IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Massimiliano Marchesiello
14th July 2024

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection through API
  - Data Collection with Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis with Data Visualization
  - Interactive Visual Analytics with Folium
  - Machine Learning Prediction
- Summary of all results
  - Exploratory Data Analysis result
  - Interactive analytics in screenshots
  - Predictive Analytics result

Introduction

- Project background and context

  SpaceX advertises Falcon 9 rocket launches on its website at a cost of $62 million, significantly lower than the $165 million charged by other providers. This cost reduction is largely due to SpaceX's ability to reuse the first stage of the rocket. By predicting the likelihood of the first stage landing successfully, we can better estimate the cost of a launch. This information is valuable for alternate companies looking to compete with SpaceX in the rocket launch market. The goal of this project is to develop a machine learning pipeline that predicts the likelihood of the first stage landing successfully.

- Problems to Address
  - Factors Influencing Successful Landings: Identify the key factors that determine whether the rocket will land successfully.
  - Feature Interactions: Analyze how various features interact and contribute to the success rate of a landing.
  - Operating Conditions: Determine the necessary operating conditions to ensure a successful landing program.

Section 1

Methodology

Methodology

Executive Summary

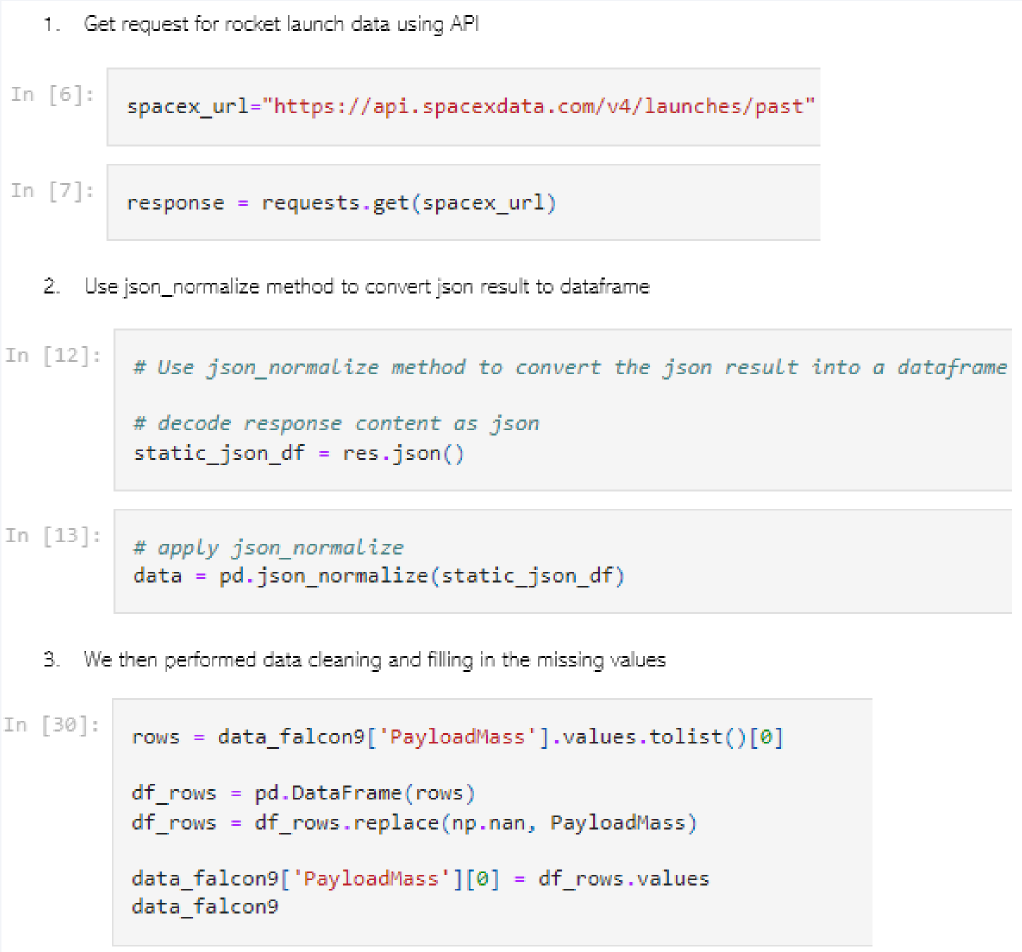
- Data collection methodology
  - Data was collected using SpaceX API and web scraping from Wikipedia.
- Perform data wrangling
  - One-hot encoding was applied to categorical features
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

- The data was collected using various methods
  - Data collection was conducted using GET requests to the SpaceX API.
  - The response content was decoded as JSON using the ‘.json()’ function and converted into a pandas DataFrame using ‘.json_normalize()’.
  - The data was then cleaned, checked for missing values, and any missing values were filled as necessary.
  - Additionally, web scraping was performed using BeautifulSoup to obtain Falcon 9 launch records from Wikipedia
  - The objective was to extract the launch records as an HTML table, parse the table, and convert it into a pandas DataFrame for further analysis.

Data Collection – SpaceX API

- We utilized GET requests to the SpaceX API to collect data, followed by data cleaning, basic wrangling, and formatting.
    - You can access the notebook via the following link: https://github.com/Marchesiello/IBM-Data-Science-Capstone-SpaceX/blob/main/Plots/Data%20Collection%20API.ipynb



1. Get request for rocket launch data using API

```
In [6]: spacex_url="https://api.spacexdata.com/v4/launches/past"
```

```
In [7]: response = requests.get(spacex_url)
```

2. Use json_normalize method to convert json result to dataframe

```
In [12]: # Use json_normalize method to convert the json result into a dataframe

         # decode response content as json
         static_json_df = res.json()
```

```
In [13]: # apply json_normalize
         data = pd.json_normalize(static_json_df)
```

3. We then performed data cleaning and filling in the missing values

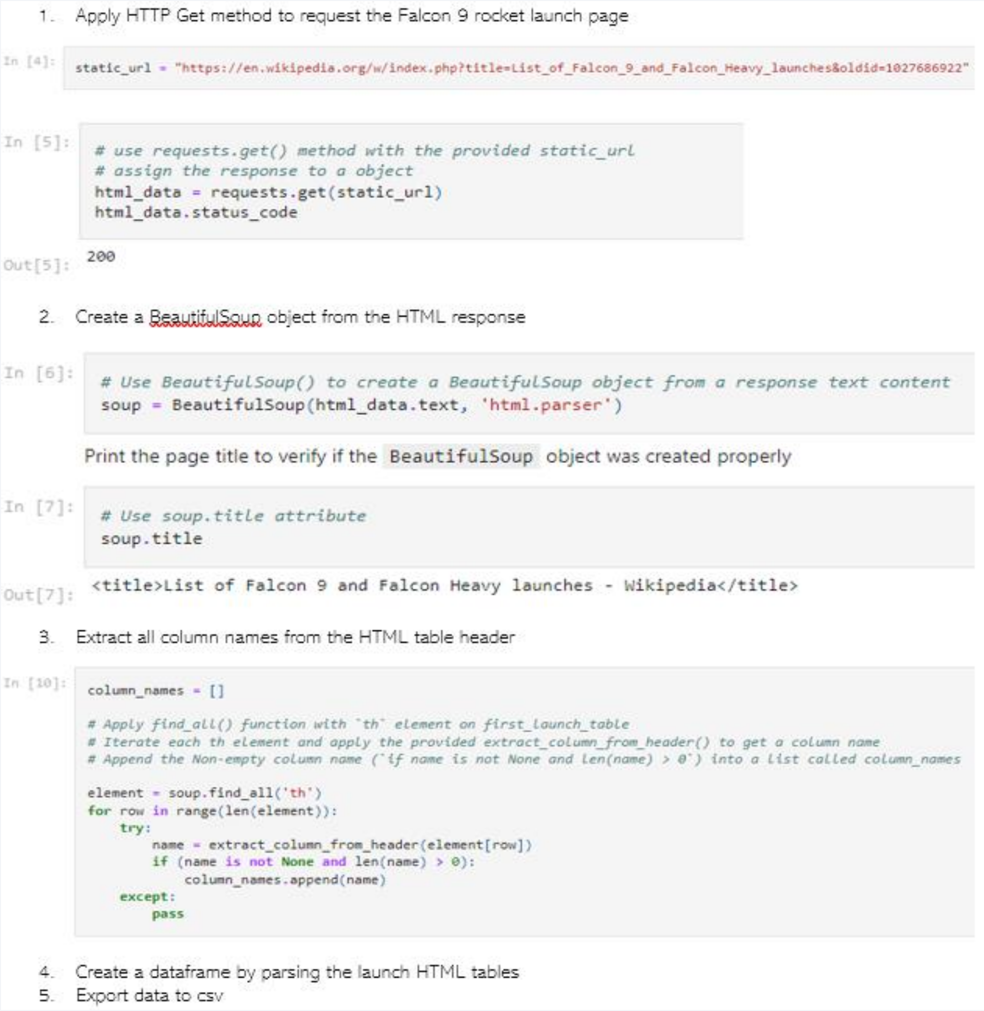
```
In [30]: rows = data_falcon9['PayloadMass'].values.tolist()[0]

         df_rows = pd.DataFrame(rows)
         df_rows = df_rows.replace(np.nan, PayloadMass)

         data_falcon9['PayloadMass'][0] = df_rows.values
         data_falcon9
```

Data Collection - Scraping

- We used web scraping with BeautifulSoup to gather Falcon 9 launch records.
- The extracted data was then parsed and converted into a pandas DataFrame.
    - You can access the notebook via the following link: https://github.com/Marchesiello/IBM-Data-Science-Capstone-SpaceX/blob/main/Plots/Data%20Collection%20with%20Web%20Scraping.ipynb



1. Apply HTTP Get method to request the Falcon 9 rocket launch page

```
In [4]: static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

```
In [5]: # use requests.get() method with the provided static_url
        # assign the response to a object
        html_data = requests.get(static_url)
        html_data.status_code

Out[5]: 200
```

2. Create a BeautifulSoup object from the HTML response

```
In [6]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
        soup = BeautifulSoup(html_data.text, 'html.parser')
```

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

```
In [7]: # Use soup.title attribute
        soup.title

Out[7]: <title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

3. Extract all column names from the HTML table header

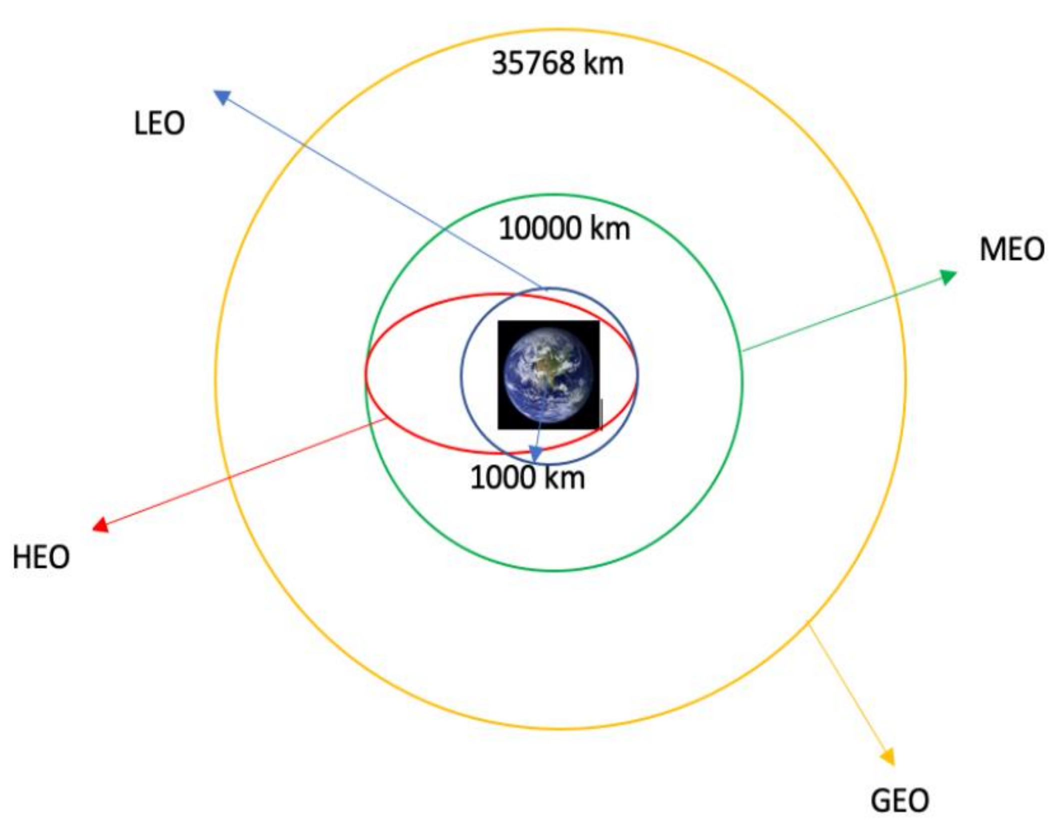
```
In [10]: column_names = []

         # Apply find_all() function with `th` element on first_launch_table
         # Iterate each th element and apply the provided extract_column_from_header() to get a column name
         # Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names

         element = soup.find_all('th')
         for row in range(len(element)):
             try:
                 name = extract_column_from_header(element[row])
                 if (name is not None and len(name) > 0):
                     column_names.append(name)
             except:
                 pass
```

