IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Keon Ung Park
Nov. 21, 2023

Outline

Executive Summary

The main objective of this report is to outline the method to **best predict the landing of the 'first phase' of the rocket after launching.** This is to reduce the costs of rocket launching because the 'first phase' is the biggest and the most expensive part of the rocket.

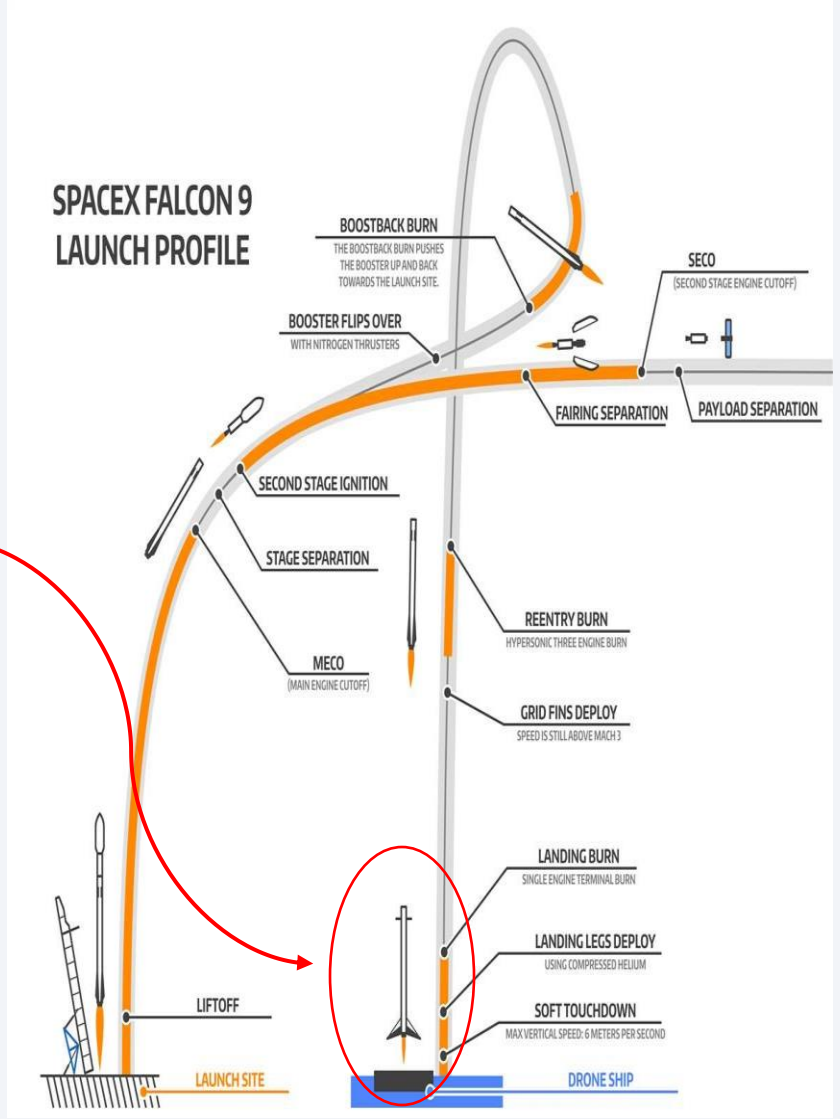
The methodologies to approach this objective is by first understanding different data and being able to **clean and wrangle to predict the success rate.** The next step is the use different tools to **visualize the data.** From the visualized data the factors of variables that affect the success rate the most can be easily understood.

After observing the visualization and extracting the most valuable factors (independent variables), data preprocessing is done to be able to **run different models for classification** such as Logistic Regression, K-Nearest Neighbors, Support Vector Machine, and Decision Trees.

The results show that all four classification models vary in the train set evaluation; however, **have same outcomes** for test set prediction.

Introduction

- Main Objective:
  - Make SpaceY the leading rocket company.
  - Minimize the costs of launching a rocket: retrieve 'first stage'.
- Background and Context:
  - Current leading rocket company is SpaceX.
  - Reason being the ability to retrieve and reuse the 'first stage' of the rocket.
- Problems and Questions:
  - Which variables contribute most to the success of 'first stage' retrieval?
  - Which Machine Learning algorithm would best predict the success of 'first stage' retrieval?

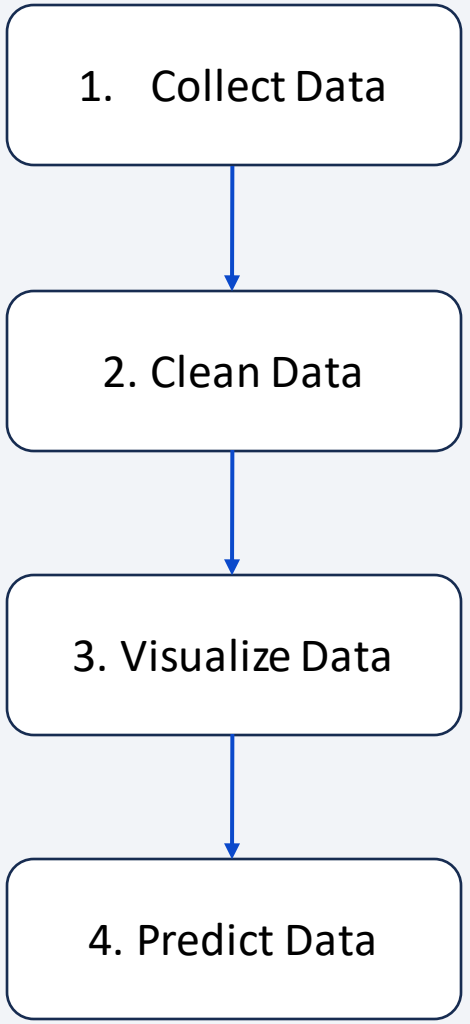


Section 1

Methodology

Methodology

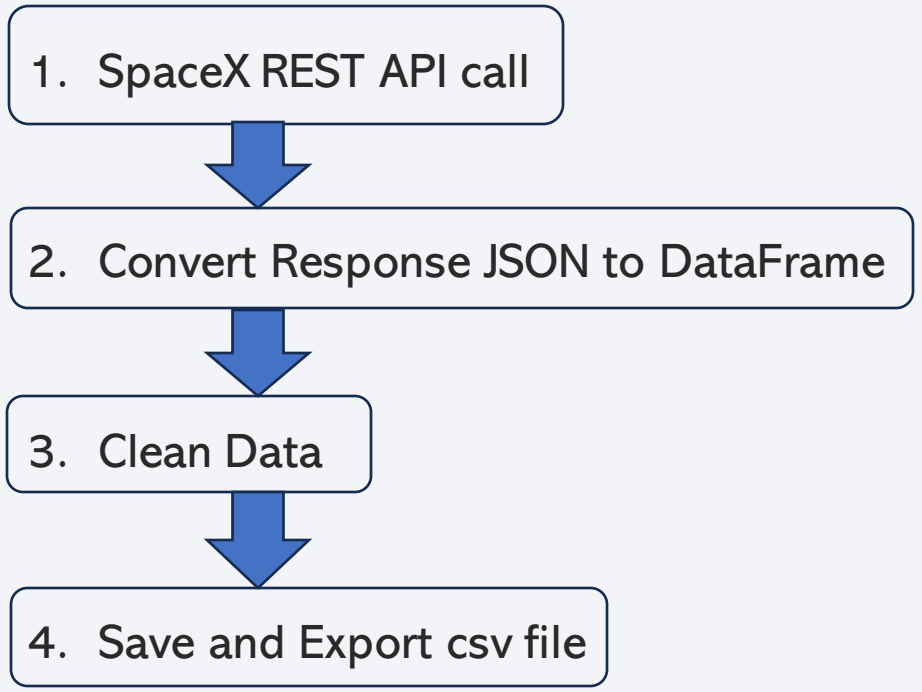
1. **Data Collection:**
   - SpaceX API
   - Web Scraping
2. **Data Wrangling:**
   - Study Data
   - Preprocess Data
3. **Interactive Visual Analytics**
   - Folium
   - Plotly Dash
4. **Predictive Analysis (Classification Models)**
   - Machine Learning Models
     a) Build
     b) Evaluate
     c) Compare



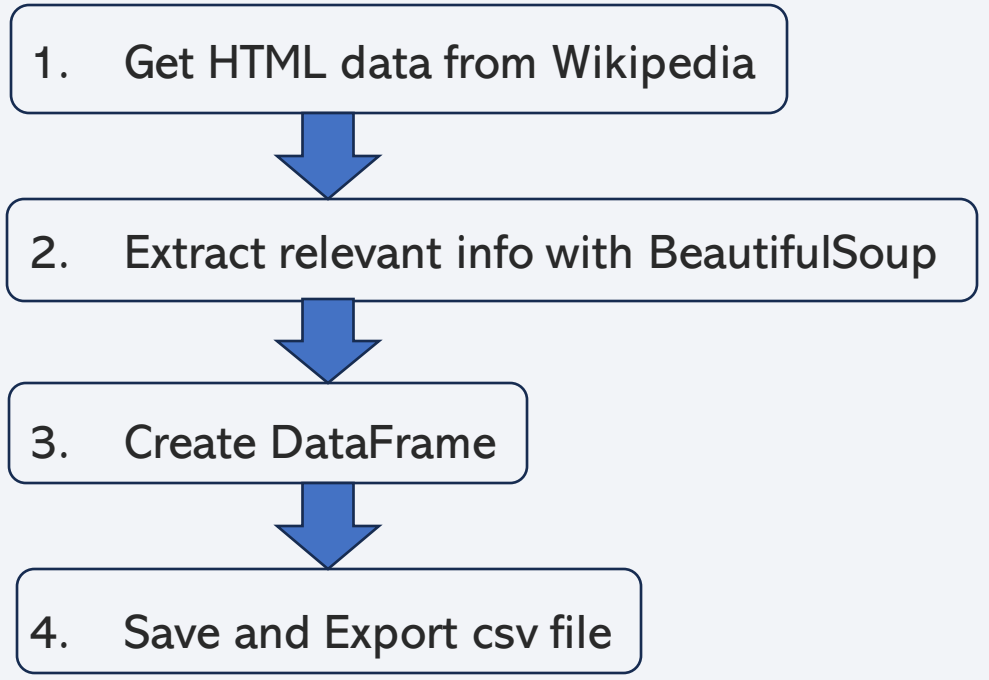
Data Collection

The Datasets were collected by:

- **SpaceX's REST API**



- **Wikipedia's Falcon9 data**



Data Collection – SpaceX API

1. SpaceX REST API calls
2. Parse and Filter Data into DataFrame
3. Clean Data
3. Save and Export csv

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

There were more API calls realized to extract various data tweaking the value in the red circle, this is just an example

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
# Create a data from launch_dict
df_launch = pd.DataFrame(launch_dict)
```

A DataFrame is created by different variables

```
# Calculate the mean value of PayloadMass column
payloadmass_mean = data_falcon9['PayloadMass'].mean()
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, payloadmass_mean)
```

Clean the data by checking for missing values, replace them with the mean of its column

