SpaceX Falcon 9 First Stage Landing Prediction

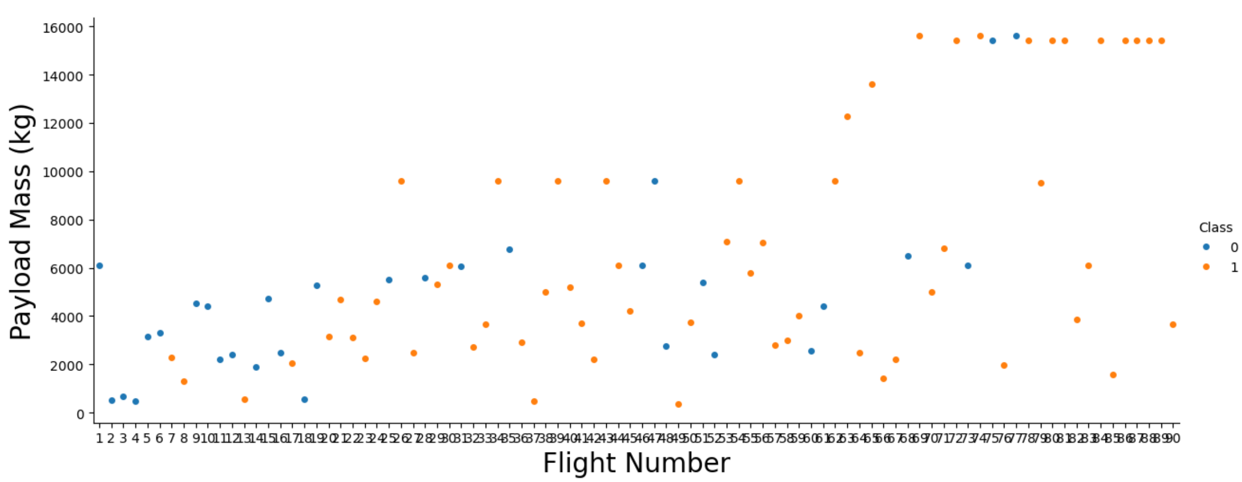
Exploratory Data Analysis and Feature Engineering using Pandas and Matplotlib

-Exploratory Data Analysis

-Preparing Data Feature Engineering

```
import piplite
await piplite.install(['numpy'])
await piplite.install(['pandas'])
await piplite.install(['seaborn'])

# pandas is a software library written for the Python programming
language for data manipulation and analysis.
import pandas as pd
#NumPy is a library for the Python programming language, adding
support for large, multi-dimensional arrays and matrices, along with a
large collection of high-level mathematical functions to operate on
these arrays
import numpy as np
# Matplotlib is a plotting library for python and pyplot gives us a
MatLab like plotting framework. We will use this in our plotter
function to plot data.
import matplotlib.pyplot as plt
#Seaborn is a Python data visualization library based on matplotlib.
It provides a high-level interface for drawing attractive and
informative statistical graphics
import seaborn as sns

from js import fetch
import io

URL = "https://cf-courses-data.s3.us.cloud-object-
storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/
dataset_part_2.csv"
resp = await fetch(URL)
dataset_part_2_csv = io.BytesIO((await resp.arrayBuffer()).to_py())
df=pd.read_csv(dataset_part_2_csv)
df.head(5)

   FlightNumber        Date BoosterVersion  PayloadMass Orbit
LaunchSite  \
0             1  2010-06-04       Falcon 9  6104.959412   LEO   CCAFS
SLC 40
1             2  2012-05-22       Falcon 9   525.000000   LEO   CCAFS
```

