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 languag
import pandas as pd
#NumPy is a library for the Python programming language, adding support f
import numpy as np
# Matplotlib is a plotting library for python and pyplot gives us a MatLa
import matplotlib.pyplot as plt
#Seaborn is a Python data visualization library based on matplotlib. It p
import seaborn as sns
```

```
<ipython-input-2-cde6ab162d36>:2: DeprecationWarning:
Pyarrow will become a required dependency of pandas in the next major rele
ase of pandas (pandas 3.0),
(to allow more performant data types, such as the Arrow string type, and b
etter 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
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)
```

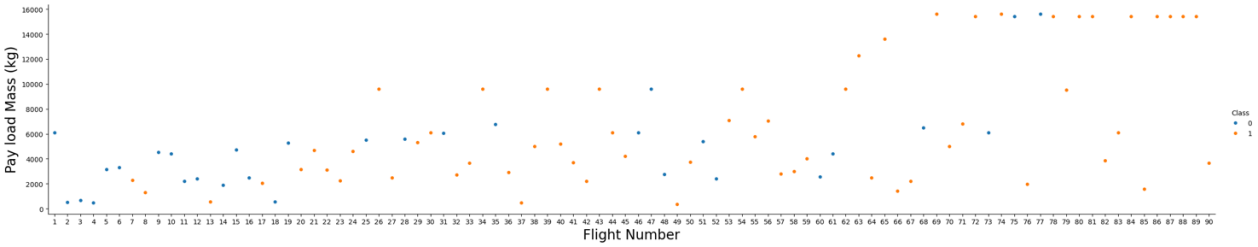
Out[3]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome
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

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, aspe
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Pay load Mass (kg)",fontsize=20)
plt.show()
```



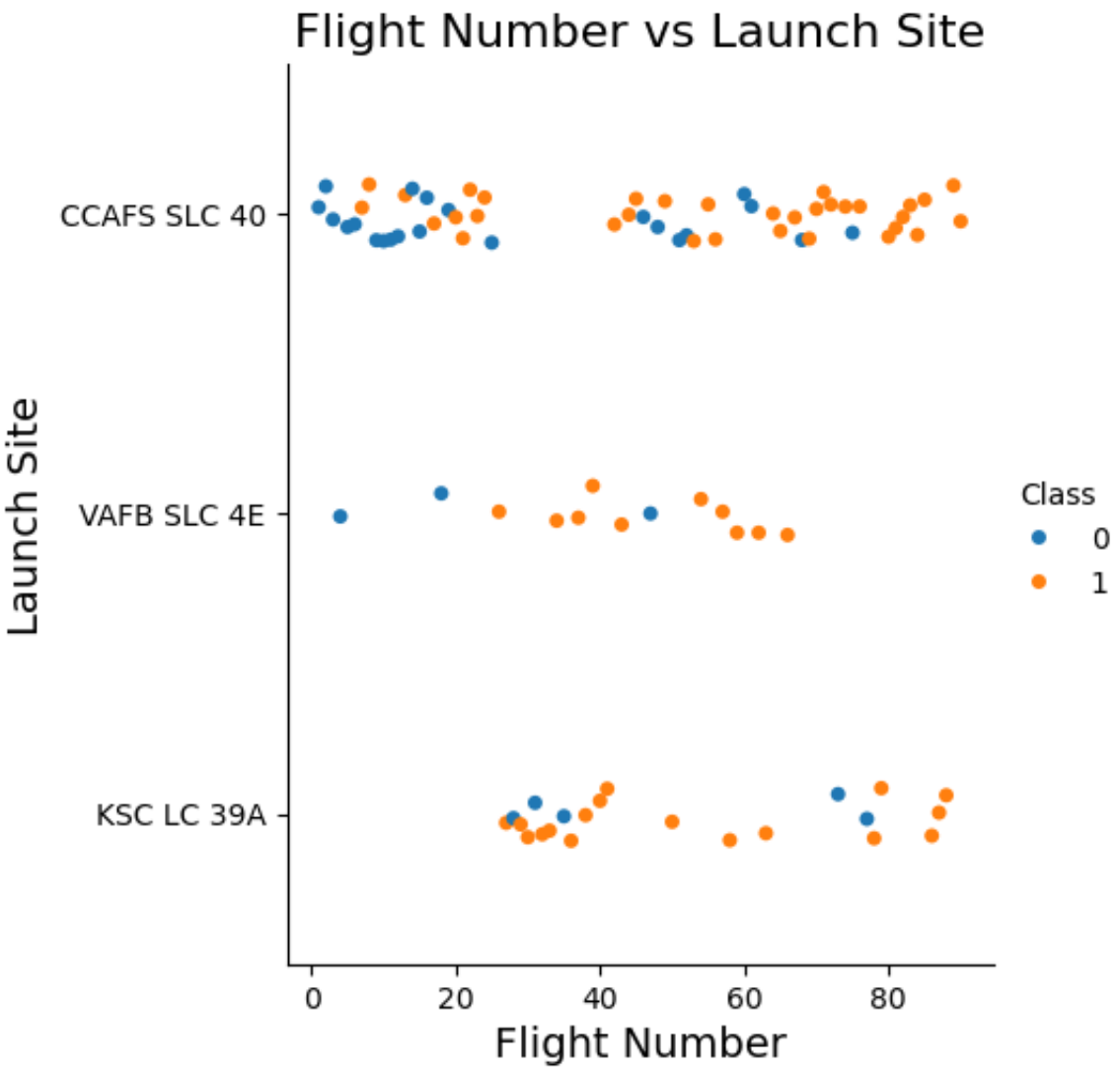
Next, let's drill down to each site visualize its detailed launch records.

