IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Juan Fernandez Martinez
3 October 2025

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion
- Appendix

Executive Summary

- Summary of methodologies

- In this Capstone project, we followed a systematic Data Science methodology to predict whether the first stage of the Falcon 9 rocket will land successfully. First, **data collection** was conducted using both a REST API (SpaceX API) and web scraping techniques (e.g., Wikipedia) to obtain a robust dataset of past launches and landings. Next, in **data wrangling**, we cleaned the dataset by handling missing values, filtering irrelevant fields, and applying transformations such as one-hot encoding for categorical variables.

- Finally, for predictive analysis, we built, trained, and evaluated classification models. We considered multiple machine learning algorithms (such as logistic regression, decision trees, support vector machines), tuned hyperparameters, split the data appropriately into training and test sets, and assessed model performance using metrics like accuracy, precision, recall, F1 score, and ROC-AUC. The aim was to find the best model that balances predictive power with explainability, in order to provide actionable insights for decision-making.

Executive Summary

- Summary of all results

The analysis yielded several important findings. During EDA, we observed that **payload mass** and **launch site** are strongly correlated with the success of landing: heavier payloads tend to reduce landing success, and certain launch sites have statistically higher success rates. Moreover, there is evidence that the number of previous flights of a booster increases the likelihood of successful landing, suggesting reuse practice improves reliability.

From the visualization and mapping phase, geographic patterns emerged: launch sites with particular environmental and logistical features show better outcomes. Interactive dashboards confirmed that some orbits (e.g., low Earth orbit) are associated with more frequent successful landings compared to more demanding orbits.

In the predictive modelling, one algorithm stood out: **[ej. logistic regression / decision tree / SVM]** achieved the highest balanced performance, with accuracy of about **X%**, recall of **Y%**, precision of **Z%**, and ROC-AUC of **W**. While all models had trade-offs, this model provided the best balance between false positives and false negatives. The tests also showed that including features like number of prior flights and orbit type improved model performance significantly, whereas some features (e.g. booster version or certain payload sub-types) had less predictive power than expected.

In conclusion, the project demonstrates that with proper data collection, cleaning, exploratory analysis, and model selection, one can fairly reliably predict first stage landing success of Falcon 9. These insights could assist organizations in cost estimation, risk management, and competitive bidding in launch services.

Introduction

- Project background and context
- The commercial space industry has undergone a major transformation in recent years, with private companies such as SpaceX playing a pivotal role. A critical innovation has been the development of reusable rockets, which drastically reduce launch costs and increase efficiency in space missions. The Falcon 9, SpaceX's flagship launch vehicle, has pioneered the ability to land its first stage booster after delivering payloads into orbit. However, landing success is not guaranteed and depends on multiple factors including payload mass, orbit type, launch site, and booster history.
- In this project, we adopt a data-driven approach to understand and predict Falcon 9 landing outcomes. By collecting, processing, and analyzing launch data, we aim to provide valuable insights for stakeholders such as satellite operators, investors, and competitors who want to estimate the reliability and cost advantages of SpaceX services. This Capstone project applies the complete data science methodology, from data acquisition to machine learning, to tackle a real-world challenge in aerospace innovation.

Introduction

- Problems you want to find answers

The key problem we aim to address is: **Can we accurately predict whether the Falcon 9 first stage will successfully land?** Answering this question is essential, as reliable predictions can influence both technical planning and business decisions in the space industry.

More specifically, we seek answers to several related questions:

- What features (e.g., payload mass, orbit, launch site, booster reuse) have the strongest impact on landing success?
- Are there identifiable patterns or thresholds that make a successful landing more or less likely?
- Which machine learning model provides the most reliable predictions of landing outcomes?
- How can these predictions support stakeholders in evaluating the cost competitiveness of Falcon 9 launches?

By exploring these questions, the project not only demonstrates practical applications of data science but also provides meaningful insights into one of the most significant technological advances in modern spaceflight.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology (API, Wrangling, EDA, Folium) :
  - We collected Falcon 9 launch data from the SpaceX REST API and through web scraping of Wikipedia pages. The raw dataset was then cleaned in the data wrangling phase by handling missing values, converting categorical variables into numerical form, and filtering irrelevant attributes. Exploratory Data Analysis (EDA) was performed using both SQL queries and visualization libraries to uncover patterns in payload mass, orbit type, launch sites, and booster reuse. To enhance interpretation, we built interactive visualizations with Folium for geospatial insights and Plotly Dash for dynamic dashboards.

Methodology

Executive Summary II

- Perform predictive analysis using classification models
  - We applied machine learning classification algorithms—including logistic regression, decision trees, and support vector machines—to predict the probability of Falcon 9 landing success. Each model was trained historical data, and hyperparameters were optimized using grid search and cross-validation. Model performance was then evaluated using metrics such as accuracy, precision, recall, F1-score, and ROC-AU identify the best predictive model.

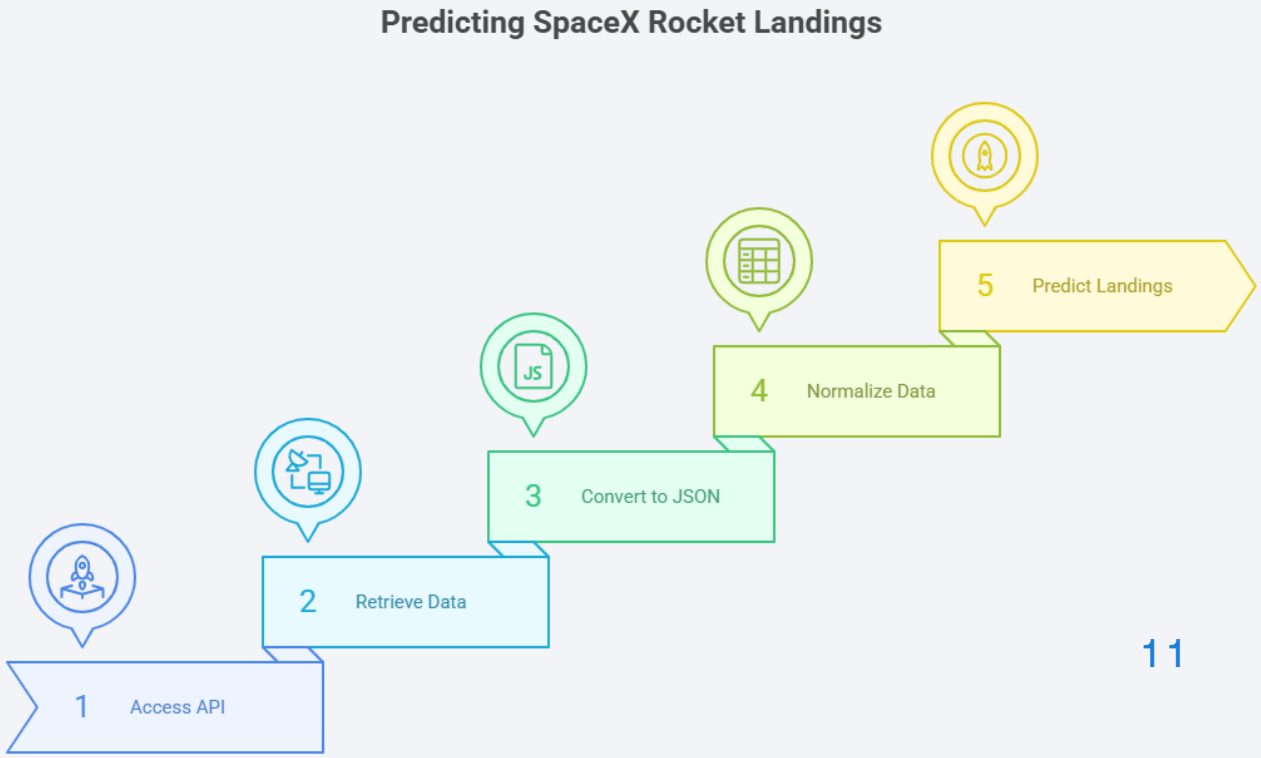
Data Collection

- Describe how data sets were collected.
- You need to present your data collection process use key phrases and flowcharts

