IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

ATEMKENG NELLY ASABA
05 September 2023

Outline

Executive Summary

- Summary of methodologies
  - Data collection using REST API
  - Data collection using web scraping
  - Data wrangling
  - EDA with SQL
  - Interactive visual analytics with Folium
  - Machine learning prediction
- Summary of all results
  - EDA results
  - Interactive analytics screenshots
  - Predictive analytics results

Introduction

- Project background and context

SpaceX advertises on its website that its Falcon 9 rocket launchers costs just 62 million dollars, this amount is way low that of the other companies in the market who provides this service at prices above 165 million dollars each. SpaceX can reuse the first stage of its launches, which is why they can provide this service at a very low rate, so if we can know if the first stage will be successful, then we can predict the cost of a launch. This information is vital to rivals in order to compete with SpaceX.

Problems to find answers

- To understand the factors that determine the landing outcome
- Determine the best conditions that would lead to a successful landing and the different variables that can affect the outcome.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data is collected using SpaceX REST API and web scraping from Wikipedia
- Perform data wrangling
  - Data was processed with one hot encoding
- 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 was collected using REST API and web scraping from Wikipedia, convert to pandas dataframe after decoding the response as Json using json_normalize().

We cleaned the data, inspect missing values and fill them as needed.

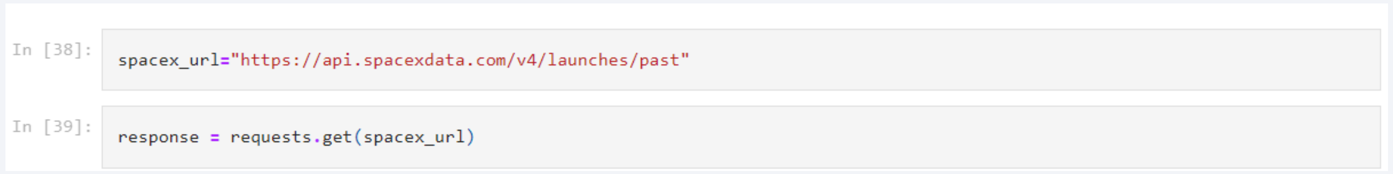
We used beautifulsoup to extract the html table records and convert to pandas dataframe for analysis.

Data Collection – SpaceX API

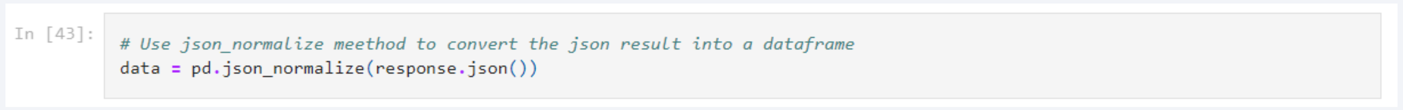
- Data collection with SpaceX REST API calls, json normalization and data cleaning and filling of missing values

- GitHub URL of the completed SpaceX API calls notebook

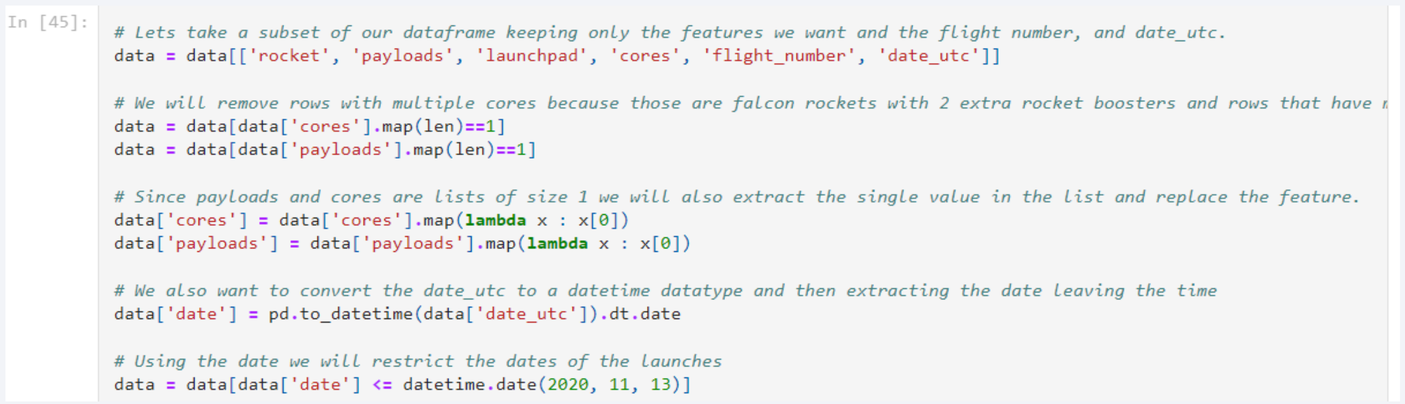
https://github.com/nellyasaba/DataScienceCapstone/blob/main/jupyter-labs-spacex-data-collection-api%20(2).ipynb



```
In [38]: spacex_url="https://api.spacexdata.com/v4/launches/past"

In [39]: response = requests.get(spacex_url)
```



```
In [43]: # Use json_normalize meethod to convert the json result into a dataframe
         data = pd.json_normalize(response.json())
```



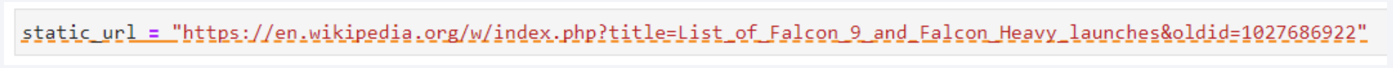
```
In [45]: # Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
         data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

         # We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and rows that have
         data = data[data['cores'].map(len)==1]
         data = data[data['payloads'].map(len)==1]

         # Since payloads and cores are lists of size 1 we will also extract the single value in the list and replace the feature.
         data['cores'] = data['cores'].map(lambda x : x[0])
         data['payloads'] = data['payloads'].map(lambda x : x[0])

         # We also want to convert the date_utc to a datetime datatype and then extracting the date leaving the time
         data['date'] = pd.to_datetime(data['date_utc']).dt.date

         # Using the date we will restrict the dates of the launches
         data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

Data Collection - Scraping

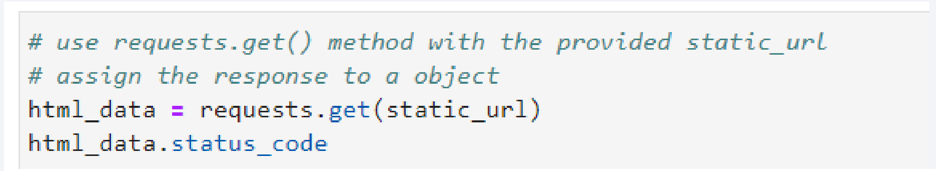
- Using web scraping on the Wikipedia page, then apply beautifulsoup from the response, there after parsing it into a pandas dataframe

GitHub URL of the completed web scraping notebook

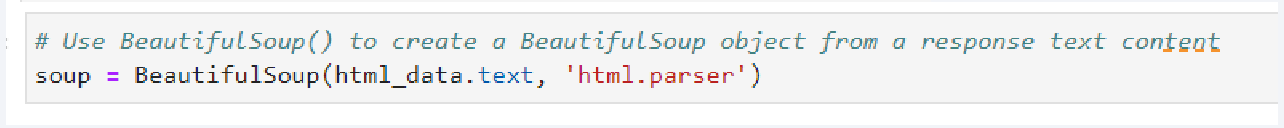
https://github.com/nellyasaba/DataScienceCapstone/blob/main/jupyter-labs-webscraping.ipynb



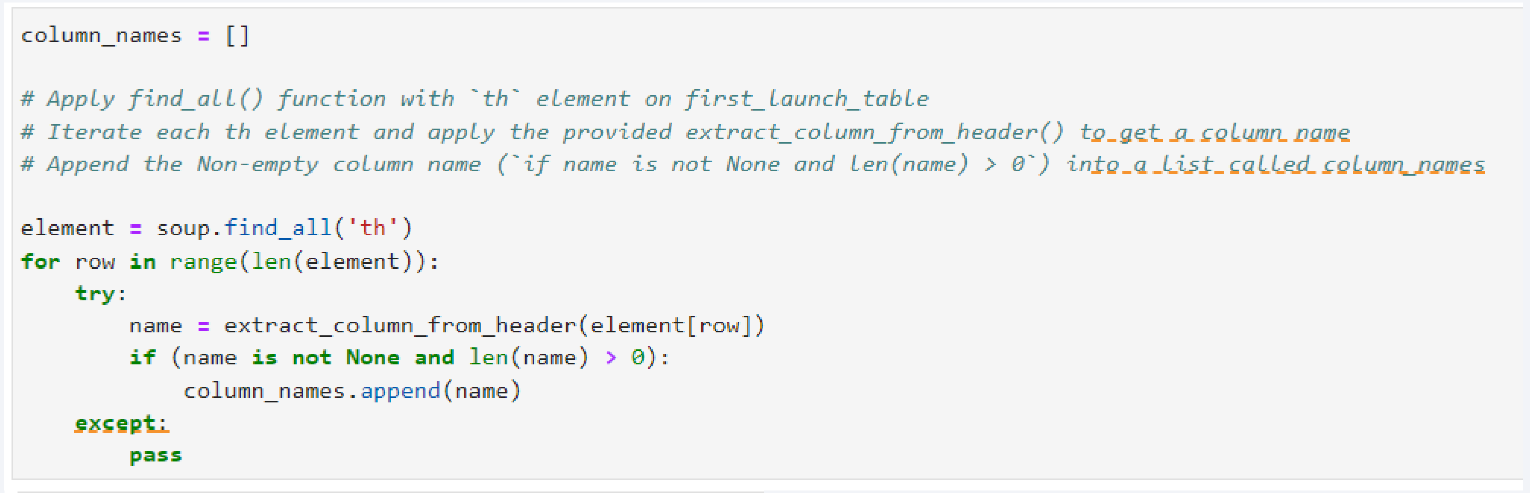
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```



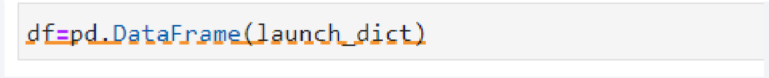
```
# use requests.get() method with the provided static_url
# assign the response to a object
html_data = requests.get(static_url)
html_data.status_code
```



```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(html_data.text, 'html.parser')
```



```
column_names = []

# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names

element = soup.find_all('th')
for row in range(len(element)):
    try:
        name = extract_column_from_header(element[row])
        if (name is not None and len(name) > 0):
            column_names.append(name)
    except:
        pass
```



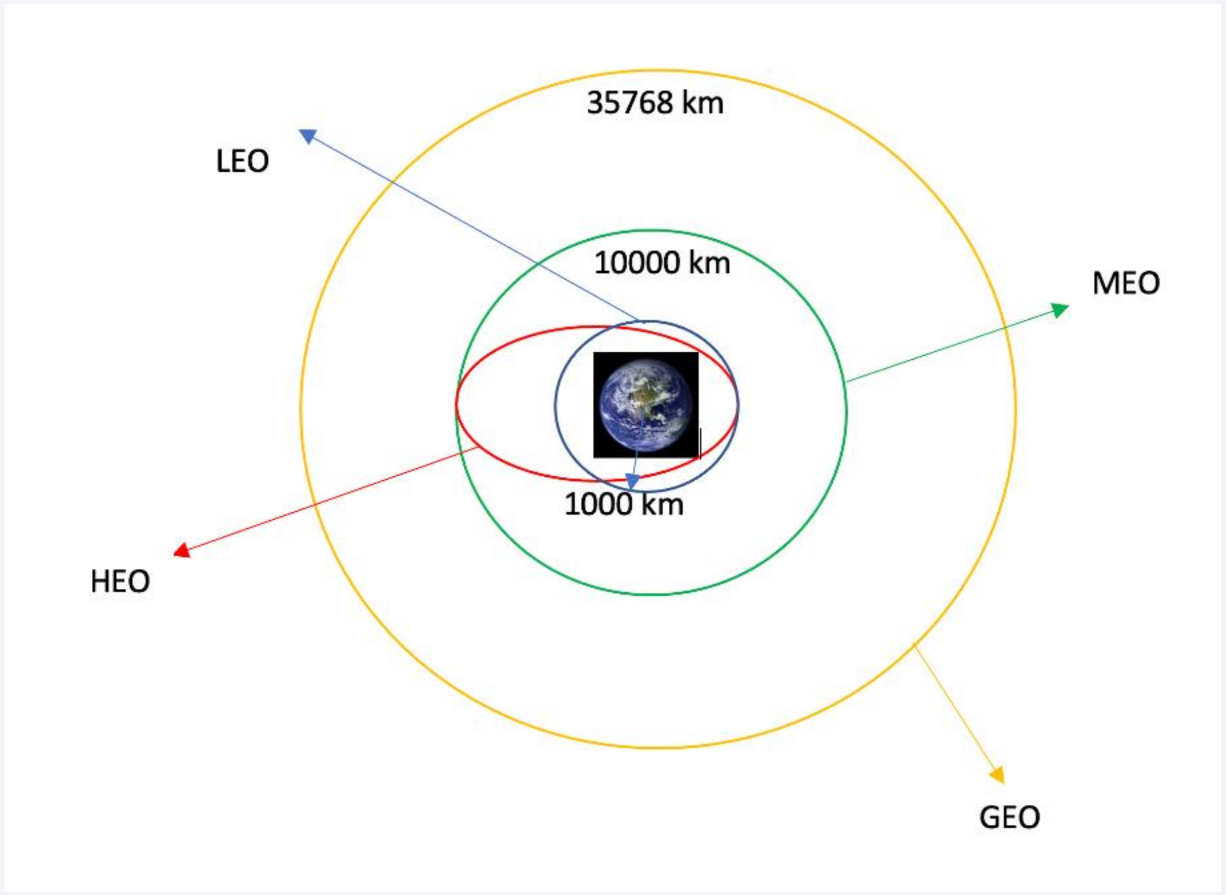
```
df=pd.DataFrame(launch_dict)
```

Data Wrangling

- We cleaned and arranged the data properly to enable easy EDA analysis on the data.
- We determined the number of launches on every site and also mission occurrence outcome as per obit.
- We then create the outcome of landing label and exported the result to csv.

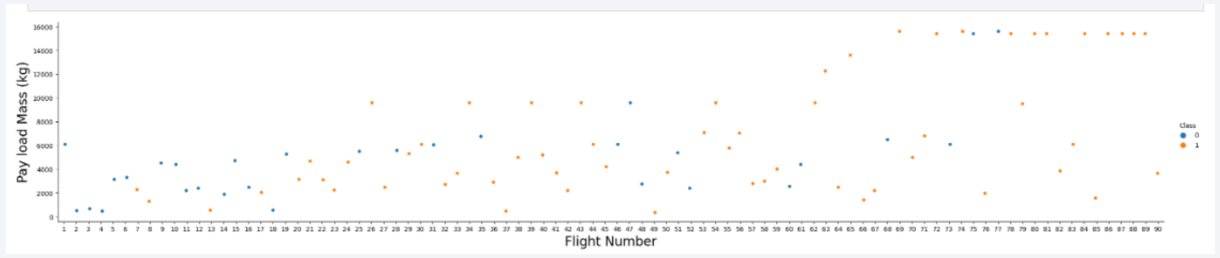
GitHub URL of completed data wrangling notebooks

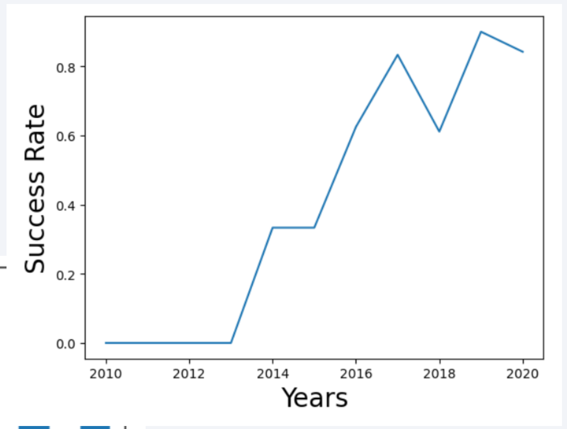
https://github.com/nellyasaba/DataScienceCapstone/blob/main/labs-jupyter-spacex-data_wrangling_jupyterlite.jupyterlite%20(2).ipynb

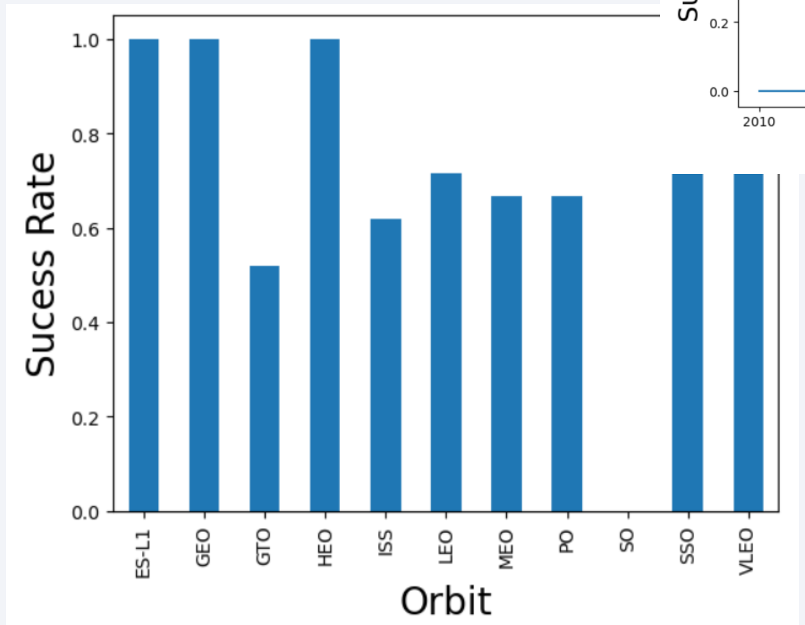


EDA with Data Visualization

- We plotted scatter plots to see how FlightNumber and PayloadMass would affect launch outcome, also FlightNumber with LaunchSite, Payload with LaunchSite, FlightNumber with Orbit type, Payload with Orbit type all scatter plots.
- We then used a bar chart to visualize relationship between success rate of each orbit type and we also plotted a line chart to visualize the launch success yearly trend.
- GitHub URL of completed EDA with data visualization notebook https://github.com/nellyasaba/DataScienceCapstone/blob/main/jupyter-labs-eda-dataviz.ipynb.jupyterlite%20(2).ipynb







EDA with SQL

- We firstly loaded the SQL extension and established connection with the database
- We created a table removing blank rows.
- We displayed the names of unique launch sites.
- We displayed 5 records of launch sites beginning with string 'CCA'
- Displayed the total payload mass carried by boosters launched by NASA(CRS)
