanding Gradient Boosting Machines | by Harshdeep Singh | Towards Data Science*. Available at: https://towardsdatascience.com/understanding-gradient-boosting-machines-9be756fe76ab (Accessed: 27 February 2021).

Sintra, M. C. D. (2019) *Fake News e a Desinformação:* Faculdade de Ciências Sociais e Humanas Universidade Nova de Lisboa. Available at: https://run.unl.pt/bitstream/10362/79564/1/Fake%20News%20e%20a%20Desinforma%C3%A7%C3%A3o_Perspetivar%20comportamentos%20e%20estrat%C3%A9gias%20informacionais.pdf.

Snell, N. *et al.* (2019) 'Manually Classified Real and Fake News Articles', in. *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, Las Vegas, NV, USA: IEEE, pp. 1405–1407. doi: 10.1109/CSCI49370.2019.00262.

Sommerville, I. (2005) 'Integrated requirements engineering: a tutorial', *IEEE Software*, 22(1), pp. 16–23. doi: 10.1109/MS.2005.13.

Srinivasan, A. V. (2019) *Stochastic Gradient Descent — Clearly Explained !!*, *Medium*. Available at: https://towardsdatascience.com/stochastic-gradient-descent-clearly-explained-53d239905d31 (Accessed: 26 February 2021).

Sukanya, M. and Biruntha, S. (2012) 'Techniques on text mining', in. *2012 IEEE International Conference on Advanced Communication Control and Computing Technologies (ICACCCT)*, Ramanathapuram, India: IEEE, pp. 269–271. doi: 10.1109/ICACCCT.2012.6320784.

Teza, P., Brandão Miguez, V., *et al.* (2013) 'Direcionadores do processo de inovação: o papel da estratégia, liderança e cultura', *Navus - Revista de Gestão e Tecnologia*, pp. 77–88. doi: 10.22279/navus.2013.v3n2.p77-88.156.

Thomas, S. (2020) *Library: Real News vs. Fake News: Fake News*. Available at: https://libguides.pace.edu/fakenews/fakenews (Accessed: 2 November 2020).

Thota, A. *et al.* (2018) 'Fake News Detection: A Deep Learning Approach', in *SMU Data Science*. Dallas: SMU Scholar (3), p. 21. Available at: https://scholar.smu.edu/datasciencereview/vol1/iss3/10 (Accessed: 15 December 2020).

Torabi Asr, F. and Taboada, M. (2019) 'Big Data and quality data for fake news and misinformation detection', *Big Data & Society*, 6(1), p. 2053951719843310. doi: 10.1177/2053951719843310.

Tripepi, G. *et al.* (2008) 'Linear and logistic regression analysis | Elsevier Enhanced Reader', *Kidney International*, 73(7), pp. 806–810. doi: https://doi.org/10.1038/sj.ki.5002787.

Turk, D. Z. (2018) 'Technology as Enabler of Fake News and a Potential Tool to Combat It'. Available at: https://www.europarl.europa.eu/RegData/etudes/IDAN/2018/619008/IPOL_IDA(2018)619008_EN.pdf (Accessed: 23 February 2020).

Twin, A. (2020) *Value Proposition: Why Consumers Should Buy a Product or Use a Service*, *Investopedia*. Available at: https://www.investopedia.com/terms/v/valueproposition.asp (Accessed: 12 February 2021).

University of Connecticut (2021) *How Can I Tell if a News Story is Fake? - How to Identify Fake News - LibGuides at University of Connecticut*. Available at: https://guides.lib.uconn.edu/fakenews/whatandhowtocheckfakenews (Accessed: 2 November 2020).

Uppal, A., Sachdeva, V. and Sharma, S. (2020) 'Fake news detection using discourse segment structure analysis', in. *2020 10th International Conference on Cloud Computing, Data Science Engineering (Confluence)*, Noida, India: IEEE, pp. 751–756. doi: 10.1109/Confluence47617.2020.9058106.

Verma, T., Renu, R. and Gaur, D. (2014) 'Tokenization and Filtering Process in RapidMiner', *International Journal of Applied Information Systems*, 7(2), pp. 16–18. doi: 10.5120/ijais14-451139.

Vida, I. K. (2012) 'The "Great Moon Hoax" of 1835', *Hungarian Journal of English and American Studies (HJEAS)*, 18(1/2), pp. 431–441.

VijayGaikwad, S., Chaugule, A. and Patil, P. (2014) 'Text Mining Methods and Techniques', *International Journal of Computer Applications*, 85(17), pp. 42–45. doi: 10.5120/14937-3507.

Wardle, C. (2018) 'The Need for Smarter Definitions and Practical, Timely Empirical Research on Information Disorder', *Digital Journalism*, 6(8), pp. 951–963. doi: 10.1080/21670811.2018.1502047.

Webb, G. I. and Yu, X. H. (eds) (2004) *AI 2004: advances in artificial intelligence: 17th Australian Joint Conference on Artificial Intelligence, Cairns, Australia, December 4-6, 2004: proceedings*. *Australian Joint Conference on Artificial Intelligence*, Berlin: Springer (Lecture notes in artificial intelligence. Lecture notes in computer science, 3339).