4. Create a dataframe by parsing the launch HTML tables
5. Export data to csv

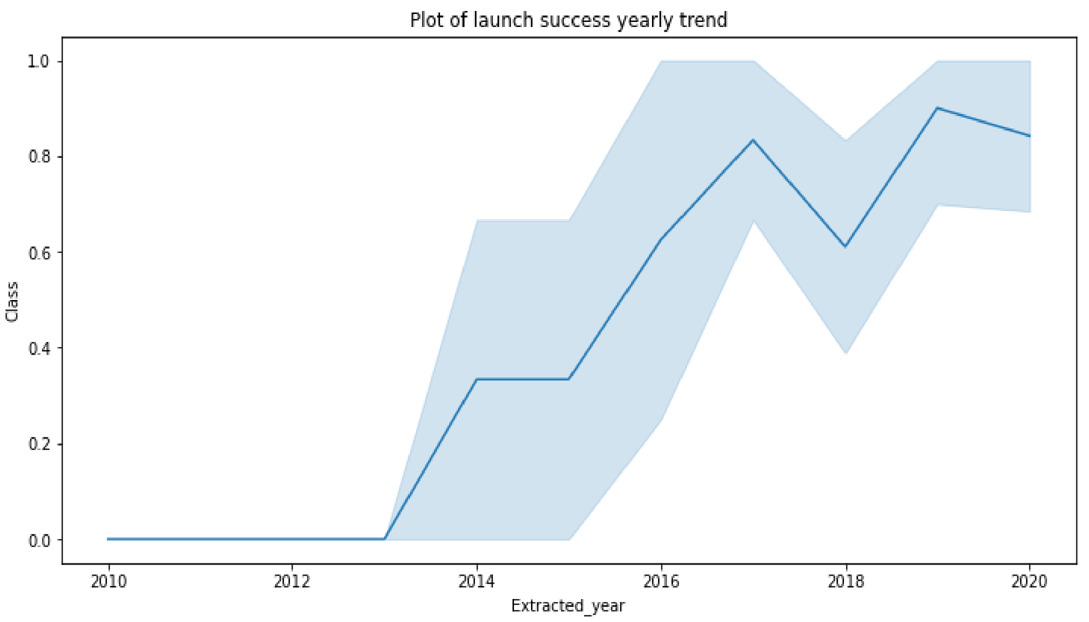
Data Wrangling

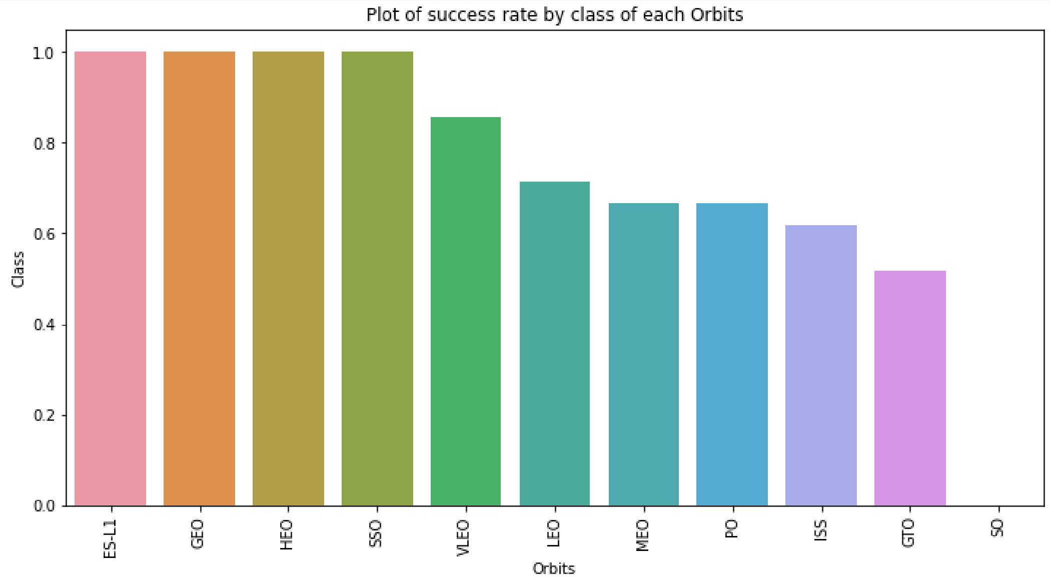


- We performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits
- We created landing outcome label from outcome column and exported the results to csv.
  - You can access the notebook via the following link: https://github.com/Marchesiello/IBM-Data-Science-Capstone-SpaceX/blob/main/Plots/Data%20Wrangling.ipynb

EDA with Data Visualization

- We explored the data by visualizing various relationships, including flight number and launch site, payload and launch site, success rate of each orbit type, flight number and orbit type, and the yearly trend of launch success





- You can access the notebook via the following link: https://github.com/Marchesiello/IBM-Data-Science-Capstone-SpaceX/blob/main/Plots/EDA%20with%20Data%20Visualization.ipynb

EDA with SQL

- We loaded the SpaceX dataset into a PostgreSQL database directly within the Jupyter notebook.
- Using SQL for exploratory data analysis (EDA), we gained insights from the data. For example, we wrote queries to discover:
    - The names of unique launch sites involved in the space missions.
    - The total payload mass carried by boosters launched by NASA (CRS).
    - The average payload mass carried by the booster version F9 v1.1.
    - The total number of successful and failed mission outcomes.
    - The failed landing outcomes on drone ships, along with their booster versions and launch site names.
    - You can access the notebook via the following link: https://github.com/Marchesiello/IBM-Data-Science-Capstone-SpaceX/blob/main/Plots/EDA%20with%20SQL.ipynb

Build an Interactive Map with Folium

- We marked all launch sites and added map elements such as markers, circles, and lines to indicate the success or failure of launches at each site on a Folium map.
- We assigned launch outcomes to classes, with 0 representing failure and 1 representing success.
- Using color-labeled marker clusters, we identified which launch sites have relatively high success rates.
- Additionally, we calculated the distances between each launch site and its nearby landmarks. We addressed questions such as:
    - Whether launch sites are located near railways, highways, and coastlines.
    - Whether launch sites maintain a certain distance from cities.

Build a Dashboard with Plotly Dash

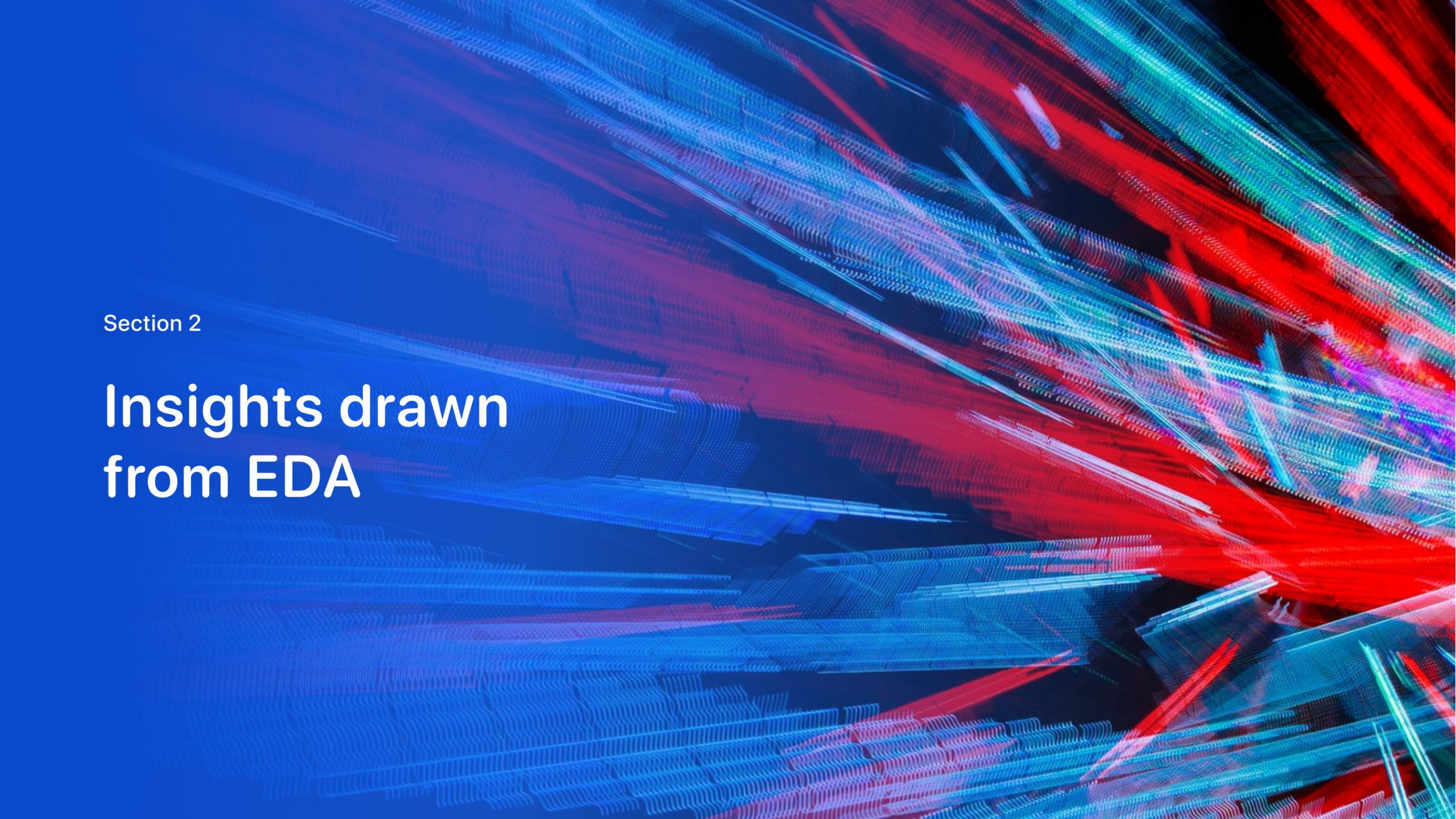
- We built an interactive dashboard using Plotly Dash. The dashboard includes:
- Pie charts displaying the total number of launches at each site.
- Scatter plots illustrating the relationship between launch outcomes and payload mass (kg) for different booster versions.
  - You can access the notebook via the following link https://github.com/Marchesiello/IBM-Data-Science-Capstone-SpaceX/blob/main/Plots/app.py

Predictive Analysis (Classification)

- We loaded the data using numpy and pandas, then transformed and split it into training and testing sets.
- We built various machine learning models and fine-tuned hyperparameters using GridSearchCV.
- Accuracy was used as the performance metric.
- Through feature engineering and algorithm tuning, we improved the model and identified the best-performing classification model.
    - You can access the notebook via the following link https://github.com/Marchesiello/IBM-Data-Science-Capstone-SpaceX/blob/main/Plots/Machine%20Learning%20Prediction.ipynb

Results

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

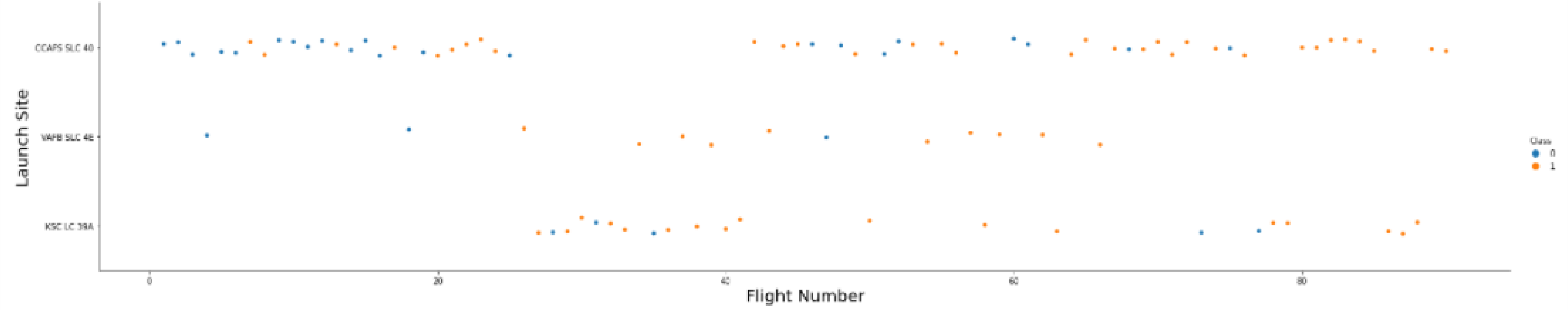


Section 2

Insights drawn from EDA

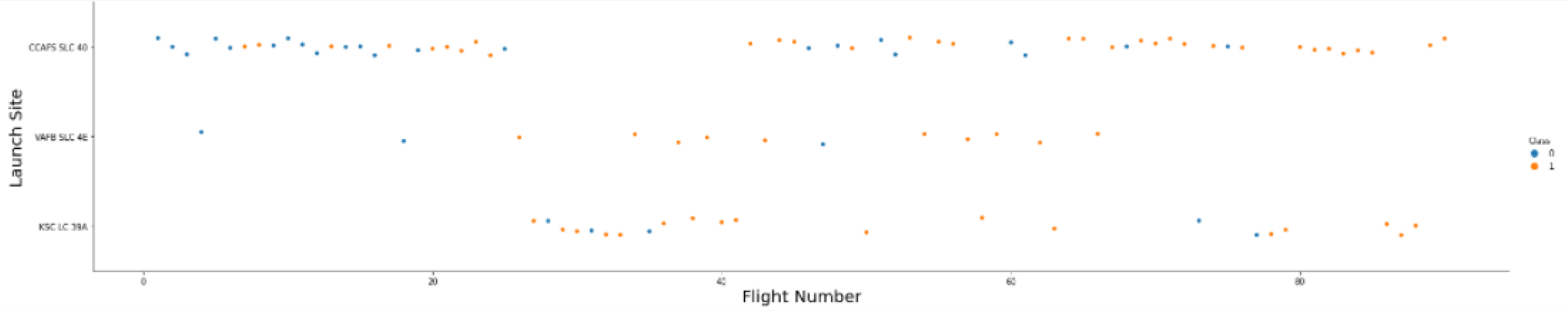
Flight Number vs. Launch Site

- The plot revealed that higher flight volumes at a launch site correlate with increased success rates.



