IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Apoorva
26TH DEC 2022

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection via API, Web Scraping
  - Exploratory Data Analysis (EDA) with Data Visualization
  - Exploratory Data Analysis(EDA) with SQL
  - Interactive Map with Folium
  - Dashboards with Plotly Dash
  - Predictive Analysis
- Summary of all results
  - Exploratory Data Analysis results
  - Interactive maps and dashboard
  - Predictive results

Introduction

- Project background and context
- With the advent of commercial space age, a pertinent question that arises is “How much does a rocket launch cost?”. The most popular space company, SpaceX, advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars, and other providers cost upwards of 165 million dollars each. Here the difference exist because SpaceX can reuse the first stage. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch. And this project aims to help SpaceY determine the price of each launch.

- Problems you want to find answers
  - determine if SpaceX will reuse the first stage.
  - determine if the first stage will land successfully,
  - predict if SpaceX will reuse the first stage.

Section 1

Methodology

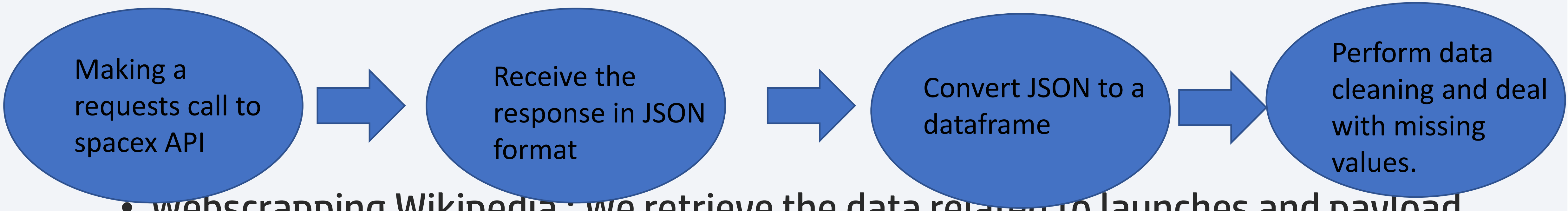
Methodology

Executive Summary

- Data collection methodology:
  - We can collect data by using SpaceX API, and also by web-scraping the wikipedia
- Perform data wrangling
  - Dealing with missing values
  - determining what would be the label for training supervised models
  - One Hot Encoding for 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
  - How to build, tune, evaluate classification models

Data Collection

- We can use two methods to collect data :
  - SpaceX API : We retrieve the data related to rockets, launches and payload



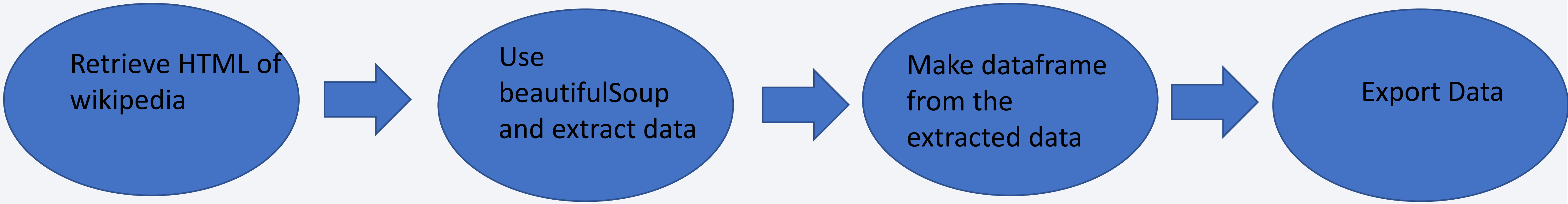
Making a requests call to spacex API

Receive the response in JSON format

Convert JSON to a dataframe

Perform data cleaning and deal with missing values.

  - Webscrapping Wikipedia : We retrieve the data related to launches and payload information.



Retrieve HTML of wikipedia

Use beautifulSoup and extract data

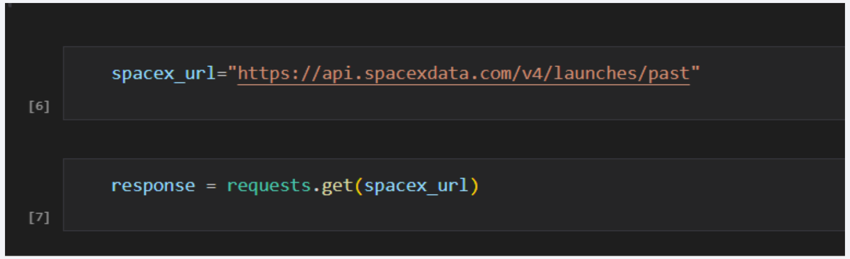
Make dataframe from the extracted data

Export Data

Data Collection – SpaceX API

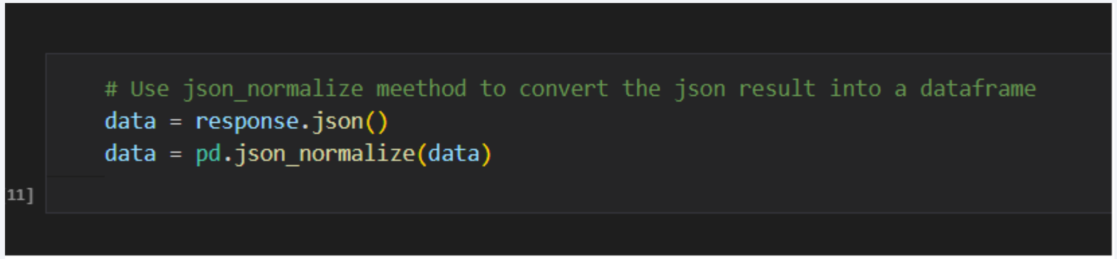
LINK TO GITHUB

I. Using the request library to get the data



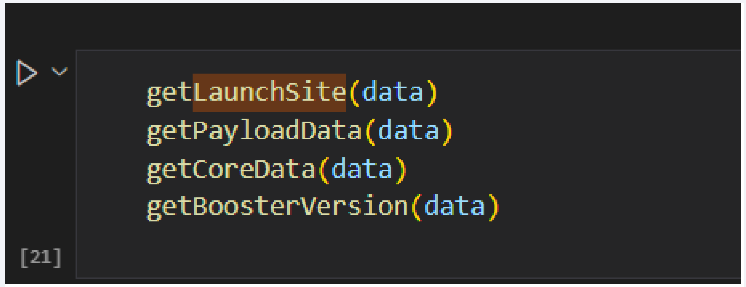
```
spacex_url="https://api.spacexdata.com/v4/launches/past"
[6]

response = requests.get(spacex_url)
[7]
```

II. Converting it to JSON



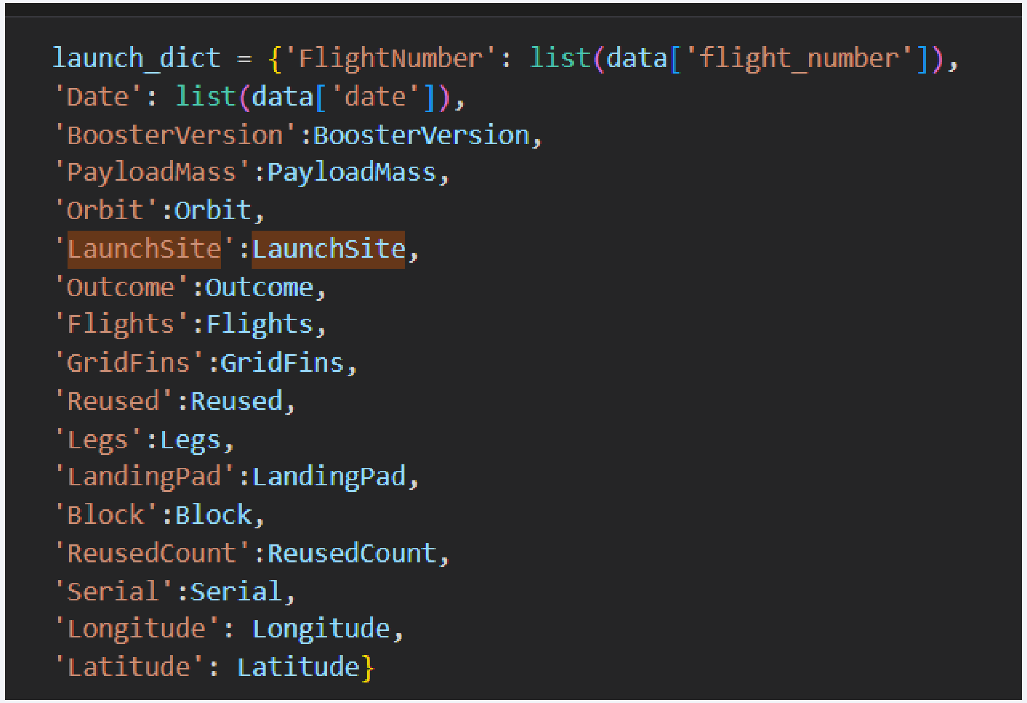
```
# Use json_normalize meethod to convert the json result into a dataframe
data = response.json()
data = pd.json_normalize(data)
11]
```

III. Tranform data



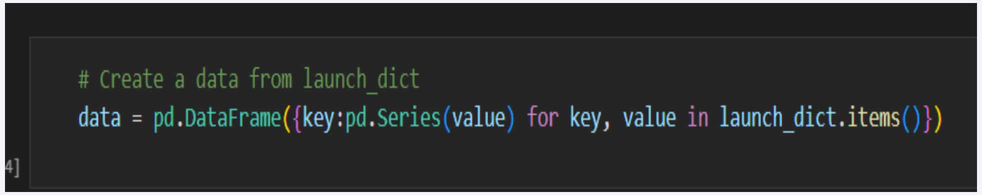
```
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
getBoosterVersion(data)
[21]
```

IV. Convert the data into a dictionary



```
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}
```

V. Create the dataframe from the dictionary

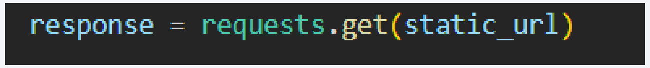


```
# Create a data from launch_dict
data = pd.DataFrame({key:pd.Series(value) for key, value in launch_dict.items()})
4]
```

Data Collection - Scraping

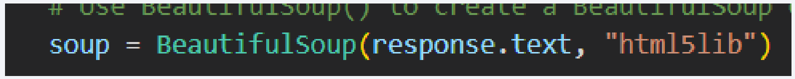
LINK TO GITHUB

I. Get response text from the url



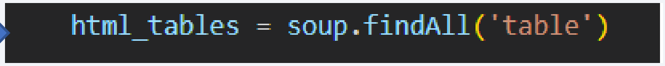
```
response = requests.get(static_url)
```

II. Creating a beautiful soup object



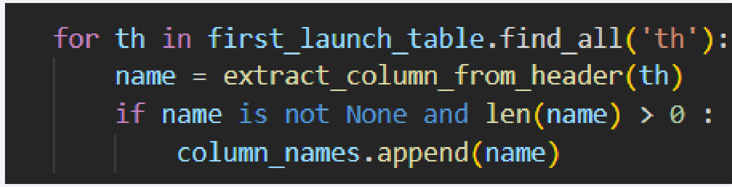
```
# Use BeautifulSoup() to create a BeautifulSoup
soup = BeautifulSoup(response.text, "html5lib")
```

