IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Jumakhan Rahyab
July 15, 2023

Outline

Executive Summary

Summary of methodologies

The analysis aims to predict the successful landing of the Falcon 9 first stage using different classification algorithms: Logistic Regression, Support Vector Machines (SVM), Decision Tree, and K-Nearest Neighbors (KNN). The data was preprocessed, and the models were trained and tested. The accuracy scores were compared to identify the best-performing model. The following is a step-by-step summary of the process, from data collection to model development:

- Data Collection via API
- Data Collection through Web Scraping
- Data Wrangling
- Exploratory Data Analysis using SQL
- Exploratory Data Analysis utilizing Data Visualization
- Interactive Visual Analytics with Folium
- Machine Learning Prediction

Executive Summary

Summary of all results

Different classification algorithms were evaluated to predict the successful landing of the Falcon 9 first stage. The results showed that SVM with the "sigmoid" kernel achieved the highest accuracy among the tested models, outperforming logistic regression, decision tree, and KNN. This suggests that SVM with the "sigmoid" kernel is a promising approach for predicting the success of Falcon 9 landings.

Introduction

Project background and context

In the context of the commercial space industry, where companies like SpaceX are making space travel more affordable through rocket reusability, this project focuses on predicting the success of Falcon 9 first stage landings. By leveraging publicly available data and employing machine learning techniques, the project aims to develop a predictive model that determines whether SpaceX will reuse the first stage for future launches. This information will be instrumental in estimating launch costs and assisting our team at Space Y, a new rocket company, in competing with SpaceX and making strategic decisions in the dynamic commercial space market.

Problems you want to find answers

The project aims to address several key questions in the context of commercial spaceflight. First, it seeks to determine the factors that contribute to the successful landing of Falcon 9 first stages, as this information is crucial in estimating launch costs. Additionally, the project seeks to understand the extent to which SpaceX can reuse the first stage and how this impacts the overall economics of space missions. By developing predictive models using machine learning techniques, the project aims to provide insights into the likelihood of first stage reuse and facilitate decision-making for Space Y, a new rocket company seeking to compete with SpaceX in the commercial space industry.

Section 1

Methodology

Methodology

Executive Summary

Data collection methodology:

Perform data wrangling

Perform exploratory data analysis (EDA) using visualization and SQL

Perform interactive visual analytics using Folium and Plotly Dash

Perform predictive analysis using classification models

Data Collection

The datasets were gathered from two sources: the SpaceX API and web scraping from the SpaceX Wiki page. To collect data via the REST API, I made a get request and decoded the response content as JSON. Using the json_normalize() function, I transformed the JSON data into a pandas dataframe. After that, I cleaned the data, checked for missing values, and normalized it for analysis.

For web scraping, I utilized the BeautifulSoup library to extract the launch records presented as an HTML table. I parsed the table and converted it into a pandas dataframe for further analysis.

Data Collection – SpaceX API

Step 1

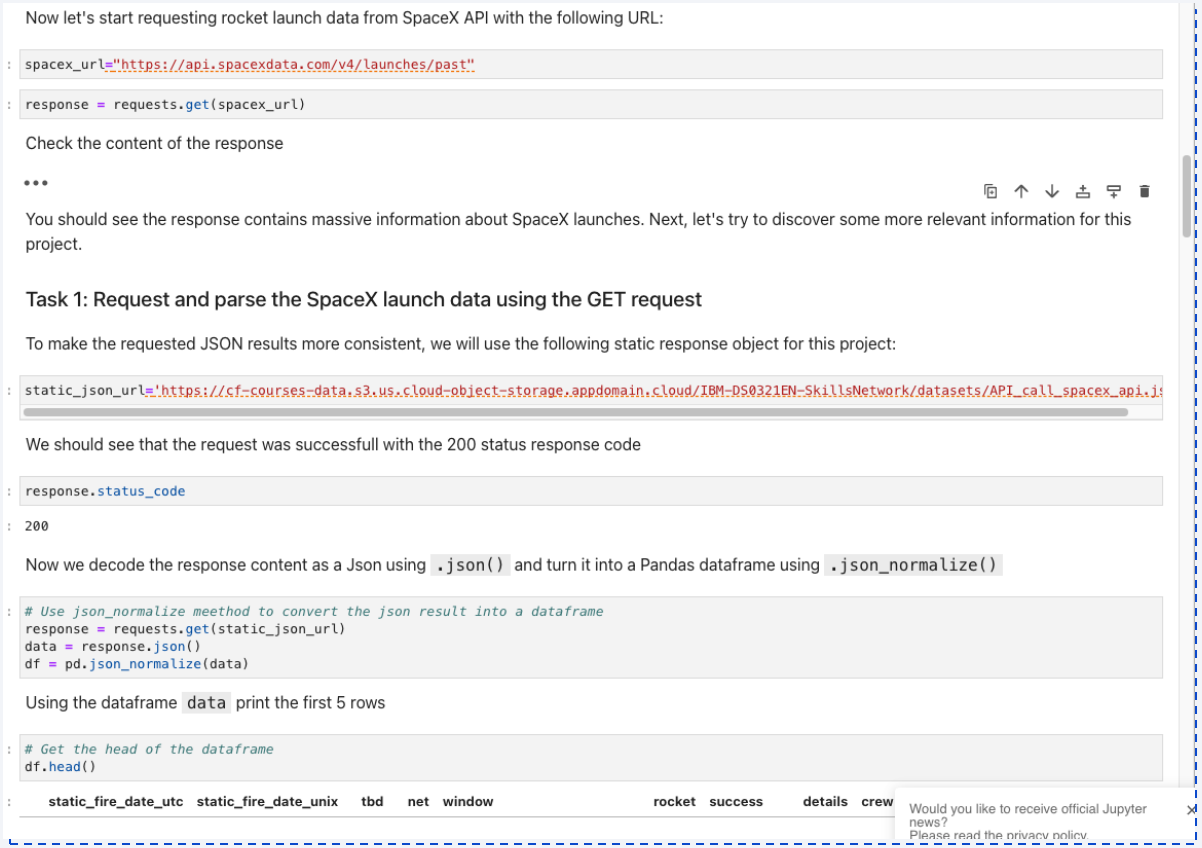
Get request for rocket launch data using API

Step 2

Use json_normalize method to convert json result to dataframe

Step 3

Performed data cleaning and filling the missing value



Now let's start requesting rocket launch data from SpaceX API with the following URL:

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

```
response = requests.get(spacex_url)
```

Check the content of the response

You should see the response contains massive information about SpaceX launches. Next, let's try to discover some more relevant information for this project.

Task 1: Request and parse the SpaceX launch data using the GET request

To make the requested JSON results more consistent, we will use the following static response object for this project:

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.j
```

We should see that the request was successfull with the 200 status response code

```
response.status_code
```

```
200
```

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

```
# Use json_normalize meethod to convert the json result into a dataframe
response = requests.get(static_json_url)
data = response.json()
df = pd.json_normalize(data)
```

Using the dataframe `data` print the first 5 rows

```
# Get the head of the dataframe
df.head()
```

	static_fire_date_utc	static_fire_date_unix	tbd	net	window	rocket	success	details	crew

Would you like to receive official Jupyter news?
Please read the privacy policy.

- GitHub URL: https://github.com/jumakhanrahyab/IBM-Applied-Data-Science-Capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

Data Collection - Scraping

Step 1

Step 1

- Request the Falcon9 Launch Wiki page from url

Step 2

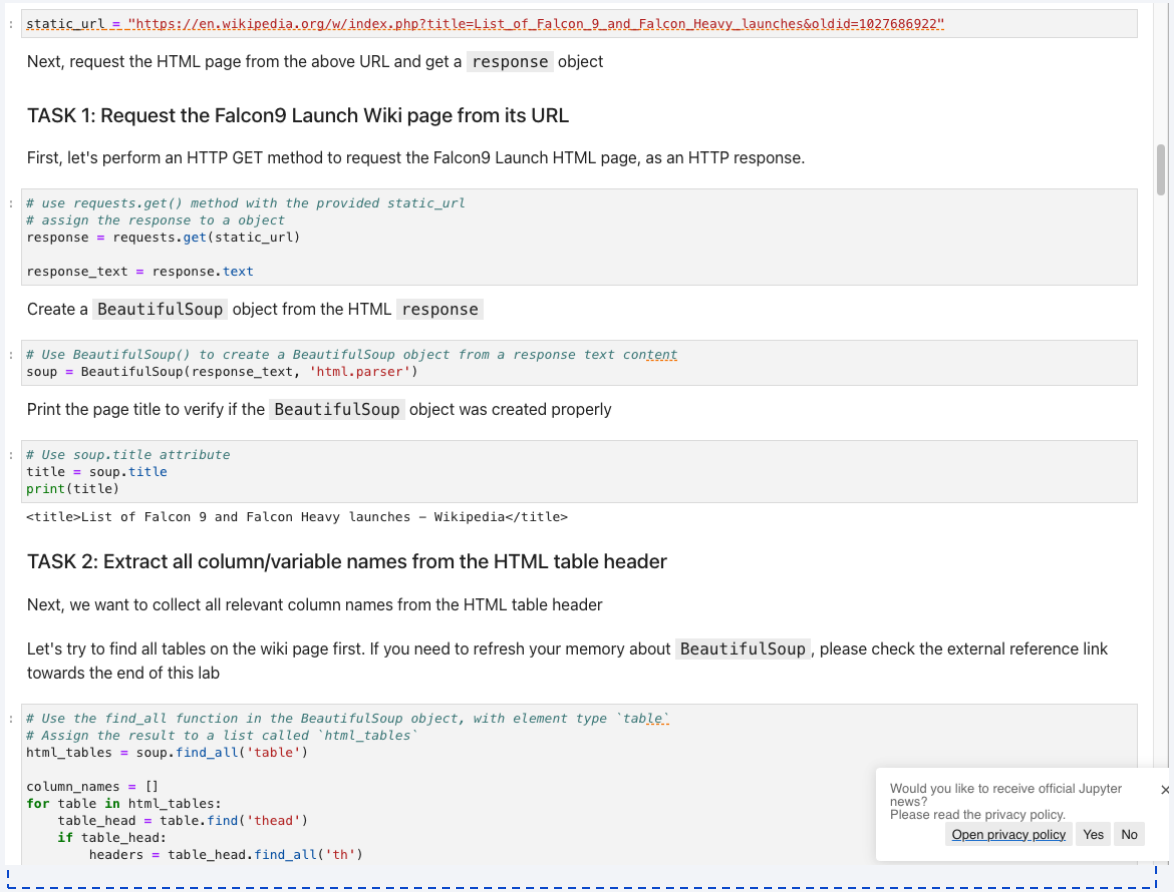
Step 2

- Create a BeautifulSoup from the HTML response

Step 3

Step 3

- Extract all column/variable names from the HTML header



