IBM Developer SKILLS NETWORK

Winning Space Race with Data Science

Ela Miljkovic

May 27, 2025

Outline

Executive Summary

Summary of methodologies

- Data Collection
- Data Wrangling
- Exploratory Data Analysis with Data Visualization
- Exploratory Data Analysis with SQL
- Building an Interactive Map with Folium
- Building a Dashboard with Plotly Dash
- Predictive Analysis (Classification)

Summary of all results

- Exploratory Data Analysis Results
- Interactive Map and Dashboard
- Predictive Results

Introduction

Project background and context

•SpaceX advertises Falcon9 rocket launches on its website with a cost of $62 million, significantly lower than its competitors (upward of $165 million each).

•SpaceX can charge lower launch costs because it can reuse Falcon9's first stage.

- If we can determine whether the first stage will land, we can determine the cost of the launch. Competitor companies can use this information to bid against SpaceX.
- The goal of this project is to produce a machine learning pipeline to predict if the first stage will land successfully.

Problems you want to find answers

- What factors will determine if the rocket landing was successful or not?
- Does the rate of successful landings increase over time?

Section 1

Methodology

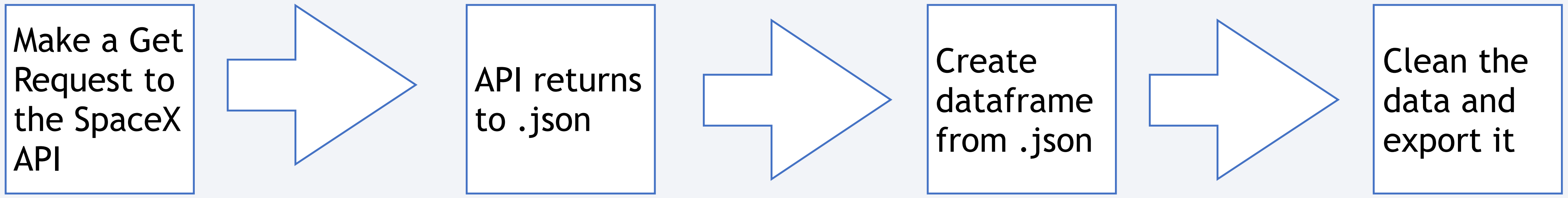
Methodology

Executive Summary

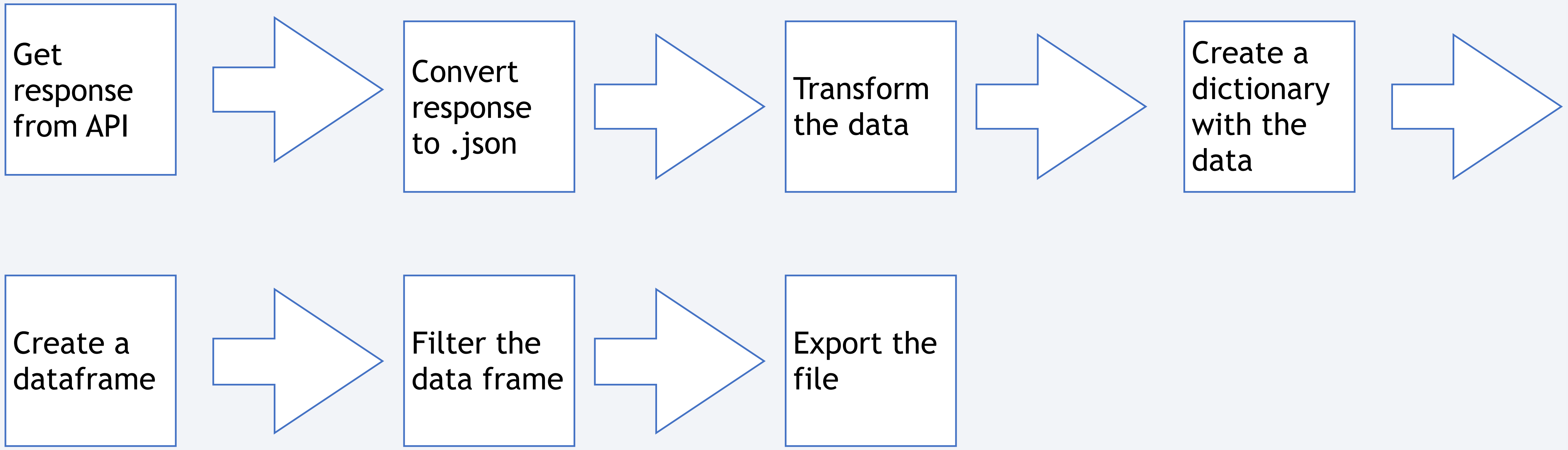
- Data collection methodology:
    - Data from SapceX API and web scraping
- Perform data wrangling
    - Raw data was cleaned, transformed, and prepared.
    - Patterns and trends were identified and missing or incomplete data was addressed and/or removed.
    - Data was formatted and structured into a usable format.
- Perform exploratory data analysis (EDA) using visualization and SQL
    - Conduct exploratory data analysis using a database to see if the data can be used to determine if Falcon9's first stage will land.
- Perform interactive visual analytics using Folium and Plotly Dash
    - Build an interactive dashboard that contains pie charts and scatter plots to analyze data with the Plotly Dash Python library.
    - Calculate distances on an interactive map by writing Python code using the Folium library.
    - Generate interactive maps, plot coordinates, and mark clusters by writing Python code using the Folium library.
- Perform predictive analysis using classification models
    - Split the data into training testing data
    - Train different classification models
    - Optimize the Hyperparameter grid search
    - Utilize your machine learning skills to build a predictive model to help a business function more efficiently

Data Collection

- Describe how data sets were collected.
  - Acquire data by the API, including rocket, launches, and payload information.


Make a Get Request to the SpaceX API
API returns to .json
Create dataframe from .json
Clean the data and export it

Data Collection – SpaceX API

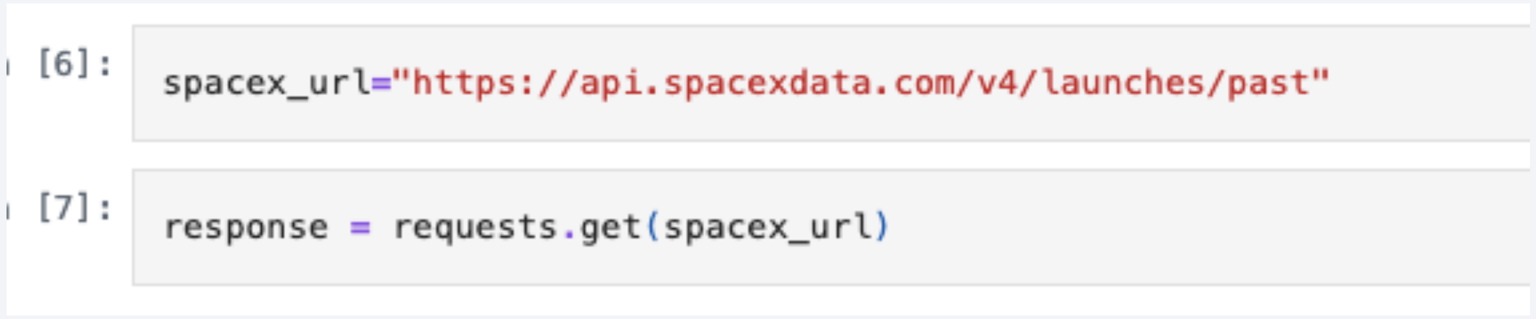


DataScienceCapstone/jupyter-labs-spacex-data-collection-api.ipynb (OR see link below):

https://github.com/elamiljkovic/DataScienceCapstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

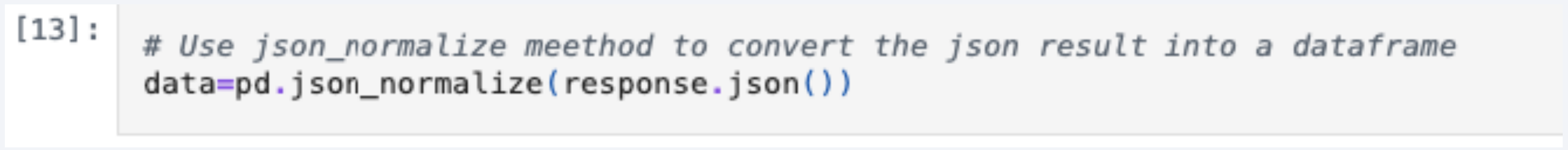
Data Collection – SpaceX API

1. Get response from API:



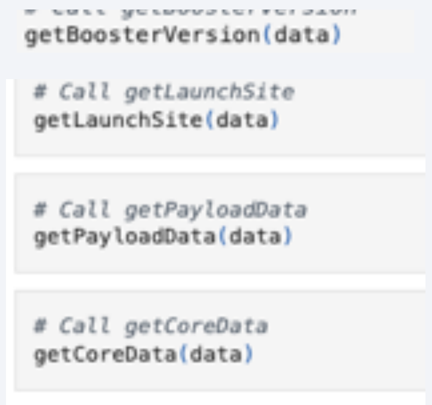
```
[6]: spacex_url="https://api.spacexdata.com/v4/launches/past"

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

2. Convert response to .JSON file:



```
[13]: # Use json_normalize meethod to convert the json result into a dataframe
      data=pd.json_normalize(response.json())
```

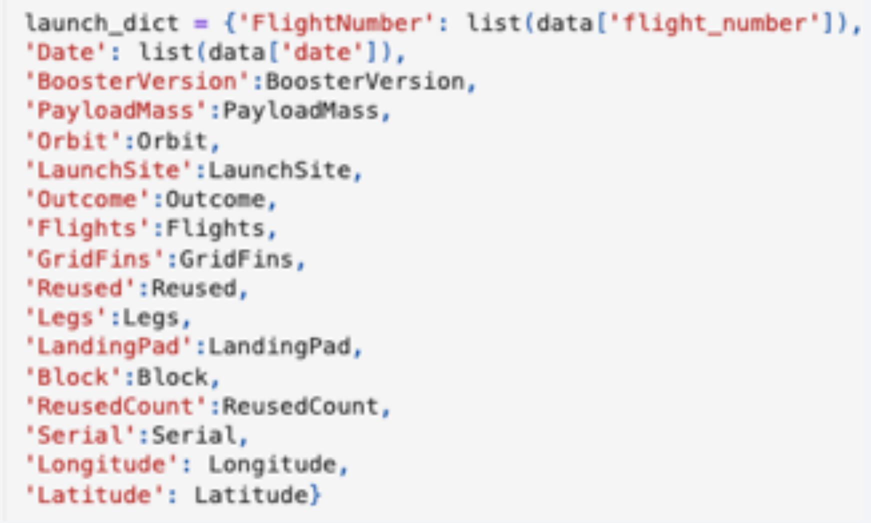
3. Transform the data:



```
getBoosterVersion(data)

# Call getLaunchSite
getLaunchSite(data)

# Call getPayloadData
getPayloadData(data)

# Call getCoreData
getCoreData(data)
```

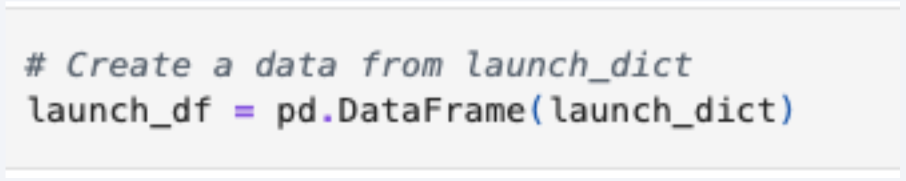
Data Collection – SpaceX API

4. Create dictionary from data:



```
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}
```

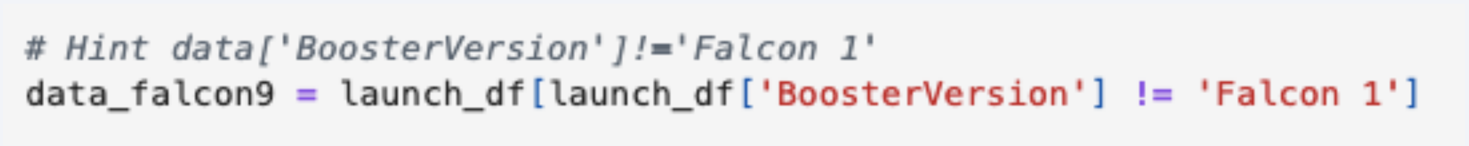
5. Create Dataframe:



```
# Create a data from launch_dict
launch_df = pd.DataFrame(launch_dict)
```

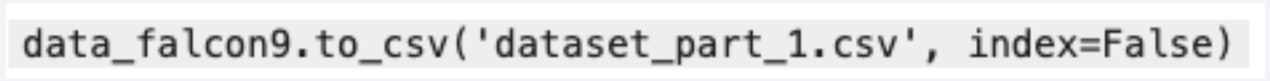
Data Collection – SpaceX API

6. Filter the Dataframe:



```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = launch_df[launch_df['BoosterVersion'] != 'Falcon 1']
```

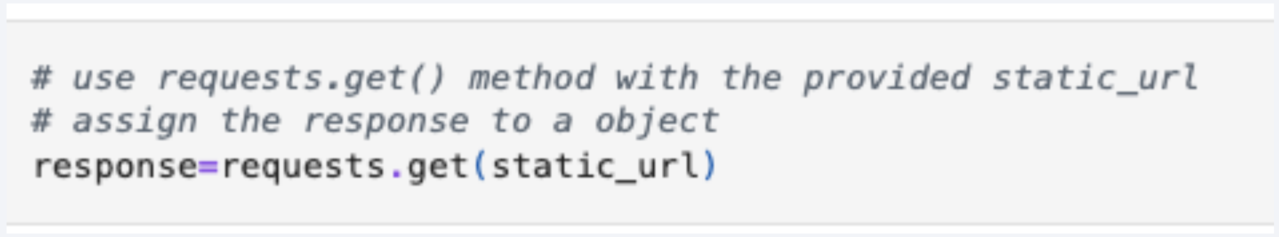
7. Export the file:



```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

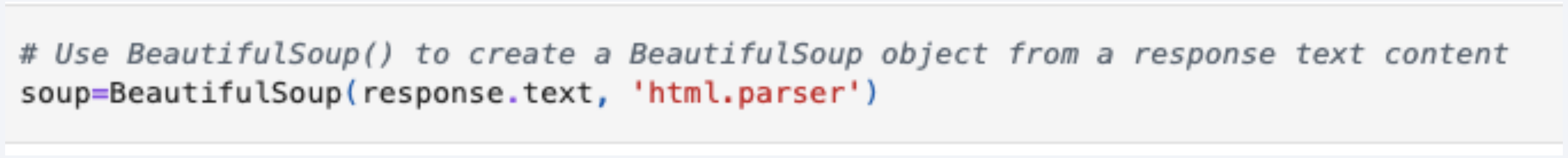
Data Collection – Data Scraping

1. Get a response from HTML:



```
# use requests.get() method with the provided static_url
# assign the response to a object
response=requests.get(static_url)
```

2. Create a BeautifulSoup Object:



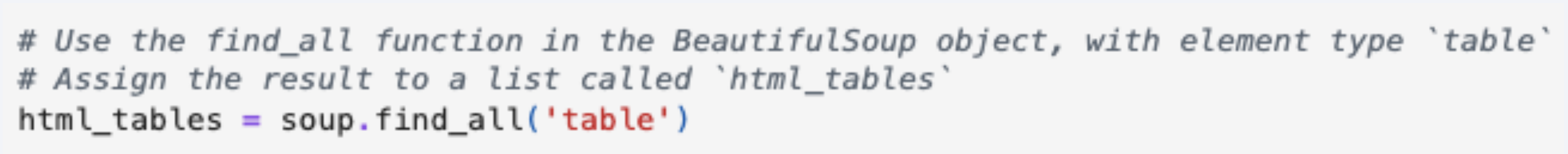
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup=BeautifulSoup(response.text, 'html.parser')
```

DataScienceCapstone/jupyter-labs-webscraping (1).ipynb OR Link below:

https://github.com/elamiljkovic/DataScienceCapstone/blob/main/jupyter-labs-webscraping%20(1).ipynb

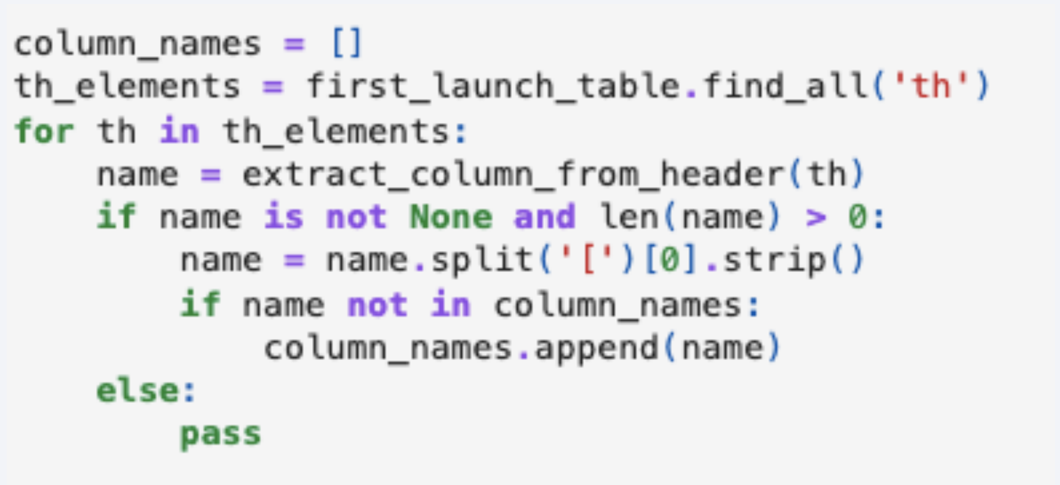
Data Collection – Data Scraping

3. Find all the tables:



```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
```

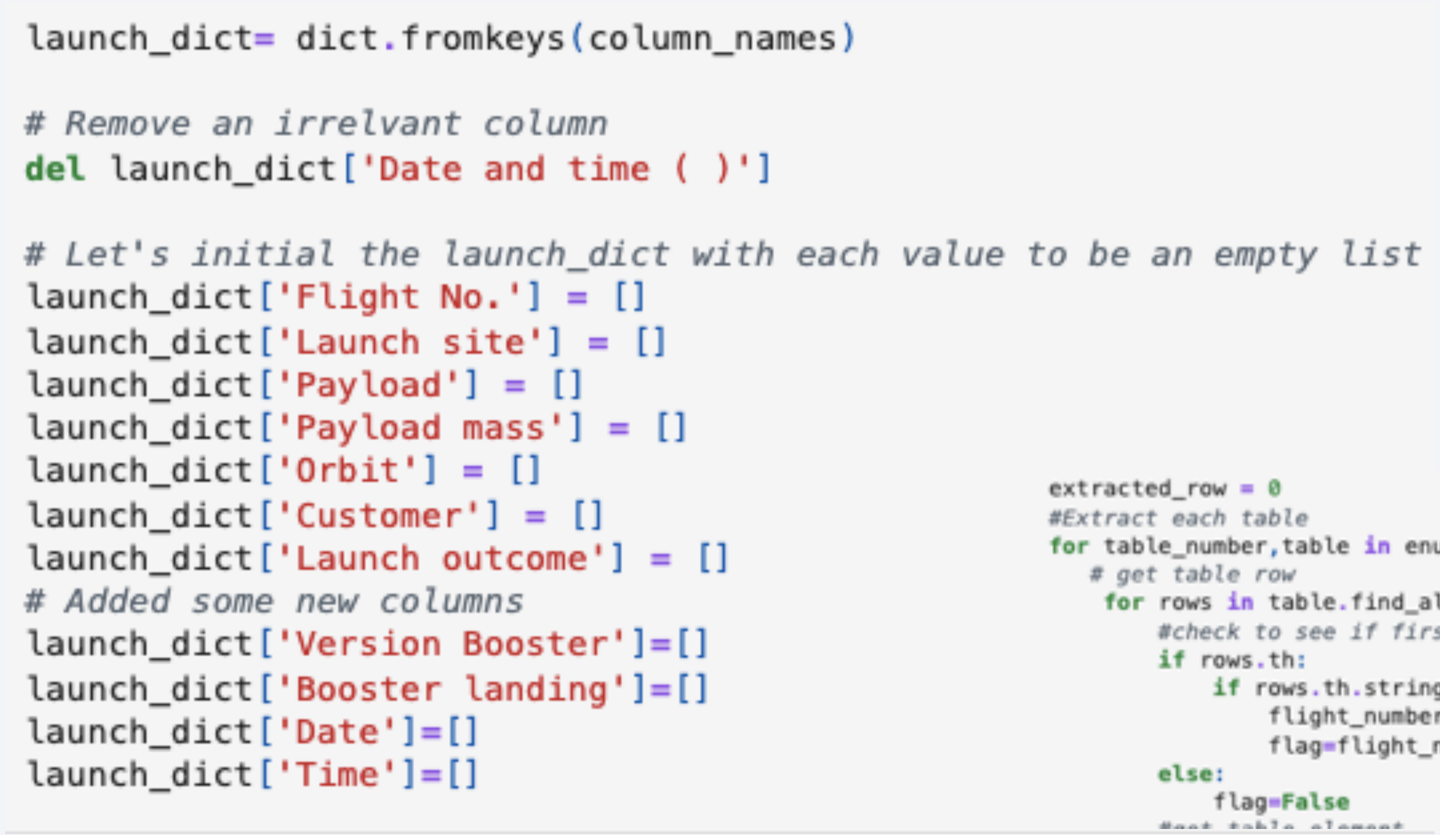
4. Get column names:



```
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:
        name = name.split('[')[0].strip()
        if name not in column_names:
            column_names.append(name)
    else:
        pass
```

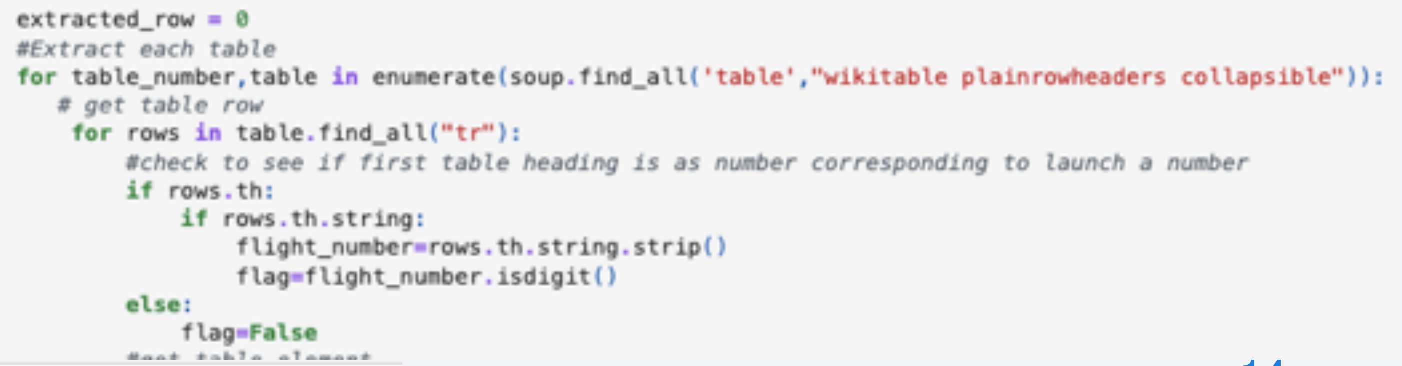
Data Collection – Data Scraping

5. Create a dictionary:



```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

6. Add the data to the keys:



```
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
```

Data Collection – Data Scraping

7. Create a Dataframe from dictionary

```
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
```

8. Export to file:

```
df.to_csv('spacex_web_scraped.csv', index=False)
```

Data Wrangling

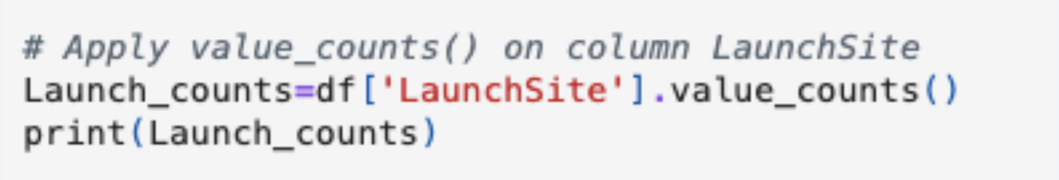
Data wrangling included calculating the number of launches on each site, the number and occurrence of each orbit, the number and occurrence of mission outcome of the orbits, and reate a landing outcome label from Outcome column

labs-jupyter-spacex-Data wrangling.ipynb

https://github.com/elamiljkovic/DataScienceCapstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

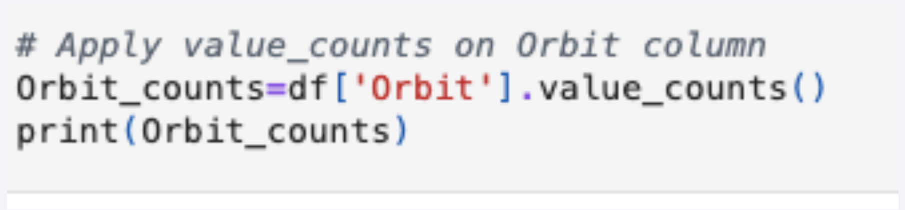
Data Wrangling

1. Calculate the launch site:



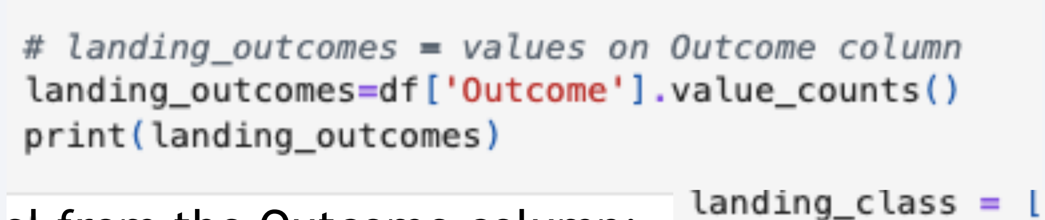
```
# Apply value_counts() on column LaunchSite
Launch_counts=df['LaunchSite'].value_counts()
print(Launch_counts)
```

2. Calculate the number and occurrence of each orbit:



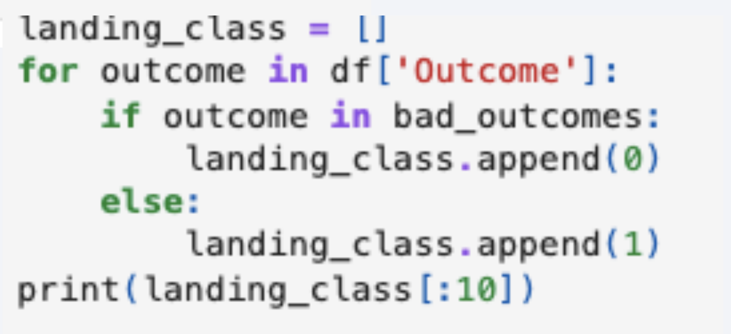
```
# Apply value_counts on Orbit column
Orbit_counts=df['Orbit'].value_counts()
print(Orbit_counts)
```

3. Calculate the number and occurrence of mission outcomes of the orbits:



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

4. Create a landing outcome label from the Outcome column:



```
landing_class = []
for outcome in df['Outcome']:
    if outcome in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
print(landing_class[:10])
```

EDA with Data Visualization

- Scatter, bar, and line graphs were created
- The first scatter graph visualized the relationship between flight number and launch site. The second scatter graph visualized the relationship between payload mass and launch site. The third scatter graph visualized the relationship between flight number and orbit type.The fourth scatter graph visualized the relationship between payload mass and orbit type.
- The first bar graph visualized the relationship between success rate of each orbit type.
- The first line graph visualized the yearly trend of launch success.

