Skills
Network

SpaceX Falcon 9 First Stage Landing Prediction

Hands-on Lab: Complete the EDA with Visualization

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

Install the below libraries

```
!pip install pandas
!pip install numpy
!pip install seaborn
!pip install matplotlib
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: numpy>=1.26.0 in /usr/local/lib/python3.12/dist-p
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packag
Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: seaborn in /usr/local/lib/python3.12/dist-package
Requirement already satisfied: numpy!=1.24.0,>=1.20 in /usr/local/lib/python3.12
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.12/dist-pac
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in /usr/local/lib/python3
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dis
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/di
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/di
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-packa
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dis
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.12
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packag
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-pack
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dis
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/di
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/di
Requirement already satisfied: numpy>=1.23 in /usr/local/lib/python3.12/dist-pac
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-packa
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dis
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.12
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packag
```

Import Libraries and Define Auxiliary Functions

We will import the following libraries the lab

```
# andas is a software library written for the Python programming language for c
import pandas as pd
#NumPy is a library for the Python programming language, adding support for lar
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 provide
import seaborn as sns
```

Exploratory Data Analysis

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

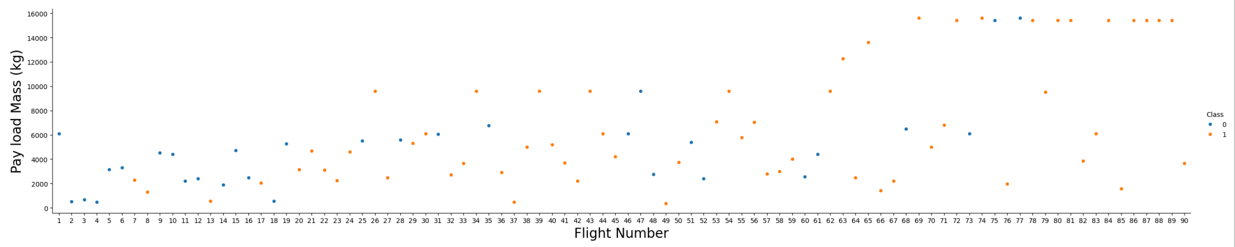
```
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cl

# If you were unable to complete the previous lab correctly you can uncomment a
#df = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomair

df.head(5)
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome
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

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.

```
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()
```

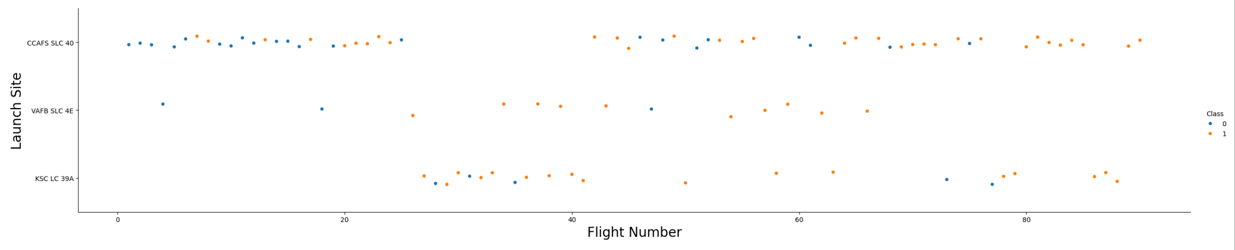


Next, let's drill down to each site visualize its detailed launch records.

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'`

