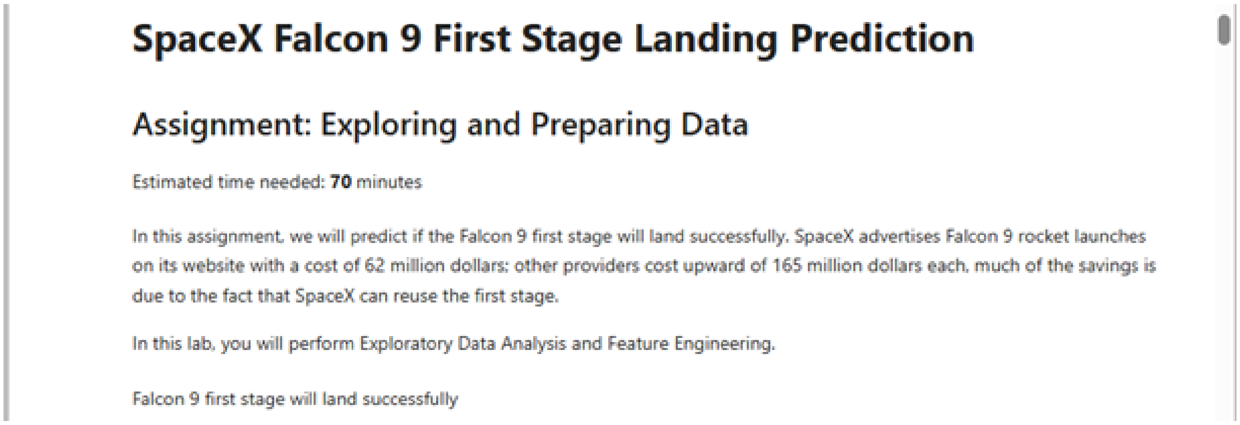


SpaceX Falcon 9 First Stage Landing Prediction

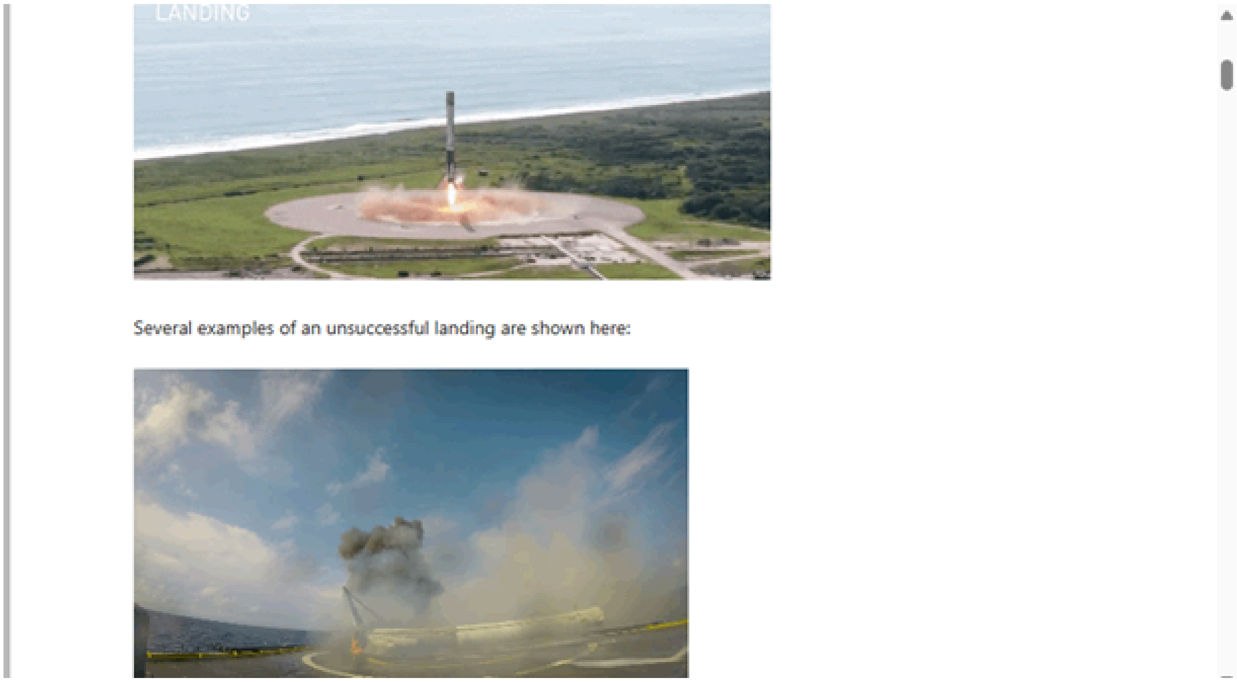
Assignment: Exploring and Preparing Data

Estimated time needed: **70** minutes

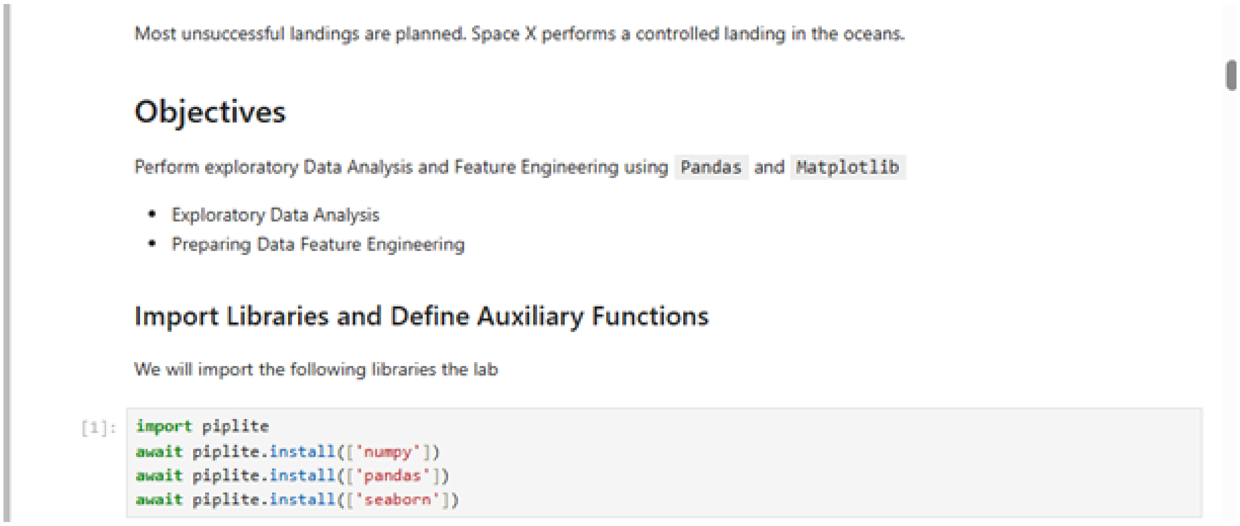
In this assignment, we will predict if the Falcon 9 first stage will land successfully. SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is due to the fact that SpaceX can reuse the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

