

ORACLE



# Red Bull Racing Honda Beginner Workshop

## Red Bull Racing Honda Developer Session

---

August 2021

# Resources

---

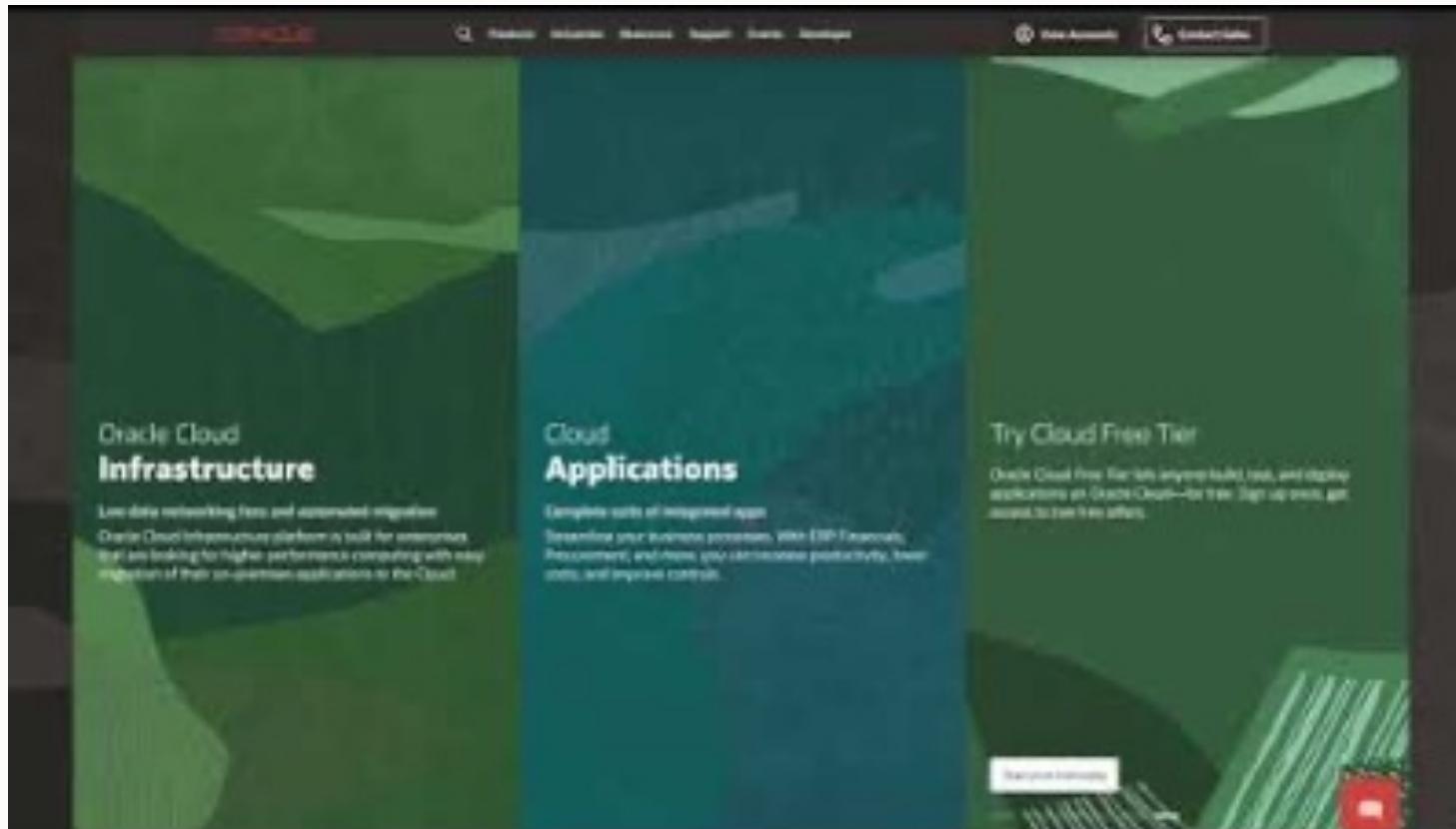


- Slack group:  
<https://bit.ly/3ixPQWT>
- Lab guide:  
[bit.ly>thisPDFplaceHolder](http://bit.ly>thisPDFplaceHolder)

# Provisioning your Free Tier Account

---

# How do I sign up for a free-tier OCI account?



# Deploy JupyterLab environment

---

# Use next URL and Click on the button “Deploy to Oracle Cloud”

<https://github.com/oracle-devrel/redbull-analytics-hol/tree/main/beginners>

The screenshot shows a GitHub page for a README.md file. The title is "Beginners Hands-On Lab". Below it is a section titled "Prerequisites" which contains text about needing an OCI free trial account and credits for Data Science operations. The "Getting Started" section has a step 1 with a red box around the "Deploy to Oracle Cloud" button. Step 2 is listed below it.

github.com/operard/formule1-analytics-hol/blob/main/beginners2/README.md

free computer ebo... microsoft openkm bigdata olivier dev amazon mcentric Bookmarks perso oracle teradata gartner

76 lines (48 sloc) | 5.06 KB

Raw Blame

## Beginners Hands-On Lab

### Prerequisites

You'll need an OCI free trial account ([click here to sign up](#)). We're going to use a ready-to-go image to install the required resources, so all you need to start is a free account.

Registered lab participants should have received \$500 in credits to use for Data Science operations.

### Getting Started

1. Click the button below to begin the deploy of the Data Science stack and custom image:  
[Deploy to Oracle Cloud](#)
2. If needed, log into your account. You should then be presented with the **Create Stack** page. Under **Stack Information** (the first screen), check the box *I have reviewed and accept the Oracle Terms of Use*. Once that box is checked, the information for the stack will be populated automatically.



# Using Resource Manager on Oracle Cloud (Terraform script)



The image shows the Oracle Cloud navigation bar. It includes the Oracle Cloud logo, a search bar with the placeholder "Search for resources, services, and documentation", a location dropdown set to "US West (Phoenix)", and two notification icons.

## Create Stack

1 Stack Information

2 Configure Variables

3 Review

Welcome!

You're here because you clicked a button to deploy cloud resources, using the [package](#) identified below.

Package URL: <https://github.com/operard/formule1-analytics-ho/releases/latest/download/redbullicompute.zip>

**1**  I have reviewed and accept the [Oracle Terms of Use](#).

Working Directory  
The `redbullicompute` folder is being used as the working directory.

Name Optional  
`redbullicompute.zip-20210802153007`

Description Optional

Create in compartment  
`devrel3 (root)`

Terraform version  
`0.14.x`

**Support for Terraform version 0.11.x ends in May 2021.**

Tags

Tagging is a metadata system that allows you to organize and track resources within your tenancy. Tags are composed of keys and values that can be attached to resources.

[Learn more about tagging](#)

Tag Namespace	Tag Key	Value

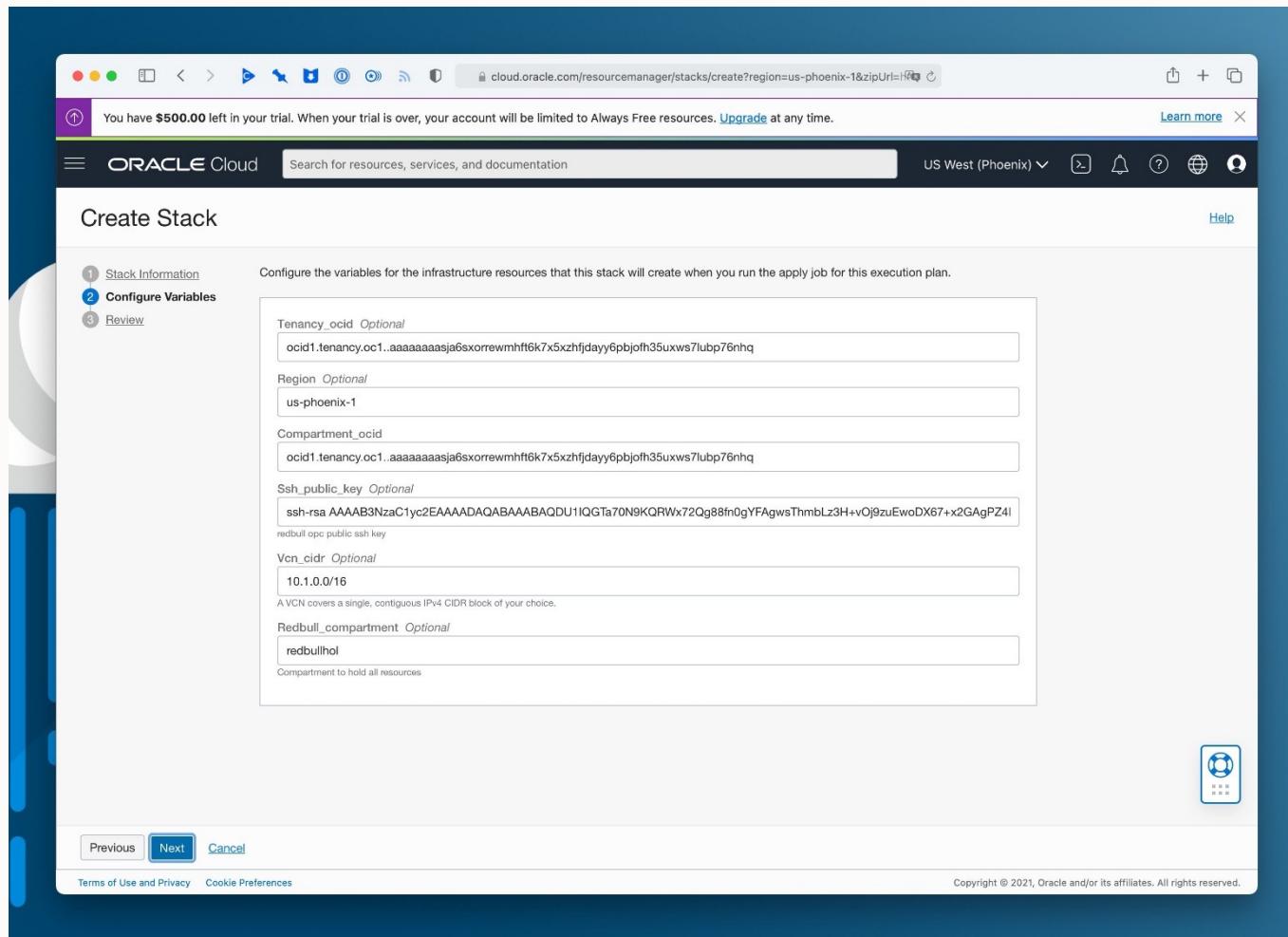
**2**

[Next](#) [Cancel](#)

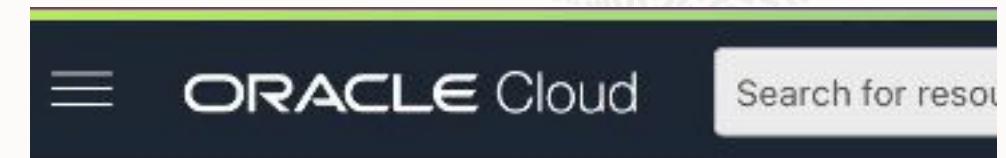
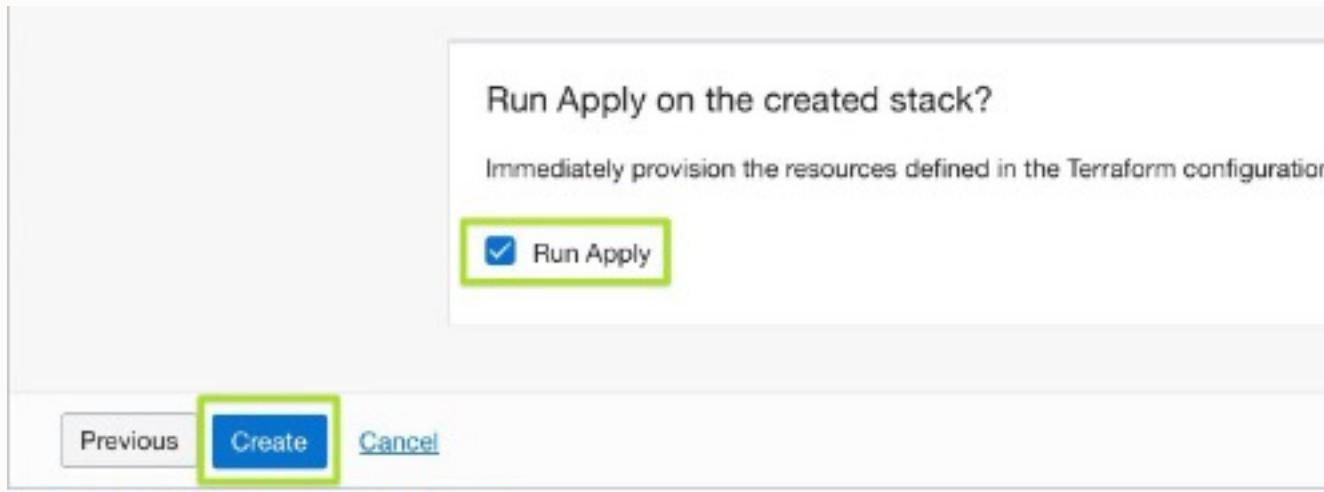
Terms of Use and Privacy | Cookie Preferences



# Using Resource Manager on Oracle Cloud (Terraform script) Step 2



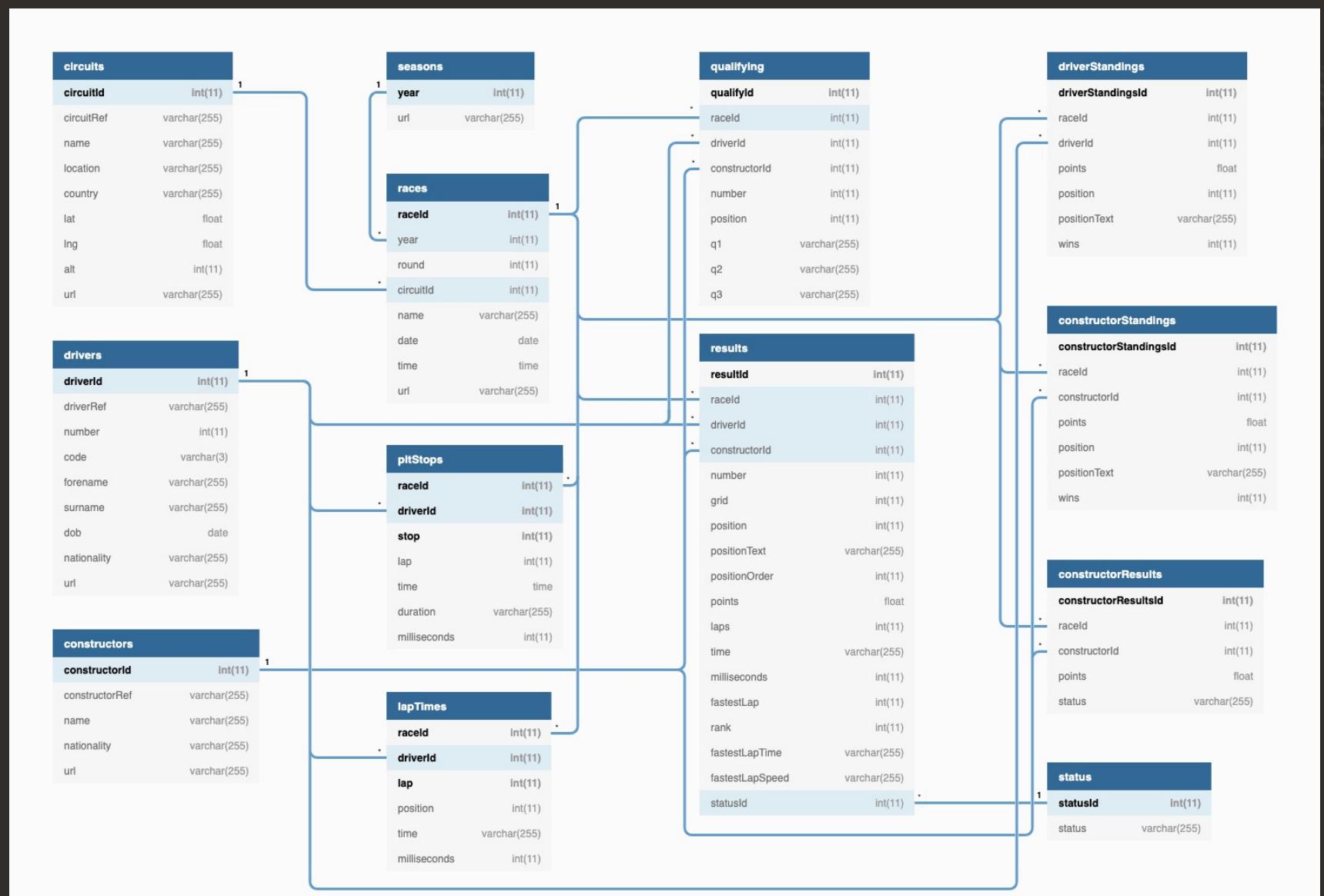
# Using Resource Manager on Oracle Cloud (Terraform script: Run Apply)



- 10 min to deploy

## 1

## Data Collection



## 2

## Data Preparation

The screenshot shows a Jupyter Notebook interface with the following content:

```
df_dum.shape  
(14272, 100)  
df_dum.to_csv('./data/f1_df_final.csv', index = False)  
df_dum.head()
```

Below the code, a table is displayed:

	season	round	weather_warm	weather_cold	weather_dry	weather_wet
14	1983	1	False	False	True	False
5	1983	1	False	False	True	False
3	1983	1	False	False	True	False
0	1983	1	False	False	True	False
6	1983	1	False	False	True	False

5 rows × 100 columns

The Jupyter interface includes a sidebar with the following file list:

- 0
- / formule1 workshop / data
- ..
- constructor\_standings.csv
- driver\_standings.csv
- f1\_df\_final.csv
- f1\_models\_dict.json
- qualifying.csv
- races.csv
- results.csv
- weather.csv

