IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Chan Wei Bin
30 June 2023

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection with API
  - Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis with Visualization
  - Interactive Visual Analytics with Folium
  - Prediction Analysis with Machine Learning
- Summary of all results
  - Exploratory Data Analysis result
  - Interactive analytics in screenshots
  - Predictive Analytics result from Machine Learning Lab

Introduction

This project aims to predict the successful landing of the Falcon 9 first stage, a critical aspect of SpaceX's rocket launches. By examining the feasibility of the first stage's recovery, we can assess the cost implications associated with launch operations. SpaceX, renowned for its innovative approach, offers Falcon 9 rocket launches at a significantly lower price point of 62 million dollars compared to other providers who charge upwards of 165 million dollars for each launch. This substantial cost advantage stems from SpaceX's ability to reuse the first stage, making it imperative to determine its landing success rate. The insights derived from this analysis will prove invaluable for any alternate company aspiring to compete with SpaceX in bidding for rocket launch contracts.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using SpaceX REST API and web scrapping from Wikipedia
- Perform data wrangling
  - Data was processed using one-hot encoding for categorical features
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using SpaceX REST API and web scrapping from Wikipedia
- Perform data wrangling
  - Data was processed using one-hot encoding for categorical features
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Data was processed using one-hot encoding for categorical features

Data Collection

- Data collection involves gathering and measuring information to answer questions and evaluate outcomes. In this project, the dataset was collected from Wikipedia using REST API and web scraping.
- For REST API, a GET request was made, and the response content was decoded as JSON. The data was transformed into a pandas dataframe using json_normalize(). Cleaning steps were performed, including handling missing values.
- Web scraping involved using BeautifulSoup to extract launch records from an HTML table. The extracted data was parsed and converted into a pandas dataframe for analysis.
- These data collection methods, including REST API and web scraping, provided the necessary dataset for analysis in this project.

Data Collection – SpaceX API

- Data were collected from api using Python
  1. Request data from URL
  2. Normalize and convert json to readable and structured dataframe
  3. Data cleansing by removing unwanted data
- Refer GitHub for more info: Data Collection with API

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

response = requests.get(spacex_url)
```

```
# Use json_normalize meethod to convert the json
data = pd.json_normalize(response.json())
```

```
# Lets take a subset of our dataframe keeping only the features we want and the flight num
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra ro
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single value in th
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting the date
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

Data Collection - Scraping

- Get data from URL, followed by extracting useful data from table:
    1. Get the table by tag <th>
    2. Get each row by <tr>
    3. Convert into list, then into dataframe
- Refer to GitHub for more: [WebScrapping]

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

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

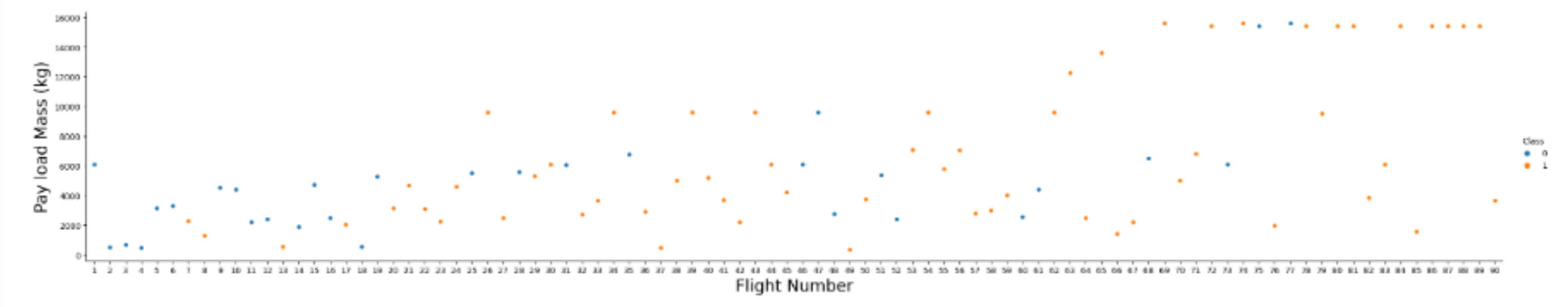
```
temp = soup.find_all('th')
# Iterate each th element and apply the provided ex
for x in range(len(temp)):
    try:
        name = extract_column_from_header(temp[x])
        if (name is not None and len(name)>0):
            column_names.append(name)
    except:
        pass
```

Data Wrangling

- Data wrangling is the process of cleaning, transforming, and organizing raw data into a format suitable for analysis. It involves tasks such as handling missing values, standardizing data types, resolving inconsistencies, and restructuring data to ensure its quality and usability for further analysis.
- To perform Data Wrangling, we will:
  1. first calculate the number of launches on each
  2. site, then calculate the number and occurrence of mission outcome per orbit type.
  3. create a landing outcome label from the outcome column.
  4. analysis, visualization, and ML

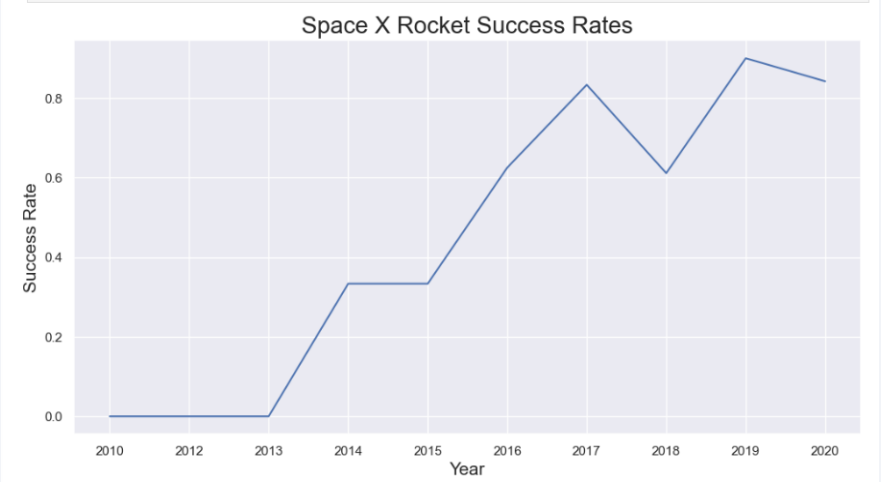
EDA with Data Visualization

- Scatterplot were used in each EDA to visually examine the relationship between two variables. They help identify patterns, trends, and potential correlations between variables, providing insights into the nature of the relationship and aiding in the identification of outliers or clusters within the data.
- Please refer to GitHub to view other graph: EDA with Viz

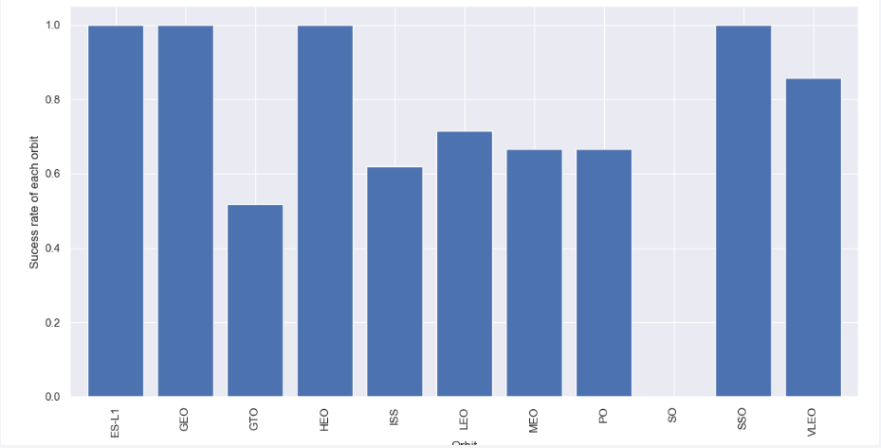


EDA with Data Visualization

- Line graphs are used in exploratory data analysis (EDA) to depict the relationship between two continuous variables over a continuous or ordered time or numerical axis. They are effective in showing trends, patterns, or changes in data over time and identifying any fluctuations, seasonality, or overall patterns in the data. Line graphs are particularly useful for analyzing time series data and making predictions based on historical trends.



- Bar graphs are used in exploratory data analysis (EDA) to visualize categorical or discrete variables and their frequencies or distributions. They provide a visual representation of the count or proportion of each category, allowing for easy comparison between different categories and identifying patterns, trends, or disparities within the data.



EDA with SQL

- SQL Queries performed included:
  - `select distinct Launch_Site from SPACEXTBL`
  - `select * from SPACEXTBL where Launch_Site like 'CCA%' limit 5`
  - `select SUM(PAYLOAD_MASS__KG_) from SPACEXTBL WHERE CUSTOMER='NASA (CRS)'`
  - `elect AVG(PAYLOAD_MASS__KG_) from SPACEXTBL WHERE Booster_Version='F9 v1.1'`
  - `select min(date) from SPACEXTBL where Mission_Outcome = 'Success'`
- Please refer to GitHub for more: SQL

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 then assigned the dataframe launch_outcomes(failure,success) to classes 0 and 1 with Red and Green markers on the map in MarkerCluster().

We then used the Haversine's formula to calculated the distance of the launch sites to various landmark to find answer to the questions of:

- How close the launch sites with railways, highways and coastlines?
- How close the launch sites with nearby cities?