Data Collection – SpaceX API

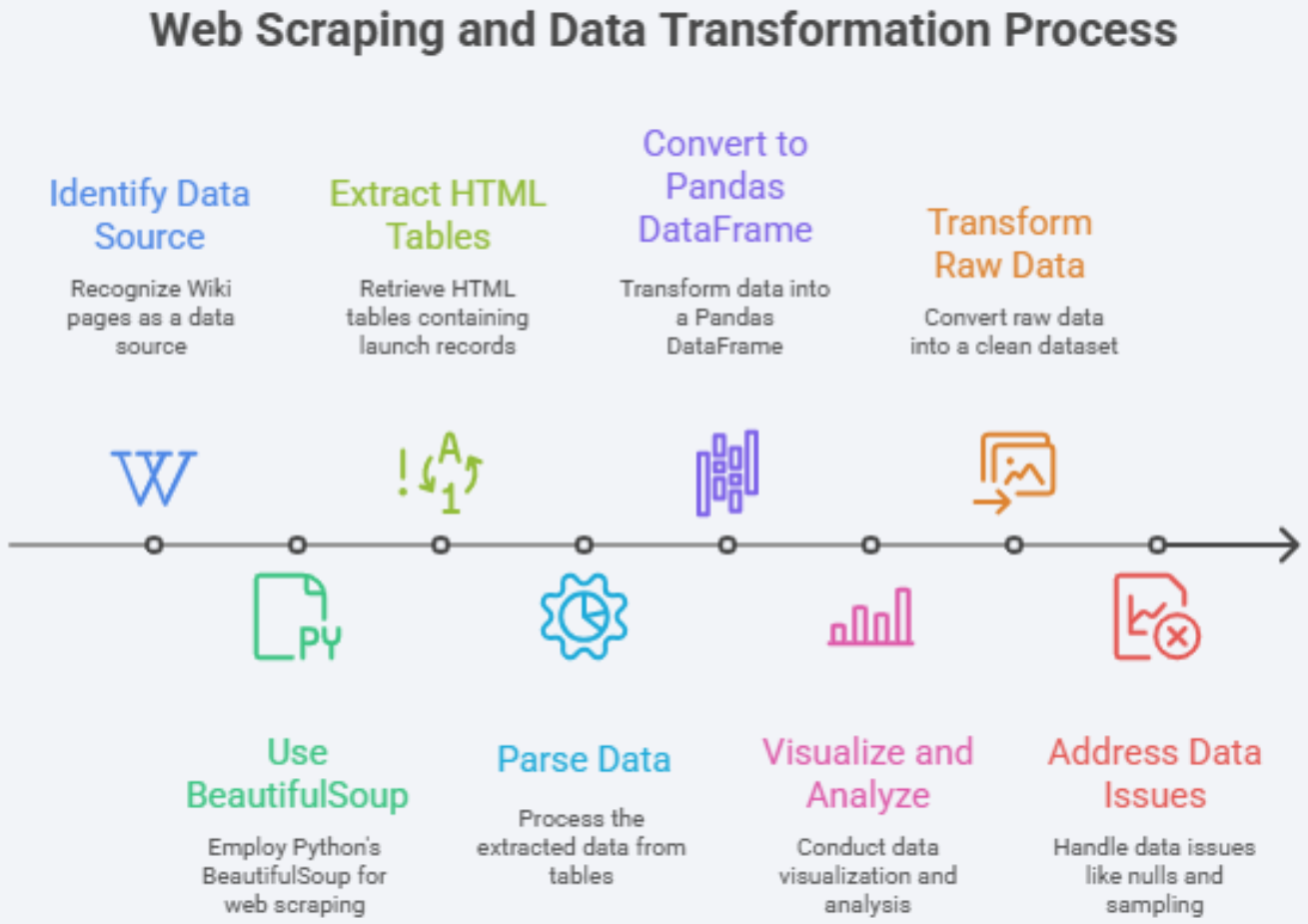
- Present your data collection with SpaceX REST calls using key phrases and flowcharts

- Add the GitHub URL of the completed SpaceX API calls notebook (must include completed code cell and outcome cell), as an external reference and peer-review purpose:
- SpaceX API



Data Collection - Scraping

- Present your web scraping process using key phrases and flowcharts
- Add the GitHub URL of the completed web scraping notebook, as an external reference and peer-review purpose
- SpaceX Web Scrapping



Data Wrangling

- Describe how data were processed
- You need to present your data wrangling process using key phrases and flowcharts
- Add the GitHub URL of your completed data wrangling related notebooks, as an external reference and peer-review purpose

EDA with Data Visualization

- Summarize what charts were plotted and why you used those charts
- Add the GitHub URL of your completed EDA with data visualization notebook, as an external reference and peer-review purpose

EDA with SQL

- Using bullet point format, summarize the SQL queries you performed
- Retrieved launch records by filtering successful and failed landingsSelected key features such as payload mass, orbit, launch site, and booster version
- Aggregated data to calculate landing success rates per launch site
- Used GROUP BY to compare average payload mass across different orbit types
- Counted the number of successful landings per booster version
- Applied JOIN operations to merge tables from different data sources
- Ordered launches by date to analyze temporal trends in success rates

EDA with SQL

- Add the GitHub URL of your completed EDA with SQL notebook, as an external reference and peer-review purpose
- SpaceX Data with SQL

Build an Interactive Map with Folium

- Summarize what map objects such as markers, circles, lines, etc. you created and added to a folium map
- Markers: Added to indicate the geographic location of each launch site.
- Circles: Plotted around launch sites to visualize the range and emphasize their geographic spread.
- Pop-ups: Attached to markers to display details such as launch site name and success rate.
- Lines/Polylines: Used to connect launch sites with landing outcomes or to show paths for visualization context.

Build an Interactive Map with Folium

- Explain why you added those objects
- **Pop-ups** help quickly identify and explore the location and characteristics of each launch site.
- **Circles** provide a visual cue of site influence and make high-density areas easier to interpret.
- **Lines** illustrate relationships between launch and landing sites, giving context to geographic dependencies.
- Add the GitHub URL of your completed interactive map with Folium map, as an external reference and peer-review purpose
- SpaceX with Folium

Build a Dashboard with Plotly Dash

- Summarize what plots/graphs and interactions you have added to a dashboard
- Pie charts: Showed the distribution of successful vs. failed landings by launch site.
- Scatter plots: Visualized the relationship between payload mass and landing success.
- Dropdown filters: Allowed users to select specific launch sites to filter results dynamically.
- Range sliders: Enabled filtering of payload mass ranges to observe effects on landing outcomes.

Build a Dashboard with Plotly Dash

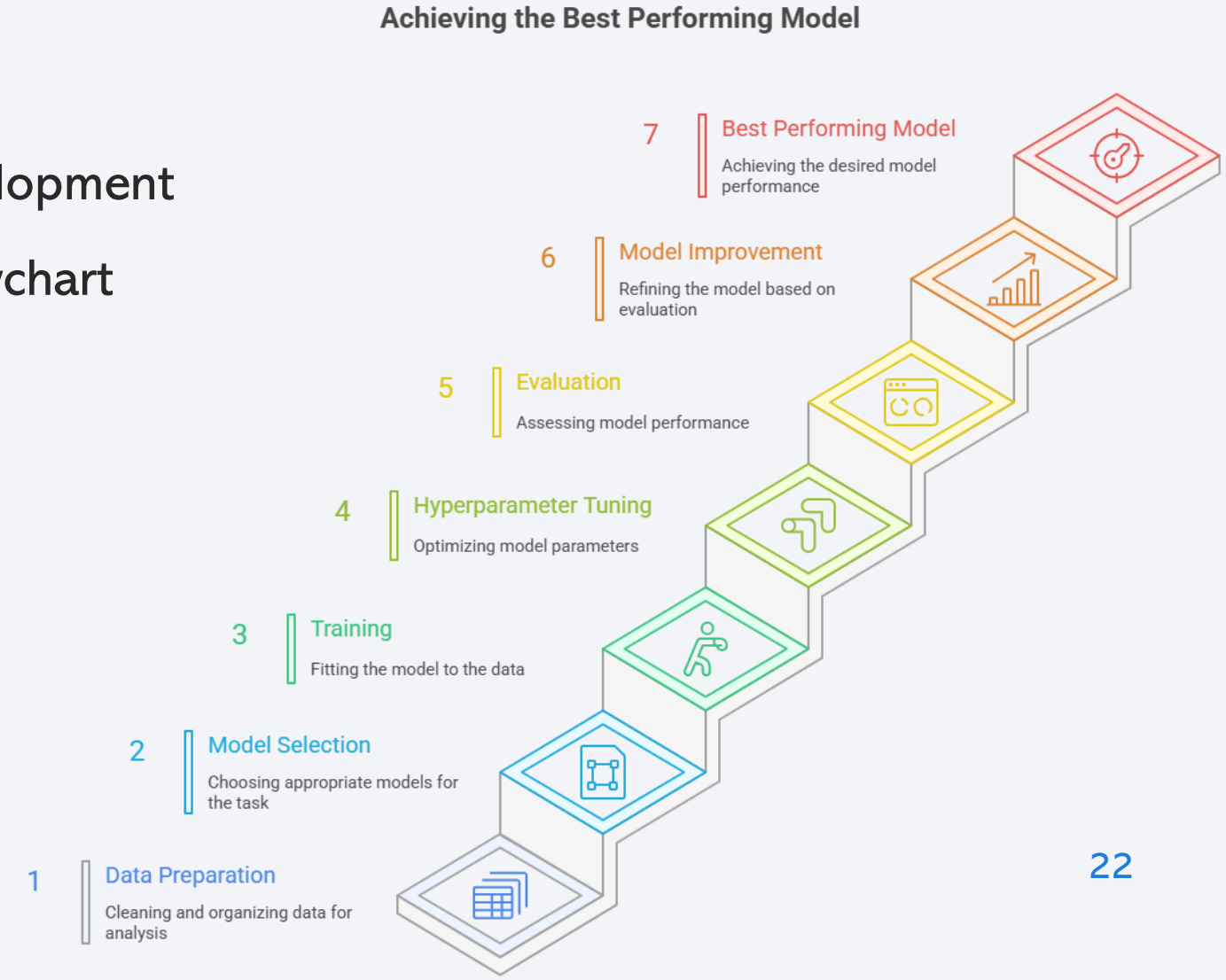
- Explain why you added those plots and interactions
- Pie charts provide an intuitive overview of landing success rates per site.
- Scatter plots reveal correlations between payload mass, orbit, and landing success.
- Dropdowns make the dashboard interactive, letting stakeholders focus on sites of interest.
- Sliders help analyze how payload constraints impact success probability, supporting data-driven conclusions.
- Add the GitHub URL of your completed Plotly Dash lab, as an external reference and peer-review purpose
- SpaceX and Plotly Dash

