IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

By Olivia Itzkovitz

Last Updated: Sunday, December 18, 2022

Outline

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

Executive Summary

Summary of methodologies:

- Data collection with API
- Data collection with web scraping
- Data wrangling
- Exploratory data analysis with data visualization
- Exploratory data analysis with SQL
- Interactive map with Folium
- Dashboard building with Plotly
- Predictive analysis with classification models

Summary of all results:

- Exploratory data analysis results
- Interactive analysis results
- Predictive analysis results

Introduction

Space travel has the ability to become more affordable for everyone, yet most rocket providers can have rocket launches cost upward of 165 million dollars. However, there is one company that provides the cheapest available option in comparison. SpaceX and their Falcon 9 rocket can cost 62 million dollars for its rocket launches. They have the ability to provide more affordable pricing because unlike the other providers, SpaceX's Falcon 9 is able to recover after the first stage, allowing it to be reused. Therefore, if the landing outcome of the first stage can be determined, so can the cost of each launch. Given this information, a solution can be obtained by asking the following questions:

- What factors are causing the first stage to crash?
- What are the circumstances in which the launches are successful?
- Which classification model will perform with the most accuracy?

Section 1

Methodology

Methodology

Executive Summary

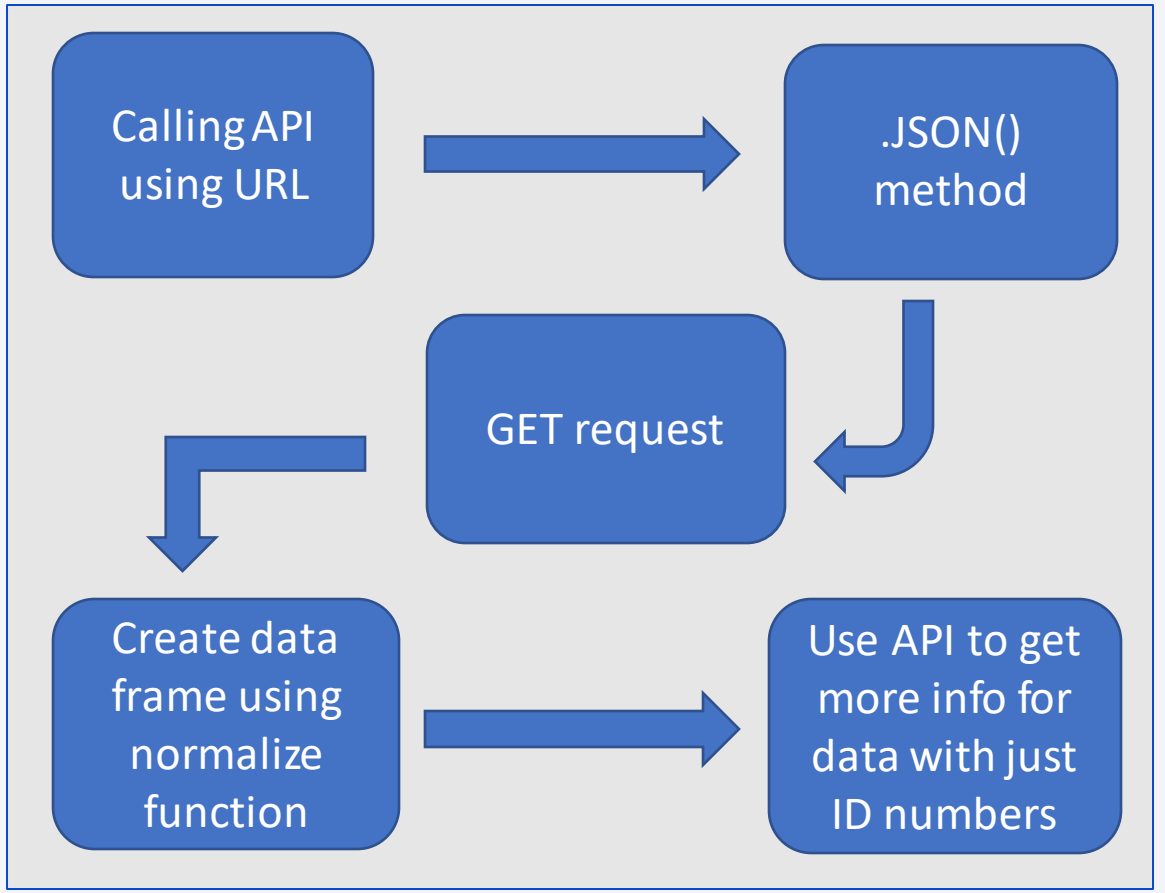
- Data collection methodology
  - Data is collected from web scraping Wikipedia
  - Data is collected using API calls. Using GET requests, data is extracted from columns of each dataset
- Perform data wrangling
  - Data is cleaned and organized to be processed and analyzed
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Split into train/test data to build, tune, and evaluate classification models

Data Collection

- To receive past launch data, we must target a specific endpoint of the API by using a URL
- Using the requests library we will carry out a GET request
- By calling the .JSON() method, we can view this result
- Our response will be a list of JSON objects, which each represents a launch
- Using the normalize function, we convert the JSON to a data frame
- We are also obtaining launch data from web scraping Wiki pages
- Using the Beautiful Soup package we are able to web scrape from html tables
- From there, we need to parse that data and convert it to a data frame in pandas Taking it from raw data to a clean dataset

Data Collection – SpaceX API

- Used column names to call the API and append the data to the lists.
- A URL was used to request rocket launch data
- Creating a data frame using the normalized JSON response
- Using the ID number of columns of data we want to gain more information from to make a subset of our data frame
- Storing that data into lists, and then into a dictionary
- Finally, take the dictionary we created from the data we obtained, and create a new data frame in pandas with all the information necessary

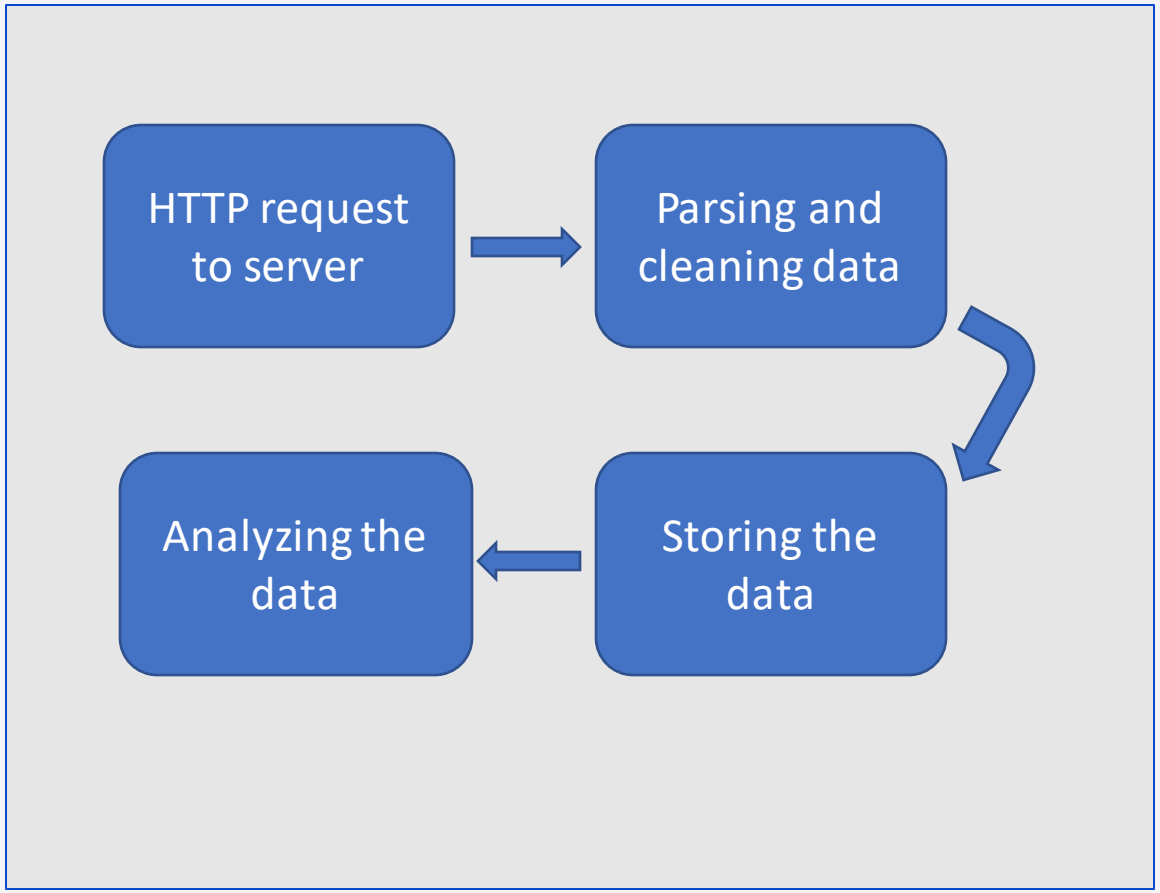


Data Collection link

Data Collection - Scraping

We start by getting the contents of useful web pages. Next we must extract the data from those web pages and start parsing it using the tool Beautiful Soup. Then, we create a data frame, making it easier to understand the data. Finally, we store the new form of data locally which gives us access to analyze it freely.