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 [5]:
```
# Plot a scatter point chart with x axis to be Flight Number and y axis t

# Scatter plot showing the relationship between FlightNumber and LaunchSi
sns.catplot(x="FlightNumber", y="LaunchSite", hue="Class", data=df)
plt.xlabel('Flight Number', fontsize=14)
plt.ylabel('Launch Site', fontsize=14)
plt.title('Flight Number vs Launch Site', fontsize=16)
plt.show()
```

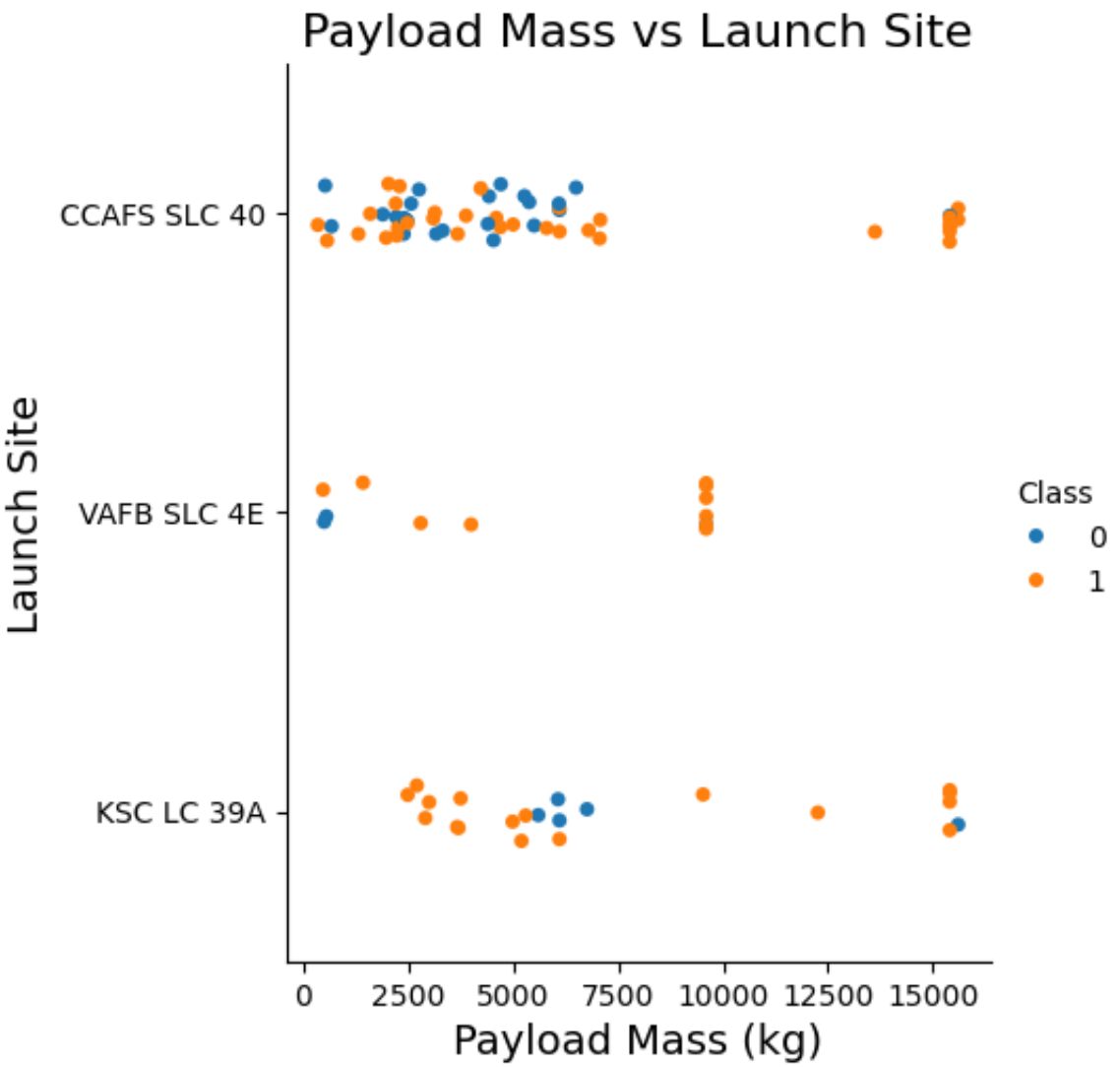


Now try to explain the patterns you found in the Flight Number vs. Launch Site scatter point plots.