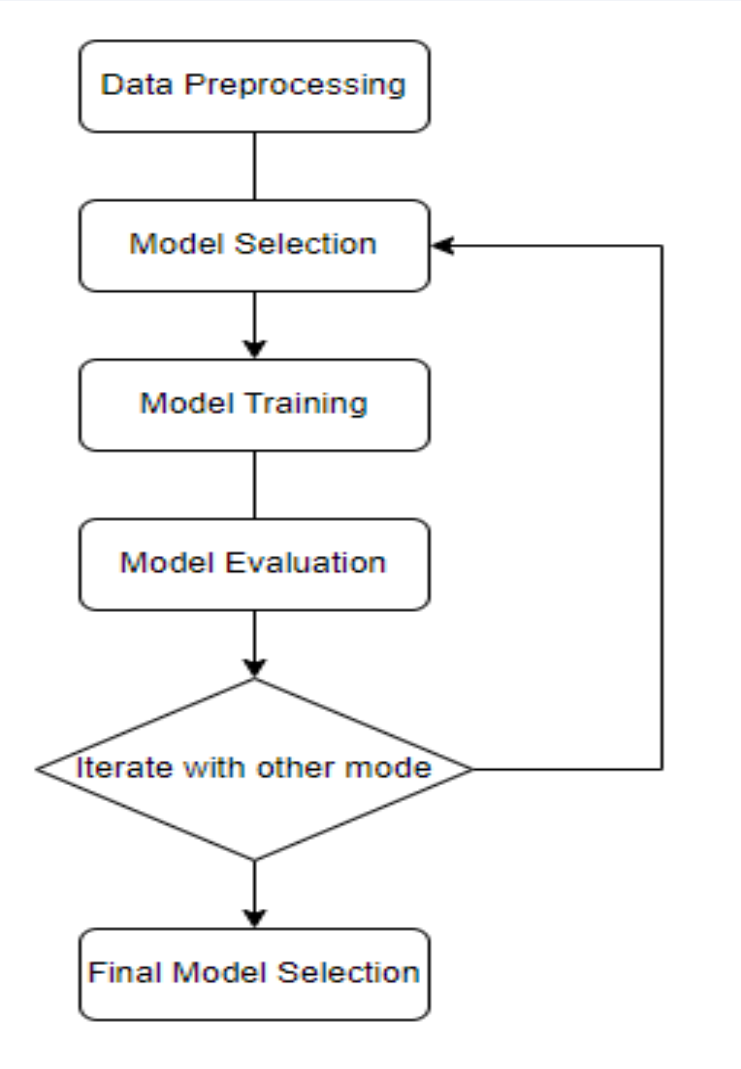
Please refer GitHub for more: Folium Viz

Build a Dashboard with Plotly Dash

An interactive dashboard was built using Plotly Dash for user to interactive with

- We plotted pie charts showing the total launches by a certain sites.
- We then plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.

Predictive Analysis (Classification)



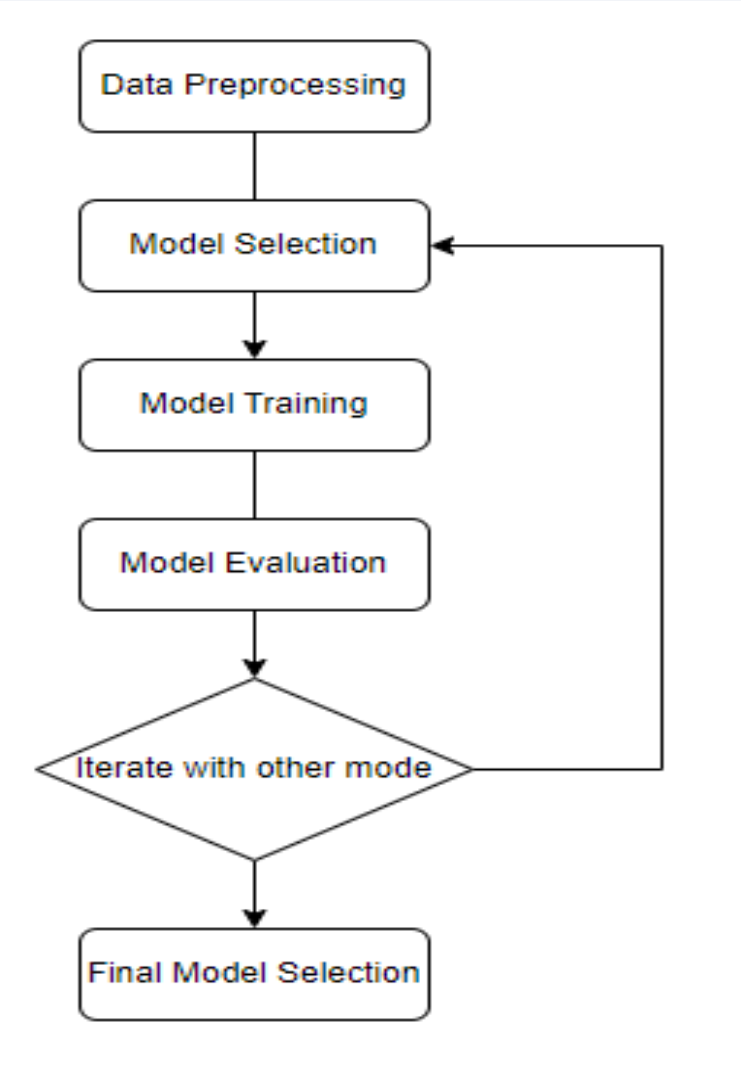
Data Preprocessing: Clean and preprocess the dataset by handling missing values

Model Selection: Choose a variety of classification algorithms suitable for the problem: logistic regression, decision trees, support vector machines

Model Training: Train each model using the training data.

Model Evaluation: Evaluate the trained models using scoring metrics

Predictive Analysis (Classification)



Final Model Selection: Select the best-performing classification model based on evaluation metrics and domain-specific requirements.