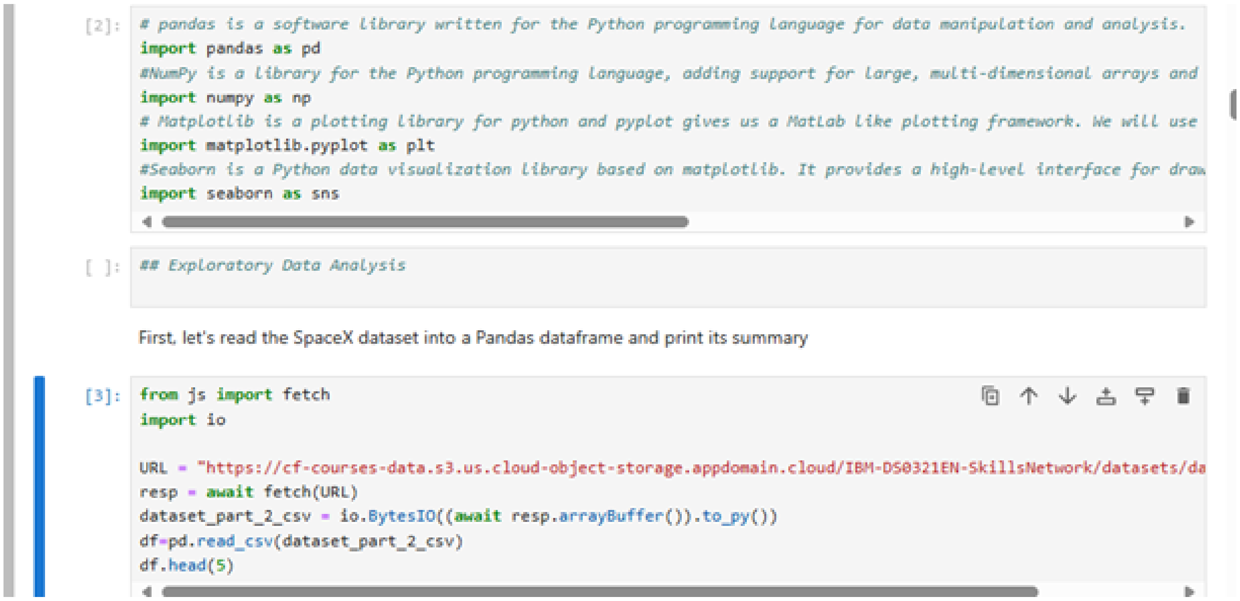
Perform exploratory Data Analysis and Feature Engineering using `Pandas` and `Matplotlib`

- Exploratory Data Analysis
- Preparing Data Feature Engineering

Import Libraries and Define Auxiliary Functions

We will import the following libraries the lab

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



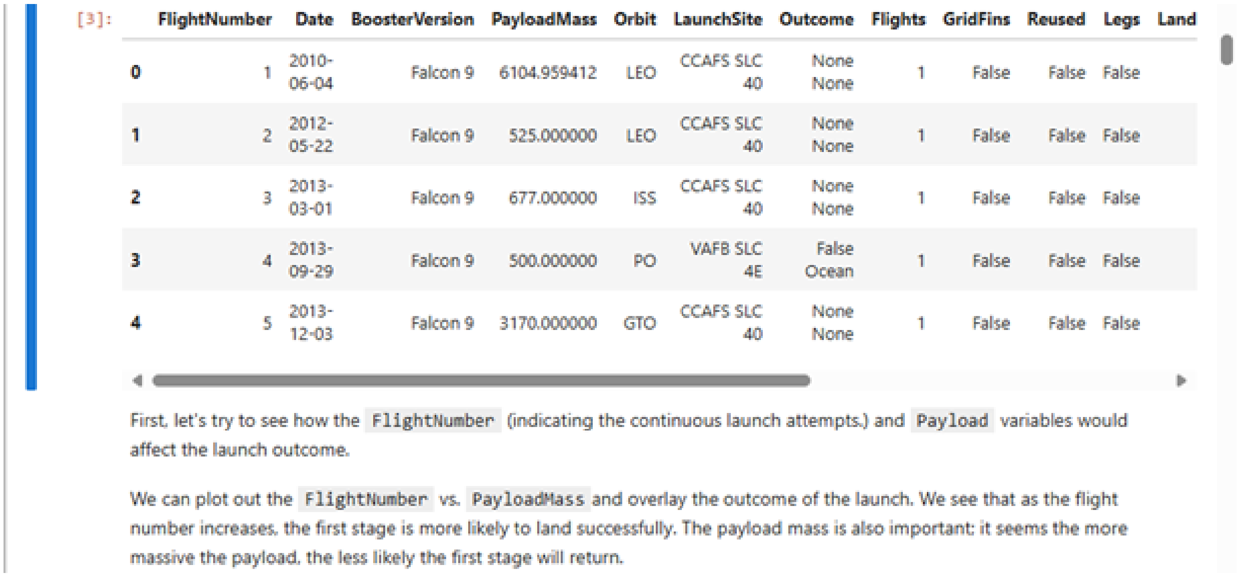
```
[2]: # pandas is a software library written for the Python programming language for data manipulation and analysis.
import pandas as pd
#NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and
import numpy as np
# Matplotlib is a plotting library for python and pyplot gives us a MatLab like plotting framework. We will use
import matplotlib.pyplot as plt
#Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for draw
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/da
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)
```

