
# Winning Space Race with Data Science

Felipe Oshiro Capitelli
09/27/2021

# Outline

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

# Executive Summary

## Summary of methodologies

- Data Collection with SpaceX REST API and Web Scraping
- Data Wrangling
- EDA with Data Visualization
- EDA with SQL
- Folium Interactive Map
- Plotly-Dash Dashboard
- Predictive Analysis (Classification)

## Summary of all results

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

# Introduction

- Project background and context

This project goal is SpaceX Falcon 9 First Stage Landing Prediction. We will predict if the Falcon 9 first stage will land successfully or not. 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. So 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
  - What variables impact the success rate of the first stage landing?
    - Payload Mass, Orbit Type, Launch Site, Rocket specifications.
  - Is there any relationship between these variables?

Section 1

# Methodology

# Methodology

## Data collection methodology

- SpaceX REST API and Web Scraping

## Perform data wrangling

- Identify and handle missing values
  - Missing Payload Mass was replaced by the mean value
- Remove irrelevant data
  - Falcon 1 data
- Determine Training Labels
  - Determine what would be the label for training supervised models
- Turning Categorical values to numeric variables
  - One-hot encoding features using get_dummies function.

## Perform exploratory data analysis (EDA) using visualization and SQL

## Perform interactive visual analytics using Folium and Plotly-Dash

## Perform predictive analysis using classification models

- How to build, tune, evaluate classification models

# Data Collection

## Data Collection using SpaceX REST API

- HTTP Request data using python requests library for different end points;
- The data was obtained in a JSON format and than converted to a Pandas DataFrame using the function json_normalize;
- The data was sliced/cleaned. Irrelevant data was removed;
- The data was exported to a CSV file.

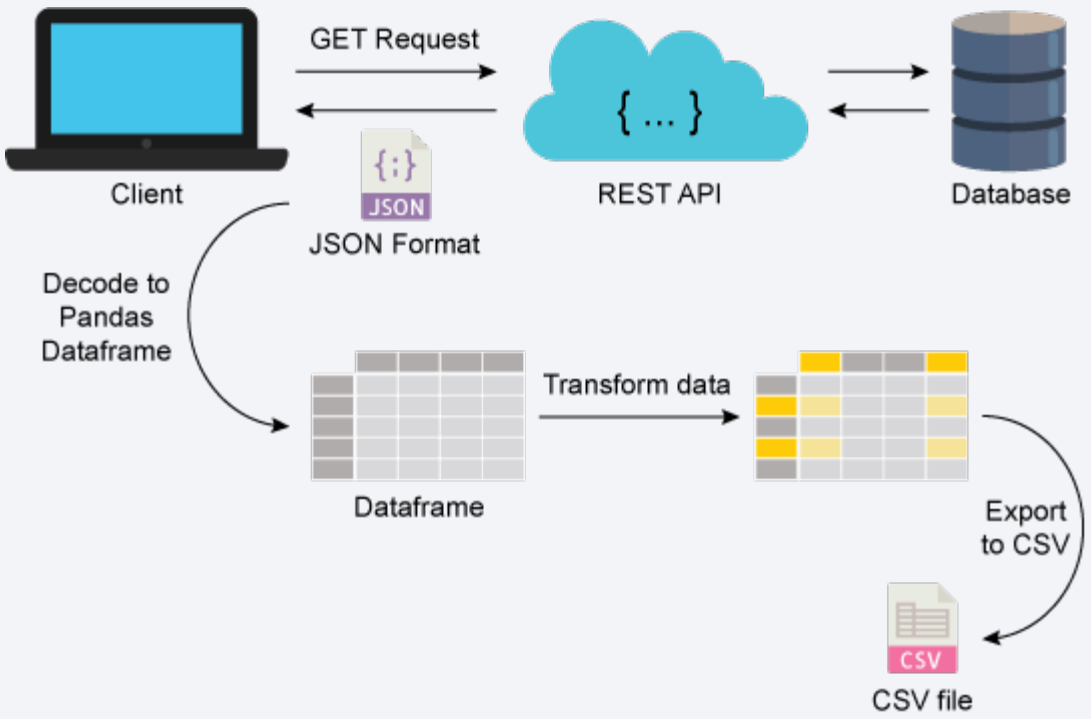
## Data Collection using Web Scraping

- HTTP Request data using python requests library;
- BeautifulSoup was used to scrap html tables from Wikipedia to obtain Falcon 9 launch records;
- The scraped data was cleaned and converted to a Pandas DataFrame;
- The data was exported to a CSV file.

# Data Collection – SpaceX API

[Access the GitHub Notebook here]

## Data Collection using SpaceX REST API – Flowchart




1. Request rocket launch data from SpaceX API (HTTP Request)

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

2. Decode the response content as a JSON and turn it into a Pandas dataframe

```
data = pd.json_normalize(response.json())
```

3. Transform the data by slicing/converting and applying functions

```
#This code has been omitted for being too big.
#Check the full code in block 13 from GitHub Notebook nº 1.
```

4. Get information about the launches using the IDs given for each launch and custom functions

```
getBoosterVersion(data); getLaunchSite(data); getPayloadData(data); getCoreData(data);
```

5. Create the dictionary `launch_dict` from the collected data

```
#This code has been omitted for being too big.
#Check the full code in block 21 from GitHub Notebook nº 1.
```

6. Create a Pandas data frame from the dictionary `launch_dict`

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

7. Filter the dataframe to only include Falcon 9 launches

```
data_falcon9 = data_spacex[data_spacex["BoosterVersion"]=="Falcon 9"]
```

8. Dealing with Missing Values

```
# Calculate the mean value of PayloadMass column
avg_PayloadMass = data_falcon9["PayloadMass"].astype("float").mean(axis=0)
# Replace the np.nan values with its mean value
data_falcon9["PayloadMass"].replace(np.nan, avg_PayloadMass, inplace = True)
```

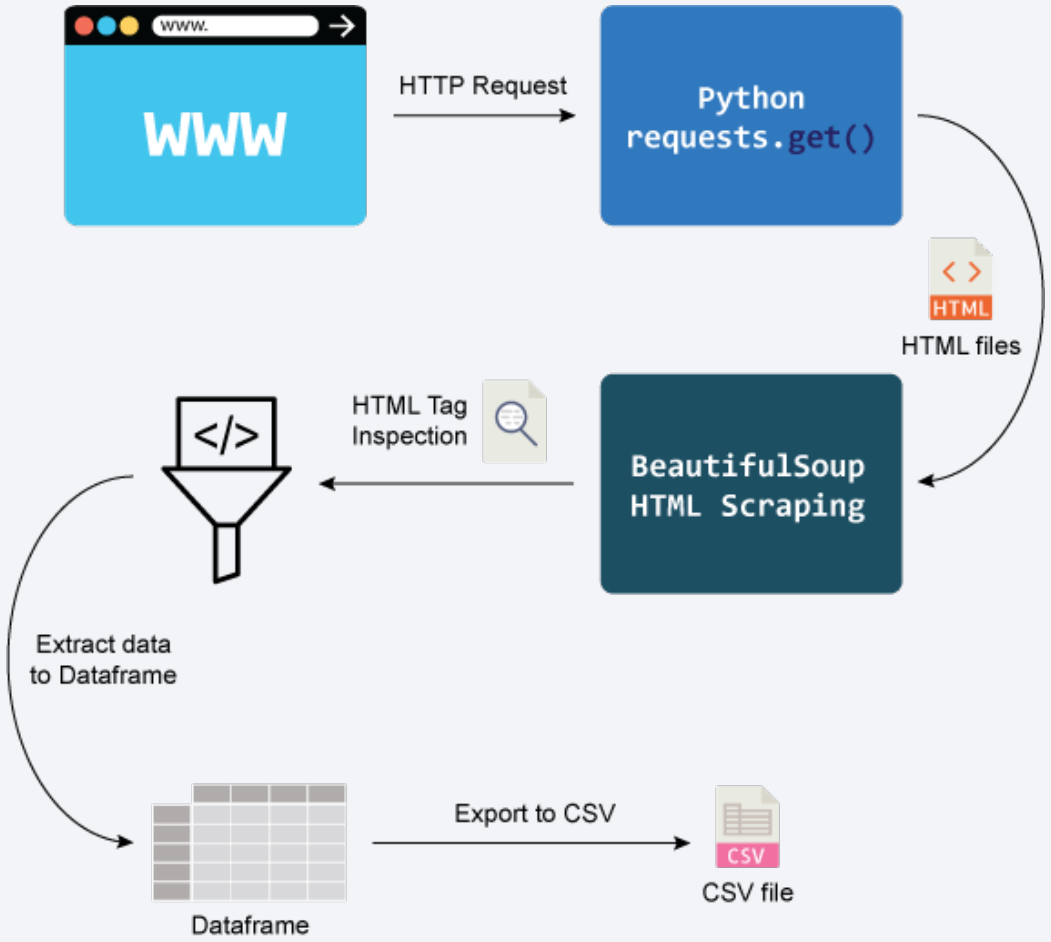
9. Export to CSV

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

# Data Collection – Web Scraping

[Access the GitHub Notebook here]

## Data Collection using Web Scraping – Flowchart




1. Getting HTTP response from the HTTP GET method

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

2. Create a BeautifulSoup object from a response text content

```
soup = BeautifulSoup(response, 'html5lib')
```

3. Assign the result to a list called `html_tables`

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

4. Iterate each th element and apply the provided `extract_column_from_header()` to get a column name

```
column_names = []
for i in range(len(first_launch_table.find_all("th"))):
    name = extract_column_from_header(first_launch_table.find_all("th")[i])
    if (name != None) and (len(name) > 0):
        column_names.append(name)
```