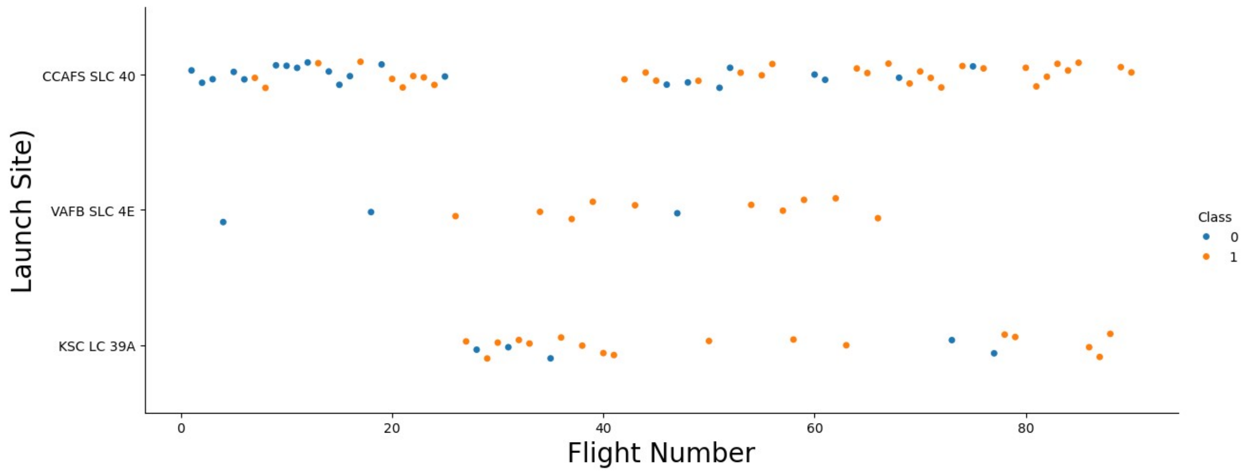
```
SLC 40
2             3  2013-03-01       Falcon 9   677.000000   ISS  CCAFS
SLC 40
3             4  2013-09-29       Falcon 9   500.000000    PO   VAFB
SLC 4E
4             5  2013-12-03       Falcon 9  3170.000000   GTO  CCAFS
SLC 40

       Outcome  Flights  GridFins  Reused   Legs LandingPad  Block  \
0    None None        1     False   False  False        NaN    1.0
1    None None        1     False   False  False        NaN    1.0
2    None None        1     False   False  False        NaN    1.0
3  False Ocean        1     False   False  False        NaN    1.0
4    None None        1     False   False  False        NaN    1.0

   ReusedCount Serial   Longitude   Latitude  Class
0            0  B0003  -80.577366  28.561857      0
1            0  B0005  -80.577366  28.561857      0
2            0  B0007  -80.577366  28.561857      0
3            0  B1003 -120.610829  34.632093      0
4            0  B1004  -80.577366  28.561857      0

sns.catplot(y="PayloadMass", x="FlightNumber", hue="Class", data=df,
aspect=2.5)
plt.xlabel("Flight Number", fontsize=20)
plt.ylabel("Payload Mass (kg)", fontsize=20)
plt.show()
```



TASK 1: Visualize the relationship between Flight Number and Launch Site

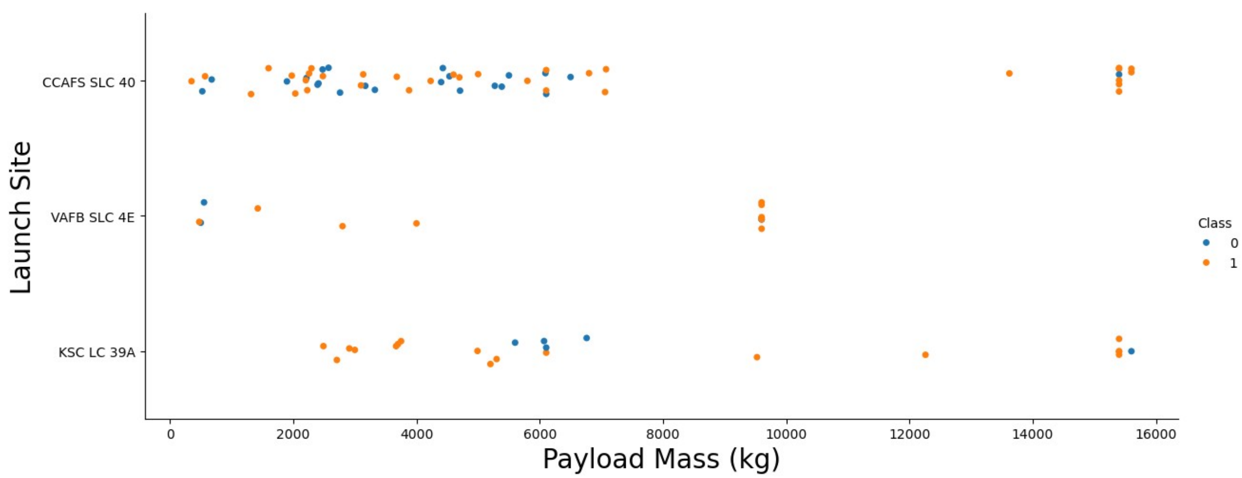
```
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df,
aspect=2.5)
plt.xlabel("Flight Number", fontsize=20)
```

```
plt.ylabel("Launch Site)", fontsize=20)
plt.show()
```



