IBM Developer
SKILLS NETWORK

# Winning Space Race with Data Science

Marijn Hagenaar
31st of December 2021

# Outline

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

# Executive Summary

**Methodologies that have been used:**

- 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 with Machine Learning

**Results that will be presented are:**

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

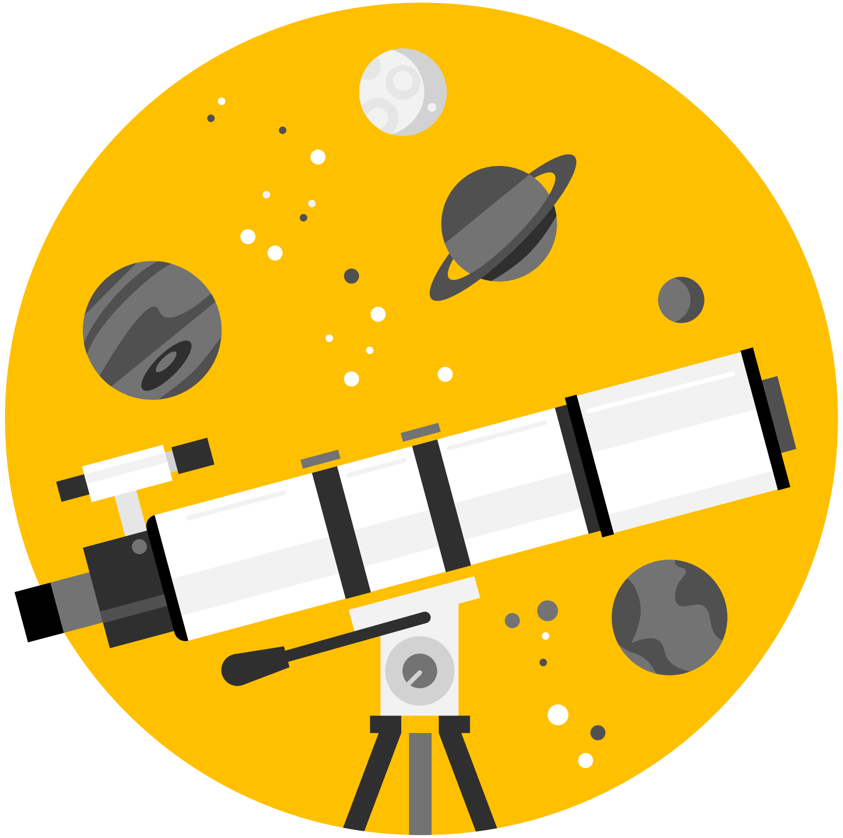
# Introduction

**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.

**Common problems that needed solving**

- What influences if the rocket will land successfully?
- The effect each relationship with certain rocket variables will impact in determining the success rate of a successful landing.
- What conditions does SpaceX have to achieve to get the best results and ensure the best rocket success landing rate.



# METHODOLOGY

# Methodology

**Data collection methodology:**

- SpaceX Rest API
- (Web Scrapping) from Wikipedia

- Performed data wrangling (Transforming data for Machine Learning)
  - One Hot Encoding data fields for Machine Learning and dropping irrelevant columns

- Performed exploratory data analysis (EDA) using visualization and SQL
  - Plotting : Scatter Graphs, Bar Graphs to show relationships between variables to show  patterns of data.

- Performed interactive visual analytics using Folium and Plotly Dash

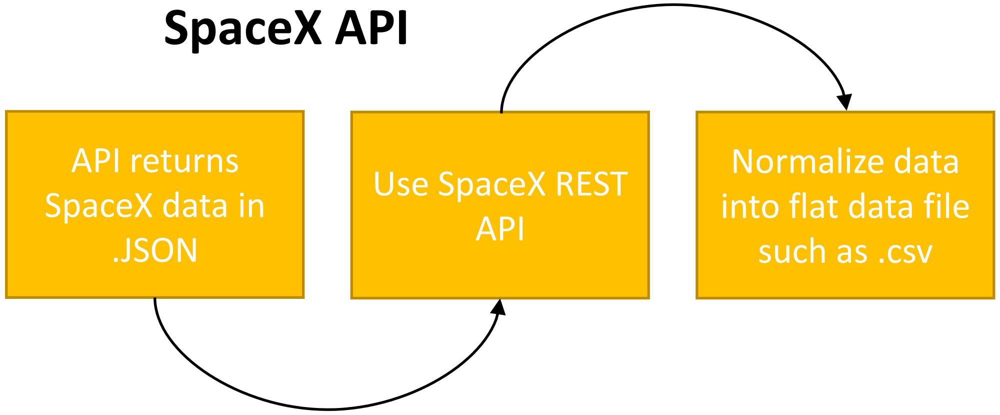
- Performed predictive analysis using classification models
  - How to build, tune, evaluate classification models

# Data Collection

**The dataset was collected by**

- 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.

## SpaceX API

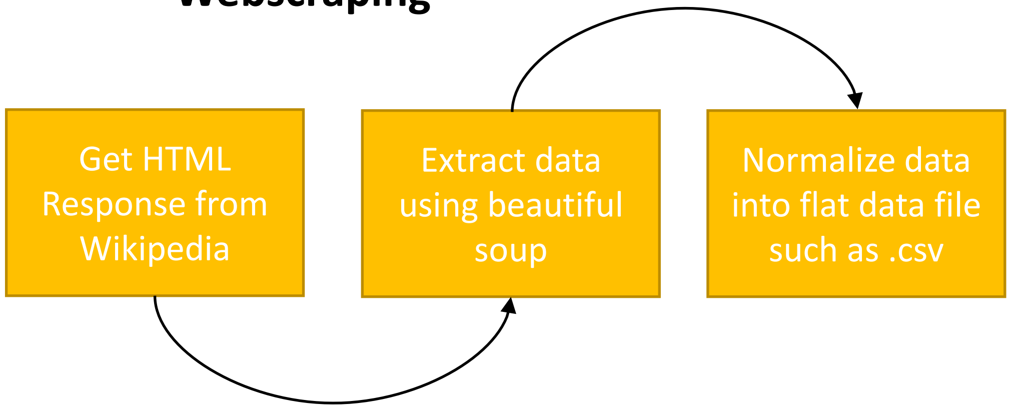


API returns SpaceX data in .JSON

Use SpaceX REST API

Normalize data into flat data file such as .csv

## Webscraping

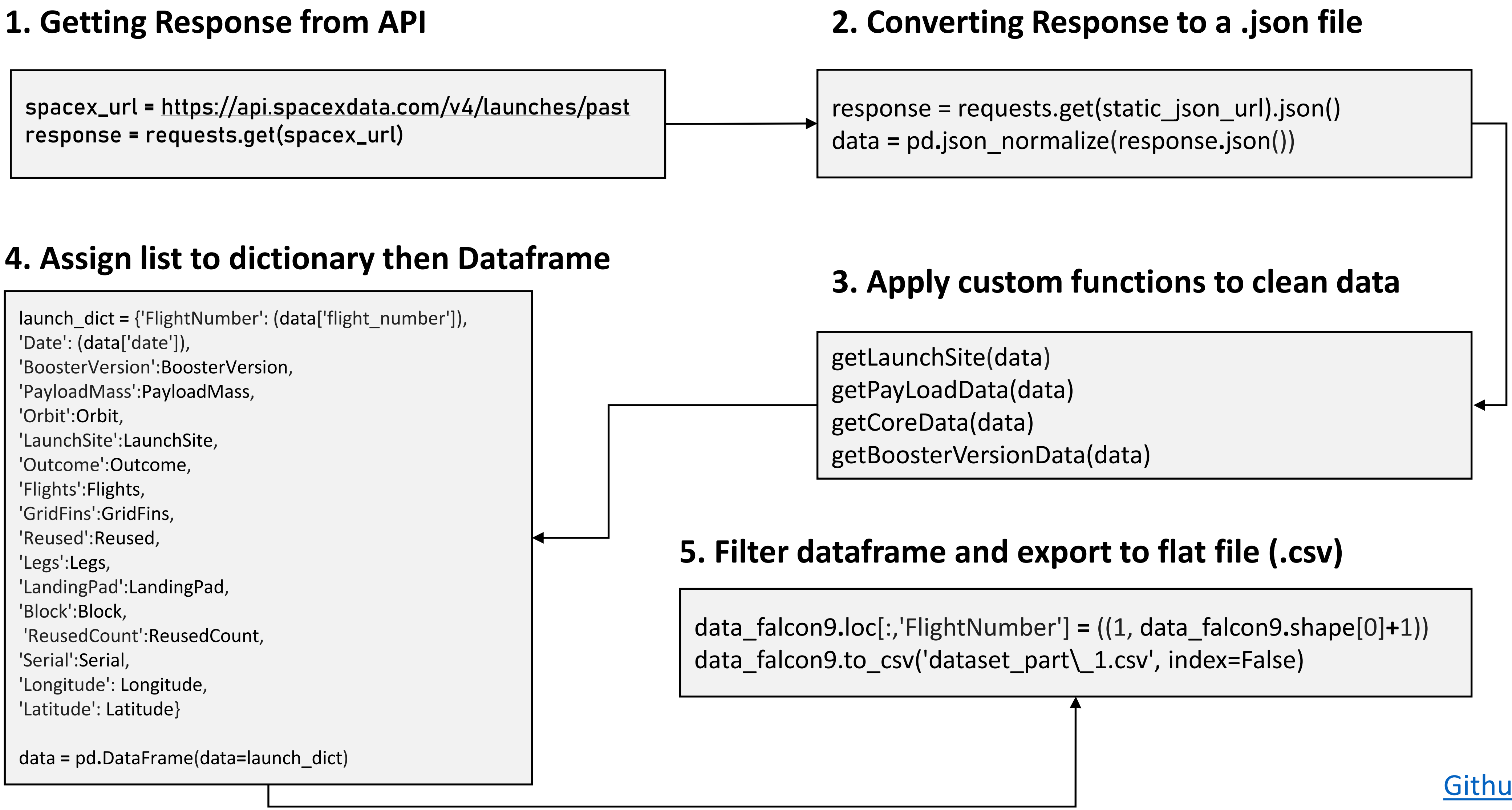


Get HTML Response from Wikipedia

Extract data using beautiful soup

Normalize data into flat data file such as .csv

# Data Collection – SpaceX API



### 1. Getting Response from API

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

### 2. Converting Response to a .json file

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

### 3. Apply custom functions to clean data

```
getLaunchSite(data)
getPayLoadData(data)
getCoreData(data)
getBoosterVersionData(data)
```

### 4. Assign list to dictionary then Dataframe

```
launch_dict = {'FlightNumber': (data['flight_number']),
'Date': (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 = pd.DataFrame(data=launch_dict)
```

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

```
data_falcon9.loc[:,'FlightNumber'] = ((1, data_falcon9.shape[0]+1))
data_falcon9.to_csv('dataset_part\_1.csv', index=False)
```

Github Notebook

# Data Collection - Scraping

### 1. Get response from HTML

Page = requests.get(static_url)

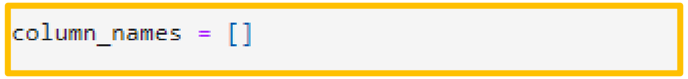
### 2. Create BeautifulSoup object

Soup = BeautifulSoup(data, ‘html5lib’)

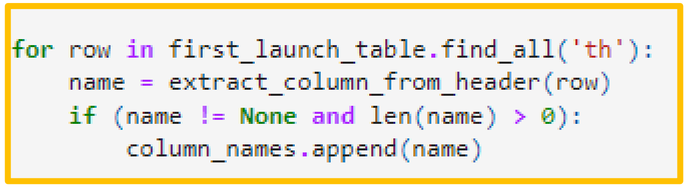
### 3. Finding tables

Html_tables = soup.find_all(‘table‘)

