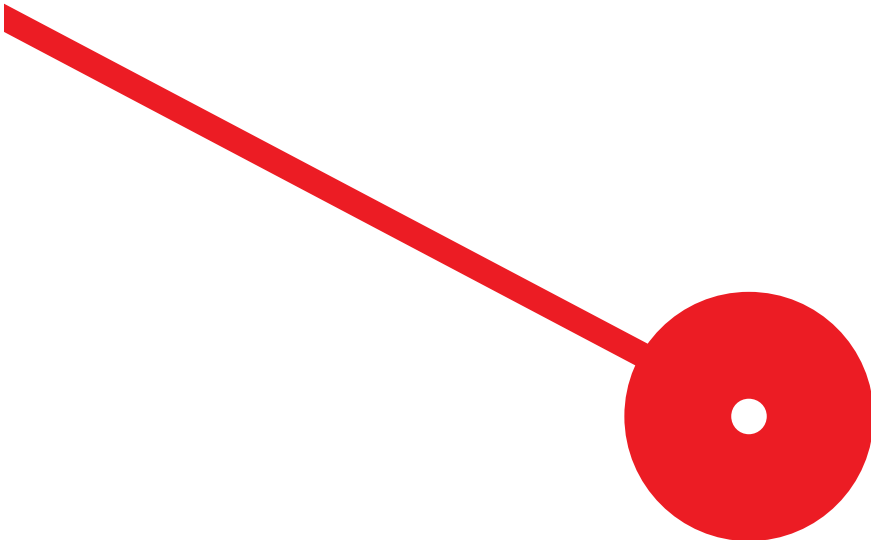
INSTITUTO
SUPERIOR
DE CONTABILIDADE
E ADMINISTRAÇÃO
DO PORTO
POLITÉCNICO
DO PORTO

M MESTRADO
BUSINESS INTELLIGENCE AND ANALYTICS

Desenvolvimento de um Modelo Preditivo como Ferramenta de Eficiência Operacional

João Diogo Dias Costa

10/2025



João Diogo Dias Costa. Desenvolvimento de um Modelo Preditivo como Ferramenta de Eficiência Operacional 10/2025

INSTITUTO
SUPERIOR
DE CONTABILIDADE
E ADMINISTRAÇÃO
DO PORTO
POLITÉCNICO
DO PORTO

M MESTRADO
BUSINESS INTELLIGENCE AND ANALYTICS

Desenvolvimento de um Modelo Preditivo como Ferramenta de Eficiência Operacional

João Diogo Dias Costa

Relatório de Estágio apresentado ao Instituto Superior de Contabilidade e Administração do Porto para a obtenção do grau de Mestre em Business Intelligence and Analytics, sob orientação da Professora Doutora Patrícia Alexandra Gregório Ramos e do Eng. Nuno Ulisses Rosalino da Costa.

João Diogo Dias Costa. Desenvolvimento de um Modelo Preditivo como Ferramenta de Eficiência Operacional
10/2025

Dedicatória

Dedico este trabalho aos meus pais, avó e namorada, que ao longo dos anos sempre me apoiaram e fizeram sacrifícios para que eu tivesse tudo o que era necessário para atingir o sucesso. São quem me faz querer ser uma melhor pessoa todos os dias, e por isso lhes dedico todo o esforço e dedicação por trás deste relatório, e seus consequentes frutos profissionais.

Agradecimentos

Em primeiro lugar, expresso a minha profunda gratidão à minha família, pelo apoio incondicional e pela dedicação constante ao longo dos anos.

À professora Patrícia Gregório Ramos, agradeço pela sua constante disponibilidade, pela clareza com que sempre respondeu às minhas dúvidas e pela orientação segura que me proporcionou ao longo deste percurso.

Ao meu coordenador de estágio na Luís Simões, Nuno Ulisses Costa, bem como aos restantes colegas da Direção de Dados, agradeço pela confiança depositada em mim, pela colaboração e pela total disponibilidade demonstrada no decorrer das minhas tarefas.

Resumo:

No âmbito do segundo semestre do mestrado em *Business Intelligence and Analytics*, do Instituto Superior de Contabilidade e Administração do Porto (ISCAP), o presente relatório tem como objetivo descrever e dar provas das tarefas desenvolvidas no estágio curricular entre 5 de março de 2025 e 13 de junho de 2025, realizado em regime híbrido na Luís Simões, uma empresa de transporte e logística com sede em Loures.

As funções e tarefas exercidas durante o estágio focaram-se na realização da previsão de produção de caixas e expedições, de um cliente, nos dois maiores armazéns da empresa, em Gaia e no Carregado.

No presente relatório encontra-se a descrição detalhada de todas as atividades executadas durante o estágio na Luís Simões, onde pude aplicar todos os conhecimentos adquiridos nas unidades curriculares do mestrado de *Business Intelligence and Analytics* do ISCAP.

Durante o estágio, tive a oportunidade de utilizar diversas ferramentas e de consolidar os meus conhecimentos teóricos através de atividades práticas com clientes, cumprindo assim todos os objetivos definidos para o período.

Em suma, o estágio consolidou a aplicação prática dos conhecimentos, culminando na implementação de modelos preditivos que resultaram em melhorias significativas na tomada de decisão e na eficiência operacional da organização. A experiência permitiu o desenvolvimento de competências técnicas e analíticas essenciais, ao mesmo tempo que identificou oportunidades de aperfeiçoamento dos modelos de previsão.

Palavras-chave: Previsão, Data Analytics, Data-Driven Decision Making, Visualização de Dados.

Abstract:

As part of the second semester of the Master's degree in Business Intelligence and Analytics at the Instituto Superior de Contabilidade e Administração do Porto (ISCAP), this report aims to describe and provide evidence of the tasks developed during the curricular internship, which took place between March 5, 2025, and June 13, 2025. The internship was conducted under a hybrid model at Luís Simões, a transport and logistics company headquartered in Loures.

The duties and tasks performed during the internship focused on forecasting a client's box production and shipments at the company's two largest warehouses, located in Gaia and Carregado.

This report provides a detailed description of all activities executed during the internship at Luís Simões, where I was able to apply all the knowledge acquired in the course units of the Master's degree in Business Intelligence and Analytics at ISCAP.

During the internship, I had the opportunity to utilize various tools and to consolidate my theoretical knowledge through practical activities involving clients, thereby fulfilling all the objectives defined for the period.

In summary, the internship solidified the practical application of knowledge, culminating in the implementation of predictive models that resulted in significant improvements in the organization's decision-making and operational efficiency. The experience allowed for the development of essential technical and analytical competencies, while also identifying opportunities for refining the forecasting models.

Keywords: Forecasting, Data Analytics, Data-Driven Decision Making, Data Visualization.

Índice Geral

Índice de Figuras

CAPÍTULO I – INTRODUÇÃO

1 Introdução

O presente relatório de estágio foi elaborado no âmbito do Mestrado em *Business Intelligence and Analytics* do Instituto Superior de Contabilidade e Administração do Porto (ISCAP), com vista à obtenção do grau de mestre. Orientado pela Professora Doutora Patrícia Gregório Ramos, do ISCAP, e pelo Diretor Nuno Ulisses Costa, da Luís Simões, este documento visa descrever e refletir sobre as atividades desenvolvidas durante o estágio curricular realizado na empresa, entre 5 de março e 13 de junho de 2025.

A Luís Simões, uma referência no setor da logística e transportes, proporcionou um ambiente profissional dinâmico, onde foi possível aplicar os conhecimentos adquiridos ao longo do mestrado e desenvolver competências práticas em *Business Intelligence*. A escolha deste estágio, em detrimento de uma dissertação, foi motivada pelo meu percurso académico em Gestão de Marketing e pela recente transição para a área de *Business Intelligence*. Este contexto tornou essencial a integração num ambiente corporativo que me permitisse consolidar a formação teórica, adquirir experiência prática e preparar-me para os desafios do mercado de trabalho.

O estágio, realizado em regime híbrido, decorreu das 9h às 18h, de segunda a sexta-feira, sob a supervisão de Nuno Ulisses Costa. A integração na empresa foi precedida de uma entrevista por videochamada, na qual foram apresentados os valores, as atividades e as expectativas da Luís Simões, bem como as minhas motivações e objetivos profissionais. Apesar da minha limitada experiência prática na área, a empresa valorizou a minha dedicação e compromisso, confiando-me tarefas que envolveram o tratamento de dados reais de clientes, exigindo rápida adaptação às dinâmicas do ambiente corporativo.

Durante o estágio, desenvolvi um projeto com o objetivo de criar um modelo preditivo para estimar a quantidade de caixas que o armazém deverá produzir na semana seguinte. Atualmente, esse processo é feito com base em estimativas manuais e imprecisas, o que dificulta a implementação de um planeamento eficiente. A ausência de um modelo confiável gera desafios significativos, como a superprodução, que resulta em custos elevados com armazenamento e possíveis desperdícios, ou a subprodução, que pode causar atrasos nas entregas e insatisfação dos clientes.

Com a implementação de um sistema de previsão baseado em dados, procurou-se resolver este problema através de uma estimativa mais precisa da procura semanal. A adoção deste modelo preditivo permite um planeamento mais eficiente da produção, alinhando-a às

reais necessidades do mercado e otimizando os recursos disponíveis. O impacto esperado inclui a redução dos custos operacionais, a minimização dos desperdícios, a melhoria na alocação de recursos como matérias-primas e mão de obra, bem como o aumento da satisfação dos clientes através de entregas mais pontuais e consistentes. Esta mudança contribui ainda para tornar a operação mais resiliente face a variações sazonais ou inesperadas na procura.

A principal motivação deste projeto reside na importância de prever com precisão a produção necessária. Antecipar a procura possibilita uma gestão mais estratégica, reduz a incerteza e melhora a tomada de decisão. Os benefícios diretos incluem uma maior eficiência operacional, a redução de perdas financeiras e o aumento da competitividade da empresa no mercado. Em suma, a previsão é essencial para transformar a produção de reativa em proativa, promovendo uma operação mais inteligente e sustentável.

Este estágio representou uma oportunidade única para aprofundar competências técnicas e interpessoais, alinhando a minha formação académica com as exigências do mercado de trabalho na área de *Business Intelligence*. O presente relatório procura, assim, documentar e refletir sobre esta experiência, destacando o seu contributo para o meu desenvolvimento profissional e académico.

Este relatório encontra-se organizado em capítulos, iniciando com a introdução, seguido da revisão de literatura, apresentação da entidade acolhedora, a descrição de todas as tarefas desenvolvidas durante o estágio, a visualização e análise dos resultados e terminando com a conclusão.

2 Revisão da Literatura

2.1 Previsão de Séries Temporais: Conceitos e Métodos

A previsão de séries temporais constitui um elemento crucial na análise de dados e na tomada de decisões, particularmente em áreas como a economia, finanças, ciências ambientais e saúde. A presente revisão de literatura tem como objetivo sintetizar os conceitos e metodologias predominantes no âmbito da previsão de séries temporais. Nos últimos anos, assistiu-se a uma evolução significativa tanto das técnicas clássicas como das mais avançadas, expandindo os limites do que é possível alcançar através da modelação preditiva.

Os modelos clássicos de séries temporais, como o Modelo Autorregressivo Integrado de Médias Móveis (ARIMA), o ARIMA Sazonal (SARIMA) e o Modelo Autorregressivo de Médias Móveis (ARMA), dominaram historicamente o panorama das metodologias de previsão, devido à sua simplicidade e eficácia na captura de padrões lineares nos dados. No entanto, as suas premissas limitam frequentemente a precisão preditiva, sobretudo em situações de relações não-lineares e complexidades inerentes aos dados do mundo real (Rathipriya et al., 2022; Sun et al., 2021). É importante destacar que os modelos ARIMA e congéneres tendem a apresentar um desempenho decrescente quando os dados subjacentes evidenciam alterações rápidas ou tendências que se afastam dos comportamentos passados (Rathipriya et al., 2022). Esta limitação levou à crescente exploração de metodologias de *machine learning* capazes de lidar de forma mais eficaz com tais complexidades.

As técnicas de *machine learning*, em particular aquelas que envolvem redes neuronais, têm ganho destaque como alternativas promissoras aos modelos tradicionais. Estudos apontam que os modelos de *deep learning*, como as redes neuronais recorrentes (RNN) e as suas variantes, como as redes Long Short-Term Memory (LSTM), são capazes de captar padrões temporais implexos, proporcionando uma maior precisão nas previsões (Ni et al., 2023; Cheng et al., 2025). Por exemplo, Rathipriya et al. destacaram a eficácia dos modelos de *deep learning* face aos modelos ARIMA tradicionais na previsão da procura farmacêutica, assinalando uma mudança significativa na aplicação da inteligência artificial neste domínio (Rathipriya et al., 2022). Além disso, a integração de modelos de *machine learning* com dados de séries temporais permite previsões mais robustas,

sobretudo em ambientes voláteis, como os mercados financeiros, onde os modelos convencionais revelam fragilidades (Gao, 2025).

Contudo, a aplicação de modelos de *machine learning* à previsão de séries temporais não está isenta de desafios. Problemas como o *overfitting* (o sobreajuste do modelo aos dados de treino, que prejudica a generalização a novos dados), a dependência de grandes volumes de dados e a reduzida interpretabilidade de modelos complexos constituem obstáculos significativos. Em cenários onde os dados de entrada podem sofrer pequenas perturbações, os resultados obtidos podem variar substancialmente, como referem Yoon et al. (2022). Tal constatação evidencia a importância de conceber modelos robustos, capazes de resistir a tais variações. Adicionalmente, a implementação de abordagens probabilísticas na previsão, que contemplam as incertezas inerentes aos dados, pode reforçar a capacidade de apoio à decisão (Yoon et al., 2022; Ghatage et al., 2023). Ao quantificarem as incertezas associadas às previsões, os modelos probabilísticos oferecem uma estrutura transparente para a compreensão dos potenciais cenários futuros (Yoon et al., 2022).

Outro avanço relevante na área prende-se com a exploração de modelos híbridos que combinam métodos clássicos com técnicas de *machine learning*, de forma a tirar partido das vantagens de ambos os paradigmas. Esta abordagem, exemplificada por modelos que integram *wavelet transforms* com técnicas tradicionais de previsão, visa captar de forma mais eficaz tanto as tendências locais como globais (Shaikh et al., 2022). Tal integração pode conduzir a um aumento da precisão preditiva, reduzindo ao mesmo tempo a complexidade estrutural típica dos modelos de *deep learning*. A experimentação com diversas estratégias híbridas revela-se, assim, uma direção promissora para a investigação futura, onde a combinação da simplicidade dos modelos clássicos com a sofisticação de modelos de *machine learning* poderá originar resultados de vanguarda (Shaikh et al., 2022).

Os métodos de séries temporais *fuzzy* constituem igualmente uma alternativa interessante, sobretudo quando se trata de dados caracterizados por elevada incerteza. Com base nos princípios da lógica *fuzzy*, estes métodos permitem transformar observações qualitativas em previsões quantitativas, proporcionando uma maior flexibilidade na especificação dos modelos e na sua adaptação às complexidades do mundo real (Hariyanto et al., 2023). A aplicação inovadora destas abordagens em contextos como a previsão da produção realça a sua crescente relevância. Por exemplo, modelos que utilizam séries temporais *fuzzy* para

prever resultados agrícolas demonstram não apenas eficácia operacional, mas também a utilidade de incorporar variáveis linguísticas que refletem o conhecimento de peritos (Arika et al., 2024).

As análises comparativas entre metodologias evidenciam a natureza complexa da previsão de séries temporais, demonstrando que não existe uma solução única e universal. Estudos que comparam a precisão preditiva de diferentes métodos univariados – desde modelos econométricos clássicos até redes neuronais avançadas – validam a necessidade de selecionar os modelos com base nas características dos dados e nos objectivos da previsão (Davidescu et al., 2021). A aplicação bem-sucedida destas técnicas, nomeadamente na previsão económica de indicadores como as taxas de desemprego, influencia significativamente as decisões estratégicas e de política pública, reforçando a integração de práticas de previsão rigorosas em contextos operacionais (Davidescu et al., 2021; Madaras, 2024).

Adicionalmente, as técnicas de modelação espaço-temporal abriram novas possibilidades de previsão, particularmente em contextos ambientais, como a previsão da qualidade do ar (Silva et al., 2023). O dinamismo destes modelos permite uma compreensão mais pormenorizada dos processos dependentes do tempo, facultando aos profissionais a capacidade de prever fenómenos espaciais que frequentemente escapam aos modelos tradicionais. Nestes casos, conjuntos de dados complexos e respetivas dependências são eficazmente geridos através de algoritmos avançados, promovendo avanços em áreas que vão desde a saúde pública ao planeamento urbano.

O campo da previsão de séries temporais continua a evoluir de forma notável. A transição dos modelos clássicos para os modelos de *machine learning* representa uma mudança de paradigma, capacitando investigadores e profissionais a alcançar níveis inéditos de capacidade preditiva. Além disso, a diversificação de modelos, a gestão robusta de erros e a adoção de metodologias adaptativas continuarão a ser elementos centrais para o aumento da precisão preditiva em múltiplas aplicações. À medida que as metodologias evoluem, a exploração contínua de abordagens inovadoras e híbridas, a par dos métodos tradicionais, assegurará que a previsão de séries temporais permaneça na vanguarda da tomada de decisão baseada em dados.

2.2 Previsão de Séries Temporais em Transporte e Logística

Os setores dos transportes e da logística são profundamente influenciados por padrões flutuantes de procura, eficiência operacional e outros fatores dinâmicos. A previsão de séries temporais nestes domínios desempenha um papel crucial na otimização da alocação de recursos, minimização de custos e melhoria da eficiência na prestação de serviços. O rápido avanço das tecnologias de *machine learning* e da capacidade computacional veio potenciar significativamente as capacidades dos modelos de previsão de séries temporais, permitindo lidar com as complexidades e variabilidades inerentes aos cenários logísticos do mundo real. Esta revisão de literatura sintetiza estudos e metodologias contemporâneos sobre a previsão de séries temporais nos setores dos transportes e da logística, com ênfase no impacto dos modelos de *machine learning* e *deep learning*.

A integração de modelos de *machine learning* na previsão de séries temporais gerou melhorias significativas na precisão preditiva e na eficácia operacional da logística de transportes. Por exemplo, Cardona discute a utilização da análise de dados para conceber produtos personalizados de seguros contra interrupções de atividade para operadores de transporte ferroviário de mercadorias, recorrendo à análise de séries temporais para extrair *insights* a partir de dados históricos de desempenho e, assim, formular previsões fundamentadas (Cardona, 2021). Esta abordagem demonstra como o pré-processamento de dados, a análise estatística e a previsão de séries temporais podem, em conjunto, reforçar a resiliência operacional dos sistemas de transporte de mercadorias em contextos de incerteza. Adicionalmente, Kotenko destaca as vantagens dos algoritmos de *machine learning* nos processos de transporte de carga, indicando que os modelos algorítmicos aumentam a precisão das previsões, reduzem custos e melhoram a eficiência das entregas (Kotenko, 2022). Estes resultados sublinham o papel central do *machine learning* na superação dos desafios dinâmicos enfrentados pela logística de transportes.

Um avanço notável na modelação de tráfego é o estudo de Hasan, que aplica técnicas de *machine learning* para prever volumes de veículos em cruzamentos urbanos, especialmente em contextos com variações induzidas por fatores externos, como é exemplo a pandemia de COVID-19. Este estudo de caso demonstra como a análise de séries temporais pode identificar e modelizar eficazmente alterações nos padrões de tráfego, contribuindo para uma melhor mobilidade urbana e estratégias de gestão de congestionamento (Hasan, 2024). De forma semelhante, o trabalho de Sharma et al.

enfatiza a eficácia das redes LSTM no pós-processamento de previsões de caudal fluvial, demonstrando a aplicabilidade das abordagens de *deep learning* a várias aplicações de previsão de séries temporais na logística e nos transportes (Sharma et al., 2022). Ambos os exemplos reiteram a necessidade de abordagens de modelação avançadas para captar dependências temporais complexas nos sistemas de transporte.

Além disso, o desempenho dos modelos de previsão de séries temporais pode ser substancialmente melhorado através de metodologias híbridas que combinam técnicas de aprendizagem superficial (*shallow learning*) e profunda. A investigação conduzida por Panapakidis et al. salienta um modelo híbrido de previsão que integra eficazmente características de ambos os paradigmas para prever o consumo de combustível em navios de passageiros, demonstrando uma notável robustez em contextos aplicacionais (Panapakidis et al., 2020). Esta evidência sugere que os modelos híbridos podem aproveitar os pontos fortes de diversos algoritmos para lidar com a complexidade das tarefas de previsão na logística de transportes. Adicionalmente, a aplicação de modelos como o *fuzzy* ARTMAP e *Cortical Learning Algorithms* (CLA) ilustram a diversidade de metodologias actualmente exploradas neste campo, com Ferreira et al. e Aoki et al. a detalharem o seu sucesso, respetivamente, na previsão energética e nos processos de aprendizagem adaptativa para previsões de séries temporais (Ferreira et al., 2020; Aoki et al., 2021).

