IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Muhammad Ammar Shahzad
10/Sep/2021

Outline

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

Executive Summary

❑Summary of methodologies

- Data collection
- Data Wrangling
- EDA with data visualization
- EDA with SQL
- Building an interactive map with Folium
- Building a dashboard with plotly dash
- Predictive analysis for each classification model

❑Summary of results

- Exploratory data analysis along with interactive visualizations
- Results of Predictive analysis

Introduction

- Project background and context

We predicted if the Falcon 9 first stage will land successfully. SpaceX advertises Falcon 9 rocket launches on its website, with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is 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. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.

- Problems you want to find answers
  - With what factors, the rocket will land successfully?
  - The effect of each relationship of rocket variables on outcome.
  - Conditions which will aid SpaceX have to achieve the best results

Section 1

Methodology

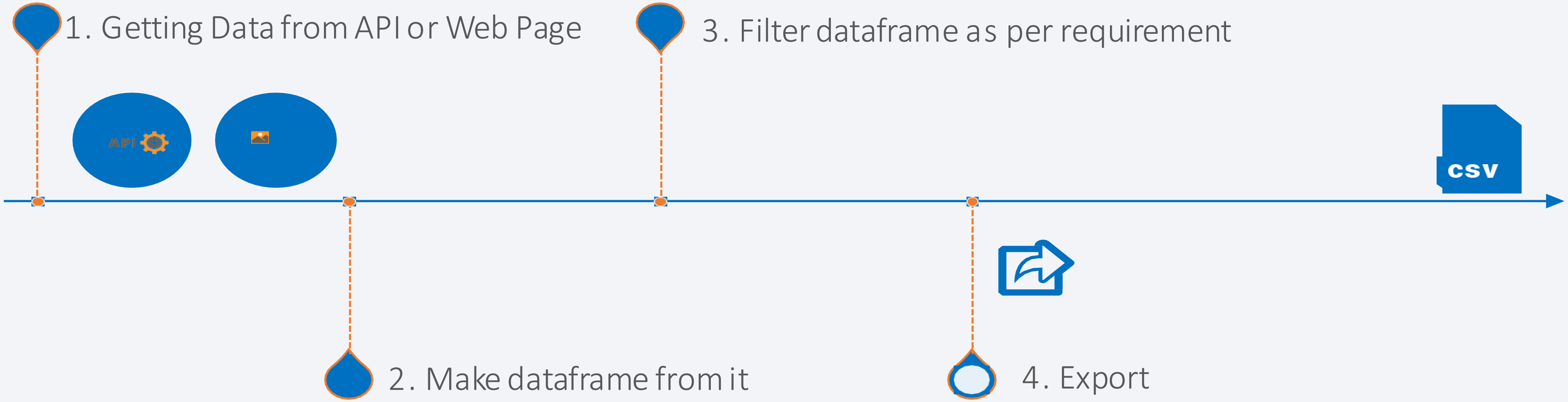
Methodology

- Data collection methodology
  - Via SpaceX Rest API
  - Web Scrapping from Wikipedia
- Perform data wrangling
  - One hot encoding data fields for machine learning and dropping irrelevant columns (Transforming data for Machine Learning)
- Perform exploratory data analysis (EDA) using visualization and SQL
  - Scatter and bar graphs to show patterns between data
- Perform interactive visual analytics using Folium and Plotly Dash
  - Using Folium and Plotly Dash Visualizations
- Perform predictive analysis using classification models Dash
  - Build and evaluate classification models

Data Collection

- We worked with SpaceX launch data that is gathered from the SpaceX REST API.
- This API will give us data about launches, including information about the rocket used, payload delivered,

  launch specifications, landing specifications, and landing outcome.
- Our goal is to use this data to predict whether SpaceX will attempt to land a rocket or not.
- The SpaceX REST API endpoints, or URL, starts with api.spacexdata.com/v4/.
- Another popular data source for obtaining Falcon 9 Launch data is web scraping Wikipedia using BeautifulSoup.



1. Getting Data from API or Web Page

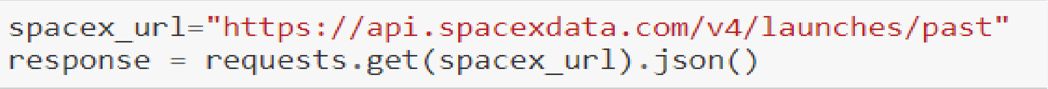
2. Make dataframe from it

3. Filter dataframe as per requirement

4. Export

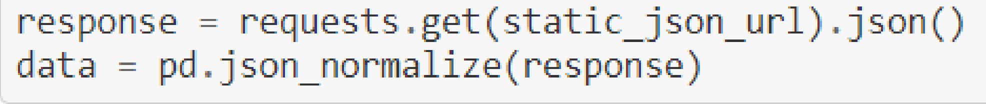
Data Collection – SpaceX API

1 .Getting Response from API



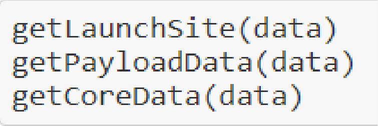
```
spacex_url="https://api.spacexdata.com/v4/launches/past"
response = requests.get(spacex_url).json()
```

2. Converting Response to a .json file

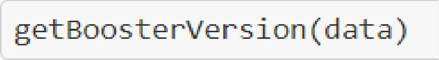


```
response = requests.get(static_json_url).json()
data = pd.json_normalize(response)
```

3. Apply custom functions to clean data

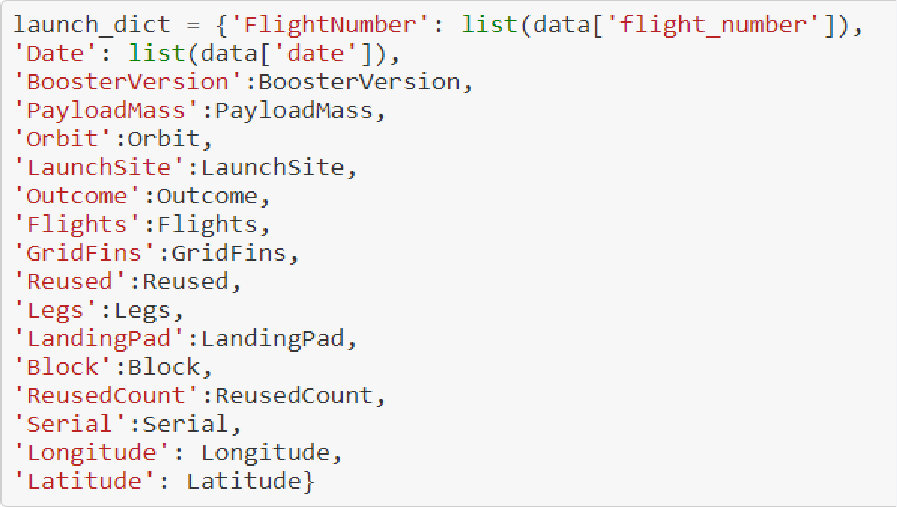


```
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
```

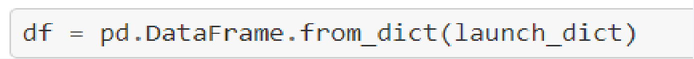


```
getBoosterVersion(data)
```

4. Assign list to dictionary then dataframe

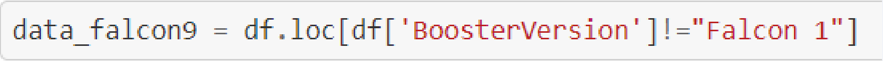


```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,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

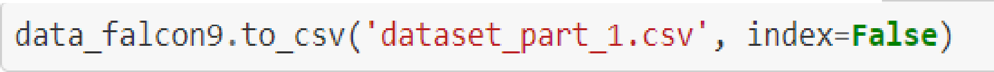


```
df = pd.DataFrame.from_dict(launch_dict)
```

5. Filter dataframe and export to flat file (.csv)



```
data_falcon9 = df.loc[df['BoosterVersion']!="Falcon 1"]
```

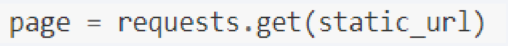


```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Github Link

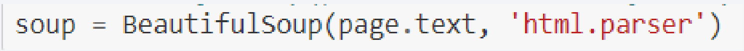
Data Collection - Scraping

1 .Getting Response from HTML



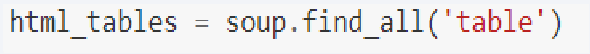
```
page = requests.get(static_url)
```

2. Creating BeautifulSoup Object



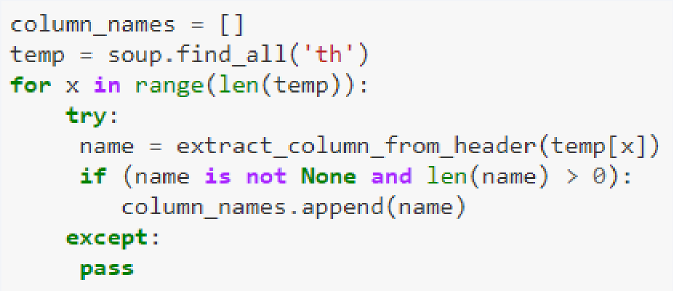
```
soup = BeautifulSoup(page.text, 'html.parser')
```

3. Finding tables



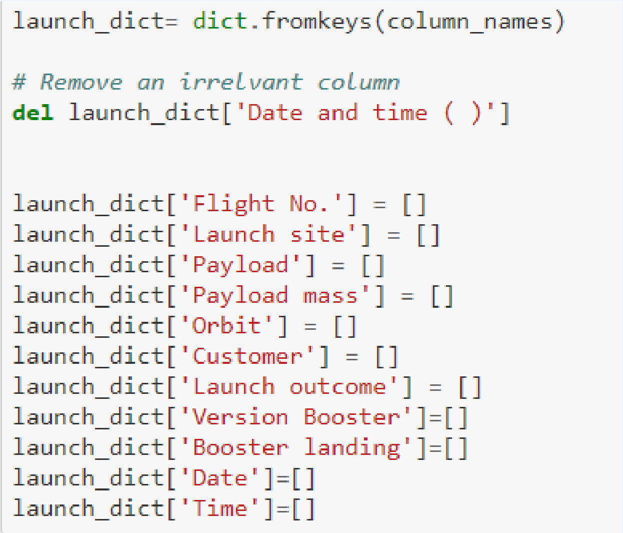
```
html_tables = soup.find_all('table')
```

4. Getting column names



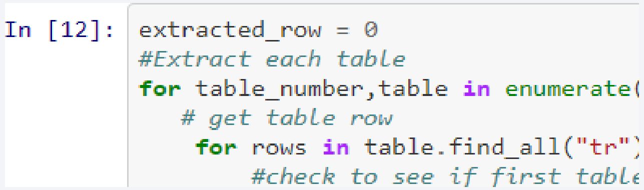
```
column_names = []
temp = soup.find_all('th')
for x in range(len(temp)):
    try:
     name = extract_column_from_header(temp[x])
     if (name is not None and len(name) > 0):
        column_names.append(name)
    except:
     pass
```

5. Creation of dictionary



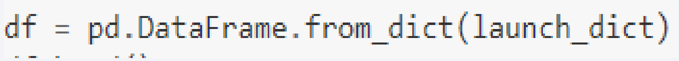
```
launch_dict= dict.fromkeys(column_names)

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


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'] = []
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

6. Appending data to keys



