PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA
Pós-graduação *Lato Sensu* em Ciência de Dados e Big Data

Ermivaldo Cunha de Jesus Júnior

ANÁLISE DE MODELOS PREDITIVOS APLICADOS EM SÉRIES TEMPORAIS

Belo Horizonte
2021

Ermivaldo Cunha de Jesus Júnior

ANÁLISE DE MODELOS PREDITIVOS APLICADOS EM SÉRIES TEMPORAIS

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Ciência de Dados e Big Data como requisito parcial à obtenção do título de especialista.

Belo Horizonte
2021

SUMÁRIO

1. Introdução

Uma série temporal é composta por uma sequência de observações determinadas em intervalos sequenciais ao longo tempo. A característica que mais se destaca neste tipo de observação é a interdependência destes dados, ou seja, a dependência que cada observação tem com as observações vizinhas. Podemos observar que em uma série temporal, cada observação é relacionada a um ponto no tempo, sendo assim é importante considerar a ordem de distribuição desses dados.

A análise das séries temporais baseia-se no uso de modelos matemáticos e estatísticos objetivando compreender e quantificar os fenômenos de variação temporal. Essa análise ocorre através da seleção de dados passados para realizar a predição de dados futuros, visando através dessa análise, construir modelos que permitem prever a série temporal futura (OLIVEIRA, 2007).

Para realizar nossas análises precisamos utilizar algumas técnicas e validações específicas, pois cada série pode apresentar padrões de comportamentos distintos como de tendência, ordem cíclica, dados estacionários, sazonalidade e etc.

Em alguns modelos de previsão é necessário que a série a ser prevista seja estacionária, pois esta característica facilita na projeção dos dados futuros. Neste trabalho será aplicado o deste de Dickey-Fuller ou ADF (*Augmented Dickey-Fuller*).

Em nossas análises vamos aplicar os seguintes modelos de previsão o Naive, Auto Regressivo, ARMA, ARIMA, SARIMA, Facebook Prophet e o modelo Holt-Winters.

O modelo Naive é um modelo mais simples, utilizaremos apenas para definição de um nível mínimo aceitável de assertividade. Este modelo consiste que o valor da série no instante t+1 é o mesmo valor da observação anterior.

Em um modelo Auto Regressivo (AR) é realizado a previsão da variável de interesse a partir da combinação linear dos valores passados desta mesma variável, por isto o nome auto regressivo, pois indica que a previsão é obtida por meio da regressão de uma variável por ela mesma.

O ARMA é composto pelo modelo Auto Regressivo (AR) com ajuste dos erros residuais o componente de Média Móvel (MA). Neste modelo é considerado a média móvel para suavizar a tendência e aplicar a regressão. Dessa forma teremos que passa o parâmetro que a ser considerado para componente de média móvel.

O Modelo ARIMA é a combinação de três componentes o componente auto regressivo (AR), componentes de integração (I) e o de médias móveis (AM). O ARIMA foi o modelo proposto por Box e Jenkins. Ele se baseia na teoria de modelagem estatística, onde é usado um algoritmo para encontrar um modelo com o menor número de erros possível, ou seja, mais acurado. Ele utiliza a correlação dos valores da série com os valores passados, a fim de capturar o seu comportamento (WERNER, 2005).

O modelo SARIMA (*Sazonal Autoregressive Integrated Moving Average*) considera os componentes (S) componente de sazonalidade, (AR) auto regressivo, (I) integração e (MA) médias móveis. Box and Jenkins (1970) generalizaram o modelo ARIMA para lidar com sazonalidade e definiram um modelo ARIMA sazonal multiplicativo, denominado SARIMA.

O Facebook Prophet tem por base o modelo aditivo e por padrão utiliza o modelo de previsão linear, porém podemos configurá-lo para utilizar a função logística. Este é um modelo que lida bem com grande volume de dados e com fortes características de sazonalidade e tendência, um modelo de fácil utilização e costuma realizar previsões com o mínimo de ajuste de seus parâmetros. Este modelo consegue ajustar bem os dados mais distantes das médias a sazonalidade e é robusto para o tratamento de dados ausentes.

O modelo Holt-Winters ES ou Suavização Exponencial de Holt-Winters, como o próprio nome sugere, é um modelo que utiliza a técnica de suavização exponencial. A suavização exponencial é uma técnica que realiza uma média ponderada onde os pesos que vão diminuindo exponencialmente, dos valores mais recentes para os mais antigos para assim realizar sua previsão, sendo assim, este modelo define que as observações mais recentes possuem um grau de relevância maior em sua previsão. O modelo Holt-Winters consiste na adição de técnicas para correção de falhas da suavização exponencial para tratamento de tendências e sazonalidades.

Por fim, temos a definição dos métodos de avaliação da acurácia, para tanto a escolha do modelo se dará, dentre outros critérios, pelo modelo que apresentar um melhor nível acurácia, ou seja, o modelo que tiver um menor índice de erros. Por isso é muito importante quantificar o nível de acurácia que cada modelo apresentar, para saber o nível de qualidade obtida.

Para definir os níveis de acurácia vamos aplicar algumas medidas de avaliação nos erros obtidos, que são elas; MAE (*Mean Absolute Error*), MAPE (*Mean Absolute*

Percentage Error), MSE (*Mean Squared Error*) e RMSE (*Root Mean Square Error*) em todos os modelos para avaliarmos o melhor modelo empregado nos dados observados.

O MAE, ou erro absoluto médio, é obtido a partir da soma dos valores dos erros absolutos dividido pela quantidade de registros, esta medida é de fácil interpretação, pois o seu produto vai estar na mesma unidade de medida do modelo testado.

O MAPE, ou média do percentual dos erros em valores absolutos, é obtido a partir da soma de todos os percentuais de erro, em valores absolutos, e divide-se pela quantidade de registros, resultando em uma média do percentual. Esta é muito famosa por ser uma medida de fácil explicação e entendimento, pois se trata de um valor em percentual, padrão muito utilizado.

O MSE, ou erro médio quadrático, neste método eleva-se ao quadrado o valor do erro e calcula-se o valor médio, este se torna muito sensível a erros que estiverem muito distantes da média, pois quando se eleva ao quadrado colocamos um maior peso nos erros com dispersão maior, importante ressaltar que os valores obtidos não estarão na mesma unidade do modelo testado.

O RMSE, ou raiz da média quadrática do erro, é basicamente o desvio padrão dos erros, eleva-se ao quando os erros, logo após calcula-se a média e aplica-se a raiz quadrada, ou seja, é a aplicação da raiz quadrada ao modelo MSE, dessa forma também possui a característica de enfatizar os valores mais distantes da média, porém ao final é anula-se a potencialização com a aplicação da raiz, retornando os valores a mesma unidade do modelo testado.

1.1. Contextualização

Com o advento da internet a comercialização por meio de lojas virtuais tem crescido exponencialmente, com isso tem surgido novas metodologias para efetivação de transações comercias, sendo uma delas a evolução do e-commerce os denominados *marketplaces*. Os marketplaces são plataformas de lojas virtuais que disponibilizam a lojistas sua estrutura para venda de produtos, atuando como uma grande vitrine de shopping center virtual, podendo assim oferecer em um único ambiente diversos produtos de variados fornecedores, entregando a facilidade para o cliente final de ter um único carrinho de compra e realizar somente um pagamento.

De um lado temos uma grande loja virtual que já possui uma credibilidade de mercado e uma estrutura robusta com acesso a milhares de clientes, do outro lado o lojista, que irá utilizar toda esta estrutura para comercializar seus produtos, normalmente pagando um comissionamento pela venda para o *marketplace*.

Este trabalho visa analisar a base de dados de vendas de uma empresa startup brasileira que atua no setor de comercio eletrônico por meio dessas plataformas de *marketplaces*.

A empresa analisada é um e-commerce, que concentra vendedores que desejam anunciar seus produtos por meio dessas plataformas de *marketplaces*, atuando como uma ponte para facilitar o acesso destes vendedores ao consumidor final em forma de uma loja única nas mais variadas plataformas de comércio eletrônico. Ela conta com mais de 300 colaboradores e possui uma carteira com mais de 9 mil lojistas que já atendem mais de 2 milhões de consumidores únicos.

Nesta oportunidade foi analisado em torno de 99 mil registros de pedidos de compras entre os anos de 2016 e 2018. Este trabalho visa oferecer um modelo preditivo que possibilite ter uma previsibilidade do volume de pedidos que serão efetuados por meio de sua plataforma, com o maior nível de acurácia, a fim de ser utilizado para prever os períodos subsequentes.

Por se tratar de um conjunto de dados que é ordenado sequencialmente no tempo, temos uma análise de séries temporais, onde iremos identificar um comportamento com base nos dados históricos que será extrapolado para o próximo período, ou seja, fazer uma previsão de qual será o seu comportamento no futuro com base nos eventos que aconteceram no passado.

Para isso iremos utilizar modelos estatísticos de análise de séries temporais para avaliar os dados observados, decompondo a tendência, a sazonalidade e os ruídos. Por fim aplicaremos os modelos preditivos a fim de realizar previsões de comportamento futuro, com o melhor índice de acerto possível.

1.2. O problema proposto

O problema proposto encontrar um modelo que consiga entregar uma previsão de pedidos que serão realizados na plataforma, com o maior nível de qualidade possível.

A abordagem deste tema foi selecionada, por ser uma problemática comum no mercado. Ter a previsibilidade do comportamento de séries temporais é de extrema importância para poder direcionar a gestão em diferentes frentes. No segmento comercial ter a previsão da demanda facilita a tomada de decisão, a elaboração de orçamentos, ter perspectiva de faturamento, ou até mesmo, determinar a estrutura que será necessária para suportar um fluxo futuro. Sendo assim, a análise de séries temporais é de extrema importância no mundo dos negócios, além de ser muito útil em análises de dados clínicos, meteorológicos, de sensores, industriais e etc.

Dessa forma, será observado a base de dados com os registros de pedidos da empresa. As bases de dados foram disponibilizadas pela própria empresa na plataforma de competições para cientistas de dados, site Kaggle, os dados estão disponibilizados publicamente e já estão devidamente anonimizados.

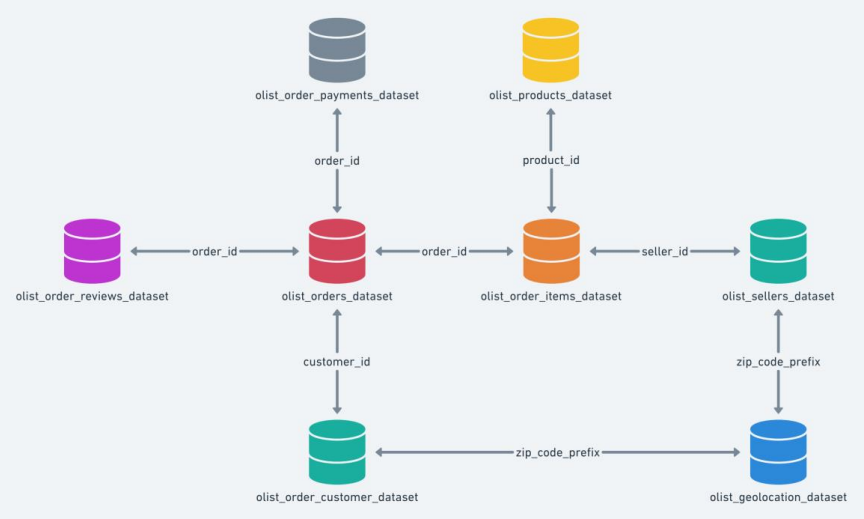
Este trabalho tem por objetivo, analisar os dados de pedidos realizados. Extraindo os comportamentos de tendência, sazonalidade e ruídos, realizando testes e aplicando modelos de previsões, a fim de realizar projeções para períodos futuros.

2. Coleta de Dados

Para desenvolvimento deste projeto foi coletado os dados do site Kaggle, em 06/07/2020, onde foram disponibilizados pela própria empresa. Iremos utilizar os Datasets com os dados de pedidos, serão 99.441 (noventa e nove mil e quatrocentos e quarenta e uma) linhas de pedidos, com 112.650 (cento e doze mil e seiscentos e cinquenta) linhas de itens de pedidos e com 32.951 (trinta e duas mil e novecentas e cinquenta e uma) linhas de produtos.

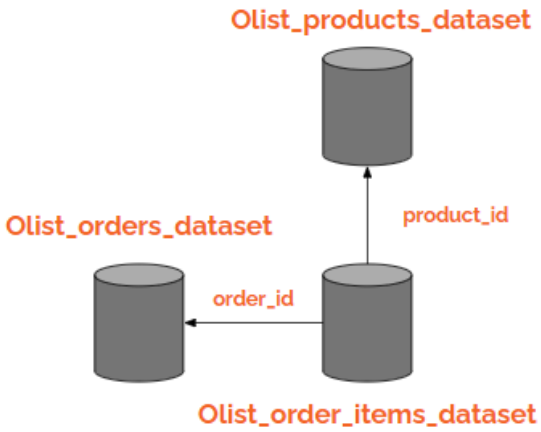
Na Figura 1, é possível visualizar a abaixo a modelagem da base de dados disponibilizada. Enquanto na Figura 2 é possível ver a modelagem das bases que serão utilizadas neste trabalho.

Figura 1: Modelagem da Base de Dados da Empresa.



Fonte: Kaggle.com

Figura 2: Modelagem da Dase de Dados Utilizada.



Fonte: Autor

2.1 Descrição das bases disponibilizadas

O Dataset Olist_orders_dataset possui dados do pedido como; identificador, data de pedido, data de aprovação, data de entrega e status. Este Data Set possui 99.441 linhas com 8 colunas. Todos os registros possuem os dados de identificador do pedido, identificador do cliente, status do pedido e data de realização.

Nome da coluna/campo	Descrição	Tipo
order_id	Identificador do pedido	object

customer_id	Identificador do cliente	object
order_status	Dados categóricos com o status do pedido, opções disponíveis; **delivered, shipped, canceled, unavailable, invoiced, processing, created, approved.**	object
order_purchase_timestamp	Data e hora de realização do pedido	object
order_approved_at	Data e hora de aprovação do pedido	object
order_delivered_carrier_date	Data e hora de entrega do pedido para a transportadora	object
order_delivered_customer_date	Data e hora de entrega do pedido para o cliente	object
order_estimated_delivery_date	Data e hora estimada de entrega do pedido	object

O Dataset Olist_order_items_dataset possui dados referente ao item do pedido, assim como; identificador do pedido, numeração do item no pedido, identificador do produto, identificador do vendedor, data limite de entrega, valor e valor do frete. Um pedido pode ter mais de um item, dessa forma o identificador do pedido não será único, porém um item pertence a apenas um pedido.

Este Dataset possui 112.650 linhas e 7 colunas.

Nome da coluna/campo	Descrição	Tipo
order_id	Código identificador do pedido	object
order_item_id	Número do item do pedido	int64
product_id	Código identificador do produto	object
seller_id	Código identificador do lojista/vendedor	object
shipping_limit_date	Data e hora limite de entrega	object
price	Valor total do item	float64
freight_value	Valor do frete do item	float64

O Dataset Olist_products_dataset possui dados referente ao produto, assim como; identificador do pedido, nome da categoria do produto, tamanhos da descrição e do nome do produto, quantidade de fotos cadastradas e medidas de tamanho dos

produtos, altura, largura e cumprimento. Um pedido pode ter mais de um item, dessa forma o identificador do pedido não será único, porém um item pertence a apenas um pedido.

Este Dataset possui 32.951 linhas e 9 colunas.

Nome da coluna/campo	Descrição	Tipo
product_id	Código identificador do produto	object
product_category_name	Nome da categoria do produto	object
product_name_lenght	Tamanho do nome do produto	float64
product_description_lenght	Tamanho da descrição do produto	float64
product_photos_qty	Quantidade de fotos do produto	float64
product_weight_g	Peso do produto	float64
product_length_cm	Comprimento do produto	float64
product_height_cm	Altura do produto	float64
product_width_cm	Largura do produto	float64

3. Processamento/Tratamento de Dados

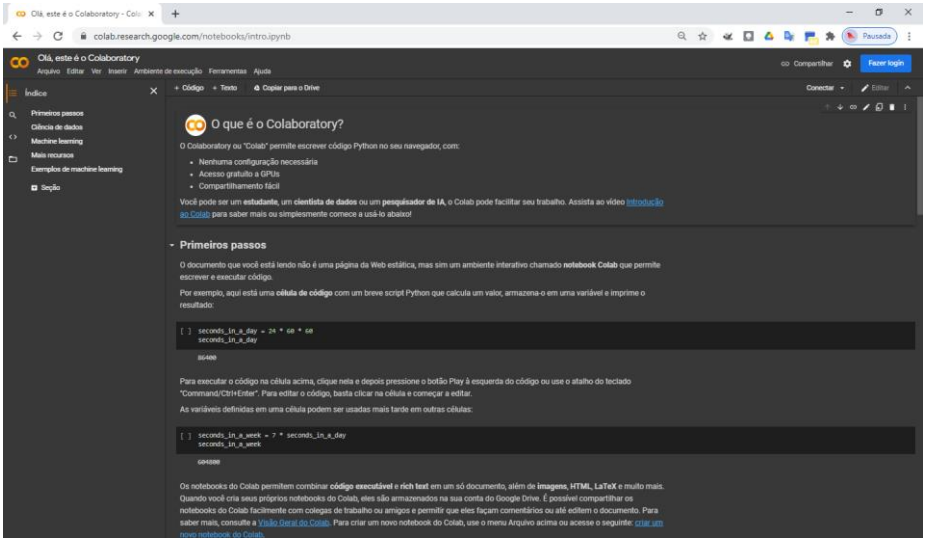
3.1 Ferramentas utilizadas

Foi escolhido a linguagem de programação Python, para criação dos scripts de análise de dados.

