



SpaceX Falcon 9 First Stage Landing Prediction

Assignment: Exploring and Preparing Data

Estimated time needed: **70** minutes

In this assignment, we will predict if the Falcon 9 first stage will land successfully. SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is due to the fact that SpaceX can reuse the first stage.

In this lab, you will perform Exploratory Data Analysis and Feature Engineering.

Falcon 9 first stage will land successfully

▼ Objectives

Perform exploratory Data Analysis and Feature Engineering using `Pandas` and `Matplotlib`

- Exploratory Data Analysis
- Preparing Data Feature Engineering

Import Libraries and Define Auxiliary Functions

We will import the following libraries the lab

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

[2]: # 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 wit
      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
      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
      import seaborn as sns
```

```
[ ]: ## Exploratory Data Analysis
```

First, let's read the SpaceX dataset into a Pandas dataframe and print its summary

```
[3]: 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)
```

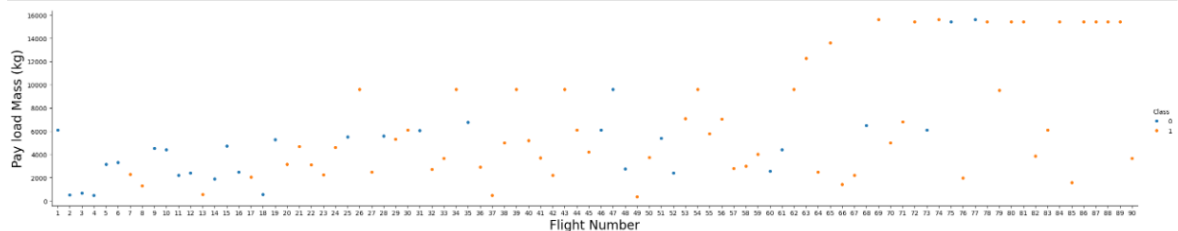
```
[3]:
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCou
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	

First, let's try to see how the `FlightNumber` (indicating the continuous launch attempts.) and `Payload` variables would affect the launch outcome.

We can plot out the `FlightNumber` vs. `PayloadMass` and overlay the outcome of the launch. We see that as the flight number increases, the first stage is more likely to land successfully. The payload mass is also important; it seems the more massive the payload, the less likely the first stage will return.

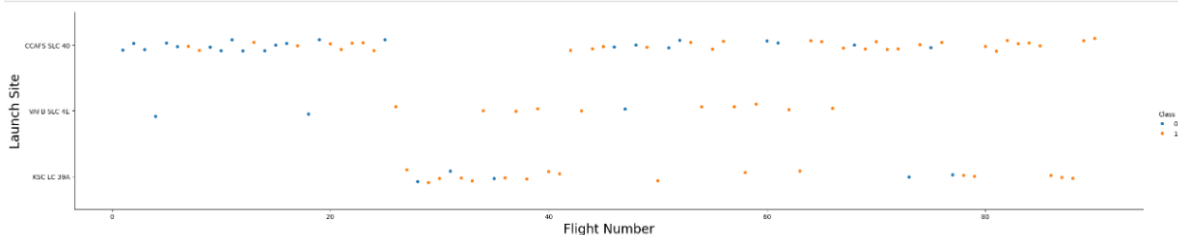
```
[4]: sns.catplot(y="PayloadMass", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number", fontsize=20)
plt.ylabel("Pay load Mass (kg)", fontsize=20)
plt.show()
```



We see that different launch sites have different success rates. `CCAFS LC-40` has a success rate of 60 %, while `KSC LC-39A` and `VAFB SLC 4E` has a success rate of 77%.

Next, let's drill down to each site visualize its detailed launch records.

```
[5]: ### TASK 1: Visualize the relationship between Flight Number and Launch Site
sns.catplot(y='LaunchSite', x='FlightNumber', hue='Class', data=df, aspect=5)
plt.xlabel("Flight Number", fontsize=20)
plt.ylabel("Launch Site", fontsize=20)
plt.show()
```



Use the function `catplot` to plot `FlightNumber` vs `LaunchSite`, set the parameter `x` parameter to `FlightNumber`, set the `y` to `Launch Site` and set the parameter `hue` to `'class'`