DataScienceCapstone/edadataviz (1).ipynb

https://github.com/elamiljkovic/DataScienceCapstone/blob/main/edadataviz%20(1).ipynb

EDA with SQL

- The SQL queries performed:
  - Displayed the names of the unique launch sites in the space mission
  - Displayed 5 records where launch sites begin with the string ‘CCA'
  - Displayed the total payload mass carried by boosters launched by NASA (CRS)
  - Displayed average payload mass carried by booster version F9 v1.1
  - Listed the date when the first successful landing outcome in ground pad was achieved
  - Listed the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
  - Listed the total number of successful and failure mission outcomes
  - Listed all the booster_versions that have carried the maximum payload mass
  - Listed the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015
  - Ranked the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

DataScienceCapstone/jupyter-labs-eda-sql-coursera_sqllite.ipynb

https://github.com/elamiljkovic/DataScienceCapstone/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

Launch sites were marked, and map objects such as markers, circles, and lines deli anted the success or failure of the launches.

Failures and successes were given 0 or 1, respectively. And the highest-success rates were delineated using color-labeled marker clusters.

DataScienceCapstone/lab_jupyter_launch_site_location.ipynb

https://github.com/elamiljkovic/DataScienceCapstone/blob/main/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- The interactive dashboard, created with Plotly Dash, included a pie chart showing success count for all launch sites and for specific launch sites and a scatter graph showing the relationship between Outcome and Payload Mass for the different booster versions.

- DataScienceCapstone/spacex-dash-app.py
- https://github.com/elamiljkovic/DataScienceCapstone/blob/main/spacex-dash-app.py

Predictive Analysis (Classification)

1. Building the model: load the dataset into NumPy and Pandas, transform the data by splitting into test and train datasets, create parameters and algorithms to GridSearchCV and fit in a dataset

2. Evaluate the model: check the accuracy of each model type, get tuned hyperparameters for each algorithm, and plot the confusion matrix

3. Improving the model: Use the feature engineering and algorithm tuning to improve the model

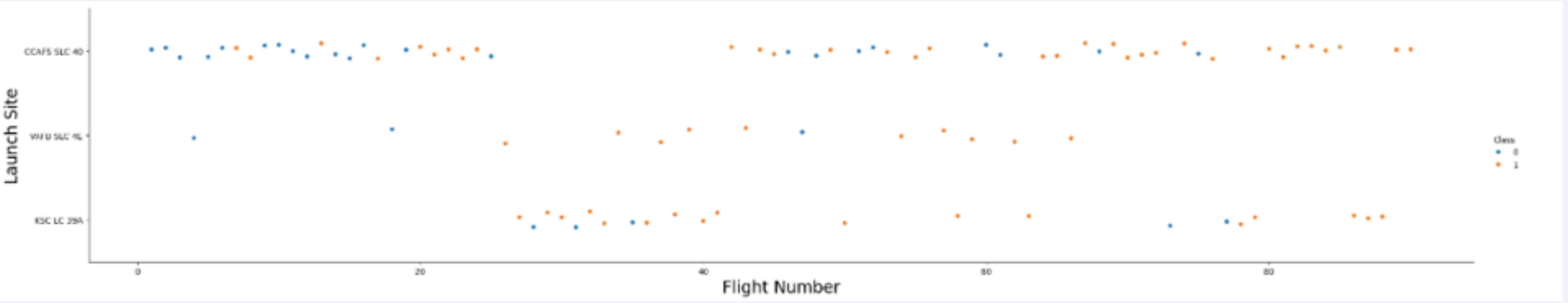
4. Find the best model: the last step is to make sure you find the model with the best accuracy score which will produce the best-performing model

DataScienceCapstone/SpaceX_Machine Learning Prediction_Part_5 (2).ipynb

Section 2

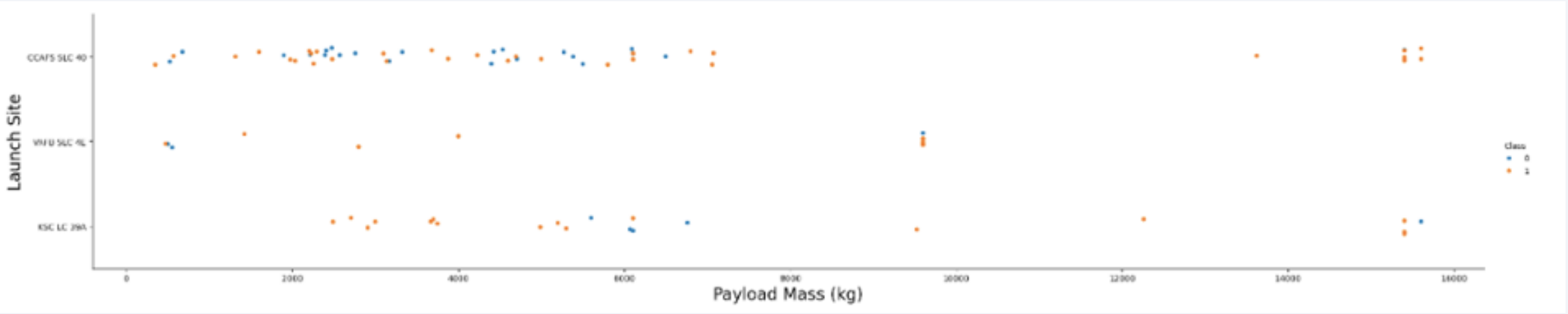
Insights drawn from EDA

Flight Number vs. Launch Site



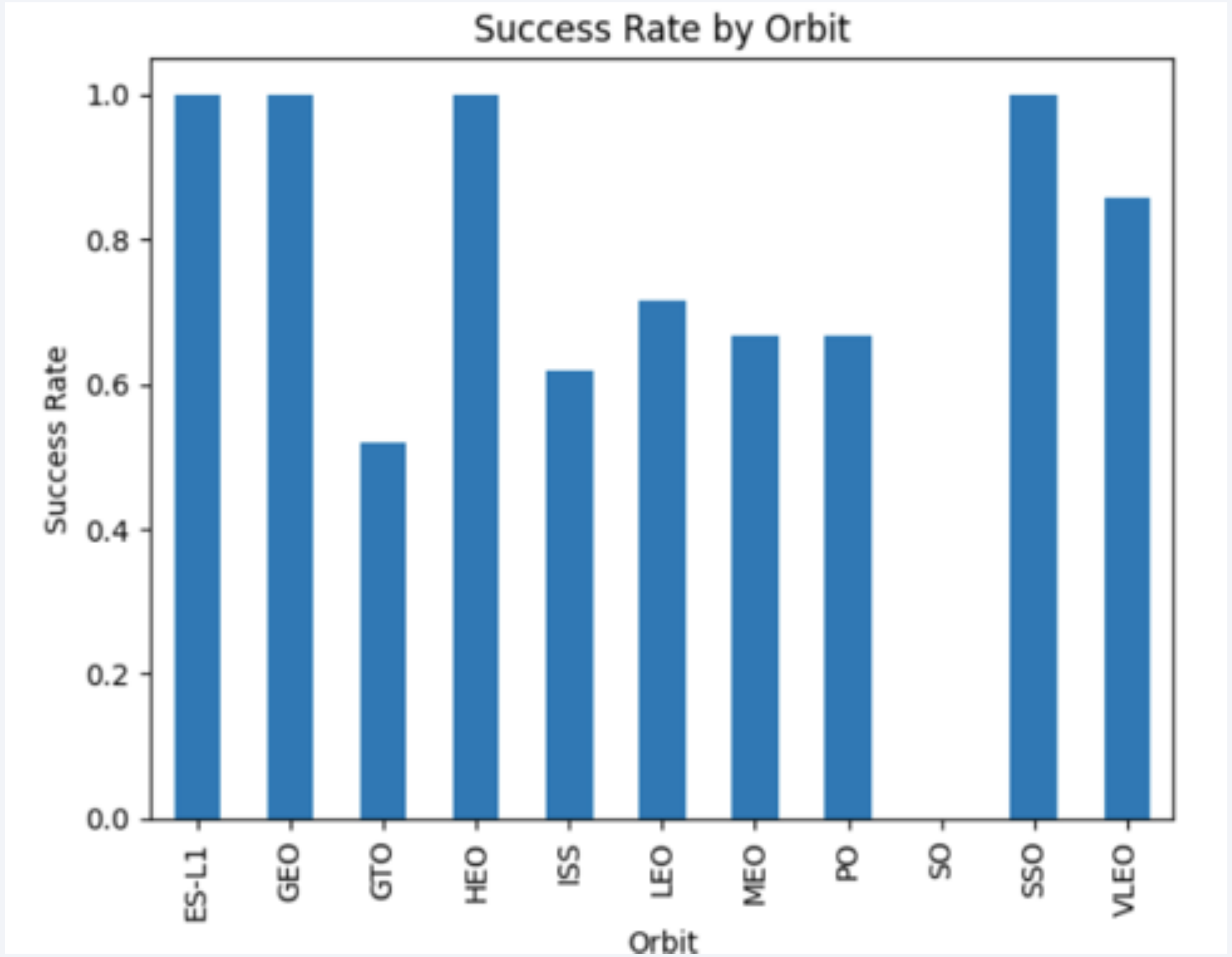
The scatter point chart shows that as the flight number increases, the first stage is more likely to land successfully.

Payload vs. Launch Site



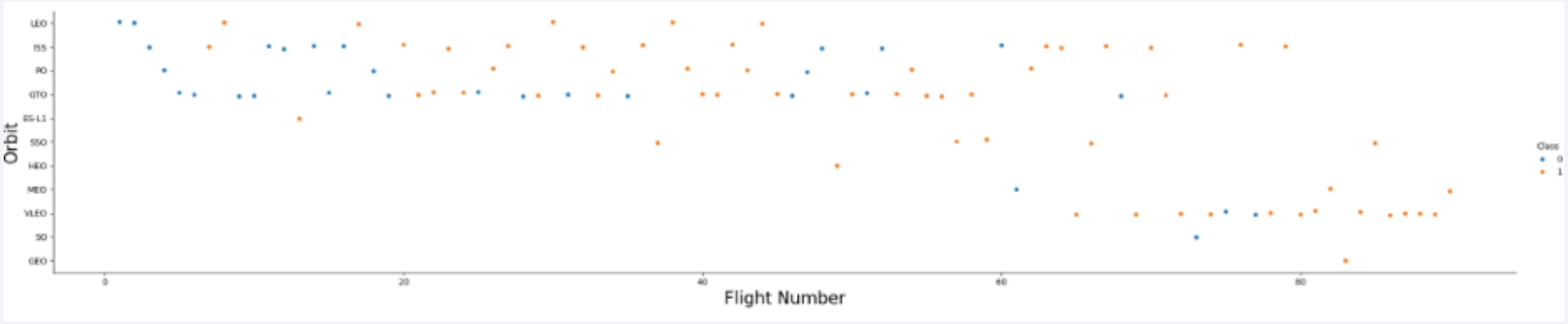
The scatter point chart shows that for the VAFB-SLC launch site there are no rockets launched for heavy payload mass(greater than 10000).

Success Rate vs. Orbit Type



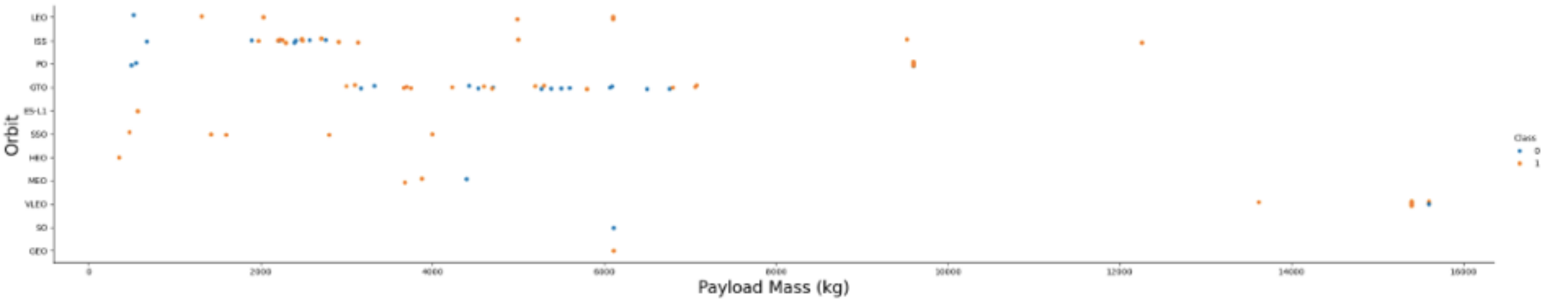
The bar chart shows that ES-L1, GEO, HEO, and SSO have the highest success rates

Flight Number vs. Orbit Type



The scatter point chart shows that in the LEO orbit, success seems to be related to the number of flights. Conversely, in the GTO orbit, there appears to be no relationship between flight number and success.