Wong, T.-T. (2015) 'Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation | Elsevier Enhanced Reader', *Pattern Recognition*, 48(9), pp. 2839–2846. doi: https://doi.org/10.1016/j.patcog.2015.03.009.

Zaremba, W., Sutskever, I. and Vinyals, O. (2015) 'Recurrent Neural Network Regularization', *arXiv:1409.2329 [cs]*. Available at: http://arxiv.org/abs/1409.2329 (Accessed: 15 June 2021).

Zervopoulos, A. *et al.* (2020) 'Hong Kong Protests: Using Natural Language Processing for Fake News Detection on Twitter', in Maglogiannis, I., Iliadis, L., and Pimenidis, E. (eds) *Artificial Intelligence Applications and Innovations*. Cham: Springer International Publishing (IFIP Advances in Information and Communication Technology), pp. 408–419. doi: 10.1007/978-3-030-49186-4_34.

Zhang, J. *et al.* (2018) 'Long short-term memory for machine remaining life prediction', *Journal of Manufacturing Systems*, 48, pp. 78–86. doi: 10.1016/j.jmsy.2018.05.011.

Zhang, S. *et al.* (2015) 'Bidirectional Long Short-Term Memory Networks for Relation Classification', in *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation*. *PACLIC 2015*, Shanghai, China, pp. 73–78. Available at: https://www.aclweb.org/anthology/Y15-1009 (Accessed: 27 February 2021).

Zhang, X. and Ghorbani, A. A. (2020) ‘An overview of online fake news: Characterization, detection, and discussion’, *Information Processing & Management*, 57(2), p. 102025. doi: 10.1016/j.ipm.2019.03.004.

Zheng, A. and Casari, A. (2018) *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists*. O’Reilly Media, Inc.

Zheng, R. and Zhang, Y. (2012) ‘Design and implementation of news collecting and filtering system based on RSS’, in. *2012 9th International Conference on Fuzzy Systems and Knowledge Discovery*, Chongqing, China: IEEE, pp. 2295–2298. doi: 10.1109/FSKD.2012.6234158.

Zhou, X. *et al.* (2019) *Fake News: Fundamental Theories, Detection Strategies and Challenges*. New York, NY, USA: Association for Computing Machinery ({WSDM ’19}). doi: 10.1145/3289600.3291382.

Anexos A – Implementação da solução

Relativamente às figuras expostas (Figura 78, Figura 79 e Figura 80) no Anexo A, as mesmas foram abordadas na Secção 6.1, 6.2 e 6.6.

Figura 78 Conjunto de sites não fidedignos (Sintra, 2019)

Title	URL	Description	Category	Publishing Date	Source	Year	Author	Content	Published Time	Classification
Professora morre cc	https://w	Autoridades e	Mundo	18/03/2021	Diário Lus	2021	Redação	A mulher tinha tomado a primeira dose da vac		3
Homem que foi mul	https://w	O homem que	Insólito	28/03/2021	Diário Lus	2021	Redação	"Boa tarde, agradeço desde já todo o apoio. De		4
OMS: SARS-CoV-2 d	http://fee	Os voos entre	Coronavíri	29/03/2021	Publico RS	2021	Inês Chaíç	Um novo estudo da Organ	07:58	3

Figura 79 Conjunto de dados

Figura 80 Notícia classificada como Verdadeira

Figura 81 Notícia classificada como Duvidosa

Anexos B – Usabilidade do Web Service

A Tabela 50 e as Figura 82, Figura 83, Figura 84, Figura 85, Figura 86 foram abordadas na Secção 7.5.

Tabela 50 Questões efetuadas e a escala de respostas no questionário

Questões	**Métrica de Avaliação**	**Escala de resposta**
Classifique com base na intuição o grau de utilização do detetor de notícias falsas	Intuição	3 Respostas (1 – Pouco Intuitivo; 2 – Mediano; 3 – Muito Intuitivo)
No que diz respeito à utilidade classifique o detetor	Utilidade	3 Respostas (1 – Inútil; 2 - Mediano; 3 – Útil)
Classifique com base na rapidez deste detetor	Rapidez	3 Respostas (1 – Lento; 2 – Mediano; 3- Rápido)
No que diz respeito à interpretação do resultado classifique este detetor	Facilidade de Interpretação	3 Respostas (1 – Díficil; 2 – Mediano; 3 - Fácil)
Recomenda ou não a utilização deste detetor de notícias falsas a outras pessoas	Recomendação	2 Respostas (1 – Não Recomenda; 2 – Recomendado)

