IBM Developer
SKILLS NETWORK

# Winning Space Race with Data Science

Allan de Andrade
11/10/2021

# Outline

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

# Executive Summary

- Summary of methodologies
    - Data Collection
    - Data wragling
    - EDA with data visualization
    - Building an interactive map with Folium
    - Building a Dashbord with Plotly Dash
    - Predictive Analysis (Classification)
- Summary of all results
    - Exploratory data analysis results
    - Interactive analytics demo in screenshots
    - Predictive analysis results

# Introduction

- Project background and context

Along this project we persecuted predict if the Falcon 9 first stage will 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

- What factors and influences are determinants to predict whether SpaceX will attempt to land a rocket or not;

- The effect of 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 success landing rate.

Section 1

# Methodology

# Methodology

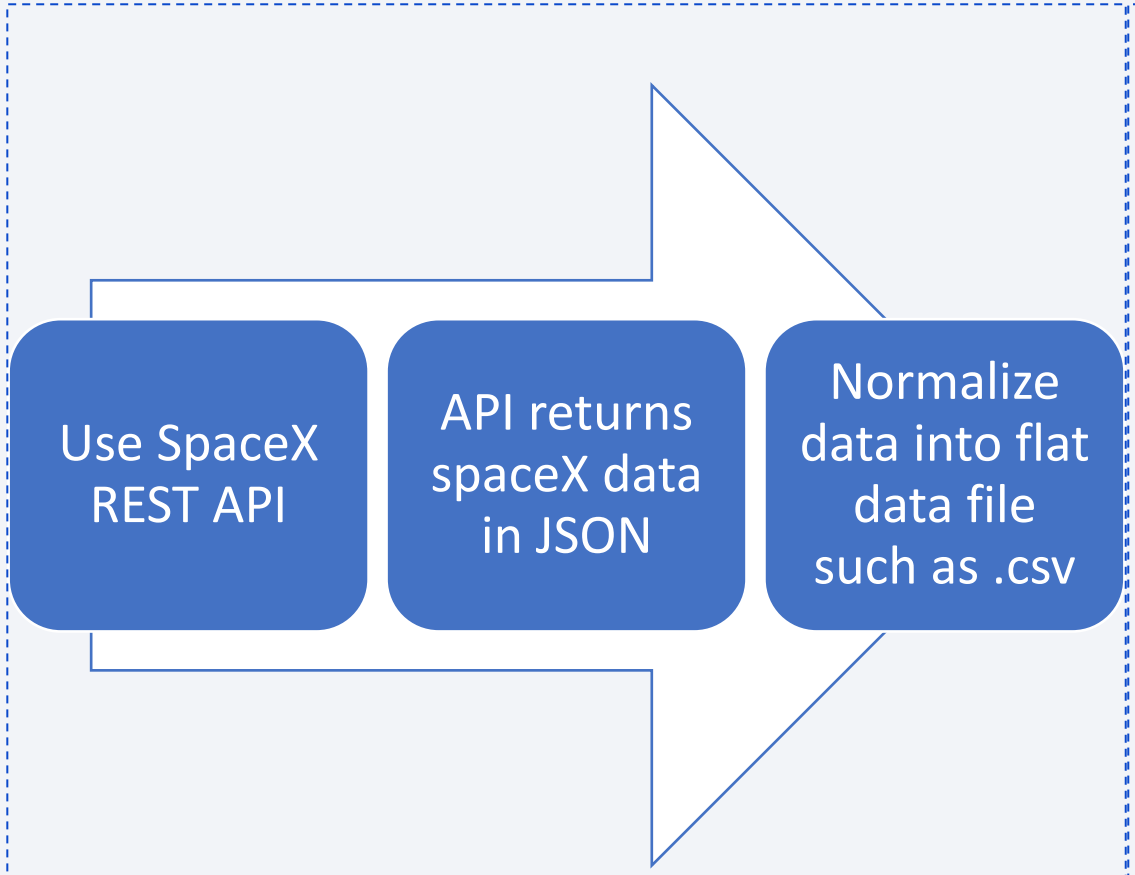
## Executive Summary

- Data collection methodology:
  - Data was collected using the SpaceX Rest API and by doing a Web Scrapping from Wikipedia
- Perform data wrangling
  - One hot Enconding data fields for Machine Learning and dropping irrelevant columns
- Perform exploratory data analysis (EDA) using visualization and SQL
  - Plotting: Scatter Graphs, Bar Graphs to show relationships between variables to show patterns of data
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

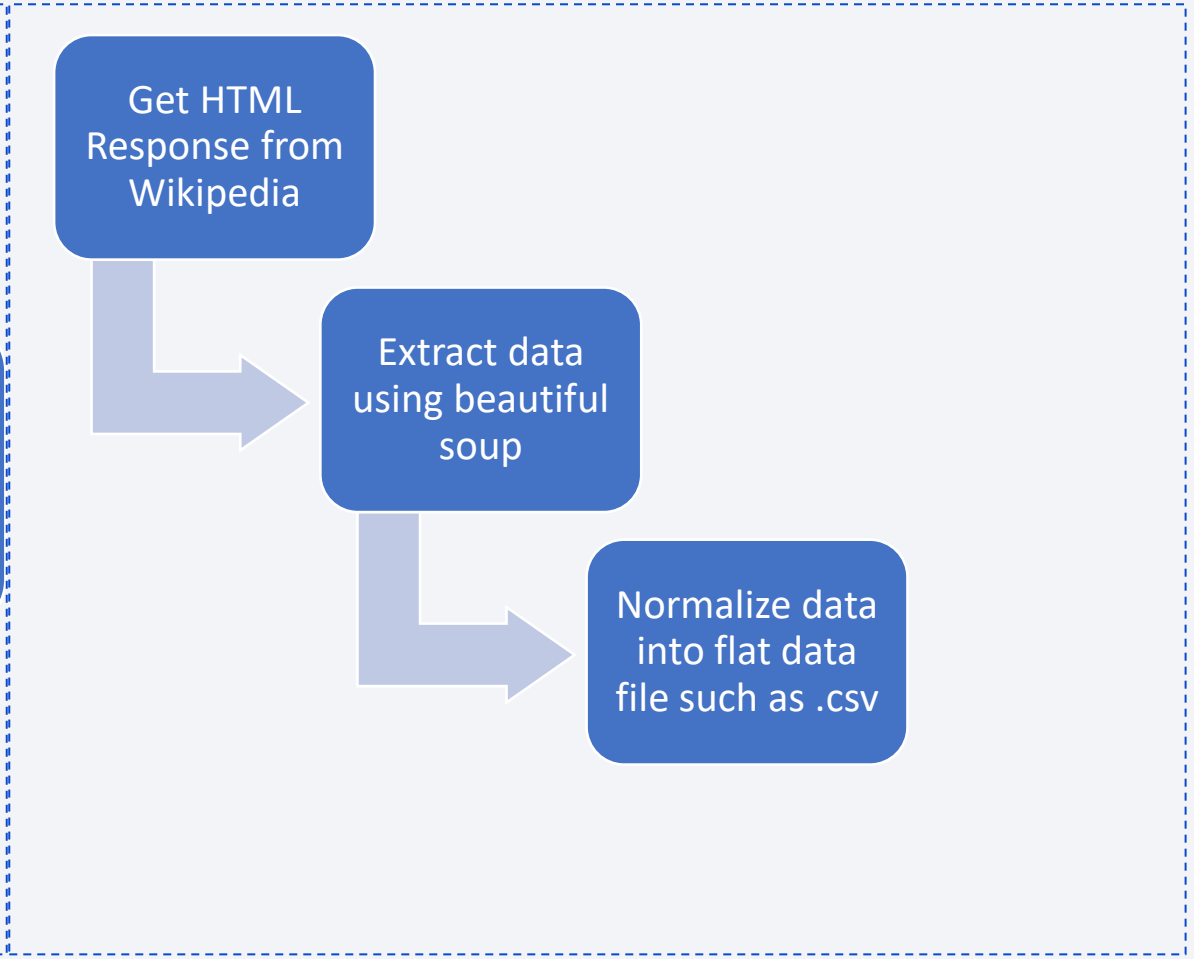
# Data Collection

- The datasets were 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 and landing specifications, and outcomes.
- Our goal is to use this data collection 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 data source for obtaining Falcon 9 Launch data is web scraping Wikipedia using BeautifulSoup.

# SpaceX API




# Web Scraping



# Data Collection – SpaceX API

GitHub link

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

```
# Use json_normalize method to convert the json result into a dataframe
response = requests.get(static_json_url).json()
data = pd.json_normalize(response)
```

