ORACLE

Red Bull Racing Honda Beginner Workshop

Red Bull Racing Honda Developer Session

August 2021

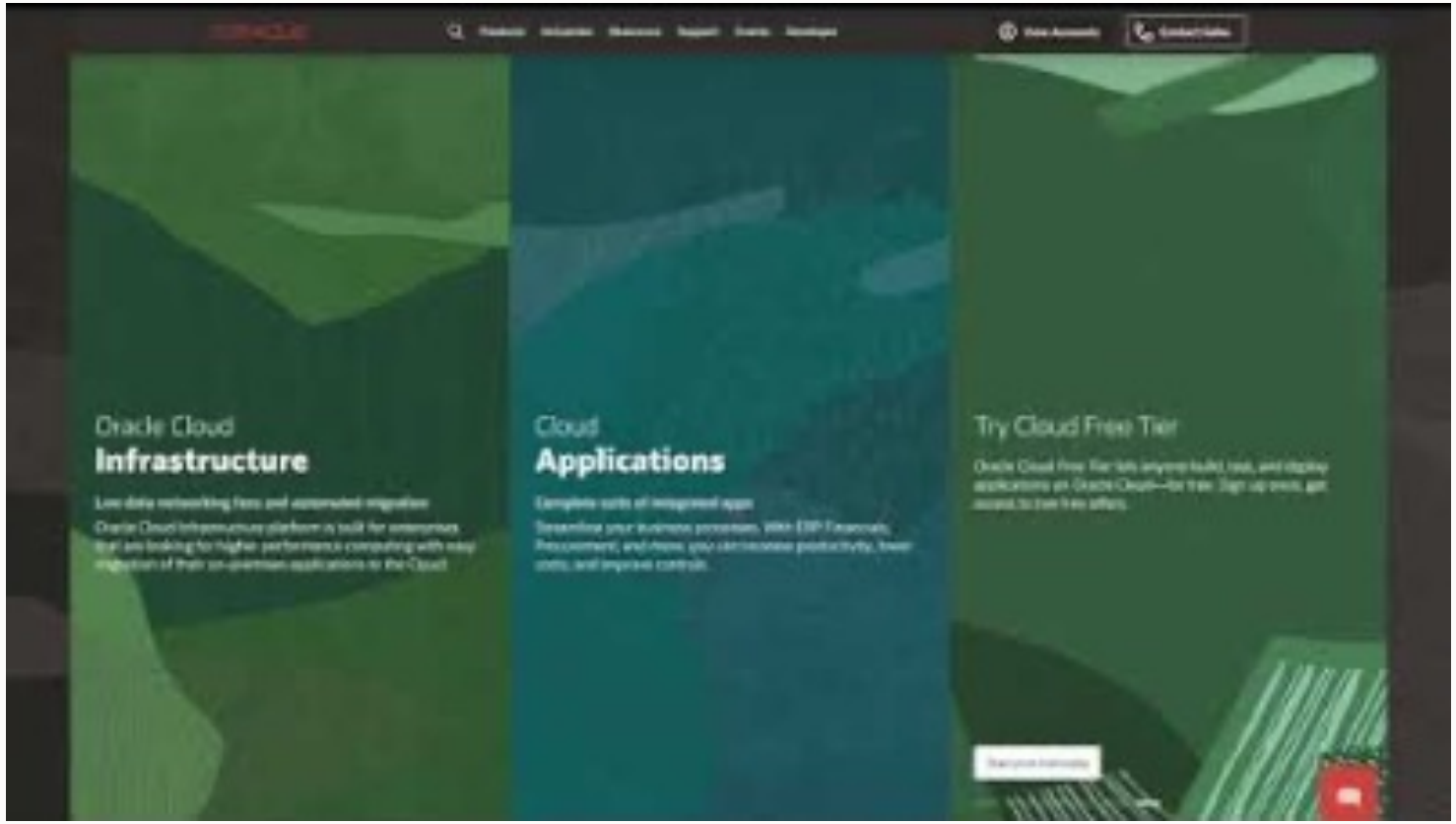
Resources

- Slack group:
  https://bit.ly/3ixPQWT
- Lab guide:

  https://bit.ly/2U5FPGW

Provisioning your Free Tier Account

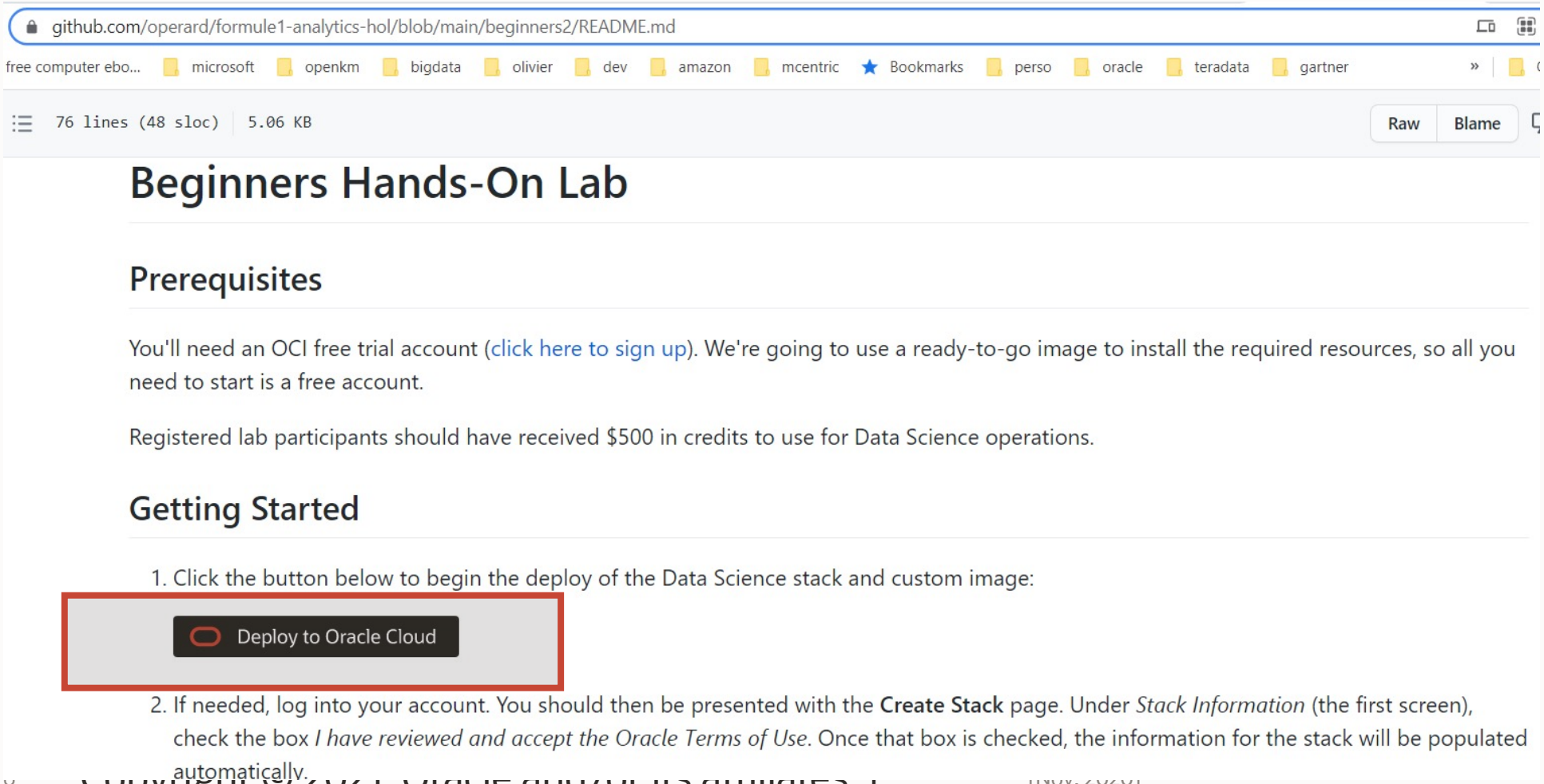
How do I sign up for a free-tier OCI account?



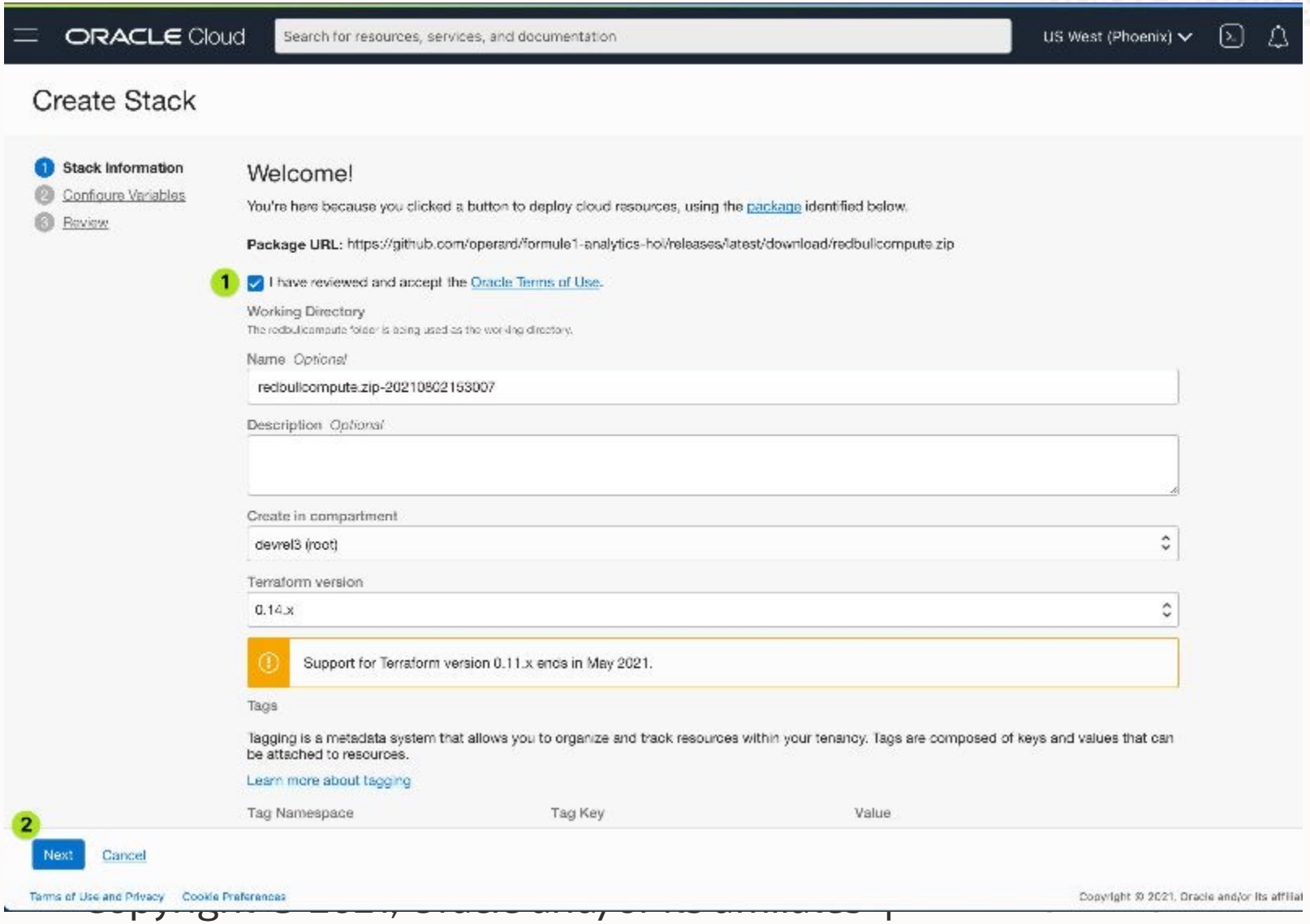
Deploy JupyterLab environment

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

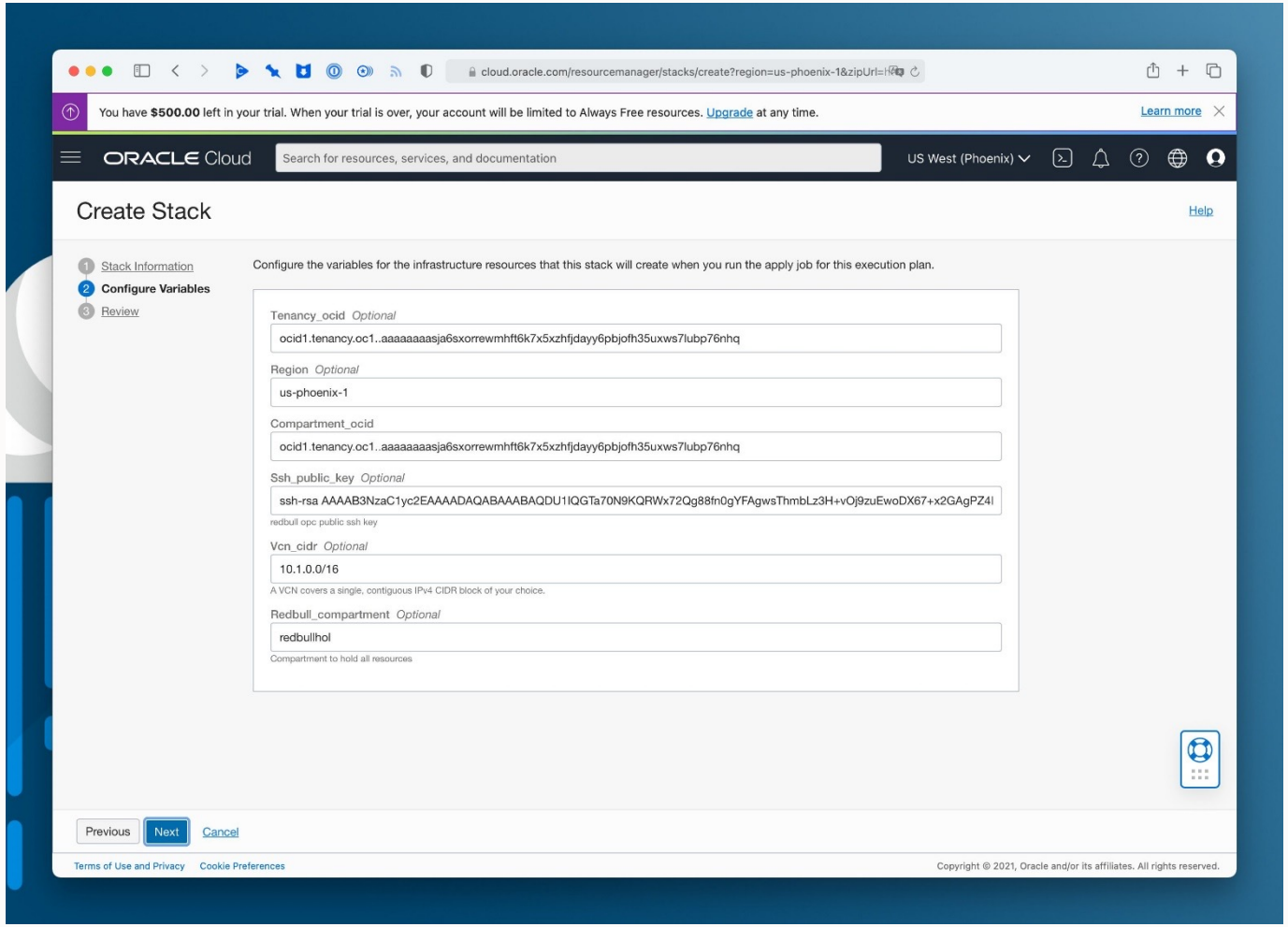
https://github.com/oracle-devrel/redbull-analytics-hol/tree/main/beginners



Step 1: Using Resource Manager on Oracle Cloud (Terraform script)

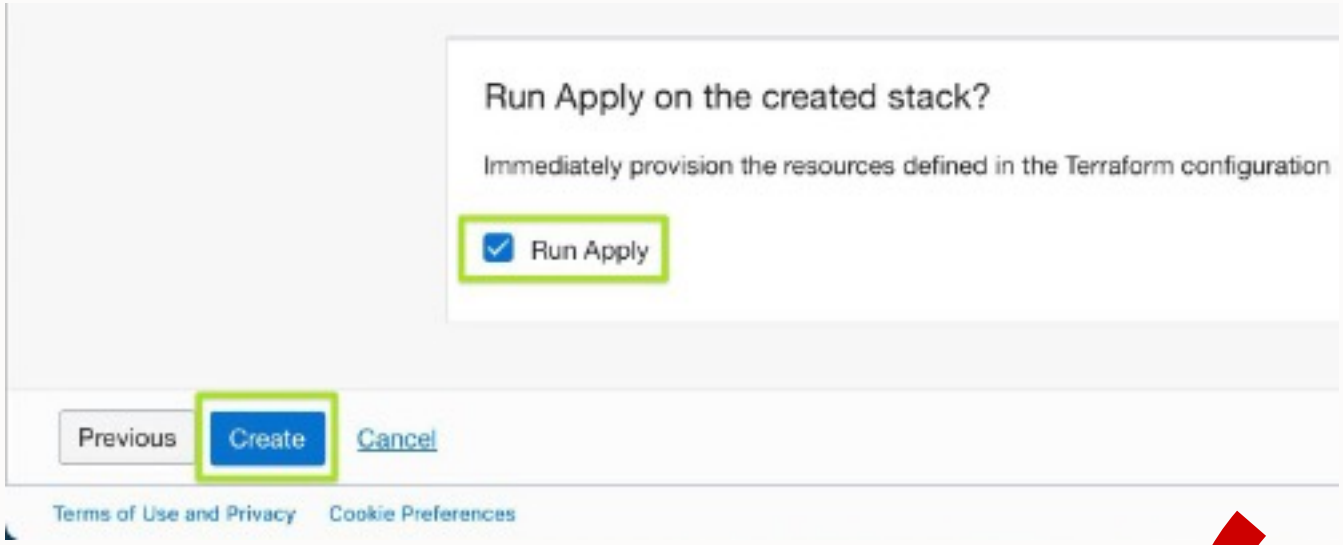


Step 2: Using Resource Manager on Oracle Cloud (Terraform script)

