IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Ignas Chilewa
November 14th, 2022

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion

Executive Summary

- **Summary of methodologies**
  i. Data Collection API
  ii. Data Collection with Web Scraping
  iii. Data Wrangling
  iv. Exploratory Data Analysis Using SQL
  v. Exploratory Data Analysis for Data Visualization
  vi. Interactive Visual Analytics and Dashboard using Folium and Ploty Dash
  vii. Prediction Analysis - ML
- Summary of all results
  i. Exploratory Data Analysis Results
  ii. Interactive Visual Analytics Results
  iii. Predictive Analytics Results

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.

  The goal of the project is to create a machine learning model that will predict if the first stage will land successfully.

- Problems we want to solve:

  i. What factors determine if the rocket will land successfully?

  ii. The interaction amongst various features that determine the success rate.

  iii. What operating conditions needs to be in place to ensure a successful landing.

Section 1

Methodology

Methodology

Executive Summary

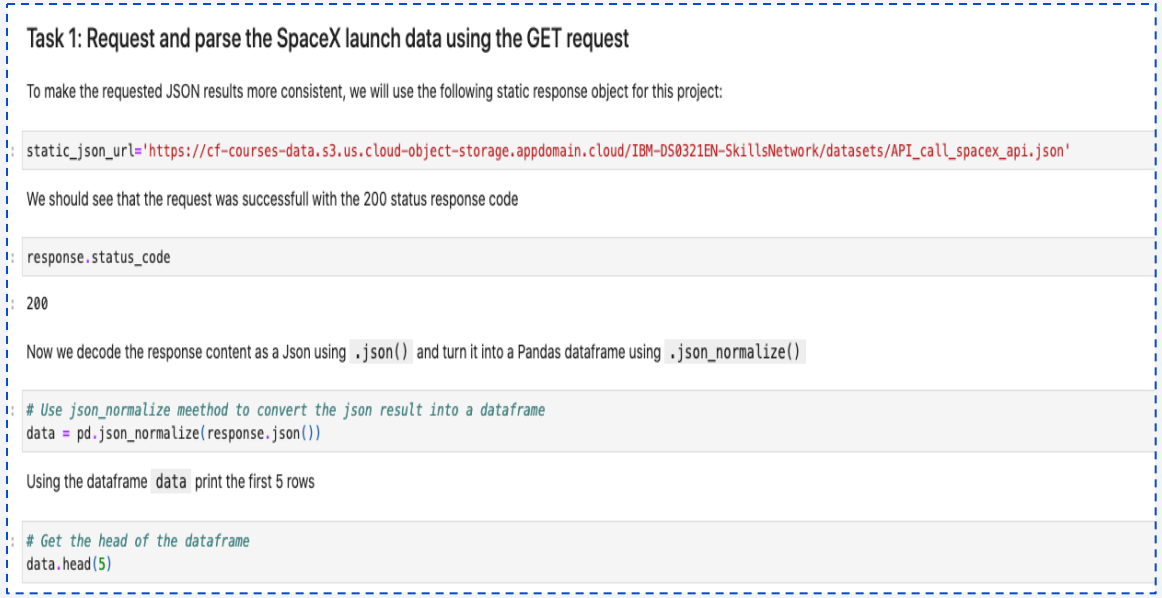
- Data collection methodology:
  - Data was collected using SpaceX API and web scraping from Wikipedia.
- Perform data wrangling
  - One-hot encoding was applied to categorical variables.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

- The data was collected using various methods
  - Data collection was done using get function of the request library to the SpaceX API.
  - Next, we decoded the response content as a Json using .json() function call and turn it into a pandas dataframe using pandas json_normalize() function.
  - We then cleaned the data, checked for missing values and fill in missing values(using their average by applying mean value) where necessary.
  - In addition, we performed web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup python library.
  - The objective was to extract the launch records as HTML table, parse the table and convert it to a pandas dataframe for future analysis.

Data Collection – SpaceX API

- We used the get request to the SpaceX API to collect data, clean the requested data and did some basic data wrangling and formatting.
- The link to the notebook is [Data Colletion Notebook]



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:

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

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

```
response.status_code
```

```
200
```

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

```
# 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

```
# Get the head of the dataframe
data.head(5)
```

Data Collection - Scraping

- We applied web scrapping technique to webscrap Falcon 9 launch records with BeautifulSoup object.
- We parsed the table and converted it into a pandas dataframe.
- The link to the notebook is [Web Scrapping Notebook]

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.

[7]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url)
```

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

[8]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content

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

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

[9]:
```
# Use soup.title attribute
soup.title
```

[9]:
```
<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

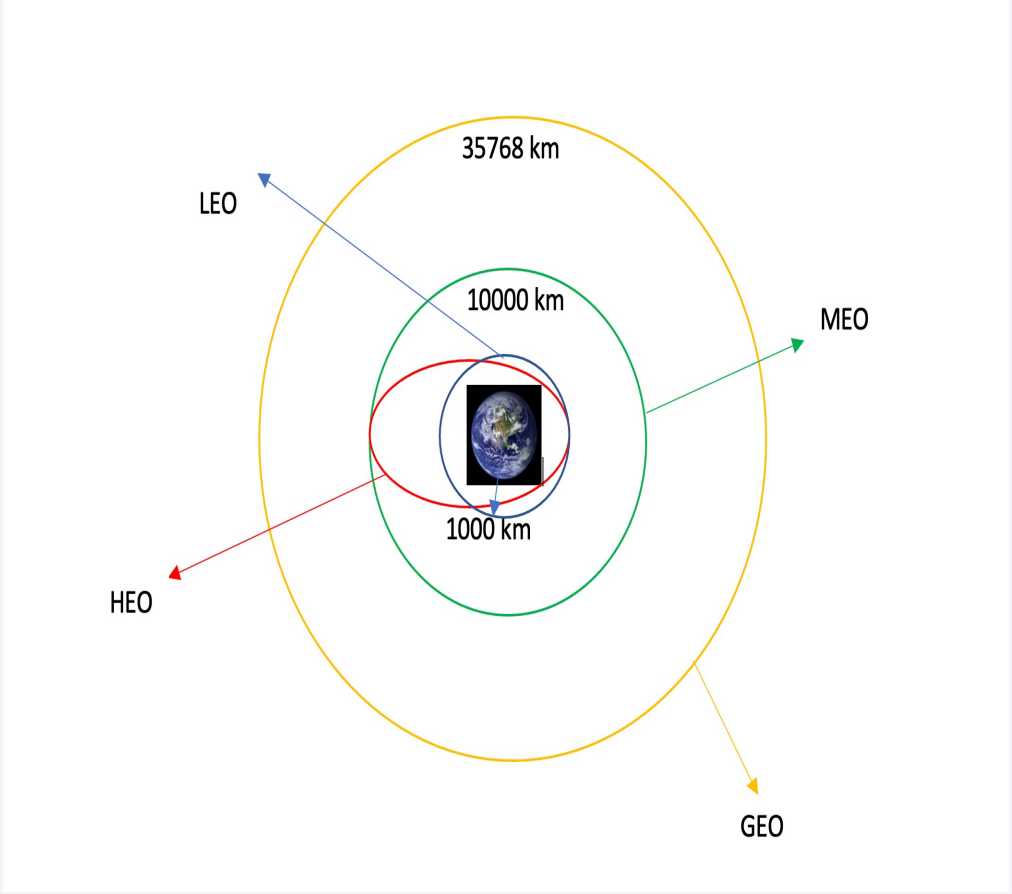
[10]:
```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all("table")
```

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

[11]:
```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

