IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Manoj Bhole
14th February 2024

Outline

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

Executive Summary

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

Introduction

- Project background and context
  - Space X 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 Space X 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 space X for a rocket launch. This goal of the project is to create a machine learning pipeline to predict if the first stage will land successfully.

- Problems you want to find answers
  - What factors determine if the rocket will land successfully?
  - The interaction amongst various features that determine the success rate of a successful landing.
  - What operating conditions needs to be in place to ensure a successful landing program.

Section 1

Methodology

Methodology

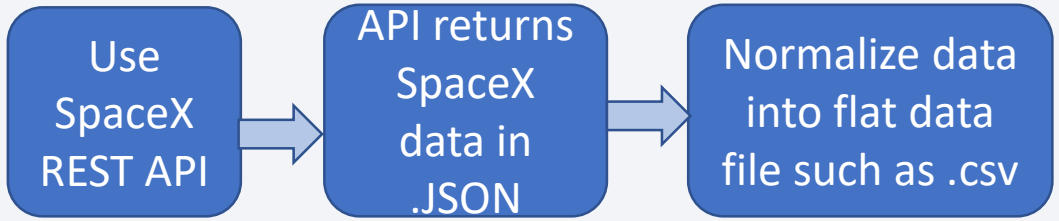
Executive Summary

- Data collection methodology:
  - Data was collected using SpaceX API and web scraping from Wikipedia.
- Perform data wrangling
  - One-hot encoding was applied to 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
  - Break dataset into training and test data
  - Develop various models (KNN, Decision Tree, etc…) using train data
  - Assess accuracy scores of models and their best parameters using test data

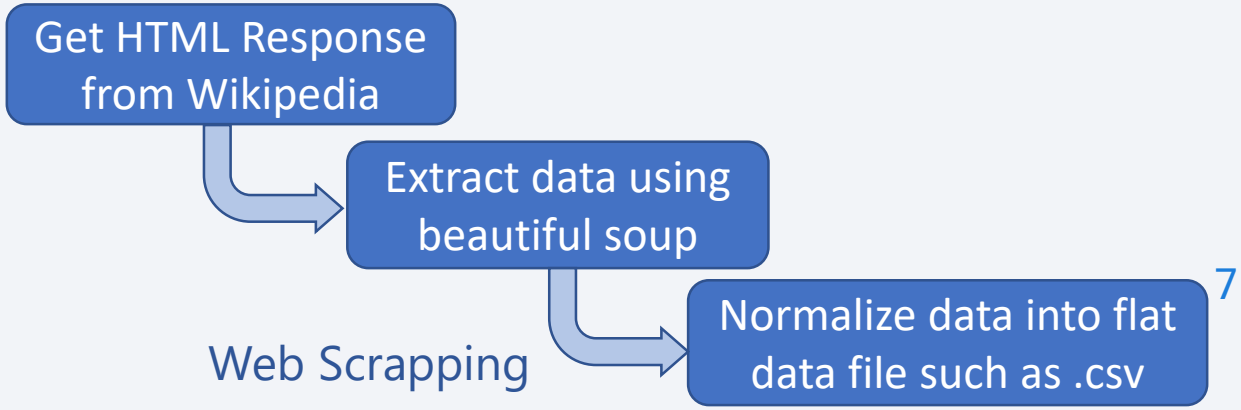
Data Collection

The following datasets 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



Web Scrapping

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

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

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}
```

```
# Create a data from launch_dict
df = pd.DataFrame(launch_dict)
```

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

```
data_falcon9 = df[df['BoosterVersion']!='Falcon 1']
```

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

[GitHub URL of the SpaceX API calls]

Data Collection - Scraping

1 .Getting Response from HTML

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

2. Creating BeautifulSoup Object

```
soup = BeautifulSoup(response,"html.parser")
```

3. Finding tables

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

4. Getting column names

```
column_names = []

headers = first_launch_table.find_all('th')

for th in headers:
    name = th.contents[0]
    if name is not None and len(name) > 0:
        column_names.append(th.getText())
```

5. Creation of dictionary

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date andtime (UTC)\n']

# Let's initial the launch_dict with each value
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',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
```

7. Converting dictionary to dataframe

```
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
```

8. Dataframe to .CSV

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

GitHub URL of the Web Scrapping calls

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 landed0means it was unsuccessful.

Perform Exploratory Data Analysis EDA on dataset

Calculate the number of launches at each site

Calculate the number and occurrence of each orbit

Calculate the number and occurrence of mission outcome per orbit type

Create a landing outcome label from Outcome column

Work out success rate for every landing in dataset

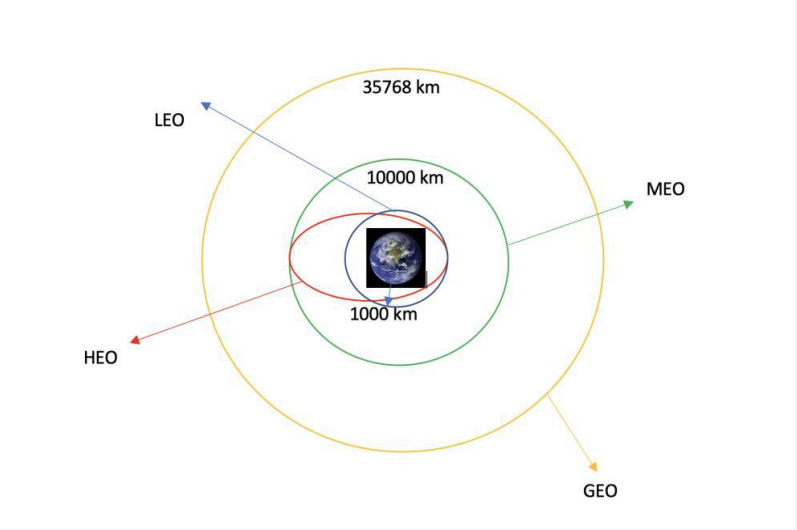


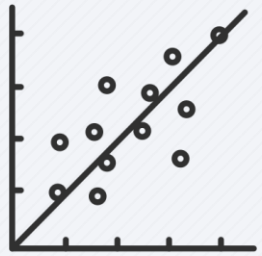
Diagram showing common orbit types SpaceX uses

GitHub URL of the Data Wrangling

EDA with Data Visualization

Scatter Graphs being drawn:

- 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



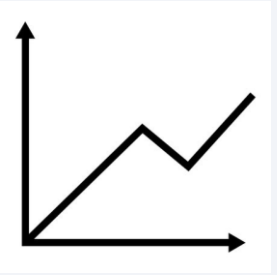
Scatter plots show how much one variable is affected by another. The relationship between two variables is called their correlation .Scatter plots usually consist of a large body of data.

Bar Graph being drawn:

- Mean Vs Orbit

A bar diagram 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.

Line Graph being drawn:



- Success Rate Vs Year

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

GitHub URL of the Data Visualization

EDA with SQL

Performed SQL queries to gather information about the dataset.

For example of some 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 URL of the EDA with SQL

Build an Interactive Map with Folium

Using a Folium map:

- We marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- We assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.
- Using the color-labeled with Green and Red markers clusters, we identified which launch sites have relatively high success rate.
- We calculated the distances between a launch site to its proximities using Haversine's formula.
- Example of some trends in which the Launch Site is situated in.
  - 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

GitHub URL of the Interactive Map

Build a Dashboard with Plotly Dash

The dashboard is built with Flask and Dash web framework.

Graphs

- **Scatter Graph showing the relationship with Outcome and Payload Mass (Kg) for the different Booster Versions**
  - It shows the relationship between two variables.
  - It is the best method to show you a non-linear pattern.
  - The range of data flow, i.e. maximum and minimum value, can be determined.
  - Observation and reading are straightforward.