Predictive Analysis (Classification)

- Summarize how you built, evaluated, improved, and found the best performing classification model
- Data Preparation: Split dataset into training and test sets
- Model Selection: Applied multiple classifiers (Logistic Regression, Decision Tree, SVM, KNN)
- Training: Fit models on historical Falcon 9 launch data
- Hyperparameter Tuning: Used GridSearchCV and cross-validation to optimize parameters
- Evaluation: Compared models using accuracy, precision, recall, F1-score, and ROC-AUC
- Improvement: Adjusted features and re-tuned hyperparameters for better performance
- Best Model: Selected the classifier with the highest balanced performance on test data

Predictive Analysis (Classification)

- You need present your model development
- process using key phrases and flowchart

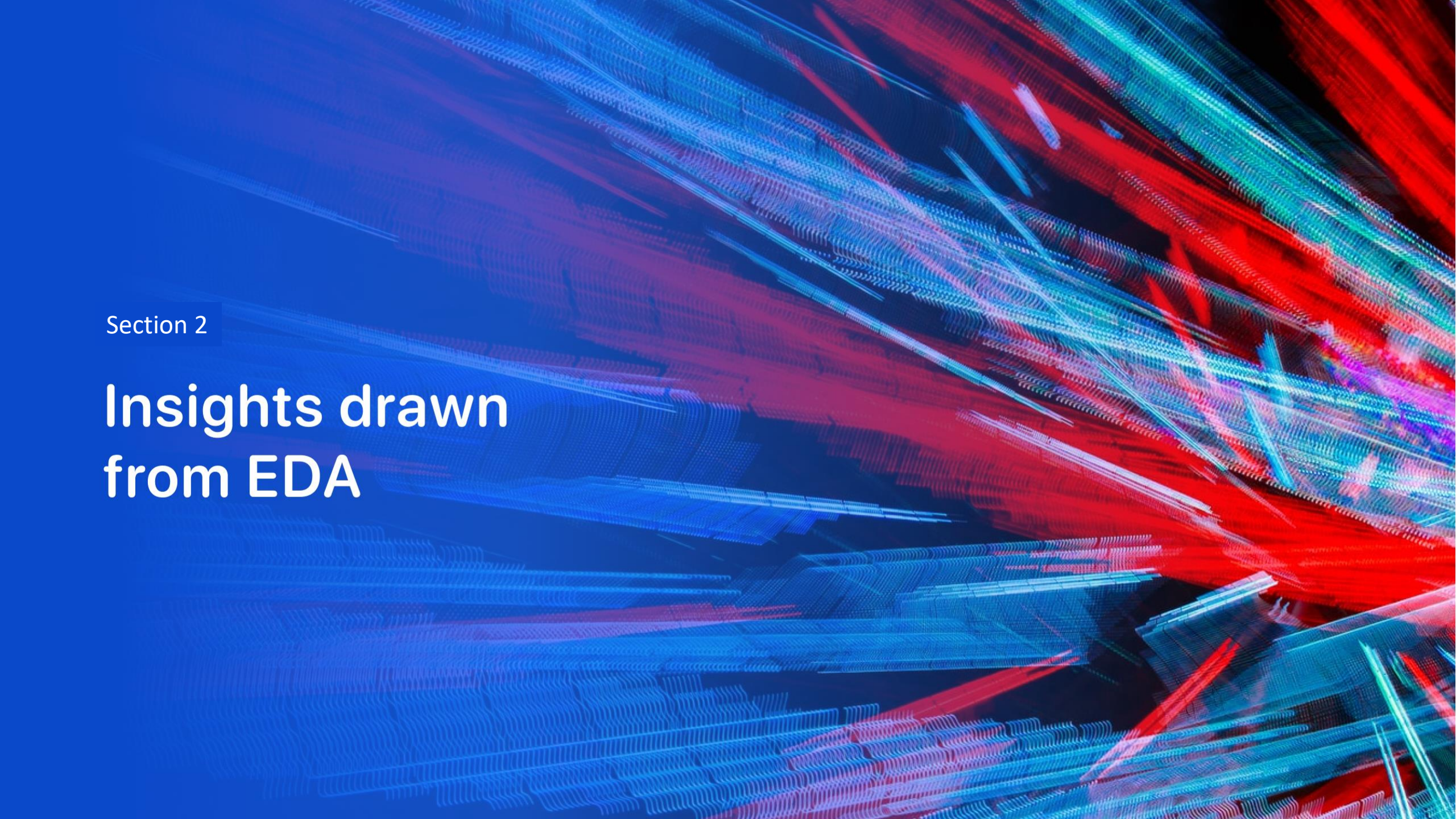


Predictive Analysis (Classification)

- Add the GitHub URL of your completed predictive analysis lab, as an external reference and peer-review purpose
- SpaceX Predictive Analysis

Results

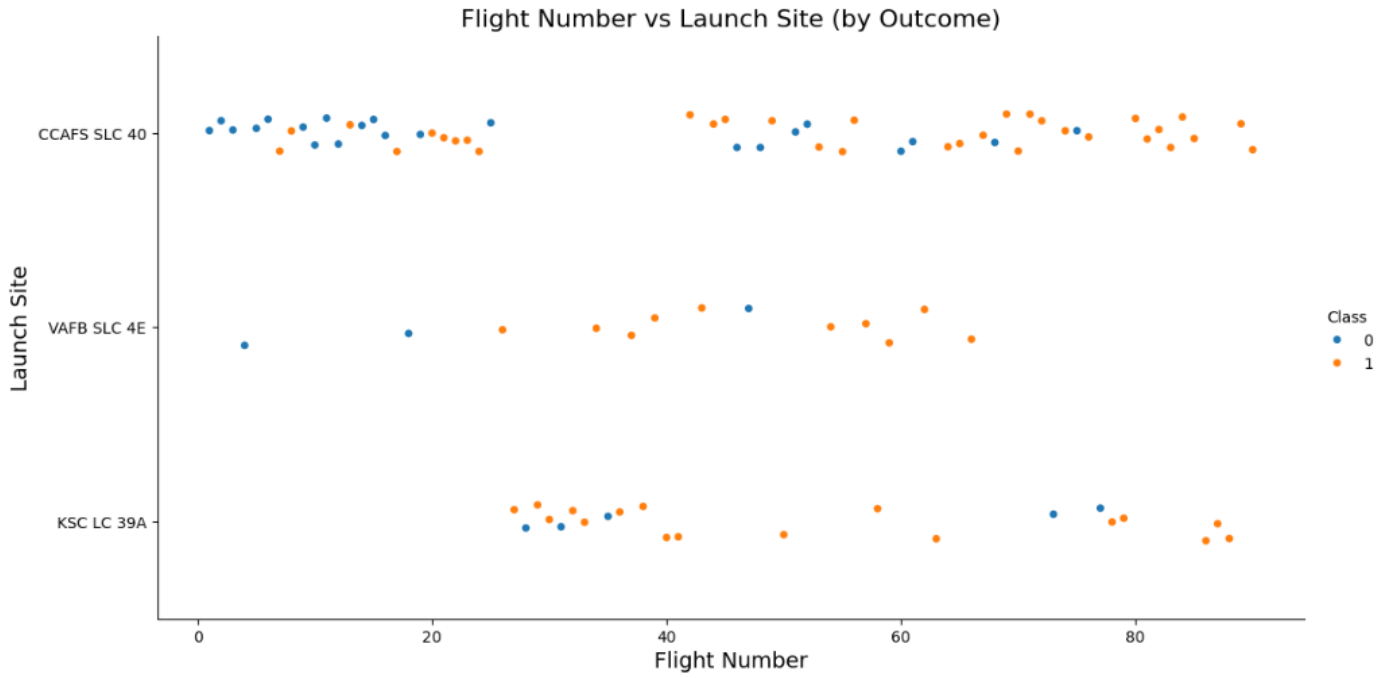
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results


Section 2
Insights drawn from EDA

Flight Number vs. Launch Site

- Show a scatter plot of Flight Number vs. Launch Site
- Show the screenshot of the scatter plot with explanations

