IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Leon Mbano
21 September 2024

Outline

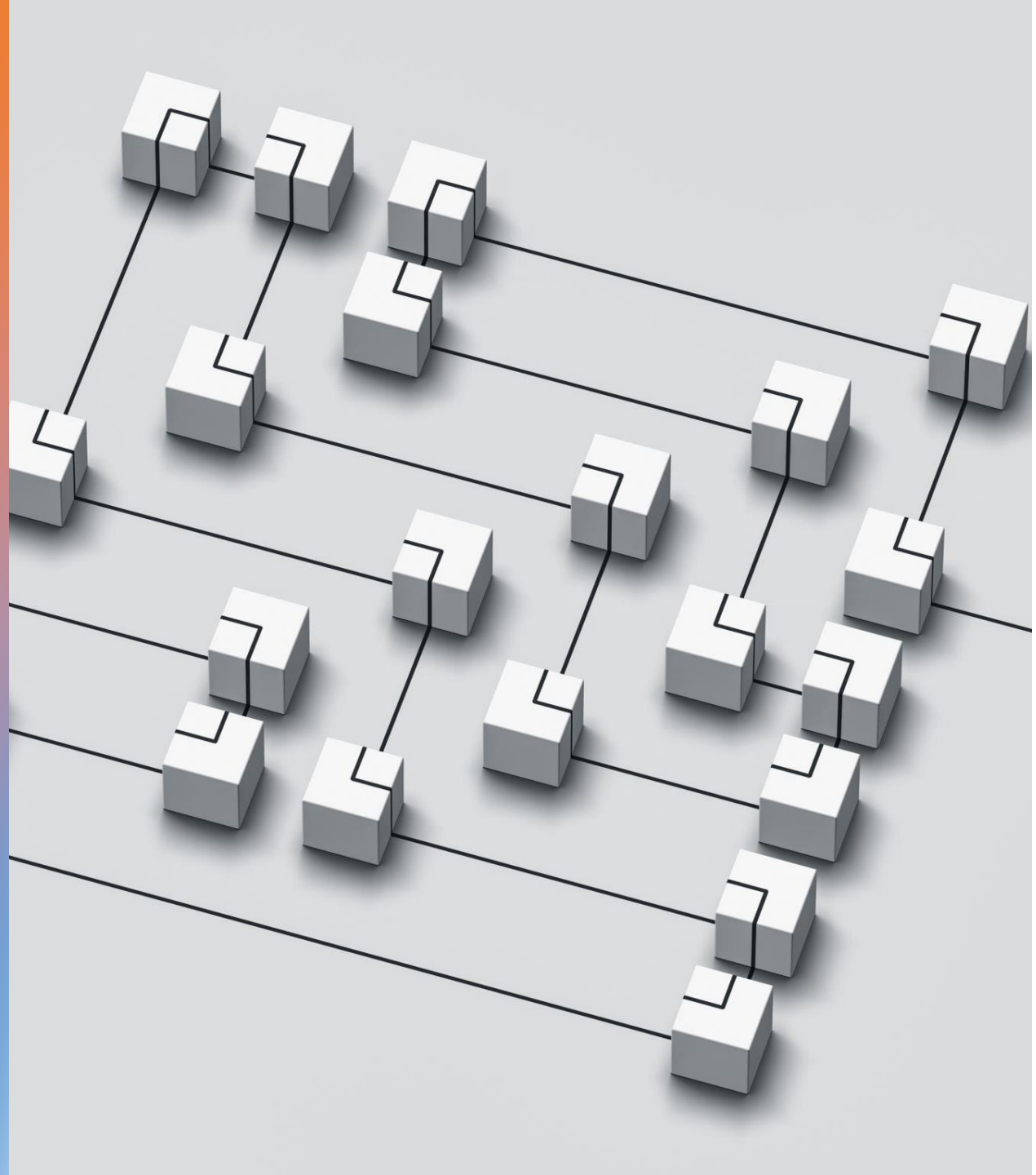
- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion
- Appendix

Executive Summary

- Summary of methodologies
- Summary of all results

Project Background:

SpaceX's reusable Falcon 9 rocket has revolutionized the space launch industry by significantly reducing costs. The ability to recover and reuse the first stage of the rocket is a key factor in SpaceX's competitive pricing, with launches costing $62 million compared to upwards of $165 million from other providers



Project Context

The goal of this capstone project is to develop a predictive model that determines whether the Falcon 9 first stage will land successfully. By accurately predicting the success of the first stage landing, the project aims to estimate the cost of a launch. This information will enable other companies to competitively bid against SpaceX for rocket launches.

