IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Dinesh Kumar Majhi
08 March 2023

Outline

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

Executive Summary

- Summary of methodologies
  - ✓ Data Collection using SpaceX API, Web Scraping and Data Wrangling
  - ✓ Exploratory Data Analysis (EDA) using SQL, Pandas and Matplotlib
  - ✓ Data Visualization using Folium and Plotly
  - ✓ Predictive Analysis using ML Classifiers
- Summary of all results
  - ✓ Results from EDA based on respective attributes and conditions
  - ✓ Results for Launch sites, success/failed launches, distances using Folium
  - ✓ Results from Interactive Dashboard on Web Application
  - ✓ Accuracy results for various algorithms for Predictive Analysis

Introduction

- Project background and context

  The target focus of this project is to understand the bid against SpaceX for a rocket launch. To determine how shall an alternate company function like SpaceX, there is a lot of work down the line to get the understanding of how the plan shall be put into execution. SpaceX launches Falcon 9 with lesser investment as compared to other providers, due to its reuse of the first stage.

  Hence, if we can determine if the first stage will land, we can estimate the cost of a launch and this information can help to bid against SpaceX

- Problems you want to find answers

  - What factors lead to a successful landing for a first stage?
  - How can we determine by correlating the attributes of a rocket for a successful outcome?
  - How can we determine the conditions to have the success rate for landing?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Using SpaceX REST API
  - Web Scraping on Wikipedia
- Perform data wrangling
  - Data is processed, then calculations are done based on the attributes like finding number of launches sites, occurrences orbits, landing outcomes, etc.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

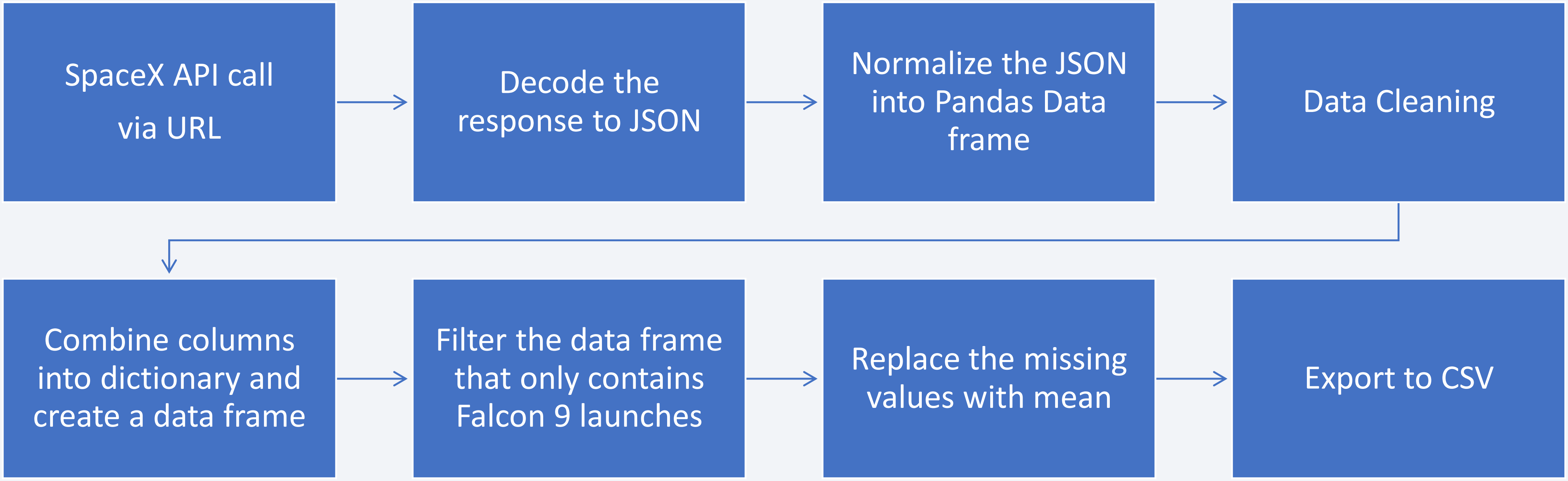
Data Collection

- Data sets are collected via SpaceX REST API and Web Scraping on Wikipedia.
  - Using SpaceX REST API
  - Source: https://api.spacexdata.com/v4

- Data sets are collected via Web Scraping on Wikipedia.
  - Using Web Scraping on Wikipedia
  - Source: https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches

Data Collection – SpaceX API

Code: https://github.com/D1N3SH-DEV/Project-Capstone-SpaceY/blob/master/Data%20Collection.ipynb



SpaceX API call via URL

Decode the response to JSON

Normalize the JSON into Pandas Data frame

Data Cleaning

Combine columns into dictionary and create a data frame

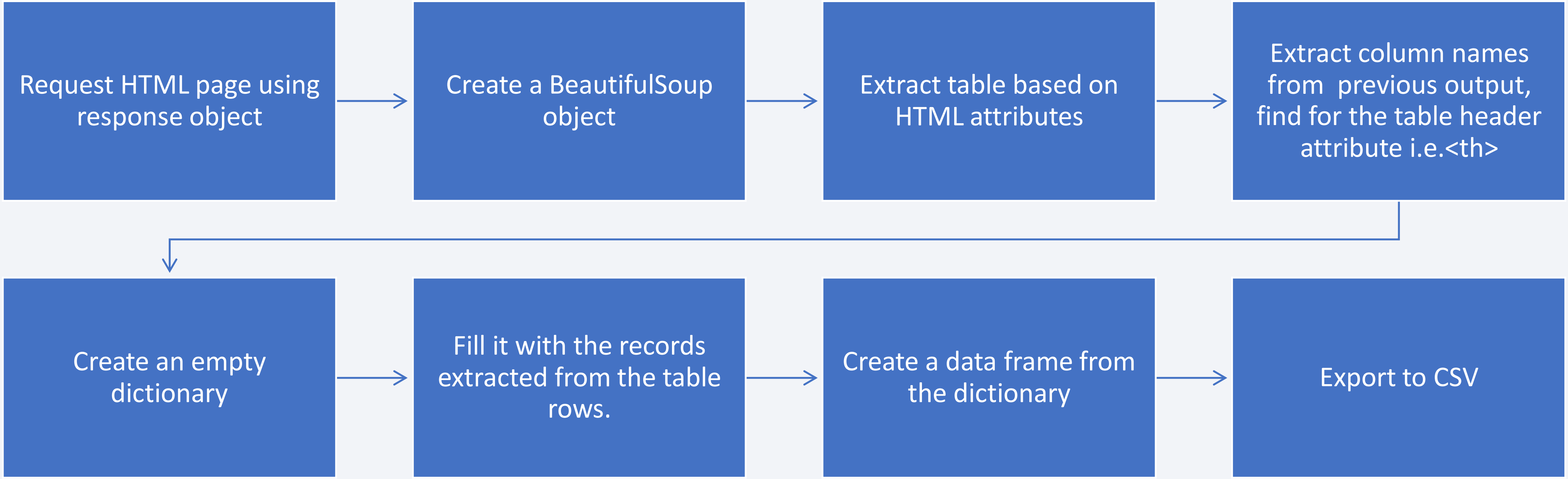
Filter the data frame that only contains Falcon 9 launches

Replace the missing values with mean

Export to CSV

Data Collection - Scraping

Code: https://github.com/D1N3SH-DEV/Project-Capstone-SpaceY/blob/master/Data%20Collection%20with%20Web%20Scrapping.ipynb



Request HTML page using response object

Create a BeautifulSoup object

Extract table based on HTML attributes

Extract column names from previous output, find for the table header attribute i.e.<th>

Create an empty dictionary

Fill it with the records extracted from the table rows.

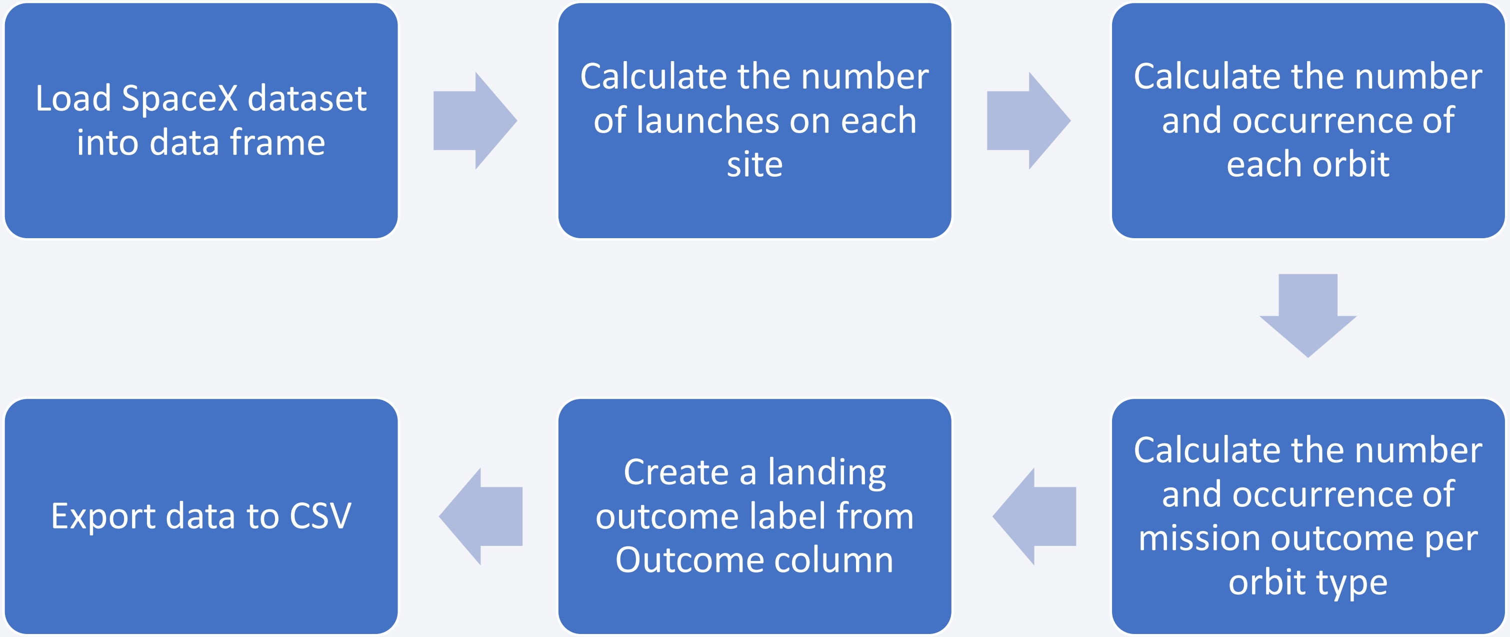
Create a data frame from the dictionary

Export to CSV

Data Wrangling

Code: https://github.com/D1N3SH-DEV/Project-Capstone-SpaceY/blob/master/Data%20Wrangling.ipynb

- There are several different cases where the booster did not land successfully, sometimes a landing was attempted but failed due to an accident.
- The labeling is done by transforming the object variables into categorical variable as 0 or 1.
- Given that, there are 6 types of outcomes:
    - True Ocean, Ture RTLS and True ASDS being success and labeled as ‘1’
    - False Ocean, False RTLS and False ASDS being failure and labeled as ‘0’



Load SpaceX dataset into data frame

Calculate the number of launches on each site

Calculate the number and occurrence of each orbit

Calculate the number and occurrence of mission outcome per orbit type

Create a landing outcome label from Outcome column

Export data to CSV

EDA with Data Visualization

Code: Link to Code

- Scatter Plots, Bar Chart and Line Chart were plotted for observations.
- Scatter Plot usage is good to determine if there is any correlation between the variables.
- Bar Chart is used to show the length of each bar proportional to the value of the item that it represents.
- Line Chart is used for a continuous data set, and we are interested in visualizing the data over a period.