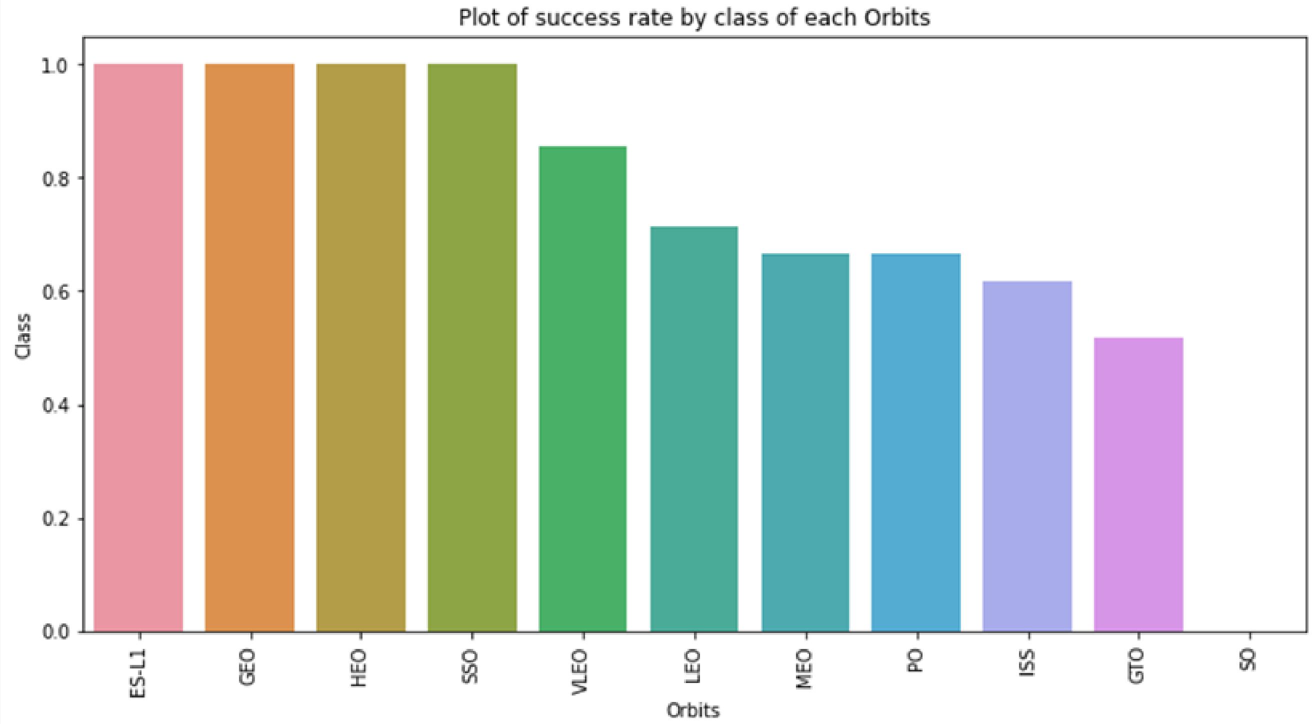
Payload vs. Launch Site Analysis

The greater the payload mass for launch site CCAFS SLC 40 the higher the success rate for the rocket.



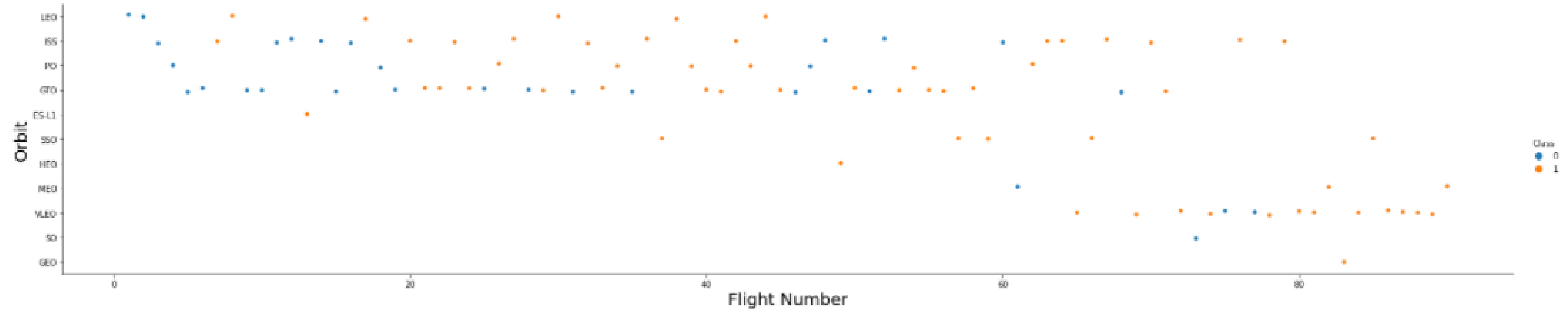
Success Rate vs. Orbit Type

- The plot indicates that ES-L1, GEO, HEO, SSO, and VLEO had the highest success rates.



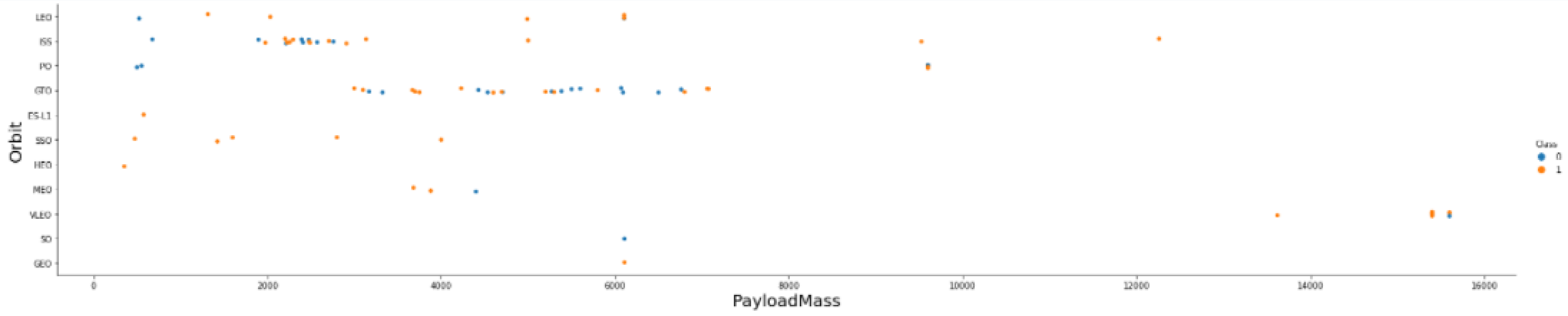
Flight Number vs. Orbit Type

- The plot below illustrates the relationship between flight number and orbit type. We observe that for the LEO orbit, success is correlated with the number of flights, whereas for the GTO orbit, there is no discernible relationship between flight number and orbit type.



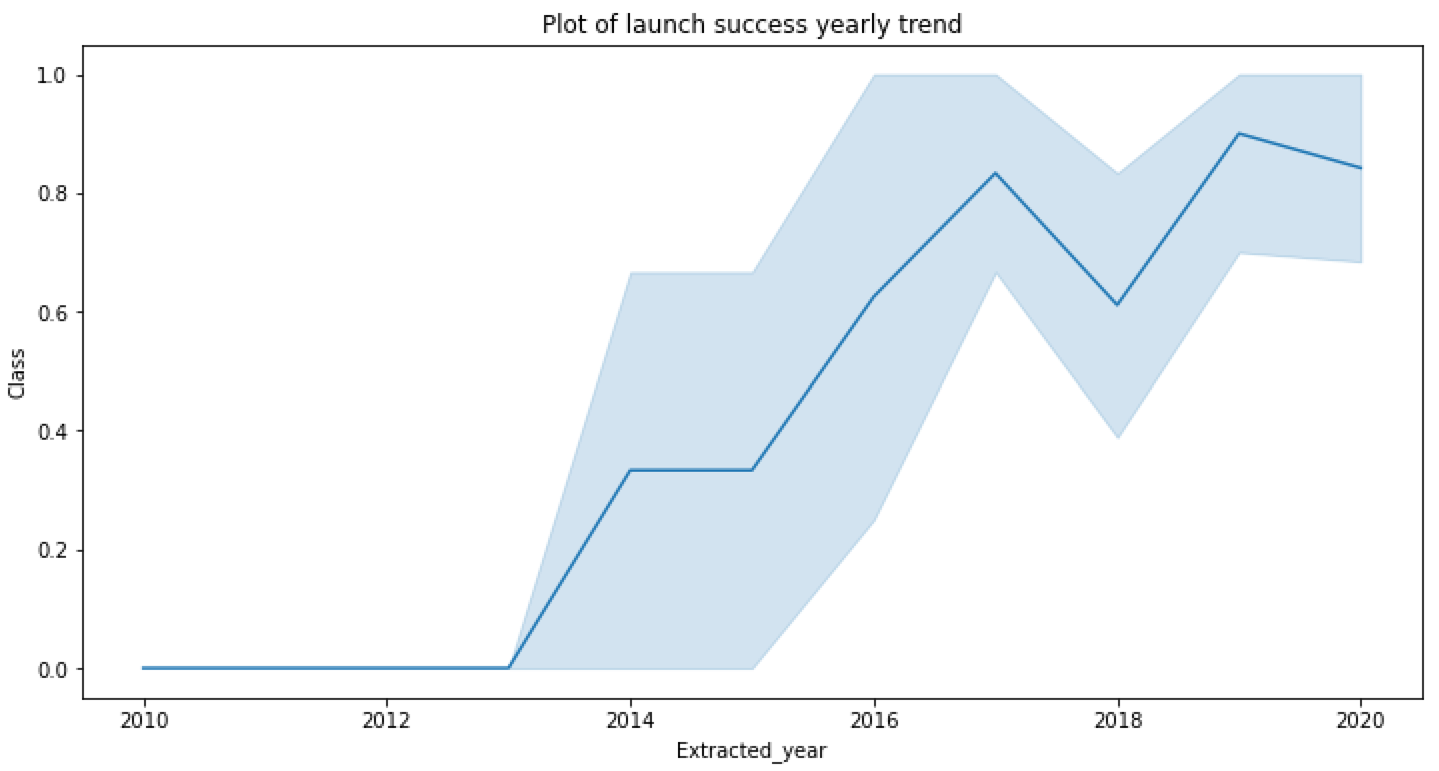
Payload vs. Orbit Type

- We notice that successful landings are more frequent for payloads in PO, LEO, and ISS orbits, especially those with heavier payloads.



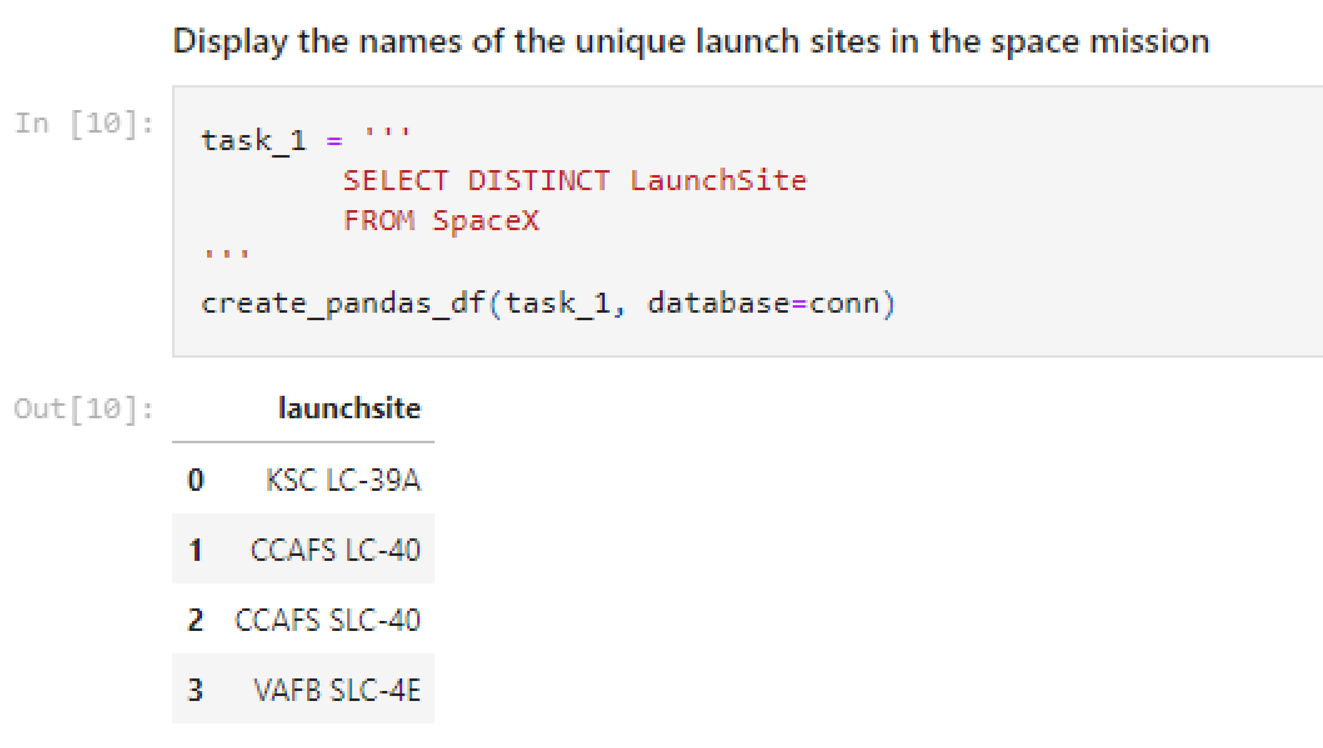
Launch Success Yearly Trend

- According to the plot, the success rate has consistently increased from 2013 to 2020.