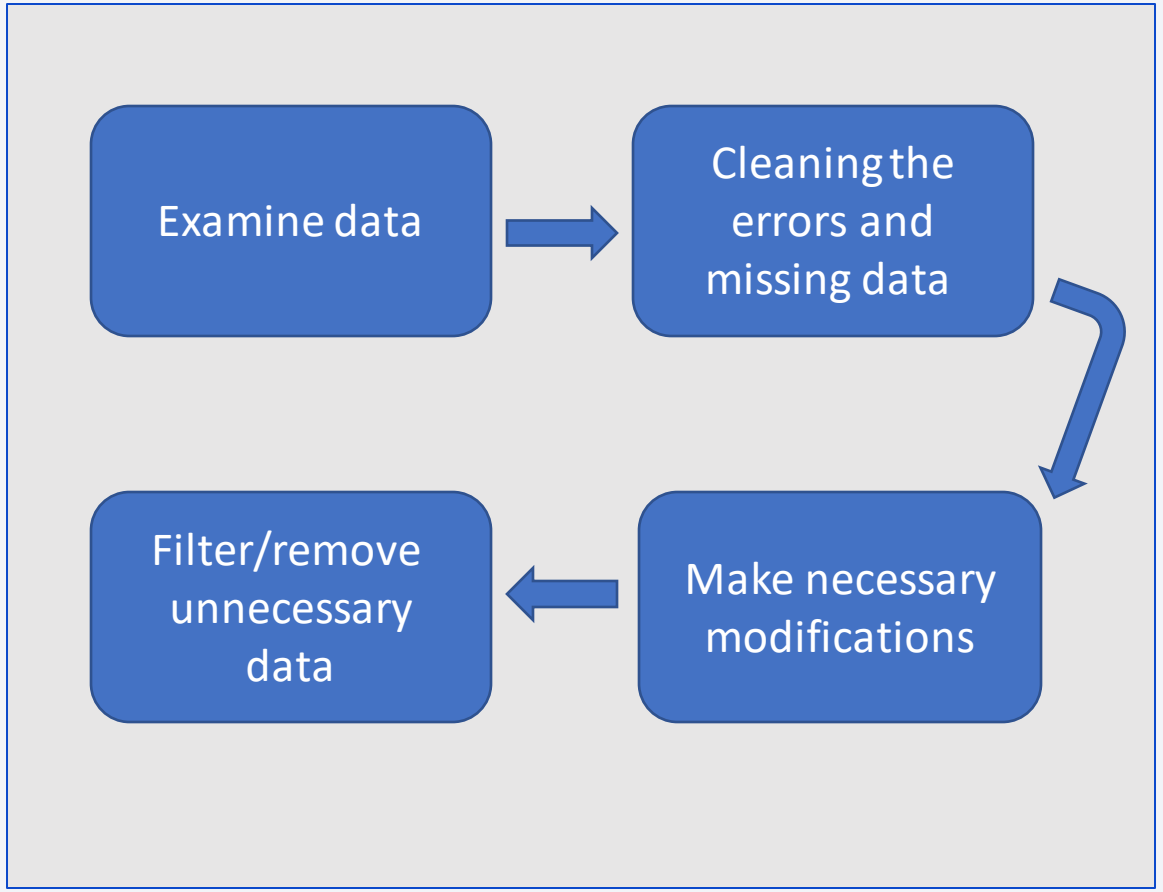
Webscraping link



Data Wrangling

To start, we examine the data and see what needs to be cleaned up and organized. This process includes; determining the percentage of missing values, replacing those missing values, establishing the datatypes of each column, calculating the number of launches, computing the number and occurrences of orbit, and assessing the landing outcomes for each launch of the data frame. We are then able to use classification variables to represent the outcome of each launch in order to be able to visualize and analyze the data easier.

Data Wrangling link



EDA with Data Visualization

There were multiple charts that were plotted for various reasons. The first one displayed is a scatter plot, used for multiple examples. It is used to determine if one variable would have an effect on the other variable and its outcome (ie. Payload vs. Launch site). The next graph is a bar chart that allows us to establish whether or not there is a correlation between orbit type and success rate. Finally, a line chart indicating the average success rate of launches per year.

EDA with Data Visualization link

EDA with SQL

The SQL queries performed include:

- Distinct – to show all the different launch site names
- Like – to find the launch sites with the specific pattern 'CCA'
- Sum and Avg – to find the total and average payload mass
- Min – to show the date of the first successful landing
- Group by – creates sub groups (ie. The success of a mission outcome)
- Count – counts the number of mission outcomes
- As – names a new column
- Subquery – used to name the max payload mass carried by a booster version
- EDA with SQL link

Build an Interactive Map with Folium

- Starting off by using Folium.Map to create an interactive map using coordinates to make Nasa Space Center the center location
- The marker and circle function are used to create a circle surrounding a popup label of the launch sites
- Creating a marker cluster object to group the successful launches together using a green marker and the unsuccessful launches together using a red marker
- The mouse position function is used to show the longitude and latitude coordinates on the map by hovering your mouse over a location
- The mouse position and the distance formula are used to get the coordinates and find the distance between launch sites and the closest coastlines, railways, highways, and cities
- Once we find the distance, we use Poly Line to draw a line to visually represent the distance between those points

Interactive Map link

Build a Dashboard with Plotly Dash

- In creating a dashboard, we start off by adding a dropdown menu to view all the launch site locations
- The filter function gives us the ability to select a specific location which will show only the information/graph associated with that location. It is used multiple times throughout this lab
- Using our callback function we create a pie chart of the total successful launches
- A range slider is then added to choose payload range to test the correlation between launch outcome and the payload mass
- A scatter plot is created to observe the correlation between the payload mass and the launch outcome and using a color label to observe the outcomes given each booster version

Dashboard with Plotly link

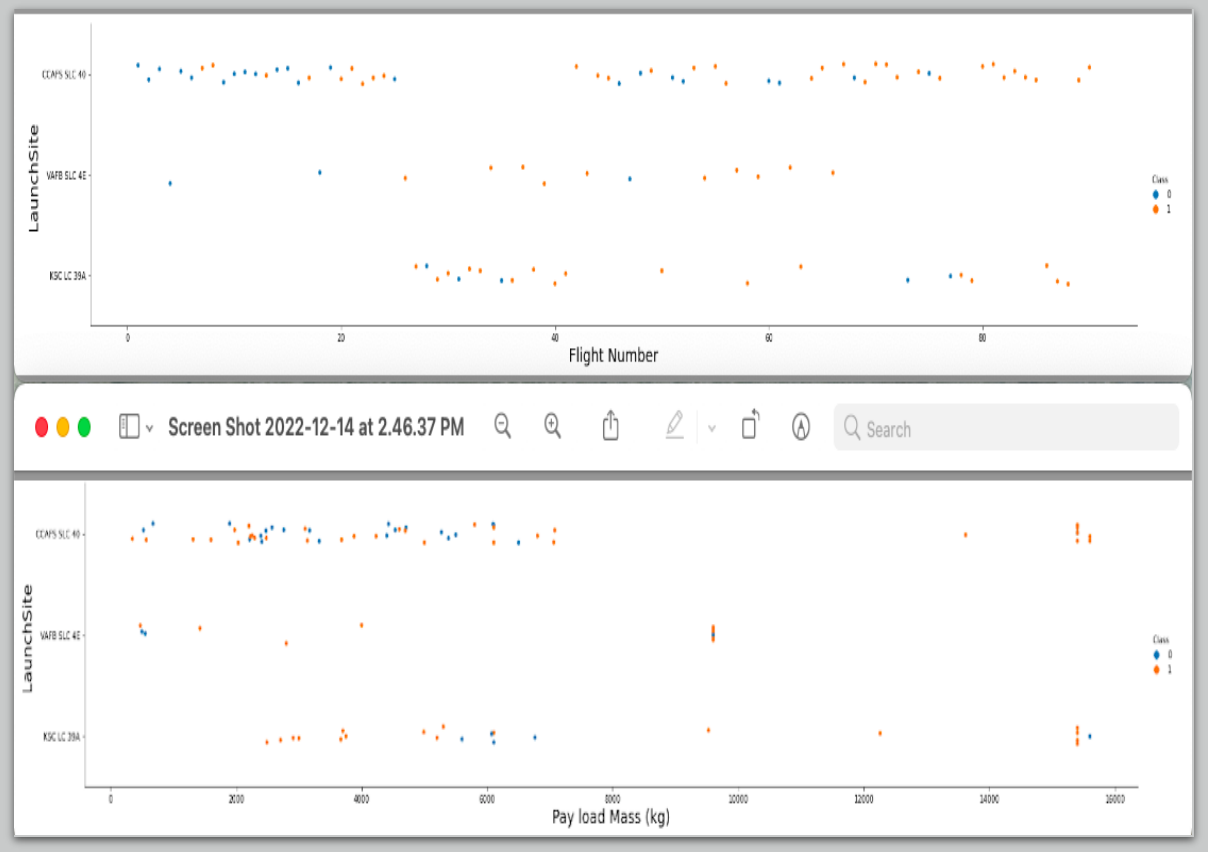
Predictive Analysis (Classification)

- To start, we must split the data for training and testing by using the function train_test_split.
- The optimal parameter values for each model is found using GridSearchCV.
- The classification models used were logistic regression, supervised vector machine, decision tree, and k nearest neighbor.
- To test each model, we must create a classifier object, find the optimal parameter values and then use them to fit the training set.
- From there, we use the test set to calculate the accuracy results using the score method.
- We then create a confusion matrix to test the accuracy of our predicted results against the actual results.
- Finally, to find the best performing method, we take the accuracy results of each test performed and compare them against the others.

