IBM Developer
SKILLS NETWORK

# Winning Space Race with Data Science

Tinnapop Duangtha
23/11/2021

## Outline

# Executive Summary

## Summary of methodologies

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

## Summary of all results

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

## Introduction

### Project background and context

We will predict if the Falcon 9 first stage will land successfully. SpaceX 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 SpaceX 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 SpaceX for a rocket launch.

### Problems you want to find answers

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

Section 1

## Methodology

# Methodology

## Executive Summary

- Data collection methodology:
  - SpaceX API and Web Scraping from https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006555&utm_id=NA-SkillsNetwork-Channel-SkillsNetworkCoursesIBMDS0321ENSkillsNetwork26802033-2021-01-01
- Perform data wrangling
  - Dropping irrelevant columns and Using the One Hot Encoding for Machine Learning.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Using Logistics Regression, SVM, Decision Tree, and K-Nearest Neighbors.
  - Using 80% of dataset to train the models.

## Data Collection

- Getting the response from SpaceX API
- Convert it to .json file
- Clean the data
- Filter the rows (We keep only 'Falcon 9')
- Export to .csv file
- Web Scraping from Wikipedia (link on previous slide)
- Convert a table to panda DataFrame
- Export to .csv file

# Data Collection – SpaceX API

Requesting rocket launch data from SpaceX API

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

```
response = requests.get(spacex_url)
```

Convert json to pandas dataframe

```
# Use json_normalize method to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

Use the API again to get information about the launches using the IDs given

Create the empty lists to stored the data and will be used to create a new dataframe

```
# Global variables
BoosterVersion = []
PayloadMass = []
Orbit = []
LaunchSite = []
Outcome = []
Flights = []
GridFins = []
Reused = []
Legs = []
LandingPad = []
Block = []
ReusedCount = []
Serial = []
Longitude = []
Latitude = []
```

Filter the dataframe to only include Falcon 9 launches

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = df.loc[df['BoosterVersion']!='Falcon 1']
```

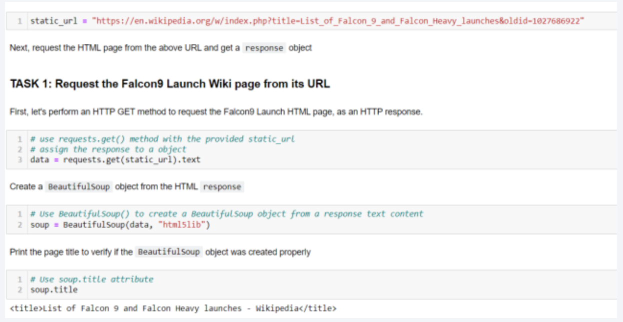
Clean the data (null)

Export to .csv

Link for Notebook:

# Data Collection - Scraping

Using BeautifulSoup to get html code



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

Next, request the HTML page from the above URL and get a `response` object

**TASK 1: Request the Falcon9 Launch Wiki page from its URL**

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

```
# 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`

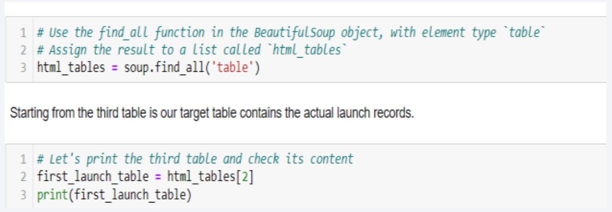
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(data, "html5lib")
```

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

```
# Use soup.title attribute
soup.title
```

```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

Extract all column/variable names from the HTML table header

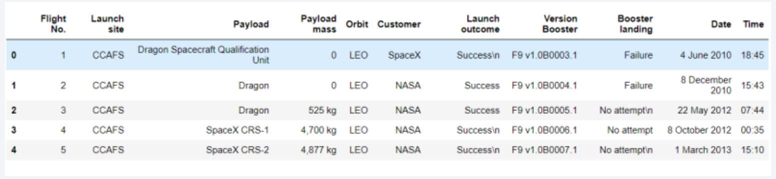


```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
```

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)
```

Create a data frame by parsing the launch HTML tables



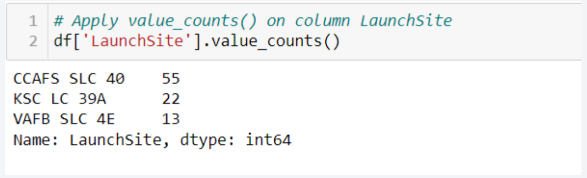
|   | Flight No. | Launch site | Payload | Payload mass | Orbit | Customer | Launch outcome | Version Booster | Booster landing | Date | Time |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | CCAFS | Dragon Spacecraft Qualification Unit | 0 | LEO | SpaceX | Success\n | F9 v1.0B0003.1 | Failure | 4 June 2010 | 18:45 |
| 1 | 2 | CCAFS | Dragon | 0 | LEO | NASA | Success | F9 v1.0B0004.1 | Failure | 8 December 2010 | 15:43 |
| 2 | 3 | CCAFS | Dragon | 525 kg | LEO | NASA | Success | F9 v1.0B0005.1 | No attempt\n | 22 May 2012 | 07:44 |
| 3 | 4 | CCAFS | SpaceX CRS-1 | 4,700 kg | LEO | NASA | Success\n | F9 v1.0B0006.1 | No attempt | 8 October 2012 | 00:35 |
| 4 | 5 | CCAFS | SpaceX CRS-2 | 4,877 kg | LEO | NASA | Success\n | F9 v1.0B0007.1 | No attempt\n | 1 March 2013 | 15:10 |

Export to .csv

Link for Notebook:

# Data Wrangling

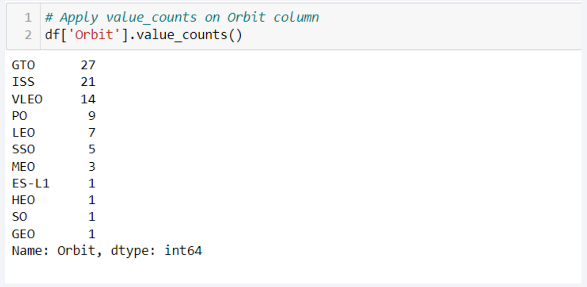
Calculate the number of launches on each site



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

```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

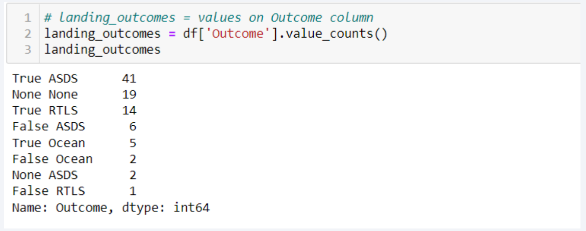
Calculate the number and occurrence of each orbit



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

```
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
```

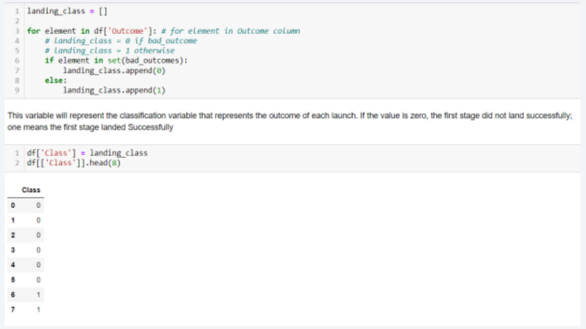
Calculate the number and occurrence of mission outcome per orbit type



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

Create a landing outcome label from Outcome column



```python
landing_class = []

for element in df['Outcome']: # for element in Outcome column
    # landing_class = 0 if bad_outcome
    # landing_class = 1 otherwise
    if element in set(bad_outcomes):
        landing_class.append(0)
    else:
        landing_class.append(1)
```

This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

