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]:

```
# 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 matrices, along with a large
collection of high-level mathematical functions to operate on these arrays

import numpy as np

# Matplotlib is a plotting library for python and pyplot gives us a MatLab
like plotting framework. We will use this in our plotter function to plot
data.

import matplotlib.pyplot as plt

# Seaborn is a Python data visualization library based on matplotlib. It
provides a high-level interface for drawing attractive and informative
statistical graphics

import seaborn as sns
```

Exploratory Data Analysis

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

In [2]:

```
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-
storage.appdomain.cloud/IBM-DS0321EN-
SkillsNetwork/datasets/dataset_part_2.csv")

# 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/IBMDeveloperSkillsNetwork-DS0701EN-
SkillsNetwork/api/dataset_part_2.csv')

df.head(5)
```

Out[2]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude	Class
0	1	2010-06-	Falcon 9	6104.9512	LEO	CCAFSLC40	None None	1	False	False	False	NaN	1.0	0	B0003	-80.577366	28.5618 57	0

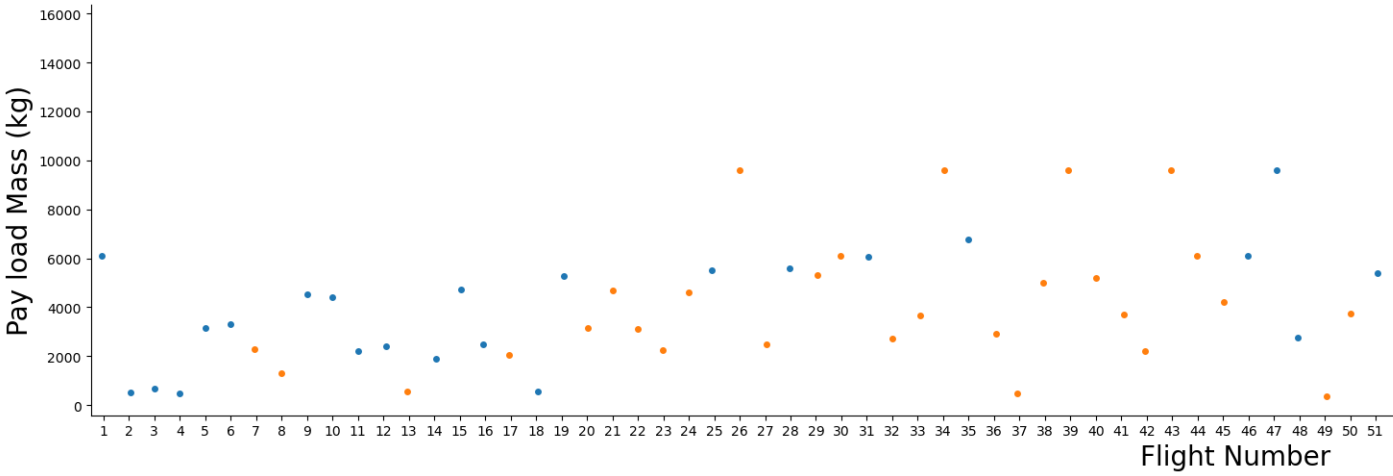
	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude	Class
		04																
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFSSLC40	None None	1	False	False	False	NaN	1.0	0	B0005	-80.577366	28.561857	0
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFSSLC40	None None	1	False	False	False	NaN	1.0	0	B0007	-80.577366	28.561857	0
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFBSLC4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003	-120.610829	34.632093	0
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFSSLC40	None None	1	False	False	False	NaN	1.0	0	B1004	-80.577366	28.561857	0