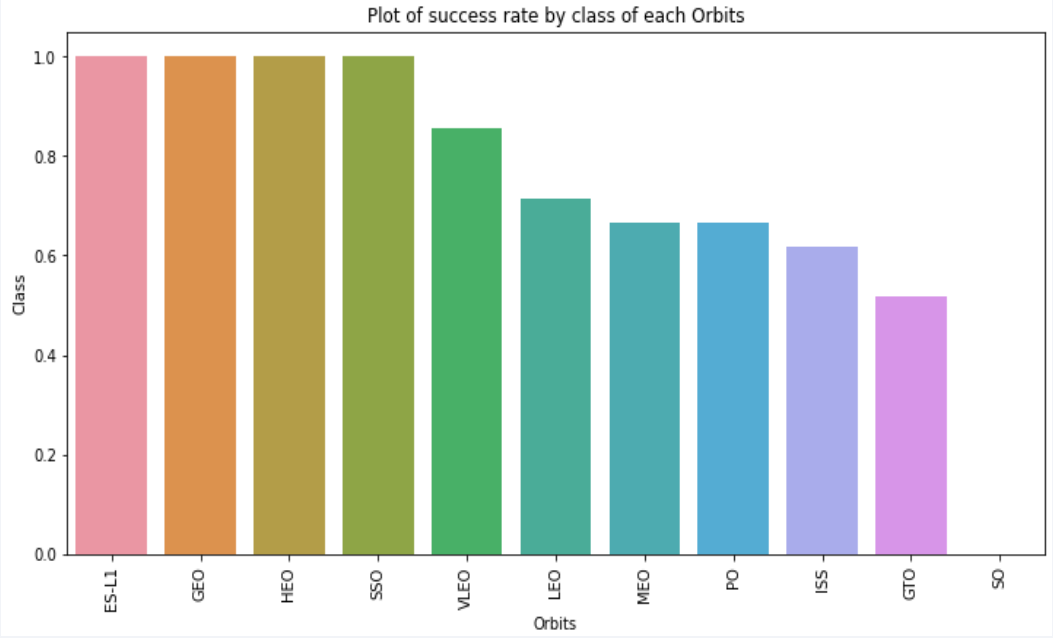
Data Wrangling

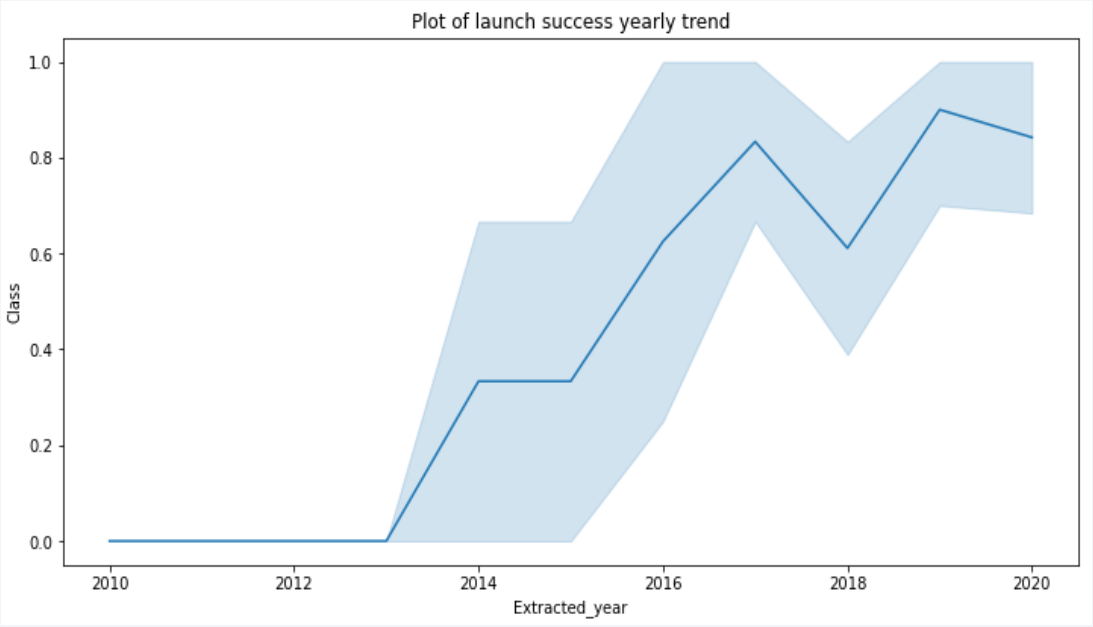
- We performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits
- We created landing outcome label(Class) from outcome column and exported the results to csv.
- The link to the notebook is Data Wrangling Notebook



EDA with Data Visualization

- We explored the data by visualizing the relationship between flight number and launch Site, payload and launch site, success rate of each orbit type, flight number and orbit type, the launch success





EDA with SQL

- We loaded the SpaceX dataset into a DB2 database.
- We applied EDA with SQL to get insight from the data. Below are sample queries:
  - The names of unique launch sites in the space mission.
  - 5 records where launch sites begin with the string 'CCA'.
  - The total payload mass carried by boosters launched by NASA (CRS)
  - The average payload mass carried by booster version F9 v1.1
  - Total number of successful and failure mission outcomes
  - The failed landing outcomes in drone ship, their booster version and launch site names.
- The link to the notebook is SQL Notebook.

Build an Interactive Map with Folium

- 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. 0 for failure, and 1 for success.
- Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate.
- We calculated the distances between a launch site to its proximities. We answered some question for instance:
  - Are launch sites near high traffic areas.
  - Do launch sites keep certain distance away from cities.