Section 1

Methodology

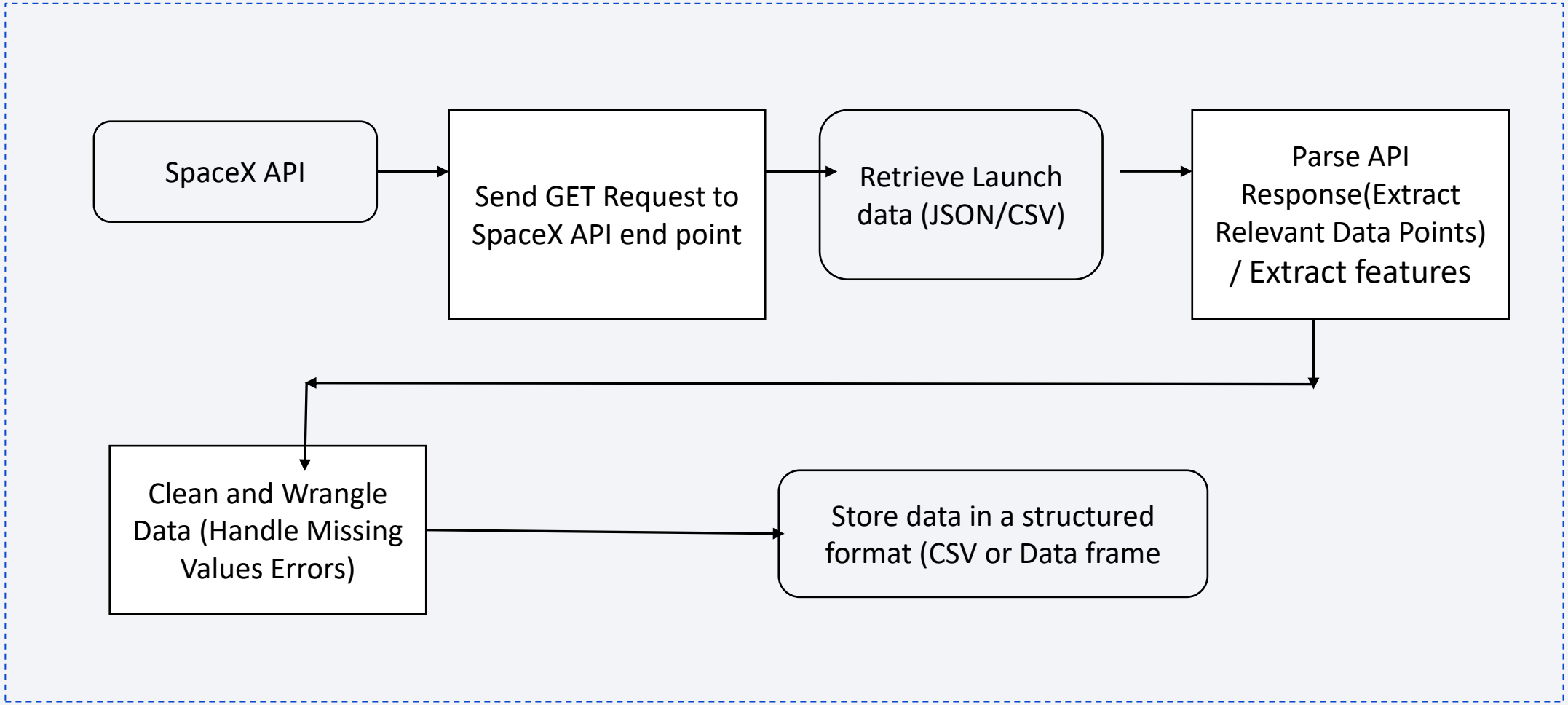
Methodology

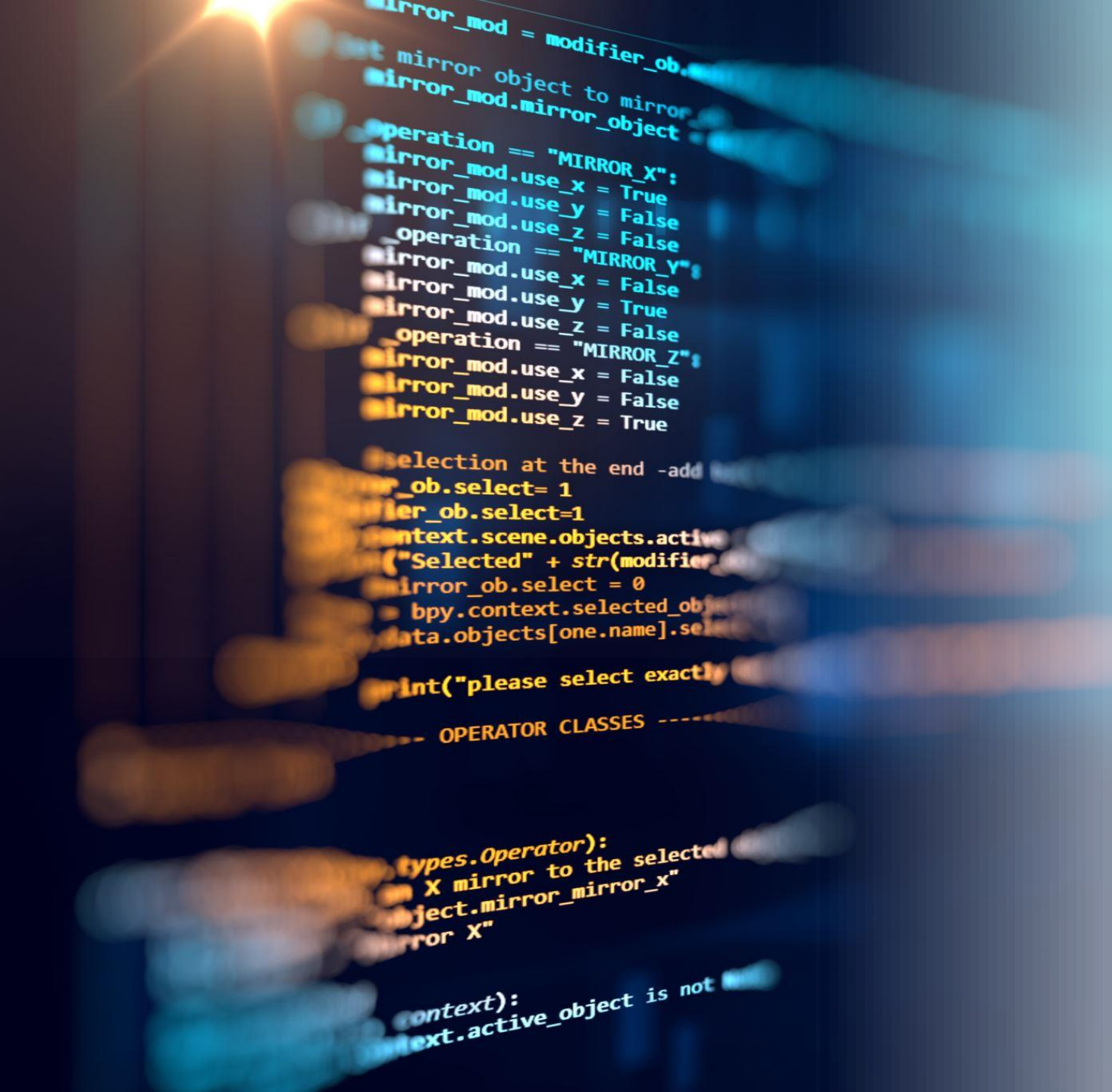
- Executive Summary
- Data collection methodology:
  - Describe how data was collected
- Perform data wrangling
  - Describe how data was processed
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive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

Data Collection

- Data for this project was collected through a GET request to the SpaceX API. The API provides access to historical data on SpaceX launches, including information relevant to predicting successful first-stage landings.

Data Collection – SpaceX API chart

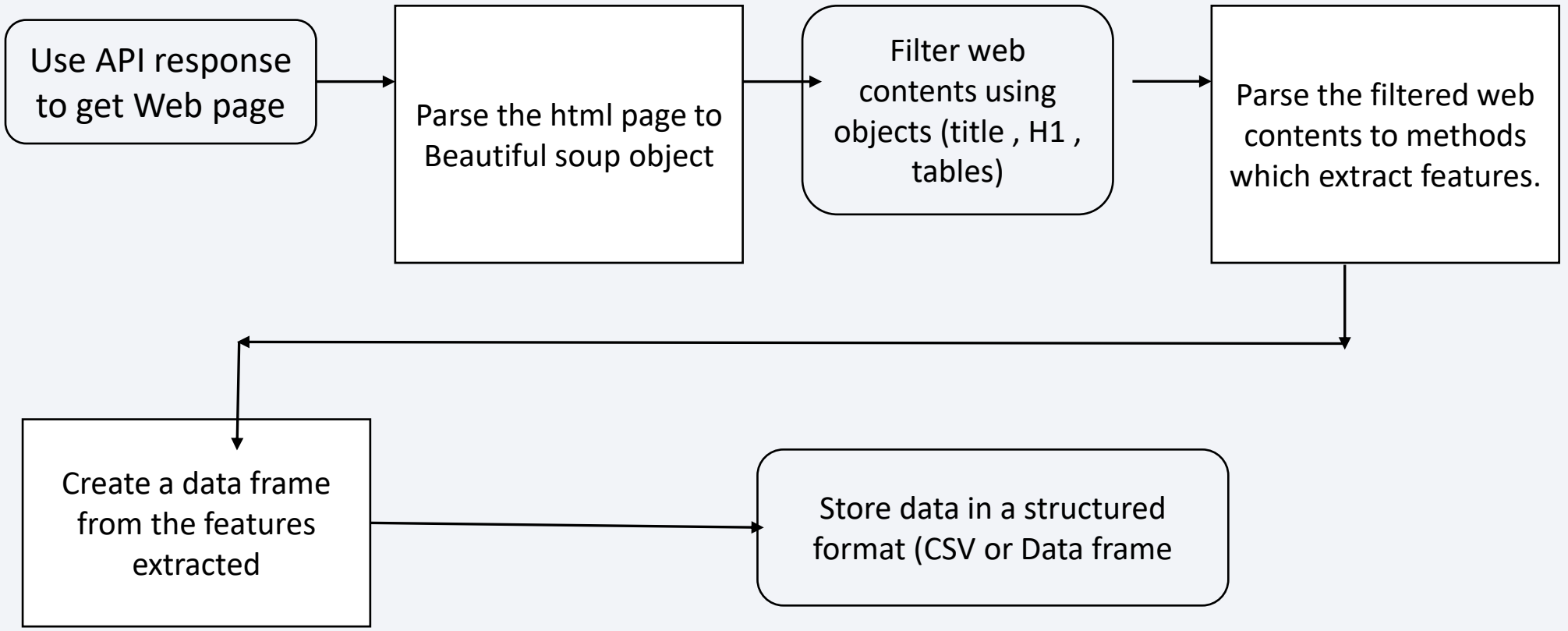




Data Collection – SpaceX API-Notebook

- Add the GitHub URL of the completed SpaceX API calls notebook (https://github.com/lmbano/Data-Science-Capstone-project/blob/main/jupyter-labs-spacex-data-collection-api.ipynb)

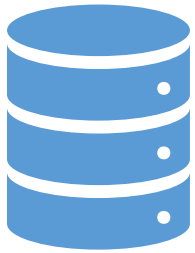
Data Collection – Scraping chart



Data Collection - Scraping

- GitHub link for the notebook : https://github.com/lmbano/Data-Science-Capstone-project/blob/main/jupyter-labs-webscraping.ipynb

Data Wrangling



Data was loaded from CSV file from previous stages and the following were performed

Viewing the first 10 rows of the data

Null checks for all variables

Data types checks

Checking the total launches per site

Checking the number of occurrence of orbits

Calculating the number of mission outcomes per orbits

Creating an Outcome label

Calculating the mean of landing class

A GitHub URL of the completed data wrangling notebook is : https://github.com/lmbano/Data-Science-Capstone-project/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

- The following chats were used in the EDA
  - Scatter plot was used to visualize relationship between variables, identify correlations and detect non-liner relationships.
  - Bar was used for categorical data frequency comparisons and patterns recognition.
  - Line chart was used to compare trends over time and changes on different categories.

- The GitHub URL of my completed EDA with data visualization notebook is : https://github.com/Imbano/Data-Science-Capstone-project/blob/main/edadataviz.ipynb

EDA with SQL

- The following sql queries were performed
  - Select count, select case, select sum(), select count(),select AVG(), Select Min(), Select max()
  - SELECT DISTINCT launch_site FROM SPACEXTABLE
  - SELECT * FROM SPACEXTABLE WHERE launch_site LIKE 'CCA%' LIMIT 5
  - SELECT Customer,sum(PAYLOAD_MASS__KG_) as total_payload_mass FROM SPACEXTABLE where Customer == 'NASA (CRS)'
  - SELECT AVG(PAYLOAD_MASS__KG_) as average_payload_mass FROM SPACEXTABLE WHERE booster_version = 'F9 v1.1'
  - SELECT MIN(Date) as first_successful_landing_date FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (ground pad)
- GitHub : https://github.com/lmbano/Data-Science-Capstone-project/blob/main/jupyter-labs-eda-sql-coursera_sqllite%20(1).ipynb

Build an Interactive Map with Folium

- The following map objects were used :
  - Circle, markers , popups ,icons
- Marker were added to pinpoint a location & provide quick identification: Allow users to quickly recognize the location.
- Popup were added to provide additional information: Offer more context about NASA Johnson Space Center.
- Enhance user engagement: Encourage users to interact with the map.
- Supplement marker information: Expand on the brief information provided by the marker.
- Icon were added to customize visual representation, Ensure the text label is clear and easy to read.
- GitHub URL : https://github.com/Imbano/Data-Science-Capstone-project/blob/main/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- A pie chart which interact with the selections from a dropdown. And display successful launches.