```
data_falcon9.to_csv('dataset_part_1.csv')
```

Save and Export dataset to a csv file

Click here for the full process and code in: Github repository

Data Wrangling

- The purpose of this project is to be able to predict the 'outcome' of the 'first phase' landing for reusability.
- The 'Outcome' list is as follows:
  - True ASDS
  - None None
  - True RTLS
  - False ASDS
  - True Ocean
  - False Ocean
  - None ASDS
  - False RTLS
- The dataset has to be wrangled to have a column with binary values being: 0 is Failure and 1 is Success for machine learning.
- All outcomes that contain 'True' is success and everything else is failure.

1. Check for all outcomes

```
# landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()
landing_outcomes
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
```

2. Classify the data

```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

3. One hot encoding: Column

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = [0 if outcome in bad_outcomes else 1 for outcome in df['Outcome']]
```


Click here for the full process and code in: Github repository

EDA with Data Visualization

- There are three types of graphs: Scatter, Bar, Line.
- **Scatter**
  - Flight Number vs. Launch Site
  - Payload vs. Launch Site
  - Flight Number vs. Orbit Type
  - Payload vs. Orbit Type
- **Bar**
  - Success Rate vs. Orbit Type
- **Line**
  - Launch Success Yearly Trend

Scatter plots are chosen to see the relationships that two independent variables hold on the success rate. (correlation)

Bar graph is to organize a categorical independent variable's value on the dependent variable

Line graphs are best to show a trend (temporal data)

Click here for the full process and code in: Github repository

EDA with SQL

- **SQL Queries**
  - All Launch Site Names
  - Launch Site Names Begin with 'CCA'
  - Total Payload Mass
  - Average Payload Mass by F9 v1.1
  - First Successful Ground Landing Date
  - Successful Drone Ship Landing with Payload between 4000 and 6000
  - Total Number of Successful and Failure Mission Outcomes
  - Boosters Carried Maximum Payload
  - 2015 Launch Records
  - Rank Landing Outcomes between 2010-06-04 and 2017-03-20

There are various SQL queries to study and understand the dataset.

Click here for the full process and code in: Github repository

Build an Interactive Map with Folium

- There are three maps:

1. **All Launch Site Locations**

   - Circles are used to pinpoint the launch site locations and markers are used to label the name of the launch site

   → To first get a bigger picture of the location of the Launch Sites

2. **Clusters of Launching by Location with Outcomes**

   - Marker cluster is used to group individual launches by launch site and color is added to each individual launch: green is success and red is failure

   → To have a better visualization of success and failure of launches by launch site on a map

3. **Infrastructure Proximities**

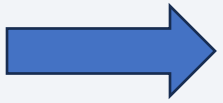
   - Mouse position is used to manually pinpoint the location of infrastructure proximities and find the latitude and longitude of each point. Then polyline is used to draw a line from the launch site to those proximities. Finally marker is used to label the distance to that point calculated by the Harvesine Formula.

   → To understand different aspects of requirements or characteristics of a Launch Site in relation to geography