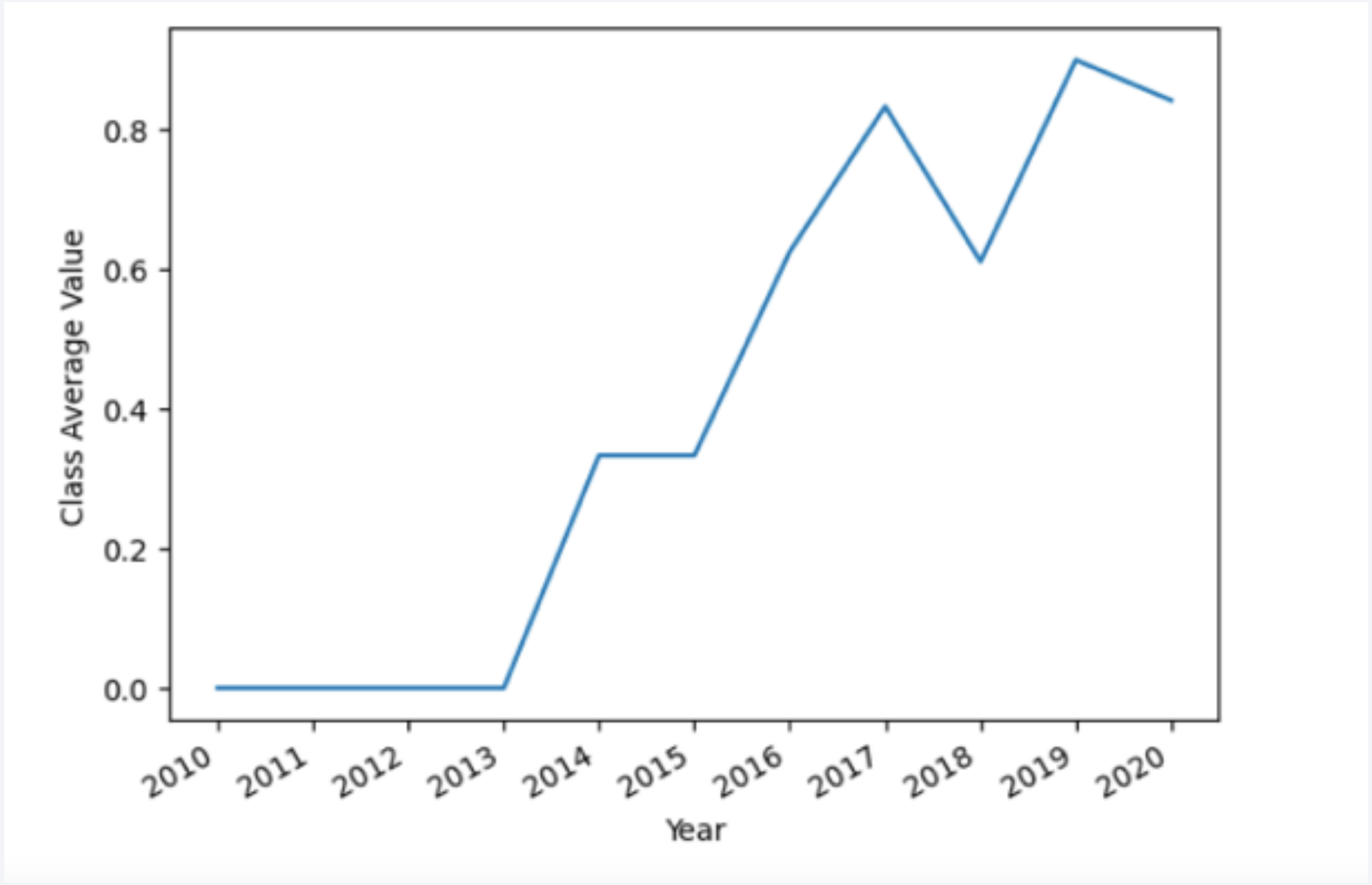
Payload vs. Orbit Type



The scatter point chart shows that with heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

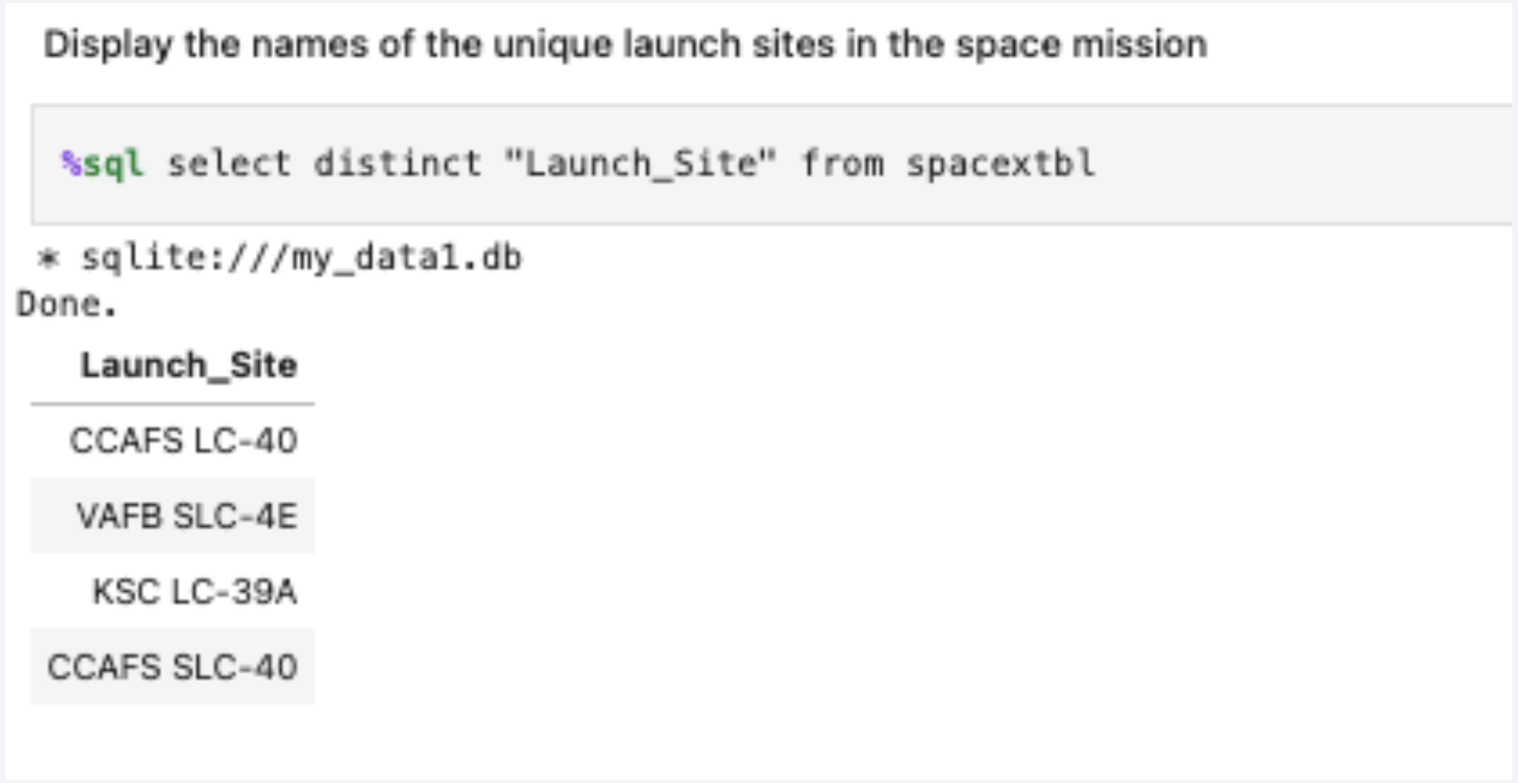
However, for GTO, it's difficult to distinguish between successful and unsuccessful landings as both outcomes are present.

Launch Success Yearly Trend



The line graph shows that the success rate since 2013 kept increasing till 2020

All Launch Site Names



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

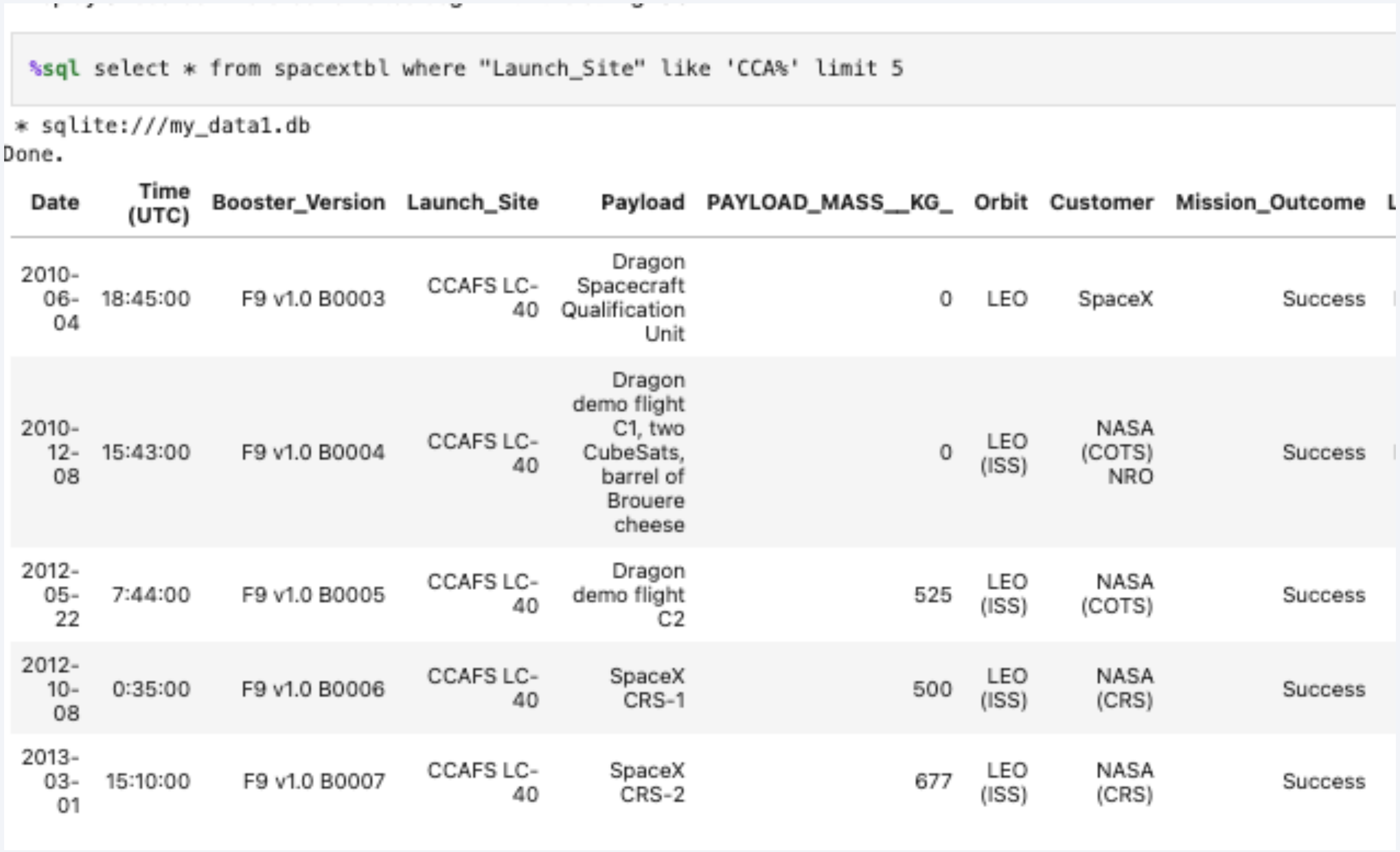
```
%sql select distinct "Launch_Site" from spacextbl
```

```
 * sqlite:///my_data1.db
Done.
```

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

The use of DISTINCT in the query leads
Launch_Site duplicates to be removed.

Launch Site Names Begin with 'CCA'



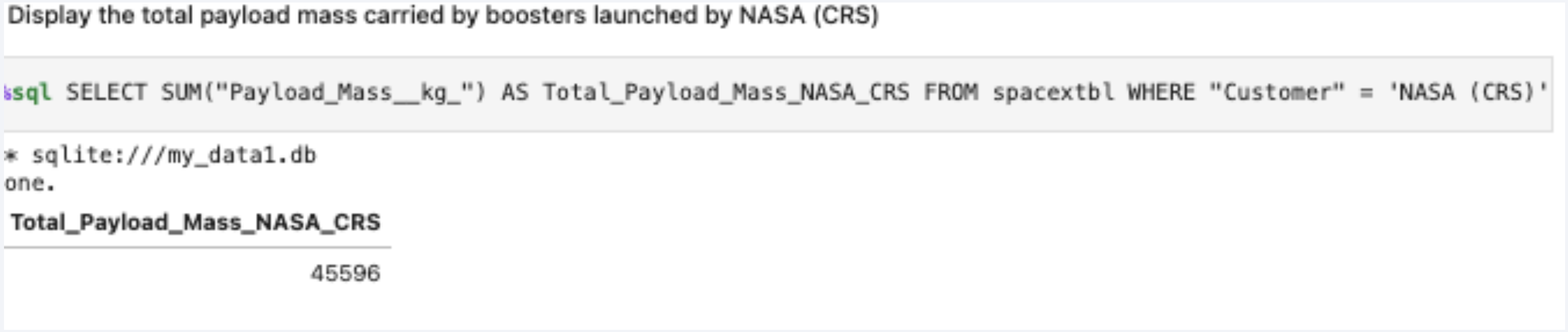
```
%sql select * from spacextbl where "Launch_Site" like 'CCA%' limit 5
```

```
* sqlite:///my_data1.db
Done.
```

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	L
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	
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	
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	

Based on these results, we can conclude that the WHERE clause is followed by LIKE clause which filters the launch sites which contains the substring CCA.
This leads to the limit being 5 and that is why we only have 5 substrings.