```
In [12]:  extracted_row = 0
          #Extract each table
          for table_number,table in enumerate(
             # get table row
              for rows in table.find_all("tr")
                  #check to see if first table
```

7. Converting dictionary to dataframe

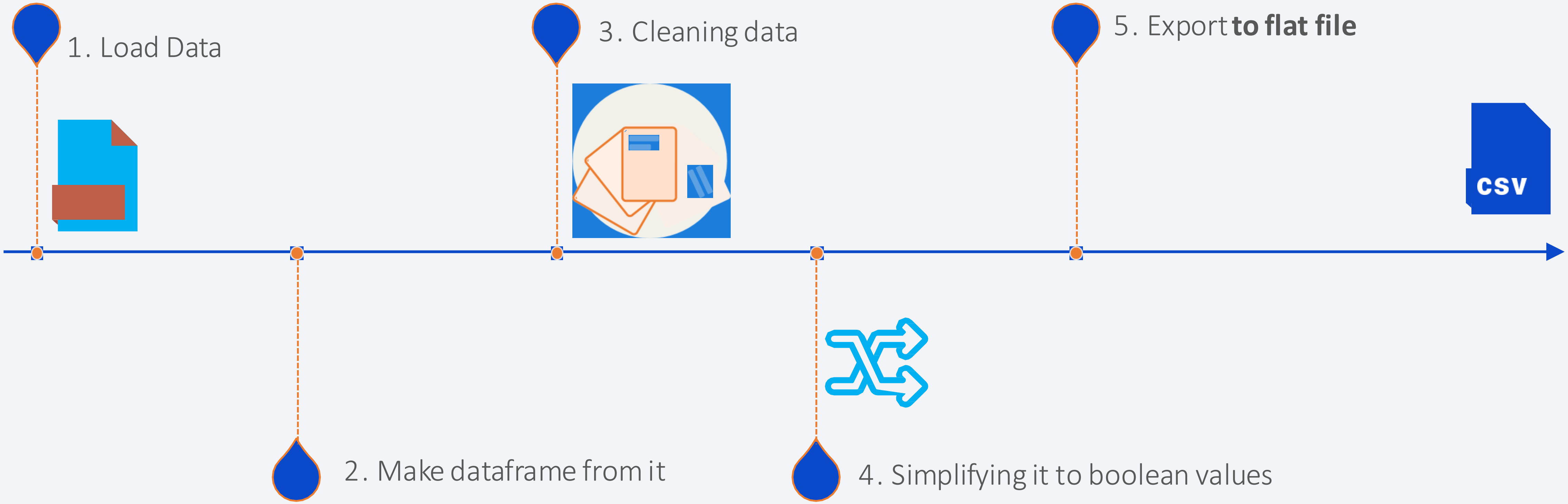


```
df = pd.DataFrame.from_dict(launch_dict)
```

github link

Data Wrangling

Data wrangling is the process of cleaning and unifying messy and complex data sets for easy access and analysis.



1. Load Data

2. Make dataframe from it

3. Cleaning data

4. Simplifying it to boolean values

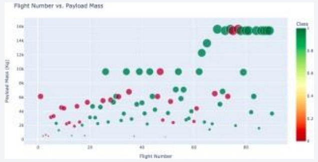
5. Export **to flat file**

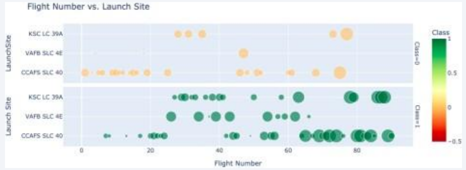
EDA with Data Visualization

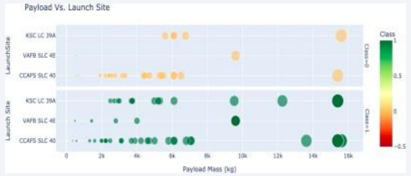
Scatter Graphs Drawn:

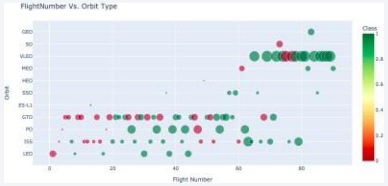
- Payload and Flight Number
- Flight Number and Launch Site
- Payload and Launch Site
- Flight Number and Orbit Type
- Payload and Orbit Type

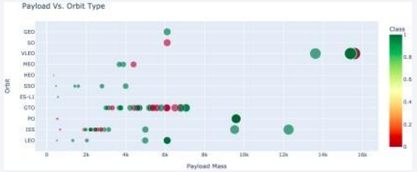
Scatter plots show dependency of attributes on each other. Once a pattern is determined from the graphs it's very easy to predict which factors will lead to maximum probability of success in both outcome and landing.







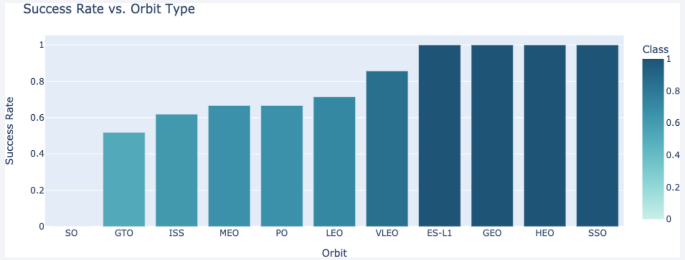




Bar Graph Drawn:

Success Rate VS. Orbit Type

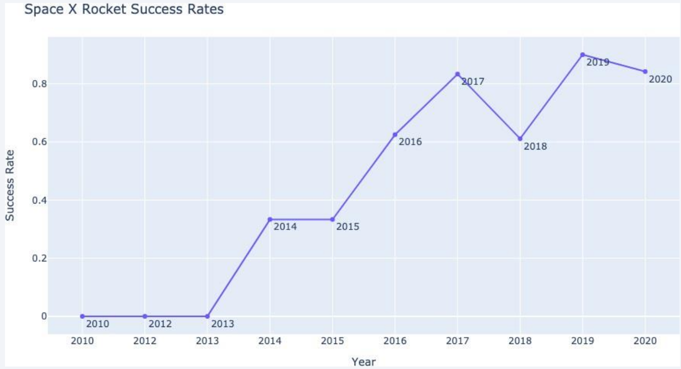
Bar graphs are easiest to interpret a relationship between attributes. Via this bar graph we can easily determine which orbits have the highest probability of success.



Line Graph Drawn:

Launch Success Yearly Trend

Line graphs are useful in that they show trends clearly and can aid in predictions for the future.



EDA with SQL

SQL is an indispensable tool for Data Scientists and analysts as most of the real-world data is stored in databases. It’s not only the standard language for Relational Database operations, but also an incredibly powerful tool for analyzing data and drawing useful insights from it. Here we use IBM’s Db2 for Cloud, which is a fully managed SQL Database provided as a service.

```
!pip install sqlalchemy==1.3.9
!pip install ibm_db_sa
!pip install ipython-sql
%load_ext sql

%sql ibm_db_sa://my-username:my-password@my-hostname:my-port/my-db-name
%sql <your query>
```

Github url

We performed SQL queries to gather information from given dataset :

- Displaying the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string 'CCA'
- Displaying the total payload mass carried by boosters launched by NASA (CRS)
- Displaying average payload mass carried by booster version F9 v1.1
- Listing the date where the successful landing outcome in drone ship was achieved
- Listing the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000
- Listing the total number of successful and failure mission outcomes
- Listing the names of the booster_versionswhich have carried the maximum payload mass
- Listing the failed landing_outcomesin drone ship, their booster versions, and launch site names for the year 2015
- Ranking 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

Build a Dashboard with Plotly Dash

Used Python Anywhere to host the website live 24/7 so your can play around with the data and view the data

- The dashboard is built with Flask and Dash web framework.

- Graphs
  - Pie Chart showing the total launches by a certain site/all sites
  - *display relative proportions of multiple classes of data.*
  - *size of the circle can be made proportional to the total quantity it represents.*

Github Dashboard Plotly source code

Scatter Graph showing the relationship with Outcome and Payload Mass (Kg) for the different Booster Versions

- It shows the relationship between two variables.
- It is the best method to show you a non-linear pattern.
- The range of data flow, i.e. maximum and minimum value, can be determined.
- Observation and reading are straightforward.

Predictive Analysis (Classification)

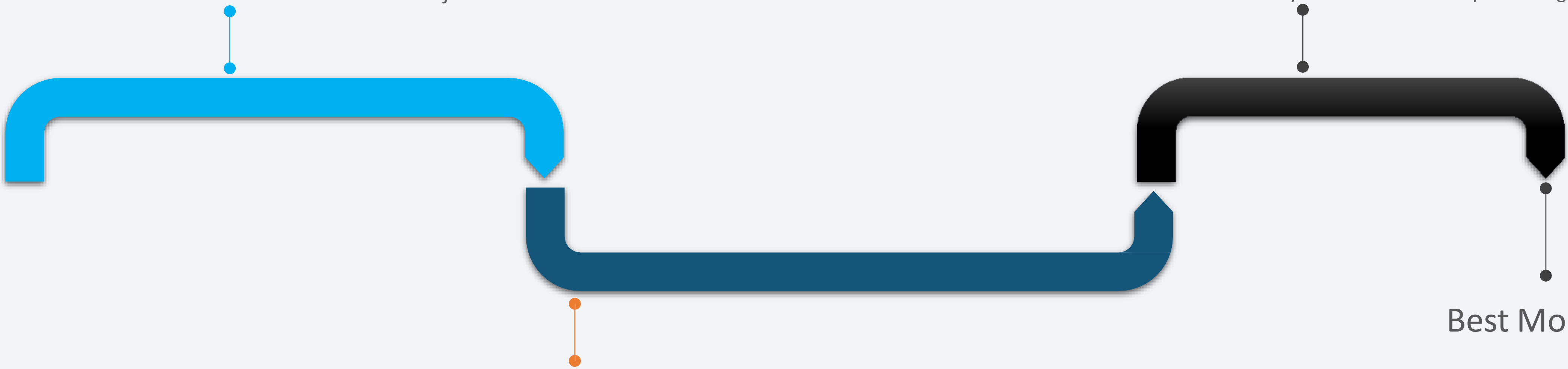
Building Model

- Load our feature engineered data into dataframe
- Transform it into NumPy arrays
- Standardize and transform data
- Split data into training and testdata sets
- Check how many test samples has been created
- List down machine learning algorithms we want to use
- Set our parameters and algorithms to GridSearchCV
- Fit our datasets into the GridSearchCV objects and train our model

