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

In [1]:
```
!pip install pandas
!pip install numpy
!pip install seaborn
!pip install matplotlib
```

```
Collecting pandas
  Downloading pandas-2.2.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (89 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 89.9/89.9 kB 7.3 MB/s eta 0:00:00
Collecting numpy>=1.23.2 (from pandas)
  Downloading numpy-2.1.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (62 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.0/62.0 kB 7.7 MB/s eta 0:00:00
Requirement already satisfied: python-dateutil>=2.8.2 in /opt/conda/lib/python3.11/site-packages (from pand
as) (2.9.0)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/lib/python3.11/site-packages (from pandas) (2024.
1)
Collecting tzdata>=2022.7 (from pandas)
  Downloading tzdata-2024.2-py2.py3-none-any.whl.metadata (1.4 kB)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.11/site-packages (from python-dateutil>=
2.8.2->pandas) (1.16.0)
Downloading pandas-2.2.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (13.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.1/13.1 MB 123.4 MB/s eta 0:00:0000:010:01
Downloading numpy-2.1.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (16.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 16.3/16.3 MB 121.1 MB/s eta 0:00:0000:0100:01
Downloading tzdata-2024.2-py2.py3-none-any.whl (346 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 346.6/346.6 kB 41.7 MB/s eta 0:00:00
Installing collected packages: tzdata, numpy, pandas
Successfully installed numpy-2.1.3 pandas-2.2.3 tzdata-2024.2
Requirement already satisfied: numpy in /opt/conda/lib/python3.11/site-packages (2.1.3)
Collecting seaborn
  Downloading seaborn-0.13.2-py3-none-any.whl.metadata (5.4 kB)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in /opt/conda/lib/python3.11/site-packages (from seabor
n) (2.1.3)
Requirement already satisfied: pandas>=1.2 in /opt/conda/lib/python3.11/site-packages (from seaborn) (2.2.
3)
Collecting matplotlib!=3.6.1,>=3.4 (from seaborn)
  Downloading matplotlib-3.9.2-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (11 kB)
Collecting contourpy>=1.0.1 (from matplotlib!=3.6.1,>=3.4->seaborn)
  Downloading contourpy-1.3.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (5.4 kB)
Collecting cycler>=0.10 (from matplotlib!=3.6.1,>=3.4->seaborn)
  Downloading cycler-0.12.1-py3-none-any.whl.metadata (3.8 kB)
Collecting fonttools>=4.22.0 (from matplotlib!=3.6.1,>=3.4->seaborn)
  Downloading fonttools-4.55.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (164 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 164.5/164.5 kB 13.4 MB/s eta 0:00:00
Collecting kiwisolver>=1.3.1 (from matplotlib!=3.6.1,>=3.4->seaborn)
  Downloading kiwisolver-1.4.7-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (6.3 kB)
Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/python3.11/site-packages (from matplotlib!
=3.6.1,>=3.4->seaborn) (24.0)
```

```
Collecting pillow>=8 (from matplotlib!=3.6.1,>=3.4->seaborn)
  Downloading pillow-11.0.0-cp311-cp311-manylinux_2_28_x86_64.whl.metadata (9.1 kB)
Collecting pyparsing>=2.3.1 (from matplotlib!=3.6.1,>=3.4->seaborn)
  Downloading pyparsing-3.2.0-py3-none-any.whl.metadata (5.0 kB)
Requirement already satisfied: python-dateutil>=2.7 in /opt/conda/lib/python3.11/site-packages (from matplo
tlib!=3.6.1,>=3.4->seaborn) (2.9.0)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/lib/python3.11/site-packages (from pandas>=1.2->s
eaborn) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in /opt/conda/lib/python3.11/site-packages (from pandas>=1.2-
>seaborn) (2024.2)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.11/site-packages (from python-dateutil>=
2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.16.0)
Downloading seaborn-0.13.2-py3-none-any.whl (294 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 294.9/294.9 kB 30.3 MB/s eta 0:00:00
Downloading matplotlib-3.9.2-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (8.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.3/8.3 MB 133.2 MB/s eta 0:00:00a 0:00:01
Downloading contourpy-1.3.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (326 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 326.2/326.2 kB 34.3 MB/s eta 0:00:00
Downloading cycler-0.12.1-py3-none-any.whl (8.3 kB)
Downloading fonttools-4.55.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (4.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.9/4.9 MB 115.9 MB/s eta 0:00:0000:01
Downloading kiwisolver-1.4.7-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.4 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.4/1.4 MB 69.1 MB/s eta 0:00:00
Downloading pillow-11.0.0-cp311-cp311-manylinux_2_28_x86_64.whl (4.4 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.4/4.4 MB 121.7 MB/s eta 0:00:0000:01
Downloading pyparsing-3.2.0-py3-none-any.whl (106 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 106.9/106.9 kB 15.2 MB/s eta 0:00:00
Installing collected packages: pyparsing, pillow, kiwisolver, fonttools, cycler, contourpy, matplotlib, sea
born
Successfully installed contourpy-1.3.1 cycler-0.12.1 fonttools-4.55.0 kiwisolver-1.4.7 matplotlib-3.9.2 pil
low-11.0.0 pyparsing-3.2.0 seaborn-0.13.2
Requirement already satisfied: matplotlib in /opt/conda/lib/python3.11/site-packages (3.9.2)
Requirement already satisfied: contourpy>=1.0.1 in /opt/conda/lib/python3.11/site-packages (from matplotli
b) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.11/site-packages (from matplotlib)
(0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /opt/conda/lib/python3.11/site-packages (from matplotli
b) (4.55.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /opt/conda/lib/python3.11/site-packages (from matplotli
b) (1.4.7)
Requirement already satisfied: numpy>=1.23 in /opt/conda/lib/python3.11/site-packages (from matplotlib) (2.
1.3)
```

```
Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/python3.11/site-packages (from matplotlib)
(24.0)
Requirement already satisfied: pillow>=8 in /opt/conda/lib/python3.11/site-packages (from matplotlib) (11.
0.0)
Requirement already satisfied: pyparsing>=2.3.1 in /opt/conda/lib/python3.11/site-packages (from matplotli
b) (3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in /opt/conda/lib/python3.11/site-packages (from matplo
tlib) (2.9.0)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.11/site-packages (from python-dateutil>=
2.7->matplotlib) (1.16.0)
```

