IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Chua Zi Long>

<30 May 2025>

Outline

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 promotes Falcon 9 rocket launches on its website at a price of $62 million, significantly lower than competitors, whose launches can exceed $165 million. A major factor in this cost reduction is SpaceX's ability to reuse the rocket's first stage. Therefore, being able to predict whether the first stage will successfully land provides insight into the potential launch cost. This information could be valuable for other companies looking to compete with SpaceX. The objective of this project is to develop a machine learning pipeline that predicts the successful landing of the first stage.

- Problems you want to find answers

  - Which elements influence the likelihood of a rocket landing successfully?
  - How do different features interact to impact the success rate of a landing?
  - What operational conditions must be met to support a reliable landing program?

Section 1

Methodology

Methodology

Executive Summary

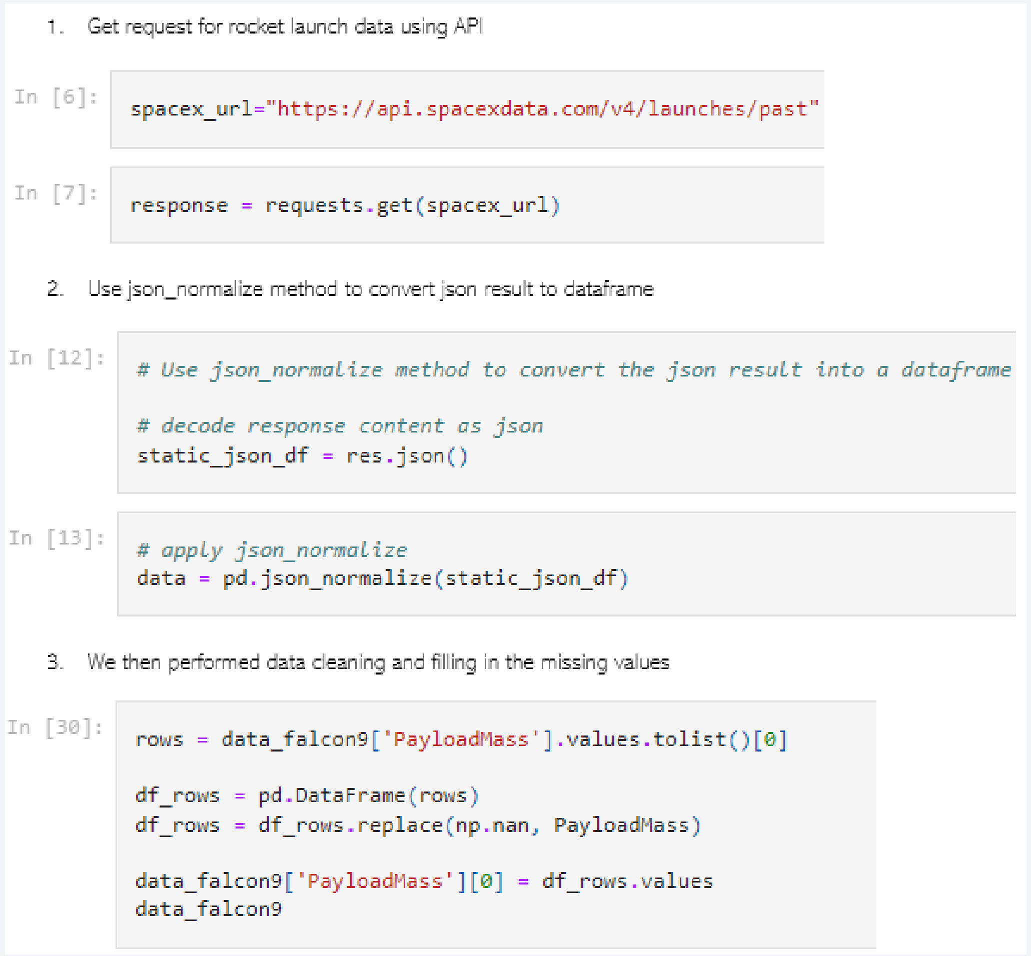
- Data collection methodology:
  - Data was collected using SpaceX API and web scraping from Wikipedia.
- Perform data wrangling
  - One-hot encoding was applied to categorical features
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using 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 was collected by sending a GET request to the SpaceX API. The response content was then decoded into JSON format using the .json() function and converted into a pandas DataFrame using .json_normalize(). After that, the data was cleaned by checking for and handling any missing values. Additionally, we used BeautifulSoup to perform web scraping on Wikipedia to obtain Falcon 9 launch records. The goal was to extract the relevant HTML table, parse its contents, and convert it into a pandas DataFrame for further analysis..

Data Collection – SpaceX API

- We retrieved data from the SpaceX API using a GET request, then cleaned the data and carried out basic wrangling and formatting tasks.



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

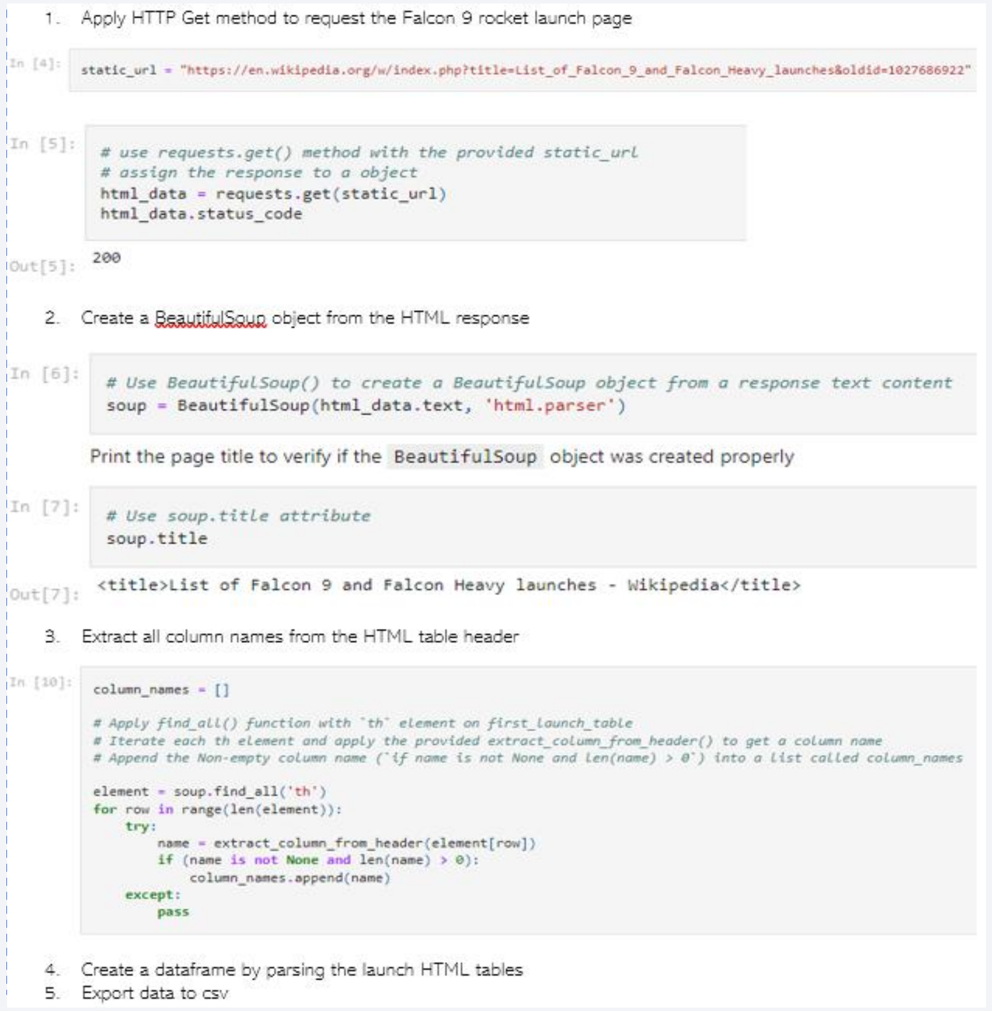
```
In [30]:  rows = data_falcon9['PayloadMass'].values.tolist()[0]

          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 extract Falcon 9 launch records, parsed the table, and converted the data into a pandas DataFrame.



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

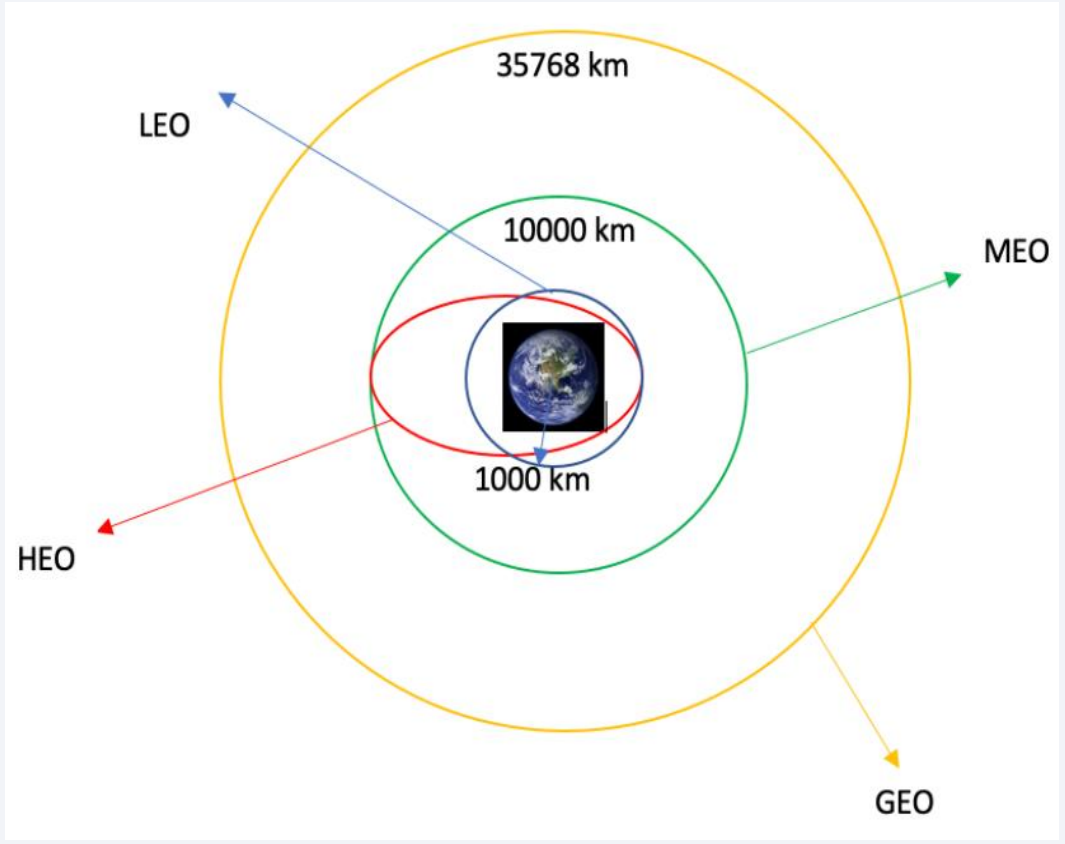
```
In [10]: column_names = []