Click here for the full process and code in: Github repository

Build a Dashboard with Plotly Dash

- There are three Interactive Dashboards:
- **Success Rate Ratio of All Launch Sites**
  - This can be observed by default (or select 'ALL' in the 'Select Launch Site' dropdown feature) with the first pie chart that appears, it shows a ratio of success rate by Launch Site



To visualize the comparison of success rate by Launch Site

- **Launch Site with Highest Success Rate**
  - By looking at the ratio with the highest success rate from the Dashboard above, select 'KSC LC-39A' to see the ratio of success to failure. Other individual site's individual ratio of success to failure can be observed by choosing through the dropdown feature.

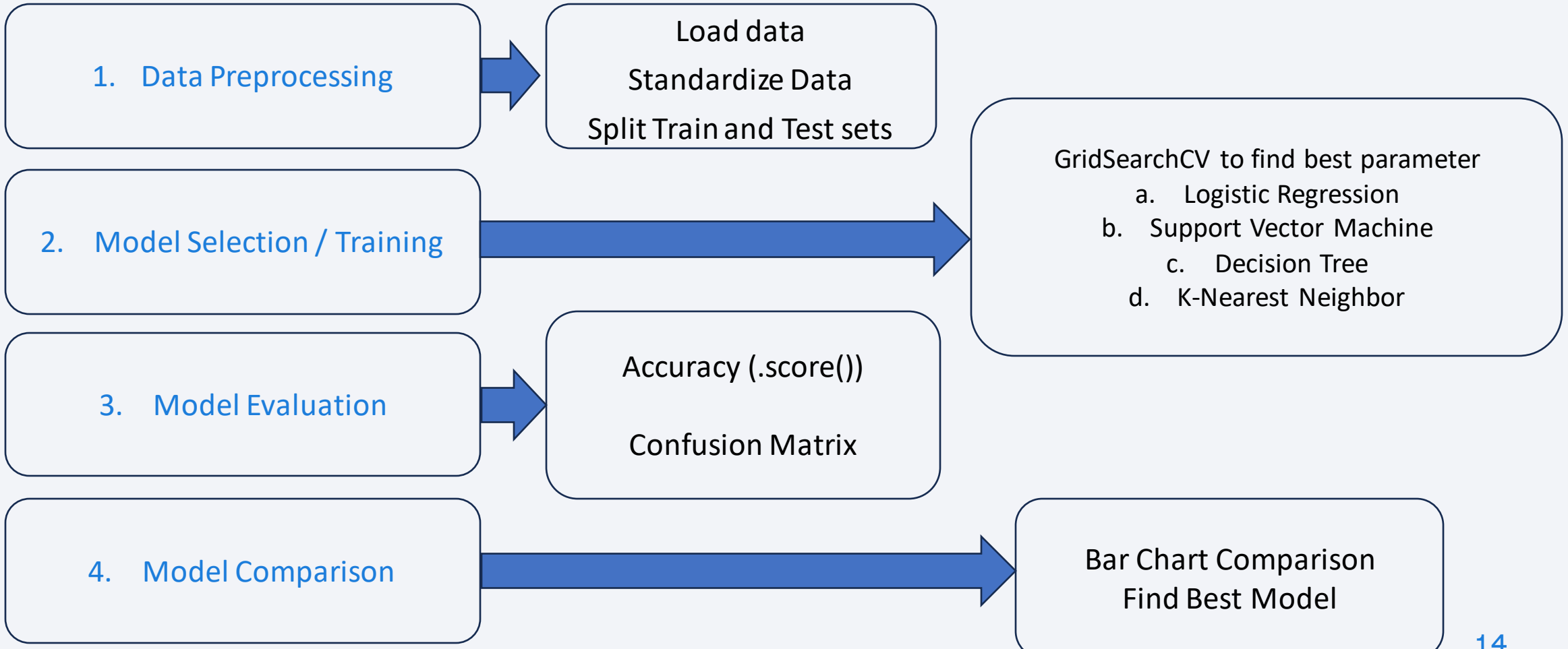
To specifically understand if it is only relatively successful or not

- Payload vs. Launch Outcome Interactive Scatter Plot
  - There is a scatter plot below the pie chart for any of the options in the dropdown feature. This shows the correlation between payload and launch outcome. There is also a slider right on top of the scatter plot to closely observe specific ranges of Payload Mass.

To analyze what range of Payload Mass is the most optimal for success

Click here for the full process and code in: Github repository

Predictive Analysis (Classification)



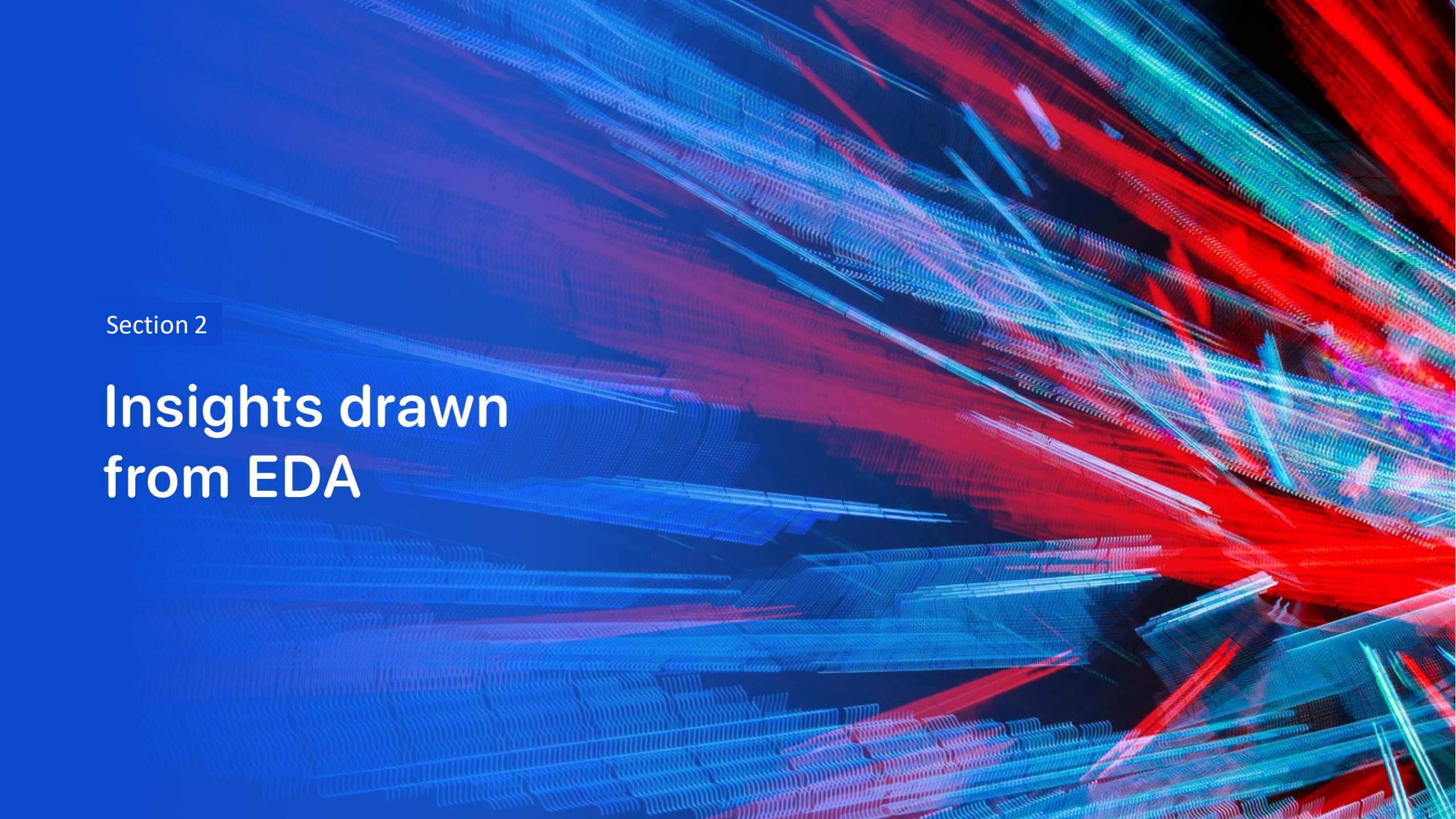
Click here for the full process and code in: Github repository

Results

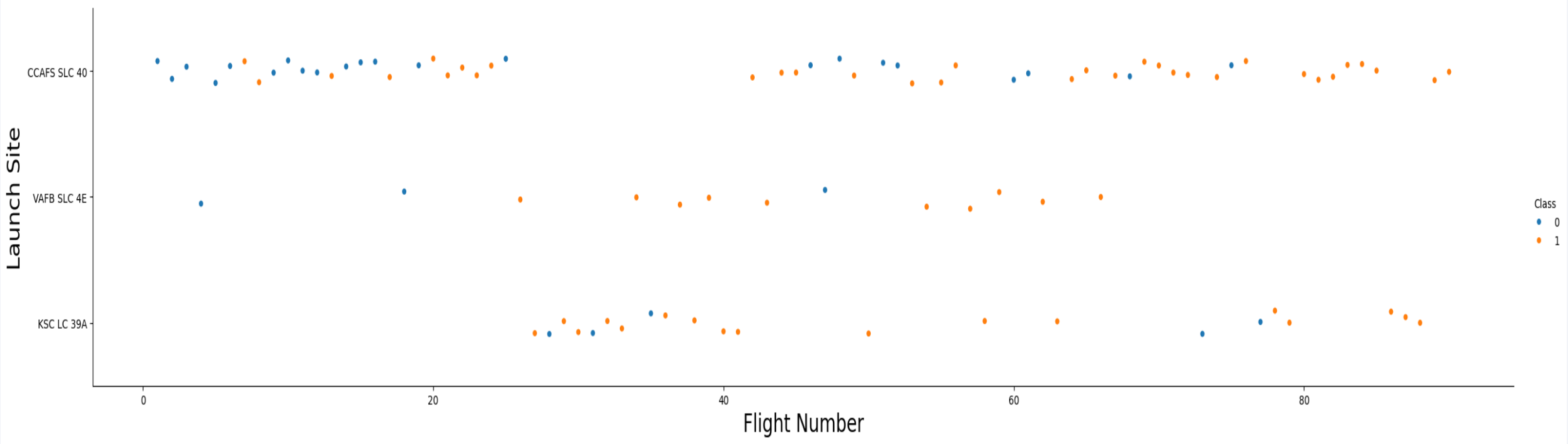
- Exploratory data analysis results
  - … section 2 Insights drawn from EDA
- Interactive analytics demo in screenshots
  - … Launch Sites Proximities Analysis
  - … Build a Dashboard with Plotly Dash
- Predictive analysis results
  - … Predictive Analysis (Classification)

Section 2

Insights drawn from EDA

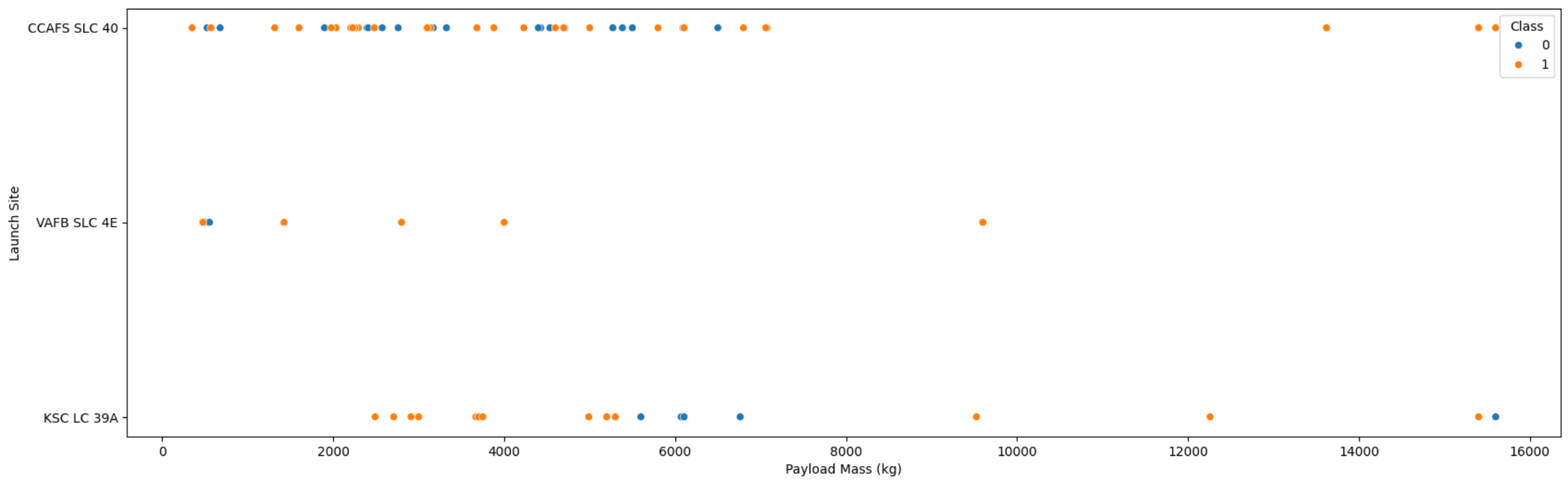


Flight Number vs. Launch Site



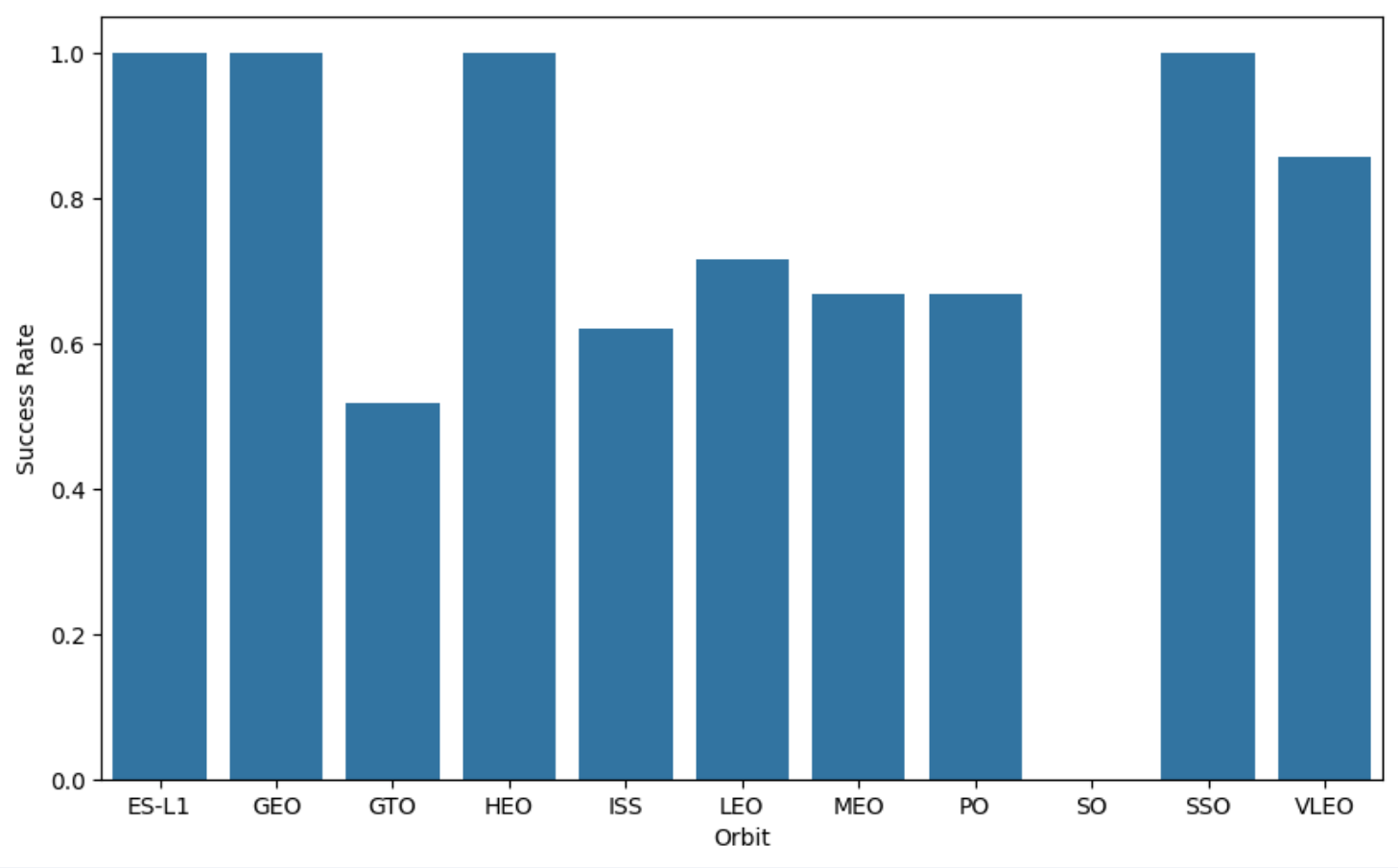
- This is a scatter plot with 'Flight Number' in the x-axis and 'Launch Site' in the y-axis
- Orange dots mean 'Success' while blue dots mean 'Failure'.

Payload vs. Launch Site



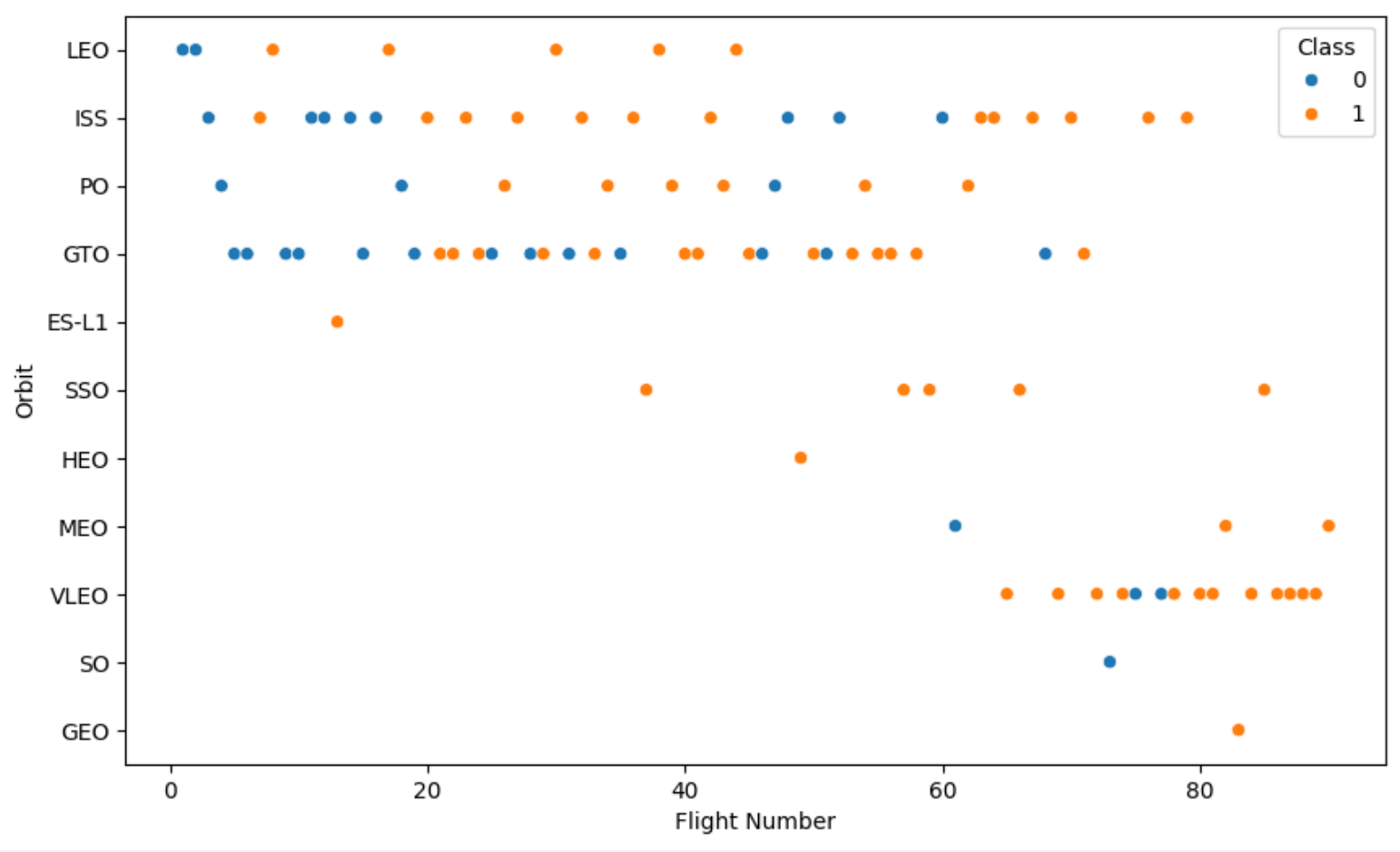
- This is a scatter plot with 'Payload Mass' in the x-axis and 'Launch Site' in the y-axis
- Orange dots mean 'Success' while blue dots mean 'Failure'.

Success Rate vs. Orbit Type



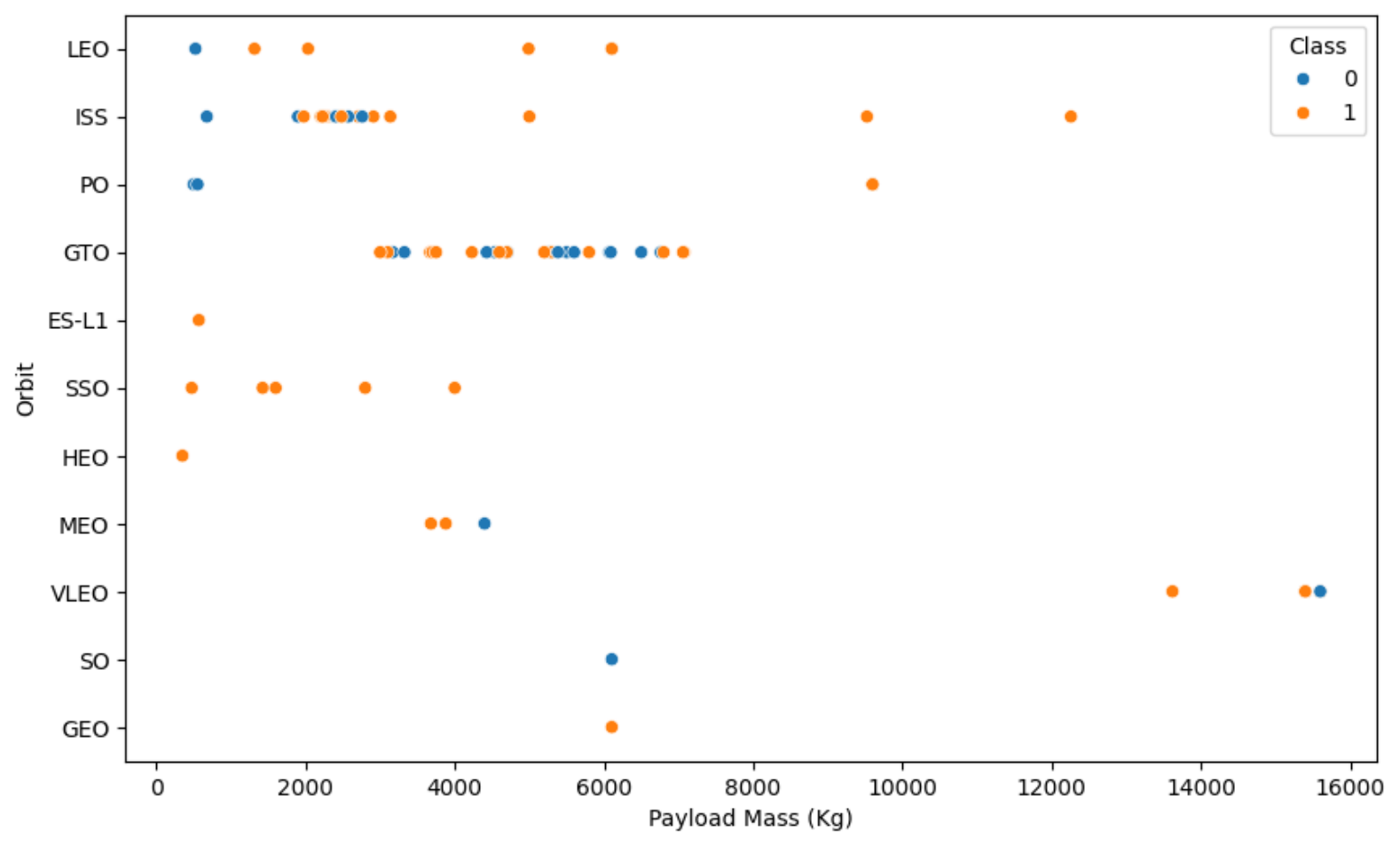
- This is a bar chart with 'Orbit' in the x-axis and 'Success Rate' in the y-axis.
- It can be seen that launching to certain orbits have a better success rate than others.
- However, the trial amount to each orbit should be considered

Flight Number vs. Orbit Type



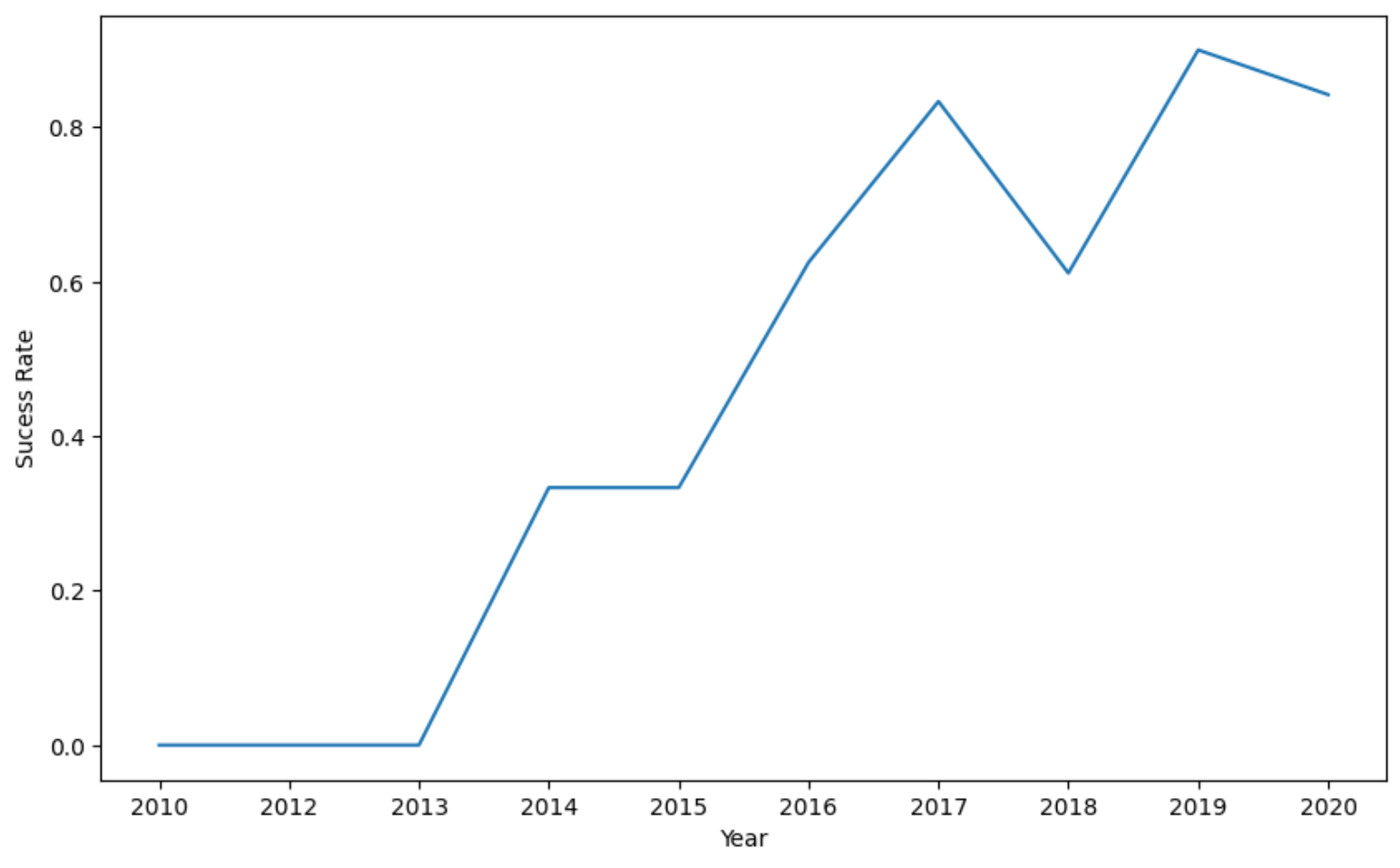
- This is a scatter plot with 'Flight Number' in the x-axis and 'Orbit' in the y-axis.
- Orange dots mean 'Success' while blue dots mean 'Failure.
- It can be seen that some orbits are more favorable than others.

Payload vs. Orbit Type



- This is a scatter plot with 'Payload Mass' in the x-axis and 'Orbit' in the y-axis.
- Orange dots mean 'Success' while blue dots mean 'Failure.
- It can be seen that payload mass is taken into consideration to certain orbits.

Launch Success Yearly Trend



- This is a line chart that shows the trend of success rate throughout the years.
- It shows that there has been a steady increase before falling down during 2018 and has gone back up.

All Launch Site Names

Query → Output