```
y = data['Class'].to_numpy()
transform = preprocessing.StandardScaler()
X = transform.fit(X).transform(X)
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size=0.2, random_state=2)
Y_test.shape
```

Finding Best Performing Classification Model

- The model with best accuracy score wins the best performing model



Best Model

Evaluating Model

- Check accuracy for each model
- Get best hyperparameters for each type of algorithms
- Plot Confusion Matrix

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

[Github link]

Results

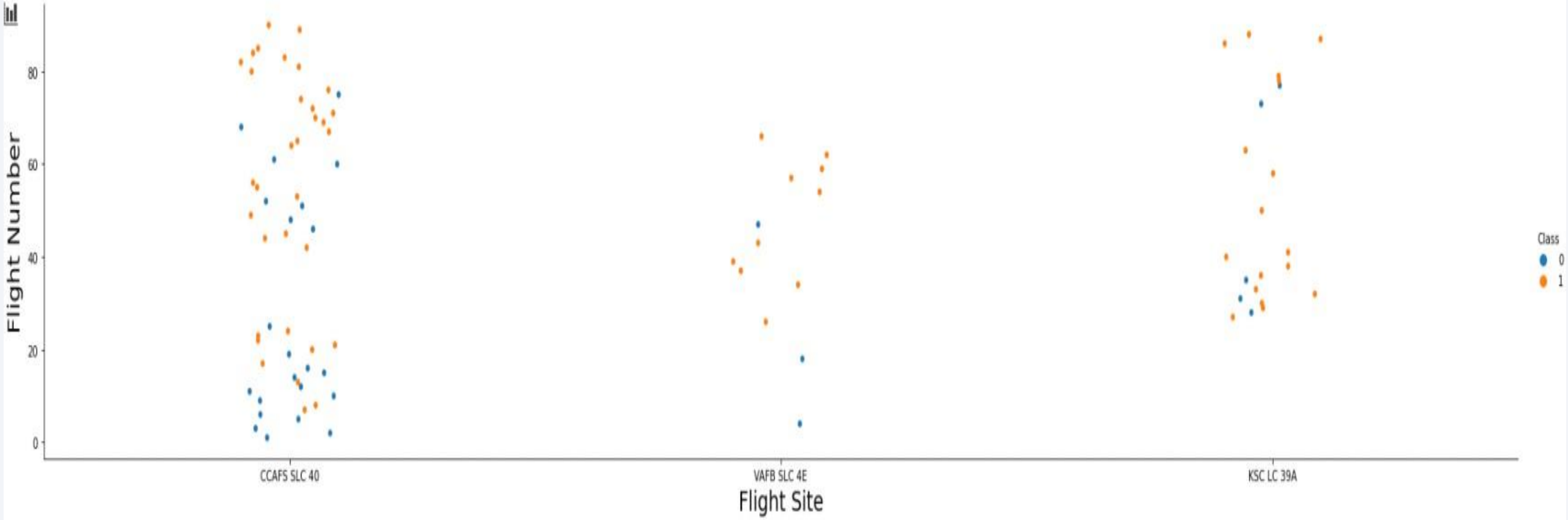
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

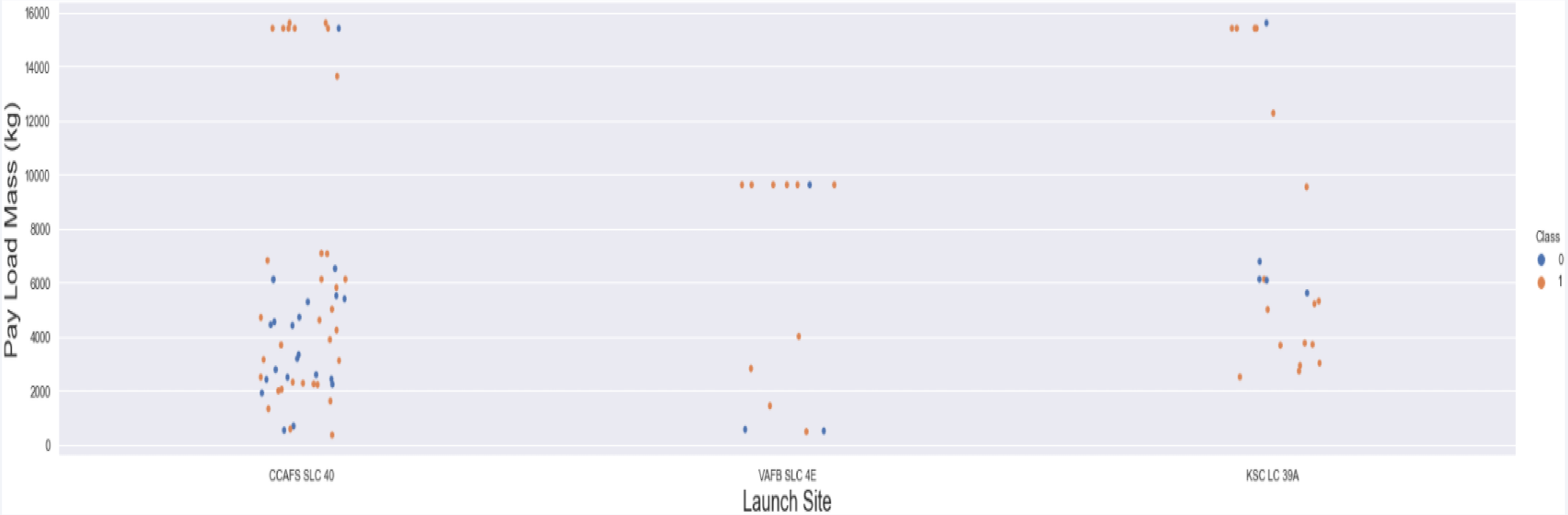
Flight Number vs. Launch Site

- With higher flight numbers (greater than 30) the success rate for the Rocket is increasing.



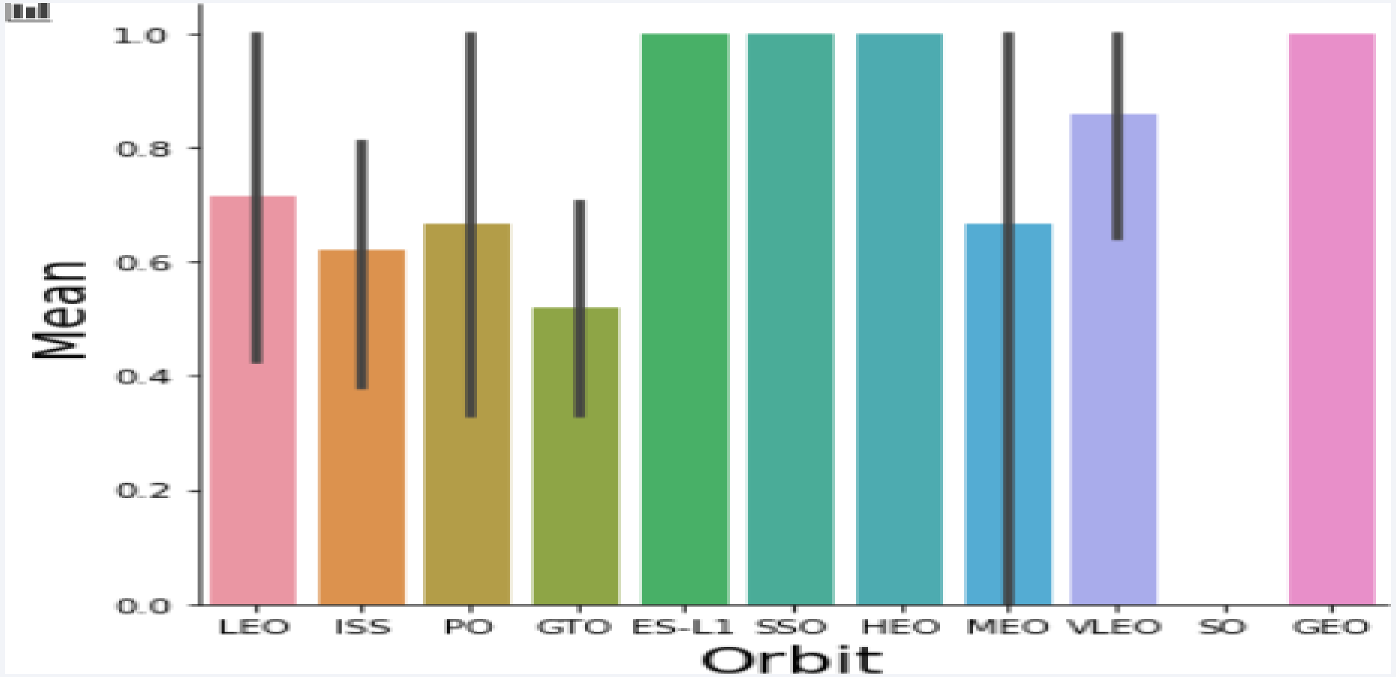
Payload vs. Launch Site

- The greater the payload mass (greater than 7000 Kg) higher the success rate for the Rocket. But there's no clear pattern to take a decision, if the launch site is dependent on Pay Load Mass for a success launch.



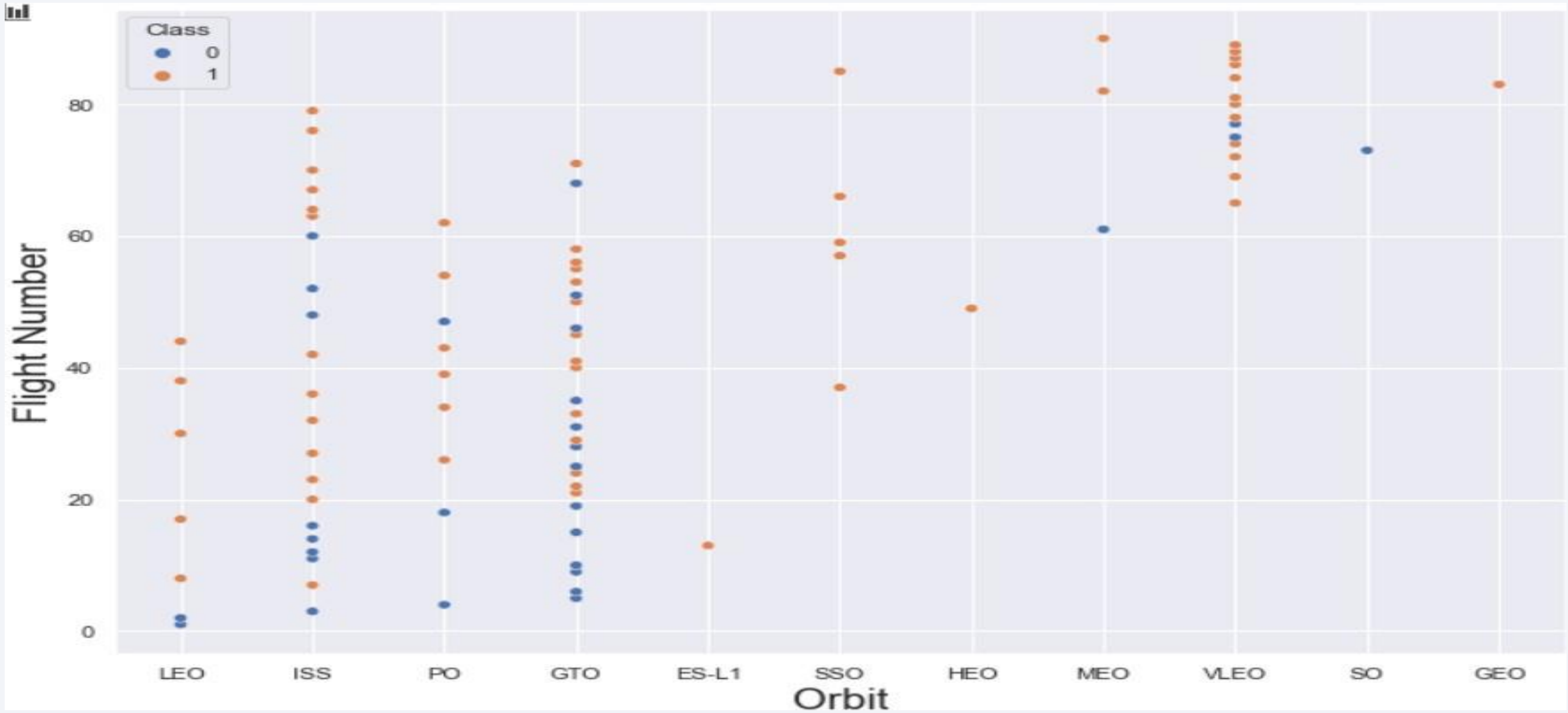
