IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Mayank Nagar
17 April 2022

Outline

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

Executive Summary

- Summary of methodologies
  - Data collection using SpaceX API
  - Data collection with web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - Complete EDA with Visualization
  - Interactive Visual Analytics with Folium and Dashboard using Ploty Dash
  - Building Machine Learning models
- Summary of all results
  - Exploratory Data Analysis result
  - Interactive analytic screenshots
  - Machine learning predictions

Introduction

- Project background and context

  SpaceX says that launching a Falcon 9 rocket costs 62 million dollars. On the other hand, their competition costs 165 million dollars. The reason for this difference is that SpaceX can reuse the rocket in the first stage. Therefore we want to predict if a rocket will land on the first stage or not. This study aims to build a machine learning pipeline to predict the success of the first stage.

- Problems you want to find answers

  - Determine a new rocket launch will be successful or not
  - Correlation between different feature and relation between various independent feature and depended feature
  - What is requirement conditions for a successful mission

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using SpaceX API and Web Scrapping form Wikipedia
- Perform data wrangling
  - One Hot Encoding is applied to the numerical 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
  - How to build, tune, evaluate classification models

Data Collection

- Data Collection stage mainly comprises two methods
  - ➢ Data Collection using SpaceX API
    - requests.get method is used to get data from SpaceX API and converted into JSON file
    - After this, the data is converted into pandas DataFrame using pd.json_normalize() method
    - We cleaned the data and the missing value in payload mass column was filled by the mean of the payload mass values
  - ➢ Data Collection using Web Scrapping from Wikipedia
    - BeautifulSoup library is used to scrap data from Wikipedia about Falcon 9 launch.
    - Launch records were extracted as HTML tables and converted into pandas DataFrame

Data Collection – SpaceX API

- We used requests.get method to collect the data from SpaceX API. We cleaned the data and converted into pandas DataFrame.

- The GitHub URL of the completed SpaceX API calls notebook (https://github.com/nagar-mayank/Applied-Data-Science-Capstone-Project/blob/main/Notebooks/jupyter-labs-spacex-data-collection-api_--qgzCRIS.ipynb)

Task 1: Request and parse the SpaceX launch data using the GET request

To make the requested JSON results more consistent, we will use the following static response object for this project:

[35]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud\
/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
response = requests.get(static_json_url)
```

We should see that the request was successfull with the 200 status response code

[36]:
```
response.status_code
```

[36]:
```
200
```

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe

using `.json_normalize()`

[37]:
```
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

Using the dataframe `data` print the first 5 rows

[39]:
```
# Get the head of the dataframe
data.head(5)
```

...

You will notice that a lot of the data are IDs. For example the rocket column has no information

about the rocket just an identification number.

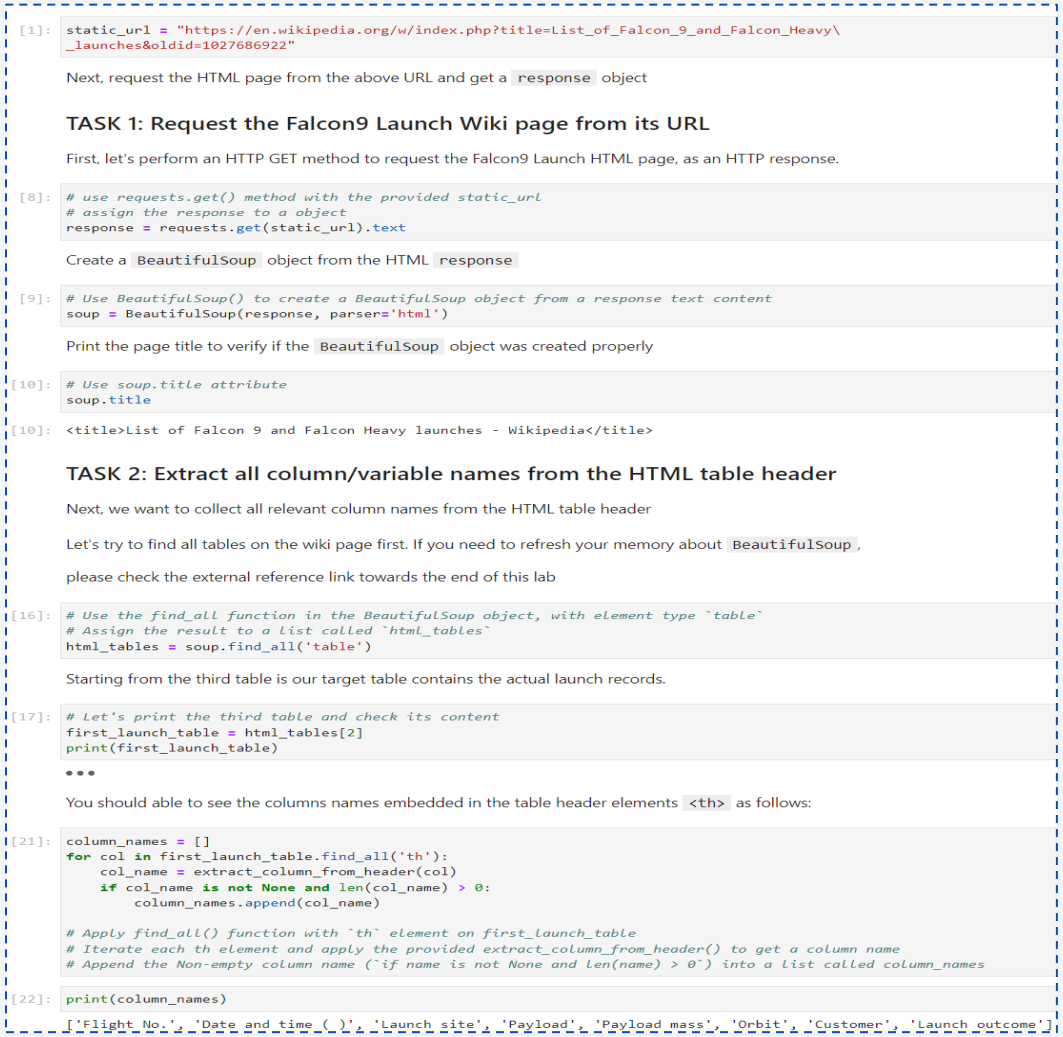
We will now use the API again to get information about the launches using the IDs given for each

launch. Specifically we will be using columns `rocket`, `payloads`, `launchpad`, and `cores`.

[34]:
```
# Lets take a subset of our dataframe keeping only the features we want and the flight number,
# and date_utc.
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 rocket
# boosters and rows that have multiple payloads in a single rocket.
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 the
# list and replace the feature.
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
# Leaving the time
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

- We used BeautifulSoup library to get the data from Wikipedia about Falcon 9 rockets
- We parsed the tables and converted into pandas DataFrame
- The GitHub URL of the completed BeautifulSoup notebook (https://github.com/nagar-mayank/Applied-Data-Science-Capstone-Project/blob/main/Notebooks/jupyter-labs-webscraping_0LxDYKsUW.ipynb)



[1]:
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy\
_launches&oldid=1027686922"
```

Next, request the HTML page from the above URL and get a `response` object

TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

[8]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url).text
```

Create a `BeautifulSoup` object from the HTML `response`

[9]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response, parser='html')
```

Print the page title to verify if the `BeautifulSoup` object was created properly

[10]:
```
# Use soup.title attribute
soup.title
```

[10]:
```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

TASK 2: Extract all column/variable names from the HTML table header

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`,

please check the external reference link towards the end of this lab

[16]:
```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
```

Starting from the third table is our target table contains the actual launch records.

[17]:
```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

...

You should able to see the columns names embedded in the table header elements `<th>` as follows:

[21]:
```
column_names = []
for col in first_launch_table.find_all('th'):
    col_name = extract_column_from_header(col)
    if col_name is not None and len(col_name) > 0:
        column_names.append(col_name)

# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names
```

