IBM DATA SCIENCE CAPSTONE PROJECT

PRESENTER: KSHITIJA BASARGE

OUTLINE

- EXECUTIVE SUMMARY
- INTRODUCTION
- METHODOLOGY
- ANALYSIS
- SQL RESULTS
- CONCLUSION
- APPENDIX

EXECUTIVE SUMMARY

- Methods:
- **Github Link:** https://github.com/KshitijaBasarge/IBM_DataScience_Assignments/tree/feature/IBM_assignments/SpaceX_Capstone_Project
  - Data Collection
  - Data Wrangling
  - EDA Data Visualization
  - EDA with SQL
    Predictive Analysis
  - Interactive Map Folium
  - Interactive Dashboard using Plotly

INTRODUCTION

- In this project we predict if the Falcon 9 first stage will land successfully. So if the first stage lands , we can also eventually determine cost of landing, as reusing them to launch reduces the satellite upward cost. We use previous data of launches of Falcon 9 rocket to predict the first stage landing related with space launch site, the payload orbit, mass, landing pad location.

METHODOLOGY

- Data Collection - API & Web Scraping
- Data Wrangling – Extract load and Transform
- Cleaning data values
- EDA with SQL, Visualization
- Interactive map with Folium, Plotly Dashboard
- Predictive Analysis using ML models

METHODOLOGY

- REST API: Get request to SpaceX API, to extract the data in JSON format, clean, transform to dataframe, and normalize it.
- WEB SCRAPING: Perform web scraping to collect Falcon 9 historical launch records from wikipedia

REST API

Now let's start requesting rocket launch data from SpaceX API with the following URL:

```
[6]: spacex_url="https://api.spacexdata.com/v4/launches/past"

[7]: response = requests.get(spacex_url)
```

Check the content of the response

```
[8]: print(response.content)
```

```
b'[{"fairings":{"reused":false,"recovery_attempt":false,"recovered":false,"ships":[]},"links":{"patch
ge":"https://images2.imgbox.com/5b/02/QcxHUb5V_o.png"},"reddit":{"campaign":null,"launch":null,"media
```

```
[14]: # Use json_normalize meethod to convert the json result into a dataframe
      data = pd.json_normalize(response.json())  # convert to flat table
      print(data.head())
```

	static_fire_date_utc	static_fire_date_unix	net	window	\
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	0.0	
1	None	NaN	False	0.0	
2	None	NaN	False	0.0	
3	2008-09-20T00:00:00.000Z	1.221869e+09	False	0.0	
4	None	NaN	False	0.0	

- Make GET request to SpaceX API
- Normalize dataframe, Filter Falcon 9, save to csv
- Loop through the request content and extract data

```
data_falcon9 = data_falcon9[data_falcon9['BoosterVersion']=='Falcon 9']
print(data_falcon9['BoosterVersion'].value_counts())
```

```
Falcon 9    90
```

```
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and rows that have
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single value in the list and replace the feature.
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting the date leaving the time
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

DATA WRANGLING

- In the SpaceX dataset there are data with different cases of successful/failed booster landings.

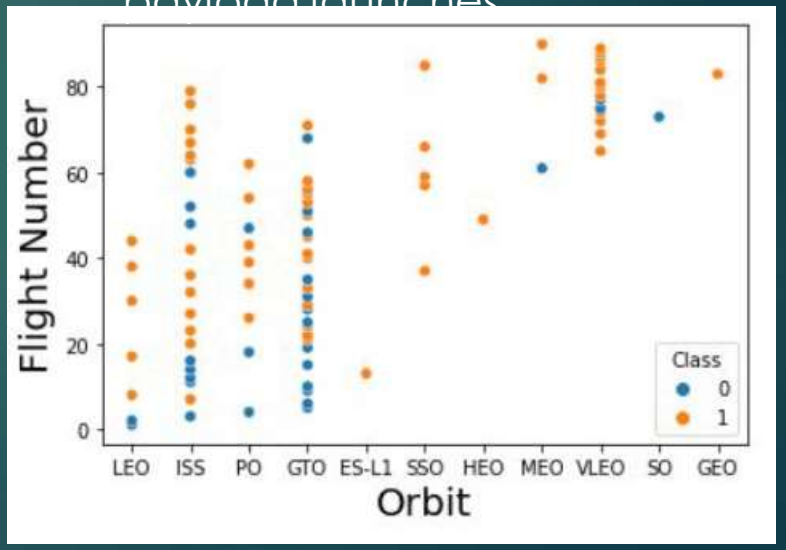
```
# Calculate the mean value of PayloadMass column
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].fillna(data_falcon9['PayloadMass'].mean())
data_falcon9.isnull().sum()
```

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            # TODO: Append the flight_number into launch_dict with key 'Flight No.'
            #print(flight_number)
            datatimelist=date_time(row[0])
            # Date value
            # TODO: Append the date into launch_dict with key 'Date'
            date = datatimelist[0].strip(',')
            #print(date)
```

```
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
```