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 [6]:
```
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y a
# Scatter plot showing the relationship between PayloadMass and LaunchSit
sns.catplot(x="PayloadMass", y="LaunchSite", hue="Class", data=df)
plt.xlabel('Payload Mass (kg)', fontsize=14)
plt.ylabel('Launch Site', fontsize=14)
plt.title('Payload Mass vs Launch Site', fontsize=16)
plt.show()
```



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

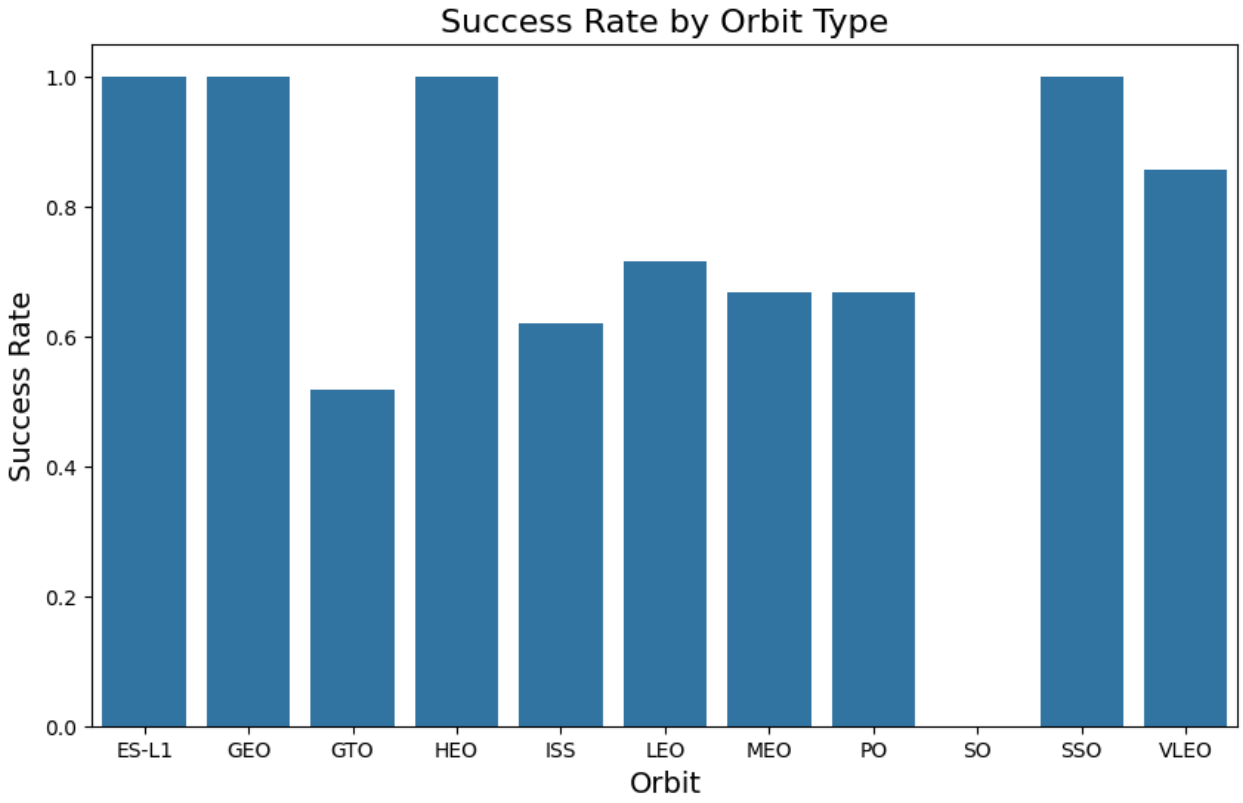
Next, we want to visually check if there are any relationship between success rate and orbit type.