```
%%sql
SELECT DISTINCT "Launch_Site"
FROM SPACEXTBL;
```

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

Logic:

DISTINCT is used to retrieve the unique values of Launch Site.

Launch Site Names Begin with 'CCA'

Query → Output

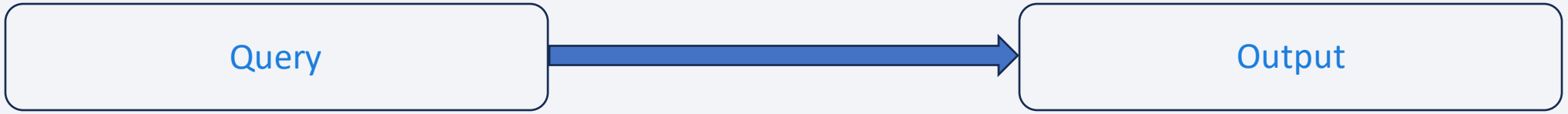
```
%%sql
SELECT *
FROM SPACEXTBL
WHERE "Launch_Site" LIKE 'CCA%'
LIMIT 5;
```

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	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	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

Logic:

WHERE is used to make a mask and **LIKE** is used to apply a value to the mask.
The CCA means anything that starts with CCA and ends with any other character for %.
LIMIT is used to get only the first 5 values.

Total Payload Mass