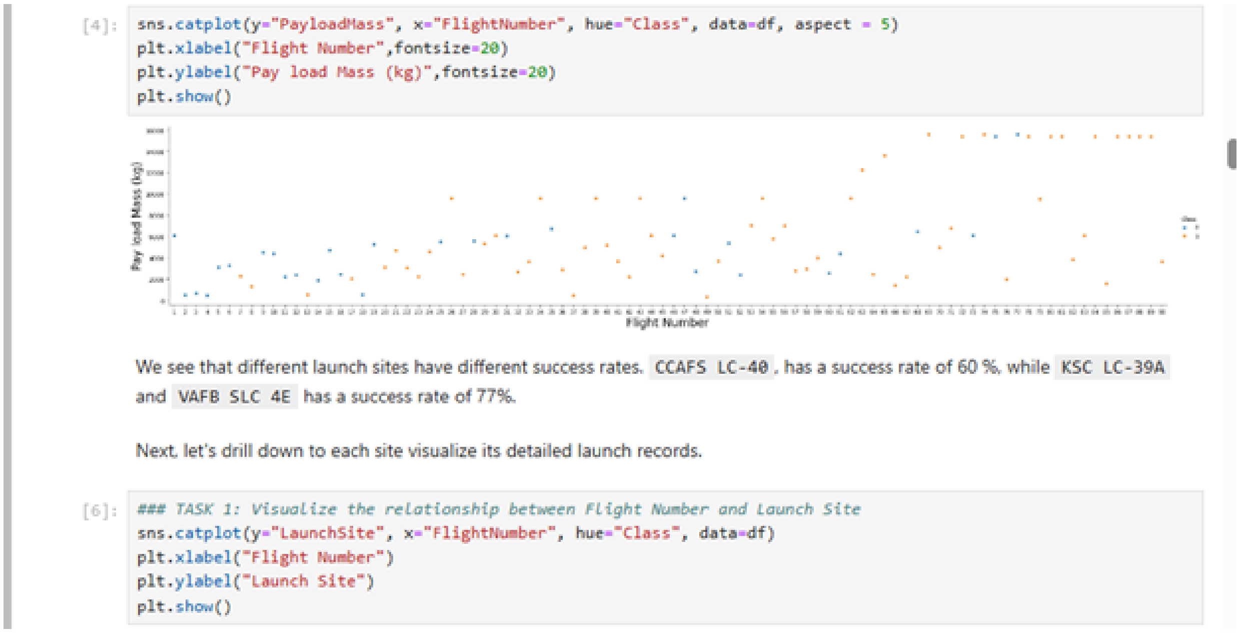


[3]:

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

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.

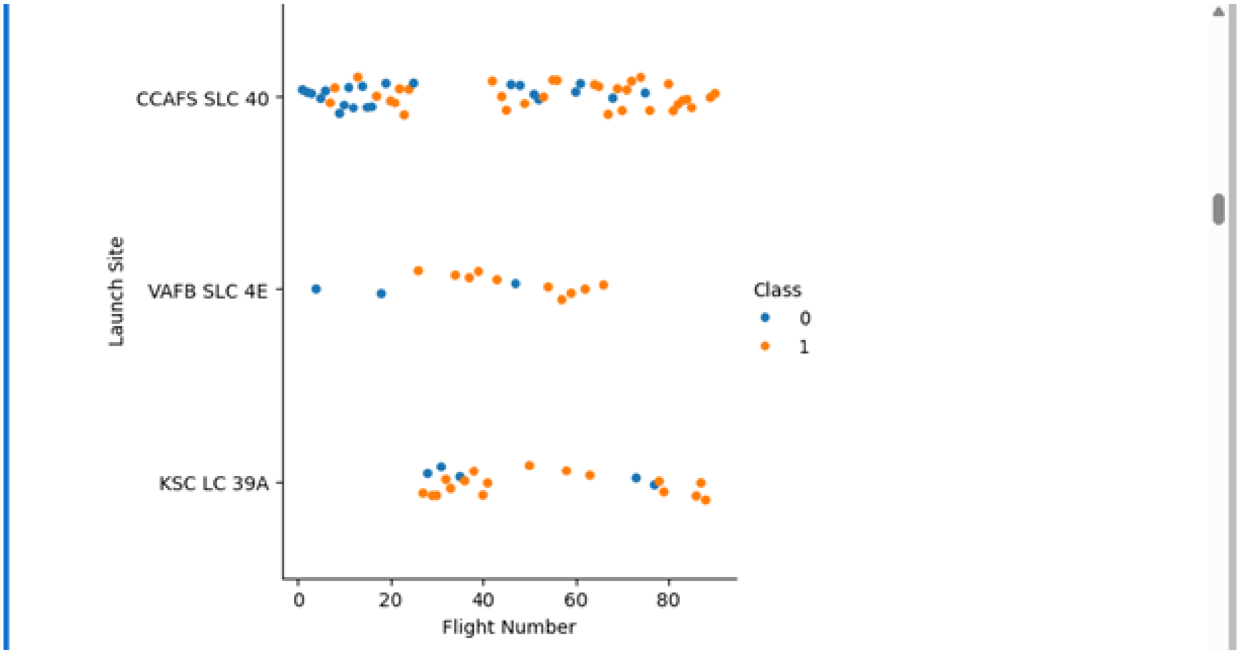


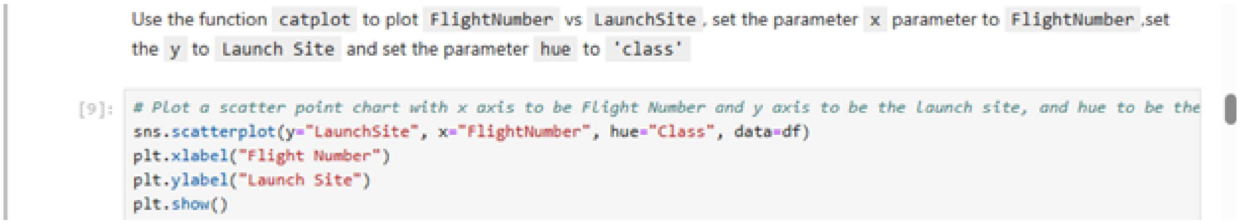
```
[4]: sns.catplot(y="PayloadMass", x="FlightNumber", hue="Class", data=df, aspect = 5)
     plt.xlabel("Flight Number",fontsize=20)
     plt.ylabel("Pay load Mass (kg)",fontsize=20)
     plt.show()
```

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.

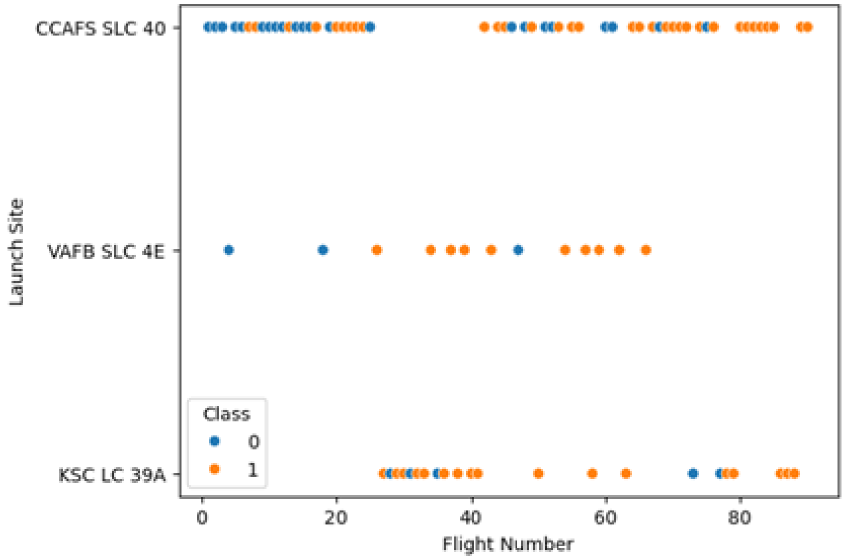
```
[6]: ### TASK 1: Visualize the relationship between Flight Number and Launch Site
     sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df)
     plt.xlabel("Flight Number")
     plt.ylabel("Launch Site")
     plt.show()
```