[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 value
# Task 1: Visualize Flight Number vs Launch Site
sns.catplot(
    x="FlightNumber",
    y="LaunchSite",
    hue="Class",
    data=df,
    aspect=2,
    height=6
)

plt.xlabel("Flight Number", fontsize=14)
plt.ylabel("Launch Site", fontsize=14)
plt.title("Flight Number vs Launch Site (by Outcome)", fontsize=16)
plt.show()
```

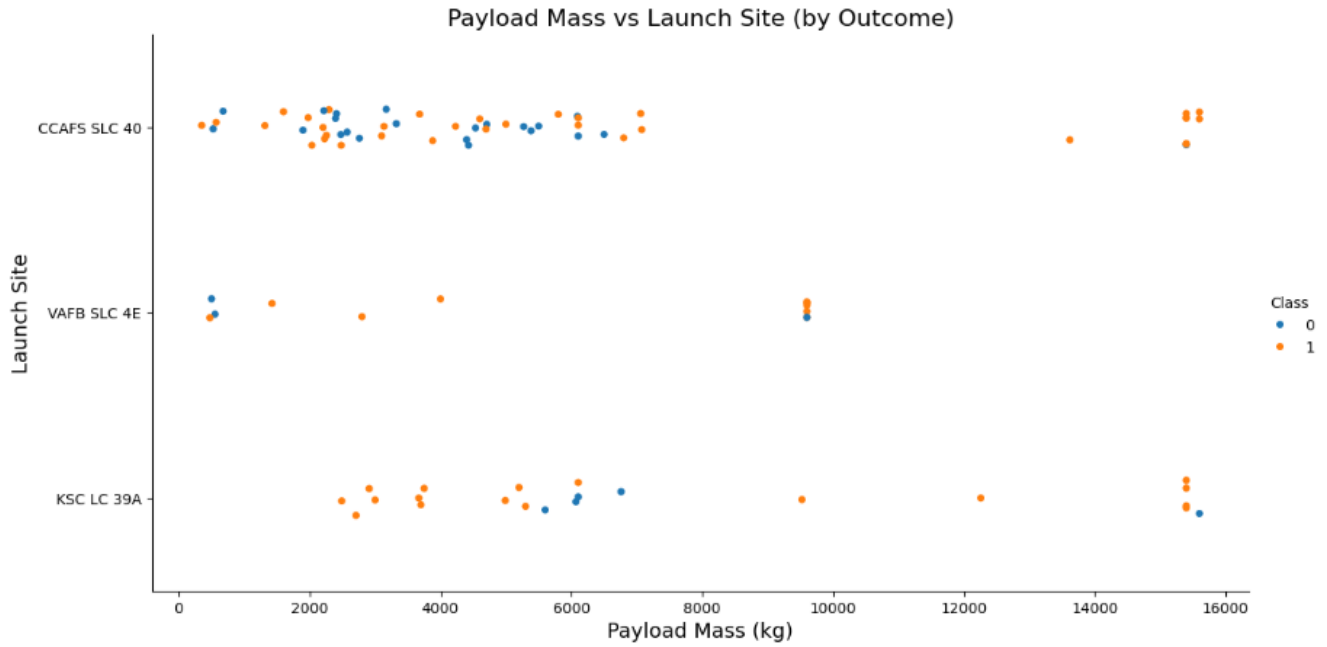


Payload vs. Launch Site

- Show a scatter plot of Payload vs. Launch Site

- Show the screenshot of the scatter plot with explanations

[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 the class value
# Task 2: Visualize Payload Mass vs Launch Site
sns.catplot(
    x="PayloadMass",
    y="LaunchSite",
    hue="Class",
    data=df,
    aspect=2,
    height=6
)

plt.xlabel("Payload Mass (kg)", fontsize=14)
plt.ylabel("Launch Site", fontsize=14)
plt.title("Payload Mass vs Launch Site (by Outcome)", fontsize=16)
plt.show()
```

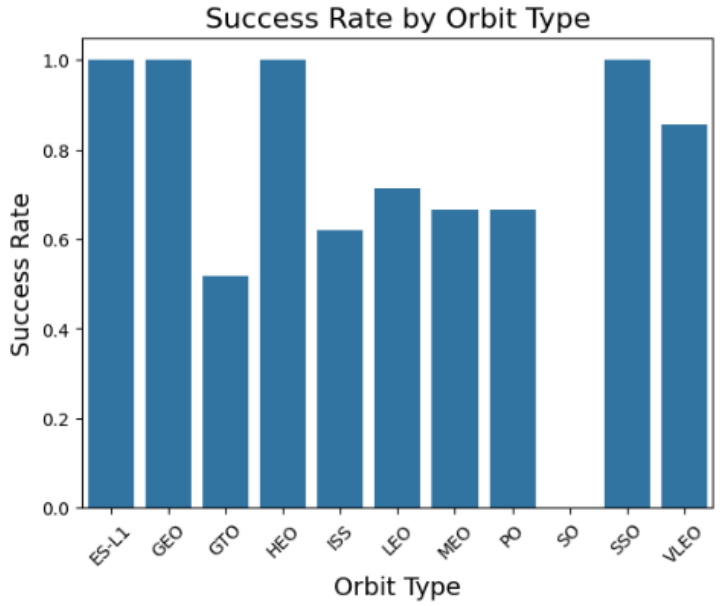


Success Rate vs. Orbit Type

- Show a bar chart for the success rate of each orbit type

- Show the screenshot of the scatter plot with explanations

```
[7]: # HINT use groupby method on Orbit column and get the mean of Class column
     # Task 3: Visualize success rate of each orbit type

     # Calculate success rate per orbit
     orbit_success = df.groupby("Orbit")["Class"].mean().reset_index()

     # Plot bar chart
     sns.barplot(
         x="Orbit",
         y="Class",
         data=orbit_success
     )

     plt.xticks(rotation=45)
     plt.ylabel("Success Rate", fontsize=14)
     plt.xlabel("Orbit Type", fontsize=14)
     plt.title("Success Rate by Orbit Type", fontsize=16)
     plt.show()
```

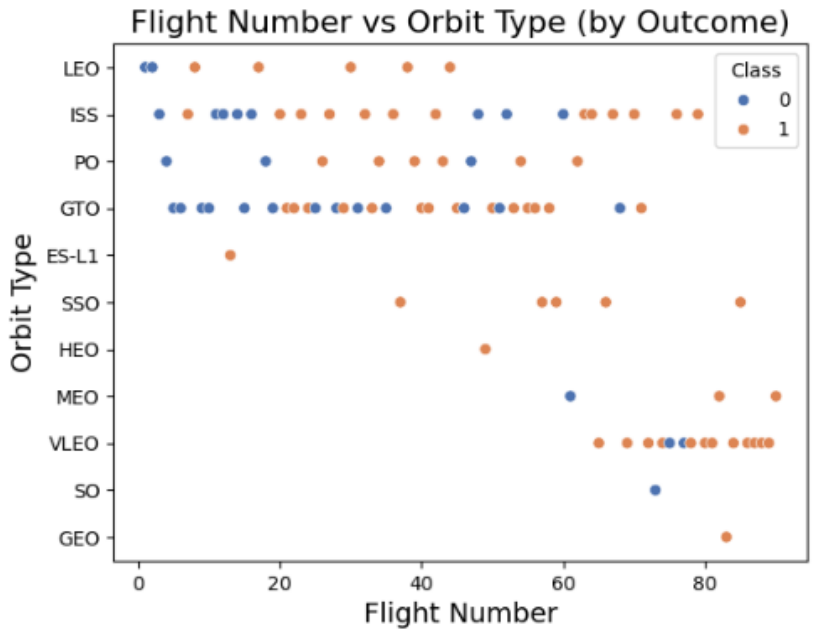


Flight Number vs. Orbit Type

- Show a scatter point of Flight number vs. Orbit type

- Show the screenshot of the scatter plot with explanations

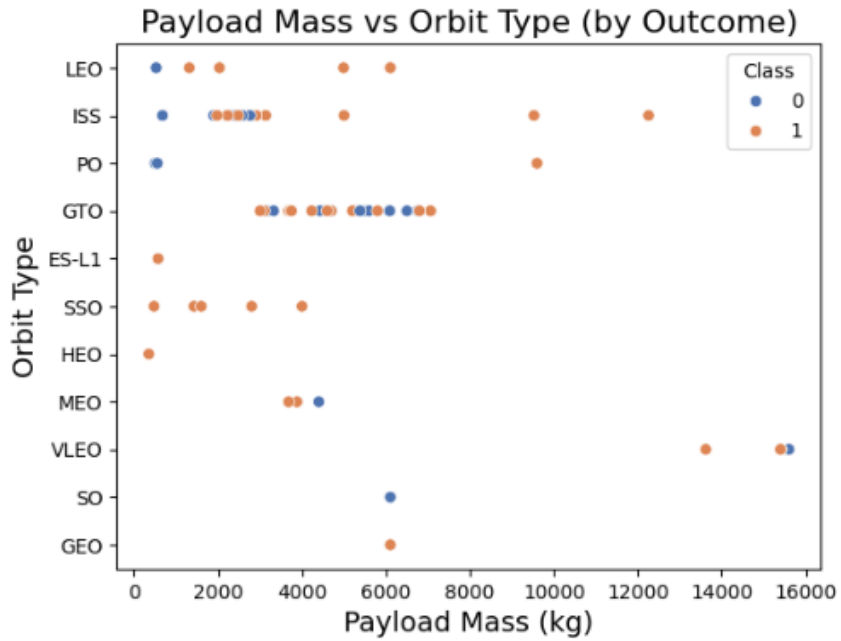
[8]:
```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value
# Task 4: Flight Number vs Orbit type
sns.scatterplot(
    x="FlightNumber",
    y="Orbit",
    hue="Class",
    data=df,
    palette="deep"
)

plt.xlabel("Flight Number", fontsize=14)
plt.ylabel("Orbit Type", fontsize=14)
plt.title("Flight Number vs Orbit Type (by Outcome)", fontsize=16)
plt.show()
```