III. Find tables



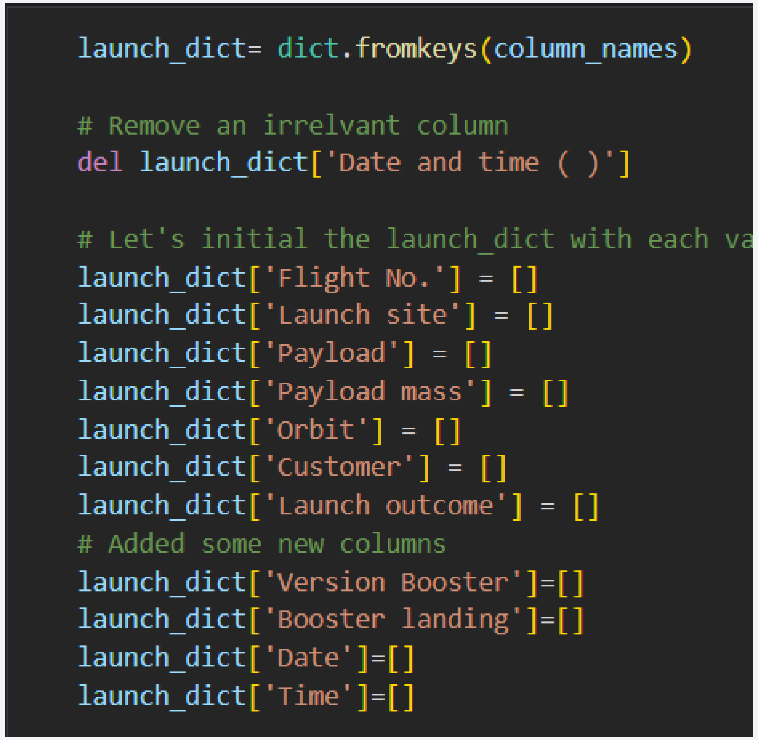
```
html_tables = soup.findAll('table')
```

IV. Find the column names



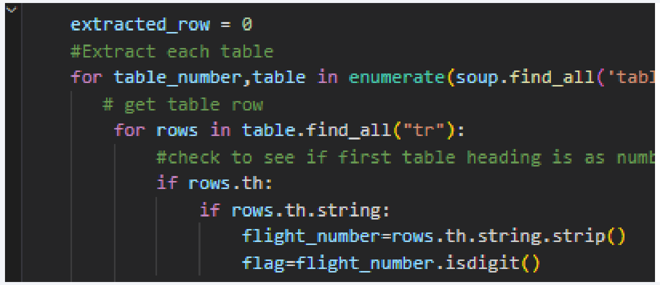
```
for th in first_launch_table.find_all('th'):
    name = extract_column_from_header(th)
    if name is not None and len(name) > 0 :
        column_names.append(name)
```

V. Creating a dictionary



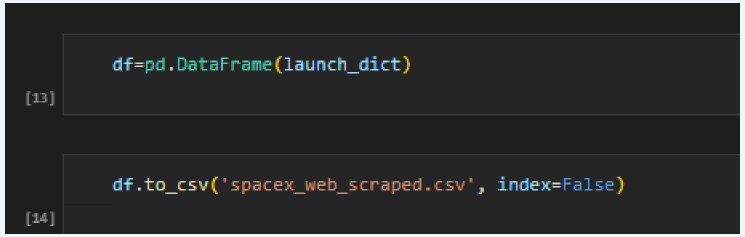
```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each va
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

VI. Add data to keys



```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('tabl
  # get table row
   for rows in table.find_all("tr"):
       #check to see if first table heading is as numb
       if rows.th:
           if rows.th.string:
               flight_number=rows.th.string.strip()
               flag=flight_number.isdigit()
```

VII. Make dataframe and then export to csv.



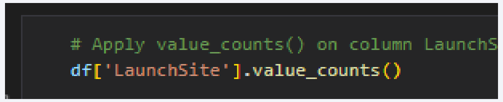
```
[13]  df=pd.DataFrame(launch_dict)

[14]  df.to_csv('spacex_web_scraped.csv', index=False)
```

Data Wrangling

LINK TO GITHUB

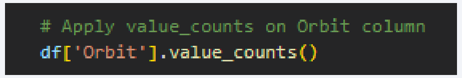
- We have to transform categorical variables where
  - 1 corresponds to True Ocean, True RTLS, True ASDS(successful mission)
  - 0 corresponds to False Ocean, False RTLS, False ASDS(failed mission)

I. Count number of LaunchSites



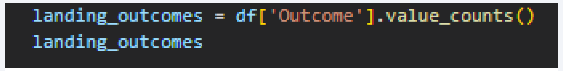
```
# Apply value_counts() on column LaunchS
df['LaunchSite'].value_counts()
```

II. Find the count of each orbit



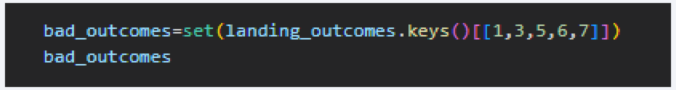
```
# Apply value_counts on Orbit column
df['Orbit'].value_counts()
```

III. Calculate number and occurrence of mission outcome per orbit type



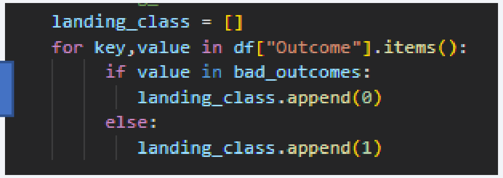
```
landing_outcomes = df['Outcome'].value_counts()
landing_outcomes
```

IV. Make a bad_outcomes array



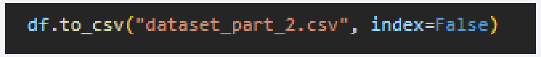
```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

V. Creating an outcomes labels outcome columns



```
landing_class = []
for key,value in df["Outcome"].items():
    if value in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

VI. Converting to CSV



