IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Soufiane Heddadi
7th July 2024

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

1. Project Context:

   - SpaceX's Cost Advantage: SpaceX advertises Falcon 9 rocket launches on its website for $62 million, significantly less than other providers' costs (upward of $165 million). The key savings come from reusing the first stage.
   - Cost Determination: By predicting whether the first stage will land successfully, we can estimate the overall launch cost. This information is valuable for bidding against SpaceX in rocket launch contracts.

2. Challenges to Address:

   - Factors for Successful Landing: We'll explore the factors influencing successful rocket landings.
   - Operating Conditions: Identifying the necessary conditions for 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

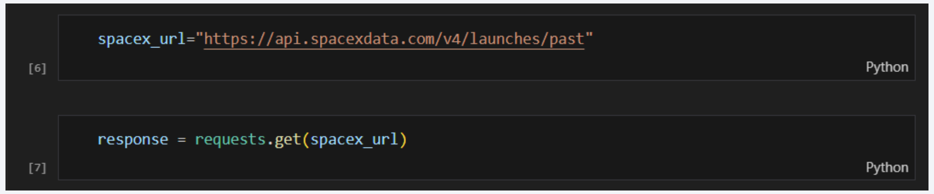
Data Collection

1. Data Collection Methods:
   - We collected data using various methods.
   - Specifically, we made a GET request to the SpaceX API.
   - After receiving the response, we decoded it as JSON using the .json() function.
   - The resulting data was transformed into a pandas dataframe using .json_normalize().
2. Data Cleaning and Augmentation:
   - We cleaned the data, addressing missing values where necessary.
   - Additionally, we performed web scraping from Wikipedia.
   - Our goal was to extract Falcon 9 launch records from an HTML table and convert them into a pandas dataframe for future analysis.

Data Collection – SpaceX API

Code Link

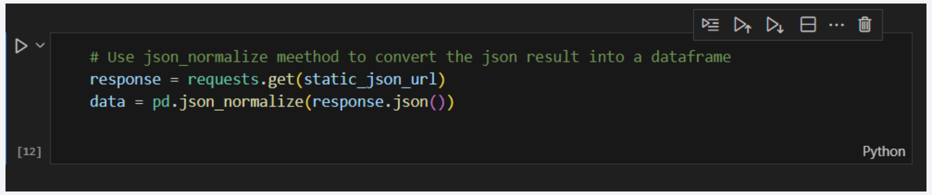
1-Requesting Rocket Launch Data From Spacex API



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

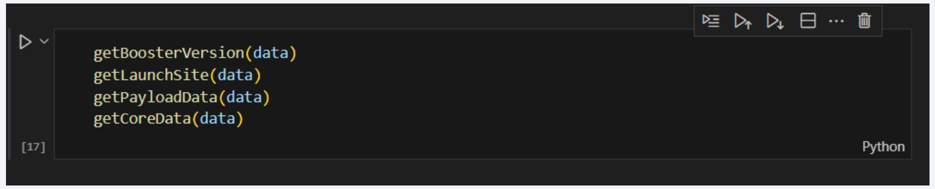
```
[7]
response = requests.get(spacex_url)
```

2-Convert The Json Result Into A Dataframe



```
[12]
# Use json_normalize meethod to convert the json result into a dataframe
response = requests.get(static_json_url)
data = pd.json_normalize(response.json())
```

3-Transform The Data

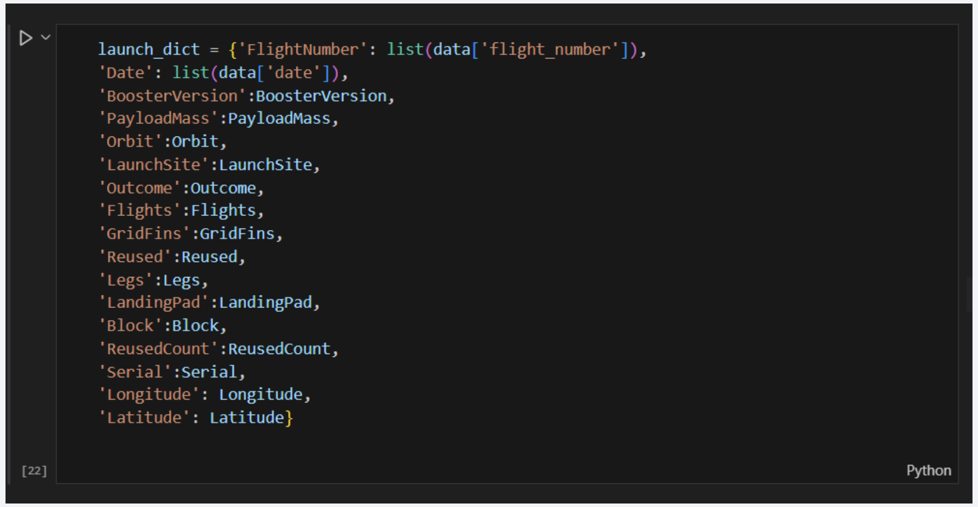


```
[17]
getBoosterVersion(data)
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
```

Data Collection – SpaceX API

Code Link

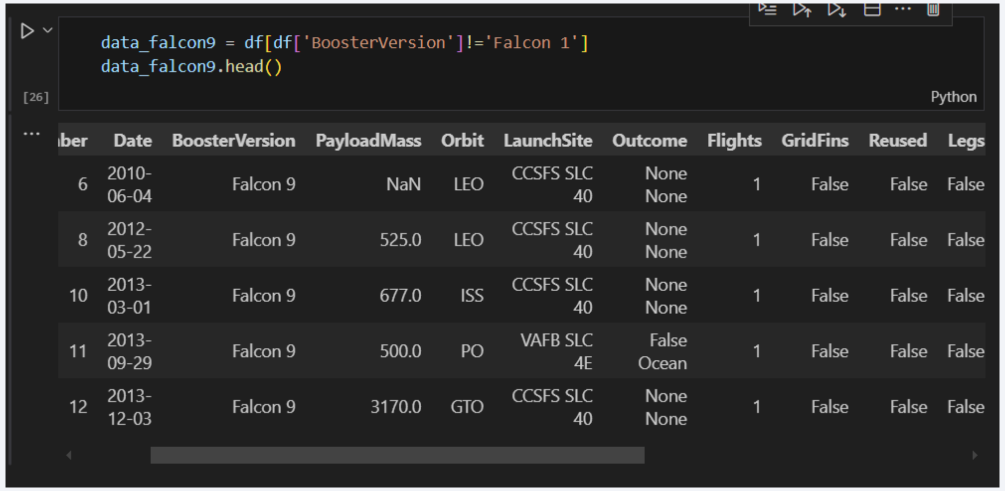
4-Construct Our Dataset Using The Data We Have Obtained. We Combine The Columns Into A Dictionary.



```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
```

[22] Python

5-Filter The Dataframe To Only Include \`Falcon 9\` Launches

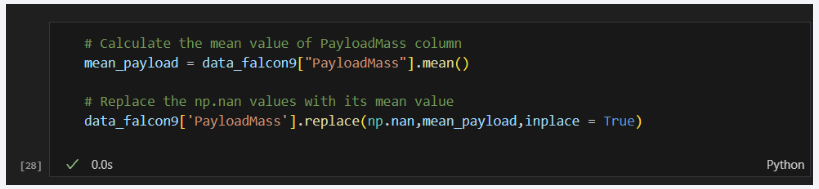


```
data_falcon9 = df[df['BoosterVersion']!='Falcon 1']
data_falcon9.head()
```

[26] Python

ber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs
6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False
8	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False
10	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	False	False	False
11	2013-09-29	Falcon 9	500.0	PO	VAFB SLC 4E	False Ocean	1	False	False	False
12	2013-12-03	Falcon 9	3170.0	GTO	CCSFS SLC 40	None None	1	False	False	False

6-Dealing with Missing Values



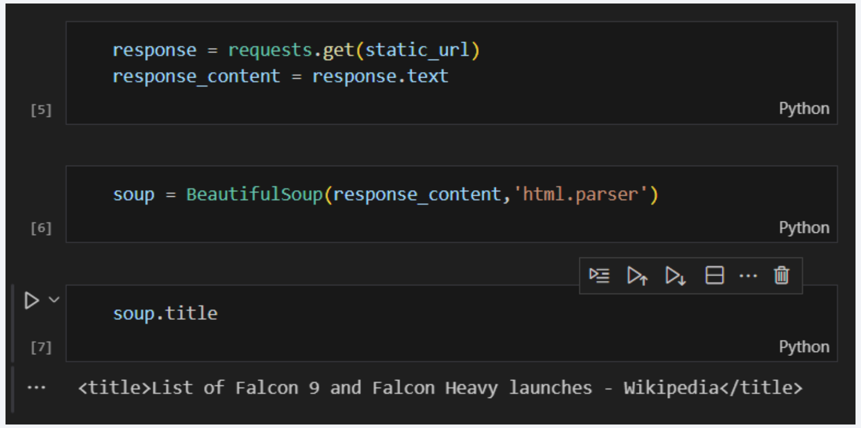
```
# Calculate the mean value of PayloadMass column
mean_payload = data_falcon9["PayloadMass"].mean()

# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'].replace(np.nan,mean_payload,inplace = True)
```

[28] ✓ 0.0s Python

Data Collection - Scraping

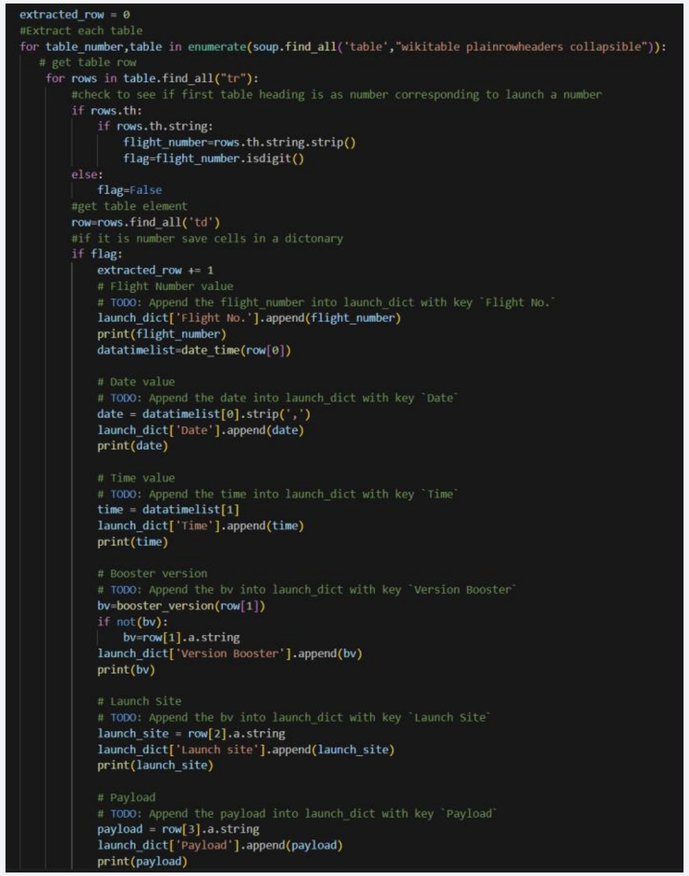
Code Link

1-Request the Falcon9 Launch Wiki page from its URL



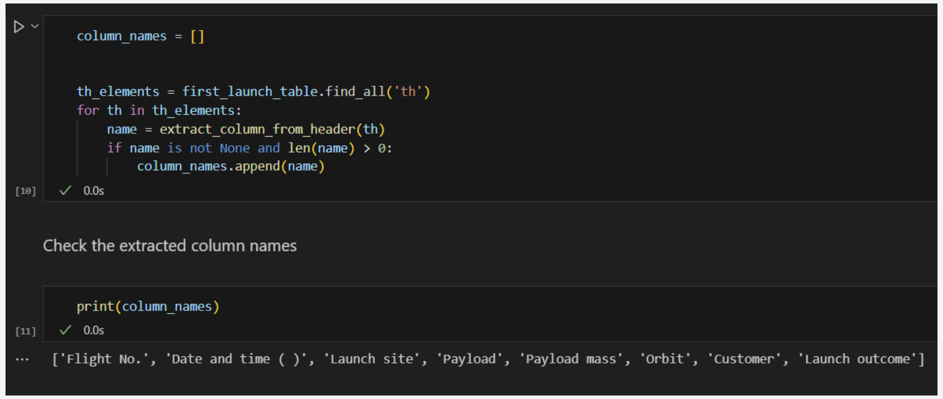
```
[5]  response = requests.get(static_url)
     response_content = response.text
                                                        Python

[6]  soup = BeautifulSoup(response_content,'html.parser')
                                                        Python

[7]  soup.title
                                                        Python
...  <title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

3-Create a data frame by parsing the launch HTML tables



```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            launch_dict['Flight No.'].append(flight_number)
            print(flight_number)
            datatimelist=date_time(row[0])

            # Date value
            # TODO: Append the date into launch_dict with key `Date`
            date = datatimelist[0].strip(',')
            launch_dict['Date'].append(date)
            print(date)

            # Time value
            # TODO: Append the time into launch_dict with key `Time`
            time = datatimelist[1]
            launch_dict['Time'].append(time)
            print(time)

            # Booster version
            # TODO: Append the bv into launch_dict with key `Version Booster`
            bv=booster_version(row[1])
            if not(bv):
                bv=row[1].a.string
            launch_dict['Version Booster'].append(bv)
            print(bv)

            # Launch Site
            # TODO: Append the bv into launch_dict with key `Launch Site`
            launch_site = row[2].a.string
            launch_dict['Launch site'].append(launch_site)
            print(launch_site)

            # Payload
            # TODO: Append the payload into launch_dict with key `Payload`
            payload = row[3].a.string
            launch_dict['Payload'].append(payload)
            print(payload)
```

2-Extract all column names from the HTML table header



```
      column_names = []


      th_elements = first_launch_table.find_all('th')
      for th in th_elements:
          name = extract_column_from_header(th)
          if name is not None and len(name) > 0:
              column_names.append(name)
[10]  ✓ 0.0s
```

Check the extracted column names