Scatter Plots	•Flight Number and Payload
	Flight Number and Launch Site
	Payload and Launch Site
	Flight Number and Orbit type
	Payload and Orbit type

Bar Chart:

- Success Rate and Orbit Type

Line Chart:

- Success Rate and Year

EDA with SQL

Code: Github Link

- For the given data set, the following were used for SQL queries:
  - ✓ 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 achieved.
  - ✓ List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
  - ✓ List the total number of successful and failure mission outcomes
  - ✓ List the names of the booster_versions which have carried the maximum payload mass. Use a subquery
  - ✓ List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.
  - ✓ Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

Build an Interactive Map with Folium

Code: https://github.com/D1N3SH-DEV/Project-Capstone-SpaceY/blob/master/Location_Analysis_Folium.ipynb

- Following are the map objects added to the Folium Map:

    1. folium.Circle, folium.map.Marker – to create red circle at NASA coordinate with label showing its name
    2. folium.Circle, folium.map.Marker, folium.features.DivIcon – to create red circles at each launch site coordinate with label showing its name
    3. folium.plugins.MarkerCluster – helps to simplify a map containing many markers having the same coordinate
    4. folium.map.Marker, folium.Icon – Icon helps to differentiate based on colors for landing outcomes
    5. folium.Marker, folium.PolyLine – markers show the distance between the launch sites and the respective locations.

- Objects are helpful to simplify the way of presenting data on a map by using markers i.e., launch sites, marking success/failed launches for each site on the map, distances between a launch site to its proximities,

Build a Dashboard with Plotly Dash

Code: https://github.com/D1N3SH-DEV/Project-Capstone-SpaceY/blob/master/Dash_interaction.py

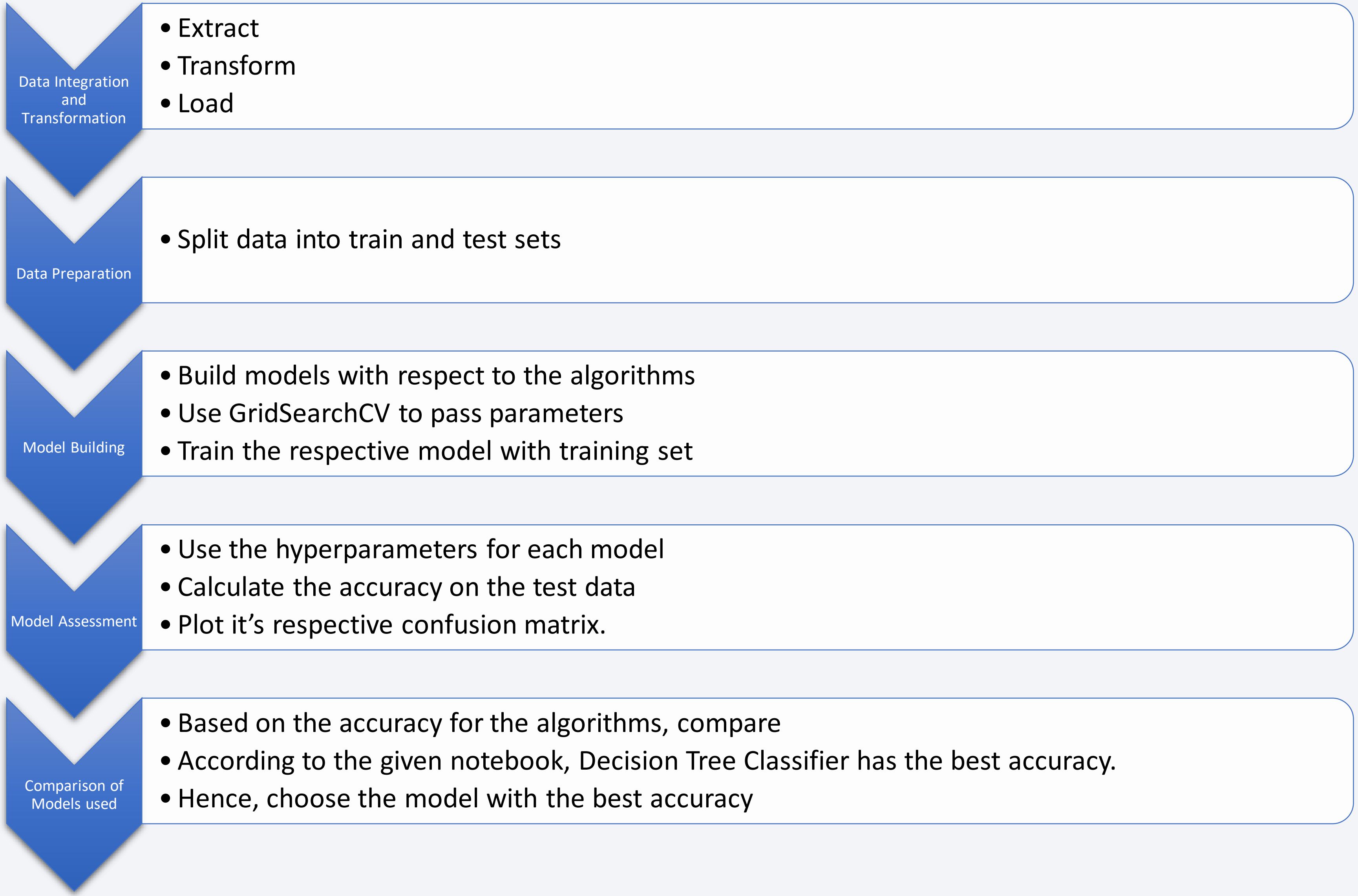
- To build this interactive dashboard following are used:

1. dcc.Dropdown – user defined dropdown option which lets user to choose the launch site/sites
2. plotly.express.pie – based on the user choice from the dropdown component, it will show the percentages of success and failure for launch site location
3. dcc.RangeSlider – this will interactively allow user to select a payload mass within the range.
4. plotly.express.scatter – this will show the relationship between success and payload mass.

Note: Here, dcc is an alias and has been imported from dash_core_component

Predictive Analysis (Classification)

Code: GitHub Link



Data Integration and Transformation

- Extract
- Transform
- Load

Data Preparation

- Split data into train and test sets

Model Building

- Build models with respect to the algorithms
- Use GridSearchCV to pass parameters
- Train the respective model with training set

Model Assessment

- Use the hyperparameters for each model
- Calculate the accuracy on the test data
- Plot it’s respective confusion matrix.

Comparison of Models used

- Based on the accuracy for the algorithms, compare
- According to the given notebook, Decision Tree Classifier has the best accuracy.
- Hence, choose the model with the best accuracy

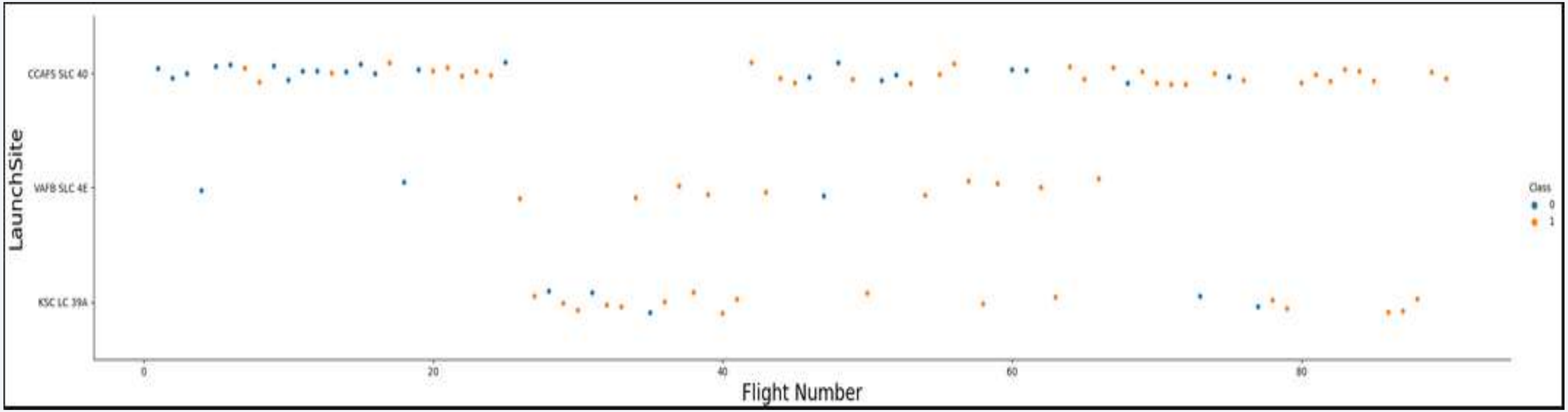
Results

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

Section 2

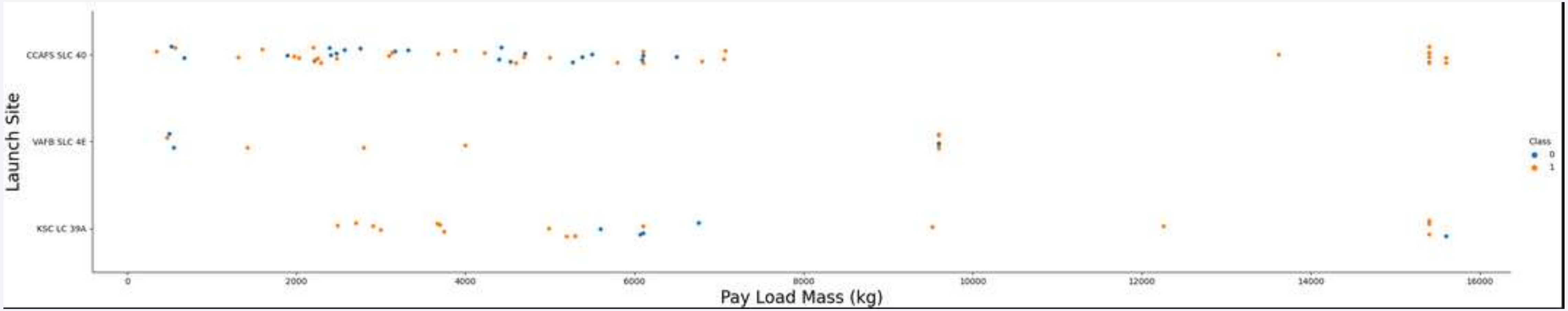
Insights drawn from EDA

Flight Number vs. Launch Site



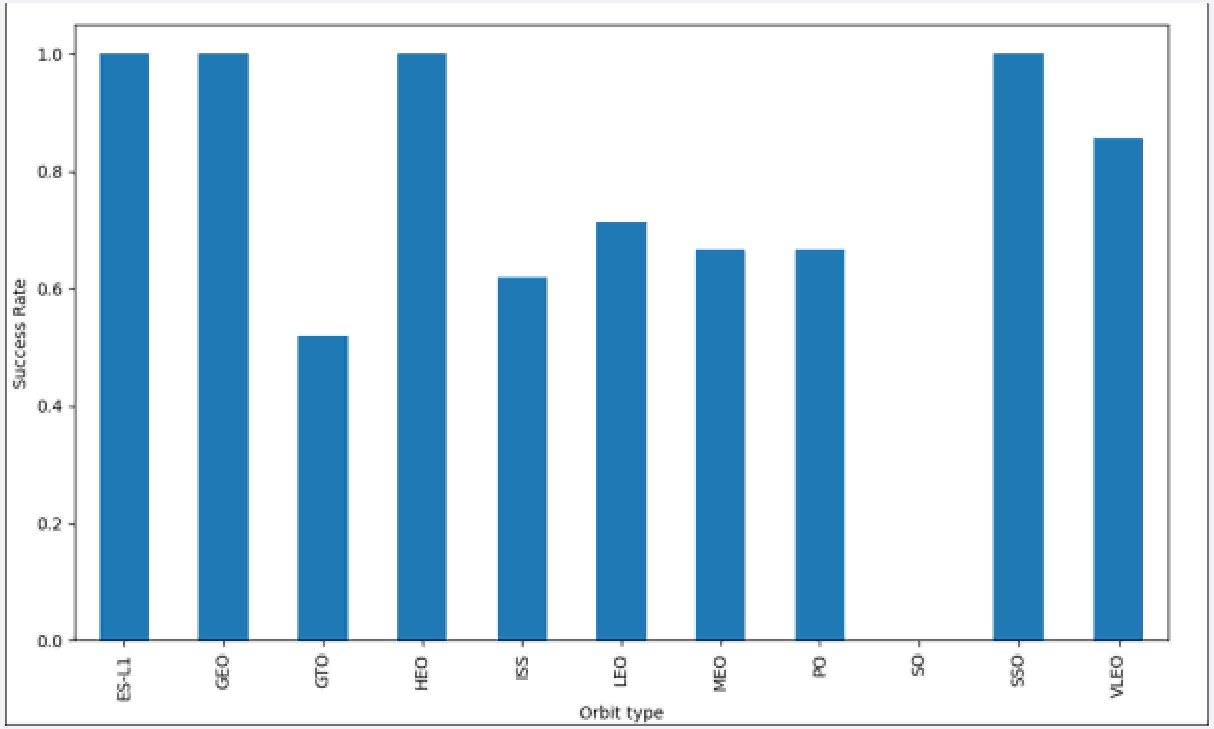
Success rate seems to be increasing for each site.