Predictive Analysis link

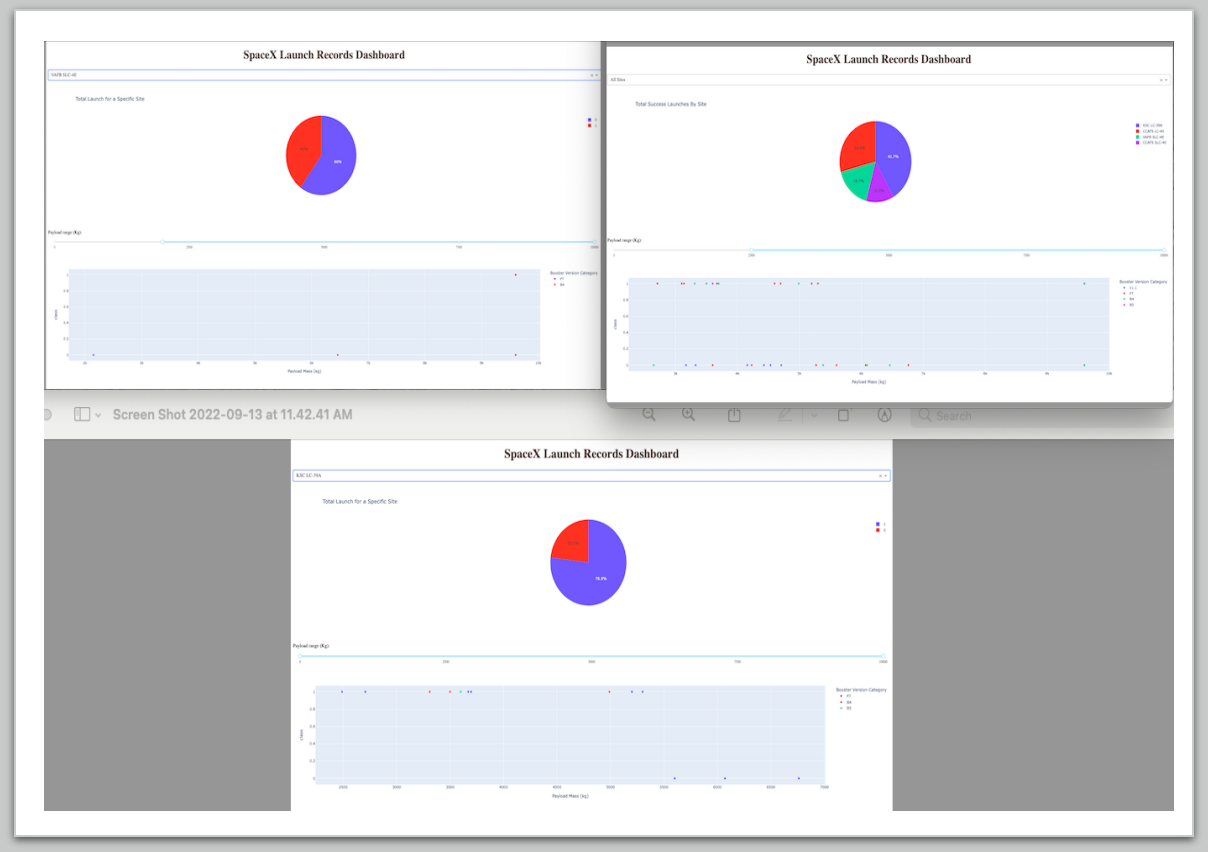
Results

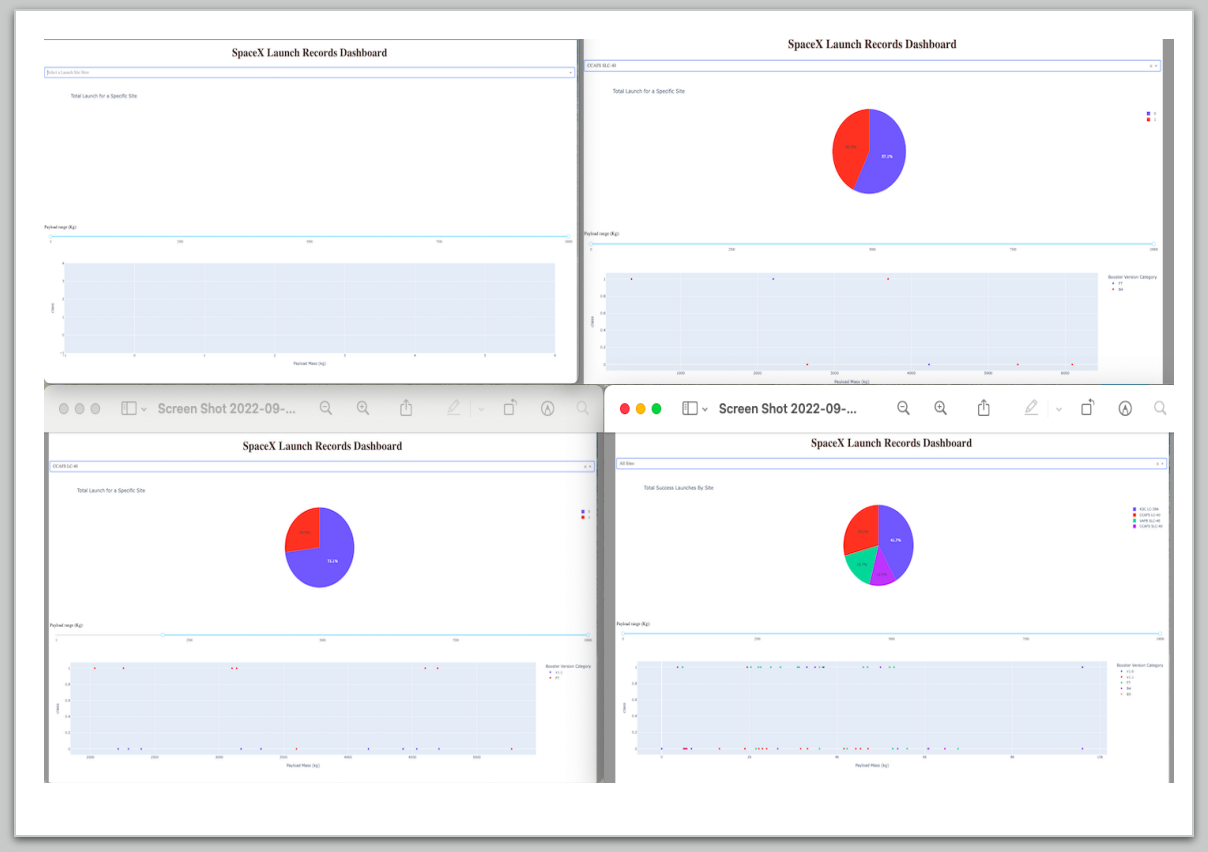
Exploratory Data Analysis Screenshots



Results

Interactive Analytics Demo Screenshots





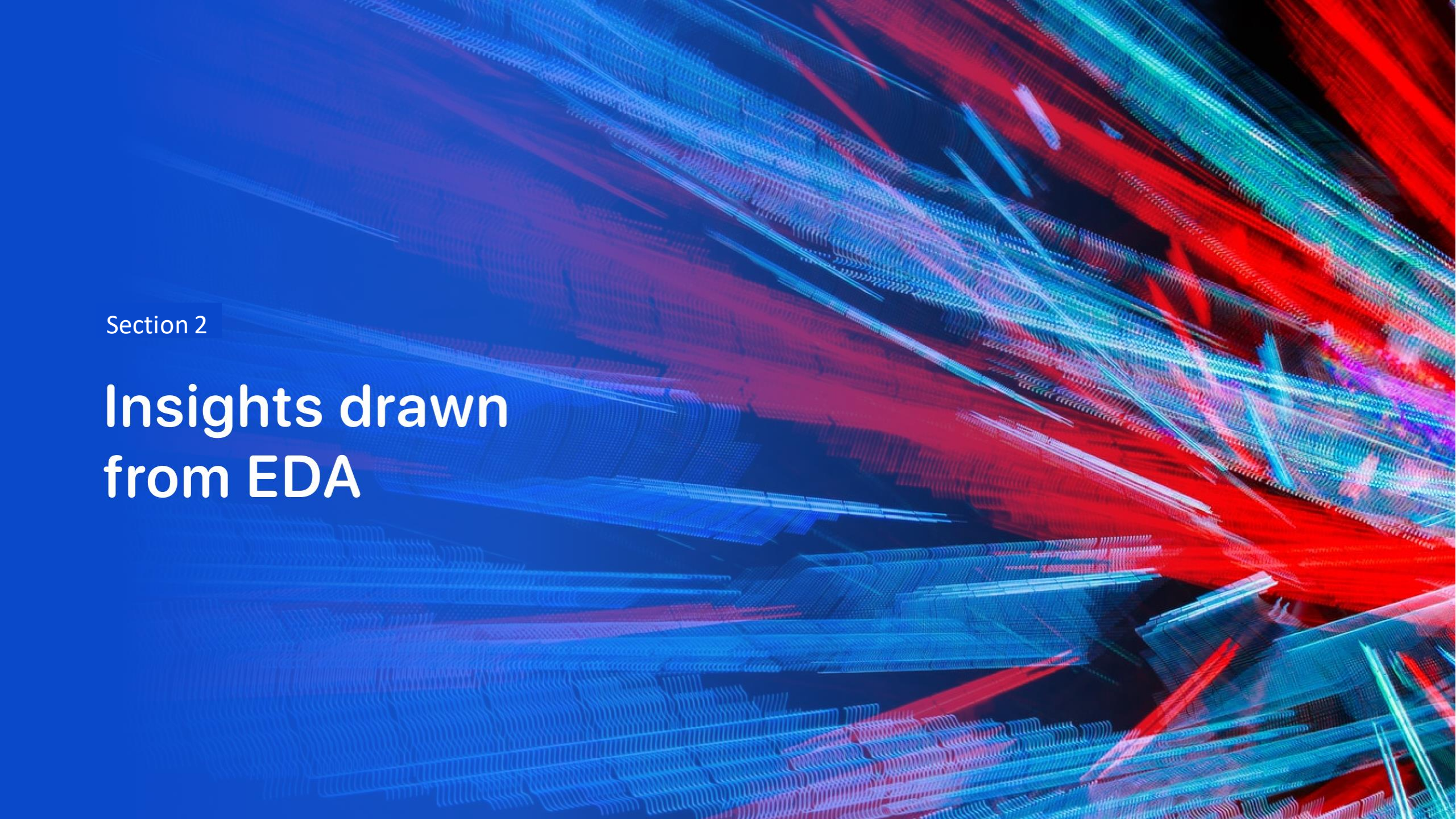
Results

Predictive Analysis Screenshot

Find the method performs best:

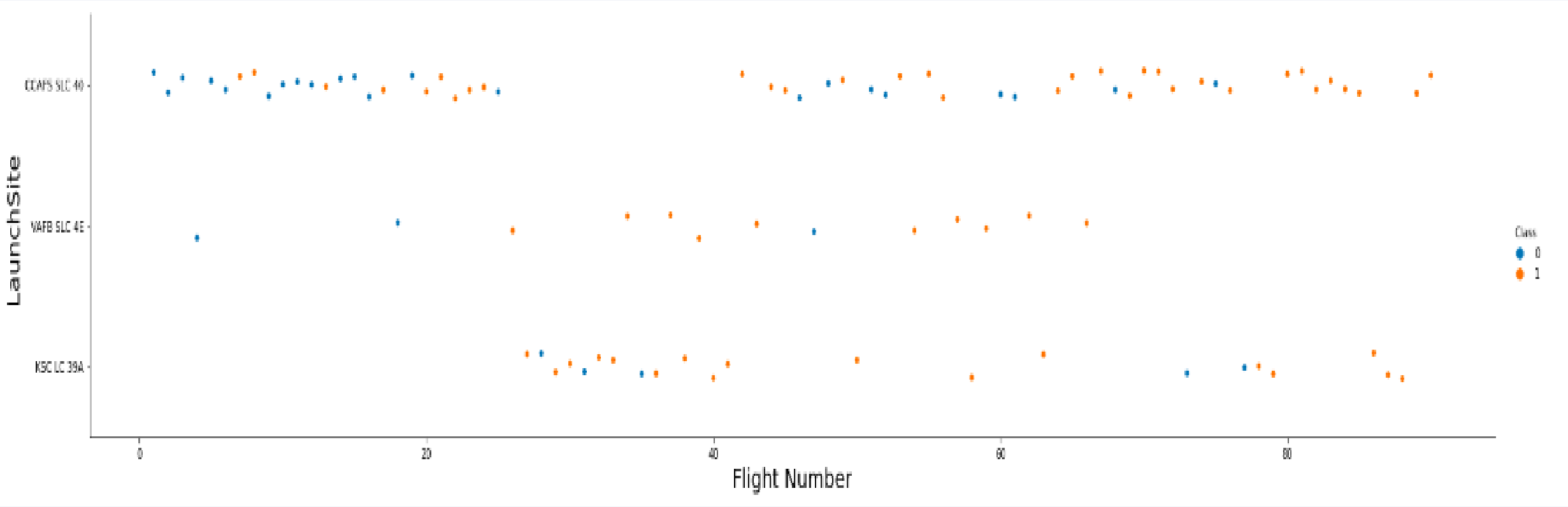
```
print("accuracy for logistic regression method :",logreg_cv.score(X_test,Y_test))

print("accuracy for supervised vector machine method :",svm_cv.score(X_test,Y_test))

print("accuracy for decision tree method :",tree_cv.score(X_test,Y_test))

print("accuracy for k nearest neighbour method :",knn_cv.score(X_test,Y_test))
```

```
accuracy for logistic regression method : 0.8333333333333334
accuracy for supervised vector machine method : 0.8333333333333334
accuracy for decision tree method : 0.9444444444444444
accuracy for k nearest neighbour method : 0.8333333333333334
```


Section 2
Insights drawn from EDA

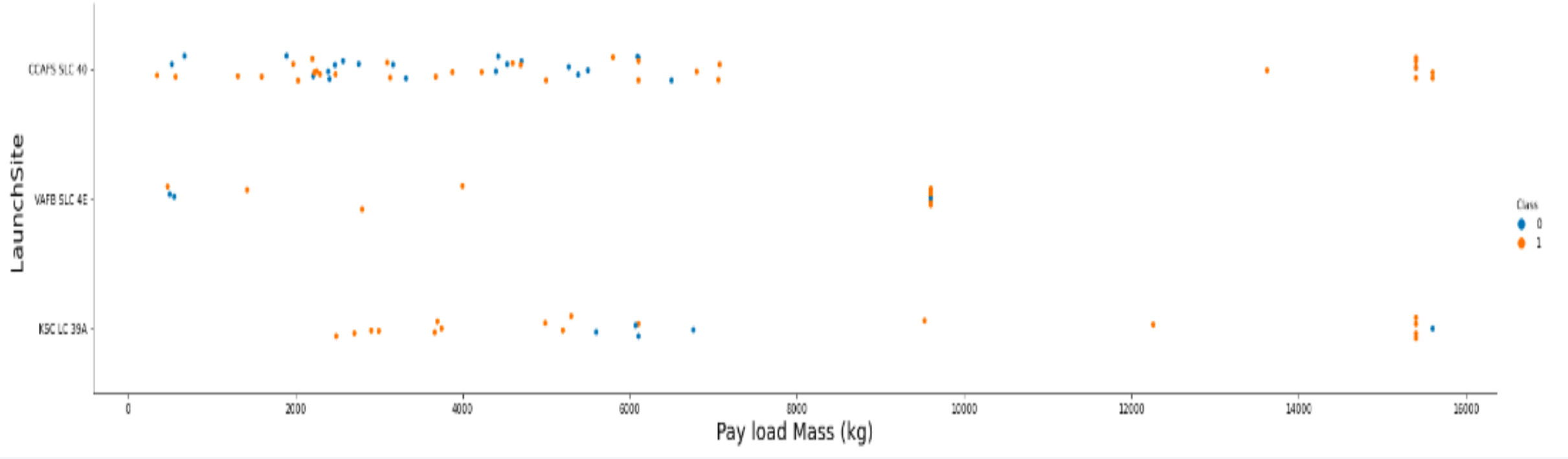
Flight Number vs. Launch Site

The class represents the landing outcome, in which blue means it's a bad outcome and orange means it's a good outcome. From this scatter plot, we can conclude that once the number of flights surpass 20, the landing success rate will be much greater among all the launch sites. Therefore, the greater the flight number, the higher the success rate.



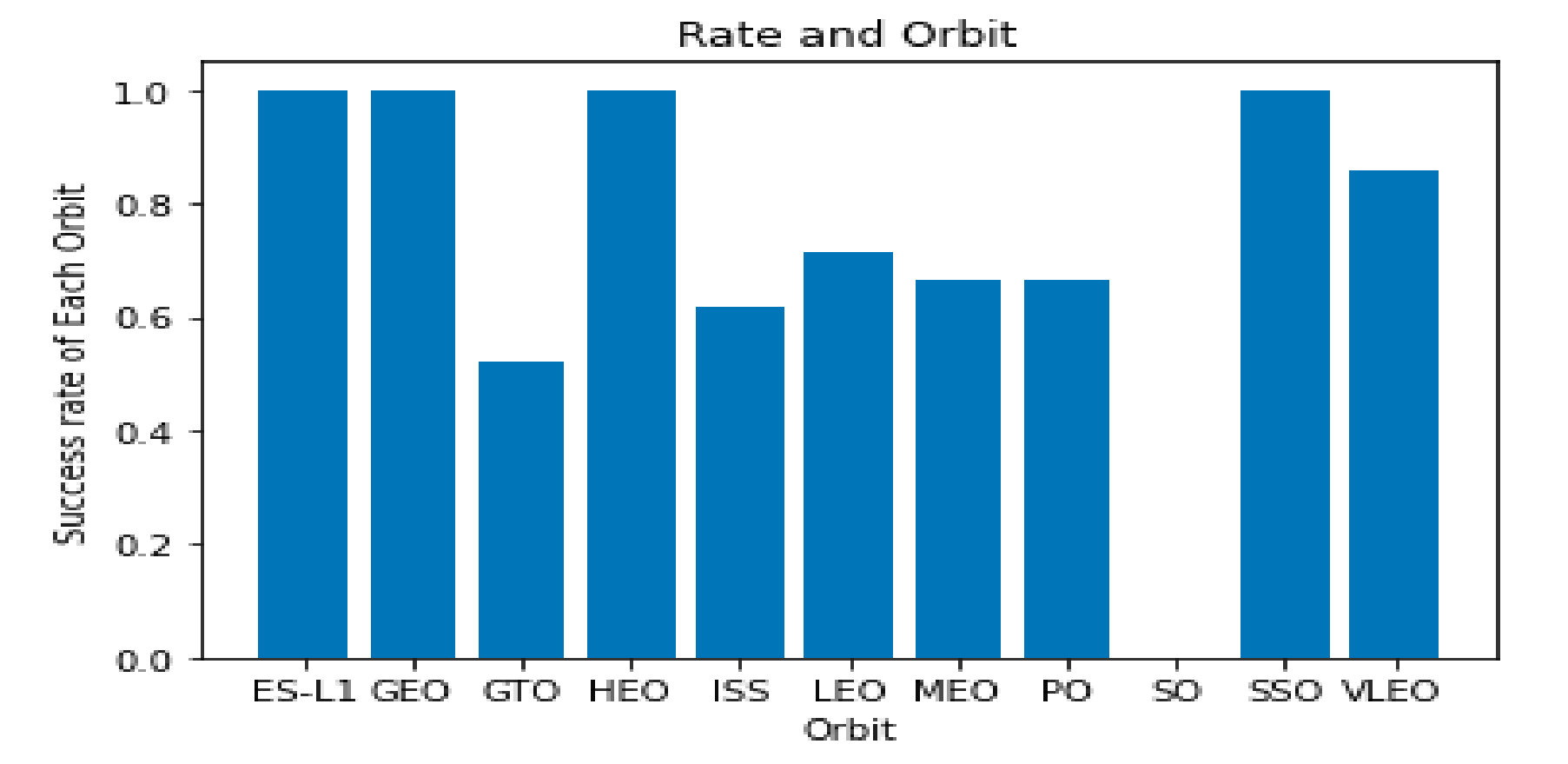
Payload vs. Launch Site

In this scatter plot, the largest payload mass renders the highest success rate for site CCAFS SLC 40, while it seems to be the opposite for site KSC LC 39A. The launch site VAFB-SLC has no activity greater than 1000kg of payload mass.