```
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()
```



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

The CCAFS SLC 40 site appears to have the highest frequency of launches across the entire span of flight numbers. The VAFB SLC 4E and KSC LC 39A sites appear to have a lower number of total launches, often starting at later flight numbers.

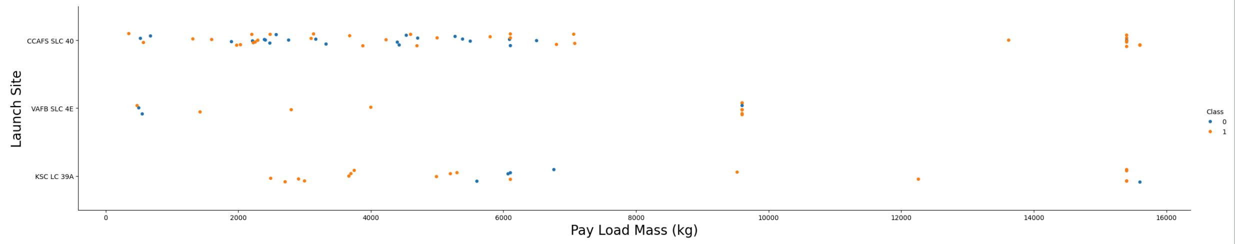
A clear pattern is that as the Flight Number increases, the launch success rate (indicated by the 'Class' hue) appears to improve for all launch sites. Also VAFB-SLC: This site seems to be associated with lower flight numbers initially, but like the others, shows a trend toward higher success in later flights.

KSC-LC 39A: This site appears to have started being used later in the Falcon 9 program (at higher flight numbers) and generally has a very high success rate from its earliest flights. This could be due to the fact that launches from this site benefited from the experience gained in the earlier CCAFS SLC 40 launches.

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.

```
sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("Pay Load Mass (kg)",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```



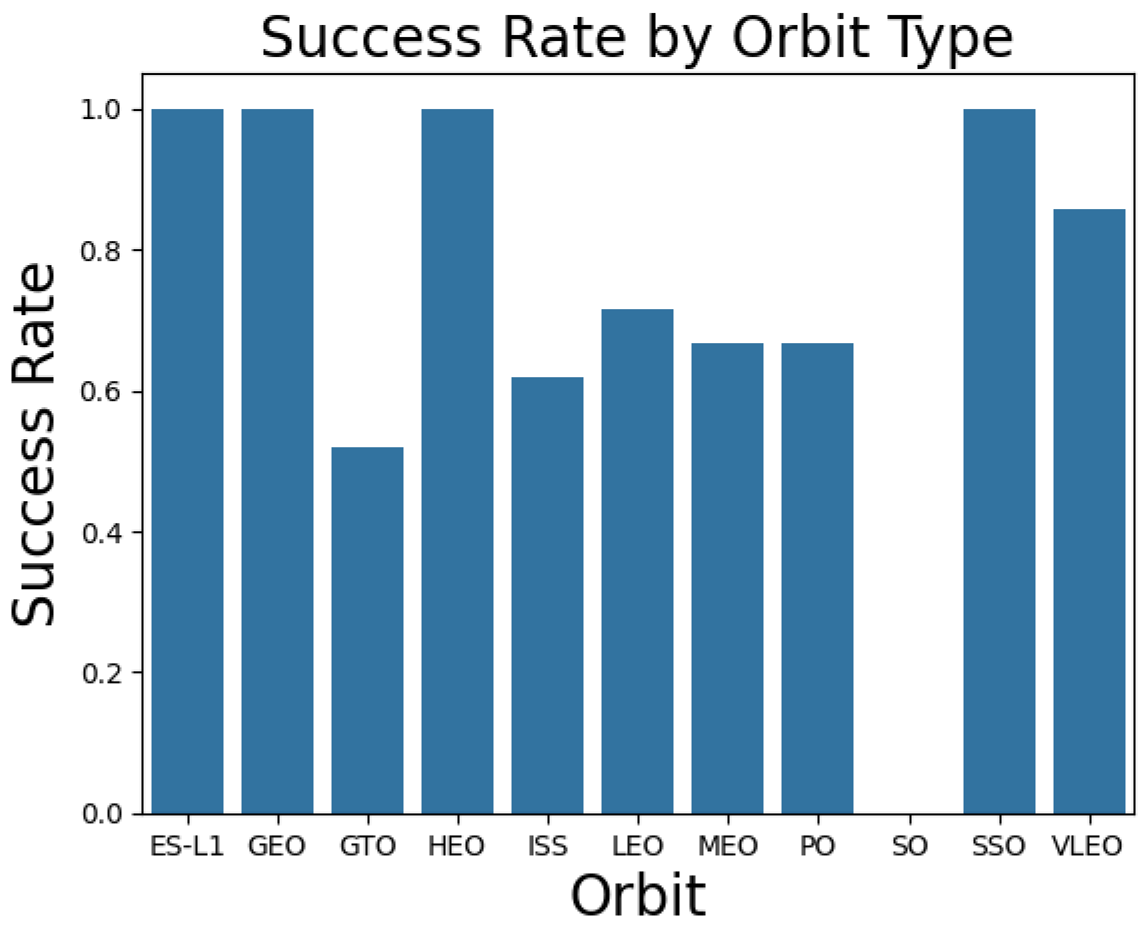
Now if you observe Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