```python
df['Class'] = landing_class
df[['Class']].head(8)
```

|   | Class |
|---|---|
| 0 | 0 |
| 1 | 0 |
| 2 | 0 |
| 3 | 0 |
| 4 | 0 |
| 5 | 0 |
| 6 | 1 |
| 7 | 1 |

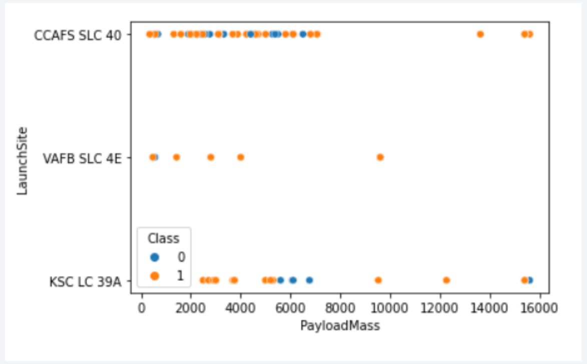
Export to .csv

Link for Notebook:

# EDA with Data Visualization

Scatter Plot

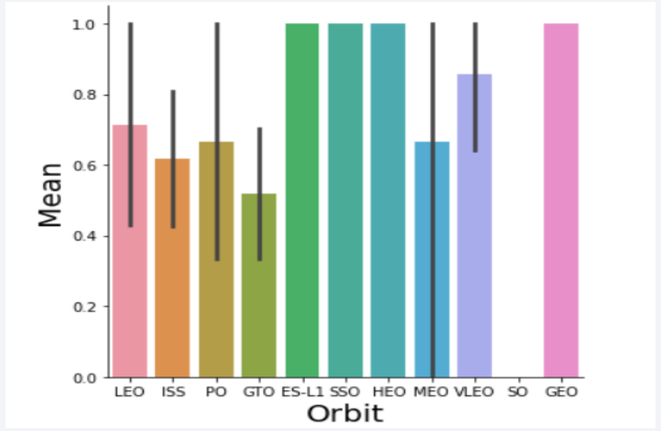
- Flight Number vs. Payload Mass
- Flight Number vs. Launch Site
- Payload Mass vs. Launch Site
- Flight Number vs. Orbit
- Payload vs. Orbit
- Payload Mass vs. Orbit



To find the relationship between variables

Bar Plot

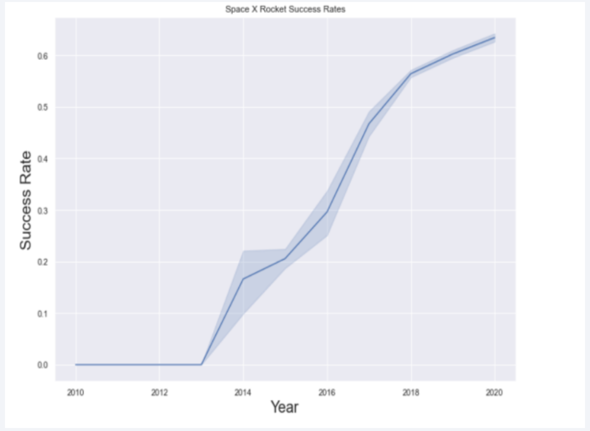
- Orbit vs Class Mean



To classify the data and easy to understand

Line Plot

- Success Rate vs. Year



To show the trend of two variables

Link for Notebook:

# EDA with SQL

**Using SQL Queries to**
- 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 successful 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.
- List the failed landing outcomes in drone ship, their booster versions, and launch site names for in year 2015
- 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

Link for Notebook:

# Build an Interactive Map with Folium

**To visualize the launch site data into a map**, We use features 'Latitude and Longitude' of each launch site and circle around it by using folium.Marker()

**Mark the success/failed launches for each site on the map.** We add a new feature called 'marker_color' to store the marker colors based on the `class` value, that is the red means class = 0 and green means class = 1 in the MarkerCluster()

**We Calculate the distances between a launch site to its proximities using latitude and longitude.** In this step, We can answer ther questions :

- Are launch sites in close proximity to railways?
- Are launch sites in close proximity to highways?
- Are launch sites in close proximity to coastline?
- Do launch sites keep certain distance away from cities?

Link for Notebook:

## Build a Dashboard with Plotly Dash

**We use the pie chart to shows the total launches by the certain site or all site**

- Display the percentage of Success vs. Failure
- It is easy to understand because the data has only 2 classes

**We use the scatter plot to shows the relationship with Outcome and Payload Mass (Kg) for the different Booster Versions**

- This plot shows the relationship between two variables.
- It is the best method to show a non-linear pattern.

Link for .py file:

## Predictive Analysis (Classification)

**Building models**

1. Create a NumPy array from the column Class in data.
2. Standardize the data in X
3. Create test/train set
4. Use Logistic Regression, SVM, Decision Tree, and KNN

**Evaluate models**

1. Calculate the accuracy on the test data (all model).
2. Get tuned hyperparameters for each type of algorithms
3. Plot the confusion matrix

**Improving models and Finding the best model**

1. Feature Engineering
2. Algorithm Tuning
3. Find the model that has the best accuracy score

Link for Notebook:

## Results

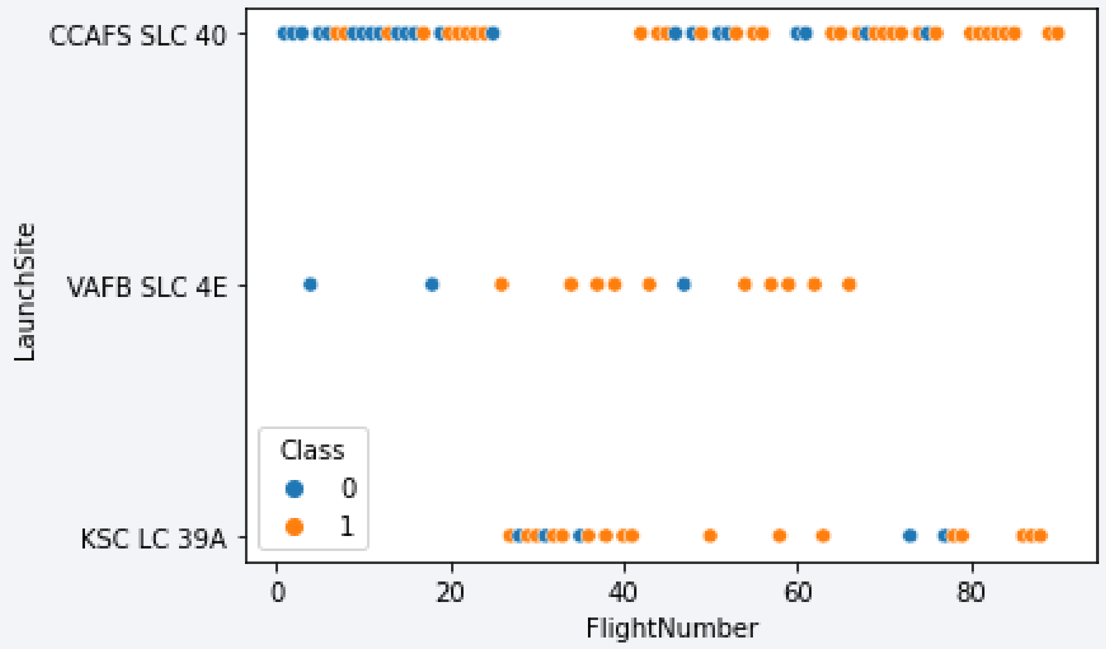
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

## Insights drawn from EDA

## Flight Number vs. Launch Site

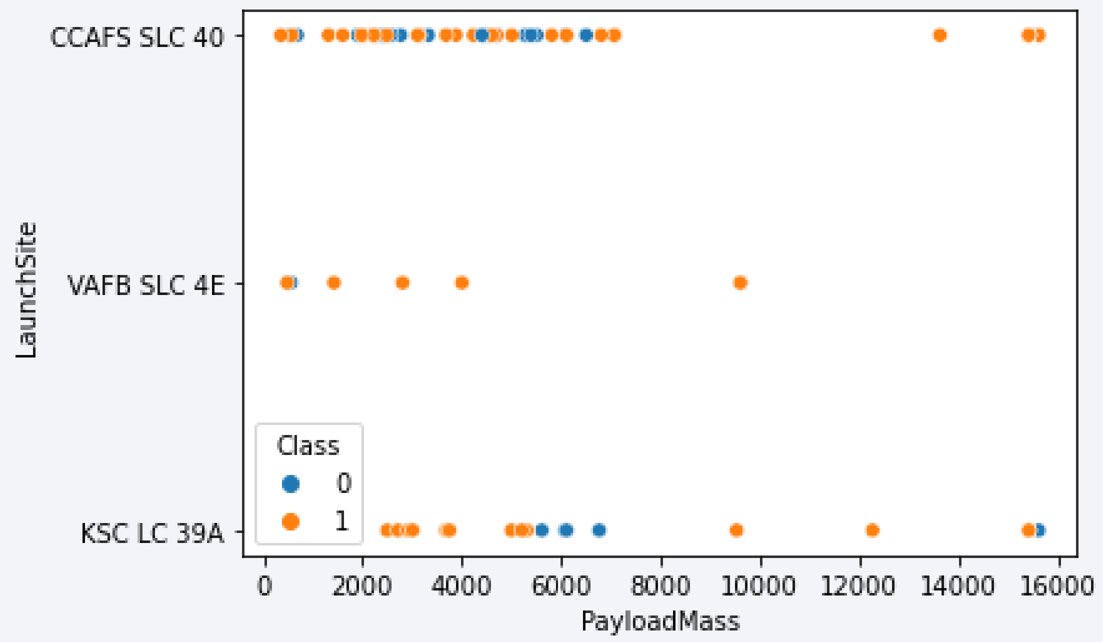
**A scatter plot of Flight Number vs. Launch Site**



We can say that when increasing the flight number, the success rate increase in all launch site. In addition, CCAFS SLC 40 has the first successful landing at the less flight number.

## Payload vs. Launch Site

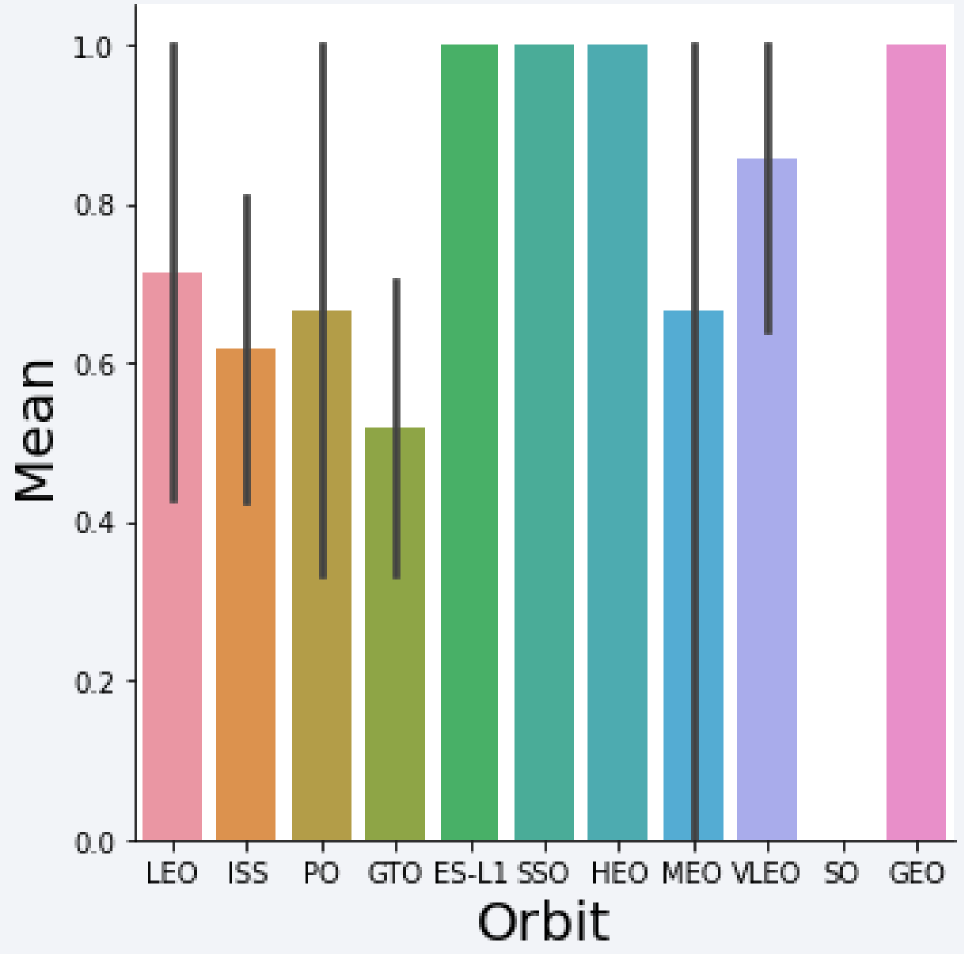
**A scatter plot of Payload vs. Launch Site**



We can say that when increasing the payload mass in the VAFB SLC 4E launch site, the success rate increase, the CCAFS SLC 40 launch site has a more successful landing at the heavy payload mass and the KSC LC 39A launch site has a random in successful.

## Success Rate vs. Orbit Type

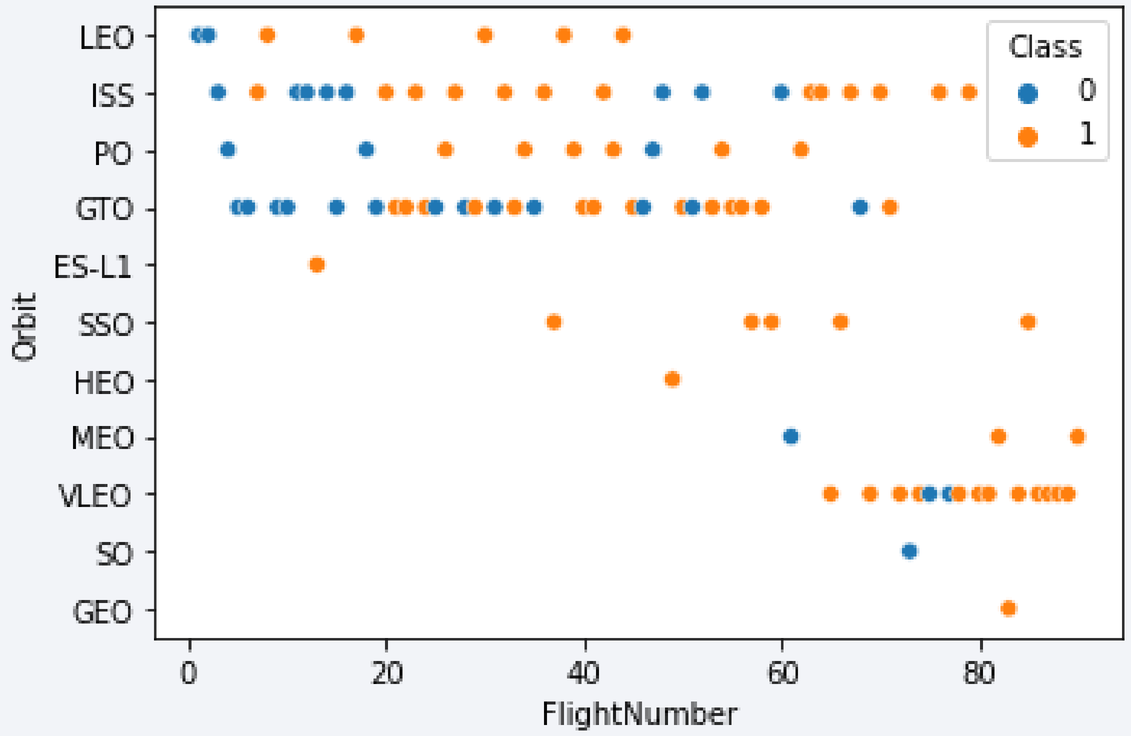
**A scatter plot of Success Rate vs. Orbit Type**



We can say that the Orbit ES-L1, SSO, HEO, and GEO have the best success rate.

# Flight Number vs. Orbit Type

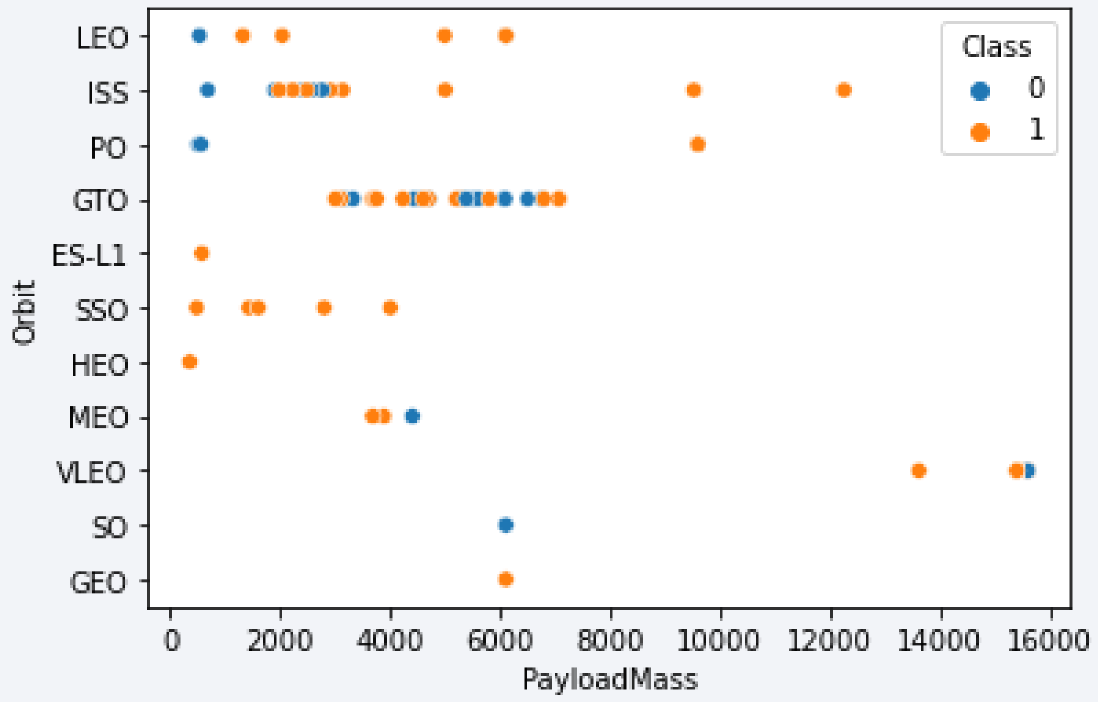
**A scatter plot of Flight Number vs. Orbit Type**



We can say that the LEO orbit success appears related to the number of flights, on the other hand, there seems to be no relationship between flight numbers when in GTO orbit.

# Payload vs. Orbit Type

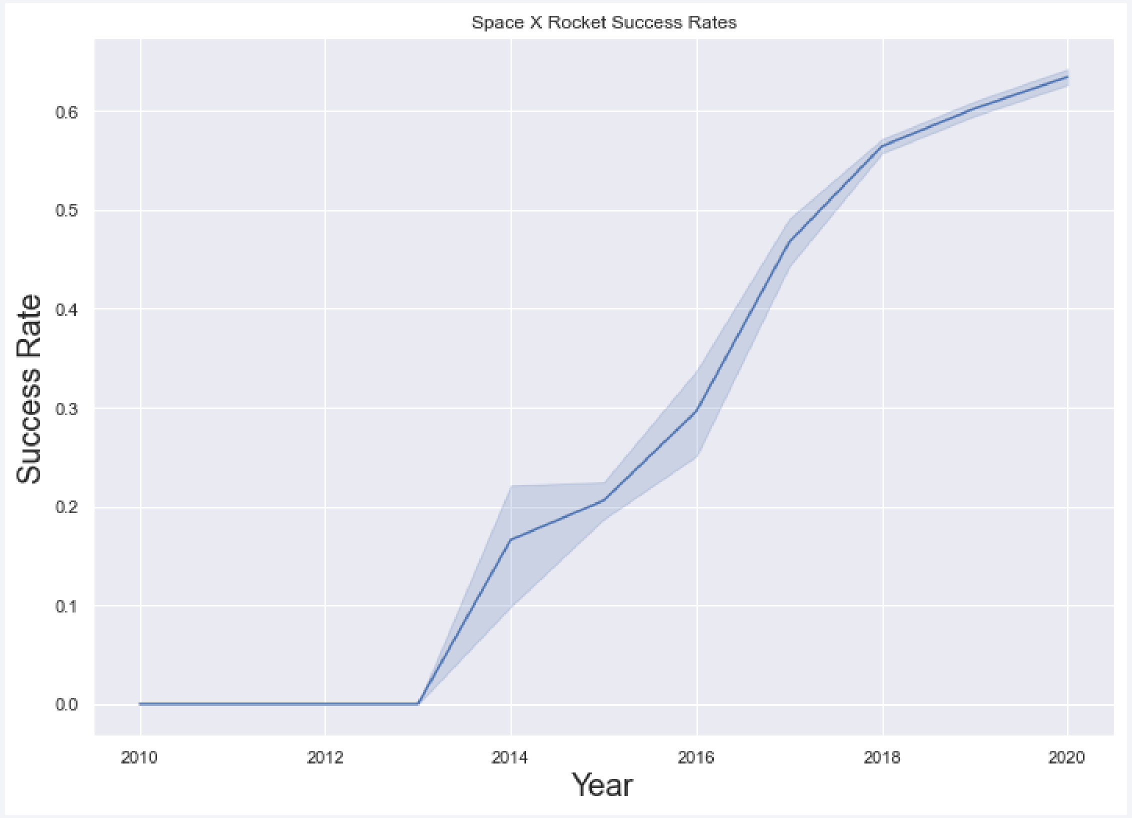
**A scatter plot of Payload vs. Orbit Type**



We can say that the heavy payloads the successful landing or positive landing rate is 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

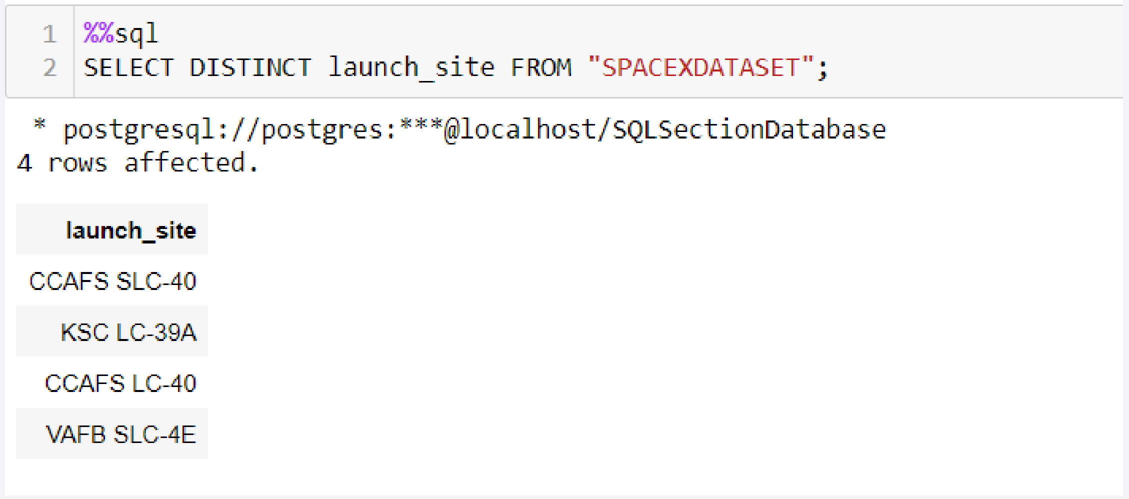
**A line plot of yearly average success rate**



We can say that the success rate since 2013 kept increasing till 2020

# All Launch Site Names

**Find the names of the unique launch sites**



