

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

Mário Marcos Martins de Souza

**MODELOS PREDITIVOS UTILIZANDO SÉRIES TEMPORAIS DAS CORRIDAS DE
FÓRMULA 1**

Belo Horizonte

2023

Mário Marcos Martins de Souza

**MODELOS PREDITIVOS UTILIZANDO SÉRIES TEMPORAIS DAS CORRIDAS DE
FÓRMULA 1**

**Trabalho de Conclusão de Curso
apresentado ao Curso de Especialização em
Ciência de Dados e Big Data.**

Belo Horizonte

2023

AGRADECIMENTO

Agradeço a Deus por mais esta etapa vencida.

A minha esposa Vanuza e ao meu filho Mário Marcos Filho pela paciência, parceria e apoio nos momentos de ausência e estudo.

Aos professores e a PUC Minas pelo conhecimento compartilhado e adquirido durante o curso.

SUMÁRIO

| | |
|---|-----------|
| 1. Introdução | 5 |
| 1.1. Contextualização | 5 |
| 1.2. O problema proposto | 11 |
| 2. Coleta de Dados | 11 |
| 3. Processamento/Tratamento de Dados | 20 |
| 4. Análise e Exploração dos Dados | 26 |
| 4.1 - Análise e Exploração dos Dados - Piloto Hamilton | 33 |
| 4.2 - Análise e Exploração dos Dados - Piloto Verstappen | 35 |
| 4.3 - Análise e Exploração dos Dados - Piloto Bottas | 36 |
| 4.4 - Análise e Exploração dos Dados - em conjunto. | 38 |
| 5. Criação de Modelos de Machine Learning | 40 |
| 5.1. Modelos de Machine Learning com Facebook Prophet | 40 |
| 5.2. Modelos de Machine Learning com AUTO-ARIMA | 42 |
| 5.3. Modelos de Machine Learning com RNN (LSTM) | 47 |
| 6. Apresentação dos Resultados | 51 |
| 6.1. Modelo Preditivo com Facebook Prophet | 51 |
| 6.2. Modelo Preditivo com AUTO-ARIMA | 54 |
| 6.3. Modelos de Machine Learning com RNN (LSTM) | 55 |
| 7. Links | 57 |

1. Introdução

1.1. Contextualização

A Fórmula 1 é um ambiente de pessoas aficionadas por este espetacular, apaixonante e fascinante mundo das corridas de carros, e que traz dados que possuem características de nos trazer para a paixão estampada nos rostos das pessoas, que são as corridas, circuitos, pilotos, construtores e os países envolvidos nesta jornada que emociona e contagia o público de uma maneira geral.

Neste sentido, o mundo das corridas são diretamente afetados por vários fatores em cada país como situações climáticas, políticas, econômicas, empresariais, de saúde e principalmente ligado ao ambiente automobilístico em diversos países, trazendo assim incertezas e afetando diretamente cada corrida em cada país por onde passam, na realização destes eventos. Contudo é um ambiente positivo e de uma paixão extrema por parte dos seus espectadores que não abrem mão de torcerem fielmente para os pilotos, principalmente os conterrâneos.

Outro ponto importante é ressaltar o uso da tecnologia neste ambiente automobilístico entrando cada mês mais no detalhe de cada peça construída de cada carro em sua performance fora e dentro das corridas.

Neste mundo automobilístico da Fórmula 1, já é aplicado conhecimentos de diversas técnicas de Modelagem Estatística e Inteligência Artificial no sentido de aplicar em séries temporais para tentar apoiar nas decisões de várias situações e analisar a performance dos diversos pilotos, times e diversas informações correlacionadas.

Este trabalho utilizará técnicas de Modelos Preditivos em séries temporais da Fórmula 1. Foi desenvolvido um script em Python com utilização de várias bibliotecas básicas de uso comum e mais algumas necessárias para o desenvolvimento deste trabalho. O objetivo é observar as tendências das corridas e prever vencedores de uma determinada temporada.

Iremos discorrer sobre a história e sobre a estrutura padrão do Machine Learning, principalmente no que diz respeito a este trabalho.

Introdução ao Machine Learning

(Brett Lantz, 2013), Se quisermos acreditar nas histórias de ficção científica, ensinar as máquinas a aprender, levará inevitavelmente a guerras apocalípticas entre as máquinas e seus criadores. Nos estágios iniciais, os computadores são ensinados a jogar jogos simples de jogo da velha e xadrez.

Mais tarde, as máquinas passam a controlar os semáforos e as comunicações, seguidas por drones militares e mísseis. A evolução das máquinas dá uma guinada sinistra quando os computadores tornam-se sencientes e aprendem a se auto-ensinar.

Não tendo mais necessidade de programadores humanos, a humanidade é então "deletada". Felizmente, no momento em que este livro foi escrito, as máquinas ainda exigiam a entrada do usuário. Suas impressões sobre o aprendizado de máquina podem ser fortemente influenciadas por esses tipos de representações de inteligência artificial na mídia de massa.

E mesmo que possa haver um toque de verdade nessas histórias, na realidade, o aprendizado de máquina está focado em aplicações mais práticas. A tarefa de ensinar um computador a aprender está mais ligada a um problema específico que seria um computador que pode jogar, refletir sobre filosofia ou responder a perguntas triviais. O aprendizado de máquina é mais parecido com treinar um funcionário do que criar um filho.

A origem do machine learning

Desde o nascimento, somos inundados com dados. Os sensores do nosso corpo - olhos, ouvidos, nariz, língua e nervos - são continuamente atacados com dados brutos que nosso cérebro traduz em imagens, sons, cheiros, sabores e texturas. Usando a linguagem, somos capazes de compartilhar essas experiências com outras pessoas.

Os primeiros bancos de dados registravam informações do ambiente observável. Os astrônomos registraram padrões de planetas e estrelas; biólogos observaram resultados de experimentos de cruzamento de plantas e animais; e as cidades registraram pagamentos de impostos, surtos de doenças e populações. Cada um deles exigia que um ser humano primeiro observe e, em segundo lugar, registre a observação. Hoje, essas observações estão cada vez mais automatizadas e registradas sistematicamente em bancos de dados computadorizados em constante crescimento.

A invenção de sensores eletrônicos contribuem adicionalmente para um aumento na riqueza dos dados registrados. Sensores especializados veem, ouvem, cheiram ou degustam. Esses sensores processam os dados de maneira muito diferente do que um ser humano faria e, de muitas maneiras, isso é um benefício. Sem a necessidade de tradução para a linguagem humana, os dados sensoriais brutos permanecem objetivos.

Entre bancos de dados e sensores, muitos aspectos de nossas vidas são registrados. Governos, empresas e indivíduos estão registrando e relatando todas as formas de informação, do monumental ao mundano. Sensores meteorológicos registram dados de temperatura e pressão, câmeras de vigilância observam calçadas e túneis de metrô, e todos os tipos de comportamentos eletrônicos são monitorados: transações, comunicações, amizades e muitos outros.

Segundo (Brett Lantz, 2013), Qualquer tarefa de aprendizado de máquina pode ser dividida em uma série de tarefas mais gerenciáveis como discriminado abaixo:

- A. Coleta de dados: Se os dados são escritos em papel, registrados em arquivo de texto e planilhas, ou armazenados em um banco de dados SQL, você precisará reuni-los em um formato eletrônico adequado para análise. Esses dados servirão como o aprendizado material que um algoritmo usa para gerar conhecimento acionável.

- B. Explorar e preparar os dados:** A qualidade de qualquer projeto de machine learning é amplamente baseada na qualidade dos dados que se usa. Esta etapa no processo de machine learning tende a exigir muita intervenção humana. Uma estatística frequentemente citada sugere que 80 por cento do esforço em machine learning é dedicado aos dados. Muito desse tempo é gasto aprendendo mais sobre os dados e suas nuances durante uma prática chamada exploração de dados.
- C. Treinar um modelo de dados:** No momento em que os dados estiverem preparados para análise, é provável que você tenha uma ideia do que espera aprender com os dados. A tarefa específica de Machine Learning informará a seleção de um algoritmo apropriado, e o algoritmo representará os dados na forma de um modelo.
- D. Avaliar o desempenho do modelo:** Porque cada modelo de machine learning resulta em uma solução tendenciosa para o problema de aprendizagem, é importante avaliar o quão bem o algoritmo aprendeu com sua experiência. Dependendo do tipo de modelo usado, pode-se avaliar a precisão de um modelo usando um conjunto de dados de teste, ou pode-se ser preciso desenvolver medidas de desempenho específicas para a aplicação pretendida.
- E. Melhorar o desempenho do modelo:** Se um melhor desempenho for necessário, ele se torna necessário utilizar estratégias mais avançadas para aumentar o desempenho do modelo. Às vezes, pode ser necessário mudar para um tipo diferente de modelo completamente. Você pode precisar complementar seus dados com mais dados, ou realizar o trabalho preparatório adicional como na etapa com mais processos.

Para os processamentos aqui descritos, vamos aplicar algoritmos de séries temporais com regressão, classificação e predição sobre os dados.

Há diversos modelos de algoritmos de análise que iremos utilizar já disponíveis com performance e desempenho adequados para o trabalho.

No projeto que compara desempenho de diversos algoritmos optamos por escolher os mais precisos após vários testes realizados, apesar de haver similaridades em alguns destes algoritmos.

Iremos aplicar alguns modelos conforme descrito abaixo:

Modelos de Machine Learning com Facebook Prophet:

O Facebook Prophet é uma ferramenta de previsão de séries temporais que usa um modelo aditivo para decompor os dados em tendência, sazonalidade e feriados, e então ajusta múltiplos modelos de regressão para cada componente para fazer previsões. Foi desenvolvido pelo time de Data Science do Facebook e é um projeto de código aberto.

O Prophet foi projetado para ser fácil de usar e oferece muitos recursos que o tornam conveniente para a previsão de séries temporais, como detecção automática de pontos de mudança, modelos de sazonalidade personalizáveis e a capacidade de lidar com dados ausentes e valores discrepantes. Ele também inclui funcionalidades integradas para visualizar previsões e diagnosticar o desempenho do modelo.

O Prophet tem sido utilizado em uma ampla gama de setores para previsão, incluindo varejo, finanças e transporte. Ele também foi utilizado para prever casos e mortes de COVID-19

Modelos de Machine Learning com AUTO-ARIMA:

AUTO-ARIMA é um modelo de machine learning usado para previsão de séries temporais. ARIMA significa Autoregressive Integrated Moving Average, que é um modelo estatístico usado para analisar dados de séries temporais.

AUTO-ARIMA é uma extensão do ARIMA que seleciona automaticamente os parâmetros ótimos para o modelo, tornando mais fácil de usar e reduzindo o risco de erro humano na seleção de parâmetros.

O modelo AUTO-ARIMA funciona analisando os dados históricos e identificando padrões e tendências. Em seguida, ele usa esses padrões e tendências para fazer previsões sobre valores futuros na série temporal. O modelo é capaz de lidar com diferentes tipos de dados de séries temporais, incluindo dados sazonais e não sazonais.

O modelo AUTO-ARIMA é amplamente utilizado em vários campos, incluindo finanças, economia e engenharia. Ele pode ser usado para uma ampla variedade de aplicações, como prever preços de ações, prever volumes de vendas e estimar a demanda por produtos ou serviços.

Para usar o AUTO-ARIMA, você geralmente precisa fornecer ao modelo dados históricos de séries temporais e especificar o número de períodos de tempo para previsão. O modelo, então, usará os dados para selecionar automaticamente os parâmetros ótimos para o modelo ARIMA e gerar uma previsão para o número especificado de períodos de tempo.

Em geral, o AUTO-ARIMA é um modelo de machine learning poderoso para previsão de séries temporais que pode ajudar a melhorar a precisão e a eficiência em várias aplicações.

Modelos de Machine Learning com RNN (LSTM):

Refere-se ao uso de modelos de rede neural recorrente (RNN), em particular a arquitetura Long Short-Term Memory (LSTM), em aplicações de aprendizado de máquina. Esses modelos são especialmente úteis para lidar com dados de séries temporais, como preços de ações ou padrões climáticos.

Os modelos LSTM são projetados para lidar com dependências temporais de longo prazo, usando estados internos para recolher informações importantes por períodos prolongados de tempo. Eles têm sido amplamente utilizados em aplicações como processamento de linguagem natural, reconhecimento de fala e previsão de séries temporais.

Para implementar modelos LSTM, podem ser utilizadas bibliotecas como Keras e TensorFlow. Essas bibliotecas fornecem implementações

pré-construídas da arquitetura LSTM, permitindo que os usuários criem modelos rapidamente sem a necessidade de codificação de baixo nível. Elas também oferecem uma variedade de ferramentas para visualizar e depurar redes neurais.

1.2. O problema proposto

No problema será utilizado Análise Exploratória e Modelagem Preditiva para extração de informações importantes de séries temporais das diversas corridas de Fórmula 1 para auxiliar na predição de qual piloto será campeão de Fórmula 1 no ano de 2023.

Os dados analisados são dados históricos disponibilizados pela FIA (Federação Internacional de Automobilismo).

O objetivo desta análise é predizer quem será o provável campeão de Fórmula 1 no ano de 2023 e é claro trazer alguns dados adicionais para análise. Iremos analisar os dados diversos dataset para chegarmos ao objetivo proposto.

Os referidos dados se tratam de informações de corridas em diversos países do mundo desde 1950.

O período a ser utilizado na Análise Exploratória e treinamento dos modelos de predição são referentes aos anos de 2018 a 2022. Serão utilizados os registros do ano de 2022 nos testes do treinamento do modelo.

2. Coleta de Dados

Os dados serão obtidos no site do Kaggle através do link: <https://www.kaggle.com/code/guarniere/f-rmula-1-equipes-com-mais-pontos-por-corrida/data> que posteriormente executaremos o notebook, quando será passado um parâmetro com os anos da corrida da fórmula 1 que iremos prever.

O desenvolvimento deste trabalho será com a utilização do Google Colab de onde vamos executar os scripts e consumir os datasets com dados da fórmula 1. E para leitura dos dados será utilizada a biblioteca Pandas na linguagem Python.