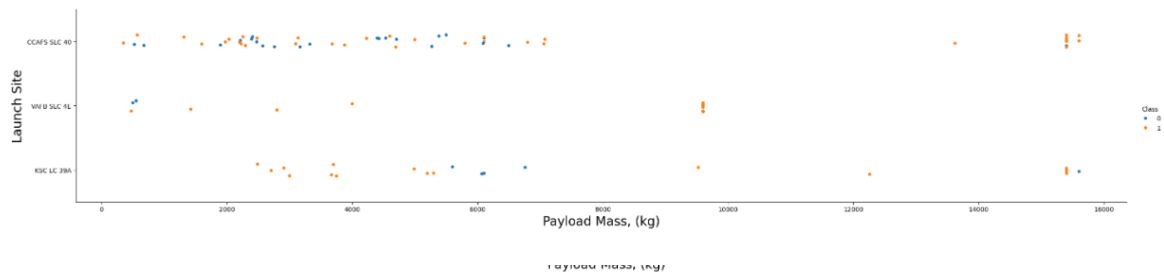
```
[ ]: # Plot a scatter point chart with x axis to be Flight Number and y axis to be the Launch site, and hue to be the class value  
## As Flight Number increases, there are relatively more successes (class=1) at all sites
```

Now try to explain the patterns you found in the Flight Number vs. Launch Site scatter point plots.

```
[ ]: ### TASK 2: Visualize the relationship between Payload and Launch Site
```

We also want to observe if there is any relationship between launch sites and their payload mass.

```
[6]: # Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the Launch site, and hue to be the class value  
sns.catplot(y='LaunchSite', x='PayloadMass', hue='Class', data=df, aspect=5)  
plt.xlabel("Payload Mass, (kg)", fontsize=20)  
plt.ylabel("Launch Site", fontsize=20)  
plt.show()
```



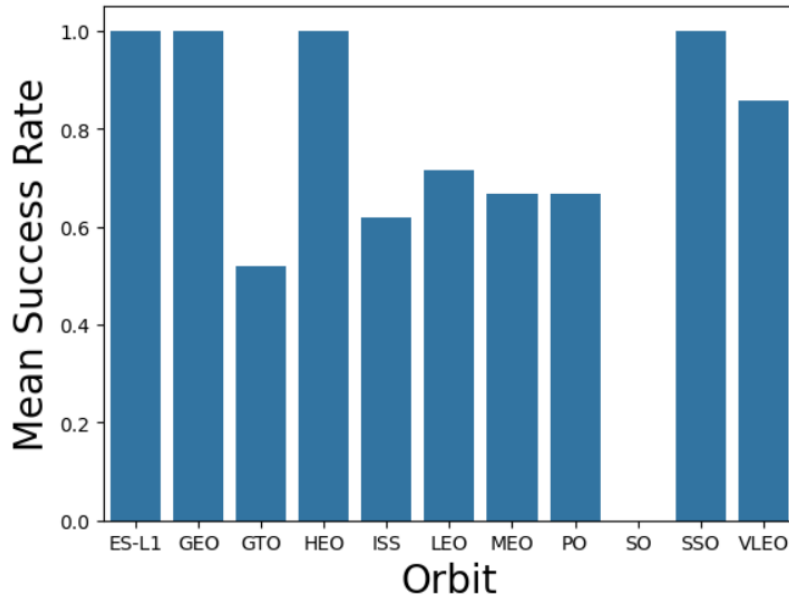
Now if you observe Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

```
[ ]: ### TASK 3: Visualize the relationship between success rate of each orbit type
```

Next, we want to visually check if there are any relationship between success rate and orbit type.

Let's create a `bar chart` for the success rate of each orbit

```
[7]: # HINT use groupby method on Orbit column and get the mean of Class column
orbit_success = df.groupby(['Orbit'])['Class'].mean()
sns.barplot(data=orbit_success)
plt.xlabel("Orbit", fontsize=20)
plt.ylabel("Mean Success Rate", fontsize=20)
plt.show()
```



```
[8]: orbit_success
```

```
[8]: Orbit
ES-L1    1.000000
GEO      1.000000
GTO      0.518519
HEO      1.000000
ISS      0.619048
LEO      0.714286
MEO      0.666667
PO       0.666667
SO       0.000000
SSO      1.000000
VLEO     0.857143
Name: Class, dtype: float64
```

Analyze the plotted bar chart try to find which orbits have high sucess rate.

ES-L1, GEO, HEO, and SSO have success rates of 100%. VLEO have a success rathe of approximately 86%. Because we used the mean, we do not know how many data points are used for each of these. Below we see there is only 1 data point for ES-L1, GEO, and HEO, which seems to provide weak evidence for correlation due to low sample size. There are 5 data points for SSO, which provides better evidence for that orbit. There are 12 data points for VLEO, so the mean accuracy of 86% seems credible.