3. Apply custom functions to clean data

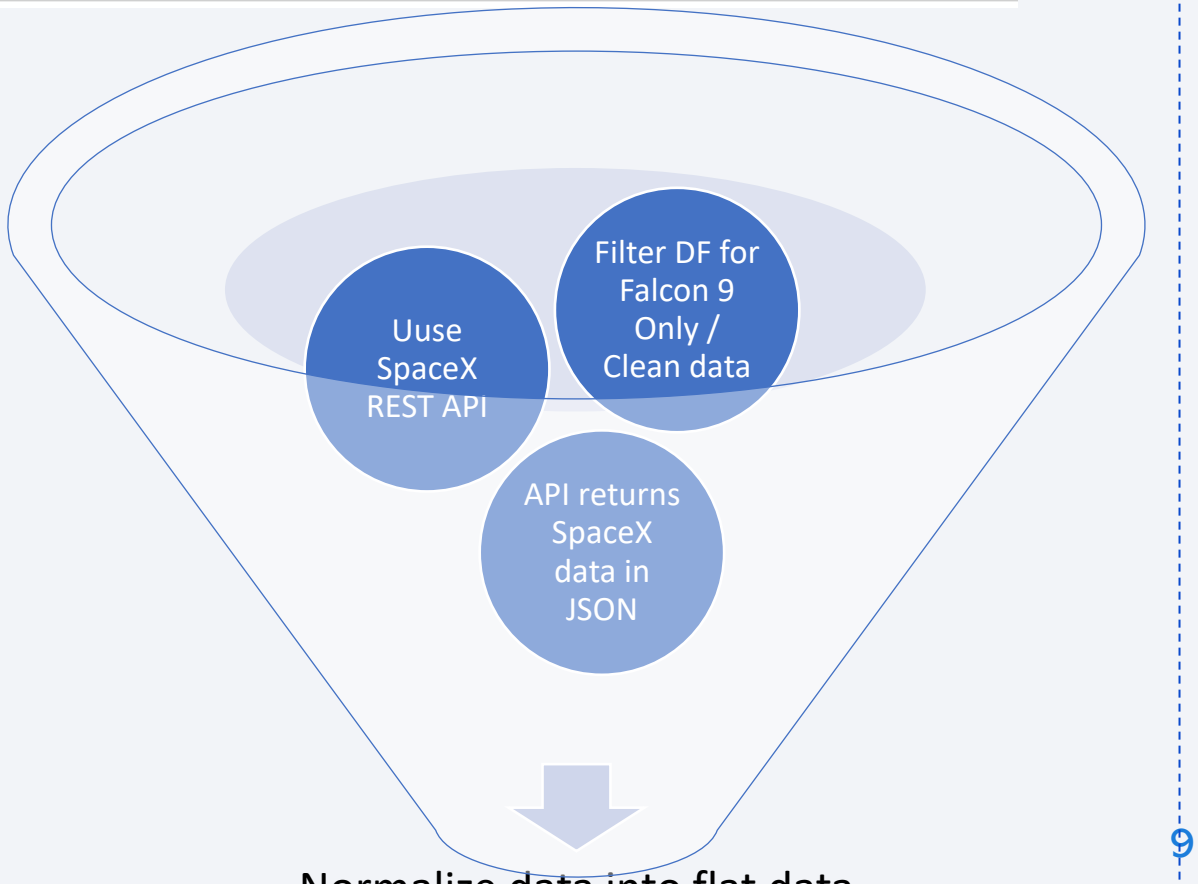
```
getBoosterVersion(data)
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
```

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}
```

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)
```




Normalize data into flat data file such as .csv

# Data Collection - Scraping

1. Getting Response from HTML:

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

2. Creating BeautifulSoup Object:

```
soup = BeautifulSoup(html, 'html5')
```

3. Finding tables:

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

4. Getting column names:

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

5. Creating a dictionary:

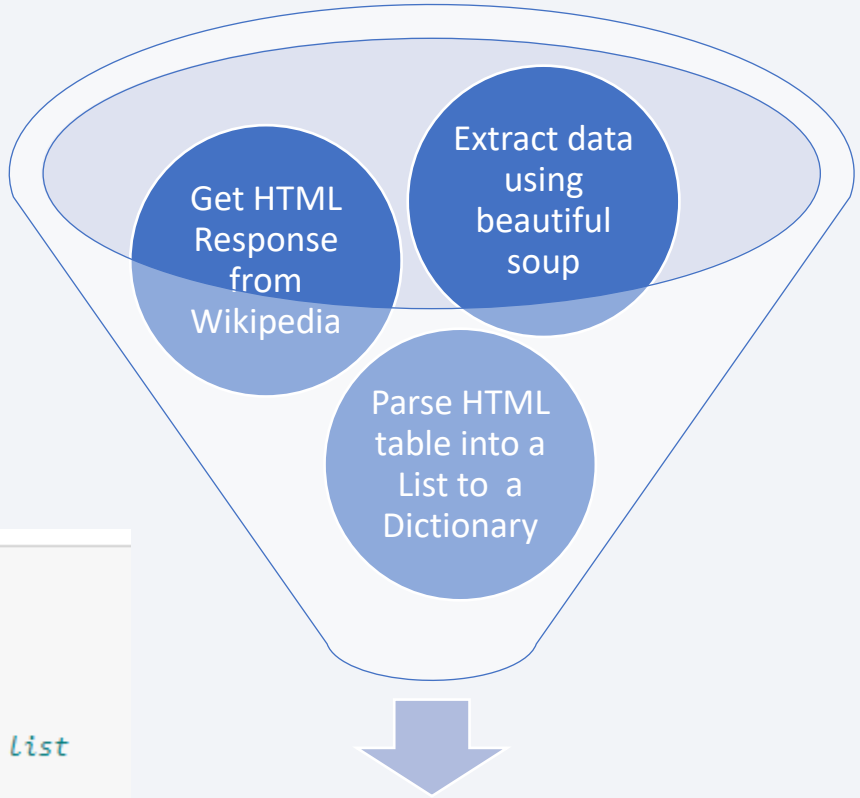
```
launch_dict= dict.fromkeys(column_names)

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

# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
```

6. Appending data to keys:

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('tab
```




Normalize data into flat data file such as .csv

7. Converting dictionary to dataframe

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

8. Dataframe to .CSV

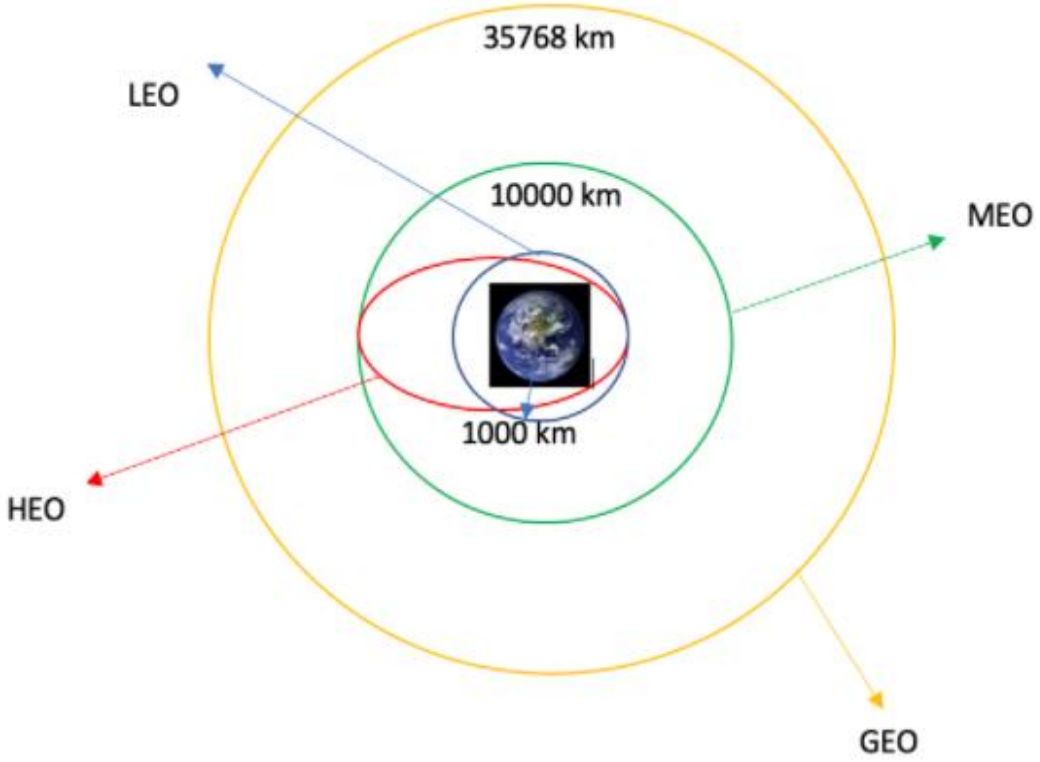
```
df.to_csv('space_web_scraped.csv', index = False)
```

# Data Wrangling

GitHub Link

- 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.

Each launch aims to an dedicated orbit, and here are a Diagram showing common orbit types SpaceX uses:




Perform Exploratory Data Analysis EDA on dataset

- Calculate the number of launches at each site
- Calculate the number and occurrence ofeach orbit

- Calculate the number and occurence of mission outcome per orbit type
- Export dataset as .CSV
- Create a landing outcome label from Outcome column
- Work out success rate for Every landing in dataset

# EDA with Data Visualization

GitHub Link

**Scatter Graphs being drawn:**

- Flight Number x Payload Mass
- Flight Number x Launch Site
- Payload x Launch Site
- Orbit x Flight Number
- Payload x Orbit Type
- Orbit x Payload Mass

**Bar Graph being drawn:**

- Mean x Orbit

**Line Graph being drawn:**

- Success Rate x Year

# EDA with SQL

GitHub Link

Performed SQL queries to gather information about the 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

GitHub Link

**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()

**Trends in which the Launch Site is situated in:**