All Launch Site Names

- We utilized the keyword DISTINCT to display only unique launch sites from the SpaceX data.



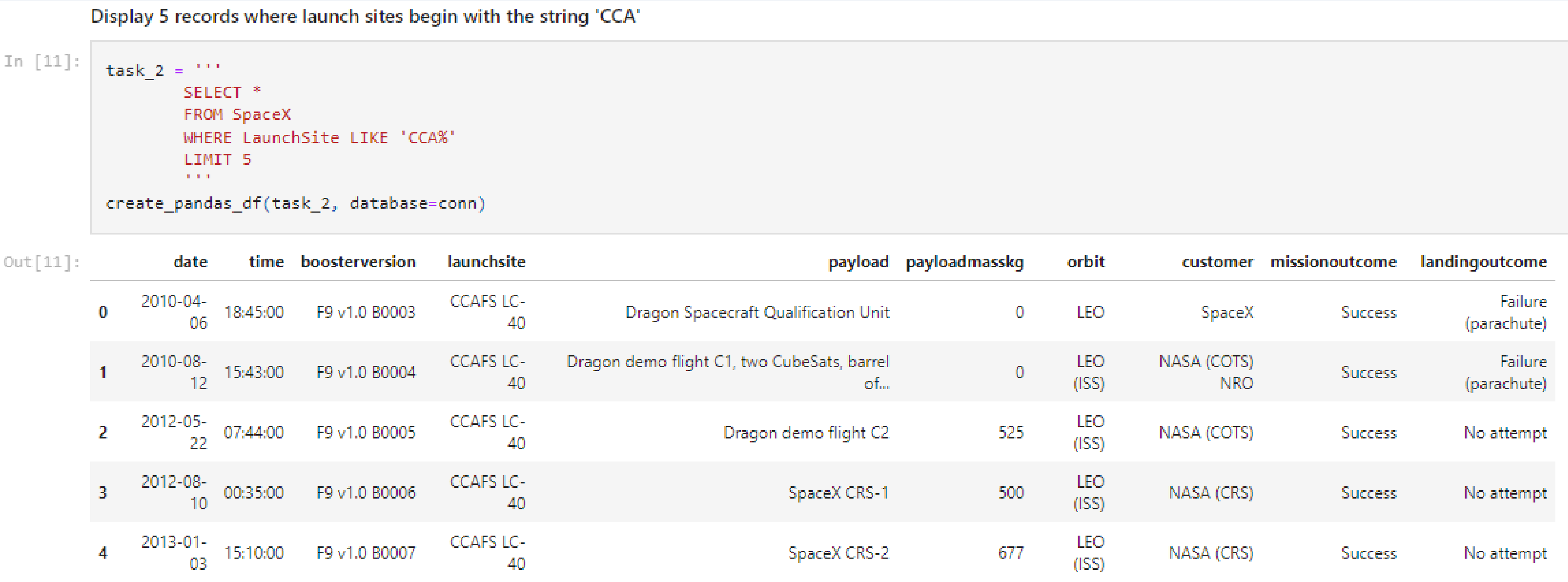
Display the names of the unique launch sites in the space mission

```
In [10]: task_1 = '''
                 SELECT DISTINCT LaunchSite
                 FROM SpaceX
         '''
         create_pandas_df(task_1, database=conn)
```

Out[10]:

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

Launch Site Names Begin with 'CCA'



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

```
In [11]:
task_2 = '''
        SELECT *
        FROM SpaceX
        WHERE LaunchSite LIKE 'CCA%'
        LIMIT 5
        '''
create_pandas_df(task_2, database=conn)
```

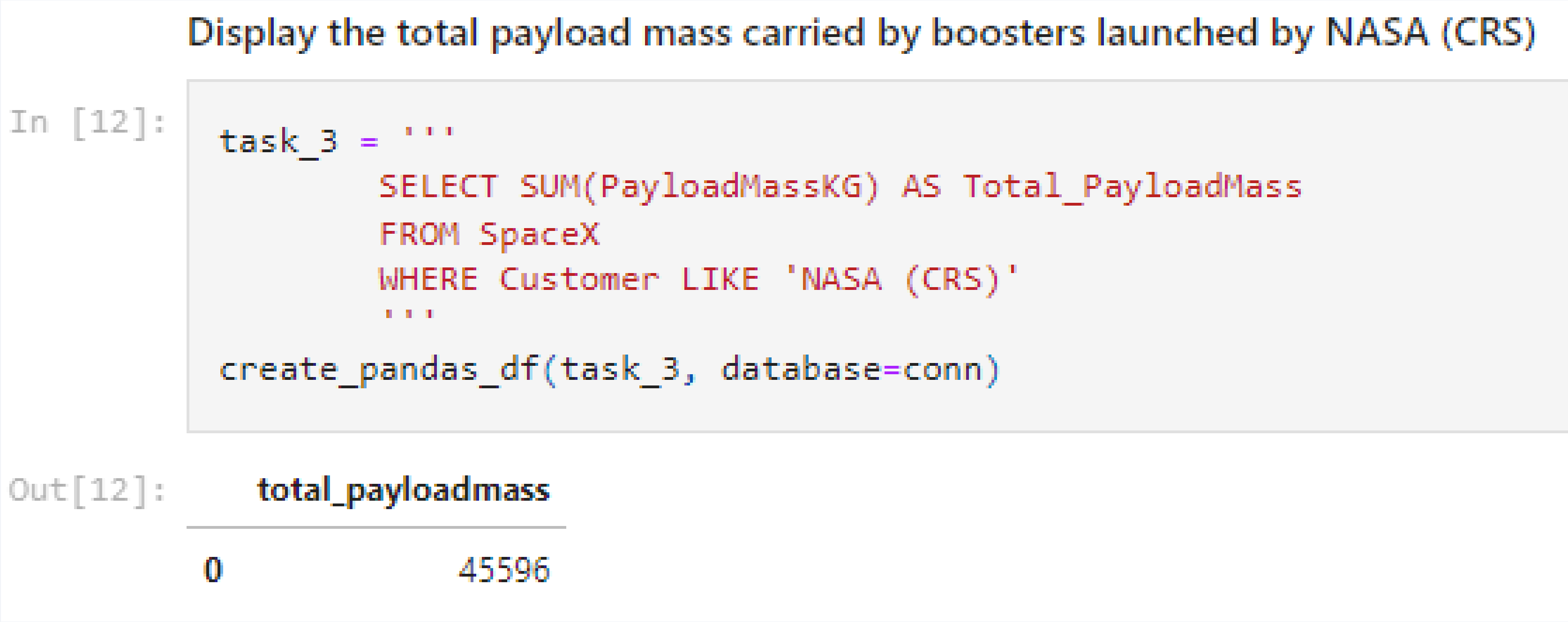
Out[11]:

	date	time	boosterversion	launchsite	payload	payloadmasskg	orbit	customer	missionoutcome	landingoutcome
0	2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	2010-08-12	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	2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
3	2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

- We employed the above query to retrieve 5 records where launch sites start with 'CCA'.

Total Payload Mass

- We determined that the total payload carried by NASA boosters was 45,596 using the following query:



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

In [12]:

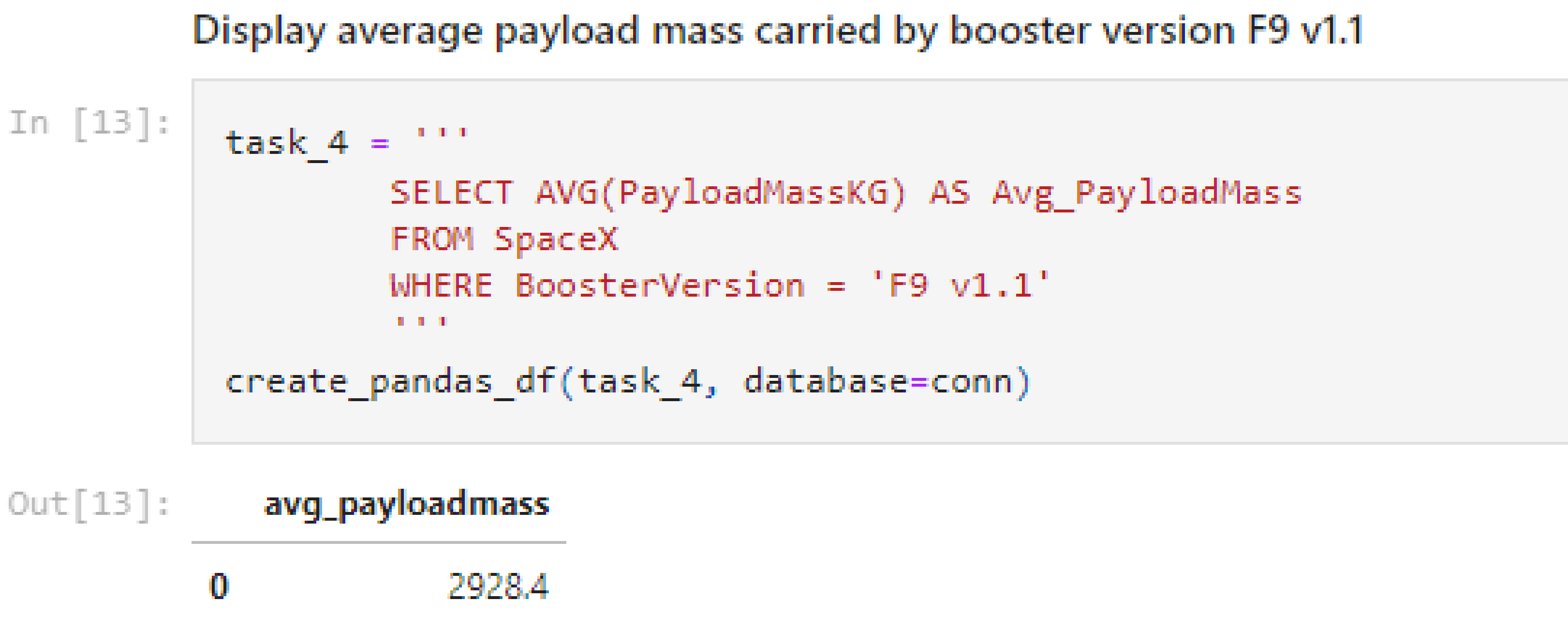
```
task_3 = '''
        SELECT SUM(PayloadMassKG) AS Total_PayloadMass
        FROM SpaceX
        WHERE Customer LIKE 'NASA (CRS)'
        '''
create_pandas_df(task_3, database=conn)
```

Out[12]:

```
   total_payloadmass
0              45596
```

Average Payload Mass by F9 v1.1

- We found that the average payload mass carried by the F9 v1.1 booster was 2,928.4.



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

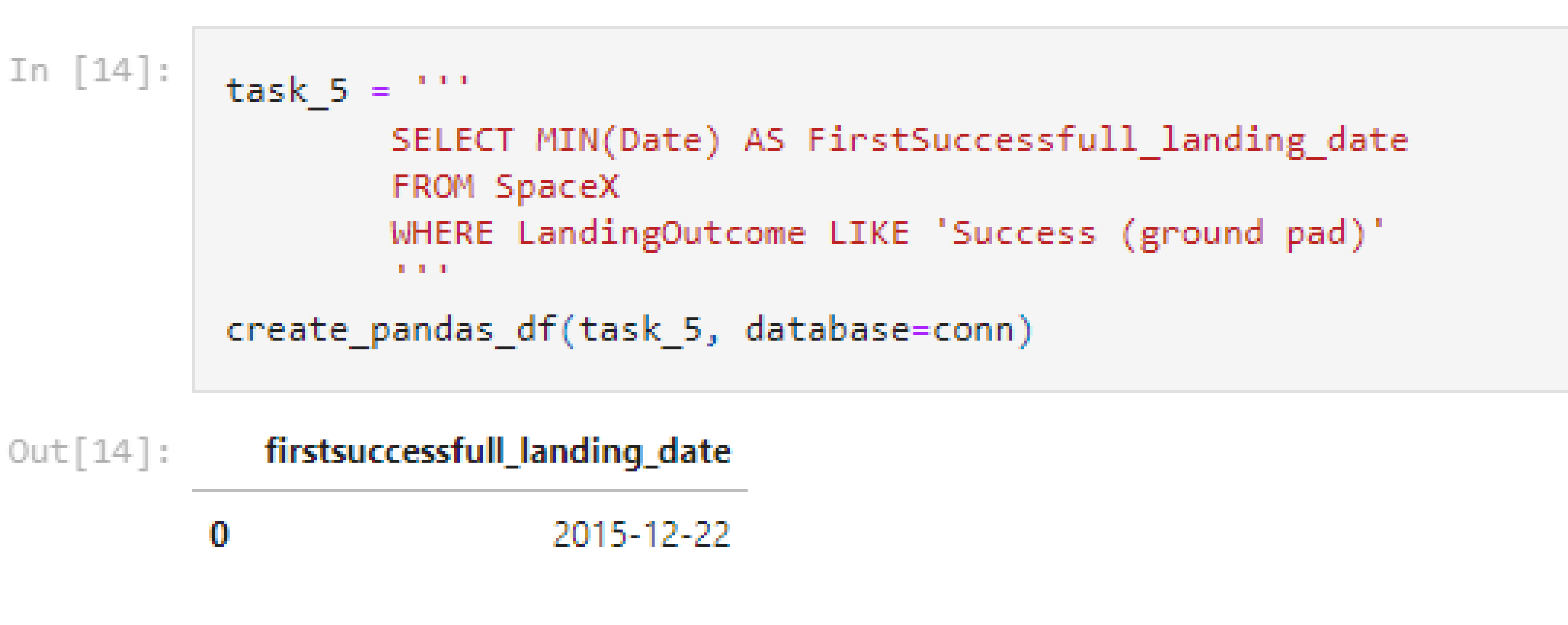
```
In [13]:
task_4 = '''
        SELECT AVG(PayloadMassKG) AS Avg_PayloadMass
        FROM SpaceX
        WHERE BoosterVersion = 'F9 v1.1'
        '''
create_pandas_df(task_4, database=conn)
```