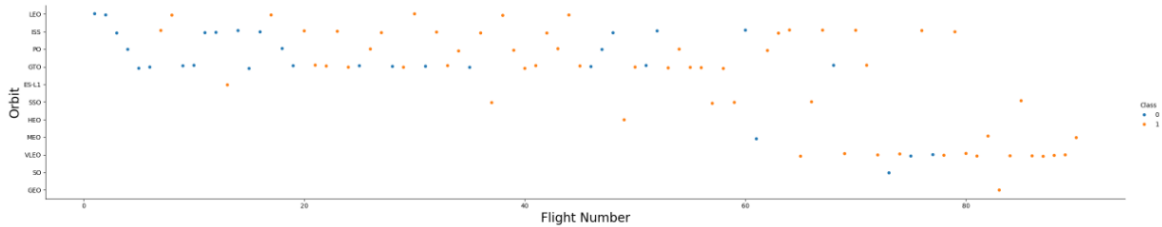
```
[9]: orbit_success_counts = df.groupby(['Orbit'])['Class'].value_counts()
orbit_success_counts
```

```
[9]: Orbit  Class
     ES-L1  1      1
     GEO    1      1
     GTO    1     14
           0     13
     HEO    1      1
     ISS    1     13
           0      8
     LEO    1      5
           0      2
     MEO    1      2
           0      1
     PO     1      6
           0      3
     SO     0      1
     SSO    1      5
     VLEO   1     12
           0      2
     Name: Class, dtype: int64
```

```
[ ]: ### TASK 4: Visualize the relationship between FlightNumber and Orbit type
```

For each orbit, we want to see if there is any relationship between FlightNumber and Orbit type.

```
[10]: # Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value
sns.catplot(y='Orbit', x='FlightNumber', hue='Class', data=df, aspect=5)
plt.xlabel("Flight Number", fontsize=20)
plt.ylabel("Orbit", fontsize=20)
plt.show()
```

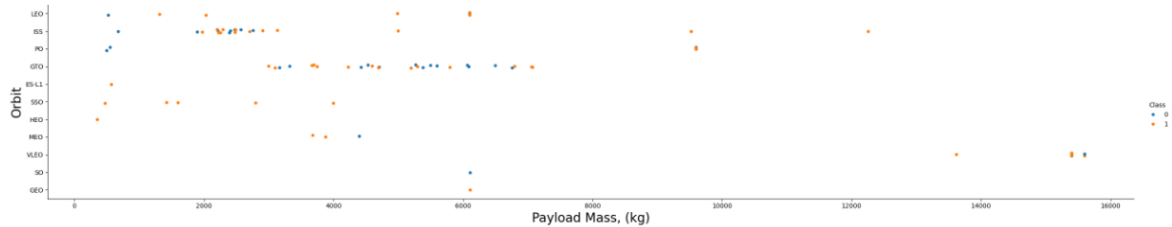


You should see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

```
[ ]: ### TASK 5: Visualize the relationship between Payload and Orbit type
```

Similarly, we can plot the Payload vs. Orbit scatter point charts to reveal the relationship between Payload and Orbit type

```
[11]: # Plot a scatter point chart with x axis to be Payload 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=5)
plt.xlabel("Payload Mass, (kg)", fontsize=20)
plt.ylabel("Orbit", fontsize=20)
plt.show()
```



With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccesful mission) are both there here.

```
[ ]: ### TASK 6: Visualize the Launch success yearly trend
```

You can plot a line chart with x axis to be Year and y axis to be average success rate, to get the average launch success trend.