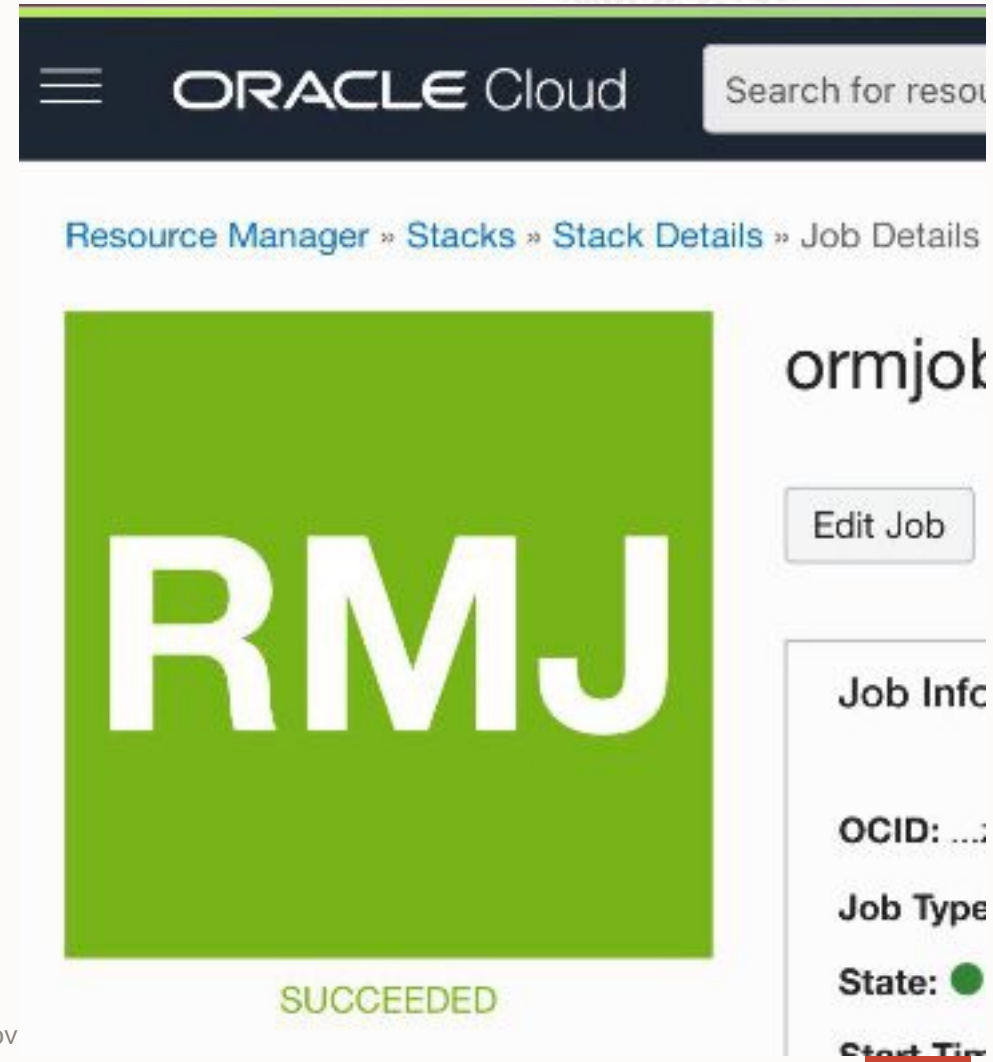


[Nov. 2020]

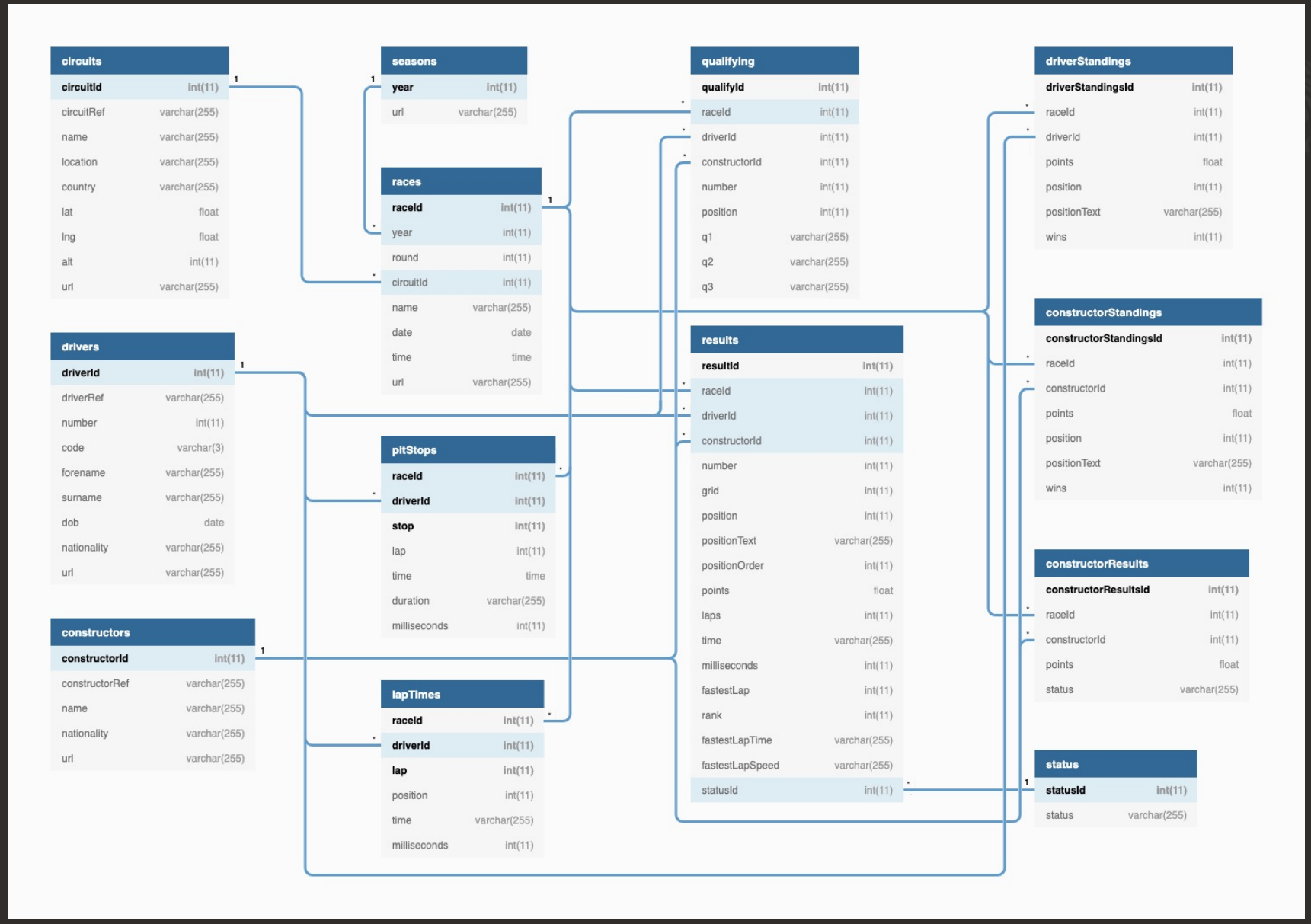
Final step:
Using Resource Manager on Oracle Cloud (Terraform script: Run Apply)



- **10 min to deploy**



1 Data Collection



2 Data Preparation

```
df_dum.shape
```

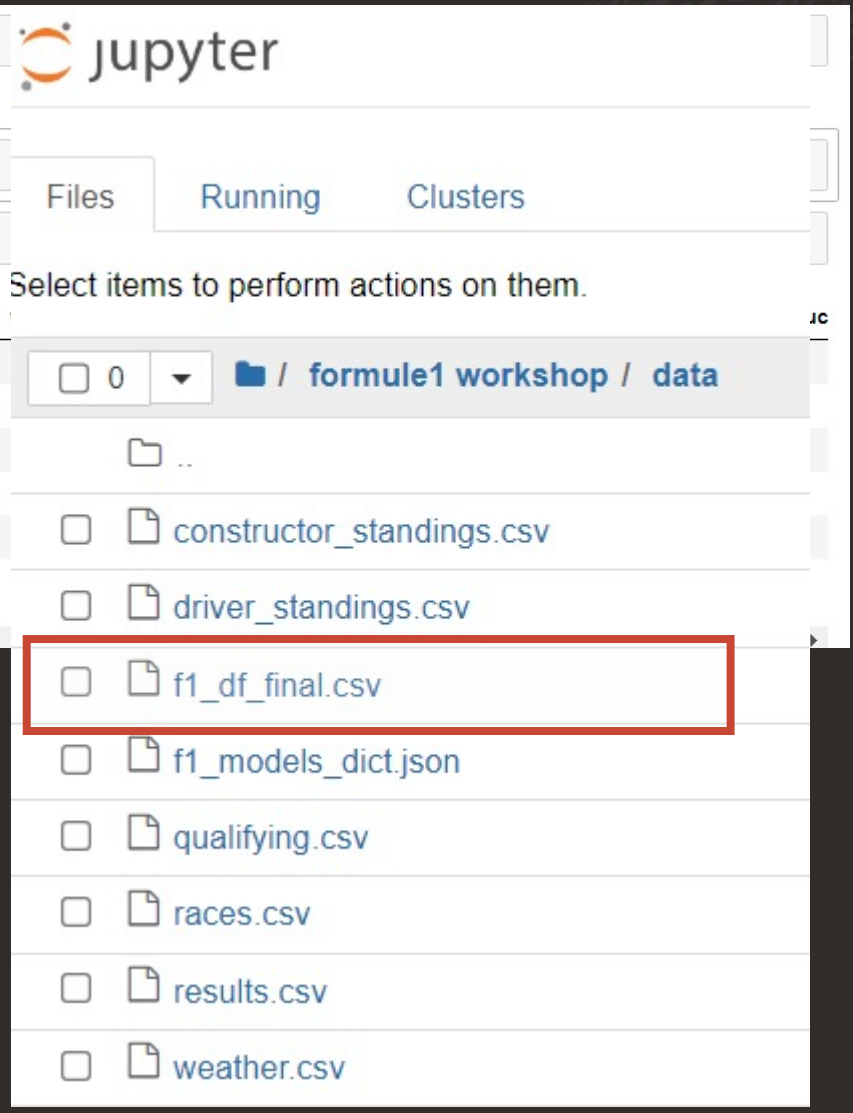
```
(14272, 100)
```

```
df_dum.to_csv('./data/f1_df_final.csv', index = False)
```

```
df_dum.head()
```

	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

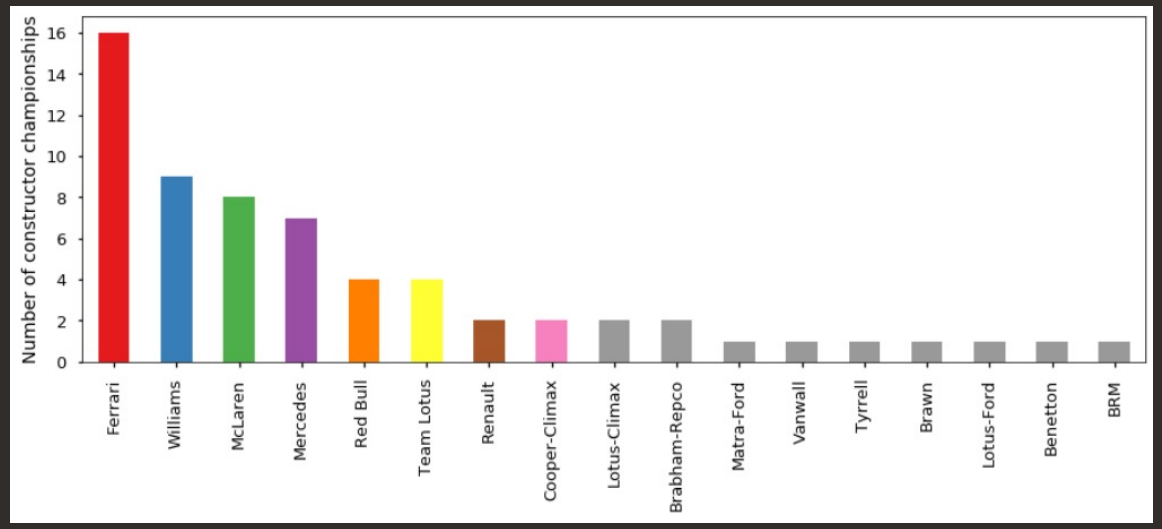
5 rows × 100 columns

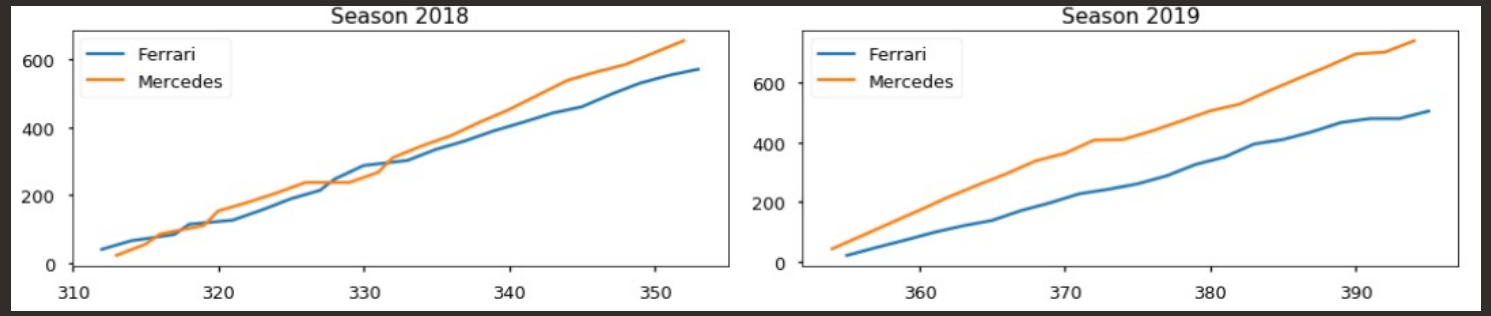


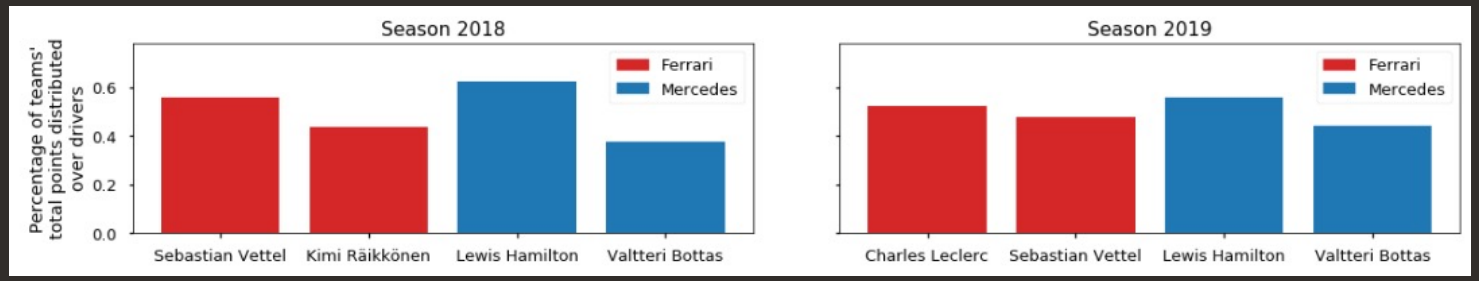
Build a F1 merged Dataset to discover the best ML model:

- 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...