Neste projeto foi utilizado a ferramenta Google Colab para criação e execução dos scripts em Python. O Google Colab é uma plataforma de notebooks online que permite a criação e execução de scripts em Python utilizando os servidores do Google para o processamento. Disponível em https://colab.research.google.com/ .

Na figura 3, foi disponibilizado uma visualização da tela inicial do Google Colab.

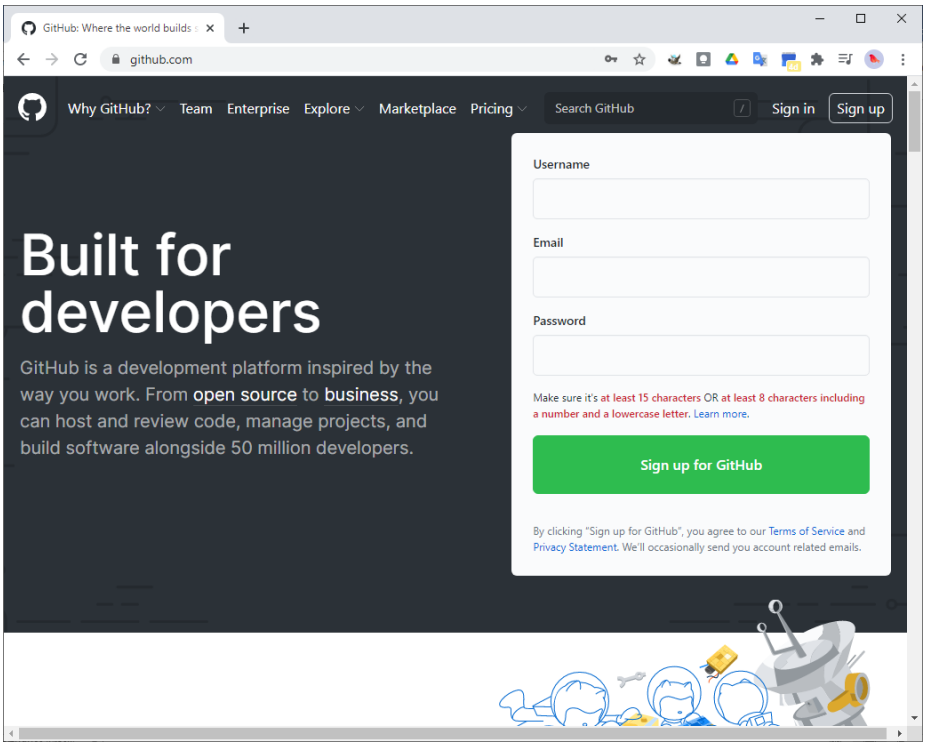
Figura 3: *Screenshot* da Tela Inicial do Google Colab.



Fonte: Autor

Para armazenamento e versionamento dos notebooks foi utilizado a ferramenta GitHub. Disponível em https://github.com/. O GitHub é uma plataforma de desenvolvimento para código aberto ou para negócios, ele nos permite desenvolver softwares, gerenciar projetos, hospedar e revisar código.

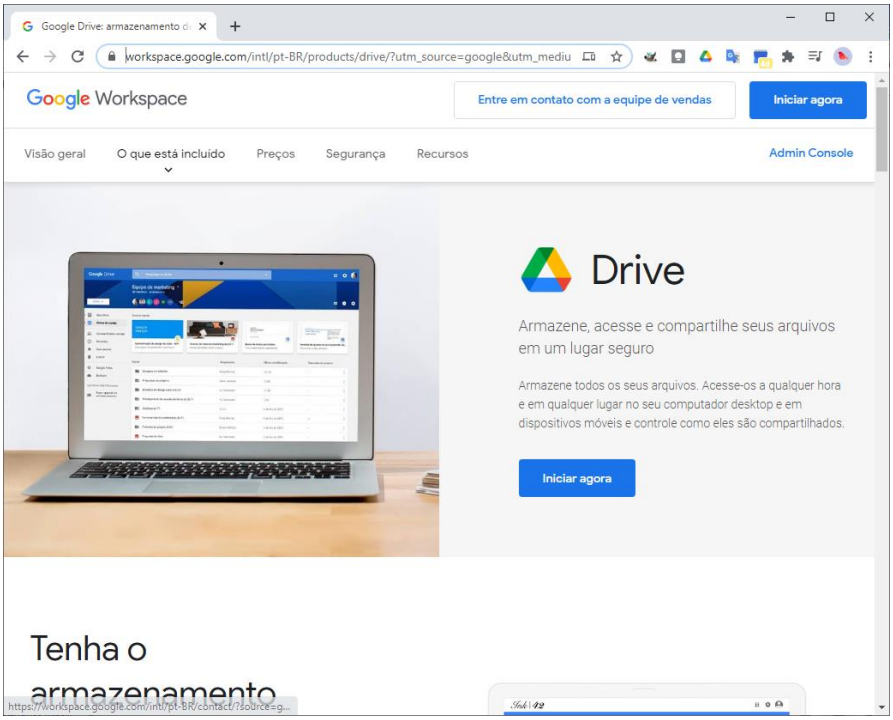
Figura 4: *Screenshot* da Tela Inicial do GitHub.



Fonte: Autor

Para o armazenamento dos Data Sets foi utilizado o Google Drive. O Google Drive é uma plataforma que possibilita o armazenamento de arquivos fotos, textos e planilhas entre outros. Disponível em https://workspace.google.com/.

Figura 5: *Screenshot* da Tela Inicial do Google Drive.



Fonte: Autor

3.2 Tratamento dos Dados

3.2.1 Colunas do tipo de dados data

Como iremos trabalhar com séries temporais foi necessário no Data Set de pedidos fazer a conversão do tipo de dados da coluna order_purchase_timestamp que estava com o tipo de dados “object”, para o tipo de dados “datetime”, possibilitando uma correta utilização para as análises de séries temporais, com isso foram criadas as seguintes colunas order_purchase_date e year_month, uma para armazenamento da data e outra consolidação do ano com o mês, respectivamente. Para o preenchimento da coluna order_purchase_date foi utilizado uma função do pacote Pandas chamada to_datetime que retorna uma coluna com o tipo de dados “timestamp”, na sequência foi utilizado esta nova coluna para preenchimento da coluna year_month, neste caso foi utilizado o método strftime(), do objeto datetime do

Python, para retornar um texto a partir da data do pedido, para escolha do formato de data escolhido, YYYY-MM, foi necessário informar o parâmetro “%Y-%m” para o método.

Figura 6: Tratamento da Variável Data.

```
##Cconverter coluna order_purchase_timestamp em datetime
df_orders['order_purchase_timestamp'] = pd.to_datetime(df_orders['order_purchase_timestamp'])
##Criar coluna com a data de venda
df_orders['order_purchase_date']=df_orders['order_purchase_timestamp'].dt.date
##Utilizar a coluna criada para criar a coluna year_month padrão YYYY-MM
df_orders['year_month']=df_orders['order_purchase_timestamp'].dt.strftime("%Y-%m")
df_orders['year']=df_orders['order_purchase_timestamp'].dt.year
df_orders['monthday']=df_orders['order_purchase_timestamp'].dt.day
df_orders['month']=df_orders['order_purchase_timestamp'].dt.month
df_orders['year_month_day']=df_orders['order_purchase_timestamp'].dt.strftime("%Y-%m-%d")
df_orders['year_week']=df_orders['order_purchase_timestamp'].dt.strftime('%Y-%W')

df_orders[['order_purchase_timestamp','order_purchase_date','year_month','year','monthday','month','year_month_day','year_week']].head(5)
```

Fonte: Autor

3.2.2 Consolidação do valor total do pedido

Na tabela de pedidos não veio a informação do valor total, então para consolidar foi preciso obter o valor total do pedido a partir da somatória dos seus itens, para isso foi necessário agrupar os itens por pedido e efetuar a somatória de seus valores gerando um novo Dataset, posteriormente foi realizado uma junção deste novo Dataset com o valor com o Dataset de pedidos.

Figura 7: Somatória dos Valores dos Itens, para Obter o Total do Pedido.

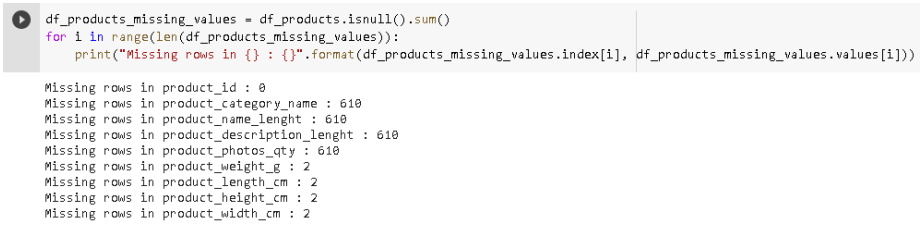
```
#definir a coluna order_id do df_orders como index
df_orders.set_index('order_id', inplace = True)
##somar a coluna price do df_items
df_001 = df_items[['order_id','price']].groupby(['order_id']).sum()

##consolidar o valor dos itens na venda
df_orders =  pd.concat([df_orders, df_001], axis=1).reindex(df_orders.index)
df_orders.head(5)
```

Fonte: Autor

3.2.3 Dados Nulos/Faltantes

No Dataset de Produto foi apurado alguns dados faltantes com podemos ver na Figura 8, porém a única informação que poderia afetar nossas análises é a informação do nome da categoria do produto, sendo assim avaliaremos o impacto.

Figura 8: Dados Nulos/Faltantes – Dataset Produtos.



```
df_products_missing_values = df_products.isnull().sum()
for i in range(len(df_products_missing_values)):
    print("Missing rows in {} : {}".format(df_products_missing_values.index[i], df_products_missing_values.values[i]))
```

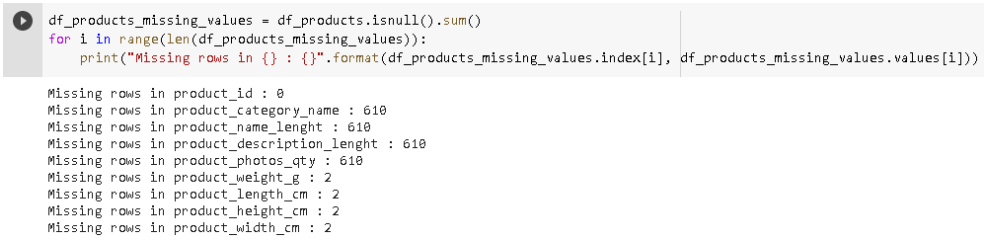
```
Missing rows in product_id : 0
Missing rows in product_category_name : 610
Missing rows in product_name_lenght : 610
Missing rows in product_description_lenght : 610
Missing rows in product_photos_qty : 610
Missing rows in product_weight_g : 2
Missing rows in product_length_cm : 2
Missing rows in product_height_cm : 2
Missing rows in product_width_cm : 2
```

Fonte: Autor

Referentes aos 610 produtos sem categorias, foram registados R$ 179.535,28 em pedidos este valor tem a representatividade de 1,32% de nossos pedidos. Vamos colocar como categoria “Outros” pois assim não irá impactar em nossas análises.

Foi verificado a existência de dados ausentes, no Dataset de pedidos, porém só foram localizados a existência de valores nulos nas colunas referentes a entrega e aprovação dos pedidos. Neste caso não houve a necessidade de tratamento, já que não utilizaremos estas variáveis em nossas análises.

Figura 9: Dados Nulos/Faltantes – Dataset Pedidos.



```
df_products_missing_values = df_products.isnull().sum()
for i in range(len(df_products_missing_values)):
    print("Missing rows in {} : {}".format(df_products_missing_values.index[i], df_products_missing_values.values[i]))
```

```
Missing rows in product_id : 0
Missing rows in product_category_name : 610
Missing rows in product_name_lenght : 610
Missing rows in product_description_lenght : 610
Missing rows in product_photos_qty : 610
Missing rows in product_weight_g : 2
Missing rows in product_length_cm : 2
Missing rows in product_height_cm : 2
Missing rows in product_width_cm : 2
```

Fonte: Autor

3.3 Bibliotecas Utilizadas

Neste trabalho foram utilizadas algumas bibliotecas escritas em Python, que são elas:

Pandas – Usada para manipulação e análise de dados, seu nome é derivado o termo “Painel de Dados”, muito utilizado em estatística e econometria. Ela permite a importação grandes massas de dados de arquivos e sua manipulação com facilidade. Com ela podemos usar objetos que nos permitem organizar os dados de forma tabular, com estruturas matriciais que possuam uma única dimensão, usando as “Series”, ou com N dimensões utilizando o “DataFrame” e já possuem uma indexação integrada, também nos permite fazer reformatação de matrizes e inversão de eixos

entre coluna e linhas ou inversamente, além disso é um software livre de código aberto. Possui funções limpeza de dados, fusão (*merge*) e junção (*joins*) conjuntos de dados, além de funcionalidades para séries temporais como geração de intervalos de datas e conversão de frequências entre outras. Além dos objetos e suas funções utilizamos a função “lag_plot” do pacote “plotting”, incluso no Pandas, para apresentação do gráfico de auto correlação.

Numpy – O Numpy foi utilizado como base para criação do Pandas, é uma ferramenta muito útil para aplicação de fórmulas matemáticas, aritméticas e estatísticas, operação com *arrays* multidimensionais e matrizes. Fornece a criação de matrizes com números aleatórios.

Matplotlib – Possui recursos básicos para criação de representações gráficas.

Seaborn – Possui recursos para criação de gráficos mais elaborados e para fins estatísticos, foi desenvolvido a partir do Matplotlib.

Statsmodels – Fornece classes e funções para estimativa e aplicação de muitos modelos estatísticos para realização de análises e exploração de dados. Entre as ferramentas disponibilizadas estão desde testes estatísticos, alguns foram utilizados neste trabalho como teste de auto correlação, correlação parcial, teste para definir se a série é estacionária ou não, até modelos de aprendizado de máquina, neste trabalho foi implementado alguns deles o AutoReg, ARMA, ARIMA, SARIMAX e Holt-Winters.

Sklearn - É uma biblioteca utilizada para aplicação de modelo de aprendizado de máquina. A biblioteca Skelearn também nos oferece o módulo metrics, que além de outras métricas, possui métricas para avaliação de desempenho de previsões. Sendo assim, vamos utilizar algumas delas, que são os métodos mean_absolute_error e mean_squared_error.

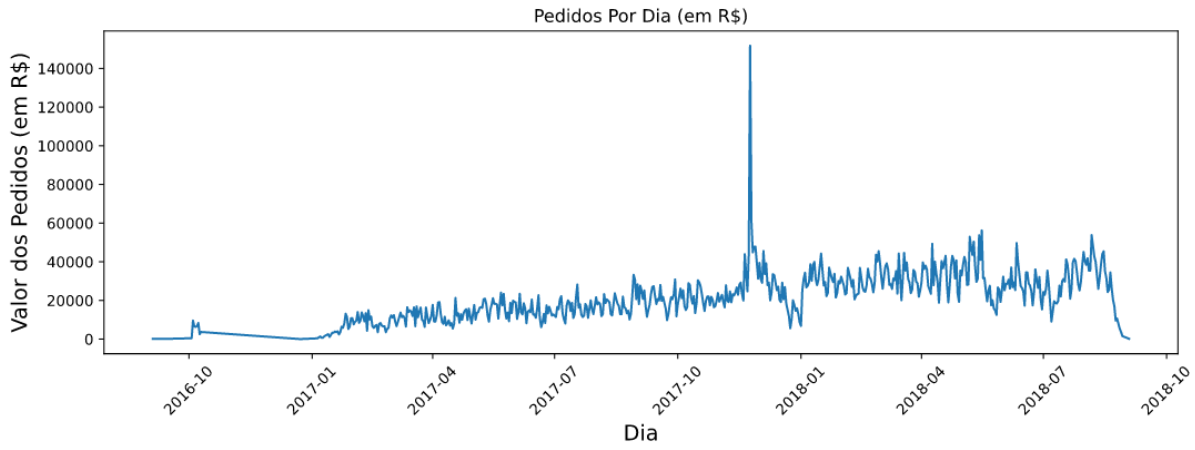
Pmdarima - É uma biblioteca que possui uma ferramenta que facilita na escolha dos parâmetros como o melhor ajuste para implementação dos modelos ARIMA. Além disso, esta biblioteca tem vários testes e também o estimador ARIMA.

Fbprophet - É a biblioteca que fornece o modelo de previsão Facebook Prophet.

4. Análise e Exploração dos Dados

Iniciando as análises de pedidos foi criado um gráfico para exibição da série temporal completa com os pedidos realizados e a soma dos pedidos realizados por dia. Veja na Figura 10.

Figura 10: Análise e Exploração dos Dados – Soma de Pedidos Realizados por Dia.