### 4. Getting column names

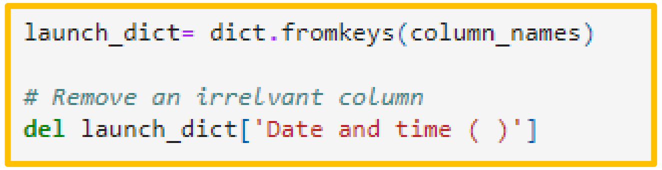


```
column_names = []
```



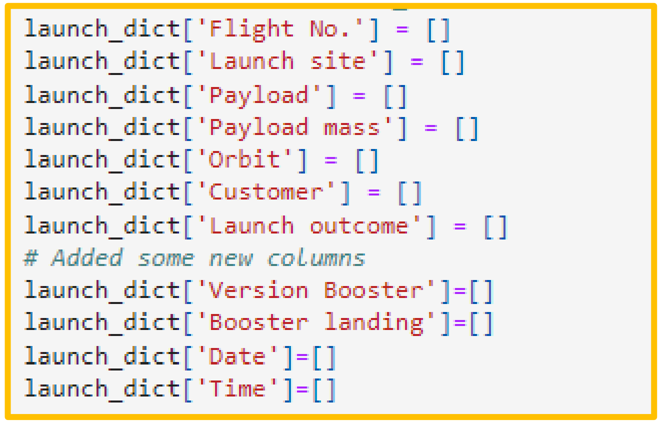
```
for row in first_launch_table.find_all('th'):
    name = extract_column_from_header(row)
    if (name != None and len(name) > 0):
        column_names.append(name)
```