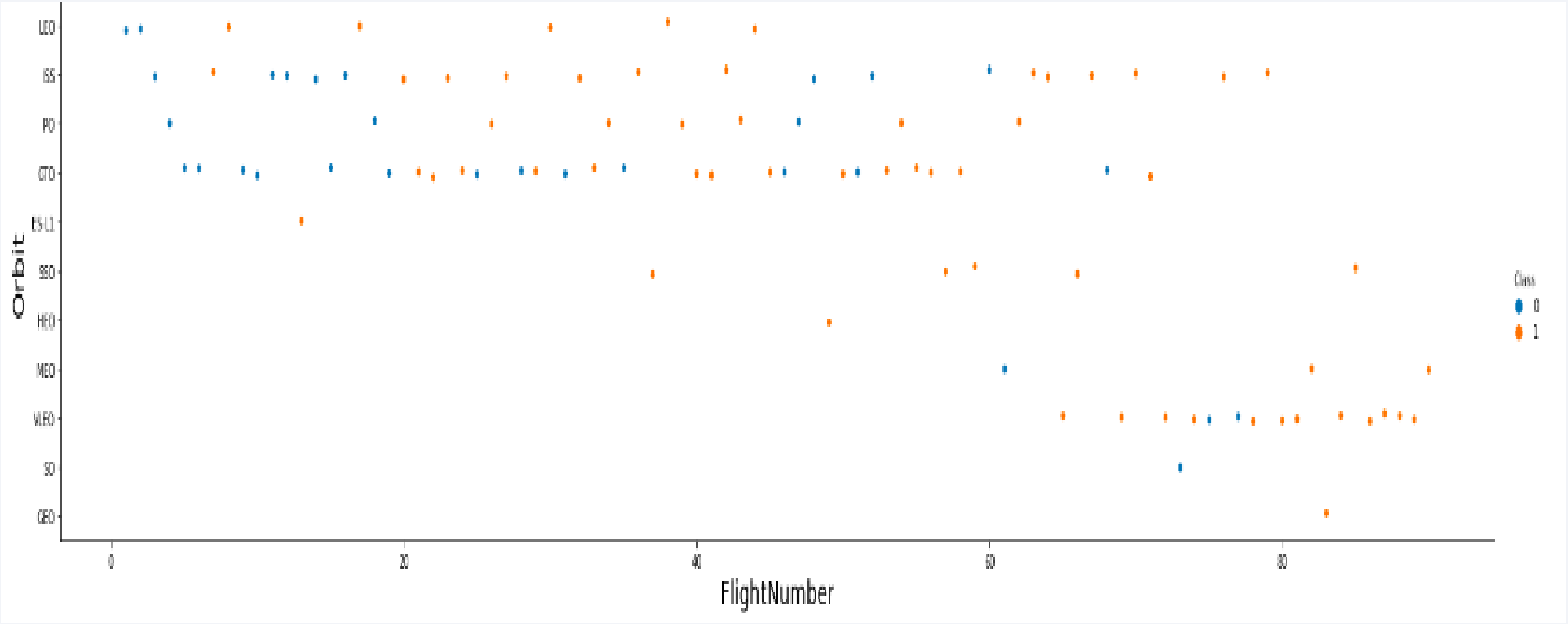
Success Rate vs. Orbit Type

The bar chart indicates there is a correlation between the orbit type and their success rate. The orbit types with the highest success rates are ES-L1, GEO, HEO, and SSO.



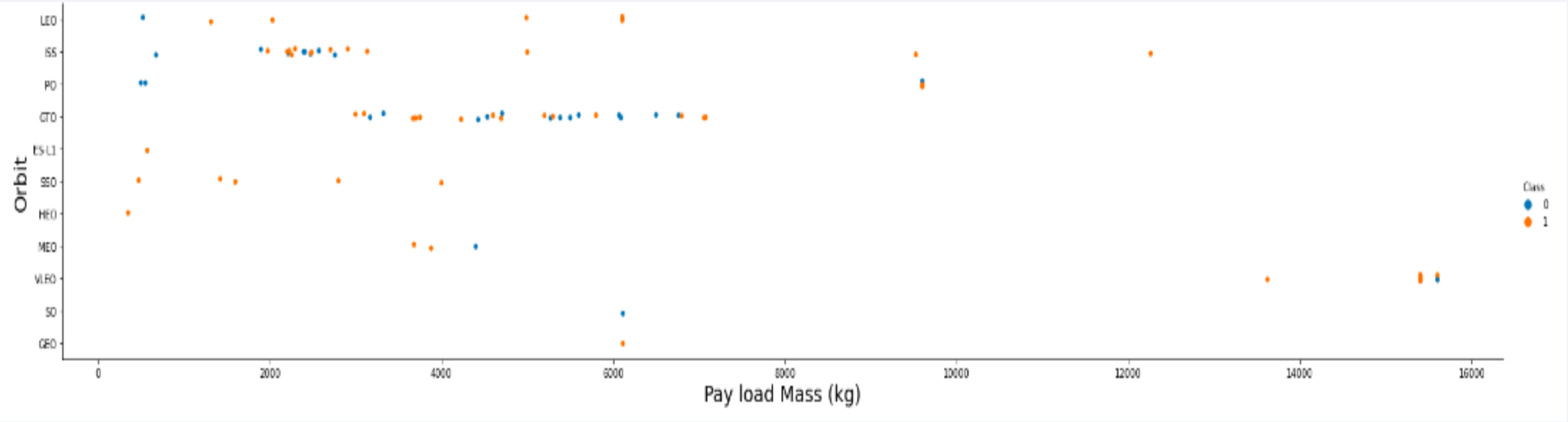
Flight Number vs. Orbit Type

In this scatter plot, the number of flights is seemingly affecting the success of the LEO orbit, whereas the GTO orbit is not affected by the flight number.



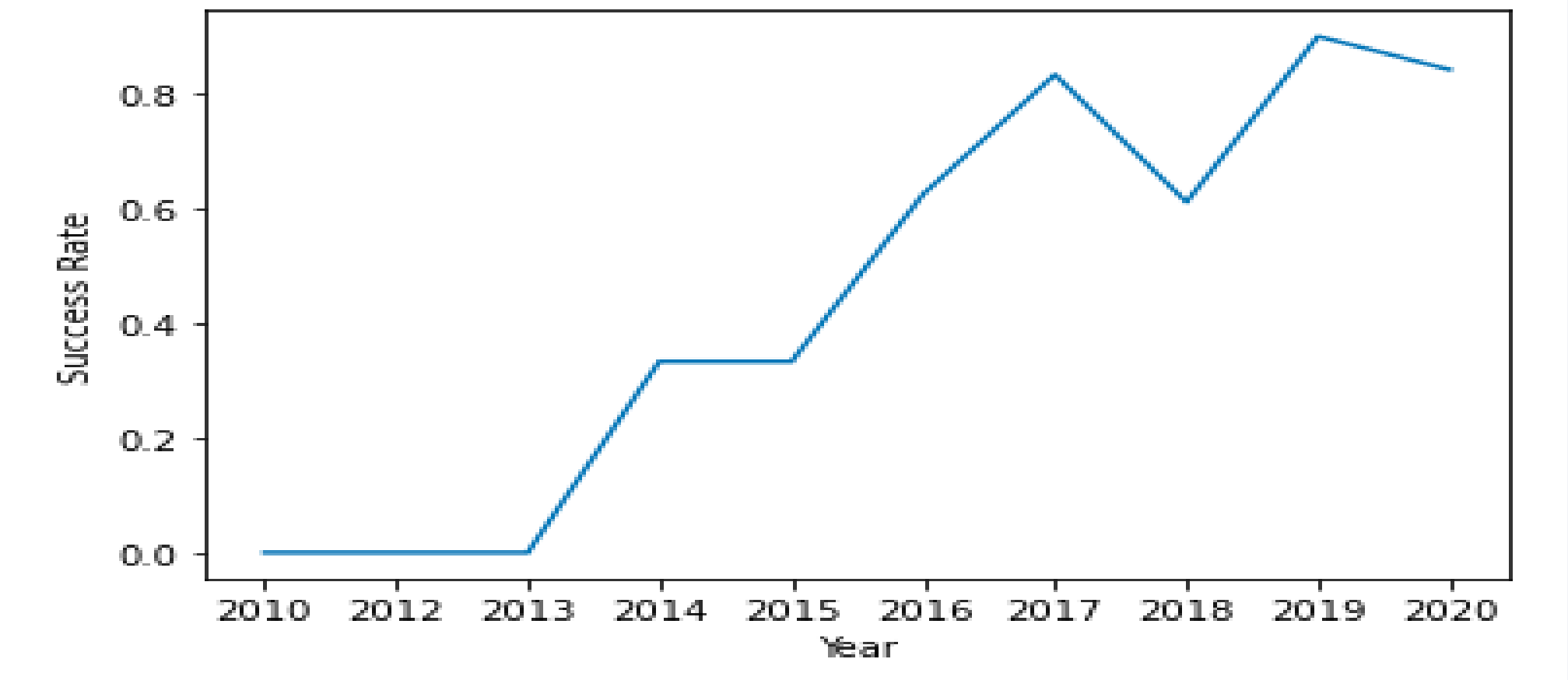
Payload vs. Orbit Type

This scatter plot is showing that for some orbit types, the payload has a positive correlation to it in which the heavier the payload mass, the higher the success rate. However, for some orbit types, it is not so obvious whether the payload mass is affecting it positively or negatively.



Launch Success Yearly Trend

The line chart shows that the first three years had a consistent rate of unsuccessful launches. However, starting in 2013, as the years increased, so did the success rate of the launches.



All Launch Site Names

We use the SQL **DISTINCT** statement for the database to select and show all the different launch site names without having any repeated.

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

Launch Site Names Begin with 'CCA'

Using the **LIKE** operator followed by the **%** allows us to find any value that starts with 'CCA.' In order to only display 5 records of this string, we use the **LIMIT** clause.

Display 5 records where launch sites begin with the string '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

Total Payload Mass

Using the **SUM** operator, we are able to calculate the total payload mass carried by boosters. In order to choose which customer we wanted to have the sum displayed for, we used the **WHERE** clause to specify.

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

```
%sql SELECT sum(payload_mass__kg_) as payload_mass from SPACEXTBL where (customer) = 'NASA (CRS)'
```

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

payload_mass
45596

Average Payload Mass by F9 v1.1

To calculate the average payload mass, we use **AVG** function. In order to choose the booster version we want displayed, we use the **WHERE** clause to specify.

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

```
%sql SELECT avg(payload_mass__kg_) as average_payload from SPACEXTBL where (booster_version) = 'F9 v1.1'
```

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

average_payload
2928.4

First Successful Ground Landing Date

On May 1st, 2017 the first successful ground pad landing was achieved. By using the **MIN** function, we are able to look for the date of the lowest value. Using the **WHERE** clause, we are able to specify which landing outcome we want to see the results of.

List the date when the first succesful landing outcome in ground pad was acheived.