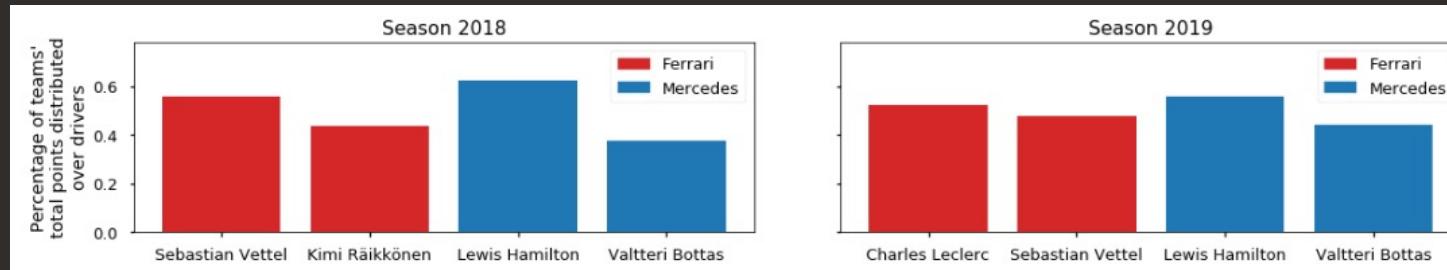
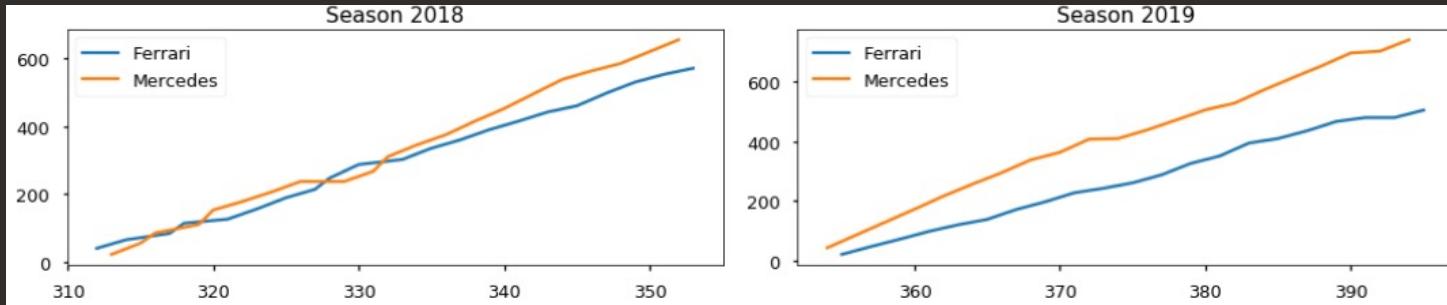
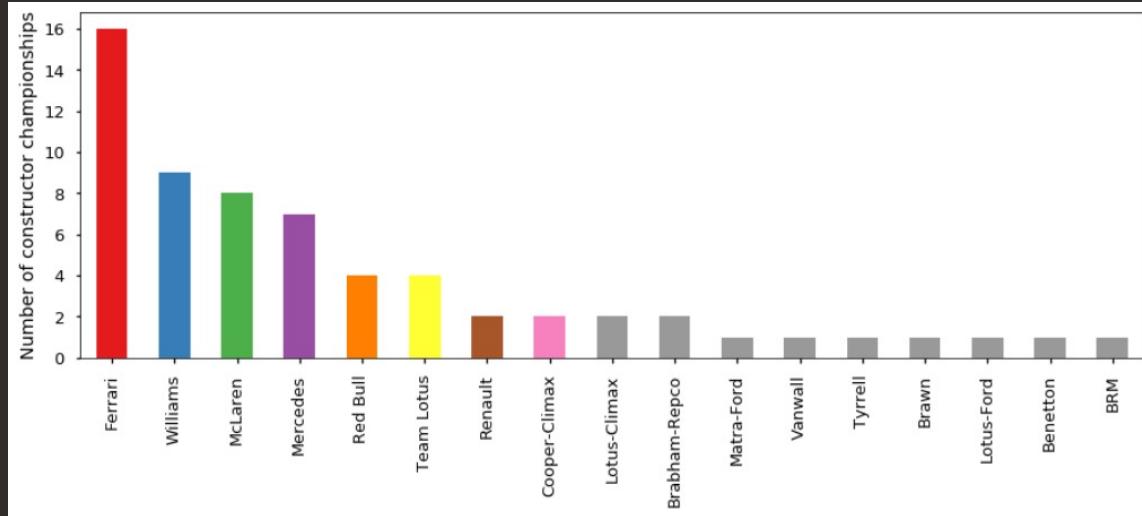
The file "f1\_df\_final.csv" is highlighted with a red box.

**Build a F1 merged Dataset to discover the best ML model:**

- Season List
- Race Schedule
- Race Results
- Qualifying Results
- Standings
- Driver Information
- Constructor Information
- Circuit Information
- Finishing Status
- Lap Times
- Pit Stops
- Qualifying Results
- Driver Information
- Lap Times
- Pit Stops

## 3

# Exploratory Data Analysis



**Explore the F1 merged Dataset:**

- Compare Red Bull / Mercedes / Ferrari
- Compare the teams' results
- Compare Constructors results.
- ...Etc...

```
import numpy as np
import pandas as pd
from sklearn.metrics import confusion_matrix, precision_score
from sklearn.linear_model import LogisticRegression, LinearRegression
from sklearn import svm
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.neural_network import MLPClassifier, MLPRegressor

models = []
models = [('linear_regression', LinearRegression()),
          ('logistic_regression', LogisticRegression()),
          ('neural_network_classifier',MLPClassifier()),
          ('nn_regressor', MLPRegressor()),
          ('svm_classifier',svm.SVC()),
          ('svm_regressor', svm.SVR()),
          ('Gradient Boosting Regressor', GradientBoostingRegressor()),
          ('random_forest_classifier', RandomForestClassifier()),
          ('random_forest_regressor', RandomForestRegressor())
         ]

results_rmse = []
results_mae = []
names = []
```

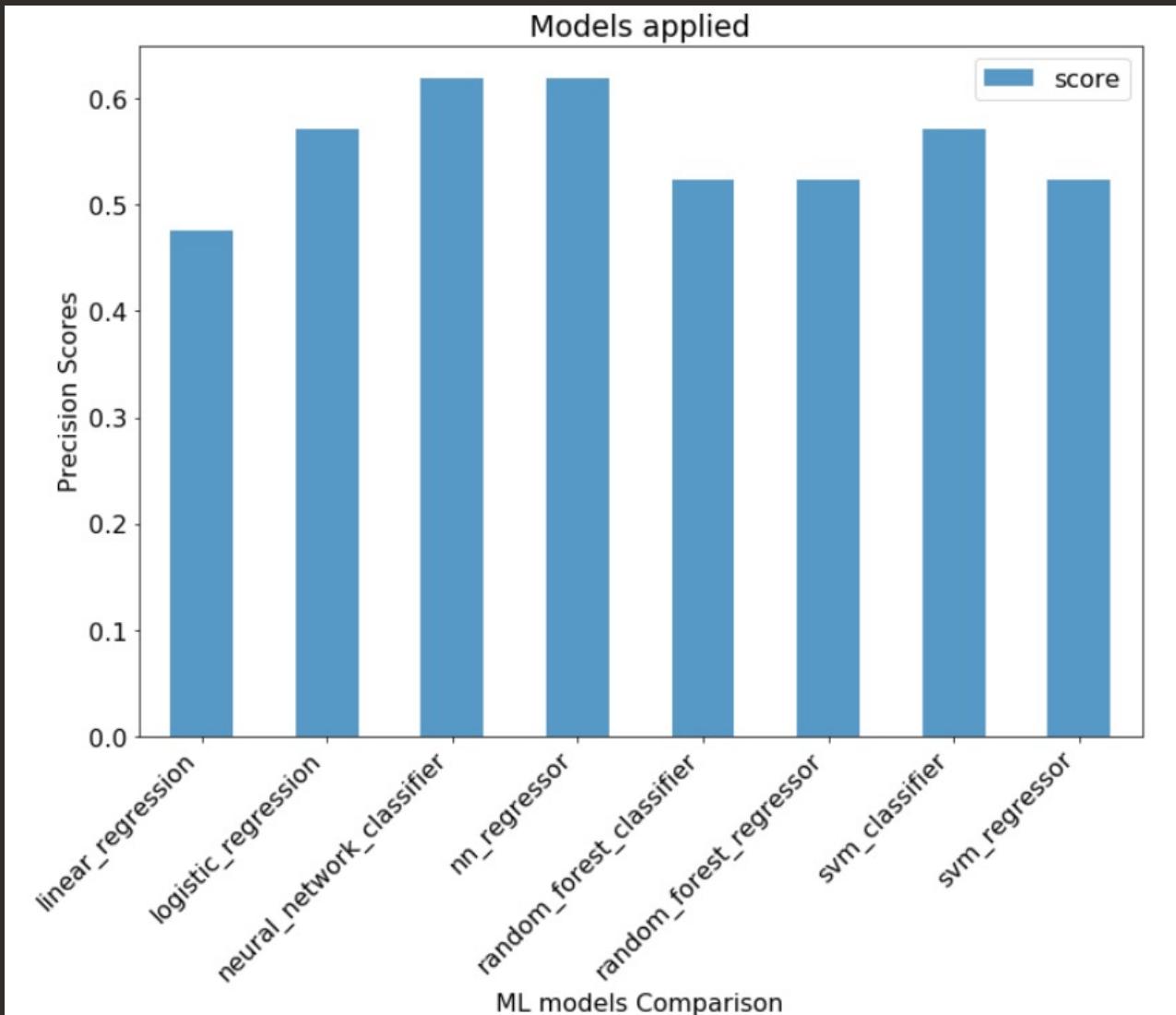
## Build several ML models with the F1 merged Dataset:

- linear\_regression
- logistic\_regression
- neural\_network\_classifier
- nn\_regressor
- random\_forest\_classifier
- random\_forest\_regressor
- svm\_classifier
- svm\_regressor



## 5

## Model Serving



It seems the Neural Network Classification is the best model to return the highest scores, correctly predicting the winner for 62% of the races.

Deploy the best trained model as a Model Serving:

- API REST using the feature importance (variables) for the best model.

## 6

## UI Access

## Formula 1 Predictor

Silverstone Circuit

Warm

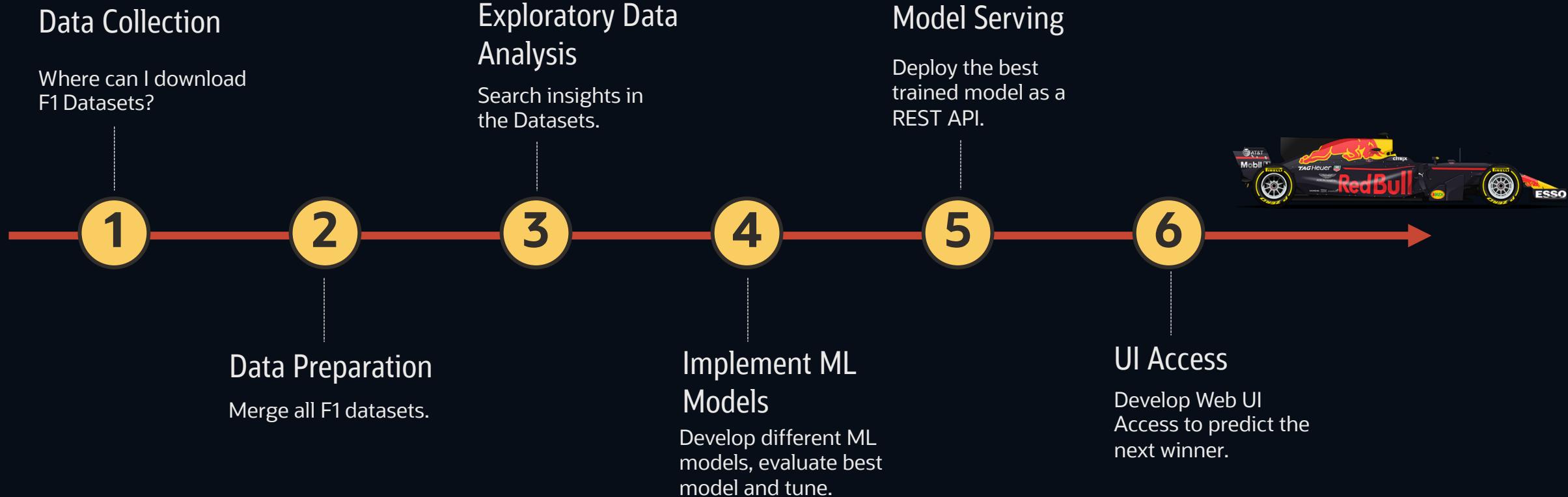
**predict**



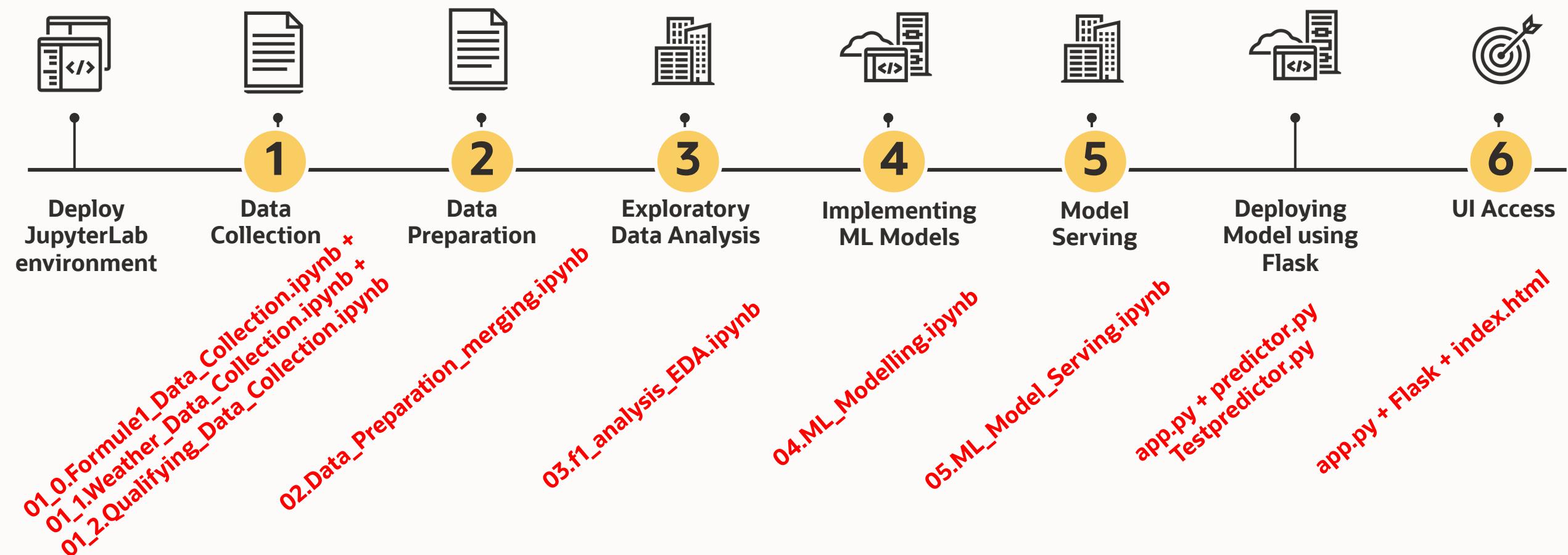
- Etc.  
**Call the Winner Prediction Algorithm:**

- Circuit
- Weather
- Qualifying Times

# At a glance: today's steps



# Red Bull Racing Honda Developer Journey Map



# Access to your compartment and Lab Image deployed.

The screenshot shows the Oracle Cloud interface. A green circle labeled '1' is on the top left. A green circle labeled '2' is on the sidebar under 'Compute'. A green circle labeled '3' is on the 'Instances' button. The main area displays the title 'Instances in redbullhol Compartment' and a table of instances. One instance, 'redbulllab1', is highlighted with a green border.

Name	State	Public IP	Private IP	Shape	OCPUs	Actions
redbulllab1	Running	192.168.1.10	192.168.1.3	VM.Standard2.2	2	[Edit]

## List Scope

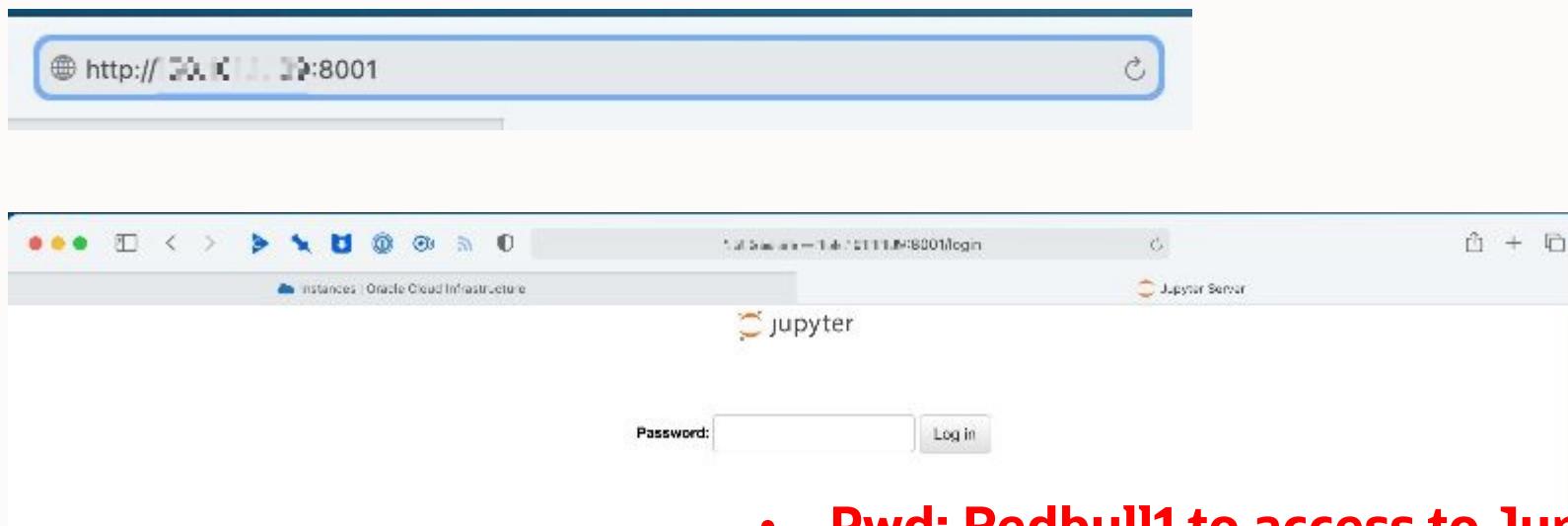
### Compartment

redbullhol

devrel3 (root)/redbullhol

## Filtering

# Access to JupyterLab on your image using the IP



- **Pwd: Redbull1 to access to JupyterLab**

# JupyterLab Access

The screenshot shows the JupyterLab interface. On the left, there is a file browser with a sidebar containing icons for File, Edit, View, Run, Kernel, Tabs, Settings, and Help. The file browser displays a list of files and directories:

Name	Last Modified
redbull-analytics-hol	10 hours ago
redbullevn	8 days ago
jupyter.pid	4 hours ago
launchjupyterlab.sh	8 days ago
nohup.log	seconds ago

A red box highlights the directory "redbull-analytics-hol". A search bar at the top of the file browser says "Filter files by name".