Out[13]:

	avg_payloadmass
0	2928.4

First Successful Ground Landing Date

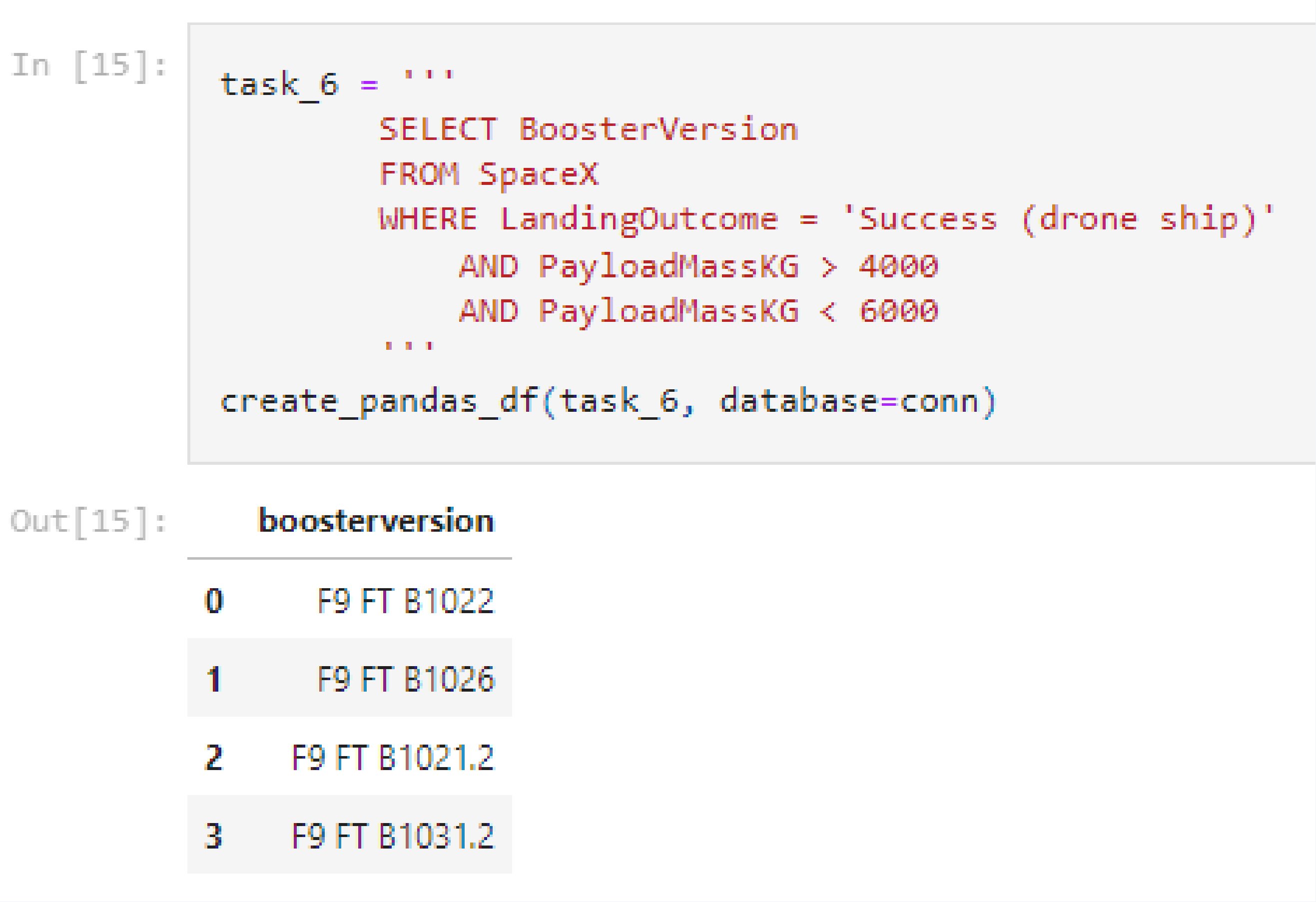
- We observed that the first successful landing on a ground pad occurred on 22nd December 2015.



```
In [14]:  task_5 = '''
                  SELECT MIN(Date) AS FirstSuccessfull_landing_date
                  FROM SpaceX
                  WHERE LandingOutcome LIKE 'Success (ground pad)'
                  '''
          create_pandas_df(task_5, database=conn)
```

```
Out[14]:      firstsuccessfull_landing_date
          0                      2015-12-22
```

Successful drone ship landings with payloads ranging between 4000 and 6000



In [15]:

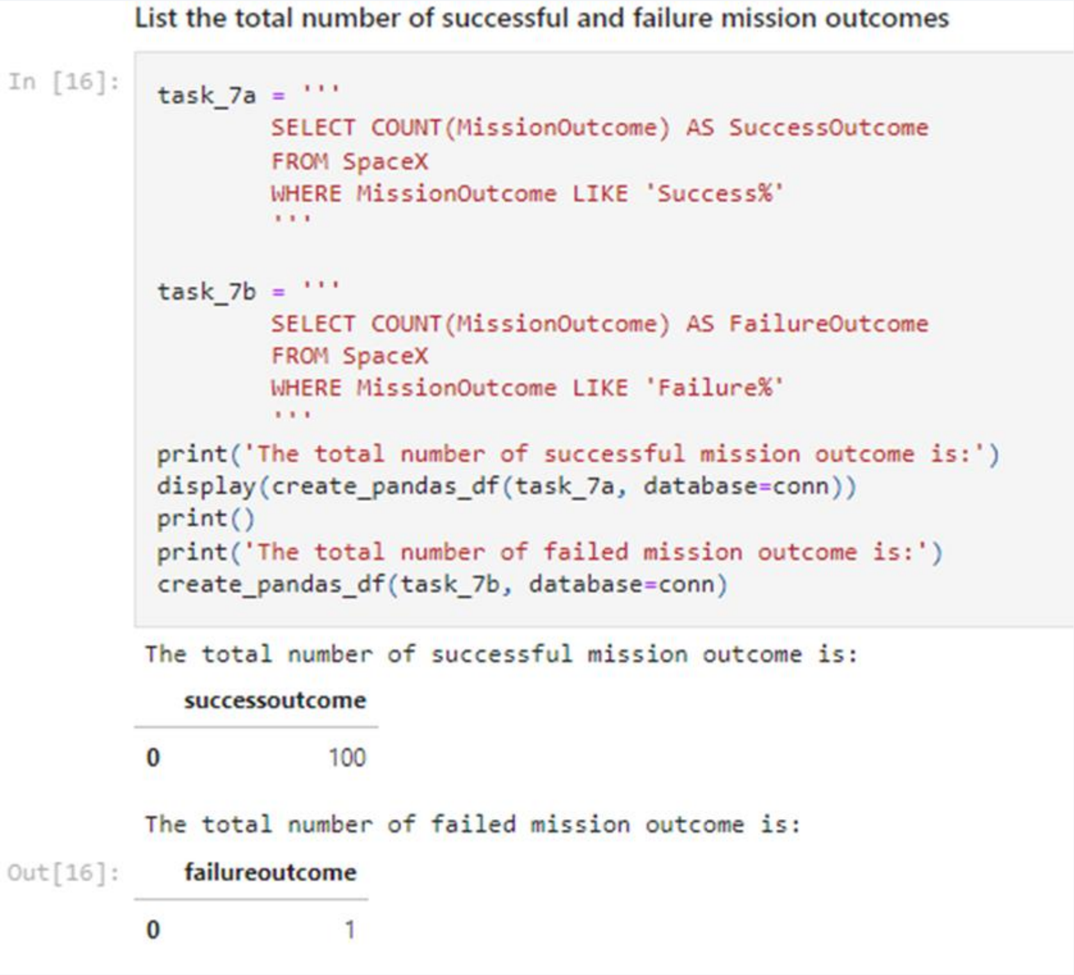
```
task_6 = '''
        SELECT BoosterVersion
        FROM SpaceX
        WHERE LandingOutcome = 'Success (drone ship)'
            AND PayloadMassKG > 4000
            AND PayloadMassKG < 6000
        '''
create_pandas_df(task_6, database=conn)
```

Out[15]:

	boosterversion
0	F9 FT B1022
1	F9 FT B1026
2	F9 FT B1021.2
3	F9 FT B1031.2

- We utilized the WHERE clause to filter boosters that successfully landed on a drone ship and applied the AND condition to identify successful landings with a payload mass between 4000 and 6000.

Total Number of Successful and Failure Mission Outcomes



List the total number of successful and failure mission outcomes

In [16]:

```
task_7a = '''
        SELECT COUNT(MissionOutcome) AS SuccessOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Success%'
        '''

task_7b = '''
        SELECT COUNT(MissionOutcome) AS FailureOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Failure%'
        '''
print('The total number of successful mission outcome is:')
display(create_pandas_df(task_7a, database=conn))
print()
print('The total number of failed mission outcome is:')
create_pandas_df(task_7b, database=conn)
```

```
The total number of successful mission outcome is:
```

	successoutcome
0	100

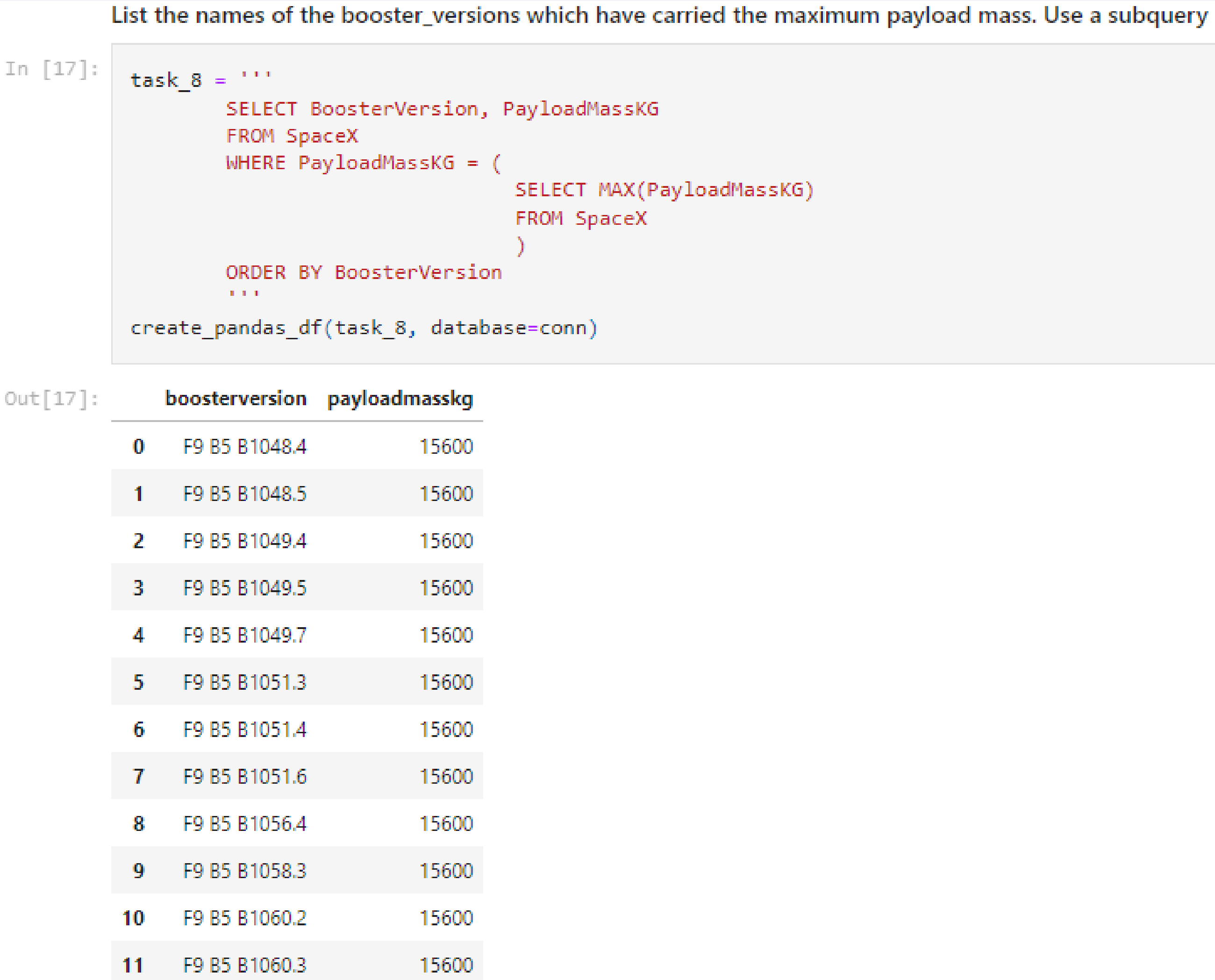
```
The total number of failed mission outcome is:
```

Out[16]:

	failureoutcome
0	1

- We used the wildcard '%' to filter for records WHERE the MissionOutcome was either a success or a failure.