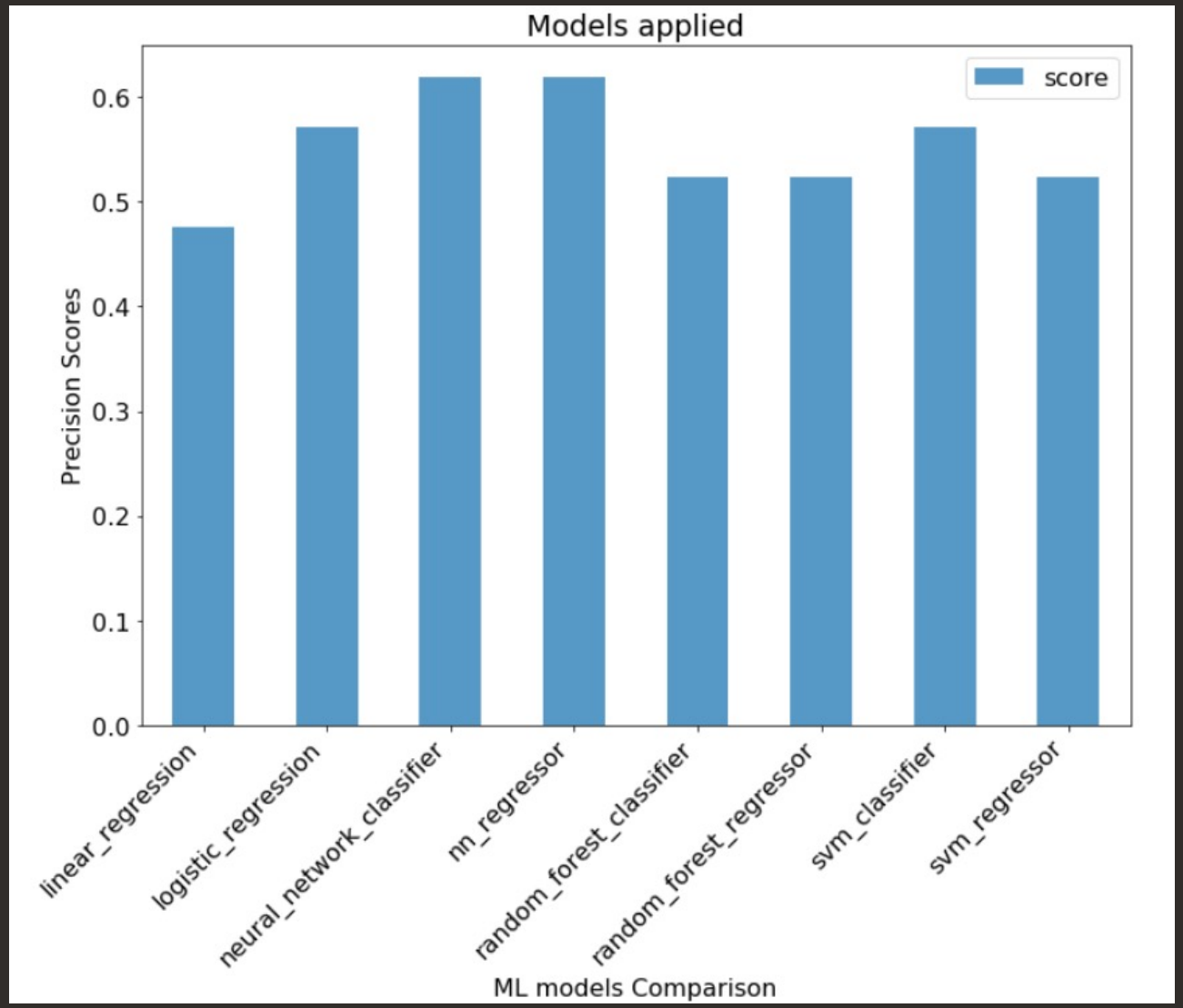
4 ML Models Implementation

```
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

- Etc.

Call the Winner Prediction Algorithm:

- Circuit
- Weather
- Qualifying Times

At a glance: today's steps

1 Data Collection

Where can I download F1 Datasets?

2 Data Preparation

Merge all F1 datasets.

3 Exploratory Data Analysis

Search insights in the Datasets.

4 Implement ML Models

Develop different ML models, evaluate best model and tune.

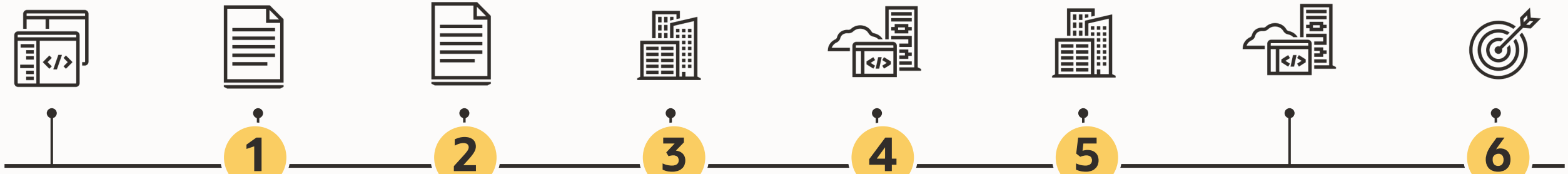
5 Model Serving

Deploy the best trained model as a REST API.

6 UI Access

Develop Web UI Access to predict the next winner.

Red Bull Racing Honda Developer Journey Map



Deploy JupyterLab environment

1 Data Collection

01_0.Formule1_Data_Collection.ipynb +
01_1.Weather_Data_Collection.ipynb +
01_2.Qualifying_Data_Collection.ipynb

2 Data Preparation

02.Data_Preparation_merging.ipynb

3 Exploratory Data Analysis

03.f1_analysis_EDA.ipynb

4 Implementing ML Models

04.ML_Modelling.ipynb

5 Model Serving

05.ML_Model_Serving.ipynb

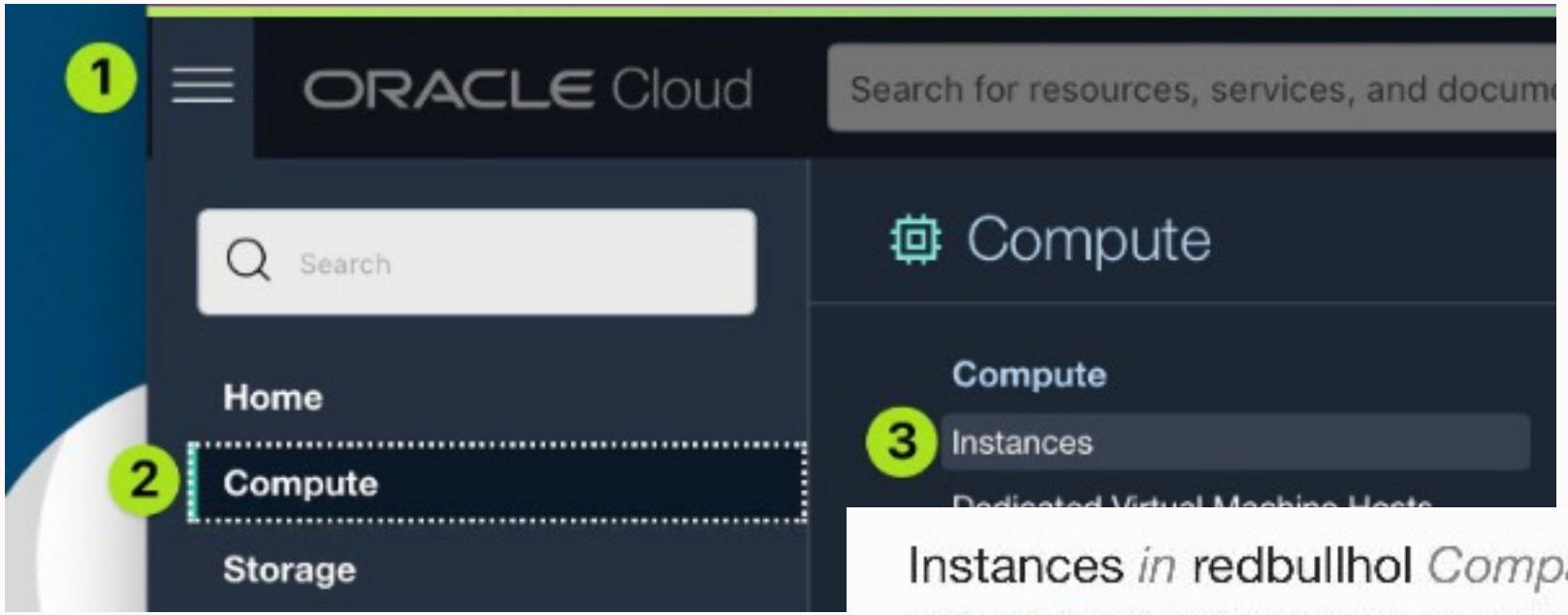
Deploying Model using Flask

app.py + predictor.py
Testpredictor.py

6 UI Access

app.py + Flask + index.html

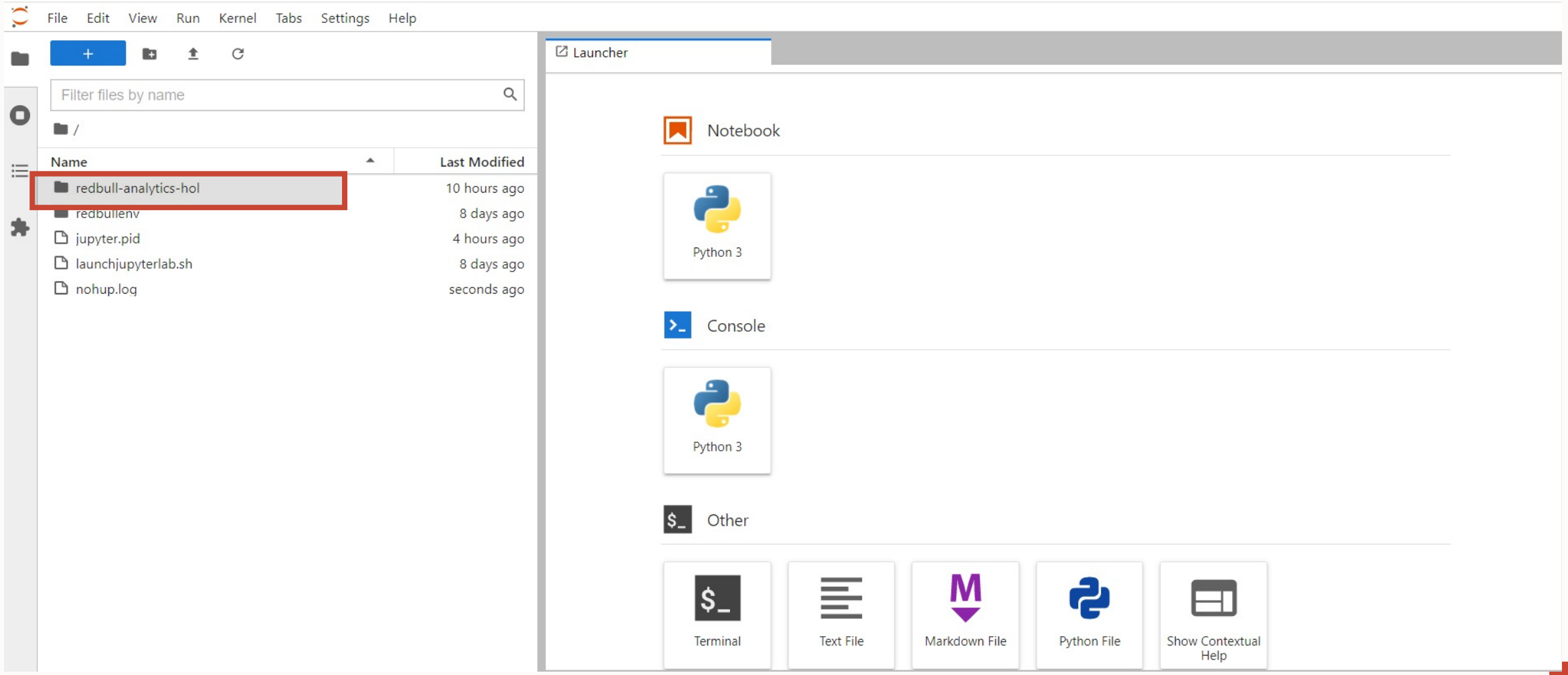
Access to your compartment and Lab Image deployed.



Access to JupyterLab on your image using the IP

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

JupyterLab Access

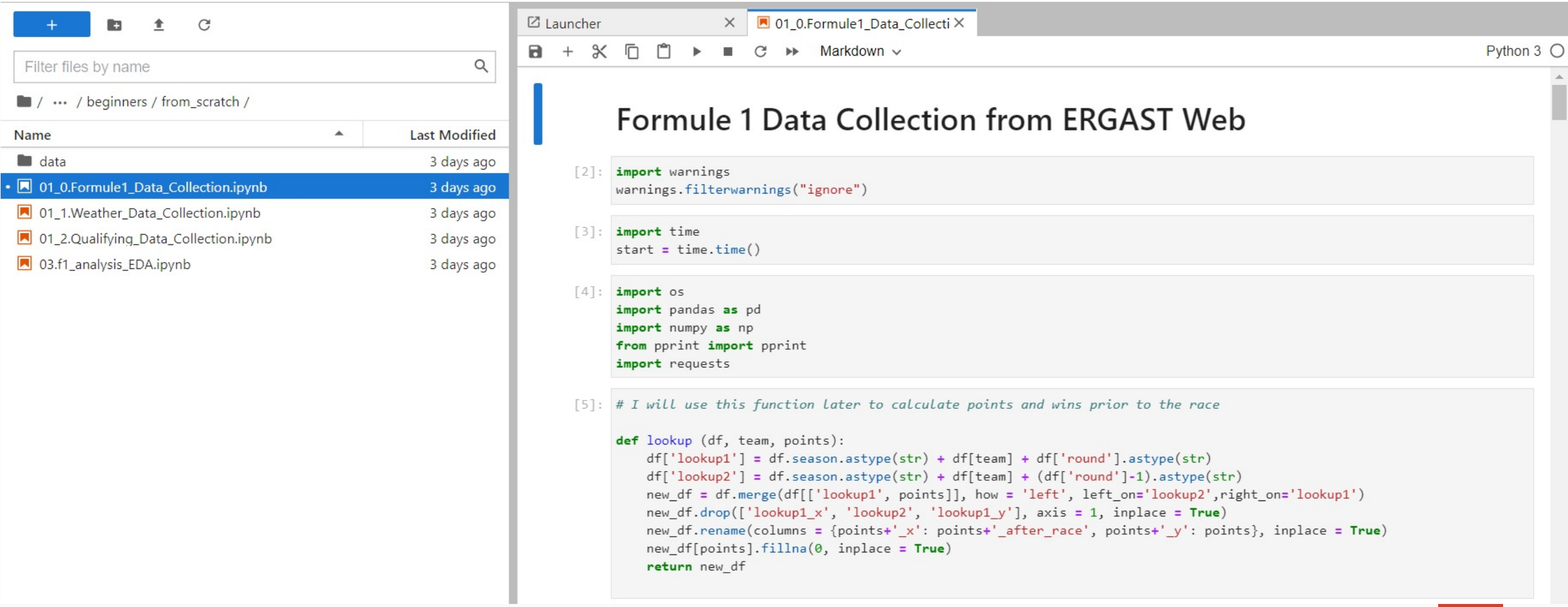


Step 1: Execute 3 notebooks to download Data and create your own CSV Files.

Use Notebooks Version in “from_scratch” Directory.

30-40 min (Offline Execution)

Download Dataset from ERGAST Web



Extracting Races & results from ERGAST Web

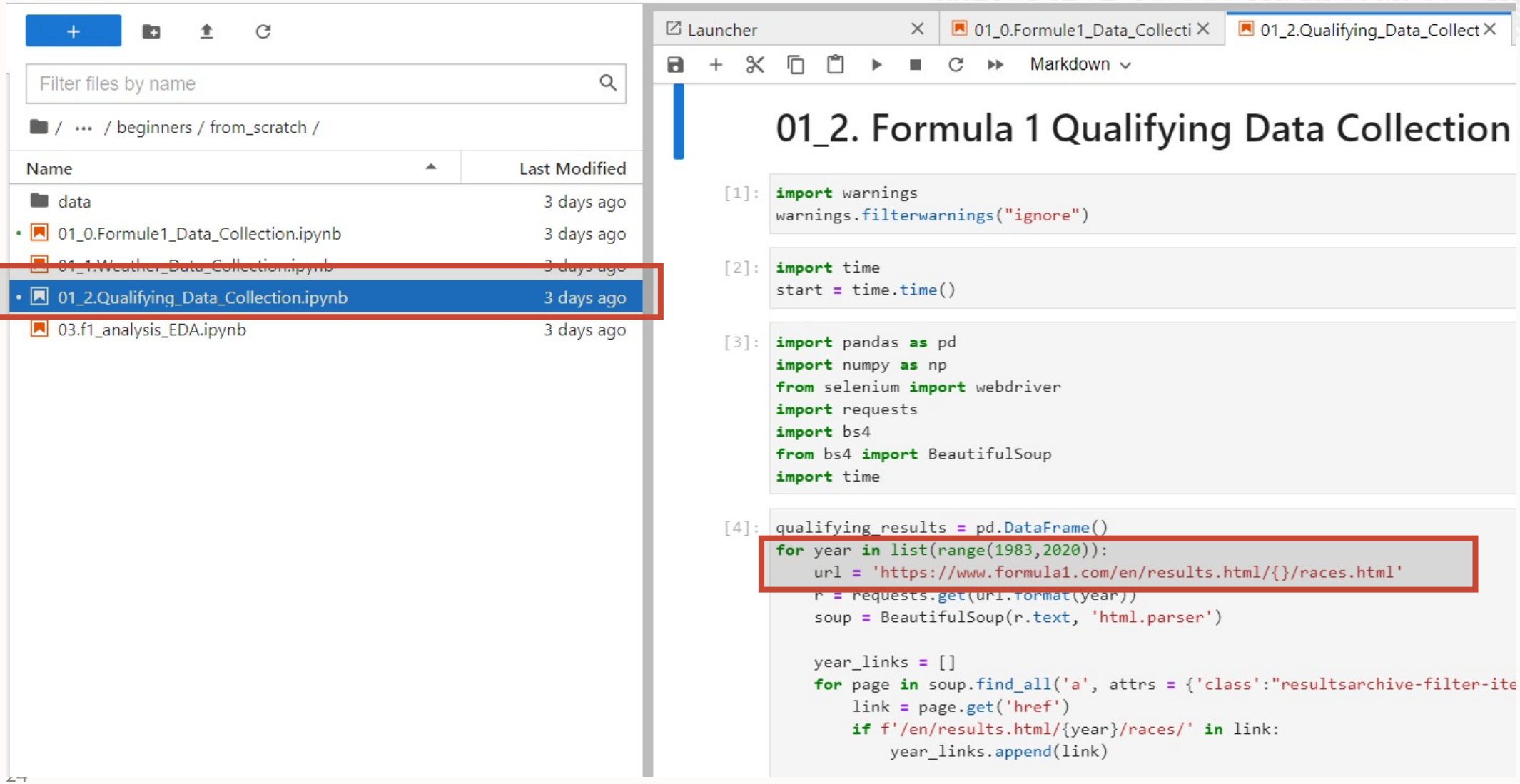
Races