Build a Dashboard with Plotly Dash

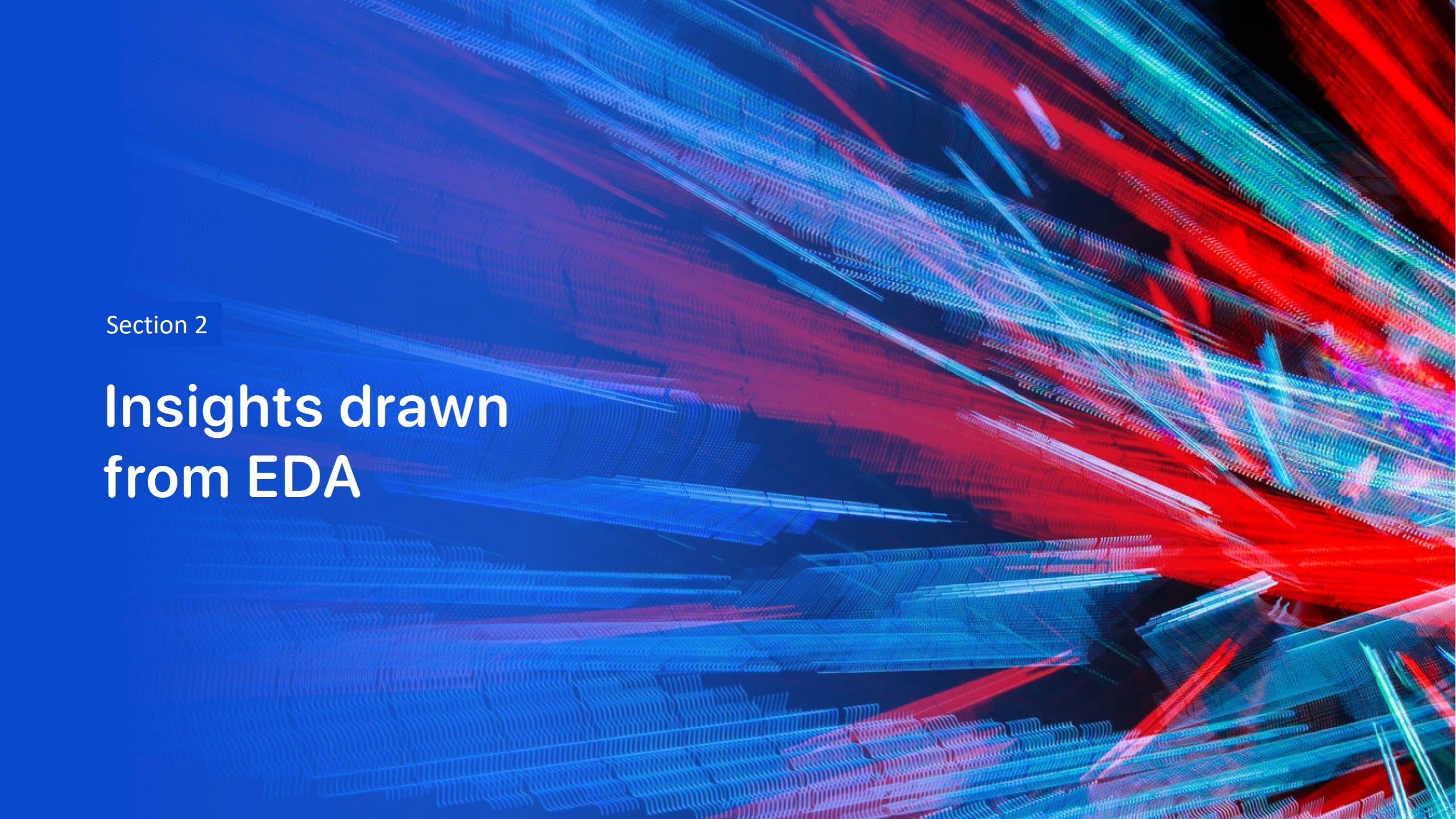
- Part of the project we built an interactive dashboard using Plotly library.
- Plotted pie charts showing the total number of launches by site.
- We plotted scatter graph showing the relationship between Outcome of landing and Payload Mass (Kg) for the different Booster versions.

Predictive Analysis (Classification)

- We created a column for the class(1 for success and 0 for failure)
- Standardize(Transformed) the data.
- Split into training data and test data.
- We built different machine learning models and tune to different hyperparameters using GridSearchCV algorithm.
- We used accuracy as well as score as the metric for our model, improved the model using feature engineering and algorithm tuning.
- We found the best performing classification model based on the score.
- The link to the notebook is Predictive Analysis Notebook.

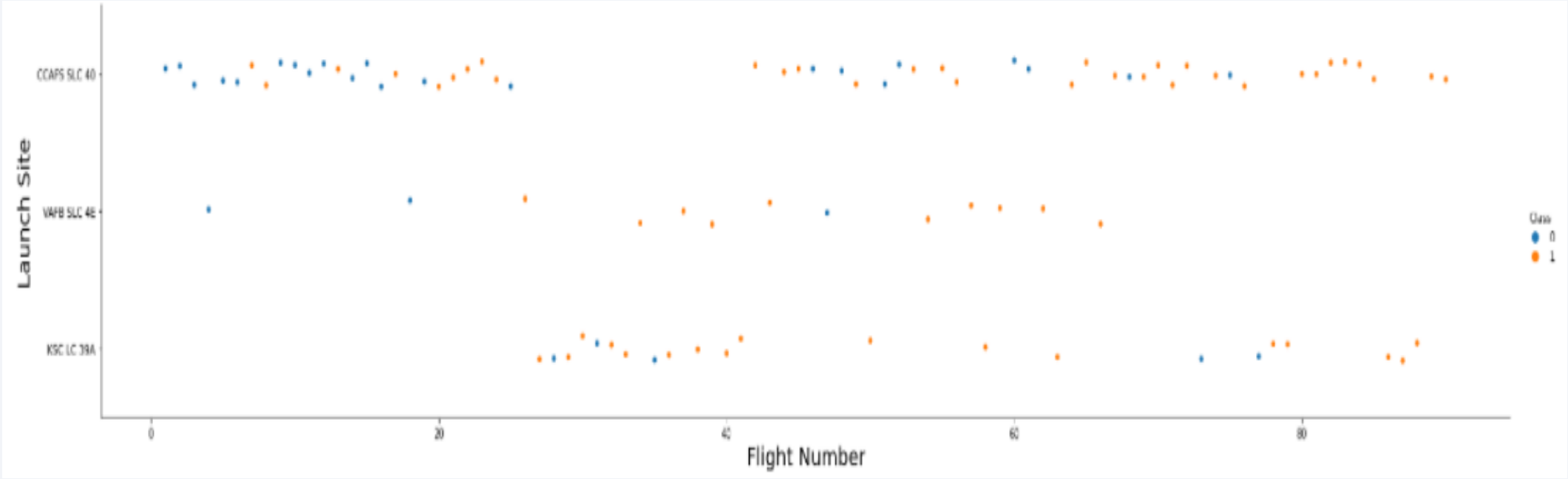
Results

- Insight drawn from EDA.
- Interactive Analytics demo.
- Predictive Analysis results.