```
%%sql
SELECT DISTINCT launch_site FROM "SPACEXDATASET";
```

```
 * postgresql://postgres:***@localhost/SQLSectionDatabase
4 rows affected.
```

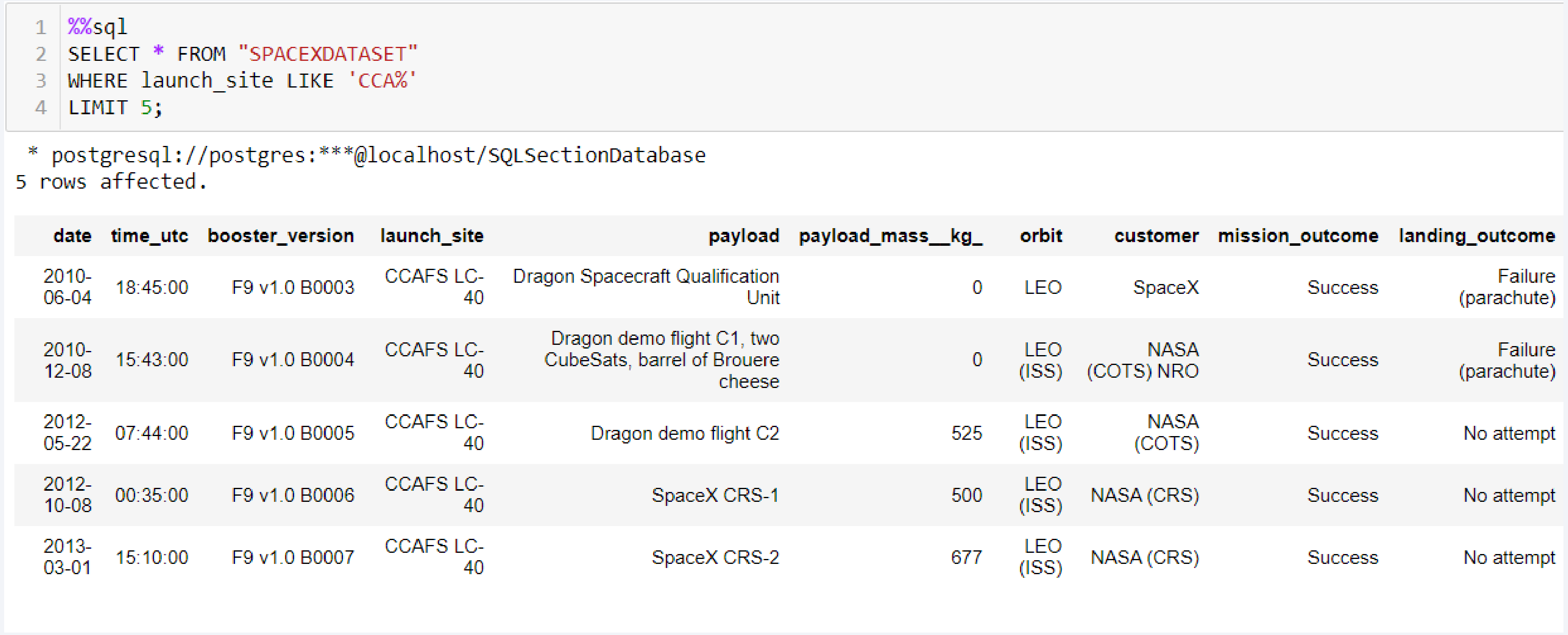
| launch_site |
|---|
| CCAFS SLC-40 |
| KSC LC-39A |
| CCAFS LC-40 |
| VAFB SLC-4E |

We select the unique name of launch sites from the SPACEXDATASET table.
We have 4 different launch sites.

Remark: I use postgresql as a RDBMS.

# Launch Site Names Begin with 'CCA'

**Find 5 records where launch sites begin with `CCA`**



```
%%sql
SELECT * FROM "SPACEXDATASET"
WHERE launch_site LIKE 'CCA%'
LIMIT 5;
```

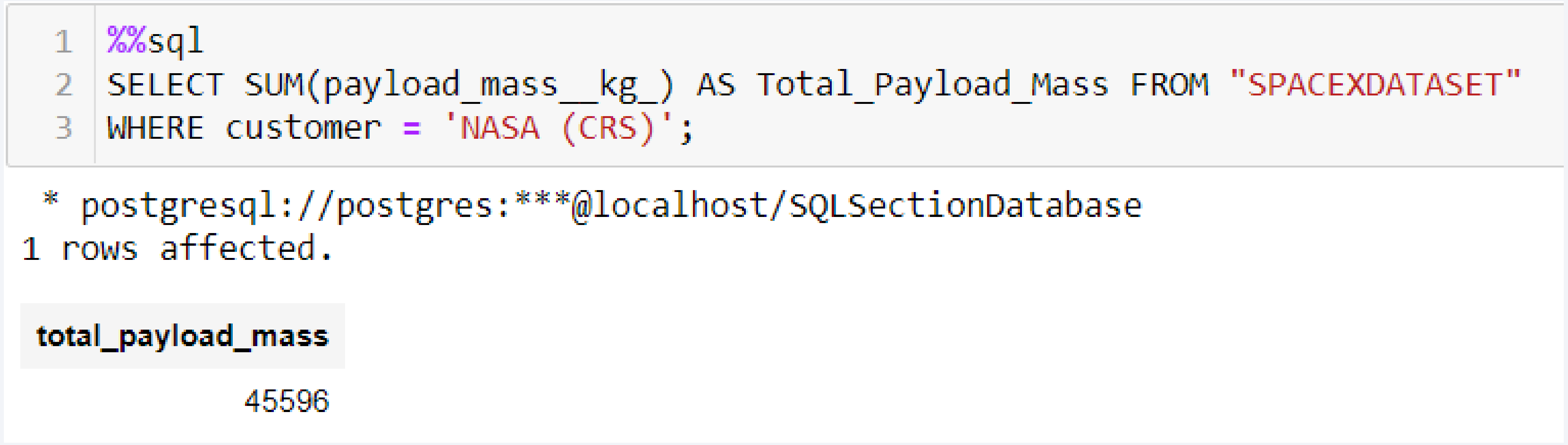
```
 * postgresql://postgres:***@localhost/SQLSectionDatabase
5 rows affected.
```

| date | time_utc | booster_version | launch_site | payload | payload_mass__kg_ | orbit | customer | mission_outcome | landing_outcome |
|---|---|---|---|---|---|---|---|---|---|
| 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 | 07:44:00 | F9 v1.0 B0005 | CCAFS LC-40 | Dragon demo flight C2 | 525 | LEO (ISS) | NASA (COTS) | Success | No attempt |
| 2012-10-08 | 00:35:00 | 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 |

We select all columns from the SPACEXDATASET table with a condition: launch sites begin with CCA then limit to show only 5 rows.

## Total Payload Mass

**Find Total Payload Mass which the customer is NASA (CRS)**



```
%%sql
SELECT SUM(payload_mass__kg_) AS Total_Payload_Mass FROM "SPACEXDATASET"
WHERE customer = 'NASA (CRS)';
```