### 5. Create a Dictionary



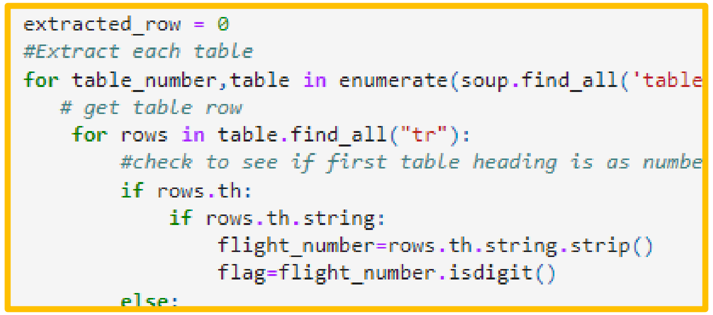
```
launch_dict= dict.fromkeys(column_names)

# 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'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

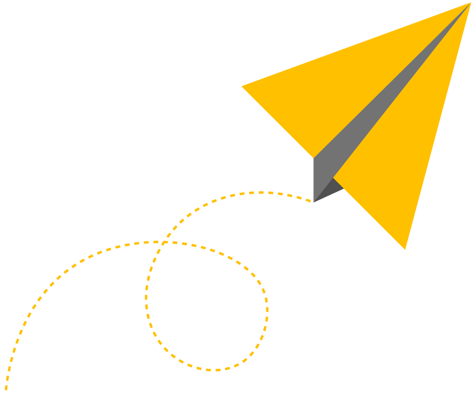
### 6. Appending data to keys



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

### 7. Converting dictionary into Dataframe

### 8. Dataframe to CSV



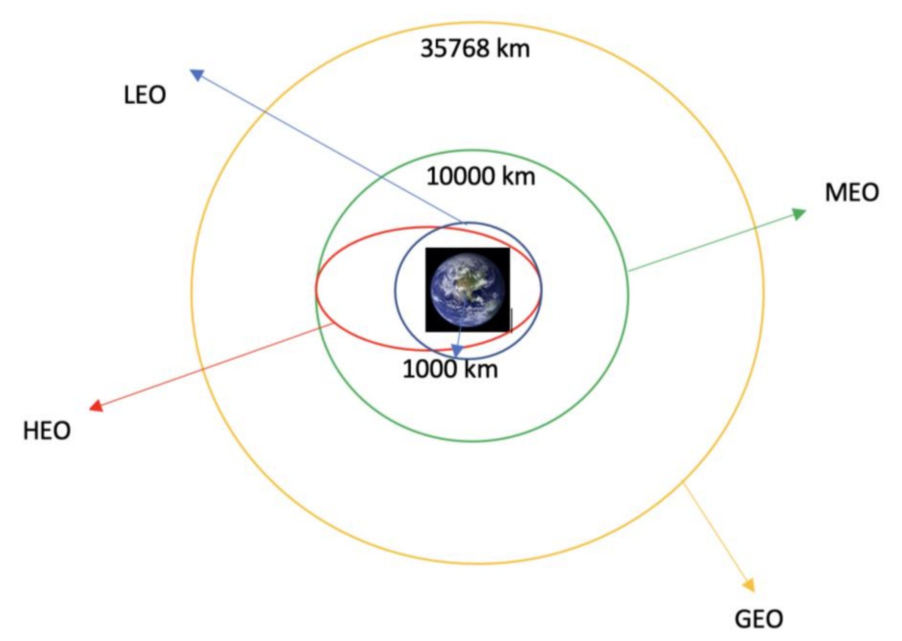
[Github Notebook]

# Data Wrangling

In the data set, there are several different cases where the booster did not land successfully. Sometimes a landing was attempted but failed due to an accident; for example, True Ocean means the mission outcome was successfully landed to a specific region of the ocean while False Ocean means the mission outcome was unsuccessfully landed to a specific region of the ocean. True RTLS means the mission outcome was successfully landed to a ground pad False RTLS means the mission outcome was unsuccessfully landed to a ground pad. True ASDS means the mission outcome was successfully landed on a drone ship False ASDS means the mission outcome was unsuccessfully landed on a drone ship. We mainly convert those outcomes into Training Labels with 1 means the booster successfully landed 0 means it was unsuccessful.

**The data was processed in the following way:**

1. Perform Exploratory Data Analysis EDA on dataset
2. Calculate the number of launches on each site
3. Calculate the number and occurrence of each orbit
4. Calculate the number and occurrence of mission outcome per orbit type
5. Create a landing outcome label from Outcome column
6. Work out success rate for every landing in dataset
7. Export dataset as .CSV



Github Notebook

# EDA with Data Visualization

Different visualizations were drawn by making use of Python packages NumPy, Pandas, Seaborn and Matplotlib

**Scatterplots**
Scatter plots show how much one variable is affected by another. The relationship between two variables is called their correlation . Scatter plots usually consist of a large body of data.

- Flight Number VS. Payload
- Mass Flight Number VS. Launch Site
- Payload VS. Launch Site
- Orbit VS. Flight Number
- Payload VS. Orbit Type
- Orbit VS. Payload Mass

**Line Graph**
Line graphs are useful in that they show data variables and trends very clearly and can help to make predictions about the results of data not yet recorded.

- Success Rate VS. Year

