IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Serkan TANRIVERDİ
April 12, 2023

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion
- Acknowledgements
- Appendix

Executive Summary

- The data was collected through the SpaceX public API and by web scrapping a Wikipedia article, the resulting dataset was then labeled adding a 'Class' field based on the landing outcome of the rockets, after that an Exploratory Data Analysis (EDA) was performed through SQL queries, visualizations using Python, Folium maps and Dash to create an interactive Dashboard. Through Feature Engineering the relevant columns were selected and later turned the categorical fields into binary variables by one-hot-encoding them. Finally four classification models (Logistic Regression, Support Vector Machine, Decision Tree, K-Nearest Neighbors) were trained using GridSearchCV from the scikit-learn library to predict the landing outcome.
- The four classification models got the same score on the test data set, 83.33%, similarly the Recall and Precision (for True positives) were the same for all of them, 100% and 80% respectively.

Introduction

- The 'SpaceY' company has hired a Data scientist to analyze the data from SpaceX, since it is planning on open their own space transportation company. SpaceX charges 62 million dollars per mission, whereas its competitors charge up to 165 million dollars per mission, most of the savings in cost come from the reutilization of their stage 1 rockets.
- 'SpaceY' needs a machine learning model to predict whether a rocket will land successfully or not, for this purpose, SpaceX's public available data will be used.
- Another useful task would be to perform an EDA of the SpaceX data to create valuable insights and answer questions like:
  - What's Orbits have a higher success rate?
  - What Booster Versions do better carrying specific payloads?
  - Which Machine Learning model will be selected and why?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - The dataset was collected from two main sources, through the SpaceX public API and by web scrapping a SpaceX Wikipedia article.
- Perform data wrangling
  - The dataset was cleaned and preprocessed to train the classification models.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Four different models were trained and tuned using GridSearchCV.

Data Collection

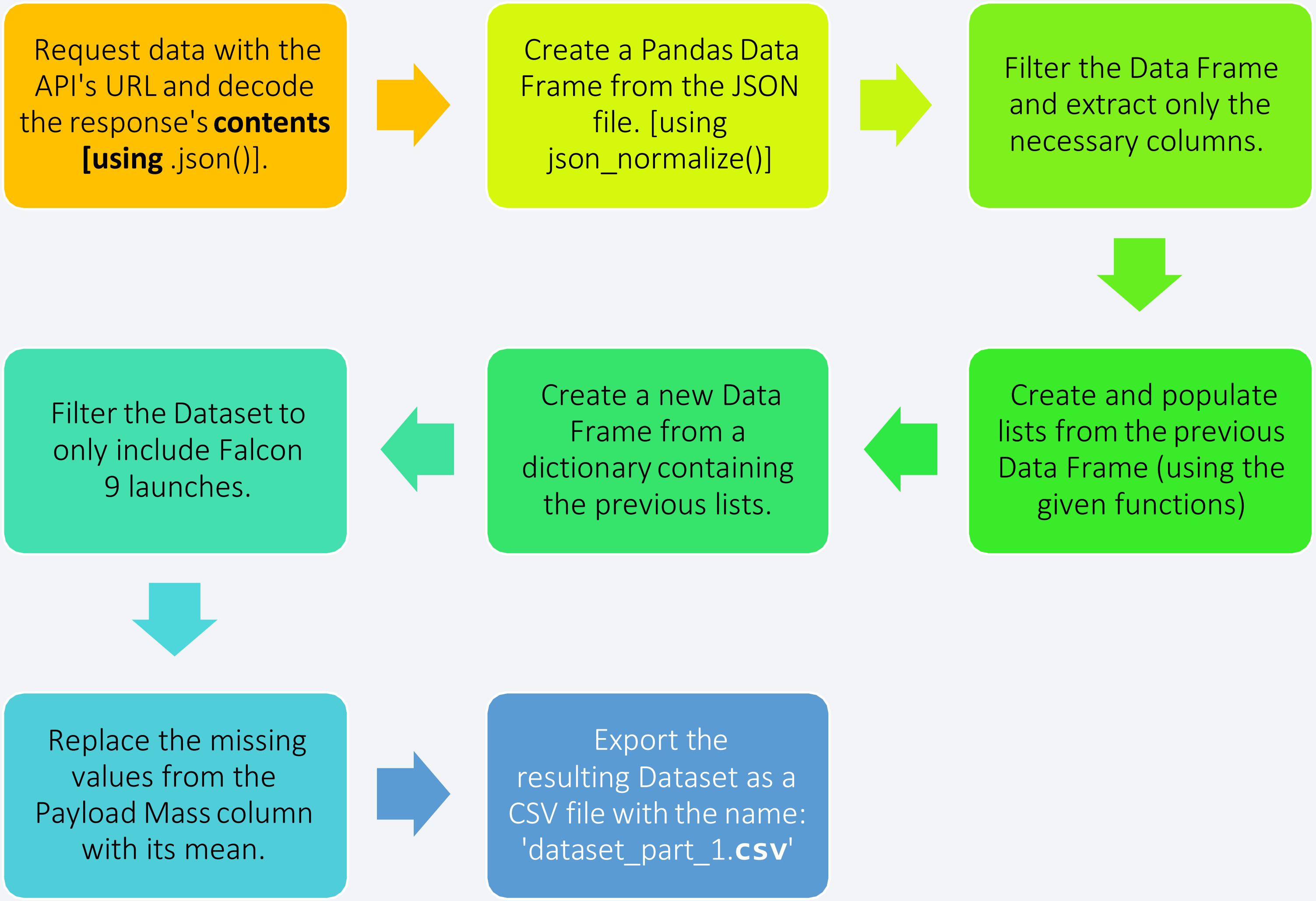
- The dataset was obtained by two means, by requesting the necessary data from the SpaceX public API and by web scrapping the tables on the SpaceX Wikipedia webpage.

- The requests library was used to get the data from the SpaceX API as a JSON file, after that the BeautifulSoup4 library was used to extract the data from the SpaceX Wikipedia page. Finally the data collected was stored in a Pandas Data Frame as a CSV file.

Data Collection - SpaceX API

- The next flowchart explains the steps taken to collect the data, access the Jupyter Notebook clicking the link below.

- Github: Data Collection through the SpaceX API.



Request data with the API's URL and decode the response's **contents [using** .json()].

Create a Pandas Data Frame from the JSON file. [using json_normalize()]

Filter the Data Frame and extract only the necessary columns.

Create and populate lists from the previous Data Frame (using the given functions)

Create a new Data Frame from a dictionary containing the previous lists.

Filter the Dataset to only include Falcon 9 launches.

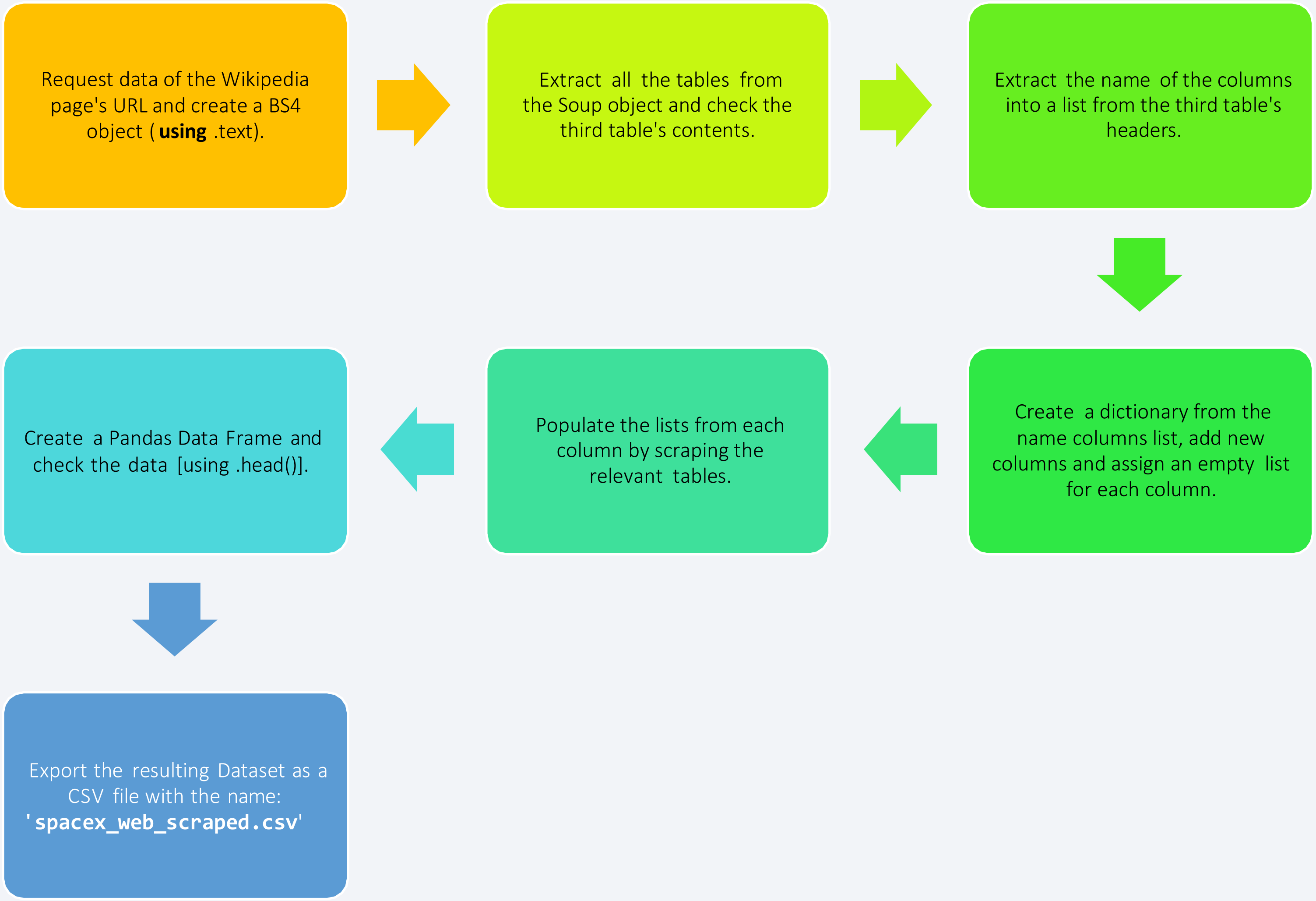
Replace the missing values from the Payload Mass column with its mean.

Export the resulting Dataset as a CSV file with the name: 'dataset_part_1.**csv**'

Data Collection - Scraping

- The following flowchart explains the process of collecting the data by web scraping, access the Jupyter Notebook clicking the link below.

- Github: Data Collection by Web Scraping.



Request data of the Wikipedia page's URL and create a BS4 object (**using** .text).

Extract all the tables from the Soup object and check the third table's contents.

Extract the name of the columns into a list from the third table's headers.

Create a dictionary from the name columns list, add new columns and assign an empty list for each column.

Populate the lists from each column by scraping the relevant tables.

Create a Pandas Data Frame and check the data [using .head()].

Export the resulting Dataset as a CSV file with the name: '**spacex_web_scraped.csv**'

Data Wrangling

- The main purpose of this stage was to label the data to later train the classification model, where 1 means a successful landing and 0 means and unsuccessful landing.

- First it was necessary to check the data types and null values for each column, after that, the amount of launches per launch site was calculated as well as the occurrence for each orbit.

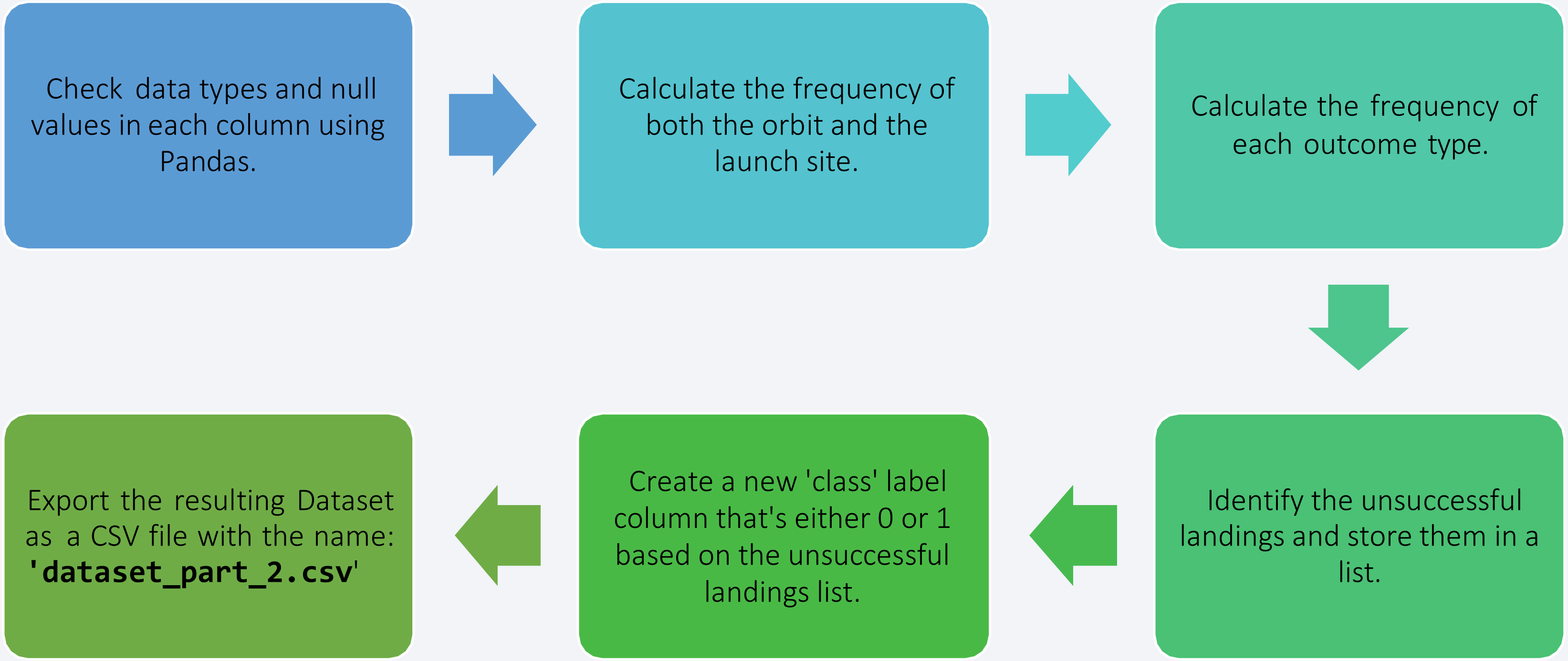
- The next step was to identify which outcomes were considered as unsuccessful landings and which were successful, the following were found:
  - Unsuccessful: 'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'.
  - Successful: 'True ASDS', 'True RTLS', 'True Ocean'.

- Finally, a list was created to store and later label each record as either 1 or 0 based on the step mentioned before.

Data Wrangling

- The next flowchart explains the approach taken, access the Jupyter Notebook clicking the link below.

- Github: Data Wrangling.



Check data types and null values in each column using Pandas.

Calculate the frequency of both the orbit and the launch site.

Calculate the frequency of each outcome type.

Identify the unsuccessful landings and store them in a list.

Create a new 'class' label column that's either 0 or 1 based on the unsuccessful landings list.

Export the resulting Dataset as a CSV file with the name: **'dataset_part_2.csv'**

EDA with Data Visualization

- The main types of chart used are scatter plots, bar plots, and line plots, these visualization were chosen because they teach us about the relationships between variables as well as with the landing success variable through the use of color.

- The scatter plots are:

  1. Flight Number vs Payload Mass (kg),
  2. Flight Number vs Launch Sites,
  3. Payload Mass (kg) vs Launch Sites,
  4. Flight Number vs Orbit,
  5. Payload Mass (kg) vs Orbit.

- The bar plot used is Success Rate per Orbit and lastly the line plot show us the trend of the Success Rate over the years.

- Github: EDA and Data Visualization.

EDA with SQL

1. Display the different Launch Sites.
2. Display 5 rows of the launch site beginning with 'CCA'.
3. Display the total Payload Mass carried for the NASA (CRS) customer.
4. Display the average Payload Mass carried by the F9 v1.1 booster.
5. Display the date of the first successful landing on a ground pad.
6. Display the boosters that had success landing on a drone ship and carried between 4000 and 6000 kg in Payload Mass.
7. Display the total number of success and failure outcomes.
8. Display the name of the boosters that carried the maximum Payload Mass.
9. Display the months of 2015, boosters, and Launch Sites that failed to land on a drone ship.
10. Rank the count of successful outcomes from 04-06-2010 to 20-03-2017.

Github: EDA with SQL.