5. Create an empty dictionary with keys from the extracted column names

```
#This code has been omitted for being too big.
#Check the full code in block 12 from GitHub Notebook nº 2.
```

6. Filling up the `launch_dict` with launch records extracted from table rows

```
#This code has been omitted for being too big.
#Check the full code in block 12 from GitHub Notebook nº 2.
```

7. Create dataframe from dictionary

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

8. Export to CSV

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

# Data Wrangling – Part 1/2

Slides 10 – 11

Access the GitHub Notebook here

During the process of collecting and performing an Exploratory Data Analysis (EDA) the data was cleaned and transformed using the following techniques:

- Identify and handle missing values
  - Missing Payload Mass was replaced by the mean value

```
# Identify null values in the dataframe
data_falcon9.isnull().sum()
```

```
# Calculate the mean value of PayloadMass column
avg_PayloadMass = data_falcon9["PayloadMass"].astype("float").mean(axis=0)
# Replace the np.nan values with its mean value
data_falcon9["PayloadMass"].replace(np.nan, avg_PayloadMass, inplace = True)
```

- Remove irrelevant data
  - Falcon 1 data

```
# Filter the dataframe to only include Falcon 9 launches
data_falcon9 = data_spacex[data_spacex["BoosterVersion"]=="Falcon 9"]
```

# Data Wrangling – Part 2/2

Slides 10 – 11

Access the GitHub Notebook here

- Determine Training Labels
  - Determine what would be the label for training supervised models

```
# Create a landing outcome label from Outcome column
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
bad_outcomes = set(landing_outcomes.keys()[[1,3,5,6,7]])
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for outcome in df["Outcome"]:
    if outcome in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
df['Class'] = landing_class
```

- Turning Categorical values to numeric variables
  - One-hot encoding features using get_dummies function.

```
# Create dummy variables to categorical columns
features_one_hot = pd.get_dummies(features[["Orbit","LaunchSite","LandingPad","Serial"]])
features_one_hot = features.join(features_one_hot)
features_one_hot.drop(["Orbit","LaunchSite","LandingPad","Serial"], axis=1, inplace=True)
```

# EDA with Data Visualization

Access the GitHub Notebook here

A **scatter plot** uses dots to represent values for two different numeric variables. The position of each dot on the horizontal and vertical axis indicates values for an individual data point. Scatter plots are used to observe relationships between variables (Chartio, 2019).

- Flight Number vs. Payload Mass
- Flight Number vs. Launch Site
- Payload Mass vs. Launch Site
- Flight Number vs. Orbit
- Payload Mass vs. Orbit

A **bar chart** is used when you want to show a distribution of data points or perform a comparison of metric values across different subgroups of your data. From a bar chart, we can see which groups are highest or most common, and how other groups compare against the others (Chartio, 2019).

- Orbit vs. Success Rate

A **line chart** uses points connected by line segments from left to right to demonstrate changes in value (Chartio,2019).

- Year vs. Success Rate

# EDA with SQL

Access the GitHub Notebook here

List of SQL queries performed to help understand the SpaceX Dataset:

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

Access the GitHub Notebook here

Slides 14 – 15

The launch success rate may depend on many factors such as payload mass, orbit type, and so on. It may also depend on the location and proximities of a launch site, i.e., the initial position of rocket trajectories. The purpose of this study is to find some geographical patterns about launch sites by analyzing the launch site locations used by SpaceX.

- Each site's location were added on a map using site's latitude and longitude coordinates by creating a **folium Map object** and adding **Circles** and **Markers** to highlight its location. As the launches happens only in 4 locations, which means that many launch records will have the exact same coordinate so **Marker Clusters** was applied to simplify the visualization.
- To differentiate successful launches than failed the Marker color green was assigned to the launches with the class = 1 (successful) and red to the class = 0 (failed).
- To calculate the distance between two points on the map the haversine formula was applied and the folium PolyLine function was used to visualize the path.

# Build an Interactive Map with Folium

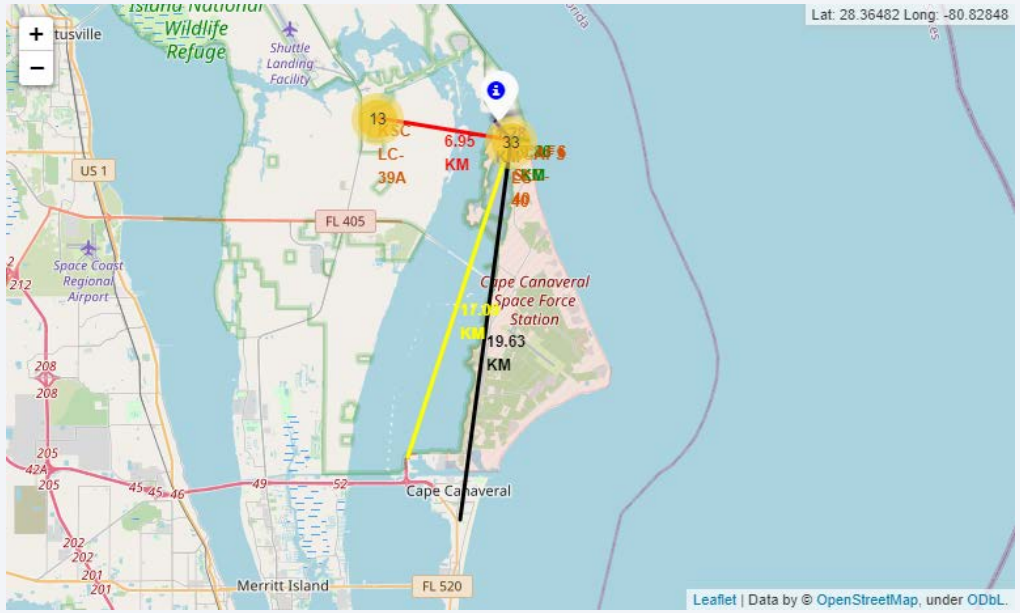
[Access the GitHub Notebook here]

Slides 14 – 15

With this method we were able calculate the distances bellow and have the following insights:

- CCADS SLC-40 to KSC LC-39A: ~ 7km
- CCADS SLC-40 to Cape Canaveral City: ~20km
- CCADS SLC-40 to the coastline: ~0.9km
- CCADS SLC-40 to the closest railway (NASA Railroad): ~1.3km
- CCADS SLC-40 to the closest highway (FL 401): ~17km

- Launch site close to Railways (Only NASA Railroad)
- Launch site close to Coastline
- Launch site not close to Highways
- Launch site keep certain distance away from cities



# Build a Dashboard with Plotly-Dash

Access the GitHub Python file here
Access the Dashboard on Heroku here

The Plotly-Dash interactive dashboard was deployed on the Heroku platform and can be accessed at the link above

The dashboard has 4 elements:

- A drop-down input was added to be able to select a specific launch site
- A pie chart graph was added to show the success and failed count for each launch site
  - To observe if a specific launch site has a higher successful launch rate.
- A range slider to select the payload mass of each launch site
  - To find if the variable payload mass is correlated to the mission outcome.
- A scatter plot with the x-axis to be payload mass and y-axis to be the launch outcome and color-labeled the Booster version on each scatter point
  - To observe how payload may be correlated with mission outcomes for a selected site;
  - To observe mission outcomes with different boosters.

# Predictive Analysis (Classification)

Access the GitHub Notebook here

## CLASSIFICATION MODEL PIPELINE

- Load the data into a Pandas DataFrame
- Transform/Standardize the data
- Split the data into training and test data
- Create/Tuning a classification model
  - Fit the model object with the training dataset
  - Find the best hyperparameters for the model using Grid Search
- Validate/Evaluate the model using the test dataset
  - Calculate the model's accuracy
  - Plot a confusion matrix

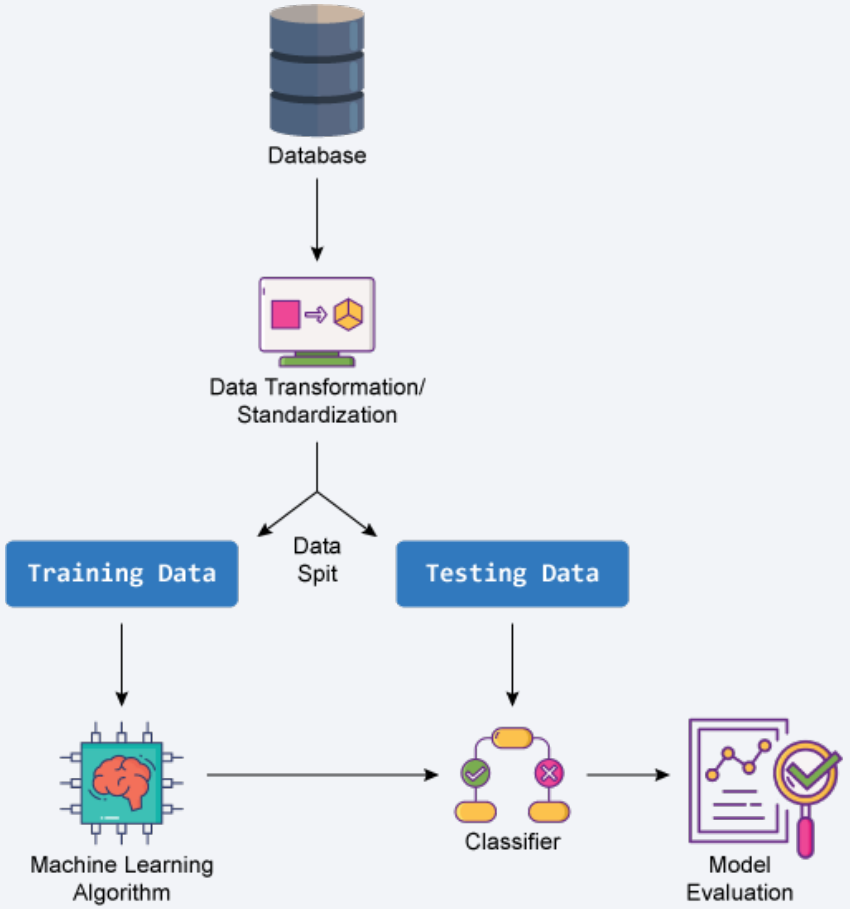
## CLASSIFICATION MODELS CREATED

- Logistic Regression
- Support Vector Machine
- Decision Tree
- K-Nearest Neighbors