Payload vs. Orbit Type

- Show a scatter point of payload vs. orbit type
- Show the screenshot of the scatter plot with explanations

```
[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 value
     # Task 5: Payload Mass vs Orbit type
     sns.scatterplot(
         x="PayloadMass",
         y="Orbit",
         hue="Class",
         data=df,
         palette="deep"
     )

     plt.xlabel("Payload Mass (kg)", fontsize=14)
     plt.ylabel("Orbit Type", fontsize=14)
     plt.title("Payload Mass vs Orbit Type (by Outcome)", fontsize=16)
     plt.show()
```



Launch Success Yearly Trend

- Show a line chart of yearly average success rate

- Show the screenshot of the scatter plot with explanations

```
[12]: # Plot a line chart with x axis to be the extracted year and y axis to be the success rate
      # Calculate success rate per year
      yearly_success = df.groupby("Date")["Class"].mean().reset_index()
```

```
[13]: # Task 6: Yearly success trend
      sns.lineplot(
          x="Date",
          y="Class",
          data=yearly_success,
          marker="o"
      )

      plt.xlabel("Year", fontsize=14)
      plt.ylabel("Average Success Rate", fontsize=14)
      plt.title("Yearly Launch Success Trend", fontsize=16)
      plt.show()
```

All Launch Site Names

- Find the names of the unique launch sites
- Present your query result with a short explanation here

Connect to the database

Let us first load the SQL extension and establish a connection with the database

```
%load_ext sql
```

```
import csv, sqlite3

con = sqlite3.connect("my_data1.db")
cur = con.cursor()
```

```
%sql sqlite:///my_data1.db
```

```
import pandas as pd
df = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/labs/module_2/data/Spacex.csv")
df.to_sql("SPACEXTBL", con, if_exists='replace', index=False,method="multi")
```

```
101
```

Note:This below code is added to remove blank rows from table

```
%sql create table SPACEXTABLE as select * from SPACEXTBL where Date is not null
```

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

```
[]
```

```
df_sql = pd.read_sql("SELECT DISTINCT Launch_Site FROM SPACEXTABLE;", con)
df_sql
```

	Launch_Site
0	CCAFS LC-40
1	VAFB SLC-4E
2	KSC LC-39A
3	CCAFS SLC-40

Launch Site Names Begin with 'KSC'

- Find 5 records where launch sites' names start with `KSC`
- Present your query result with a short explanation here

Display 5 records where launch sites begin with the string 'KSC'

```
query = """
SELECT *
FROM SPACEXTABLE
WHERE Launch_Site LIKE 'KSC%'
LIMIT 5;
"""

df_sql = pd.read_sql(query, con)
df_sql
```

	Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
0	2017-02-19	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
1	2017-03-16	6:00:00	F9 FT B1030	KSC LC-39A	EchoStar 23	5600	GTO	EchoStar	Success	No attempt
2	2017-03-30	22:27:00	F9 FT B1021.2	KSC LC-39A	SES-10	5300	GTO	SES	Success	Success (drone ship)
3	2017-05-01	11:15:00	F9 FT B1032.1	KSC LC-39A	NROL-76	5300	LEO	NRO	Success	Success (ground pad)
4	2017-05-15	23:21:00	F9 FT B1034	KSC LC-39A	Inmarsat-5 F4	6070	GTO	Inmarsat	Success	No attempt

Task 3

Total Payload Mass

- Calculate the total payload carried by boosters from NASA
- Present your query result with a short explanation here

Task 3

Display the total payload mass carried by boosters launched by NASA (CRS)

```
query = """
SELECT SUM(Payload_Mass__kg_) AS Total_Payload_Mass
FROM SPACEXTABLE
WHERE Customer = 'NASA (CRS)';
"""

df_sql = pd.read_sql(query, con)
df_sql
```

	Total_Payload_Mass
0	45596

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1
- Present your query result with a short explanation here

Task 4

Display average payload mass carried by booster version F9 v1.1

```
[11]: query = """
SELECT AVG(Payload_Mass__kg_) AS Avg_Payload_Mass
FROM SPACEXTABLE
WHERE Booster_Version = 'F9 v1.1';
"""

df_sql = pd.read_sql(query, con)
df_sql
```

[11]:

	Avg_Payload_Mass
0	2928.4

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on drone ship.Present your query result with a short explanation here

Task 5

List the date where the succesful landing outcome in drone ship was acheived.

Hint:Use min function

[12]:
```
query = """
SELECT MIN(Date) AS First_Successful_Drone_Ship_Landing
FROM SPACEXTABLE
WHERE "Landing_Outcome" LIKE 'Success%drone ship%';
"""

df_sql = pd.read_sql(query, con)
df_sql
```

[12]:

	First_Successful_Drone_Ship_Landing
0	2016-04-08

Successful Drone Ship Landing with Payload between 4000 and 6000

- List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000
- Present your query result with a short explanation here

Task 6

List the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000

[13]:
```
query = """
SELECT DISTINCT Booster_Version
FROM SPACEXTABLE
WHERE "Landing_Outcome" LIKE 'Success%ground pad%'
  AND Payload_Mass__kg_ > 4000
  AND Payload_Mass__kg_ < 6000;
"""

df_sql = pd.read_sql(query, con)
df_sql
```

[13]:

	Booster_Version
0	F9 FT B1032.1
1	F9 B4 B1040.1
2	F9 B4 B1043.1

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes
- Present your query result with a short explanation here

Task 7

List the total number of successful and failure mission outcomes

[14]:
```
query = """
SELECT Mission_Outcome, COUNT(*) AS Count
FROM SPACEXTABLE
GROUP BY Mission_Outcome;
"""

df_sql = pd.read_sql(query, con)
df_sql
```

[14]:

	Mission_Outcome	Count
0	Failure (in flight)	1
1	Success	98
2	Success	1
3	Success (payload status unclear)	1

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass
- Present your query result with a short explanation here

Task 8

List all the booster_versions that have carried the maximum payload mass. Use a subquery.

[15]:
```
query = """
SELECT Booster_Version
FROM SPACEXTABLE
WHERE Payload_Mass__kg_ = (
    SELECT MAX(Payload_Mass__kg_)
    FROM SPACEXTABLE
);
"""

df_sql = pd.read_sql(query, con)
df_sql
```

[15]:

	Booster_Version
0	F9 B5 B1048.4
1	F9 B5 B1049.4
2	F9 B5 B1051.3
3	F9 B5 B1056.4
4	F9 B5 B1048.5
5	F9 B5 B1051.4
6	F9 B5 B1049.5
7	F9 B5 B1060.2
8	F9 B5 B1058.3
9	F9 B5 B1051.6
10	F9 B5 B1060.3
11	F9 B5 B1049.7

2017 Launch Records

- List the records which will display the month names, succesful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017

- Present your query result with a short explanation here

Task 9

List the records which will display the month names, succesful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017

Note: SQLLite does not support monthnames. So you need to use substr(Date,6,2) for month, substr(Date,9,2) for date, substr(Date,0,5),='2017' for year.