Build an Interactive Map with Folium

- Circle objects were first created to show the launch sites on the map, the names of the maps were also added as markers.
- After that, icon markers were created, and clustered in a Marker Cluster object to show the landing outcomes and the amount of total launches per launch site.
- Finally polyline objects and other markers were created to draw lines to key places on the map, such as the closest coastline, highway, railroad, and city; the markers showed the distance in kilometers from the Launch site to each key place.
- GitHub: Launch site analysis with Folium
- Nbviewer: Launch site analysis with Folium (If the link above doesn't render the maps)

Build a Dashboard with Plotly Dash

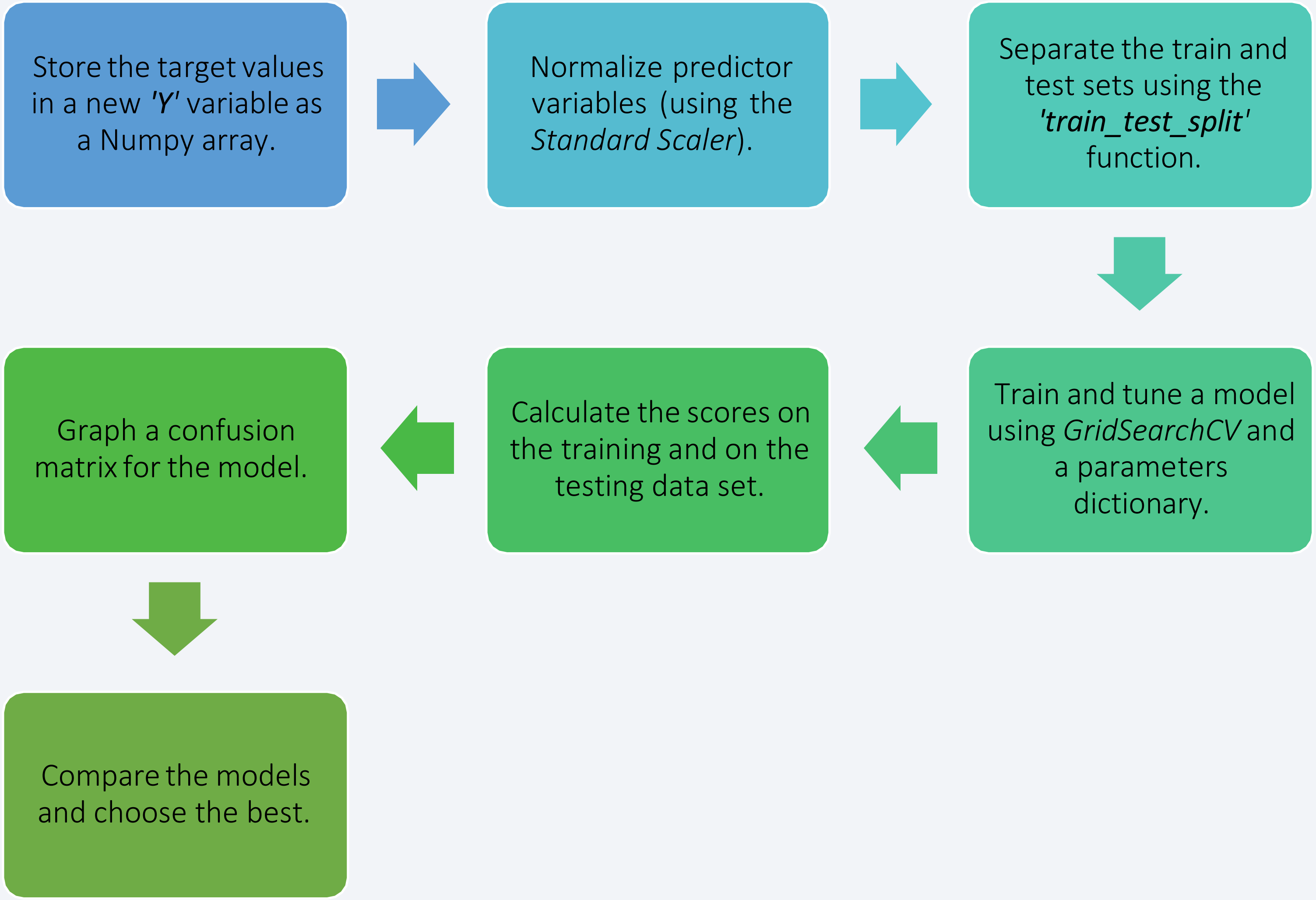
- The first plot is a pie chart that shows the overall success rate for each launch site, that is if you pick all sites, whereas if you pick any other specific launch site it shows you the success and failure rates for the chosen launch site, this graph helped us find the most and least successful launching site.
- The second plot build with dash is a scatter plot, it shows the relationship between Payload Mass (x-axis) and the Class or label of success (y-axis), and the boosters used, represented by the color of each data point, furthermore it uses a range that let the user customize the range of Payload Mass to display in the plot given all sites or a specific launch site, with this plot we were able to see which boosters are more effective for a given range of Payload Mass.
- GitHub: Plotly Dashboard.

Predictive Analysis (Classification)

- The first steps took were to preprocess the data to train the models, separating the predictors and target variables, standardizing (normalizing) the predictor variables, and then using the 'train_test_split' scikit-learn function to get the training and test sets.
- After that, the models were trained and tuned with the 'SearchGridCV' function and a parameters dictionary; the following four models were trained:
    1. Logistic Regression model.
    2. SVM model
    3. Decision Tree model
    4. KNN model
- It was found, after calculating their respective scores on the test sets and graphing a confusion matrix, that all four models performed equally well, that is a score of 83.33%, this is probably due to the small size of the dataset.

Predictive Analysis (Classification)

- The next flowchart explains steps done, access the Jupyter Notebook clicking the link below.
- GitHub: Model development for SpaceX.



Store the target values in a new *'Y'* variable as a Numpy array.

Normalize predictor variables (using the *Standard Scaler*).

Separate the train and test sets using the *'train_test_split'* function.

Train and tune a model using *GridSearchCV* and a parameters dictionary.

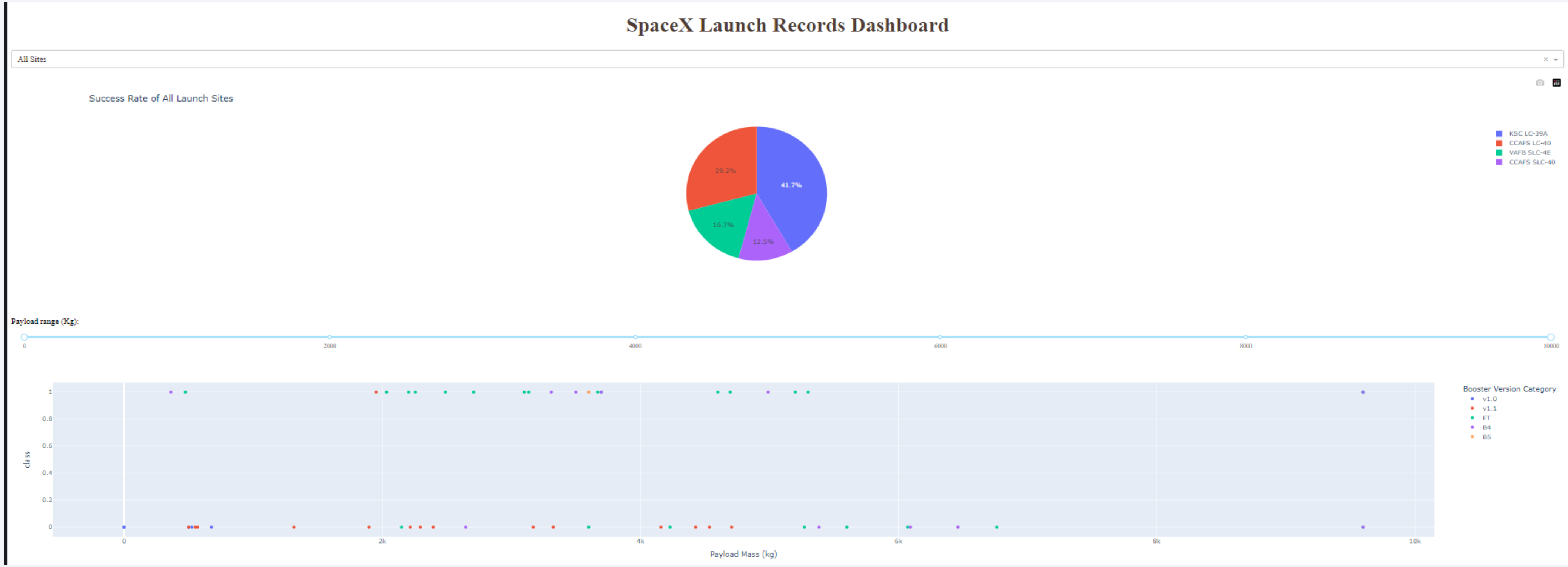
Calculate the scores on the training and on the testing data set.

Graph a confusion matrix for the model.

Compare the models and choose the best.

Results

- For the EDA many valuable insights were found, some of them are, the success rate over the years, the boosters that are more likely to succeed, and the success rate of the orbits.
- It was found that all classification models performed equally well, mainly because of the size of the dataset, and so to choose the best model other benefits were considered such as the estimated risk and the score on the training set.
- As for the Interactive Dashboard, here's a preview of how it looks like:



SpaceX Launch Records Dashboard

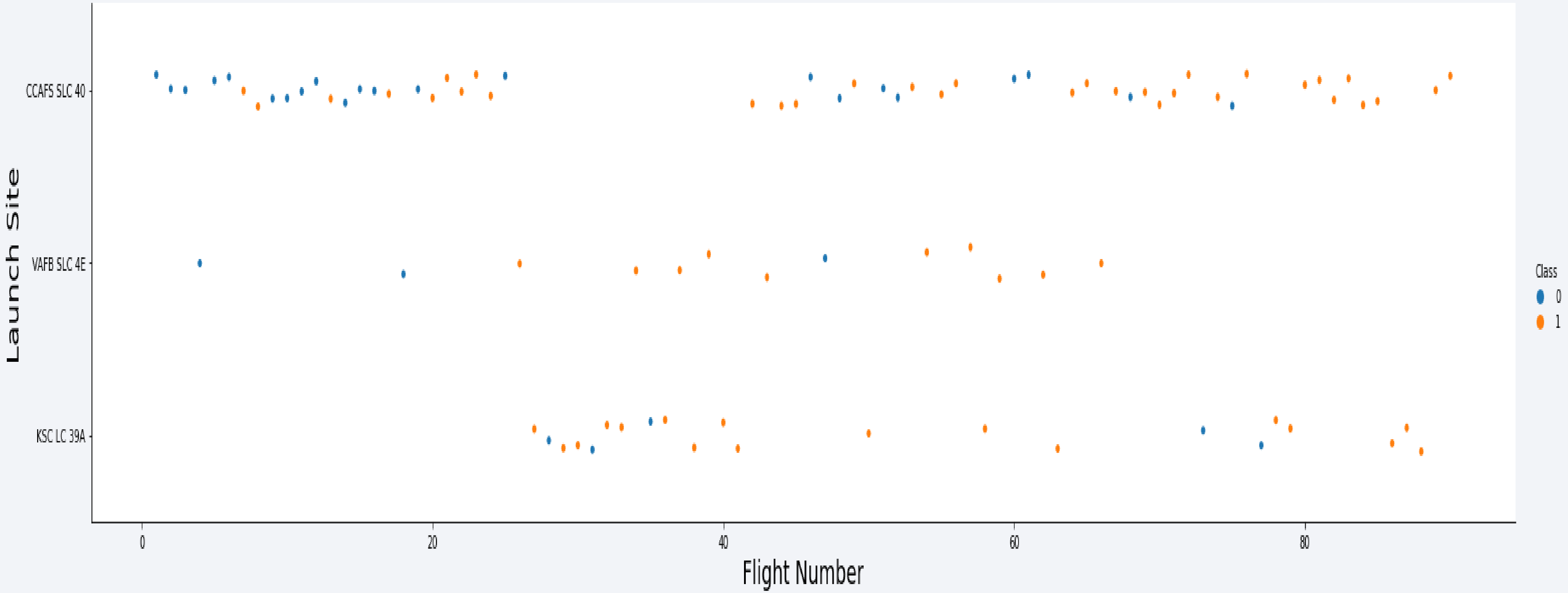
All Sites

Payload range (Kg):

Section 2

Insights drawn from EDA

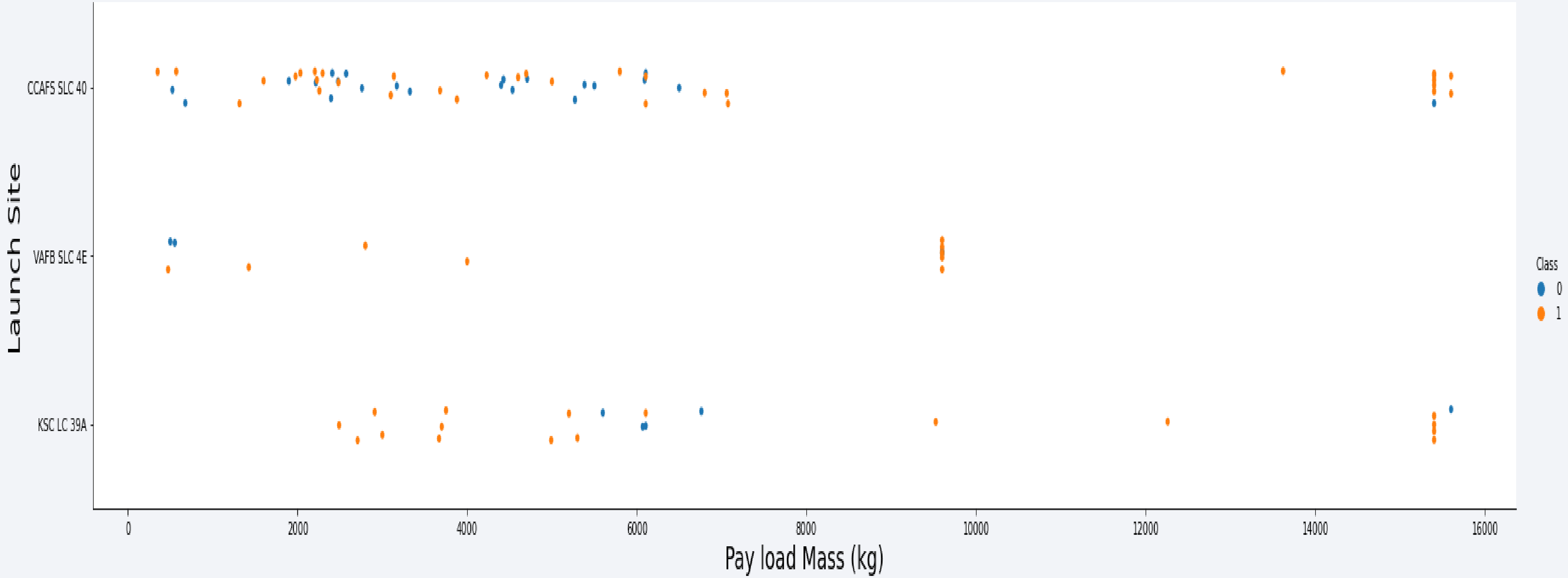
Flight Number vs. Launch Site



The graph shows the success landings as orange dots and failed ones as blue dots.

The graph shows that the main launch site is CCAFS SLC-40 and that since Flight number 20 the success rate has been increasing on all launch sites.

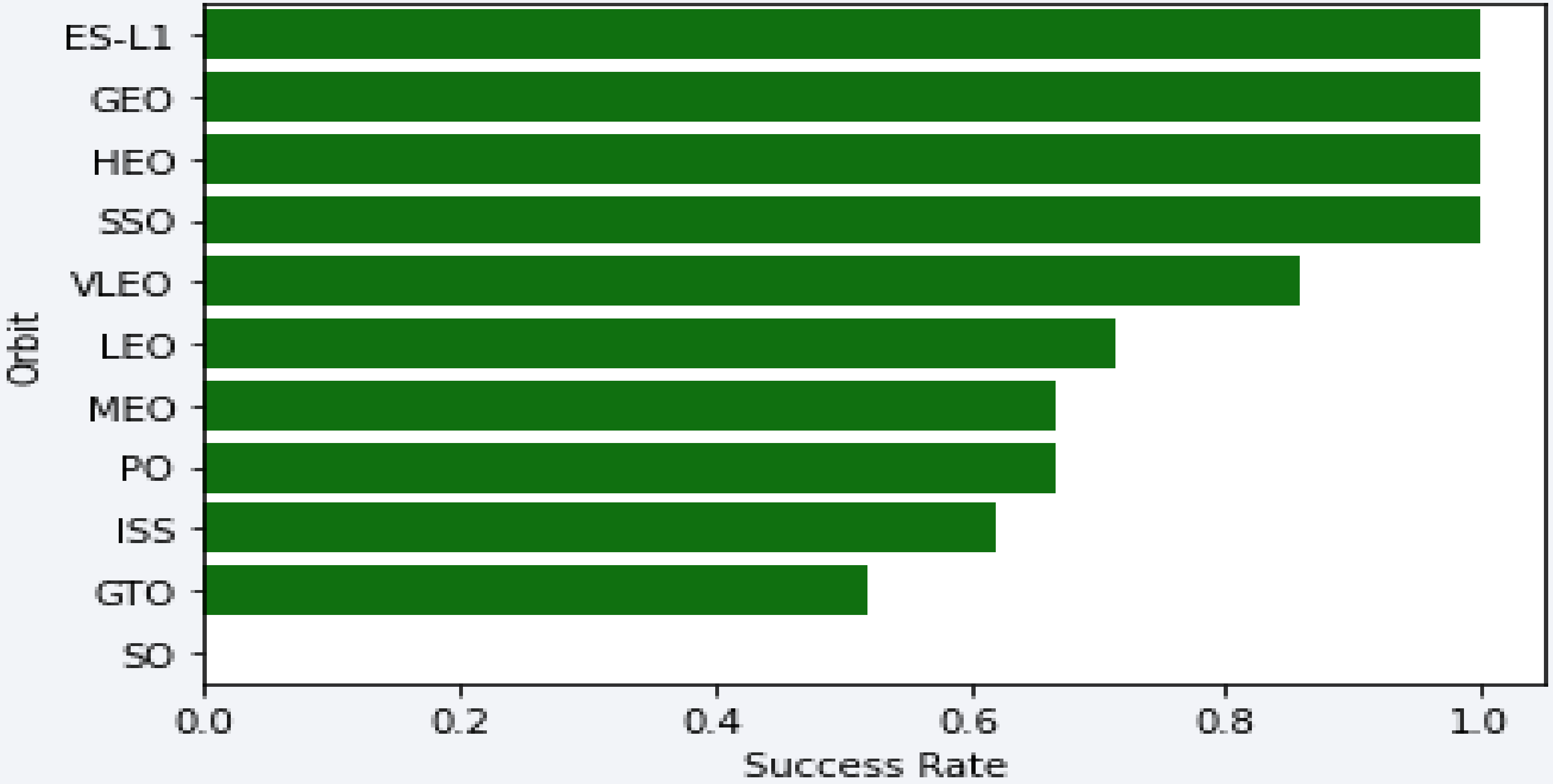
Payload vs. Launch Site



The graph shows the success landings as orange dots and failed ones as blue dots.