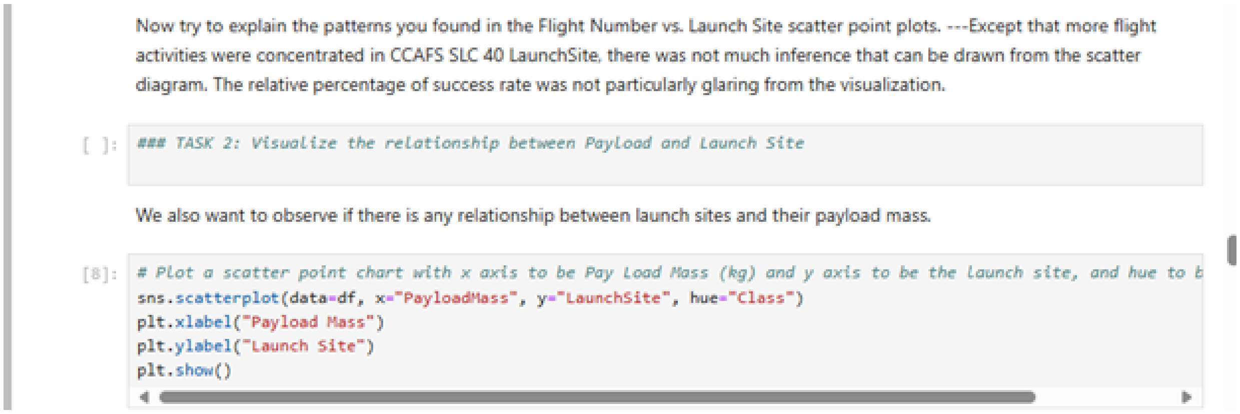




Use the function `catplot` to plot `FlightNumber` vs `LaunchSite`, set the parameter `x` parameter to `FlightNumber`,set the `y` to `Launch Site` and set the parameter `hue` to `'class'`

```
[9]: # 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
     sns.scatterplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df)
     plt.xlabel("Flight Number")
     plt.ylabel("Launch Site")
     plt.show()
```



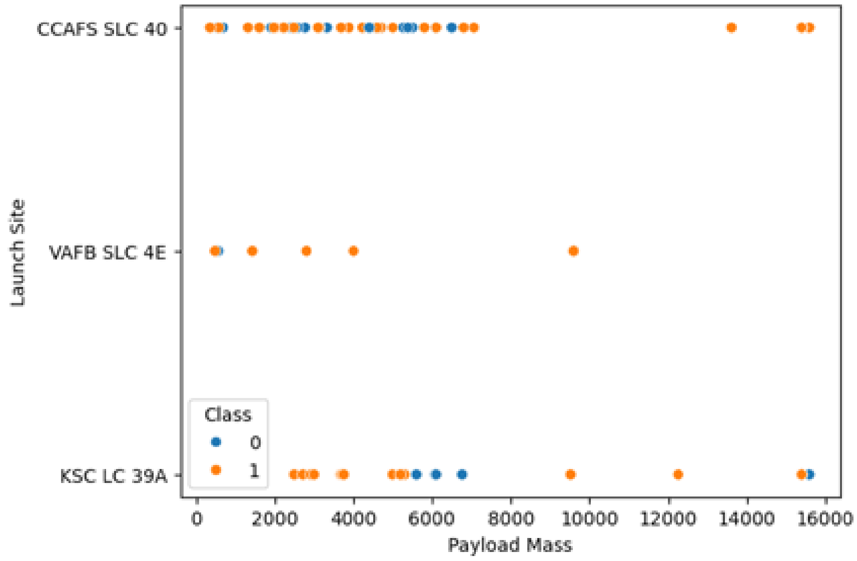


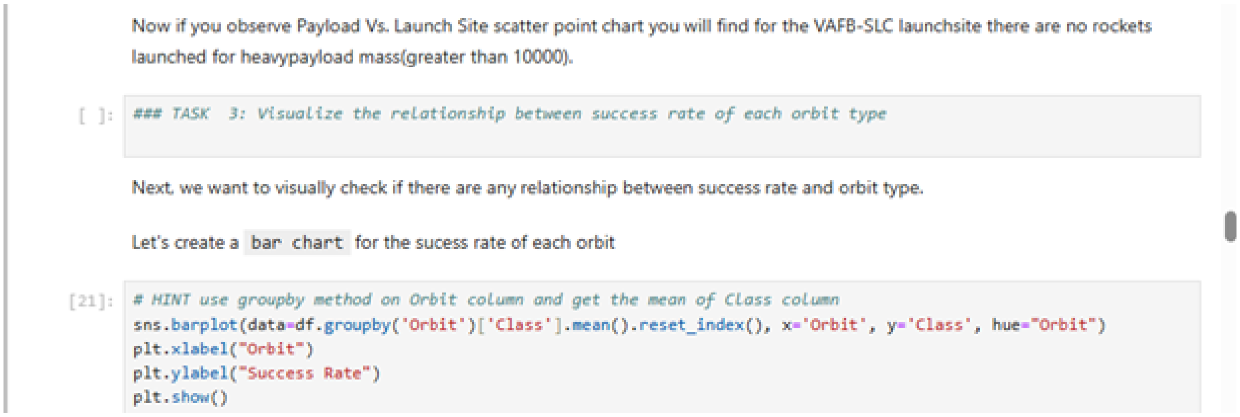
Now try to explain the patterns you found in the Flight Number vs. Launch Site scatter point plots. ---Except that more flight activities were concentrated in CCAFS SLC 40 LaunchSite, there was not much inference that can be drawn from the scatter diagram. The relative percentage of success rate was not particularly glaring from the visualization.

```
[ ]: ### 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.