```
 * postgresql://postgres:***@localhost/SQLSectionDatabase
1 rows affected.
```

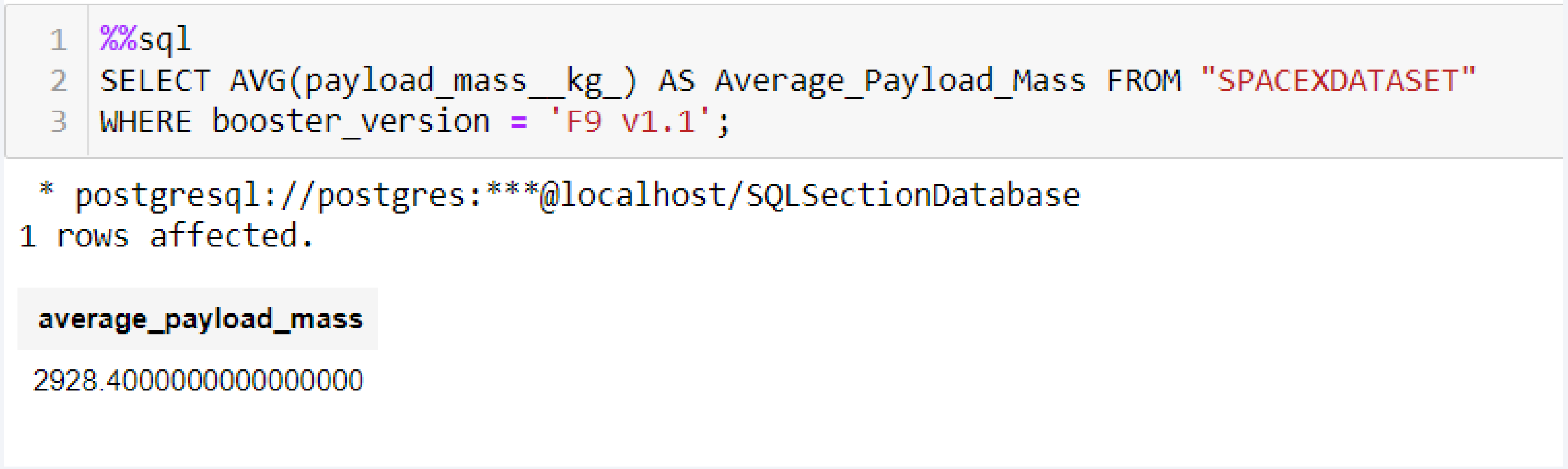
| total_payload_mass |
|---|
| 45596 |

We select the sum of the payload mass column and name it from the SPACEXDATASET table with a condition: the customer is NASA (CRS).

We have the total payload mass which the customer is NASA (CRS) --> 45596 kg.

# Average Payload Mass by F9 v1.1

**Find the Average Payload Mass by F9 v1.1**



```
%%sql
SELECT AVG(payload_mass__kg_) AS Average_Payload_Mass FROM "SPACEXDATASET"
WHERE booster_version = 'F9 v1.1';
```

```
 * postgresql://postgres:***@localhost/SQLSectionDatabase
1 rows affected.
```

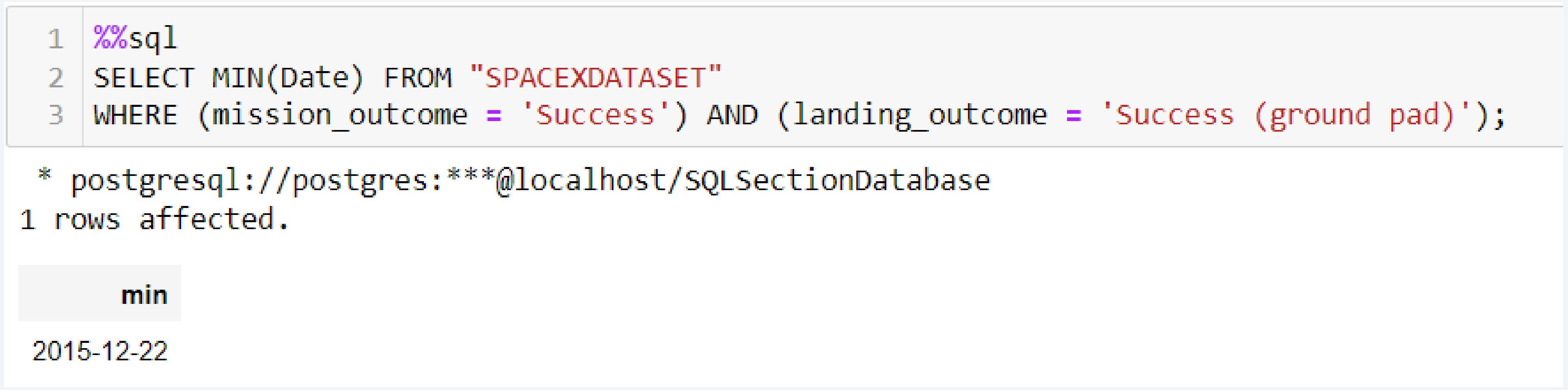
| average_payload_mass |
|---|
| 2928.4000000000000000 |

We select the average of the payload mass column and name it from the SPACEXDATASET table with a condition: the booster version is F9 V1.1.

We have the average payload mass which the booster version is F9 V1.1 --> 2928.4 kg.

## First Successful Ground Landing Date

**Find the dates of the first successful landing outcome on ground pad**



```
%%sql
SELECT MIN(Date) FROM "SPACEXDATASET"
WHERE (mission_outcome = 'Success') AND (landing_outcome = 'Success (ground pad)');
```

```
 * postgresql://postgres:***@localhost/SQLSectionDatabase
1 rows affected.
```

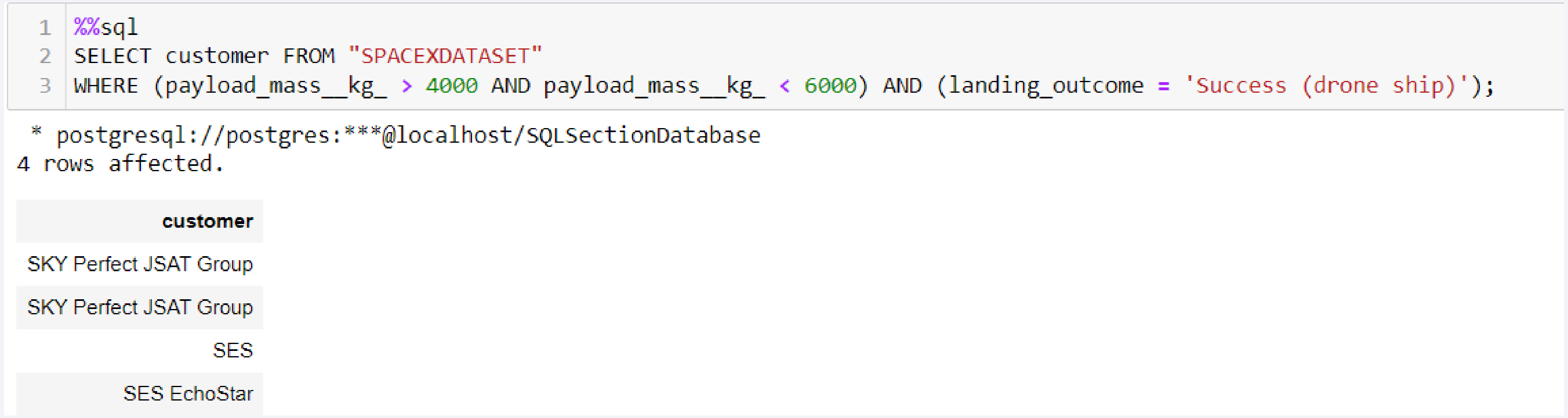
| min |
|---|
| 2015-12-22 |

We select the minimum of date column from the SPACEXDATASET table with the conditions: the mission outcome and the landing outcome are success.

We have the first date which  the mission outcome and the landing outcome are success
--> 22 December 2015.

## Successful Drone Ship Landing with Payload between 4000 and 6000

**List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000**



```
%%sql
SELECT customer FROM "SPACEXDATASET"
WHERE (payload_mass__kg_ > 4000 AND payload_mass__kg_ < 6000) AND (landing_outcome = 'Success (drone ship)');
```

```
 * postgresql://postgres:***@localhost/SQLSectionDatabase
4 rows affected.
```

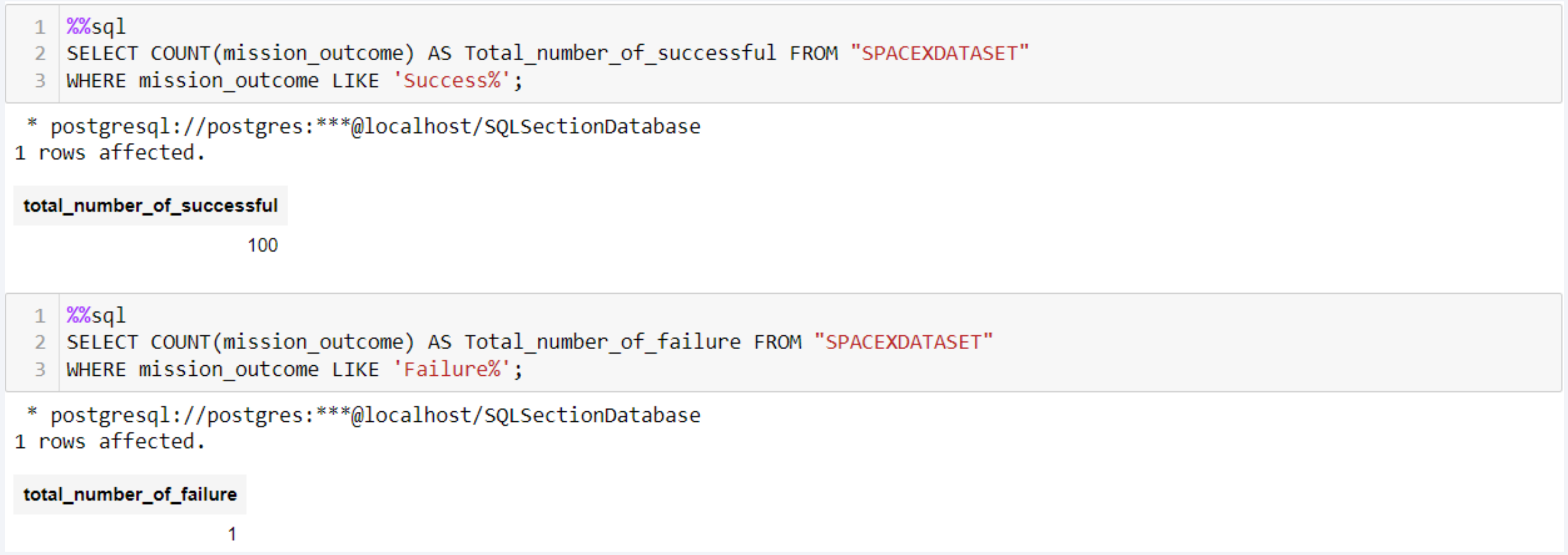
| customer |
|---:|
| SKY Perfect JSAT Group |
| SKY Perfect JSAT Group |
| SES |
| SES EchoStar |

We select the customer column from the SPACEXDATASET table with the conditions: the payload mass greater than 4000 but less than 6000 and successfully landed on a drone ship.

We have only 4 customers including SKY Perfect JSAT Group, SKY Perfect JSAT Group, SES, and SES EchoStar which have successfully landed on a drone ship and had payload mass greater than 4000 but less than 6000.

## Total Number of Successful and Failure Mission Outcomes

**Calculate the total number of successful and failure mission outcomes**



```
%%sql
SELECT COUNT(mission_outcome) AS Total_number_of_successful FROM "SPACEXDATASET"
WHERE mission_outcome LIKE 'Success%';
```

```
 * postgresql://postgres:***@localhost/SQLSectionDatabase
1 rows affected.
```

| total_number_of_successful |
|---|
| 100 |