EDA (Exploratory Data Analysis) with SQL , Visualization

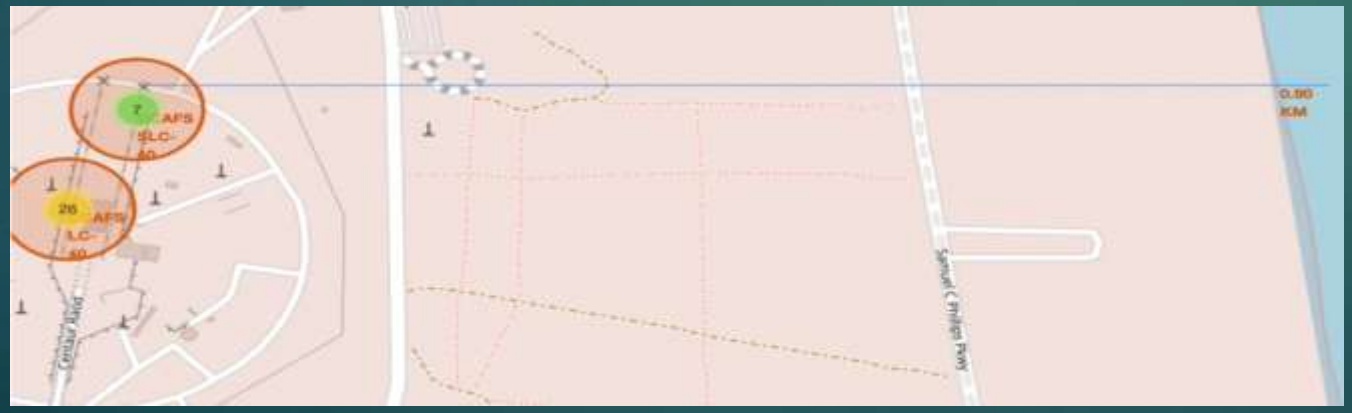
- Visualization of Flight number and Launch Sites
  - Most Launches from CCAFS-SLC-40
  - Fewer launches from VAFB SLC 4E site
- Payload and Launch Sites
  - CCAFS SLC 40 has more higher payload launches



- Flight Number vs Orbit Visualization
  - Most flights are to ISS, GTO, VLEO
  - Most falis are for ISS, GTO

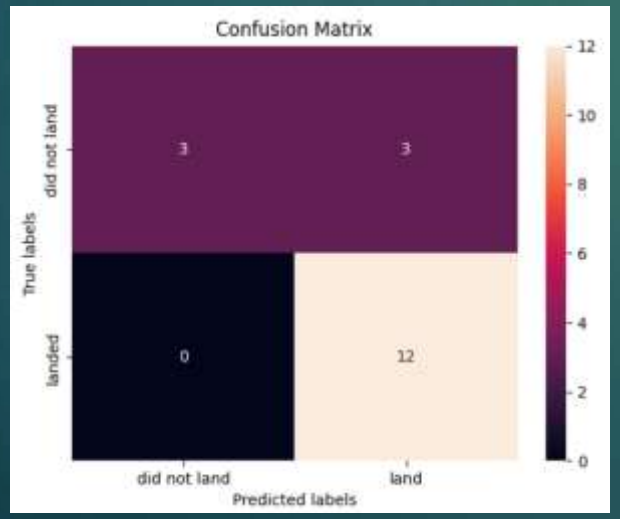
Visualization of Launches on Map

- Folium Circle and Marker for each launch site on site map
  - 3 Separate Launch are displayed in maps



Predictive Analysis

- Models includes for analysis:
  - KNN
  - Decision Tree
  - Logistic Regression
  - Support vector Machine
- Calculate accuracy on the test data. After analysing
  the KNN was the best Model with accuracy of 77% and
  Best score of 87%



INTERACTIVE DASHBORAD: SpaceX Launch Records Dashboard

- Visualization of:

Success Launch Sites

Visualize payload from different sites with range Slider for interacting with the plot

- Range Slider we can view sites that failed and succeeded for each booster version and the Payload they were carrying