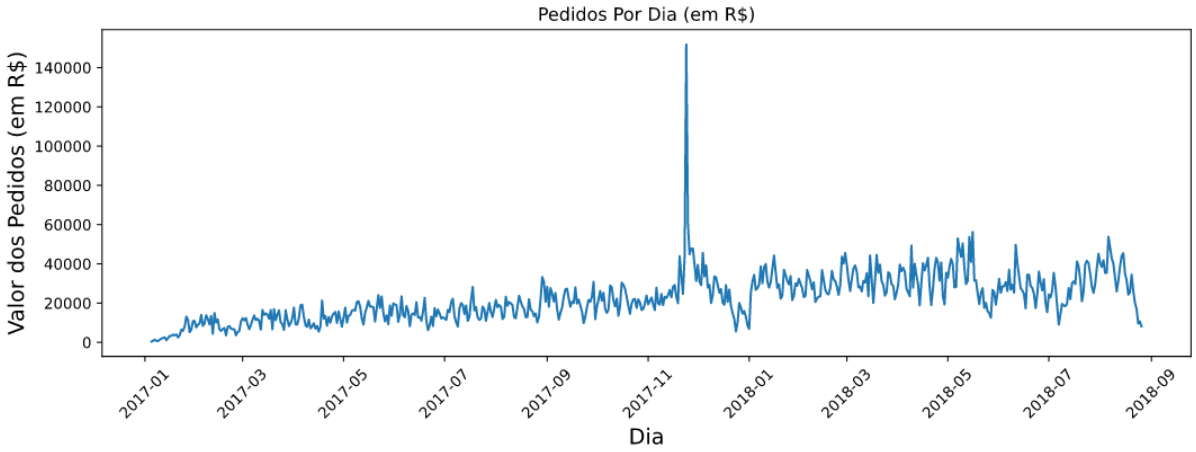
Fonte: Autor

Nesta representação gráfica, observamos que temos alguns valores muito discrepante no ano 2016 e no mês de setembro de 2018. Dessa forma, vamos expurgar de nossos dados para estudo os registros referentes a estes períodos, a fim de evitar alterações no resultado das análises ou previsões equivocadas, posteriormente cabe uma análise mais aprofundada juntamente ao fornecedor dos dados para descobrir o motivo destas discrepâncias.

Podemos observar que os valores de pedidos apresentam uma tendência aparentemente crescente e também uma possível sazonalidade.

Vejamos agora a representação diária sem estes dados discrepantes.

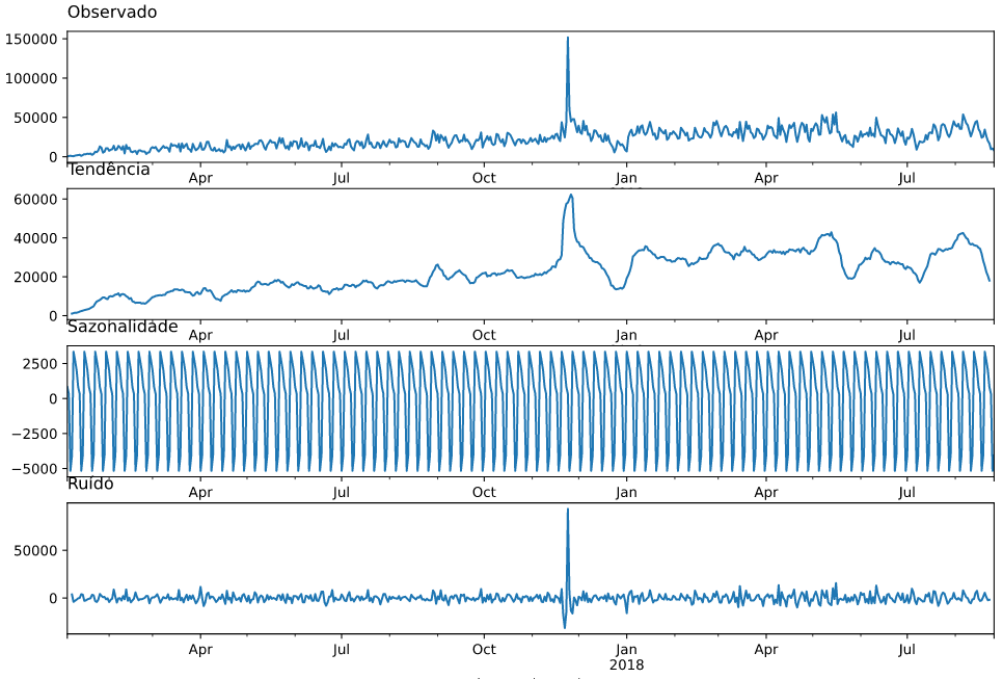
Figura 10: Análise e Exploração dos Dados – Série Temporal que Será Analisada.



Fonte: Autor

A fim de isolar os componentes da série analisada, vamos utilizar o procedimento de decomposição, para poder visualizar os padrões de sazonalidade, tendência e ruídos.

Figura 11: Análise e Exploração dos Dados – Gráficos de Decomposição dos Componentes de Tendência, Sazonalidade e Ruídos.



Fonte: Autor

Podemos observar que temos um comportamento forte de sazonalidade e um leve comportamento de tendência, acredito que o comportamento de tendência está amortizado na escala do gráfico por causa deste comportamento atípico, ou ruído, mais a frente vamos aprofundar neste ponto.

4.1. Levantamento de Questões

As questões levantadas foram as seguintes;

- Qual é o status com a maior quantidade de vendas e quantos pedidos temos com este status;
- Para quantas categorias de produtos, foi aberto pelo menos um pedido;
- Qual foi o melhor dia de vendas e quanto foi vendido neste dia;
- Qual o dia de semana com maior venda;
- Qual foi o melhor mês de vendas e quanto foi vendido neste mês;

4.1.1. Qual é o status com a maior quantidade de vendas e quantos pedidos temos com este status;

Temos 7 tipos de status categóricos na variável “order_status”. São eles; “delivered”, “shipped”, “canceled”, “invoiced”, “processing”, “unavailable” e approved.

Figura 12: Levantamento de Questões – Pedidos por status.

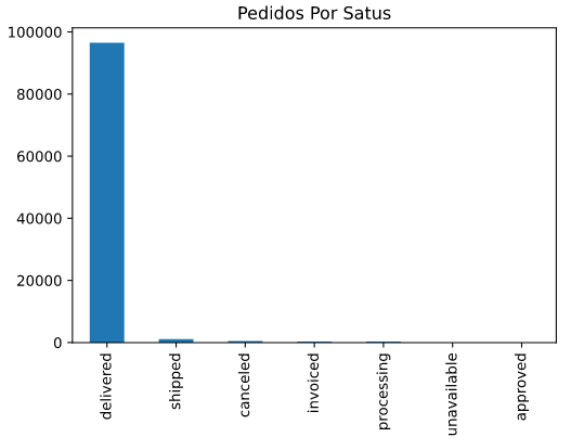
```
orders_status = pd.value_counts(df_orders['order_status'])
orders_status
```

```
delivered      96478
shipped         1106
canceled         461
invoiced         312
processing       301
unavailable        6
approved           2
Name: order_status, dtype: int64
```

Fonte: Autor

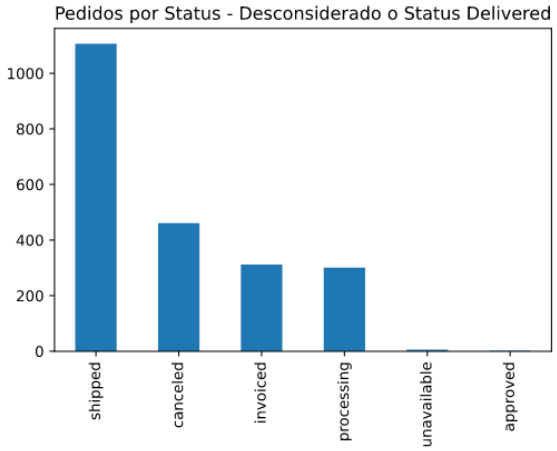
Dado a variável categórica “order_status” ou status do pedido, vamos observar a sua distribuição.

Figura 13: Levantamento de Questões – Visualização Gráfica Pedidos por Status.



Fonte: Autor

Figura 14: Levantamento de Questões – Status com poucos pedidos.



Fonte: Autor

Como podemos observar, existe uma grande concentração de pedidos com o status “delivered”, sendo assim, geramos uma representação gráfica na Figura 13 para fazer uma apresentação visual destes dados, com esta visualização podemos notar que além da grande quantidade de pedidos marcados como “delivered”, existe também uma pequena quantidade de pedidos com outros status, para uma melhor visualização foi gerado um segundo gráfico sem os pedidos marcados com “delivered”, na Figura 14. Observamos dentre estes demais status, pedidos que estão marcados como “canceled”, “processing” e “unavailable”, iremos considerar que estes pedidos não foram concluídos, pois estão marcados como “cancelado”, “em processamento” e “indisponível”, respectivamente. Dessa forma, será considerado como pedido concluído, apenas os pedidos marcados com os status “delivered”,

“shipped”, “invoiced” e “approved”. Com esta alteração foram passamos a ter 111.744 registros no Dataset de itens e 97.898 registros no Dataset de pedidos.

4.1.2. Para quantas categorias de produtos, foi aberto pelo menos um pedido;

Temos 74 categorias associadas aos produtos, todos os produtos possuem pelo menos um item de pedido, a categoria com o menor número de pedidos é a categoria “seguros_e_servicos” e a categoria associada ao maior número de itens é a categoria “cama_mesa_banho” com 11.089 itens associados.

Abaixo algumas estatísticas do agrupamento de itens por categoria de produtos.

Figura 15: Levantamento de Questões – Quantidade de Itens Vendidos por Categoria.

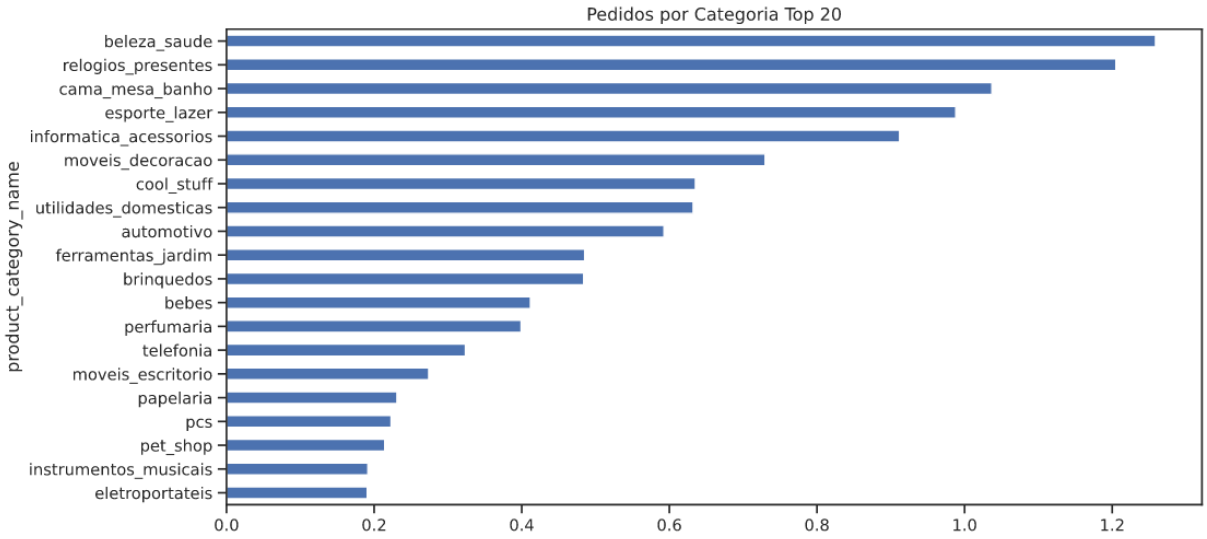
```
orders_products_category = pd.value_counts(df_items_filtred['product_category_name'])
orders_products_category = orders_products_category.sort_values(ascending=False)
print("Temos {} itens em {} categorias diferentes.".format(df_items_filtred.shape[0], orders_products_category.shape[0]))
orders_products_category.head()
```

```
Temos 111744 itens em 74 categorias diferentes.
cama_mesa_banho           11089
beleza_saude               9619
esporte_lazer              8566
moveis_decoracao           8259
informatica_acessorios     7764
Name: product_category_name, dtype: int64
```

Fonte: Autor

Vejamos abaixo um gráfico com soma dos itens de pedidos por categoria, podemos observar que a categoria com o maior valor não é a categoria com produtos mais vendidos, neste caso a categoria com o maior valor é a categoria “beleza_saude” enquanto a categoria mais vendida encontra-se em terceiro lugar no ranking, a categoria “cama_mesa_banho”. Limitei a visualização as 20 primeiras categorias.

Figura 16: Levantamento de Questões – Quantidade de Itens Vendidos por Categoria.



Fonte: Autor

4.1.3. Qual foi o melhor dia de vendas e quanto foi vendido neste dia;

O dia com o maior volume de vendas foi dia 27/11/2017, uma sexta-feira. Foi a última sexta-feira de novembro de 2017, costuma no brasil a última sexta-feira de novembro ser fomentado um dia de ofertas mais intensas, denominado de Black Friday, sendo assim, o fator de ser uma data especial para compras é de se esperar um movimento atípico. Neste dia foi vendido R$ 151.777,84.

Figura 17: Levantamento de Questões – Dia de Maior Volume de Vendas.

```
df_orders_filtred_day.sort_values(by='price',ascending=False).head(1)
```

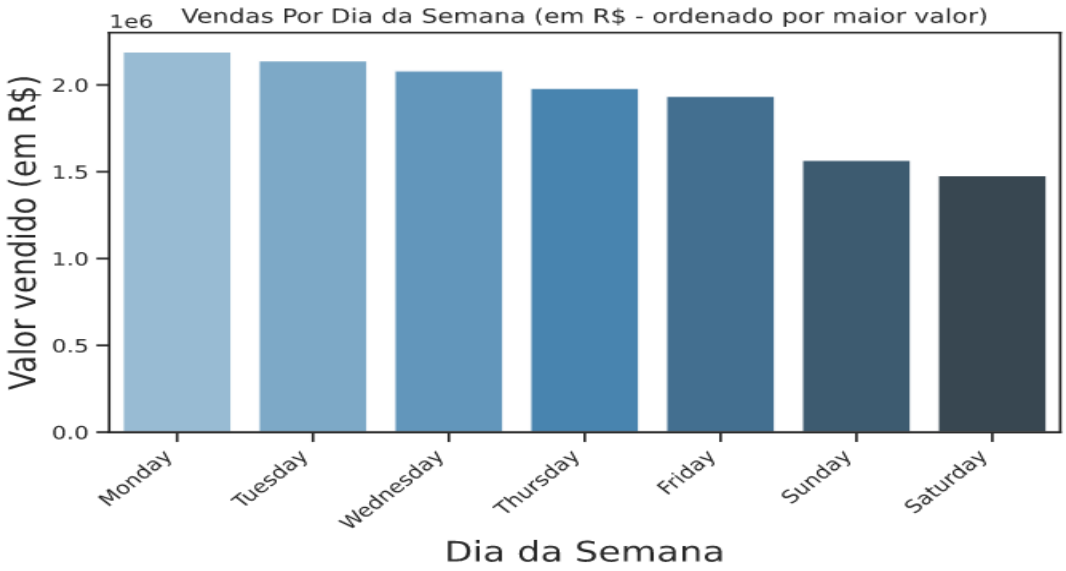
	price
order_purchase_date	
2017-11-24	151777.84

Fonte: Autor

4.1.4. Qual o dia de semana com maior venda;

O dia de semana com maior valor de vendas e com o maior valor médio de vendas é a segunda-feira.

Figura 18: Levantamento de Questões – Quantidade de Itens Vendidos por Dia da Semana.

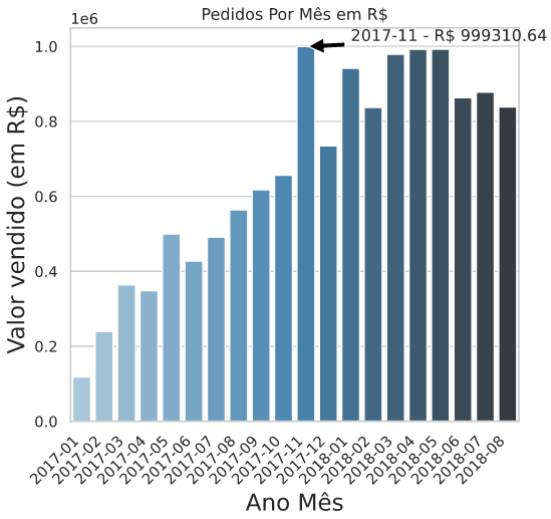


Fonte: Autor

4.1.4. Qual foi o melhor mês de vendas e quanto foi vendido neste mês;

O mês com maior volume de vendas foi o mês de novembro de 2017, provavelmente influenciado pelo Black Friday. O volume de vendas deste mês foi R$ 999.310,64.

Figura 19: Levantamento de Questões – Mês com Maior Volume de Vendas.



Fonte: Autor

5. Criação de Modelos de Machine Learning

Nesta seção, iremos aplicar os modelos de Mahine Learning na série de dados para realizar a previsão dos comportamentos futuros visando obter a melhor acurácia possível.

Primeiramente vamos aplicar um modelo mais simples, visando a facilidade e para ter um parâmetro de comparação com um menor índice de validação aceitável, este modelo é o Naive. Na sequência será aplicado o modelo Auto Regressivo e posteriormente o modelo ARIMA, como a série tem o componente sazonal será aplicado o modelo SARIMA e logo após o modelo Facebook Prophet e por fim o modelo de Suavização Exponencial (Holt-Winters).