On the right, there is a "Launcher" panel with the following sections:

- Notebook**: Shows a Python 3 icon.
- Console**: Shows a Python 3 icon.
- Other**: Shows icons for Terminal, Text File, Markdown File, Python File, and Show Contextual Help.

Step 1: Execute 3 notebooks to download Data and create your own CSV Files.

Use Notebooks Version in “from\_scratch” Directory.

---

30-40 min (Offline Execution)

# Download Dataset from ERGAST Web

The screenshot shows a Jupyter Notebook environment. On the left, there is a file browser window titled 'Launcher' showing a directory structure under '/ ... / beginners / from\_scratch /'. The file '01\_0.Formule1\_Data\_Collection.ipynb' is selected and highlighted in blue. The main workspace contains a title 'Formule 1 Data Collection from ERGAST Web' and several code cells:

```
[2]: import warnings
warnings.filterwarnings("ignore")

[3]: import time
start = time.time()

[4]: import os
import pandas as pd
import numpy as np
from pprint import pprint
import requests

[5]: # I will use this function later to calculate points and wins prior to the race

def lookup (df, team, points):
    df['lookup1'] = df.season.astype(str) + df[team] + df['round'].astype(str)
    df['lookup2'] = df.season.astype(str) + df[team] + (df['round']-1).astype(str)
    new_df = df.merge(df[['lookup1', points]], how = 'left', left_on='lookup2',right_on='lookup1')
    new_df.drop(['lookup1_x', 'lookup2', 'lookup1_y'], axis = 1, inplace = True)
    new_df.rename(columns = {points+'_x': points+'_after_race', points+'_y': points}, inplace = True)
    new_df[points].fillna(0, inplace = True)
    return new_df
```

# Extracting Races & results from ERGAST Web

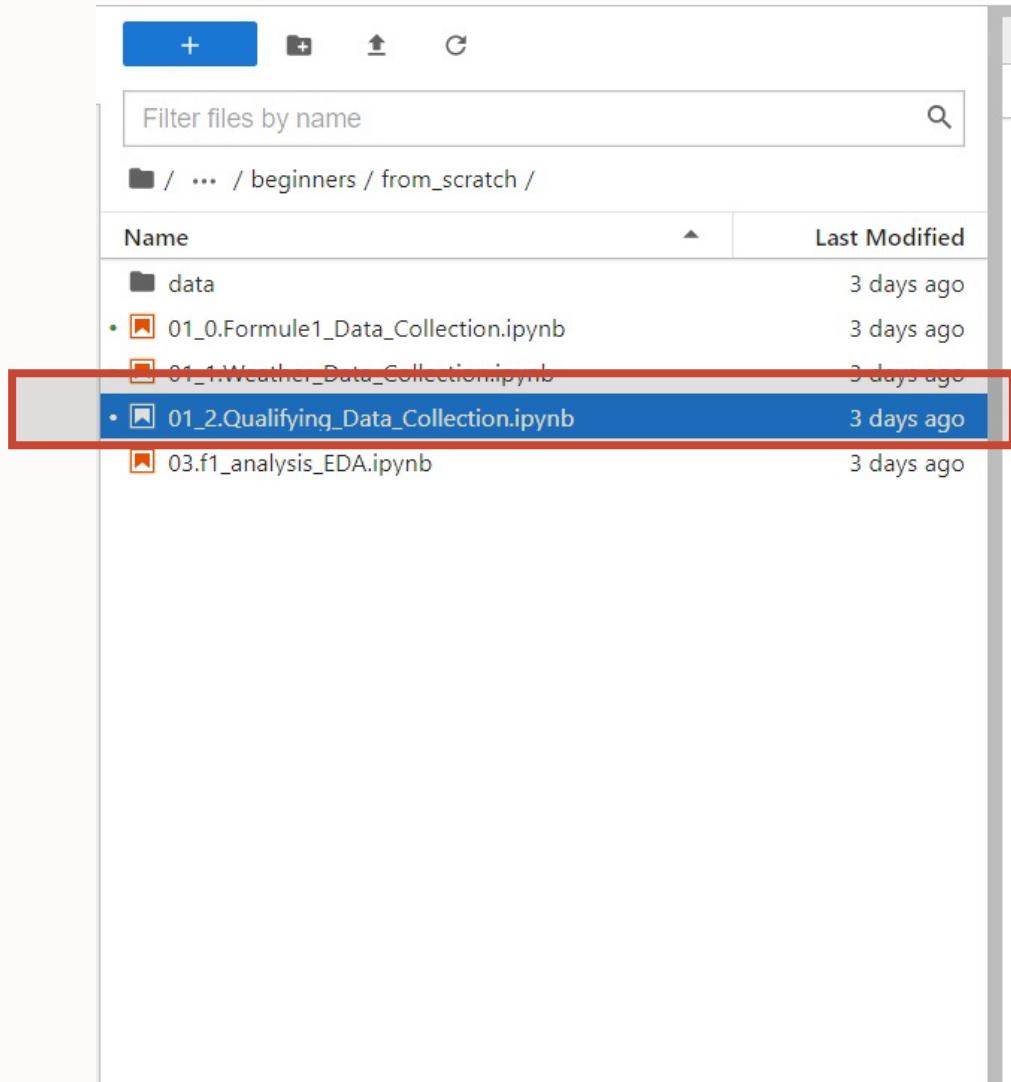
## Races

```
races = {'season': [],  
         'round': [],  
         'circuit_id': [],  
         'lat': [],  
         'long': [],  
         'country': [],  
         'date': [],  
         'url': []}  
  
for year in list(range(1950,2020)):  
  
    url = 'https://ergast.com/api/f1/{}/json'
```

## Results

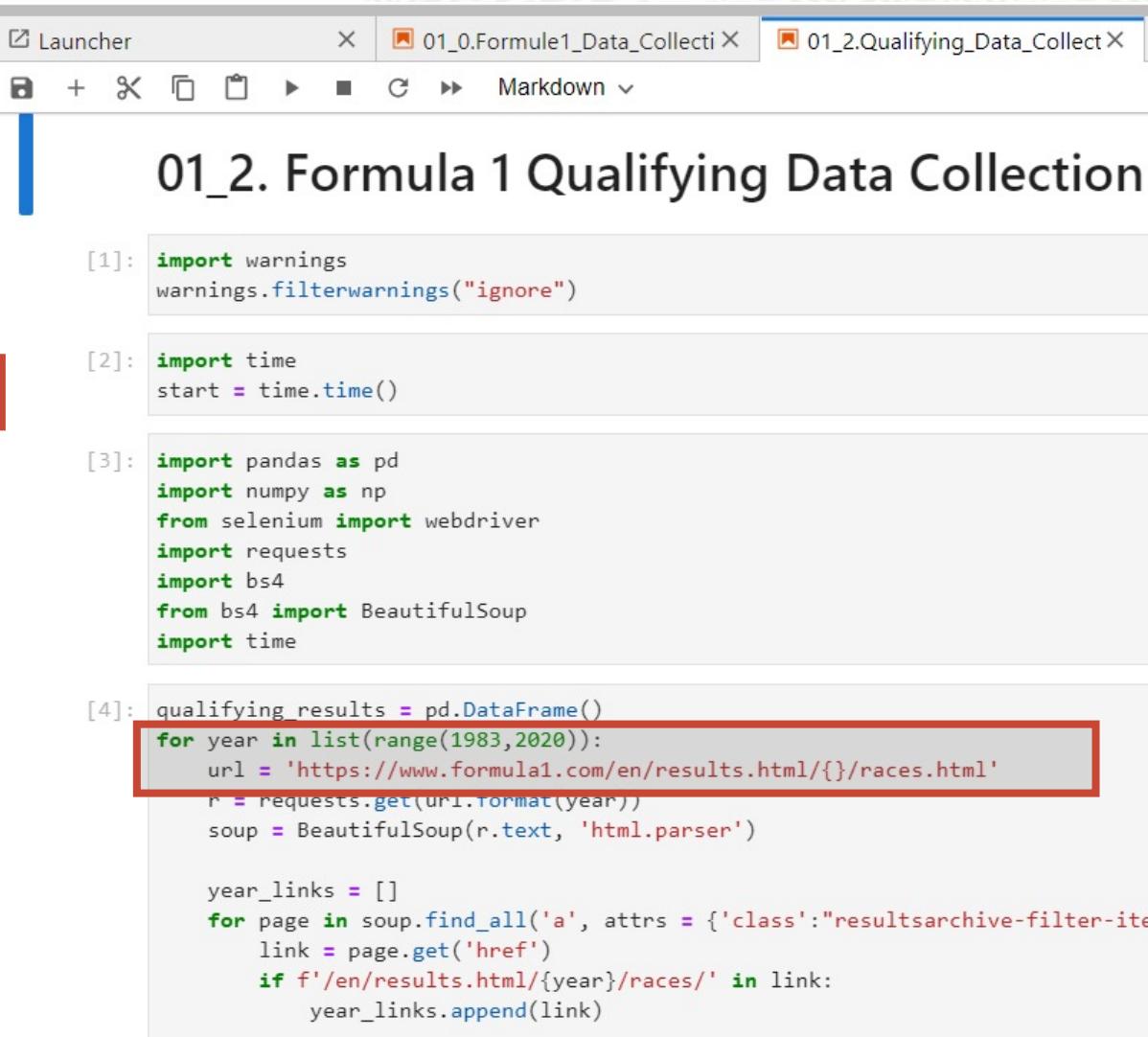
```
results = {'season': [],  
          'round':[],  
          'circuit_id':[],  
          'driver': [],  
          'date_of_birth': [],  
          'nationality': [],  
          'constructor': [],  
          'grid': [],  
          'time': [],  
          'status': [],  
          'points': [],  
          'podium': [],  
          'url': []}  
  
for n in list(range(len(rounds))):  
    for i in rounds[n][1]:  
  
        url = 'http://ergast.com/api/f1/{}/{}//results.json'
```

# Extracting the Qualifying Dataset from Formula 1



A screenshot of a file browser interface. At the top, there are buttons for creating a new folder (+), creating a new file (New), uploading files (Upload), and deleting files (Delete). Below this is a search bar labeled "Filter files by name" with a magnifying glass icon. The main area shows a list of files under the path "... / beginners / from\_scratch /". The columns are "Name" and "Last Modified". The files listed are:

- data (3 days ago)
- 01\_0.Formule1\_Data\_Collection.ipynb (3 days ago)
- 01\_1.Weather\_Data\_Collection.ipynb (3 days ago)
- 01\_2.Qualifying\_Data\_Collection.ipynb (3 days ago) This file is selected.
- 03.f1\_analysis\_EDA.ipynb (3 days ago)



A screenshot of a Jupyter Notebook cell titled "01\_2. Formula 1 Qualifying Data Collection". The cell contains the following Python code:

```
[1]: import warnings
warnings.filterwarnings("ignore")

[2]: import time
start = time.time()

[3]: import pandas as pd
import numpy as np
from selenium import webdriver
import requests
import bs4
from bs4 import BeautifulSoup
import time

[4]: qualifying_results = pd.DataFrame()
for year in list(range(1983,2020)):
    url = 'https://www.formula1.com/en/results.html/{}/races.html'
    r = requests.get(url.format(year))
    soup = BeautifulSoup(r.text, 'html.parser')

    year_links = []
    for page in soup.find_all('a', attrs = {'class':'resultsarchive-filter-item-link'}):
        link = page.get('href')
        if f'/en/results.html/{year}/races/' in link:
            year_links.append(link)
```

The code uses the `pd.DataFrame()` constructor to create an empty DataFrame named `qualifying_results`. It then iterates over the years from 1983 to 2020. For each year, it constructs a URL to the Formula 1 results page and sends a GET request using `requests.get()`. The response is parsed with `BeautifulSoup` to find all links corresponding to race pages. These links are collected in a list named `year_links`.

Step 1: Execute 3 notebooks to check Datasets downloaded.

Use Notebooks Version in “beginners” Directory.

---

# 01\_0.Formule1\_Data\_Collection.ipynb

+  Filter files by name	
/ redbull-analytics-hol / beginners /	
Name	
data	8 days ago
data_f1	8 hours ago
docs	8 hours ago
from_scratch	3 days ago
models	8 days ago
web	2 hours ago
01_0.Formule1_Data_Collection.ipynb	8 hours ago
01_1.Weather_Data_Collection.ipynb	8 hours ago
01_2.Qualifying_Data_Collection.ipynb	8 hours ago
02.Data_Preparation_merging.ipynb	8 hours ago
03.f1_analysis_EDA.ipynb	4 hours ago
04.ML_Modelling.ipynb	7 hours ago
05.ML_Model_Serving.ipynb	3 hours ago
README.md	7 hours ago

Launcher 01\_0.Formule1\_Data\_Collection.ipynb 05.ML\_Model\_Serving.ipynb 01\_0.Formule1\_Data\_Collection.ipynb 01\_2.Qualifying\_Data\_Collection.ipynb 01\_1.Weather\_Data\_Collection.ipynb

## Formula 1 Data Collection from ERGAST Web

First we're going to collect our data. For this lab we're pulling from ERGAST. We've gone ahead and collected the data in advance to save time during this lab. If you want to see how it was done and find code you can use in your own projects, check out the `from_scratch` folder.

```
[1]: import warnings  
warnings.filterwarnings("ignore")
```

```
[2]: import time  
start = time.time()
```

```
[3]: import os  
import pandas as pd  
import numpy as np  
from pprint import pprint  
import requests
```

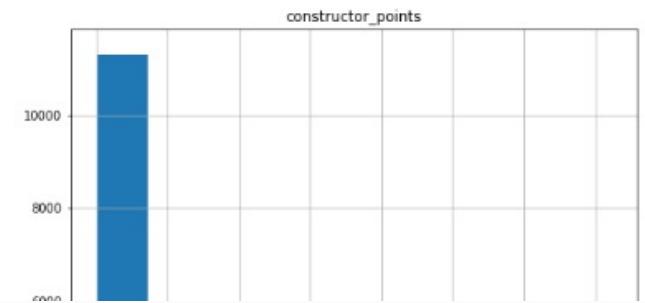
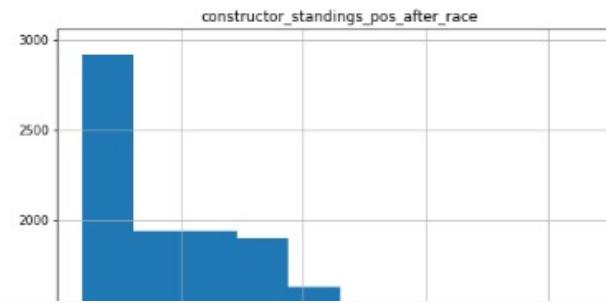
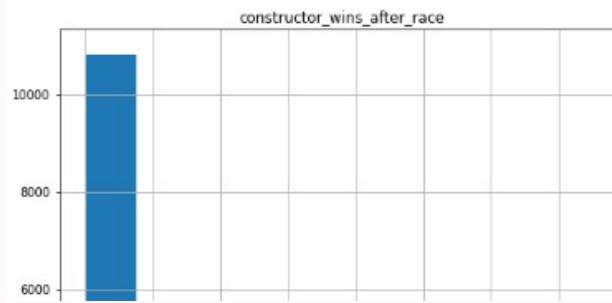
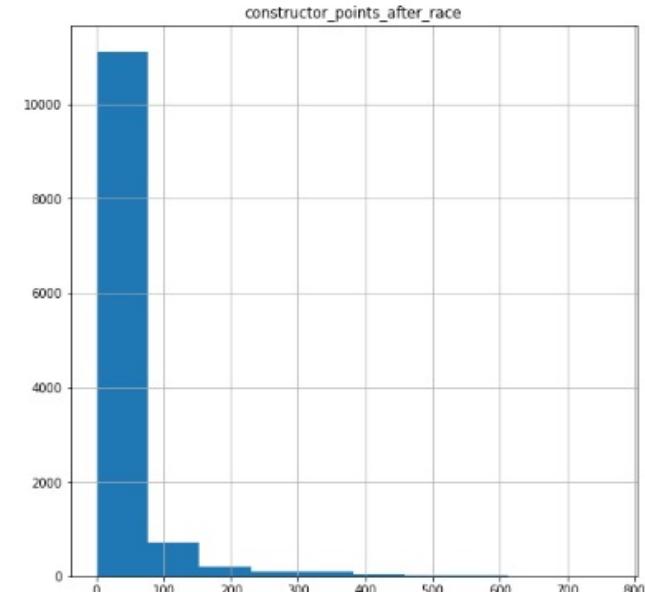
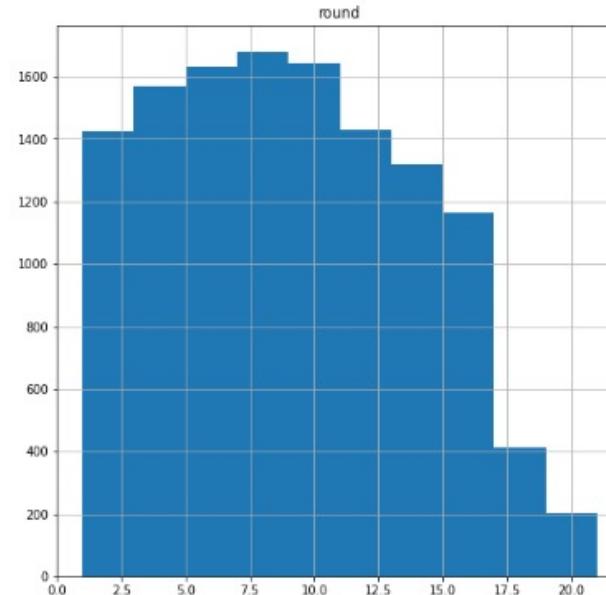
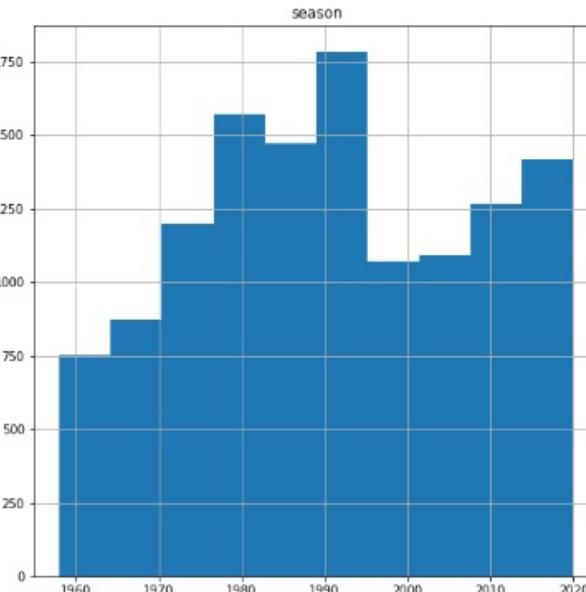
- **Check Races Dataset**
- **Races Results**
- **Driver Standings**
- **Team/Constructor Standings**