In [ ]:

Import Libraries and Define Auxiliary Functions

We will import the following libraries the lab

In [2]:

```
# andas is a software library written for the Python programming language for data manipulation and analysi
import pandas as pd
#NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays
import numpy as np
# Matplotlib is a plotting library for python and pyplot gives us a MatLab like plotting framework. We wil
import matplotlib.pyplot as plt
#Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for
import seaborn as sns
print("Done")
```

```
Done
```

In [ ]:

Exploratory Data Analysis

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

In [3]:

```
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetw

# If you were unable to complete the previous lab correctly you can uncomment and load this csv
```

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

df.head(5)
```

Out[3]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	La
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	

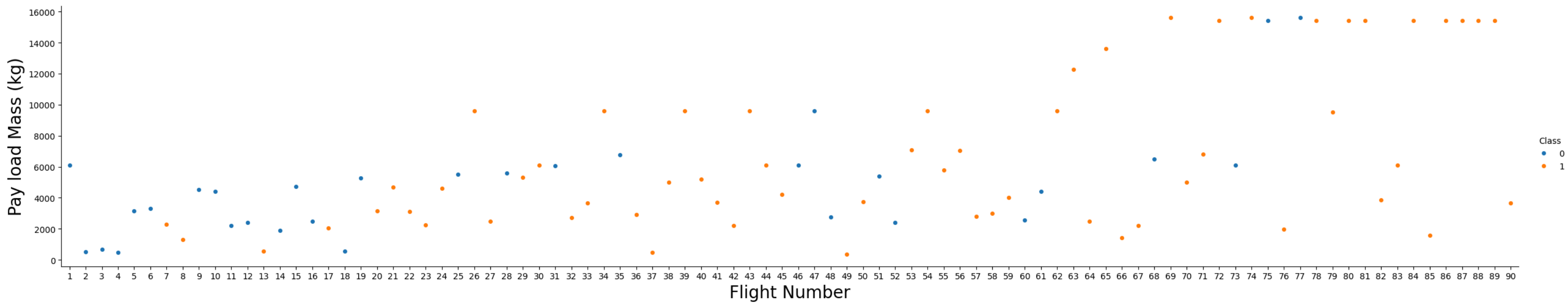
In [ ]:

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.

In [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()
```