```
%%sql
SELECT COUNT(mission_outcome) AS Total_number_of_failure FROM "SPACEXDATASET"
WHERE mission_outcome LIKE 'Failure%';
```

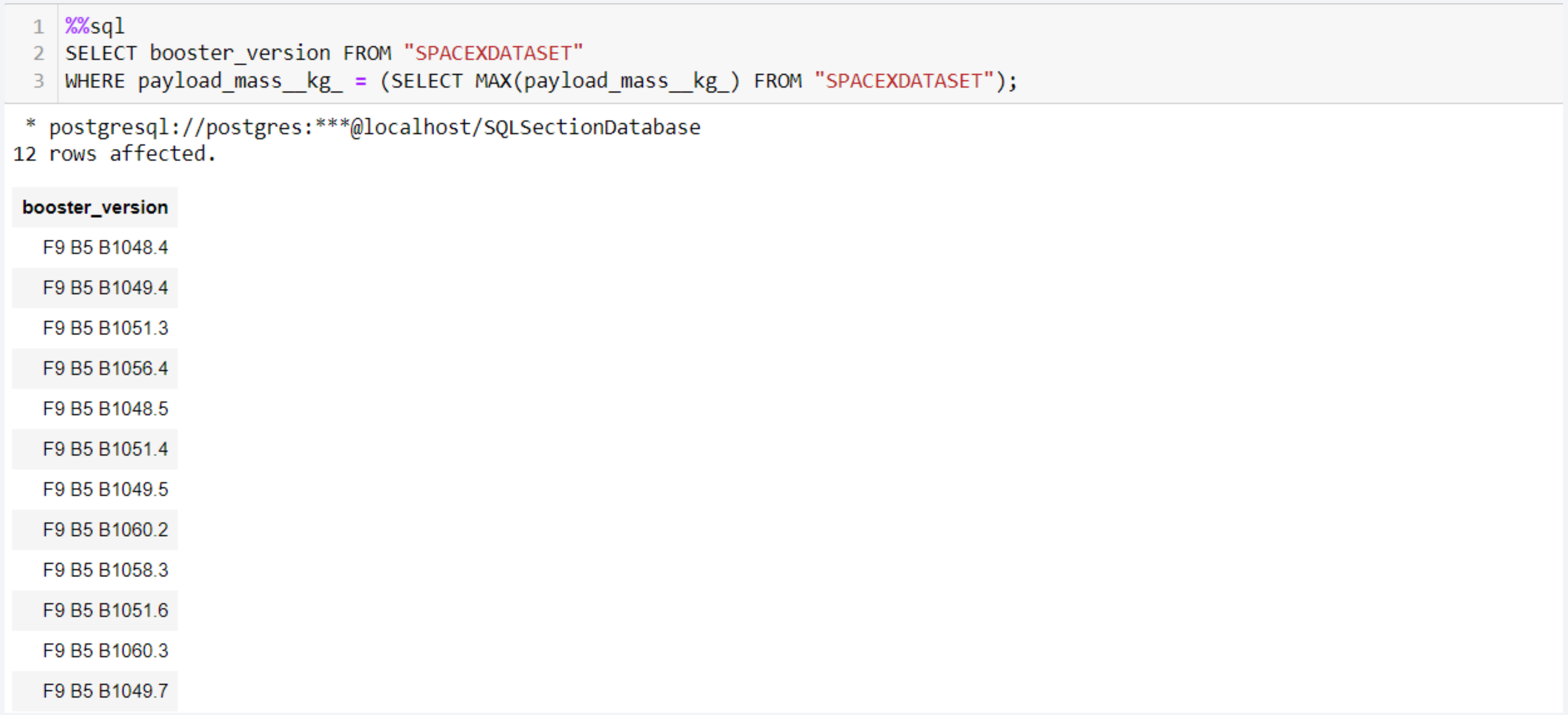
```
 * postgresql://postgres:***@localhost/SQLSectionDatabase
1 rows affected.
```

| total_number_of_failure |
|---|
| 1 |

As you can see, the total number of successfully landing is 100 and the total number of unsuccessful landings is only 1.

## Boosters Carried Maximum Payload

**List the names of the booster which have carried the maximum payload mass**



```
%%sql
SELECT booster_version FROM "SPACEXDATASET"
WHERE payload_mass__kg_ = (SELECT MAX(payload_mass__kg_) FROM "SPACEXDATASET");
```

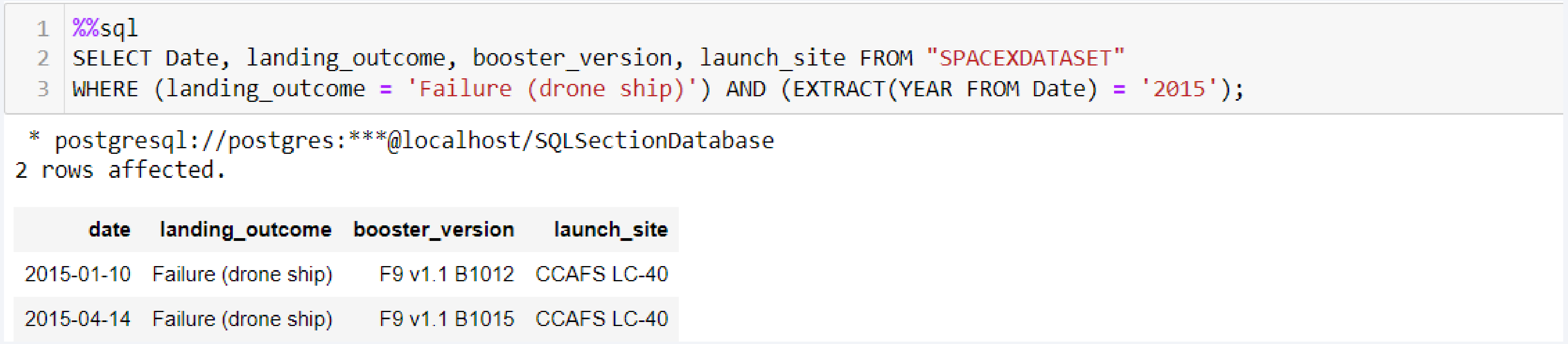
```
 * postgresql://postgres:***@localhost/SQLSectionDatabase
12 rows affected.
```

| 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 |

As you can see, We got 12 booster versions which have carried the maximum payload mass.

# 2015 Launch Records

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



```
%%sql
SELECT Date, landing_outcome, booster_version, launch_site FROM "SPACEXDATASET"
WHERE (landing_outcome = 'Failure (drone ship)') AND (EXTRACT(YEAR FROM Date) = '2015');
```

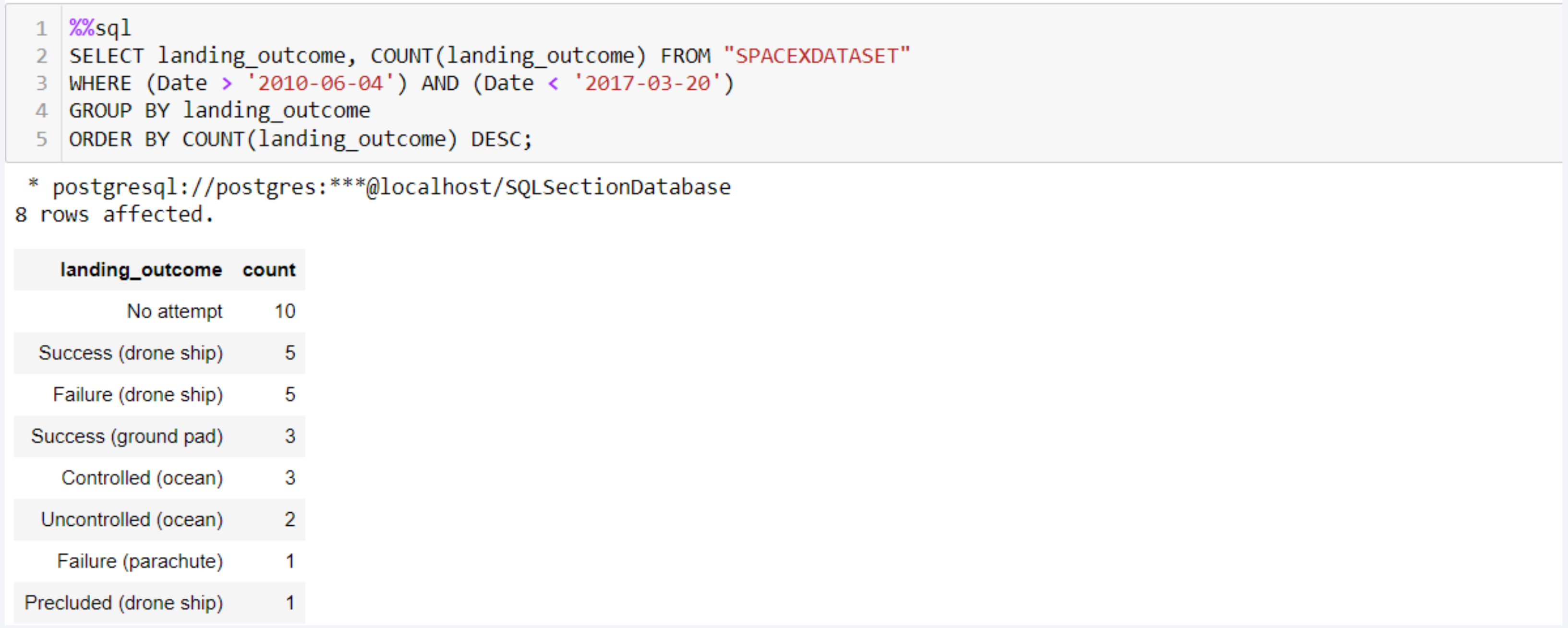
```
 * postgresql://postgres:***@localhost/SQLSectionDatabase
2 rows affected.
```

| date | landing_outcome | booster_version | launch_site |
|---|---|---|---|
| 2015-01-10 | Failure (drone ship) | F9 v1.1 B1012 | CCAFS LC-40 |
| 2015-04-14 | Failure (drone ship) | F9 v1.1 B1015 | CCAFS LC-40 |

As you can see, We got 2 times (in 2015) when landing outcome was failed on 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)) between the date 2010-06-04 and 2017-03-20, in descending order**



```
%%sql
SELECT landing_outcome, COUNT(landing_outcome) FROM "SPACEXDATASET"
WHERE (Date > '2010-06-04') AND (Date < '2017-03-20')
GROUP BY landing_outcome
ORDER BY COUNT(landing_outcome) DESC;
```