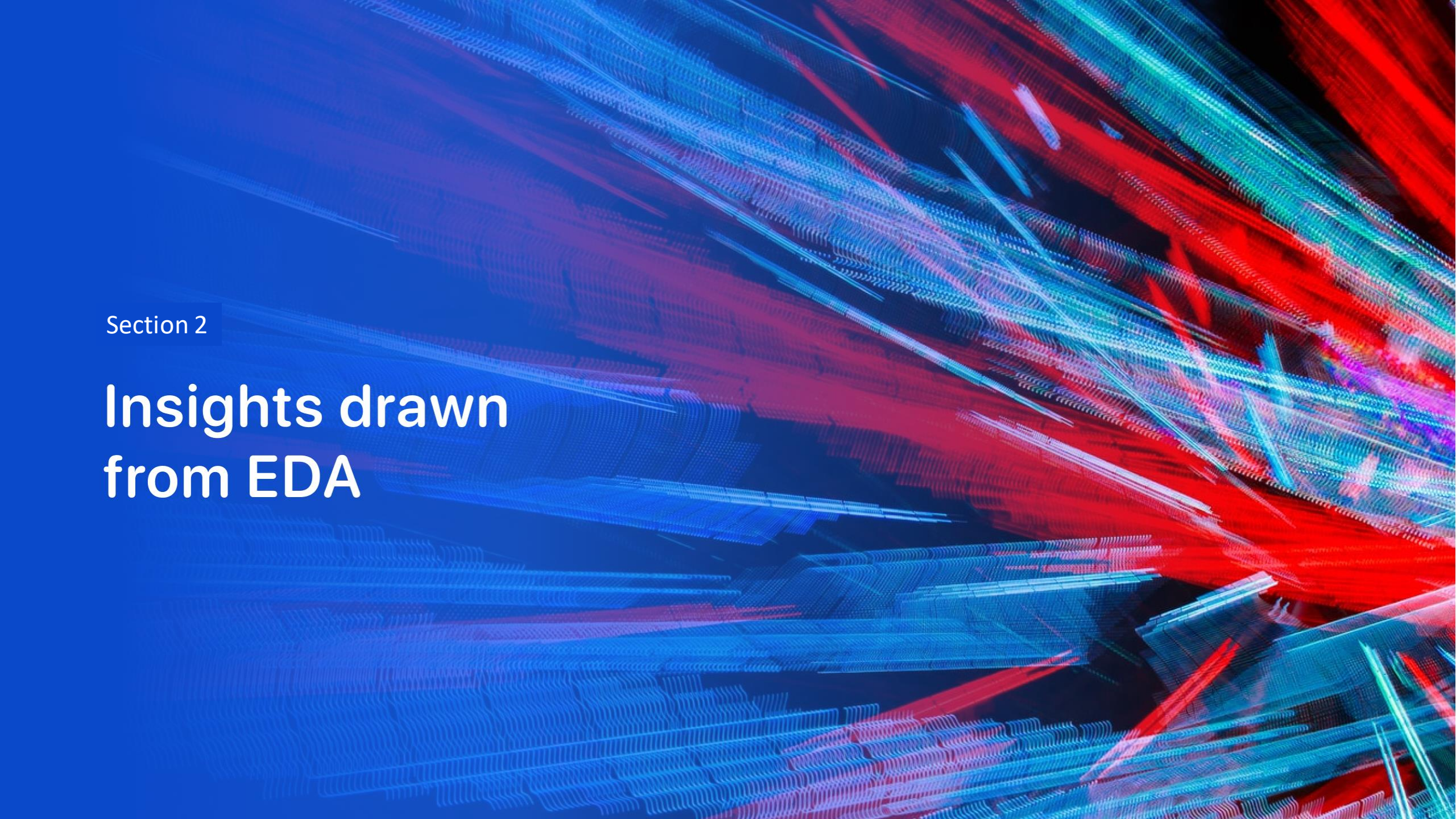
For more details please refer GitHub

Results

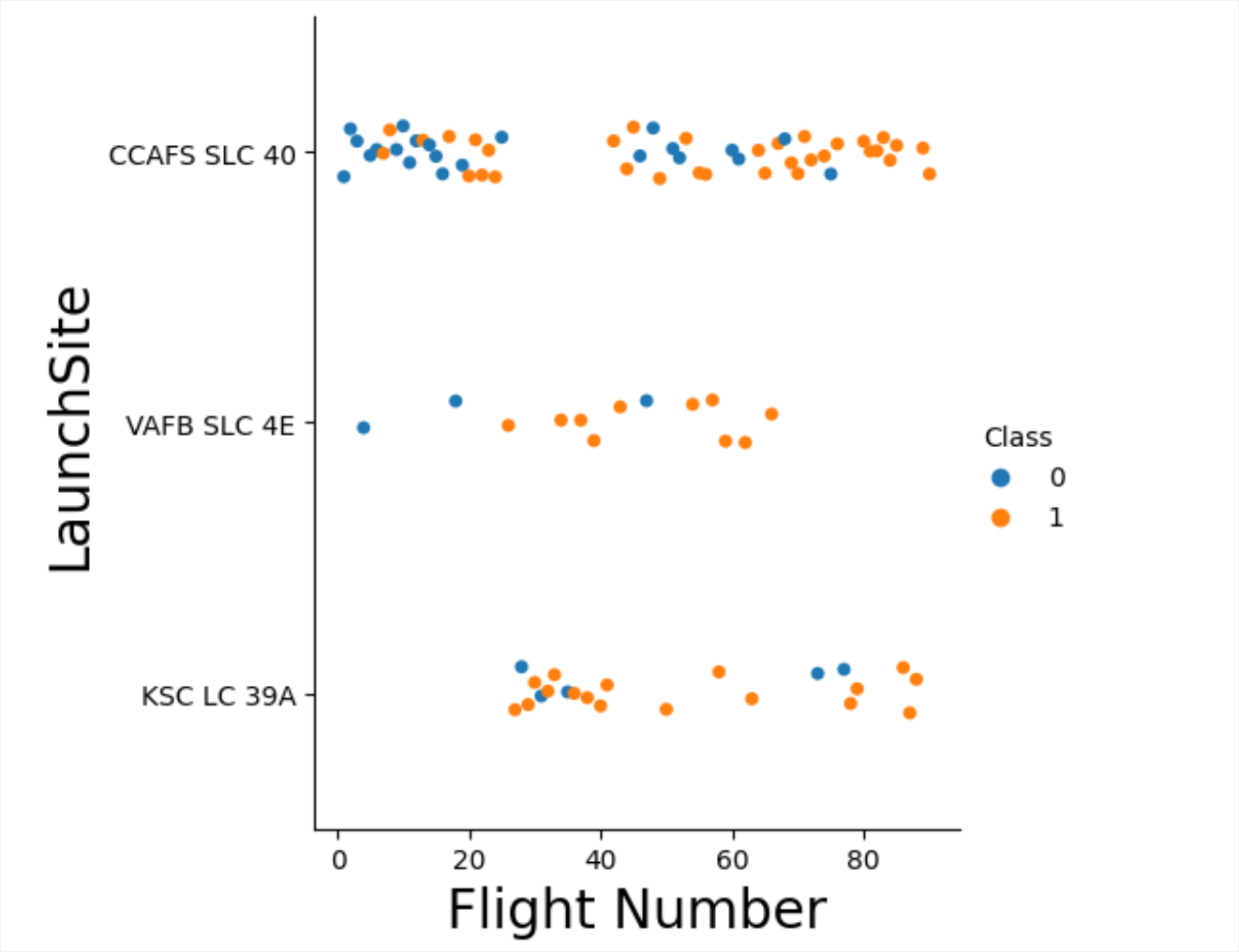
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

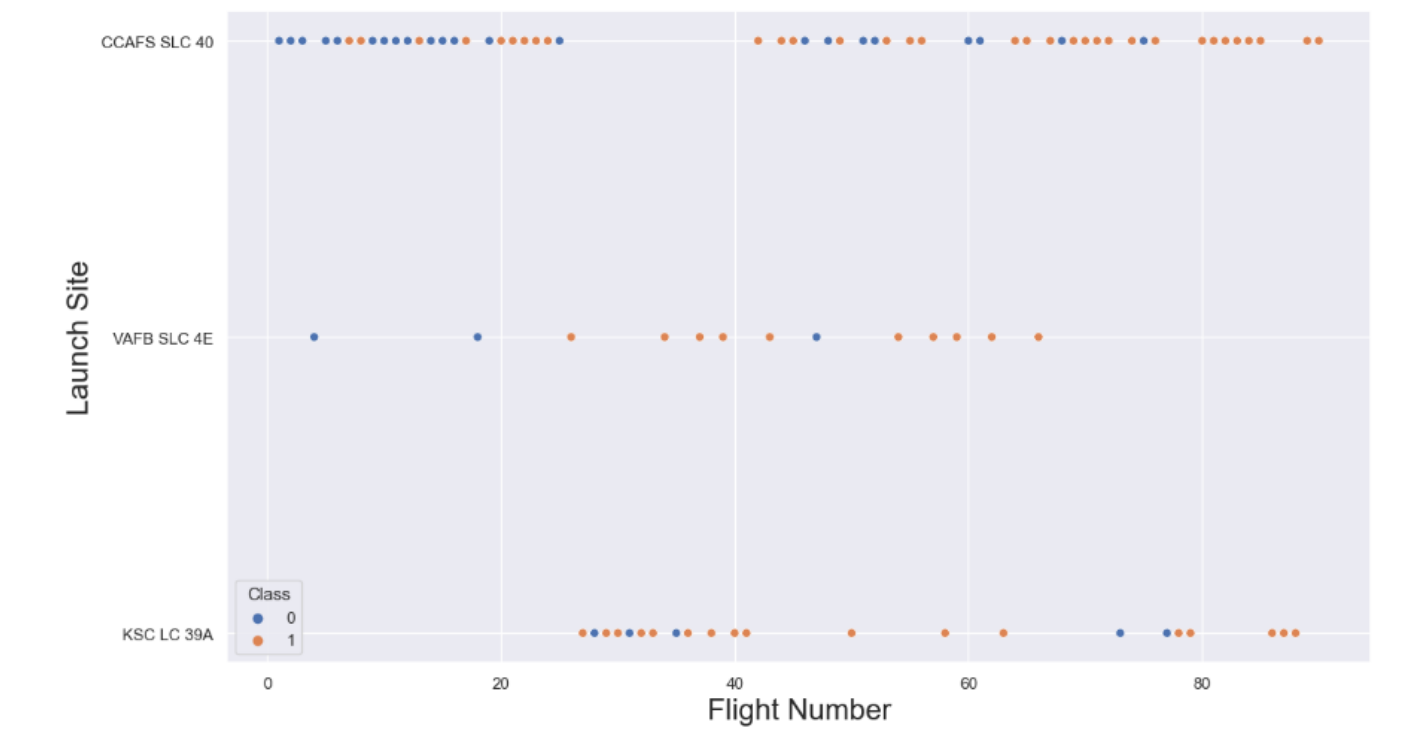


Flight Number vs. Launch Site



The catplot of Flight Number vs LaunchSite does not really show clear relationship

Flight Number vs. Launch Site



However, by scatterplot, we can clearly see that

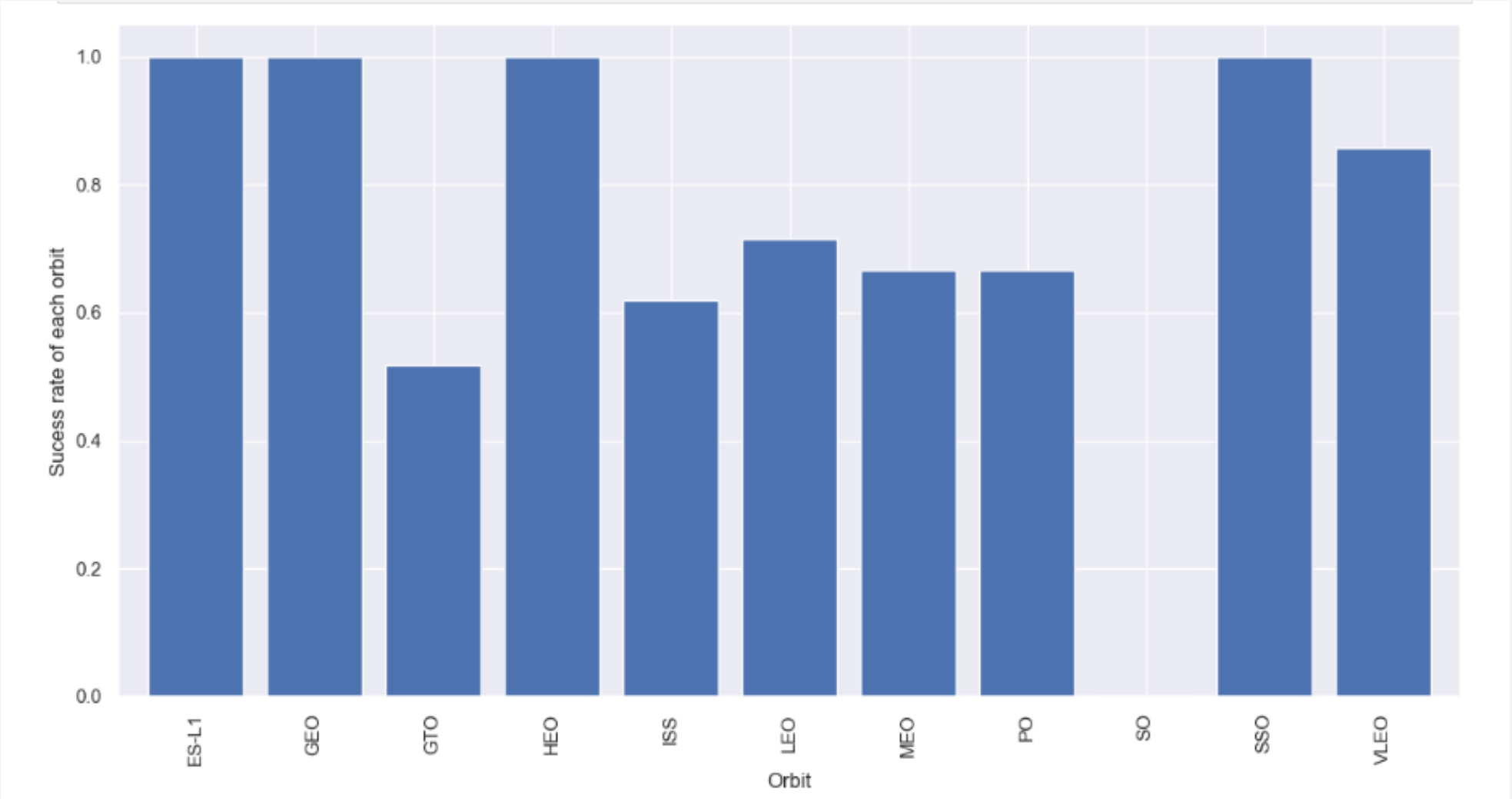
1. earlier flights has lower success rate
2. Later flight has higher success rate
3. CCAFS SLC 40 has highest launch rate