**Bar Chart**
A bar chart makes it easy to compare sets of data between different groups at a glance. The graph represents categories on one axis and a discrete value in the other. The goal is to show the relationship between the two axes. Bar charts can also show big changes in data over time.

- Mean VS. Orbit

Github Notebook

# EDA with SQL

**Performed SQL queries to gather information about the dataset**

Questions we were asked about the data we needed information about. Which we are using SQL queries to get the answers in the dataset:

- Displaying the names of the unique launch sites in the space mission
- Displaying 5 records where launch sites begin with the string 'KSC’
- 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_versions which have carried the maximum payload mass
- Listing the records which will display the month names, successful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017
- Ranking the count of successful landing_outcomes between the date 2010-06-04 and 2017-03-20 in descending order.

Github Notebook

# Build an Interactive Map with Folium

To visualize the Launch Data into an **interactive map**. We took the *Latitude* and *Longitude* Coordinates at each launch site and added a Circle Marker around each launch site with a label of the name of the launch site.

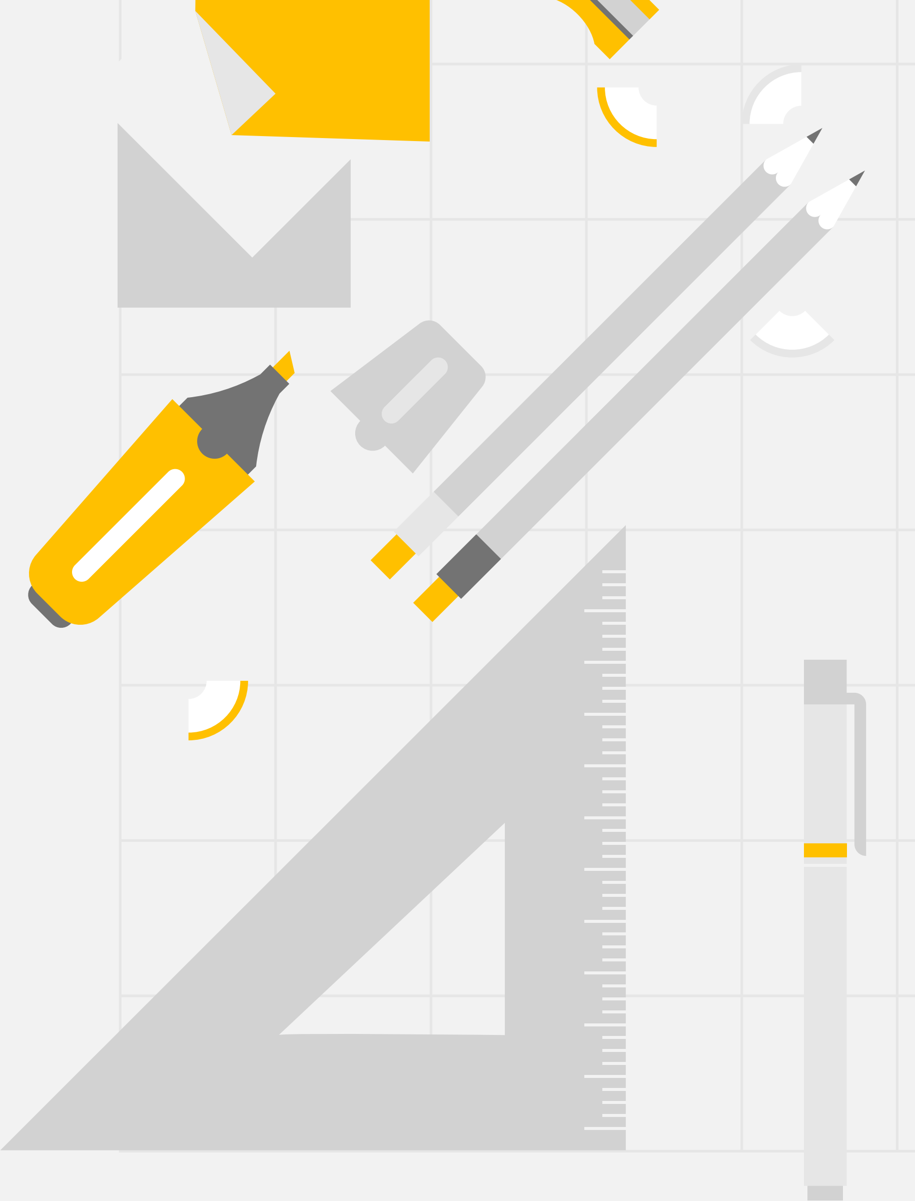
We assigned the dataframe launch_outcomes(failures, successes) to classes **0 and 1** with **Green** and **Red** markers on the map in a MarkerCluster()

Using **Haversine's formula** we calculated the *distance* from the Launch Site to various landmarks to find various trends about what is around the Launch Site to measure patterns. **Lines** are drawn on the map to measure distance to landmark.

Example of trends: launch sites close to railways, highways and coastlines.

Github Notebook

# Build a Dashboard with Plotly Dash



*The Dashboard was built with Flask and Dash Web framework. Different graphs were built to give more insight in the type of launches and relationships between variables.*

**Scatter Plot**
This chart was used to show the relationship between Outcome and Playload Mass (kg) for different Booster versions. The plot shows relationships between two variables and is the best method to show a non-linear pattern. The range of the data flow, i.e. max and min can be deterimed.