         # 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

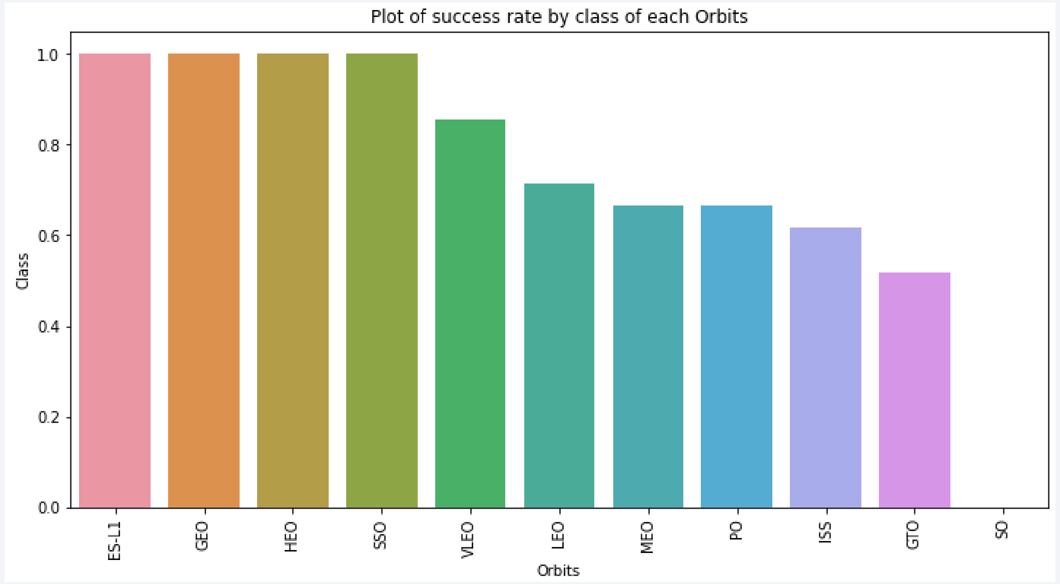
Data Wrangling

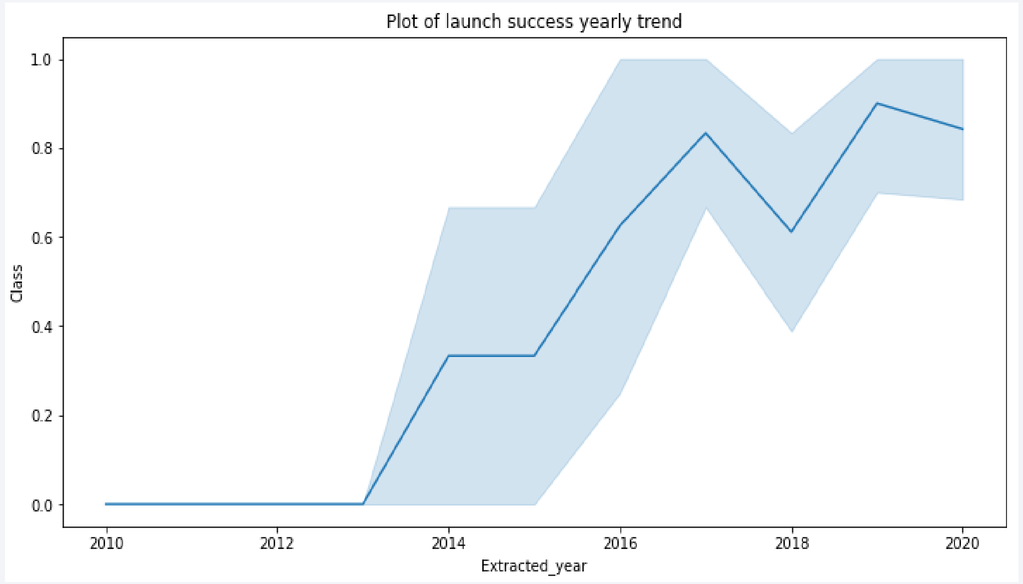


- We conducted exploratory data analysis and identified the training labels. We analyzed the number of launches per site and the frequency of each orbit type. Additionally, we derived the landing outcome labels from the outcome column and exported the processed data to a CSV file.

EDA with Data Visualization

- We examined the data by visualizing various relationships, including flight number versus launch site, payload versus launch site, success rates across different orbit types, flight number versus orbit type, and the yearly trend of launch success.





EDA with SQL

- We imported the SpaceX dataset into a PostgreSQL database directly from within the Jupyter notebook. Using SQL for exploratory data analysis (EDA), we gained insights by writing queries to discover, for example: The distinct names of launch sites used in the missions. The total payload mass carried by NASA (CRS) booster launches. The average payload mass for the F9 v1.1 booster version. The total count of successful and failed mission outcomes. Details of failed drone ship landings, including booster versions and associated launch sites.

Build an Interactive Map with Folium

- We visualized all launch sites on a Folium map, adding map elements such as markers, circles, and lines to indicate the success or failure of launches at each site. Launch outcomes were categorized as binary classes: 0 for failure and 1 for success. Using color-coded marker clusters, we identified which launch sites demonstrated relatively high success rates. We also calculated distances from each launch site to nearby features and addressed questions such as: Are launch sites located near railways, highways, or coastlines? Do launch sites maintain a certain distance from urban areas?

Build a Dashboard with Plotly Dash

- We developed an interactive dashboard using Plotly Dash. The dashboard includes pie charts displaying the total number of launches by specific sites, and scatter plots illustrating the relationship between launch outcomes and payload mass (kg) across different booster versions.

Predictive Analysis (Classification)

- We loaded the data with NumPy and pandas, performed data transformation, and divided the dataset into training and testing sets. We developed several machine learning models and optimized their hyperparameters using GridSearchCV. Accuracy was used as the evaluation metric, and we enhanced the model through feature engineering and algorithm tuning. Ultimately, we identified the best-performing classification model.

Results

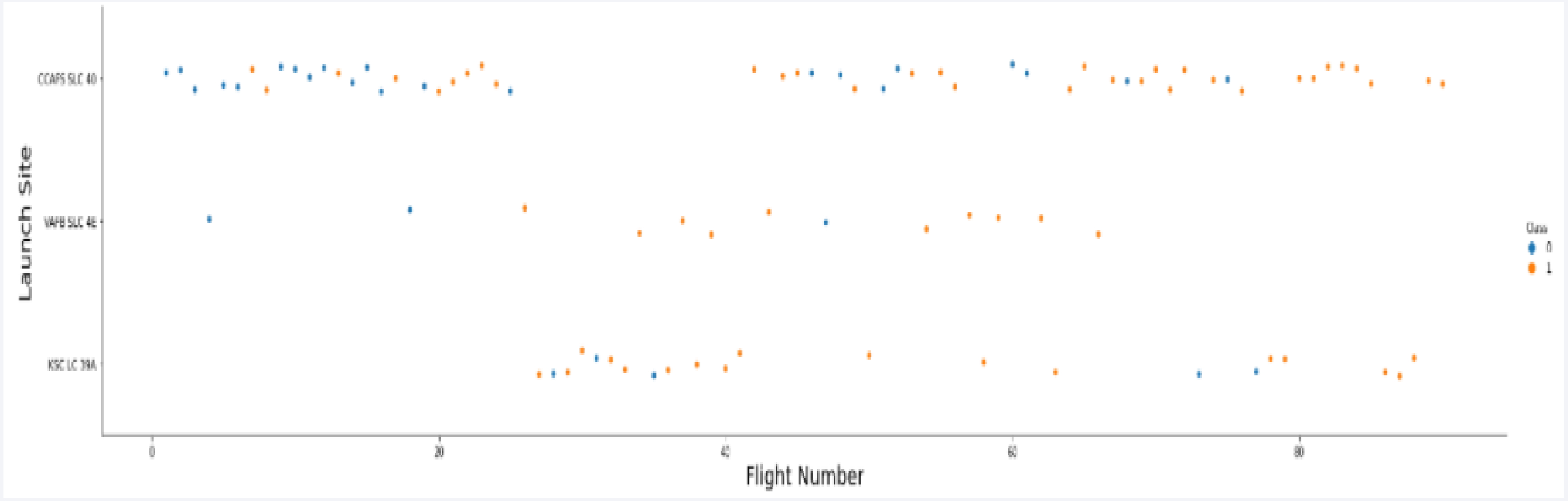
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

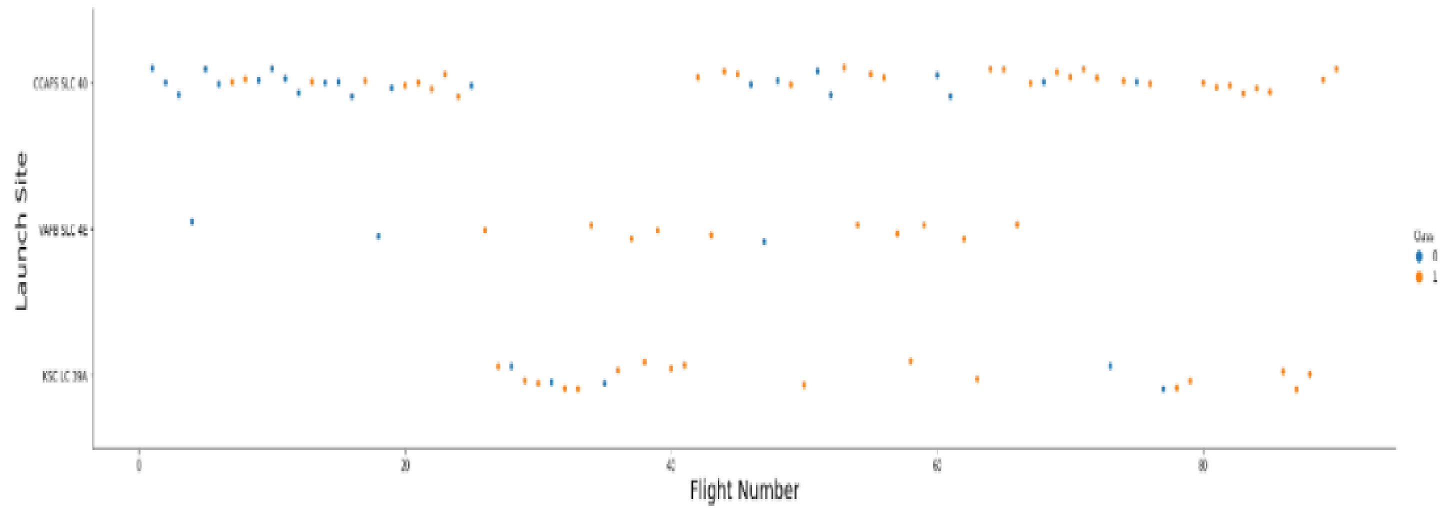
Insights drawn from EDA

Flight Number vs. Launch Site

- The plot revealed that launch sites with a higher number of flights tend to have higher success rates.