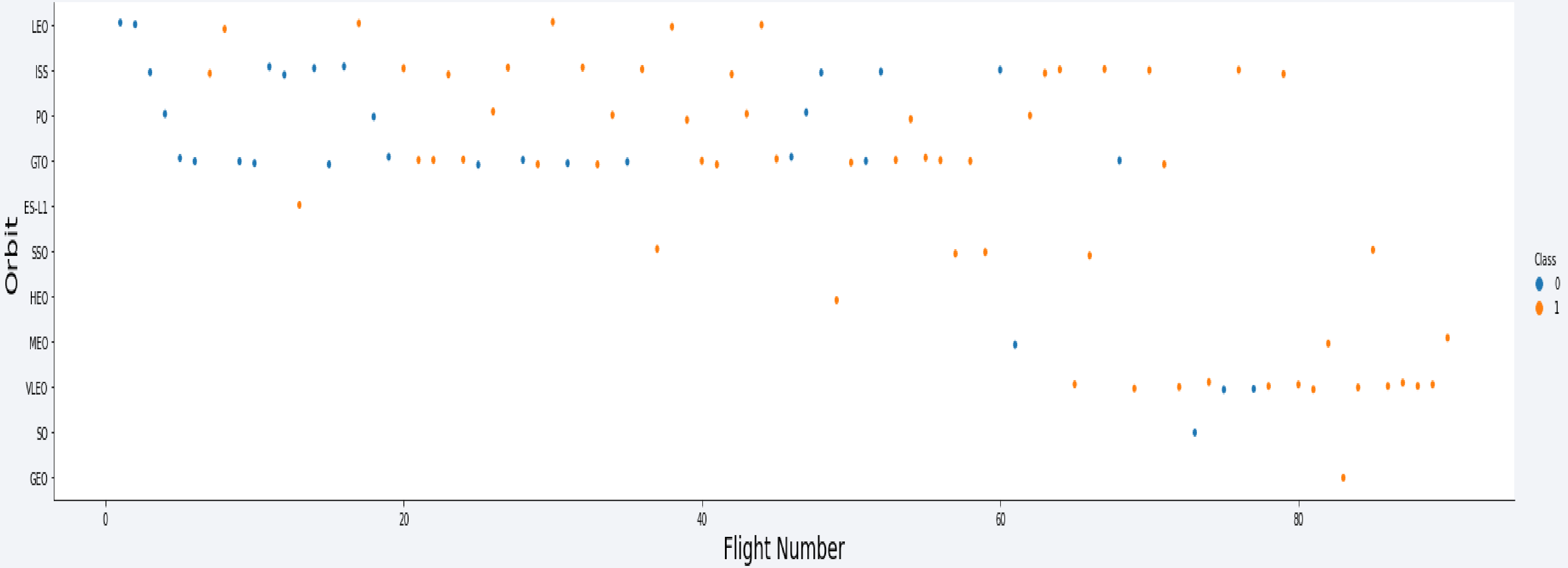
In this graph we see that the higher Payload Mass the higher the success rate, and that some launch sites have a better success rate for specific Payloads, VAFB SLC-4E does great with smaller Payloads.

Success Rate vs. Orbit Type



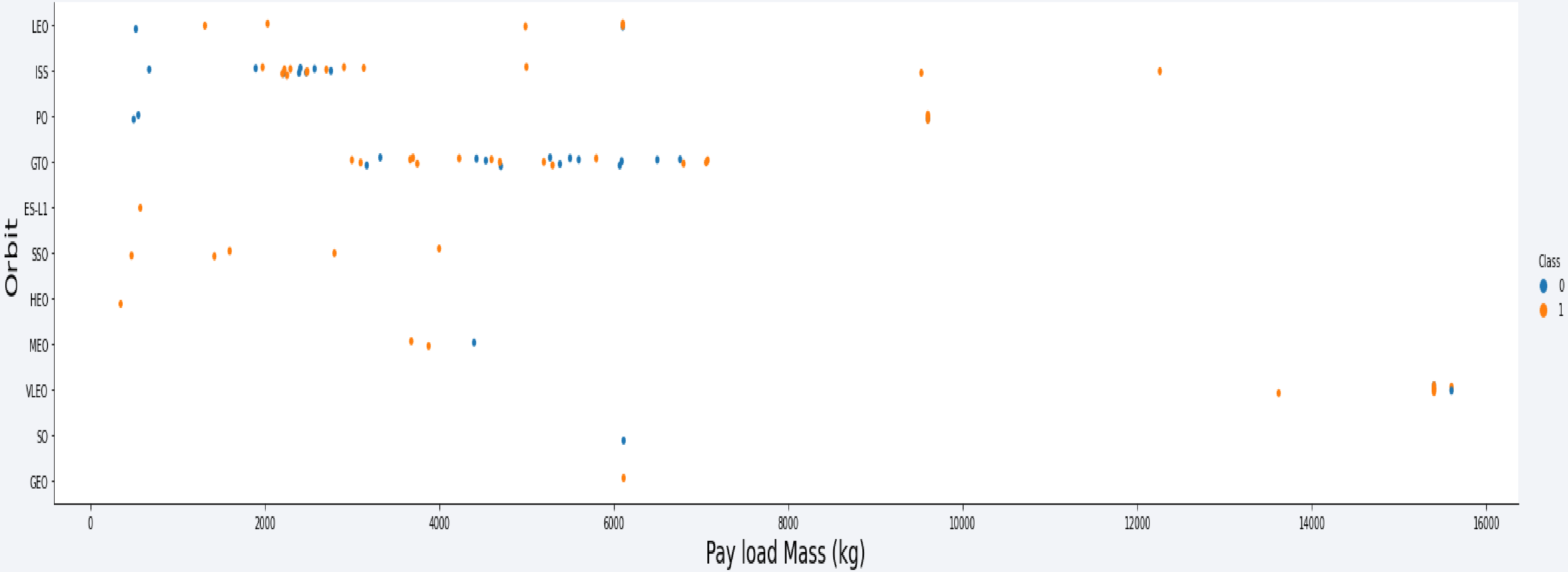
The bar plot shows that the boosters on missions to orbits such as ES-L1, GEO, HEO, SSO are guaranteed to land back successfully, meanwhile the worst orbit to send a rocket to is GTO with about 50% success rate.

Flight Number vs. Orbit Type



In this graph we can see that there may be a relationship between LEO and Flight number, as success increases with the Flight number, that doesn't seem to be true for GTO; another thing to notice is that orbits with the highest success rate have fewer missions that other orbits (ES-L1, GEO, SSO, HEO).

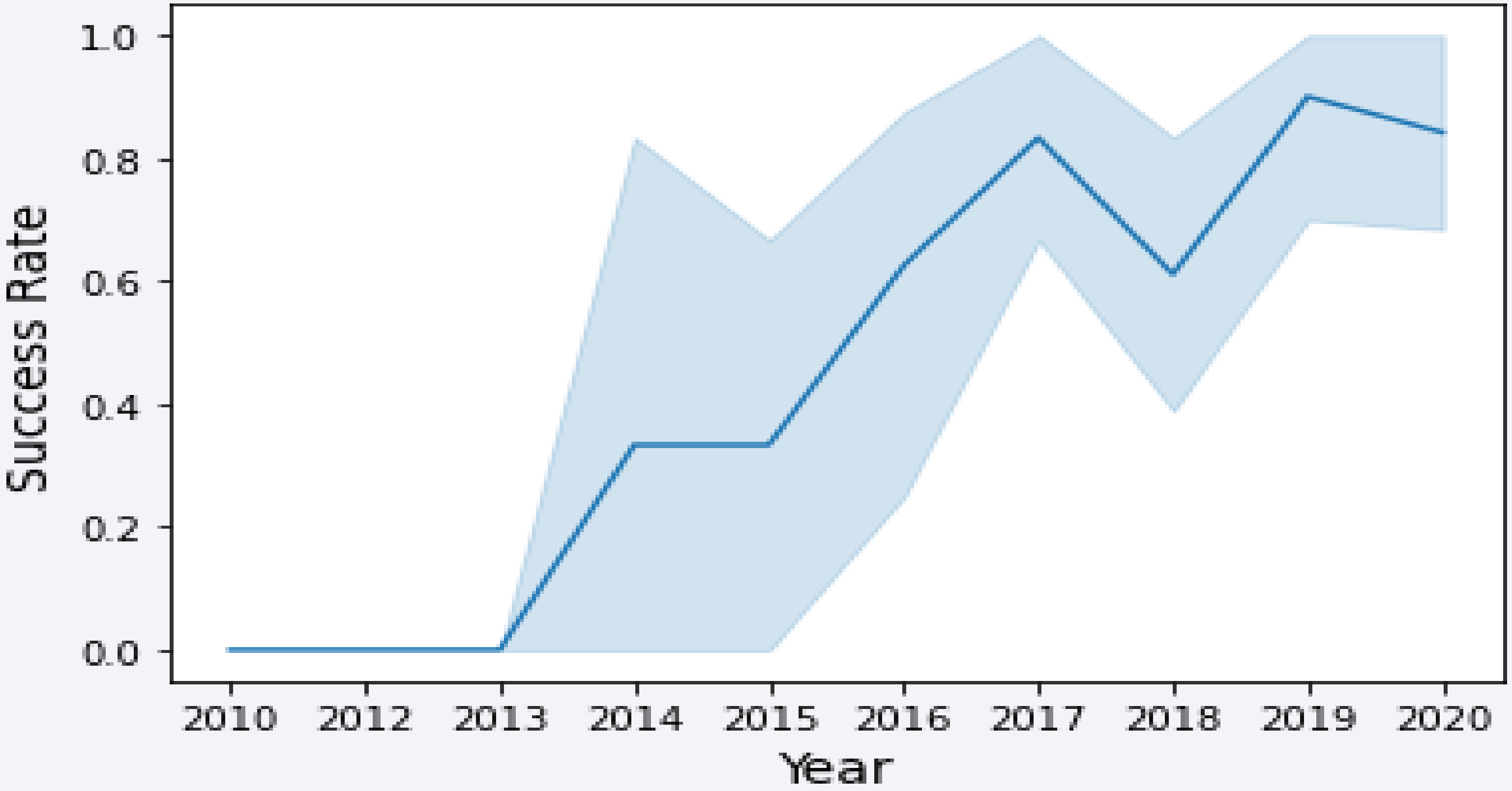
Payload vs. Orbit Type



Heavier Payloads seem to produce better outcomes for orbits like VLEO, LEO, ISS and PO.

On the other hand, orbits such as ES-L1, SSO, GEO and HEO seem to have a higher success rate for lighter Payloads.

Launch Success Yearly Trend



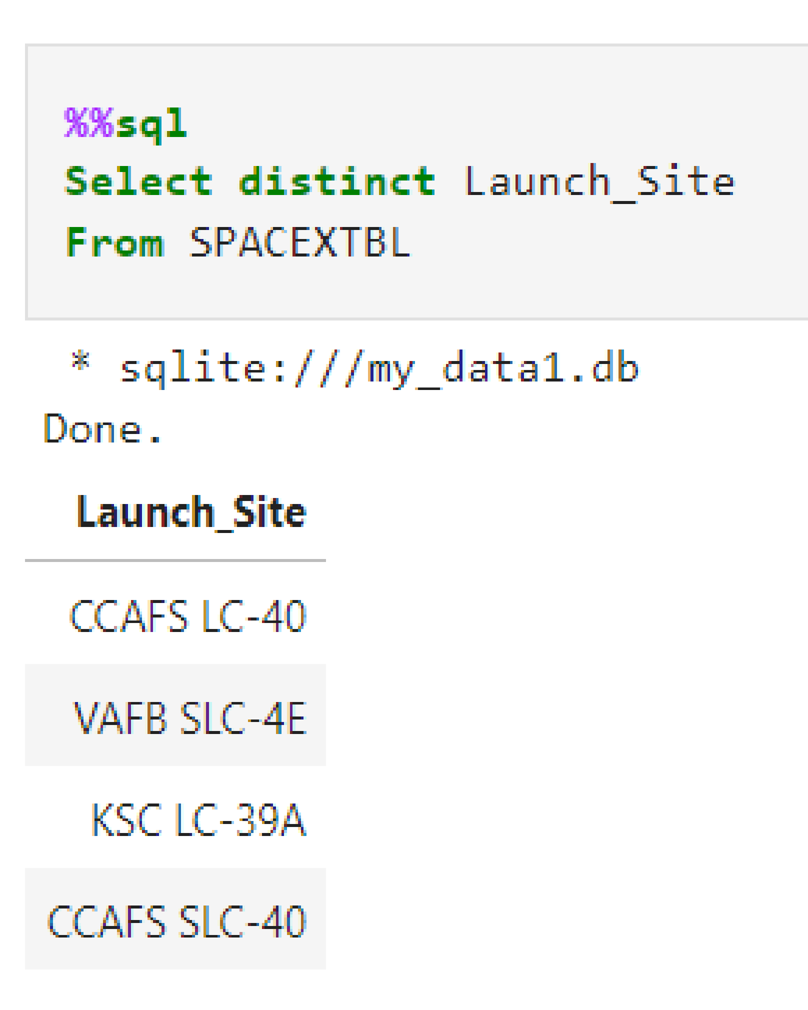
Success rate has been increasing over the years, the peak was in 2019 at about 90%.

Every year since 2013, except for 2018, the success rate increased.

All Launch Site Names

The next SQL query uses the Distinct clause to show the unique names for the launch sites.

As it be seen in the screenshot only four launch sites were found.



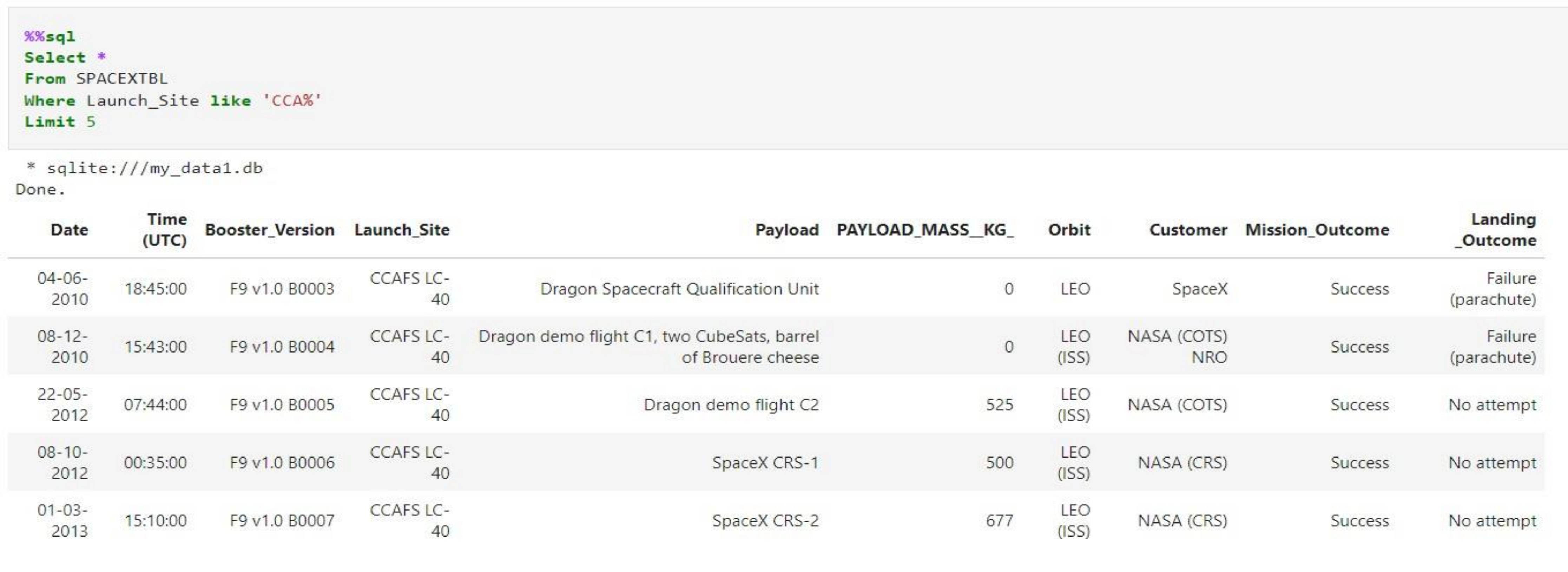
```
%%sql
Select distinct Launch_Site
From SPACEXTBL
```

```
 * 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'



```
%%sql
Select *
From SPACEXTBL
Where Launch_Site like 'CCA%'
Limit 5
```

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

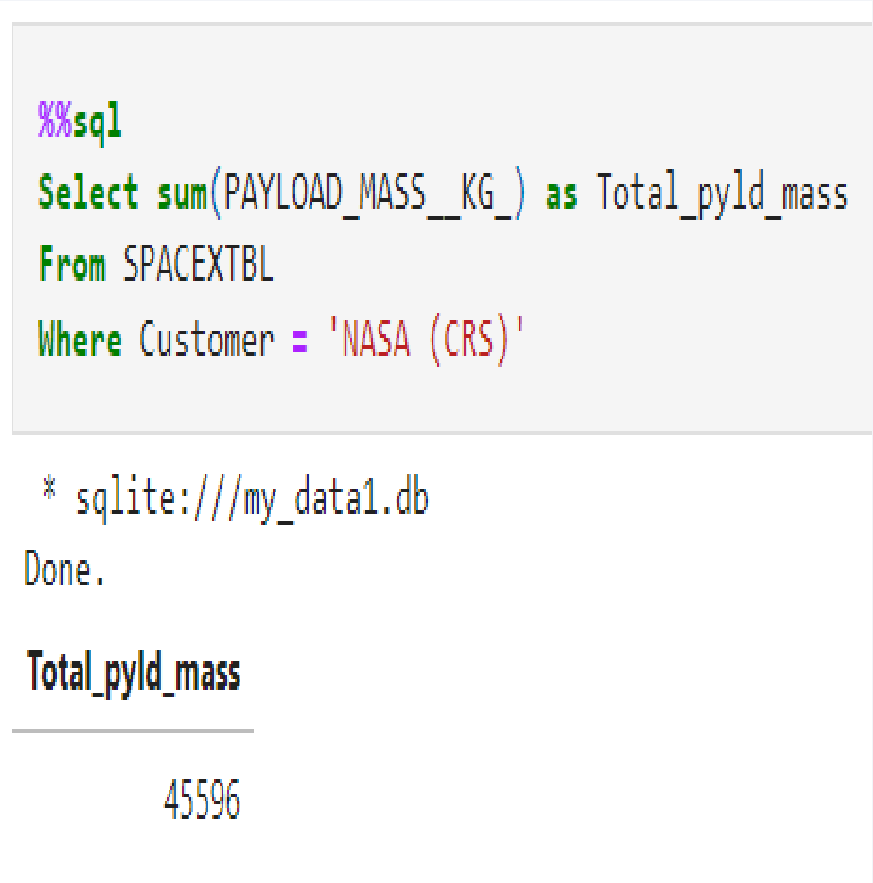
Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

The query displays all columns of the table, and then uses a wildcard in the where clause to filter only the data that starts with 'CCA', after that the limit clause shows only the first five records.

Total Payload Mass from NASA

The next query sums all the payload mass carried by SpaceX's rockets for the NASA customer.

The total payload mass found is 45596 kg.