```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

Next, request the HTML page from the above URL and get a `response` object

TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

```
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url)

response_text = response.text
```

Create a `BeautifulSoup` object from the HTML `response`

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response_text, 'html.parser')
```

Print the page title to verify if the `BeautifulSoup` object was created properly

```
# Use soup.title attribute
title = soup.title
print(title)
```

```
<title>List of Falcon 9 and Falcon Heavy launches – Wikipedia</title>
```

TASK 2: Extract all column/variable names from the HTML table header

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please check the external reference link towards the end of this lab

```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')

column_names = []
for table in html_tables:
    table_head = table.find('thead')
    if table_head:
        headers = table_head.find_all('th')
```


- GitHub URL: https://github.com/jumakhanrahyab/IBM-Applied-Data-Science-Capstone/blob/main/jupyter-labs-webscraping.ipynb

Data Wrangling

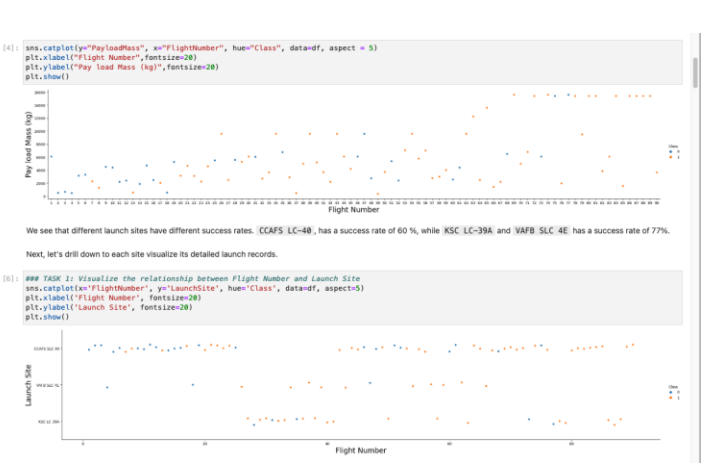
- During the data wrangling process, I performed several tasks to prepare the data for analysis. I calculated the number of launches on each launch site to gain insights into the distribution of launch activities. I also determined the number and occurrence of each orbit, providing information about the frequency and types of orbital paths. Additionally, I examined the number and occurrence of mission outcomes for each orbit type, allowing for a deeper understanding of success rates across different orbits. Lastly, I created a landing outcome label by categorizing the data in the Outcome column, enabling further analysis and prediction related to the landing success of missions. These data wrangling tasks played a crucial role in organizing and structuring the data for subsequent analysis and modeling steps.

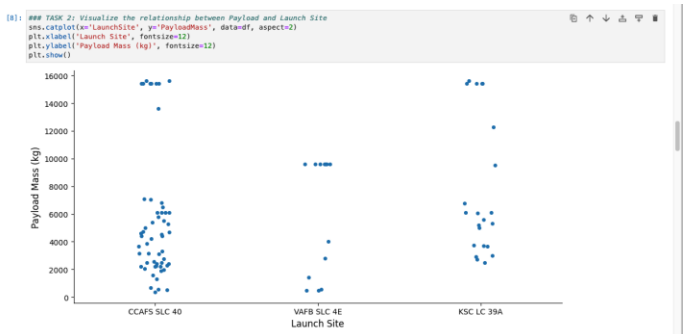
- GitHub URL: https://github.com/jumakhanrahyab/IBM-Applied-Data-Science-Capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

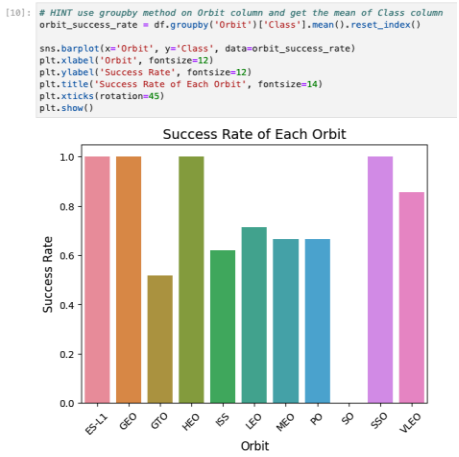
EDA with Data Visualization

Summary of charts plotted

During the analysis of Falcon 9 launches, I created and utilized various charts to gain insights and understand patterns in the data. For instance, I used a scatter chart to explore the correlation between payload mass and launch success, allowing for the identification of any potential relationships between these variables. These charts were instrumental in uncovering trends, patterns, and relationships within the Falcon 9 launch data, facilitating data-driven decision-making and further exploration of the factors influencing launch success.







- GitHub URL: https://github.com/jumakhanrahyab/IBM-Applied-Data-Science-Capstone/blob/main/IBM_labs_module_2_jupyter-labs-eda-dataviz.ipynb

EDA with SQL

- Summary of the SQL queries performed
  - Query of the names of the launch sites.
  - Query of 5 records where launch sites begin with the string 'CCA'.
  - Query of the total payload mass carried by booster launched by NASA (CRS).
  - Query of the average payload mass carried by booster version F9 v1.1.
  - Query of date when the first successful landing outcome in ground pad was achieved.
  - Query of names of the boosters which have success in drone ship and have payload mass 4000 but less than 6000.
  - Query of total number of successful and failure mission outcomes.
  - Query of names of the booster_versions which have carried the maximum payload mass.
