IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Shahrul
25WW5

Outline

Executive Summary

- Summary of methodologies
  - Data Collection: Collected data from an API to predict Falcon 9's first-stage landing success.
  - Data Preprocessing: Cleaned and formatted the dataset for analysis.
  - Modeling: Applied classification algorithms such as Decision Trees, K-Nearest Neighbors, and Support Vector Machines (SVM) for prediction.
  - Hyperparameter Tuning: Used GridSearchCV for model optimization and cross-validation to find the best parameters.
  - Evaluation: Assessed model accuracy using test data and determined the best-performing model.
- Summary of all results
  - Best Model: Support Vector Machine (SVM) with the RBF kernel showed the highest accuracy on the validation dataset.
  - Model Accuracy: The SVM model achieved a high accuracy rate in predicting landing success.
  - Model Insights: The prediction model can be used for estimating launch costs and improving competitive bidding strategies.

Introduction

- Project background and context
    - This project predicts the success of the Falcon 9 first stage landing. SpaceX’s ability to reuse the first stage reduces launch costs from $165 million to $62 million, making it a major cost-saving factor. Predicting landing success can help estimate launch costs and provide competitive insights for other companies.
- Problems you want to find answers
    - Will the Falcon 9 first stage land successfully?
    - How does landing success impact launch cost?
    - How can this prediction help other companies compete with SpaceX?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Describe how data was collected
- Perform data wrangling
  - Describe how data was processed
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

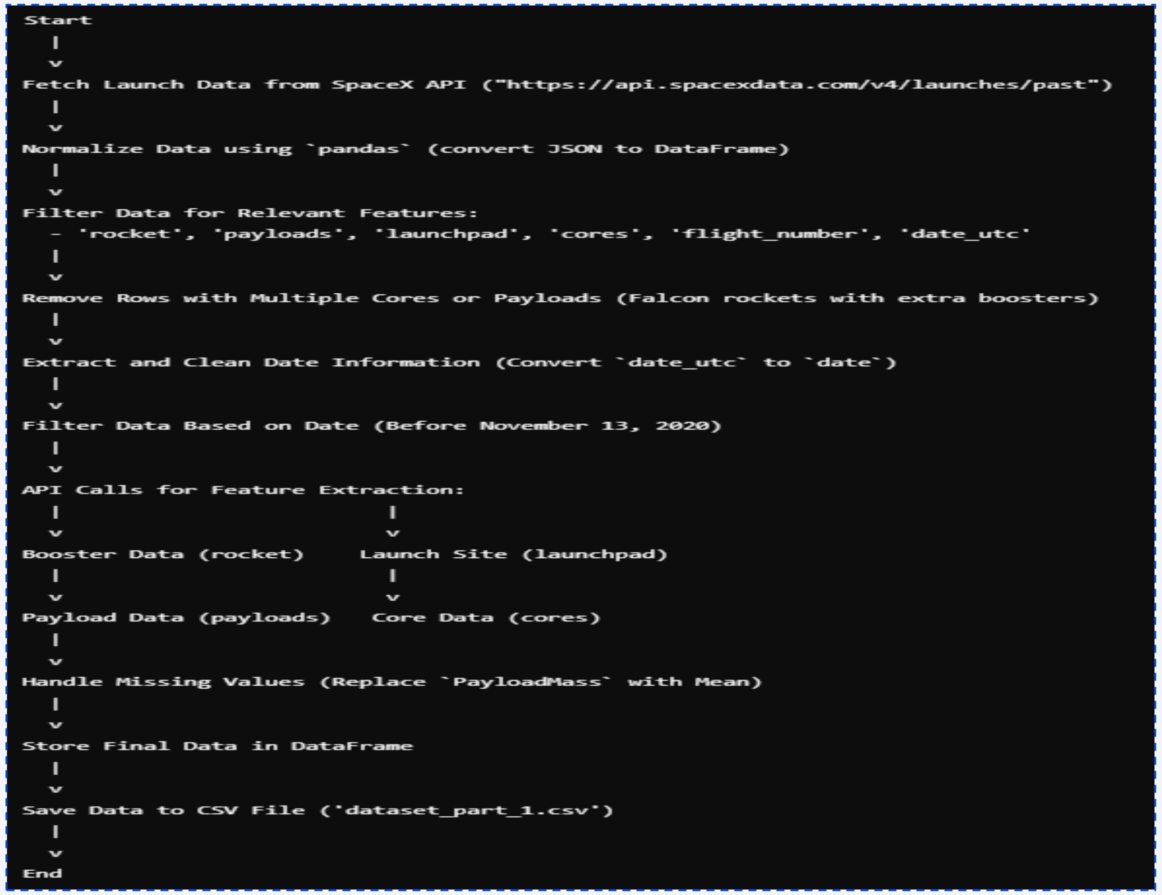
Data Collection

- API Integration: Data was collected from a publicly available API providing information on Falcon 9 rocket launches.
- Data Access: Utilized a Python library (like requests or urllib) to send GET requests to the API.
- Data Retrieval: API responses were in JSON format, containing launch details such as mission IDs, date, and landing success.
- Data Formatting: Cleaned the raw data by handling missing values, normalizing features, and encoding categorical variables.
- Feature Extraction: Key variables like launch date, rocket type, and weather conditions were extracted for analysis.
- Data Storage: The data was stored in a structured format (e.g., pandas DataFrame) for easy manipulation and model training.

Data Collection – SpaceX API

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

- https://github.com/nizamfadzillah/testrepo/blob/main/Module1/jupyter-labs-spacex-data-collection-api-v2.ipynb

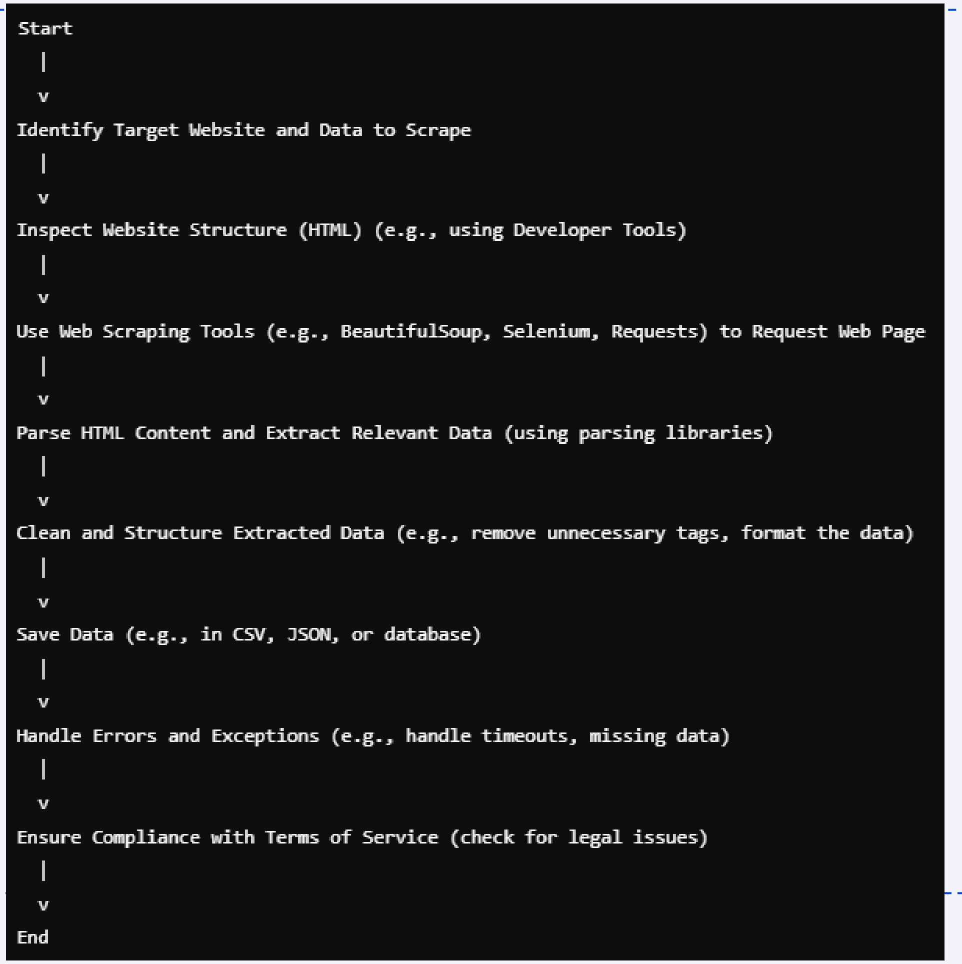


```
Start
  |
  v
Fetch Launch Data from SpaceX API ("https://api.spacexdata.com/v4/launches/past")
  |
  v
Normalize Data using `pandas` (convert JSON to DataFrame)
  |
  v
Filter Data for Relevant Features:
  - 'rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc'
  |
  v
Remove Rows with Multiple Cores or Payloads (Falcon rockets with extra boosters)
  |
  v
Extract and Clean Date Information (Convert `date_utc` to `date`)
  |
  v
Filter Data Based on Date (Before November 13, 2020)
  |
  v
API Calls for Feature Extraction:
  |                         |
  v                         v
Booster Data (rocket)     Launch Site (launchpad)
  |                         |
  v                         v
Payload Data (payloads)    Core Data (cores)
  |
  v
Handle Missing Values (Replace `PayloadMass` with Mean)
  |
  v
Store Final Data in DataFrame
  |
  v
Save Data to CSV File ('dataset_part_1.csv')
  |
  v
End
```

Data Collection - Scraping

- Present your web scraping process using key phrases and flowcharts

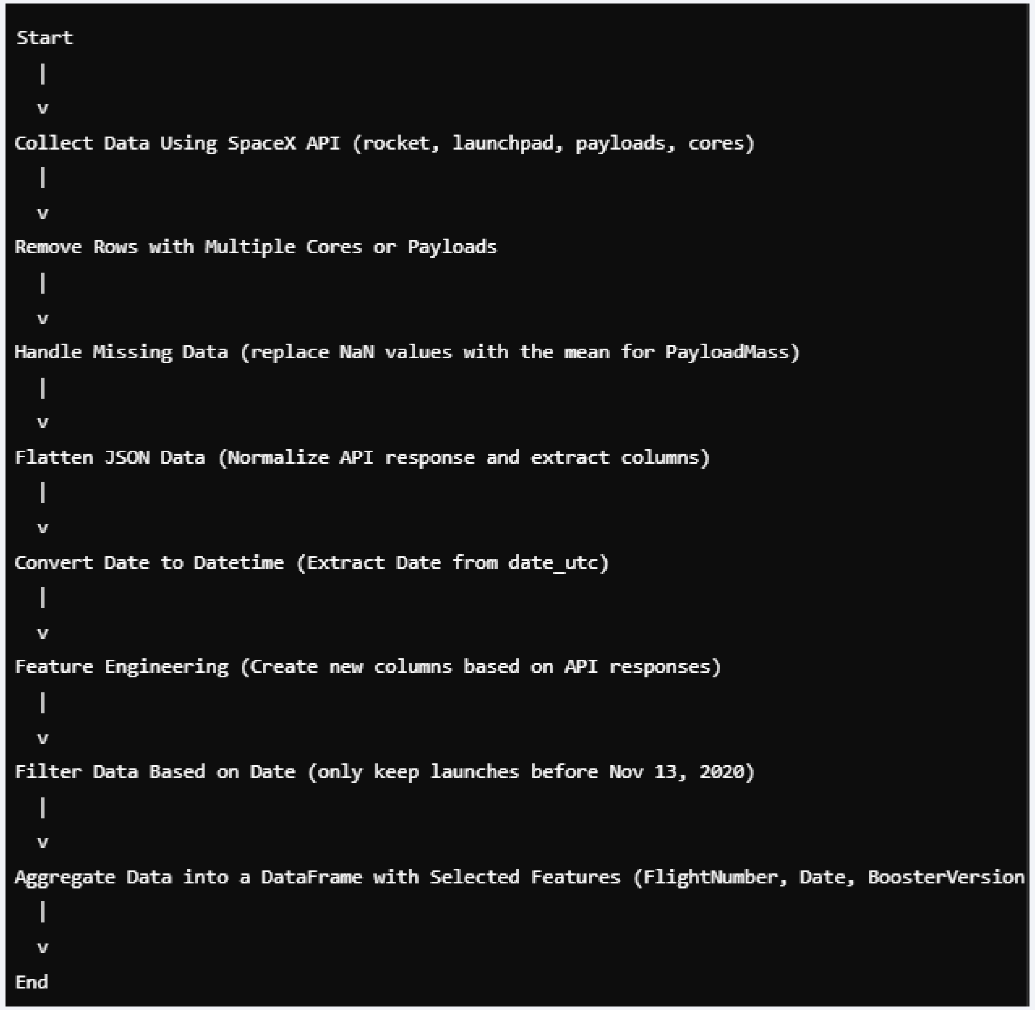
- https://github.com/nizamfadzillah/testrepo/blob/main/Module1/jupyter-labs-webscraping.ipynb



```
Start
  |
  v
Identify Target Website and Data to Scrape
  |
  v
Inspect Website Structure (HTML) (e.g., using Developer Tools)
  |
  v
Use Web Scraping Tools (e.g., BeautifulSoup, Selenium, Requests) to Request Web Page
  |
  v
Parse HTML Content and Extract Relevant Data (using parsing libraries)
  |
  v
Clean and Structure Extracted Data (e.g., remove unnecessary tags, format the data)
  |
  v
Save Data (e.g., in CSV, JSON, or database)
  |
  v
Handle Errors and Exceptions (e.g., handle timeouts, missing data)
  |
  v
Ensure Compliance with Terms of Service (check for legal issues)
  |
  v
End
```