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

In [7]:
```
# HINT use groupby method on Orbit column and get the mean of Class colum
# Group by Orbit and calculate the mean success rate (Class)
orbit_success = df.groupby('Orbit')['Class'].mean().reset_index()

# Bar chart showing the success rate for each orbit type
plt.figure(figsize=(10,6))
sns.barplot(x='Orbit', y='Class', data=orbit_success)
plt.xlabel('Orbit', fontsize=14)
plt.ylabel('Success Rate', fontsize=14)
plt.title('Success Rate by Orbit Type', fontsize=16)
```

```
plt.show()
```

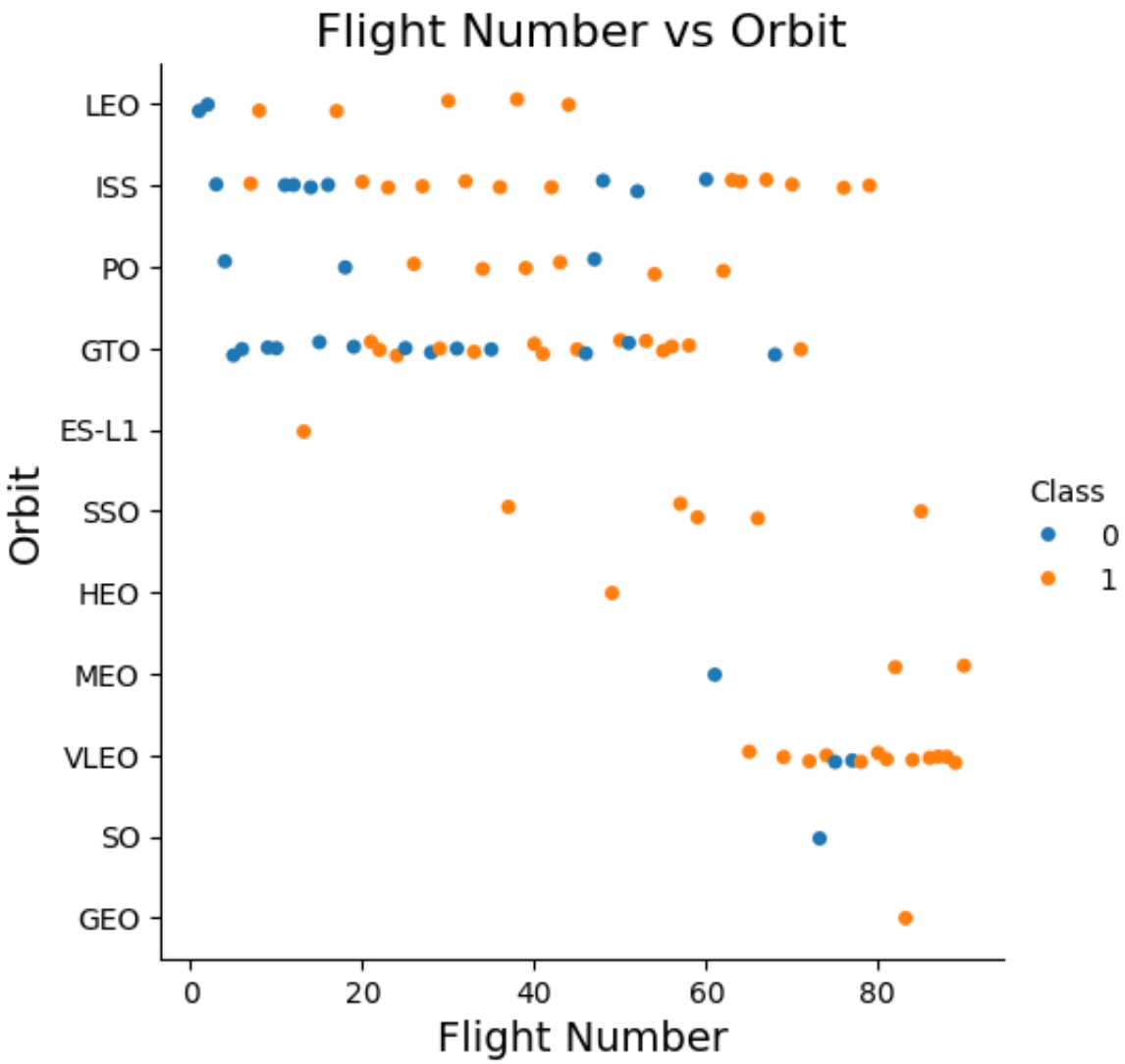


Analyze the plotted bar chart to identify which orbits have the highest success rates.

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 [8]:
```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to
# Scatter plot showing the relationship between FlightNumber and Orbit
sns.catplot(x="FlightNumber", y="Orbit", hue="Class", data=df)
plt.xlabel('Flight Number', fontsize=14)
plt.ylabel('Orbit', fontsize=14)
plt.title('Flight Number vs Orbit', fontsize=16)
plt.show()
```