## BEST PERFORMING CLASSIFICATION MODEL

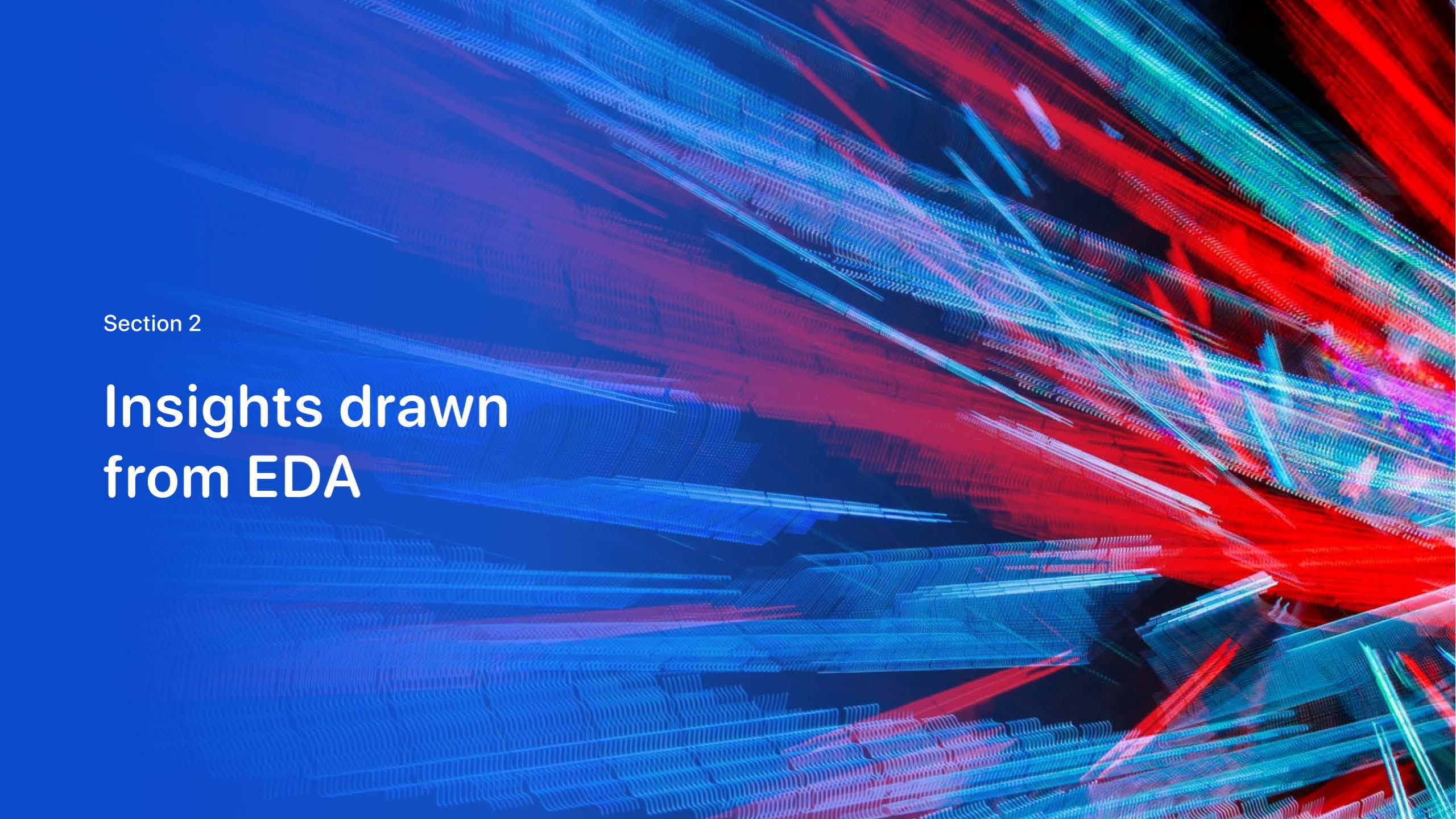
- The best performing model was chosen by evaluating the model's accuracy

## CLASSIFICATION MODEL PIPELINE FLOWCHART



# Results

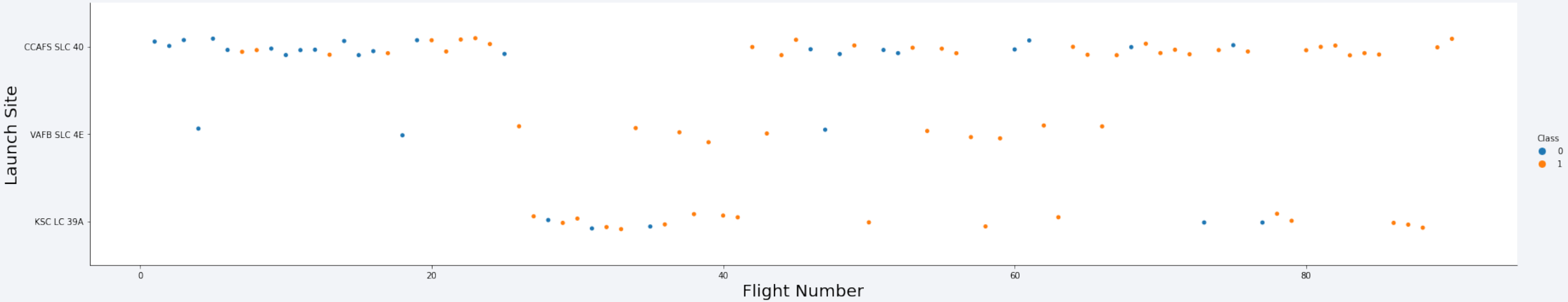
- Exploratory Data Analysis (EDA) results
- Interactive analytics demo in screenshots
- Predictive analysis results



Section 2

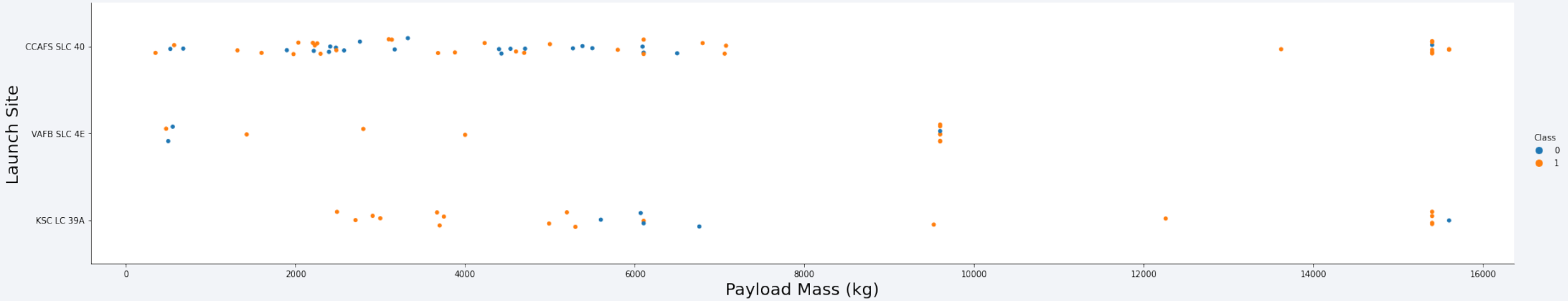
# Insights drawn from EDA

# Flight Number vs. Launch Site




- The number of successful launches increased with the number of launches;
- Successful launches have a positive correlation with the amount of launches.