```
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'
```

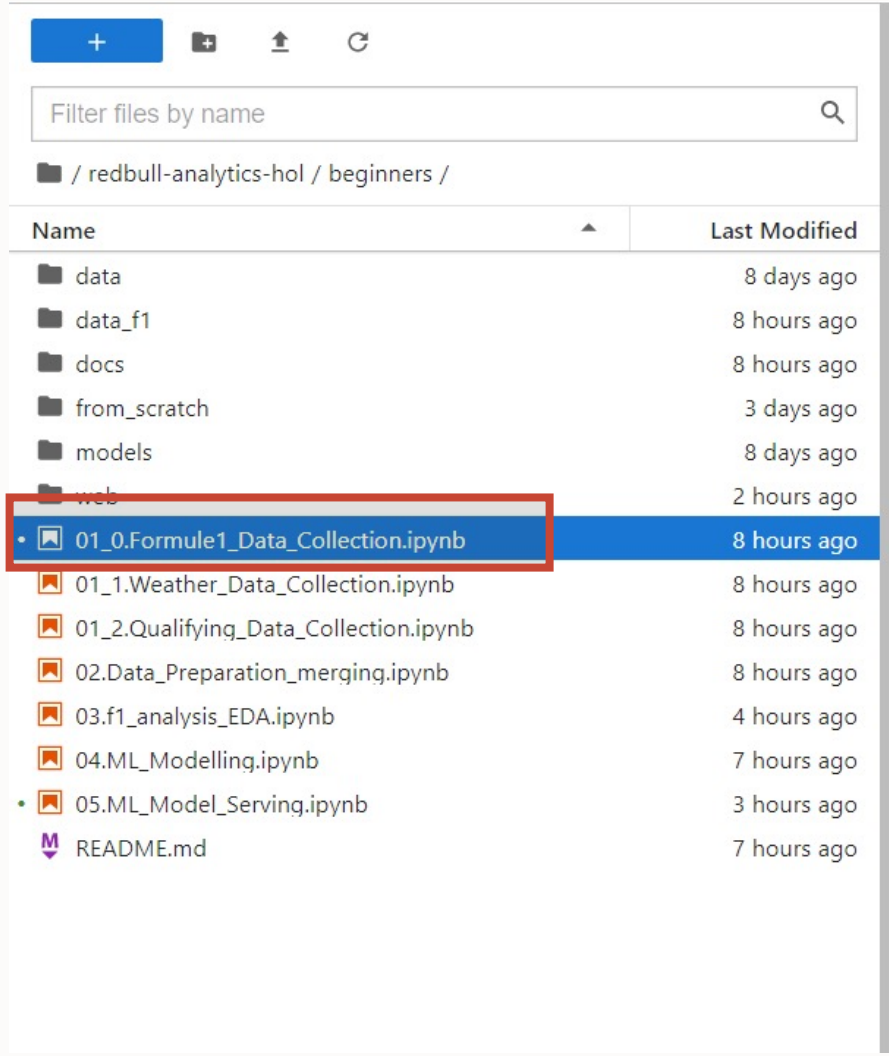
Extracting the Qualifying Dataset from Formula 1

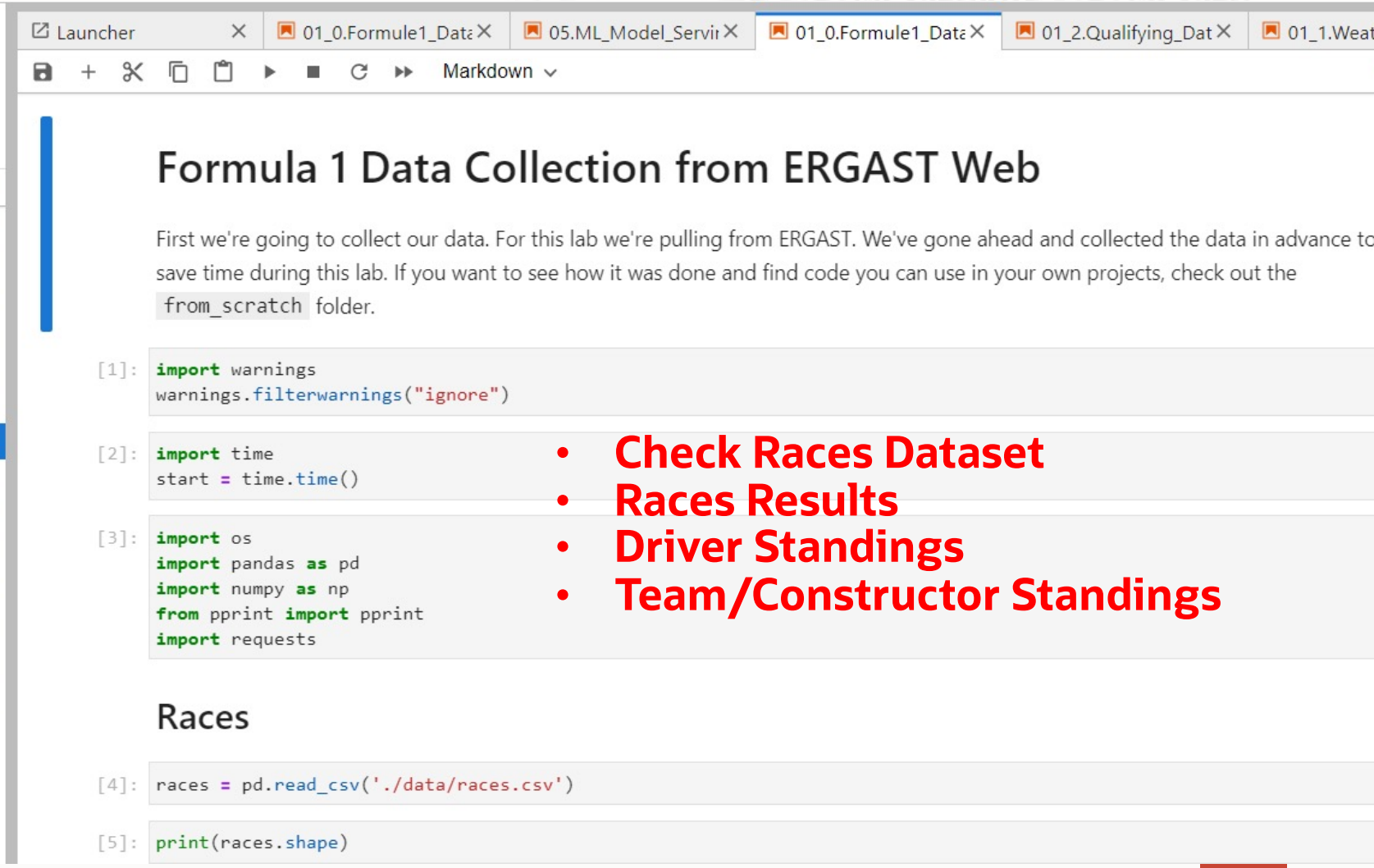


Step 1: Execute 3 notebooks to check Datasets downloaded.

Use Notebooks Version in “beginners” Directory.

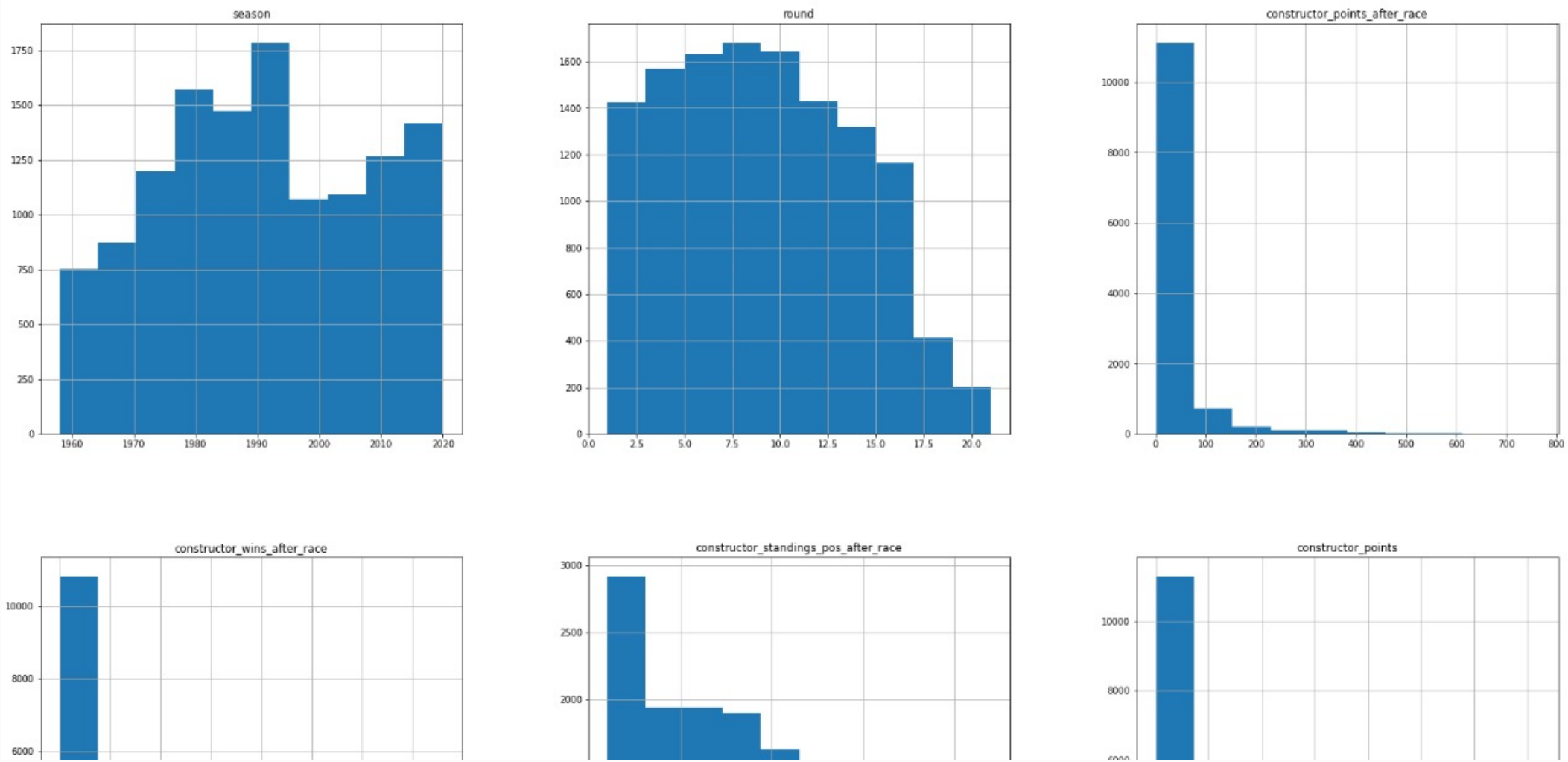
01_0.Formule1_Data_Collection.ipynb



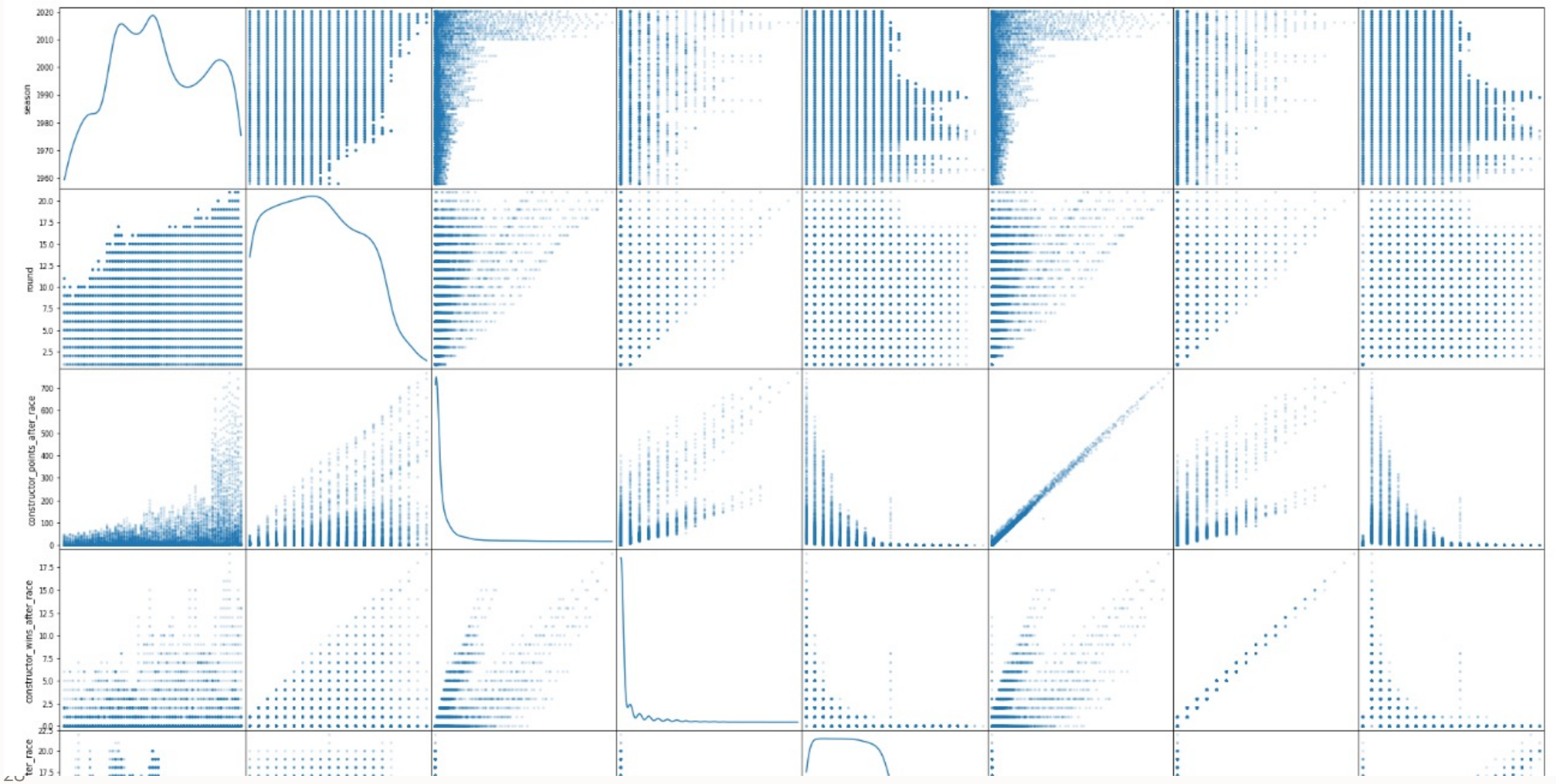


- **Check Races Dataset**
- **Races Results**
- **Driver Standings**
- **Team/Constructor Standings**

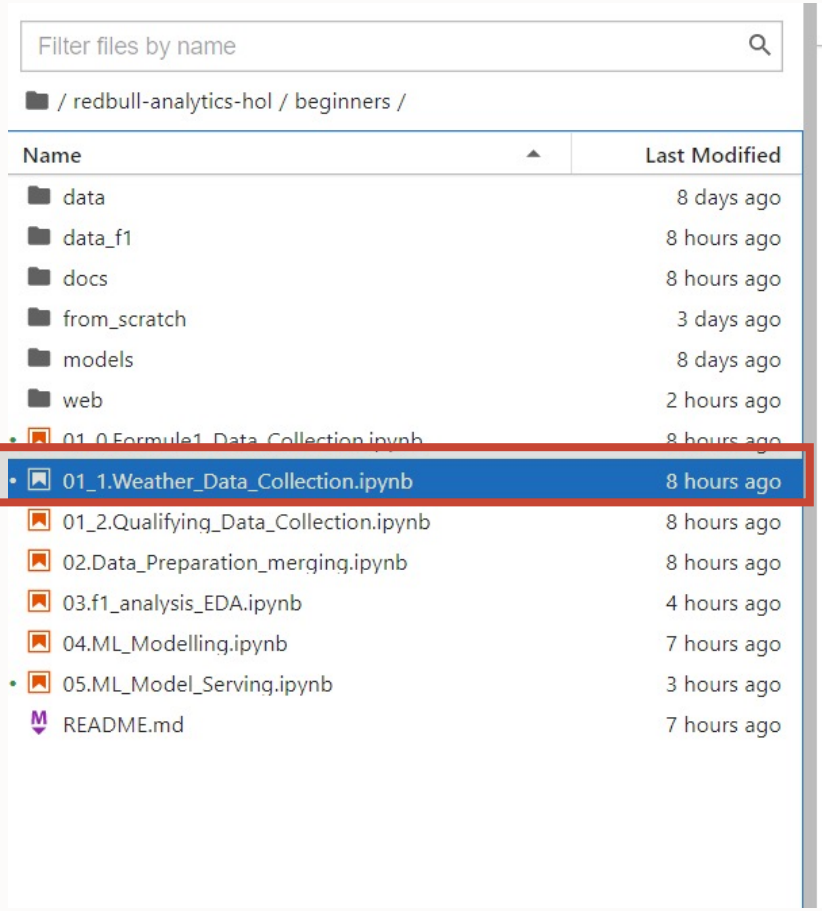
01_0.Formule1_Data_Collection.ipynb: Feature Distributions

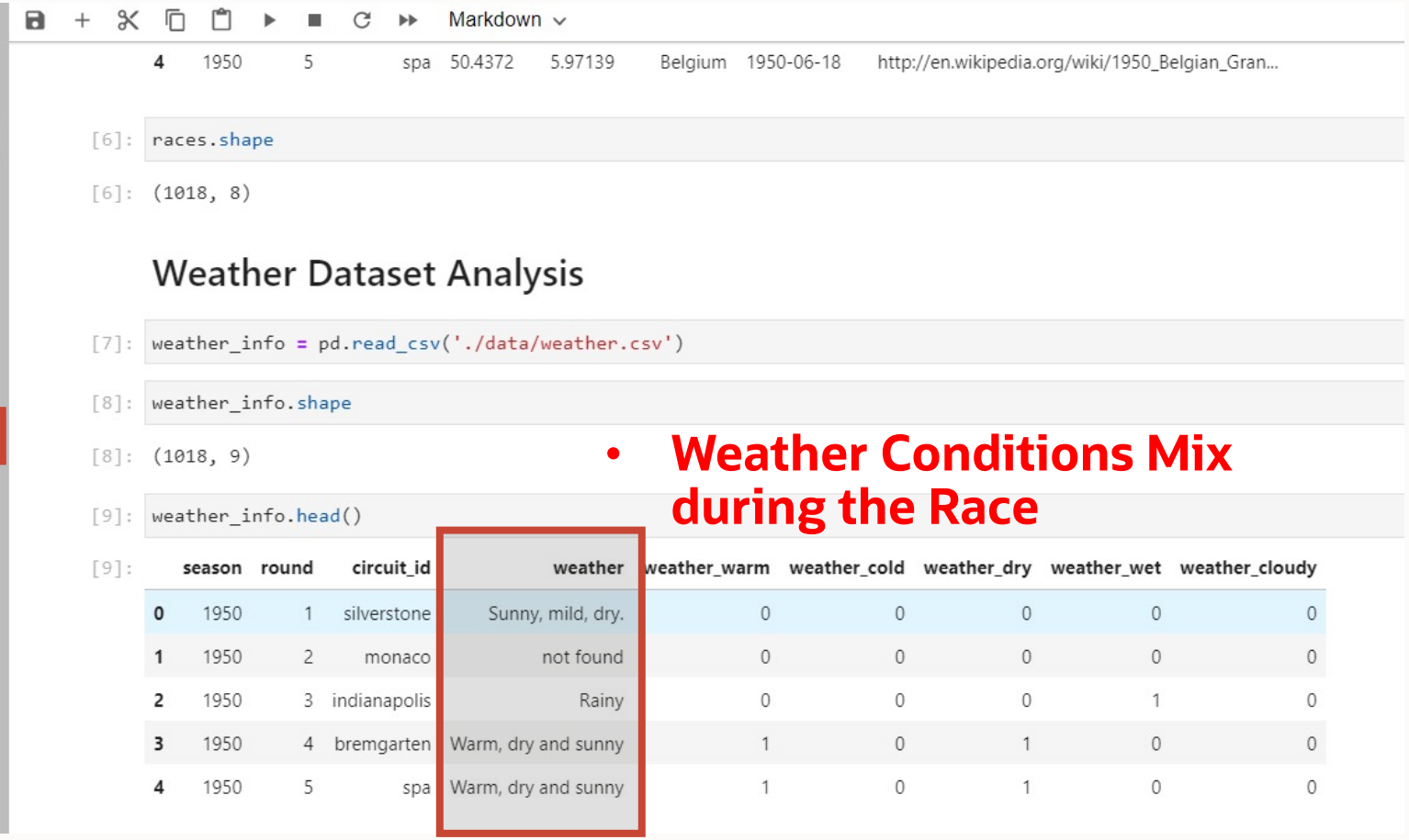


01_0.Formule1_Data_Collection.ipynb: Feature-Feature Relationships



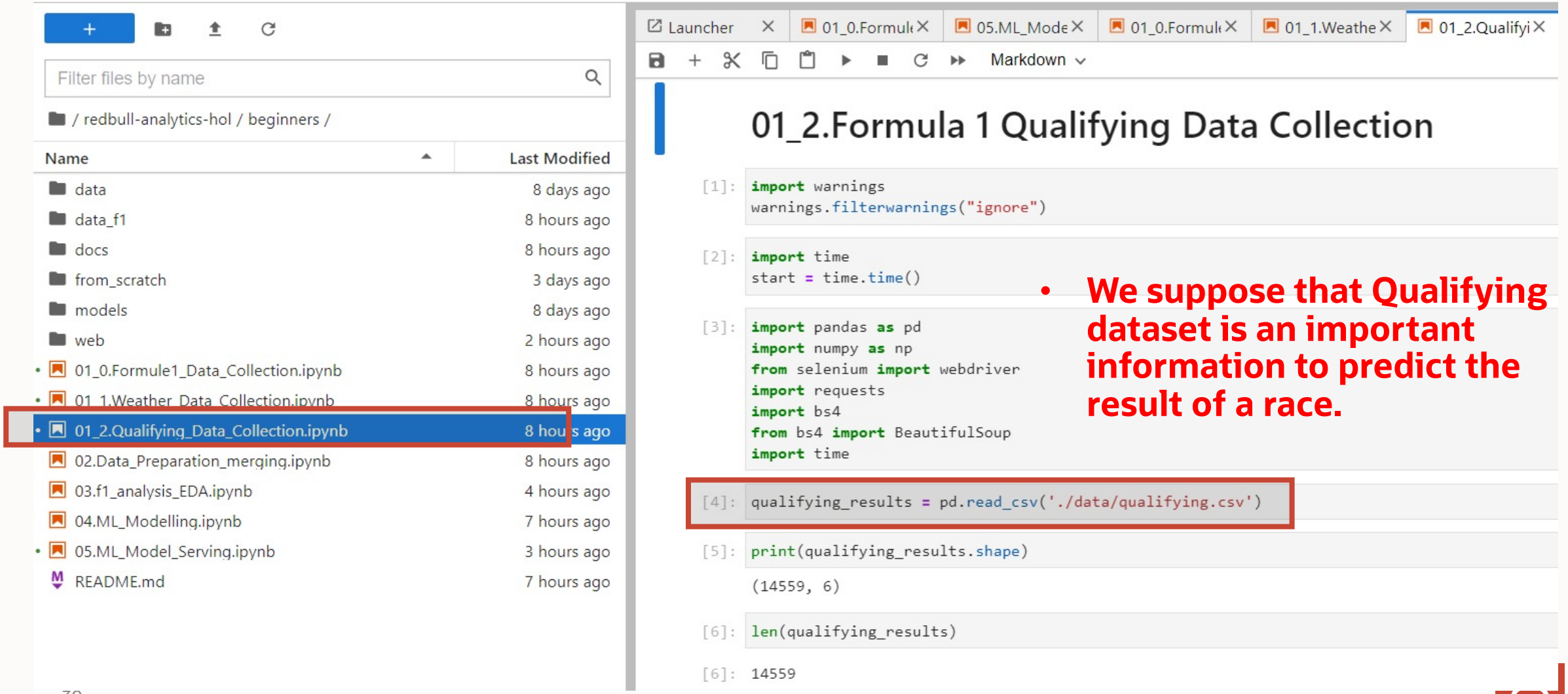
01_1.Weather_Data_Collection.ipynb





- **Weather Conditions Mix during the Race**

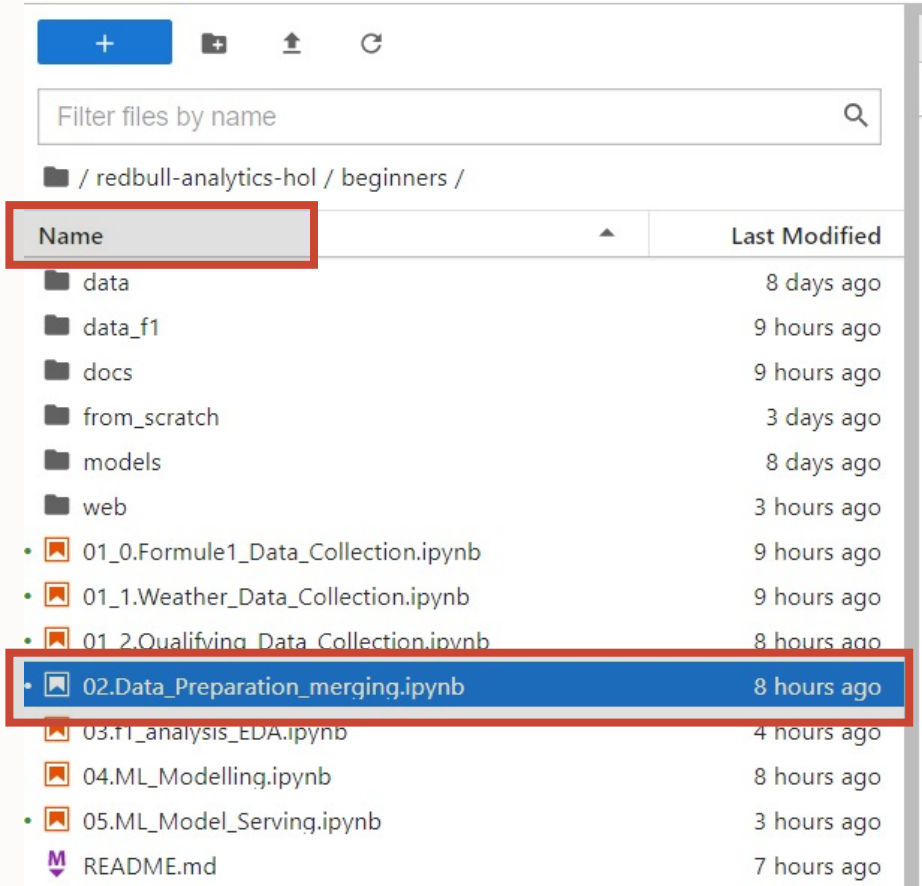
01_2.Qualifying_Data_Collection.ipynb

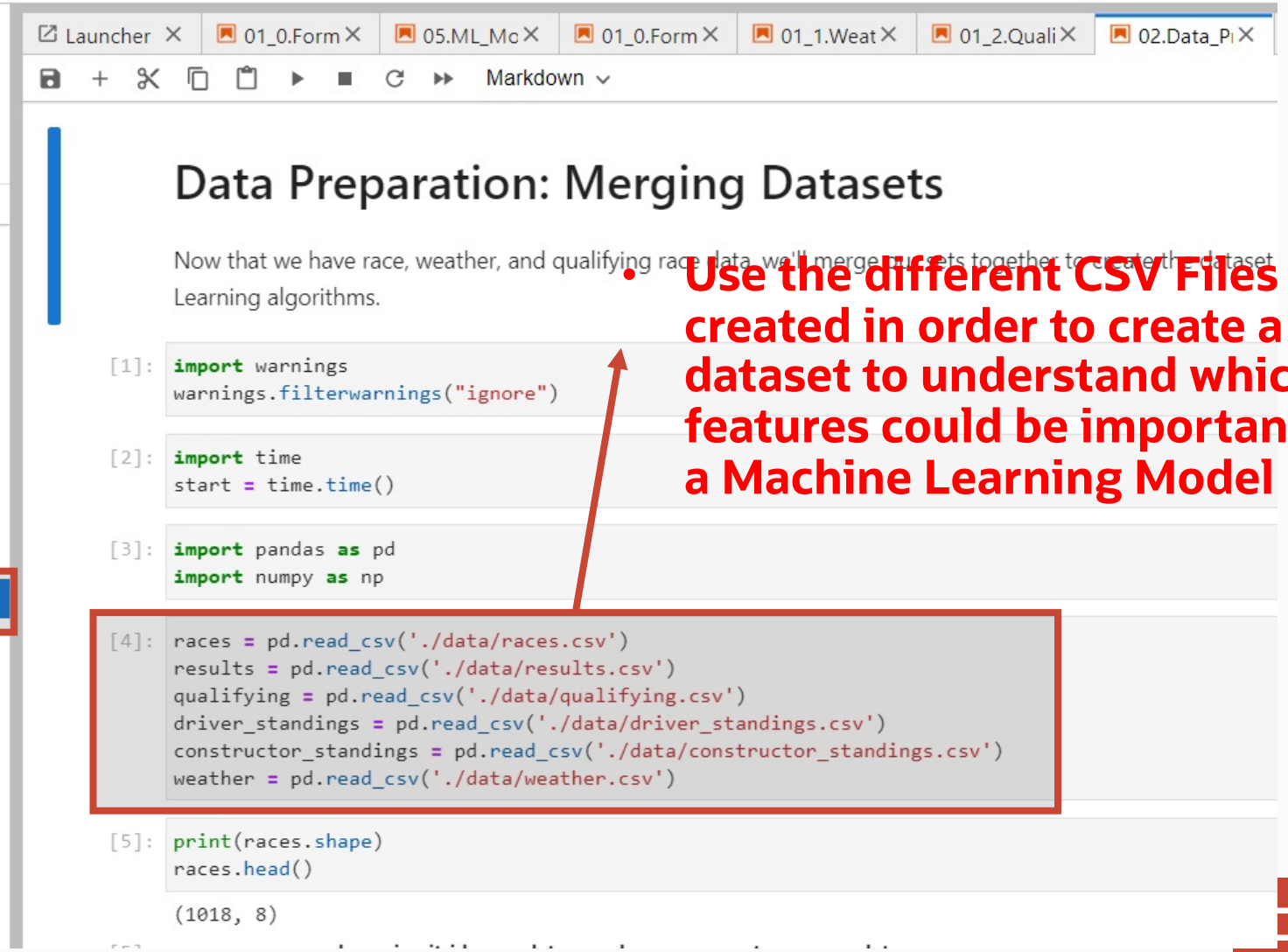


- **We suppose that Qualifying dataset is an important information to predict the result of a race.**

Step 2: Data Preparation.

02.Data_Preparation_merging.ipynb

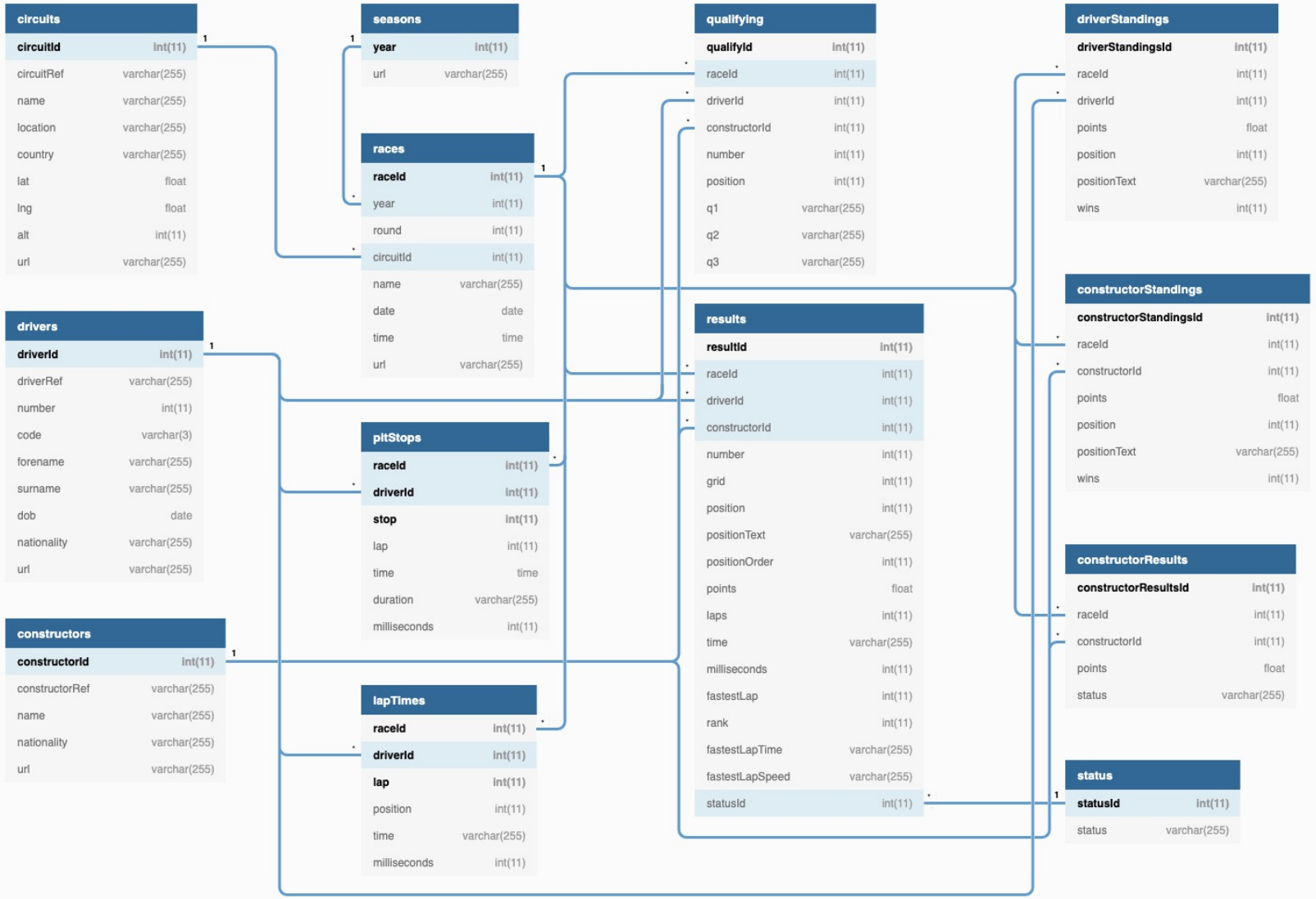




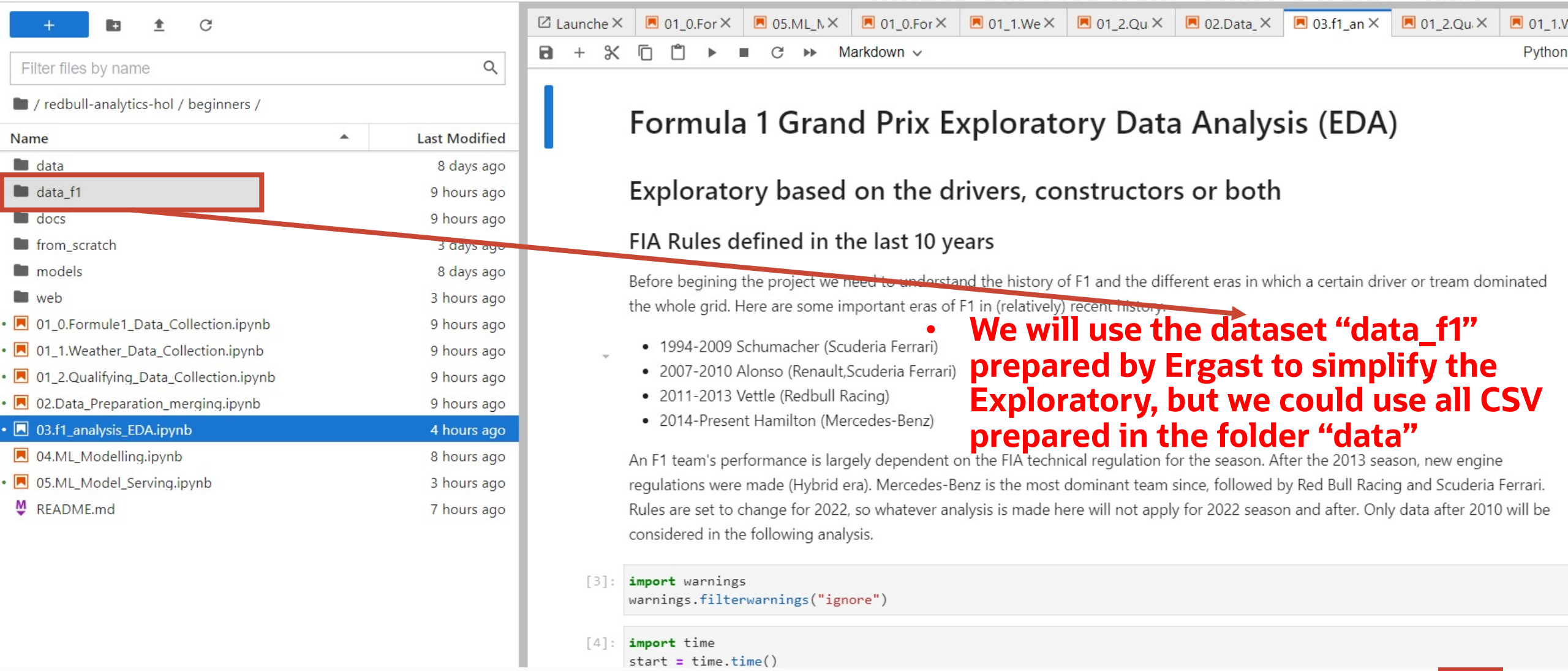