```
      print(column_names)
[11]  ✓ 0.0s
...   ['Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome']
```

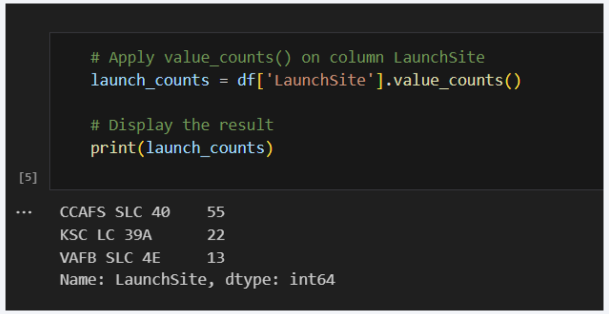
Data Wrangling

Code Link

In the dataset, there are several cases where the booster did not land successfully.

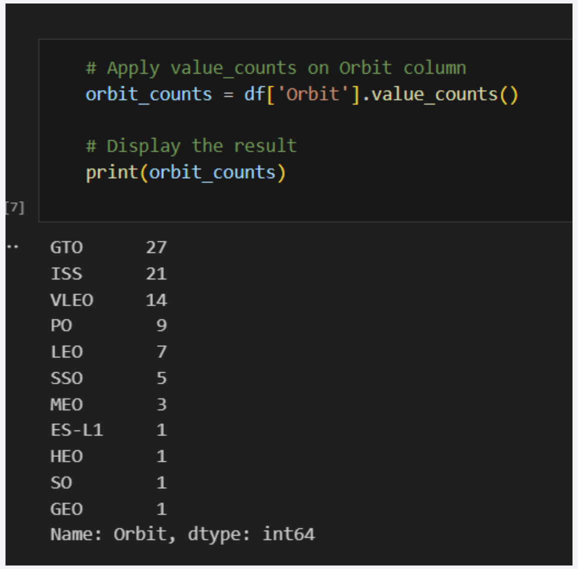
- True Ocean, True RTLS, True ASDS means the mission has been successful.
- False Ocean, False RTLS, False ASDS means the mission was a failure.

1- Calculate launches number for each site



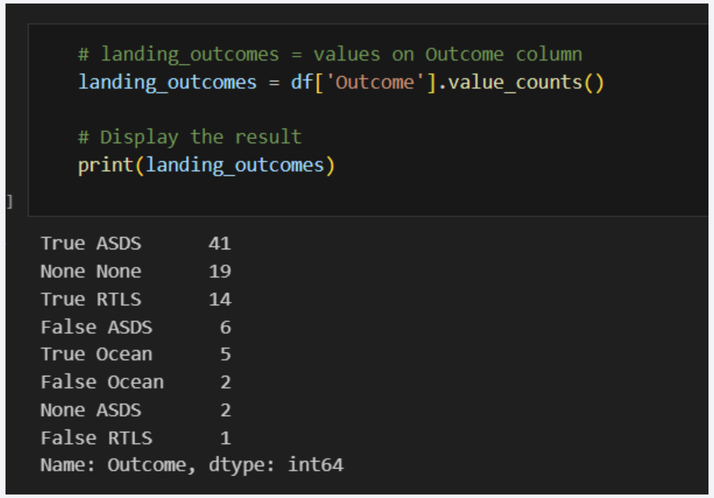
```
# Apply value_counts() on column LaunchSite
launch_counts = df['LaunchSite'].value_counts()

# Display the result
print(launch_counts)
[5]
...  CCAFS SLC 40    55
     KSC LC 39A      22
     VAFB SLC 4E     13
     Name: LaunchSite, dtype: int64
```

2- Calculate the number and occurrence of each orbit



```
# Apply value_counts on Orbit column
orbit_counts = df['Orbit'].value_counts()

# Display the result
print(orbit_counts)
[7]
...  GTO      27
     ISS      21
     VLEO     14
     PO        9
     LEO       7
     SSO       5
     MEO       3
     ES-L1     1
     HEO       1
     SO        1
     GEO       1
     Name: Orbit, dtype: int64
```

3- Calculate number and occurrence of mission outcome per orbit type



```
# landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()