```
df.to_csv("dataset_part_2.csv", index=False)
```

LINK TO GITHUB

EDA with Data Visualization

- I used the following graphs :
  - Scatter Plots
    - This is used to show the correlation between different variables
  - Bar Graphs
    - This is used to show the relationship between categorical data(on x-axis) and discrete numeric data(on y-axis)
  - Line Graphs
    - Line graphs are used **to track changes over short and long periods of time.**

Y-AXIS	X-AXIS	TYPE OF GRAPH
Payload Mass	Flight Number	Scatter Plot
Launch Site	Flight Number	Scatter Plot
Launch Site	Payload	Scatter Plot
Flight Number	Orbit	Scatter Plot
Orbity type	Payload	Scatter Plot
Payload Mass	Orbit	Scatter Plot
Success Rate	Orbit	Bar graph
Success Rate	Year	Line graph

EDA with SQL

LINK TO GITHUB

- The SQL queries done to perform EDA were :
  - •Displaying 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.
  - •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 date04-06-2010 and 20-03-2017in descending order.

Build an Interactive Map with Folium

[LINK TO GITHUB]

- Create a folium `Map` object, with an initial center location to be NASA Johnson Space Center at Houston, Texas.
- use `folium.Circle` to add a highlighted circle area with a text label on a specific coordinate name(folium.Circle, folium.map.Marker).
- Red circles at each launch site coordinates with label showing launch site name (folium.Circle, folium.map.Marker, folium.features.DivIcon).
- The grouping of points in a cluster to display multiple and different information for the same coordinates (folium.plugins.MarkerCluster).
- Green marrkers to show successful landing and red to show unsuccessful landings. (folium.map.Marker, folium.Icon).
- Plot a line between them launch site to key locations to show the distance. (folium.map.Marker, folium.PolyLine, folium.features.DivIcon*)*
- These objects are created in order to understand better the problem and visualize our data like showing on the map the launch sites, their surroundings and the number of successful and unsuccessful landings.

Build a Dashboard with Plotly Dash

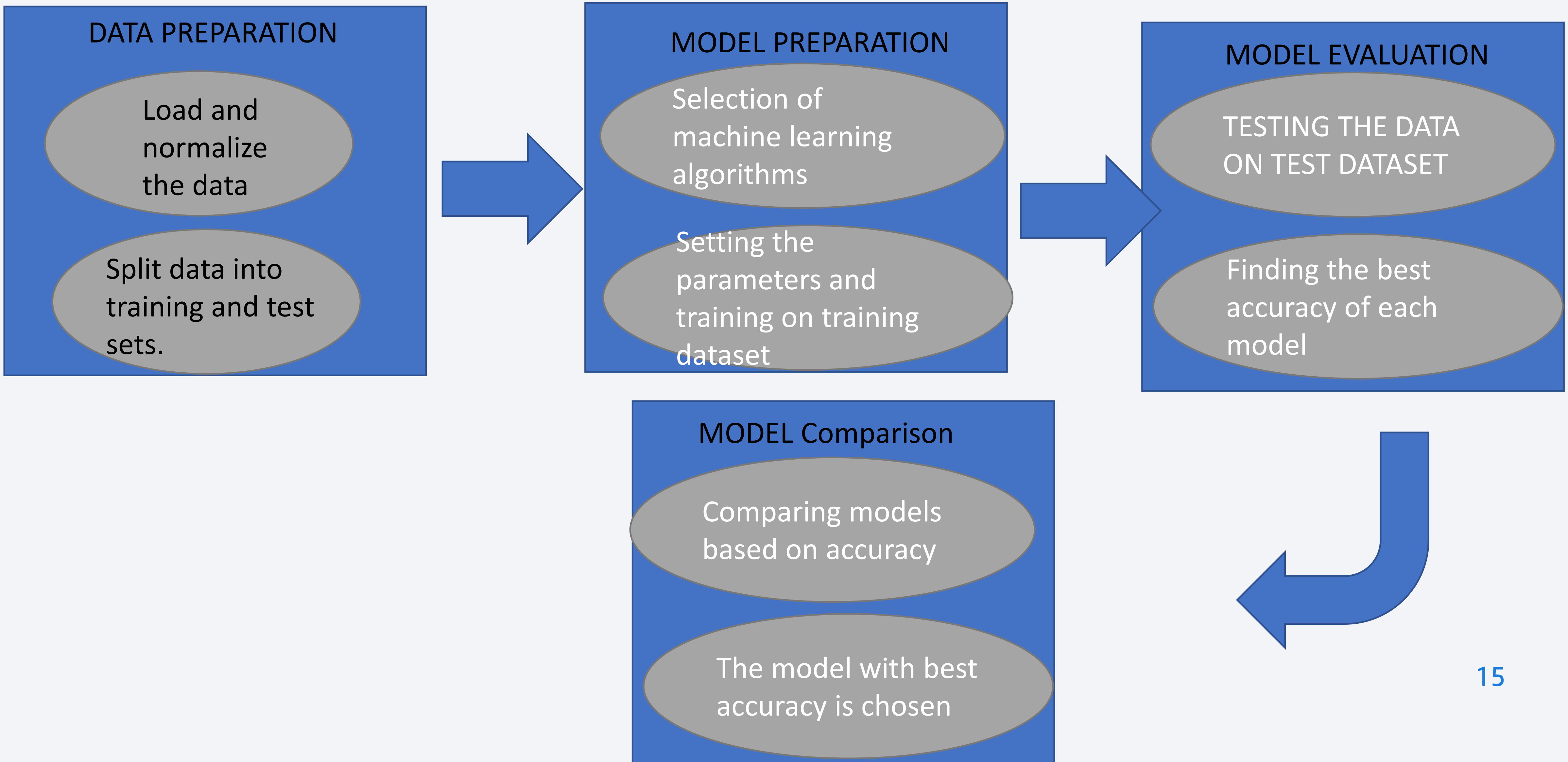
LINK TO GITHUB

The following components are part of the dashboard :

- Scatter chart shows the relationship between two variables, in particular Success vs Payload Mass (plotly.express.scatter).
- Dropdownallows a user to choose the launch site or all launch sites (dash_core_components.Dropdown).
- Pie chart shows the total success and the total failure for the launch site chosen with the dropdown component(plotly.express.pie).
- Rangeslider allows a user to select a payload mass in a fixed range (dash_core_components.RangeSlider).

LINK TO GITHUB

Predictive Analysis (Classification)



DATA PREPARATION

Load and normalize the data

Split data into training and test sets.

MODEL PREPARATION

Selection of machine learning algorithms

Setting the parameters and training on training dataset

MODEL EVALUATION

TESTING THE DATA ON TEST DATASET

Finding the best accuracy of each model

MODEL Comparison

Comparing models based on accuracy

The model with best accuracy is chosen

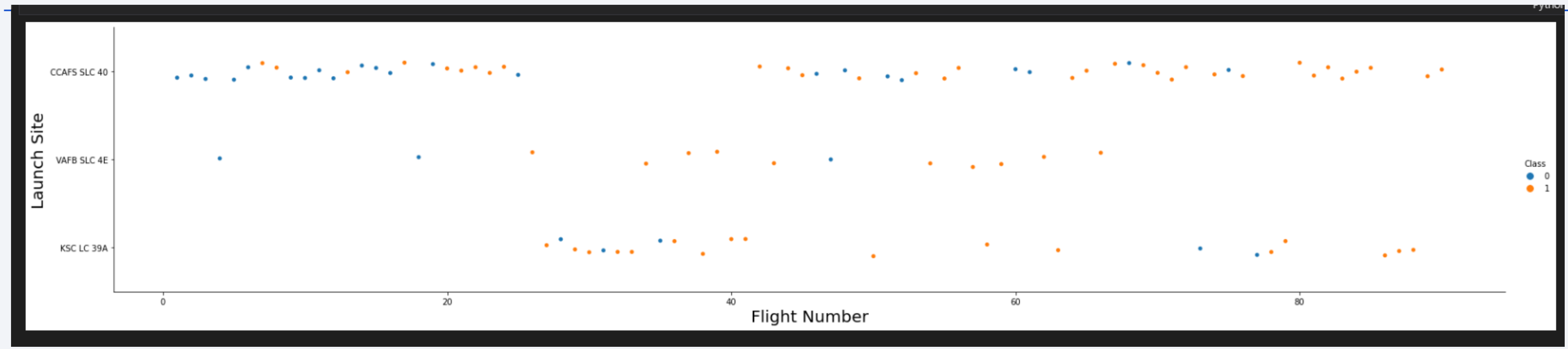
Results

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

Section 2

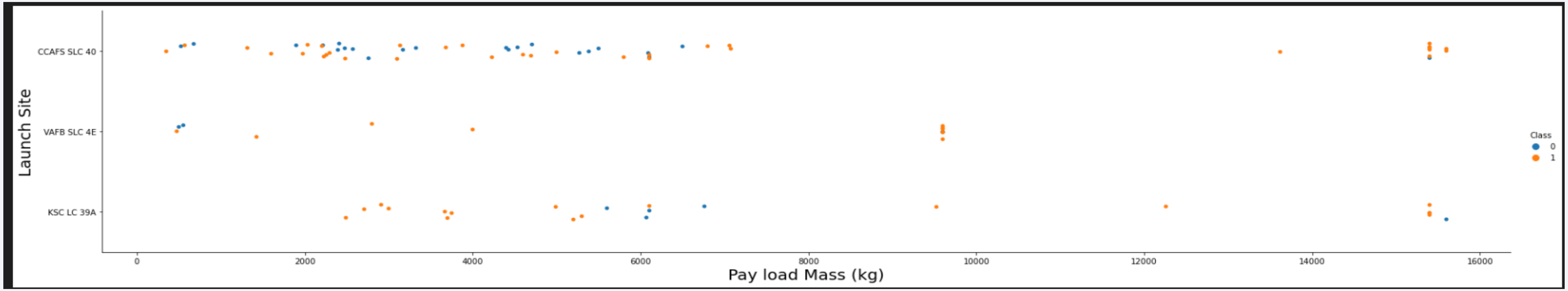
Insights drawn from EDA

Flight Number vs. Launch Site



- The above plot shows that the success rate is increasing with every launch site.

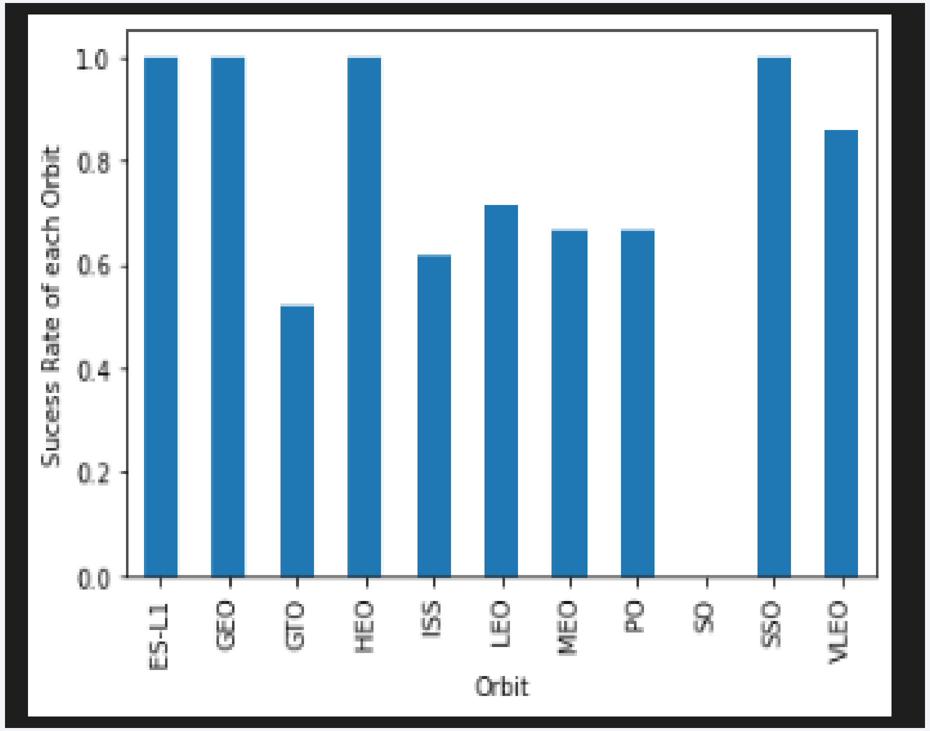
Payload vs. Launch Site



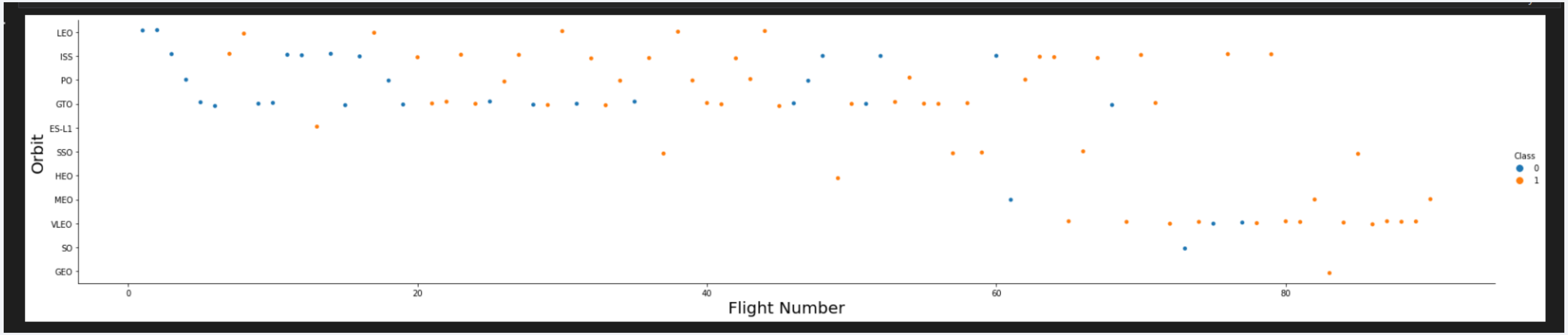
- We can observe that for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).
- Also from the plot we can see that heavier payload(>10000 kg) can result a failure of landing.

Success Rate vs. Orbit Type

- We observe that orbit SO results in 0% success.
- We also observe that orbits ES-L1, GEO, HEO, SSO have the best success rate.

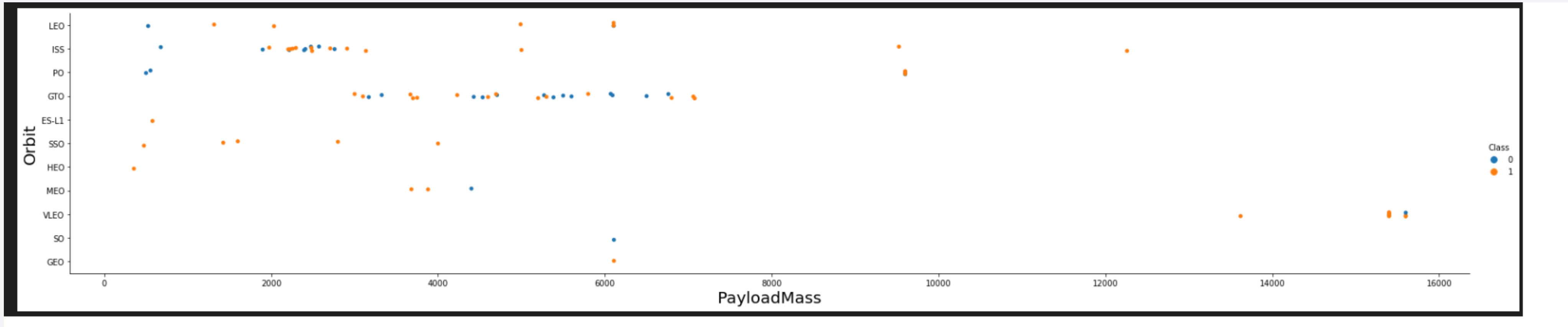


Flight Number vs. Orbit Type



- We can observe that in the LEO orbit the Success appears related to the number of flights.
- On the other hand, there seems to be no relationship between flight number when in GTO orbit.
- We can observe that for most of the orbits the success is observed at later stage which might be a result of previous learnings.

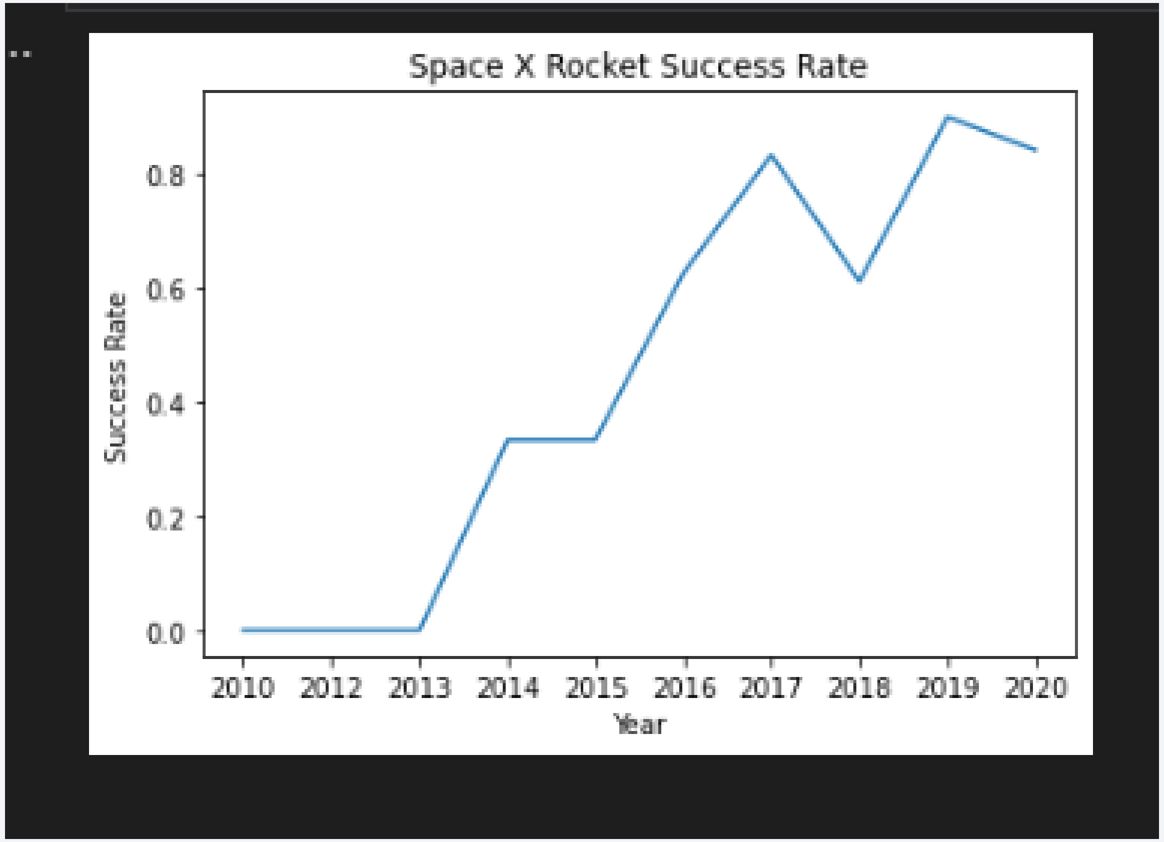
Payload vs. Orbit Type



- The weight of the payloads can have a great influence on the success rate of the launches in certain orbits. For example, heavier payloads improve the success rate for the LEO orbit. Another finding is that decreasing the payload weight for a GTO orbit improves the success of a launch.

Launch Success Yearly Trend

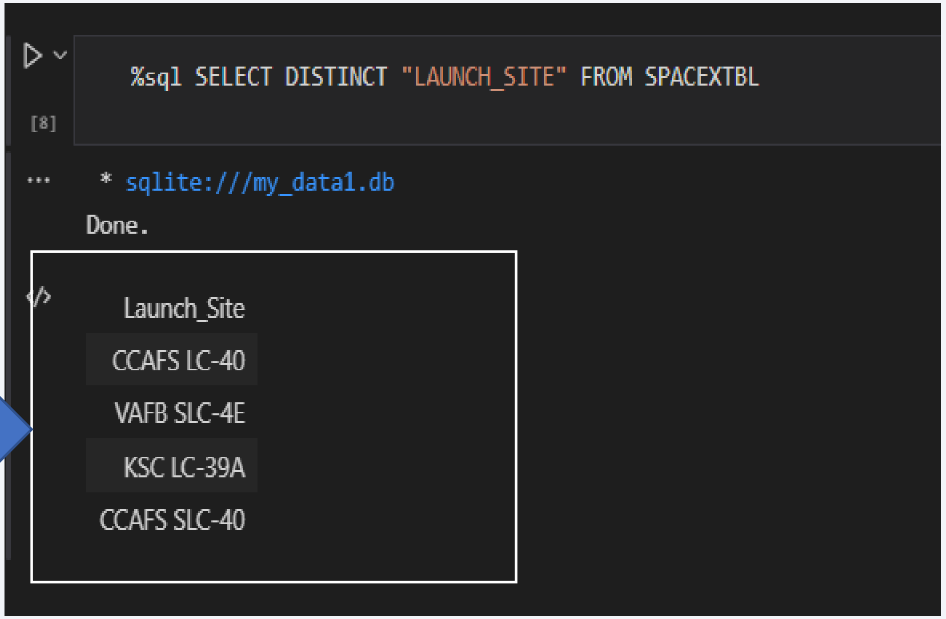
- Since 2013, we can see an increase in the Space X Rocket success rate(except from 2017-2018).



All Launch Site Names

- This query helps us in selecting all of the distinct launch site names from the dataset.

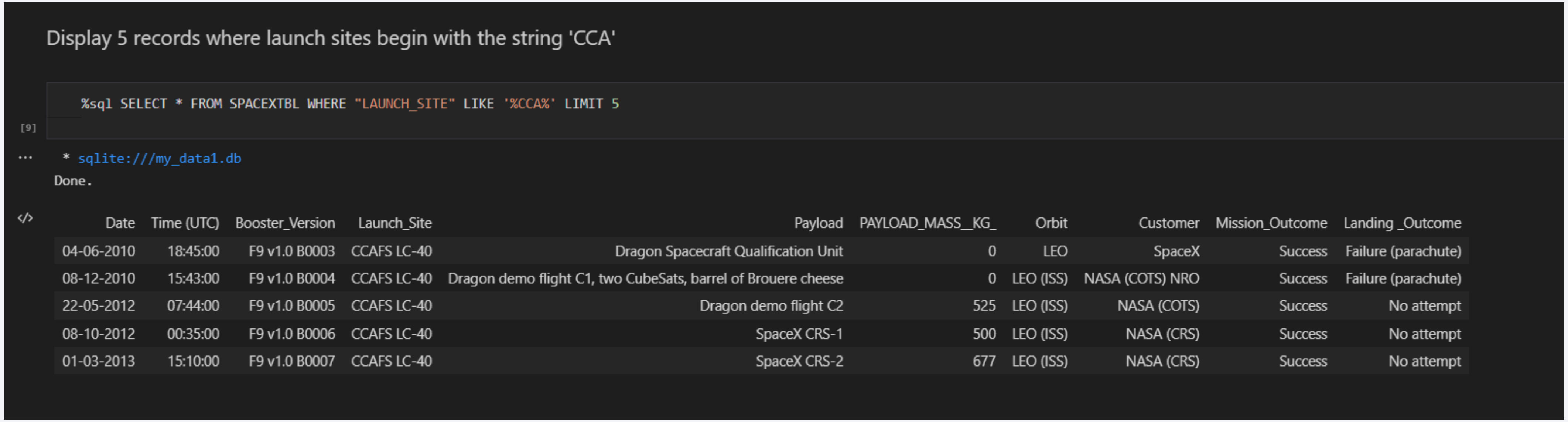
RESULT



