IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Judith Boluwatito TOBOGBE
12/04/2023
Judithboluwatito@gmail.com

Outline

Executive Summary

Summary of methodologies :

- Data Collection: The first step was to collect data using a Get request to the SpaceX API. The data was then subjected to data wrangling and formatting to ensure that it was in a usable format.
- Exploratory Data Analysis (EDA): EDA was performed to determine the training labels. This involves analyzing the data to find patterns, trends, and relationships that can be used to build models.
- SQL Queries: SQL queries were executed to gain better insights into the datasets stored in the database. This helps to identify trends and patterns in the data that may not be immediately apparent.
- Feature Engineering: EDA and feature engineering were performed using Pandas and Matplotlib to further refine the data for modeling. This involves transforming the data to create new features that may be more informative or useful for building models.
- Location Analysis: Geographical patterns about launch sites were found using Folium, which is a Python library used for visualizing geospatial data.
- Dashboard Visualization: Interactive real-time dashboards were built using Plotly Dash to visualize the data and provide insights into trends and patterns.
- Classification Modeling: Finally, classification models were built, trained, and tested to determine the best performance. This involves applying machine learning algorithms to the data to predict outcomes based on a set of input features.
- **Summary of all results**
- Data Collection: The data was collected, cleaned, formatted, and exported to a CSV file.
- Data Analysis: The data was analyzed, labeled with dependent and target variables, and split into training and testing sets. Maps, charts, and plots were created to gain insights into launch sites, landing success rates, payload mass, and booster versions.
- Machine Learning Models: Several machine learning models were developed and evaluated to determine the best models, best accuracy, and confusion matrix. The goal of these models was to predict outcomes based on input variables and provide insights into the relationships between variables in the dataset.

Introduction

SpaceX has emerged as a leader in the commercial space travel industry by making space travel more affordable through their innovative use of reusable rocket technology. One key factor that contributes to SpaceX's success in achieving this goal is their ability to reuse the first stage of their rockets, which significantly reduces the cost of launches. As a company in the same industry, SpaceY aims to understand and replicate this success. Therefore, this project will use SpaceX as a case study to analyze their approach to launching rockets, specifically focusing on the success of their first stage landings.

The primary objective of this project is to predict the success of future Falcon 9 launches based on data obtained from an API. We will collect, clean, and analyze the data, identifying trends and insights to help us build several classification models. By doing so, we aim to develop an accurate and reliable model that can predict whether the first stage of a Falcon 9 rocket will land successfully. Through this project, we hope to gain insights into the factors that contribute to successful first stage landings, and to apply these insights to inform our own approach to rocket launches as a company in the commercial space travel industry.

Section 1

Methodology

Methodology

Executive Summary

- Send a GET request to SpaceX's API and scrape the Falcon 9 launch table from its Wikipedia URL page
- Perform data wrangling, cleaning, and formatting
- Perform exploratory data analysis (EDA) using visualization and SQL techniques
- Load the data into tables in a DB2 database and execute SQL queries to better understand the dataset
- Build a Folium map object to display launch site coordinates and markers showing proximities to coastlines, railroads, highways, and cities
- Use Plotly Dash to build an interactive dashboard that displays SpaceX data in real time
- Perform EDA to determine the training labels and create a column for the target variable ("Class")
- Standardize the data, split it into training and test datasets, and use classification models to test for accuracy
- Determine the best performing model and use it to predict the success of future Falcon 9 launches

Data Collection

- Describe how data sets were collected.
- Import Libraries and define functions
- Request rocket launch data from SpaceX API with URL
- Request & Parse SpaceX launch data using GET Request
- Decode the data and turn it into a Pandas dataframe
- Filter the dataframe to only include our target variable
- Deal with missing values and replace them
- Export the cleaned data into CSV
- https://github.com/Judithboluwatito/SPACEX-DATA-COLLECTION-API

Data Collection - WebScraping

Scrapping Falcon9 Launch table from its Wiki URL page and parsing it to a dataframe

- Import libraries
- Define functions to scrape HTML table
- Request the falcon 9 Launch wiki page from its URL
- Extract all Columns / Variable names from HTML table header
- Create a dataframe by parsing the launch HTML table
- Export the table to CSV
- Add the GitHub URL of the completed web scraping notebook, as an external reference and peer-review purpose
- https://github.com/Judithboluwatito/2-SPACEX-WEBSCRAPING/blob/main/2-SPACEX-WEBSCRAPING.ipynb

Data Wrangling

- The task at hand involves performing exploratory analysis and determining training tables. The following steps are involved in this process:
- Import necessary libraries and define auxiliary functions.
- Load the dataset and clean the data.
- Calculate the number of launches that have taken place on each site.
- Calculate the number of occurrences of each orbit.
- Calculate the number of occurrences of mission outcomes per orbit type.
- Create a landing outcome label from the outcome column to be used as the target categorical variable.
- Export the analysed data into CSV format.
- You need to present your data wrangling process using key phrases and flowcharts
- Add the GitHub URL of your completed datha wrangling related notebooks, as an external reference and peer-review purpose
- https://github.com/Judithboluwatito/SPACEX-DATA-WRANGLING/blob/main/3-SPACEX-DATA%20WRANGLING-SkillsNetwork_labs_module_1_L3_labs-jupyter-spacex-data_wrangling_jupyterlite.jupyterlite.ipynb

EDA with Data Visualization

PERFORMING EXPLORATORY DATA ANALYSIS (EDA) AND FEATURE ENGINEERING USING PANDAS AND MATPLOTLIB

- Import required libraries and define any auxiliary functions.
- Load the dataset and perform EDA to visualize trends in the data.
- Visualize the relationship between flight number and launch site.
- Visualize the relationship between payload and launch site.
- Visualize the relationship between success rate of each orbit type.
- Visualize the relationship between flight number and orbit type.
- Visualize the relationship between payload and orbit type.
- Visualize the launch success yearly trend.
- Create a dummy variable for feature engineering using one-hot encoding.
- Convert all numeric columns to float64 data type.
- Export the final data to a CSV file.

https://github.com/Judithboluwatito/EDA/blob/main/5-SPACEX%20EXPLORING%20AND%20PREPARING%20DATA%20killsNetwork_labs_module_2_jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

EDA with SQL

- **Understanding the SpaceX dataset, loading the data into the tables in DB2 database and executing SQL queries to understand the SpaceX dataset**
- Download the SpaceX dataset and connect to the DB2 database.
- Display the names of unique launch sites to explore the data.
- Display the records of launch sites with CCA.
- Display the total payload mass carried by boosters launched by NASA (CRS).
- Display the average payload mass carried by booster version F9 V1.1.
- List the date when the first successful landing outcome in the past was achieved.
- Display the names of boosters with success in drone ships with payload between 4000-6000.
- List the total number of successful and failure mission outcomes.
- Display the names of booster versions that have carried the maximum payload mass.
- Query the drone failure outcome, booster version, and launch site for 2015.
- Rank the successful landing outcomes from June 2010 to March 2017.
- **Add the GitHub URL of your completed EDA with SQL notebook, as an external reference and peer-review purpose**
- https://github.com/Judithboluwatito/SQL-SPACEX-LAB/blob/main/SPACEX%20SQL%20LAB%20.ipynb

Build an Interactive Map with Folium

Launch Site Location Analysis using Folium Maps

- Build a Folium map object with NASA launch site coordinates as the center.
- Use the folium marker object to mark all launch sites on the map.
- Mark successful and failed mission sites on the map using a different marker.
- Calculate the distance between launch site locations and their closest proximity to highways, railroads, coastlines, and cities.
- Use the insights obtained from the map to draw conclusions about the suitability of launch site locations.
- https://github.com/Judithboluwatito/Foliunm-maps-/blob/main/6-SPACEX%20IBM-DS0321EN-SkillsNetwork_labs_module_3_lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

Develop a real-time interactive dashboard to visualize the SpaceX data using Plotly Dash

- Create an input component for the dashboard with dropdown lists and range sliders to display the pie chart and scatter point chart
- Include a dropdown input component to select the launch site
- Implement a callback function that generates a "Success pie chart" based on the selected launch site dropdown
- Include a range slider to select the payload
- Add a callback function to display the "Success payload scatter plot" based on the selected payload range
- Launch the interactive web dashboard using a private IP and port: 127.0.0.1 / 8050.
- **https://github.com/Judithboluwatito/SPACEX-DASH-INTERACTIVE-/blob/main/SPACEX%20Dash%20Interactive%20.ipynb**

Predictive Analysis (Classification)

Perform exploratory data analysis to determine the training labels, standardize the data, split it into training and test sets for classification, and test the models for accuracy to find the best performing model:

- Import the required libraries and load the dataset.
- Define a function to plot the confusion matrix.
- Create a NumPy array with the "Class" column and assign it to the variable Y.
- Standardize the data in X and assign it to the variable X.
- Split the data into training and testing sets.
- Create a logistic regression object and a GridSearchCV object to find the best parameters, which are {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}. The accuracy is 0.8464285714285713. Determine the test accuracy and plot the confusion matrix.
- Create an SVM object and a GridSearchCV object to find the best parameters, which are {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}. The accuracy is 0.8482142857142856. Determine the test accuracy and plot the confusion matrix.
- Create a decision tree classifier object and a GridSearchCV object to find the best parameters, which are {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 4, 'min_samples_split': 2, 'splitter': 'random'}. The accuracy is 0.9017857142857144. Determine the test accuracy and plot the confusion matrix.
- Create a KNN object and a GridSearchCV object to find the best parameters, which are {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}. The accuracy is 0.8482142857142858. Determine the test accuracy and plot the confusion matrix.
- Compare all the models to determine the best performing one, which is the decision tree classifier with a training accuracy of 0.9017857142857144.
- https://github.com/Judithboluwatito/SPACEX-MACHINE-LEARNING/blob/main/7-SPACEX%20MACHINE%20LEARNNG%20-%20IBM-DS0321EN-SkillsNetwork_labs_module_4_SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Results

Exploratory data analysis results

- Based on the scatter plot of Flight number Vs Payload mass, it appears that different launch sites have varying success rates. For instance, the CCAFS LC-40 site has a success rate of 60%, while KSC LC-39A and VAFB SLC 4E have a success rate of 77%.
- The Payload Vs. Launch Site scatter point chart shows that there are no rockets launched for heavy payload mass (greater than 10000) at the VAFB-SLC launch site.

- The scatter plot between orbit and Flight number suggests that in the LEO orbit, the success rate appears related to the number of flights. However, there seems to be no relationship between flight number when in GTO orbit.
- The scatter plot between Orbit and Payload mass reveals that for heavy payloads, the successful landing or positive landing rate is higher for Polar, LEO, and ISS. However, for GTO, it is difficult to distinguish as both positive landing rate and negative landing (unsuccessful mission) are present.
- **Interactive analytics demo in screenshots**
- **Predictive analysis results**
- Based on the predictive analysis, we used the train_test_split function to split the data into 72 training samples and 18 test samples, with a test size of 0.2 and random state of 2.
- We then used four different classification models - Logistic regression, SVM, Decision tree, and KNN - and obtained their best parameters using GridSearchCV. The best parameters for each model were:
- Logistic regression: {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
- SVM: {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
- Decision tree: {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 4, 'min_samples_split': 2, 'splitter': 'random'}
- KNN: {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
- We then plotted a confusion matrix for each model to visualize its performance.
- Interestingly, all models had the same accuracy on the test data, which was 0.8333333333333334. This suggests that all four models are similarly effective at predicting the outcome variable.