# Display the result
print(landing_outcomes)
]
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
False Ocean     2
None ASDS       2
False RTLS      1
Name: Outcome, dtype: int64
```

EDA with Data Visualization

Code Link

We explored the data by Creating:

- ***Catplot of PayloadMass vs. FlightNumber :***
  - Purpose: Useful for comparing relationships between categorical and continuous variables.
- ***Scatter plots "FlightNumber vs. LaunchSite" , "PayloadMass vs. LaunchSite", "FlightNumber vs. Orbit", "PayloadMass vs. Orbit"***
  - Purpose: Effective for showing correlations between two continuous variables with an additional categorical variable to highlight success.
- ***Bar chart of success rate by Orbit type:***
  - Purpose: Bar charts effectively highlight differences in categorical data.

EDA with SQL

Code Link

- We performed SQL queries to gather and understand data from dataset:
  - Installed required libraries: sqlalchemy, ipython-sql, pandas.
  - Loaded SQL extension.
  - Set up SQLite connection.
  - Imported SpaceX CSV data into a DataFrame.
  - Saved the DataFrame to an SQLite database.
  - Created a new SQL table SPACEXTABLE with non-null dates.
  - Queried for unique launch sites.
  - Dropped the existing table to avoid conflicts.

Build an Interactive Map with Folium

[Code Link]

Summary of Map Objects Added to the Folium Map:

- Circle and marker at NASA Johnson Space Center:
  - Purpose: Mark the location of NASA Johnson Space Center.
  - Reason: To visually highlight and label this significant location on the map.
- Circles and markers for each launch site:
  - Purpose: Indicate the geographic locations of SpaceX launch sites.
  - Reason: To provide a visual reference and easy identification of each launch site.

Explanation of Added Objects:

- Circle and marker at NASA Johnson Space Center:
  - Added to highlight the starting reference point and its significance as a key NASA facility.
- Circles and markers for each launch site:
  - Added to represent and label the locations of SpaceX launch sites, making it easier to see their geographic distribution.

Build a Dashboard with Plotly Dash

Code Link

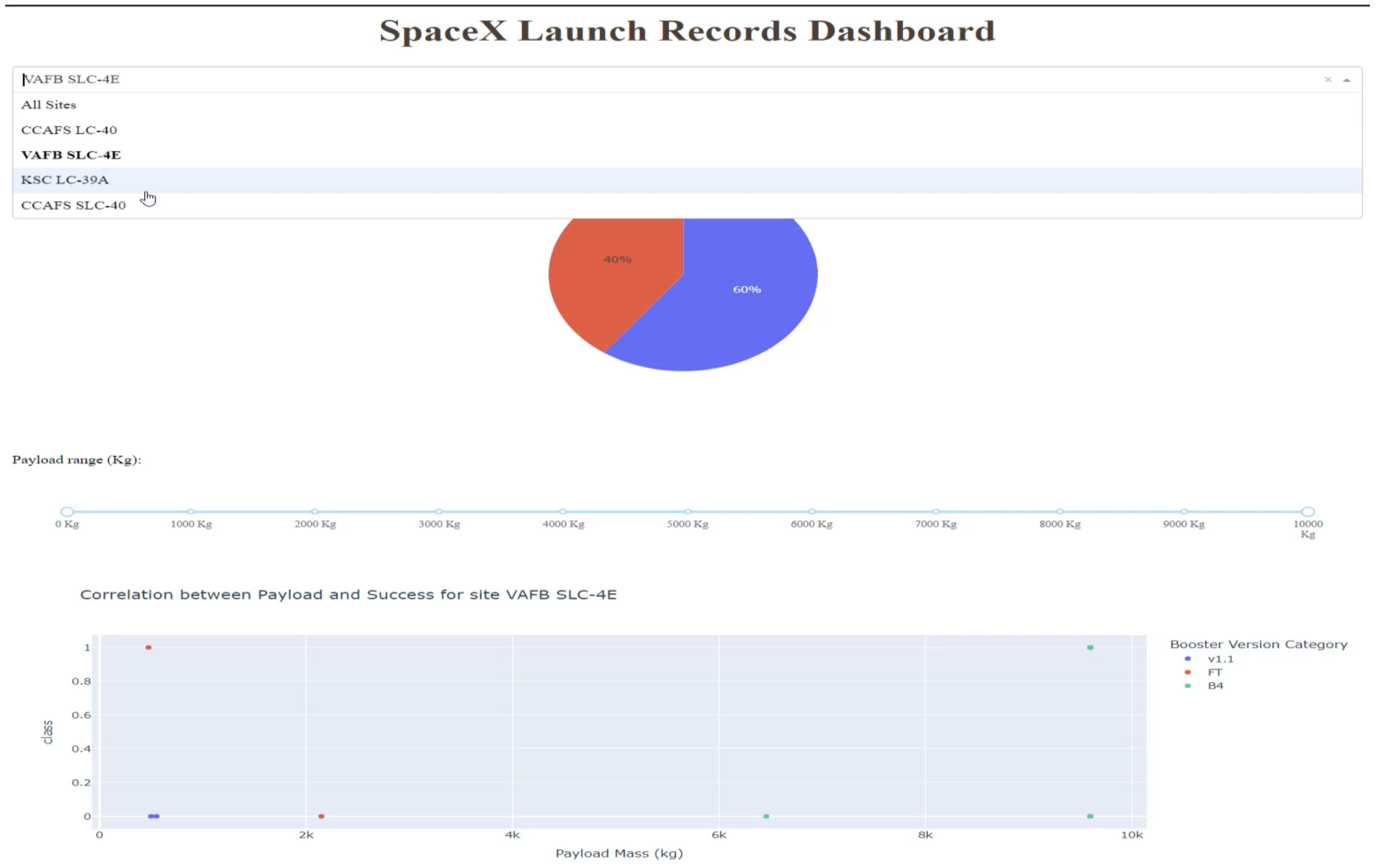
- Dashboard has dropdown, pie chart, rangeslider and scatter plot components :
  - Added a dropdown menu to select different launch sites, with options for 'All Sites' and each specific site.
  - Created a callback function to display a pie chart based on the selected launch site, showing success counts for all sites or specific site success and failure counts.
  - Added a range slider to select payload range, with values from 0 kg to 10,000 kg in 1,000 kg intervals.
  - Created a callback function to display a scatter plot of payload vs. launch outcome, color-coded by booster version, based on selected site and payload range.

Predictive Analysis (Classification)

Code Link

- We began by loading our dataset with numpy and pandas, transforming the data, and then splitting it into training and testing sets.
- We selected various machine learning algorithms and used GridSearchCV to tune their hyperparameters.
- We trained our models with the training data.
- To evaluate the models, we identified the best hyperparameters, computed accuracy, and plotted confusion matrices.
- We compared the models based on their accuracy.
- By using feature engineering and algorithm tuning, we improved the models.
- In the end, we selected the best-performing classification model.

Results


SpaceX Launch Records Dashboard

VAFB SLC-4E

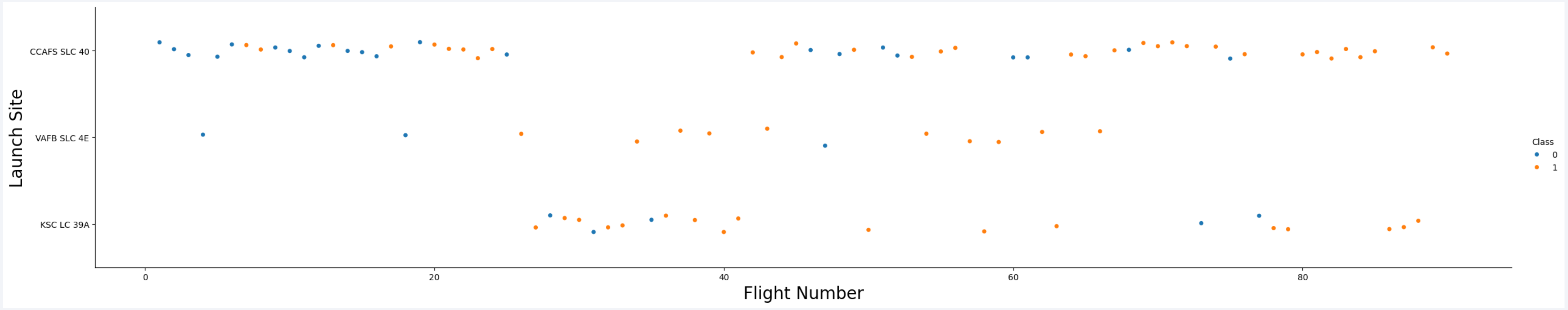
All Sites
CCAFS LC-40
VAFB SLC-4E
KSC LC-39A
CCAFS SLC-40

Section 2

Insights drawn from EDA

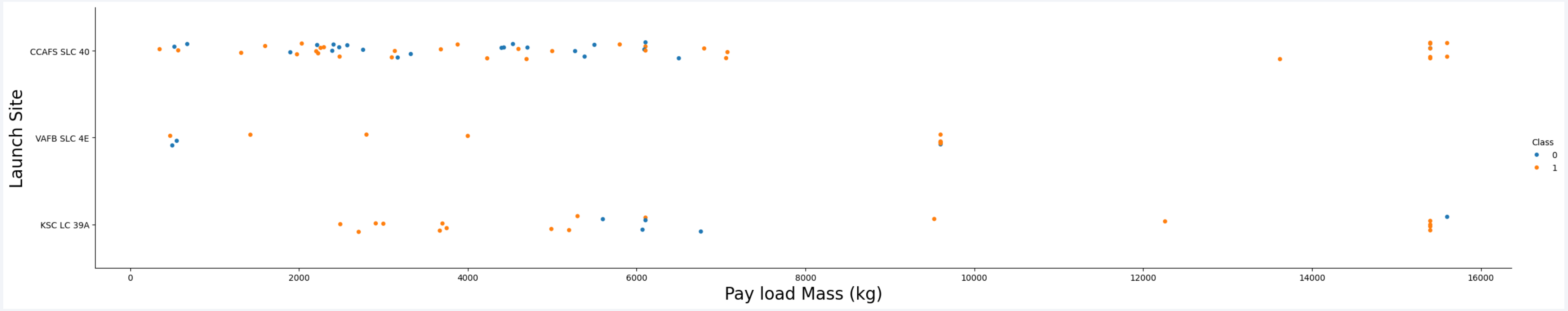
Flight Number vs. Launch Site

- The plot shows that SpaceX's landing success rate improves over time, with more recent flights (higher flight numbers) having a higher proportion of successful landings (Class 1) across all launch sites.
- Additionally, CCAFS SLC 40 and KSC LC 39A have more launches and a higher number of successful landings compared to VAFB SLC 4E.



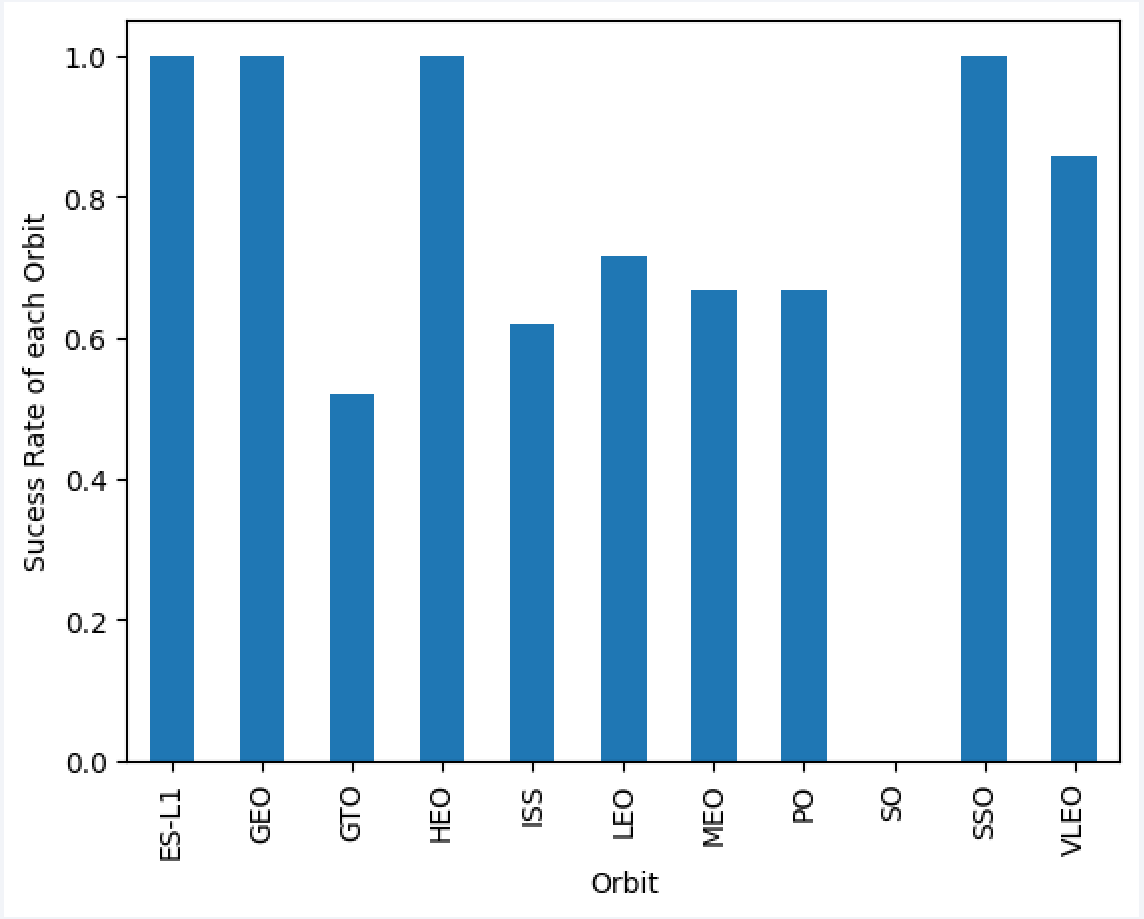
Payload vs. Launch Site

- The plot indicates that for the VAFB SLC 4E launch site, there have been no launches with heavy payloads (greater than 10,000 kg). This suggests that VAFB SLC 4E is primarily used for lighter payload missions, while CCAFS SLC 40 and KSC LC 39A handle a wider range of payload masses, including heavier ones.



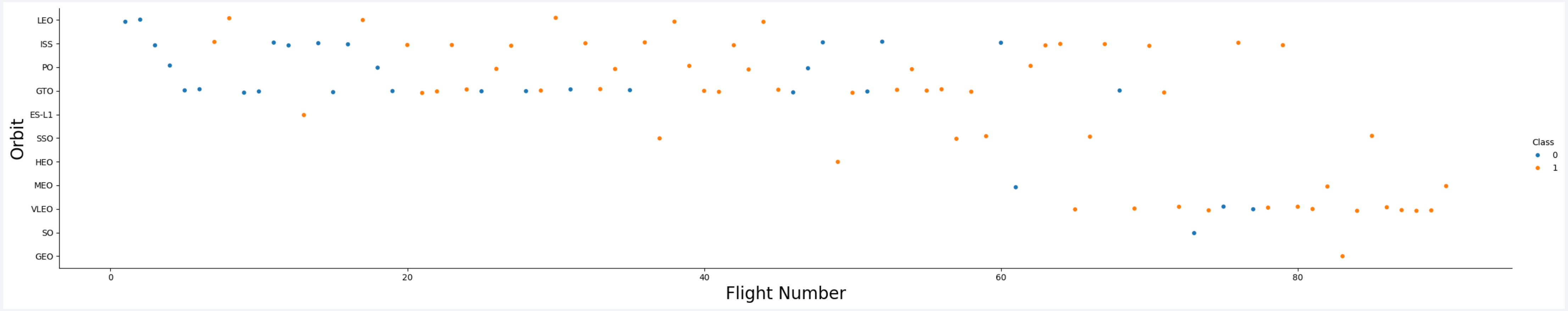
Success Rate vs. Orbit Type

The chart shows that SpaceX has the highest landing success rates for ES-L1, GEO, HEO, SSO, and VLEO orbits, while GTO missions have the lowest success rate. LEO, MEO, and PO orbits have moderate success rates.



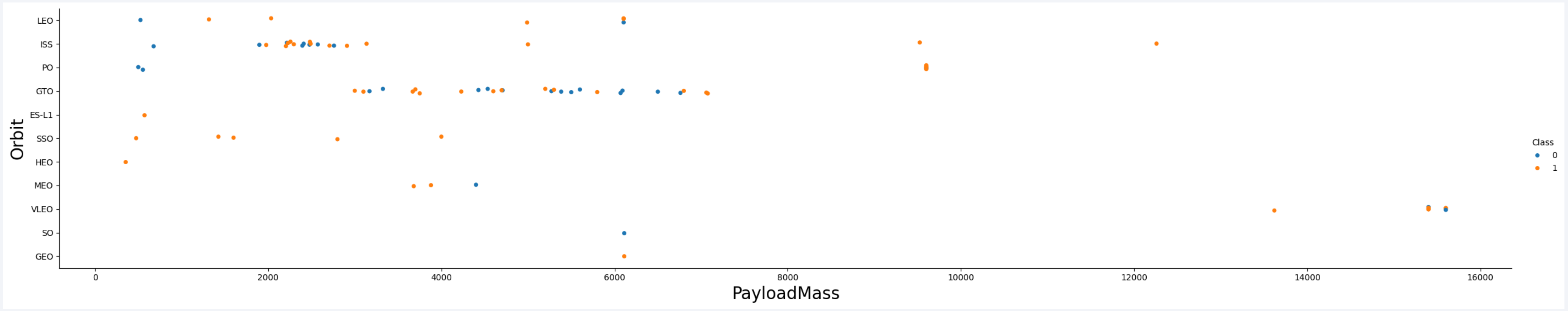
Flight Number vs. Orbit Type

- The plot indicates that SpaceX's landing success rates are highest for ES-L1, GEO, HEO, SSO, and VLEO orbits, while GTO missions have the lowest success rate. The success rates for LEO, MEO, and PO orbits fall in the moderate range.



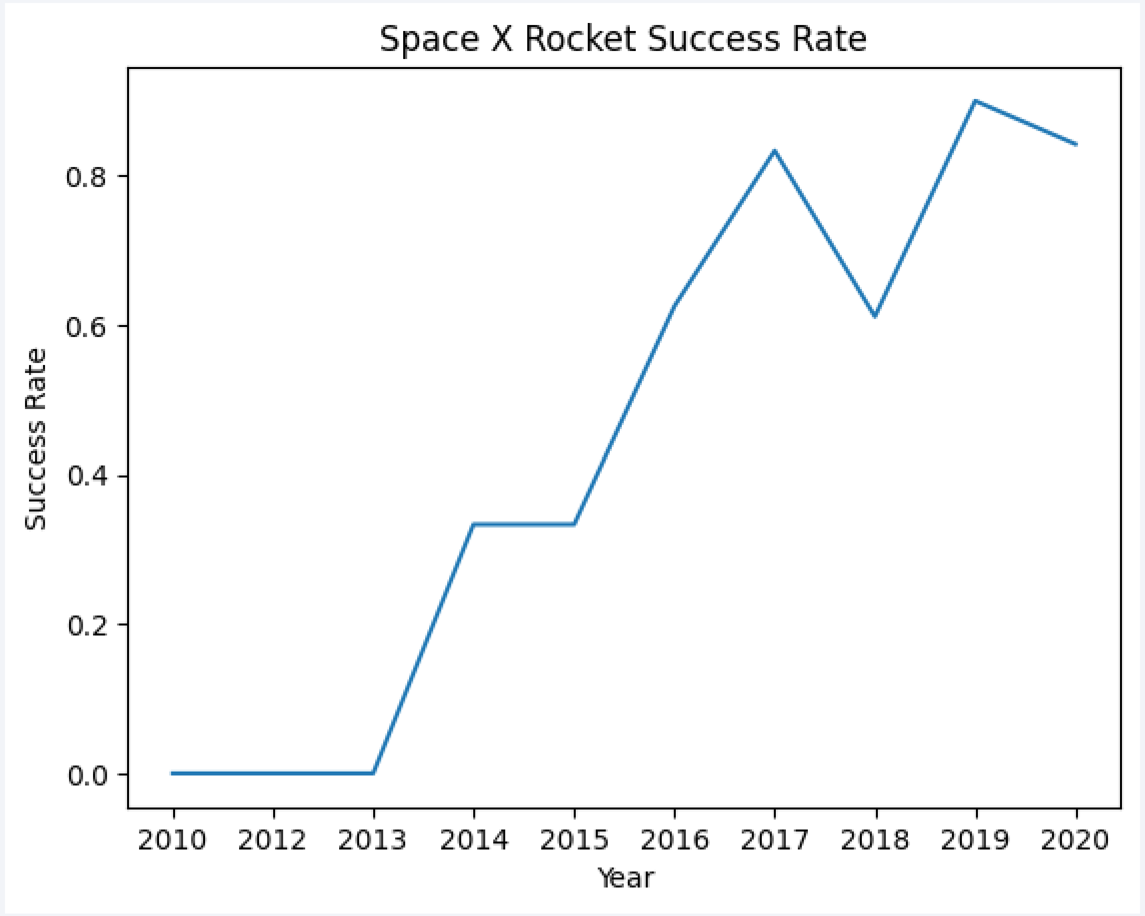
Payload vs. Orbit Type

- The scatter plot shows that for the majority of orbits, SpaceX achieves a mix of successful (Class 1) and failed (Class 0) landings across various payload masses, with heavier payloads (greater than 10,000 kg) being successfully launched primarily to GEO and ES-L1 orbits. There are fewer successful landings for lower payload masses in orbits such as LEO and GTO.



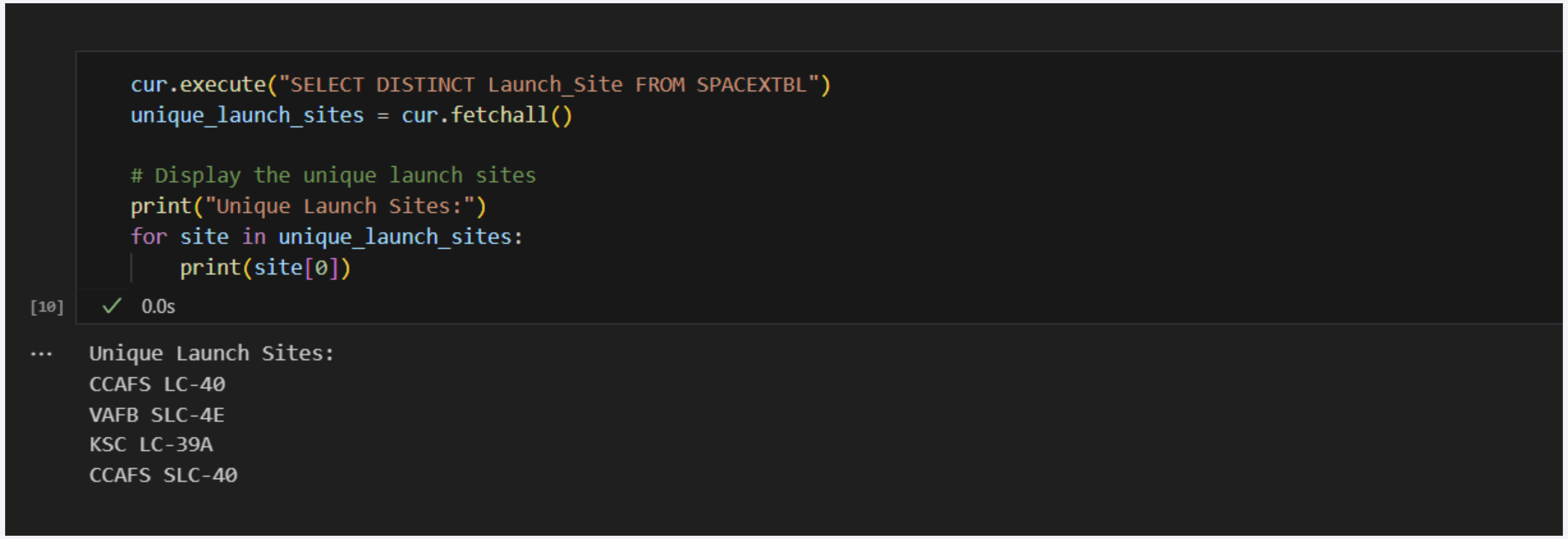
Launch Success Yearly Trend

The line chart shows a significant improvement in SpaceX's rocket success rate over time, starting from near 0% in 2013 to over 80% by 2020. The most notable increases occur between 2015 and 2018, reflecting continuous advancements and improvements in their technology and processes.



All Launch Site Names

We queried the database to retrieve a list of unique launch sites from the SpaceX launch data. Then, we fetched and displayed these unique launch sites, confirming the distinct locations from which SpaceX has launched rockets.



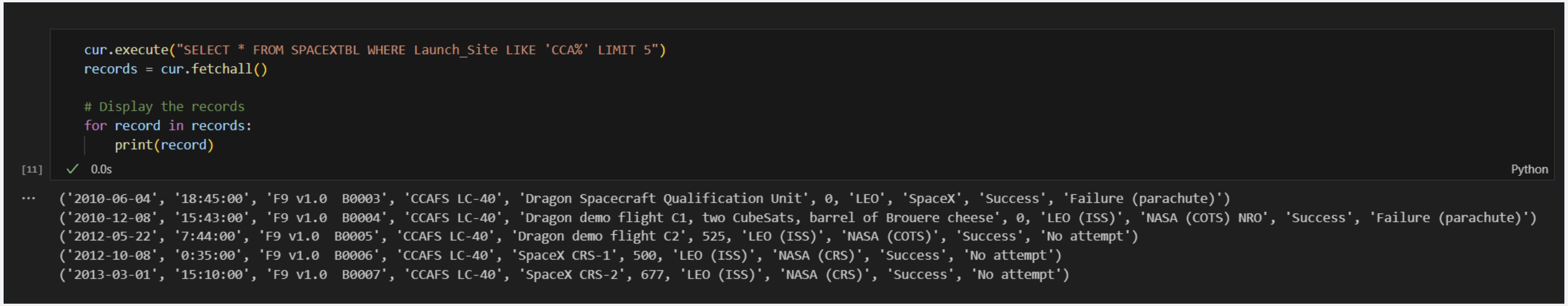
```
cur.execute("SELECT DISTINCT Launch_Site FROM SPACEXTBL")
unique_launch_sites = cur.fetchall()