Hint:Use min function

```
%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

The list of names below are the booster versions that had successful drone ship landing outcomes. We use the **WHERE** clause to specify the landing outcome we want results for. We use the **BETWEEN** function to display the payload mass within the given range.

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

```
%sql select Booster_Version from SPACEXTBL where "Landing _Outcome" = 'Success (drone ship)' and PAYLOAD_M
```

```
 * 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

The total number of successful and failure mission outcomes were calculated by using the **COUNT** function, which returns the number of rows that are in the mission outcome. The total was then separated into subgroups of the same type by using the **GROUP BY** statement.

List the total number of successful and failure mission outcomes

```
%sql SELECT Mission_outcome, count(Mission_Outcome) as Outcome from SPACEXTBL group by mission_outcome;
```

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

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

Boosters Carried Maximum Payload

In order to find the names of booster versions that have carried the maximum payload mass, we use a subquery within the other query statement. This includes using a **WHERE** clause and the **MAX** function as a way to be specific.

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

```
%sql SELECT Booster_version,payload_mass__kg_ from SPACEXTBL where payload_mass__kg_ = (select max(payload_mass__kg_)from SPACEXTBL);
```

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

Booster_Version	PAYLOAD_MASS__KG_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

2015 Launch Records

During the month of January and April of the year 2015, there were two different booster versions that experienced failure in its drone ship landing. These unsuccessful landing outcomes both were executed at the same launch site CCAFS LC-40.

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.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.

```
BOOSTER_VERSION, LAUNCH_SITE,"Landing _Outcome", MISSION_OUTCOME FROM SPACEXTBL WHERE "Landing _Outcome" =
```

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

Date	Booster_Version	Launch_Site	Landing _Outcome	Mission_Outcome
10-01-2015	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)	Success
14-04-2015	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)	Success

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

Between the years 2010 and 2017, there were 14 successful landing outcomes. Of those 14 successful outcomes, 6 were ground pad landings, while the other 8 were drone ship landings.

Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

```
l SELECT "Landing _Outcome", DATE, COUNT(*) AS COUNT_SUCCESS FROM SPACEXTBL WHERE DATE BETWEEN '04-06
```

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

Landing _Outcome	Date	COUNT_SUCCESS
Success (ground pad)	18-07-2016	6
Controlled (ocean)	18-04-2014	3
Failure (drone ship)	10-01-2015	4
No attempt	08-10-2012	10
Success (drone ship)	08-04-2016	8

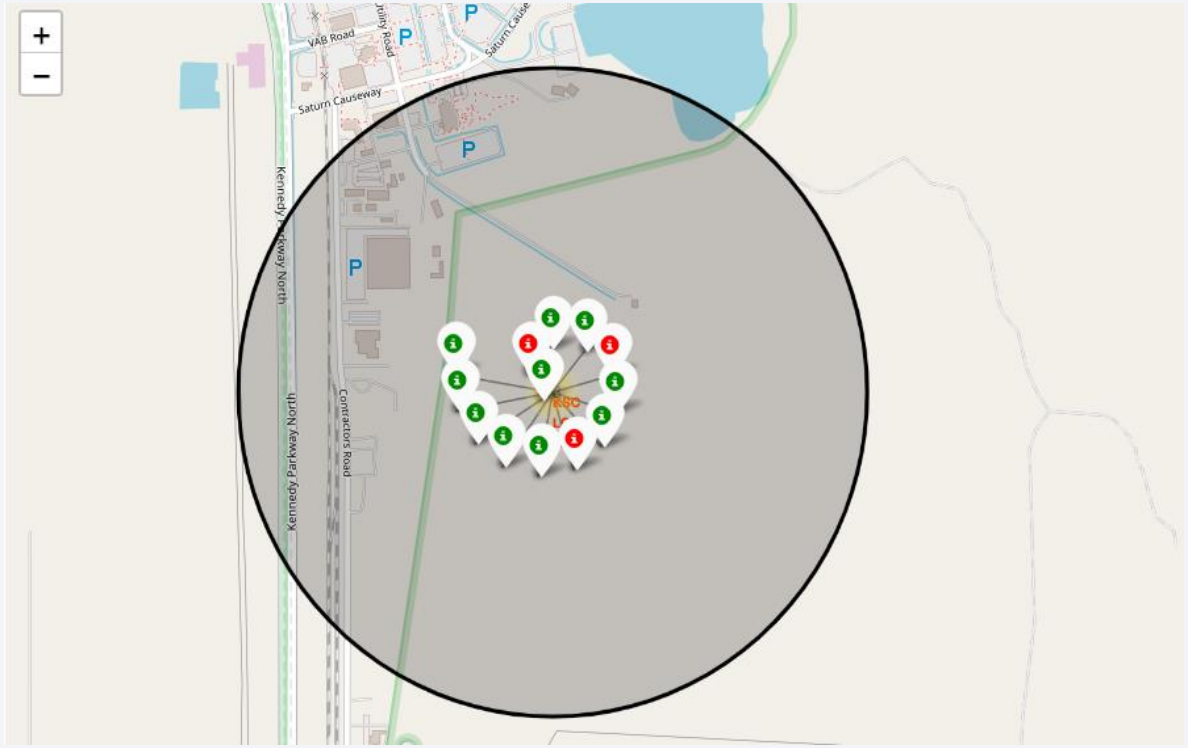

Section 3
Launch Sites
Proximities Analysis

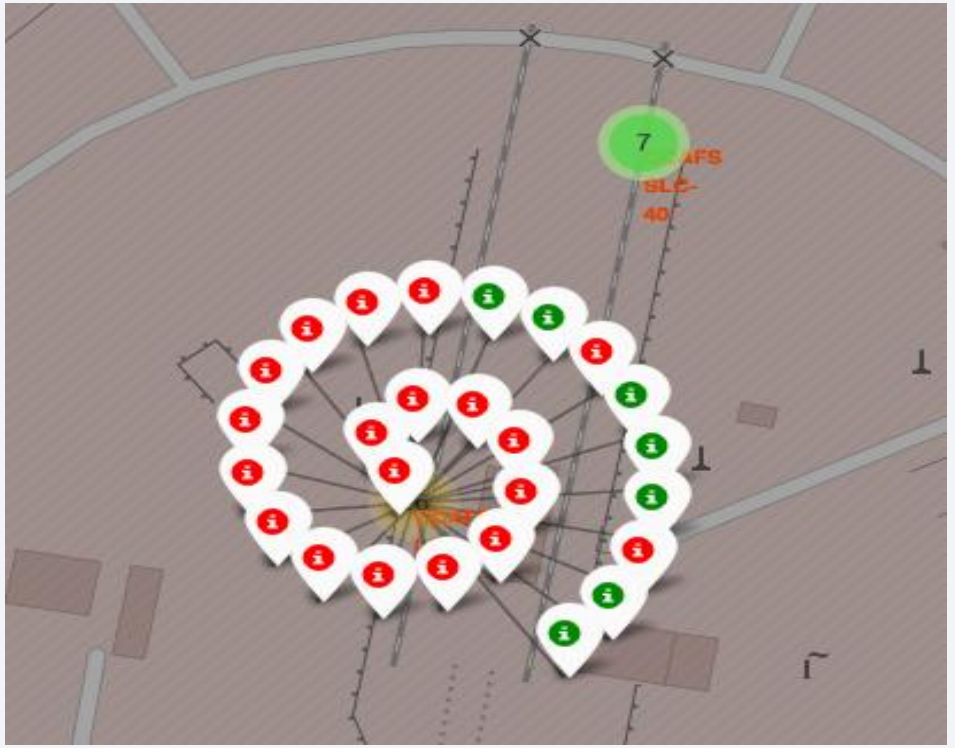
Folium Map Launch Sites

These maps show the markers of the launch sites both normally and zoomed in. The location coordinates are within the circle marker. The launch sites are not far in proximity to the equator line as well as the coast.

Folium Map Launch Outcomes

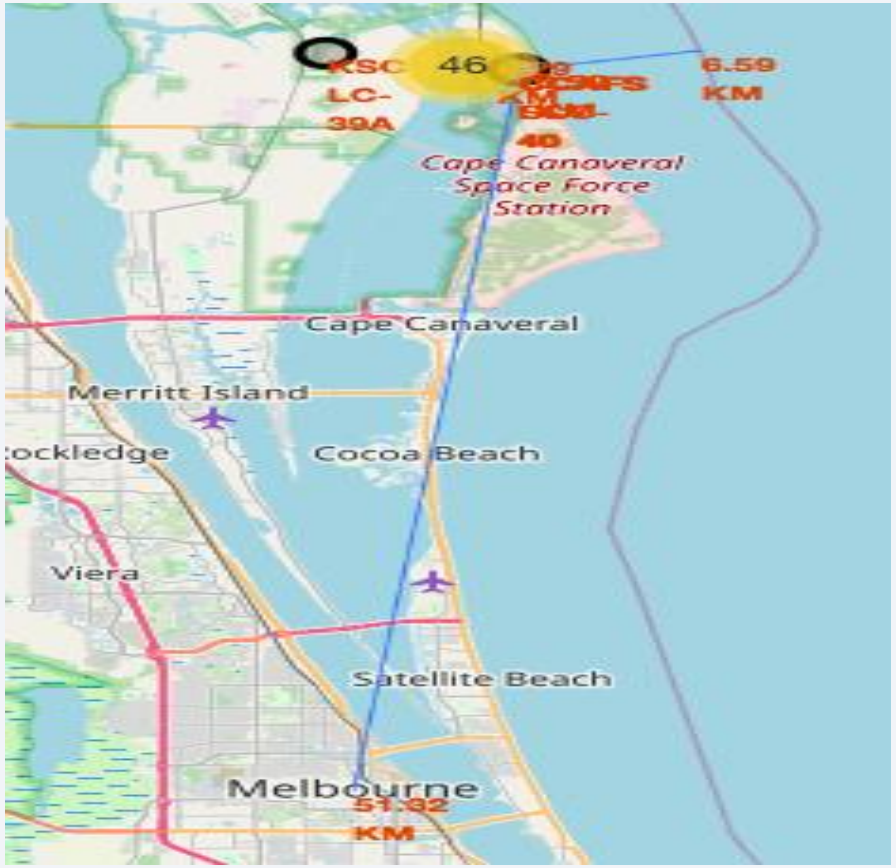
The screenshots of the maps are showing the different launch site locations. Once you zoom in, you are able to see the color labelled markers. These markers indicate the outcomes of the launches, green representing successful and red representing unsuccessful. Looking at both, the map on the left is displaying more green markers and therefore executing more successful launch outcomes at their sites.

