

File Edit View Run Kernel Tabs Settings Help

Launcher edadataviz.ipynb Python (Pyodide)

Skills Network

## 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

PERFECTING PROPULSIVE LANDING

Several examples of an unsuccessful landing are shown here:

APRIL 2016 FIRST SUCCESSFUL DRONESHIP LANDING

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 pipelite
await pipelite.install(['numpy'])
await pipelite.install(['pandas'])
await pipelite.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 matrices.
import numpy as np
# Matplotlib is a plotting library for python and pyplot gives us a MatLab like plotting framework. We will use this to plot our 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 statistical graphics.
import seaborn as sns
```

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

```
[3]: from js import fetch
import io

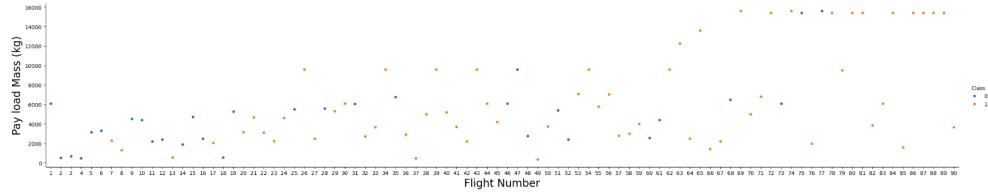
URL = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/data
resp = await fetch(URL)
dataset_part_2_csv = io.BytesIO(await resp.arrayBuffer()).to_py()
df=pd.read_csv(dataset_part_2_csv)
df.head(5)
```

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

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.

```
[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()
```



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

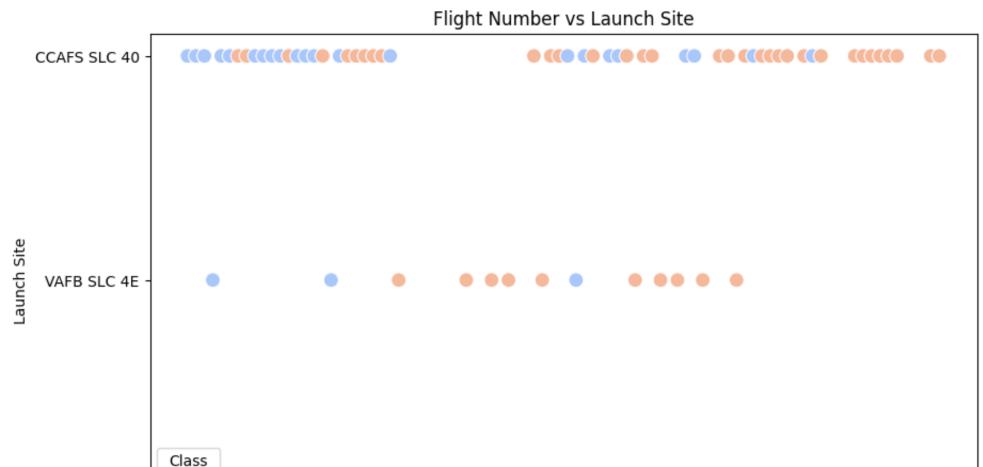
### TASK 1: Visualize the relationship between Flight Number and Launch Site

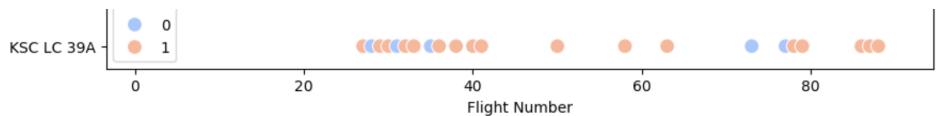
Use the function `catplot` to plot `FlightNumber` vs `LaunchSite`, set the parameter `x` parameter to `FlightNumber`, set the `y` to `Launch Site` and set the parameter `hue` to 'class'

```
[5]: # 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
plt.figure(figsize=(10, 6))
scatter_plot = sns.scatterplot(data=df, x='FlightNumber', y='LaunchSite', hue='Class', palette='coolwarm', s=100)