```
[8] %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'



Display 5 records where launch sites begin with the string 'CCA'

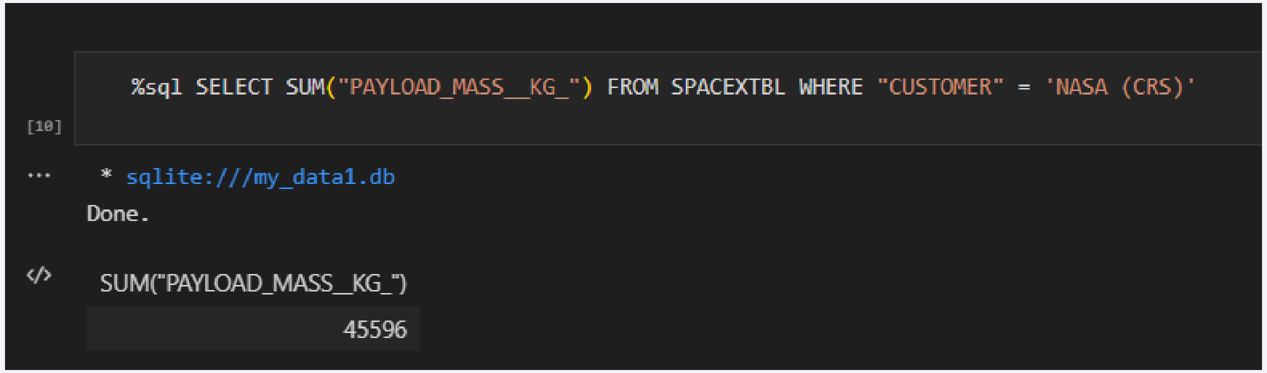
```
%sql SELECT * FROM SPACEXTBL WHERE "LAUNCH_SITE" LIKE '%CCA%' LIMIT 5
```

```
 * 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

- The above query displays 5 records with WHERE clause followed by LIKE clause filters launch sites that contain the substring CCA.LIMIT 5 shows 5 records from filtering.

Total Payload Mass

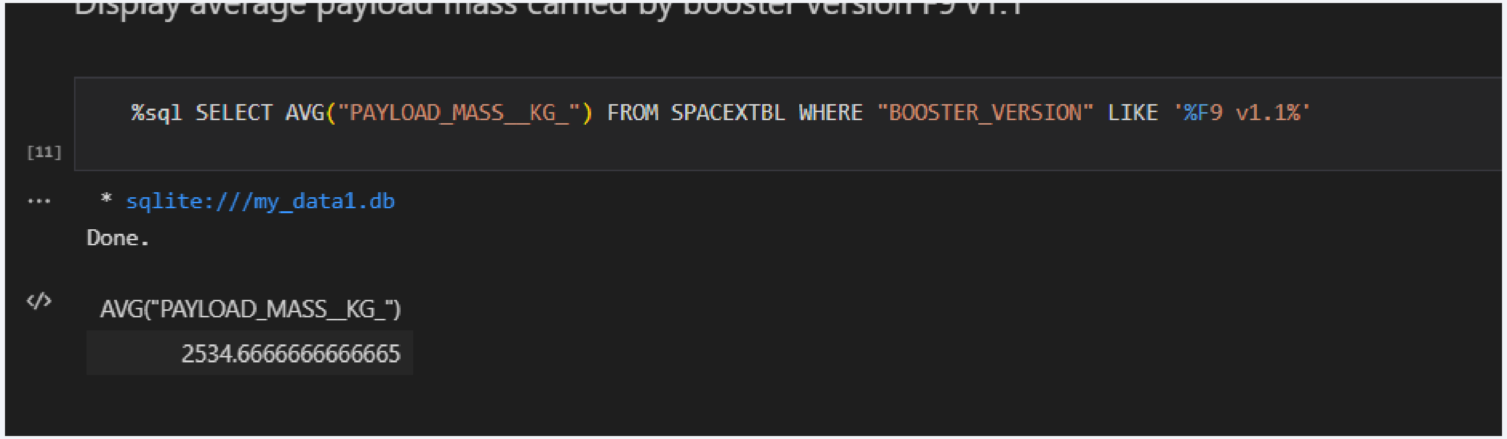


```
[10]  %sql SELECT SUM("PAYLOAD_MASS__KG_") FROM SPACEXTBL WHERE "CUSTOMER" = 'NASA (CRS)'
```

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

 SUM("PAYLOAD_MASS__KG_")
                    45596
```

- The above query gives us the total payload mass by using aggregation “SUM” function.