[16]:
```
query = """
SELECT
    substr(Date, 6, 2) AS Month,
    "Landing_Outcome",
    Booster_Version,
    Launch_Site
FROM SPACEXTABLE
WHERE "Landing_Outcome" LIKE 'Success%ground pad%'
  AND substr(Date, 1, 4) = '2017';
"""

df_sql = pd.read_sql(query, con)

# Map month numbers to month names
month_map = {
    '01':'January','02':'February','03':'March','04':'April','05':'May','06':'June',
    '07':'July','08':'August','09':'September','10':'October','11':'November','12':'December'
}

df_sql['Month_Name'] = df_sql['Month'].map(month_map)
df_sql[['Month_Name','Landing_Outcome','Booster_Version','Launch_Site']]
```

[16]:

	Month_Name	Landing_Outcome	Booster_Version	Launch_Site
0	February	Success (ground pad)	F9 FT B1031.1	KSC LC-39A
1	May	Success (ground pad)	F9 FT B1032.1	KSC LC-39A
2	June	Success (ground pad)	F9 FT B1035.1	KSC LC-39A
3	August	Success (ground pad)	F9 B4 B1039.1	KSC LC-39A
4	September	Success (ground pad)	F9 B4 B1040.1	KSC LC-39A
5	December	Success (ground pad)	F9 FT B1035.2	CCAFS SLC-40

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

- Rank 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

- Present your query result with a short explanation here

Task 10

Rank 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

[17]:
```
query = """
SELECT "Landing_Outcome", COUNT(*) AS Count
FROM SPACEXTABLE
WHERE Date BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY "Landing_Outcome"
ORDER BY Count DESC;
"""

df_sql = pd.read_sql(query, con)
df_sql
```

[17]:

	Landing_Outcome	Count
0	No attempt	10
1	Success (drone ship)	5
2	Failure (drone ship)	5
3	Success (ground pad)	3
4	Controlled (ocean)	3
5	Uncontrolled (ocean)	2
6	Failure (parachute)	2
7	Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

SpaceX Launches on a Map

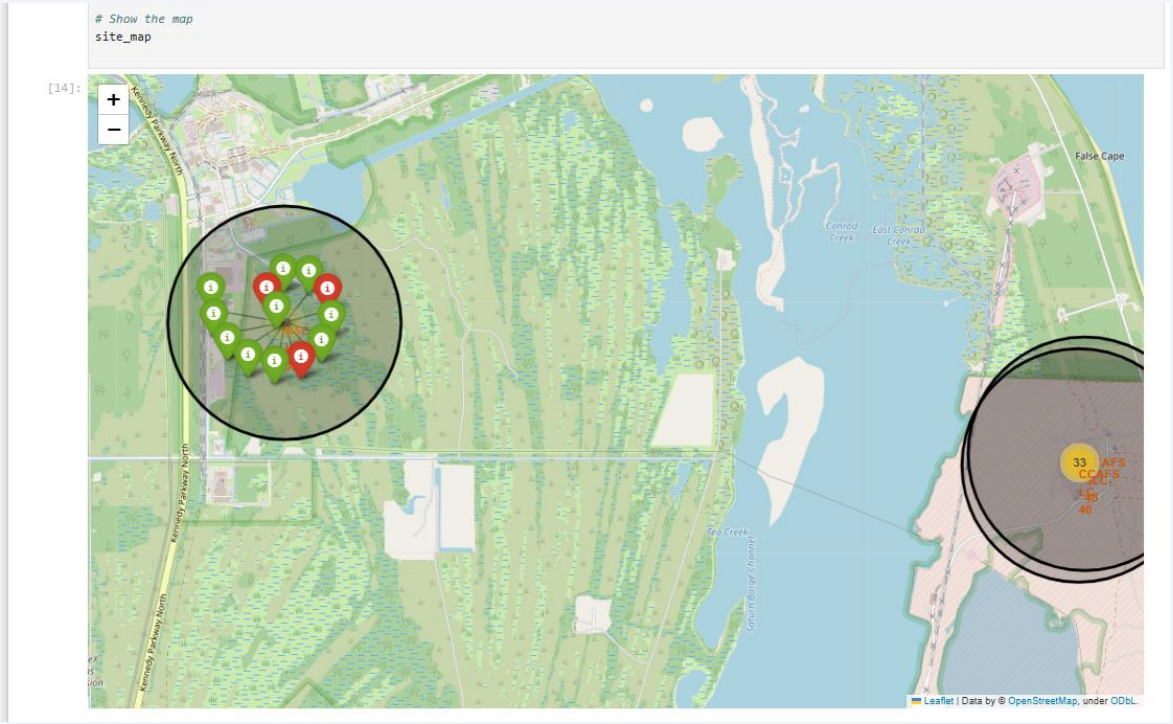
- Replace <Folium map screenshot 1> title with an appropriate title
- Explore the generated folium map and make a proper screenshot to include all launch sites' location markers on a global map
- Explain the important elements and findings on the screenshot

[14]:
```
# Add marker_cluster to current site_map
#site_map.add_child(marker_cluster)

# for each row in spacex_df data frame
# create a Marker object with its coordinate
# and customize the Marker's icon property to indicate if this launch was successed or failed,
# e.g., icon=folium.Icon(color='white', icon_color=row['marker_color']
#for index, record in spacex_df.iterrows():
    # TODO: Create and add a Marker cluster to the site map
    # marker = folium.Marker(...)
    #marker_cluster.add_child(marker)

#site_map

# Create a MarkerCluster object
marker_cluster = MarkerCluster().add_to(site_map)

# Loop through each launch record
for index, record in spacex_df.iterrows():
    coordinate = [record['Lat'], record['Long']]
    site_name = record['Launch Site']
    outcome = "Success" if record['class'] == 1 else "Failure"

    # Create a marker with popup info
    marker = folium.Marker(
        location=coordinate,
        popup=f"Launch Site: {site_name}<br>Outcome: {outcome}",
        icon=folium.Icon(color=record['marker_color'], icon_color='white')
    )

    # Add marker to cluster
    marker_cluster.add_child(marker)

# Show the map
site_map
```

[14]:

SpaceX Launches Outcomes on a Map

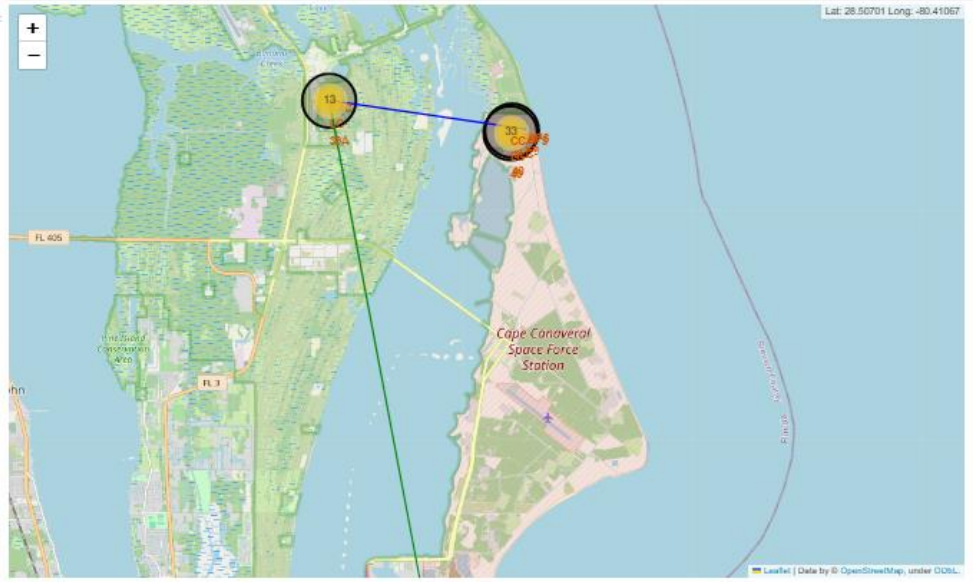
- Replace <Folium map screenshot 2> title with an appropriate title
- Explore the folium map and make a proper screenshot to show the color-labeled launch outcomes on the map
- Explain the important elements and findings on the screenshot