```
[8]: # 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 b
     sns.scatterplot(data=df, x="PayloadMass", y="LaunchSite", hue="Class")
     plt.xlabel("Payload Mass")
     plt.ylabel("Launch Site")
     plt.show()
```





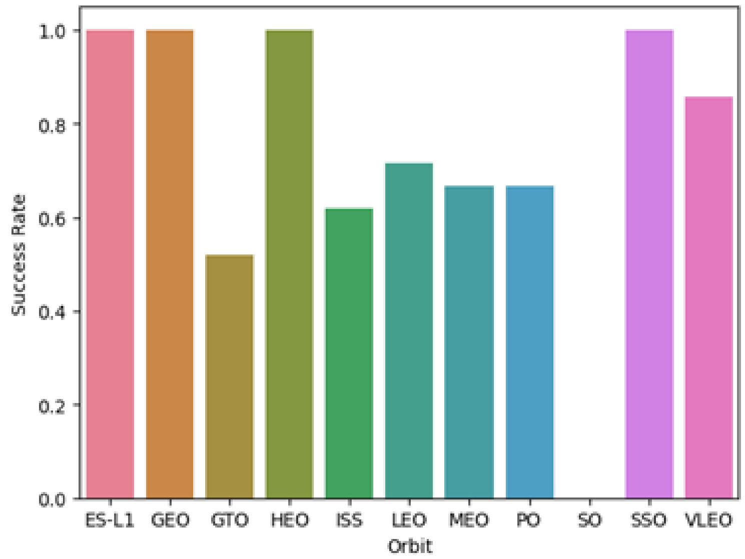
Now if you observe Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

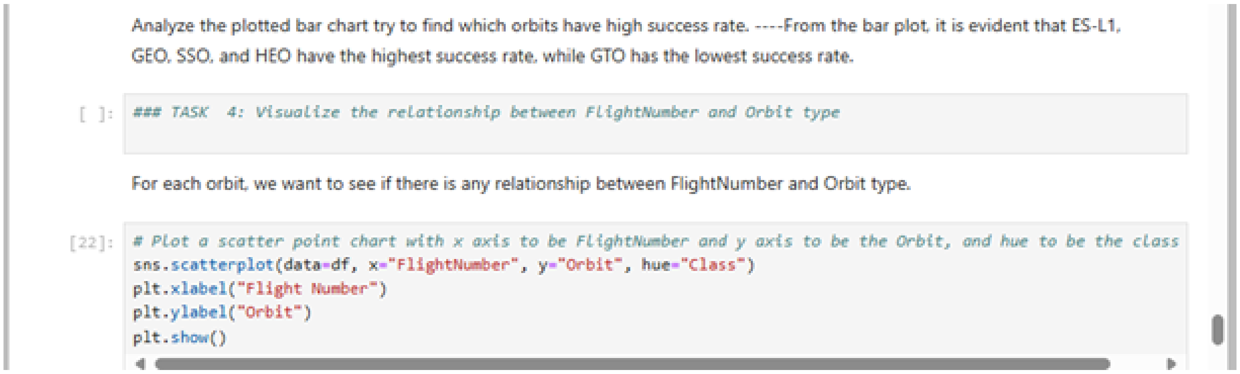
```
[ ]: ### TASK  3: Visualize the relationship between success rate of each orbit type
```

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

```
[21]: # HINT use groupby method on Orbit column and get the mean of Class column
      sns.barplot(data=df.groupby('Orbit')['Class'].mean().reset_index(), x='Orbit', y='Class', hue="Orbit")
      plt.xlabel("Orbit")
      plt.ylabel("Success Rate")
      plt.show()
```



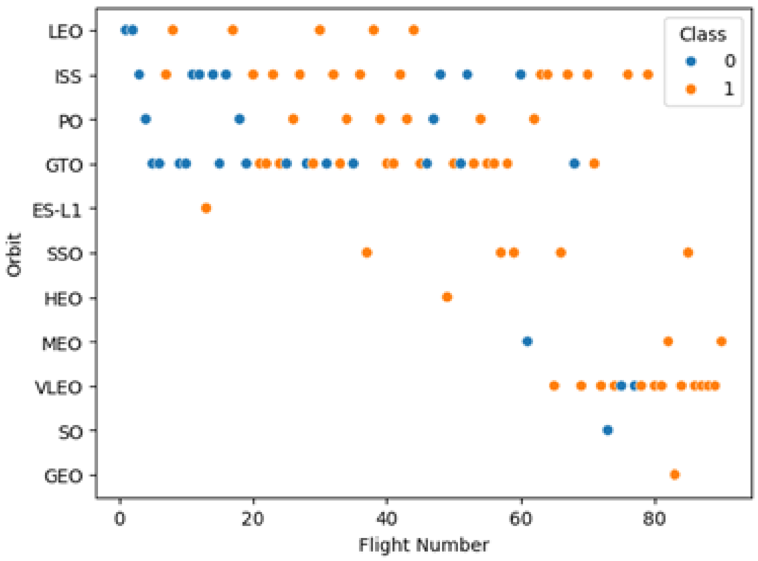


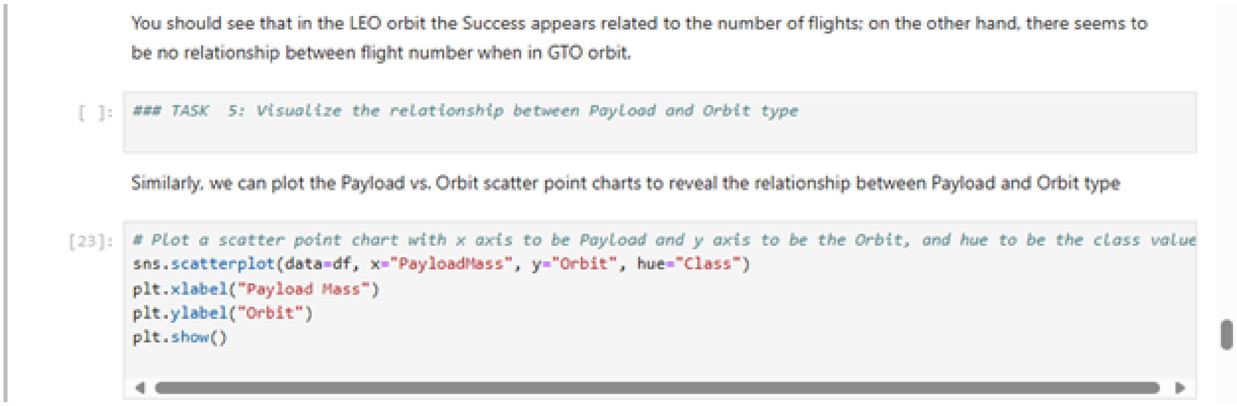
Analyze the plotted bar chart try to find which orbits have high success rate. ----From the bar plot, it is evident that ES-L1, GEO, SSO, and HEO have the highest success rate, while GTO has the lowest success rate.