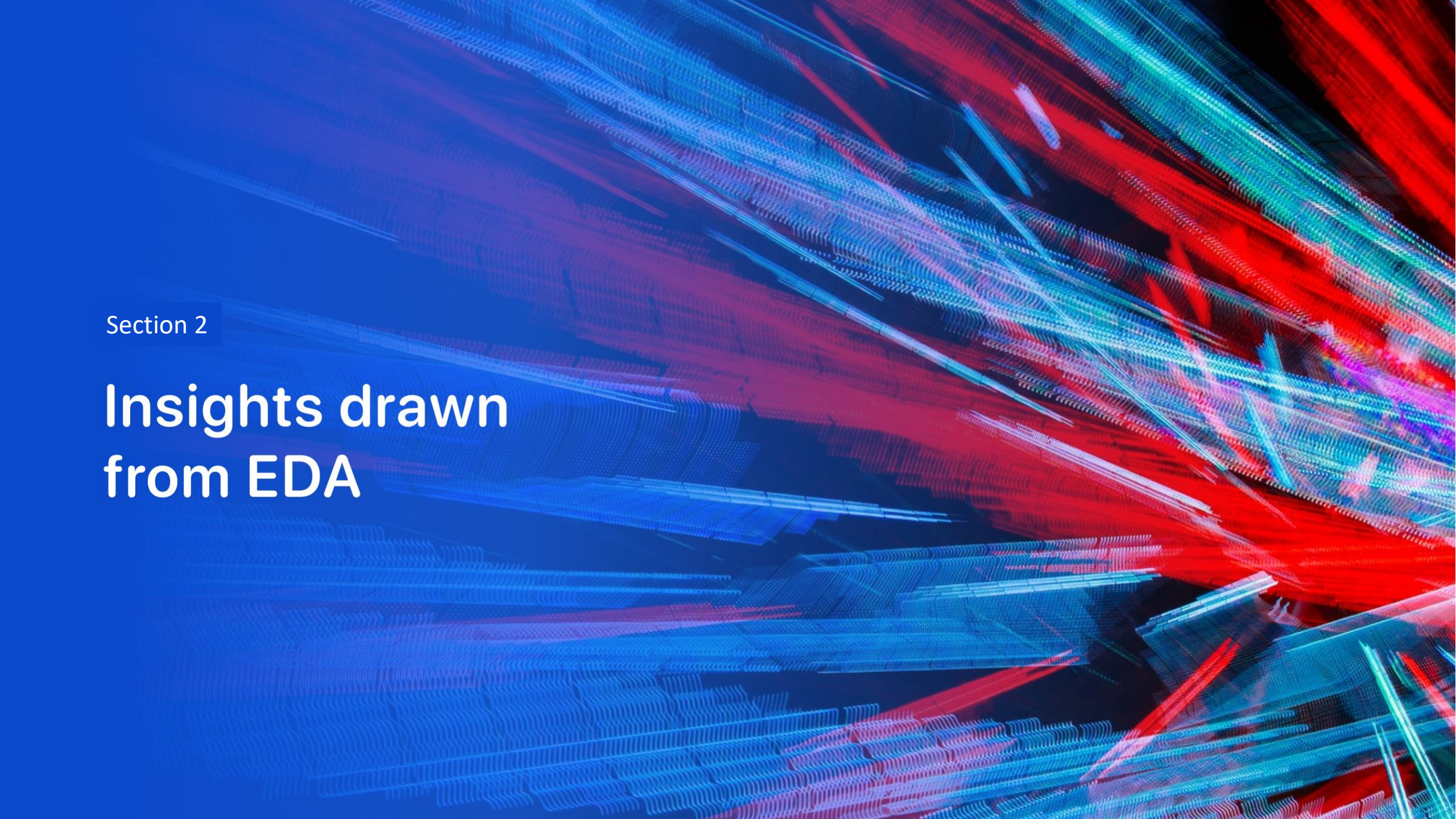
- The dropdown is to offer user interaction with the dashboard and flexibility to selections.
- GitHub URL : https://github.com/Imbano/Data-Science-Capstone-project/blob/main/spacex_dash_app%20(1).py

Predictive Analysis (Classification)

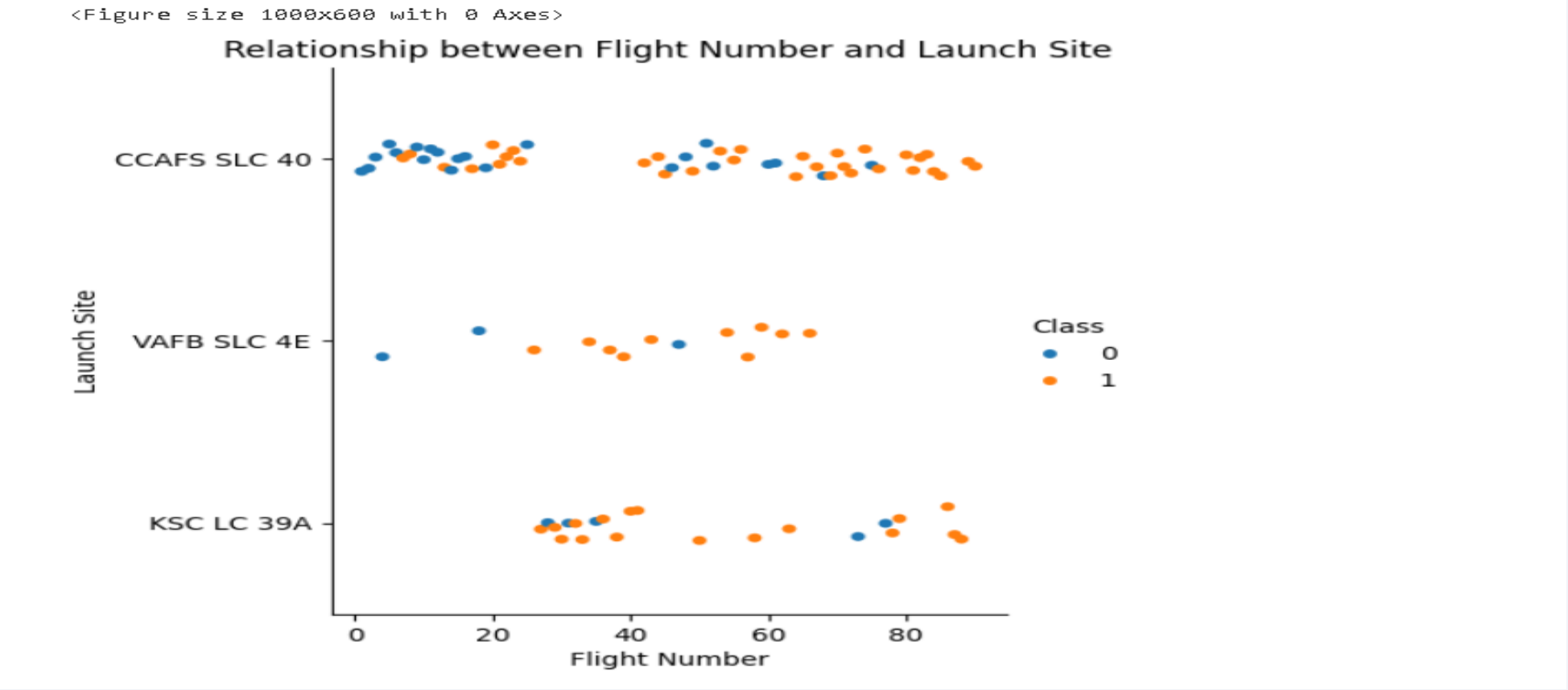
- The data was split into featres and labels , and was split into training and tesing sets using 20% as test size.
- A logisticRegression model was developed and passed to a grid search object for trained using the training data and evaluated using test data.
- GitHub URL : https://github.com/Imbano/Data-Science-Capstone-project/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Results

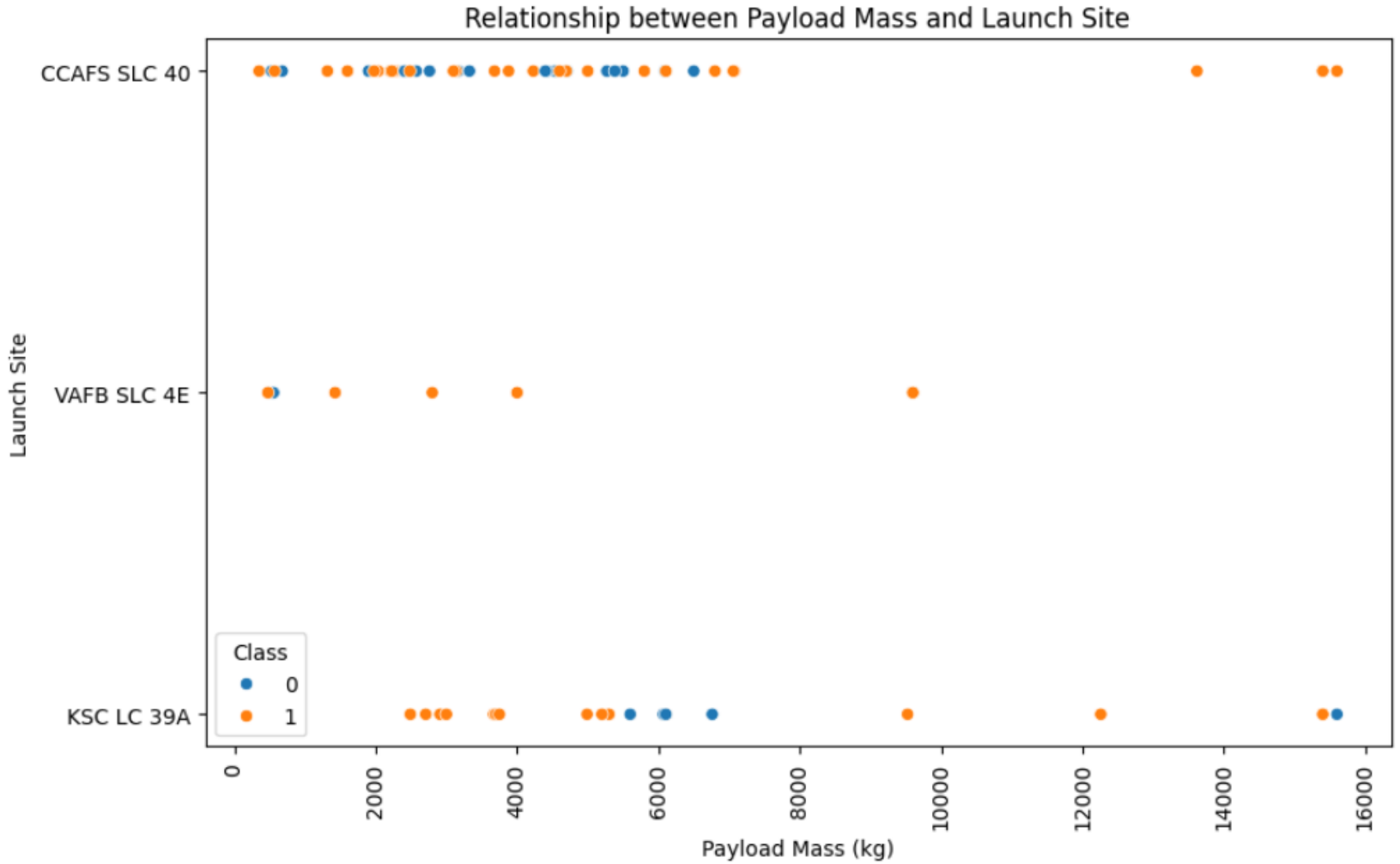
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results