# Display the unique launch sites
print("Unique Launch Sites:")
for site in unique_launch_sites:
    print(site[0])
```

[10] ✓ 0.0s

```
Unique Launch Sites:
CCAFS LC-40
VAFB SLC-4E
KSC LC-39A
CCAFS SLC-40
```

Launch Site Names Begin with 'CCA'

Query the database to select all records from the SpaceX launch data where the launch site begins with 'CCA', limited to the first five entries. We then fetched and displayed these records, showing detailed information about each launch from the specified site.



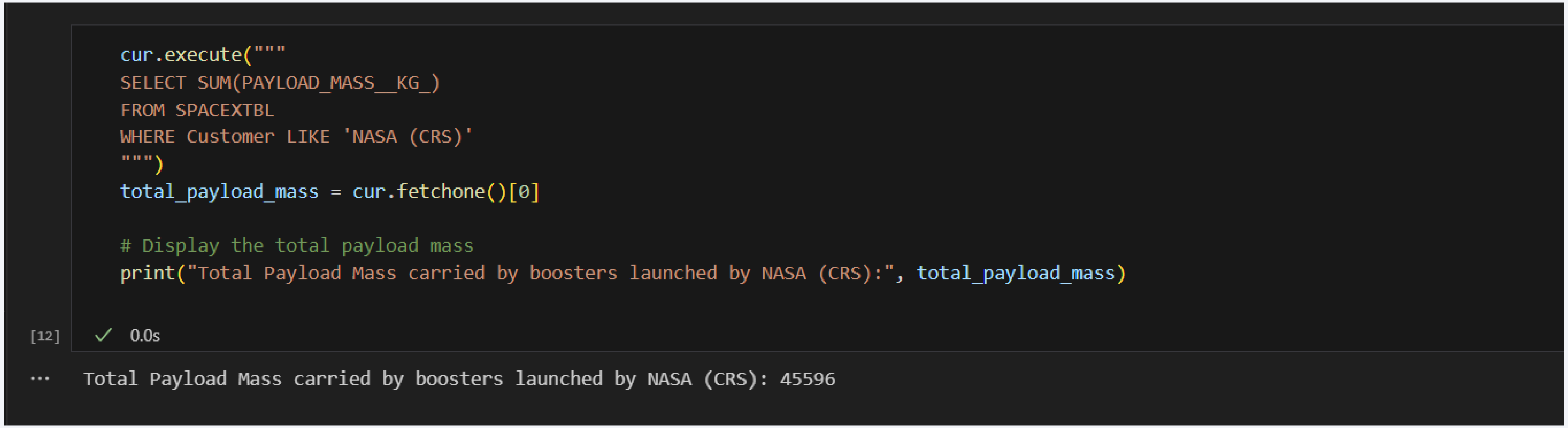
```python
cur.execute("SELECT * FROM SPACEXTBL WHERE Launch_Site LIKE 'CCA%' LIMIT 5")
records = cur.fetchall()

# Display the records
for record in records:
    print(record)
```

[11] ✓ 0.0s Python

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

Total Payload Mass

We queried the database to calculate the total payload mass carried by boosters launched by NASA (CRS). The result, 45596 kg, was then displayed.



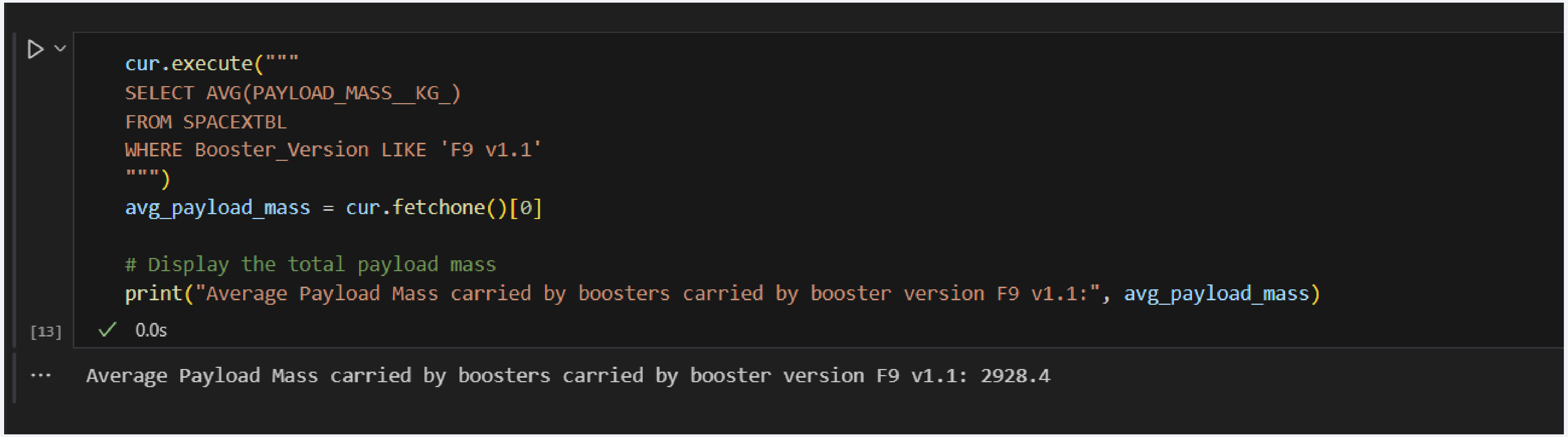
```
cur.execute("""
SELECT SUM(PAYLOAD_MASS__KG_)
FROM SPACEXTBL
WHERE Customer LIKE 'NASA (CRS)'
""")
total_payload_mass = cur.fetchone()[0]

# Display the total payload mass
print("Total Payload Mass carried by boosters launched by NASA (CRS):", total_payload_mass)
```

[12] ✓ 0.0s

```
Total Payload Mass carried by boosters launched by NASA (CRS): 45596
```

Average Payload Mass by F9 v1.1

We calculated the average payload mass carried by boosters of version F9 v1.1. The resulting average payload mass is 2928.4 kg.



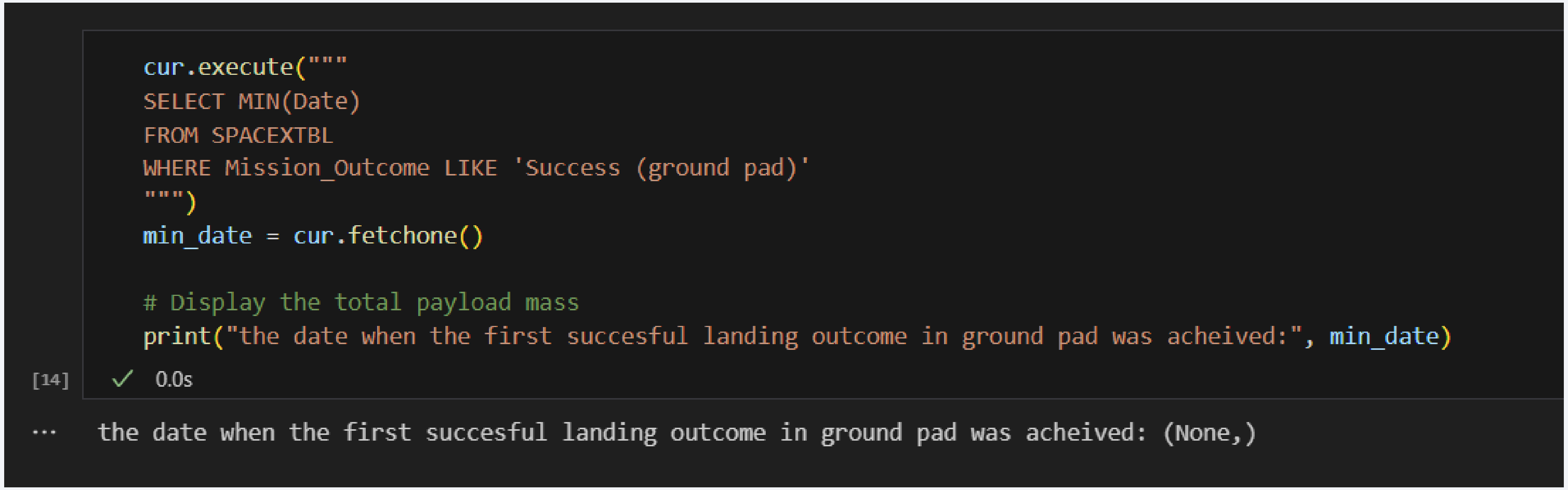
```
cur.execute("""
SELECT AVG(PAYLOAD_MASS__KG_)
FROM SPACEXTBL
WHERE Booster_Version LIKE 'F9 v1.1'
""")
avg_payload_mass = cur.fetchone()[0]

# Display the total payload mass
print("Average Payload Mass carried by boosters carried by booster version F9 v1.1:", avg_payload_mass)
```

[13] ✓ 0.0s

```
Average Payload Mass carried by boosters carried by booster version F9 v1.1: 2928.4
```

First Successful Ground Landing Date

We attempted to find the date of the first successful landing outcome on a ground pad, but no records matching this criterion were found in the database.



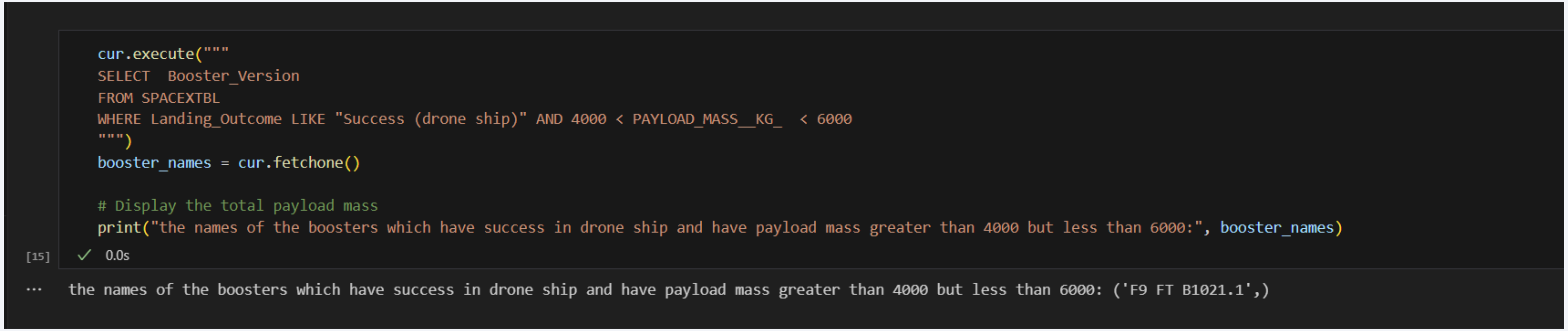
```
cur.execute("""
SELECT MIN(Date)
FROM SPACEXTBL
WHERE Mission_Outcome LIKE 'Success (ground pad)'
""")
min_date = cur.fetchone()

# Display the total payload mass
print("the date when the first succesful landing outcome in ground pad was acheived:", min_date)
```

```
[14]  ✓ 0.0s
```

```
... the date when the first succesful landing outcome in ground pad was acheived: (None,)
```

Successful Drone Ship Landing with Payload between 4000 and 6000

The query retrieves the names of boosters that have successfully landed on a drone ship and carried a payload mass between 4000 and 6000 kg. The result shows that the booster 'F9 FT B1021.1' meets these criteria.



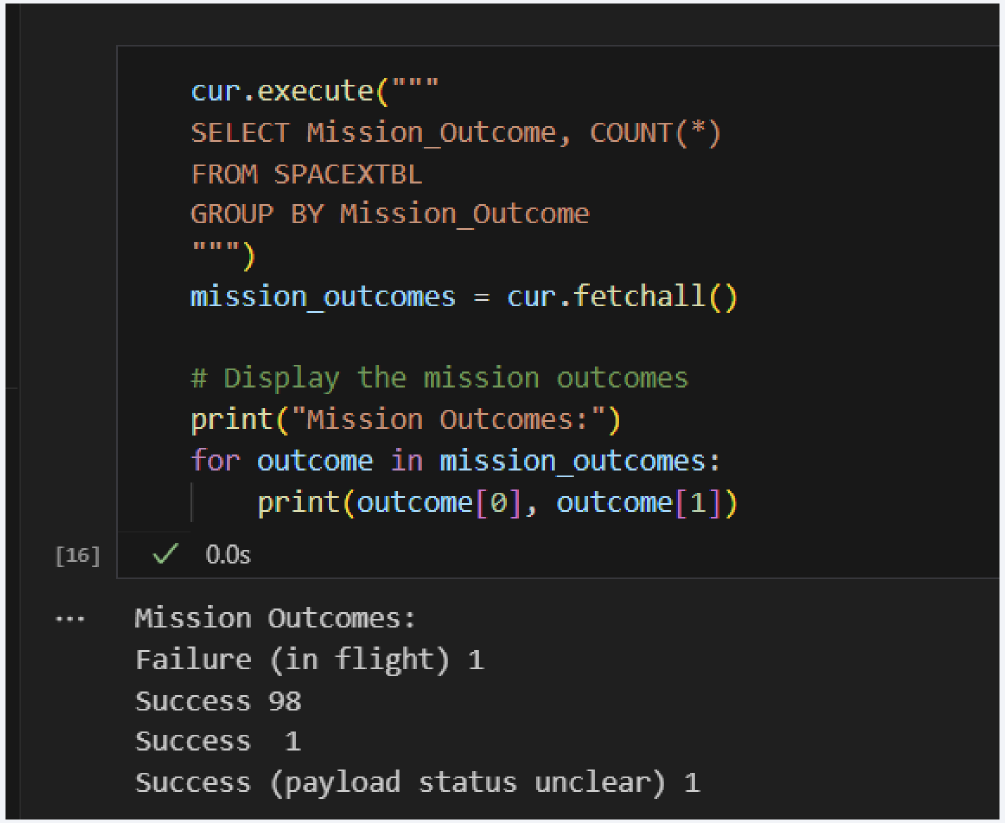
```
cur.execute("""
SELECT  Booster_Version
FROM SPACEXTBL
WHERE Landing_Outcome LIKE "Success (drone ship)" AND 4000 < PAYLOAD_MASS__KG_  < 6000
""")
booster_names = cur.fetchone()