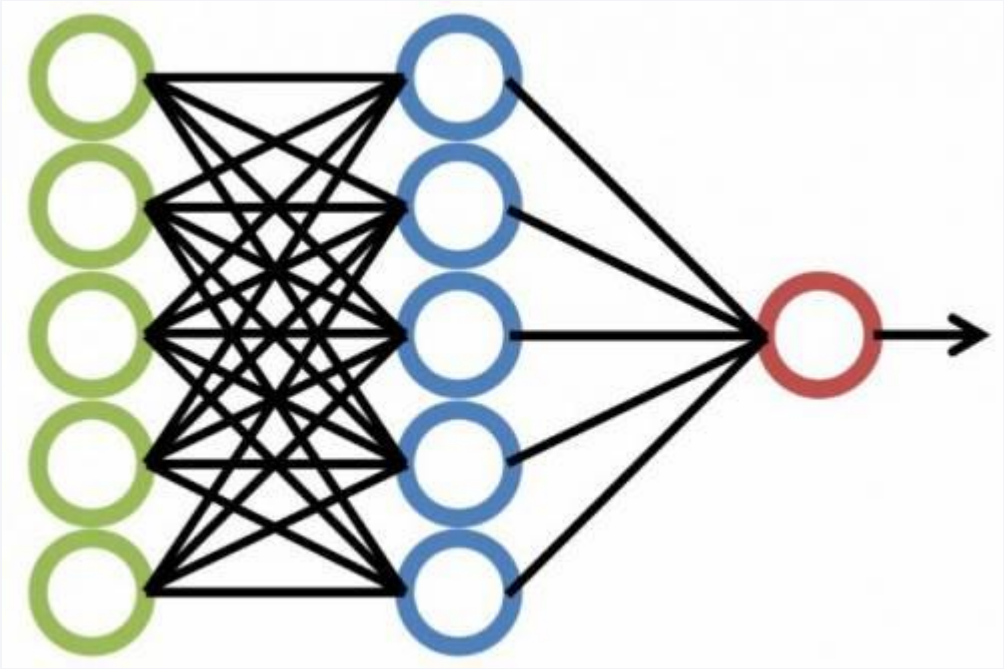
- **Pie Chart showing the total launches by a certain site/all sites**
  - *display relative proportions of multiple classes of data.*
  - *size of the circle can be made proportional to the total quantity it represents.*

GitHub URL of the Dashboard Code

Predictive Analysis (Classification)

➢ BUILDING MODEL

- Load 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 GridSearchCVobjects 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

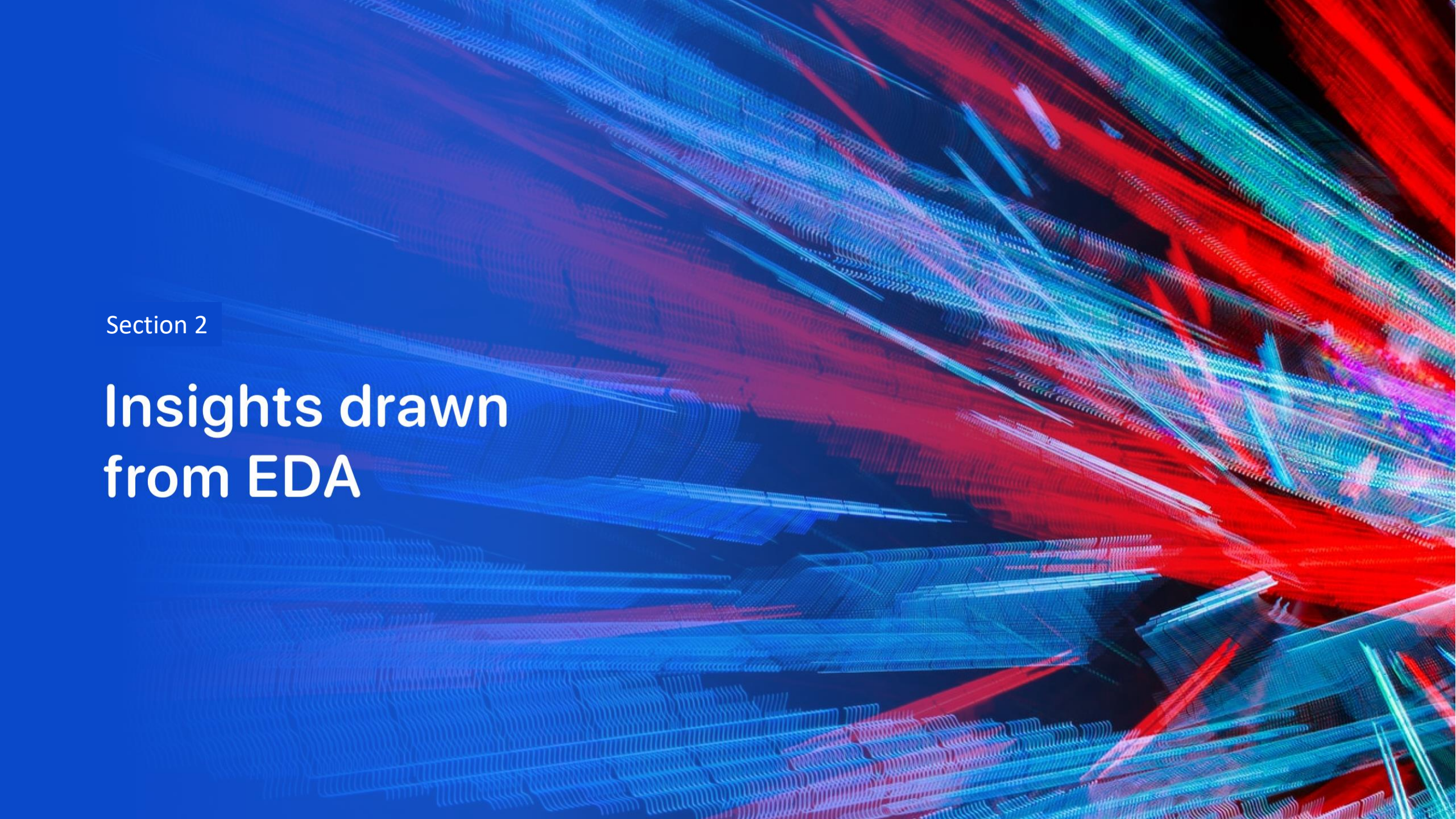
➢ FINDING THE BEST PERFORMING CLASSIFICATION MODEL

- 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.

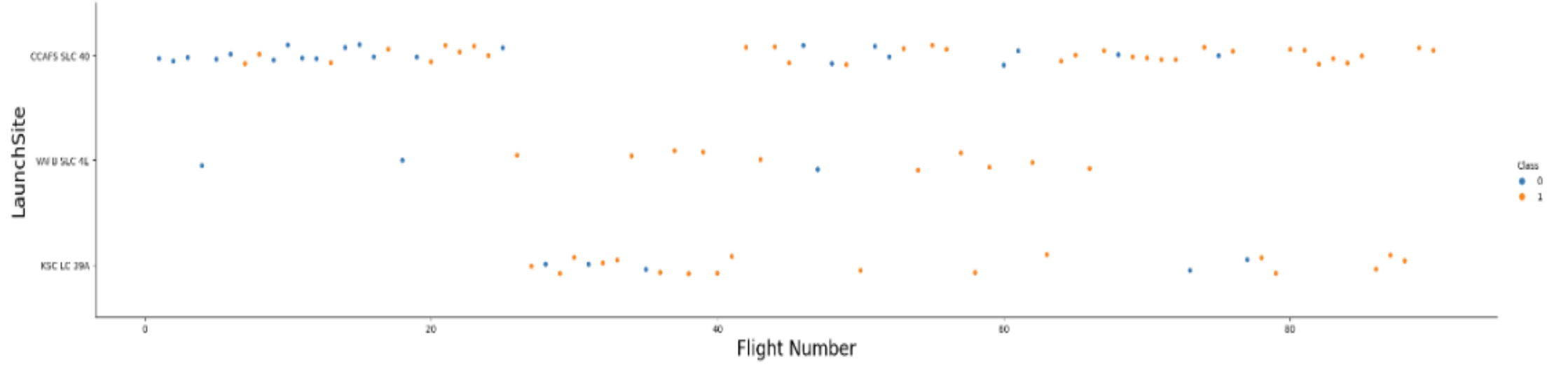
GitHub URL of the Predictive Analysis Code