- Displayed the average payload mass carried by booster version F9 v1.1
- We listed the date the first successfuly landing outcome was acheived in ground pad
- We listed names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- We listed the total number of successful and failure mission outcomes
- We listed the names of the booster_versions which have carried the maximum payload mass
- We listed the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015
- We then ranked the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.
- https://github.com/nellyasaba/DataScienceCapstone/blob/main/jupyter-labs-eda-sql-coursera_sqllite%20(1).ipynb

Build an Interactive Map with Folium

- We used circle markers to mark launch sites on the folium map using their latitude and longitude coordinates.
- We also created markers for all launch records, assigning a green marker to successful launch of class=1 and a red marker for unsuccesssful records with class=0
- We added a mouseposition on the map to easily know the coordinates while exploring the map and finding coordinates of any points of interest such is railroad, highway, and cities proximity.
- GitHub URL of completed interactive map with Folium map

https://github.com/nellyasaba/DataScienceCapstone/blob/main/lab_jupyter_launch_site_location.jupyterlite%20(1).ipynb

Build a Dashboard with Plotly Dash

- The dashboard has a pie chart with all launch site being presented and its being updated by a dropdown menu of the various launch sites and the pie chart show the success rates of the different launch sites.
- There is also a slider to select the payload range and a scatter plot to show the correlation between the payload and launch success.
- GitHub URL of completed Plotly Dash lab

https://github.com/nellyasaba/DataScienceCapstone/blob/main/app.py

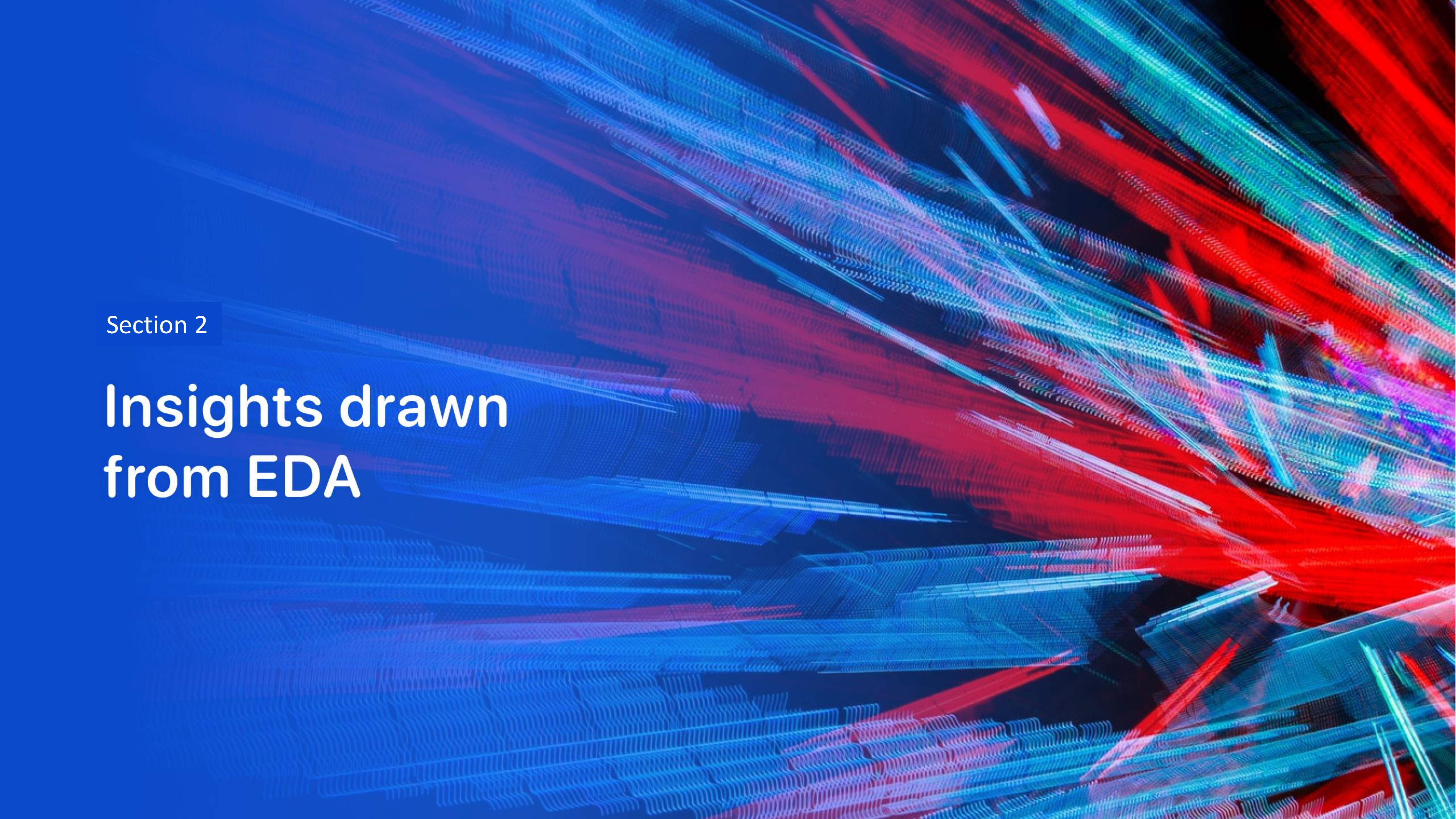
Predictive Analysis (Classification)