Os desafios distintivos colocados pela dinâmica das cadeias de abastecimento requerem abordagens de previsão inovadoras. Wang e Han descrevem a utilização de redes LSTM para a previsão da oferta em estações de tratamento de água em zonas rurais, o que sublinha a relevância deste método na gestão eficaz de cadeias de abastecimento e alocação de recursos (Wang & Han, 2024). De igual modo, a revisão conduzida por Kotenko identifica fatores críticos que influenciam os resultados logísticos, contribuindo assim para o desenvolvimento de modelos de *machine learning* orientados para a melhoria das previsões de consumo de combustível em entregas de cereais (Kotenko, 2022). Estas aplicações específicas evidenciam a necessidade de modelos de previsão precisos, adaptados a cenários logísticos concretos, de modo a potenciar a eficiência global das cadeias de abastecimento.

A previsão da procura continua a ser um componente essencial na logística de transportes, influenciando significativamente a gestão de inventários e a produtividade das cadeias logísticas. A revisão de Abdullahi et al. investiga a eficácia de diversos algoritmos de

machine learning na previsão de vendas a retalho, um contexto semelhante ao de transportes e logística, onde as flutuações na procura são comuns (Abdullahi et al., 2021). A adaptabilidade e robustez de modelos como as florestas aleatórias (Random Forest) e as máquinas de vetores de suporte (*Support Vector Machines*) oferecem perspectivas valiosas sobre a sua potencial aplicação no planeamento de transportes, otimização de rotas e gestão de inventários.

Importa salientar que a otimização dos modelos de previsão não é uma tarefa trivial, como demonstrado por Kourentzes et al., que analisam o desfasamento entre os modelos estatísticos otimizados e o desempenho real de inventários em ambientes operacionais (Kourentzes et al., 2020). Os resultados obtidos enfatizam a necessidade de alinhar as metodologias de previsão com métricas de aplicação prática, de modo a assegurar que as previsões se traduzam em decisões mais eficazes na logística de transportes. A análise simultânea de métricas de inventário e modelos preditivos pode oferecer uma visão holística, ajustada às operações logísticas de precisão.

Adicionalmente, os fatores ambientais desempenham um papel crítico na previsão em transportes. A incorporação de abordagens probabilísticas na previsão de condições meteorológicas, conforme discutido por Kirkwood et al., contribui para a criação de mecanismos abrangentes de apoio à decisão, fundamentais para o planeamento logístico (Kirkwood et al., 2021). De modo complementar, Loken et al. demonstram as aplicações de *machine learning* na produção de previsões meteorológicas severas de curto prazo, com impacto direto nos horários e nas respostas operacionais dos sistemas de transporte (Loken et al., 2020). Estes estudos sublinham a importância de integrar fatores externos nos modelos de previsão, garantindo um planeamento operacional robusto na logística de transportes.

As implicações da previsão de séries temporais vão além da mera precisão preditiva. A previsão eficaz pode proporcionar benefícios estratégicos, como a redução de custos de transporte, a melhoria da prestação de serviços e o aumento da agilidade operacional. Os avanços nas arquiteturas de *deep learning*, particularmente na previsão de séries temporais, possibilitam a criação de modelos mais sofisticados, capazes de se adaptar à complexidade dos dados logísticos. À medida que as ferramentas de análise de dados em tempo real e de previsão evoluem, também aumentará a capacidade dos setores dos transportes e da logística de responder de forma dinâmica a alterações na procura e a condições externas.

O panorama da previsão de séries temporais nos setores dos transportes e da logística está em rápida transformação, impulsionado pelos avanços nas metodologias de *machine learning* e *deep learning*. A síntese dos estudos analisados demonstra a eficácia de diversos modelos na melhoria da precisão preditiva, otimização de recursos e gestão de flutuações da procura. À medida que se assiste a evoluções nesta área, a integração crescente de metodologias de previsão especializadas com métricas operacionais contribuirá substancialmente para o aumento da eficiência e resiliência das operações logísticas. A investigação futura deverá continuar a explorar técnicas preditivas inovadoras, configurações híbridas de modelos e a interação entre variáveis externas e estratégias operacionais internas, com vista a refinar as capacidades da previsão de séries temporais na logística de transportes.

2.3 Utilização do Prophet para Previsão nos Transportes e na Logística

A integração da análise preditiva nos setores dos transportes e da logística tornou-se cada vez mais essencial, à medida que as indústrias procuram eficiência, fiabilidade e crescimento num mercado em rápida evolução. Entre a vasta gama de modelos de previsão explorados neste domínio, o modelo *Prophet*, desenvolvido pelo *Facebook*, destacou-se como uma ferramenta robusta e capaz de lidar com dados de séries temporais caracterizados por diversas tendências e influências sazonais. Esta revisão sintetiza vários estudos que ilustram a aplicabilidade e eficácia do modelo *Prophet* especificamente em contextos de transportes e logística.

Uma área fundamental de aplicação do modelo *Prophet* é a previsão dos volumes de carga, crucial para a gestão da cadeia de abastecimento. Num estudo de Chen et al., foi evidenciada a eficácia do modelo *Prophet*-LSTM na previsão da carga máxima em redes de distribuição, demonstrando a versatilidade do modelo na previsão de cargas, diretamente relacionada com as necessidades da gestão de carga nos sistemas de transporte (Chen et al., 2023). De forma semelhante, a investigação de Majhi et al. destaca a aplicabilidade do modelo *Prophet* na previsão da procura em cadeias de abastecimento agrícolas, indicando que previsões precisas podem melhorar substancialmente a gestão destas cadeias, através de um planeamento de inventário mais eficaz e redução do desperdício alimentar (Majhi et al., 2023).

Noutro contexto importante, o desempenho do modelo *Prophet* na previsão do fluxo de passageiros em sistemas de transporte urbano foi explorado por Lu et al. Através do

desenvolvimento de um modelo combinado *Prophet*-GRU (*Gated Recurrent Unit*), o estudo demonstrou que o *Prophet* consegue captar com sucesso os padrões complexos associados às flutuações da procura de passageiros, especialmente à medida que as redes ferroviárias urbanas se expandem e as dinâmicas de serviço mudam (Lu et al., 2023). Isto destaca o potencial do modelo para apoiar o planeamento operacional e a otimização do serviço em contextos urbanos, componentes essenciais da logística.

No domínio dos serviços de *carsharing*, o *Prophet* também revelou resultados promissores. Alencar et al. demonstraram a capacidade do modelo para prever com precisão a procura, analisando dados reais de múltiplos serviços de *carsharing* em Vancouver, Canadá. Esta capacidade é crucial para gerir a logística nos serviços de mobilidade partilhada, onde a previsibilidade da procura contribui para a eficiência operacional e satisfação dos utilizadores (Alencar et al., 2021). A integração destas ferramentas preditivas pode conduzir a uma prestação de serviço mais fiável, melhorando a experiência dos utilizadores nestas opções urbanas em crescimento.

Adicionalmente, a aplicação do modelo *Prophet* na logística estende-se à análise dos preços de exportação de produtos agrícolas. Jin aplicou um modelo *Prophet* suportado por técnicas de *deep learning* para prever tendências de preços agrícolas, oferecendo assim *insights* relevantes para a gestão logística e das cadeias de abastecimento para exportadores. Este modelo auxilia os intervenientes na tomada de decisões informadas relativas a estratégias de preços e operações logísticas num mercado competitivo (Jin, 2023).

As capacidades preditivas do modelo *Prophet* alinham-se também com as necessidades da previsão da procura de eletricidade em operações logísticas que dependem fortemente do consumo energético. Por exemplo, Dong e Yan analisaram o modelo *Prophet* no âmbito da previsão de consumo elétrico, enfatizando o seu papel na gestão energética — um aspeto crucial para os sistemas de transporte que dependem de fornecimento energético constante para garantir eficiência operacional (Dong & Yan, 2024).

Além disso, a robustez do modelo *Prophet* no tratamento das diversas complexidades dos dados de séries temporais tem sido amplamente reconhecida. Kwarteng e Andreevich realizaram análises comparativas que posicionam o modelo *Prophet* favoravelmente face a métodos tradicionais como o SARIMA, evidenciando a sua capacidade no tratamento de tendências não lineares e padrões sazonais comuns nos conjuntos de dados

relacionados com a logística (Kwarteng & Andreevich, 2024). Esta vantagem comparativa sublinha a utilidade do *Prophet* no desenho de processos preditivos mais eficazes em transportes e logística, onde a compreensão dos padrões de procura é vital.

Os avanços tecnológicos, como a incorporação de *machine learning* e *deep learning* nos modelos de previsão, estão a potenciar ainda mais a utilidade do modelo *Prophet*. Yang et al. destacaram um modelo híbrido *Prophet*-CEEMDAN-ARBiLSTM que apresentou desempenho superior na previsão de carga a curto prazo, demonstrando como abordagens híbridas podem combinar o potencial do *Prophet* com metodologias avançadas de redes neuronais para melhorar a precisão da previsão. Estes modelos híbridos podem ser cruciais em aplicações ligadas à logística de transportes, onde a análise de dados em tempo real é cada vez mais necessária para se adaptar a condições mutáveis (Yang et al., 2024).

Adicionalmente, a exploração de combinações de modelos preditivos pode levar a melhorias nos indicadores de desempenho. Hindarto et al. sugeriram que a combinação do modelo *Prophet* com redes neuronais pode proporcionar melhores previsões para operações logísticas, especialmente em cenários que envolvem a previsão do *throughput* de contentores. Esta abordagem abre caminho para investigações futuras em modelos híbridos, com potencial para prever com maior precisão na logística de transportes (Hindarto et al., 2023).

A exploração do modelo *Prophet* no contexto da previsão de procura também se estende a setores como o turismo, onde dinâmicas particulares exigem previsões precisas. Jung et al. avaliaram o desempenho do modelo *Prophet* na previsão da procura turística para eventos empresariais, ilustrando a sua eficácia face a modelos neuronais mais complexos como o LSTM. Esta investigação sugere que a simplicidade e eficácia do *Prophet* o tornam especialmente adequado para aplicações logísticas relacionadas com movimentos turísticos (Jung et al., 2024).

Por fim, como salientado por Kolková e Ključnikov, a tendência para a integração da inteligência artificial na previsão realça a relevância do modelo *Prophet* tanto para pequenas como para grandes empresas do setor logístico. Os seus resultados sublinham a necessidade de processos de tomada de decisão baseados em dados robustos, que possam melhorar a prestação de serviços e a eficiência operacional em diversas escalas empresariais (Kolková & Ključnikov, 2022).

Em suma, o modelo *Prophet* tem-se revelado uma ferramenta poderosa nos setores dos transportes e da logística, com aplicações que vão desde a previsão de volumes de carga à previsão de fluxos de passageiros e tendências nos preços de exportação agrícola. A sua eficácia na captura de tendências, variações sazonais e a sua capacidade de integração com outras metodologias avançadas posicionam-no como um recurso essencial para intervenientes que pretendem otimizar operações e planeamento estratégico em cadeias de abastecimento cada vez mais complexas. À medida que cresce a procura por previsões precisas num panorama logístico em constante evolução, a investigação e aplicação contínua do modelo *Prophet* provavelmente irão gerar novos *insights* e inovações.

2.4 Previsão Hierárquica de Séries Temporais em Transporte e Logística

A aplicação da previsão hierárquica de séries temporais (HTS) nos setores dos transportes e da logística é uma área fundamental para a melhoria dos processos de tomada de decisão. As estruturas de previsão hierárquica caracterizam-se por arranjos multiníveis que podem representar variações em regiões geográficas, categorias de produtos ou contextos temporais. Estas características permitem análises detalhadas e a capacidade de agregar e desagregar previsões em diferentes níveis organizacionais, conduzindo a uma maior precisão nas previsões e na alocação de recursos.

Um dos principais enquadramentos utilizados na previsão hierárquica de séries temporais são as abordagens *bottom-up*, *top-down* e de combinação ótima propostas por Hyndman et al. (Mohamed, 2023). Estas metodologias oferecem caminhos distintos para reconciliar previsões entre diferentes níveis hierárquicos. Por exemplo, a abordagem *bottom-up* agrega previsões dos níveis inferiores para criar estimativas nos níveis superiores, garantindo que a soma das previsões nos níveis inferiores está alinhada com as correspondentes dos níveis superiores (Taghiyeh et al., 2020). Por outro lado, a abordagem *top-down* gera uma previsão ao nível organizacional mais elevado e distribui-a pelos níveis inferiores com base em proporções estabelecidas. Esta necessidade de coerência entre previsões nos diferentes níveis da hierarquia é crítica em contextos logísticos, onde a distribuição precisa dos recursos depende dessa congruência (Mohamed, 2023).

Uma vantagem significativa da previsão hierárquica em logística e transportes é a sua capacidade de responder às diversas necessidades de informação dos intervenientes em diferentes níveis de gestão. Conforme referido por Taghiyeh et al. (2020), estas estruturas

hierárquicas facilitam a disseminação de dados relevantes para várias funções organizacionais, apoiando eficazmente a tomada de decisão em múltiplos níveis. Esta abordagem em múltiplas camadas assegura que as variações na procura possam ser identificadas e geridas de forma mais eficiente, permitindo às empresas reagir rapidamente às mudanças do mercado (Taghiyeh et al., 2020).

Além disso, avanços em metodologias, como abordagens híbridas que combinam técnicas *top-down* e *bottom-up*, podem melhorar a precisão e coerência das previsões. Rehman et al. (2022) e Abolghasemi et al. (2020) descrevem inovações como a agregação por tamanho de passo, que refinam o processo de previsão ao abordar sinergias internas dentro da estrutura hierárquica, reduzindo assim as margens de erro entre níveis. Estas estratégias híbridas promovem estimativas de procura mais precisas, particularmente importantes em setores onde a coordenação logística é influenciada por diferentes níveis de complexidade na cadeia de abastecimento.

O foco na coerência nas previsões hierárquicas de séries temporais surge como uma preocupação primordial, especialmente em indústrias onde a precisão desempenha um papel crucial na eficiência operacional. Alcançar coerência significa garantir que a soma das previsões nos níveis inferiores corresponda às previsões agregadas nos níveis superiores, tarefa frequentemente dificultada pela natureza dinâmica de muitas indústrias, incluindo os transportes e a logística. Zhang e Li (2021) discutem como estas previsões coerentes não só aumentam a confiança na tomada de decisão dos intervenientes, mas também mitigam os riscos associados à gestão de inventários e planeamento de distribuição.

Adicionalmente, os estudos enfatizam cada vez mais a integração de técnicas de *machine learning* e *deep learning* para reforçar a eficácia da previsão hierárquica de séries temporais. A aplicação de arquiteturas de *deep learning*, como redes Long Short-Term Memory (LSTM), tem demonstrado resultados promissores na manutenção da coerência das previsões através de diferentes estruturas hierárquicas (Sagheer et al., 2021). Estes métodos computacionais avançados permitem modelizar relações complexas e interações em séries temporais hierárquicas, frequentemente presentes em dados logísticos (Sagheer et al., 2021).

Para além disso, a literatura indica que a utilização de abordagens de aprendizagem estruturada pode conduzir a um desempenho superior nas previsões. Zhou et al. (2023)

destacam métodos de otimização baseados em tarefas na previsão hierárquica para captar eficazmente as interdependências entre os níveis da hierarquia. Estas metodologias são particularmente úteis na logística, onde a necessidade de gerir considerações geográficas e temporais é essencial para manter os níveis de serviço e a satisfação do cliente (Wang, 2023).

As complexidades envolvidas na previsão da procura dentro de estruturas hierárquicas refletem também desafios mais amplos enfrentados pelos gestores na área da logística, exigindo a adoção de modelos sofisticados de previsão. Os métodos de previsão da procura logística, categorizados em tradicionais e inteligentes, realçam a importância da seleção de modelos adequados com base em cenários específicos e na disponibilidade de dados (Kramarz & Kmiecik, 2022). Este processo de seleção é crucial, especialmente em períodos marcados pela volatilidade e pela alteração das preferências dos consumidores, que requerem respostas ágeis e bem fundamentadas por parte dos operadores logísticos.

A implementação da previsão hierárquica de séries temporais na logística proporciona um enquadramento organizado que apoia processos eficazes de tomada de decisão em vários níveis de gestão. A utilização de diversas abordagens, desde métodos tradicionais *bottom-up* e *top-down* até técnicas avançadas de *machine learning*, permite às organizações operar com maior precisão e capacidade de resposta. A crescente complexidade das operações na cadeia de abastecimento exige a evolução contínua destas metodologias de previsão para melhorar a coerência e a precisão nas previsões, garantindo que os intervenientes nos transportes e na logística alinhem melhor as suas estratégias com os padrões reais de procura.

A partir das diversas contribuições académicas nesta revisão, é evidente que as práticas inovadoras desenvolvidas na previsão hierárquica não só prometem maior precisão, como também refletem tendências mais amplas para práticas de tomada de decisão integradas nos transportes e na logística (Burba & Chen, 2021; Jahangir & Quilty, 2022). Este espaço ativo de investigação destaca o potencial para avanços futuros que possam refinar ainda mais as metodologias, promover eficiências e, em última análise, transformar as capacidades operacionais no setor logístico.

CAPÍTULO III – APRESENTAÇÃO DA ENTIDADE DE ACOLHIMENTO

3 Luís Simões - Logística Integrada

A Luís Simões Logística Integrada, SA destaca-se como um operador logístico de referência, líder no mercado de fluxos rodoviários entre Portugal e Espanha. Com uma presença consolidada há mais de 30 anos no mercado espanhol, a empresa iniciou a sua atividade em Loures, em 1948, e desde então tem vindo a expandir-se de forma sustentável, acompanhando as exigências do setor e as necessidades dos seus clientes.

Atualmente, a Luís Simões gere uma frota de 1.621 viaturas, entre veículos próprios e subcontratados, e conta com mais de 2.600 colaboradores diretos, o que evidencia a sua dimensão e capacidade de resposta. A sua operação abrange toda a Península Ibérica, com mais de 25 armazéns distribuídos por 10 regiões, ultrapassando os 400.000 m² de capacidade instalada. Estes números traduzem-se numa média mensal de cerca de 7,9 milhões de unidades de *picking*, 4,5 milhões de unidades de *copacking* e 1.744 rotas de distribuição diárias, transportando anualmente cerca de 8 milhões de toneladas e percorrendo mais de 100 milhões de quilómetros.

3.1 Missão, Visão e Política

A missão da Luís Simões consiste em garantir soluções eficientes e competitivas de transportes, logística e serviços auxiliares, promovendo a satisfação dos clientes e da sociedade em geral, sob os pontos de vista económico, social e ambiental. A visão da empresa é ser a referência ibérica em termos de qualidade de serviço no setor dos transportes e da logística, apostando na inovação, sustentabilidade e excelência operacional.

A política da empresa assenta no cumprimento rigoroso dos requisitos legais e regulamentares, bem como na promoção da sensibilização e envolvimento dos colaboradores na implementação das políticas da LS. A Luís Simões compromete-se ainda a desenvolver as competências dos seus colaboradores, disponibilizando os recursos necessários à melhoria contínua dos processos, com especial foco na qualidade, segurança alimentar, ambiente, segurança e saúde no trabalho e responsabilidade social.

3.2 Valores e Compromissos

A atuação da Luís Simões é orientada por valores fundamentais como a orientação para o cliente, respeito pelas pessoas, sustentabilidade, confiança e lealdade. Estes valores refletem-se na aposta contínua da qualificação dos colaboradores, na promoção de práticas sustentáveis e responsáveis, e na transparência das relações com clientes, fornecedores e restantes partes interessadas.

A empresa privilegia a eficiência e eficácia dos processos, procurando sempre superar as expectativas dos clientes através de soluções inovadoras e tecnologicamente avançadas. Ao nível ambiental, a Luís Simões implementa medidas para minimizar os impactos das suas atividades, promovendo a redução de resíduos, emissões e consumo de energia. No domínio da segurança e saúde no trabalho, garante condições adequadas para a prevenção de riscos profissionais e o bem-estar dos seus colaboradores. Por fim, assume um compromisso ativo com a responsabilidade social, desenvolvendo ações que visam a valorização pessoal, profissional e familiar, bem como iniciativas de solidariedade junto da comunidade.

3.3 Serviços da Luís Simões

A Luís Simões oferece uma vasta gama de serviços integrados no setor da logística e dos transportes, posicionando-se como um operador de referência na Península Ibérica. Estes serviços abrangem diferentes áreas de atividade, desde a logística tradicional ao transporte internacional, passando por soluções promocionais, aluguer e manutenção de veículos pesados, seguros e parcerias estratégicas com transportadores.

3.3.1 Logística Integrada

A Luís Simões disponibiliza soluções logísticas inovadoras, com integração e verticalização de serviços à escala ibérica, suportadas por uma rede de Centros de Operações Logísticas e Plataformas Regionais. Destacam-se:

- **Gestão de Armazenamento:** Armazenamento de produtos alimentares e não alimentares em diferentes condições de temperatura (ambiente controlado e refrigerado), adaptando-se às necessidades específicas de cada cliente.
- ***E-commerce*:** Soluções ajustadas ao canal *B2C* (*Business-to-Consumer*), com processos de *pick & pack* diferenciados e embalamento adaptado, ideal para setores como alimentação, perfumaria e cosmética.