Section 2
Insights drawn from EDA

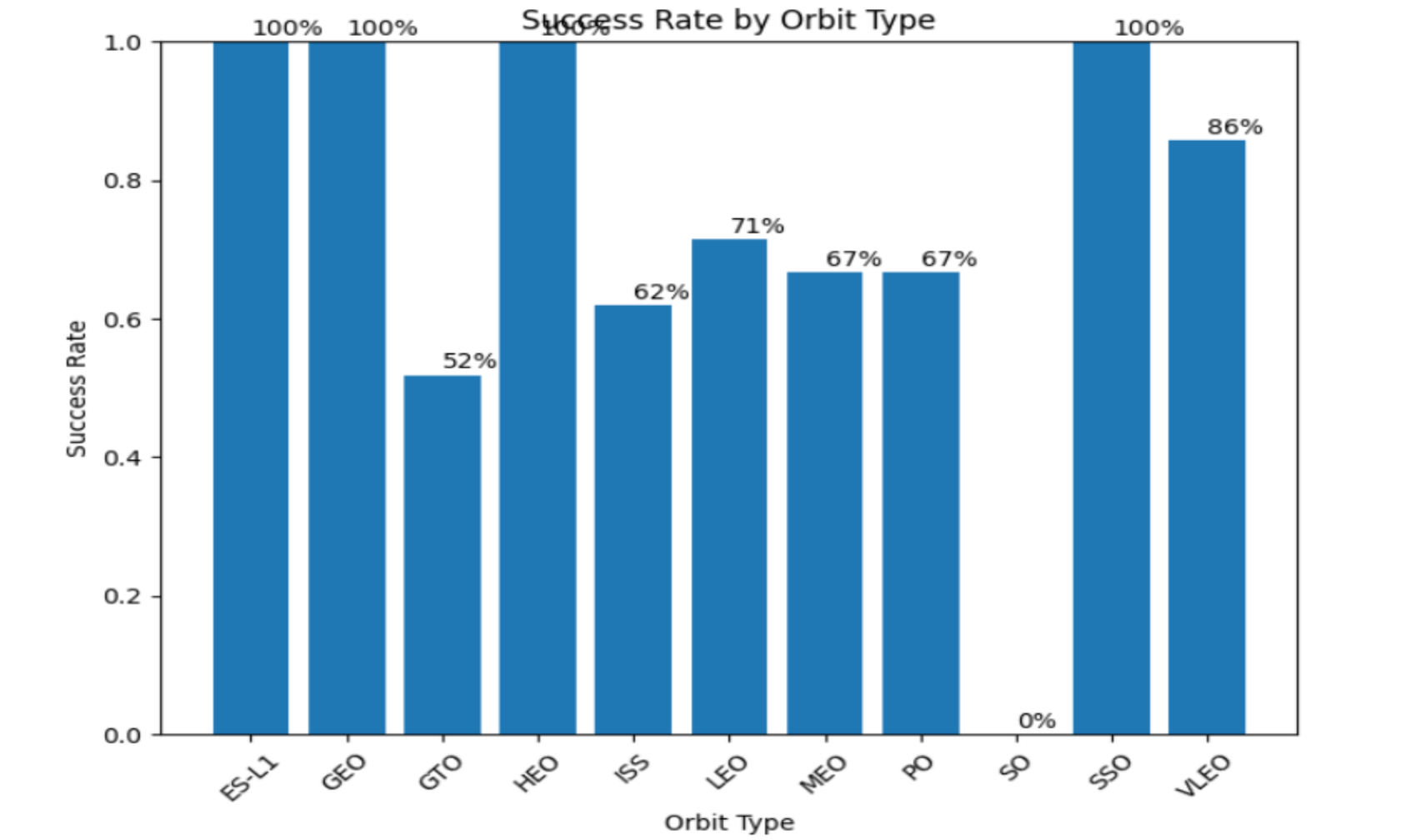
Flight Number vs. Launch Site



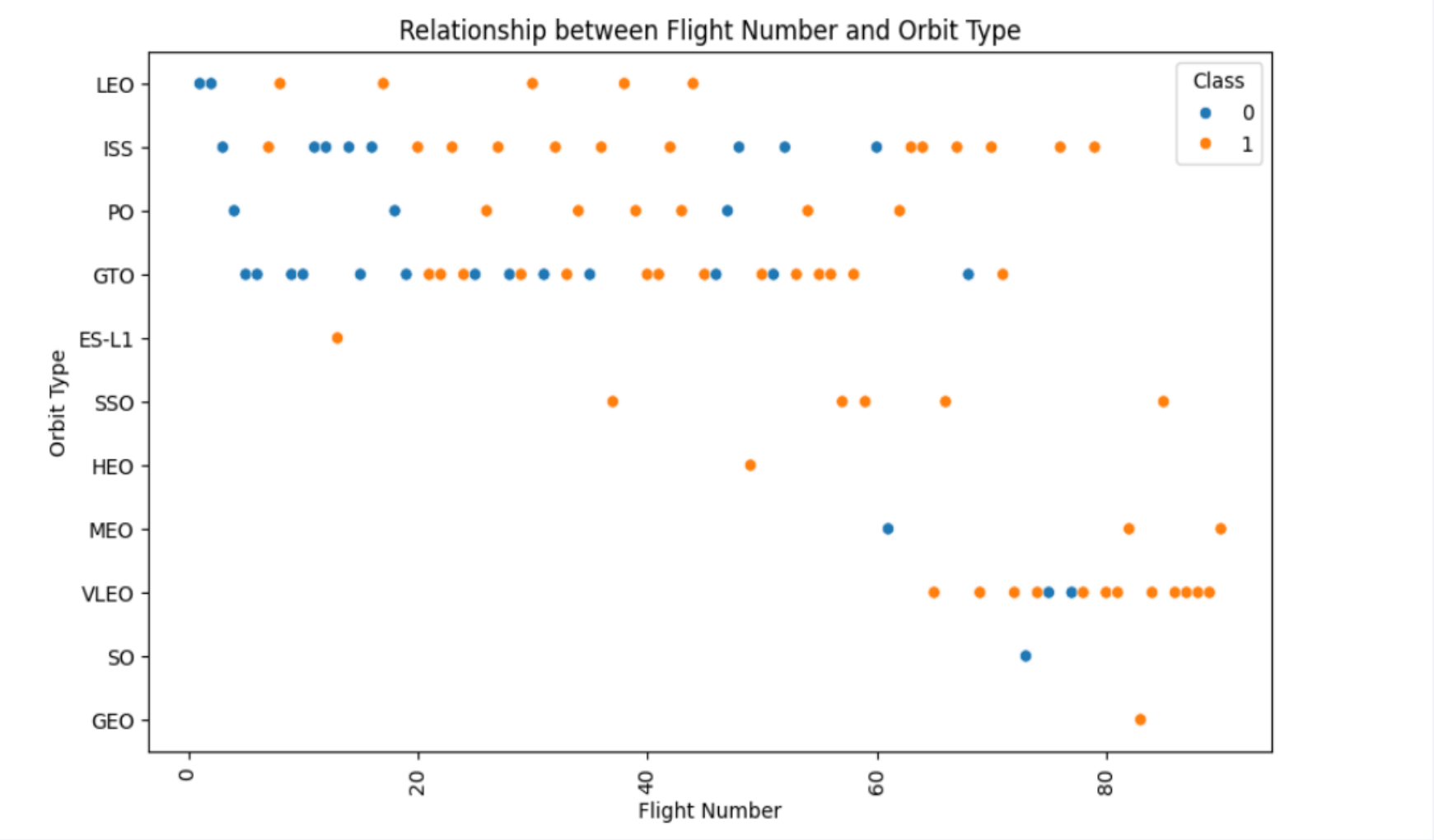
Payload vs. Launch Site



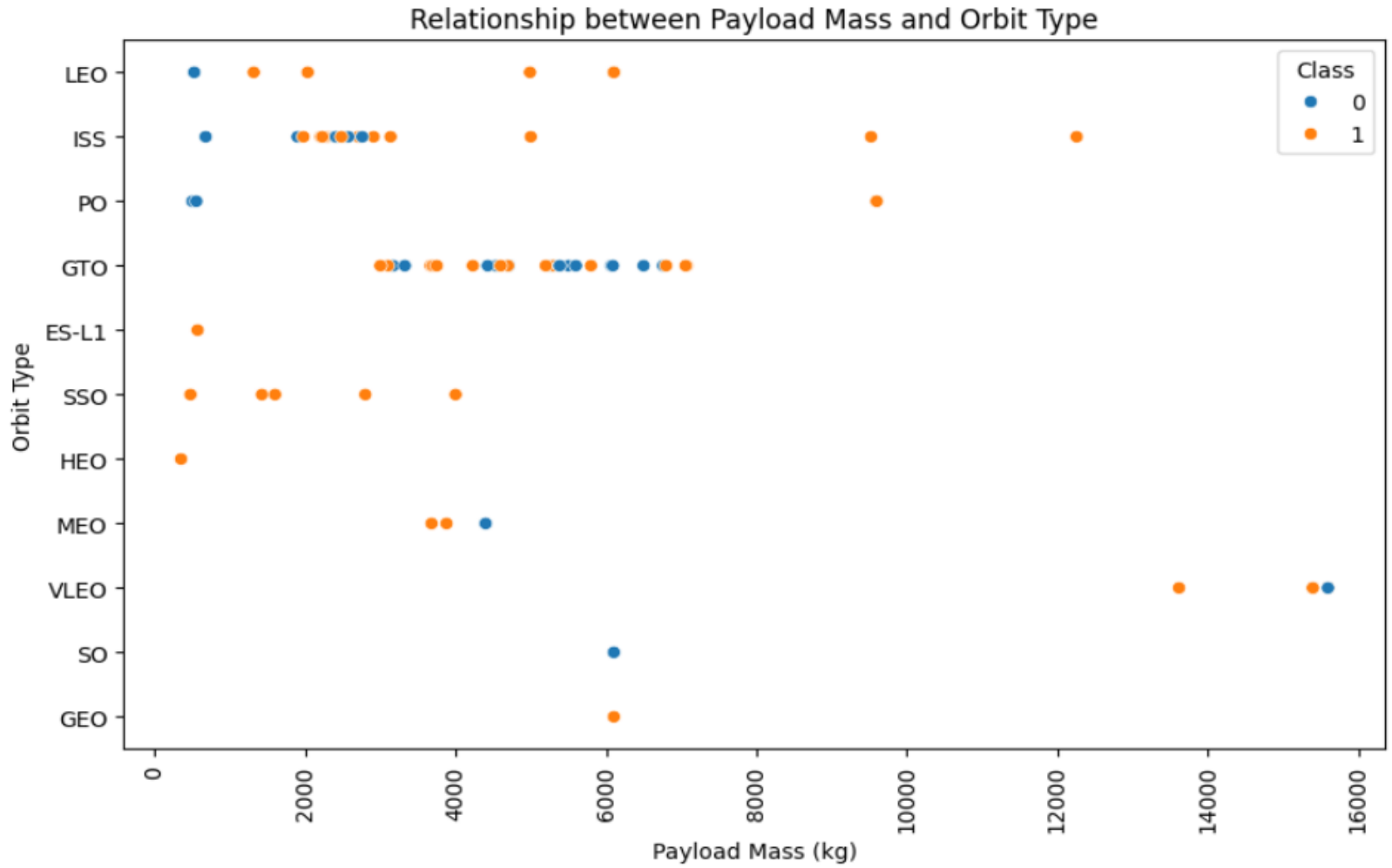
Success Rate vs. Orbit Type



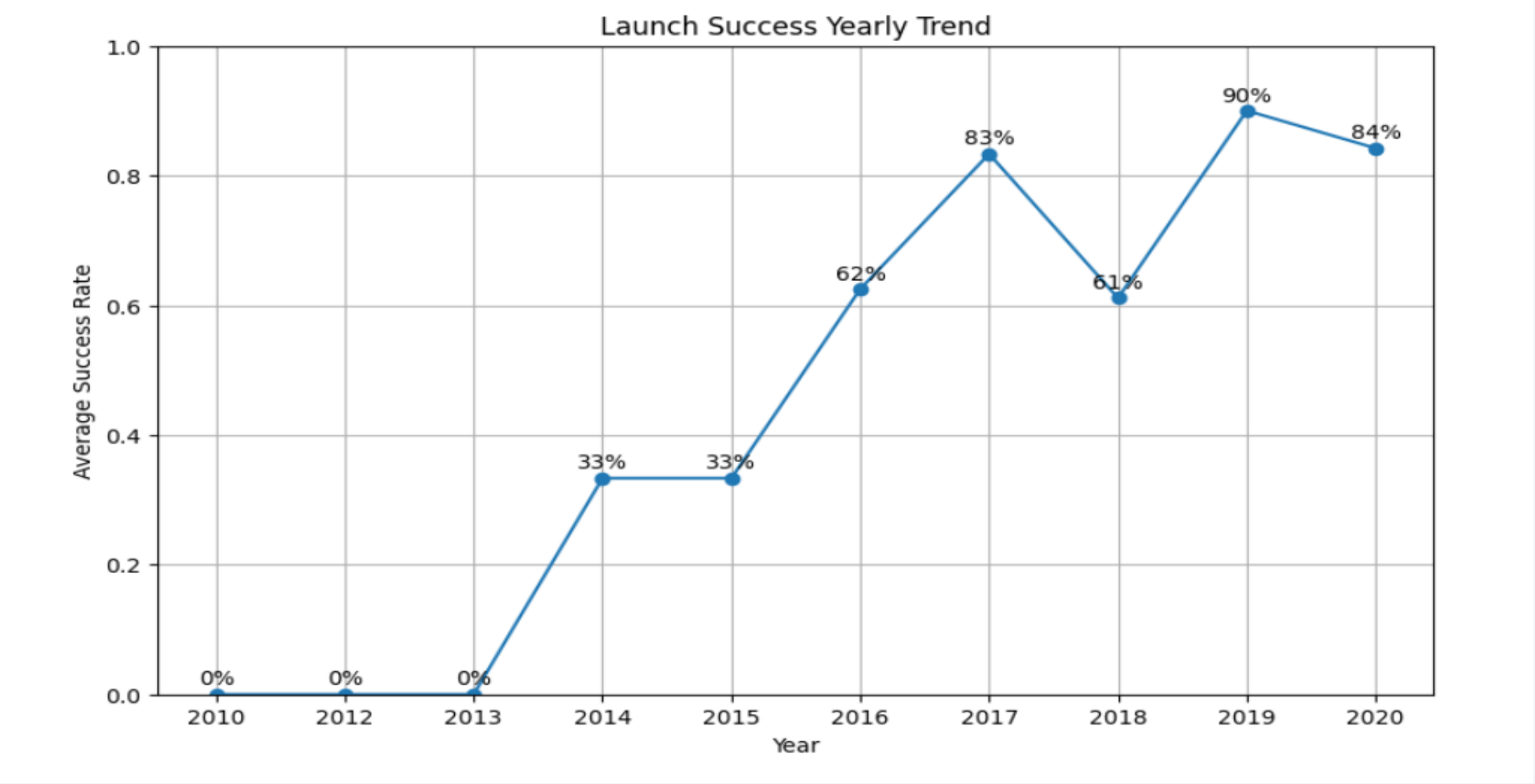
Flight Number vs. Orbit Type



Payload vs. Orbit Type



Launch Success Yearly Trend



All Launch Site Names

```
# Query unique launch site names
cur.execute("SELECT DISTINCT launch_site FROM SPACEXTABLE")

# Fetch all results
unique_launch_sites = cur.fetchall()

# Display unique launch site names
for launch_site in unique_launch_sites:
    print(launch_site[0])
```

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

Launch Site Names Begin with 'CCA'

Task 2

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

```
cur.execute("""
    SELECT *
    FROM SPACEXTABLE
    WHERE launch_site LIKE 'CCA%'
    LIMIT 5
""")

records = cur.fetchall()

for record in records:
    print(record)
```

```
('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', 'Succes
s', '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

Task 3

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

```
cur.execute("""