Results

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

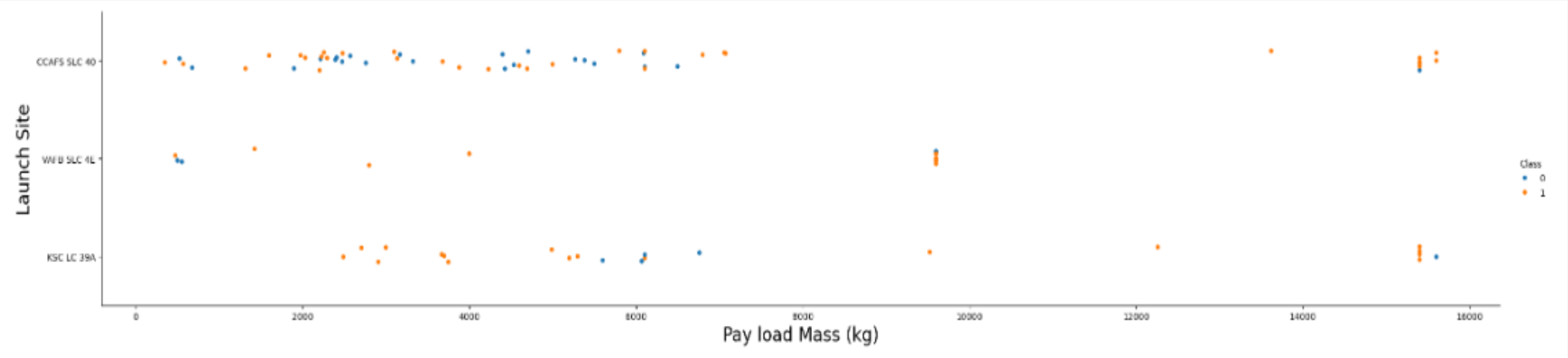

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



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

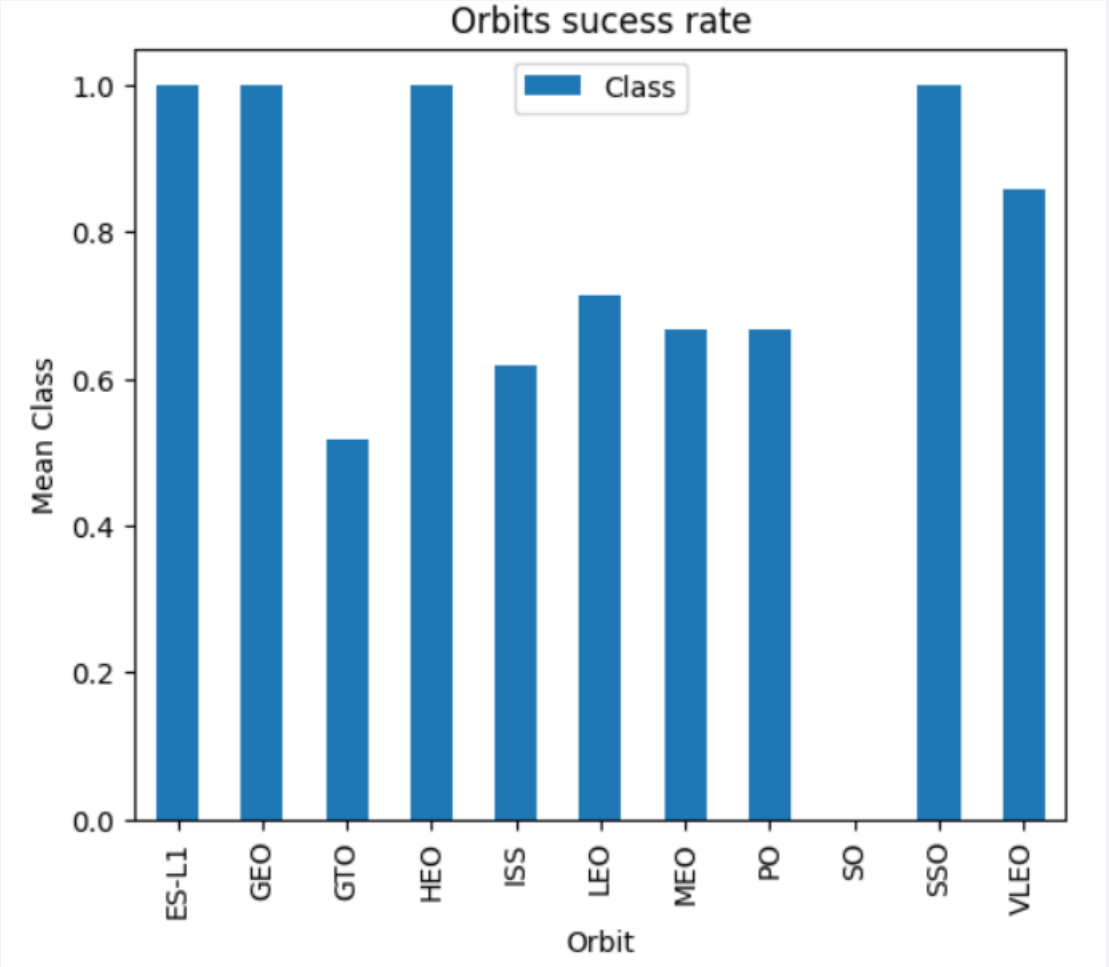
Payload vs. Launch Site



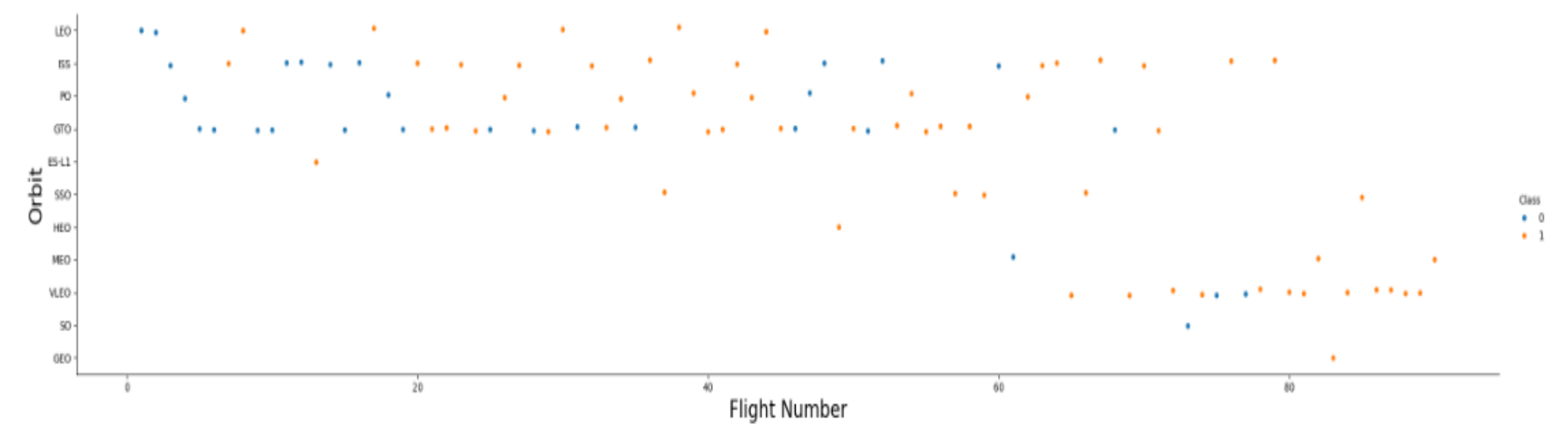
- The greater the payload mass for Launch Site CCAFS SLC 40 the higher the success rate for the Rocket.
- There is not quite a clear pattern to be found using this visualization to make a decision if the Launch Site is dependent on Pay Load Mass for a success launch.