# Payload vs. Launch Site

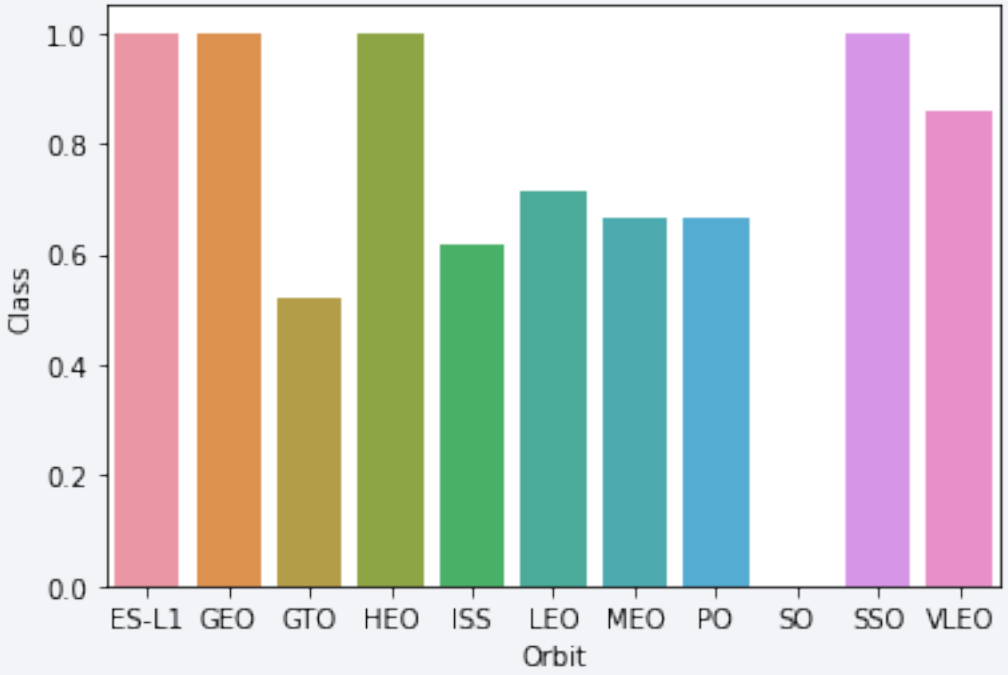



- The observed behavior was that we can have a positive correlation between the payload mass and success rate but as the payload increases we have fewer points to analyze;
- The CCADS SLC 40 launch site has most of the number of launches and successful and failed launches are almost equally distributed with the payload mass;
- The majority of failed launches happened with a payload mass of around 6,000 kg for both the KSC LC 39A and CCADS SLC 40 launch sites.

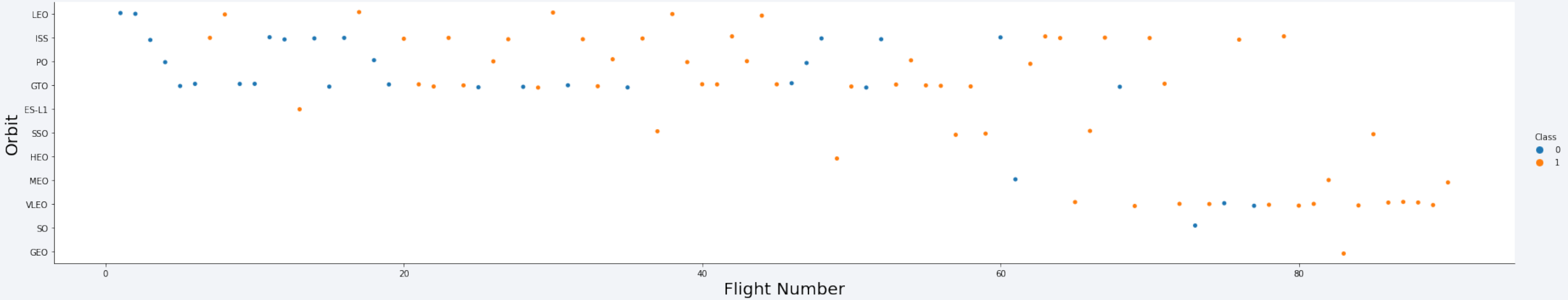
# Success Rate vs. Orbit Type

The orbits with highest success rate were:

- ES-L1
- GEO
- HEO
- SSO

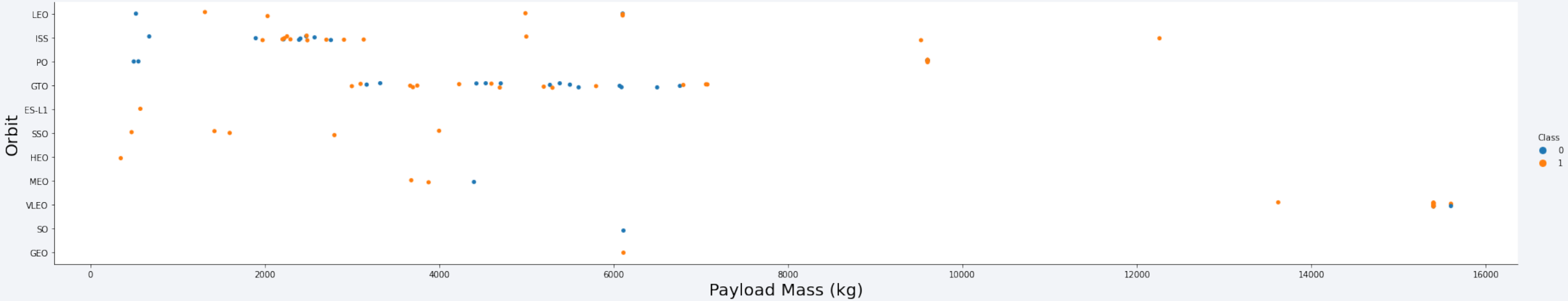


# Flight Number vs. Orbit Type




- The LEO orbit successful outcome appears related to the number of flights;
- There seems to be no relationship between flight number when in GTO orbit.

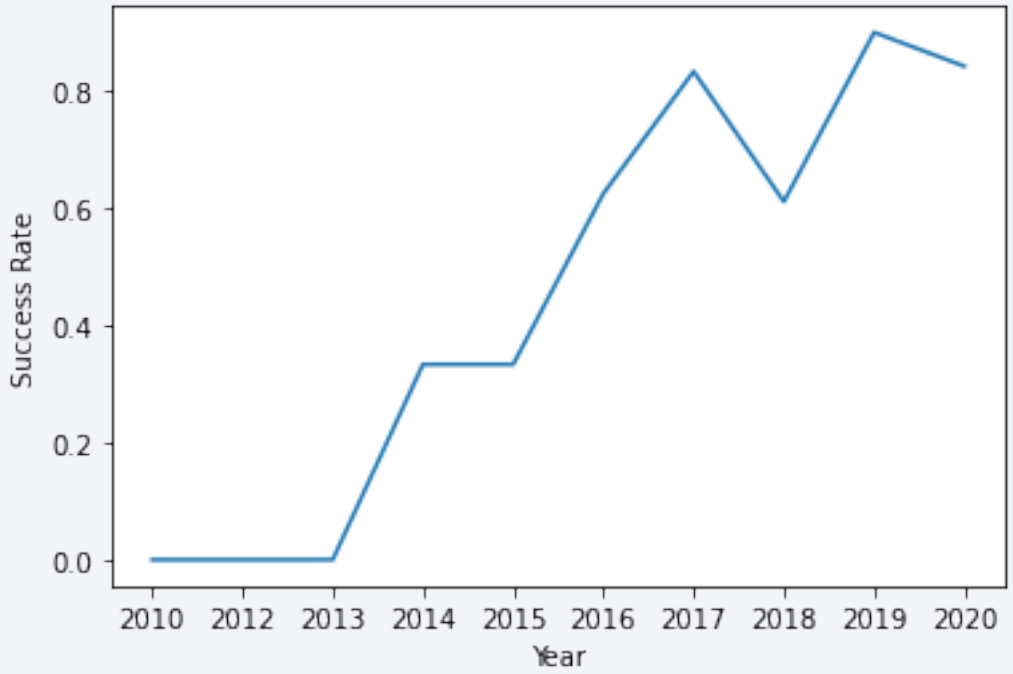
# Payload vs. Orbit Type




- Heavy payloads have a negative influence on GTO orbits and positive on LEO, ISS and PO.

# Launch Success Yearly Trend

- The success rate kept increasing since 2013 till 2020.



# All Launch Site Names

## SQL Query:

**INPUT:**

```
SELECT UNIQUE(launch_site) FROM SPACEXDATASET
```

**OUTPUT:**

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

The **UNIQUE** constraint ensures that all values in a column are different.

# Launch Site Names Begin with 'CCA'

## SQL Query:

**INPUT:**

```
SELECT * FROM SPACEXDATASET WHERE launch_site LIKE 'CCA%' LIMIT 5
```

**OUTPUT:**

| 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 |

The **LIKE** operator is used in a **WHERE** clause to search for a specified pattern in a column.

The **LIMIT** clause select a limited number of records.

# Total Payload Mass

## SQL Query:

**INPUT:**

```
    %sql SELECT SUM(payload_mass_kg) AS total_payload_mass_kg FROM SPACEXDATASET
WHERE customer = 'NASA (CRS)'
```

**OUTPUT:**

| total_payload_mass_kg |
|---|
| 45596 |

The **SUM** function returns the total sum of a numeric column.

The **WHERE** clause is used to filter records. In this case it was used to filter the customer as “NASA (CRS)”.

# Average Payload Mass by F9 v1.1

## SQL Query:

**INPUT:**

```
    SELECT AVG(payload_mass_kg) AS avg_payload_mass_kg FROM SPACEXDATASET WHERE
booster_version = 'F9 v1.1’
```

**OUTPUT:**

| avg_payload_mass_kg |
| --- |
| 2928 |

The **AVG** function returns the average value of a numeric column.

The **WHERE** clause is used to filter records. In this case it was used to filter the booster version as “F9 v1.1”.

# First Successful Ground Landing Date

## SQL Query:

**INPUT:**

```
    SELECT MIN(date) AS date FROM SPACEXDATASET WHERE landing__outcome = 'Success
(ground pad)'
```

**OUTPUT:**

| date |
| --- |
| 2015-12-22 |

The **MIN** function returns the smallest value of the selected column. In this case it was used to filter the first date record.

The **WHERE** clause is used to filter records. In this case it was used to filter the landing outcome as "Success (ground pad)".

# Successful Drone Ship Landing with Payload between 4000 and 6000

## SQL Query:

**INPUT:**

```
    SELECT booster_version FROM SPACEXDATASET WHERE (landing__outcome = 'Success
(drone ship)') AND (payload_mass_kg BETWEEN 4000 AND 6000)
```

**OUTPUT:**

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

The **WHERE** clause is used to filter records. In this case it was used to filter the landing outcome as “Success (drone ship)”.

