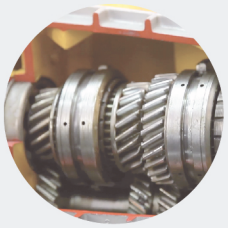
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA E GESTÃO INDUSTRIAL






Modelos de Previsão Aplicados no Setor Automóvel

GONÇALO FACEIRA TEIXEIRA DA FONSECA CARDOSO
outubro de 2023

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

MODELOS DE PREVISÃO APLICADOS NO SETOR AUTOMÓVEL

Gonçalo Faceira Teixeira da Fonseca Cardoso

2023

Instituto Superior de Engenharia do Porto

Departamento de Engenharia Mecânica

P.PORTO


P.PORTO

Instituto Superior de Engenharia do Porto

MODELOS DE PREVISÃO APLICADOS NO SETOR AUTOMÓVEL

Gonçalo Faceira Teixeira da Fonseca Cardoso

Estudante n.º 1150460

Dissertação apresentada ao Instituto Superior de Engenharia do Porto para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia e Gestão Industrial, realizada sob a orientação do Doutora Sandra Cristina de Faria Ramos.

2023

Instituto Superior de Engenharia do Porto

Departamento de Engenharia Mecânica

P.PORTO


P.PORTO

AGRADECIMENTOS

Gostaria de expressar os meus sinceros agradecimentos às seguintes pessoas que foram importantes apoios e incentivos para a conclusão desta dissertação e que sem os quais não se teria tornado uma realidade.

Ao coordenador do Mestrado de Engenharia de Gestão Industrial, Manuel Pereira Lopes, por toda a dedicação em oferecer um ambiente académico propício ao aprendizado, além de nos aconselhar e direcionar para a escolha do tema da dissertação mais acertado às nossas características e preferências.

À minha orientadora, Sandra Ramos, por todo o conhecimento transmitido e todos os conselhos, pois foram sem dúvida fundamentais para manter a direção correta.

A todos os docentes do ISEP que fizeram parte deste processo, por todos os conhecimentos partilhados responsáveis por expandir os meus horizontes acadêmicos.

Aos meus amigos e amigas, que sempre me apoiaram e motivaram durante toda esta caminhada, que partilharam comigo a sua experiência e também o seu conhecimento. Por sempre me desafiarem a fazer crescer, a pensar grande e a tornar o impossível no possível.

À minha namorada, Margarida Lima, por todo o apoio, carinho, compreensão, incentivo constante e que através da sua presença proporcionou equilíbrio, força e motivação, transformando esta fase desafiadora mais leve e gratificante.

Por último, tendo plena consciência que nada disto teria sido possível, dirijo um agradecimento à minha família, em especial aos meus pais, Paula Alexandra e Luis Cardoso, por me terem proporcionado a oportunidade de estar onde estou, pelo amor incondicional, apoio constante, por me incentivarem a nunca desistir e a nunca deixar de lutar pelos meus objetivos. Por acreditarem em mim em todos os momentos desta jornada.

Esta dissertação não teria sido possível sem o suporte de cada um de vocês.

Muito obrigado(a)!

página propositadamente em branco

RESUMO

Atualmente, a previsão de valores de indicadores de gestão, tais como vendas, procura, rentabilidade, etc., desempenha um papel importantíssimo no mundo empresarial, contribuindo, de forma impactante, nas decisões estratégicas das organizações. As previsões tradicionais, baseadas em fórmulas simples ou na opinião de peritos, apenas, estão a ficar obsoletas e a ser substituídas por modelos de previsão automatizados e mais modernos. Isto porque, muitas vezes, existe complexidade nas tendências dos dados disponíveis que não são corretamente detetados pelos métodos mais simples.

O presente trabalho teve como objetivo principal a aplicação de algoritmos de previsão baseados em modelos estatísticos e em métodos de aprendizagem automática para melhorar a precisão das previsões de matrículas futuras da Toyota Caetano Portugal. Os trabalhos iniciaram com uma revisão sobre métodos de previsão, seguindo-se uma análise exploratória exaustiva dos dados disponíveis e a modelação via modelos estatísticos e modelos de aprendizagem automática da procura e das vendas de automóveis da Toyota Caetano Portugal. A seleção do melhor método foi efetuada com base em métricas adequadas.

Através da análise dos resultados, observou-se que o modelo baseado em redes neuronais foi o que produziu erros mais baixos. Além disso, as previsões obtidas com esse modelo apresentaram valores próximos dos valores reais e são mais precisas do que aquelas anteriormente realizadas pela Toyota Caetano Portugal.

Como contribuição, este trabalho demonstra a viabilidade da aplicação de algoritmos de previsão no setor automóvel de forma a melhorar a eficácia das projeções de matrículas. Estes resultados têm uma importância prática tremenda, pois ajuda a empresa a tomar decisões mais corretas, e a realizar o plano de matrículas e os orçamentos com maior confiança.

PALAVRAS-CHAVE

Mercado Automóvel; Previsão de Matrículas; *Data Mining*, *Machine Learning*; Algoritmos de Previsão.

página propositadamente em branco

ABSTRACT

Currently, the effectiveness of forecasts plays an important role in the business world, making a significant contribution to strategic decisions. For this reason, traditional forecasts, based on simple formulas, are becoming obsolete and are being replaced by more modern and automated forecasting models. This is because complexities in data trends are often not accurately detected.

The purpose of this work is to apply forecasting algorithms based on machine learning to enhance the accuracy of Toyota Caetano Portugal's future registrations. In this academic work, in addition to adopting a quantitative approach through descriptive data analysis, a standard process model (CRISP-DM) with an adapted structure for the case study was applied, and various forecasting algorithms were explored.

Through results analysis, we observed that the neural network model achieved remarkably low errors, standing out as the one that best learned the complex patterns and seasonality from the data. Additionally, it generated close predictions to the actual values and more accurate than those previously made by Toyota Caetano Portugal's. These results highlight the importance of considering seasonality when dealing with temporal data.

As a contribution, this work demonstrates the feasibility of applying forecasting algorithms in the automotive sector to improve the registrations projections effectiveness. These acquired results hold tremendous practical importance, as they help the company to make more correct decisions, and to carry out the registration plan and budgets with greater confidence.

KEYWORDS

Car Market; Registrations Forecast; Data mining; Machine Learning; Forecasting Algorithms.

página propositadamente em branco

ÍNDICE

página propositadamente em branco

ÍNDICE DE FIGURAS

página propositadamente em branco

ÍNDICE DE TABELAS

página propositadamente em branco

LISTAS DE SIGLAS

Lista de Siglas

ISEP	Instituto Superior de Engenharia do Porto
EQM	Erro Quadrático Médio
EPAM	Erro Percentual Absoluto Médio
REQM	Raiz do Erro Quadrático Médio
EAM	Erro Absoluto Médio
RNA	Redes Neuronais Artificiais
ARIMA	Médias Móveis Autorregressivas Integradas
SARIMA	Médias Móveis Autorregressivas Integradas e Sazonais
TCAP	Toyota Caetano Portugal

página propositadamente em branco

1. INTRODUÇÃO

Este capítulo é dedicado à introdução do tema em análise. Numa fase inicial, é realizado um enquadramento do tema, com intuito mostrar a relevância do mesmo. Em seguida, são apresentados os objetivos da investigação e as opções metodológicas. O capítulo termina com uma breve apresentação da empresa em estudo e com uma breve descrição da estrutura deste documento.

1.1. Enquadramento e pertinência

Com o progresso contínuo na evolução tecnológica, as empresas deparam-se frequentemente com inúmeros desafios, nomeadamente no que concerne destaque no mercado e a importância de ir ao encontro das necessidades crescentes dos clientes Nesse contexto, e como afirmado por Cândido, Berlotti & Bedin (2017), atualmente, para atingir melhores desempenhos e resultados é necessário inovar, o que pressupõe a utilização de novas ferramentas e soluções (Cândido, Bertotti, & Bedin, 2017).

Em todos os negócios, como na importação de carros, e até mesmo a nível das concessões, é importante a utilização de técnicas estatísticas e técnicas de aprendizagem automática[1]. Através destas técnicas é possível prever, com determinada margem de erro, as vendas esperadas, o que tem um impacto direto em todas as ações de alta importância (Giering, 2008). A empresa, tendo em sua posse um modelo de previsão de vendas preciso e eficaz, poderá beneficiar com a identificação de potenciais riscos e tomar decisões estratégicas (Kohli & Godwin, 2020).

Ao longo das últimas décadas, os resultados das previsões têm demonstrado melhorias substanciais de desempenho ao utilizarem-se técnicas de aprendizagem automática (Mosavi, Ozturk, & Chau, 2018). Nesse sentido, na prática, procura-se aplicar vários algoritmos de previsão para indicadores. Através da exploração dos mesmos, pretende-se determinar quais são aqueles que desempenham melhores previsões, quando aplicados a históricos de dados das organizações. A escolha do algoritmo a utilizar irá sempre depender de fatores como o tipo de problema que se pretende resolver, o tamanho da base de dados e o número de variáveis envolvidas (Mahesh, 2020).

Os métodos estatísticos para previsão constituem uma boa alternativa às técnicas de aprendizagem automática, em particular, em problemas onde o histórico de dados é reduzido. Estas técnicas têm também a vantagem de as previsões serem obtidas através de expressões paramétricas o que facilita o processo de obtenção das mesmas. Os métodos de aprendizagem automática são mais exigentes, no que respeita ao esforço computacional necessário para obter as previsões.

Na Toyota Caetano Portugal, todos os anos são desenvolvidos orçamentos com a finalidade de representar o que irá decorrer no ano seguinte, e para tal, é necessário prever as matrículas com o maior acerto possível, para não existirem grandes desvios. Atualmente, os planos de vendas são desenvolvidos com base em previsões obtidas com fórmulas simples e com base na sensibilidade que o gestor tem sobre negócio. Contudo, já foi demonstrado que estas previsões, por serem altamente dependentes do gestor e por não considerarem históricos de dados adequados, nem sempre apresentam erros baixos.

[1] Do Ingês, *machine learning*

1.2. Questão e objetivos de investigação

A questão de investigação deste trabalho é: "Os algoritmos de previsões (estatísticos e de aprendizagem automática) são suficientemente eficazes para realizar previsões fidedignas, justificando a sua aplicação em ambiente empresarial?"

Como consequência desta questão de investigação, o objetivo central desta dissertação é aplicar técnicas estatísticas e de aprendizagem automática, com o intuito de prever a quantidade de matrículas que a Toyota Caetano Portugal terá em períodos futuros. Este objetivo deu origem aos seguintes objetivos específicos:

- Estudar diversos algoritmos de aprendizagem automática e diversos métodos estatísticos com aplicação em problemas de previsão;
- Aplicar os métodos/algoritmos adequados para prever indicadores de vendas da Toyota Caetano Portugal.
- Selecionar o melhor método/algoritmo com base em métricas de avaliação de desempenho;
- Avaliar e contrastar os resultados obtidos com o método com melhor desempenho com as previsões da Toyota Caetano Portugal.

1.3. Opções metodológicas

No presente estudo é adotada uma metodologia de investigação quantitativa, focando toda a análise em dados numéricos e padrões estatísticos. Com esta análise, pretende-se compreender os dados presentes nas bases de dados fornecidas pela Toyota Caetano Portugal.

Ao longo do trabalho, é utilizada uma estrutura de uma metodologia de processo padrão (CRISP-DM), com a finalidade de guiar tanto a parte da revisão bibliográfica, como o capítulo dos métodos utilizados e aplicação.

São explorados algoritmos de previsão, de forma a conseguir realizar previsões de matrículas com o maior acerto possível, e selecionar aquele que mais se destaca. Os algoritmos utilizados foram os seguintes: alisamento exponencial simples, alisamento exponencial sazonal, ARIMA, SARIMA e, por fim, redes neuronais.

Todas a análise, exploratória e de modelação, foi executada com recurso à linguagem de programação Python. Esta escolha deve-se ao facto de esta linguagem estar a ganhar uma enorme expressão, devido à sua grande interatividade e preocupação pelo constante crescimento das bibliotecas científicas (Pedregosa, et al., 2011). Além disso, é valorizada a sua eficácia na aplicação de algoritmos em séries temporais, à sua flexibilidade, sintaxe simples e ainda na enorme possibilidade em explorar diversas bibliotecas especializadas. É exemplo disso o caso da Ploty.graph_objects, responsável pela visualização de gráficos, ou da Pmdarima e Keras, que oferecem a elaboração de modelos preditivos, como o algoritmo ARIMA e redes neuronais.

1.4. Apresentação da empresa

Para a realização deste estudo, foram fornecidos diversos dados e informações pela Toyota Caetano Portugal, S.A., empresa do Grupo Salvador Caetano.

O Grupo Salvador Caetano foi criado em 1926, em Vila Nova de Gaia e, está atualmente presente em 41 países e 3 continentes, onde está envolvido em várias áreas de negócio. Iniciou o seu percurso com a criação de uma fábrica de carroçarias de autocarros e, em 1968, assina contrato com a Toyota Motor Corporation (marca japonesa) para a importação e distribuição da Toyota no mercado português. Com uma história em Portugal de mais de 50 anos, a empresa conseguiu colocar a Toyota na 5º posição da marca mais vendida em Portugal, no ano de 2022, e em 1.º em todo o mundo.

A Toyota Caetano Portugal está presente no mercado português com uma vasta gama de modelos de ligeiros de passageiros e ligeiros comerciais. A gama inclui soluções a combustão, híbridas, híbridas Plug-in, elétricas e ainda a hidrogénio.

1.5. Estrutura do trabalho

Este documento encontra-se divido em seis capítulos.

No Capítulo 1, encontra-se uma breve introdução ao problema, os objetivos a alcançar, a metodologia adotada e uma breve descrição da empresa.

No Capítulo 2 é feita uma revisão bibliográfica onde estão presentes os conceitos fundamentais, a exposição de alguns trabalhos relevantes na área, e ainda é aplicado uma metodologia de processo padrão com o objetivo de orientar o trabalho no seu contexto teórico. Esta metodologia (CRISP-DM) fornece uma estrutura detalhada e divida da seguinte forma: compreensão do negócio, compreensão dos dados, preparação dos dados, modelação, avaliação e por fim implantação.

No Capítulo 3 são apresentados os métodos utilizados e a sua aplicação num caso real, numa marca do setor automóvel. Neste capítulo é aplicado novamente a metodologia de processo padrão para transferir, da teoria para a prática, toda informação referida no capítulo anterior.

A apresentação dos resultados obtidos e a sua discussão são apresentados no Capítulo 4. Além disso, as previsões obtidas são também comparadas com as alcançadas pela Toyota Caetano Portugal.

Por fim, no Capítulo 5 apresentam-se as conclusões finais deste estudo, assim como as suas limitações e investigações futuras.

Além dos capítulos já mencionados, outros elementos importantes presentes na dissertação são o Resumo e o Abstract, os índices e listas que facilitam a navegação, as citações presentes nas Referências Bibliográficas, e ainda os apêndices.

2. REVISÃO BIBLIOGRÁFICA

Este capítulo apresenta uma revisão sobre os conceitos e os instrumentos fundamentais para o desenvolvimento de todo o trabalho. Esta revisão foca, essencialmente, dois aspetos: (1) um conjunto de trabalhos descritos na literatura da área do problema em estudo e (2) a metodologia CRISP-DM com descrição de diferentes métodos/algoritmos de previsão e de um conjunto de métricas de erro.

2.1. Introdução

As previsões podem-se dividir em procura a curto-médio prazo, que são úteis para planos de produção/vendas e, previsões a longo prazo, que são úteis para estratégias empresarias.

A criação de um orçamento futuro envolve várias estratégias e previsões pensadas a longo prazo, pois considera também fatores que incluem vontade corporativa, com base na evolução da tecnologia e do estilo de vida subsequente (Kato, 2020).

A indústria automóvel é um dos setores mais competitivos, no qual o rigor, a flexibilidade e a agilidade constituem os fatores críticos de sucesso. Além disso, é o setor mais rentável e importante, a nível económico, da atualidade. Todo o processo através da qual se produz ou se vende um carro é possível através de previsões corretas e precisas. Estas previsões são bastante importantes nesta área de negócio, pois fornecem informações valiosas para muitas partes interessadas, de forma a tomar decisões e investimentos ponderados (Fleurke, 2017).

No setor automóvel e como noutro setor, existem vários fatores que podem destabilizar e influenciar as previsões calculadas, isto é, pode haver ocorrências externas, indiretamente e diretamente conectadas com o setor, que prejudicam os prognósticos.

Após dois anos de pandemia Covid-19, todas as linhas de produção, transportes e fornecimento de materiais tiveram restrições e atrasos, prejudicando de certa forma todos os negócios e em grande peso o negócio automóvel (Ramani, Ghosh, & Sodhi, 2022). Não só a pandemia, mas também o baixo fornecimento de semicondutores e a guerra entre a Rússia e a Ucrânia, influenciam negativamente as previsões, que se tornaram instáveis e de pouco acerto. Os semicondutores são vitais para incontáveis aplicações por toda a parte do mundo e no momento de produção de um veículo, com inúmeros elementos tecnológicos, a sua utilização é vital. Esta indústria está a sofrer desafios em grande escala, devido à pandemia, afetando o setor automóvel (Konrad-Adenauer-Stiftung's Regional Economic Programme Asia (SOPAS), 2021). Após o fim da pandemia, tudo parecia estar a caminhar para voltar ao normal, contudo a guerra na Ucrânia levou novamente a ruturas massivas de abastecimento. Inúmeros fabricantes de automóveis foram impactados por paragens de produção, pois vários componentes de sistema de cablagem eram fornecidos por empresas ucranianas (Klee, Janson, & Leimeister, 2023).

Segundo Qu et al. (2022), uma previsão correta e precisa pode não só ajudar a ajustar os planos de marketing de forma dinâmica, mas também ajudar a economia e setores dos transportes a decisões políticas.

2.2. Alguns trabalhos

O tema abordado, relacionado com a previsão de dados para auxiliar empresas e negócios a tomar as decisões futuras, é um tema que tem vindo a crescer e a ter mais impacto ao longo dos tempos. Essa é a razão pela qual se têm vindo a ser feitos inúmeros estudos e aplicações nesta área, uma vez que os resultados têm sido amplamente comprovados.

Foi realizado um estudo por Fleurke (2017) que consistiu na recolha e na utilização de dados de vendas de automóveis da Holanda e dos EUA. Através da aplicação de vários modelos de previsão baseados na média, foram avaliados os resultados e determinados padrões. Neste estudo foram aplicados oito diferentes modelos de previsão como: alisamento exponencial, autorregressivo integrado de médias móveis (ARIMA), redes neuronais, autorregressão vetorial, modelo de alisamento exponencial, floresta aleatória[2], modelo linear generalizado e o modelo sazonal *naive*. Segundo o artigo, os modelos com maior simplicidade, alisamento exponencial e o modelo linear generalizado, superaram os modelos mais complicados como ARIMA e a autorregressão vetorial.

Zhao et al. (2022) aproveitaram os dados referentes à recente pandemia e através do número de casos de Covid-19, recolhidos entre 1 de novembro de 2021 e 17 de fevereiro de 2022, decidiram aplicar e comparar os modelos ARIMA, regressão linear múltipla e Prophet. Após execução dos três modelos, chegou-se à conclusão de que o modelo ARIMA foi o que apresentou melhor desempenho, apresentando valores de erro absoluto médio (EAM), erro médio absoluto percentual (EPAM) e a raiz do erro quadrático médio (REQM) inferiores aos dos restantes modelos. Para Zhao et al. (2022), um modelo de previsão apropriado deve ter em conta as características dos dados e o tamanho da sua amostra.

No setor do turismo, também podem ser aplicados modelos de previsão, como comprovado por Goh e Law (2002). Estes autores realizaram um estudo onde aplicaram o modelo SARIMA, uma variante do modelo ARIMA que considera sazonalidade, bem como o modelo ARIMA, para previsões relacionadas a Hong Kong. Este estudo envolveu dez modelos de séries temporais diferentes e concluiu-se que o modelo ARIMA e SARIMA apresentaram os melhores resultados. No entanto, quando não existe nenhuma intervenção obvia, o SARIMA conseguiu gerar melhores previsões (Goh & Law, 2002).

Em cenários de existência de volatilidade no setor do turismo, é fundamental realizar previsões de curto prazo, para corresponder rapidamente às variações do mercado. Reconhecendo essa necessidade, Kon e Turner (2005) decidiram aplicar redes neuronais à procura turística em Singapura. Os resultados obtidos demonstraram que a utilização de redes neuronais superou as abordagens tradicionais, levando a concluir que este modelo tem um desempenho mais eficaz na previsão de tendências e padrões no setor turístico (Kon & Turner, 2005).

Ansuj et al. (1996) aplicaram modelos de redes neuronais e ARIMA numa base de dados de vendas, de uma empresa localizada no Brasil, abrangendo o período de 1979 a dezembro de 1989. Além de analisados os resultados de cada modelo individualmente, foram também comparados e concluiu-se que as redes neuronais obtiveram previsões mais precisas em comparação com o modelo ARIMA (Ansuj, et al., 1996).

2 Random Forest na literatura em Inglês

Mesmo na área da saúde, os modelos de previsão têm uma ampla gama de aplicações. Desai et al. (2023), utilizaram as redes neuronais para prever fatores de risco de paragem cardíaca súbita numa amostra de pacientes jovens, com ascendência asiática, nos Estados Unidos. Os resultados demonstraram-se excelentes, reforçando a eficácia desses modelos na identificação de riscos. A aplicação destes modelos é benéfica e ajuda a reduzir significativamente a ocorrência de paragens cardíacas súbitas, ao identificá-los (Desai, et al., 2023).

2.3. Exploração de dados com a metodologia CRISP-DM

Atualmente, são recolhidos grandes quantidade de dados diariamente e é um desafio cada vez maior serem analisadas por humanos. Para colmatar esta situação e aumentar a rapidez da análise dos dados, surgem as técnicas de exploração de dados (Gonçalves, et al.,2020).

A exploração de dados tem com propósito retirar o máximo de informação e conhecimento de uma base de dados. Através desse processo, pretende-se que sejam detetados padrões relevantes que possam contribuir para a análise e compreensão dos mesmos (Duque, Silvas, & Godinho, 2022).

O processamento de enormes quantidades de dados, para apoiar os processos de decisão, está a tornar-se uma área cada vez mais importante nas empresas. Como Fortuny, Martens e Provost (2013) referiram, uma das mais valias da multiplicação de grandes bases de dados é a possibilidade de trabalhar com um grande número de dados. “Bigger is better”: quanto maior for a nossa base de estudo, a informação que podemos trabalhar, melhor vão ser os resultados pois a precisão irá ser significativamente melhor.

Com a inovação constante da tecnologia, a qualidade de estudo dos dados consegue ser cada vez mais detalhada e precisa. No entanto, acompanhar rigorosamente as metodologias e os processos de um projeto podem-se tornar num grande desafio para as empresas de *data science* se não tiverem meios e métodos que os ajudem a guiar na direção correta (Schröer, Kruse, & Gómez, 2021). Metodologias como CRISP-DM (Cross Industry Standard Process for Data Mining) (Plotnikova, Dumas, & Milani, 2022) desempenham um papel crucial nas organizações e na orientação de projetos, proporcionando estruturas adequadas a todas as fases de desenvolvimento.

Para obter benefícios das técnicas de exploração de dados, as organizações adotaram técnicas e processos padronizados com o objetivo de gerir e guiar projetos, mais concretamente o CRISP-DM. Esta técnica, pretende fornecer uma estrutura simples dividida em seis fases, de forma a ajudar no processo de análise dos dados.

Segundo Schröer, Kruse e Gómez (2021) a metodologia CRISP-DM divide-se em seis fases distintas seguintes, como se pode ver na Figura 1:

- Compreensão do negócio;
- Compreensão dos dados;
- Preparação dos dados;
- Modelação;
- Avaliação;
- Implantação.

Ao seguir esta estrutura, a metodologia CRISP-DM fornece um guia completo para o desenvolvimento de um trabalho. Após esta análise, seremos capazes de facilmente identificar os próximos passos e os tópicos a abordar.

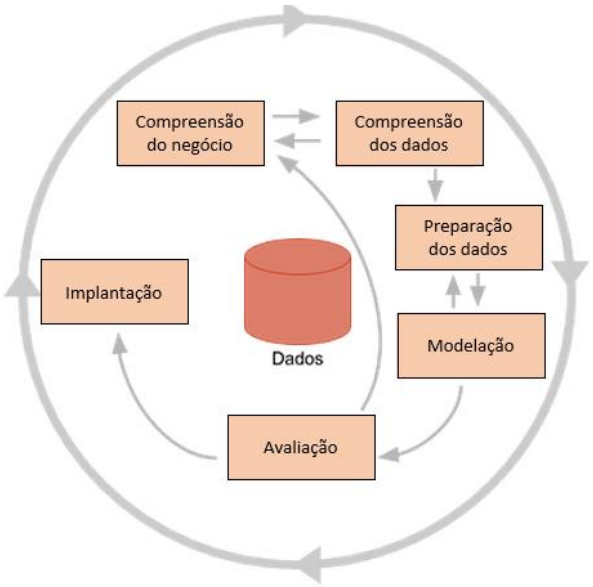


Figura 1 - Fases da metodologia CRISP-DM

No contexto do trabalho, a fase de implantação não será abordada, devido ao facto de as soluções e modelos propostos não terem sido aplicados em ambiente empresarial. Portando, o foco estará nas fases anteriores, como a análise do negócio e dos dados, a preparação dos dados e modelação dos algoritmos e, por fim, a avaliação dos resultados.

