IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Rayhaan Dustagheer
21st May 2024

Outline

Executive Summary

Summary of Methodologies:

This project follows the below mentioned steps:

- Data collection and Data wrangling
- Exploratory Data Analysis and Feature engineering
- Data analytics through interactive visual analytics
- Data modelling for Predictive Analysis (Classification)

Summary of Results:

1. Exploratory Data Analysis (EDA) results
2. Geospatial analysis and Interactive Dashboards
3. Predictive analysis through classification models

Introduction

- SpaceX's Falcon 9 rockets are launched at a cost of approximately $62 million, significantly less expensive than other providers, whose costs typically exceed $165 million. A major factor in these savings is SpaceX's ability to land and reuse the rocket's first stage.
- By predicting whether the first stage will successfully land, we can estimate the launch cost and use this information to evaluate if an alternative company should compete with SpaceX for a rocket launch contract.
- The goal of this project is to predict whether the SpaceX Falcon 9 first stage will successfully land.

Section 1

Methodology

Methodology

- Data collection methodology:
  - Making GET requests to the SpaceX REST API
  - Web Scraping
- Perform data wrangling
  - Using the .fillna() method to remove NaN values
  - Using the .value_counts() method to determine the following: Number of launches on each site, number and frequency of mission outcomes per orbit type, frequency of each orbit
  - Create an outcome of 0 when booster did not land successfully and an outcome of 1 when booster did land successfully

Methodology (continued)

- Exploratory Data Analysis
  - Using sql to manipulate and query SpaceX dataset
  - For visualization of statistical graphs, Matplotlib and Pandas was used for pattern and relationship detection
- Interactive Visual Analytics
  - Analyzed geospatial data using Folium and created interactive dashboard using Plotly Dash
- Data Modelling and Evaluation
  - The dataset was preprocessed and standardized using Scikit-Learn library
  - The dataset was split into a test set, train set and validation set
  - 4 models were used and the best hyperparameters for each model was determined using the GridSearch with Cross validation algorithm
  - The different models were trained, a confusion matrix for each model was plotted and the accuracy score of each model was calculated to determine

Data Collection

- Data for the Space X launches were collected using 2 main methods: Rest API and Web Scraping.
- The data after being fetched from the Rest API were put in a Pandas Dataframe for further analysis
- The other part of the dataset was obtained through web scraping and then the returned data was parse to extract data from tables present in the html page.

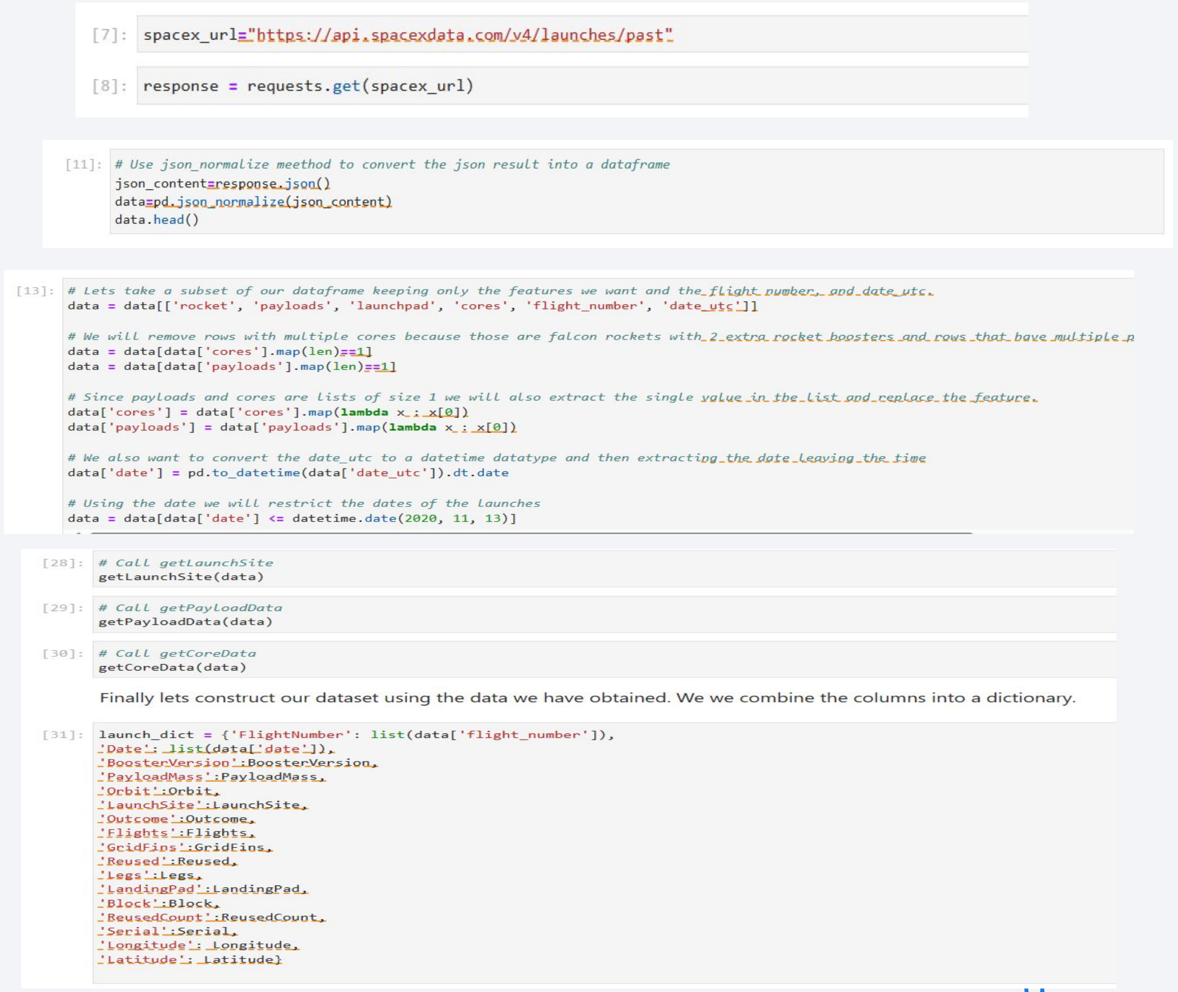
Data Collection – SpaceX API

A GET request was made to SpaceX REST API. The response was then converted to a json file which was then imported into a Pandas DataFrame

The data was then processed to obtain the relevant features such as rocket, payloads, lauchpad and so on. The dates were displayed in the correct format

For each feature, a list was created and defined functions were called to map the relevant features to the lists. These lists were then used to construct a dictionary that had keys matching the list of features

This dictionary was then converted to a Pandas Dataframe to store the final dataset. The dataframe was then filtered out to incorporate only Falcon 9 Launches



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

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

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

```
[13]: # 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 multiple p
      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)]
```

```
[28]: # Call getLaunchSite
      getLaunchSite(data)

[29]: # Call getPayloadData
      getPayloadData(data)

[30]: # Call getCoreData
      getCoreData(data)
```

Finally lets construct our dataset using the data we have obtained. We we combine the columns into a dictionary.

```
[31]: 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}
```

Data Collection - Scraping

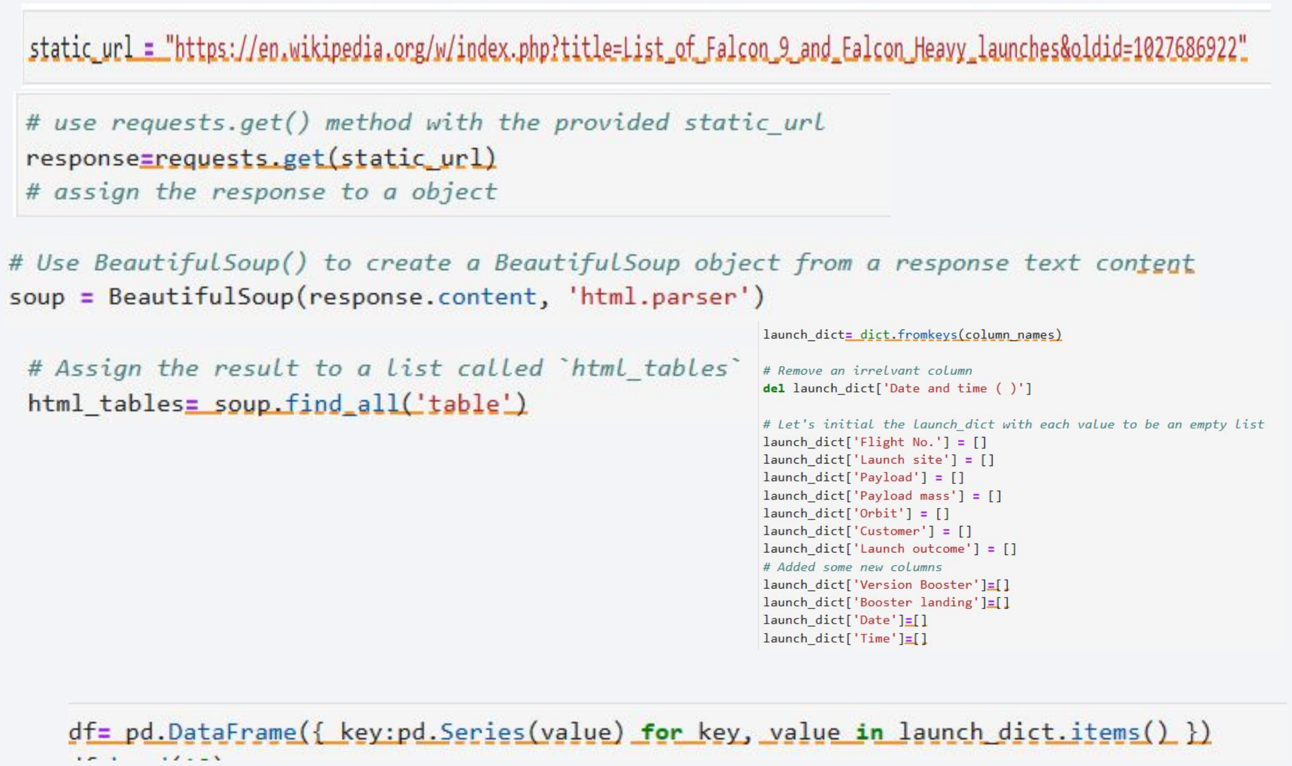
1. Request the HTML page from the static URL through HTTP GET and store the response

2. A Beautiful Soup object is then created and http response is assigned to that object for further parsing and scraping

3. The data is analyzed and the html table tags is applied to the Beautiful Soup object to retrieve all relevant tables

4. A dictionary of column names is created. Custom functions are applied to retrieve data from tables and added to dictionary.

5. The dictionary of data is then converted to a pandas Dataframe for further processing and storage into a csv file



```
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
response=requests.get(static_url)
# assign the response to a object
```

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

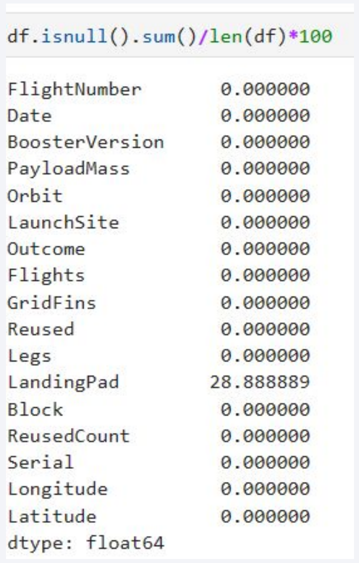
```
# Assign the result to a list called `html_tables`
html_tables= soup.find_all('table')
```

```
launch_dict= dict.fromkeys(column_names)

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

# Let's initial the launch_dict with each value to be an empty list
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']=[]
```

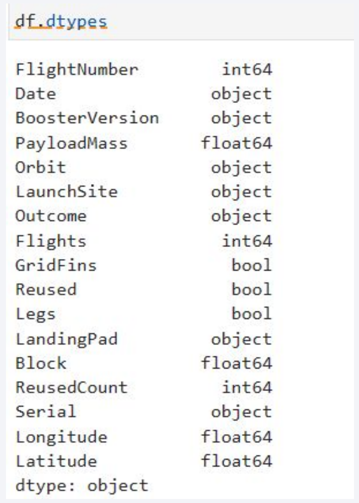
```
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
```

Data Wrangling

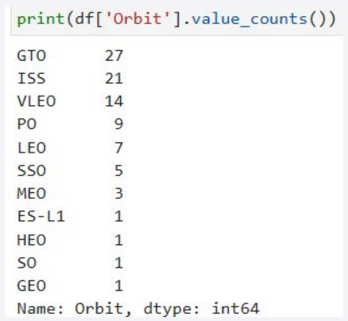
- To obtain a good prediction later on, the dataset has to be cleaned and processed to provide the relevant intended and correct values
- Here for each feature the percentage and number of missing values was calculated.
- The type of data for each feature was determined
- Launches are done from specific sites, and each launch is done to a specific orbit. For each launch the outcome is either a failure or success. To better understand the data, the value counts for the Orbit, Launch Sites and Outcome feature was calculated respectively.