Data Wrangling

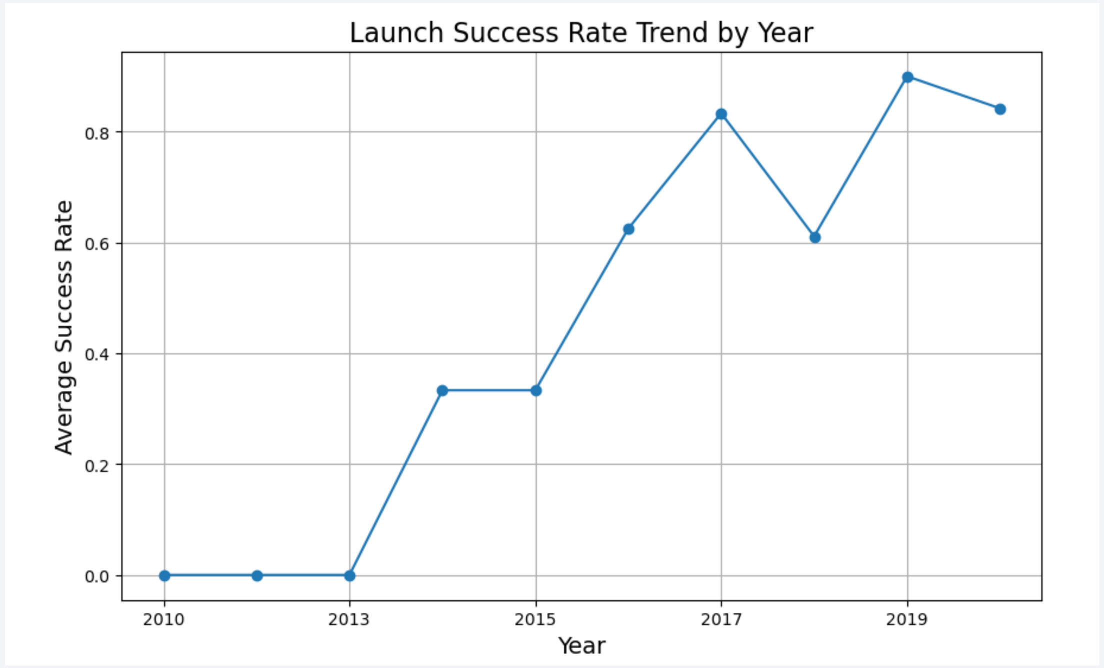
- Describe how data were processed
- You need to present your data wrangling process using key phrases and flowcharts
- https://github.com/nizamfadzillah/testrepo/blob/main/Module1/labs-jupyter-spacex-Data%20wrangling-v2.ipynb



```
Start
  |
  v
Collect Data Using SpaceX API (rocket, launchpad, payloads, cores)
  |
  v
Remove Rows with Multiple Cores or Payloads
  |
  v
Handle Missing Data (replace NaN values with the mean for PayloadMass)
  |
  v
Flatten JSON Data (Normalize API response and extract columns)
  |
  v
Convert Date to Datetime (Extract Date from date_utc)
  |
  v
Feature Engineering (Create new columns based on API responses)
  |
  v
Filter Data Based on Date (only keep launches before Nov 13, 2020)
  |
  v
Aggregate Data into a DataFrame with Selected Features (FlightNumber, Date, BoosterVersion
  |
  v
End
```

EDA with Data Visualization

- Summarize Launch Success Rate Trend increase by year
- https://github.com/nizamfadzillah/testrepo/blob/main/Module2/jupyter-labs-eda-dataviz-v2.ipynb



EDA with SQL

- Using bullet point format, summarize the SQL queries you performed
  - Task 1: Retrieved unique launch sites from the dataset.
  - Task 2: Displayed 5 records with launch sites starting with 'CCA'.
  - Task 3: Summed the total payload mass for NASA (CRS) missions.
  - Task 4: Calculated the average payload mass for F9 v1.1 boosters.
  - Task 5: Found the date of the first successful ground pad landing.
  - Task 6: Listed boosters with successful drone ship landings and payload mass between 4000 and 6000 kg.
  - Task 7: Counted the total number of successful and failed missions.
  - Task 8: Identified boosters that carried the maximum payload mass.
  - Task 9: Listed records with failure outcomes on drone ships for the year 2015, including month names.
  - Task 10: Ranked landing outcomes between 2010-06-04 and 2017-03-20 by count, in descending order
- https://github.com/nizamfadzillah/testrepo/blob/main/Module2/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

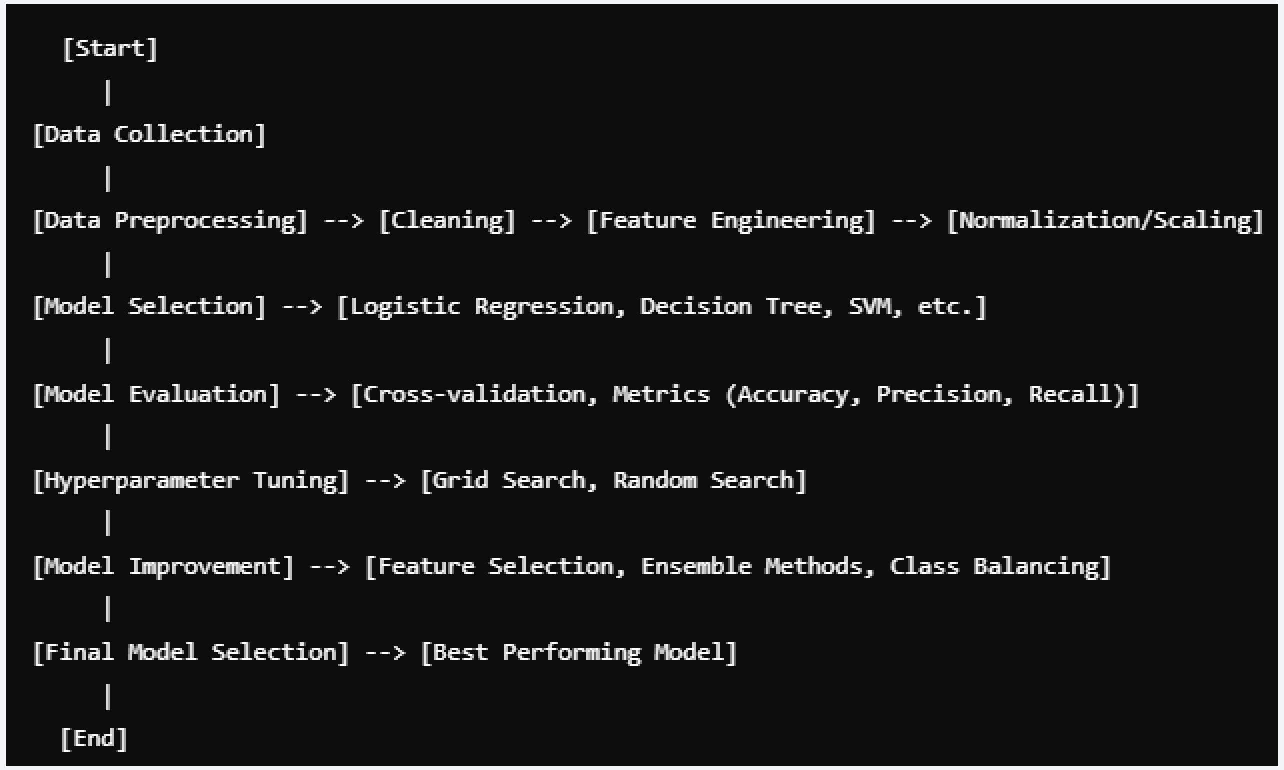
- Summarize what map objects such as markers, circles, lines, etc. you created and added to a folium map
  - Marker(s): These were added to specific locations on the map, marked by latitude and longitude coordinates. Markers were chosen to represent specific points of interest, such as locations tied to the data or relevant features.
  - Circle(s): Circles were added to highlight areas around specific markers, providing a visual representation of a certain radius or impact area. This helps to emphasize certain regions or focus on a particular area.
  - Polyline/Line(s): Lines or polylines were added to show connections or paths between points. These help visualize relationships between different locations or the flow of movement, whether spatial or directional.
  - Popup(s): Popups were associated with markers, offering additional context or detailed information about the location when clicked. This helps to provide more information interactively.
  - Choropleth (if applicable): This might have been used to color regions based on a variable, allowing the user to quickly assess spatial patterns.These objects were added to improve the visualization by making key data points more accessible, interactive, and visually engaging. By adding markers, circles, lines, and popups, I enhanced the map's ability to communicate relevant data effectively.
- https://github.com/nizamfadzillah/testrepo/blob/main/Module3/lab-jupyter-launch-site-location-v2.ipynb

Build a Dashboard with Plotly Dash

- Summarize what plots/graphs and interactions you have added to a dashboard
  - Interactivity: The dropdown, slider, and callbacks make the dashboard interactive, allowing users to filter and explore different aspects of the dataset dynamically. Users can focus on specific launch sites, payload ranges, or analyze success/failure by different parameters.
  - Data Exploration: The pie chart and scatter plot provide visual insights into key aspects of SpaceX launches—success rates by site, success versus payload size, and the influence of booster versions on launch success. This allows users to easily interpret complex relationships within the data.
  - User Engagement: The combination of interactivity and insightful visualizations enhances user engagement, allowing for a more customized and detailed exploration of the SpaceX launch data.
- https://github.com/nizamfadzillah/testrepo/blob/main/Module3/Build%20an%20Interactive%20Dashboard%20with%20Ploty%20Dash.py

Predictive Analysis (Classification)