Payload vs. Launch Site



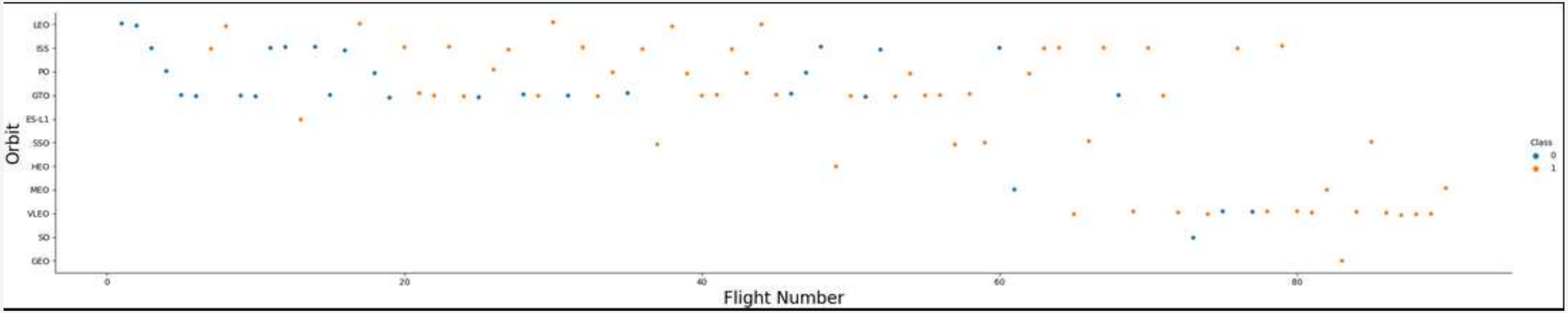
Based on the observation we can see that, heavier payload can be considered for the outcome to be successful but at certain threshold (i.e., >10000) payload being too heavy will definitely outcome failure.

Success Rate vs. Orbit Type



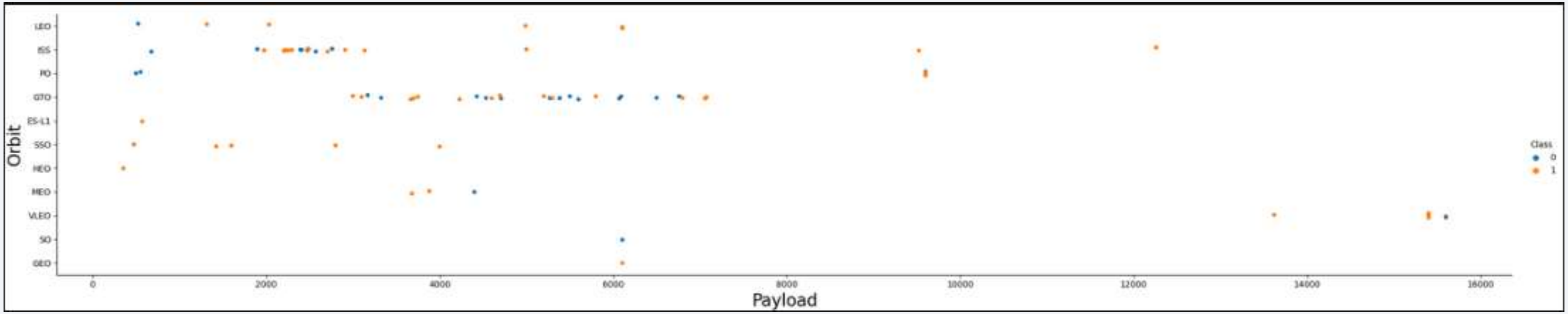
According to the observation we see that **ES-L1**, **GEO**, **HEO** and **SSO** orbit type have the best success rates among other types.

Flight Number vs. Orbit Type



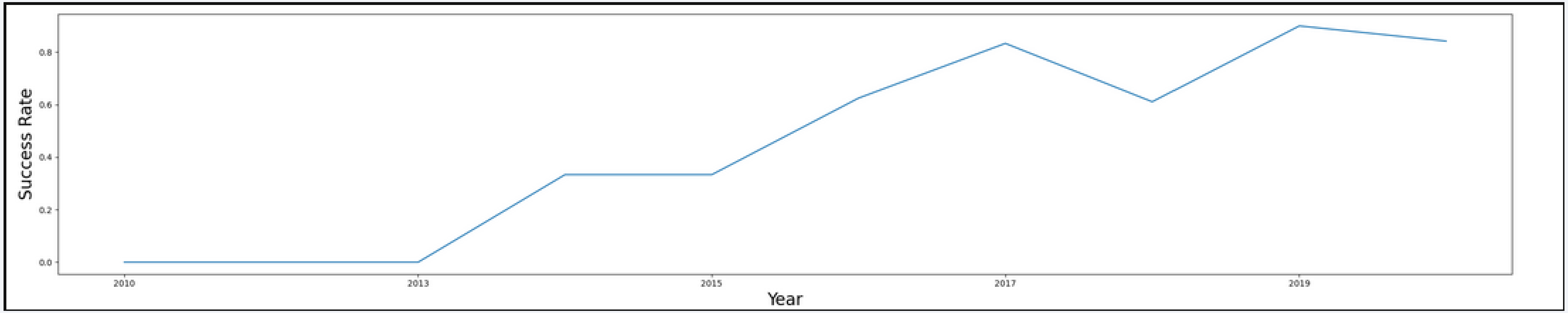
We can see that for the LEO orbit the success appears related to the number of flights and on the other hand there seems to be no relationship between flight number when in GTO orbit.

Payload vs. Orbit Type



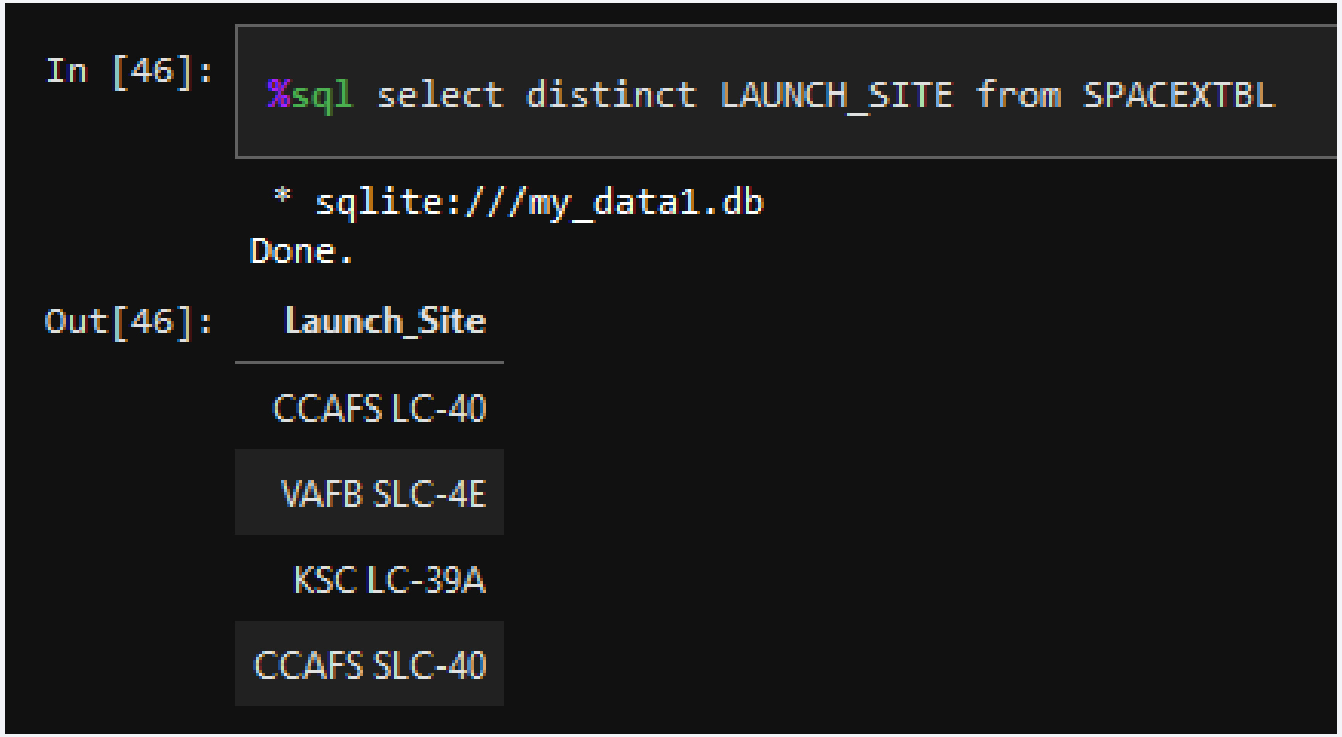
According to the plot we observe that heaviest payload have success for the LEO orbit, heavier payload have success for the GTO orbit and lighter payload have success for ISS orbit.