Payload vs. Launch Site

This scatter plot shows

1. Most launches with payload > 7000kg has higher success rate
2. KSC KC 39A success rate is 100% at payload < 6000kg

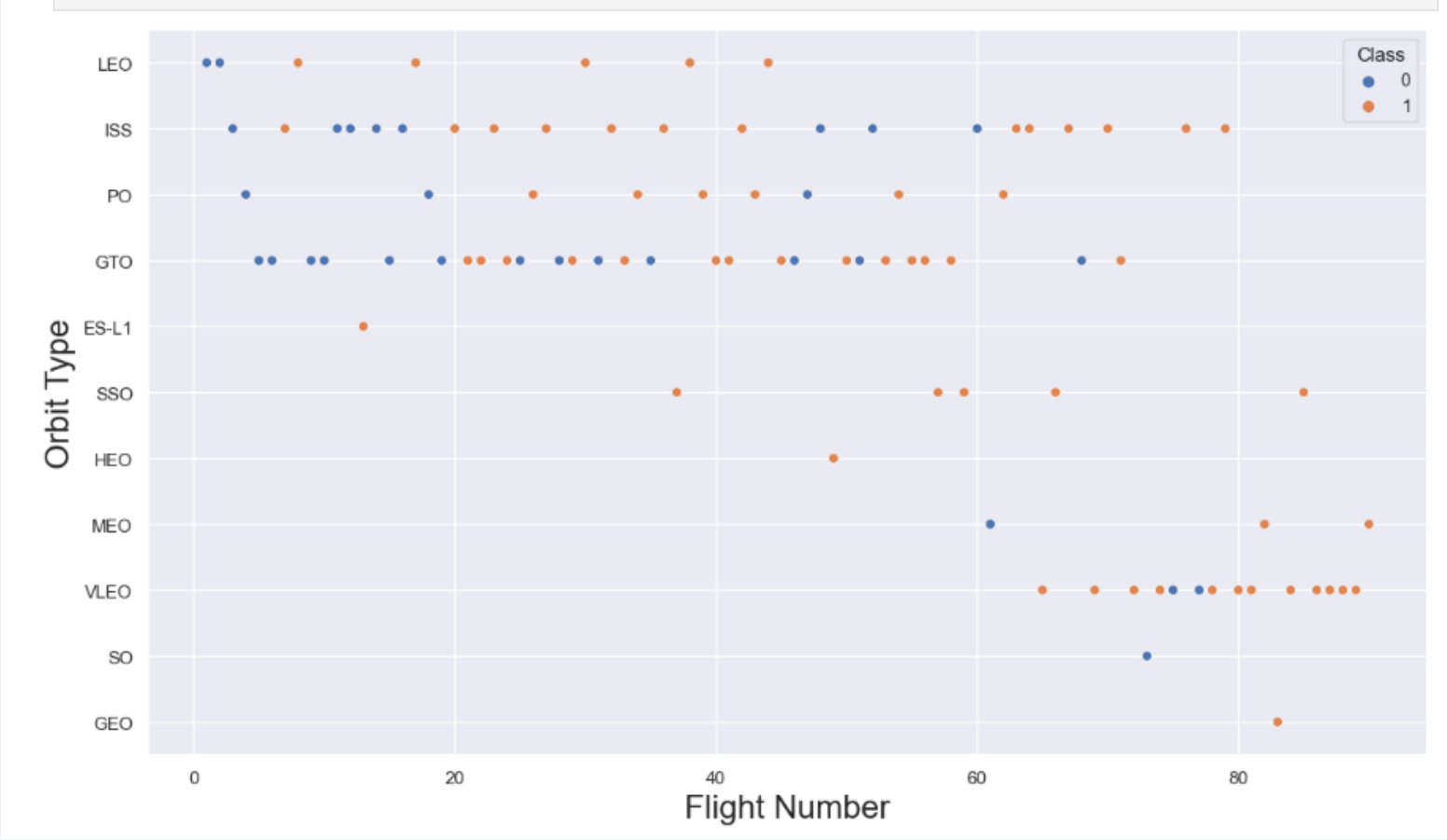
Success Rate vs. Orbit Type



Flight Number vs. Orbit Type

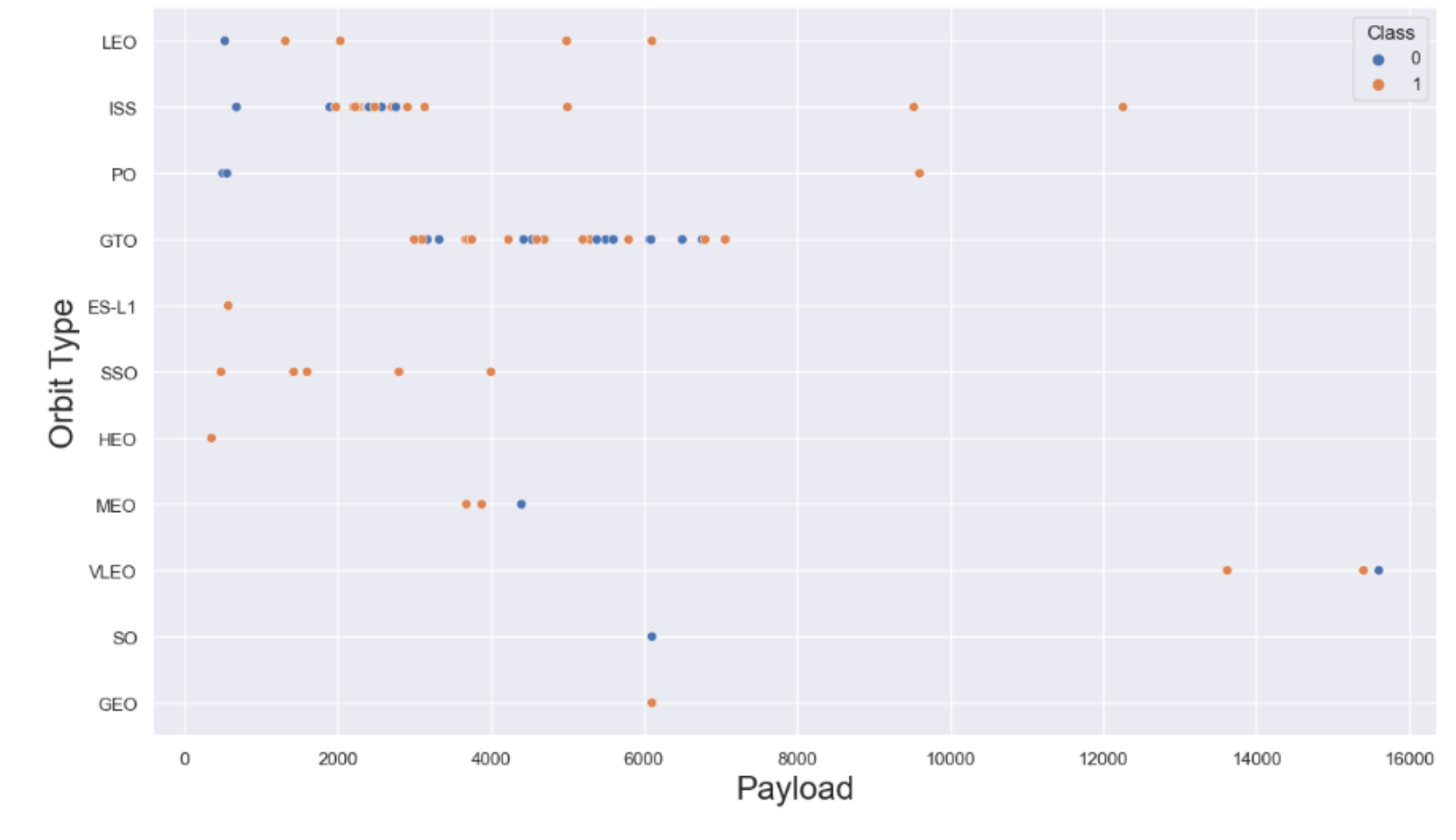
- From the scatter plot we observed:

1. Success rate increases with higher flight number
2. GTO has higher failure rate compare to other orbits

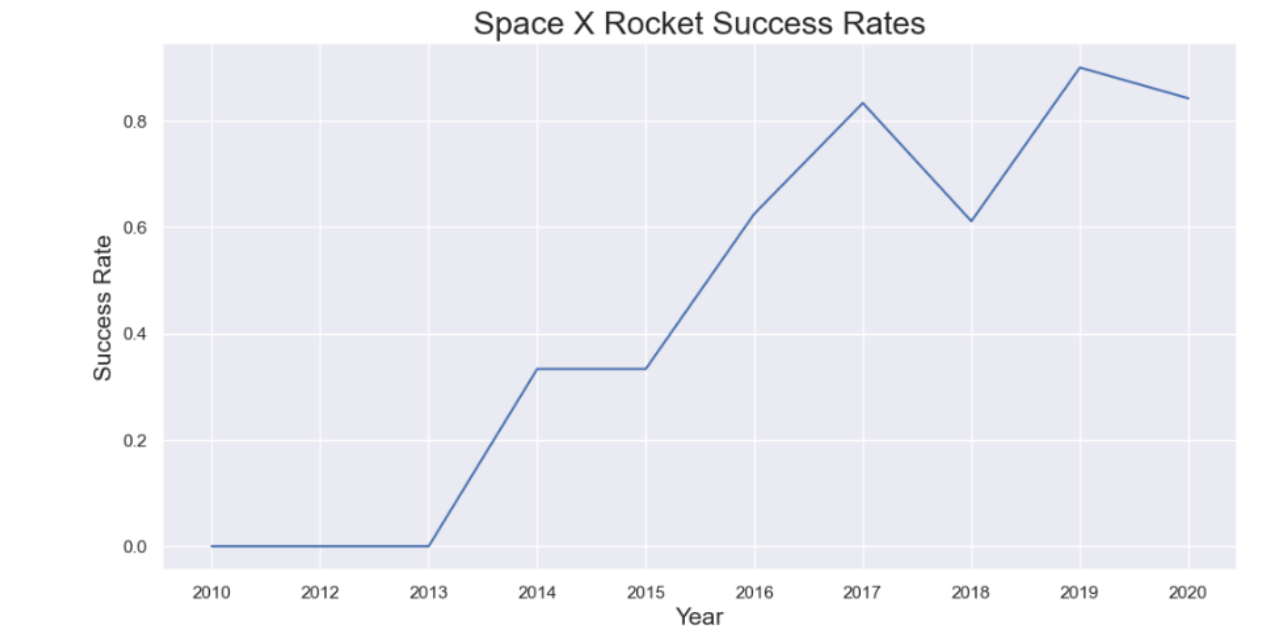


Payload vs. Orbit Type

- Heavy payloads are better with ISS
- SSO have 100% success rate on all tested payload
- PO, LEO, ISS has low success rate at very low payload



Launch Success Yearly Trend



- The success rate improved on two period: 2013-2017 and 2018-2019

All Launch Site Names

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

```
%sql select distinct Launch_Site from SPACEXTBL
```

- The query will get all disctinct launch_site name from table

Launch Site Names Begin with 'CCA'

```
%sql select * from SPACEXTBL where Launch_Site like 'CCA%' limit 5
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outc
06/04/2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0.0	LEO	SpaceX	Success	Failure (paracl
12/08/2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0.0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (paracl
22/05/2012	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525.0	LEO (ISS)	NASA (COTS)	Success	No att
10/08/2012	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500.0	LEO (ISS)	NASA (CRS)	Success	No att
03/01/2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677.0	LEO (ISS)	NASA (CRS)	Success	No att

- Limit keyword will set the number of record show, with wildcard query 'Like' and "%"

Total Payload Mass

```
%sql select SUM(PAYLOAD_MASS__KG_) from SPACEXTBL WHERE CUSTOMER='NASA (CRS)'
```

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

SUM(PAYLOAD_MASS__KG_)
45596.0

- The query will return Total by 'SUM' keyword

Average Payload Mass by F9 v1.1

```
%sql select AVG(PAYLOAD_MASS__KG_) from SPACEXTBL WHERE Booster_Version='F9 v1.1'
```

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

AVG(PAYLOAD_MASS__KG_)
2928.4

- The query AVG will return mean value of condition and query

First Successful Ground Landing Date

- Min keyword is used to get the smallest/earliest date of query

```
%sql select min(date) from SPACEXTBL where Mission_Outcome = 'Success'
```

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

min(date)
01/06/2014

Successful Drone Ship Landing with Payload between 4000 and 6000

Done.

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

- **%sql** select distinct Booster_Version from SPACEXTBL where (PAYLOAD_MASS__KG_ between 4000 and 6000) and Landing_Outcome like 'Success%drone ship%'

- To get all distinct name of Booster
- Between keyword to get numerical value between 4000 and 6000

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;
```

```
 ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud
ne.
```

Successful Mission	Failure Mission
100	1

By using sub query, and categorizing Failure and Success data to get 2 count return

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);
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

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 BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WHERE (substr(Date,7,4)='2015') AND LANDING_OUTCOME = 'Failure (dro
```

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

Booster_Version	Launch_Site
F9 v1.1 B1012	CCAFS LC-40
F9 v1.1 B1015	CCAFS LC-40

- The Date function is not available using SQLite, so substr() on date string is used

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

```
%sql SELECT COUNT(LANDING__OUTCOME) AS "Rank success count between 2010-06-04 and 2017-03-20" FROM SPACEX \
WHERE LANDING__OUTCOME LIKE '%Success%' AND DATE > '2010-06-04' AND DATE < '2017-03-20' ;
```

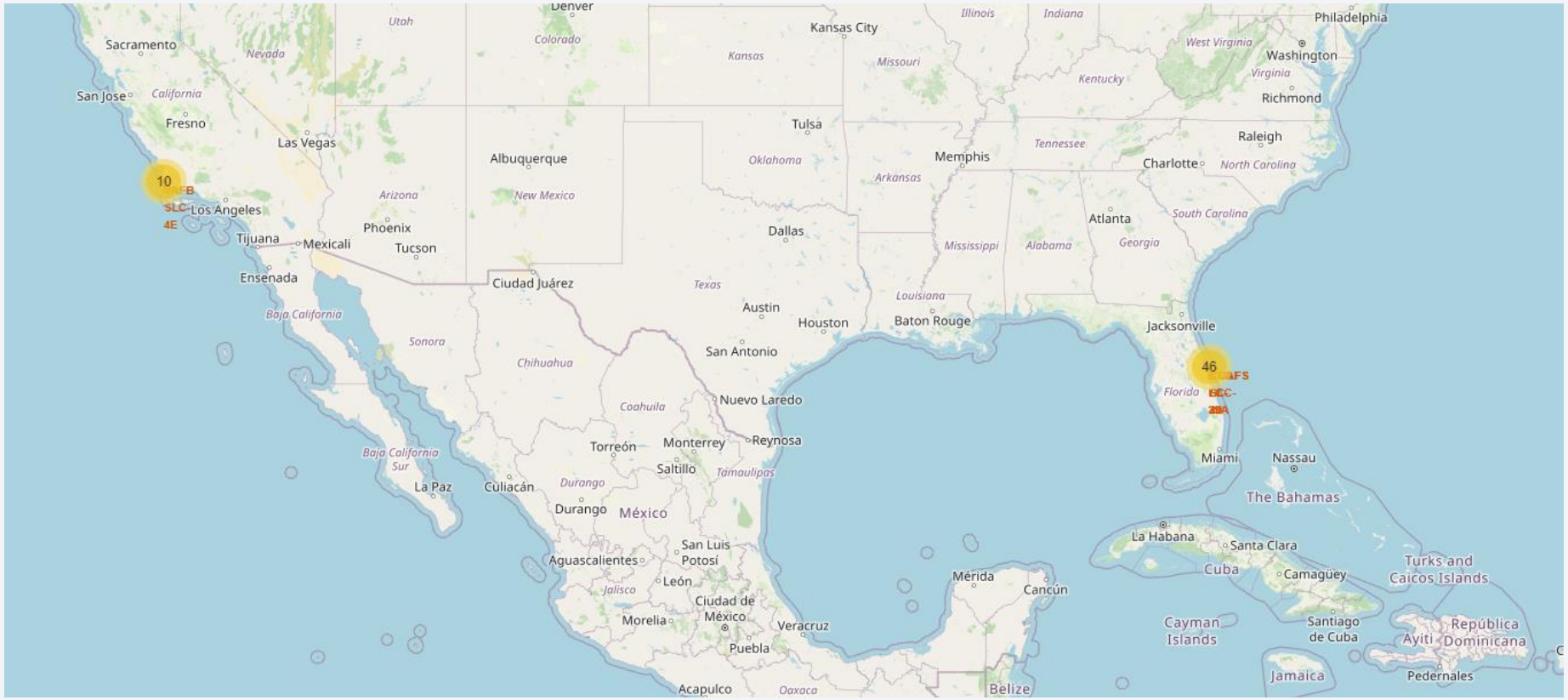
```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

Rank success count between 2010-06-04 and 2017-03-20
8

- By using between keyword, we can get the value of success rate using wildcard search