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. PayloadMassand 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 [3]:

```
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.

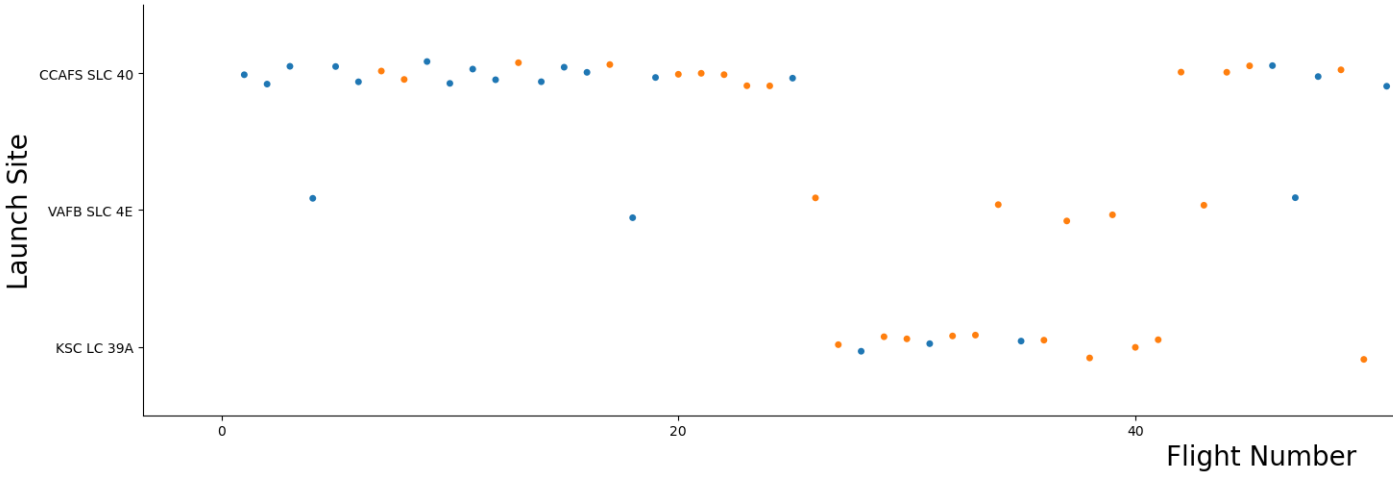
TASK 1: Visualize the relationship between Flight Number and Launch Site

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 [4]:

```
# 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 class value

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.

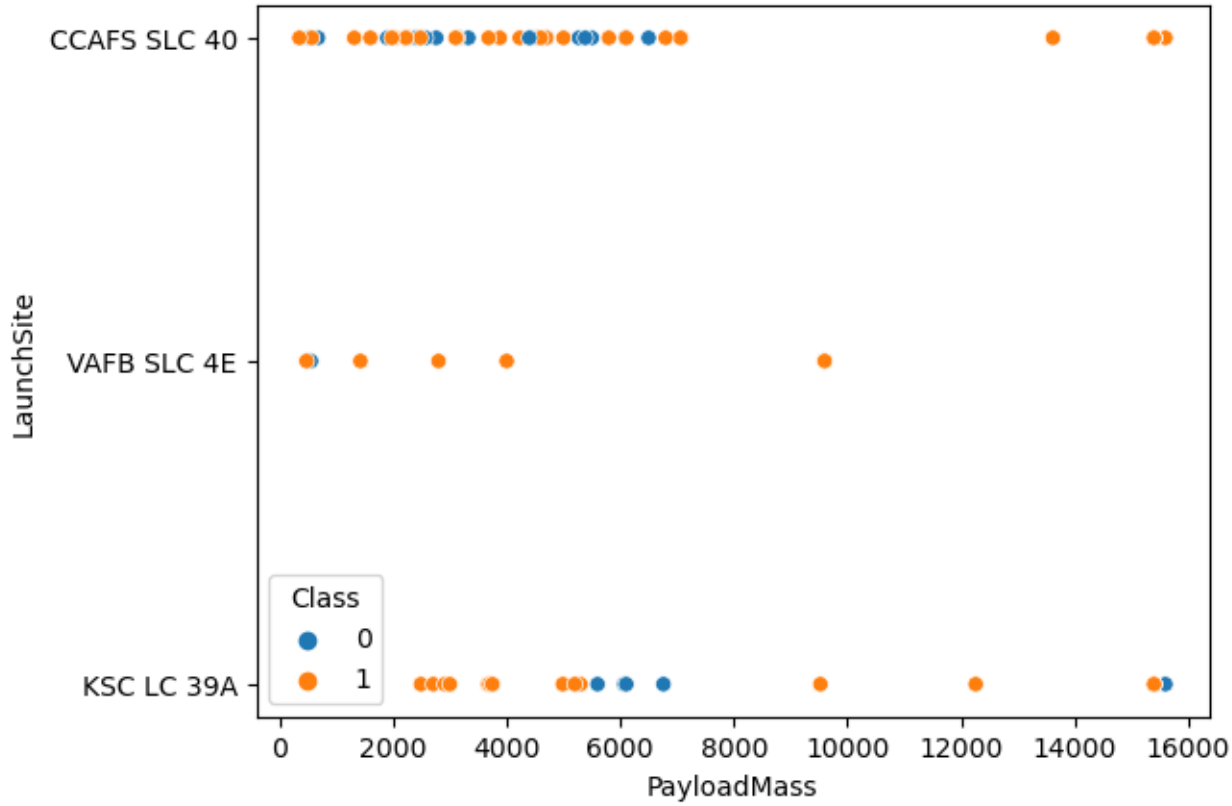
TASK 2: Visualize the relationship between Payload and Launch Site