# Display the total payload mass
print("the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000:", booster_names)
```

```
[15]  ✓ 0.0s
```

```
the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000: ('F9 FT B1021.1',)
```

Total Number of Successful and Failure Mission Outcomes

The query groups mission outcomes by their type and counts the number of occurrences of each outcome. The results show the distribution of mission outcomes, indicating that there are 98 successes, 1 in-flight failure, 1 success with unclear payload status, and another success counted separately for some reason.



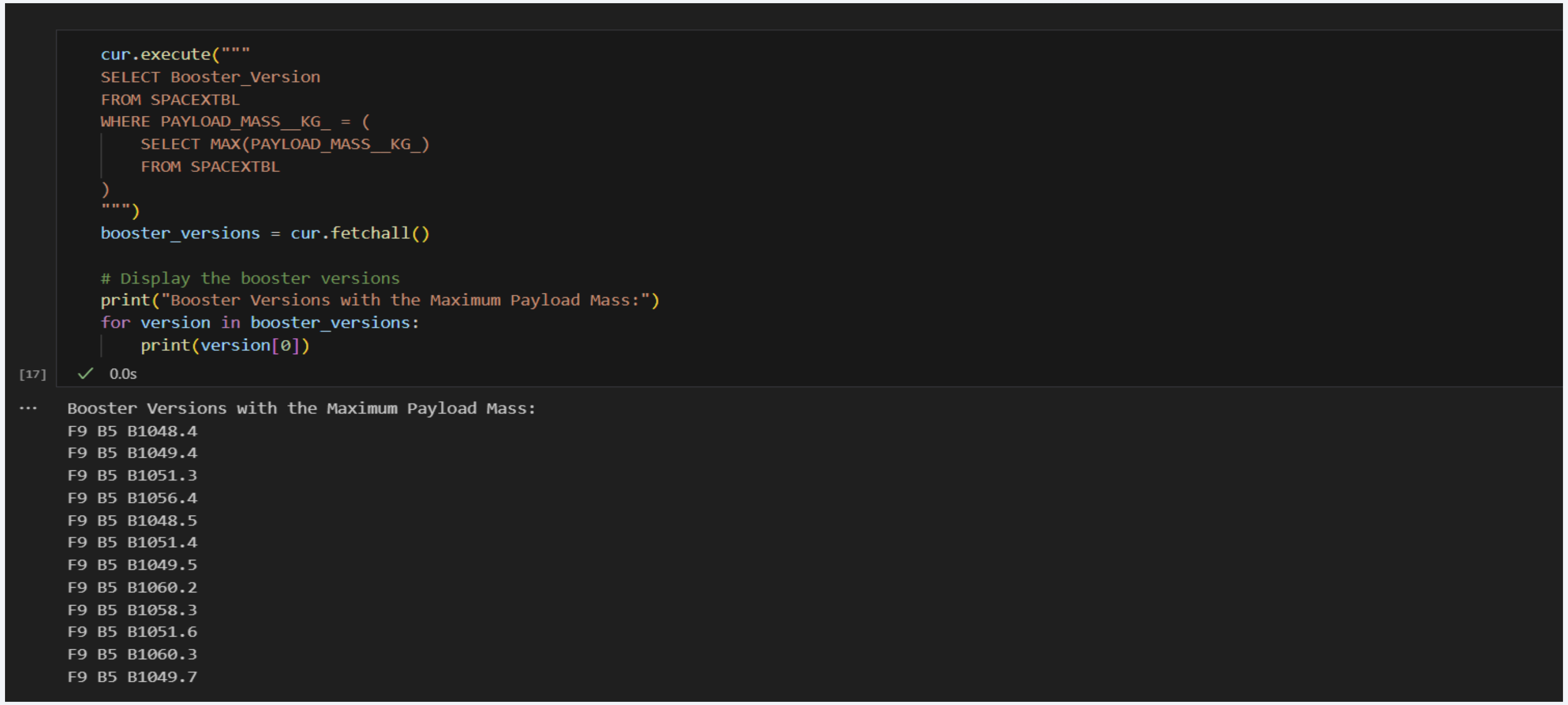
```
cur.execute("""
SELECT Mission_Outcome, COUNT(*)
FROM SPACEXTBL
GROUP BY Mission_Outcome
""")
mission_outcomes = cur.fetchall()

# Display the mission outcomes
print("Mission Outcomes:")
for outcome in mission_outcomes:
    print(outcome[0], outcome[1])
```

[16] ✓ 0.0s

```
Mission Outcomes:
Failure (in flight) 1
Success 98
Success  1
Success (payload status unclear) 1
```

Boosters Carried Maximum Payload

The query identifies booster versions that carried the maximum payload mass by selecting records where the payload mass equals the highest payload mass in the table. The results show the booster versions, all of which are "F9 B5" with different serial numbers, that achieved the maximum payload mass.



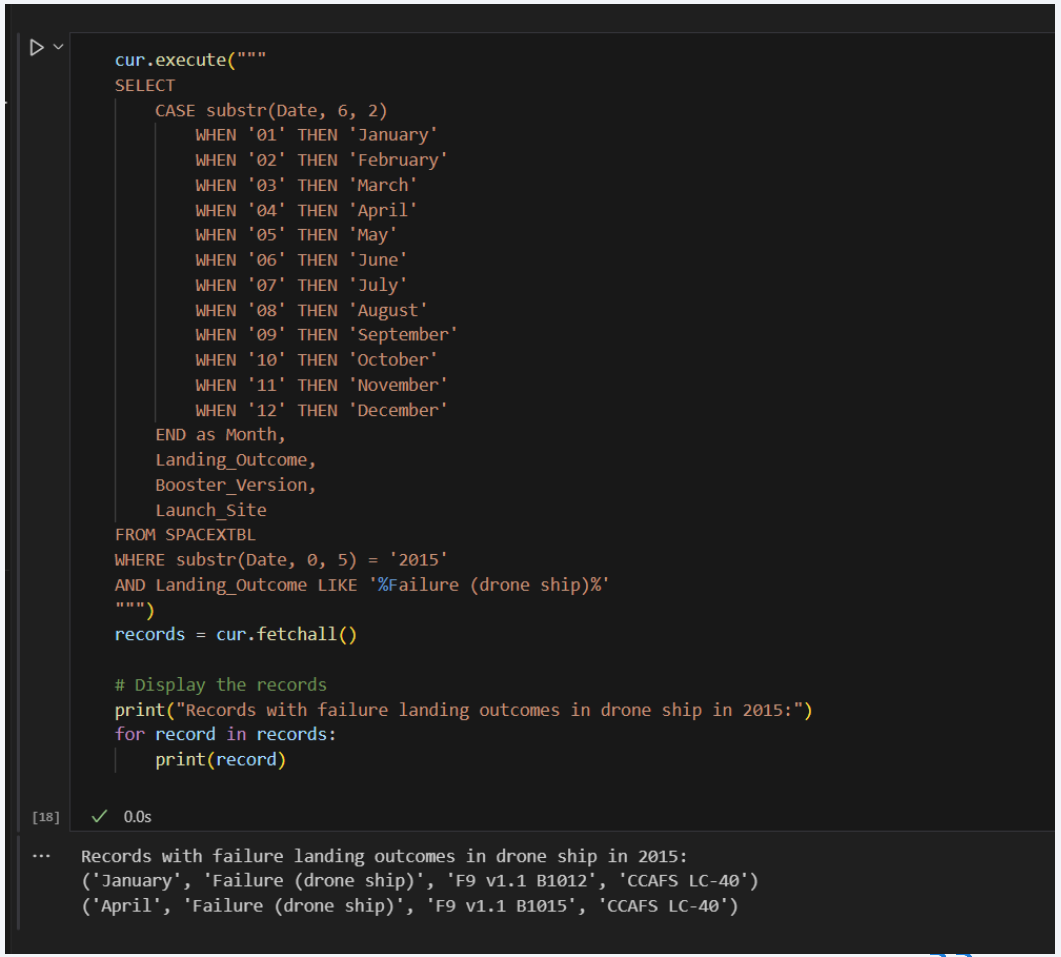
```
cur.execute("""
SELECT Booster_Version
FROM SPACEXTBL
WHERE PAYLOAD_MASS__KG_ = (
    SELECT MAX(PAYLOAD_MASS__KG_)
    FROM SPACEXTBL
)
""")
booster_versions = cur.fetchall()

# Display the booster versions
print("Booster Versions with the Maximum Payload Mass:")
for version in booster_versions:
    print(version[0])
```

[17] ✓ 0.0s

```
Booster Versions with the Maximum Payload Mass:
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

The query retrieves records from the SPACEXTBL table for the year 2015 where the landing outcome was a failure on a drone ship. It converts the numeric month part of the date to its full name, selects additional fields (Landing_Outcome, Booster_Version, Launch_Site), and filters records to include only those from 2015 with the specified landing outcome. The results are fetched and displayed, showing the month, landing outcome, booster version, and launch site for each matching record.



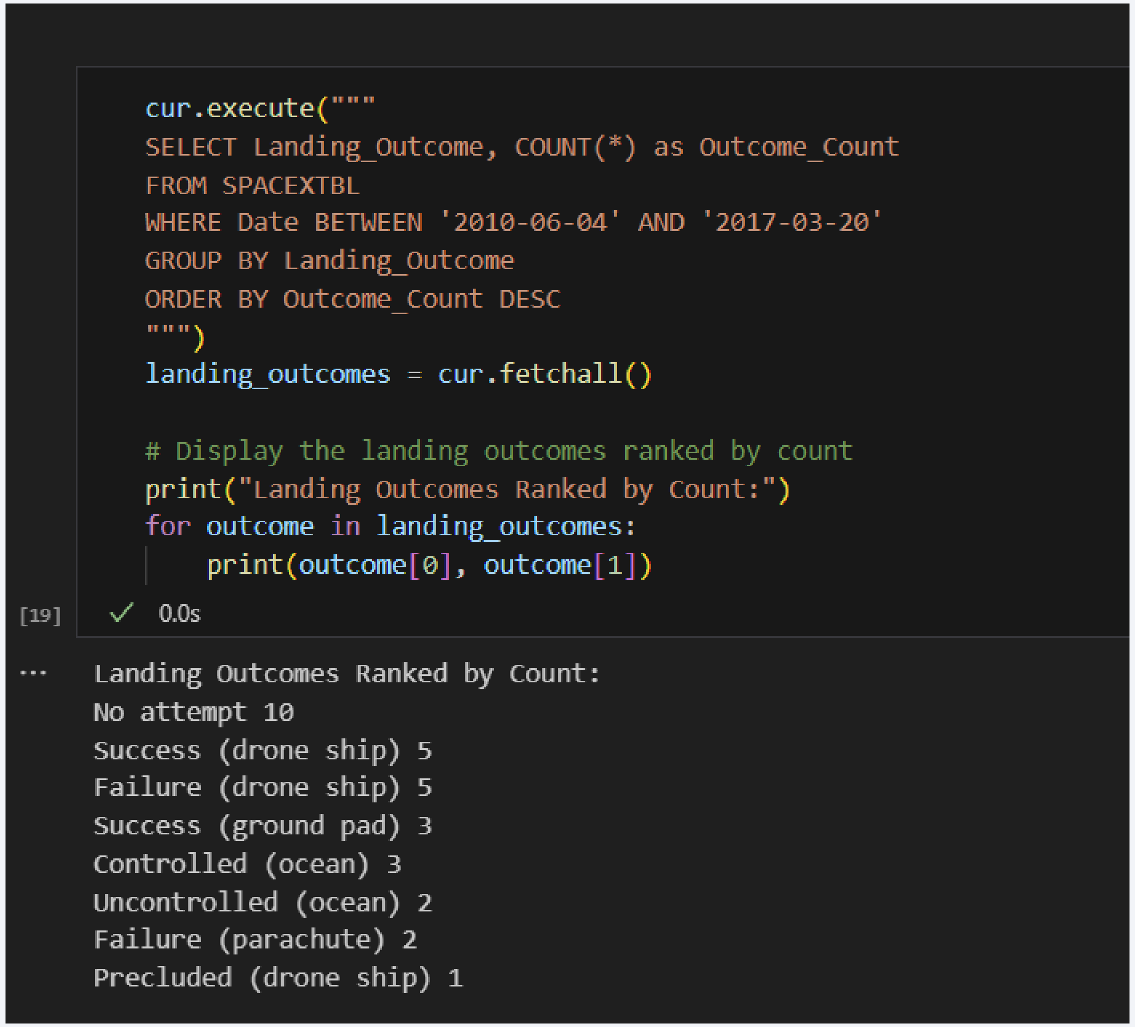
```
cur.execute("""
SELECT
    CASE substr(Date, 6, 2)
        WHEN '01' THEN 'January'
        WHEN '02' THEN 'February'
        WHEN '03' THEN 'March'
        WHEN '04' THEN 'April'
        WHEN '05' THEN 'May'
        WHEN '06' THEN 'June'
        WHEN '07' THEN 'July'
        WHEN '08' THEN 'August'
        WHEN '09' THEN 'September'
        WHEN '10' THEN 'October'
        WHEN '11' THEN 'November'
        WHEN '12' THEN 'December'
    END as Month,
    Landing_Outcome,
    Booster_Version,
    Launch_Site
FROM SPACEXTBL
WHERE substr(Date, 0, 5) = '2015'
AND Landing_Outcome LIKE '%Failure (drone ship)%'
""")
records = cur.fetchall()

# Display the records
print("Records with failure landing outcomes in drone ship in 2015:")
for record in records:
    print(record)
```

[18] ✓ 0.0s

```
Records with failure landing outcomes in drone ship in 2015:
('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

This query retrieves and counts different landing outcomes from the SPACEXTBL table between June 4, 2010, and March 20, 2017. The results are grouped by Landing_Outcome, ordered by count in descending order, fetched, and then displayed. The output shows the landing outcomes ranked by their frequency.