Iremos executar como treinamento um período de 05 anos, iniciando com o ano 2018 até 2022.

Iniciamos com a montagem do drive no google colab.

Importando as Bibliotecas

```
import pandas as pd
import numpy as np
import datetime
import matplotlib.pyplot as plt
import plotly.graph_objects as go
import plotly.express as px
```

Acessando o Drive:

```
from google.colab import drive
drive.mount('/content/drive')
```

1 - importando a tabela circuits.csv para o dataset de treinamento:

```
circuits = pd.read_csv('../content/drive/My Drive/Colab Notebooks/Kaggle_F1_world/circuits.csv');
```

O Dataset obtido resultante da tabela circuits.csv terá o formato descrito na tabela abaixo:

| Nome da coluna | Descrição | Tipo |
|----------------|--------------------------------|---------|
| circuitId | Id do circuito | Id |
| circuitRef | Nome de referência do circuito | String |
| name | Nome atual do circuito | String |
| location | Cidade | String |
| country | País | String |
| lat | Latitude | String |
| lng | Longitude | String |
| alt | Altitude | Integer |
| url | Url wikipedia | String |

2 - importando a tabela constructor_results.csv para o dataset de treinamento

```
constructor_results = pd.read_csv('../content/drive/My Drive/Colab Notebooks/Kaggle_F1_world/constructor_results.csv')
```

O Dataset obtido resultante da tabela constructor_results.csv terá o formato descrito na tabela abaixo:

| Nome da coluna | Descrição | Tipo |
|----------------------|--------------------------------|---------|
| constructorResultsId | Id do circuito | Id |
| racelId | Nome de referência do circuito | Id |
| constructorId | Nome atual do circuito | Id |
| points | Cidade | Integer |
| status | País | String |

3 - importando a tabela constructor_standings.csv para o dataset de treinamento

```
constructor_standings = pd.read_csv('../content/drive/My Drive/Colab Notebooks/Kaggle_F1_world/constructor_standings.csv')
```

O Dataset obtido resultante da tabela constructor_standings.csv terá o formato descrito na tabela abaixo:

| Nome da coluna | Descrição | Tipo |
|------------------------|------------------------|---------|
| constructorStandingsId | Id do circuito | Id |
| racelId | Id do race | Id |
| constructorId | Id do construtor | Id |
| points | Pontos | Integer |
| position | Posição final | Integer |
| positionText | Posição final em texto | String |
| wins | Número de vitórias | Integer |

4 - importando a tabela constructors.csv para o dataset de treinamento

```
constructors = pd.read_csv('../content/drive/My Drive/Colab Notebooks/Kaggle_F1_world/constructors.csv')
```

O Dataset obtido resultante da tabela constructor.csv terá o formato descrito na tabela abaixo:

| Nome da coluna | Descrição | Tipo |
|-----------------------|----------------------------------|--------|
| constructorId | Id | Id |
| constructorRef | Nome de referência do construtor | String |
| name | Nome atual do construtor | String |
| nationality | País | String |
| url | Url Wikipedia | String |

5 - importando a tabela driver_standings.csv para o dataset de treinamento

```
driver_standings = pd.read_csv('../content/drive/My Drive/Colab Notebooks/Kaggle_F1_world/driver_standings.csv')
```

O Dataset obtido resultante da tabela driver_standings.csv terá o formato descrito na tabela abaixo:

| Nome da coluna | Descrição | Tipo |
|--------------------------|------------------------|---------|
| driverStandingsId | Id | Id |
| raceId | Id da corrida | Id |
| driverId | id do motorista | Id |
| points | Pontos | Integer |
| position | Posição Final | Integer |
| positionText | Posição final no texto | Integer |
| wins | Número de Vitórias | Integer |

6 - importando a tabela drivers.csv para o dataset de treinamento

```
drivers_raw_df = pd.read_csv('../content/drive/My Drive/Colab Notebooks/Kaggle_F1_world/drivers.csv')
```

O Dataset obtido resultante da tabela drivers.csv terá o formato descrito na tabela abaixo:

| Nome da coluna | Descrição | Tipo |
|------------------|-------------------------|--------|
| driverId | Identificador do piloto | Id |
| driverRef | Referência do motorista | String |
| number | Número do piloto | String |
| code | Código do piloto | String |

| | | |
|-------------|--------------------------|--------|
| forename | Nome próprio | String |
| surname | Sobrenome | String |
| dob | Data de nascimento | Date |
| nationality | Nacionalidade | String |
| url | Página do piloto na wiki | String |

7 - importando a tabela lap_times.csv para o dataset de treinamento

```
lap_times = pd.read_csv('../content/drive/My Drive/Colab Notebooks/Kaggle_F1_world/lap_times.csv')
```

O Dataset obtido resultante da tabela lap_times.csv terá o formato descrito na tabela abaixo:

| Nome da coluna | Descrição | Tipo |
|----------------|----------------------------|---------|
| raceId | Identificador da corrida | Id |
| driverId | Identificador do motorista | Id |
| lap | Número de voltas | Integer |
| position | Posição Final | Integer |
| time | Tempo | String |
| milliseconds | Tempo em milisegundos | Integer |

8 - importando a tabela pit_stops.csv para o dataset de treinamento

```
pit_stops = pd.read_csv('../content/drive/My Drive/Colab Notebooks/Kaggle_F1_world/pit_stops.csv')
```

O Dataset obtido resultante da tabela pit_stops.csv terá o formato descrito na tabela abaixo:

| Nome da coluna | Descrição | Tipo |
|----------------|----------------------------|---------|
| raceId | Identificador da corrida | Id |
| driverId | Identificador do motorista | Id |
| stop | Número de paradas | Integer |
| lap | Número de voltas | Integer |

| | | |
|---------------------|--|----------------|
| | | |
| time | Tempo de parada | Date |
| duration | Duração da parada | String |
| milliseconds | Duração da parada em milisegundos | Integer |

9 - importando a tabela qualifying.csv para o dataset de treinamento

```
qualifying = pd.read_csv('../content/drive/My Drive/Colab Notebooks/Kaggle_F1_world/qualifying.csv')
```

O Dataset obtido resultante da tabela qualifying.csv terá o formato descrito na tabela abaixo:

| Nome da coluna | Descrição | Tipo |
|----------------------|------------------------------------|----------------|
| qualifyId | Identificador | Id |
| raceId | Identificador da corrida | Id |
| driverId | Identificador do motorista | Id |
| constructorId | Identificador do construtor | Id |
| number | Número do carro | Integer |
| position | Posição Final | Integer |
| q1 | Tempo de Qualificação 1 | String |
| q2 | Tempo de Qualificação 2 | String |
| q3 | Tempo de Qualificação 3 | String |

10 - importando a tabela races.csv para o dataset de treinamento

```
races = pd.read_csv('../content/drive/My Drive/Colab Notebooks/Kaggle_F1_world/races.csv')
```

O Dataset obtido resultante da tabela races.csv terá o formato descrito na tabela abaixo:

| Nome da coluna | Descrição | Tipo |
|------------------|---|----------------|
| raceId | Número Acumulativo da corrida no geral | Id |
| year | Ano da corrida | Integer |
| round | Número da corrida no ano | Integer |
| circuitId | Número identificação do circuito | id |
| name | Nome do circuito | String |
| date | Data da corrida | Date |
| time | Hora da corrida | String |

| | | |
|-------------|-----------------------------|--------|
| url | Informações sobre a corrida | String |
| fp1_date | Data da qualificação P1 | Date |
| fp1_time | Hora da qualificação P1 | String |
| fp2_date | Data da qualificação P1 | Date |
| fp2_time | Hora da qualificação P2 | String |
| fp3_date | Data da qualificação P1 | Date |
| fp3_time | Hora da qualificação P3 | String |
| quali_date | Data da qualificação | Date |
| quali_time | Hora da qualificação | String |
| sprint_date | Data da corrida Sprint | Date |
| sprint_time | Hora da corrida Sprint | String |

11 - importando a tabela results.csv para o dataset de treinamento

```
results = pd.read_csv('./content/drive/My Drive/Colab Notebooks/Kaggle_F1_world/results.csv')
```

O Dataset obtido resultante da tabela results.csv terá o formato descrito na tabela abaixo:

| Nome da coluna | Descrição | Tipo |
|-----------------|---------------------------------|---------|
| resultId | Identificação do Resultado | Id |
| raceId | Identificação da corrida | Id |
| driverId | Identificação do piloto | Id |
| constructorId | Identificação do Construtor | Id |
| number | Número do Piloto | Integer |
| grid | Número do grid | Id |
| position | Posição no grid | String |
| positionText | Texto de posição | String |
| positionOrder | Ordem de posição | String |
| points | Pontos | Integer |
| laps | Voltas | Integer |
| time | Tempo | String |
| milliseconds | Milisegundos | String |
| fastestLap | Volta mais rápida | String |
| rank | Classificação | String |
| fastestLapTime | Tempo de Volta mais rápida | String |
| fastestLapSpeed | Velocidade de Volta mais Rápida | String |
| statusId | Identificação do Status | Id |

12 - importando a tabela seasons.csv para o dataset de treinamento

```
seasons = pd.read_csv('../content/drive/My Drive/Colab Notebooks/Kaggle_F1_world/seasons.csv')
```

O Dataset obtido resultante da tabela seasons.csv terá o formato descrito na tabela abaixo:

| Nome da coluna | Descrição | Tipo |
|----------------|---------------|---------|
| year | Ano | Integer |
| url | Url wikipedia | String |

13 - importando a tabela sprint_results.csv para o dataset de treinamento

```
sprint_results = pd.read_csv('../content/drive/My Drive/Colab Notebooks/Kaggle_F1_world/sprint_results.csv')
```

O Dataset obtido resultante da tabela sprint_results.csv terá o formato descrito na tabela abaixo:

| Nome da coluna | Descrição | Tipo |
|----------------|-----------------------------|---------|
| resultId | Identificação do Resultado | Id |
| raceId | Identificação da corrida | Id |
| driverId | Identificação do piloto | Id |
| constructorId | Identificação do Construtor | Id |
| number | Número do Piloto | Integer |
| grid | Número do grid | Id |
| position | Posição no grid | String |
| positionText | Texto de posição | String |
| positionOrder | Ordem de posição | String |
| points | Pontos | Integer |
| laps | Volts | Integer |
| time | Tempo | String |
| milliseconds | Milisegundos | String |
| fastestLap | Volta mais rápida | String |
| fastestLapTime | Tempo de Volta mais rápida | String |
| statusId | Identificação do Status | Integer |

14 - importando a tabela status.csv para o dataset de treinamento

```
status = pd.read_csv('../content/drive/My Drive/Colab Notebooks/Kaggle_F1_world/status.csv')
```

O Dataset obtido resultante da tabela status.csv terá o formato descrito na tabela abaixo:

| Nome da coluna | Descrição | Tipo |
|----------------|-------------------------|---------|
| statusid | Identificação do status | Integer |
| status | status | String |

Os dados de treinamento totalizam 1880 de instâncias que deverão ser divididos entre amostra de treinamento e amostra de teste para desenvolvimento e validação do modelo preditivo.

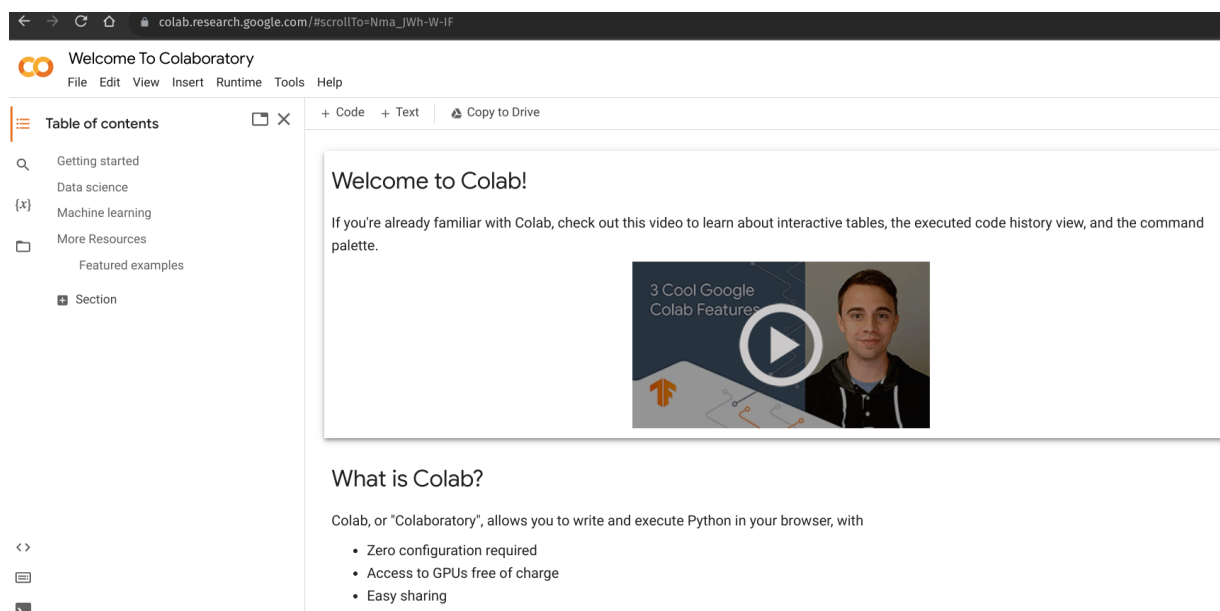
Foi utilizado para este trabalho os anos de 2018 a 2022 nos modelos de predição.

3. Processamento/Tratamento de Dados

Ferramentas utilizadas:

Para processamento, tratamento e análise de dados continuaremos a utilizar a linguagem Python, via Google Colab, disponível em https://colab.research.google.com/#scrollTo=Nma_JWh-W-IF (figura 3.1) para facilitar o acompanhamento e reprodução do que foi realizado.

Figura 3.1: Screenshot do Google colab



Fonte: Autor.

Após processamento em separado das bases da Fórmula 1 chegou o momento de unificar as bases.