```
%%sql
SELECT SUM("PAYLOAD_MASS__KG_") AS "Total Payload Mass (NASA)"
FROM SPACEXTBL
WHERE "Customer" == "NASA (CRS)";
```

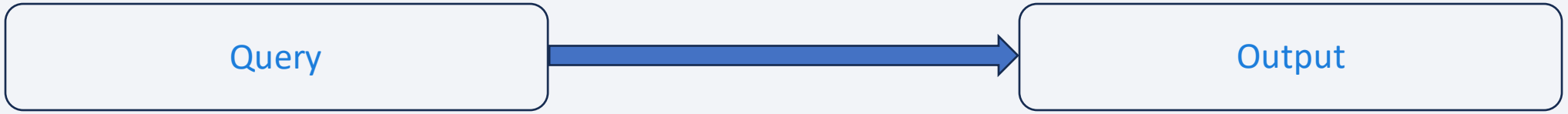
Total Payload Mass (NASA)
45596

Logic:

SUM is used to get the total amount.

WHERE is used to create a mask with a boolean to search for NASA(CRS)'s Payload Mass.

Average Payload Mass by F9 v1.1




```
%%sql
SELECT AVG("PAYLOAD_MASS__KG_") AS "Average Payload Mass (F9 v1.1)"
FROM SPACEXTBL
WHERE "Booster_Version" LIKE "F9 v1.1%";
```

Average Payload Mass (F9 v1.1)
2534.6666666666665

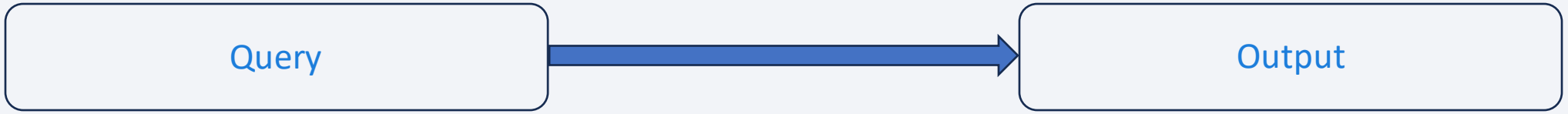
Logic:

AVG is used to get the average amount.

WHERE is used to create a mask with a Booster version that

LIKE matches names that start with 'F9 v1.1' and ends with any other set of characters %.

First Successful Ground Landing Date




```
%%sql
SELECT MIN("Date") AS "First successful landing"
FROM SPACEXTBL;
```

First successful landing
2010-06-04

Logic:

MIN is used to get the lowest / smallest value, hence the date with the smallest value (First Date of Success).

Successful Drone Ship Landing with Payload between 4000 and 6000

Query → Output

```
%%sql
SELECT "Booster_Version"
FROM SPACEXTBL
WHERE "Landing_Outcome" LIKE "Success%" AND ("PAYLOAD_MASS__KG_" > 4000 AND "PAYLOAD_MASS__KG_" < 6000);
```

Booster_Version
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1032.1
F9 B4 B1040.1
F9 FT B1031.2
F9 B4 B1043.1
F9 B5 B1046.2
F9 B5 B1047.2
F9 B5 B1048.3
F9 B5 B1051.2
F9 B5B1060.1
F9 B5 B1058.2
F9 B5B1062.1

Logic:

WHERE is used to create a mask for values within Landing Outcome that has Success as its value and Payload Mass is between the range of 4000 to 6000.

Total Number of Successful and Failure Mission Outcomes

Query

Output

```
%%sql
SELECT "Landing_Outcome" AS "Landing Outcome", COUNT("Landing_Outcome") AS "Frequency"
FROM SPACEXTBL
GROUP BY 1;
```

Landing Outcome	Frequency
Controlled (ocean)	5
Failure	3
Failure (drone ship)	5
Failure (parachute)	2
No attempt	21
No attempt	1
Precluded (drone ship)	1
Success	38
Success (drone ship)	14
Success (ground pad)	9
Uncontrolled (ocean)	2

Logic:

COUNT is used to get the frequency if each unique value in Landing Outcome.

Boosters Carried Maximum Payload

Query → Output

```
%%sql
SELECT "Booster_Version"
FROM SPACEXTBL
WHERE "PAYLOAD_MASS__KG_" == (SELECT MAX("PAYLOAD_MASS__KG_") FROM SPACEXTBL);
```

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

Logic:

WHERE is used to create a mask with a boolean in which the Payload Mass of the Booster Version matches the **MAX** heaviest Payload Mass.

2015 Launch Records

Query → Output

```
%%sql
SELECT substr("Date",6,2) AS "Month", substr("Date",1,4) AS "Year", "Landing_Outcome", "Booster_Version", "Launch_Site"
FROM SPACEXTBL
WHERE ("Landing_Outcome" NOT LIKE "%Success%") AND ("Year"= "2015");
```

Month	Year	Landing_Outcome	Booster_Version	Launch_Site
01	2015	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
02	2015	Controlled (ocean)	F9 v1.1 B1013	CCAFS LC-40
03	2015	No attempt	F9 v1.1 B1014	CCAFS LC-40
04	2015	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40
04	2015	No attempt	F9 v1.1 B1016	CCAFS LC-40
06	2015	Precluded (drone ship)	F9 v1.1 B1018	CCAFS LC-40

Logic:

Substr is used to get the string in the position of the second parameter with a range of the third parameter. Therefore, (6,2) is to get the month and (1,4) is to get the year.
WHERE is used to get every but not **NOT LIKE** success, so failure and that is in the year 2015.

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

Query → Output