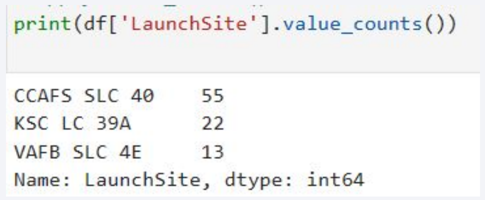
```
df.isnull().sum()/len(df)*100

FlightNumber       0.000000
Date               0.000000
BoosterVersion     0.000000
PayloadMass        0.000000
Orbit              0.000000
LaunchSite         0.000000
Outcome            0.000000
Flights            0.000000
GridFins           0.000000
Reused             0.000000
Legs               0.000000
LandingPad        28.888889
Block              0.000000
ReusedCount        0.000000
Serial             0.000000
Longitude          0.000000
Latitude           0.000000
dtype: float64
```



```
df.dtypes

FlightNumber        int64
Date               object
BoosterVersion     object
PayloadMass       float64
Orbit              object
LaunchSite         object
Outcome            object
Flights             int64
GridFins             bool
Reused               bool
Legs                 bool
LandingPad         object
Block             float64
ReusedCount         int64
Serial             object
Longitude         float64
Latitude          float64
dtype: object
```

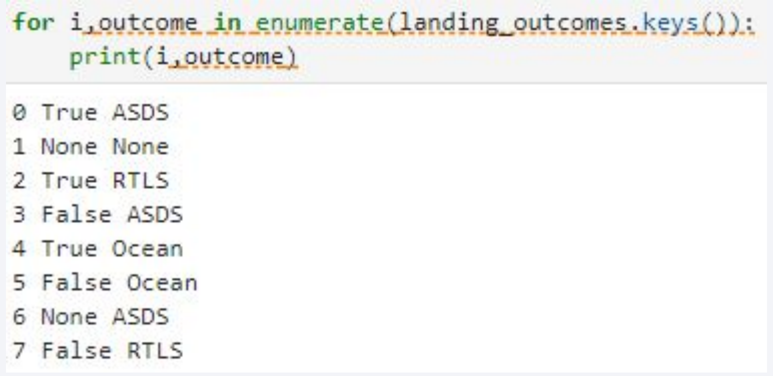


```
print(df['Orbit'].value_counts())

GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
ES-L1     1
HEO       1
SO        1
GEO       1
Name: Orbit, dtype: int64
```



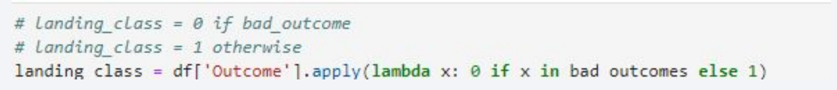
```
print(df['LaunchSite'].value_counts())

CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

Data Wrangling (continued)

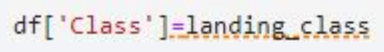


```
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
```

```
0 True ASDS
1 None None
2 True RTLS
3 False ASDS
4 True Ocean
5 False Ocean
6 None ASDS
7 False RTLS
```



```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = df['Outcome'].apply(lambda x: 0 if x in bad_outcomes else 1)
```