Datasets – junção entre os arquivos races, results e drivers

O script para esta fase para base geral de amostra de teste e de treinamento é o seguinte:

Dataset de treinamento:

```
### Merge para o dataset de amostra de TREINAMENTO de 2018 até 2022
races_2018_2022_train = races[races['year']>=2018]
races_2018_2022_train = races_2018_2022_train.merge(results,
left_on='raceId', right_on='raceId', how='inner')
races_2018_2022_train = races_2018_2022_train.merge(drivers_raw_df,
left_on='driverId', right_on='driverId', how='inner')

###Amostra os 5 primeiros registros do dataset de treinamento
races_2018_2022_train.head()
```

Nessa unificação vamos:

1. Unir as bases no modo INNER JOIN, ou seja, vamos considerar toda e qualquer corrida, independentemente do registro se referir a apenas uma das bases.
2. Tratar valores ausentes após a junção na etapa posterior de processamento e tratamento de dados.
3. Tratar valores duplicados após a junção na etapa posterior de processamento e tratamento de dados.

Depois de realizado o processo de importação dos dados, inclusive utilizando Join, obteve-se os datasets listados abaixo:

| Nome do Dataset | Descrição |
|-----------------------|---|
| races_2018_2022_train | Conjunto de dados de treinamento na temporada no período do ano de 2018 ao ano de 2022. |

O dataset de treinamento consta de 1880 registros e o de teste com 260 registros.

Tratamento de dados ausentes:

Felizmente os dados dos datasets importados já vem com uma boa estrutura, praticamente pronto para iniciar a análise exploratória. Faremos apenas um tratamento objetivo, que é a remoção de colunas desnecessária no contexto.

Alguns dos campos apresentam alto índice de nulidade e outros são desnecessários e em análise inicial verificamos que devem ser excluídos das bases:

- Colunas: 'constructorId', 'grid', 'position', 'positionText', 'positionOrder', 'laps', 'time_x', 'milliseconds', 'fastestLap', 'fastestLapTime', 'fastestLapSpeed', 'statusId', 'url_x', 'fp1_date', 'fp1_time', 'fp2_date', 'fp2_time', 'fp3_date', 'fp3_time', 'quali_date', 'quali_time', 'sprint_date', 'sprint_time', 'driverRef', 'statusId', 'dob', 'constructorId', 'milliseconds', 'fastestLap', 'fastestLapTime', 'fastestLapSpeed', 'url_y', 'time_y', 'surname', 'nationality', 'name', 'date', 'resultId', 'number_x', 'number_y'

As colunas acima serão removidas, até mesmo porque foram inseridas recentemente para incorporar corridas sprints, ou seja, ainda tem dados somente do ano de 2021 e 2022, que por sua vez não irá agregar no objetivo final.

```

✓ [33] 1 races_2018_2022_train.dtypes
0s
    raceId      int64
    year        int64
    round       int64
    circuitId   int64
    name        object
    date        object
    time_x      object
    url_x       object
    fp1_date    object
    fp1_time    object
    fp2_date    object
    fp2_time    object
    fp3_date    object
    fp3_time    object
    quali_date  object
    quali_time  object
    sprint_date object
    sprint_time object
    resultId    int64
    driverId    int64
    constructorId int64
    number_x    object
    grid        int64
    position    object
    positionText object
    positionOrder int64
    points      float64
    laps        int64
    time_y      object
    milliseconds object
    fastestLap  object
    rank        object
    fastestLapTime object
    fastestLapSpeed object
    statusId    int64
    driverRef   object
    number_y    object
    code        object
    forename    object
    surname     object
    dob         object
    nationality  object
    url_y       object
    dtype: object

```

Figura 3.2: Screenshot Listando os objetos antes de exclusão das colunas.

###Removendo colunas do dataset de treinamento

```

races_2018_2022_train=races_2018_2022_train.drop(columns=['construc
torId', 'grid', 'position', 'positionText', 'positionOrder', 'laps', 'time_x',
'milliseconds', 'fastestLap', 'fastestLapTime', 'fastestLapSpeed',
'statusId','url_x', 'fp1_date', 'fp1_time','fp2_date','fp2_time', 'fp3_date',
'fp3_time', 'quali_date', 'quali_time', 'sprint_date', 'sprint_time',
'driverRef', 'statusId', 'dob', 'constructorId', 'milliseconds', 'fastestLap',
'fastestLapTime', 'fastestLapSpeed', 'url_y', 'time_y', 'surname',
'nationality', 'name', 'date', 'resultId', 'number_x','number_y'])

```



```

1 #Objetos no dataset
2 train.dtypes

raceId      int64
year        int64
round       int64
circuitId   int64
driverId    int64
points      float64
rank        object
code        object
forename    object
dtype: object

```

Figura 3.3: Screenshot Listando os objetos após a exclusão das colunas.

O dataset de treinamento consta de 1880 registros e 9 colunas.

Novo Dataset tratado com as colunas removidas e pronto para análise exploratória.

| Nome do Dataset | Descrição |
|-----------------|---|
| train | Conjunto de dados de treinamento na temporada no período do ano de 2018 ao ano de 2022. |


```
1 #Amostragem dos registros no dataset
2 train.head()
```

| | raceId | year | round | circuitId | driverId | points | rank | code | forename |
|---|--------|------|-------|-----------|----------|--------|------|------|-----------|
| 0 | 989 | 2018 | 1 | 1 | 20 | 25.0 | 4 | VET | Sebastian |
| 1 | 990 | 2018 | 2 | 3 | 20 | 25.0 | 7 | VET | Sebastian |
| 2 | 991 | 2018 | 3 | 17 | 20 | 4.0 | 9 | VET | Sebastian |
| 3 | 992 | 2018 | 4 | 73 | 20 | 12.0 | 4 | VET | Sebastian |
| 4 | 993 | 2018 | 5 | 4 | 20 | 12.0 | 2 | VET | Sebastian |

Novos Datasets tratados com as colunas removidas e pronto para Análise Exploratória em 3 novos datasets pelos 3 melhores pilotos de fórmula 1 no intervalo temporal de 2018 a 2022.

Datasets dos 3 melhores pilotos entre 2018 e 2022 da fórmula 1

```
1 #dataset train HAM
2 train_HAM = train[(train['code']=='HAM')]
3 #dataset train VER
4 train_VER = train[(train['code']=='VER')]
5 #dataset train BOT
6 train_BOT = train[(train['code']=='BOT')]
```

Tratamento de dados duplicados ou não relevantes para o contexto

Não há registros duplicados, já que cada registro corresponde a uma situação diferente para análise, ou seja uma corrida de fórmula 1 diferente por temporada.

Por fim, para esta fase estes passos de limpeza dos dados reduzem o número de elementos únicos de linguagem, e além de oferecer um ganho computacional no momento do processamento, são a chave para interpretação de alguns modelos de aprendizado de máquinas. (GENTZKOW, KELLY, e TADD, 2019).

4. Análise e Exploração dos Dados

A Análise Exploratória utilizando a linguagem Python irá demonstrar alguns comandos para facilitar o entendimento dos dados.

Essa parte é muito importante para entendermos e analisarmos a base de dados para que possamos observar atentamente os insights, desvios e situações que somente com esta análise será possível identificar e por consequência avançarmos no projeto de ciência de dados.

#Escolha do período a ser utilizado nos testes.

#OBS. As datas abaixo somente serão utilizadas futuramente nos modelos de previsão para testar os algoritmos

`test_start_date = '2022-01-01'`

`test_end_date = '2022-12-31'`

Escolha dos pilotos a serem analisados

`pilotos = ['HAM', 'VER', 'BOT']`

Datasets dos 3 melhores pilotos entre 2018 e 2022 da fórmula 1

```
✓ [28] 1 #dataset train HAM
0s      2 train_HAM = train[(train['code']=='HAM')]
        3 #dataset train VER
        4 train_VER = train[(train['code']=='VER')]
        5 #dataset train BOT
        6 train_BOT = train[(train['code']=='BOT')]
```

Depois de importar a base é importante visualizar esses dados para que consigamos começar a entender como eles estão distribuídos.

Dessa forma vamos ter uma noção de como os mesmos estão organizados e se esse padrão se mantém em todo o arquivo.

Em seguida, vamos utilizar funções para confirmar qual é o primeiro e o último ano da nossa base de dados. Assim podemos garantir qual é o nosso período total na base com todos os registros e também nas bases por piloto.

Confirmando o Período de Análise

✓
0s

```
1 #Ano inicio dos trabalhos
2 ano_inicio = min(train.year)
3 print(ano_inicio)
```

2018

```
1 #Ano Fim dos trabalhos
2 ano_fim = max(train.year)
3 print(ano_fim)
```

2022

Agora vamos utilizar um comando muito importante que vai dar as informações gerais da nossa base de dados.

Descobrimos os valores nulos e tipo de dados

✓
s

```
[28] 1 train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1880 entries, 0 to 1879
Data columns (total 9 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   raceId      1880 non-null   int64
1   year        1880 non-null   int64
2   round       1880 non-null   int64
3   circuitId   1880 non-null   int64
4   driverId    1880 non-null   int64
5   points      1880 non-null   float64
6   rank        1880 non-null   object
7   code        1880 non-null   object
8   forename    1880 non-null   object
dtypes: float64(1), int64(5), object(3)
memory usage: 146.9+ KB
```

Aqui você consegue visualizar as colunas, a quantidade de valores não nulos e o tipo de cada uma dessas informações.

Podemos utilizar o comando `isnull().sum()` para obter essas mesmas informações, mas de uma forma um pouco diferente.

```
1 #verificando a existência de valores nulos
2 train.isnull().sum()
```

| | |
|--------------|---|
| raceId | 0 |
| year | 0 |
| round | 0 |
| circuitId | 0 |
| driverId | 0 |
| points | 0 |
| rank | 0 |
| code | 0 |
| forename | 0 |
| dtype: int64 | |

Sabendo dessas informações, nós podemos entender melhor esses valores e analisar as informações estatísticas.

Em seguida nós temos as análises estatísticas, onde temos a contagem, média, desvio padrão, mínimo, máximo e os quartis do dataset principal.

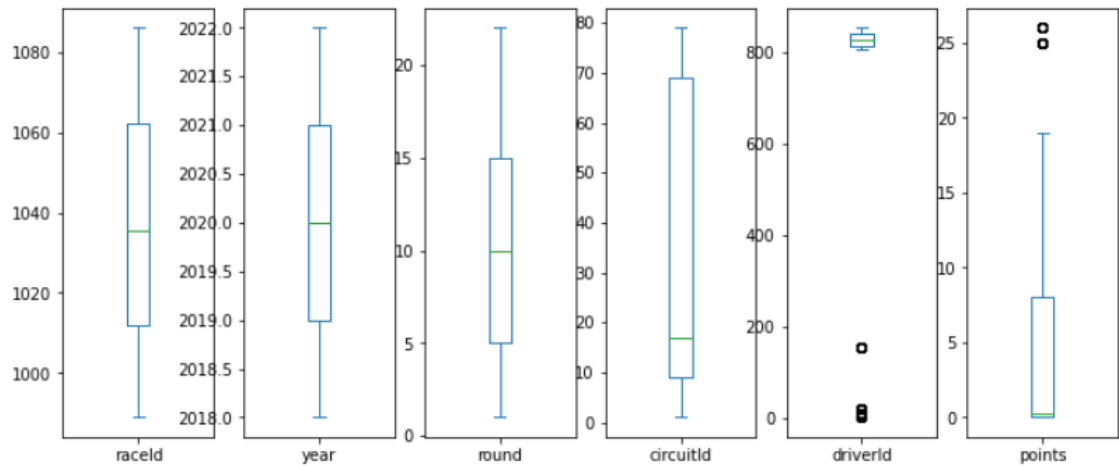
Analisando as informações estatísticas

```
1 train.describe()
```

| | raceId | year | round | circuitId | driverId | points |
|--------------|-------------|-------------|-------------|-------------|-------------|-------------|
| count | 1880.000000 | 1880.000000 | 1880.000000 | 1880.000000 | 1880.000000 | 1880.000000 |
| mean | 1036.808511 | 2019.840426 | 10.202128 | 29.308511 | 663.636702 | 5.058777 |
| std | 28.637789 | 1.371055 | 5.869940 | 27.289872 | 330.824574 | 7.212994 |
| min | 989.000000 | 2018.000000 | 1.000000 | 1.000000 | 1.000000 | 0.000000 |
| 25% | 1012.000000 | 2019.000000 | 5.000000 | 9.000000 | 815.000000 | 0.000000 |
| 50% | 1035.500000 | 2020.000000 | 10.000000 | 17.000000 | 830.000000 | 0.250000 |
| 75% | 1062.000000 | 2021.000000 | 15.000000 | 69.000000 | 844.000000 | 8.000000 |
| max | 1086.000000 | 2022.000000 | 22.000000 | 79.000000 | 855.000000 | 26.000000 |

Para entender melhor as informações nós podemos utilizar o boxplot.

```
[47] 1 #Entendendo melhor as informações
      2 train.plot(kind='box', figsize=(12,5),subplots=True);
```



Visualizando outliers só para entender o que está acontecendo com esses dados fora do padrão.

```
1 #0 que seria esses outliers?
2 train[train['points'] >=15]
```

| | raceId | year | round | circuitId | driverId | points | rank | code | forename |
|------|--------|------|-------|-----------|----------|--------|------|------|-----------|
| 0 | 989 | 2018 | 1 | 1 | 20 | 25.0 | 4 | VET | Sebastian |
| 1 | 990 | 2018 | 2 | 3 | 20 | 25.0 | 7 | VET | Sebastian |
| 5 | 994 | 2018 | 6 | 6 | 20 | 18.0 | 7 | VET | Sebastian |
| 6 | 995 | 2018 | 7 | 7 | 20 | 25.0 | 2 | VET | Sebastian |
| 8 | 997 | 2018 | 9 | 70 | 20 | 15.0 | 2 | VET | Sebastian |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 1686 | 1076 | 2022 | 3 | 1 | 847 | 15.0 | 4 | RUS | George |
| 1689 | 1079 | 2022 | 6 | 4 | 847 | 15.0 | 3 | RUS | George |
| 1691 | 1081 | 2022 | 8 | 73 | 847 | 15.0 | 4 | RUS | George |
| 1695 | 1085 | 2022 | 12 | 34 | 847 | 15.0 | 3 | RUS | George |
| 1696 | 1086 | 2022 | 13 | 11 | 847 | 15.0 | 9 | RUS | George |

279 rows × 9 columns

Analisando outliers

Aqui temos somente um piloto no ano de 2018 (Sebastian Vettel) que ficou no top 5 por muito mais tempo que os outros pilotos da nossa base, ou seja dominando a F1 em 2018.