- Summarize how you built, evaluated, improved, and found the best performing classification model
- https://github.com/nizamfadzillah/testrepo/blob/main/Module4/SpaceX-Machine-Learning-Prediction-Part-5-v1.ipynb



```
  [Start]
     |
[Data Collection]
     |
[Data Preprocessing] --> [Cleaning] --> [Feature Engineering] --> [Normalization/Scaling]
     |
[Model Selection] --> [Logistic Regression, Decision Tree, SVM, etc.]
     |
[Model Evaluation] --> [Cross-validation, Metrics (Accuracy, Precision, Recall)]
     |
[Hyperparameter Tuning] --> [Grid Search, Random Search]
     |
[Model Improvement] --> [Feature Selection, Ensemble Methods, Class Balancing]
     |
[Final Model Selection] --> [Best Performing Model]
     |
  [End]
```

Results

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

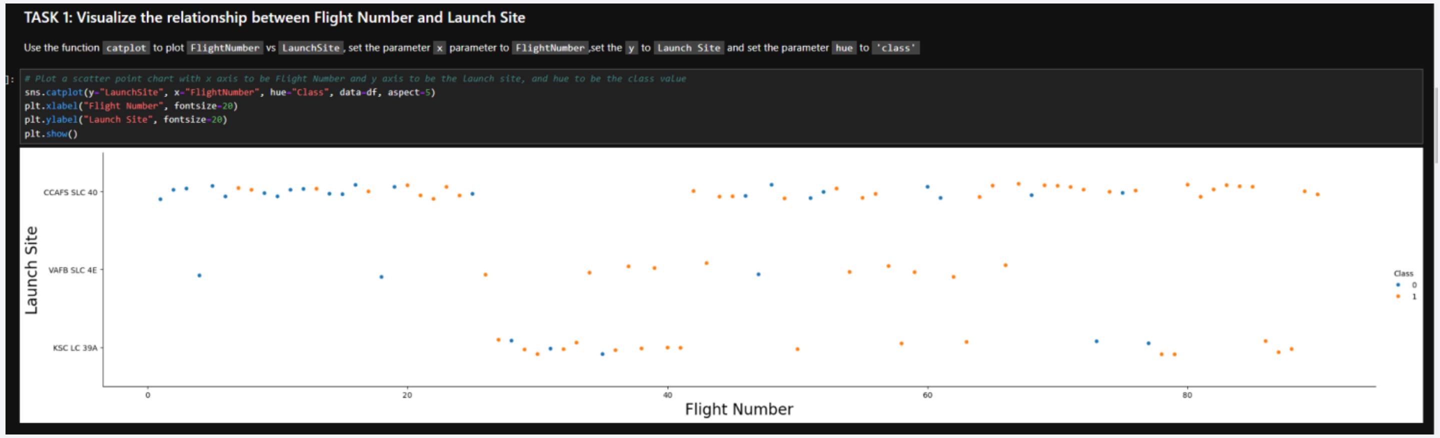
Section 2

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

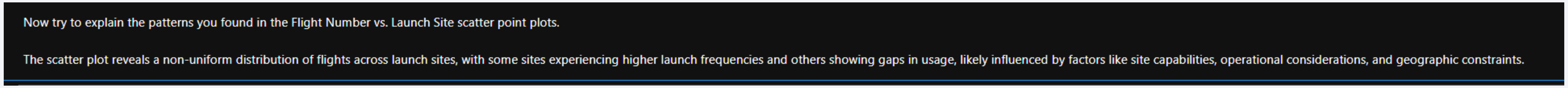


TASK 1: Visualize the relationship between Flight Number and Launch Site

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'`

```
# 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()
```

CCAFS SLC 40
VAFB SLC 4E
KSC LC 39A
Launch Site
0
20
40
60
80
Flight Number
Class
0
1

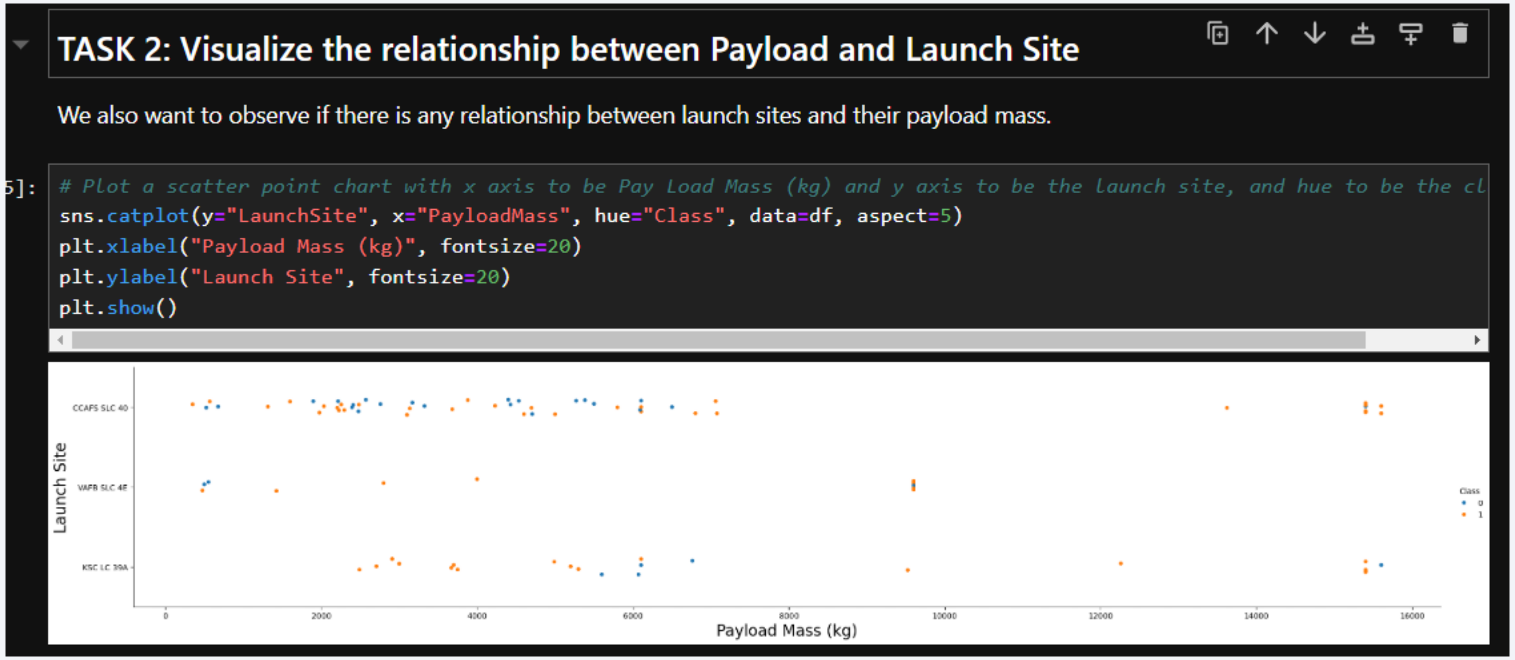


Now try to explain the patterns you found in the Flight Number vs. Launch Site scatter point plots.

The scatter plot reveals a non-uniform distribution of flights across launch sites, with some sites experiencing higher launch frequencies and others showing gaps in usage, likely influenced by factors like site capabilities, operational considerations, and geographic constraints.

Payload vs. Launch Site

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



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.

```
# 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 cl
sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect=5)
plt.xlabel("Payload Mass (kg)", fontsize=20)
plt.ylabel("Launch Site", fontsize=20)
plt.show()
```

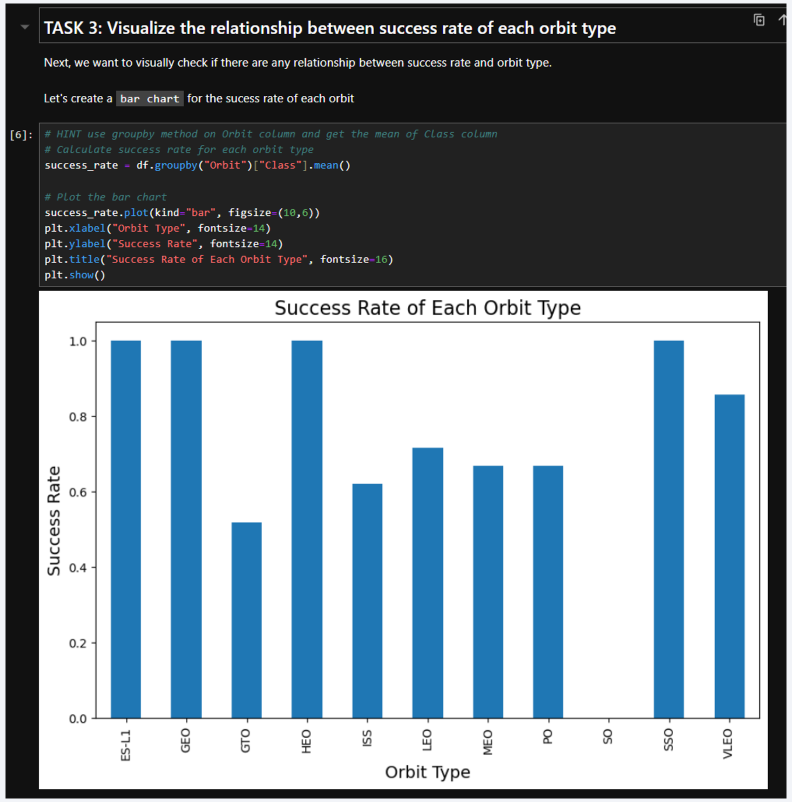
Now if you observe Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

The scatter plot reveals that the VAFB-SLC launch site has not been used for any rockets with a payload mass exceeding 10,000 kg, suggesting potential limitations in handling heavy payloads at this location.

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



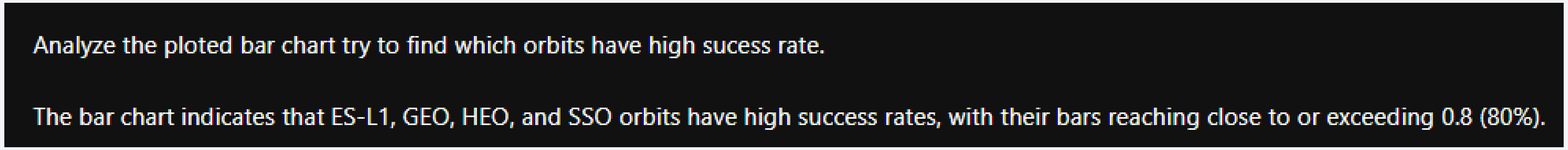
TASK 3: Visualize the relationship between success rate of each orbit type

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

[6]:
```
# HINT use groupby method on Orbit column and get the mean of Class column
# Calculate success rate for each orbit type
success_rate = df.groupby("Orbit")["Class"].mean()

# Plot the bar chart
success_rate.plot(kind="bar", figsize=(10,6))
plt.xlabel("Orbit Type", fontsize=14)
plt.ylabel("Success Rate", fontsize=14)
plt.title("Success Rate of Each Orbit Type", fontsize=16)
plt.show()
```

Success Rate of Each Orbit Type
Success Rate
1.0
0.8
0.6
0.4
0.2
0.0
ES-L1
GEO
GTO
HEO
ISS
LEO
MEO
PO
SO
SSO
VLEO
Orbit Type



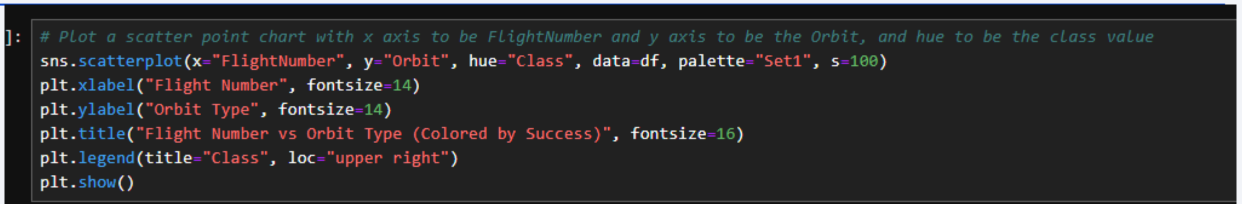
Analyze the ploted bar chart try to find which orbits have high sucess rate.