TASK 2: Visualize the relationship between Payload Mass and Launch Site

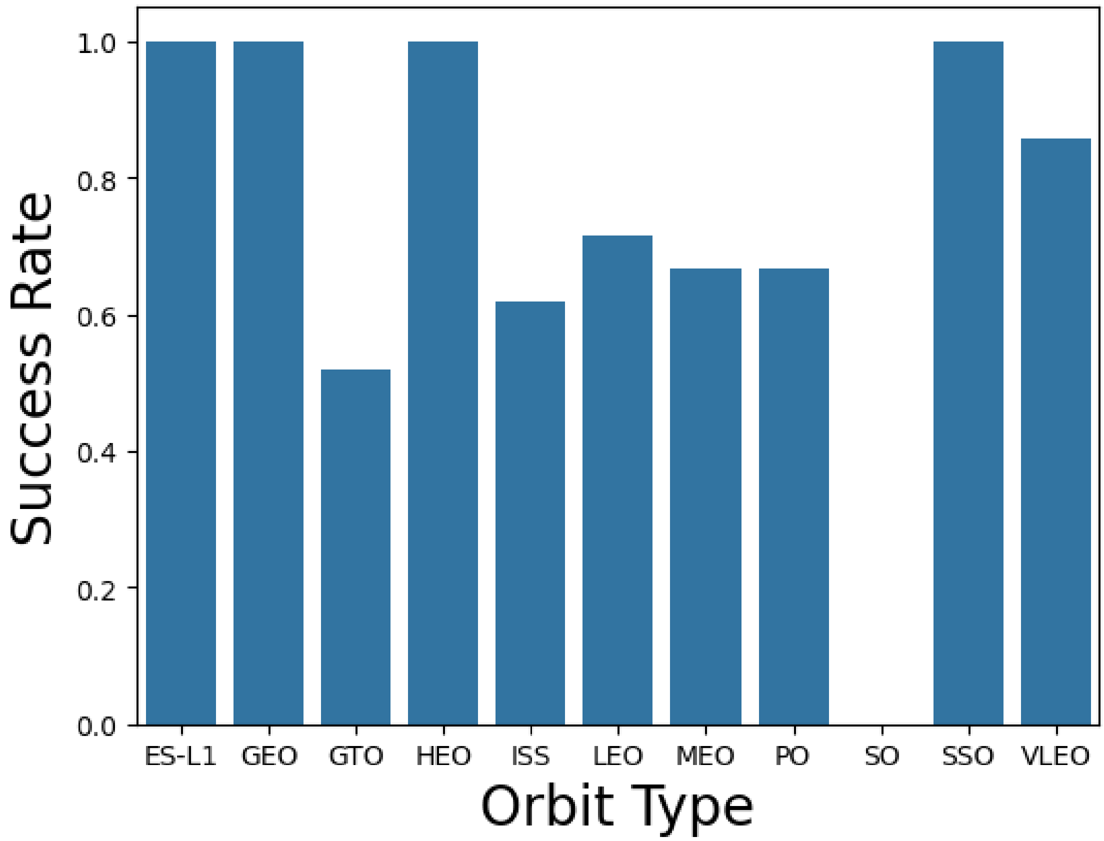
```
sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df,
aspect=2.5)
plt.xlabel("Payload Mass (kg)", fontsize=20)
plt.ylabel("Launch Site", fontsize=20)
plt.show()
```