```
 * postgresql://postgres:***@localhost/SQLSectionDatabase
8 rows affected.
```

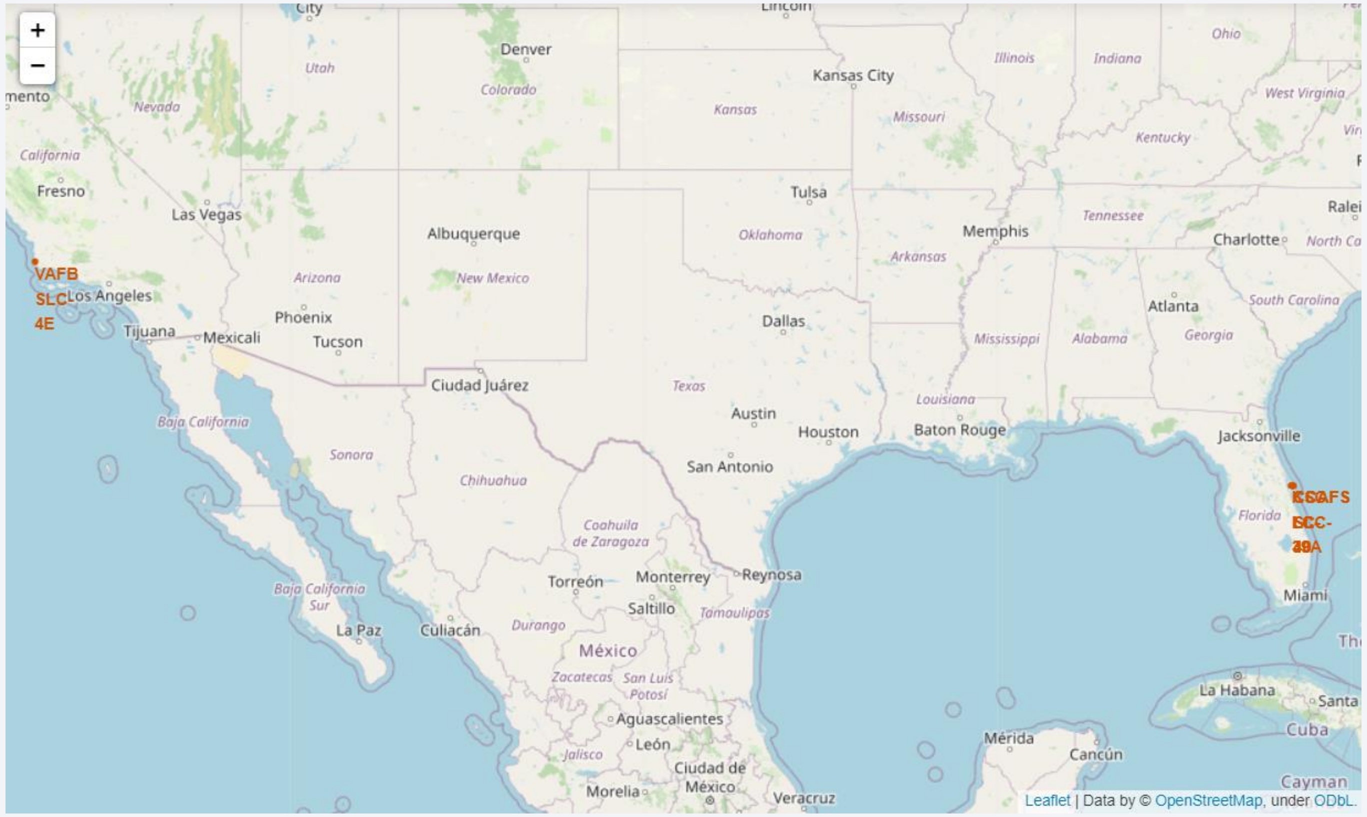
| landing_outcome | count |
|---|---|
| No attempt | 10 |
| Success (drone ship) | 5 |
| Failure (drone ship) | 5 |
| Success (ground pad) | 3 |
| Controlled (ocean) | 3 |
| Uncontrolled (ocean) | 2 |
| Failure (parachute) | 1 |
| Precluded (drone ship) | 1 |

As you can see, We got 10 times of No attempt and other landing outcomes that were ordered in descending order.

Section 4

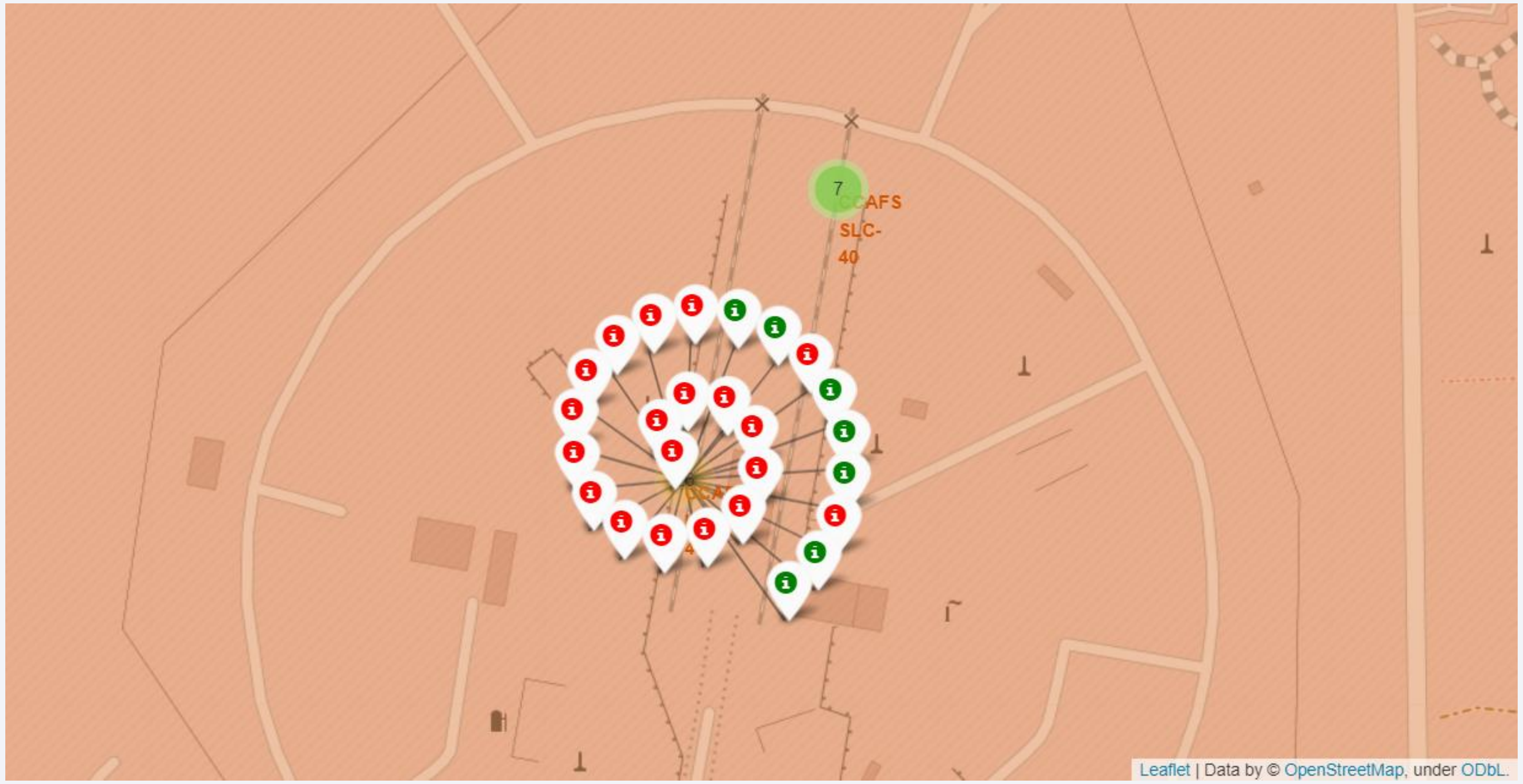
## Launch Sites Proximities Analysis

## All launch sites in Folium map



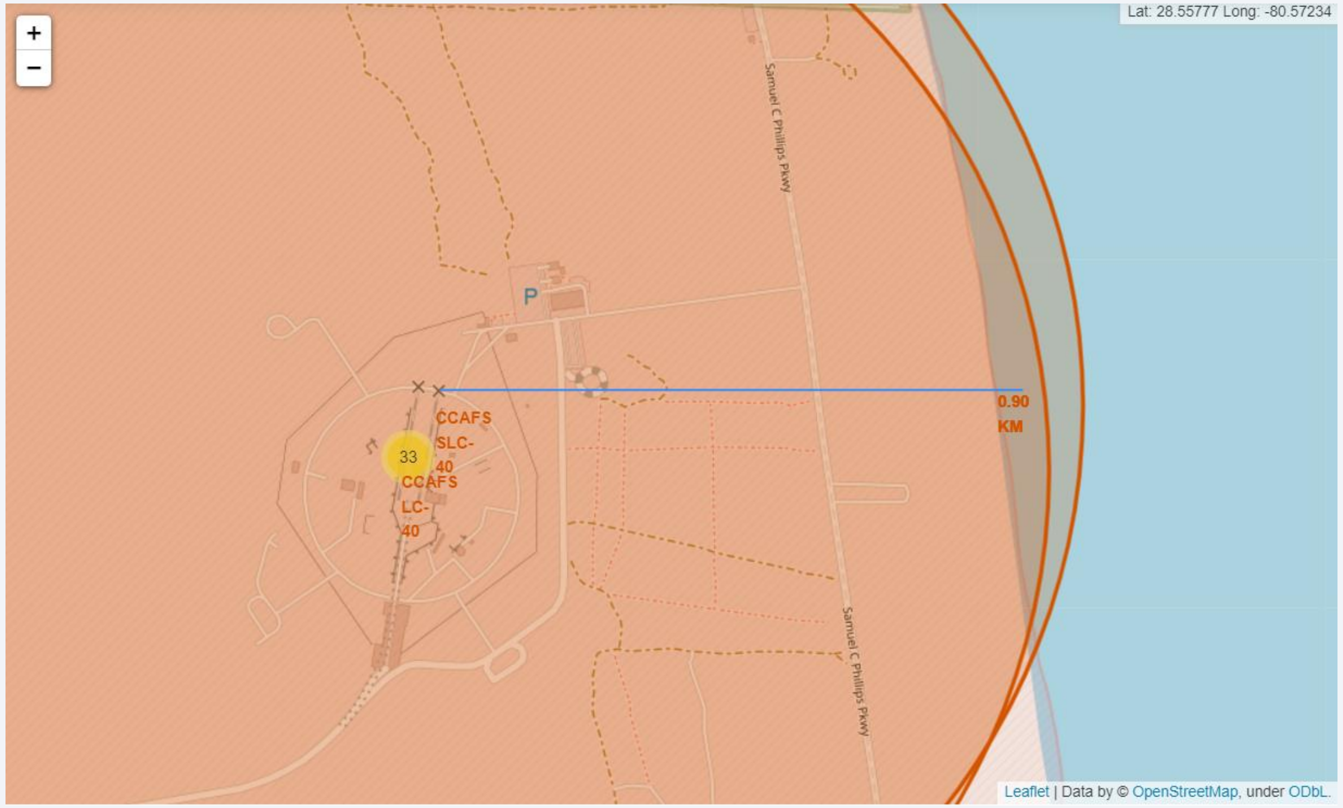
As you can see, all launch sites are in very close proximity to the coasts line in the USA, and all launch sites aren't in proximity to the equator line. (25 – 35 **°N**)

# Color-labeled launch outcomes

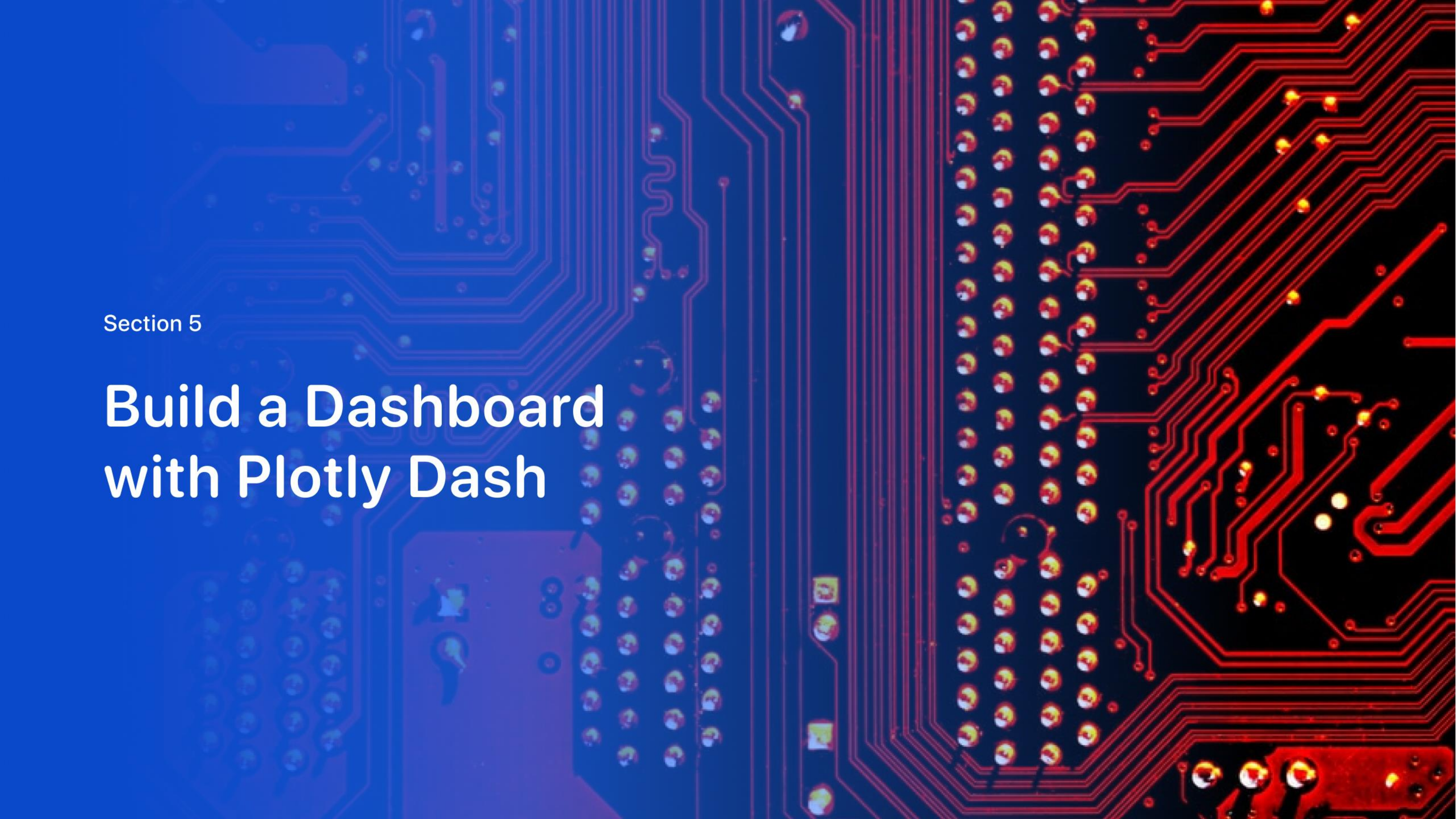


In this picture, It is the CCFAS LC-40 launch site while the color-labeled markers in marker clusters, Green Marker shows successful Launches and Red Marker shows failures.

## The distance between launch site and coastline


Lat: 28.55777 Long: -80.57234
0.90 KM
CCAFS SLC-40
33
CCAFS LC-40
Samuel C Phillips Pkwy
P