```
%%sql
SELECT "Landing_Outcome", COUNT("Landing_Outcome") AS "Frequency"
FROM SPACEXTBL
WHERE "Date" BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY 1
ORDER BY "Frequency" DESC;
```

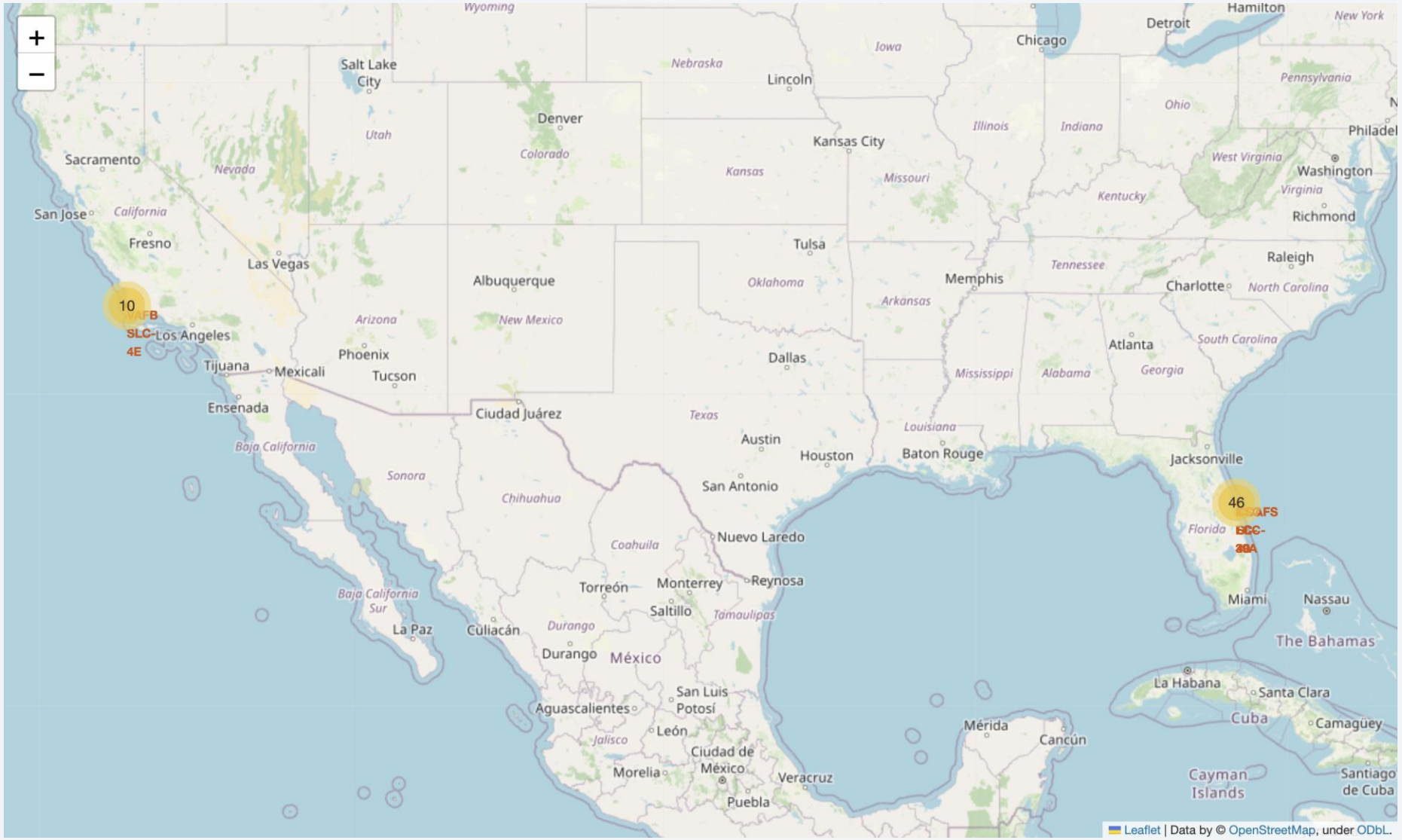
Landing_Outcome	Frequency
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Logic:

COUNT is used to get the frequency of Landin Outcome.
WHERE is used to create a mask of values of Date between 2010-06-04 and 2017-03-20.
GROUP BY to group the results by Landing Outcome.
ORDER BY DESC to get them in descending order from the most frequent outcome.

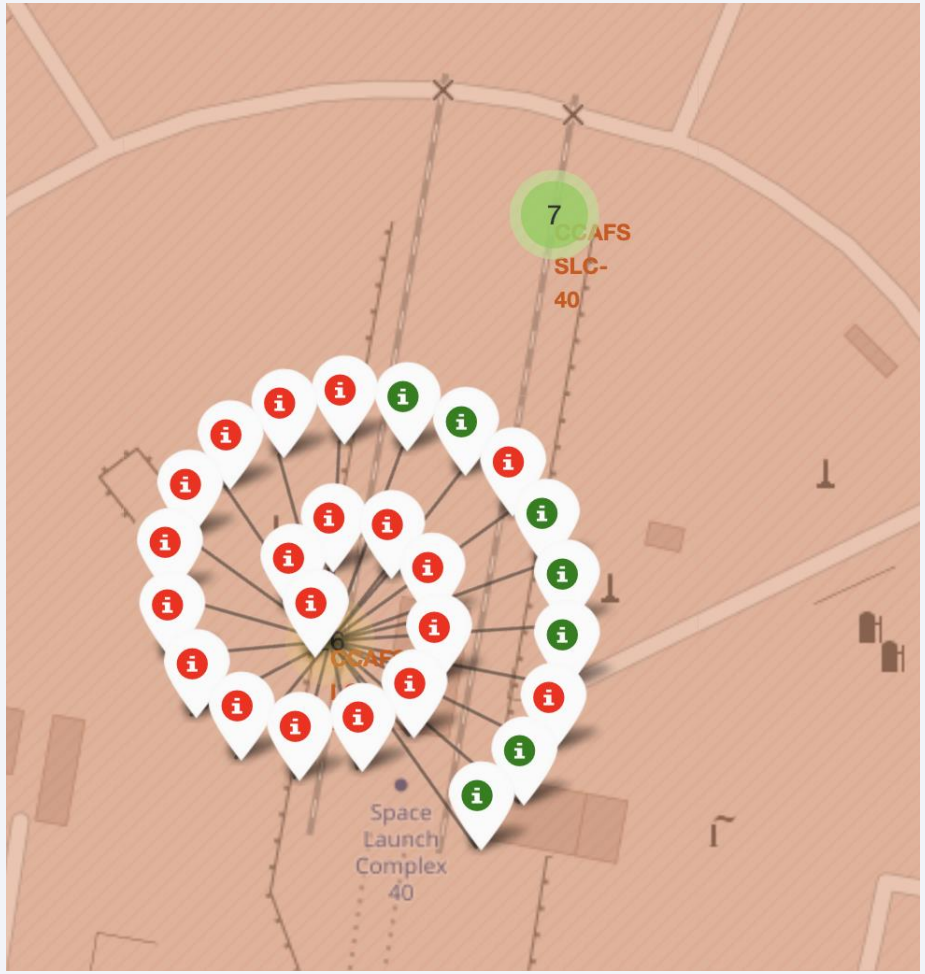

Section 3
Launch Sites
Proximities Analysis

Map of Launch Site Locations



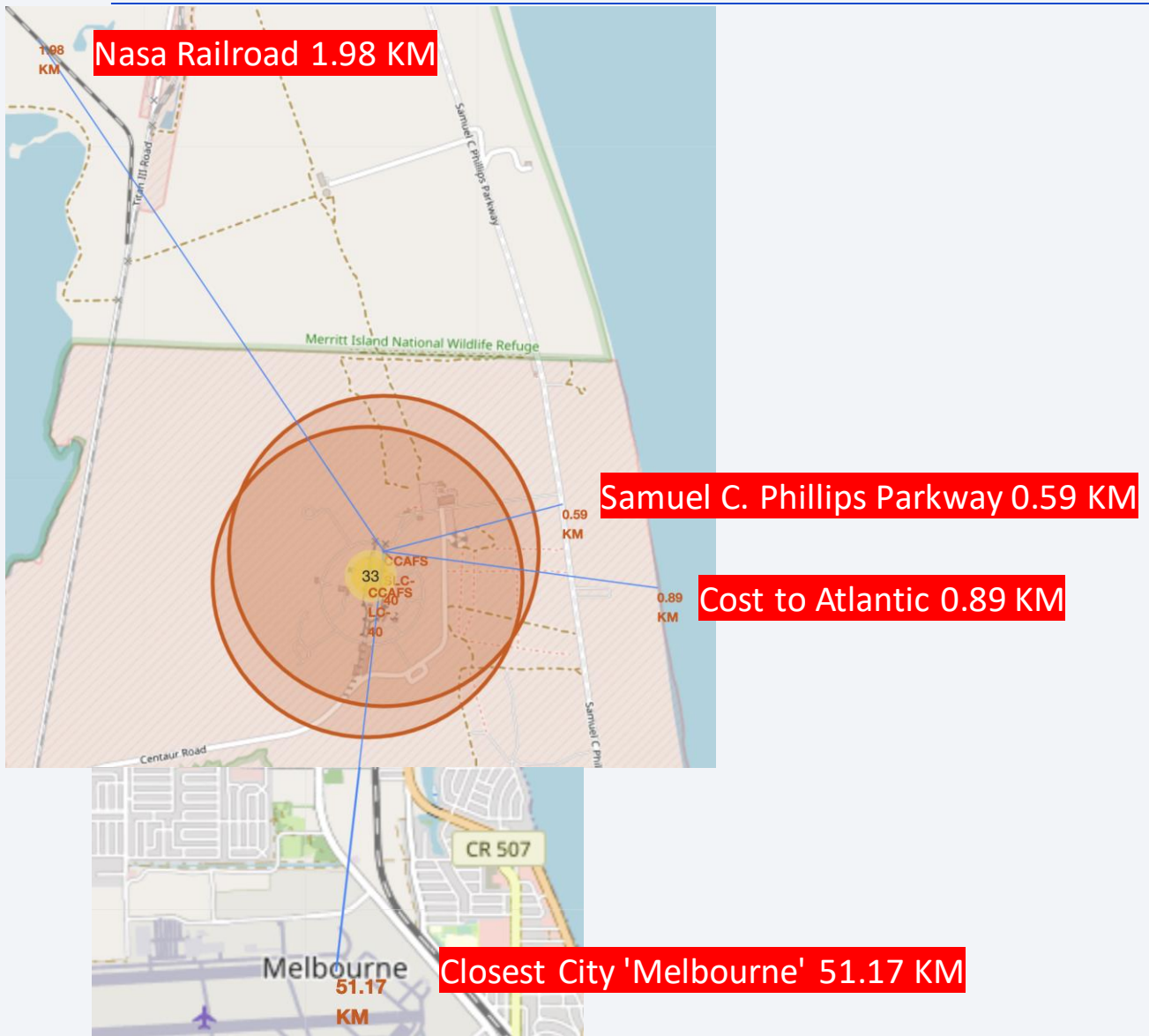
- It can be seen that the Launch Sites are located to both the East and West coasts.
- It is also important to notice that it is not just any coast but a coast heading to oceans.

Map of Launch Site Success Rate

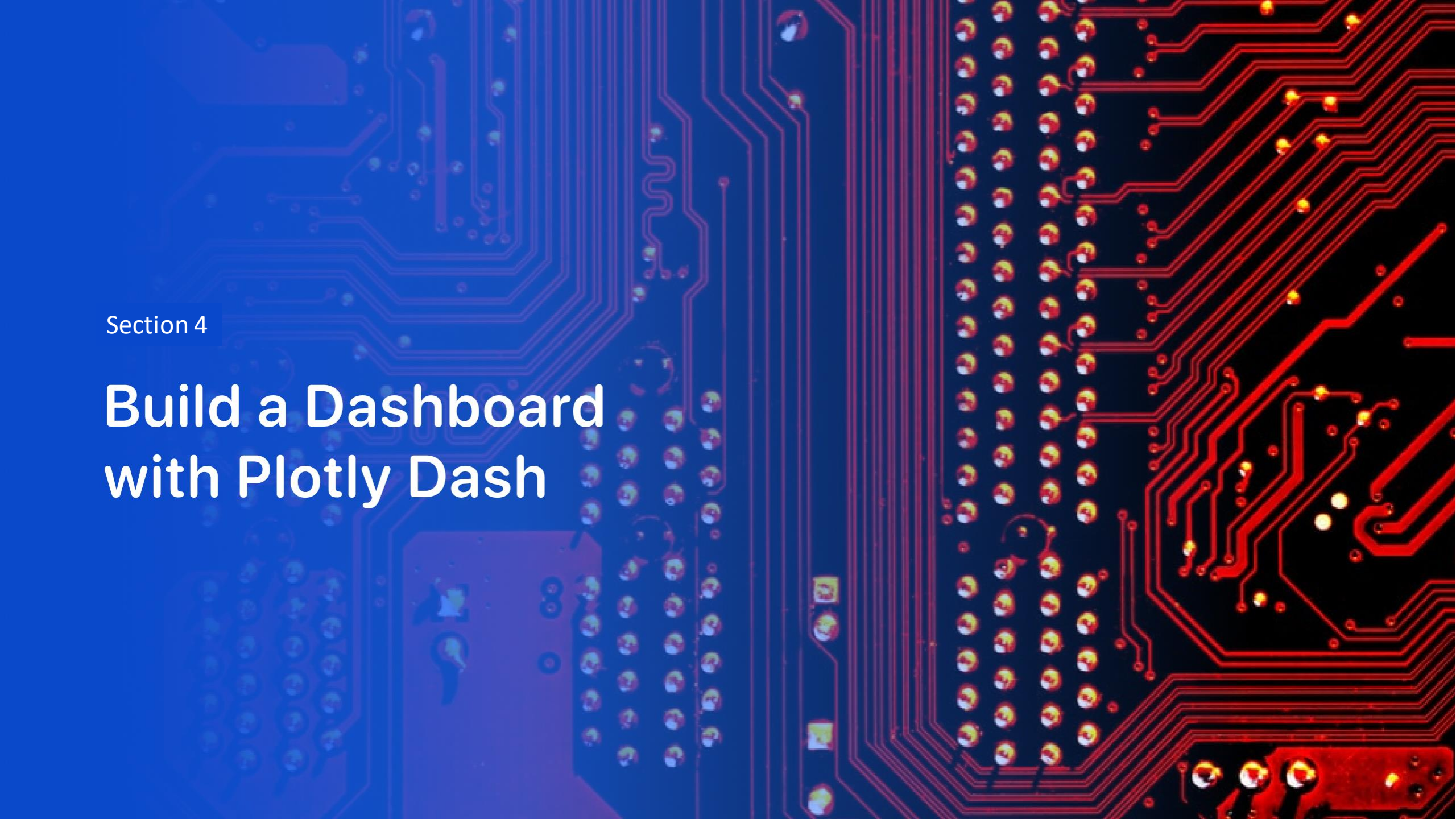


- When clicking to one of the clusters in the previous map, smaller clusters appear.
- When clicking to the smaller clusters (which are Launch Sites), these points are shown.
- The Red points mean 'Failure' while Green points mean 'Success'.

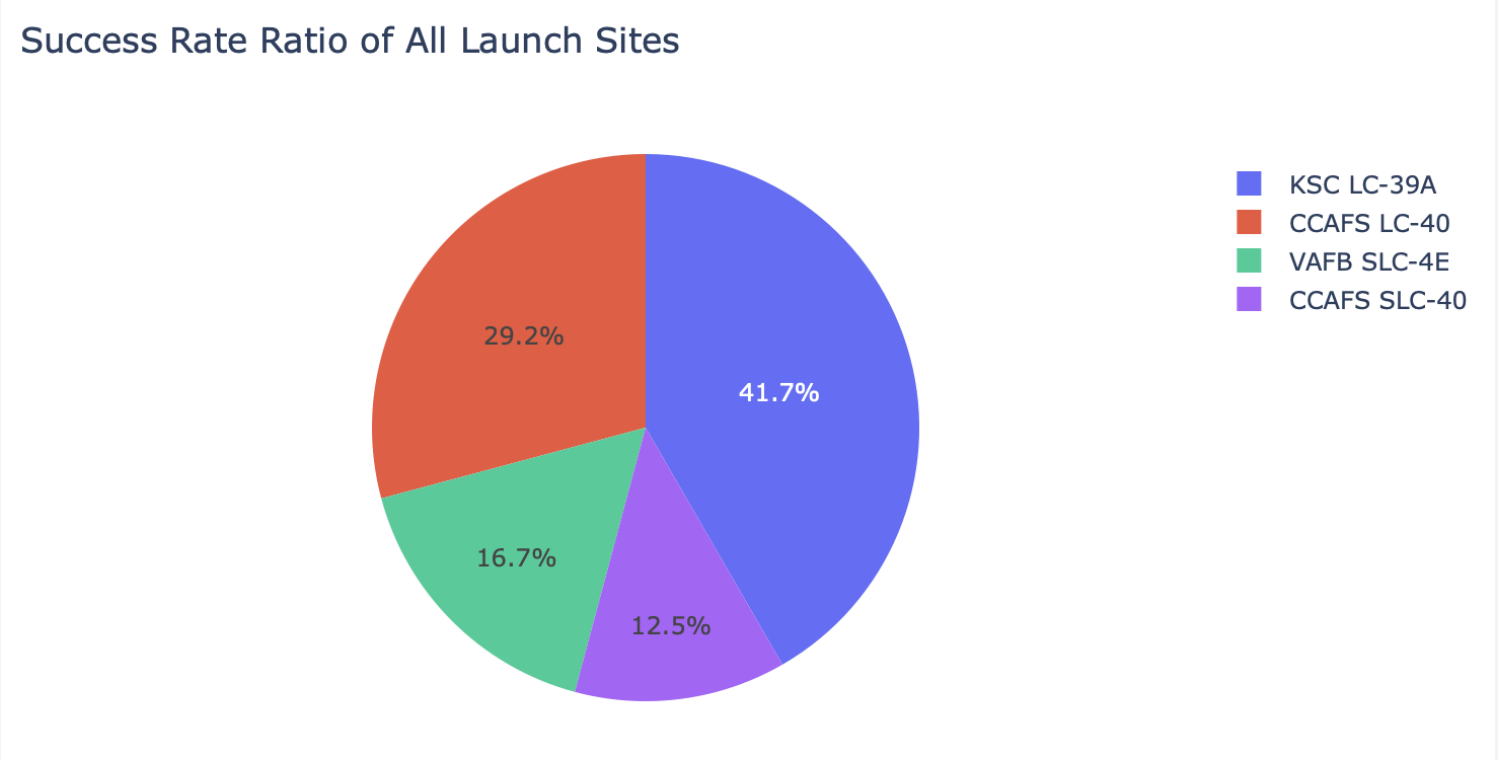
Map of Launch Site Proximities



- It can be seen that launch sites are close to oceans to lessen collateral damage.
- It has close highways and railways maybe for evacuation purposes.
- They have a certain distance to close cities for the same purpose of lessening collateral damage to a denser population

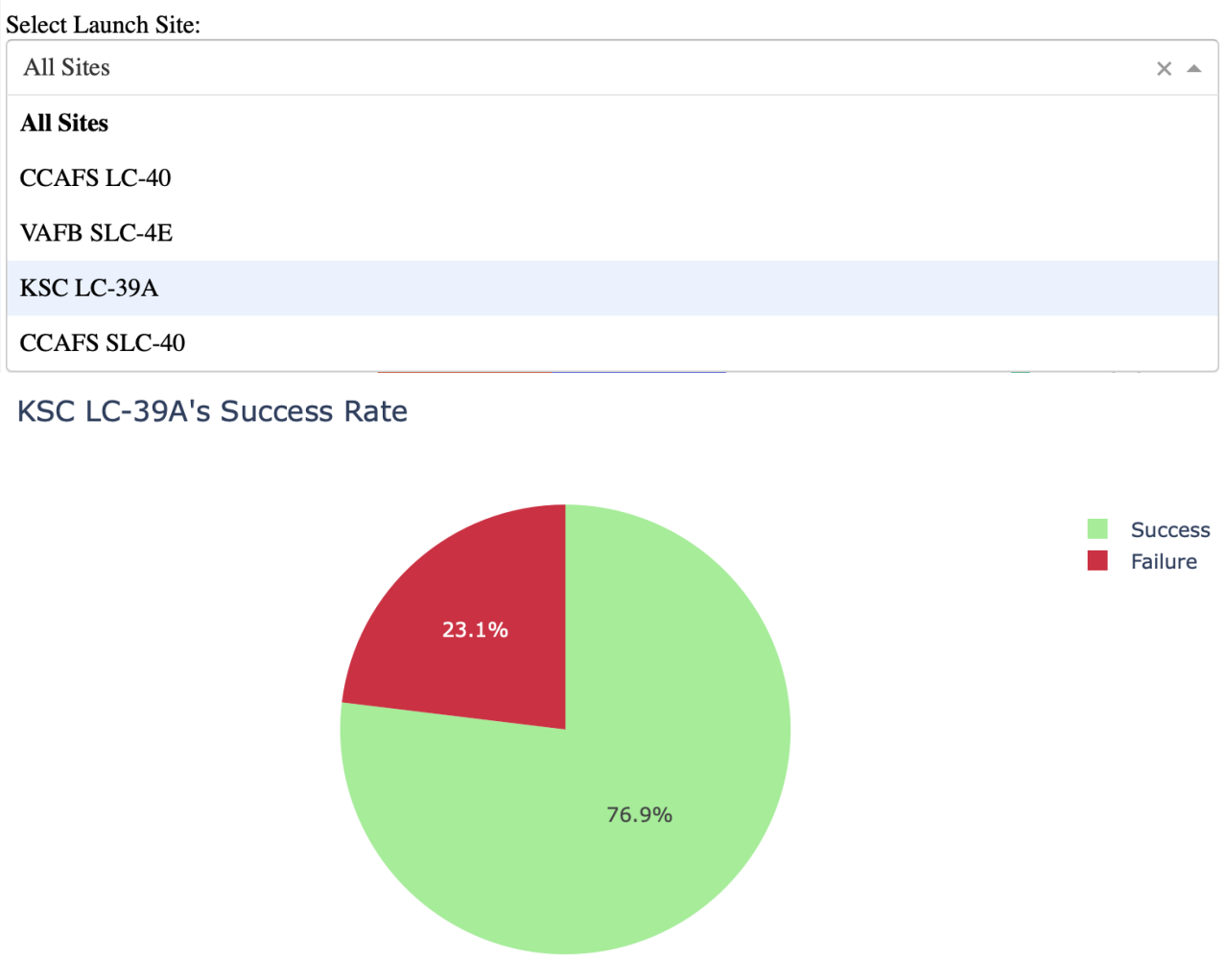

Section 4
Build a Dashboard with Plotly Dash

Success Rate Ratio of All Sites



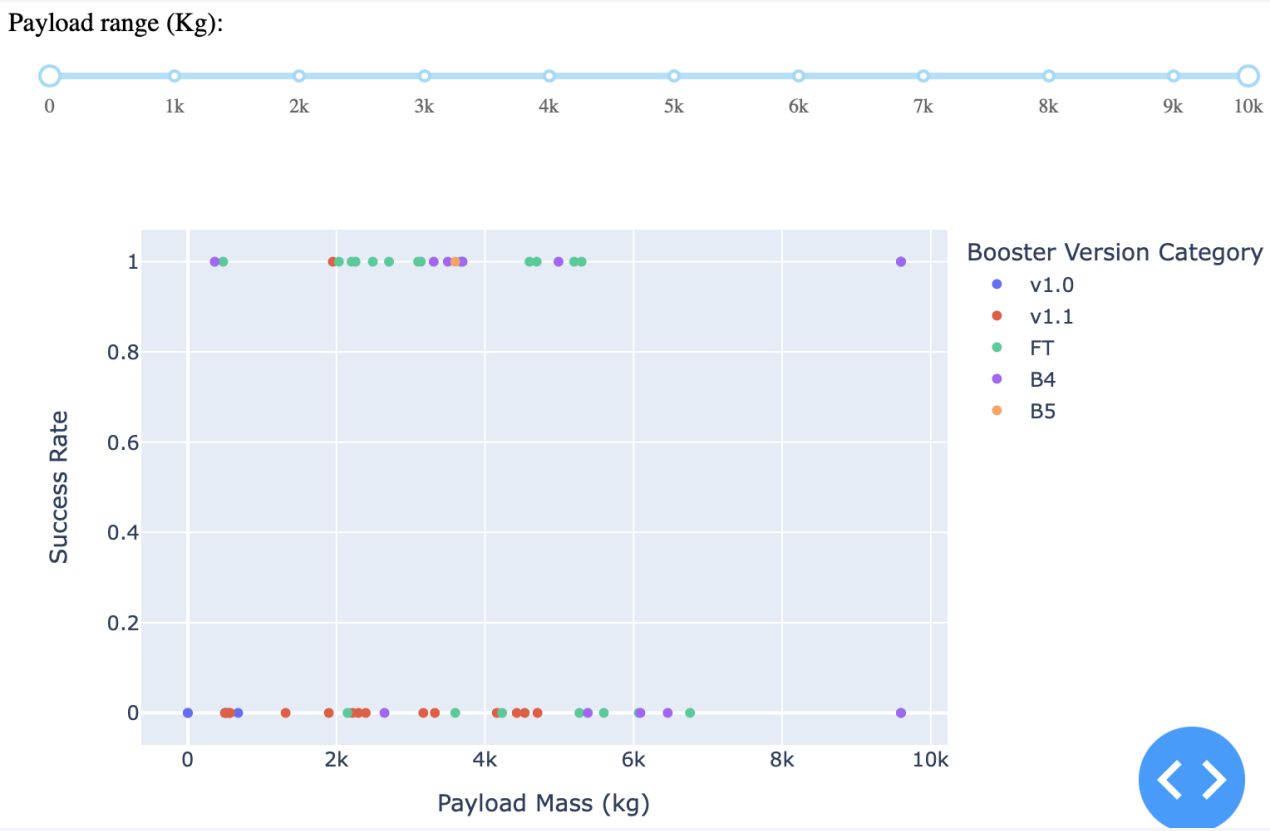
- This pie chart shows the ratio of success rate from all Launch Sites.
- The highest success rate is held by "KSC LC-39A" Launch Site.