```
%%sql
Select sum(PAYLOAD_MASS__KG_) as Total_pyld_mass
From SPACEXTBL
Where Customer = 'NASA (CRS)'
```

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

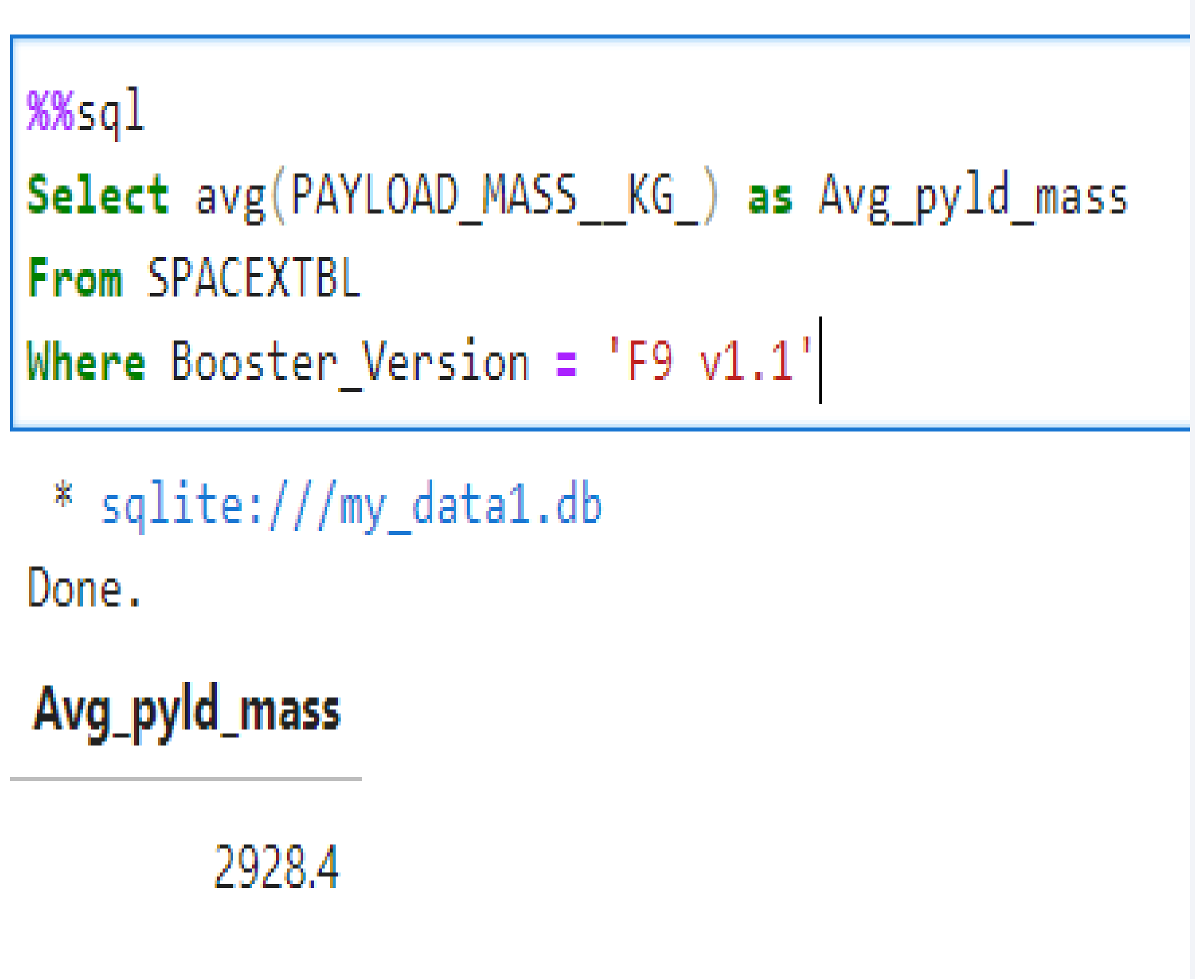
Total_pyld_mass
45596

Average Payload Mass by F9 v1.1

The next query calculates the average payload mass for the booster version F9 v1.1.

A wildcard was used later as well because it was noticed that some of them started with F9 v1.1 but had other codes following that.

The average payload found is 2928.4 kg, using the wildcard I found it to be 2534.67 kg.



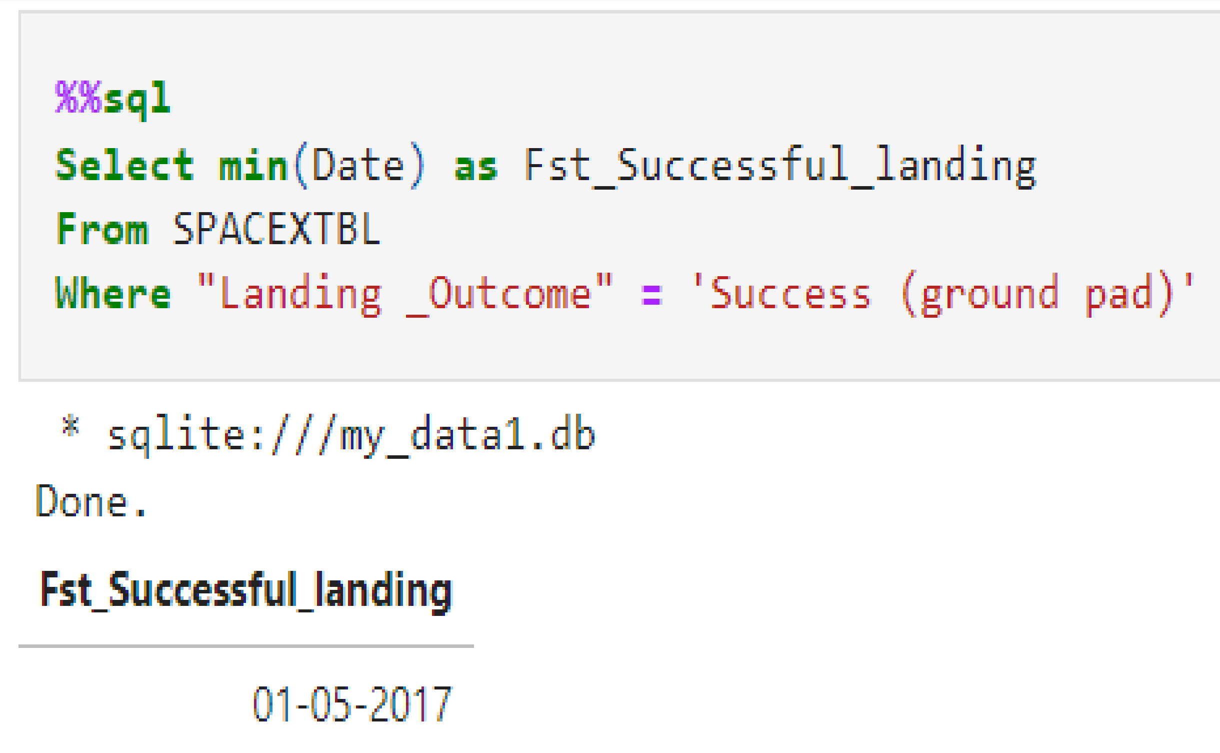
```
%%sql
Select avg(PAYLOAD_MASS__KG_) as Avg_pyld_mass
From SPACEXTBL
Where Booster_Version = 'F9 v1.1'
```

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

Avg_pyld_mass
2928.4

First Successful Ground Landing Date

Using the minimum function on the Date column, and then filtering only the Success on ground pads using a where clause it was found that the date of the first successful landing on a ground pad is the May 1st, 2017.



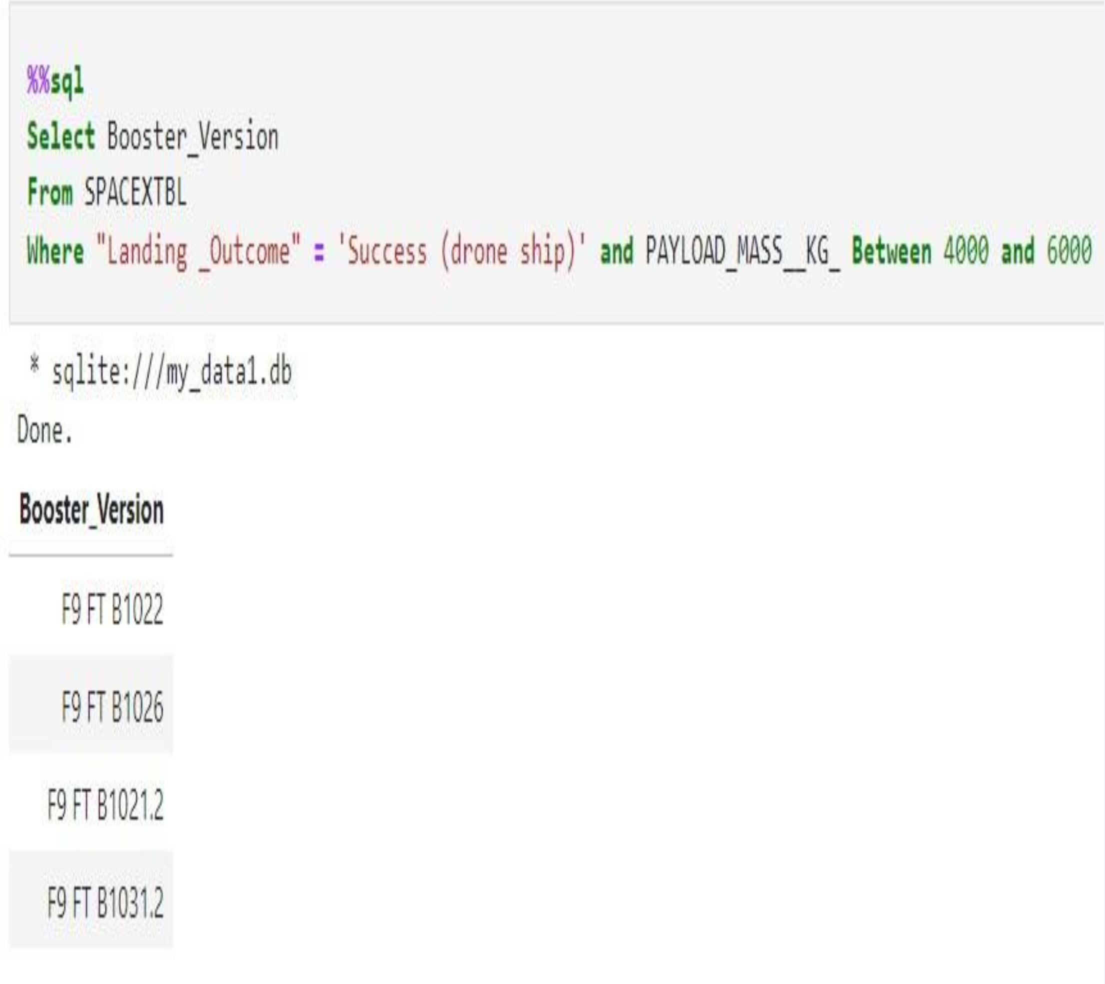
```
%%sql
Select min(Date) as Fst_Successful_landing
From SPACEXTBL
Where "Landing _Outcome" = 'Success (ground pad)'
```

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

Fst_Successful_landing
01-05-2017

Successful Drone Ship Landing with Payload between 4000 and 6000

The next query picks the Booster version column and then filters the data to find those boosters which landed successfully on a drone ship and were carrying between 4000 and 6000 kg in payload mass.



```
%%sql
Select Booster_Version
From SPACEXTBL
Where "Landing _Outcome" = 'Success (drone ship)' and PAYLOAD_MASS__KG_ Between 4000 and 6000
```

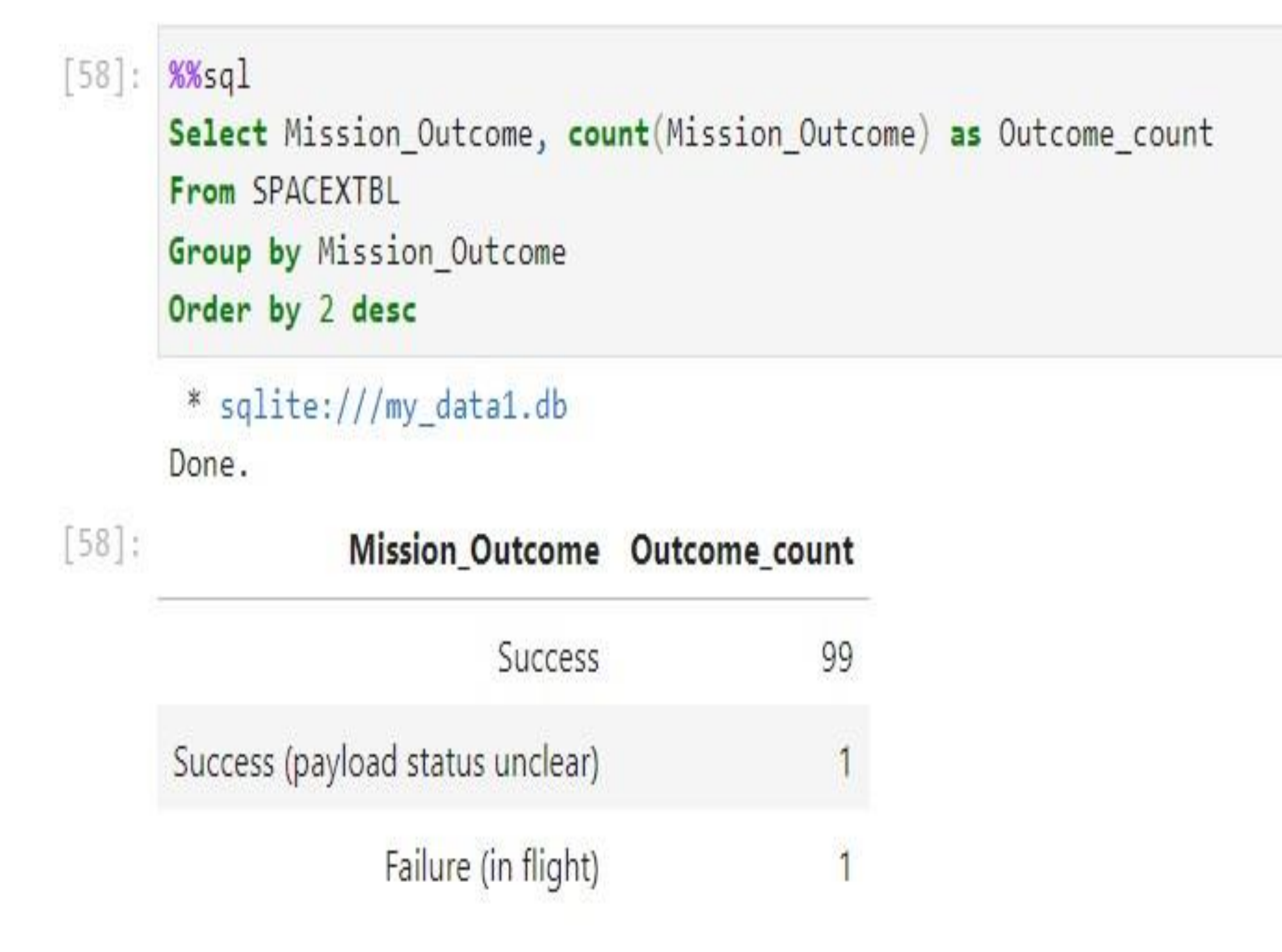
```
 * sqlite:///my_data1.db
Done.
```

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

Total Number of Successful and Failure Mission Outcomes

It was found that only 1 mission failed out of 101 missions on the database.

The vast majority are successful missions, with just one having an unclear payload status



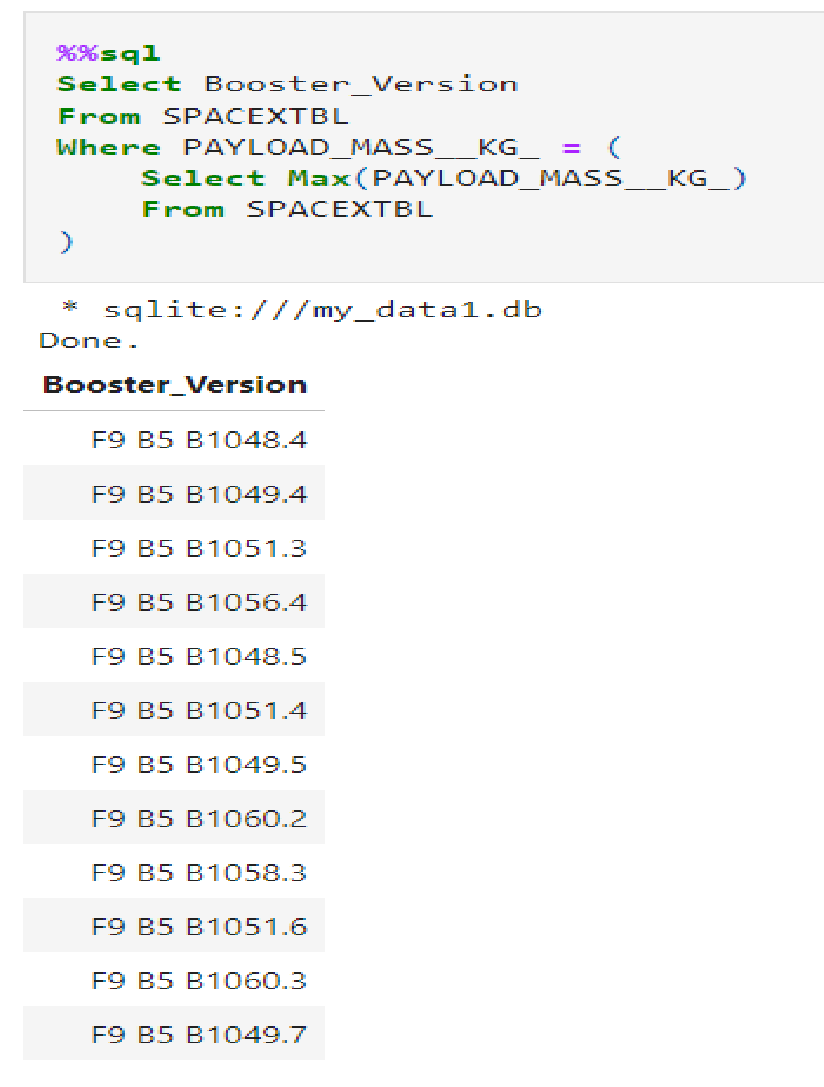
```
[58]: %%sql
      Select Mission_Outcome, count(Mission_Outcome) as Outcome_count
      From SPACEXTBL
      Group by Mission_Outcome
      Order by 2 desc