Leaflet | Data by © OpenStreetMap, under ODbL.

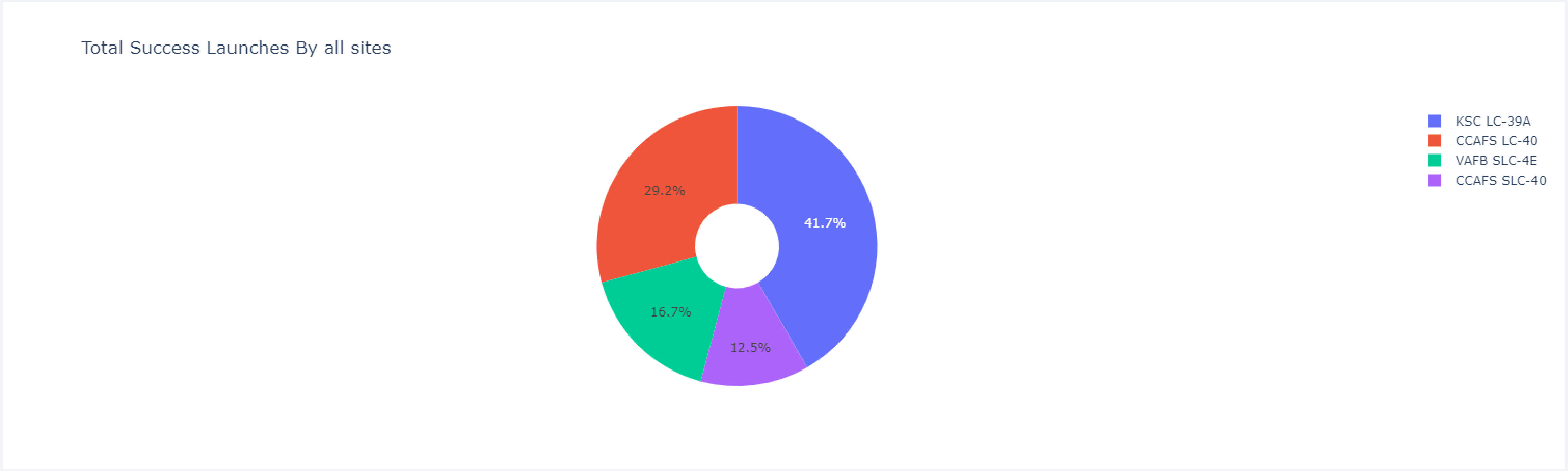
We can use the MousePosition and calculate the distance between the coastline point and the launch site. In this picture, it takes 0.9 km away from CCFAS LC-40 launch site.



Section 5

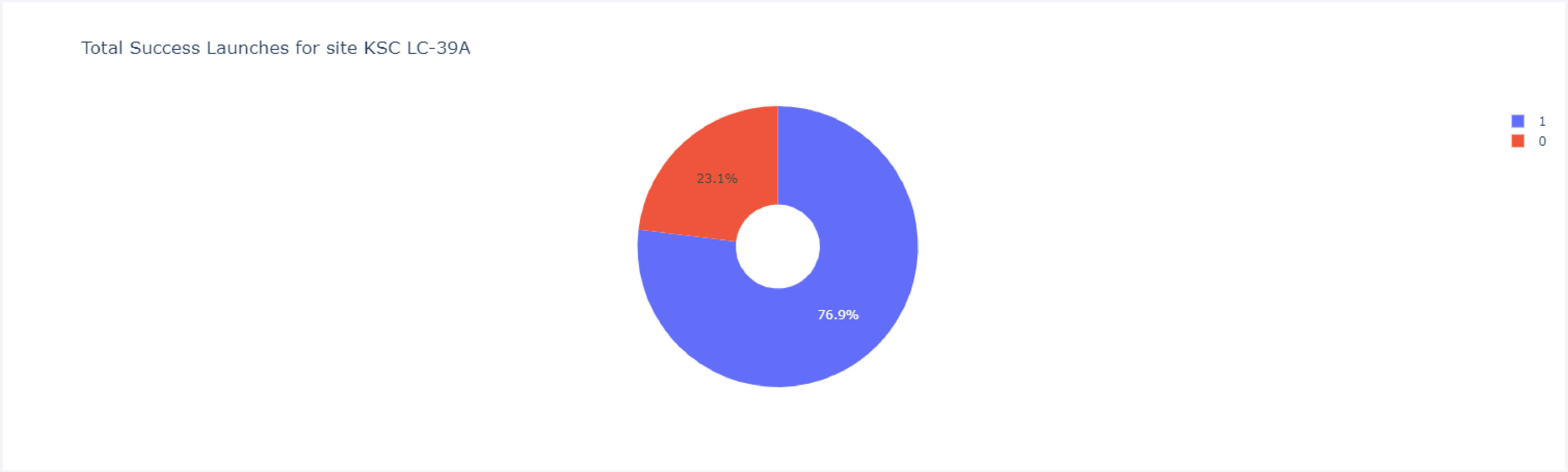
## Build a Dashboard with Plotly Dash

## Launch success count for all sites



We can see that KSC LC-39A had the most successful launches from all the sites. (41.7%)

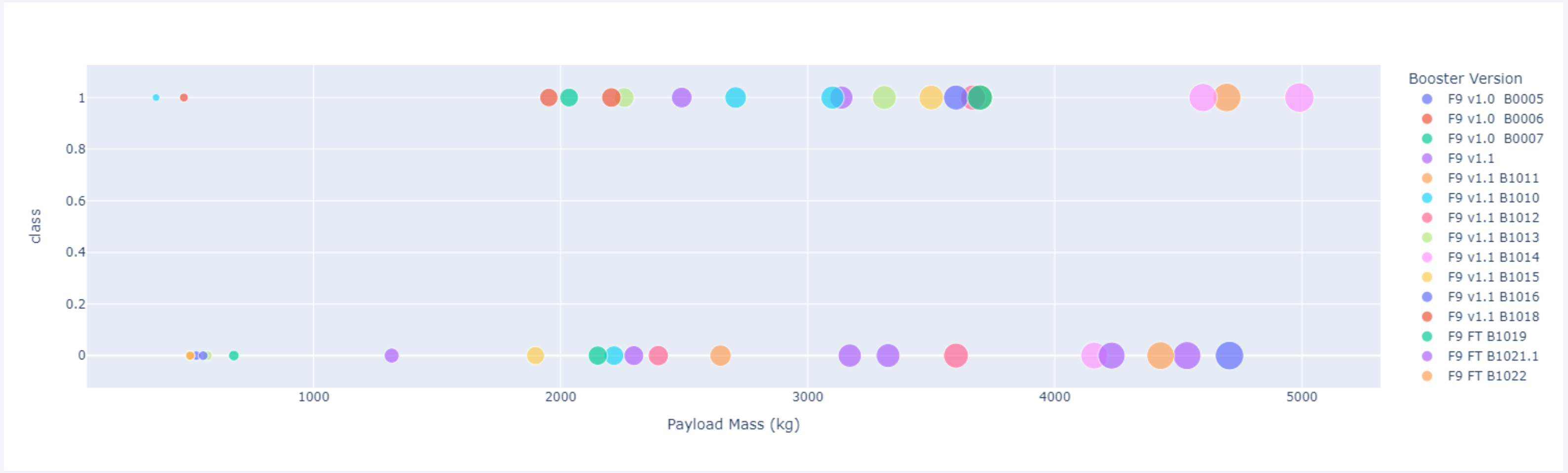
## The launch site with highest launch success ratio



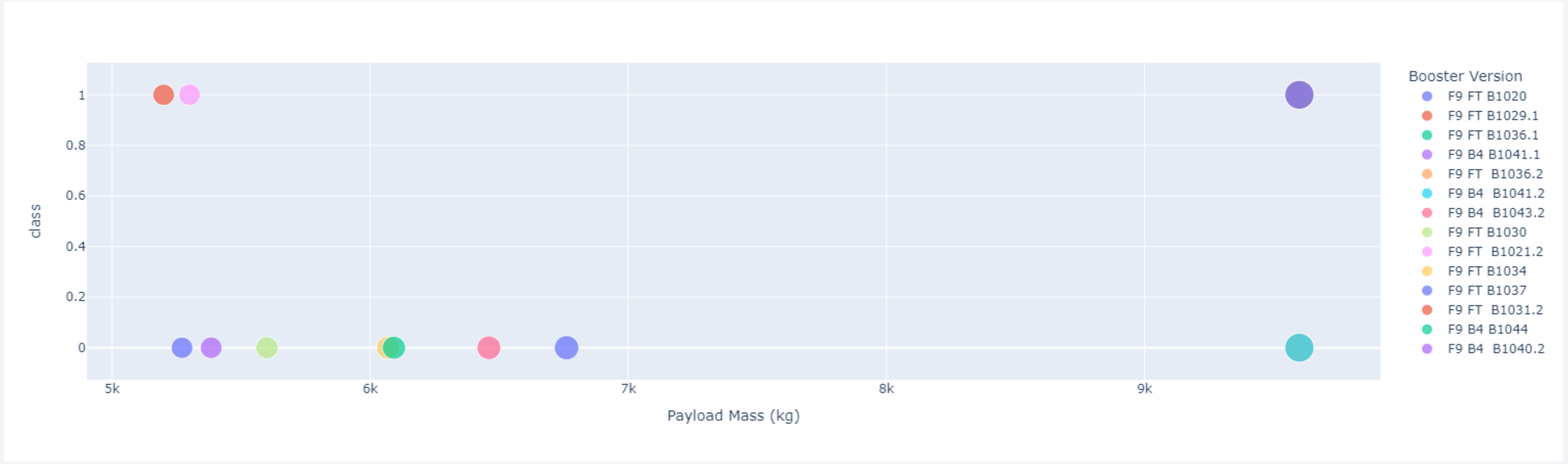
KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate.

# Payload vs. Launch Outcome scatter plot

## 0 – 5000 kg.



## 5000 – 10000 kg.



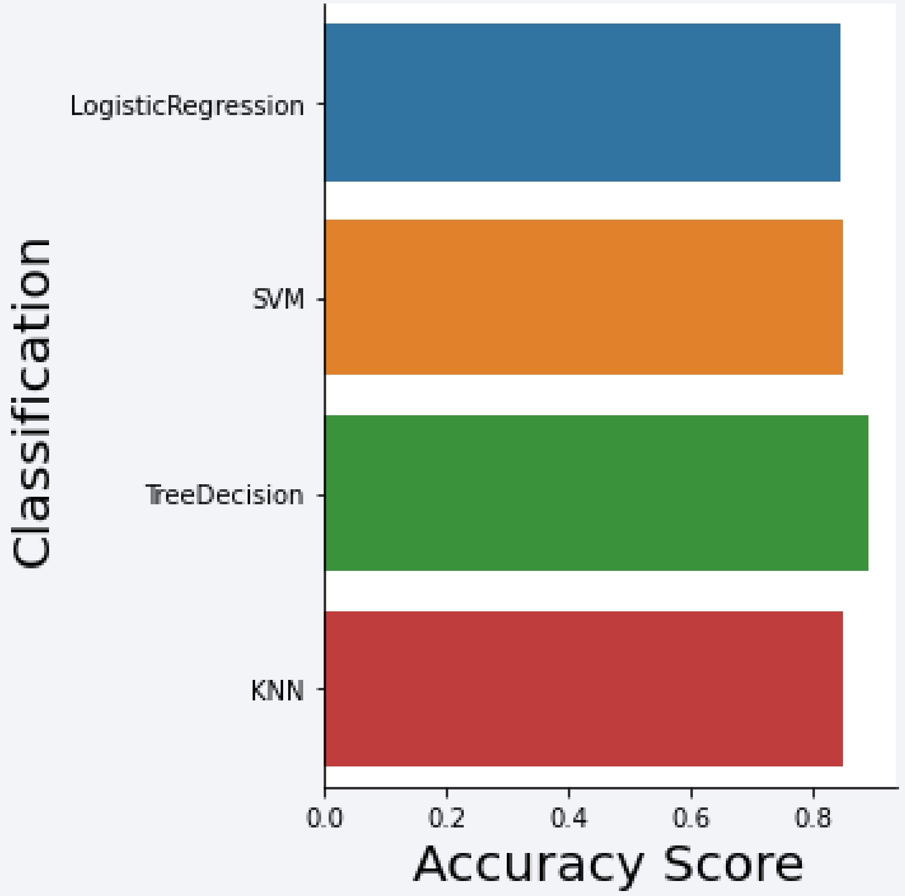
As you can see, the success rates for low weighted payloads is higher than the heavily weighted payloads.

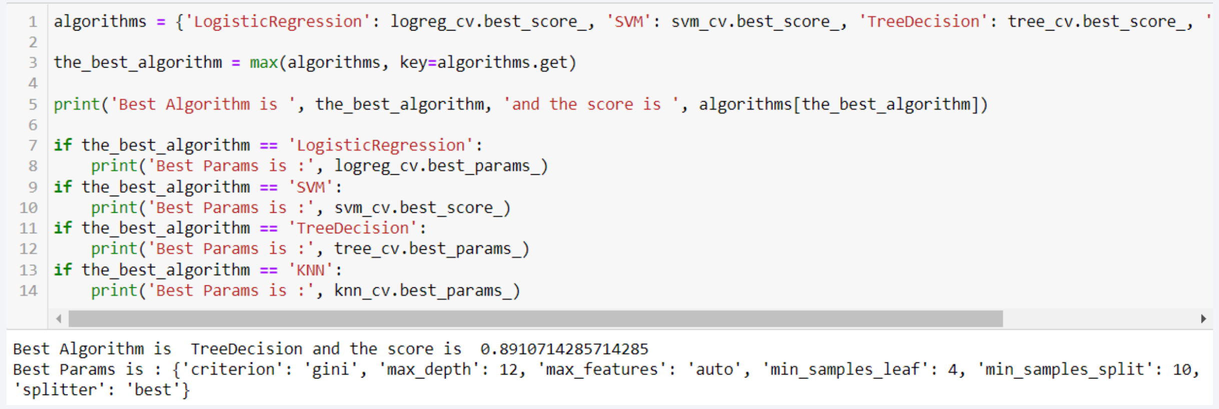
Section 6

## Predictive Analysis (Classification)

# Classification Accuracy

## Bar chart of all model accuracy





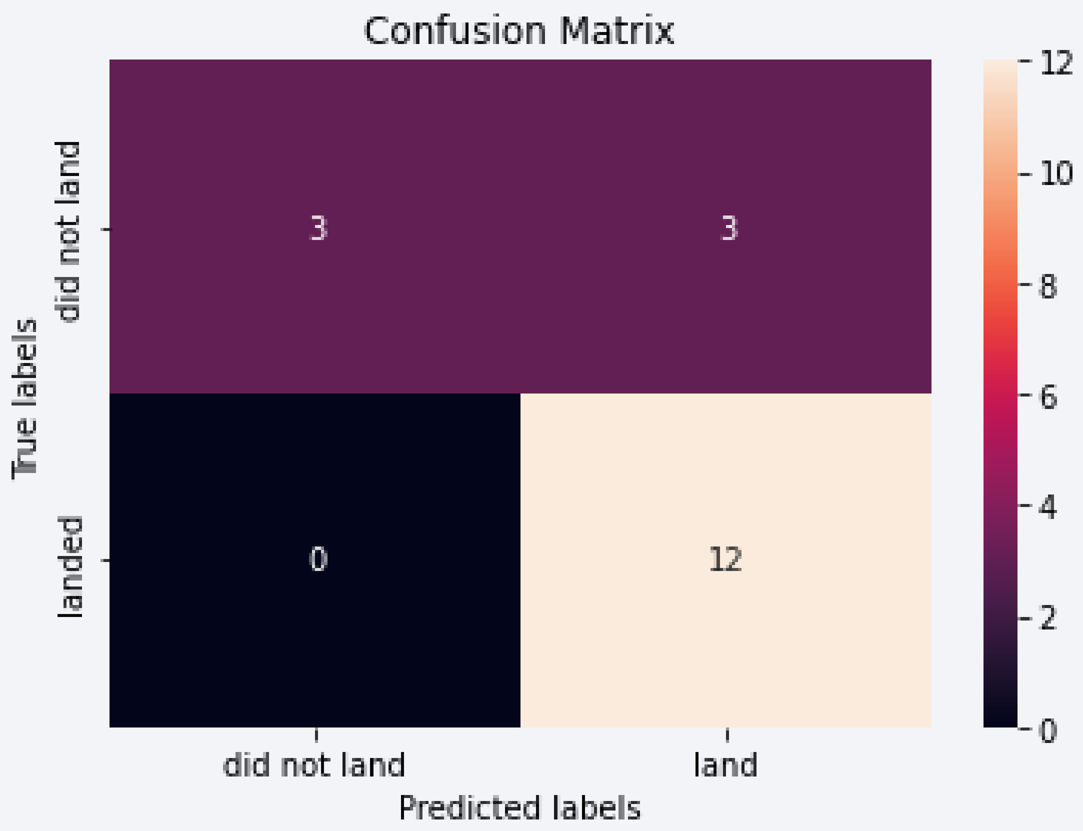
```
algorithms = {'LogisticRegression': logreg_cv.best_score_, 'SVM': svm_cv.best_score_, 'TreeDecision': tree_cv.best_score_, '