Total Payload Mass


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

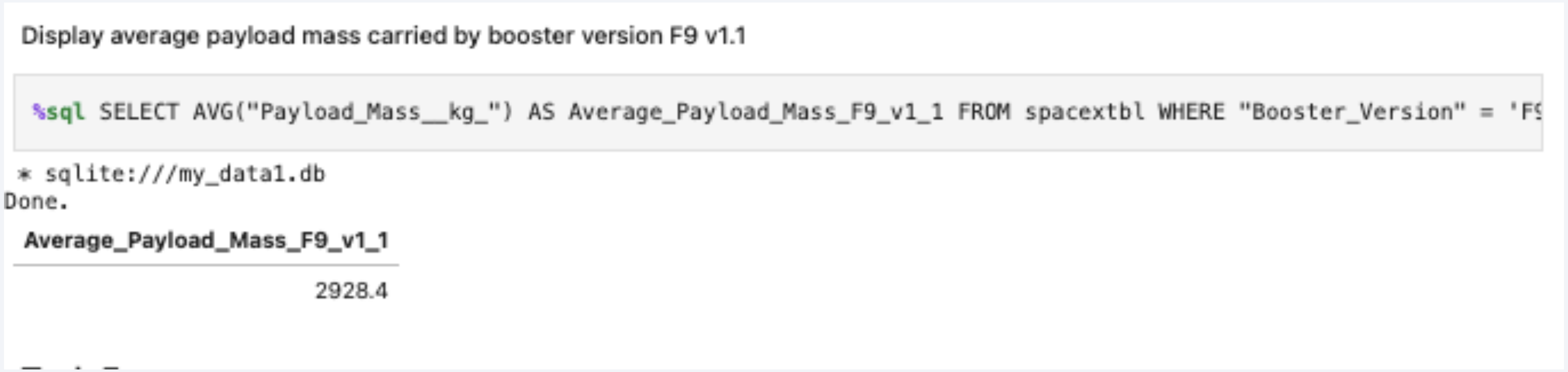
```
sql SELECT SUM("Payload_Mass__kg_") AS Total_Payload_Mass_NASA_CRS FROM spacextbl WHERE "Customer" = 'NASA (CRS)'
```

```
* sqlite:///my_data1.db
one.
```

Total_Payload_Mass_NASA_CRS
45596

Based on this query result, we can conclude that the query will return the sum of all of the payload masses for all of the customers are NASA(CRS). The result being 45,596.

Average Payload Mass by F9 v1.1


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

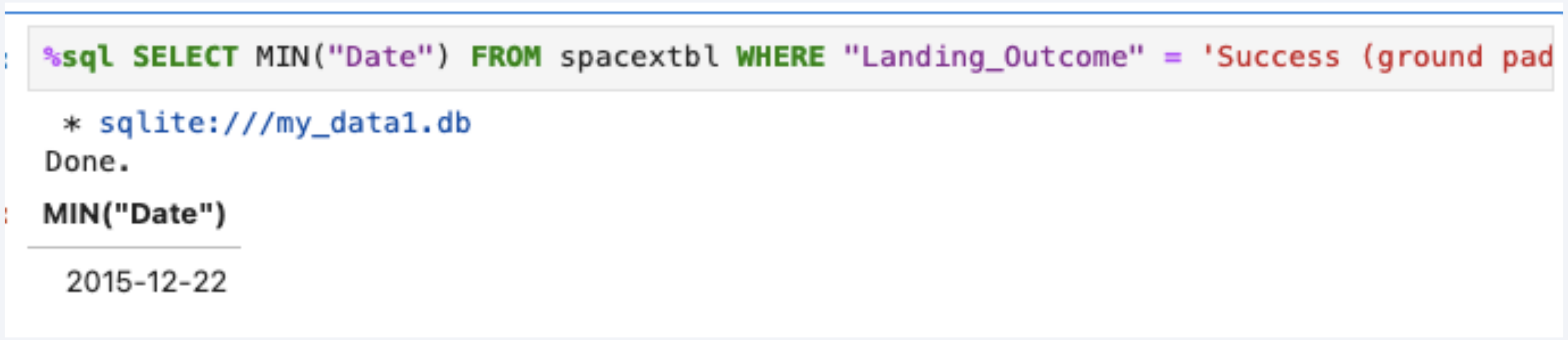
```
%sql SELECT AVG("Payload_Mass__kg_") AS Average_Payload_Mass_F9_v1_1 FROM spacextbl WHERE "Booster_Version" = 'F9
```

```
 * sqlite:///my_data1.db
Done.
```

Average_Payload_Mass_F9_v1_1
2928.4

Based on the query result, we were able to find the average payload mass carried by Booster version F9 v1.1 . The average payload mass was 2928.4

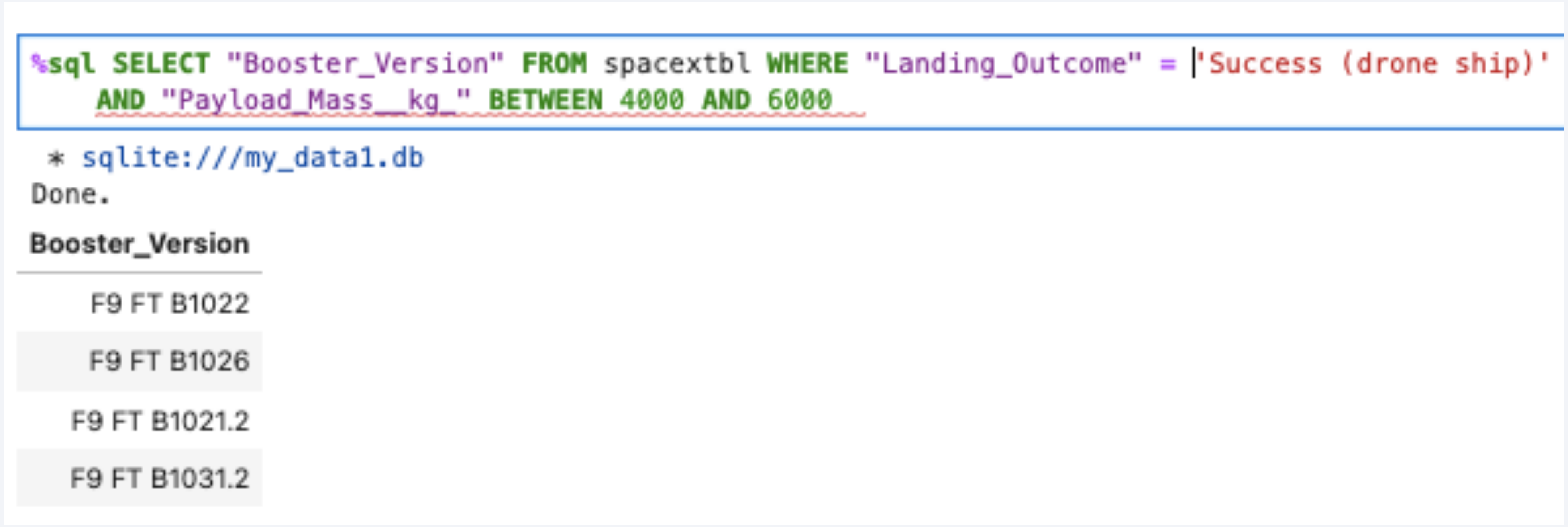
First Successful Ground Landing Date



```
%sql SELECT MIN("Date") FROM spacextbl WHERE "Landing_Outcome" = 'Success (ground pad

 * sqlite:///my_data1.db
Done.

MIN("Date")

 2015-12-22
```

Based on this query result, we found the date for the first successful ground landing. The first success successful landing date is 2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000



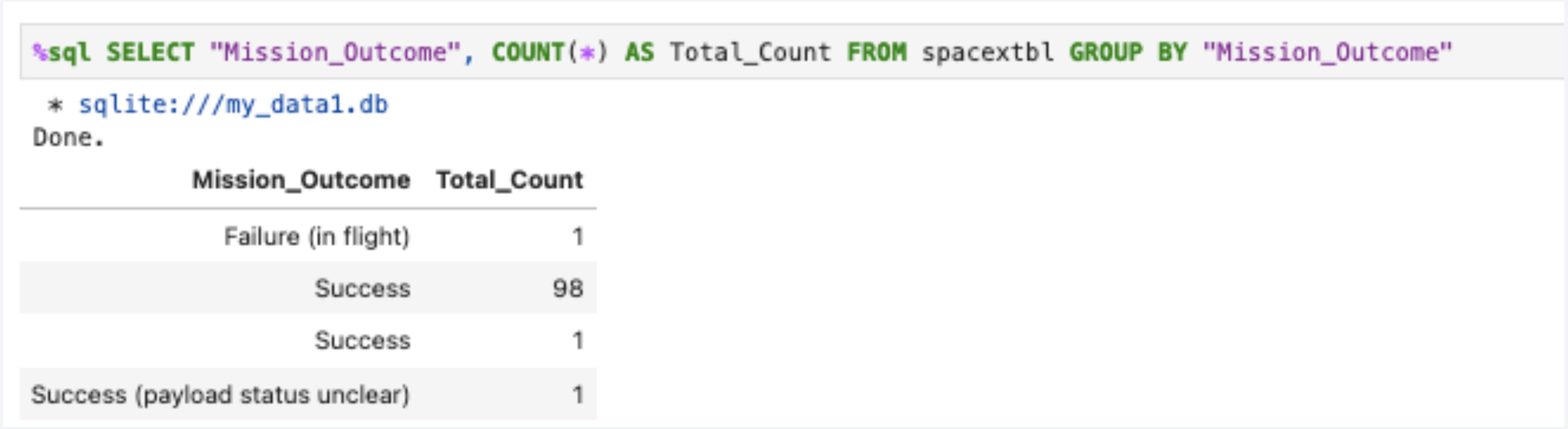
```
%sql SELECT "Booster_Version" FROM spacextbl WHERE "Landing_Outcome" = 'Success (drone ship)'
    AND "Payload_Mass__kg_" BETWEEN 4000 AND 6000
```

```
 * sqlite:///my_data1.db
Done.
```

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

This query illustrates the booster version where the landing was successful and the payload is between 4000 and 6000kg. The WHERE and AND clauses filters main objective is to filter the dataset.

Total Number of Successful and Failure Mission Outcomes



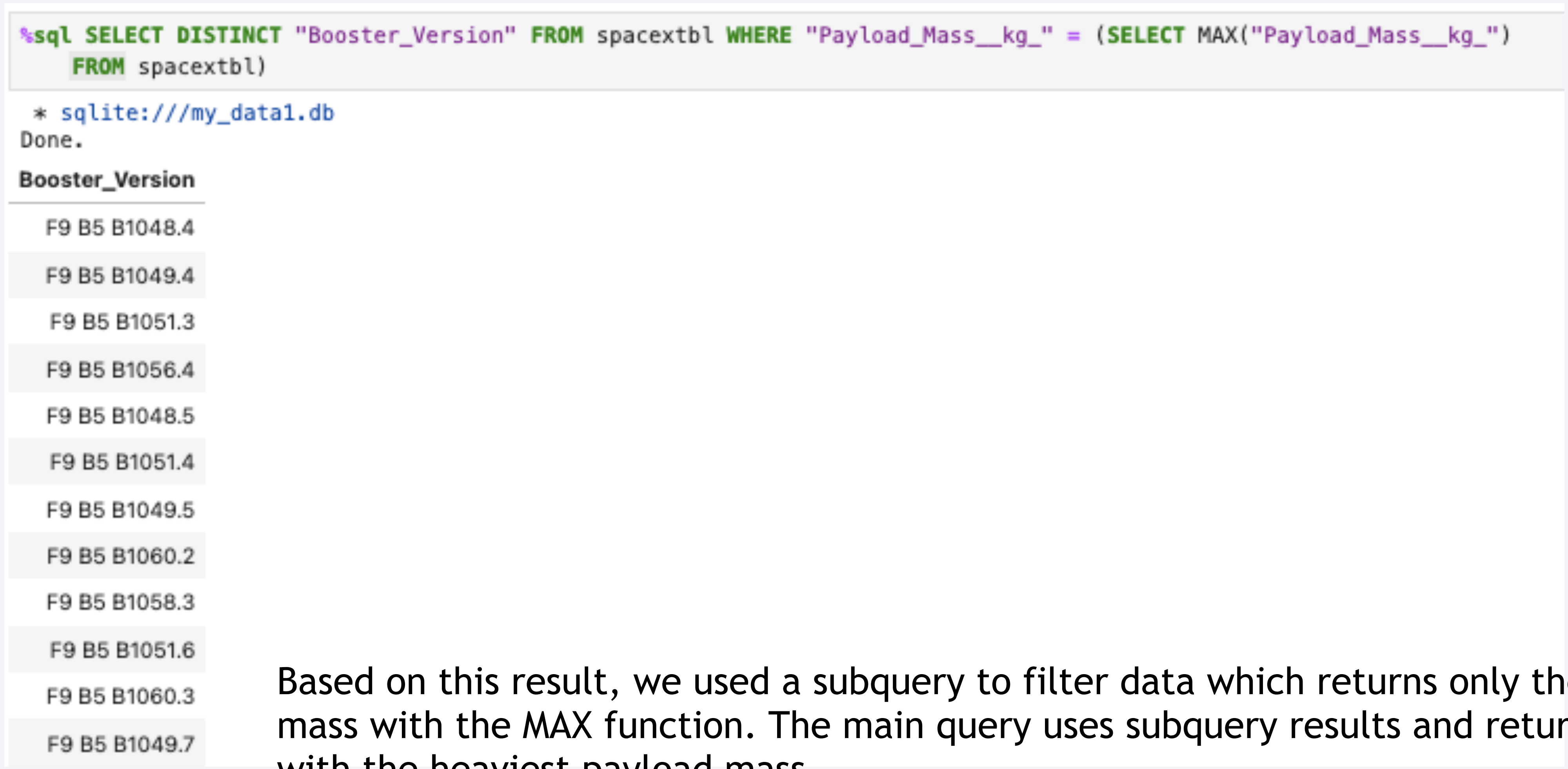
```
%sql SELECT "Mission_Outcome", COUNT(*) AS Total_Count FROM spacextbl GROUP BY "Mission_Outcome"
```

```
 * sqlite:///my_data1.db
Done.
```

Mission_Outcome	Total_Count
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Based on the first SELECT, the subqueries shows the return results. The first subquery shows the mission failing. The second subquery is the mission being successful.