# Adding title and labels
scatter_plot.set_title('Flight Number vs Launch Site')
scatter_plot.set_xlabel('Flight Number')
scatter_plot.set_ylabel('Launch Site')

# Display the plot
plt.show()
```



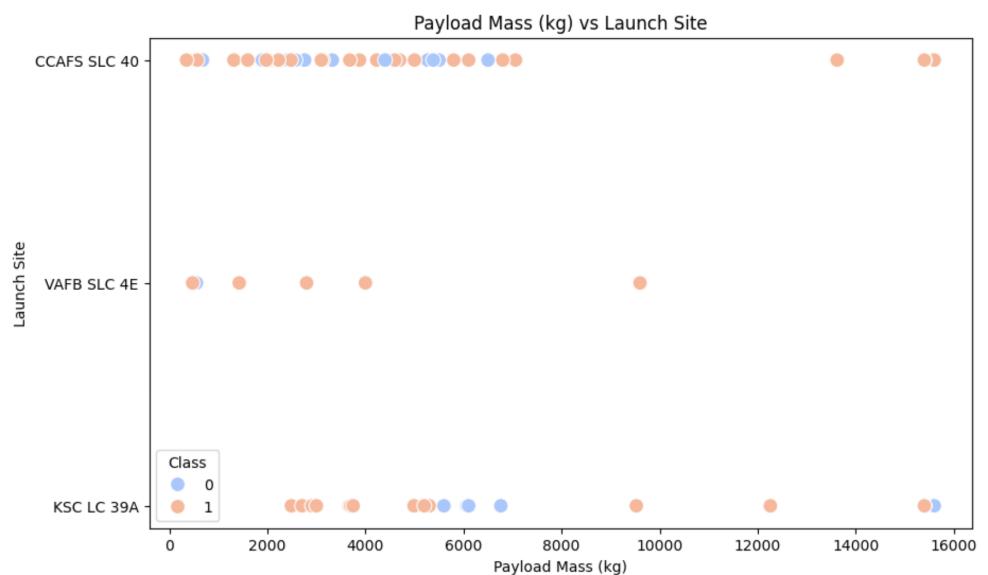


Now try to explain the patterns you found in the Flight Number vs. Launch Site scatter point plots.

### 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.

```
[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 to be
plt.figure(figsize=(10, 6))
scatter_plot = sns.scatterplot(data=df, x='PayloadMass', y='LaunchSite', hue='Class', palette='coolwarm', s=100)
scatter_plot.set_title('Payload Mass (kg) vs Launch Site')
scatter_plot.set_xlabel('Payload Mass (kg)')
scatter_plot.set_ylabel('Launch Site')
plt.show()
```



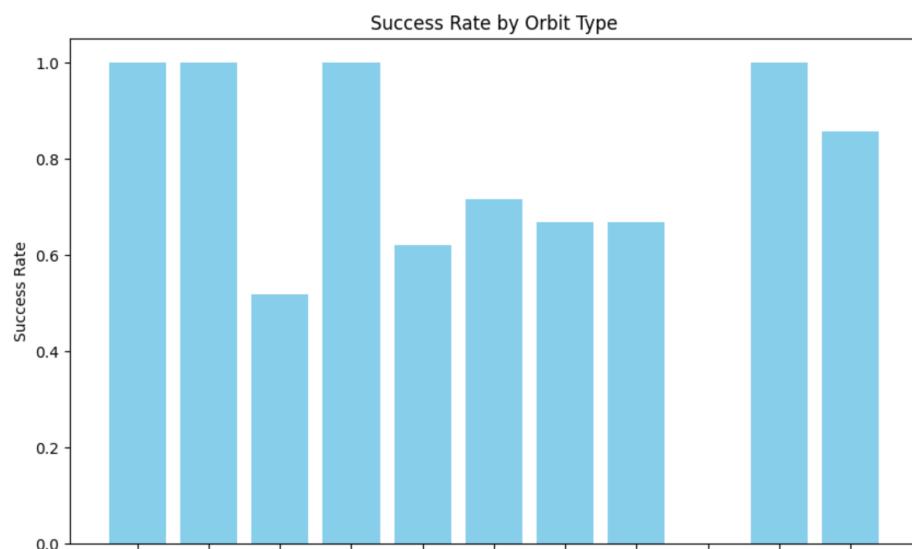
Now if you observe Payload Mass Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavy payload mass(greater than 10000).

### 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