Success Rate vs. Orbit Type

ES-L1, GEO, HEO, SSO has highest Success rates



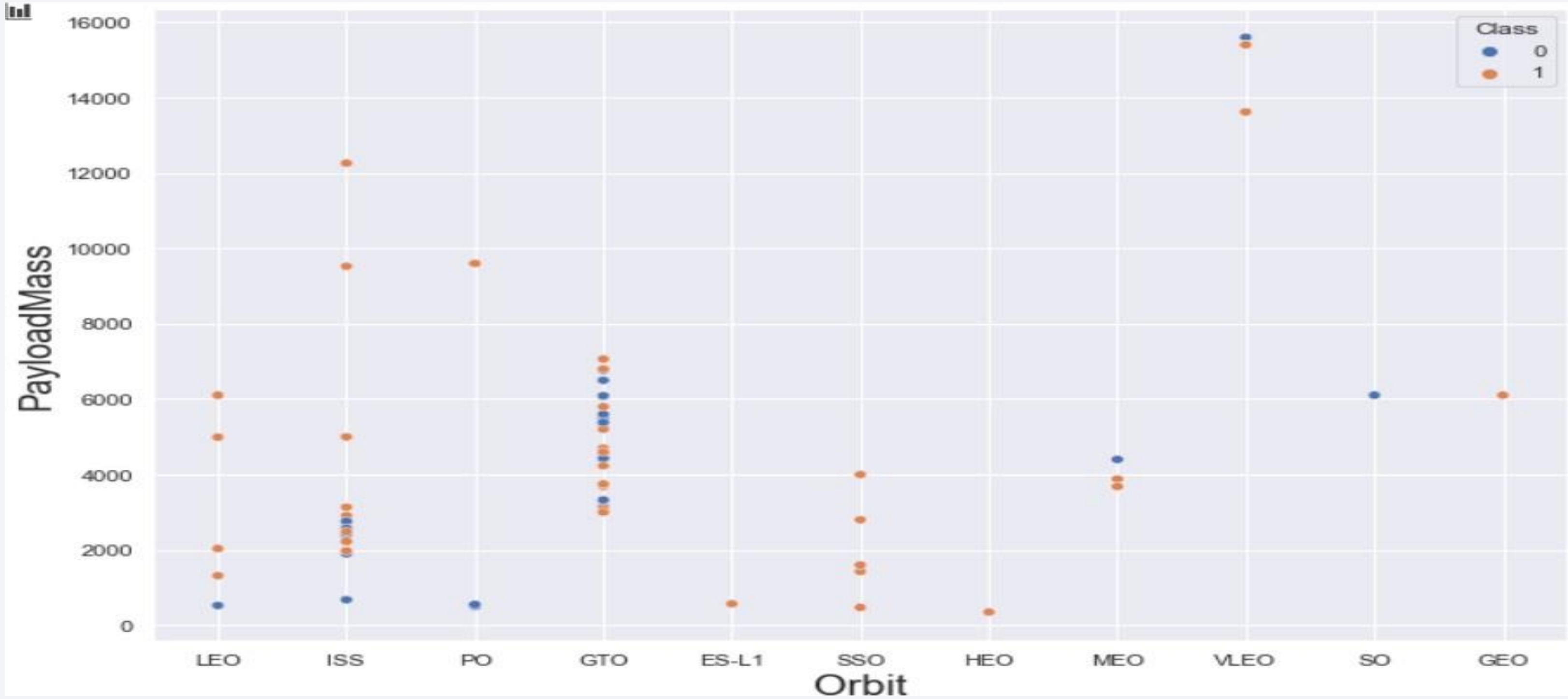
Flight Number vs. Orbit Type

- We see that for **LEO** orbit the success increases with the number of flights
- On the other hand, there seems to be no relationship between flight number and the **GTO** orbit.



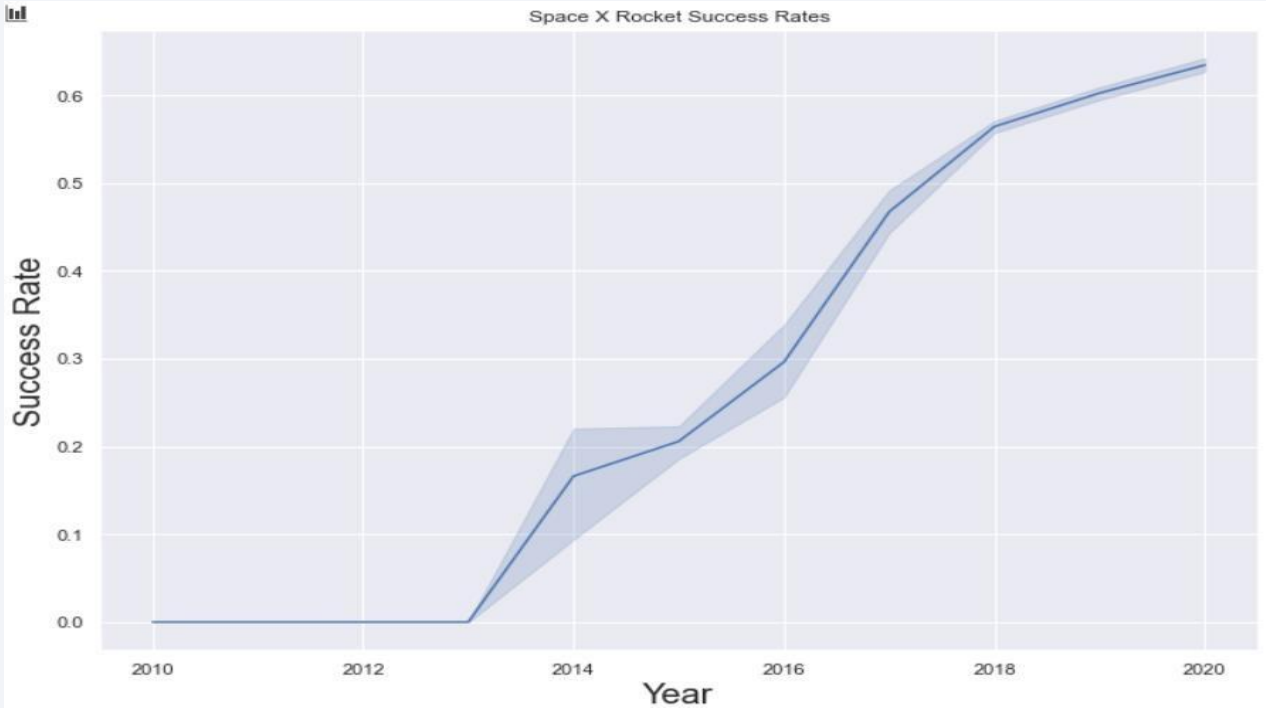
Payload vs. Orbit Type

- We observe that heavy payloads have a negative influence on MEO, GTO, VLEO orbits
- Positive on LEO, ISS orbits

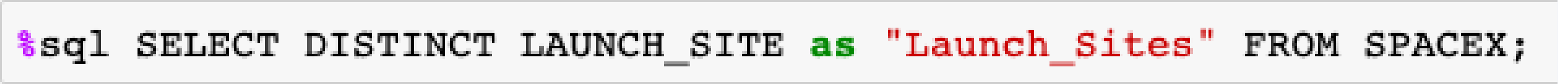


Launch Success Yearly Trend

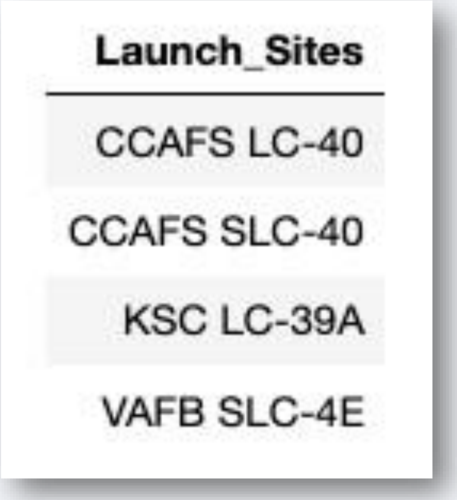
We can observe that the sucess rate since 2013 kept increasing relatively though there is slight dip after 2019.



All Launch Site Names



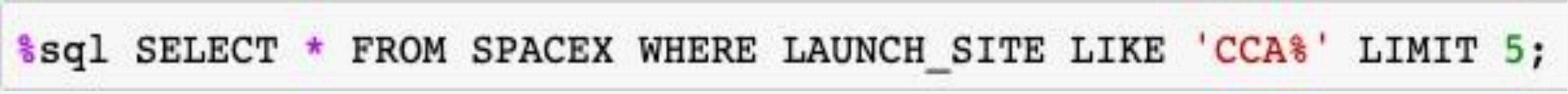
```
%sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEX;
```



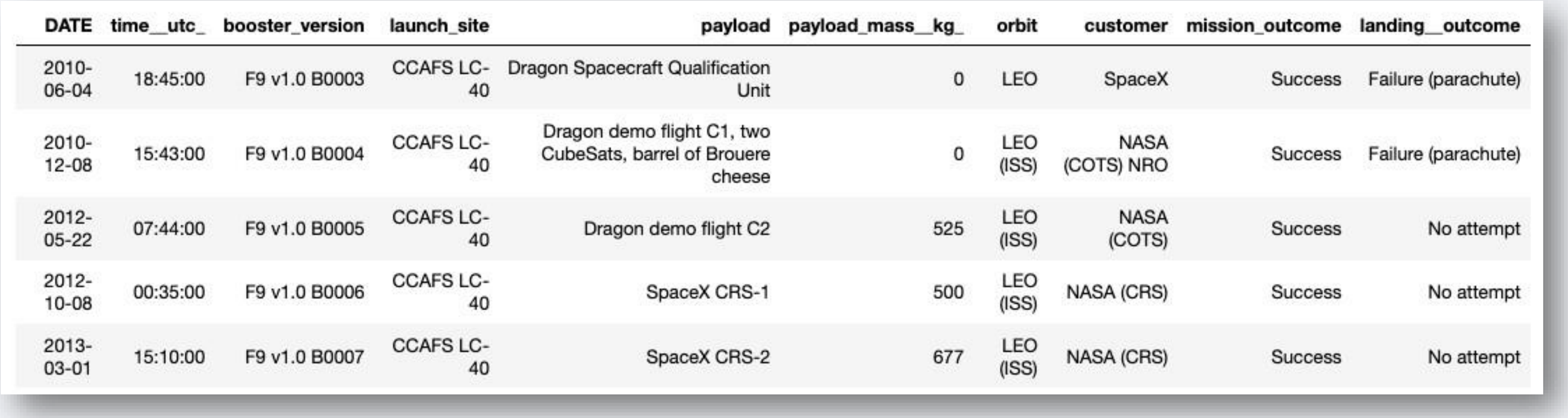
Launch_Sites
CCAFS LC-40
CCAFS SLC-40
KSC LC-39A
VAFB SLC-4E

Using the word DISTINCT in the query we pull unique values for Launch_Site column from table SPACEX

Launch Site Names Begin with 'CCA'

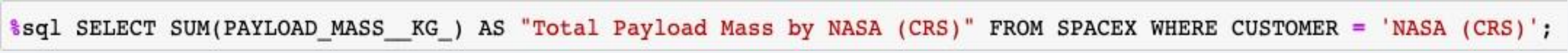