- The data is being loaded using numpy and pandas, there after its transformed then split into training and testing sets.
- The different models are used to evaluate the data and get the accuracy score.
- Tune different hyperparameters of all the models
- Implore feature engineering
- Find the best performing model based on the best accuracy score.

https://github.com/nellyasaba/DataScienceCapstone/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite%20(2).ipynb

Results

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

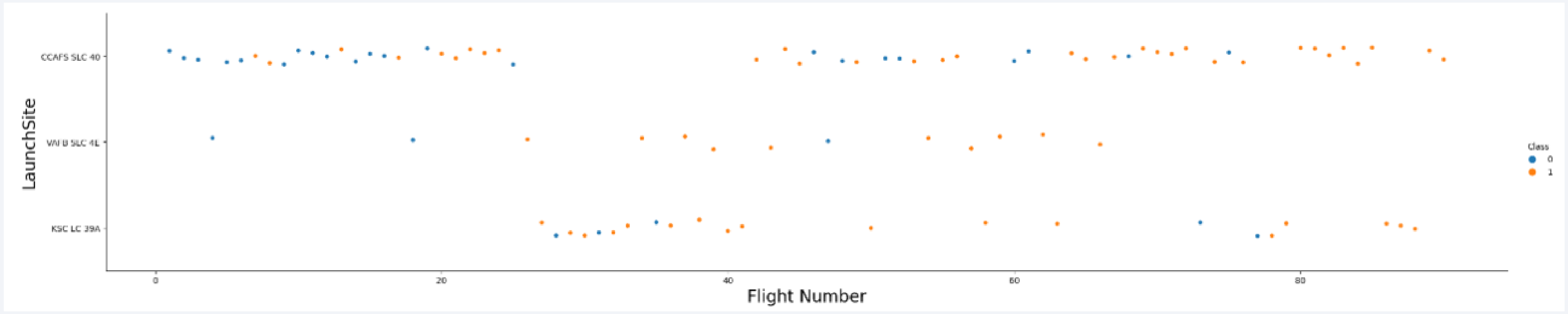


Section 2

Insights drawn from EDA

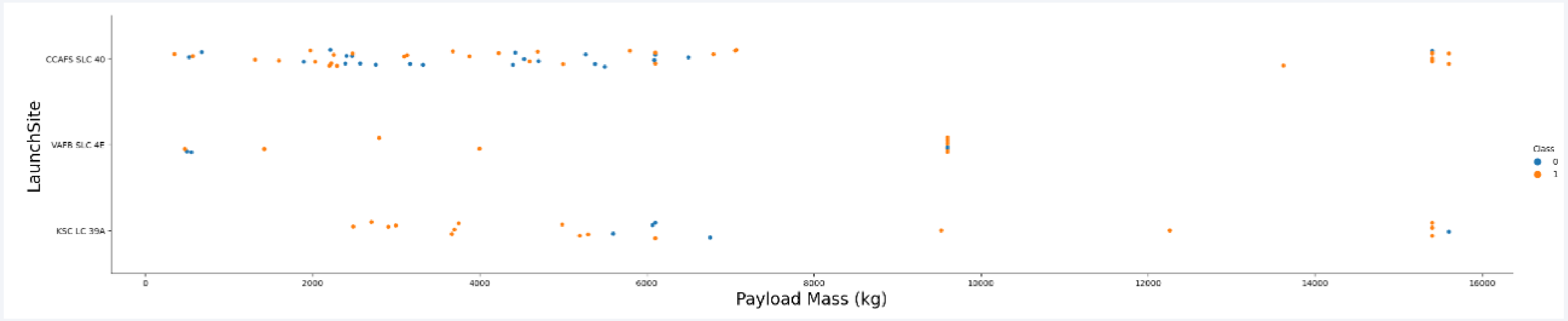
Flight Number vs. Launch Site

- The chart shows that the higher the flight number, the more successful launches are attained.



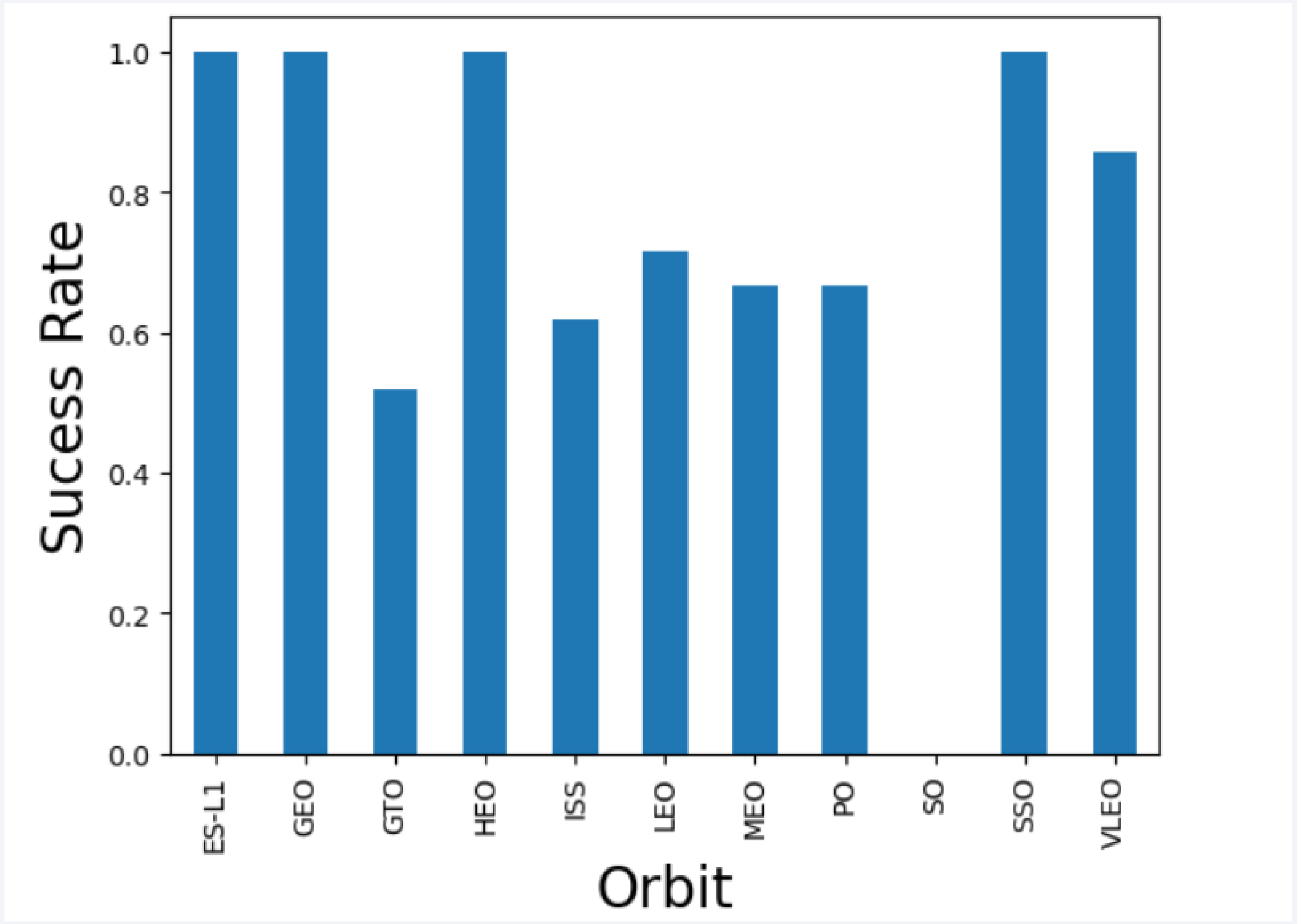
Payload vs. Launch Site

- It can be seen that VAFB-SLC launchsite has no launches for payload mass greater than 10000.



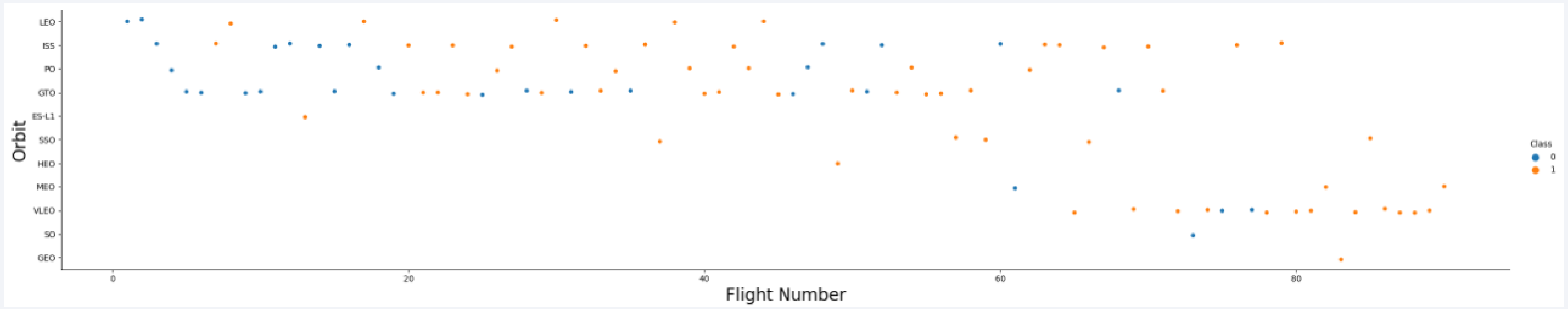
Success Rate vs. Orbit Type

- This chart shows that there is a relationship between success rate and orbit type.
- It can be seen that some orbits have 100% success rate, while others like “SO” have a 0% success rate.



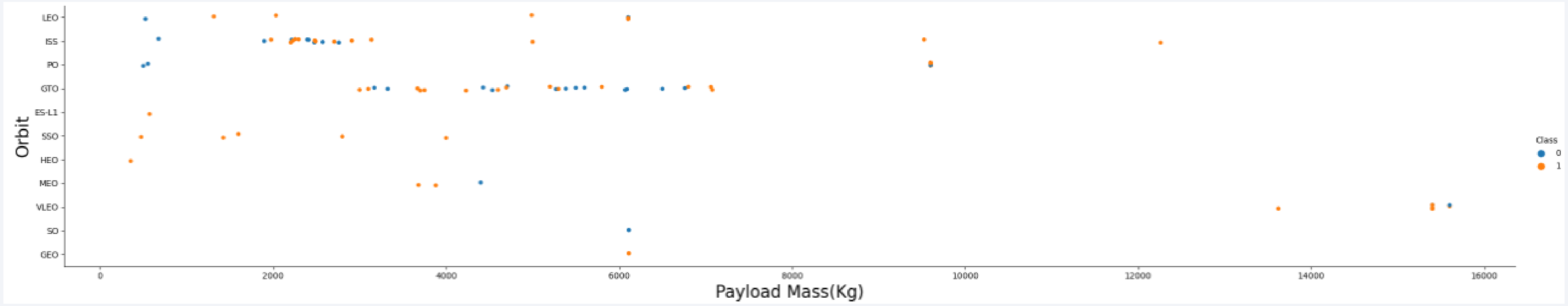
Flight Number vs. Orbit Type

- It can be seen that one orbit type, the LEO has a higher success rate that relates to higher flight number, but the others appear unrelated.



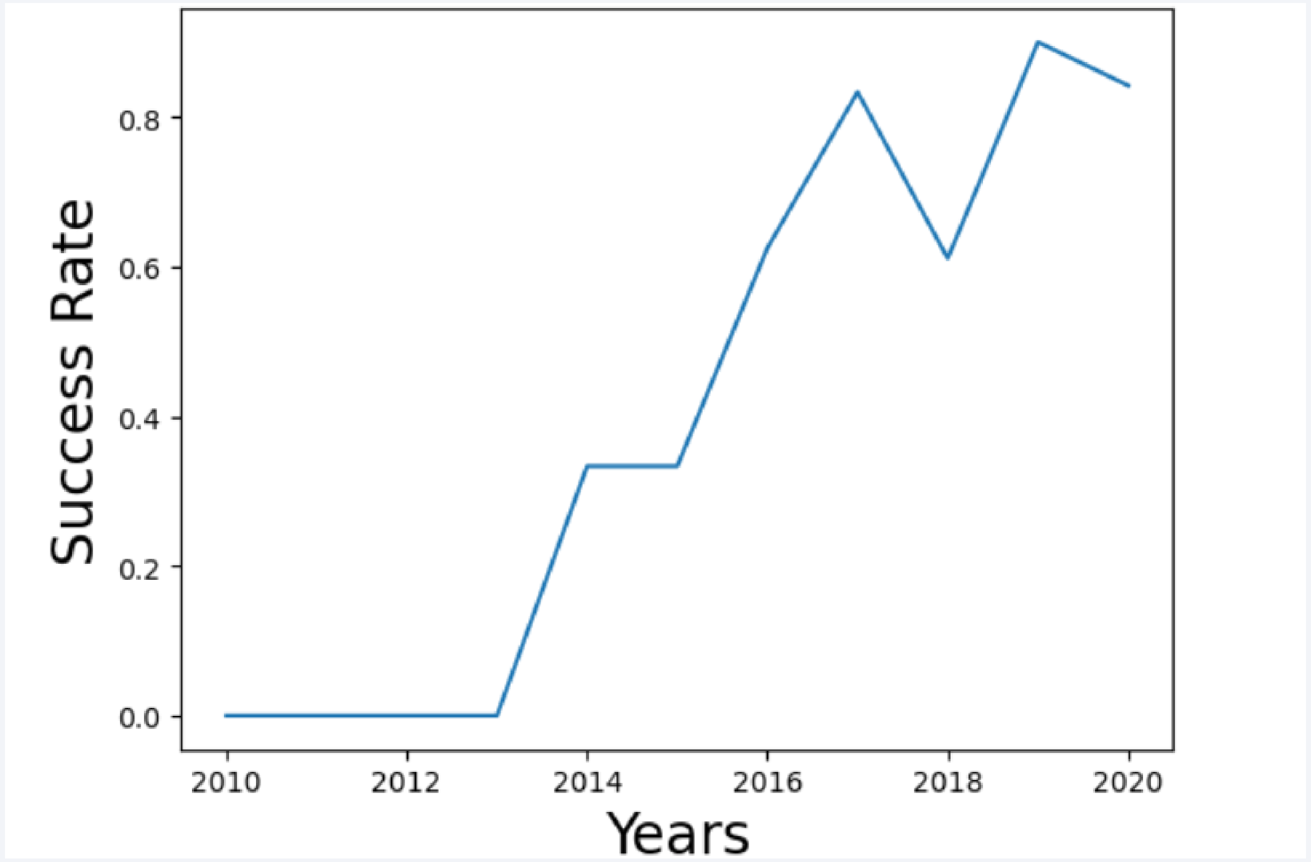
Payload vs. Orbit Type

- For heavy payloads, success rate is high for LEO, ISS and PO while the others seem mixt up.



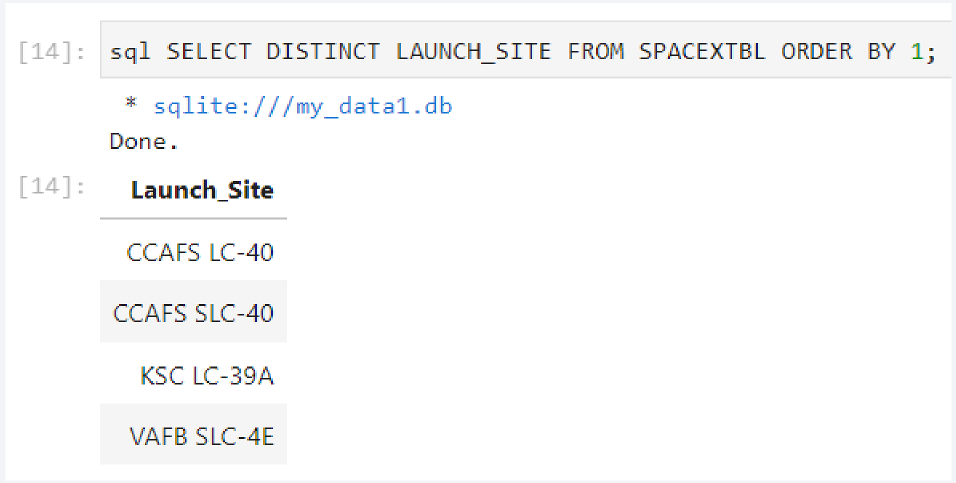
Launch Success Yearly Trend

- It is evident in this line chart that the success rate has been increasing since 2013, there was a little down trend in 2017 but overall its been positive and keeps getting better as the years increase.



All Launch Site Names

We use the keyword DISTINCT to get the unique name of launch sites.

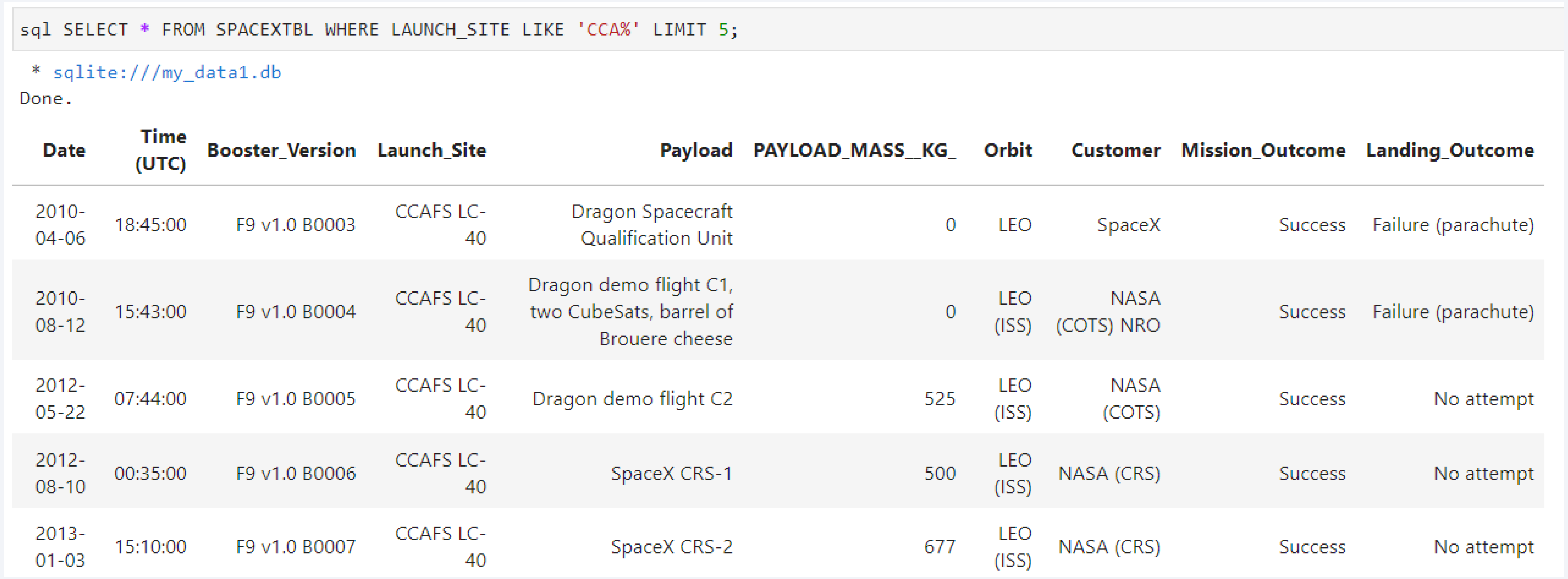