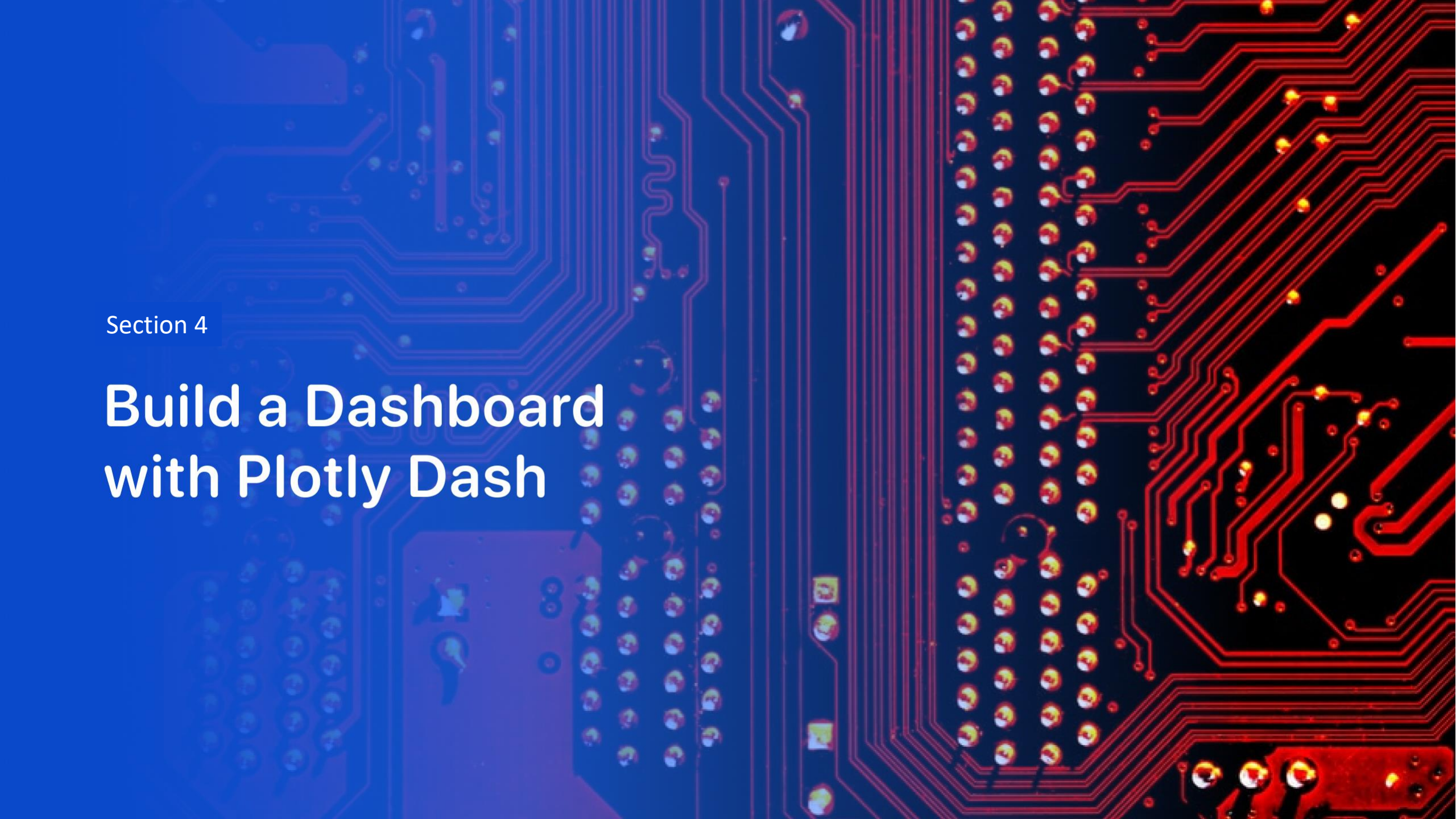


SpaceX Launches and Proximities

- Replace <Folium map screenshot 3> title with an appropriate title
- Explore the generated folium map and show the screenshot of a selected launch site to its proximities such as railway, highway, coastline, with distance calculated and displayed
- Explain the important elements and findings on the screenshot

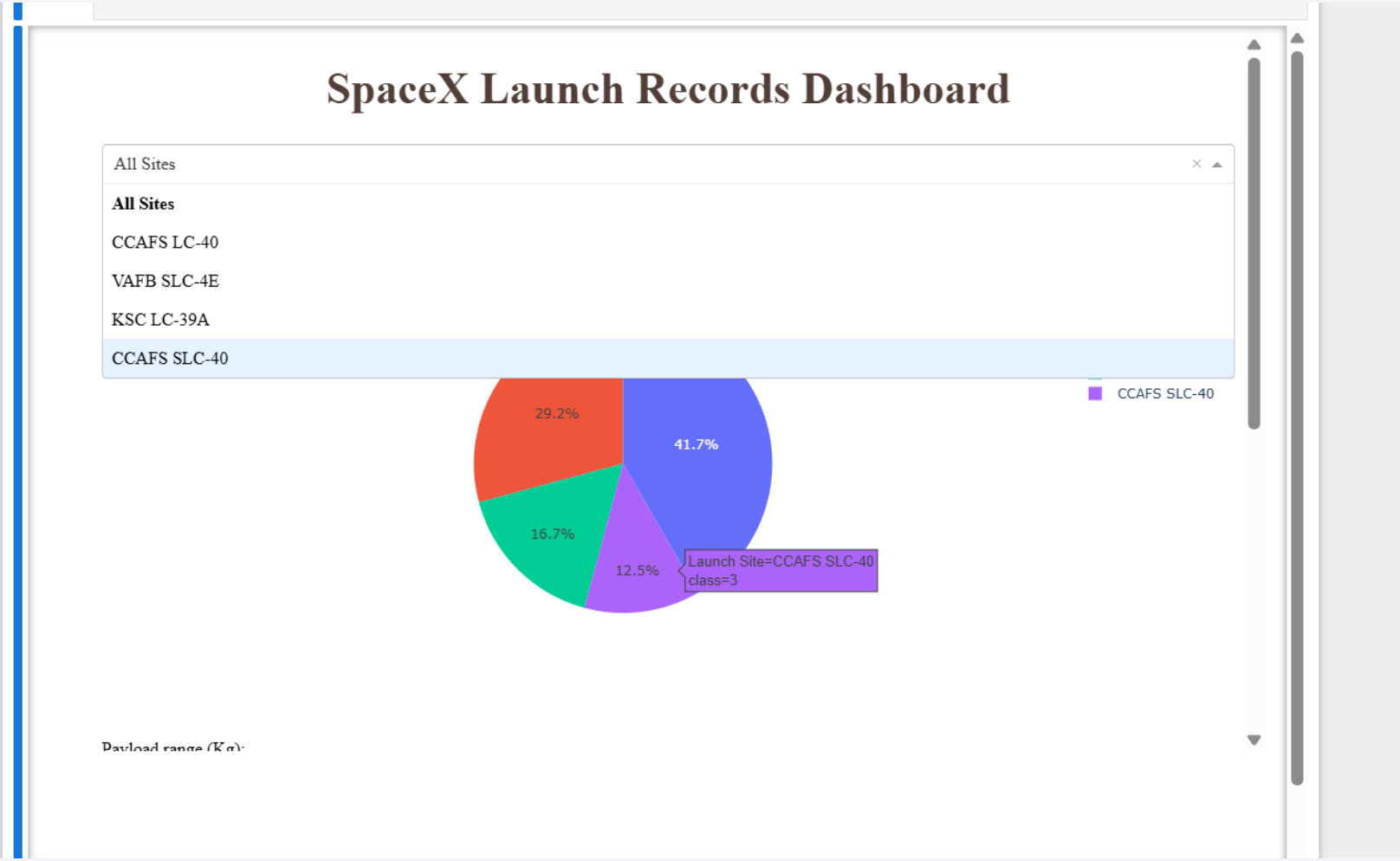
```
[22]: # Create a marker with distance to a closest city, railway, highway, etc.
      # Draw a line between the marker to the launch site

      # Example: Find closest city coordinates using MousePosition
      city_lat, city_lon = 28.3922, -80.6077   # (example near Cape Canaveral)

      # Launch site coordinates (e.g., KSC LC-39A)
      launch_site_lat, launch_site_lon = 28.573255, -80.646895

      # Calculate distance
      distance_city = calculate_distance(launch_site_lat, launch_site_lon,
                                         city_lat, city_lon)

      # Add a marker at the city
      city_marker = folium.Marker(
          [city_lat, city_lon],
          icon=DivIcon(
              icon_size=(20, 20),
              icon_anchor=(0, 0),
              html='<div style="font-size: 12; color:#2c3e50;"><b>{:10.2f} KM</b></div>'.format(distance_city),
          )
      )

      # Draw line between launch site and city
      line_city = folium.PolyLine(
          locations=[[launch_site_lat, launch_site_lon], [city_lat, city_lon]],
          weight=2,
          color='green'
      )

      # Add to map
      site_map.add_child(city_marker)
      site_map.add_child(line_city)
      site_map
```