- "KSC LC-39A" also holds a little more than 2/5 of all successes.

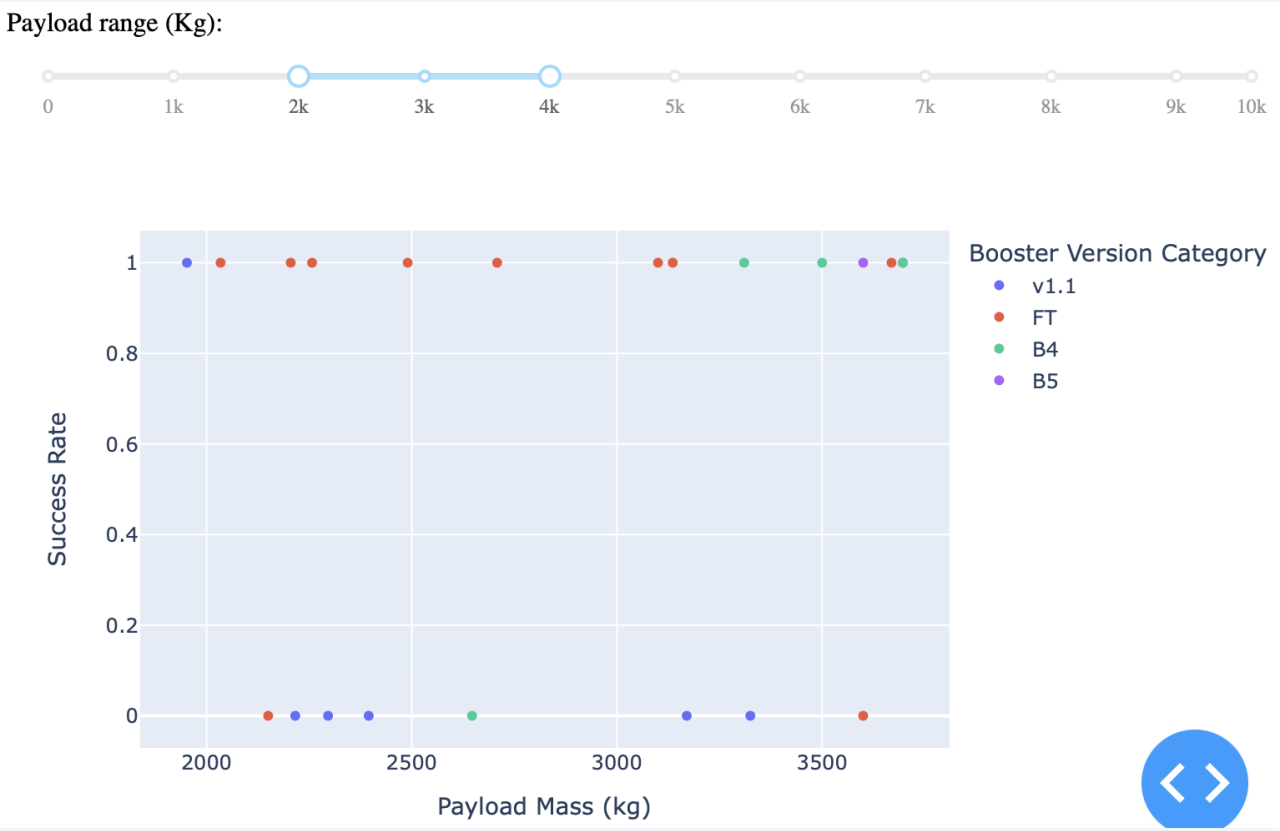
Highest Launch Success Ratio (KSC LC-39A)



- There is an interactive dropdown system to choose the Launch Site of interest.
- This pie chart shows the success rate of "KSC LC-39A", which has the highest success rate ratio from all the launch sites.
- It can be seen that the success rate of this launch site is nearly ¾.

Payload vs. Launch Outcome (All Sites)





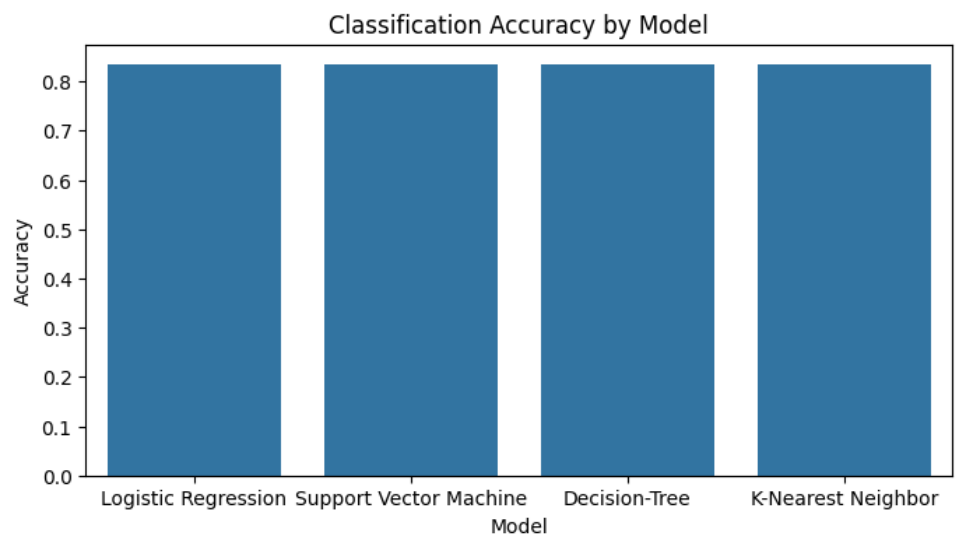
- This is a scatter plot with 'Payload Mass' on the x-axis and 'Success Rate'.
- The different colors are different Booster Versions.
- It can be seen that success rate is highest in between a range of 2000~4000

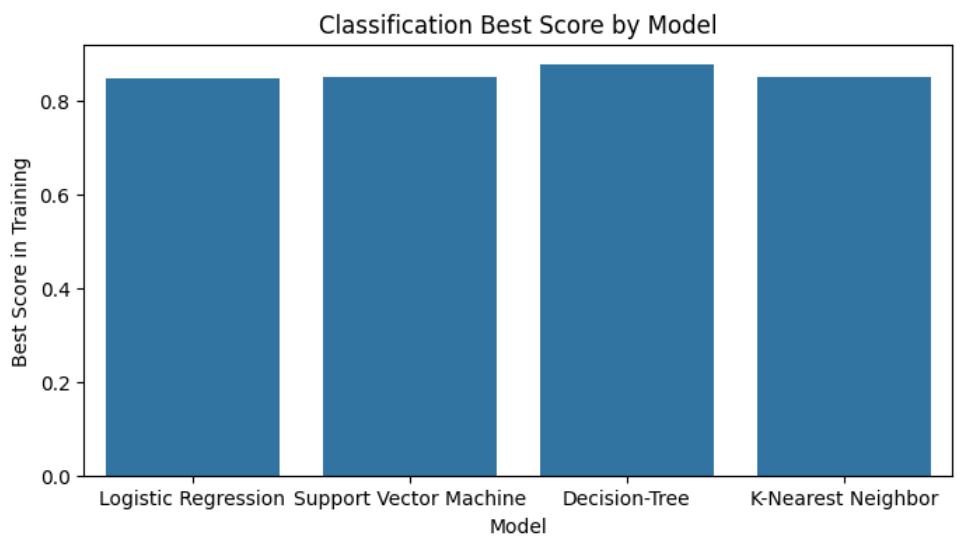
- There is an interactive slider at the top of the scatter plot.
- This plot shows the zoomed and selected 2000~4000 range where success rate seems to be highest.
- It can be seen more clearly.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

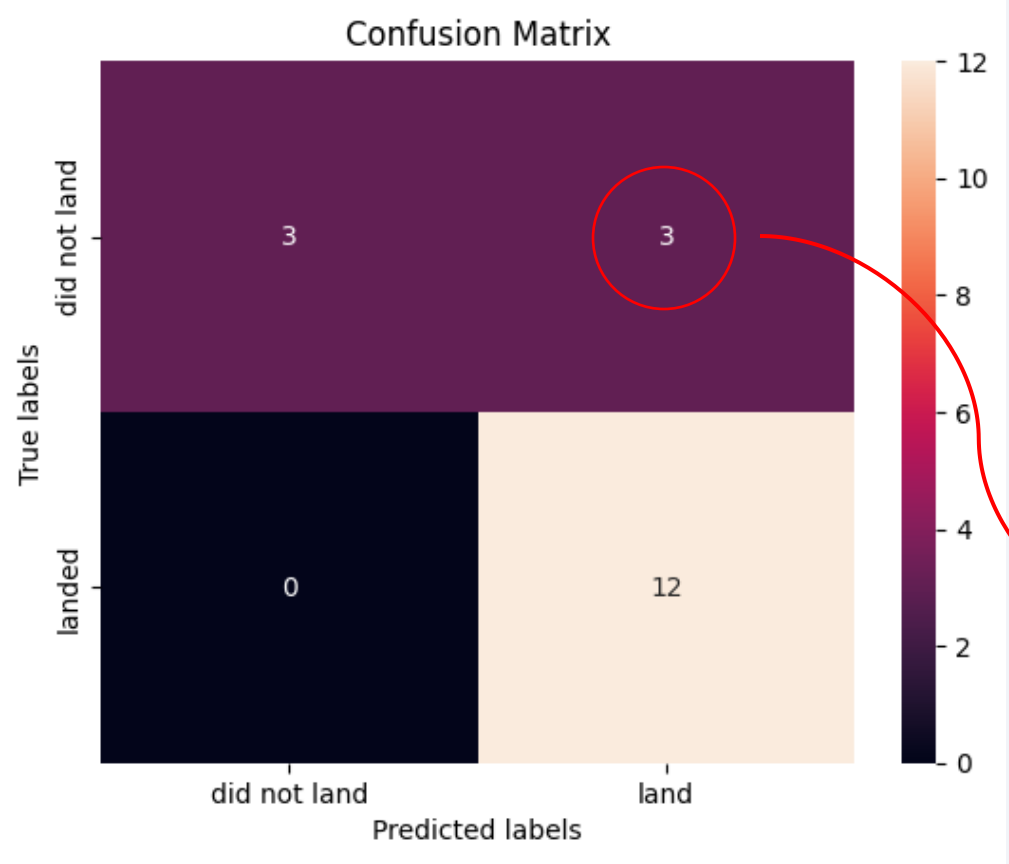




	Model	Accuracy	Train
0	Logistic Regression	0.833333	0.846429
1	Support Vector Machine	0.833333	0.848214
2	Decision-Tree	0.833333	0.875000
3	K-Nearest Neighbor	0.833333	0.848214

- The table above shows the exact values of the bar graphs.
- The graph on top is for the column 'Accuracy' and the graph below is for the column 'Train'.
- The 'Accuracy' is the accuracy of the model on a training set, while 'Train' is the best score from the GridSearchCV's best parameters.

Confusion Matrix



- All four models: Logistic Regression, SVM, Decision Tree, and KNN had the same resulting Confusion Matrices.
- The confusion matrix shows that:
  1. All models predicted 'land' (success) well with a 100% accuracy.