Com isso você já começa a entender um pouco mais não só o outlier, mas os seus dados no geral.

Entendemos também que no ano de 2022 o piloto (George Russell), foi muito constante na temporada.

```

1 #Continuando com soma de alguns valores
2 train.code.value_counts()

```

```

RIC      94
VER      94
BOT      94
GAS      94
SAI      94
LEC      94
HAM      93
STR      93
VET      92
PER      92
RAI      79
RUS      73
NOR      73
OCO      73
MAG      72
GIO      60
GRO      57
ALO      56
LAT      52
ALB      51
HUL      47
KVY      38
TSU      35
MSC      35
KUB      23
MAZ      22
ERI      21
SIR      21
VAN      21
HAR      21
ZHO      13
FIT       2
AIT       1
Name: code, dtype: int64

```

Continuando a análise dá só uma olhada para entender melhor os dados utilizados na contagem das corridas por pilotos. Veja que Ricciardo é uma surpresa ter participado de todas as corridas desde 2018 até 2022.

Então com essa análise acabamos descobrindo informações novas, pois pilotos mais constantes como Ricciardo, mostrando constância em participação em corridas, mas você pode nunca ter ouvido falar dessa performance e o mesmo apareceu diversas vezes no Top 5.

Algumas informações para análise da base com agrupamento por coluna e por nome do Piloto.

É muito importante não só analisar, mas conhecer as informações que estamos trabalhando, pois isso vai facilitar muito a execução do nosso projeto.

```
1 #Total de pontos por Piloto no período de 2018 a 2022
2 func = lambda x: x.points.sum()/x.forename.nunique()
3 data = train[train['forename'].isin(train.forename)].groupby('forename') \
4 .apply(func).sort_values(ascending=False).reset_index(name = 'Total Geral')
5 data.head(10)
```

| | forename | Total Geral |
|---|-----------|-------------|
| 0 | Lewis | 1698.5 |
| 1 | Max | 1371.5 |
| 2 | Valtteri | 1059.0 |
| 3 | Charles | 724.0 |
| 4 | Sebastian | 652.0 |
| 5 | Sergio | 592.0 |
| 6 | Carlos | 562.5 |
| 7 | Daniel | 473.0 |
| 8 | Lando | 378.0 |
| 9 | Pierre | 325.0 |



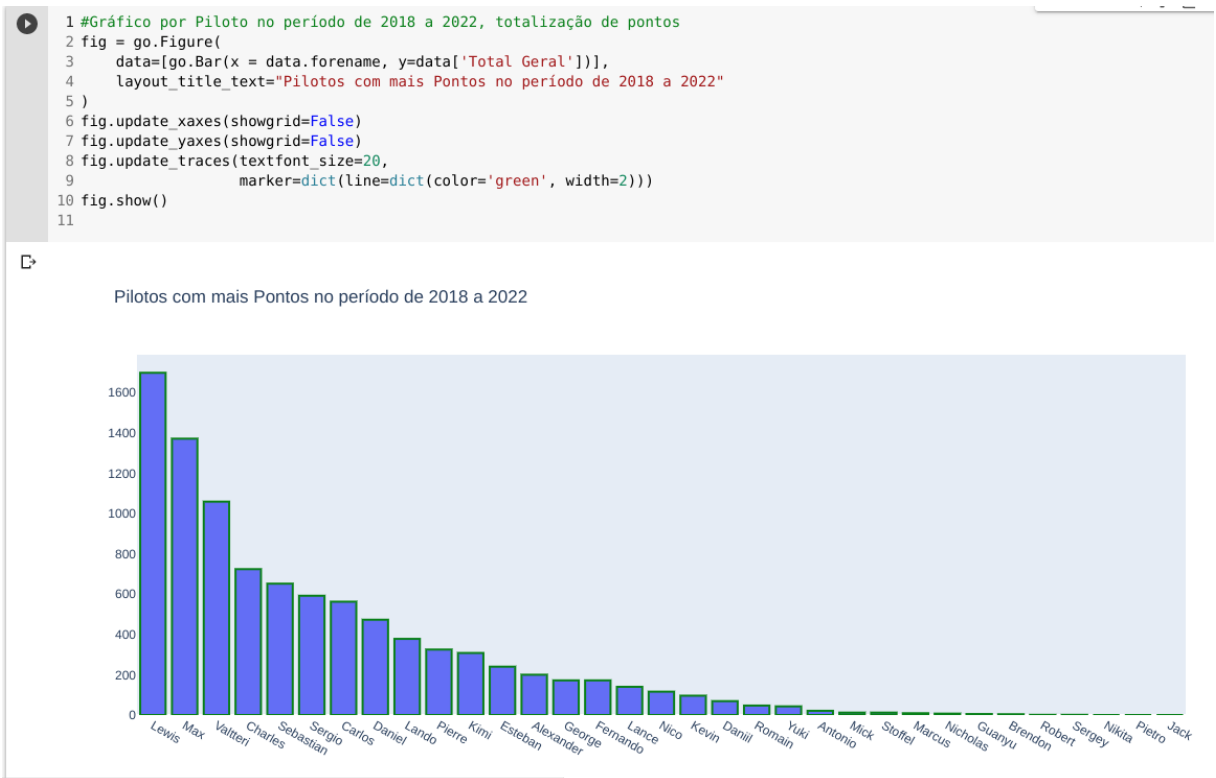


Figura 1: Gráfico com agrupamento por piloto dos totais de pontos das temporadas de 2018 a 2022.

4.1 - Análise e Exploração dos Dados - Piloto Hamilton

Após a coleta dos dados, foi realizada uma descrição estatística do dataset conforme mostra a figura abaixo:

```

1 train_HAM.describe()

```

| | raceId | year | round | circuitId | driverId | points |
|-------|-------------|-------------|-----------|-----------|----------|-----------|
| count | 93.000000 | 93.000000 | 93.000000 | 93.000000 | 93.0 | 93.000000 |
| mean | 1036.709677 | 2019.838710 | 10.139785 | 29.591398 | 1.0 | 18.263441 |
| std | 28.923655 | 1.385408 | 5.900611 | 27.439355 | 0.0 | 7.639141 |
| min | 989.000000 | 2018.000000 | 1.000000 | 1.000000 | 1.0 | 0.000000 |
| 25% | 1012.000000 | 2019.000000 | 5.000000 | 9.000000 | 1.0 | 15.000000 |
| 50% | 1035.000000 | 2020.000000 | 10.000000 | 17.000000 | 1.0 | 19.000000 |
| 75% | 1062.000000 | 2021.000000 | 15.000000 | 69.000000 | 1.0 | 25.000000 |
| max | 1086.000000 | 2022.000000 | 22.000000 | 79.000000 | 1.0 | 26.000000 |

Figura 2: Descrição estatística piloto Hamilton

Em seguida foi verificado que não há valor nulo no dataset HAM

```

1 #Valores Nulos dataset Hamilton
2 train_HAM.isnull().sum()

```

| | |
|-----------|-------|
| raceId | 0 |
| year | 0 |
| round | 0 |
| circuitId | 0 |
| driverId | 0 |
| points | 0 |
| rank | 0 |
| code | 0 |
| forename | 0 |
| dtype: | int64 |

Figura 3: Valores Nulos

Plotando-se gráfico de série temporal, obtém-se o resultado apresentado:

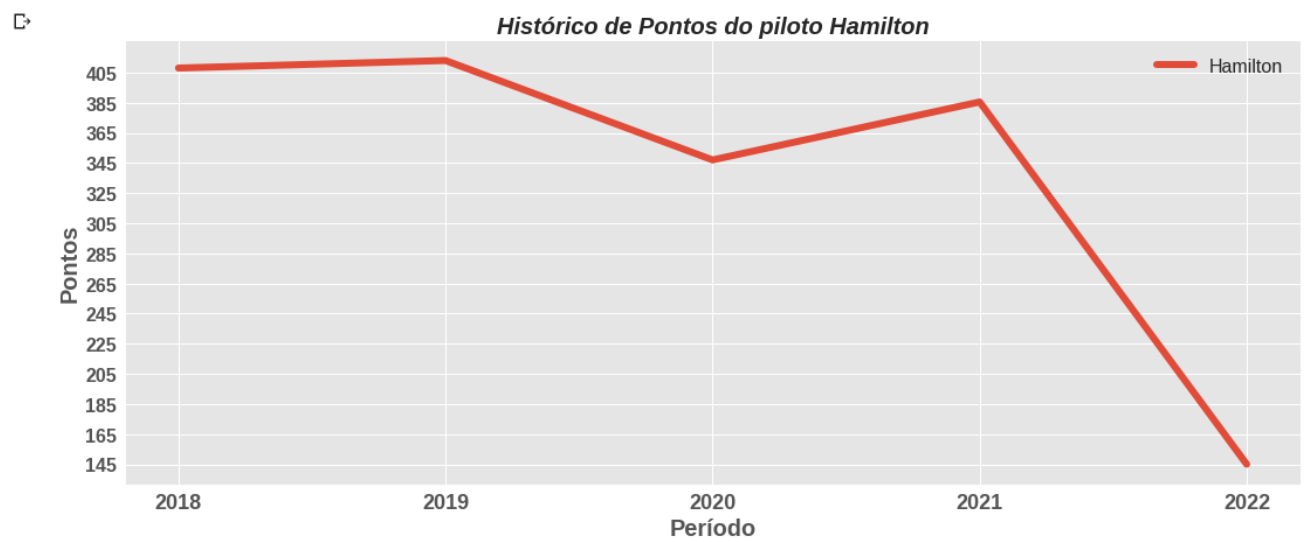


Figura 4: Gráfico com dados do piloto Hamilton desde 2018 a 2022

Observar-se que o gráfico apresenta picos, representando uma queda brusca da temporada 2021 para o ano 2022.

4.2 - Análise e Exploração dos Dados - Piloto Verstappen

Após a coleta dos dados, foi realizada uma descrição estatística do dataset conforme mostra a figura abaixo:

1 #Estatísticas Dataset Verstappen
2 train_VER.describe()

| | raceId | year | round | circuitId | driverId | points |
|-------|---------|---------|-------|-----------|----------|--------|
| count | 94.00 | 94.00 | 94.0 | 94.00 | 94.0 | 94.00 |
| mean | 1036.81 | 2019.84 | 10.2 | 29.31 | 830.0 | 14.59 |
| std | 28.78 | 1.38 | 5.9 | 27.43 | 0.0 | 8.83 |
| min | 989.00 | 2018.00 | 1.0 | 1.00 | 830.0 | 0.00 |
| 25% | 1012.25 | 2019.00 | 5.0 | 9.00 | 830.0 | 10.00 |
| 50% | 1035.50 | 2020.00 | 10.0 | 17.00 | 830.0 | 15.00 |
| 75% | 1061.75 | 2021.00 | 15.0 | 69.00 | 830.0 | 23.50 |
| max | 1086.00 | 2022.00 | 22.0 | 79.00 | 830.0 | 26.00 |

Figura 5: Descrição estatística piloto Verstappen

Em seguida foi verificado que não há valor nulo no dataset VER.

1 #Valores Nulos dataset Verstappen
2 train_VER.isnull().sum()

| | |
|--------------|---|
| raceId | 0 |
| year | 0 |
| round | 0 |
| circuitId | 0 |
| driverId | 0 |
| points | 0 |
| rank | 0 |
| code | 0 |
| forename | 0 |
| dtype: int64 | |

Figura 6: Valores Nulos

Plotando-se gráfico de série temporal, obtém-se o resultado apresentado:

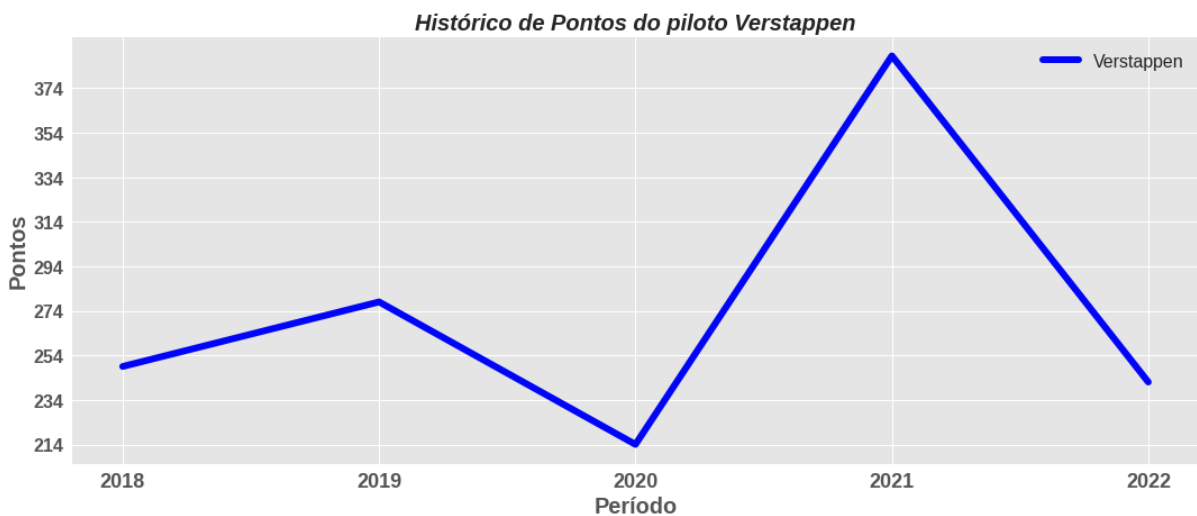


Figura 7: Gráfico com dados do piloto Verstappen desde 2018 a 2022

Observa-se que o gráfico apresenta picos, representando uma piora na performance da temporada 2021 para 2022.