Boosters Carried Maximum Payload



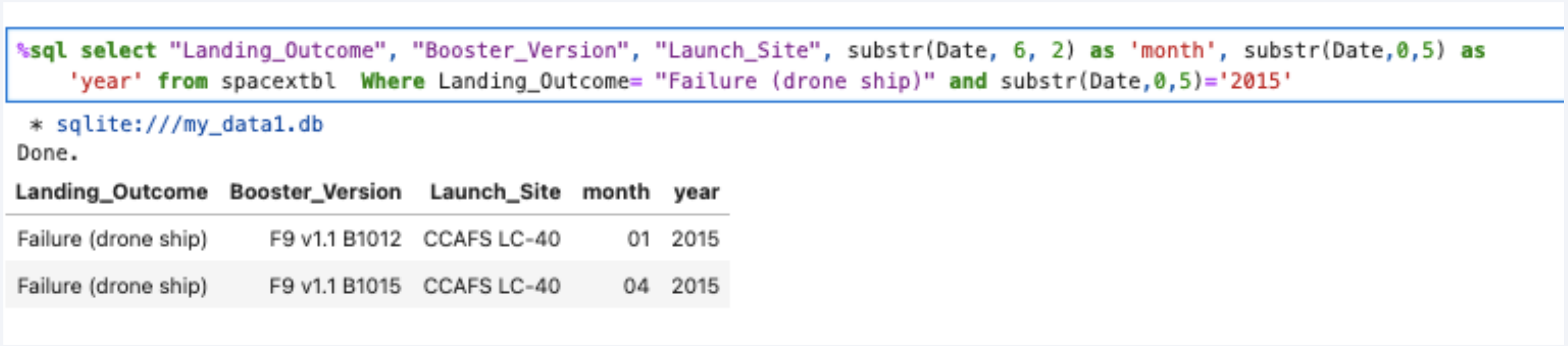
```
%sql SELECT DISTINCT "Booster_Version" FROM spacextbl WHERE "Payload_Mass__kg_" = (SELECT MAX("Payload_Mass__kg_")
    FROM spacextbl)
```

```
 * sqlite:///my_data1.db
Done.
```

Booster_Version
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

Based on this result, we used a subquery to filter data which returns only the heaviest payload mass with the MAX function. The main query uses subquery results and returns booster versions with the heaviest payload mass.

2015 Launch Records



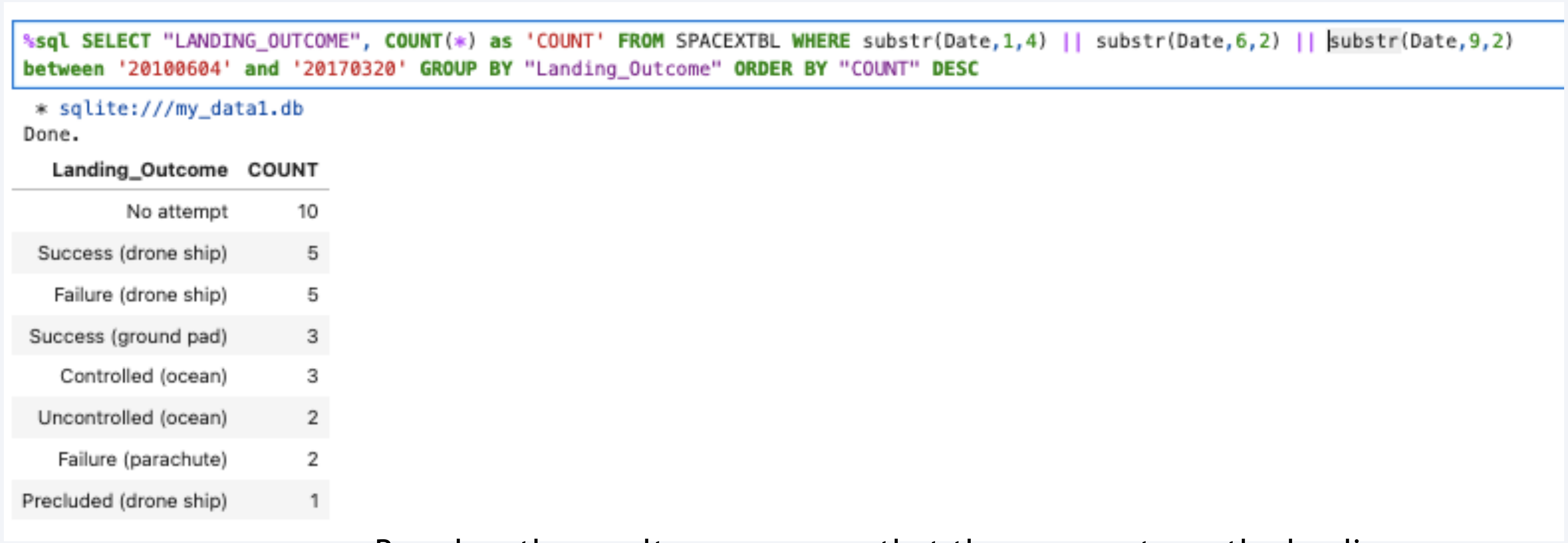
```
%sql select "Landing_Outcome", "Booster_Version", "Launch_Site", substr(Date, 6, 2) as 'month', substr(Date,0,5) as
    'year' from spacextbl  Where Landing_Outcome= "Failure (drone ship)" and substr(Date,0,5)='2015'
```

```
 * sqlite:///my_data1.db
Done.
```

Landing_Outcome	Booster_Version	Launch_Site	month	year
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40	01	2015
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40	04	2015

The query results illustrates that the month, booster version, and launch site and where the landing was unsuccessful and the landing date is 2015. the Substr function processes dates in order to take the month or the year. Substr(Date, 6,2,) shows month. Substr(DATE, 0,5,) shows year.

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



```
%sql SELECT "LANDING_OUTCOME", COUNT(*) as 'COUNT' FROM SPACEXTBL WHERE substr(Date,1,4) || substr(Date,6,2) || substr(Date,9,2)
between '20100604' and '20170320' GROUP BY "Landing_Outcome" ORDER BY "COUNT" DESC
```

```
 * sqlite:///my_data1.db
Done.
```

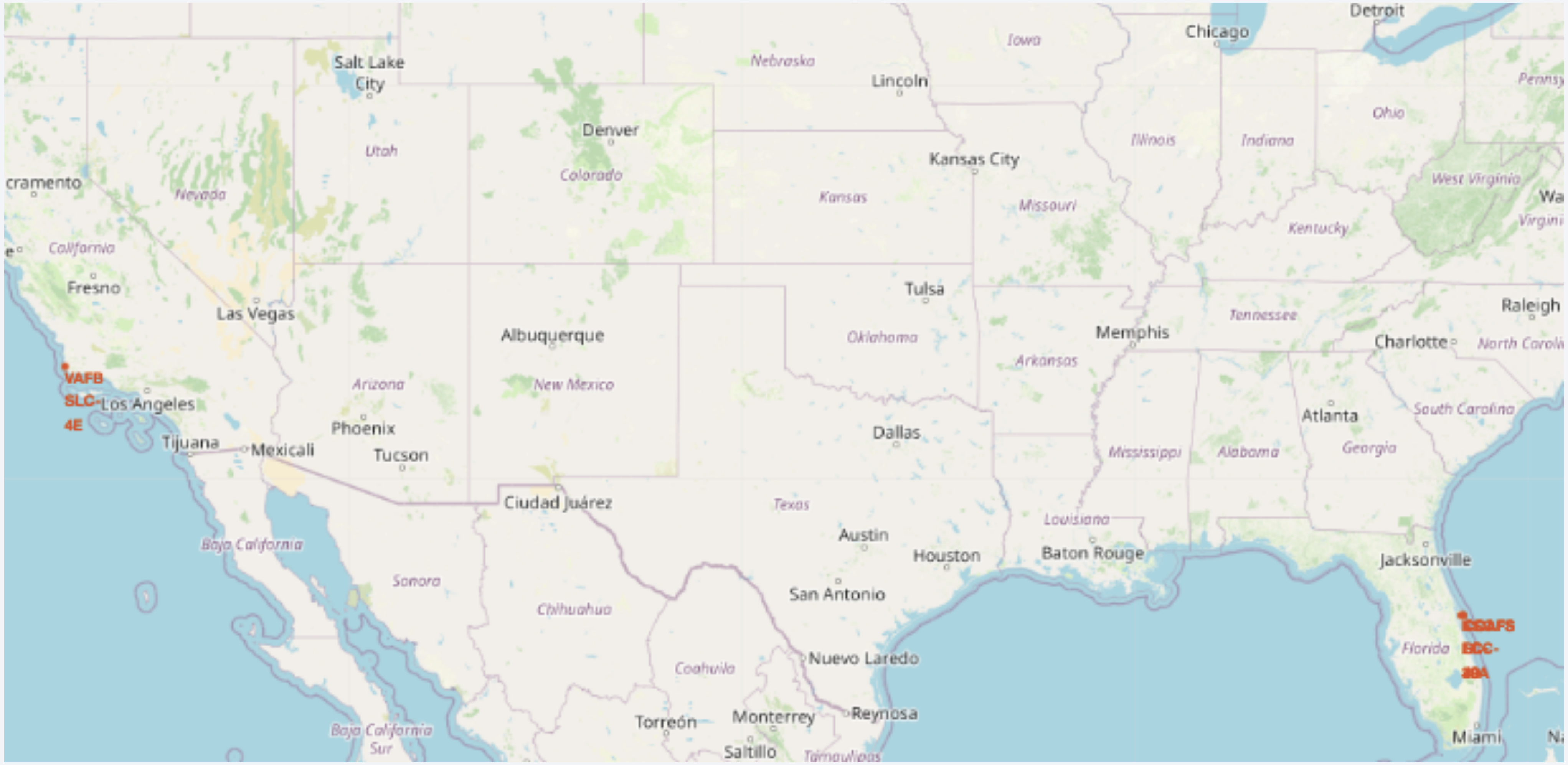
Landing_Outcome	COUNT
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Based on the results, we can see that the query returns the landing outcomes and their count where the mission is successful and the date between 2010-06-04 and 2017-03-20. The Landing_Outcomes with No attempt has the highest count and the Precluded(drone ship) has the lowest count. The number count goes down for each Landing_outcome.