- **Logística Inversa:** Gestão integral de recusas, devoluções e levantamentos em destinatários, assegurando a eficiência nas cadeias de abastecimento.
- **Gestão de Operações In-House:** Operação em armazéns de matéria-prima e produto acabado, tanto em instalações próprias como nas dos clientes.
- **Gestão de Inventários:** Elevados níveis de precisão de stock (cerca de 99,95%), com controlo rigoroso em todas as fases do processo logístico.
- ***Picking & Packaging*:** Preparação de encomendas e soluções de *packaging* adaptadas ao canal de distribuição, com tecnologias como *pick & put-to-light* para maior produtividade e redução de erros.
- **Rastreabilidade:** Monitorização dos produtos ao longo de toda a cadeia logística, desde a receção até ao destino final.
- **Entrepostos Fiscais e Aduaneiros:** Gestão de produtos sujeitos a impostos especiais e sob regime de suspensão de imposto, incluindo operações de carga, descarga e emissão de declarações de trânsito.

3.3.2 Transporte

O serviço de transporte da Luís Simões é um dos pilares da sua oferta, destacando-se pela flexibilidade, fiabilidade e cobertura ibérica e internacional:

- **Transporte Rodoviário Nacional e Internacional:** Operações de carga completa (*FTL – Full Truckload*) e parcial (*LTL – Less-than-Truckload*), com especial enfoque nos mercados ibéricos, França, Alemanha, Itália, Benelux e Reino Unido.
- **Distribuição Ibérica:** Serviço multi-cliente e multi-temperatura, consolidando volumes de diferentes origens para otimizar entregas e sinergias.
- **Veículos Dedicados:** Soluções de transporte dedicadas, adaptadas às necessidades específicas de cada cliente, incluindo opções multi-temperatura.
- **Transporte Intermodal (*Short Sea Shipping*):** Alternativa sustentável ao transporte rodoviário convencional, com rotas marítimas de curta distância para o norte da Europa.
- ***Gigaliner*:** Transporte de grandes volumes e pesos, ideal para integração vertical de processos industriais e distribuição.

3.3.3 Logística Promocional

A Luís Simões apoia as marcas na execução de campanhas promocionais e comerciais, oferecendo:

- ***Co-packing*:** Produção de packs promocionais, etiquetagem, embalagem e reembalagem, com recurso a equipamentos de alto débito.
- **Gestão de Eventos:** Transporte e montagem de materiais promocionais em eventos e pontos de venda.
- **Implementações em Loja:** Soluções chave-na-mão, desde o design até à instalação de expositores.
- **Gestão de Consumíveis e Material Promocional:** Seleção, aquisição e gestão de materiais para campanhas.
- **Reparação de Expositores:** Reutilização e manutenção de materiais promocionais, promovendo a sustentabilidade.

3.3.4 RETA – Aluguer, Venda e Manutenção de Pesados

A RETA, integrada no grupo Luís Simões, funciona como um conceito "*one-stop-shop*" para empresas de transporte:

- **Venda e Aluguer de Pesados:** Frota de semirreboques e tratores novos e usados, com opções flexíveis de aluguer.
- **Serviços de Oficina:** Manutenção e reparação multimarca, abrangendo todas as áreas técnicas dos veículos.
- **Loja de Peças:** Oferta de peças e equipamentos para manutenção e segurança de viaturas comerciais.

3.3.5 Diagonal Seguros

A Luís Simões disponibiliza soluções de seguros personalizadas para particulares e empresas:

- **Seguros Particulares:** Proteção pessoal e patrimonial, com análise das melhores opções do mercado.
- **Seguros Empresariais:** Cobertura de riscos associados a colaboradores, infraestruturas e equipamentos, protegendo a atividade principal das empresas.

3.3.6 Parcerias com Transportadores

A empresa mantém uma rede de mais de 700 parceiros transportadores em toda a Península Ibérica, oferecendo:

1. **Ocupação Permanente dos Veículos:** Garantia de carga regular e pagamento seguro.
2. **Vantagens Operacionais:** Descontos em combustíveis, acesso a postos de abastecimento, cartão de combustível sem taxas e acompanhamento personalizado.
3. **Portal LS:** Plataforma digital para gestão e acompanhamento dos serviços e faturação.

3.3.7 Direção de Dados

- Como e quando surgiu?

A Direção de Dados foi formalmente estabelecida no início de 2025, como um resultado direto da evolução do projeto de *reporting* iniciado cerca de quatro anos antes. Este projeto nasceu com o propósito de criar uma fonte única e confiável de dados, consolidando toda a informação relevante para o negócio. Num primeiro momento, foram identificados os indicadores essenciais e definidas as prioridades para o desenvolvimento de relatórios. Com o passar do tempo, a uniformização e partilha contínua dos dados demonstraram a sua importância estratégica, levando à criação desta direção.

- Independência da Direção de Dados:

A independência da Direção de Dados é crucial para garantir a imparcialidade e a objetividade na gestão e análise da informação. Ao operar de forma autónoma, esta direção pode evitar influências externas ou conflitos de interesses que possam comprometer a integridade dos dados. Além disso, a independência permite uma visão holística e neutra sobre as operações e estratégias da empresa, assegurando que todas as áreas de negócio têm acesso a *insights* confiáveis e consistentes. Isto fomenta a transparência e a confiança nas decisões baseadas em dados, que são cada vez mais centrais para o sucesso organizacional.

- Objetivos estratégicos:

Os objetivos estratégicos da Direção de Dados devem ser amplamente orientados para maximizar o valor da informação no suporte às decisões organizacionais. Entre os principais objetivos destacam-se:

1. Governança de dados: Implementar políticas robustas para gestão e proteção de dados, garantindo conformidade com normas e regulamentações.
2. Qualidade da informação: Assegurar a integridade, precisão e consistência dos dados, criando uma base sólida para análise e *reporting*.
3. Inovação e automação: Adotar tecnologias avançadas para melhorar a acessibilidade e eficiência na manipulação de dados, incluindo inteligência artificial e *machine learning*.
4. Tomada de decisão informada: Promover o uso de dados como ferramenta central para decisões estratégicas em todas as áreas de negócio.
5. Transparência e acessibilidade: Facilitar a partilha de dados de forma transparente entre departamentos, garantindo que todos têm acesso aos *insights* necessários.
6. Suporte ao crescimento: Identificar tendências e oportunidades através de análises preditivas, contribuindo para o desenvolvimento e adaptação da empresa no mercado.

Esses objetivos refletem a centralidade dos dados na transformação digital e na competitividade organizacional, permitindo à Direção de Dados consolidar-se como um pilar estratégico para o crescimento sustentável.

- Impacto interno:

A informação, embora seja gerida pela Direção de Dados, pertence a toda a organização e deve ser acessível a todos. Temos garantido a qualidade, a acessibilidade e a confiabilidade dos dados, utilizando ferramentas como o Power BI para permitir decisões mais informadas e estratégicas. O impacto deste trabalho é evidente no crescente interesse e procura por parte de toda a organização, que reconhece o valor de decisões fundamentadas em dados. Este esforço tem consolidado a Direção de Dados como um pilar essencial na transformação digital e na evolução da LS rumo a um futuro mais competitivo e sustentável.

CAPÍTULO IV – ATIVIDADES REALIZADAS NO ESTÁGIO

Durante o meu estágio na Luís Simões, trabalhei em regime híbrido, alternando quatro dias de teletrabalho com um dia presencial no escritório da Rechousa, em Gaia. Tanto a entrevista de admissão como o processo de integração na equipa foram realizados através de videoconferência. Todas as semanas estive presente nas reuniões de Ponto de Situação, que incluíam todos os colaboradores da direção de dados, constituída por um diretor de dados, um engenheiro de dados e três analistas, para dar seguimento aos projetos que cada um era responsável, individuais ou transversais.

4 Atividades Realizadas no Estágio

Ao longo do mês de março, participei em diversas atividades de integração e formação, incluindo a apresentação à equipa, acolhimento institucional, demonstração dos principais negócios da empresa e do projeto de *Reporting*. Assisti à primeira reunião com a Direção de Dados, onde pude sugerir ideias para a melhoria dos relatórios existentes. Tive acesso a relatórios operacionais para análise e previsão, e fui contextualizado relativamente às funções do departamento. Realizei um *breakdown* de produção de um relatório, análise exploratória de dados e investigação de algoritmos preditivos para estimar a quantidade de caixas a movimentar nos armazéns de Gaia e Carregado. Testei diferentes modelos de previsão (*AutoArima*, *Prophet*, *SARIMA*, *TimeGPT*), incluindo a introdução de variáveis externas como feriados nacionais, tendo optado pelo modelo *Prophet* devido aos melhores resultados obtidos. No final do mês, as previsões para a semana seguinte estavam concluídas para ambos os armazéns.

Em abril, foquei-me na previsão da quantidade de expedições e participei na apresentação do símbolo do departamento, contribuindo com sugestões de melhoria. Concluí a previsão de expedições e aprofundei a análise da previsão de caixas, recorrendo a técnicas de *Hierarchical Forecasting* com o *Prophet* para múltiplas séries temporais num mesmo armazém. Iniciei a implementação dos modelos desenvolvidos na plataforma *Fabric*, plataforma unificada de Análise de Dados integra todo o ciclo de trabalho, da coleta e processamento de dados à criação de relatórios. Reúne ferramentas como *Data Engineering*, *Data Factory*, *Data Science*, *Real Time Intelligence* e *Data Warehouse* para simplificar a gestão e visualização de dados em tempo real na Luís Simões. Paralelamente, iniciei a elaboração do relatório de estágio e realizei investigação de artigos científicos para fundamentação teórica, análise de resultados e discussão, terminando o mês com a revisão técnica e final do relatório.

Para aplicação prática do sistema preditivo, o Cliente X foi selecionado devido à sua importância estratégica, cobertura nacional, frequência elevada de encomendas e histórico de dados consistente. A sua representatividade operacional torna-o ideal para validar este tipo de análise.

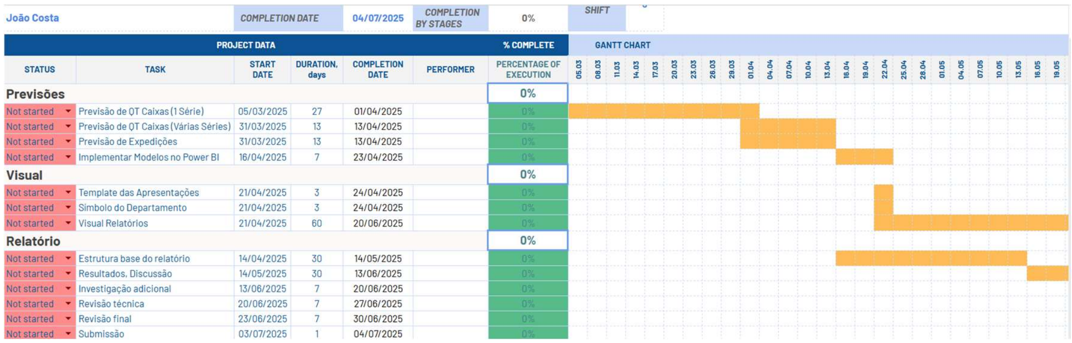
A inexistência de previsões estruturadas para este cliente tem originado desafios, como sobrecargas logísticas e necessidade de ajustes de última hora na alocação de recursos. A adoção de um modelo preditivo fiável permite antecipar necessidades, otimizar recursos e responder de forma mais eficaz a variações de procura e sazonalidade. Este sistema contribui assim para operações mais estáveis, controlo de custos e aumento da capacidade de resposta, reforçando a competitividade da empresa e a satisfação dos seus clientes.

Com o objetivo de organizar e dar seguimento às minhas tarefas, elaborei um calendário que incluía as datas de início e de conclusão do estágio, bem como a distribuição das tarefas ao longo do tempo e por etapas. Estruturei as tarefas em três categorias principais: Previsões, Visual e Relatório.

Na primeira coluna do calendário, classifiquei o estado de cada tarefa com os seguintes indicadores: *Not Started*, *In Process*, *Completed* e *Suspended*. A segunda coluna, intitulada Task, descrevia as tarefas a realizar. As três colunas seguintes – *Start Date*, Duration (*days*) e *Completion Date* – indicavam, respetivamente, a data de início, a duração prevista (em dias) e a data de conclusão de cada tarefa.

A sexta coluna, designada *Percentage of Execution*, registava o progresso de cada tarefa em percentagem, sendo atribuídos 0% para *Not Started* e *Suspended*, 25% para *In Process* e 100% para *Completed*.

À direita destas colunas, elaborei um diagrama de *Gantt* que permitia uma visualização mais dinâmica e intuitiva da distribuição temporal das tarefas.



João Costa		COMPLETION DATE		04/07/2025	COMPLETION BY STAGES	0%
PROJECT DATA						% COMPLETE
STATUS	TASK	START DATE	DURATION, days	COMPLETION DATE	PERFORMER	PERCENTAGE OF EXECUTION
Previsões						0%
Not started	Previsão de QT Caixas (1 Série)	05/03/2025	27	01/04/2025		0%
Not started	Previsão de QT Caixas (Várias Séries)	31/03/2025	13	13/04/2025		0%
Not started	Previsão de Expedições	31/03/2025	13	13/04/2025		0%
Not started	Implementar Modelos no Power BI	16/04/2025	7	23/04/2025		0%
Visual						0%
Not started	Template das Apresentações	21/04/2025	3	24/04/2025		0%
Not started	Símbolo do Departamento	21/04/2025	3	24/04/2025		0%
Not started	Visual Relatórios	21/04/2025	60	20/06/2025		0%
Relatório						0%
Not started	Estrutura base do relatório	14/04/2025	30	14/05/2025		0%
Not started	Resultados, Discussão	14/05/2025	30	13/06/2025		0%
Not started	Investigação adicional	13/06/2025	7	20/06/2025		0%
Not started	Revisão técnica	20/06/2025	7	27/06/2025		0%
Not started	Revisão final	23/06/2025	7	30/06/2025		0%
Not started	Submissão	03/07/2025	1	04/07/2025		0%

Figura 1 - Calendário de Tarefas de Estágio

4.1 Tarefas:

Tarefa 1 - Previsão de Série Temporal Única: Realizar a previsão da quantidade de caixas que serão produzidas para a semana seguinte, numa determinada operação, num determinado armazém, para um cliente específico.

1. Exploração Inicial dos Dados

Importação dos dados históricos a partir de um arquivo .xlsx.

- 2589 registos e 10 colunas.

- Verificação inicial das variáveis e tipos.

2. Pré-processamento dos Dados

Limpeza, conversão e tratamento de dados:

- Remoção de colunas irrelevantes.

- Conversão de tipos: int, datetime.

- Preenchimento de nulos com 0.

- Remoção de datas inválidas (1970-01-01).

3. Filtragem e Agregação

Filtragem por operação 'A' e agrupamento semanal:

- Seleção da operação 'A'.

- Remoção de colunas redundantes.

- Agregação por semana com resample('W-SUN').

4. Análise Exploratória

Visualização e testes iniciais:

- Plotagem da série temporal.

- Média móvel e desvio padrão.

- Teste de Dickey-Fuller Aumentado (ADF).

5. Transformações para Estacionaridade

Estabilização da série temporal:

- Transformação logarítmica (Box-Cox).

- Subtração de média móvel.

- Suavização exponencial (ewm).

6. Decomposição da Série

Separar componentes da série:

- Extração de tendência, sazonalidade e resíduos.

- Avaliação se resíduos são ruído branco.

7. Análise dos Resíduos

Verificação de ruído e autocorrelação:

- Teste de Ljung-Box.

- Análise ACF e PACF.

8. Modelagem com *Prophet*

Previsão com modelo *Prophet*:

- Uso de sazonalidades semanais e anuais.

- Geração da previsão (yhat) para próxima semana.

Pipeline de Análise de Séries Temporais

9. Avaliação da Previsão

Comparação entre previsão e valor real:

- Cálculo do erro absoluto.

- Cálculo do MAPE (Erro Percentual Absoluto Médio).

Tarefa 2 - Previsão de Múltiplas Séries Temporais: Realizar a previsão da quantidade de caixas que serão produzidas para a semana seguinte, numa determinada operação, num determinado armazém, para um cliente específico.

1. Exploração Inicial dos Dados

Importação dos dados históricos a partir de um arquivo .xlsx.

- 2589 registos e 10 colunas.

- Verificação inicial das variáveis e tipos.

2. Pré-processamento dos Dados

Limpeza, conversão e tratamento de dados:

- Remoção de colunas irrelevantes.

- Conversão de tipos: int, datetime.

- Preenchimento de nulos com 0.

- Remoção de datas inválidas (1970-01-01).

3. Filtragem e Agregação

Filtragem por operação 'A' e agrupamento semanal:

- Seleção da operação 'A'.

- Remoção de colunas redundantes.

- Agregação por semana com resample('W-SUN').

4. Análise Exploratória

Visualização e testes iniciais:

- Plotagem da série temporal.

- Média móvel e desvio padrão.

- Teste de Dickey-Fuller Aumentado (ADF).

5. Transformações para Estacionaridade

Estabilização da série temporal:

- Transformação logarítmica (Box-Cox).

- Subtração de média móvel.

- Suavização exponencial (ewm).

6. Decomposição da Série

Separar componentes da série:

- Extração de tendência, sazonalidade e resíduos.

- Avaliação se resíduos são ruído branco.

7. Análise dos Resíduos

Verificação de ruído e autocorrelação:

- Teste de Ljung-Box.

- Análise ACF e PACF.

8. Modelagem com *Prophet*

Previsão com modelo *Prophet*:

- Uso de sazonalidades semanais e anuais.

- Geração da previsão (yhat) para próxima semana.

Pipeline de Análise de Séries Temporais

9. Avaliação da Previsão

Comparação entre previsão e valor real:

- Cálculo do erro absoluto.

- Cálculo do MAPE (Erro Percentual Absoluto Médio).

Tarefa 3: Realizar a previsão da quantidade de expedições que serão produzidas para a semana seguinte, para um cliente específico.

1. Importação de Bibliotecas

Carregamento dos pacotes necessários:

- pandas, numpy: manipulação de dados
- matplotlib, seaborn: visualização gráfica
- statsmodels, scipy: modelagem de séries temporais
- sklearn: métricas de avaliação
- Supressão de warnings
- Ativação de gráficos inline (Colab).

2. Importação dos Dados

Leitura do ficheiro de dados a partir do *Google Drive*:

- Arquivo: data3.xlsx
- Contém registos históricos de expedições
- Armazenamento inicial no DataFrame df_final.

3. Visualização e Preparação Inicial

Exploração da estrutura dos dados:

- Exibição das primeiras linhas
- Listagem das colunas disponíveis
- Compreensão geral do conteúdo.

4. Limpeza e Pré-processamento

Padronização e tratamento dos dados:

- Remoção de colunas irrelevantes

- Preenchimento de valores ausentes com 0

- Conversão de colunas para tipo inteiro.

5. Seleção de Cliente e Filtragem

Foco em um único cliente para análise:

- Filtragem do cliente com ID Y

- Isolamento da sua série temporal.

6. Indexação Temporal e Agregação

Conversão e organização por tempo:

- Conversão da coluna DATA para datetime

- Definição como índice

- Agregação semanal por soma de expedições.

7. Visualização da Série Temporal

Análise visual da série histórica:

- Gráfico de expedições semanais

- Identificação de tendências e padrões sazonais.

8. Decomposição da Série

Separação dos componentes da série:

- Tendência, sazonalidade e resíduos

- Auxilia na compreensão da estrutura.

9. Teste de Estacionariedade (ADF)

Verificação da natureza da série:

- Aplicação do teste Dickey-Fuller Aumentado

- Determinação da necessidade de diferenciação

10. Análise ACF e PACF

Estudo de autocorrelações:

- Geração dos gráficos ACF e PACF

- Apoio na escolha de parâmetros do modelo.

11. Modelagem ARIMA

Construção do modelo de previsão:

- Modelo ARIMA(1,1,1)

- Série diferenciada para garantir estacionariedade.

12. Previsão da Semana Seguinte

Geração da estimativa de expedições:

- Previsão para a próxima semana

- Baseada no modelo ajustado para o cliente Y

13. Avaliação da Previsão

Comparação entre previsão e valor real:

- Cálculo do erro absoluto.

- Cálculo do MAPE (Erro Percentual Absoluto Médio).

Tarefa 4: Realizar a previsão da quantidade de caixas que serão produzidas para a semana seguinte, numa determinada operação, num determinado armazém, para um cliente específico, com base em várias séries.