Section 2

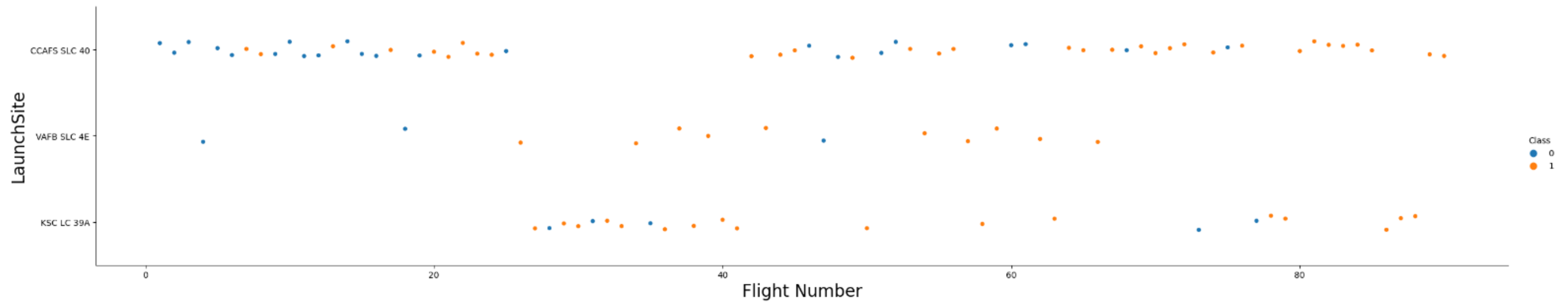
Insights drawn from EDA

Flight Number vs. Launch Site

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'`

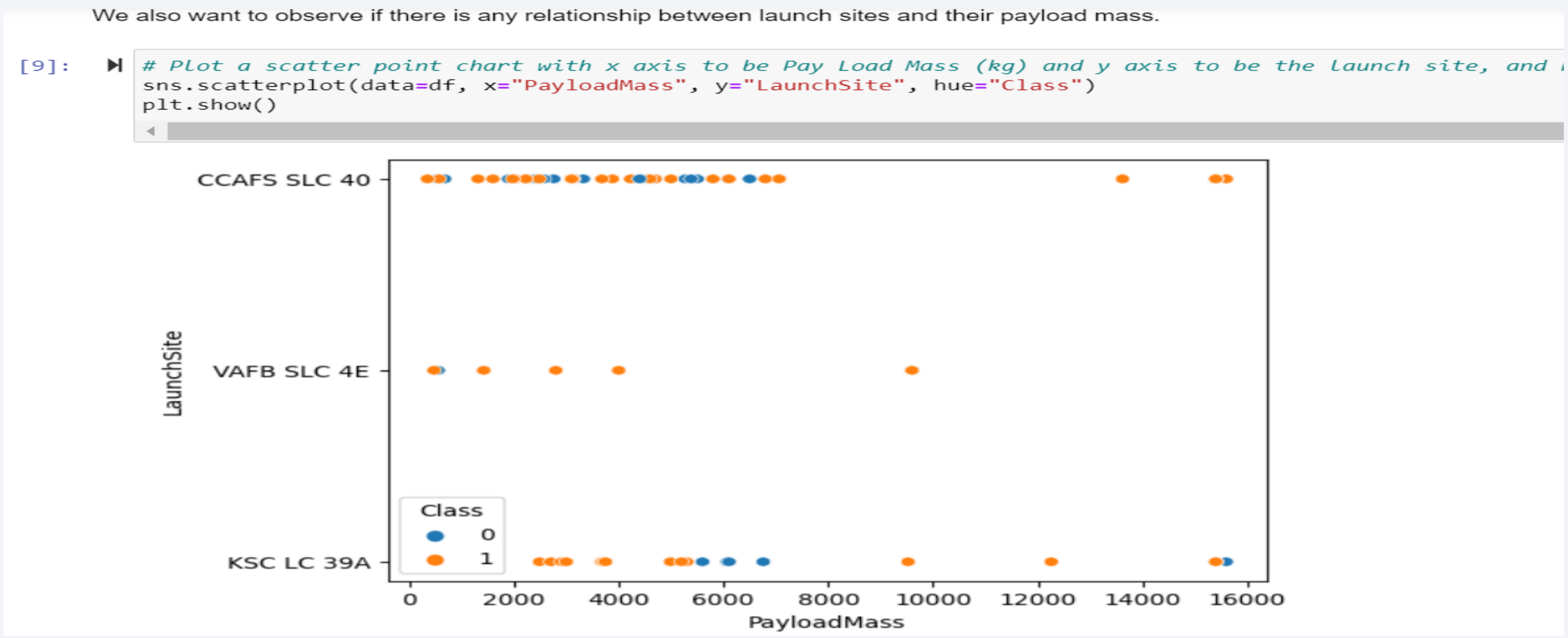
```
In [4]:
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class value
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("LaunchSite",fontsize=20)
plt.show()
```



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

Launches from flight number 18 and onwards had a higher success rate at the VAFB SLC launch site.

Payload vs. Launch Site

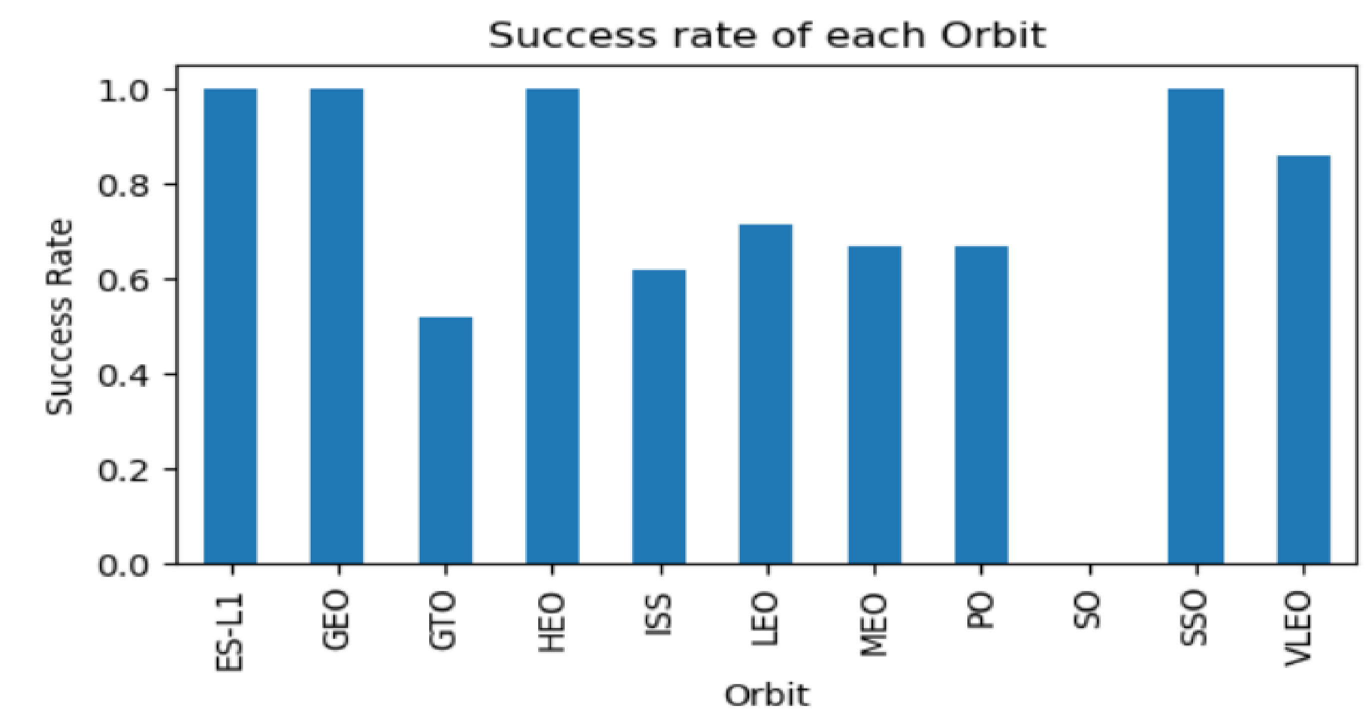


We also want to observe if there is any relationship between launch sites and their payload mass.

```
[9]:  # Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and
      sns.scatterplot(data=df, x="PayloadMass", y="LaunchSite", hue="Class")
      plt.show()
```

VAFB SLC had success with payload mass from 1000, while KSC had no success at +/-6000.

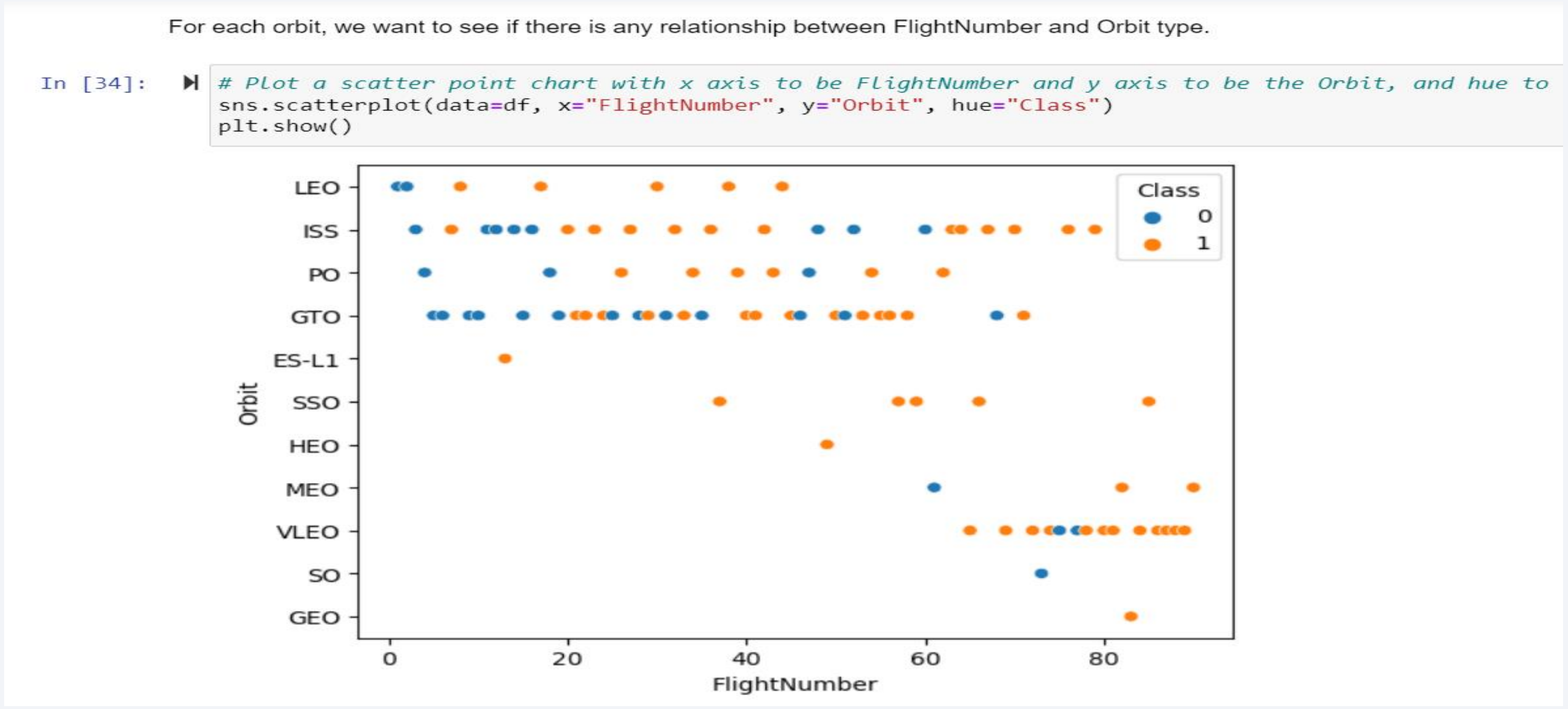
Success Rate vs. Orbit Type



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

High mission success rates were observed for orbits ES-L1, GEO, HEO, SSO, and VLEO