```
Are launch sites in close proximity to highways? No
Are launch sites in close proximity to coastline? Yes
```

# Build a Dashboard with Plotly Dash

GitHub Link

**- The Dashboard is built with Plotly Dash:**

**Graphs:**

- Pie Chart showing the total launches by a certain site/all sites
**-** *display relative proportions of multiple classes of data.*

**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.

# Predictive Analysis (Classification)

GitHub Link

**BUILDING MODEL**
- **L**oad our dataset into NumPy and Pandas
- Transform Data
- Split our data into training and test data sets
- Check how many test samples we have
- Decide which type of machine learning algorithms we want to use
- Set our parameters and algorithms to GridSearchCV
- Fit our datasets into the GridSearchCV objects and train our dataset.

**EVALUATING MODEL**
- Check accuracy for each model
- Get tuned hyperparameters for each type of algorithms
- Plot Confusion Matrix

**IMPROVING MODEL**
- Feature Engineering
- Algorithm Tuning

**FINDING THE BEST PERFORMING CLASSIFICATION MODEL**
- The model with the best accuracy score wins the best performing model
- In the notebook there is a dictionary of algorithms with scores at the bottom of the notebook

# Results

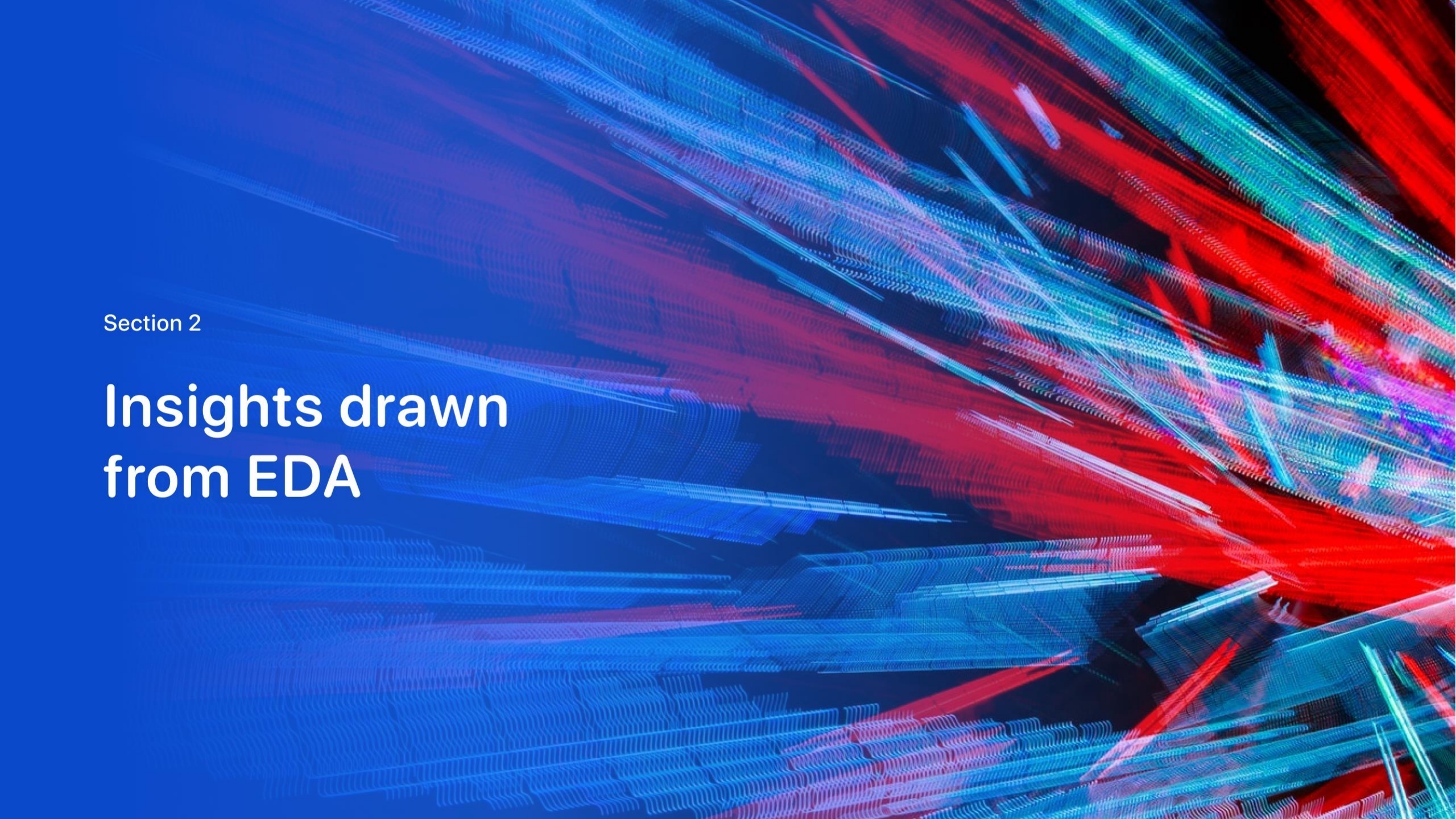
Exploratory data analysis results

Interactive analytics demo in screenshots

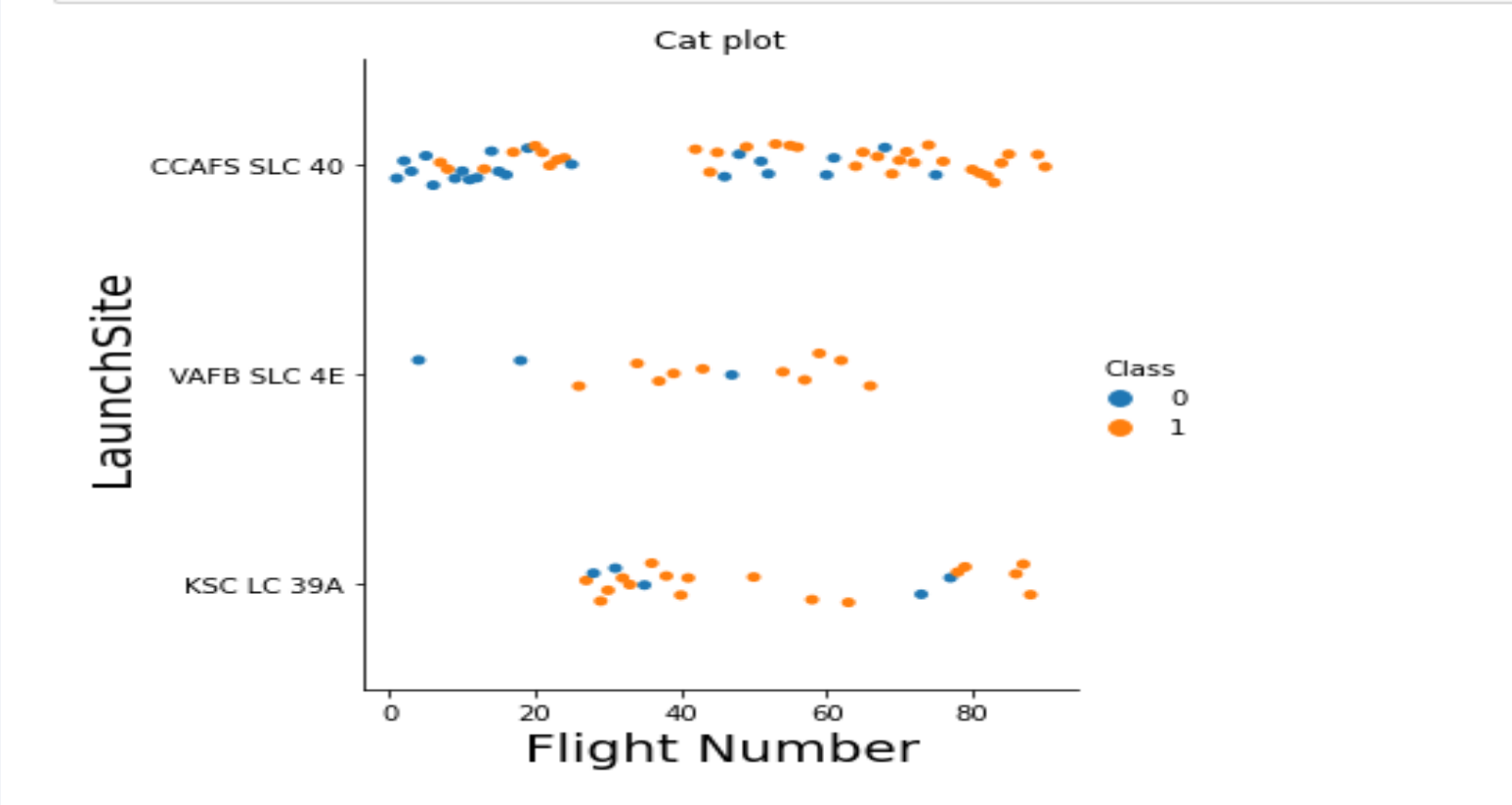
Predictive analysis results

Section 2

# Insights drawn from EDA

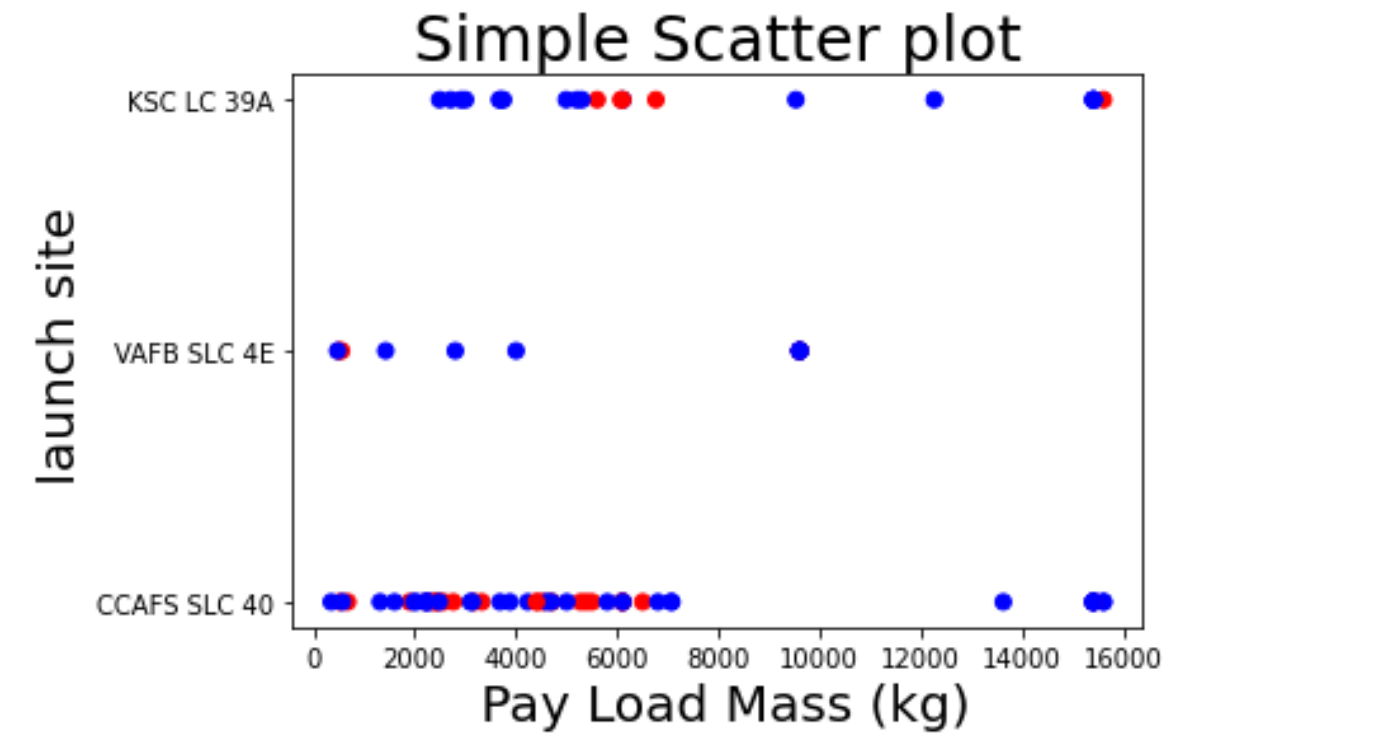


# Flight Number vs. Launch Site




The more amount of flights at a launch site the greater the success rate at a launch site.

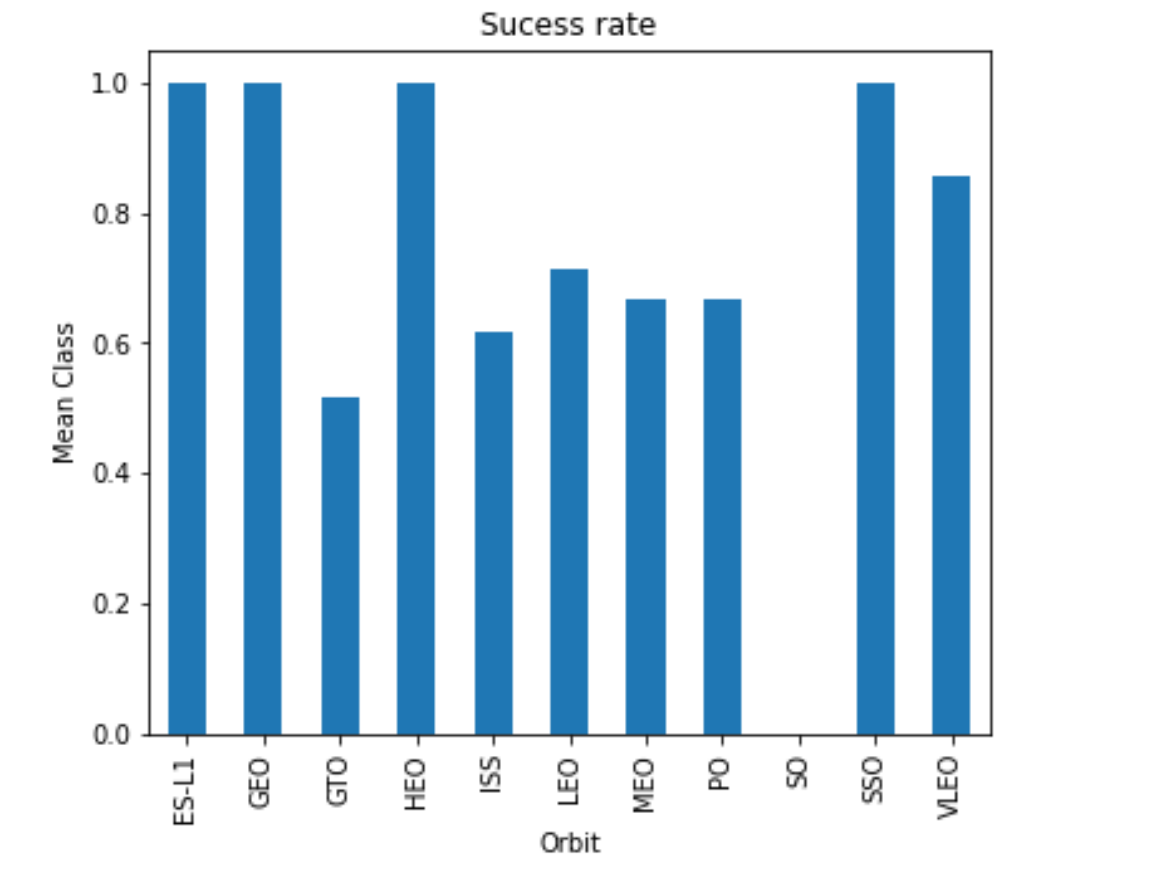
# Payload vs. Launch Site




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 dependant on Pay Load Mass for a success launch.

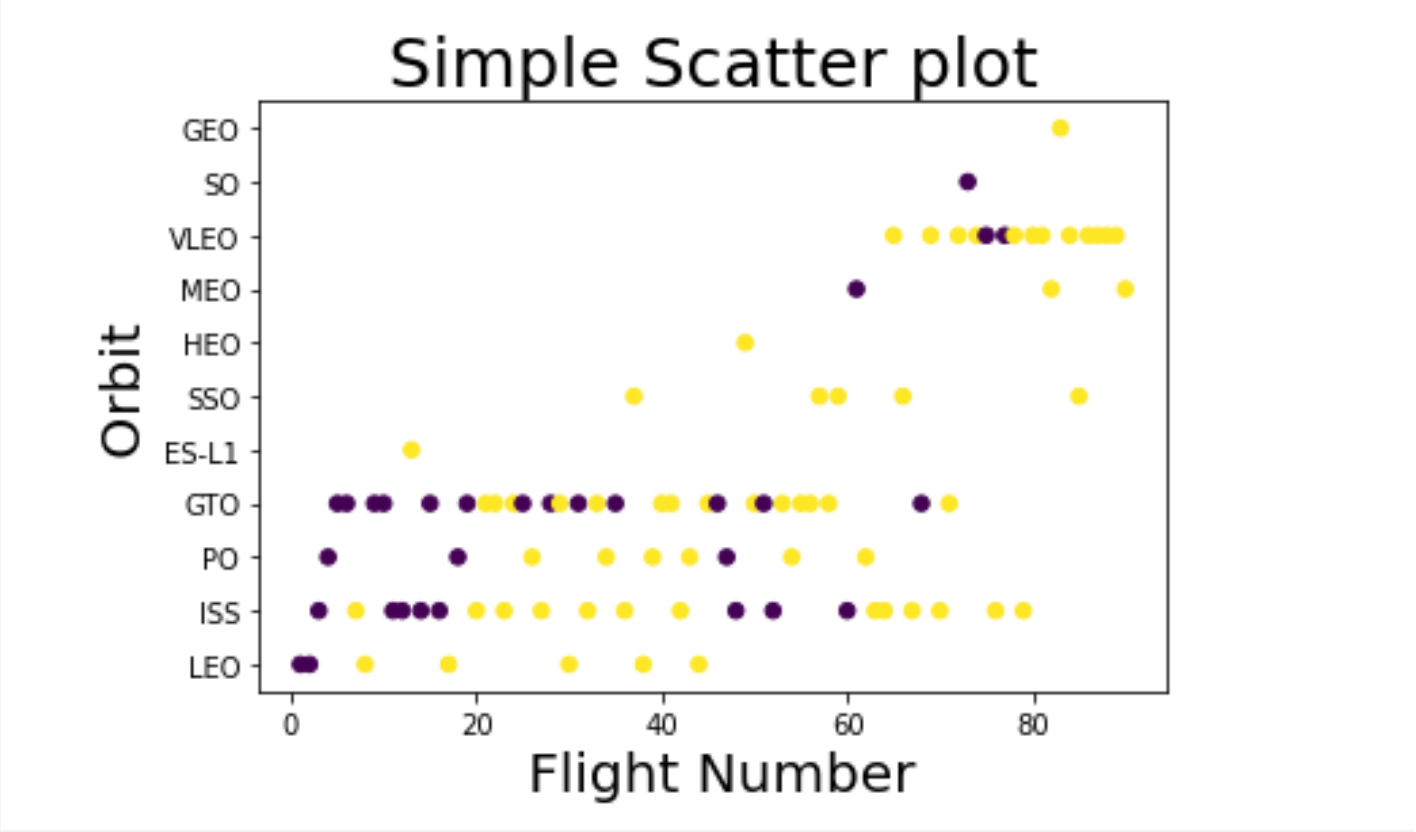
# Success Rate vs. Orbit Type

Orbit Geo, HEO, Sso, ES-L1 has the best Success Rate



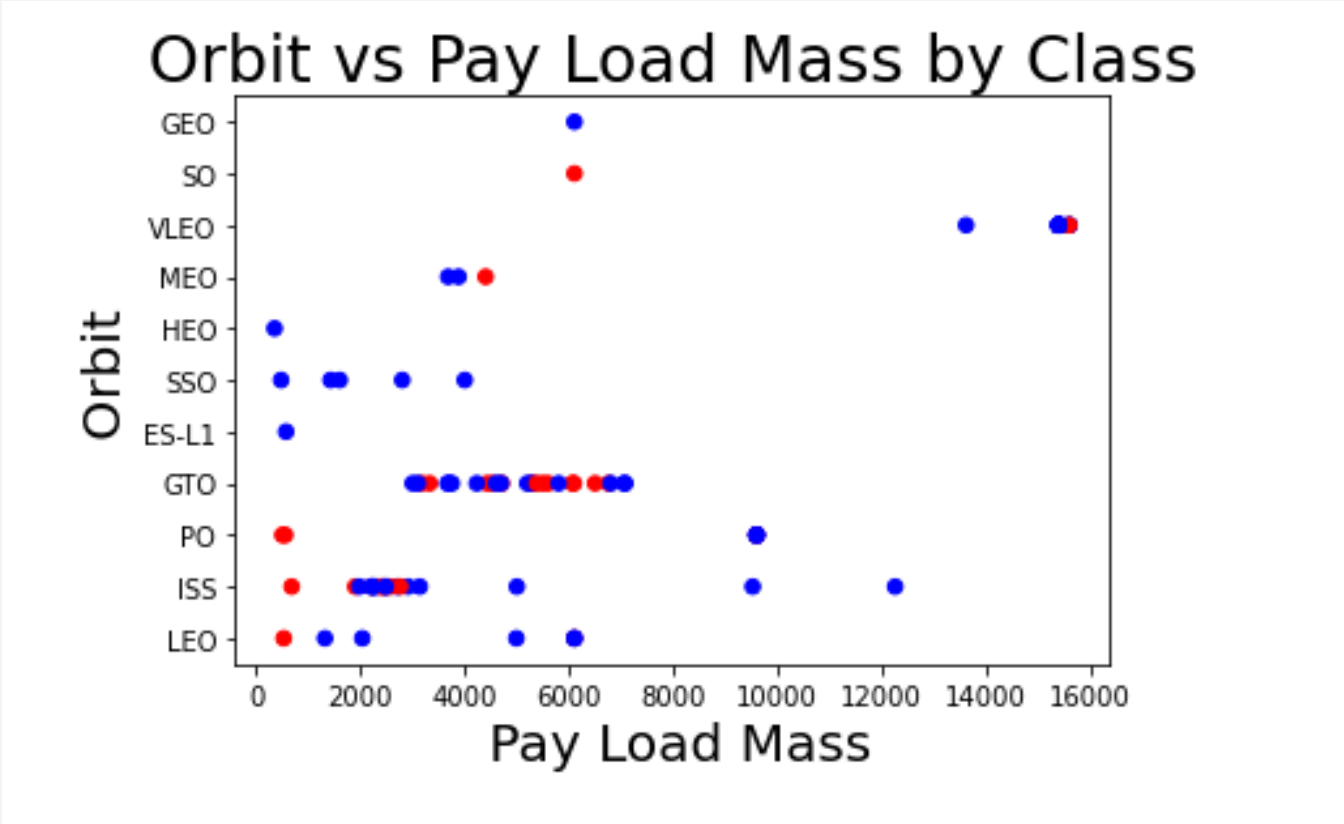
# Flight Number vs. Orbit Type

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.



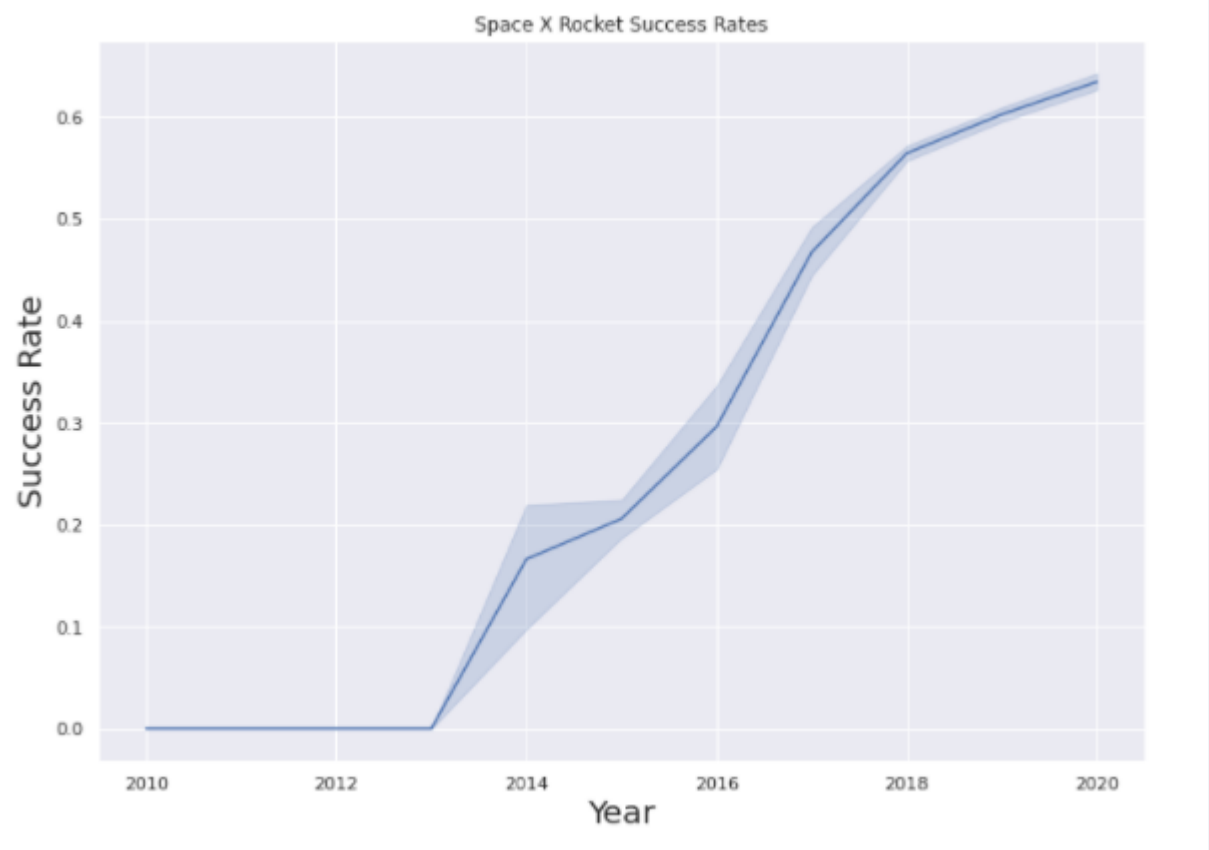
# Payload vs. Orbit Type

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



# Launch Success Yearly Trend

Observe that the success rate since 2013 kept increasing till 2020.



# All Launch Site Names