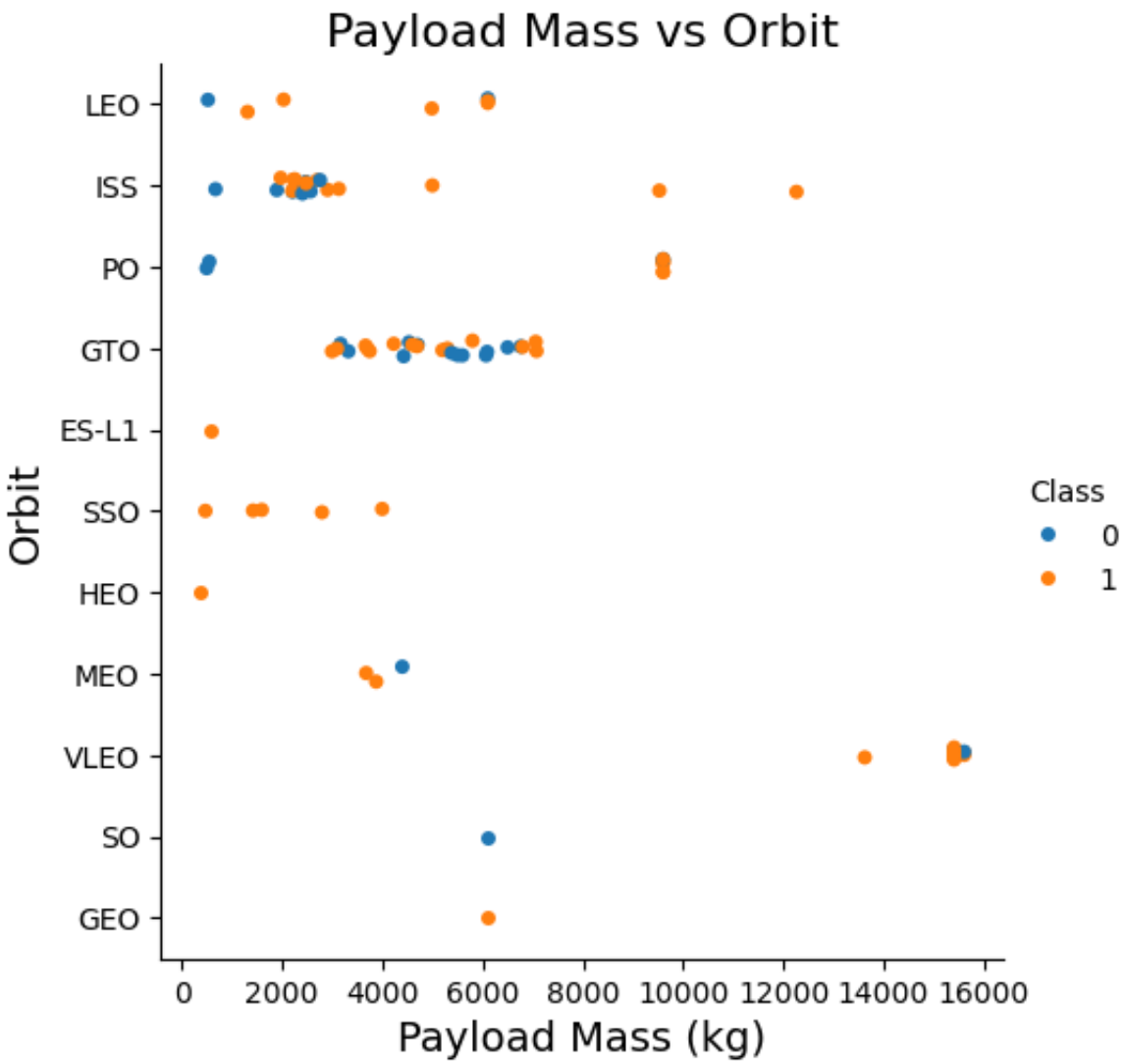


You can observe that in the LEO orbit, success seems to be related to the number of flights. Conversely, in the GTO orbit, there appears to be no relationship between flight number and success.

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 [9]:
```
# Plot a scatter point chart with x axis to be Payload Mass and y axis to
# Scatter plot showing the relationship between PayloadMass and Orbit
sns.catplot(x="PayloadMass", y="Orbit", hue="Class", data=df)
plt.xlabel('Payload Mass (kg)', fontsize=14)
plt.ylabel('Orbit', fontsize=14)