5.1. Preparação dos Dados

5.1.1. Métricas de Avaliação de Desempenho

Para posterior análise foi realizado o armazenamento dos resultados das métricas de avaliação dos modelos e dos erros residuais, foram criados dois Datasets nomeados por df_results e df_resids, respectivamente. Como descrito anteriormente vamos aplicar as seguintes métricas MAE, MAPE, MSE e RMSE.

Figura 20: Métricas de Avaliação de Desempenho - Data frame df_results e df_resids.

```
df_results = pd.DataFrame(columns=['Model', 'MAE', 'MAPE', 'MSE','RMSE'])
df_resids = pd.DataFrame()
```

Fonte: Autor

Então, será criado uma função para cálculo da métrica MAPE com o nome mean_absolute_percentage_error e outra para aplicação de todas as métricas e armazenamento dos resultados obtidos, esta última será chamada prediction_teste_error.

Figura 21: Métricas de Avaliação de Desempenho - Funções mean_absolute_percentage_error e prediction_teste_error.

```
def mean_absolute_percentage_error (y_true, y_pred):
  y_true, y_pred = np.array(y_true), np.array(y_pred)
  return np.mean(np.abs((y_true-y_pred)/y_true))*100

def prediction_teste_error (model, y_true, y_pred):
  MAE = mean_absolute_error(y_true=y_true, y_pred=y_pred)
  MAPE = mean_absolute_percentage_error(y_true=y_true, y_pred=y_pred)
  MSE = mean_squared_error(y_true=y_true, y_pred=y_pred)
  RMSE = np.sqrt(MSE)
  global df_results,df_resids
  df_results = df_results[(df_results.Model != model)]
  df_results.loc[df_results.size+1] = [model,MAE,MAPE,MSE,RMSE]
  #df_resids = df_resids[(df_resids.Model != model)]
  df_resids[model] = y_true-y_pred
  print("Modelo {} Resultados: MAE : {}, MAPE : {}, MSE : {}, RMSE : {} ".format(model, MAE, MAPE, MSE, RMSE))
```

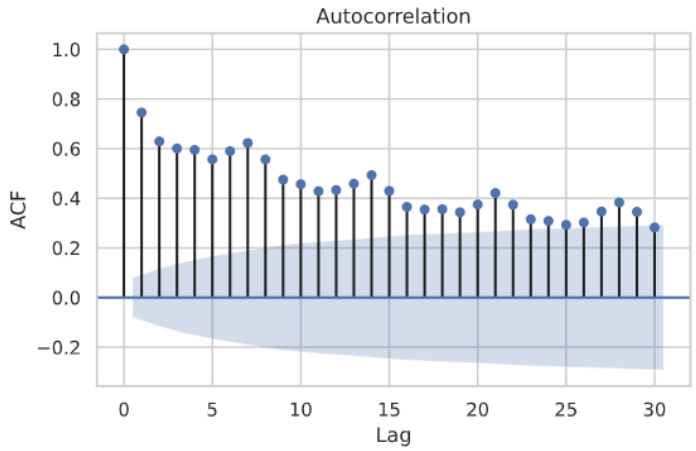
Fonte: Autor

5.1.2. Auto correlação e Auto correlação Parcial

A auto correlação é uma medida que determina o grau de correlação linear da variável observada com ela própria, para obtenção do cálculo de auto correlação é levando em consideração a interdependência linear das observações.

Para avaliação de auto correlação da variável observada vamos utilizar a funções plot_acf do módulo Statesmodels. Na Figura 22 temos o gráfico obtido.

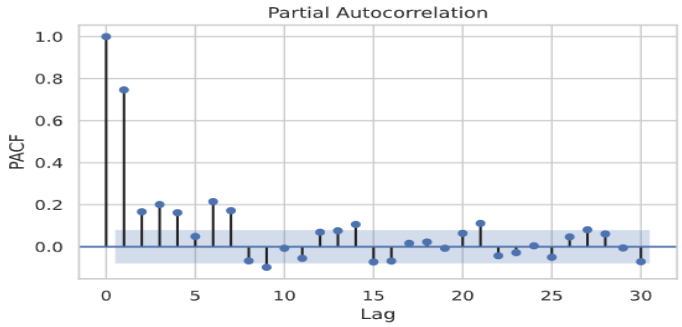
Figura 22: Auto correlação e Auto correlação Parcial - Auto Correlação.



Fonte: Autor

Para a auto correlação parcial é verificado a correlação entre duas variáveis sem ser considerado a interdependência. Será utilizado a auto correlação parcial para definição do valor para a variável *p.*

Figura 23: Auto correlação e Auto correlação Parcial – Auto Correlação Parcial.



Fonte: Autor

5.1.3. Base de Treino e Teste

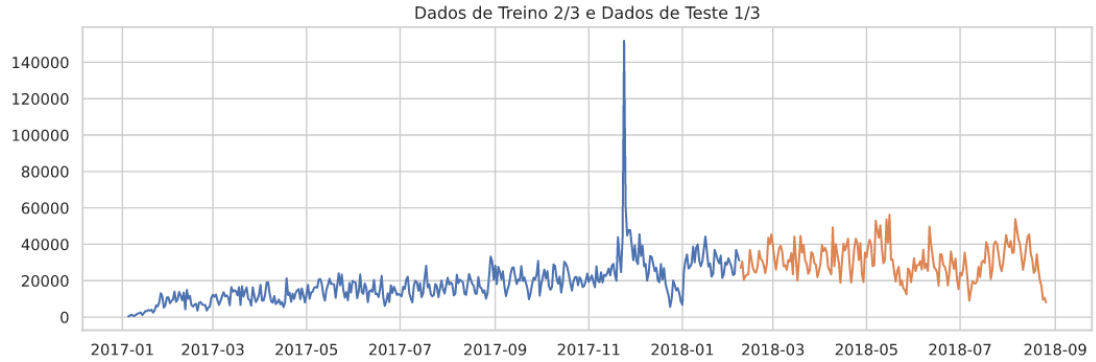
Vamos fazer a separação da base entre treino e teste, então será utilizado $\frac{2}{3}$ (dois terços) das observações para treino dos modelos e $\frac{1}{3}$ (um terço) para teste das previsões realizadas. Será apresentado abaixo o código utilizado para divisão da base em dois Datasets o ds_day_train e o ds_day_teste, logo após uma representação gráfica das duas bases.

Figura 24: Base de Treino e Teste - Código Divisão das Bases.

```
train_size = int(len(df_orders_filtred_day_pred) * 2/3)
ds_day_train = df_orders_filtred_day_pred.price[:train_size]
ds_day_test = df_orders_filtred_day_pred.price[train_size:]

plt.figure(figsize=(13,4))
plt.plot(ds_day_train)
plt.plot(ds_day_test)
plt.title("Dados de Treino 2/3 e Dados de Teste 1/3")
plt.show()
```

Fonte: Autor

Figura 23: Base de Treino e Teste – Representação Gráfica das Bases de Treino e Teste.



Fonte: Autor

5.1.4. Estacionariedade

Se os valores da série temporal não forem estacionários e sazonais, serão necessárias transformações de diferenciação mais complexas. Antes de usar a diferenciação para transformar valores sazonais não estacionários em valores estacionários, precisamos verificar se os dados ao longo do tempo mostram uma variação sazonal constante. Para estabilizar a variância, aplique uma transformação de pré-diferenciação apropriada Box and Jenkins (1970).

Uma série estacionária é a série cujo, suas observações mantêm sua média, variância e covariância constantes no decorrer do tempo, nela não existe tendência de alta ou tendência de baixa. Observando o padrão gráfico conseguimos identificar se uma série é estacionária ou não, porém, temos como realizar alguns testes estatísticos para identifica-lo, em nosso trabalho vamos aplicar um teste chamado de Dickey-Fuller (ADF). O ADF considera a hipótese nula como não estacionária e a hipótese alternativa como estacionária. A interpretação deste modelo se dá da seguinte forma, em sua execução obteremos o valor de teste, p-valor e os níveis de significância com seus valores críticos, para que possamos descartar a hipótese nula, devemos observar se o p-valor é menor que o nível de significância escolhido e o valor do teste tem que ser menor que o valor crítico do nível escolhido.

Definimos a uma função chamada de adf_teste para obtenção dos valores do teste ADF. Abaixo temos o script de criação da função e o resultado que obtivemos,

quando passamos a nossa base de treinamento para saber se ela é estacionária ou não.

Figura 23: Estacionariedade - Função adf_teste.

```
def adf_test(y):
    print('Resultado do Teste Dickey-Fuller:')
    dftest = adfuller(y, autolag="AIC")
    dfoutput = pd.Series(dftest[0:4], index=['Teste','Valor p', 'Nº de lags', 'Nº de observações'])
    for key, value in dftest[4].items():
        dfoutput['Valor Crítico ({})'.format(key)] = value
    print(dfoutput)
```

Fonte: Autor

Figura 24: Estacionariedade - Resultado Teste ADF.

```
adf_test(ds_day_train)
```

```
Resultado do Teste Dickey-Fuller:
Teste                  -2.852638
Valor p                 0.051139
Nº de lags              6.000000
Nº de observações     392.000000
Valor Crítico (1%)     -3.447142
Valor Crítico (5%)     -2.868941
Valor Crítico (10%)    -2.570713
dtype: float64
```

Fonte: Autor

Como era de se esperar, já que a série observada possui o componente de tendência, podemos observar que o teste ADF resultou em uma série não estacionária pois o p-valor é maior que 5%.

Para tornar uma a série estacionária pode-se aplicar a diferenciação de primeira ordem, então será aplicado e realizado o teste novamente.

Figura 25: Estacionariedade - Resultado Teste ADF Após a Diferenciação.

```
ds_day_train_diff = ds_day_train - ds_day_train.shift()
ds_day_train_diff.dropna(inplace=True)
adf_test(ds_day_train_diff)
```

```
Resultado do Teste Dickey-Fuller:
Teste                  -1.349614e+01
Valor p                 3.039300e-25
Nº de lags              5.000000e+00
Nº de observações       3.920000e+02
Valor Crítico (1%)     -3.447142e+00
Valor Crítico (5%)     -2.868941e+00
Valor Crítico (10%)    -2.570713e+00
dtype: float64
```

Fonte: Autor

Podemos observar que ao aplicar a diferenciação foi possível chegar em um p-valor bem menor que 5%. Assim temos uma série estacionária.

5.2. Modelo Naive

Como citado anteriormente, este modelo é o mais simples de previsão, o objetivo dele é a aplicação da diferenciação das observações, no qual, o valor previsto para o período atual é o valor do período imediatamente anterior, ou seja, o valor de *t* é o mesmo que *t-1*.

Na figura abaixo, Figura 26, foi adicionado na série uma variável chamada "naive" com o resultado da diferenciação de primeira ordem da variável "price".

Figura 26: Modelo Naive - Diferenciação de Primeira Ordem.

```
df_prev_naive =  pd.DataFrame(ds_day_test)
df_prev_naive['naive'] = df_prev_naive.shift(1)
df_prev_naive.head()
```

order_purchase_date	price	naive
2018-02-08	27074.86	NaN
2018-02-09	30571.07	27074.86
2018-02-10	20579.17	30571.07
2018-02-11	22350.18	20579.17
2018-02-12	23255.73	22350.18

Fonte: Autor

5.3. Modelo Auto Regressivo

O modelo auto regressivo é uma regressão linear baseada nas observações significativamente correlacionados, sendo assim, primeiramente devemos olhar para auto correlação e auto correlação parcial da variável analisada, a fim de obter a quantidade de observações (ou lags) com as correlações mais significativas. Como observamos anteriormente, a série analisada possui uma forte correlação no lag 2, vamos passar este valor para a variável *p* para treinar o modelo.

Figura 27: Modelo Auto Regressivo – Treino e Resumo.

```
df_prev_ar_sts =  pd.DataFrame(ds_day_train.copy())
#Treino
ar2 = AutoReg(df_prev_ar_sts.price, lags=2, old_names=False).fit()
print(ar2.summary())
```

```
                            AutoReg Model Results
==============================================================================
Dep. Variable:                  price   No. Observations:                  399
Model:                     AutoReg(2)   Log Likelihood               -4139.180
Method:               Conditional MLE   S.D. of innovations           8161.691
Date:                Fri, 05 Feb 2021   AIC                             18.035
Time:                        20:50:02   BIC                             18.075
Sample:                    01-07-2017   HQIC                            18.050
                         - 02-07-2018
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const       4754.8844    812.135      5.855      0.000    3163.129    6346.640
price.L1       0.5697      0.049     11.528      0.000       0.473       0.667
price.L2       0.1722      0.049      3.486      0.000       0.075       0.269
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1            1.2688           +0.0000j            1.2688            0.0000
AR.2           -4.5781           +0.0000j            4.5781            0.5000
-----------------------------------------------------------------------------
```

Fonte: Autor

5.4. Modelo ARMA

Este modelo é uma combinação do modelo anterior AR com o componente de média móvel. A adição do parâmetro de médias móveis visa modelar os eventos de erros da série temporal. Seus parâmetros são (*p*, *q*) onde *p* é a quantidade observações para o componente de auto regressão e *q* define a quantidade observações para o componente de média móvel. Sendo assim este modelo é denotado por ARMA(p,q).

O ARMA geralmente é utilizado na configuração ARMA(1,1), foi realizado alguns testes com configurações diferentes. A configuração que melhor se comportou foi a ARMA(2,2).

Figura 28: Modelo ARMA – Treino e Resumo.

```
#Treino
df_prev_arma_train_3 = ARMA(ds_day_train,freq='D',order=(2,2)).fit()
#Previsão
df_prev_arma_test_3 = ARMA(ds_day_test,freq='D',order=(2,2)).fit(df_prev_arma_train_3.params)
print(df_prev_arma_test_3.summary())
```

```
                              ARMA Model Results
==============================================================================
Dep. Variable:                  price   No. Observations:                  200
Model:                     ARMA(2, 2)   Log Likelihood               -2063.935
Method:                       css-mle   S.D. of innovations           7328.579
Date:                Sat, 06 Feb 2021   AIC                           4139.870
Time:                        13:07:35   BIC                           4159.660
Sample:                    02-08-2018   HQIC                          4147.879
                         - 08-26-2018
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const        3.039e+04   1709.211     17.779      0.000     2.7e+04    3.37e+04
ar.L1.price     1.1900      0.225      5.287      0.000       0.749       1.631
ar.L2.price    -0.2760      0.186     -1.486      0.137      -0.640       0.088
ma.L1.price    -0.6245      0.223     -2.800      0.005      -1.062      -0.187
ma.L2.price    -0.0899      0.114     -0.787      0.431      -0.314       0.134
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1            1.1438           +0.0000j            1.1438            0.0000
AR.2            3.1679           +0.0000j            3.1679            0.0000
MA.1            1.3421           +0.0000j            1.3421            0.0000
MA.2           -8.2898           +0.0000j            8.2898            0.5000
-----------------------------------------------------------------------------
```

Fonte: Autor

5.5. Modelo ARIMA

É o modelo auto regressivo de médias móveis integrados, ele é composto pelo componente de auto regressão (AR), componentes de integração (I) que é a diferenciação da série a fim de torná-la estacionária e o componente de médias móveis (MA). Neste modelo temos por parâmetros (*p*, *d, q*) que são eles; *p* é a ordem do lag onde informamos o número e observações passadas para serem consideradas pelo componente de auto regressão, *d* é o grau de diferenciação pelo qual informamos o número de diferenciações a serem realizadas no componente de integração e por fim o parâmetro *q*, que é a ordem de média móvel onde informamos o tamanho da janela de média móvel a ser considerada.