```
[7]: # HINT use groupby method on Orbit column and get the mean of Class column
success_rate = df.groupby('Orbit')['Class'].mean().reset_index()
success_rate.columns = ['Orbit', 'Success Rate']
plt.figure(figsize=(10, 6))
bar_chart = plt.bar(success_rate['Orbit'], success_rate['Success Rate'], color='skyblue')
plt.title('Success Rate by Orbit Type')
plt.xlabel('Orbit Type')
plt.ylabel('Success Rate')
plt.show()
```

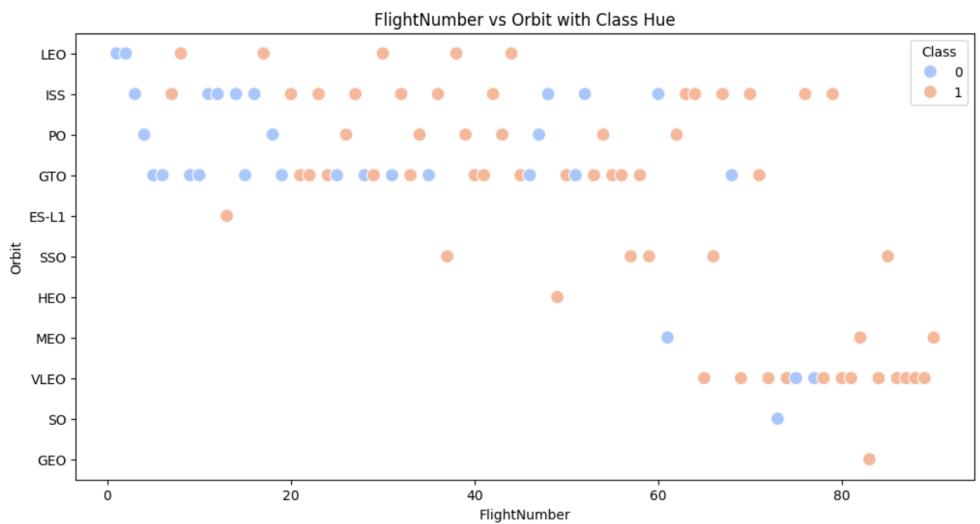


Analyze the plotted bar chart to identify which orbits have the highest success rates.

#### 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.

```
[8]: # Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class variable
plt.figure(figsize=(12, 6))
scatter_plot = sns.scatterplot(data=df, x='FlightNumber', y='Orbit', hue='Class', palette='coolwarm', s=100)
scatter_plot.set_title('FlightNumber vs Orbit with Class Hue')
scatter_plot.set_xlabel('FlightNumber')
scatter_plot.set_ylabel('Orbit')
plt.show()
```

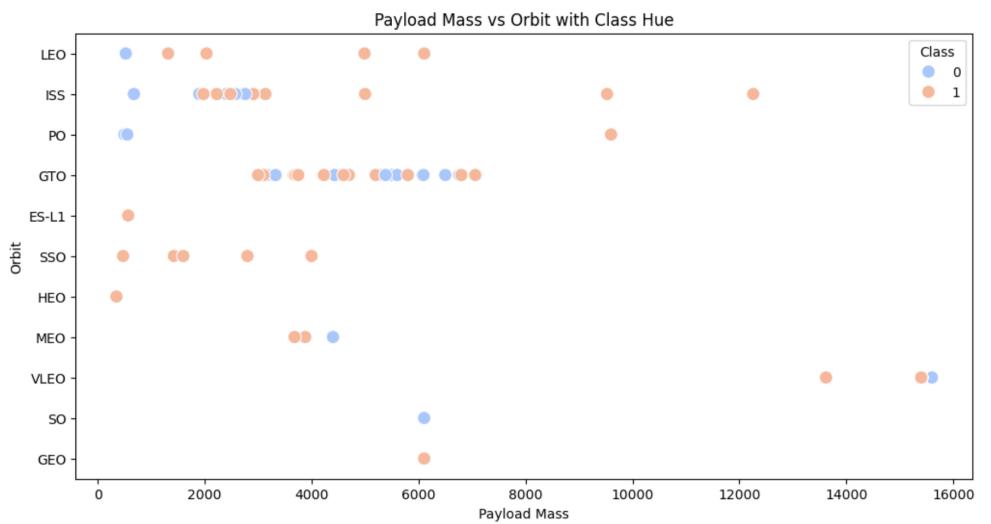


You can observe that in the LEO orbit, success seems to be related to the number of flights. Conversely, in the GTO orbit, there appears to be no relationship between flight number and success.

#### 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