```
SELECT DISTINCT launch_site FROM spacextable
```

This Query brings a list with all the unique Launch Site Names used by SpaceX. To do so, we select from the table named *spacextable* the unique values at the column named *launch_site.*

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

# Launch Site Names Begin with 'CCA'

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

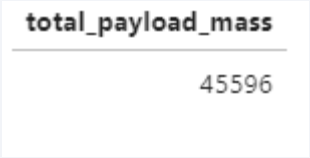
Using the query Limit 5 we can retrieve the first 5 instances from the table and with like and the percentage in the end suggests that the Launch_Site name must start with ‘CCA’

| DATE | time__utc_ | booster_version | launch_site | payload | payload_mass__kg_ | orbit | customer | mission_outcome | landingout |
|---|---|---|---|---|---|---|---|---|---|
| 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

```
SELECT sum(payload_mass__kg_) AS Total_Payload_Mass FROM spacextable \
WHERE customer = 'NASA (CRS)';
```

This Query calculate the total payload carried by boosters from NASA. To do so, we use the function SUM that returns the total in the selected column. At the same time, we use the WHERE clause to filter the dataset to only perform calculations on customer is equal to 'NASA (CRS)'.



| total_payload_mass |
| --- |
| 45596 |

# Average Payload Mass by F9 v1.1

```
SELECT avg(payload_mass__kg_) AS average_payload_mass_F9_v1 FROM
spacextable WHERE booster_version = 'F9 v1.1'
```

This QUERY calculate the average payload mass carried by booster version F9 v1.1. To do so, we calculate the average of the items where the column booster_version matchs ‘'F9 v1.1’ from the table and then rename.

| average_payload_mass_f9_v1 |
| --- |
| 2928 |

# First Successful Ground Landing Date

```
SELECT min(DATE) FROM spacextable WHERE LandingOut = 'Success (ground pad)'
```

This Query find the dates of the first successful landing outcome on ground pad. To do so, we use the MIN function that returns the minimum date from the column Date. . At the same time, we use the WHERE clause to filter the dataset to only perform calculations on LandingOut is equal to 'Success (ground pad)'.

| 1 |
| --- |
| 2015-12-22 |

# Successful Drone Ship Landing with Payload between 4000 and 6000

```
SELECT Booster_version FROM spacextable WHERE LandingOut = 'Success
(drone ship)' and payload_mass__kg_ > 4000 and payload_mass__kg_ < 6000
```

This Query list the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000. To do so, first we select only the Booster_Version. Then, , we use the WHERE clause to filter the dataset to only perform calculations on LandingOut is equal to 'Success (drone ship)' . After all, we use the AND clause to specify additional filter conditions.

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

# Total Number of Successful and Failure Mission Outcomes

```
SELECT MISSION_OUTCOME, COUNT(*) AS Number_of_sucess_or_failure FROM
spacextable GROUP BY MISSION_OUTCOME
```

This Query calculate the total number of successful and failure mission outcomes. To do so, we redesign our data group by the Mission Outcome.

| mission_outcome | number_of_sucess_or_failure |
|---|---|
| Failure (in flight) | 1 |
| Success | 99 |
| Success (payload status unclear) | 1 |

# Boosters Carried Maximum Payload

```
SELECT DISTINCT Booster_Version, MAX(PAYLOAD_MASS_KG_) AS [Maximum
Payload Mass] FROM spacextable GROUP BY Booster_Version ORDER BY
[Maximum Payload Mass] DESC
```

This Query list the names of the booster which have carried the maximum payload mass. To do so, we use

# 2015 Launch Records

```
SELECT booster_version, launch_site FROM spacextable WHERE LandingOut =
'Failure (drone ship)' and DATE LIKE '2015-%'
```

This query list the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015. To do so, we use some conditions like the year and the type of outcome. Then we returns the booster version and the launch site.

| booster_version | launch_site |
|---|---|
| F9 v1.1 B1012 | CCAFS LC-40 |
| F9 v1.1 B1015 | CCAFS LC-40 |

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