The **BETWEEN** operator selects values within a given range. In this case it was used to filter the payload mass between 4,000 and 6,000kg.

# Total Number of Successful and Failure Mission Outcomes

## SQL Query:

**INPUT:**

```
    SELECT COUNT(*) AS SUCCESS, (SELECT COUNT(*) AS FAILURE FROM SPACEXDATASET
WHERE mission_outcome LIKE 'Failure%') FROM SPACEXDATASET WHERE mission_outcome
LIKE 'Success%'
```

**OUTPUT:**

| success | failure |
|---|---|
| 100 | 1 |

The **COUNT** function returns the number of rows that matches a specified criterion. In this case it returns the number of rows with success and failure as value.

The **LIKE** operator is used in a **WHERE** clause to search for a specified pattern in a column. In this case when a record starts with Success and Failure.

# Boosters Carried Maximum Payload

## SQL Query:

**INPUT:**

```
    %sql SELECT booster_version FROM SPACEXDATASET WHERE payload_mass_kg = (SELECT
MAX(payload_mass_kg) FROM SPACEXDATASET) ORDER BY booster_version ASC
```

**OUTPUT:**

| *booster_version | | | |
|---|---|---|---|
| F9 B5 B1048.4 | F9 B5 B1049.5 | F9 B5 B1051.4 | F9 B5 B1058.3 |
| F9 B5 B1048.5 | F9 B5 B1049.7 | F9 B5 B1051.6 | F9 B5 B1060.2 |
| F9 B5 B1049.4 | F9 B5 B1051.3 | F9 B5 B1056.4 | F9 B5 B1060.3 |

*The table was modified to have 4 columns to fit the data on the slide. The output has only 1 column with all the records.

The **MAX** function returns the largest value of the selected column. In this case it was used to filter the maximum value of payload mass.

A **subquery** is a SQL query nested inside a larger query. In this case it was used to find the maximum value of payload mass to filter the booster version.

The **ORDER BY** keyword is used to sort the result-set in ascending or descending order. **ASC** specify the order as ascending.

# 2015 Failed Launch Records

## SQL Query:

**INPUT:**

```
    SELECT landing__outcome, booster_version, launch_site FROM SPACEXDATASET where
(landing__outcome = 'Failure (drone ship)') AND (YEAR(date) = 2015)
```

**OUTPUT:**

| landing__outcome | booster_version | launch_site |
|---|---|---|
| Failure (drone ship) | F9 v1.1 B1012 | CCAFS LC-40 |
| Failure (drone ship) | F9 v1.1 B1015 | CCAFS LC-40 |

The **WHERE** clause is used to filter records. In this case it was used to filter the landing outcome as "Failure (drone ship)".

The **YEAR** function returns the year part for a specified date. In this case the **YEAR** function was used to filter the year of the date records.

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

## SQL Query:

**INPUT:**

```
    SELECT landing__outcome, COUNT(landing__outcome) AS total FROM SPACEXDATASET
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY landing__outcome ORDER
BY TOTAL DESC
```

**OUTPUT:**

| landing__outcome | total |
|---|---|
| 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 |

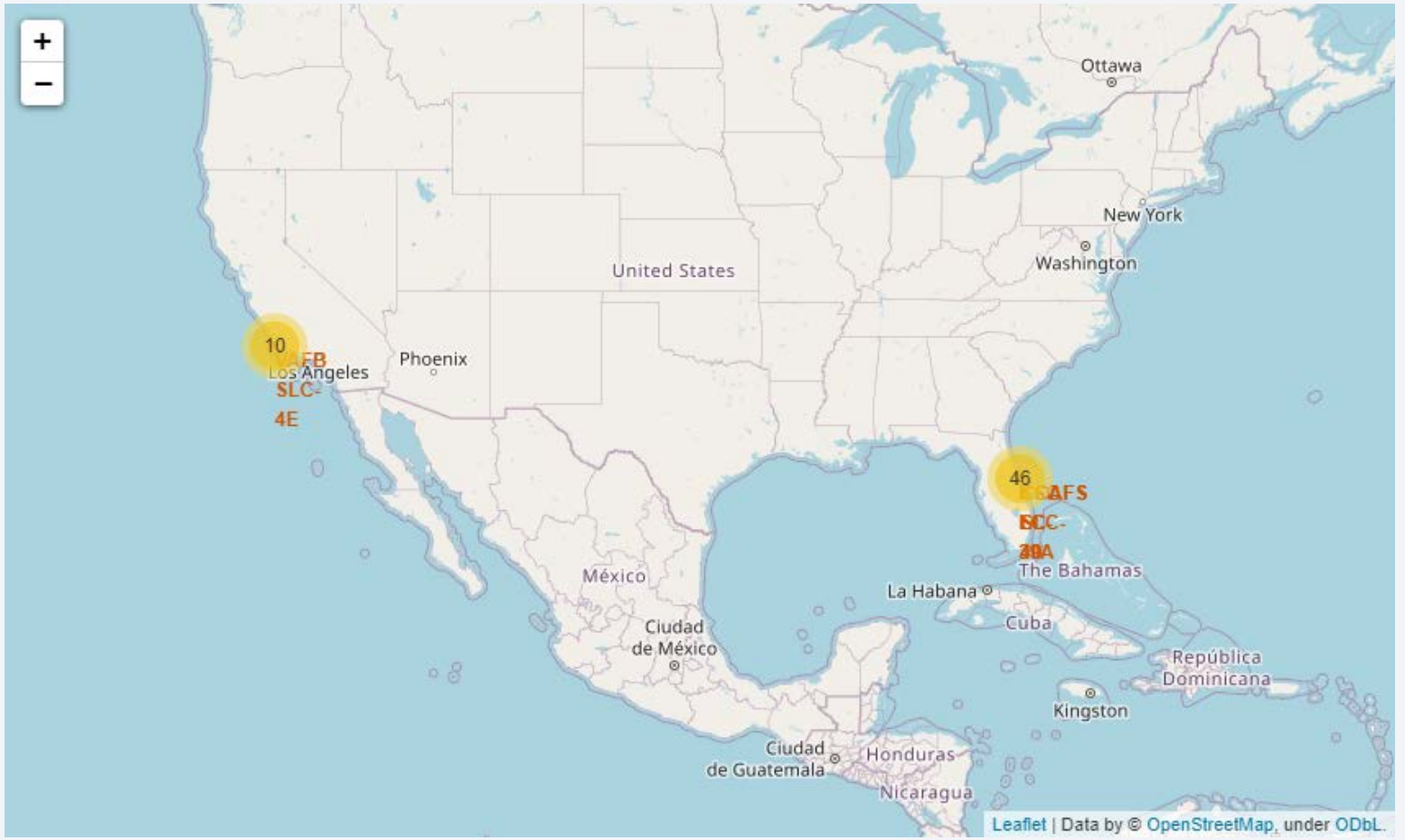
The **GROUP BY** statement groups rows that have the same values into summary rows. In this case it was used to group each landing outcome. The **GROUP BY** statement is often used with aggregate functions such as **COUNT**.

The **COUNT** function was used to return the number of rows of each landing outcome.

The **ORDER BY** keyword is used to sort the result-set in ascending or descending order. **DESC** specify the order as descending.


Section 4
Launch Sites
Proximities Analysis

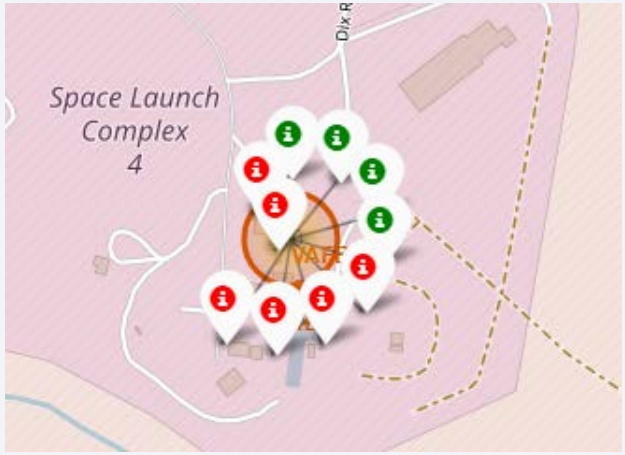
# Folium Interactive Map – All Launch Sites Locations




All SpaceX launch sites are located in the United States of America at the following locations:

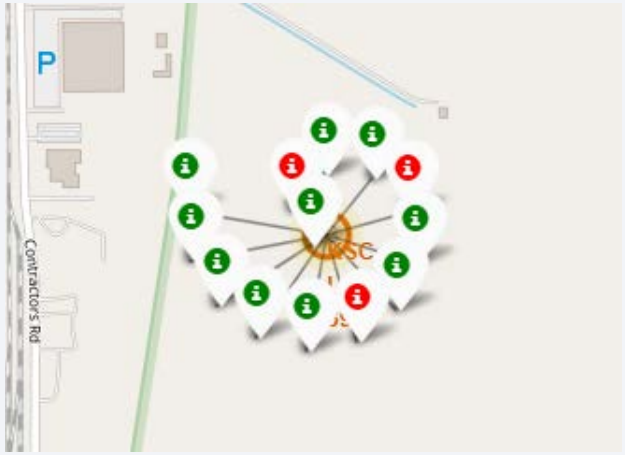
- Vandenberg Space Force Base, Santa Barbara, California (VAFB SLC-4E – 10 launches).
- Kennedy Space Center, Merritt Island, Florida (KSC LC-39A – 13 launches).
- Cape Canaveral Space Force Station, Cape Canaveral, Florida (CCAFS LC-40 – 26 launches, CCAFS SLC-40 – 7 launches).