**Pie Chart**
This chars was used to show the total launches by certain site (or all sites). It displayed relative proportions of multiple classes of data. The size of the circle could be made proportional to the total quantity of what it represents.

[Github Notebook]

# Predictive Analysis (Classification)

To perform the predictive analysis there were a few steps taken:

1

**Building the model**

- Load the dataset into NumPy and Pandas
- Transform the data
- Split the data into training and test datasets
- Check the amount of test samples
- Decide which type of Machine Learning algorithm is used
- Set the parameters and algorithms to GridSearchCV
- Fit the datasets into the GridSearchCV objects and train the datast

2

**Evaluating the model**

- Check the accuracy of the models
- Get tuned hyperparameters for each of the algorithms
- Plot Confusion Matrixes

3

**Improving the model**

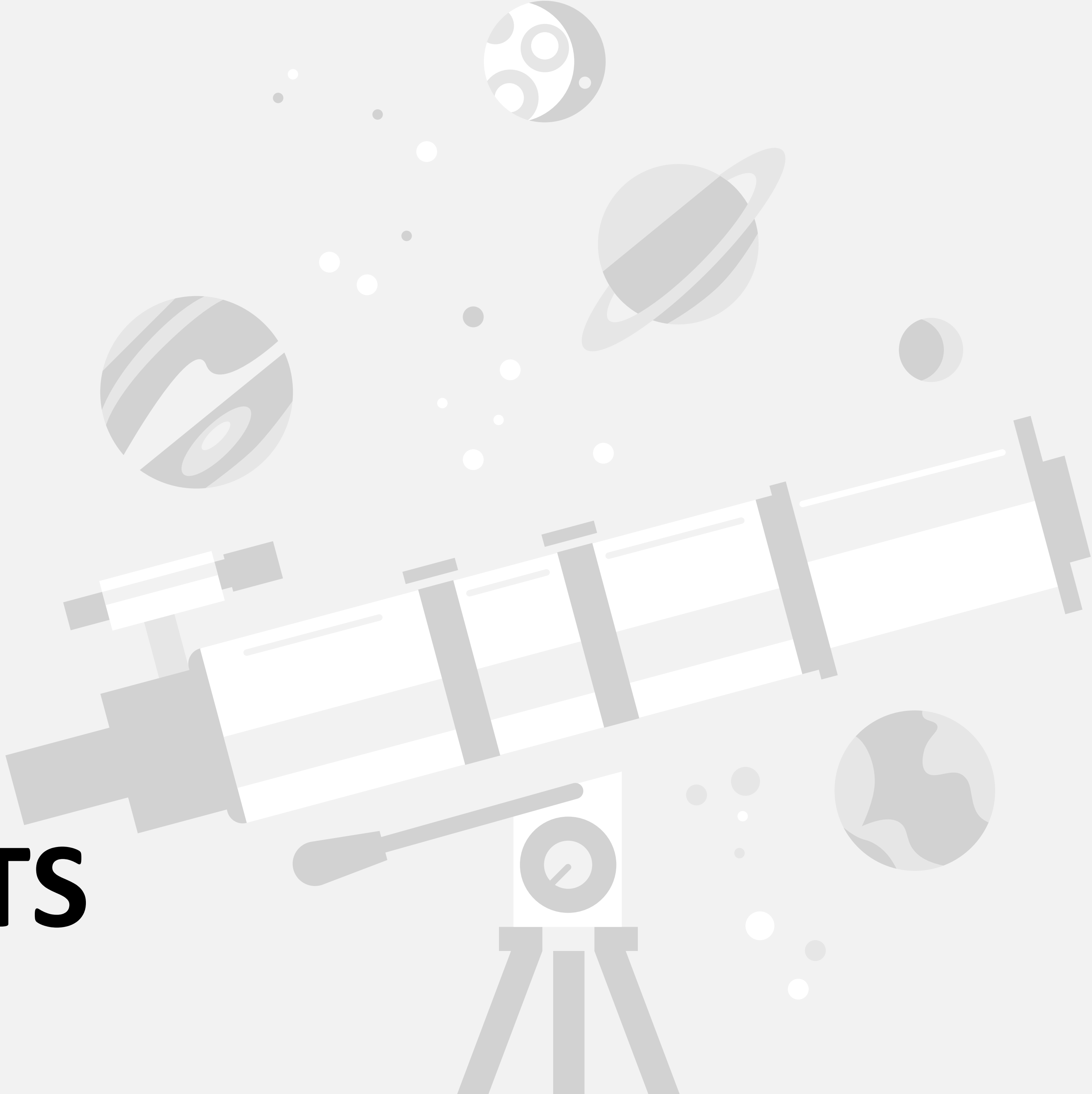
- Feature Engineering
- Algorithm Tuning

4

**Finding the best performing model**

- The model with the best accuracy score wins and is the best performing model
- All scores can be found in the Notebook

Github Notebook



# RESULTS

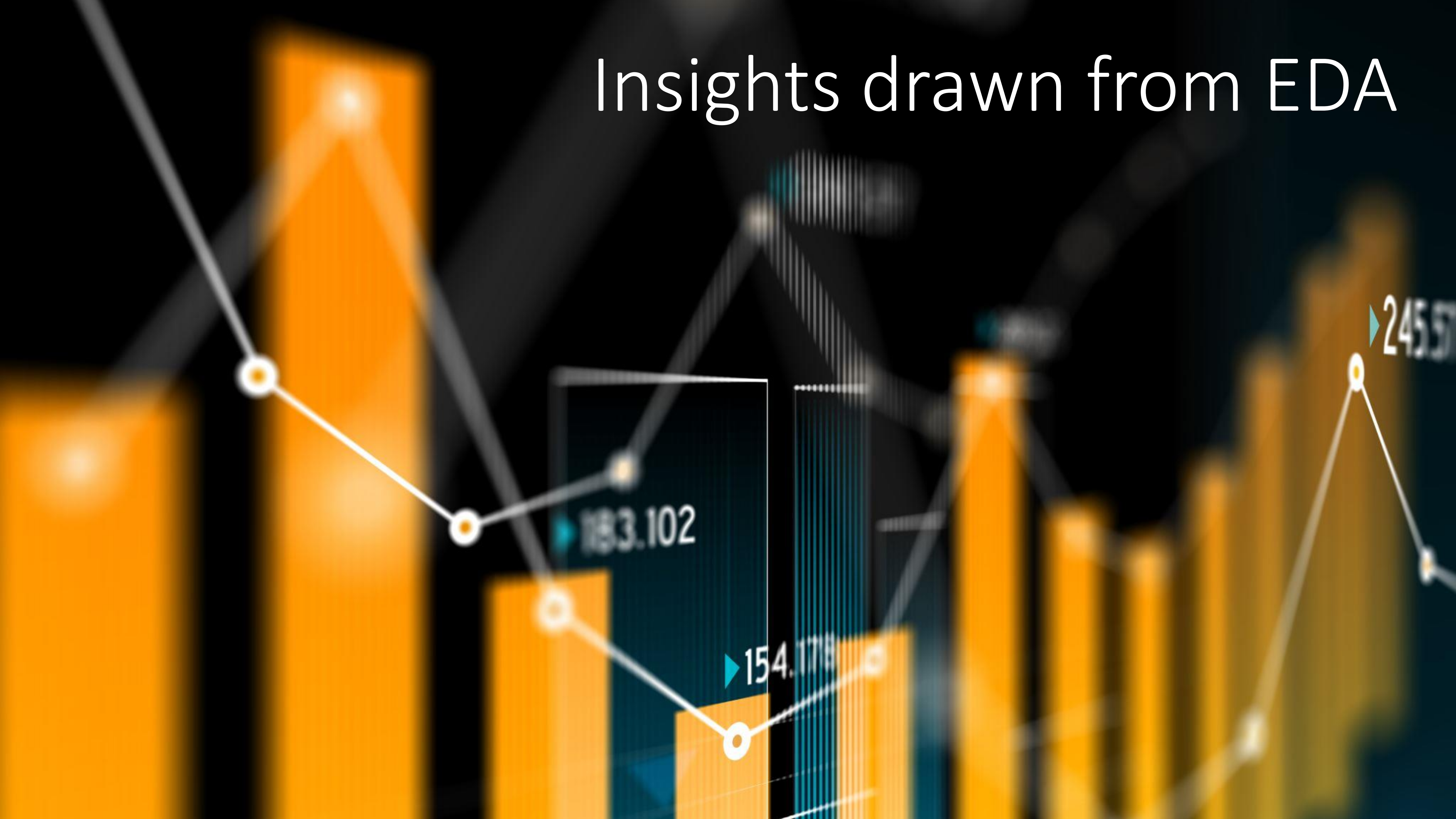
# Results

Exploratory data analysis results

Interactive analytics demo in screenshots

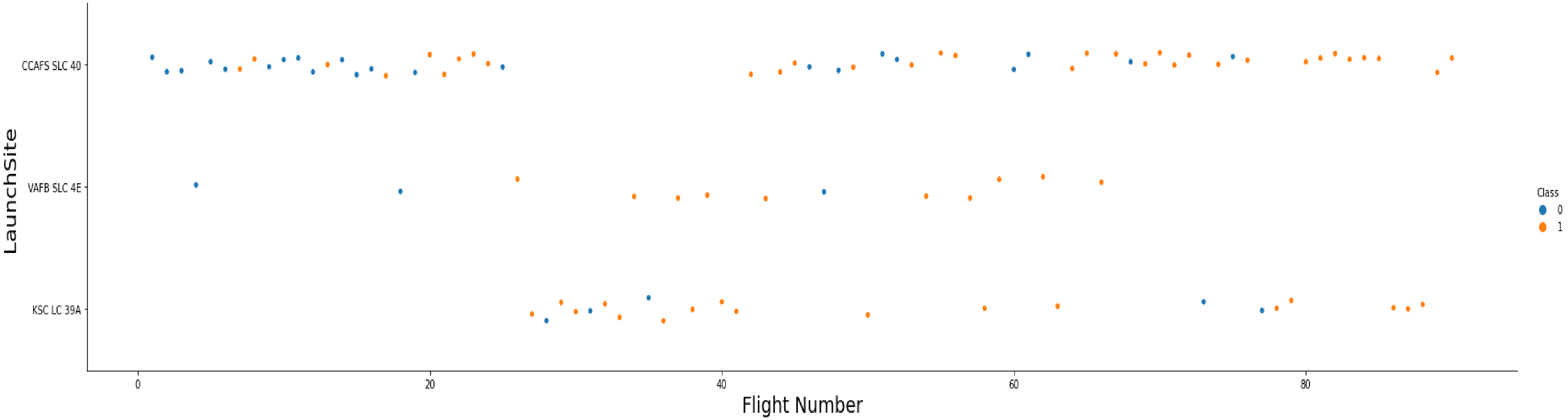
Predictive analysis results

# Insights drawn from EDA



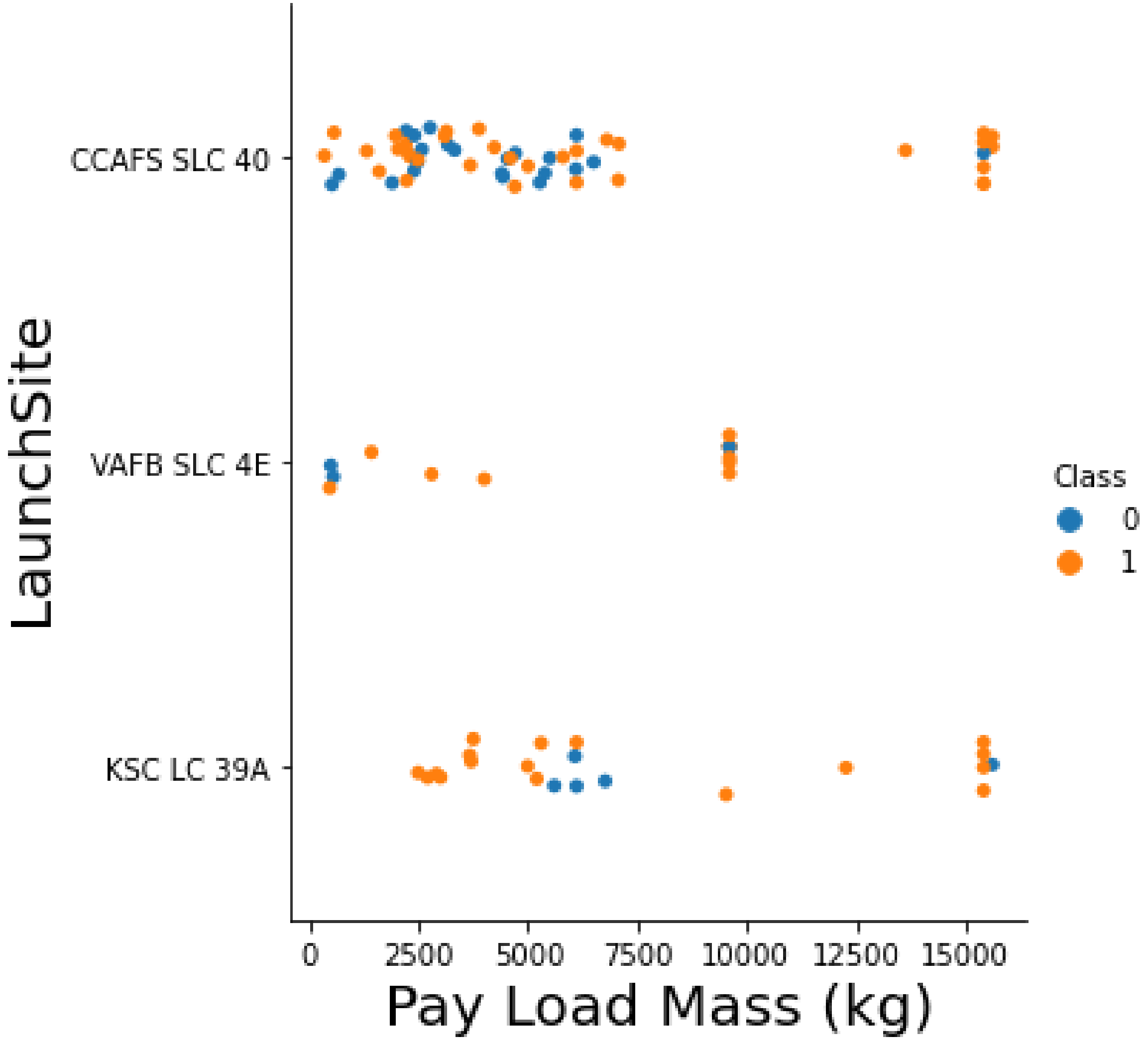
# Flight Number vs. Launch Site

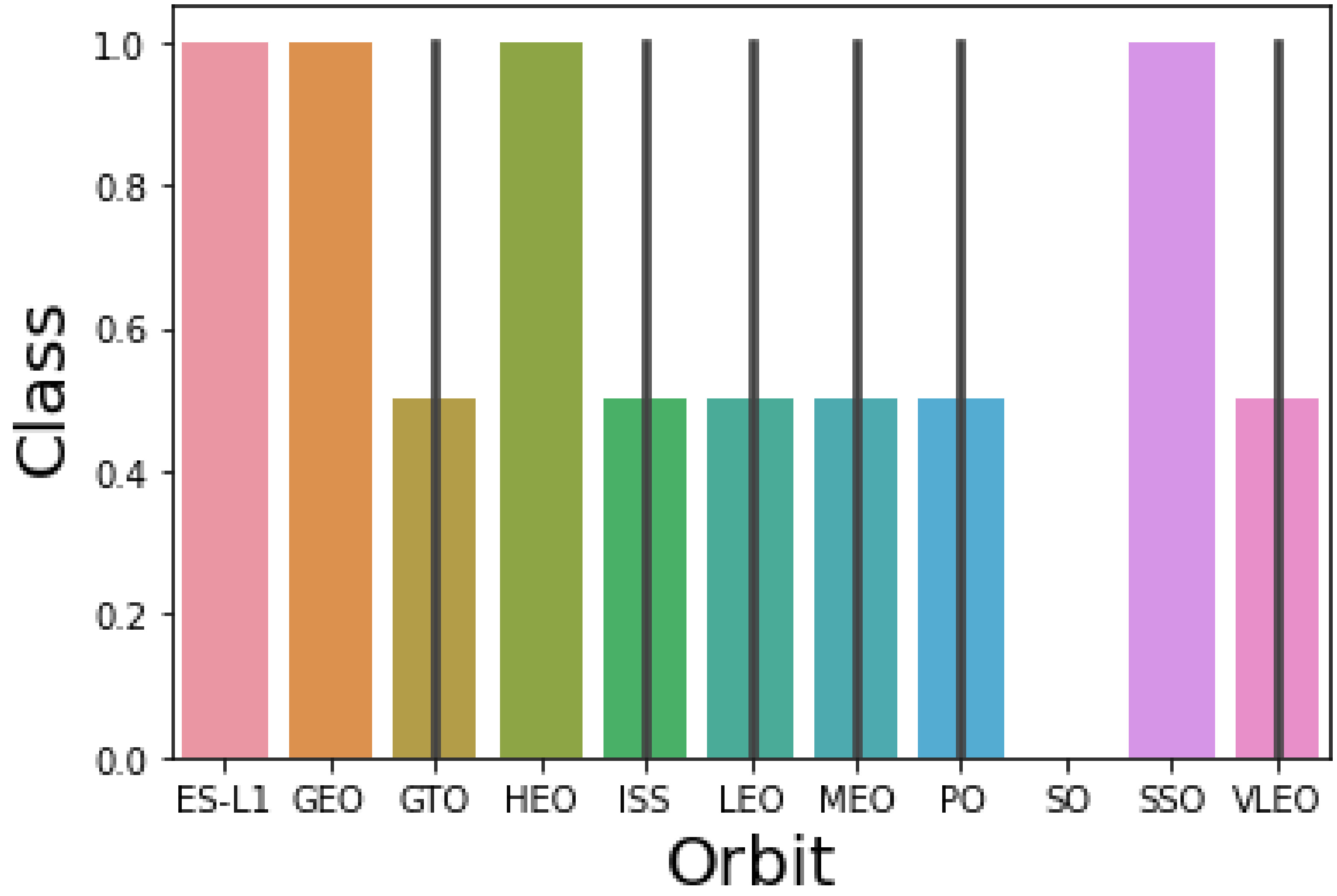
The greater number of flights at a launch site the greater the success rate at a launch site



# Payload vs. Launch Site

The greater the payload mass for Launch Site CCAFS SLC 40 the higher the success rate for the Rocket. There is not quite a clear pattern to be found using this visualization to make a decision if the Launch Site is dependent on Pay Load Mass for a success launch.



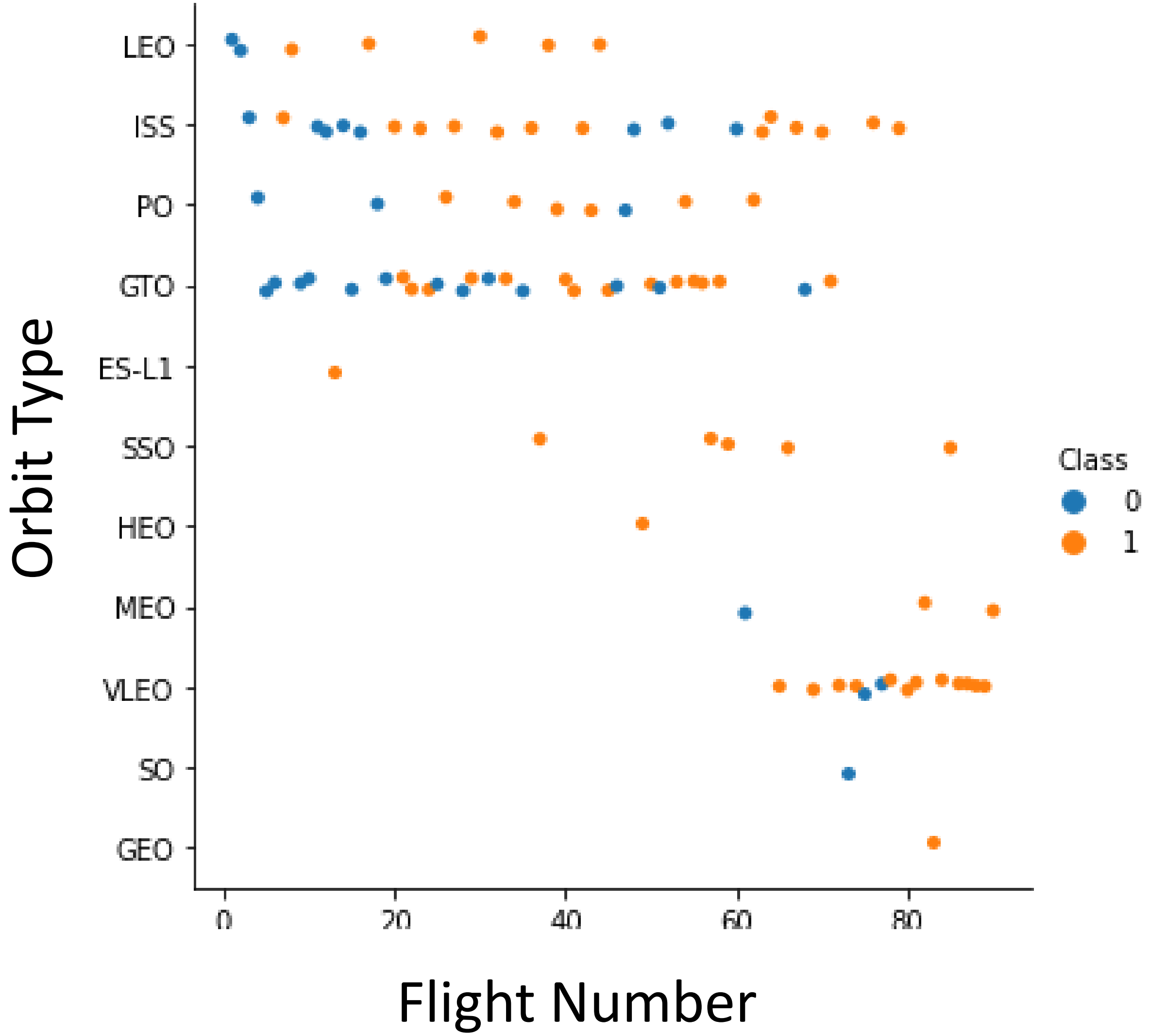


# Success Rate vs. Orbit Type

Orbit GEO,HEO,SSO,ES-L1 has the best Success Rate

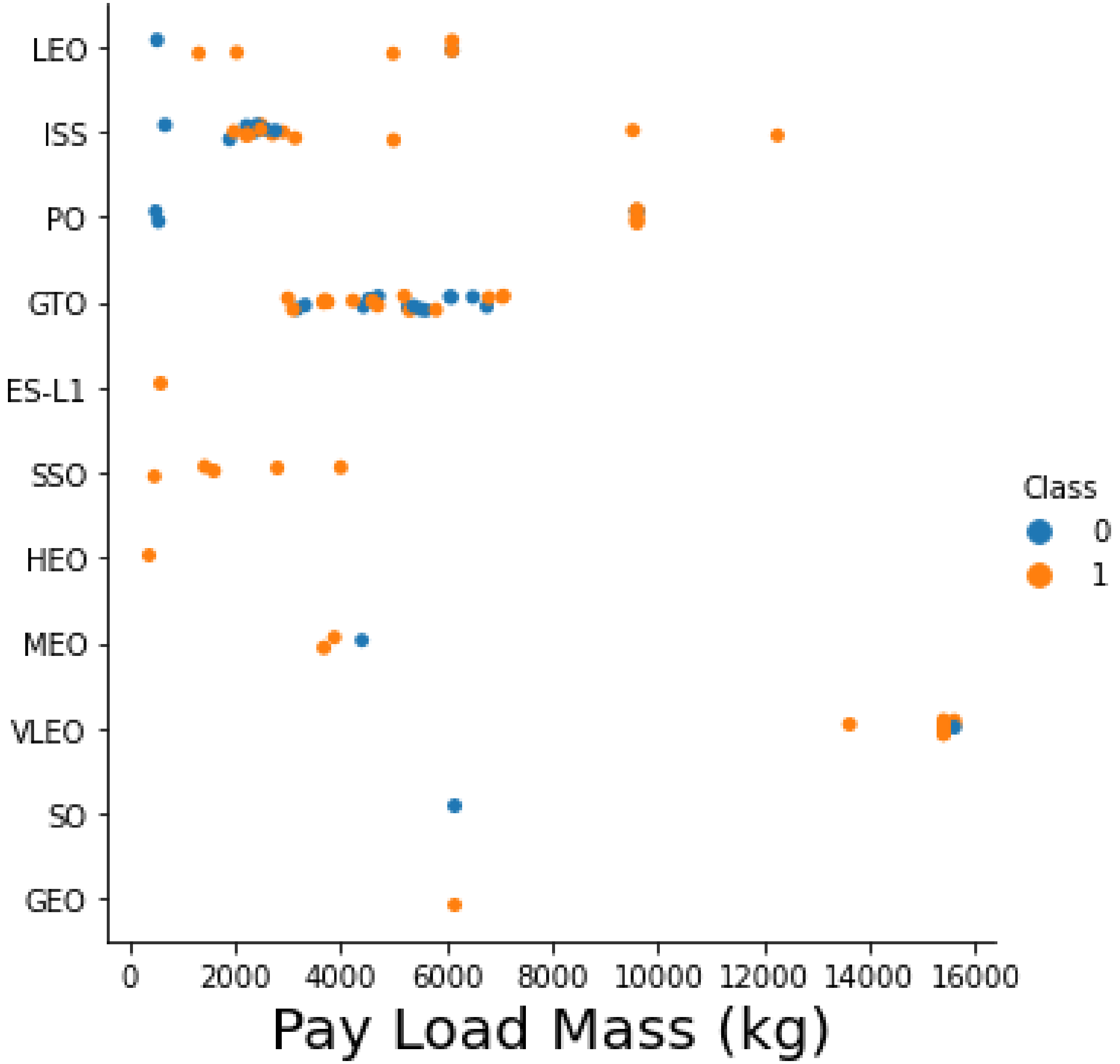
# Flight Number vs. Orbit Type

You should see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit



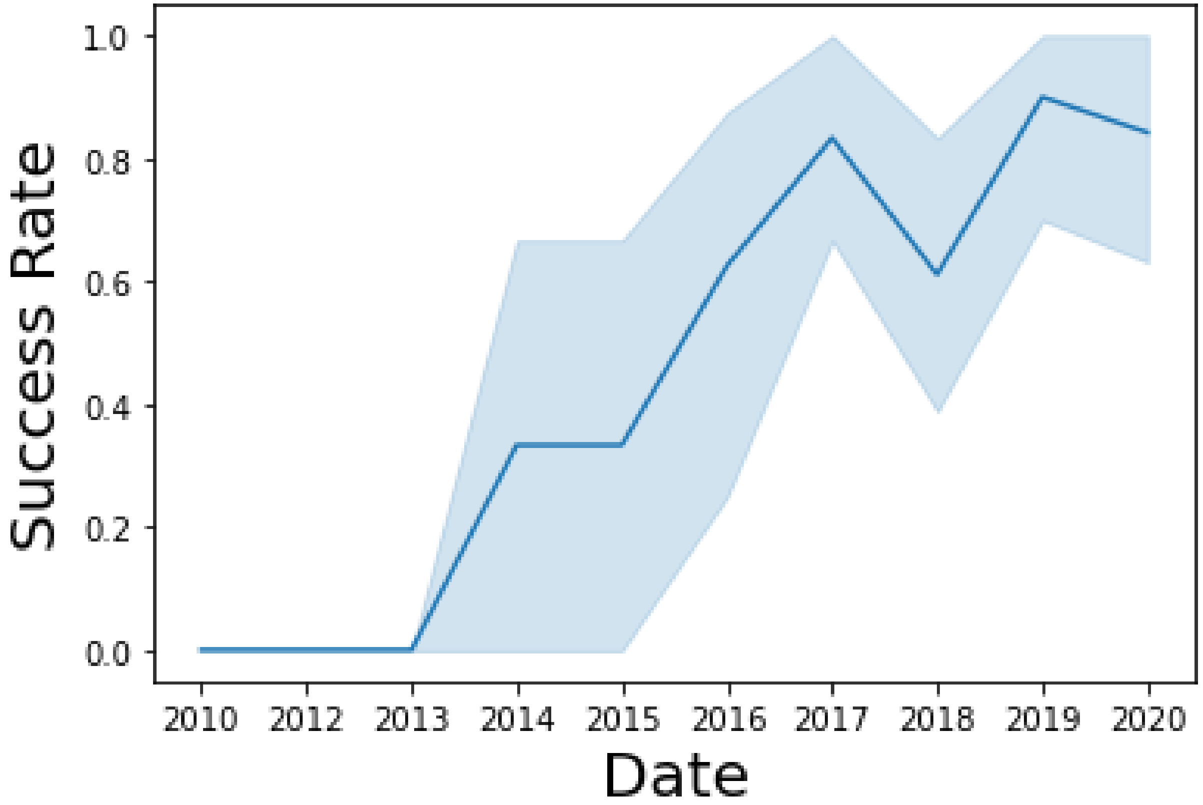
# Payload vs. Orbit Type

You should observe that Heavy payloads have a negative influence on GTO orbits and positive on GTO and Polar LEO (ISS) orbits.



# Launch Success Yearly Trend

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



# EDA with SQL

# All Launch Site Names

**SQL Query**