```
df['Class']=landing_class
```

- The following context gives an overview of launch sites, orbits and outcomes.

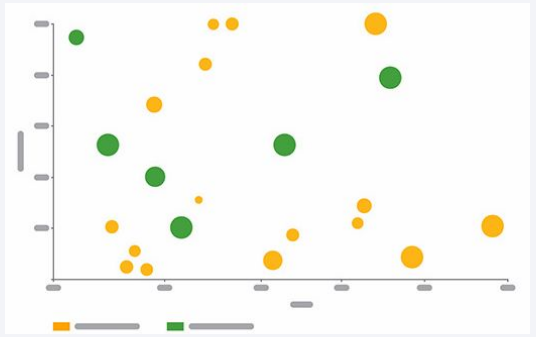
  True Ocean – the mission successfully landed in a specific ocean region.
  False Ocean – the mission unsuccessfully landed in a specific ocean region.
  True RTLS – the mission successfully landed on a ground pad.
  False RTLS – the mission unsuccessfully landed on a ground pad.
  True ASDS – the mission successfully landed on a drone ship.
  False ASDS – the mission unsuccessfully landed on a drone ship.
  None ASDS and None None – these indicate a failure to land.

- Data Wrangling Process Overview:

1. To determine if a booster will successfully land, it's useful to create a binary column (1 for success, 0 for failure). This can be achieved through the following steps:
2. Define a set of unsuccessful (bad) outcomes, called bad_outcome.
3. Create a list, landing_class, where each element is 0 if the corresponding row in Outcome is in bad_outcome, otherwise it's 1.
4. Add a Class column to the DataFrame that contains the values from the landing_class list.
5. Export the DataFrame as a .csv file.

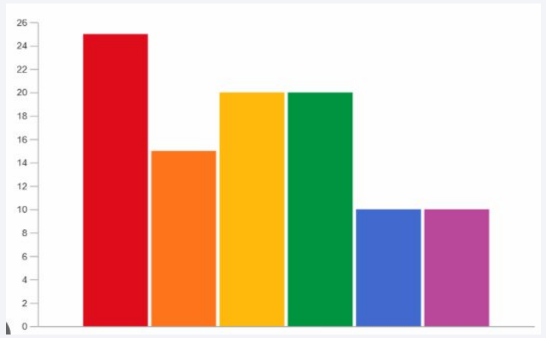
EDA with Data Visualization

Scatter Charts



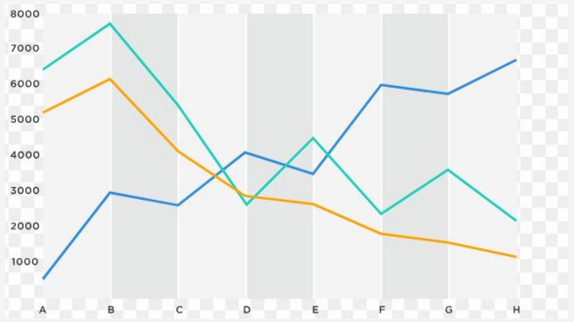
Scatter charts are useful for observing relationships or correlations between two numerical variables.
Scatter charts were used to view the relationship of:
- Flight number and launch sites
- Orbit type and flight number
- Payload and Launch Site
- Payload and Orbit Type

Bar Charts



Bar charts are used to compare numerical values with categorical variables. Depending on the data size, you can use either horizontal or vertical bar charts.
Bar chart was used to visualize the relationship between:
- Success Rate and Orbit type

Line charts



Line charts display numerical values on both axes and are typically used to illustrate changes in a variable over time.

A line chart was used to illustrate the relationship between:
- Success Rate and Year (i.e. the launch success yearly trend)

EDA with SQL

To extract insights from the dataset, several SQL queries were executed. These queries aimed to:

- Display the names of the unique launch sites in the space missions.
- Retrieve 5 records where the launch sites start with the string 'CCA'.
- Calculate the total payload mass carried by boosters launched by NASA (CRS).
- Compute the average payload mass carried by the booster version F9 v1.1.
- Identify the date of the first successful landing on a ground pad.
- List the names of boosters that successfully landed on a drone ship with a payload mass between 4000 and 6000 kg.
- Count the total number of successful and failed mission outcomes.
- Identify the booster versions that carried the maximum payload mass.
- List the failed landing outcomes on drone ships, along with their booster versions and launch site names for the year 2015.
- Rank the landing outcomes (such as Failure on a drone ship or Success on a ground pad) between 2010-06-04 and 2017-03-20 in descending order.

Build an Interactive Map with Folium

The following steps were taken to visualize the launch data on an interactive map:

1. Marking Launch Sites on the Map:

   - Initialize the map using a Folium Map object.

   - For each launch site, add a folium.Circle and a folium.Marker to the map.

2. Indicating Launch Success/Failure on the Map:

   - Given that many launches share the same coordinates, it's practical to cluster them together.

   - Before clustering, assign marker colors: green for successful launches (class = 1) and red for failed launches (class = 0).

   - For each launch, add a folium.Marker to a MarkerCluster() object.

   - Create an icon with a text label, setting the icon color to the previously determined marker color.

3. Calculating Distances Between Launch Sites and Their Proximities:

   - To analyze the proximity of launch sites, calculate the distances between points using their latitude and longitude values.

   - Mark a point using the latitude and longitude values and create a folium.Marker object to display the distance.

   - To show the distance line between two points, draw a folium.PolyLine and add it to the map.

Build a Dashboard with Plotly Dash

The following visualizations were incorporated into a Plotly Dash dashboard for an interactive exploration of the data:

1. *Pie Chart (px.pie()):*

- Displays the total number of successful launches per site.
- Clearly highlights which sites have the highest success rates.
- Includes a filter option using a dcc.Dropdown() to view the success/failure ratio for individual sites.

2. *Scatter Plot (px.scatter()):*

- Illustrates the correlation between the outcome (success or failure) and payload mass (kg).
- Features a RangeSlider() to filter the data by different payload mass ranges.
- Additionally, includes a filter to sort the data by booster version.

Predictive Analysis (Classification)

For the purpose of this exercise, multiple classifications were built and the best performing one was selected for prediction

Model Creation

To prepare the dataset for model development, the following steps were undertaken:

1. Load the Dataset:

Import the dataset for analysis.

2. Data Transformation:

Perform necessary data transformations, including standardization and preprocessing.

3. Split Data:

Divide the data into training and test sets using train_test_split().

4. Algorithm Selection:

Determine the most suitable machine learning algorithms for the task.

5. Model Training:

For each selected algorithm:

Create a GridSearchCV object along with a dictionary of parameters.

Fit the GridSearchCV object to these parameters.

Train the model using the training dataset.

Evaluation of model

For each selected algorithm, perform the following steps using the GridSearchCV output:

1. Evaluate Tuned Hyperparameters:

Retrieve and examine the best hyperparameters (best_params_).

2. Assess Accuracy:

Check the accuracy scores (score and best_score_).

3. Analyze Confusion Matrix:

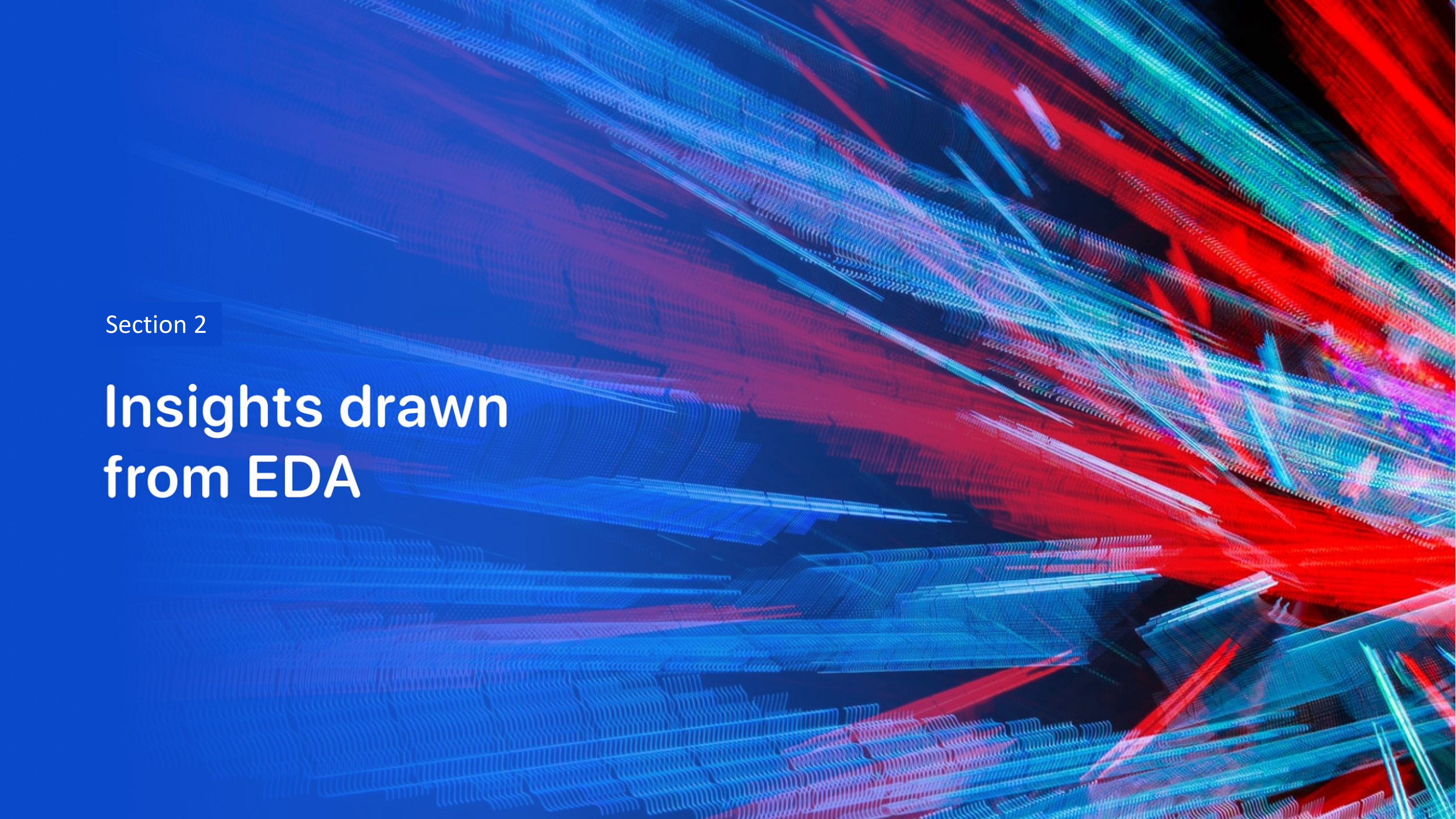
Plot and review the Confusion Matrix to evaluate model performance.

Choosing best model

1. Evaluate the accuracy scores of all the selected algorithms.
2. The model exhibiting the highest accuracy score is identified as the top-performing model

Results

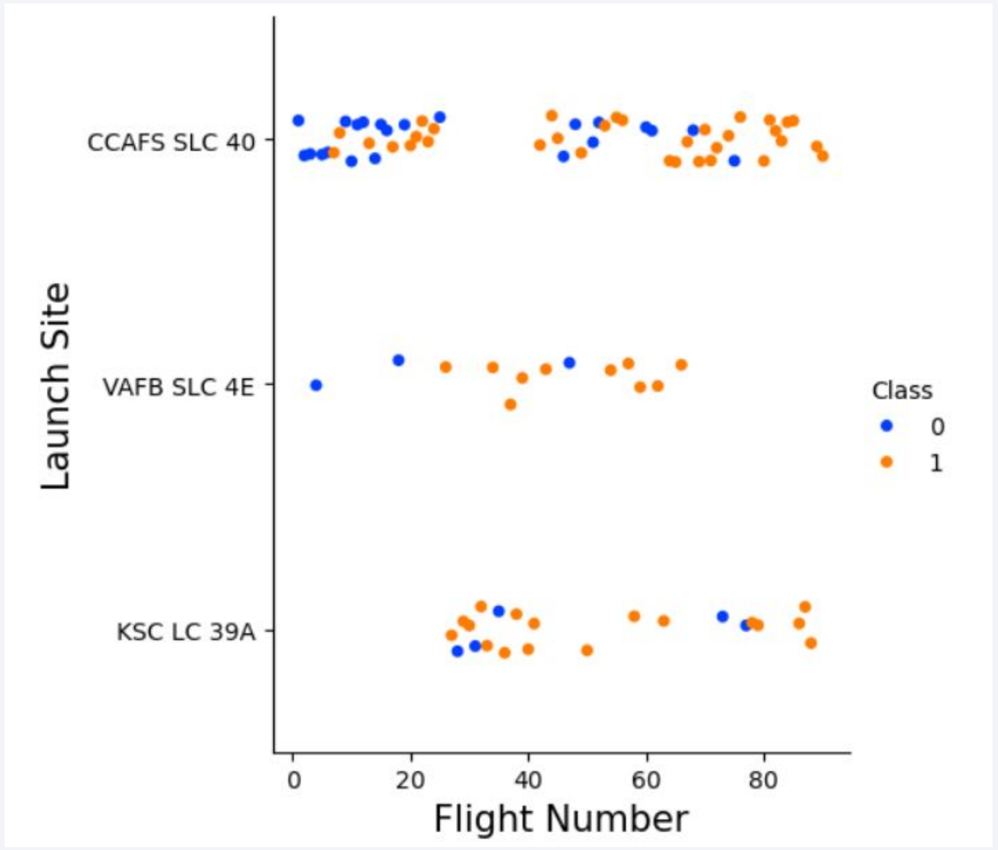
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results



Section 2

Insights drawn from EDA

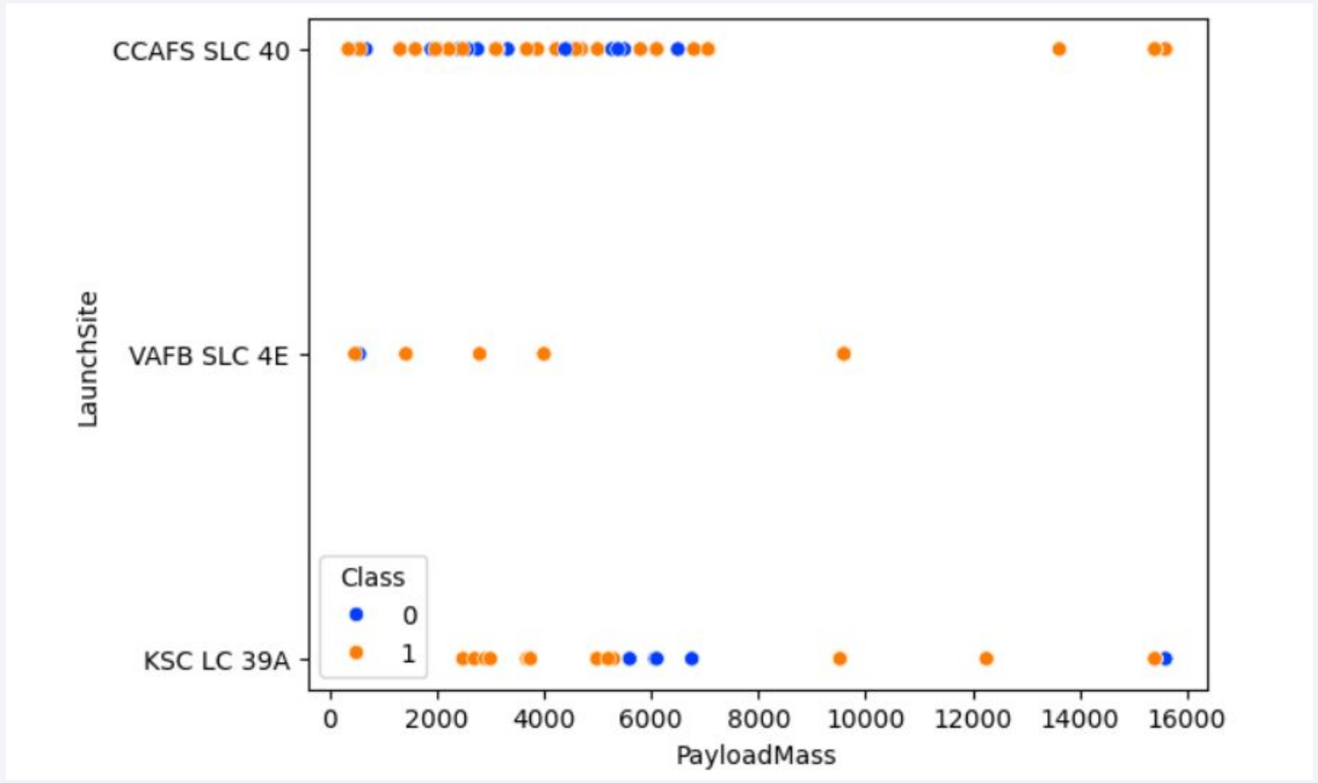
Flight Number vs. Launch Site



The scatter plot comparing Launch Site and Flight Number reveals the following insights:

- Success rates at a launch site improve as the number of flights increases.
- Early flights (flight numbers below 30) predominantly launched from CCAFS SLC 40 were mostly unsuccessful.
- VAFB SLC 4E also exhibited a similar pattern, with initial flights showing lower success rates.
- KSC LC 39A did not have early flights, resulting in higher success rates for launches from this site.
- After reaching a flight number of approximately 30, there is a notable increase in successful landings (Class = 1).

Payload vs. Launch Site



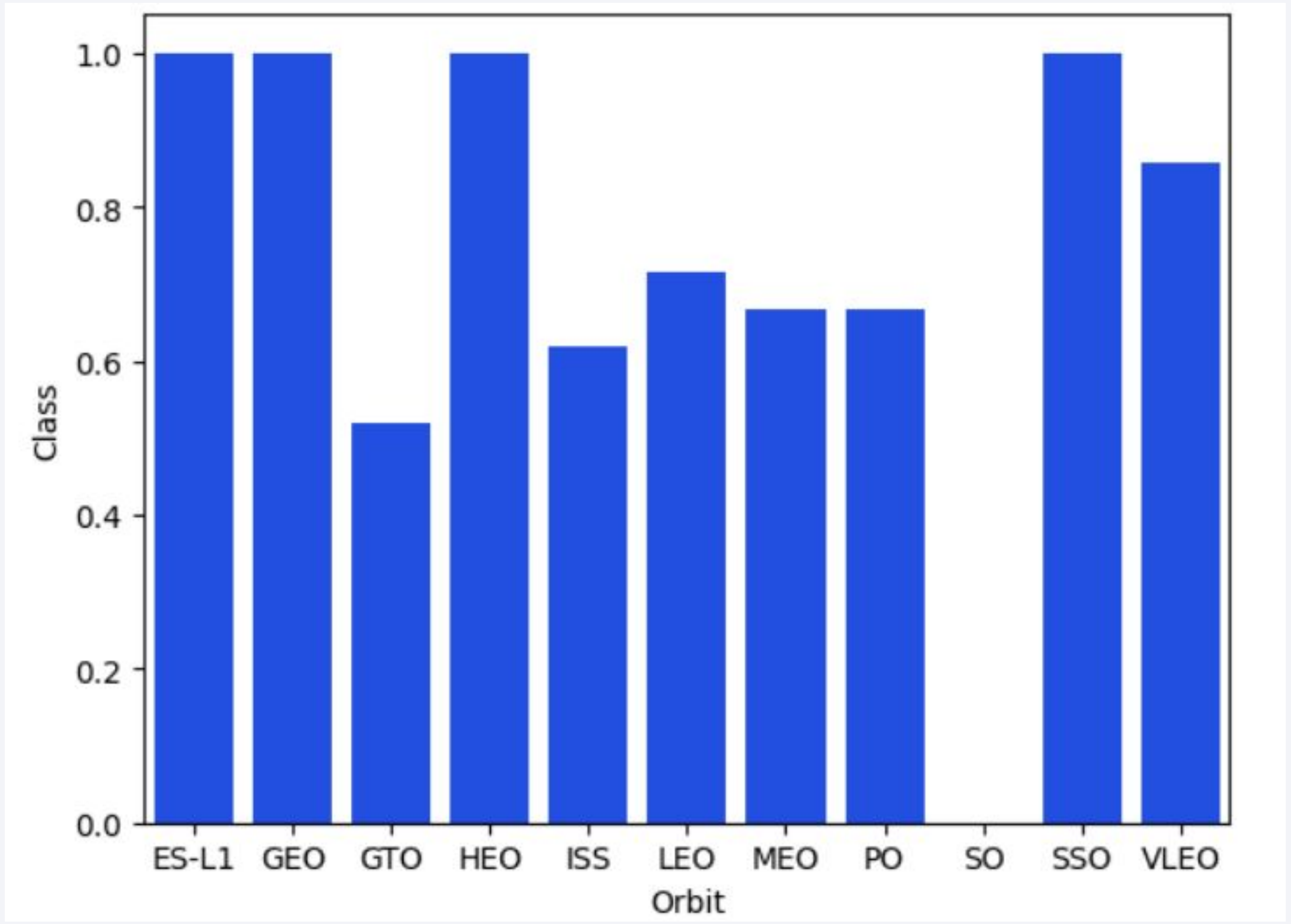
The scatter plot comparing Launch Site and Payload Mass indicates the following points:

- For payloads exceeding approximately 7000 kg, unsuccessful landings are rare, though data for these heavier launches is sparse.

- There is no apparent relationship between payload mass and the success rate at any specific launch site.

- All launch sites handled a range of payload masses, with CCAFS SLC 40 typically launching lighter payloads, although there are some exceptions.

Success Rate vs. Orbit Type



The bar chart depicting Success Rate versus Orbit Type reveals the following orbits with a perfect (100%) success rate:

- ES-L1 (Earth-Sun First Lagrangian Point)
- GEO (Geostationary Orbit)
- HEO (High Earth Orbit)
- SSO (Sun-synchronous Orbit)

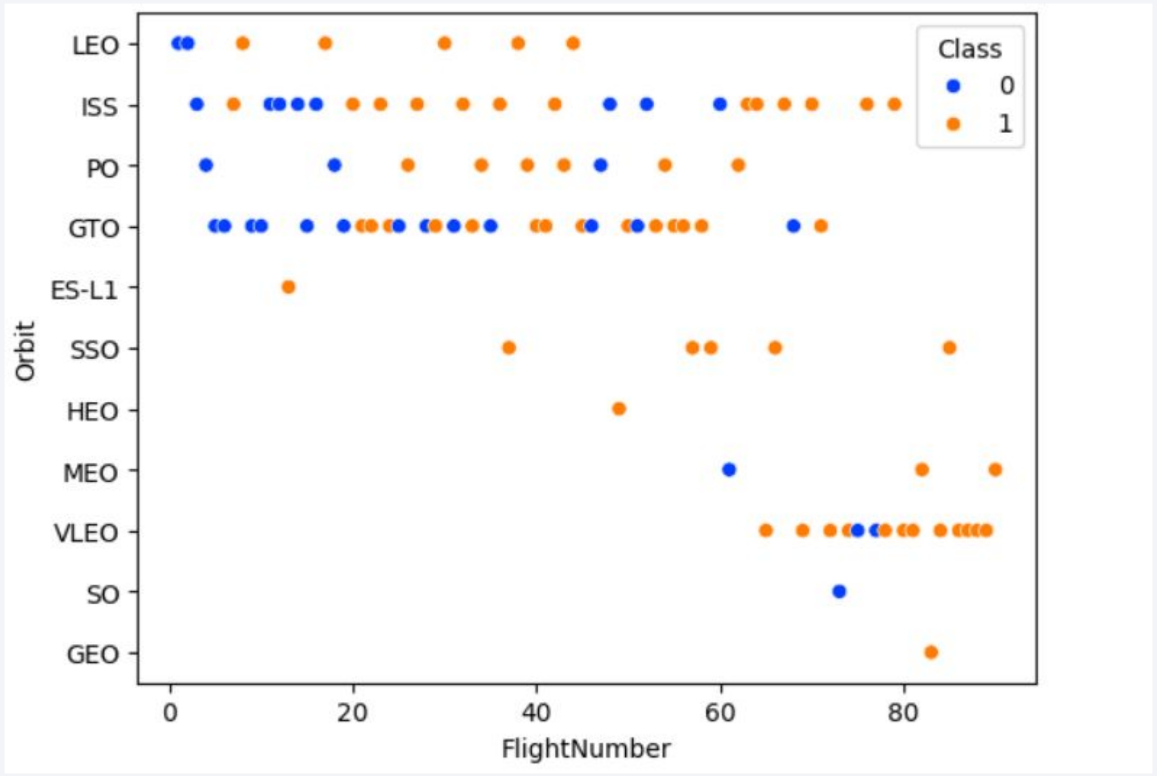
Conversely, the orbit with the lowest success rate (0%) is:

- SO (Heliocentric Orbit)

Flight Number vs. Orbit Type

This scatter plot of Orbit Type versus Flight Number reveals several insights not shown in previous plots:

- The perfect (100%) success rates for GEO, HEO, and ES-L1 orbits are due to each having only one flight.
- The 100% success rate for SSO is notable, with five successful flights.
- There is minimal correlation between Flight Number and Success Rate for GTO.
- In general, the success rate improves as the Flight Number increases. This trend is especially pronounced for LEO, where failures occurred only during the early flights with lower flight numbers.



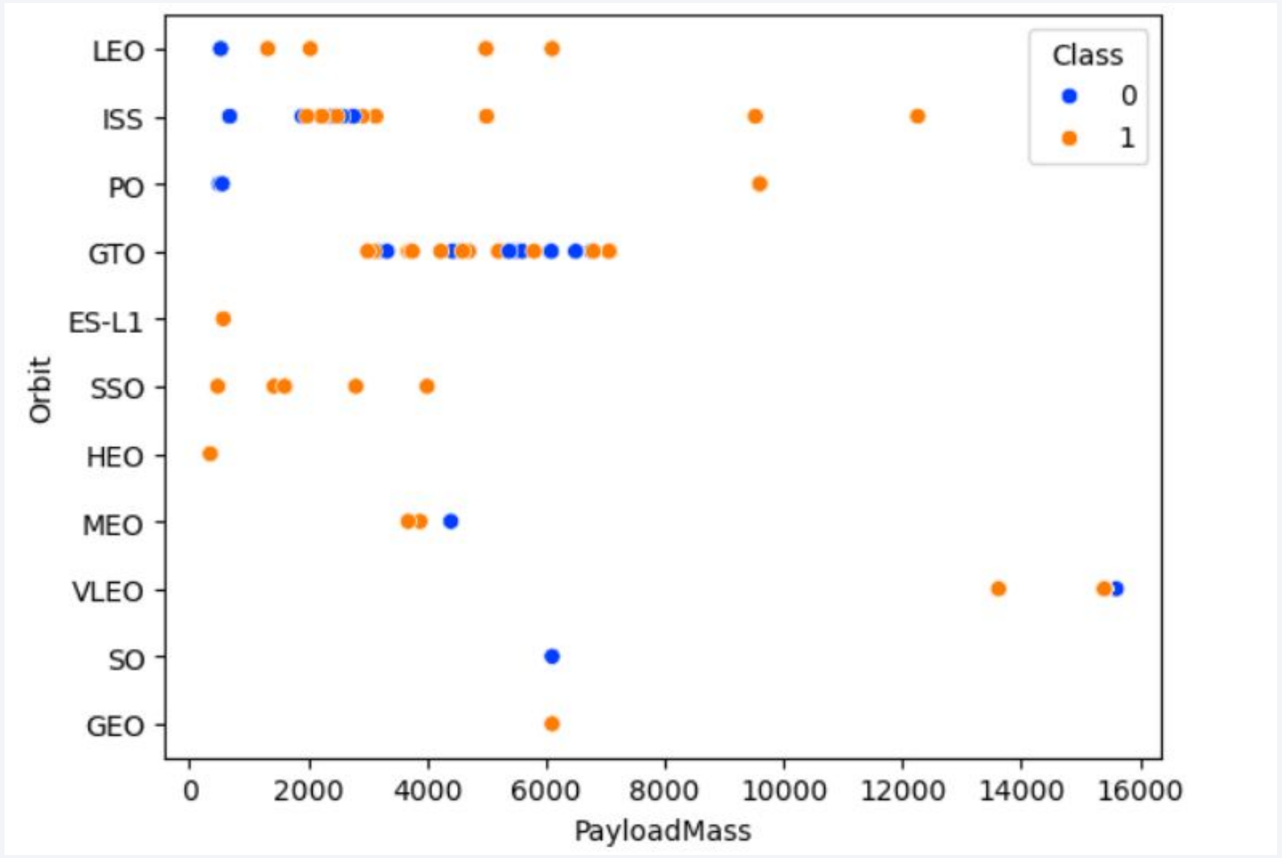
Payload vs. Orbit Type

This scatter plot of Orbit Type versus Payload Mass indicates that:

- PO (despite limited data points), ISS, and LEO orbits achieve higher success rates with heavier payloads.

- For GTO, the link between payload mass and success rate remains ambiguous.

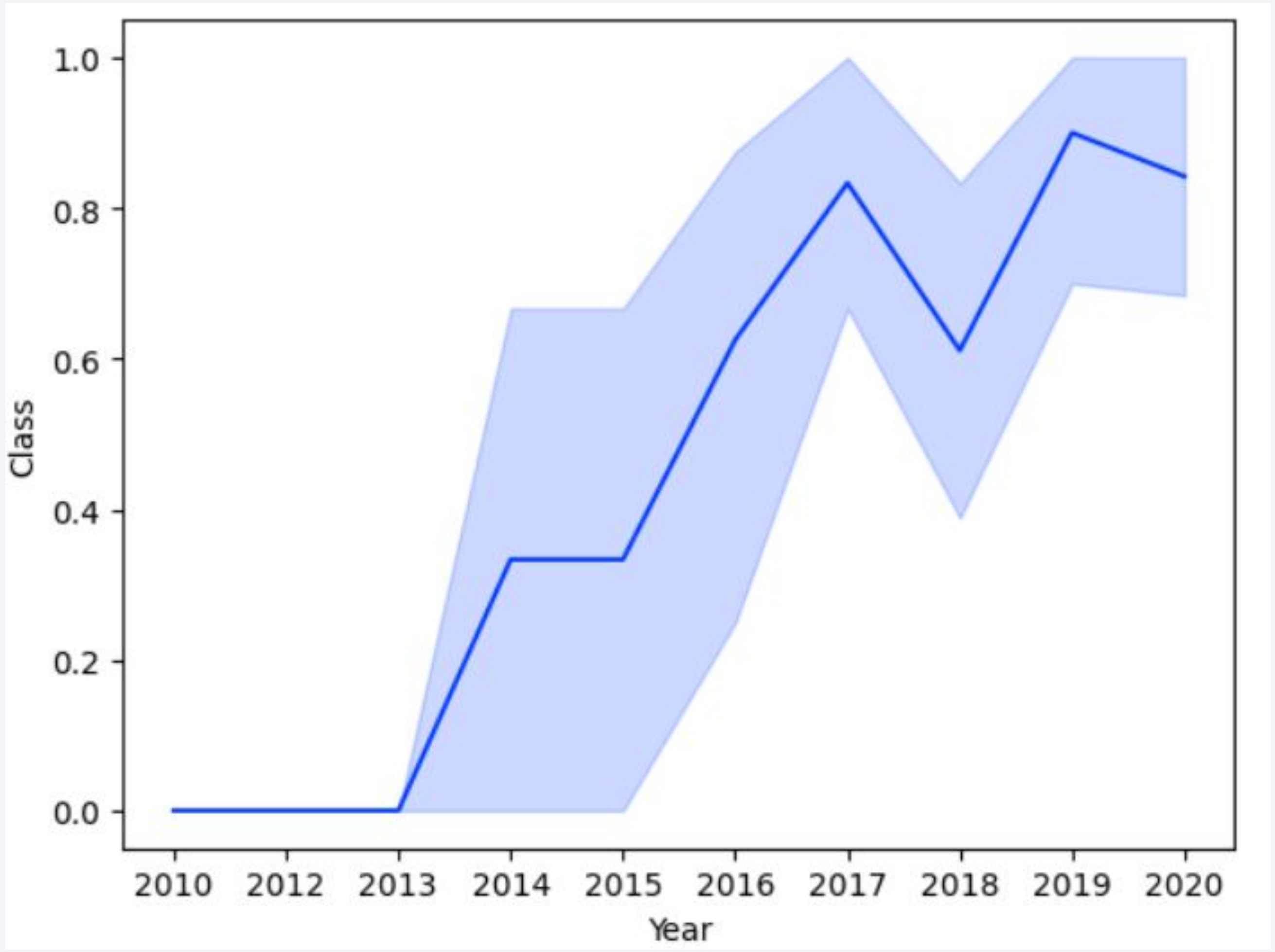
- VLEO (Very Low Earth Orbit) launches typically carry heavier payloads, which is logically expected.



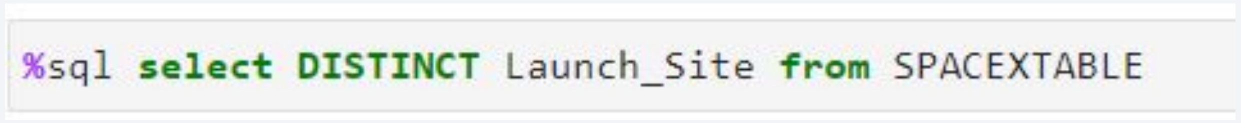
Launch Success Yearly Trend

The line chart of yearly average success rate shows that:

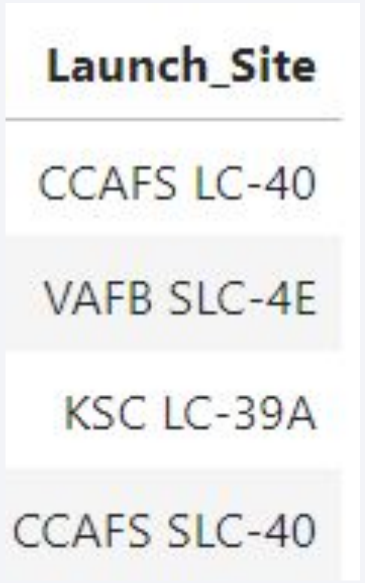
- Between 2010 and 2013, all landings were unsuccessful (as the success rate is 0).
- After 2013, the success rate generally increased, despite small dips in 2018 and 2020.
- After 2016, there was always a greater than 50% chance of success.



All Launch Site Names



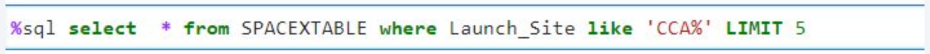
```
%sql select DISTINCT Launch_Site from SPACEXTABLE
```



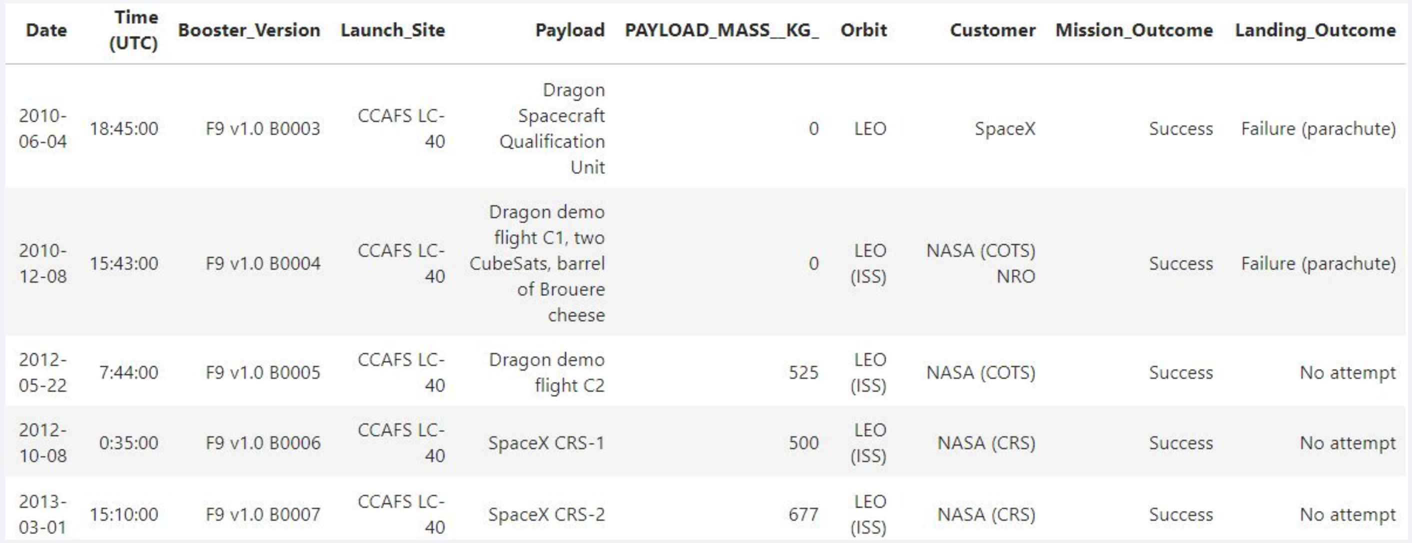
Launch_Site
CCAFS LC-40
VAFB SLC-4E
KSC LC-39A
CCAFS SLC-40

- Here the word DISTINCT is used to select only the unique names of lauch sites

Launch Site Names Begin with 'CCA'



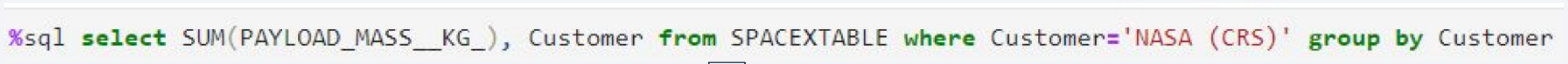
```
%sql select  * from SPACEXTABLE where Launch_Site like 'CCA%' LIMIT 5
```



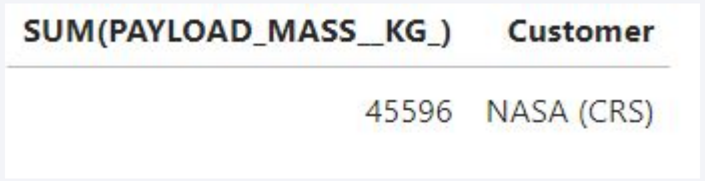
Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight 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

- Here the keyword LIKE is used to find pattern of CCA in Launch Sites and the keyword LIMIT is used to show only the 5 top records

Total Payload Mass



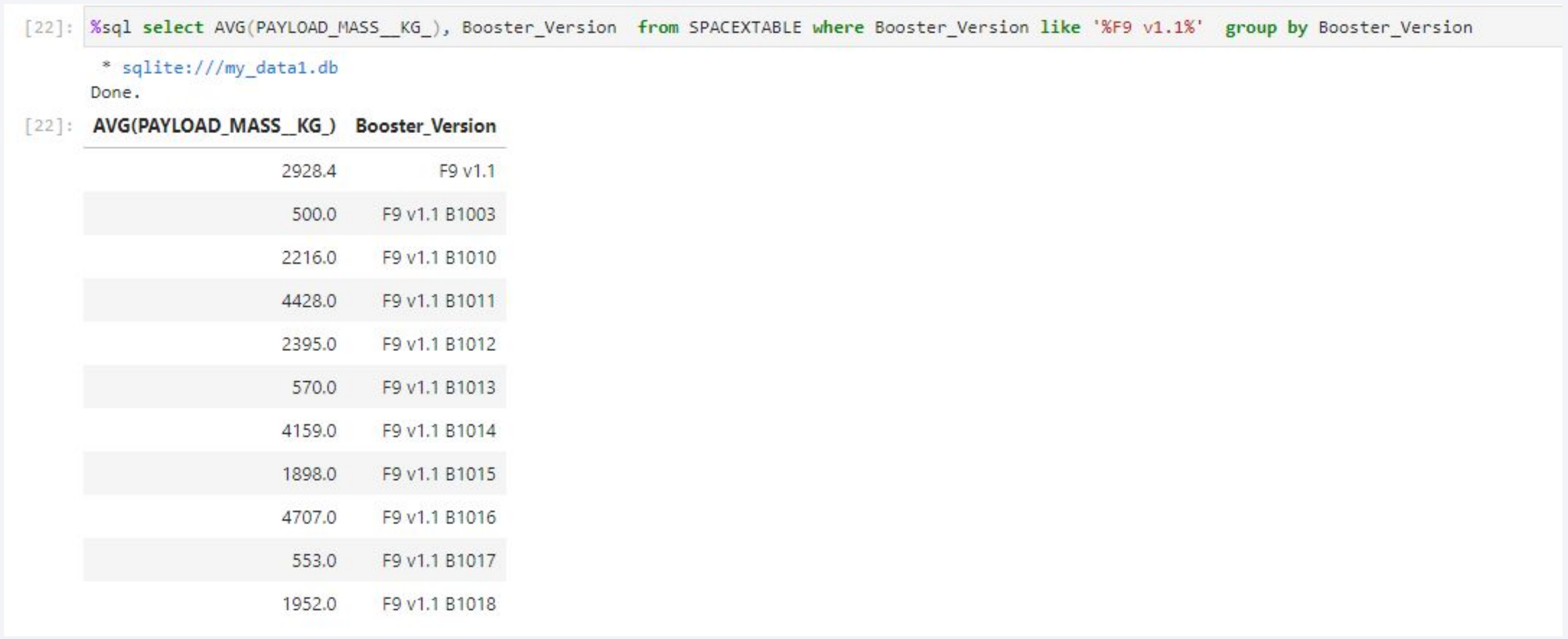
```
%sql select SUM(PAYLOAD_MASS__KG_), Customer from SPACEXTABLE where Customer='NASA (CRS)' group by Customer
```



SUM(PAYLOAD_MASS__KG_)	Customer
45596	NASA (CRS)

- The WHERE clause is used to check for only NASA (CRS) customers while the SUM does a summation of all the Payload Mass of the NASA customers
- The group by clause is used to group the sum by customers

Average Payload Mass by F9 v1.1