Section 2
Insights drawn from EDA

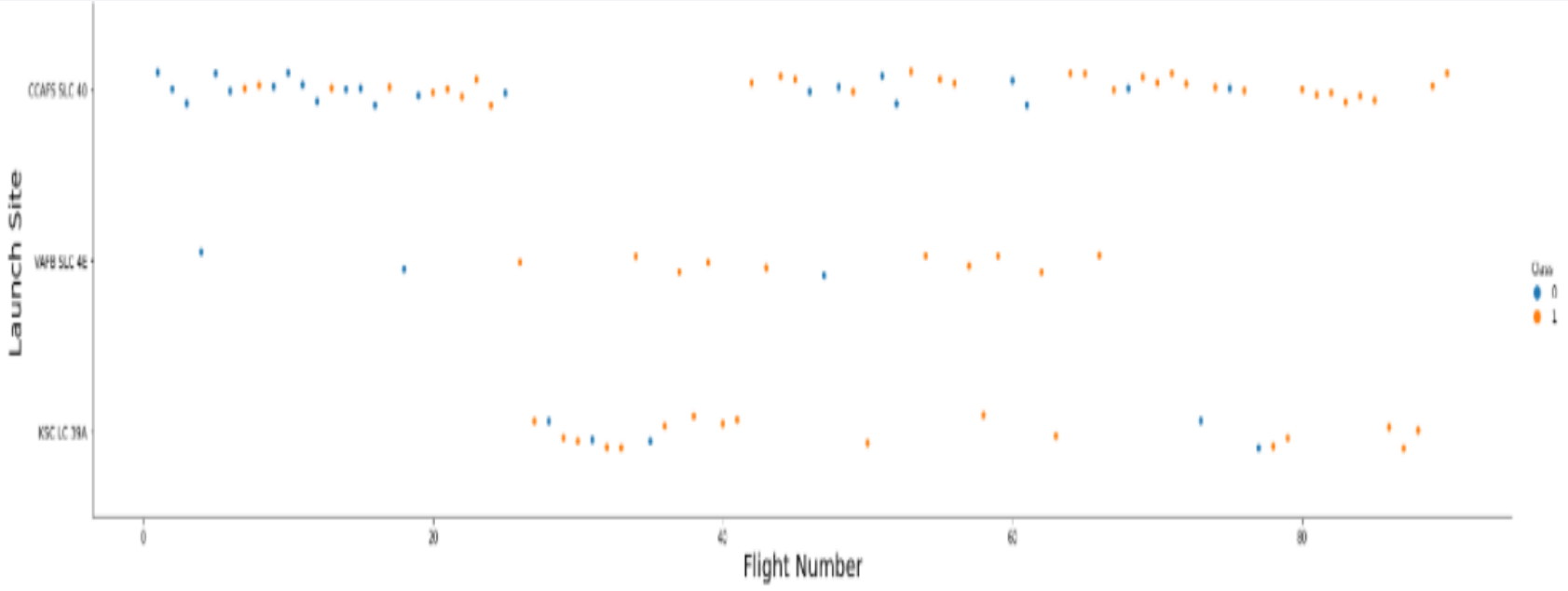
Flight Number vs. Launch Site

- From the scattered plot, we found that the larger the flight amount at a launch site, the greater the success rate at a launch site particular for the two sites.



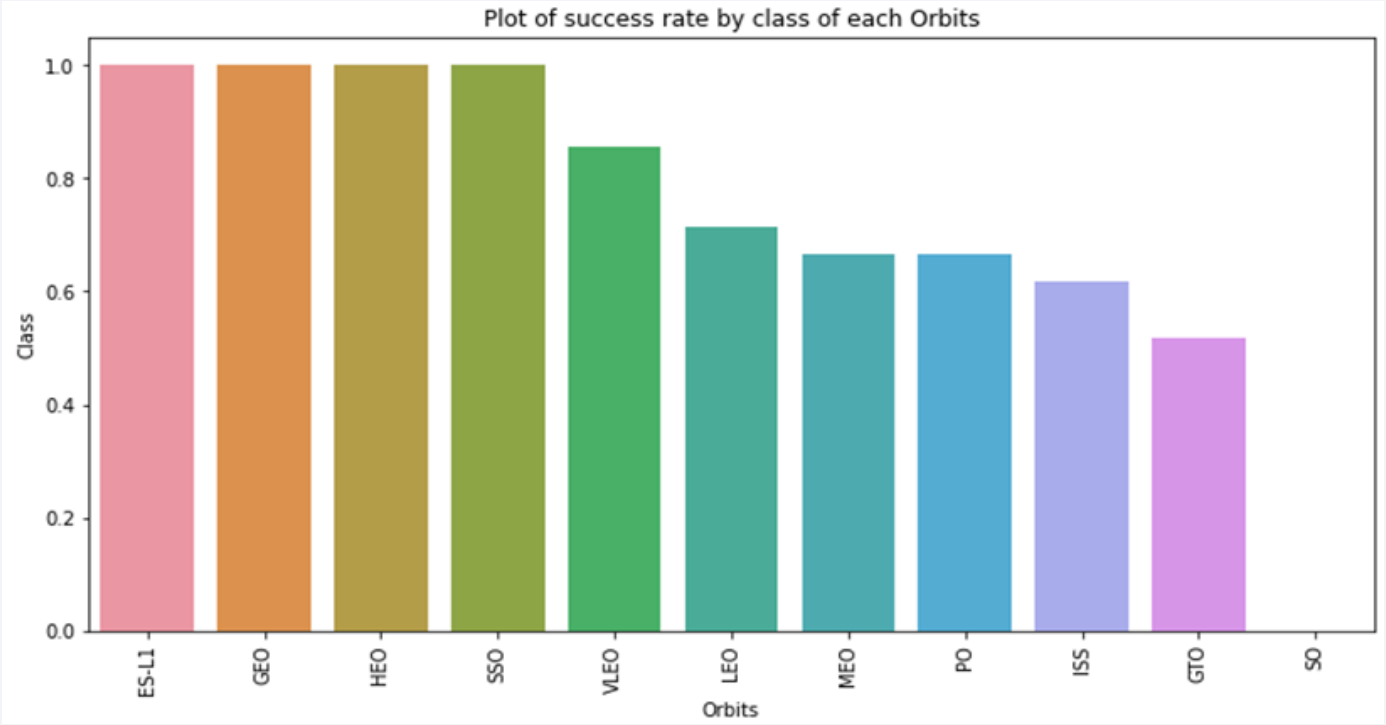
Payload vs. Launch Site

- We have seen that the greater the payload mass for the Launch Site CCAFS SLC 40 the higher the success rate .