  - Query of failed landing_outcomes in drone ship, their booster versions, and launch sites names for in year 2015.
  - Query of Rank the count of landing outcomes or success between the date 2010-06-04 and 2017-03-20, in descending order.

- GitHub URL: https://github.com/jumakhanrahyab/IBM-Applied-Data-Science-Capstone/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- Summary of map objects such as markers, circles, lines, etc. created and added to a folium map
  - Created and added markers to represent launch sites on the folium map.
  - Created and added circles to represent the proximity of the launch sites to the coastline.
  - Created and added lines to connect the launch site coordinates with the coordinates of the closest coastline.
  - Created and added a marker with distance information between the launch site and the closest coastline.
  - Created and added markers to represent the locations of successful Falcon 9 landings on the folium map.

- GitHub URL: https://github.com/jumakhanrahyab/IBM-Applied-Data-Science-Capstone/blob/main/IBM_labs_module_3_lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- Summary of plots/graphs and interactions added to a dashboard
  - Added a dropdown list to enable selection of launch sites.
  - Added a pie chart to show the total successful launches count for all sites. If a specific launch site is selected, it shows the success vs. failed counts for that site.
  - Added a range slider to select payload range.
  - Added a scatter chart to show the correlation between payload and launch success. The chart updates based on the selected launch site and payload range.
  - Implemented callback functions to update the charts based on user inputs.
  - Created an interactive dashboard layout using Dash HTML components.

- GitHub URL: https://github.com/jumakhanrahyab/IBM-Applied-Data-Science-Capstone/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

Summary of the process of building, evaluating, improving, and finding the best performing classification model

- Built a classification model for predicting the successful landing of Falcon 9 using various algorithms such as Logistic Regression, Support Vector Machines, Decision Trees, and K-Nearest Neighbors.
- Split the dataset into training and testing data using train_test_split.
- Utilized GridSearchCV to perform a grid search and find the best hyperparameters for each algorithm.
- Evaluated the performance of each model by calculating the accuracy score on the test data.
- Iterated on the models by adjusting the hyperparameters and re-evaluating their performance.
- Determined the best performing model based on the highest accuracy score on the test data.
- The logistic regression model was found to be the best performing model for predicting the successful landing of Falcon 9.

- GitHub URL: https://github.com/jumakhanrahyab/IBM-Applied-Data-Science-Capstone/blob/main/IBM_labs_module_4_SpaceX_Machine_Learning_Prediction_Part_5.ipynb

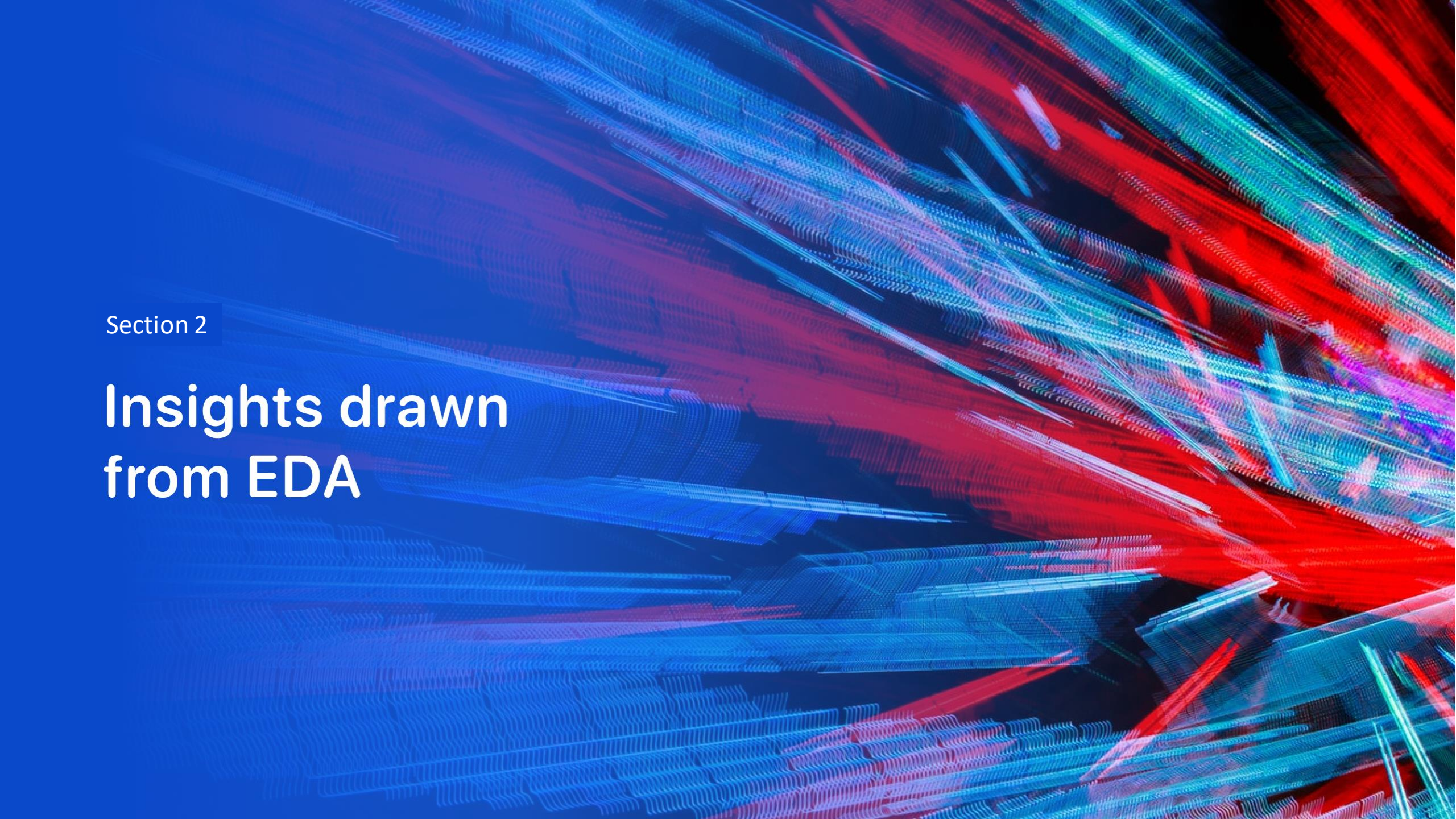
Results

Exploratory data analysis results

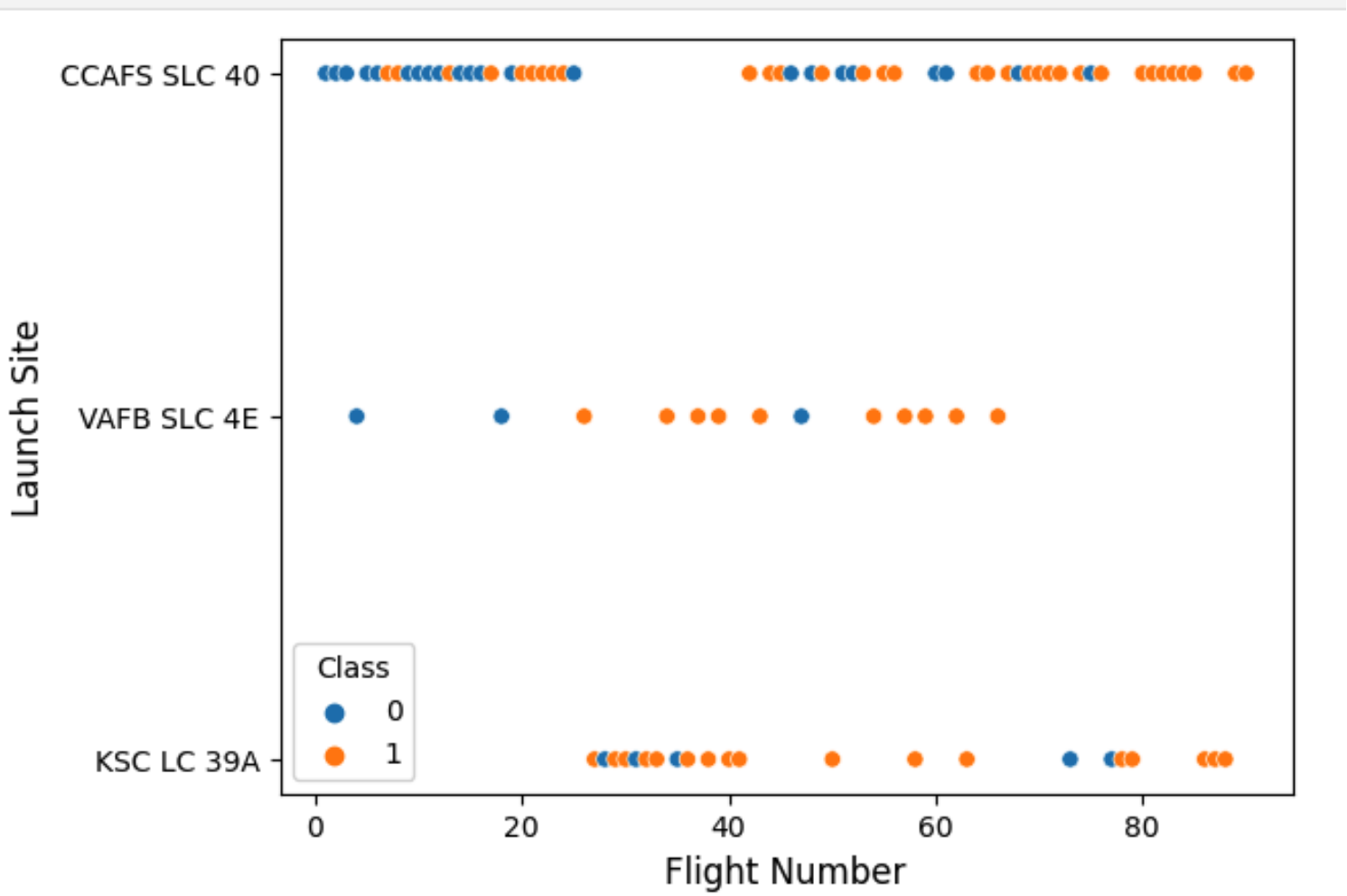
- Explored the Falcon 9 dataset and performed various data analysis tasks.
- Calculated the number of launches on each site and analyzed the distribution.
- Examined the number and occurrence of different orbits.
- Investigated the mission outcomes for each orbit type.
- Created a landing outcome label based on the outcome column.

Section 2

Insights drawn from EDA

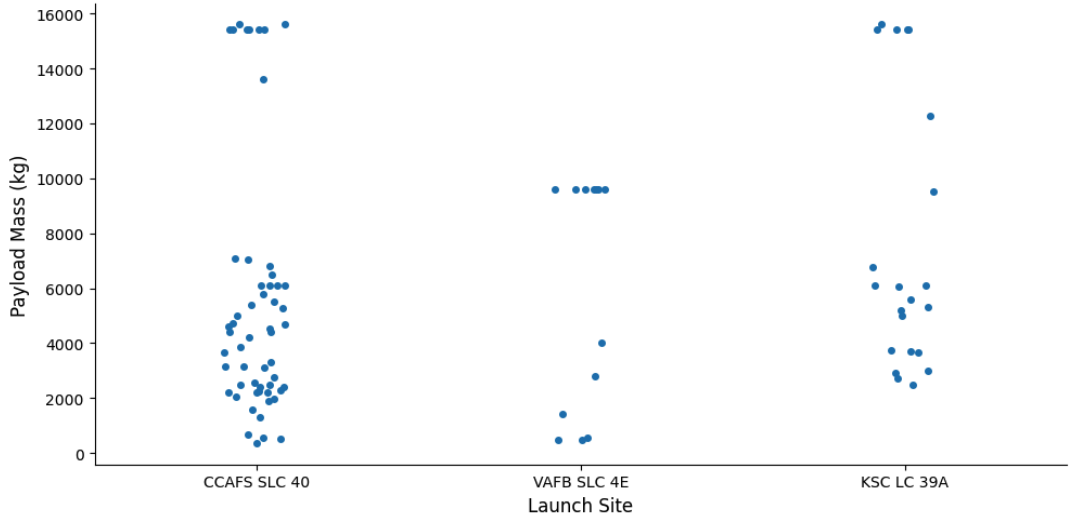


Flight Number vs. Launch Site

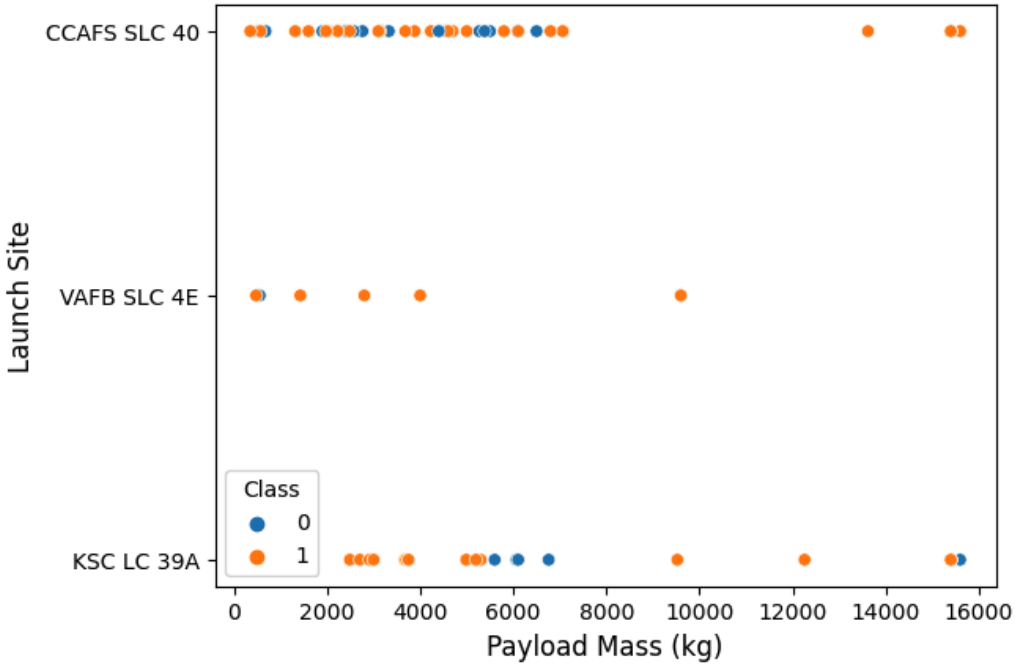


Payload vs. Launch Site