```
[14]: sql SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL ORDER BY 1;

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

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

Launch Site Names Begin with 'CCA'

- Here we use the LIKE ‘CCA%’ to get the right results and limit by 5



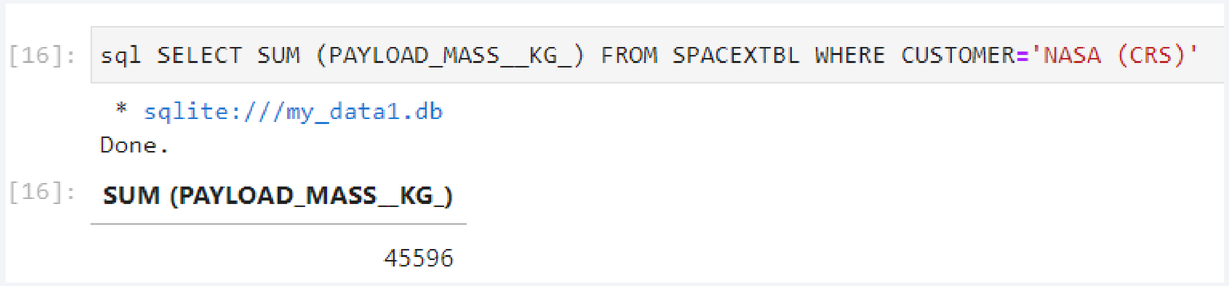
```
sql SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
```

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

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

Total Payload Mass

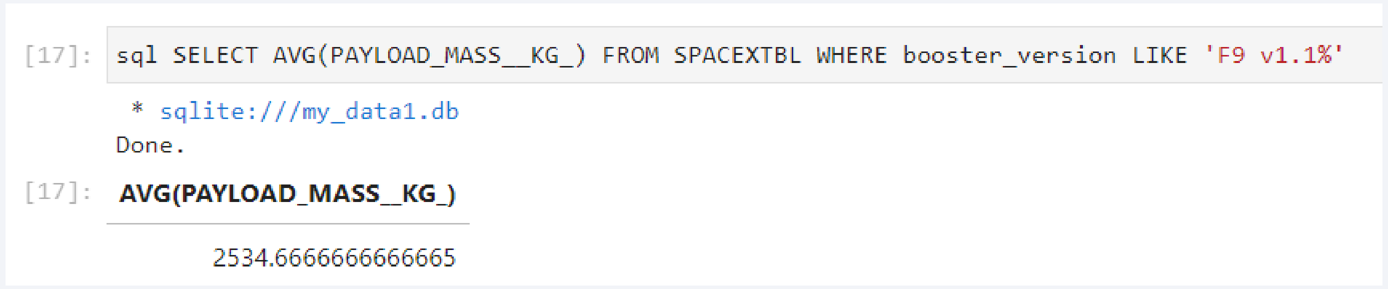
We use the “SUM” command as can be seen below to get the total payload mass.



```
[16]: sql SELECT SUM (PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE CUSTOMER='NASA (CRS)'

       * sqlite:///my_data1.db
      Done.

[16]: SUM (PAYLOAD_MASS__KG_)
                        45596
```

Average Payload Mass by F9 v1.1

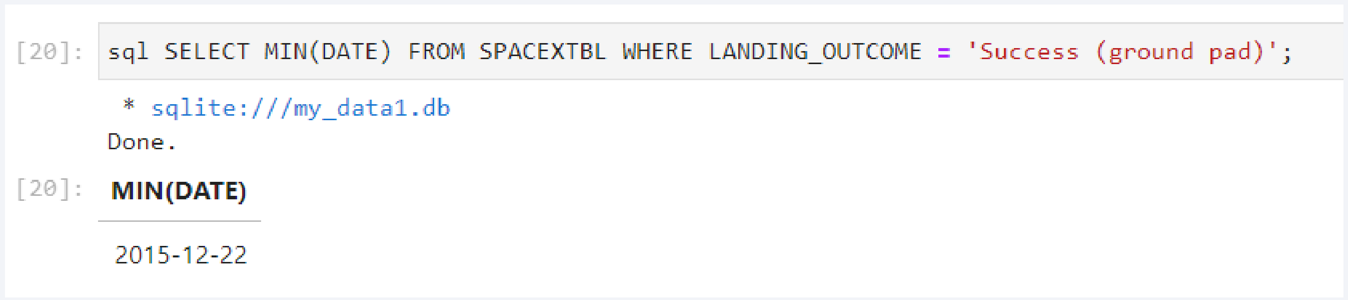
Here we use the “AVG” command with the “LIKE” option to get the required result.



```
[17]: sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE booster_version LIKE 'F9 v1.1%'

       * sqlite:///my_data1.db
      Done.

[17]: AVG(PAYLOAD_MASS__KG_)
              2534.6666666666665
```

First Successful Ground Landing Date

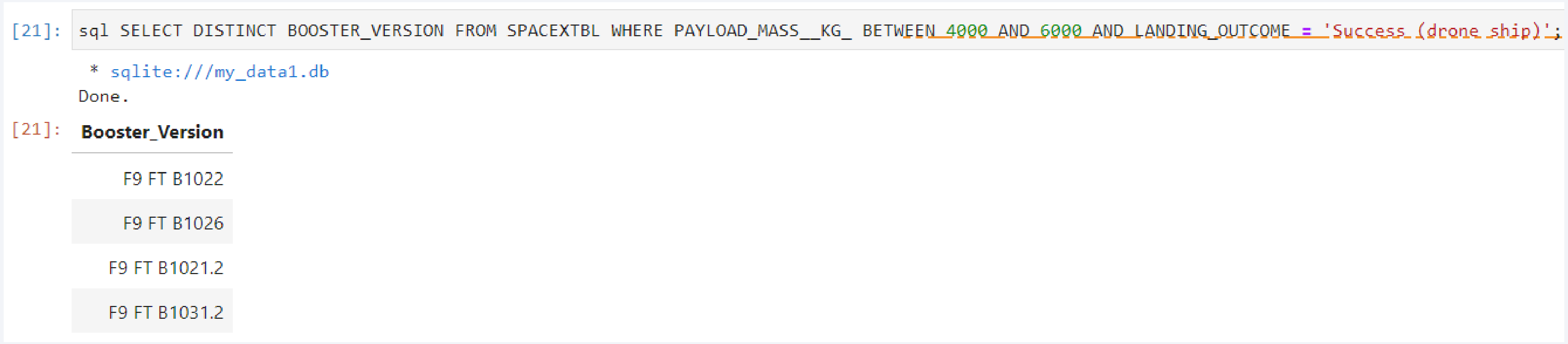
- Here we use the “MIN” command to get the result from the date column.



```
[20]: sql SELECT MIN(DATE) FROM SPACEXTBL WHERE LANDING_OUTCOME = 'Success (ground pad)';

 * sqlite:///my_data1.db
Done.

[20]: MIN(DATE)

      2015-12-22
```

Successful Drone Ship Landing with Payload between 4000 and 6000

- Here we used the “DISTINCT” and “BETWEEN” functions to get our result.