```
select DISTINCT Launch_Site from SPACEXTBL

select unique(launch_site) from SPACEXTBL
```

**Query Explanation**

Using the word DISTINCT in the query means that it will only show Unique values in the Launch_Site column from SPACEXTBL

**RESULT**

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

# Launch Site Names Begin with 'CCA'

**SQL Query**

```
SELECT * from SPACEXTBL where
(LAUNCH_SITE) LIKE 'CCA%' LIMIT 5
```

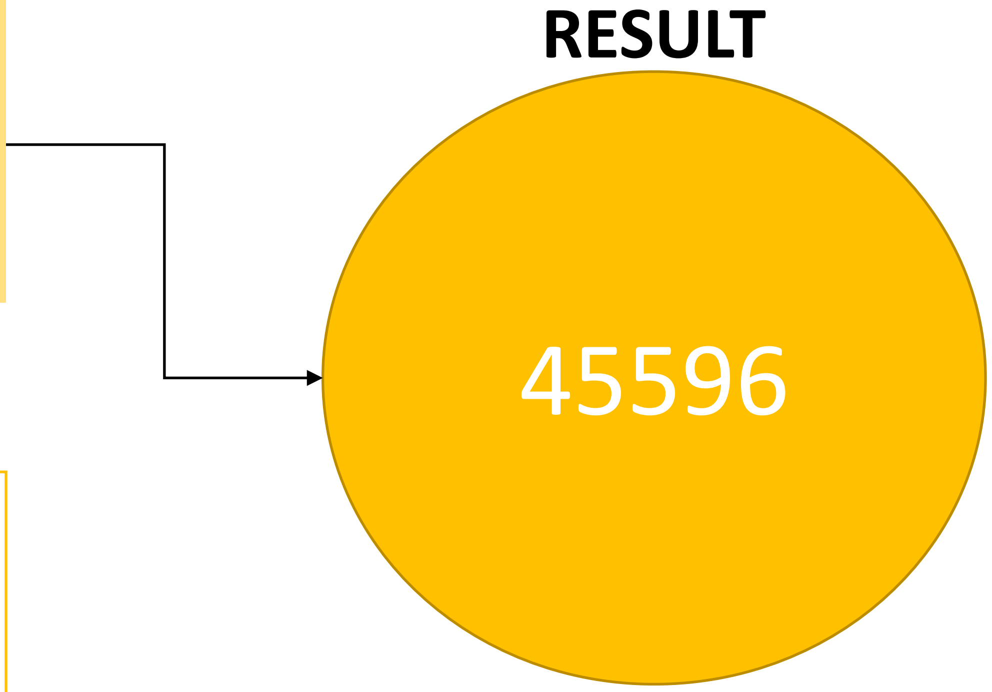
**Query Explanation**
Using the word TOP 5 in the query means that it will only show 5 records from SPACEXTBL and LIKE keyword has a wild card with the words ‘KSC%’ the percentage in the end suggests that the Launch_Site name must start with KSC.

| 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 Query**

```
SELECT sum(payload_mass__kg_) as
sum_payload from SPACEXTBL where
(customer) = 'NASA (CRS)'
```



**RESULT**

45596

**Query Explanation**

Using the function SUM summates the total in the column PAYLOAD_MASS_KG_ The WHERE clause filters the dataset to only perform calculations on Customer NASA (CRS)

# Average Payload Mass by F9 v1.1

**SQL Query**

```
SELECT avg(payload_mass__kg_) as
average_payload from SPACEXTBL where
(booster_version) = 'F9 v1.1'
```

**RESULT**

2928

**Query Explanation**

Using the function AVG works out the average in the column PAYLOAD_MASS_KG_ The WHERE clause filters the dataset to only perform calculations on Booster_version F9 v1.1

# First Successful Ground Landing Date

**SQL Query**

```
SELECT min(date) from SPACEXTBL where
landing__outcome = 'Success (ground pad)'
```

**RESULT**

**22nd of December 2015**

**Query Explanation**

Using the function MIN works out the minimum date in the column Date The WHERE clause filters the dataset to only perform calculations on Landing_Outcome Success (ground pad)

# Successful Drone Ship Landing with Payload between 4000 and 6000

**SQL Query**

```
select BOOSTER_VERSION from SPACEXTBL where
LANDING__OUTCOME='Success (drone ship)' and
PAYLOAD_MASS__KG_ BETWEEN 4001 and 5999
```

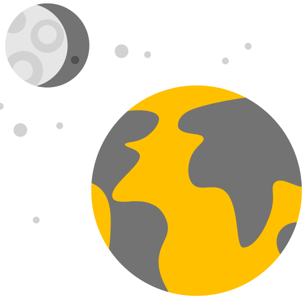
**RESULT**

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

**Query Explanation**

Selecting only Booster_Version The WHERE clause filters the dataset to Landing_Outcome = Success (drone ship) The AND clause specifies additional filter conditions Payload_MASS_KG_ > 4001 AND Payload_MASS_KG_ < 5999

# Total Number of Successful and Failure Mission Outcomes



**Query**

```
SELECT MISSION_OUTCOME,
COUNT(MISSION_OUTCOME) AS OUTCOME FROM
SPACEXTBL GROUP BY MISSION_OUTCOME
```

**Query Explanation**
A much harder query I must say, we used subqueries here to produce the results. The LIKE ‘%foo%’ wildcard shows that in the record the foo phrase is in any part of the string in the records for example. PHRASE “(Drone Ship was a Success)” LIKE ‘%Success%’ Word ‘Success’ is in the phrase the filter will include it in the dataset

| Mission outcome | Outcome |
|---|---|
| Failure (in flight) | 1 |
| Success | 99 |
| Success (payload status unclear) | 1 |

# Boosters Carried Maximum Payload

**Query**

```
SELECT BOOSTER_VERSION FROM
SPACEXTBL WHERE PAYLOAD_MASS__KG_
= (SELECT MAX(PAYLOAD_MASS__KG_)
FROM SPACEXTBL)
```

**Query Explanation**
Using the word DISTINCT in the query means that it will only show Unique values in the Booster_Version column from SPACEXTBL GROUP BY puts the list in order set to a certain condition. DESC means its arranging the dataset into descending order

| Names |
|---|
| 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 |

# 2015 Launch Records

**Query**

```
SELECT DATE, BOOSTER_VERSION,
LAUNCH_SITE, LANDING__OUTCOME
FROM SPACEXTBL WHERE
LANDING__OUTCOME = 'Failure (drone
ship)' AND YEAR(DATE) = 2015
```

**Query Explanation**

The function SELECT is used to only select Failed drone Ships. WHERE clause filters Year to be 2015.

| DATE | booster_version | launch_site | landing__outcome |
|---|---|---|---|
| 2015-01-10 | F9 v1.1 B1012 | CCAFS LC-40 | Failure (drone ship) |
| 2015-04-14 | F9 v1.1 B1015 | CCAFS LC-40 | Failure (drone ship) |

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

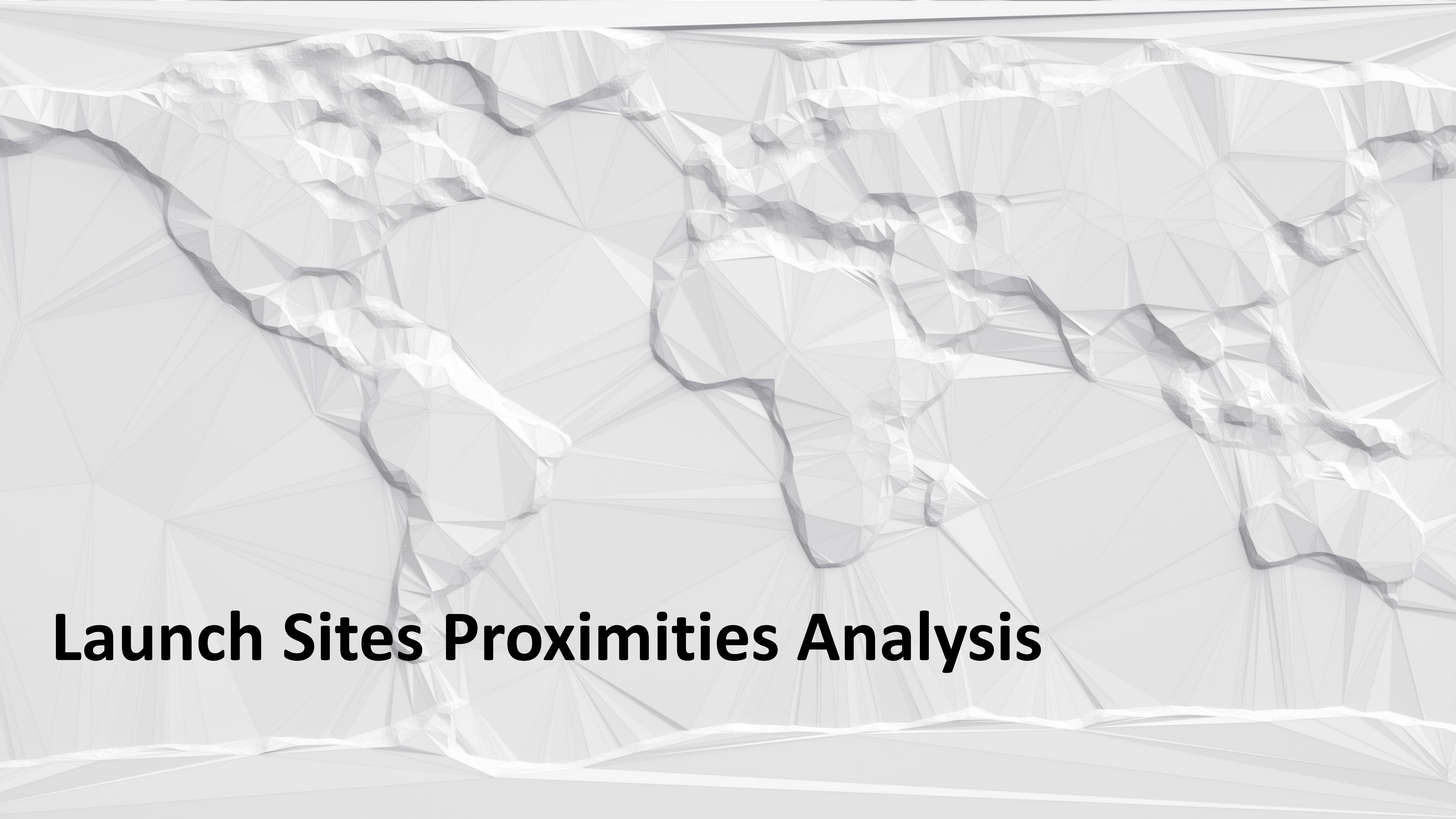
**Query**

