



# 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



Several examples of an unsuccessful landing are shown here:



Most unsuccessful landings are planned. Space X performs a controlled landing in the oceans.

## 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

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

```
In [2]: # pandas is a software library written for the Python programming language
import pandas as pd
#NumPy is a library for the Python programming language, adding support for
import numpy as np
# Matplotlib is a plotting library for python and pyplot gives us a MatLab like
import matplotlib.pyplot as plt
#Seaborn is a Python data visualization library based on matplotlib. It provides
import seaborn as sns
```

```
<ipython-input-2-cde6ab162d36>:2: DeprecationWarning:
Pyyarrow will become a required dependency of pandas in the next major release of pandas (pandas 3.0),
(to allow more performant data types, such as the Arrow string type, and better interoperability with other libraries)
but was not found to be installed on your system.
If this would cause problems for you,
please provide us feedback at https://github.com/pandas-dev/pandas/issues/54466
```

```
import pandas as pd
```

## Exploratory Data Analysis

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

```
In [3]: from js import fetch
import io

URL = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IE
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)
```

```
Out[3]:
```

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

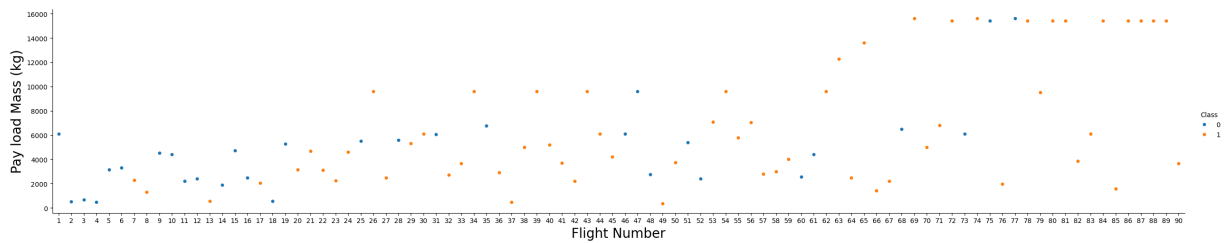
```
In [20]: df['Class'].value_counts()
```

```
Out[20]: Class
1      60
0      30
Name: count, dtype: int64
```

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 also appears to be a factor; even with more massive payloads, the first stage often returns successfully.

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



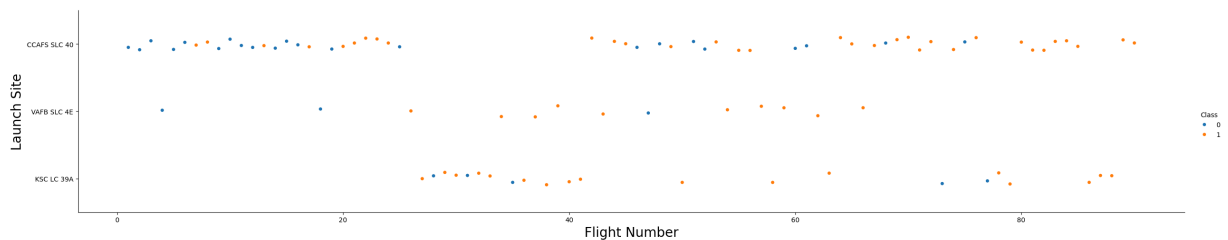
Next, let's drill down to each site visualize its detailed launch records.

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

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'`

```
In [8]: # Plot a scatter point chart with x axis to be Flight Number and y axis to be Launch Site
sns.catplot(y="LaunchSite", x="FlightNumber", hue='Class', data=df, aspect=
plt.xlabel("Flight Number", fontsize=20)
plt.ylabel("Launch Site", fontsize=20)
```

```
Out[8]: Text(32.35652916666665, 0.5, 'Launch Site')
```



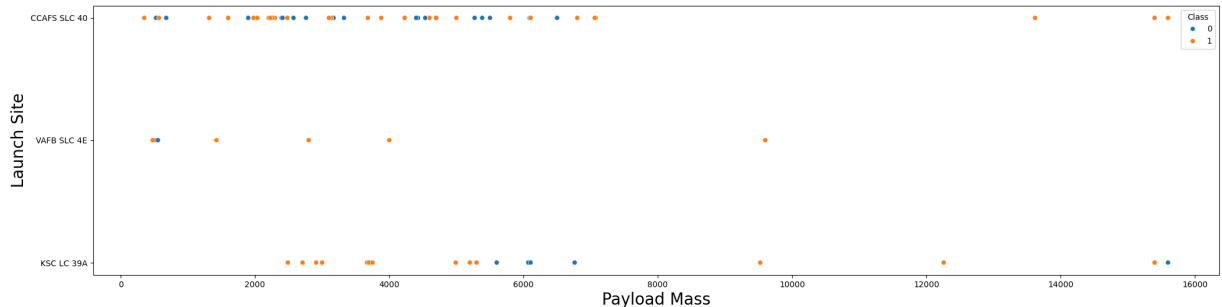
Now try to explain the patterns you found in the Flight Number vs. Launch Site scatter point plots.