Flight Number vs. Orbit Type

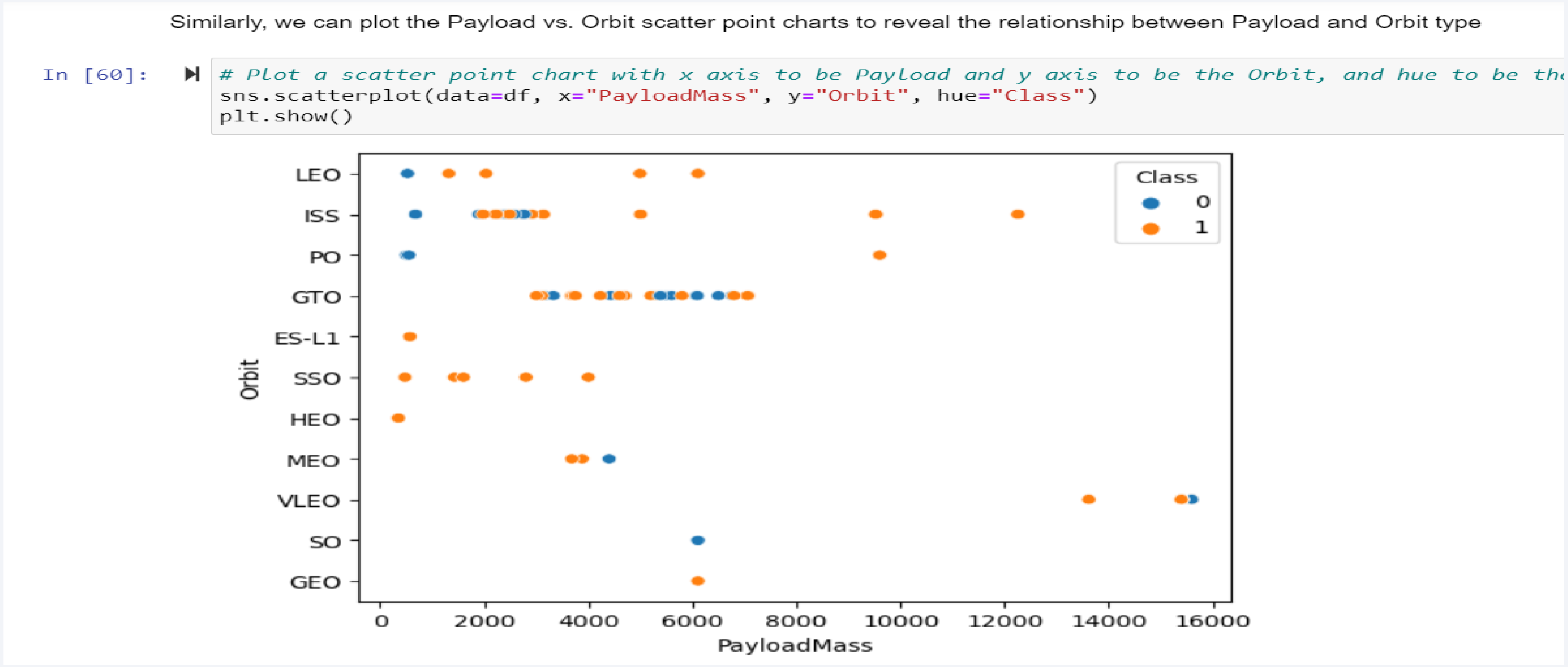


For each orbit, we want to see if there is any relationship between FlightNumber and Orbit type.

```
In [34]:
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to
sns.scatterplot(data=df, x="FlightNumber", y="Orbit", hue="Class")
plt.show()
```

There appears to be a relationship between success and flight number in the LEO orbit, while no such relationship exists in the GTO orbit.

Payload vs. Orbit Type



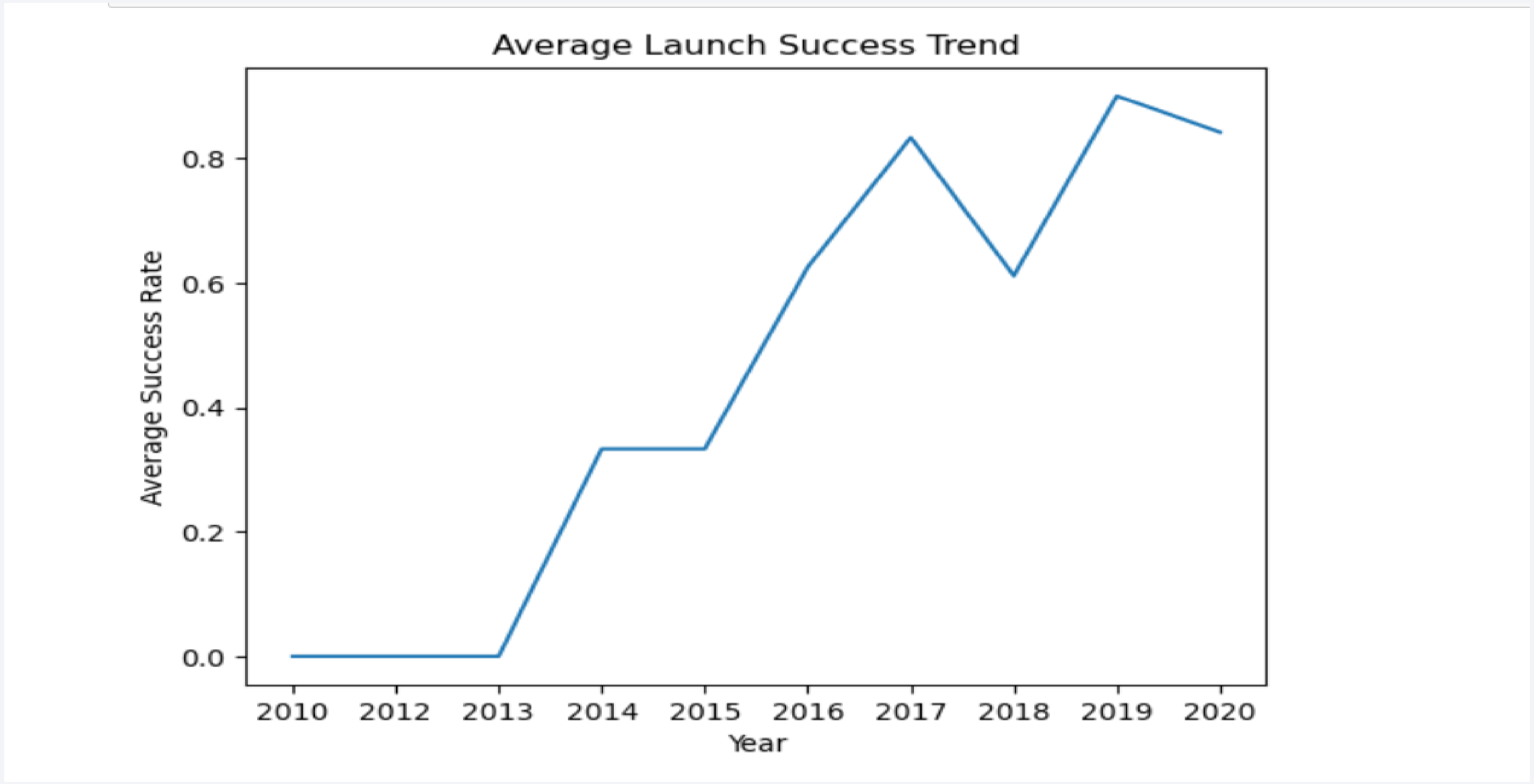
Similarly, we can plot the Payload vs. Orbit scatter point charts to reveal the relationship between Payload and Orbit type

In [60]:

```
# Plot a scatter point chart with x axis to be Payload and y axis to be the Orbit, and hue to be the
sns.scatterplot(data=df, x="PayloadMass", y="Orbit", hue="Class")
plt.show()
```

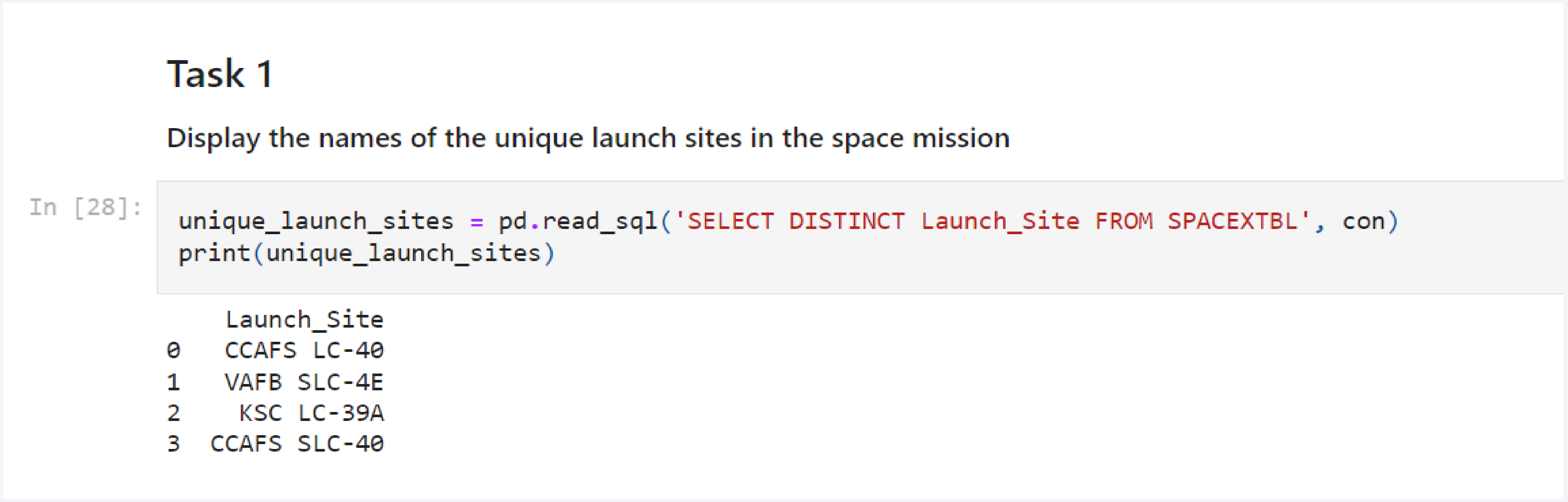
For heavy payloads, successful landings are more likely for Polar, LEO, and ISS orbits.
However, for GTO orbit, it is difficult to distinguish between successful and unsuccessful missions.

Launch Success Yearly Trend



Success rates have been consistently increasing since 2013 until 2020.

All Launch Site Names



Task 1

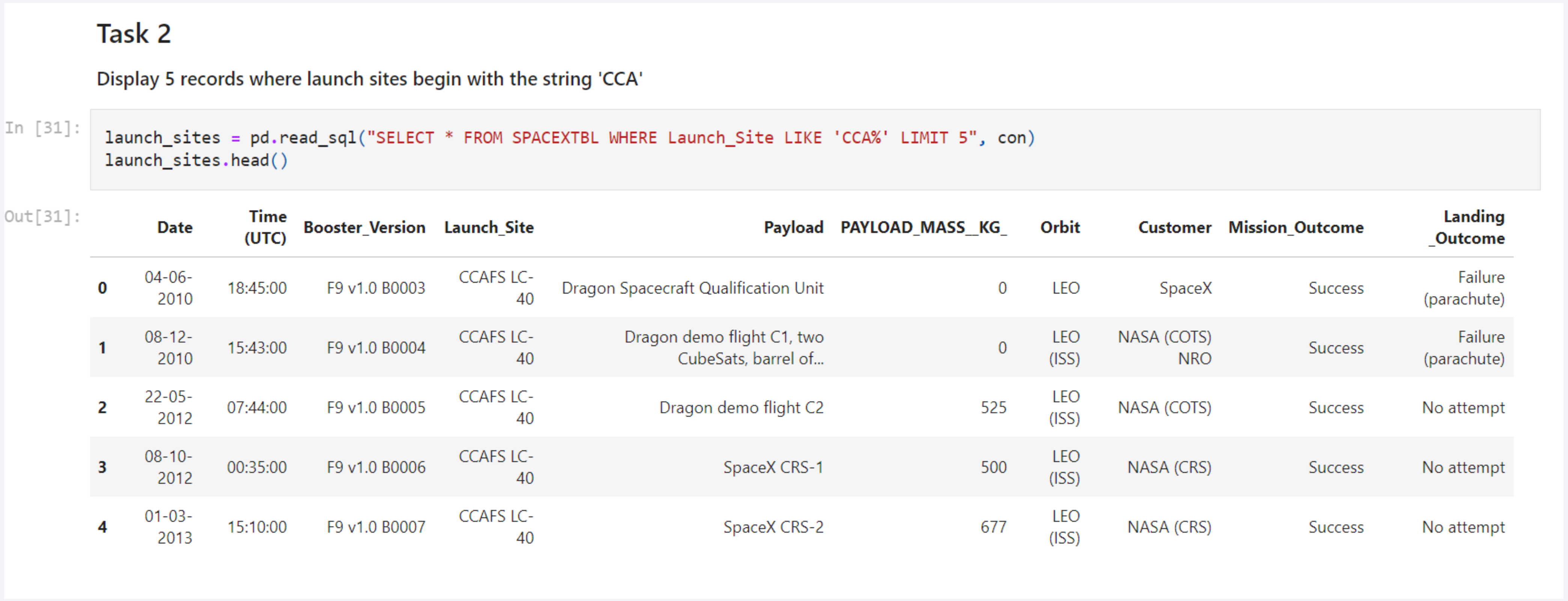
Display the names of the unique launch sites in the space mission

```
In [28]:
unique_launch_sites = pd.read_sql('SELECT DISTINCT Launch_Site FROM SPACEXTBL', con)
print(unique_launch_sites)
```

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

There are four distinct launch sites in the space mission: CCAFS LC, VAFB SLC, KSC LC, CCAFS SLC.