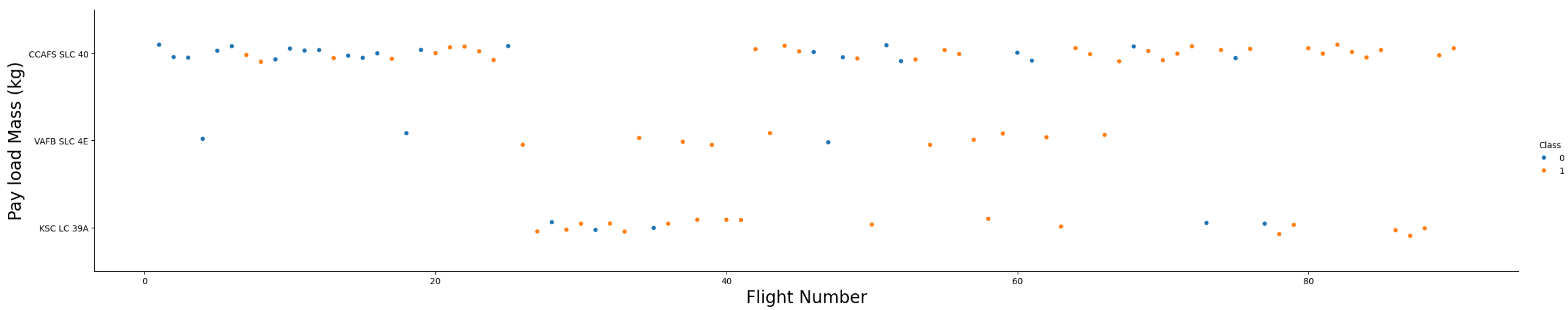
In [ ]:

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 [5]:
```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to l
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Pay load Mass (kg)",fontsize=20)
plt.show()
```