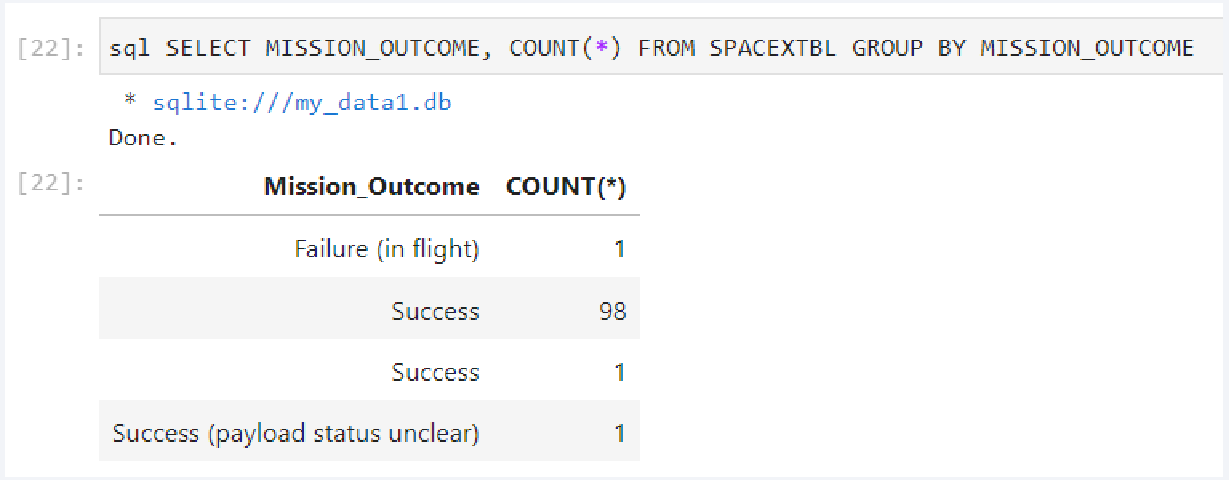
```
[21]: sql SELECT DISTINCT BOOSTER_VERSION FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000 AND LANDING_OUTCOME = 'Success (drone ship)';

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

[21]:

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

Total Number of Successful and Failure Mission Outcomes

- Here we used the “GROUP BY” on the mission outcome column to get our result.



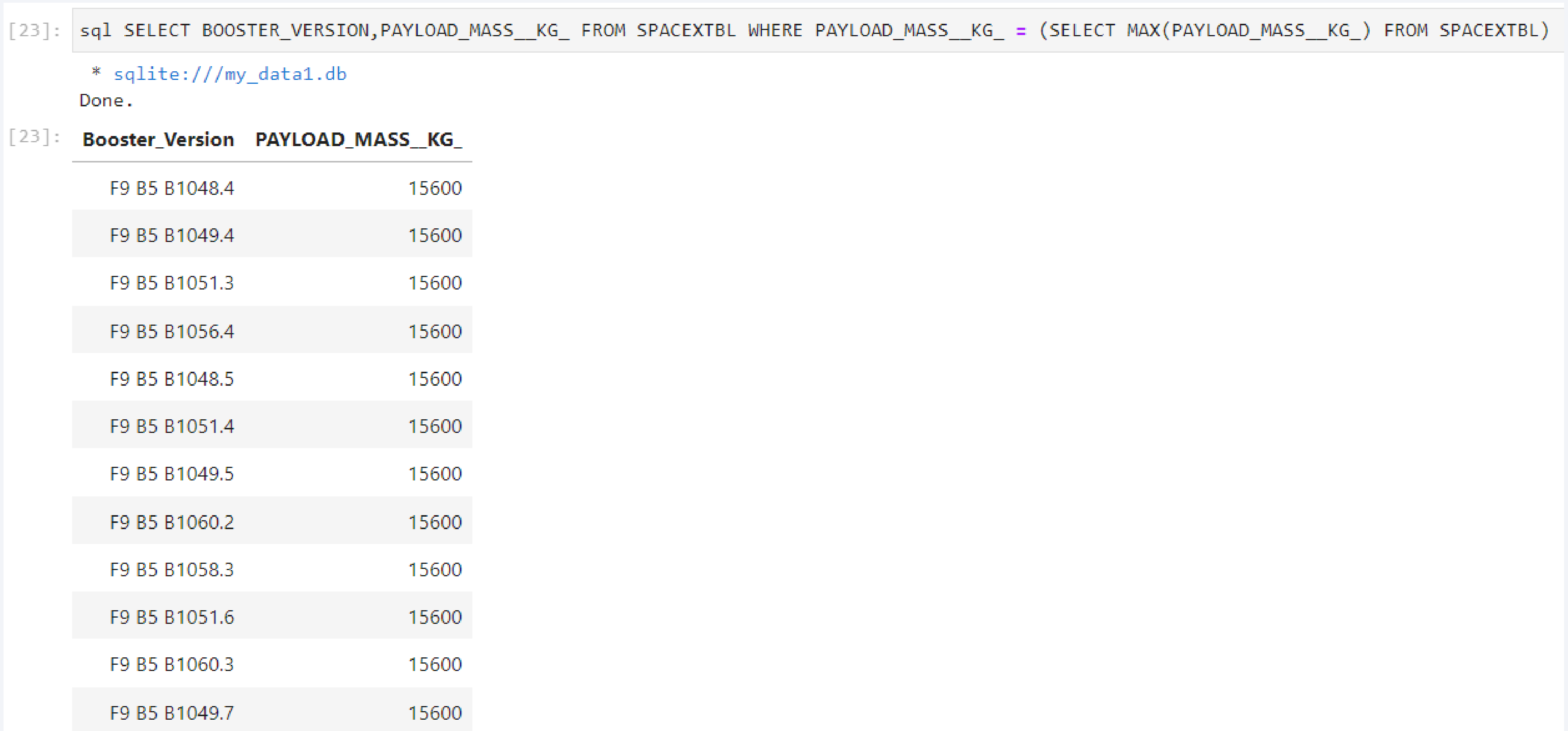
```
[22]: sql SELECT MISSION_OUTCOME, COUNT(*) FROM SPACEXTBL GROUP BY MISSION_OUTCOME
 * sqlite:///my_data1.db
Done.
```

[22]:

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

Boosters Carried Maximum Payload

- Here we use the "SELECT MAX" function to get our result.



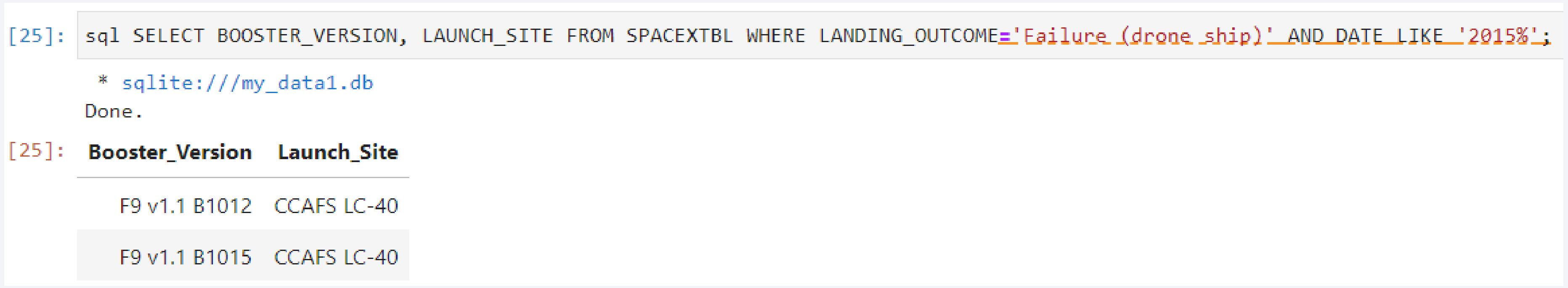
```
[23]: sql SELECT BOOSTER_VERSION,PAYLOAD_MASS__KG_ FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL)

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

[23]:

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

- Here we use the Failure on LANDING_OUTCOME and precise 2015% date like function.



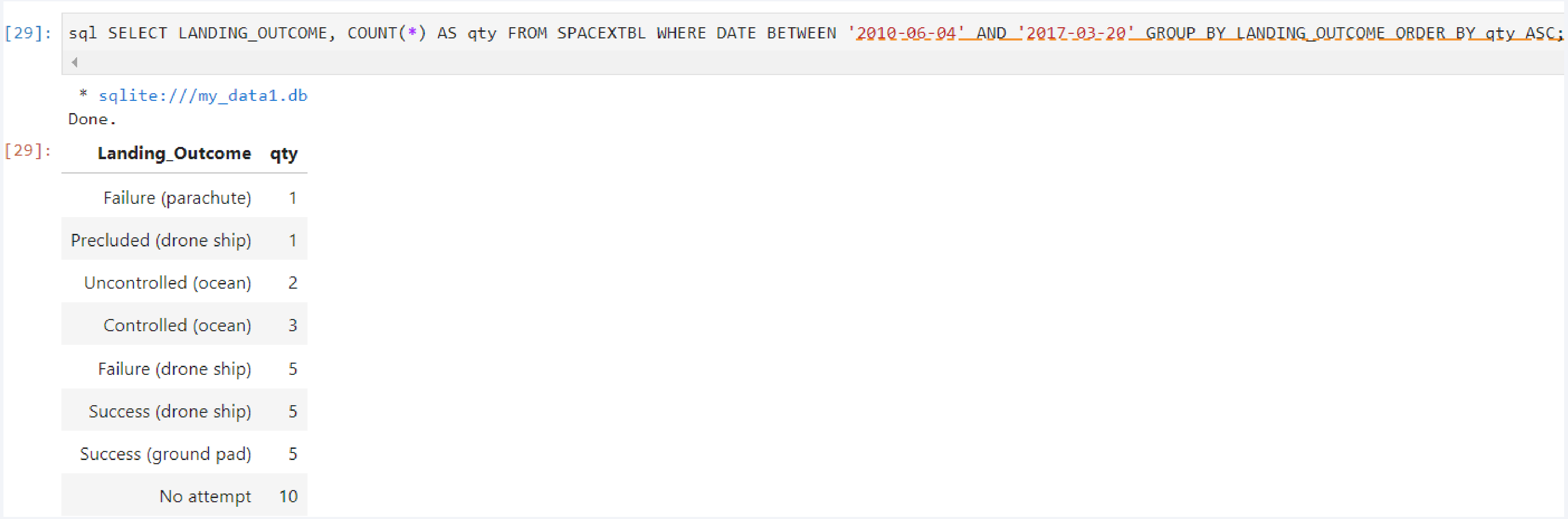
```
[25]: sql SELECT BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WHERE LANDING_OUTCOME='Failure (drone ship)' AND DATE LIKE '2015%';

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

[25]:

Booster_Version	Launch_Site
F9 v1.1 B1012	CCAFS LC-40
F9 v1.1 B1015	CCAFS LC-40

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

Here we use the "BETWEEN" on the date column and group by LANDING_OUTCOME



```
[29]: sql SELECT LANDING_OUTCOME, COUNT(*) AS qty FROM SPACEXTBL WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY LANDING_OUTCOME ORDER BY qty ASC;

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

[29]:

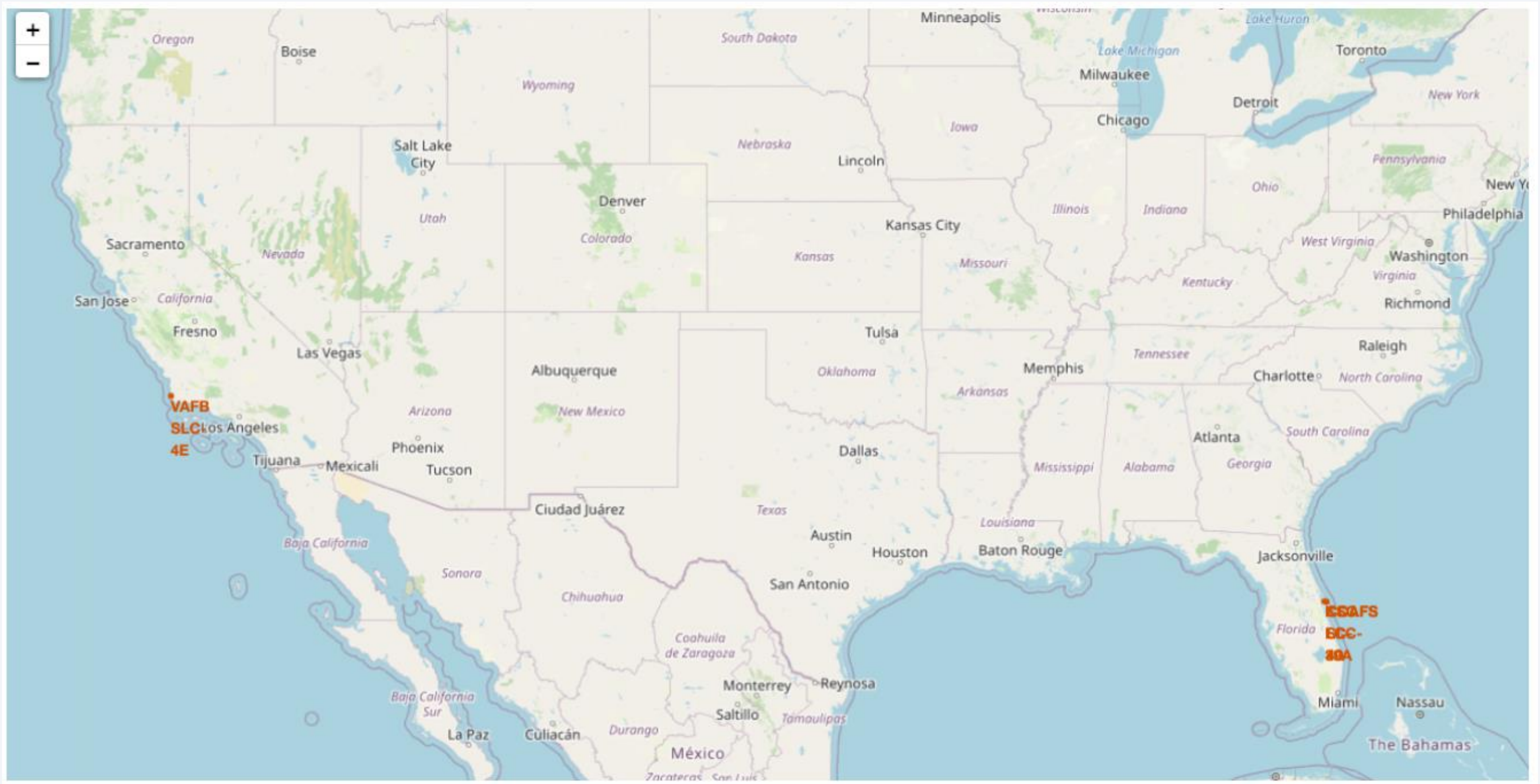
Landing_Outcome	qty
Failure (parachute)	1
Precluded (drone ship)	1
Uncontrolled (ocean)	2
Controlled (ocean)	3
Failure (drone ship)	5
Success (drone ship)	5
Success (ground pad)	5
No attempt	10