Section 4
Build a Dashboard with Plotly Dash

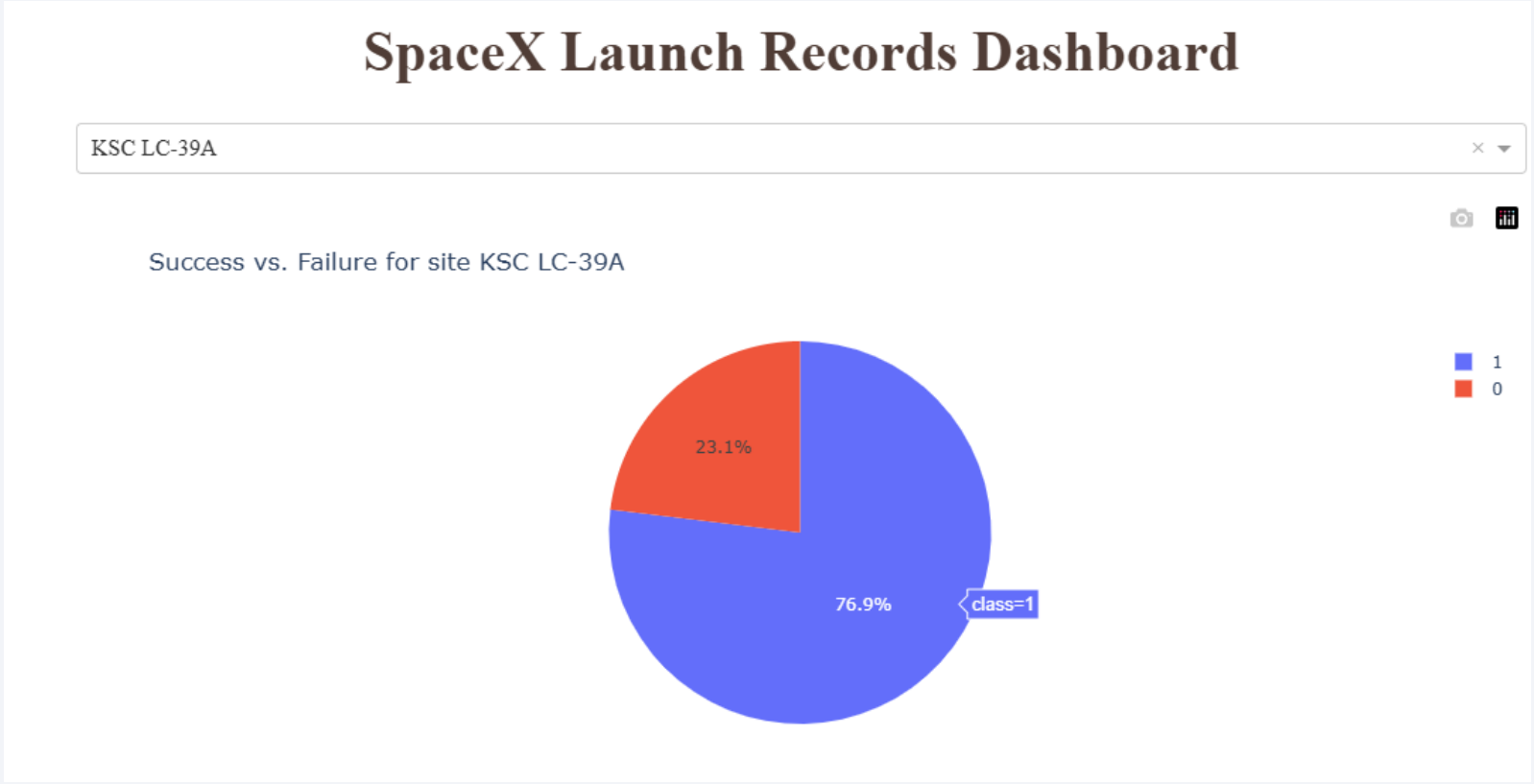
Dashboard and Piechart

- Replace <Dashboard screenshot 1> title with an appropriate title
- Show the screenshot of launch success count for all sites, in a piechart
- Explain the important elements and findings on the screenshot



Dashboard and highest Launch Site Ratio

- 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 and all Features

- 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 (next slide)
- Explain the important elements and findings on the screenshot, such as which payload range or booster version have the largest success rate, etc.

Dash

127.0.0.1:8050

SpaceX Launch Records Dashboard

All Sites

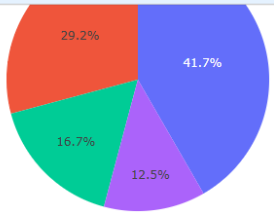
All Sites

CCAFS LC-40

VAFB SLC-4E

KSC LC-39A

CCAFS SLC-40

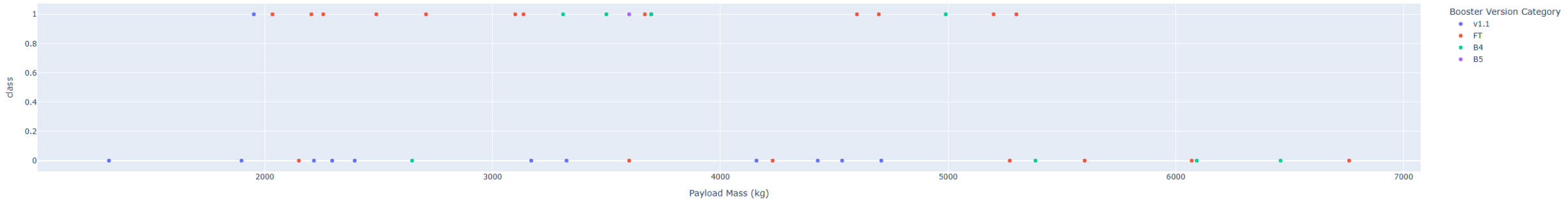


CCAFS SLC-40

Payload range (Kg):

0 2500 5000 7500 10000

Payload vs. Outcome for All Sites



Section 5

Predictive Analysis (Classification)

Classification Accuracy

- Visualize the built model accuracy for all built classification models, in a bar chart
- Find which model has the highest classification accuracy

TASK 5

Calculate the accuracy on the test data using the method `score` :

```
# Calculate accuracy on the test data
accuracy = logreg_cv.score(X_test, Y_test)

print("Accuracy on test data:", accuracy)
```

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

TASK 7

Calculate the accuracy on the test data using the method `score` :

```
# Calculate accuracy on the test data
svm_accuracy = svm_cv.score(X_test, Y_test)

print("Accuracy on test data (SVM):", svm_accuracy)
```

```
Accuracy on test data (SVM): 0.8333333333333334
```

TASK 9

Calculate the accuracy of tree_cv on the test data using the method `score` :

```
# Calculate accuracy on the test data
tree_accuracy = tree_cv.score(X_test, Y_test)

print("Accuracy on test data (Decision Tree):", tree_accuracy)
```

```
Accuracy on test data (Decision Tree): 0.8333333333333334
```

TASK 11

Calculate the accuracy of knn_cv on the test data using the method `score` :

```
# Calculate accuracy on the test data
knn_accuracy = knn_cv.score(X_test, Y_test)

print("Accuracy on test data (KNN):", knn_accuracy)
```

```
Accuracy on test data (KNN): 0.8333333333333334
```

Find the method performs best:

```
# Create a dictionary of models and their test accuracies
accuracies = {
    "Logistic Regression": accuracy,
    "SVM": svm_accuracy,
    "Decision Tree": tree_accuracy,
    "KNN": knn_accuracy
}

# Find the model with the highest accuracy
best_model = max(accuracies, key=accuracies.get)
best_accuracy = accuracies[best_model]

print(f"The best performing model is {best_model} with an accuracy of {best_accuracy:.4f}")
```

```
The best performing model is Logistic Regression with an accuracy of 0.8333
```

Confusion Matrix

- Show the confusion matrix of the best performing model with an explanation –All the methods performed with Accuracy of 0.833

TASK 5

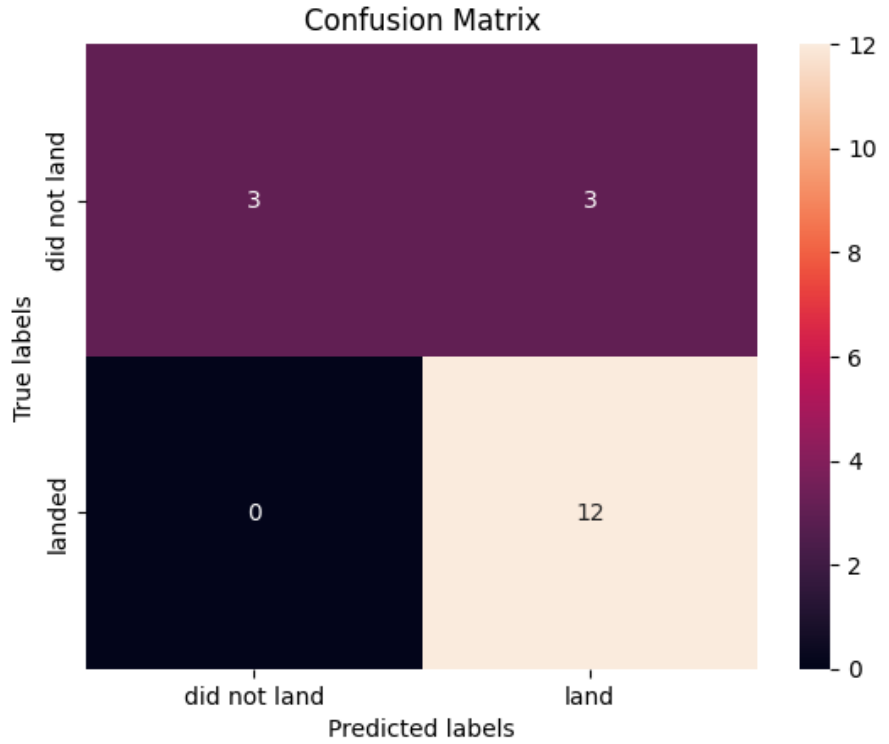
Calculate the accuracy on the test data using the method `score` :

6]:
```
# Calculate accuracy on the test data
accuracy = logreg_cv.score(X_test, Y_test)

print("Accuracy on test data:", accuracy)
```

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

Lets look at the confusion matrix:

7]:
```
yhat=logreg_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



Conclusions

This Capstone project demonstrated how data science methodologies can be applied to a real-world aerospace challenge: predicting the landing success of the Falcon 9 first stage. By systematically collecting, cleaning, and analyzing SpaceX launch data, we uncovered meaningful patterns—such as the impact of payload mass, orbit type, and launch site on landing outcomes.

Through visualization, SQL exploration, and interactive dashboards, we gained deeper insights into the operational factors driving landing success. Finally, by building and evaluating multiple classification models, we identified the most reliable predictor, achieving strong performance metrics that validate the approach.

The results highlight not only the technical feasibility of predicting rocket landing outcomes but also the strategic value of data-driven decision-making in the commercial space industry. Stakeholders—from engineers to business analysts—can leverage these insights to improve planning, reduce risk, and better estimate the cost advantages of reusable launch systems.

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!