Figura 29: Modelo ARIMA – Treino e Resumo.

```
#Treino
df_prev_arima_train_2 = ARIMA(ds_day_train,freq='D',order=(3,1,2)).fit()
#Previsão
df_prev_arima_test_2 = ARIMA(ds_day_test,freq='D',order=(3,1,2)).fit(df_prev_arima_train_2.params)
print(df_prev_arima_test_2.summary())
```

```
                             ARIMA Model Results
==============================================================================
Dep. Variable:                D.price   No. Observations:                  199
Model:                 ARIMA(3, 1, 2)   Log Likelihood               -2040.698
Method:                       css-mle   S.D. of innovations           6841.819
Date:                Sat, 06 Feb 2021   AIC                           4095.395
Time:                        14:04:14   BIC                           4118.449
Sample:                    02-09-2018   HQIC                          4104.726
                         - 08-26-2018
==============================================================================
                    coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          -74.9712    275.351     -0.272      0.785    -614.648     464.706
ar.L1.D.price    0.8227      0.073     11.286      0.000       0.680       0.966
ar.L2.D.price   -0.4603      0.084     -5.490      0.000      -0.625      -0.296
ar.L3.D.price   -0.3895      0.082     -4.775      0.000      -0.549      -0.230
ma.L1.D.price   -1.2657      0.061    -20.713      0.000      -1.385      -1.146
ma.L2.D.price    0.8473      0.079     10.688      0.000       0.692       1.003
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1            0.6405           -0.7952j            1.0210           -0.1421
AR.2            0.6405           +0.7952j            1.0210            0.1421
AR.3           -2.4629           -0.0000j            2.4629           -0.5000
MA.1            0.7469           -0.7889j            1.0863           -0.1294
MA.2            0.7469           +0.7889j            1.0863            0.1294
-----------------------------------------------------------------------------
```

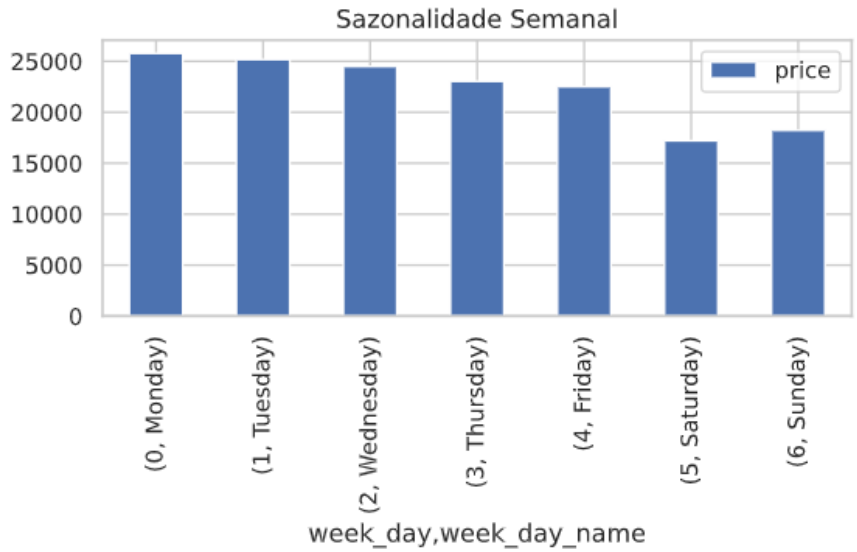
Fonte: Autor

5.6. Modelo SARIMA

Este modelo considera o componente sazonal juntamente com os componentes auto regressivos, de integração e de médias móveis. Uma série temporal é considerada sazonal quando os fenômenos se repetem em um intervalo idêntico de tempo, a série quando sazonal possui comportamentos facilmente previsíveis ocorrendo em intervalos regulares de tempo. Dessa forma, vamos passar como parâmetros, os elementos de tendência de sazonalidade para os componentes de auto regressão, integração e médias móveis, também vamos passar um parâmetro adicional para o período de sazonalidade. Os elementos de tendências são; *p* ordem de AR, *d* ordem de diferenças e *q* ordem de MA. Os elementos sazonais são; *P* ordem de AR sazonal, *D* ordem das diferenças sazonal, *Q* ordem de MA sazonal e *s* número de períodos referentes ao ciclo sazonal. Sendo denotado da seguinte forma SARIMA (p,d,q)(P,D,Q)s.

Para simples conferência da sazonalidade vamos avaliamos a média de valores de pedidos por dia de semana, podemos observar na Figura 30 que temos um componente sazonal semanal onde temos um maior valor médio para o início da semana o que vai caindo nos dias subsequentes, sendo assim fica definido o período sazonal *s*=7.

Figura 30: Modelo SARIMA – Sazonalidade.



Fonte: Autor

Figura 31: Modelo SARIMA – Treino e Resumo.

```
#Treino
df_prev_sarima_train = SARIMAX(ds_day_train,freq='D',order=(3,1,2), seasonal_order=(3,1,2,7)).fit()
#Previsao
df_prev_sarima_test = SARIMAX(ds_day_test,freq='D',order=(3,1,2), seasonal_order=(3,1,2,7)).fit(df_prev_sarima_train.params)
print(df_prev_sarima_test.summary())
```

```
                                SARIMAX Results
==========================================================================================
Dep. Variable:                             price   No. Observations:                  200
Model:             SARIMAX(3, 1, 2)x(3, 1, 2, 7)   Log Likelihood               -1991.076
Date:                           Sat, 06 Feb 2021   AIC                           4004.153
Time:                                   14:07:43   BIC                           4039.985
Sample:                                 02-08-2018   HQIC                          4018.665
                                      - 08-26-2018
Covariance Type:                            opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.7253      0.849     -0.854      0.393      -2.390       0.939
ar.L2          0.1362      1.077      0.126      0.899      -1.975       2.247
ar.L3         -0.0165      0.440     -0.037      0.970      -0.879       0.846
ma.L1          0.2791      0.805      0.346      0.729      -1.300       1.858
ma.L2         -0.4600      0.822     -0.560      0.576      -2.071       1.151
ar.S.L7       -0.8122      1.324     -0.614      0.539      -3.406       1.782
ar.S.L14       0.0199      0.398      0.050      0.960      -0.761       0.800
ar.S.L21       0.0444      0.278      0.160      0.873      -0.501       0.590
ma.S.L7        0.0124      1.360      0.009      0.993      -2.653       2.678
ma.S.L14      -0.6938      0.975     -0.712      0.477      -2.605       1.217
sigma2      1.098e+08   1.25e-08   8.76e+15      0.000     1.1e+08     1.1e+08
==========================================================================================
Ljung-Box (L1) (Q):                   0.35   Jarque-Bera (JB):                 0.86
Prob(Q):                              0.55   Prob(JB):                         0.65
Heteroskedasticity (H):               0.81   Skew:                             0.16
Prob(H) (two-sided):                  0.39   Kurtosis:                         3.03
==========================================================================================
```

Fonte: Autor

5.7. Aplicação da Função Auto ARIMA

A função Auto Arima vai nos ajudar a encontrar o melhor modelo com a composição de parâmetros que vai nos sugerir um melhor ajuste para nossa previsão, dessa forma vamos aplicá-lo para avaliar a qualidade de sua previsão.

Esta função tem alguns parâmetros, dentre eles vamos precisar configura os parâmetros a seguir;

- start_p = 1 e max_p = 6 - range dos parâmetros para teste de AR,
- start_q = 1 e max_q = 6 - range dos parâmetros para teste de MA,
- d= 1 - parâmetro de integração usado para séries não estacionárias,
- Seazonal = True - define que a série tem o componente sazonal,
- start_P = 0 - define o grau de diferenciação para o AR sazonal,
- D = 1 - parâmetro de integração sazonal,
- m = 7 - número de observações do ciclo sazonal,
- trace = True - exibir as configurações testadas,
- error_action = 'ignore' - determina para ignorar os erros,
- suppress_warnings = True - suprimir as mensagens
- stepwise = True - modo de execução True indica modo Stepwise e False o modo Parallelized, que serão definidos abaixo.

No modo de execução Parallelized é utilizado uma abordagem mais ingênua pois é utilizado vários hiperparâmetros de forma aleatória, para que, assim na força bruta encontre uma combinação de hiperparâmetros, este modo tem um tempo maior de execução.

A execução Stepwise, ou na tradução literal passo a passo, segue a estratégia proposta por Hyndman and Khandakar (2018). Dessa forma, ele busca as melhores combinações com um menor grau de AIC, BIC, ACc ou outro tipo de avaliação definida será escolhido, não precisando passar por todas as combinações possíveis, por este motivo ele possui uma execução mais rápida. Um exemplo prático como definimos que o modelo é sazonal ele não realizará os testes de parâmetros não sazonais.

Figura 32: Função Auto ARIMA – Treino e Resumo Modo Stepwise.

```
print(auto_arima_stpw_t.summary())
```

```
                                SARIMAX Results
==========================================================================================
Dep. Variable:                                  y   No. Observations:                  599
Model:             SARIMAX(2, 1, 0)x(0, 1, [1], 7)   Log Likelihood               -6153.861
Date:                             Sat, 06 Feb 2021   AIC                          12315.721
Time:                                     14:11:33   BIC                          12333.249
Sample:                                          0   HQIC                         12322.549
                                               - 599
Covariance Type:                               opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.4392      0.021    -20.736      0.000      -0.481      -0.398
ar.L2         -0.2661      0.025    -10.770      0.000      -0.315      -0.218
ma.S.L7       -0.9290      0.038    -24.167      0.000      -1.004      -0.854
sigma2      1.02e+08   1.11e-10   9.16e+17      0.000    1.02e+08    1.02e+08
===================================================================================
Ljung-Box (L1) (Q):                   2.03   Jarque-Bera (JB):            263685.30
Prob(Q):                              0.15   Prob(JB):                         0.00
Heteroskedasticity (H):               2.52   Skew:                             6.22
Prob(H) (two-sided):                  0.00   Kurtosis:                       105.73
===================================================================================
```

Fonte: Autor

Figura 33: Função Auto ARIMA – Treino e Resumo Modo Parallelized.

```
print(auto_arima_stpw_f.summary())
```

```
                                SARIMAX Results
==========================================================================================
Dep. Variable:                                  y   No. Observations:                  599
Model:             SARIMAX(4, 1, 0)x(0, 1, [1], 7)   Log Likelihood               -6136.902
Date:                             Sat, 06 Feb 2021   AIC                          12285.803
Time:                                     14:15:21   BIC                          12312.094
Sample:                                          0   HQIC                         12296.045
                                               - 599
Covariance Type:                               opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.5063      0.026    -19.283      0.000      -0.558      -0.455
ar.L2         -0.3761      0.045     -8.366      0.000      -0.464      -0.288
ar.L3         -0.2373      0.043     -5.457      0.000      -0.323      -0.152
ar.L4         -0.0402      0.039     -1.029      0.303      -0.117       0.036
ma.S.L7       -0.9297      0.035    -26.190      0.000      -0.999      -0.860
sigma2      9.556e+07    3.9e-10   2.45e+17      0.000    9.56e+07    9.56e+07
===================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):            290952.41
Prob(Q):                              0.91   Prob(JB):                         0.00
Heteroskedasticity (H):               2.72   Skew:                             6.74
Prob(H) (two-sided):                  0.00   Kurtosis:                       110.86
===================================================================================
```

Fonte: Autor

5.8. Modelo Facebook Prophet

O Facebook Prophet foi criado pelo Facebook para facilitar a implementação de modelos de séries temporais, tornando-os mais automatizados, sem a necessidade de muito conhecimento para sua implementação.

Para aplicação do modelo temos que seguir uma regra básica que é criar um Data Frame com as seguintes colunas ds para armazenamento da variável temporal e a coluna y para armazenamento da variável observada.

Figura 34: Modelo Facebook Prophet – Preparação Data Frame, Treino e Previsão.

```
#Ajsute DF para utilização do Prophet
df_prev_prophet_train = pd.DataFrame(data=ds_day_train,index=ds_day_train.index)
df_prev_prophet_train.dropna(inplace=True)
df_prev_prophet_train.reset_index(inplace=True)
df_prev_prophet_train.columns=['ds','y']

#Treino
prophet_model = Prophet(changepoint_prior_scale=0.2,interval_width=0.5)
prophet_model.fit(df_prev_prophet_train)

prophet_forecast = prophet_model.make_future_dataframe(periods=len(ds_day_test), freq='D',include_history=False)
prophet_forecast = prophet_model.predict(prophet_forecast)
```

Fonte: Autor

5.9. Modelo Holt-Winters

Este é um modelo indicado para séries que apresentam comportamentos de tendência lienar e movimentos sazonais, ele possui duas variações aditivo e multiplicativo, no modelo multiplicativo considera que a amplitude da sazonalidade é variante no decorrer do tempo e possui uma variação crescente, enquanto que no modelo aditivo as variações das variações sazonais são aproximadamente constantes. Em nosso trabalho vamos aplicar a variação aditiva tendência e multiplicativa para sazonalidade.

A configuração do modelo está na Figura 35, logo abaixo.

Figura 35: Modelo Holt-Winters – Treino e Resumo.

```
#Treino
fitted = HWES(df_prev_holtwinters_train,seasonal_periods=7,trend='add',seasonal='mul').fit()
#Resultado
print(fitted.summary())
```

```
                       ExponentialSmoothing Model Results
==============================================================================
Dep. Variable:                    price   No. Observations:                  399
Model:             ExponentialSmoothing   SSE                 25075726407.728
Optimized:                         True   AIC                        7186.526
Trend:                         Additive   BIC                        7230.404
Seasonal:                Multiplicative   AICC                       7187.471
Seasonal Periods:                     7   Date:                Sat, 06 Feb 2021
Box-Cox:                          False   Time:                        14:02:12
Box-Cox Coeff.:                    None
==============================================================================
                         coeff                 code              optimized
------------------------------------------------------------------------------
smoothing_level         0.3585714                alpha                 True
smoothing_trend            0.0001                 beta                 True
smoothing_seasonal      0.0836646                gamma                 True
initial_level           18045.705                  l.0                 True
initial_trend           229.80714                  b.0                 True
initial_seasons.0       0.0219942                  s.0                 True
initial_seasons.1       0.0507811                  s.1                 True
initial_seasons.2       0.0749153                  s.2                 True
initial_seasons.3       0.0393213                  s.3                 True
initial_seasons.4       0.0373380                  s.4                 True
initial_seasons.5       0.0795131                  s.5                 True
initial_seasons.6       0.1028095                  s.6                 True
------------------------------------------------------------------------------
```

Fonte: Autor

6. Apresentação dos Resultados

Nesta seção será apresentado os resultados obtidos. Primeiramente vamos passar por cada modelo fazendo uma breve análise das métricas obtidas. Analisaremos a distribuição dos erros obtidos.

Ao final, será realizado uma comparação entre todos os modelos, na sequência o levantamento de qual será o modelo que foi eleito que melhor irá atender à necessidade em questão.

6.1. Modelo Naive

Para o modelo Naive, considerando as métricas escolhidas, obtivemos o resultado apresentado na Figura 35. Logo após na Figura 36 podemos observar a representação gráfica do modelo, considerando o mesmo período que utilizaremos

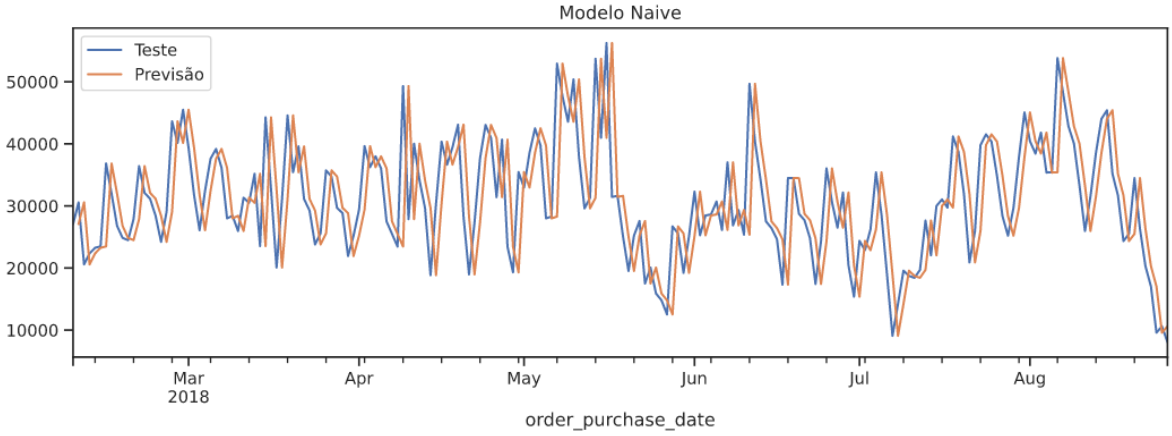
como teste nos demais modelos, na sequência na Figura 37, temos a representação gráfica de todas as observações da série analisada juntamente com a previsão.

Figura 35: Modelo Naive – Avaliação.

```
Modelo Naive Resultados: MAE : 6466.879648241223, MAPE : 21.95394997737745, MSE : 67976123.12699176, RMSE : 8244.763376046141
```

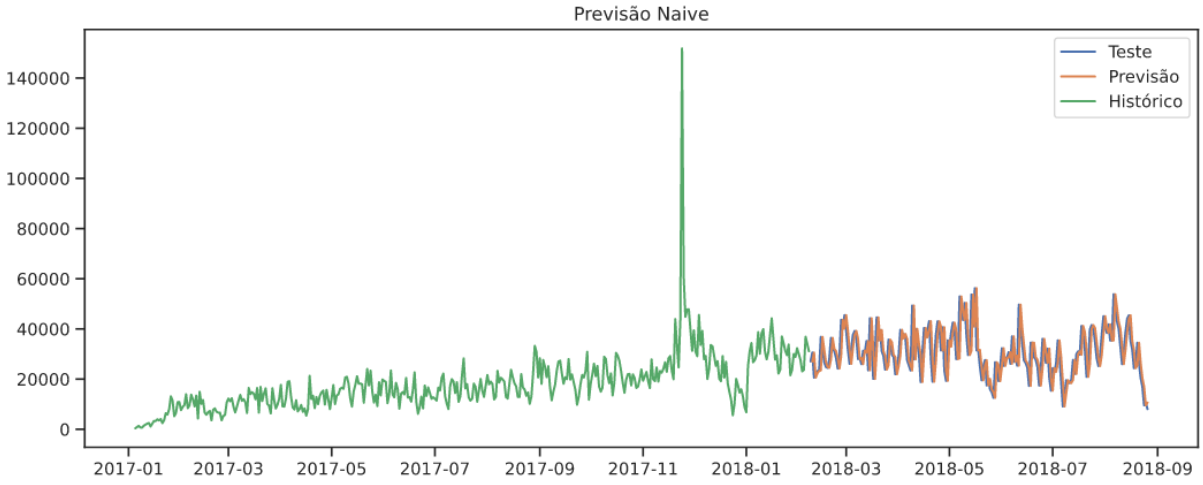
Fonte: Autor

Figura 36: Modelo Naive – Previsão e Teste.



Fonte: Autor

Figura 37: Modelo Naive – Treino, Previsão e Teste.



Fonte: Autor

6.2. Modelo Auto Regressivo

Podemos observar abaixo, na Figura 38, a aplicação da previsão para o modelo auto regressivo. A partir do modelo treinado, obtemos os coeficientes de regressão

para serem aplicados as observações e realizar a previsão, como vamos realizar a previsão sob os dados de teste, vamos montar nossa equação com ele, sendo assim, vamos fazer uma iteração sob os parâmetros retornados do treinamento a fim de gerar a equação executada nos dados de teste.