# Folium Interactive Map – Success/failed launches by icon color

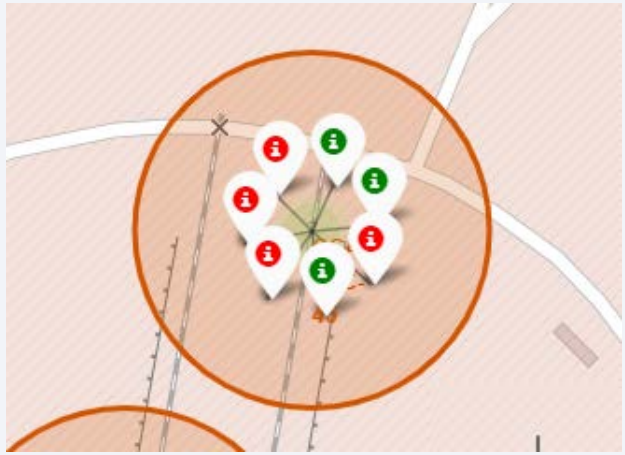



VAFB SLC-4E

CCAFS LC-40



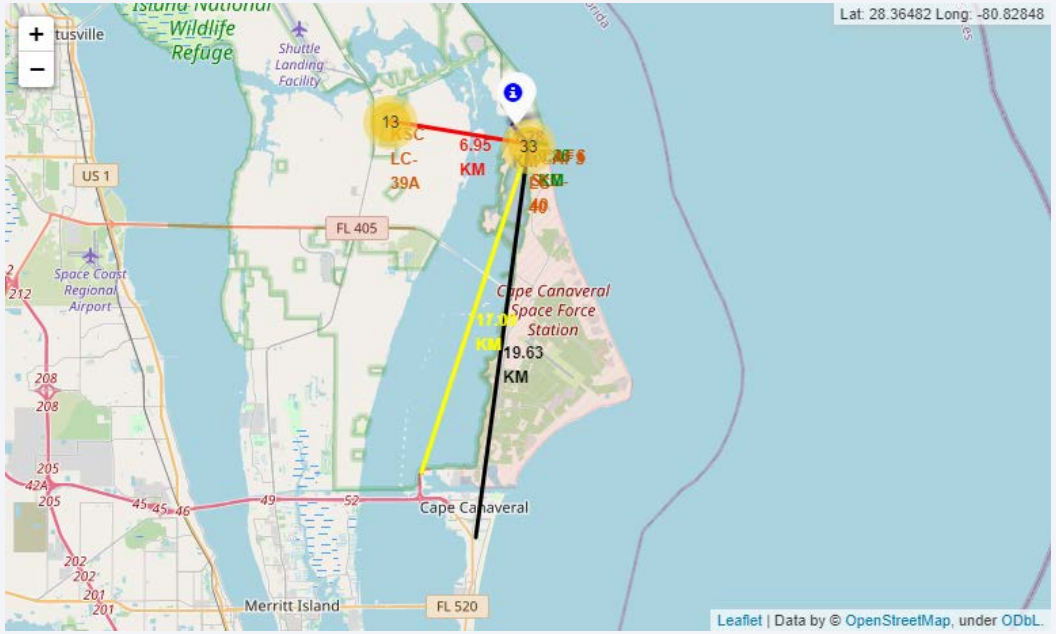

KSC LC-39A



CCAFS SLC-40

- Green icon color means that the launch was successful;
- Red icon color means that the launch was failed.

# Folium Interactive Map – Distances Between A Launch Site To Its Proximities

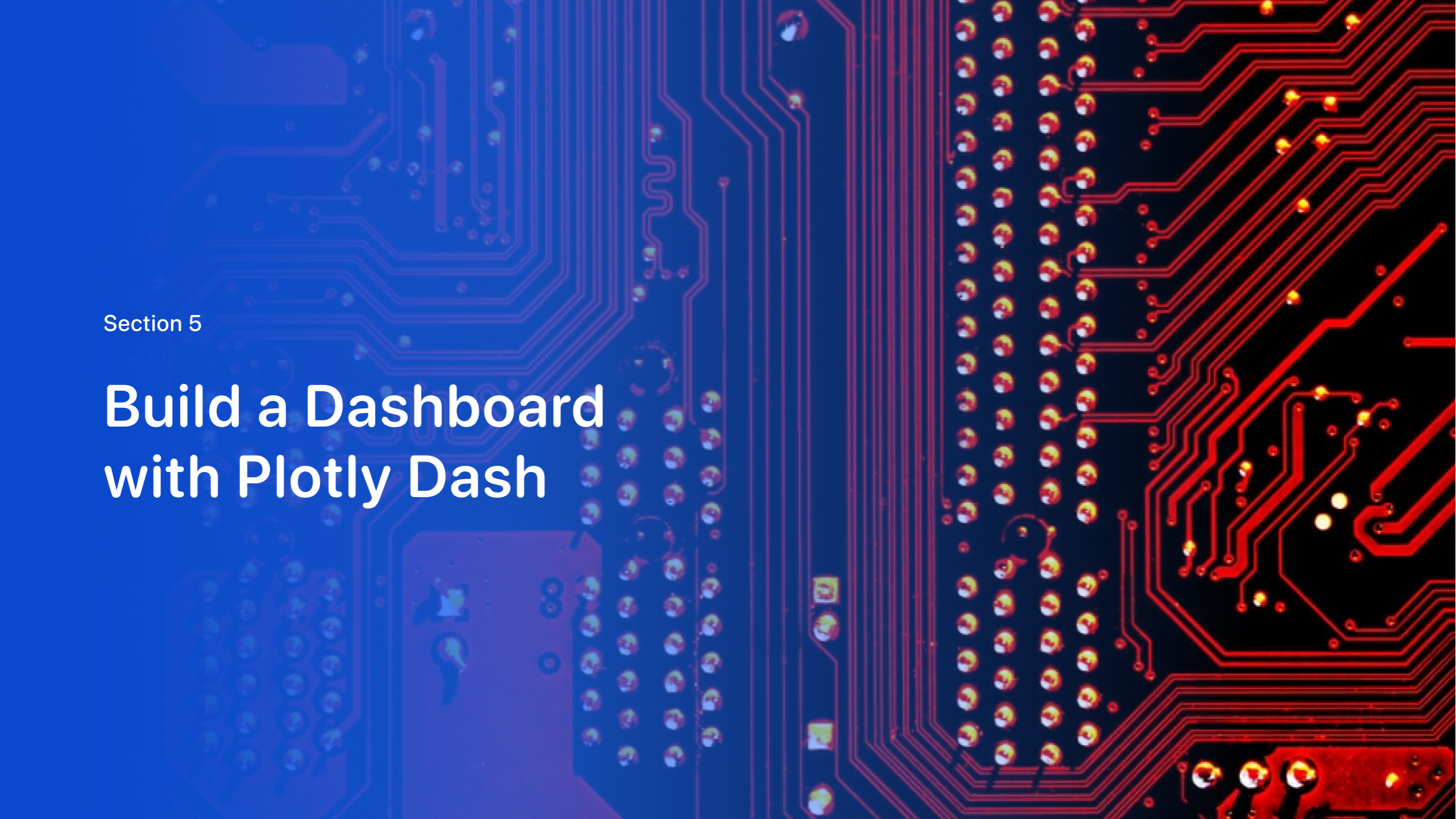



Calculated distances based on the CCADS SLC-40 launch site:

- CCADS SLC-40 and KSC LC-39A: ~ 7km;
- CCADS SLC-40 and Cape Canaveral City: ~20km;
- CCADS SLC-40 and the coastline: ~0.9km;
- CCADS SLC-40 and the closest railway (NASA Railroad): ~1.3km;
- CCADS SLC-40 and the closest highway (FL 401): ~17km.

Some conclusions:

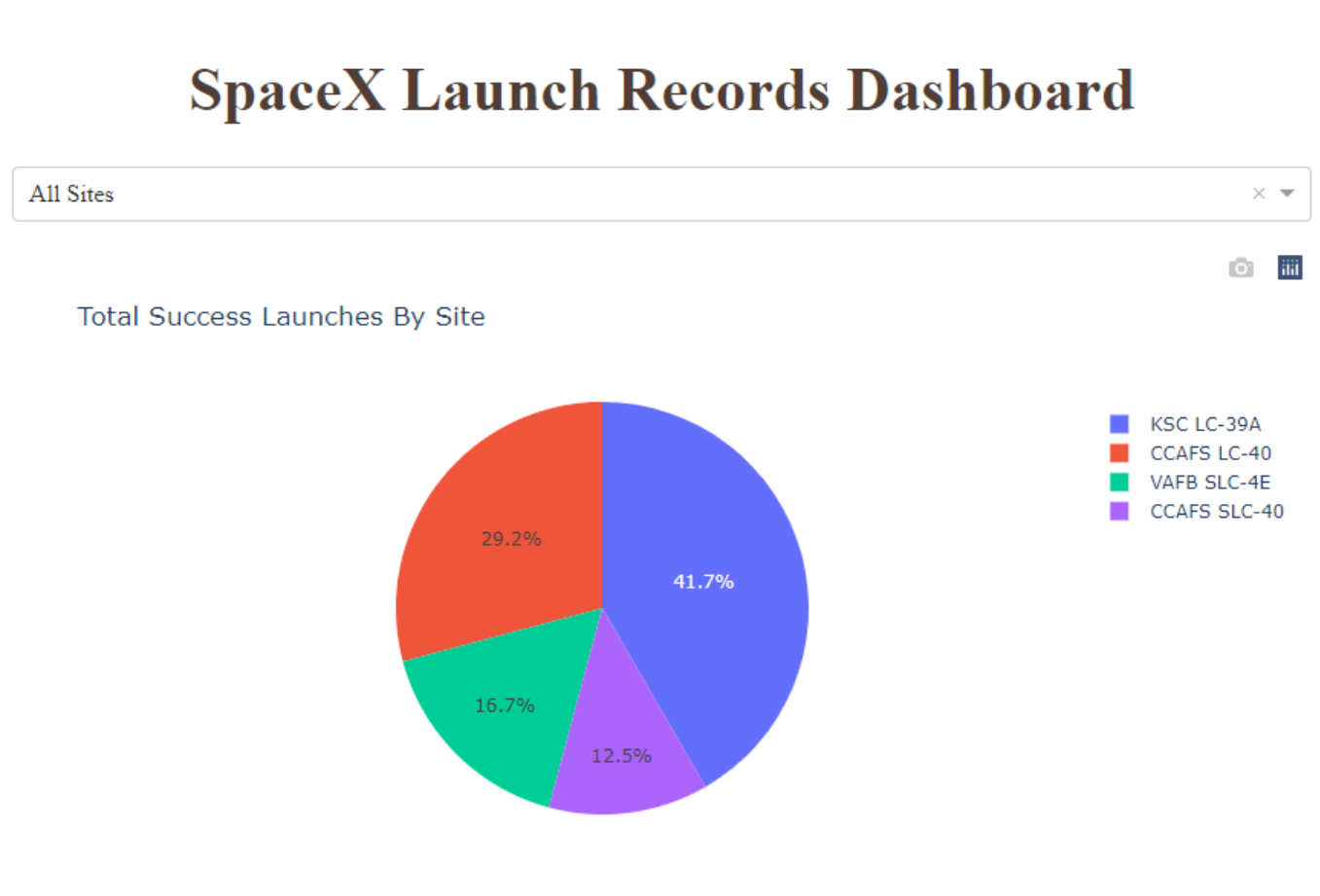
- Launch site close to Railways (Only NASA Railroad);
- Launch site close to Coastline;
- Launch site not close to Highways;
- Launch site keep certain distance away from cities.



Section 5

# Build a Dashboard with Plotly Dash

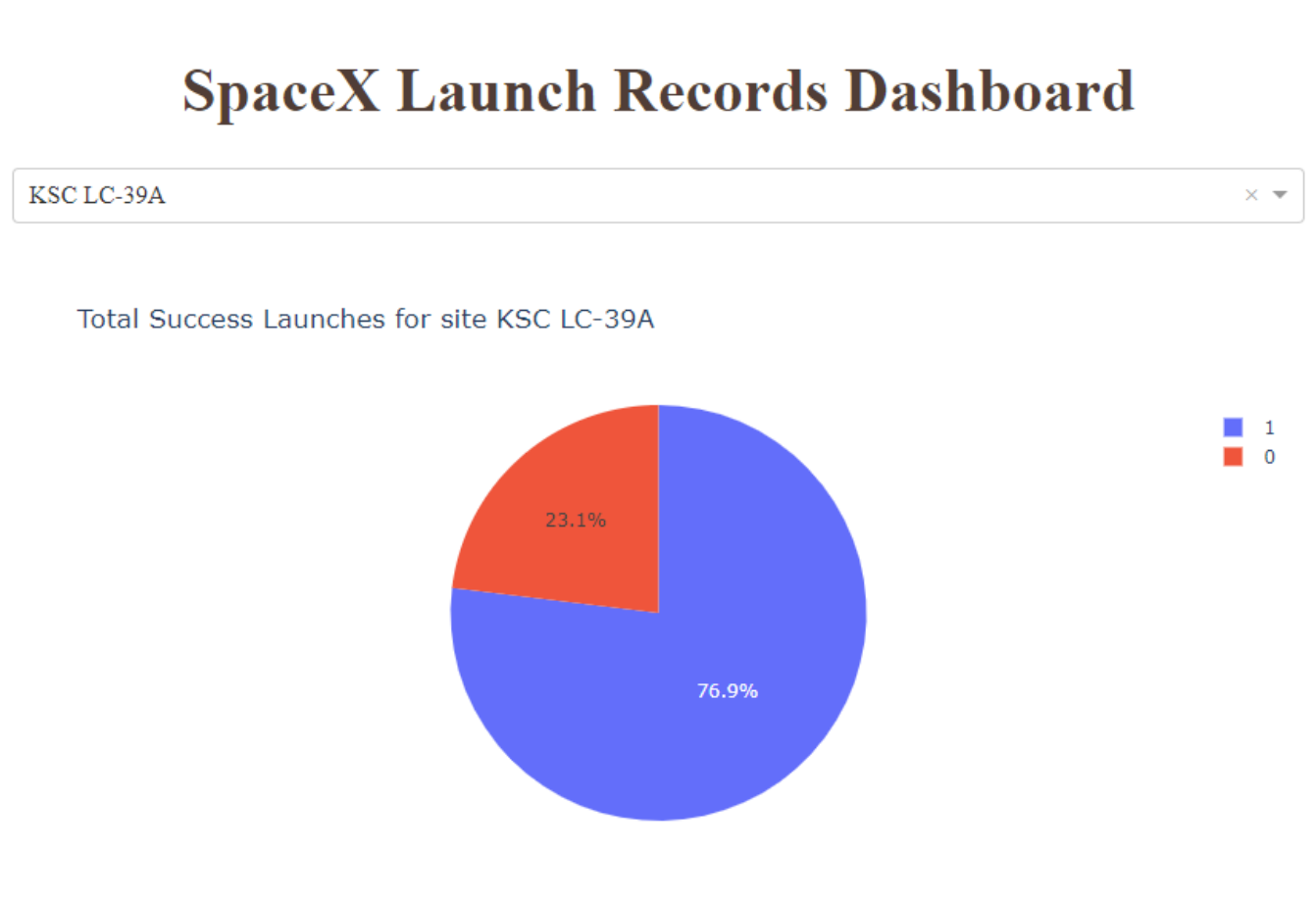
# Ploty-Dash Dashboard – Total Success Launches by Site




The screenshot of the dashboard shows the launch success ratio of all launch sites.

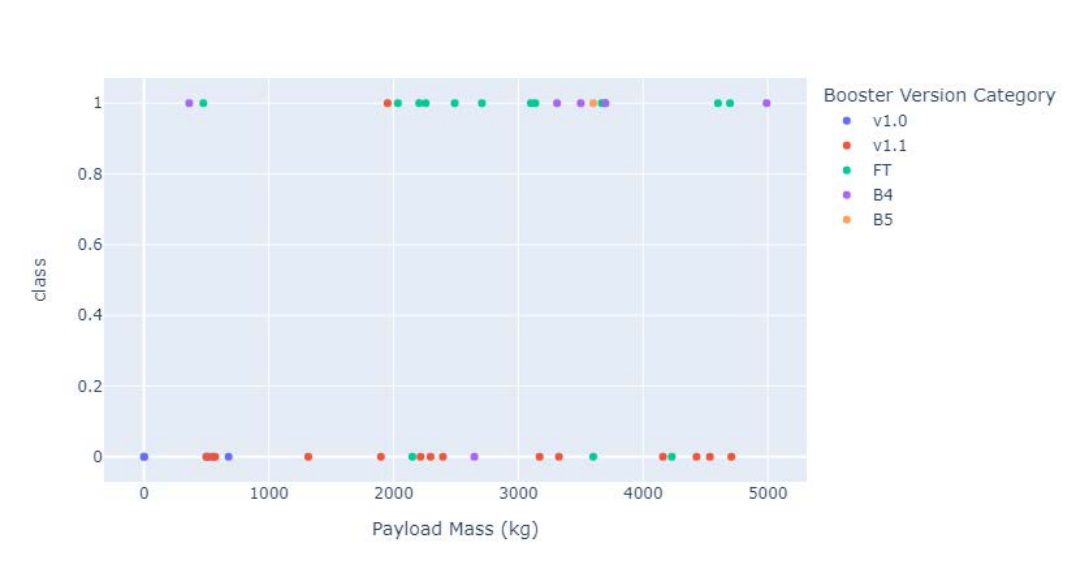
The KSC LC-39A has the highest launch success ratio.

# Ploty-Dash Dashboard – Site With The Highest Launch Success Ratio

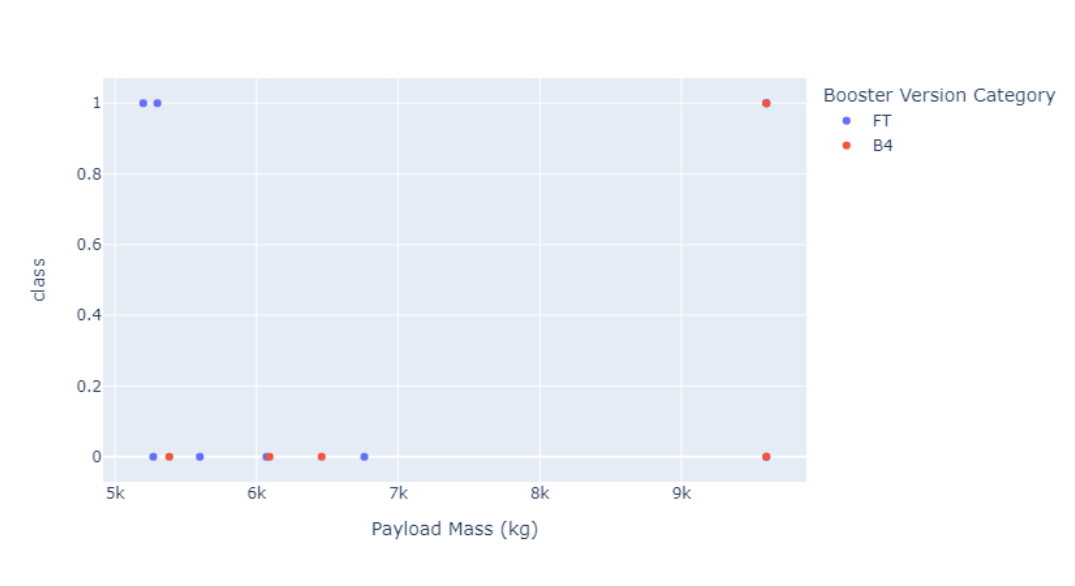



KSC LC-39A has the highest launch success ratio of 76.9% and 23.1% of failure.