Section 3

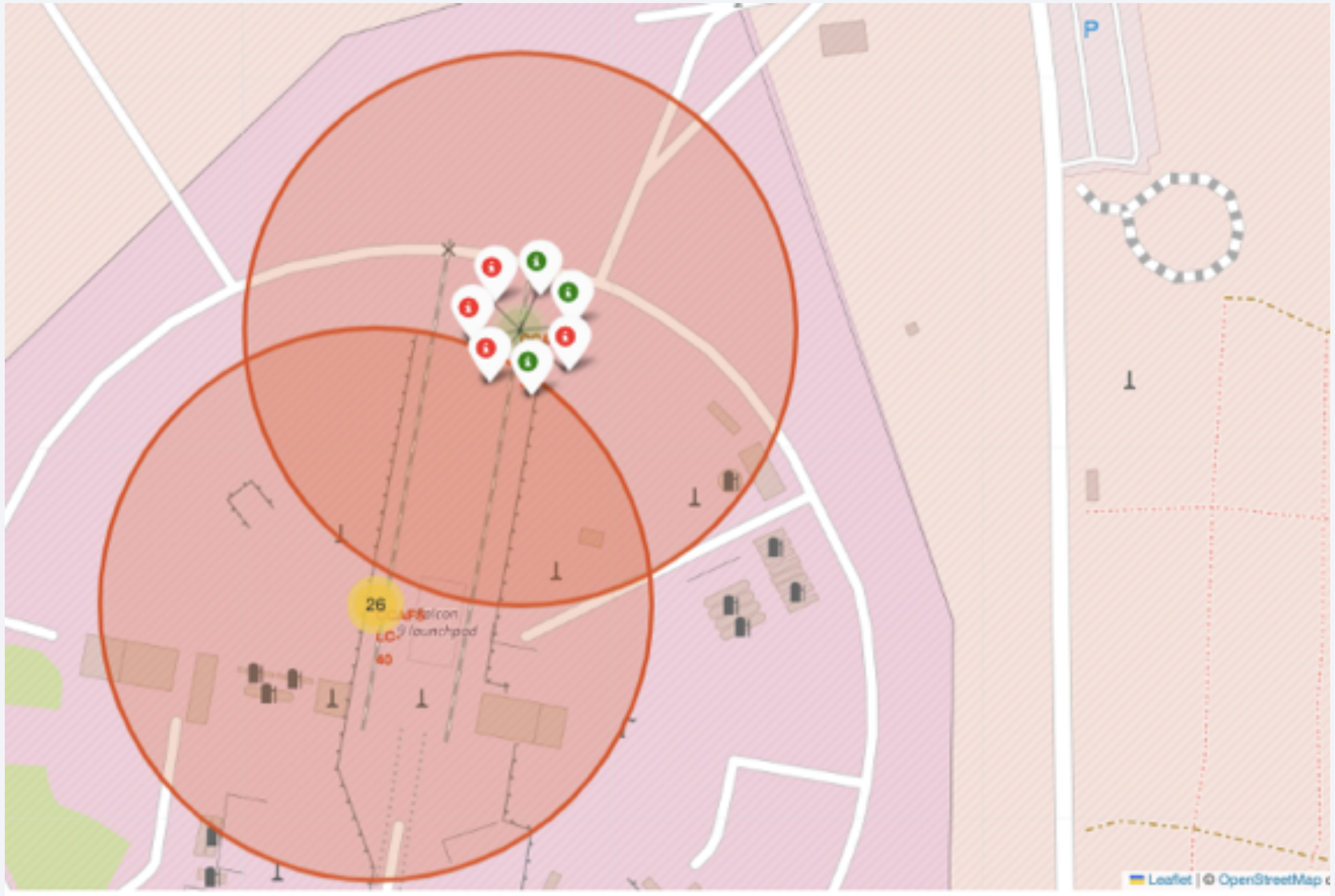
Launch Sites Proximities Analysis

All Launch Sites



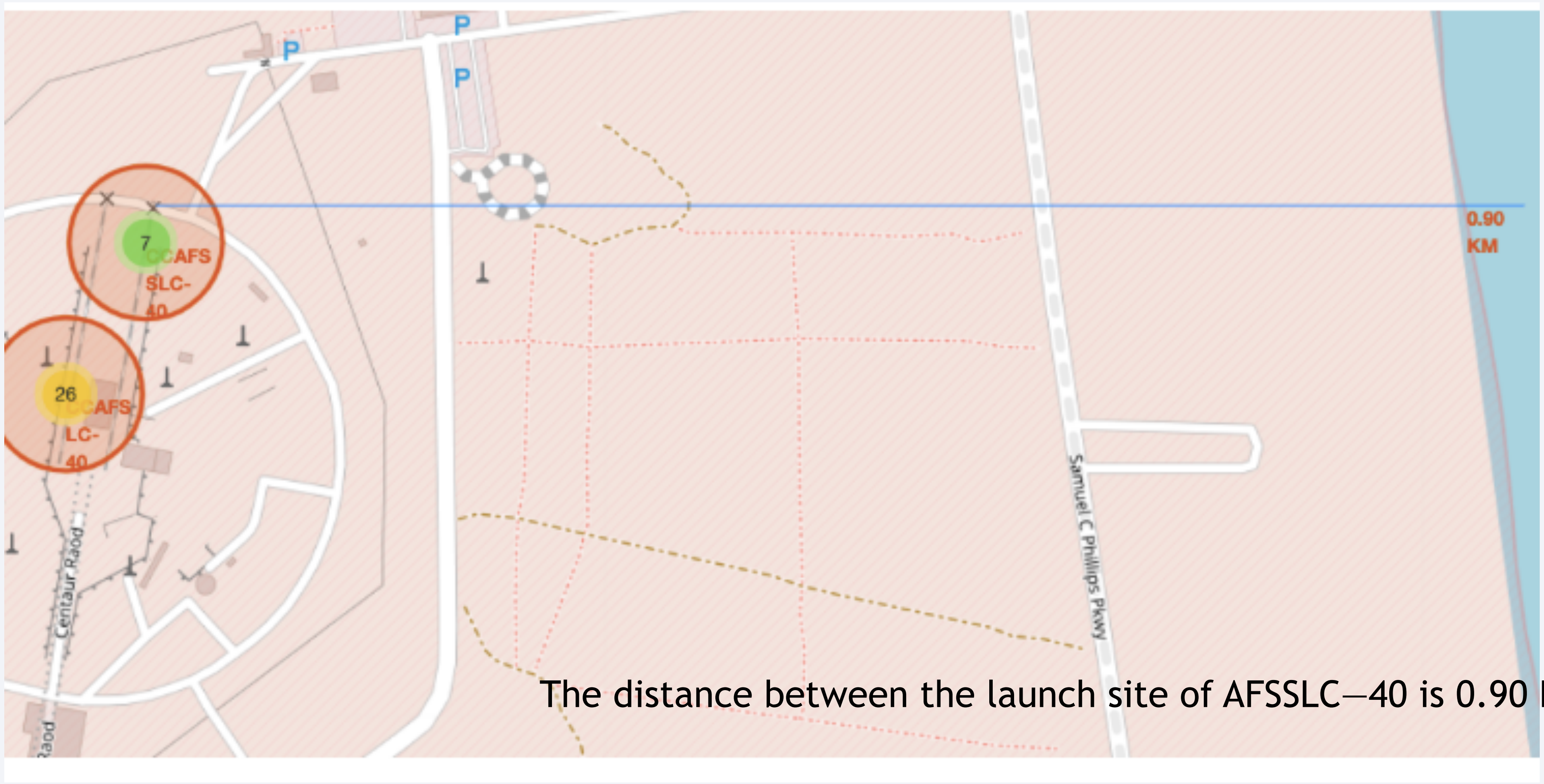
The landing sites are labeled with a mark on the map showing each sites' name

Success/Failed Launch Sites

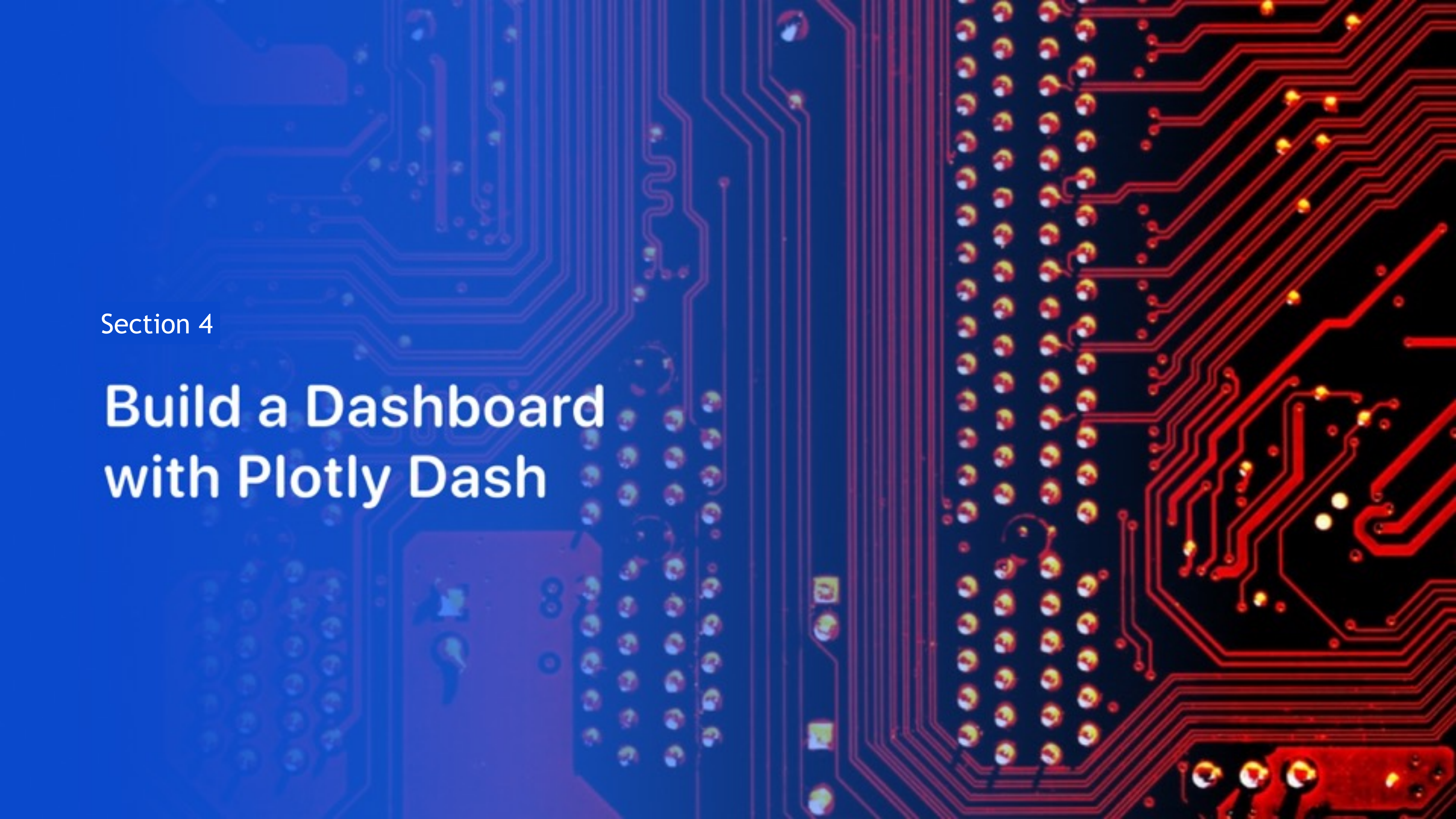


The launch sites are assembled as a group on the map. Then when you press on the
launch site you will see either a red dot or a green dot. The red dot resembles a
unsuccessful launch and a green label resembles a successful launch.

Launch Site Proximity to Railway, Highway, Coastline, with Distance Calculated and Displayed



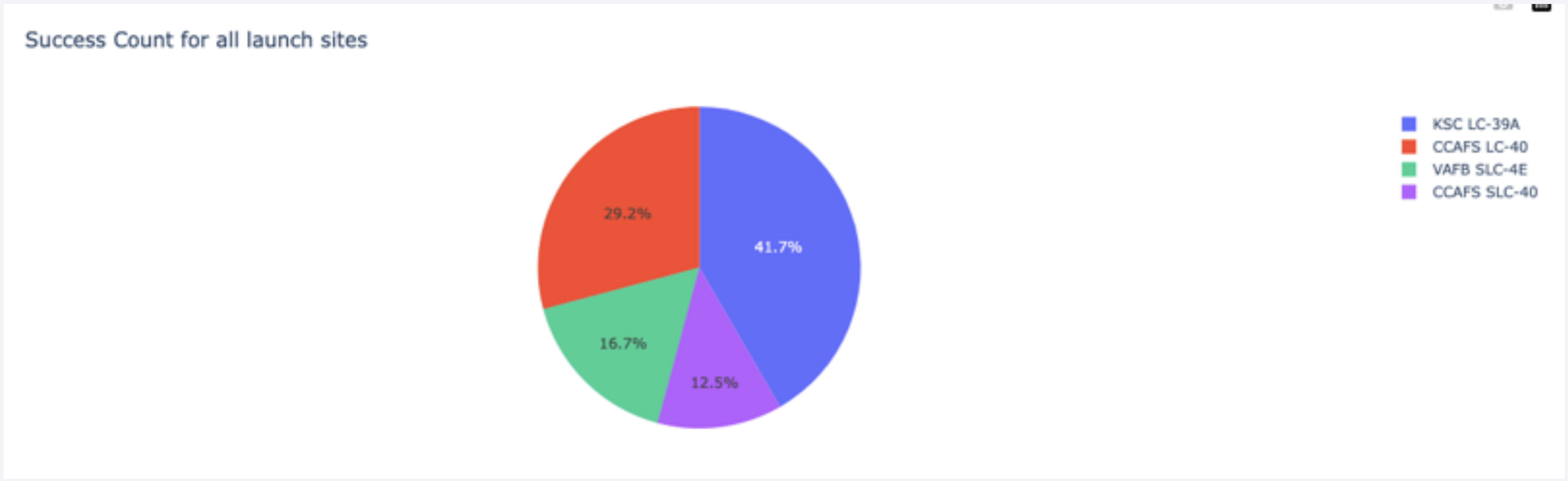
The distance between the launch site of AFSSLC—40 is 0.90 KM.