```
### TASK 2: Visualize the relationship between Payload and Launch Site
sns.catplot(x='LaunchSite', y='PayloadMass', data=df, aspect=2)
plt.xlabel('Launch Site', fontsize=12)
plt.ylabel('Payload Mass (kg)', fontsize=12)
plt.show()
```

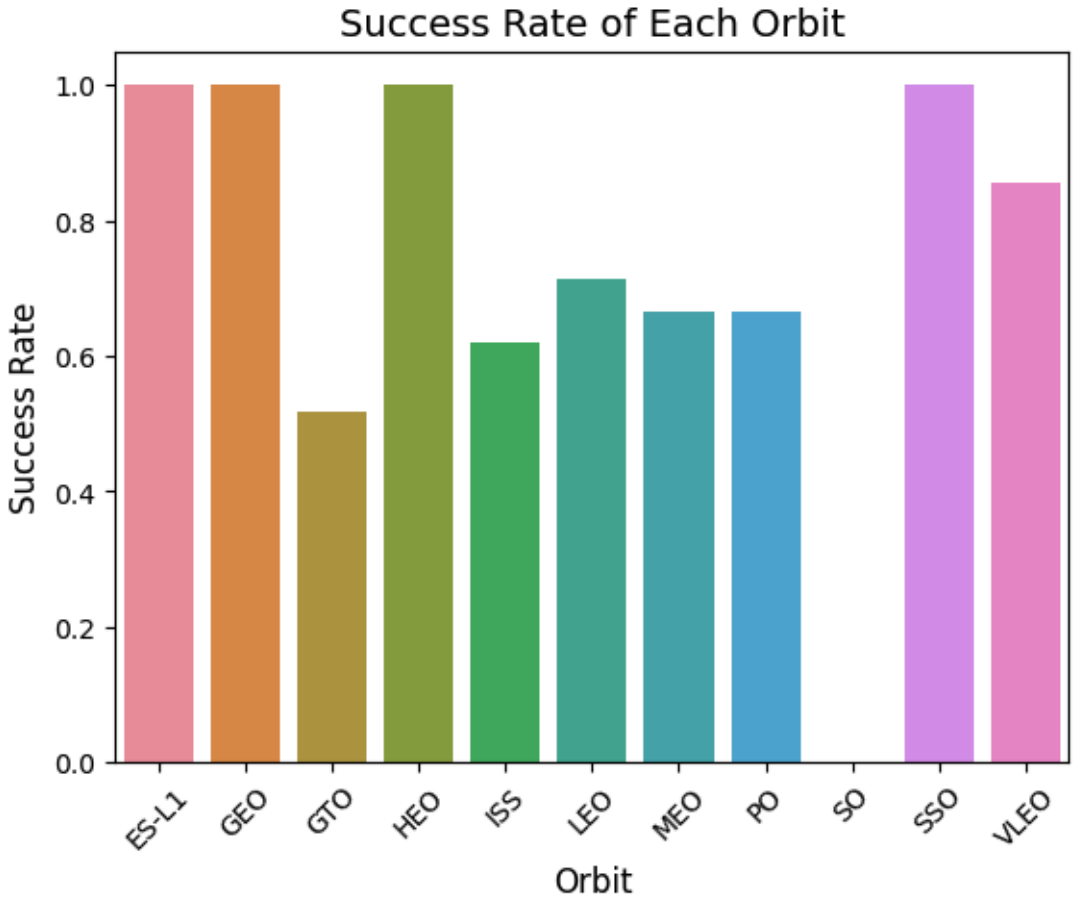



```
sns.scatterplot(x='PayloadMass', y='LaunchSite', hue='Class', data=df)
plt.xlabel('Payload Mass (kg)', fontsize=12)
plt.ylabel('Launch Site', fontsize=12)
plt.show()
```



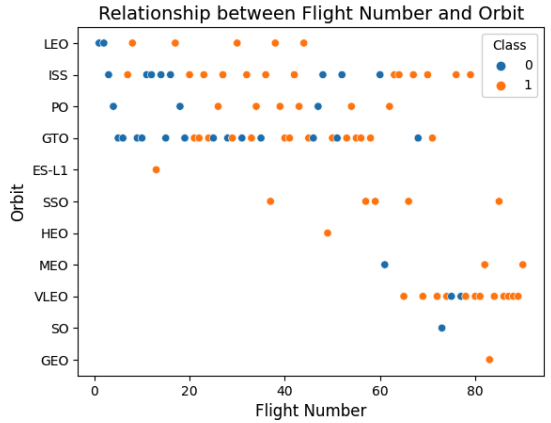
Success Rate vs. Orbit Type

```
# HINT use groupby method on Orbit column and get the mean of Class colum
orbit_success_rate = df.groupby('Orbit')['Class'].mean().reset_index()

sns.barplot(x='Orbit', y='Class', data=orbit_success_rate)
plt.xlabel('Orbit', fontsize=12)
plt.ylabel('Success Rate', fontsize=12)
plt.title('Success Rate of Each Orbit', fontsize=14)
plt.xticks(rotation=45)
plt.show()
```



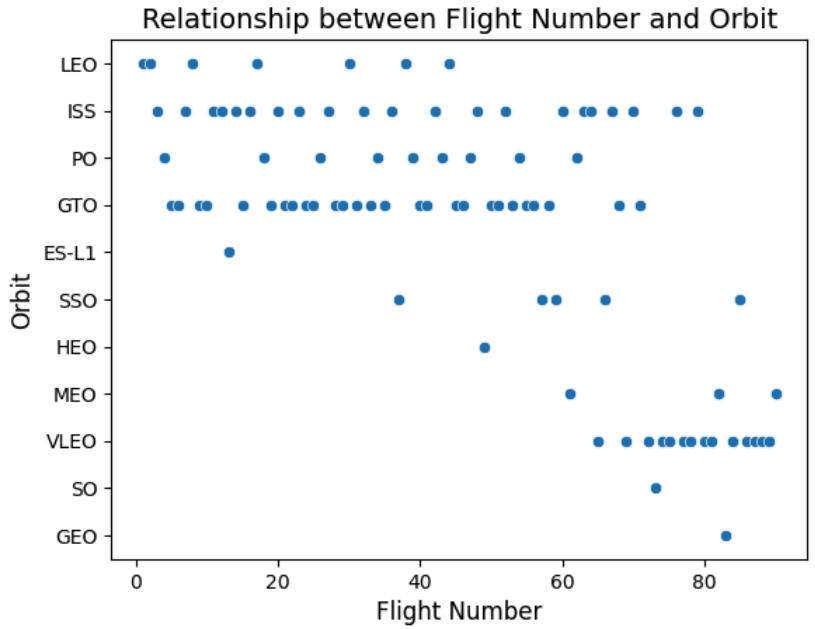
Flight Number vs. Orbit Type