```
[ ]: ### 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.

```
[22]: # Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class
      sns.scatterplot(data=df, x="FlightNumber", y="Orbit", hue="Class")
      plt.xlabel("Flight Number")
      plt.ylabel("Orbit")
      plt.show()
```



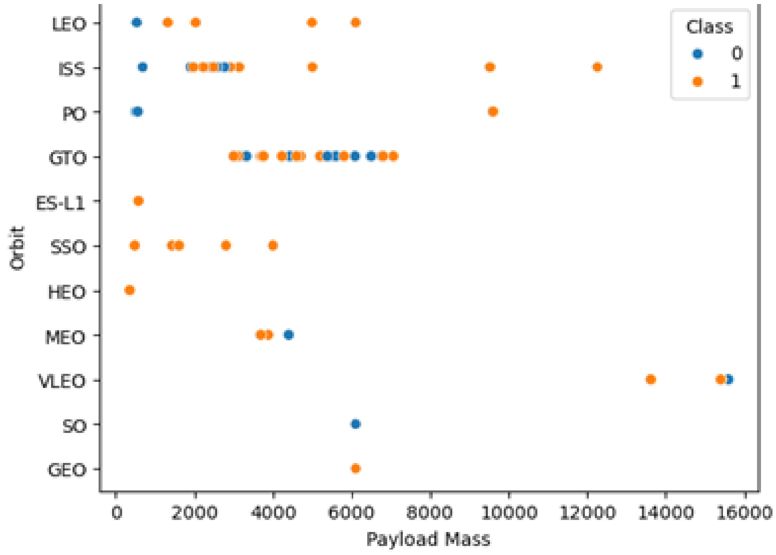


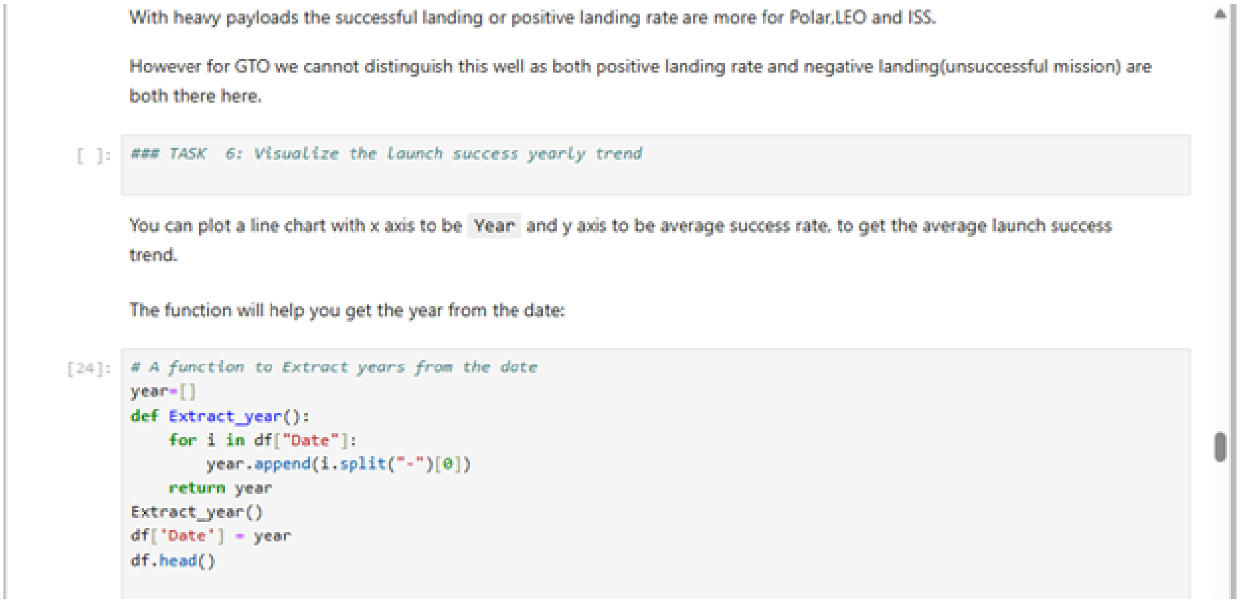
You should see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

```
[ ]: ### 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

```
[23]: # 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.scatterplot(data=df, x="PayloadMass", y="Orbit", hue="Class")
      plt.xlabel("Payload Mass")
      plt.ylabel("Orbit")
      plt.show()
```





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

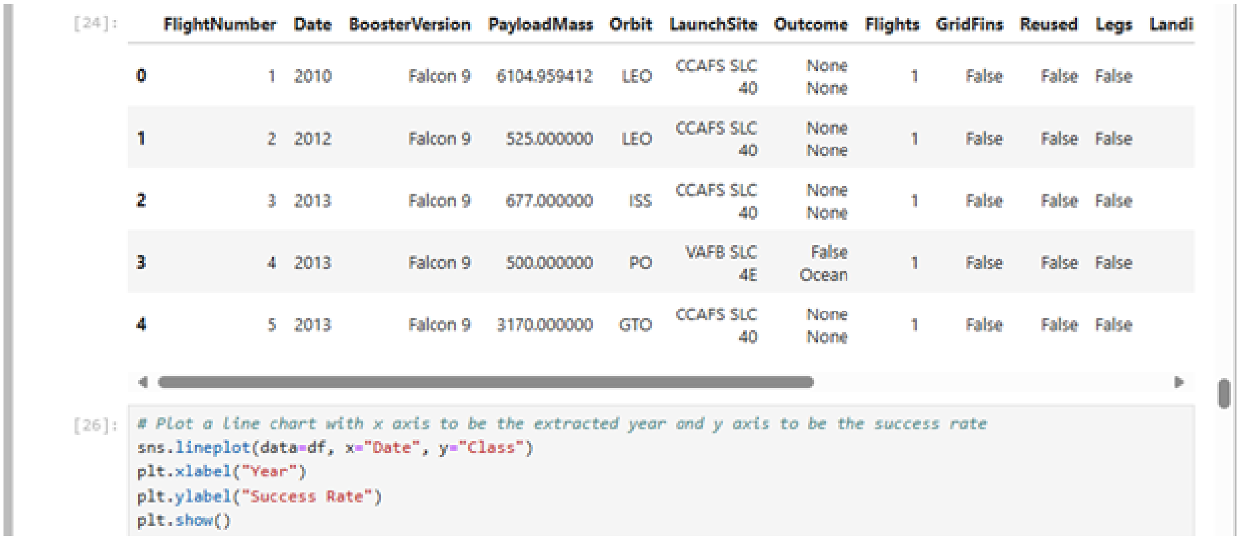
However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.