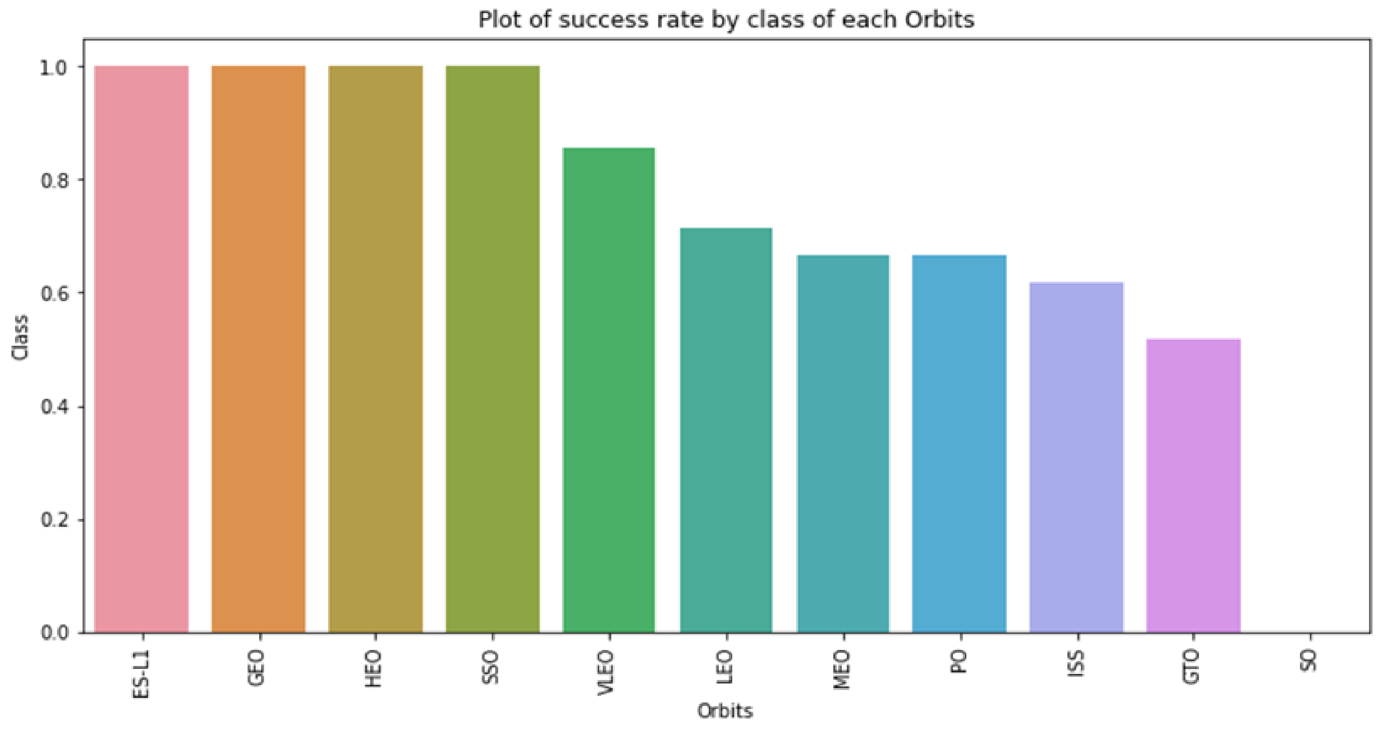
Payload vs. Launch Site



The greater the payload mass, the higher the success rate. (for launch site CCAFS SLC 40)

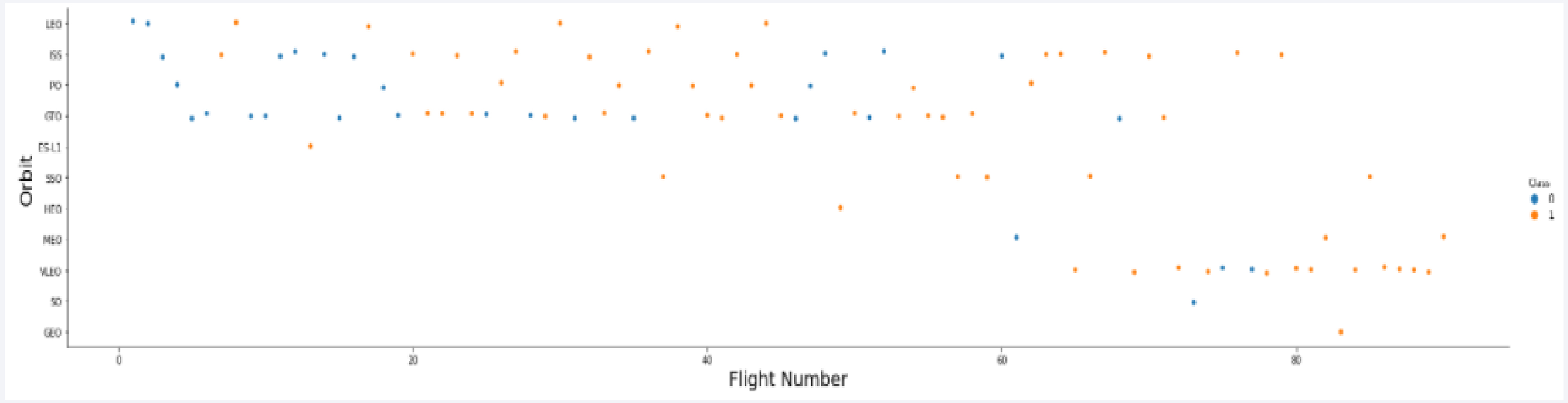
Success Rate vs. Orbit Type

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



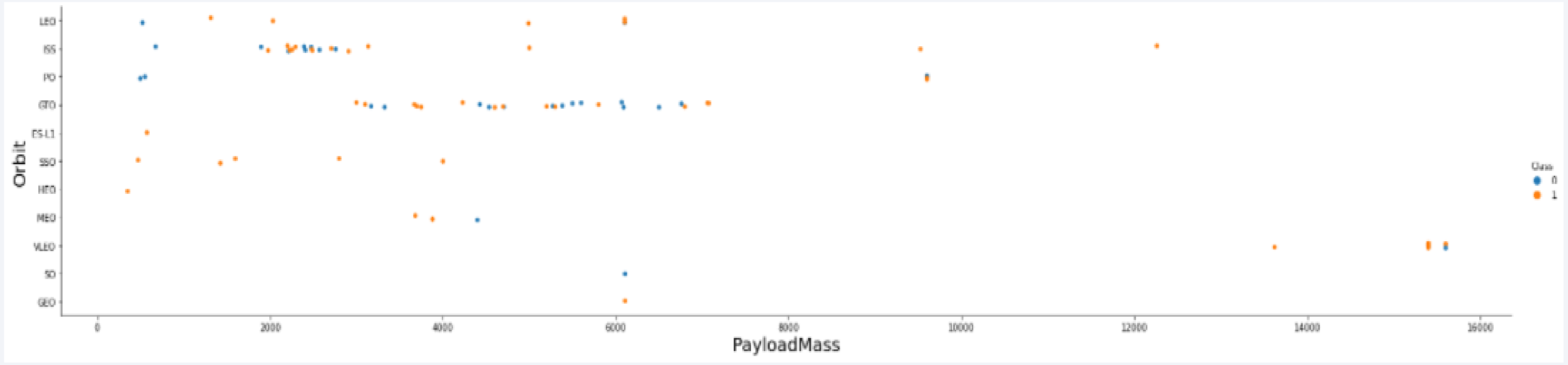
Flight Number vs. Orbit Type

- The plot below illustrates Flight Number versus Orbit Type. We observe that in the LEO orbit, success correlates with the number of flights, while in the GTO orbit, no such relationship exists between flight number and orbit.



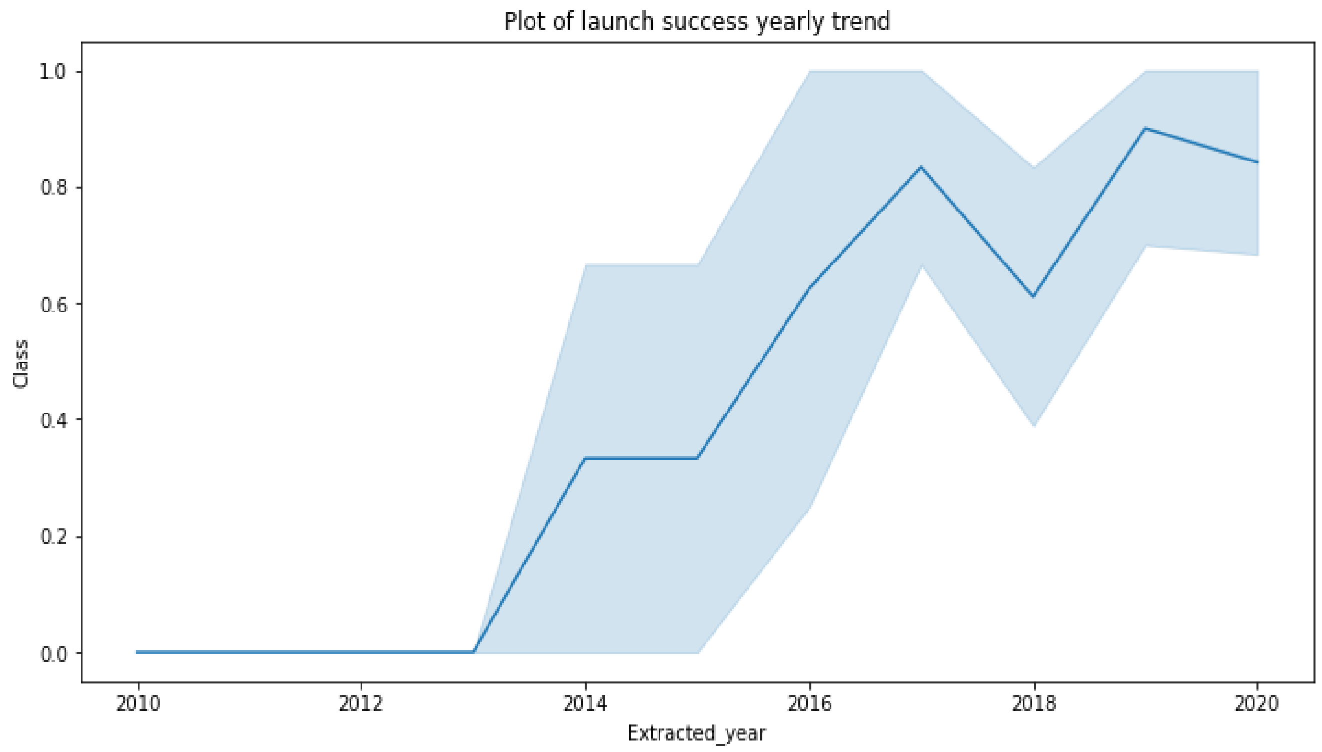
Payload vs. Orbit Type

- We can see that successful landings are more common with heavy payloads in the PO, LEO, and ISS orbits.



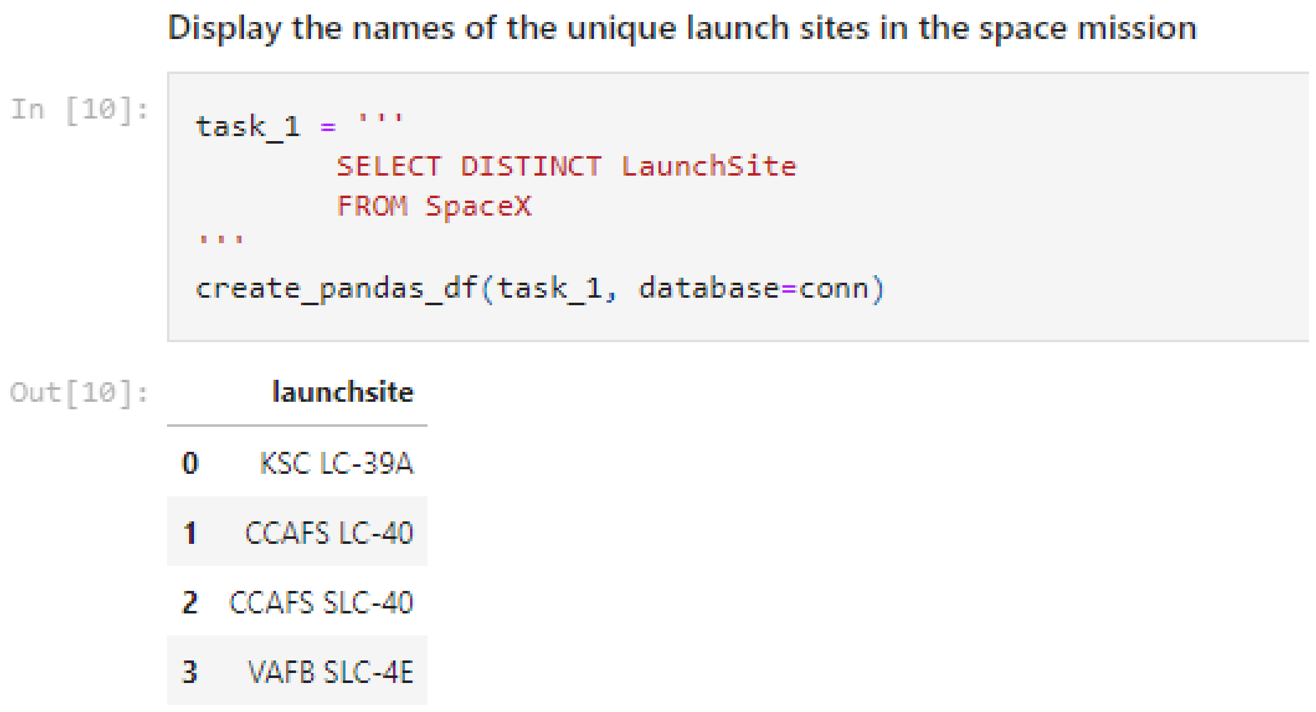
Launch Success Yearly Trend

- The plot shows that the success rate steadily increased from 2013 through 2020.