Success Rate vs. Orbit Type

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

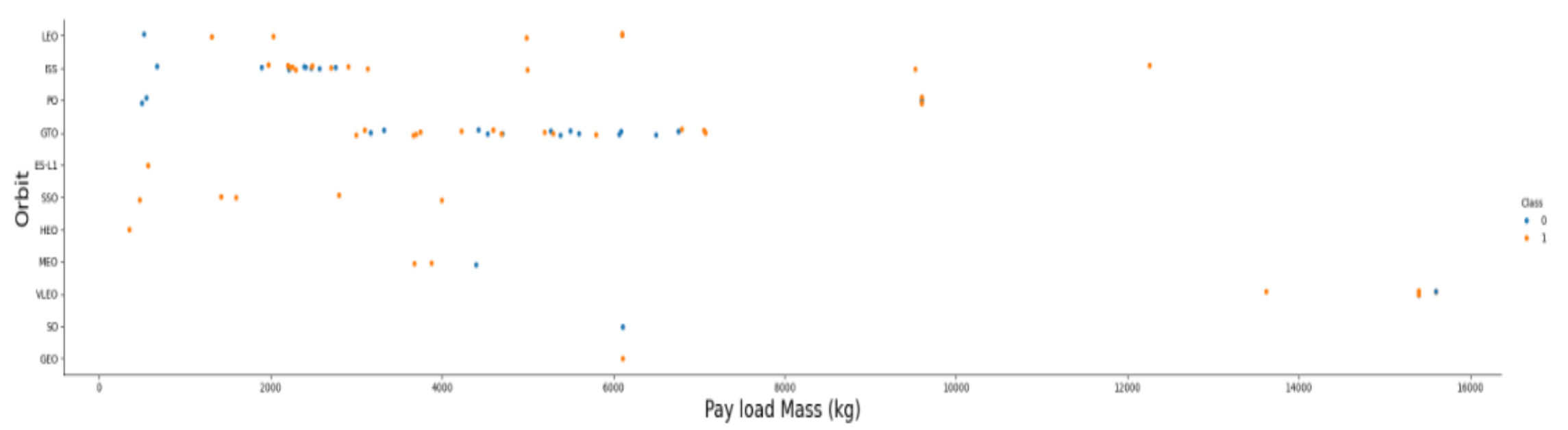


Flight Number vs. Orbit Type



- 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.

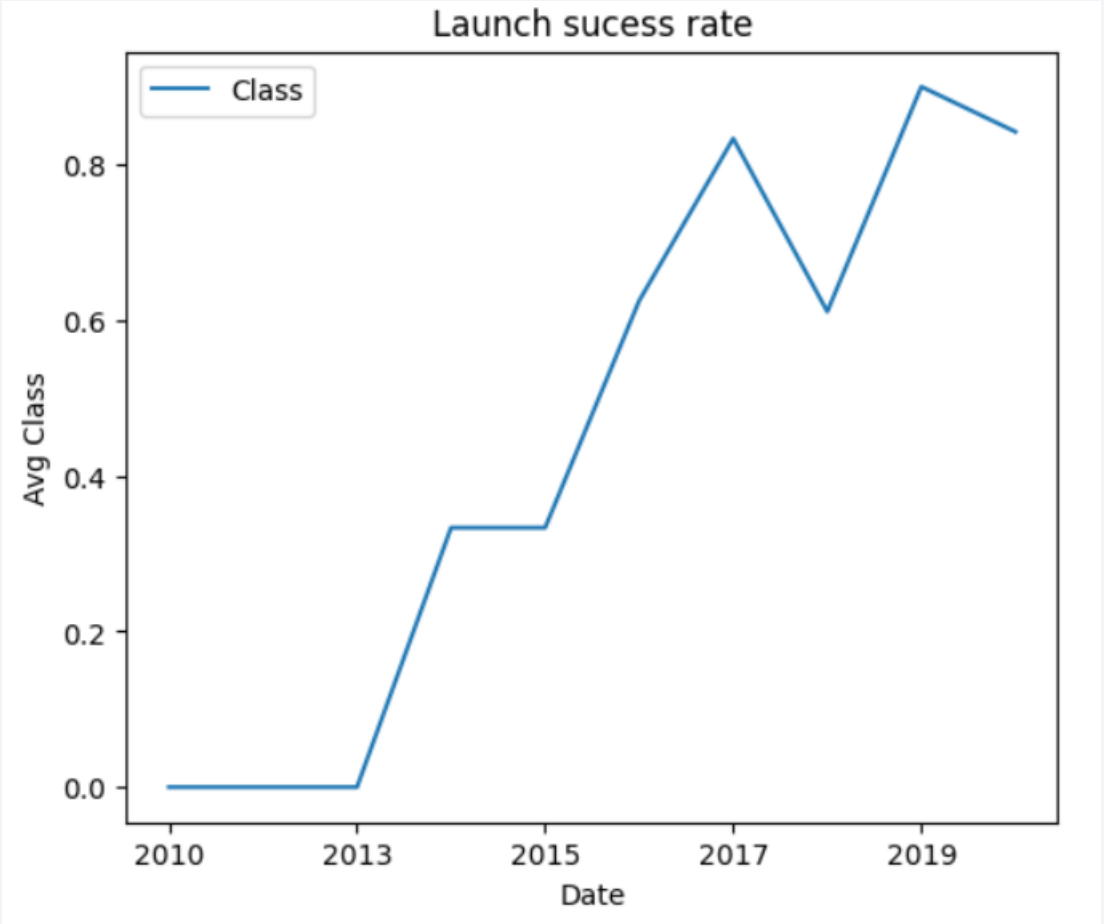
Payload vs. Orbit Type



- 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



All Launch Site Names

SQL Query

select DISTINCT Launch_Site from SPACEXTBL

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

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

Launch Site Names Begin with 'CCA'

SQL Query

```
select * from SPACEXTBL WHERE Launch_Site LIKE
'CCA%' LIMIT 5
```

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	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

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

Total Payload Mass Carried By Boosters From NASA

SQL Query

```
select SUM(PAYLOAD_MASS__KG_) from SPACEXTBL
WHERE Customer = 'NASA (CRS)'
```

SUM(PAYLOAD_MASS__KG_)
45596

- **QUERY EXPLAINATION**
  - 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_) from SPACEXTBL
WHERE Booster_Version = 'F9 v1.1'
```

AVG(PAYLOAD_MASS__KG_)
2928.4

- **QUERY EXPLAINATION**
  - 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) FirstSuccessfulLandingDate from SPACEXTBL
WHERE Landing_Outcome = 'Success (drone ship)'
```

FirstSuccessfulLandingDate
2016-04-08

- **QUERY EXPLAINATION**
  - 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**

Successful Drone Ship Landing with Payload between 4000 and 6000

SQL Query

```
select Booster_Version from SPACEXTBL WHERE PAYLOAD_MASS__KG_ Between 4000 and 6000
AND Landing_Outcome = 'Success (drone ship)'
```

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

- **QUERY EXPLAINATION**
  - 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**

Total Number of Successful and Failure Mission Outcomes

SQL Query

```
SELECT (SELECT COUNT(Mission_Outcome) from SPACEXTBL where Mission_Outcome LIKE
'%Success%') as Successful_Mission_Outcomes, (SELECT COUNT(Mission_Outcome) from
SPACEXTBL where Mission_Outcome LIKE '%Failure%') as Failure_Mission_Coutcomes
```

Successful_Mission_Outcomes	Failure_Mission_Coutcomes
100	1

- **QUERY EXPLAINATION**
- 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

Boosters Carried Maximum Payload

SQL Query

```
select Booster_Version from SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (select
MAX(PAYLOAD_MASS__KG_) from SPACEXTBL)
```

Booster_Version
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

➢ **QUERY EXPLAINATION**

- We used subquery here to get maximum payload and compared it with whole data to get boosters

2015 Launch Records

SQL Query

```
select substr(Date, 6,2) as month, Booster_Version, Launch_Site from SPACEXTBL WHERE
substr(Date,0,5)='2015' AND landing_outcome like 'failure%'
```

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

- **QUERY EXPLAINATION**
- We have Date field in database stored as VARCHAR.
- We extracted month and year from Date field using substring function in sql.

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

SQL Query

```
select landing_outcome,COUNT(*) as Count_landing_outcomes from SPACEXTBL WHERE Date between '2010-06-04' and '2017-03-20' GROUP BY landing_outcome order by Count_landing_outcomes DESC
```

Landing_Outcome	Count_landing_outcomes
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

- **QUERY EXPLAINATION**
- We have filtered data based on Date field and then taken count of each landing outcome and the count arranged in descending order.