Figura 38: Modelo Auto Regressivo – Equação de auto regressão.

$$X_t = \phi_0 + \phi_1 X_{t-1} + \cdots + \phi_p X_{t-p} + \varepsilon_t$$

Fonte: Portalaction.com.br

Figura 39: Modelo Auto Regressivo – Previsão e Avaliação do Modelo.

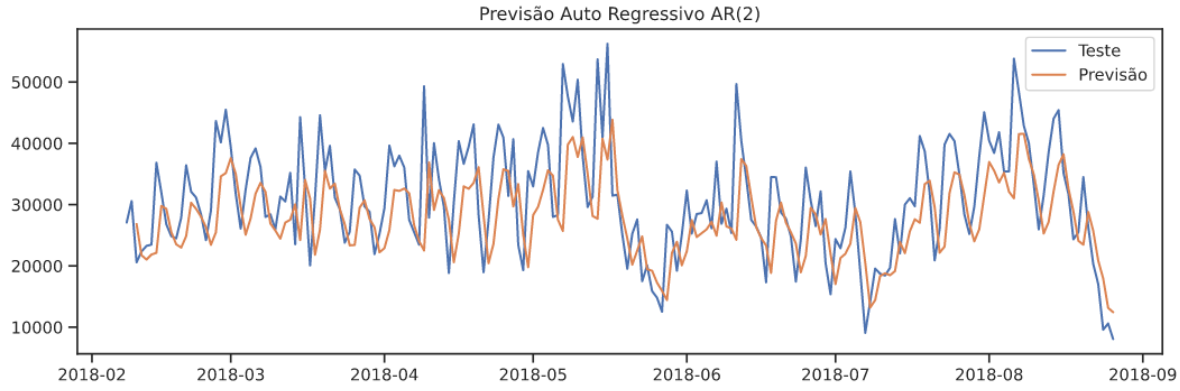
```
#Previsão
y_pred_ar2 = ar2.params[0]
for i, coef in enumerate(ar2.params[1:]):
  y_pred_ar2 += coef * ds_day_test.shift(i + 1)
#avaliação do modelo
y_true = ds_day_test[2:].values
y_pred_ar2.dropna(inplace=True)
prediction_teste_error('Teste - Auto Regressivo AR(2)',y_true=y_true,y_pred=y_pred_ar2)
```

```
Modelo Teste - Auto Regressivo AR(2) Resultados: MAE : 6100.111162953462, MAPE : 19.835342242404543, MSE : 66421841.23579456, RMSE : 8149.959584917864
```

Fonte: Autor

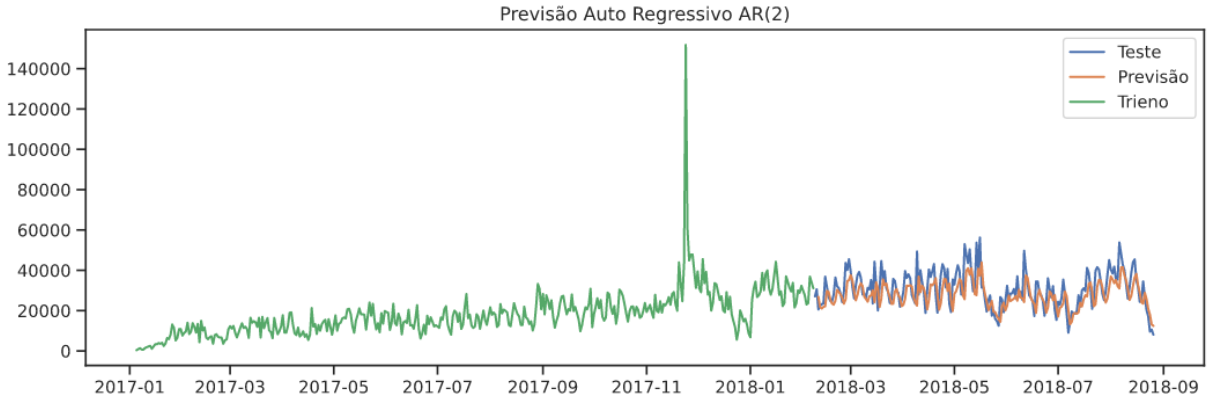
Como esperado tivemos os resultados melhores que o modelo Naive. Abaixo temos as representações gráficas dos dados de Treinamento, Previsão e teste.

Figura 40: Modelo Auto Regressivo – Previsão e Teste.



Fonte: Autor

Figura 41: Modelo Auto Regressivo – Treino, Previsão e Teste.



Fonte: Autor

6.3. Modelo ARMA

Neste modelo, diferente do modelo anterior, não será necessário fazer a iteração com os parâmetros, a própria função da biblioteca irá aplicar os parâmetros obtidos no treinamento. Foram realizados testes com as configurações ARMA(1,1), ARMA(1,2), ARMA(2,2) e ARMA(2,3). Na configuração do modelo ARMA, obtivemos o melhor nível de acerto com a configuração seguinte, para o grau de autor egressão utilizamos p = 2 e para a média móvel q = 2, sendo assim, ARMA(2,2).

Figura 43: Modelo ARMA – Código da Previsão e Avaliação do Modelo.

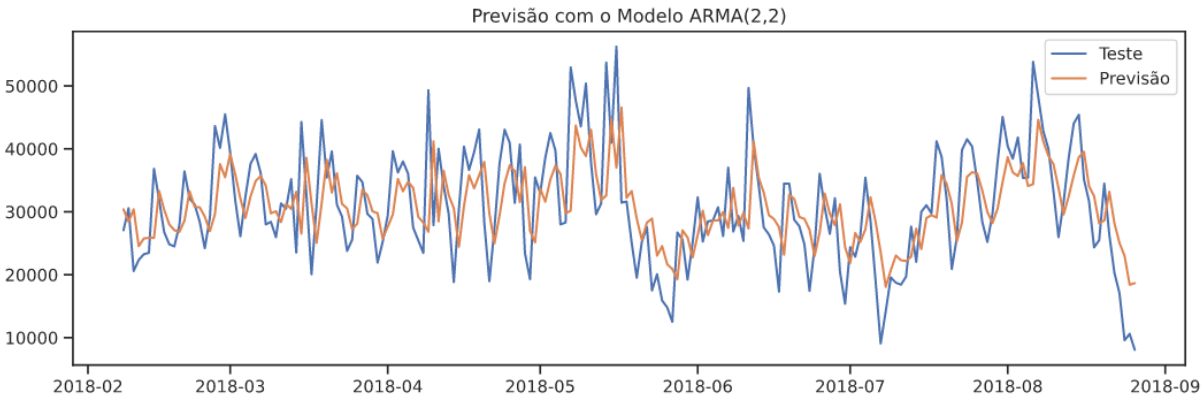
```
#Teste do modelo
#Dados para teste
y_true_arma = ds_day_test.values
#Avaliação do modelo
prediction_teste_error('Previsão ARMA(2,2)',y_true=y_true_arma,y_pred=df_prev_arma_test_3.predict())
```

```
Modelo Previsão ARMA(2,2) Resultados: MAE : 5731.920328415472, MAPE : 21.177033500906525, MSE : 53733931.33064007, RMSE : 7330.343193237276
```

Fonte: Autor

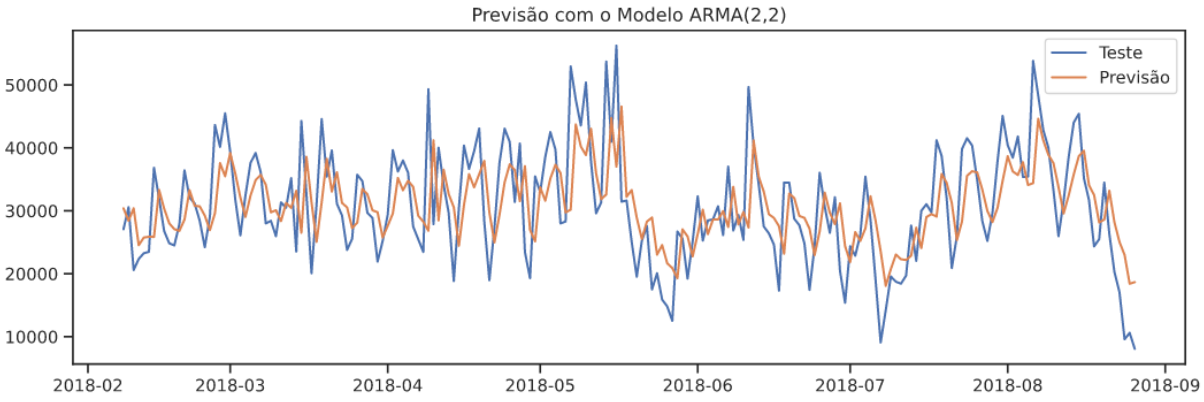
O modelo ARMA, resultou em um modelo com o percentual do erro maior que o apresentado pelo modelo AR, porém quando olhamos o MSE e o RMSE, podemos observar que, temos uma distribuição dos erros em níveis menores, sendo assim o modelo ARMA tem seus erros mais próximos de zero, apesar de estar com a média mais alta. Abaixo segue as visualizações gráficas dos dados.

Figura 44: Modelo ARMA – Previsão e Teste.



Fonte: Autor

Figura 45: Modelo ARMA – Treino, Previsão e Teste.



Fonte: Autor

6.4. Modelo ARIMA

Neste modelo, vamos adicionar o componente de integração, como vimos que no modelo ARMA tivemos o melhor resultado com a configura de p = 2 e q = 2, vamos considerá-los e adicionar o componente de integração aplicando o valor d = 1, ficando da seguinte forma ARIMA(2,1,2).

Figura 46: Modelo ARIMA – Código da Previsão e Avaliação do Modelo.

```
#Treino
df_prev_arima_train_1 = ARIMA(ds_day_train,freq='D',order=(2,1,2)).fit()
#Previsão
df_prev_arima_test_1 = ARIMA(ds_day_test,freq='D',order=(2,1,2)).fit(df_prev_arima_train_1.params)
#Avaliação
y_true = ds_day_test[1:].values
y_pred_arima_1 = df_prev_arima_test_1.predict(typ='levels')
prediction_teste_error('Previsão ARIMA(2,1,2)',y_true=y_true,y_pred=y_pred_arima_1)
```

```
Modelo Previsão ARIMA(2,1,2) Resultados: MAE : 5529.024957174562, MAPE : 19.58374513687546, MSE : 51870707.04280326, RMSE : 7202.132117838665
```

Fonte: Autor

Observamos que tivemos um melhor nível de acerto ao considerar a componente de integração, porém, após realizar alguns testes, foi observado que se considerarmos a configuração ARIMA(3,1,2), teremos de ter uma maior assertividade. Veja abaixo a assertividade desta configuração.

Figura 47: Modelo ARIMA – Avaliação do Modelo.

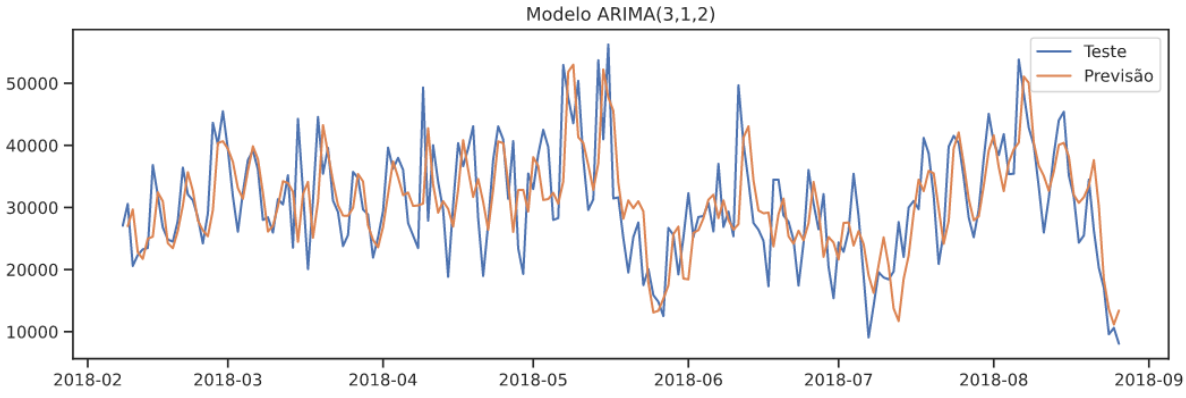
```
#Avaliação
y_true = ds_day_test[1:].values
y_pred_arima_2 = df_prev_arima_test_2.predict(typ='levels')
prediction_teste_error('Previsão ARIMA(3,1,2)',y_true=y_true,y_pred=y_pred_arima_2)
```

```
Modelo Previsão ARIMA(3,1,2) Resultados: MAE : 5227.105152547895, MAPE : 18.4069066953502, MSE : 47047273.92239205, RMSE : 6859.101539005823
```

Fonte: Autor

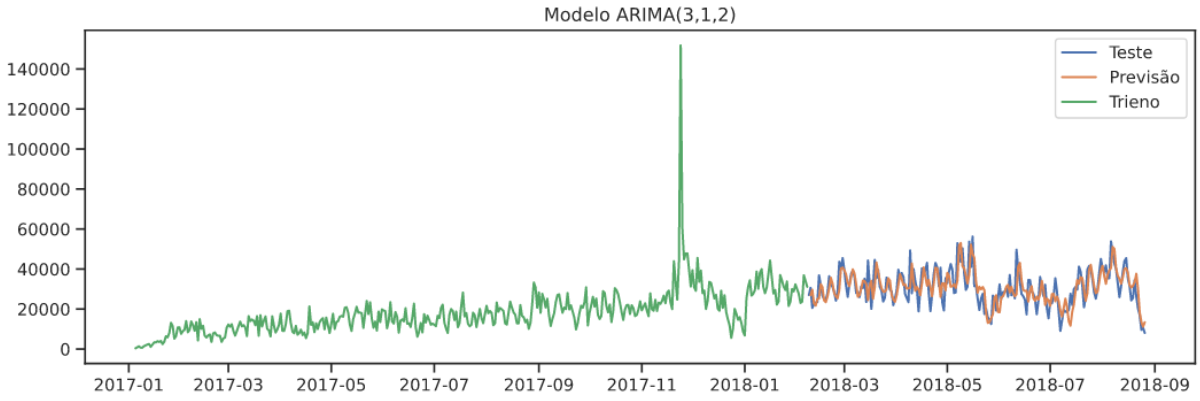
Agora vamos as representações gráficas, para comparação entre os dados previstos, dados de treinamento e teste.

Figura 48: Modelo ARIMA – Previsão e Teste.



Fonte: Autor

Figura 49: Modelo ARIMA – Treino, Previsão e Teste.



Fonte: Autor

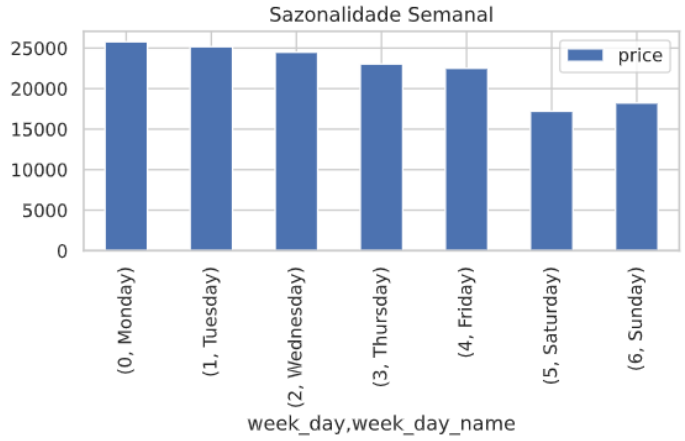
6.5. Modelo SARIMA

Agora vamos adicionar o componente de sazonalidade. Como esta série tem um componente sazonal forte, esperamos ter um resultado mais apurado na aplicação deste modelo. Abaixo em Figura 50 e em Figura 51, podemos observar a sazonalidade semanal, sendo assim vamos considerar como período sazonal s=7.

Figura 50: Modelo SARIMA - Componente Sazonal.

Fonte: Autor

Figura 51: Modelo SARIMA - Sazonalidade Semanal.



Fonte: Autor

Sendo assim, a configuração do modelo ficará da seguinte forma SARIMAX(3,1,2)(3,1,2,7).

Figura 52: Modelo SARIMA – Avaliação do Modelo.