```
[22]: %sql select AVG(PAYLOAD_MASS__KG_), Booster_Version  from SPACEXTABLE where Booster_Version like '%F9 v1.1%'  group by Booster_Version
```

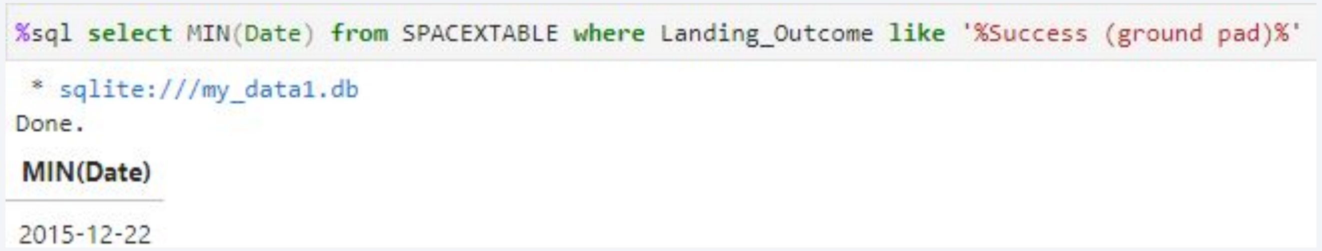
```
 * sqlite:///my_data1.db
Done.
```

[22]:

AVG(PAYLOAD_MASS__KG_)	Booster_Version
2928.4	F9 v1.1
500.0	F9 v1.1 B1003
2216.0	F9 v1.1 B1010
4428.0	F9 v1.1 B1011
2395.0	F9 v1.1 B1012
570.0	F9 v1.1 B1013
4159.0	F9 v1.1 B1014
1898.0	F9 v1.1 B1015
4707.0	F9 v1.1 B1016
553.0	F9 v1.1 B1017
1952.0	F9 v1.1 B1018

To obtain the average payload mass for the F9 v1.1 booster version, the AVG keyword is used in conjunction with the WHERE clause to give the desired result.

First Successful Ground Landing Date



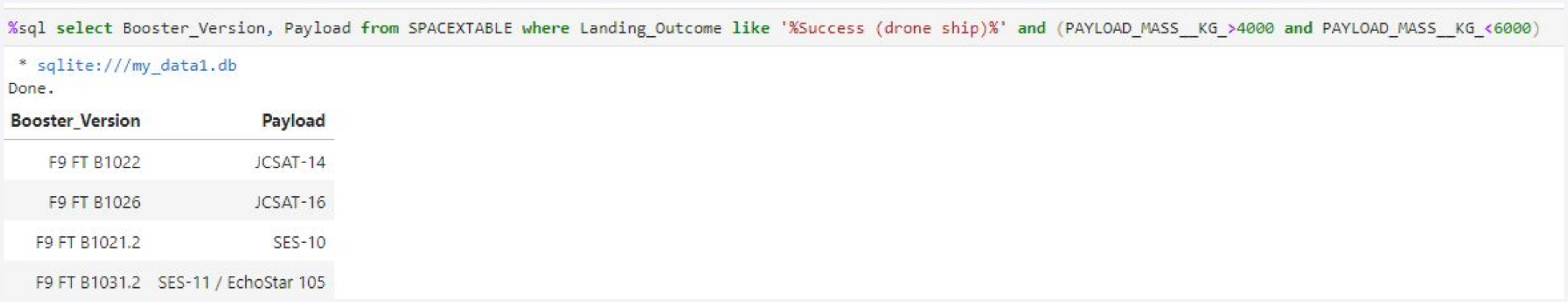
```
%sql select MIN(Date) from SPACEXTABLE where Landing_Outcome like '%Success (ground pad)%'
```

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

MIN(Date)
2015-12-22

The MIN keyword is used here to find the earliest date and the LIKE clause is used to limit the results to Success (ground pad) Landing outcomes

Successful Drone Ship Landing with Payload between 4000 and 6000



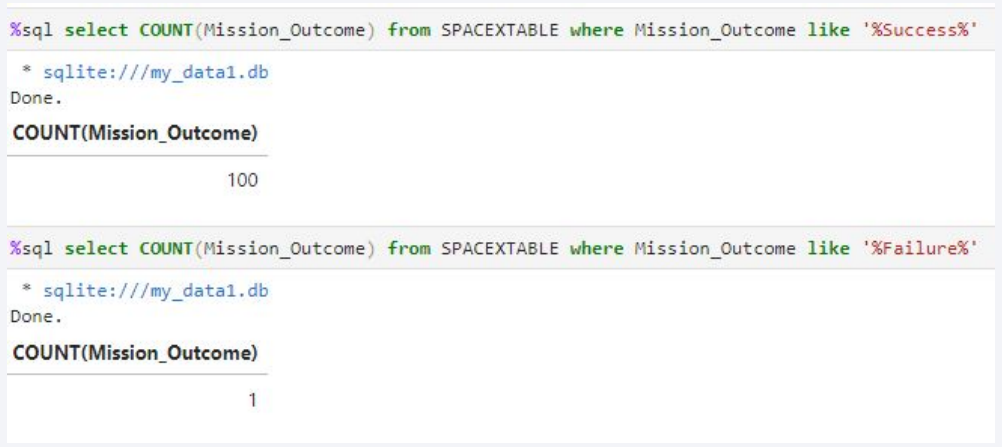
```
%sql select Booster_Version, Payload from SPACEXTABLE where Landing_Outcome like '%Success (drone ship)%' and (PAYLOAD_MASS__KG_>4000 and PAYLOAD_MASS__KG_<6000)
```

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

Booster_Version	Payload
F9 FT B1022	JCSAT-14
F9 FT B1026	JCSAT-16
F9 FT B1021.2	SES-10
F9 FT B1031.2	SES-11 / EchoStar 105

Here a condition of PAYLOAD_MASS_KG > 4000 and PAYLOAD_MASS_KG <6000 is used to find the corresponding data for Landing outcome of Success (drone ship)

Total Number of Successful and Failure Mission Outcomes