```
cur.execute("""
SELECT Landing_Outcome, COUNT(*) as Outcome_Count
FROM SPACEXTBL
WHERE Date BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY Landing_Outcome
ORDER BY Outcome_Count DESC
""")
landing_outcomes = cur.fetchall()

# Display the landing outcomes ranked by count
print("Landing Outcomes Ranked by Count:")
for outcome in landing_outcomes:
    print(outcome[0], outcome[1])
```

[19] ✓ 0.0s

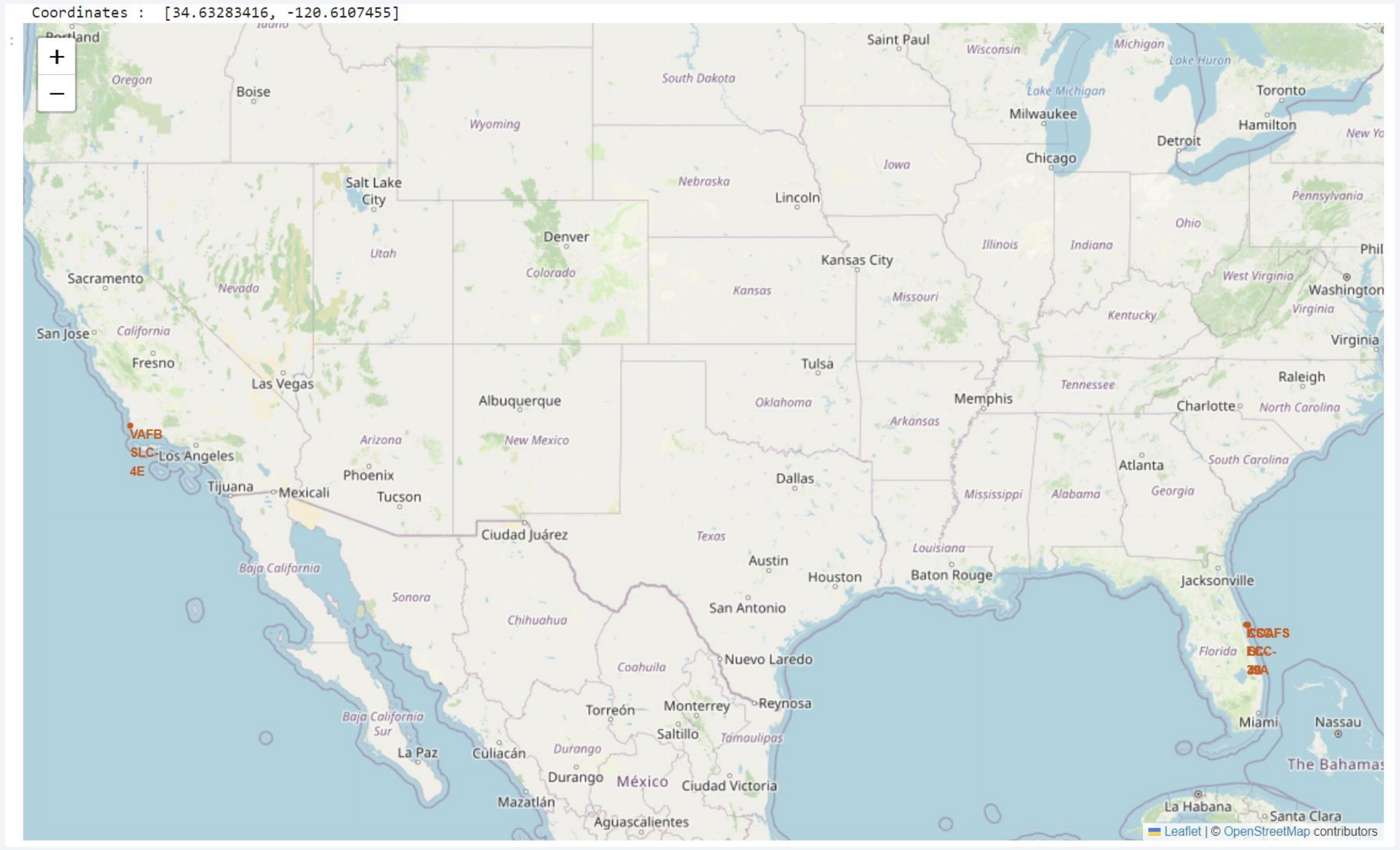
```
Landing Outcomes Ranked by Count:
No attempt 10
Success (drone ship) 5
Failure (drone ship) 5
Success (ground pad) 3
Controlled (ocean) 3
Uncontrolled (ocean) 2
Failure (parachute) 2
Precluded (drone ship) 1
```

Section 3

Launch Sites Proximities Analysis

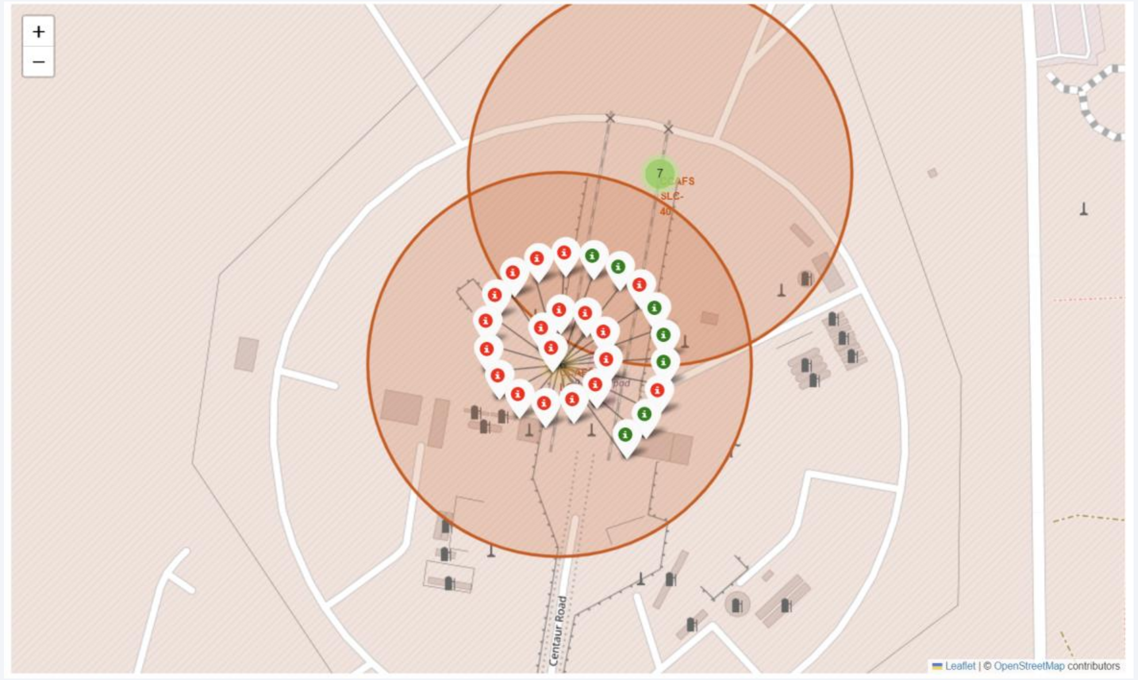
Generated Map With Marked Launch Sites

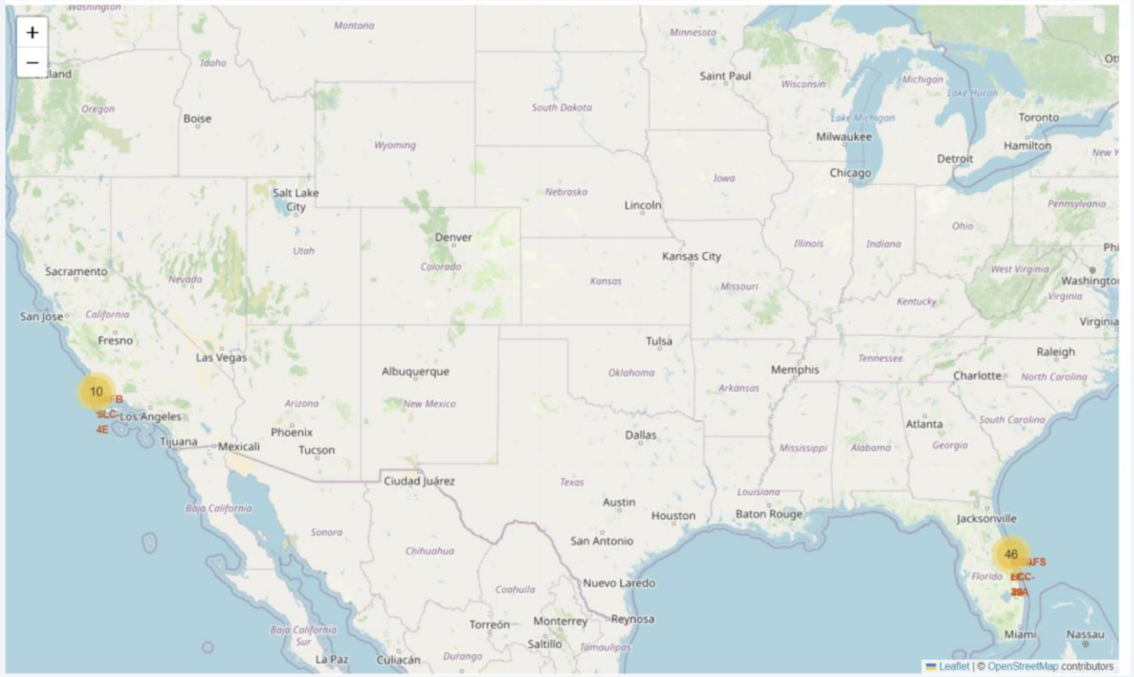
The screenshot shows launch site markers on a folium map, clustered mainly on the east and west coasts of the United States, specifically in California and Florida. This distribution indicates strategic locations for safer launches over open water, enhancing flexibility and access to various orbits.


Coordinates : [34.63283416, -120.6107455]

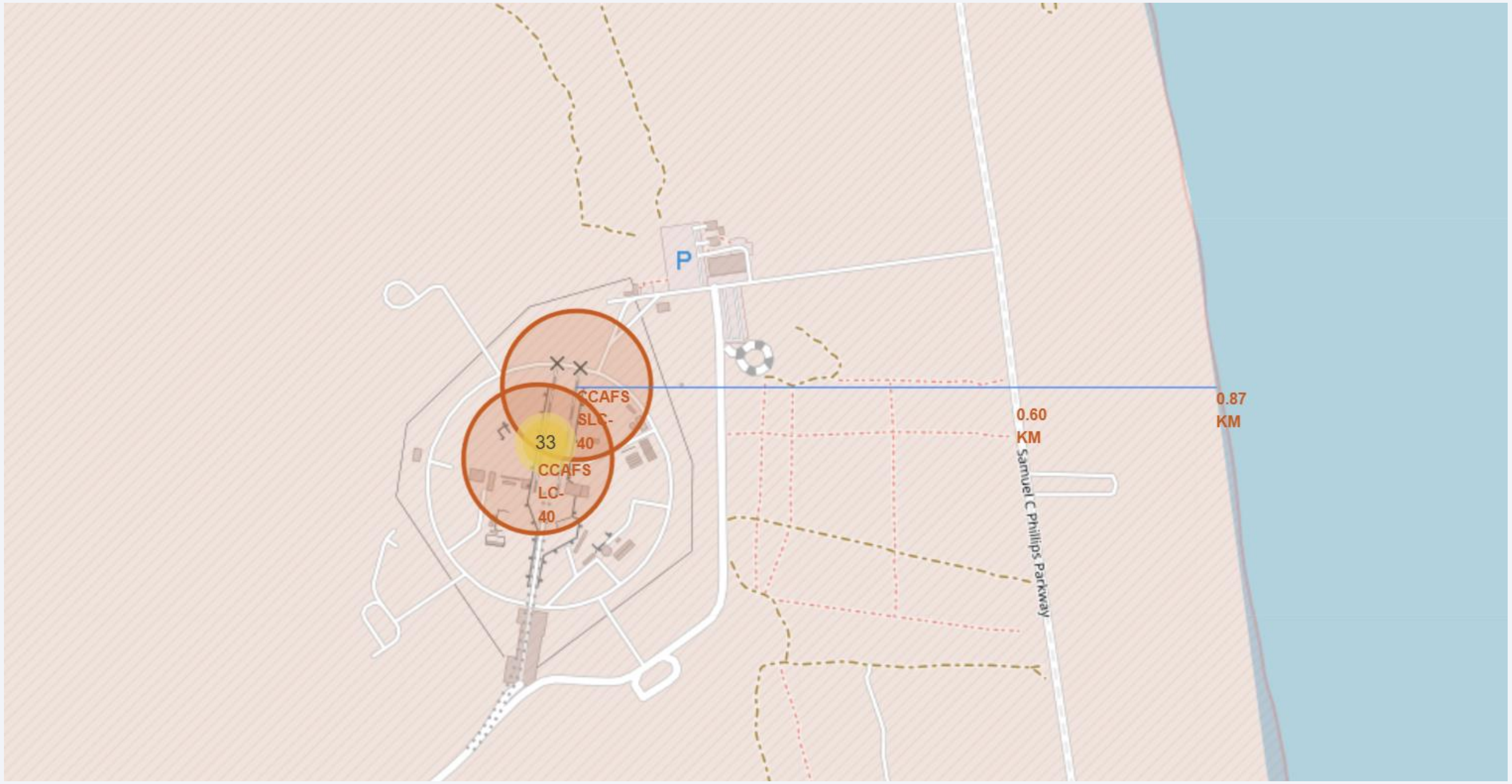
Mark the success/failed launches for each site on the map

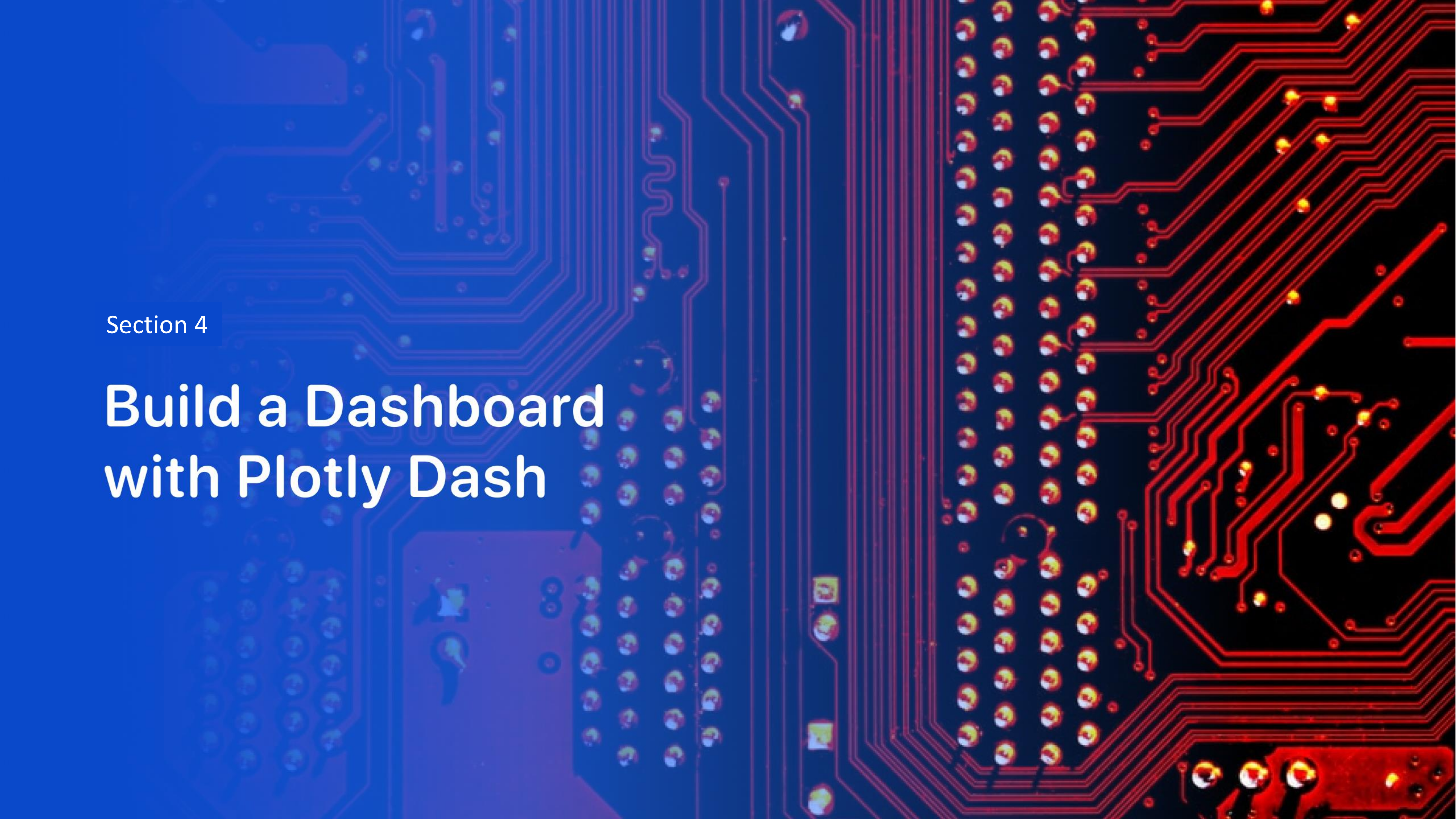
we created markers for all launch records. If a launch was successful, then we use a green marker and if a launch was failed, we use a red marker.





Calculate the distances between a launch site to its proximities