Section 4

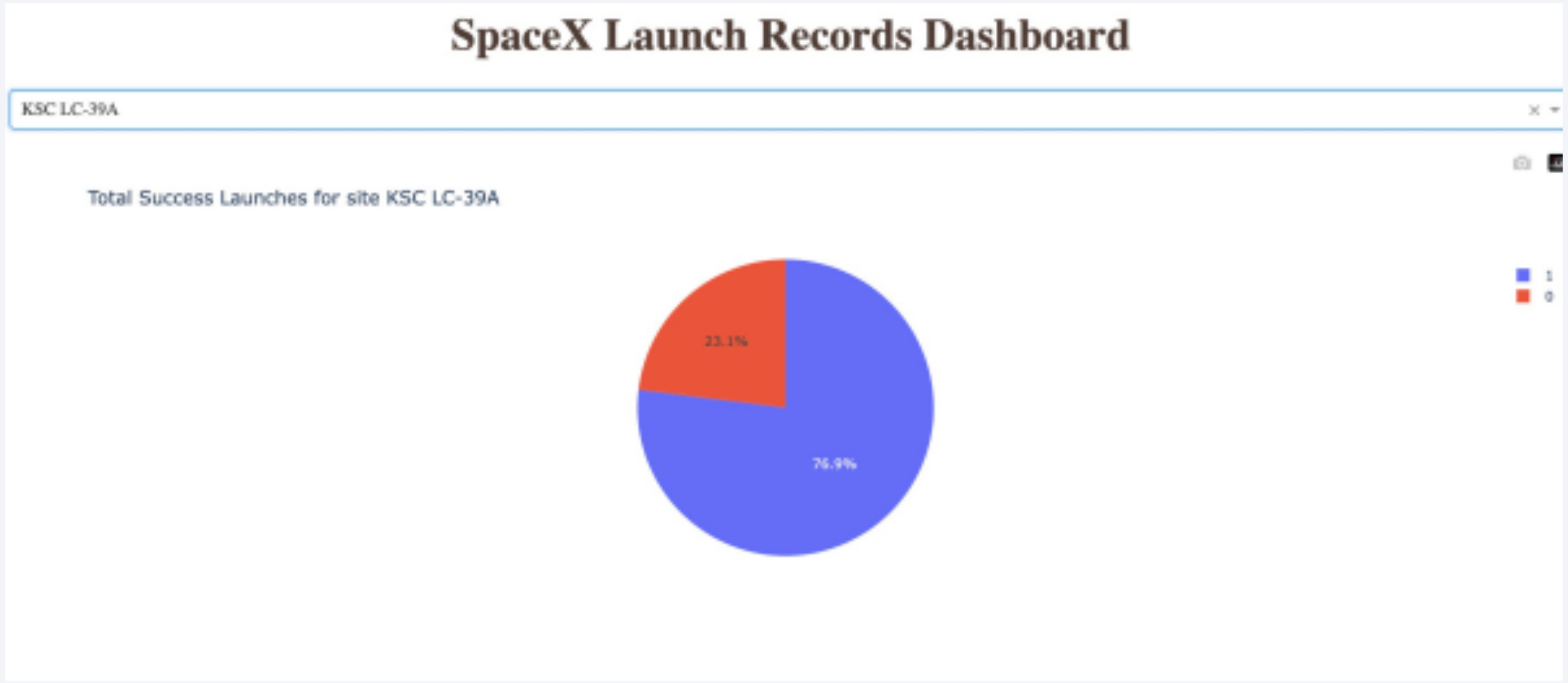
Build a Dashboard with Plotly Dash

Launch Success for All Sites



Based on the total launch success for all of the Launch Sites, the launch site with the highest success rate is KSC-LC-39A with 41.7%. The second highest is CCAFS LC-40 with 29.2% and the third highest is VAFB SLC-4E with 16.7%. The lowest success rate is CCAFS SLC-40 with 12.5%

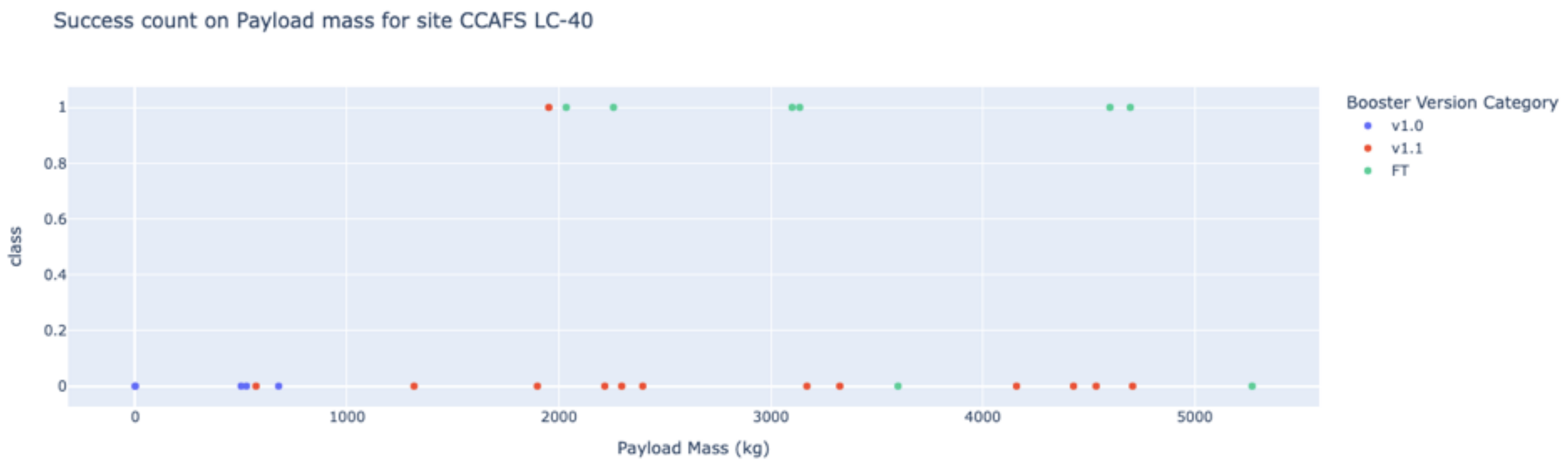
Highest Launch Site Success Ratio


SpaceX Launch Records Dashboard

KSC LC-39A

The highest success rate is KSC LC-39A with success rate is 41.7%.The successful landing rate is 76.9% and the unsuccessful landing rate is 23.1%.

Payload vs Launch Outcome Scatterplot for all Sites



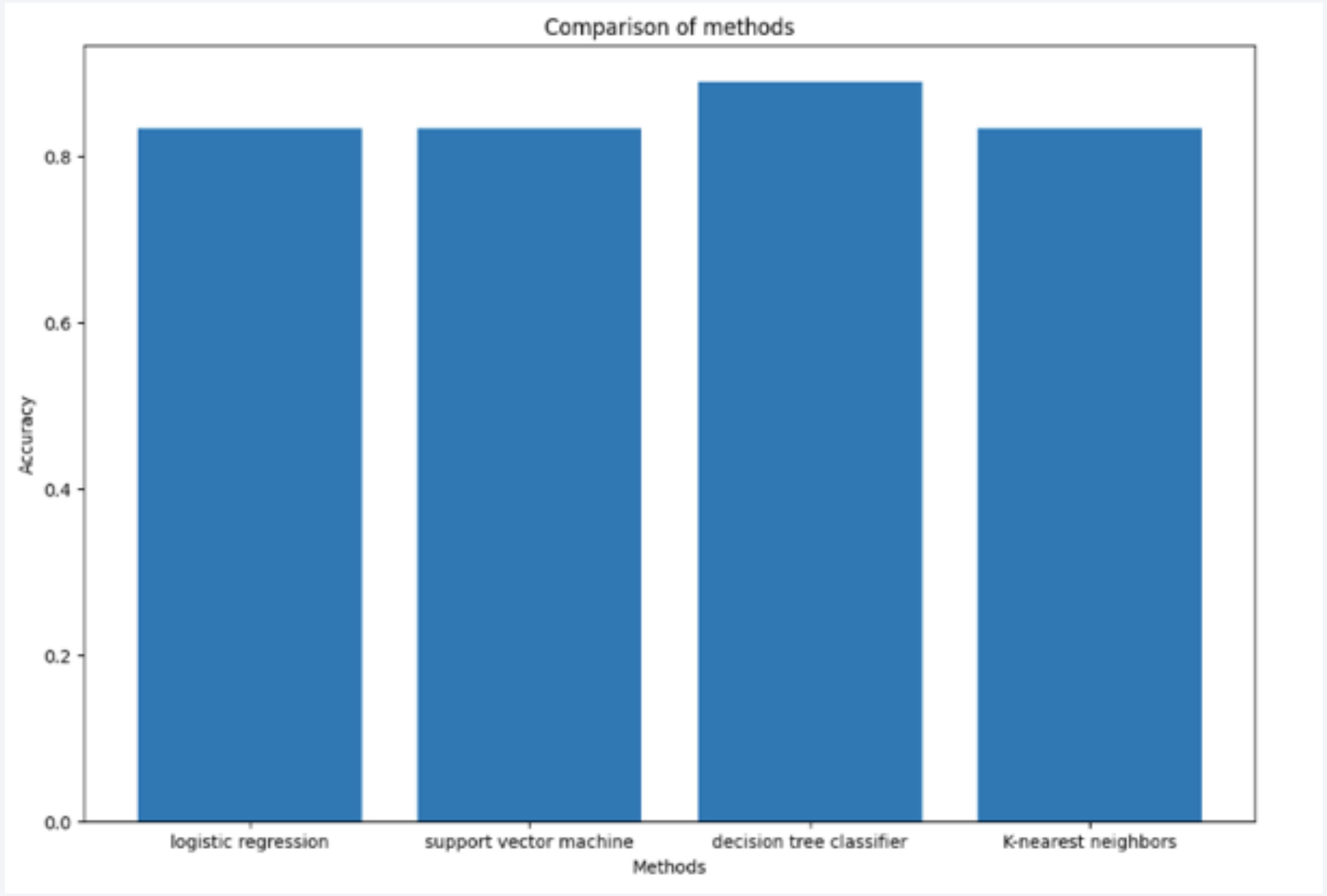
According to the scatter plot the payload range with the highest success launch is from 2,000kg and 4,000kg because it has the most plot points. The second highest payload range is from 4,000kg to 6,000kg because it has the second most plot points.

• The booster version with the highest successful launches is FT which are the green dots. FT has approximately 20 dots. The second highest is B4 which are the purple dots and they have approximately 14 dots

Section 5

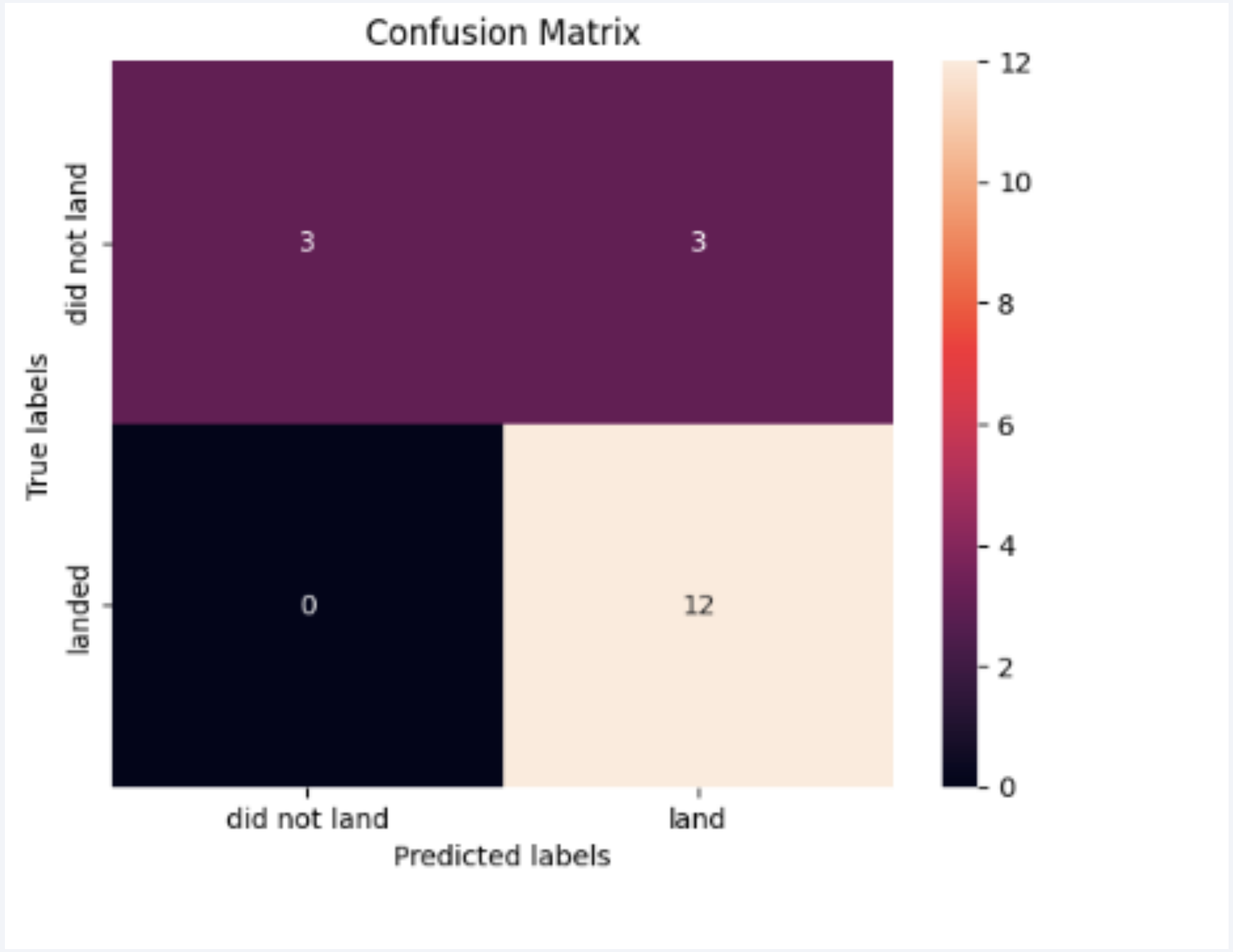
Predictive Analysis (Classification)

Classification Accuracy



The model with best accurate score is the tree classifier with a score of 0.8625. Therefore, we must use decision tree classifier as the model.

Confusion Matrix



The confusion matrix for the decision tree classifier illustrates that the classifier can interpret the different classes. However, the biggest problem is it cannot distinguish the difference between the false positives.

- For example, was it really a successful landing or was it actually an unsuccessful landing.

Conclusions

ES-L1,GEO,HEO and SSO are the orbit types with the highest success launch rates.

- Lower payloads leads to a higher performance compared to heavier payloads
- The model with best score was the decision tree classifier.
- The launch site with the highest success rate was the KSC-LC-39A with 41.7%.
- Lastly, the more experience that SpaceX has the greater the chance that it has to land successfully.

Thank you!