### Races

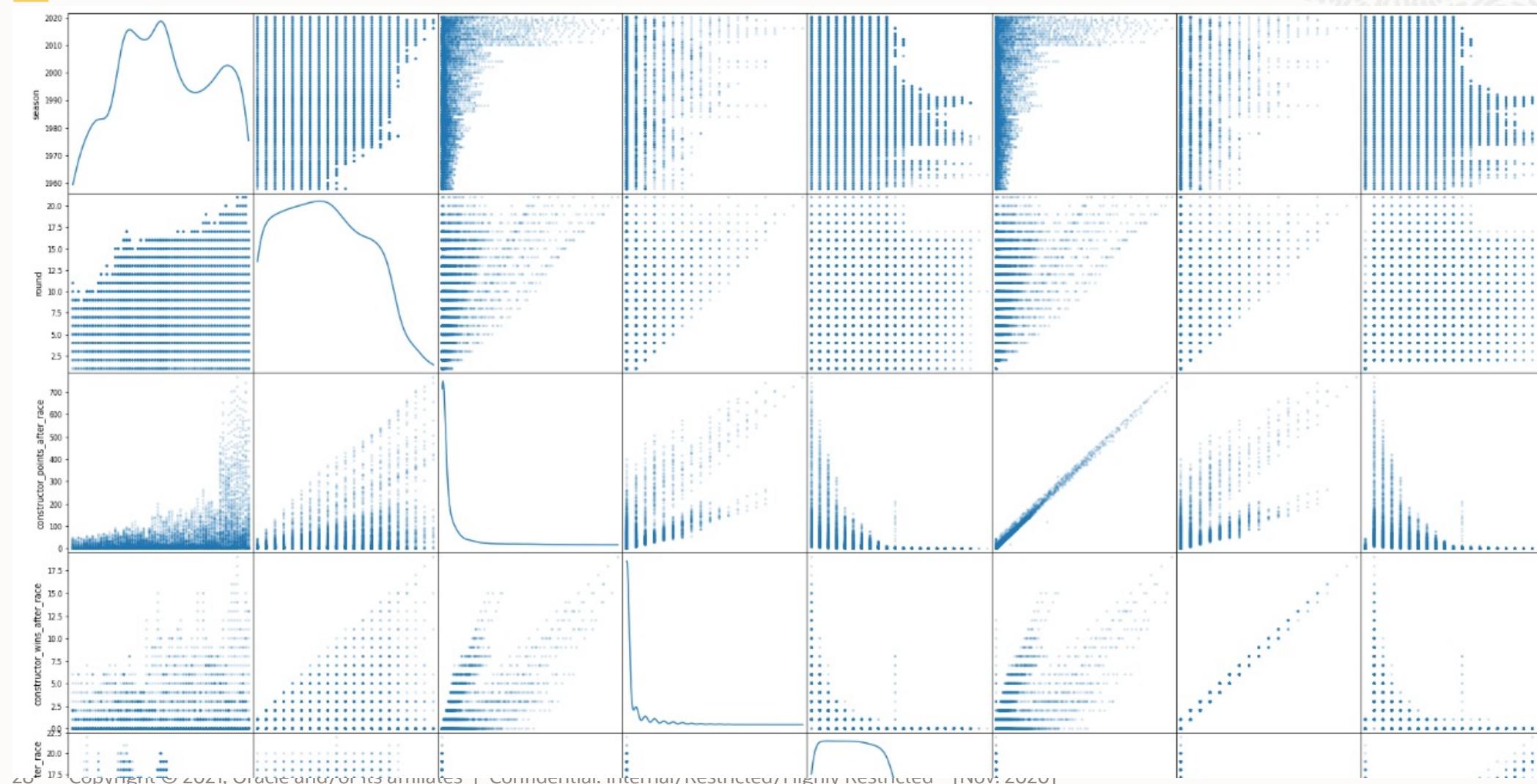
```
[4]: races = pd.read_csv('../data/races.csv')
```

```
[5]: print(races.shape)
```

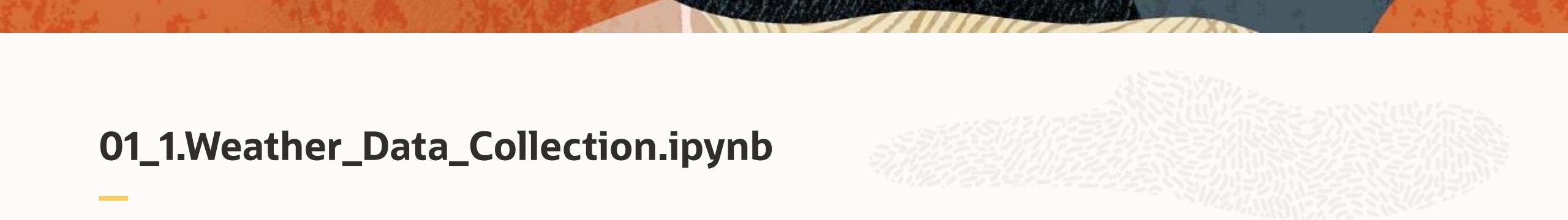
# 01\_0.Formule1\_Data\_Collection.ipynb: Feature Distributions



# 01\_0.Formule1\_Data\_Collection.ipynb: Feature-Feature Relationships



# 01\_1.Weather\_Data\_Collection.ipynb



Filter files by name

/ redbull-analytics-hol / beginners /

Name	Last Modified
data	8 days ago
data_f1	8 hours ago
docs	8 hours ago
from_scratch	3 days ago
models	8 days ago
web	2 hours ago
• 01_0.Formule1_Data_Collection.ipynb	8 hours ago
• 01_1.Weather_Data_Collection.ipynb	8 hours ago
01_2.Qualifying_Data_Collection.ipynb	8 hours ago
02.Data_Preparation_merging.ipynb	8 hours ago
03.f1_analysis_EDA.ipynb	4 hours ago
04.ML_Modelling.ipynb	7 hours ago
• 05.ML_Model_Serving.ipynb	3 hours ago
README.md	7 hours ago

[6]: races.shape

[6]: (1018, 8)

## Weather Dataset Analysis

[7]: weather\_info = pd.read\_csv('./data/weather.csv')

[8]: weather\_info.shape

[8]: (1018, 9)

[9]: weather\_info.head()

[9]:

	season	round	circuit_id	weather	weather_warm	weather_cold	weather_dry	weather_wet	weather_cloudy
0	1950	1	silverstone	Sunny, mild, dry.	0	0	0	0	0
1	1950	2	monaco	not found	0	0	0	0	0
2	1950	3	indianapolis	Rainy	0	0	0	1	0
3	1950	4	bremgarten	Warm, dry and sunny	1	0	1	0	0
4	1950	5	spa	Warm, dry and sunny	1	0	1	0	0

• **Weather Conditions Mix during the Race**

## 01\_2.Qualifying\_Data\_Collection.ipynb

A screenshot of a file explorer interface. At the top, there are icons for creating a new file (+), creating a new folder (+), upload (up arrow), and download (down arrow). Below that is a search bar labeled "Filter files by name" with a magnifying glass icon. The main area shows a list of files and folders under the path "/redbull-analytics-hol / beginners /". The list includes:

Name	Last Modified
data	8 days ago
data_f1	8 hours ago
docs	8 hours ago
from_scratch	3 days ago
models	8 days ago
web	2 hours ago
01_0.Formule1_Data_Collection.ipynb	8 hours ago
01_1.Weather Data Collection.ipynb	8 hours ago
01_2.Qualifying_Data_Collection.ipynb	8 hours ago
02.Data_Preparation_merging.ipynb	8 hours ago
03.f1_analysis_EDA.ipynb	4 hours ago
04.ML_Modelling.ipynb	7 hours ago
05.ML_Model_Serving.ipynb	3 hours ago
README.md	7 hours ago

A screenshot of a Jupyter Notebook titled "01\_2.Formula 1 Qualifying Data Collection". The notebook has a toolbar at the top with icons for Launcher, file operations, and Markdown dropdown. The main content area contains the following code:

```
[1]: import warnings  
warnings.filterwarnings("ignore")  
  
[2]: import time  
start = time.time()  
  
[3]: import pandas as pd  
import numpy as np  
from selenium import webdriver  
import requests  
import bs4  
from bs4 import BeautifulSoup  
import time  
  
[4]: qualifying_results = pd.read_csv('./data/qualifying.csv')  
  
[5]: print(qualifying_results.shape)  
(14559, 6)  
  
[6]: len(qualifying_results)  
  
[6]: 14559
```

**• We suppose that Qualifying dataset is an important information to predict the result of a race.**

## Step 2: Data Preparation.

---

## 02.Data\_Preparation\_merging.ipynb

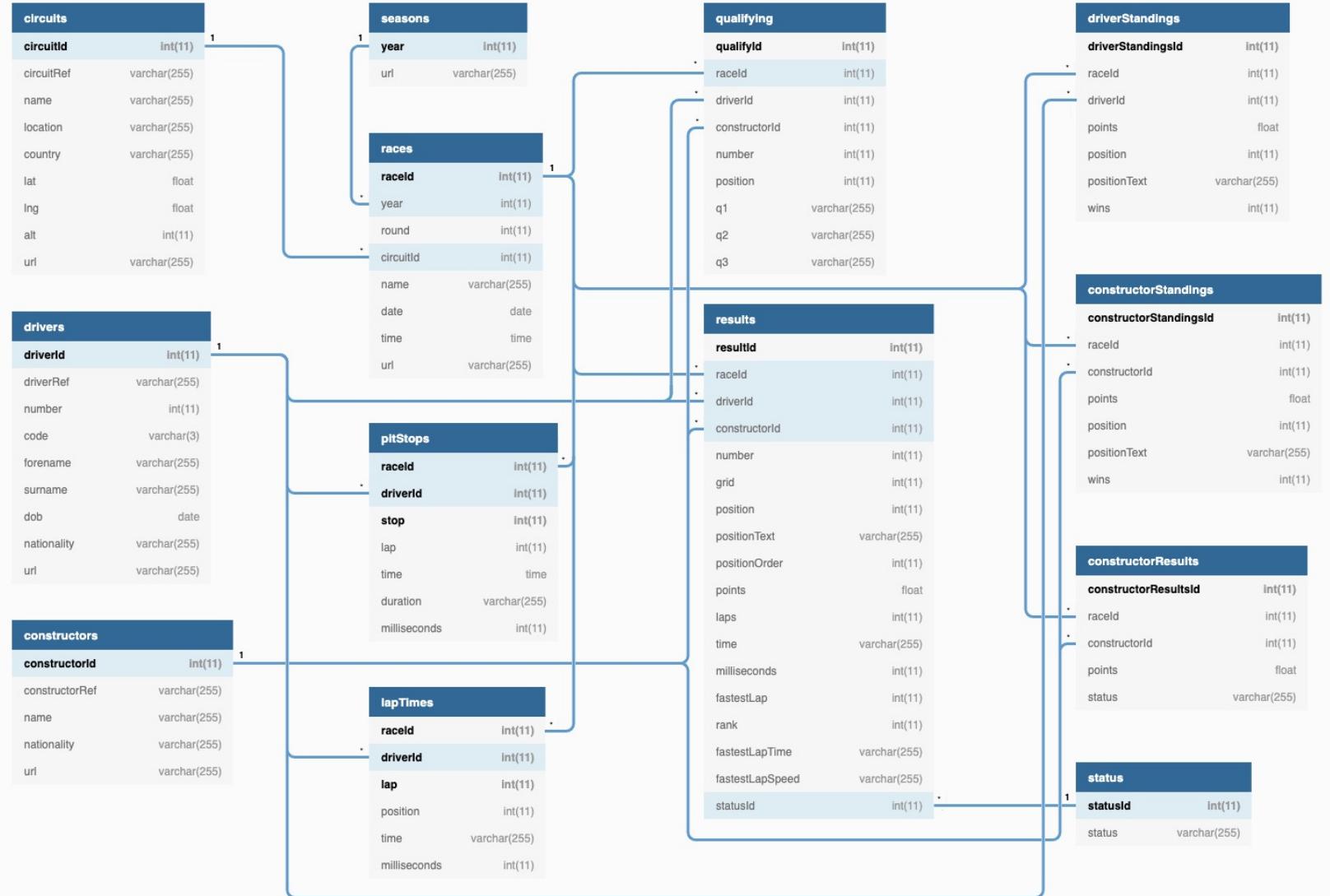
The screenshot shows a Jupyter Notebook interface. On the left, a file browser lists various Jupyter notebooks and files. The file '02.Data\_Preparation\_merging.ipynb' is selected and highlighted with a blue border. A red box highlights the 'Name' column header in the file list. On the right, the notebook content is displayed. The title 'Data Preparation: Merging Datasets' is shown. Below it, a note says: 'Now that we have race, weather, and qualifying race data, we'll merge our sets together to create the dataset Learning algorithms.' A large red callout box with the text 'Use the different CSV Files created in order to create a dataset to understand which features could be important for a Machine Learning Model' points to the code cell [4]. The code cells contain the following Python code:

```
[1]: import warnings  
warnings.filterwarnings("ignore")  
  
[2]: import time  
start = time.time()  
  
[3]: import pandas as pd  
import numpy as np  
  
[4]: races = pd.read_csv('./data/races.csv')  
results = pd.read_csv('./data/results.csv')  
qualifying = pd.read_csv('./data/qualifying.csv')  
driver_standings = pd.read_csv('./data/constructor_standings.csv')  
constructor_standings = pd.read_csv('./data/constructor_standings.csv')  
weather = pd.read_csv('./data/weather.csv')  
  
[5]: print(races.shape)  
races.head()
```

At the bottom, a footer bar includes the number '32', the text 'Copyright © 2021, Oracle and/or its affiliates | Confidential: Internal/Restricted/Highly Restricted [Nov. 2020]', and a small '101848' icon.