[22]:
```
print(column_names)
```

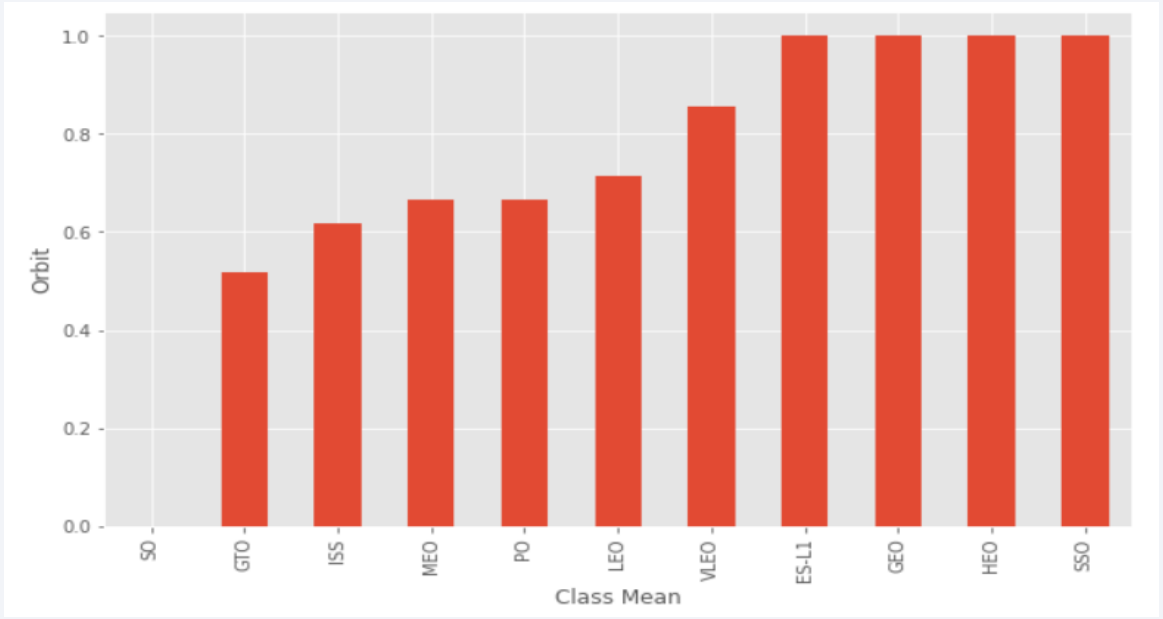
```
['Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome']
```

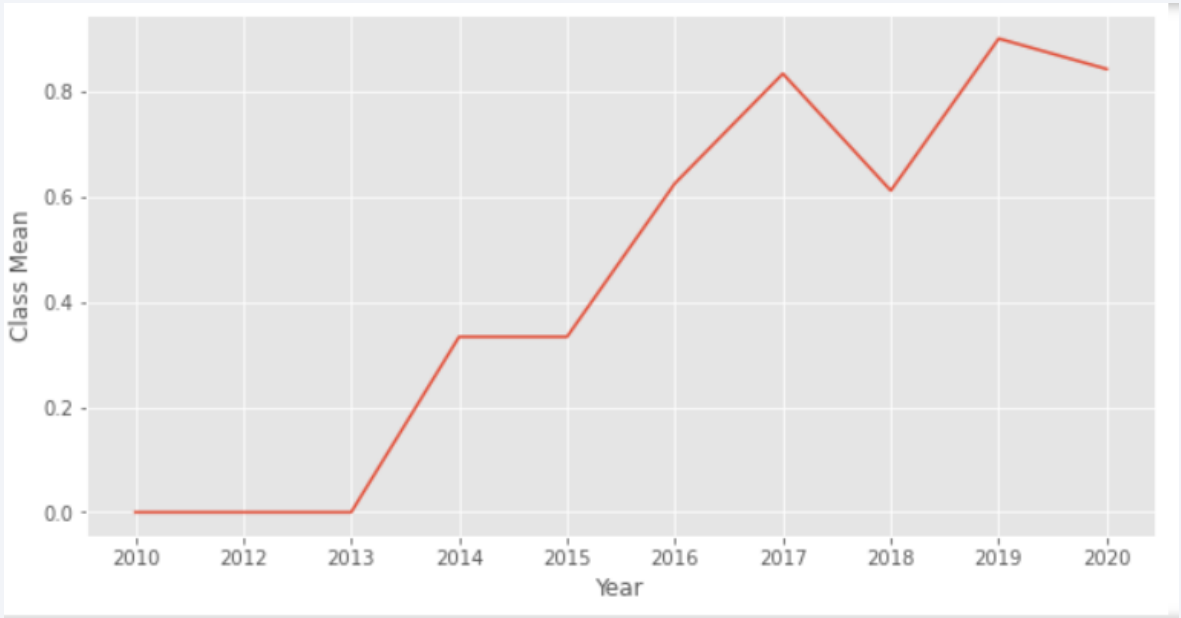
Data Wrangling

- We calculated the number of launches on each site and the occurrence of each orbit
- We calculated all the outcomes and converted into outcome label (0-1 class)
- The GitHub URL of the completed Data Wrangling notebook (https://github.com/nagar-mayank/Applied-Data-Science-Capstone-Project/blob/main/Notebooks/Data_wrangling_ZekK73CQ9.ipynb)

EDA with Data Visualization

- We explored the data by visualizing the relation between ‘FlightNumber’ and ‘PayloadMass’, ‘FlightNumber’ and ‘LaunchSite’, ‘PayloadMass’ and ‘LauchSite’, ‘Orbit’ and ‘ClassMean’, ‘FlightNumber’ and ‘Orbit’, ‘PayloadMass’ and ‘Orbit’, ‘Year’ and ‘ClassMean’





The GitHub URL of the completed Data Visualization notebook (https://github.com/nagar-mayank/Applied-Data-Science-Capstone-Project/blob/main/Notebooks/eda-dataviz_fkU4Y76dt.ipynb)

EDA with SQL

- We used Jupyter Notebook magic method to connect to the SQL database
- We performed following tasks
  - 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 acheived.
  - 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. Use a subquery
  - 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
- The GitHub URL of the completed SQL EDA notebook (https://github.com/nagar-mayank/Applied-Data-Science-Capstone-Project/blob/main/Notebooks/eda-sql-coursera_xN21VGyFr.ipynb)

Build an Interactive Map with Folium

- We added all launch sites as circle object and markers with color based on the success of the mission.
- Different color-based markers shows us which launch site has a relatively high success rate.
- We calculated the distances between a launch site to its proximities.
- The GitHub URL of the completed launch site location notebook (https://github.com/nagar-mayank/Applied-Data-Science-Capstone-Project/blob/main/Notebooks/launch_site_location.ipynb)

Build a Dashboard with Plotly Dash

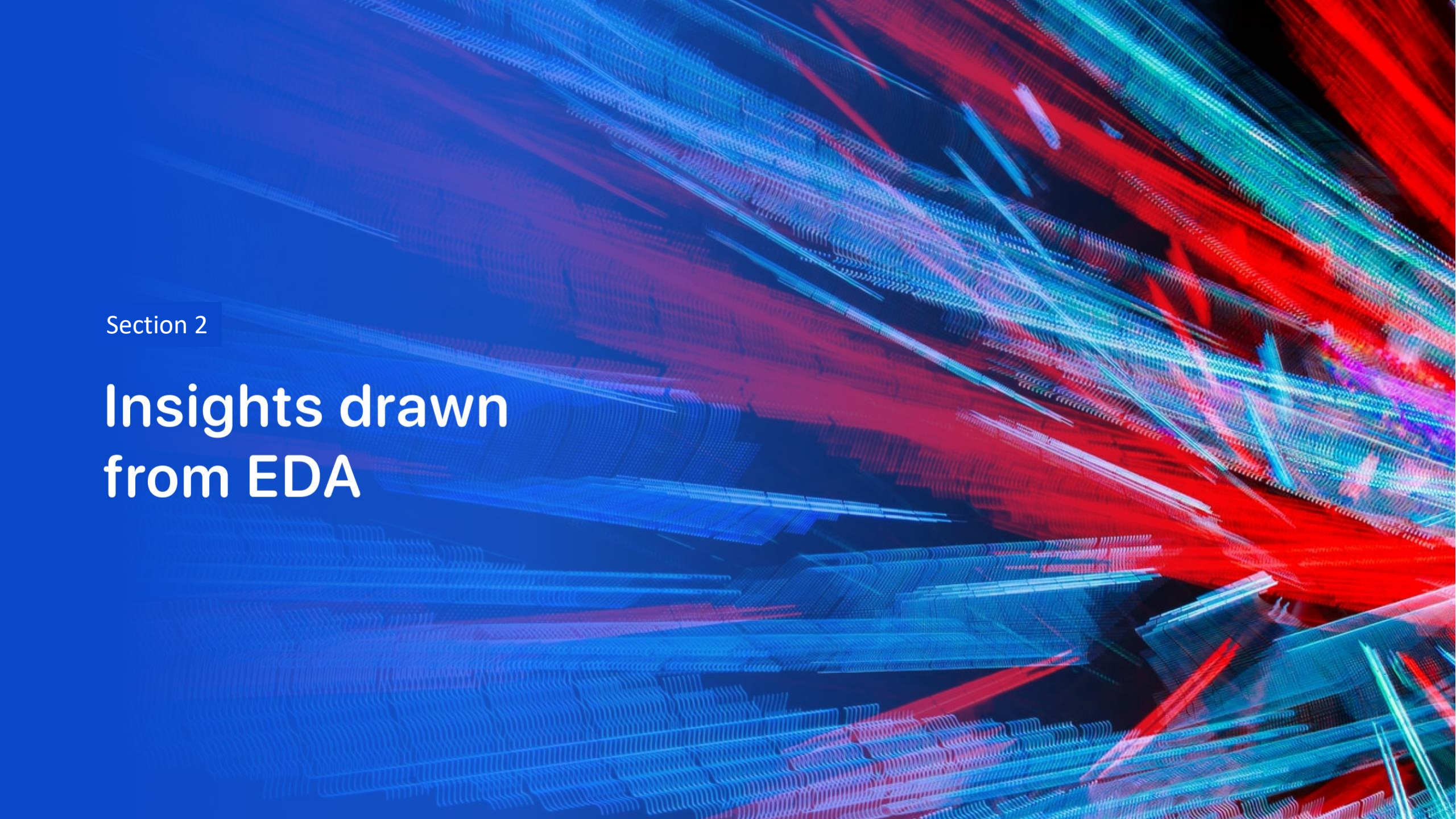
- We plotted a pie chart showing the total number of launches by a particular sites
- We plotted a scatter plot showing the relation between Outcome and payloas mass for a particular booster version
- The GitHub URL of the completed spacex dash app.py (https://github.com/nagar-mayank/Applied-Data-Science-Capstone-Project/blob/main/Notebooks/spacex_dash_app.py)