2.3.1. Compreensão do negócio (CRISP-DM)

Numa primeira fase, é fundamental ter um objetivo bem estabelecido para conseguir desenvolver um fio condutor em todo o trabalho e, assim, chegar ao resultado esperado. Trata-se de clarificar o funcionamento do negócio, os objetivos e as restrições que influenciam, direta ou indiretamente, no estudo.

O objetivo desta fase do CRISP-DM passa então por evidenciar todos os fatores importantes que podem influenciar no resultado do trabalho. Isto envolve a descoberta de vários factos com maior detalhe relacionados com o negócio, isto é, recursos, restrições, suposições de acontecimentos anteriores, ou que poderão acontecer no futuro, e impactar os resultados.

Nesta fase é também necessário clarificar os objetivos e critérios de sucesso do trabalho, no sentido de desenvolver modelos capazes de responder ao pretendido e, assim, chegar à solução ideal.

Em suma, conforme Wirth e Hipp (1968), para uma boa compreensão do negócio é necessário passar por quatro fases:

1. Determinar os objetivos do negócio – pretende-se selecionar os objetivos que se pretende atingir e os critérios de sucesso;

2. Avaliar a situação - avaliar a disponibilidade de recursos e requisitos do projeto, assim como os riscos associados ao mesmo;
3. Determinar as metas de DM – é necessário definir qual é o principal objetivo com a utilização da data mining;
4. Desenvolver um plano para o projeto – passa por selecionar as tecnologias a utilizar em cada etapa do projeto e as respetivas abordagens de pesquisa.

2.3.2. Compreensão dos dados (CRISP-DM)

A compreensão dos dados envolve uma análise detalhada e rigorosa das bases de dados disponibilizadas para estudo.

Esta fase é das mais críticas pois é o ponto de partida para a qualidade dos resultados e irá evitar problemas inesperados em fases posteriores, proporcionando assim um bom funcionamento dos modelos de previsão. Permite assim o envolvimento com as bases de dados e explorá-las através de gráficos e tabelas, podendo, assim, determinar a qualidade dos dados e retirar conclusões dessas etapas.

Segundo Wirth e Hipp (1968), esta fase também se divide em quatro etapas:

1. Recolha da ou das bases de dados – adquirir os dados necessários e importá-las para a ferramenta de análise;
2. Descrição dos dados – passa por descrever detalhadamente cada uma das variáveis presentes na base de dados, tal como o seu formato;
3. Explorar os dados – consultar e criar relações de dados com ajuda a gráficos ou tabelas visuais;
4. Verificar a qualidade dos dados – passa por analisar os erros presentes nas bases de dados recolhidas.

2.3.3. Preparação dos dados (CRISP-DM)

A fase de preparação dos dados é, talvez, a mais importante de toda a metodologia CRISP-DM. Isto deve-se ao facto de maior parte do esforço e tempo, de praticamente todos os projetos e trabalhos, ser dedicada a essa etapa. Tarefas como a seleção dos dados, limpeza, construção, integração e formatação são fundamentais para o bom desenvolvimento de todas as próximas etapas. Assim, é fundamental investir bastante tempo e atenção nesta fase.

Uma regra comum é a de que, entre 50% a 80% do esforço do projeto, irá cair sobre a fase de preparação de dados. (Saltz, 2021).

Segundo Wirth e Hipp (1968), esta fase está dividida em cinco tarefas:

1. Selecionar os dados - Determinar o tamanho da base de dados necessária para o trabalho;
2. Realizar a limpeza dos dados - Eliminar todos os dados que não nos interessam, dados que são irrelevantes para o estudo e ainda dados errados. Além disso, é importante eliminar dados em branco;

3. Contruir os dados - Criar atributos através de outros;
4. Integrar os dados - Criar conjuntos de dados através de várias fontes de dados;
5. Formatar os dados – Formatar os dados com formatação errada.

2.3.4. Modelação (CRISP-DM)

Os modelos de previsão consistem no processamento de um conjunto de dados e condições atuais, visando prever o valor futuro da variável alvo. Estes modelos procuram estimar o valor mais próximo da realidade para um espaço temporal inicialmente definido.

O processo empírico da previsão depende de agentes que afetam diretamente na previsão de todo o estudo, isto é, é necessário recolher e escolher cuidadosamente todos os fatores dependentes para adquirir uma correta previsão do sistema. (Mamun, et al., 2020)

Segundo Gail et al. (2009), métodos ou modelos de previsão estão presentes também em variadas áreas de negócio, incluindo na medicina, física, meteorologia e finanças. Na era atual em que vivemos, está a ser particularmente importante no campo da medicina, isto devido ao impacto que tem ao prever doenças em sua fase precoce e rapidamente atacar com tratamento específico.

Paralelamente, aprendizagem automática é o estudo científico dos algoritmos e modelos estatísticos utilizados pelos computadores para realizar uma respetiva tarefa sem ser explicitamente programada (International Journal of Science and Research, 2020).

Na atualidade, grande parte das plataformas sociais e de pesquisa, utilizam algoritmos de forma a estudar os comportamentos e necessidades das pessoas e, assim, fornecer um serviço ou um bem que se enquadre no estilo de vida e no gosto pessoal de cada um.

Uma das grandes vantagens de utilizar um algoritmo é a sua facilidade de aprender com os dados e agir automaticamente.

Segundo Wirth e Hipp (1968), esta fase está dividida em quatro tarefas:

1. Selecionar o modelo a utilizar – existem vários modelos de previsão, desta forma é necessário identificar aqueles podem ser aplicados no trabalho em causa;
2. Divisão dos dados – A base de dados tem de ser dividida entre dados de treino, dados de teste e, por vezes, até mesmo dados de validação.
3. Contruir o modelo – através dos dados de treino é desenvolvido o modelo. Esta tarefa envolve o ajuste de parâmetros para captar melhor relações entre variáveis.
4. Avaliar o modelo – Pretende-se avaliar os resultados em comparação com os dados reais.

A previsão de séries temporais é uma técnica frequentemente utilizada no campo da aprendizagem automática. Ela desempenha um papel importante na análise de dados sequencias ao longo do tempo, encontrando diversas áreas possíveis de aplicar, tais como finanças, saúde e economia. Além disso, também permite a análise de dados de vendas ao longo do tempo, identificando padrões e tendências nos dados históricos de vendas, permitindo prever valores futuros.

Para realizar previsões com alto valor de eficácia, irão ser utilizados os seguintes algoritmos:

1. Alisamento exponencial

O algoritmo de alisamento exponencial pretende construir previsões de valores futuros, através de médias ponderadas das observações presentes em dados históricos. Isto é possível atribuindo pesos com valores superiores aos dados mais recentes e pesos inferiores nos dados mais distantes (Fomby, 2008).

Os modelos de alisamento exponencial, como o alisamento exponencial simples e alisamento exponencial sazonal, desempenham um papel crucial no estudo de previsões de séries temporais. No entanto, caso seja pretendido estudar as tendências e sazonalidades dos dados, o método de alisamento exponencial sazonal é mais apropriado.

1.1. Alisamento exponencial simples

Segundo Fomby (2008), o algoritmo de alisamento exponencial é representado pela seguinte equação:

$$L_t = \alpha y_t + (1-\alpha) L_{t-1}$$

Onde:

L_t é o valor previsto para o período t.

α é a constante de alisamento, com valor entre 0 e 1.

y_t é o valor observado na série temporal no período t.

L_{t-1} é o valor previsto para o período t-1.

Para calcular o valor observado y_t na série temporal, no período t, podemos recorrer à seguinte equação:

$$y_t = u_t + a_t$$

Onde:

y_t é o valor observado na série temporal no período t.

u_t é o valor médio da série temporal no período t.

a_t é o erro ou componente aleatória na série temporal.

Através da primeira equação podemos realizar previsões de curto prazo em séries temporais. Esta permite calcular o valor previsto para o período t através da multiplicação de um fator α com o valor observado y_t e o valor previsto anterior L_{t-1} com o fator (1- α).

O valor de y_t é calculado com base na segunda equação, onde multiplica o valor médio da série temporal com o erro presente. No contexto do algoritmo, nem sempre esta segunda equação é aplicada, pois existe maior preocupação em trabalhar os dados históricos e realizar previsões futuras.

1.2. Alisamento exponencial sazonal

O algoritmo de alisamento exponencial sazonal, e conforme dito por Fomby (2008), é uma extensão do algoritmo de alisamento exponencial simples que leva em consideração a sazonalidade dos dados de séries temporais.

Para aplicar este algoritmo temos de recorrer a três equações:

$$y_t = u_t + S_{t,p} + a_t$$

$$L_t = \alpha(y_t - S_{t-P}) + (1 - \alpha)\, L_{t-1}$$

$$S_t = \delta(y_t - L_t) + (1 - \delta)\, S_{t-P}$$

Onde:

y_t é o valor observado na série temporal no período t.

u_t é o valor médio da série temporal.

$S_{t,\mathrm{p}}$ representa a componente sazonal no momento t com um parâmetro p que define a sazonalidade.

a_t é o erro ou componente aleatória na série temporal.

L_t é o valor previsto para o período t.

α é a constante de alisamento, com valor entre 0 e 1.

S_t é o valor estimado da componente sazonal no momento t.

δ é o coeficiente de alisamento para a componente sazonal.

$S_{t-\mathrm{P}}$ é a componente sazonal que representa o desvio do período t em relação à média sazonal no mesmo período t – P.

P é o período da sazonalidade.

Estas três equações anteriormente mostradas, são utilizadas num processo normal de alisamento exponencial sazonal. Assim, a primeira equação modela os dados observados, como a soma dos componentes exógeno, sazonal e erro. Já a segunda equação é responsável por estimar o nível de suavizado subjacente, enquanto a terceira estima a componente sazonal.

2. ARIMA (Médias Móveis Autorregressivas Integradas)

Este algoritmo tem como objetivo a previsão de séries temporais, combinando elementos de modelos autorregressivos e modelos de médias móveis.

O modelo ARIMA é influenciado pelos valores históricos e a tendência dos dados. Assim, ajusta os coeficientes autorregressivos e das médias móveis para desenvolver as dependências das séries temporais. Com base nestes componentes, este modelo realiza previsões do futuro, considerando tanto a autocorrelação quanto a influência dos erros passados.

Esta fase está dividida em três tarefas:

3. Auto-regressão (AR): modela a dependência dos valores futuros da série temporal com base em seus valores passados.
4. Integração (I): modela a tendência da série temporal, tratando a diferença entre os valores observados e a média de longo tempo.
5. Média móvel (MA): modela a sazonalidade da série temporal, tratando a dependência dos erros de previsão com base em suas médias móveis.

Segundo Fattah et al. (2018), estas três tarefas, anteriormente referidas, identificam-se como os seguintes parâmetros (p, d, q).

Onde:

3. p é o número de termos autorregressivos;
4. d representa a ordem de diferenciação da série;
5. q refere-se ao número de médias móveis.

O algoritmo ARIMA oferece a flexibilidade de ajustar o modelo para adaptar-se à natureza intrínseca dos dados. Através dos parâmetros acima conseguimos ajustar as variáveis e encontrar previsões mais precisas dos valores futuros da série (Ho & XIE, 1998).

A equação geral de um modelo ARIMA combina os três componentes da seguinte maneira:

$$A_t = \psi_0 + \phi_1 A_{t-1} + \phi_2 A_{t-2} + \ldots + \phi_p A_{t-p} - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \ldots - \theta_q \varepsilon_{t-q} + \varepsilon_t$$

Onde:

A_t é o valor da série temporal que foi transformado para tornar a série estacionária.

ψ_0 representa a componente de tendência.

$\phi_1, \phi_2, \ldots, \phi_p$ dizem respeito aos coeficientes autorregressivos.

$\theta_1, \theta_2, \ldots, \theta_p$ são os coeficientes das médias móveis.

$\varepsilon_{t-1}, \varepsilon_{t-2}, \ldots, \varepsilon_{t-q}$ são os erros aleatórios nos períodos anteriores.

ε_t é o erro aleatório no tempo t.

Esta equação combina estes elementos, acima mencionados, para prever o valor futuro da série temporal transformada A_t, tendo em consideração componentes como a tendência, dados históricos, erros passados e o termo de erro atual. Isto permite ao modelo ARIMA capturar dependências temporais, variações não explicadas e as tendências para conseguir realizar previsões com maios eficácia.

3. SARIMA (Médias Móveis Autorregressivas Integradas e Sazonais)

O modelo SARIMA, acaba por ser uma extensão do modelo ARIMA que incorpora componentes sazonais. Além de incluir na sua equação matemática termos como autorregressão, integração e diferenciação, este adiciona termos sazonais para lidar com padrões sazonais presentes nos dados.

Este modelo adiciona parâmetros sazonais ao modelo ARIMA, resultando assim numa notação de parâmetros completa como (p,d,q)(P,D,Q)s (Dindarloo, et al., 2016).

Onde:

P é a ordem do termo autorregressivo sazonal;
d representa a ordem de diferenciação sazonal;
q refere-se ao número de médias móveis sazonais;
s representa o período sazonal.

Conforme Andueza et al. (2023), a equação do modelo SARIMA pode ser representada da seguinte forma:

$$\phi_p\ (\mathrm{B}^{\mathrm{m}})\phi_p\ (\mathrm{B})(1-\mathrm{B}^{\mathrm{m}})^{\mathrm{D}}(1-B)^d\ y_t\ =\ \Theta_Q(\mathrm{B}^{\mathrm{m}})\theta_q(\mathrm{B})w_t$$

Onde:

y_t diz respeito à série temporal não estacionária.

w_t é o processo de ruído branco gaussiano.

$\phi_p\ (\mathrm{B})$ refere-se ao polinômio autorregressivo não sazonal

$\theta_q(\mathrm{B})$ refere-se ao polinômio de média móvel não sazonal.

D diz respeito à diferenciação sazonal.

$\phi_p\ (\mathrm{B}^{\mathrm{m}})$ é o polinômio autorregressivo sazonal.

$\Theta_Q(\mathrm{B}^{\mathrm{m}})$ é o polinômio de média móvel sazonal.

As equações para o modelo autorregressivo sazonal e modelo das médias móveis são respetivamente:

$$\phi_p\ (\mathrm{B}^{\mathrm{m}}) = 1 - \phi_1\ (\mathrm{B}^{\mathrm{m}}) - \phi_2\ (\mathrm{B}^{2\mathrm{m}}) + \ldots + \phi_p\ \left(\mathrm{B}^{\mathrm{Pm}}\right)$$

$$\Theta_Q(\mathrm{B}^{\mathrm{m}}) = 1 + \Theta_1(\mathrm{B}^{\mathrm{m}}) + \Theta_2(\mathrm{B}^{2\mathrm{m}}) + \ldots + \Theta_Q\left(\mathrm{B}^{\mathrm{Qs}}\right)$$

O modelo SARIMA é uma ferramenta extremamente importante no estudo de séries temporais sazonais, pois permite identificar seja tendências como padrões sazonais e, assim, alcançar previsões com maior precisão ao longo do tempo.

4. Redes Neuronais Artificiais

As redes neuronais artificiais (RNA) são sistemas compostos por unidades de processamento chamadas de neurónios, que estão totalmente conectadas entre si através de pesos diferenciados. Assim, os neurónios estão agrupados em camadas, onde cada um deles recebe um conjunto de sinais de entrada, que são essencialmente as saídas dos neurónios anteriores. Cada neurônio realiza uma soma ponderada não linear das entradas e envia o resultado para os neurónios da próxima camada. (Loureiro, Miguéis, & Silva, 2018).

As camadas são assim divididas em três tipos:

1. Camadas de entrada: Recebe os dados de entrada e direciona para a camada seguinte.
2. Camadas intermédias (ocultas): Fazem parte neurónios que não estão em contacto direto com as entradas ou saídas. Esta camada é bastante importante pois realiza transformações complexas à medida que são propagados pela rede.
3. Camadas de saída: Produz a saída final da rede neuronal com base nas transformações realizadas anteriormente.

As RNA possuem variados tipos de arquiteturas e propósitos diferentes, adaptado a cada trabalho e objetivo pretendido.

Neste trabalho iremos abordar as redes neuronais recorrentes (RNN). Neste contexto, as LSTM (memória de longo e curto prazo) são uma variação das RNN que foram especificamente desenvolvidas para trabalhar com dependências temporais em dados sequenciais, como séries temporais. A principal diferença entre um RNN e a arquitetura LSTM, é que este último possui uma célula de memória mais complexa (Tax, et al.,2017)

Segundo Tax et al. (2017), o modelo LSTM pode ser aplicado através das seguintes fórmulas:

$$f_t = \text{sigmoid}(W_f[h_{t-1}, x_t] + b_f)$$

Primeiramente é aplicada a fórmula de esquecimento, que decide a quantidade da memória anterior que deve ser descartada. Neste caso, o peso W_f vai multiplicar as entradas, h_{t-1} e x_t , somar com um viés e aplicar a sigmoide em cima disso.

$$i_t = \text{sigmoid}(W_i[h_{t-1}, x_t] + b_i)$$

$$\hat{\text{C}}_t = \tanh(W_c[h_{t-1}, x_t] + b_c)$$

A equação da porta de entrada decide a quantidade de nova memória que deve ser adicionado à memória da célula. Esta etapa envolve duas equações, a equação i_t para realmente calcular a nova informação e a porta de entrada propriamente dita $\hat{\text{C}}_t$.

$$\text{C}_t = f_t * C_{t-1} + \text{i}_i * \hat{\text{C}}_t$$

De seguida, a memória da célula é atualizada com recurso à equação C_t, combinando a memória anterior com a nova informação, tendo em conta as equações anteriormente referidas: porta de esquecimento e porta de entrada.

$$o_t = \text{sigmoid}(W_o[h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(\text{C}_t)$$

Por fim, temos a equação da porta de saída o_t e a representação da equação que define a quantidade de memória da célula que irá ser utilizada para calcular a saída final h_t.

Nessas equações:

f_t é a saída da porta de esquecimento.

i_t é a saída da porta de entrada.

$\hat{\text{C}}_t$ é a nova informação que pode ser adicionada à memória da célula.

C_t é a memória atualizada.

o_t é a saída da porta de saída.

h_t é a saída da célula LSTM para o momento t.

sigmoid é a função de ativação.

$\tanh$ é a função tangente hiperbólica.

W_f, W_i, W_c e W_o são as matrizes dos pesos.

b_f, b_i, b_c e b_o são os vetores do viés.

Observando a pesquisa feita por, Tsai (2016), as redes neuronais precisam de ser divididas em três conjuntos de dados para a sua aplicação e, assim, conseguir explorar e adquirir resultados com

maior previsão. A maior parte da base de dados em estudo deverá concentrar-se no conjunto de dados de treino e dividir a restante para validação e teste.

Os dados de treino desempenham um papel fundamental no modelo, pois permitem aprender tendências e padrões nos dados. Posteriormente, os dados de teste ajudam a estimar o desempenho do modelo, fornecendo uma estimativa da eficácia em realizar previsões nos dados não vistos. Entre o momento de treino e teste, temos a presença da etapa de validação, onde visa verificar a eficácia quanto à capacidade de gerar dados durante o treino.

Para Julia et al. (1996), as redes neuronais artificiais têm inúmeras vantagens em relação aos algoritmos de simulação tradicional, tais como, a capacidade de aprender com a experiência, processar rapidamente, processar informações em paralelo, podem ajustar-se ao ambiente em mudança e são até mesmo resistentes a dados ausentes, o que justifica o seu potencial de utilização.

Por outro lado, existem também algumas limitações, como não ser totalmente fácil obter resultados satisfatórios, encontrar uma arquitetura eficaz e interpretar os pesos atribuídos aos diferentes componentes do sistema (Julia, et al.,1996).

2.3.5. Avaliação (CRISP-DM)

Na fase da avaliação, os resultados das previsões adquiridas por meio dos modelos anteriormente referidos, são examinados através da comparação com os dados reais. Portanto, além dos resultados serem interpretados, eles serão também avaliados quanto à sua eficácia com auxílio de várias métricas de desempenho.

As métricas de desempenho são vitais para realizar a avaliação em várias áreas. Neste contexto, em trabalhos de machine learning as métricas são usadas para estabelecer uma comparação entre as previsões adquiridas pelo conjunto de dados de treino e os dados reais presentes no conjunto de dados de teste (Botchkarev, 2018).

Assim, segundo Wirth e Hipp (1968), esta etapa está dividida em três fases:

1. Avaliar os resultados – passa por analisar os resultados adquiridos segundo critérios de sucesso estabelecidos numa fase inicial da compreensão do negócio.
2. Realizar uma revisão – é elaborada uma revisão mais completa a todos os dados adquiridos, todos os resultados obtidos e determinar se houve algum fator ou tarefa que foi de alguma forma esquecida. É importante perceber se falhou algum passo que comprometa a credibilidade e qualidade dos resultados obtidos.
3. Determinar os próximos passos – nesta parte é importante evidenciar possíveis ações futuras e determinar se deve prosseguir com a implantação.

Existem variadas técnicas de desempenho que podem fornecer diferentes leituras de erros. Neste trabalho serão aplicadas quatro métricas (erro quadrático médio, erro absoluto médio, raiz do erro quadrático médio e erro percentual absoluto médio) para avaliar a eficácia dos resultados e conseguir, através dos erros, comparar e determinar o modelo que teve melhor desempenho.

Assim, as métricas utilizadas foram as seguintes:

- Erro Quadrático Médio (EQM)

Uma das métricas utilizadas para validar a qualidade dos dados e determinar o erro associado, é o erro quadrático médio (EQM), onde permite avaliar a média dos quadrados entre a diferença do valor real observado e o valor previsto pelo modelo. Segundo Botchkarev (2018), a equação escreve-se da seguinte forma:

$$EQM = \frac{1}{n}\sum_{j=1}^{n}\left(A_j - P_j\right)^2$$

- Erro absoluto médio (EAM)

O erro médio absoluto é outra métrica de avaliação que calcula a média das diferenças absolutas entre os valores observados e os previstos. Esta métrica é expressa da seguinte forma:

$$EAM = \frac{1}{n}\sum_{j=1}^{n}|A_j - P_j|$$

- Raiz do erro quadrático médio (REQM)

A raiz do erro quadrático médio calcula a raiz quadrática da média dos quadrados entre a diferença do valor real observado e o valor previsto. Por outras palavras, a REQM pode ser calculada através da raiz quadrática dos valores do erro quadrático médio. Esta métrica é expressa da seguinte forma:

$$REQM = \sqrt{\frac{1}{n}\sum_{j=1}^{n}\left(A_j - P_j\right)^2}$$

ou

$$REQM = \sqrt{EQM}$$

- Erro percentual absoluto médio (EPAM)

Esta métrica de avaliação de desempenho fornece uma visão percentual dos erros alcançados, permitindo, assim, uma rápida e fácil análise. Esta, calcula a média das diferenças percentuais absolutas entre a diferença do valor real observado e o valor previsto. Esta métrica é expressa da seguinte forma:

$$EPAM = \frac{100}{n}\sum_{j=1}^{n}\frac{|A_j - P_j|}{|A_j|}$$

Onde:

n = é o número de amostras.

A_j = valores observados.

P_j = valores previstos pelo modelo.

Estas métricas de desempenho são utilizadas para avaliar a qualidade das previsões alcançadas por modelos de previsão em relação aos valores reais observados. Cada uma destas métricas acima mencionadas oferecem uma perspetiva diferente sobre o desempenho do modelo, permitindo entender quão longe está dos valores reais.

3. MÉTODOS E APLICAÇÃO

Tendo como finalidade a extração total da informação dos dados para o trabalho, foi utilizada uma metodologia de processo padrão (CRISP-DM). Esta metodologia tem como objetivo a orientação do trabalho em análise desde o seu período inicial até à implementação.

Inicialmente, compreendeu-se o negócio, qual o funcionamento do mesmo, os seus requisitos e tentar definir, de forma mais clara, os padrões e detalhes fundamentais aos passos seguintes do processo.

De seguida, foi analisada toda a base de dados de forma a entender a qualidades, o formato, tamanho e as características dos dados apresentados.

Todos estes dados foram tratados, transformados e estruturados para que estejam de acordo com os requisitos necessários para a análise. É importante ter uma base de dados com qualidade para que os algoritmos sejam também eles aplicados com eficácia.

Na fase de modelação, foram utilizados vários algoritmos para criar modelos preditivos, de forma a realizar previsões sobre valores futuros da série temporal.

De seguida, os modelos aplicados foram avaliados no que concerne o seu desempenho e eficácia. Foram aplicadas várias métricas para avaliar a exatidão do modelo preditivo.