```
%sql select COUNT(Mission_Outcome) from SPACEXTABLE where Mission_Outcome like '%Success%'
```

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

COUNT(Mission_Outcome)
100

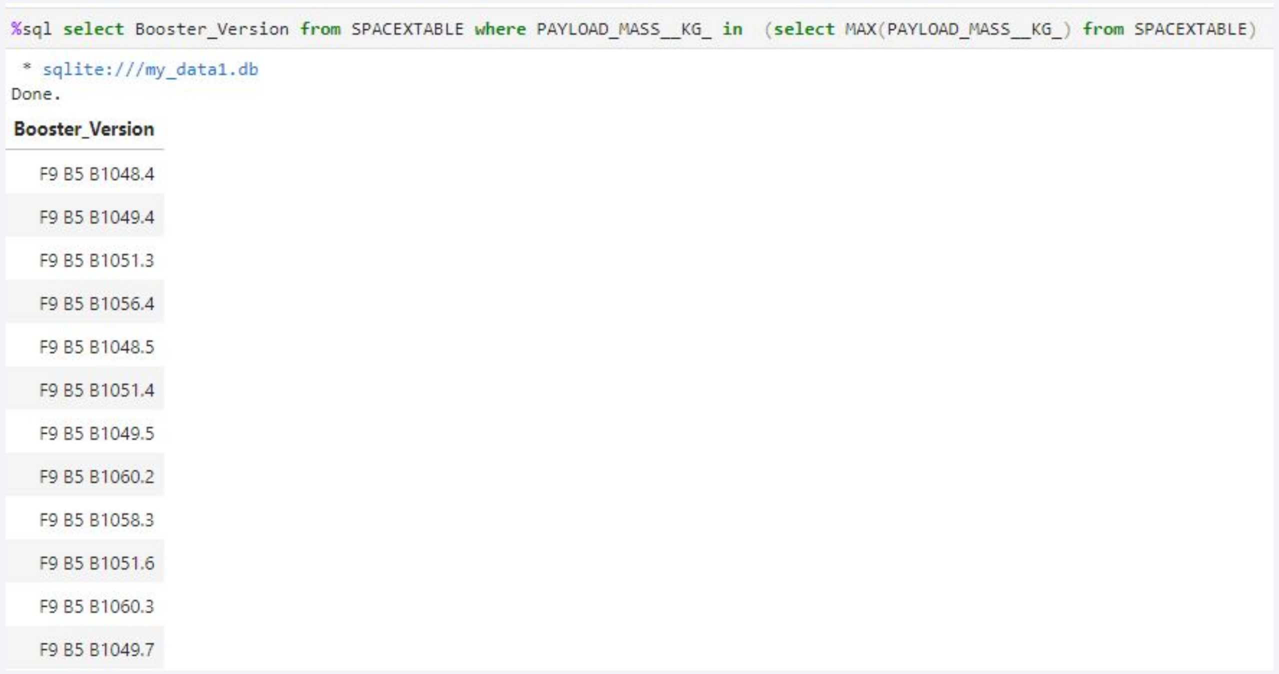
```
%sql select COUNT(Mission_Outcome) from SPACEXTABLE where Mission_Outcome like '%Failure%'
```

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

COUNT(Mission_Outcome)
1

- Here there are 2 queries: one for success and one for failure
- A count method was used to check for both success and failure.

Boosters Carried Maximum Payload



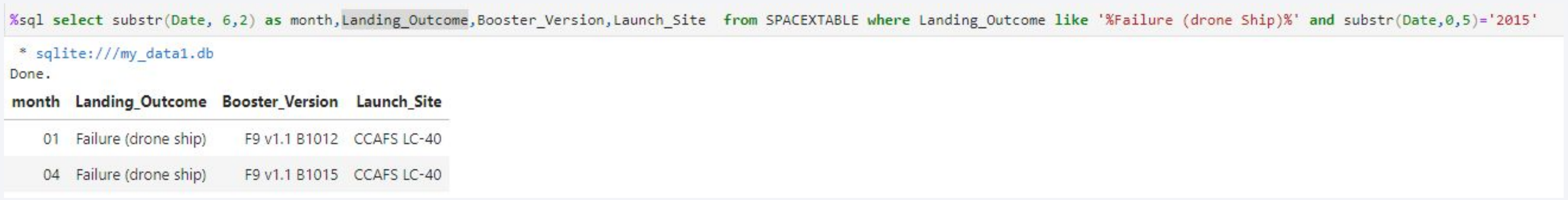
```
%sql select Booster_Version from SPACEXTABLE where PAYLOAD_MASS__KG_ in  (select MAX(PAYLOAD_MASS__KG_) from SPACEXTABLE)
```

```
 * 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

- Here a subquery has been used. The inner query uses MAX to get the highest payload mass and the outer queries uses IN to check for every booster version that has payload mass as the max payload mass

2015 Launch Records

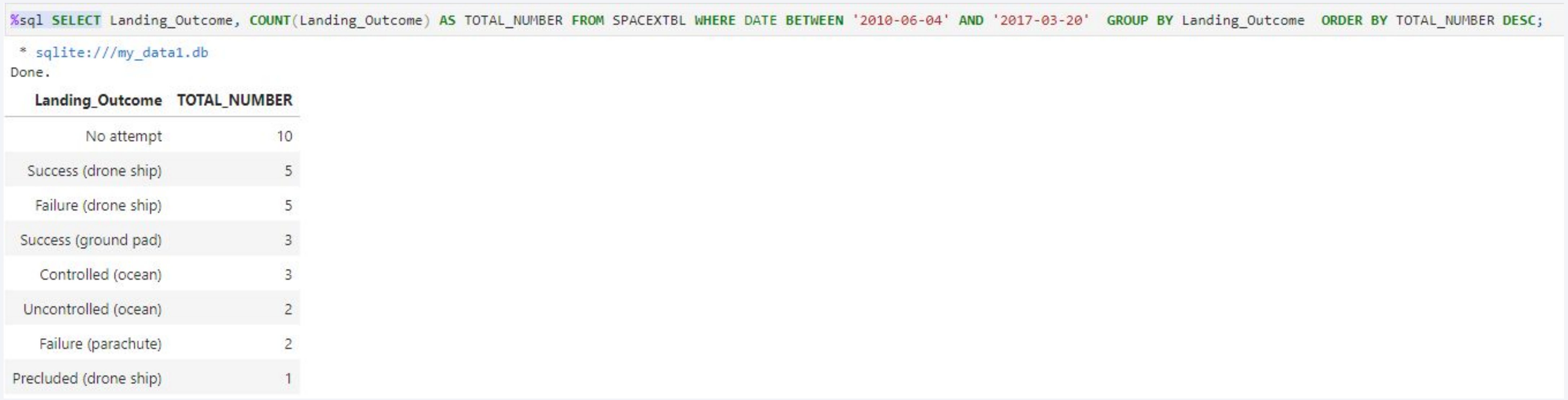


```
%sql select substr(Date, 6,2) as month,Landing_Outcome,Booster_Version,Launch_Site  from SPACEXTABLE where Landing_Outcome like '%Failure (drone Ship)%' and substr(Date,0,5)='2015'
```

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

- A combination of WHERE clause and AND logical operator is used here. The WHERE clause is used to check the different criteria.