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

[58]:

Mission_Outcome	Outcome_count
Success	99
Success (payload status unclear)	1
Failure (in flight)	1

Boosters Carried Maximum Payload



```
%%sql
Select Booster_Version
From SPACEXTBL
Where PAYLOAD_MASS__KG_ = (
    Select Max(PAYLOAD_MASS__KG_)
    From SPACEXTBL
)
```

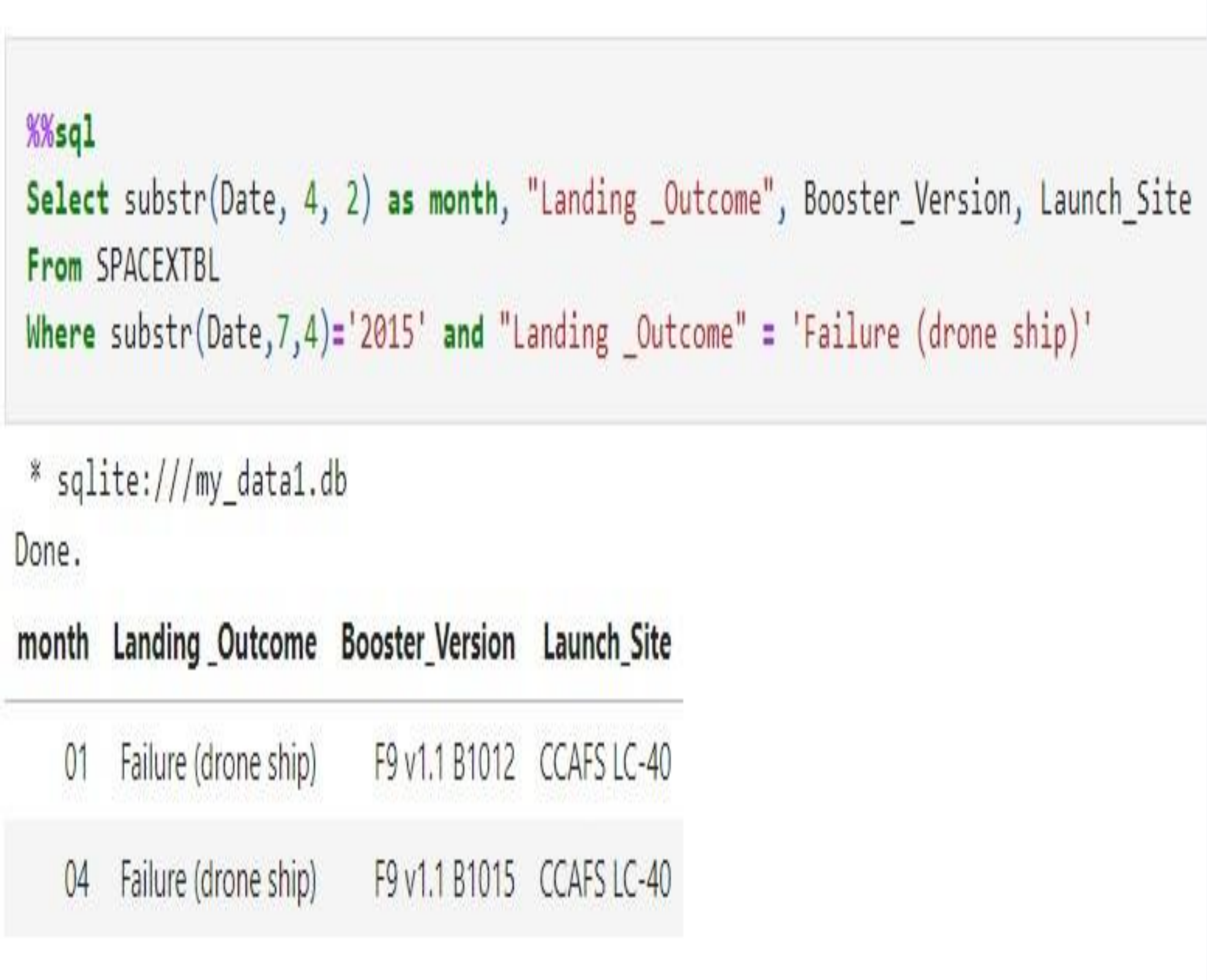
```
 * sqlite:///my_data1.db
Done.
```

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

This query uses a subquery to get the maximum payload mass of the whole table, and then filters it to show all the boosters that carried the maximum payload mass.

The boosters follow this pattern F9 B5 B10xx.x

2015 Launch Records



```
%%sql
Select substr(Date, 4, 2) as month, "Landing _Outcome", Booster_Version, Launch_Site
From SPACEXTBL
Where substr(Date,7,4)='2015' and "Landing _Outcome" = 'Failure (drone ship)'
```

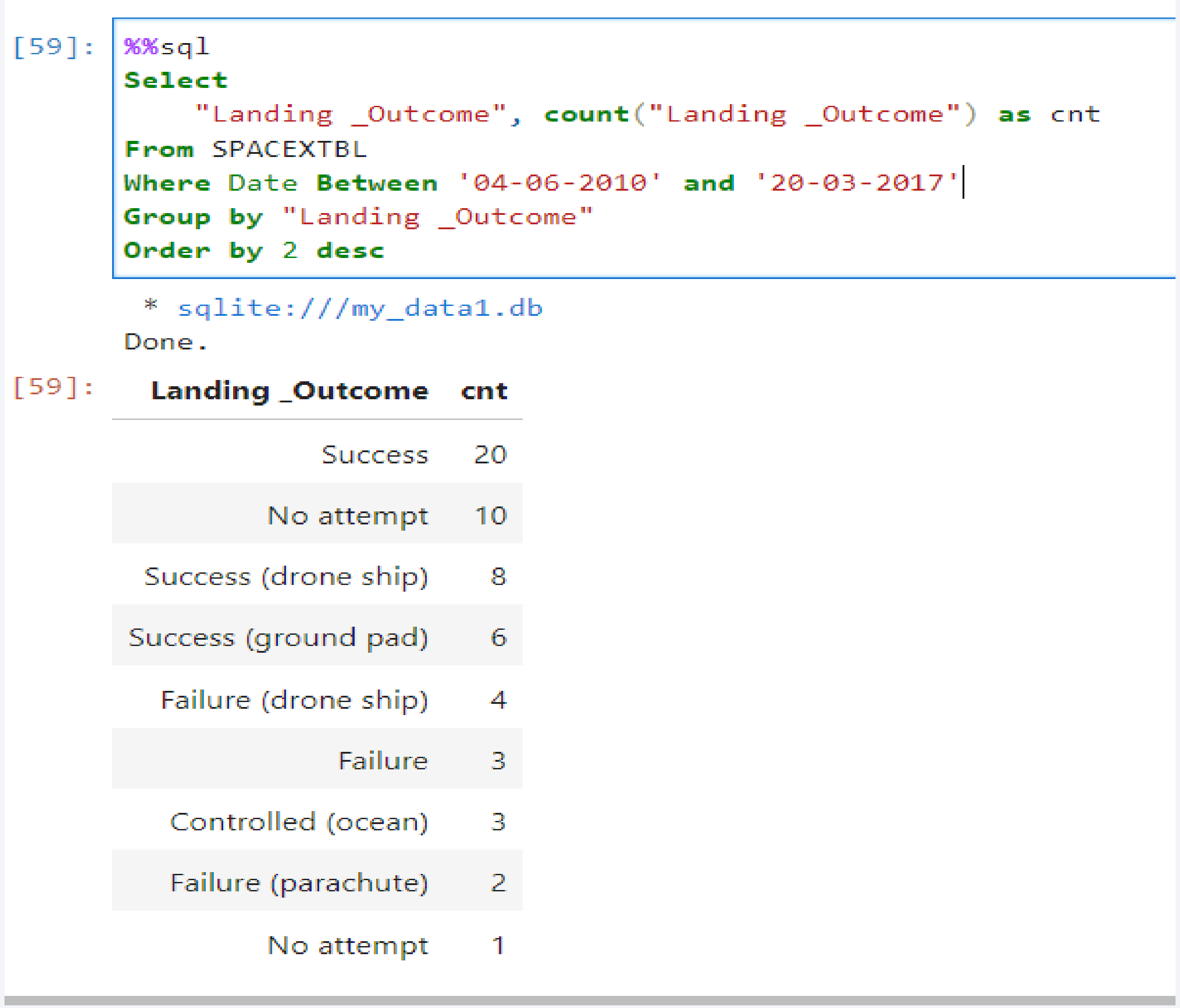
```
 * sqlite:///my_data1.db
Done.
```

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

The query uses the substr function to extract the date and year of the Date column, then we filter only for the year 2015 and Failure on drone ships.

Both of the results were launched from CCAFS LC-40, with booster versions such as F9 v1.1 B1012 and F9 v1.1 B1015.