Predictive Analysis (Classification)

- First, we standardize the dataset and split it into train and test dataset
- Then, we searched the best parameters for LogisticRegression, SVM, DicisionTreeClassifier, KNeighboursClassifier using GridSearchCV from sklearn library
- We used accuracy as the metric for the models
- The GitHub URL of the completed Machine learning prediction Notebook (https://github.com/nagar-mayank/Applied-Data-Science-Capstone-Project/blob/main/Notebooks/SpaceX_Machine_Learning_Prediction_pmBMWQ6ia.ipynb)

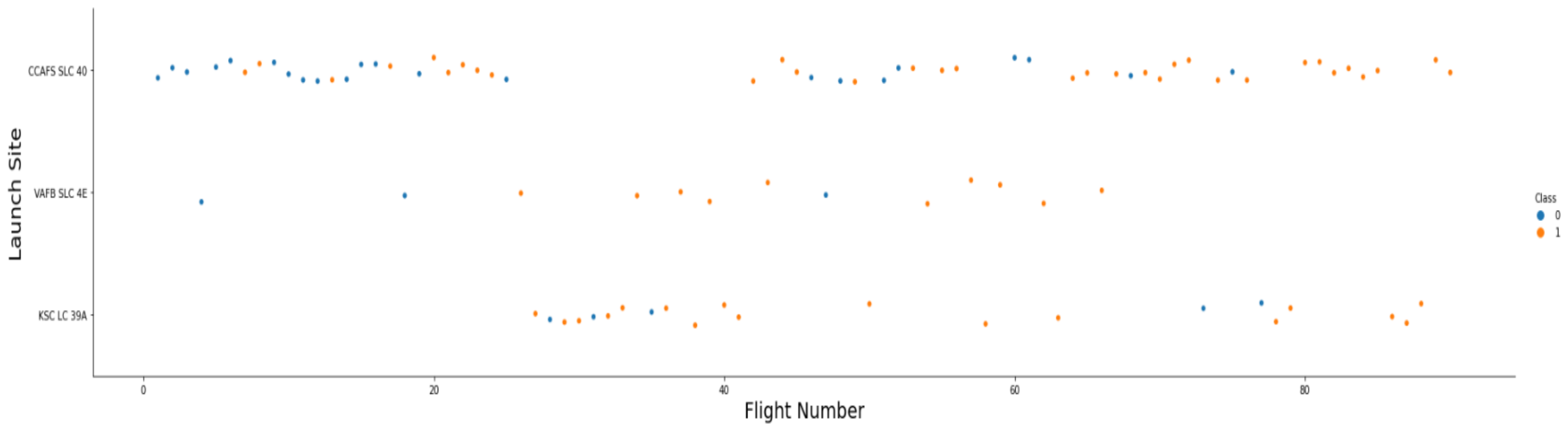
Results

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


Section 2
Insights drawn from EDA

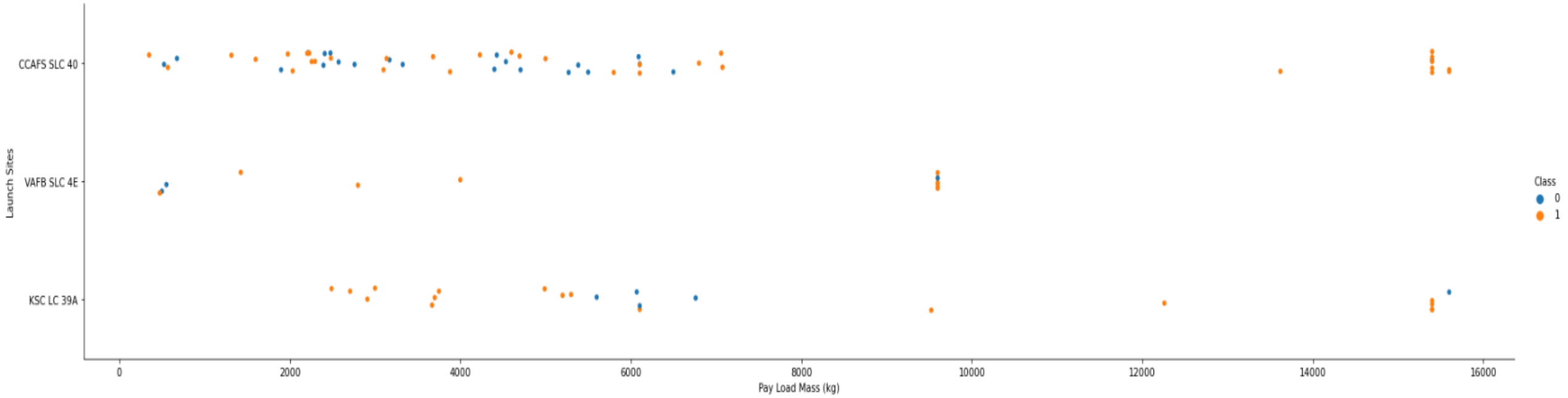
Flight Number vs. Launch Site

From the plot, it is clear that the larger the number of FlightNumber greater the success rate at the launch site



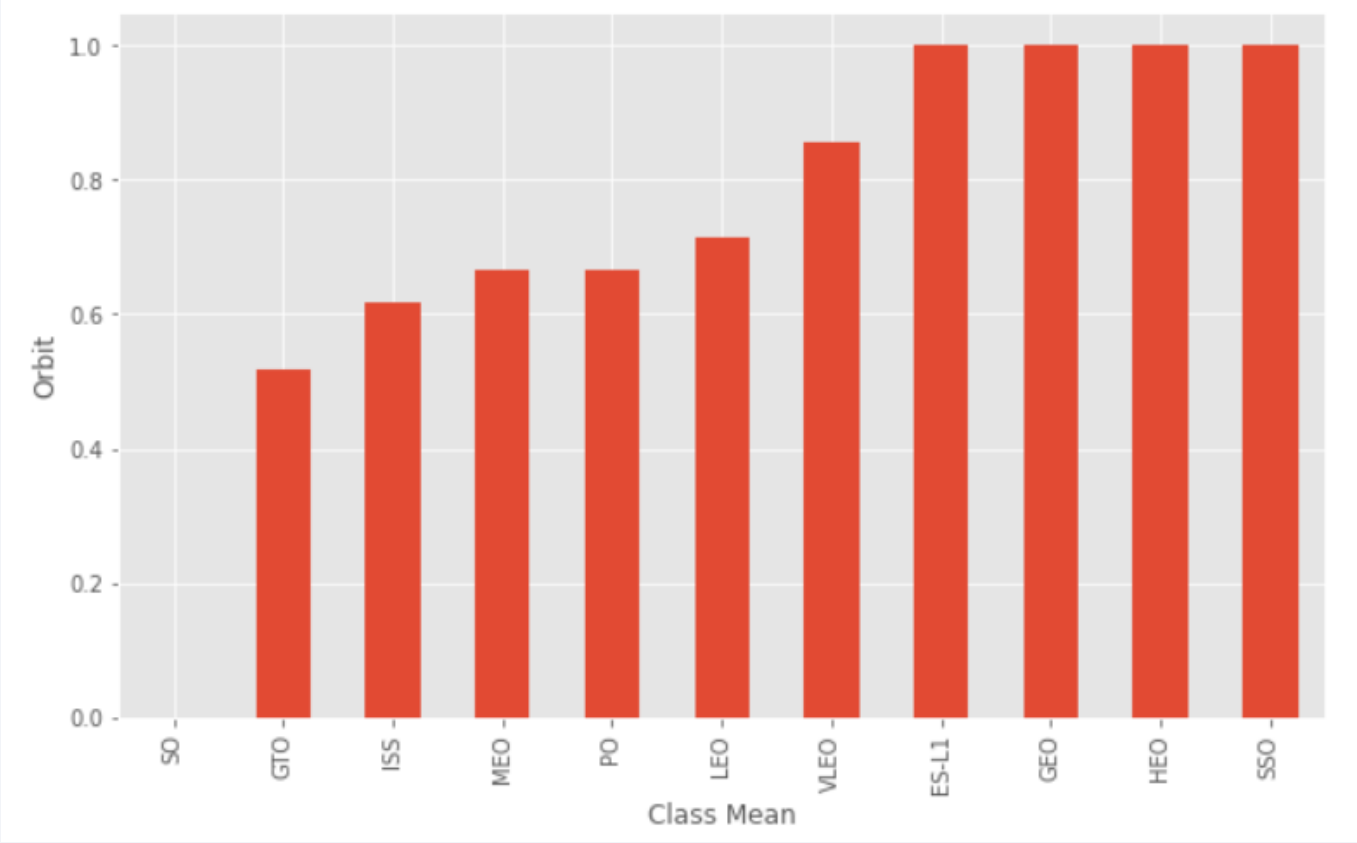
Payload vs. Launch Site

From the plot, it is clear that the larger the payload mass greater the success rate for CCAF5 SLC 40 Launch site and KSC LC 39A launch site had some unsuccessful mission when payload mass was around 6000 Kg



