IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Sahana H R

18 April 2023

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 from Machine Learning Lab

Introduction

- Project background and context

Space X advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because Space X can reuse the first stage. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against space X for a rocket launch. This goal of the project is to create a machine learning pipeline to predict if the first stage will land successfully.

- Problems you want to find answers
  - What factors determine if the rocket will land successfully?
  - The interaction amongst various features that determine the success rate of a successful landing.
  - What operating conditions needs to be in place to ensure a successful landing program?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Describe how data was collected
- Perform data wrangling
  - Describe how data was processed
- 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

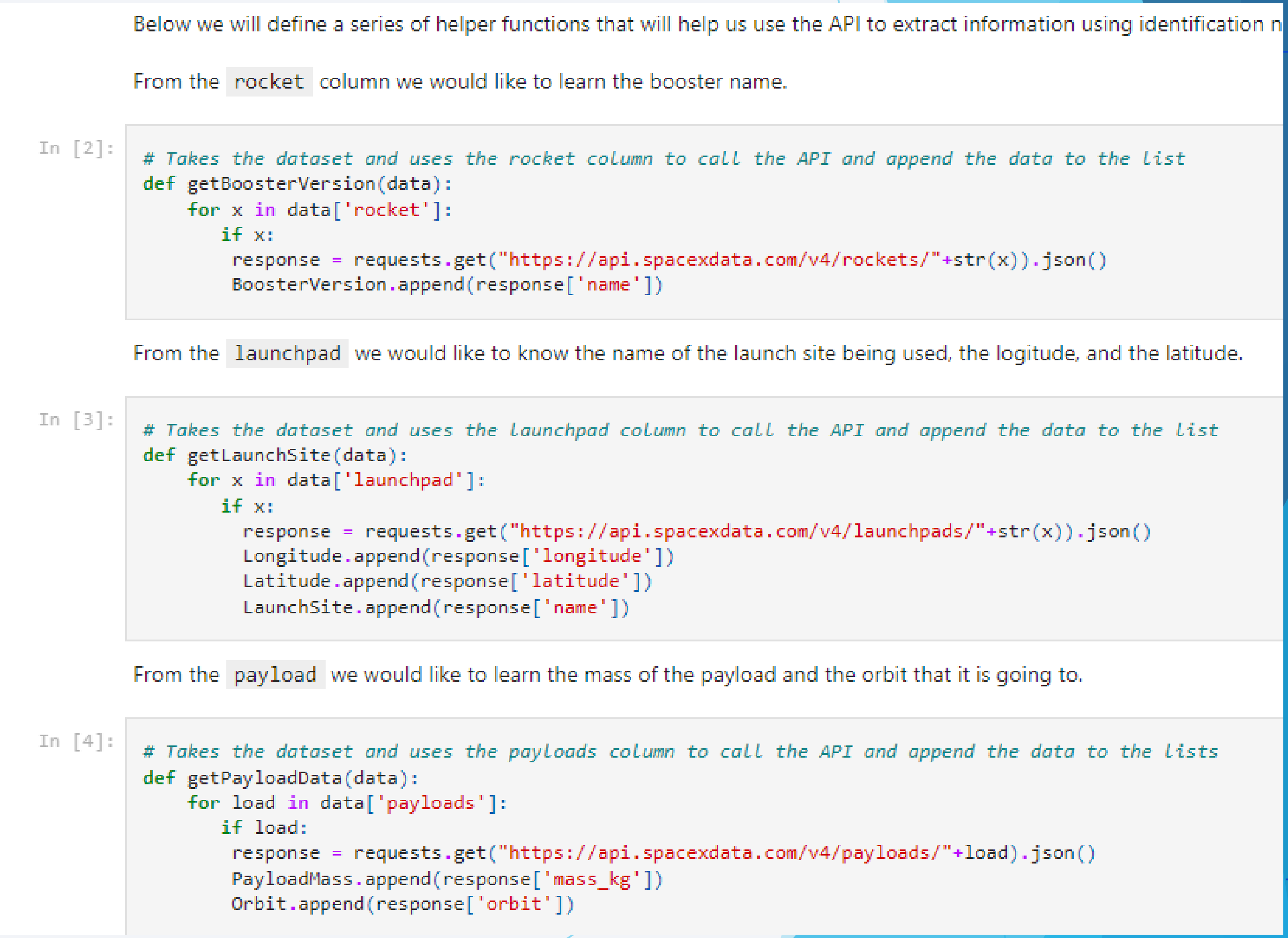
Data Collection

- The data was collected using various methods
  - Data collection was done using get request to the SpaceX API.
  - Next, we decoded the response content as a Json using .json() function call and turn it into a pandas dataframe using .json_normalize().
  - We then cleaned the data, checked for missing values and fill in missing values where necessary.
  - In addition, we performed web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.
  - The objective was to extract the launch records as HTML table, parse the table and convert it to a pandas dataframe for future analysis.

Data Collection – SpaceX API

- SpaceX API data collected using getrequest.
- Performed data cleaning and wrangling.
- GitHub URL of the completed SpaceX API calls:

https://github.com/sahana-hr/Final-course/blob/main/jupyter-labs-spacex-data-collection-api.ipynb



Below we will define a series of helper functions that will help us use the API to extract information using identification n

From the `rocket` column we would like to learn the booster name.

In [2]:

```
# Takes the dataset and uses the rocket column to call the API and append the data to the list
def getBoosterVersion(data):
    for x in data['rocket']:
       if x:
        response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
        BoosterVersion.append(response['name'])
```

From the `launchpad` we would like to know the name of the launch site being used, the logitude, and the latitude.

In [3]:

```
# Takes the dataset and uses the launchpad column to call the API and append the data to the list
def getLaunchSite(data):
    for x in data['launchpad']:
       if x:
         response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
         Longitude.append(response['longitude'])
         Latitude.append(response['latitude'])
         LaunchSite.append(response['name'])
```

From the `payload` we would like to learn the mass of the payload and the orbit that it is going to.

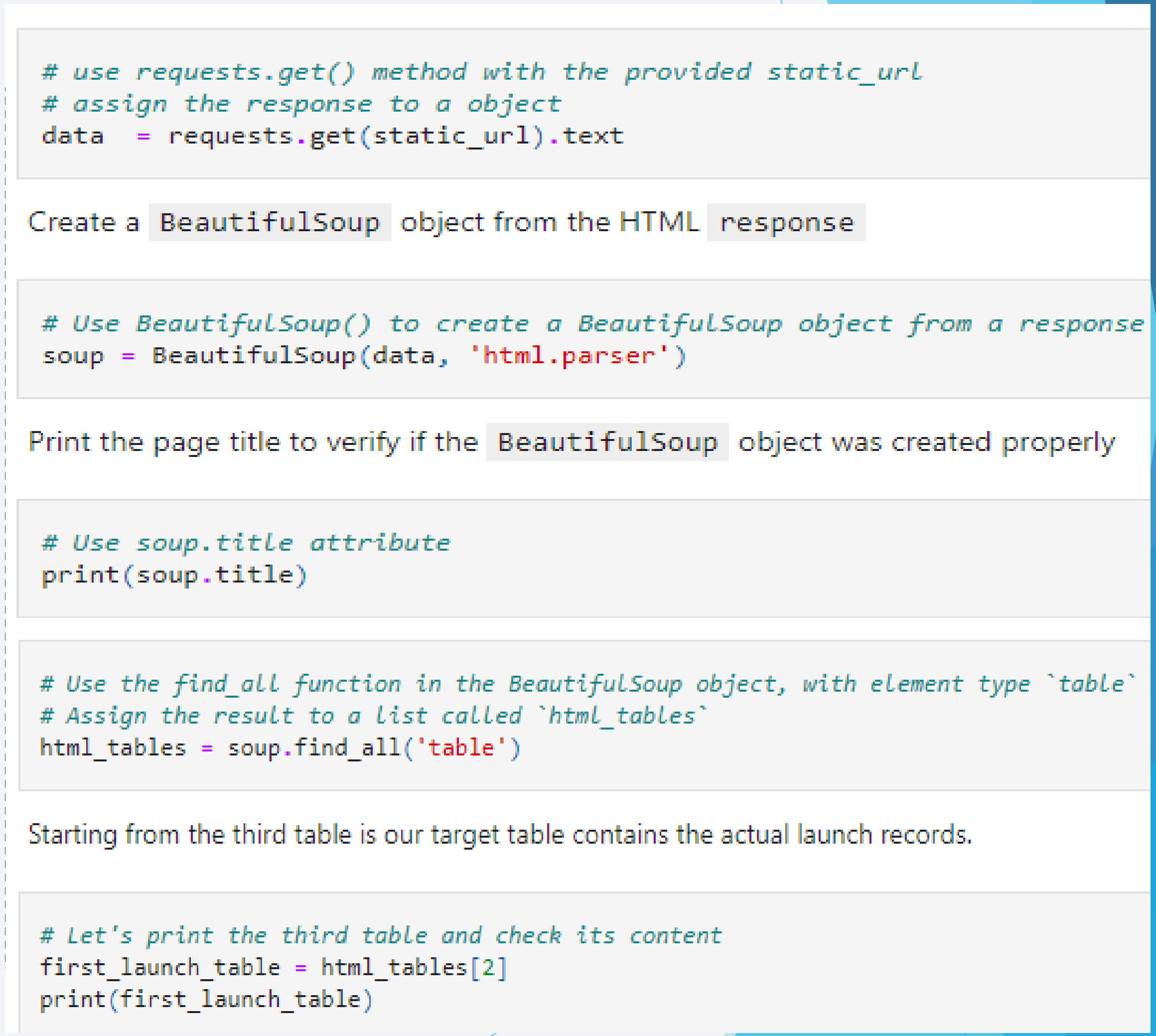
In [4]:

```
# Takes the dataset and uses the payloads column to call the API and append the data to the lists
def getPayloadData(data):
    for load in data['payloads']:
       if load:
        response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
        PayloadMass.append(response['mass_kg'])
        Orbit.append(response['orbit'])
```