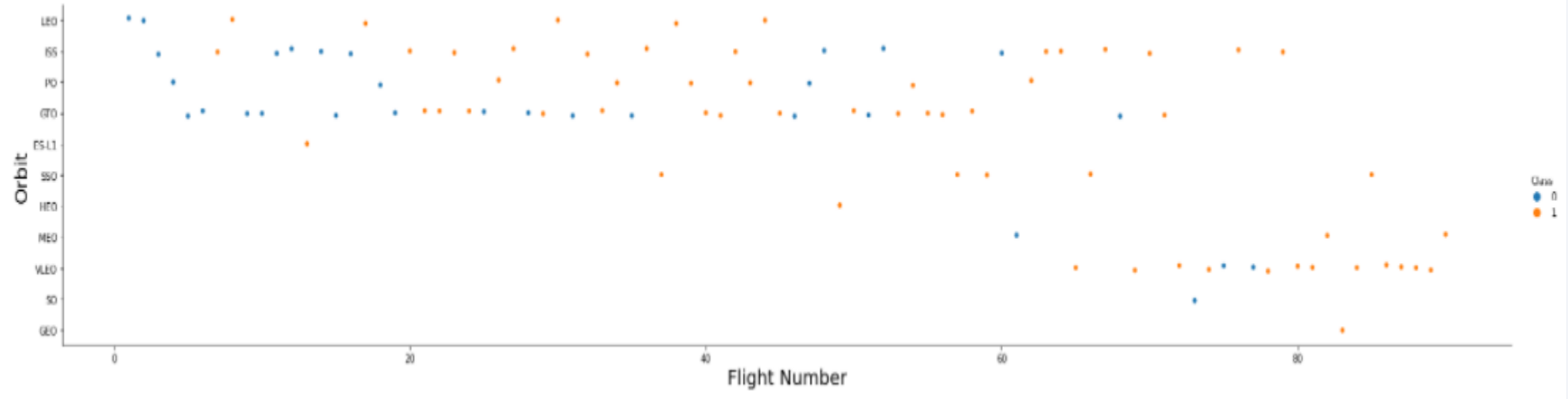
Success Rate vs. Orbit Type

- From the bar chat, we can see that ES-L1, GEO, HEO, SSO, VLEO had the most success rate of other sites.



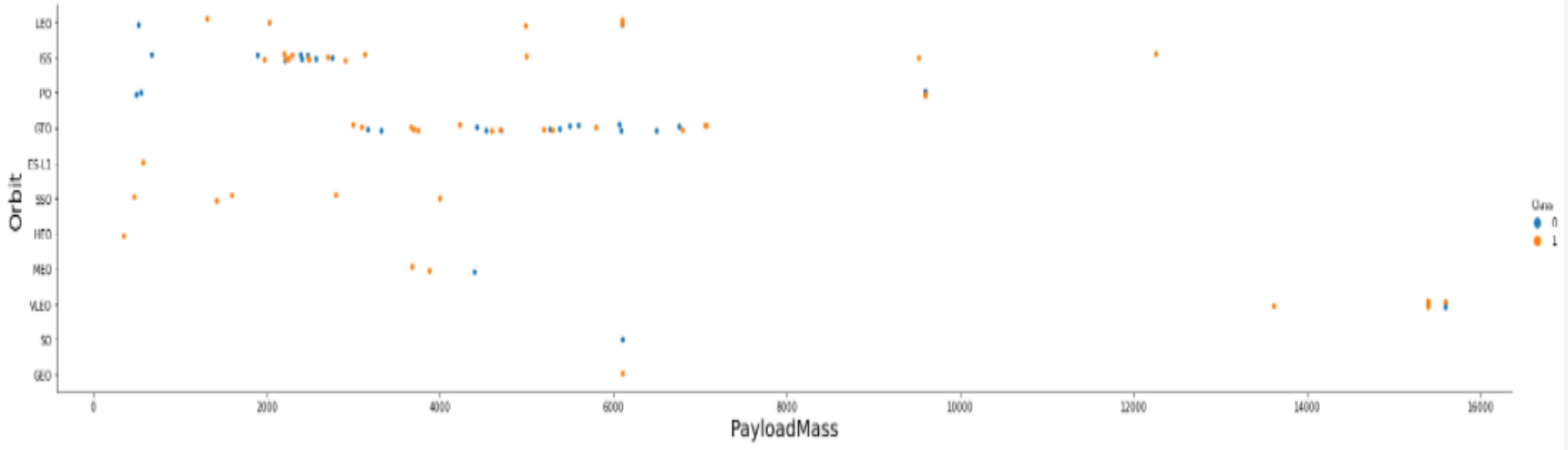
Flight Number vs. Orbit Type

The scatter chat below shows the Flight Number vs. Orbit type. We observe that in the LEO orbit, success is related to the number of flights involved whereas in the GTO, there is no relationship between flight number and the orbit., essential no change with increase on the flight.