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

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

```
In [19]: # Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis
plt.figure(figsize=(25,6))
sns.scatterplot(x="PayloadMass", y="LaunchSite", hue='Class', data=df)
plt.xlabel("Payload Mass", fontsize=20)
plt.ylabel("Launch Site", fontsize=20)
```

```
Out[19]: Text(0, 0.5, 'Launch Site')
```



Now if you observe Payload Mass 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).

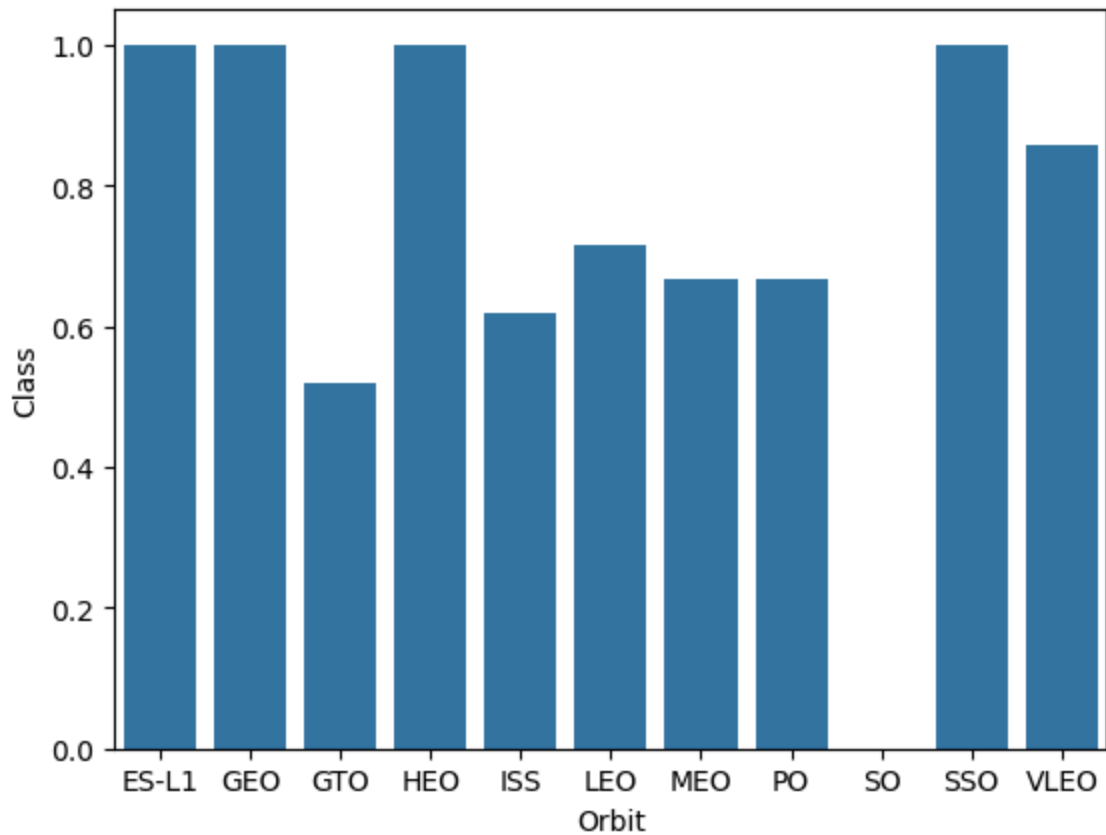
### TASK 3: Visualize the relationship between success rate of each orbit type

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

```
In [22]: # HINT use groupby method on Orbit column and get the mean of Class column
success_rate = df.groupby("Orbit")["Class"].mean().reset_index()
sns.barplot(x=success_rate["Orbit"], y=success_rate["Class"])
```

```
Out[22]: <AxesSubplot:xlabel='Orbit', ylabel='Class'>
```