Launch Success Yearly Trend



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

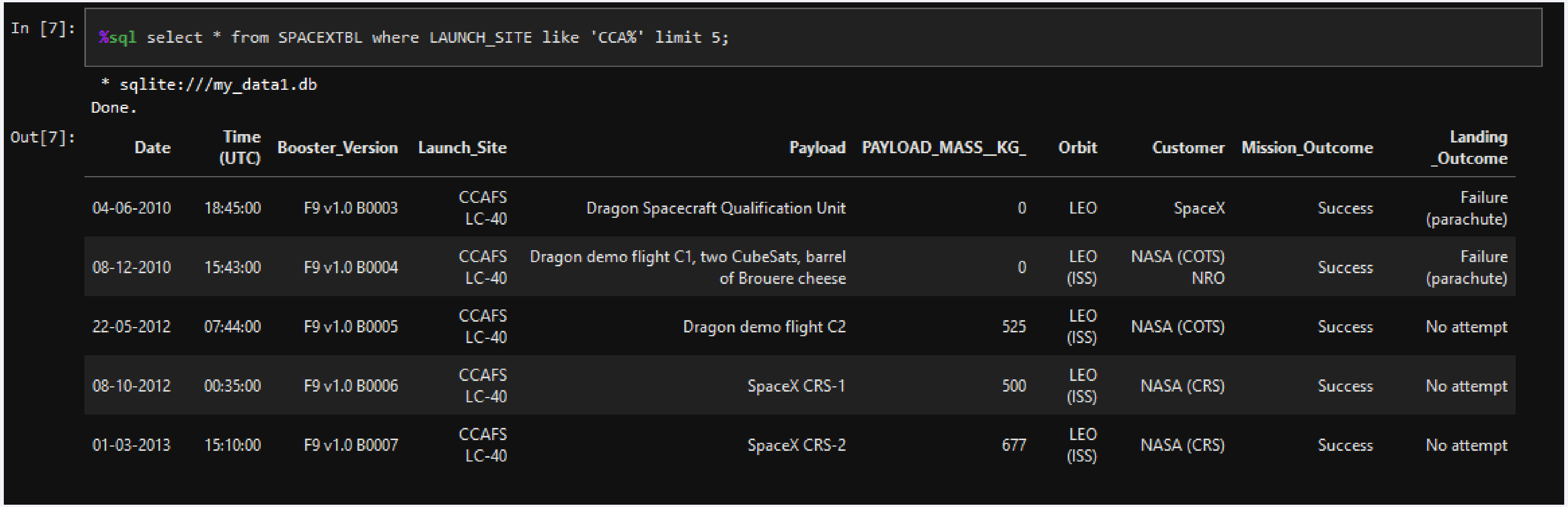
All Launch Site Names



```
In [46]: %sql select distinct LAUNCH_SITE from SPACEXTBL

          * sqlite:///my_data1.db
         Done.

Out[46]:   Launch_Site
          CCAFS LC-40
          VAFB SLC-4E
          KSC LC-39A
         CCAFS SLC-40
```

Using ‘DISTINCT’ in the following query, we can provide the unique launch sites.

Launch Site Names Begin with 'CCA'



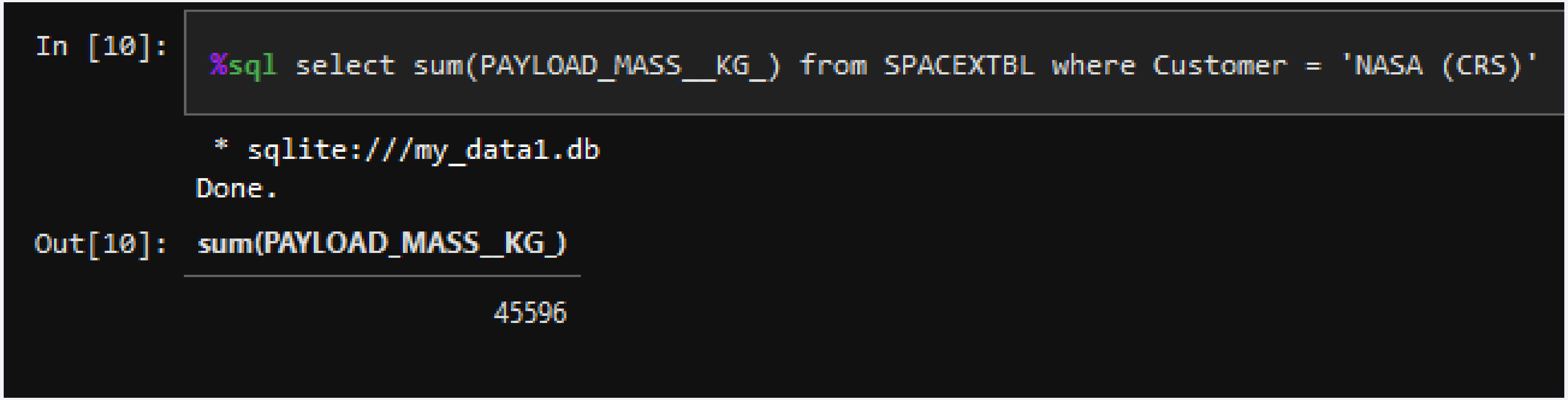
```
In [7]: %sql select * from SPACEXTBL where LAUNCH_SITE like 'CCA%' limit 5;

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

Out[7]:

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

Using a ‘WHERE’ clause followed by ‘LIKE’, filter the launch sites that substrings ‘CCA’.
Using ‘LIMIT’ clause, the result has been contained till 5 records.

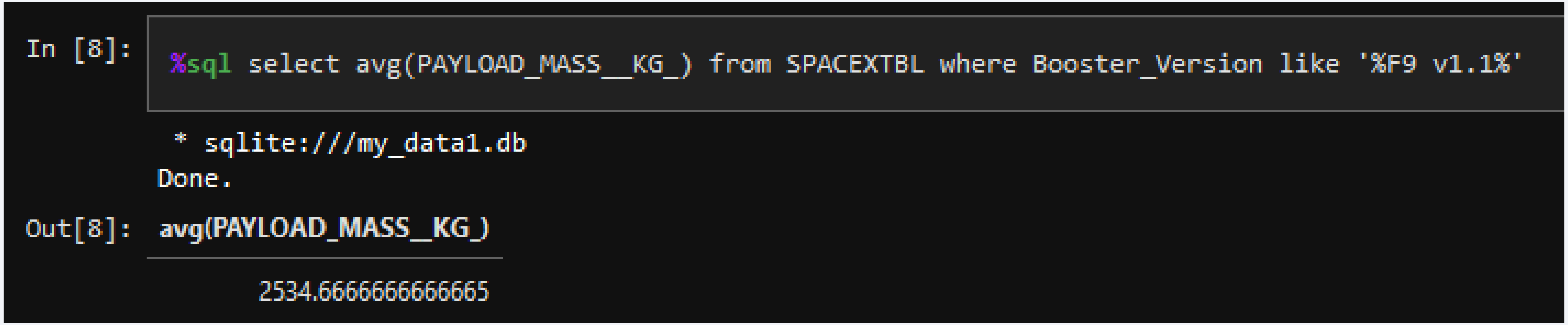
Total Payload Mass



```
In [10]:  %sql select sum(PAYLOAD_MASS__KG_) from SPACEXTBL where Customer = 'NASA (CRS)'

           * sqlite:///my_data1.db
          Done.

Out[10]:  sum(PAYLOAD_MASS__KG_)
                           45596
```

The following query results the total sum of payload mass based on customers prevailing from NASA (CRS)

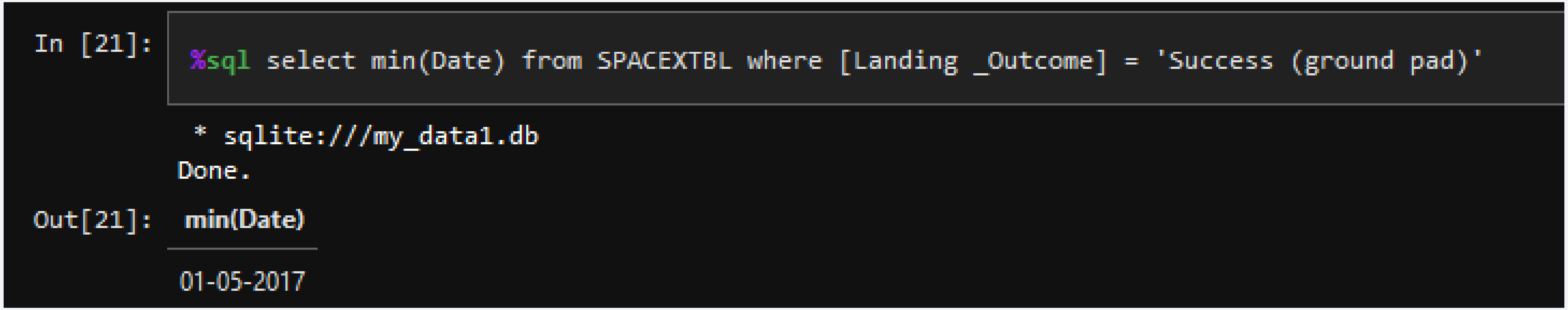
Average Payload Mass by F9 v1.1



```
In [8]:  %sql select avg(PAYLOAD_MASS__KG_) from SPACEXTBL where Booster_Version like '%F9 v1.1%'

          * sqlite:///my_data1.db
         Done.

Out[8]:  avg(PAYLOAD_MASS__KG_)
                  2534.6666666666665
```

The following query results the average payload mass for the Booster Version F9 v1.1.
avg() is used to return the average of the column and filtered using ‘WHERE’ and ‘LIKE’ clause.

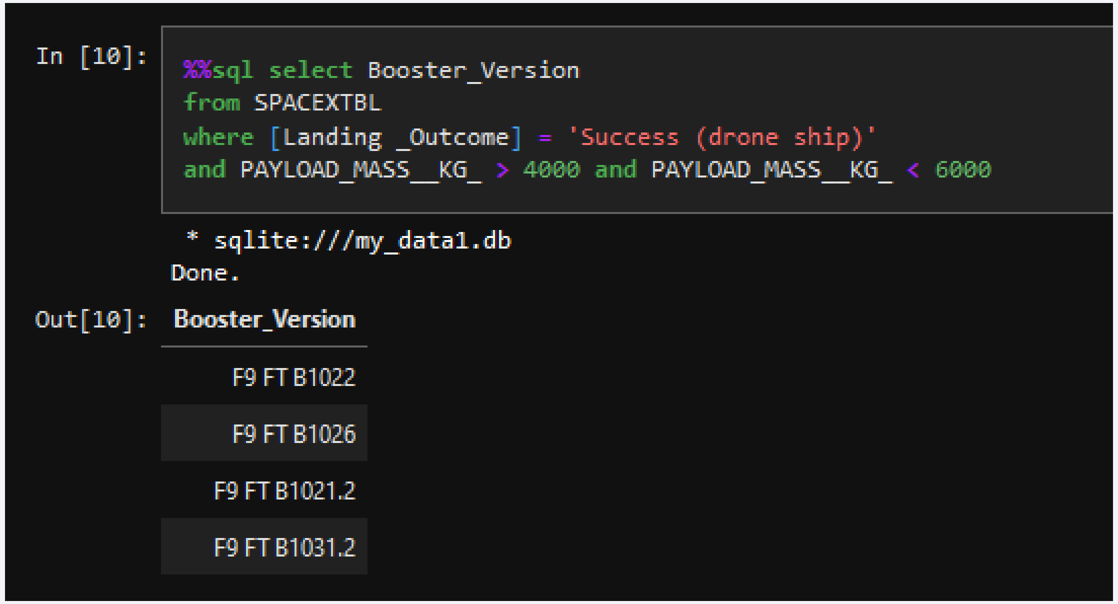
First Successful Ground Landing Date



```
In [21]: %sql select min(Date) from SPACEXTBL where [Landing _Outcome] = 'Success (ground pad)'

          * sqlite:///my_data1.db
         Done.