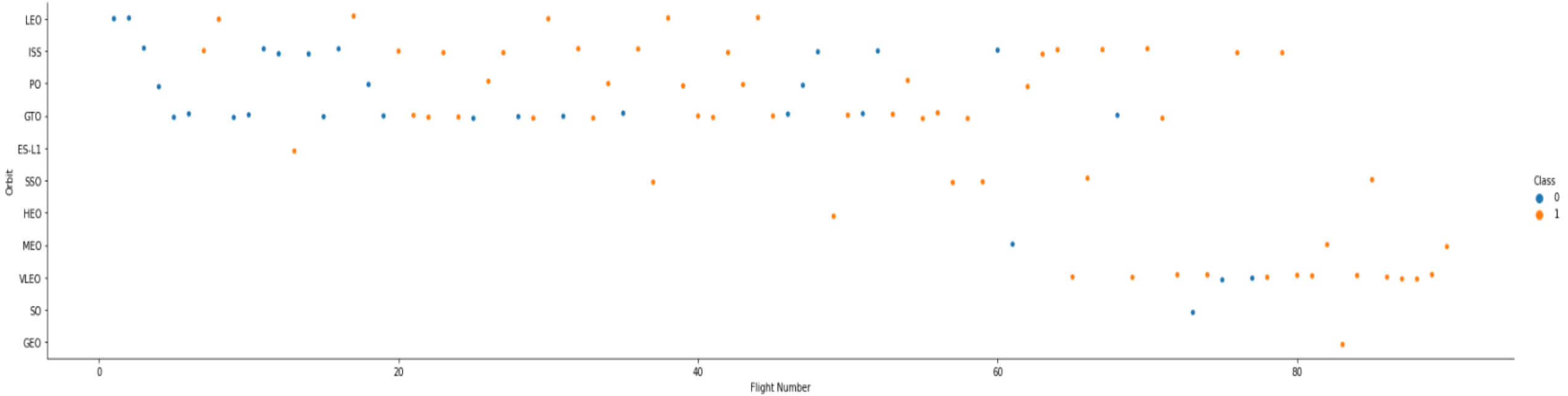
Success Rate vs. Orbit Type

From this bar chart, it is clear that satellites with SO Orbit have minimum successful missions and the satellites with ES-L1, GEO, HEO, SSO Orbits has maximum successful missions



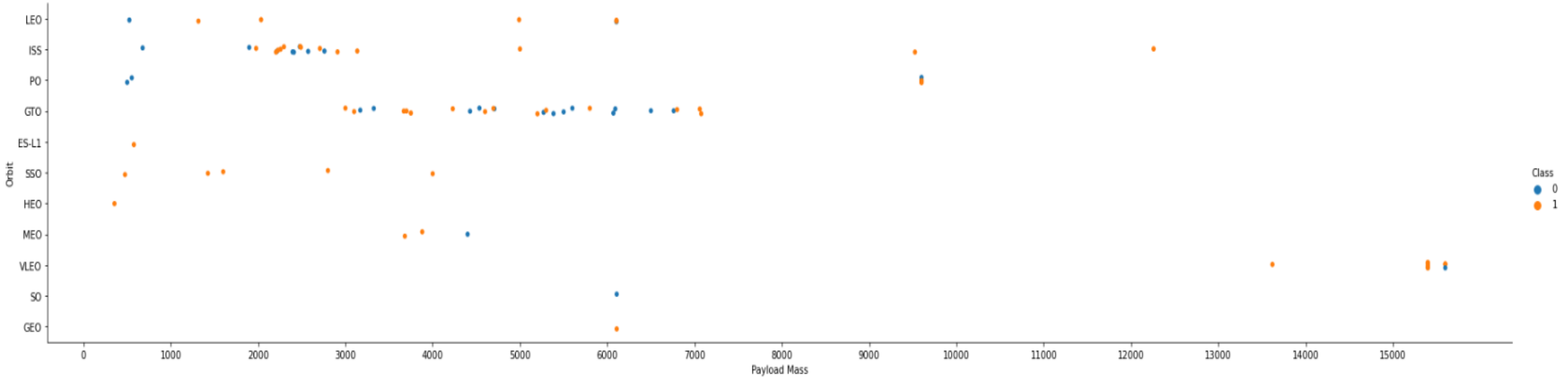
Flight Number vs. Orbit Type

From the plot it is clear that the success of the LEO Orbit satellite is related to the flight number and of the GTO Orbit satellite is not related to the flight number



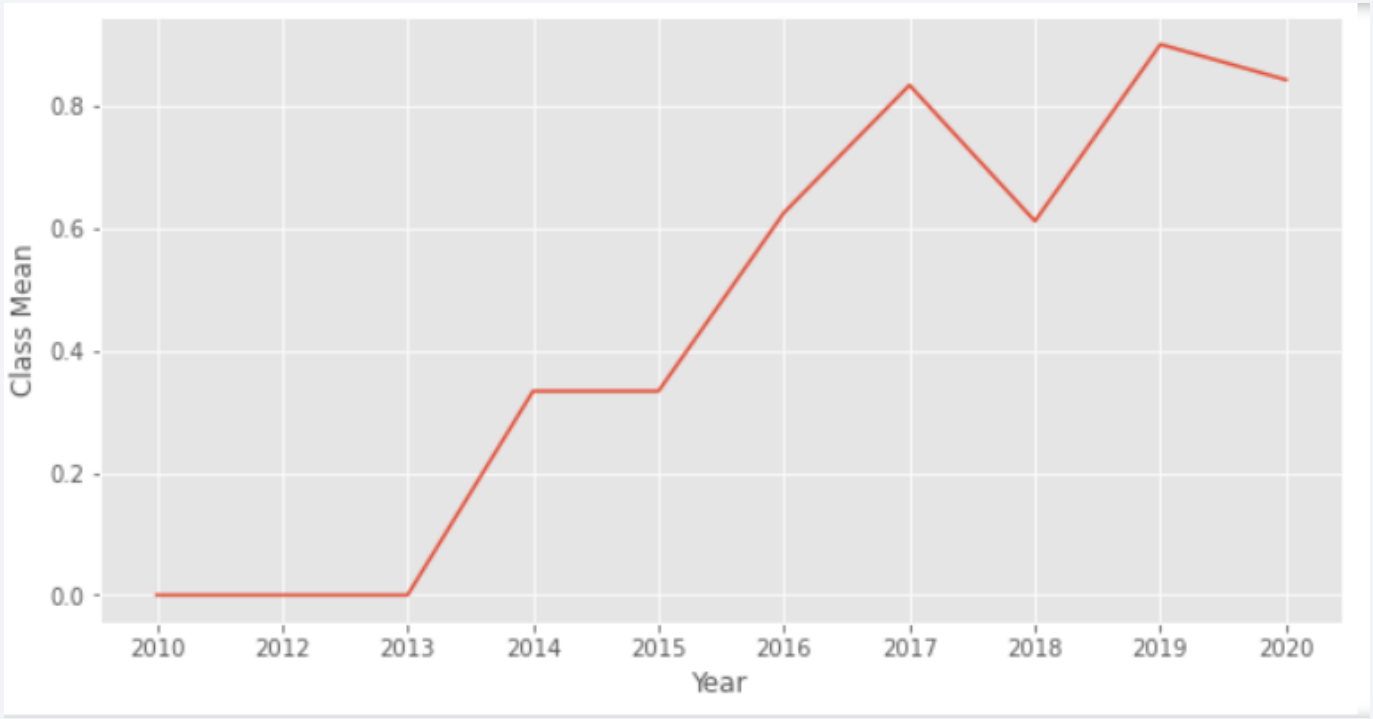
Payload vs. Orbit Type

From the plot, It is clear that the greater the payload mass greater the success rate of the LEO, ISS Orbit satellites



Launch Success Yearly Trend

From the line chart, it is clear that success rate is increasing over the years



All Launch Site Names

We used DISTINCT keyword to get the distinct launch sites from the dataset

Display the names of the unique launch sites in the space mission

```
%sql SELECT DISTINCT(LAUNCH_SITE) FROM SPACEXTBL;
```

```
 * ibm_db_sa://
Done.
```

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

Launch Site Names Begin with 'CCA'

We used WHERE and LIKE keywords to get the 5 records where launch site start with “CCA”

Display 5 records where launch sites begin with the string 'CCA'

```
%sql SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5
```

```
 * ibm_db_sa://vpz78947:
Done.
```

DATE	time_utc_	booster_version	launch_site	payload	payload_mass_kg_	orbit	customer	mission_outcome	landing_outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

We used SUM inbuild function to sum the payload mass

Display the total payload mass carried by boosters launched by NASA (CRS)

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE CUSTOMER='NASA (CRS)'
```

```
 * ibm_db_sa://
Done.
```

1
45596

Average Payload Mass by F9 v1.1

We used AVG inbuild function to get the average of payload mass

Display average payload mass carried by booster version F9 v1.1

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE BOOSTER_VERSION LIKE 'F9 v1.1%';
```

```
 * ibm_db_sa://
Done.
```

1
2534

First Successful Ground Landing Date

We used the MIN inbuild function to get the first date on which the landing outcome in the ground pad was successful

List the date when the first successful landing outcome in ground pad was acheived.