```
# 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(x='FlightNumber', y='Orbit', hue='Class', data=df)
plt.xlabel('Flight Number', fontsize=12)
plt.ylabel('Orbit', fontsize=12)
plt.title('Relationship between Flight Number and Orbit', fontsize=14)
plt.show()
```



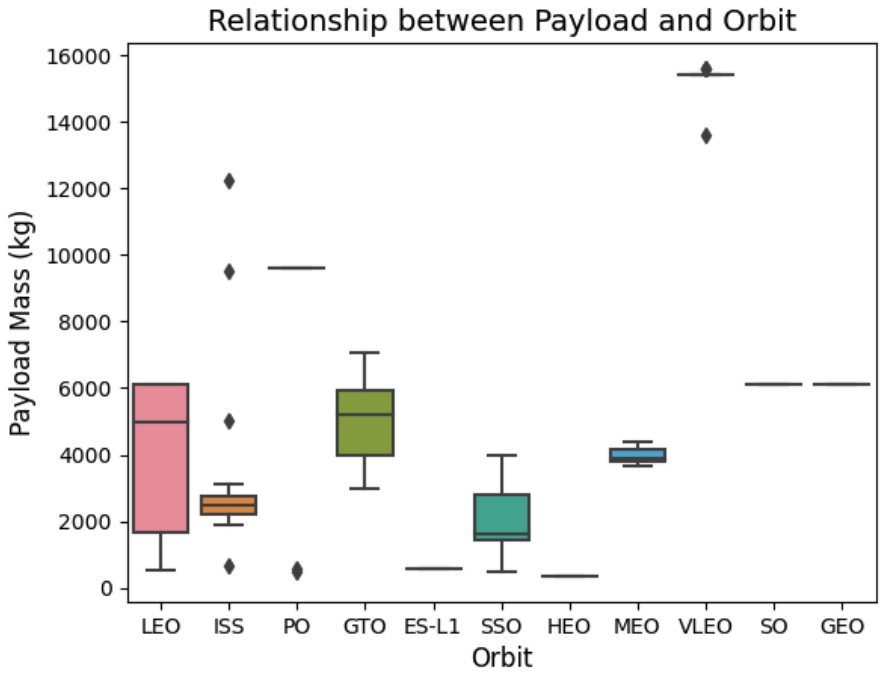
You should see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

```
### TASK  4: Visualize the relationship between FlightNumber and Orbit type
sns.scatterplot(x='FlightNumber', y='Orbit', data=df)
plt.xlabel('Flight Number', fontsize=12)
plt.ylabel('Orbit', fontsize=12)
plt.title('Relationship between Flight Number and Orbit', fontsize=14)
plt.show()
```

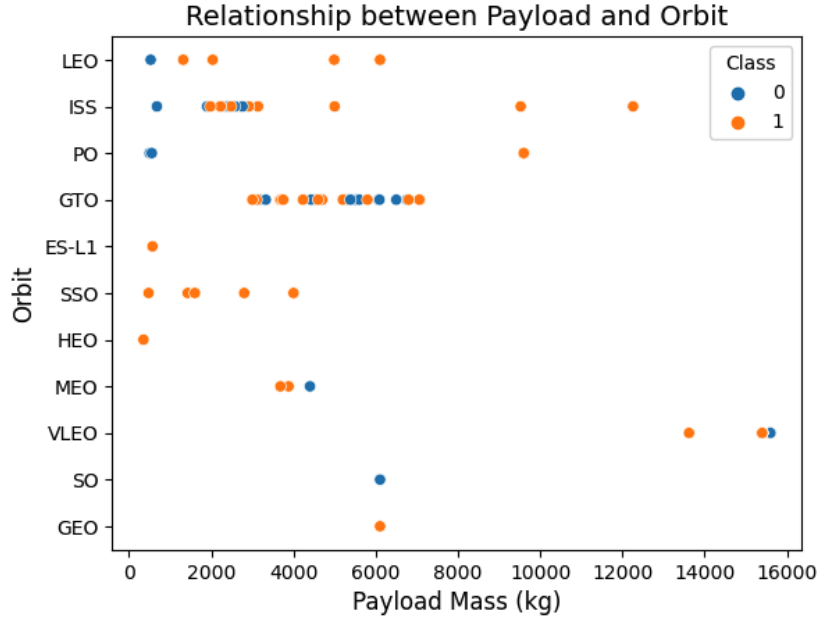


Payload vs. Orbit Type

```
sns.boxplot(x='Orbit', y='PayloadMass', data=df)
plt.xlabel('Orbit', fontsize=12)
plt.ylabel('Payload Mass (kg)', fontsize=12)
plt.title('Relationship between Payload and Orbit', fontsize=14)
plt.show()
```



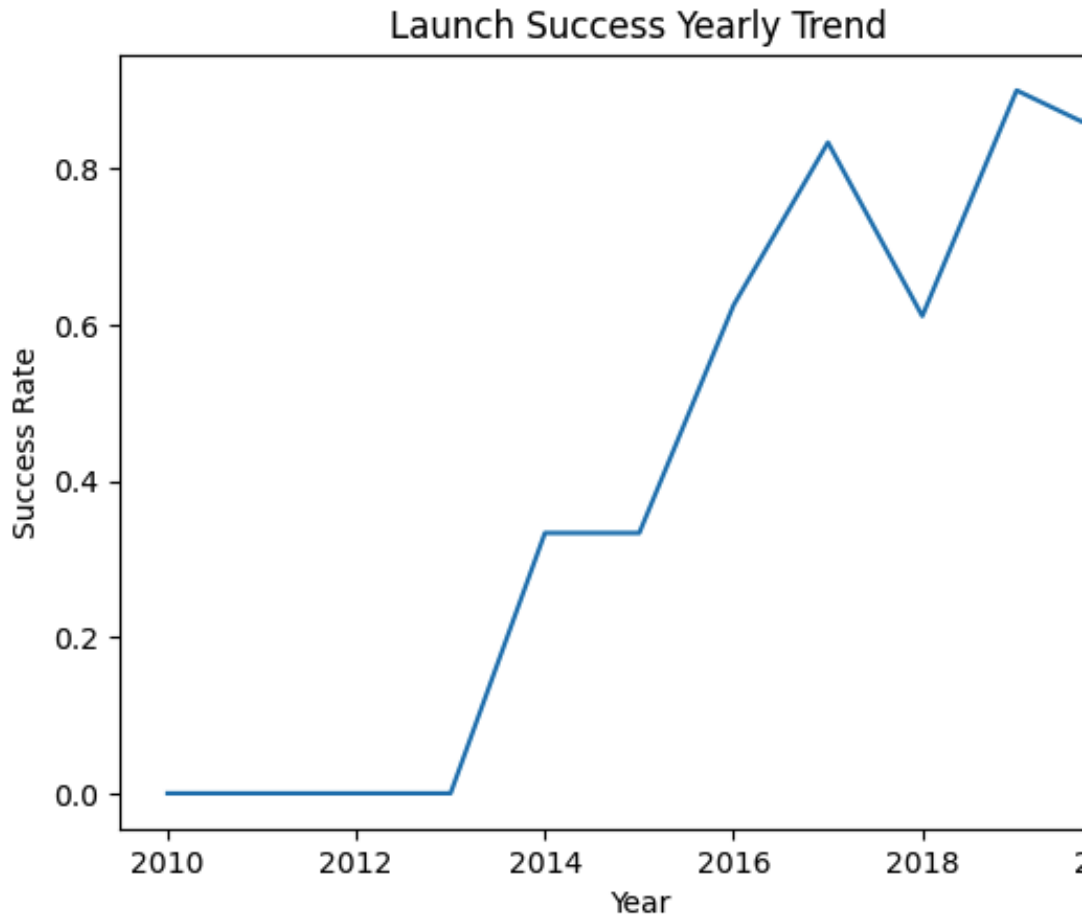

```
sns.scatterplot(x='PayloadMass', y='Orbit', hue='Class', data=df)
plt.xlabel('Payload Mass (kg)', fontsize=12)
plt.ylabel('Orbit', fontsize=12)
plt.title('Relationship between Payload and Orbit', fontsize=14)
plt.show()
```



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

Launch Success Yearly Trend

```
plt.plot(success_rate.index, success_rate.values)
plt.xlabel('Year')
plt.ylabel('Success Rate')
plt.title('Launch Success Yearly Trend')
plt.show()
```



you can observe that the sucess rate since 2013 kept increasing till 2020

All Launch Site Names

- Using the key word DISTINCT to show only unique launch sites from the SpaceX data.

```
[9]: %sql SELECT DISTINCT Launch_Site FROM SPACEXTBL;
 * sqlite:///my_data1.db
Done.
```

[9]:

Launch_Site
CCAFS LC-40
VAFB SLC-4E
KSC LC-39A
CCAFS SLC-40

None

Launch Site Names Begin with 'CCA'

- 5 records where launch sites begin with `CCA`

```
[14]: %sql SELECT * FROM SPACEXTBL WHERE Launch_Site = 'CCAFS LC-40' LIMIT 5;
 * sqlite:///my_data1.db
Done.
```

[14]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
06/04/2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0.0	LEO	SpaceX	Success	Failure (parachute)
12/08/2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0.0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
22/05/2012	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525.0	LEO (ISS)	NASA (COTS)	Success	No attempt
10/08/2012	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500.0	LEO (ISS)	NASA (CRS)	Success	No attempt
03/01/2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677.0	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- Calculating the total payload carried by boosters from NASA

```
[25]: %sql SELECT SUM(PAYLOAD_MASS__KG_) AS TotalPayloadMass FROM SPACEXTBL WHERE Customer LIKE 'NASA (CRS)%';
 * sqlite:///my_data1.db
Done.
[25]:
```

TotalPayloadMass
48213.0

Average Payload Mass by F9 v1.1

- Calculating the average payload mass carried by booster version F9 v1.1

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS AveragePayloadMass FROM SPACEXTBL WHERE `Booster_Version` = 'F9 v1.1';
```

```
 * sqlite:///my_data1.db
Done.
```

AveragePayloadMass
2928.4

First Successful Ground Landing Date

- Finding the dates of the first successful landing outcome on ground pad

```
%sql SELECT MIN(Date) AS FirstSuccessfulLandingDate FROM SPACEXTBL WHERE `Landing_Outcome` = 'Success (ground pad)';
```

```
 * sqlite:///my_data1.db
Done.
```