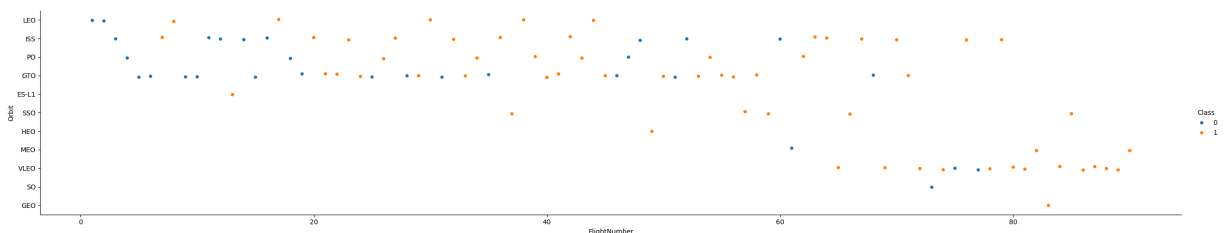
Analyze the plotted bar chart to identify which orbits have the highest success rates.

## 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.

```
In [25]: # Plot a scatter point chart with x axis to be FlightNumber and y axis to be Orbit
sns.catplot(data=df, x="FlightNumber", y="Orbit", hue='Class', aspect=5)
```

```
Out[25]: <seaborn.axisgrid.FacetGrid at 0x9f99198>
```



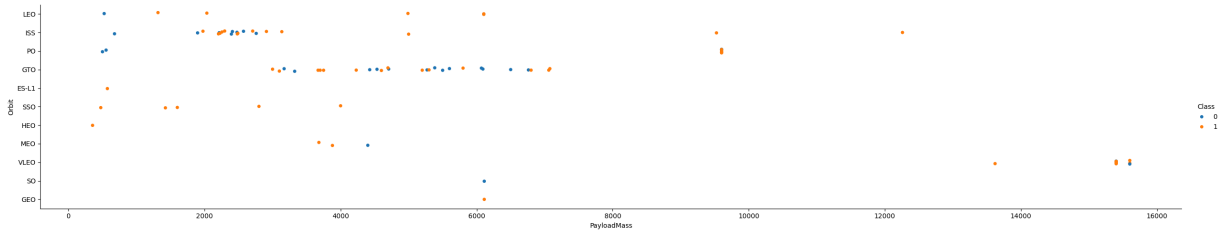
You can observe that in the LEO orbit, success seems to be related to the number of flights. Conversely, in the GTO orbit, there appears to be no relationship between flight number and success.

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

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

```
In [26]: # Plot a scatter point chart with x axis to be Payload Mass and y axis to be Orbit
sns.catplot(data=df, x="PayloadMass", y="Orbit", hue='Class', aspect=5)
```

```
Out[26]: <seaborn.axisgrid.FacetGrid at 0xbde71e0>
```



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

However, for GTO, it's difficult to distinguish between successful and unsuccessful landings as both outcomes are present.

## 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:

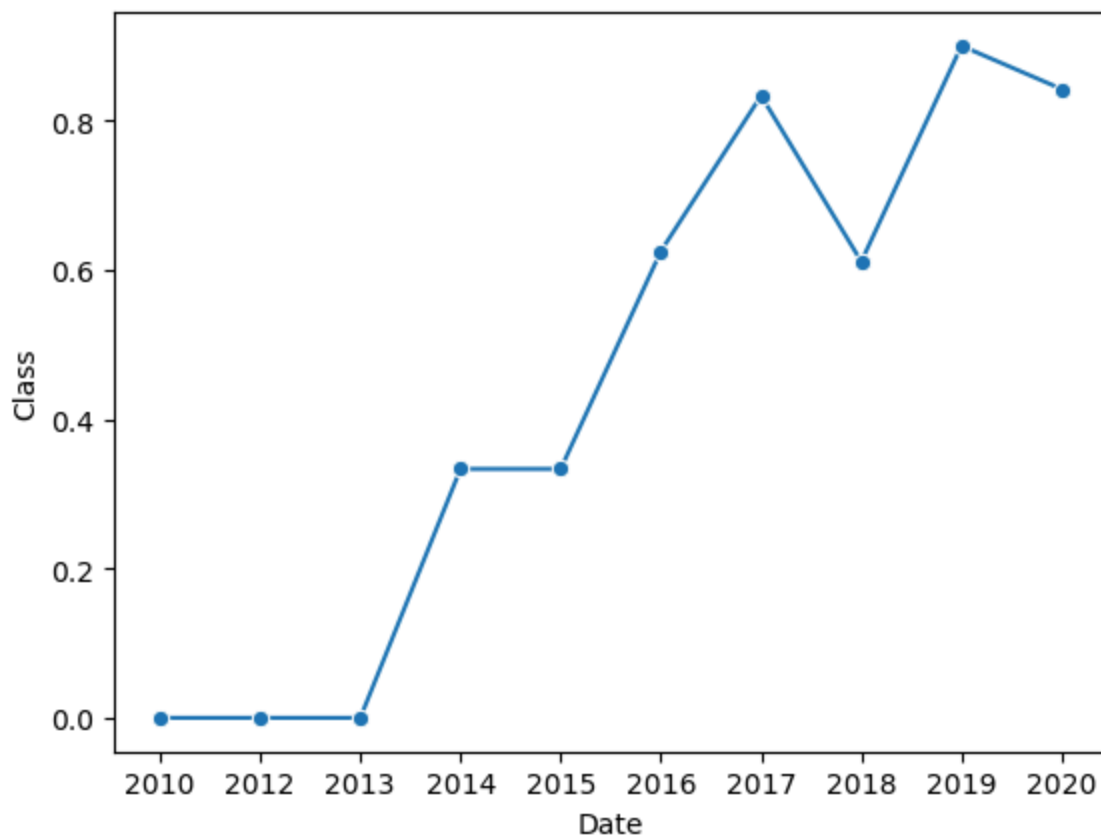
```
In [27]: # A function to Extract years from the date
year=[]
def Extract_year():
    for i in df["Date"]:
        year.append(i.split("-")[0])
    return year
Extract_year()
df['Date'] = year
df.head()
```