Data Collection - Scraping

- Requesting the Falcon9 Launch Wiki page from its URL
- Extraction of all column names from the HTML table header
- Creation of data frame by parsing the launch HTML tables
- GitHub URL of the completed web scraping notebook:

https://github.com/sahana-hr/Final-course/blob/main/jupyter-labs-webscraping.ipynb



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

Create a `BeautifulSoup` object from the HTML `response`

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

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

```
# Use soup.title attribute
print(soup.title)
```

```
# 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')
```

Starting from the third table is our target table contains the actual launch records.

```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

Data Wrangling

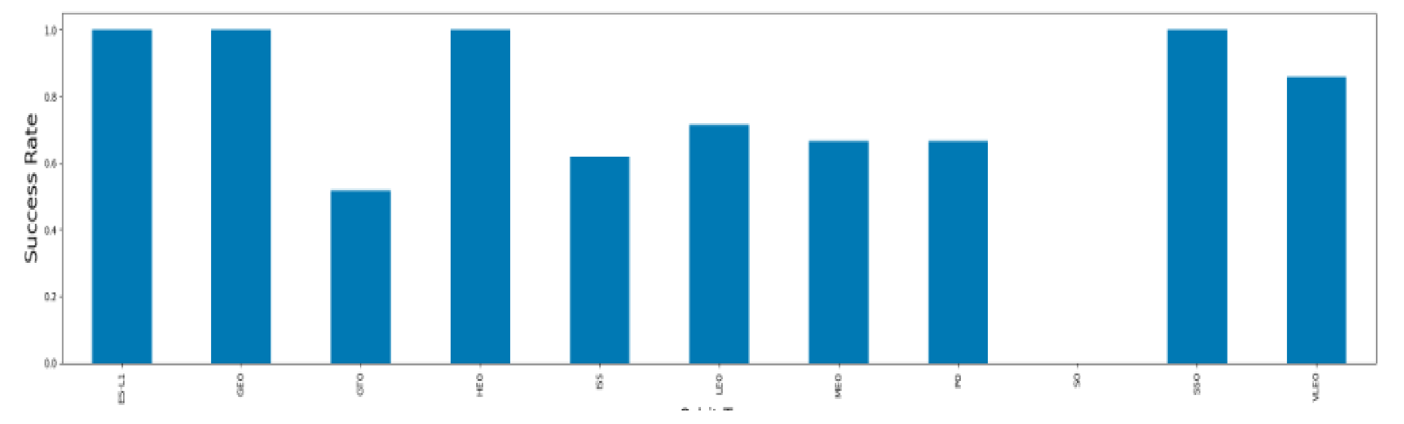
- Data Wrangling is the process of cleaning and unifying messy and complex data sets for easy access and Exploratory Data Analysis (EDA).
- We will first calculate the number of launches on each site, then calculate the number and occurrence of mission outcome per orbit type.
- We then create a landing outcome label from the outcome column. This will make it easier for further analysis, visualization, and ML. Lastly, we will export the result to a CSV.

- GitHub URL of the completed data wrangling notebook:

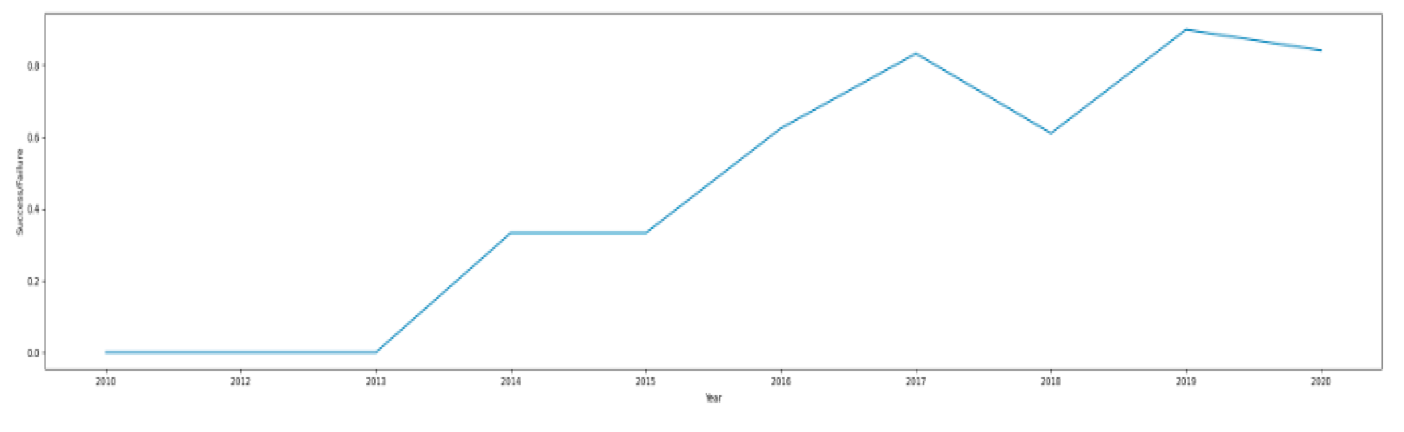
https://github.com/sahana-hr/Final-course/blob/main/labs-jupyter-spacex-data_wrangling_jupyterlite.jupyterlite.ipynb

EDA with Data Visualization

- Barchart to visualize the relationship between success rate of each orbit



- Line chart to visualize launch success yearly trend



- GitHub URL of the completed data visualization notebook:

https://github.com/sahana-hr/Final-course/blob/main/jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

EDA with SQL

SQL queries performed:

- Display the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string 'CCA'
- Display the total payload mass carried by boosters launched by NASA (CRS)
- Display average payload mass carried by booster version F9 v1.1
- List the date when the first succesful landing outcome in ground pad was acheived.
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- List the total number of successful and failure mission outcomes
- List the names of the booster_versions which have carried the maximum payload mass. Use a subquery
- List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.
- Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

GitHub URL of completed EDA with SQL notebook:

https://github.com/sahana-hr/Final-course/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- Marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- Assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.
- Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate.
- Calculated the distances between a launch site to its proximities. Answered some question for instance:
  - Are launch sites near railways, highways and coastlines.
  - Do launch sites keep certain distance away from cities.
- GitHub URL of completed interactive map with Folium map:

https://github.com/sahana-hr/Final-course/blob/main/lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

- Built an interactive dashboard with Plotly dash.
- Plotted pie charts showing the total launches by a certain sites.
- Plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.
- GitHub URL of completed Plotly Dash lab:

https://github.com/sahana-hr/Final-course/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

- Loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- Built different machine learning models and tune different hyperparameters using GridSearchCV.
- Used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- Found the best performing classification model.
- GitHub URL of completed predictive analysis lab:

https://github.com/sahana-hr/Final-course/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Results

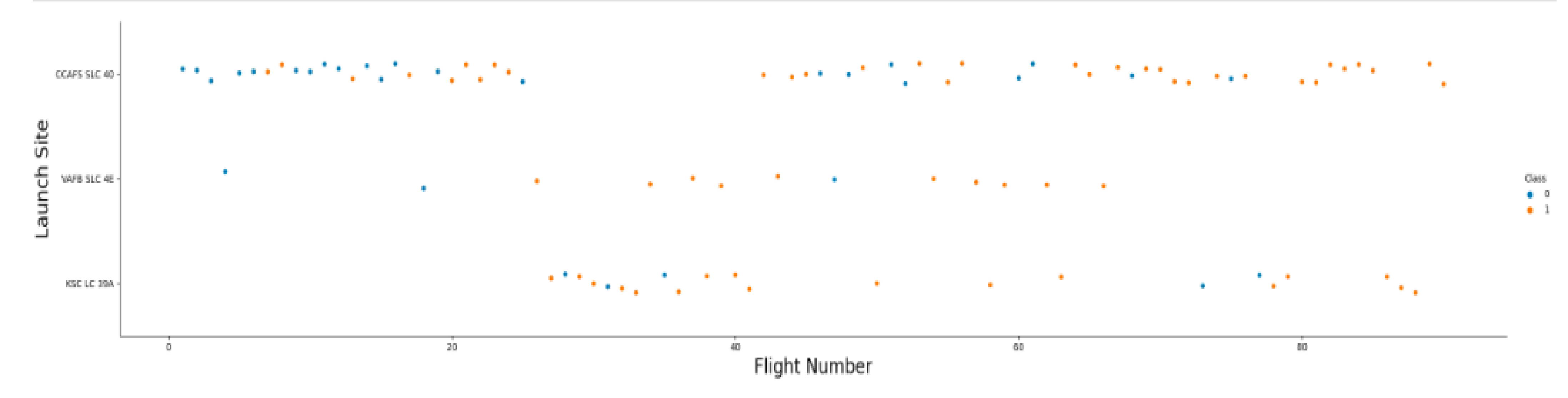
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

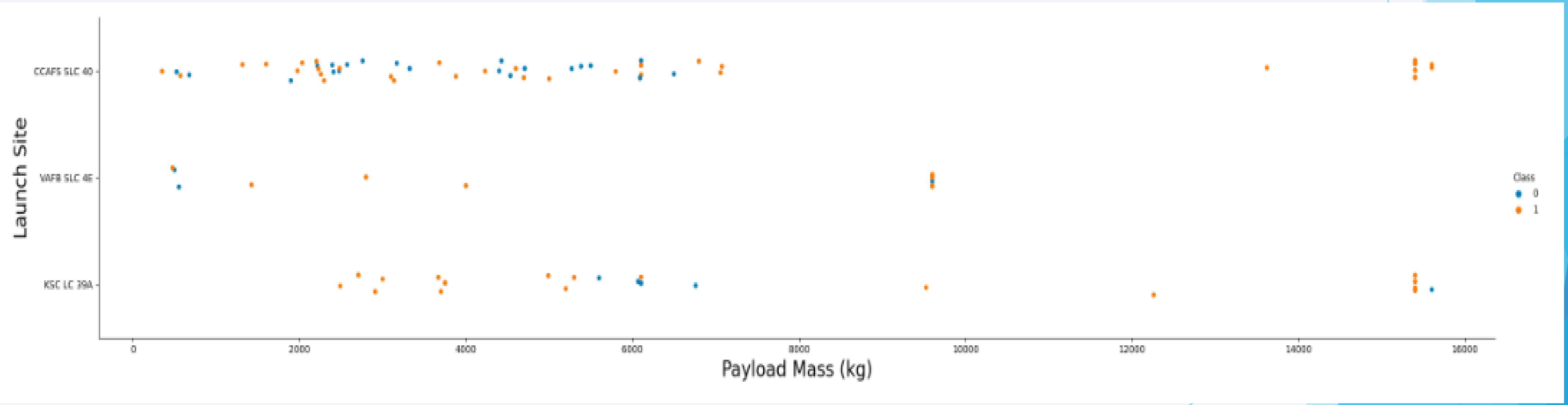
- This scatter plot shows that the larger the flights amount of the launch site, the greater the the success rate will be.
- However, site CCAFS SLC40 shows the least pattern of this.



Payload vs. Launch Site

Scatter plot of Payload vs. Launch Site

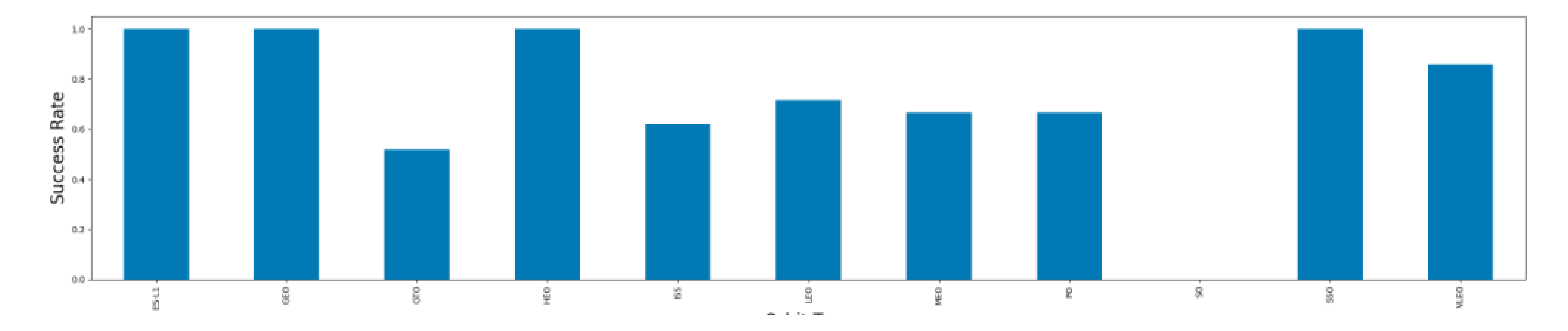
VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).



Success Rate vs. Orbit Type

Bar chart for the success rate of each orbit type

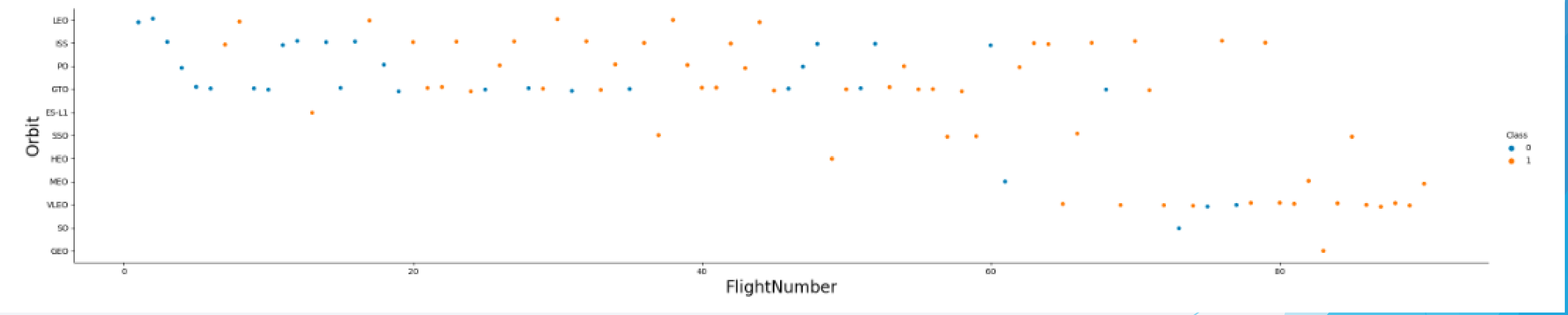
Some orbits has 100% success rate such as SSO, HEO, GEO AND ES-L1 while SO orbit produced 0% rate of success.



Flight Number vs. Orbit Type

Scatter point of Flight number vs. Orbit type

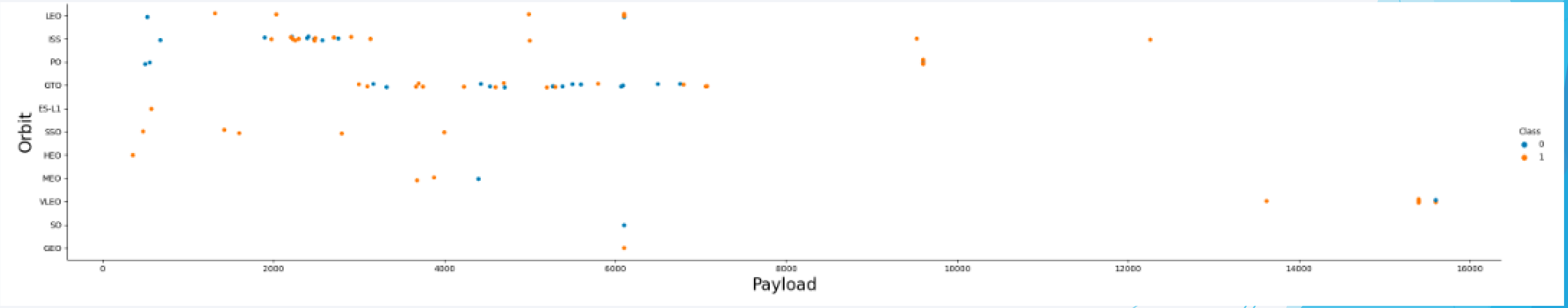
In the LEO orbit, success is related to the number of flights whereas in the GTO orbit, there is no relationship between flight number and the orbit.



Payload vs. Orbit Type

Scatter point of payload vs. orbit type

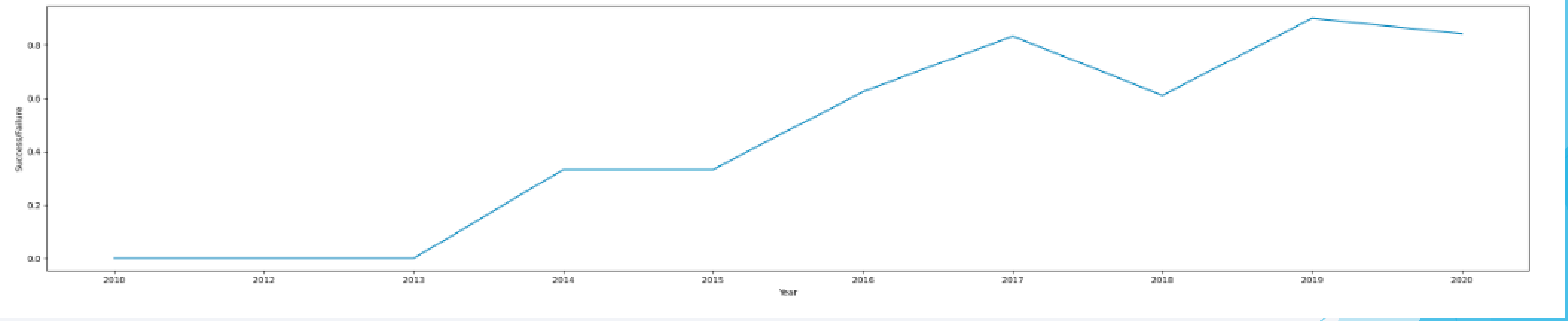
- With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.
- However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.



Launch Success Yearly Trend

Line chart of yearly average success rate.

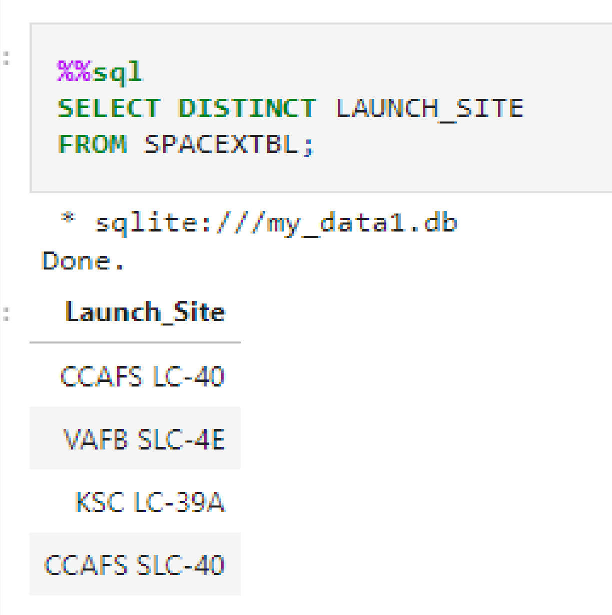
The success rate since 2013 kept increasing till 2020.



All Launch Site Names

Names of the unique launch sites

Used the key word **DISTINCT** to show only unique launch sites from the SpaceX data. There are 4 unique launch sites.