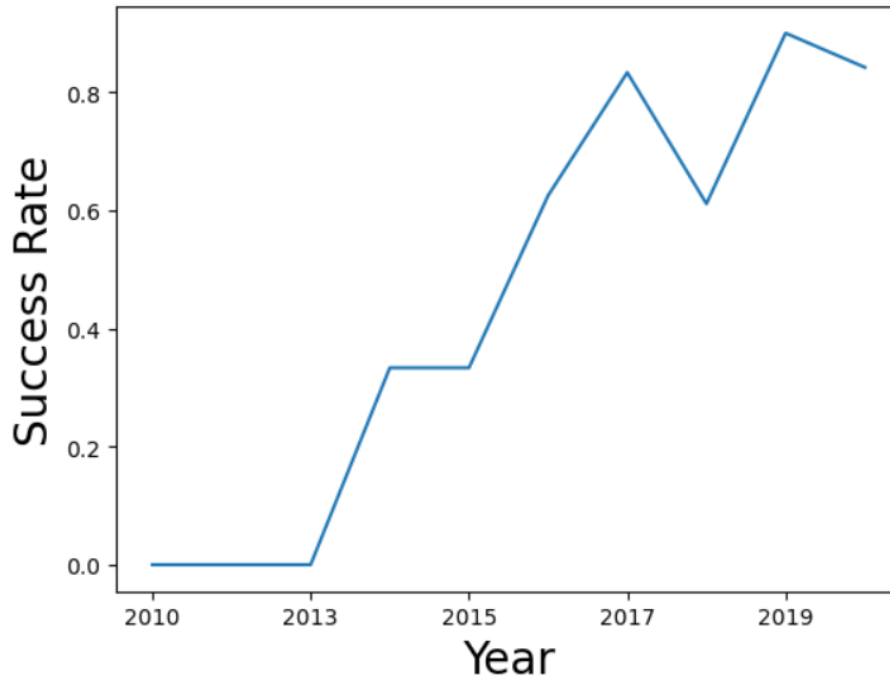
The function will help you get the year from the date:

```
[12]: # A function to Extract years from the date
year=[]
def Extract_year():
    for i in df["Date"]:
        year.append(i.split("-")[0])
    return year
Extract_year()
df['Date'] = year
df.head()
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCour
0	1	2010	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	
1	2	2012	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	
2	3	2013	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	
3	4	2013	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	
4	5	2013	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	

```
[13]: # Plot a line chart with x axis to be the extracted year and y axis to be the success rate
```

```
[13]: # Plot a line chart with x axis to be the extracted year and y axis to be the success rate
success_by_year = df.groupby(['Date'])['Class'].mean()
success_by_year.plot.line()
plt.xlabel("Year", fontsize=20)
plt.ylabel("Success Rate", fontsize=20)
plt.show()
```



you can observe that the success rate since 2013 kept increasing till 2020

```
[ ]: ## Features Engineering
```

By now, you should obtain some preliminary insights about how each important variable would affect the success rate, we will select the features that will be used in success prediction in the future module.

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

```
[14]:
```

	FlightNumber	PayloadMass	Orbit	LaunchSite	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial
0	1	6104.959412	LEO	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B0003
1	2	525.000000	LEO	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B0005
2	3	677.000000	ISS	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B0007
3	4	500.000000	PO	VAFB SLC 4E	1	False	False	False	NaN	1.0	0	B1003
4	5	3170.000000	GTO	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B1004

```
[15]: ### TASK 7: Create dummy variables to categorical columns
features_one_hot = pd.get_dummies(data=features, columns=['Orbit', 'LaunchSite', 'LandingPad', 'Serial'])
features_one_hot.head()
```

```
[15]:
```

	FlightNumber	PayloadMass	Flights	GridFins	Reused	Legs	Block	ReusedCount	Orbit_ES-L1	Orbit_GEO	...	Serial_B1048	Serial_B1049	Serial_B1
0	1	6104.959412	1	False	False	False	1.0	0	0	0	...	0	0	
1	2	525.000000	1	False	False	False	1.0	0	0	0	...	0	0	
2	3	677.000000	1	False	False	False	1.0	0	0	0	...	0	0	
3	4	500.000000	1	False	False	False	1.0	0	0	0	...	0	0	
4	5	3170.000000	1	False	False	False	1.0	0	0	0	...	0	0	

5 rows × 80 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.

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

```
[16]: features_one_hot = features_one_hot.astype('float64')
features_one_hot.head()
# HINT: use astype function
```

```
[16]:
```

	FlightNumber	PayloadMass	Flights	GridFins	Reused	Legs	Block	ReusedCount	Orbit_ES-L1	Orbit_GEO	...	Serial_B1048	Serial_B1049	Serial_B1
0	1.0	6104.959412	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	
1	2.0	525.000000	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	
2	3.0	677.000000	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	
3	4.0	500.000000	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	
4	5.0	3170.000000	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	

5 rows × 80 columns



We can now export it to a **CSV** for the next section, but to make the answers consistent, in the next lab we will provide data in a pre-selected date range.


```
[17]: features_one_hot.dtypes
```

```
[17]: 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: 80, dtype: object
```

```
features_one_hot.to_csv('dataset_part_3.csv', index=False)
```

Authors

[Pratiksha Verma](#)