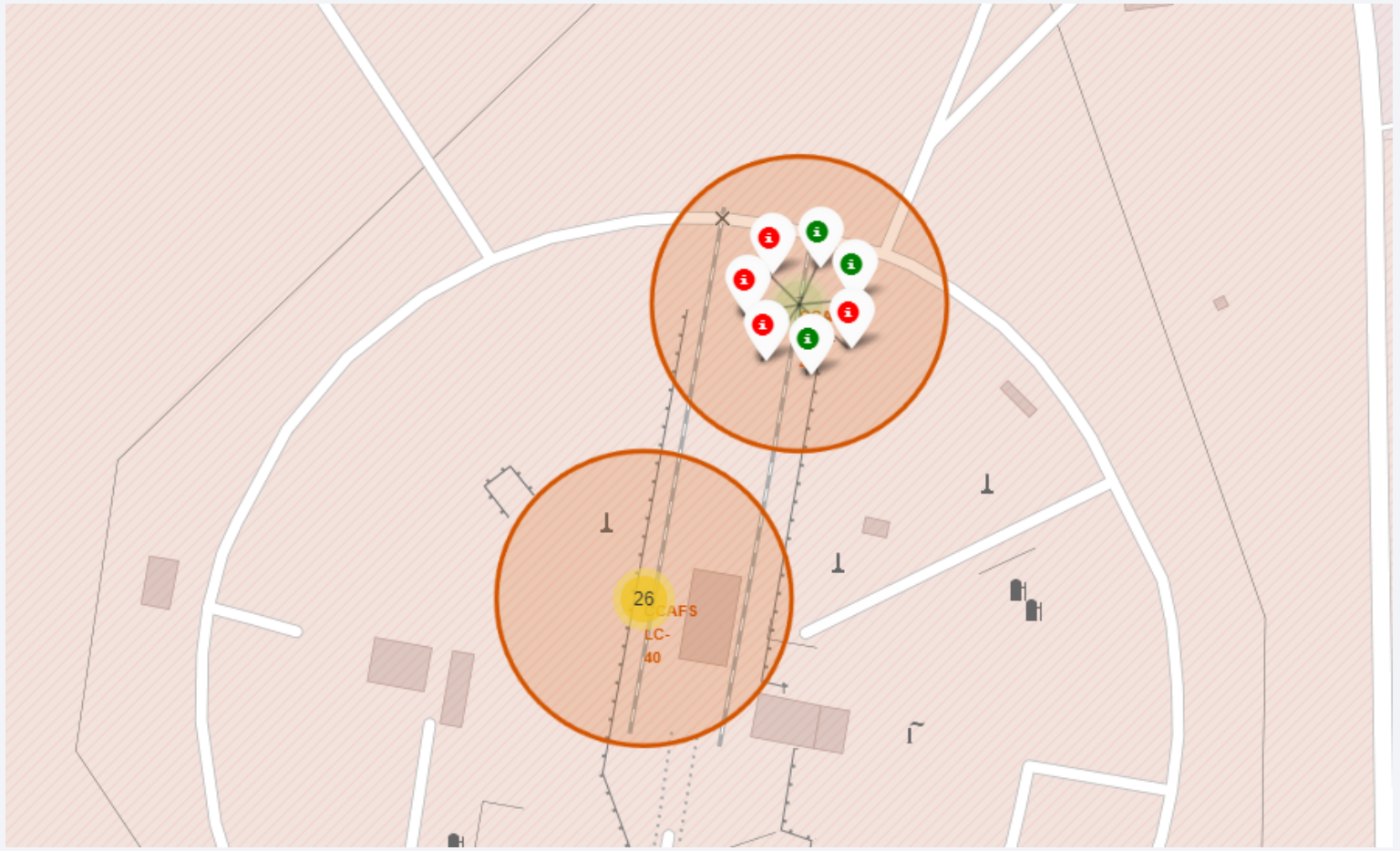

Section 3
Launch Sites
Proximities Analysis

Launch Sites



Markers are labelled on SpaceX launch sites

Launch Status



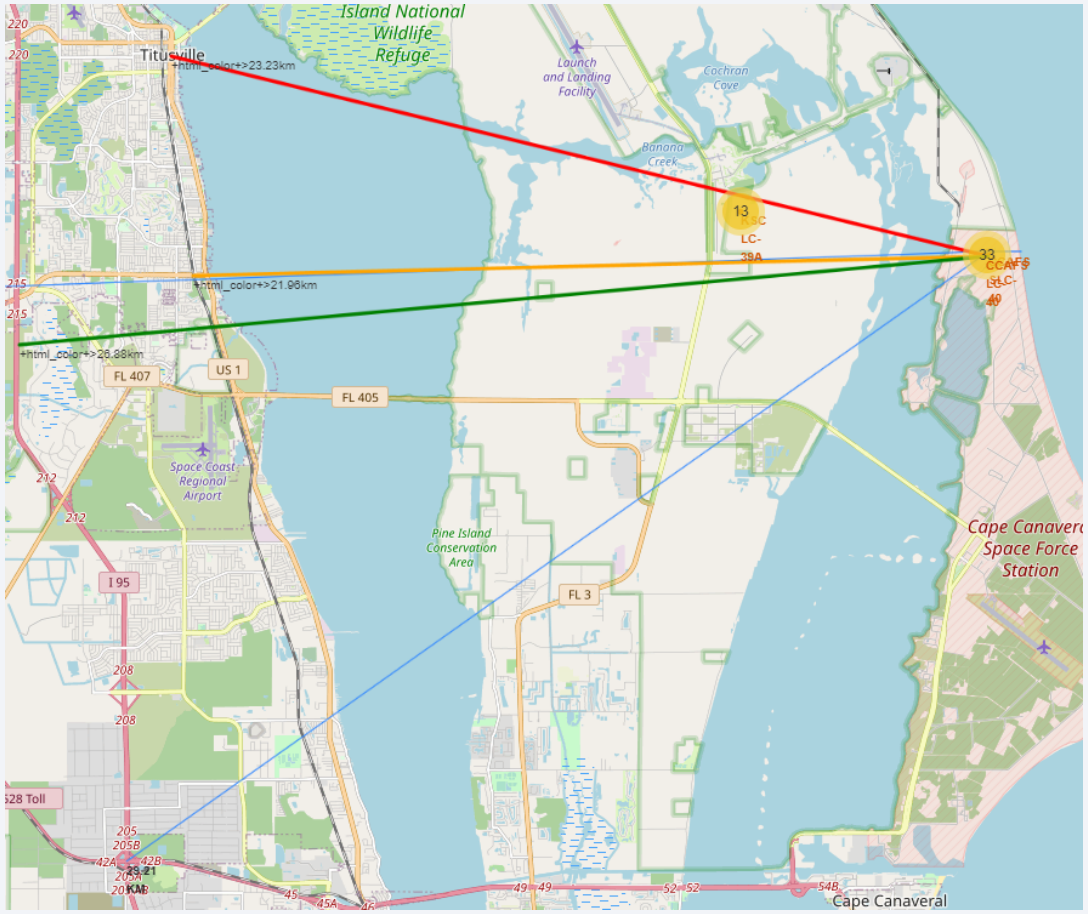
Green Marker

Show successful launches

Red Marker

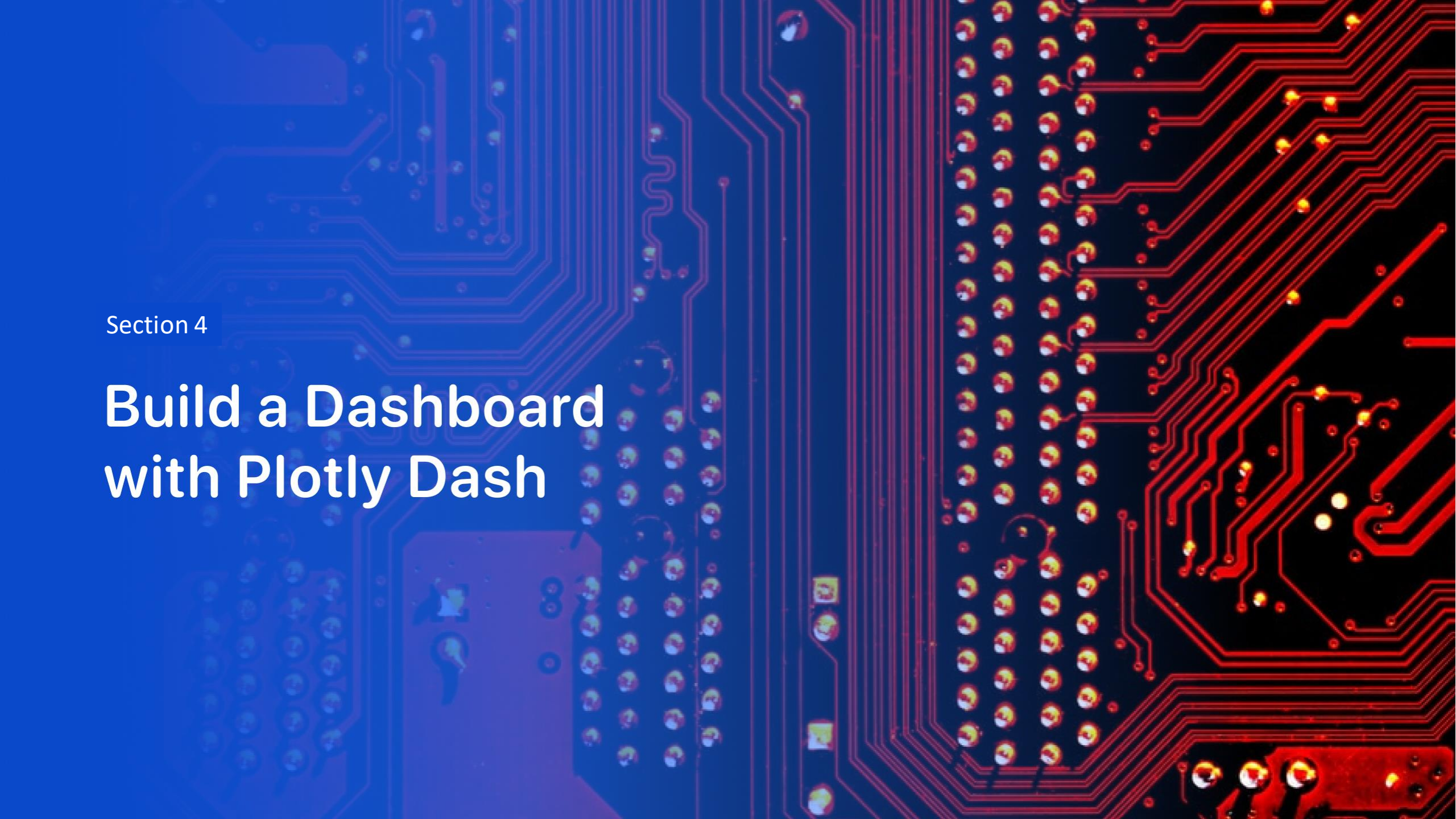
Show failed launches

Distance To Proximities

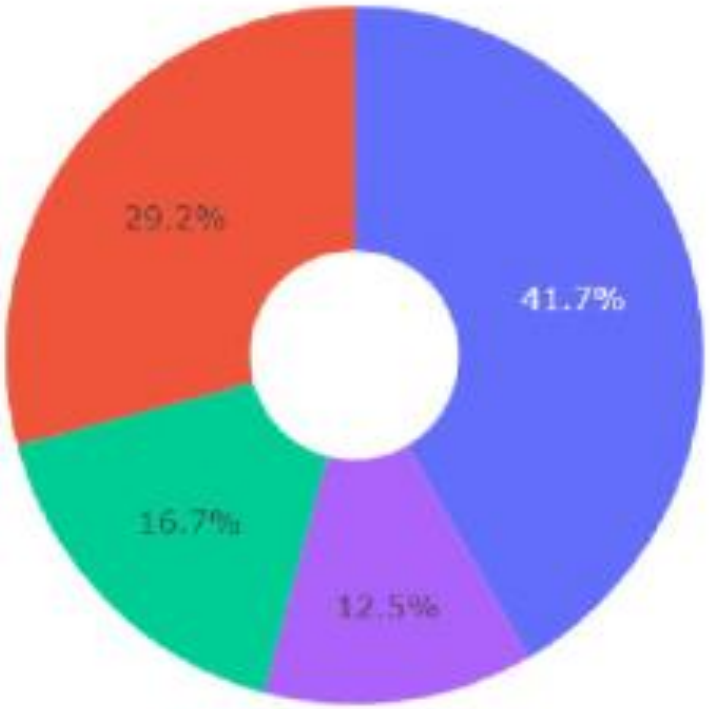


Distance to proximities can be measured to determine:

1. Are launch sites in close to proximity to railways? No.
2. Are launch sites in close to proximity to highways? No.
3. Are launch sites in close to proximity to coastlines? No.


Section 4
Build a Dashboard with Plotly Dash

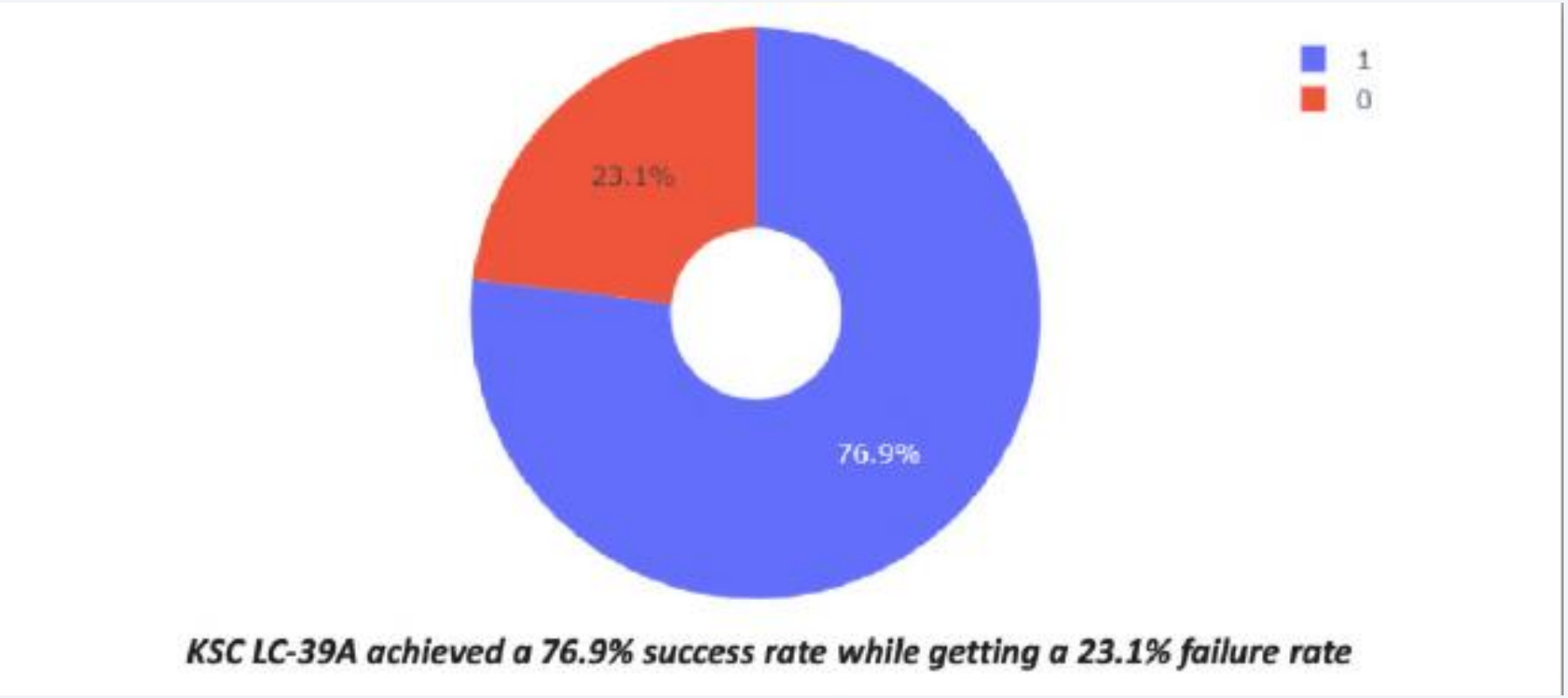
The Success Percentage by Sites



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

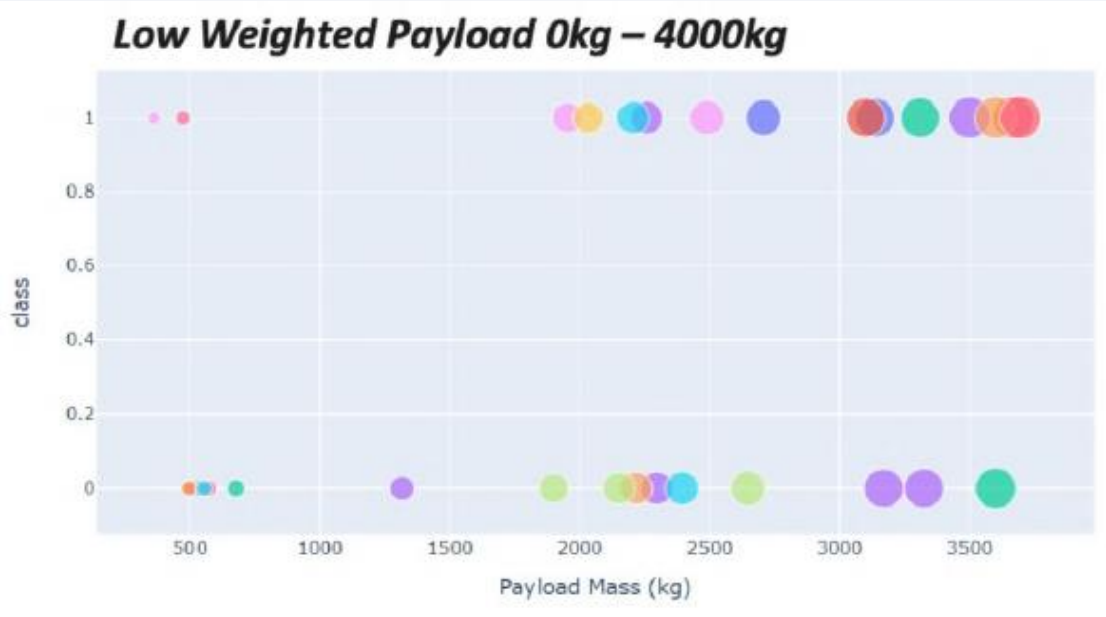
We can see that KSC LC-39A had the most successful launches from all the sites