The bar chart indicates that ES-L1, GEO, HEO, and SSO orbits have high success rates, with their bars reaching close to or exceeding 0.8 (80%).

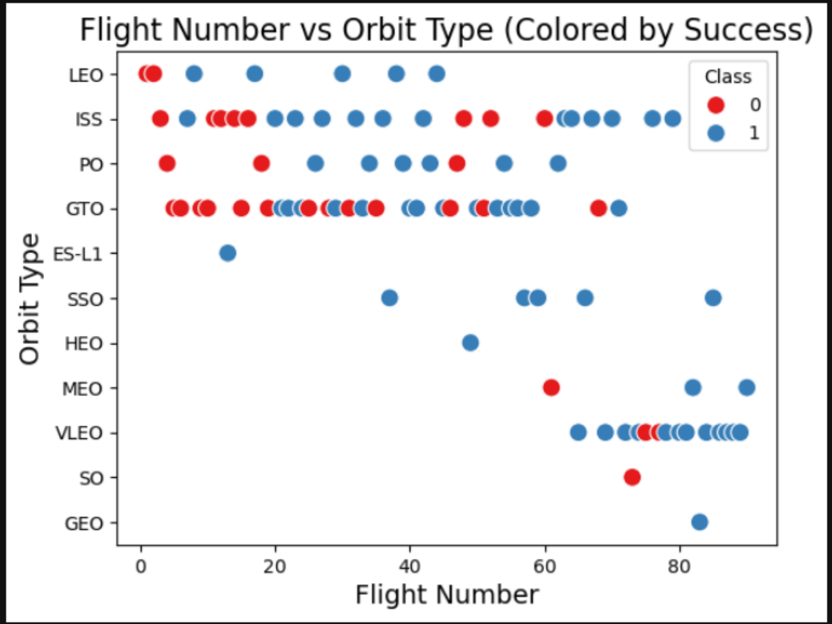
Flight Number vs. Orbit Type

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

- Show the screenshot of the scatter plot with explanations



```
]: # 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, palette="Set1", s=100)
plt.xlabel("Flight Number", fontsize=14)
plt.ylabel("Orbit Type", fontsize=14)
plt.title("Flight Number vs Orbit Type (Colored by Success)", fontsize=16)
plt.legend(title="Class", loc="upper right")
plt.show()
```

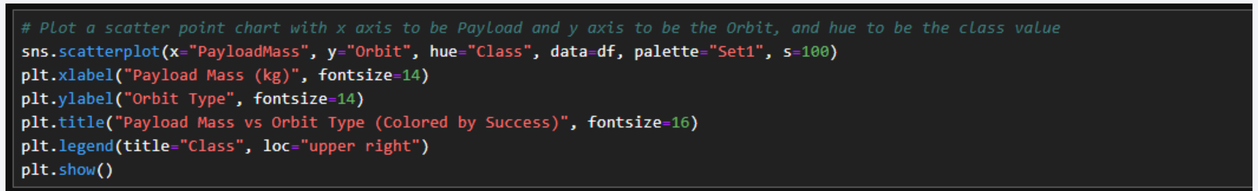


You should see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

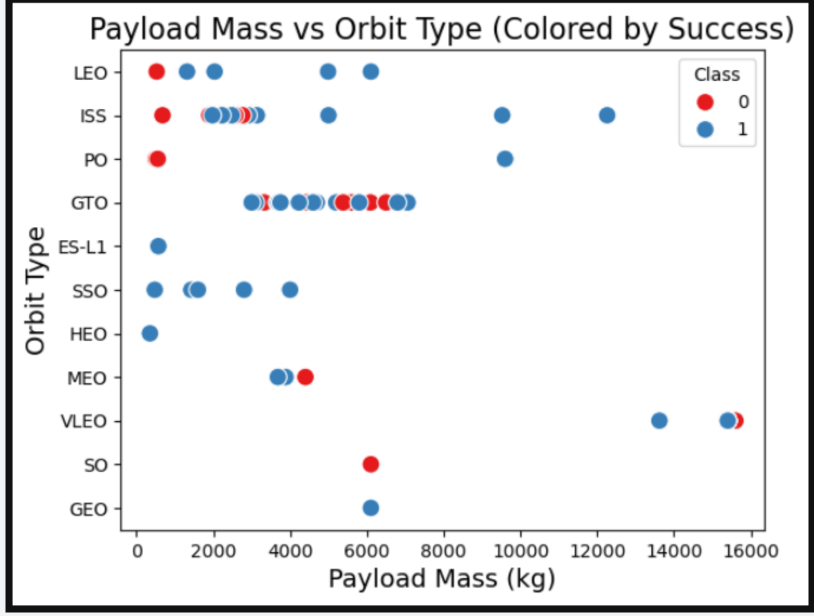
The scatter plot suggests a potential correlation between success rate and flight number for LEO orbits, while no such relationship is apparent for GTO orbits.

Payload vs. Orbit Type

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



```
# 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(x="PayloadMass", y="Orbit", hue="Class", data=df, palette="Set1", s=100)
plt.xlabel("Payload Mass (kg)", fontsize=14)
plt.ylabel("Orbit Type", fontsize=14)
plt.title("Payload Mass vs Orbit Type (Colored by Success)", fontsize=16)
plt.legend(title="Class", loc="upper right")
plt.show()
```



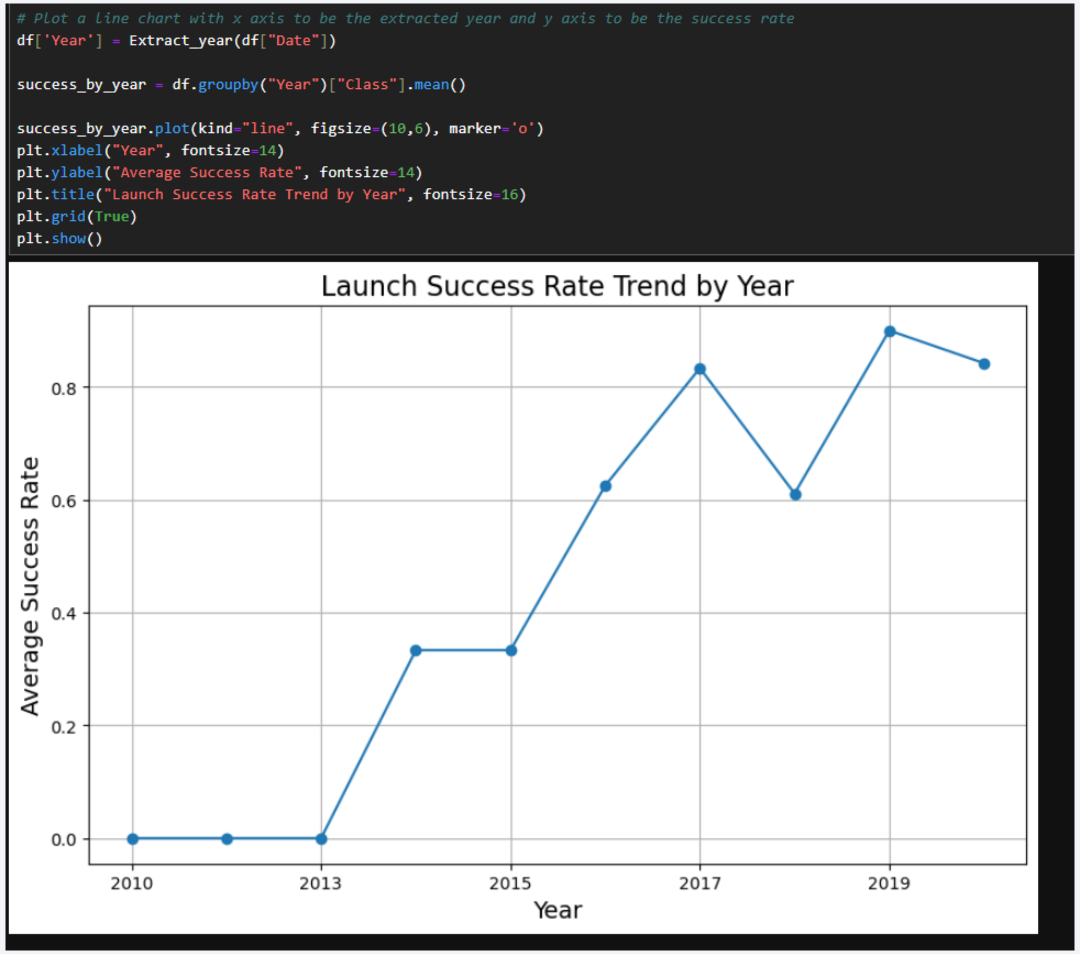
With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.

Launch Success Yearly Trend

- Show a line chart of yearly average success rate

- Show the screenshot of the scatter plot with explanations

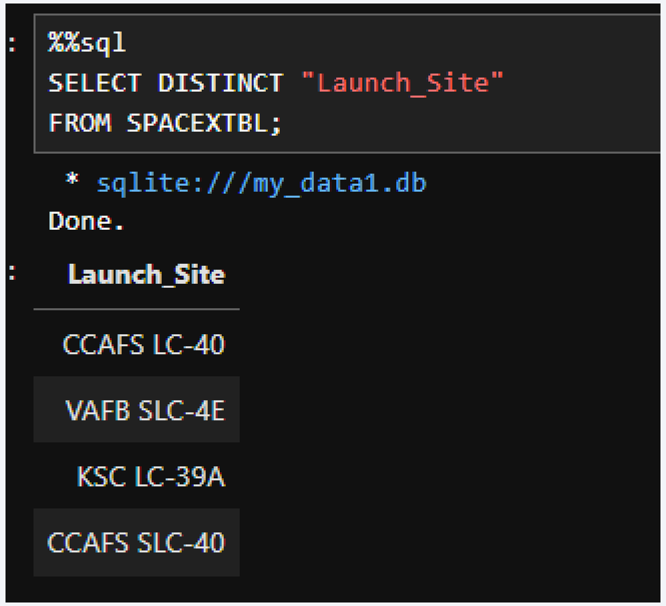


```
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
df['Year'] = Extract_year(df["Date"])

success_by_year = df.groupby("Year")["Class"].mean()

success_by_year.plot(kind="line", figsize=(10,6), marker='o')
plt.xlabel("Year", fontsize=14)
plt.ylabel("Average Success Rate", fontsize=14)
plt.title("Launch Success Rate Trend by Year", fontsize=16)
plt.grid(True)
plt.show()
```

You can observe that the success rate since 2013 kept increasing till 2017 (stable in 2014) and after 2015 it started increasing.