All Launch Site Names

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

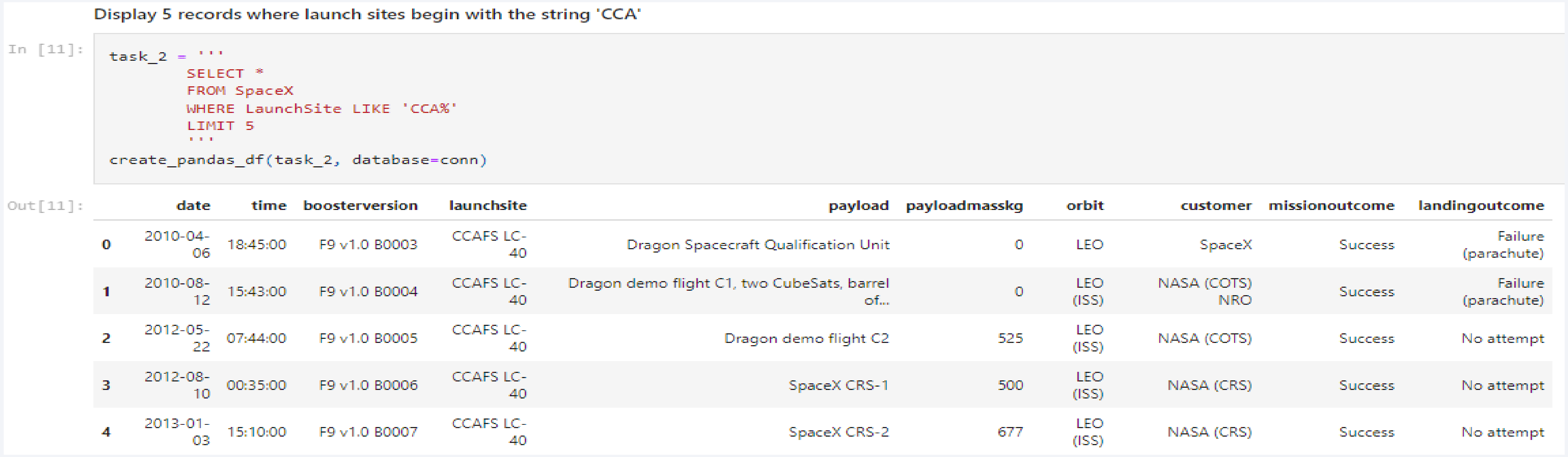


Display the names of the unique launch sites in the space mission

```
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)
```

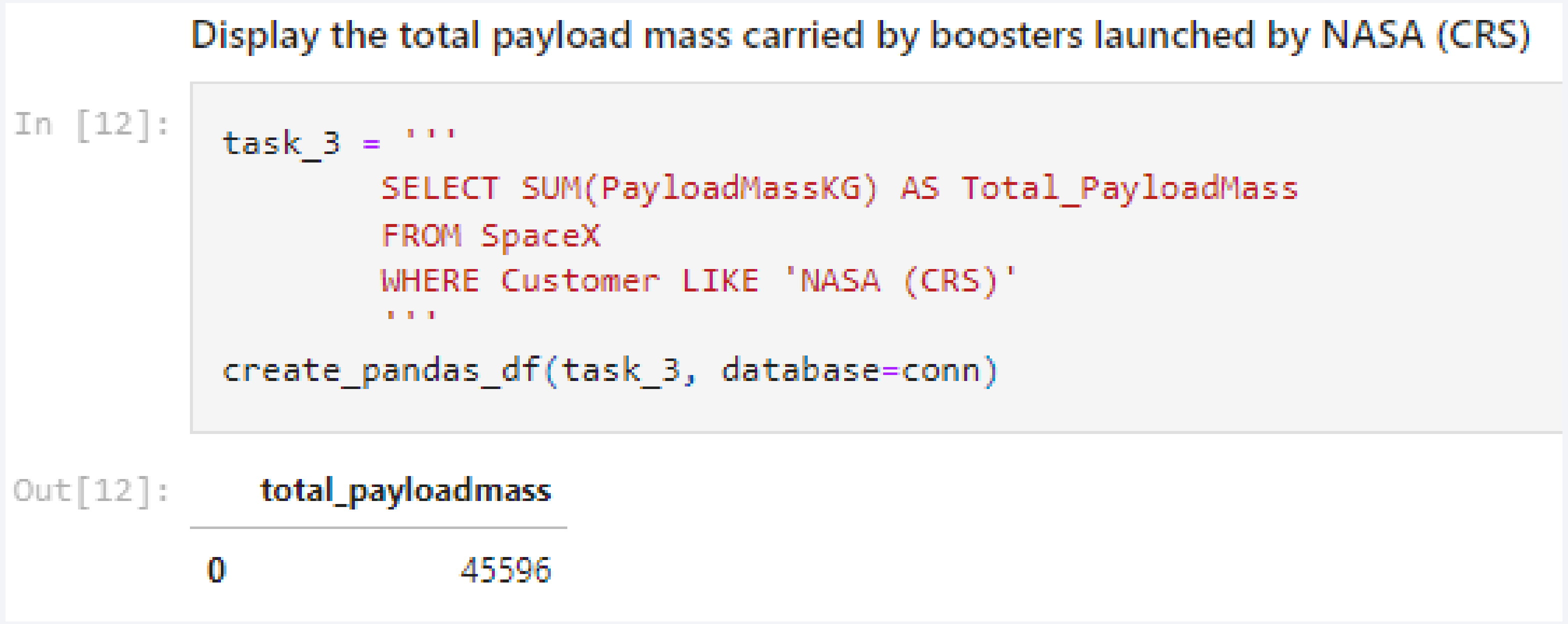
Out[11]:

	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 used the above query to retrieve 5 records where the launch site names start with "CCA."

Total Payload Mass

- We calculated the total payload carried by boosters from NASA as 45596 using the query below



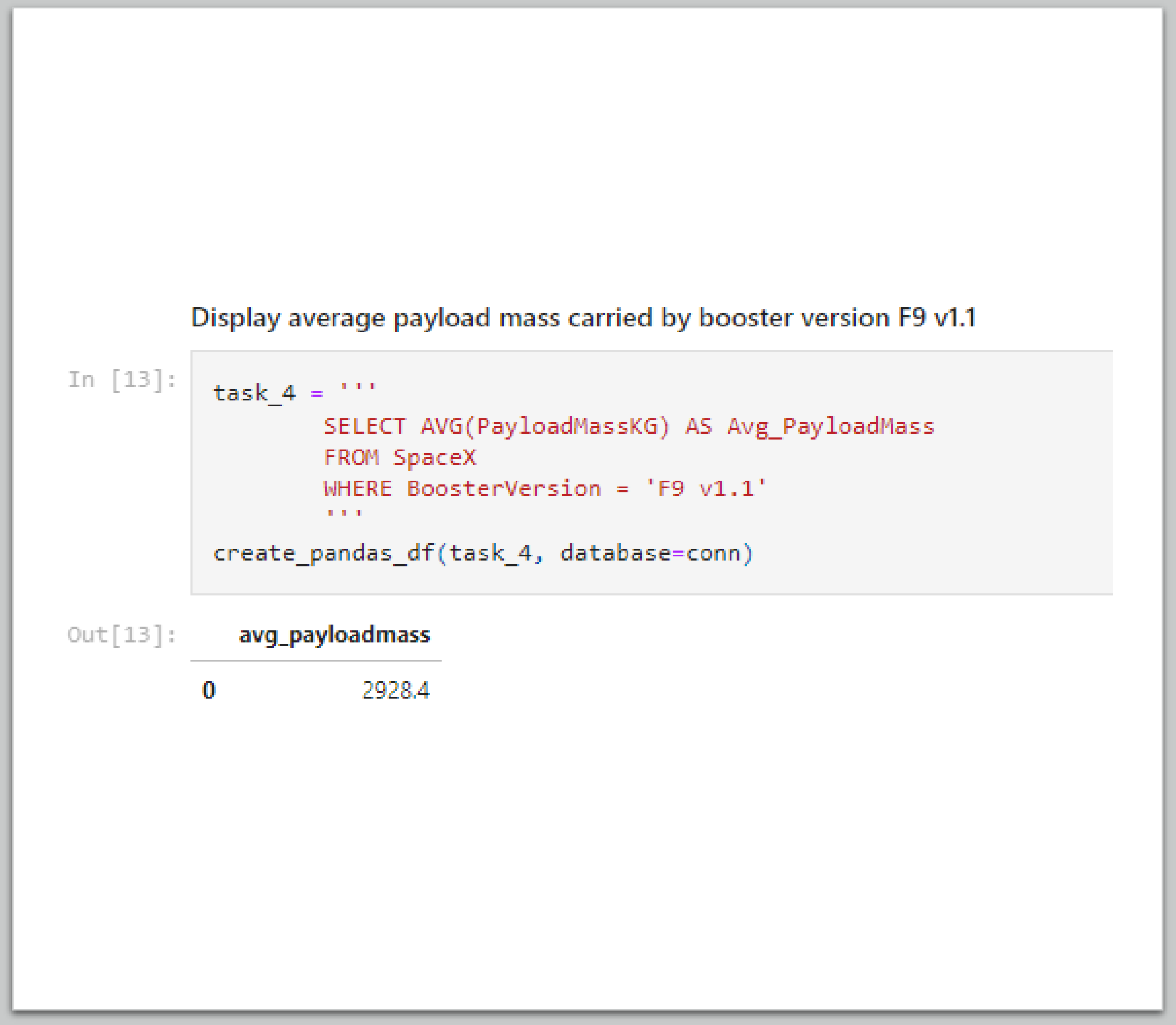
Display the total payload mass carried by boosters launched by NASA (CRS)

```
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 determined that the average payload mass carried by the F9 v1.1 booster version is 2,928.4.



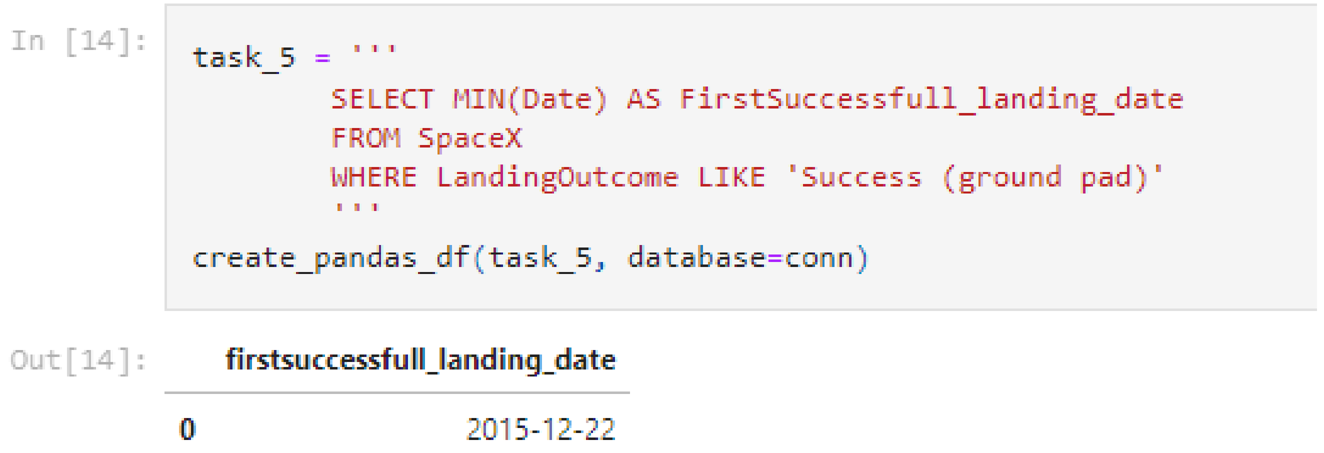
Display average payload mass carried by booster version F9 v1.1