1. Exploração Inicial dos Dados

- Leitura de dados Excel

- Visualização de colunas e registos

- Importação de bibliotecas essenciais

2. Pré-processamento dos Dados

- Remoção de colunas irrelevantes

- Tratamento de valores nulos com 0

- Conversão de colunas para tipos adequados

- Conversão da data para tipo datetime

3. Filtragem e Agregação

- Reamostragem semanal dos dados por cliente (soma de caixas)

- Geração de matriz DataFrame com datas x clientes

4. Análise Exploratória

- Criação de séries temporais por cliente

- Visualização com gráficos

- Cálculo de média móvel e desvio padrão

5. Transformações para Estacionaridade

- Aplicação de log nas séries dos clientes para suavização

- Conversão de dados para formato longo (long format), preparando para modelagem hierárquica

6. Decomposição da Série

- Construção da matriz de soma representando a hierarquia

- Separação da série total e séries individuais por cliente para análise independente

7. Análise dos Resíduos

- Comparação entre previsão total direta e soma das previsões individuais (Reconciliation: *Bottom-Up*)

- Avaliação da consistência entre níveis da hierarquia

8. Modelagem com *Prophet*

- Previsão *Top Level*: soma total de caixas por semana

  - Treino do modelo *Prophet* no total
  - Previsão da próxima semana
  - Cálculo de erro e MAPE

- Previsão Bottom Level:

  - *Loop* para treinar *Prophet* para cada cliente
  - Previsão semanal e reversão do log

- Previsão Hierárquica:

- Previsão com *Prophet* por ID único

9. Avaliação da Previsão

- Exemplo de previsão para cliente X

- Visualização em gráfico

Tarefa 5: Implementação do modelo *Prophet* no *Fabric*

1. Inicialização do Ambiente

- Utiliza SparkSession para iniciar o processamento distribuído de dados.

2. Acesso aos Dados no Delta Lake

- Conecta-se ao Lakehouse e converte os dados da tabela Delta para um DataFrame Pandas.

3. Exploração Inicial dos Dados

- Exibe as primeiras linhas, tipos de dados, estatísticas descritivas e valores únicos.

4. Limpeza e Seleção de Colunas

- Remove colunas irrelevantes e mantém apenas aquelas essenciais para a análise.

5. Remoção de Registos Inválidos

- Elimina registos com datas inválidas, como "1970-01-01", e ordena os dados.

6. Filtragem por Armazém

- Seleciona apenas os registos em que cod_col_sid é igual a 'A'.

7. Análise de Média e Desvio

- Aplica uma média móvel e o desvio padrão com janela de 12 semanas para identificar tendências.

8. Agregação por Semana

- Agrupa os dados semanalmente, somando os valores da coluna "caixas".

9. Formatação para o *Prophet*

- Renomeia as colunas para os nomes exigidos pelo modelo *Prophet*: "ds" (datas) e "y" (valores).

10. Criação e Ajuste do Modelo *Prophet*

- Treina o modelo *Prophet*, configurando sazonalidade semanal e anual.

11. Previsão da Próxima Semana

- Gera a previsão para o próximo domingo utilizando o modelo ajustado.

12. Avaliação da Previsão

Comparação entre previsão e valor real:

- Cálculo do erro absoluto.

- Cálculo do MAPE (Erro Percentual Absoluto Médio).

5 Visualização e Análise de Resultados

O modelo *Prophet* apresentou um desempenho superior ao SARIMA na previsão da quantidade semanal de caixas produzidas, sobretudo devido à sua capacidade de capturar de forma mais robusta as características intrínsecas dos dados.

Ao contrário do SARIMA, que pressupõe linearidade nas tendências e exige transformações rigorosas para alcançar estacionariedade, o *Prophet* é inerentemente flexível para modelizar tendências não lineares e múltiplas sazonalidades – como as variações semanais observadas na produção logística. Além disso, o *Prophet* lida de forma nativa com *outliers* e mudanças estruturais abruptas na série temporal, fenómenos comuns em ambientes operacionais sujeitos a interrupções, picos de procura ou falhas de equipamento. A sua arquitetura baseada em modelos aditivos permite a inclusão explícita de feriados, eventos sazonais e fatores externos (como campanhas promocionais ou paragens programadas), que impactam diretamente a produção e são difíceis de incorporar de forma eficaz no SARIMA sem ajustes manuais complexos.

Outro ponto crítico é a automação da deteção de pontos de mudança (*changepoints*), que o *Prophet* realiza de forma adaptativa, ajustando automaticamente a taxa de variação da tendência em momentos de rutura estrutural. Em contrapartida, o SARIMA exige parametrização rígida (p, d, q) e (P, D, Q) com testes estatísticos manuais, sendo altamente sensível a violações de pressupostos.

5.1 Previsão de Produção – Março/Abril 2025

As previsões semanais de produção, geradas com base no modelo *Prophet*, demonstraram níveis de precisão consistentemente elevados, com desvios mínimos entre os valores previstos e os realizados. A análise comparativa entre as abordagens de série única e múltiplas séries revela *insights* importantes sobre a robustez do modelo em diferentes configurações.

5.1.1 Precisão Semanal – Unidade A

A Figura 2 apresenta a precisão semanal da previsão de produção na Unidade A ao longo de quatro semanas (10 de março a 6 de abril de 2025), comparando duas estratégias de modelação.

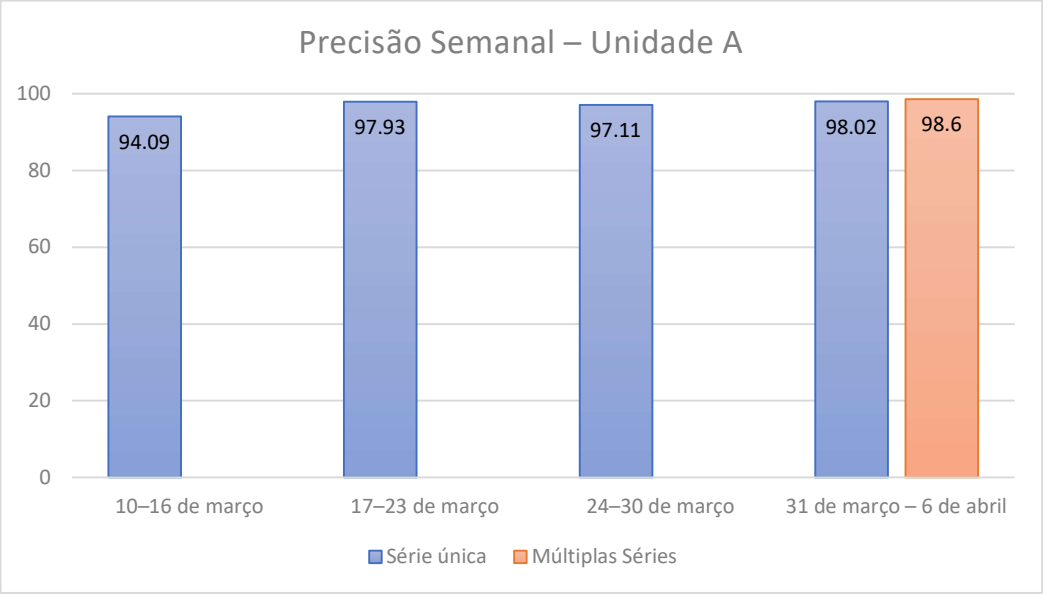


Figura 2 - Precisão Semanal – Unidade A

A abordagem de série única (em azul) manteve precisão acima de 94% em todas as semanas, com um pico de 98.02% na última semana. Este desempenho reflete a capacidade do *Prophet* em capturar a sazonalidade semanal e a tendência de crescimento observada na produção da Unidade A, mesmo com variações moderadas no volume de caixas.

A inclusão da abordagem de múltiplas séries (em laranja) apenas na semana de 31 de março a 6 de abril, com precisão de 98.60%, sugere uma melhoria marginal (+0.58 pontos percentuais) quando se considera a interação com outras unidades ou variáveis exógenas (e.g., expedições, stock intermédio). Esta diferença, embora pequena, é estatisticamente relevante e indica que a modelação hierárquica ou multivariada pode refinar ainda mais as previsões em períodos de maior complexidade operacional.

A consistência da precisão acima de 97% nas semanas centrais (17–30 de março) corrobora a estabilidade do modelo face a ruídos operacionais, enquanto o aumento progressivo até 98.6% na última semana pode estar associado a uma redução de incerteza (e.g., menor variabilidade na programação ou maior disponibilidade de recursos).

5.1.2 Precisão Semanal – Unidade B

A Figura 3 ilustra o desempenho preditivo na Unidade B, utilizando exclusivamente a abordagem de série única.

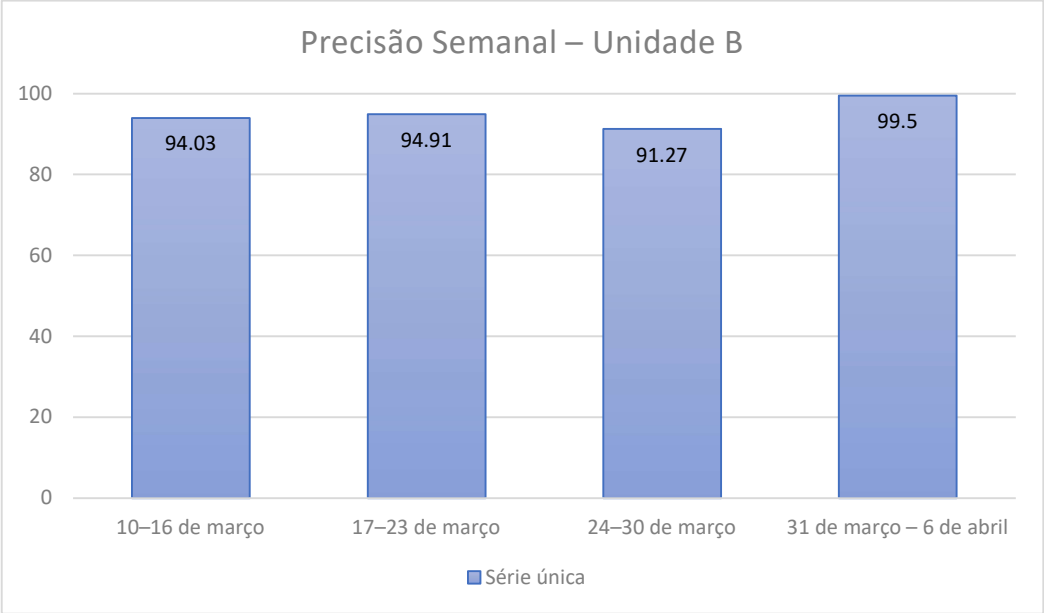


Figura 3 - Precisão Semanal – Unidade B

Observa-se um padrão de variabilidade mais acentuado do que na Unidade A. As duas primeiras semanas apresentam precisão próxima de 94%, indicando bom alinhamento entre previsão e realidade. Contudo, a semana de 24–30 de março regista uma queda abrupta para 91.27%, sugerindo a presença de um evento disruptivo não capturado pelo modelo – possivelmente uma paragem não programada, falha de fornecedor ou pico de absentismo.

Este *outlier* é compensado de forma notável na semana seguinte, com uma subida da precisão para 99.50%, o valor mais elevado de toda a análise. Tal recuperação sugere que o *Prophet* ajustou dinamicamente os seus parâmetros internos (especialmente a flexibilidade da tendência e o peso dos *changepoints*) para acomodar a perturbação anterior, demonstrando resiliência adaptativa. Este comportamento reforça a vantagem do *Prophet* em contextos operacionais voláteis, onde o SARIMA tenderia a propagar o erro.

5.2 Previsão de Expedições – Março/Abril 2025

As previsões semanais de expedições, igualmente baseadas no *Prophet*, apresentaram uma excelente correspondência com os valores reais, com precisão média superior a 96%. A análise focou-se no Cliente Y, um destino crítico em termos de volume e exigência de serviço.

5.2.1 Precisão Semanal – Cliente Y

A Figura 4 revela um padrão de alta precisão com duas oscilações significativas. As semanas de 10–16 de março e 24–30 de março registaram precisão próxima da perfeição (98.73% e 99.77%, respetivamente), indicando que o modelo capturou com exatidão os padrões de expedição neste cliente – possivelmente devido a uma programação estável ou a uma sazonalidade bem definida.

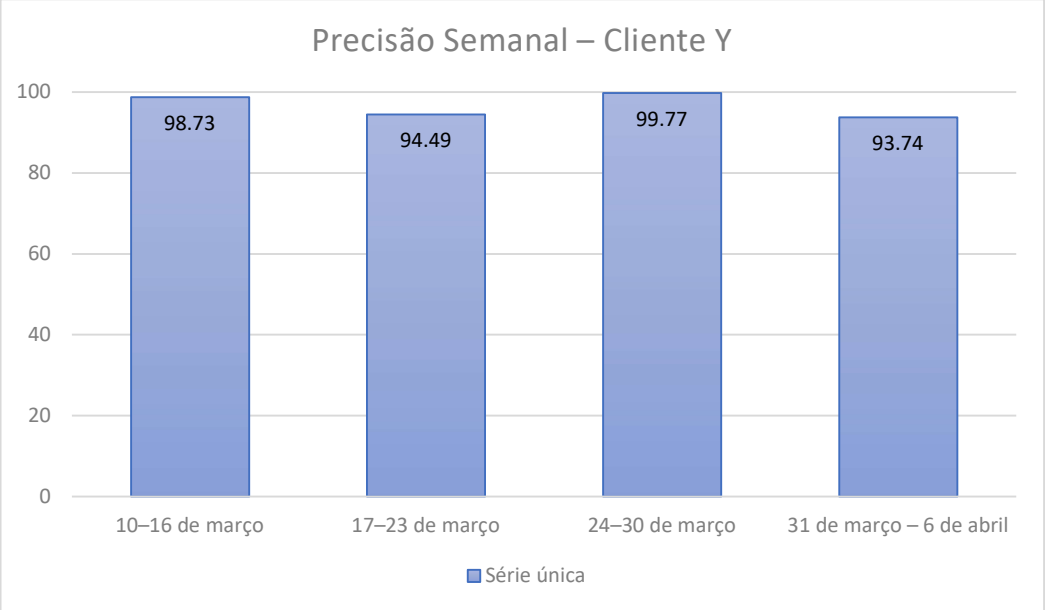


Figura 4 - Precisão Semanal – Cliente Y

Em contrapartida, as semanas de 17–23 de março (94.49%) e 31 de março–6 de abril (93.74%) mostram quedas de cerca de 5 pontos percentuais, coincidentes com períodos de potencial congestão logística ou alterações na prioridade de entrega. A proximidade temporal entre estas duas quedas (separadas por uma semana de excelência) sugere a presença de efeitos de compensação – por exemplo, adiantamento de expedições numa semana para cumprir prazos na seguinte.

A capacidade do *Prophet* em manter a precisão média acima de 96% mesmo com estas flutuações demonstra a sua eficácia em prever não apenas volumes, mas também padrões de variabilidade na expedição, um aspeto crítico para a gestão de stocks e planeamento de transporte.

5.3 Discussão Integrada

A análise conjunta das Figuras 2, 3 e 4 permite extrair três conclusões estratégicas:

1- Robustez do *Prophet* em múltiplos contextos: O modelo manteve precisão média superior a 95% em produção (Unidades A e B) e expedições (Cliente Y), mesmo perante disrupções localizadas (e.g., 91.27% na Unidade B).
2- Vantagem da modelação multivariada: A melhoria observada com múltiplas séries na Unidade A (98.60% vs. 98.02%) sugere que a integração de variáveis exógenas (e.g., expedições, stock, feriados) pode elevar a precisão em cenários complexos.
3- Identificação de pontos de melhoria operacional: As quedas de precisão (91.27% na Unidade B; 93.74% no Cliente Y) funcionam como sinais *early warning* de ineficiências logísticas, permitindo intervenções proativas (e.g., reforço de capacidade, revisão de contratos de transporte).

Futuras iterações do modelo poderão beneficiar da inclusão de variáveis explicativas adicionais (e.g., índices de ocupação de armazém, taxas de rejeição, condições climatéricas) e da aplicação de técnicas de reconciliação hierárquica para alinhar previsões de produção e expedição em tempo real.

6 Conclusão

O estágio curricular realizado na Luís Simões Logística Integrada constituiu uma etapa fundamental na minha formação académica e profissional. Ao longo deste período, tive a oportunidade de aplicar, em contexto real de negócio, os conhecimentos adquiridos ao longo do curso, especialmente na área de *Business Intelligence and Analytics*. A experiência permitiu-me não só consolidar competências técnicas, como também desenvolver capacidades analíticas, pensamento crítico e comunicação em equipa.

As soluções implementadas, nomeadamente a aplicação de modelos preditivos com recurso a ferramentas como *Prophet* e *Microsoft Fabric*, trouxeram melhorias significativas aos processos de análise de dados da organização, promovendo uma tomada de decisão mais informada e uma maior eficiência operacional. A automatização de tarefas revelou-se essencial para uma gestão mais eficaz dos recursos disponíveis.

O envolvimento direto em reuniões da Direção de Dados, a participação ativa no desenvolvimento de projetos preditivos e a análise prática de dados reais permitiram-me compreender os desafios do setor logístico e contribuir para soluções com impacto operacional. Este contacto com a realidade empresarial foi enriquecido por um ambiente de trabalho colaborativo e por uma equipa sempre disponível para apoiar e partilhar conhecimento.

Para além dos resultados alcançados, foi possível identificar oportunidades de melhoria no modelo de previsão. Entre estas, destaca-se a possibilidade de aperfeiçoar a granularidade temporal das previsões, passando de previsões semanais para previsões diárias, de forma a aumentar a capacidade de resposta às variações da procura. Outra vertente promissora seria explorar o impacto de fatores externos, como as condições meteorológicas, em particular em períodos de maior sensibilidade logística, permitindo assim ajustar as previsões com base em variáveis contextuais e melhorar a previsão do modelo. Estas melhorias poderiam contribuir para uma gestão mais proativa e adaptativa dos fluxos operacionais, reforçando o valor acrescentado da análise preditiva no processo de decisão.

Em suma, este estágio foi determinante para a consolidação do meu perfil profissional na área de dados e *analytics*, proporcionando-me uma base sólida para enfrentar os desafios do mercado de trabalho e para continuar a evoluir como profissional neste setor em constante transformação.

Abdullahi, M., Aimufua, G., & Muhammad, U. (2021). Application of sales forecasting model based on machine learning algorithms. *AIMS iSTEAMS Proceedings, 28*, 205–216. https://doi.org/10.22624/AIMS/ISTEAMS-2021/V28P17

Abolghasemi, M., Hyndman, R., Spiliotis, E., & Bergmeir, C. (2020). Model selection in reconciling hierarchical time series. *arXiv*. https://doi.org/10.48550/arXiv.2010.10742

Alencar, V., Pessamilio, L., Rooke, F., Bernardino, H., & Vieira, A. (2021). Forecasting the carsharing service demand using uni and multivariable models. *Journal of Internet Services and Applications, 12*(1), Article 8. https://doi.org/10.1186/s13174-021-00137-8

Aoki, T., Takadama, K., & Satō, H. (2021). Adaptive synapse arrangement in cortical learning algorithm. *Journal of Advanced Computational Intelligence and Intelligent Informatics, 25*(4), 450–466. https://doi.org/10.20965/jaciii.2021.p0450

Arika, A., Daratullaila, D., Sirait, K., & Sari, R. (2024). Fuzzy time series dalam meramalkan jumlah produksi karet di Sumatra Utara. *Majalah Ilmiah Matematika dan Statistika, 24*(1), 39–50. https://doi.org/10.19184/mims.v24i1.35257

Burba, D., & Chen, T. (2021). A trainable reconciliation method for hierarchical time-series. *arXiv*. https://doi.org/10.48550/arXiv.2101.01329

Cardona, J. (2021). Using data analytics & machine learning to design business interruption insurance products for rail freight operators. *SSRN Electronic Journal*. https://doi.org/10.36443/10259/6876

Chen, Z., Wang, C., Lv, L., Fan, L., Wen, S., & Xiang, Z. (2023). Research on peak load prediction of distribution network lines based on Prophet-LSTM model. *Sustainability, 15*(15), Article 11667. https://doi.org/10.3390/su151511667

Cheng, M., Liu, Z., Tao, X., Liu, Q., Zhang, Z., Pan, T., … Chen, E. (2025). A comprehensive survey of time series forecasting: Concepts, challenges, and future directions. *TechRxiv*. https://doi.org/10.36227/techrxiv.174430535.53879341/v1

Davidescu, A., Apostu, S., & Paul, A. (2021). Comparative analysis of different univariate forecasting methods in modelling and predicting the Romanian unemployment

rate for the period 2021–2022. *Entropy, 23*(3), Article 325. https://doi.org/10.3390/e23030325

Dong, Y., & Yan, C. (2024). Efficiency evaluation of electricity demand prediction models in lower-income countries: A case study of Panama. *Journal of Physics: Conference Series, 2874*(1), Article 012007. https://doi.org/10.1088/1742-6596/2874/1/012007

Ferreira, W., Grout, I., & Silva, A. (2020). Forecasting energy time-series data using a fuzzy ARTMAP neural network. *2020 International Conference on Power, Energy and Innovations (ICPEI)*, 1–4. https://doi.org/10.1109/ICPEI49860.2020.9431435

Gao, Y. (2025). Time series analysis in financial markets and biological phenomena. *Theoretical and Natural Science, 86*(1), 32–36. https://doi.org/10.54254/2753-8818/2025.20330