Out[21]: min(Date)
         01-05-2017
```

This query results the first successful landing date.
min() is used in ‘Date’ column to return the first date including the ‘WHERE’ clause to filter the success landing.

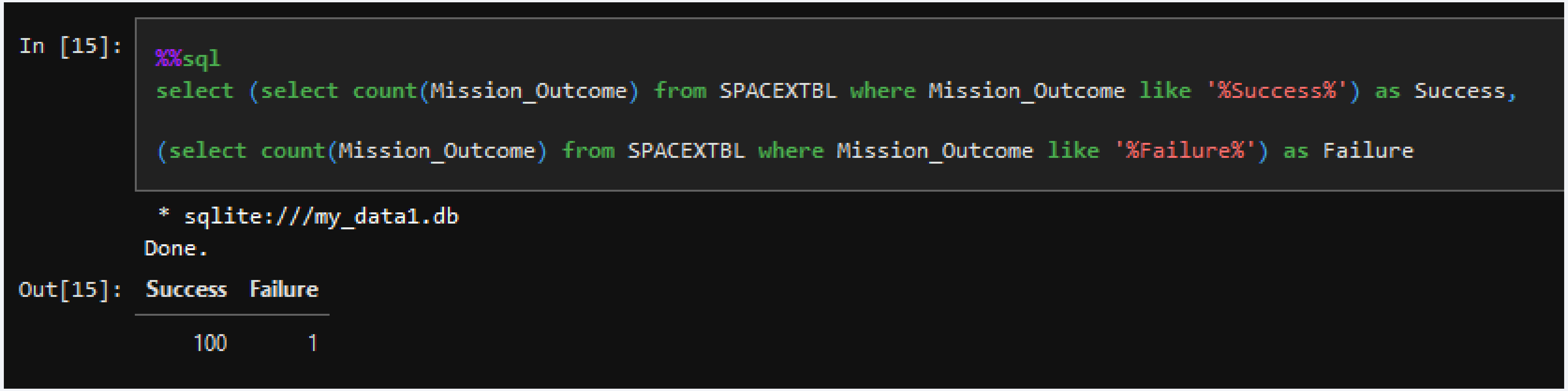
Successful Drone Ship Landing with Payload between 4000 and 6000



```
In [10]:  %%sql select Booster_Version
          from SPACEXTBL
          where [Landing _Outcome] = 'Success (drone ship)'
          and PAYLOAD_MASS__KG_ > 4000 and PAYLOAD_MASS__KG_ < 6000

           * sqlite:///my_data1.db
          Done.

Out[10]:  Booster_Version
                 F9 FT B1022
                 F9 FT B1026
               F9 FT B1021.2
               F9 FT B1031.2
```

This query returns the successful drone ship landing.
Condition is satisfied by using ‘WHERE’ clause followed by the payload.

Total Number of Successful and Failure Mission Outcomes



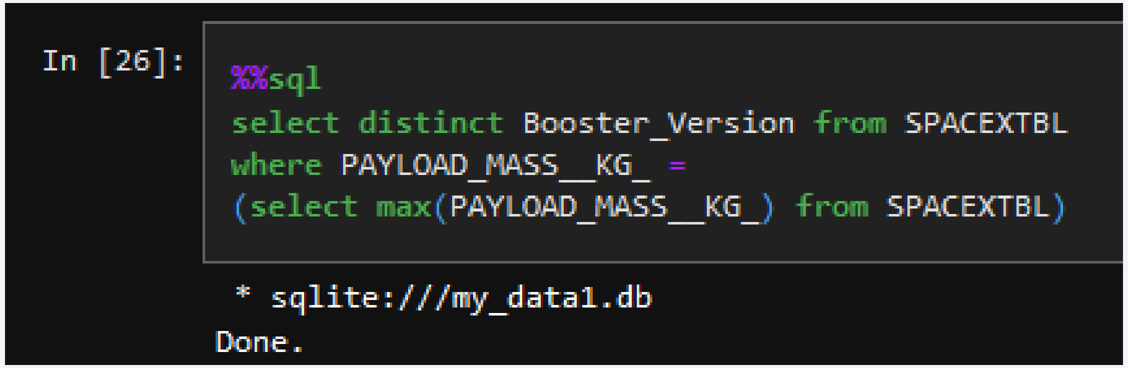
```
In [15]: %%sql
         select (select count(Mission_Outcome) from SPACEXTBL where Mission_Outcome like '%Success%') as Success,

         (select count(Mission_Outcome) from SPACEXTBL where Mission_Outcome like '%Failure%') as Failure

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

Out[15]:

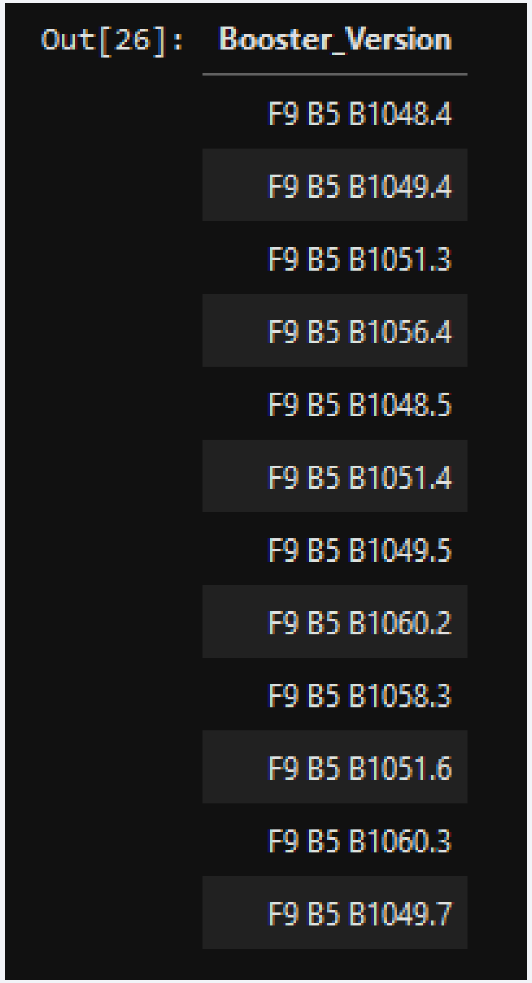
Success	Failure
100	1

This query results the total counts of success and failure mission outcomes.
The reason to use a sub-query is to bring fairness to the query and the output.
Conditions are satisfied by ‘WHERE’ and ‘LIKE’ clause.
The column name has been manually given by ‘AS’ clause.

Boosters Carried Maximum Payload



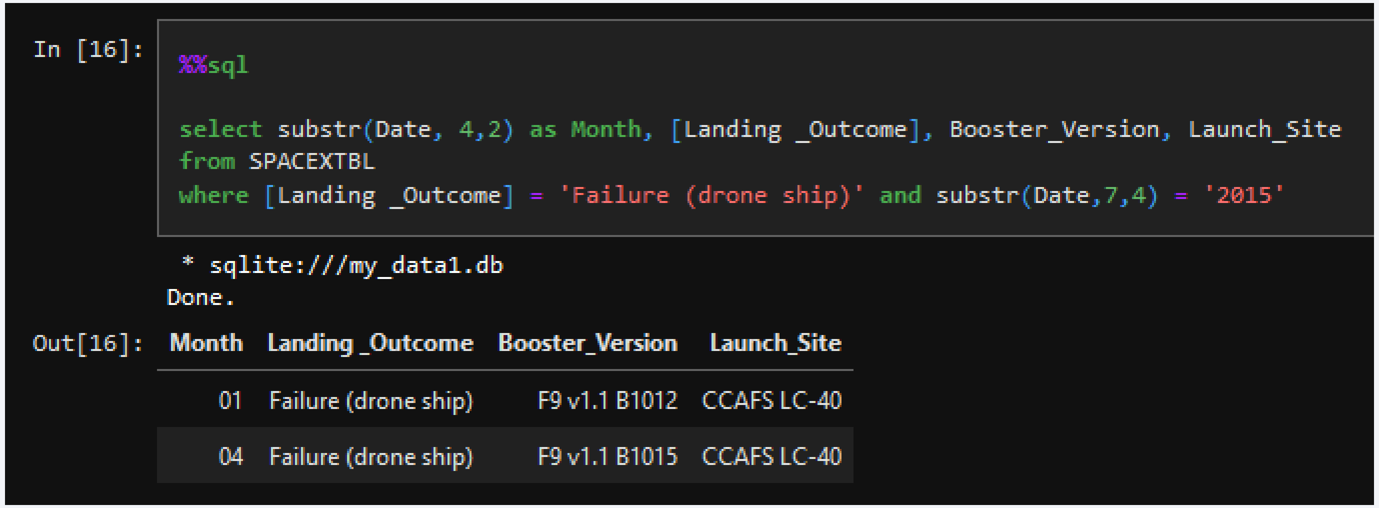
```
In [26]:  %%sql
          select distinct Booster_Version from SPACEXTBL
          where PAYLOAD_MASS__KG_ =
          (select max(PAYLOAD_MASS__KG_) from SPACEXTBL)

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



```
Out[26]:  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 results the boosters carrying maximum payload. ‘DISTINCT’ clause is used to uniquely identify the Booster versions. The sub-query returns the maximum value for the payload using the max().

2015 Launch Records



In [16]:

```
%%sql

select substr(Date, 4,2) as Month, [Landing _Outcome], Booster_Version, Launch_Site
from SPACEXTBL
where [Landing _Outcome] = 'Failure (drone ship)' and substr(Date,7,4) = '2015'
```

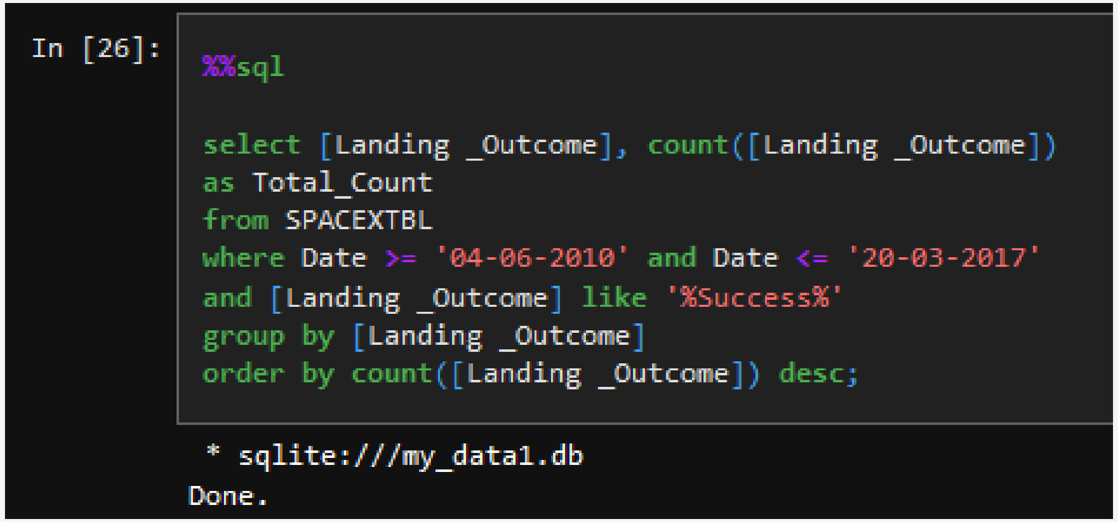
```
 * sqlite:///my_data1.db
Done.
```

Out[16]:

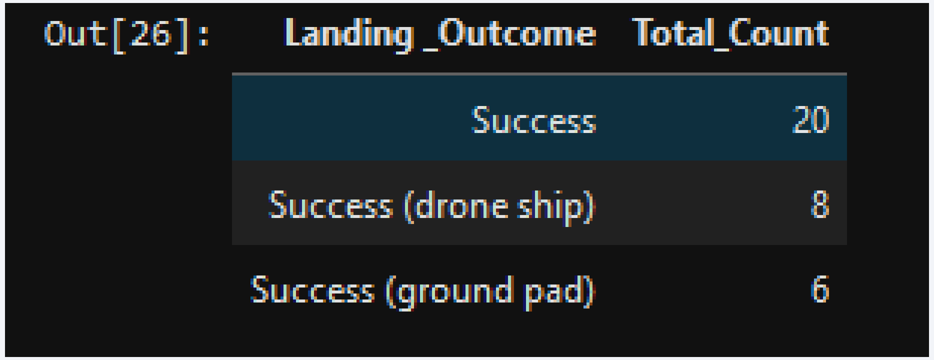
Month	Landing _Outcome	Booster_Version	Launch_Site
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

This query results Month, Landing Outcome, Booster Version and Launch Site.
substr(Date, 4,2) is used to get the month.
substr(Date, 7,4) is used to get the year.
Condition satisfied by the ‘WHERE’ clause.
The condition is fulfilled by passing the landing outcome to be failure in drone ship and the year to be 2015.