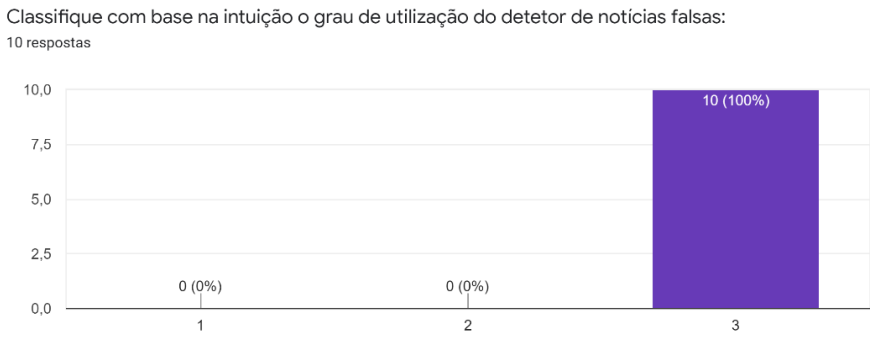


Figura 82 Resultados sobre a intuição do detetor

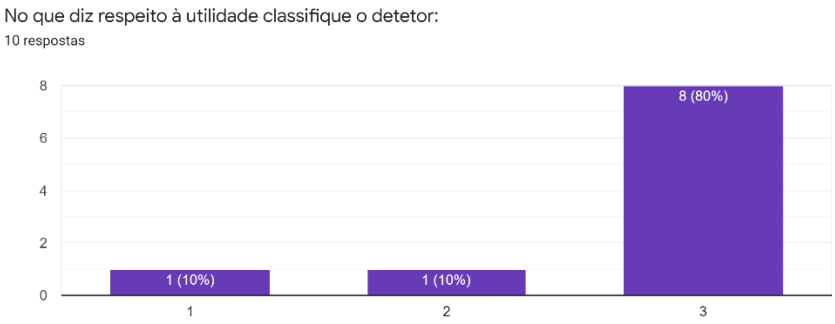


Figura 83 Resultados sobre a utilidade do detetor

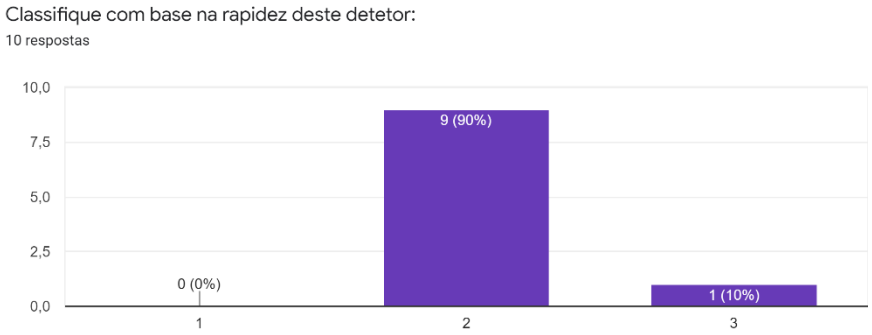


Figura 84 Resultados sobre a rapidez do detetor

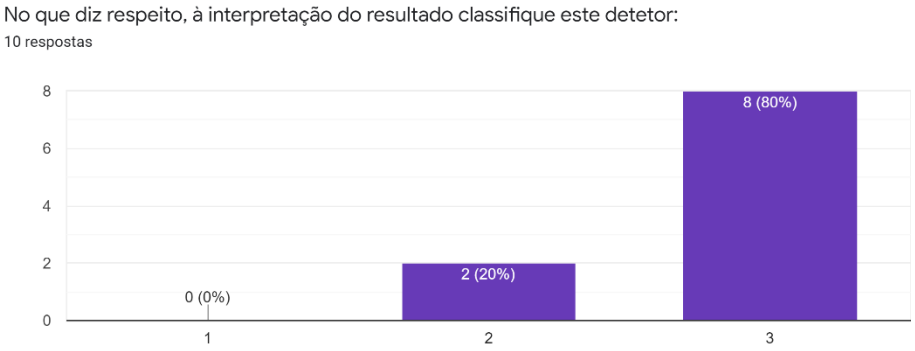


Figura 85 Resultados sobre a interpretação dos resultados do detetor

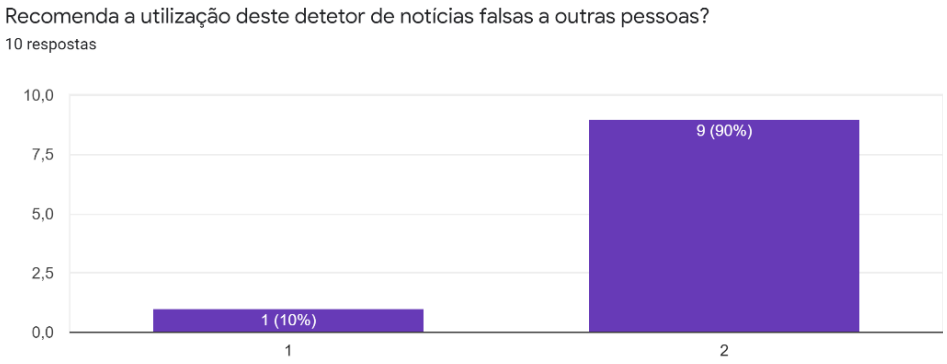


Figura 86 Resultados sobre a recomendação do detetor