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



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

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

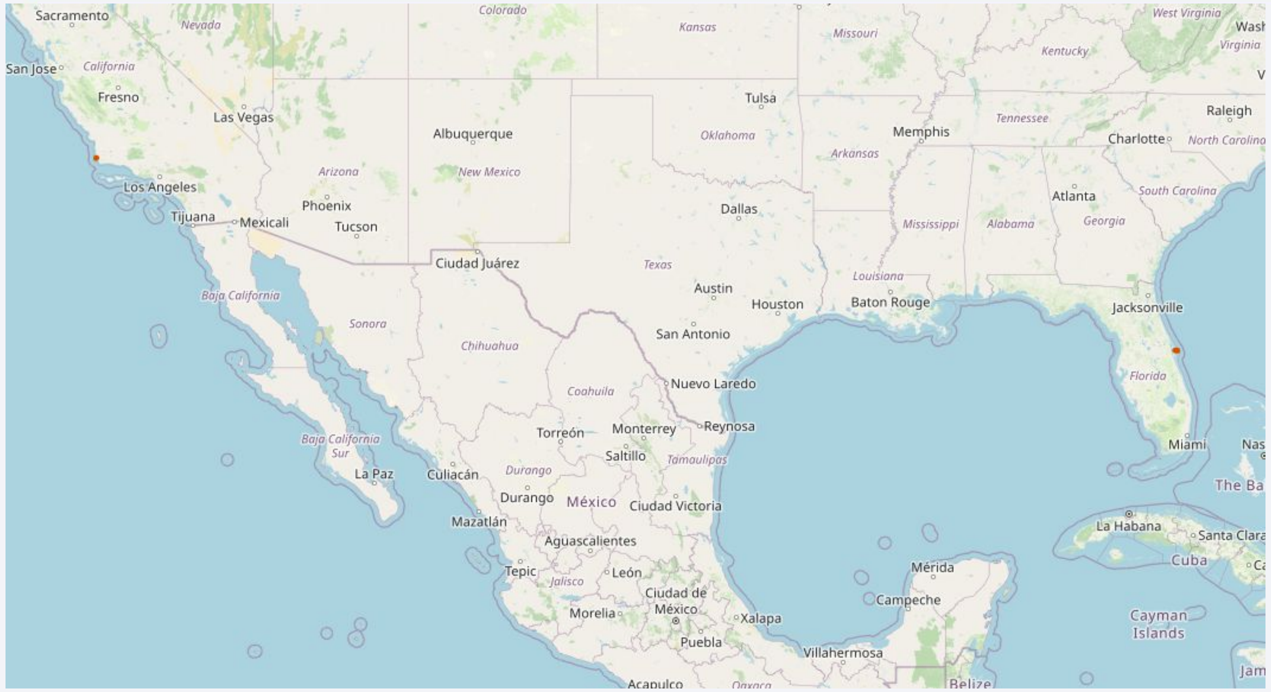
Landing_Outcome	TOTAL_NUMBER
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

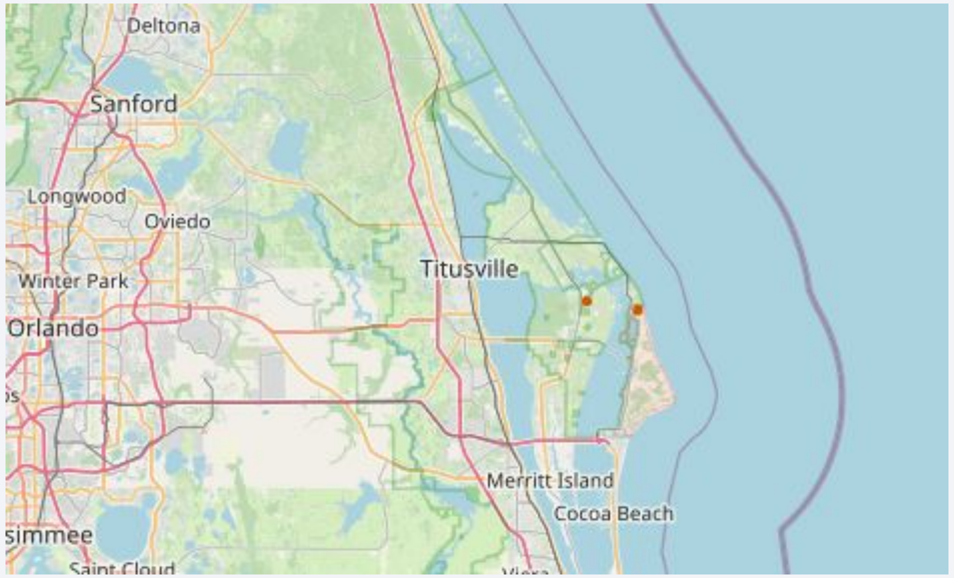
- Here the COUNT aggregate is used to calculate the total number of landing outcomes, and a BETWEEN statement is introduced to match the date criteria.
- Since an aggregate function was used, a group by statement is use together with an ordering by descending statement

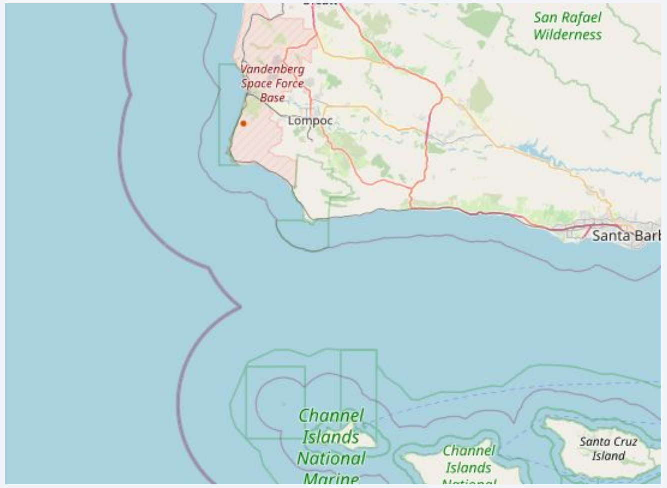
Section 3

Launch Sites Proximities Analysis

Overview of location of the 3 launch sites

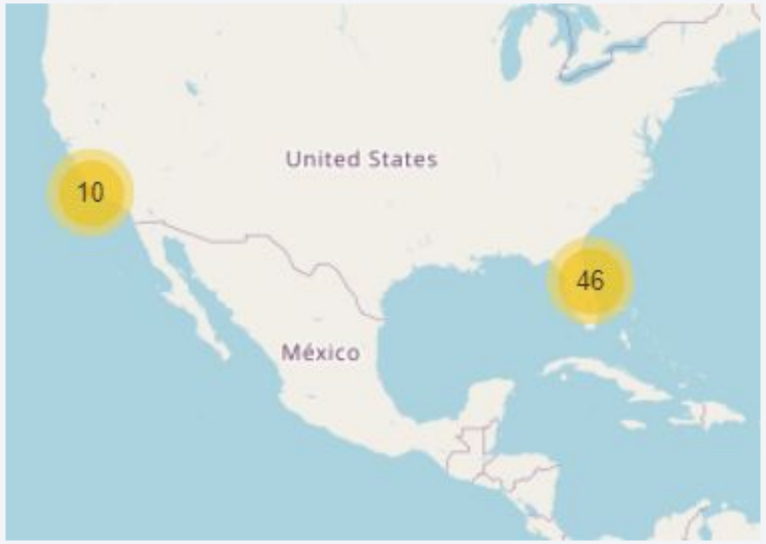


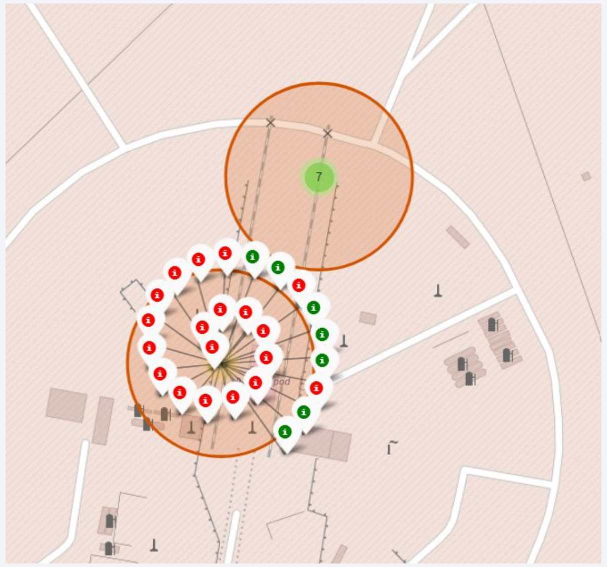


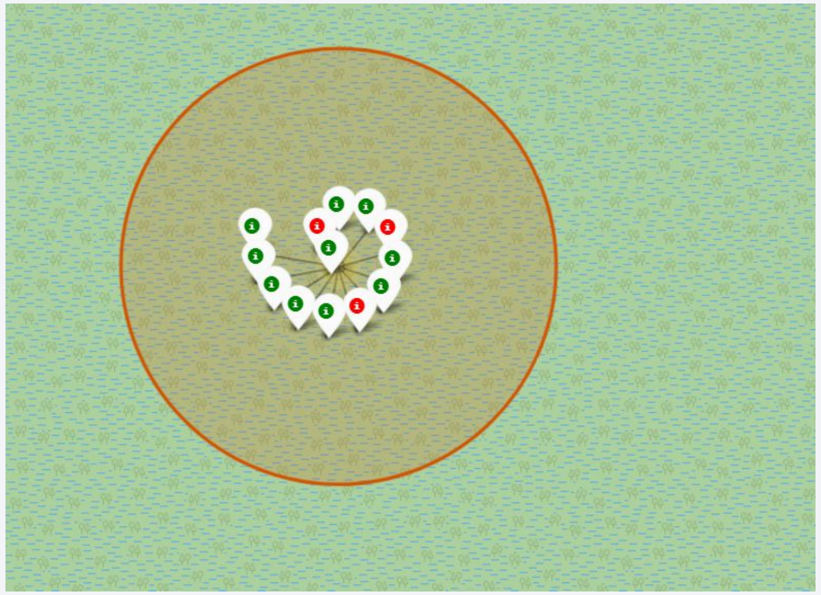


- These are the screenshots of the launch sites magnified and birds view and we can conclude that all the sites are located at the coastal regions

Indicators for successful and failed launches

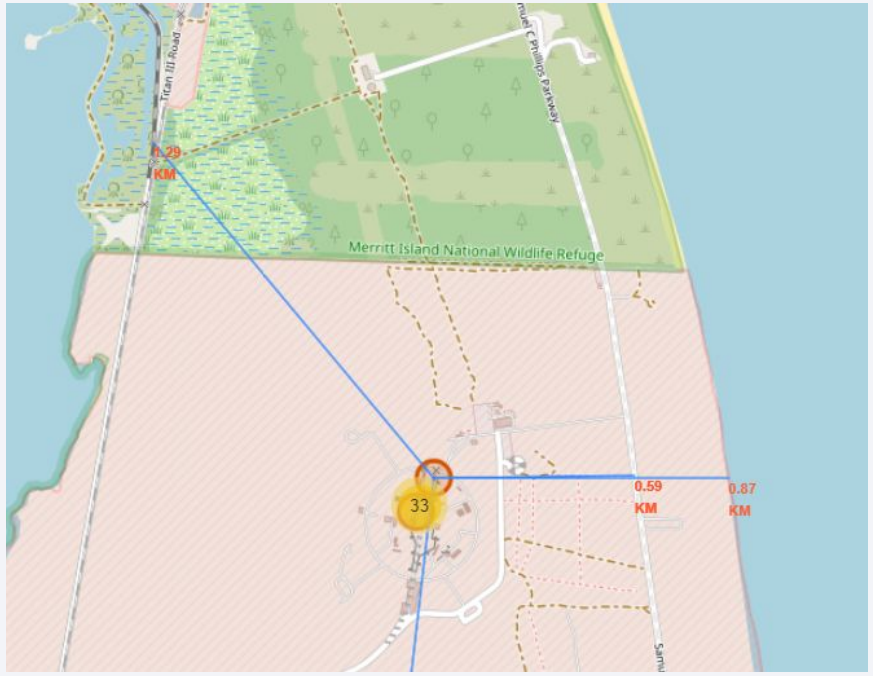


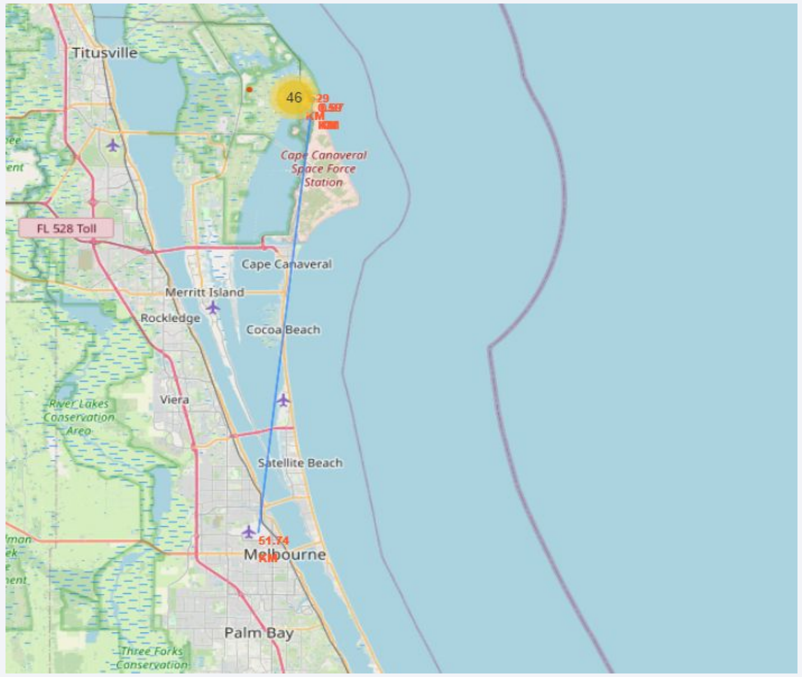




There are multiple launches and they have been grouped into clusters. The green ones represent the successful launches while the red ones represent the unsuccessful ones.