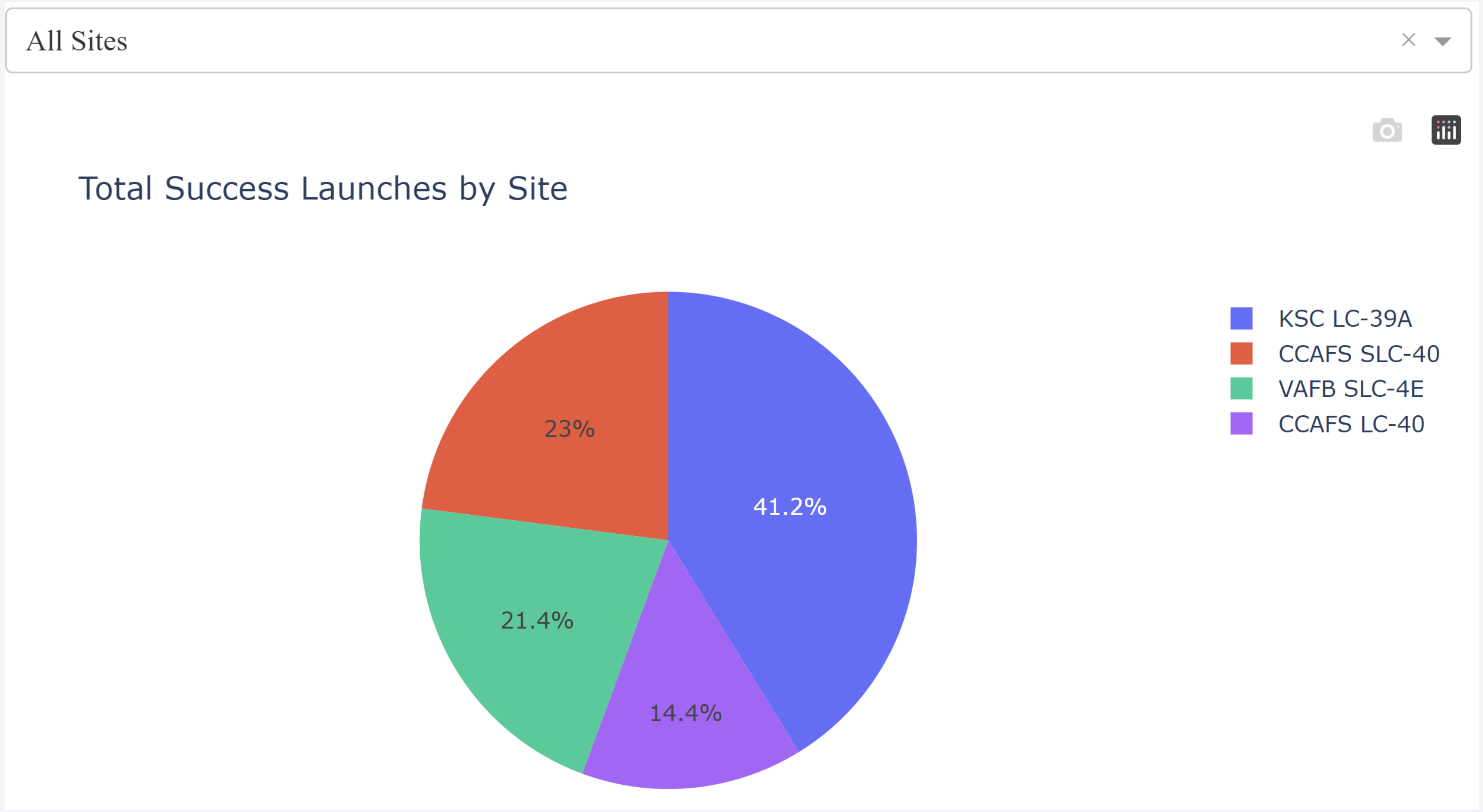




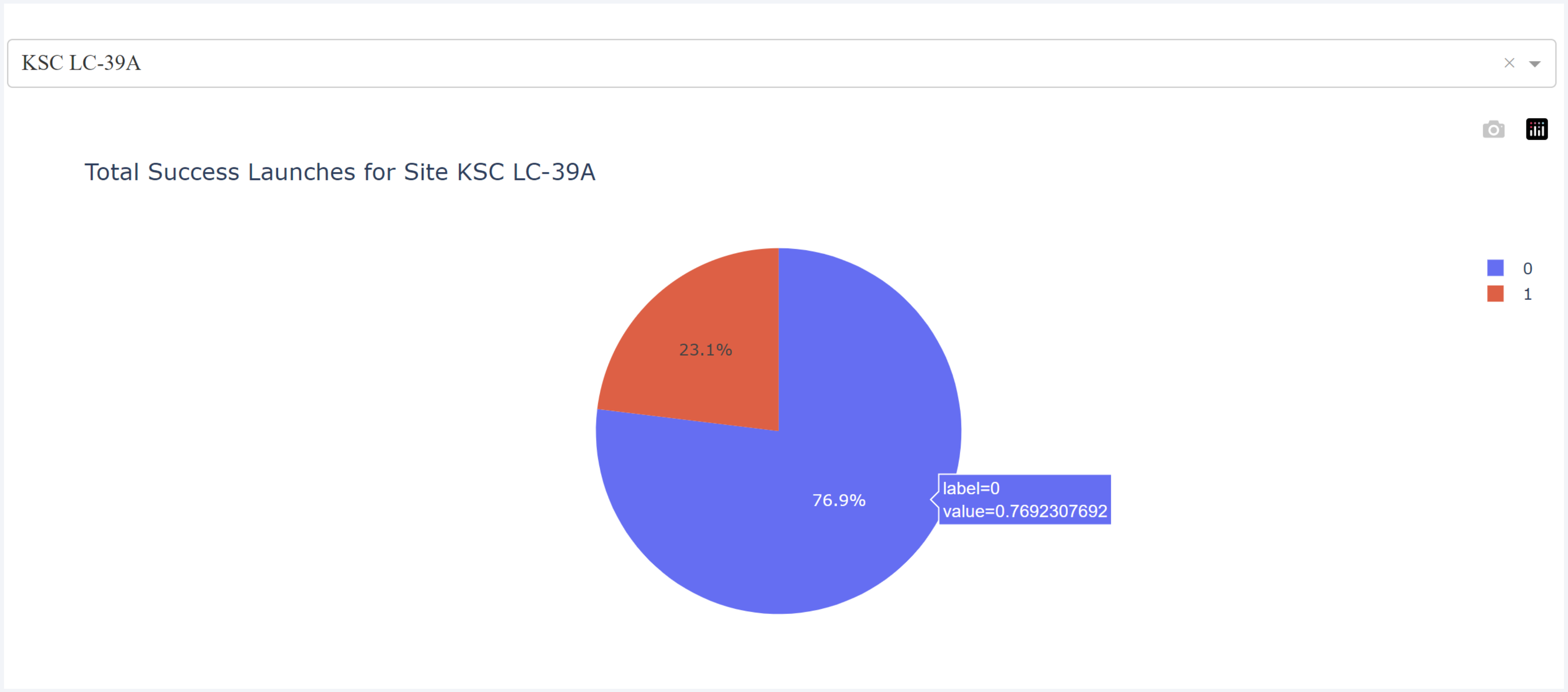
Section 4

Build a Dashboard with Plotly Dash

Count for all sites, in a Piechart

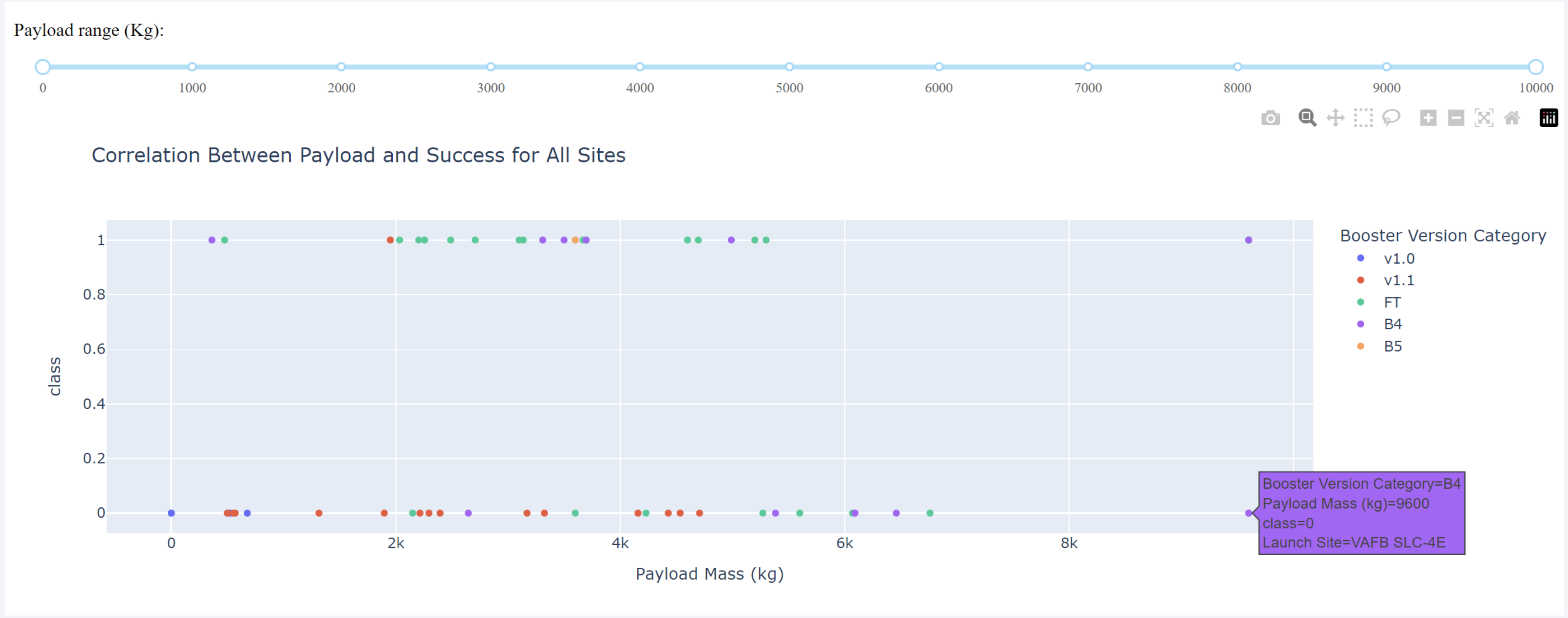


Piechart for the launch site with highest launch success ratio



Scatter plot of Payload vs Launch Outcome for all sites, with different payload

- In the scatter plot, payload mass doesn't significantly impact success rate, and various booster versions exhibit similar reliability.

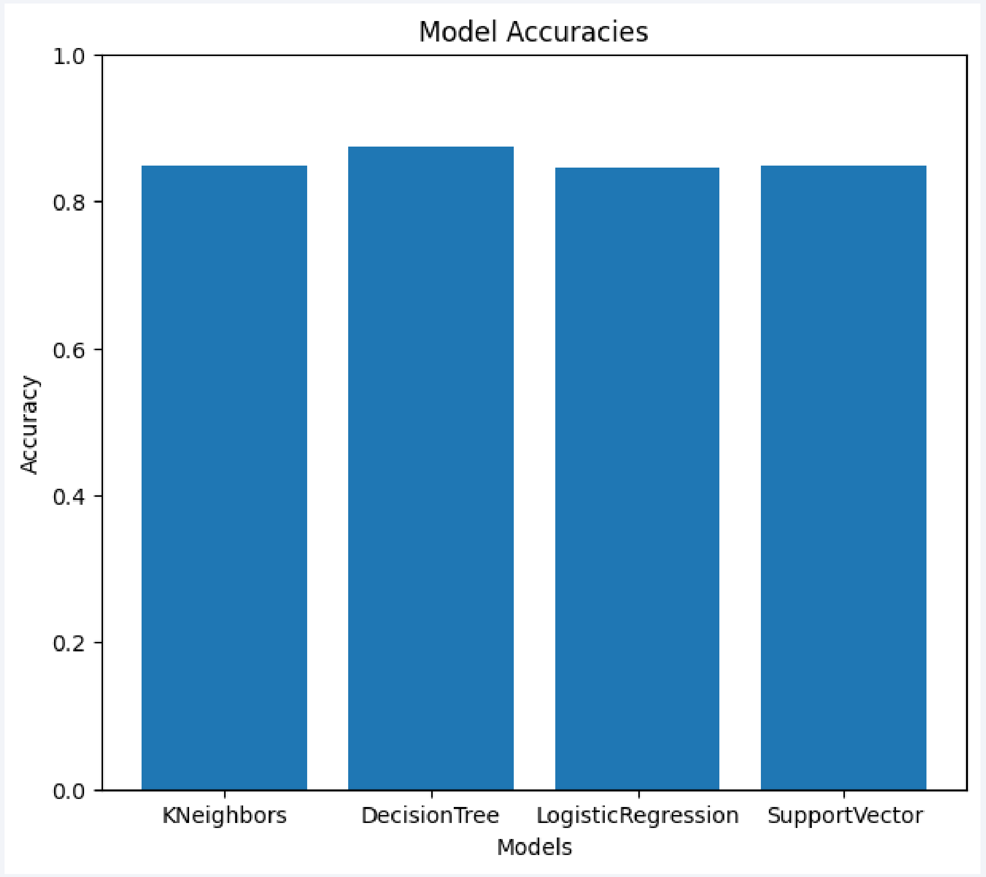


Section 5

Predictive Analysis (Classification)

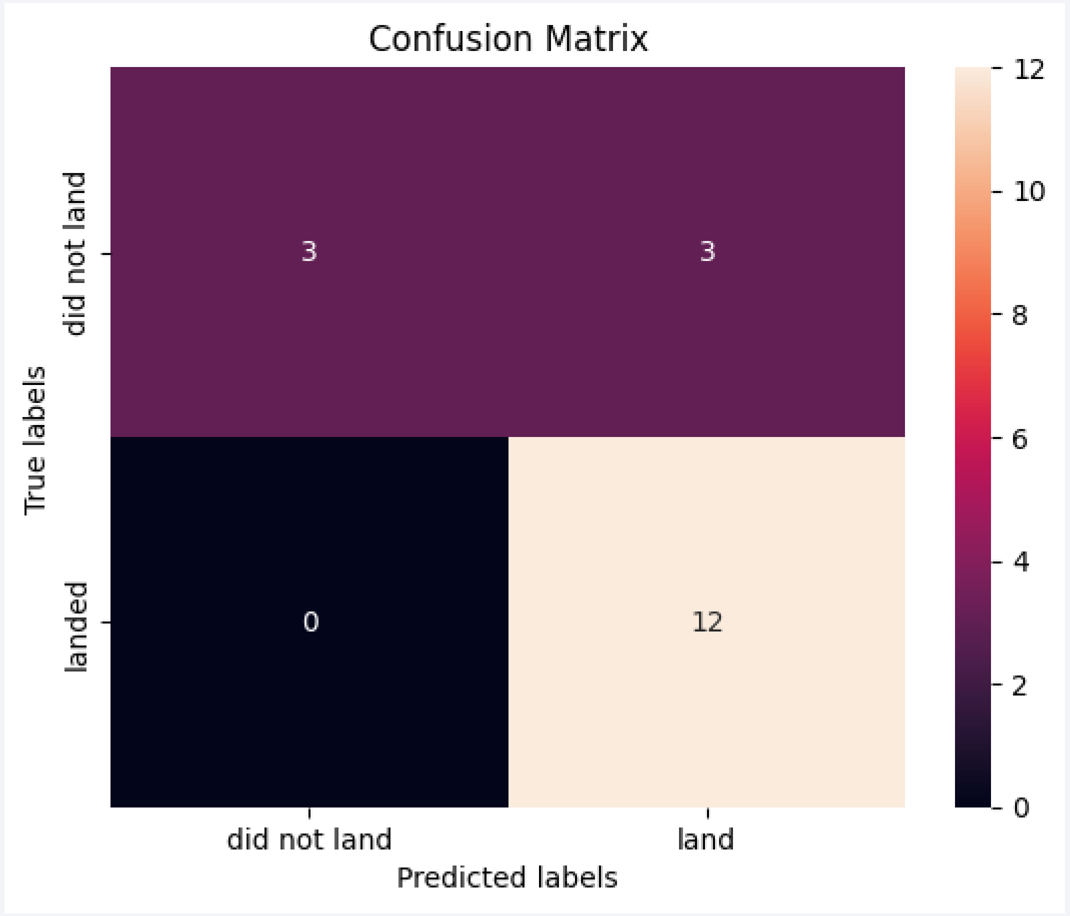
Classification Accuracy

- Best Model:
  - Model: Decision Tree Classifier
  - Accuracy Score: 0.8732
- Best Hyperparameters:
  - Criterion: gini
  - Max Depth: 6
  - Max Features: auto
  - Min Samples Leaf: 2
  - Min Samples Split: 5
  - Splitter: random



Confusion Matrix

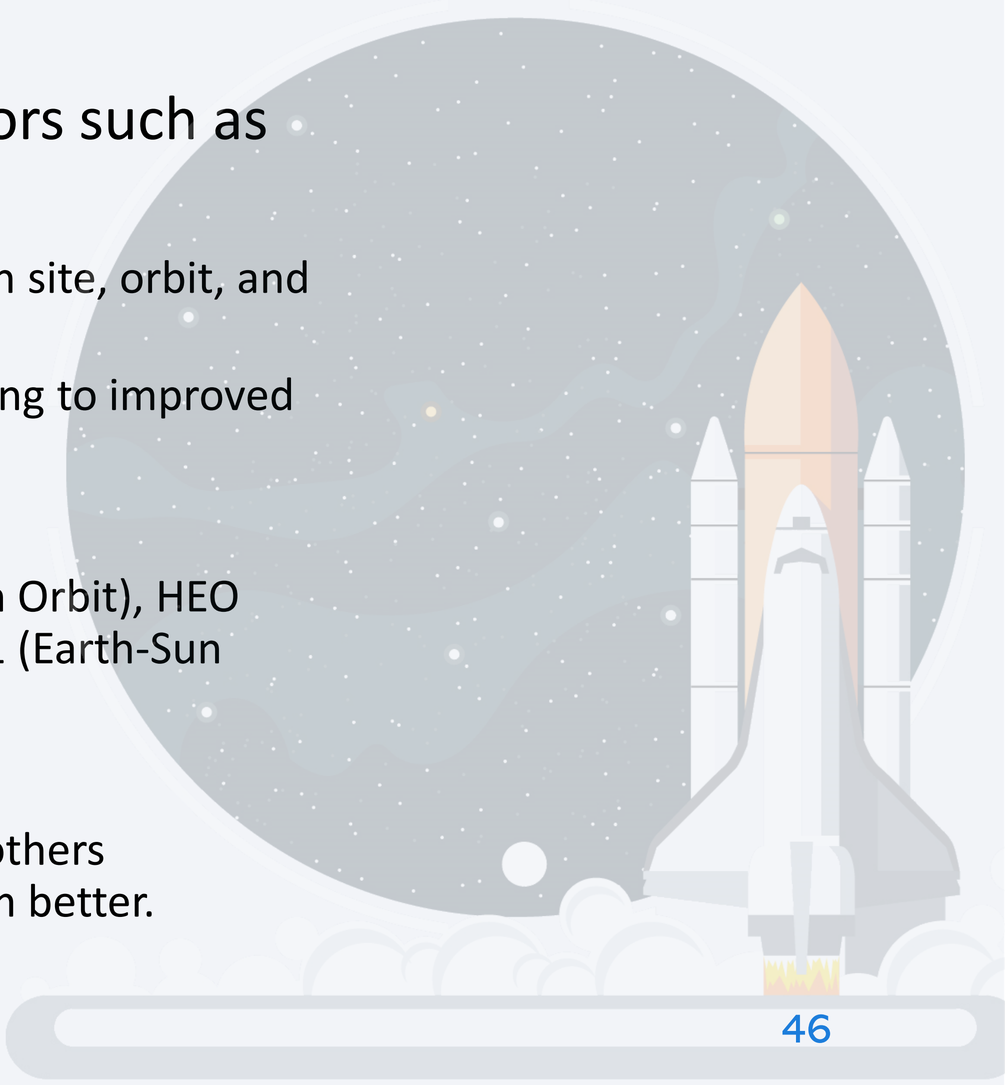
- The matrix shows that the Decision Tree Classifier has a high number of true positives and true negatives, indicating that the model accurately predicts the majority of the test instances. The false positives and false negatives are relatively low, suggesting that the model has a good balance between sensitivity and specificity.



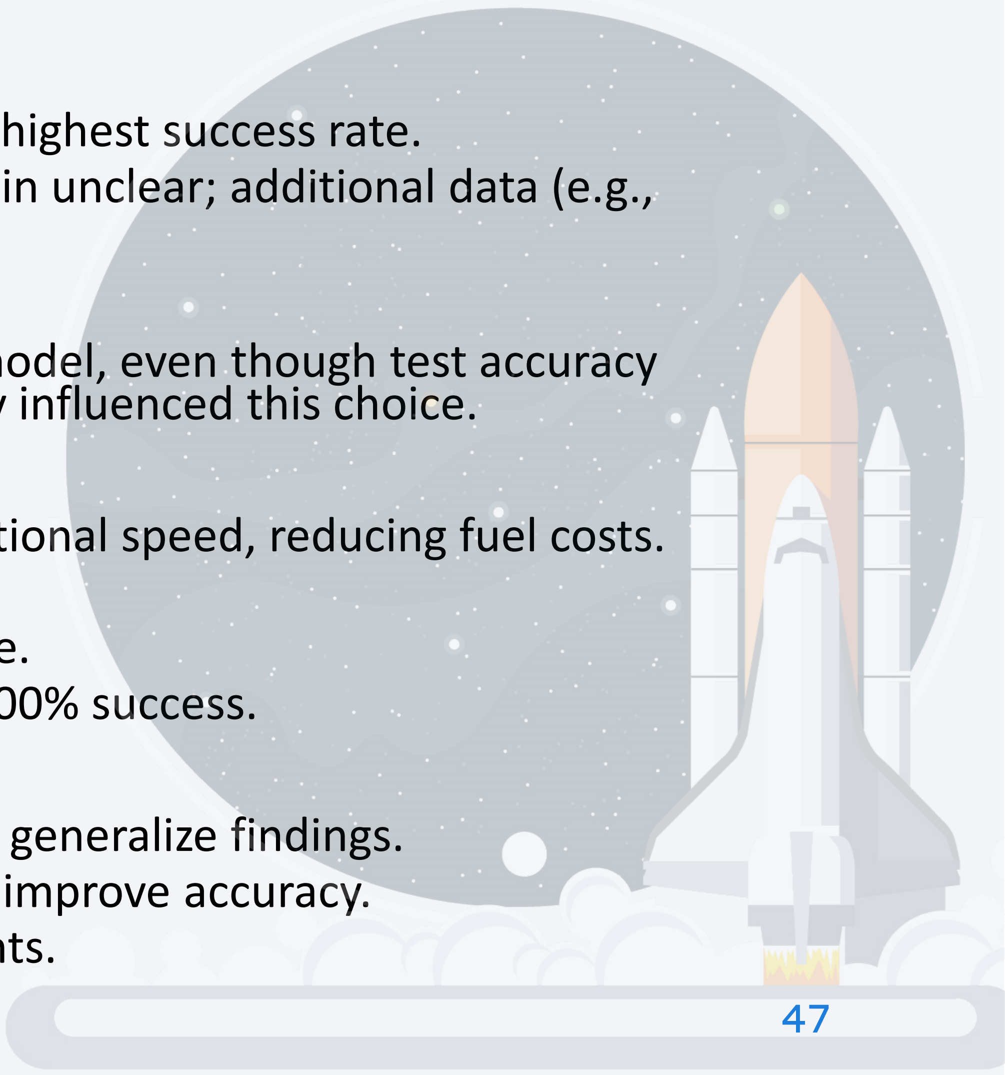
Conclusions

The success of a mission is influenced by several factors such as

1. Factors Influencing Launch Success:
   - Launch success depends on several factors, including the launch site, orbit, and cumulative experience from previous launches.