Ghatage, N., Patil, P., & Shinde, S. (2023). Lightweight RNN-based model for adaptive time series forecasting with concept drift detection in smart homes. *Journal Européen des Systèmes Automatisés, 56*(6), 981–991. https://doi.org/10.18280/jesa.560609

Hariyanto, S., Sumanto, Y., & Khabibah, S. (2023). Average-based fuzzy time series Markov chain based on frequency density partitioning. *Journal of Applied Mathematics, 2023*, Article 9319883. https://doi.org/10.1155/2023/9319883

Hasan, F. (2024). Traffic modeling using machine learning methods for predicting vehicle numbers at junctions: A case study in Colombo, Sri Lanka. *Research Square*. https://doi.org/10.21203/rs.3.rs-4130255/v1

Hindarto, D., Hendrata, F., & Hariadi, M. (2023). The application of Neural Prophet time series in predicting rice stock at rice stores. *Journal of Computer Networks, Architecture and High Performance Computing, 5*(2), 668–681. https://doi.org/10.47709/cnahpc.v5i2.2725

Jahangir, M., & Quilty, J. (2022). Temporal hierarchical reconciliation for consistent water resources forecasting across multiple timescales: An application to precipitation

forecasting. *Water Resources Research, 58*(6), Article e2021WR031862. https://doi.org/10.1029/2021WR031862

Jin, L. (2023). Exploration of cross-border e-commerce and its logistics supply chain innovation and development path for agricultural exports based on deep learning. *Applied Mathematics and Nonlinear Sciences, 9*(1), Article 01529. https://doi.org/10.2478/amns.2023.2.01529

Jung, S., Zhang, R., Chen, Y., & Joe, S. (2024). Optimizing demand forecasting for business events tourism: A comparative analysis of cutting-edge models. *Journal of Hospitality and Tourism Insights, 8*(1), 370–390. https://doi.org/10.1108/JHTI-12-2023-0960

Kirkwood, C., Economou, T., Odbert, H., & Pugeault, N. (2021). A framework for probabilistic weather forecast post-processing across models and lead times using machine learning. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 379*(2194), Article 20200099. https://doi.org/10.1098/rsta.2020.0099

Kolková, A., & Ključnikov, A. (2022). Demand forecasting: AI-based, statistical and hybrid models vs practice-based models - the case of SMEs and large enterprises. *Economics & Sociology, 15*(4), 39–62. https://doi.org/10.14254/2071-789X.2022/15-4/2

Kotenko, V. (2022). Application of algorithmic models of machine learning to the freight transportation process. *Transport Technologies, 2022*(2), 10–21. https://doi.org/10.23939/tt2022.02.010

Kotenko, V. (2022). Machine learning algorithmic models for forecasting fuel consumption by vehicles of the grain crops delivery. *Central Ukrainian Scientific Bulletin: Technical Sciences, 6*(37), 173–182. https://doi.org/10.32515/2664-262X.2022.6(37).1.173-182

Kourentzes, N., Trapero, J., & Barrow, D. (2020). Optimising forecasting models for inventory planning. *International Journal of Production Economics, 225*, Article 107597. https://doi.org/10.1016/j.ijpe.2019.107597

Kwarteng, S., & Andreevich, P. (2024). Comparative analysis of ARIMA, SARIMA and Prophet model in forecasting. *RD: Journal of Recent Developments, 5*(4), 110–120. https://doi.org/10.11648/j.rd.20240504.13

Loken, E., Clark, A., & Karstens, C. (2020). Generating probabilistic next-day severe weather forecasts from convection-allowing ensembles using random forests. *Weather and Forecasting, 35*(4), 1605–1631. https://doi.org/10.1175/WAF-D-19-0258.1

Lu, Y., Ye, M., Zhang, R., Zhao, Y., Wu, Y., & Guo, X. (2023). Urban rail transit passenger flow forecasting based on Prophet-GRU combined model. *Proceedings of SPIE, 12741*, Article 2686724. https://doi.org/10.1117/12.2686724

Madaras, S. (2024). Deep learning algorithm forecasting the unemployment rates in the Central European countries. *Economics and Business, 38*(1), 86–102. https://doi.org/10.7250/eb-2024-0006

Majhi, S., Bano, R., Srichan, S., Acharya, B., Al-Rasheed, A., Alqahtani, M., … Soufiene, B. (2023). Food price index prediction using time series models: A study of cereals, millets and pulses. *Research Square*. https://doi.org/10.21203/rs.3.rs-2999898/v1

Mohamed, R. (2023). Enhancing forecast accuracy using combination methods for the hierarchical time series approach. *PLoS ONE, 18*(7), Article e0287897. https://doi.org/10.1371/journal.pone.0287897

Ni, G., Zhang, X., Ni, X., Cheng, X., & Meng, X. (2023). A WOA-CNN-BiLSTM-based multi-feature classification prediction model for smart grid financial markets. *Frontiers in Energy Research, 11*, Article 1198855. https://doi.org/10.3389/fenrg.2023.1198855

Panapakidis, I., Sourtzi, V., & Dagoumas, A. (2020). Forecasting the fuel consumption of passenger ships with a combination of shallow and deep learning. *Electronics, 9*(5), Article 776. https://doi.org/10.3390/electronics9050776

Rathipriya, R., Rahman, A., Dhamodharavadhani, S., Meero, A., & Yoganandan, G. (2022). Demand forecasting model for time-series pharmaceutical data using shallow and deep neural network model. *Neural Computing and Applications, 35*(2), 1945–1957. https://doi.org/10.1007/s00521-022-07889-9

Rehman, H., Wan, G., & Rafique, R. (2022). A hybrid approach with step-size aggregation to forecasting hierarchical time series. *Journal of Forecasting, 42*(1), 176–192. https://doi.org/10.1002/for.2895

Sagheer, A., Hamdoun, H., & Youness, H. (2021). Deep LSTM-based transfer learning approach for coherent forecasts in hierarchical time series. *Sensors, 21*(13), Article 4379. https://doi.org/10.3390/s21134379

Shaikh, W., Shah, S., Pandhiani, S., & Solangi, M. (2022). Wavelet decomposition impacts on traditional forecasting time series models. *Computer Modeling in Engineering & Sciences, 130*(3), 1517–1532. https://doi.org/10.32604/cmes.2022.017822

Sharma, S., Ghimire, G., & Siddique, R. (2022). Machine learning for postprocessing ensemble streamflow forecasts. *Journal of Hydroinformatics, 25*(1), 126–139. https://doi.org/10.2166/hydro.2022.114

Silva, K., López-Gonzales, J., Turpo-Chaparro, J., Cano, E., & Rodrigues, P. (2023). Spatio-temporal visualization and forecasting of PM_{10} in the Brazilian state of Minas Gerais. *Scientific Reports, 13*(1), Article 30365. https://doi.org/10.1038/s41598-023-30365-w

Sun, J., Dong, H., Gao, Y., Fang, Y., & Kong, Y. (2021). The short-term load forecasting using an artificial neural network approach with periodic and nonperiodic factors: A case study of Tai'an, Shandong Province, China. *Computational Intelligence and Neuroscience, 2021*, Article 1502932. https://doi.org/10.1155/2021/1502932

Taghiyeh, S., Lengacher, D., Sadeghi, A., Sahebi-Fakhrabad, A., & Handfield, R. (2020). A multi-phase approach for product hierarchy forecasting in supply chain management: Application to MonarchFx Inc. *arXiv*. https://doi.org/10.48550/arXiv.2006.08931

Wang, S., & Han, S. (2024). Supply forecasting method of rural water plants based on long short-term memory network. *Proceedings of SPIE, 13079*, Article 3048740. https://doi.org/10.1117/12.3048740

Wang, Y. (2023). Overview of logistics demand forecasting methods. *Frontiers in Business, Economics and Management, 9*(2), 251–255. https://doi.org/10.54097/fbem.v9i2.9293

Yang, J., Zhang, X., Chen, W., & Fei, R. (2024). Prophet–CEEMDAN–ARBiLSTM-based model for short-term load forecasting. *Future Internet, 16*(6), Article 192. https://doi.org/10.3390/fi16060192

Yoon, T., Park, Y., Ryu, E., & Wang, Y. (2022). Robust probabilistic time series forecasting. *arXiv*. https://doi.org/10.48550/arXiv.2202.11910

Zhang, C., & Li, R. (2021). A novel closed-loop clustering algorithm for hierarchical load forecasting. *IEEE Transactions on Smart Grid, 12*(1), 432–441. https://doi.org/10.1109/TSG.2020.3015000

Zhou, F., Chen, P., Ma, L., Liu, Y., Wang, S., Zhang, J., … Yun, H. (2023). Sloth: Structured learning and task-based optimization for time series forecasting on hierarchies. *Proceedings of the AAAI Conference on Artificial Intelligence, 37*(9), 11417–11425. https://doi.org/10.1609/aaai.v37i9.26350

Anexo I – Código Python da Tarefa 1

Tarefa 1

Realizar a previsão da quantidade de caixas que serão produzidas para a semana seguinte, numa determinada operação, num determinado armazém, para um cliente específico.

1. Exploração Inicial dos Dados

```
# Importação Bibliotecas
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from pandas.plotting import autocorrelation_plot
from pandas import DataFrame
from pandas import concat
import numpy as np
from math import sqrt

from sklearn.metrics import mean_squared_error
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.stattools import acf
from statsmodels.tsa.stattools import pacf
from statsmodels.tsa.arima_model import ARIMA
from scipy.stats import boxcox

import seaborn as sns
sns.set_style("whitegrid")
import matplotlib.pyplot as plt
from matplotlib.pylab import rcParams
from matplotlib import colors
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')


# Importação dos dados
from google.colab import drive
drive.mount('/content/drive')
dir_path = "/content/drive/MyDrive/Estágio LS/"

dados_iniciais = pd.read_excel(dir_path + "datatualizada_2023_2.xlsx")


import pandas as pd

df_final = dados_iniciais


# visualização e descricao dos dados
df_final
```

2. Pré-processamento dos Dados

```
# Expandir as colunas do dataframe
print(df_final.columns.tolist())
```

```
# Ocultar as colunas que não interessam
dados_tranformados = df_final.copy()

dados_tranformados = dados_tranformados.drop(columns=['OE', 'Tipo Operação','Ano Fim Operação', 'Mês fim operaçãc

# Display the updated DataFrame
print(dados_tranformados.head())


# Verificando se há valores ausentes
print(dados_tranformados.isna().sum())


# Preenchendo valores ausentes com 0
dados_tranformados.fillna(0, inplace=True)


# Display the updated DataFrame
print(dados_tranformados.head())


# Converter as colunas para inteiros para evitar problemas com decimais
dados_tranformados['Cliente'] = dados_tranformados['Cliente'].astype(int)
dados_tranformados['Caixas'] = dados_tranformados['Caixas'].astype(int)

# Mostrar o resultado
print(dados_tranformados)



# Verificar os tipos das colunas
print(dados_tranformados[['Data do fim da operação']].dtypes)

# Verificar os primeiros registos das colunas para garantir que os dados estejam no formato correto
print(dados_tranformados[['Data do fim da operação']].head())



# Converter para o formato datetime
dados_tranformados['Data do fim da operação'] = pd.to_datetime(dados_tranformados['Data do fim da operação'])

# Mostrar o resultado
print(dados_tranformados[['Data do fim da operação']])


# Removendo as linhas com "1970-01-01"
df = dados_tranformados[dados_tranformados["Data do fim da operação"] != "1970-01-01"]

print(df)


soma_total = df['Caixas'].sum()
print(f"Soma total de caixas: {soma_total}")


# Converter a coluna para o tipo datetime
df['Data do fim da operação'] = pd.to_datetime(df['Data do fim da operação'], dayfirst=True)

# Ordenar da mais antiga para a mais recente
df = df.sort_values(by='Data do fim da operação')


# Filtrar onde a coluna COL é igual a 'A'
df_a = df[df['COL'] == 'A']

print(df_a)


#DF_A

# Aplique o rolling apenas na coluna numérica "Caixas"
rolling_mean = df_a['Caixas'].rolling(window=12).mean()
rolling_std = df_a['Caixas'].rolling(window=12).std()
```

```
plt.figure(figsize=(20, 6))
plt.plot(df_a['Data do fim da operação'], df_a['Caixas'], color='cornflowerblue', label='Original')  # Usando a co
plt.plot(df_a['Data do fim da operação'], rolling_mean, color='firebrick', label='Rolling Mean')
plt.plot(df_a['Data do fim da operação'], rolling_std, color='limegreen', label='Rolling Std')

# Adicionando rótulos e título
plt.xlabel('Date', size=12)
plt.ylabel('Caixas', size=12)
plt.legend(loc='upper left')
plt.title('Box Production A', size=14)

# Exibindo o gráfico
plt.show()


contagem = df_a['Caixas'].apply(lambda x: pd.isna(x) or x == 0 or str(x).strip() == '').sum()

print(f"Número de valores 0, nulos ou em branco: {contagem}")


# Remover as colunas indesejadas
df_a.drop(columns=["Região", "COL", "Cliente"], inplace=True)

# Definir a coluna de datas como índice
df_a.set_index("Data do fim da operação", inplace=True)

# Agregar por semana somando as "Caixas"
df_weekly = df_a.resample('W-SUN').sum()

# Exibir as primeiras linhas
print(df_weekly.head())


df_validacao = df_weekly.copy()


# Eliminar última linha de df_weekly
df_weekly = df_weekly.iloc[:-1]


df_weekly
```

3. Análise Exploratória

```
# Verificar os tipos de dados de cada coluna do DataFrame
print(df_weekly.dtypes)


#DF_A

# Aplique o rolling apenas na coluna numérica "Caixas"
rolling_mean = df_weekly['Caixas'].rolling(window=12).mean()
rolling_std = df_weekly['Caixas'].rolling(window=12).std()

plt.figure(figsize=(20, 6))
plt.plot(df_weekly.index, df_weekly['Caixas'], color='cornflowerblue', label='Original')  # Usando a coluna de da
plt.plot(df_weekly.index, rolling_mean, color='firebrick', label='Rolling Mean')
plt.plot(df_weekly.index, rolling_std, color='limegreen', label='Rolling Std')

# Adicionando rótulos e título
plt.xlabel('Date', size=12)
plt.ylabel('Caixas', size=12)
plt.legend(loc='upper left')
plt.title('Box Production R', size=14)

# Exibindo o gráfico
```

```
plt.show()
```

```
df_weekly
```

```
pd.set_option('display.max_rows', None)  # Mostra todas as linhas
print(df_weekly)
```

```
plt.figure(figsize = (20,6))
plt.plot(df_weekly['Caixas'], color = 'cornflowerblue')
plt.xlabel('Date', size = 12)
plt.ylabel('Box Production R', size = 12)
plt.show()
```

```
plt.figure(figsize = (20,6))
plt.hist(df_weekly['Caixas'], color = 'cornflowerblue')
plt.xlabel('Box Production R', size = 12)
plt.ylabel('Frequency', size = 12)
plt.show()
```

```
print("Data Shape: {}".format(df_weekly.shape))
```

```
value_1  = df_weekly.iloc[:88]  # First 87 rows
value_2  = df_weekly.iloc[88:]  # Remaining rows (87 to max)

# Check the new shapes
print(value_1.shape)
print(value_2.shape)
```

```
value_top = value_1.filter(items=['Caixas'])
```

```
value_bottom = value_2.filter(items=['Caixas'])
```

```
print("Mean of value_top: {}".format(round(value_top.mean()[0],3)))
print("Mean of value_bottom: {}".format(round(value_bottom.mean()[0],3)))
```

```
print("Variance of value_top: {}".format(round(value_top.var()[0],3)))
print("Variance of value_bottom: {}".format(round(value_bottom.var()[0],3)))
```

4. Transformações para Estacionaridade

```
df_weekly2 = df_weekly[['Caixas']]
```

```
df_weekly2
```

```
series = df_weekly2['Caixas']

# Plotar a ACF
plt.figure(figsize=(12, 5))
plt.subplot(121)  # 1 linha, 2 colunas, primeiro gráfico
plot_acf(series, lags=40, ax=plt.gca())  # Ajuste o número de lags conforme necessário
plt.title('Autocorrelation Function (ACF)')

# Plotar a PACF
plt.subplot(122)  # 1 linha, 2 colunas, segundo gráfico
```

```
plot_pacf(series, lags=40, ax=plt.gca())  # Ajuste o número de lags conforme necessário
plt.title('Partial Autocorrelation Function (PACF)')

# Ajustar o layout e mostrar o gráfico
plt.tight_layout()
plt.show()


def adfuller_test(ts, window = 12):

    movingAverage = ts.rolling(window).mean()
    movingSTD = ts.rolling(window).std()

    plt.figure(figsize = (20,6))
    orig = plt.plot(ts, color='cornflowerblue',
                    label='Original')
    mean = plt.plot(movingAverage, color='firebrick',
                    label='Rolling Mean')
    std = plt.plot(movingSTD, color='limegreen',
                   label='Rolling Std')
    plt.legend(loc = 'upper left')
    plt.title('Rolling Statistics', size = 14)
    plt.show(block=False)

    adf = adfuller(ts, autolag='AIC')

    print('ADF Statistic: {}'.format(round(adf[0],3)))
    print('p-value: {}'.format(round(adf[1],3)))
    print("###################################")
    print('Critical Values:')

    for key, ts in adf[4].items():
         print('{}: {}'.format(key, round(ts,3)))
    print("###################################")

    if adf[0] > adf[4]["5%"]:
        print("ADF > Critical Values")
        print ("Failed to reject null hypothesis, time series is non-stationary.")
    else:
        print("ADF < Critical Values")
        print ("Reject null hypothesis, time series is stationary.")

adfuller_test(df_weekly2, window = 12)


df_log_scaled = df_weekly2
df_log_scaled['Caixas'] = boxcox(df_log_scaled['Caixas'], lmbda=0.0)
plt.figure(figsize = (20,6))
plt.plot(df_log_scaled, color = 'cornflowerblue')
plt.xlabel('Date', size = 12)
plt.ylabel('Box Production R', size = 12)
plt.title("After Logarithmic Transformation", size = 14)
plt.show()


moving_avg = df_log_scaled.rolling(window=12).mean()
df_log_scaled_ma = df_log_scaled - moving_avg
df_log_scaled_ma.dropna(inplace=True)
plt.figure(figsize = (20,6))
plt.plot(df_log_scaled_ma, color = 'cornflowerblue')
plt.xlabel('Date', size = 12)
plt.ylabel('Box Production', size = 12)
plt.title("After Moving Average", size = 14)
plt.show()


df_log_scaled_ma_ed = df_log_scaled_ma.ewm(halflife=12, min_periods=0, adjust=True).mean()
df_lsma_sub_df_lsma_ed = df_log_scaled_ma - df_log_scaled_ma_ed
plt.figure(figsize = (20,6))
plt.plot(df_lsma_sub_df_lsma_ed - df_log_scaled_ma_ed, color='cornflowerblue')
```

```
plt.xlabel('Date', size = 12)
plt.ylabel('Box Production R', size = 12)
plt.title("After Exponential Decay Transformation", size = 14)
plt.show()


adfuller_test(df_lsma_sub_df_lsma_ed, window = 12)
```

5. Decomposição da Série

```
df_lsma_sub_df_lsma_ed = df_lsma_sub_df_lsma_ed.asfreq('W')  # Define a frequência como semanal (W)



df_lsma_sub_df_lsma_ed


df_lsma_sub_df_lsma_ed = df_lsma_sub_df_lsma_ed.fillna(method='ffill')



rcParams['figure.figsize']=10,8
df_seasonal_decompose = seasonal_decompose(df_lsma_sub_df_lsma_ed,
                                           model='additive')
df_seasonal_decompose.plot()
plt.show()
```

6. Análise dos Resíduos

```
import pandas as pd
import numpy as np
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.stats.diagnostic import acorr_ljungbox
import matplotlib.pyplot as plt
from matplotlib import rcParams

# Configurar o tamanho da figura
rcParams['figure.figsize'] = 10, 8

# Realizar a decomposição
df_seasonal_decompose = seasonal_decompose(df_lsma_sub_df_lsma_ed, model='additive')

# Extrair os resíduos
residuals = df_seasonal_decompose.resid.dropna()

# Análise dos resíduos
print("Média dos resíduos:", residuals.mean())
print("Variância dos resíduos:", residuals.var())

# Plotar os gráficos ACF e PACF dos resíduos
plt.figure(figsize=(12, 4))
plt.subplot(121)
plot_acf(residuals, lags=40, ax=plt.gca())
plt.title('ACF dos Resíduos')
plt.subplot(122)
plot_pacf(residuals, lags=40, ax=plt.gca())
plt.title('PACF dos Resíduos')
plt.tight_layout()
plt.show()

# Teste de Ljung-Box para autocorrelação
lb_test = acorr_ljungbox(residuals, lags=[10, 20, 30], return_df=True)
```

```
print("Teste de Ljung-Box para autocorrelação nos resíduos:")
print(lb_test)


auto_c_f = acf(df_lsma_sub_df_lsma_ed, nlags=20)
partial_auto_c_f = pacf(df_lsma_sub_df_lsma_ed, nlags=20, method='ols')

fig, axs = plt.subplots(1, 2, figsize =(12,5))

plt.subplot(121)
plt.plot(auto_c_f)
plt.axhline(y=0, linestyle='--', color='limegreen')
plt.axhline(y=-1.96/np.sqrt(len(df_lsma_sub_df_lsma_ed)),
            linestyle='--', color='firebrick')
plt.axhline(y=1.96/np.sqrt(len(df_lsma_sub_df_lsma_ed)),
            linestyle='--', color='firebrick')
plt.title('Autocorrelation Function', size = 14)

plt.subplot(122)
plt.plot(partial_auto_c_f)
plt.axhline(y=0, linestyle='--', color='limegreen')
plt.axhline(y=-1.96/np.sqrt(len(df_lsma_sub_df_lsma_ed)),
            linestyle='--', color='firebrick')
plt.axhline(y=1.96/np.sqrt(len(df_lsma_sub_df_lsma_ed)),
            linestyle='--', color='firebrick')
plt.title('Partial Autocorrelation Function', size = 14)

plt.tight_layout()


from statsmodels.tsa.stattools import adfuller

# Realizar o teste ADF
result = adfuller(df_lsma_sub_df_lsma_ed)
print('Estatística ADF:', result[0])
print('p-valor:', result[1])
print('Valores Críticos:', result[4])


df_weekly


df_weeklyx = df_weekly.copy()


df_weeklyx


df_weeklyx.shape
```