Launch Site Names Begin with 'CCA'



Task 2

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

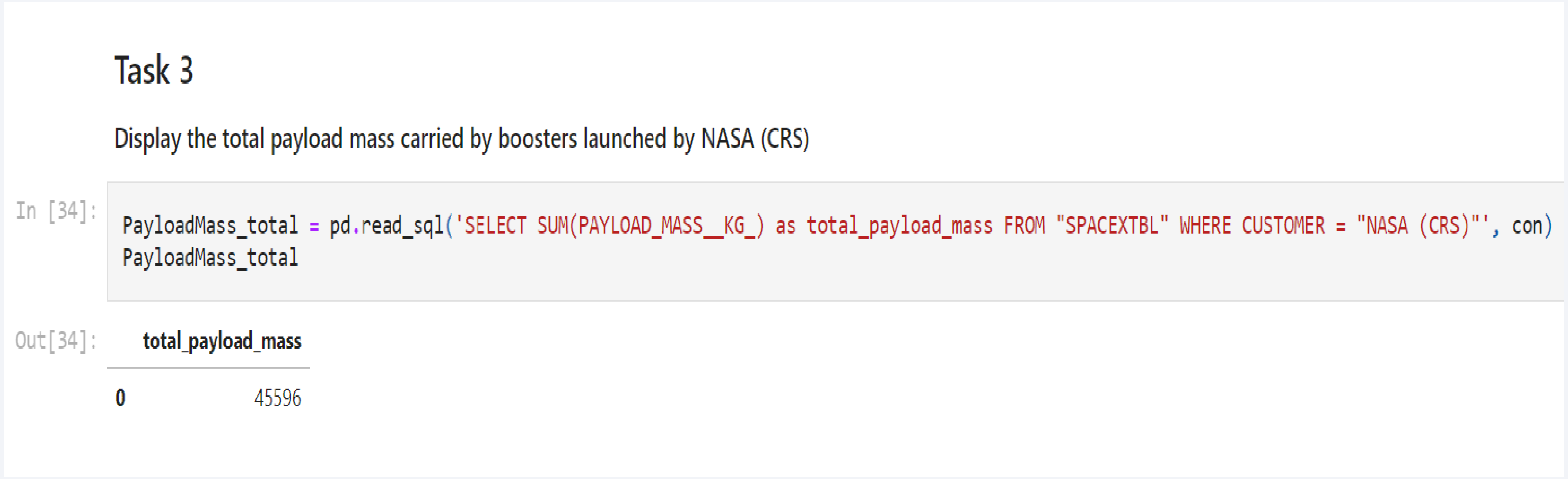
```
In [31]: launch_sites = pd.read_sql("SELECT * FROM SPACEXTBL WHERE Launch_Site LIKE 'CCA%' LIMIT 5", con)
         launch_sites.head()
```

Out[31]:

	Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
0	04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	08-12-2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2	22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
3	08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

NASA and SpaceX launched 5 successful missions to LEO orbit from CCA.

Total Payload Mass



Task 3

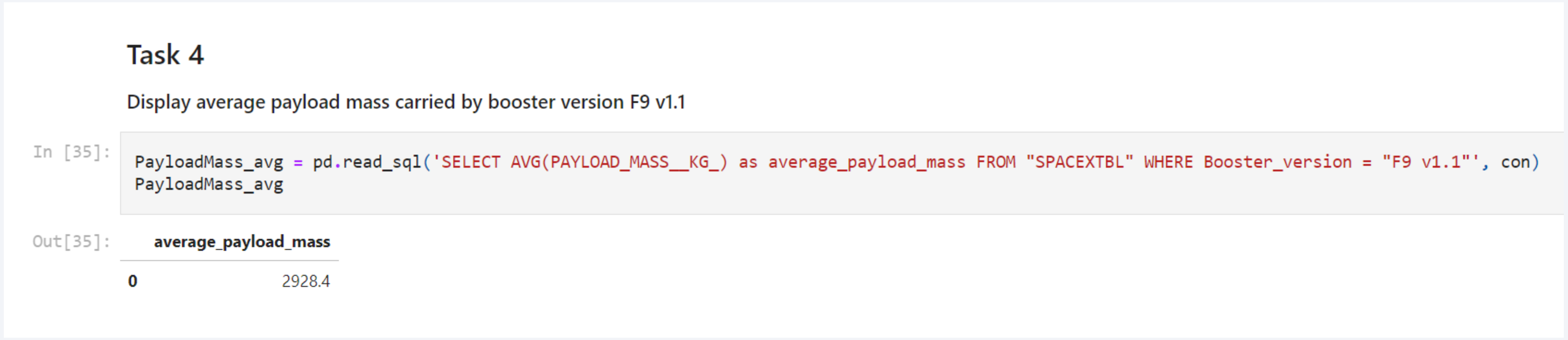
Display the total payload mass carried by boosters launched by NASA (CRS)

```
In [34]: PayloadMass_total = pd.read_sql('SELECT SUM(PAYLOAD_MASS__KG_) as total_payload_mass FROM "SPACEXTBL" WHERE CUSTOMER = "NASA (CRS)"', con)
         PayloadMass_total
```

```
Out[34]:    total_payload_mass
         0               45596
```

NASA (CRS) launched boosters with a total payload mass of 45596.

Average Payload Mass by F9 v1.1



Task 4

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

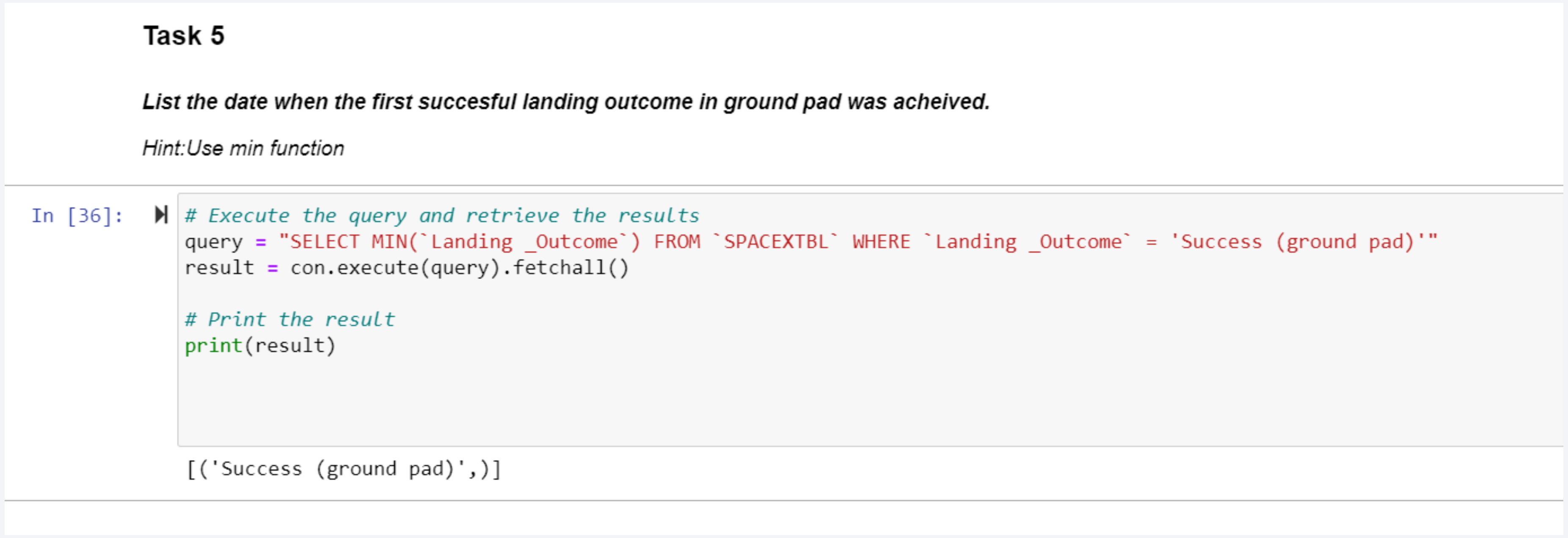
```
In [35]: PayloadMass_avg = pd.read_sql('SELECT AVG(PAYLOAD_MASS__KG_) as average_payload_mass FROM "SPACEXTBL" WHERE Booster_version = "F9 v1.1"', con)
         PayloadMass_avg
```

Out[35]:

	average_payload_mass
0	2928.4

Avg. payload mass for booster F9 v1.1 is 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

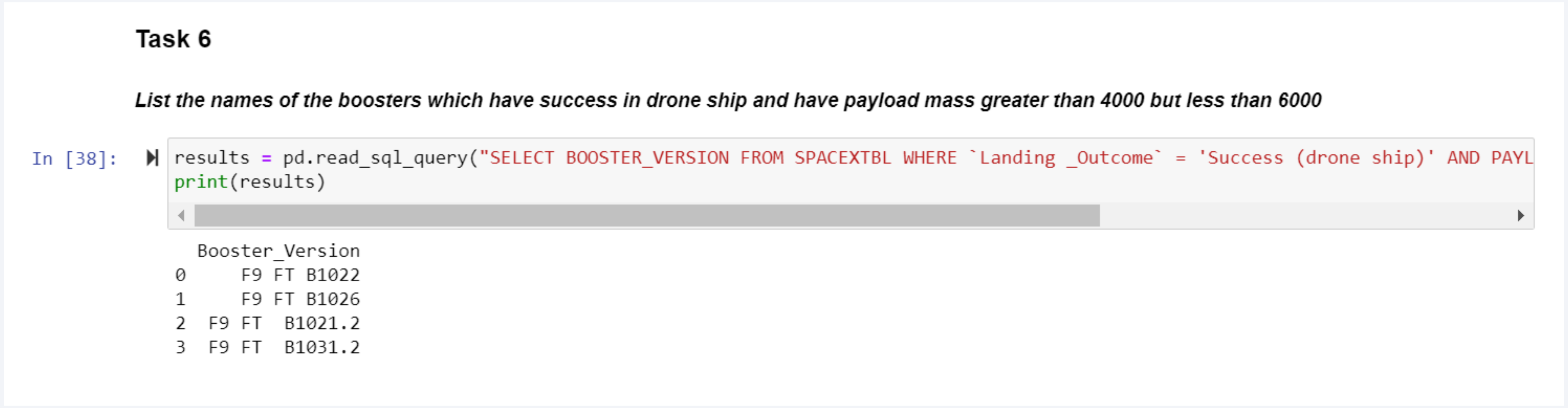
```
In [36]:  # Execute the query and retrieve the results
          query = "SELECT MIN(`Landing _Outcome`) FROM `SPACEXTBL` WHERE `Landing _Outcome` = 'Success (ground pad)'"
          result = con.execute(query).fetchall()

          # Print the result
          print(result)
```

```
[('Success (ground pad)',)]
```

No record was found for successful ground landing date.