TASK 3: Visualize the relationship between success rate of each orbit type¶

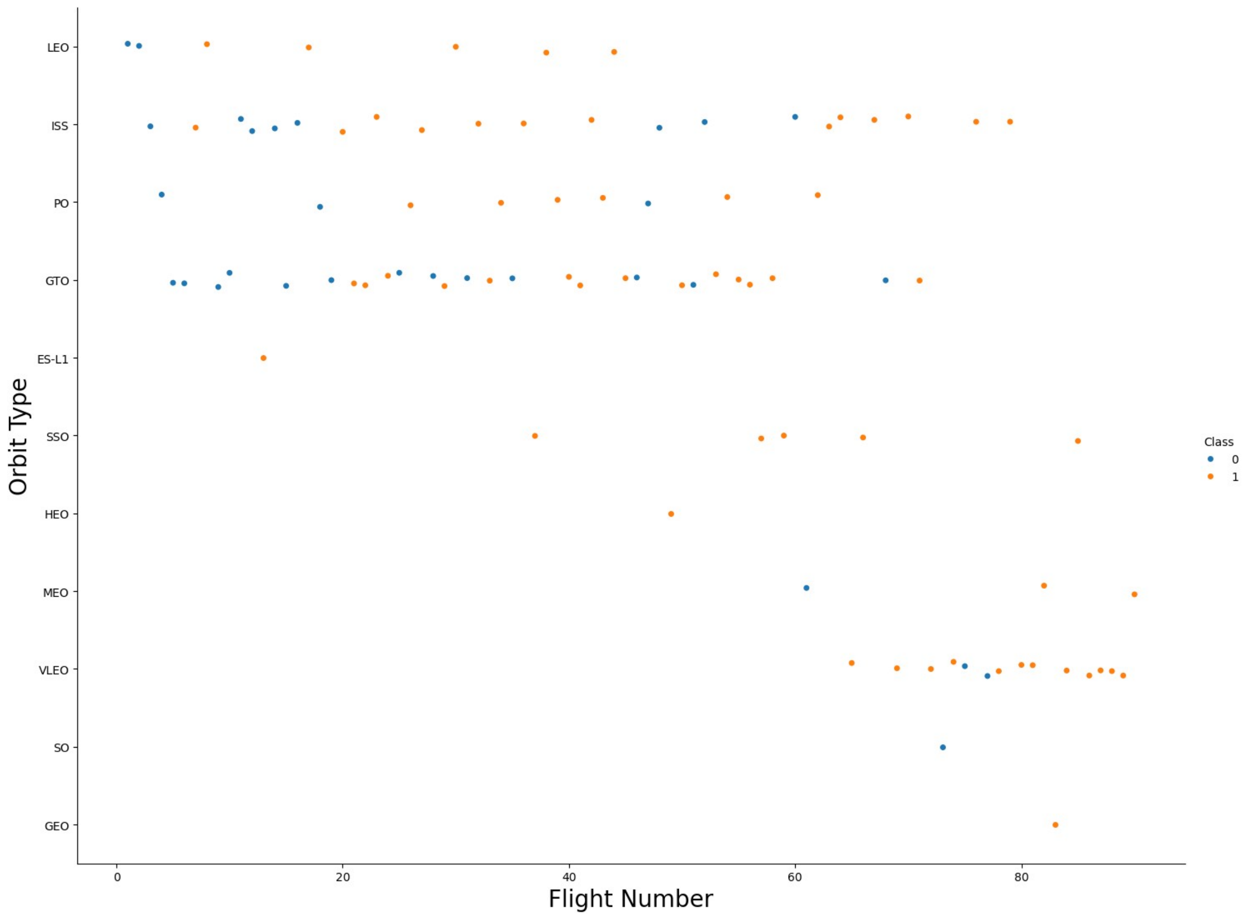
```
df_orbit = df.groupby(df['Orbit'], as_index=False).agg({"Class":
"mean"})
#df_orbit
sns.barplot(y="Class", x="Orbit", data=df_orbit)
```

```
plt.xlabel("Orbit Type", fontsize=20)
plt.ylabel("Success Rate", fontsize=20)
plt.show()
```