7. Modelagem com Prophet

```
df_weeklyx


# Criar uma cópia do DataFrame df_weekly para df_weeklyz
df_weeklyz = df_weekly.copy()

# Resetar o índice e adicionar como uma nova coluna
df_weeklyz = df_weeklyz.reset_index()

# Mostrar as primeiras linhas do novo DataFrame
print(df_weeklyz.head())


df_weeklyz = df_weeklyz.rename(columns={'Data do fim da operação': 'ds', 'Caixas': 'y'})
print(df_weeklyz.head())
```

```
from prophet import Prophet


# Criar uma cópia do DataFrame df_weekly para df_weeklyz
df_weeklyz = df_weekly.copy()

# Resetar o índice e adicionar como uma nova coluna
df_weeklyz = df_weeklyz.reset_index()

# Mostrar as primeiras linhas do novo DataFrame
print(df_weeklyz.head())


df_weeklyz


# Criar o DataFrame com os dados
data = df_weeklyz

data['Data do fim da operação'] = pd.to_datetime(data['Data do fim da operação'])


# Renomear as colunas
data = data.rename(columns={'Data do fim da operação': 'ds', 'Caixas': 'y'})


from prophet import Prophet

# Ajustar o modelo com todos os dados
model = Prophet(weekly_seasonality=True, yearly_seasonality=True, daily_seasonality=False)
model.add_country_holidays(country_name='PT')
model.fit(data)

# Prever as próximas semana (frequência semanal aos domingos)
future_dates = model.make_future_dataframe(periods=1, freq='W-SUN')
forecast = model.predict(future_dates)

next_week_forecast = forecast[forecast['ds'] == '2025-06-01'][['ds', 'yhat', 'yhat_lower', 'yhat_upper']]

print("Previsão para a próxima semana:")
print(next_week_forecast)
```

8. Avaliação da Previsão

```
real_value = float(input("valor real"))
error = real_value - next_week_forecast['yhat'].values[0]
mape = abs(error / real_value) * 100

print(f"\nValidação:")
print(f"Valor real: {real_value}")
print(f"Valor previsto: {next_week_forecast['yhat'].values[0]:.2f}")
print(f"Erro absoluto: {error:.2f}")
print(f"MAPE: {mape:.2f}%")
```

Anexo II – Código Python da Tarefa 2

Tarefa 2

Realizar a previsão da quantidade de caixas que serão produzidas para a semana seguinte, numa determinada operação, num determinado armazém, para um cliente específico.

1. Exploração Inicial dos Dados

```
# Importação Bibliotecas
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from pandas.plotting import autocorrelation_plot
from pandas import DataFrame
from pandas import concat
import numpy as np
from math import sqrt

from sklearn.metrics import mean_squared_error
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.stattools import acf
from statsmodels.tsa.stattools import pacf
from statsmodels.tsa.arima_model import ARIMA
from scipy.stats import boxcox

import seaborn as sns
sns.set_style("whitegrid")
import matplotlib.pyplot as plt
from matplotlib.pylab import rcParams
from matplotlib import colors
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')


# Importação dos dados
from google.colab import drive
drive.mount('/content/drive')
dir_path = "/content/drive/MyDrive/Estágio LS/"

dados_iniciais = pd.read_excel(dir_path + "datatualizada_2023_2.xlsx")


import pandas as pd

df_final = dados_iniciais


# visualização e descricao dos dados
df_final
```

2. Pré-processamento dos Dados

```
# Expandir as colunas do dataframe
print(df_final.columns.tolist())



# Ocultar as colunas que não interessam
dados_tranformados = df_final.copy()
```

```
dados_tranformados = dados_tranformados.drop(columns=['OE', 'Tipo Operação','Ano Fim Operação', 'Mês fim operação'

# Display the updated DataFrame
print(dados_tranformados.head())


# Verificando se há valores ausentes
print(dados_tranformados.isna().sum())


# Preenchendo valores ausentes com 0
dados_tranformados.fillna(0, inplace=True)


# Display the updated DataFrame
print(dados_tranformados.head())


# Converter as colunas para inteiros para evitar problemas com decimais
dados_tranformados['Cliente'] = dados_tranformados['Cliente'].astype(int)
dados_tranformados['Caixas'] = dados_tranformados['Caixas'].astype(int)

# Mostrar o resultado
print(dados_tranformados)



# Verificar os tipos das colunas
print(dados_tranformados[['Data do fim da operação']].dtypes)

# Verificar os primeiros registos das colunas para garantir que os dados estejam no formato correto
print(dados_tranformados[['Data do fim da operação']].head())



# Converter para o formato datetime
dados_tranformados['Data do fim da operação'] = pd.to_datetime(dados_tranformados['Data do fim da operação'])

# Mostrar o resultado
print(dados_tranformados[['Data do fim da operação']])


# Removendo as linhas com "1970-01-01"
df = dados_tranformados[dados_tranformados["Data do fim da operação"] != "1970-01-01"]

print(df)


soma_total = df['Caixas'].sum()
print(f"Soma total de caixas: {soma_total}")


# Converter a coluna para o tipo datetime
df['Data do fim da operação'] = pd.to_datetime(df['Data do fim da operação'], dayfirst=True)

# Ordenar da mais antiga para a mais recente
df = df.sort_values(by='Data do fim da operação')


# Filtrar onde a coluna COL é igual a 'C'
df_b = df[df['COL'] == 'B']

print(df_b)


#DF_B

# Aplique o rolling apenas na coluna numérica "Caixas"
rolling_mean = df_b['Caixas'].rolling(window=12).mean()
rolling_std = df_b['Caixas'].rolling(window=12).std()
```

```
plt.figure(figsize=(20, 6))
plt.plot(df_b['Data do fim da operação'], df_b['Caixas'], color='cornflowerblue', label='Original')  # Usando a c
plt.plot(df_b['Data do fim da operação'], rolling_mean, color='firebrick', label='Rolling Mean')
plt.plot(df_b['Data do fim da operação'], rolling_std, color='limegreen', label='Rolling Std')

# Adicionando rótulos e título
plt.xlabel('Date', size=12)
plt.ylabel('Caixas', size=12)
plt.legend(loc='upper left')
plt.title('Box Production C', size=14)

# Exibindo o gráfico
plt.show()


contagem = df_b['Caixas'].apply(lambda x: pd.isna(x) or x == 0 or str(x).strip() == '').sum()

print(f"Número de valores 0, nulos ou em branco: {contagem}")


# Remover as colunas indesejadas
df_b.drop(columns=["Região", "COL", "Cliente"], inplace=True)

# Definir a coluna de datas como índice
df_b.set_index("Data do fim da operação", inplace=True)

# Agregar por semana somando as "Caixas"
df_weekly = df_b.resample('W-SUN').sum()

# Exibir as primeiras linhas
print(df_weekly.head())


df_validacao = df_weekly.copy()


# Eliminar última linha de df_weekly
df_weekly = df_weekly.iloc[:-1]


df_weekly
```

3. Análise Exploratória

```
# Verificar os tipos de dados de cada coluna do DataFrame
print(df_weekly.dtypes)


#DF_B

# Aplique o rolling apenas na coluna numérica "Caixas"
rolling_mean = df_weekly['Caixas'].rolling(window=12).mean()
rolling_std = df_weekly['Caixas'].rolling(window=12).std()

plt.figure(figsize=(20, 6))
plt.plot(df_weekly.index, df_weekly['Caixas'], color='cornflowerblue', label='Original')  # Usando a coluna de dat
plt.plot(df_weekly.index, rolling_mean, color='firebrick', label='Rolling Mean')
plt.plot(df_weekly.index, rolling_std, color='limegreen', label='Rolling Std')

# Adicionando rótulos e título
plt.xlabel('Date', size=12)
plt.ylabel('Caixas', size=12)
plt.legend(loc='upper left')
plt.title('Box Production', size=14)

# Exibindo o gráfico
plt.show()
```

```
df_weekly
```

```
pd.set_option('display.max_rows', None)  # Mostra todas as linhas
print(df_weekly)
```

```
plt.figure(figsize = (20,6))
plt.plot(df_weekly['Caixas'], color = 'cornflowerblue')
plt.xlabel('Date', size = 12)
plt.ylabel('Box Production R', size = 12)
plt.show()
```

```
plt.figure(figsize = (20,6))
plt.hist(df_weekly['Caixas'], color = 'cornflowerblue')
plt.xlabel('Box Production R', size = 12)
plt.ylabel('Frequency', size = 12)
plt.show()
```

```
print("Data Shape: {}".format(df_weekly.shape))
```

```
value_1  = df_weekly.iloc[:88]  # First 87 rows
value_2  = df_weekly.iloc[88:]  # Remaining rows (87 to max)

# Check the new shapes
print(value_1.shape)
print(value_2.shape)
```

```
value_top = value_1.filter(items=['Caixas'])
```

```
value_bottom = value_2.filter(items=['Caixas'])
```

```
print("Mean of value_top: {}".format(round(value_top.mean()[0],3)))
print("Mean of value_bottom: {}".format(round(value_bottom.mean()[0],3)))
```

```
print("Variance of value_top: {}".format(round(value_top.var()[0],3)))
print("Variance of value_bottom: {}".format(round(value_bottom.var()[0],3)))
```

4. Transformações para Estacionaridade

```
df_weekly2 = df_weekly[['Caixas']]
```

```
df_weekly2
```

```
series = df_weekly2['Caixas']

# Plotar a ACF
plt.figure(figsize=(12, 5))
plt.subplot(121)  # 1 linha, 2 colunas, primeiro gráfico
plot_acf(series, lags=40, ax=plt.gca())  # Ajuste o número de lags conforme necessário
plt.title('Autocorrelation Function (ACF)')

# Plotar a PACF
plt.subplot(122)  # 1 linha, 2 colunas, segundo gráfico
plot_pacf(series, lags=40, ax=plt.gca())  # Ajuste o número de lags conforme necessário
plt.title('Partial Autocorrelation Function (PACF)')
```

```
# Ajustar o layout e mostrar o gráfico
plt.tight_layout()
plt.show()


def adfuller_test(ts, window = 12):

    movingAverage = ts.rolling(window).mean()
    movingSTD = ts.rolling(window).std()

    plt.figure(figsize = (20,6))
    orig = plt.plot(ts, color='cornflowerblue',
                    label='Original')
    mean = plt.plot(movingAverage, color='firebrick',
                    label='Rolling Mean')
    std = plt.plot(movingSTD, color='limegreen',
                   label='Rolling Std')
    plt.legend(loc = 'upper left')
    plt.title('Rolling Statistics', size = 14)
    plt.show(block=False)

    adf = adfuller(ts, autolag='AIC')

    print('ADF Statistic: {}'.format(round(adf[0],3)))
    print('p-value: {}'.format(round(adf[1],3)))
    print("###################################")
    print('Critical Values:')

    for key, ts in adf[4].items():
         print('{}: {}'.format(key, round(ts,3)))
    print("###################################")

    if adf[0] > adf[4]["5%"]:
        print("ADF > Critical Values")
        print ("Failed to reject null hypothesis, time series is non-stationary.")
    else:
        print("ADF < Critical Values")
        print ("Reject null hypothesis, time series is stationary.")

adfuller_test(df_weekly2, window = 12)


df_log_scaled = df_weekly2
df_log_scaled['Caixas'] = boxcox(df_log_scaled['Caixas'], lmbda=0.0)
plt.figure(figsize = (20,6))
plt.plot(df_log_scaled, color = 'cornflowerblue')
plt.xlabel('Date', size = 12)
plt.ylabel('Box Production R', size = 12)
plt.title("After Logarithmic Transformation", size = 14)
plt.show()


moving_avg = df_log_scaled.rolling(window=12).mean()
df_log_scaled_ma = df_log_scaled - moving_avg
df_log_scaled_ma.dropna(inplace=True)
plt.figure(figsize = (20,6))
plt.plot(df_log_scaled_ma, color = 'cornflowerblue')
plt.xlabel('Date', size = 12)
plt.ylabel('Box Production', size = 12)
plt.title("After Moving Average", size = 14)
plt.show()


df_log_scaled_ma_ed = df_log_scaled_ma.ewm(halflife=12, min_periods=0, adjust=True).mean()
df_lsma_sub_df_lsma_ed = df_log_scaled_ma - df_log_scaled_ma_ed
plt.figure(figsize = (20,6))
plt.plot(df_lsma_sub_df_lsma_ed - df_log_scaled_ma_ed, color='cornflowerblue')
plt.xlabel('Date', size = 12)
plt.ylabel('Box Production R', size = 12)
```

```
plt.title("After Exponential Decay Transformation", size = 14)
plt.show()


adfuller_test(df_lsma_sub_df_lsma_ed, window = 12)
```

5. Decomposição da Série

```
df_lsma_sub_df_lsma_ed = df_lsma_sub_df_lsma_ed.asfreq('W')  # Define a frequência como semanal (W)



df_lsma_sub_df_lsma_ed


df_lsma_sub_df_lsma_ed = df_lsma_sub_df_lsma_ed.fillna(method='ffill')



rcParams['figure.figsize']=10,8
df_seasonal_decompose = seasonal_decompose(df_lsma_sub_df_lsma_ed,
                                           model='additive')
df_seasonal_decompose.plot()
plt.show()
```

6. Análise dos Resíduos

```
import pandas as pd
import numpy as np
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.stats.diagnostic import acorr_ljungbox
import matplotlib.pyplot as plt
from matplotlib import rcParams

# Configurar o tamanho da figura
rcParams['figure.figsize'] = 10, 8

# Realizar a decomposição
df_seasonal_decompose = seasonal_decompose(df_lsma_sub_df_lsma_ed, model='additive')

# Extrair os resíduos
residuals = df_seasonal_decompose.resid.dropna()

# Análise dos resíduos
print("Média dos resíduos:", residuals.mean())
print("Variância dos resíduos:", residuals.var())

# Plotar os gráficos ACF e PACF dos resíduos
plt.figure(figsize=(12, 4))
plt.subplot(121)
plot_acf(residuals, lags=40, ax=plt.gca())
plt.title('ACF dos Resíduos')
plt.subplot(122)
plot_pacf(residuals, lags=40, ax=plt.gca())
plt.title('PACF dos Resíduos')
plt.tight_layout()
plt.show()

# Teste de Ljung-Box para autocorrelação
lb_test = acorr_ljungbox(residuals, lags=[10, 20, 30], return_df=True)
print("Teste de Ljung-Box para autocorrelação nos resíduos:")
print(lb_test)
```

```
auto_c_f = acf(df_lsma_sub_df_lsma_ed, nlags=20)
partial_auto_c_f = pacf(df_lsma_sub_df_lsma_ed, nlags=20, method='ols')

fig, axs = plt.subplots(1, 2, figsize =(12,5))

plt.subplot(121)
plt.plot(auto_c_f)
plt.axhline(y=0, linestyle='--', color='limegreen')
plt.axhline(y=-1.96/np.sqrt(len(df_lsma_sub_df_lsma_ed)),
            linestyle='--', color='firebrick')
plt.axhline(y=1.96/np.sqrt(len(df_lsma_sub_df_lsma_ed)),
            linestyle='--', color='firebrick')
plt.title('Autocorrelation Function', size = 14)

plt.subplot(122)
plt.plot(partial_auto_c_f)
plt.axhline(y=0, linestyle='--', color='limegreen')
plt.axhline(y=-1.96/np.sqrt(len(df_lsma_sub_df_lsma_ed)),
            linestyle='--', color='firebrick')
plt.axhline(y=1.96/np.sqrt(len(df_lsma_sub_df_lsma_ed)),
            linestyle='--', color='firebrick')
plt.title('Partial Autocorrelation Function', size = 14)

plt.tight_layout()


from statsmodels.tsa.stattools import adfuller

# Realizar o teste ADF
result = adfuller(df_lsma_sub_df_lsma_ed)
print('Estatística ADF:', result[0])
print('p-valor:', result[1])
print('Valores Críticos:', result[4])


df_weekly


df_weeklyx = df_weekly.copy()


df_weeklyx


df_weeklyx.shape
```

7. Modelagem com Prophet

```
df_weeklyx


# Criar uma cópia do DataFrame df_weekly para df_weeklyz
df_weeklyz = df_weekly.copy()

# Resetar o índice e adicionar como uma nova coluna
df_weeklyz = df_weeklyz.reset_index()

# Mostrar as primeiras linhas do novo DataFrame
print(df_weeklyz.head())


df_weeklyz = df_weeklyz.rename(columns={'Data do fim da operação': 'ds', 'Caixas': 'y'})
print(df_weeklyz.head())


from prophet import Prophet
```

```
# Criar uma cópia do DataFrame df_weekly para df_weeklyz
df_weeklyz = df_weekly.copy()

# Resetar o índice e adicionar como uma nova coluna
df_weeklyz = df_weeklyz.reset_index()

# Mostrar as primeiras linhas do novo DataFrame
print(df_weeklyz.head())
```

```
df_weeklyz
```

```
# Criar o DataFrame com os dados
data = df_weeklyz

data['Data do fim da operação'] = pd.to_datetime(data['Data do fim da operação'])
```

```
# Renomear as colunas
data = data.rename(columns={'Data do fim da operação': 'ds', 'Caixas': 'y'})
```

```
from prophet import Prophet

# Ajustar o modelo com todos os dados
model = Prophet(weekly_seasonality=True, yearly_seasonality=True, daily_seasonality=False)
model.add_country_holidays(country_name='PT')
model.fit(data)

# Prever as próximas semana (frequência semanal aos domingos)
future_dates = model.make_future_dataframe(periods=1, freq='W-SUN')
forecast = model.predict(future_dates)

next_week_forecast = forecast[forecast['ds'] == '2025-06-01'][['ds', 'yhat', 'yhat_lower', 'yhat_upper']]

print("Previsão para a próxima semana:")
print(next_week_forecast)
```

8. Avaliação da Previsão

```
real_value = float(input("valor real"))
error = real_value - next_week_forecast['yhat'].values[0]
mape = abs(error / real_value) * 100

print(f"\nValidação:")
print(f"Valor real: {real_value}")
print(f"Valor previsto: {next_week_forecast['yhat'].values[0]:.2f}")
print(f"Erro absoluto: {error:.2f}")
print(f"MAPE: {mape:.2f}%")
```

Anexo III – Código Python da Tarefa 3

Tarefa 3

Realizar a previsão da quantidade de expedições que serão produzidas para a semana seguinte, para um cliente específico.