```
SELECT LANDING__OUTCOME, COUNT(*) AS
COUNT_LAUNCHES FROM SPACEXTBL WHERE
DATE BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY LANDING__OUTCOME ORDER BY
COUNT_LAUNCHES DESC;
```

**Query Explanation**

Function COUNT counts records in column WHERE filters data LIKE (wildcard) AND (conditions) AND (conditions).

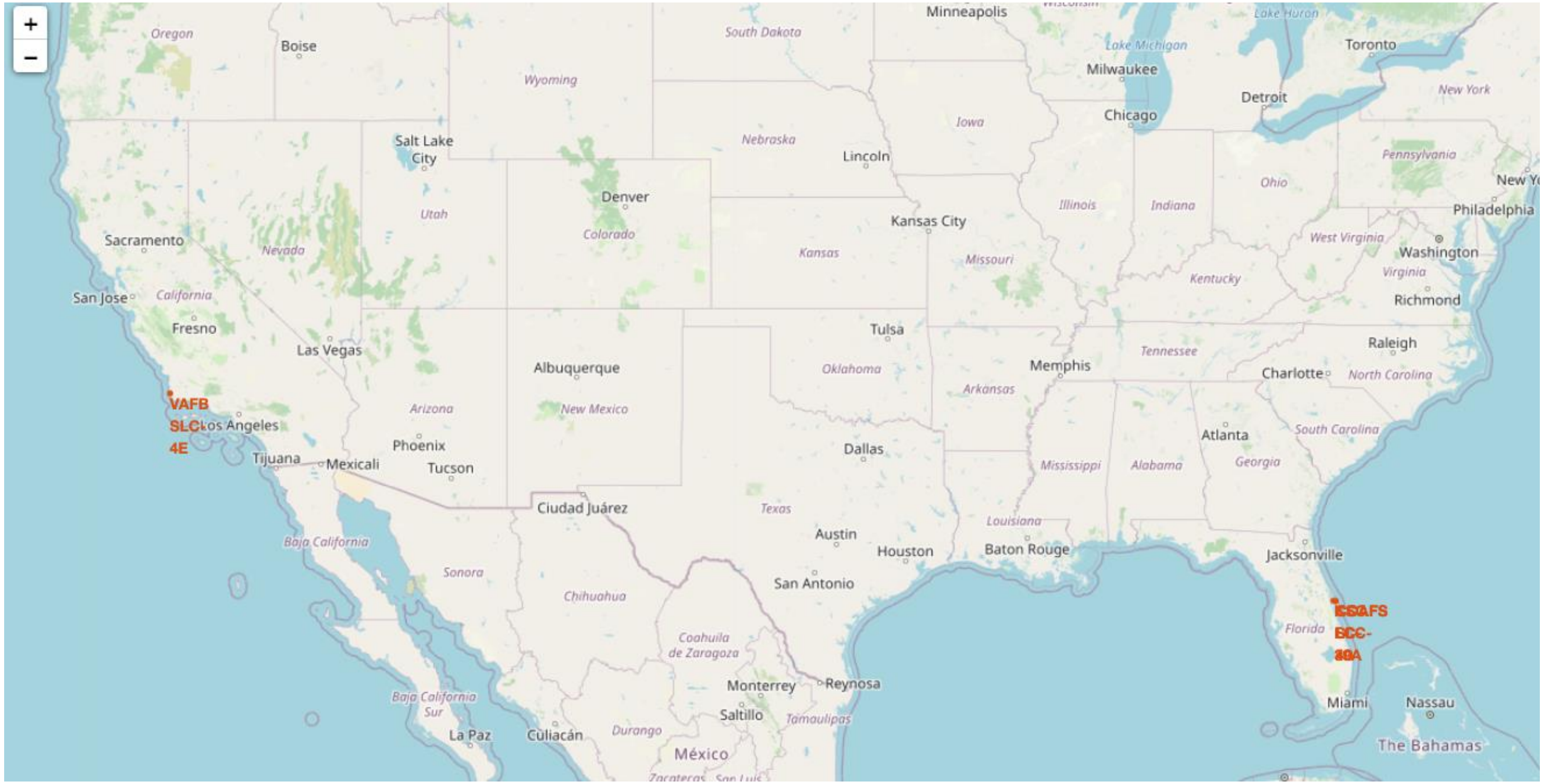
| landing__outcome | count_launches |
|---|---|
| No attempt | 10 |
| Failure (drone ship) | 5 |
| Success (drone ship) | 5 |
| Controlled (ocean) | 3 |
| Success (ground pad) | 3 |
| Failure (parachute) | 2 |
| Uncontrolled (ocean) | 2 |
| Precluded (drone ship) | 1 |



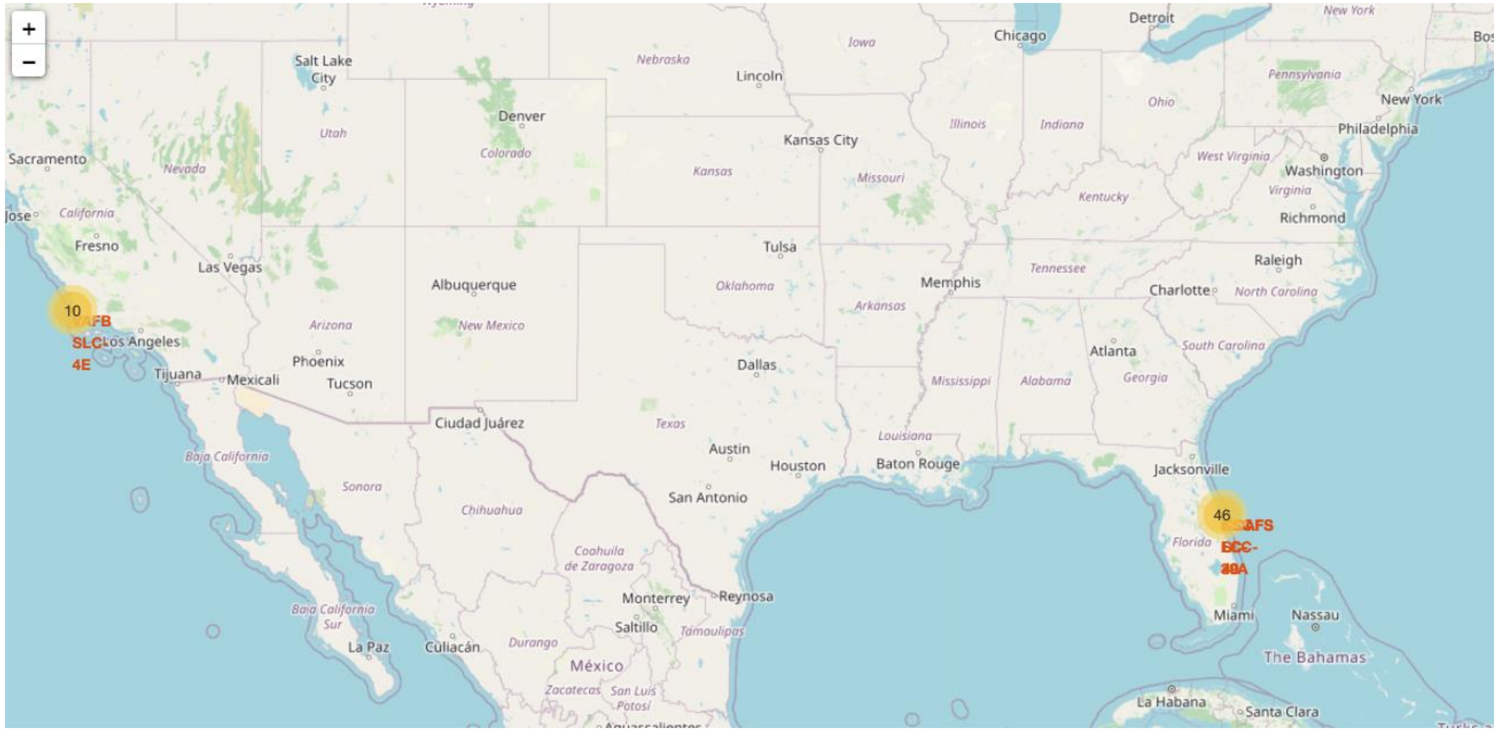
# Launch Sites Proximities Analysis

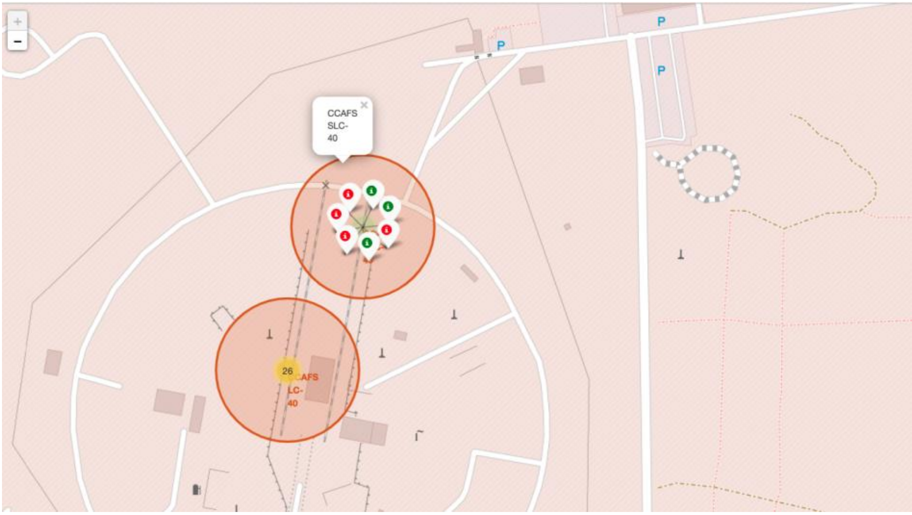
# All launch sites global map markers

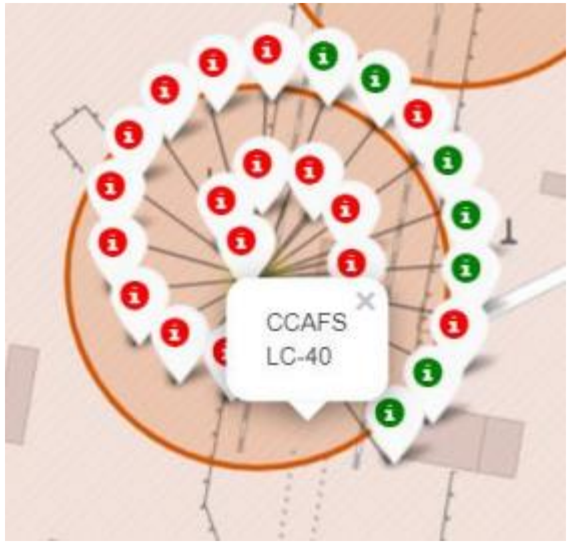
We can see that the SpaceX launch sites are in the United States of America coasts. Florida and California



# Color-labeled Launch outcomes

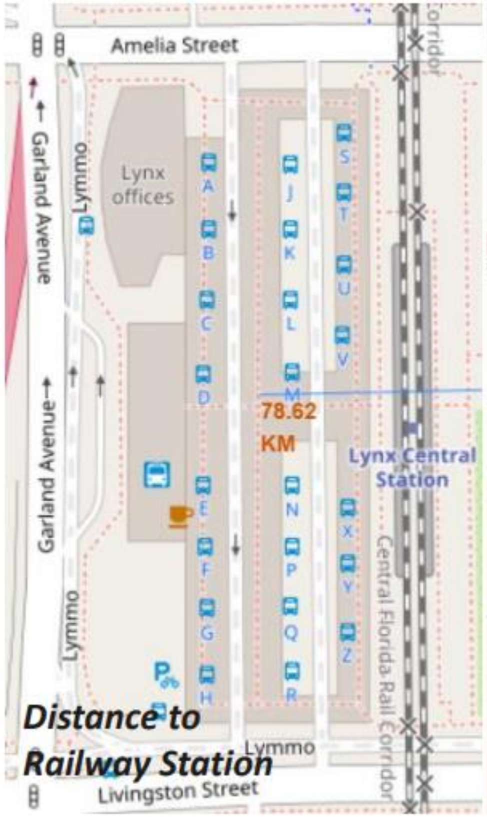


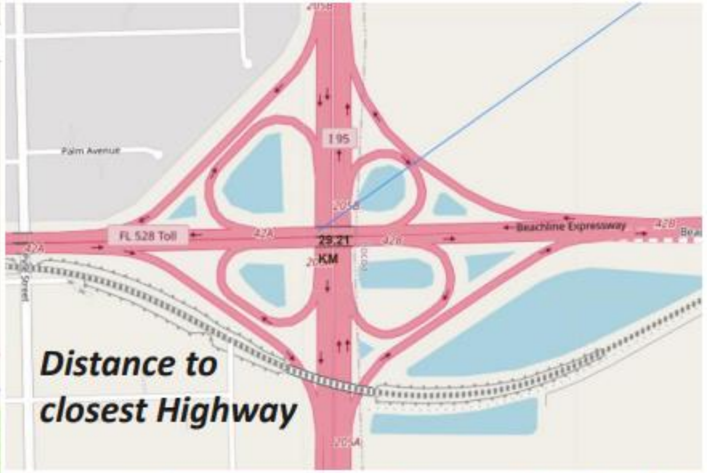


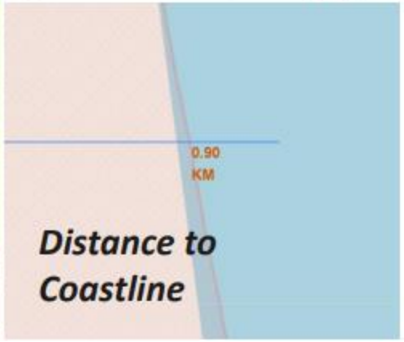


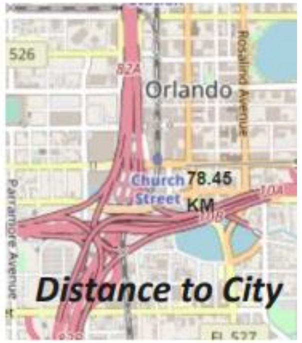
**Green Marker** shows successful Launches and **Red Marker** shows Failure

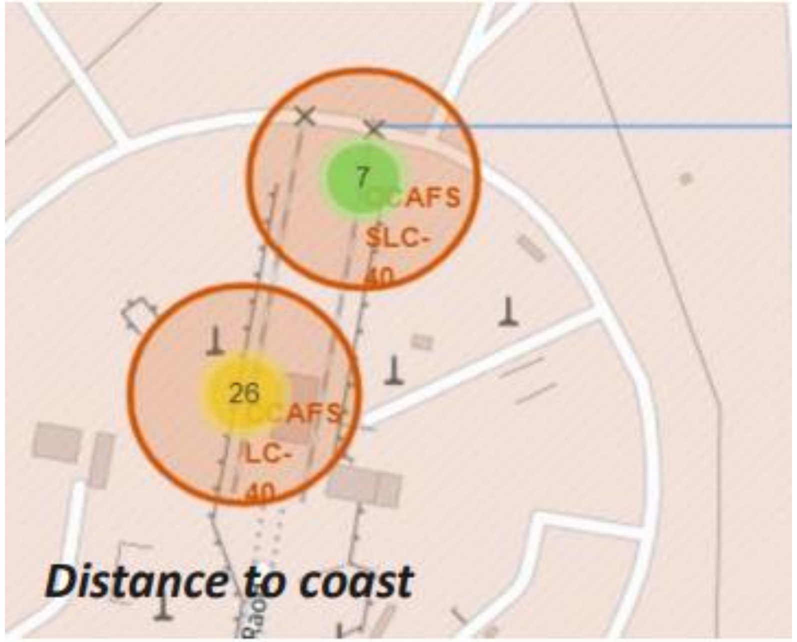
# Selected launch site to its proximities











Are launch sites in close proximity to railways? ✗

Are launch sites in close proximity to highways? ✗

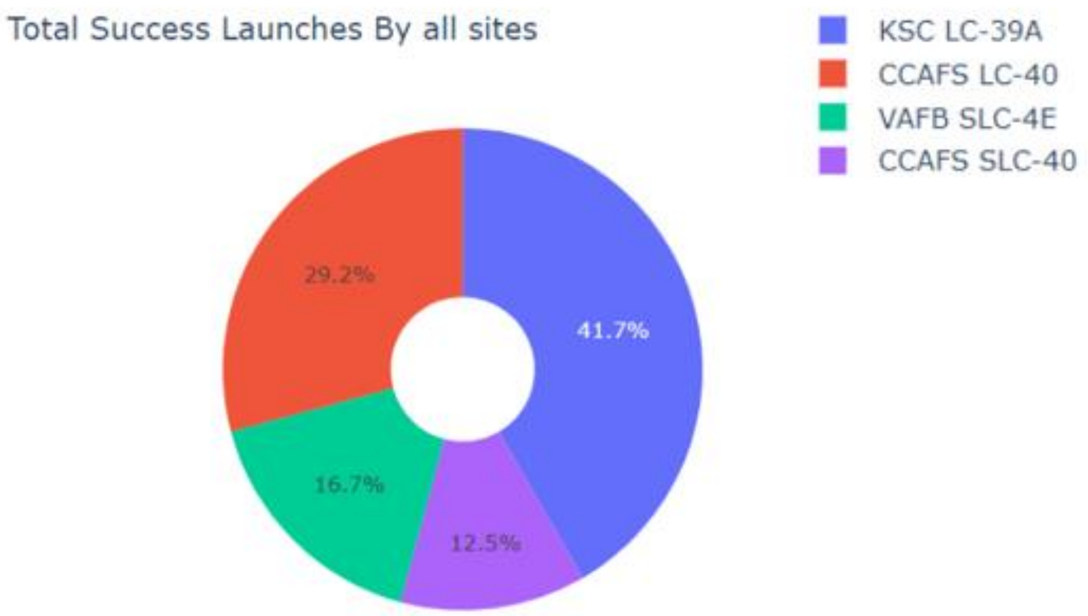
Are launch sites in close proximity to coastline? ✓

Do launch sites keep certain distance away from cities? ✓

# Build a Dashboard with Plotly Dash

## Dashboard

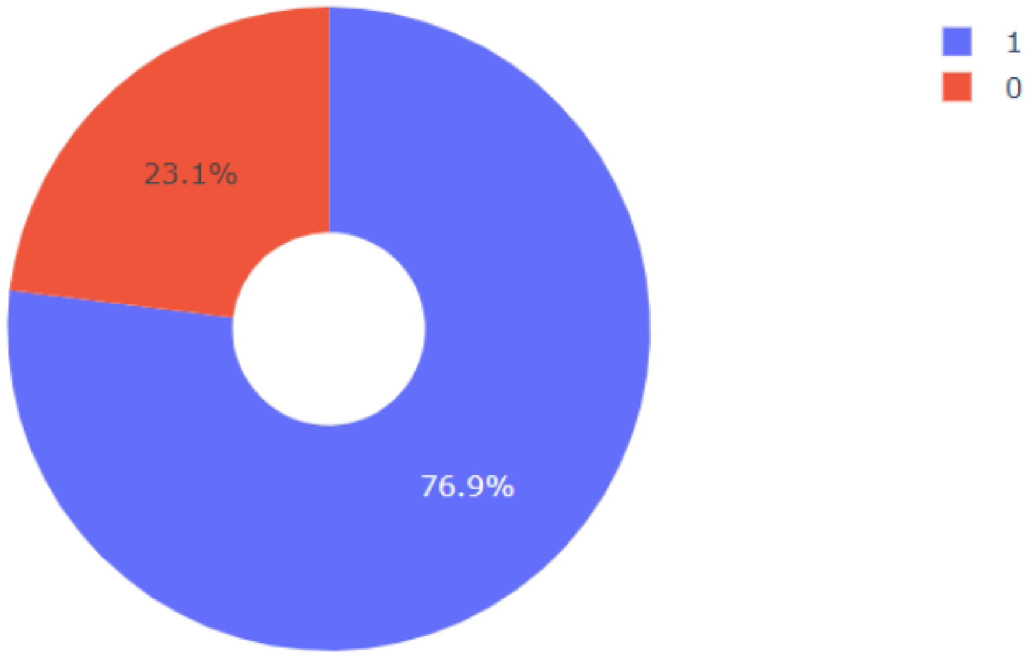
Pie chart showing the success percentage achieved by each launch site



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