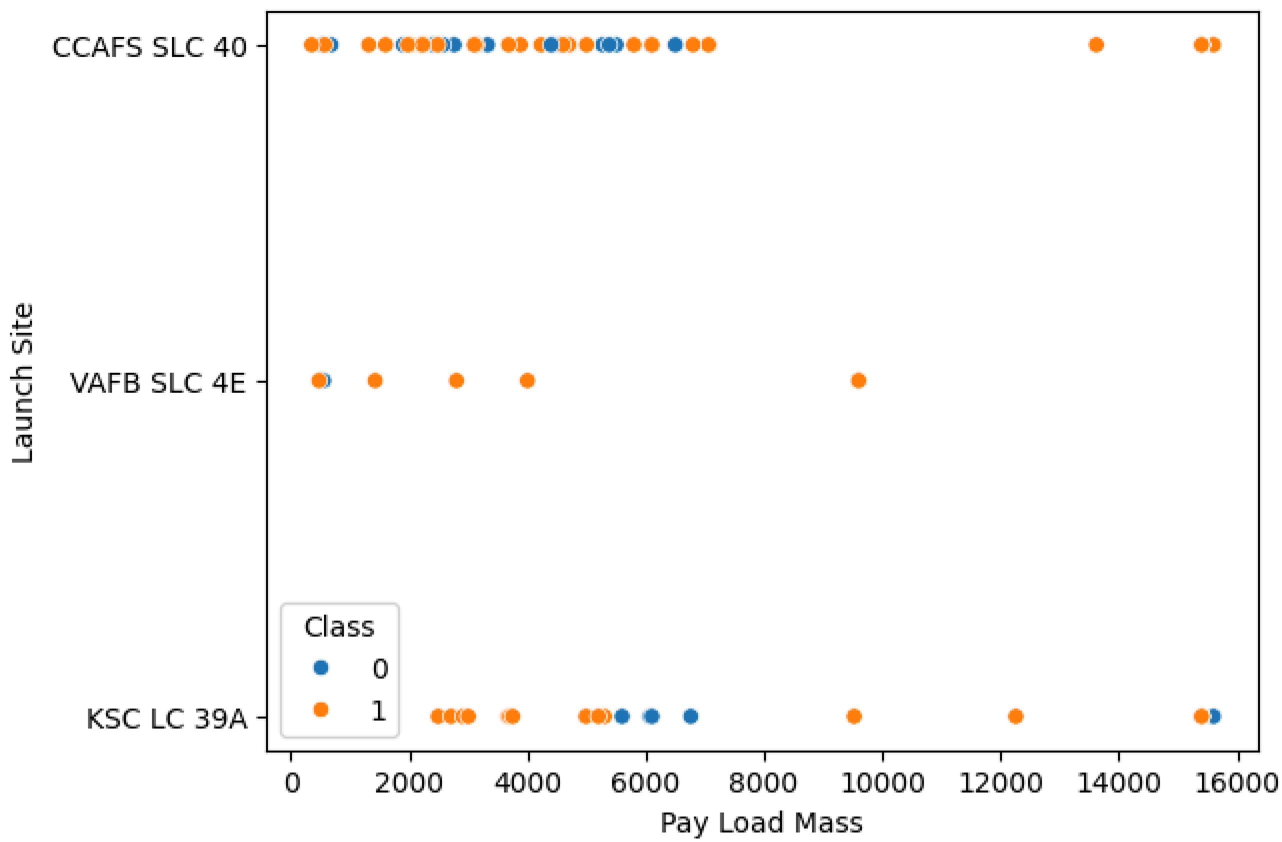
In [ ]:

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.