All Launch Site Names

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

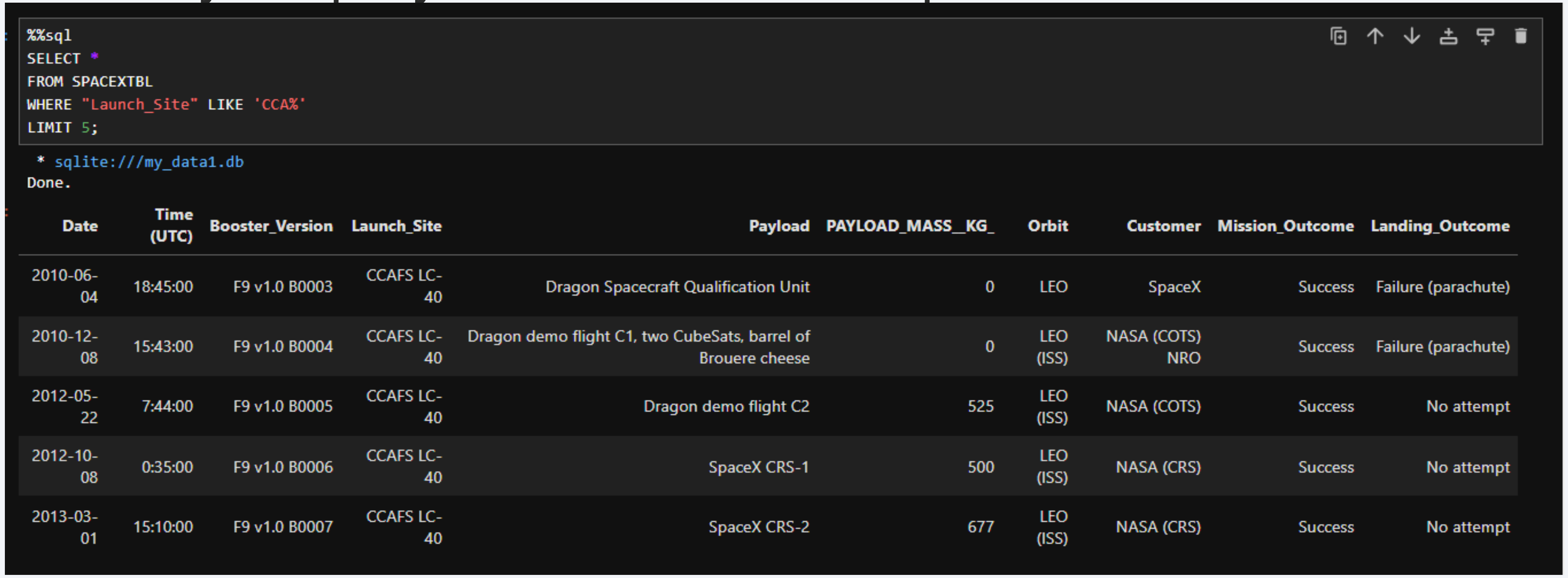


```
%%sql
SELECT DISTINCT "Launch_Site"
FROM SPACEXTBL;

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

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

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`
- Present your query result with a short explanation here



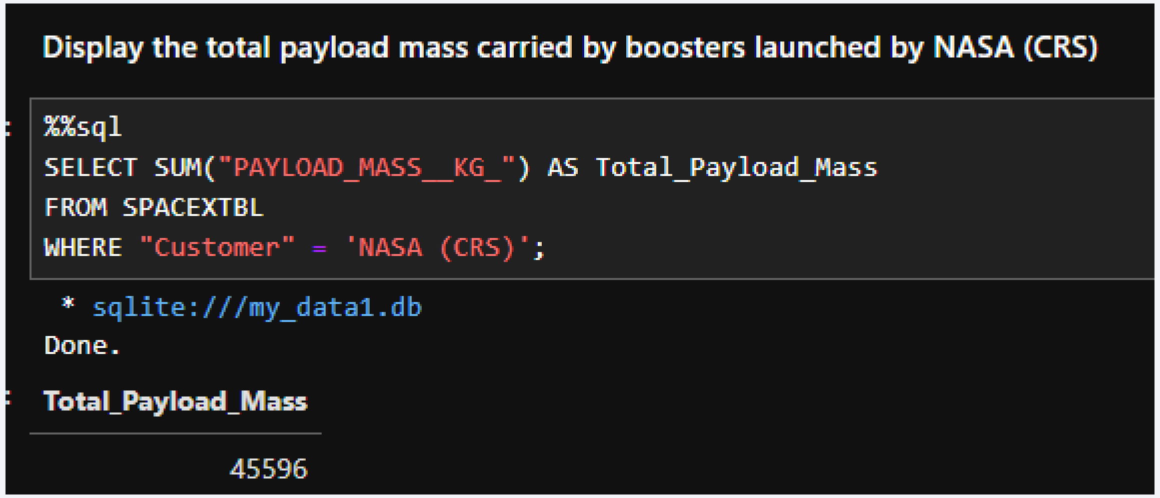
```
%%sql
SELECT *
FROM SPACEXTBL
WHERE "Launch_Site" LIKE 'CCA%'
LIMIT 5;
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

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


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

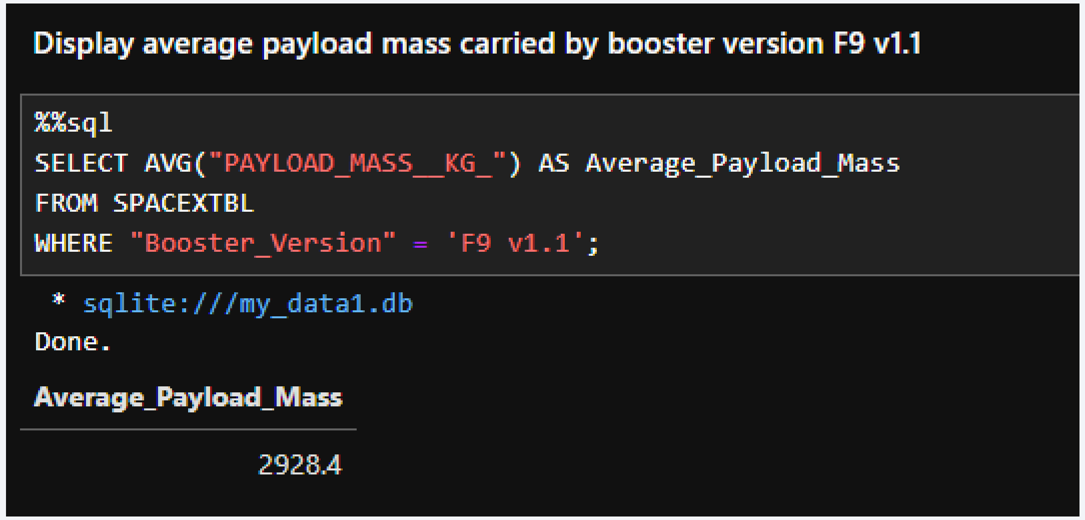
```
%%sql
SELECT SUM("PAYLOAD_MASS__KG_") AS Total_Payload_Mass
FROM SPACEXTBL
WHERE "Customer" = 'NASA (CRS)';
```

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

Total_Payload_Mass
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



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

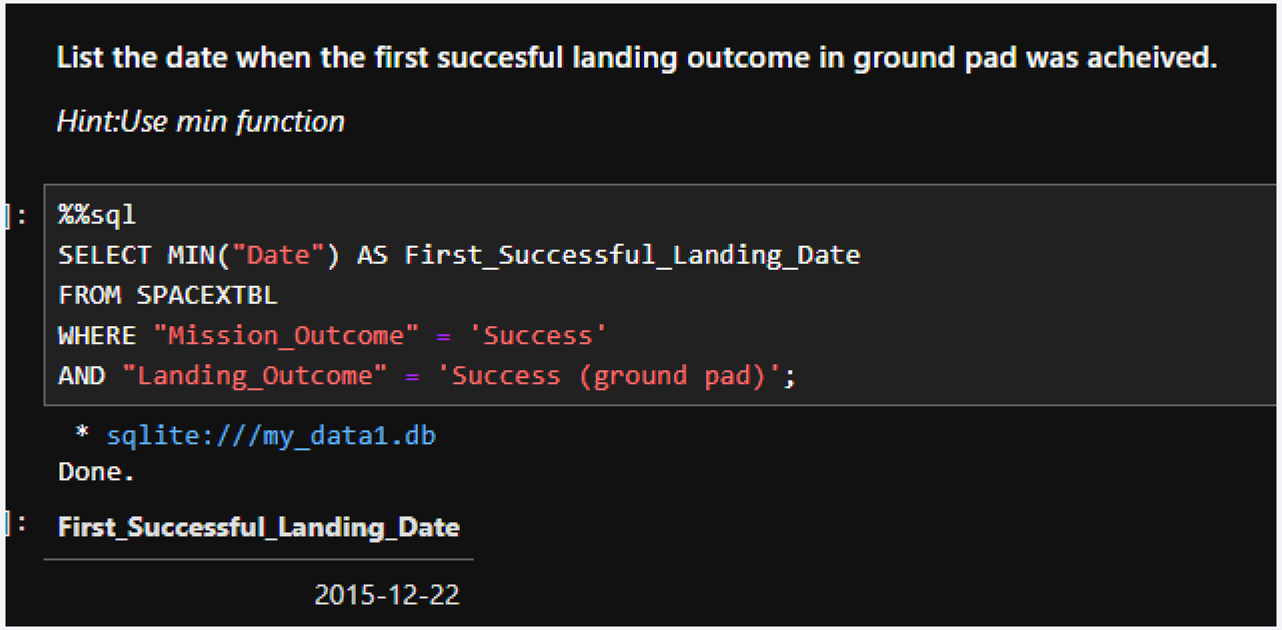
```
%%sql
SELECT AVG("PAYLOAD_MASS__KG_") AS Average_Payload_Mass
FROM SPACEXTBL
WHERE "Booster_Version" = 'F9 v1.1';
```

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

Average_Payload_Mass
2928.4

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad
- Present your query result with a short explanation here



List the date when the first succesful landing outcome in ground pad was acheived.

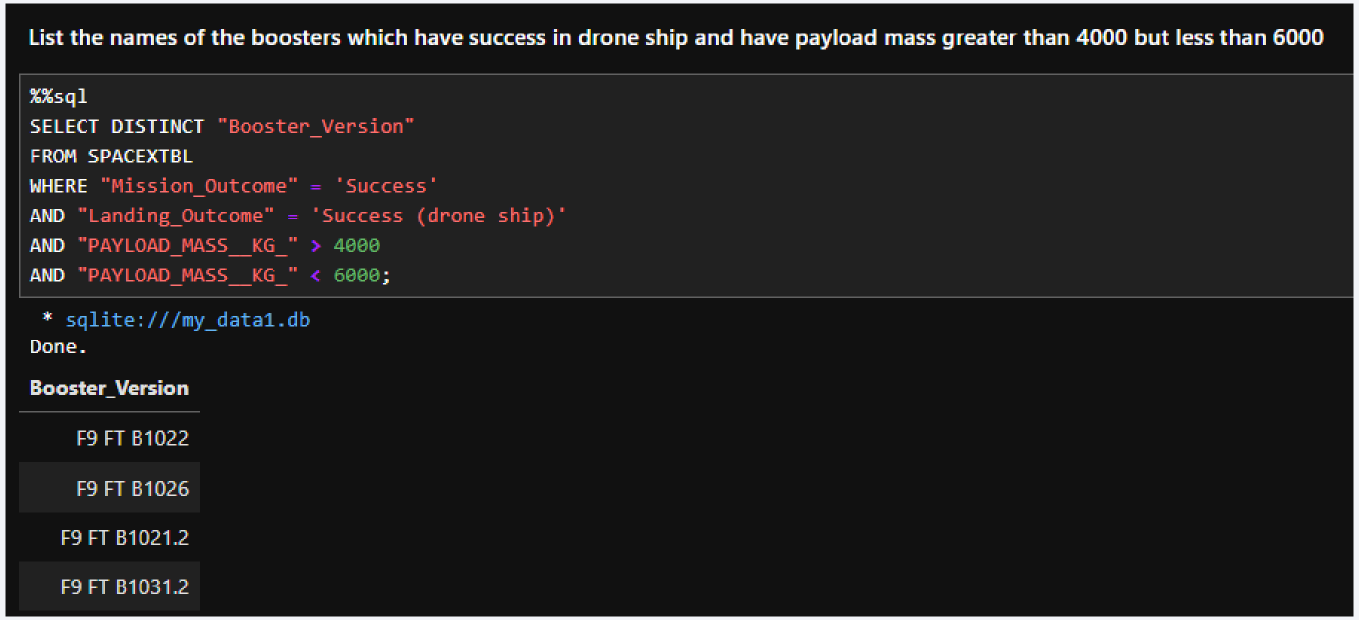
Hint:Use min function

```
]: %%sql
   SELECT MIN("Date") AS First_Successful_Landing_Date
   FROM SPACEXTBL
   WHERE "Mission_Outcome" = 'Success'
   AND "Landing_Outcome" = 'Success (ground pad)';

    * sqlite:///my_data1.db
   Done.
]: First_Successful_Landing_Date
                      2015-12-22
```