```
SELECT COUNT(Landing_Outcome) AS sl FROM dbo.spacextable WHERE (Landing_Outcome LIKE
'%Success%') AND (Date >'04-06-2010') AND (Date < '20-03-2017’)
```

This Query 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. To do so, we use function Count that counts records in column and Where to filter data. At last, we use conditions with AND and LIKE.

Section 4

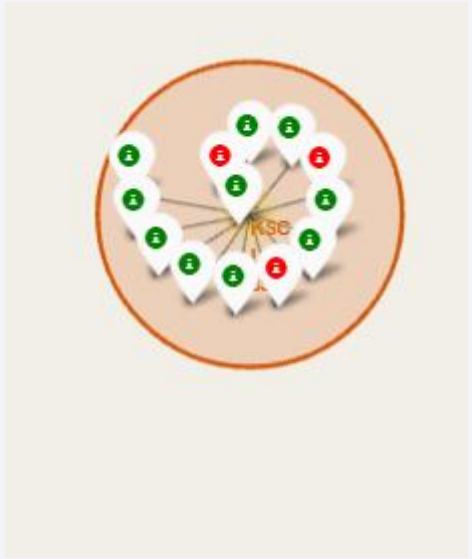
# Launch Sites Proximities Analysis

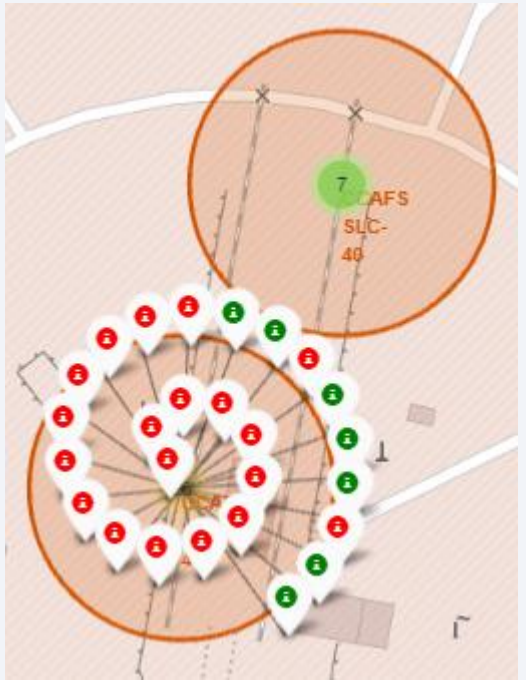
# All Launch sites global map markers

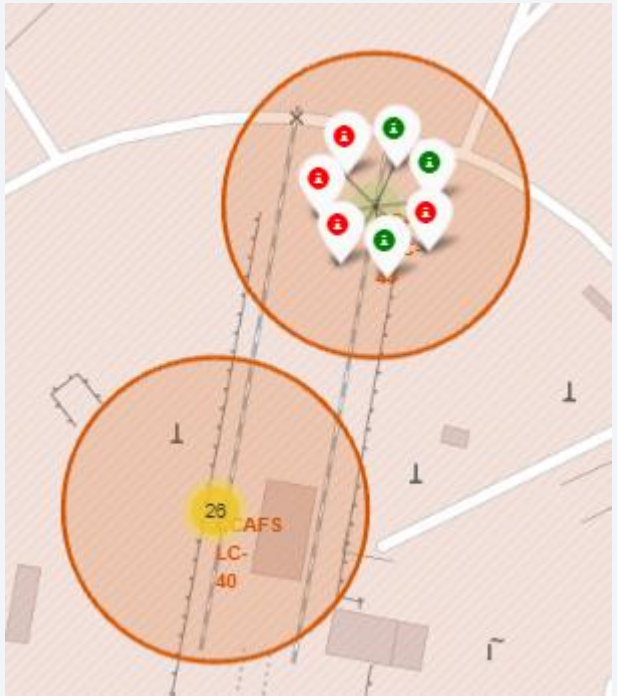
The screenshoot show us that all the Launch sites were located at the United States of America coasts

# Colour Labelled Markers

- By the left, we show the California Launch Site;
- The others three screenshoots  shows the Launch Sites located at Florida;
- The green Marker represents successful Launches and Red Marker represents Failures Launches.

# CCAFS-SLC-40 distance to landmarks

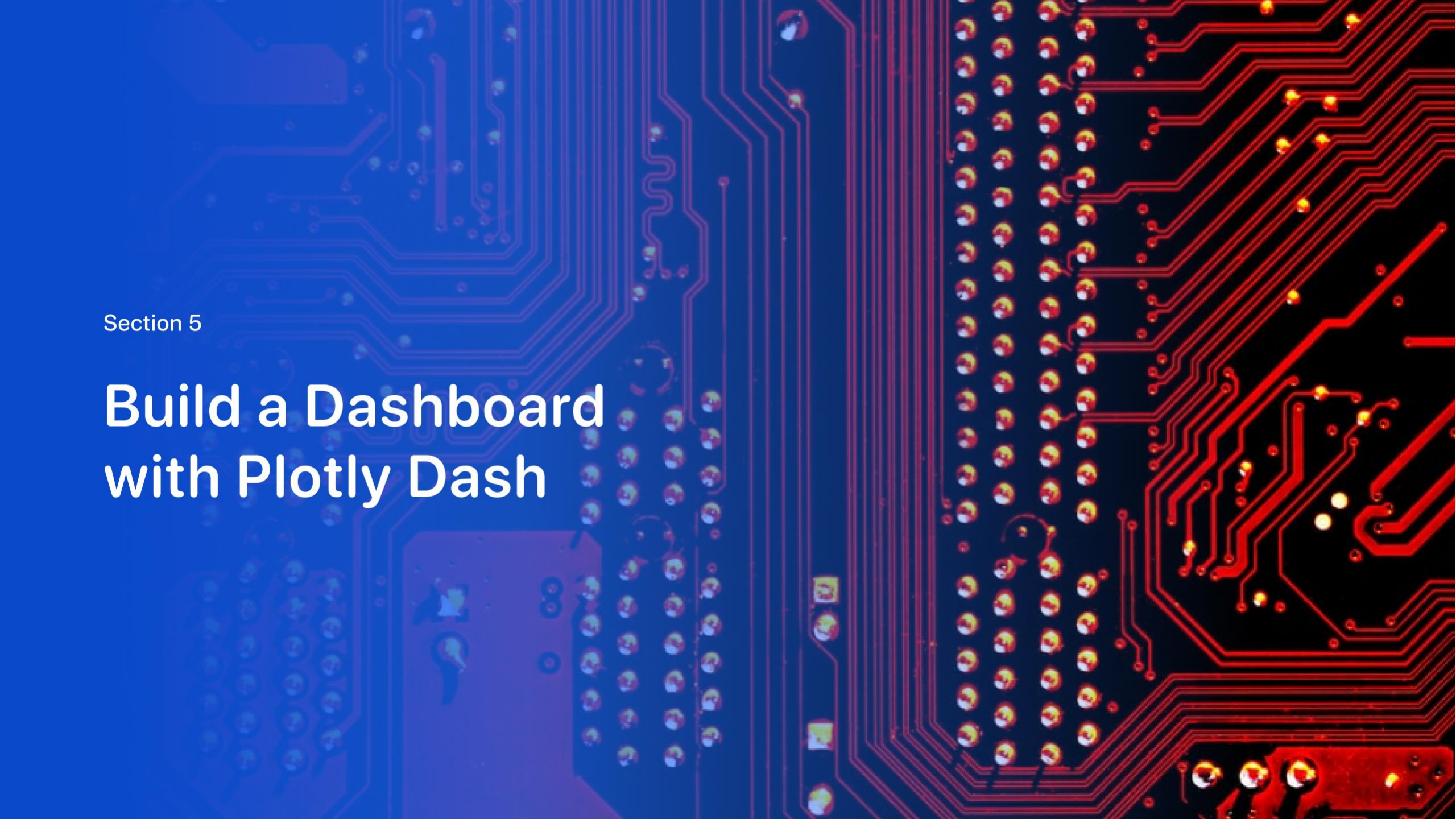
- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

- Distance from Highway:


Distance from cost:

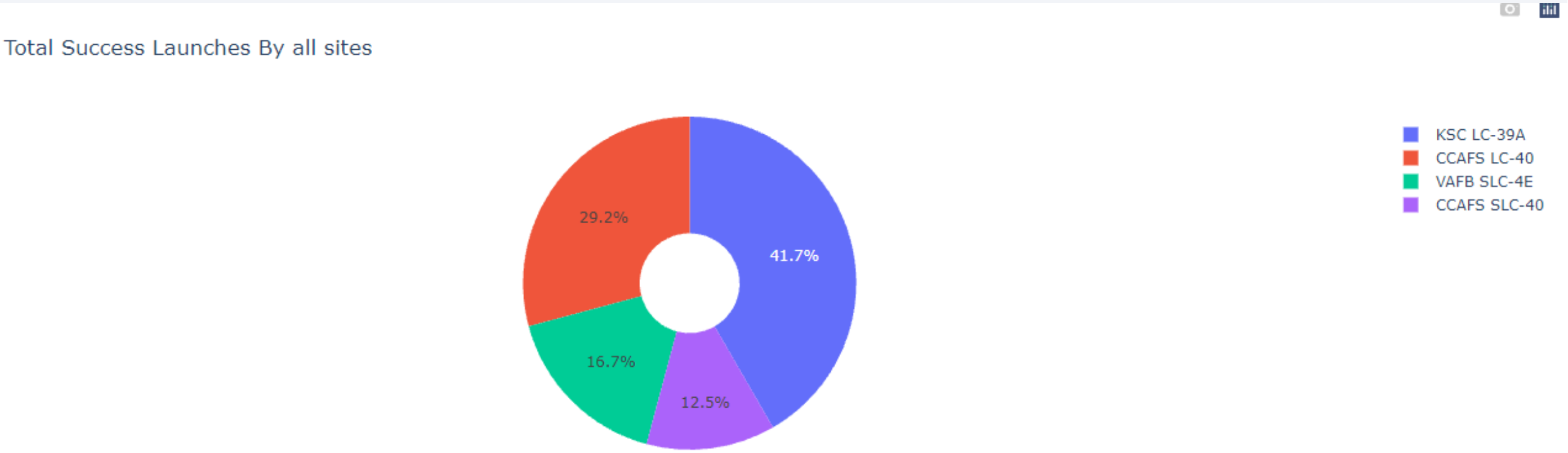
Section 5

# Build a Dashboard with Plotly Dash



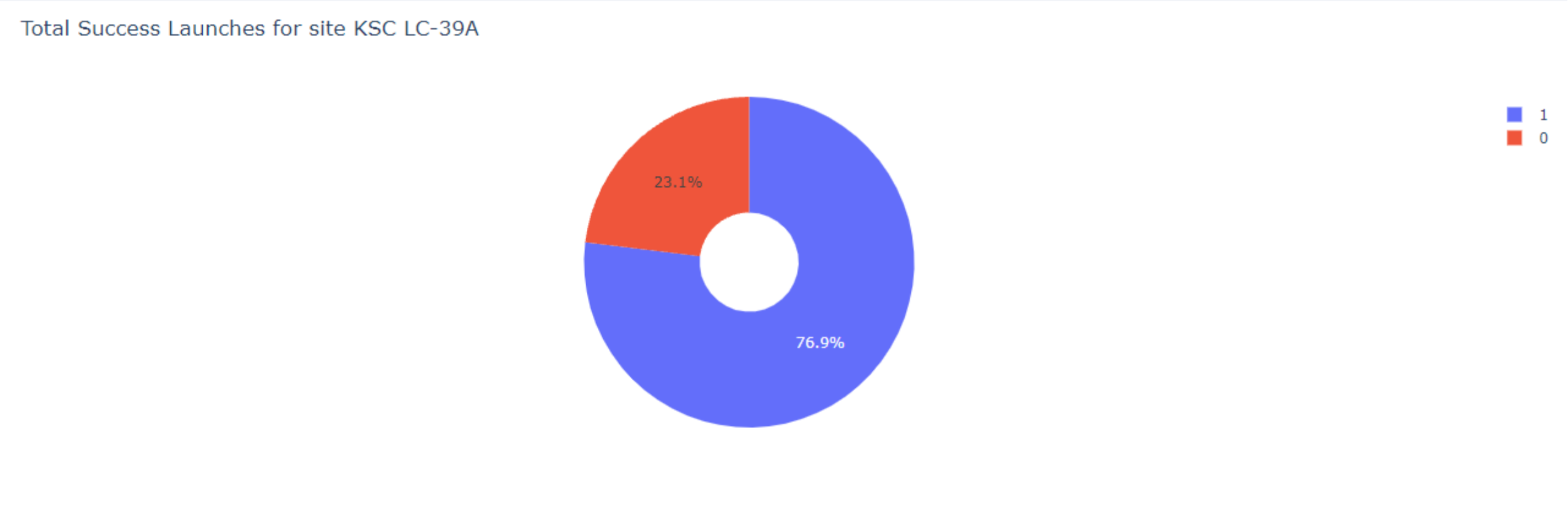
# Dashboard: Pie chart showing the success percentage achieved group by launch site

- A logical conclusion, as the pie chart shows, KSC LC-39A had the most part of successful launches when compared to others



# Dashboard: Pie Chart for the launch site with highest Launch Success Ratio

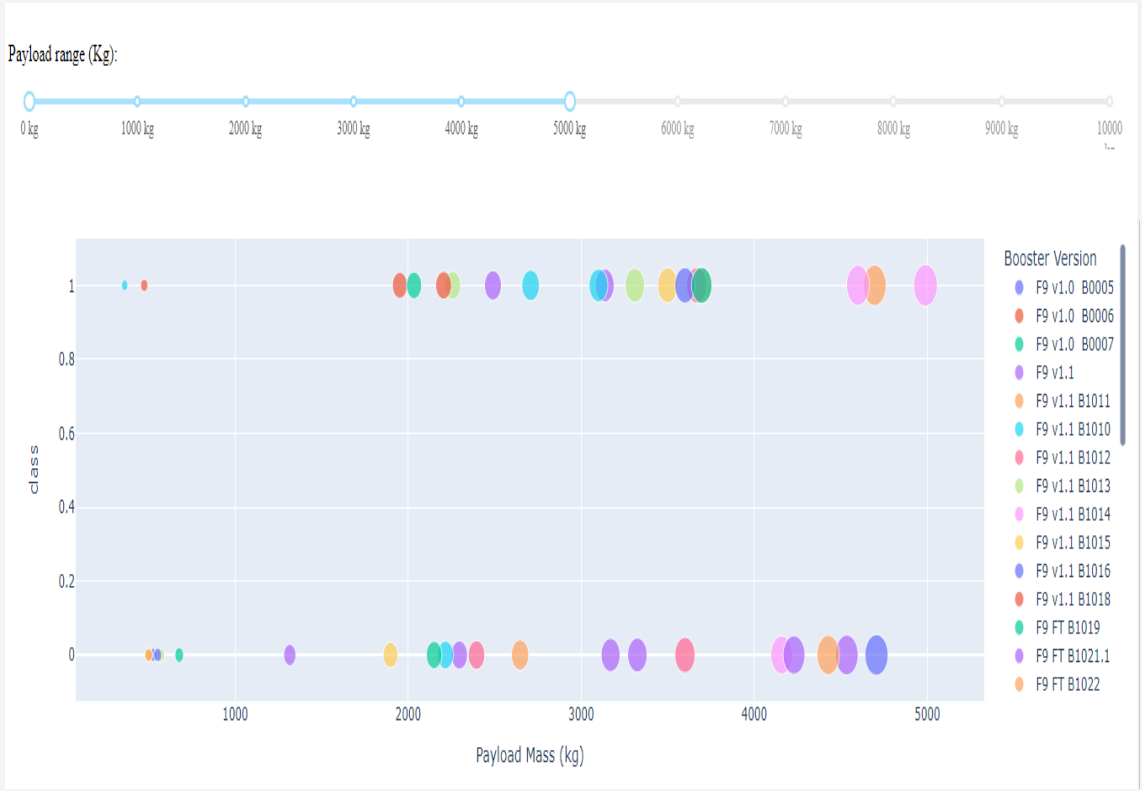
Due to the last graph, let's take a closer look at the KSC LC-39A successful rate. As we can see, this site launch achieved more then ¾ of success rate.



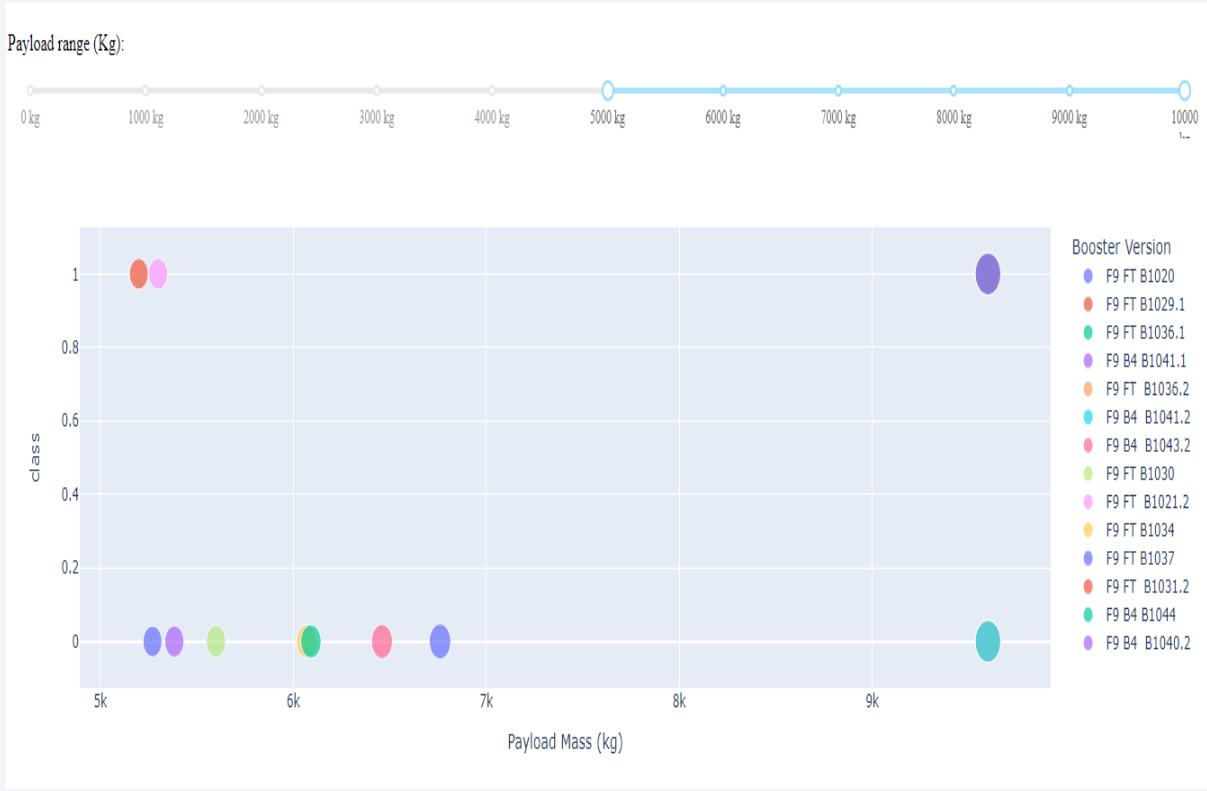
# Dashboard: Payload x Launch Outcome scatter plot for all sites, with different payload selected in the range slider

From the graphics below, we can infer that the greater the weight payloads, the lower will be the probability of success.

## Low Weighted Payload 0 Kg – 5000 Kg




## Heavy Weighted Payload 5000 Kg – 10000 Kg



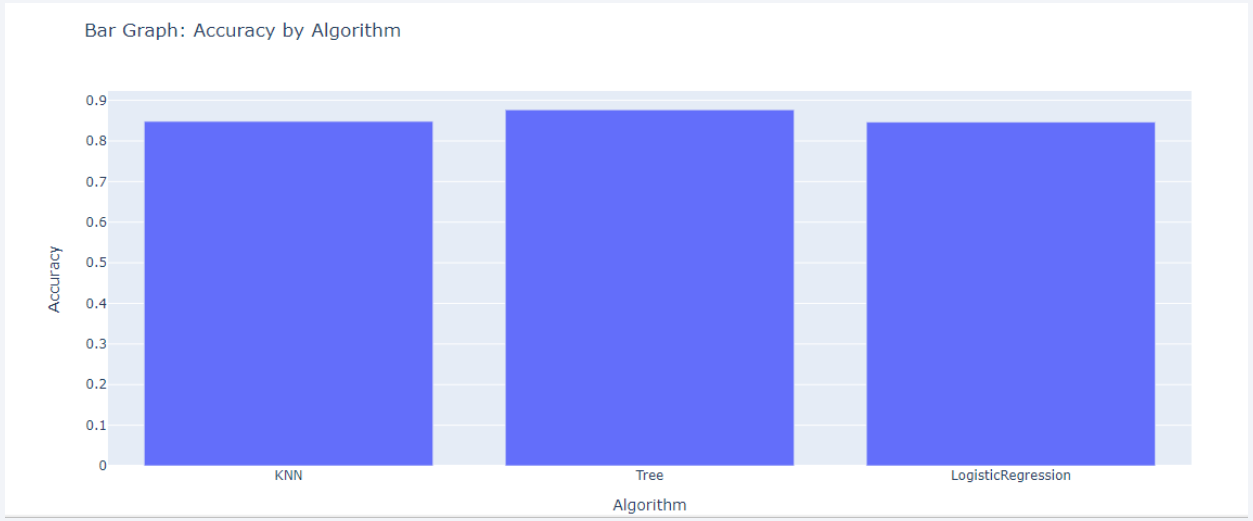
Section 6

# Predictive Analysis (Classification)

# Classification Accuracy

The Best Algorithm is the  Decision Tree with a score of 0.8767857142857143