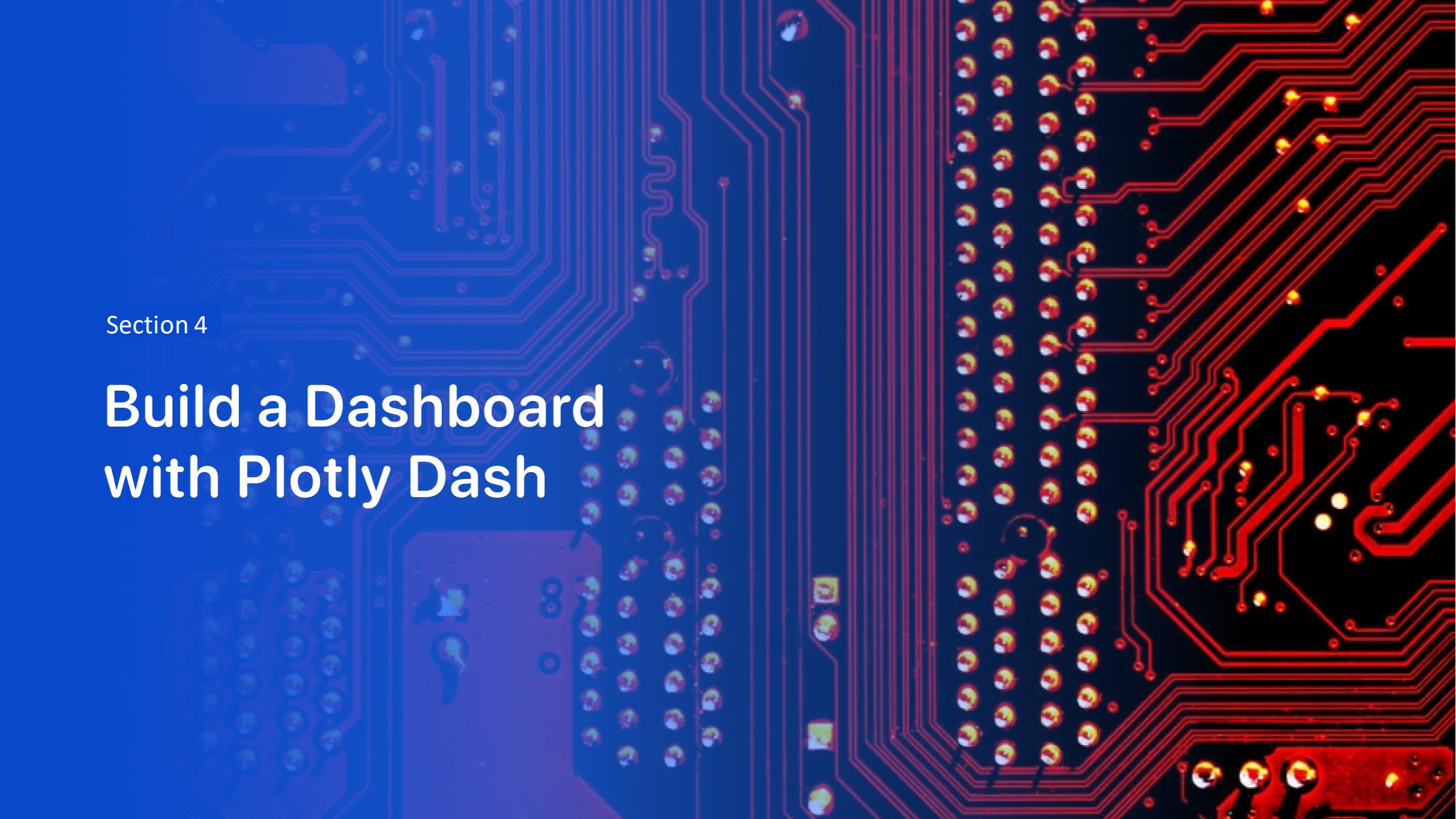




Folium Distance Map

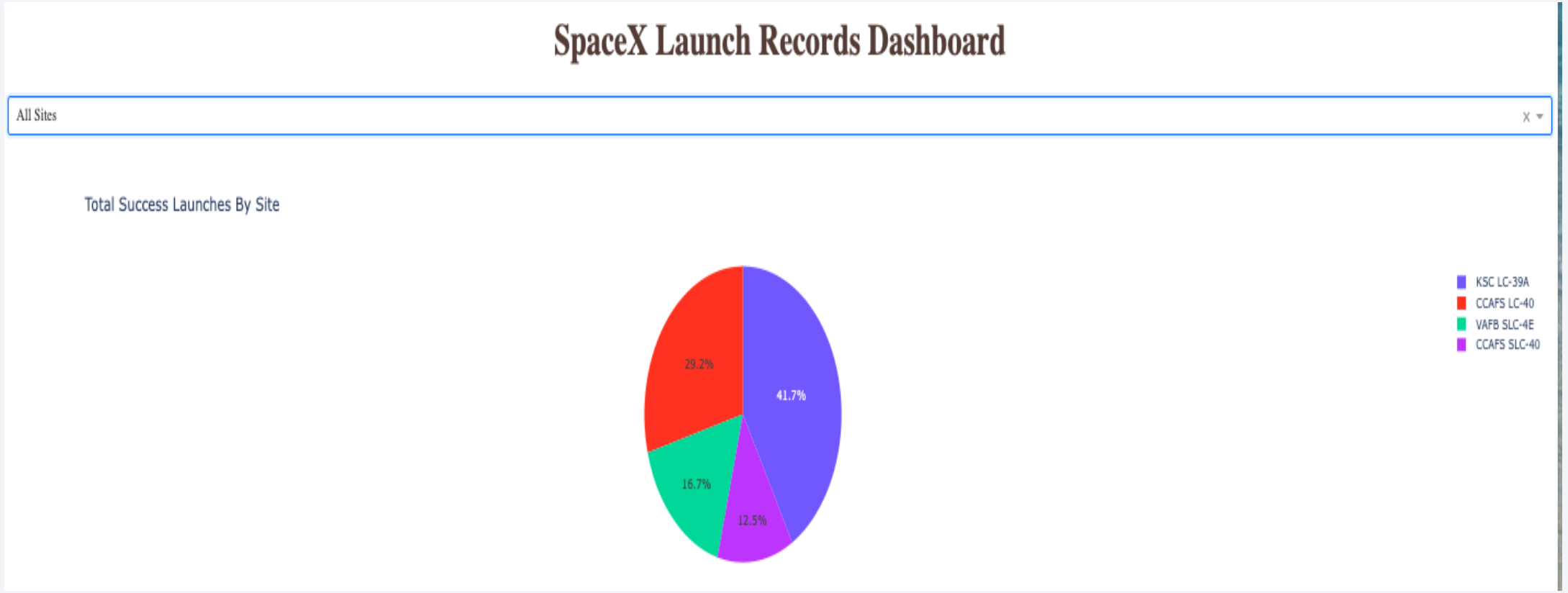
These maps show the distances between the launch site locations and their closest railway, highway and city. The proximity of these locations are connected by a polyline generated by longitude, latitude and weight. The findings of this map were that the launch sites are located near railways and highways, but are quite far from cities.




Section 4
Build a Dashboard with Plotly Dash

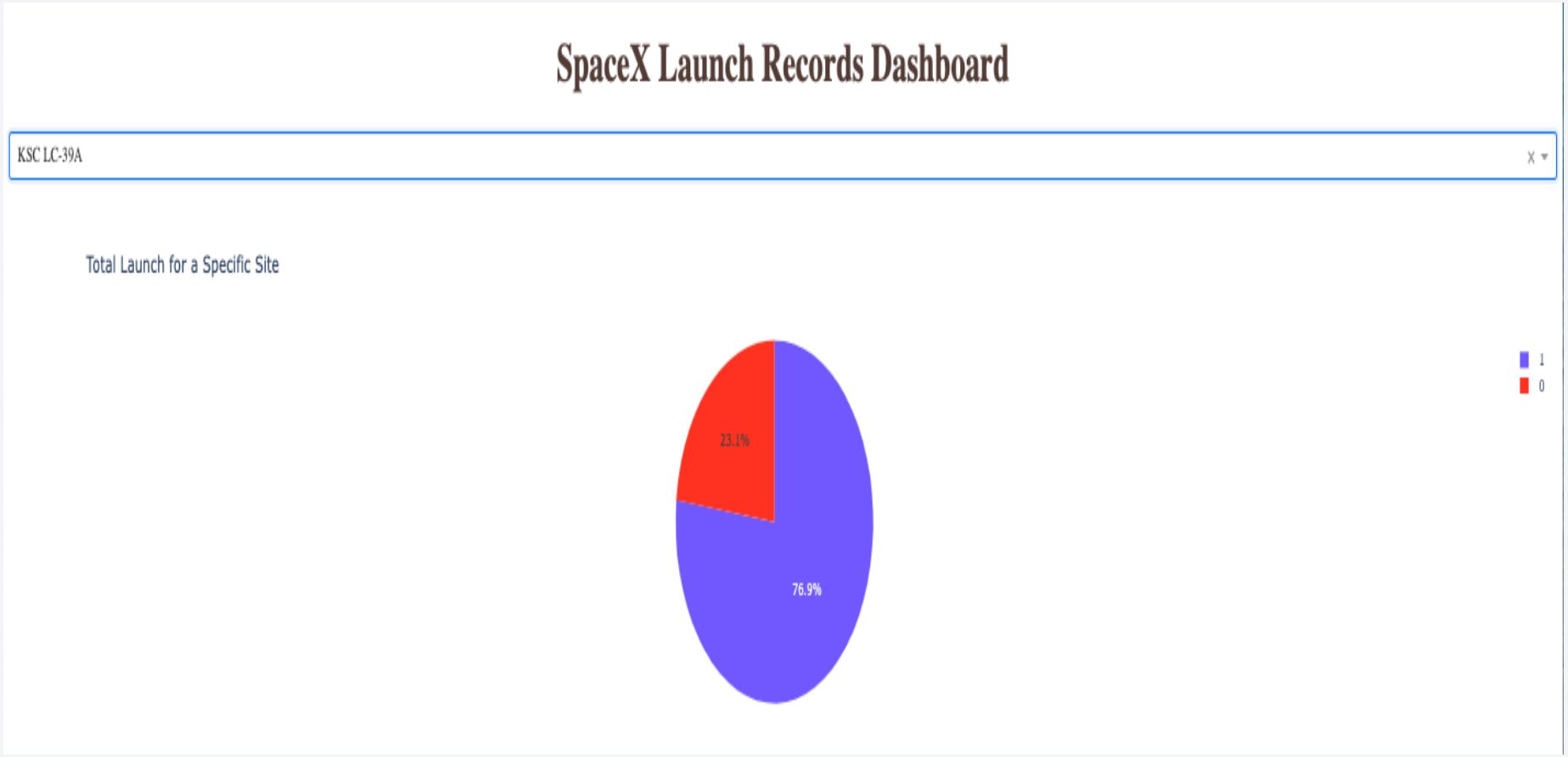
Total Successful Launches

This graph shows the success rate of each launch site. The site KSC LC- 39A has the highest with a 41.7% rate of total successful launches.