Average Payload Mass by F9 v1.1



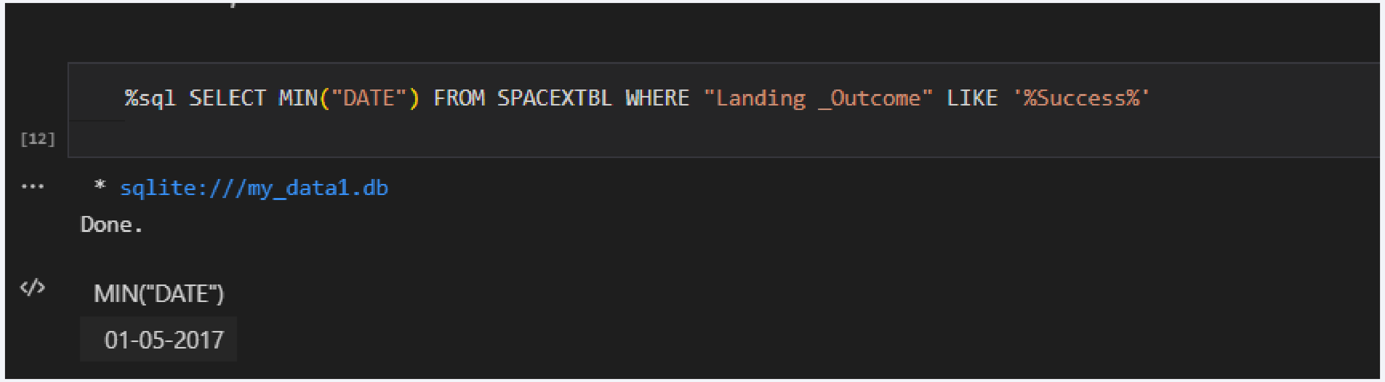
Display average payload mass carried by booster version F9 v1.1

```
%sql SELECT AVG("PAYLOAD_MASS__KG_") FROM SPACEXTBL WHERE "BOOSTER_VERSION" LIKE '%F9 v1.1%'
```

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

 AVG("PAYLOAD_MASS__KG_")
       2534.6666666666665
```

- We can calculate the average payload here by using “AVG” function.

First Successful Ground Landing Date



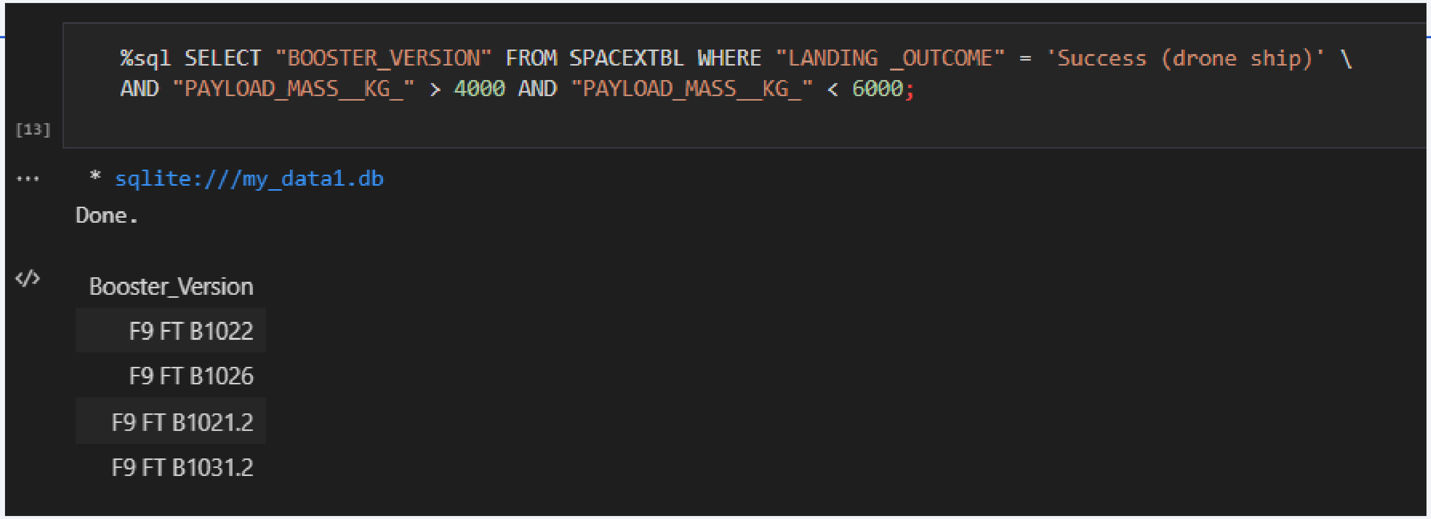
```
%sql SELECT MIN("DATE") FROM SPACEXTBL WHERE "Landing _Outcome" LIKE '%Success%'
```

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

MIN("DATE")
01-05-2017

- We can Find the date of the first successful landing outcome on ground pad as we are using the MIN function on the date attribute.

Successful Drone Ship Landing with Payload between 4000 and 6000

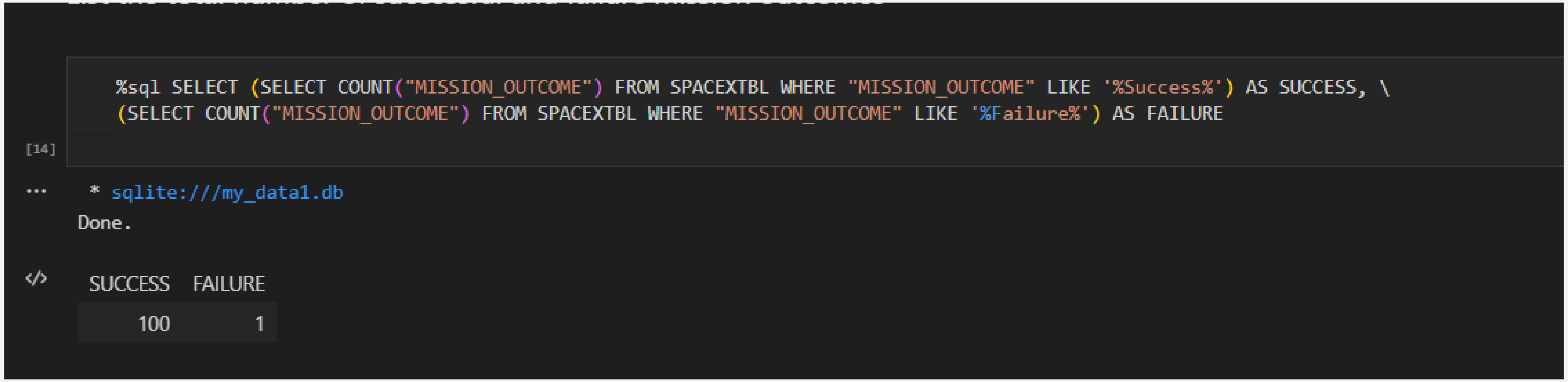


```
[13]
%sql SELECT "BOOSTER_VERSION" FROM SPACEXTBL WHERE "LANDING _OUTCOME" = 'Success (drone ship)' \
AND "PAYLOAD_MASS__KG_" > 4000 AND "PAYLOAD_MASS__KG_" < 6000;
```

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

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

- By using the above query we can list the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000 as we are applying two conditions
  - Successful landing outcome
  - Payload mass >4000 and < 6000

Total Number of Successful and Failure Mission Outcomes

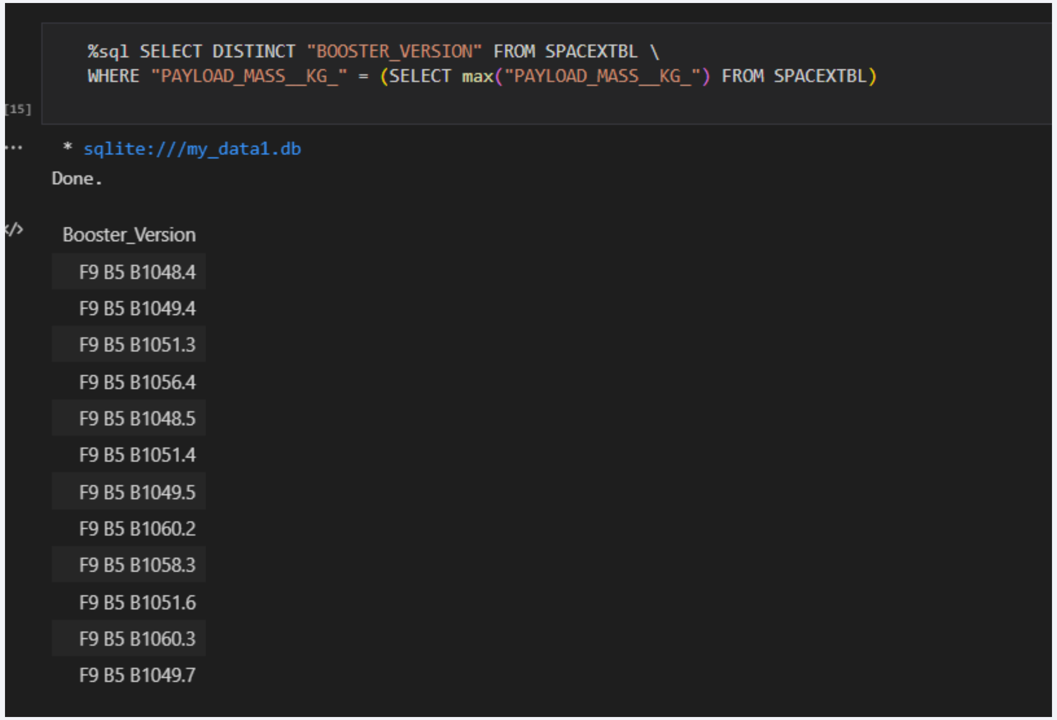


```
%sql SELECT (SELECT COUNT("MISSION_OUTCOME") FROM SPACEXTBL WHERE "MISSION_OUTCOME" LIKE '%Success%') AS SUCCESS, \
(SELECT COUNT("MISSION_OUTCOME") FROM SPACEXTBL WHERE "MISSION_OUTCOME" LIKE '%Failure%') AS FAILURE
```

```
[14]
 * sqlite:///my_data1.db
Done.

SUCCESS  FAILURE
    100        1
```

- By using the count function we can count the total number of successes and failures.

Boosters Carried Maximum Payload



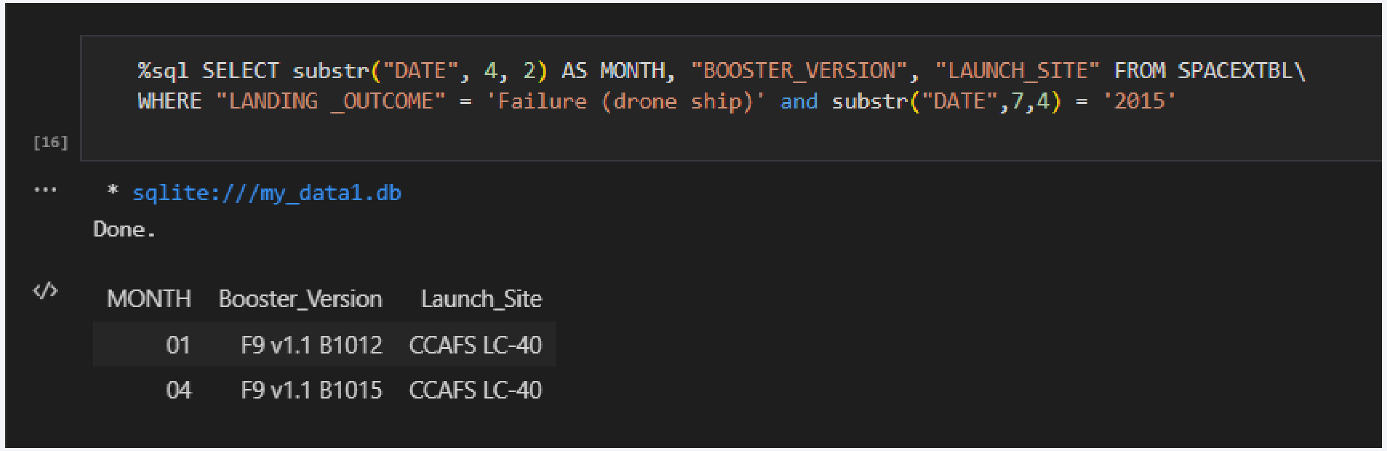
```
%sql SELECT DISTINCT "BOOSTER_VERSION" FROM SPACEXTBL \
WHERE "PAYLOAD_MASS__KG_" = (SELECT max("PAYLOAD_MASS__KG_") FROM SPACEXTBL)
```

```
 * 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

- By using the above query we can list the names of the booster which have carried the maximum payload mass by using the MAX function.