the_best_algorithm = max(algorithms, key=algorithms.get)

print('Best Algorithm is ', the_best_algorithm, 'and the score is ', algorithms[the_best_algorithm])

if the_best_algorithm == 'LogisticRegression':
    print('Best Params is :', logreg_cv.best_params_)
if the_best_algorithm == 'SVM':
    print('Best Params is :', svm_cv.best_score_)
if the_best_algorithm == 'TreeDecision':
    print('Best Params is :', tree_cv.best_params_)
if the_best_algorithm == 'KNN':
    print('Best Params is :', knn_cv.best_params_)
```

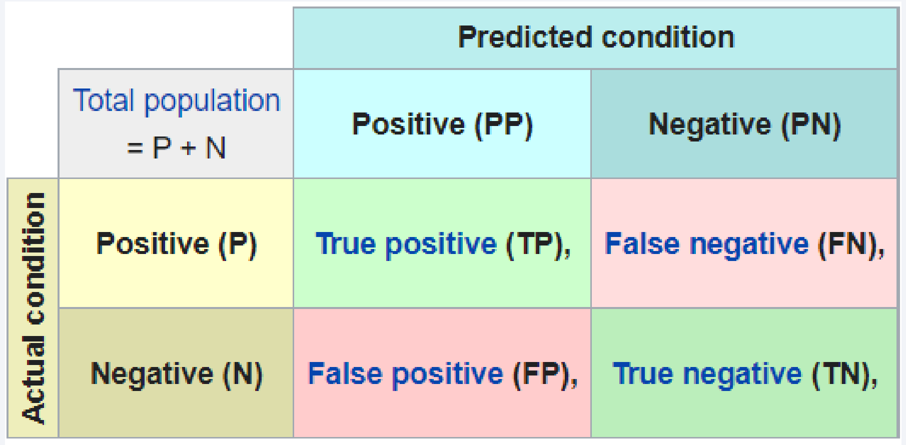
```
Best Algorithm is  TreeDecision and the score is  0.8910714285714285
Best Params is : {'criterion': 'gini', 'max_depth': 12, 'max_features': 'auto', 'min_samples_leaf': 4, 'min_samples_split': 10,
'splitter': 'best'}
```

The model that has the highest accuracy is **Decision Tree (0.891071)**

## Confusion Matrix

### Decision Tree's Confusion Matrix





| | | Predicted condition | |
|---|---|---|---|
| | Total population = P + N | Positive (PP) | Negative (PN) |
| Actual condition | Positive (P) | True positive (TP), | False negative (FN), |
| | Negative (N) | False positive (FP), | True negative (TN), |

Examining the confusion matrix, we see that Tree can distinguish between the different classes only landed. We see that the major problem are true positive and false negative.

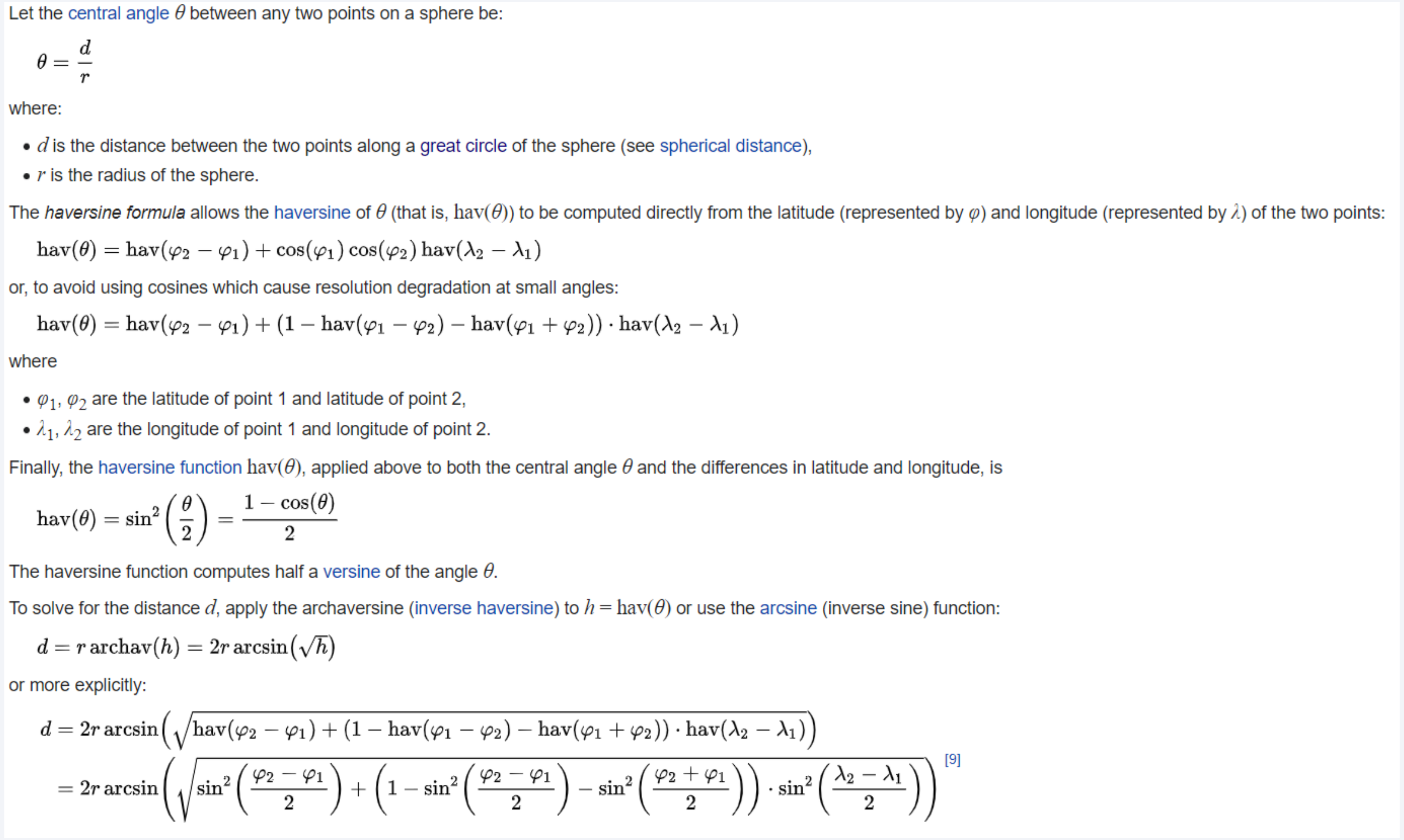
## Conclusions

- When increasing the flight number, the success rate increase in all launch site. In addition, CCAFS SLC 40 has the first successful landing at the less flight number.
- The Orbit ES-L1, SSO, HEO, and GEO have the best success rate.
- KSC LC-39A had the most successful launches from all the sites. (41.7% from all sites)
- The success rates for low weighted payloads is higher than the heavily weighted payloads.
- The Decision Tree Algorithm is the best for Machine Learning for this dataset which has an accuracy about 0.891.

# Appendix

**Haversine formula**

We use the formula to calculate the distance between two point on the earth.



Let the central angle $\theta$ between any two points on a sphere be:

$$\theta = \frac{d}{r}$$

where:

- $d$ is the distance between the two points along a great circle of the sphere (see spherical distance),
- $r$ is the radius of the sphere.

The *haversine formula* allows the haversine of $\theta$ (that is, $\operatorname{hav}(\theta)$) to be computed directly from the latitude (represented by $\varphi$) and longitude (represented by $\lambda$) of the two points:

$$\operatorname{hav}(\theta) = \operatorname{hav}(\varphi_2 - \varphi_1) + \cos(\varphi_1)\cos(\varphi_2)\operatorname{hav}(\lambda_2 - \lambda_1)$$

or, to avoid using cosines which cause resolution degradation at small angles:

$$\operatorname{hav}(\theta) = \operatorname{hav}(\varphi_2 - \varphi_1) + (1 - \operatorname{hav}(\varphi_1 - \varphi_2) - \operatorname{hav}(\varphi_1 + \varphi_2)) \cdot \operatorname{hav}(\lambda_2 - \lambda_1)$$

where

- $\varphi_1, \varphi_2$ are the latitude of point 1 and latitude of point 2,
- $\lambda_1, \lambda_2$ are the longitude of point 1 and longitude of point 2.

Finally, the haversine function $\operatorname{hav}(\theta)$, applied above to both the central angle $\theta$ and the differences in latitude and longitude, is

$$\operatorname{hav}(\theta) = \sin^2\left(\frac{\theta}{2}\right) = \frac{1 - \cos(\theta)}{2}$$

The haversine function computes half a versine of the angle $\theta$.

To solve for the distance $d$, apply the archaversine (inverse haversine) to $h = \operatorname{hav}(\theta)$ or use the arcsine (inverse sine) function:

$$d = r \operatorname{archav}(h) = 2r \arcsin\left(\sqrt{h}\right)$$

or more explicitly:

$$\begin{aligned} d &= 2r \arcsin\left(\sqrt{\operatorname{hav}(\varphi_2 - \varphi_1) + (1 - \operatorname{hav}(\varphi_1 - \varphi_2) - \operatorname{hav}(\varphi_1 + \varphi_2)) \cdot \operatorname{hav}(\lambda_2 - \lambda_1)}\right) \\ &= 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \left(1 - \sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) - \sin^2\left(\frac{\varphi_2 + \varphi_1}{2}\right)\right) \cdot \sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right) \end{aligned}$$ [9]

## Thank you!