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



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

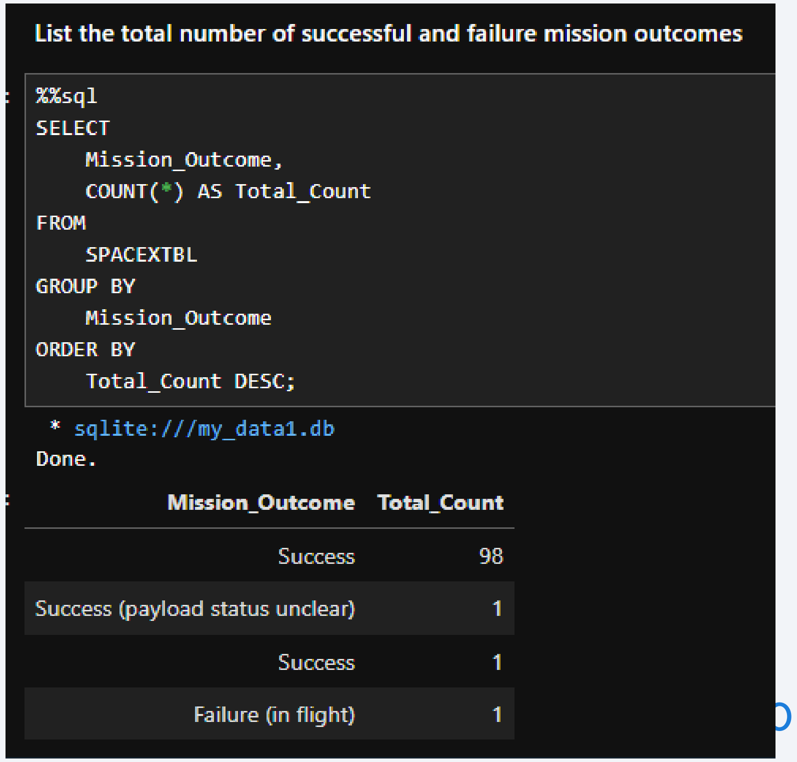
```
%%sql
SELECT DISTINCT "Booster_Version"
FROM SPACEXTBL
WHERE "Mission_Outcome" = 'Success'
AND "Landing_Outcome" = 'Success (drone ship)'
AND "PAYLOAD_MASS__KG_" > 4000
AND "PAYLOAD_MASS__KG_" < 6000;
```

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

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

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



List the total number of successful and failure mission outcomes

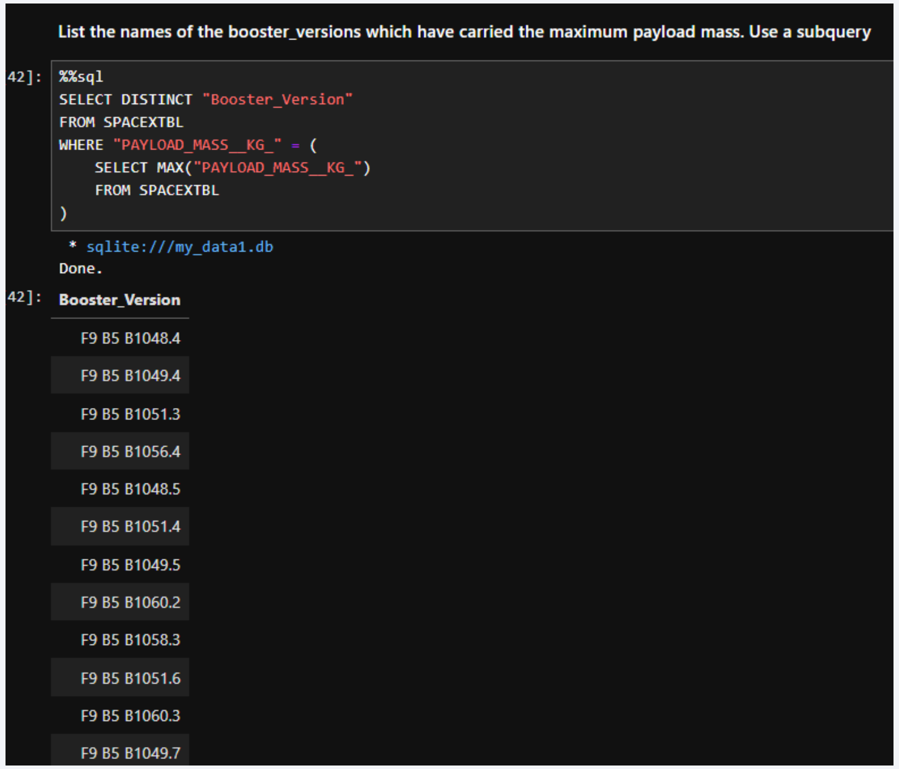
```
%%sql
SELECT
    Mission_Outcome,
    COUNT(*) AS Total_Count
FROM
    SPACEXTBL
GROUP BY
    Mission_Outcome
ORDER BY
    Total_Count DESC;
```

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

Mission_Outcome	Total_Count
Success	98
Success (payload status unclear)	1
Success	1
Failure (in flight)	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



List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

```
42]: %%sql
SELECT DISTINCT "Booster_Version"
FROM SPACEXTBL
WHERE "PAYLOAD_MASS__KG_" = (
    SELECT MAX("PAYLOAD_MASS__KG_")
    FROM SPACEXTBL
)
```

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

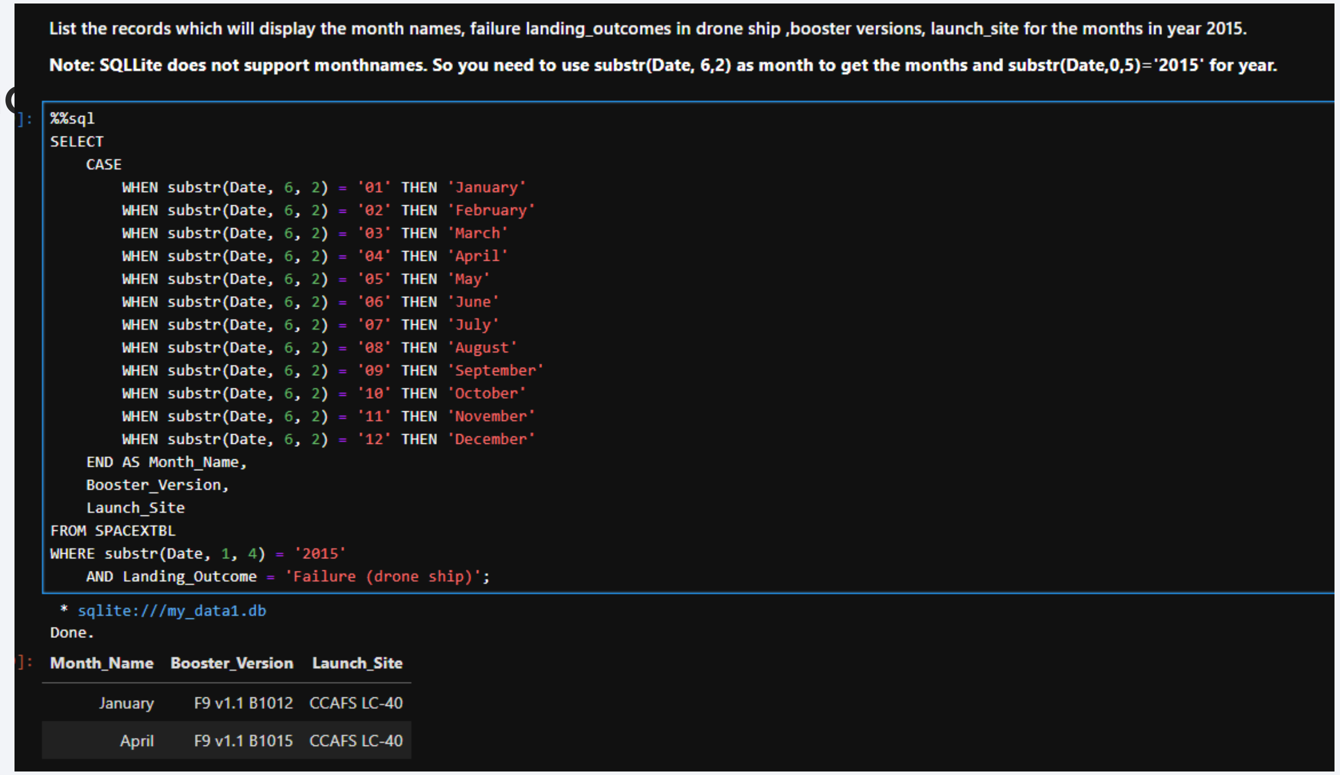
42]:

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

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

- Present your query result with a sh



List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

```
%%sql
SELECT
    CASE
        WHEN substr(Date, 6, 2) = '01' THEN 'January'
        WHEN substr(Date, 6, 2) = '02' THEN 'February'
        WHEN substr(Date, 6, 2) = '03' THEN 'March'
        WHEN substr(Date, 6, 2) = '04' THEN 'April'
        WHEN substr(Date, 6, 2) = '05' THEN 'May'
        WHEN substr(Date, 6, 2) = '06' THEN 'June'
        WHEN substr(Date, 6, 2) = '07' THEN 'July'
        WHEN substr(Date, 6, 2) = '08' THEN 'August'
        WHEN substr(Date, 6, 2) = '09' THEN 'September'
        WHEN substr(Date, 6, 2) = '10' THEN 'October'
        WHEN substr(Date, 6, 2) = '11' THEN 'November'
        WHEN substr(Date, 6, 2) = '12' THEN 'December'
    END AS Month_Name,
    Booster_Version,
    Launch_Site
FROM SPACEXTBL
WHERE substr(Date, 1, 4) = '2015'
    AND Landing_Outcome = 'Failure (drone ship)';
```