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

In [5]:

```
# 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 be the class value

sns.scatterplot(data=df, x="PayloadMass", y="LaunchSite", hue="Class")
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.

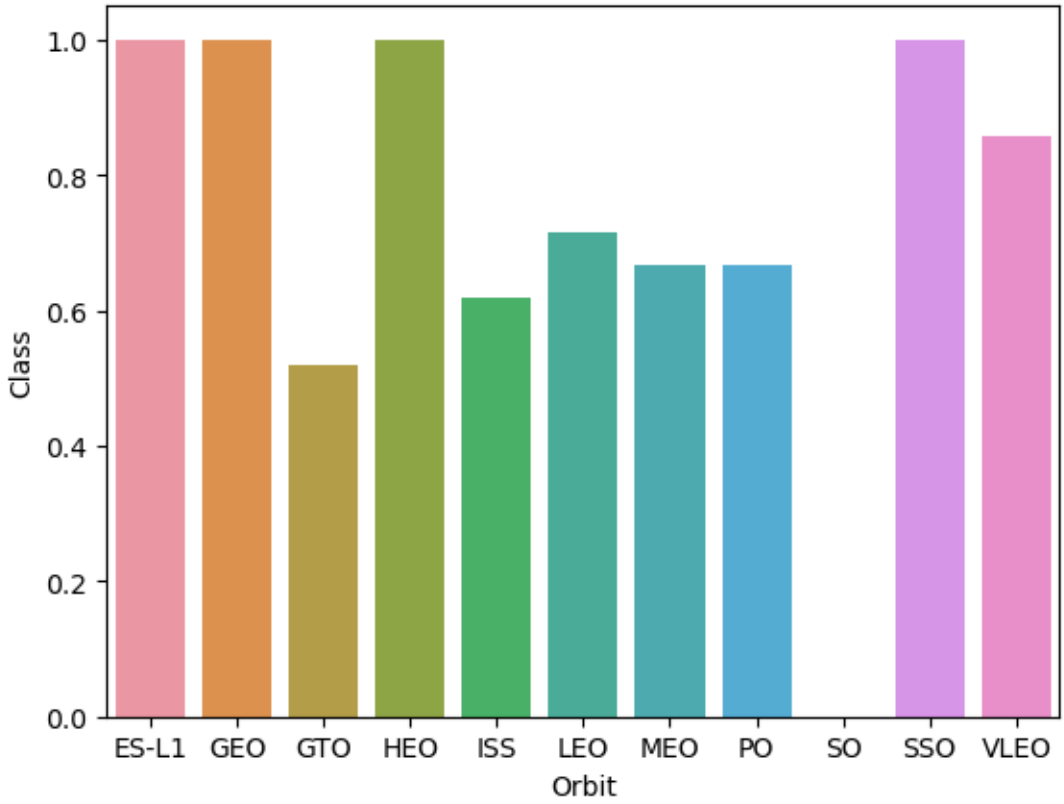
Let's create a bar chart for the sucess rate of each orbit

In [6]:

```
# HINT use groupby method on Orbit column and get the mean of Class column

df_orbitsuccess = df.groupby('Orbit').mean().reset_index()

sns.barplot(data=df_orbitsuccess, x='Orbit', y='Class')
plt.show()
```



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

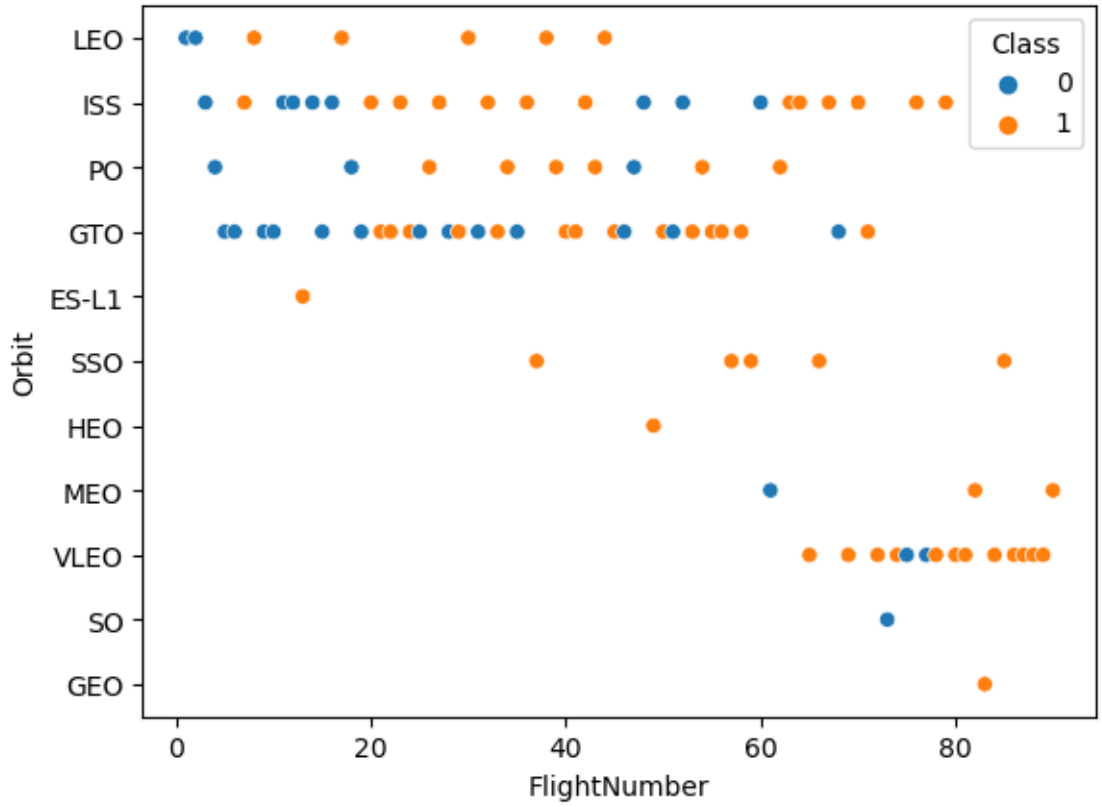
TASK 4: Visualize the relationship between FlightNumber and Orbit type

For each orbit, we want to see if there is any relationship between FlightNumber and Orbit type.

In [7]:

```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to
be the Orbit, and hue to be the class value

sns.scatterplot(data=df, x='FlightNumber', y='Orbit', hue='Class')
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.