Boosters Carried Maximum Payload



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

```
In [17]:  task_8 = '''
                  SELECT BoosterVersion, PayloadMassKG
                  FROM SpaceX
                  WHERE PayloadMassKG = (
                                          SELECT MAX(PayloadMassKG)
                                          FROM SpaceX
                                          )
                  ORDER BY BoosterVersion
                  '''
          create_pandas_df(task_8, database=conn)
```

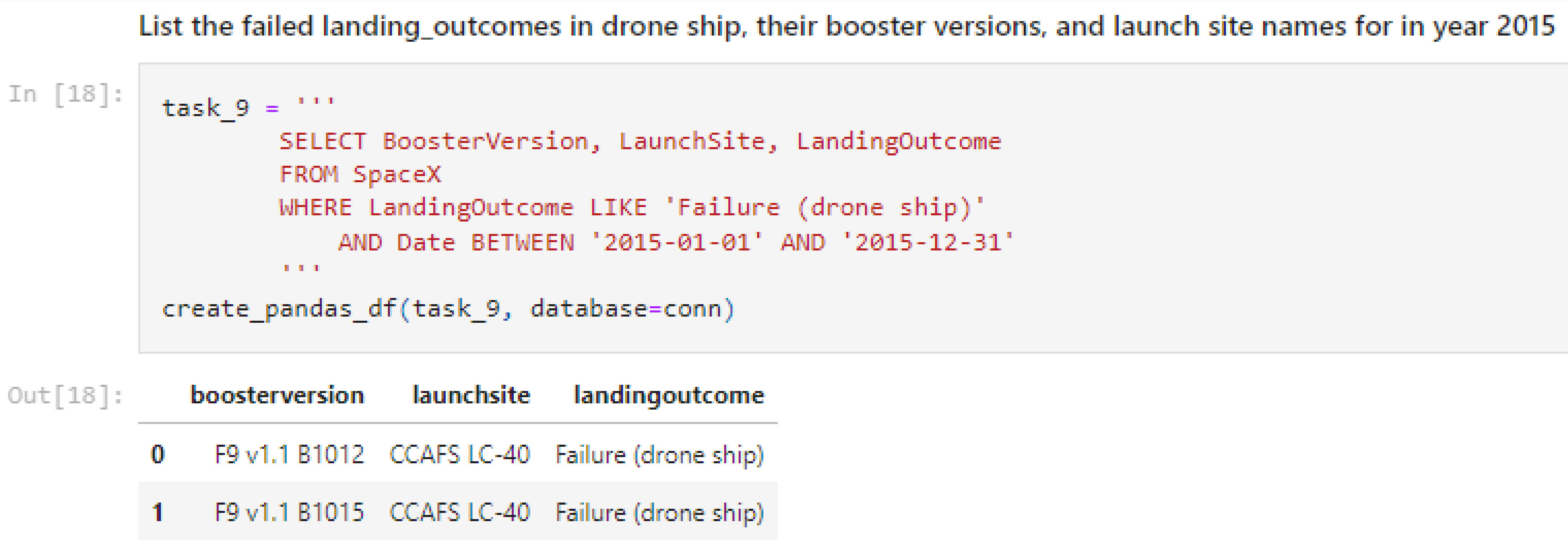
Out[17]:

	boosterversion	payloadmasskg
0	F9 B5 B1048.4	15600
1	F9 B5 B1048.5	15600
2	F9 B5 B1049.4	15600
3	F9 B5 B1049.5	15600
4	F9 B5 B1049.7	15600
5	F9 B5 B1051.3	15600
6	F9 B5 B1051.4	15600
7	F9 B5 B1051.6	15600
8	F9 B5 B1056.4	15600
9	F9 B5 B1058.3	15600
10	F9 B5 B1060.2	15600
11	F9 B5 B1060.3	15600

- We identified the booster that carried the maximum payload by using a subquery in the WHERE clause along with the MAX() function.

2015 Launch Records

- We used a combination of the WHERE clause, LIKE, AND, and BETWEEN conditions to filter for failed drone ship landing outcomes, along with their booster versions and launch site names for the year 2015.



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

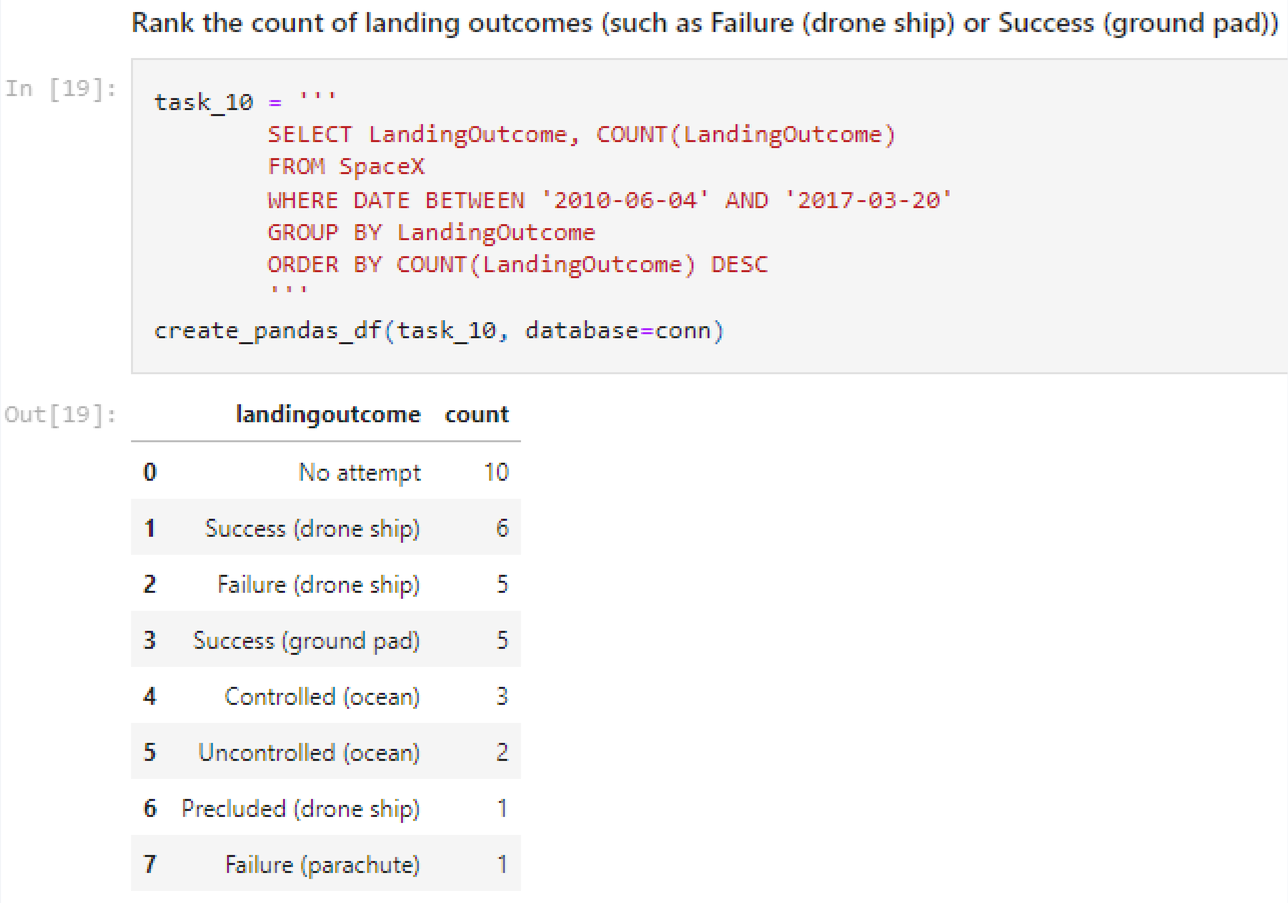
In [18]:

```
task_9 = '''
        SELECT BoosterVersion, LaunchSite, LandingOutcome
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Failure (drone ship)'
            AND Date BETWEEN '2015-01-01' AND '2015-12-31'
        '''
create_pandas_df(task_9, database=conn)
```

Out[18]:

	boosterversion	launchsite	landingoutcome
0	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
1	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

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))

```
In [19]: task_10 = '''
                 SELECT LandingOutcome, COUNT(LandingOutcome)
                 FROM SpaceX
                 WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
                 GROUP BY LandingOutcome
                 ORDER BY COUNT(LandingOutcome) DESC
                 '''
         create_pandas_df(task_10, database=conn)
```

Out[19]:

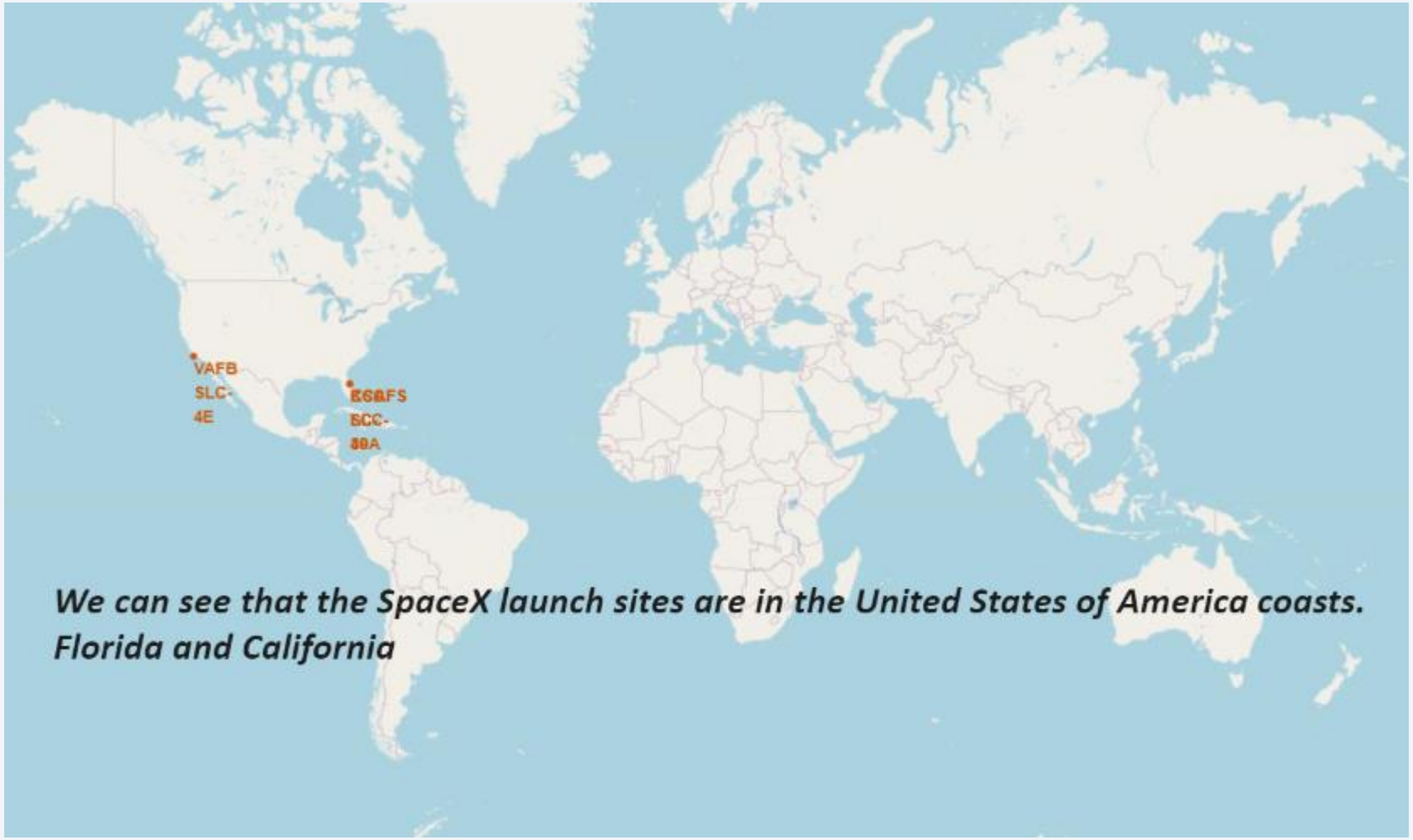
	landingoutcome	count
0	No attempt	10
1	Success (drone ship)	6
2	Failure (drone ship)	5
3	Success (ground pad)	5
4	Controlled (ocean)	3
5	Uncontrolled (ocean)	2
6	Precluded (drone ship)	1
7	Failure (parachute)	1

- We selected landing outcomes and their COUNT from the data, using the WHERE clause to filter for landing outcomes BETWEEN 2010-06-04 and 2010-03-20.
- We applied the GROUP BY clause to group the landing outcomes and the ORDER BY clause to sort these groups in descending order.