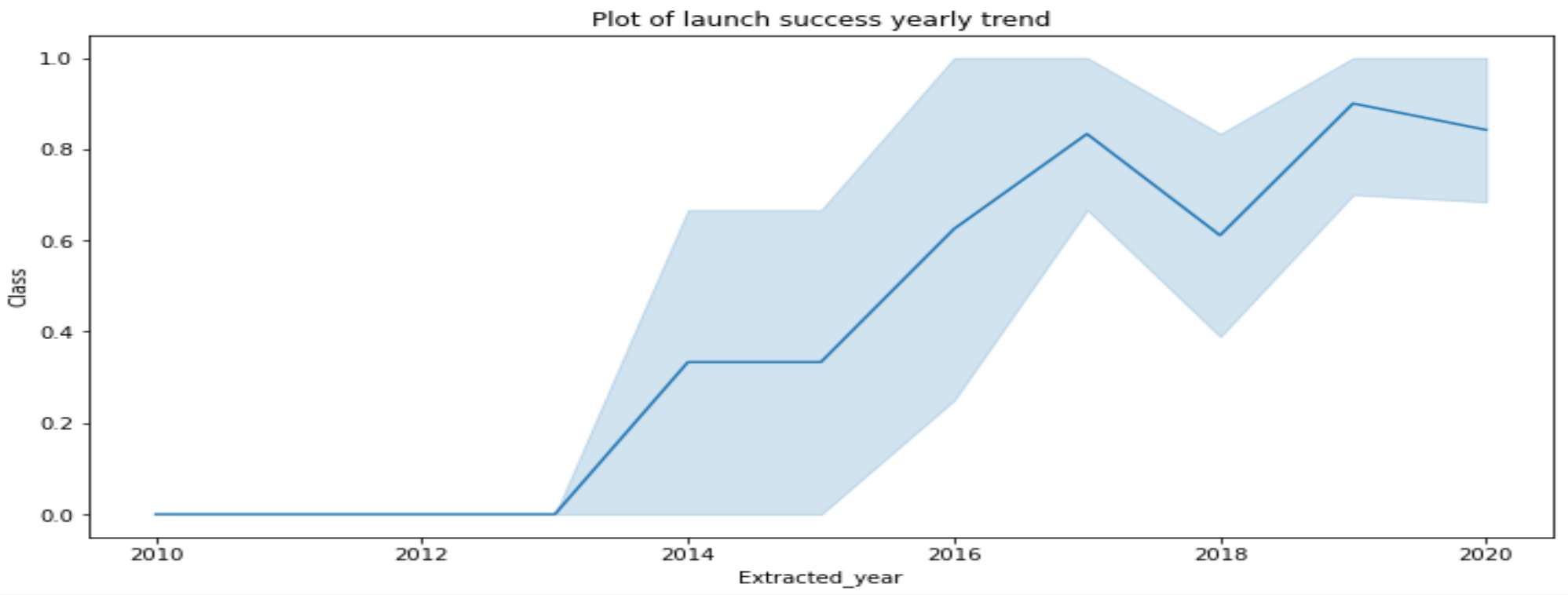
Payload vs. Orbit Type

We did observe that with heavy payload mass, the successful landing are more for PO, LEO and ISS orbits on other orbit there is no direct relationship



Launch Success Yearly Trend

From the chart below, we can see that success rate since 2013 kept on increasing till 2020. This shows there is relationship between outcome with time.



All Launch Site Names

We used the **DISTINCT clause** to show only unique launch sites from the SpaceX data.

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

In [10]:
```
task_1 = '''
        SELECT DISTINCT LaunchSite
        FROM SpaceX
'''
create_pandas_df(task_1, database=conn)
```

Out[10]:

	launchsite
0	KSC LC-39A
1	CCAFS LC-40
2	CCAFS SLC-40
3	VAFB SLC-4E

Launch Site Names Begin with 'CCA'

We used the query above to display 5 records where launch sites begin with `CCA`. The LIMIT 5 clause is what we used to limit the data. And the LIKE is useful for pattern search.

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

In [11]:
```
task_2 = '''
        SELECT *
        FROM SpaceX
        WHERE LaunchSite LIKE 'CCA%'
        LIMIT 5
        '''
create_pandas_df(task_2, database=conn)
```

Out[11]:

	date	time	boosterversion	launchsite	payload	payloadmasskg	orbit	customer	missionoutcome	landingoutcome
0	2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	2010-08-12	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2	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
3	2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	2013-01-03	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 calculated the total payload carried by boosters from NASA as 45596 using the query below. The SUM clause on the payloadmassKg and the where clause is what was needed to achieve this.

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

In [12]:
```
task_3 = '''
        SELECT SUM(PayloadMassKG) AS Total_PayloadMass
        FROM SpaceX
        WHERE Customer LIKE 'NASA (CRS)'
        '''
create_pandas_df(task_3, database=conn)
```

Out[12]:

	total_payloadmass
0	45596

Average Payload Mass by F9 v1.1

We calculated the average payload mass carried by booster version F9 v1.1 and the results as well as the query can be seen below.

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

In [13]:
```
task_4 = '''
        SELECT AVG(PayloadMassKG) AS Avg_PayloadMass
        FROM SpaceX
        WHERE BoosterVersion = 'F9 v1.1'
        '''
create_pandas_df(task_4, database=conn)
```

Out[13]:

	avg_payloadmass
0	2928.4

First Successful Ground Landing Date

We did observe that the dates of the first successful landing outcome on ground pad was 22nd December 2015.. The snapshot can be seen below.

In [14]:
```
task_5 = '''
        SELECT MIN(Date) AS FirstSuccessfull_landing_date
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Success (ground pad)'
        '''
create_pandas_df(task_5, database=conn)
```

Out[14]:

	firstsuccessfull_landing_date
0	2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

We used the **WHERE** clause to filter for boosters which have successfully landed on drone ship and applied the **AND** condition to determine successful landing with payload mass greater than 4000 but less than 6000.. We could have used BETWEEN clause to check for payload mass too.

In [15]:
```
task_6 = '''
        SELECT BoosterVersion
        FROM SpaceX
        WHERE LandingOutcome = 'Success (drone ship)'
            AND PayloadMassKG > 4000
            AND PayloadMassKG < 6000
        '''
create_pandas_df(task_6, database=conn)
```

Out[15]:

	boosterversion
0	F9 FT B1022
1	F9 FT B1026
2	F9 FT B1021.2
3	F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

We used pattern search like keyword with '%' to filter for successful or a failure Mission Outcome .

List the total number of successful and failure mission outcomes

In [16]:
```
task_7a = '''
          SELECT COUNT(MissionOutcome) AS SuccessOutcome
          FROM SpaceX
          WHERE MissionOutcome LIKE 'Success%'
          '''

task_7b = '''
          SELECT COUNT(MissionOutcome) AS FailureOutcome
          FROM SpaceX
          WHERE MissionOutcome LIKE 'Failure%'
          '''
print('The total number of successful mission outcome is:')
display(create_pandas_df(task_7a, database=conn))
print()
print('The total number of failed mission outcome is:')
create_pandas_df(task_7b, database=conn)
```

Out[16]:

The total number of successful mission outcome is:

	successoutcome
0	100

The total number of failed mission outcome is:

	failureoutcome
0	1

Boosters Carried Maximum Payload

We determined the booster that have carried the maximum payload using a subquery in the **WHERE** clause and the **MAX** function. And we sorted the results accordingly.

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

In [17]:

```
task_8 = '''
        SELECT BoosterVersion, PayloadMassKG
        FROM SpaceX
        WHERE PayloadMassKG = (
                                SELECT MAX(PayloadMassKG)
                                FROM SpaceX
                                )
        ORDER BY BoosterVersion
        '''
create_pandas_df(task_8, database=conn)
```

Out[17]:

	boosterversion	payloadmasskg
0	F9 B5 B1048.4	15600
1	F9 B5 B1048.5	15600
2	F9 B5 B1049.4	15600
3	F9 B5 B1049.5	15600
4	F9 B5 B1049.7	15600
5	F9 B5 B1051.3	15600
6	F9 B5 B1051.4	15600
7	F9 B5 B1051.6	15600
8	F9 B5 B1056.4	15600
9	F9 B5 B1058.3	15600
10	F9 B5 B1060.2	15600
11	F9 B5 B1060.3	15600

2015 Launch Records

We used a combinations of the **WHERE** clause, **LIKE**, **AND**, and **BETWEEN** conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015. We could have used the YEAR function too to accomplish the same.

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

In [18]:
```
task_9 = '''
        SELECT BoosterVersion, LaunchSite, LandingOutcome
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Failure (drone ship)'
            AND Date BETWEEN '2015-01-01' AND '2015-12-31'
        '''
create_pandas_df(task_9, database=conn)
```

Out[18]:

	boosterversion	launchsite	landingoutcome
0	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
1	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

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

We selected Landing outcomes and the **COUNT** of landing outcomes from the data and used the **WHERE** clause to filter for landing outcomes **BETWEEN** 2010-06-04 to 2010-03-20.