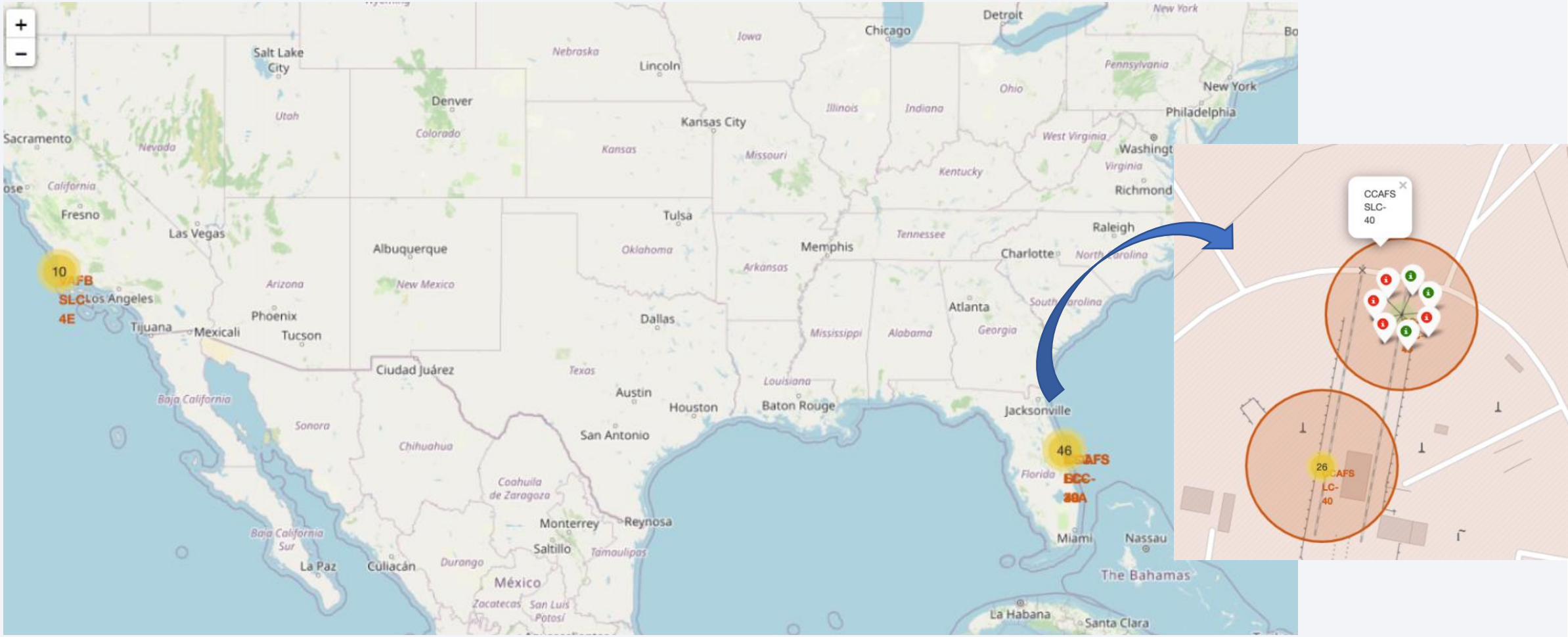
Section 3

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

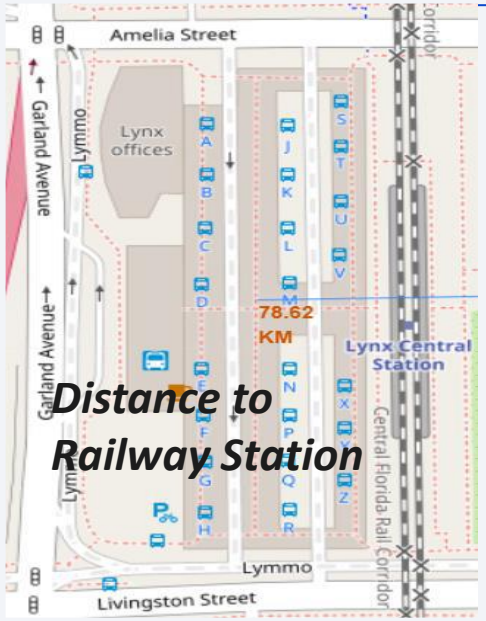
Color Labelled Markers

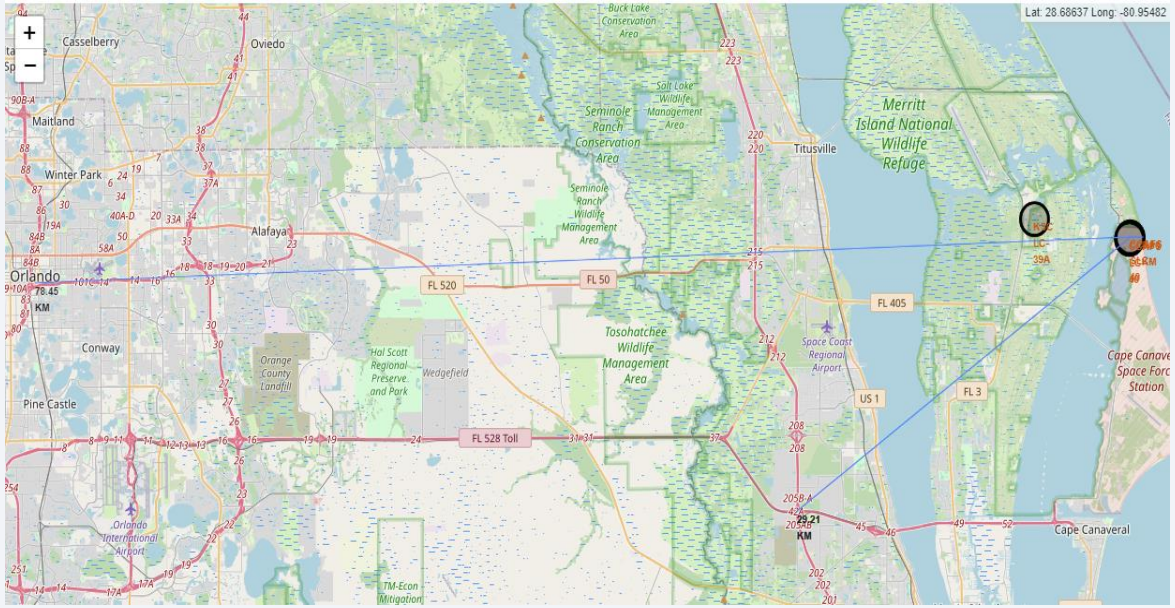


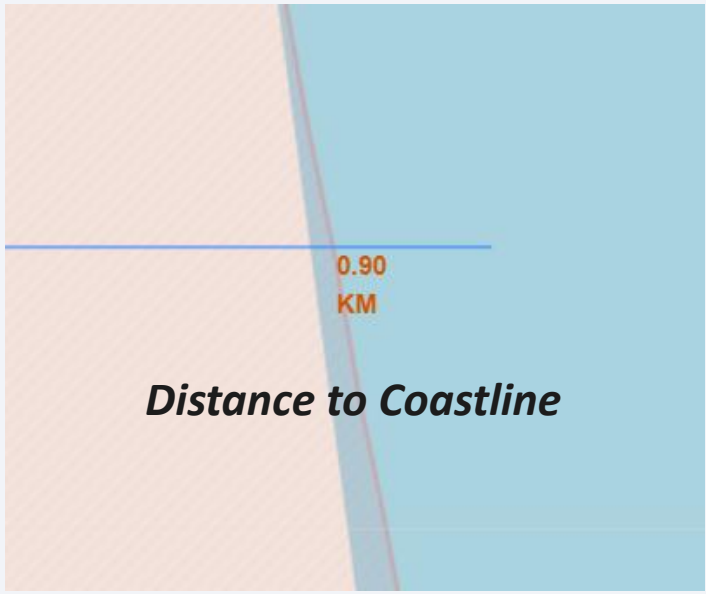
Florida Launch Sites

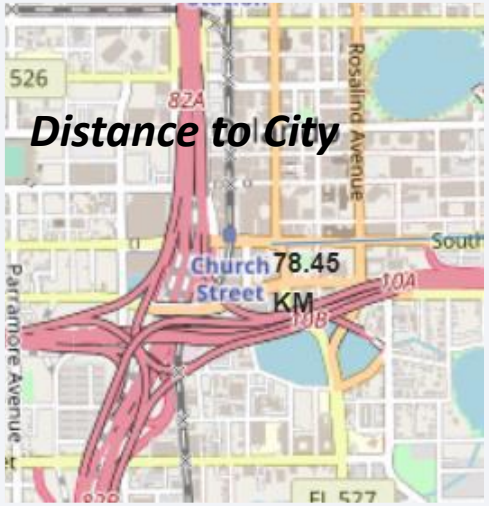
Green Marker shows successful Launches and Red Marker shows Failures