```
%sql SELECT * FROM SPACEX WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
```

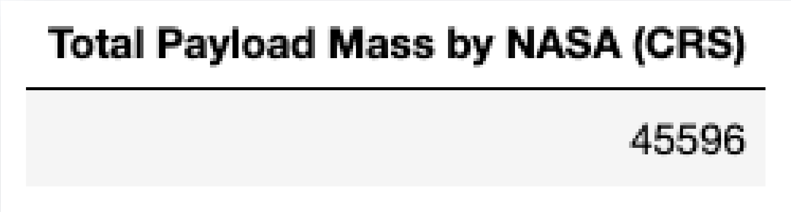


DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	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

Total Payload Mass

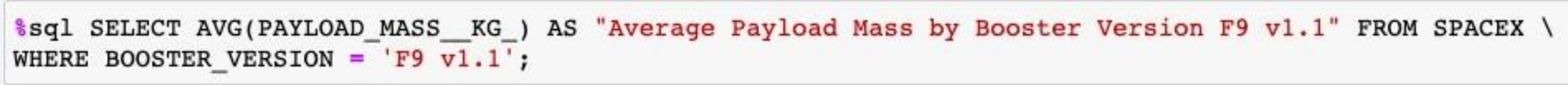


```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS "Total Payload Mass by NASA (CRS)" FROM SPACEX WHERE CUSTOMER = 'NASA (CRS)';
```

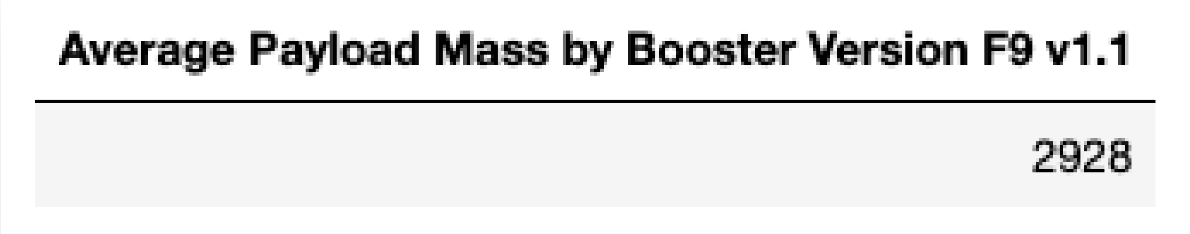


Total Payload Mass by NASA (CRS)
45596

Average Payload Mass by F9 v1.1

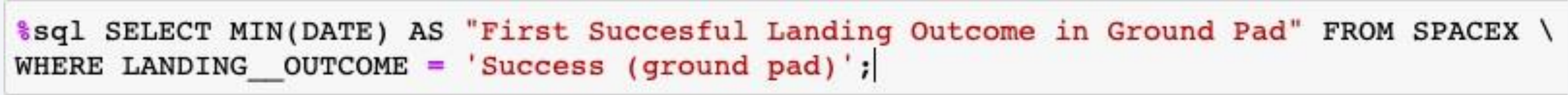