    SELECT Customer,sum(PAYLOAD_MASS__KG_) as total_payload_mass
    FROM SPACEXTABLE
    where Customer == 'NASA (CRS)'
""")

total_payload_mass = cur.fetchall()

total_payload_mass
```

```
[('NASA (CRS)', 45596)]
```

```
results
```

```
[('NASA (CRS)', 45596)]
```

Average Payload Mass by F9 v1.1

Task 4

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

```
cur.execute("""
    SELECT AVG(PAYLOAD_MASS__KG_) as average_payload_mass
    FROM SPACEXTABLE
    WHERE booster_version = 'F9 v1.1'
""")

average_payload_mass = cur.fetchall()
average_payload_mass
```

```
[(2928.4,)]
```

First Successful Ground Landing Date

Task 5

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

Hint:Use min function

```
cur.execute("""
    SELECT MIN(Date) as first_successful_landing_date
    FROM SPACEXTABLE
    WHERE Landing_Outcome = 'Success (ground pad)'
    """)

first_successful_landing_date = cur.fetchone()[0]

print("First Successful Landing Date (Ground Pad):", first_successful_landing_date)
```

```
First Successful Landing Date (Ground Pad): 2015-12-22
```

Successful Drone Ship Landing with Payload between 4000 and 6000

Task 6

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