Section 4

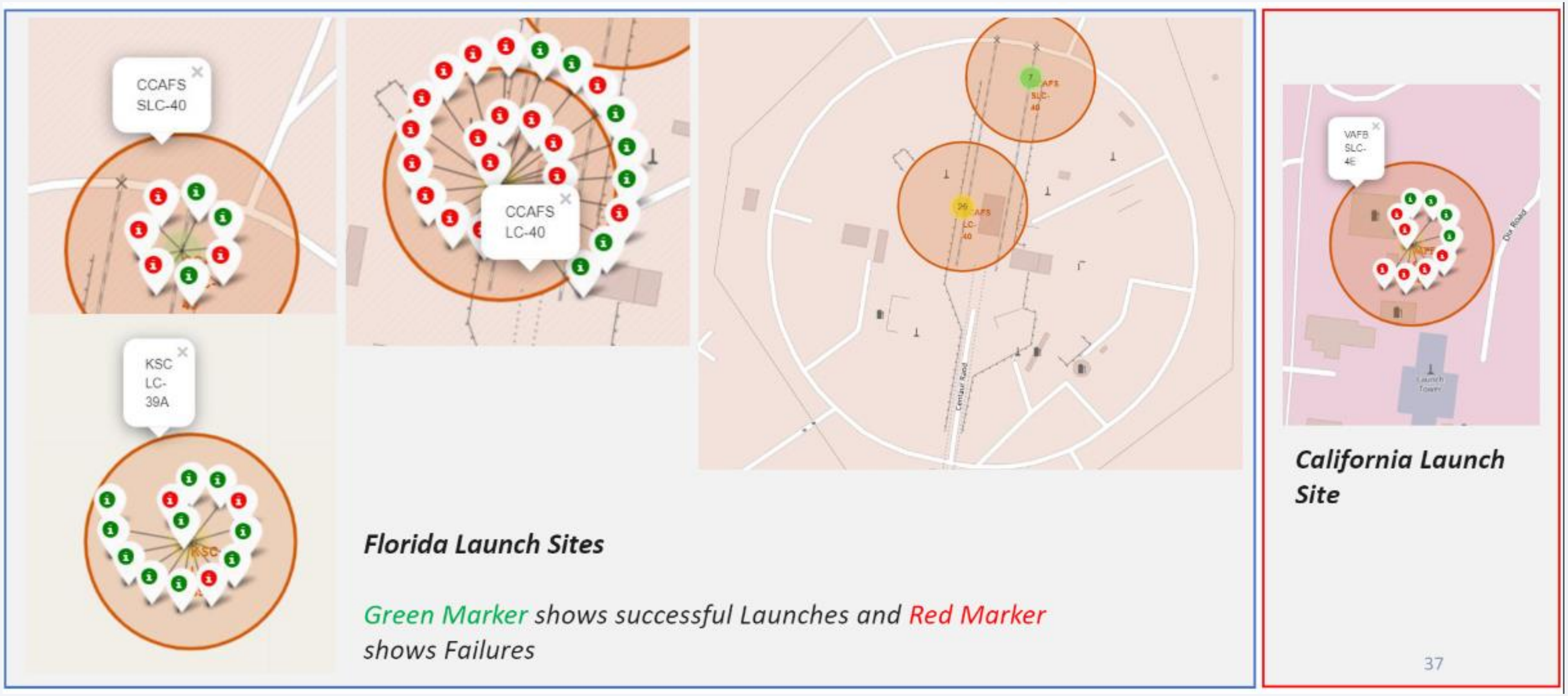
Launch Sites Proximities Analysis

All launch sites global map markers



We can see that the SpaceX launch sites are in the United States of America coasts. Florida and California

Markers showing launch sites with color labels

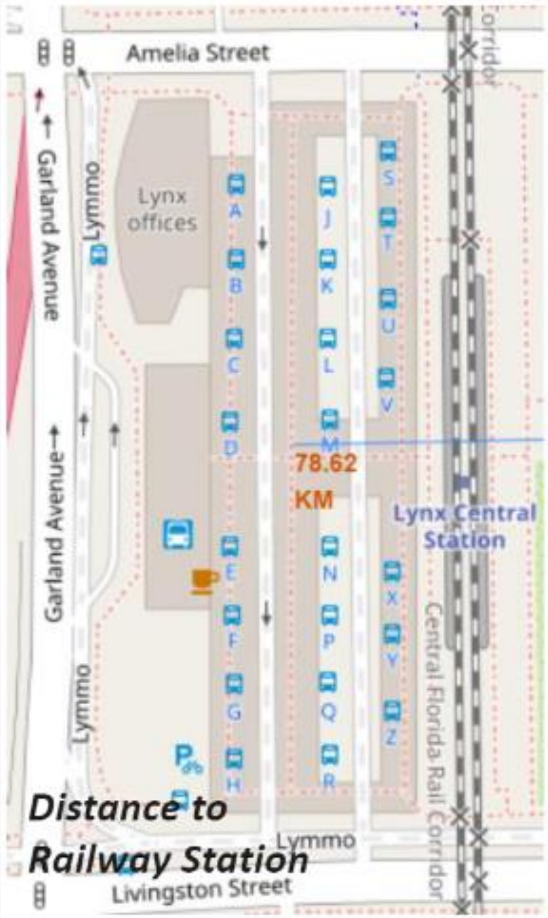


Florida Launch Sites

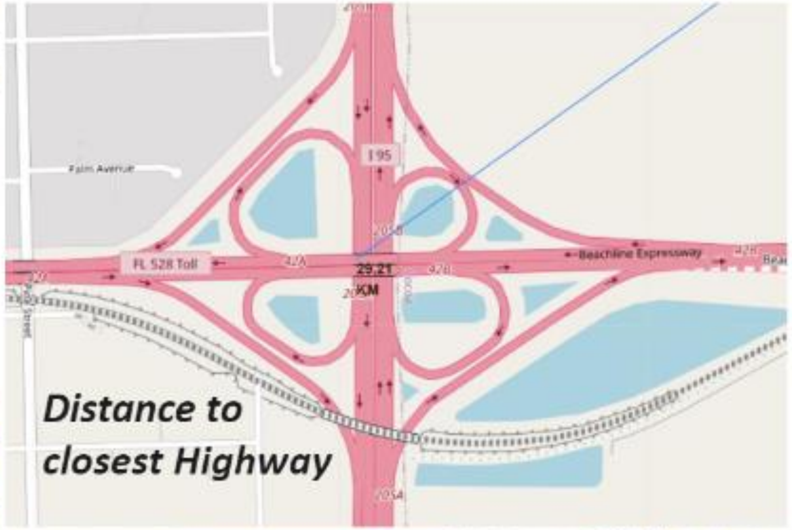
Green Marker shows successful Launches and Red Marker shows Failures

California Launch Site

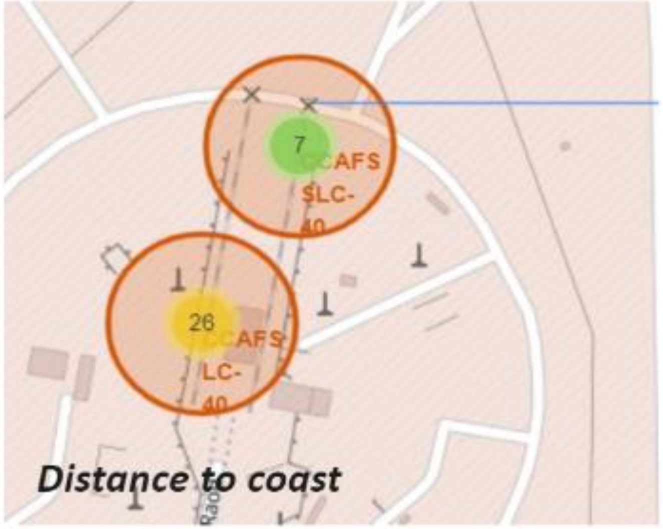
Launch Site distance to landmarks



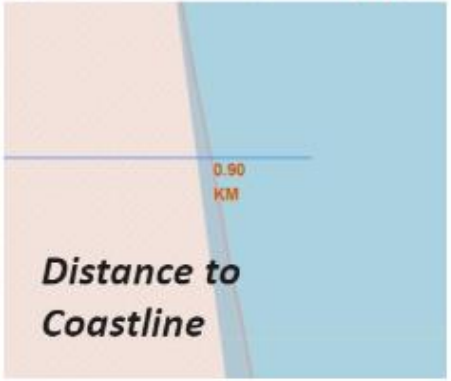
Distance to Railway Station



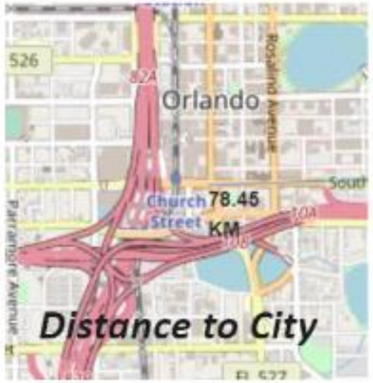
Distance to closest Highway



Distance to coast



Distance to Coastline

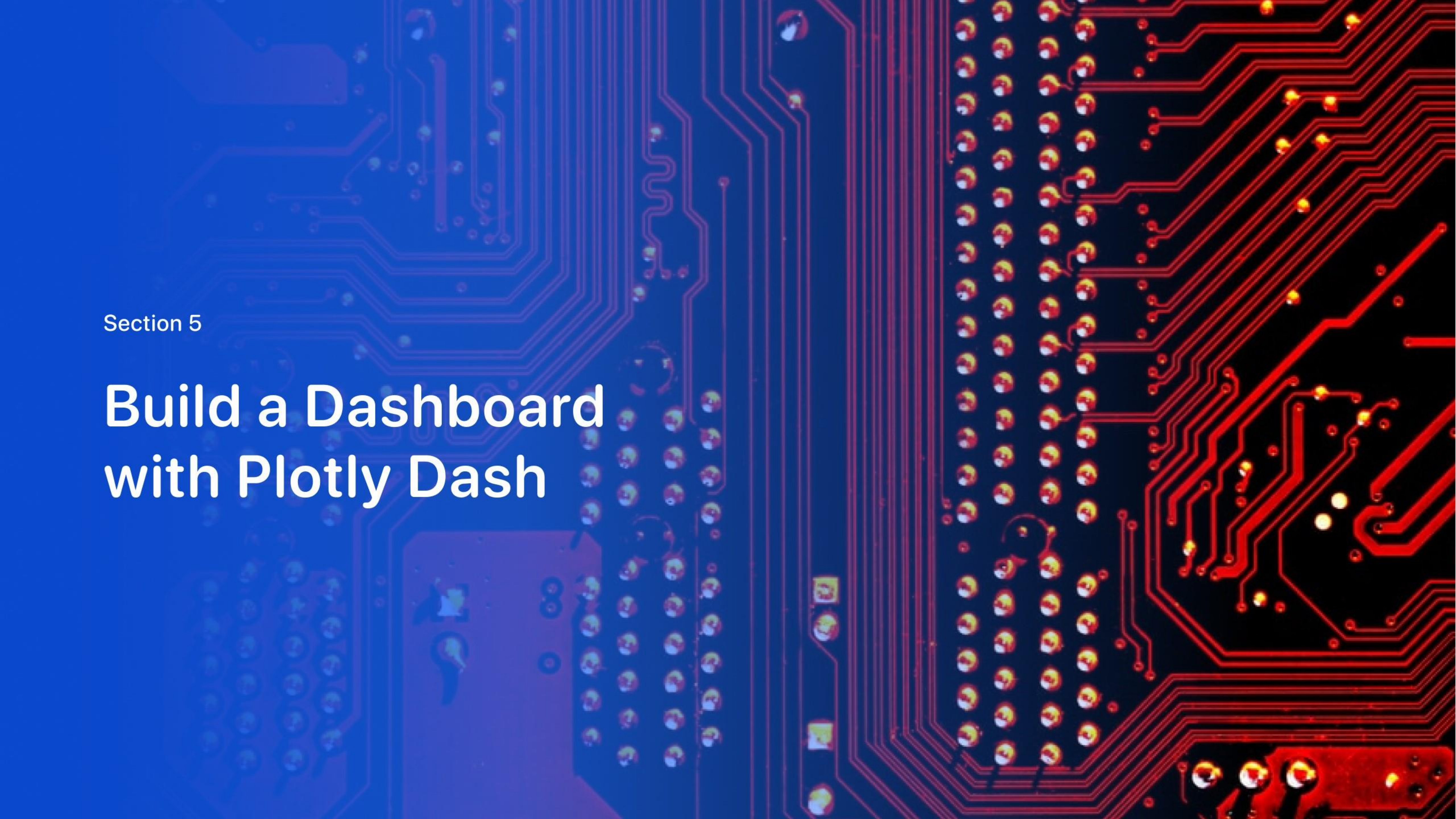


Distance to City

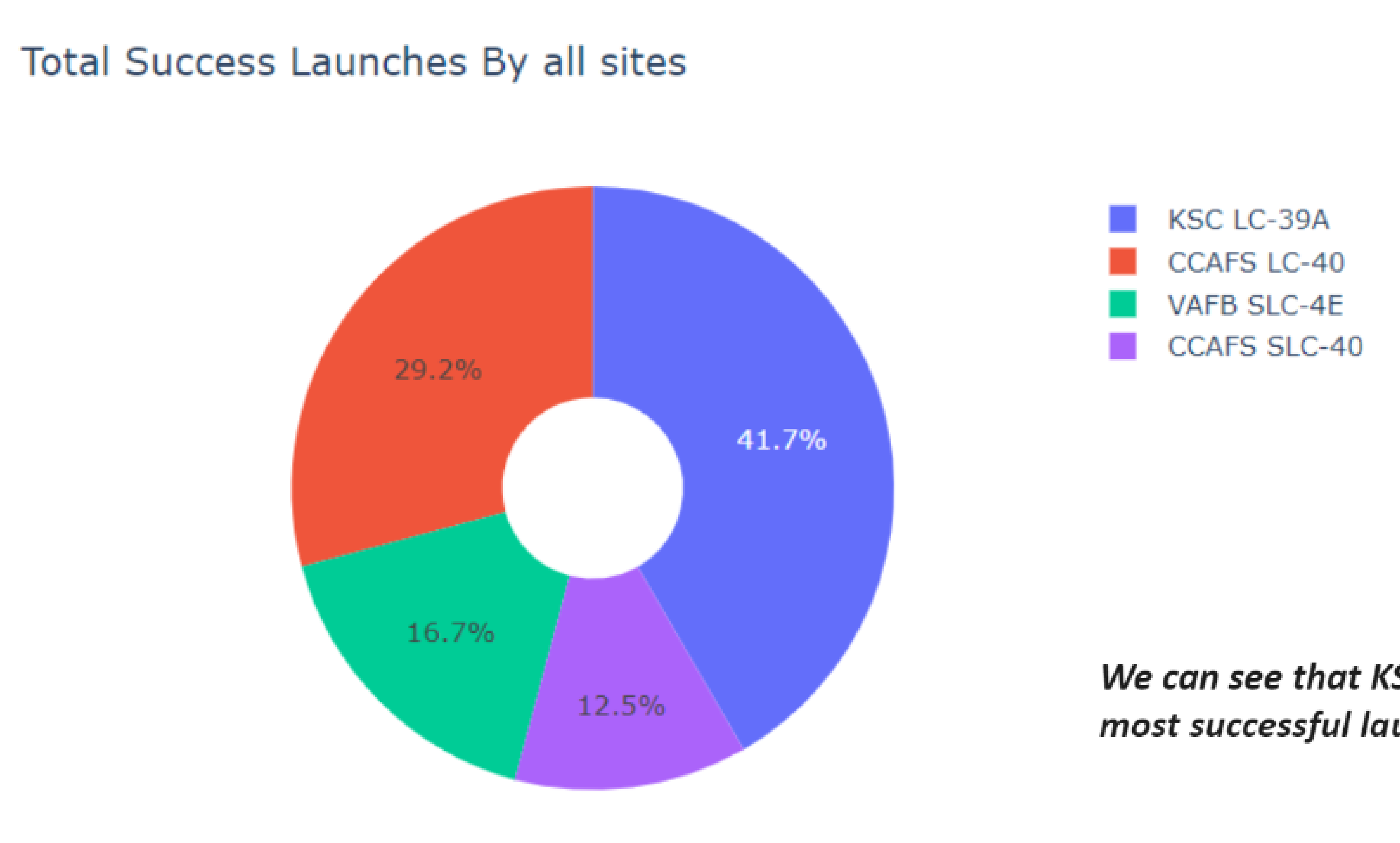
- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