2015 Launch Records

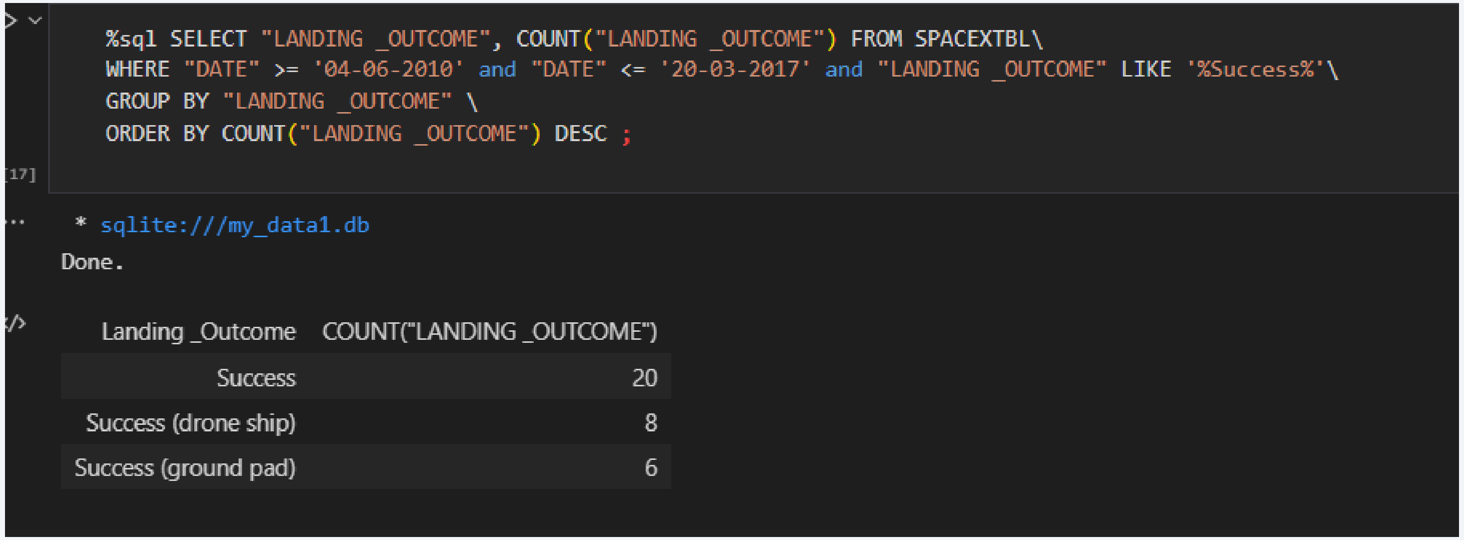


```
[16]
%sql SELECT substr("DATE", 4, 2) AS MONTH, "BOOSTER_VERSION", "LAUNCH_SITE" FROM SPACEXTBL\
WHERE "LANDING _OUTCOME" = 'Failure (drone ship)' and substr("DATE",7,4) = '2015'
```

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

MONTH  Booster_Version   Launch_Site
   01  F9 v1.1 B1012     CCAFS LC-40
   04  F9 v1.1 B1015     CCAFS LC-40
```

- By the above query we can list the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015 by setting the conditions as
  - LANDING_OUTCOME = FAILURE
  - And searching for substring 2015 in date

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