Hint:Use min function

```
%sql SELECT MIN(DATE) FROM SPACEXTBL WHERE LANDING__OUTCOME='Success (ground pad)'
```

```
 * ibm_db_sa://
Done.
```

1
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

We used WHERE and AND keywords to get the successful drone ship landing with payload between 4000 and 6000

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE LANDING__OUTCOME='Success (drone ship)' AND PAYLOAD_MASS__KG_>4000 AND PAYLOAD_MASS__KG_<6000;
```

```
 * ibm_db_sa://
Done.
```

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

Total Number of Successful and Failure Mission Outcomes

We used the COUNT inbuilt function and grouped by mission outcome column using GROUP BY keyword

List the total number of successful and failure mission outcomes

```
%sql SELECT MISSION_OUTCOME, COUNT(*) FROM SPACEXTBL WHERE MISSION_OUTCOME IN ('Failure (in flight)', 'Success') GROUP BY MISSION_OUTCOME;
```

```
 * ibm_db_sa://
Done.
```

mission_outcome	2
Failure (in flight)	1
Success	99

Boosters Carried Maximum Payload

We used subquery to get the names of boosters carried maximum payload

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

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

```
 * ibm_db_sa://
Done.
```

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

2015 Launch Records

We used LIKE and AND keywords in WHERE clause to get the results

List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

```
%%sql
SELECT BOOSTER_VERSION, LAUNCH_SITE, LANDING__OUTCOME, DATE FROM SPACEXTBL WHERE DATE LIKE '2015%' AND LANDING__OUTCOME='Failure (drone ship)';
```

```
 * ibm_db_sa://
Done.
```

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

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

We used AND keyword in the WHERE clause and the further result is grouped by landing outcome and sorted by the count

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

```
%sql SELECT LANDING__OUTCOME, COUNT(*) AS count FROM SPACEXTBL WHERE DATE>'2010-06-04' AND DATE<'2017-03-20' GROUP BY LANDING__OUTCOME ORDER BY COUNT(*) DESC;
```