# Step 3: Exploratory Data Analysis.

# Data Analysis using ERGAST Data Model



# 03.f1\_analysis\_EDA.ipynb

The screenshot shows a Jupyter Notebook interface. On the left, there's a sidebar with file navigation buttons (+, -, ↑, ↓, C) and a search bar labeled "Filter files by name". Below it is a file tree for the directory "/redbull-analytics-hol / beginners /". A red box highlights the "data\_f1" folder. The main area shows a notebook cell with the title "Formula 1 Grand Prix Exploratory Data Analysis (EDA)" and subtitle "Exploratory based on the drivers, constructors or both". It discusses FIA Rules defined in the last 10 years and mentions important eras of F1. A large red annotation box covers the text about FIA rules and eras, and contains the following text in red:

• We will use the dataset “data\_f1” prepared by Ergast to simplify the Exploratory, but we could use all CSV prepared in the folder “data”

An F1 team's performance is largely dependent on the FIA technical regulation for the season. After the 2013 season, new engine regulations were made (Hybrid era). Mercedes-Benz is the most dominant team since, followed by Red Bull Racing and Scuderia Ferrari. Rules are set to change for 2022, so whatever analysis is made here will not apply for 2022 season and after. Only data after 2010 will be considered in the following analysis.

```
[3]: import warnings  
warnings.filterwarnings("ignore")  
  
[4]: import time  
start = time.time()
```

File Edit View Run Kernel Tabs Settings Help

Terminal Untitled.i Environm Launcher install\_be 05.ML\_M 01\_1.Wea 01\_2.Qua 02.Data\_F 03.Redbu 04.ML\_m 04.ML\_M

Notebook

Environment Explorer Python 3 Notebook Examples Python [conda env:dataexpl\_p37] Python [conda env:root] \*

Console

Python 3 Python [conda env:dataexpl\_p37] Python [conda env:root] \*

Other

Terminal Text File Markdown File Show Contextual Help

**Click on Terminal to open it.**

**Download F1 Dataset :**  
wget [http://ergast.com/downloads/f1db\\_csv.zip](http://ergast.com/downloads/f1db_csv.zip)  
unzip f1db\_csv.zip



# Prepare the environment with extended datasets from Ergast

ORACLE Cloud Redbull Beginners HOL Notebook-0 ? Sign Out

File Edit View Run Kernel Tabs Settings Help

Terminal 3 Terminal 2 Terminal 4 03.f1\_analysis\_EDA.ipynb 01\_0.Formule1\_Data\_Collect

**File Explorer:**

- / formule1-analytics-hol / beginners /
- Name      Last Modified
- bak
- data
- data\_f1** (selected)
- 01\_0.Formule1\_Data\_Collection.ipynb
- 01\_1.Weather\_Data\_Collection.ipynb
- 01\_2.Qualifying\_Data\_Collection.ipynb
- 02.Data\_Preparation\_merging.ipynb
- 03.f1\_analysis\_EDA.ipynb
- 04.ML\_Modelling.ipynb
- 05.ML\_Model\_Serving.ipynb
- lab1\_data\_20210722.tar.gz
- lab1\_notebook\_20210722.tar.gz
- README.md
- v1

**Terminal 2 Output:**

```
01_0.Formule1_Data_Collection.ipynb 02.Data_Preparation_merging.ipynb 04.ML_Modelling.ipynb data lab1_notebook_20210722.tar.gz  
01_1.Weather_Data_Collection.ipynb 03.f1_analysis_EDA.ipynb 05.ML_Model_Serving.ipynb datatst README.md  
01_2.Qualifying_Data_Collection.ipynb 03.Redbull_EDA.ipynb bak lab1_data_20210722.tar.gz v1  
(base) bash-4.2$ pwd  
/home/datascienc/formule1-analytics-hol/beginners  
(base) bash-4.2$ mkdir data_f1  
(base) bash-4.2$ cd data_f1  
(base) bash-4.2$ wget http://ergast.com/downloads/f1db_csv.zip  
--2021-07-26 10:26:52-- http://ergast.com/downloads/f1db_csv.zip  
Resolving ergast.com (ergast.com)... 185.229.22.110  
Connecting to ergast.com (ergast.com)|185.229.22.110|:80... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 5617030 (5.4M) [application/zip]  
Saving to: 'f1db_csv.zip'  
  
100%[=====] 5,617,030 34.7MB/s in 0.2s  
  
2021-07-26 10:26:52 (34.7 MB/s) - 'f1db_csv.zip' saved [5617030/5617030]
```

**Terminal 3 Output:**

```
(base) bash-4.2$ unzip f1db_csv.zip  
Archive: f1db_csv.zip  
inflating: circuits.csv  
inflating: constructor_results.csv  
inflating: constructors.csv  
inflating: constructor_standings.csv  
inflating: drivers.csv  
inflating: driver_standings.csv  
inflating: lap_times.csv  
inflating: pit_stops.csv  
inflating: qualifying.csv  
inflating: races.csv  
inflating: results.csv  
inflating: seasons.csv  
inflating: status.csv  
(base) bash-4.2$
```

**Annotations:**

- A red arrow points from the text "List of files prepared by ERGAST" to the "data\_f1" folder in the File Explorer.
- A red box highlights the "f1db\_csv.zip" download and extraction process in Terminal 2.
- A red box highlights the list of CSV files extracted from the zip archive in Terminal 3.

**List of files prepared by ERGAST**

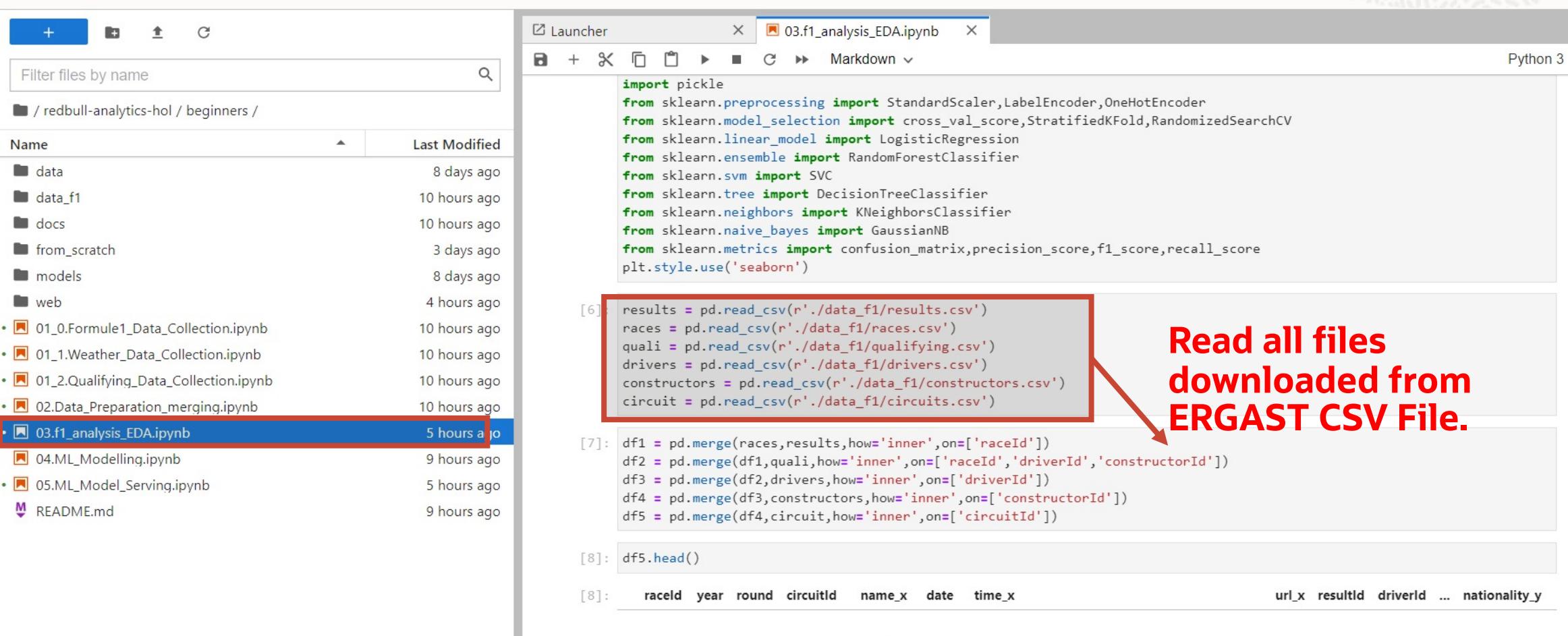
- Pit Stops
- Lap times

# 03.f1\_analysis\_EDA.ipynb

The screenshot shows a Jupyter Notebook interface. On the left, there's a file browser with a sidebar for filtering files by name. The main area displays a notebook titled "Formula 1 Grand Prix Exploratory Data Analysis (EDA)". The title is highlighted with a red border. Below the title, the text "Exploratory based on the drivers, constructors or both" is followed by "FIA Rules defined in the last 10 years". A red arrow points from the text "Execute this notebook" to the first code cell. The code cells contain the following Python code:

```
[3]: import warnings  
warnings.filterwarnings("ignore")  
  
[4]: import time  
start = time.time()
```

# Execute the notebook



The screenshot shows a Jupyter Notebook interface. On the left, a file browser displays a directory structure under '/redbull-analytics-hol / beginners /'. A file named '03.f1\_analysis\_EDA.ipynb' is selected and highlighted with a red box. On the right, the code editor shows a Python script for data analysis. The code imports various machine learning and data manipulation libraries. It then reads five CSV files ('results.csv', 'races.csv', 'qualifying.csv', 'drivers.csv', 'constructors.csv') into pandas DataFrames. These DataFrames are merged sequentially to create a final DataFrame 'df5'. Finally, the head of 'df5' is displayed. A large red arrow points from the explanatory text below to the code block where the CSV files are being read.

```
import pickle
from sklearn.preprocessing import StandardScaler, LabelEncoder, OneHotEncoder
from sklearn.model_selection import cross_val_score, StratifiedKFold, RandomizedSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix, precision_score, f1_score, recall_score
plt.style.use('seaborn')

[6]: results = pd.read_csv(r'./data_f1/results.csv')
races = pd.read_csv(r'./data_f1/races.csv')
quali = pd.read_csv(r'./data_f1/qualifying.csv')
drivers = pd.read_csv(r'./data_f1/drivers.csv')
constructors = pd.read_csv(r'./data_f1/constructors.csv')
circuit = pd.read_csv(r'./data_f1/circuits.csv')

[7]: df1 = pd.merge(races, results, how='inner', on=['raceId'])
df2 = pd.merge(df1, quali, how='inner', on=['raceId', 'driverId', 'constructorId'])
df3 = pd.merge(df2, drivers, how='inner', on=['driverId'])
df4 = pd.merge(df3, constructors, how='inner', on=['constructorId'])
df5 = pd.merge(df4, circuit, how='inner', on=['circuitId'])

[8]: df5.head()

[8]: racelid year round circuitId name_x date time_x
      url_x resultId driverId ... nationality_y
```

Read all files downloaded from ERGAST CSV File.

# Cleaning Dataset and Replace some data

## Cleaning Dataset

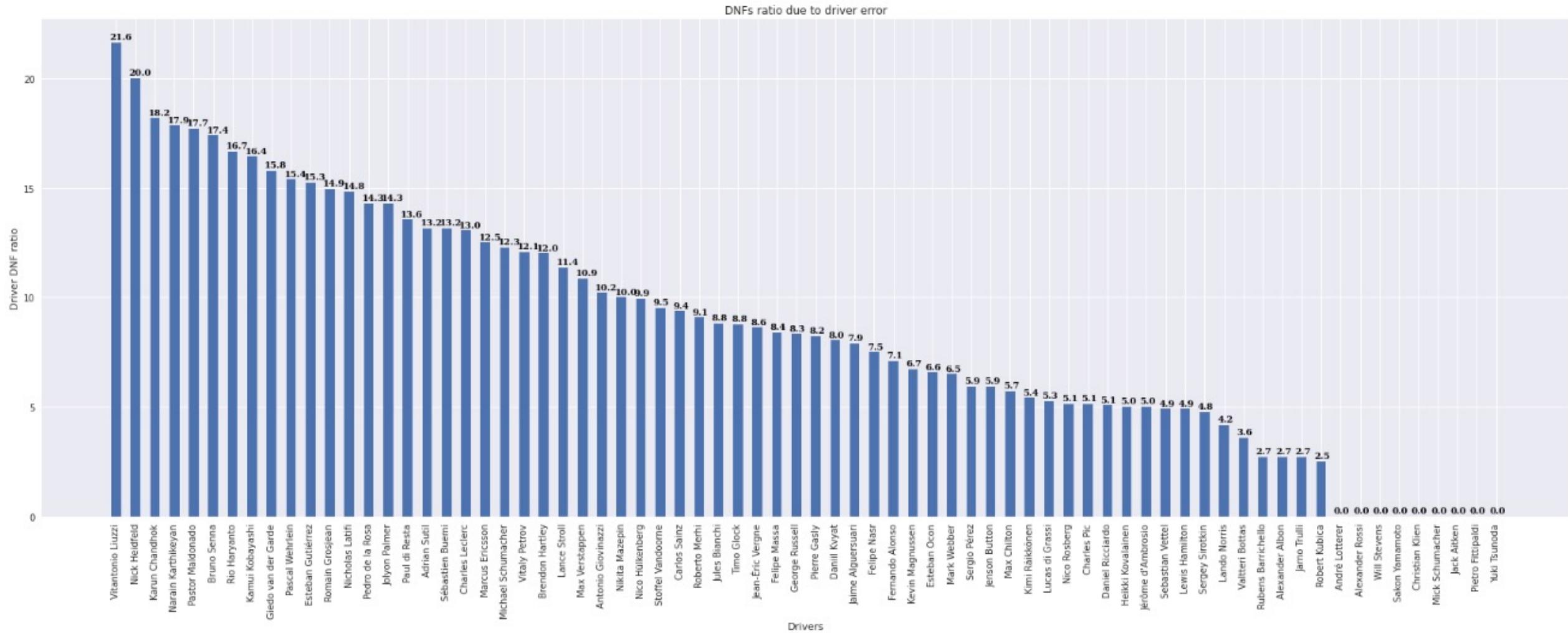
```
: #Some of the constructors changed their name over the year so replacing old names with current name
data['constructor'] = data['constructor'].apply(lambda x: 'Aston Martin' if x=='Force India' else x)
data['constructor'] = data['constructor'].apply(lambda x: 'Aston Martin' if x=='Racing Point' else x)
data['constructor'] = data['constructor'].apply(lambda x: 'Alfa Romeo' if x=='Sauber' else x)
data['constructor'] = data['constructor'].apply(lambda x: 'Alpine F1' if x=='Lotus F1' else x)
data['constructor'] = data['constructor'].apply(lambda x: 'Alpine F1' if x=='Renault' else x)
data['constructor'] = data['constructor'].apply(lambda x: 'AlphaTauri' if x=='Toro Rosso' else x)

: data['driver_nationality'] = data['driver_nationality'].apply(lambda x: str(x)[:3])
data['constructor_nationality'] = data['constructor_nationality'].apply(lambda x: str(x)[:3])
data['country'] = data['country'].apply(lambda x: 'Bri' if x=='UK' else x)
data['country'] = data['country'].apply(lambda x: 'Ame' if x=='USA' else x)
data['country'] = data['country'].apply(lambda x: 'Fre' if x=='Fra' else x)
data['country'] = data['country'].apply(lambda x: str(x)[:3])
data['driver_home'] = data['driver_nationality'] == data['country']
data['constructor_home'] = data['constructor_nationality'] == data['country']
data['driver_home'] = data['driver_home'].apply(lambda x: int(x))
data['constructor_home'] = data['constructor_home'].apply(lambda x: int(x))

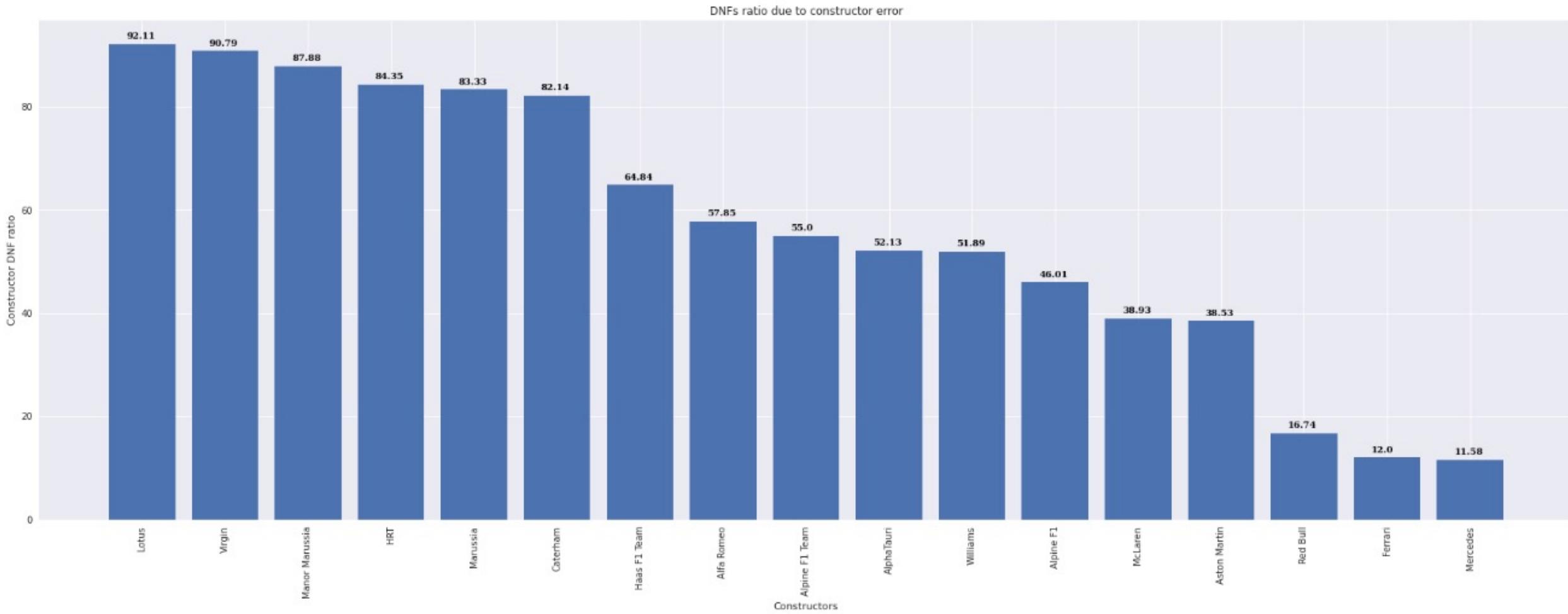
: #reasons for DNF(did not finish)
data['driver_dnf'] = data['statusId'].apply(lambda x: 1 if x in [3,4,20,29,31,41,68,73,81,97,82,104,107,130,137] else 0)
data['constructor_dnf'] = data['statusId'].apply(lambda x: 1 if x not in [3,4,20,29,31,41,68,73,81,97,82,104,107,130,137]
data.drop(['forename','surname'],1,inplace=True)
```