Por fim, os resultados foram analisados e comparados com a realidade do negócio para tentar compreender se foi alcançado o objetivo, e posteriormente, implementar no negócio.

3.1. Compreensão do negócio (CRISP-DM)

O objetivo principal desta dissertação é, assim, utilizar técnicas de aprendizagem automática e técnicas estatísticas para prever a quantidade de matrículas futuras da Toyota Caetano Portugal (TCAP). A TCAP é o representante oficial da Toyota Motor Corporation no mercado português. Dedica-se, numa das suas atividades, à importação e venda de uma vasta gama de veículos de alta qualidade.

Esta análise permitirá, portanto, auxiliar a equipa de planeamento a tomar melhores decisões, no momento de realizar previsões para o orçamento seguinte.

Os objetivos específicos deste trabalho são, desta forma:

- Explorar diferentes algoritmos de aprendizagem automática e técnicas estatísticas;
- Comparar os melhores resultados atingidos com os dados previstos pela TCAP.

O tema em estudo e o objetivo a que se propõe é atual e consiste numa necessidade recorrente, não só no mercado automóvel, mas como noutros mercados onde a previsão de séries temporais é necessária. Utilizar estes mecanismos de previsão permite obter valores bastante próximos da realidade, assim como a redução do tempo de trabalho.

Ao serem aplicadas, as técnicas de aprendizagem automática e estatísticas, vão permitir não só analisar padrões históricos, como realizar a projeção de tendências e sazonalidades futuras. A linha temporal de previsões está focada num horizonte de curto a médio prazo, de forma a corresponder às necessidades.

Os critérios de sucesso para esta dissertação incluem:

- Obter modelos de previsão eficazes com baixa percentagem de erro;
- Obter melhores previsões do que a Toyota Caetano Portugal.

Para realizar o estudo em causa foi fornecida, pela TCAP, a base de dados de matrículas, base de dados de contratos e a previsão de matrículas estabelecido no orçamento de 2022 e 2023.

Foram também utilizadas ferramentas de programação para desenvolver os modelos de previsão, como foi o caso do Python.

É importante realçar que, apesar dos algoritmos serem baseados em dados históricos e fatores conhecidos, existem sempre imprevistos externos e desconhecidos que resultam em mudanças significativas. Estas mudanças, que não podem ser previstas e influenciam exponencialmente no erro das previsões:

- Guerra;
- Pandemias;
- Falta de fornecimento;
- Crise.

Neste trabalho adotou-se uma abordagem de pesquisa quantitativa para analisar padrões e tendências relacionadas com as matrículas registadas e os contratos efetuados. Com o objetivo de quantificar esses aspetos, foram recolhidos dados mensuráveis desde o ano de 2015 até junho de 2023.

Por meio de análise estatística, pretendeu-se identificar relações e comportamentos presentes nas matrículas e nos contratos ao longo desse espaço temporal. Sobre estes dados foram executados algoritmos de previsão, permitindo assim a obtenção de previsões de valores para o futuro. Ao utilizar esta metodologia, pretende-se que este trabalho possa fornecer informações importantes para a empresa e demais interessados, contribuindo em grande escala para a tomada de decisão correta, baseada em estatísticas e previsões próximas da realidade.

3.2. Compreensão dos dados (CRISP-DM)

Identificado o objetivo na fase anterior (Compreensão do negócio), torna-se necessário compreender as bases de dados utilizadas no decorrer de toda a análise e as técnicas a aplicar.

Foram exploradas duas bases de dados:

- Base de dados de matrículas;
- Base de dados de contratos;
- Base de dados de matrículas previstas pela Toyota.

Os dados em estudo foram disponibilizados pela TCAP, sendo, portanto, dados reais da empresa. No que concerne o conteúdo das bases de dados, encontraram-se variáveis nominais, variáveis de data e ainda variáveis numéricas. Na tabela que se segue consegue obter-se toda a informação referente à matrícula do veículo após a venda da mesma, nomeadamente, dados característicos do veículo e especificações, dados da concessão, códigos do vendedor e ainda valores de venda.

Tabela 1 - Descrição das variáveis da base de dados matrículas.

Variável	Tipo	Descrição
COD_CONCESSAO	Inteiro	Código da concessão
NOME_CONCESSAO	Texto	Nome da concessão
MARCA	Texto	Marca do veículo
MODELO	Texto	Código do modelo
VERSAO	Texto	Código da versão
CHASSI	Texto	Código do chassi
VIN	Texto	Nº do chassi
MOTIVO_DESPACHO_ESPECIAL	Texto	Motivo de despacho especial
MATRICULA	Texto	Matrícula do veículo
DT_MATRICULA	Data	Data da matrícula
DT_VENDA_CONC	Data	Data de venda à concessão
DT_VENDA_PUBLICO	Data	Data de venda ao cliente
GAMA	Texto	Modelo do veículo
COD_COR	Inteiro	Código da cor do veículo
COD_VENDEDOR	Inteiro	Código do vendedor
ZONA_CONCELHO_VENDA	Inteiro	Código do concelho da venda
PUBLICO	Float	Preço de venda ao público
NUM_HOMOLOGACAO	Inteiro	Nº da Homologação
NUM_EXTENSAO	Inteiro	Nº da extensão da homologação
CILINDRADA	Inteiro	Cilindrada do veículo
CO2_G_KM	Inteiro	Consumo de CO2
PARTICULAS_MG_KM	Inteiro	Partículas emitidas
TIPO_ENCOMENDA	Texto	Tipo de venda
COD_VENDEDOR_ENC	Inteiro	Código do vendedor que encomendou
COD_STAND	Inteiro	Código do stand
DESIGNACAO_STAND	Texto	Designação do stand
TIPO_VENDEDOR_ENC	Texto	Função do vendedor que encomendou
DATA_FACTURA_CONC	Data	Data de fatura à concessão
VAL_BASE_DEBITADO	Float	Preço base do para a concessão
VAL_ISV	Float	Valor de ISV
VAL_IUC_DEBITADO	Float	Valor de IUC
VAL_DESCONTO	Float	Valor do desconto
VAL_INCENTIVO	Float	Valor do incentivo
VAL_BASE_CONC	Float	Preço base do para a concessão
VAL_BASE_PUBL	Float	Preço base para o público
VAL_IUC_TABELADO	Float	Valor de IUC

Pode-se também obter informações dos contratos realizados através da tabela abaixo, que permite retirar informação sobre a data da realização do contrato, da viatura pretendida e suas designações.

Tabela 2 - Descrição das variáveis da base de dados contratos.

Variável	Tipo	Descrição
Marca	Texto	Marca do veículo
Híbrido	Texto	Se o veículo tem motorização híbrida
Gama	Texto	Modelo do veículo
Versão	Texto	Versão do veículo
Código	Texto	Código da versão do veículo
CC	Inteiro	Nº de contratos
Ano	Inteiro	Ano do contrato
Mês	Texto	Mês do contrato
Dia	Inteiro	Dia do contrato

Por último, e com o objetivo de comparar com previsões realizadas pela marca, recorreu-se a uma base de dados de orçamento de vendas, onde foi retirada a informação necessária para o estudo (Tabela 3).

Tabela 3 - Descrição das variáveis da base de dados matrículas previstas

Variável	Tipo	Descrição
Data	Data	Data do mês previsto
Previsão Matrículas	Inteiro	Nº de matrículas previstas

Analisando o conteúdo das bases de dados, e tendo em conta os campos fundamentais ao trabalho, foi possível perceber que a consistência dos dados é boa e não apresenta erros que prejudiquem a eficácia dos algoritmos.

Verificou-se que não existiam registos em duplicado, campos em branco e preenchimento incorreto nos dados essenciais para o estudo. Os valores foram também validados através de relatórios de vendas disponibilizados.

Os dados de cada base de dados estão armazenados em forma de tabela, o que facilita a sua utilização e análise. Uma estrutura bem definida do conjunto de dados permitiu uma rápida e fácil importação e a aplicação de várias técnicas de análise.

Foi também realizada uma análise aos dados fornecidos, de forma a compreender a base de dados, as suas tendências e padrões. Através de métricas estatísticas foi possível saber como é que os dados estão distribuídos.

Na tabela abaixo é possível perceber que a base de dados de matrículas contém dados de 2015 até junho de 2023. No total dos oito anos completos, 2018 foi o ano em que a média de viaturas matriculadas foi superior, atingindo os 993 veículos. Por outro lado, no ano de 2020, foi onde se verificou a média mais baixa, o desvio padrão foi o segundo melhor, o que significa que as

matrículas foram mais constantes ao longo do espaço temporal, próximos da média, sendo possível de comprovar pelos seus valores mínimos e máximos.

Tabela 4 - Estatísticas da base de dados matrículas.

	Contagem	Média	Desvio Padrão	Mínimo	25%	50%	75%	Máximo
2015	12	798	273	523	612	718	920	1474
2016	12	822	312	493	618	706	959	1408
2017	12	866	309	479	657	792	1002	1448
2018	12	993	224	604	873	912	1232	1332
2019	12	930	201	704	811	870	971	1350
2020	12	662	214	150	616	728	782	911
2021	12	864	284	481	620	838	954	1387
2022	12	940	231	608	782	938	1116	1318
2023	6	1144	236	831	1005	1107	1330	1442

Com o auxílio da Figura 2, além de confirmar a informação anteriormente descrita, percebe-se que desde 2020 que as matrículas seguem uma tendência crescente. Retira-se também que existem alguns *outliers* no ano de 2015, 2019 e 2020 que podem prejudicar no bom desempenho dos algoritmos.

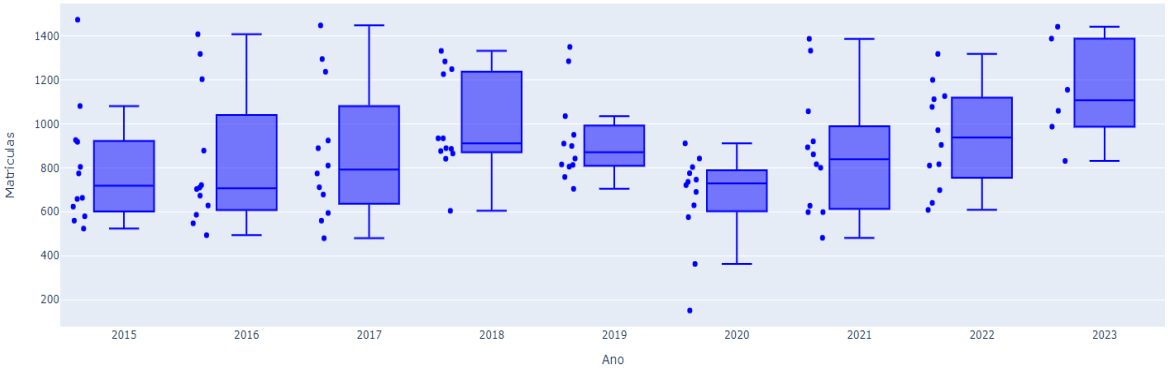


Figura 2 - Boxplot do número de matrículas por ano.

Além a análise descritiva anterior, também se analisou, do ponto de vista descritivo, a base de dados de contratos, abrangendo o mesmo período de 2015 a junho de 2023. Os resultados desta análise estão apresentados na Tabela 5. Da análise desta tabela, foi possível perceber que o comportamento desta série temporal parece ser semelhante ao comportamento da série das matrículas, o que significa que o momento da matrícula do veículo está de certa forma alinhado com o momento da realização do contrato.

Por outro lado, na Figura 3, consegue-se visualizar que o desvio padrão atingiu um valor máximo em 2021, o que quer dizer que nesse ano a distribuição dos contratos por mês foi realizada de forma menos uniforme, o que é comprovado pelo seu limite inferior e superior.

Tabela 5 - Estatísticas da base de dados contratos.

	Contagem	Média	Desvio Padrão	Mínimo	25%	50%	75%	Máximo
2015	12	796	212	411	670	811	982	1072
2016	12	840	239	508	684	816	988	1213
2017	12	873	268	510	725	846	966	1575
2018	12	985	247	738	850	903	1018	1562
2019	12	937	256	672	780	884	972	1565
2020	12	709	258	127	632	763	916	971
2021	12	1006	402	451	776	845	1241	1955
2022	12	1092	224	764	936	1088	1214	1460
2023	6	1112	239	838	1016	1084	1108	1554

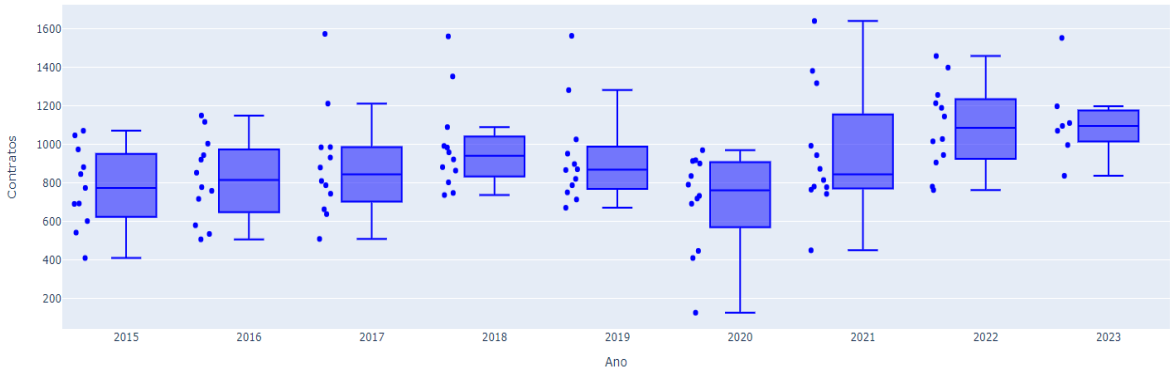


Figura 3 - Boxplot do número de contratos por ano.

Com o o objetivo de perceber a evolução temporal do número de matrículas e do número de contratos ao longo dos anos do histórico, foram obtidos gráficos de séries temporais (Figura 4).

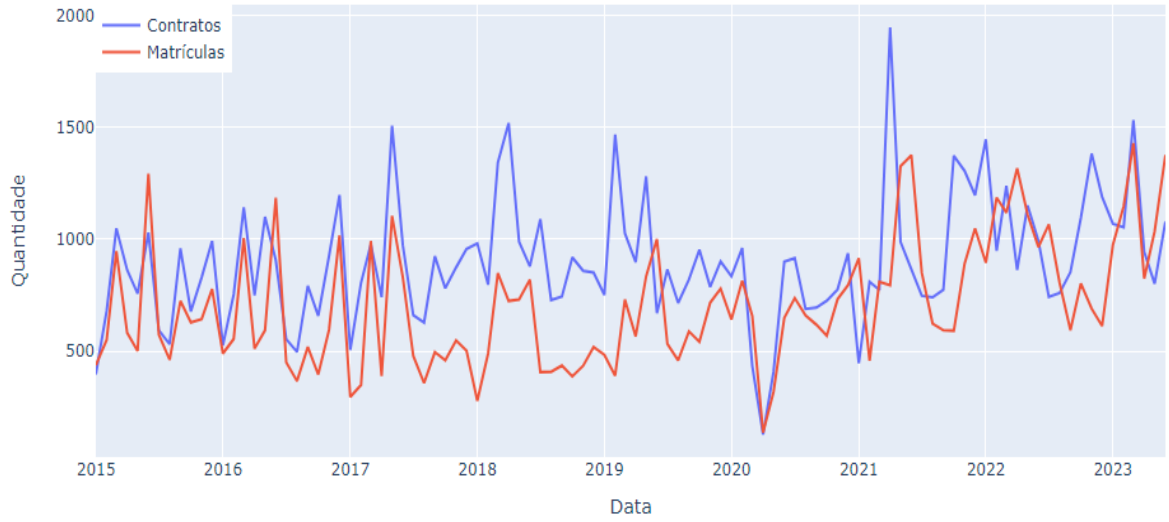


Figura 4 – Evolução temporal do número de matrículas mensais e do número de contratos mensais no período entre 2015 e 2023.

A partir da Figura 4 consegue-se concluir que tanto a procura, como as matrículas, seguem uma tendência crescente desde 2015, com uma quebra em 2020. Esta quebra deu-se devido à pandemia de COVID-19, com origem em Wuhan, que impactou não só a China, mas também todo o mundo.

Além disso, é possível visualizar, através dos picos do gráfico, que o início do ano é sempre a altura onde recai maior número de matrículas, à exceção de janeiro. O gráfico de séries temporais apresenta um crescimento todos os anos até o mês de junho, atinge um dos valores mais baixos em agosto, e vai recuperando os valores até ao fim do ano, em ritmo mais lento. O gráfico mostra também que, apesar dos valores das duas linhas seguirem uma variação idêntica na maioria das vezes, é possível notar que, por vezes, a procura antecede as matrículas. Isto é, o fechado um contrato nem sempre resulta numa matrícula imediata nesse mês e poderá adiar para o próximo.

A Figura 5 e a Figura 6 visam investigar a distribuição das matrículas e da procura por modelo de veículo. Cada modelo é representado por uma barra, cuja altura indica a quantidade de matrículas ou contratos registados para cada modelo. Através da análise dos gráficos, pode-se identificar quais são os modelos mais pretendidos pelos clientes. De acordo com a Figura 5, o modelo com maior número de matrículas desde 2015 é o Yaris, com quase o dobro do número do modelo Hilux. Explorando o resto do gráfico, percebe-se que existem modelos cujo número de matrículas é nulo ou próximo de zero. Isto pode ser justificado pelo baixo fornecimento de viaturas, pelo preço, pela tipologia do veículo e até mesmo pelo tempo de existência do mesmo.

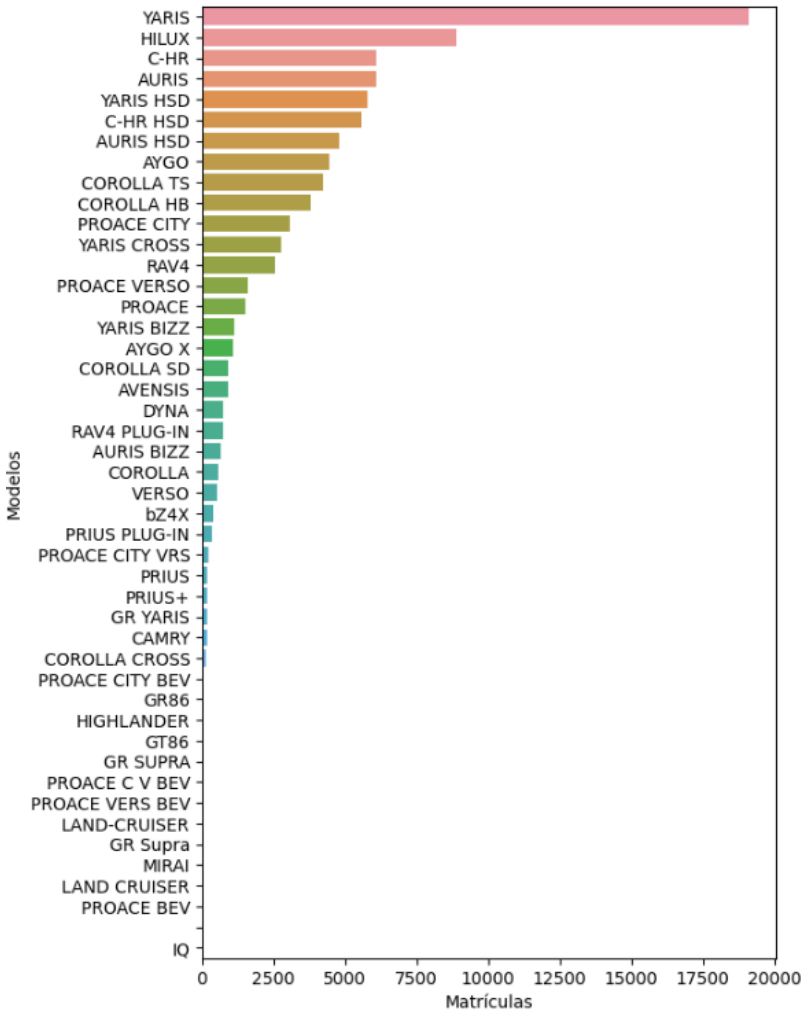


Figura 5 – Número de matriculas por modelo de veículo.

De acordo com a Figura 6, a diferença entre o modelo Yaris para a Hilux, no que toca a número de contratos, é inferior ao verificado na figura anterior. No entanto, os quatro modelos com maior número de contratos são exatamente os mesmos que no gráfico de matrículas por modelo, o que leva a concluir que o maior peso das bases de dados está repartido por estes quatro modelos.

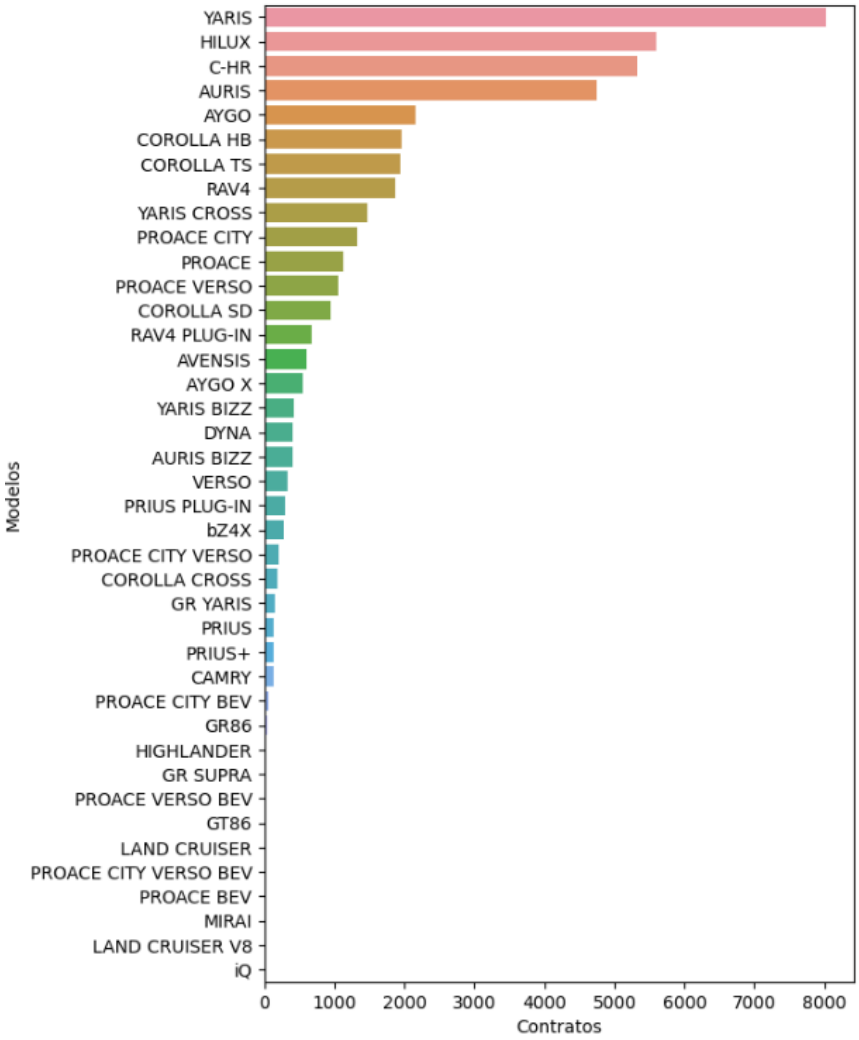


Figura 6 - Número de contratos por modelo de veículo.

3.3. Preparação dos dados (CRISP-DM)

Durante a fase de preparação dos dados, que precede a Modelação, foi essencial realizar a preparação dos mesmos, isto é, analisar cuidadosamente o conteúdo da base de dados e selecionar aqueles que mais se adequam ao trabalho.

Alguma desta preparação de dados foi realizada num momento anterior à compreensão dos dados, de forma a apresentar gráficos corretamente elaborados. Como foi o caso de organizar os valores no formato adequado e até mesmo agrupar por categorias relevantes.

Posto isso, foi necessário realizar diversas tarefas, tais como a seleção de tabelas, registo e atributos, limpeza, construção de novos atributos e transformação dos dados originais.

Foram selecionadas de forma minuciosa todas as informações com maior relevância para o estudo, descartando todas as outras, de forma a não criar ruído nos resultados.

Conforme se pode confirmar pela Figura 7, a base de dados de matrículas apresenta colunas com informação em branco em algumas células, não relevantes ao estudo, como é o caso de "MOTIVO_DESPACHO_ESPECIAL", "COD_VENDEDOR", "COD_STAND", "DESIGNACAO_STAND" e ainda "TIPO_VENDEDOR_ENC".

```
contagem_valores_nulos_matriculas = base_matriculas.isnull().sum()

print(contagem_valores_nulos_matriculas)
```

```
COD_CONCESSAO                 0
NOME_CONCESSAO                0
MARCA                         0
MODELO                        0
VERSAO                        0
CHASSI                        0
VIN                           0
MOTIVO_DESPACHO_ESPECIAL    85183
MATRICULA                     0
DT_MATRICULA                  0
DT_VENDA_CONC                 0
DT_VENDA_PUBLICO              0
GAMA                          0
COD_COR                       0
COD_VENDEDOR              46273
ZONA_CONCELHO_VENDA           0
PUBLICO                       0
NUM_HOMOLOGACAO               0
NUM_EXTENSAO                  0
CILINDRADA                    0
CO2_G_KM                      0
PARTICULAS_MG_KM              0
TIPO_ENCOMENDA                0
COD_VENDEDOR_ENC              0
COD_STAND                 32250
DESIGNACAO_STAND          32263
TIPO_VENDEDOR_ENC          2403
DATA_FACTURA_CONC             0
VAL_BASE_DEBITADO             0
VAL_ISV                       0
VAL_IUC_DEBITADO              0
VAL_DESCONTO                  0
VAL_INCENTIVO                 0
VAL_BASE_CONC                 0
VAL_BASE_PUBL                 0
VAL_IUC_TABELADO              0
```

Figura 7 - Contagem de valores nulos na base de dados de matrículas.