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 sucess rate of each orbit

```
# HINT use groupby method on Orbit column and get the mean of Class column
orbit_success_rate = df.groupby('Orbit')['Class'].mean().reset_index()
sns.barplot(x='Orbit', y='Class', data=orbit_success_rate)
plt.xlabel('Orbit', fontsize=20)
plt.ylabel('Success Rate', fontsize=20)
plt.title('Success Rate by Orbit Type', fontsize=20)
plt.show()
```

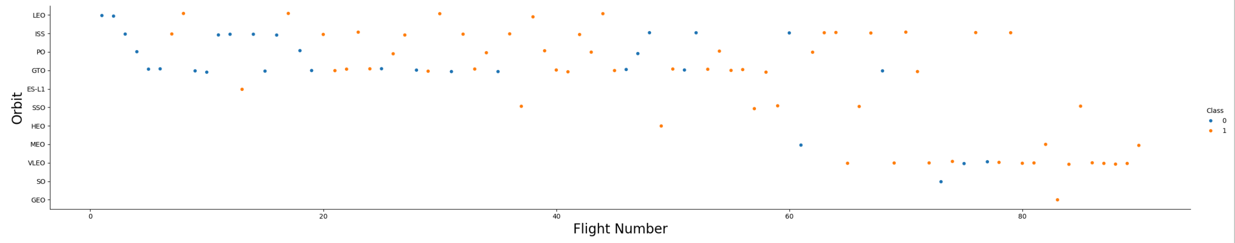


Analyze the ploted bar chart try to find which orbits have high sucess rate.

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.

```
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()
```