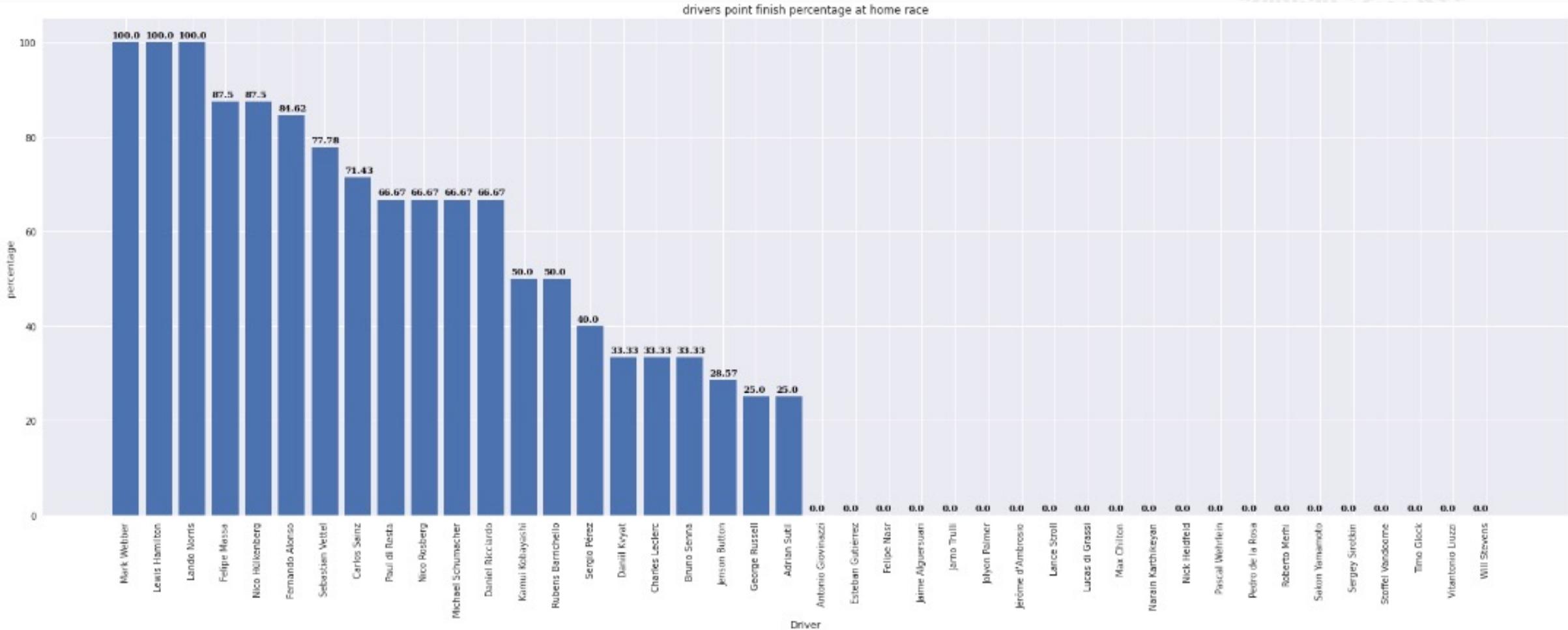
# Check “Did Not Finish” (DNF) Ratio per Driver



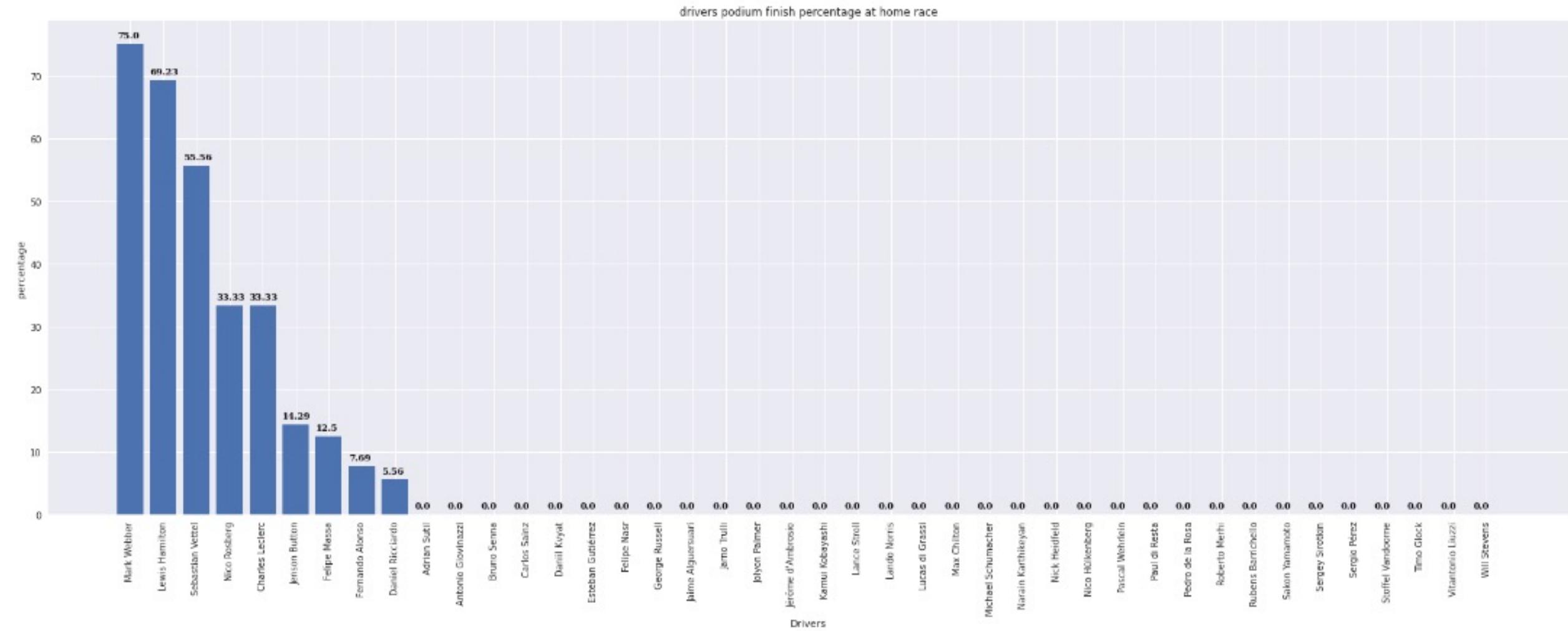
# Check DNF Ratio per Constructor



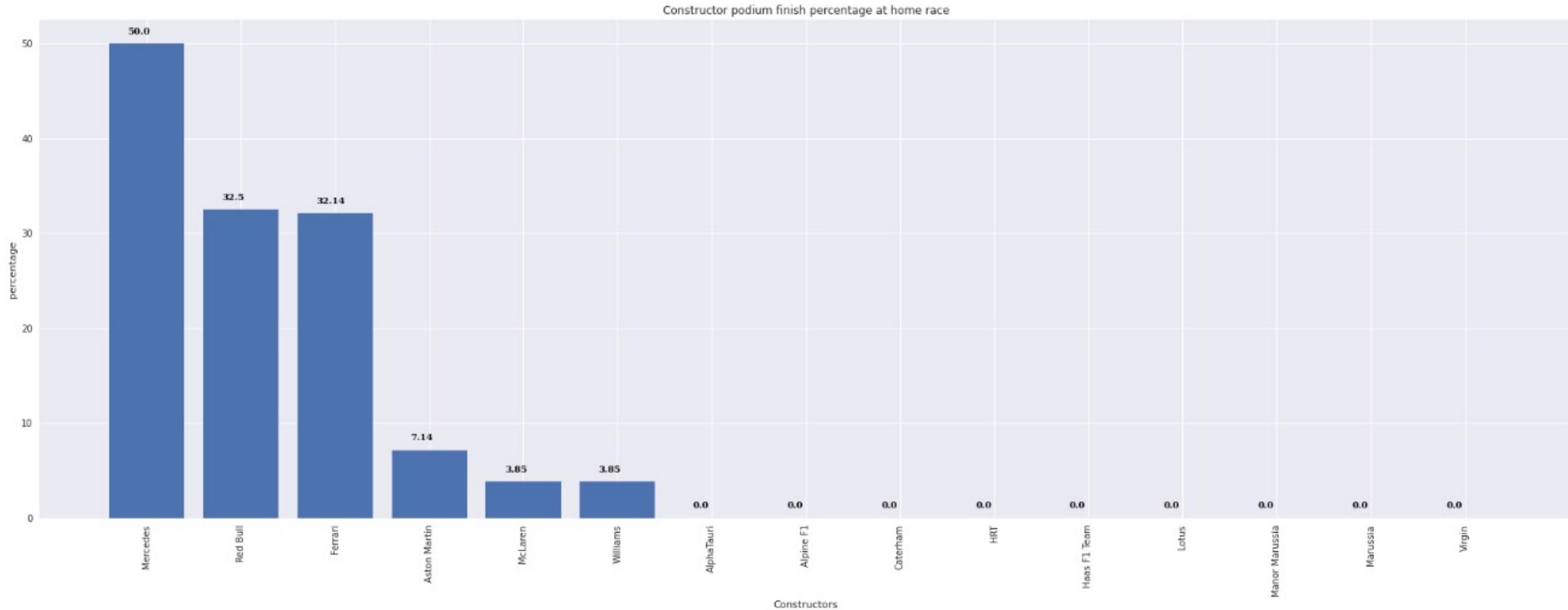
# Check Driver Points Percentage



# Check Driver Podiums Percentage



# Check Constructor Podiums



## Step 4: Execute ML Models.

---

# Hypothesis for Machine Learning Model

---

I want to predict the possible next winner (top 5) for a race.

My hypothesis:

- Depends on Qualifying Position.
- Depends on Constructor / Driver.
- Depends on Constructor Reliability.
- Depends on GP Race / Circuit.
- Depends on Driver Confidence:
  - Do Not Finish
  - Points Number

# Using 6 Classification Methods

---

```
y_dict = {1:'Podium Finish',  
          2:'Points Finish',  
          3:'No Points Finish'  
      }
```

We will create 3 categories:

- The driver has finished in the Podium
- The driver has finished in the points (From 4 to 10 position)
- The driver has finished outside points (after 10)

We can predict each driver for each race can finish in which category ?

## 6 Classification Methods:

- Logistic Regression
- Decision Tree
- Random Forest
- Support Vector Machine
- Gaussian Naives Bayes
- k Nearest Neighbor

# **Execute Notebook “04.ML\_Modelling.ipynb”**

ORACLE Cloud Redbull Beginners HOL Notebook-0

① Sign Out

The screenshot shows a Jupyter Notebook environment with several tabs open. The left sidebar displays a file tree for the directory `/formula1-analytics-hol / beginners /`. The notebook tab `04.ML_Modelling.ipynb` is active, containing the following code:

```
[2]: import numpy as np
import pandas as pd
from sklearn.metrics import confusion_matrix, precision_score
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression, LinearRegression
from sklearn import svm
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.neural_network import MLPClassifier, MLPRegressor

np.set_printoptions(precision=4)

[3]: data = pd.read_csv('./data/f1_df_final.csv')

[4]: data.head()

[4]:
```

The output of the `data.head()` command is shown as a table:

	season	round	weather_warm	weather_cold	weather_dry	weather_wet	weather_cloudy	driver	grid	podium	...	constructor_minardi	constructor_prost	constructor_red_bull	constructor_toyota
0	1983	1	False	False	True	False	False	keke_rosberg	1	15	...	0	0	0	0
1	1983	1	False	False	True	False	False	prost	2	6	...	0	0	0	0
2	1983	1	False	False	True	False	False	tambay	3	4	...	0	0	0	0
3	1983	1	False	False	True	False	False	piquet	4	1	...	0	0	0	0
4	1983	1	False	False	True	False	False	warwick	5	7	...	0	0	0	0

A red box highlights the line `data = pd.read_csv('./data/f1_df_final.csv')`, and a red arrow points from it to the text **Using the Dataset prepared.**

## Step 4: ML Model Exercise for Developer.

---

# Modify final section of Notebook “04.ML\_Modelling.ipynb” to implement a Model using “Driver” and “F1 Team/Constructor” Information.

Build your X dataset with next columns:

- GP\_name
- quali\_pos to predict the classification cluster (1,2,3)
- constructor
- driver
- position
- driver confidence
- constructor\_reliability
- active\_driver
- active\_constructor

• **Build Dataset.** 

[ ]:

Filter the dataset for this Model "Driver + Constructor" all active drivers and constructors

[ ]:

Create Standard Scaler and Label Encoder for the different features in order to have a similar scale for all features

[ ]:

# Build your x dataset

---

```
x =  
data[['GP_name','quali_pos','constructor','driver','position','driver_confidence','constructor_reliability','active_driver','active_constructor']]
```

Build your X dataset with next columns:

- GP\_name
- quali\_pos to predict the classification cluster (1,2,3)
- constructor
- driver
- position
- driver confidence
- constructor\_reliability
- active\_driver
- active\_constructor

```
: x = data[['GP_name','quali_pos','constructor','driver','position','driver_confidence','constructor_reliability','active_driver']]
```

# Modify final section of Notebook “04.ML\_Modelling.ipynb” to implement a Model using “Driver” and “F1 Team/Constructor” Information.

Build your X dataset with next columns:

- GP\_name
- quali\_pos to predict the classification cluster (1,2,3)
- constructor
- driver
- position
- driver confidence
- constructor\_reliability
- active\_driver
- active\_constructor

```
[ ]:
```

Filter the dataset for this Model "Driver + Constructor" all active drivers and constructors

```
[ ]:
```

Create Standard Scaler and Label Encoder for the different features in order to have a similar scale for all features

```
[ ]:
```

## Filter by Active Drivers and Active Constructors

---

```
x = x[x['active_constructor']==1]  
x = x[x['active_driver']==1]
```

```
x = x[(x['active_constructor']==1) & (x['active_driver']==1)]
```

Filter the dataset for this Model "Driver + Constructor" all active drivers and constructors

```
x = x[(x['active_constructor']==1) & (x['active_driver']==1)]
```

# Modify final section of Notebook “04.ML\_Modelling.ipynb” to implement a Model using “Driver” and “F1 Team/Constructor” Information.

Build your X dataset with next columns:

- GP\_name
- quali\_pos to predict the classification cluster (1,2,3)
- constructor
- driver
- position
- driver confidence
- constructor\_reliability
- active\_driver
- active\_constructor

```
[ ]:
```

Filter the dataset for this Model "Driver + Constructor" all active drivers and constructors

```
[ ]:
```

Create Standard Scaler and Label Encoder for the different features in order to have a similar scale for all features

```
[ ]:
```



## Scale all features

---

```
sc = StandardScaler()  
le = LabelEncoder()  
x_c['GP_name'] = le.fit_transform(x_c['GP_name'])  
x_c['constructor'] = le.fit_transform(x_c['constructor'])  
X_c = x_c.drop(['position','active_constructor'],1)  
y_c = x_c['position'].apply(lambda x: position_index(x))
```

Create Standard Scaler and Label Encoder for the different features in order to have a similar scale for all features

```
sc = StandardScaler()  
le = LabelEncoder()  
x['GP_name'] = le.fit_transform(x['GP_name'])  
x['constructor'] = le.fit_transform(x['constructor'])  
x['driver'] = le.fit_transform(x['driver'])  
x['GP_name'] = le.fit_transform(x['GP_name'])
```

# Modify final section of Notebook “04.ML\_Modelling.ipynb” to implement a Model using “Driver” and “F1 Team/Constructor” Information.

In our case, we want to calculate the cluster of final position for each driver using the "position\_index" function

```
[81]: # Implement X, y
```

Applied the same list of ML Algorithms for cross validation of different models

And Store the accuracy Mean Value in order to compare with previous ML Models

```
[72]: mean_results = []
results = []
name = []
```

```
[80]: # cross validation for different models
```

Use the same boxplot plotter used in the previous Models

```
[79]: # Implement boxplot
```

## Comparing The 3 ML Models

Let's see mean score of our three assumptions.

## Prepare Dataset & Predicted variable.

---

```
X = x.drop(['position','active_driver','active_constructor'],1)  
y = x['position'].apply(lambda x: position_index(x))
```

### Prepare the X (Features dataset) and y for predicted value.

In our case, we want to calculate the cluster of final position for each driver using the "position\_index" function

```
: X = x.drop(['position','active_driver','active_constructor'],1)  
y = x['position'].apply(lambda x: position_index(x))
```

# Modify final section of Notebook “04.ML\_Modelling.ipynb” to implement a Model using “Driver” and “F1 Team/Constructor” Information.

In our case, we want to calculate the cluster of final position for each driver using the "position\_index" function

```
[81]: # Implement X, y
```

Applied the same list of ML Algorithms for cross validation of different models

And Store the accuracy Mean Value in order to compare with previous ML Models

```
[72]: mean_results = []
results = []
name = []
```

```
[80]: # cross validation for different models
```

Use the same boxplot plotter used in the previous Models

```
[79]: # Implement boxplot
```

## Comparing The 3 ML Models

Let's see mean score of our three assumptions.

# Execute Cross Validation ML Model.

---

```
#cross validation for different models
models =
[LogisticRegression(),DecisionTreeClassifier(),RandomForestClassifier(),SVC(),GaussianNB(),KNeighborsClassifier()]
names =
['LogisticRegression','DecisionTreeClassifier','RandomForestClassifier','SVC','GaussianNB','KNeighborsClassifier']
model_dict = dict(zip(models,names))
for model in models:
    cv = StratifiedKFold(n_splits=10,random_state=1,shuffle=True)
    result = cross_val_score(model,X,y,cv=cv,scoring='accuracy')
    mean_results.append(result.mean())
    results.append(result)
    name.append(model_dict[model])
print(f'{model_dict[model]} : {result.mean()}')
```

# Modify final section of Notebook “04.ML\_Modelling.ipynb” to implement a Model using “Driver” and “F1 Team/Constructor” Information.