4.3 - Análise e Exploração dos Dados - Piloto Bottas

Após a coleta dos dados, foi realizada uma descrição estatística do dataset conforme mostra a figura abaixo:

```
1 #Estatísticas Dataset Bottas
2 train_BOT.describe()
```

| | raceId | year | round | circuitId | driverId | points |
|-------|---------|---------|-------|-----------|----------|--------|
| count | 94.00 | 94.00 | 94.0 | 94.00 | 94.0 | 94.00 |
| mean | 1036.81 | 2019.84 | 10.2 | 29.31 | 822.0 | 11.27 |
| std | 28.78 | 1.38 | 5.9 | 27.43 | 0.0 | 7.80 |
| min | 989.00 | 2018.00 | 1.0 | 1.00 | 822.0 | 0.00 |
| 25% | 1012.25 | 2019.00 | 5.0 | 9.00 | 822.0 | 4.00 |
| 50% | 1035.50 | 2020.00 | 10.0 | 17.00 | 822.0 | 12.00 |
| 75% | 1061.75 | 2021.00 | 15.0 | 69.00 | 822.0 | 18.00 |
| max | 1086.00 | 2022.00 | 22.0 | 79.00 | 822.0 | 26.00 |

Figura 8: Descrição estatística piloto Bottas

Em seguida foi verificado que não há valor nulo no dataset BOT

```

1 #Valores Nulos dataset Bottas
2 train_BOT.isnull().sum()

raceId      0
year        0
round       0
circuitId   0
driverId    0
points      0
rank        0
code        0
forename    0
dtype: int64

```

Figura 9: Valores Nulos

Plotando-se gráfico de série temporal, obtém-se o resultado apresentado:

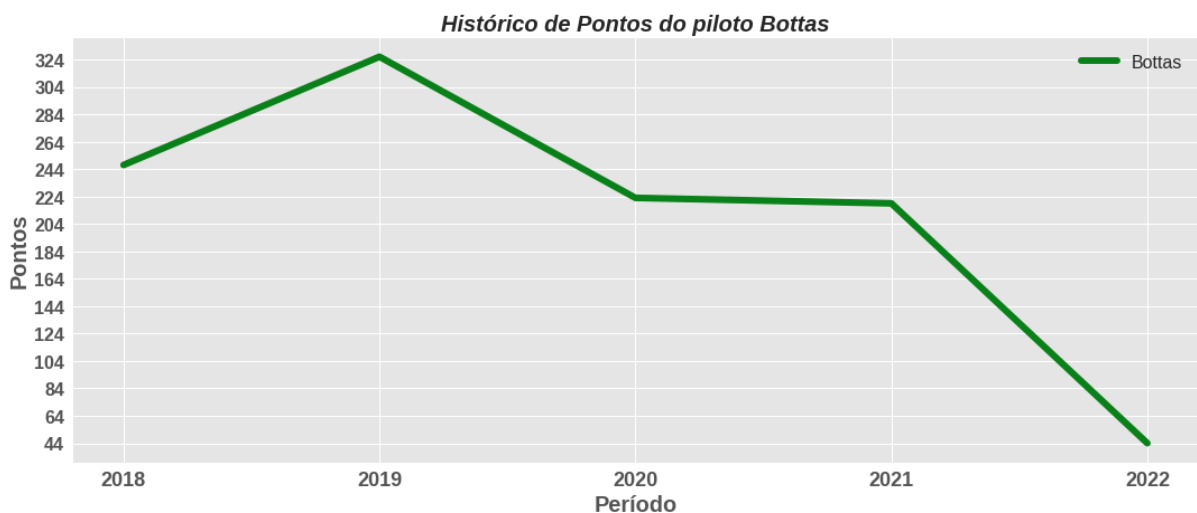


Figura 10: Gráfico com dados do piloto Bottas desde 2018 a 2022

Observa-se que o gráfico apresenta picos, representando uma piora na performance da temporada 2018 para 2022.

4.4 - Análise e Exploração dos Dados - em conjunto.

A figura 11 apresenta um gráfico comparativo de desempenho dos três melhores pilotos entre 2018 e 2022 de forma conjunta. Para isso, foi utilizado os pontos agrupados por ano (points) já que este considera ponto por corrida mas já agrupado no dataset contendo todos os registros.

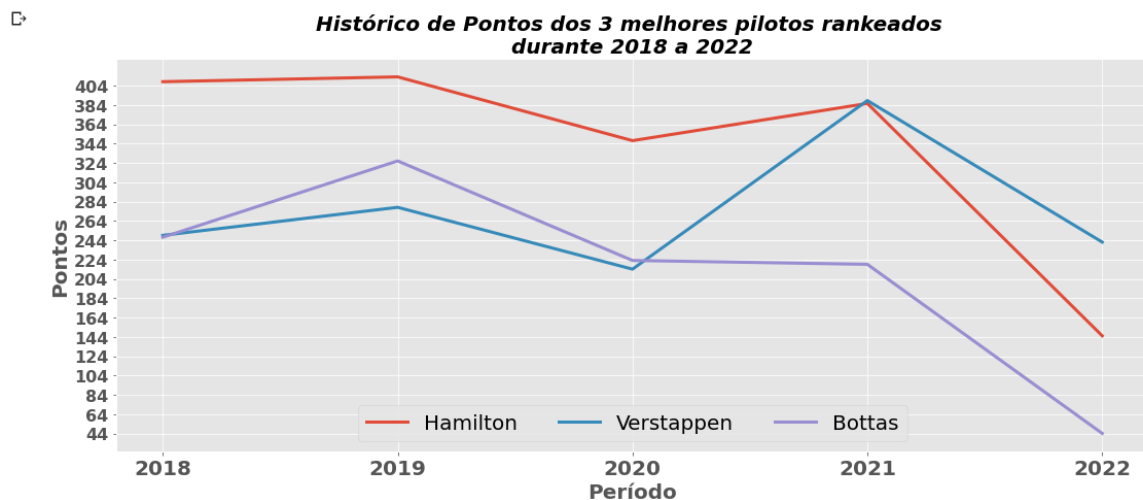


Figura 11: Gráfico comparativo com os 3 melhores pilotos entre 2018 e 2022

Claramente podemos perceber que o piloto Hamilton, foi o que teve a maior pontuação no período elencado e que teve uma queda brusca de 2021 para 2022. Além de que o terceiro piloto Bottas também veio em uma curva descendente desde então.

Utilizamos também a correlação de Pearson para verificar alguma relação entre as informações no Dataset principal.

```
1 train.corr()
```

| | raceId | year | round | circuitId | driverId | points |
|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| raceId | 1.000000 | 0.979201 | 0.051772 | 0.147711 | 0.082044 | -0.001576 |
| year | 0.979201 | 1.000000 | -0.138826 | 0.135590 | 0.084460 | -0.001231 |
| round | 0.051772 | -0.138826 | 1.000000 | 0.119678 | -0.017182 | -0.001601 |
| circuitId | 0.147711 | 0.135590 | 0.119678 | 1.000000 | 0.007729 | 0.002520 |
| driverId | 0.082044 | 0.084460 | -0.017182 | 0.007729 | 1.000000 | -0.172977 |
| points | -0.001576 | -0.001231 | -0.001601 | 0.002520 | -0.172977 | 1.000000 |

A figura 12 mostra um mapa de correlação, com ele podemos verificar que os pontos estão fortemente correlacionados entre si.

```
1 #Função para criação do mapa de correlação de pearson entre os resultados dos pilotos
2 sns.heatmap(train.corr(),annot = True, cmap='Reds');
```

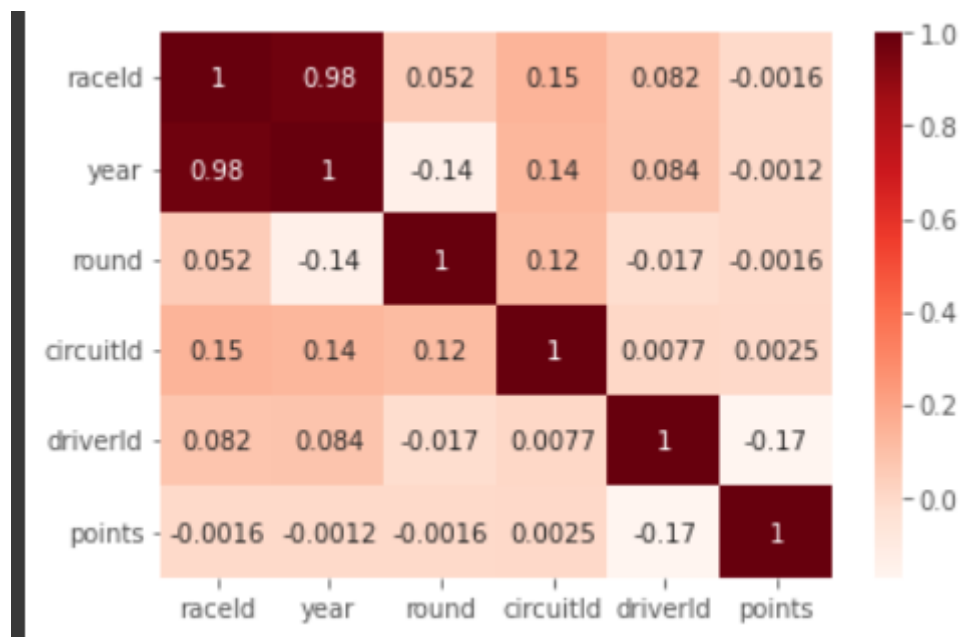


Figura 12: Mapa de correlação do dataset com informações entre 2018 e 2022

Podemos notar uma correlação mais forte entre ano (year) e corridas (raceId) e também entre pilotos (driverId) e corridas (raceId).

5. Criação de Modelos de Machine Learning

Nesta seção, são apresentados os modelos preditivos desenvolvidos em Python para o dataset do conjunto dos 3 (três) melhores pilotos entre 2018 e 2022 para prever o ano de 2023, utilizando a biblioteca Facebook Prophet, AUTO-ARIMA e RNN (Recurrent Neural Network) utilizando a arquitetura LSTM (Long Short Term Memory).

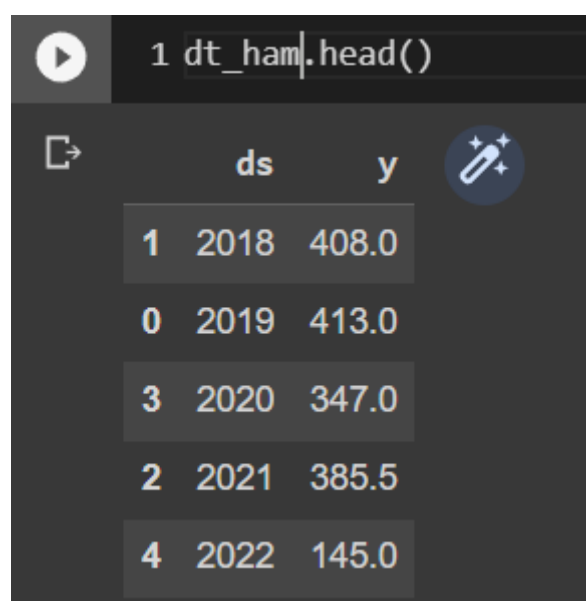
5.1. Modelos de Machine Learning com Facebook Prophet

Facebook Prophet: É uma biblioteca criada pelo Facebook com o objetivo de ser utilizada em sua própria rede social.

O motivo era facilitar a criação de modelos sem necessariamente ter mais mão de obra especializada para isto. E com o intuito de modelar sazonalidades.

Neste modelo, os dados de treinamento serão reduzidos em apenas duas colunas para atender a especificação da biblioteca.

A coluna “ds” representa o ano da temporada e “y” o total de pontos do piloto Hamilton nas temporadas.



The image shows a Jupyter Notebook interface. At the top, a code cell contains the command `1 dt_ham.head()`. Below it, the output is displayed as a table with two columns: 'ds' (date) and 'y' (total points). The table contains five rows of data, indexed from 0 to 4.

| | ds | y |
|---|------|-------|
| 1 | 2018 | 408.0 |
| 0 | 2019 | 413.0 |
| 3 | 2020 | 347.0 |
| 2 | 2021 | 385.5 |
| 4 | 2022 | 145.0 |

Figura 13: dados dataset piloto Hamilton

Segue o código utilizado para este modelo:

```
prophet_model = Prophet(changepoint_prior_scale=0.05,  
interval_width=0.95, daily_seasonality=False)  
prophet_model.fit(train_set)  
  
y_pred = prophet_model.make_future_dataframe(periods=365, freq='D')  
y_pred = prophet_model.predict(test_set)
```

Descrição:

Changepoint_prior_scale: Modula a flexibilidade da seleção automática de ponto de mudança.

Interval_width: Intervalos de incerteza previstos para a previsão.

Daily_seasonality: Ajusta a sazonalidade diária

5.2. Modelos de Machine Learning com AUTO-ARIMA

AUTO-ARIMA: Foi feito um breve estudo da série temporal, considerando sazonalidade, estacionalidade, além de correlação.