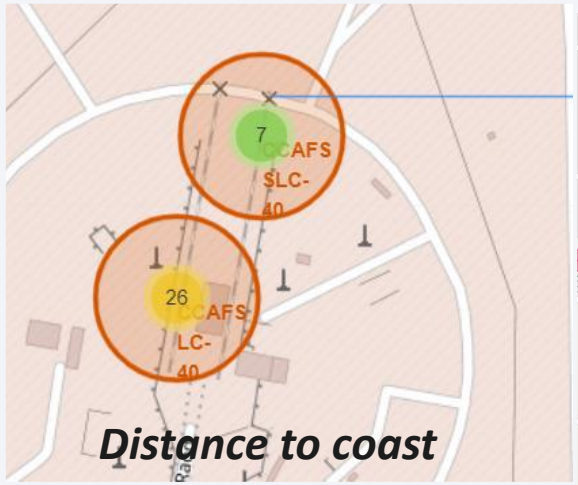
Working out Launch Sites distance to landmarks to find trends with Haversine formula using CCAFS-SLC-40 as a reference



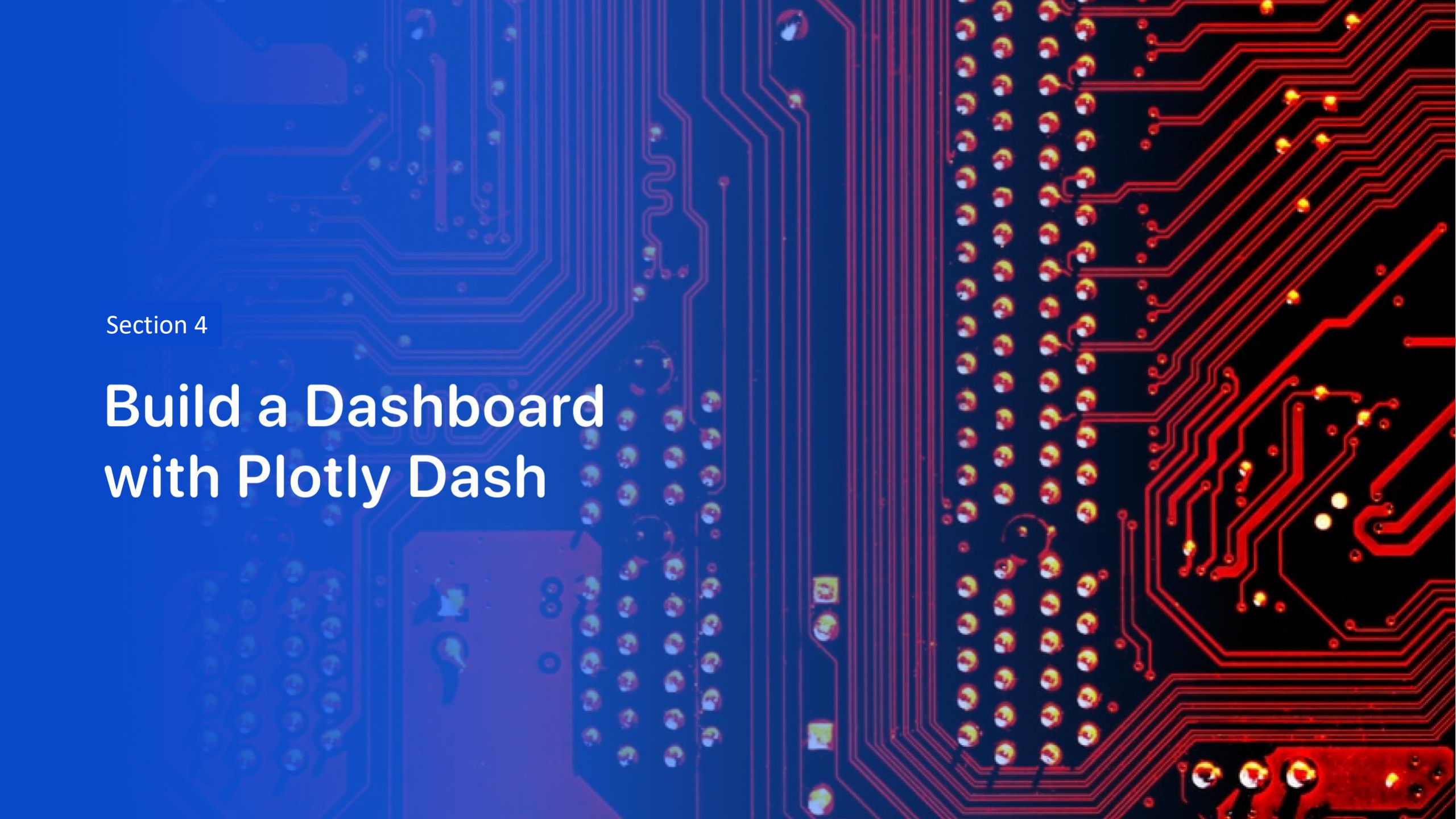




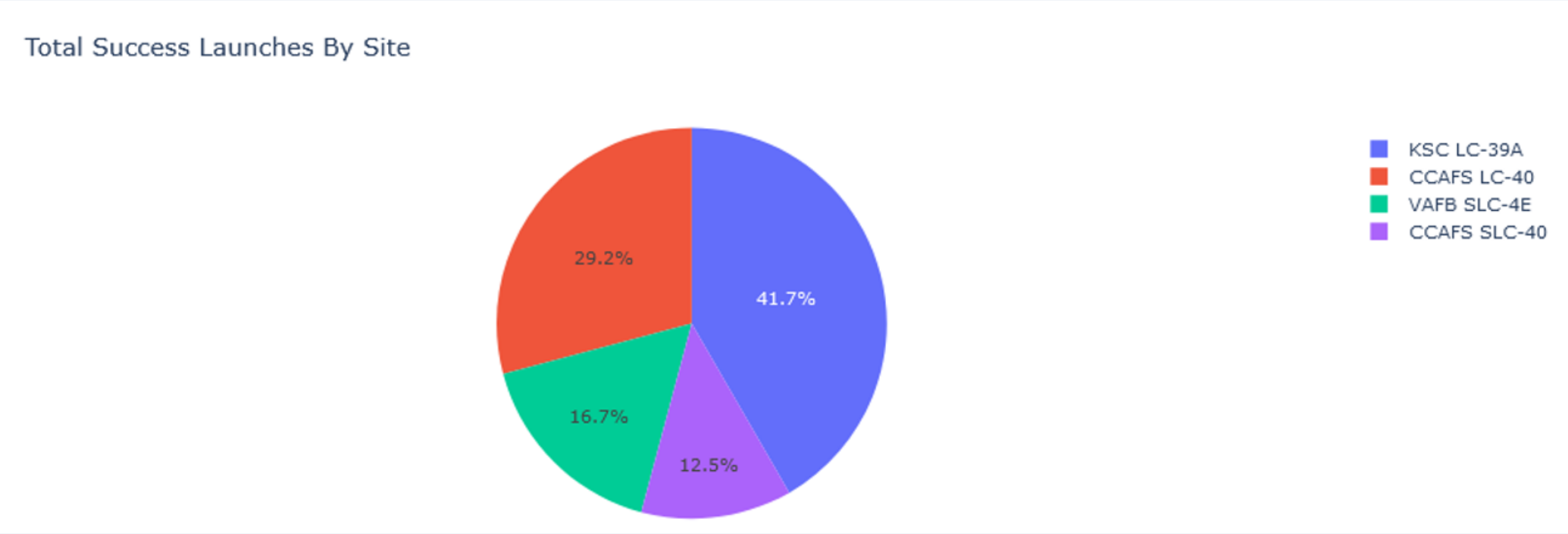




- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

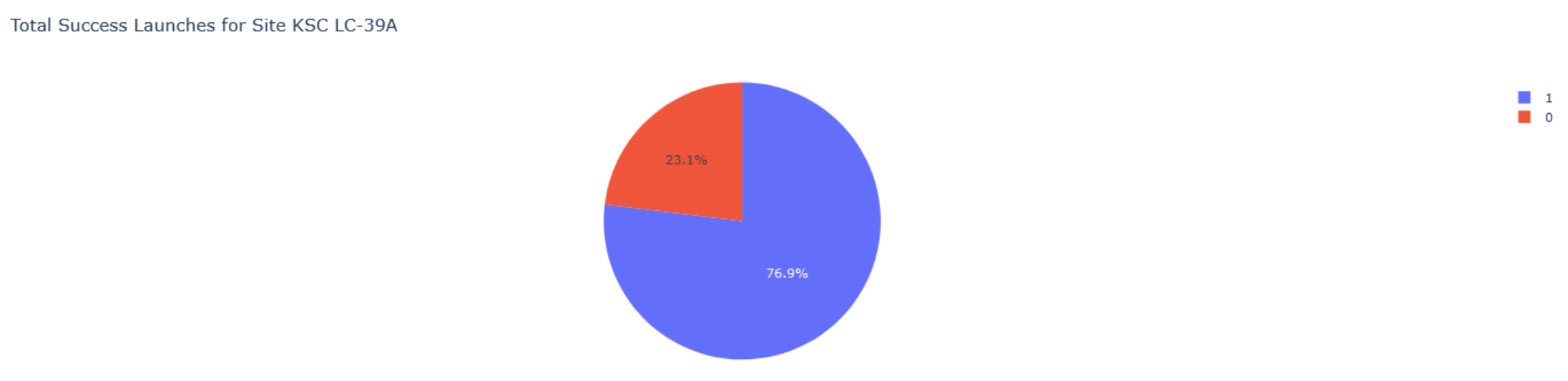

Section 4
Build a Dashboard with Plotly Dash

DASHBOARD – Pie chart showing the success percentage achieved by each launch site