```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS "Average Payload Mass by Booster Version F9 v1.1" FROM SPACEX \
WHERE BOOSTER_VERSION = 'F9 v1.1';
```

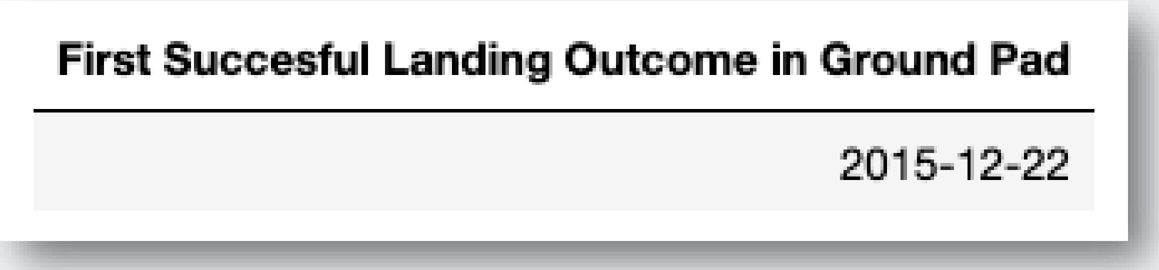


Average Payload Mass by Booster Version F9 v1.1
2928

First Successful Ground Landing Date



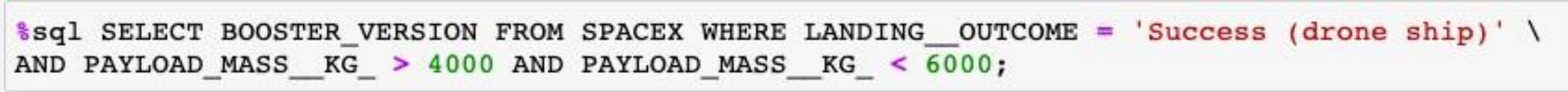
```
%sql SELECT MIN(DATE) AS "First Succesful Landing Outcome in Ground Pad" FROM SPACEX \
WHERE LANDING__OUTCOME = 'Success (ground pad)';
```



First Succesful Landing Outcome in Ground Pad
2015-12-22

Using the function MIN works out the minimum date in the column Date and WHERE clause filters the data to only perform calculations on Landing_Outcome with values "Success (ground pad)".

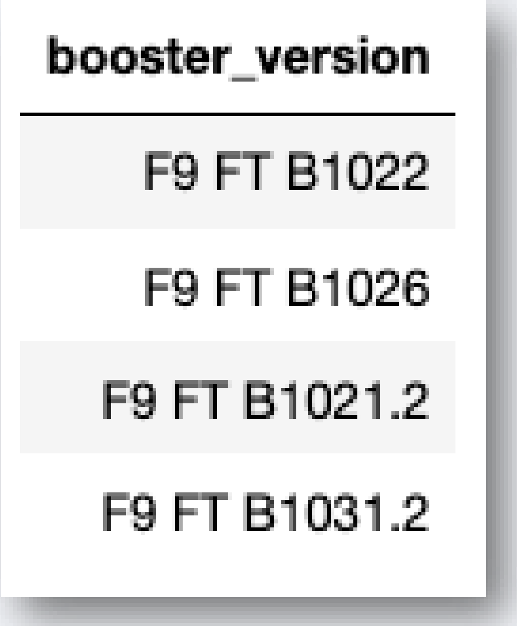
Successful Drone Ship Landing with Payload between 4000 and 6000



```
%sql SELECT BOOSTER_VERSION FROM SPACEX WHERE LANDING__OUTCOME = 'Success (drone ship)' \
AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;
```

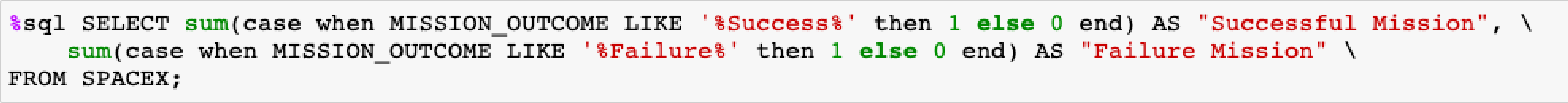
Selecting only Booster_Version,
WHERE clause filters the dataset to Landing_Outcome = Success (drone ship)

AND clause specifies additional filter conditions Payload_MASS_KG_ > 4000 AND Payload_MASS_KG_ < 6000

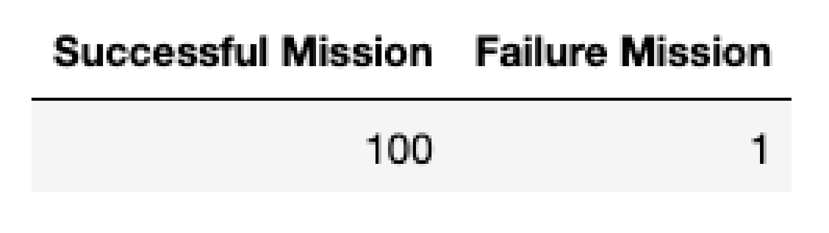


booster_version
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

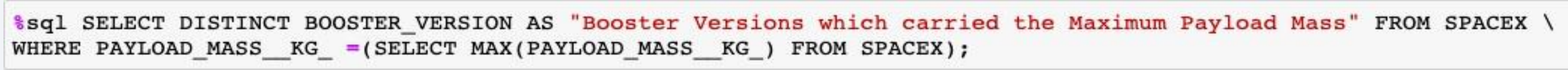


```
%sql SELECT sum(case when MISSION_OUTCOME LIKE '%Success%' then 1 else 0 end) AS "Successful Mission", \
    sum(case when MISSION_OUTCOME LIKE '%Failure%' then 1 else 0 end) AS "Failure Mission" \
FROM SPACEX;
```



Successful Mission	Failure Mission
100	1

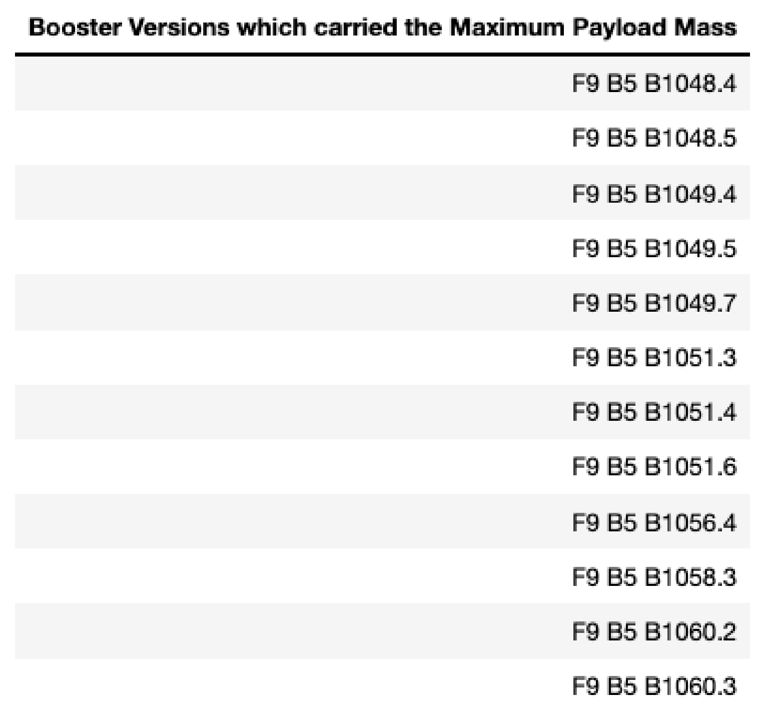
Boosters Carried Maximum Payload



```
%sql SELECT DISTINCT BOOSTER_VERSION AS "Booster Versions which carried the Maximum Payload Mass" FROM SPACEX \
WHERE PAYLOAD_MASS__KG_ =(SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEX);
```

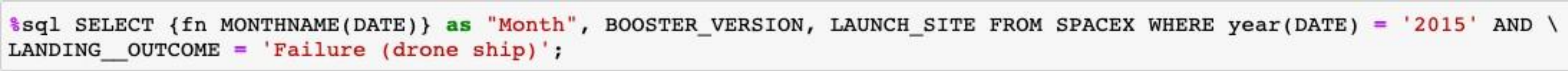
Using the function MAX works out the maximum payload in the column PAYLOAD_MASS___KG_ in sub query.

WHERE clause filters Booster Version which had that maximum payload



Booster Versions which carried the Maximum Payload Mass
F9 B5 B1048.4
F9 B5 B1048.5
F9 B5 B1049.4
F9 B5 B1049.5
F9 B5 B1049.7
F9 B5 B1051.3
F9 B5 B1051.4
F9 B5 B1051.6
F9 B5 B1056.4
F9 B5 B1058.3
F9 B5 B1060.2
F9 B5 B1060.3

2015 Launch Records

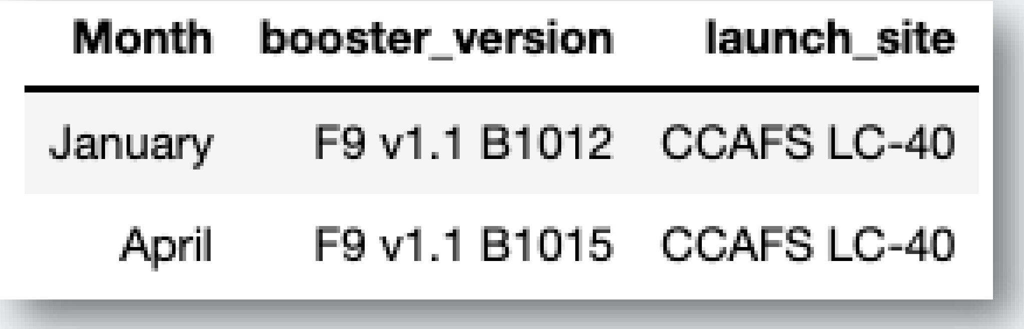


```
%sql SELECT {fn MONTHNAME(DATE)} as "Month", BOOSTER_VERSION, LAUNCH_SITE FROM SPACEX WHERE year(DATE) = '2015' AND \
LANDING__OUTCOME = 'Failure (drone ship)';
```

We need to 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.

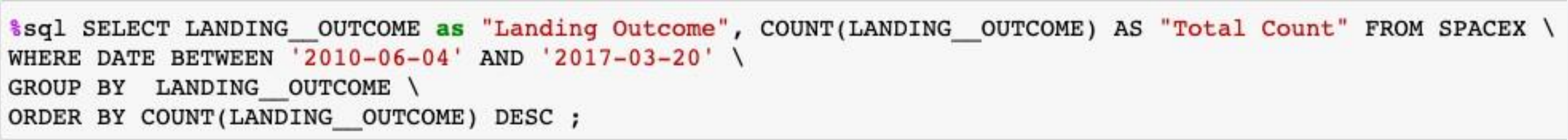
Via year function we extract the year and future where clause 'Failure (drone ship)' fetches our required values.

Also, am using {fn MONTHNAME(DATE)} to get the Month name.



Month	booster_version	launch_site
January	F9 v1.1 B1012	CCAFS LC-40
April	F9 v1.1 B1015	CCAFS LC-40

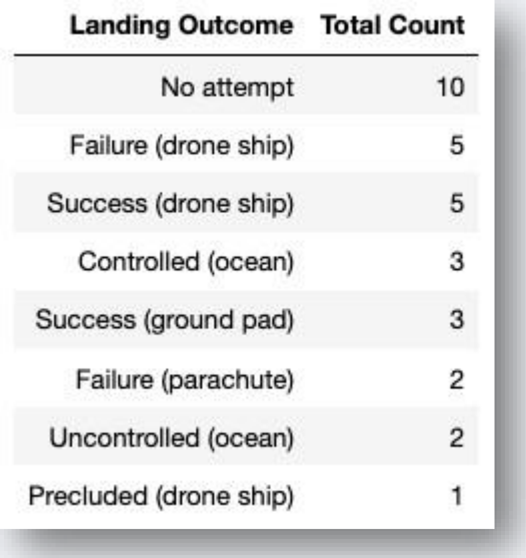
Rank Landing Outcomes Between 2010-06-04 and 2017-03-20