```
%sql SELECT "LANDING _OUTCOME", COUNT("LANDING _OUTCOME") 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.
```

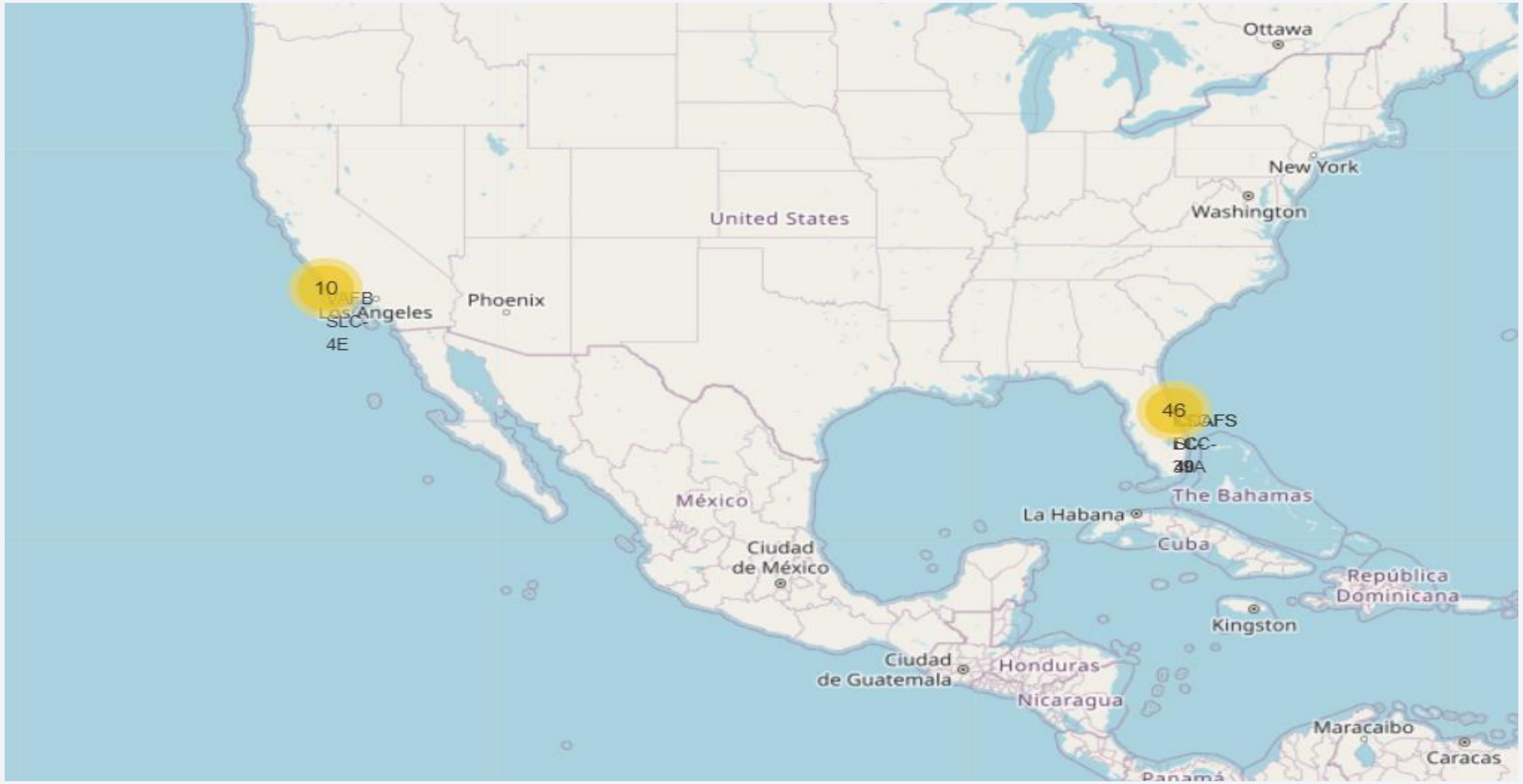
Landing _Outcome	COUNT("LANDING _OUTCOME")
Success	20
Success (drone ship)	8
Success (ground pad)	6

- By using the above query we can show the results in descending order between the date 2010-06-04 and 2017-03-20 by using the function “ORDER BY”

Section 3

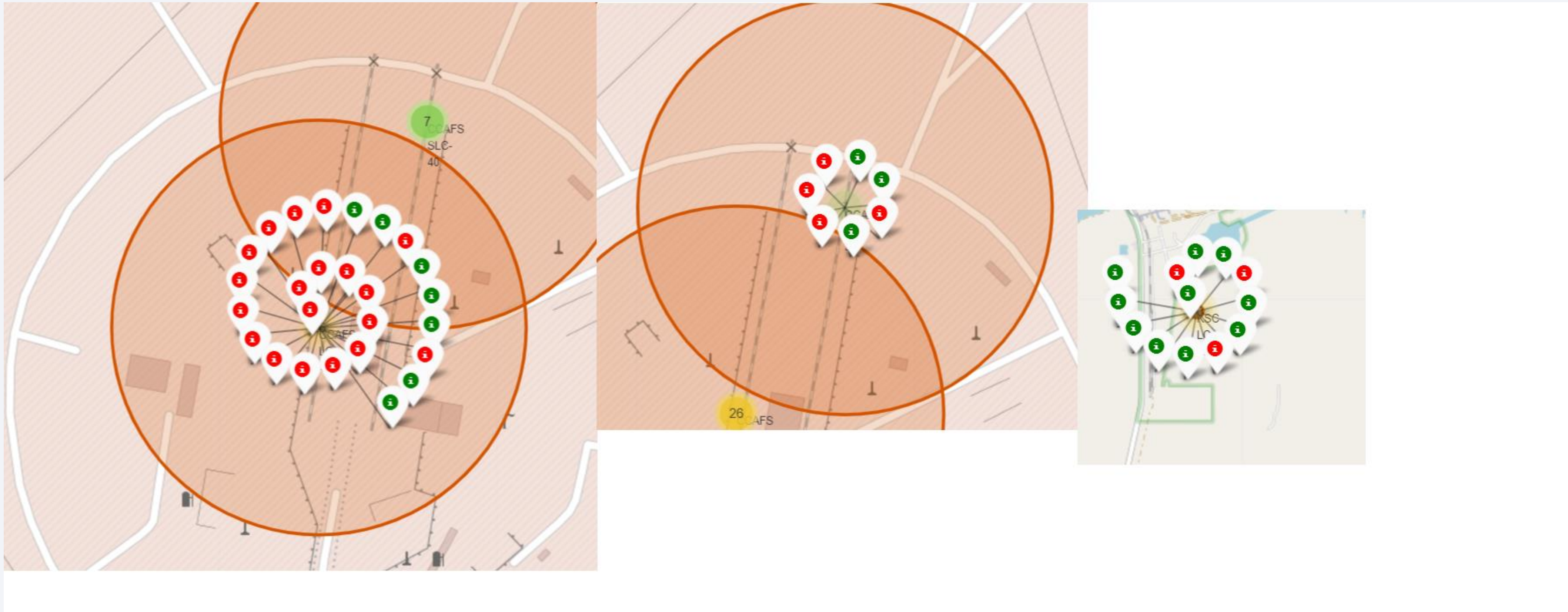
Launch Sites Proximities Analysis

Launch Sites on Folium Map



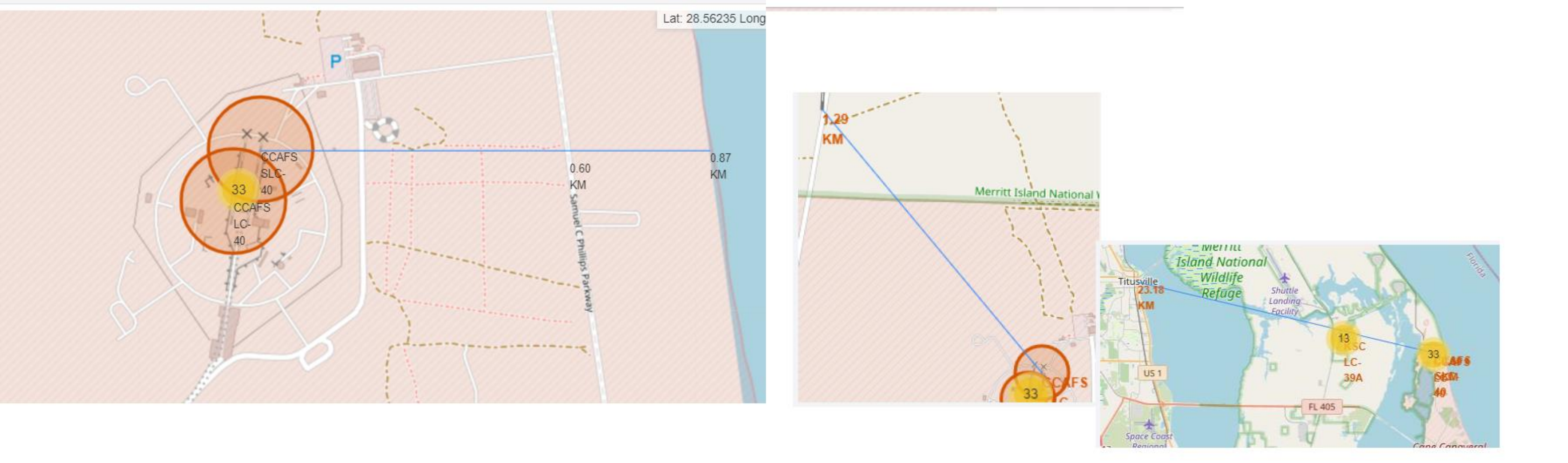
- The above screenshot shows the launch sites of SpaceX which are located on the coastlines of USA.

Color labelled markers for each site.

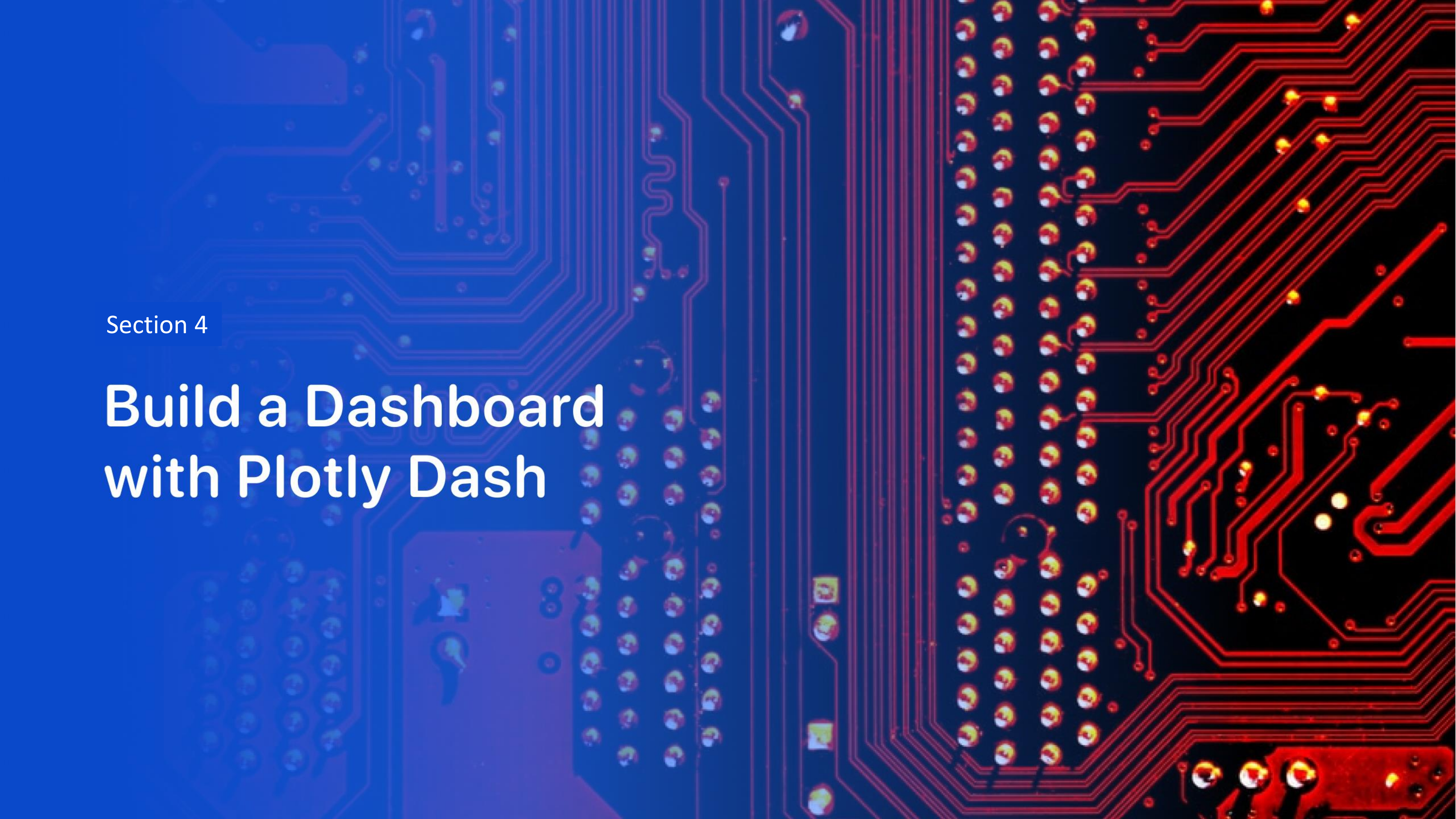


- We can see here, the green represents successful landings while the red markers represent the failed ones.
- KSC LC-39A has the highest number of successes.

Distance of CCAFS SLC-40 FROM VARIOUS PLACES



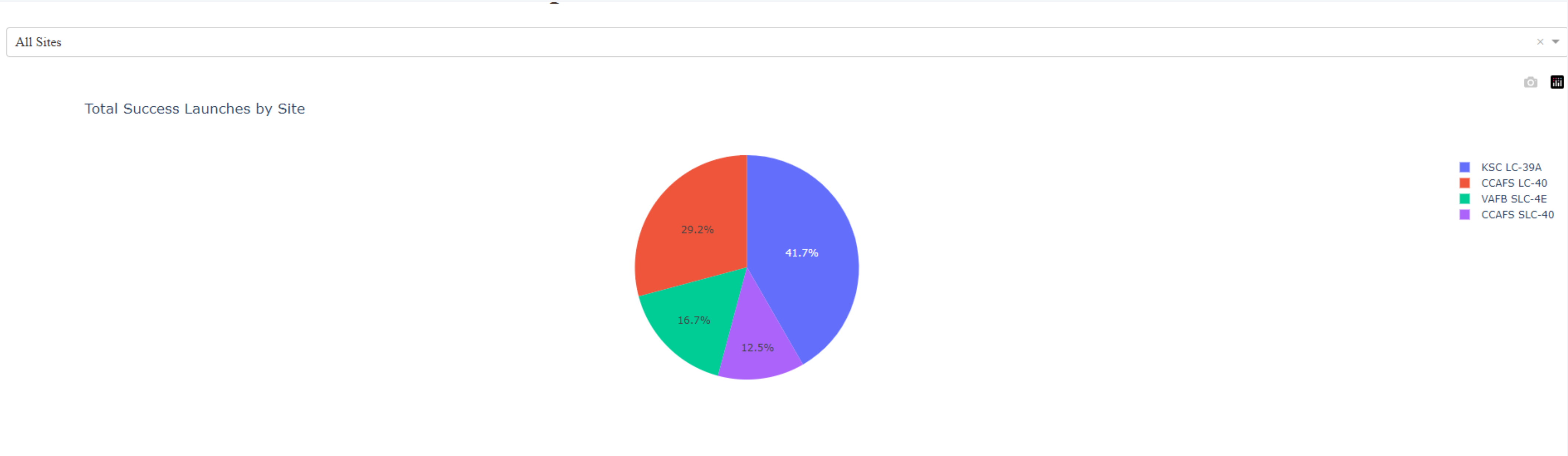
- From the above map we can conclude that CCAFS SLC-40 is near to railways, highways and coastline. It is not too far from the city.



Section 4

Build a Dashboard with Plotly Dash

Total success Launches by site

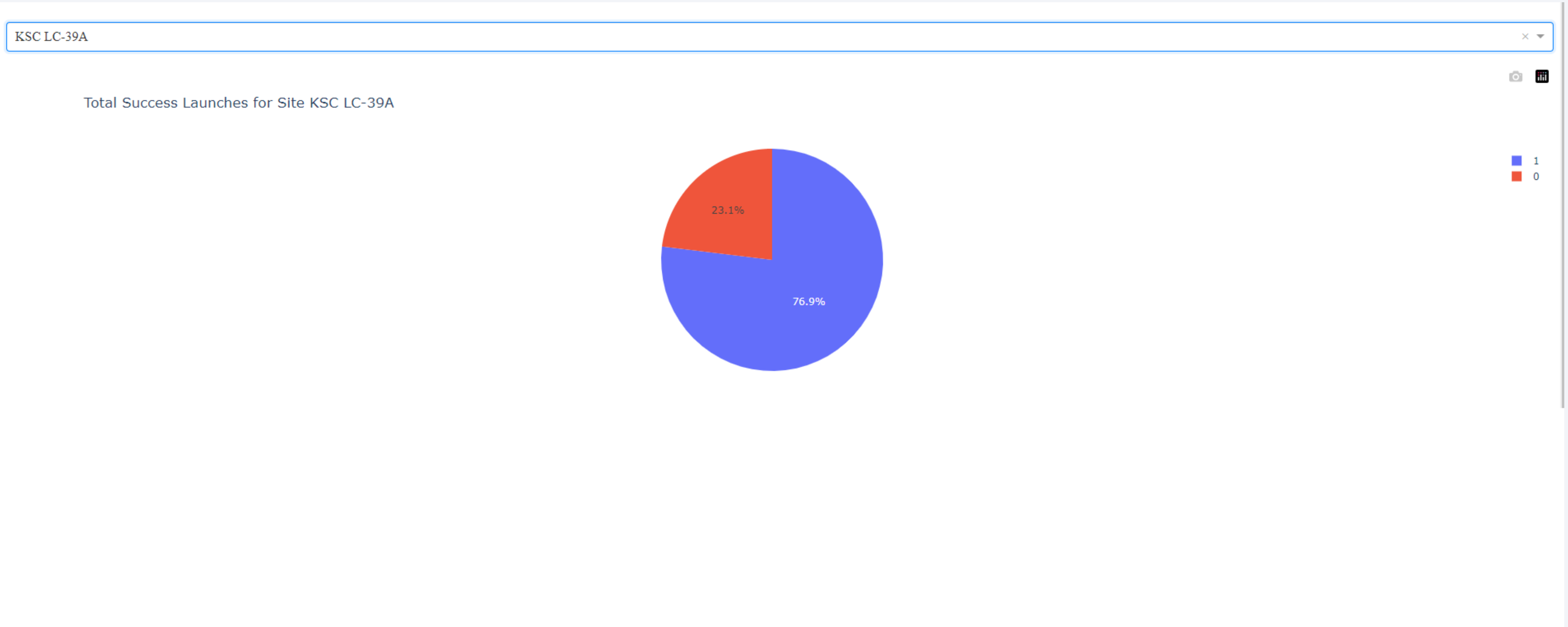


All Sites

Total Success Launches by Site

- Most successful Site - KSC LC – 39A

Success launches for the most successful site

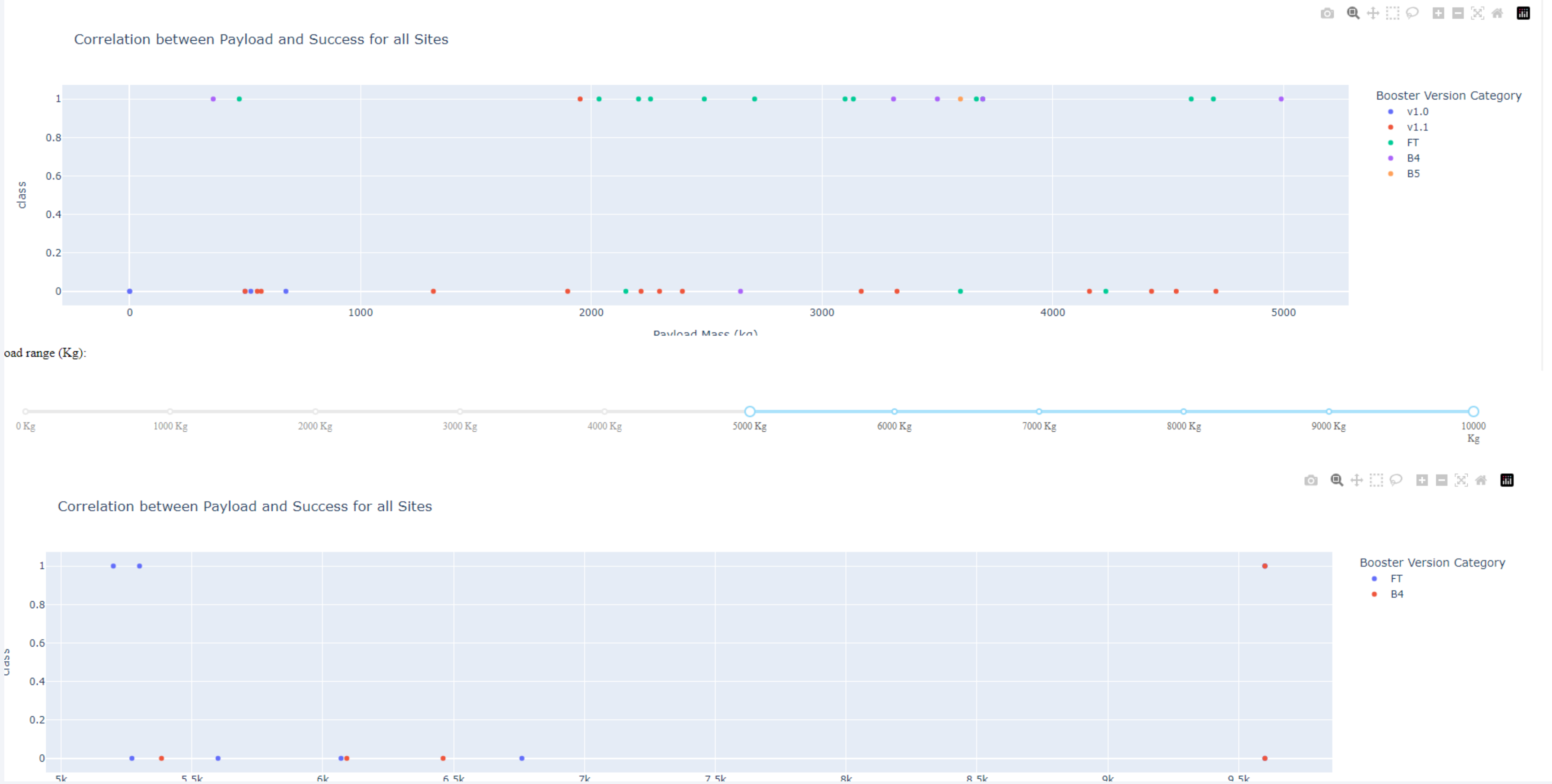


KSC LC-39A

Total Success Launches for Site KSC LC-39A

- As KSC LC 39A is the most successful site, we looked deeper in its pie chart and we found out that it has a success rate of 76.9%

Payload vs. Launch Outcome scatter plot for all sites

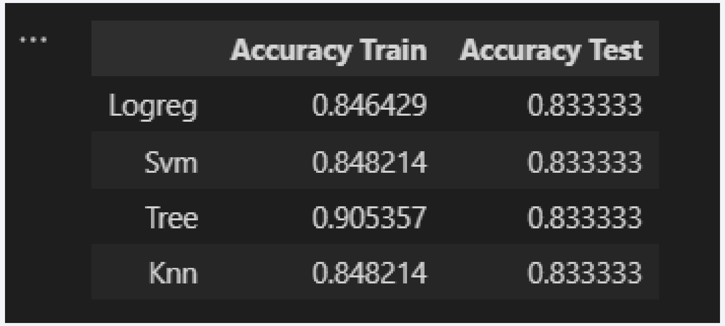


- We can see that lighter payload had more chances for success as compared to heavier payload.

Section 5

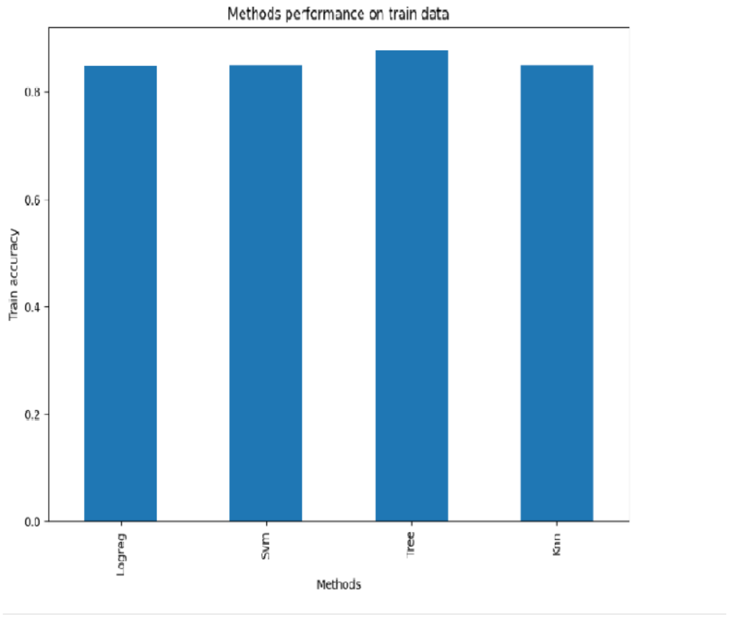
Predictive Analysis (Classification)

Classification Accuracy

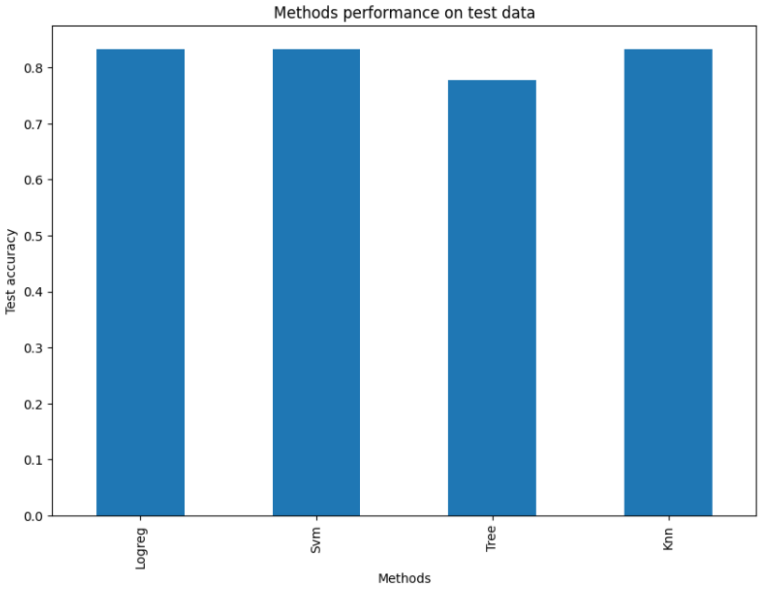


	Accuracy Train	Accuracy Test
Logreg	0.846429	0.833333
Svm	0.848214	0.833333
Tree	0.905357	0.833333
Knn	0.848214	0.833333