```
%%sql
SELECT DISTINCT LAUNCH_SITE
FROM SPACEXTBL;
```

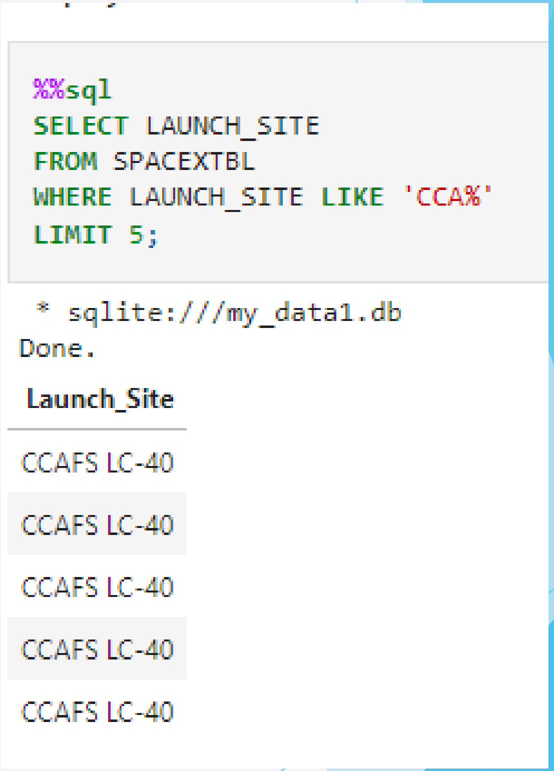
```
 * sqlite:///my_data1.db
Done.
```

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

Launch Site Names Begin with 'CCA'

5 records where launch sites begin with `CCA`

Used the query above to display 5 records where launch sites begin with `CCA`



```
%%sql
SELECT LAUNCH_SITE
FROM SPACEXTBL
WHERE LAUNCH_SITE LIKE 'CCA%'
LIMIT 5;
```

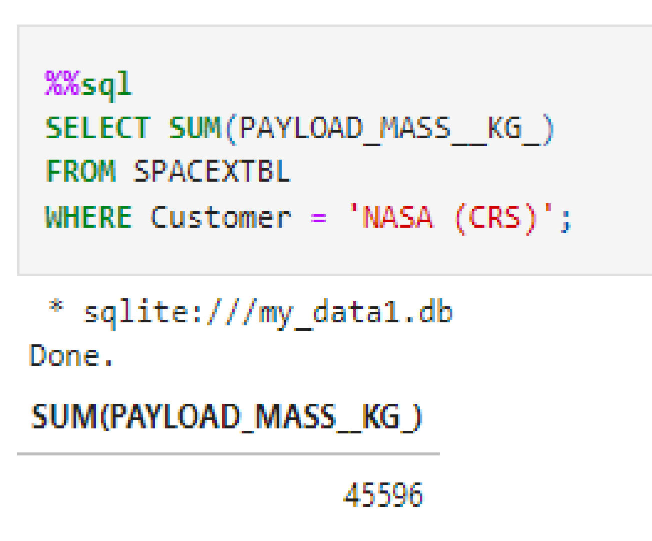
```
 * sqlite:///my_data1.db
Done.
```

Launch_Site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

Total Payload Mass

Total payload carried by boosters from NASA

Calculated the total payload carried by boosters from NASA as 45596 using the query below.



```
%%sql
SELECT SUM(PAYLOAD_MASS__KG_)
FROM SPACEXTBL
WHERE Customer = 'NASA (CRS)';
```

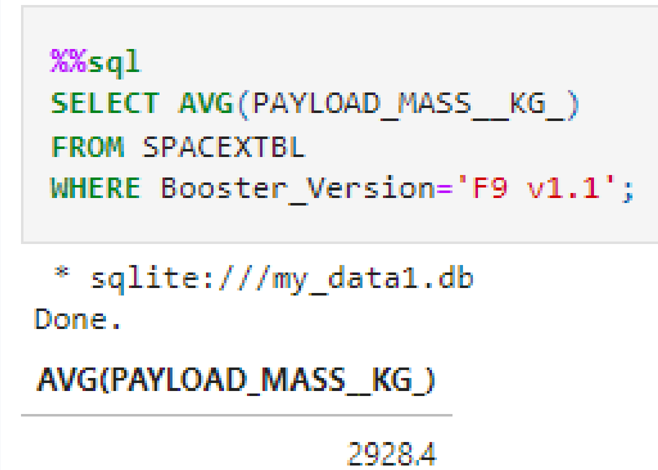
```
 * sqlite:///my_data1.db
Done.
```

SUM(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

Calculate the average payload mass carried by booster version F9 v1.1

We calculated the average payload mass carried by booster version F9 v1.1 as 2928.4



```
%%sql
SELECT AVG(PAYLOAD_MASS__KG_)
FROM SPACEXTBL
WHERE Booster_Version='F9 v1.1';
```

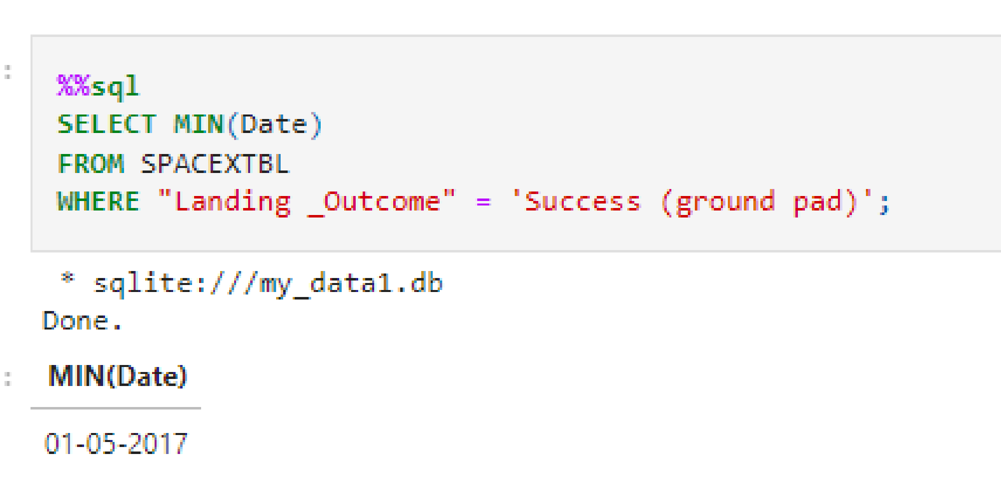
```
 * sqlite:///my_data1.db
Done.
```

AVG(PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

First successful landing outcome on ground pad

Date of the first successful landing outcome on ground pad was 1st June 2017



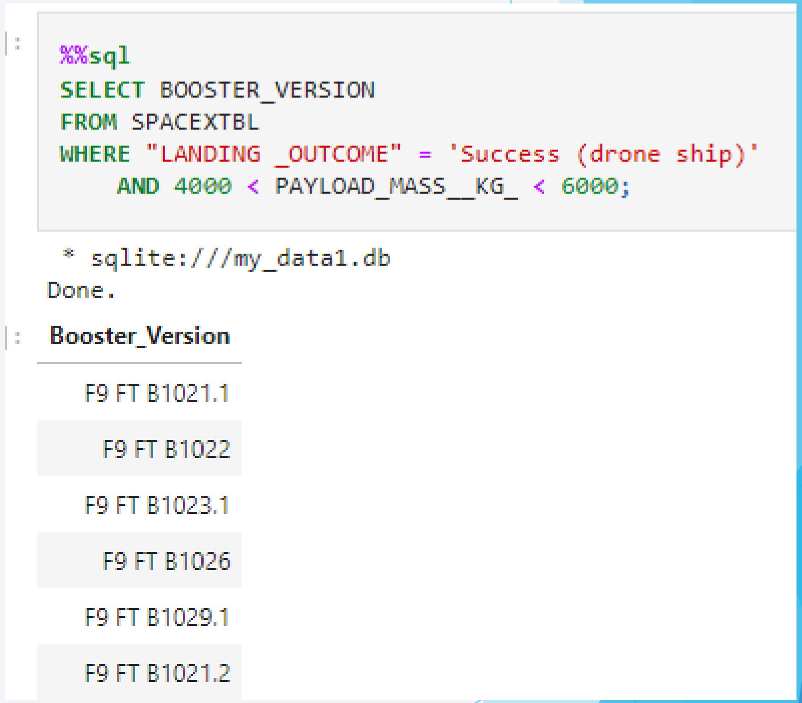
```
%%sql
SELECT MIN(Date)
FROM SPACEXTBL
WHERE "Landing _Outcome" = 'Success (ground pad)';
```

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

MIN(Date)
01-05-2017

Successful Drone Ship Landing with Payload between 4000 and 6000

Used the **WHERE** clause to filter for boosters which have successfully landed on drone ship and applied the **AND** condition to determine successful landing with payload mass greater than 4000 but less than 6000



```
%%sql
SELECT BOOSTER_VERSION
FROM SPACEXTBL
WHERE "LANDING _OUTCOME" = 'Success (drone ship)'
    AND 4000 < PAYLOAD_MASS__KG_ < 6000;
```

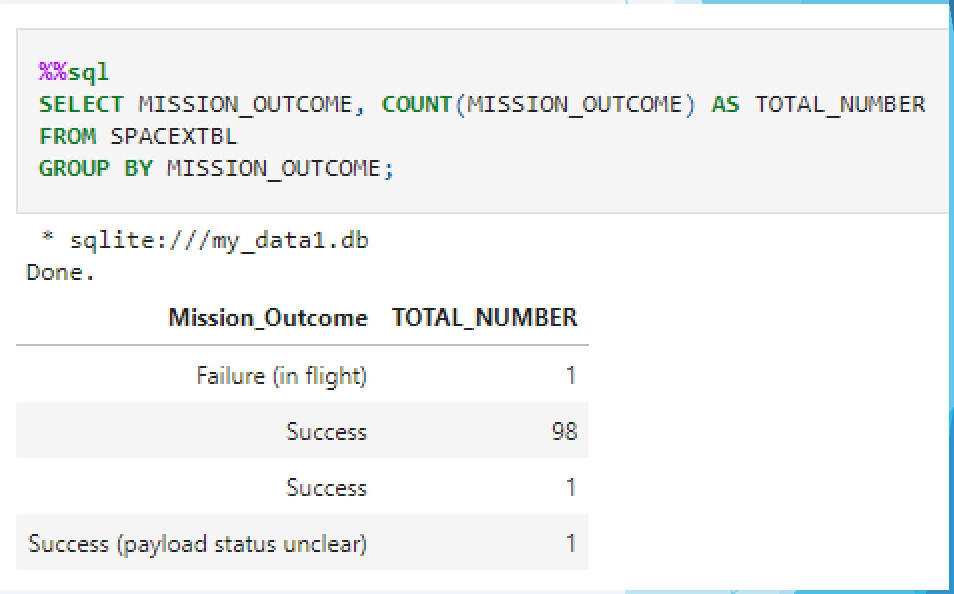
```
 * sqlite:///my_data1.db
Done.
```

Booster_Version
F9 FT B1021.1
F9 FT B1022
F9 FT B1023.1
F9 FT B1026
F9 FT B1029.1
F9 FT B1021.2

Total Number of Successful and Failure Mission Outcomes

Total number of successful and failure mission outcomes

Used **GROUPBY** to filter MissionOutcome was a success or a failure.