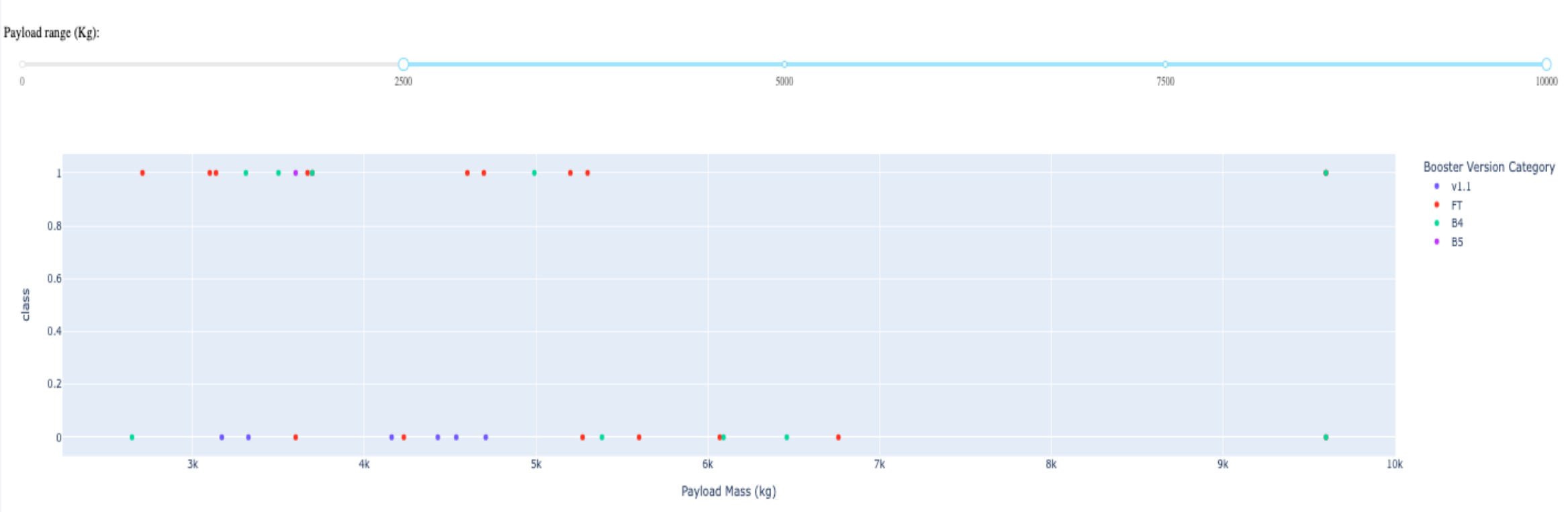
Highest Launch Success Ratio of A Site

The pie chart below displays the highest launch success ratio of a site. The launch site that is showing 76.9% success rate to a 23.1% failure rate is KSC LC-39A.



All Sites vs. Payload

This graph shows that given a lower payload mass within a larger range, the success rate of launches will be greater. This seems to apply for most booster versions.



Section 5

Predictive Analysis (Classification)

Classification Accuracy

Given the following accuracy test results from executing the score method, the decision tree model has the highest classification accuracy.

TASK 12

Find the method performs best:

In [34]:

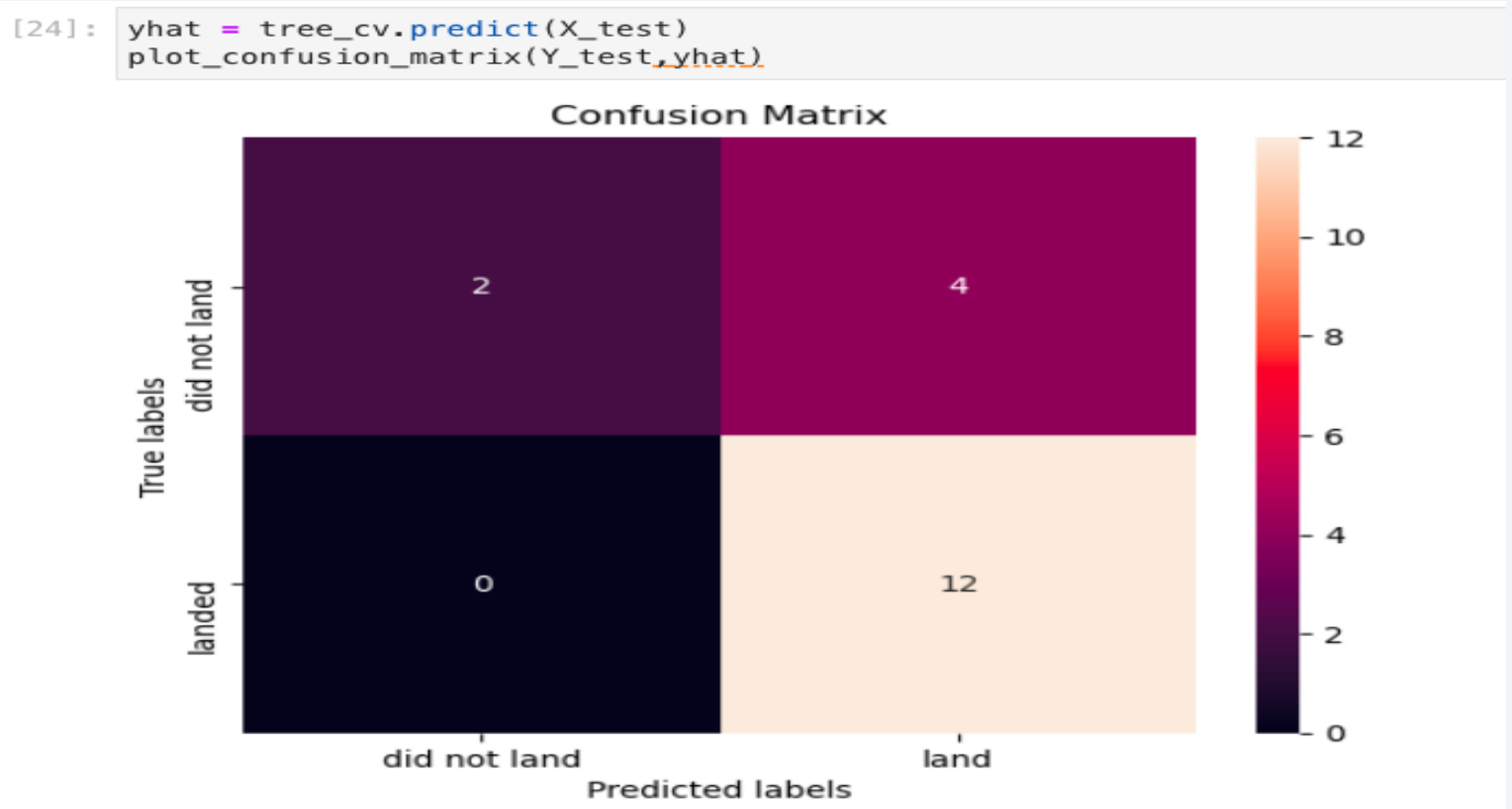
```
print("accuracy for logistic regression method :",logreg_cv.score(X_test,Y_test))

print("accuracy for supervised vector machine method :",svm_cv.score(X_test,Y_test))

print("accuracy for decision tree method :",tree_cv.score(X_test,Y_test))

print("accuracy for k nearest neighbour method :",knn_cv.score(X_test,Y_test))
```

```
accuracy for logistic regression method : 0.8333333333333334
accuracy for supervised vector machine method : 0.8333333333333334
accuracy for decision tree method : 0.9444444444444444
accuracy for k nearest neighbour method : 0.8333333333333334
```

Confusion Matrix

The confusion matrix for the decision tree classifier shows that there are 14 labels that were identified accurately, which is what we are trying to maximize. The four false positives cause the level of accuracy to be lower by having the labels inaccurately identified.

```
[24]: yhat = tree_cv.predict(X_test)
      plot_confusion_matrix(Y_test,yhat)
```



Conclusions

We can conclude that:

- A flight number increase positively affects the success rate of the launch sites
- ES-L1, GEO, HEO, and SSO are the orbit types that have the highest success rates
- KSC LC- 39A is the launch site with the highest launch success rate
- The landing type, booster version and launch site can all be possible factors contributing to the first stage crashing
- The best performing classification model with the greatest accuracy is the decision tree classifier

Appendix

Github doesn't always render images of Jupyter notebook therefore, I have linked the nbviewer.org versions to make sure everything is visible.

- Data collection rendered link
- Web scraping rendered link
- Data wrangling rendered link
- Data visualization rendered link
- EDA with SQL rendered link
- Interactive folium map rendered link
- Dashboard rendered link
- Predictive analysis rendered link
- Github capstone repository

Thank you!