In [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
sns.scatterplot(x="PayloadMass",y="LaunchSite",data=df, hue="Class")
plt.xlabel("Pay Load Mass")
plt.ylabel("Launch Site")
plt.show()
```



In [ ]:

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

In [7]:
```
# HINT use groupby method on Orbit column and get the mean of Class column
grouped_orbits = df.groupby(by=['Orbit'])['Class'].mean().sort_values(ascending=False).reset_index()
fig, ax=plt.subplots(figsize=(12,6))
ax = sns.barplot(x = 'Orbit', y = 'Class',data=grouped_orbits)
ax.set_title('Plot of success rate by class of each Orbits', fontdict={'size':12})
ax.set_ylabel('Class', fontsize = 10)
ax.set_xlabel('Orbits', fontsize = 10)
plt.show()
```

In [ ]:

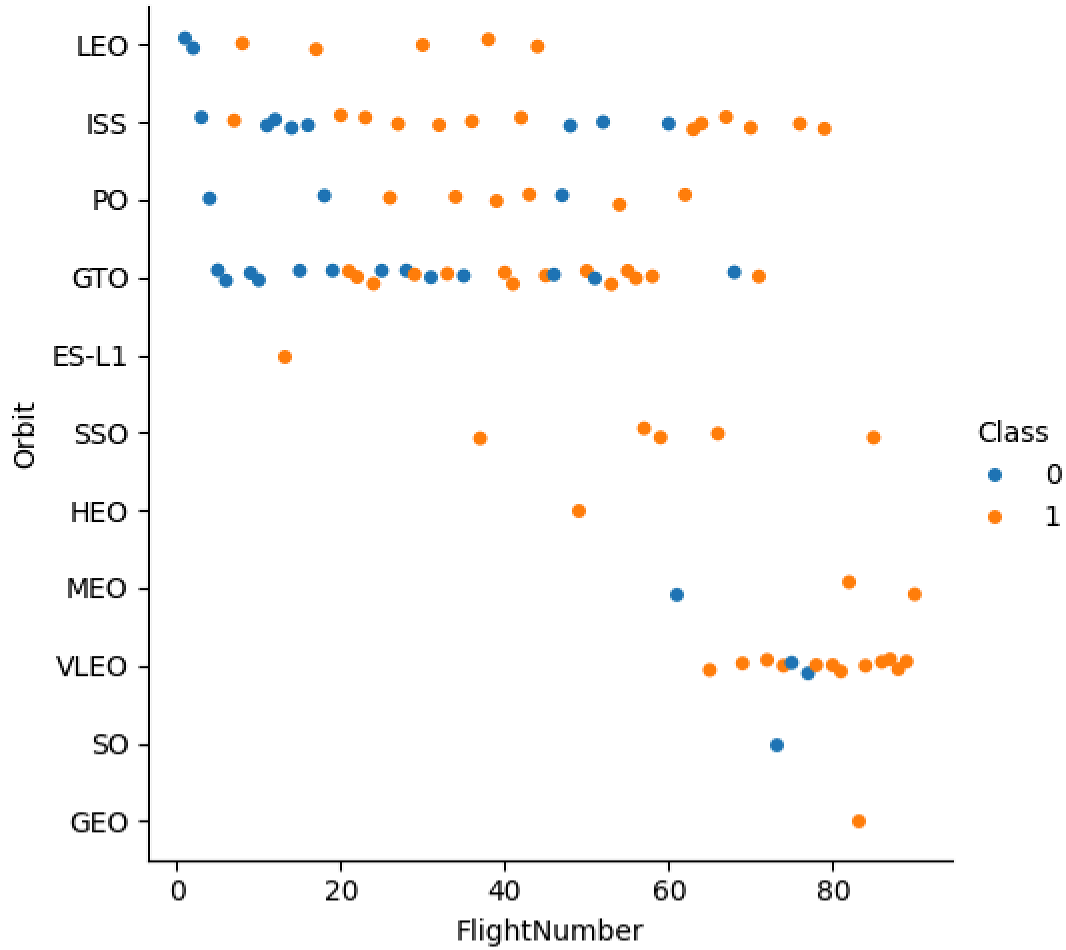
Analyze the ploted bar chart try to find which orbits have high sucess rate.

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.

In [8]:

```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the c
sns.catplot(x="FlightNumber",y="Orbit",data=df,hue="Class")
plt.show()
```



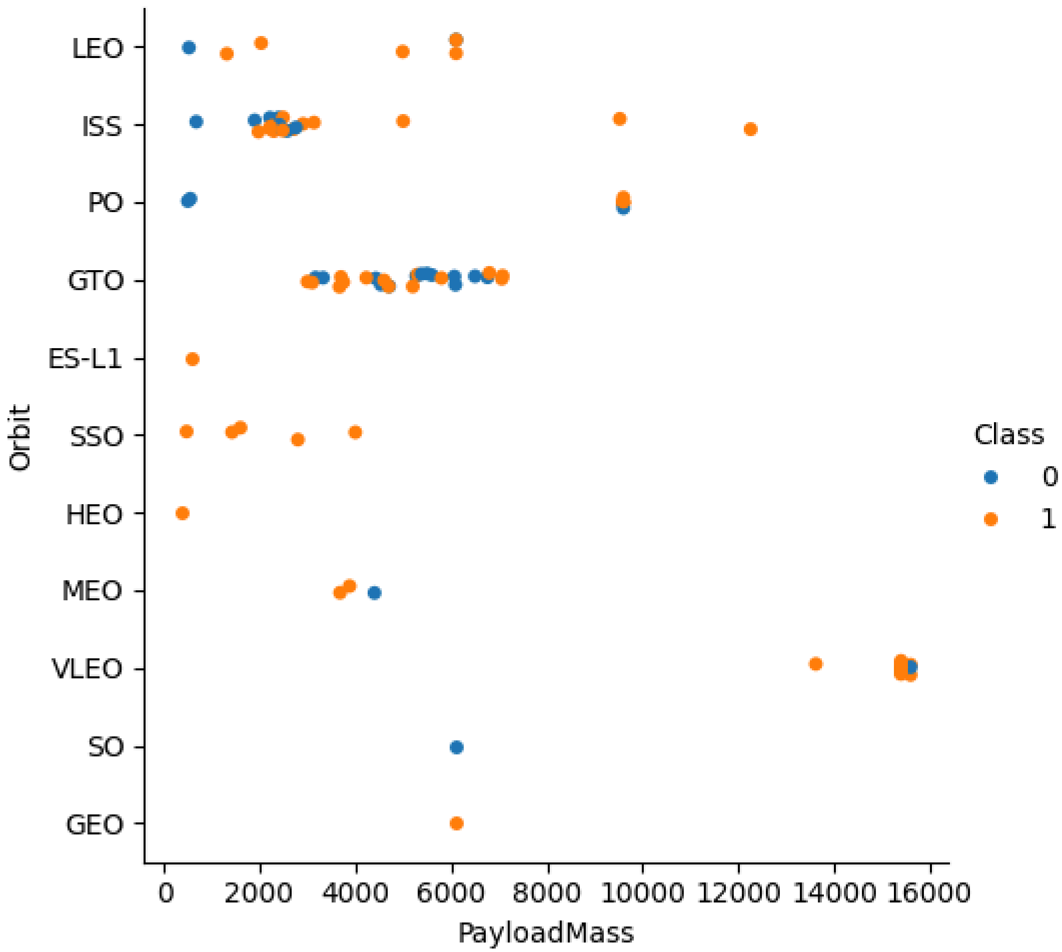
In [ ]:

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

In [9]:
```
# Plot a scatter point chart with x axis to be Payload and y axis to be the Orbit, and hue to be the class
sns.catplot(x="PayloadMass",y="Orbit",data=df,hue="Class")
plt.show()
```



In [ ]:

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:

In [10]:
```
# 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[10]:

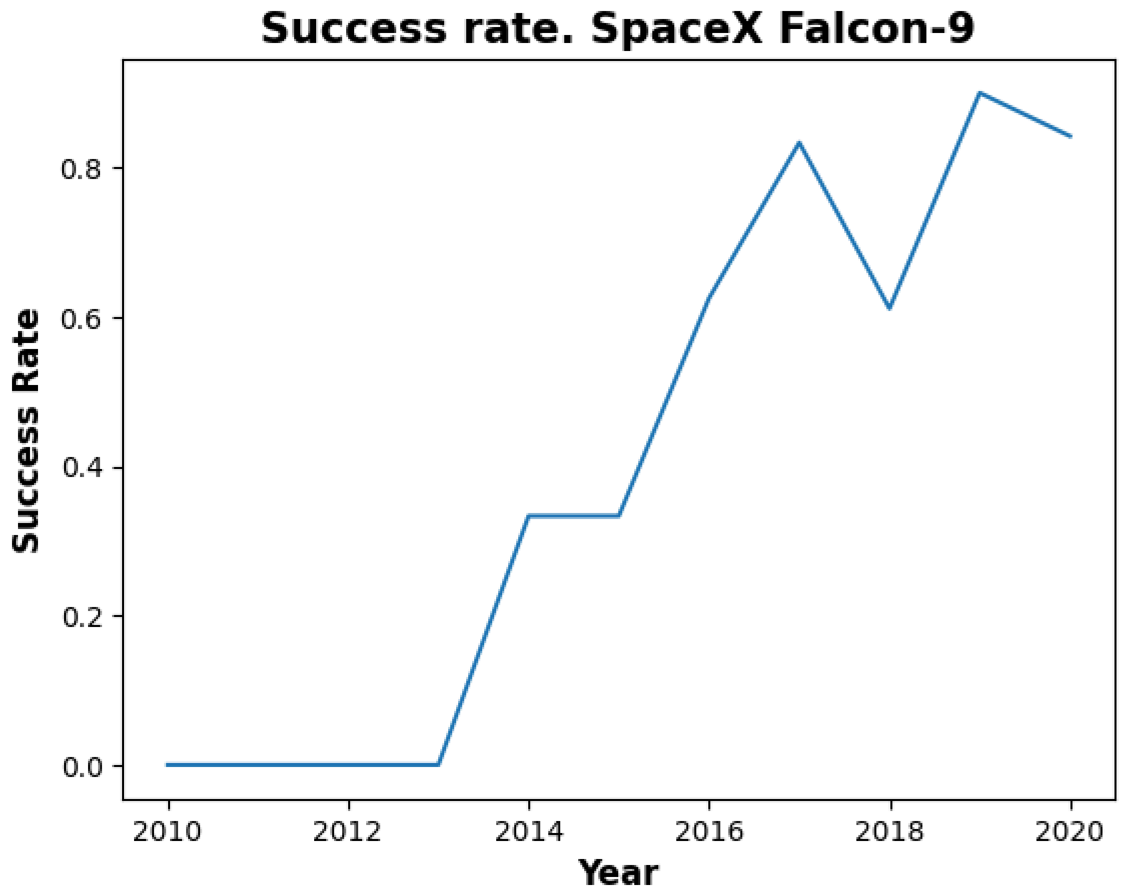
	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	Lan
0	1	2010	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	
1	2	2012	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	
2	3	2013	Falcon 9	677.000000	ISS	CCAFS SLC 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	