```
%%sql
SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) AS TOTAL_NUMBER
FROM SPACEXTBL
GROUP BY MISSION_OUTCOME;
```

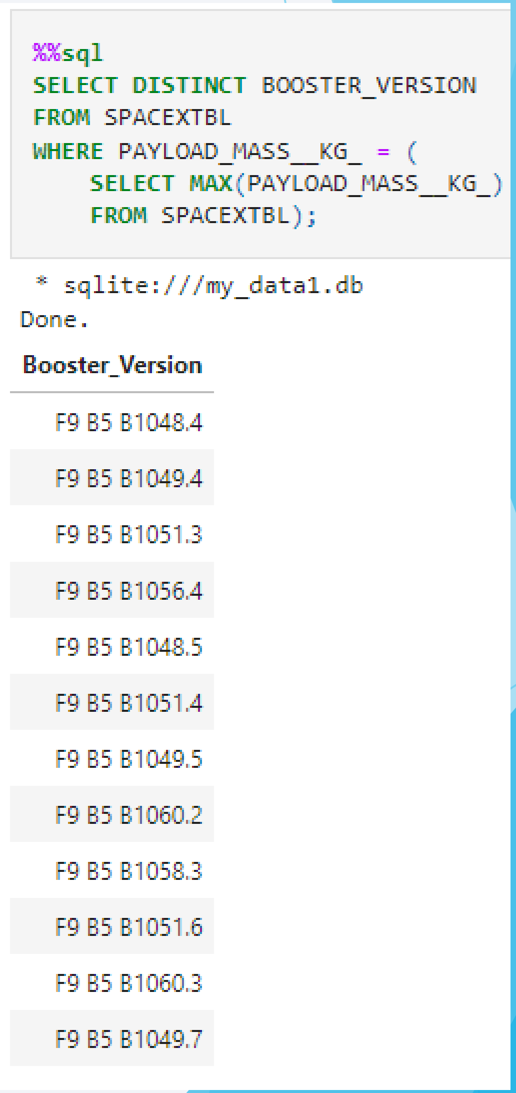
```
 * sqlite:///my_data1.db
Done.
```

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

Boosters Carried Maximum Payload

Booster which have carried the maximum payload mass

Determined the booster that have carried the maximum payload using a subquery in the **WHERE** clause and the **MAX()** function.



```
%%sql
SELECT DISTINCT BOOSTER_VERSION
FROM SPACEXTBL
WHERE PAYLOAD_MASS__KG_ = (
    SELECT MAX(PAYLOAD_MASS__KG_)
    FROM SPACEXTBL);
```

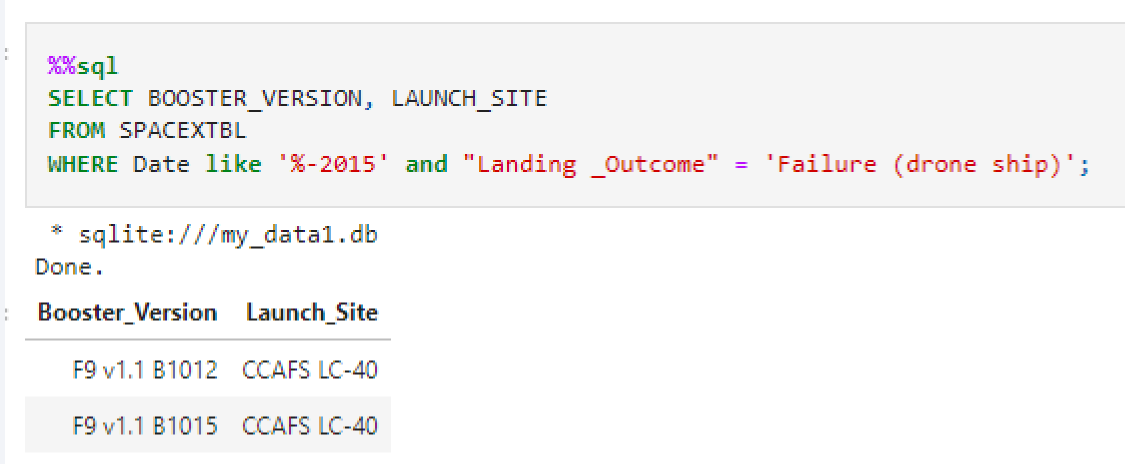
```
 * sqlite:///my_data1.db
Done.
```

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

2015 Launch Records

List of failed landing outcomes in drone ship, their booster versions, and launch site names for in year 2015.

Used a combinations of the **WHERE** clause, **LIKE**, **AND**, and **BETWEEN** conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015.



```
%%sql
SELECT BOOSTER_VERSION, LAUNCH_SITE
FROM SPACEXTBL
WHERE Date like '%-2015' and "Landing _Outcome" = 'Failure (drone ship)';
```

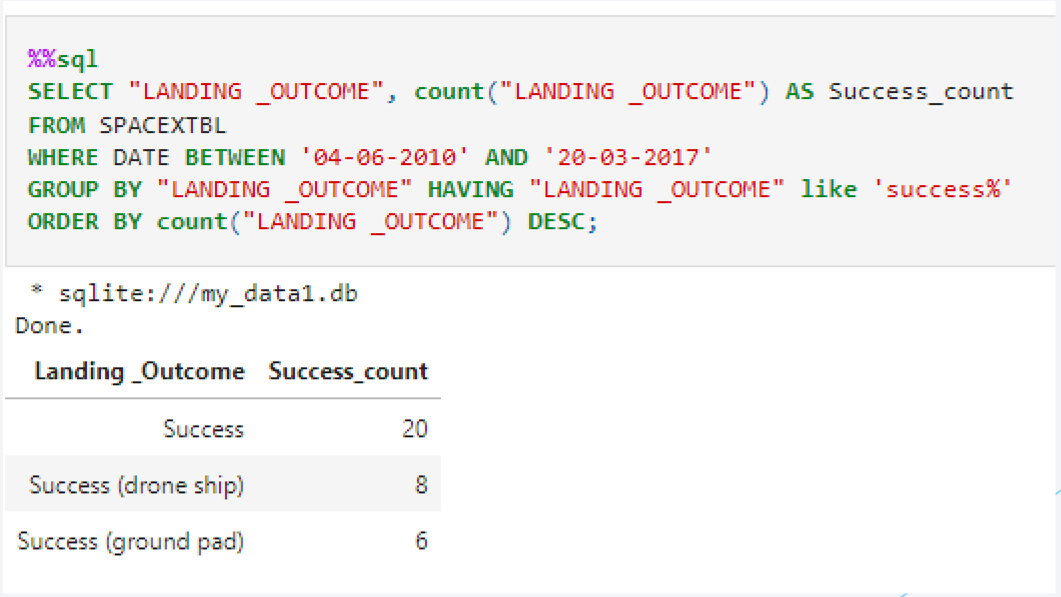
```
 * sqlite:///my_data1.db
Done.
```

Booster_Version	Launch_Site
F9 v1.1 B1012	CCAFS LC-40
F9 v1.1 B1015	CCAFS LC-40

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)) between the date 2010-06-04 and 2017-03-20, in descending order.

We applied the **GROUP BY** clause to group the landing outcomes and the **ORDER BY** clause to order the grouped landing outcome in descending order.