```
 * ibm_db_sa:/
Done.
```

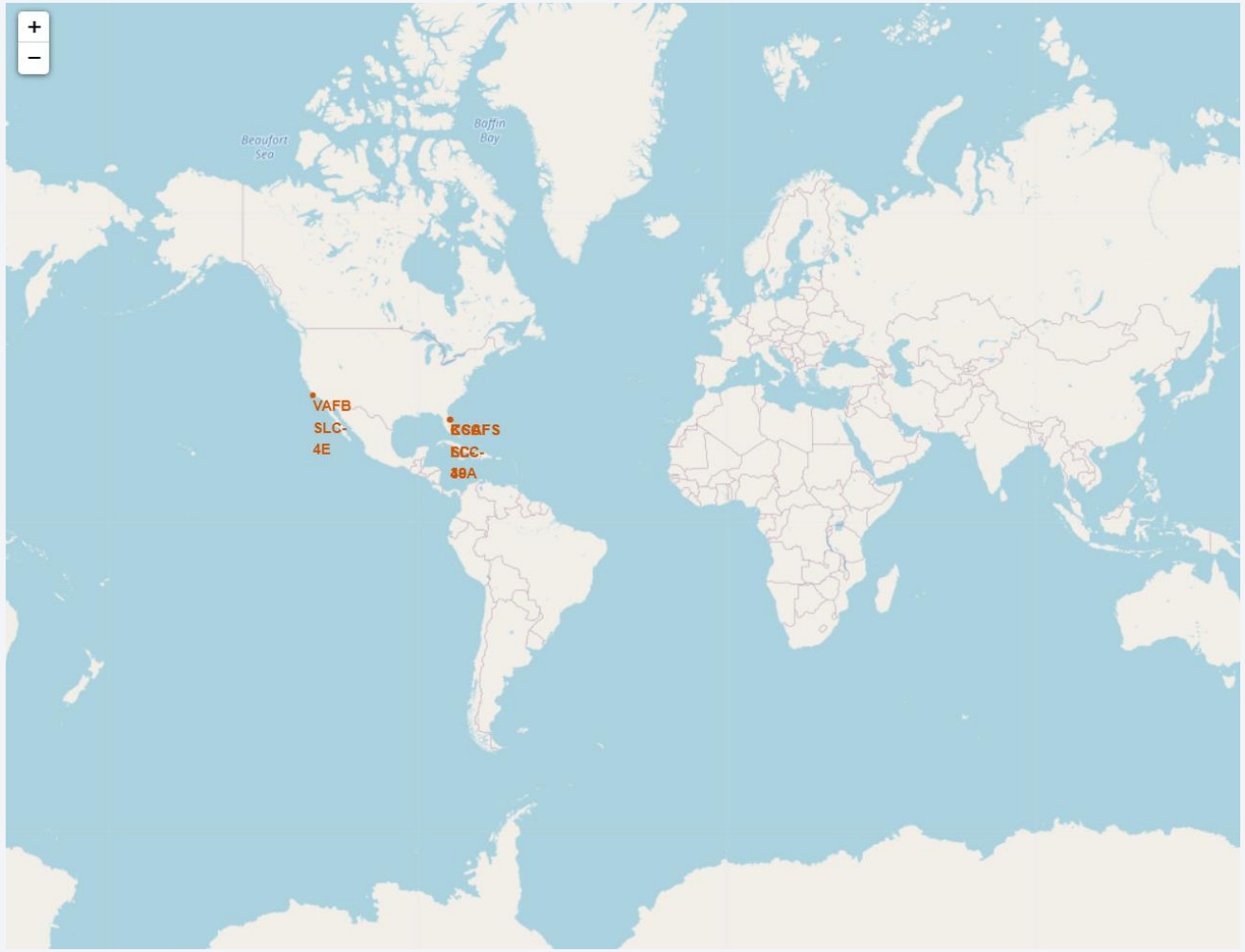
landing__outcome	COUNT
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Uncontrolled (ocean)	2
Failure (parachute)	1
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

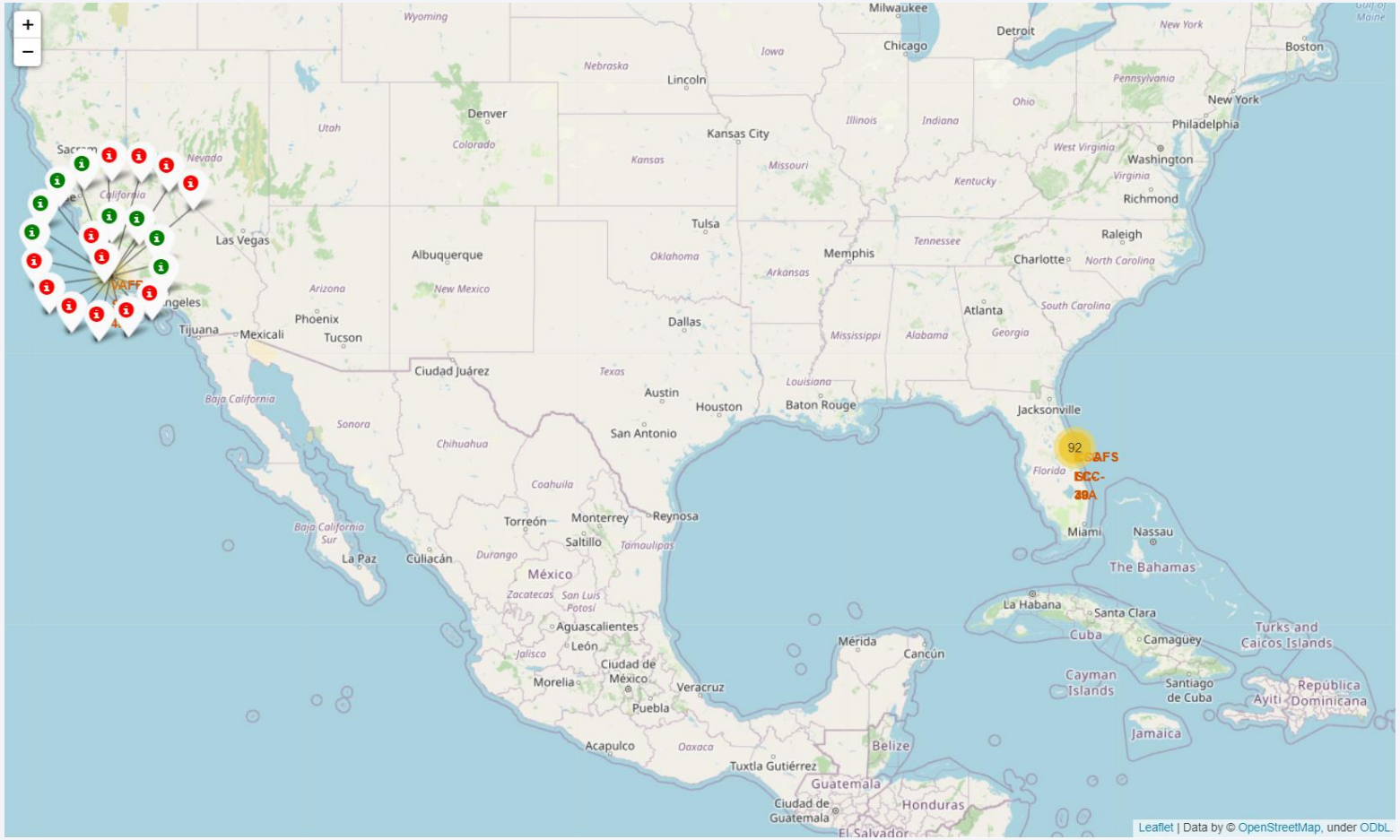
All Launch Sites on A Global Map

From this map, we can see that all launch sites are on the United States of America's coasts

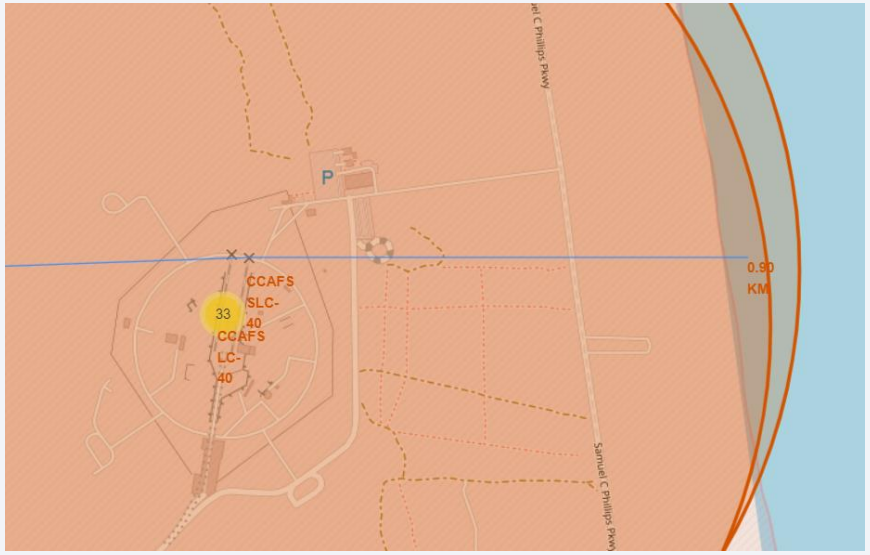


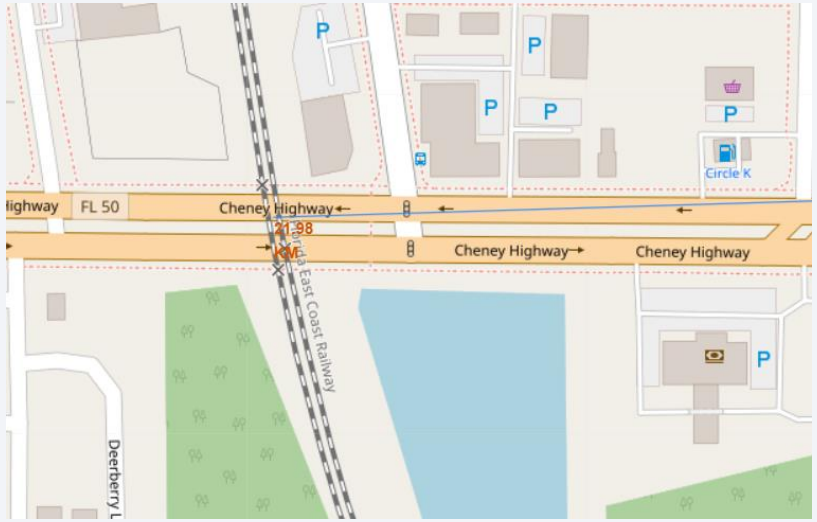
Color Markers Showing Outcome of a Launch for a Launch Site

Green Markers shows a successful mission and Red Markers shows a unsuccessful mission

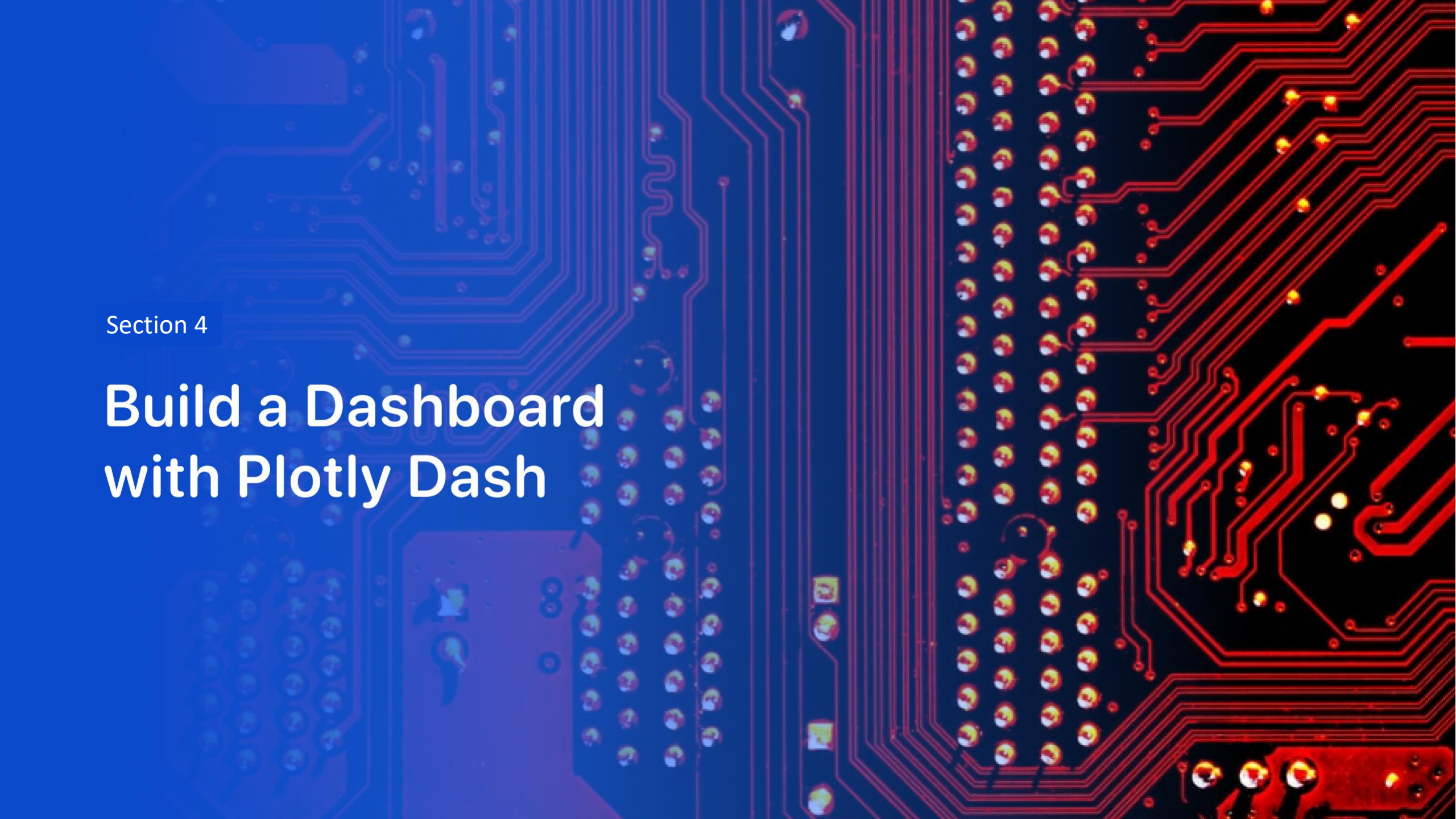


Launch Site distances to landmarks

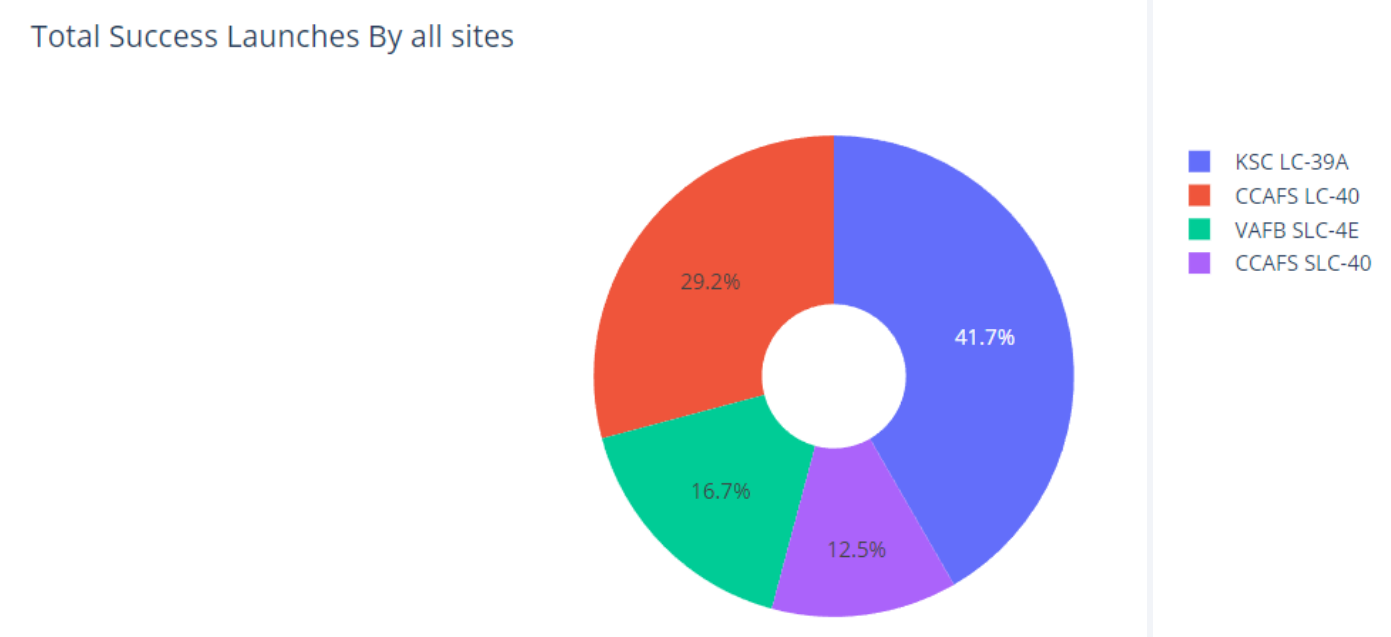




Distance from the nearest highway is 78.45 KM