A base de dados contratos não apresentava nenhum campo em branco, conforme se pode confirmar na imagem abaixo:

```
contagem_valores_nulos = base_contratos.isnull().sum()

print(contagem_valores_nulos)
```

```
Marca           0
Híbrido         0
Gama            0
Versão          0
Código Sufixo   0
CC              0
Ano             0
Mês             0
Dia             0
```

Figura 8 - Contagem de valores nulos na base de dados de contratos.

Na base de dados de matrículas, sentiu-se a necessidade de alterar a coluna "DT_MATRICULA", (Figura 9) para variável do tipo data e assim construir uma linha temporal.

DT_MATRICULA
20150106
20150310
20150617
20150728
20150625
...

Figura 9 - Formato da coluna DT_MATRICULA.

Além de alterar a mesma variável para o tipo data, foi também criada uma coluna “Ano_Mês” (figura 10) no formato “AAAA-MM-DD”.

Ano_Mês
2015-04-01
2015-04-01
2015-04-01
2015-04-01
2015-04-01
...

Figura 10 - Formato da coluna Ano_Mês.

Uma nova coluna denominada “Ano_Mês (Figura 12) foi também criada para agregar as informações temporais previamente distribuídas em três colunas distintas (“Ano”, “Mês” e “Dia”), conforme podemos ver na Figura 11. Ao combinar toda a informação numa única coluna, foi possível representar cada data de contrato de forma mais compacta e conveniente.

Ano	Mês	Dia
2015	Abr	12
2015	Abr	14
2015	Abr	29
2015	Abr	2
2015	Abr	7
...	...	...

Figura 11 - Formato das colunas Ano, Mês e Dia.

Ano_Mês
2015-04-01
2015-04-01
2015-04-01
2015-04-01
2015-04-01
...

Figura 12 - Formato da coluna Ano_Mês.

Estas alterações foram necessárias para o bom funcionamento dos algoritmos, das métricas de avaliação utilizadas e ainda para manter uma lógica perante as duas bases de dados, isto é, na apresentação da informação de dada.

De seguida, foi sentida a necessidade de criar uma tabela (Figura 13), que apresentasse apenas a informação importante para o estudo. No caso da base de dados de matrículas, como uma linha

dizia respeito ao registo de uma matrícula, teve de ser feita uma contagem agrupada pela coluna "Ano_Mês". Após essa contagem, criou-se uma tabela de matrículas com duas colunas denominadas "Data" e "Matriculas", onde iria constar a informação do "Ano_Mês" e a contagem de matrículas, respetivamente.

	Data	Matriculas
0	2015-01-01	523
1	2015-02-01	623
2	2015-03-01	1081
3	2015-04-01	658
4	2015-05-01	579
...	...	...
97	2023-02-01	1155
98	2023-03-01	1442
99	2023-04-01	831
100	2023-05-01	1059
101	2023-06-01	1388

Figura 13 - Tabela matrículas pronta para análise.

Na tabela dos contratos existe uma coluna com o número de contratos realizados, e por isso, em vez de realizar uma contagem, foi necessário realizar a soma da coluna "CC", agrupada pela coluna "Ano_Mês". Essas duas colunas deram origem ao desenvolvimento de uma tabela contratos (Figura 14) com duas colunas ("Data" e "Contratos"), dizendo respeito à informação de "Ano_Mês" e o resultado da soma dos contratos agrupados.

	Data	Contratos
0	2015-01-01	411
1	2015-02-01	694
2	2015-03-01	1072
3	2015-04-01	883
4	2015-05-01	775
...	...	...
97	2023-02-01	1072
98	2023-03-01	1554
99	2023-04-01	998
100	2023-05-01	838
101	2023-06-01	1097

Figura 14 - Tabela de contratos pronta para análise.

Após a obtenção das tabelas referidas acima, foi necessário dividir ambas as bases de dados em conjunto de dados de treino e conjunto de dados de teste. Uma porção maior dos dados foi considerada para os dados de treino do modelo, enquanto uma menor foi reservada para testar o modelo.

Como o objetivo é prever as matrículas para um espaço temporal de, no mínimo, um ano, dividiu-se ambas as bases de dados da seguinte forma:

Figura 15 - Divisão dados treino e teste.

Na figura 15, pode visualizar-se a distribuição temporal dos dados utilizada em todos os algoritmos, à exceção das redes neuronais, onde foi necessário atribuir grande parte dos registos, de janeiro de 2015 até dezembro de 2021, ao conjunto de dados de treino. Isto porque, quanto maior fosse a base de dados de treino, maior era a probabilidade de os algoritmos aprenderem com o histórico dos dados.

Após a fase de treino, o conjunto de teste, correspondente a um ano e seis meses, foi utilizado para avaliar o desempenho das previsões alcançadas. Este conjunto de dados não entra no processo de treino, o que proporciona uma avaliação mais realista do desempenho das previsões.

Na Figura 16 pode-se observar que, para as redes neuronais, a repartição foi feita em três partes distintas: treino, validação e teste. A maior porção dos dados é reservada para o treino (70%), enquanto os dados de validação e teste contém uma proporção do mesmo tamanho, 15% cada um, atingindo os 30% restantes que sobraram dos dados de treino. Esta divisão não foi feita de forma aleatória, por isso a repartição irá respeitar a sequencia temporal do conjunto de dados, isto é, o início da base de dados irá cair nos dados de treino, depois nos de validação e por último nos dados de teste. Esta divisão foi utilizada, especificamente, no algoritmo de redes neuronais para realizar uma verificação dos dados de treino e melhorá-los, de forma a fornecer melhores previsões.

Figura 16 - Divisão dados treino, validação e teste

3.4. Modelação e Avaliação (CRISP-DM)

Na etapa de modelação, a seleção dos algoritmos mais adequados a um problema de séries temporais é um passo crucial para obter resultados precisos e eficazes. Existem diferentes algoritmos possíveis de utilizar, desenvolvidos para ser aplicados em diferentes cenários. A escolha vai depender sempre das características dos dados e do objetivo final da análise.

Neste trabalho, foram aplicados diversos algoritmos de previsão, com o intuito de identificar qual deles se mostrava mais eficaz para prever as matrículas ao longo do tempo. Os algoritmos utilizados foram os seguintes:

- Algoritmo Alisamento Exponencial Simples;

- Algoritmo Alisamento Exponencial Sazonal;
- Algoritmo SARIMA;
- Algoritmo ARIMA;
- Algoritmo Redes Neuronais.

3.4.1. Alisamento Exponencial Simples

Inicialmente, foi aplicado, à base de dados matrículas e contratos, o algoritmo de alisamento exponencial simples com o propósito de realizar previsões tendo em conta os valores passados da série temporal. Foram atribuídos pesos exponenciais decrescentes aos valores passados, permitindo que os valores mais recentes tenham maior influência na previsão.

Observando os resultados obtidos através do histórico da base de dados de matrículas, conseguiu-se perceber que o modelo foi capaz de capturar padrões de tendência suave, contudo, bastante limitado a identificar sazonalidades mais complexas.

Conforme se pode observar na Figura 17, a previsão tem um valor constante de 852 matrículas. Isto acontece porque este algoritmo calcula a média ponderada dos valores passados para prever valores de matrículas futuras.

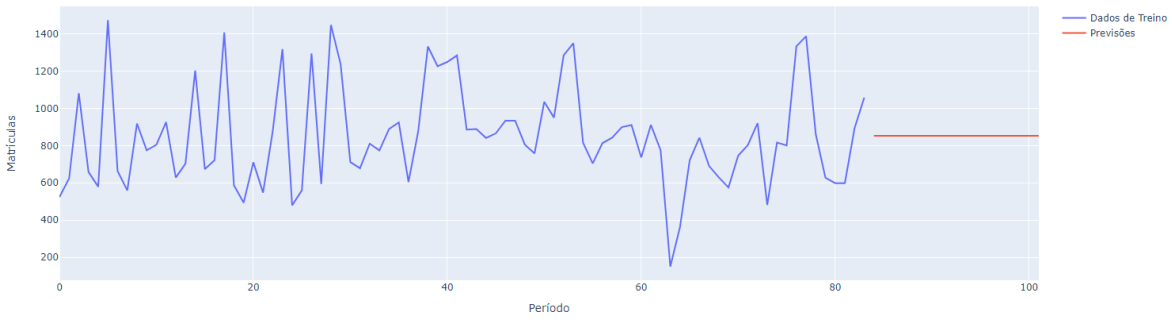


Figura 17 – Série de matrículas (azul) e previsão (laranja) obtida pelo método Alisamento Exponencial Simples.

De seguida, foi aplicado o mesmo algoritmo à base de dados dos contratos. Como se pode observar na Figura 18, o algoritmo conseguiu prever um valor superior de matrículas, cerca de 1117 no mesmo período temporal.

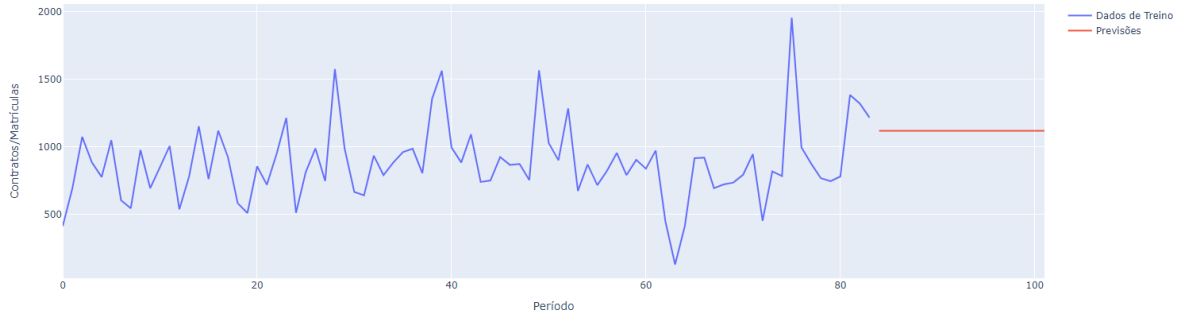


Figura 18 - Série de contratos (azul) e previsão (laranja) obtida pelo método Alisamento Exponencial Simples.

De forma a avaliar o desempenho do algoritmo no conjunto de teste, foram aplicadas algumas métricas de avaliação de desempenho. Os resultados desta avaliação estão na Tabela 6.

Tabela 6 – Avaliação do desempenho do método Alisamento Exponencial Simples.

	EQM	REQM	EAM	EPAM
Dados das Matrículas	81450,2	285,4	234,5	28,5%
Dados dos Contratos	69223,4	263,1	212,2	18,9%

Surpreendentemente, a aplicação do algoritmo na base de dados de contratos trouxe melhores resultados ao compararem-se as métricas de avaliação com os resultados obtidos na base de dados de matrículas. Esta diferença pode dever-se ao facto de a informação histórica contida na base de dados de contratos ser mais propícia à aplicação do algoritmo, isto é, pode conter dados com uma sazonalidade inferior, menos impactada pelos fatores externos.

3.4.2. Alisamento Exponencial Sazonal

Tendo em consideração a possibilidade de existência de sazonalidade nos dados, foi aplicado o método Alisamento Exponencial Sazonal. Na figura abaixo, são apresentados os resultados da aplicação do método Alisamento Exponencial Sazonal aos dados das matrículas. De acordo com esta figura, o método aplicado parece conseguir não só detetar a tendência geral dos dados, como também revelou capacidade de identificar alguns padrões sazonais.

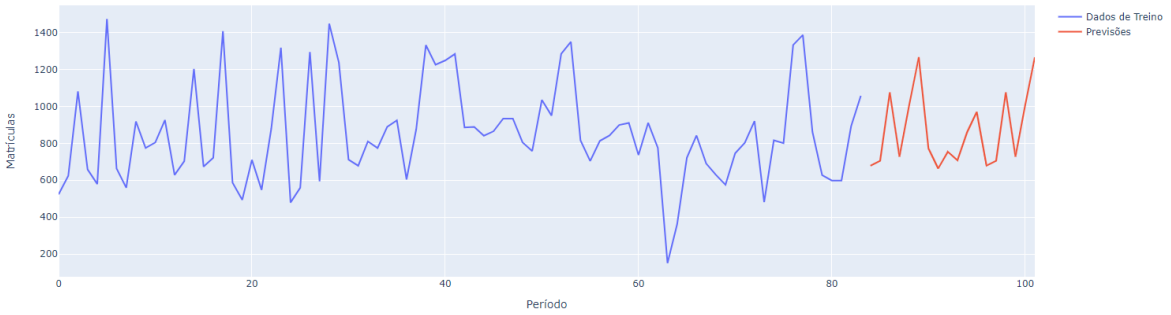


Figura 19 – Série de matrículas (azul) e previsão (laranja) obtida pelo método Alisamento Exponencial Siazonal.

Na Figura 20, apresentam-se as previsões para o número de contratos obtidas com o método Alisamento Exponencial Sazonal.

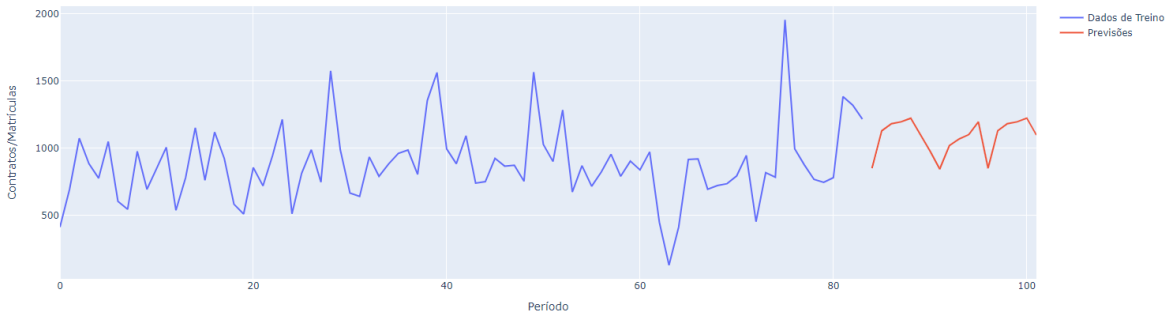


Figura 20 - Série de contratos (azul) e previsão (laranja) obtida pelo método Alisamento Exponencial Siazonal.

Comparando as previsões para o número de contratos com as previsões para o número de matrículas, pode concluir-se que, para os contratos, existe um padrão sazonal mais constante, com picos menos salientes.

Na Tabela 7, apresentam-se os resultados da avaliação do desempenho do método Alisamento Exponencial Siazonal. Comparando os resultados da Tabela 7 com os da Tabela 6, consegue-se concluir que o método de Alisamento Exponencial Sazonal apresenta melhor desempenho. Isto pode representar que, tendo em conta os dados do estudo, além da tendência dos dados é também importante ter em consideração os padrões sazonais dos mesmos.

Tabela 7 - Avaliação do desempenho do método Alisamento Exponencial Sazonal.

	EQM	REQM	EAM	EPAM
Alisamento exponencial sazonal - Matrículas	82741,0	287,6	242,5	27,9%
Alisamento exponencial sazonal - Contratos	61031,8	247,0	196,2	18,1%

3.4.3. ARIMA

Foi também utilizado o algoritmo ARIMA para a análise de séries temporais, pois este tem a capacidade de efetuar uma modelação sólida e robusta de tendências. Além disso, oferece a possibilidade de alterar manualmente os parâmetros do modelo (p, d, q), concedendo um nível de controlo mais aprofundado na adaptação do modelo às características dos dados.

Neste trabalho, inicialmente, o ajuste dos valores dos parâmetros foi feito de forma manual. Iniciou o estudo com $p = 21$, $d = 1$ e $q = 1$, o que representa que a série foi diferenciada uma vez para a tornar estacionária ($d = 1$) e que a variável em estudo foi escrita em função de 21 valores passadas.

Aplicando o método ARIMA com os valores anteriores à base de dados matrículas, obtiveram-se as previsões presentes na Figura 21.

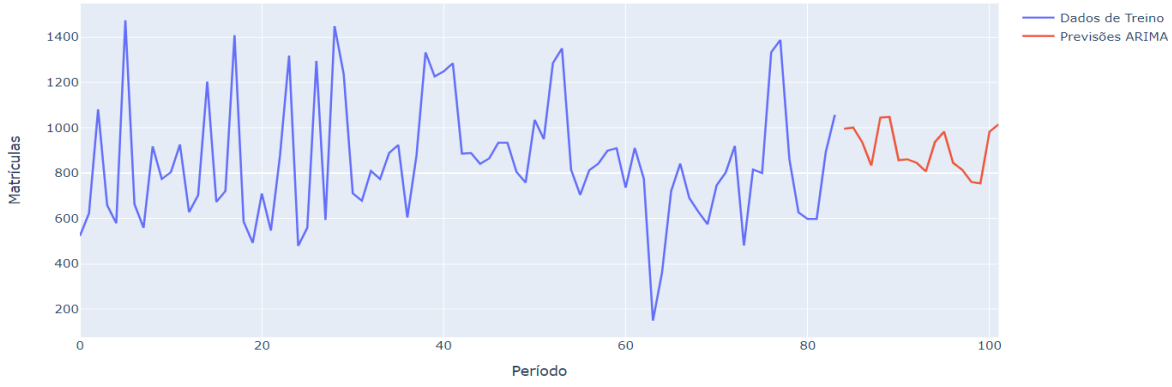


Figura 21 - Série de matrículas (azul) e previsão (laranja) obtida pelo método ARIMA.

As previsões obtidas apresentaram uma amplitude inferior àquela que foi observada nos dados de treino. Isto levanta a possibilidade de que o algoritmo não está a aprender corretamente com o histórico da base de dados de matrículas. Uma amplitude inferior sugere que o modelo pode estar subestimado a variabilidade ou não está a recolher corretamente os padrões dos dados.

De seguida, e como alternativa complementar, utilizou-se o método auto-ARIMA, de modo a identificar automaticamente os melhores parâmetros a utilizar. Esta abordagem permitiu uma exploração sistemática das diferentes configurações dos parâmetros na tentativa de atingir previsões otimizadas para os dados das matrículas. Os parâmetros foram os seguintes: $p = 3, d = 0$ e $q = 4$.

A Figura 22 apresenta as previsões do número de matrículas. É possível concluir que, apesar das previsões demonstrarem uma sazonalidade constante, a amplitude diminuiu. Isto parece mostrar que, neste caso, a automatização dos parâmetros poderá não ser a melhor abordagem para identificar as configurações ideais.

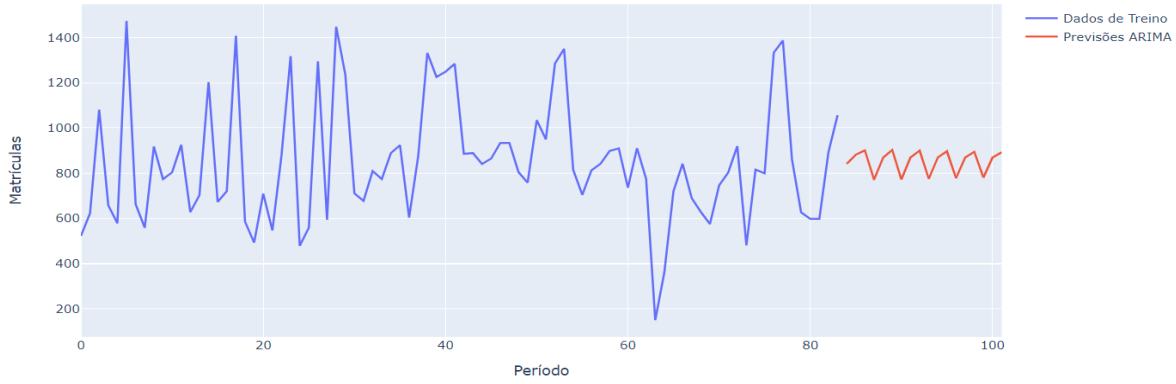


Figura 22 - Série de matrículas (azul) e previsão (laranja) obtida pelo método auto-ARIMA.

Analisando o desempenho do algoritmo aplicado à base de dados de contratos (Figura 23), pode-se observar que, com os mesmos parâmetros manuais utilizados anteriormente $(p = 21, d = 1, q = 1)$, a amplitude segue uma tendência semelhante àquela que foi previamente observada nos dados de matrículas. Isto pode significar que, efetivamente, o algoritmo não é adequado para os dados em questão, uma vez que parece não estar a capturar efetivamente as tendências e flutuações, apresentando dificuldades em aprender e modelar os padrões da série temporal de contratos.

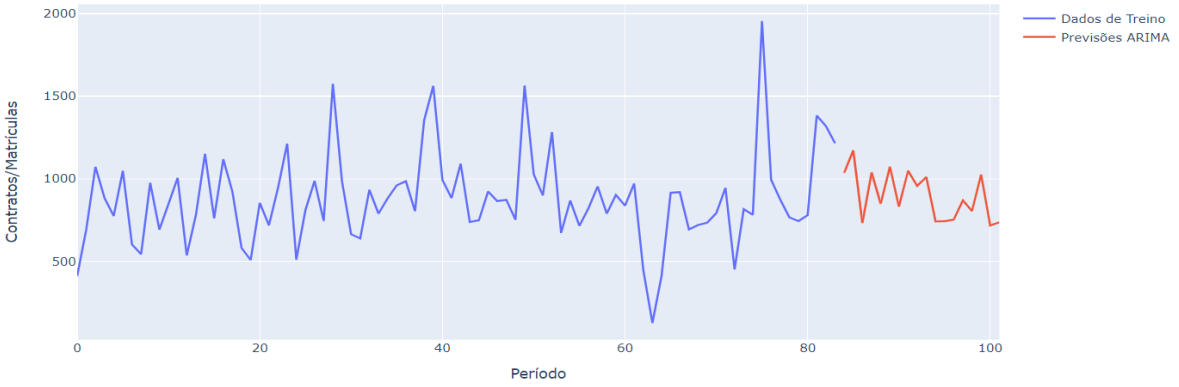


Figura 23 - Série de contratos (azul) e previsão (laranja) obtida pelo método ARIMA.

De igual forma, tentou-se também aplicar o método auto-ARIMA à base de dados dos contratos. Os resultados estão representados na Figura 24. Este método devolveu previsões quase constantes, o que indica que os dados de contratos não são bem modelados pelo método auto-ARIMA.

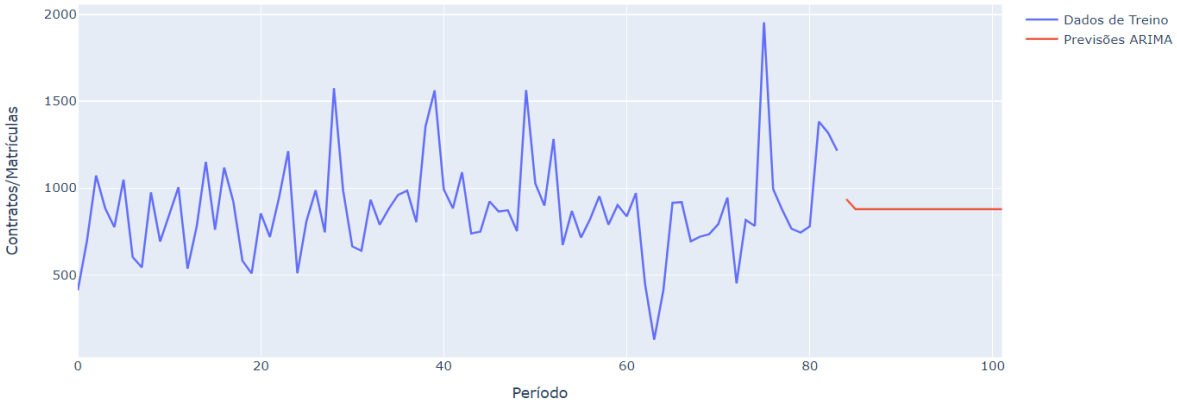


Figura 24 - Série de contratos (azul) e previsão (laranja) obtida pelo método auto-ARIMA.

Na Tabela 8, apresentam-se os resultados da análise de desempenho do método ARIMA e do método auto-ARIMA. Analisando os resultados obtidos, é evidente que o método ARIMA aplicado à base de dados de matrículas apresentou um desempenho mais satisfatório. Além disso, apesar do esforço em tentar encontrar padrões mais eficazes por meio de métodos automatizados, como o auto-ARIMA, os resultados são piores com este método.