```
[70]: Text(0.5, 1.0, 'Methods performance on train data')
```



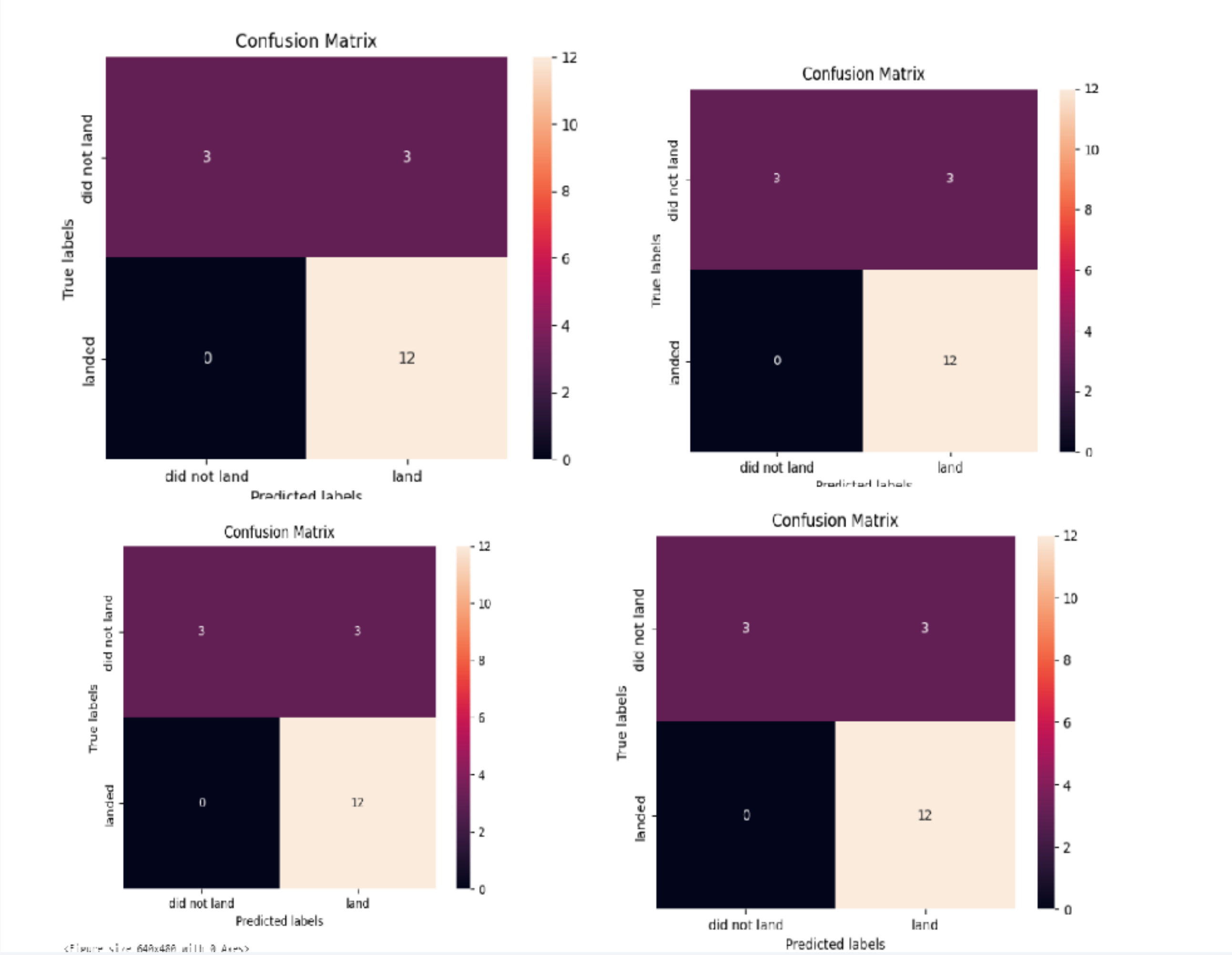
```
[71]: Text(0.5, 1.0, 'Methods performance on test data')
```



- We can see that all the models performed equally well on the test dataset. Find which model has the highest classification accuracy. In order to differentiate amongst them we would need to perform testing on more data. Decision tree model had the best accuracy on the training dataset and can be chosen as our current model.

Confusion Matrix

- All models had similar performance on the test dataset hence the confusion matrix for each of them is similar to that of others.
- A confusion matrix helps us in understanding about false positives and false negatives of a model. And all of the models for the current project, suffer from the issue of false positives.



```
<Figure size 640x480 with 0 Axes>
```

Conclusions

Through this project we could conclude that :

- The success or failure of a launch depends on multiple factors like the payload, launch site, orbit etc.
- We found out that for most of the orbits, lighter the payload the better were the chances of success.
- We also looked at the data to figure out the best site which turns out to be KSC LC-39A with the maximum number of successes.
- For the current dataset, all of the models performed equally well hence we would have to expand our dataset in order to find the best model. But if to choose, Decision Tree Algorithm would be chosen as it performed well on the training data set.

Appendix

- LINK TO GITHUB REPOSITORY : LINK

Thank you!