```
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

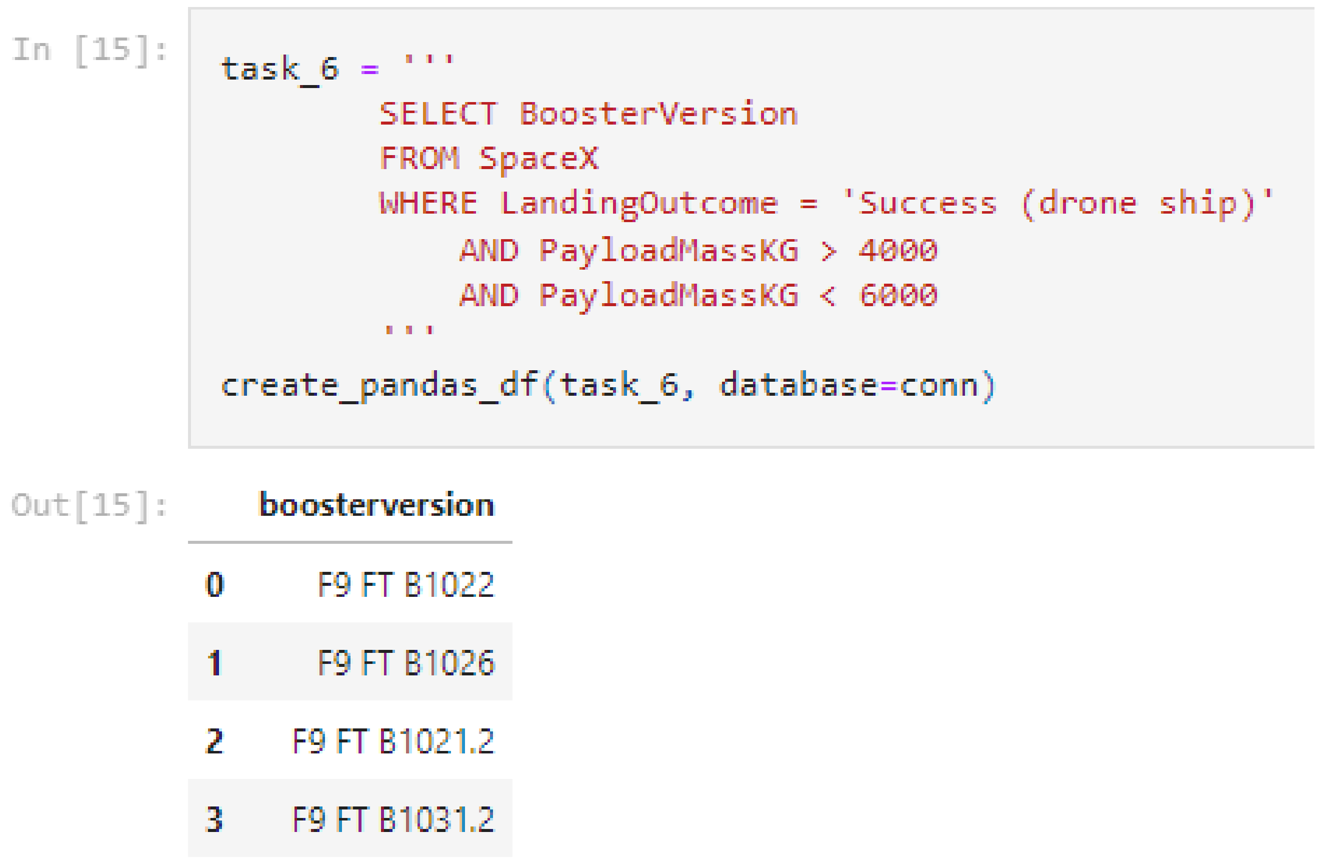
- We noted that the first successful landing on a ground pad occurred on December 22, 2015.



```
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 Landing with Payload between 4000 and 6000



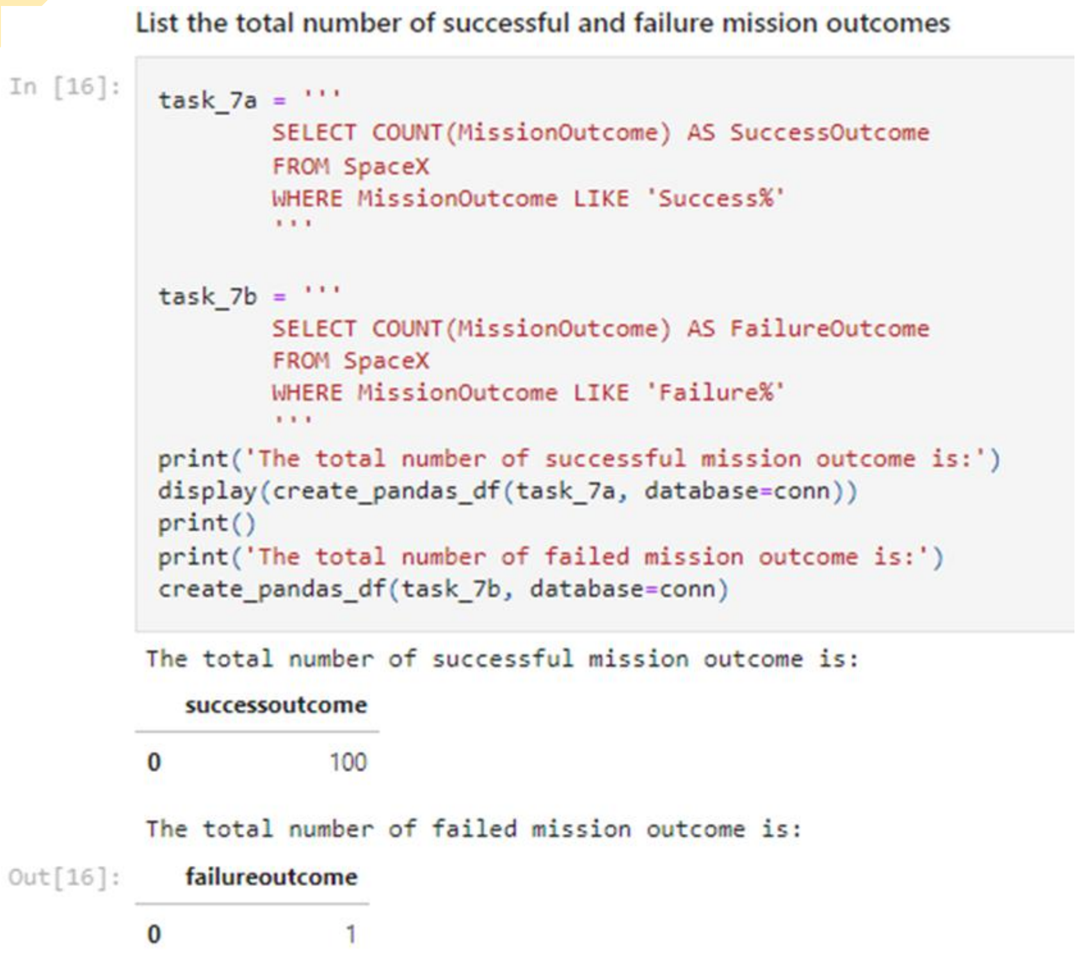
```
In [15]:
task_6 = '''
        SELECT BoosterVersion
        FROM SpaceX
        WHERE LandingOutcome = 'Success (drone ship)'
            AND PayloadMassKG > 4000
            AND PayloadMassKG < 6000
        '''
create_pandas_df(task_6, database=conn)
```

Out[15]:

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

- We applied the WHERE clause to filter boosters that successfully landed on the drone ship and used the AND condition to select those 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:
```

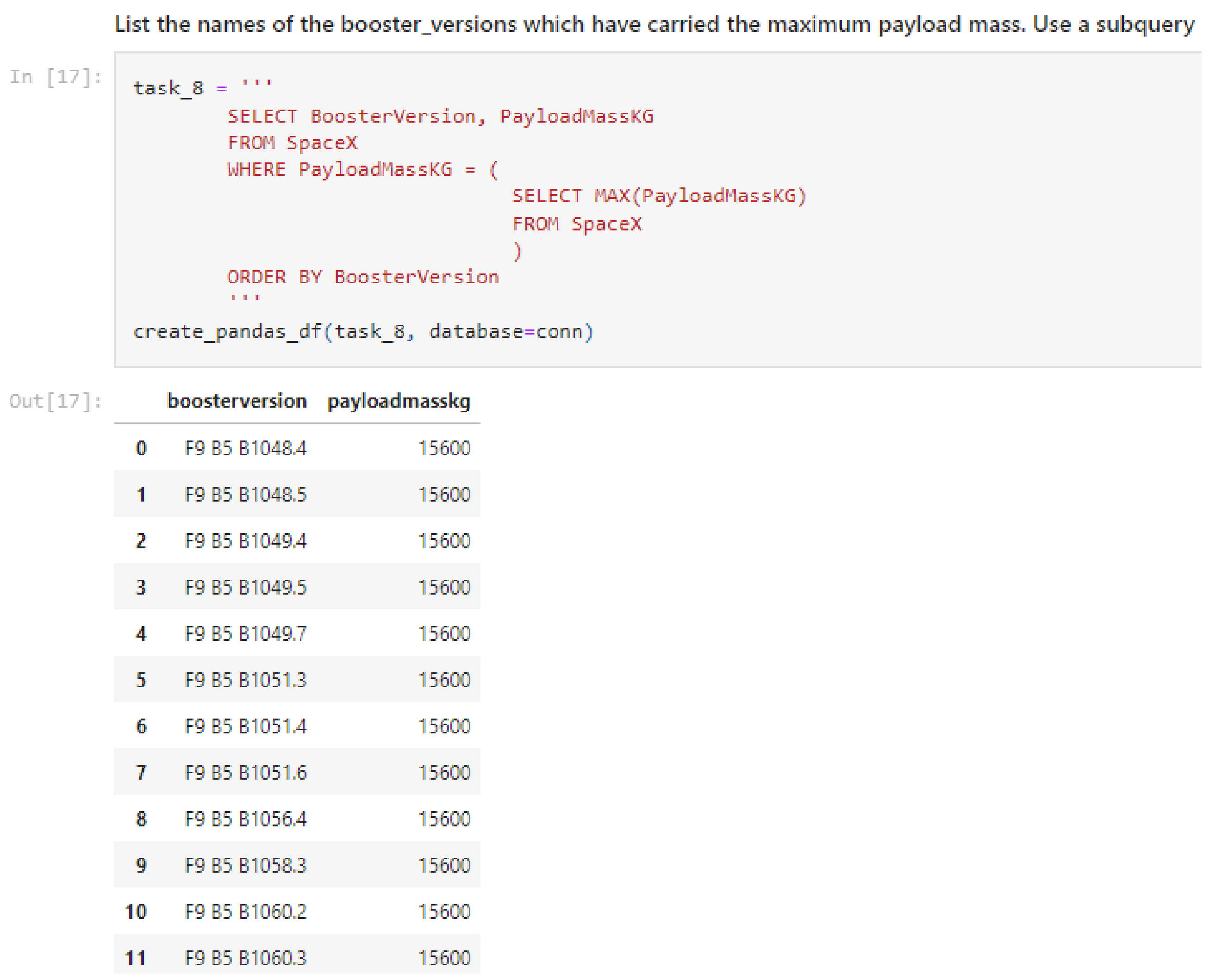
Out[16]:

	failureoutcome
0	1

- We used the wildcard character '%' in the WHERE clause to filter records where the MissionOutcome was either a success or a failure.