Tabela 8 - Avaliação do desempenho do método ARIMA.

	EQM	REQM	EAM	EPAM
ARIMA - Matrículas	75589,8	275,0	215,7	23,8%
ARIMA - Contratos	96716,6	311,0	262,0	29,2%
ARIMA com auto-ARIMA - Matrículas	83574,4	289,1	241,8	28,4%
ARIMA com auto-ARIMA - Contratos	74115,5	272,2	226,3	25,6%

3.4.4. SARIMA

Reconhecendo o impacto da sazonalidade dos dados em análise, optou-se por aplicar uma variação do método ARIMA, incorporando a modelação da sazonalidade - modelo SARIMA.

Apesar do algoritmo ARIMA ser eficaz a detetar tendências e flutuações irregulares, a sua capacidade de lidar com a sazonalidade é limitada. Ao aplicar o método SARIMA, pretende-se colmatar essa limitação, permitindo uma modelação mais precisa dos dados e detetar padrões sazonais.

A aplicação do modelo SARIMA também exigiu ajustes manuais nos parâmetros, tanto para a parte não sazonal (p, d, q) do modelo quanto para a parte sazonal (P, D, Q, S). Após algumas tentativas e ajustes, foram identificados os parâmetros que resultaram em melhores previsões, isto é, $p = 21, d = 1, q = 1,\ P = 0,\ D = 1,\ Q = 1$ e por fim $S = 12$. Cada um destes parâmetros desempenha uma função especifica na modelação de tendências, flutuações e sazonalidades na série temporal.

Na Figura 25, pode-se observar que o modelo conseguiu detetar padrões sazonais presentes nos dados. As previsões obtidas apresentam uma amplitude que corresponde aos dados de treino, o que mostra que o modelo foi capaz de capturar variações sazonais de forma adequada.

Figura 25 - Série de matriculas (azul) e previsão (laranja) obtida pelo método SARIMA.

Procurando obter valores dos parâmetros otimizados e assim prever com mais eficácia o número de matrículas, optou-se mais uma vez por recorrer a mecanismos automatizados para a sua definição. Os parâmetros automáticos foram: $p = 0, d = 0, q = 1, P = 2, D = 0, Q = 0$ e $S = 12$. Este método é semelhante ao auto-ARIMA, porém tem a particularidade de considerar tanto a parte não sazonal quanto a parte sazonal. A Figura 26 mostras as previsões obtidas.

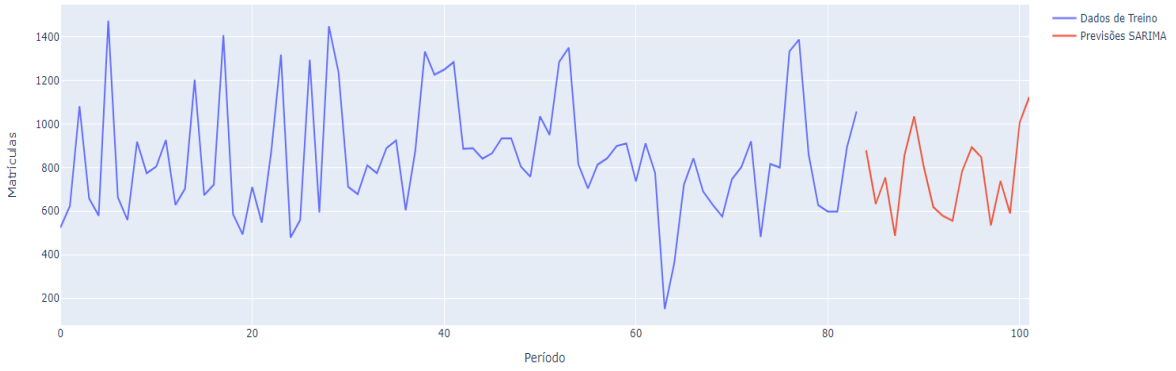


Figura 26 - Série de matriculas (azul) e previsão (laranja) obtida pelo método auto-SARIMA.

Aplicando o mesmo algoritmo à base de dados de contratos com os seguintes parâmetros manuais $p = 21, d = 1, q = 1, P = 0, D = 1, Q = 1, S = 12$, observaram-se características distintas em relação aos padrões presentes na série temporal. Analisando a Figura 27, onde estão apresentados os resultados das previsões, concluiu-se que há uma tendência de crescimento com o tempo, sugerindo um aumento gradual nos valores das matrículas. Além disso, a sazonalidade parece baixa, indicando variações menos regulares em intervalos específicos.
No que toca à avaliação visual da linha das previsões, estas parecem não ter sido eficazes, além de que a sua amplitude parece não corresponder aos dados históricos.

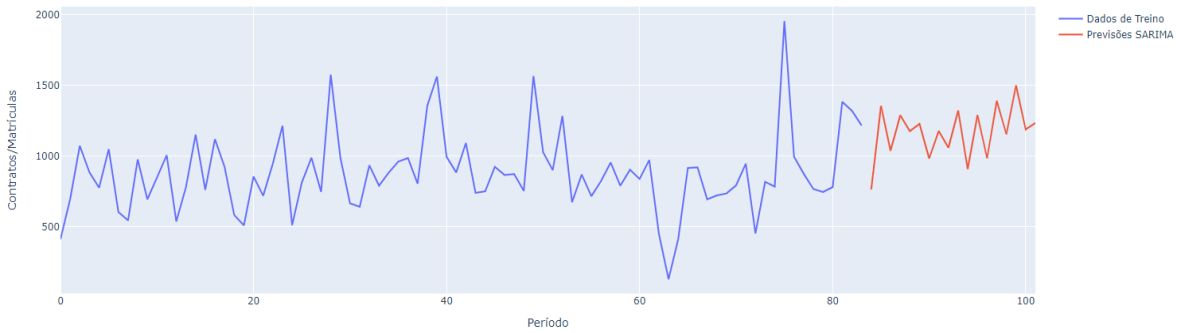


Figura 27 - Série de contratos (azul) e previsão (laranja) obtida pelo método SARIMA.

Além de aplicação do modelo SARIMA à base de dados de contratos através de configuração manual dos parâmetros, foi também decidido testar se com parâmetros automáticos os resultados atingidos seriam mais coerentes e interpretáveis.

Os parâmetros automáticos foram: $p = 0, d = 0, q = 1, P = 0, D = 0, Q = 0\ e\ S = 12$. Ao observar o gráfico resultante (Figura 28), percebeu-se que a previsão foi gerada apenas para um mês, e o restante dos meses não apresentou qualquer previsão. Esse resultado não corresponde com a expectativa de previsões pretendidas.

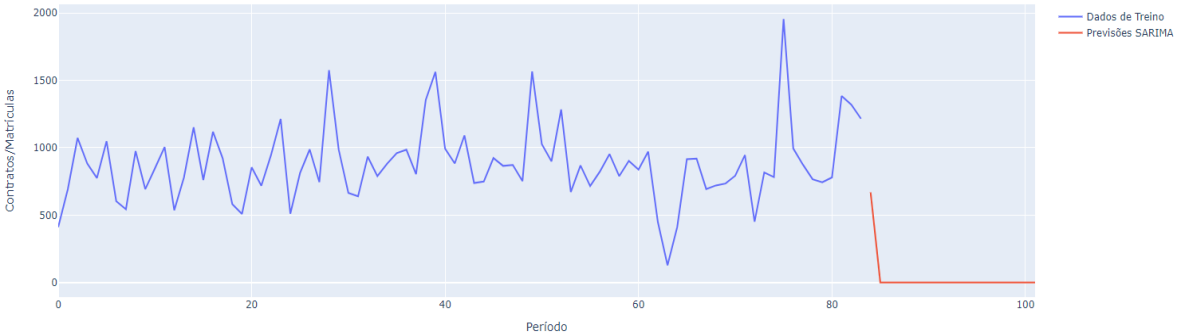


Figura 28 - Série de contratos (azul) e previsão (laranja) obtida pelo método auto-SARIMA.

Nesta abordagem, é interessante notar que, apesar das limitações visuais apresentadas pela linha de previsões, gerada pelo algoritmo aplicado à base de dados de contratos com os parâmetros manuais, os resultados obtidos por meio da métrica EPAM foram os mais positivos, atingindo uma taxa de erro de 21,3% (Tabela 9). Esta observação ressalta a importância de avaliar os resultados não apenas com recurso das visualizações gráficas das previsões, mas também considerando as métricas quantitativas que fornecem uma média objetiva da precisão do modelo.

Tabela 9 - Avaliação do desempenho do método SARIMA.

	EQM	REQM	EAM	EPAM
SARIMA - Matrículas	71754,0	267,9	221,5	22,5%
SARIMA - Contratos	100732,7	317,4	249,0	21,3%
SARIMA com auto-SARIMA - Matrículas	139044,5	372,8	290,5	38,1%
SARIMA com auto-SARIMA - Contratos	1030802,9	1015,3	970,8	2614,8%

3.4.5. Redes Neuronais

Dado que a base de dados apresenta alguma complexidade e, possivelmente, algumas relações não lineares, optou-se por explorar o algoritmo de redes neuronais, no sentido de captar padrões difíceis de detetar. Com este algoritmo, pretende-se que seja realizada uma análise mais completa e detalhada da base de dados, com o objetivo de capturar sazonalidades irregulares e relações não detetadas em métodos anteriores.

Na Figura 29 é possível visualizar que a base de dados foi repartida em dados de treino, dados de validação e por fim, dados de teste, que serviram para avaliar a eficácia dos resultados das previsões realizadas. Estas previsões mostram um maior sentido face à sazonalidade e aos padrões dos dados de treino e validação, o que justifica a sua utilização.

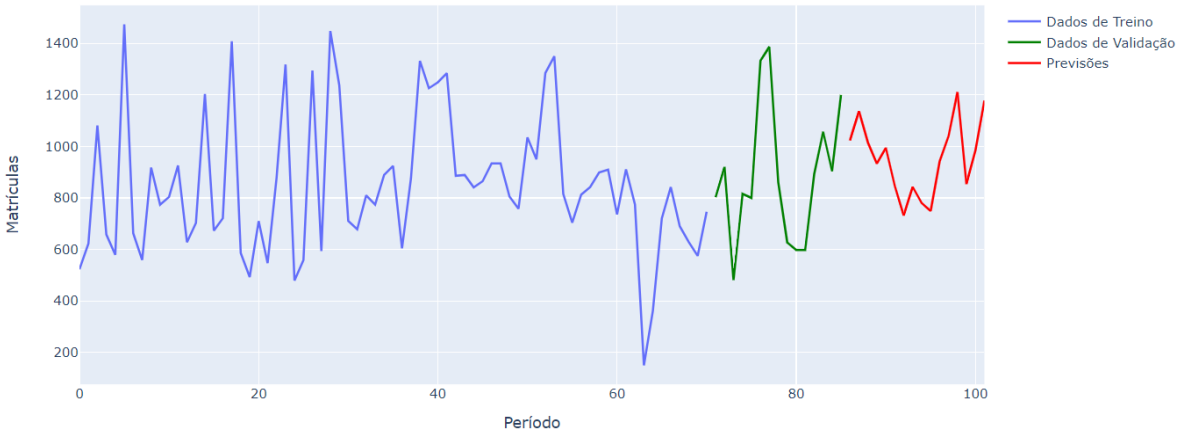


Figura 29 - Gráfico de redes neuronais aplicado à base de dados de matrículas.

Após a exploração meticulosa do gráfico resultante, pretendeu-se garantir a otimização dos resultados através do desenvolvimento de um gráfico que ilustra as evoluções de “Train loss” e “Val loss”, em função do número de épocas. Esta abordagem permitiu entender se o número de épocas (100) estava de tal forma bem definido para o bom funcionamento do algoritmo. Observando a Figura 30, pode-se concluir que houve um ponto de inflexão próximo das 30 épocas, na qual ambas as curvas passam a apresentar uma tendência de estabilização. Esta estabilização é percetível pois ambas as curvas deixam de apresentar um declínio acentuado.

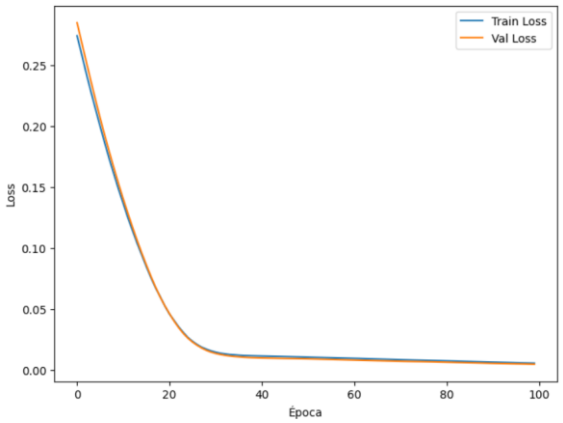


Figura 30 - Gráfico Train Loss e Val Loss (matrículas).

Por outro lado, aplicando o algoritmo de redes neuronais à base de dados de contratos, resultados apresentados na Figura 31, pode-se notar que, apesar de se ter obtido uma sazonalidade visualmente mais aprimorada em comparação com outros algoritmos, essa representação parece não corresponder e acompanhar tanto a realidade quanto nas previsões realizadas na base de dados de matrículas.

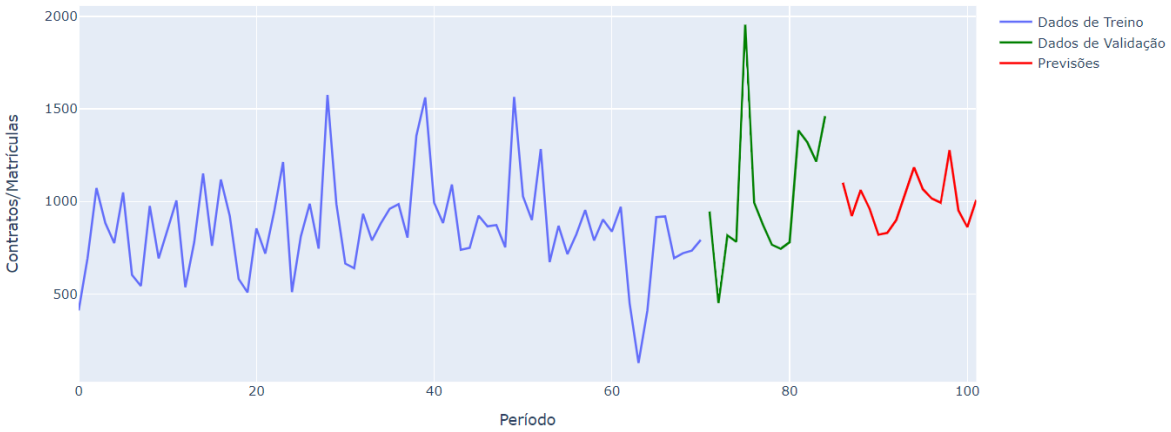


Figura 31 - Gráfico de redes neuronais aplicado à base de dados de contratos.

Assim como na base de dados de matrículas, aqui também foi realizado o gráfico que ilustra as evoluções de “Train loss” e “Val loss”. Esta ilustração (Figura 32) permite confirmar que os resultados previstos não apresentaram tanta eficácia, comparando com os atingidos na base de dados de matrículas, isto porque as linhas apresentam uma maior distancia entre si. Esta maior distância significa que o modelo não tem tanta capacidade para aprender com os dados de treino e gerar novos dados.

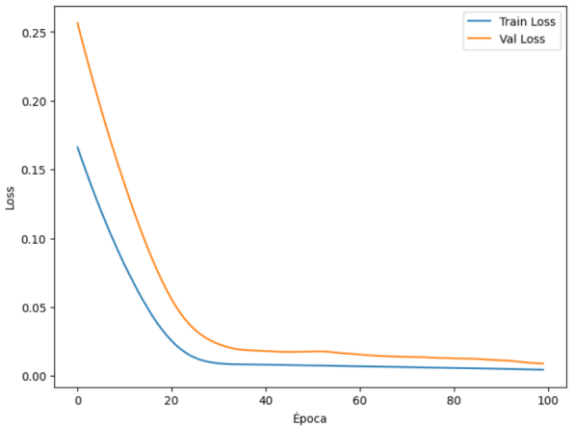


Figura 32 - Gráfico Train Loss e Val Loss (Contratos).

Também é evidente uma estabilização das linhas de “Train loss” e “Val loss”. Pode-se então interpretar que esta estabilização se deve ao facto de o algoritmo começar a convergir para um ponto de equilíbrio. À medida que o mesmo é submetido a mais épocas de treino, é absorvida mais informação e aprende mais com os padrões presentes nos dados de treino e validação.

Aplicando as métricas de avaliação para ambas as bases de dados, obtiveram-se os resultados apresentados na Tabela 10, considerando 100 épocas.

Tabela 10 - Redes neuronais (100 épocas) – Avaliação de desempenho.

	EQM	REQM	EAM	EPAM
Redes Neuronais – Matrículas (100 épocas)	9107,0	95,4	81,0	8,1%
Redes Neuronais - Contratos (100 épocas)	64233,8	253,4	201,3	20,0%

A confirmação das conclusões retiradas anteriormente, baseada nos valores de EPAM, fortalece a análise e o desempenho do algoritmo de redes neuronais. O algoritmo redes neuronais aplicados à base de dados de matrículas mostrou uma maior capacidade de aprender os padrões subjacentes e aplicá-los com sucesso a cenários não vistos.

Devido à estabilização visual das curvas de "Train loss" e "Val loss" antes das 100 épocas, decidiu-se explorar o desempenho do algoritmo com a alteração desse parâmetro. Para tal, foi aplicado novamente o algoritmo nas bases de dados, mas alterando o parâmetro para 75 épocas (Tabela 11) e 50 épocas (Tabela 12).

Tabela 11 - Redes neuronais (75 épocas) - Avaliação de desempenho.

	EQM	REQM	EAM	EPAM
Redes Neuronais – Matrículas (75 épocas)	19037.5	138.0	117.2	11.7%
Redes Neuronais - Contratos (75 épocas)	62213.1	249.4	208.5	20.8%

Tabela 12 - Redes neuronais (50 épocas) - Avaliação de desempenho.

	EQM	REQM	EAM	EPAM
Redes Neuronais – Matrículas (50 épocas)	28368.1	168.4	142.0	14.2%
Redes Neuronais – Contratos (50 épocas)	62163.3	249.3	207.8	20.7%

Após terem sido explorado diferentes cenários, ajustando o número de épocas para 75 e 50, foi evidente que a escolha deste parâmetro contribuiu, em grande parte, para o bom funcionamento do algoritmo de redes neuronais e a sua aprendizagem. Constatou-se que os resultados atingidos com 100 épocas superaram aqueles atingidos com 75 e 50 o que demonstra que, por vezes, é necessário ter um treino mais extenso para captar melhor padrões mais complexos.

Por fim, concluiu-se que apesar de o gráfico de "Train loss" e "Val loss" ter estabilizado visualmente antes das 100 épocas, a continuação do treino resultou num aprimoramento das previsões.

4. RESULTADOS E DISCUSSÃO

Este capítulo apresenta os resultados obtidos ao longo deste trabalho, juntamente com uma análise detalhada, resultante da aplicação das técnicas referidas anteriormente. Assim, dado o caráter quantitativo deste trabalho, recorre-se a visualizações gráficas e tabelas para ilustrar os resultados, permitindo uma discussão aprofundada sobre aquele que melhor satisfaz os nossos objetivos e é determinante para o fator de sucesso do trabalho.

A análise crítica e a interpretação dos resultados atingidos foram fundamentais para retirar conclusões significativas, as quais proporcionaram uma compreensão profunda do tema em estudo. Ao longo do trabalho, foram utilizados vários algoritmos de previsão que captaram padrões, tendências e até mesmo sazonalidades presentes nos dados. Prevê-se, assim, compreender o desempenho dos algoritmos e até recolher aquele que apresentou melhores resultados do que aqueles previstos anteriormente pela Toyota Caetano Portugal.

4.1. Apresentação de resultados

O desempenho de cada algoritmo foi avaliado anteriormente por meio de quatro métricas de avaliação. Estas métricas desempenham um papel fundamental na análise e interpretação dos resultados obtidos, uma vez que permitem, facilmente, identificar a precisão dos mesmos.

Os valores das métricas são importantes pois permitiram entender quão bem os modelos estavam ajustados aos dados, e realizar reajustes sempre que fosse necessário. Além disso, permitem ainda comparar os resultados atingidos pelos diferentes algoritmos, de forma a fornecer informações valiosas no decorrer do trabalho e até mesmo nas conclusões finais.

Como se pode observar na Tabela 13, todos os algoritmos aplicados no decorrer do trabalho foram avaliados consoante a sua eficácia. As métricas de avaliação foram as mesmas, o que permite uma comparação justa entre modelos. É importante realçar que, todos estes resultados obtidos tiverem em consideração os dados de teste, inicialmente definidos, e as previsões alcançadas por cada um.

Na Tabela 13 pode-se observar que existem variações significativas nas métricas de desempenho. A aplicação do algoritmo de Alisamento Exponencial Simples obteve o valor de 28,5% e 18,9% no conjunto de dados de matrículas e contratos, respetivamente.

O algoritmo Alisamento Exponencial Sazonal alcançou uma taxa de erro de 27,9% na base de dados de matrículas e 18,1% na base de dados de contratos. Ainda assim, estes resultados apresentaram uma melhoria pouco significativa face ao algoritmo de Alisamento Exponencial Simples.

O método ARIMA apresentou resultados com uma taxa de erro de 23,8% para as matrículas e 29,2% para os contratos. Na base de dados de matrículas e recorrendo à automatização dos parâmetros, os resultados demonstraram-se piores, tendo obtido valores de 28,4%. Já na base de dados de contratos, os resultados com os parâmetros automáticos melhoraram ligeiramente para os 25,6%.

O método SARIMA, que incorpora a sazonalidade no método ARIMA, apresentou uma taxa de erro de 22,5% na base de dados de matrículas e 21,3% na base de dados de contratos. A situação demonstrou-se menos favorável quando foram automatizados os parâmetros, pois atingiu valores de 38,1% na base de dados de matrículas e 2614,8% na base de dados de contratos.

Por fim, o algoritmo de redes neuronais foi aplicado com 100, 75 e 50 épocas. Com 100 épocas, o erro médio para as matrículas foi de 8,1% e 20,0% para os contratos. Aplicando o algoritmo com 75 épocas, foram atingidos erros ligeiramente superiores, 11,7% para as matrículas e 20,8% para os contratos. Com 50 épocas, vemos as taxas de erro a aumentar, tendo atingido os 14,2% na base de dados de matrículas e 20,7% na base de dados de contratos.

Concluiu-se, assim, que o algoritmo de redes neuronais, aplicado à base de dados de matrículas, conseguiu alcançar melhores resultados de taxas de erro, isto é, 8,1%.

Tabela 13 – Resumo da avaliação de desempenho de todos os algoritmos.

	EQM	REQM	EAM	EPAM
Alisamento exponencial simples - Matrículas	81450,2	285,4	234,5	28,5%
Alisamento exponencial simples - Contratos	69223,4	263,1	212,2	18,9%
Alisamento exponencial sazonal - Matrículas	82741,0	287,6	242,5	27,9%
Alisamento exponencial sazonal - Contratos	61031,8	247	196,2	18,1%
ARIMA - Matrículas	75589,8	275	215,7	23,8%
ARIMA - Contratos	96716,6	311	262	29,2%
ARIMA com auto-ARIMA - Matrículas	83574,4	289,1	241,8	28,4%
ARIMA com auto-ARIMA - Contratos	74115,5	272,2	226,3	25,6%
SARIMA - Matrículas	71754,0	267,9	221,5	22,5%
SARIMA - Contratos	100732,7	317,4	249	21,3%
SARIMA com auto-SARIMA - Matrículas	139044,5	372,8	290,5	38,1%
SARIMA com auto-SARIMA - Contratos	1030802,9	1015,3	970,8	2614,8%
Redes Neuronais – Matrículas (100 épocas)	9107,0	95,4	81,0	8,1%
Redes Neuronais – Contratos (100 épocas)	64233,8	253,4	201,3	20,0%
Redes Neuronais – Matrículas (75 épocas)	19037,5	138	117,2	11,7%
Redes Neuronais – Contratos (75 épocas)	62213,1	249,4	208,5	20,8%
Redes Neuronais – Matrículas (50 épocas)	28368,1	168,4	142	14,2%
Redes Neuronais – Contratos (50 épocas)	62163,3	249,3	207,8	20,7%

4.2. Discussão de resultados

A análise de resultados, apresentados na Tabela 13, fornece a possibilidade de retirar conclusões valiosas sobre o desempenho dos diversos algoritmos aplicados ao conjunto de dados em estudo.

Começando por explorar a avaliação dos métodos ARIMA e SARIMA, torna-se evidente que ambos enfrentam dificuldades em capturar tendências e sazonalidades intrínsecas aos dados. Os erros superiores a 20% demonstram que as previsões não conseguiram refletir adequadamente os valores reais, demonstrando a complexidade de trabalhar com esses padrões. Mesmo recorrendo a mecanismos automatizados para melhorar o desempenho, não se conseguiu verificar avanços nos resultados obtidos, pois os mesmos não conseguiram definir parâmetros satisfatórios para a eficácia dos algoritmos.