plt.title('Payload Mass vs Orbit', fontsize=16)
plt.show()
```



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 [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]:

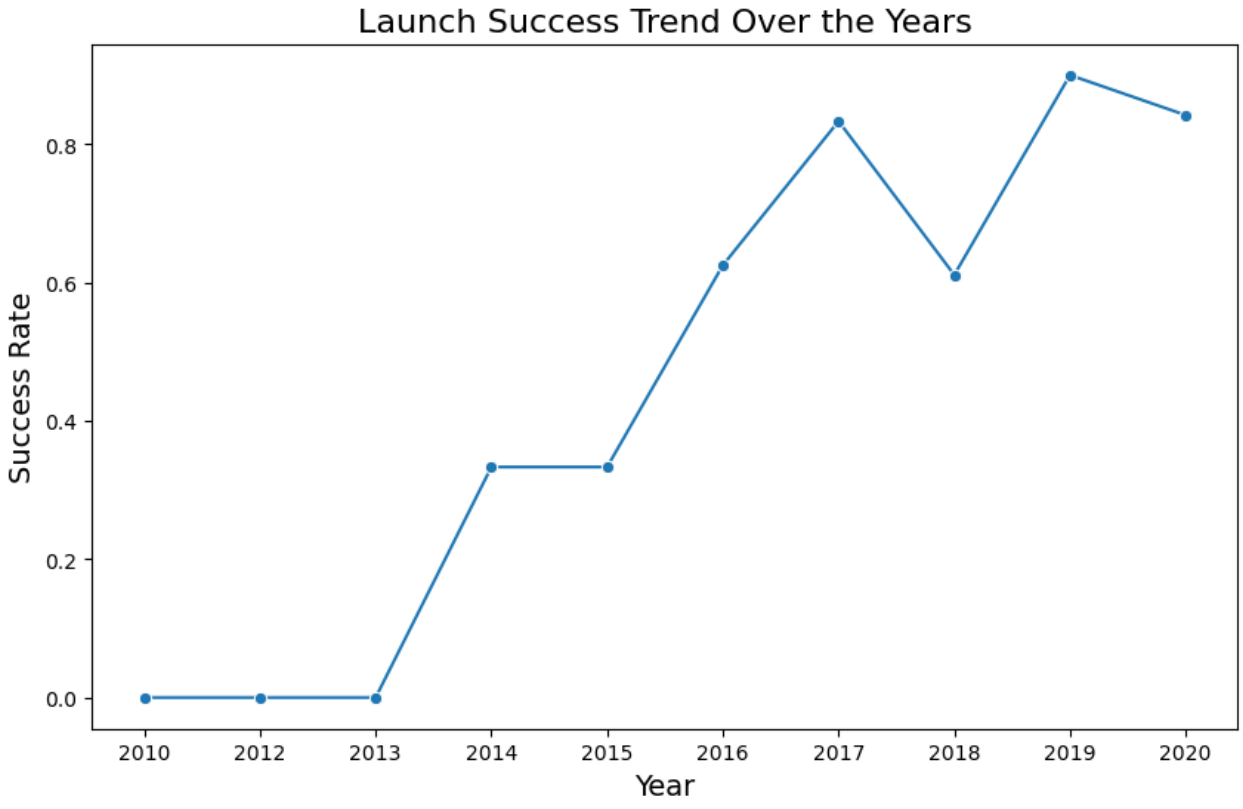
	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome
0	1	2010	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None
1	2	2012	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None
2	3	2013	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None
3	4	2013	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean
4	5	2013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