```
[ ]: ### 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:

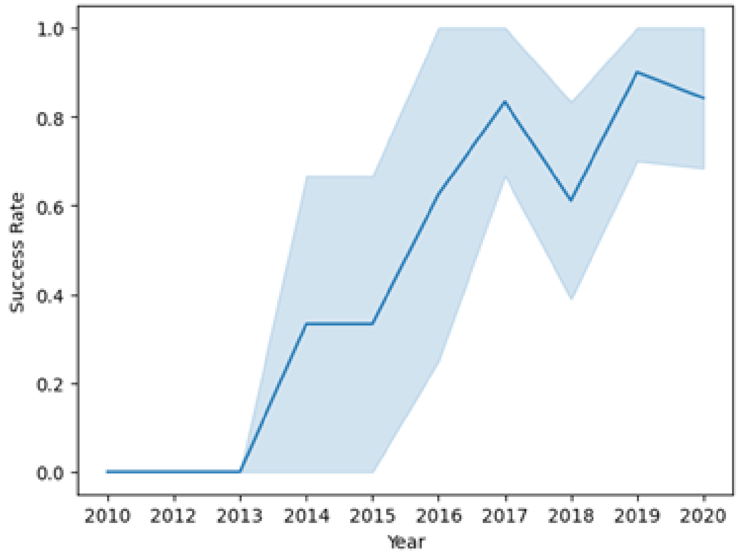
```
[24]: # A function to Extract years from the date
year=[]
def Extract_year():
    for i in df["Date"]:
        year.append(i.split("-")[0])
    return year
Extract_year()
df['Date'] = year
df.head()
```

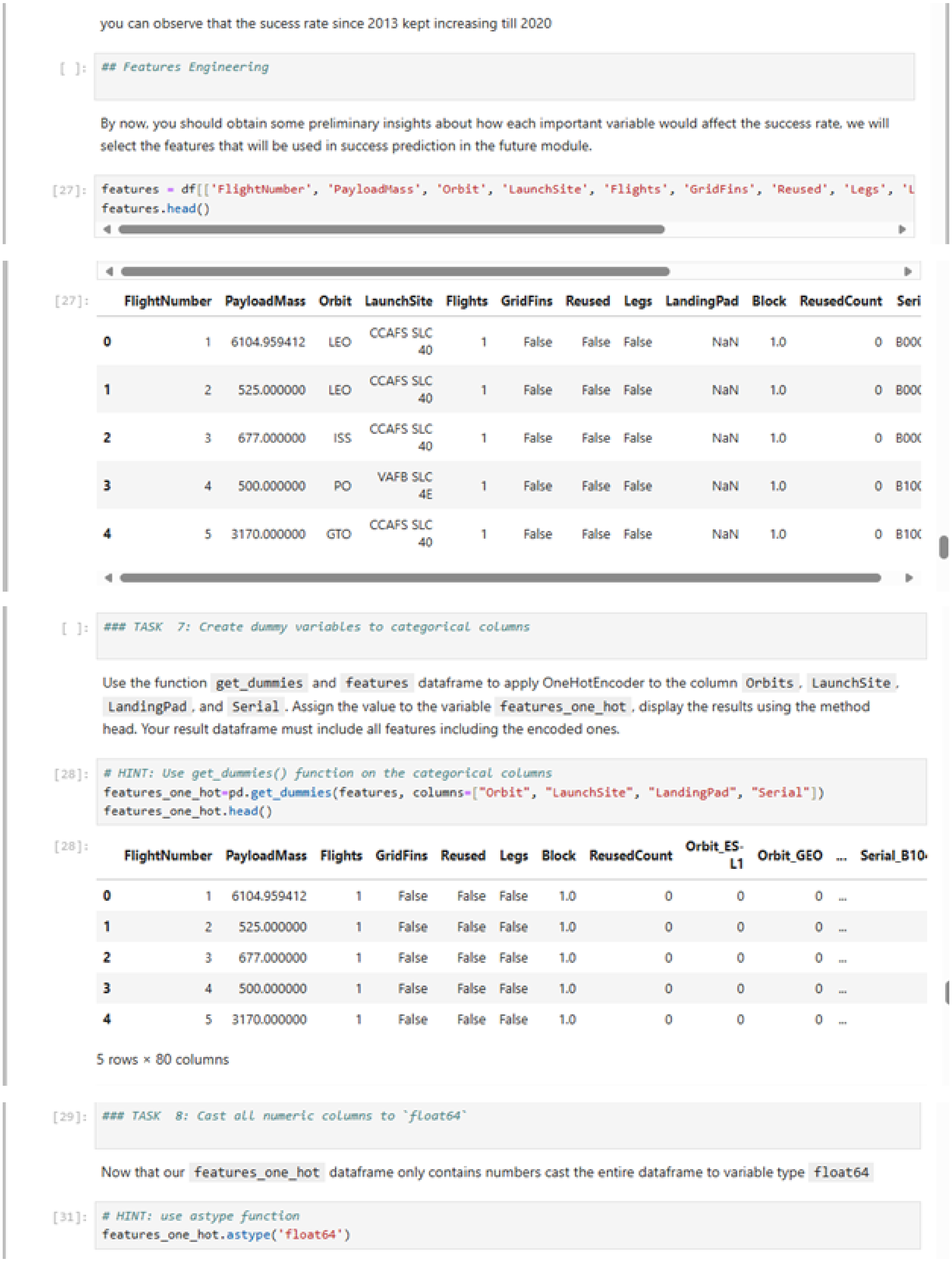


[24]:

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

```
[26]: # Plot a line chart with x axis to be the extracted year and y axis to be the success rate
sns.lineplot(data=df, x="Date", y="Class")
plt.xlabel("Year")
plt.ylabel("Success Rate")
plt.show()
```





you can observe that the sucess 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.