Boosters Carried Maximum Payload

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



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

In [17]:

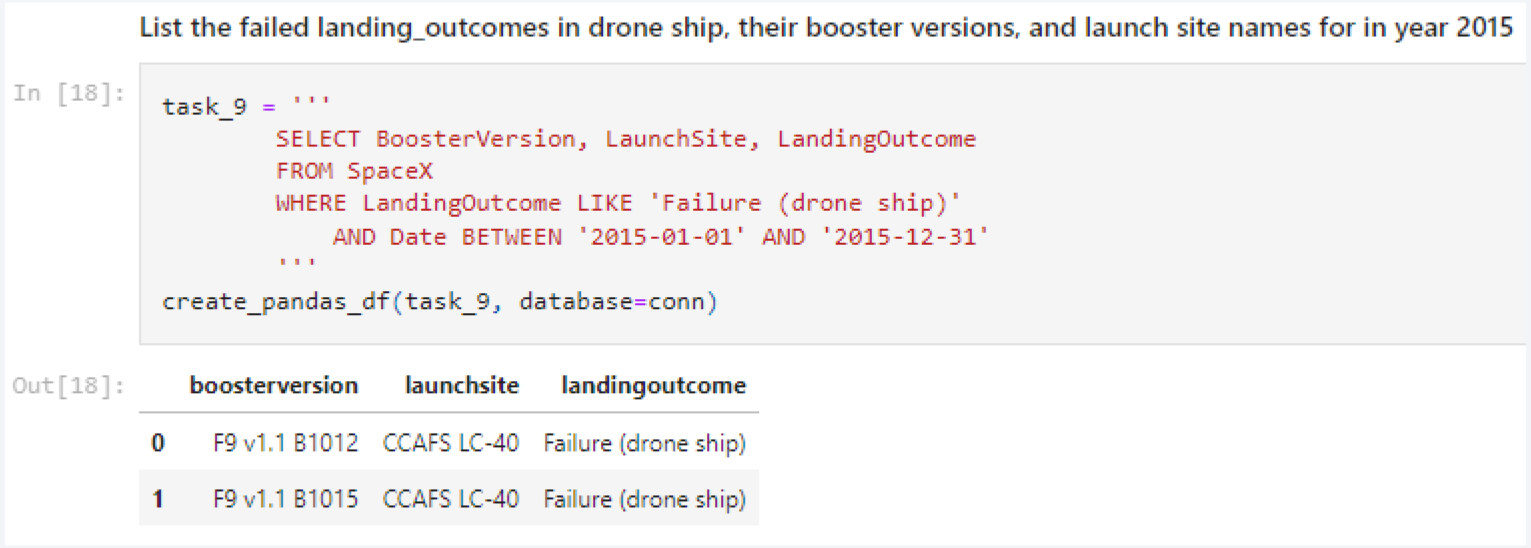
```
task_8 = '''
        SELECT BoosterVersion, PayloadMassKG
        FROM SpaceX
        WHERE PayloadMassKG = (
                                SELECT MAX(PayloadMassKG)
                                FROM SpaceX
                                )
        ORDER BY BoosterVersion
        '''
create_pandas_df(task_8, 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

2015 Launch Records

- We combined the WHERE clause with LIKE, AND, and BETWEEN conditions to filter for failed drone ship landings, including their booster versions and launch site names, specifically for the year 2015.



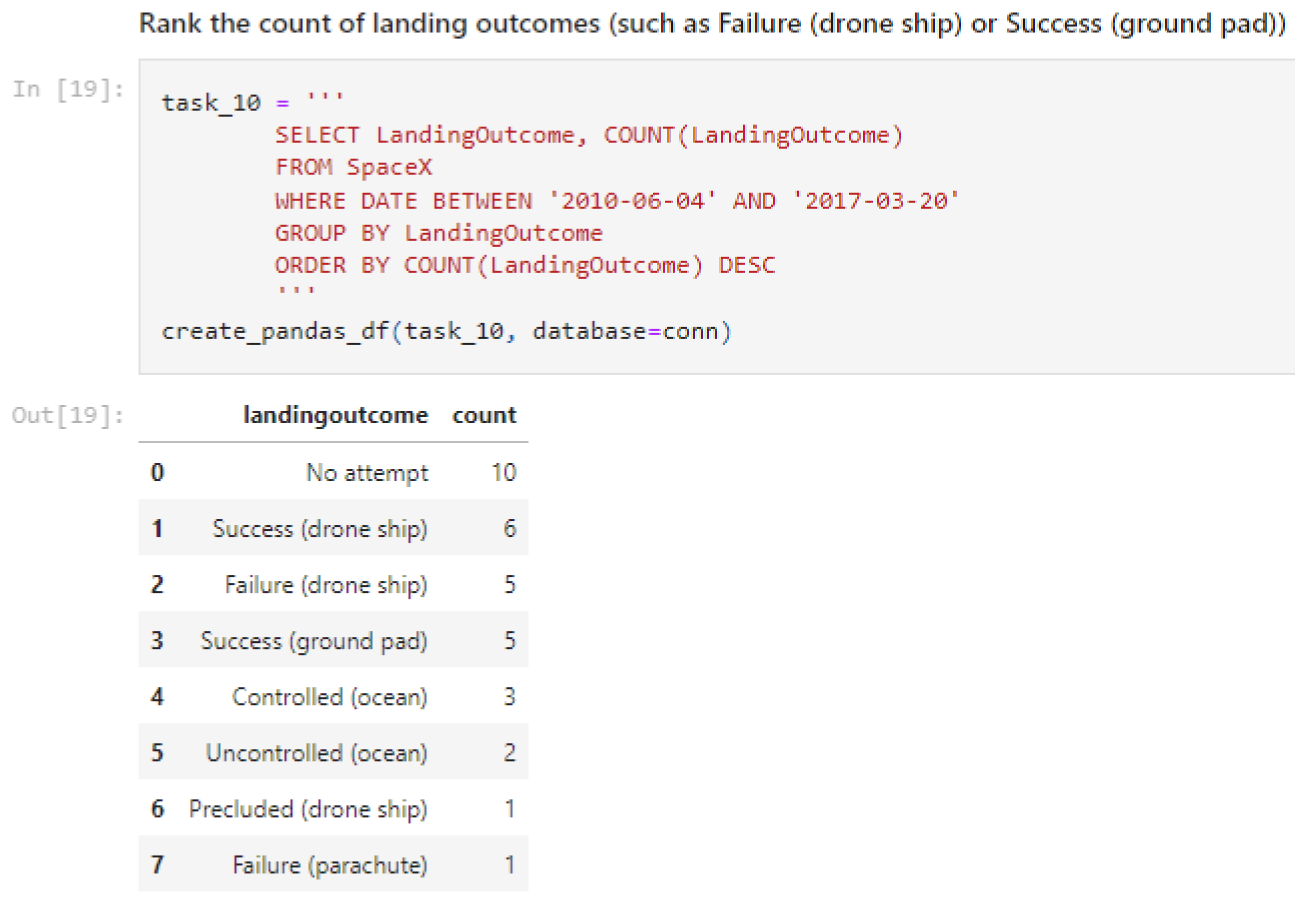
List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

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)
```

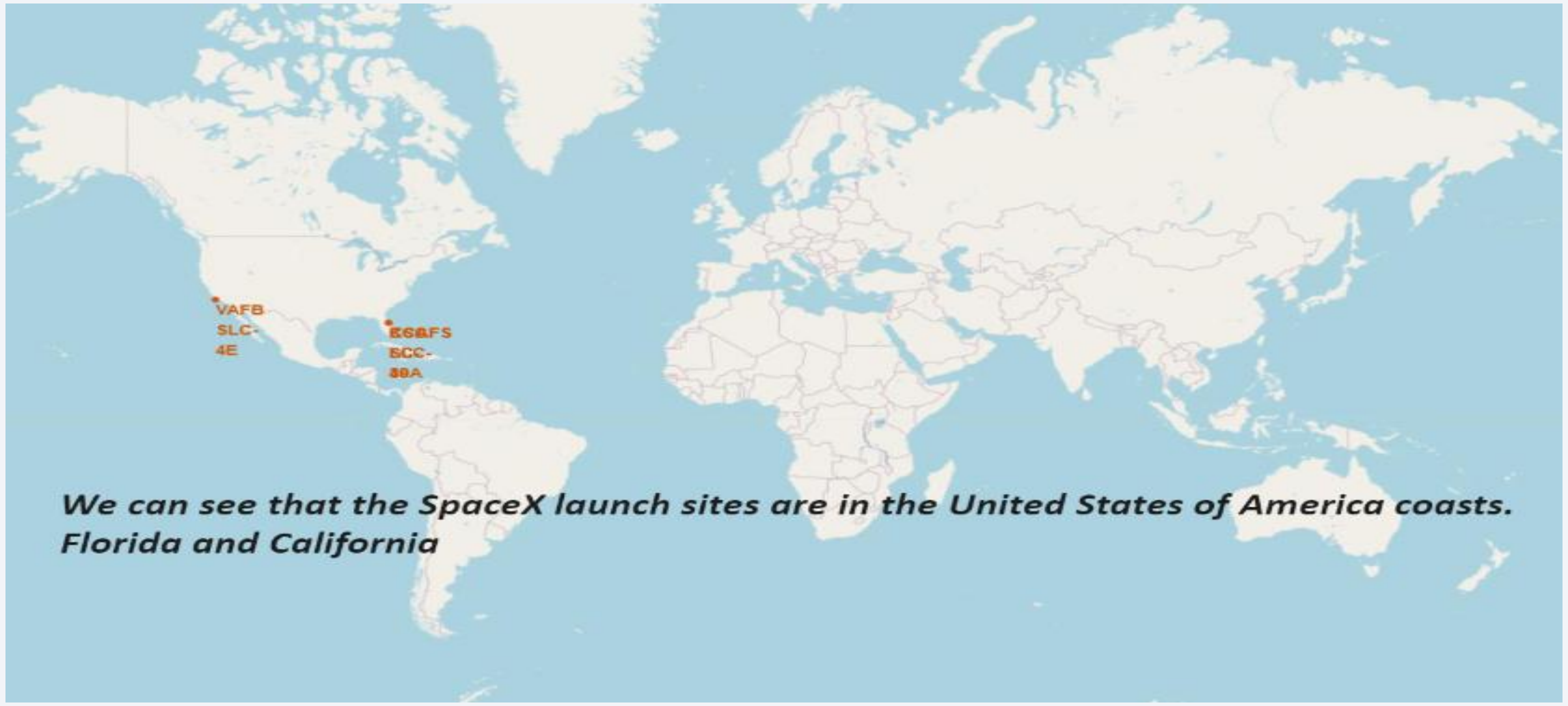
```
Out[19]:
        landingoutcome  count
0           No attempt     10
1   Success (drone ship)    6
2   Failure (drone ship)    5
3  Success (ground pad)     5
4    Controlled (ocean)     3
5  Uncontrolled (ocean)     2
6  Precluded (drone ship)   1
7    Failure (parachute)    1
```

- We selected the landing outcomes along with their counts and used the WHERE clause to filter records with landing dates between 2010-03-20 and 2010-06-04.Then, we applied the GROUP BY clause to group the landing outcomes and used ORDER BY to sort the groups in descending order.