Section 3

Launch Sites Proximities Analysis

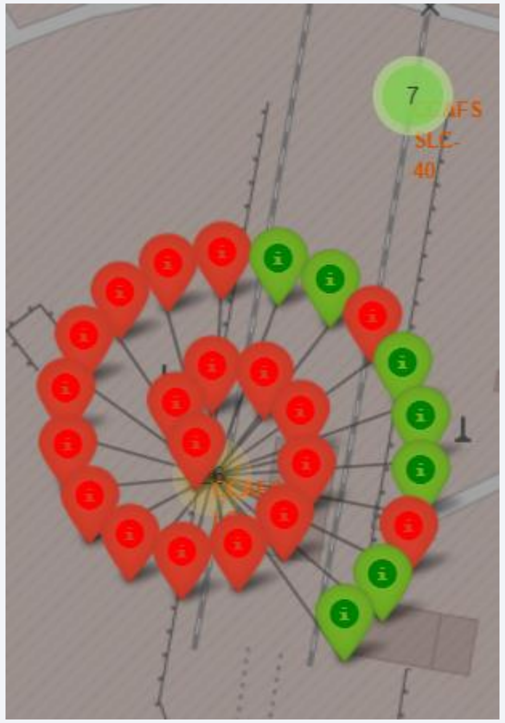
Location of all Launch Sites

- The locations are all in the USA

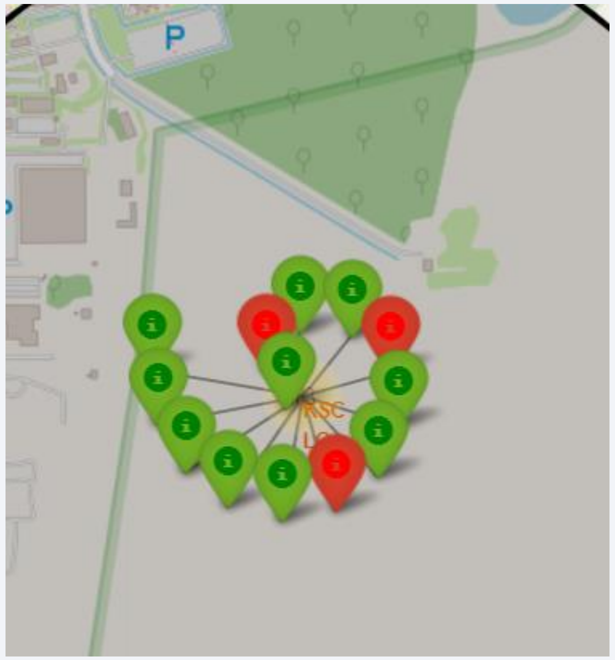


Map with color-label of Launch outcomes

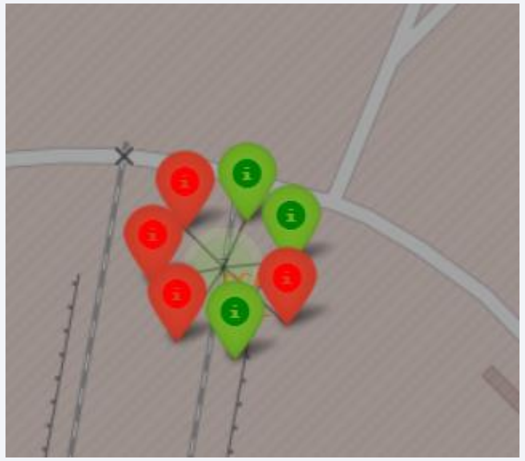
- The green marks are successful launches while red and failed.



CCAFS LC-40



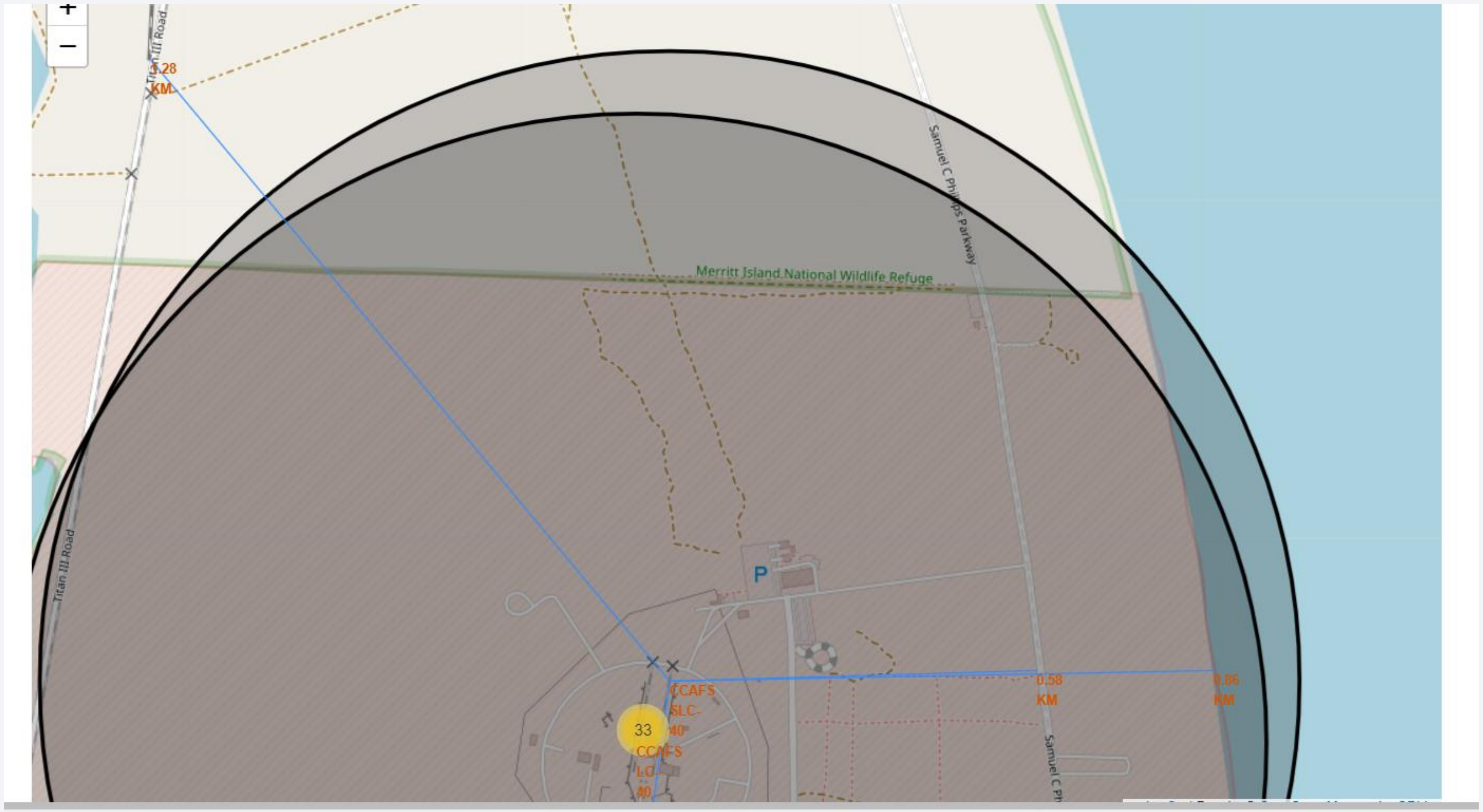
KSC LC-39A



CCAFS SLC-40

Launch site proximities to Landmarks

- We can see that the site is in proximity to coastline, but not with highways, railways and cities.

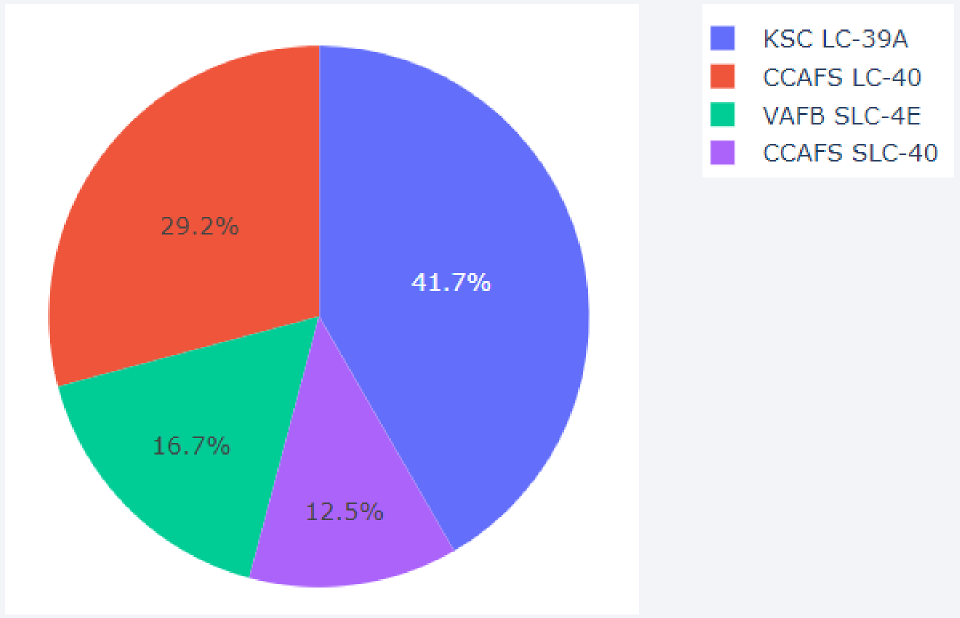


Section 4

Build a Dashboard with Plotly Dash

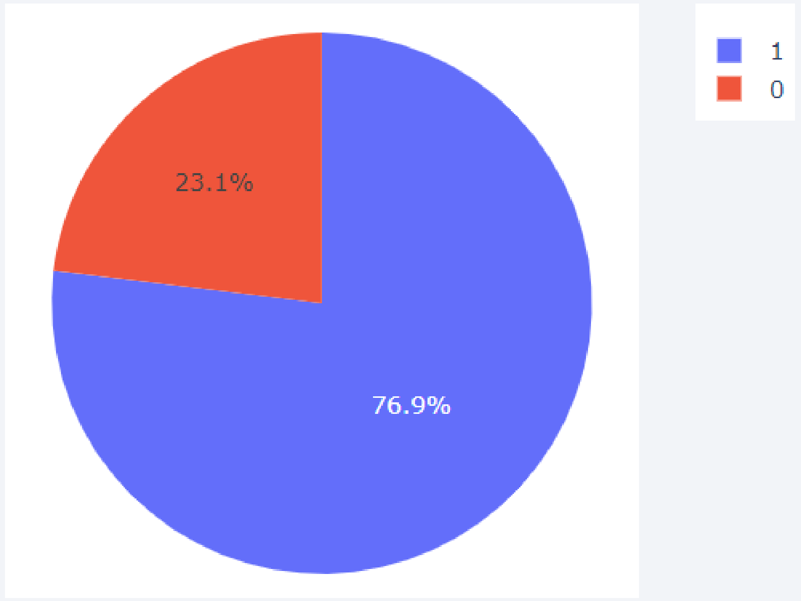
Success Count on all Launch Sites

- We can see that KSC LC-39A has the highest success rate, while CCAFS SLC-40 has the lowest success rate