Ao analisar o algoritmo de Alisamento Exponencial Simples, apesar de apresentar resultados satisfatórios de erro (18,9%), ao ser aplicado à base de dados de contratos, as previsões obtidas não apresentavam qualquer sazonalidade. Esta observação reforça a importância de considerar a sazonalidade para obter precisões precisas e relevantes.

Contrastando com o Alisamento Exponencial Simples, o método de Alisamento Exponencial Sazonal apresentou resultados satisfatórios de erro (18,1%), ao ser aplicado à base de dados de contratos. Além disso, destacou-se por capturar de forma mais eficaz a sazonalidade quando comparado com o algoritmo exponencial simples.

Destacando-se pelos resultados surpreendentes, o algoritmo de redes neuronais mostrou-se bastante capaz de captar tanto as tendências, como as flutuações e padrões sazonais. Este, foi testado com diferentes números de épocas (100, 75 e 50), e revelou-se o mais promissor entre os algoritmos estudados. Aplicado à base de dados de matrículas, o algoritmo de redes neuronais atingiu um erro médio de 11%, enquanto quando aplicado na base de dados de contratos apresentou uma média de erro de 20,5%.

Notavelmente, o algoritmo de redes neuronais, quando aplicado à base de dados de matrículas, com 100 épocas, demonstrou um desempenho considerável, com um erro mínimo de 8,1%. Isto traduz-se numa precisão bastante significativa, tendo em conta a complexidade do conjunto de dados e do negócio em si. De forma prática, significa que, em média, as previsões teriam um desvio de apenas 81 carros por mês, considerando um total de 1000 carros vendidos por mês. Estes resultados comprovam a vantagem da utilização das redes neuronais para proporcionar previsões de matrículas em constante evolução.

Esta proximidade entre as previsões e as matrículas reais pode ser claramente observada recorrendo à Figura 33, onde a linha vermelha representa as previsões alcançadas e a linha amarela corresponde às matrículas reais. A proximidade é evidente, validando assim os resultados obtidos pelo erro absoluto médio (EPAM) de apenas 8,1%. Essa baixa taxa de erro reflete a capacidade de o algoritmo de redes neuronais capturar com sucesso o comportamento dos dados reais.

Através dos dados de treino e validação, o algoritmo conseguiu identificar e aprender padrões, tendências e sazonalidades presentes nos dados históricos. O algoritmo mostrou-se assim capaz de projetar de maneira precisa e eficaz as previsões de matrículas futuras.

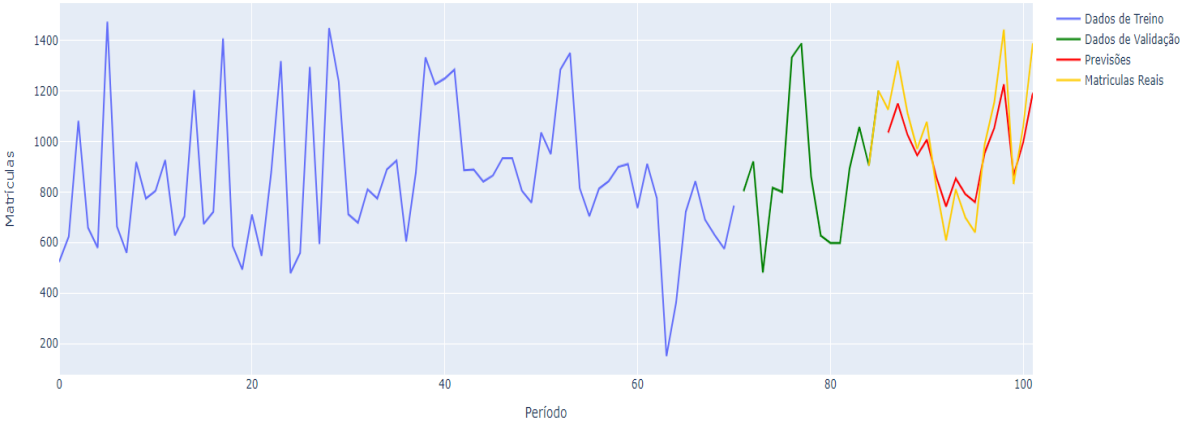


Figura 33 – Gráfico de redes neuronais (base de dados de matrículas) e matrículas reais

Ao analisarem-se os dados previstos pela Toyota para o mesmo espaço de tempo (Tabela 14), notou-se que os valores de erro estão ligeiramente acima em comparação com a maioria dos modelos previamente estudados. O erro percentual absoluto médio atingiu o valor de 29%, indicando que, apesar do algoritmo de redes neuronais ter surpreendido com os resultados atingidos (8.1%), 83% dos algoritmos testados demonstraram desempenho superior em relação aos valores utilizados no orçamento da Toyota.

Tabela 14 - Previsões Toyota - Avaliação

	EQM	REQM	EAM	EPAM
Previsões Toyota	118598,9	344,4	312,8	29.0%

Observando a Figura 34, notou-se que a proximidade face às matrículas reais é maior nas previsões alcançadas pelo algoritmo, do que as previsões realidades pela Toyota. Mais uma vez, isto comprova o sucesso do algoritmo que, não só apresentou uma taxa de erro baixa, como também acompanha melhor a sazonalidade dos dados e conseguiu prever melhor as matrículas.

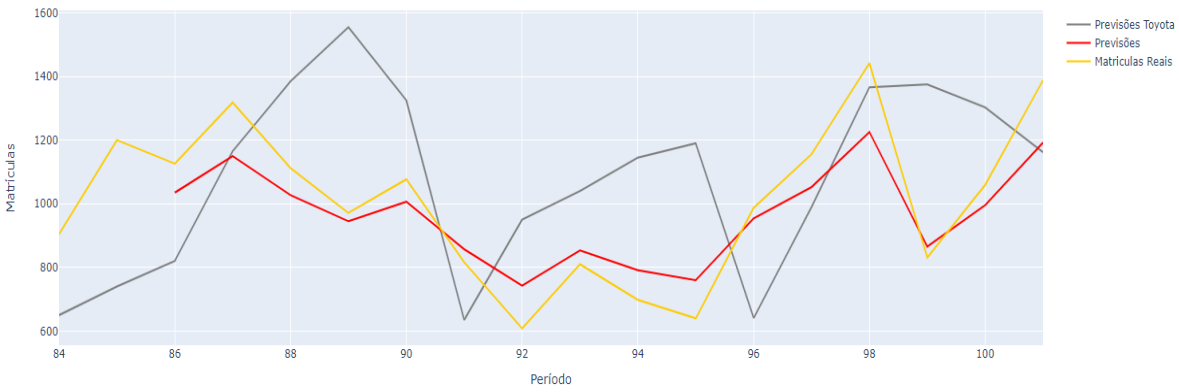


Figura 34 - Gráfico representativo de melhores previsões, matrículas reais e previsões Toyota

5. CONCLUSÃO

Após a modelação das séries com vários tipos de métodos, de avaliar o desempenho de cada método com métricas adequadas e discutir os resultados, este é o capítulo que encerra este documento com uma descrição das principais conclusões finais, assim como, uma descrição das limitações e de investigação futuras.

5.1. Conclusões finais

Neste trabalho aplicaram-se vários métodos de previsão a bases de dados de matrículas e contratos, para prever a quantidade de matrículas futuras da TCAP.

O desenvolvimento do trabalho seguiu a estrutura de uma metodologia de processo padrão (CRISP-DM). Um trabalho deste género pode, desde o início, enfrentar várias dificuldades, como as decisões iniciais sobre qual conjunto de dados escolher, quais os algoritmos a aplicar e quais ferramentas escolher para a execução, que podem afetar o sucesso do trabalho. Assim, a utilização da metodologia referida foi importante para guiar o desenvolvimento do trabalho apresentado neste documento.

Com base nos resultados obtidos, é evidente que a sazonalidade desempenha um papel significativo na precisão das previsões dos dados em análise. Ficou claro que, modelos como Alisamento Exponencial Simples e o ARIMA tiveram um desempenho inferior em comparação com as suas versões sazonais, ou seja, com o Alisamento Exponencial Sazonal e SARIMA. Esta constatação destaca a importância de considerar a sazonalidade quando se trabalha com séries temporais, uma vez que, conforme se percebeu, modelo sazonais conseguem captar melhor tendências e padrões sazonais nos dados.

Paralelemente, o modelo de redes neuronais superou as expectativas ao apresentar previsões mais eficazes, com um valor de erro percentual absoluto médio de 8.1%, quando aplicado à base de dados de matrículas. Este modelo destacou-se perante os outros algoritmos, demonstrando uma notável capacidade de identificar padrões e sazonalidades nos dados. Além dos resultados de erro baixos, a sua linha gráfica de previsão mostrou-se bastante próxima aos valores reais observados, o que justifica a sua utilização.

Ao comparar as previsões dos algoritmos com os resultados alcançados pela TCAP, 83% dos algoritmos testados demonstraram um desempenho superior e justificaram assim a sua utilização. Esses resultados fortalecem a resposta à questão central da dissertação: “Os algoritmos de previsões (estatísticos e de aprendizagem automática) são suficientemente eficazes para realizar previsões fidedignas, justificando a sua aplicação em ambiente empresarial?”

Concluiu-se que, com base nas previsões alcançadas, a aplicação de algoritmos de aprendizagem automática é plenamente justificada e pode resultar em previsões altamente precisas e confiáveis, fornecendo informações valiosas para a tomada de decisão nas empresas.

Estes resultados tem uma importância tremenda no contexto empresarial pois, tem a capacidade de melhorar o planeamento e assim aprimorar as decisões estratégias dentro da TCAP. Com precisões mais precisas, a empresa terá mais condições de projetar planos de vendas futuros, otimizar operações e responder com agilidade às flutuações do mercado.

Em retrospetiva, ao avaliar os resultados deste trabalho em relação aos critérios de sucesso estabelecidos inicialmente, pode-se, assim, afirmar que os objetivos foram alcançados. Os critérios de sucesso, que incluíam a obtenção de modelos de previsão eficazes com baixa percentagem de erro e a obtenção de previsões mais eficazes do que a TCAP, foram amplamente atendidos.

Além disso, é importante perceber que os resultados e as conclusões atingidas com este trabalho não se limitam ao contexto específico da TCAP. Os modelos de aprendizagem automática têm um enorme potencial de serem aplicados noutras empresas do setor automóvel e afins. A capacidade de prever com eficácia e capturar padrões sazonais pode oferecer uma vantagem competitiva significativa. Portanto, os resultados não só respondem aos nossos objetivos, mas também podem representar soluções bastante significativas para outras empresas que procuram melhorar a tomada de decisão.

5.2. Limitações e investigação futura

Embora o trabalho seja relevante e útil, é importante reconhecer as suas limitações e apontar as direções para futuros trabalhos. Uma das principais limitações notáveis desde estudo é a seleção dos algoritmos de previsão pois, existem ainda inúmeros algoritmos que podem ser explorados e aplicados aos dados em estudo. Explorar uma maior quantidade e variedade de algoritmos pode enriquecer o conjunto de resultados e fornecer informações importantes para o desenvolvimento do trabalho.

Além disso, a base de dados em estudo pode ser melhorada e expandida. Ao aumentar a quantidade de dados disponível para estudo, os algoritmos irão conseguir recolher melhor padrões e tendências dos dados históricos e assim fornecer previsões com maior eficácia.

Por fim, é importante realizar uma exploração minuciosa dos parâmetros dos algoritmos ARIMA, SARIMA e redes neuronais para melhorar, ainda mais, o desempenho das previsões. Nem sempre é fácil ajustar os parâmetros ao conjunto de dados a explorar, contudo um ajuste preciso e específico poderá ajudar os algoritmos a obter previsões ainda mais precisas e eficazes.

REFERÊNCIAS BIBLIOGRÁFICAS

Andueza, A., Arco-Osuna, M. D., Fornés, B., González-Crespo, R., & Martín-Álvarez, J. M. (2023). Using the Statistical Machine Learning Models ARIMA and SARIMA to Measure the Impact of Covid-19 on Official Provincial Sales of Cigarettes in Spain, p. 15.

Ansuj, A. P., Camargo, M. E., Radharamanan, R., & Petry, D. G. (1996). *SALES FORECASTING USING TIME SERIES AND NEURAL NETWORKS.*

Botchkarev, A. (2018). *Performance Metrics (Error Measures) in Machine Learning Regression, Forecasting and Prognostics: Properties and Typology, p. 37.*

Botchkarev, A. (2018). *Evaluating performance of regression machine learning models using multiple error metrics in Azure Machine Learning Studio, p. 16.*

Cândido, A. C., Bertotti, P. S., & Bedin, J. (2017). *O potencial das ferramentas atuais de gestão & negócios aplicas às unidades de informação, p. 18.*

Desai, R., Mohammed, A. S., Gurram, P., Srikanth, S., Vyas, A., Katukuri, N., . . . Sachdeva, R. (2023). *Predicting Risk of Cardiac Arrest in Young Asian Americans: Insights from an Artificial Neural Network Analysis of the Nationwide Cohort.*

Dindarloo, S., Hower, J. C., Bagherieh, A., & Trimble, A. S. (2016). *Fundamental evaluation of petrographic effects on coal grindability by seasonal autoregressive integrated moving average (SARIMA), p. 6.*

Duque, J., Silvas, F., & Godinho, A. (2022). *Data Mining applied to Knowledge Management, p. 7.*

Fattah, J., Ezzine, L., Aman, Z., Moussami, H. E., & Lachhab, A. (2018). *Forecasting of demand using ARIMA model, p. 9.*

Fleurke, S. (2017). *Forecasting Automobile Sales using an Ensemble of Methods.*

Fomby, T. B. (2008). *Exponential Smoothing Models, p. 23.*

Fortuny, E., Martens, D., & Provost, F. (2013). *Predictive modeling with big data: Is bigger really better? Big Data.*

Gail, M., Krickeberg, K., Samet, J., Tsiatis, A., & Wong, W. (2009). *Clinical Prediction Models.*

Giering, M. (2008). *Retail Sales Prediction and Item Recommendations Using Customer Demographics at Store Level, p. 6.*

Goh, C., & Law, R. (2002). *Modeling and forecasting tourism demand for arrivals with stochastic nonstationary seasonality and intervention.*

Gonçalves, C., Ferreira, D., Neto, C., Abelha, A., & Machado, J. (2020). *Prediction of Mental Illness Associated with Unemployment Using Data Mining, p. 6.*

Ho, S. L., & XIE, M. (1998). *THE USE OF ARIMA MODELS FOR RELIABILITY FORECASTING AND ANALYSIS, p. 4.*

International Journal of Science and Research. (2020). *Machine Learning Algorithms-A Review.*

JULA, P., HOUSHYAR, A., SEVERANCE, F. L., & SAWHNEY, A. (1996). APPLICATION OF ARTIFICIAL NEURAL NETWORKS IN INTERACTIVE SIMULATION, p. 4.

Kato, T. (2020). *Demand Prediction in the Automobile Industry Independent of Big Data. Annals of Data Science.*

Klee, S., Janson, A., & Leimeister, J. M. (2023). *Automotive Manufacturers and Their Stumble from one Supply Crisis to Another: Procurement Departments Could be the Game Changer by Using Data Analytics, but….*

Kohli, S., & Godwin, G. T. (2020). *Sales Prediction Using Linear and KNN Regression.*

Kon, S. C., & Turner, L. W. (2005). *Neural network forecasting of tourism demand.*

Konrad-Adenauer-Stiftung's Regional Economic Programme Asia (SOPAS). (2021). *Age of Ferment: Developments in Asian-European Trade Relations.*

Loureiro, A. L., Miguéis, V. L., & Silva, L. F. (2018). *Exploring the use of deep neural networks for sales forecasting in fashion retail, p. 13.*

Mahesh, B. (2020). *Machine Learning Algorithms - A Review, p. 6.*

Mamun, A., Sohel, M., Mohammad, N., Sunny, S., Dipta, D., & Hossain, E. (2020). *A Comprehensive Review of the Load Forecasting Techniques Using Single and Hybrid Predictive Models. In IEEE Access. Institute of Electrical and Electronics Engineers Inc.*

Mosavi, A., Ozturk, P., & Chau, K.-w. (2018). *Flood Prediction Using Machine Learning Models: Literature Review, p. 40.*

Parsonson, B. S., Baer, D. M. (1986). *The Graphic Analysis of Data.*

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion , B., Grisel, O., . . . Duchesnay , E. (2011). *Scikit-learn: Machine Learning in Python, p. 6*.

Plotnikova, V., Dumas, M., & Milani, F. P. (2022). *Applying the CRISP-DM data mining process in the financial services industry: Elicitation of adaptation requirements, p. 17.*

Qu, F., Wang, Y. T., Hou, W. H., Zhou, X. Y., Wang, X. K., Li, J. B., & Wang, J. Q. (2022). *Forecasting of Automobile Sales Based on Support Vector Regression Optimized by the Grey Wolf Optimizer Algorithm. Mathematics.*

Ramani, V., Ghosh, D., & Sodhi, M. M. S. (2022). *Understanding systemic disruption from the Covid-19-induced semiconductor shortage for the auto industry. Omega (United Kingdom).*

Saltz, J. S. (2021). *CRISP-DM for Data Science: Strengths, Weaknesses and Potential Next Steps*

Schröer, C., Kruse, F., & Gómez, J. (2021). *A systematic literature review on applying CRISP-DM process model. Procedia Computer Science, 181.*

Tax, N., Verenich, I., La Rosa, M., & Dumas, M. (2017). *Predictive Business Process Monitoring with LSTM Neural Networks.*

Tsai, C. (2016). *Recognizing Handwritten Japanese Characters Using Deep Convolutional Neural Networks, p. 7.*

Wirth, R., & Hipp, J. (1968). *CRISP-DM: Towards a Standard Process Model for Data Mining, p. 11.*

Zhao, D., Zhang, R., Zhang, H., & He, S. (2022). *Prediction of global omicron pandemic using ARIMA, MLR, and Prophet models. Scientific Reports, 12(1).*