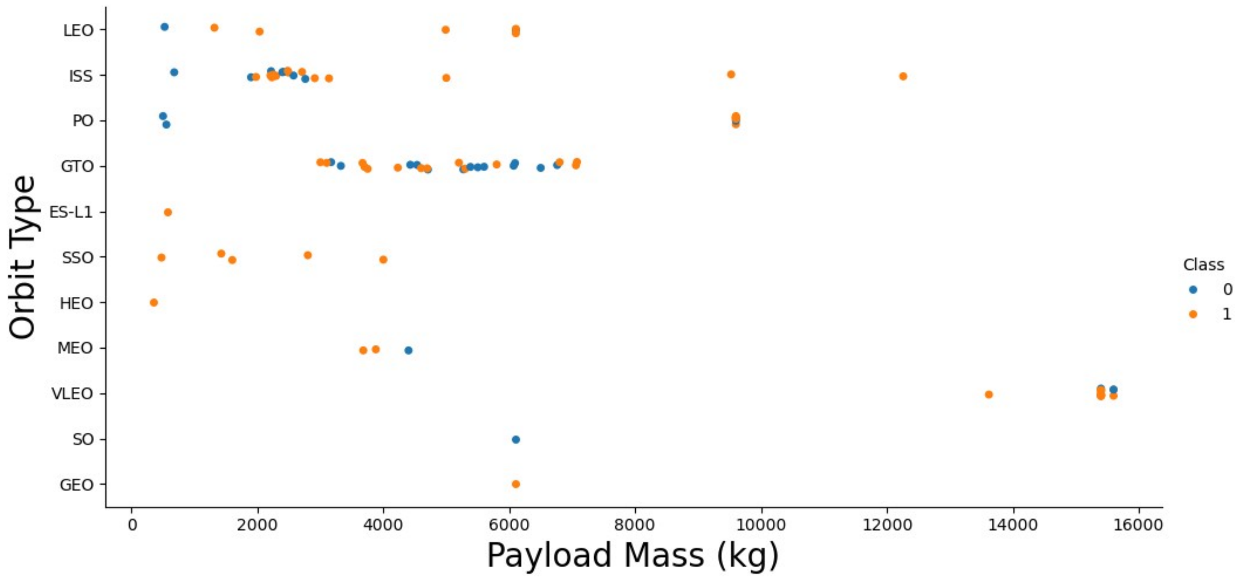
TASK 4: Visualize the relationship between FlightNumber and Orbit type

```
sns.catplot(y="Orbit", x="FlightNumber", hue="Class", data=df,
aspect=1.3, height=11)
plt.xlabel("Flight Number", fontsize=20)
plt.ylabel("Orbit Type", fontsize=20)
plt.show()
```