```
Best Params are:  {'criterion': 'gini', 'max_depth': 12, 'max_features': 'auto', 'min_samples_leaf': 4, 'min_samples_split': 5, 'splitter': 'random'}
```

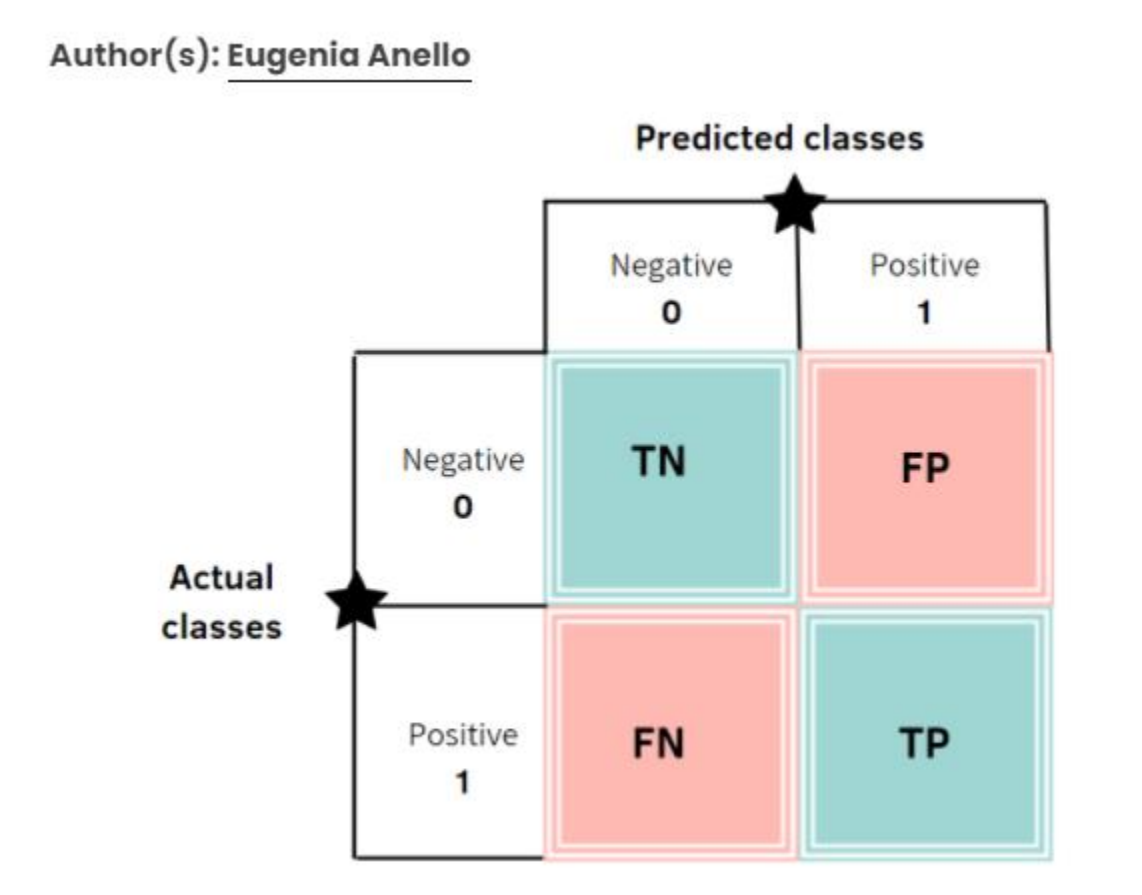


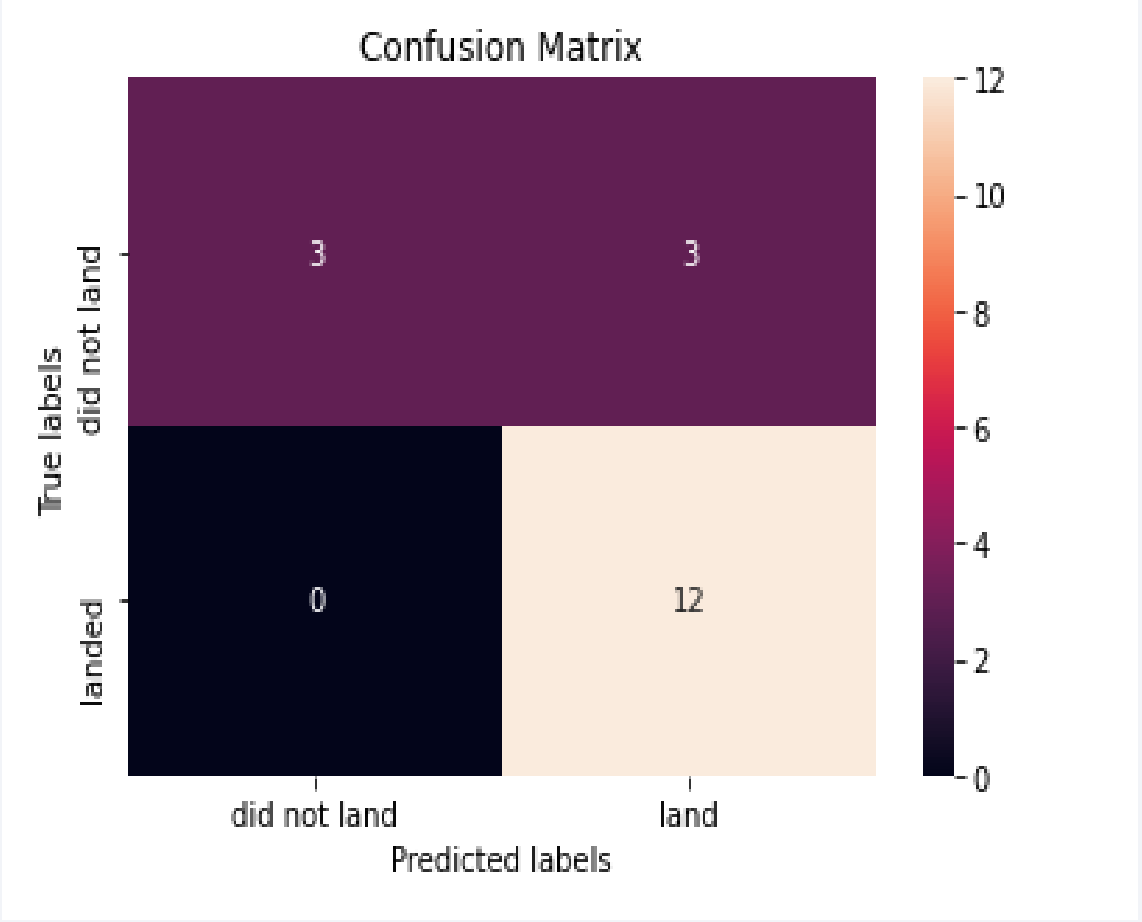