Section 3

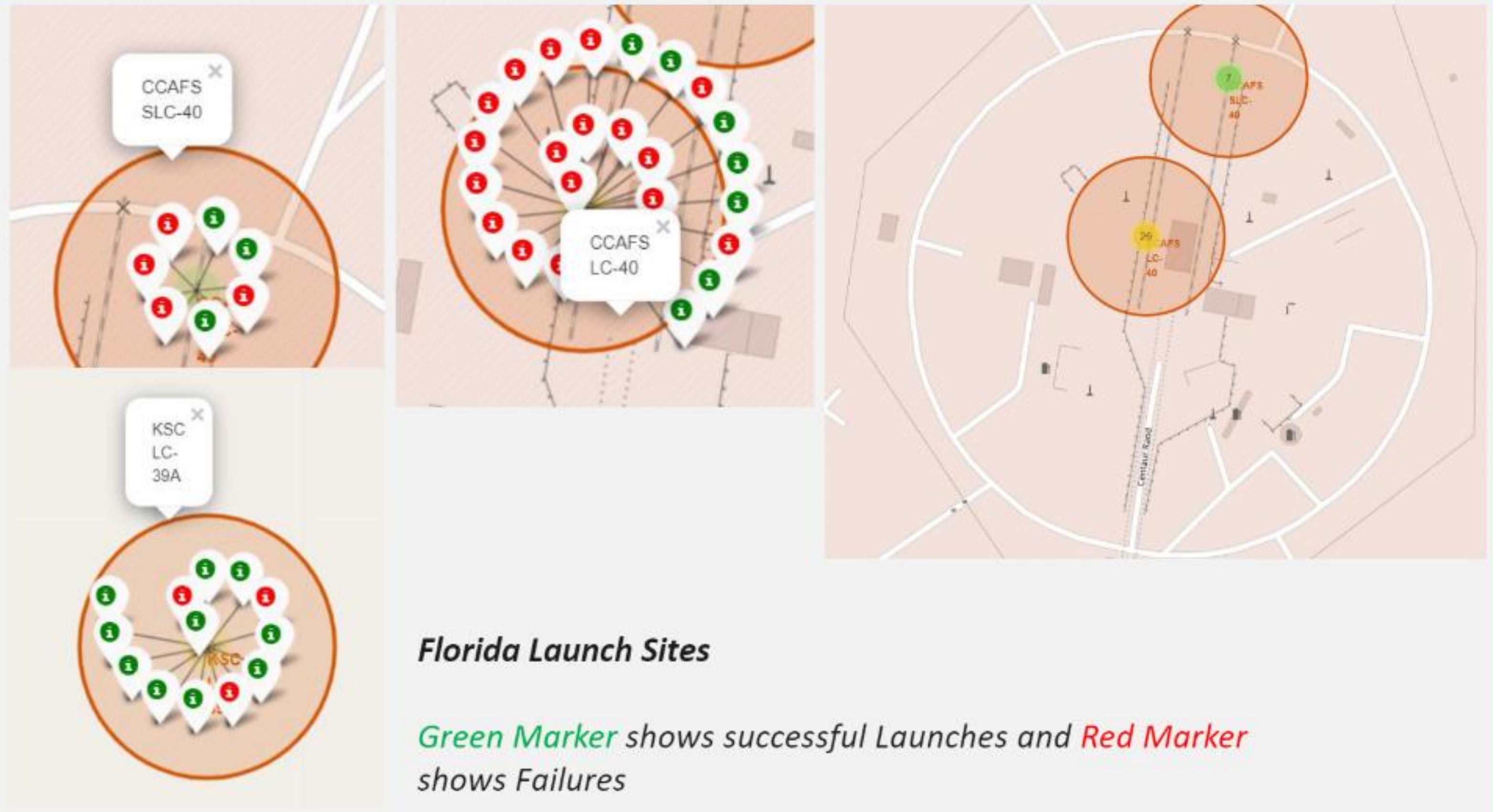
Launch Sites Proximities Analysis

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

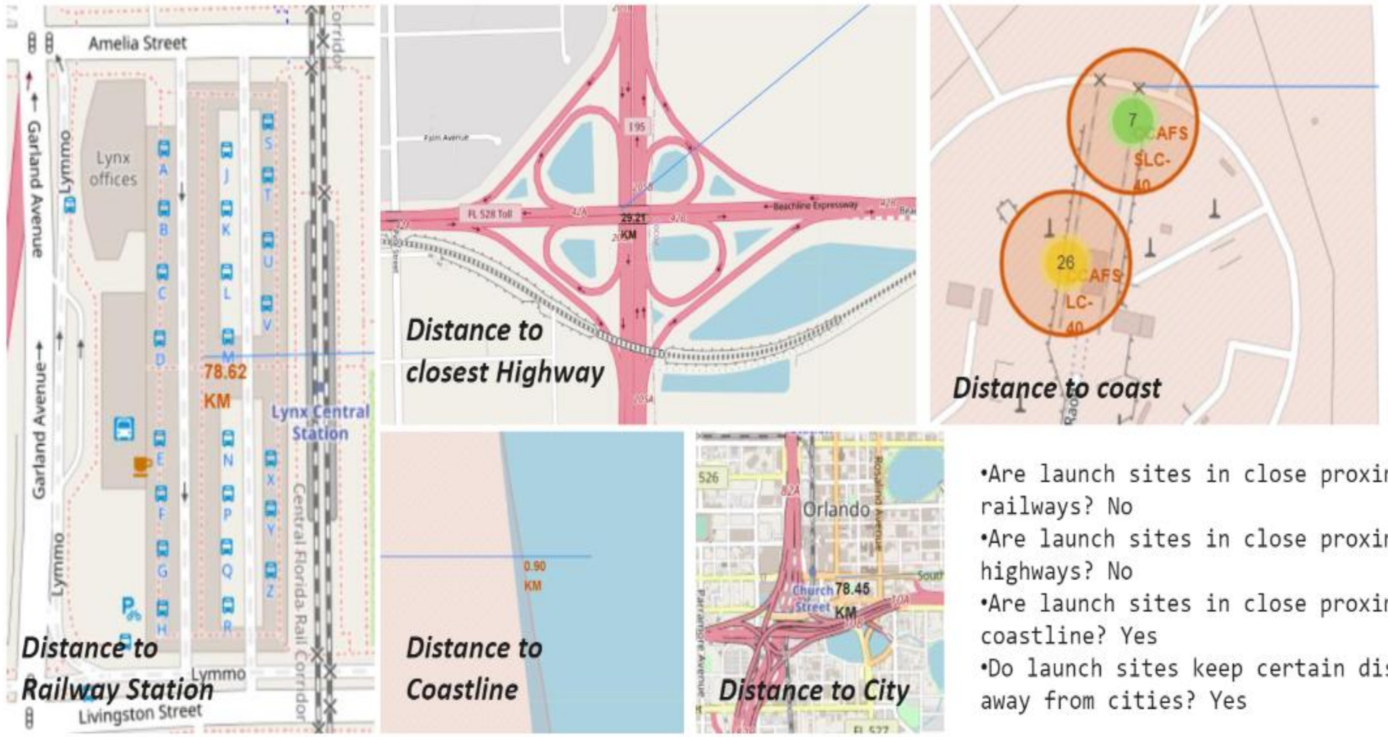
Green Marker shows successful Launches and Red Marker shows Failures

VAFB SLC-4E
Dir Road

California Launch Site

37

Launch Site distance to landmarks

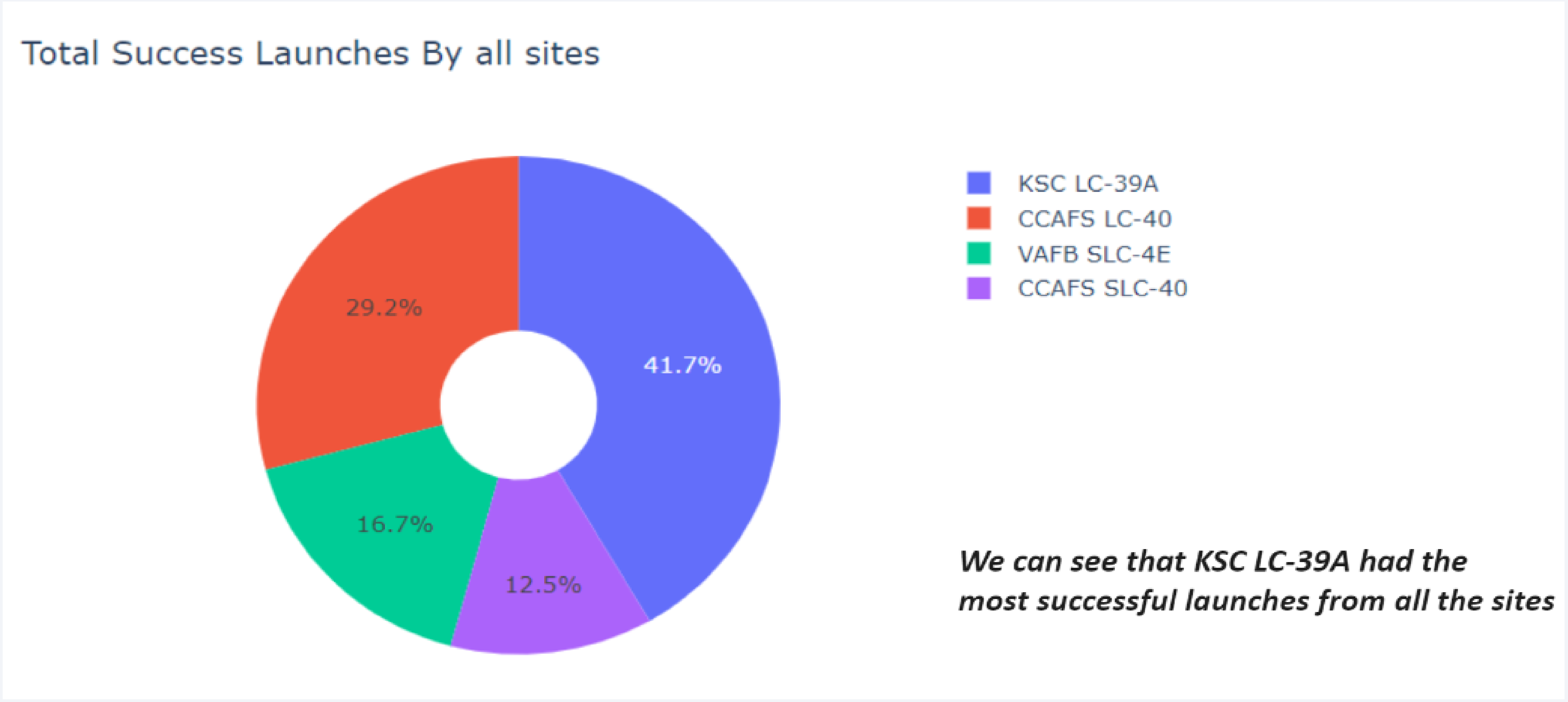


•Are launch sites in close proximity to railways? No
•Are launch sites in close proximity to highways? No
•Are launch sites in close proximity to coastline? Yes
•Do launch sites keep certain distance away from cities? Yes