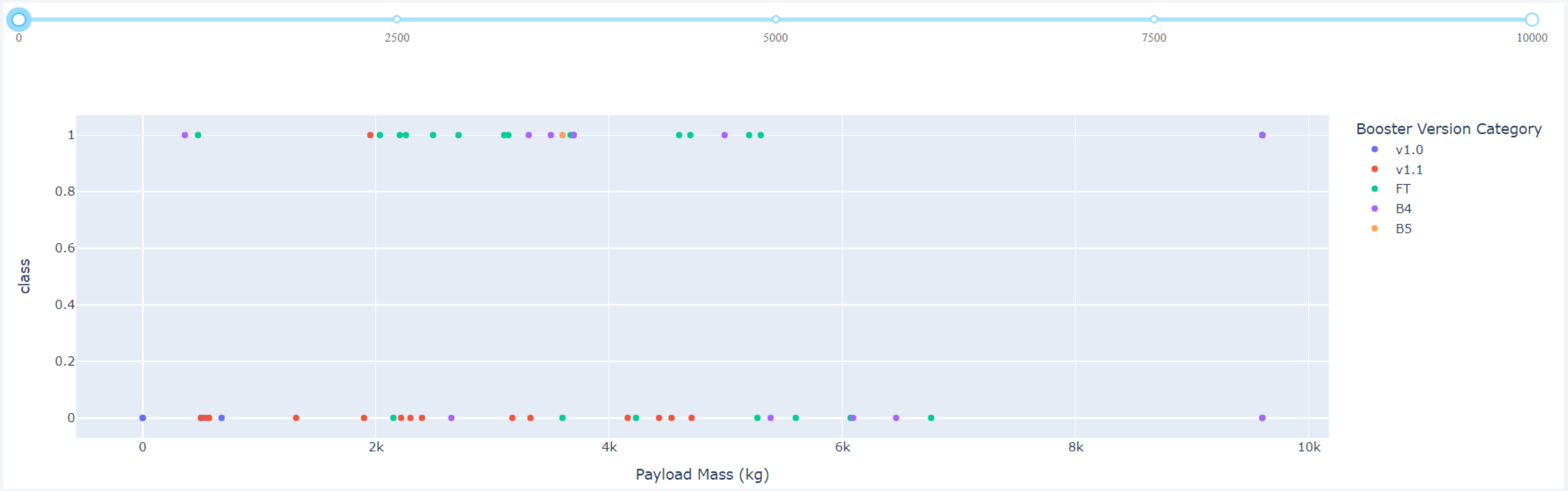
Launch Site with the highest success ratio

- We can see that KSC LC-39A has a success ratio of 76.9%



Payload vs. Launch Outcome scatter plot

- It is seen that payloads of 2000-4000 have the highest launch success rates while payloads of 6000-8000 have the lowest success rate.
- Its also seen that booster version FT has the highest success rate.

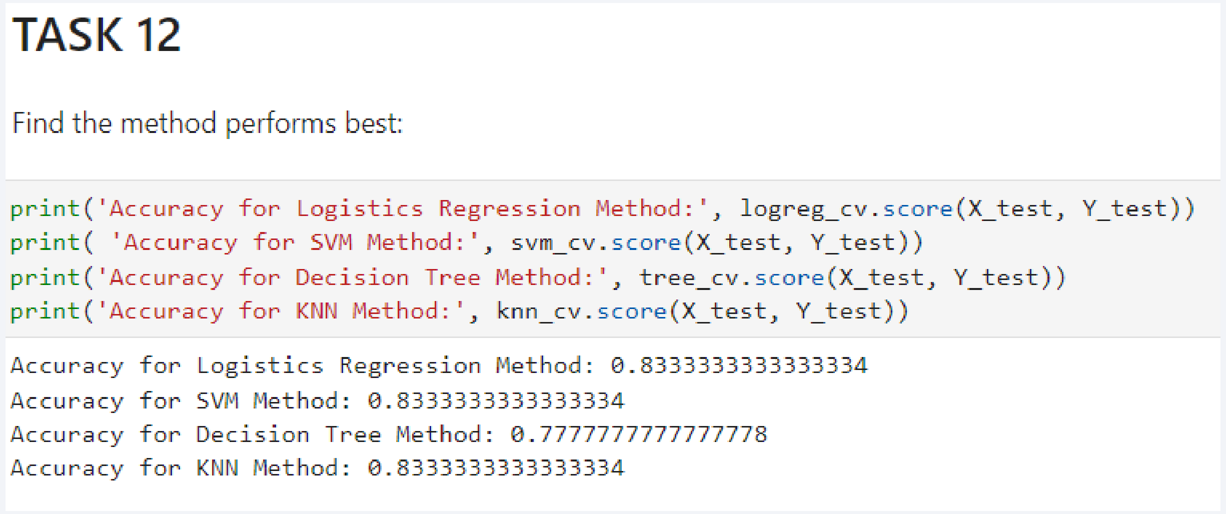


Section 5

Predictive Analysis (Classification)

Classification Accuracy

- On my test data, Logistics Regression, SVM and KNN all provided 0.83 as the best



TASK 12

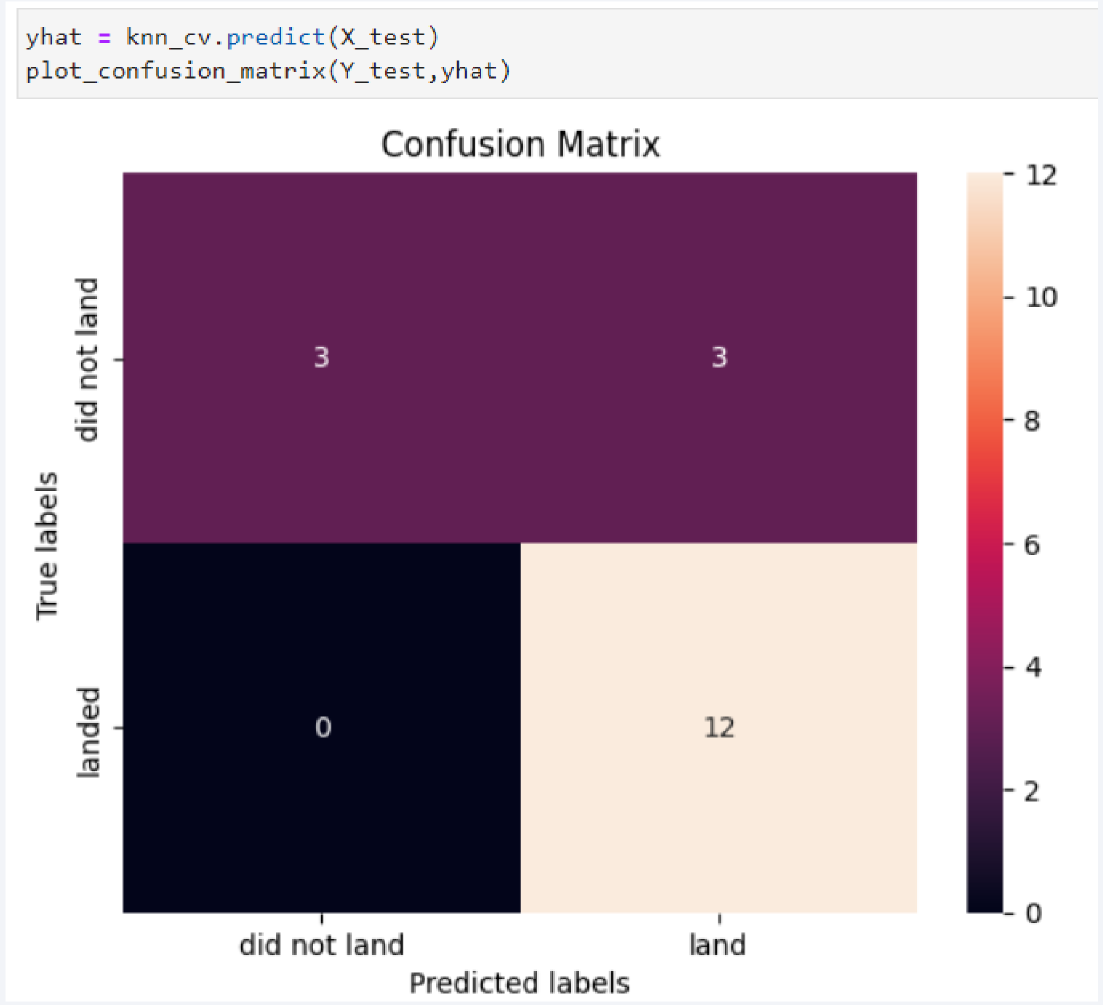
Find the method performs best:

```
print('Accuracy for Logistics Regression Method:', logreg_cv.score(X_test, Y_test))
print( 'Accuracy for SVM Method:', svm_cv.score(X_test, Y_test))
print('Accuracy for Decision Tree Method:', tree_cv.score(X_test, Y_test))
print('Accuracy for KNN Method:', knn_cv.score(X_test, Y_test))
```

```
Accuracy for Logistics Regression Method: 0.8333333333333334
Accuracy for SVM Method: 0.8333333333333334
Accuracy for Decision Tree Method: 0.7777777777777778
Accuracy for KNN Method: 0.8333333333333334
```

Confusion Matrix

- This model gives a 0.83 accuracy.



```
yhat = knn_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```

Conclusions

- The larger the flight amount the more successful launching output is expected
- The successes in flight launching has greatly improved in the recent years
- KSC LC-39A has the most successful launches
- Payloads between 2000-4000 are best
- Booster version FT had the best success rate
- Logistics Regression, SVM and KNN all provided the same accuracy on test data

Appendix

- None

Thank you!