```
[9]: # Plot a scatter point chart with x axis to be Payload Mass and y axis to be the Orbit, and hue to be the class variable
plt.figure(figsize=(12, 6))
scatter_plot = sns.scatterplot(data=df, x='PayloadMass', y='Orbit', hue='Class', palette='coolwarm', s=100)
scatter_plot.set_title('Payload Mass vs Orbit with Class Hue')
scatter_plot.set_xlabel('Payload Mass')
scatter_plot.set_ylabel('Orbit')
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

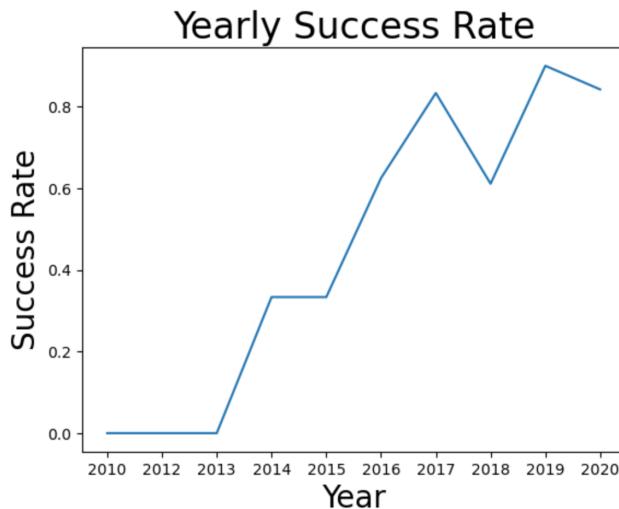
You can plot a line chart with x axis to be `Year` and y axis to be average success rate, to get the average launch success trend.

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

```
[14]: # A function to Extract years from the date
def Extract_year(date_series):
    return date_series.str.split("-").str[0]
```

```
[15]: # Plot a line chart with x axis to be the extracted year and y axis to be the success rate
df['Year'] = Extract_year(df['Date'])
df['Class'] = pd.to_numeric(df['Class'], errors='coerce')
years = df.groupby('Year')['Class'].mean()

sns.lineplot(x=years.index, y=years)
plt.xlabel('Year', fontsize=20)
plt.ylabel('Success Rate', fontsize=20)
plt.title('Yearly Success Rate', fontsize=24)
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.

```
[19]: features = df[['FlightNumber', 'PayloadMass', 'Orbit', 'LaunchSite', 'Flights', 'GridFins', 'Reused', 'Legs', 'LandingPad', 'Block', 'ReusedCount', 'Serial']]
features.head()
```

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

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

```
[20]: Orbit_ES-11 Orbit_GEO Orbit_GTO Orbit_HEO Orbit_ISS Orbit_LEO Orbit_MEO Orbit_PO Orbit_SO Orbit_SSO ... Serial_B
```