In [11]:

```
# Plot a line chart with x axis to be the extracted year and y axis to be
# Extract year from Date column
df['Year'] = df['Date'].apply(lambda x: x.split('-')[0])

# Group by Year and calculate the mean success rate (Class)
yearly_success = df.groupby('Year')['Class'].mean().reset_index()

# Line plot showing the success rate over the years
plt.figure(figsize=(10,6))
sns.lineplot(x='Year', y='Class', data=yearly_success, marker='o')
plt.xlabel('Year', fontsize=14)
plt.ylabel('Success Rate', fontsize=14)
plt.title('Launch Success Trend Over the Years', fontsize=16)
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.

In [12]:
```
features = df[['FlightNumber', 'PayloadMass', 'Orbit', 'LaunchSite', 'Fli
features.head()
```

Out[12]:

	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
3	4	500.000000	PO	VAFB SLC 4E	1	False	False	False
4	5	3170.000000	GTO	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.

In [13]:
```
# HINT: Use get_dummies() function on the categorical columns
# Create dummy variables for categorical columns
df_dummies = pd.get_dummies(df, columns=['Orbit', 'LaunchSite', 'LandingP
df_dummies.head()
```

Out[13]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Outcome	Flights	GridFins
0	1	2010	Falcon 9	6104.959412	None None	1	False
1	2	2012	Falcon 9	525.000000	None None	1	False
2	3	2013	Falcon 9	677.000000	None None	1	False
3	4	2013	Falcon 9	500.000000	False Ocean	1	False
4	5	2013	Falcon 9	3170.000000	None None	1	False

5 rows × 87 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 [15]:

```
# HINT: use astype function
# Check the data types of each column
print(df.dtypes)

# Select only the numeric columns
numeric_columns = df.select_dtypes(include=['float64', 'int64']).columns

# Cast only the numeric columns to float64
df[numeric_columns] = df[numeric_columns].astype('float64')

# Now you can verify if the casting was successful
print(df[numeric_columns].dtypes)
```

```
FlightNumber        int64
Date               object
BoosterVersion     object
PayloadMass       float64
Orbit              object
LaunchSite         object
Outcome            object
Flights             int64
GridFins             bool
Reused               bool
Legs                 bool
LandingPad         object
Block             float64
ReusedCount         int64
Serial             object
Longitude         float64
Latitude          float64
Class               int64
Year               object
dtype: object
FlightNumber    float64
PayloadMass     float64
Flights         float64
Block           float64
ReusedCount     float64
Longitude       float64
Latitude        float64
Class           float64
dtype: object
```

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