Inicialmente realizamos a decomposição, obtendo componentes conforme a figura 14:



Figura 14: Decomposição piloto Hamilton

Plotando a autocorrelação:

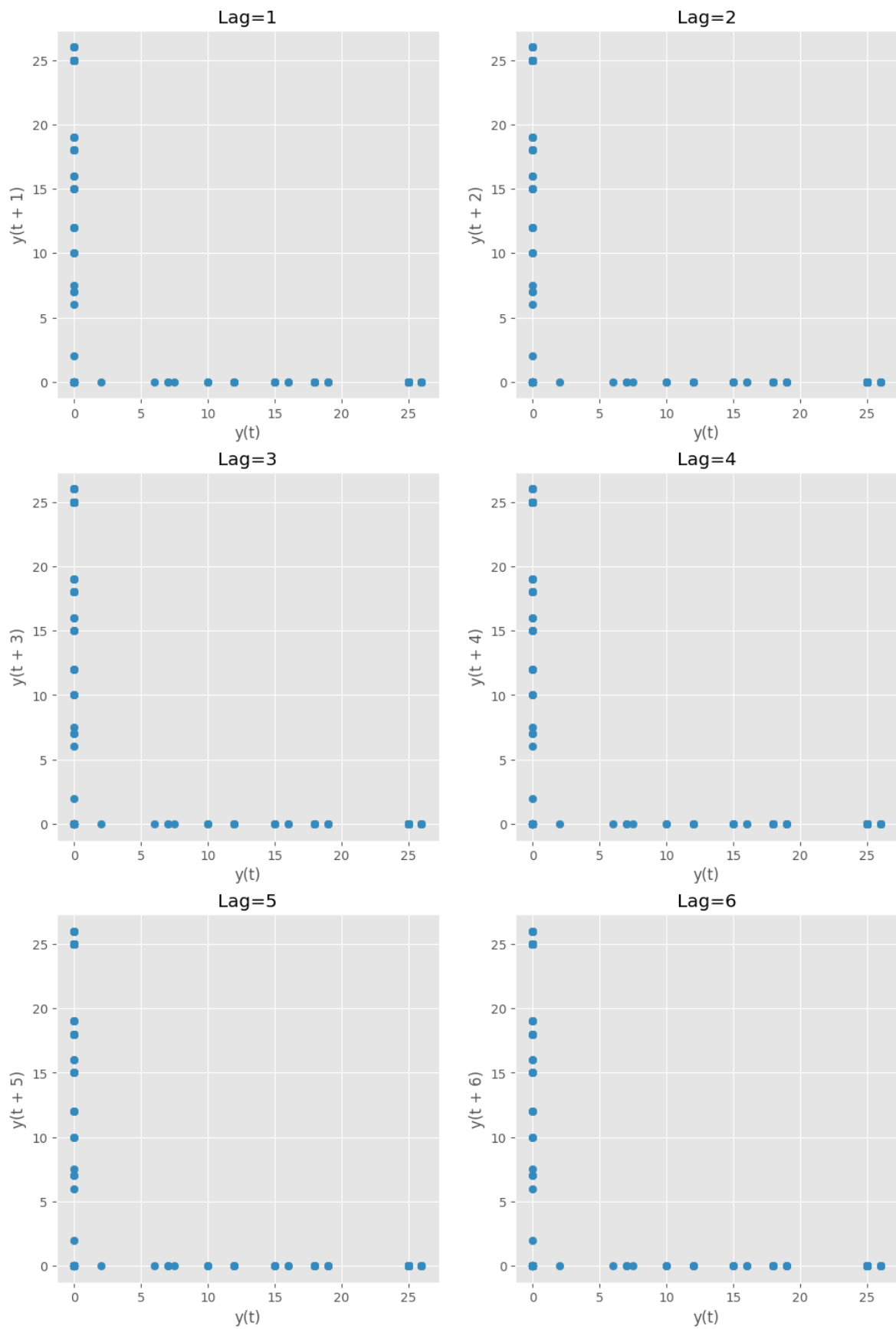


Figura 15: Autocorrelação da série temporal do piloto Hamilton

A plotagem da autocorrelação parece linear nas bordas.

A estacionalidade foi verificada com o teste Dickey-Fuller, obtendo-se o seguinte resultado:

#Verificando a estacionalidade através da função abaixo:

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

```
Resultado do Teste Dickey-Fuller:
Teste -5.044113
Valor p 0.000018
Nº de lags 20.000000
Nº de observações 1338.000000
Valor Crítico (1%) -3.435247
Valor Crítico (5%) -2.863703
Valor Crítico (10%) -2.567921
dtype: float64
```

Já era esperado pela análise dos gráficos anteriores e como o valor p é muito menor do que 0.05, concluindo que a série é estacionária.

Plotando os gráficos ACF (Autocorrelation) e PACF (Partial Autocorrelation).

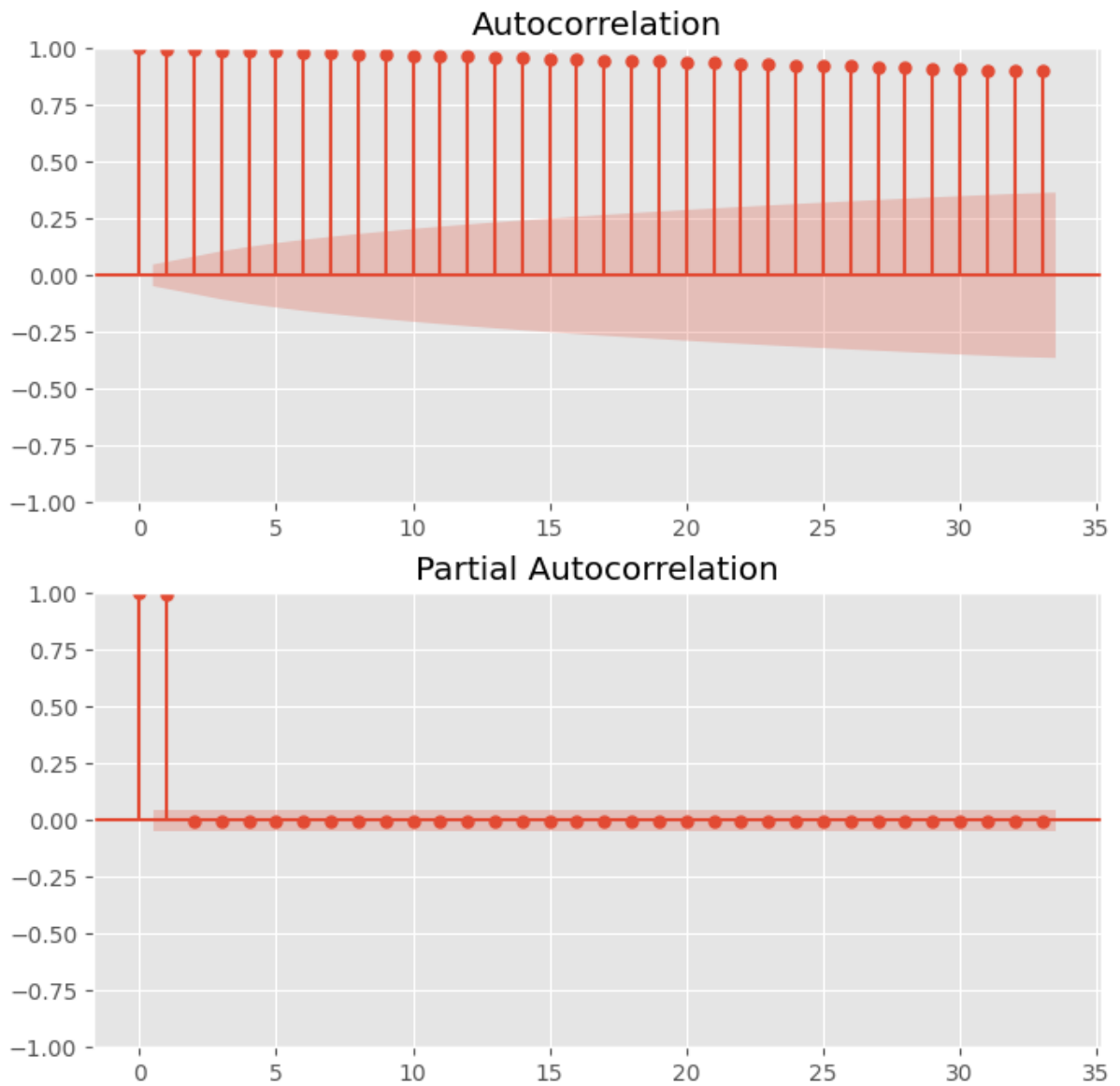


Figura 16: Gráficos ACF e PACF

Código onde foi realizado a previsão

```
#Definição do modelo e realizando a previsão
model = pm.auto_arima(train_data_ham['points'], start_p=1, start_q=1,
                      #test='adf',          # usa o adftest para encontrar
o melhor valor 'd'

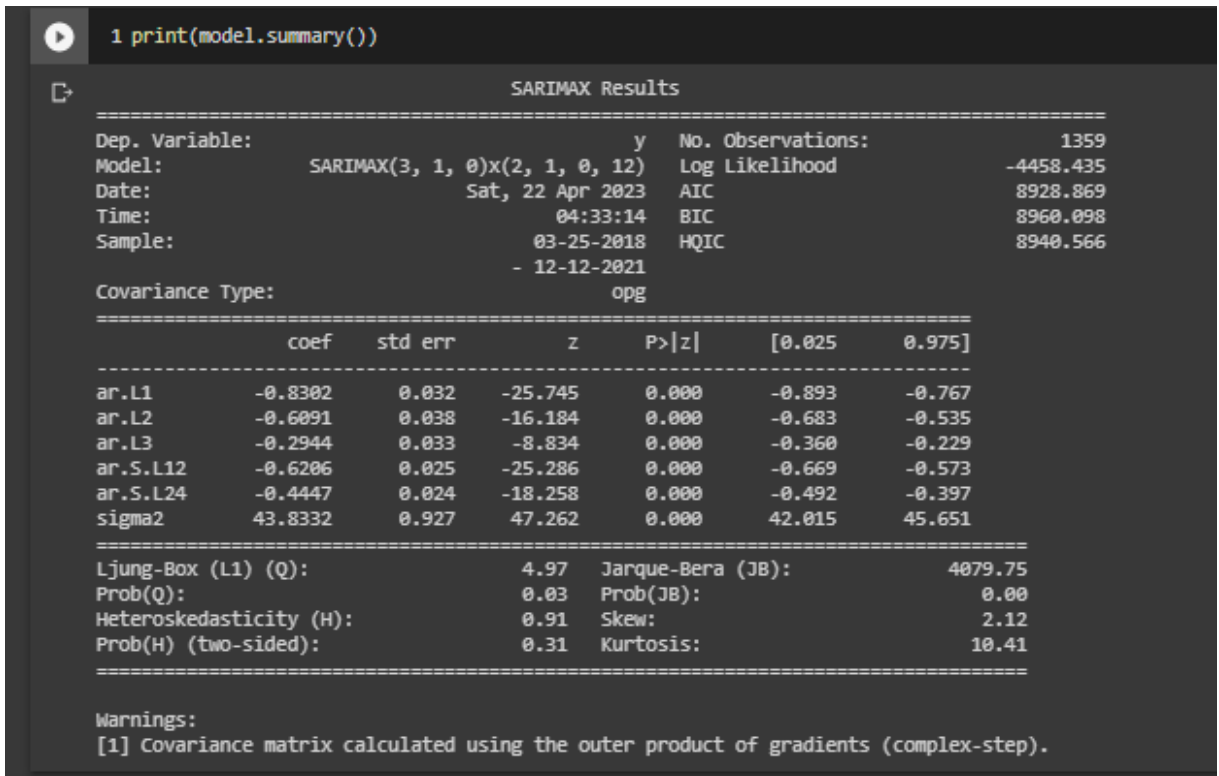
                      max_p=3, max_q=3, # Máximo 'p' e 'q'
                      m=12,             # Frequência da série
                      d=1,
                      stationary = False, #Estacionaridade
                      seasonal=True, #Sazonalidade
                      start_P=0,
                      D=1,
                      trace=True, #Se TRUE, a lista de modelos ARIMA
considerados será reportada.

                      error_action='ignore',
                      suppress_warnings=True,
                      stepwise=True) #Se TRUE, fará a seleção gradual
(mais rápido). Caso contrário, ele pesquisará todos os modelos.

#Treinamento do modelo
model.fit(train_data_ham['points'])
```

Imprimindo Summary do modelo

```
1 print(model.summary())
```



SARIMAX Results

| Dep. Variable: | | y | No. Observations: | 1359 |
|------------------|--------------------------------|-----|-------------------|-----------|
| Model: | SARIMAX(3, 1, 0)x(2, 1, 0, 12) | | Log Likelihood | -4458.435 |
| Date: | Sat, 22 Apr 2023 | | AIC | 8928.869 |
| Time: | 04:33:14 | | BIC | 8960.098 |
| Sample: | 03-25-2018 | | HQIC | 8940.566 |
| | - 12-12-2021 | | | |
| Covariance Type: | | opg | | |

| | coef | std err | z | P> z | [0.025 | 0.975] |
|----------|---------|---------|---------|-------|--------|--------|
| ar.L1 | -0.8302 | 0.032 | -25.745 | 0.000 | -0.893 | -0.767 |
| ar.L2 | -0.6091 | 0.038 | -16.184 | 0.000 | -0.683 | -0.535 |
| ar.L3 | -0.2944 | 0.033 | -8.834 | 0.000 | -0.360 | -0.229 |
| ar.S.L12 | -0.6206 | 0.025 | -25.286 | 0.000 | -0.669 | -0.573 |
| ar.S.L24 | -0.4447 | 0.024 | -18.258 | 0.000 | -0.492 | -0.397 |
| sigma2 | 43.8332 | 0.927 | 47.262 | 0.000 | 42.015 | 45.651 |

| | | | |
|-------------------------|------|-------------------|---------|
| Ljung-Box (L1) (Q): | 4.97 | Jarque-Bera (JB): | 4079.75 |
| Prob(Q): | 0.03 | Prob(JB): | 0.00 |
| Heteroskedasticity (H): | 0.91 | Skew: | 2.12 |
| Prob(H) (two-sided): | 0.31 | Kurtosis: | 10.41 |

Warnings:

[1] Covariance matrix calculated using the outer product of gradients (complex-step).