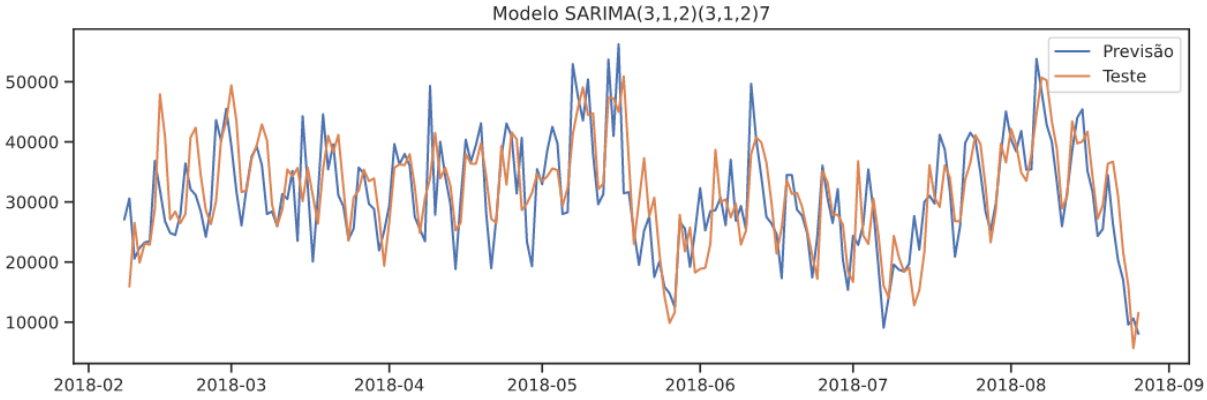
```
#Avaliação
y_true = ds_day_test[1:].values
df_prev_sarima_pred = df_prev_sarima_test.predict(typ='levels')[1:]
prediction_teste_error('Previsão SARIMA(3,1,2)(3,1,2)7',y_true=y_true,y_pred=df_prev_sarima_pred)
```

```
Modelo Previsão SARIMA(3,1,2)(3,1,2)7 Resultados: MAE : 5116.197412310387, MAPE : 18.218369739886903, MSE : 42767356.43852531, RMSE : 6539.675560647127
```

Fonte: Autor

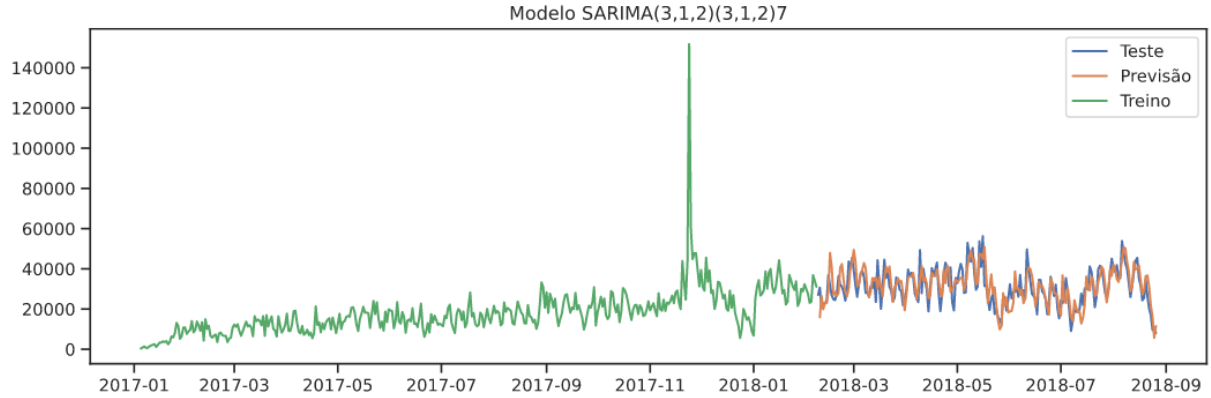
Como esperado o modelo ficou mais ajustado, importante observar que estamos diminuindo a dispersão dos erros, vamos visualizar a previsão no gráfico.

Figura 53: Modelo SARIMA – Previsão e Teste.



Fonte: Autor

Figura 53: Modelo SARIMA – Treino, Previsão e Teste.



Fonte: Autor

6.6. Aplicação da Função Auto ARIMA

Agora iremos aplicar a função Auto ARIMA e avaliar o modelo que for escolhido. Primeiramente vamos considerar a abordagem mais "inteligente", configurando o parâmetro stepwise=True.

Figura 54: Função Auto ARIMA - Código Função auto_arima (stepwise=True).

```
auto_arima_stpw_t = auto_arima(df_orders_filtred_day
                               , start_p=1, max_p=6, start_q=1, max_q=6, m=7
                               , Seasonal=True, start_P=0,d=1,D=1
                               , trace=True
                               , error_action='ignore', suppress_warnings=True
                               , stepwise=True)
```

Fonte: Autor

A função definiu o modelo SARIMAX(2,1,0)(0,1,1,7). Vamos aplica-lo e fazer a apuração de suas métricas.

Figura 55: Função Auto ARIMA - Treino e Previsão e Avaliação do Modelo SARIMAX(2,1,0)(0,1,1,7).

```
#Treino
auto_arima_f_train = SARIMAX(ds_day_train,freq='D',order=(4,1,0), seasonal_order=(0,1,1,7)).fit()
#Previsão
auto_arima_f_test = SARIMAX(ds_day_test,freq='D',order=(4,1,0), seasonal_order=(0,1,1,7)).fit(auto_arima_f_train.params)
#Avaliação
y_true_auto_arima = ds_day_test.values
y_pred_auto_arima = auto_arima_f_test.predict(typ='levels')[1:]
prediction_teste_error('Auto Arima (SARIMAX(4,1,0)(0,1,1)7)',y_true=y_true[1:],y_pred=y_pred_auto_arima)
```

```
Modelo Auto Arima (SARIMAX(4,1,0)(0,1,1)7) Resultados: MAE : 5196.025778792153, MAPE : 18.442043386912776, MSE : 42504593.08898531, RMSE : 6519.554669529607
```

Fonte: Autor

Vamos avaliar a execução do Auto ARIMA, com o parâmetro stepwise = False, ou seja, em seu modo Parallelized.

Figura 56: Função Auto ARIMA - Código Função auto_arima (stepwise=False).

```
auto_arima_stpw_f = auto_arima(df_orders_filtred_day
                               , start_p=1, max_p=6, start_q=1, max_q=6, m=7
                               , Seasonal=True, start_P=0,d=1,D=1
                               , trace=True
                               , error_action='ignore', suppress_warnings=True
                               , stepwise=False)
```

Fonte: Autor

A função escolheu o modelo SARIMAX(4,1,0)(0,1,1,7). Observamos que o método de avaliação AIC é menor, ou seja, esta configuração ficou melhor que a anterior, vamos aplicar o modelo para ver como será a nova previsão.

Figura 57: Função Auto ARIMA – Treino, Previsão e Avaliação do Modelo SARIMAX(4,1,0)(0,1,1,7) (stepwise=False).

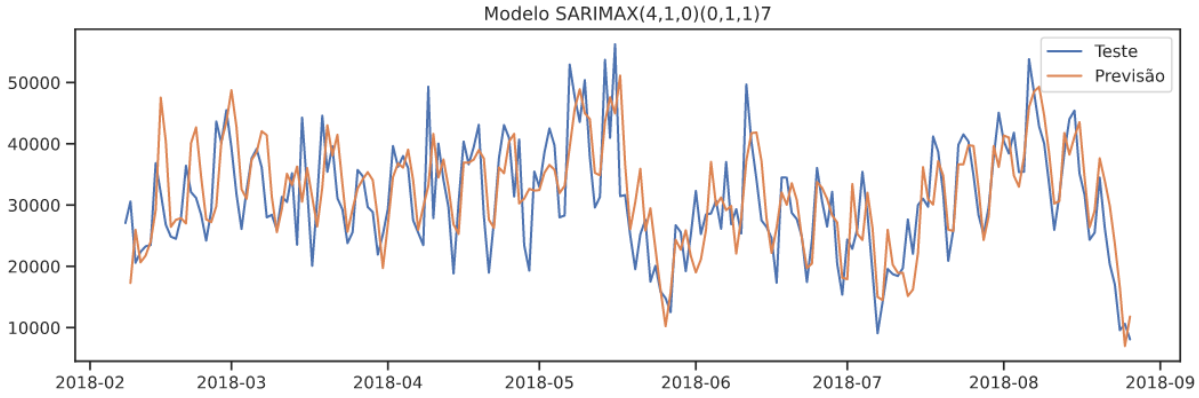
```
#Treino
auto_arima_f_train = SARIMAX(ds_day_train,freq='D',order=(4,1,0), seasonal_order=(0,1,1,7)).fit()
#Previsão
auto_arima_f_test = SARIMAX(ds_day_test,freq='D',order=(4,1,0), seasonal_order=(0,1,1,7)).fit(auto_arima_f_train.params)
#Avaliação
y_true_auto_arima = ds_day_test.values
y_pred_auto_arima = auto_arima_f_test.predict(typ='levels')[1:]
prediction_teste_error('Auto Arima (SARIMAX(4,1,0)(0,1,1)7)',y_true=y_true[1:],y_pred=y_pred_auto_arima)
```

```
Modelo Auto Arima (SARIMAX(4,1,0)(0,1,1)7) Resultados: MAE : 5196.025778792153, MAPE : 18.442043386912776, MSE : 42504593.08898531, RMSE : 6519.554669529607
```

Fonte: Autor

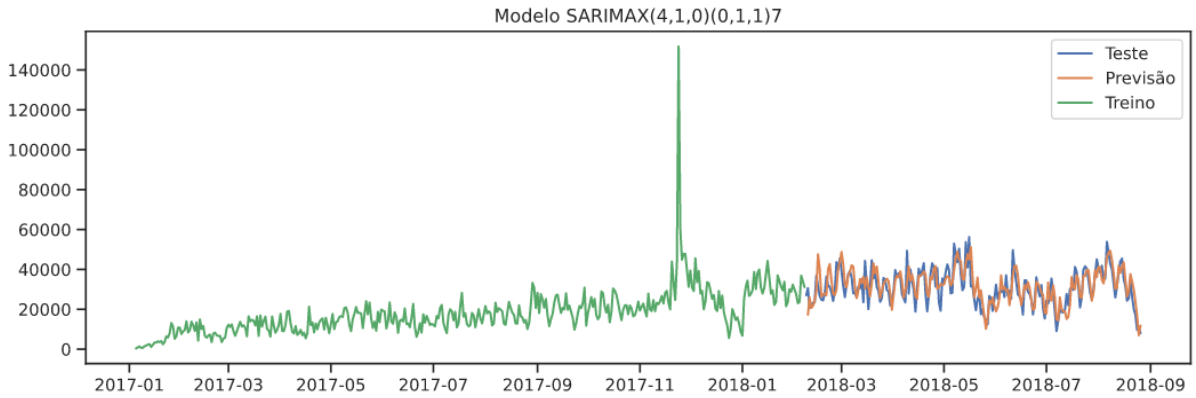
Como tivemos um melhor nível de acerto com a configuração SARIMAX(4,1,0)(0,1,1,7), vamos fazer a sua representação gráfica.

Figura 58: Função Auto ARIMA – Gráfico Previsão e Teste (stepwise=Falso).



Fonte: Autor

Figura 59: Função Auto ARIMA – Gráfico Treino, Previsão e Teste (stepwise=Falso).



Fonte: Autor

6.7. Modelo Facebook Prophet

Para o Facebook Prophet, foi necessário fazer um ajuste no hiperparâmetro changepoint_prior_scale=0.2, este hiperparâmetro define o grau prioridade para os pontos que tiverem as maiores variações, ou seja, define a significância que o modelo dará para estas observações, neste caso, foi necessário diminuir para ter um resultado mais satisfatório. Abaixo segue a execução do modelo.

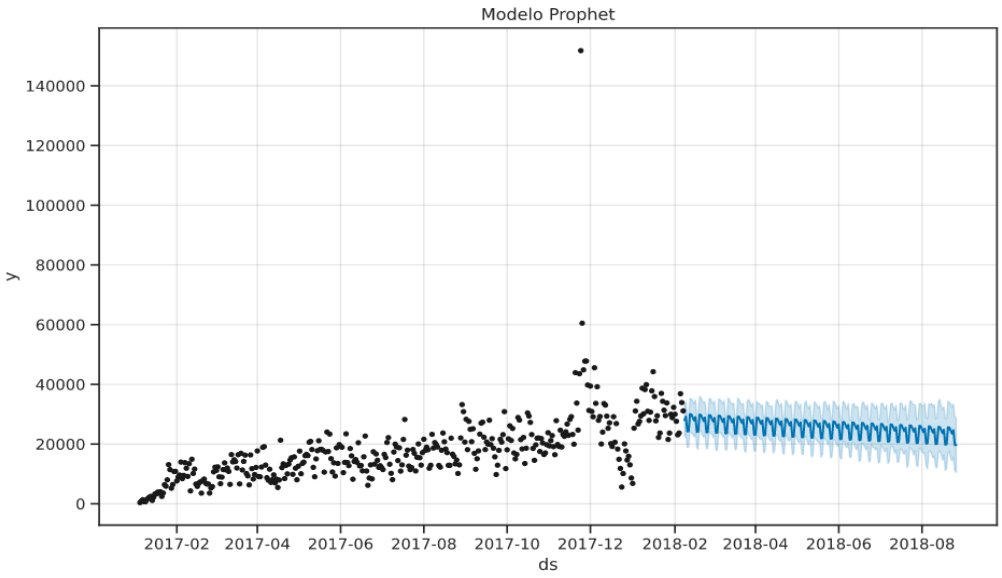
Figura 60: Modelo Prophet - Execução do Modelo.

```
#Ajsute DF para utilização do Prophet
df_prev_prophet_train = pd.DataFrame(data=ds_day_train,index=ds_day_train.index)
df_prev_prophet_train.dropna(inplace=True)
df_prev_prophet_train.reset_index(inplace=True)
df_prev_prophet_train.columns=['ds','y']

#Treino
prophet_model = Prophet(changepoint_prior_scale=0.2,interval_width=0.5)
prophet_model.fit(df_prev_prophet_train)

prophet_forecast = prophet_model.make_future_dataframe(periods=len(ds_day_test), freq='D',include_history=False)
prophet_forecast = prophet_model.predict(prophet_forecast)

#Avaliação
test = pd.DataFrame(ds_day_test)
test.reset_index(inplace=True)
test.columns = ['ds', 'y']
cmp_df = make_comparison_dataframe(test, prophet_forecast)
cmp_df['resid'] = cmp_df['y']-cmp_df['yhat']

prediction_teste_error('Previsão Prophet',y_true=cmp_df['y'],y_pred=cmp_df['yhat'])
```

```
INFO:fbprophet:Disabling yearly seasonality. Run prophet with yearly_seasonality=True to override this.
INFO:fbprophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
Modelo Previsão Prophet Resultados: MAE : 7644.659048719121, MAPE : 24.68231758562615, MSE : 94980257.73418626, RMSE : 9745.781535320102
```

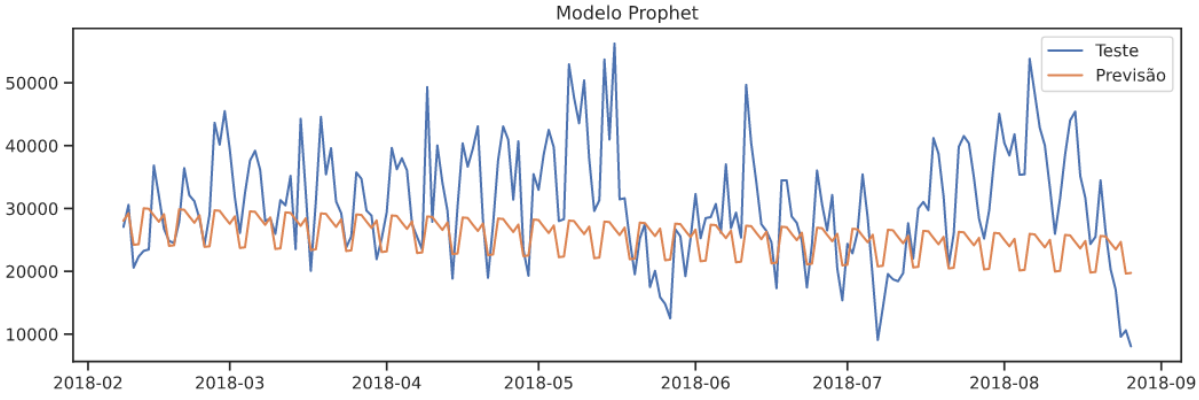
Fonte: Autor

Figura 61: Modelo Prophet - Gráfico Padrão do Prophet.



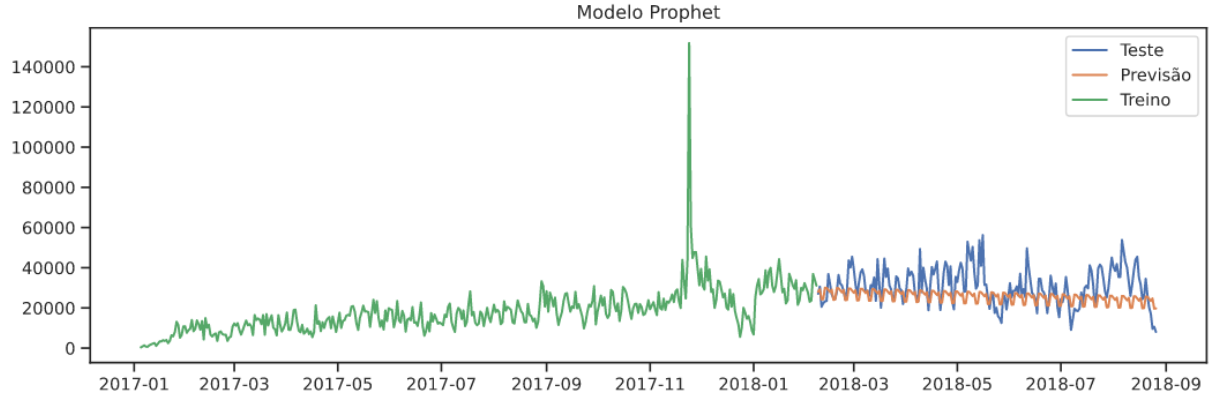
Fonte: Autor

Figura 62: Modelo Prophet - Previsão e Teste.



Fonte: Autor

Figura 63: Modelo Prophet - Treino, Previsão e Teste.