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



```
In [26]:  %%sql

          select [Landing _Outcome], count([Landing _Outcome])
          as Total_Count
          from SPACEXTBL
          where Date >= '04-06-2010' and Date <= '20-03-2017'
          and [Landing _Outcome] like '%Success%'
          group by [Landing _Outcome]
          order by count([Landing _Outcome]) desc;

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



Out[26]:

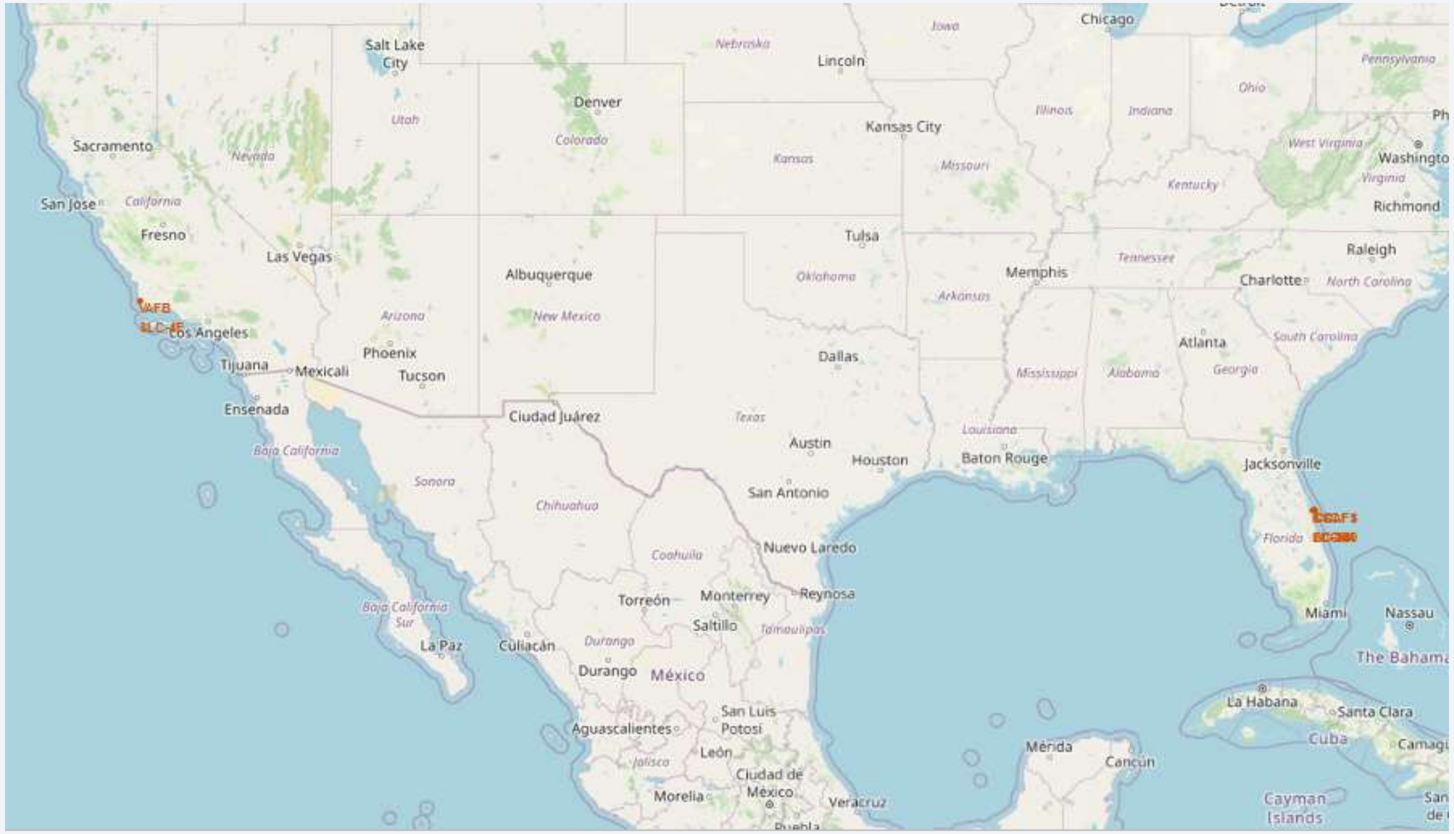
Landing _Outcome	Total_Count
Success	20
Success (drone ship)	8
Success (ground pad)	6

This query results the landing outcomes and its count respectively.
Condition is satisfied by ‘WHERE’ clause.
The condition is the date between 04-06-2010 and 20-03-2017 & ‘Success’.
‘GROUP BY’ clause returns groups by landing outcome.
Ranking is done by ‘ORDER BY’ clause based on the count() of landing outcome in descending order by using ‘DESC’ clause.

Section 3

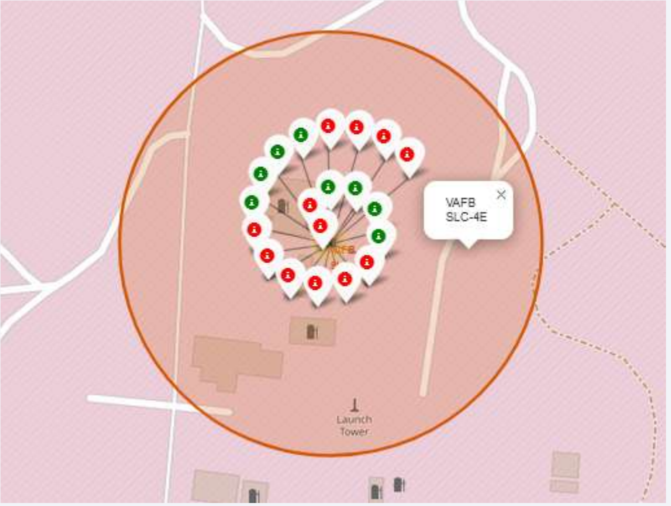
Launch Sites Proximities Analysis

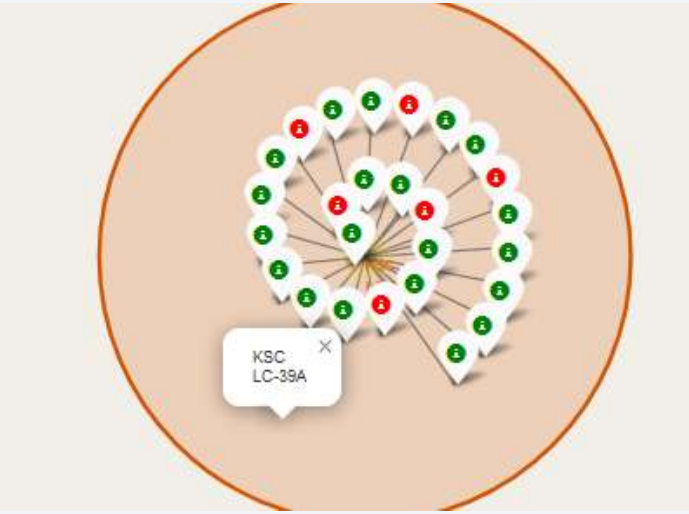
Folium Map with Launch Locations

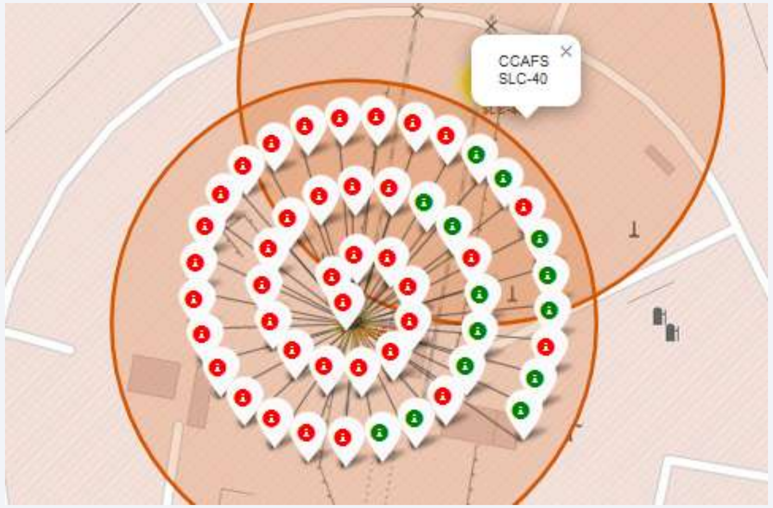


We can observe that the launch locations are situated near coastal area.

Folium Map with colored labeled markers





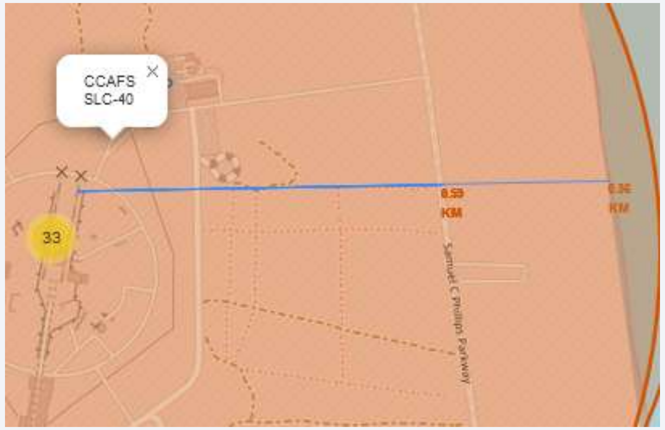


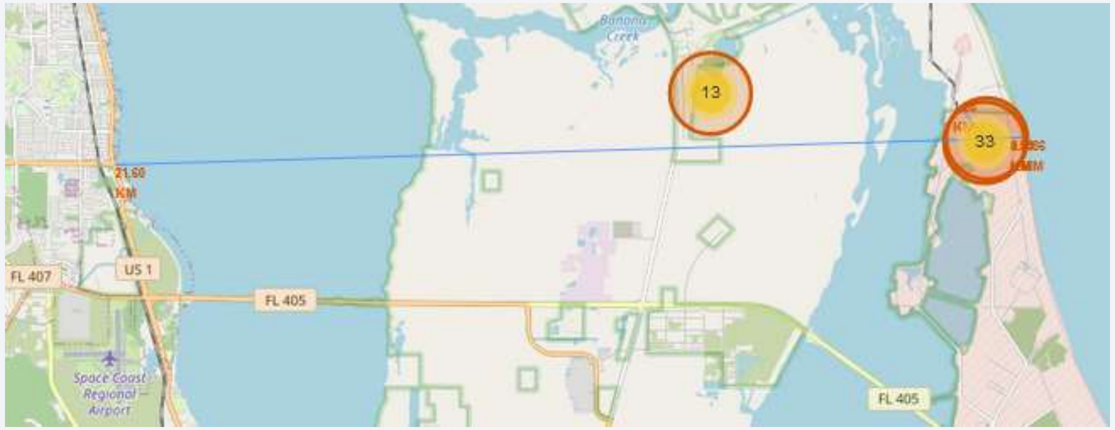
Observe that the labels show:

- Red markers show the unsuccessful launches
- Green markers show the successful launches

Hence, we can consider by the above result that **KSC LC-39A** have the high success rate.

Folium Map – Distances between CCAFS SLC-40 and its proximities



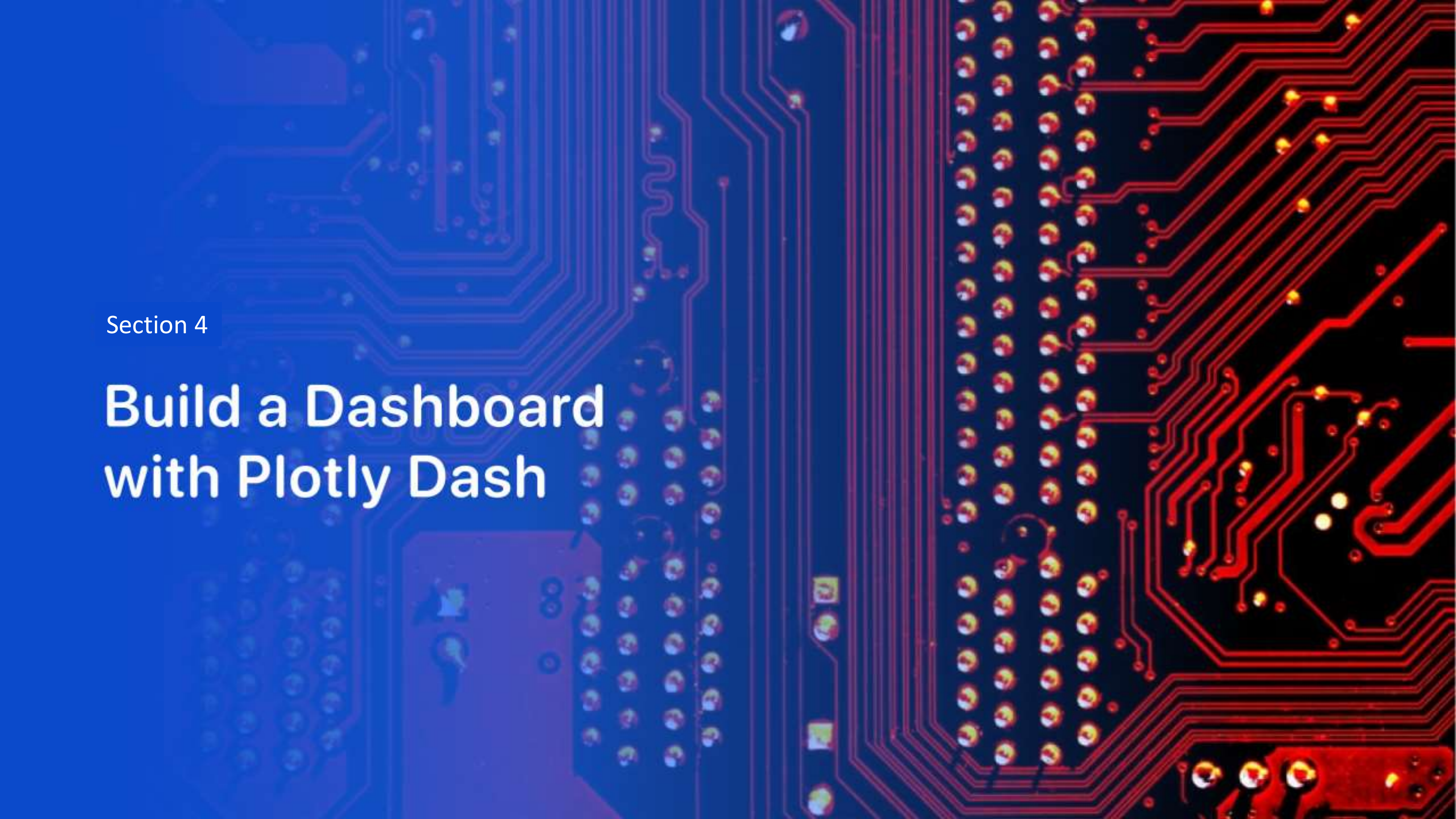


Based on the above observations, we conclude:

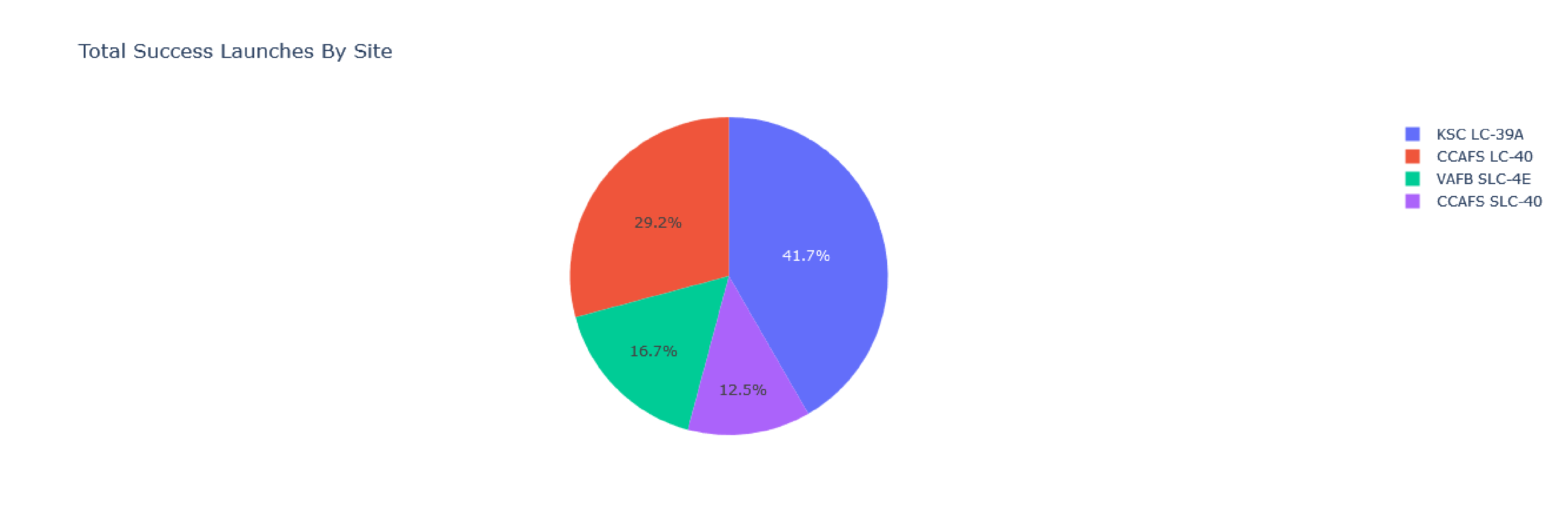