We applied the **GROUP BY** clause to group the landing outcomes and the **ORDER BY** clause to order the grouped landing outcome in descending order.

The Count function return the "count" nae as default, we could have changed it.

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad))

In [19]:
```
task_10 = '''
        SELECT LandingOutcome, COUNT(LandingOutcome)
        FROM SpaceX
        WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
        GROUP BY LandingOutcome
        ORDER BY COUNT(LandingOutcome) DESC
        '''
create_pandas_df(task_10, database=conn)
```

Out[19]:

	landingoutcome	count
0	No attempt	10
1	Success (drone ship)	6
2	Failure (drone ship)	5
3	Success (ground pad)	5
4	Controlled (ocean)	3
5	Uncontrolled (ocean)	2
6	Precluded (drone ship)	1
7	Failure (parachute)	1

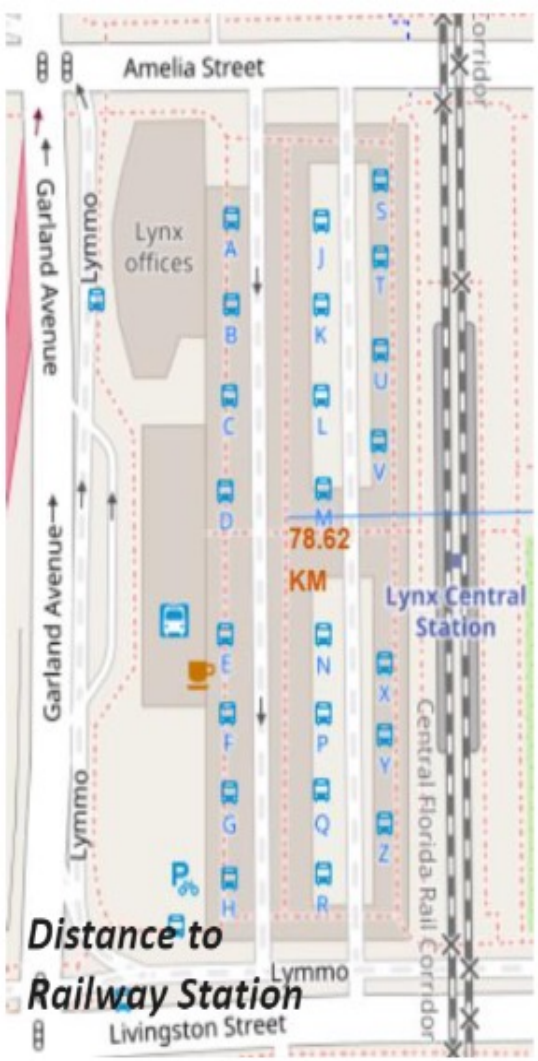

Section 3
Launch Sites
Proximities Analysis

All launch sites global map markers

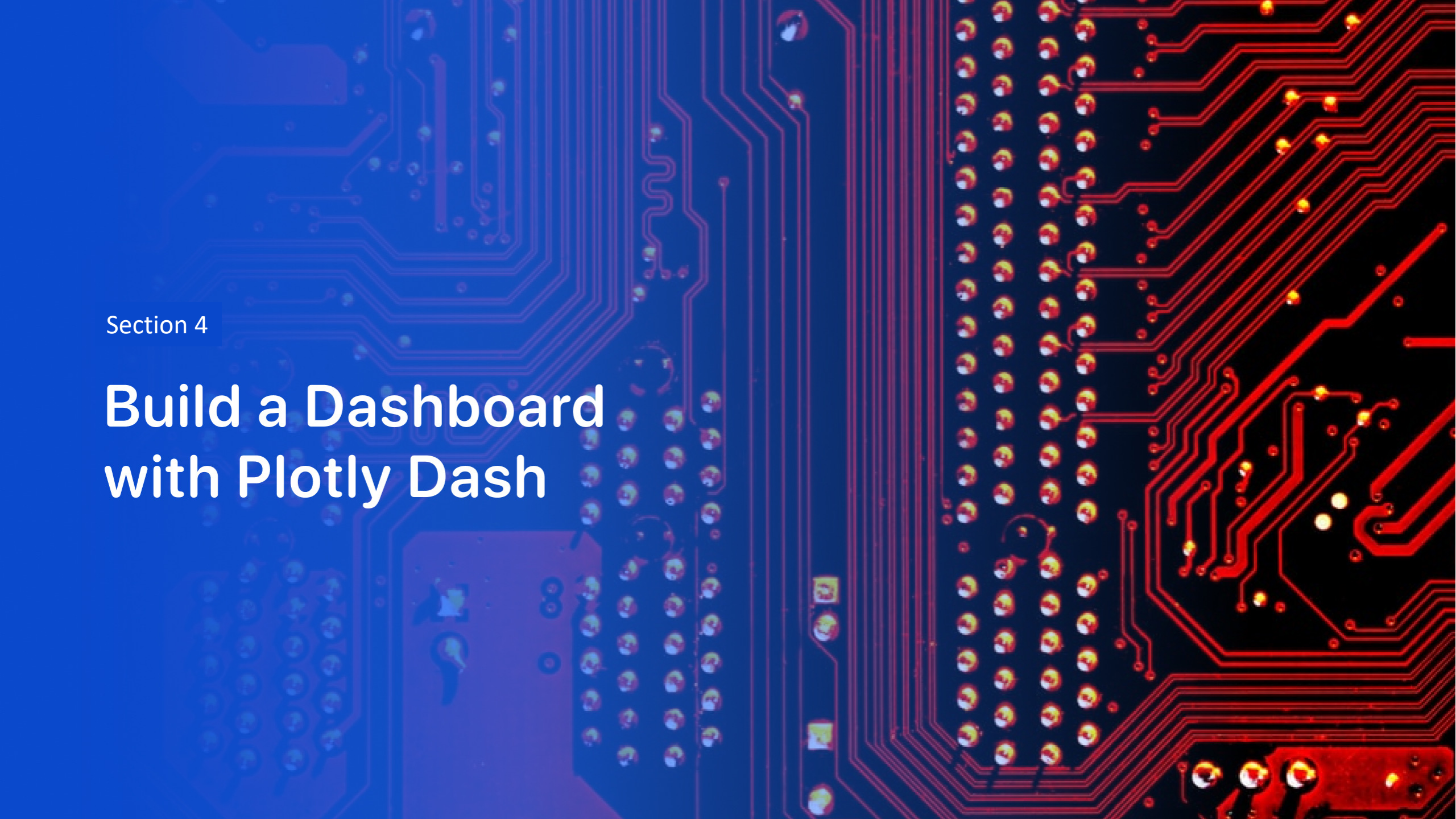
We can see that the SpaceX launch sites are in the United States of America coasts. Florida and California