```
%sql SELECT LANDING__OUTCOME as "Landing Outcome", COUNT(LANDING__OUTCOME) AS "Total Count" FROM SPACEX \
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' \
GROUP BY  LANDING__OUTCOME \
ORDER BY COUNT(LANDING__OUTCOME) DESC ;
```

Selecting only LANDING__OUTCOME,
WHERE clause filters the data with DATE BETWEEN '2010-06-04' AND '2017-03-20'

Grouping by LANDING__OUTCOME
Order by COUNT(LANDING__OUTCOME) in Descending Order.

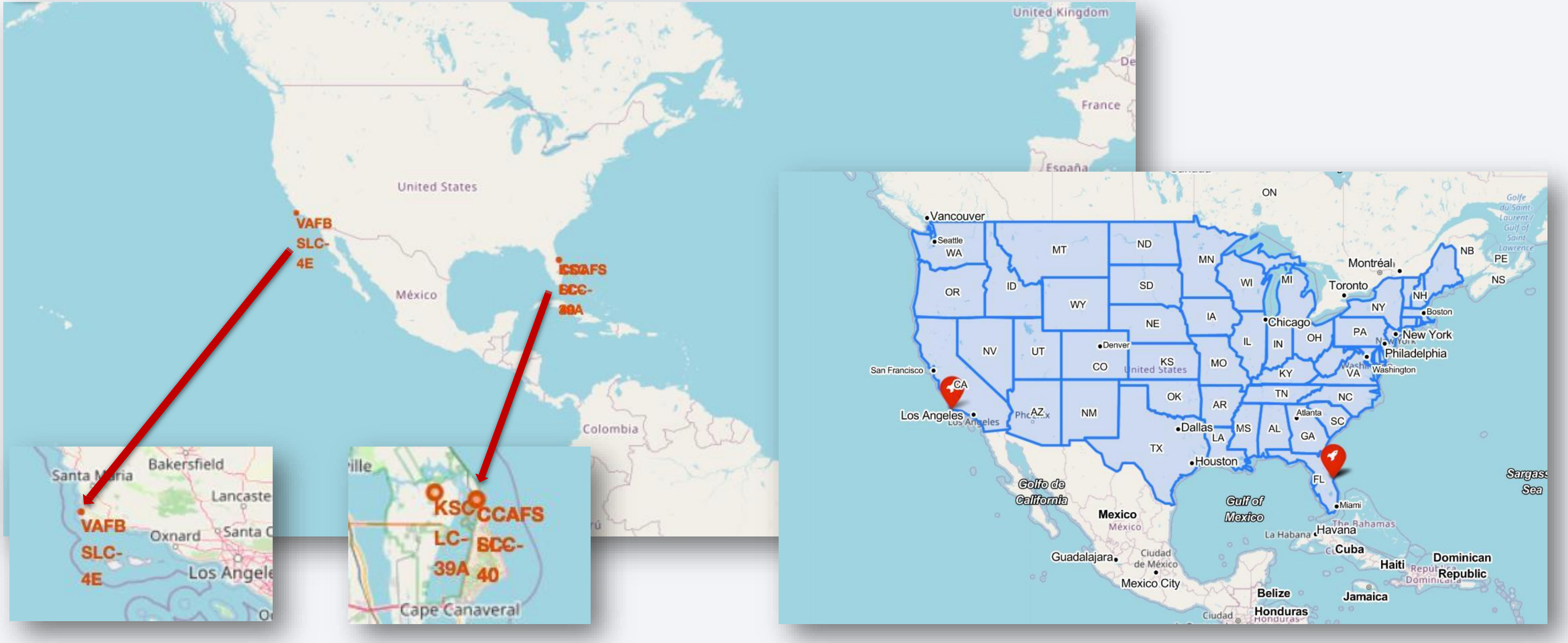


Landing Outcome	Total Count
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Failure (parachute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

Section 4

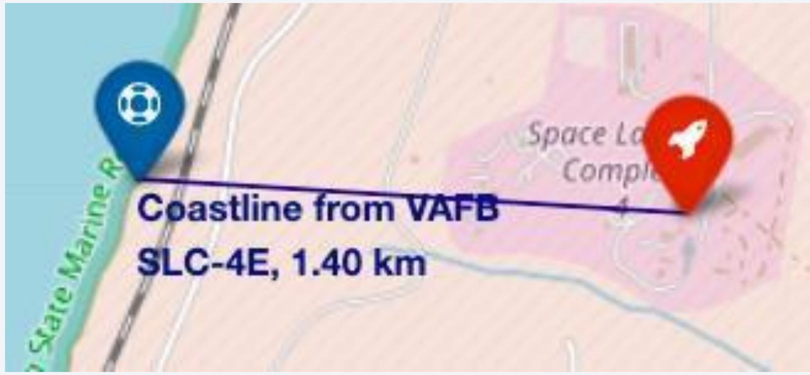
Launch Sites Proximities Analysis

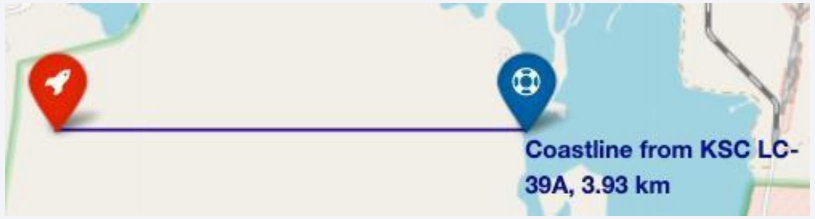
<Folium Map Screenshot 1>

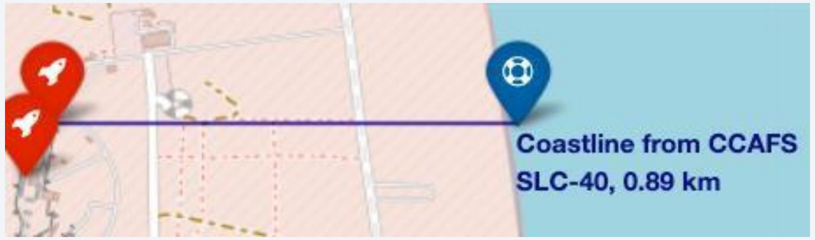


United States
México
Colombia
United Kingdom
France
España
VAFB
SLC-
4E
CCAFS
SLC-
40
Santa Maria
Bakersfield
Lancaster
Oxnard
Santa C
Los Angeles
KSC
LC-
39A
CCAFS
SLC-
40
Cape Canaveral
Vancouver
Seattle
San Francisco
Los Angeles
Denver
Dallas
Houston
Chicago
Atlanta
Miami
Toronto
Montréal
Boston
New York
Philadelphia
Washington
Gulf of Mexico
Golfo de California
Mexico
Guadalajara
Mexico City
Havana
Cuba
Haiti
Dominican Republic
Jamaica
Belize
Honduras
Sargasso Sea

<Folium Map Screenshot 2>

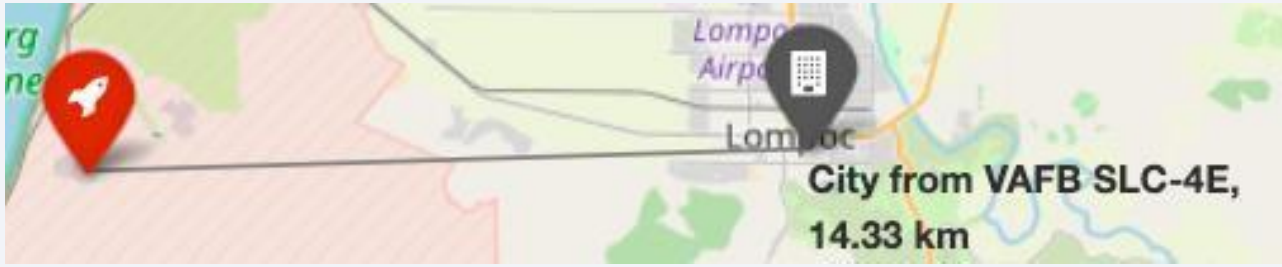


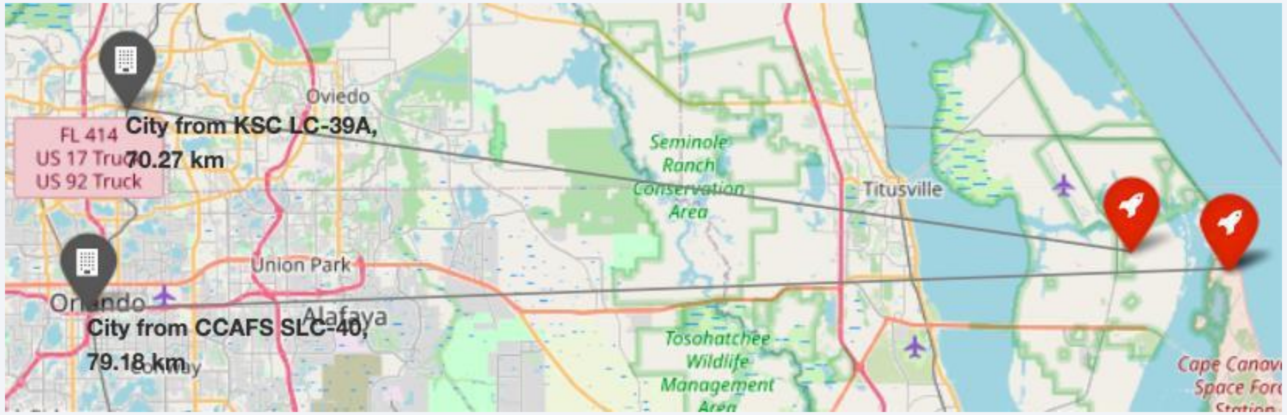




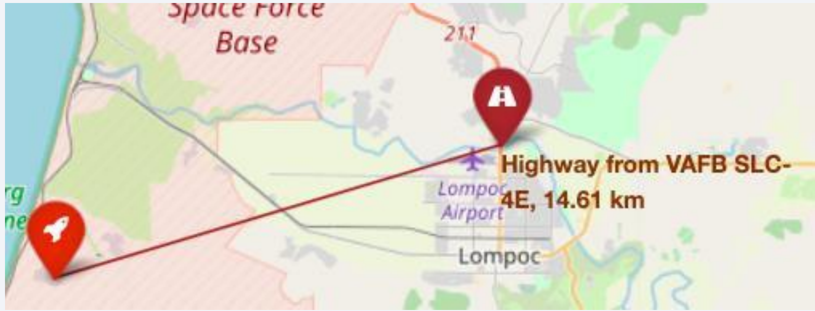
Distance for all launch sites from coastline is less than 4 Km.

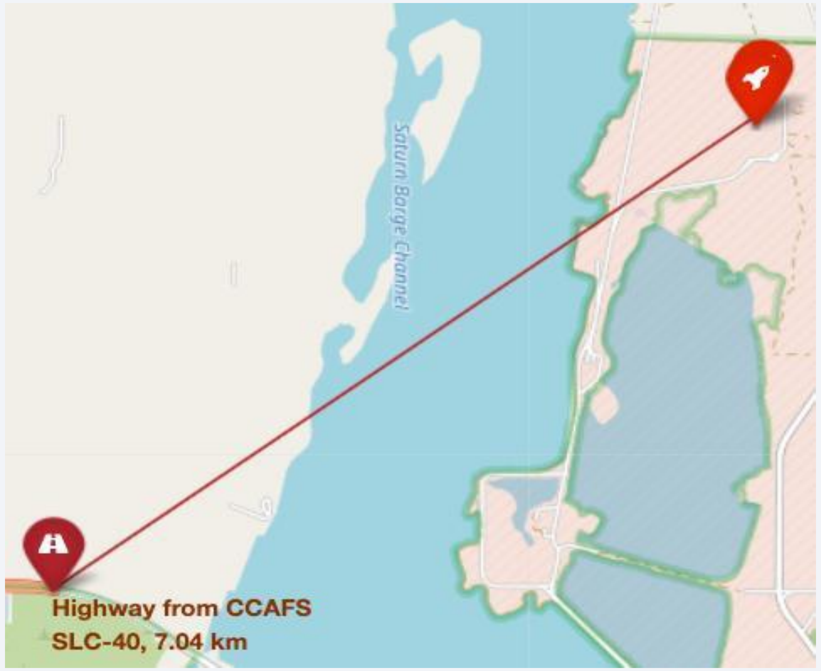
Distance for all launch sites from cities is greater than 14 Km for all sites. So, launch sites are far away from cities.

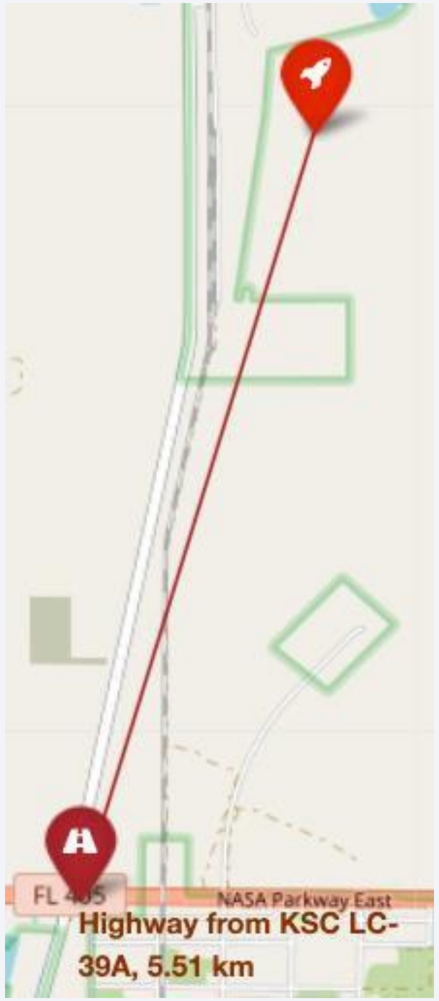




<Folium Map Screenshot 3>







Distance for all launch sites from highways is greater than 5 Km for all sites. So, launch sites are relatively far away from highways.

Conclusion:

- Are all launch sites in proximity to the Equator line?

  No (4000 Km > distance > 3000 Km)

- Are launch sites in close proximity to railways?

  Yes (2 Km > distance > .5 Km)

- Are launch sites in close proximity to highways?

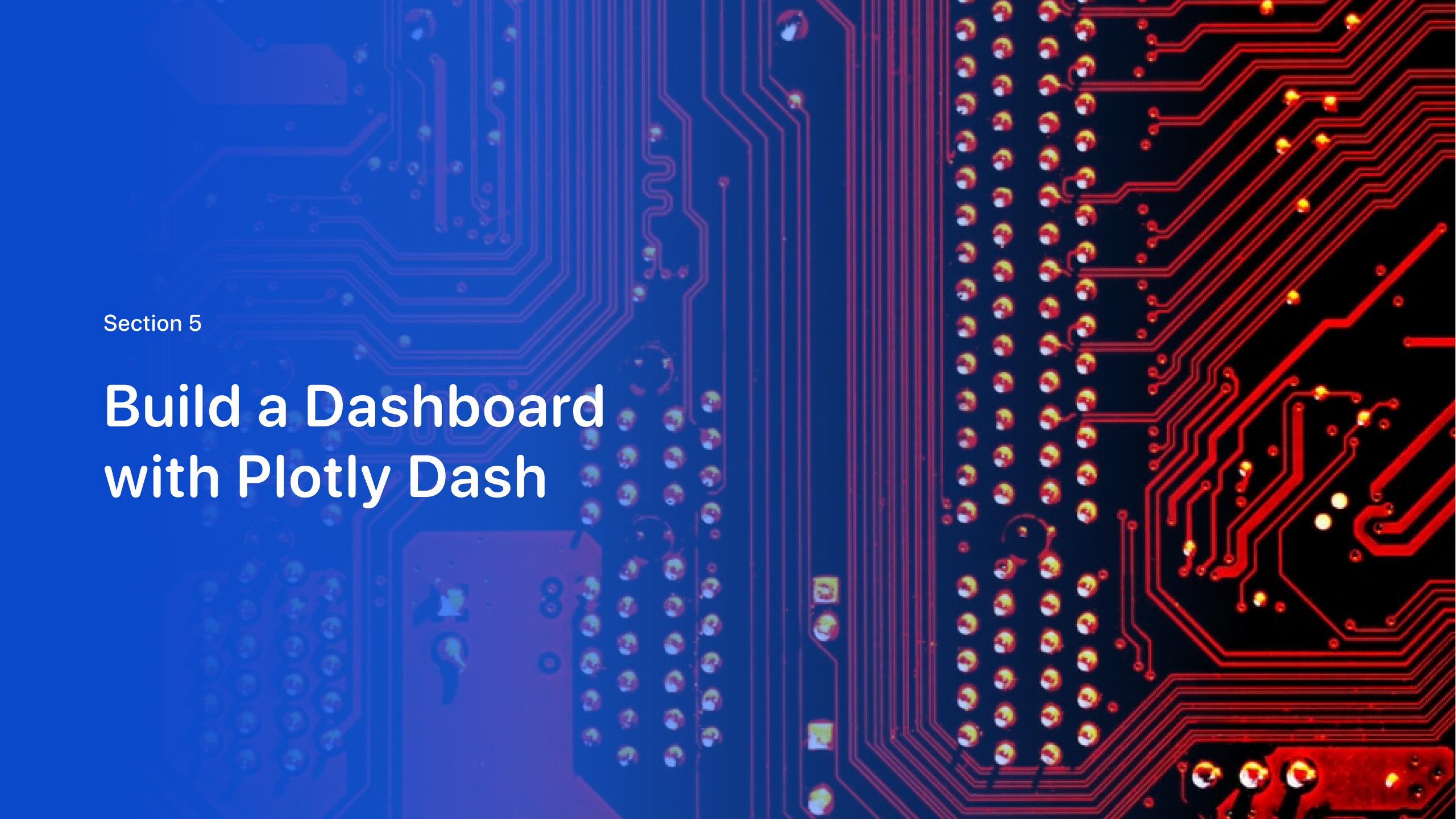
  No (15 Km > distance > 5 Km)

- Are launch sites in close proximity to coastline?

  Yes (5 Km > distance > .5 Km)

- Do launch sites keep certain distance away from cities?

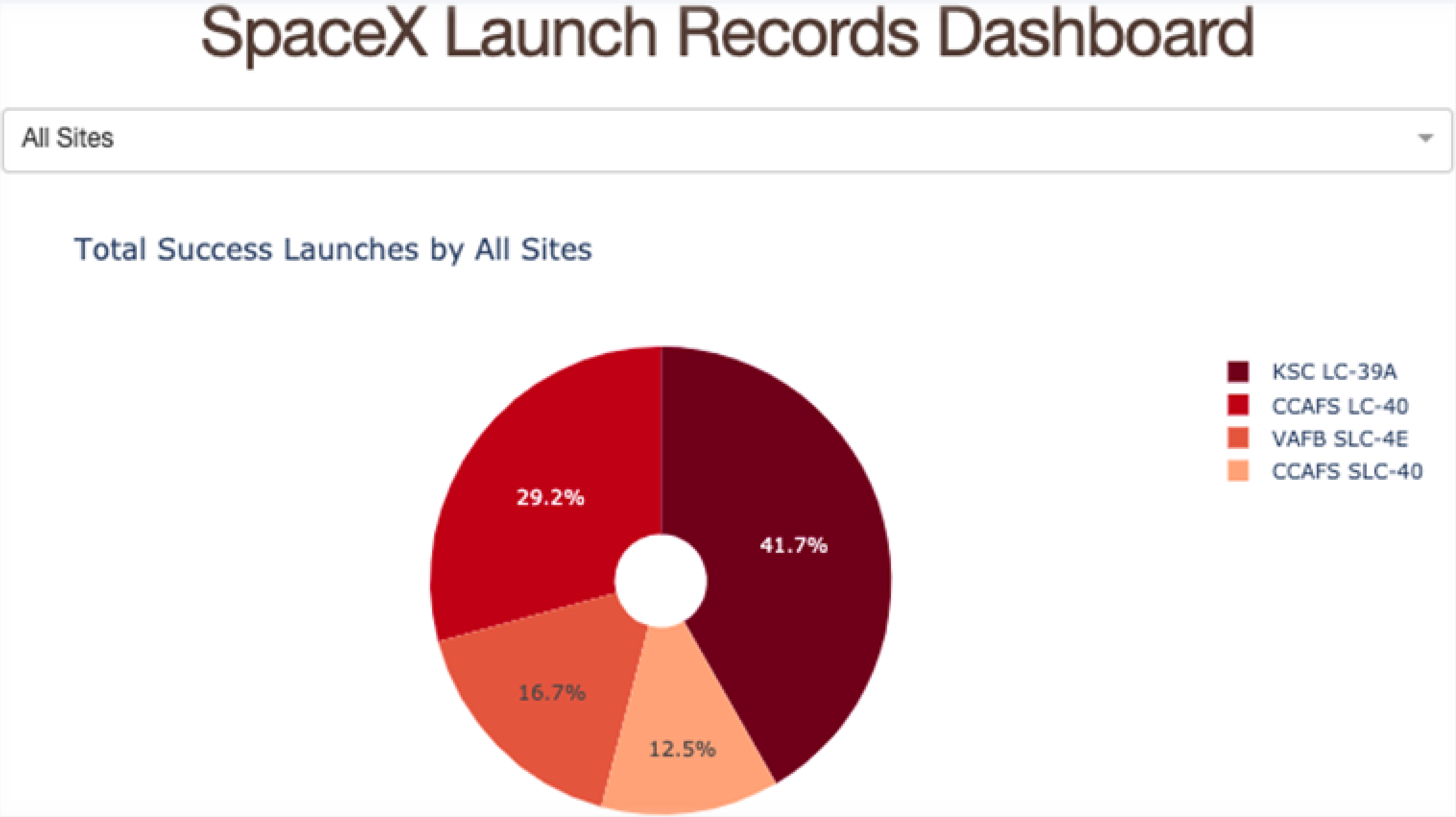
  Yes (15 Km > distance > 80 Km)



Section 5

Build a Dashboard with Plotly Dash

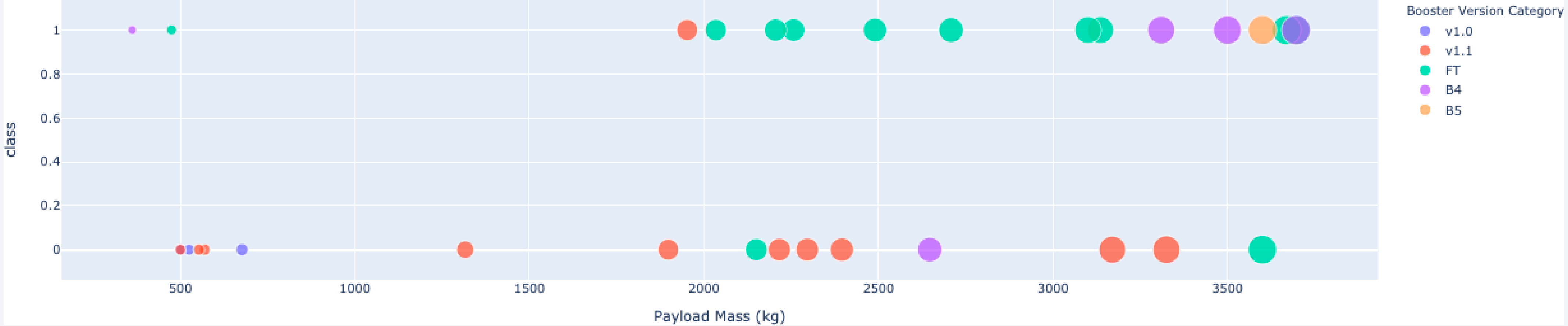