  2. However, there was a deficit in predicting 'did not land' (failure). There are 3 values out of 6 that were incorrectly predicted. The values from the top right corner show a false positive.

Conclusions

- There were various variables to consider to predict the Outcome (success, failure).
- Through visualization, some patterns were recognized.
    a. As Flight Number increased, the success rate also appeared to increase (might be a learning rate through trial and error).
    b. There is a range of Payload Mass that seems optimal.
    c. It seems that targeting certain orbits increase the success rate of landing.
- After one hot encoding and running machine learning models:
    a. It seems that although there are some slight differences when training the data, the tests accuracies had no difference.
    b. The Confusion Matrices were all identical to each other.
- In conclusion, although the machine learning algorithms were fairly good at predicting with 83.33%, it seems that there could be other important domain knowledge that would help the prediction better. (mostly good at predicting 'land' but not 'did not land')

Appendix

- Code Snippet for the Bar Graphs in the 'Classification Accuracy' slide.

```
# Create the Dataset
acc_data = pd.DataFrame()
acc_data['Model'] = ['Logistic Regression', 'Support Vector Machine', 'Decision-Tree', 'K-Nearest Neighbor']
acc_data['Accuracy'] = [logreg_acc, svm_acc, tree_acc, knn_acc]
acc_data['Train'] = [logreg_bs, svm_bs, tree_bs, knn_bs]
acc_data
```

```
# Make the Bar Graph of Model Training
fig = plt.figure(figsize=(8,4))
sns.barplot(acc_data, x='Model', y='Train')
plt.title('Classification Best Score by Model')
plt.ylabel('Best Score in Training')
plt.show()
```

```
# Make the Bar Graph of Model Accuracy
fig = plt.figure(figsize=(8,4))
sns.barplot(acc_data, x='Model', y='Accuracy')
plt.title('Classification Accuracy by Model')
plt.show()
```

Thank you!