APÊNDICE A

```
!pip3 install virtualenv #Instalar a biblioteca virtualenv para
chamar o método abaixo
!virtualenv pyenv #Criar um ambiente virtual pyenv
!source /content/pyenv/bin/activate #Ativar o ambiente virtual e
isolar o atual
!pip3 -q install plotly --upgrade #Biblioteca utilizada para criar
visualizações interativas
!pip3 -q install yellowbrick #Biblioteca que fornece visualizações de
análises de modelo
!pip3 install pmdarima #Biblioteca utilizada para ajustar
automáticamente os modelos ARIMA e SARIMA.
import pandas as pd #Importa a biblioteca pandas para a tratamento de
dados.
import numpy as np #Importa a biblioteca numpy para operações
matemáticas e numéricas eficientes.
import seaborn as sns #Importa a biblioteca seaborn para visualização
de dados estatísticos.
import matplotlib.pyplot as plt #Importa a biblioteca matplotlib para
criar gráficos.
import plotly.express as px #Importa a biblioteca plotly express para
visualização interativa de dados.
import sqlite3  #Importa a biblioteca sqlite3 para trabalhar com
bancos de dados SQLite.
import plotly.graph_objects as go #Importa a biblioteca plotly
graph_objects para criar gráficos mais complexos.
from statsmodels.tsa.holtwinters import SimpleExpSmoothing #Importa o
modelo de alisamento exponencial simples da statsmodels.
from statsmodels.tsa.holtwinters import
ExponentialSmoothing  #Importa o modelo de alisamento exponencial da
statsmodels.
from sklearn.metrics import mean_squared_error  #Importa a função
para cálculo do EQM.
from sklearn.metrics import mean_absolute_error  #Importa a função
para cálculo do EAM.
from sklearn.preprocessing import MinMaxScaler  #Importa a classe
MinMaxScaler para normalização de dados.
from keras.models import Sequential  #Importa a classe Sequential da
biblioteca Keras para construir de modelos sequenciais.
from keras.layers import Dense, LSTM  #Importa as classes Dense e
LSTM da biblioteca Keras para o algoritmos de redes neuronais.
import statsmodels.api as sm  #Importa a API da statsmodels para
análise estatística e modelagem.
import pmdarima as pm  #Importa a biblioteca pmdarima para modelagem
de séries temporais com ARIMA.
from sklearn.neighbors import NearestNeighbors  # Importa a classe
NearestNeighbors da scikit-learn para vizinhos mais próximos.
```

```
from sklearn.model_selection import TimeSeriesSplit  #Importa a
classe TimeSeriesSplit da scikit-learn para divisão de séries
temporais em conjuntos de treino e teste.
from google.colab import drive

drive.mount('/content/drive') #Importar as bases de dados

base_matriculas =
pd.read_excel('/content/Histórico_de_Matriculas.xlsx') #Ler a base de
dados de matrículas
base_matriculas #Analisar a base de dados de matrículas

contagem_valores_nulos_matriculas = base_matriculas.isnull().sum()
#Contar os dados em branco presentes na base de dados de matriculas
print(contagem_valores_nulos_matriculas) #Observar os resultados
base_matriculas['DT_MATRICULA'] =
pd.to_datetime(base_matriculas['DT_MATRICULA'], format='%Y%m%d')
#Separar a coluna em 3. Ano, mês e dia.
base_matriculas['ano'] = base_matriculas['DT_MATRICULA'].dt.year
#Retirar o ano
base_matriculas['mes'] = base_matriculas['DT_MATRICULA'].dt.mont
#Retirar o mes
base_matriculas['dia'] = base_matriculas['DT_MATRICULA'].dt.day
#Retirar o dia
#Criar a coluna Ano_Mês
base_matriculas = pd.DataFrame(base_matriculas)
base_matriculas ['DT_MATRICULA'] = pd.to_datetime(base_matriculas
['DT_MATRICULA'])
base_matriculas ['Ano_Mês'] = base_matriculas
['DT_MATRICULA'].dt.strftime('%Y %b')
base_matriculas  = base_matriculas .sort_values(by=['Ano_Mês'])
base_matriculas ['Ano_Mês'] = pd.to_datetime(base_matriculas
['Ano_Mês'])
base_matriculas
Contagem_Matriculas_Tabela_análise =
base_matriculas.groupby(['Ano_Mês']).count() #Agrupar o número de
linhas da base de dados pela coluna Ano_Mês
Contagem_Matriculas_análise =
Contagem_Matriculas_Tabela_análise['COD_CONCESSAO'] #Contar as
matrículas para cada mês
Contagem_Matriculas_análise #Visualizar os resultados obtidos
tabela_matriculas_análise = Contagem_Matriculas_análise.reset_index()
#Definir um índice para cada linha da tabela anterior
tabela_matriculas_análise =
tabela_matriculas_análise.rename(columns={'COD_CONCESSAO':
'Matriculas', 'Ano_Mês': 'Data'}) #Alterar o nome das colunas
tabela_matriculas_análise #Visualizar os resultados obtidos
```

```
Contagem_Matriculas_Tabela =
base_matriculas.groupby(['Ano_Mês','GAMA']).count() #Agrupar o número
de linhas por Ano_Mês e GAMA
Contagem_Matriculas = Contagem_Matriculas_Tabela['COD_CONCESSAO']
#Contar as matrículas para cada Ano_Mês separados por cada GAMA
tabela_matriculas = Contagem_Matriculas.reset_index() #Definir um
índice para cada linha da tabela anterior
tabela_matriculas =
tabela_matriculas.rename(columns={'COD_CONCESSAO': 'Matriculas',
'Ano_Mês': 'Data'}) #Alterar o nome das colunas
tabela_matriculas #Visualizar os resultados obtidos

base_contratos = pd.read_excel('/content/Base de dados
Contratos.xlsx') #Ler a base de dados de contratos
base_contratos #Analisar a base de dados de contratos
contagem_valores_nulos = base_contratos.isnull().sum() #Contar os
dados em branco presentes na base de dados de contratos
print(contagem_valores_nulos) #Observar os resultados
#Passar Mês para N° de Mês
meses = {'Jan': 1,'Fev': 2,'Mar': 3,'Abr': 4,'Mai': 5,'Jun': 6,'Jul':
7,'Ago': 8,'Set': 9,'Out': 10,'Nov': 11,'Dez': 12}
base_contratos['N° Mês'] = base_contratos['Mês'].map(meses)
base_contratos
#Utilizar Coluna Ano, N°Mês e primeiro dia do Mês para criar uma
coluna e formato data
base_contratos['Ano_Mês'] = base_contratos['Ano'].astype(str)  + '-
'  + base_contratos['N° Mês'].astype(str) + '-01'
base_contratos['Ano_Mês'] = pd.to_datetime(base_contratos['Ano_Mês'])
soma_contratos_análise=base_contratos.groupby(['Ano_Mês'])['CC'].sum(
) #Somar o n° de contratos de cada mês
soma_contratos_análise #Observar o resultado
tabela_contratos_análise = soma_contratos_análise.reset_index()
#Definir um índice para cada linha da tabela anterior
tabela_contratos_análise =
tabela_contratos_análise.rename(columns={'CC': 'Contratos',
'Ano_Mês': 'Data'}) #Alterar o nome das colunas
tabela_contratos_análise #Visualizar os resultados obtidos
soma_contratos=base_contratos.groupby(['Ano_Mês','Gama'])['CC'].sum()
#Agrupar o número de linhas por Ano_Mês e GAMA
#Somar as matrículas para cada Ano_Mês separados por cada GAMA
tabela_contratos = soma_contratos.reset_index() #Definir um índice
para cada linha da tabela anterior
tabela_contratos = tabela_contratos.rename(columns={'CC':
'Contratos', 'Ano_Mês': 'Data', 'Gama':'GAMA'}) #Alterar o nome das
colunas
tabela_contratos #Visualizar os resultados obtidos
```

```
tabela_junta= pd.merge(tabela_contratos, tabela_matriculas,
on=['Data','GAMA'], how='inner') #Juntar as tabelas matrículas e
contratos com Data e GAMA
tabela_junta #Visualizar resultado

#Desenvolver estatisticas de Matriculas e Contratos por ano
estatisticas_contratos =
tabela_contratos_análise.groupby(tabela_contratos_análise['Data'].dt.
year)['Contratos'].describe().round()
estatisticas_matriculas =
tabela_matriculas_análise.groupby(tabela_matriculas_análise['Data'].d
t.year)['Matriculas'].describe().round()
print("Estatísticas de Contratos por ano:")
print(estatisticas_contratos)
print("Estatísticas das Matrículas por ano:")
print(estatisticas_matriculas)

#Desenvolver estatisticas de Matriculas e Contratos por GAMA
estatisticas_contratos =
tabela_junta.groupby("GAMA")['Contratos'].describe().round()
estatisticas_matriculas =
tabela_junta.groupby("GAMA")['Matriculas'].describe().round()
print("Estatísticas de Contratos por modelo:")
print(estatisticas_contratos)
print("Estatísticas das Matrículas por modelo:")
print(estatisticas_matriculas)

#Desenvolver gráfico de matrículas e contratos ao longo dos meses e
anos
fig = go.Figure()

dados_agrupados = tabela_junta.groupby(tabela_junta['Data']).sum()

# Adicionar linhas no gráfico
fig.add_trace(go.Scatter(x=dados_agrupados.index,
y=dados_agrupados['Contratos'], name='Contratos'))
fig.add_trace(go.Scatter(x=dados_agrupados.index,
y=dados_agrupados['Matriculas'], name='Matrículas'))

# Configurar o layout do gráfico
fig.update_layout(
    title='Gráfico de Séries Temporais por mês',
    xaxis=dict(title='Data'),
    yaxis=dict(title='Quantidade'),
    showlegend=True,
    legend=dict(x=0, y=1),
    height=500,
    width=1000
)
```

```
#Desenvolver um gráfico de barras de contratos por modelo
plt.figure(figsize=(6, 10))
sns.countplot(y =
base_contratos['Gama'],order=base_contratos['Gama'].value_counts().in
dex)
plt.xlabel('Contratos')
plt.ylabel('Modelos')

#Desenvolver um gráfico de barras de matrículas por modelo
plt.figure(figsize=(6, 10))
sns.countplot(y =
base_matriculas['GAMA'],order=base_matriculas['GAMA'].value_counts().
index)
plt.xlabel('Matrículas')
plt.ylabel('Modelos')

#Criar Box Plot de Matrículas por ano
fig = go.Figure()

# Criar o box plot
fig.add_trace(go.Box(
    x=tabela_matriculas_análise['Data'].dt.year,
    y=tabela_matriculas_análise['Matriculas'],
    name='Matrículas',
    marker=dict(color='blue'),
    boxpoints='all'
))

# Configurar o layout do gráfico
fig.update_layout(
    title='Box Plot de Matrículas por Ano',
    xaxis=dict(title='Ano'),
    yaxis=dict(title='Matrículas'),
    showlegend=False
)

# Exibir o gráfico
fig.show()
#Criar Box Plot de Contratos por ano
fig = go.Figure()


# Criar o box plot
fig.add_trace(go.Box(
    x=tabela_contratos_análise['Data'].dt.year,
    y=tabela_contratos_análise['Contratos'],
    name='Contratos',
    marker=dict(color='blue'),
```

```
    boxpoints='all'
))

# Configurar o layout do gráfico
fig.update_layout(
    title='Box Plot de Contratos por Ano',
    xaxis=dict(title='Ano'),
    yaxis=dict(title='Contratos'),
    showlegend=False
)

# Exibir o gráfico
fig.show()

#Criar Repartição dos dados (Treino/Teste) para serem considerados
nos algoritmos
pd_matriculas = pd.DataFrame(tabela_matriculas_análise)
pd_contratos = pd.DataFrame(tabela_contratos_análise)

# Definir mascaras de treino e teste para matrículas
mascara_matriculas_treino = (pd_matriculas['Data'] <= '2021-12-01')
mascara_matriculas_test = (pd_matriculas['Data'] > '2021-12-01')
serie_treino_matriculas =
pd_matriculas.loc[mascara_matriculas_treino, 'Matriculas']
serie_test_matriculas = pd_matriculas.loc[mascara_matriculas_test,
'Matriculas']

# Definir mascaras de treino e teste para contratos
mascara_contratos_treino = (pd_contratos['Data'] <= '2021-12-01')
mascara_contratos_test = (pd_contratos['Data'] > '2021-12-01')
serie_treino_contratos =
pd_contratos.loc[mascara_contratos_treino,'Contratos']
serie_test_contratos =
pd_contratos.loc[mascara_contratos_test,'Contratos']

#Validar se bases têm o mesmo tamanho de treino e teste
print(serie_treino_matriculas.shape[0])
print(serie_test_matriculas.shape[0])
print(serie_treino_contratos.shape[0])
print(serie_test_contratos.shape[0])
if serie_treino_matriculas.shape[0] !=
serie_treino_contratos.shape[0]:
  print('Dados de treino nao coincidem!')
if serie_test_matriculas.shape[0] != serie_test_contratos.shape[0]:
  print('Dados de teste nao coincidem!')
  print(serie_test_contratos)
  print(serie_treino_contratos)
```

```
#Aplicar o algoritmo de alisamento exponencial simples na base de
dados de matrículas
# Selecionar coluna de interesse
serie_treino_matriculas =
pd_matriculas.loc[mascara_matriculas_treino,'Matriculas']

# Aplicar o algoritmo de alisamento exponencial simples
modelo = SimpleExpSmoothing(serie_treino_matriculas)
modelo_ajustado = modelo.fit()

# Realizar previsões
previsoes1 =
modelo_ajustado.predict(start=len(serie_treino_matriculas),
end=len(serie_treino_matriculas)+ 17)

# Criar gráfico
fig = go.Figure()

# Adicionar dados de treino ao gráfico
fig.add_trace(go.Scatter(
    x=serie_treino_matriculas.index,
    y=serie_treino_matriculas,
    mode='lines',
    name='Dados de Treino'
))

# Adicionar previsões ao gráfico
fig.add_trace(go.Scatter(
    x=previsoes1.index,
    y=previsoes1,
    mode='lines',
    name='Previsões'
))

# Configurar layout do gráfico
fig.update_layout(
    xaxis_title='Período',
    yaxis_title='Matrículas',
    title='Previsão de Matrículas usando Alisamento Exponencial
Simples',
    legend=dict(
        traceorder='normal'
    )
)

# Exibir gráfico
fig.show()
#Calcular EQM
```

```
serie_test_matriculas =
pd_matriculas.loc[mascara_matriculas_test,'Matriculas']
eqm = mean_squared_error(serie_test_matriculas, previsoes1)

print("Erro Quadrático Médio (EQM):", eqm)

#Calcular REQM
serie_test_matriculas = pd_matriculas.loc[mascara_matriculas_test,
'Matriculas']
eqm = mean_squared_error(serie_test_matriculas, previsoes1)
reqm = np.sqrt(eqm)

print("Raiz do Erro Quadrático Médio (REQM):", reqm)

# Calcular EAM
serie_test_matriculas = pd_matriculas.loc[mascara_matriculas_test,
'Matriculas']
eam = mean_absolute_error(serie_test_matriculas, previsoes1)

print("Erro Absoluto Médio (EAM):", eam)

#Calcular EPAM
erro_absoluto = abs(serie_test_matriculas - previsoes1)
epam = (erro_absoluto.mean() / previsoes1.mean()) * 100

print("Erro Percentual Absoluto Médio (EPAM):", epam)

#Aplicar o algoritmo de alisamento exponencial sazonal na base de
dados de matrículas
# Criar a série temporal a partir da coluna "Matriculas"
serie_treino_matriculas =
pd_matriculas.loc[mascara_matriculas_treino,'Matriculas']

# Aplicar o algoritmo de alisamento exponencial sazonal
modelo = ExponentialSmoothing(serie_treino_matriculas,
seasonal='add', seasonal_periods=12)
modelo_ajustado = modelo.fit()

# Realizar previsões
previsoes2 =
modelo_ajustado.predict(start=len(serie_treino_matriculas),
end=len(serie_treino_matriculas)+17)

# Criar figura
fig = go.Figure()

# Adicionar dados de treino ao gráfico
fig.add_trace(go.Scatter(
    x=serie_treino_matriculas.index,
```

```
    y=serie_treino_matriculas,
    mode='lines',
    name='Dados de Treino'
))

# Adicionar previsões ao gráfico
fig.add_trace(go.Scatter(
    x=previsoes2.index,
    y=previsoes2,
    mode='lines',
    name='Previsões'
))

# Configurar layout do gráfico
fig.update_layout(
    xaxis_title='Período',
    yaxis_title='Matrículas',
    title='Previsão de Matrículas usando Alisamento Exponencial
Sazonal',
    legend=dict(
        traceorder='normal'
    )
)

# Exibir gráfico
fig.show()

#Calcular EQM
serie_test_matriculas =
pd_matriculas.loc[mascara_matriculas_test,'Matriculas']
eqm = mean_squared_error(serie_test_matriculas, previsoes2)

print("Erro Quadrático Médio (EQM):", eqm)

#Calcular REQM
serie_test_matriculas = pd_matriculas.loc[mascara_matriculas_test,
'Matriculas']
eqm = mean_squared_error(serie_test_matriculas, previsoes2)
reqm = np.sqrt(eqm)

print("Raiz do Erro Quadrático Médio (REQM):", reqm)

#Calcular EAM
serie_test_matriculas = pd_matriculas.loc[mascara_matriculas_test,
'Matriculas']
eam = mean_absolute_error(serie_test_matriculas, previsoes2)

print("Erro Absoluto Médio (EAM):", eam)
```

```
#Calcular EPAM
erro_absoluto = abs(serie_test_matriculas - previsoes2)
epam = (erro_absoluto.mean() / previsoes2.mean()) * 100

print("Erro Percentual Absoluto Médio (EPAM):", epam)


#Aplicar algoritmo de redes neuronais na base de dados de matrículas

from sklearn.model_selection import train_test_split

# Converter em pandas DataFrame
pd_matriculas = pd.DataFrame(tabela_matriculas_análise)

# Normalizar os dados de entrada
scaler = MinMaxScaler(feature_range=(0, 1))
serie_matriculas_normalizada =
scaler.fit_transform(pd_matriculas['Matriculas'].values.reshape(-1,
1))

# Preparar os dados de treino, validação e teste
X_train, X_temp, y_train, y_temp =
train_test_split(serie_matriculas_normalizada,
serie_matriculas_normalizada, test_size=0.3, shuffle=False)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp,
test_size=0.5, shuffle=False)

# Função para criar o conjunto de dados de entrada e saída
def create_dataset(dataset, janela_tempo=1):
    data_X, data_Y = [], []
    for i in range(len(dataset) - janela_tempo +1  ): # +1 para
complementar a posição 0 da matriz
        data_X.append(dataset[i:i+janela_tempo])
        data_Y.append(dataset[i+janela_tempo-1])
    return np.array(data_X), np.array(data_Y)

janela_tempo = 1

# Criar o conjunto de dados de treino, validação e teste
X_train, y_train = create_dataset(X_train, janela_tempo)
X_val, y_val = create_dataset(X_val, janela_tempo)
X_test, y_test = create_dataset(X_test, janela_tempo)

# Criar o modelo de redes neuronais
modelo = Sequential()
modelo.add(LSTM(50, input_shape=(janela_tempo, 1)))
modelo.add(Dense(1))
modelo.compile(loss='mean_squared_error', optimizer='adam')
```

```
# Função para exibir o gráfico de treino e validação
def plot_loss(history):
    plt.figure(figsize=(8, 6))
    plt.plot(history.history['loss'], label='Train Loss')
    plt.plot(history.history['val_loss'], label='Val Loss')
    plt.xlabel('Época')
    plt.ylabel('Loss')
    plt.title('Gráfico de Train Loss e Val Loss')
    plt.legend()
    plt.show()

# Treinar o modelo com o conjunto de treino
history = modelo.fit(X_train, y_train, epochs=100, batch_size=32,
validation_data=(X_val, y_val), verbose=1)

# Realizar previsões para o conjunto de teste
previsoes_redes1_normalizadas = modelo.predict(X_test)
previsoes_redes1 =
scaler.inverse_transform(previsoes_redes1_normalizadas)

# Exibir gráfico com as previsões
fig = go.Figure()

# Adicionar dados de treino ao gráfico
fig.add_trace(go.Scatter(
    x=pd_matriculas.index[:len(y_train)],
    y=scaler.inverse_transform(y_train).flatten(),
    mode='lines',
    name='Dados de Treino'
))

# Adicionar dados de validação ao gráfico
fig.add_trace(go.Scatter(
    x=pd_matriculas.index[len(y_train):len(y_train)+len(y_val)],
    y=scaler.inverse_transform(y_val).flatten(),
    mode='lines',
    name='Dados de Validação',
    line=dict(color='green')
))

# Adicionar previsões ao gráfico
fig.add_trace(go.Scatter(
    x=pd_matriculas.index[len(y_train)+len(y_val): len(y_train) +
len(y_val) + len(y_test)],
    y=previsoes_redes1.flatten(),
    mode='lines',
    name='Previsões',
    line=dict(color='red')
))
```

```
# Adicionar Matriculas reais ao gráfico
fig.add_trace(go.Scatter(
    x=serie_test_matriculas.index,
    y=serie_test_matriculas,
    mode='lines',
    name='Matriculas Reais',
    line=dict(color='#FFCC00')
))

# Configurar layout do gráfico
fig.update_layout(
    xaxis_title='Período',
    yaxis_title='Matrículas',
    title='Previsão de Matrículas usando Redes Neuronais',
    legend=dict(
        traceorder='normal'
    )
)

# Exibir gráfico
fig.show()
plot_loss(history)

# Calcular o EQM
eqm = mean_squared_error(y_test_real, previsoes_redes1)

print("Erro Quadrático Médio (EQM):", eqm)


# Calcular o REQM
eqm = mean_squared_error(y_test_real, previsoes_redes1)
reqm = np.sqrt(eqm)

print("Raiz do Erro Quadrático Médio (REQM):", reqm)


#Calcular o EAM
eam = mean_absolute_error(y_test_real, previsoes_redes1)
print("EAM:", eam)


#Calcular o EPAM
# Realizar previsões para os dados de teste
previsoes_redes1_normalizadas = modelo.predict(X_test)
previsoes_redes1 =
scaler.inverse_transform(previsoes_redes1_normalizadas)

# Desfazer a normalização dos dados de teste para obter os valores
reais
```

```
y_test_real = scaler.inverse_transform(y_test)

# Calcular o erro absoluto
erro_absoluto = abs(y_test_real - previsoes_redes1)

# Calcular o EPAM
epam = (erro_absoluto.mean() / y_test_real.mean()) * 100

# Exibir o valor do EPAM
print("Erro Percentual Absoluto Médio (EPAM):", epam)

#Aplicar algoritmo ARIMA na base de dados de matrículas
# Criar a série temporal a partir da coluna "Matriculas"
serie_treino_matriculas =
pd_matriculas.loc[mascara_matriculas_treino, 'Matriculas']

# Identificar os parâmetros do modelo ARIMA
order = (21, 1, 1)  # Parâmetros (p, d, q)

# Criar e ajustar o modelo ARIMA aos dados de treino
modelo_arima = sm.tsa.ARIMA(serie_treino_matriculas, order=order)
modelo_arima_ajustado = modelo_arima.fit()

# Realizar previsões
previsoes_arima1 =
modelo_arima_ajustado.predict(start=len(serie_treino_matriculas),
end=len(serie_treino_matriculas) + 17)

# Criar figura
fig = go.Figure()

# Adicionar dados de treino ao gráfico
fig.add_trace(go.Scatter(
    x=serie_treino_matriculas.index,
    y=serie_treino_matriculas,
    mode='lines',
    name='Dados de Treino'
))

# Adicionar previsões do ARIMA ao gráfico
fig.add_trace(go.Scatter(
    x=previsoes_arima1.index,
    y=previsoes_arima1,
    mode='lines',
    name='Previsões ARIMA'
))

# Configurar layout do gráfico
fig.update_layout(
```

```
    xaxis_title='Período',
    yaxis_title='Matrículas',
    title='Previsão de Matrículas usando ARIMA',
    legend=dict(
        traceorder='normal'
    )
)

# Exibir gráfico
fig.show()



# Calcular EQM
serie_test_matriculas =
pd_matriculas.loc[mascara_matriculas_test,'Matriculas']
eqm = mean_squared_error(serie_test_matriculas, previsoes_arima1)

print("Erro Quadrático Médio (EQM):", eqm)


# Calcular REQM
serie_test_matriculas = pd_matriculas.loc[mascara_matriculas_test,
'Matriculas']
eqm = mean_squared_error(serie_test_matriculas, previsoes_arima1)
reqm = np.sqrt(eqm)

print("Raiz do Erro Quadrático Médio (REQM):", reqm)


# Calcular EAM
serie_test_matriculas = pd_matriculas.loc[mascara_matriculas_test,
'Matriculas']
eam = mean_absolute_error(serie_test_matriculas, previsoes_arima1)

print("Erro Absoluto Médio (EAM):", eam)


# Calcular EPAM
erro_absoluto = abs(serie_test_matriculas - previsoes_arima1)
epam = (erro_absoluto.mean() / previsoes_arima1.mean()) * 100

print("Erro Percentual Absoluto Médio (EPAM):", epam)

#Aplicar algoritmo Arima com AutoARIMA na base de dados matrículas
# Identificar os melhores parâmetros usando o AutoARIMA
modelo_arima = pm.auto_arima(serie_treino_matriculas, seasonal=False,
suppress_warnings=True)
order2 = modelo_arima.order
```

```
order2

# Criar a série temporal a partir da coluna "Matriculas"
serie_treino_matriculas =
pd_matriculas.loc[mascara_matriculas_treino, 'Matriculas']

# Utilizar os parâmetros AutoARIMA
order = order2  # Parâmetros (p, d, q)

# Criar e ajustar o modelo ARIMA aos dados de treino
modelo_arima = sm.tsa.ARIMA(serie_treino_matriculas, order=order)
modelo_arima_ajustado = modelo_arima.fit()

# Realizar previsões
previsoes_arima2 =
modelo_arima_ajustado.predict(start=len(serie_treino_matriculas),
end=len(serie_treino_matriculas) + 17)

# Criar figura
fig = go.Figure()

# Adicionar dados de treino ao gráfico
fig.add_trace(go.Scatter(
    x=serie_treino_matriculas.index,
    y=serie_treino_matriculas,
    mode='lines',
    name='Dados de Treino'
))

# Adicionar previsões do ARIMA ao gráfico
fig.add_trace(go.Scatter(
    x=previsoes_arima2.index,
    y=previsoes_arima2,
    mode='lines',
    name='Previsões ARIMA'
))

# Configurar layout do gráfico
fig.update_layout(
    xaxis_title='Período',
    yaxis_title='Matrículas',
    title='Previsão de Matrículas usando ARIMA e AutoARIMA',
    legend=dict(
        traceorder='normal'
    )
)

# Exibir gráfico
fig.show()
```

```
# Calcular EQM
serie_test_matriculas =
pd_matriculas.loc[mascara_matriculas_test,'Matriculas']
eqm = mean_squared_error(serie_test_matriculas, previsoes_arima2)

print("Erro Quadrático Médio (EQM):", eqm)

# Calcular REQM
serie_test_matriculas = pd_matriculas.loc[mascara_matriculas_test,
'Matriculas']
eqm = mean_squared_error(serie_test_matriculas, previsoes_arima2)
reqm = np.sqrt(eqm)

print("Raiz do Erro Quadrático Médio (REQM):", reqm)

#Calcular EAM
serie_test_matriculas = pd_matriculas.loc[mascara_matriculas_test,
'Matriculas']
eam = mean_absolute_error(serie_test_matriculas, previsoes_arima2)

print("Erro Absoluto Médio (EAM):", eam)

# Calcular EPAM
erro_absoluto = abs(serie_test_matriculas - previsoes_arima2)
epam = (erro_absoluto.mean() / previsoes_arima2.mean()) * 100

print("Erro Percentual Absoluto Médio (EPAM):", epam)

#Aplicar o algoritmo SARIMA na base de dados de matrículas
# Criar a série temporal a partir da coluna "Matriculas"
serie_treino_matriculas =
pd_matriculas.loc[mascara_matriculas_treino, 'Matriculas']