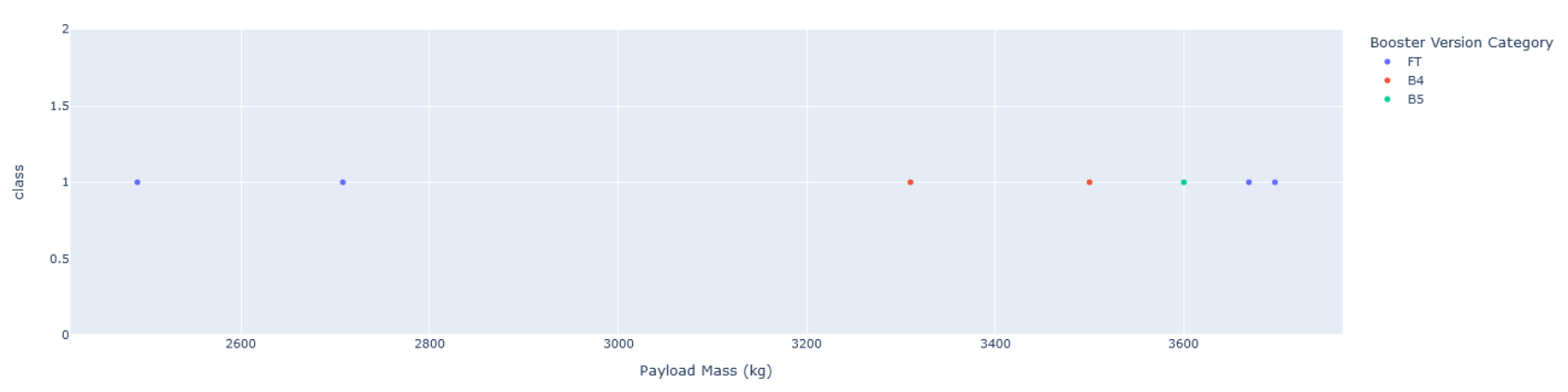
- We can see that KSC LC-39A had the most successful launches from all the sites

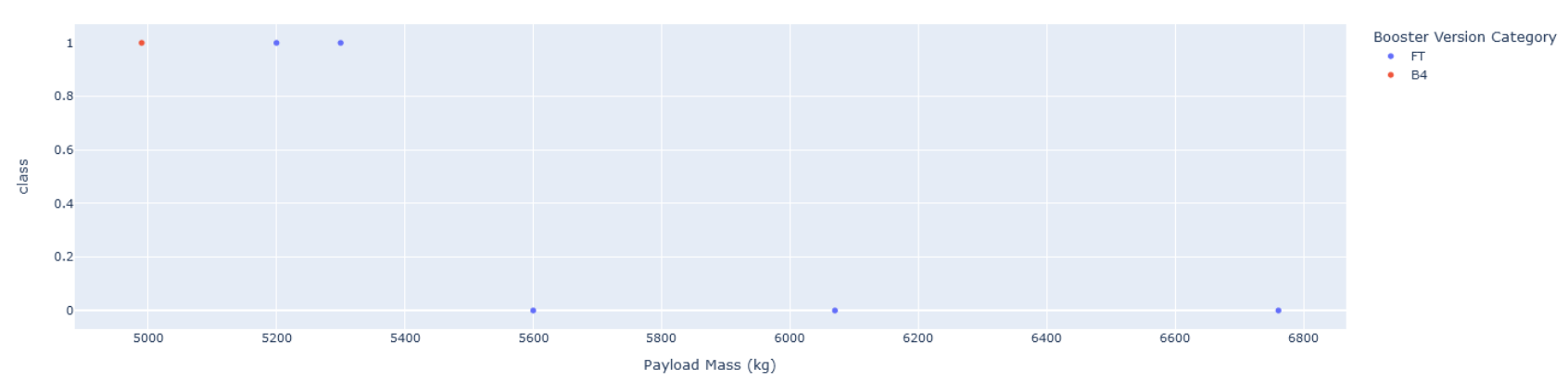
DASHBOARD – Pie chart for the launch site with highest launch success ratio



- 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





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

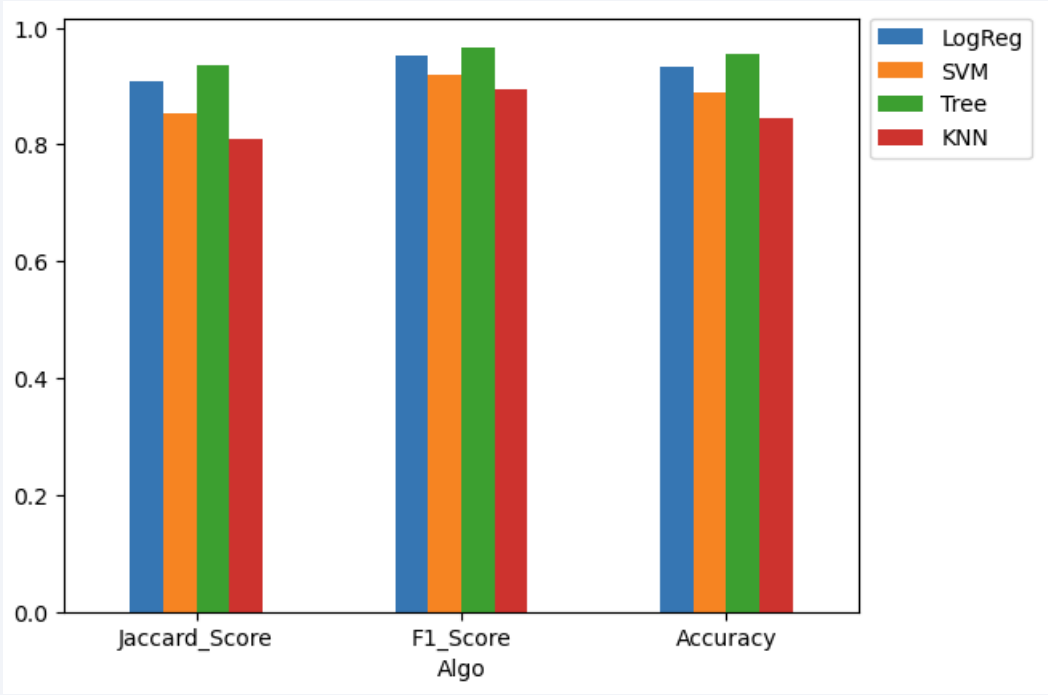
Section 5

Predictive Analysis (Classification)