FirstSuccessfulLandingDate
01/08/2018

Successful Drone Ship Landing with Payload between 4000 and 6000

- Listing the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

```
]: %sql SELECT Booster_Version AS BoosterName FROM SPACEXTBL WHERE `Landing_Outcome` = 'Success (drone ship)' AND `PAYLOAD_MASS__KG_` > 4000 AND `PAYLOAD_MASS__KG_` < 6000;
 * sqlite:///my_data1.db
Done.
]:
```

BoosterName
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

- Calculating the total number of successful and failure mission outcomes

```
]: %sql SELECT `Mission_Outcome`, COUNT(*) AS Count FROM SPACEXTBL GROUP BY `Mission_Outcome`;
 * sqlite:///my_data1.db
Done.
```

]:

Mission_Outcome	Count
None	898
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Boosters Carried Maximum Payload

- Listing the names of the booster which have carried the maximum payload mass

```
]: %sql SELECT DISTINCT Booster_Version FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
 * sqlite:///my_data1.db
Done.
```

]:

Booster_Version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

2015 Launch Records

- Listing the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

```
%sql SELECT strftime('%m', Date) AS Month, Landing_Outcome, Booster_Version, Launch_Site FROM SPACEXTBL WHERE substr(Date, 7, 4) = '2015' AND
```

```
 * sqlite:///my_data1.db
Done.
```

Month	Landing_Outcome	Booster_Version	Launch_Site
None	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
None	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

```
]: %sql SELECT Landing_Outcome, COUNT(*) AS Count FROM SPACEXTBL WHERE Date BETWEEN '2010-06-04' AND '2017-03-20' AND (Landing_Outcome = 'Failu
 * sqlite:///my_data1.db
Done.
]:
```

Landing_Outcome	Count

Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

- Ranking the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order


Section 3
Launch Sites
Proximities Analysis

Location of NASA Johnson Space Center

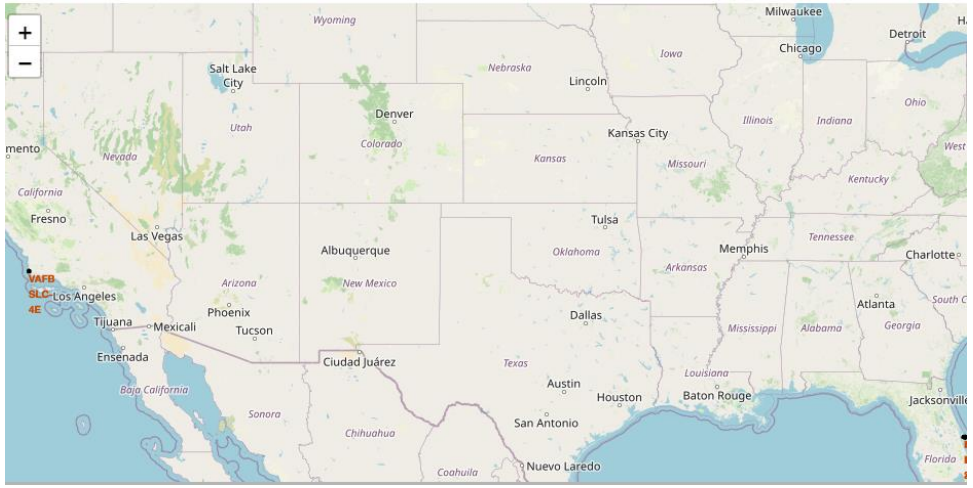
- Showing the location NASA Johnson Space Center

```
circle = folium.Circle(nasa_coordinate, radius=1000, color='#d35400', fill=True).add_child(folium.Popup('NASA Johnson Space Center'))
# Create a blue circle at NASA Johnson Space Center's coordinate with a icon showing its name
marker = folium.map.Marker(
    nasa_coordinate,
    # Create an icon as a text label
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % 'NASA JSC',
        )
    )
site_map.add_child(circle)
site_map.add_child(marker)
```

Location of all the Launch Sites

- Location of all SpaceX Launch Sites located

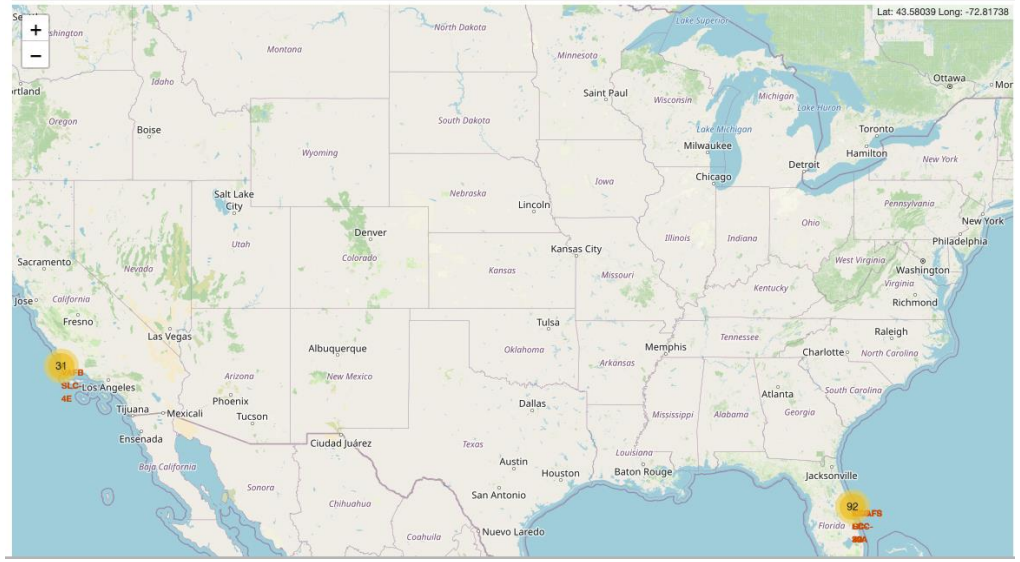
```
    lat = site['Lat']
    lon = site['Long']

    # Create a Circle object for the launch site with a popup label showing its name
    circle = folium.Circle(
        location=[lat, lon],
        radius=1000,
        color='#000000',
        fill=True
    ).add_child(folium.Popup(launch_site))
    site_map.add_child(circle)

    # Create a Marker object for the launch site with a custom icon
    marker = folium.Marker(
        location=[lat, lon],
        icon=DivIcon(
            icon_size=(20, 20),
            icon_anchor=(0, 0),
            html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % launch_site,
        )
    )
    site_map.add_child(marker)

site_map
```

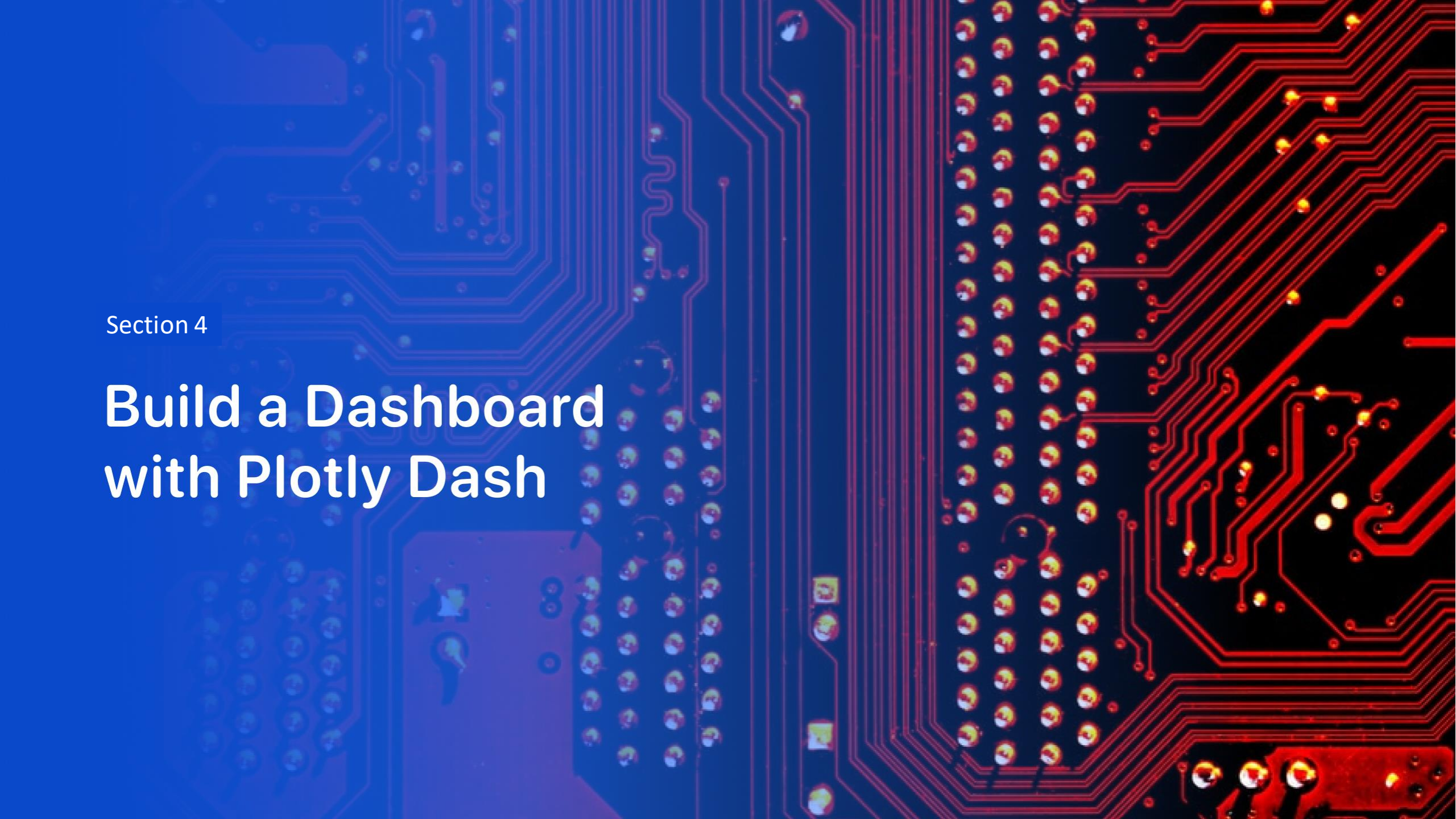


Launch Sites Distance to Landmarks

Note: Couldn't calculate the distance to Landmarks.