```
]: cur.execute("""
    SELECT Booster_Version
    FROM SPACEXTABLE
    WHERE landing_outcome = 'Success (drone ship)'
     AND PAYLOAD_MASS__KG_ > 4000
    AND PAYLOAD_MASS__KG_ < 6000
""")

boosters = cur.fetchall()

for booster in boosters:
    print(booster[0])
```

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

Total Number of Successful and Failure Mission Outcomes

```
cur.execute("""
    SELECT
        SUM(CASE WHEN landing_outcome  LIKE 'success%' THEN 1 ELSE 0 END) AS successful_missions,
        SUM(CASE WHEN landing_outcome like 'Failure%' THEN 1 ELSE 0 END) AS failed_missions
    FROM SPACEXTABLE
""")

results = cur.fetchone()

print("Successful Missions:", results[0])
print("Failed Missions:", results[1])
```

```
Successful Missions: 61
Failed Missions: 10
```

Boosters Carried Maximum Payload

Task 8

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

```
cur.execute("""
    SELECT Booster_Version
    FROM SPACEXTABLE
    WHERE PAYLOAD_MASS__KG_ = (
        SELECT MAX(PAYLOAD_MASS__KG_)
        FROM SPACEXTABLE
    )
""")

max_payload_boosters = cur.fetchall()

for booster in max_payload_boosters:
    print(booster[0])
```

```
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 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.

```
cur.execute("""
    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,
        Landing_Outcome,
        Booster_Version,
        Launch_Site
    FROM
        SPACEXTABLE
    WHERE
        substr(Date, 0, 5) = '2015'
        AND landing_outcome = 'Failure (drone ship)'

""")

records = cur.fetchall()

for record in records:
    print(record)
```

```
('January', 'Failure (drone ship)', 'F9 v1.1 B1012', 'CCAFS LC-40')
('April', 'Failure (drone ship)', 'F9 v1.1 B1015', 'CCAFS LC-40')
```

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

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.

```
cur.execute("""
    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
""")

landing_outcomes = cur.fetchall()

for i, outcome in enumerate(landing_outcomes):
    print(f"Rank {i+1}: {outcome[0]} - {outcome[1]}")
```

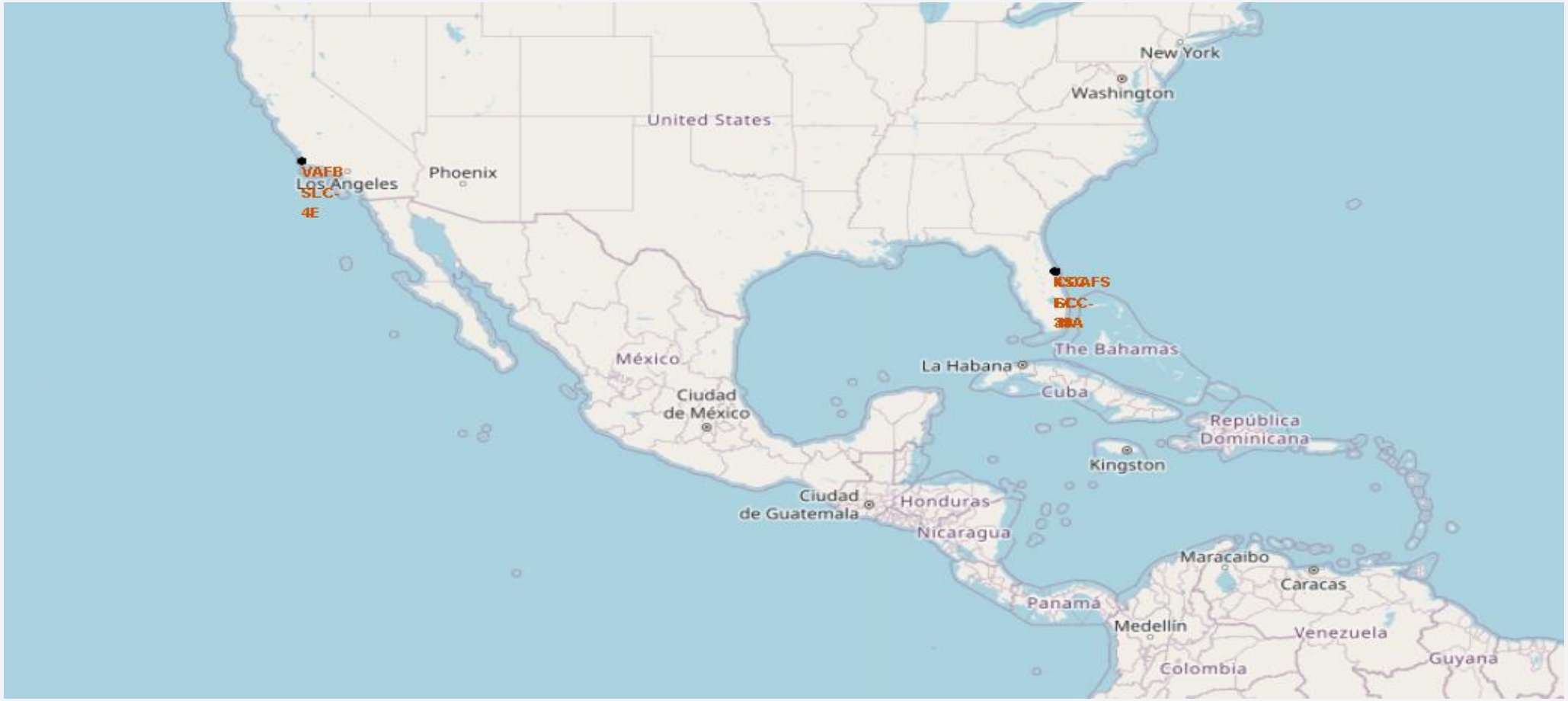
```
Rank 1: No attempt - 10
Rank 2: Success (drone ship) - 5
Rank 3: Failure (drone ship) - 5
Rank 4: Success (ground pad) - 3
Rank 5: Controlled (ocean) - 3
Rank 6: Uncontrolled (ocean) - 2
Rank 7: Failure (parachute) - 2
Rank 8: Precluded (drone ship) - 1
```

Section 3

Launch Sites Proximities Analysis

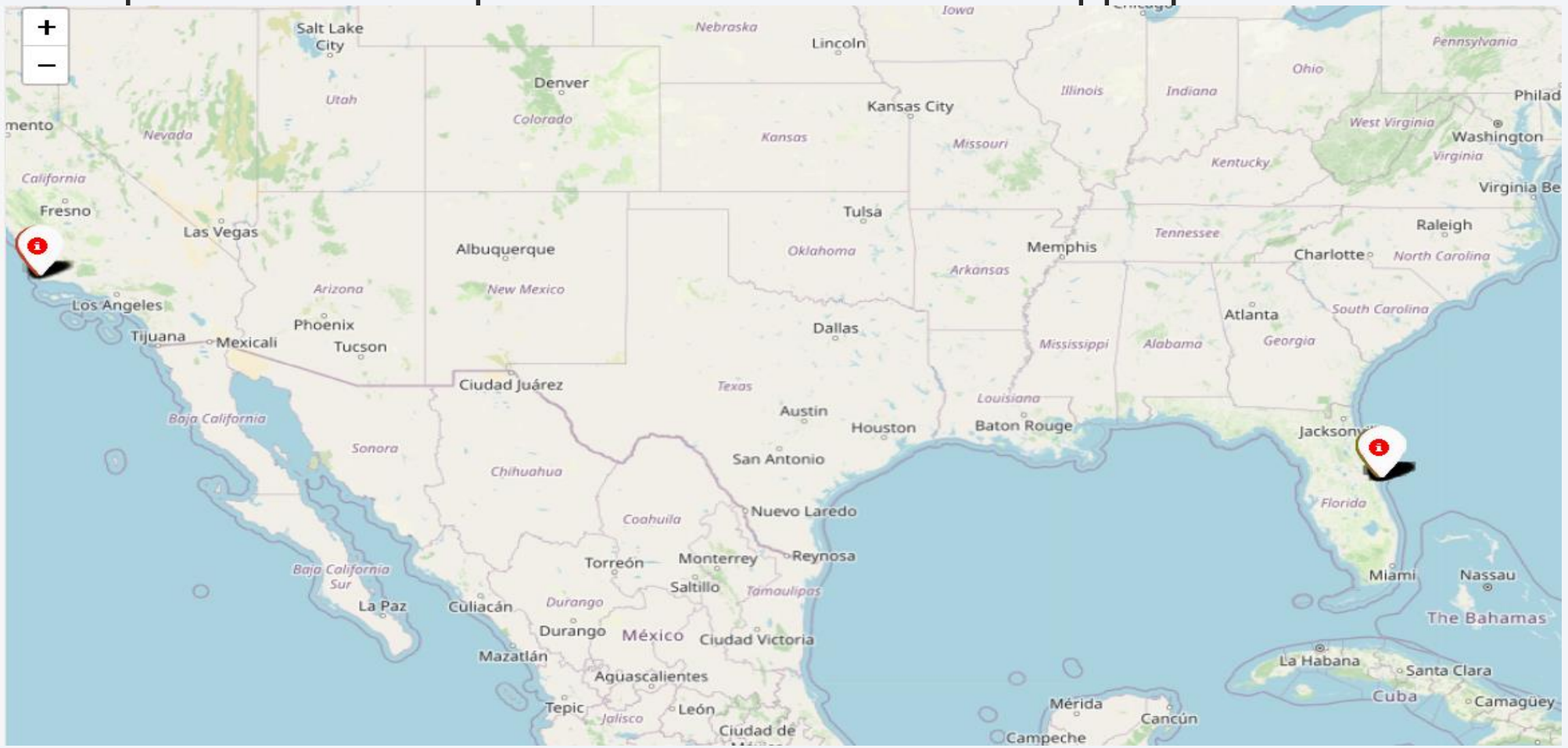
NASA Johnson Space Center at Houston, Texas Map