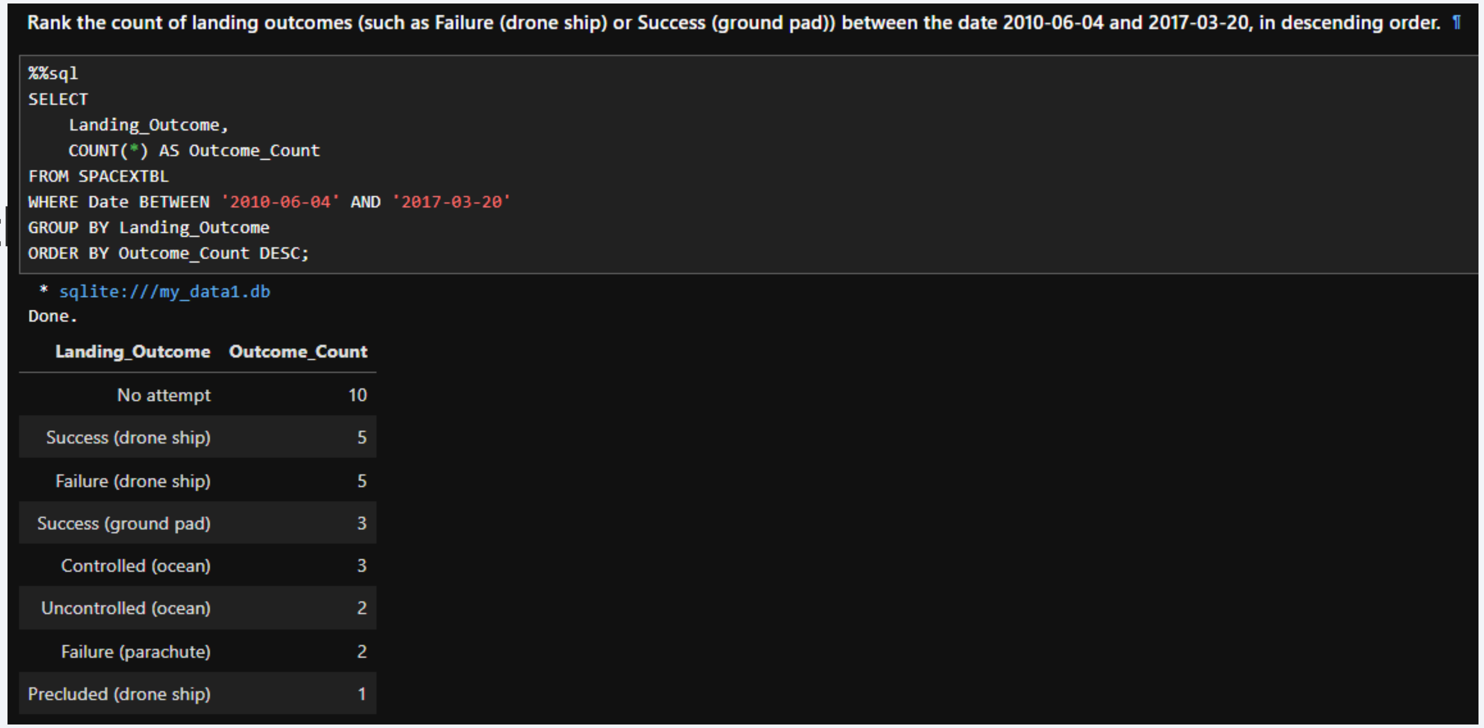
```
 * sqlite:///my_data1.db
Done.
```

Month_Name	Booster_Version	Launch_Site
January	F9 v1.1 B1012	CCAFS LC-40
April	F9 v1.1 B1015	CCAFS LC-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 wit



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. ¶

```
%%sql
SELECT
    Landing_Outcome,
    COUNT(*) AS Outcome_Count
FROM SPACEXTBL
WHERE Date BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY Landing_Outcome
ORDER BY Outcome_Count DESC;
```

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

Landing_Outcome	Outcome_Count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

<Folium Map Screenshot 1>

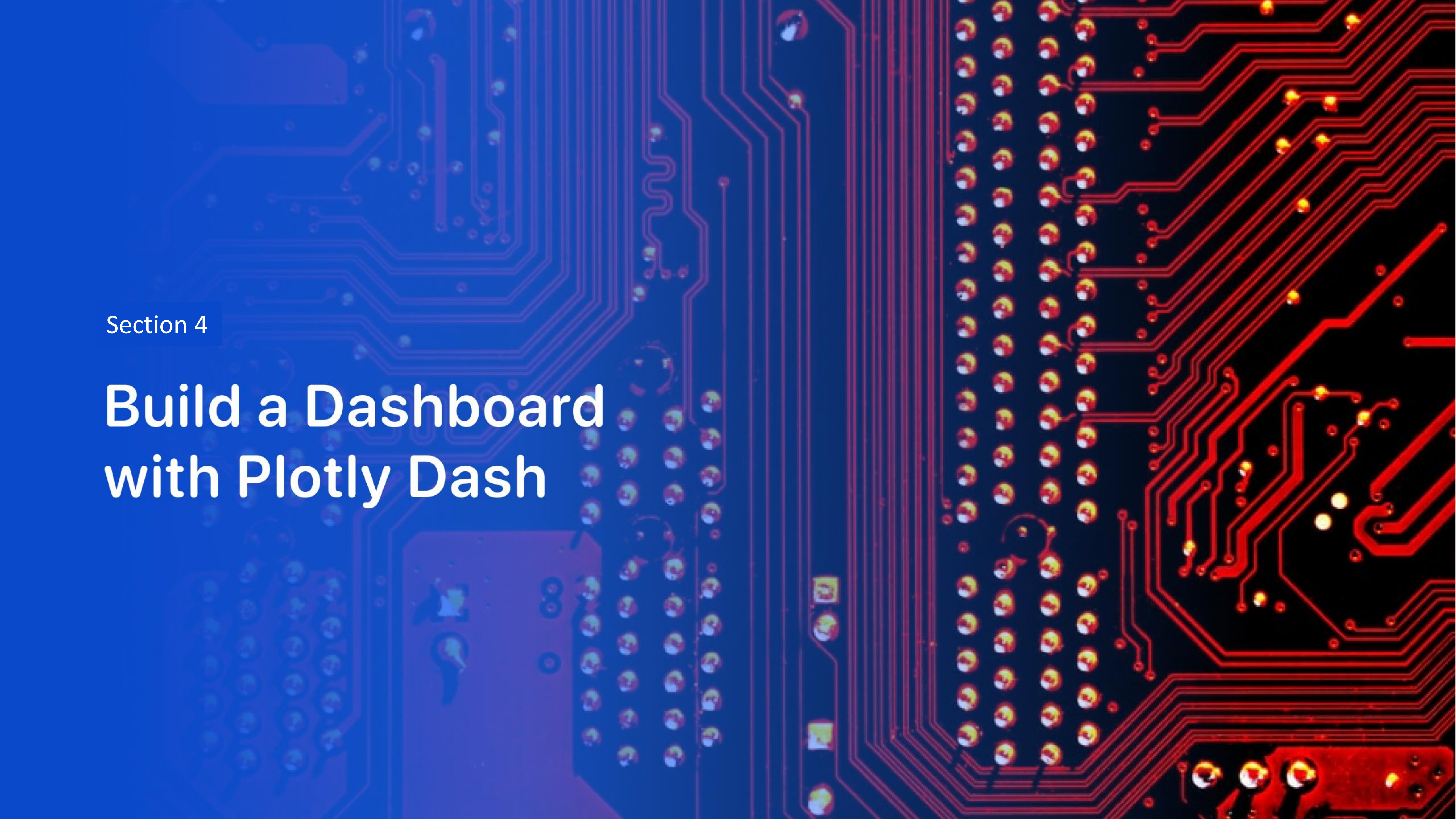
- Replace <Folium map screenshot 1> title with an appropriate title
- Explore the generated folium map and make a proper screenshot to include all launch sites' location markers on a global map
- Explain the important elements and findings on the screenshot

<Folium Map Screenshot 2>

- 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

<Folium Map Screenshot 3>

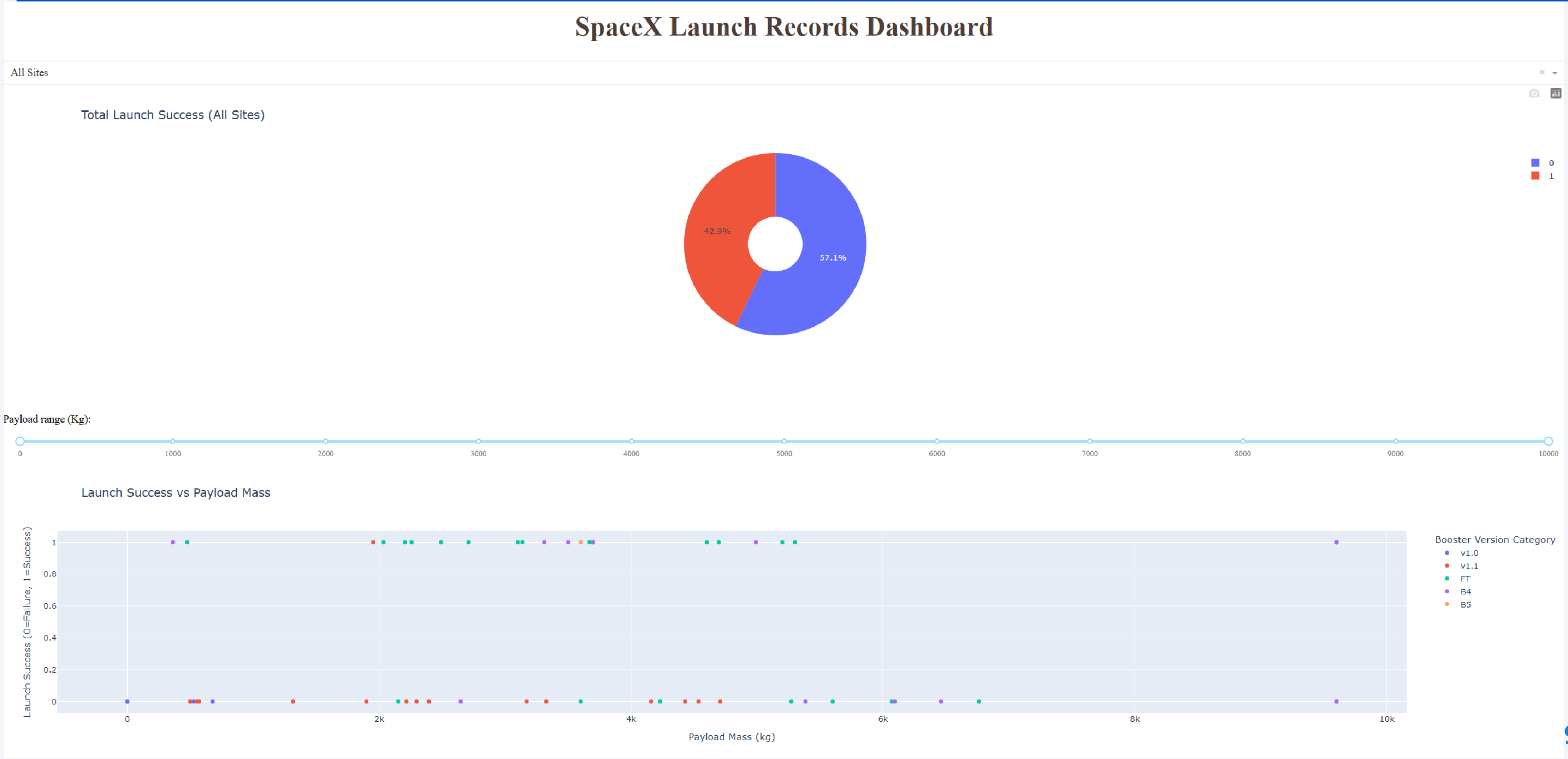
- Replace <Folium map screenshot 3> title with an appropriate title
- Explore the generated folium map and show the screenshot of a selected launch site to its proximities such as railway, highway, coastline, with distance calculated and displayed
- Explain the important elements and findings on the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