Out [27]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	FI
0	1	2010	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None	None
1	2	2012	Falcon 9	525.000000	LEO	CCAFS SLC 40	None	None
2	3	2013	Falcon 9	677.000000	ISS	CCAFS SLC 40	None	None
3	4	2013	Falcon 9	500.000000	PO	VAFB SLC 4E	False	Ocean
4	5	2013	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None	None

In [31]: `# Plot a line chart with x axis to be the extracted year and y axis to be the yearly_success_rate = df.groupby("Date")["Class"].mean().reset_index() sns.lineplot(data=yearly_success_rate, x="Date", y="Class", marker='o')`

Out [31]: `<AxesSubplot:xlabel='Date', ylabel='Class'>`



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.

```
In [32]: features = df[['FlightNumber', 'PayloadMass', 'Orbit', 'LaunchSite', 'Flights', 'GridFins', 'Reused', 'Legs', 'LandingPad']]
features.head()
```

```
Out [32]:
```

	FlightNumber	PayloadMass	Orbit	LaunchSite	Flights	GridFins	Reused	Legs	LandingPad
0	1	6104.959412	LEO	CCAFS SLC 40	1	False	False	False	None
1	2	525.000000	LEO	CCAFS SLC 40	1	False	False	False	None
2	3	677.000000	ISS	CCAFS SLC 40	1	False	False	False	None
3	4	500.000000	PO	VAFB SLC 4E	1	False	False	False	None
4	5	3170.000000	GTO	CCAFS SLC 40	1	False	False	False	None

## 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.

```
In [34]: # HINT: Use get_dummies() function on the categorical columns
features_one_hot = pd.get_dummies(features, columns=["Orbit", "LaunchSite", "LandingPad", "Serial"])
print(features_one_hot.head())
```

	FlightNumber	PayloadMass	Flights	GridFins	Reused	Legs	Block	\
0	1	6104.959412	1	False	False	False	1.0	
1	2	525.000000	1	False	False	False	1.0	
2	3	677.000000	1	False	False	False	1.0	
3	4	500.000000	1	False	False	False	1.0	
4	5	3170.000000	1	False	False	False	1.0	

	ReusedCount	Orbit_ES-L1	Orbit_GEO	...	Serial_B1048	Serial_B1049	\
0	0	False	False	...	False	False	
1	0	False	False	...	False	False	
2	0	False	False	...	False	False	
3	0	False	False	...	False	False	
4	0	False	False	...	False	False	

	Serial_B1050	Serial_B1051	Serial_B1054	Serial_B1056	Serial_B1058	\
0	False	False	False	False	False	
1	False	False	False	False	False	
2	False	False	False	False	False	
3	False	False	False	False	False	
4	False	False	False	False	False	

	Serial_B1059	Serial_B1060	Serial_B1062
0	False	False	False
1	False	False	False
2	False	False	False
3	False	False	False
4	False	False	False

[5 rows x 80 columns]

## 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`

```
In [ ]: # HINT: use astype function
features_one_hot.astype(_)
```

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.

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

## Authors

Pratiksha Verma

IBM Corporation 2022. All rights reserved.