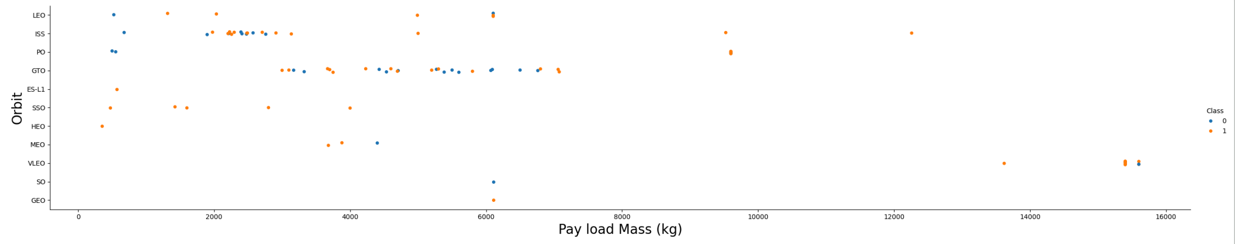


You should see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

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

```
sns.catplot(y="Orbit", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("Pay load 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(unsuccessful 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.

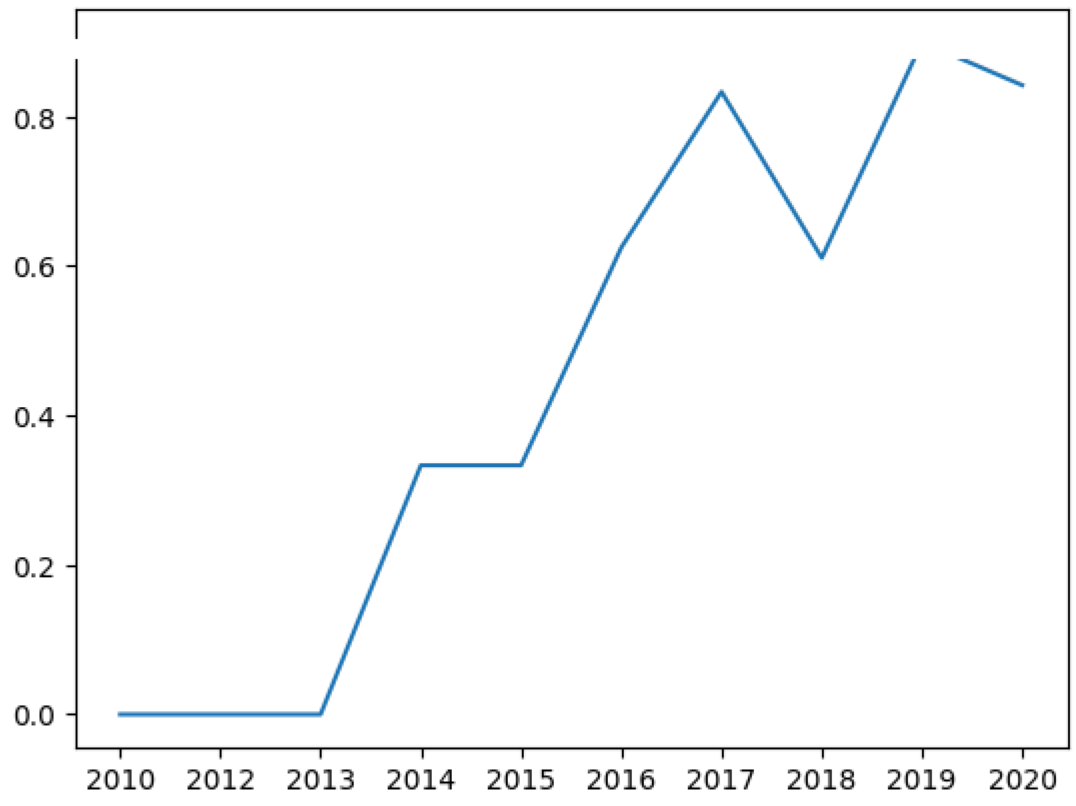
The function will help you get the year from the date:

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

```
# Plot a line chart with x axis to be the extracted year and y axis to be the s
df['year'] = Extract_year(df['Date'])
success_rate_by_year = df.groupby('year')['Class'].mean()
```

```
plt.plot(success_rate_by_year.index, success_rate_by_year.values)
```

```
[<matplotlib.lines.Line2D at 0x786fc482ae70>]
```



You can observe that the success rate since 2013 kept increasing till 2017 (stable in 2014) and after 2015 it started increasing.

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.

```
features = df[['FlightNumber', 'PayloadMass', 'Orbit', 'LaunchSite', 'Flights',
features.head()
```

	FlightNumber	PayloadMass	Orbit	LaunchSite	Flights	GridFins	Reused	Legs
0	1	6104.959412	LEO	CCAFS SLC 40	1	False	False	False
1	2	525.000000	LEO	CCAFS SLC 40	1	False	False	False
2	3	677.000000	ISS	CCAFS SLC 40	1	False	False	False

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.

```
# HINT: Use get_dummies() function on the categorical columns
features_one_hot = pd.get_dummies(features, columns=['Orbit', 'LaunchSite', 'La
display(features_one_hot.head())
```

	FlightNumber	PayloadMass	Flights	GridFins	Reused	Legs	Block	ReusedCoun
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	

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`

```
# HINT: use astype function
features_one_hot = features_one_hot.astype('float64')
```

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

Joseph Santarcangelo has a PhD in Electrical Engineering, his research focused on using machine learning, signal processing, and computer vision to determine how videos impact human cognition. Joseph has been working for IBM since he completed his PhD.

Nayef Abou Tayoun is a Data Scientist at IBM and pursuing a Master of Management in Artificial intelligence degree at Queen's University.

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2021-10-12	1.1	Lakshmi Holla	Modified markdown
2020-09-20	1.0	Joseph	Modified Multiple Areas
2020-11-10	1.1	Nayef	updating the input data

Copyright © 2020 IBM Corporation. All rights reserved.