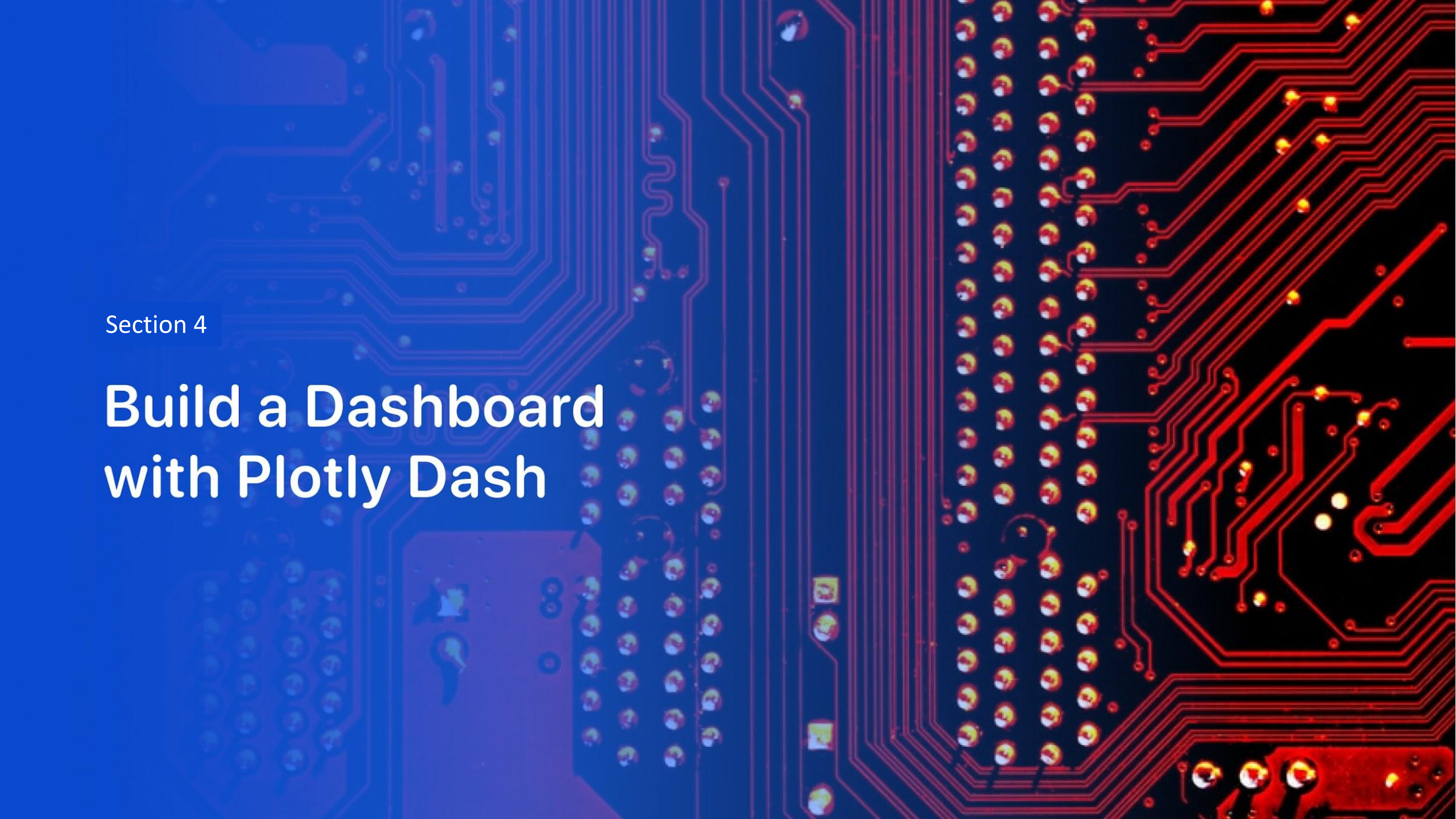
Proximities of the CCAFAS SLC-40 launch site





Here we can conclude the following:

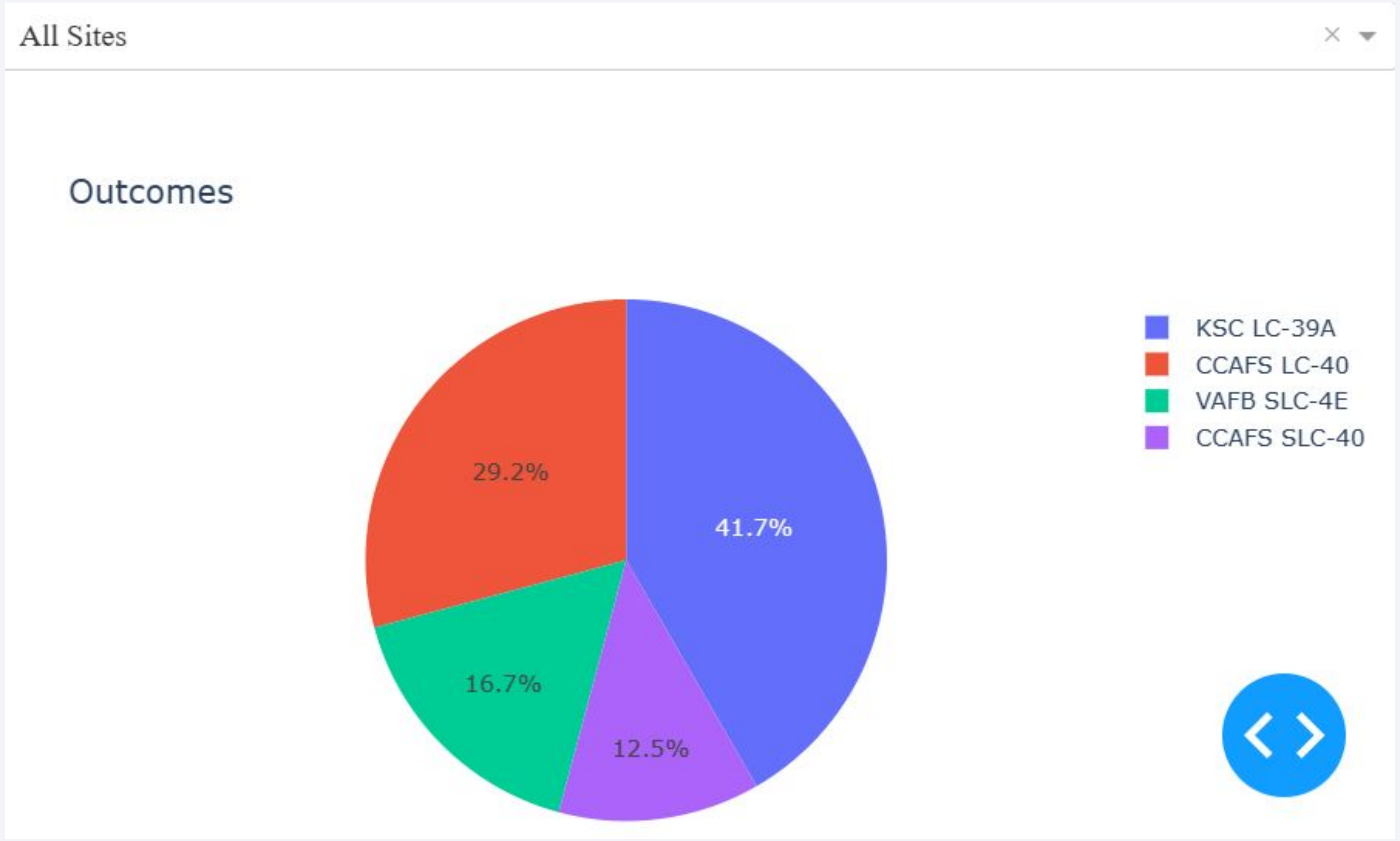
- The launch sites are close to coastline with the latter being only 0.87 km away
- The launch sites are close to highways with the nearest highway being 0.59 km away
- The nearest railway is 1.29 km away
- The launch sites are distant from cities with the specified CCAFS SLC-40 being 51.74 km away



Section 4

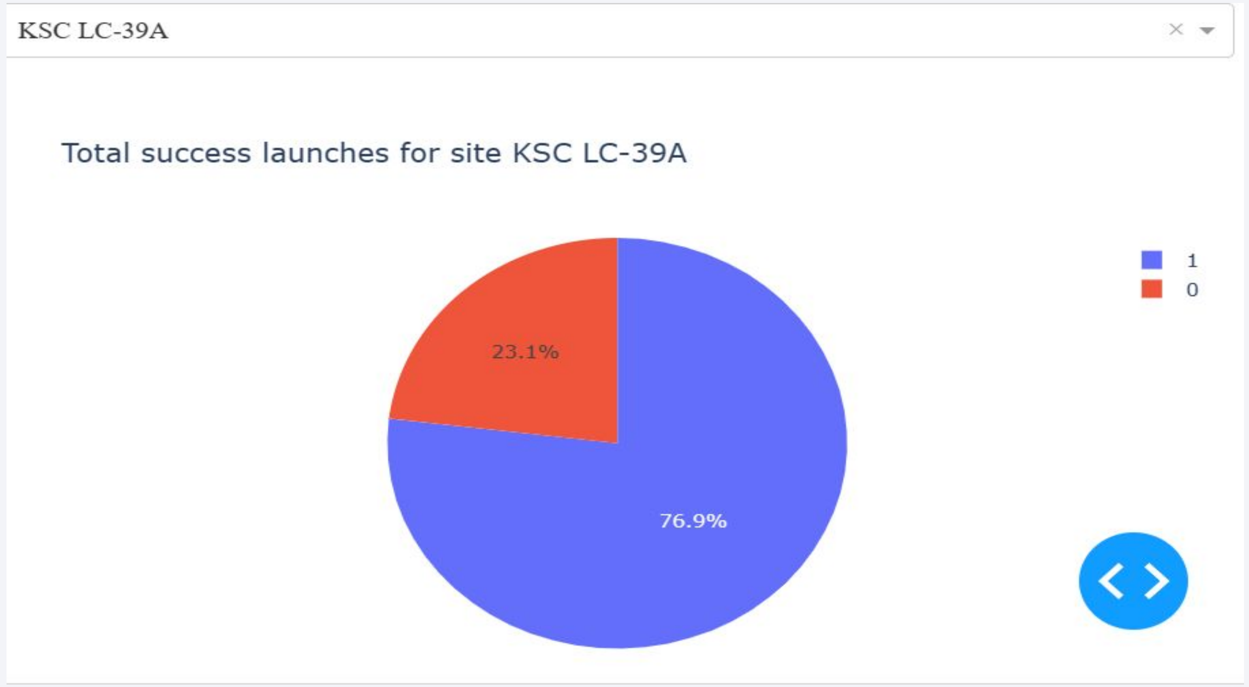
Build a Dashboard with Plotly Dash

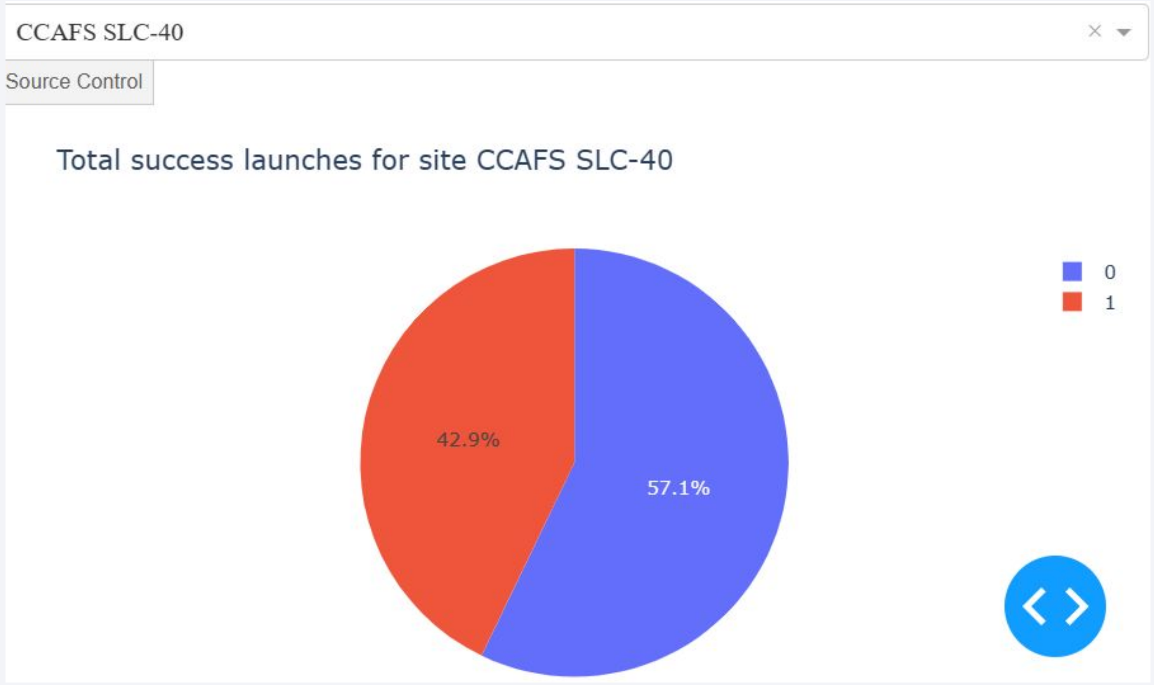
The success launch rate for all sites



- Here we can conclude that the booster that had the highest success rate is KSC LC -39A while the one that had the lowest success rate is CCAFS -SLC-40