```
In [31]: print(pd.DataFrame(list(algorithms.items()),
                            columns=['Algorithm', 'Accuracy']))
```

| | Algorithm | Accuracy |
|---|---|---|
| 0 | KNN | 0.848214 |
| 1 | Tree | 0.876786 |
| 2 | LogisticRegression | 0.846429 |

# Confusion Matrix

- Looking at the confusion matrix we can see that tree can classify classes differently. The problem are under de scope of false positives



# Conclusions

- Orbit GEO, Heo, SSO, ES-L1 has the best Success Rate
- KSC LC-39A had the most successful launches from all sites
- Low weighted payloads perform better than the heavier payloads
- The Tree Classifier Algorithm is the best for Machine Learning for this dataset

# Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

# Appendix – SQL queries

## Query to display 5 records launched on Friday

```
%sql SELECT * FROM spacextable where DAYNAME(DATE)='Friday' LIMIT 5
```

```
 * ibm_db_sa://rbl67840:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/BLUDB
Done.
```

| DATE | time__utc_ | booster_version | launch_site | payload | payload_mass__kg_ | orbit | customer | mission_outcome | landingout |
|---|---|---|---|---|---|---|---|---|---|
| 2010-06-04 | 18:45:00 | F9 v1.0 B0003 | CCAFS LC-40 | Dragon Spacecraft Qualification Unit | 0 | LEO | SpaceX | Success | Failure (parachute) |
| 2013-03-01 | 15:10:00 | F9 v1.0 B0007 | CCAFS LC-40 | SpaceX CRS-2 | 677 | LEO (ISS) | NASA (CRS) | Success | No attempt |
| 2014-04-18 | 19:25:00 | F9 v1.1 | CCAFS LC-40 | SpaceX CRS-3 | 2296 | LEO (ISS) | NASA (CRS) | Success | Controlled (ocean) |
| 2016-03-04 | 23:35:00 | F9 FT B1020 | CCAFS LC-40 | SES-9 | 5271 | GTO | SES | Success | Failure (drone ship) |
| 2016-04-08 | 20:43:00 | F9 FT B1021.1 | CCAFS LC-40 | SpaceX CRS-8 | 3136 | LEO (ISS) | NASA (CRS) | Success | Success (drone ship) |

# Appendix - SQL queries

## Retrieve the most recent date from the spacex table

```
%sql SELECT max(Date) AS last_launch from spacextable
```

```
 * ibm_db_sa://rbl67840:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/BLUDB
Done.
```

| last_launch |
| --- |
| 2020-12-06 |

## A Query that display the minimum payload mass

```
%sql select min(payload_mass__kg_) AS Minimum_payload_mass from spacextable
```

```
 * ibm_db_sa://rbl67840:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/BLUDB
Done.
```

| minimum_payload_mass |
| --- |
| 0 |

# Thank you!