Successful Drone Ship Landing with Payload between 4000 and 6000


Task 6

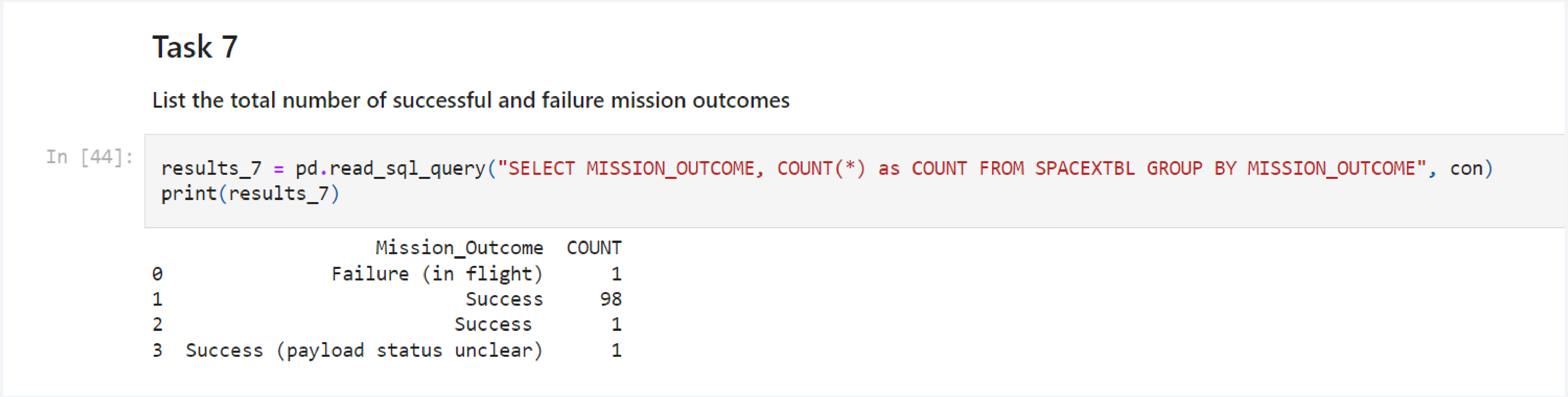
List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

In [38]:
```
results = pd.read_sql_query("SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE `Landing _Outcome` = 'Success (drone ship)' AND PAYL
print(results)
```

```
  Booster_Version
0     F9 FT B1022
1     F9 FT B1026
2   F9 FT  B1021.2
3   F9 FT  B1031.2
```

4 successful drone ship landings were recorded for boosters carrying a payload mass between 4000 and 6000.

Total Number of Successful and Failure Mission Outcomes



Task 7

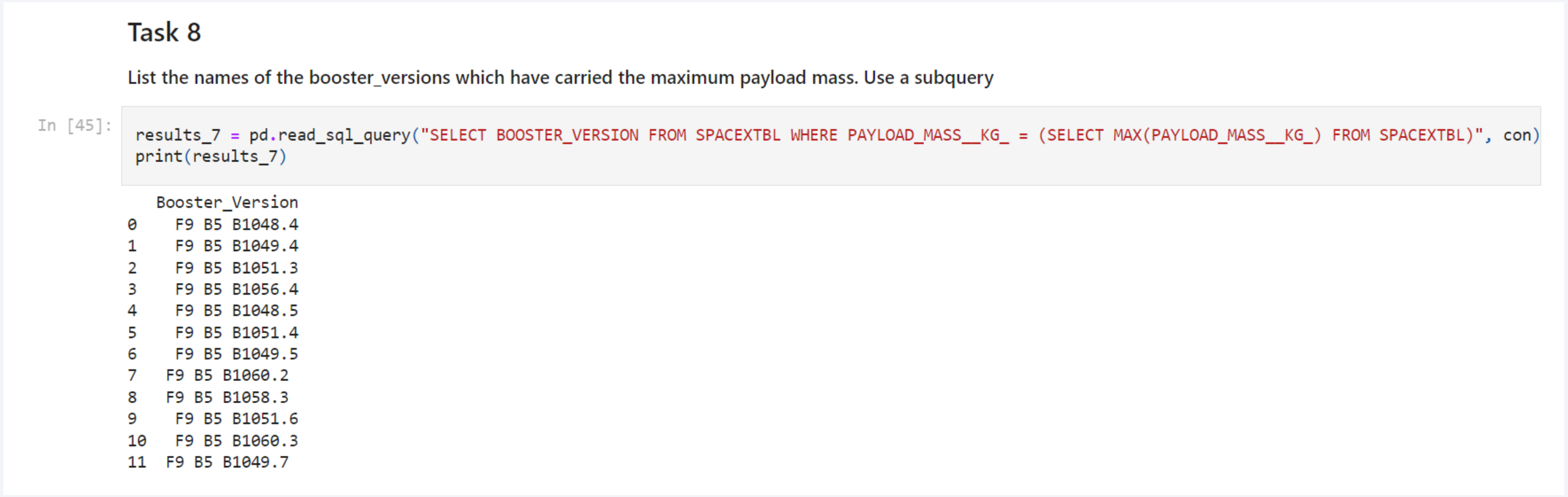
List the total number of successful and failure mission outcomes

```
In [44]:  results_7 = pd.read_sql_query("SELECT MISSION_OUTCOME, COUNT(*) as COUNT FROM SPACEXTBL GROUP BY MISSION_OUTCOME", con)
          print(results_7)
```

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

There were 100 successful missions and 1 failure.

Boosters Carried Maximum Payload



Task 8

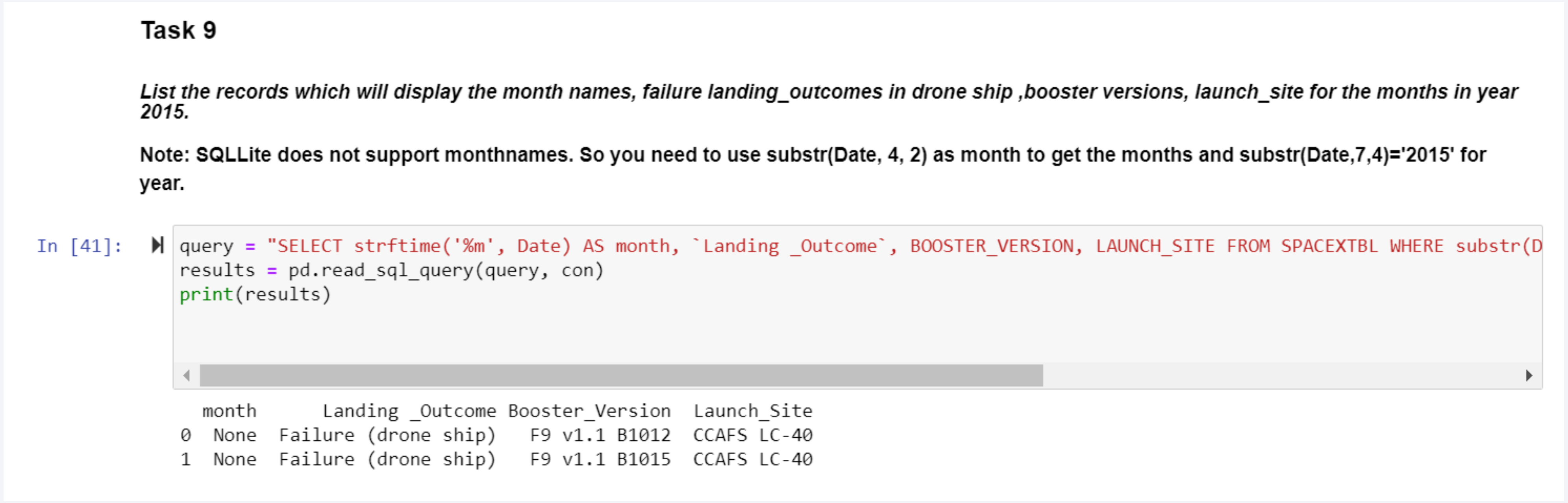
List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

```
In [45]: results_7 = pd.read_sql_query("SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL)", con)
         print(results_7)
```

```
   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
```

12 F9 B5 Boosters carried max payload mass.

2015 Launch Records



Task 9

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, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.

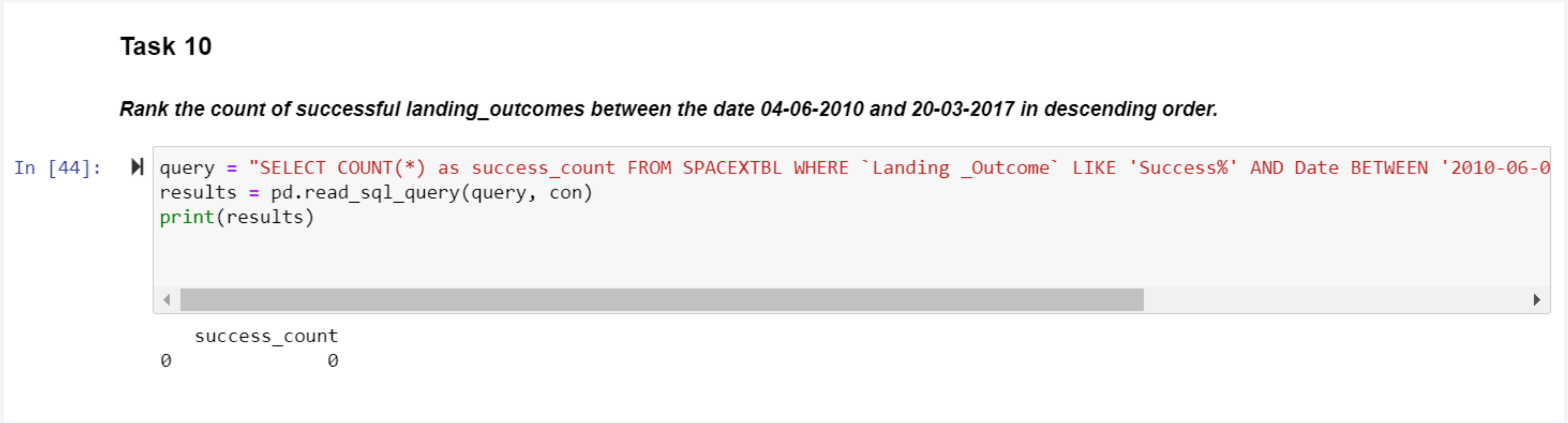
In [41]:

```
query = "SELECT strftime('%m', Date) AS month, `Landing _Outcome`, BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WHERE substr(D
results = pd.read_sql_query(query, con)
print(results)
```

```
  month       Landing _Outcome Booster_Version  Launch_Site
0  None  Failure (drone ship)    F9 v1.1 B1012  CCAFS LC-40
1  None  Failure (drone ship)    F9 v1.1 B1015  CCAFS LC-40
```

Two records were found for drone ship landing outcomes, booster versions, and launch sites for the months in 2015.