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



```
[59]: %%sql
      Select
          "Landing _Outcome", count("Landing _Outcome") as cnt
      From SPACEXTBL
      Where Date Between '04-06-2010' and '20-03-2017'
      Group by "Landing _Outcome"
      Order by 2 desc

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

[59]:

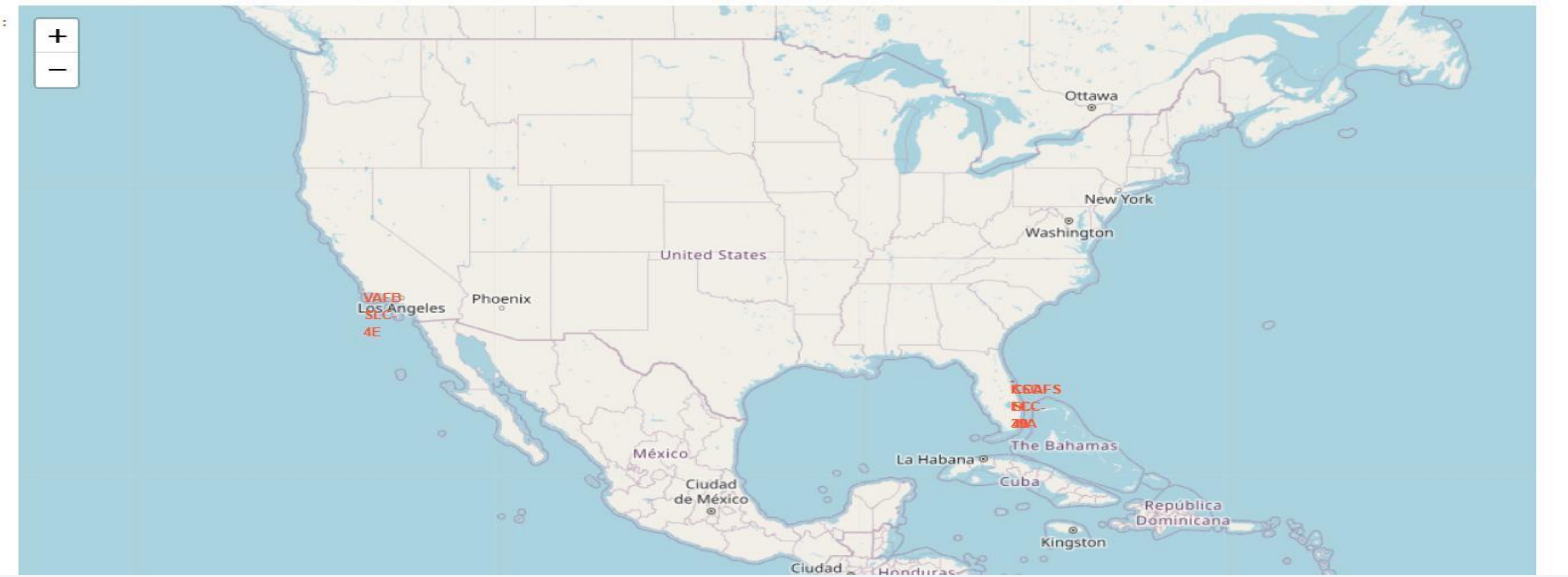
Landing _Outcome	cnt
Success	20
No attempt	10
Success (drone ship)	8
Success (ground pad)	6
Failure (drone ship)	4
Failure	3
Controlled (ocean)	3
Failure (parachute)	2
No attempt	1

It was found that the majority of the missions land back successfully, most of the failures on these dates happened trying to land on a drone ship, with 4 occurrences.

Section 3

Launch Sites Proximities Analysis

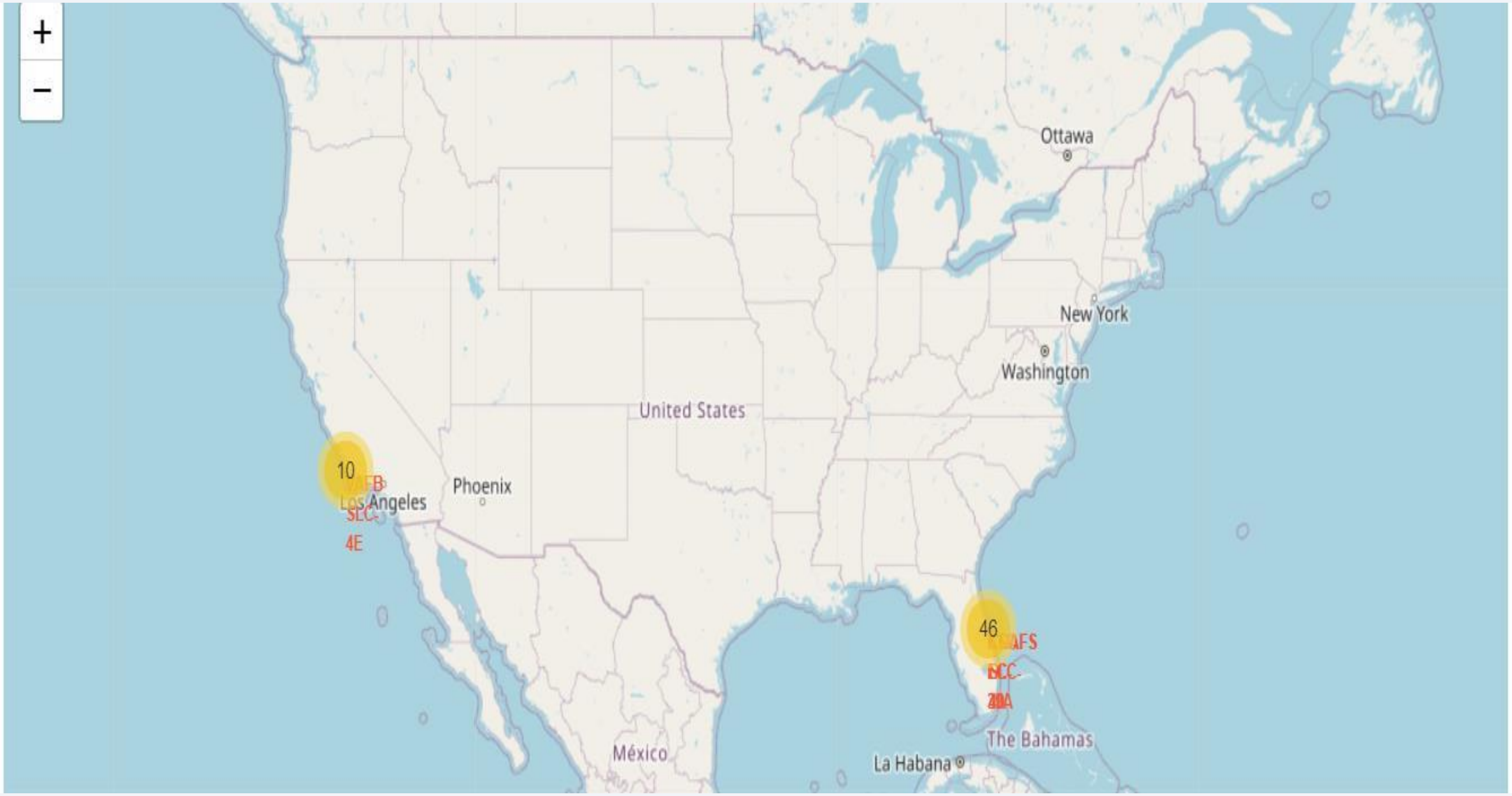
Launch sites in the US

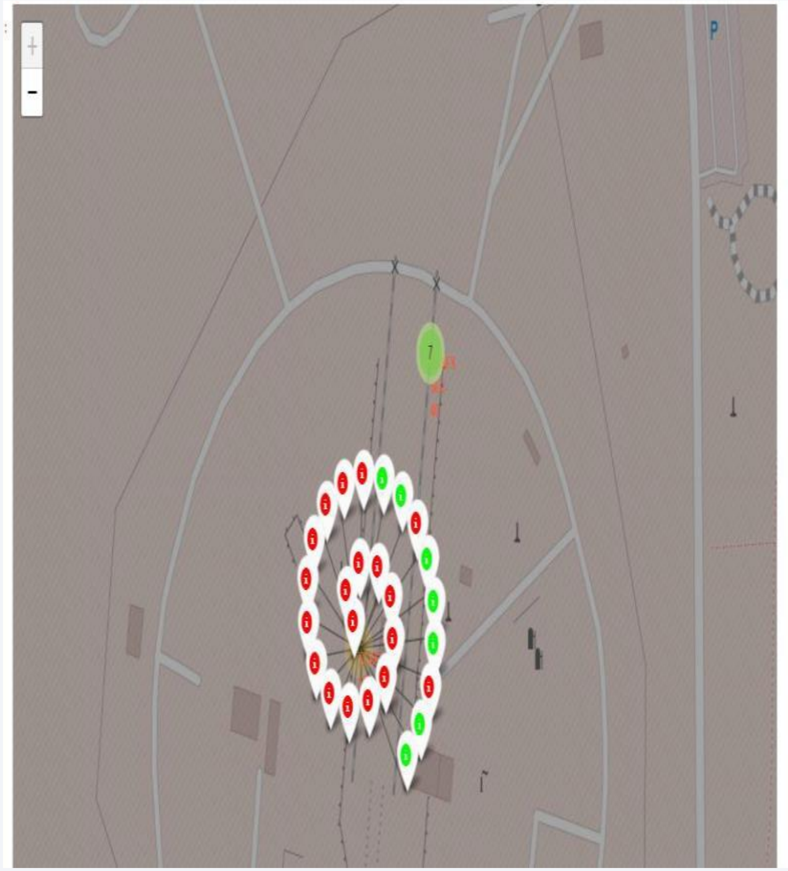


Three out of the four launch sites are in Florida, and one is in California. Most of the launches occur in Florida.

All launch sites are near coastlines, and seem to try to get close to the equator.

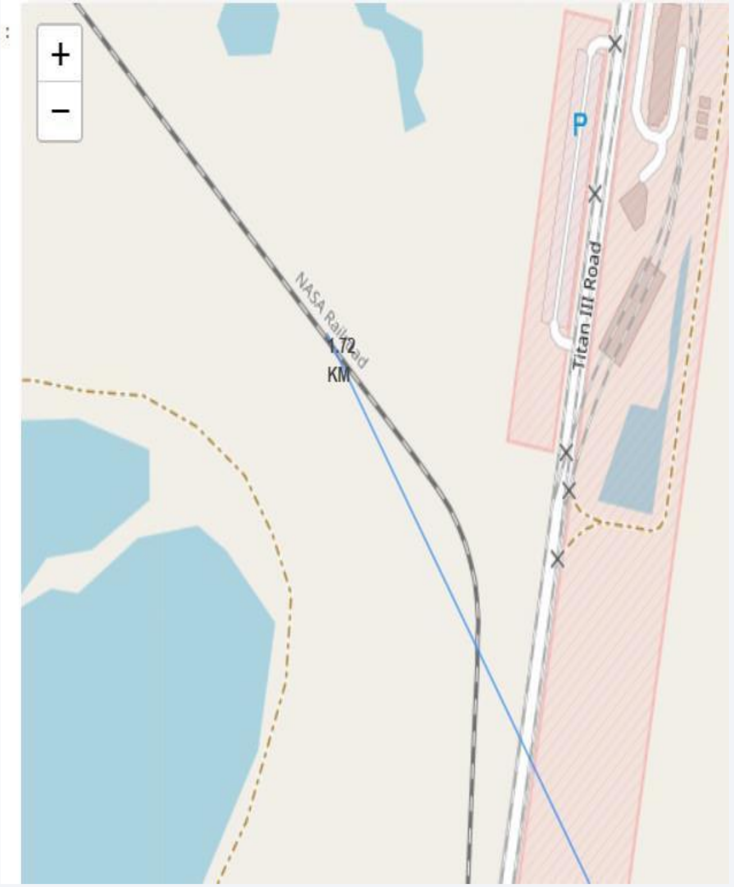
Number of launches for each Launch Site and their landing outcome

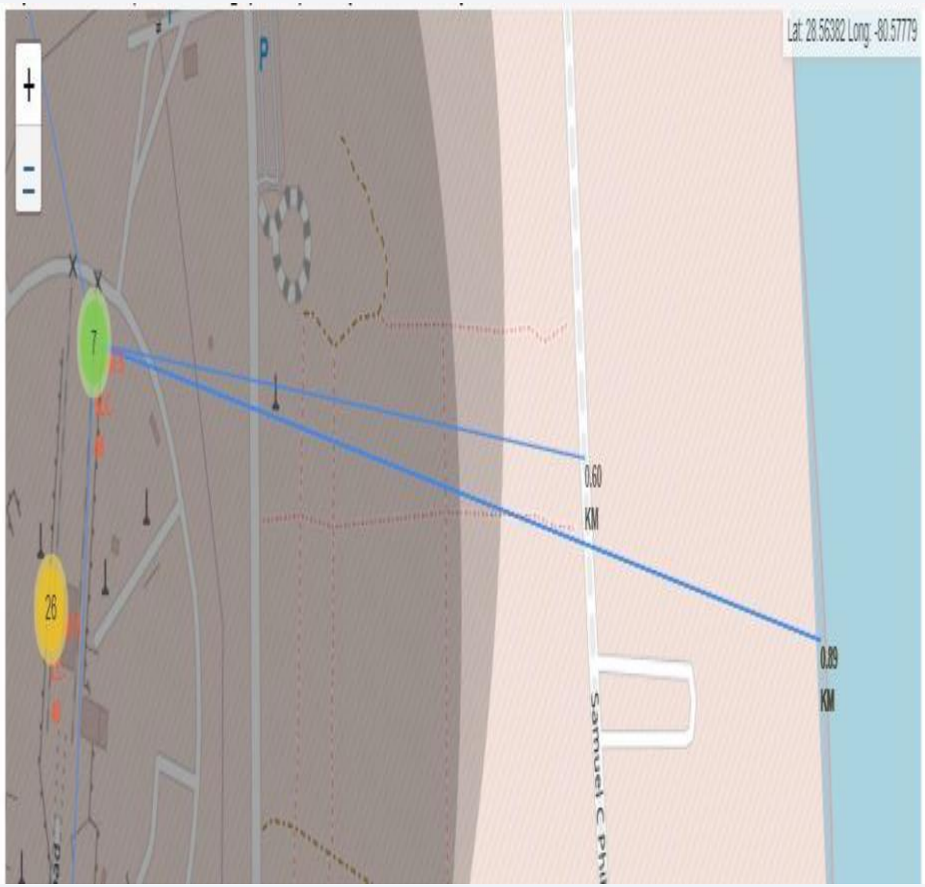


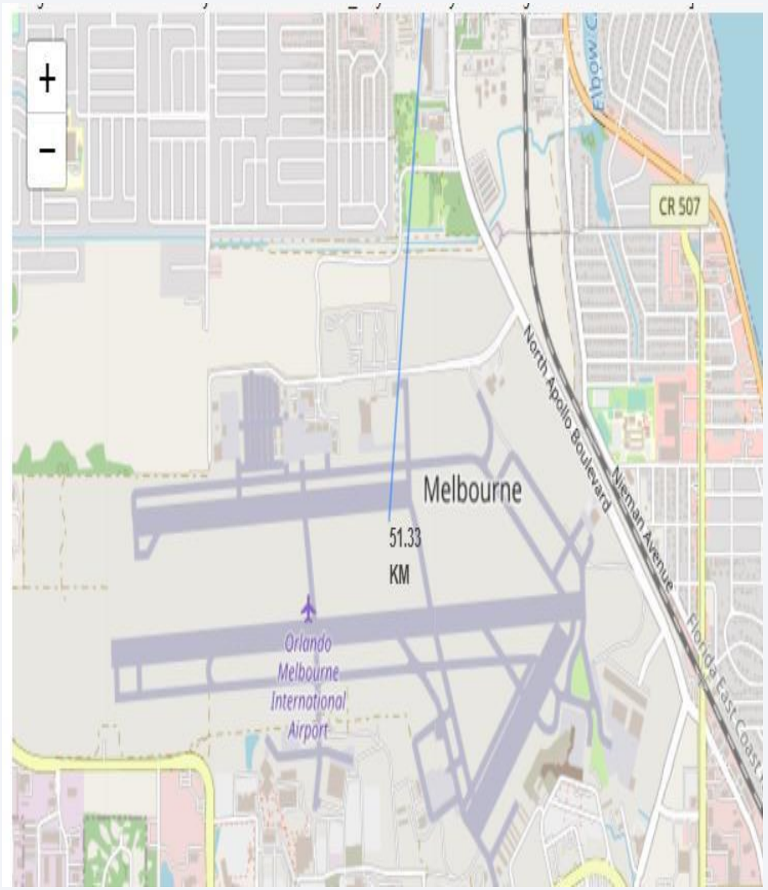


The map shows the number of launches that took place in each site, and if you zoom like the image in the right and then click on a launch site you'll see the icons, the green icons are the successful landings and the red the unsuccessful ones.

Distance from a Launch Site to key locations

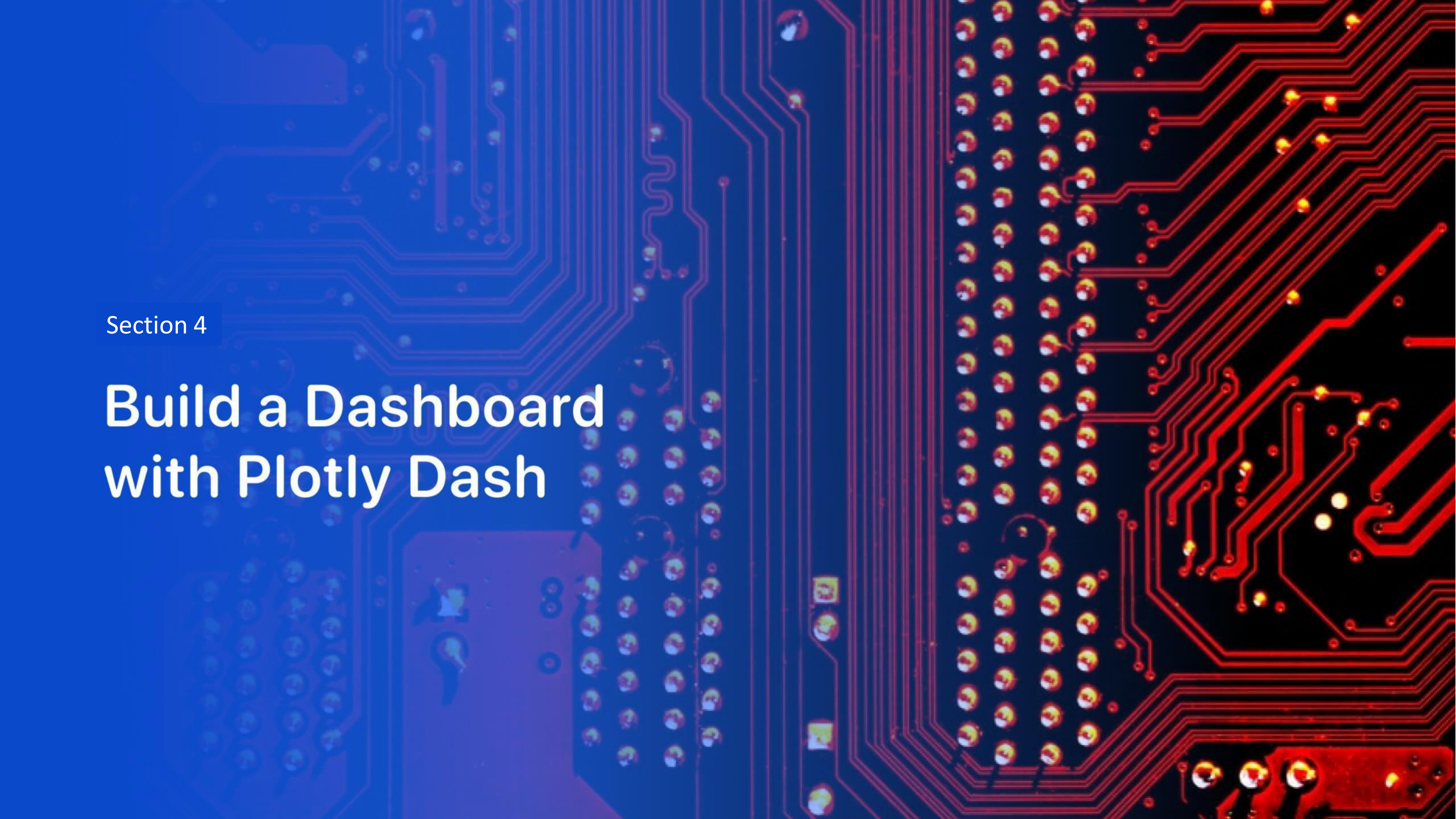






The four key places are a coastline (0.89 km), a highway (0.6 km), a railroad (1.72 km) and the city of Melbourne (51.33 km).

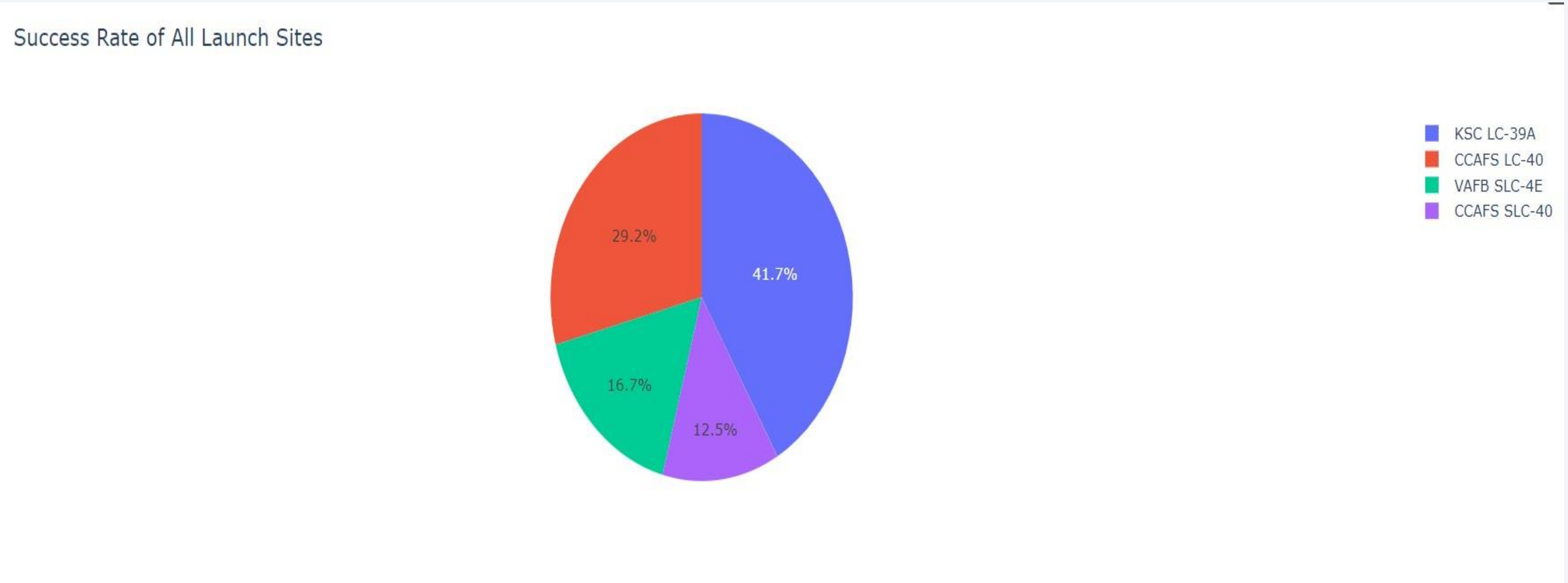
It was found that for all launch sites those four locations are almost always near, except for the city, this is due probably because a launch site needs to transport many elements and people very as quickly as possible.



Section 4

Build a Dashboard with Plotly Dash

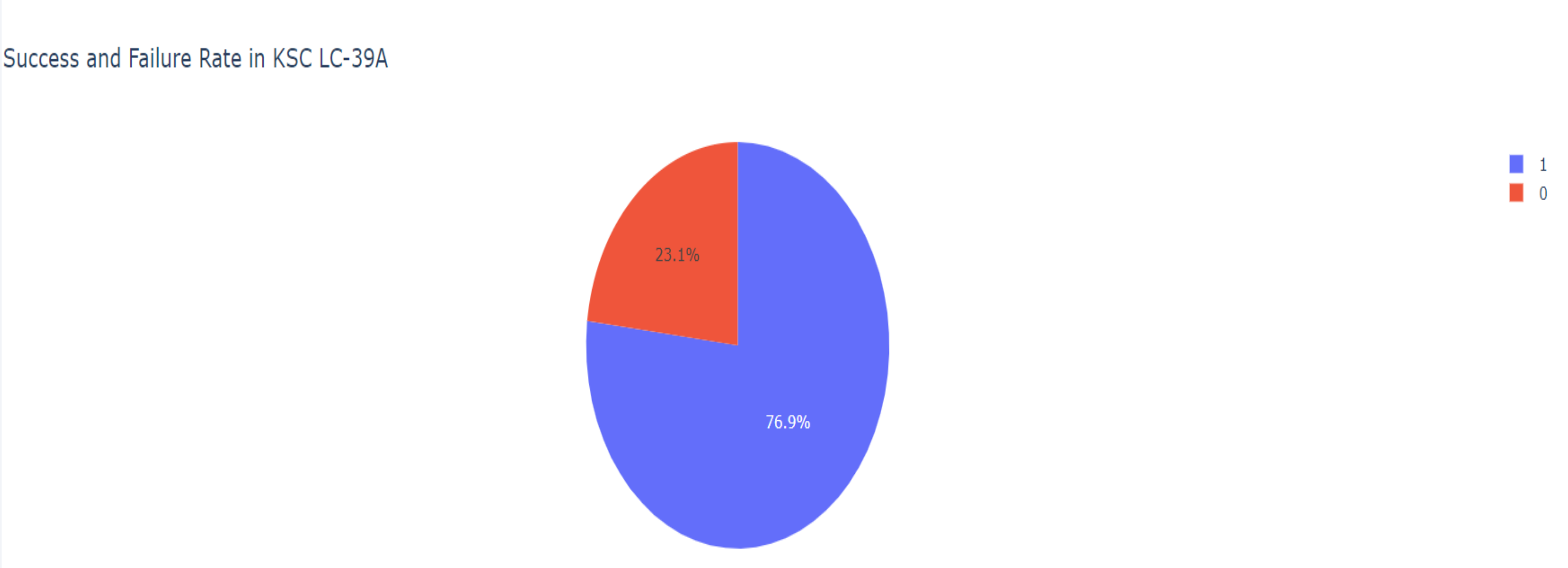
Success rate of all launch sites.



The launch site with the highest success rate is KSC LC-39A, which is located in Florida.

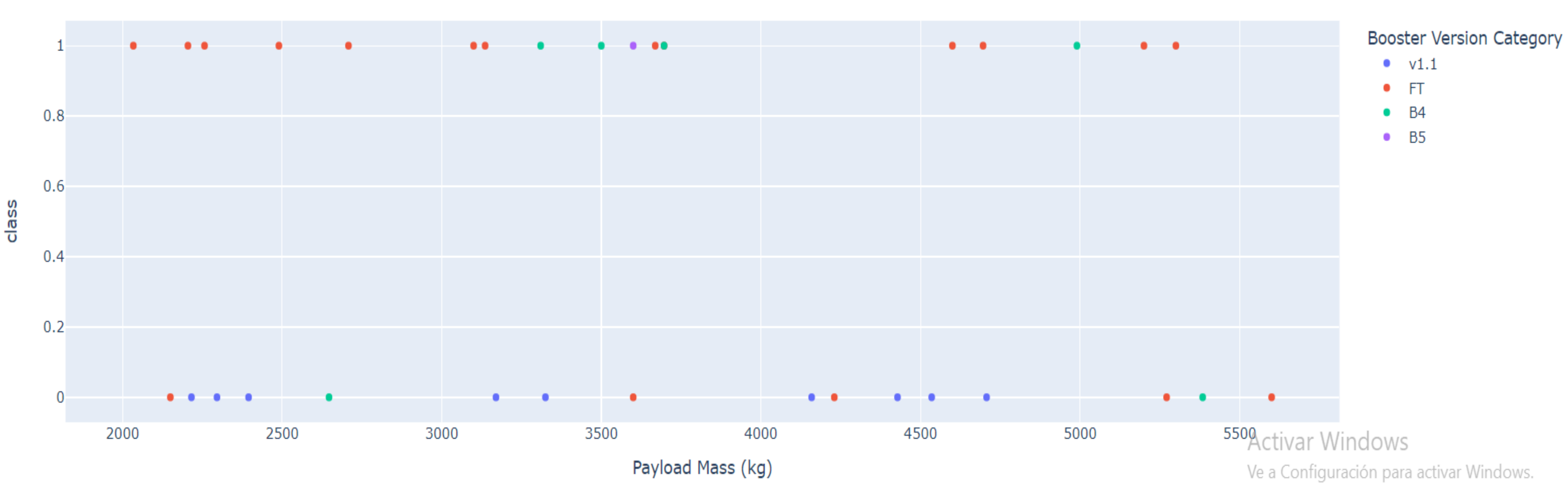
If we were to consider CCAFS LC-40 and CCAFS SLC-40 as one launch site, they would have a success rate that is equal to the success rate of the KSC LC-39A, 41.7%.

The launch site with the highest success rate



This launch site has only 13 out of the 56 launches in total from which 10 have had a successful landing

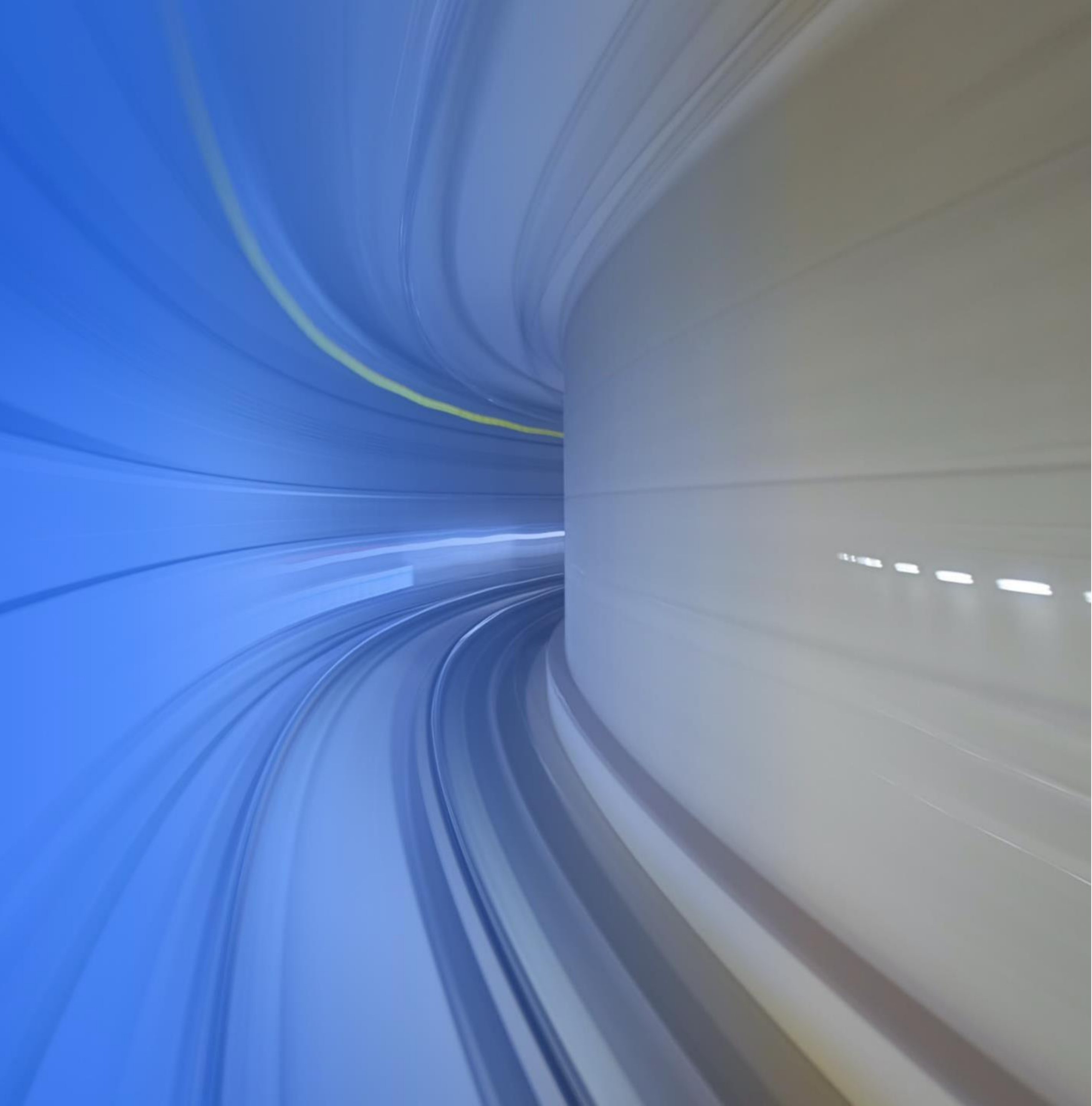
Payload vs. Launch Outcome, and Booster Version



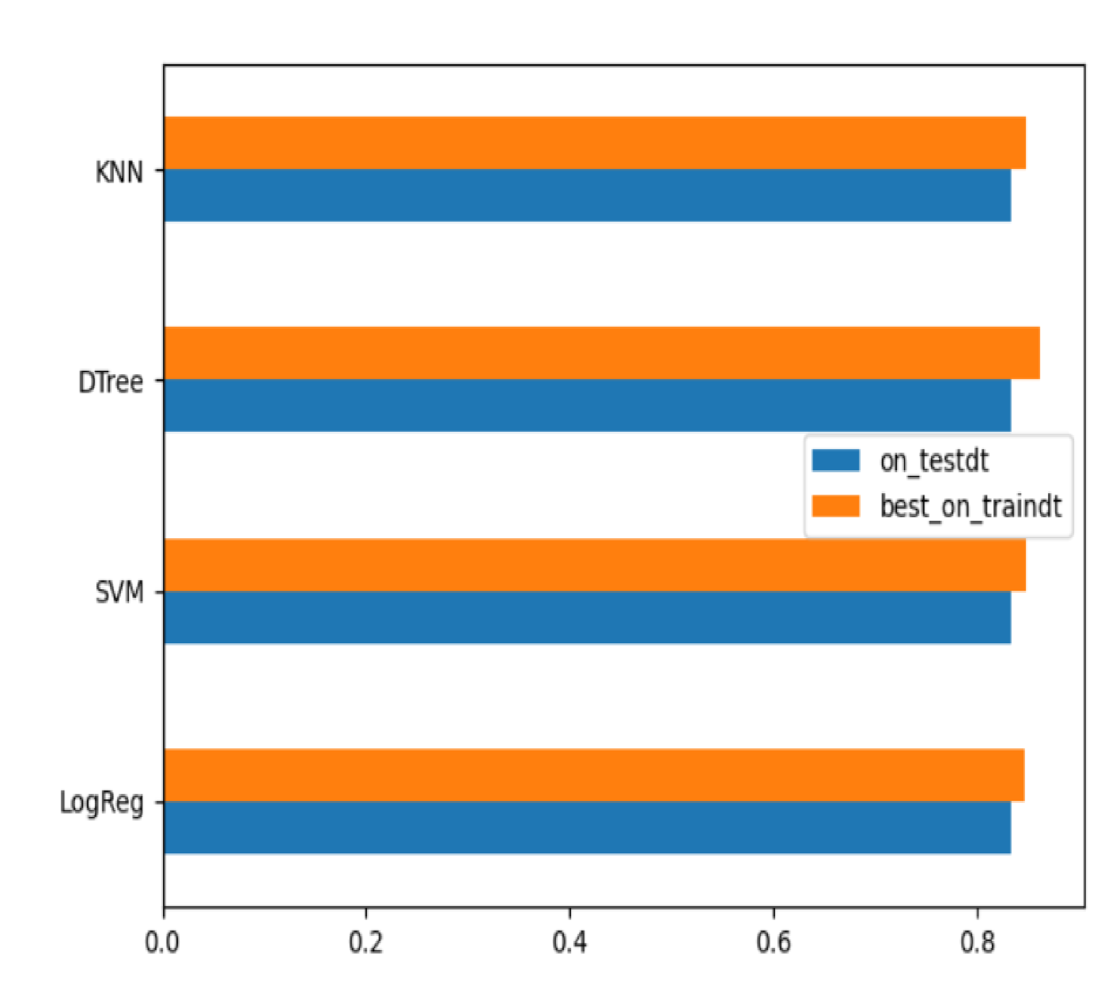
The picture here only shows a range for payload from 2000 to 6000 kg which is the range with the highest successful outcomes, the boosters can also be identified by the color of the dots. The most successful booster for this range is FT, and the least successful one is the B5 booster.

Section 5

Predictive Analysis (Classification)



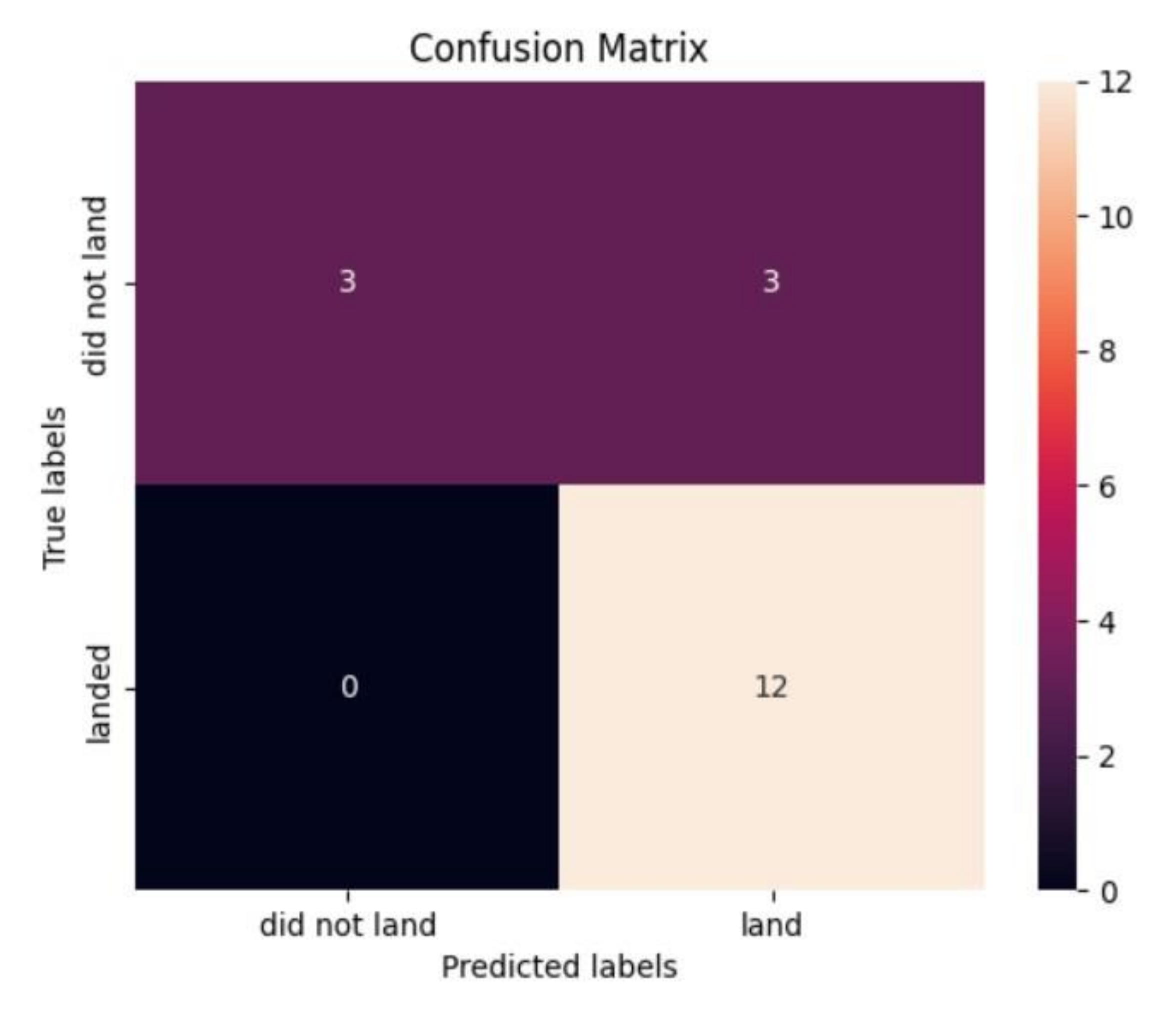
Classification Accuracy



Since all models got the same score on the test dataset, 83.33%, It was necessary to look at other benefits each model brings to the table, that's why the best score on the training set is also graphed.

In the end, I reckon it would be better to choose Logistic Regression since it also offers the probability of a successful landing, which can be used as an estimation of the risk.

Confusion Matrix



Since all the models got the same score, the Confusion Matrix looks the same for all of them.

The matrix shows that the models have an excellent recall (100%) and a good precision (80%) for the landed predicted values.

Which means that they can predict pretty accurately when a rocket will land back.

Conclusions

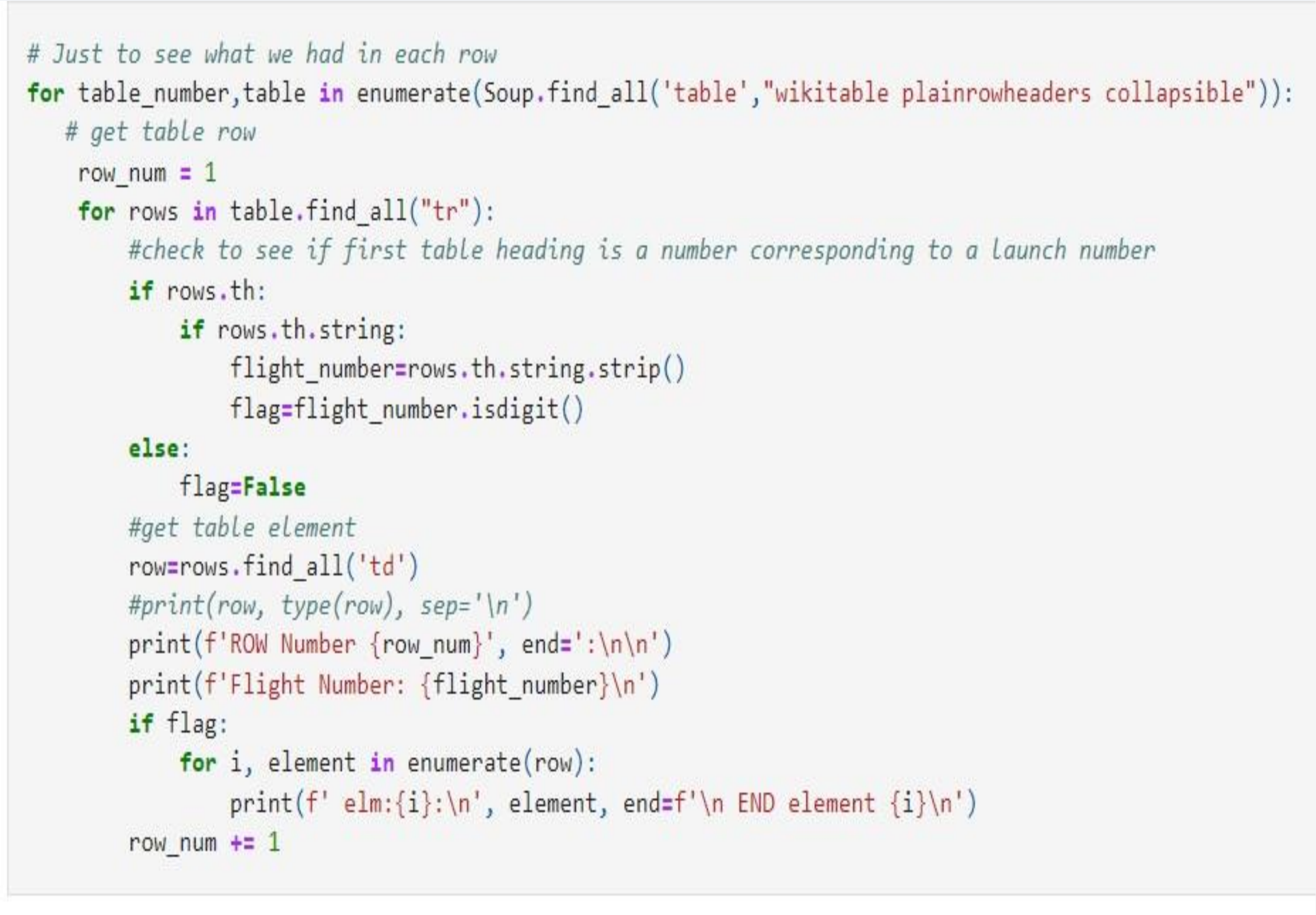
- SpaceX is perfecting the landing of their rockets on all launch sites.
- A specific Payload Mass and Orbit seems to result in a better outcome.
- The most successful Orbits with more than one launch are SSO, VLEO and LEO
- The most successful Launch site is KSC LC-39A, as for the Booster Version it is FT.
- The Logistic Regression model will allow us to predict pretty accurately when a rocket will land back successfully, as well as gauge and estimation of the risk.
- It is worth to notice that the Dataset is small, which means the model must be adjusted when the dataset grows significantly.
- The emphasis of the model could later be set to predict when a rocket will NOT land, so that we can have higher savings, it costs more to check a rocket than to rebuild it.

Acknowledgements

I am joyful for all the quality content and knowledge that it was transmitted to me through-out the courses, I want to thank the IBM staff:

- Joseph Santarcangelo
- Yan Luo
- Rav Ahuja
- Lakshmi Holla
- Azim Hirjani
- Grace Barker
- Rachael Jones
- Kathleen Bergner
- Bella West
- Simer Preet
- Lauren Hall
- Hunter Bay
- Tanya Singh
- Grace Barker
- Om Singh
- Malika Singla
- Duvvana Mrutyunjaya Naidu
- Anita Verma
- And the Coursera staff!

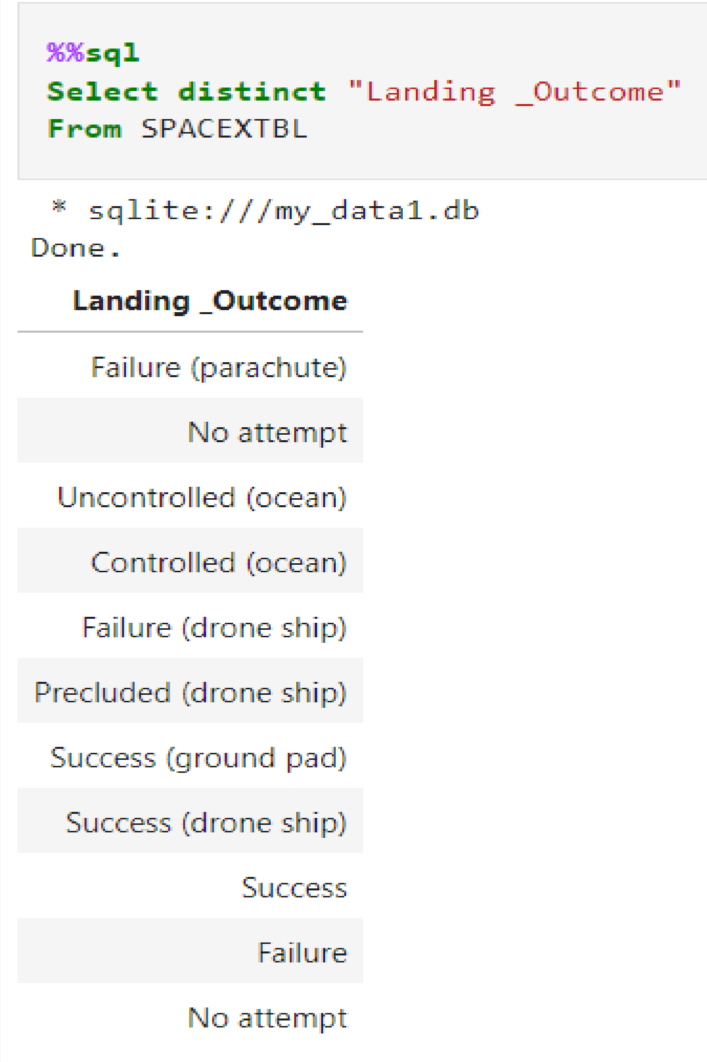
Appendix