# Identificar os parâmetros do modelo SARIMA
order = (21, 1, 1)  # Parâmetros (p, d, q)
seasonal_order = (0, 1, 1, 12)  # Parâmetros (P, D, Q, S)

# Criar e ajustar o modelo SARIMA aos dados de treino
modelo_sarima = sm.tsa.SARIMAX(serie_treino_matriculas, order=order,
seasonal_order=seasonal_order)
modelo_sarima_ajustado = modelo_sarima.fit()

# Realizar previsões
previsoes_sarima1 =
modelo_sarima_ajustado.predict(start=len(serie_treino_matriculas),
end=len(serie_treino_matriculas) + 17)

# Criar figura
```

```
fig = go.Figure()

# Adicionar dados de treino ao gráfico
fig.add_trace(go.Scatter(
    x=serie_treino_matriculas.index,
    y=serie_treino_matriculas,
    mode='lines',
    name='Dados de Treino'
))

# Adicionar previsões do SARIMA ao gráfico
fig.add_trace(go.Scatter(
    x=previsoes_sarima1.index,
    y=previsoes_sarima1,
    mode='lines',
    name='Previsões SARIMA'
))

# Configurar layout do gráfico
fig.update_layout(
    xaxis_title='Período',
    yaxis_title='Matrículas',
    title='Previsão de Matrículas usando SARIMA',
    legend=dict(
        traceorder='normal'
    )
)

# Exibir gráfico
fig.show()

#Calcular EQM
serie_test_matriculas =
pd_matriculas.loc[mascara_matriculas_test,'Matriculas']
eqm1 = mean_squared_error(serie_test_matriculas, previsoes_sarima1)

print("Erro Quadrático Médio (EQM):", eqm1)

#Calcular REQM
serie_test_matriculas = pd_matriculas.loc[mascara_matriculas_test,
'Matriculas']
eqm = mean_squared_error(serie_test_matriculas, previsoes_sarima1)
reqm = np.sqrt(eqm)

print("Raiz do Erro Quadrático Médio (REQM):", reqm)


#Calcular EAM
```

```
serie_test_matriculas = pd_matriculas.loc[mascara_matriculas_test,
'Matriculas']
eam = mean_absolute_error(serie_test_matriculas, previsoes_sarima1)

print("Erro Absoluto Médio (EAM):", eam)


#Calcular EPAM
erro_absoluto = abs(serie_test_matriculas - previsoes_sarima1)
epam = (erro_absoluto.mean() / previsoes_sarima1.mean()) * 100

print("Erro Percentual Absoluto Médio (EPAM):", epam)


#Calcular algoritmo SARIMA com AutoSARIMA na base de dados de
matriculas
# Identificar os melhores parâmetros utilizando AutoARIMA para SARIMA
modelo_sarima = pm.auto_arima(serie_treino_matriculas,
seasonal=True,m=12, suppress_warnings=True)

order1 = modelo_sarima.order
seasonal_order1 = modelo_sarima.seasonal_order

print("SARIMA Order:", order1)
print("SARIMA Seasonal Order:", seasonal_order1)



# Criar a série temporal a partir da coluna "Matriculas"
serie_treino_matriculas =
pd_matriculas.loc[mascara_matriculas_treino, 'Matriculas']

# Identificar os parâmetros do modelo SARIMA
order = order1  # Parâmetros (p, d, q)
seasonal_order = seasonal_order1 # Parâmetros (P, D, Q, S)

# Criar e ajustar o modelo SARIMA aos dados de treino
modelo_sarima = sm.tsa.SARIMAX(serie_treino_matriculas, order=order,
seasonal_order=seasonal_order)
modelo_sarima_ajustado = modelo_sarima.fit()

# Realizar previsões
previsoes_sarima2 =
modelo_sarima_ajustado.predict(start=len(serie_treino_matriculas),
end=len(serie_treino_matriculas) + 17)

# Criar figura
fig = go.Figure()
```

```
# Adicionar dados de treino ao gráfico
fig.add_trace(go.Scatter(
    x=serie_treino_matriculas.index,
    y=serie_treino_matriculas,
    mode='lines',
    name='Dados de Treino'
))

# Adicionar previsões do SARIMA ao gráfico
fig.add_trace(go.Scatter(
    x=previsoes_sarima2.index,
    y=previsoes_sarima2,
    mode='lines',
    name='Previsões SARIMA'
))

# Configurar layout do gráfico
fig.update_layout(
    xaxis_title='Período',
    yaxis_title='Matrículas',
    title='Previsão de Matrículas usando SARIMA e AutoSARIMA',
    legend=dict(
        traceorder='normal'
    )
)

# Exibir gráfico
fig.show()



# Calcular EQM
serie_test_matriculas =
pd_matriculas.loc[mascara_matriculas_test,'Matriculas']
eqm = mean_squared_error(serie_test_matriculas, previsoes_sarima2)

print("Erro Quadrático Médio (EQM):", eqm)


# Calcular REQM
serie_test_matriculas = pd_matriculas.loc[mascara_matriculas_test,
'Matriculas']
eqm = mean_squared_error(serie_test_matriculas, previsoes_sarima2)
reqm = np.sqrt(eqm)

print("Raiz do Erro Quadrático Médio (REQM):", reqm)


# Calcular EAM
```

```
serie_test_matriculas = pd_matriculas.loc[mascara_matriculas_test,
'Matriculas']
eam = mean_absolute_error(serie_test_matriculas, previsoes_sarima2)

print("Erro Absoluto Médio (EAM):", eam)


# Calcular EPAM
erro_absoluto = abs(serie_test_matriculas - previsoes_sarima2)
epam = (erro_absoluto.mean() / previsoes_sarima2.mean()) * 100

print("Erro Percentual Absoluto Médio (EPAM):", epam)

# Aplicar algoritmo alisamento exponencial simples na base de dados
contratos
# Selecionar a coluna de interesse
serie_treino_contratos =
pd_contratos.loc[mascara_contratos_treino,'Contratos']

# Aplicar o algoritmo de alisamento exponencial
modelo = SimpleExpSmoothing(serie_treino_contratos)
modelo_ajustado = modelo.fit()

# Realizar previsões
previsoes3 =
modelo_ajustado.predict(start=len(serie_treino_contratos),
end=len(serie_treino_contratos)+ 17)

# Criar figura
fig = go.Figure()

# Adicionar dados de treino ao gráfico
fig.add_trace(go.Scatter(
    x=serie_treino_contratos.index,
    y=serie_treino_contratos,
    mode='lines',
    name='Dados de Treino'
))

# Adicionar previsões ao gráfico
fig.add_trace(go.Scatter(
    x=previsoes3.index,
    y=previsoes3,
    mode='lines',
    name='Previsões'
))

# Configurar layout do gráfico
fig.update_layout(
```

```
    xaxis_title='Período',
    yaxis_title='Contratos/Matrículas',
    title='Previsão de Matrículas usando Alisamento Exponencial',
    legend=dict(
        traceorder='normal'
    )
)

# Exibir gráfico
fig.show()

# Calcular EQM
serie_test_matriculas =
pd_matriculas.loc[mascara_matriculas_test,'Matriculas']
eqm = mean_squared_error(serie_test_matriculas, previsoes3)

print("Erro Quadrático Médio (EQM):", eqm)

# Calcular REQM
serie_test_matriculas =
pd_matriculas.loc[mascara_matriculas_test,'Matriculas']
eqm = mean_squared_error(serie_test_matriculas, previsoes3)
reqm = np.sqrt(eqm)

print("Raiz do Erro Quadrático Médio (REQM):", reqm)

# Calcular EAM
serie_test_matriculas =
pd_matriculas.loc[mascara_matriculas_test,'Matriculas']
eam = mean_absolute_error(serie_test_matriculas, previsoes3)

print("Erro Absoluto Médio (EAM):", eam)

# Calcular EPAM
erro_absoluto = abs(serie_test_matriculas - previsoes3)
epam = (erro_absoluto.mean() / previsoes3.mean()) * 100

print("Erro Percentual Absoluto Médio (EPAM):", epam)

# Aplicar algorimo alisamento exponencial sazonal na base de dados
contratos
# Criar a série temporal a partir da coluna "Contratos"
serie_treino_contratos =
pd_contratos.loc[mascara_contratos_treino,'Contratos']

# Aplicar o algoritmo de alisamento exponencial sazonal
modelo = ExponentialSmoothing(serie_treino_contratos, seasonal='add',
seasonal_periods=12)
modelo_ajustado = modelo.fit()
```

```
# Realizar previsões
previsoes4 =
modelo_ajustado.predict(start=len(serie_treino_contratos),
end=len(serie_treino_contratos)+17)

# Criar figura
fig = go.Figure()

# Adicionar dados de treino ao gráfico
fig.add_trace(go.Scatter(
    x=serie_treino_contratos.index,
    y=serie_treino_contratos,
    mode='lines',
    name='Dados de Treino'
))

# Adicionar previsões ao gráfico
fig.add_trace(go.Scatter(
    x=previsoes4.index,
    y=previsoes4,
    mode='lines',
    name='Previsões'
))


# Configurar layout do gráfico
fig.update_layout(
    xaxis_title='Período',
    yaxis_title='Contratos/Matrículas',
    title='Previsão de Matrículas usando Alisamento Exponencial
Sazonal',
    legend=dict(
        traceorder='normal'
    )
)

# Exibir gráfico
fig.show()

# Calcular EQM
serie_test_matriculas =
pd_matriculas.loc[mascara_matriculas_test,'Matriculas']
eqm = mean_squared_error(serie_test_matriculas, previsoes4)

print("Erro Quadrático Médio (EQM):", eqm)

# Calcular REQM
```

```
serie_test_matriculas =
pd_matriculas.loc[mascara_matriculas_test,'Matriculas']
eqm = mean_squared_error(serie_test_matriculas, previsoes4)
reqm = np.sqrt(eqm)

print("Raiz do Erro Quadrático Médio (REQM):", reqm)

# Calcular EAM
serie_test_matriculas =
pd_matriculas.loc[mascara_matriculas_test,'Matriculas']
eam = mean_absolute_error(serie_test_matriculas, previsoes4)

print("Erro Absoluto Médio (EAM):", eam)

# Calcular EPAM
erro_absoluto = abs(serie_test_matriculas - previsoes4)
epam = (erro_absoluto.mean() / previsoes4.mean()) * 100

print("Erro Percentual Absoluto Médio (EPAM):", epam)

# Aplicar algoritmo de redes neuronais na base de dados de contratos
from sklearn.model_selection import train_test_split
pd_contratos = pd.DataFrame(tabela_contratos_análise)

# Normalizar os dados de entrada
scaler = MinMaxScaler(feature_range=(0, 1))
serie_contratos_normalizada =
scaler.fit_transform(pd_contratos['Contratos'].values.reshape(-1, 1))

# Preparar os dados de treino, validação e teste
X_train, X_temp, y_train, y_temp =
train_test_split(serie_contratos_normalizada,
serie_contratos_normalizada, test_size=0.3, shuffle=False)
X_val, X_test, y_val, y_test1 = train_test_split(X_temp, y_temp,
test_size=0.5, shuffle=False)

# Função para criar o conjunto de dados de entrada e saída
def create_dataset(dataset, janela_tempo=1):
    data_X, data_Y = [], []
    for i in range(len(dataset) - janela_tempo +1  ): # +1 para
complementar a posição 0 da matriz
        data_X.append(dataset[i:i+janela_tempo])
        data_Y.append(dataset[i+janela_tempo-1])
    return np.array(data_X), np.array(data_Y)

janela_tempo = 1

# Criar o conjunto de dados de treino, validação e teste
X_train, y_train = create_dataset(X_train, janela_tempo)
```

```
X_val, y_val = create_dataset(X_val, janela_tempo)
X_test, y_test1 = create_dataset(X_test, janela_tempo)

# Criar o modelo de redes neuronais
modelo = Sequential()
modelo.add(LSTM(50, input_shape=(janela_tempo, 1)))
modelo.add(Dense(1))
modelo.compile(loss='mean_squared_error', optimizer='adam')

# Função para criar o gráfico de treino e validação
def plot_loss(history):
    plt.figure(figsize=(8, 6))
    plt.plot(history.history['loss'], label='Train Loss')
    plt.plot(history.history['val_loss'], label='Val Loss')
    plt.xlabel('Época')
    plt.ylabel('Loss')
    plt.title('Gráfico de Train Loss e Val Loss')
    plt.legend()
    plt.show()

# Treinar o modelo com o conjunto de treino
history = modelo.fit(X_train, y_train, epochs=100, batch_size=32,
validation_data=(X_val, y_val), verbose=1)

# Realizar previsões para o conjunto de teste
previsoes_redes2_normalizadas = modelo.predict(X_test)
previsoes_redes2 =
scaler.inverse_transform(previsoes_redes2_normalizadas)

# Exibir gráfico com as previsões
fig = go.Figure()

# Adicionar dados de treino ao gráfico
fig.add_trace(go.Scatter(
    x=pd_contratos.index[:len(y_train)],
    y=scaler.inverse_transform(y_train).flatten(),
    mode='lines',
    name='Dados de Treino'
))

# Adicionar dados de validação ao gráfico
fig.add_trace(go.Scatter(
    x=pd_contratos.index[len(y_train):len(y_train)+len(y_val)-1],
    y=scaler.inverse_transform(y_val).flatten(),
    mode='lines',
    name='Dados de Validação',
    line=dict(color='green')
))
```

```
# Adicionar previsões ao gráfico
fig.add_trace(go.Scatter(
    x=pd_contratos.index[len(y_train)+len(y_val): len(y_train) +
len(y_val) + len(y_test1)],
    y=previsoes_redes2.flatten(),
    mode='lines',
    name='Previsões',
    line=dict(color='red')
))
# Adicionar Matriculas reais ao gráfico
fig.add_trace(go.Scatter(
    x=serie_test_matriculas.index,
    y=serie_test_matriculas,
    mode='lines',
    name='Matriculas Reais'
))

# Configurar layout do gráfico
fig.update_layout(
    xaxis_title='Período',
    yaxis_title='Contratos/Matrículas',
    title='Previsão de Matrículas usando Redes Neuronais',
    legend=dict(
        traceorder='normal'
    )
)

# Exibir gráfico
fig.show()
plot_loss(history)

# Calcular o EQM
eqm = mean_squared_error(y_test_real, previsoes_redes2)

print("Erro Quadrático Médio (EQM):", eqm)

# Calcular o REQM
eqm = mean_squared_error(y_test_real, previsoes_redes2)
reqm = np.sqrt(eqm)
print("Raiz do Erro Quadrático Médio (REQM):", reqm)

# Calcular EAM
eam = mean_absolute_error(y_test_real, previsoes_redes2)

print("EAM:", eam)

# Calcular EPAM
previsoes_redes2_normalizadas = modelo.predict(X_test)
```

```
previsoes_redes2 =
scaler.inverse_transform(previsoes_redes2_normalizadas)
erro_absoluto = abs(y_test_real - previsoes_redes2 )
epam = (erro_absoluto.mean() / y_test_real.mean()) * 100

print("Erro Percentual Absoluto Médio (EPAM):", epam)

# Aplicar algoritmo ARIMA na base de dados de contratos
# Criar a série temporal a partir da coluna "Contratos"
serie_treino_contratos = pd_contratos.loc[mascara_contratos_treino,
'Contratos']

# Identificar os parâmetros do modelo ARIMA
order = (21, 1, 1)  # Parâmetros (p, d, q)

# Criar e ajustar o modelo ARIMA aos dados de treino
modelo_arima = sm.tsa.ARIMA(serie_treino_contratos, order=order)
modelo_arima_ajustado = modelo_arima.fit()

# Realizar previsões
previsoes_arima3 =
modelo_arima_ajustado.predict(start=len(serie_treino_contratos),
end=len(serie_treino_contratos) + 17)

# Criar figura
fig = go.Figure()

# Adicionar dados de treino ao gráfico
fig.add_trace(go.Scatter(
    x=serie_treino_contratos.index,
    y=serie_treino_contratos,
    mode='lines',
    name='Dados de Treino'
))

# Adicionar previsões do ARIMA ao gráfico
fig.add_trace(go.Scatter(
    x=previsoes_arima3.index,
    y=previsoes_arima3,
    mode='lines',
    name='Previsões ARIMA'
))

# Configurar layout do gráfico
fig.update_layout(
    xaxis_title='Período',
    yaxis_title='Contratos/Matrículas',
    title='Previsão de Matrículas usando ARIMA',
    legend=dict(
```

```
        traceorder='normal'
    )
)

# Exibir gráfico
fig.show()

# Calcular EQM
serie_test_matriculas =
pd_matriculas.loc[mascara_matriculas_test,'Matriculas']
eqm = mean_squared_error(serie_test_matriculas, previsoes_arima3)

print("Erro Quadrático Médio (EQM):", eqm)

# Calcular REQM
serie_test_matriculas =
pd_matriculas.loc[mascara_matriculas_test,'Matriculas']
eqm = mean_squared_error(serie_test_matriculas, previsoes_arima3)
reqm = np.sqrt(eqm)

print("Raiz do Erro Quadrático Médio (REQM):", reqm)

# Calcular EAM
serie_test_matriculas =
pd_matriculas.loc[mascara_matriculas_test,'Matriculas']
eam = mean_absolute_error(serie_test_matriculas, previsoes_arima3)

print("Erro Absoluto Médio (EAM):", eam)

# Calcular EPAM
erro_absoluto = abs(serie_test_matriculas - previsoes_arima3)
epam = (erro_absoluto.mean() / previsoes_arima3.mean()) * 100

print("Erro Percentual Absoluto Médio (EPAM):", epam)

# Aplicar algoritmo ARIMA com AutoARIMA na base de dados de contratos
# Identificar os melhores parâmetros usando o AutoARIMA
modelo_arima = pm.auto_arima(serie_treino_contratos, seasonal=False,
suppress_warnings=True)
order4 = modelo_arima.order
order4

# Criar a série temporal a partir da coluna "Contratos"
serie_treino_contratos = pd_contratos.loc[mascara_contratos_treino,
'Contratos']

# Utilizar os parâmetros AutoARIMA
order = order4  # Parâmetros (p, d, q)
```

```
# Criar e ajustar o modelo ARIMA aos dados de treino
modelo_arima = sm.tsa.ARIMA(serie_treino_contratos, order=order)
modelo_arima_ajustado = modelo_arima.fit()

# Realizar previsões
previsoes_arima4 =
modelo_arima_ajustado.predict(start=len(serie_treino_contratos),
end=len(serie_treino_contratos) + 17)

# Criar figura
fig = go.Figure()

# Adicionar dados de treino ao gráfico
fig.add_trace(go.Scatter(
    x=serie_treino_contratos.index,
    y=serie_treino_contratos,
    mode='lines',
    name='Dados de Treino'
))

# Adicionar previsões do ARIMA ao gráfico
fig.add_trace(go.Scatter(
    x=previsoes_arima4.index,
    y=previsoes_arima4,
    mode='lines',
    name='Previsões ARIMA'
))

# Configurar layout do gráfico
fig.update_layout(
    xaxis_title='Período',
    yaxis_title='Contratos/Matrículas',
    title='Previsão de Matrículas usando ARIMA e AutoARIMA',
    legend=dict(
        traceorder='normal'
    )
)

# Exibir gráfico
fig.show()

# Calcular EQM
serie_test_matriculas =
pd_matriculas.loc[mascara_matriculas_test,'Matriculas']
eqm = mean_squared_error(serie_test_matriculas, previsoes_arima4)

print("Erro Quadrático Médio (EQM):", eqm)

# Calcular REQM
```

```
serie_test_matriculas = pd_matriculas.loc[mascara_matriculas_test,
'Matriculas']
eqm = mean_squared_error(serie_test_matriculas, previsoes_arima4)
reqm = np.sqrt(eqm)

print("Raiz do Erro Quadrático Médio (REQM):", reqm)

# Calcular EAM
serie_test_matriculas = pd_matriculas.loc[mascara_matriculas_test,
'Matriculas']
eam = mean_absolute_error(serie_test_matriculas, previsoes_arima4)

print("Erro Absoluto Médio (EAM):", eam)

# Calcular EPAM
erro_absoluto = abs(serie_test_matriculas - previsoes_arima4)
epam = (erro_absoluto.mean() / previsoes_arima4.mean()) * 100

print("Erro Percentual Absoluto Médio (EPAM):", epam)

# Aplicar algoritmo SARIMA na base de dados de contratos
# Criar a série temporal a partir da coluna "Contratos"
serie_treino_contratos = pd_contratos.loc[mascara_contratos_treino,
'Contratos']

# Identificar os parâmetros do modelo SARIMA
order = (21, 1, 1)  # Parâmetros (p, d, q)
seasonal_order = (0, 1, 1, 12)  # Parâmetros (P, D, Q, S)

# Criar e ajustar o modelo SARIMA aos dados de treino
modelo_sarima = sm.tsa.SARIMAX(serie_treino_contratos, order=order,
seasonal_order=seasonal_order)
modelo_sarima_ajustado = modelo_sarima.fit()

# Realizar previsões
previsoes_sarima3 =
modelo_sarima_ajustado.predict(start=len(serie_treino_contratos),
end=len(serie_treino_contratos) + 17)

# Criar figura
fig = go.Figure()

# Adicionar dados de treino ao gráfico
fig.add_trace(go.Scatter(
    x=serie_treino_contratos.index,
    y=serie_treino_contratos,
    mode='lines',
    name='Dados de Treino'
))
```

```
# Adicionar previsões do SARIMA ao gráfico
fig.add_trace(go.Scatter(
    x=previsoes_sarima3.index,
    y=previsoes_sarima3,
    mode='lines',
    name='Previsões SARIMA'
))

# Configurar layout do gráfico
fig.update_layout(
    xaxis_title='Período',
    yaxis_title='Contratos/Matrículas',
    title='Previsão de Matrículas usando SARIMA',
    legend=dict(
        traceorder='normal'
    )
)

# Exibir gráfico
fig.show()

# Calcular EQM
serie_test_matriculas =
pd_matriculas.loc[mascara_matriculas_test,'Matriculas']
eqm = mean_squared_error(serie_test_matriculas, previsoes_sarima3)

print("Erro Quadrático Médio (EQM):", eqm)

# Calcular REQM
serie_test_matriculas = pd_matriculas.loc[mascara_matriculas_test,
'Matriculas']
eqm = mean_squared_error(serie_test_matriculas, previsoes_sarima3)
reqm = np.sqrt(eqm)

print("Raiz do Erro Quadrático Médio (REQM):", reqm)

# Calcular EAM
serie_test_matriculas = pd_matriculas.loc[mascara_matriculas_test,
'Matriculas']
eam = mean_absolute_error(serie_test_matriculas, previsoes_sarima3)

print("Erro Absoluto Médio (EAM):", eam)

# Calcular EPAM
erro_absoluto = abs(serie_test_matriculas - previsoes_sarima3)
epam = (erro_absoluto.mean() / previsoes_sarima3.mean()) * 100

print("Erro Percentual Absoluto Médio (EPAM):", epam)
```

```
# Aplicar algoritmo SARIMA com AutoSARIMA na base de dados de
contratos
# Identificar os melhores parâmetros com AutoARIMA para SARIMA
modelo_sarima = pm.auto_arima(serie_treino_contratos,
seasonal=True,m=12, suppress_warnings=True)
order3 = modelo_sarima.order
seasonal_order3 = modelo_sarima.seasonal_order

print("SARIMA Order:", order3)
print("SARIMA Seasonal Order:", seasonal_order3)

#Criar a série temporal a partir da coluna "Contratos"
serie_treino_contratos = pd_contratos.loc[mascara_contratos_treino,
'Contratos']

# Identificar os parâmetros do modelo SARIMA
order = order3  # Parâmetros (p, d, q)
seasonal_order = seasonal_order3 # Parâmetros (P, D, Q, S)

# Criar e ajustar o modelo SARIMA aos dados de treino
modelo_sarima = sm.tsa.SARIMAX(serie_treino_contratos, order=order,
seasonal_order=seasonal_order)
modelo_sarima_ajustado = modelo_sarima.fit()

# Realizar previsões
previsoes_sarima4 =
modelo_sarima_ajustado.predict(start=len(serie_treino_contratos),
end=len(serie_treino_contratos) + 17)

# Criar figura
fig = go.Figure()

# Adicionar dados de treino ao gráfico
fig.add_trace(go.Scatter(
    x=serie_treino_contratos.index,
    y=serie_treino_contratos,
    mode='lines',
    name='Dados de Treino'
))

# Adicionar previsões do SARIMA ao gráfico
fig.add_trace(go.Scatter(
    x=previsoes_sarima4.index,
    y=previsoes_sarima4,
    mode='lines',
    name='Previsões SARIMA'
))
```

```
# Configurar layout do gráfico
fig.update_layout(
    xaxis_title='Período',
    yaxis_title='Contratos/Matrículas',
    title='Previsão de Matrículas usando SARIMA e AutoSARIMA',
    legend=dict(
        traceorder='normal'
    )
)

# Exibir gráfico
fig.show()

# Calcular EQM
serie_test_matriculas =
pd_matriculas.loc[mascara_matriculas_test,'Matriculas']
eqm = mean_squared_error(serie_test_matriculas, previsoes_sarima4)

print("Erro Quadrático Médio (EQM):", eqm)

# Calcular REQM
serie_test_matriculas = pd_matriculas.loc[mascara_matriculas_test,
'Matriculas']
eqm = mean_squared_error(serie_test_matriculas, previsoes_sarima4)
reqm = np.sqrt(eqm)

print("Raiz do Erro Quadrático Médio (REQM):", reqm)

# Calcular EAM
serie_test_matriculas = pd_matriculas.loc[mascara_matriculas_test,
'Matriculas']
eam = mean_absolute_error(serie_test_matriculas, previsoes_sarima4)

print("Erro Absoluto Médio (EAM):", eam)

# Calcular EPAM
erro_absoluto = abs(serie_test_matriculas - previsoes_sarima4)
epam = (erro_absoluto.mean() / previsoes_sarima4.mean()) * 100


print("Erro Percentual Absoluto Médio (EPAM):", epam)

# Comparar as previsões alcançadas pela Toyota com o melhor algoritmo
e as matrículas reais
# Ler a base de dados onde estão as previsões feitas pela Toyota
base_previsoes_toyota = pd.read_excel('/content/Previsões-Orçamento
Toyota.xlsx')
pd_previsoes_toyota = pd.DataFrame(base_previsoes_toyota)
# Renomear os índices dos períodos para começar em 84
```

```
pd_previsoes_toyota.index = range(84, 84 + len(pd_previsoes_toyota))
previsoes_toyota = pd_previsoes_toyota['Previsão Matriculas']
previsoes_toyota

# Criar figura
fig = go.Figure()

# Adicionar dados de treino ao gráfico
fig.add_trace(go.Scatter(
    x=previsoes_toyota.index,
    y=previsoes_toyota,
    mode='lines',
    name='Previsões Toyota',
    line=dict(color='grey')
))

# Adicionar previsões ao gráfico
fig.add_trace(go.Scatter(
    x=pd_matriculas.index[len(y_train)+len(y_val): len(y_train) +
len(y_val) + len(y_test)],
    y=previsoes_redes1.flatten(),
    mode='lines',
    name='Previsões',
    line=dict(color='red')
))
# Adicionar Matriculas reais ao gráfico
fig.add_trace(go.Scatter(
    x=serie_test_matriculas.index,
    y=serie_test_matriculas,
    mode='lines',
    name='Matriculas Reais',
    line=dict(color='#FFCC00')
))

# Configurar layout do gráfico
fig.update_layout(
    xaxis_title='Período',
    yaxis_title='Matrículas',
    title='Comparação de previsões com Matrículas Reais',
    legend=dict(
        traceorder='normal'
    )
)
fig.show()

# Calcular EQM para as previsões da Toyota
serie_test_matriculas =
pd_matriculas.loc[mascara_matriculas_test,'Matriculas']
eqm = mean_squared_error(serie_test_matriculas, previsoes_toyota)
```

```
print("Erro Quadrático Médio (EQM):", eqm)

# Calcular REQM para as previsões da Toyota
serie_test_matriculas = pd_matriculas.loc[mascara_matriculas_test,
'Matriculas']
eqm = mean_squared_error(serie_test_matriculas, previsoes_toyota)
reqm = np.sqrt(eqm)

print("Raiz do Erro Quadrático Médio (REQM):", reqm)

# Calcular EAM para as previsões da Toyota
serie_test_matriculas = pd_matriculas.loc[mascara_matriculas_test,
'Matriculas']

# Calcular o EAM
eam = mean_absolute_error(serie_test_matriculas, previsoes_toyota)

print("Erro Absoluto Médio (EAM):", eam)

# Calcular EPAM para as previsões da Toyota
erro_absoluto = abs(serie_test_matriculas - previsoes_toyota)
epam = (erro_absoluto.mean() / previsoes_toyota.mean()) * 100

print("Erro Percentual Absoluto Médio (EPAM):", epam)
```