Figura 17: Summary com SARIMAX

5.3. Modelos de Machine Learning com RNN (LSTM)

RNN (LSTM - Long Short Term Memory): Abaixo segue o código desenvolvido por etapas:

```
#Criando da LSTM utilizando a biblioteca Keras
from keras.models import Sequential
from keras.layers import LSTM, Dropout, Dense

# Inicialização da RNN
model_rnn = Sequential()

# Adiciona a primeira camada LSTM com o Dropout
model_rnn.add(LSTM(units = 128, return_sequences = True, input_shape =
(x_train_lstm.shape[1], 1)))
model_rnn.add(Dropout(0.3))
```

```
# Adiciona a segunda camada LSTM com o Dropout
model_rnn.add(LSTM(units = 64))
model_rnn.add(Dropout(0.3))

# Adiciona a camada de saída
model_rnn.add(Dense(units=1))

# Compila a RNN, neste caso utilizando o otimizador 'Adam'
model_rnn.compile(optimizer = 'adam', loss = 'mean_squared_error',
metrics=['mean_absolute_error'])

# Faz o treinamento da RNN utilizando o dataset de treinamento
model_rnn.fit(x_train_lstm, y_train_lstm, epochs = 100, batch_size =
32)

#Mostra o resumo do modelo
model_rnn.summary()
```


O início do código envolve a inicialização da RNN por meio da função `Sequential()`. O modelo que foi desenvolvido utiliza duas camadas LSTM com Dropout e uma camada densa de saída com ativação linear. O Dropout tem a finalidade de diminuir o overfitting, pois ele temporariamente desativa um percentual de neurônios artificiais, nesse caso, 30%.

A camada de entrada, que é a primeira camada, possui 128 neurônios do tipo LSTM, a segunda camada tem 64 e a última camada tem apenas uma saída linear. A quantidade de neurônios foi escolhida empírica e baseada em literatura científica.

Para a compilação, foi utilizado o otimizador Adam, que apresentou bons resultados. Ademais, o erro MSE (Mean Squared Error) foi usado como função de perda e a métrica escolhida foi o MAE (Mean Absolute Error).

Na etapa de treinamento utilizou-se o seguinte código:

```
model_rnn.fit(x_train_lstm, y_train_lstm, epochs = 100, batch_size = 32)
```

O modelo desenvolvido pode ser verificado na figura 18.

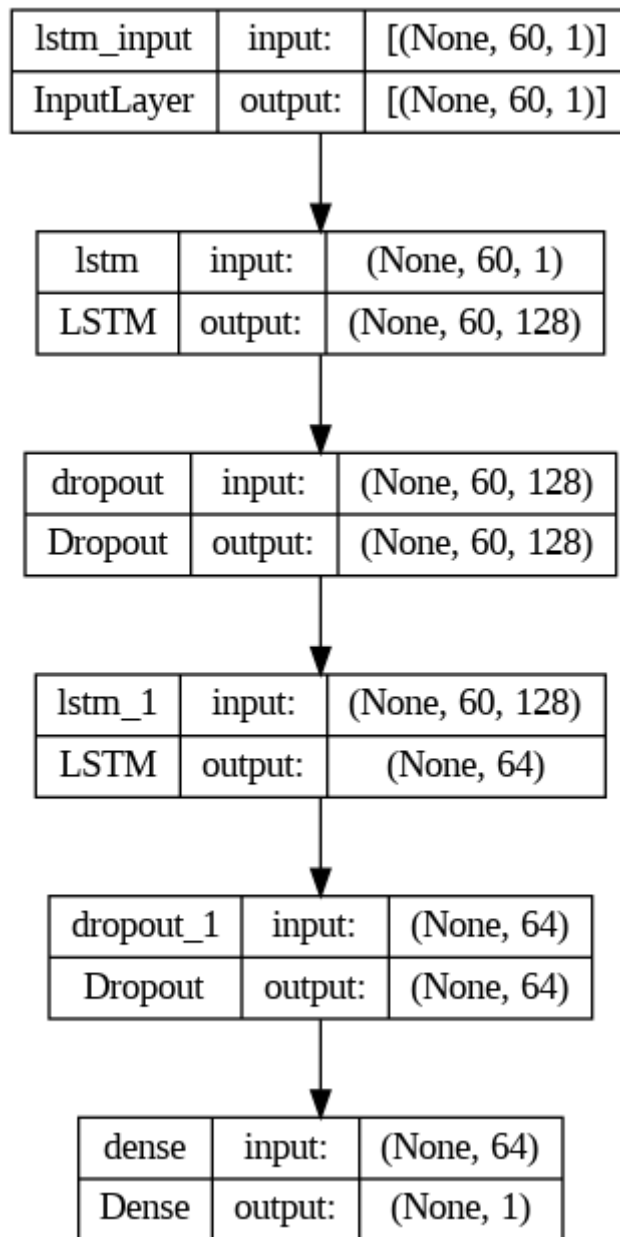


Figura 18: Modelo RNN

6. Apresentação dos Resultados

6.1. Modelo Preditivo com Facebook Prophet

Na figura 19 mostra o resultado da previsão para o ano de 2023. Pode-se perceber que para o ano previsto, os pontos do piloto Hamilton apresentam forte tendência de queda no início da temporada mas com subida durante a mesma.

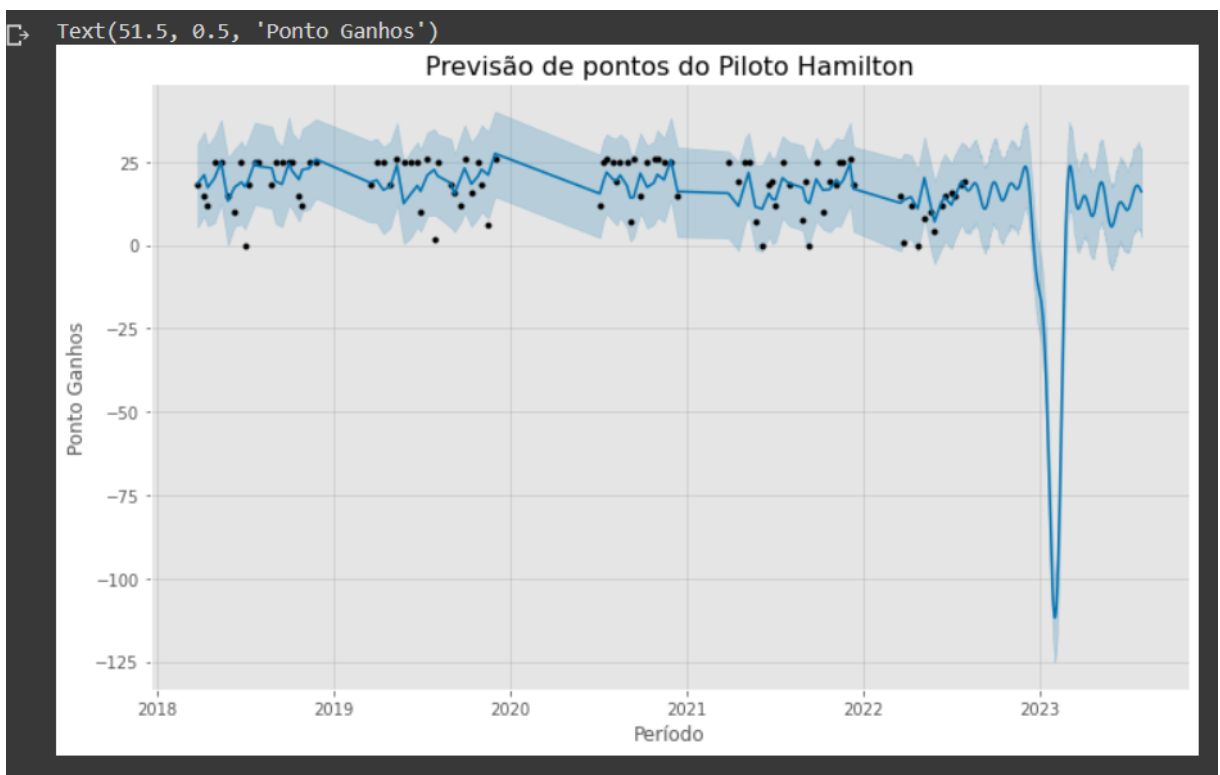


Figura 19: Resultado da previsão entre 2019 a 2022 com previsão para 2023.

A figura 20 demonstra os componentes decompostos da série temporal prevista. Conforme analisado na figura 19, observa-se que há uma tendência de queda de performance no início da temporada e subida durante a mesma do piloto Hamilton.

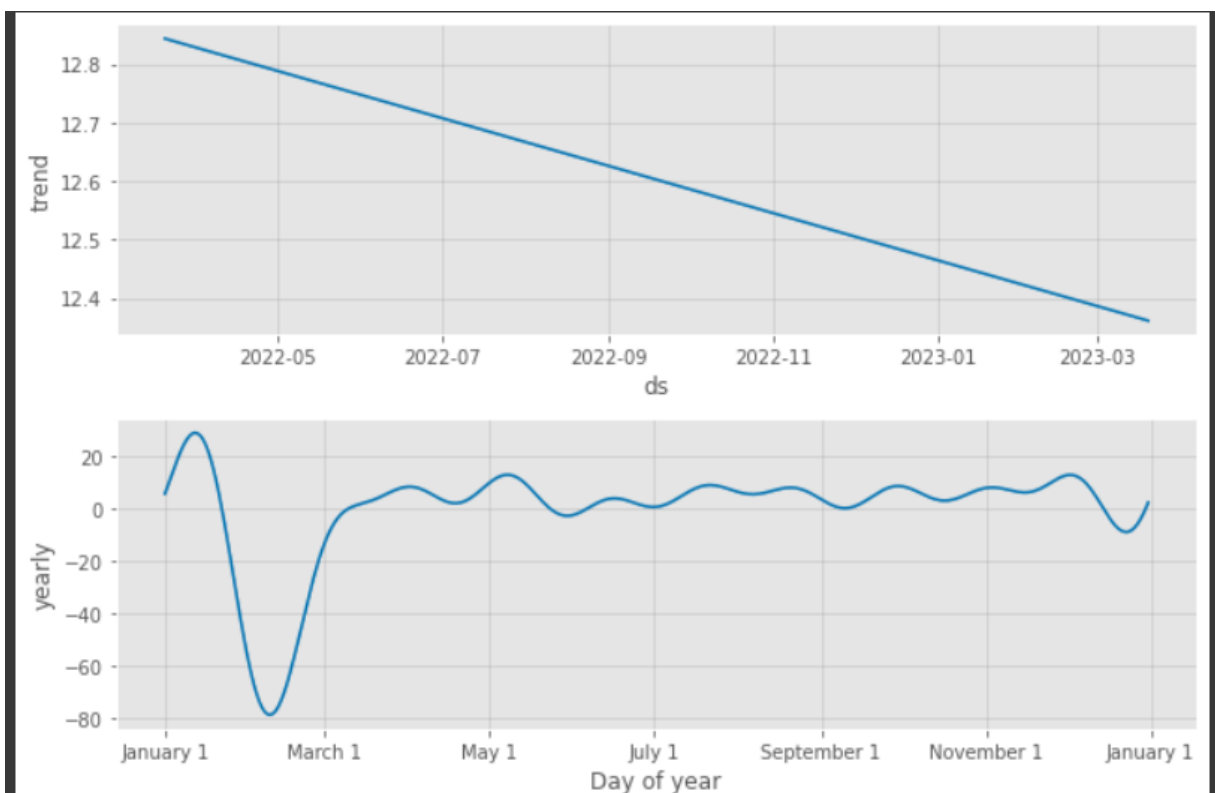


Figura 20: Componentes da previsão de pontos do piloto Hamilton.

A figura 21 mostra os dados de treinamento, teste, previsão e sua banda de variação feita pelo Prophet para o piloto Hamilton.

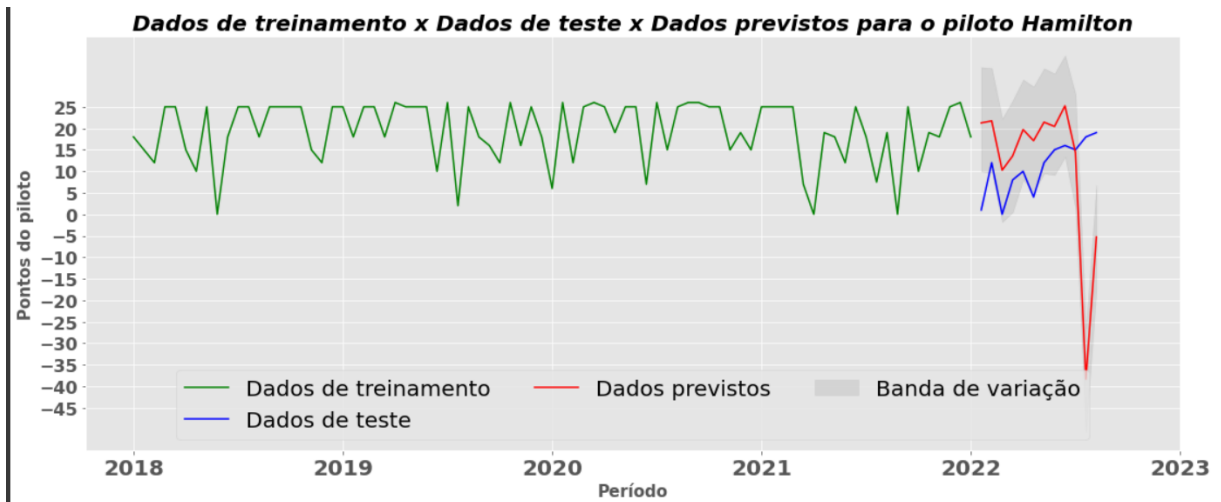


Figura 15: Informações de treinamento, teste e previsão em conjunto.

A previsão ficou além dos testes, uma vez que o período de testes foi mais curto do que 21 meses da previsão.

A figura 22 mostra os dados de teste e dados previstos para melhor entendimento.