Fonte: Autor

Neste modelo, ao aplicar o ajuste no hiperparâmetro changepoint_prior_scale, tivemos um maior nível de acerto, porém ele nos retornou uma leve tendência de queda. Podemos observar também que este modelo projetou muito bem os fenômenos de sazonalidade, tendo em vista que a série possui um componente forte de sazonalidade.

6.8. Modelo Holt-Winters

Para o modelo Holt Winters foi considerado o ajuste de tendência aditiva e sazonalidade multiplicativa e definido o período de sazonalidade semanal para 7 períodos. Na Figura 64 está a definição, treinamento, resultado e avaliação do modelo.

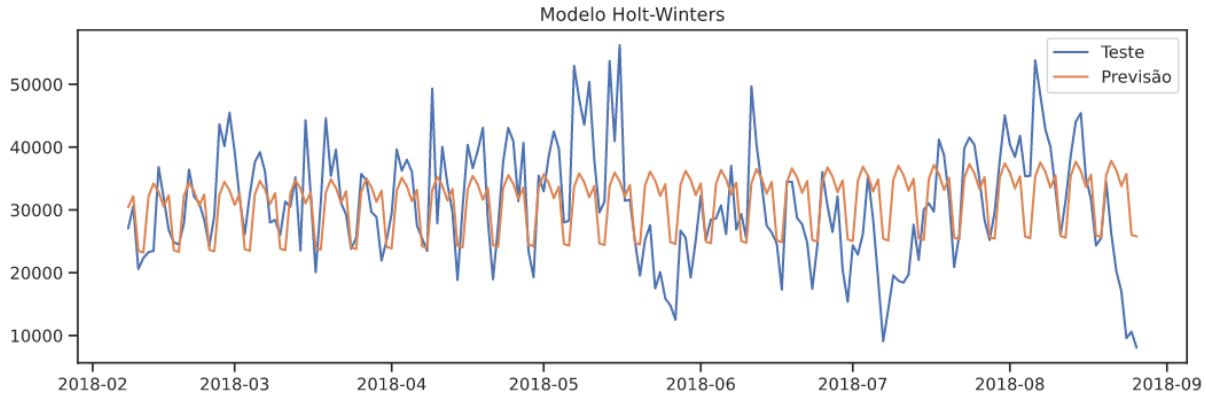
Figura 63: Modelo Holt Winters - Avaliação.

```
forecast = fitted.forecast(steps=200)
prediction_teste_error('Previsão Holt-Winters',y_true=df_prev_holtwinters_test,y_pred=forecast)
```

```
Modelo Previsão Holt-Winters Resultados: MAE : 6264.604431066753, MAPE : 25.162211489164104, MSE : 65626474.90973051, RMSE : 8101.016905903265
```

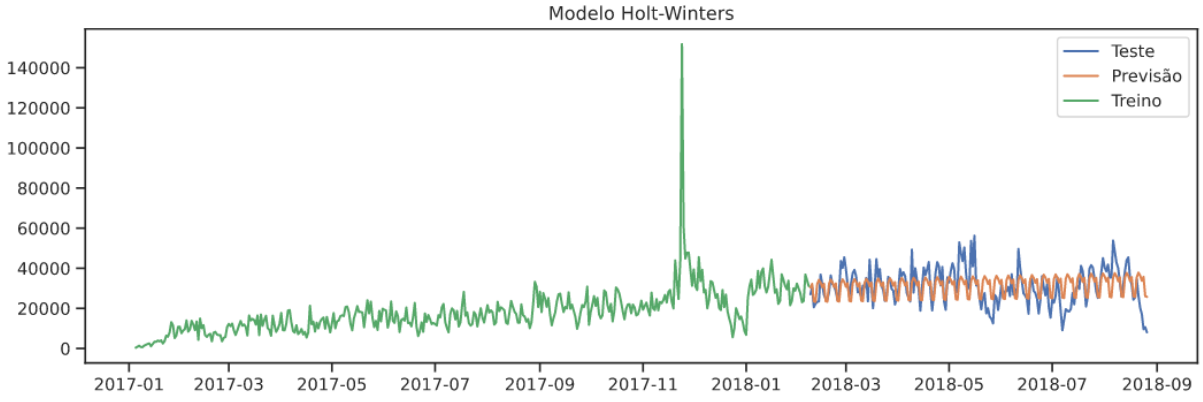
Fonte: Autor

Figura 64: Modelo Holt Winters – Previsão e Teste.



Fonte: Autor

Figura 65: Modelo Holt Winters – Treino, Previsão e Teste.



Fonte: Autor

Assim como no modelo Facebook Prophet o modelo Hotl-Winters projetou muito bem o componente de tendência, e a sazonalidade, porém não conseguimos extrapolar os ruídos, ficando assim com uma acurácia menor sobre as métricas aplicadas.

6.9. Avaliação dos Modelos

Na figura abaixo, Figura 66, temos uma tabela resumo com os resultados das métricas de apuração.

Figura 66: Avaliação dos Modelos - Quadro Resumo.

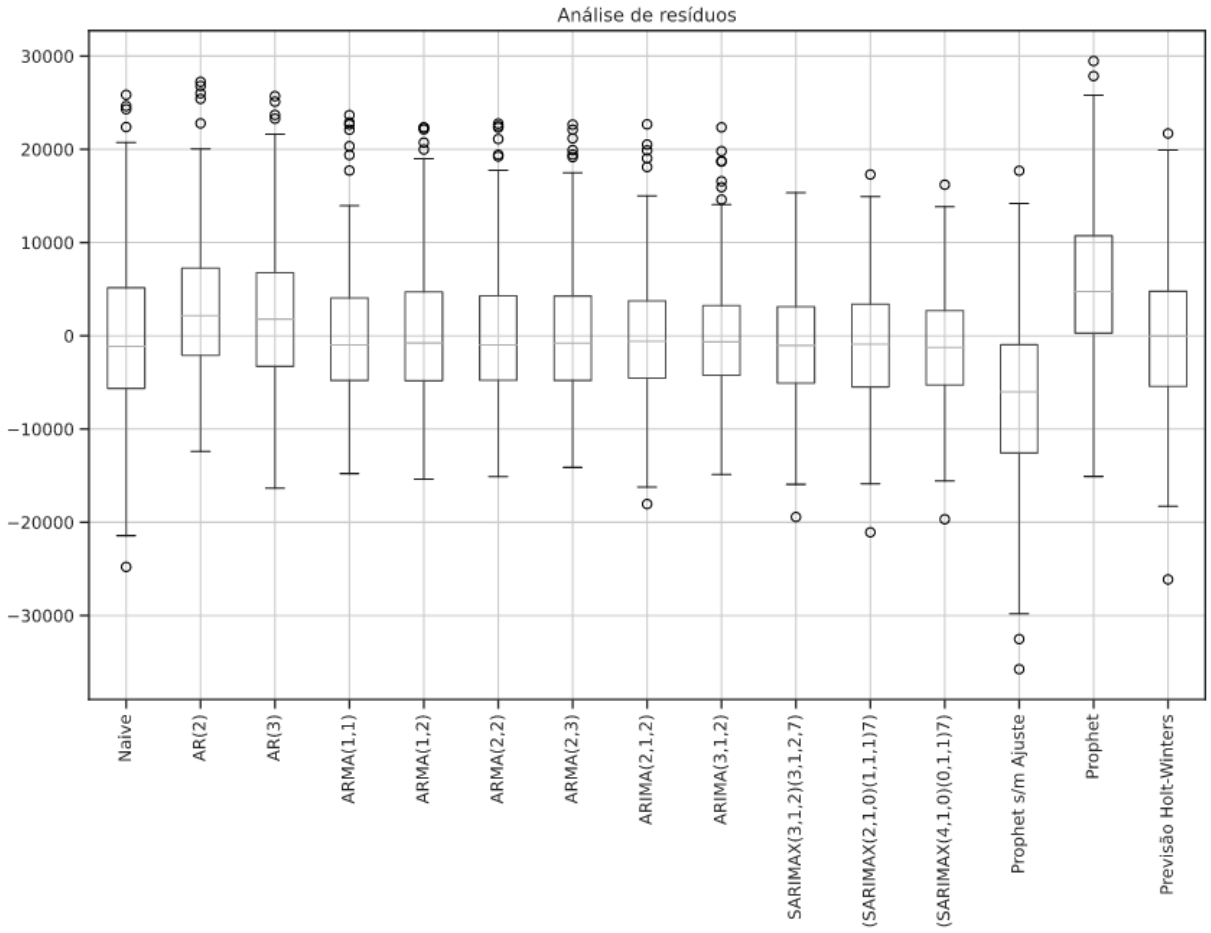
Model	MAE	MAPE	MSE	RMSE
Naive	6466.88	21.95	6.797612e+07	8244.76
AR(2)	6100.11	19.84	6.642184e+07	8149.96
AR(3)	6219.80	20.76	6.473259e+07	8045.66
ARMA(1,1)	5737.59	21.22	5.433920e+07	7371.51
ARMA(1,2)	5773.79	21.33	5.424582e+07	7365.18
ARMA(2,2)	5731.92	21.18	5.373393e+07	7330.34
ARMA(2,3)	5742.49	21.22	5.330552e+07	7301.06
ARIMA(2,1,2)	5529.02	19.58	5.187071e+07	7202.13
ARIMA(3,1,2)	5227.11	18.41	4.704727e+07	6859.10
SARIMAX(3,1,2)(3,1,2,7)	5116.20	18.22	4.276736e+07	6539.68
(SARIMAX(2,1,0)(1,1,1)7)	5256.88	18.47	4.361534e+07	6604.19
(SARIMAX(4,1,0)(0,1,1)7)	5196.03	18.44	4.250459e+07	6519.55
Prophet s/m Ajuste	8989.31	40.31	1.308509e+08	11439.01
Prophet	7644.66	24.68	9.498026e+07	9745.78
Previsão Holt-Winters	6264.60	25.16	6.562647e+07	8101.02

Fonte: Autor

Podemos observar que o modelo que tem a menor média absoluta dos erros e está com o menor percentual de erro é o modelo SARIMAX(3,1,2)(3,1,2)7, cujos os hiperparâmetros foram definidos manualmente.

Porém, o modelo que teve a menor dispersão entre os erros, e seus erros permaneceram mais próximos de zero, foi o modelo definido pela função Auto ARIMA, quando configuramos com o parâmetro Stepwise = False. Neste caso o modelo que foi sugerido foi o (SARIMAX(4,1,0)(0,1,1)7). Para observarmos de forma mais clara a dispersão podemos visualizar o gráfico de caixa na Figura 67, que está logo abaixo.

Figura 66: Avaliação dos Modelos – Dispersão dos Erros.



Fonte: Autor

Foi apresentado abaixo, mais algumas estatísticas dos erros.

Figura 66: Avaliação dos Modelos – Estatísticas dos Erros.

	Naive	AR(2)	AR(3)	ARMA(1,1)	ARMA(1,2)	ARMA(2,2)	ARMA(2,3)	ARIMA(2,1,2)	ARIMA(3,1,2)	SARIMAX(3,1,2)(3,1,2,7)	(SARIMAX(2,1,0)(1,1,1)7)	(SARIMAX(4,1,0)(0,1,1)7)	Prophet s/m Ajuste	Prophet	Previsão Holt-Winters
count	199.0	198.0	197.0	200.0	200.0	200.0	200.0	199.0	199.0	199.0	199.0	199.0	200.0	200.0	200.0
mean	-95.0	3119.0	2289.0	13.0	34.0	33.0	36.0	-29.0	-5.0	-752.0	-849.0	-1003.0	-6668.0	5333.0	-475.0
std	8265.0	7548.0	7733.0	7390.0	7384.0	7349.0	7319.0	7220.0	6876.0	6513.0	6566.0	6458.0	9318.0	8178.0	8107.0
min	-24788.0	-12399.0	-16353.0	-14778.0	-15364.0	-15106.0	-14118.0	-18044.0	-14883.0	-19434.0	-21079.0	-19675.0	-35757.0	-15081.0	-26133.0
25%	-5642.0	-2096.0	-3281.0	-4795.0	-4828.0	-4780.0	-4795.0	-4534.0	-4234.0	-5073.0	-5490.0	-5284.0	-12566.0	275.0	-5427.0
50%	-1130.0	2159.0	1793.0	-991.0	-761.0	-980.0	-774.0	-598.0	-653.0	-1036.0	-901.0	-1247.0	-6006.0	4740.0	-30.0
75%	5153.0	7272.0	6757.0	4046.0	4714.0	4279.0	4262.0	3744.0	3254.0	3121.0	3400.0	2706.0	-961.0	10729.0	4770.0
max	25845.0	27241.0	25709.0	23669.0	22338.0	22759.0	22648.0	22681.0	22357.0	15344.0	17299.0	16202.0	17701.0	29466.0	21700.0

Fonte: Autor

Concluímos, em nossas análises que, o modelo que conseguimos chegar em uma configuração mais assertiva de previsão foi o SARIMAX(3,1,2)(3,1,2)7. Pois o mesmo possui o maior nível de acurácia e uma dispersão de erros aceitável.

É importante considerar que para previsões onde erros com valores maiores não são tolerados, por exemplo exames médicos ou projeção de preços, deve-se optar por modelos com o menor índices de MSE e RMSE, pois estas métricas penalizam este tipo de acontecimento, para nossa previsão é mais importante ter uma média mais acertada do que uma menor dispersão.

7. Links

Os materiais envolvidos neste trabalho incluem uma apresentação em vídeo divulgada na plataforma do Youtube, Datasets e notebook com os *scritps* disponibilizados na plataforma do Github e Google Drive, os links estão listados abaixo.

Link para o vídeo: https://www.youtube.com/watch?v=P1PwPvnyDVo
Link para o repositório: https://github.com/junior8801/TCC_PUC_MINAS

REFERÊNCIAS

VANDERPLAS, Jake. **Python Data Science Handbook: Essential Tools for Working with Data.** Cidade: Sebastopol. O'Reilly Media, Inc, 2017.

WERNER, Liane. **Um modelo composto para realizar previsão de demanda através da integração da combinação de previsões e do ajuste baseado na opinião.** Cidade: Porto Algre. 2004. Disponível em: < https://www.lume.ufrgs.br/bitstream/handle/10183/4189/000453479.pdf?sequence=1 >. Acesso em: 13/01/2021.

REIS, Marcelo Meneses. **Análise de Séries Temporais.** Disponível em: < https://www.inf.ufsc.br/~marcelo.menezes.reis/Cap4.pdf >. Acesso em: 13/01/2021.

MANCUSO, Aline Castello Branco. **Uma Investigação de Desempenho de Métodos de Combinação de Previsões: Simulada e Aplicada.** Disponível em: < https://www.lume.ufrgs.br/bitstream/handle/10183/75918/000891325.pdf?sequence=1 >. Acesso em: 13/01/2021.

EHLERS, Ricardo Sanders. **Análise de Séries Temporais.** Disponível em: < http://www.each.usp.br/rvicente/AnaliseDeSeriesTemporais.pdf >. Acesso em: 13/01/2021.

OLIVEIRA, Pedro Carvalho. **Séries Temporais: Analisar o Passado, Predizer o Futuro.** Disponível em: < https://student.dei.uc.pt/~pcoliv/reports/ct_timeseries.pdf >. Acesso em: 13/01/2021.

PORTILLA, José. **Using Python and Auto ARIMA to Forecast Seasonal Time Series.** Disponível em: < https://medium.com/@josemarcialportilla/usingpython-and-auto-arima-to-forecast-seasonal-time-series-90877adff03c >. Acesso em: 13/01/2021.

ZEVIANI, Walmes Marques. **Manipulação de dados no Python.** Disponível em: < http://leg.ufpr.br/~walmes/ensino/dsbd/01-python-pandas.html#1 >. Acesso em: 13/01/2021.

MIGON, Hélio. **Análise de Séries Temporais.** Disponível em: < http://www.dme.ufrj.br/dani/pdf/slidespartefrequentista.pdf >. Acesso em: 13/01/2021.

BECKER, Marcel Henrique. **Modelos para Previsão em Séries Temporais: uma Aplicação para a Taxa de Desemprego na Região Metropolitana de Porto Alegre.** Disponível em: < https://www.lume.ufrgs.br/bitstream/handle/10183/29107/000775660.pdf?sequence=1#:~:text=Os%20m%C3%A9todos%20para%20previs%C3%A3o%20de,passado%20(WHEEL%2C%20WRIGHT%2C%201985 >. Acesso em: 13/01/2021.

MATTOS, Rogério Silva. **Tendências e Raízes Unitárias.** Disponível em: < https://www.ufjf.br/wilson_rotatori/files/2011/05/Tendencias-e-Raizes-Unitarias-2018.pdf >. Acesso em: 13/01/2021.

VELASCO, Leandro Henz. **Previsão de Demanda de Acessos Móveis no Sistema de Telefonia Brasileiro.** Disponível em: < https://www.lume.ufrgs.br/bitstream/handle/10183/13454/000641161.pdf?...1 >. Acesso em: 13/01/2021.

PANDAS. Disponível em: < https://pandas.pydata.org/ >. Acesso em: 13/01/2021.

NUMPY. Disponível em: < https://numpy.org/ >. Acesso em: 13/01/2021.

SEABORN. Disponível em: < https://seaborn.pydata.org/ >. Acesso em: 13/01/2021.

PROPHET. Disponível em: < https://facebook.github.io/prophet/docs/quick_start.html >. Acesso em: 13/01/2021.

PORTALACTION. Disponível em: < http://www.portalaction.com.br/series-temporais >. Acesso em: 13/01/2021.