In [ ]:

In [11]:
```
# 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
```

In [12]:
```
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
df['Year'] = pd.DataFrame(Extract_year(df['Date'])).astype('int')
sns.lineplot(x = df['Year'].unique() , y = df.groupby(['Year'])['Class'].mean())
#plt.xlabel("Years",fontsize=20)
#plt.ylabel("Success Rate",fontsize=20)
plt.xlabel("Year",fontsize=12, fontweight='bold')
plt.ylabel("Success Rate",fontsize=12, fontweight='bold')
#plt.plot(X,total_launches)
plt.title("Success rate. SpaceX Falcon-9", fontsize=15, fontweight='bold')
plt.show()
```



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.

In [13]:
```
features = df[['FlightNumber', 'PayloadMass', 'Orbit', 'LaunchSite', 'Flights', 'GridFins', 'Reused', 'Leg
features.head()
```

Out[13]:

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

In [ ]:

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 [14]:

```
features_one_hot=pd.get_dummies(features, columns=['Orbit','LaunchSite', 'LandingPad', 'Serial'])
features_one_hot
```

Out[14]:

	FlightNumber	PayloadMass	Flights	GridFins	Reused	Legs	Block	ReusedCount	Orbit_ES-L1	Orbit_GEO	...	Seria
0	1	6104.959412	1	False	False	False	1.0	0	False	False	...	
1	2	525.000000	1	False	False	False	1.0	0	False	False	...	
2	3	677.000000	1	False	False	False	1.0	0	False	False	...	
3	4	500.000000	1	False	False	False	1.0	0	False	False	...	
4	5	3170.000000	1	False	False	False	1.0	0	False	False	...	
...	...	...	...	...	...	...	...	...	...	...	...	
85	86	15400.000000	2	True	True	True	5.0	2	False	False	...	
86	87	15400.000000	3	True	True	True	5.0	2	False	False	...	
87	88	15400.000000	6	True	True	True	5.0	5	False	False	...	
88	89	15400.000000	3	True	True	True	5.0	2	False	False	...	
89	90	3681.000000	1	True	False	True	5.0	0	False	False	...	

90 rows × 80 columns

In [ ]:

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 [15]:

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

Out[15]: (90, 80)

In [ ]:

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)`

In [ ]:

```

```

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