Section 4

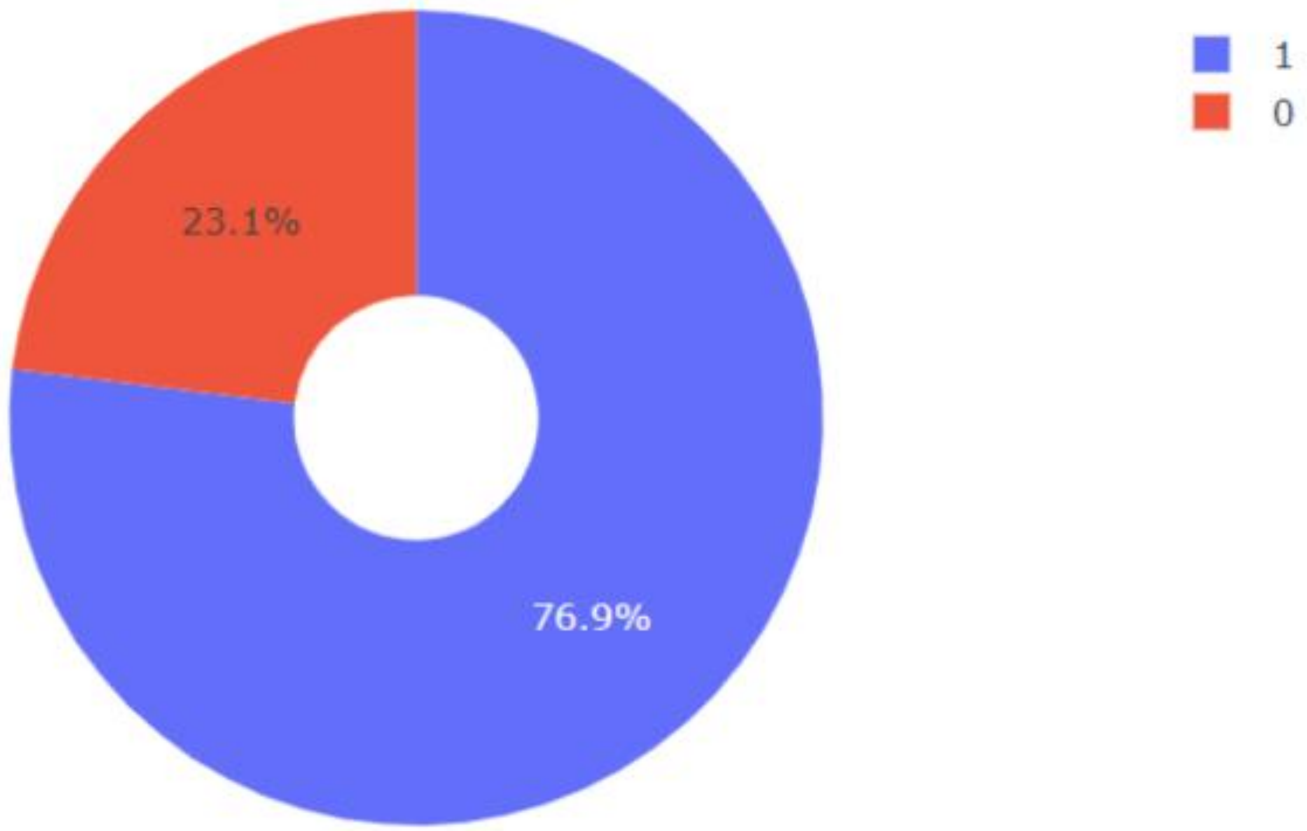
Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site



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

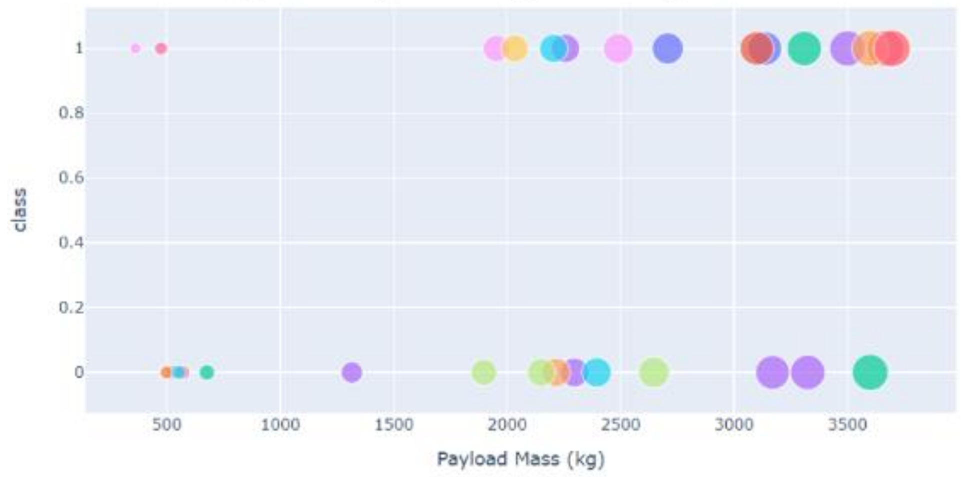
Pie chart showing the Launch site with the highest launch success ratio



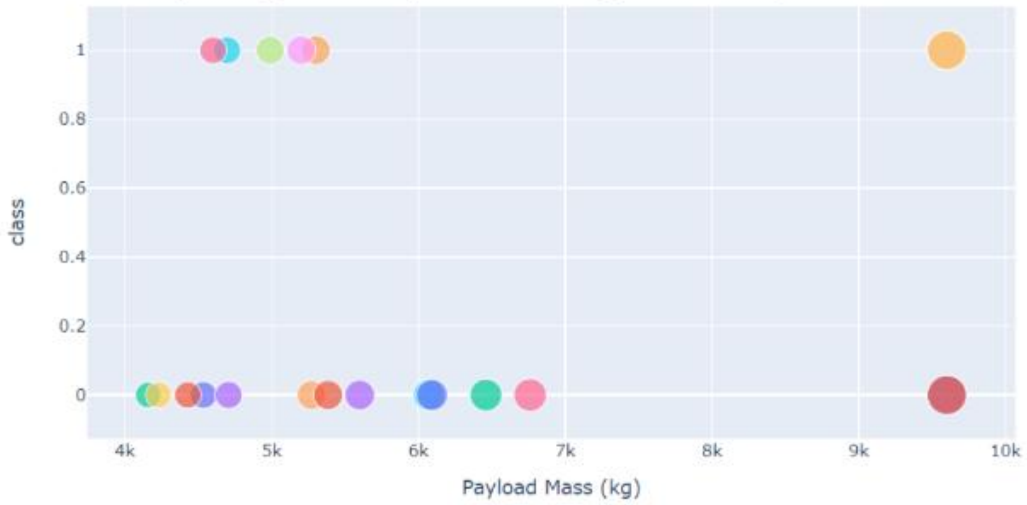
KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

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



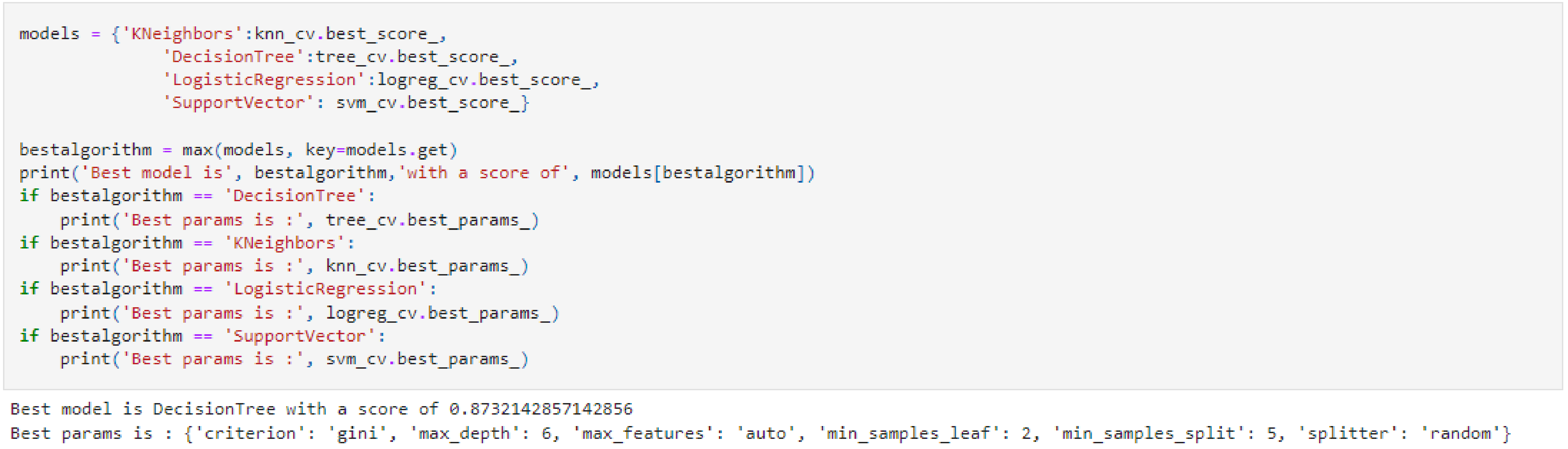
We can see the success rates for low weighted payloads is higher than the heavy weighted payloads

Section 5

Predictive Analysis (Classification)

Classification Accuracy

- The decision tree classifier is the model with the highest classification accuracy



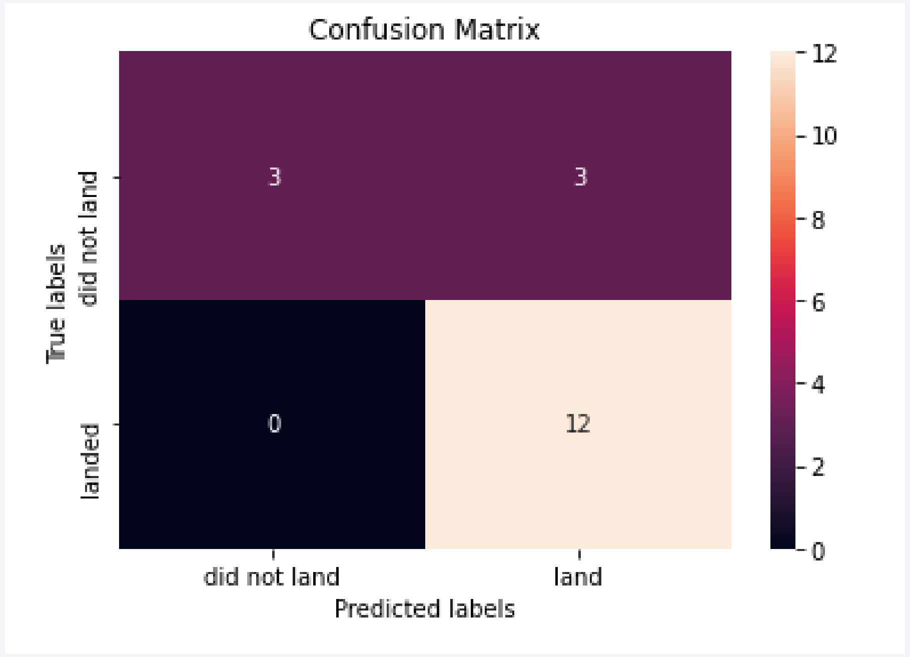
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

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 indicates that it can differentiate between the classes; however, the main issue lies in false positives—cases where unsuccessful landings are incorrectly predicted as successful.



Conclusions

We can conclude the following: Launch sites with a higher number of flights tend to have higher success rates. The launch success rate steadily increased from 2013 to 2020.The orbits ES-L1, GEO, HEO, SSO, and VLEO showed the highest success rates. KSC LC-39A recorded the most successful launches among all sites. The decision tree classifier proved to be the most effective machine learning model for this task.

Thank you!