<Dashboard Screenshot 1>



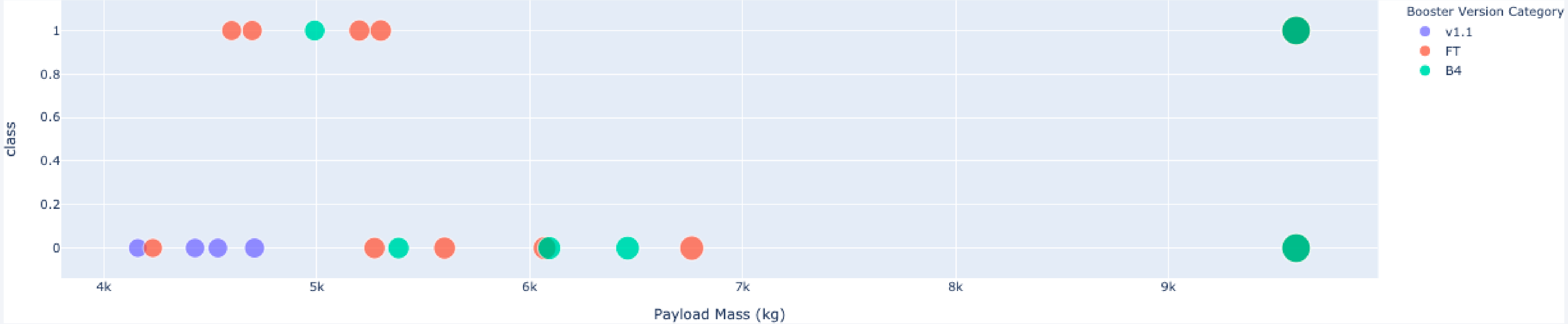
SpaceX Launch Records Dashboard

We can see that KSC LC-39A had the most successful launches from all the sites.

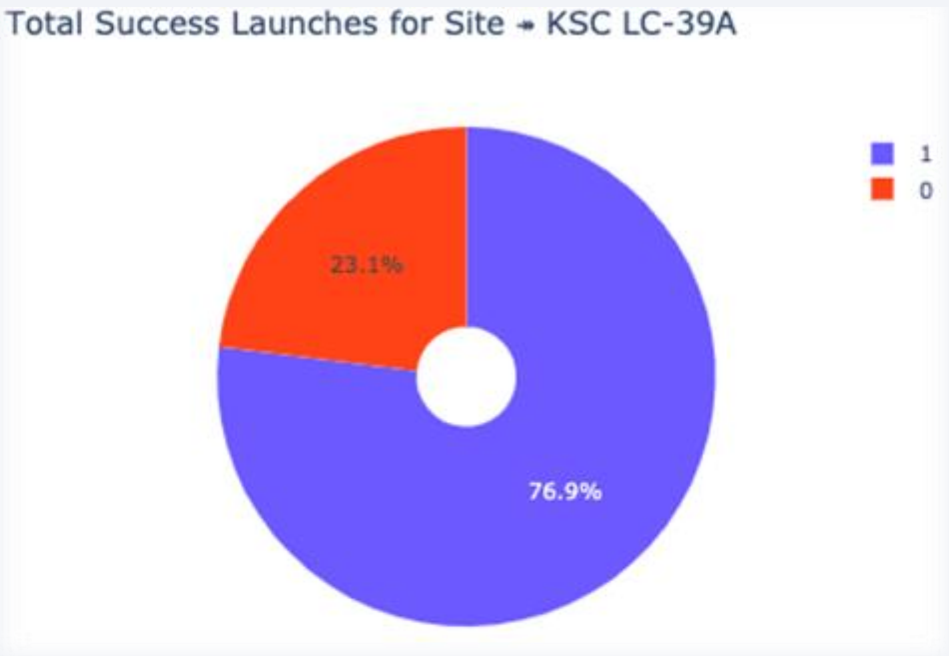
<Dashboard Screenshot 2>



Low Weighted Payload 4000kg – 10000kg



<Dashboard Screenshot 3>



KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate.

After visual analysis using the dashboard, we are able to obtain some insights to answer these questions:	Which site has the highest launch success rate?	**KSC LC – 39A**
	Which payload range(s) has the highest launch success rate?	**2000 Kg – 10000 Kg**
	Which payload range(s) has the lowest launch success rate?	**0 Kg – 1000 Kg**
	Which F9 Booster version (v1.0, v1.1, FT, B4, B5, etc.) has the highest launch success rate?	**FT**

Section 6

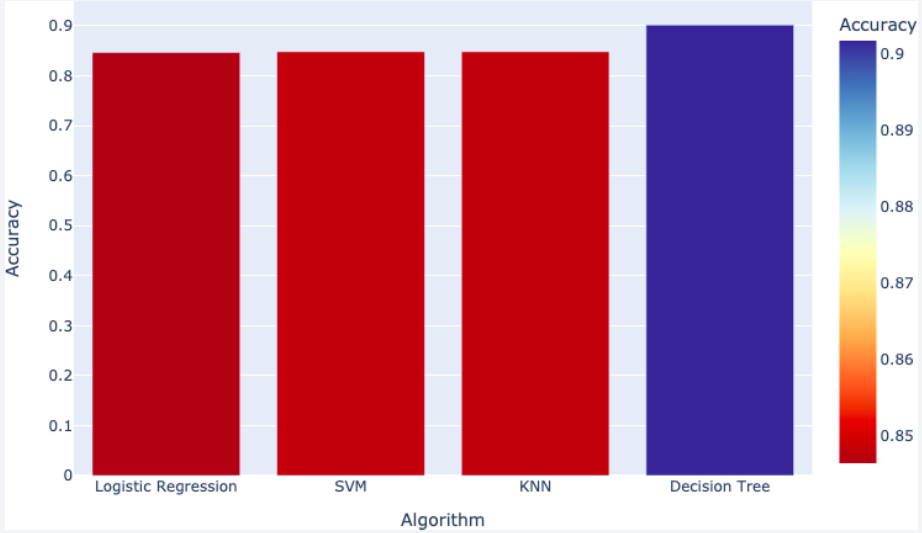
Predictive Analysis (Classification)

Classification Accuracy

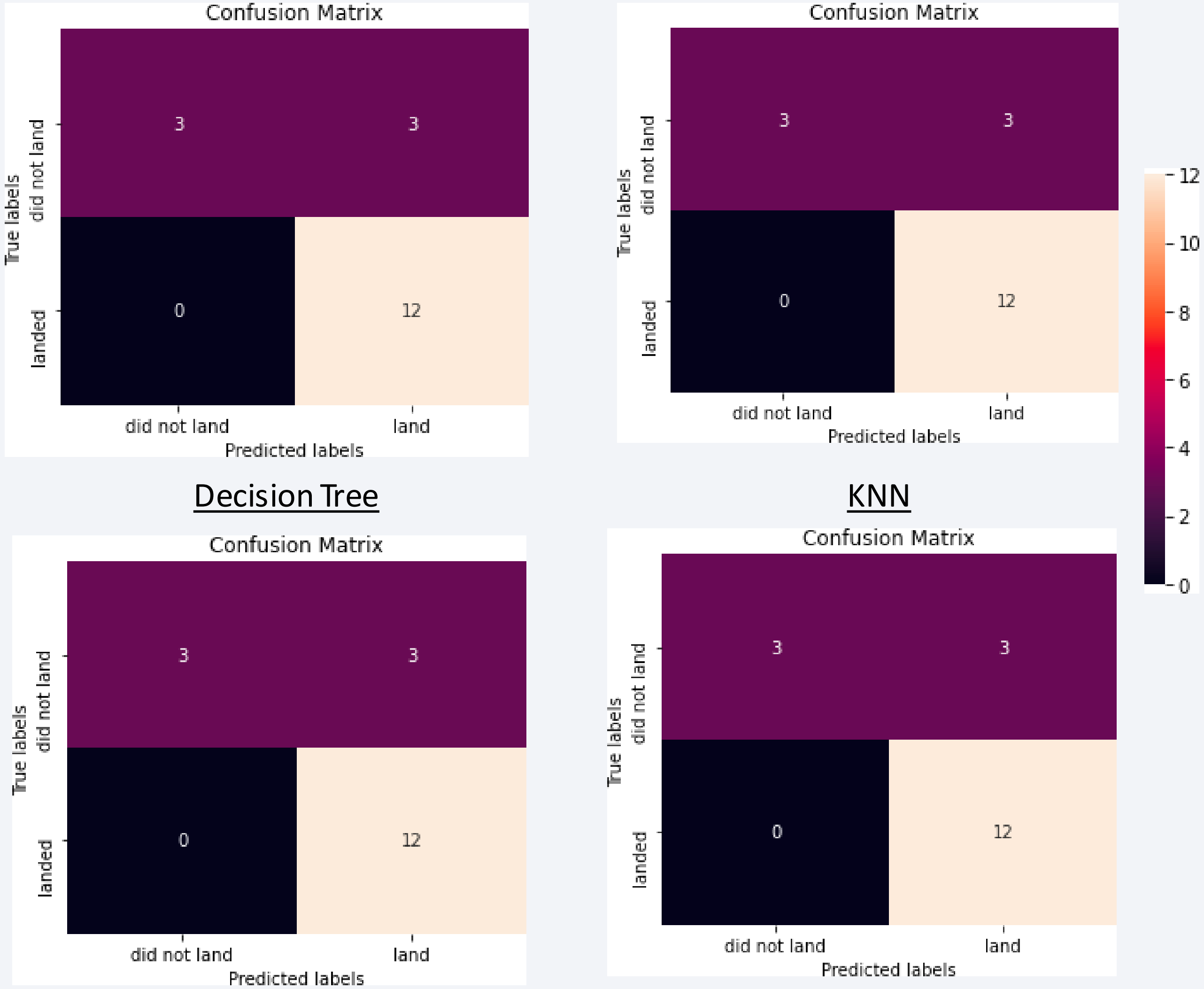
As you can see our accuracy is extremely close, but we do have a clear winner which performs best - "**Decision Tree**" with a score of 0.90178.

Algorithm	Accuracy	Accuracy on Test Data	Tuned Hyperparameters
Logistic Regression	0.846429	0.833334	{'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
SVM	0.848214	0.833334	{'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
KNN	0.848214	0.833334	{'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
Decision Tree	0.901786	0.833334	{'criterion': 'gini', 'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'splitter': 'best'}

We trained four different models which each had an 83% accuracy rate.



Confusion Matrix



Decision Tree

KNN

Accuracy: (TP+TN)/Total = (12+3)/18 = 0.83333

Misclassification Rate: (FP+FN)/Total = (3+0)/18 = 0.1667

True Positive Rate: TP/Actual Yes = 12/12 = 1

False Positive Rate: FP/Actual No = 3/6 = 0.5

True Negative Rate: TN/Actual No = 3/6 = 0.5

Precision: TP/Predicted Yes = 12/15 = 0.8

Prevalence: Actual yes/Total = 12/18 = 0.6667

Conclusions

- Orbits ES-L1, GEO, HEO, SSO has highest Sucess rates
- Success rates for SpaceX launches has been increasing relatively with time and it looks like soon they will reach the required target
- KSC LC-39A had the most successful launches but inceasing payload mass seems to have negative impact on success

- KSC LC-39A had the most successful launches but inceasing payload mass seems to have negative impact on success

Appendix

- Interactive Plotly
- Folium
- IBM cognos visualization tool
- Decision Tree Construction

Thank you!