[27]:
```
features = df[['FlightNumber', 'PayloadMass', 'Orbit', 'LaunchSite', 'Flights', 'GridFins', 'Reused', 'Legs', 'L
features.head()
```

[27]:

	FlightNumber	PayloadMass	Orbit	LaunchSite	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Seri
0	1	6104.959412	LEO	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B00
1	2	525.000000	LEO	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B00
2	3	677.000000	ISS	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B00
3	4	500.000000	PO	VAFB SLC 4E	1	False	False	False	NaN	1.0	0	B10
4	5	3170.000000	GTO	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B10

[ ]:
```
### 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.

[28]:
```
# HINT: Use get_dummies() function on the categorical columns
features_one_hot=pd.get_dummies(features, columns=["Orbit", "LaunchSite", "LandingPad", "Serial"])
features_one_hot.head()
```

[28]:

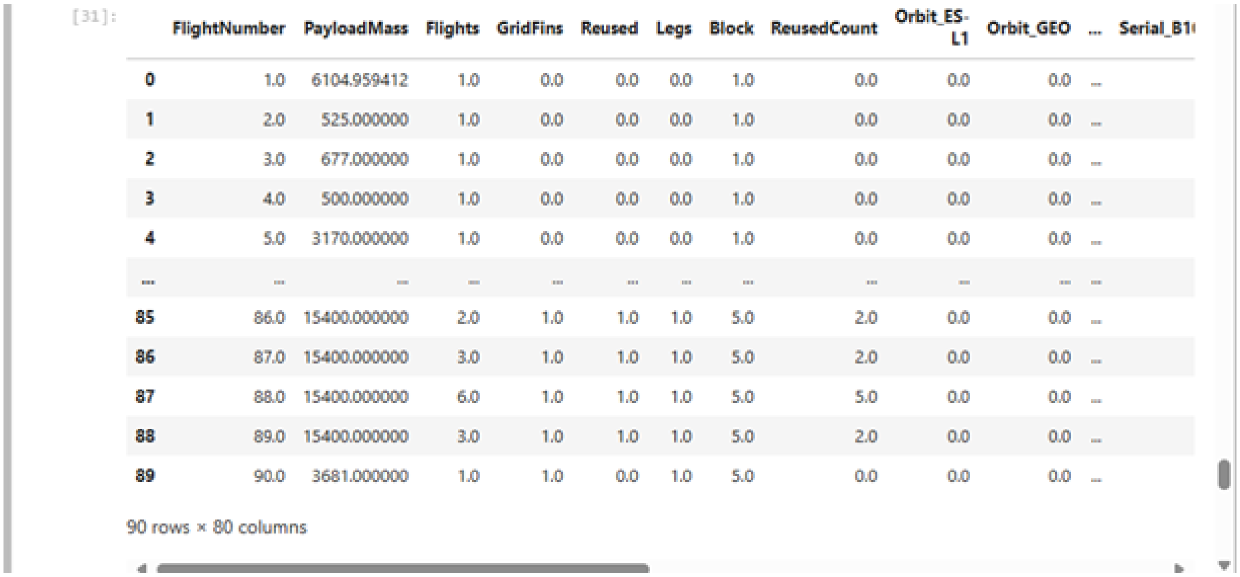
	FlightNumber	PayloadMass	Flights	GridFins	Reused	Legs	Block	ReusedCount	Orbit_ES-L1	Orbit_GEO	...	Serial_B10
0	1	6104.959412	1	False	False	False	1.0	0	0	0	...	
1	2	525.000000	1	False	False	False	1.0	0	0	0	...	
2	3	677.000000	1	False	False	False	1.0	0	0	0	...	
3	4	500.000000	1	False	False	False	1.0	0	0	0	...	
4	5	3170.000000	1	False	False	False	1.0	0	0	0	...	

5 rows × 80 columns

[29]:
```
### 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`

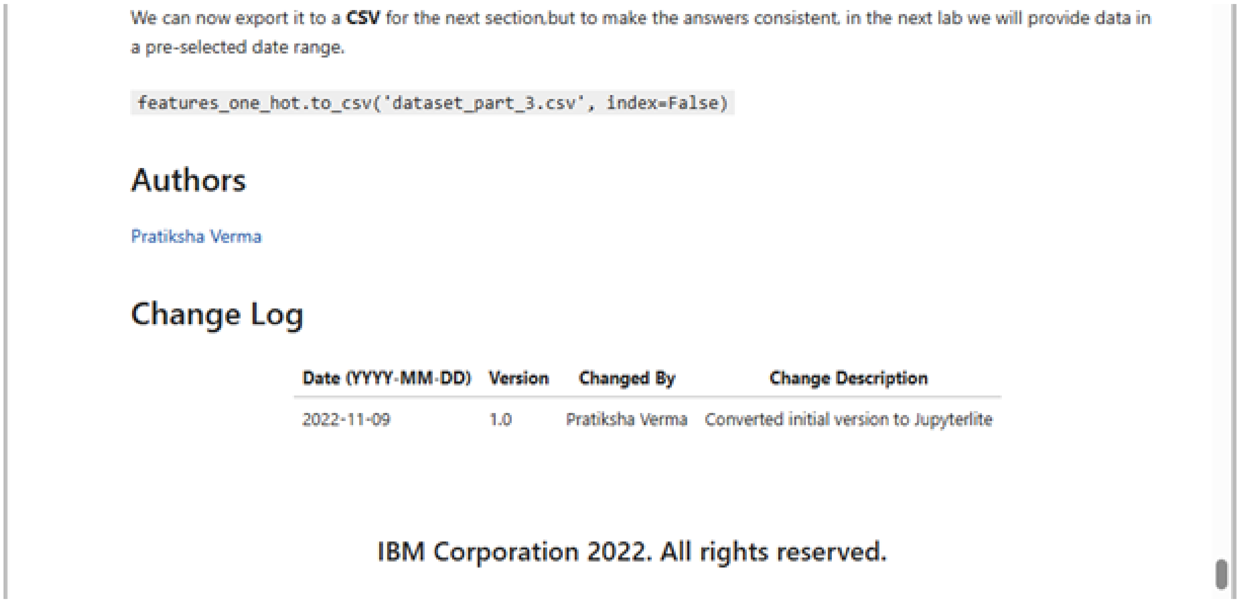
[31]:
```
# HINT: use astype function
features_one_hot.astype('float64')
```



[31]:

	FlightNumber	PayloadMass	Flights	GridFins	Reused	Legs	Block	ReusedCount	Orbit_ES-L1	Orbit_GEO	...	Serial_B1
0	1.0	6104.959412	1.0	0.0	0.0	0.0	1.0	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	...	
2	3.0	677.000000	1.0	0.0	0.0	0.0	1.0	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	...	
4	5.0	3170.000000	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
85	86.0	15400.000000	2.0	1.0	1.0	1.0	5.0	2.0	0.0	0.0	...	
86	87.0	15400.000000	3.0	1.0	1.0	1.0	5.0	2.0	0.0	0.0	...	
87	88.0	15400.000000	6.0	1.0	1.0	1.0	5.0	5.0	0.0	0.0	...	
88	89.0	15400.000000	3.0	1.0	1.0	1.0	5.0	2.0	0.0	0.0	...	
89	90.0	3681.000000	1.0	1.0	0.0	1.0	5.0	0.0	0.0	0.0	...	

90 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.

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

Authors

Pratiksha Verma

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2022-11-09	1.0	Pratiksha Verma	Converted initial version to Jupyterlite