1. CCAFS SLC-40 is close to coastline with a distance 0.86 km.
2. CCAFS SLC-40 is close to highways with a distance 0.59 km.
3. CCAFS SLC-40 is close to railways with a distance 1.23 km.
4. CCAFS SLC-40 is fairly close to city with a distance 21.60 km.

Section 4

Build a Dashboard with Plotly Dash

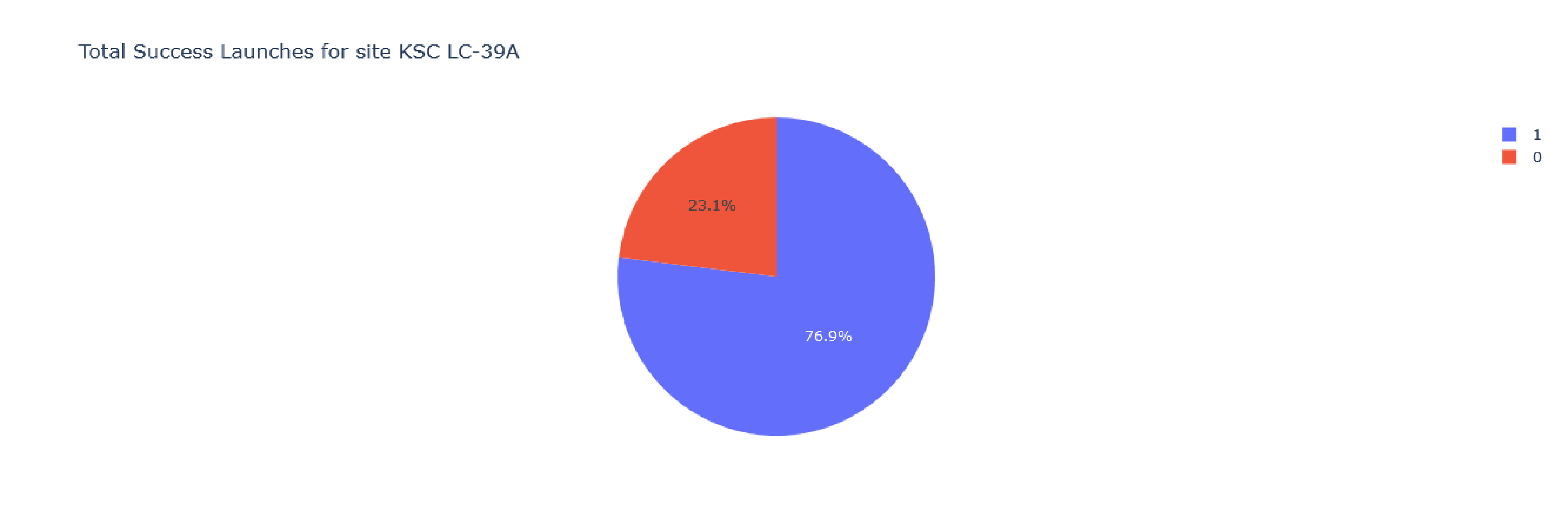


Plotly Dashboard – Total Success Launches By Site



Based on the observation we can rank the highest success launch by site to KSC LC-39A and least success launches by CCAFS SLC-40.

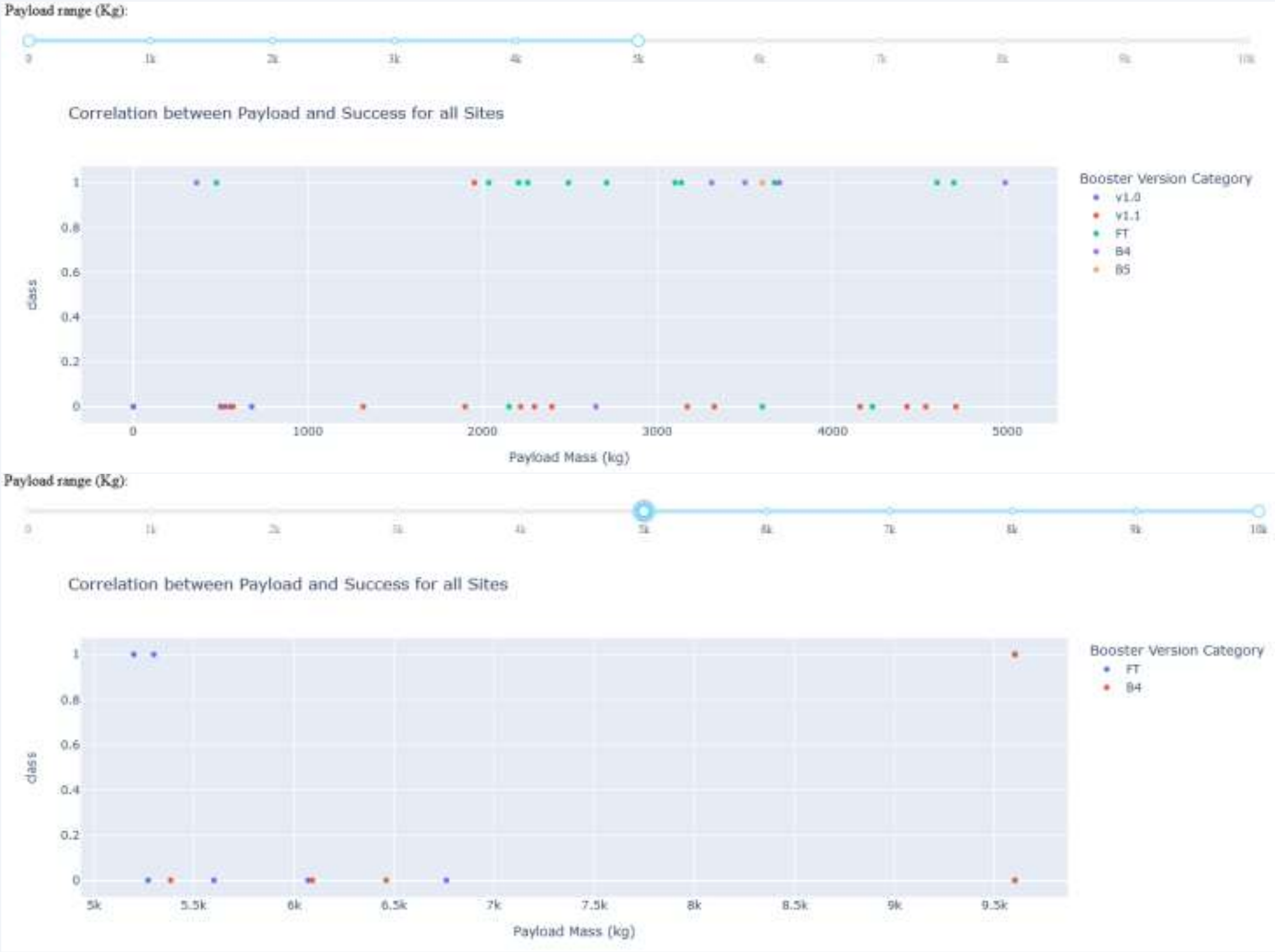
Plotly Dashboard – Total Success Launches for KSC LC-39A



Based on the result we can say that for KSC LC-39A launch site:

- 76.9% have been successful
- 23.1% have been failed

Plotly Dashboard – Payload vs Launch Outcome for all sites with different payloads



First Range (0-5000 Kg):

- Success rate is high when payload is light.

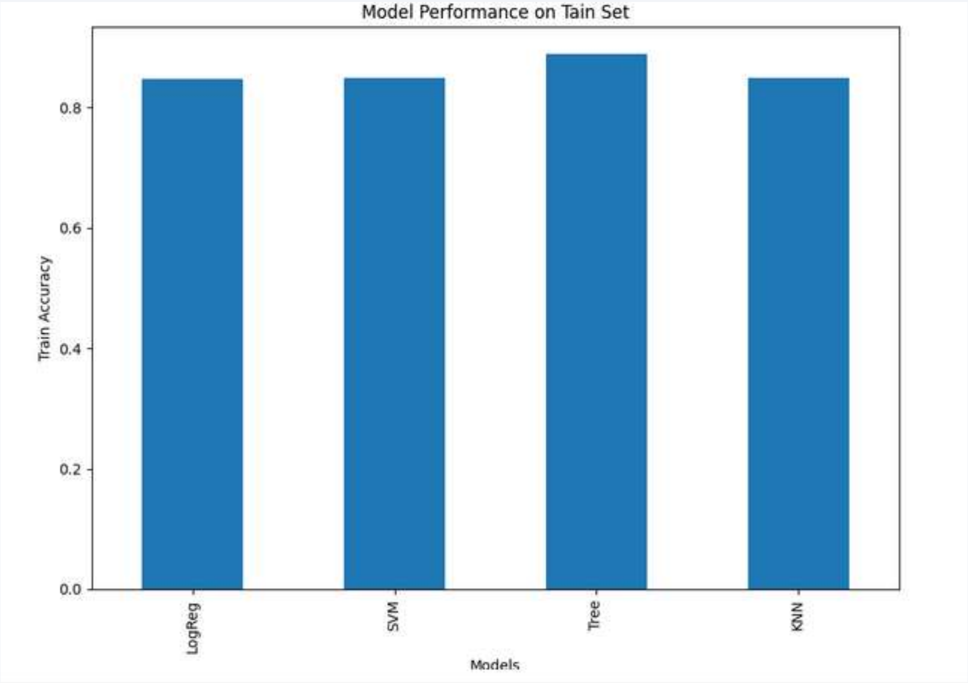
Second Range (5000-10000 Kg):

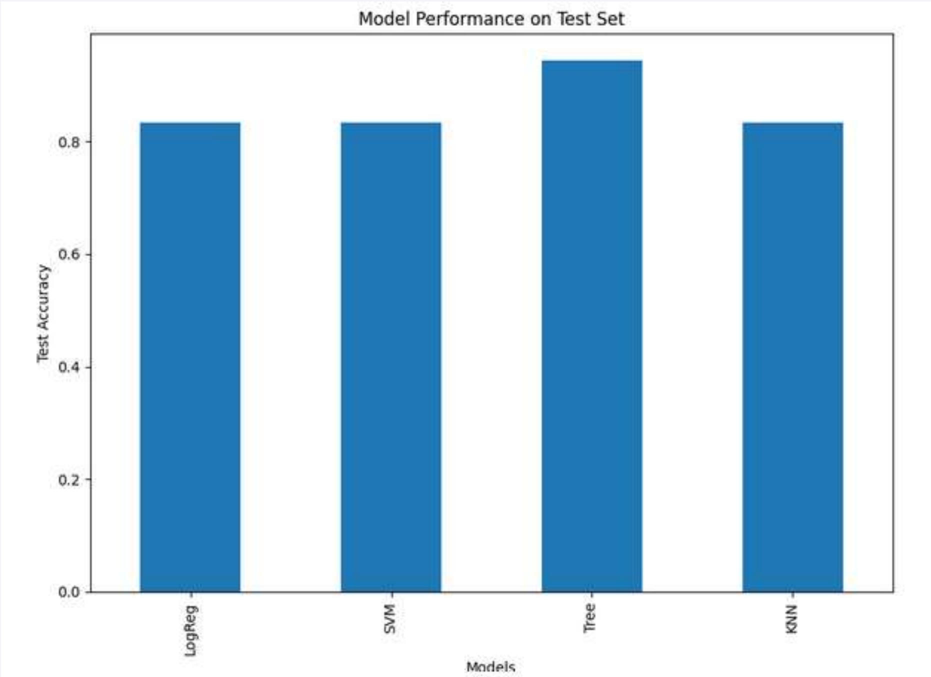
- Comparatively have less success rate with heavy payload.

Section 5

Predictive Analysis (Classification)

Classification Accuracy



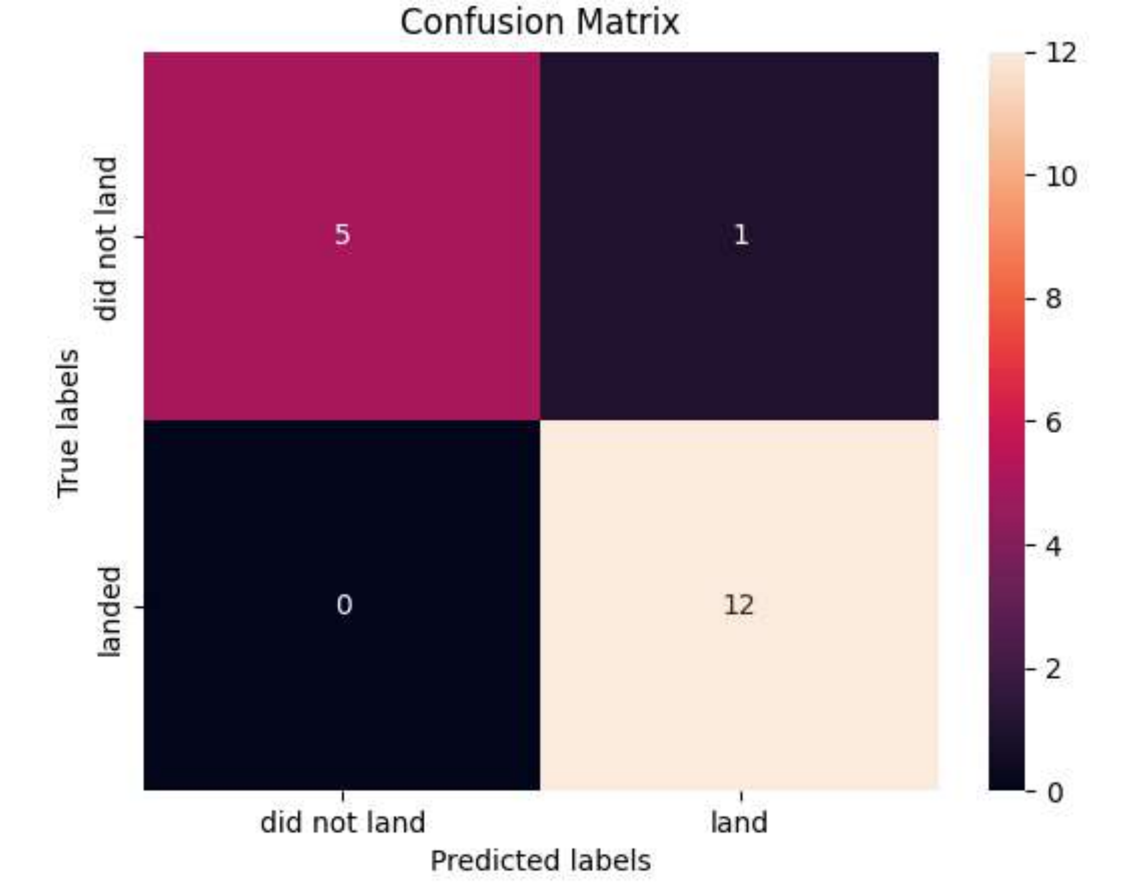


Based on the above observation, we can say:

- Training Accuracy is better using Decision Tree
- Testing Accuracy is best using Decision Tree

Hence, ideal model to work on with would be Decision Tree for the test data.

Confusion Matrix



Given is the confusion matrix for the Decision Tree.

- The best error rate is 0.0
- The worst error rate is 1.0

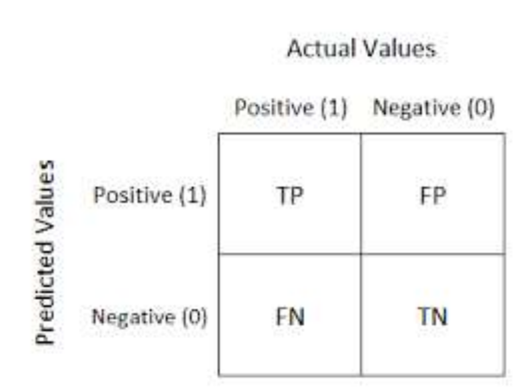
As the total number of the dataset is 18.

Based on the observation we can calculate the error rate as follows:

Error = (FP+FN/Total number of the data set)