## Dashboard

Pie chart for the launch site with highest launch success ratio

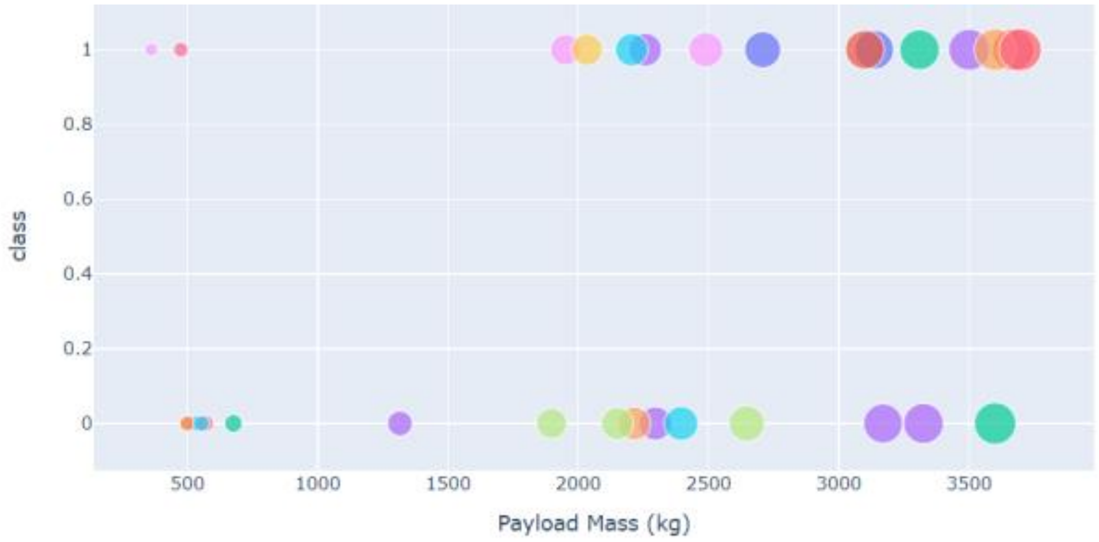


*KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate*

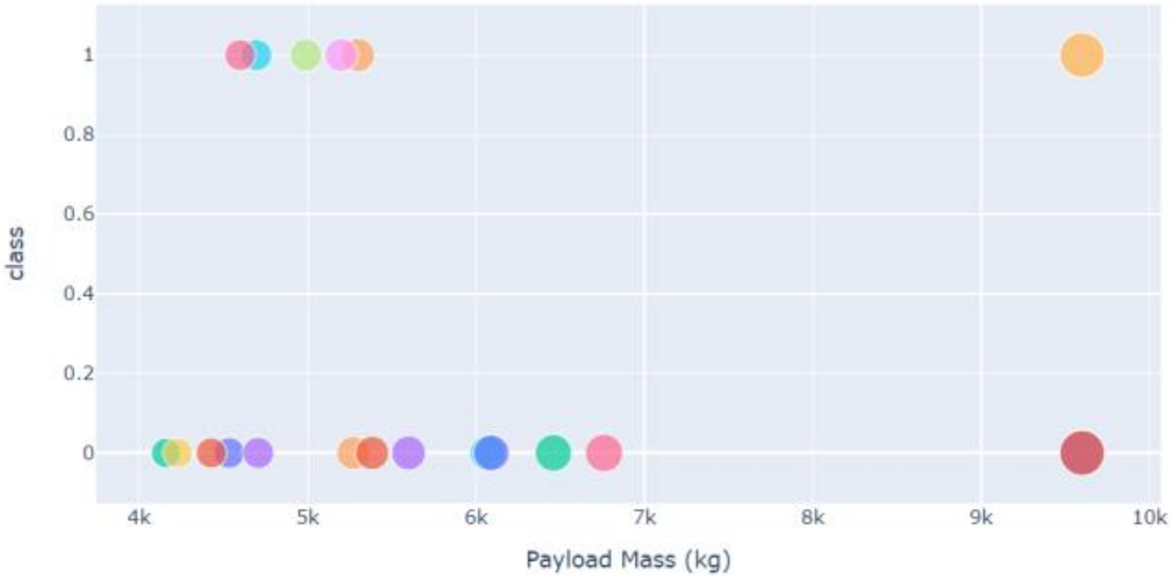
# Dashboard

## Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider

**Low Weighted Payload 0kg – 4000kg**



**Heavy Weighted Payload 4000kg – 10000kg**



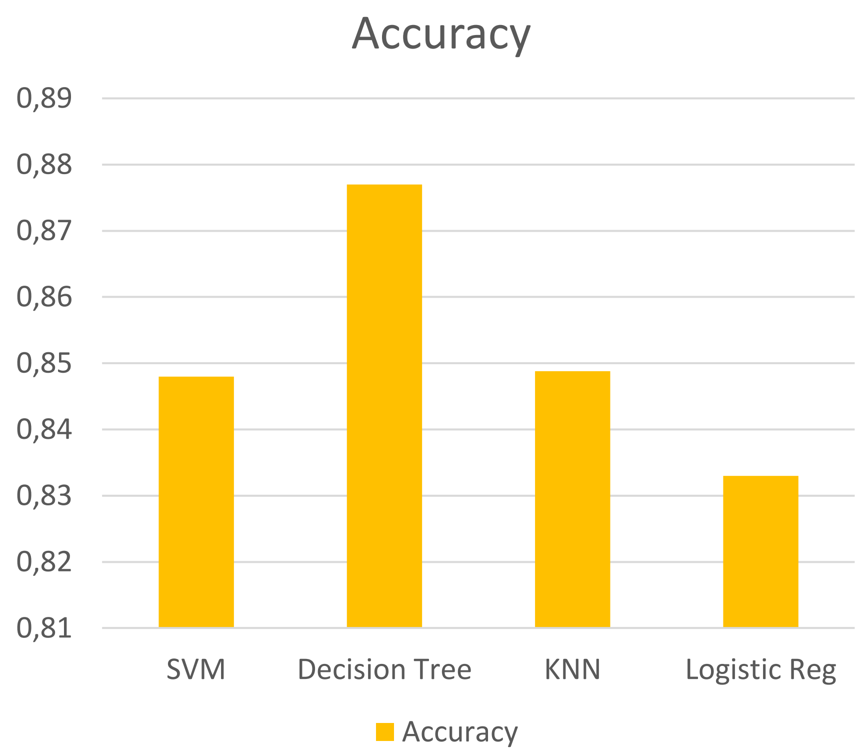
We can see the success rates for low weighted payloads is higher than the heavy weighted payloads

# Predictive analysis

(Classification)

# Classification Accuracy

As you can see the accuracy is quite close, but we do have a clear winner!

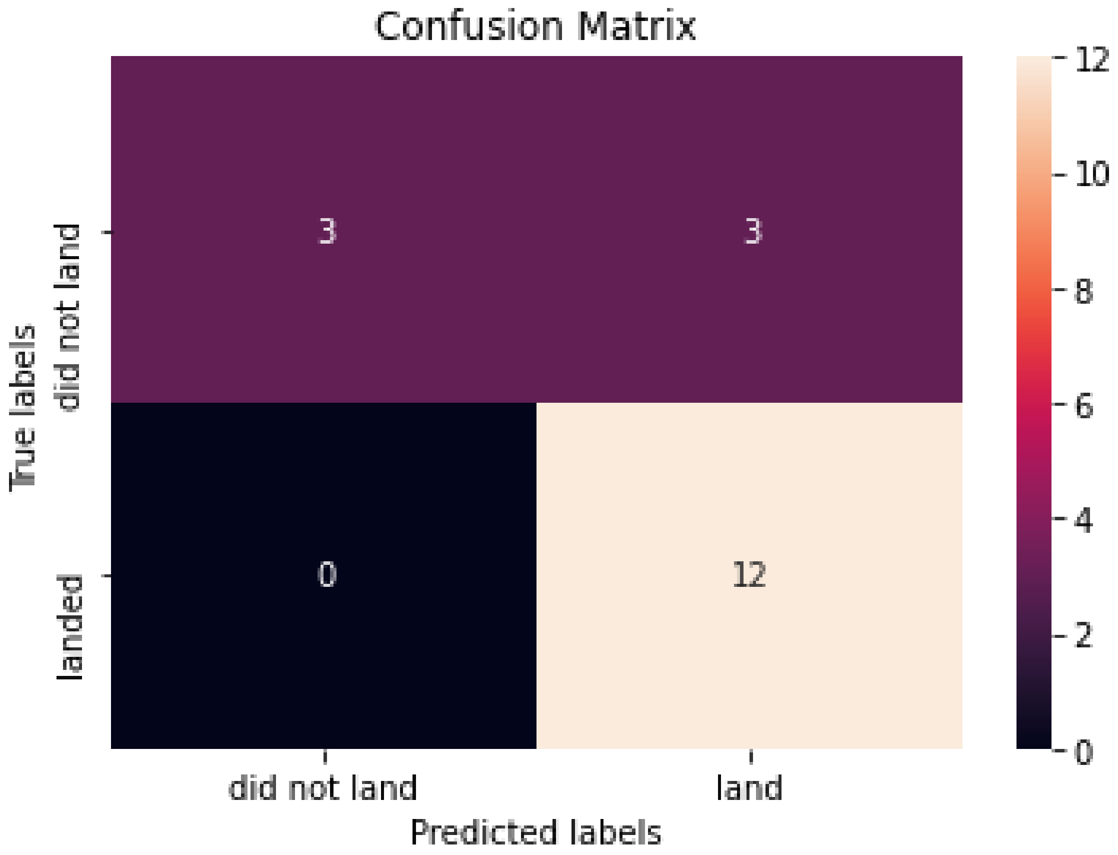


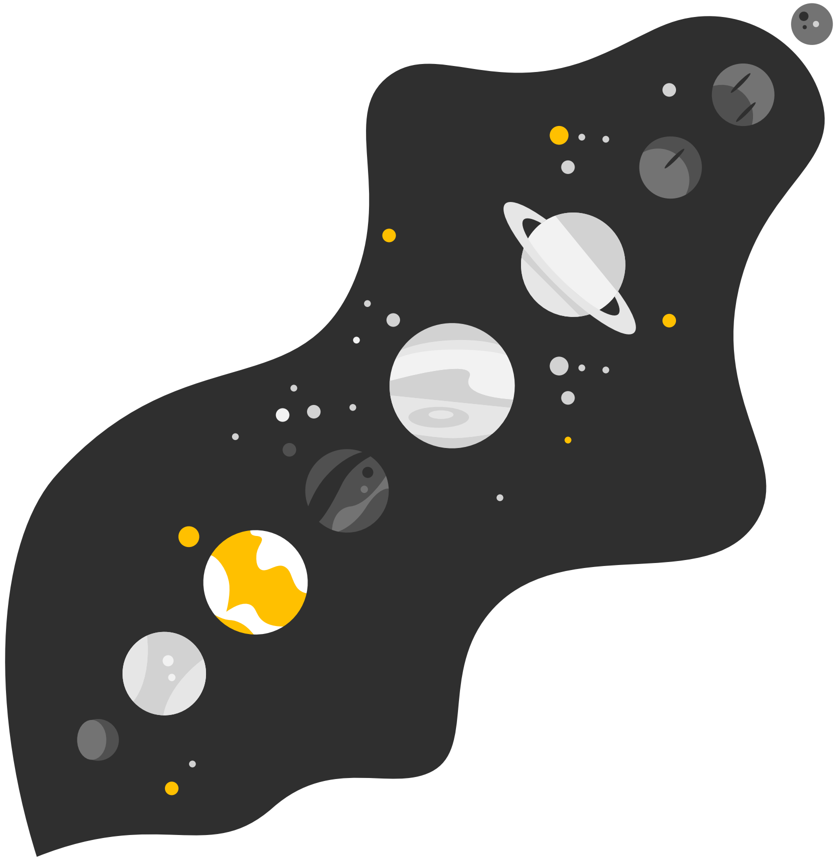
| Classifier | Accuracy score |
|---|---|
| SVM | 0.848 |
| Decision Tree | 0.877 |
| KNN | 0.8488 |
| Logistic Reg | 0.833 |

After selecting the best hyperparameters for the decision tree classifier using the validation data, we achieved 83.33% accuracy on the test data.

# Confusion Matrix

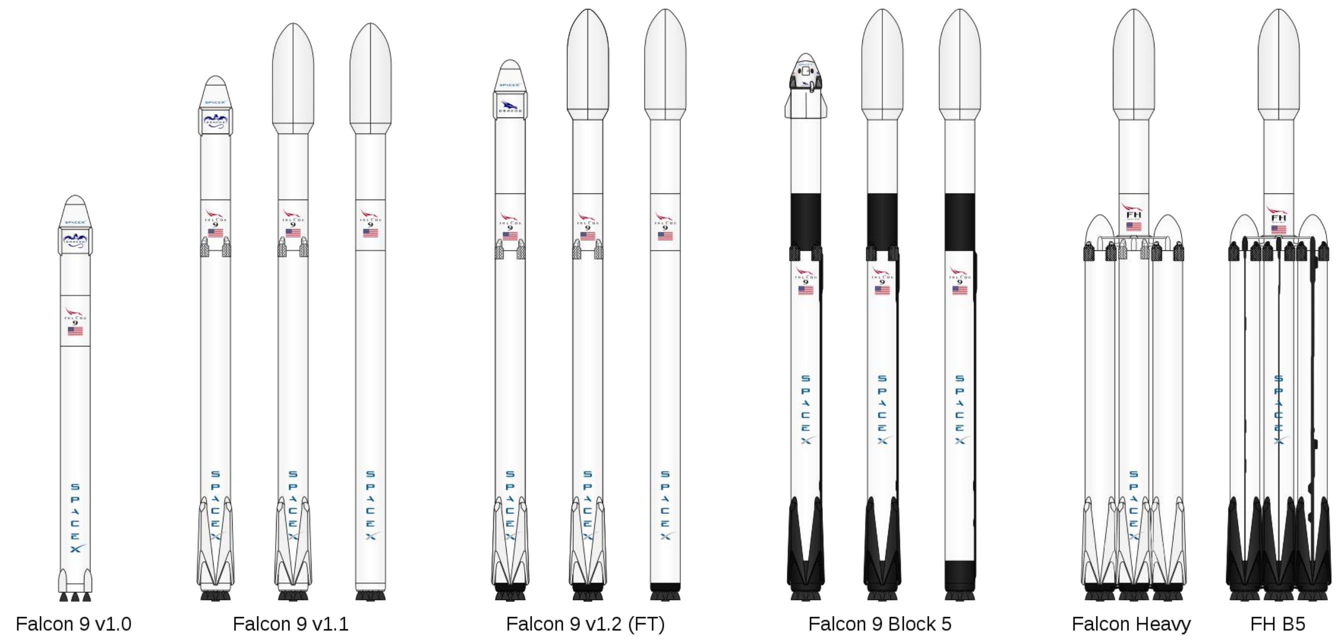
Examining the confusion matrix, we see that Tree can distinguish between the different classes. We see that the major problem is false positives





# Conclusions

- The Tree Classifier Algorithm is the best for Machine Learning for this dataset
- Low weighted payloads perform better than the heavier payloads
- The success rates for SpaceX launches is directly proportional time in years they will eventually perfect the launches
- We can see that KSC LC-39A had the most successful launches from all the sites
- Orbit GEO,HEO,SSO,ES-L1 has the best Success Rate



# Appendix

More information about SpaceX Launch can be found here:

- List of Falcon 9 and Falcon heavy Launches
- Space X Falcon 9 landing with Reinforcement Learning

# THANK YOU!