Section 4

Build a Dashboard with Plotly Dash

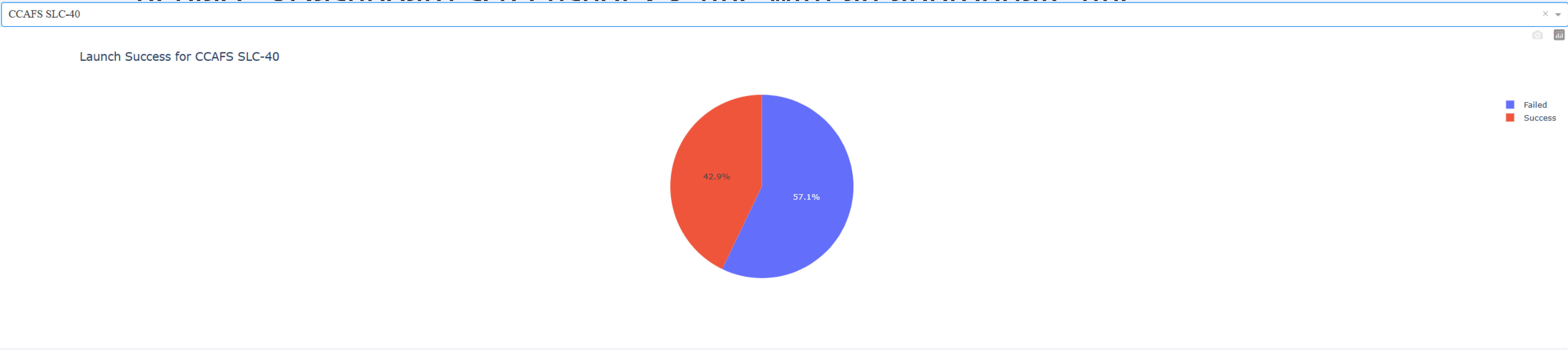
<Dashboard Screenshot 1>


SpaceX Launch Records Dashboard

All Sites

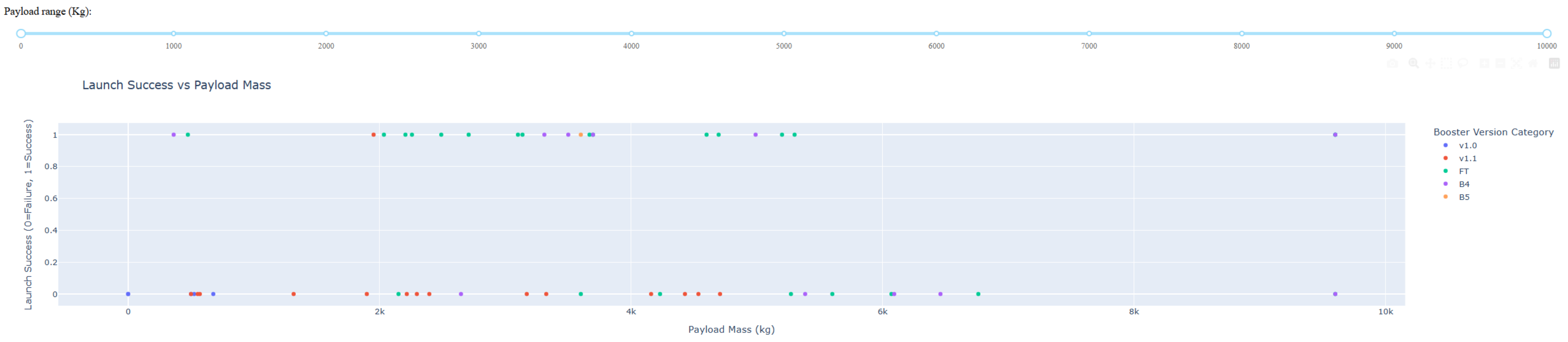
<Dashboard Screenshot 2>

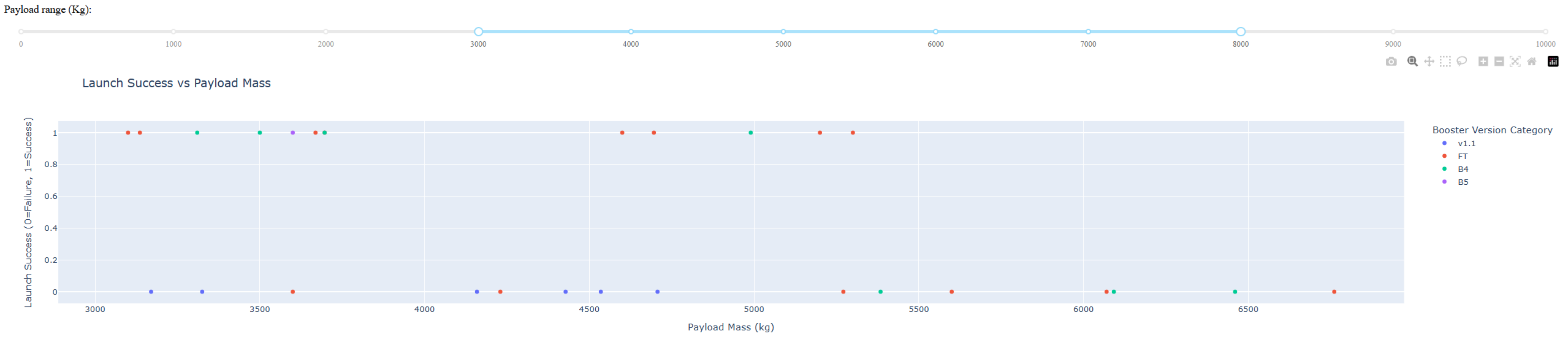
- Replace <Dashboard screenshot 2> title with an appropriate title


CCAFS SLC-40

Launch Success for CCAFS SLC-40

<Dashboard Screenshot 3>


Payload range (Kg):


Payload range (Kg):

Section 5

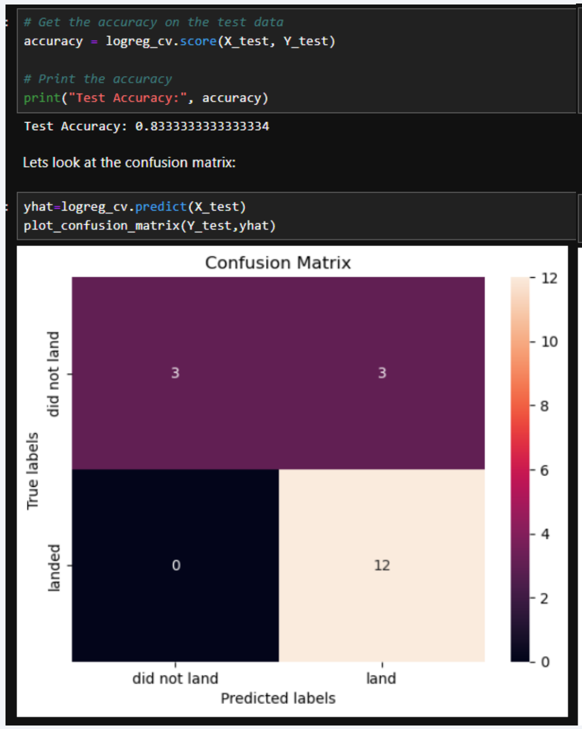
Predictive Analysis (Classification)

Classification Accuracy

- All model have same accuracy

Confusion Matrix

- Show the confusion matrix of the best performing model with an explanation

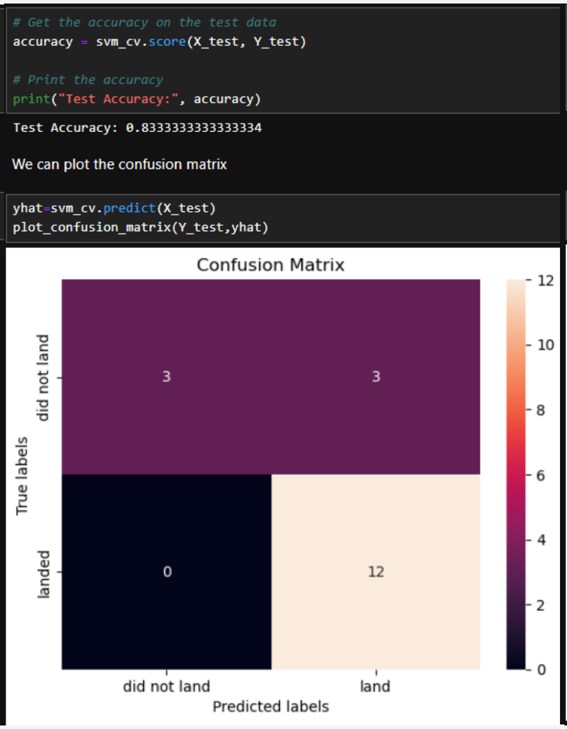


```
# Get the accuracy on the test data
accuracy = logreg_cv.score(X_test, Y_test)

# Print the accuracy
print("Test Accuracy:", accuracy)
```

```
Test Accuracy: 0.8333333333333334
```

Lets look at the confusion matrix:

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

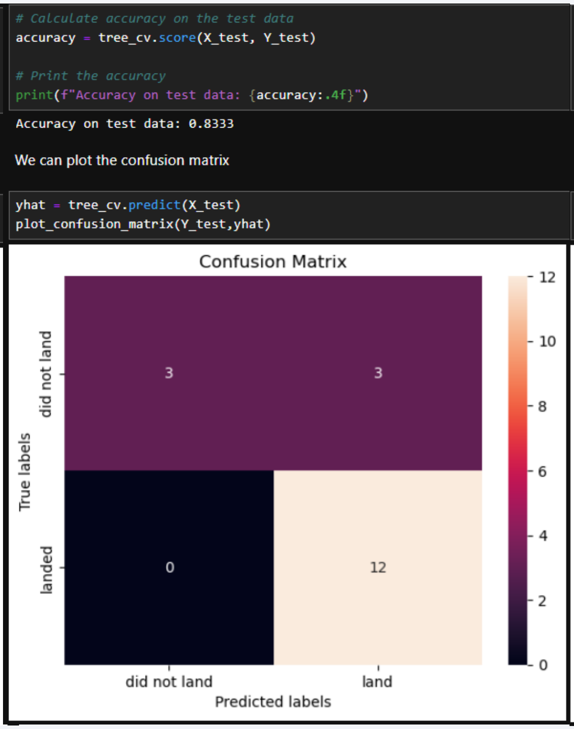


```
# Get the accuracy on the test data
accuracy = svm_cv.score(X_test, Y_test)

# Print the accuracy
print("Test Accuracy:", accuracy)
```

```
Test Accuracy: 0.8333333333333334
```

We can plot the confusion matrix

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

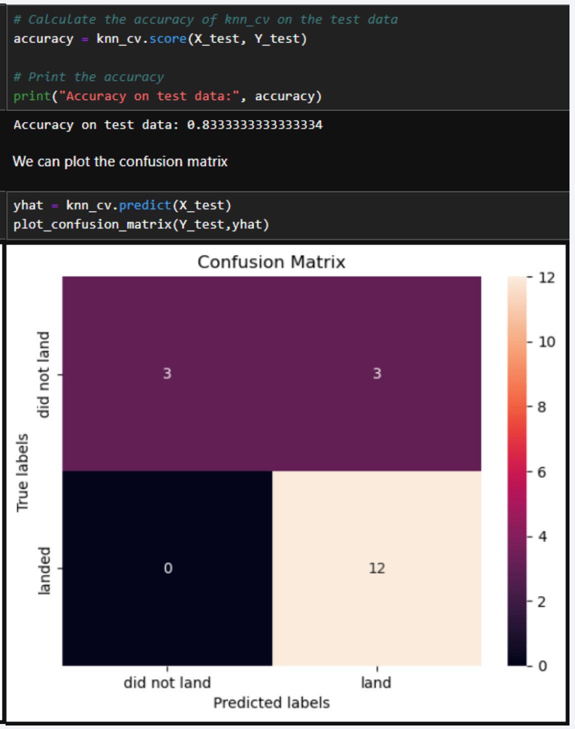


```
# Calculate accuracy on the test data
accuracy = tree_cv.score(X_test, Y_test)

# Print the accuracy
print(f"Accuracy on test data: {accuracy:.4f}")
```

```
Accuracy on test data: 0.8333
```

We can plot the confusion matrix

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



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

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

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

We can plot the confusion matrix

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

Conclusions

- Point 1
- Point 2
- Point 3
- Point 4
- …

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!