# Ploty-Dash Dashboard – Payload vs. Launch Outcome




Payload mass range of 0 to 5,000kg




Payload mass range of 5,000kg to 10,000kg

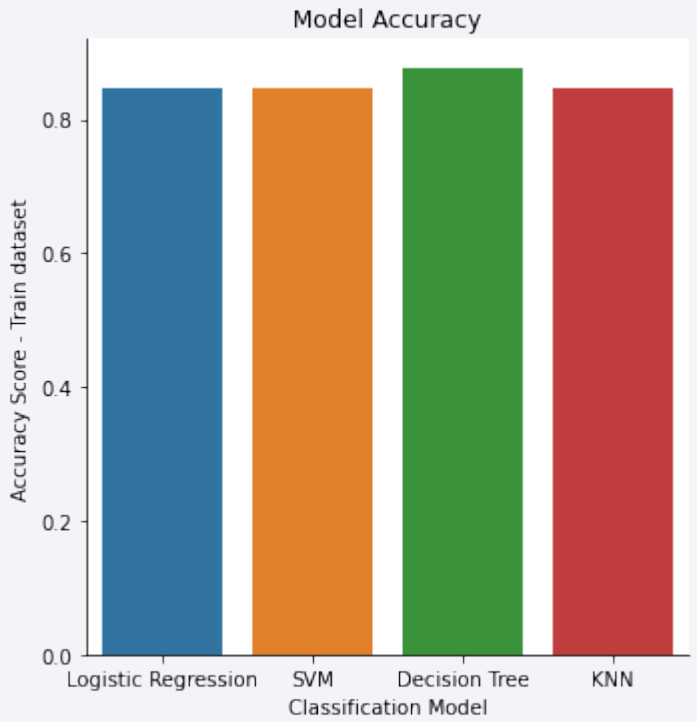
The observed behavior was that we can have a positive correlation between the payload mass and success rate but as the payload increases the correlation goes from positive to negative. So lower payload mass seems to have a higher success rate than heavy payloads.

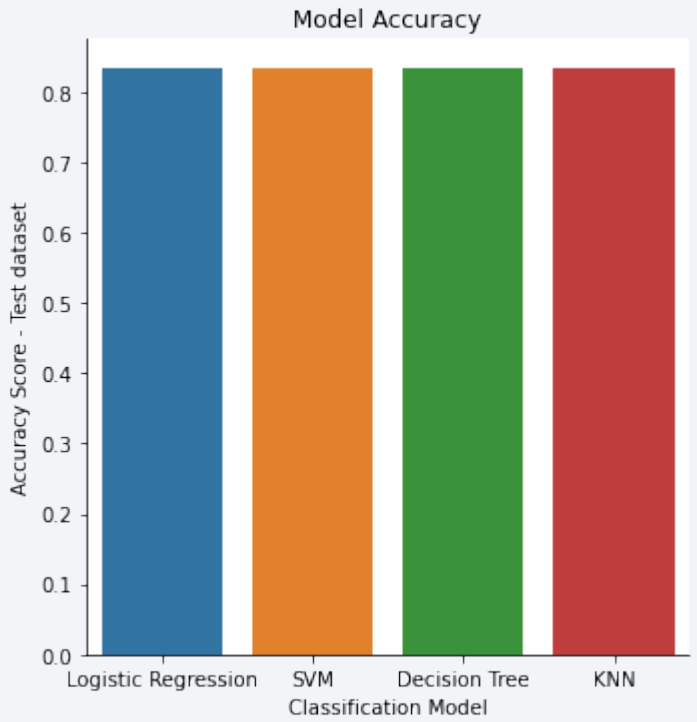
Section 6

# Predictive Analysis (Classification)

# Classification Accuracy

| Model | Training Accuracy | Test Accuracy |
|---|---|---|
| Logistic Regression | 0.846429 | 0.833333 |
| SVM | 0.848214 | 0.833333 |
| Decision Tree | 0.876786 | 0.833333 |
| KNN | 0.848214 | 0.833333 |



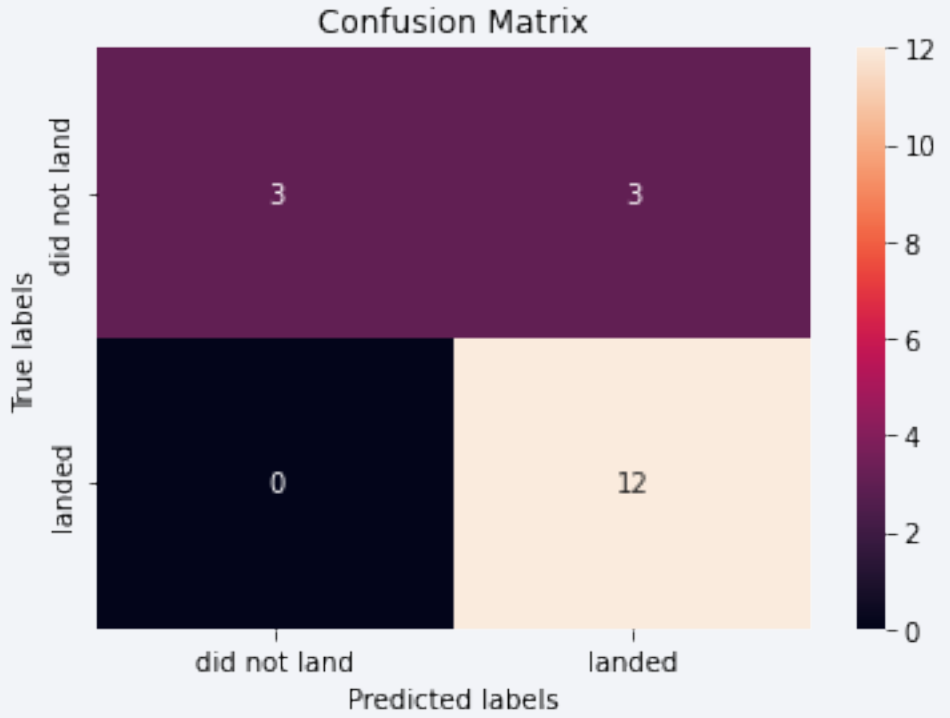

All the models got the same accuracy using the test dataset. So the model was chosen by the accuracy score of the training dataset. With the training dataset the highest score was of the Decision Tree Model with 0.876786 of accuracy.

# Confusion Matrix

A confusion matrix is a summary of prediction results on a classification problem.

The number of correct and incorrect predictions are summarized with count values and broken down by each class.

Examining the confusion matrix, we can see that all the models can distinguish between the different classes but the major problem is false positives.



# Conclusions

### Chosen Classification Model

- All models got the same accuracy with the test dataset, however, the decision tree algorithm got the best accuracy with the training dataset. So the chosen model was the Decision Tree.

### Insights of the Exploratory Data Analysis (EDA)

- Successful launches have a positive correlation with the amount of launches based on the chart of Flight Number vs. Launch Site and the Launch Success Yearly Trend;
- Payload mass have a positive correlation with the success rate but as the payload increases the correlation goes from positive to negative. So lower payload mass seems to have a higher success rate than heavy payloads;
- The majority of failed launches happened with a payload mass of around 6,000 kg for both the KSC LC 39A and CCADS SLC 40 launch sites;
- The Average Payload Mass by F9 v1.1 is 2,928kg;
- ES-L1, GEO, HEO and SSO are the orbit types with the highest success rate.

### Insights of Folium Interactive Map

- CCAFS LC-40 was the place where most launches took place (26 launches);
- The evaluated launch site is close to Railways (Only NASA Railroad);
- The evaluated launch site is close to Coastline;
- The evaluated launch site is not close to Highways (FL 401);
- The evaluated launch site is keep certain distance away from cities.

### Insights of Ploty-Dash Dashboard

- KSC LC-39A had the highest launch success ratio of 76.9% among all launch sites used.

# Appendix

- Database Datefix
  - Datefix.ipynb (Jupyter Notebook on GitHub)
- Haversine Formula
  - Haversine.py (Python file on GitHub)
- Heroku: Cloud Application Platform
  - https://dash.plotly.com/deployment
  - Source code (GitHub folder)

# References

Chartio (2019). A Complete Guide to Scatter Plots. Retrieved from https://chartio.com/learn/charts/what-is-a-scatter-plot/

Chartio (2019). A Complete Guide to Bar Charts. Retrieved from https://chartio.com/learn/charts/line-chart-complete-guide/

Chartio (2019). A Complete Guide to Line Charts. Retrieved from https://chartio.com/learn/charts/bar-chart-complete-guide/

# Thank you!