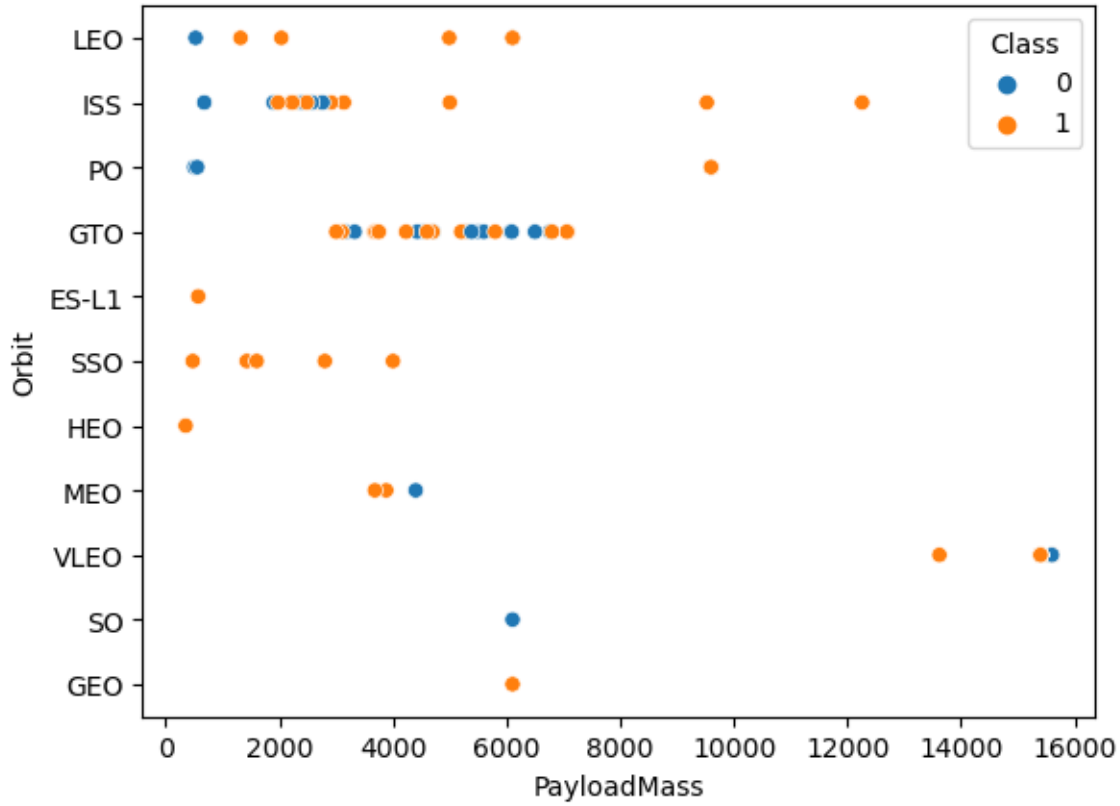
TASK 5: Visualize the relationship between Payload and Orbit type

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

In [8]:

```
# 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.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:

In [9]:

```
# 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
```

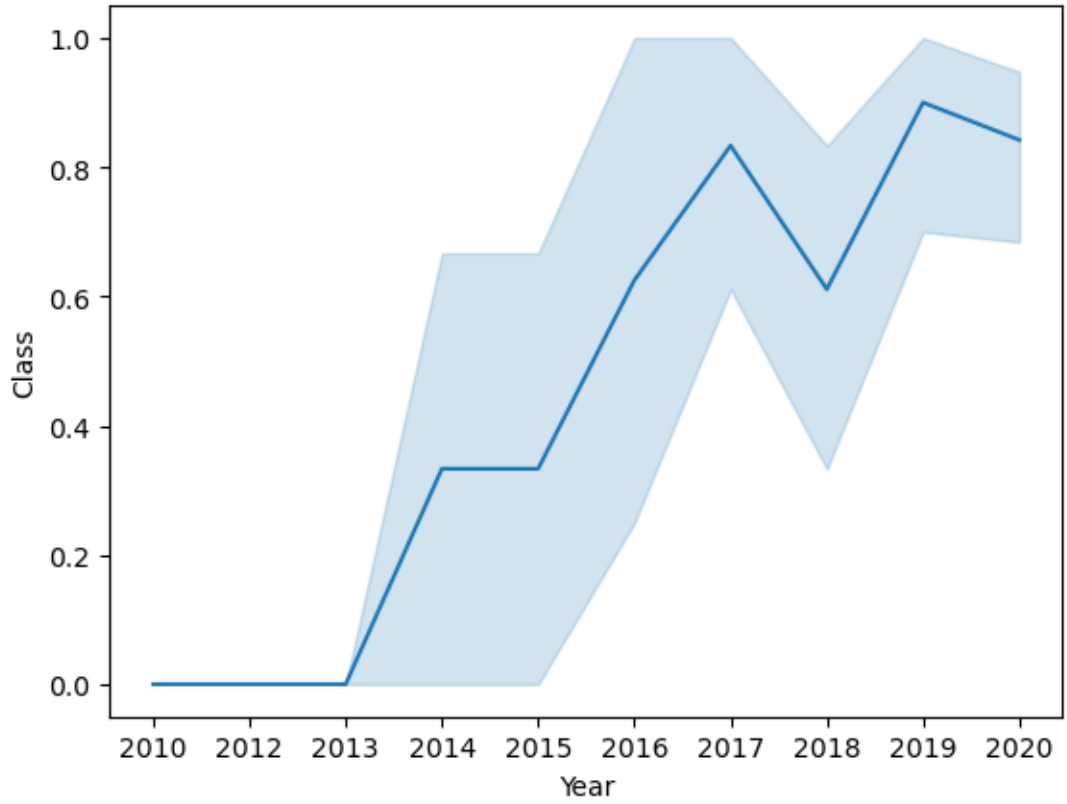
In [10]:

```
df['Year'] = Extract_year('Date')
```

In [11]:

```
# 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='Year', y='Class')
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',
'Flights', 'GridFins', 'Reused', 'Legs', 'LandingPad', 'Block',
'ReusedCount', 'Serial']]
features.head()
```

Out[12]:

	FlightNumber	PayloadMass	Orbit	LaunchSite	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial
0	1	6104.959412	LEO	CCAFSSLC 40	1	False	False	False	NaN	1.0	0	B0003
1	2	525.000000	LEO	CCAFSSLC 40	1	False	False	False	NaN	1.0	0	B0005
2	3	677.000000	ISS	CCAFSSLC 40	1	False	False	False	NaN	1.0	0	B0007

	FlightNumber	PayloadMass	Orbit	LaunchSite	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial
3	4	500.000000	PO	VAFB SLC 4E	1	False	False	False	NaN	1.0	0	B1003
4	5	3170.000000	GTO	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B1004

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

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

Out[13]:

	FlightNumber	PayloadMass	Flights	GridFins	Reused	Legs	Block	ReusedCount	Orbit_ES-L1	Orbit_GEO	...	Serial_B1048	Serial_B1049	Serial_B1050	Serial_B1051	Serial_B1054	Serial_B1056	Serial_B1058	Serial_B1059	Serial_B1060	Serial_B1062
0	1	6104.959412	1	False	False	False	1.0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	2	525.000000	1	False	False	False	1.0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	3	677.000000	1	False	False	False	1.0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

	FlightNumber	PayloadMass	Flights	GridFins	Reused	Legs	Block	ReusedCount	Orbit_ES-L1	Orbit_GEO	...	Serial_B1048	Serial_B1049	Serial_B1050	Serial_B1051	Serial_B1054	Serial_B1056	Serial_B1058	Serial_B1059	Serial_B1060	Serial_B1062
3	4	500.000000	1	False	False	False	1.0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	5	3170.000000	1	False	False	False	1.0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 80 columns