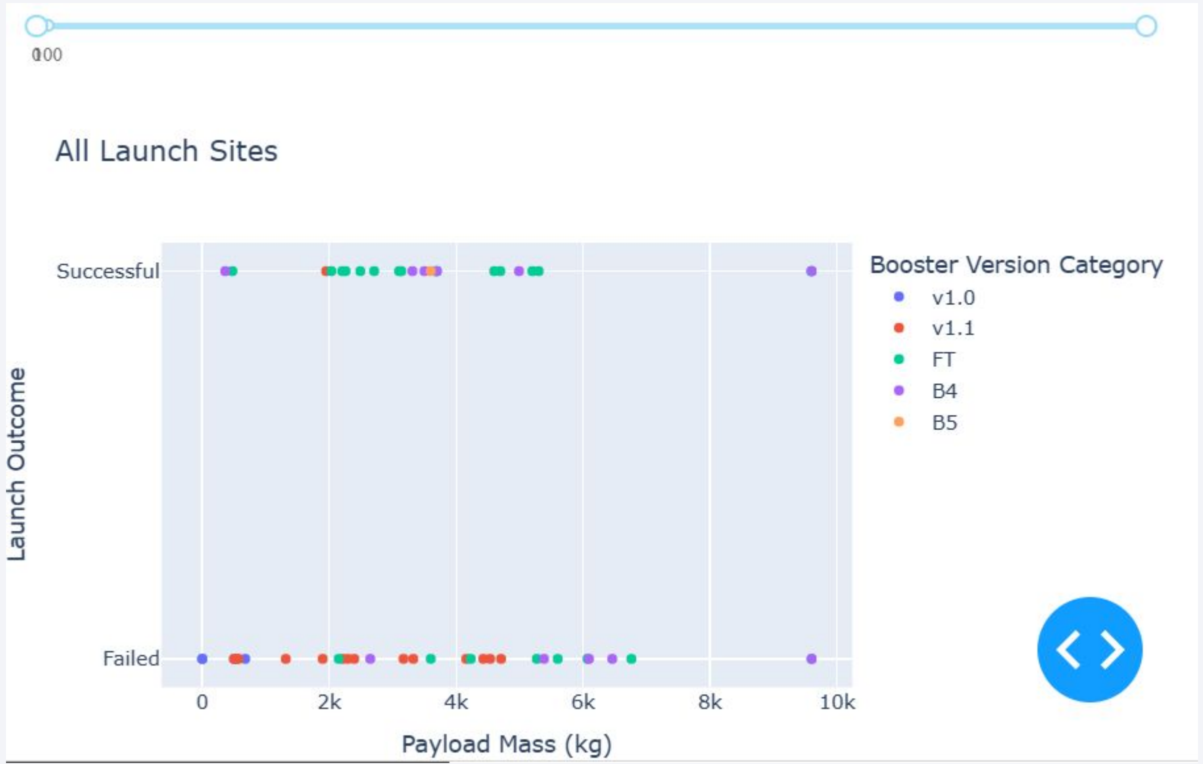
Percentage of success and failure for lowest and highest successes

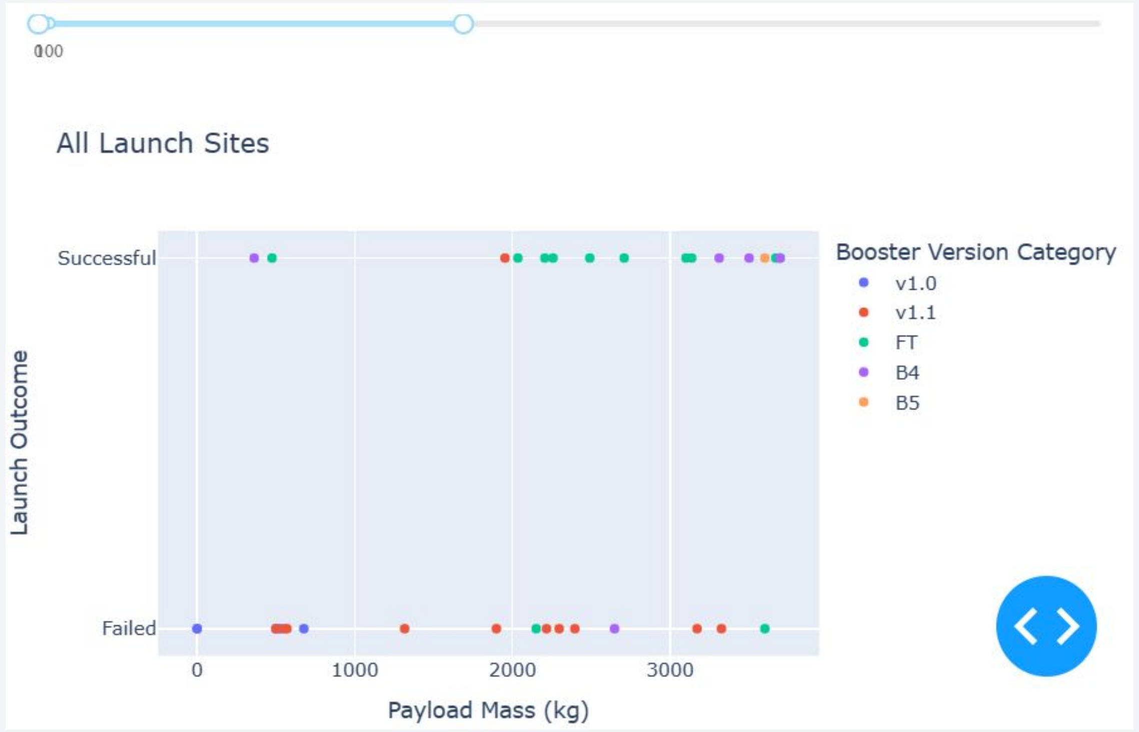




- The booster with the highest launch success rate is KSC LC-39A while the booster with the lowest launch success rate is CCAFS SLC-40

Correlation between payload mass and landing outcomes for different boosters



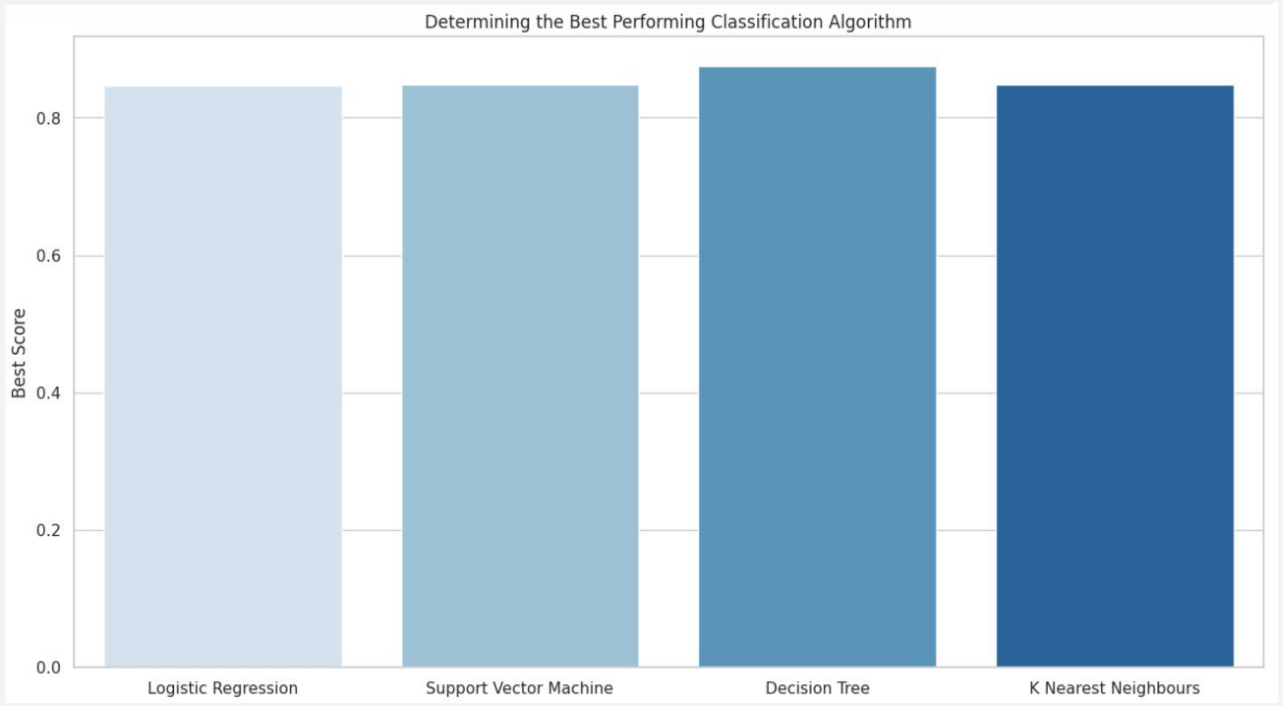


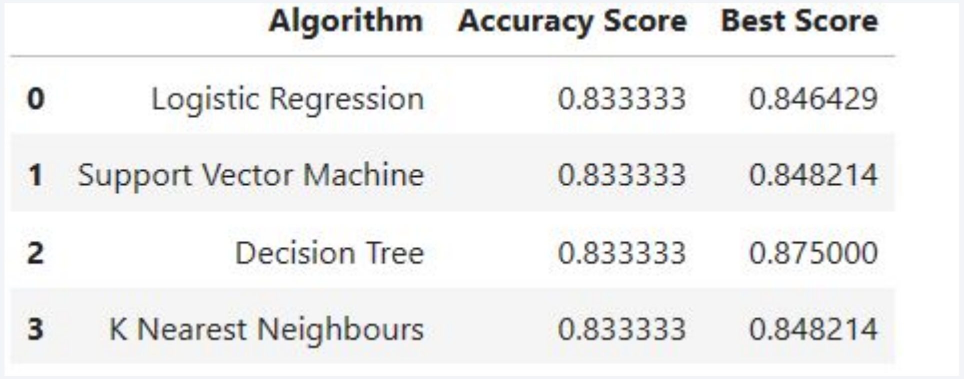
- Here we can conclude that there is a correlation between the landing outcomes and the payload for different ranges
- FT Booster for example has a higher success range between the range 2000 and 4000 kg

Section 5

Predictive Analysis (Classification)

Classification Accuracy

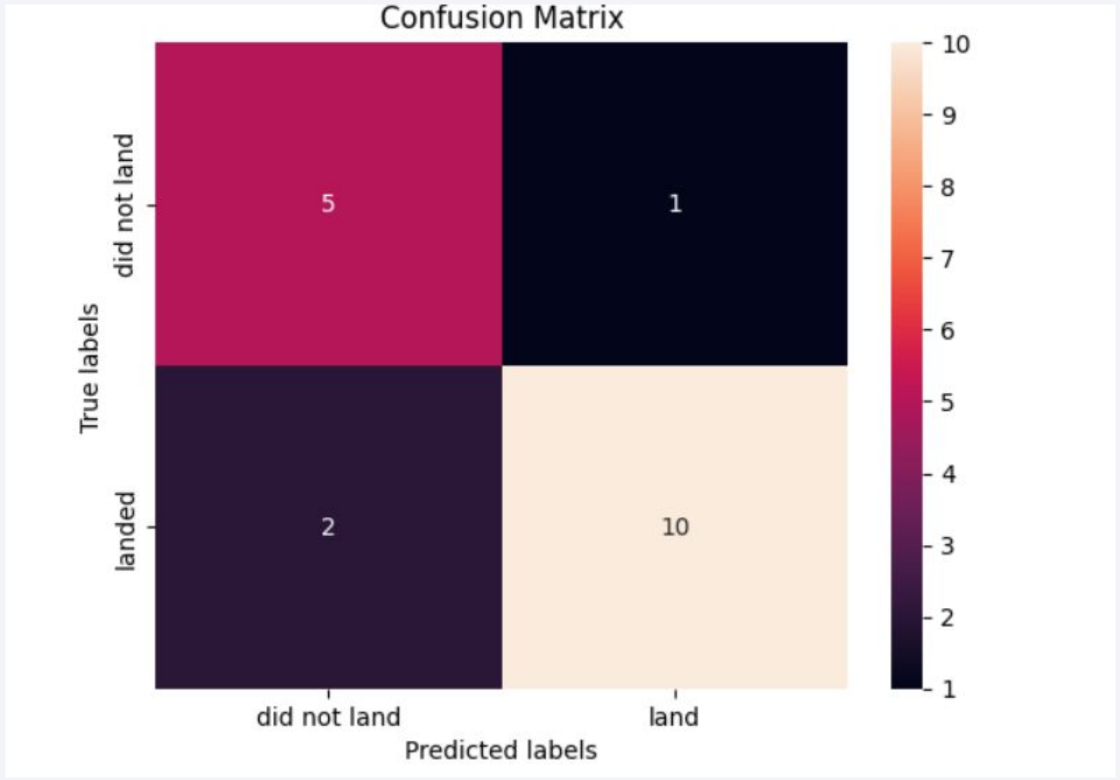




	Algorithm	Accuracy Score	Best Score
0	Logistic Regression	0.833333	0.846429
1	Support Vector Machine	0.833333	0.848214
2	Decision Tree	0.833333	0.875000
3	K Nearest Neighbours	0.833333	0.848214

- Plotting the accuracy score vs the classifiers we can see that the Decision tree has the highest score and hence is the best classifier

Confusion Matrix



- Out of the 18 predictions, 5 out of 6 true negatives were predicted and 10 out of 12 true positives were predicted
- Hence we can conclude that from the confusion matrix the Decision tree is the best classifier

Conclusions

- The success rate at a launch site improves as the number of flights increases, with early flights often failing. Increased experience leads to higher success rates.
  From 2010 to 2013, all landings failed, resulting in a 0% success rate during that period.
  Post-2013, the success rate generally trended upwards, although there were minor declines in 2018 and 2020.
  Since 2016, the likelihood of a successful landing has consistently exceeded 50%.

• The orbit types ES-L1, GEO, HEO, and SSO all have a perfect success rate of 100%.|
  The 100% success rate for GEO, HEO, and ES-L1 orbits is due to having only one flight each into these orbits.
  The 100% success rate for SSO is particularly notable, with five successful flights.
  The orbit types PO, ISS, and LEO demonstrate higher success rates with heavier payloads.
  VLEO (Very Low Earth Orbit) launches also involve heavier payloads, which aligns with expectations.

• Launch site KSC LC-39 A achieved the highest number of successful launches, accounting for 41.7% of all successful launches. It also boasted the highest success rate, with 76.9% of its launches being successful.

• The success rate for large payloads (over 4000kg) is lower compared to lighter payloads.

• The Decision Tree model is the top-performing classification model, achieving an accuracy of 94.44%.

Thank you!