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



Task 10

Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

In [44]:
```
query = "SELECT COUNT(*) as success_count FROM SPACEXTBL WHERE `Landing _Outcome` LIKE 'Success%' AND Date BETWEEN '2010-06-0
results = pd.read_sql_query(query, con)
print(results)
```

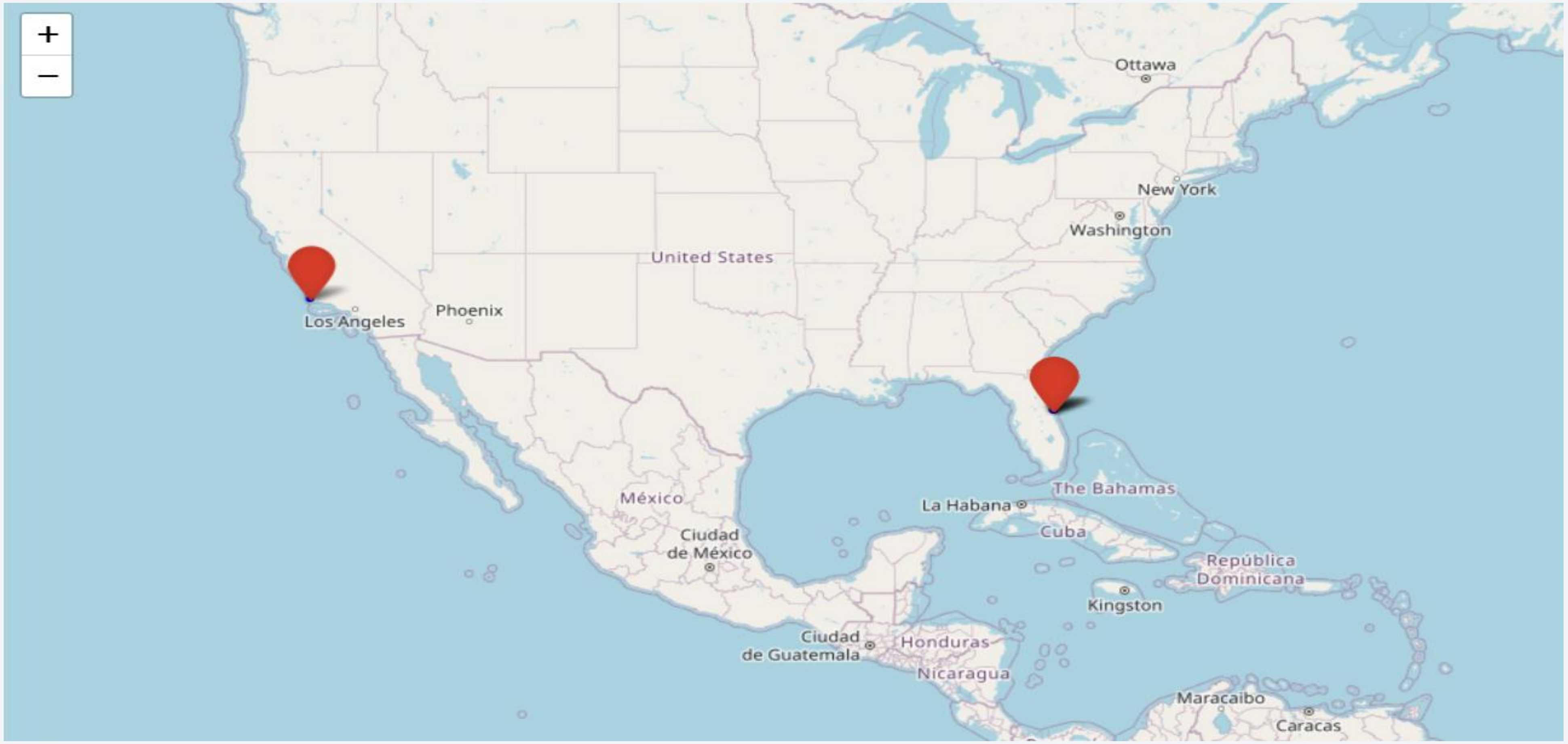
```
   success_count
0              0
```

No landing outcome records were found from June 2010 to March 2017.

Section 3

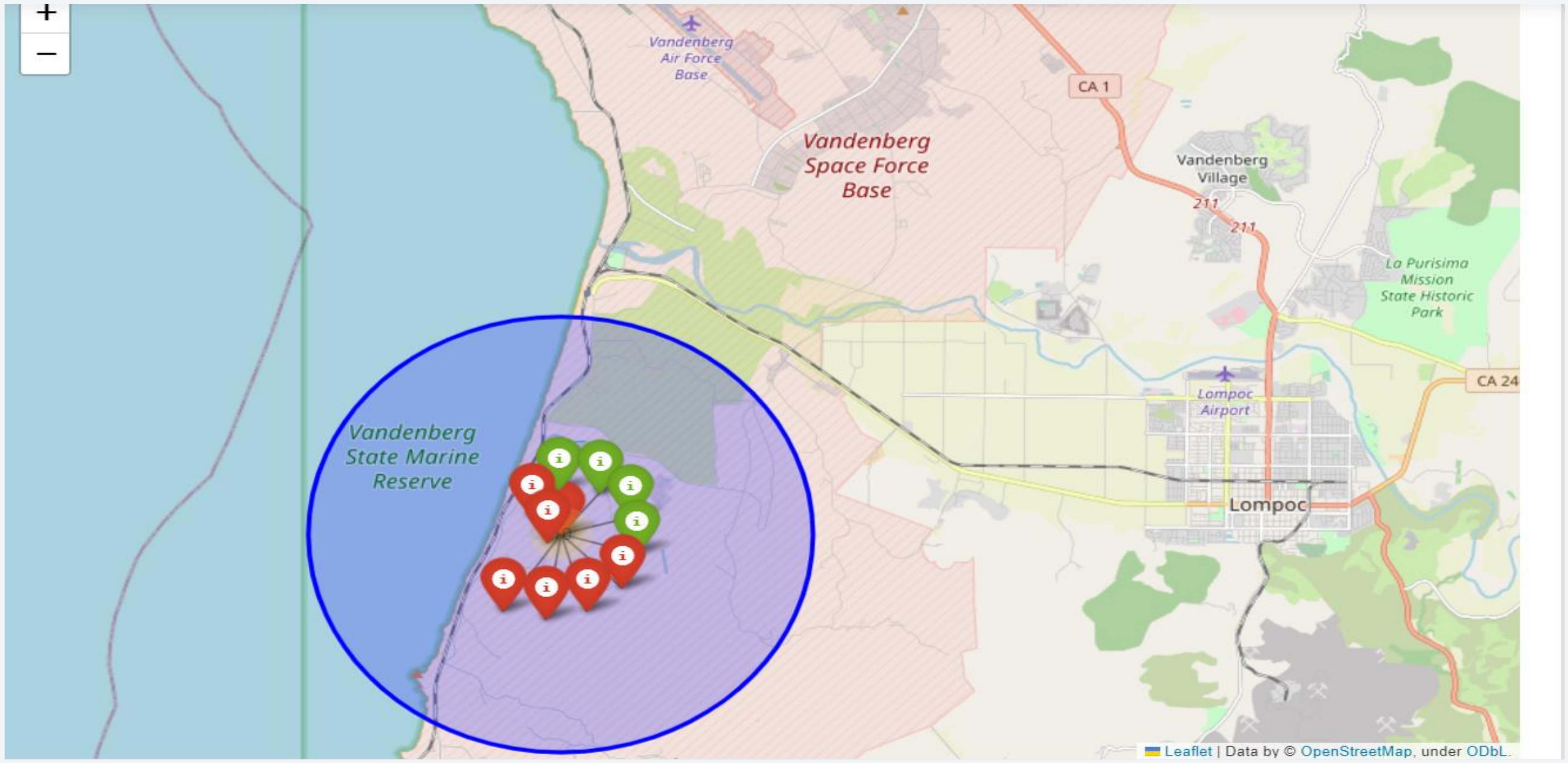
Launch Sites Proximities Analysis

<Folium Map Screenshot 1>



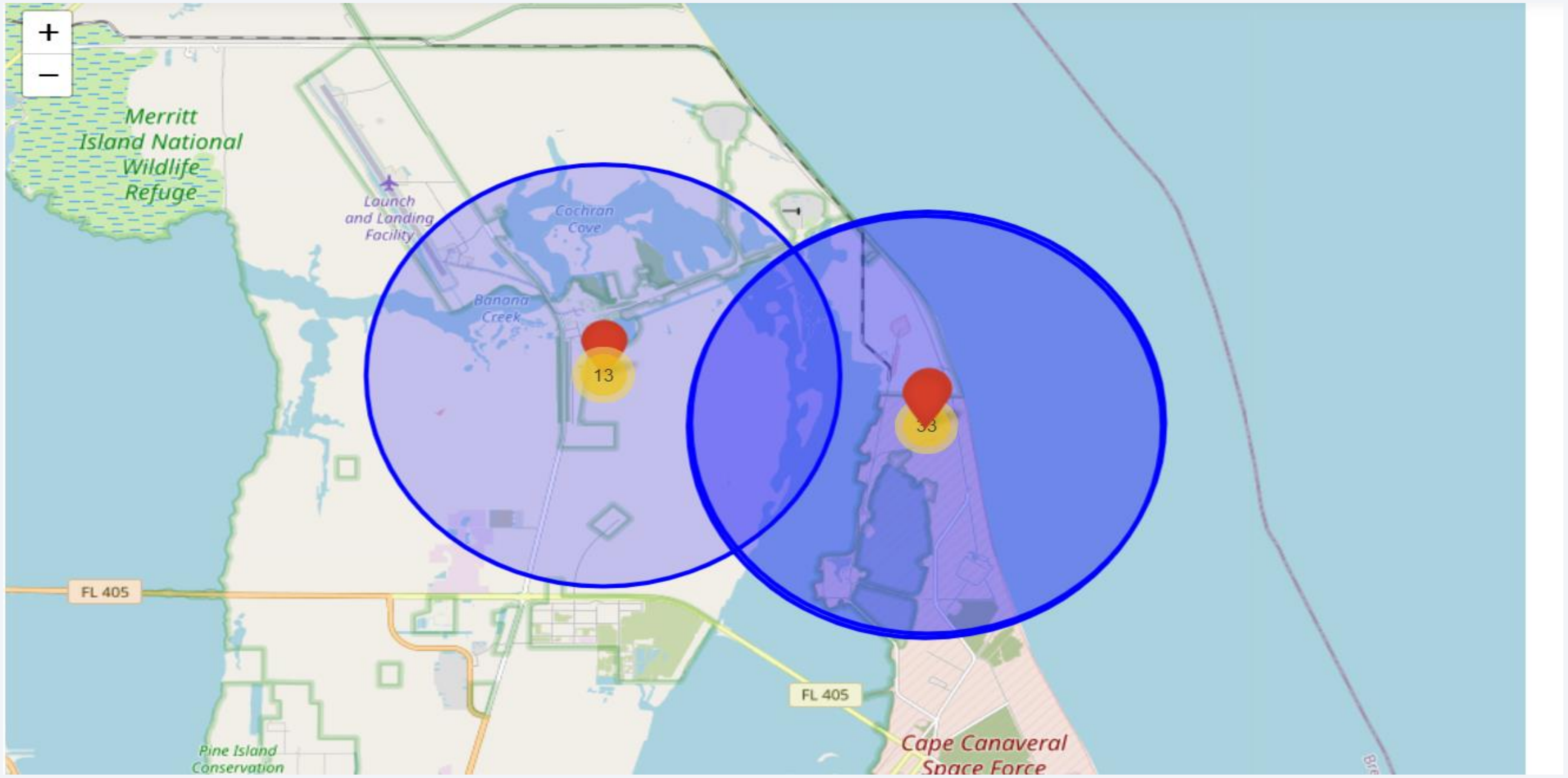
The map displays the locations of the launch sites used by SpaceX Falcon 9 missions.