- **Use the different CSV Files created in order to create a dataset to understand which features could be important for a Machine Learning Model**

Step 3: Exploratory Data Analysis.

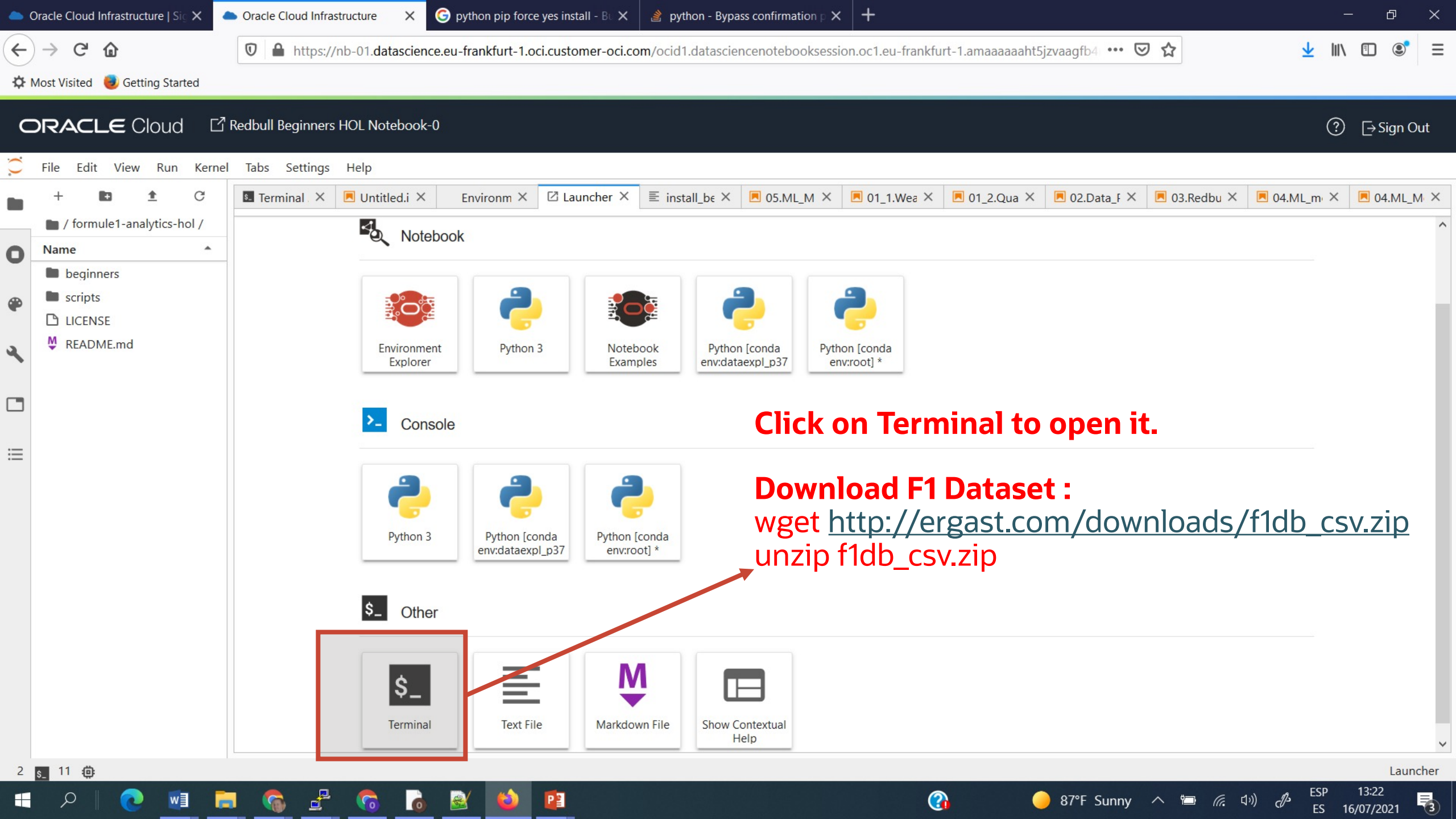
Data Analysis using ERGAST Data Model



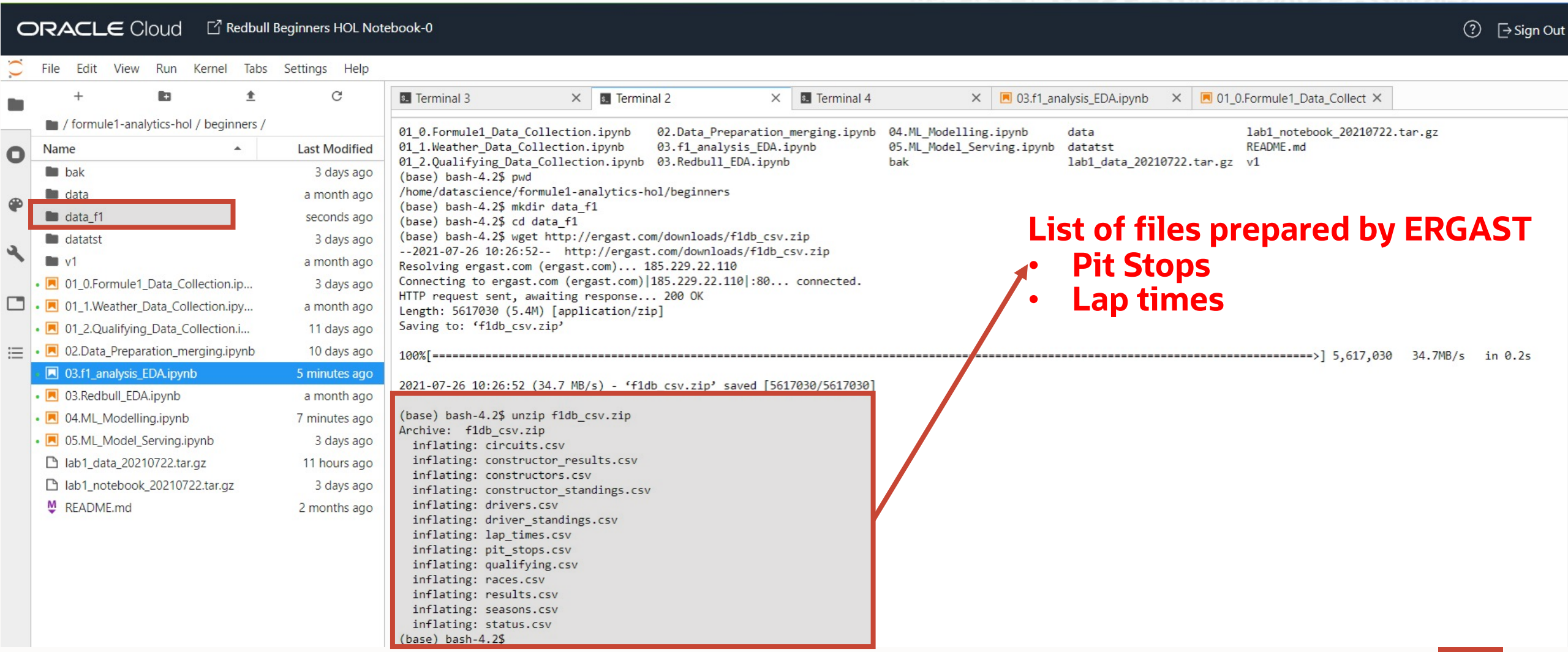
03.f1_analysis_EDA.ipynb



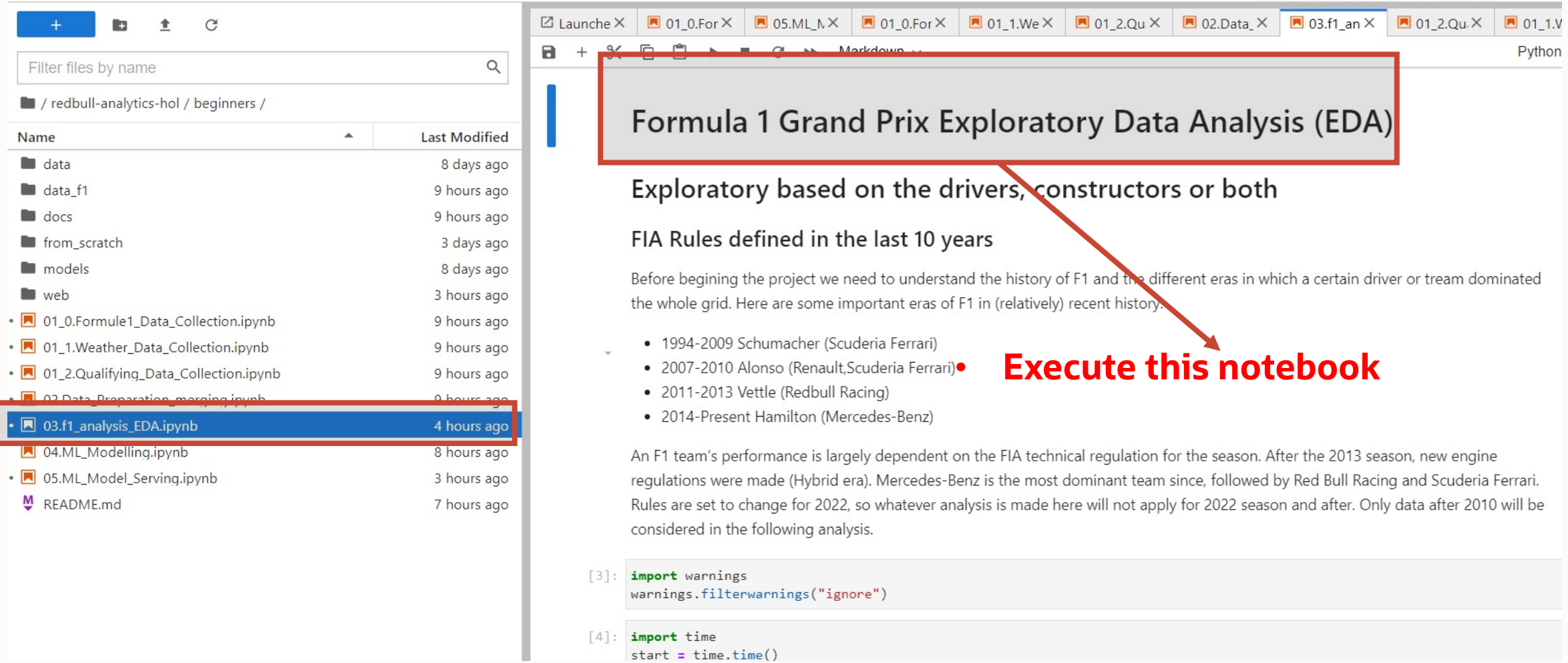
- **We will use the dataset “data_f1” prepared by Ergast to simplify the Exploratory, but we could use all CSV prepared in the folder “data”**


Oracle Cloud Infrastructure
Oracle Cloud Infrastructure
python pip force yes install
python - Bypass confirmation
https://nb-01.datascience.eu-frankfurt-1.oci.customer-oci.com/ocid1.datasciencenotebooksession.oc1.eu-frankfurt-1.amaaaaaaht5jzvaagfb4
Most Visited
Getting Started
ORACLE Cloud
Redbull Beginners HOL Notebook-0
Sign Out
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
/ formule1-analytics-hol /
Name
beginners
scripts
LICENSE
README.md
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
unzip f1db_csv.zip
Launcher
87°F Sunny
ESP
ES
13:22
16/07/2021

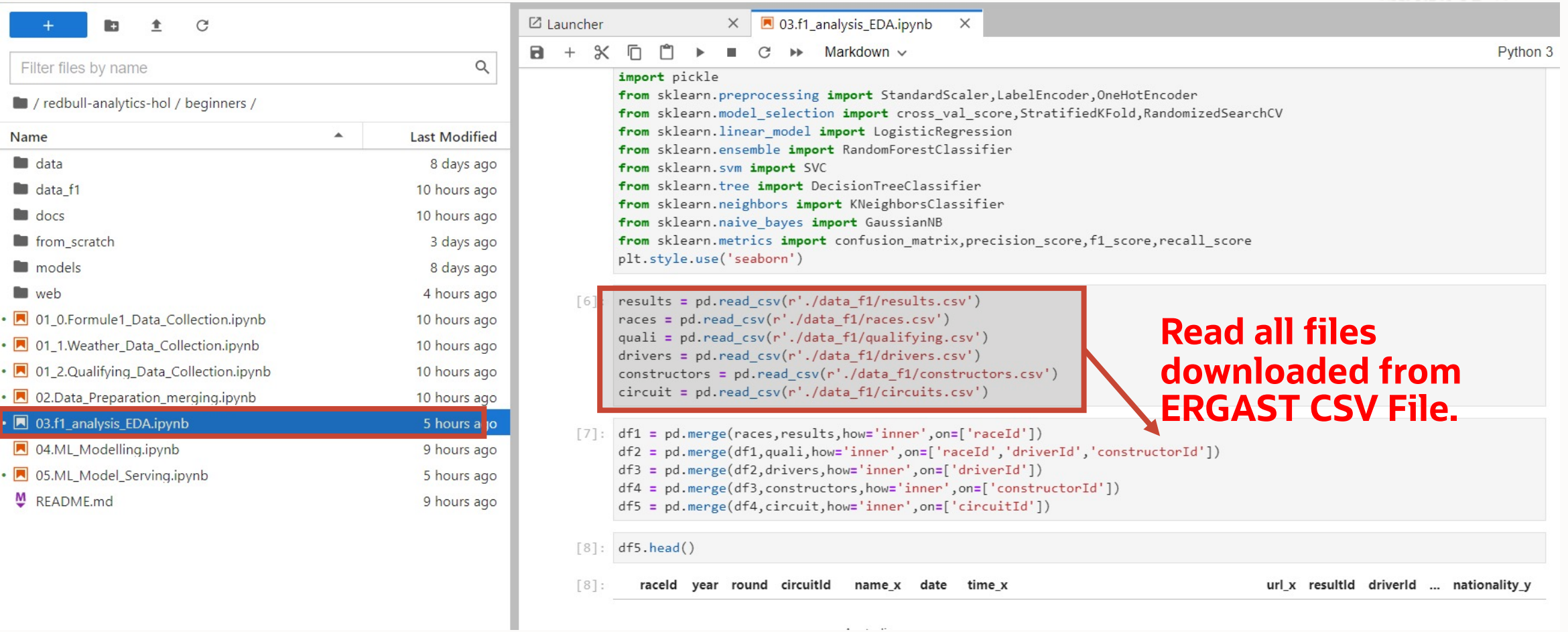
Prepare the environment with extended datasets from Ergast



03.f1_analysis_EDA.ipynb



Execute the notebook



Cleaning Dataset and Replace some data

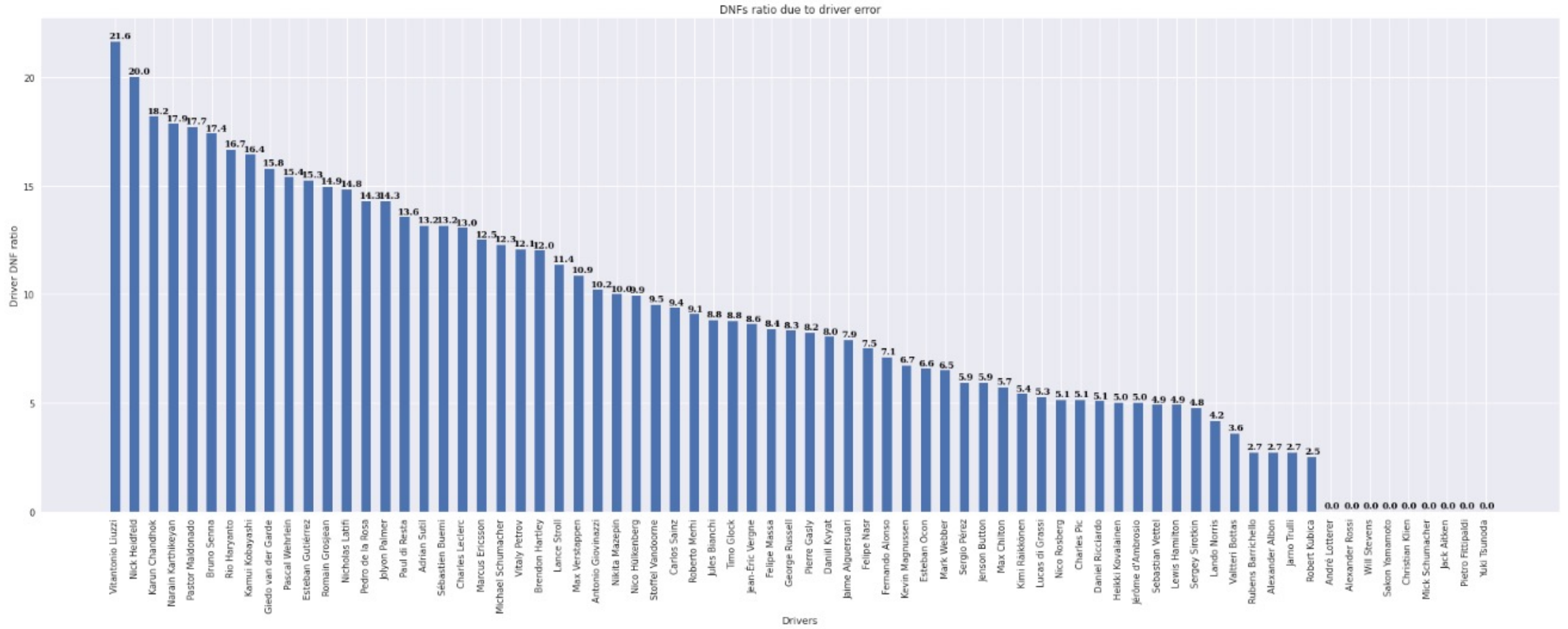
Cleaning Dataset