Classification Accuracy

- As you can see our accuracy is extremely close but we do have a winner its down to decimal places! using this function

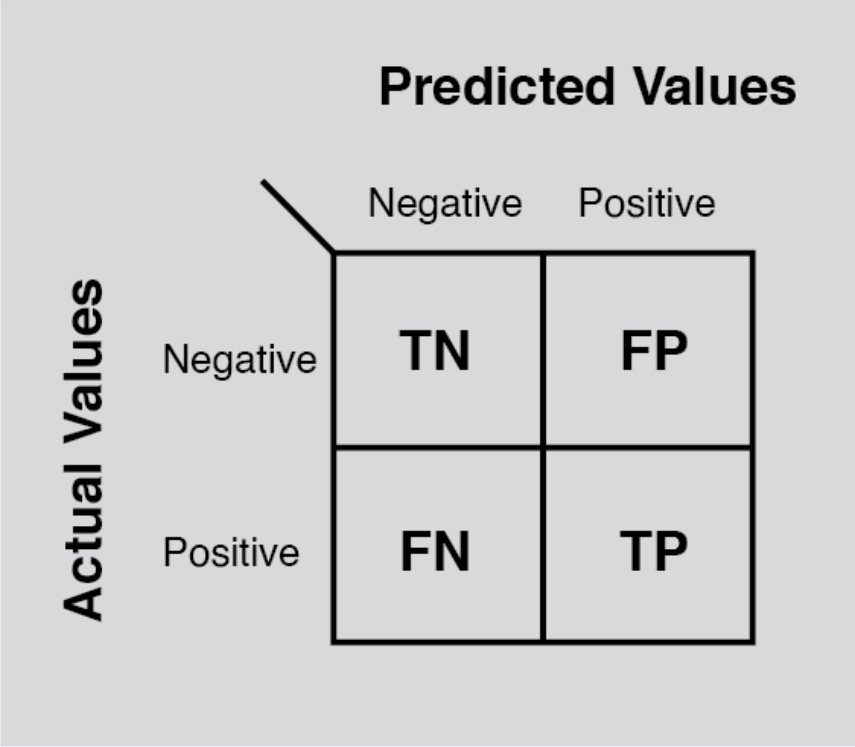
	LogReg	SVM	Tree	KNN
Jaccard_Score	0.907692	0.852941	0.936508	0.808219
F1_Score	0.951613	0.920635	0.967213	0.893939
Accuracy	0.933333	0.888889	0.955556	0.844444

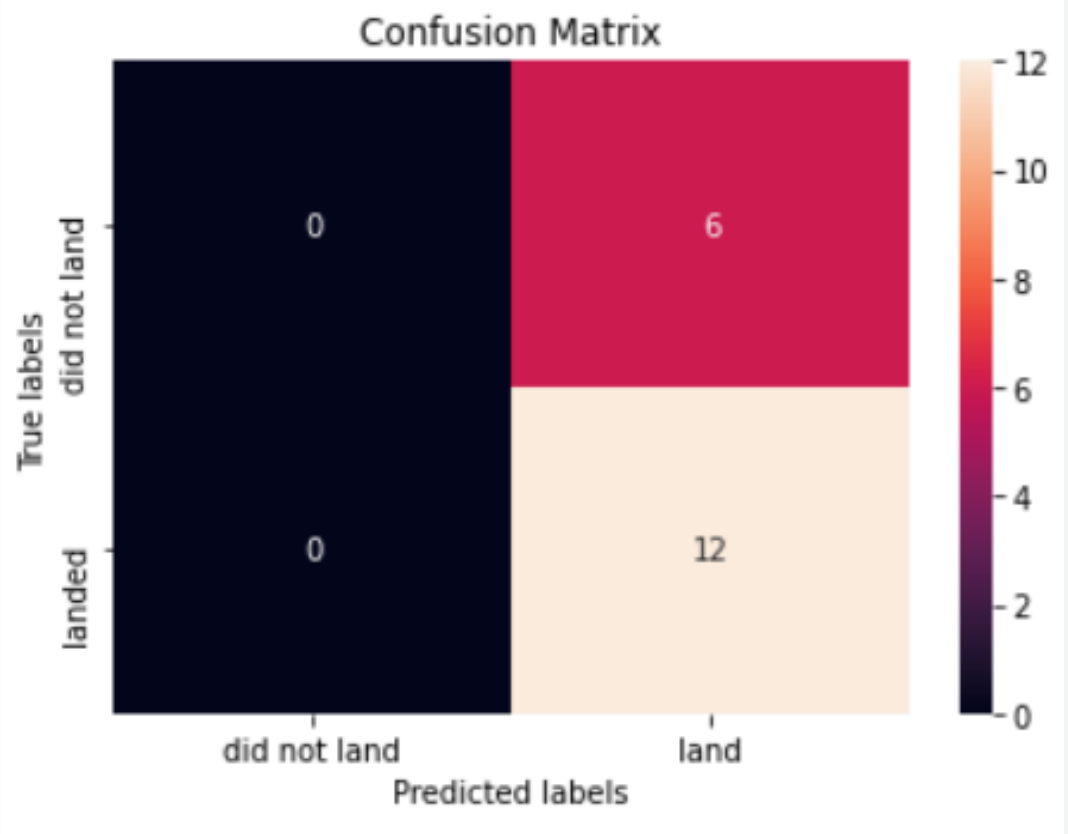


- The tree algorithm wins!!
- After selecting the best hyper parameters for the decision tree classifier using the validation data, we achieved 83.33% accuracy on the test data.

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

➢ Haversine formula

Introduction

The haversine formula determines the great-circle distance between two points on a sphere given their longitudes and latitudes. Important in navigation, it is a special case of a more general formula in spherical trigonometry, the law of haversines, that relates the sides and angles of spherical triangles.

Usage

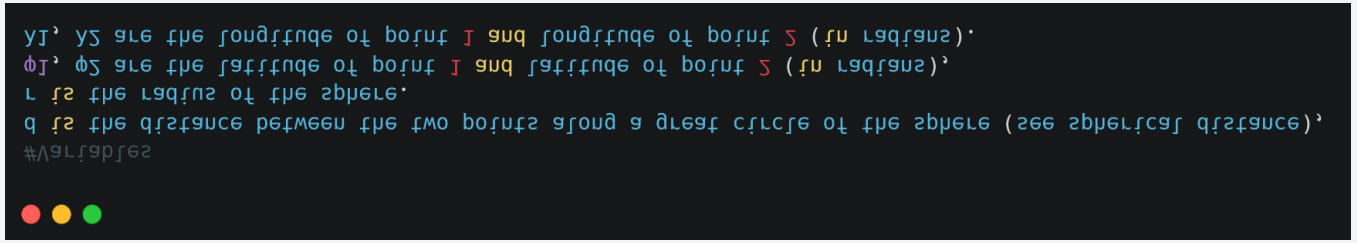
Why did I use this formula? First of all, I believe the Earth is round/elliptical. I am not a Flat Earth Believer! Jokes aside when doing Google research for integrating my ADGGoogleMaps API with a Python function to calculate the distance using two distinct sets of {longitudinal, latitudinal} list sets. Haversine was the trigonometric solution to solve my requirements above.

Formula

$$a = \sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos\varphi 1 \cdot \cos\varphi 2 \cdot \sin^2\left(\frac{\Delta\lambda}{2}\right)$$

$$c = 2 \cdot \operatorname{atan2}\left(\sqrt{a}, \sqrt{(1-a)}\right)$$

$$d = R \cdot c$$



Thank you!