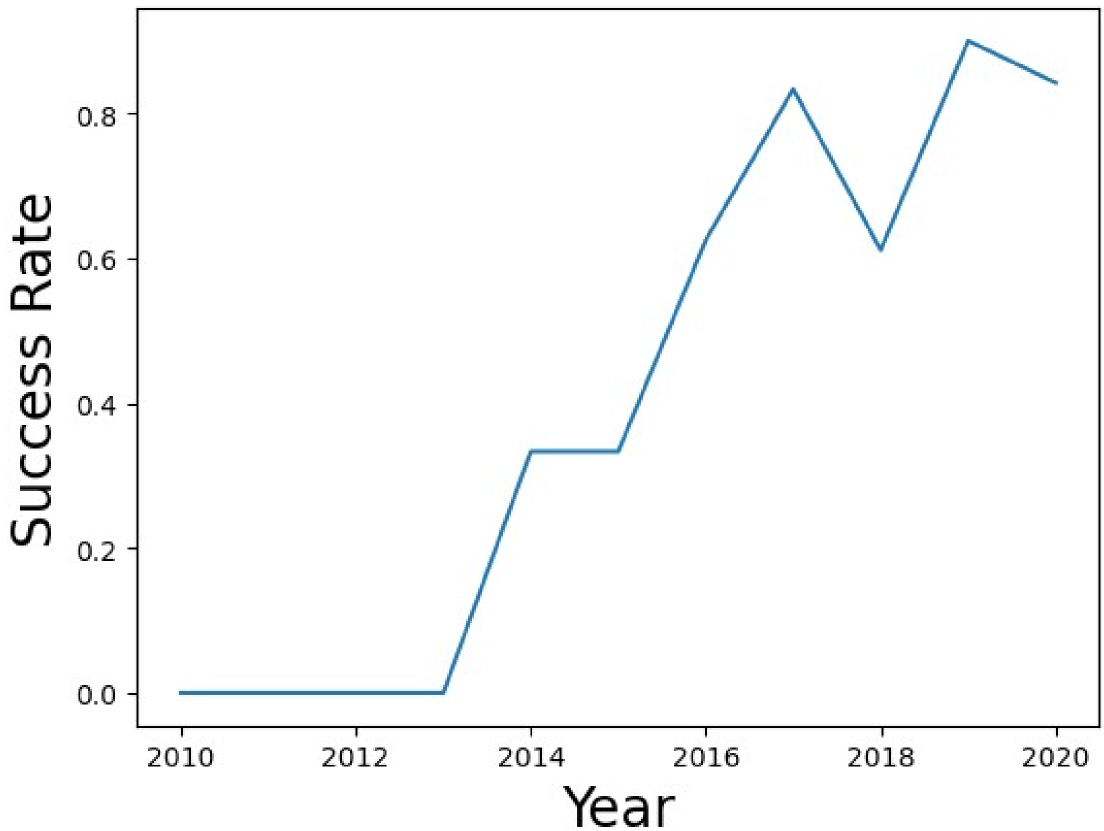
TASK 5: Visualize the relationship between Payload Mass and Orbit type

```
# Plot a scatter point chart with x axis to be Payload Mass and y axis to be the Orbit, and hue to be the class value

sns.catplot(y="Orbit", x="PayloadMass", hue="Class", data=df, aspect=2)
plt.xlabel("Payload Mass (kg)", fontsize=20)
plt.ylabel("Orbit Type", fontsize=20)
plt.show()
```



TASK 6: Visualize the launch success yearly trend

```
# add year column
df["Year"] = pd.DatetimeIndex(df["Date"]).year.astype(int)

df_year = df.groupby(df['Year'], as_index=False).agg({"Class":
"mean"})
#df_orbit
sns.lineplot(y="Class", x="Year", data=df_year)
plt.xlabel("Year", fontsize=20)
plt.ylabel("Success Rate", fontsize=20)
plt.show()
```



Features Engineering

We will select the features that will be used in success prediction in the future module.

```
features = df[['FlightNumber', 'PayloadMass', 'Orbit', 'LaunchSite',
'Flights', 'GridFins', 'Reused', 'Legs', 'LandingPad', 'Block',
'ReusedCount', 'Serial']]
features.head()

   FlightNumber  PayloadMass Orbit    LaunchSite  Flights  GridFins
Reused  \
0             1  6104.959412   LEO  CCAFS SLC 40        1     False
False
1             2   525.000000   LEO  CCAFS SLC 40        1     False
False
2             3   677.000000   ISS  CCAFS SLC 40        1     False
False
3             4   500.000000    PO   VAFB SLC 4E        1     False
False
4             5  3170.000000   GTO  CCAFS SLC 40        1     False
False
```

```
    Legs LandingPad  Block  ReusedCount Serial
0  False        NaN    1.0            0  B0003
1  False        NaN    1.0            0  B0005
2  False        NaN    1.0            0  B0007
3  False        NaN    1.0            0  B1003
4  False        NaN    1.0            0  B1004
```

TASK 7: Create dummy variables to categorical columns

Use the function get_dummies and features dataframe to apply OneHotEncoder to the column Orbits, LaunchSite, LandingPad, and Serial. Assign the value to the variable features_one_hot, display the results using the method head. Your result dataframe must include all features including the encoded ones.

TASK 8: Cast all numeric columns to float64

Now that our features_one_hot dataframe only contains numbers, cast the entire dataframe to variable type float64

```
features_one_hot = features_one_hot.astype(float)
features_one_hot.dtypes
```

```
FlightNumber    float64
PayloadMass     float64
Flights         float64
GridFins        float64
Reused          float64
                 ...
Serial_B1056    float64
Serial_B1058    float64
Serial_B1059    float64
Serial_B1060    float64
Serial_B1062    float64
Length: 81, dtype: object
```