In our case, we want to calculate the cluster of final position for each driver using the "position\_index" function

```
[81]: # Implement X, y
```

Applied the same list of ML Algorithms for cross validation of different models

And Store the accuracy Mean Value in order to compare with previous ML Models

```
[72]: mean_results = []
results = []
name = []
```

```
[80]: # cross validation for different models
```

Use the same boxplot plotter used in the previous Models

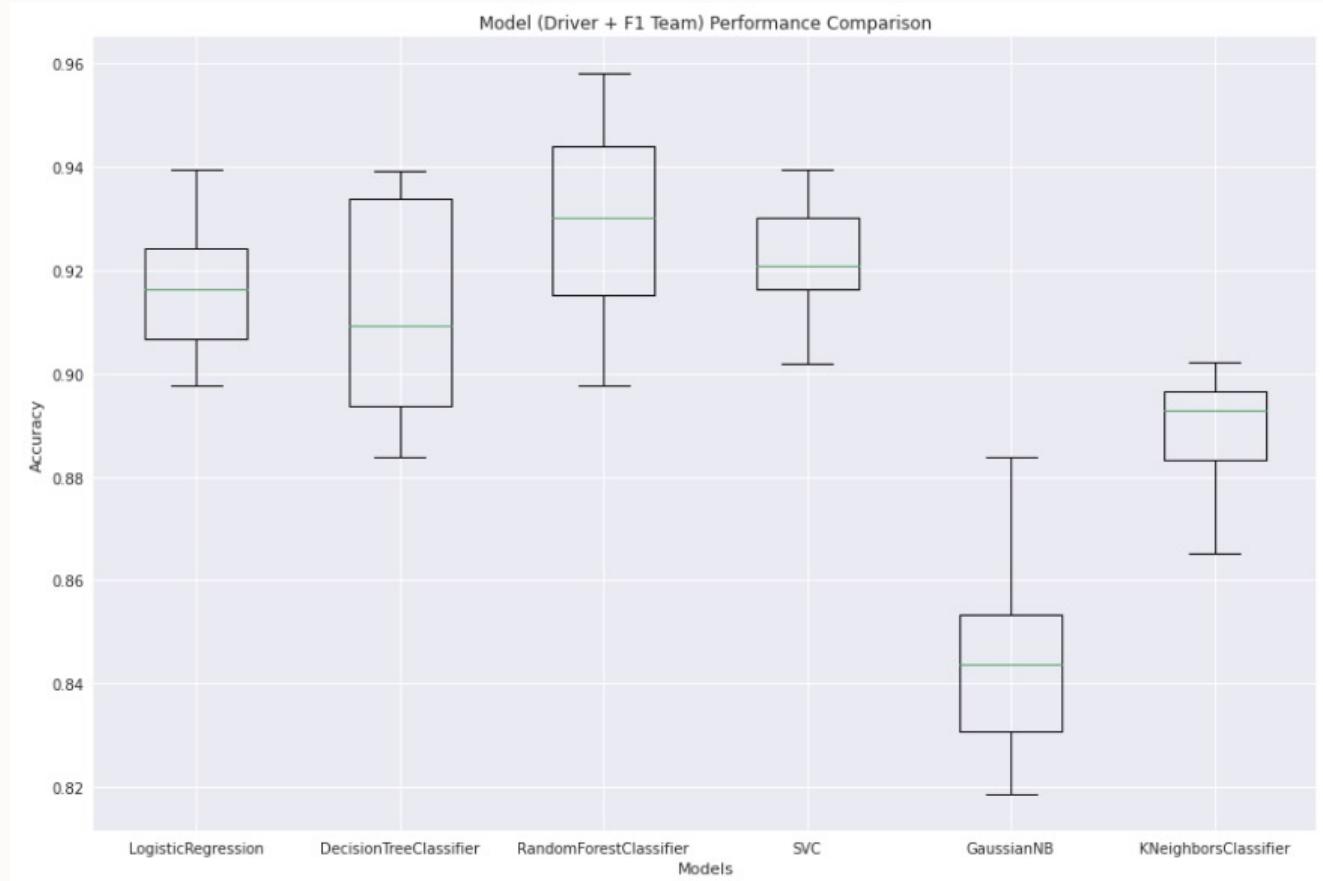
```
[79]: # Implement boxplot
```

## Comparing The 3 ML Models

Let's see mean score of our three assumptions.

# Plot ML Models Accuracy.

```
plt.figure(figsize=(15,10))  
plt.boxplot(x=results,labels=name)  
plt.xlabel('Models')  
plt.ylabel('Accuracy')  
plt.title('Model (Driver + F1 Team) Performance Comparison')  
plt.show()
```



# Modify final section of Notebook “04.ML\_Modelling.ipynb” to implement a Model using “Driver” and “F1 Team/Constructor” Information.

In our case, we want to calculate the cluster of final position for each driver using the "position\_index" function

```
[81]: # Implement X, y
```

Applied the same list of ML Algorithms for cross validation of different models

And Store the accuracy Mean Value in order to compare with previous ML Models

```
[72]: mean_results = []
results = []
name = []
```

```
[80]: # cross validation for different models
```

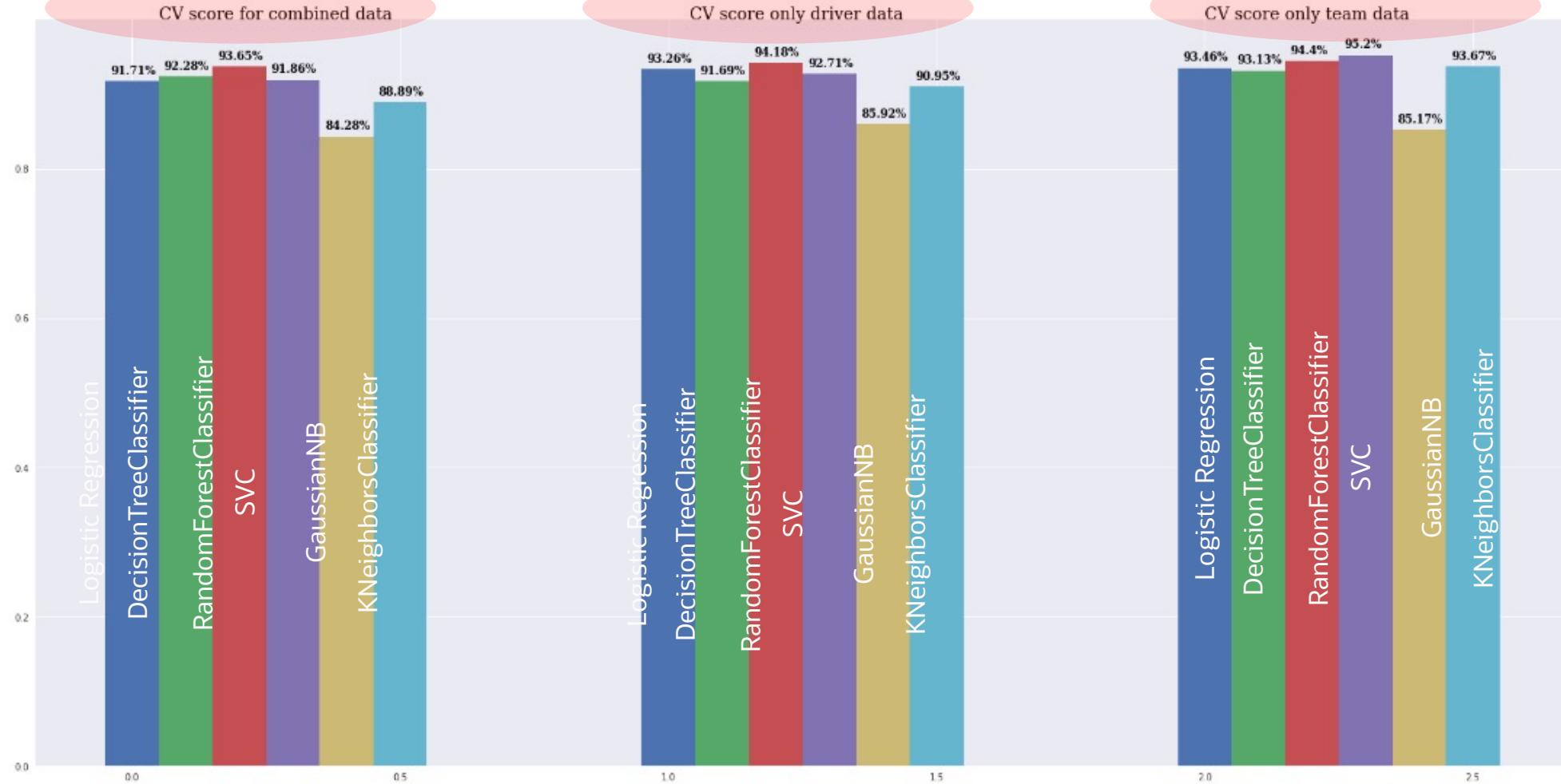
Use the same boxplot plotter used in the previous Models

```
[79]: # Implement boxplot
```

## Comparing The 3 ML Models

Let's see mean score of our three assumptions.

# Comparing the 3 ML Models applied



## Step 5: Execute Model Serving.

---

# 05.ML\_Model\_Serving.ipynb

```
[34]: driver_confidence_dict_str = {}
for key , value in driver_confidence_dict.items():
    # Correct for New Drivers
    if value == 1.0:
        value = 0.10
    driver_confidence_dict_str[key] = np.array([value])
print ("%s: %s" % (key, value))

save_obj(driver_confidence_dict_str, 'driver_dict' )
```

```
Adrian Sutil: 0.868421052631579
Alexander Albon: 0.972972972972973
Alexander Rossi: 0.1
André Lotterer: 0.1
Antonio Giovinazzi: 0.9
Brendon Hartley: 0.88
Bruno Senna: 0.8260869565217391
Carlos Sainz: 0.9069767441860466
Charles Leclerc: 0.8571428571428572
Charles Pic: 0.9487179487179487
Christian Klien: 0.1
Daniel Ricciardo: 0.9494949494949495
Daniil Kvyat: 0.9196428571428571
Esteban Gutiérrez: 0.847457627118644
Esteban Ocon: 0.935064935064935
Felipe Massa: 0.9161290322580645
Felipe Nasr: 0.925
Fernando Alonso: 0.9293478260869565
George Russell: 0.9183673469387755
Giedo van der Garde: 0.8421052631578947
```

Our ML model must use  
Drivers confidence / points  
and Team/Constructor  
reliability / Points

# Using 2021 Dataset to Calculate the Confidence / reliability and Points per Constructor / Driver after each race

```
: drivers = data2021['driver'].unique().tolist()
print(drivers)

['George Russell', 'Pierre Gasly', 'Nikita Mazepin', 'Mick Schumacher', 'Antonio Giovinazzi', 'Kimi Räikkönen', 'Lance Stroll', 'Sebastian Vettel', 'Max Verstappen', 'Sergio Pérez', 'Yuki Tsunoda', 'Lewis Hamilton', 'Esteban Ocon', 'Fernando Alonso', 'Charles Leclerc', 'Carlos Sainz', 'Lando Norris', 'Daniel Ricciardo', 'Nicholas Latifi', 'Valtteri Bottas']

: constructors = data2021['constructor'].unique().tolist()
print(constructors)

['Williams', 'AlphaTauri', 'Haas F1 Team', 'Alfa Romeo', 'Aston Martin', 'Red Bull', 'Mercedes', 'Alpine F1', 'Ferrari', 'McLaren']

: results = {'race': [],
            'driver':[],
            'constructor':[],
            'points':[]}

for race in races:
    for driver in drivers:
        clst = data2021.loc[(data2021['GP_name']==race)&(data2021['driver']==driver)][['constructor'].unique().tolist()]
        plst = data2021.loc[(data2021['GP_name']==race)&(data2021['driver']==driver)][['points'].unique().tolist()]
        for c in clst:
            for p in plst:
                results['race'].append(race)
                results['driver'].append(driver)
                results['constructor'].append(c)
                results['points'].append(p)
```

# Using 2021 Dataset to Calculate the Confidence / reliability and Points per Constructor / Driver after each race

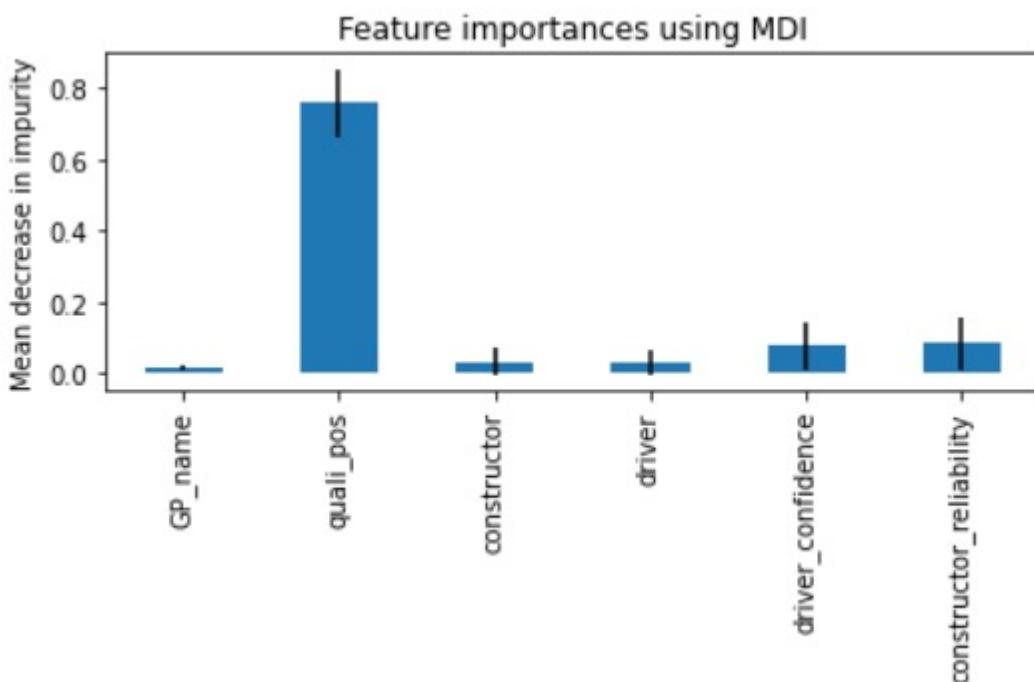
## Generate Qualifying dataset for Predictor

```
: qualif = {'race': [],
            'driver':[],
            'quali_pos':[]}
        }
for race in races:
    for driver in drivers:
        qlst = data2021.loc[(data2021['GP_name']==race)&(data2021['driver']==driver)]['quali_pos'].unique().tolist()
        for q in qlst:
            qualif['race'].append(race)
            qualif['driver'].append(driver)
            qualif['quali_pos'].append(q)
```

```
: qualif2021= pd.DataFrame(qualif)
print (qualif2021)
```

	race	driver	quali_pos
0	Bahrain International Circuit	George Russell	15
1	Bahrain International Circuit	Pierre Gasly	5
2	Bahrain International Circuit	Nikita Mazepin	19
3	Bahrain International Circuit	Mick Schumacher	18
4	Bahrain International Circuit	Antonio Giovinazzi	12
..	...	...	...

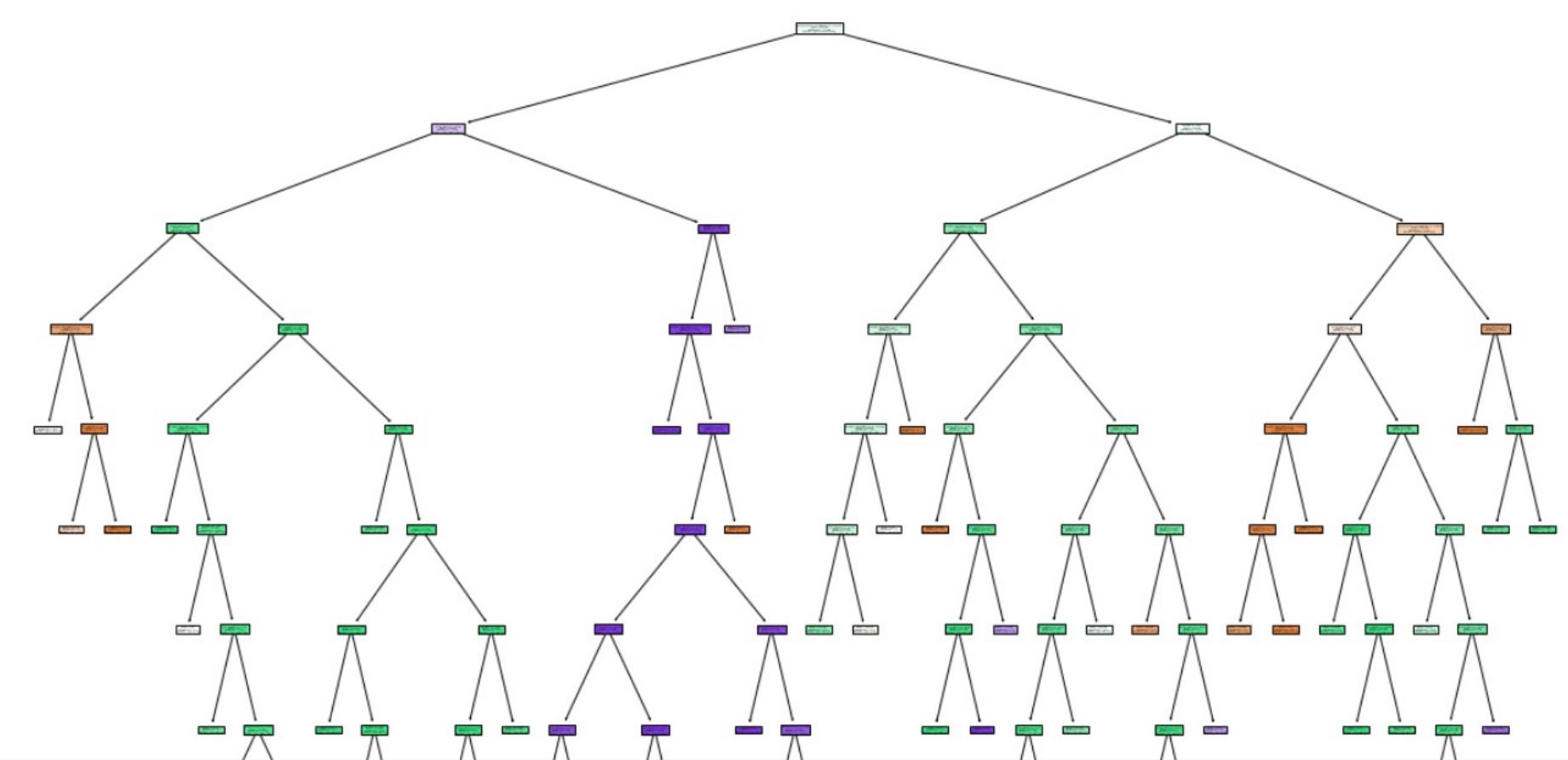
## 05.ML\_Model\_Serving.ipynb



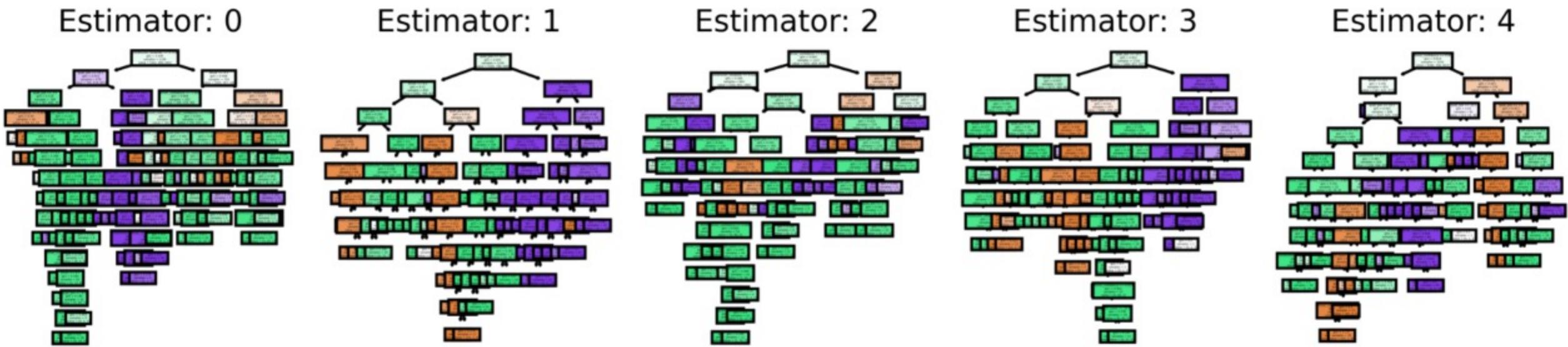
We can detect the **Feature Importance** for our Best ML Model.