<Folium Map Screenshot 2>

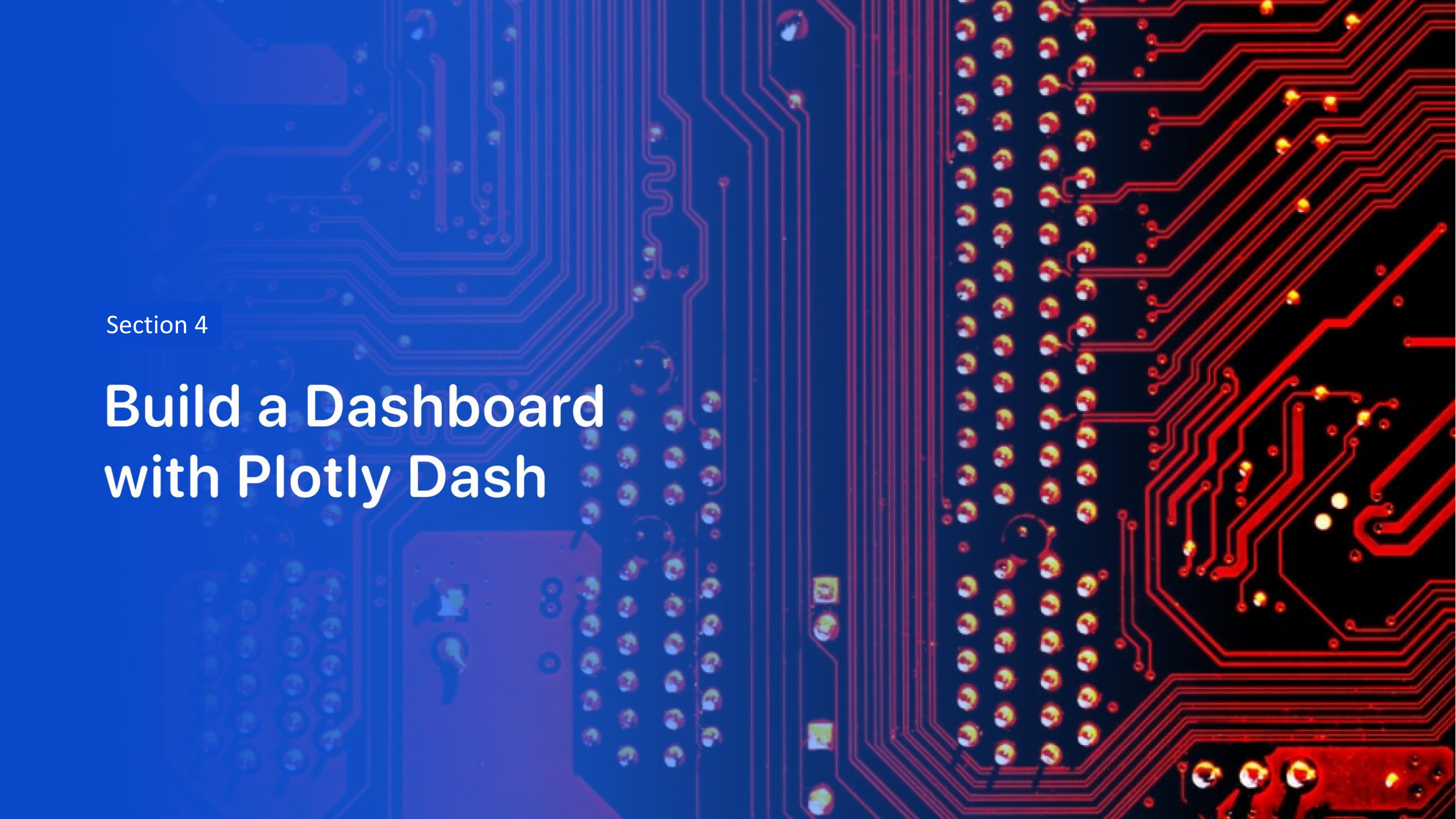

+
−
Vandenberg Air Force Base
Vandenberg Space Force Base
CA 1
Vandenberg Village
211
La Purisima Mission State Historic Park
Lompoc Airport
CA 24
Lompoc
Vandenberg State Marine Reserve
Leaflet | Data by © OpenStreetMap, under ODbL.

Folium map with clusters and markers indicating successful and failed mission launch sites

<Folium Map Screenshot 3>


+
−
Merritt Island National Wildlife Refuge
Launch and Landing Facility
Cochran Cove
Banana Creek
13
33
FL 405
FL 405
Pine Island Conservation
Cape Canaveral Space Force

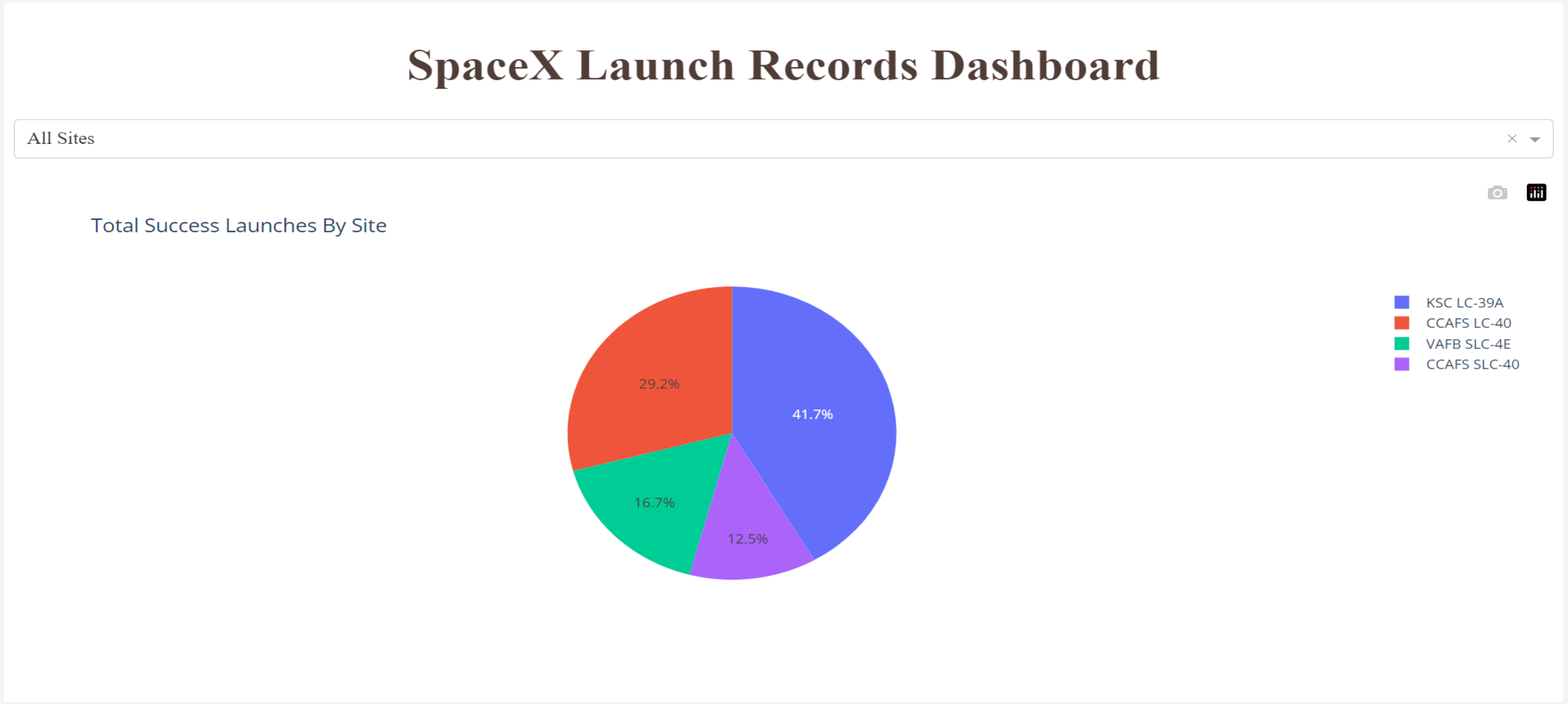
Folium map showing launch site proximities to highways, railroads, coastlines and cities



Section 4

Build a Dashboard with Plotly Dash

<Dashboard Screenshot 1>

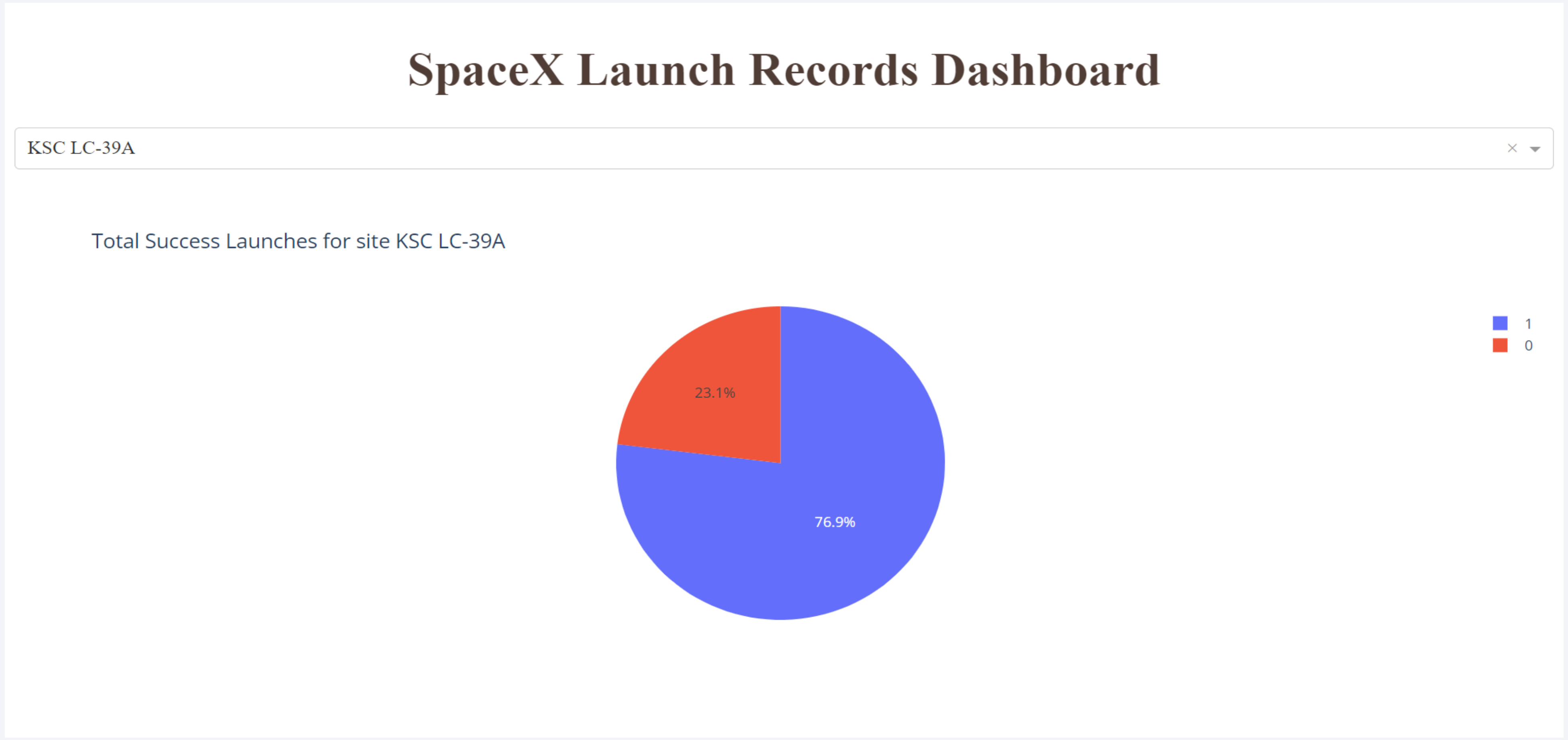

SpaceX Launch Records Dashboard

All Sites

Total Success Launches By Site

Pie chart displays successful launches by launch sites, with KSC LC 39A and CCAFS LC 40 having the highest success rates.