```
#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)
```

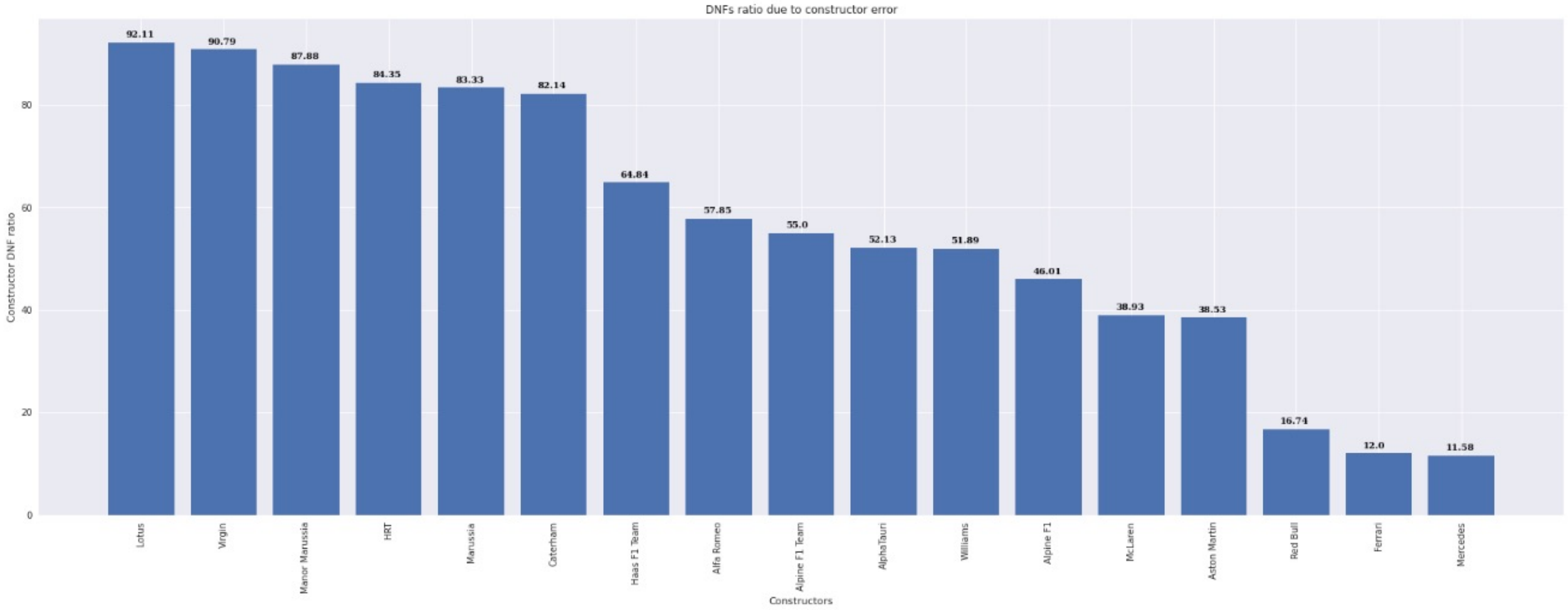
```
data['driver_nationality'] = data['driver_nationality'].apply(lambda x: str(x)[:3])
data['constructor_nationality'] = data['constructor_nationality'].apply(lambda x: str(x)[:3])
data['country'] = data['country'].apply(lambda x: 'Bri' if x=='UK' else x)
data['country'] = data['country'].apply(lambda 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)
```

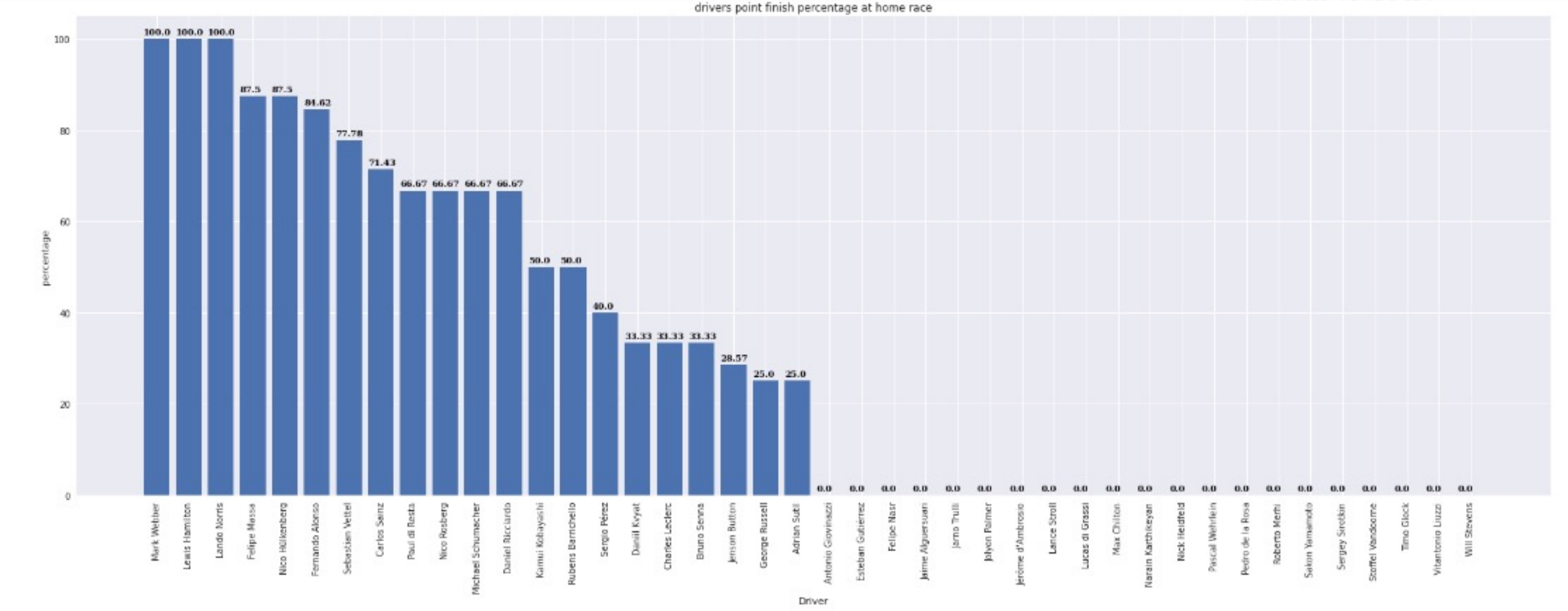
Check “Did Not Finish” (DNF) Ratio per Driver



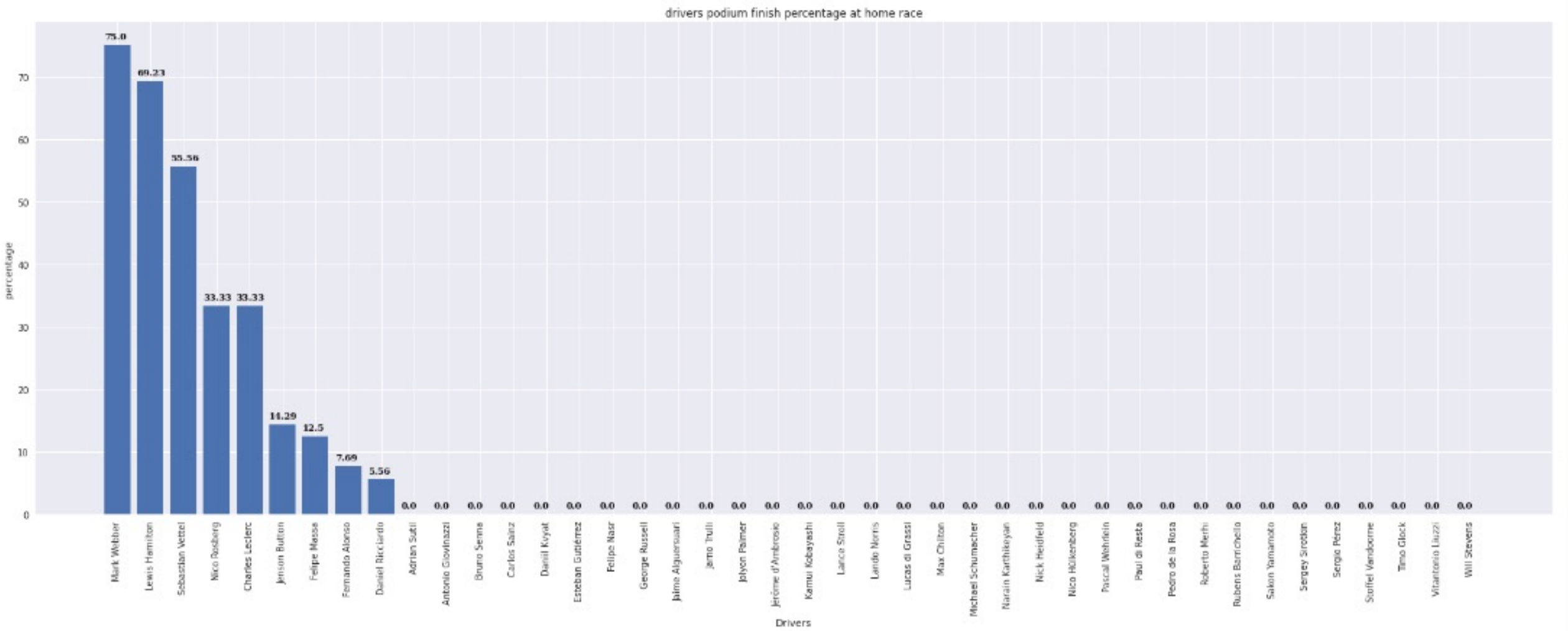
Check DNF Ratio per Constructor



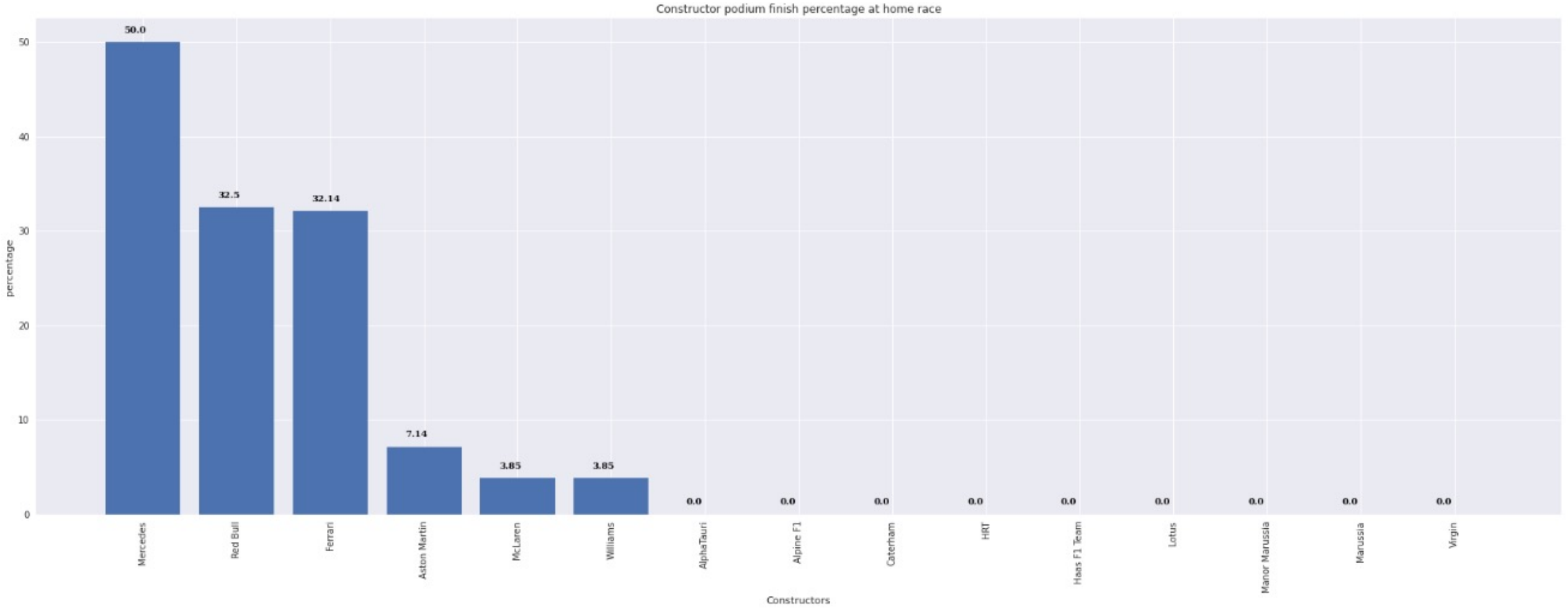
Check Driver Points Percentage



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 | Sign Out

File Edit View Run Kernel Tabs Settings Help

/ formule1-analytics-hol / beginners /

Name	Last Modified
bak	seconds ago
data	5 minutes ago
data_f1	38 minutes ago
datatst	3 days ago
models	5 minutes ago
v1	a month ago
01_0.Formule1_Data_Collection.ip...	3 days ago
01_1.Weather_Data_Collection.ipy...	a month ago
01_2.Qualifying_Data_Collection.ip...	11 days ago
02.Data_Preparation_merging.ipynb	10 days ago
03.f1_analysis_EDA.ipynb	14 minutes ago
04.ML_Modelling.ipynb	5 minutes ago
05.ML_Model_Serving.ipynb	3 days ago
lab1_data_20210722.tar.gz	12 hours ago
lab1_notebook_20210722.tar.gz	3 days ago
README.md	2 months ago

Terminal 3 | Terminal 2 | 04.ML_Modelling.ipynb | install_beginner_env.sh | Terminal 4

Markdown | Python [conda env:dataexpl_p37_cpu_v1]

Search the best Machine Learning Algorithms to predict

[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')
```