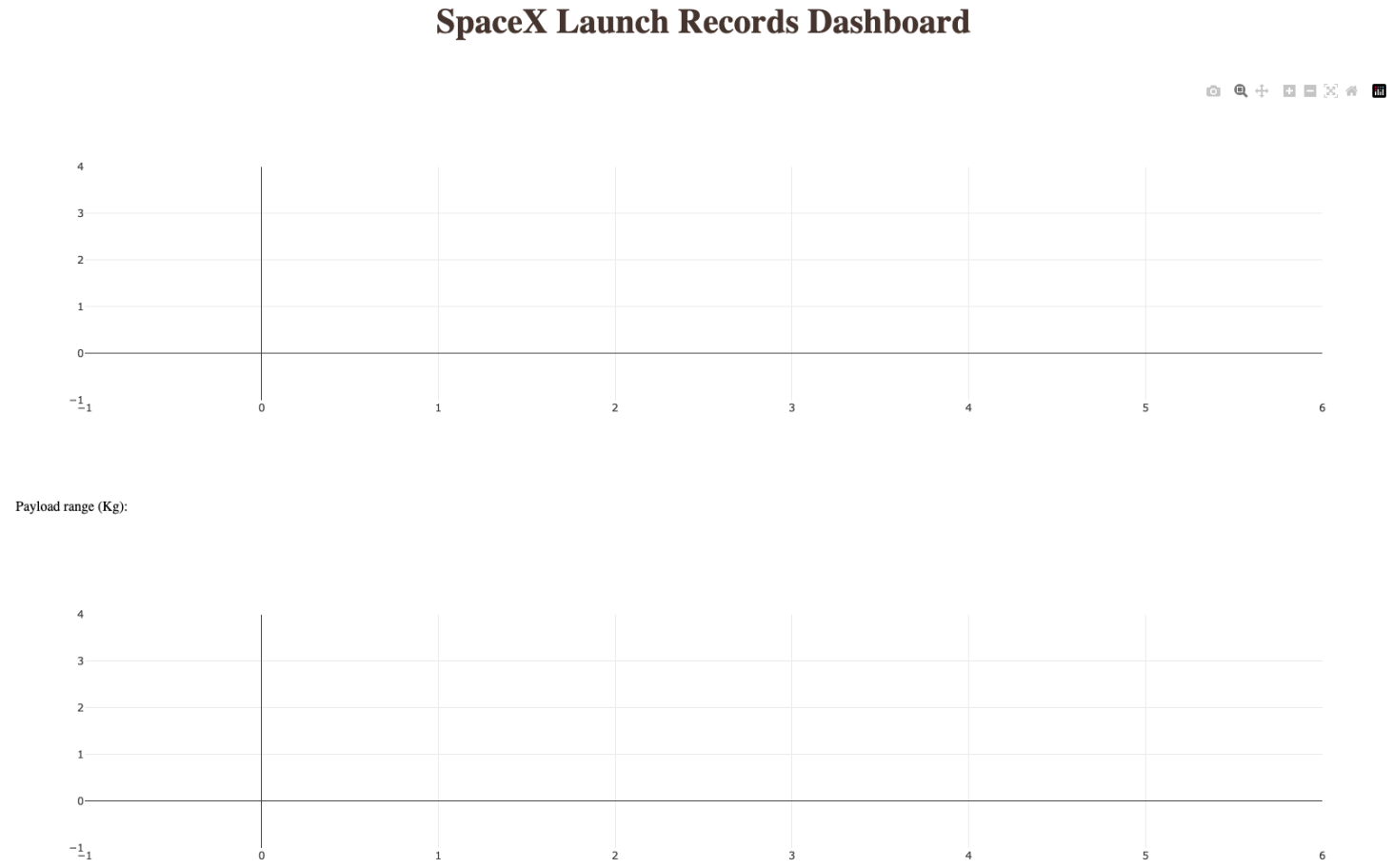
```
# Add mouse position to get the coordinate (Lat, Long) for a mouse over on the map
formatter = "function(num) {return L.Util.formatNum(num, 5);};"
mouse_position = MousePosition(
    position='topright',
    separator=' Long: ',
    empty_string='NaN',
    lng_first=False,
    num_digits=20,
    prefix='Lat:',
    lat_formatter=formatter,
    lng_formatter=formatter,
)

site_map.add_child(mouse_position)
site_map
```