1. Exploração Inicial dos Dados

```
#import libraries
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from pandas.plotting import autocorrelation_plot
from pandas import DataFrame
from pandas import concat
import numpy as np
from math import sqrt

from sklearn.metrics import mean_squared_error
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.stattools import acf
from statsmodels.tsa.stattools import pacf
from statsmodels.tsa.arima_model import ARIMA
from scipy.stats import boxcox

import seaborn as sns
sns.set_style("whitegrid")
import matplotlib.pyplot as plt
from matplotlib.pylab import rcParams
from matplotlib import colors
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')


# importação dos dados
from google.colab import drive
drive.mount('/content/drive')
dir_path = "/content/drive/MyDrive/Estágio LS/"

dados_iniciais = pd.read_excel(dir_path + "data3.xlsx")


import pandas as pd

df_final = dados_iniciais
```

```
# visualização e descricao dos dados
df_final
```

2. Pré-processamento dos Dados

```
# Expandir as colunas do dataframe
print(df_final.columns.tolist())
```

```
# Ocultar as colunas que não interessam
dados_tranformados = df_final.copy()

dados_tranformados = dados_tranformados.drop(columns=['CLIENTE.1', 'DIST O EXP', 'DIST D

# Display the updated DataFrame
print(dados_tranformados.head())
```

```
# Verificando se há valores ausentes
print(dados_tranformados.isna().sum())
```

```
# Preenchendo valores ausentes com 0 (se necessário), ou você pode usar outra estratégia
dados_tranformados.fillna(0, inplace=True)
```

```
# Display the updated DataFrame
print(dados_tranformados.head())
```

```
# Convertendo as colunas para inteiros para evitar problemas com decimais
dados_tranformados['CLIENTE'] = dados_tranformados['CLIENTE'].astype(int)
dados_tranformados['Count of EXP'] = dados_tranformados['Count of EXP'].astype(int)

# Mostrar o resultado
print(dados_tranformados)
```

```
# Verificar os tipos das colunas
print(dados_tranformados[['DT CARGA E']].dtypes)

# Verificar os primeiros registros das colunas para garantir que os dados estejam no form
print(dados_tranformados[['DT CARGA E']].head())
```

```
# Agora converta para o formato datetime
dados_tranformados['DT CARGA E'] = pd.to_datetime(dados_tranformados['DT CARGA E'])

# Mostrar o resultado
print(dados_tranformados[['DT CARGA E']])


# Removendo as linhas com "1970-01-01"
df = dados_tranformados[dados_tranformados["DT CARGA E"] != "1970-01-01"]

print(df)


soma_total = df['Count of EXP'].sum()
print(f"Soma total de expedições: {soma_total}")


df_exp = df.copy()


#DF_EXP

# Aplique o rolling apenas na coluna numérica "Caixas"
rolling_mean = df_exp['Count of EXP'].rolling(window=12).mean()
rolling_std = df_exp['Count of EXP'].rolling(window=12).std()

plt.figure(figsize=(20, 6))
plt.plot(df_exp['DT CARGA E'], df_exp['Count of EXP'], color='cornflowerblue', label='Ori
plt.plot(df_exp['DT CARGA E'], rolling_mean, color='firebrick', label='Rolling Mean')
plt.plot(df_exp['DT CARGA E'], rolling_std, color='limegreen', label='Rolling Std')

# Adicionando rótulos e título
plt.xlabel('Date', size=12)
plt.ylabel('Count of EXP', size=12)
plt.legend(loc='upper left')
plt.title('EXP', size=14)

# Exibindo o gráfico
plt.show()


contagem = df_exp['Count of EXP'].apply(lambda x: pd.isna(x) or x == 0 or str(x).strip()

print(f"Número de valores 0, nulos ou em branco: {contagem}")


df_exp


# Remover as colunas indesejadas
df_exp.drop(columns=['CLIENTE'], inplace=True)

# Definir a coluna de datas como índice
df_exp.set_index("DT CARGA E", inplace=True)

# Agregar por semana somando as "Contagem de EXP"
df_weekly = df_exp.resample('W-SUN').sum()
```

```
# Exibir as primeiras linhas
print(df_weekly.head())


df_validacao = df_weekly.copy()


# Eliminar a última linha de df_weekly
df_weekly = df_weekly.iloc[:-2]


df_weekly
```

3. Análise Exploratória

```
# Verificar os tipos de dados de cada coluna do DataFrame
print(df_weekly.dtypes)


#DF_EXP

# Aplique o rolling apenas na coluna numérica "Contagem de EXP"
rolling_mean = df_weekly['Count of EXP'].rolling(window=12).mean()
rolling_std = df_weekly['Count of EXP'].rolling(window=12).std()

plt.figure(figsize=(20, 6))
plt.plot(df_weekly.index, df_weekly['Count of EXP'], color='cornflowerblue', label='Origi
plt.plot(df_weekly.index, rolling_mean, color='firebrick', label='Rolling Mean')
plt.plot(df_weekly.index, rolling_std, color='limegreen', label='Rolling Std')

# Adicionando rótulos e título
plt.xlabel('Date', size=12)
plt.ylabel('Expedições', size=12)
plt.legend(loc='upper left')
plt.title('EXP', size=14)

# Exibindo o gráfico
plt.show()



df_weekly


pd.set_option('display.max_rows', None)  # Mostra todas as linhas
print(df_weekly)



plt.figure(figsize = (20,6))
plt.plot(df_weekly['Count of EXP'], color = 'cornflowerblue')
plt.xlabel('Date', size = 12)
```

```
plt.ylabel('EXP', size = 12)
plt.show()


plt.figure(figsize = (20,6))
plt.hist(df_weekly['Count of EXP'], color = 'cornflowerblue')
plt.xlabel('EXP', size = 12)
plt.ylabel('Frequency', size = 12)
plt.show()


print("Data Shape: {}".format(df_weekly.shape))


# Assuming df is your DataFrame with shape (100, 1)
value_1  = df_weekly.iloc[:99]  # First 50 rows
value_2  = df_weekly.iloc[99:]  # Remaining rows (51 to 100)

# Check the new shapes
print(value_1.shape)  # Should print (50, 1)
print(value_2.shape)  # Should print (50, 1)



value_top = value_1.filter(items=['Count of EXP'])



value_bottom = value_2.filter(items=['Count of EXP'])


print("Mean of value_top: {}".format(round(value_top.mean()[0],3)))
print("Mean of value_bottom: {}".format(round(value_bottom.mean()[0],3)))


print("Variance of value_top: {}".format(round(value_top.var()[0],3)))
print("Variance of value_bottom: {}".format(round(value_bottom.var()[0],3)))
```

4. Transformações para Estacionaridade

```
df_weekly2 = df_weekly[['Count of EXP']]


df_weekly2


# Supondo que a coluna com os dados de produção se chama 'Production'
series = df_weekly2['Count of EXP']

# Plotar a ACF
plt.figure(figsize=(12, 5))
plt.subplot(121)  # 1 linha, 2 colunas, primeiro gráfico
plot_acf(series, lags=40, ax=plt.gca())  # Ajuste o número de lags conforme necessário
plt.title('Autocorrelation Function (ACF)')
```

```
# Plotar a PACF
plt.subplot(122)  # 1 linha, 2 colunas, segundo gráfico
plot_pacf(series, lags=40, ax=plt.gca())  # Ajuste o número de lags conforme necessário
plt.title('Partial Autocorrelation Function (PACF)')

# Ajustar o layout e mostrar o gráfico
plt.tight_layout()
plt.show()


def adfuller_test(ts, window = 12):

    movingAverage = ts.rolling(window).mean()
    movingSTD = ts.rolling(window).std()

    plt.figure(figsize = (20,6))
    orig = plt.plot(ts, color='cornflowerblue',
                    label='Original')
    mean = plt.plot(movingAverage, color='firebrick',
                    label='Rolling Mean')
    std = plt.plot(movingSTD, color='limegreen',
                   label='Rolling Std')
    plt.legend(loc = 'upper left')
    plt.title('Rolling Statistics', size = 14)
    plt.show(block=False)

    adf = adfuller(ts, autolag='AIC')

    print('ADF Statistic: {}'.format(round(adf[0],3)))
    print('p-value: {}'.format(round(adf[1],3)))
    print("###################################")
    print('Critical Values:')

    for key, ts in adf[4].items():
         print('{}: {}'.format(key, round(ts,3)))
    print("###################################")

    if adf[0] > adf[4]["5%"]:
        print("ADF > Critical Values")
        print ("Failed to reject null hypothesis, time series is non-stationary.")
    else:
        print("ADF < Critical Values")
        print ("Reject null hypothesis, time series is stationary.")

adfuller_test(df_weekly2, window = 12)


df_log_scaled = df_weekly2
df_log_scaled['Count of EXP'] = boxcox(df_log_scaled['Count of EXP'], lmbda=0.0)
plt.figure(figsize = (20,6))
plt.plot(df_log_scaled, color = 'cornflowerblue')
plt.xlabel('Date', size = 12)
plt.ylabel('EXP A', size = 12)
plt.title("After Logarithmic Transformation", size = 14)
plt.show()
```

```
moving_avg = df_log_scaled.rolling(window=12).mean()
df_log_scaled_ma = df_log_scaled - moving_avg
df_log_scaled_ma.dropna(inplace=True)
plt.figure(figsize = (20,6))
plt.plot(df_log_scaled_ma, color = 'cornflowerblue')
plt.xlabel('Date', size = 12)
plt.ylabel('EXP', size = 12)
plt.title("After Moving Average", size = 14)
plt.show()


df_log_scaled_ma_ed = df_log_scaled_ma.ewm(halflife=12, min_periods=0, adjust=Tru
df_lsma_sub_df_lsma_ed = df_log_scaled_ma - df_log_scaled_ma_ed
plt.figure(figsize = (20,6))
plt.plot(df_lsma_sub_df_lsma_ed - df_log_scaled_ma_ed, color='cornflowerblue')
plt.xlabel('Date', size = 12)
plt.ylabel('EXP A', size = 12)
plt.title("After Exponential Decay Transformation", size = 14)
plt.show()


adfuller_test(df_lsma_sub_df_lsma_ed, window = 12)
```

5. Decomposição da Série

```
df_lsma_sub_df_lsma_ed = df_lsma_sub_df_lsma_ed.asfreq('W')  # Define a frequência como s



df_lsma_sub_df_lsma_ed


df_lsma_sub_df_lsma_ed = df_lsma_sub_df_lsma_ed.fillna(method='ffill')



import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose
from matplotlib import rcParams

rcParams['figure.figsize'] = 10, 8

# Garantir que o índice é datetime e tem frequência
df_lsma_sub_df_lsma_ed.index = pd.to_datetime(df_lsma_sub_df_lsma_ed.index)
df_lsma_sub_df_lsma_ed = df_lsma_sub_df_lsma_ed.asfreq('W')

# Decomposição com período de 7 (sazonalidade semanal)
df_seasonal_decompose = seasonal_decompose(df_lsma_sub_df_lsma_ed, model='additive', peri

df_seasonal_decompose.plot()
plt.show()
```

6. Análise dos Resíduos

```
import pandas as pd
import numpy as np
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.stats.diagnostic import acorr_ljungbox
import matplotlib.pyplot as plt
from matplotlib import rcParams

# Configurar o tamanho da figura
rcParams['figure.figsize'] = 10, 8

# Realizar a decomposição
df_seasonal_decompose = seasonal_decompose(df_lsma_sub_df_lsma_ed, model='additive', peri

# Extrair os resíduos
residuals = df_seasonal_decompose.resid.dropna()

# Análise dos resíduos
print("Média dos resíduos:", residuals.mean())
print("Variância dos resíduos:", residuals.var())

# Plotar os gráficos ACF e PACF dos resíduos
plt.figure(figsize=(12, 4))
plt.subplot(121)
plot_acf(residuals, lags=40, ax=plt.gca())
plt.title('ACF dos Resíduos')
plt.subplot(122)
plot_pacf(residuals, lags=39, ax=plt.gca())
plt.title('PACF dos Resíduos')
plt.tight_layout()
plt.show()

# Teste de Ljung-Box para autocorrelação
lb_test = acorr_ljungbox(residuals, lags=[10, 20, 30], return_df=True)
print("Teste de Ljung-Box para autocorrelação nos resíduos:")
print(lb_test)


auto_c_f = acf(df_lsma_sub_df_lsma_ed, nlags=20)
partial_auto_c_f = pacf(df_lsma_sub_df_lsma_ed, nlags=20, method='ols')

fig, axs = plt.subplots(1, 2, figsize =(12,5))

plt.subplot(121)
plt.plot(auto_c_f)
plt.axhline(y=0, linestyle='--', color='limegreen')
plt.axhline(y=-1.96/np.sqrt(len(df_lsma_sub_df_lsma_ed)),
            linestyle='--', color='firebrick')
plt.axhline(y=1.96/np.sqrt(len(df_lsma_sub_df_lsma_ed)),
            linestyle='--', color='firebrick')
plt.title('Autocorrelation Function', size = 14)
```

```
plt.subplot(122)
plt.plot(partial_auto_c_f)
plt.axhline(y=0, linestyle='--', color='limegreen')
plt.axhline(y=-1.96/np.sqrt(len(df_lsma_sub_df_lsma_ed)),
            linestyle='--', color='firebrick')
plt.axhline(y=1.96/np.sqrt(len(df_lsma_sub_df_lsma_ed)),
            linestyle='--', color='firebrick')
plt.title('Partial Autocorrelation Function', size = 14)

plt.tight_layout()


from statsmodels.tsa.stattools import adfuller

# Realizar o teste ADF
result = adfuller(df_lsma_sub_df_lsma_ed)  # Substitua 'df' pelo nome da sua série
print('Estatística ADF:', result[0])
print('p-valor:', result[1])
print('Valores Críticos:', result[4])


df_weekly


df_weeklyx = df_weekly.copy()


df_weeklyx


df_weeklyx.shape


# Resetar o índice para transformar "DT CARGA E" em uma coluna
df_weeklyx = df_weeklyx.reset_index()

# Renomear as colunas para o formato do Prophet
df_weeklyx = df_weeklyx.rename(columns={'DT CARGA E': 'ds', 'Count of EXP': 'y'})
```

7. Modelização do Prophet

```
import pandas as pd
from prophet import Prophet

# Optional: Sort data just in case
df_weeklyx = df_weeklyx.sort_values('ds')

print("--- Input Data Sample (Last 5 rows) ---")
print(df_weeklyx.tail())
print("\n")
```

```
# --- 2. Initialize and Fit Prophet Model ---

# Add Portuguese holidays
# Prophet will automatically include standard PT holidays for the years in your data
model = Prophet(
    yearly_seasonality=True,
    weekly_seasonality=False,
    daily_seasonality=False,
    holidays_prior_scale=10.0, # Default is 10, adjust if holidays seem over/under-fitted
    changepoint_prior_scale=0.1, # Increased from default 0.05 to allow more trend flexib
    seasonality_prior_scale=15.0, # Increased from default 10 to allow more seasonal flex
    seasonality_mode='multiplicative' # Uncomment this line to TRY multiplicative seasona
)

# Add country holidays for Portugal
model.add_country_holidays(country_name='PT')

# Use df_weeklyx to fit the model
model.fit(df_weeklyx)


# --- 3. Create Future DataFrame ---
future = model.make_future_dataframe(periods=1, freq='W')

print("--- Future DataFrame ---")
print(future.tail()) # Show the date we are forecasting for
print("\n")


# --- 4. Make Prediction ---
forecast = model.predict(future)


# --- 5. Display Forecast ---
print("--- Forecast ---")
print(forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail(1))
print("\n")


# --- !! IMPORTANT FOR EVALUATION !! ---
# --- 6. Visualize and Evaluate ---

print("--- Next Steps ---")
print("1. Examine the plots below to see if the fit has improved.")
print("2. Calculate MAPE on a hold-out TEST SET (e.g., last few months) for a true perfor
print("3. Adjust 'changepoint_prior_scale' and 'seasonality_prior_scale' further if neede
print("4. Try 'seasonality_mode=multiplicative' if seasonal patterns scale with the trend

# Visualize the forecast
fig1 = model.plot(forecast)
# plt.title("Forecast vs Actuals") # Requires matplotlib
# plt.show() # Requires matplotlib

# Visualize the components (trend, holidays, yearly seasonality)
fig2 = model.plot_components(forecast)
```

```
# plt.show() # Requires matplotlib
```

8. Avaliação da Previsão

```
# Filtra a linha para a data desejada
target_date = pd.to_datetime('2025-03-30')
row = forecast[forecast['ds'] == target_date]

if row.empty:
    print("Data 2025-03-30 não encontrada no DataFrame forecast.")
else:
    real_value = float(input("Valor real: "))
    predicted_value = row['yhat'].values[0]
    error = real_value - predicted_value
    mape = abs(error / real_value) * 100

    print(f"\nValidação:")
    print(f"Valor real: {real_value}")
    print(f"Valor previsto: {predicted_value:.2f}")
    print(f"Erro absoluto: {error:.2f}")
    print(f"MAPE: {mape:.2f}%")
```

Anexo IV – Código Python da Tarefa 4

Tarefa 4

Realizar a previsão da quantidade de caixas que serão produzidas para a semana seguinte, numa determinada operação, num determinado armazém, para um cliente específico, com base em várias séries.

1. Exploração Inicial dos Dados

```
#import libraries
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from pandas.plotting import autocorrelation_plot
from pandas import DataFrame
from pandas import concat
import numpy as np
from math import sqrt

from sklearn.metrics import mean_squared_error
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.stattools import acf
from statsmodels.tsa.stattools import pacf
from statsmodels.tsa.arima_model import ARIMA
from scipy.stats import boxcox

import seaborn as sns
sns.set_style("whitegrid")
import matplotlib.pyplot as plt
from matplotlib.pylab import rcParams
from matplotlib import colors
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')


# importação dos dados
from google.colab import drive
drive.mount('/content/drive')
dir_path = "/content/drive/MyDrive/Estágio LS/"

dados_iniciais = pd.read_excel(dir_path + "varias_series.xlsx")
```

```
import pandas as pd

df_final = dados_iniciais


# visualização e descricao dos dados
df_final
```

2. Pré-processamento dos Dados e Análise Exploratória

```
# Expandir as colunas do dataframe
print(df_final.columns.tolist())
```

```
# Ocultar as colunas que não interessam
dados_tranformados = df_final.copy()

dados_tranformados = dados_tranformados.drop(columns=['OE', 'Tipo Operação','Ano Fim Oper

# Display the updated DataFrame
print(dados_tranformados.head())


# Verificar se há valores ausentes
print(dados_tranformados.isna().sum())


# Preencher valores ausentes com 0
dados_tranformados.fillna(0, inplace=True)


# Display the updated DataFrame
print(dados_tranformados.head())


dados_tranformados = dados_tranformados[dados_tranformados["COL"] != "LEIX"]


# Converter as colunas para inteiros para evitar problemas com decimais
dados_tranformados['Cliente'] = dados_tranformados['Cliente'].astype(int)
dados_tranformados['Caixas'] = dados_tranformados['Caixas'].astype(int)

# Mostrar o resultado
print(dados_tranformados)
```

```
# Verificar os tipos das colunas
print(dados_tranformados[['Data do fim da operação']].dtypes)

# Verificar os primeiros registos das colunas para garantir que os dados estejam no forma
```

```
print(dados_tranformados[['Data do fim da operação']].head())



# Converter para o formato datetime
dados_tranformados['Data do fim da operação'] = pd.to_datetime(dados_tranformados['Data d

# Mostrar o resultado
print(dados_tranformados[['Data do fim da operação']])


# Remover as linhas com "1970-01-01"
df = dados_tranformados[dados_tranformados["Data do fim da operação"] != "1970-01-01"]

print(df)


soma_total = df['Caixas'].sum()
print(f"Soma total de caixas: {soma_total}")


# Converter a coluna para o tipo datetime
df['Data do fim da operação'] = pd.to_datetime(df['Data do fim da operação'], dayfirst=Tr

# Ordenar da mais antiga para a mais recente
df = df.sort_values(by='Data do fim da operação')


dados_tranformados = dados_tranformados.sort_values(by=["Cliente", "Data do fim da operaç


print(dados_tranformados.groupby("Cliente")["Data do fim da operação"].min())
print(dados_tranformados.groupby("Cliente")["Data do fim da operação"].max())


clientes_dict = dict(tuple(dados_tranformados.groupby("Cliente")))



clientes_dict
```

```
import matplotlib.pyplot as plt

# Tamanho base dos gráficos
plt.figure(figsize=(10, 5))

# Loop por cada cliente
for cliente, df_cliente in clientes_dict.items():
    plt.figure(figsize=(10, 4))
    plt.plot(df_cliente["Data do fim da operação"], df_cliente["Caixas"], marker='o')
    plt.title(f"Série Temporal - Cliente: {cliente}")
    plt.xlabel("Data")
    plt.ylabel("Caixas")
    plt.grid(True)
    plt.tight_layout()
    plt.show()
```

```
import matplotlib.pyplot as plt

for cliente, df_cliente in clientes_dict.items():
    # Ordena por data (importante para o rolling)
    df_cliente = df_cliente.sort_values("Data do fim da operação")

    # Aplica rolling
    rolling_mean = df_cliente['Caixas'].rolling(window=12).mean()
    rolling_std = df_cliente['Caixas'].rolling(window=12).std()

    # Cria o gráfico
    plt.figure(figsize=(20, 6))
    plt.plot(df_cliente['Data do fim da operação'], df_cliente['Caixas'], color='cornflow
    plt.plot(df_cliente['Data do fim da operação'], rolling_mean, color='firebrick', labe
    plt.plot(df_cliente['Data do fim da operação'], rolling_std, color='limegreen', label

    # Rótulos e título
    plt.xlabel('Date', size=12)
    plt.ylabel('Caixas', size=12)
    plt.legend(loc='upper left')
    plt.title(f'Box Production R - Cliente: {cliente}', size=14)

    plt.tight_layout()
    plt.show()
```

```
clientes_dict
```

```
dados_tranformados
```

```
# Remover as colunas "Região" e "COL"
dados_tranformados = dados_tranformados.drop(columns=["Região", "COL"], errors='ignore')

# Converter a coluna "Data do fim da operação" para o formato datetime, se necessário
```

```
dados_tranformados["Data do fim da operação"] = pd.to_datetime(dados_tranformados["Data d

# Inicializar uma lista para armazenar os dados por cliente
clientes_agregados_semanal = []

# Iterar sobre cada cliente no dicionário
for cliente, df_cliente in clientes_dict.items():
    # Filtrar dados do cliente
    df_cliente = dados_tranformados[dados_tranformados["Cliente"] == cliente]

    # Definir "Data do fim da operação" como índice
    df_cliente.set_index("Data do fim da operação", inplace=True)

    # Agregar por semana somando as caixas
    df_cliente_semanal = df_cliente.resample('W').sum()

    # Adicionar a coluna do cliente no DataFrame
    df_cliente_semanal['Cliente'] = cliente

    # Adicionar os dados à lista (renomeando a coluna 'Caixas' para o nome do cliente)
    clientes_agregados_semanal.append(df_cliente_semanal[['Caixas']].rename(columns={'Cai

# Concatenar todos os DataFrames de clientes numa única estrutura
df_final_semanal = pd.concat(clientes_agregados_semanal, axis=1)

# Exibir as primeiras linhas do DataFrame final
print(df_final_semanal.head())
```

```
# Substituir os NaN por 0
df_final_semanal = df_final_semanal.fillna(0)

# Exibir as primeiras linhas do DataFrame final
print(df_final_semanal.head())
```

```
df_final_semanal
```

```
df_validacao = df_final_semanal.copy()
```

```
# Eliminar a última linha de df_weekly
df_weekly = df_final_semanal.iloc[:-1]
```

```
df_weekly
```

Clique duas vezes (ou prima Enter) para editar.