```
# Just to see what we had in each row
for table_number,table in enumerate(Soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    row_num = 1
    for rows in table.find_all("tr"):
        #check to see if first table heading is a number corresponding to a launch 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')
        #print(row, type(row), sep='\n')
        print(f'ROW Number {row_num}', end=':\n\n')
        print(f'Flight Number: {flight_number}\n')
        if flag:
            for i, element in enumerate(row):
                print(f' elm:{i}:\n', element, end=f'\n END element {i}\n')
        row_num += 1
```

1. This python snippet, helped me to check the elements for each row, only for those rows that had a flight number.

Appendix



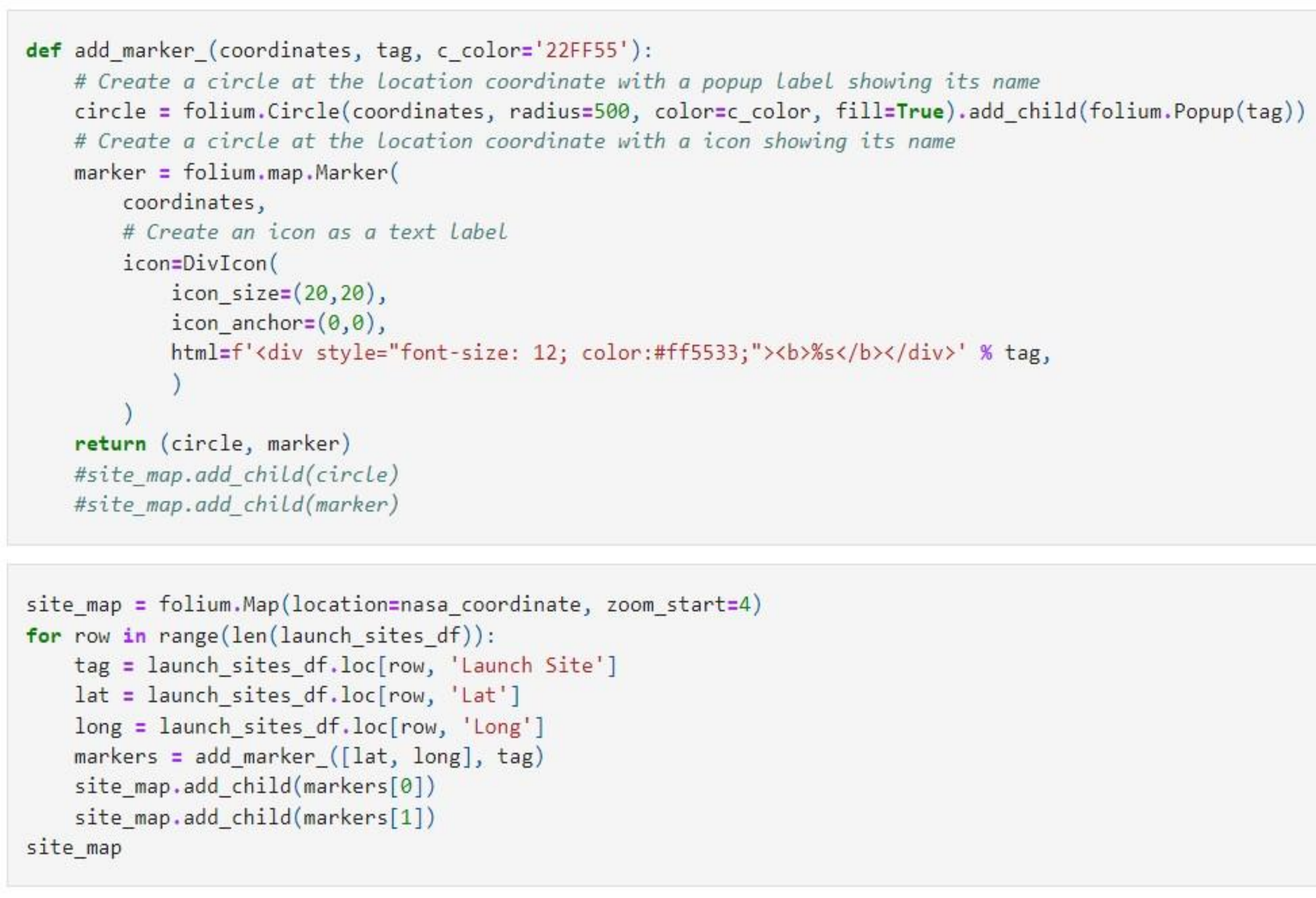
```
%%sql
Select distinct "Landing _Outcome"
From SPACEXTBL
```

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

Landing _Outcome
Failure (parachute)
No attempt
Uncontrolled (ocean)
Controlled (ocean)
Failure (drone ship)
Precluded (drone ship)
Success (ground pad)
Success (drone ship)
Success
Failure
No attempt

2. This SQL snippet was used to get all the possible values for the Landing Outcome column.

Appendix

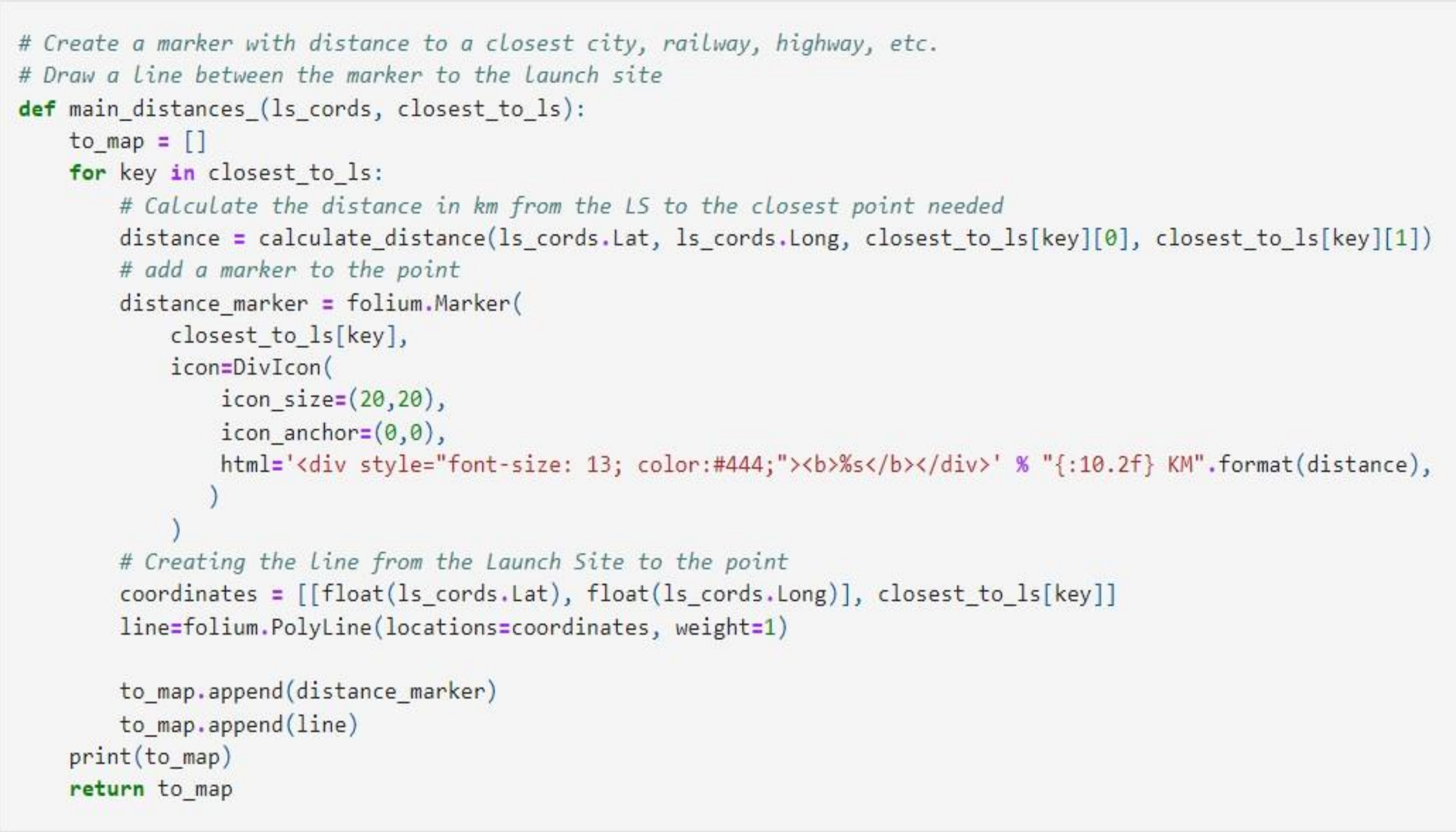