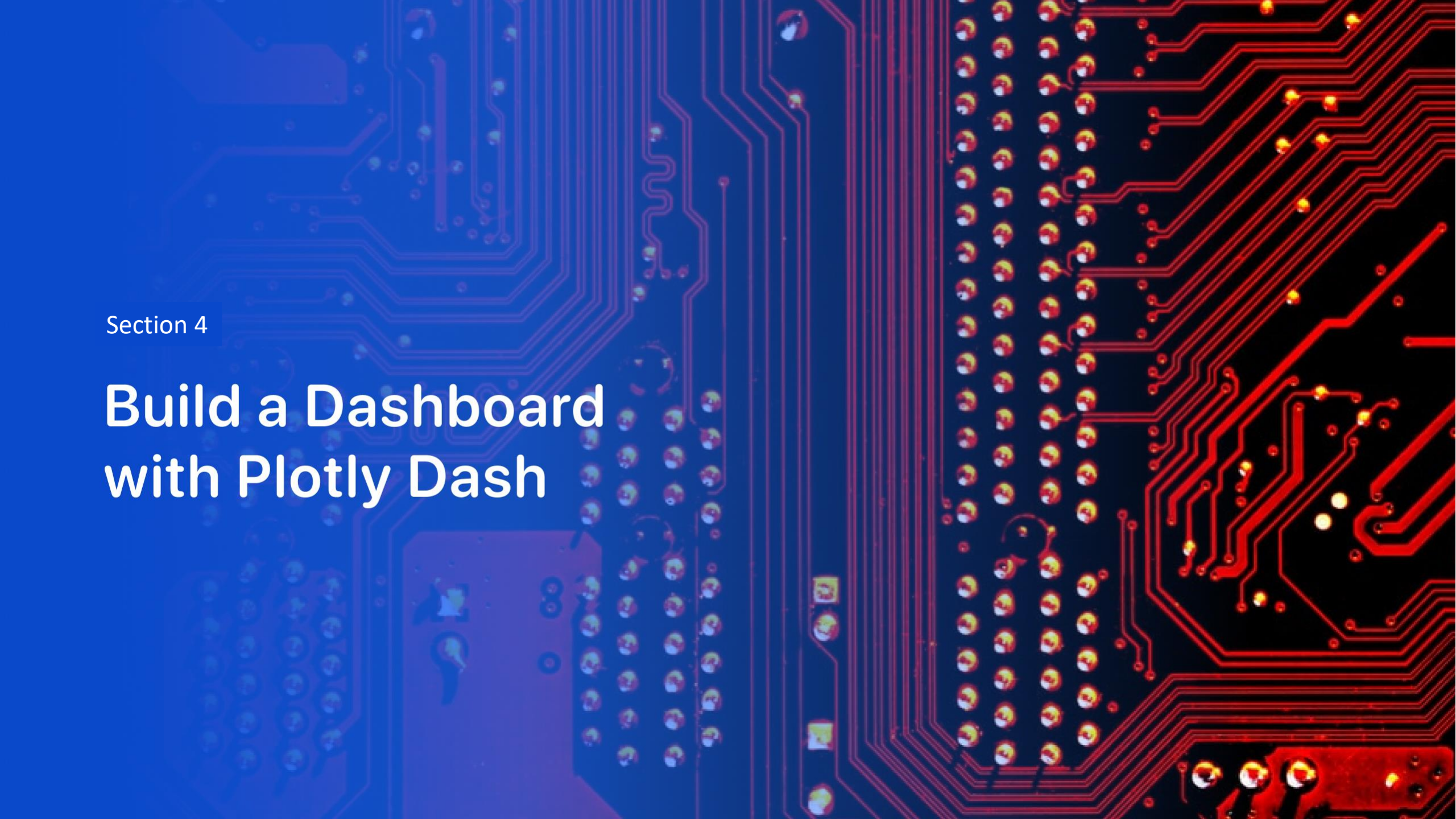
Railway distances



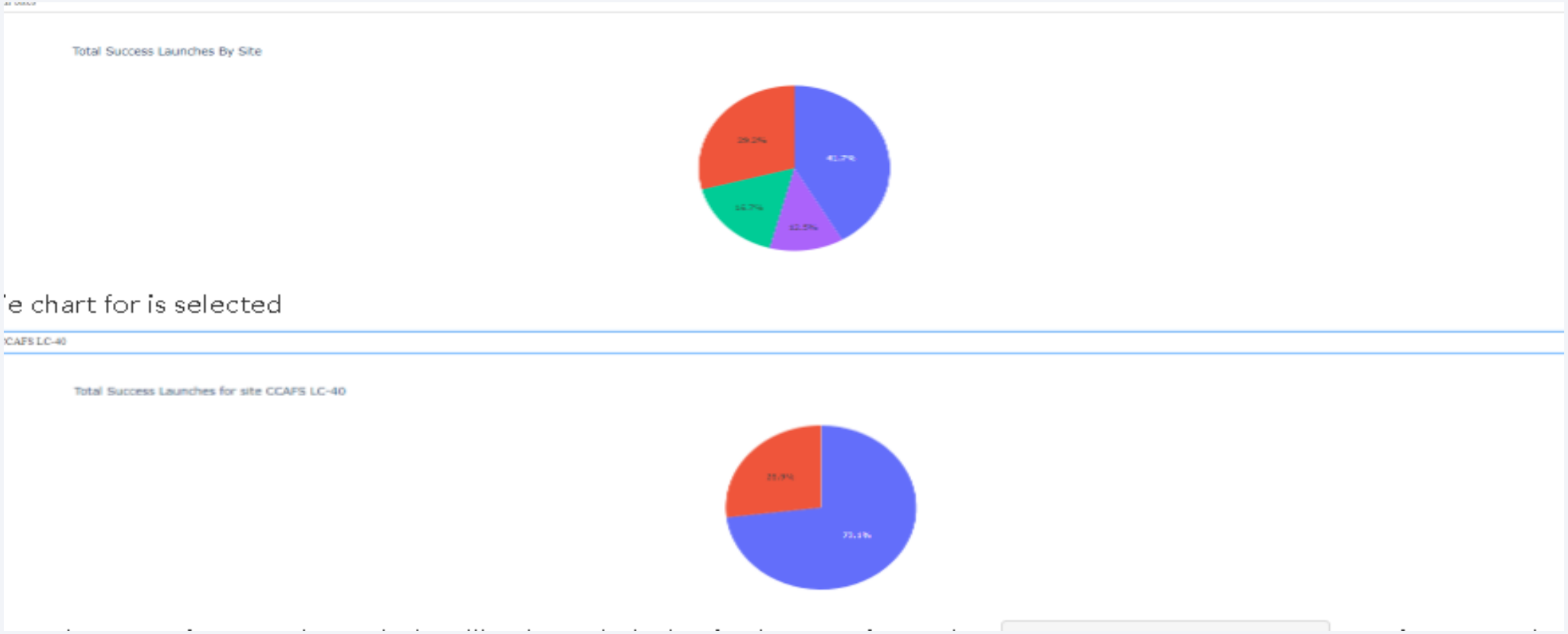
Mapping

- Replace <Folium map screenshot 3> title with an appropriate title

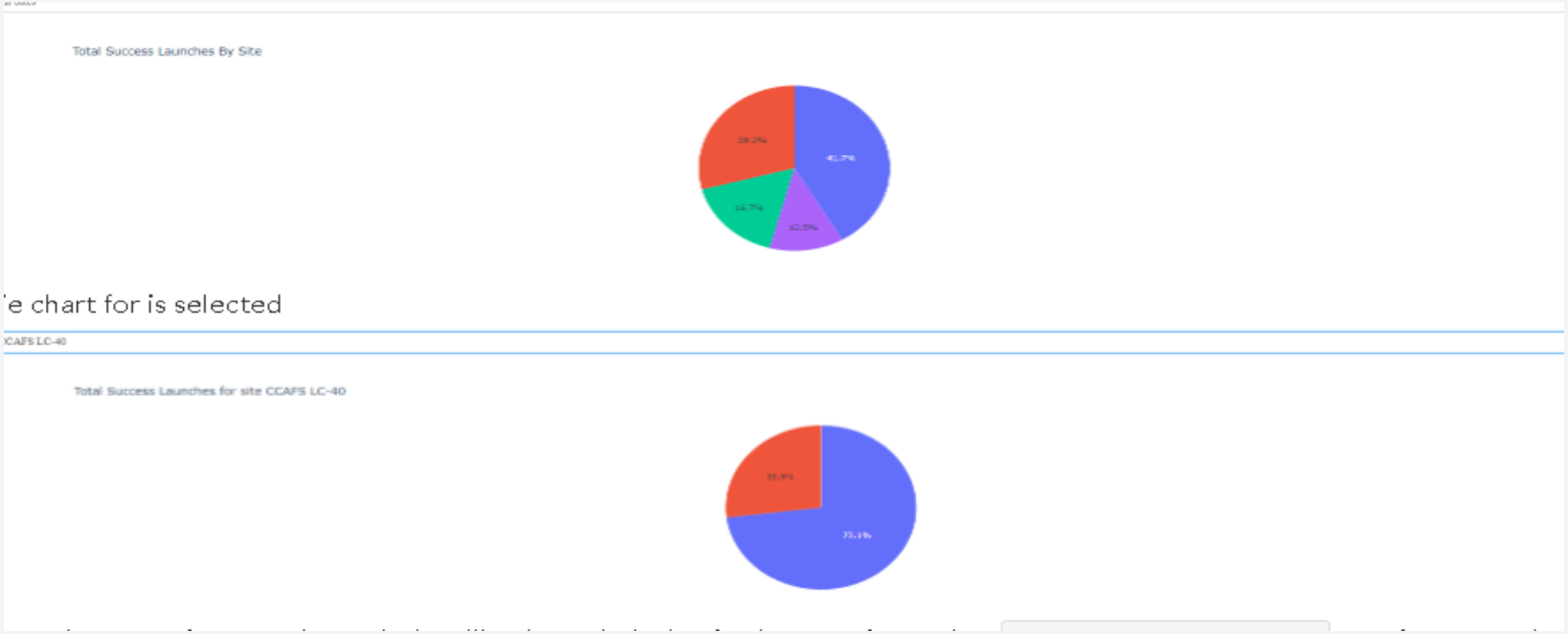



Section 4
Build a Dashboard with Plotly Dash

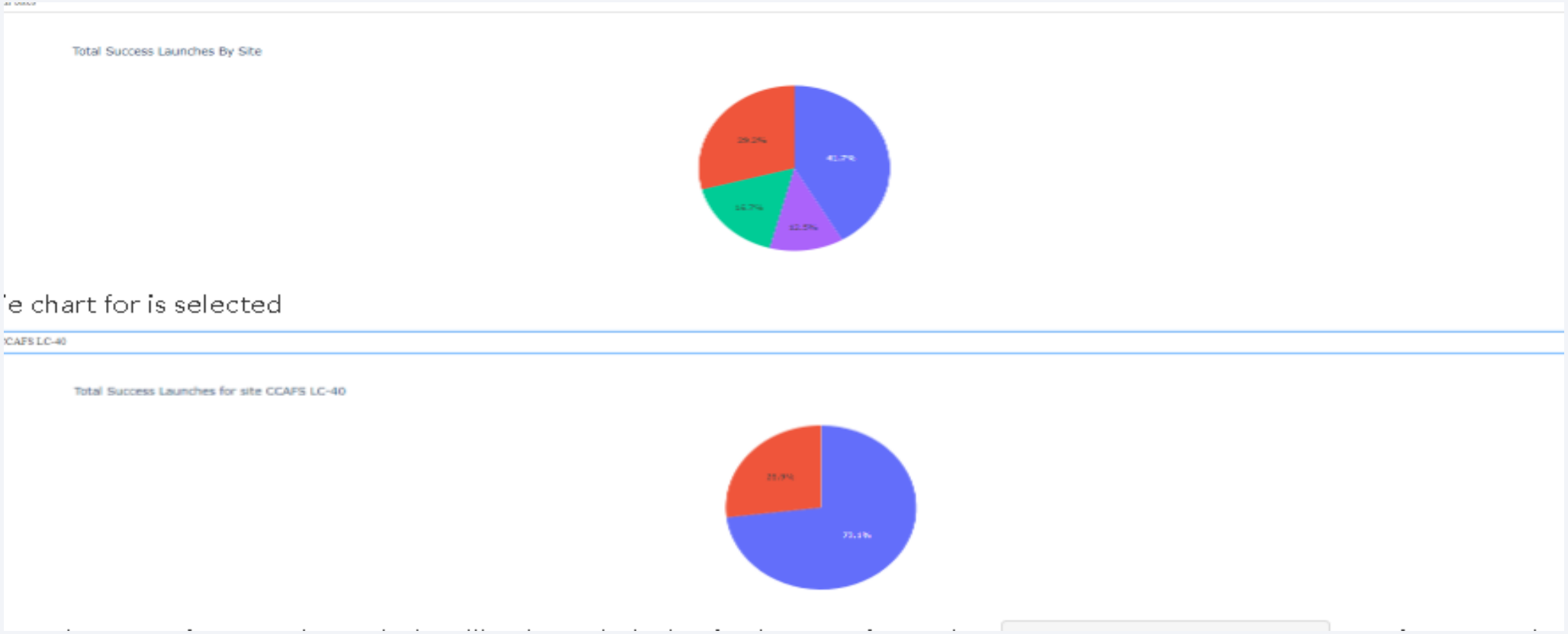
launch success count for all sites



Highest launch success



<Dashboard Screenshot 3>



Section 5

Predictive Analysis (Classification)

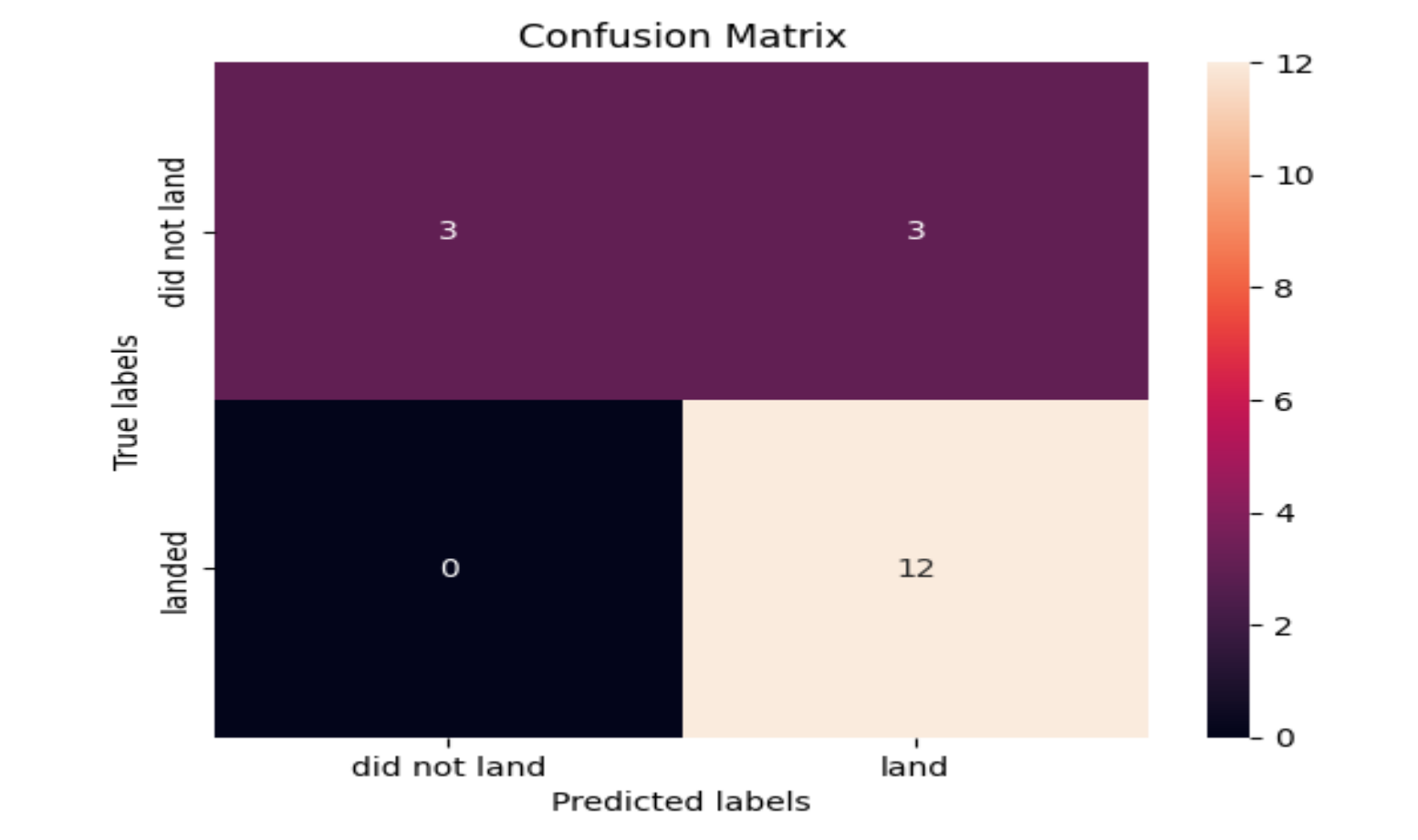
Classification Accuracy

Find the method performs best:

```
0]:
# Accuracy scores
logreg_accuracy = logreg_cv.score(X_test, Y_test)
svm_accuracy = svm_cv.score(X_test, Y_test)
tree_accuracy = tree_cv.score(X_test, Y_test)
knn_accuracy = knn_cv.score(X_test, Y_test)

# Print accuracy scores
print("Logistic Regression Accuracy: ", logreg_accuracy)
print("SVM Accuracy: ", svm_accuracy)
print("Decision Tree Accuracy: ", tree_accuracy)
print("KNN Accuracy: ", knn_accuracy)

# Find the best-performing method
best_method = max({"Logistic Regression": logreg_accuracy,
                   "SVM": svm_accuracy,
                   "Decision Tree": tree_accuracy,
                   "KNN": knn_accuracy}, key=dict.get)

print("Best-performing method: ", best_method)
```

```
Logistic Regression Accuracy:  0.8333333333333334
SVM Accuracy:  0.8333333333333334
Decision Tree Accuracy:  0.8333333333333334
KNN Accuracy:  0.8333333333333334
```

Confusion Matrix



Conclusions

- The models were able to predict well on the data.

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!