The **Qualifying Position** is the main feature for our trained model.

## 05.ML\_Model\_Serving: RF Explanation for First Decision Tree



# 05.ML\_Model\_Serving: RF Explanation for 5 first Decision Trees



## Step 6: Deploy UI Access to predict first 5 drivers.

---

# Write the predictor “predictor.py” using the Model

```
from pickle import load  
import pandas as pd  
import pickle
```

```
from sklearn.preprocessing import LabelEncoder
```

```
driver_dict = pickle.load(open('./driver_dict','rb'))  
constructor_dict = pickle.load(open('./constructor_dict','rb'))  
clf = pickle.load(open('./RandomForestClassifier_light.pkl','rb'))  
data = pd.read_csv('../data_f1/cleaned_data.csv')  
y_dict = {1:'Podium Finish',
```

```
    2:'Points Finish',  
    3:'No Points Finish'  
}
```

```
le_d = LabelEncoder()  
le_d.fit(data[['driver']])  
le_c = LabelEncoder()  
le_c.fit(data[['constructor']])  
le_gp = LabelEncoder()  
le_gp.fit(data[['GP name']])
```

```
def pred(driver,constructor,quali,circuit):
```

```
    gp = le_gp.fit_transform([circuit]).max()  
    quali_pos = quali  
    constructor_enc = le_c.transform([constructor]).max()  
    driver_enc = le_d.transform([driver]).max()  
    driver_confidence = driver_dict[driver].max()
```

```
    constructor_reliability = constructor_dict[constructor].max()  
    prediction = clf.predict([[gp,quali_pos,constructor_enc,driver_enc,driver_confidence,constructor_reliability]]).max()  
    #print(clf.predict([[gp,quali_pos,constructor_enc,driver_enc,driver_confidence,constructor_reliability]]))
```

```
    #return y_dict[prediction]  
    return prediction, driver_confidence, constructor_reliability
```

Load the Trained Model and the dataset saved.

Create a Prediction Method

Call “predict” from Loaded Model

# How to test the Model Predictor

The Model depends on Qualifying Result: Main feature

```
import predictor
import pandas as pd

active_drivers =[['Daniel Ricciardo','McLaren','7'],
['Mick Schumacher','Haas F1 Team','19'],
['Carlos Sainz','Ferrari','9'],
['Valtteri Bottas','Mercedes','3'],
['Lance Stroll','Aston Martin','15'],
['George Russell','Williams','8'],
['Lando Norris','McLaren','6'],
['Sebastian Vettel','Aston Martin','10'],
['Kimi Räikkönen','Alfa Romeo','17'],
['Charles Leclerc','Ferrari','4'],
['Lewis Hamilton','Mercedes','1'],
['Yuki Tsunoda','AlphaTauri','16'],
['Max Verstappen','Red Bull','2'],
['Pierre Gasly','AlphaTauri','12'],
['Fernando Alonso','Alpine F1','11'],
['Sergio Pérez','Red Bull','5'],
['Esteban Ocon','Alpine F1','13'],
['Antonio Giovinazzi','Alfa Romeo','14'],
['Nikita Mazepin','Haas F1 Team','20'],
['Nicholas Latifi','Williams','18']]
```

```
#active_drivers.head()
res = []
for row in active_drivers:
    #for elem in row:
    driver = row[0]
    constructor = row[1]
    quali = row[2]
    circuit = "Silverstone Circuit"
    my_.long_prediction, driver_confidence, constructor_reliability = predictor.pred(driver,constructor,quali,circuit)
```

# Write the app.py using Flask

```
@app.route("/images/<path:path>")
def static_dir(path):
    return send_from_directory("images", path)

@app.route('/')
def get_input():
    return render_template('index.html')

@app.route('/predict',methods=['POST'])
def predict_position():
    circuit = request.form['circuit']
    weather = request.form['weather']

    res = predictor.pred(circuit, weather)
    #print (res)
    df = pd.DataFrame(res, columns = {'Driver','Prediction'})
    return render_template('index.html',tables=[df.to_html(classes='driver')])

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8080, debug=True)
```

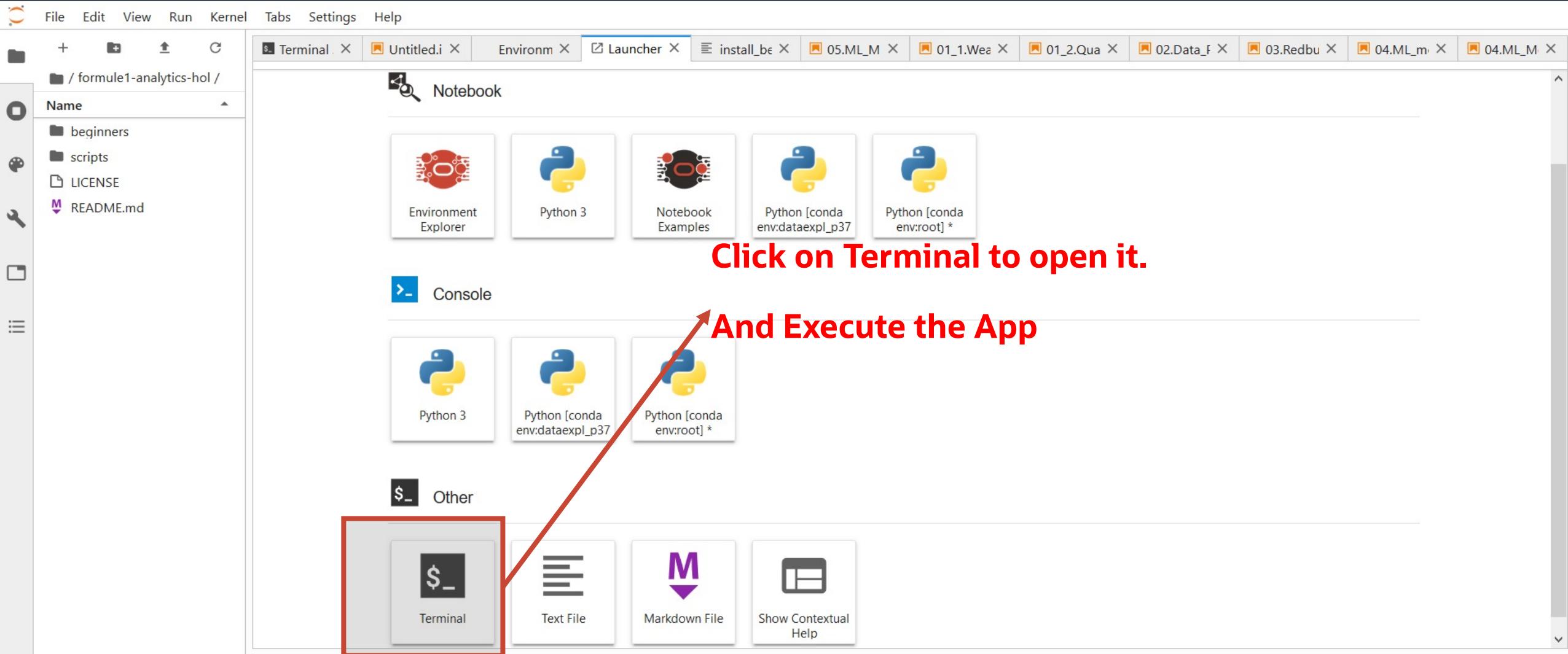
Call Predictor

Open a Server App

ORACLE Cloud

↗ Redbull Beginners HOL Notebook

?



opc@redbull-lab1:~

03.f1\_analysis\_EDA.ipynb

03.f1\_analysis\_EDA\_oli.ipynb

```
#!/bin/bash

#source /home/oracle/.bashrc
# Activate Redbull Environment
source redbullevn/bin/activate

cd /home/opc

if [ "$1" = "start" ]
then
nohup jupyter-lab --ip=0.0.0.0 --port=8001 > ./nohup.log 2>&1 &
echo $! > /home/opc/jupyter.pid
else
kill $(cat /home/opc/jupyter.pid)
fi
(redbullevn) ./launchapp.sh start
(redbullevn) tail -f nohup_app.log
* Environment: production

WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: on
/home/opc/redbullevn/lib64/python3.6/site-packages/scikit-learn/utils/validation.py:63: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
    return f(*args, **kwargs)
* Running on all addresses.
WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://10.0.0.224:8080/ (Press CTRL+C to quit)
* Restarting with stat
/home/opc/redbullevn/lib64/python3.6/site-packages/scikit-learn/utils/validation.py:63: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
    return f(*args, **kwargs)
* Debugger is active!
* Debugger PIN: 112-951-674
```

Execute next command to start:

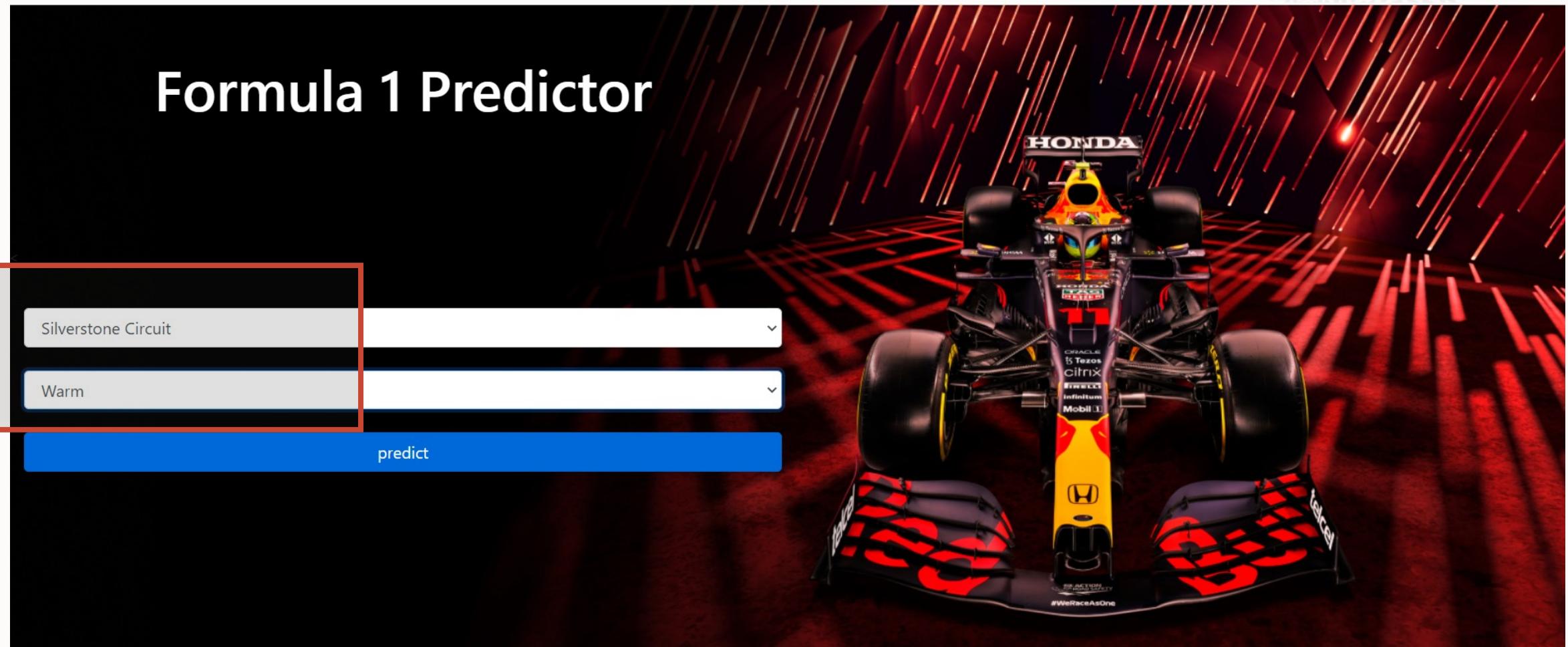
**./launchapp.sh start**

To stop:

**./launchapp.sh stop**

**Test using your VM IP: <http://<xxxx.xxx.xxx.xxx>:8080/>**

# Formula 1 Predictor

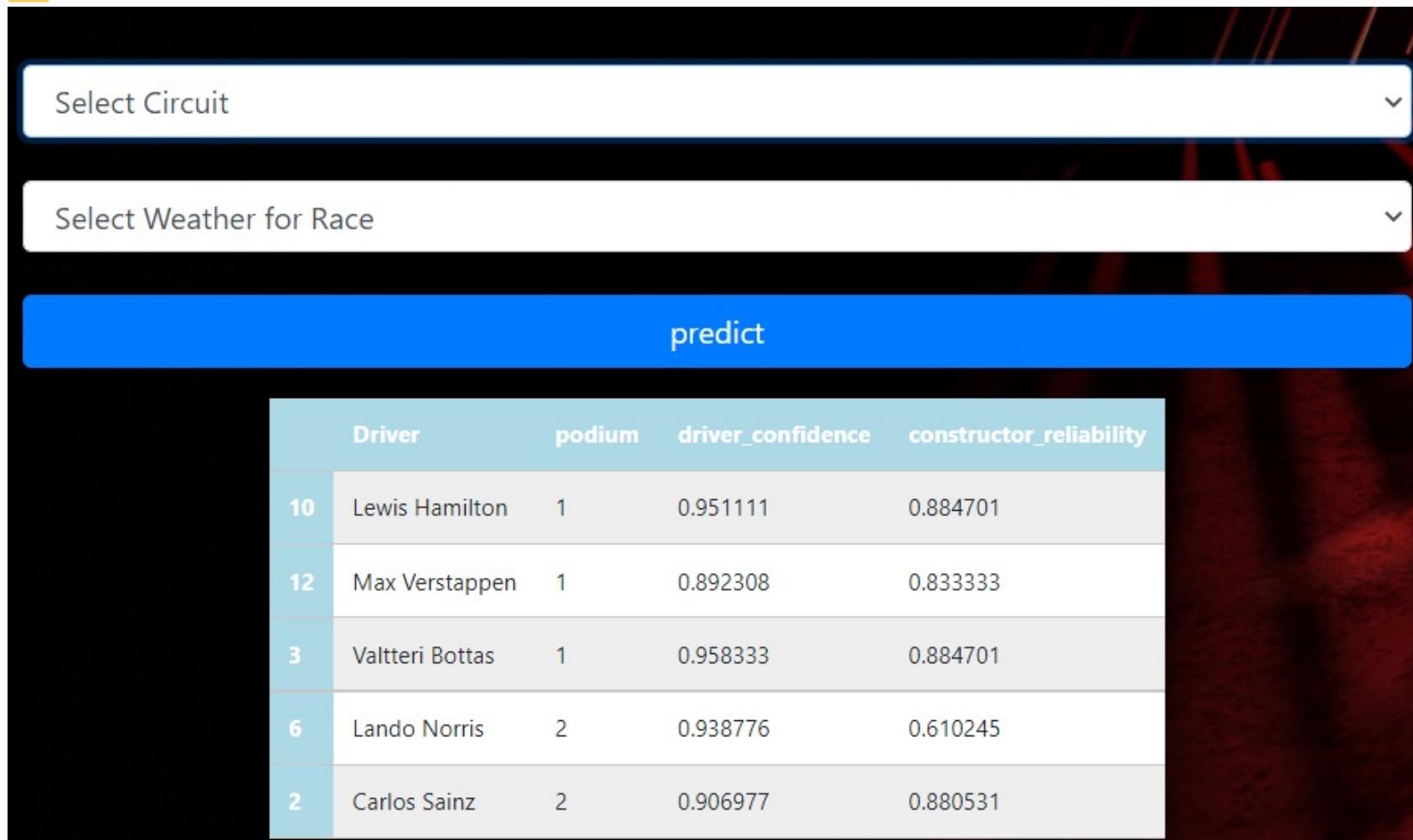


The interface features a sidebar on the left with two dropdown menus:

- Silverstone Circuit
- Warm

Below the dropdowns is a blue button labeled "predict".

# Prediction Results to win for the Best 5 drivers : Silverstone 2021



Select Circuit

Select Weather for Race

**predict**

	Driver	podium	driver_confidence	constructor_reliability
10	Lewis Hamilton	1	0.951111	0.884701
12	Max Verstappen	1	0.892308	0.833333
3	Valtteri Bottas	1	0.958333	0.884701
6	Lando Norris	2	0.938776	0.610245
2	Carlos Sainz	2	0.906977	0.880531

## Other Hypothesis to perform my Machine Learning Model

---

I want to predict the possible next winner (top 5) for a race.

My hypothesis:

- Depends on Qualifying Position.
- Depends on Team / Driver.
- Depends on Team Reliability.
- Depends on GP Race / Circuit.
- Depends on Driver Confidence:
  - Do Not Finish
  - Points Number
- Depends on Race Weather ?
- Depends on Tyres Strategy ?
- Depends on Engine ?

# Thank You

---

**DevRel Team**



A person's torso and head are visible from the side, wearing a patterned shirt with vertical stripes in orange, yellow, and blue.

# ORACLE

O