Error = (1+0/5+12+0+1) = 0.055

As this confusion matrix has the lowest False Positives(1) and least Error Rate(0.055), we can consider this model to be the best model.



Actual Values

Predicted Values	Positive (1)	Negative (0)
Positive (1)	TP	FP
Negative (0)	FN	TN

Conclusions

- To determine the successful landing outcome can be evaluated based on various characteristics. Before starting analysis, we are dependent on the previous records to get idea of whole process which is reasonable. By the help of previous records, we can train models, assess the accuracy on training data, test data, and finally try to test on out-of-sample data.
- By the help of EDA, we could possibly get the scenario based on various relations like payload being moderate, we observed that there was high success rate, etc.
- Using Bar chart, we made clear that ES-L1, GEO, HEO and SSO orbit type have the best success rates among other types.
- Based on the yearly trend we observed the success rate increasing, reason being the knowledge gain from the historical data, models have been trained such way that they are able to produce reasonable conclusions.
- For our purpose, we have used classifier models, to generate the accuracies based on the diverse models, we compared the accuracies for each model.
- We inferred that the Decision Tree model seems to be the best for the prediction due to the lowest error rate and False Positives among the other models.

Appendix

- Data Source:
  - https://api.spacexdata.com/v4
  - https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches
- GitHub Repository:
  - https://github.com/D1N3SH-DEV/Project-Capstone-SpaceY/tree/master
  - This repository contains all the code for the given project.

Thank you!