- **Using the Dataset prepared.**

[4]:
```
data.head()
```

[4]:

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

5 rows × 100 columns

4 13 Python [conda env:dataexpl_p37_cpu_v1] | Idle

Mode: Command Ln 1, Col 1 04.ML_Modelling.ipynb

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

[ ]:

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

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

[ ]:

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

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

[ ]:

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

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 ech 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

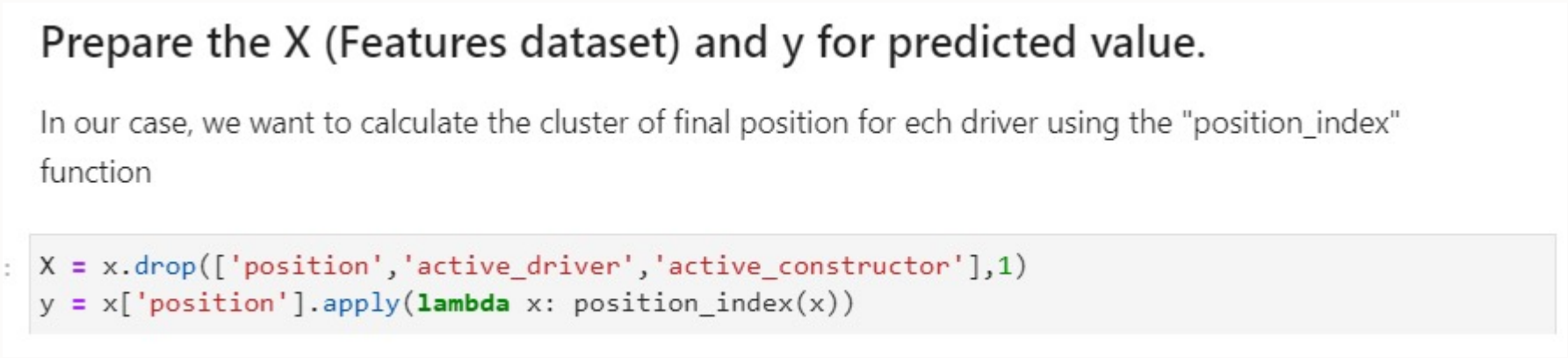
[79]:
```
# Implement boxplot
```

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))



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 ech 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 ech 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

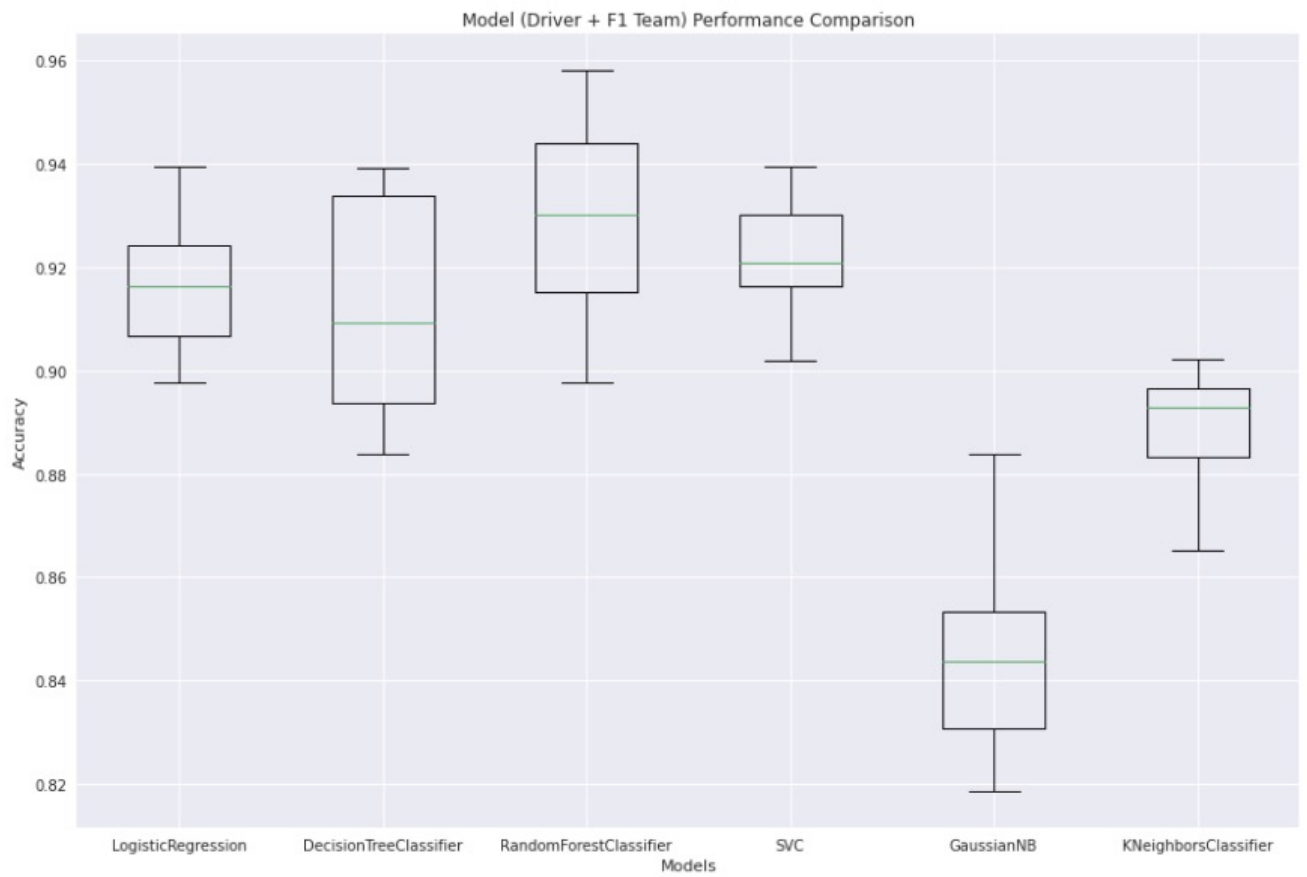
[79]:
```
# Implement boxplot
```

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 ech 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
```

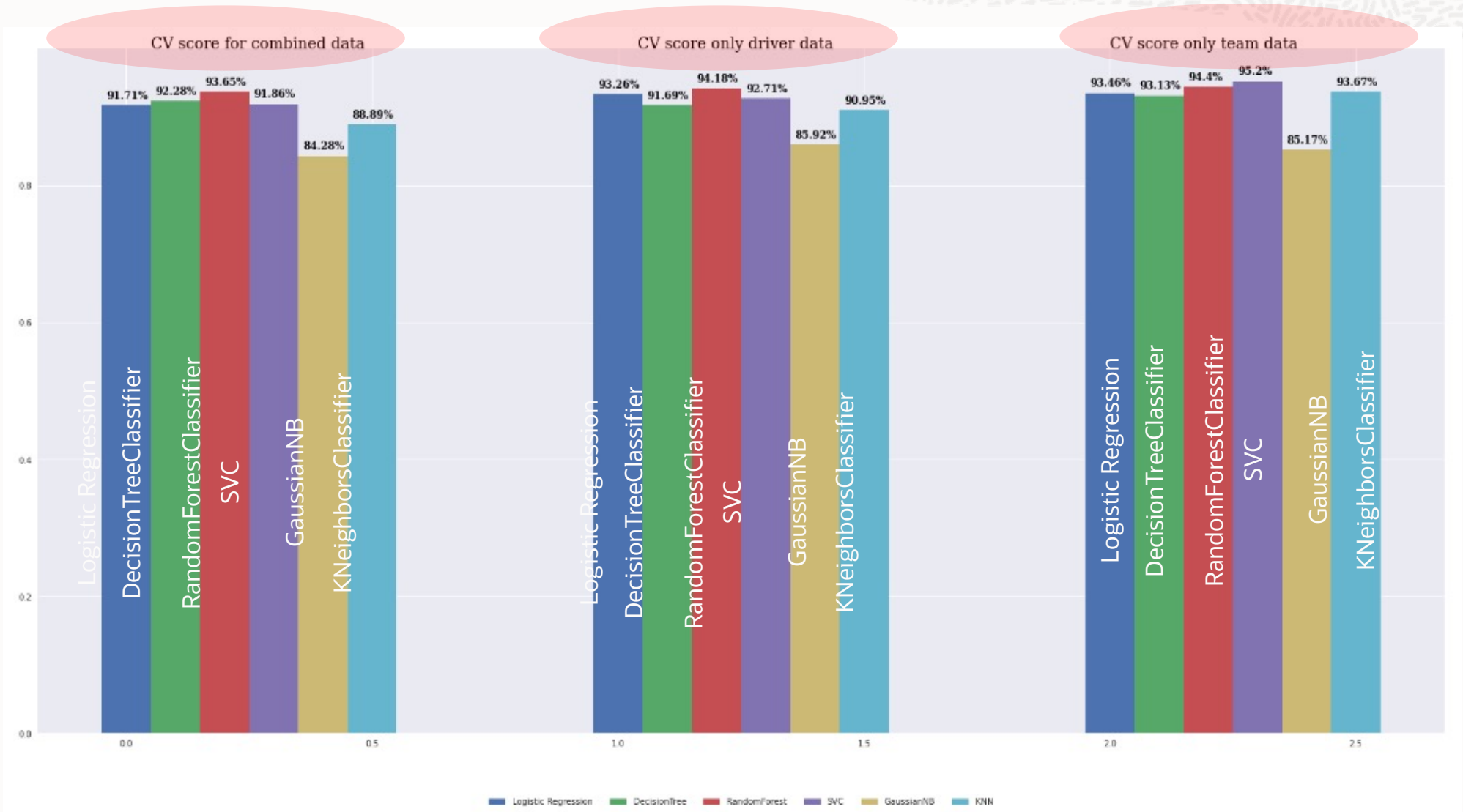
Use the same boxplot plotter used in the previous Models

[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

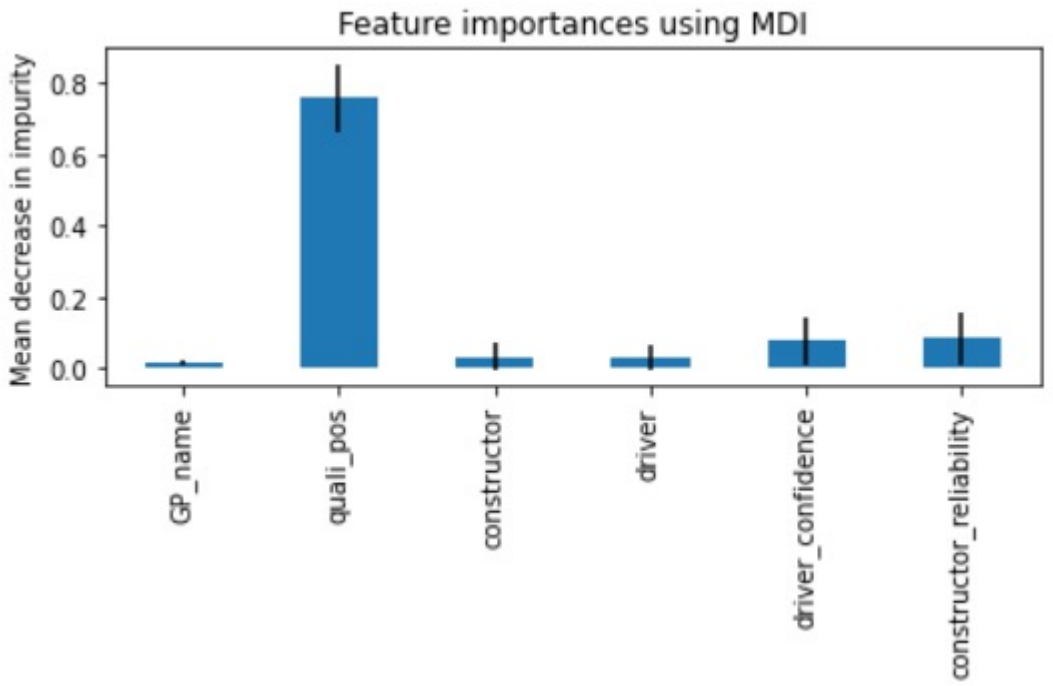
Generate Qualifying dataset for Predictor