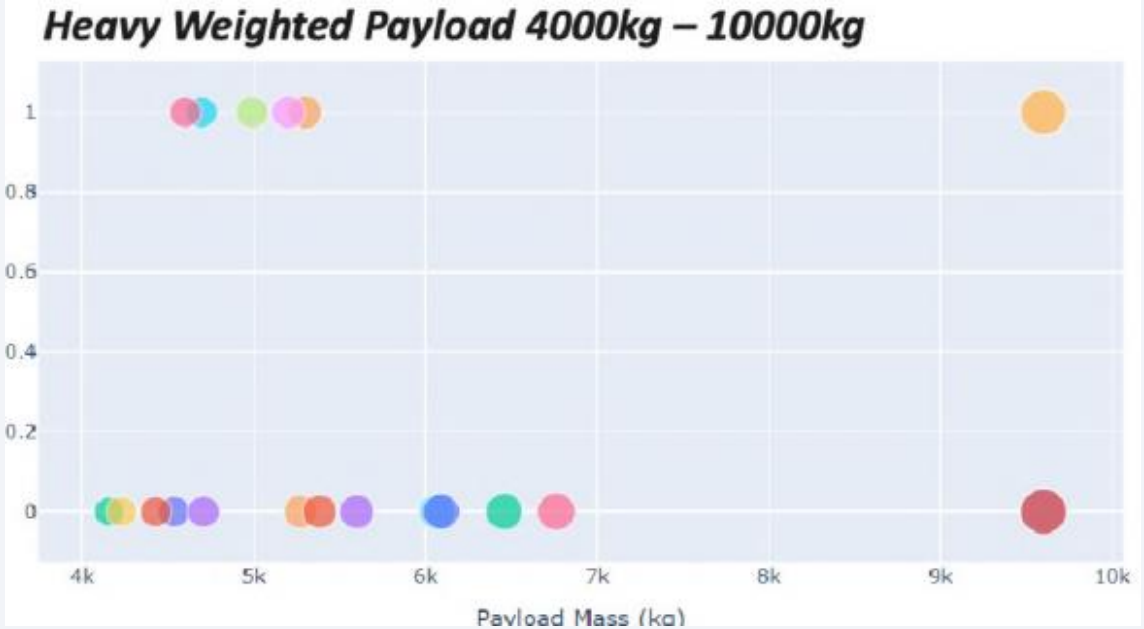
Highest Launch Success Ratio



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

<Dashboard Screenshot 3>



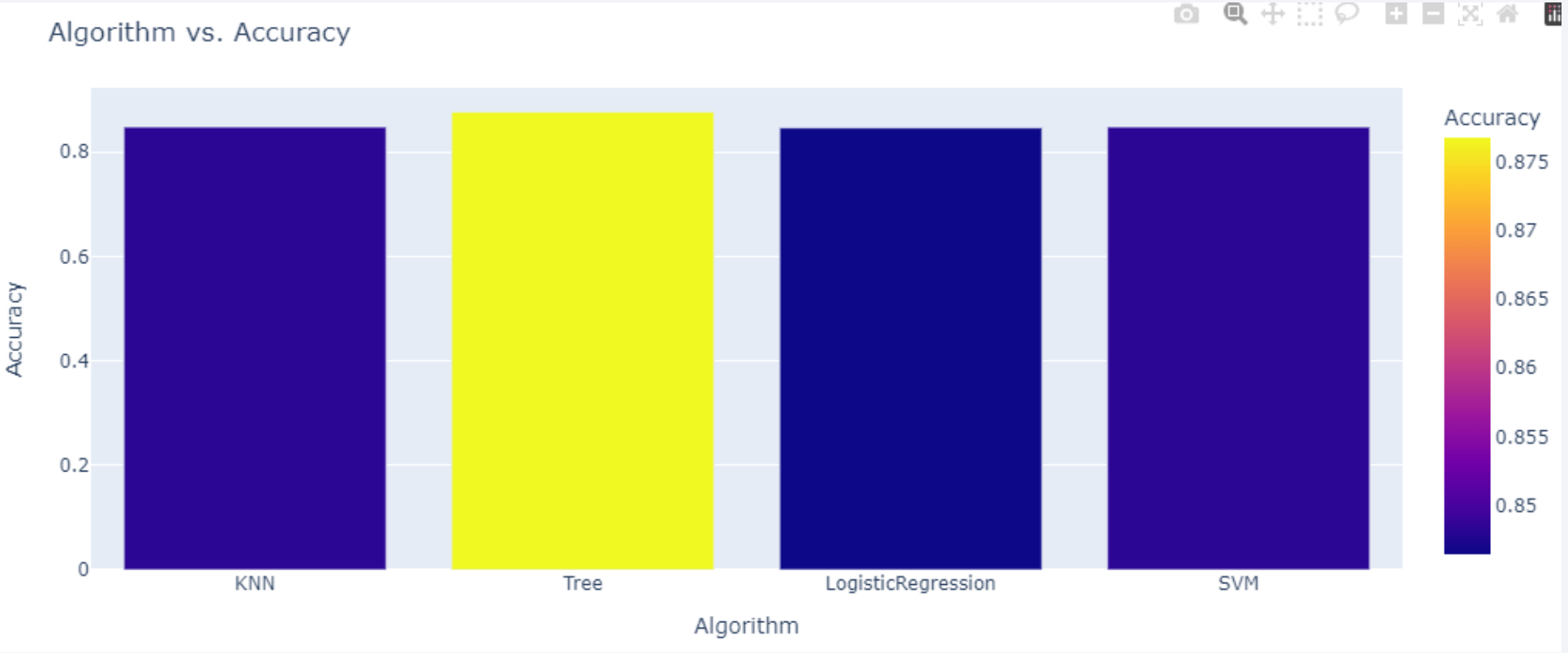


Section 5

Predictive Analysis (Classification)

Classification Accuracy

- All method are having almost the same performance but we can see from the bar chart, Decision tree has slightly higher perfomance compare to others



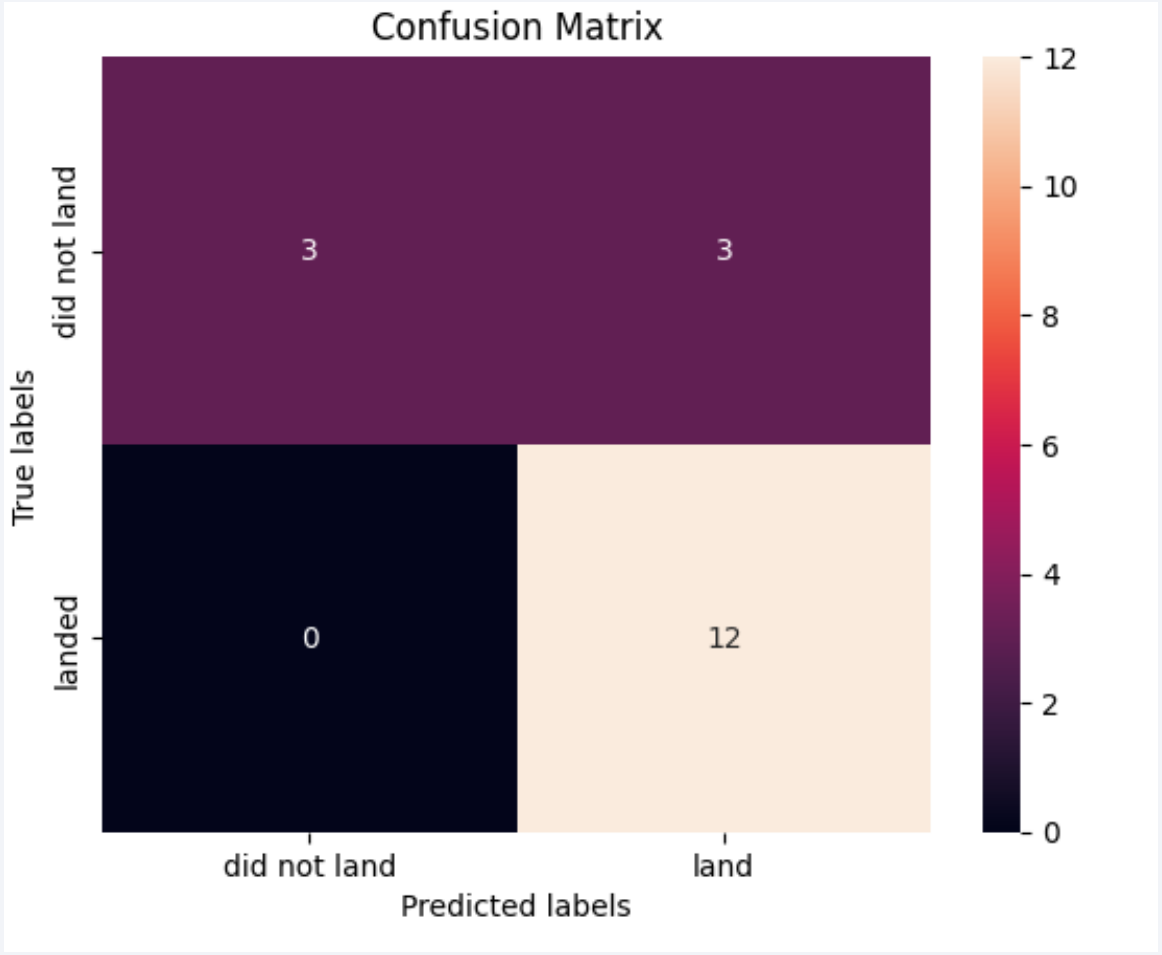
Classification Accuracy

- We can get the best score in numerical va

```
algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_, 'SVM': svm_cv
bestalgorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
if bestalgorithm == 'Tree':
    print('Best Params is :',tree_cv.best_params_)
if bestalgorithm == 'KNN':
    print('Best Params is :',knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best Params is :',logreg_cv.best_params_)
if bestalgorithm == 'SVM':
    print('Best Params is :',svm_cv.best_params_)
```

```
Best Algorithm is Tree with a score of 0.8767857142857143
Best Params is : {'criterion': 'gini', 'max_depth': 4, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 2,
'splitter': 'random'}
```

Confusion Matrix



The confusion matrix summarize performance of a classification algorithm. While in this case, all confusion matrices across all method are the same.

Conclusions

- ML Model: The models are having almost identical performance with Decision tree having slightly better scoring.
- Launch success rate is increasing every year, and the trend is expected to be continuing
- Orbits ES-L1, GEO, HEO, SSO has highest success rate

Thank you!