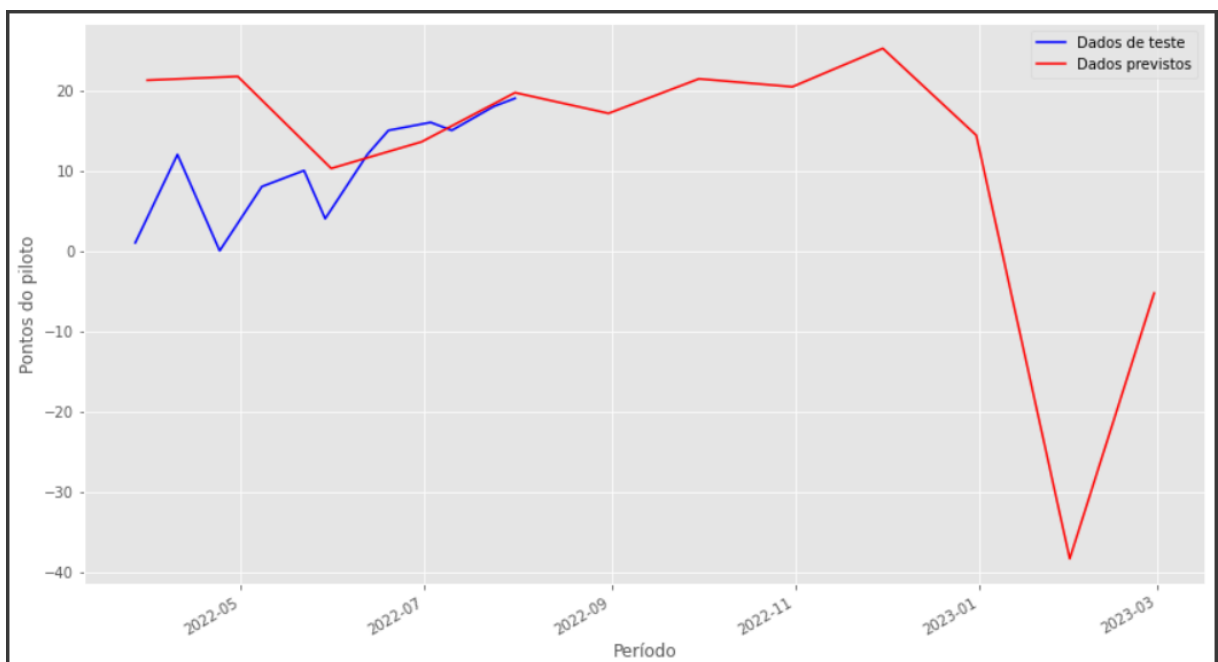


Figura 22: Dados de teste e dados previstos

Erro obtido:

MAE: 14.498288322746289

MSE: 406.5002679784543

RMSE: 20.161851799337636

6.2. Modelo Preditivo com AUTO-ARIMA

A figura 23 mostra o resultado da predição para o ano de 2023, percebe-se que para o ano previsto, a tendência é de que o piloto reduza sua performance nas corridas.

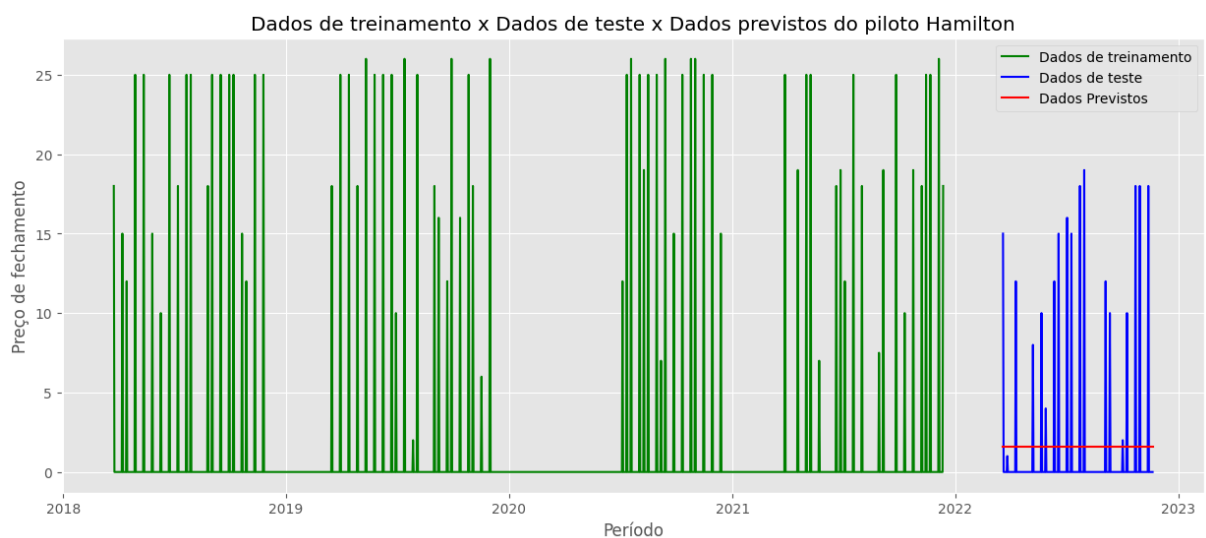


Figura 23: Gráfico Dados de treinamento x Dados de teste e dados previstos

Erro obtido:

MSE: 1711.5378475852874

MAE: 31.589117061015155

RMSE: 41.37073660916962

6.3. Modelos de Machine Learning com RNN (LSTM)

A figura 24 mostra o resultado da predição para o ano de 2023, percebe-se que para o ano previsto, a tendência é de que o piloto reduza sua performance nas corridas. E que os dados de teste se ajustam de forma muito aproximada com os dados previstos pela RNN.

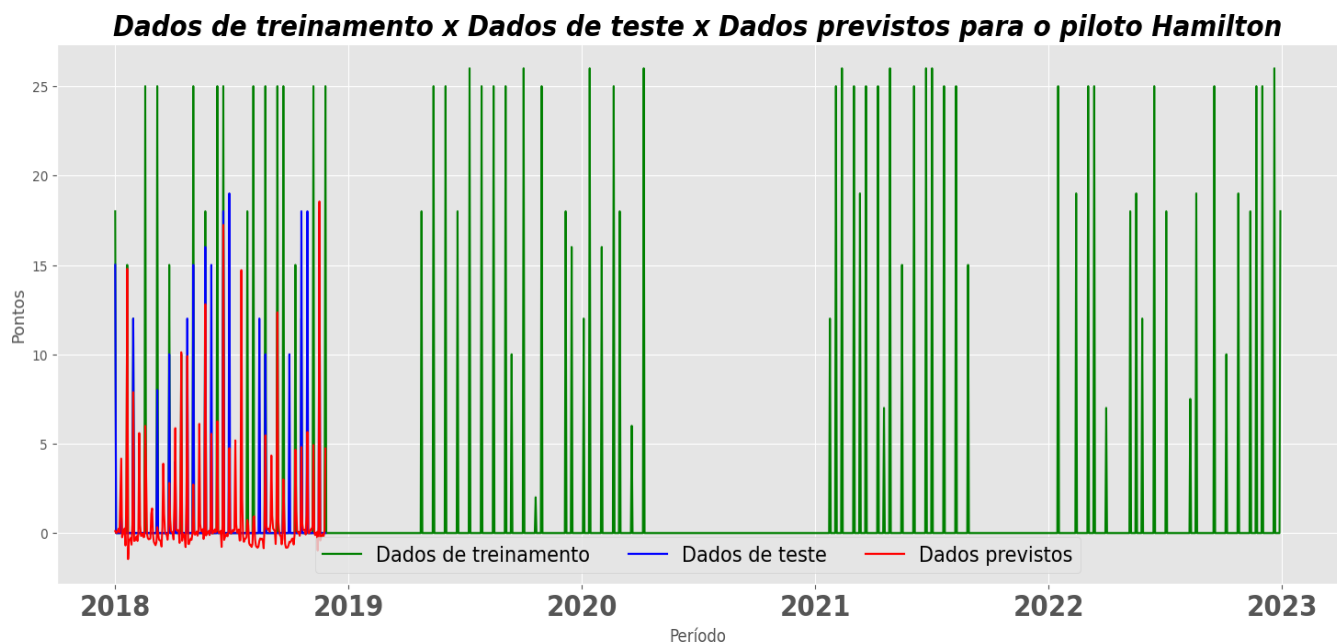


Figura 24: Gráfico Dados de treinamento x Dados de teste e dados previstos

Os dados previstos se ajustam bem nos dados de teste. A figura 25 mostra os dados de teste e dados previstos para comparação.

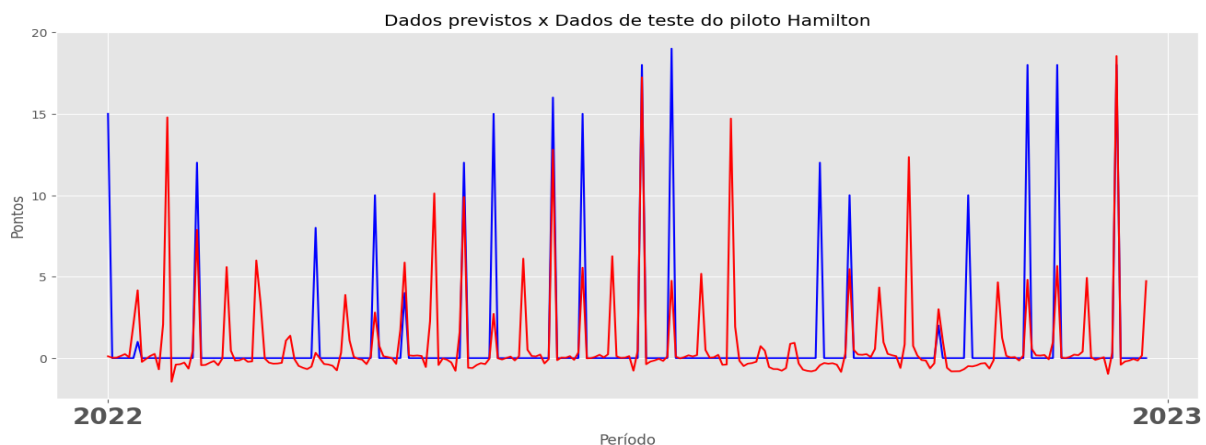


Figura 25: Dados de treinamento, teste em conjunto

Erro obtido:

MAE: 1.310531280253355

MSE: 10.045841890554923

RMSE: 3.1695176116492747

Conclusão:

Após realizar este trabalho e avaliar os modelos Facebook Prophet, Auto-ARIMA e RNN, foram obtidos os erros de previsão para selecionar o modelo ideal. É importante ressaltar que, dependendo do contexto de previsão, pode ser necessário avaliar outros modelos, como o SARIMA, além de outras técnicas de modelagem.

Vale lembrar que, mesmo para este conjunto específico de dados, é importante prestar atenção aos detalhes do processo de coleta, pré-processamento e seleção de recursos relevantes, bem como avaliar continuamente o modelo para garantir que ele permaneça preciso e útil.

Porém, é importante entender que os métodos e técnicas utilizados podem variar de acordo com o tipo de dados, período e contexto específico, exigindo uma abordagem personalizada para cada caso. Sendo assim, é fundamental avaliar cuidadosamente o problema em questão e selecionar a metodologia mais adequada para atingir os objetivos desejados.

7. Links

- Link para o repositório Github com o conteúdo:
https://github.com/MarioMarcos/TCC_PUCMINAS_2021.git
- Link para o vídeo resumo do trabalho no youtube:
[\(10\) TCC Ciência de Dados e Big Data PUC Minas - YouTube](https://www.youtube.com/watch?v=76812qLHMwQ)
<https://www.youtube.com/watch?v=76812qLHMwQ>

REFERÊNCIAS

(Escovedo & Koshiyama, 2020) Tatiana Escovedo & Adriano S. Koshiyama. “Introdução a Data Science — Algoritmos de Machine Learning e métodos de análise”. São Paulo, Ed. Casa do Código, 2020.

Machine Learning: Conceitos e Modelos - Parte I: Aprendizado

Supervisionado, disponível em

https://medium.com/@tatianae_79457/machine-learning-conceitos-e-modelos-f0373bf4f445

Machine Learning: Conceitos e Modelos - Parte II deste artigo, que trata de aprendizado não-supervisionado, está disponível em

https://medium.com/@tatianae_79457/machine-learning-conceitos-e-modelos-parte-ii-aprendizado-n%C3%A3o-supervisionado-fb6d83e4a520.

(Brett Lantz, 2013) - Machine Learning with R: Learn how to use R to apply powerful machine learning methods and gain an insight into real-world applications, Published by Packt Publishing Ltd. Livery Place 35 Livery Street Birmingham B3 2PB, UK

Formula 1 - Ergast Developer API data

Detailed, historical records of Formula 1 races:

<https://www.kaggle.com/sveneschlbeck/formula-1-ergast-developer-api-data>

Formula 1 World Championship (1950 - 2022)

F1 race data from 1950 to 2022:

[Formula 1 World Championship \(1950 - 2022\) | Kaggle](#)

Inteligência Artificial e Machine Learning: O Guia Completo:

<https://www.udemy.com/course/inteligencia-artificial-machine-learning-guia-completo/learn/lecture/21719360#overview>

FACEBOOK Prophet. Disponível em: <<https://opensource.facebook.com/>>.

FACEBOOK Prophet. Disponível também em:

https://facebook.github.io/prophet/docs/quick_start.html#python-api

Taylor, S. J., & Letham, B. (2018). Forecasting at scale. The American Statistician, 72(1), 37-45.

Documentação do Prophet: <https://facebook.github.io/prophet/>

Repositório do Prophet no GitHub: <https://github.com/facebook/prophet>

**Forecasting with ARIMA in R by Rob Hyndman:
<https://otexts.com/fpp2/arma.html>**

**AUTO-ARIMA forecasting in Python by Jason Brownlee:
<https://machinelearningmastery.com/arma-for-time-series-forecasting-with-python/>**

**Time Series Forecasting with ARIMA and Python by Victor Ekpuk:
<https://towardsdatascience.com/time-series-forecasting-with-arma-and-python-be8d9b27a9d6>**

Understanding LSTM Networks por Christopher Olah: Este artigo fornece uma explicação detalhada da arquitetura LSTM e seu uso em aprendizado profundo.

A Beginner's Guide to LSTMs and Recurrent Neural Networks por Ondrej Hubacek: Este tutorial fornece um guia passo a passo para implementar modelos LSTM usando o TensorFlow.

Time Series Prediction Using LSTM Deep Neural Networks por Jason Brownlee: Este tutorial fornece um exemplo prático do uso de modelos LSTM para previsão de séries temporais.