   - Notably, there has been a knowledge gain over time, contributing to improved success rates.
2. Orbits with High Success Rates:
   - Orbits with the best success rates are GEO (Geostationary Earth Orbit), HEO (Highly Elliptical Orbit), SSO (Sun-Synchronous Orbit), and ES-L1 (Earth-Sun Lagrange Point 1).
3. Payload Mass and Success:
   - Payload mass matters: Some orbits favor light payloads, while others accommodate heavier ones. Generally, lighter payloads perform better.



Conclusions



4. Launch Sites and Success:
    - KSC LC-39A stands out as the best launch site, with the highest success rate.
    - However, the reasons behind site-specific success remain unclear; additional data (e.g., atmospheric conditions) could provide insights.
5. Machine Learning Model Choice:
    - The Decision Tree Algorithm was selected as the best model, even though test accuracy across models was similar. Its superior training accuracy influenced this choice.
6. Additional Insights from Research:
    - Launch sites near the equator benefit from Earth’s rotational speed, reducing fuel costs.
    - All launch sites are close to coastlines.
    - Launch success rates have generally increased over time.
    - Orbits ES-L1, GEO, HEO, and SSO consistently achieve 100% success.
7. Considerations for Future Work:
    - A larger dataset could enhance predictive analytics and generalize findings.
    - Feature analysis or principal component analysis might improve accuracy.
    - Exploring models like XGBoost could yield further insights.

Thank you!