<Dashboard Screenshot 2>

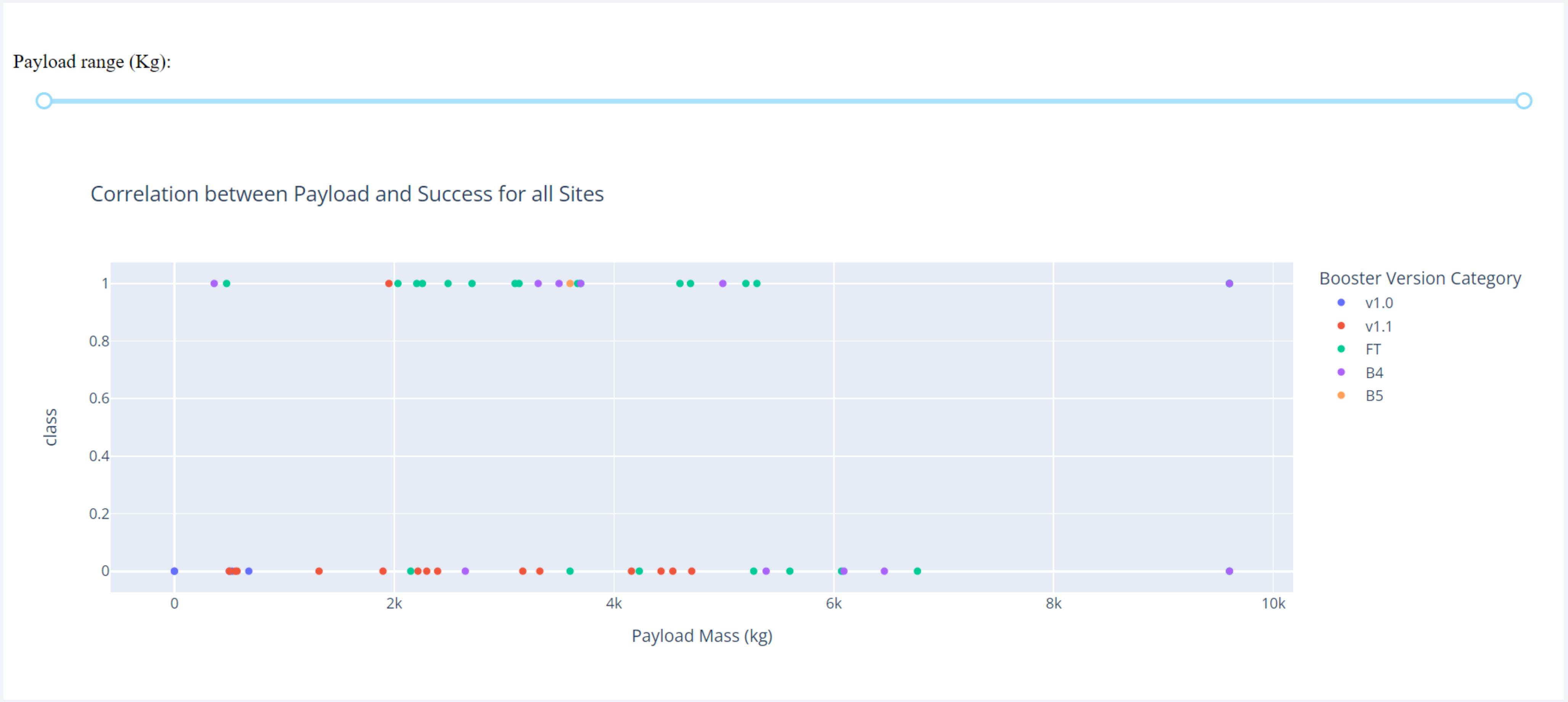

SpaceX Launch Records Dashboard

KSC LC-39A

Total Success Launches for site KSC LC-39A

Pie chart: KSC LC-39A - 76.9% success, 23.1% failed missions.

<Dashboard Screenshot 3>

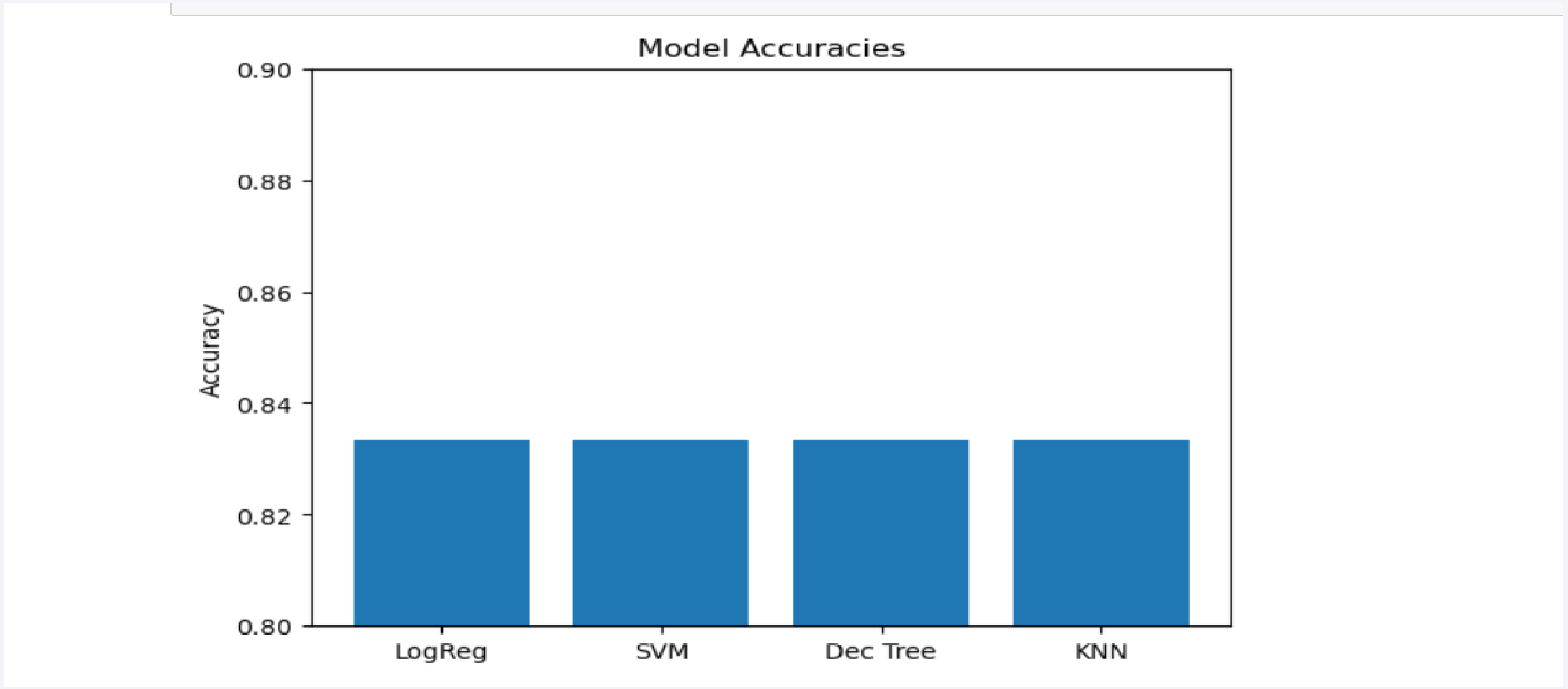

Payload range (Kg):

The scatter plot indicates that booster v1.0 and FT had the highest success rate for payloads up to 10k and 6k maximum, respectively, while others had success rates below 5k.

Section 5

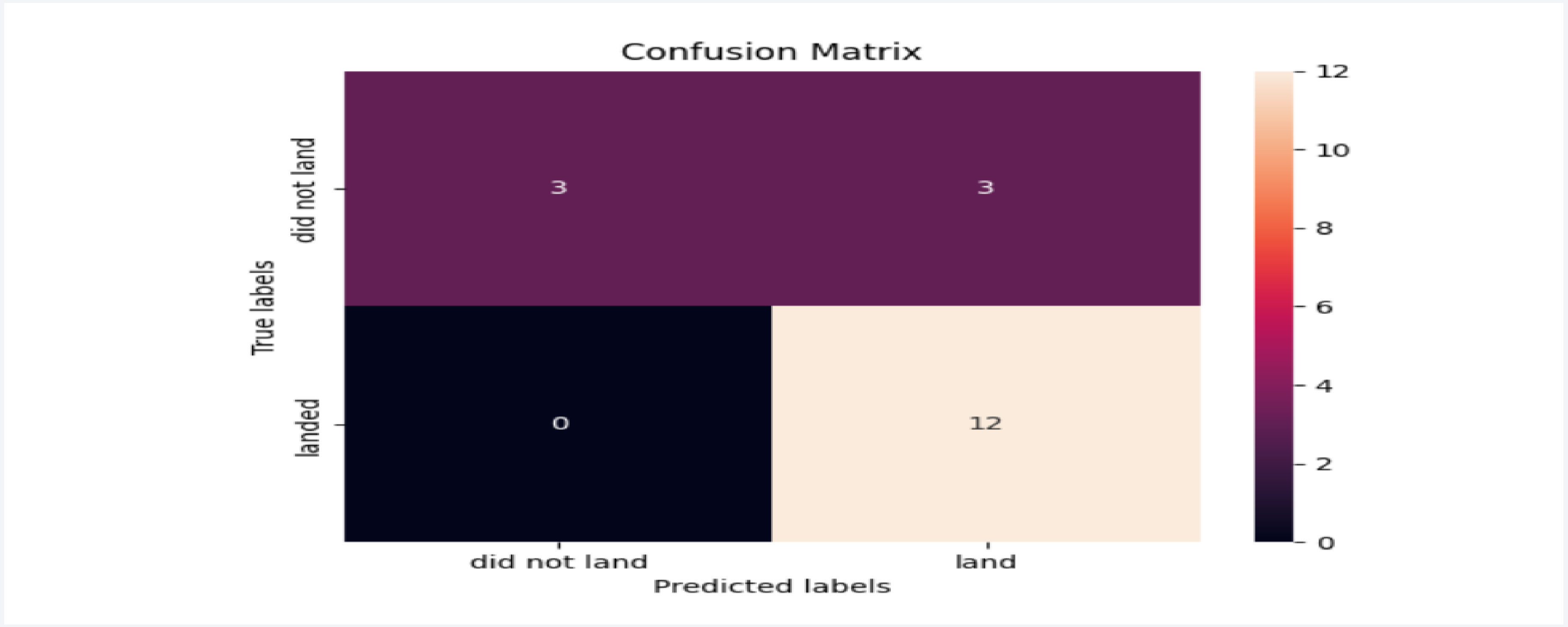
Predictive Analysis (Classification)

Classification Accuracy



Test set accuracy for all models was 0.8333333333333334, but the Decision tree classifier had a higher accuracy of 0.9017857142857144 in the training set.

Confusion Matrix



The confusion matrix reveals that the Decision tree classifier effectively identifies the different classes, but there are several false positives.

Conclusions

- Based on the bar chart visualization, we found that the orbits with the highest success rates for missions are ES LI, GEO, HEO, and SSO.
- The scatter plot we generated from the Plotly interactive dashboard revealed that most booster versions were successful in launching payloads with masses between 2,000 to 6,000, with FT being the most successful booster version followed by B4, which has a payload capacity of 10,000.
- We analyzed the data using a Plotly pie chart and found that KSC LC-39A was the launch site with the highest success rate of 41.7%, followed by CCAFS LC-40 with 29.2%, VAFB SLC-4E with 16.7%, and CCAFS SLC-40 with the lowest success rate of 12.5%.
- Using the Folium map, we discovered that successful launches were mostly located near highways, railroads, and coastlines, but not within the proximity of any city.
- Our classification model showed that the Decision tree classifier was the best prediction model, with a training accuracy of 0.9017857142857144 and a test accuracy of 0.8333333333333334, which was the same as other models tested.

Appendix

- During the SQL lab, an extra code was required to force install pandas, which was: "pip install pandas --force-reinstall".
- In the Folium map lab, the "geopy" package library was unable to be installed despite several attempts, which resulted in the geodesic distance not being calculated. The error message displayed was: "NameError: name 'geodesic' is not defined".
- All the notebooks, codes, and assets related to this project are available in the GitHub URL link provided.
- Please note that the GitHub URL link has not been provided in this conversation, so you will need to insert the appropriate link if you are sharing this information with someone.

Thank you!