```
%%sql
SELECT "LANDING _OUTCOME", count("LANDING _OUTCOME") AS Success_count
FROM SPACEXTBL
WHERE DATE BETWEEN '04-06-2010' AND '20-03-2017'
GROUP BY "LANDING _OUTCOME" HAVING "LANDING _OUTCOME" like 'success%'
ORDER BY count("LANDING _OUTCOME") DESC;
```

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

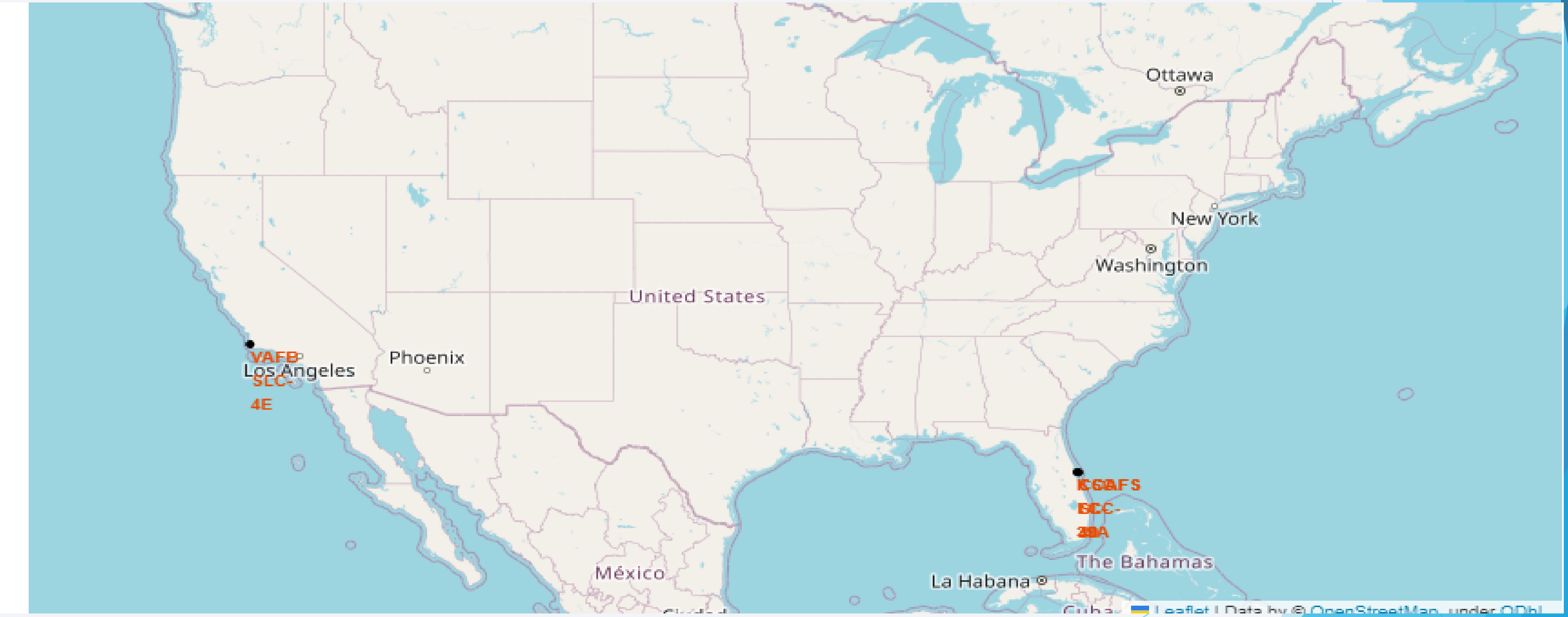
Landing _Outcome	Success_count
Success	20
Success (drone ship)	8
Success (ground pad)	6

Section 3

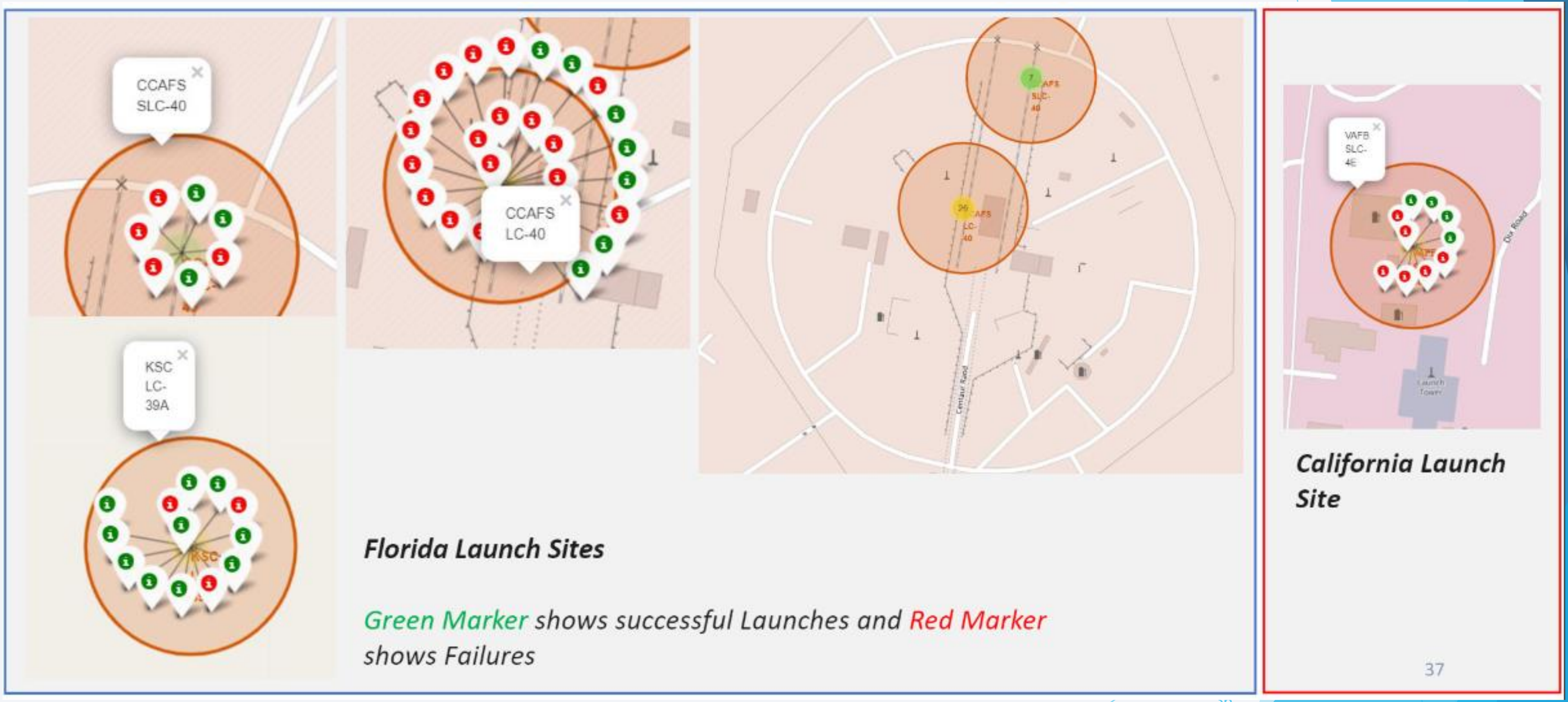
Launch Sites Proximities Analysis

Location of All launch sites global map

Launch sites are in United States of America coats. Florida and California



Markers showing launch sites with color labels

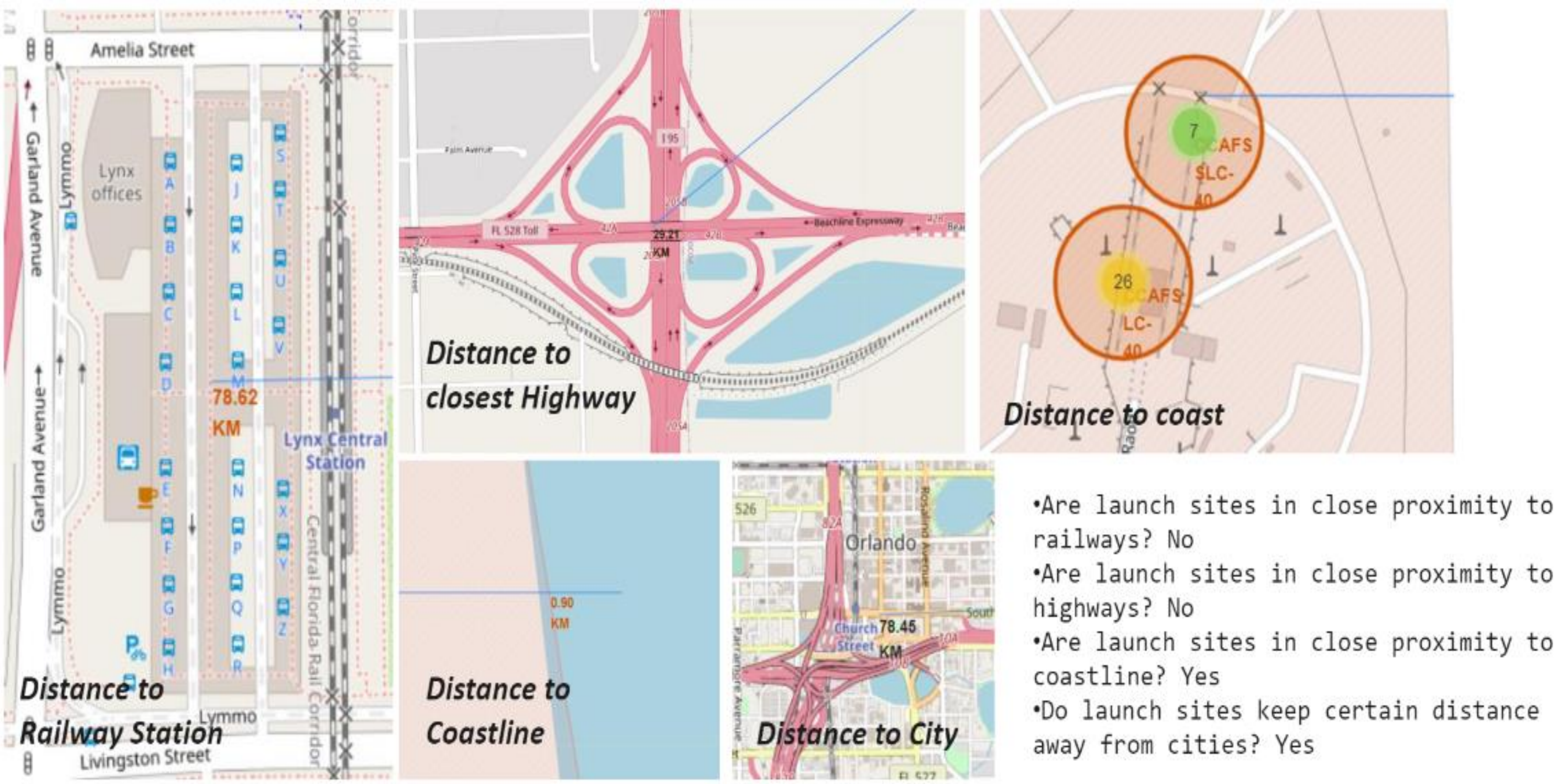


Florida Launch Sites

Green Marker shows successful Launches and Red Marker shows Failures

California Launch Site

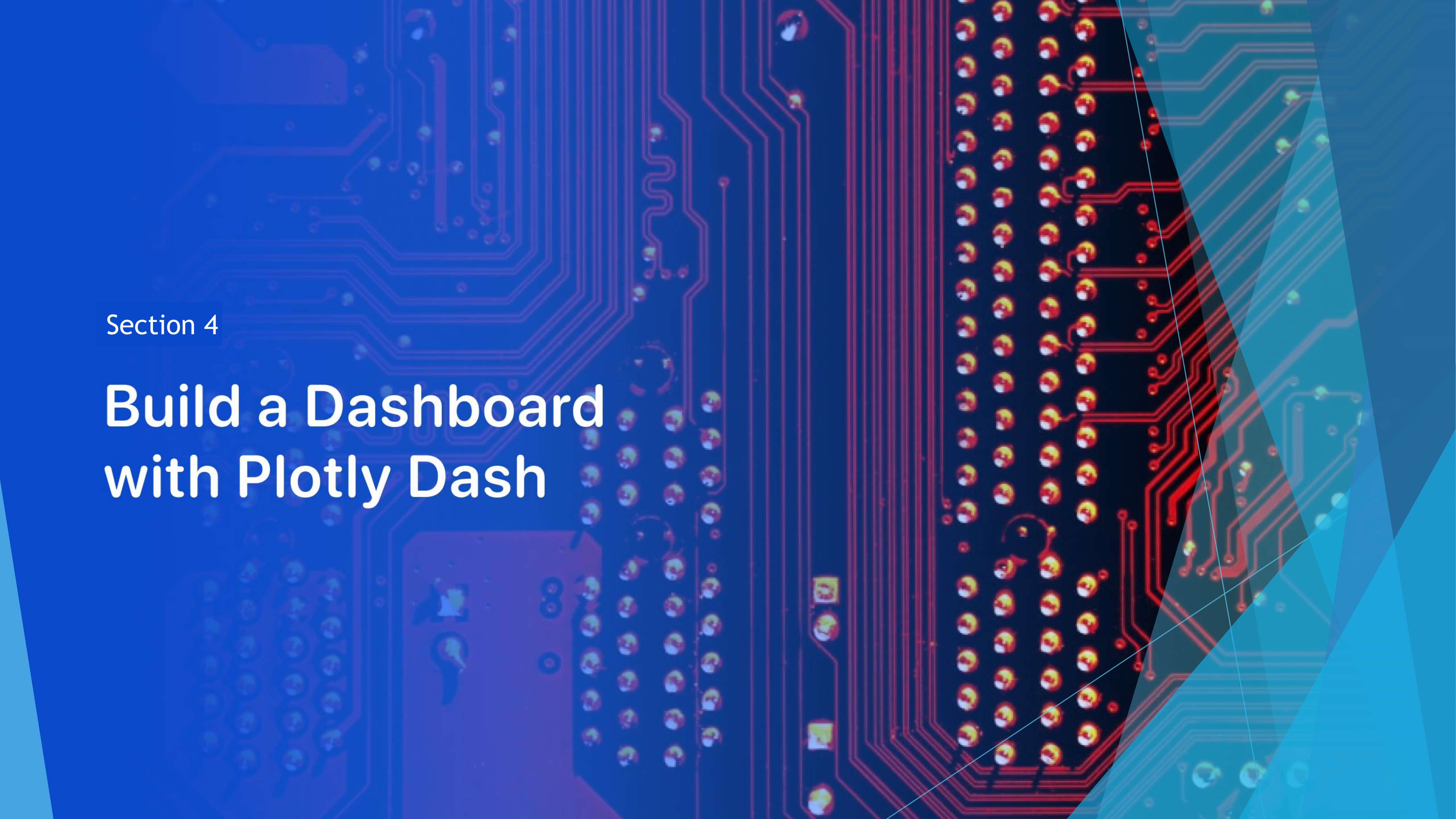
Launch Site distance to landmarks



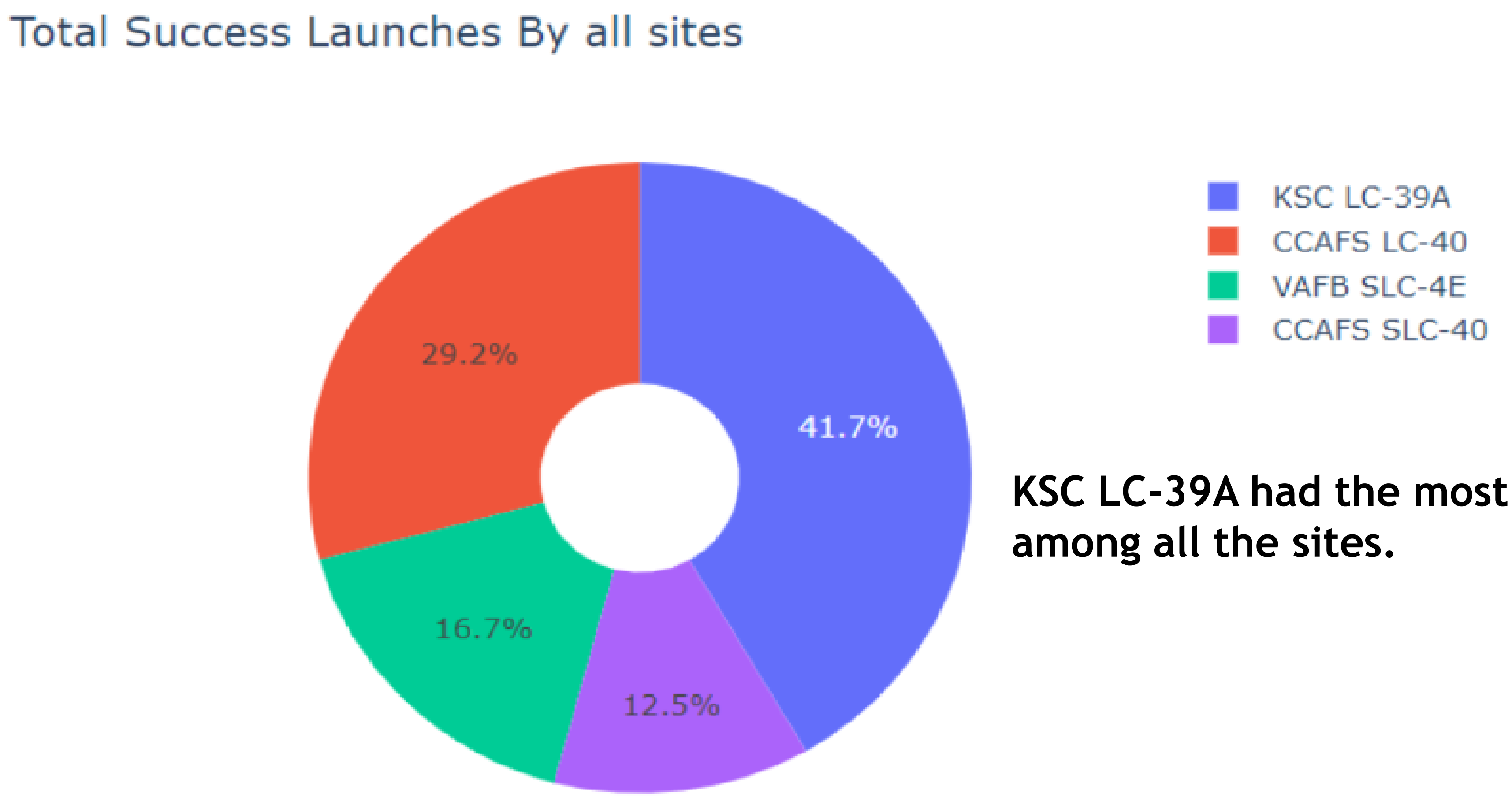
- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

Section 4

Build a Dashboard with Plotly Dash

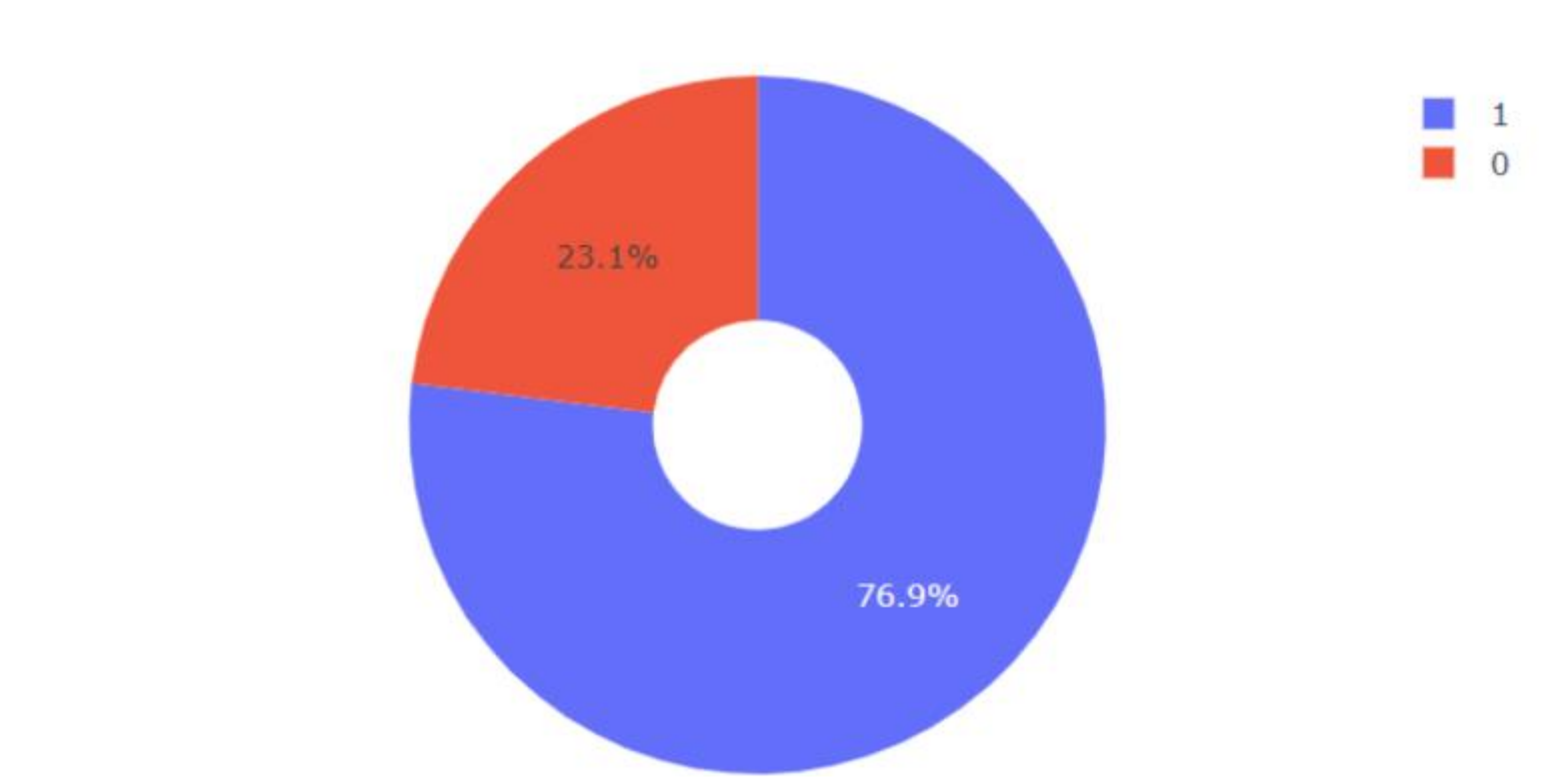


Pie chart showing the success percentage achieved by each launch site



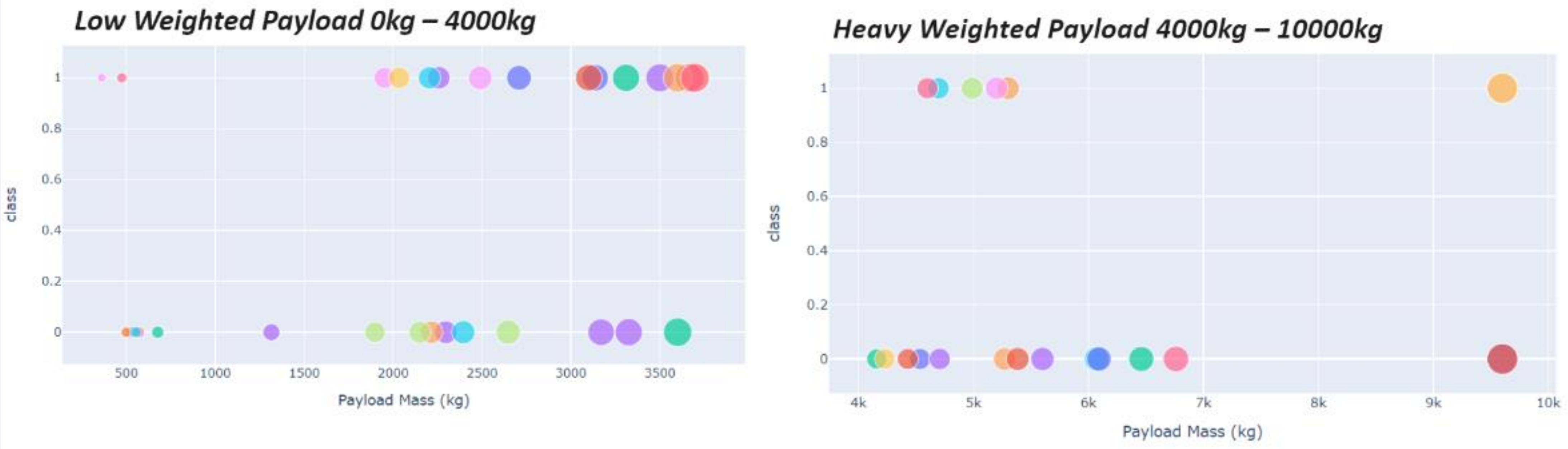
KSC LC-39A had the most successful launches among all the sites.

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



KSC LC-39A is the highest launch site with 76.9% success rate

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



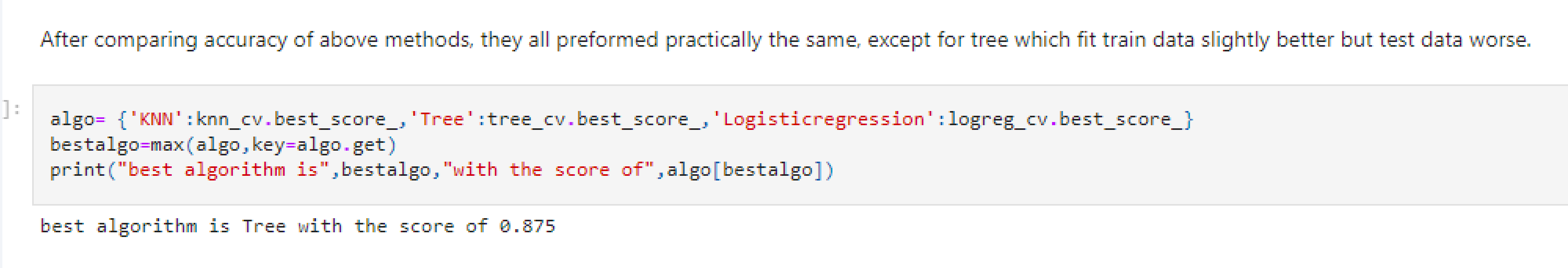
Success rate for low weighted payload is higher than heavy weighted payload

Section 5

Predictive Analysis (Classification)

Classification Accuracy

The best algorithm to be the Tree Algorithm which have the highest classification accuracy.



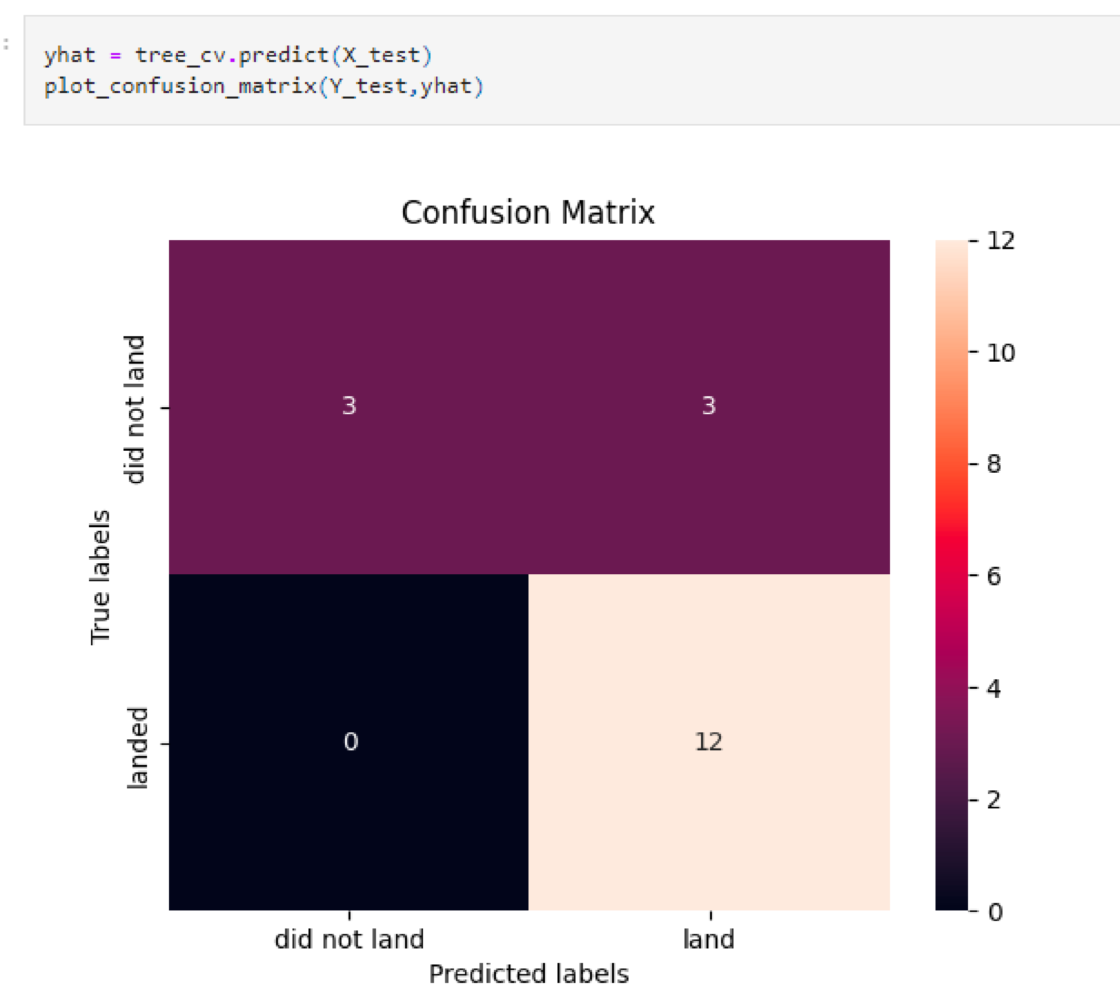
After comparing accuracy of above methods, they all preformed practically the same, except for tree which fit train data slightly better but test data worse.

```
]:  algo= {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'Logisticregression':logreg_cv.best_score_}
    bestalgo=max(algo,key=algo.get)
    print("best algorithm is",bestalgo,"with the score of",algo[bestalgo])
```

```
best algorithm is Tree with the score of 0.875
```

Confusion Matrix

The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes. The major problem is the false positives .i.e., unsuccessful landing marked as successful landing by the classifier.



```
yhat = tree_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```

Conclusions

These are the conclusions can be drawn:

- KSC LC-39A had the most successful launches of any sites.
- Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
- The larger the flight amount at a launch site, the greater the success rate at a launch site.
- Launch success rate started to increase in 2013 till 2020.
- SSO orbit have the most success rate; 100% and more than 1 occurrence.
- The Tree classifier is the best machine learning algorithm for this task.

Thank you!