Distance from the coastline is 0.90 KM
The distance from the nearest railway line is 28.98 KM
It is essential for the safety of civilians that the launch site is at a proper distance from all city's landmarks and crowded area

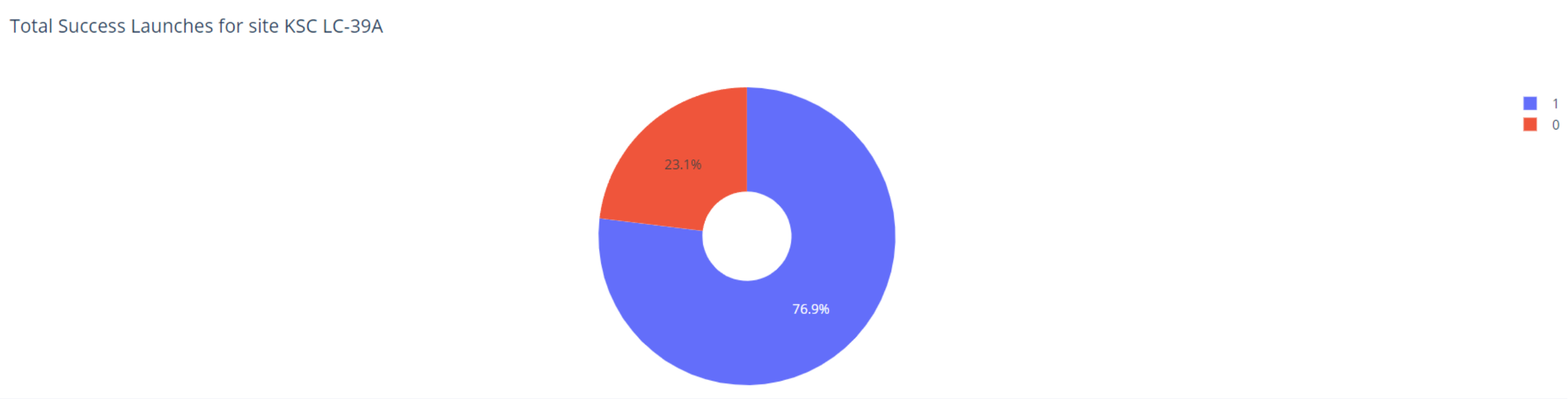

Section 4
Build a Dashboard with Plotly Dash

Pie Chart of Launch Success for All Sites



From the pie chart, it is clear that KSC LC-39A has the most successful missions

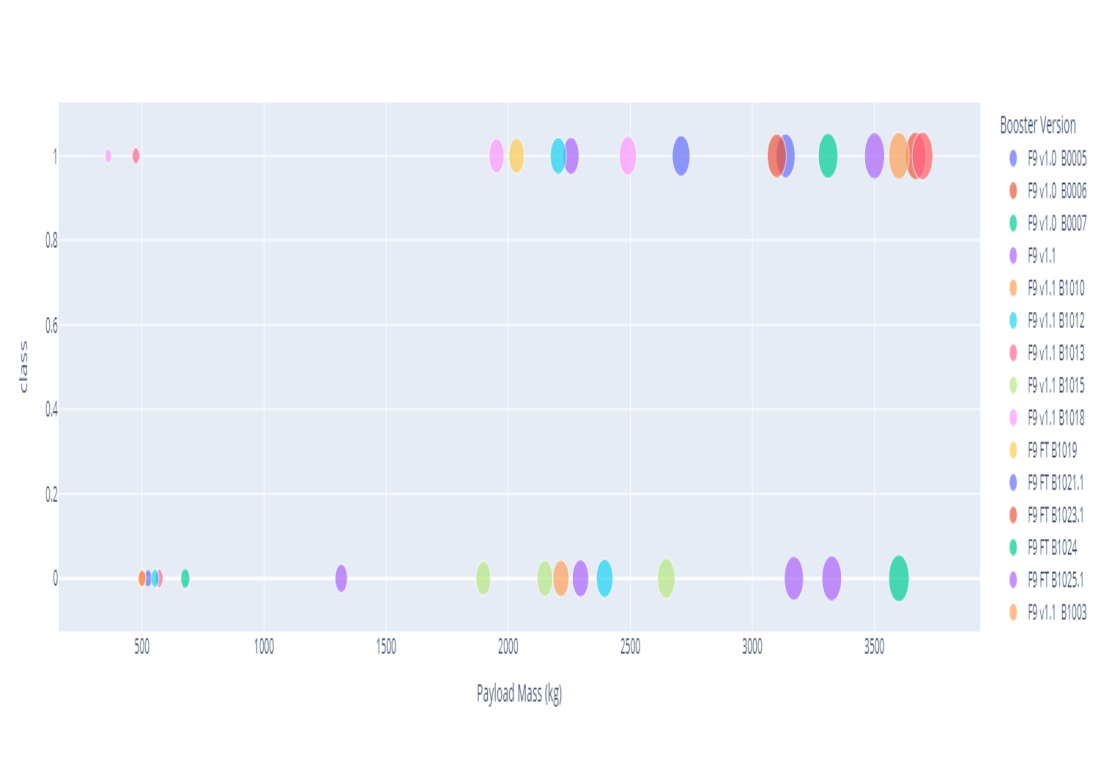
Pie Chart of Highest Success Rated Site



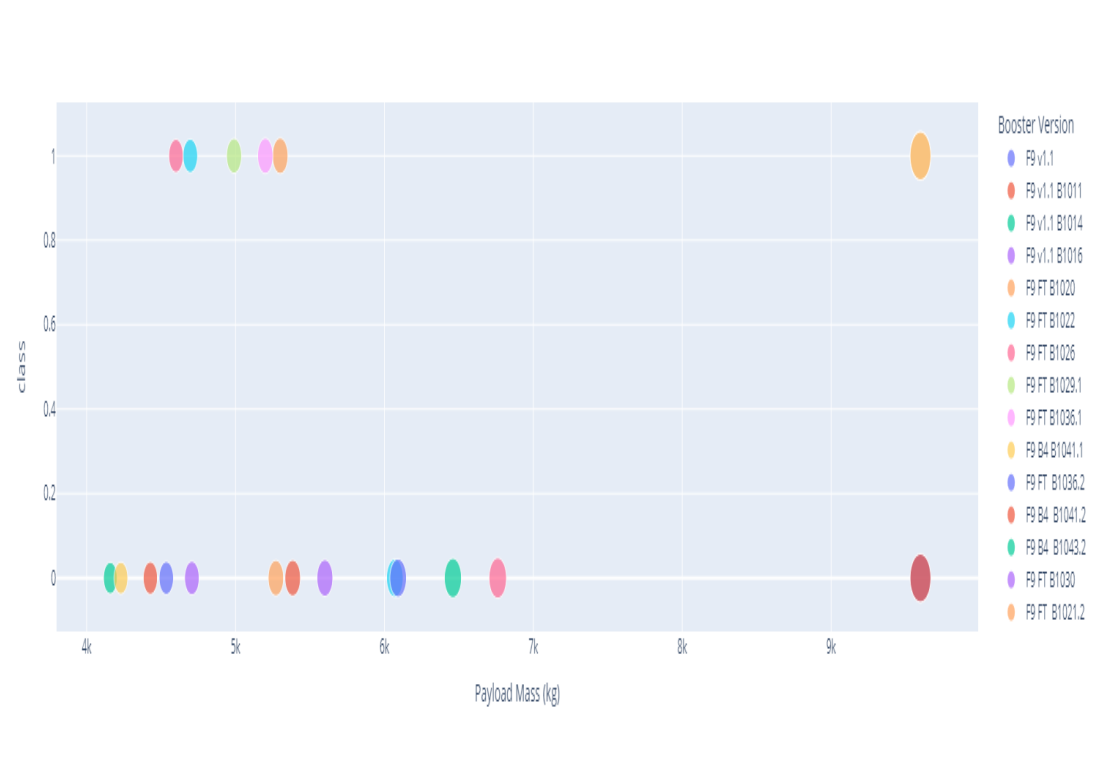
From the pie chart, it is clear that KSC LC-39A has 76.9% success rate

Scatter Plot for Different Payload Ranges

Low weighted payload (<4000 KG)



High weighted payload (4000<10000 KG)

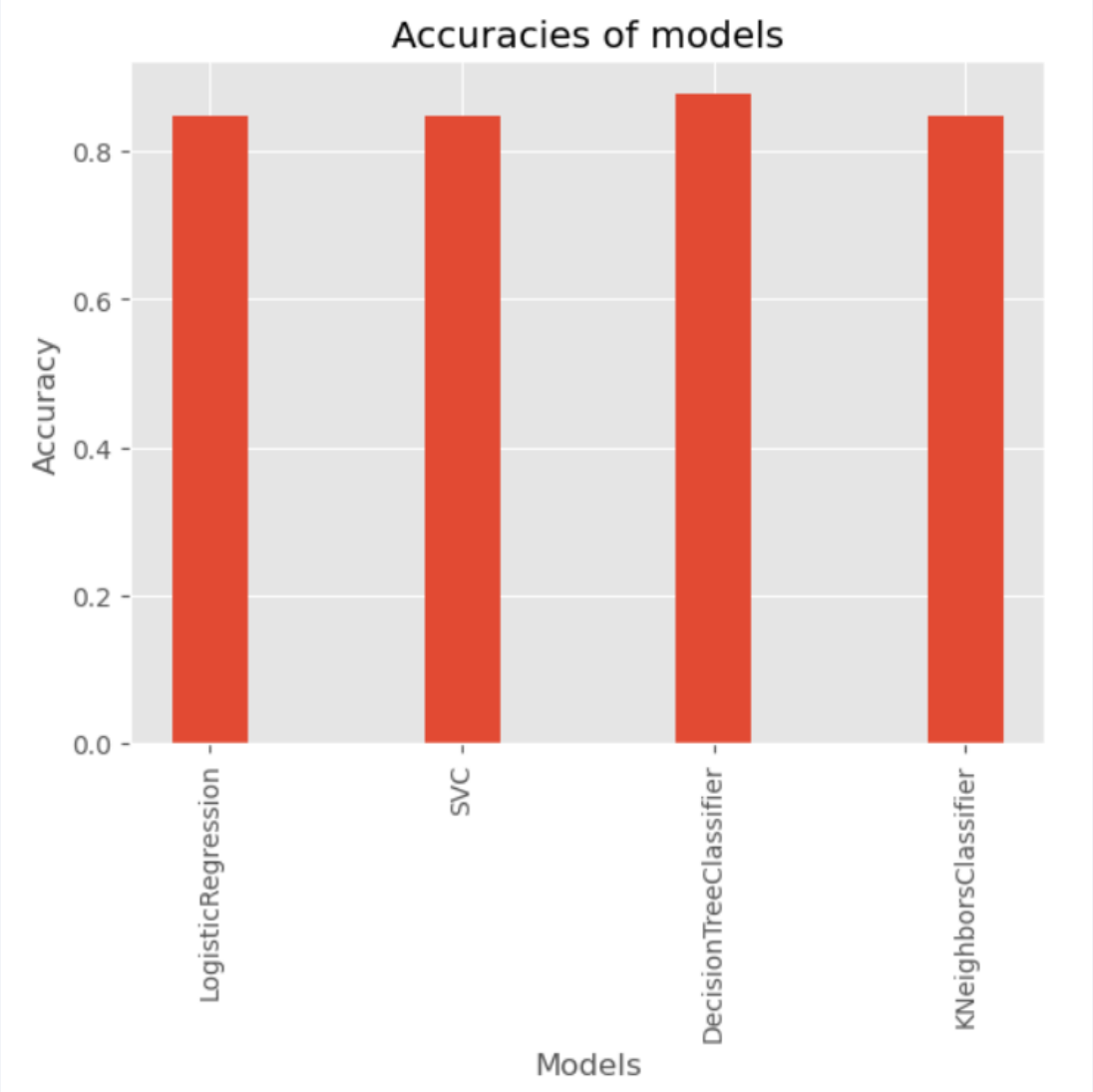