```python
def add_marker_(coordinates, tag, c_color='22FF55'):
    # Create a circle at the location coordinate with a popup label showing its name
    circle = folium.Circle(coordinates, radius=500, color=c_color, fill=True).add_child(folium.Popup(tag))
    # Create a circle at the location coordinate with a icon showing its name
    marker = folium.map.Marker(
        coordinates,
        # Create an icon as a text label
        icon=DivIcon(
            icon_size=(20,20),
            icon_anchor=(0,0),
            html=f'<div style="font-size: 12; color:#ff5533;"><b>%s</b></div>' % tag,
            )
        )
    return (circle, marker)
    #site_map.add_child(circle)
    #site_map.add_child(marker)
```

```python
site_map = folium.Map(location=nasa_coordinate, zoom_start=4)
for row in range(len(launch_sites_df)):
    tag = launch_sites_df.loc[row, 'Launch Site']
    lat = launch_sites_df.loc[row, 'Lat']
    long = launch_sites_df.loc[row, 'Long']
    markers = add_marker_([lat, long], tag)
    site_map.add_child(markers[0])
    site_map.add_child(markers[1])
site_map
```

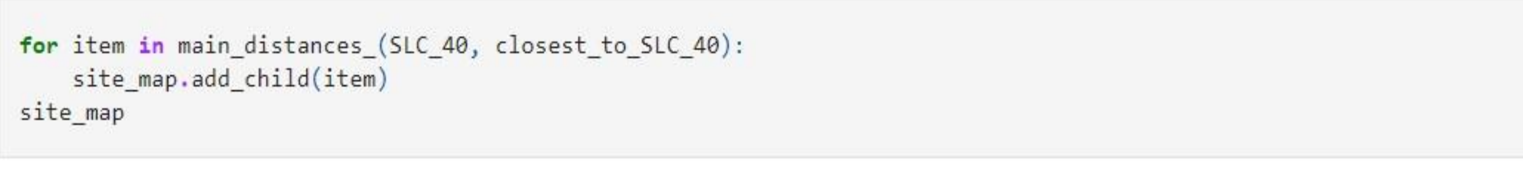
3. The first snippet is a function that produces the circle and the marker, in the following snippet uses the previous function to get the markers for each launch site and adds them to the map.

Appendix



```
# Create a marker with distance to a closest city, railway, highway, etc.
# Draw a line between the marker to the launch site
def main_distances_(ls_cords, closest_to_ls):
    to_map = []
    for key in closest_to_ls:
        # Calculate the distance in km from the LS to the closest point needed
        distance = calculate_distance(ls_cords.Lat, ls_cords.Long, closest_to_ls[key][0], closest_to_ls[key][1])
        # add a marker to the point
        distance_marker = folium.Marker(
            closest_to_ls[key],
            icon=DivIcon(
                icon_size=(20,20),
                icon_anchor=(0,0),
                html='<div style="font-size: 13; color:#444;"><b>%s</b></div>' % "{:10.2f} KM".format(distance),
               )
            )
        # Creating the line from the Launch Site to the point
        coordinates = [[float(ls_cords.Lat), float(ls_cords.Long)], closest_to_ls[key]]
        line=folium.PolyLine(locations=coordinates, weight=1)

        to_map.append(distance_marker)
        to_map.append(line)
    print(to_map)
    return to_map
```



```
for item in main_distances_(SLC_40, closest_to_SLC_40):
    site_map.add_child(item)
site_map
```

4. The function above calculates the distance from the Launch site to a key location, then creates a marker and polyline and appends it to a list, which later is added to the map.

Thank you!