```
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
..                             ...                 ...        ...
```

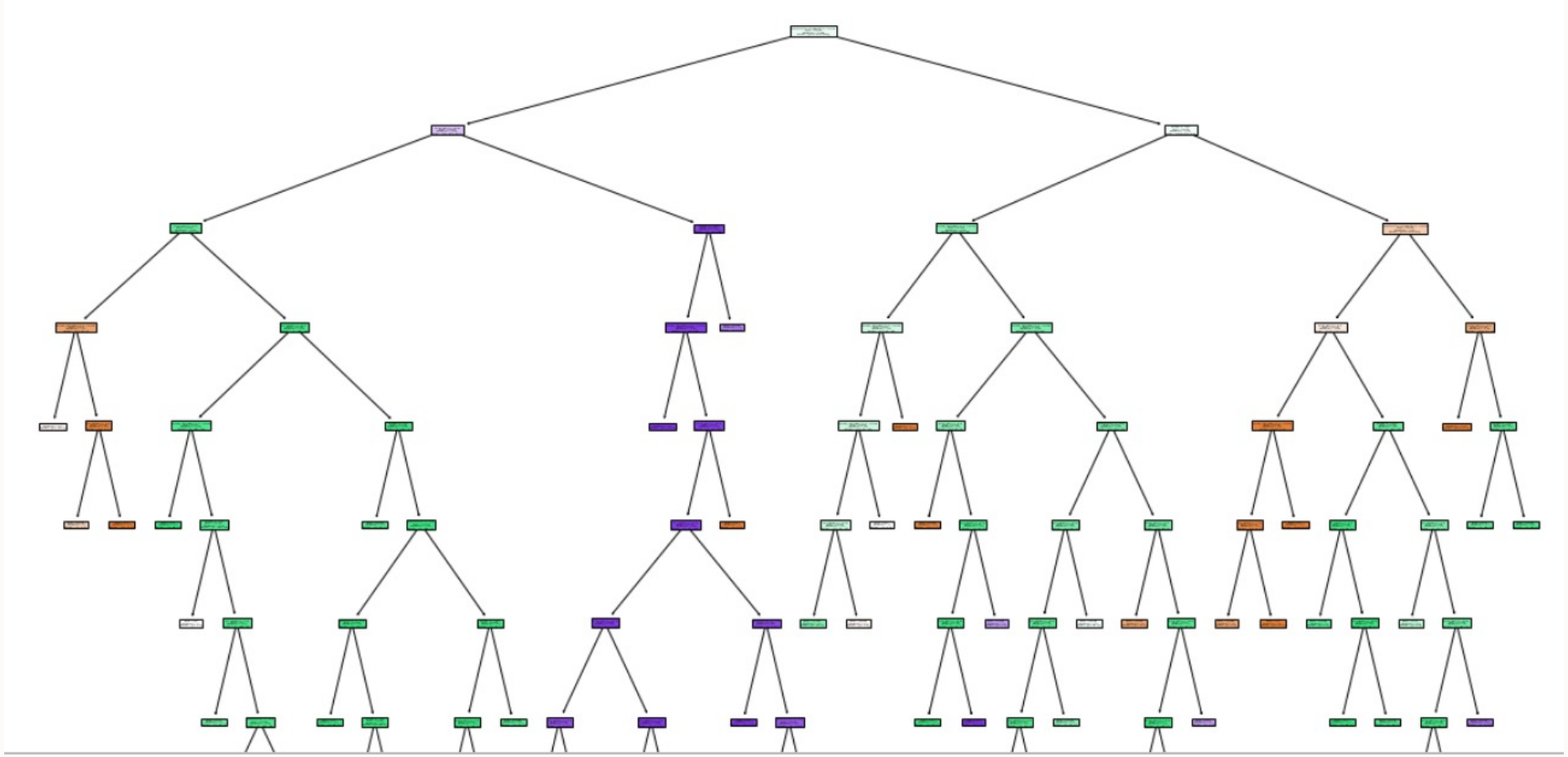
05.ML_Model_Serving.ipynb



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

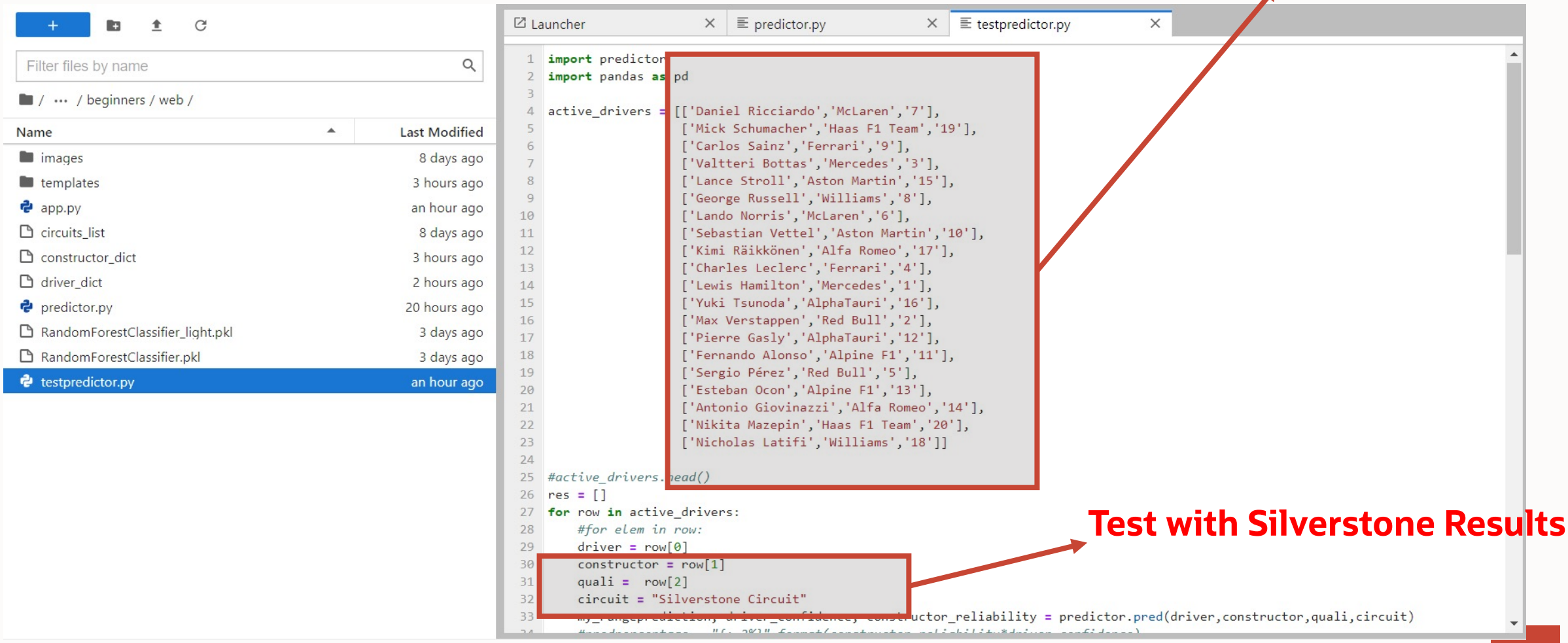
Load the Trained Model and the dataset saved.

Create a Prediction Method

Call “predict” from Loaded 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
```

How to test the Model Predictor



The Model depends on Qualifying Result: Main feature

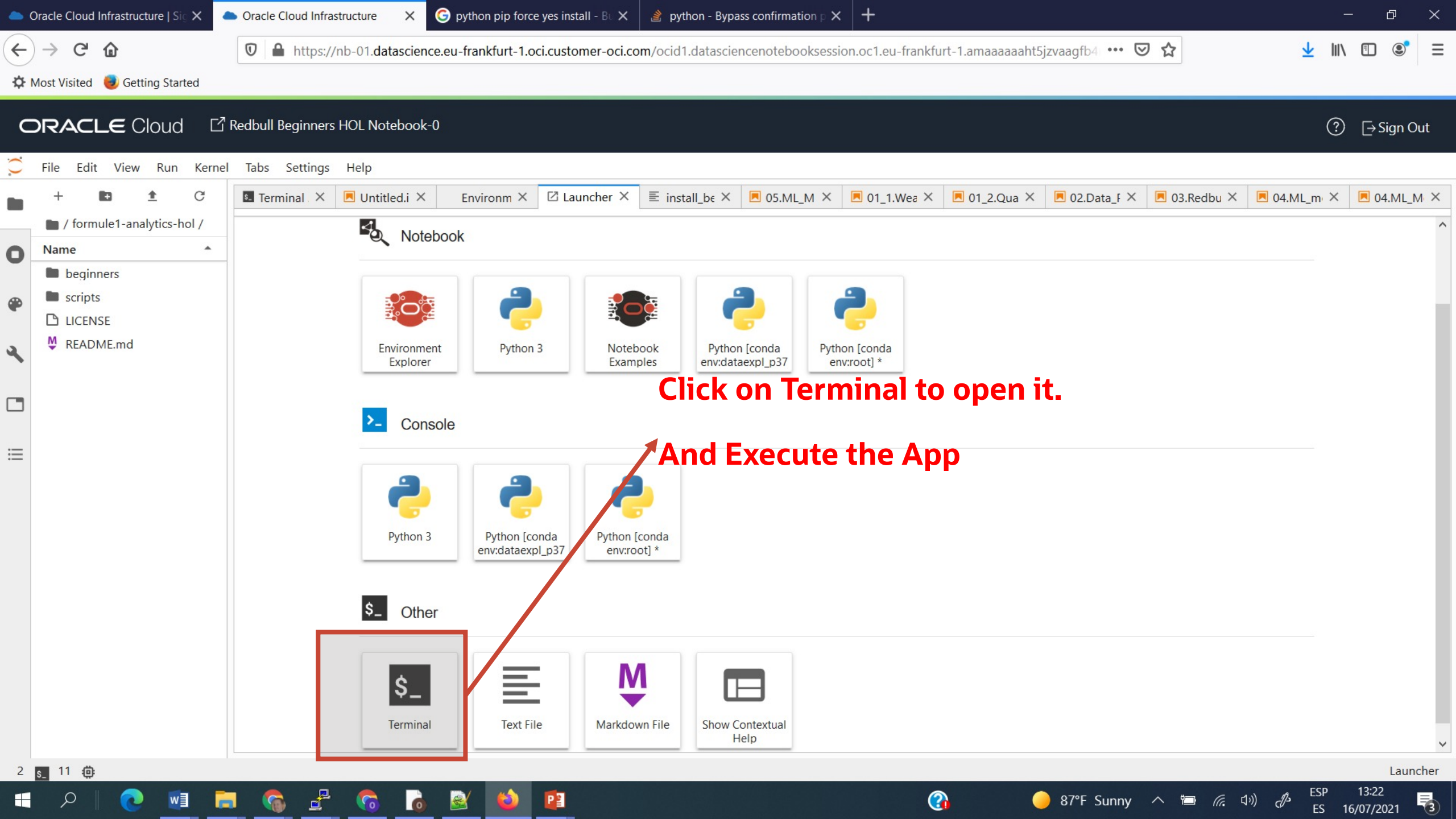
Test with Silverstone Results

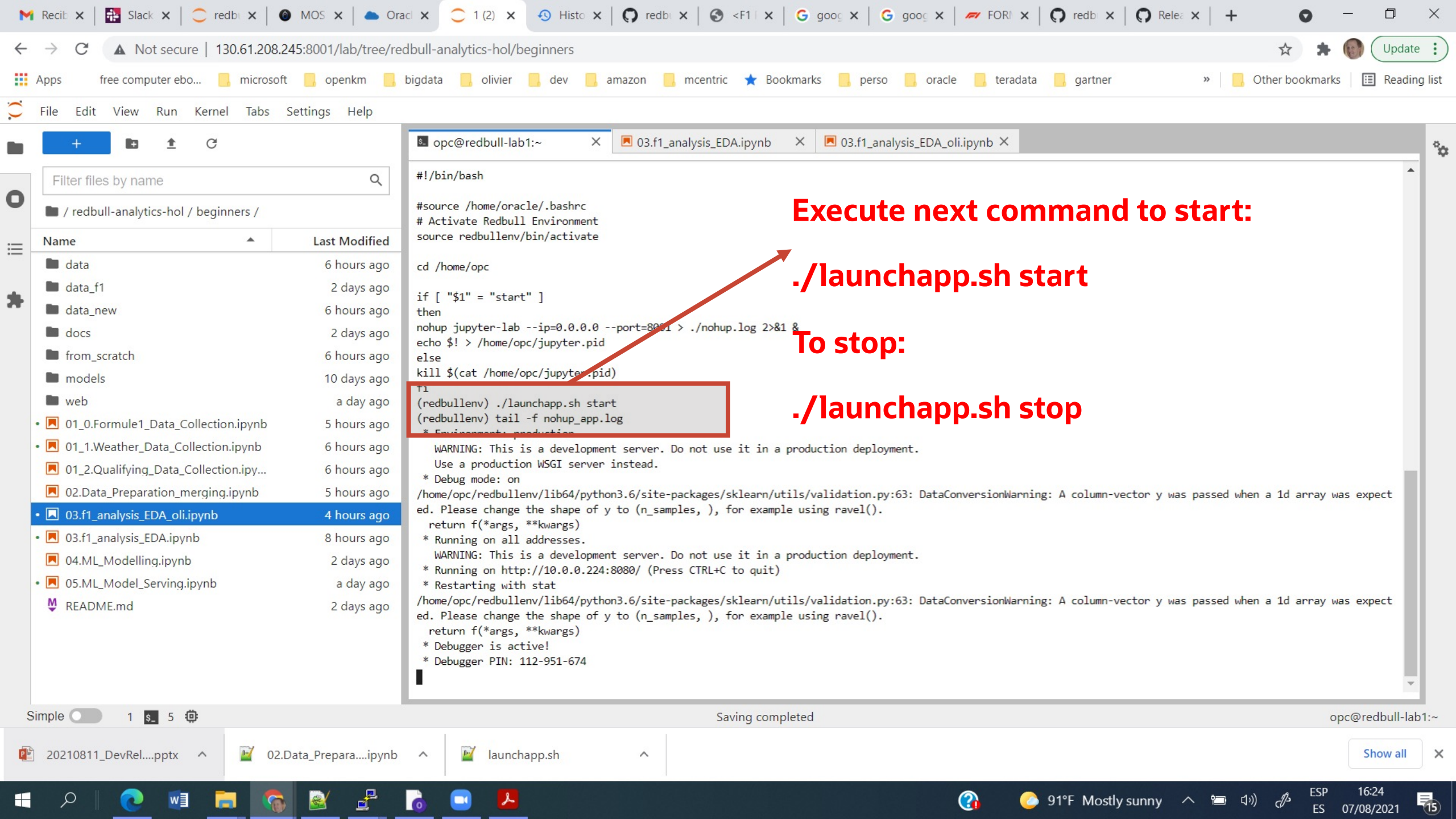
Write the app.py using Flask

```python
@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

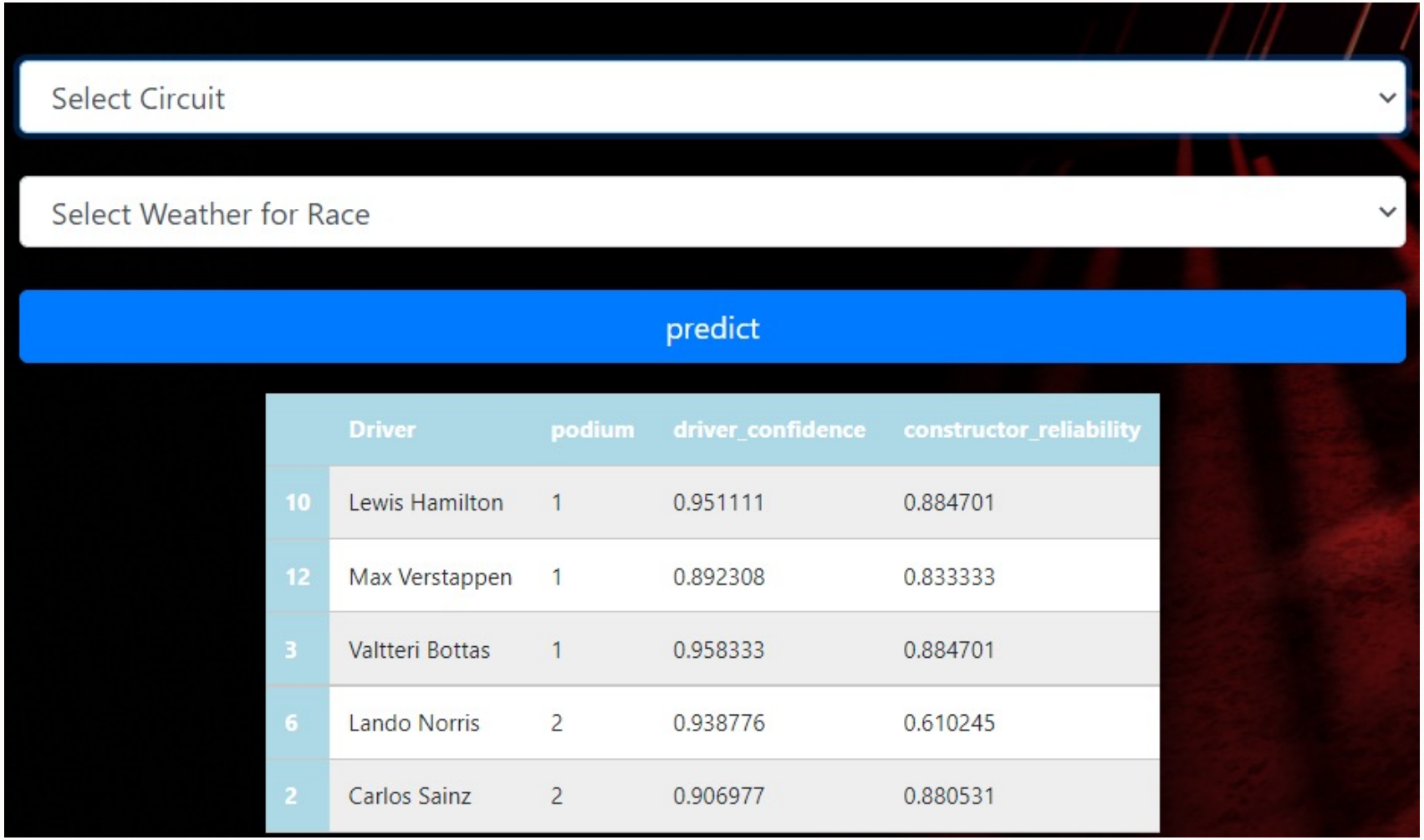
Open a Server App


Oracle Cloud Infrastructure
Oracle Cloud Infrastructure
python pip force yes install
python - Bypass confirmation
https://nb-01.datascience.eu-frankfurt-1.oci.customer-oci.com/ocid1.datasciencenotebooksession.oc1.eu-frankfurt-1.amaaaaaaht5jzvaagfb4
Most Visited
Getting Started
ORACLE Cloud
Redbull Beginners HOL Notebook-0
Sign Out
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
/ formule1-analytics-hol /
Name
beginners
scripts
LICENSE
README.md
Notebook
Environment Explorer
Python 3
Notebook Examples
Python [conda env:dataexpl_p37
Python [conda env:root] *
Click on Terminal to open it.
And Execute the App
Console
Python 3
Python [conda env:dataexpl_p37
Python [conda env:root] *
Other
Terminal
Text File
Markdown File
Show Contextual Help
2
11
Launcher
87°F Sunny
ESP
ES
13:22
16/07/2021


Execute next command to start:
./launchapp.sh start
To stop:
./launchapp.sh stop
#!/bin/bash
#source /home/oracle/.bashrc
Activate Redbull Environment
source redbullenv/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
(redbullenv) ./launchapp.sh start
(redbullenv) tail -f nohup_app.log
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/redbullenv/lib64/python3.6/site-packages/sklearn/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/redbullenv/lib64/python3.6/site-packages/sklearn/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

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

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



	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 ?

Resources while fine-tuning your model

- Slack group:
  https://bit.ly/3ixPQWT
- Lab guide:
  https://bit.ly/2U5FPGW

Thank You

DevRel Team

[Nov. 2020]

ORACLE