Section 5

Build a Dashboard with Plotly Dash

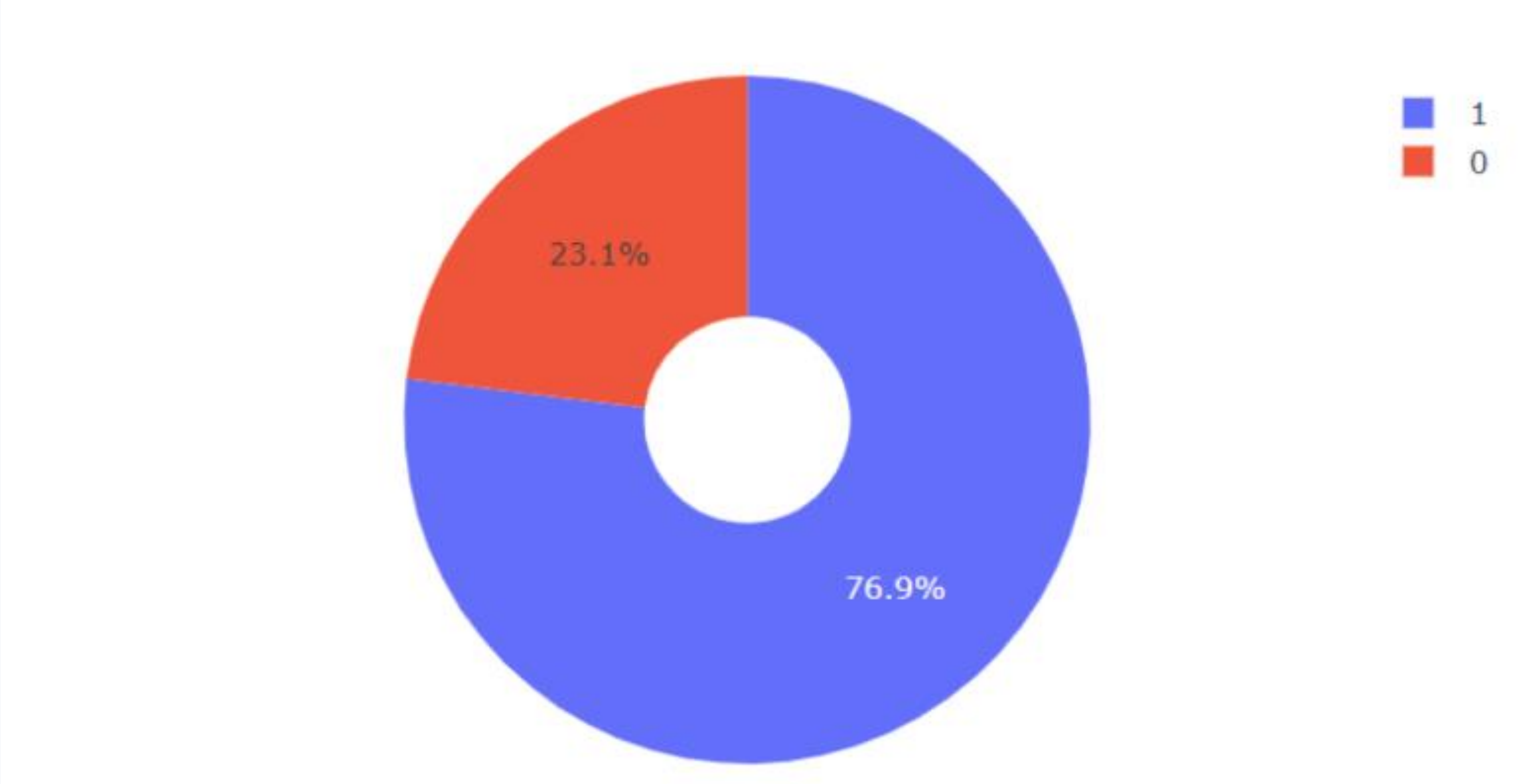


Pie chart displaying the success percentage of each launch site



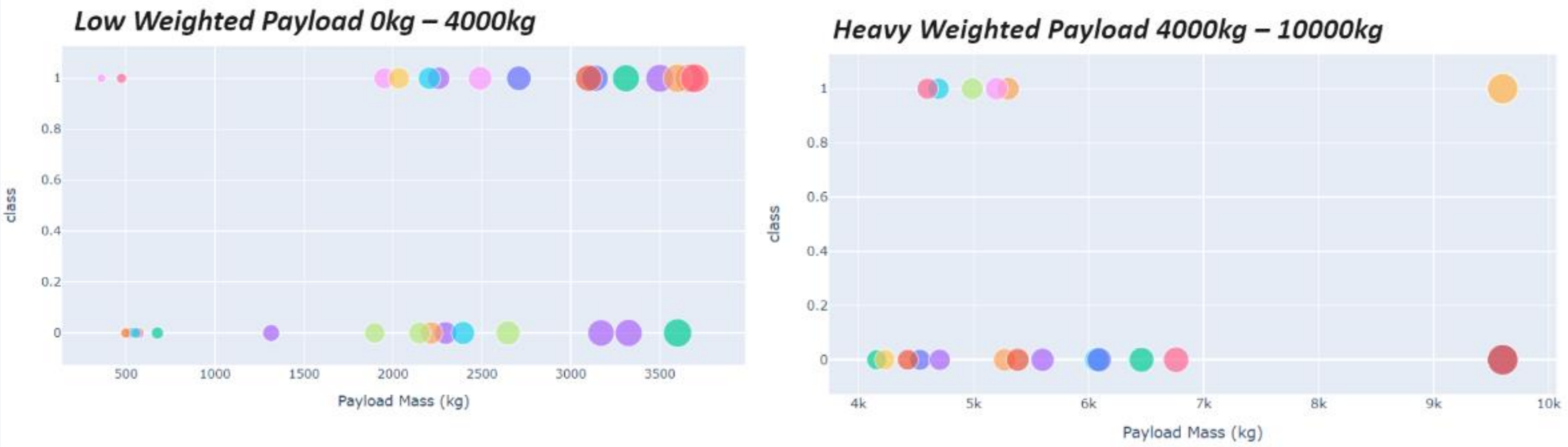
We can see that KSC LC-39A had the most successful launches from all the sites

Pie chart illustrating the launch site with the highest success ratio



KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

Scatter plot of Payload vs Launch Outcome for all sites, with different payload selected in the range slider



Low Weighted Payload 0kg – 4000kg

Heavy Weighted Payload 4000kg – 10000kg

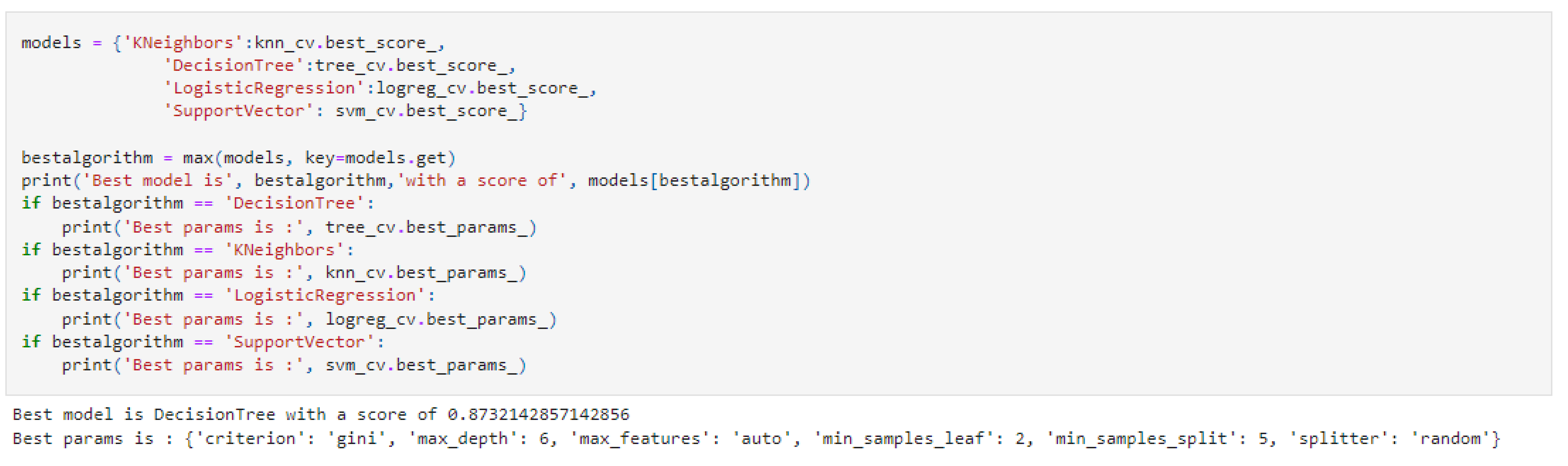
We can see the success rates for low weighted payloads is higher than the heavy weighted payloads

Section 6

Predictive Analysis (Classification)

Classification Accuracy

- The decision tree classifier achieved the highest classification accuracy among the models.



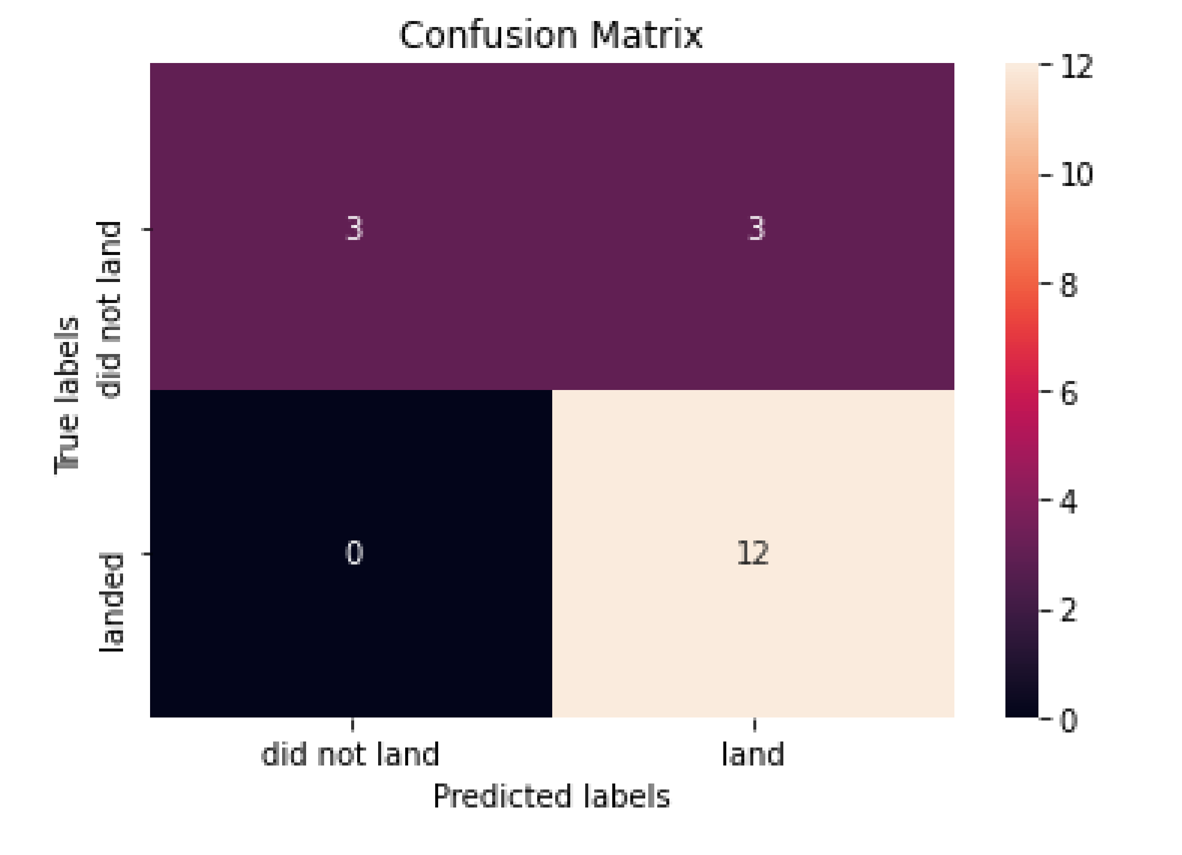
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)
```

```
Best model is DecisionTree with a score of 0.8732142857142856
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```

Confusion Matrix

- The confusion matrix for the decision tree classifier demonstrates its ability to distinguish between different classes. However, the primary issue is the occurrence of false positives, where unsuccessful landings are incorrectly marked as successful by the classifier.



Conclusions

- We can conclude that:
  - The larger the number of flights at a launch site, the higher the success rate.
  - The launch success rate increased steadily from 2013 to 2020.
  - Orbits ES-L1, GEO, HEO, SSO, and VLEO had the highest success rates.
  - KSC LC-39A had the most successful launches of any site.
  - The decision tree classifier is the most effective machine learning algorithm for this task.

Thank you!