From the scatter plot, it is clear that mission with low payload weight is higher than high payload weight.

Section 5

Predictive Analysis (Classification)

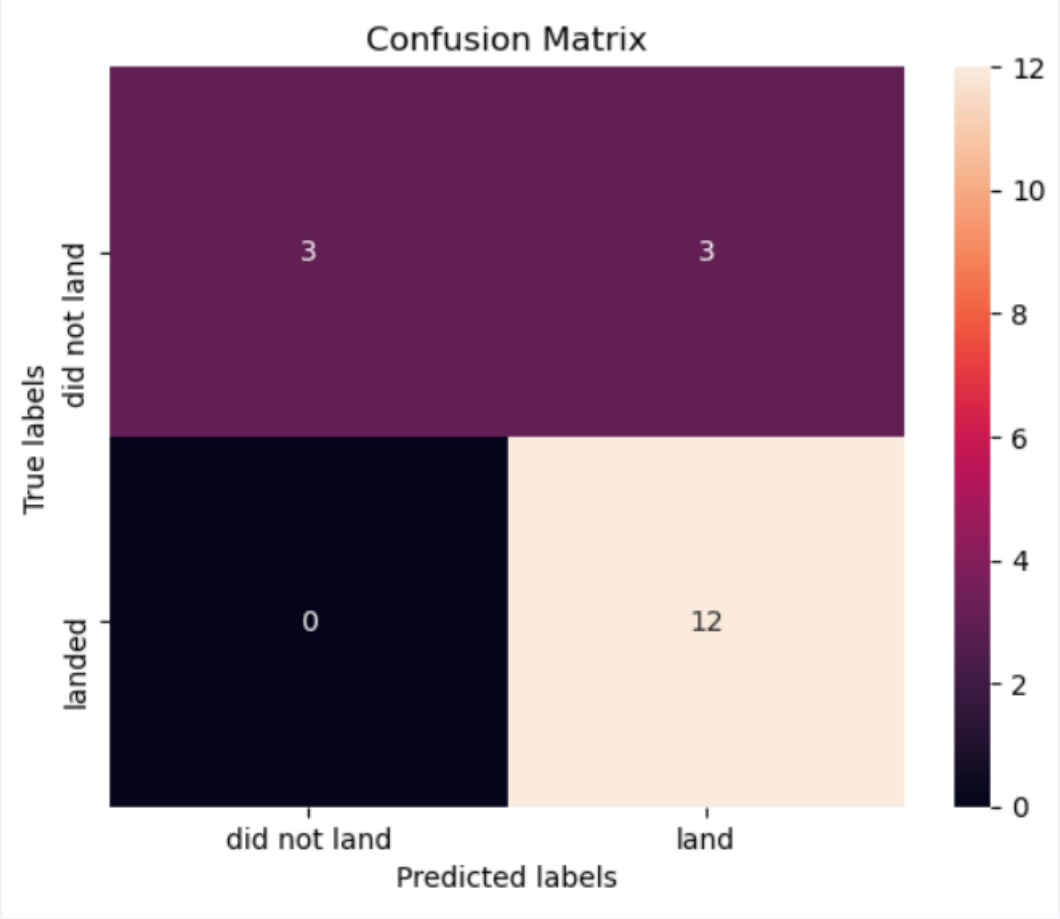
Classification Accuracy



From the bar chart, it is clear that Decision Tree Classifier has the highest accuracy.

Confusion Matrix

From the Confusion Matrix of Decision Tree Classifier, it is clear that it is very robust when class is 1. The model struggles with False Positives



Conclusions

From this study, we can conclude that

- Larger the Flight Number greater the success rate
- Mission success rate is increasing from 2013 till 2020
- KSC LC-39A has the most successful rate over all sites
- The Decision Tree Classifier is the best model to predict the success of the mission

Thank you!