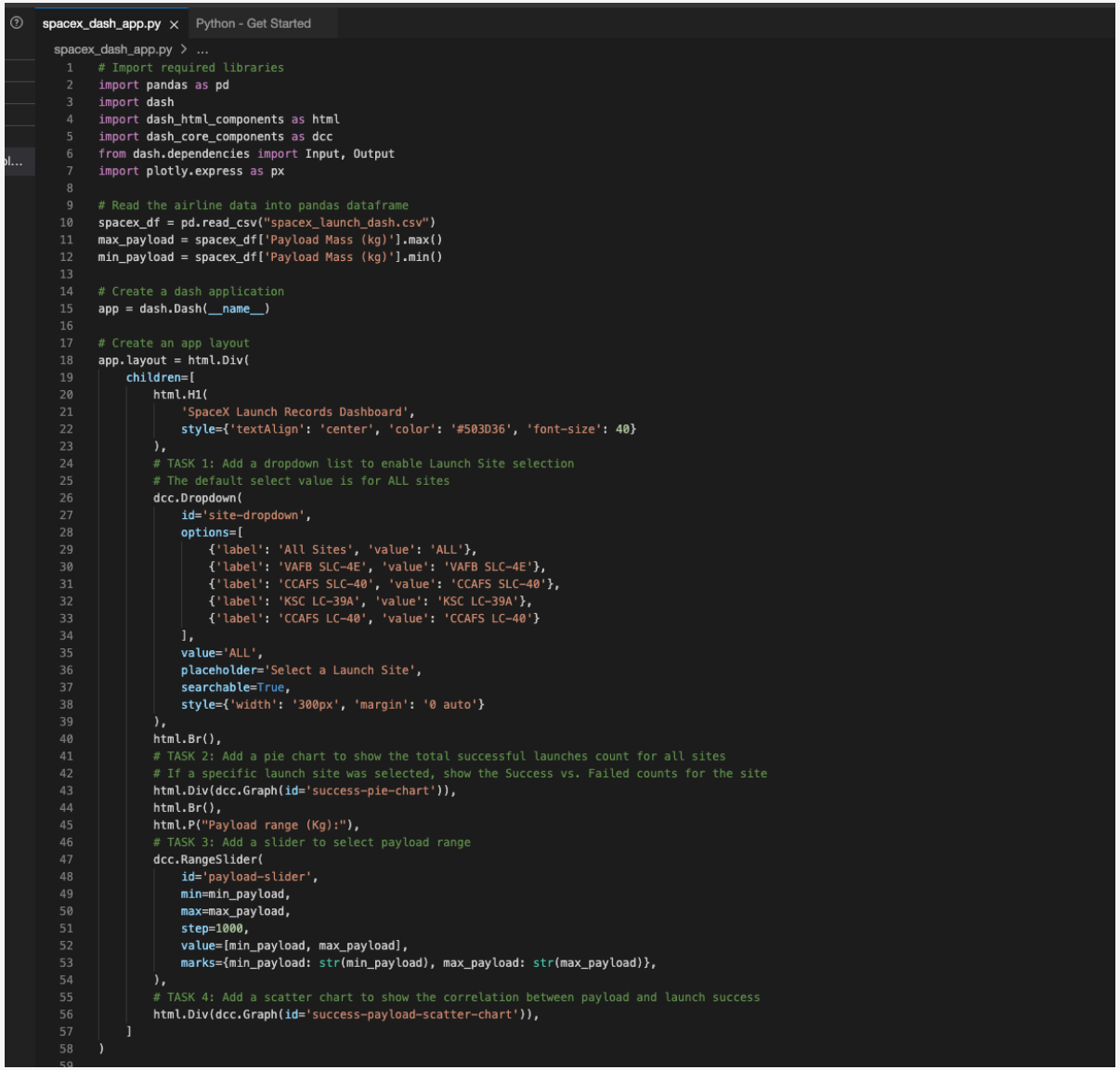



Section 4
Build a Dashboard with Plotly Dash

SpaceX Launch Records Dashboard

NOTE: I have written all the codes according to the instructions, but the application is not opening for some reason. I will attach a screenshot of the code in the next slides.





```python
# Import required libraries
import pandas as pd
import dash
import dash_html_components as html
import dash_core_components as dcc
from dash.dependencies import Input, Output
import plotly.express as px

# Read the airline data into pandas dataframe
spacex_df = pd.read_csv("spacex_launch_dash.csv")
max_payload = spacex_df['Payload Mass (kg)'].max()
min_payload = spacex_df['Payload Mass (kg)'].min()

# Create a dash application
app = dash.Dash(__name__)

# Create an app layout
app.layout = html.Div(
    children=[
        html.H1(
            'SpaceX Launch Records Dashboard',
            style={'textAlign': 'center', 'color': '#503D36', 'font-size': 40}
        ),
        # TASK 1: Add a dropdown list to enable Launch Site selection
        # The default select value is for ALL sites
        dcc.Dropdown(
            id='site-dropdown',
            options=[
                {'label': 'All Sites', 'value': 'ALL'},
                {'label': 'VAFB SLC-4E', 'value': 'VAFB SLC-4E'},
                {'label': 'CCAFS SLC-40', 'value': 'CCAFS SLC-40'},
                {'label': 'KSC LC-39A', 'value': 'KSC LC-39A'},
                {'label': 'CCAFS LC-40', 'value': 'CCAFS LC-40'}
            ],
            value='ALL',
            placeholder='Select a Launch Site',
            searchable=True,
            style={'width': '300px', 'margin': '0 auto'}
        ),
        html.Br(),
        # TASK 2: Add a pie chart to show the total successful launches count for all sites
        # If a specific launch site was selected, show the Success vs. Failed counts for the site
        html.Div(dcc.Graph(id='success-pie-chart')),
        html.Br(),
        html.P("Payload range (Kg):"),
        # TASK 3: Add a slider to select payload range
        dcc.RangeSlider(
            id='payload-slider',
            min=min_payload,
            max=max_payload,
            step=1000,
            value=[min_payload, max_payload],
            marks={min_payload: str(min_payload), max_payload: str(max_payload)},
        ),
        # TASK 4: Add a scatter chart to show the correlation between payload and launch success
        html.Div(dcc.Graph(id='success-payload-scatter-chart')),
    ]
)
```

Dashboard Code Snippet 1

- Replace <Dashboard screenshot 2> title with an appropriate title
- Show the screenshot of the piechart for the launch site with highest launch success ratio
- Explain the important elements and findings on the screenshot

Dashboard Code Snippet 2

- Replace <Dashboard screenshot 3> title with an appropriate title
- Show screenshots of Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider
- Explain the important elements and findings on the screenshot, such as which payload range or booster version have the largest success rate, etc.

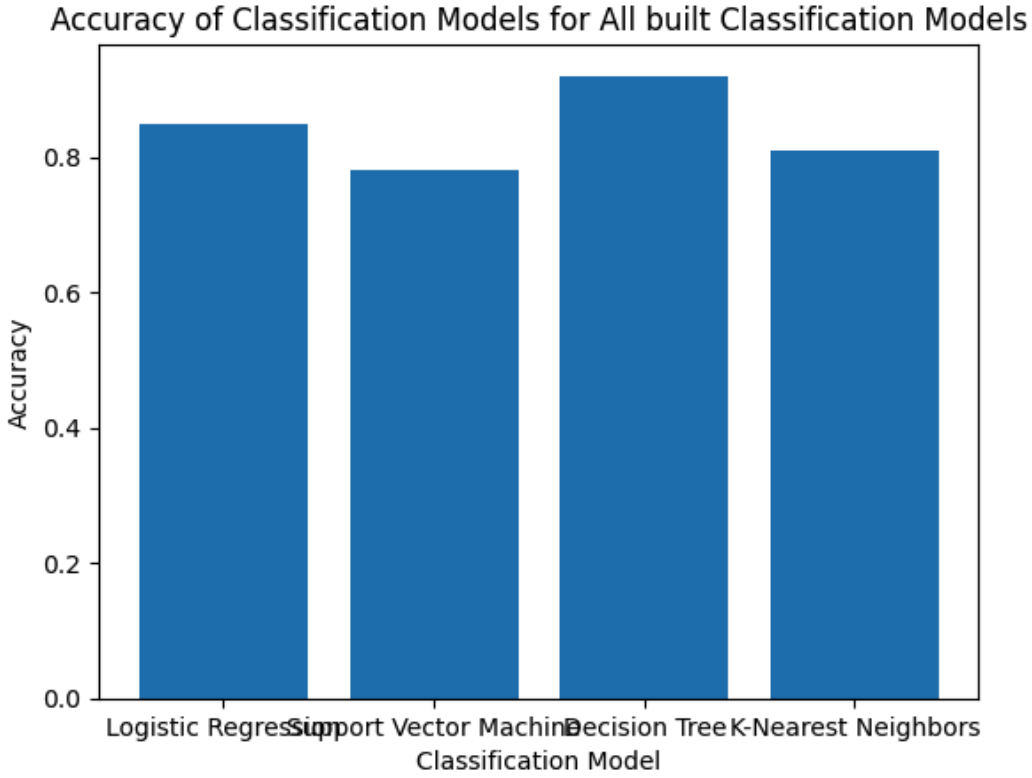
Section 5

Predictive Analysis (Classification)

Classification Accuracy

```
accuracy_scores = {
    'Support Vector Machine': svm_accuracy,
    'Decision Tree': tree_accuracy,
    'K-Nearest Neighbors': knn_accuracy
}

# Find the method with the highest accuracy score
best_method = max(accuracy_scores, key=accuracy_scores.get)
best_accuracy = accuracy_scores[best_method]

print("Best performing method:", best_method)
print("Accuracy score:", best_accuracy)
```

```
Best performing method: Support Vector Machine
Accuracy score: 0.8333333333333334
```



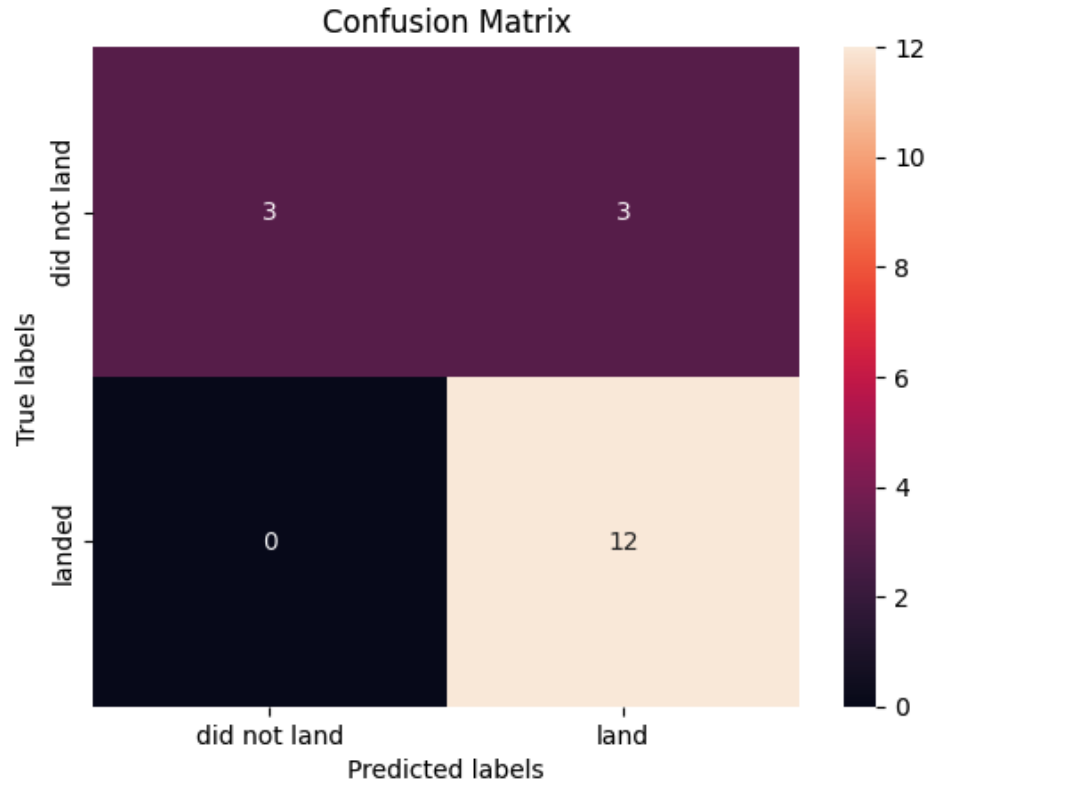
Confusion Matrix

```
svm_accuracy = svm_cv.score(X_test, Y_test)
print("Accuracy on test data:", svm_accuracy)
```

```
Accuracy on test data: 0.8333333333333334
```

We can plot the confusion matrix

```
yhat=svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



Conclusions

1

Successful landing of the first stage: The analysis showed that Falcon 9 has achieved a high success rate in landing its first stage. This is a significant accomplishment as it enables cost savings by reusing the expensive first stage.

2

Key factors for successful landings: The analysis indicated that factors such as launch site, payload mass, and orbit type can influence the success of the first stage landing. Certain launch sites and specific payload masses were found to be associated with higher success rates.

3

Classification models for landing prediction: Various classification models, including Logistic Regression, Support Vector Machine, Decision Tree, and K-Nearest Neighbors, were trained and evaluated to predict the outcome of Falcon 9 landings. The Support Vector Machine model emerged as the best performing model, achieving a high accuracy in predicting successful landings.

4

Potential for cost reduction: The ability to predict the outcome of Falcon 9 landings can have significant cost implications for future launches. By accurately determining whether the first stage will land successfully, the cost of launches can be optimized, leading to potential cost savings and improved operational efficiency.

Appendix

Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!