The first thing to do in any data analysis task is to plot the data. Graphs enable many features of the data to be visualised, including patterns, unusual observations, changes over time, and relationships between variables. The features that are seen in plots of the data must then be incorporated, as much as possible, into the forecasting methods to be used.

```
# Verificar os tipos de dados de cada coluna do DataFrame
print(df_weekly.dtypes)
```

3. Hierarchical Forecast

```
df_weekly['y'] = df_weekly.sum(axis=1)
```

```
df_weekly
```

```
# 2. Transforma o índice (datas) numa coluna chamada 'ds'
df_weekly = df_weekly.reset_index().rename(columns={'index': 'ds'})
```

```
# 3. Mantém apenas as colunas 'ds' e 'y'
df_prepared = df_weekly[['Data do fim da operação', 'y']]
```

```
# Renomeando as colunas
df_prepared = df_prepared.rename(columns={'Data do fim da operação': 'ds'})
```

```
df_prepared
```

```
from prophet import Prophet
```

```
df_prepared_copy = df_prepared.iloc[:-1].copy()
```

```
# Ajustar o modelo com todos os dados até 2025-03-30
model = Prophet(weekly_seasonality=True, yearly_seasonality=True, daily_seasonality=False
model.add_country_holidays(country_name='PT')
model.fit(df_prepared_copy)

# Prever a próxima semana (2025-03-30)
future_dates = model.make_future_dataframe(periods=1, freq='W-SUN')
forecast = model.predict(future_dates)
next_week_forecast = forecast[forecast['ds'] == '2025-03-30'][['ds', 'yhat', 'yhat_lower'

print("Previsão para 2025-03-30:")
print(next_week_forecast)
```

```
# Se tens o valor real de 2025-03-30, adiciona-o aqui
real_value = float(input("valor real"))
error = real_value - next_week_forecast['yhat'].values[0]
mape = abs(error / real_value) * 100

print(f"\nValidação para 2025-03-30:")
print(f"Valor real: {real_value}")
print(f"Valor previsto: {next_week_forecast['yhat'].values[0]:.2f}")
print(f"Erro absoluto: {error:.2f}")
print(f"MAPE: {mape:.2f}%")


df_weekly


# Ocultar as colunas que não interessam
df_bottom = df_weekly.copy()

df_bottom = df_bottom.drop(columns=['y'])

# Display the updated DataFrame
print(df_bottom.head())


from prophet import Prophet
import pandas as pd

df_bottom = df_bottom.copy()
df_bottom = df_bottom.rename(columns={'Data do fim da operação': 'ds'})  # renomeia para



print(df_bottom.columns.tolist())



print(df_bottom.head())



df_bottom2 = df_bottom.iloc[:-1]


from prophet import Prophet
import pandas as pd
import numpy as np

# Cópia do DataFrame original
df = df_bottom2.copy()

# Garante que a coluna de datas se chame 'ds' e está no formato datetime
df = df.rename(columns={'Data do fim da operação ': 'ds'})
df['ds'] = pd.to_datetime(df['ds'])

# Lista de colunas a serem processadas (exclui 'ds')
series_cols = [col for col in df.columns if col != 'ds']
```

```
print(f"Séries encontradas: {series_cols}")  # DEBUG: lista de séries

# Guarda previsões
forecast_dict = {}

# Loop para cada série
for col in series_cols:
    print(f"Processando série: {col}")  # DEBUG

    # Prepara o DataFrame para o Prophet
    df_prophet = df[['ds', col]].rename(columns={col: 'y'})

    # Trata possíveis NaNs
    df_prophet = df_prophet.dropna()

    if df_prophet.empty:
        print(f"Série {col} está vazia após dropna(). Pulando.")
        continue

    # No início do loop, aplica a transformação log (se necessário)
    df_prophet['y'] = df_prophet['y'].apply(lambda x: np.log1p(x))  # log1p(x) = log(x +

    # Modelo Prophet
    model = Prophet()
    model.fit(df_prophet)

    # Previsão de 1 semana
    future = model.make_future_dataframe(periods=1, freq='W')
    forecast = model.predict(future)

    # Reverte o log após a previsão
    forecast['yhat'] = forecast['yhat'].apply(lambda x: np.expm1(x))  # reverte log1p

    # Garante que os valores previstos não sejam negativos
    forecast['yhat'] = forecast['yhat'].apply(lambda x: max(0, x))  # Força yhat a ser >=

    # Pega a previsão da próxima semana (última linha)
    forecast_next = forecast[['ds', 'yhat']].iloc[-1]

    # Salva no dicionário
    forecast_dict[col] = {
        'serie': col,
        'ds': forecast_next['ds'],
        'yhat': forecast_next['yhat']
    }

# Monta o DataFrame final com os forecasts
forecast_df = pd.DataFrame.from_dict(forecast_dict, orient='index').reset_index(drop=True

# Exibe o DataFrame com as previsões para todas as séries
print(forecast_df)
```

```
df_top = df_prepared


#TOP LEVEL
df_top



#BOTTOM LEVEL
df_bottom


!pip install statsforecast hierarchicalforecast


!pip install prophet


import statsforecast
print(statsforecast.__version__)


# Assuming df_bottom contains your data with 'ds' as date and the rest as customer column
customer_columns = [col for col in df_bottom.columns if col != 'ds']

S = pd.DataFrame(np.eye(len(customer_columns)), index=customer_columns, columns=customer_
S.loc['total'] = 1  # Total sums all customers
S = S[customer_columns]

# Combine bottom and top levels
df_combined = df_bottom.copy()
# Access the 'y' column instead of 'total'
df_combined['total'] = df_top.set_index('ds')['y'].reindex(df_bottom['ds']).values

# Prepare data for Prophet
data_long = pd.melt(df_combined, id_vars=['ds'], value_vars=df_combined.columns[1:],
                    var_name='unique_id', value_name='y')
data_long['unique_id'] = data_long['unique_id'].astype(str)

print("S columns:", S.columns.tolist())
print("S index:", S.index.tolist())
print("Y_df unique_ids:", data_long['unique_id'].unique().tolist())

from prophet import Prophet

# Define the prophet_forecast function here
def prophet_forecast(df, unique_id, horizon):
    df_series = df[df['unique_id'] == unique_id][['ds', 'y']].dropna().copy()

    # Verifica se tem ao menos 2 linhas para ajustar o modelo
    if len(df_series) < 2:
        print(f"Aviso: único id '{unique_id}' tem menos de 2 linhas de dados para ajustar
        # Retorna DataFrame vazio ou um DataFrame com NaNs para não quebrar o código
        future_dates = pd.date_range(start=df_series['ds'].max() if not df_series.empty e
        return pd.DataFrame({'ds': future_dates, 'unique_id': unique_id, 'yhat': [float('

    model = Prophet()
```

```
        model.fit(df_series)
        future = model.make_future_dataframe(periods=horizon, freq='W')
        forecast = model.predict(future)
        forecast['unique_id'] = unique_id
        return forecast[['ds', 'unique_id', 'yhat']]

    horizon = 1  # Next week

    forecasts = []
    for unique_id in ['total'] + customer_columns:
        forecast = prophet_forecast(data_long, unique_id, horizon)
        forecasts.append(forecast)
    forecasts_df = pd.concat(forecasts)

    #Make sure forecasts_df is defined before you use it.
    print("Y_hat_df unique_ids:", forecasts_df['unique_id'].unique().tolist()) # Make sure fo


    forecasts_pivot = forecasts_df.pivot(index='ds', columns='unique_id', values='yhat')

    print("S columns:", S.columns.tolist())
    print("S index:", S.index.tolist())
    print("Y_df unique_ids:", data_long['unique_id'].unique().tolist())
    print("Y_hat_df unique_ids:", forecasts_df['unique_id'].unique().tolist())
    print("Y_hat_df pivot columns:", forecasts_pivot.columns.tolist())
    print("Y_df pivot columns:", data_long.pivot(index='ds', columns='unique_id', values='y')


    print("data_long shape:", data_long.shape)
    print("forecasts_pivot shape:", forecasts_pivot.shape)
    print("data_long head:\n", data_long.head())
    print("forecasts_pivot head:\n", forecasts_pivot.head())


    import pandas as pd
    import numpy as np
    from prophet import Prophet
    import matplotlib.pyplot as plt

    # Load data
    # df_bottom = pd.read_csv('bottom_level.csv', parse_dates=['ds'])
    # df_top = pd.read_csv('top_level.csv', parse_dates=['ds'])

    # Ensure 'ds' is a column and datetime
    if df_bottom.index.name == 'ds':
        df_bottom = df_bottom.reset_index()
    if df_top.index.name == 'ds':
        df_top = df_top.reset_index()
    df_bottom['ds'] = pd.to_datetime(df_bottom['ds'])
    df_top['ds'] = pd.to_datetime(df_top['ds'])

    # Convert customer column names to strings
    df_bottom.columns = [str(col) if col != 'ds' else col for col in df_bottom.columr

    # Rename top-level column
    df_top = df_top.rename(columns={df_top.columns[1]: 'total'})
```

```
# Verify data consistency
df_bottom_sum = df_bottom.drop(columns='ds').sum(axis=1)
if not np.allclose(df_bottom_sum, df_top.set_index('ds')['total'].reindex(df_bott
    print("Warning: Top-level total does not match sum of bottom-level customers.

# Combine bottom and top levels
df_combined = df_bottom.copy()
df_combined['total'] = df_top.set_index('ds')['total'].reindex(df_bottom['ds']).v

# Prepare data for Prophet (long format: unique_id, ds, y)
data_long = pd.melt(df_combined, id_vars=['ds'], value_vars=df_combined.columns[1
                    var_name='unique_id', value_name='y')
data_long['unique_id'] = data_long['unique_id'].astype(str)

# Handle missing values
if data_long['y'].isna().any():
    data_long['y'] = data_long['y'].fillna(method='ffill')

# Define the hierarchy
customer_columns = [col for col in df_bottom.columns if col != 'ds']
all_unique_ids = ['total'] + customer_columns

# Function to forecast with Prophet
def prophet_forecast(df, unique_id, horizon):
    df_series = df[df['unique_id'] == unique_id][['ds', 'y']].copy()
    model = Prophet(
        seasonality_mode='additive',
        yearly_seasonality=True,
        weekly_seasonality=True,
        daily_seasonality=False
    )
    model.fit(df_series)
    future = model.make_future_dataframe(periods=horizon, freq='W-SUN')
    forecast = model.predict(future)
    forecast['unique_id'] = unique_id
    return forecast[['ds', 'unique_id', 'yhat']]

# Forecast for all series (1-week horizon)
horizon = 4  # Next week
forecasts = []
for unique_id in all_unique_ids:
    forecast = prophet_forecast(data_long, unique_id, horizon)
    forecasts.append(forecast)
forecasts_df = pd.concat(forecasts)

# Filter to the forecast date (next week)
forecasts_df = forecasts_df[forecasts_df['ds'] > data_long['ds'].max()]
forecasts_df = forecasts_df.rename(columns={'yhat': 'Prophet'})
forecasts_df['Prophet'] = forecasts_df['Prophet'].astype(float)

# Manual BottomUp reconciliation
# For BottomUp, use bottom-level forecasts (customers) and sum for total
bottom_forecasts = forecasts_df[forecasts_df['unique_id'].isin(customer_columns)]
total_forecast = bottom_forecasts.groupby('ds')['Prophet'].sum().reset_index()
```

```
total_forecast['unique_id'] = 'total'
total_forecast['Prophet'] = total_forecast['Prophet'].astype(float)

# Combine reconciled forecasts
reconciled_forecasts = pd.concat([bottom_forecasts, total_forecast])

# Extract forecast for Customer 'X'
forecast_x = reconciled_forecasts[reconciled_forecasts['unique_id'] == 'X']

# Visualize results
plt.figure(figsize=(10, 6))
plt.scatter(forecast_x['ds'], forecast_x['Prophet'], label='Forecast for X', colo
plt.axvline(x=data_long['ds'].max(), color='gray', linestyle='--', label='Last Hi
plt.legend()
plt.title("Customer 'x' 1-Week Forecast with Prophet (BottomUp)")
plt.xlabel("Date")
plt.ylabel("Box Production")
plt.grid(True)
plt.show()

# Print forecast
print("Forecast for Customer 'x':")
print(forecast_x[['ds', 'Prophet']])
```

4. Avaliação da Previsão

```
# Filtra a linha para a data desejada
target_date = pd.to_datetime('2025-04-06')
row = forecast_x[forecast_x['ds'] == target_date]

if row.empty:
    print("Data 2025-04-06 não encontrada no DataFrame forecast.")
else:
    real_value = float(input("Valor real: "))
    predicted_value = row['Prophet'].values[0]
    error = real_value - predicted_value
    mape = abs(error / real_value) * 100
```

Anexo V – Código Python da Tarefa 5

Tarefa 5

Implementação do modelo Prophet no Fabric

1. Inicialização do Ambiente

```
from pyspark.sql import SparkSession

# Initialize Spark session
spark = SparkSession.builder.appName("ExplorarProd").getOrCreate()

# Reduce log verbosity
spark.sparkContext.setLogLevel("ERROR")

# Check current schema
print("Current schema:")
spark.sql("SELECT current_schema()").show()

# List all available schemas
print("Available schemas:")
spark.sql("SHOW SCHEMAS").show()

# List all catalogs
print("Available catalogs:")
spark.sql("SHOW CATALOGS").show()
```

2. Acesso aos Dados no Delta Lake

```
from deltalake import DeltaTable, write_deltalake
table_path = 'abfss://889d17e4-adaf-4897-9fc9-f85337241e3e@onelake.dfs.fabric.microsoft.com/ed0db12f-df4a-4549-8246-9cd5d9a0e7cd/Tables/P
storage_options = {"bearer_token": notebookutils.credentials.getToken('storage'), "use_fabric_endpoint": "true"}
dt = DeltaTable(table_path, storage_options=storage_options)

# Carregar todos os dados para pandas
full_data = dt.to_pyarrow_dataset().to_table().to_pandas()
display(full_data)
```

3. Exploração Inicial dos Dados

```
# Ver as primeiras linhas
print(full_data.head())

# Resumo das colunas, tipos de dados e valores nulos
print(full_data.info())

# Estatísticas descritivas (para colunas numéricas)
print(full_data.describe())

# Contar valores únicos em uma coluna específica (ex.: 'coluna_nome')
print(full_data['caixas'].value_counts())
```

4. Limpeza e Seleção de Colunas

```
# Listar as colunas do DataFrame limited_data
print("Colunas da tabela:", full_data.columns.tolist())


# Ocultar as colunas que não interessam
dados_tranformados = full_data.copy()

dados_tranformados = full_data.drop(columns=['OEs', 'tipo_operacao','REGIAO_AGRUP'])

# Display the updated DataFrame
print(dados_tranformados.head())


print(dados_tranformados[dados_tranformados["cod_col_sid"] == "R"].tail())
```

```
# Verificar se há valores ausentes
print(dados_tranformados.isna().sum())


# Verificar os tipos das colunas
print(dados_tranformados[['dt_fim_operacao']].dtypes)

# Verificar os primeiros registros das colunas para garantir que os dados estejam no formato correto
print(dados_tranformados[['dt_fim_operacao']].head())
```

5. Remoção de Registos Inválidos

```
# Removendo as linhas com "1970-01-01"
df = dados_tranformados[dados_tranformados["dt_fim_operacao"] != "1970-01-01"]

print(df)


# Ordenar da mais antiga para a mais recente
df = df.sort_values(by='dt_fim_operacao')
```

6. Filtragem por Armazém

```
# Filtrar onde a coluna COL é igual a 'A'
df_a = df[df['cod_col_sid'] == 'A']

print(df_a)
```

7. Análise de Média e Desvio

```
!pip install statsmodels


!pip install seaborn


#import libraries
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from pandas.plotting import autocorrelation_plot
from pandas import DataFrame
from pandas import concat
import numpy as np
from math import sqrt

from sklearn.metrics import mean_squared_error
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.stattools import acf
from statsmodels.tsa.stattools import pacf
from statsmodels.tsa.arima_model import ARIMA
from scipy.stats import boxcox

import seaborn as sns
sns.set_style("whitegrid")
import matplotlib.pyplot as plt
from matplotlib.pylab import rcParams
from matplotlib import colors
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')


#DF_A

# Aplique o rolling apenas na coluna numérica "Caixas"
rolling_mean = df_a['caixas'].rolling(window=12).mean()
rolling_std = df_a['caixas'].rolling(window=12).std()

plt.figure(figsize=(20, 6))
plt.plot(df_a['dt_fim_operacao'], df_a['caixas'], color='cornflowerblue', label='Original')  # Usando a coluna de data para o eixo X
plt.plot(df_a['dt_fim_operacao'], rolling_mean, color='firebrick', label='Rolling Mean')
plt.plot(df_a['dt_fim_operacao'], rolling_std, color='limegreen', label='Rolling Std')
```

```
# Adicionando rótulos e título
plt.xlabel('Date', size=12)
plt.ylabel('Caixas', size=12)
plt.legend(loc='upper left')
plt.title('Box Production R', size=14)

# Exibindo o gráfico
plt.show()
```

```
contagem = df_a['caixas'].apply(lambda x: pd.isna(x) or x == 0 or str(x).strip() == '').sum()

print(f"Número de valores 0, nulos ou em branco: {contagem}")
```

8. Agregação por Semana

```
# Remover as colunas indesejadas
df_a.drop(columns=["regiao_dist", "cod_col_sid", "cod_cliente"], inplace=True)

# Definir a coluna de datas como índice
df_a.set_index("dt_fim_operacao", inplace=True)

# Agregar por semana somando as "Caixas"
df_weekly = df_a.resample('W-SUN').sum()

# Exibir as primeiras linhas
print(df_weekly.head())
```

```
df_validacao = df_weekly.copy()
```

```
#Eliminar a última linha de df_weekly
df_weekly = df_weekly.iloc[:-1]
```

```
df_weekly
```

9. Formatação para o Prophet

```
df_weekly
```

```
# Criar uma cópia do DataFrame df_weekly para df_weeklyz
df_weeklyz = df_weekly.copy()

# Resetar o índice e adicionar como uma nova coluna
df_weeklyz = df_weeklyz.reset_index()

# Mostrar as primeiras linhas do novo DataFrame
print(df_weeklyz.head())
```

```
df_weeklyz = df_weeklyz.rename(columns={'dt_fim_operacao': 'ds', 'caixas': 'y'})
print(df_weeklyz.head())
```

10. Criação e Ajuste do Modelo Prophet

```
!pip install prophet
```

```
from prophet import Prophet
```

```
# Criar uma cópia do DataFrame df_weekly para df_weeklyz
df_weeklyz = df_weekly.copy()

# Resetar o índice e adicionar como uma nova coluna
df_weeklyz = df_weeklyz.reset_index()

# Mostrar as primeiras linhas do novo DataFrame
print(df_weeklyz.head())
```

```
df_weeklyz
```

```
# Criar o DataFrame com os dados até 2025-03-30
data = df_weeklyz

data['dt_fim_operacao'] = pd.to_datetime(data['dt_fim_operacao'])


# Renomeando as colunas
data = data.rename(columns={'dt_fim_operacao': 'ds', 'caixas': 'y'})


data
```

11. Previsão da Próxima Semana

```
data['ds'] = data['ds'].dt.tz_localize(None)

# Ajustar o modelo com todos os dados até 2025-03-30
model = Prophet(weekly_seasonality=True, yearly_seasonality=True, daily_seasonality=False)
model.add_country_holidays(country_name='PT')
model.fit(data)

# Prever a próxima semana
future_dates = model.make_future_dataframe(periods=1, freq='W-SUN')
forecast = model.predict(future_dates)

next_weeks_forecast = forecast[forecast['ds'] == '2025-06-08'][['ds', 'yhat', 'yhat_lower', 'yhat_upper']]

print("Previsão para as próximas semana:")
print(next_weeks_forecast)
```