Markers showing launch sites with color labels

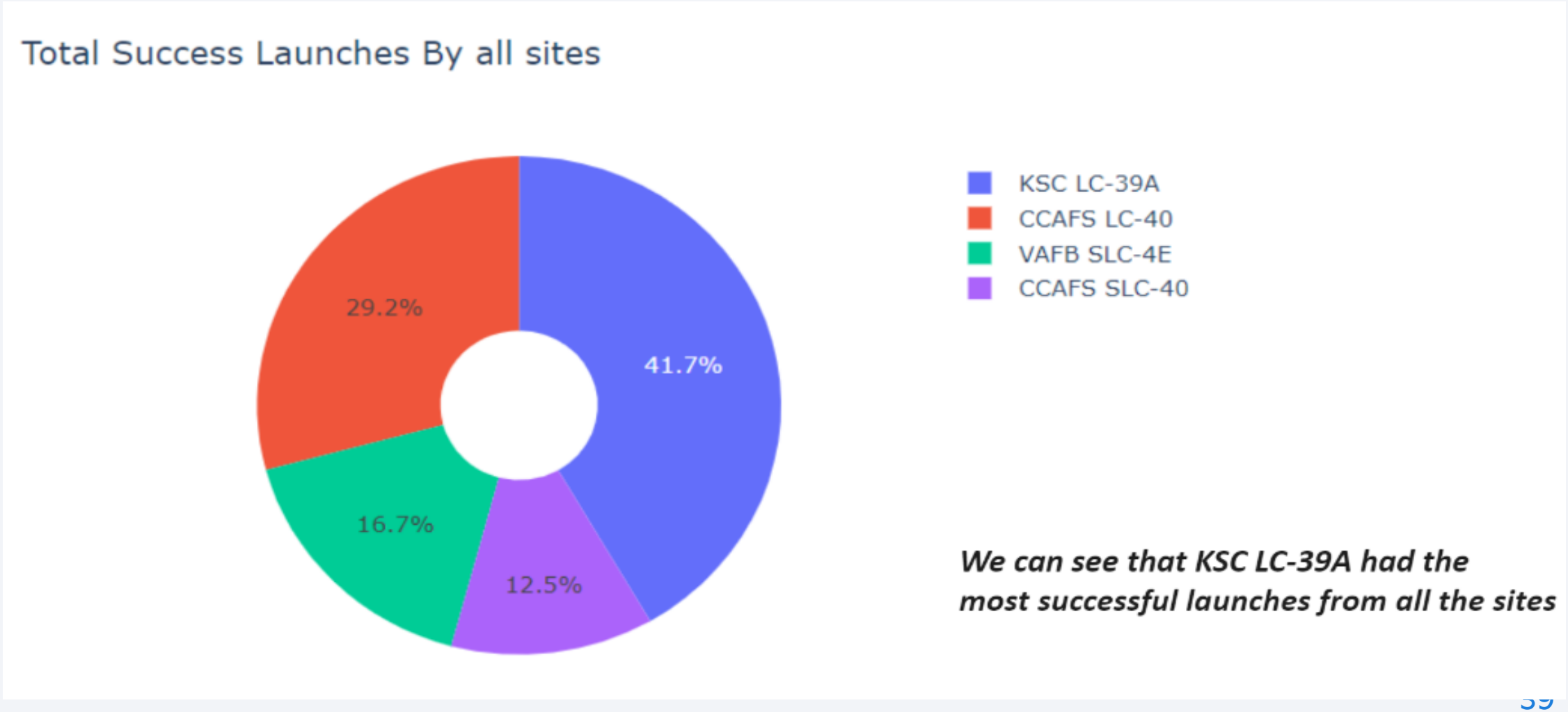
Launch Site distance to landmarks



- 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

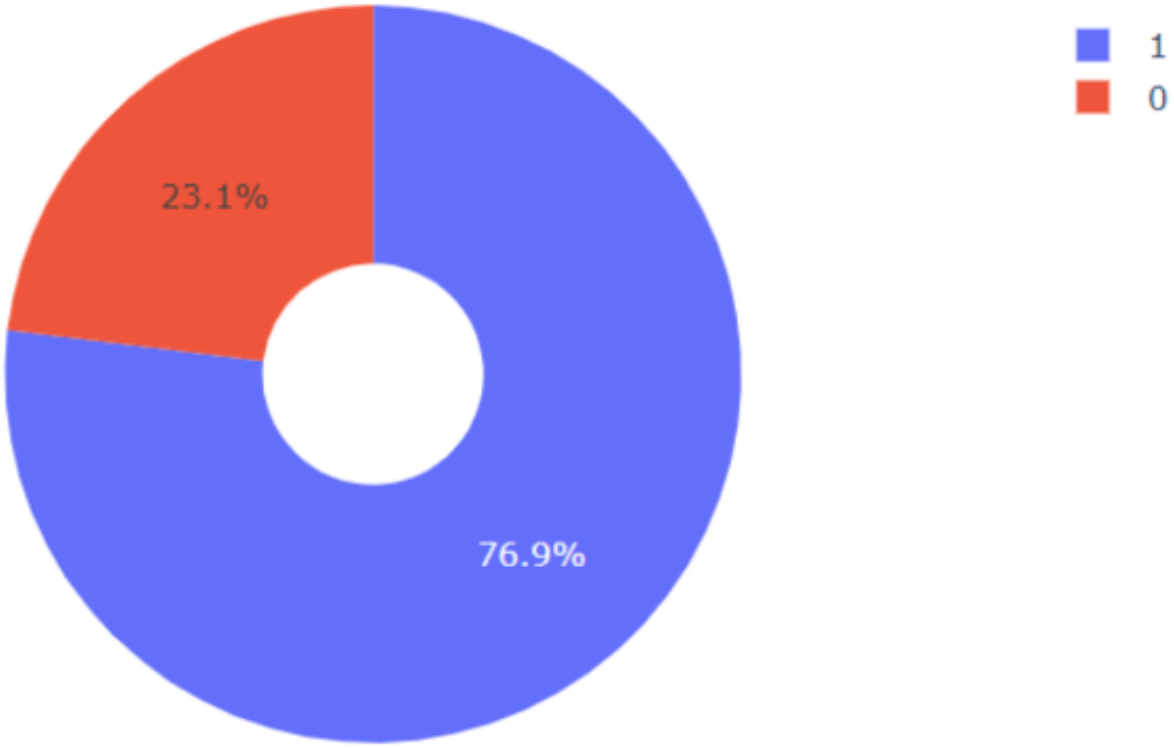

Section 4
Build a Dashboard with Plotly Dash

Pie Chart showing the success rate by Launch Site



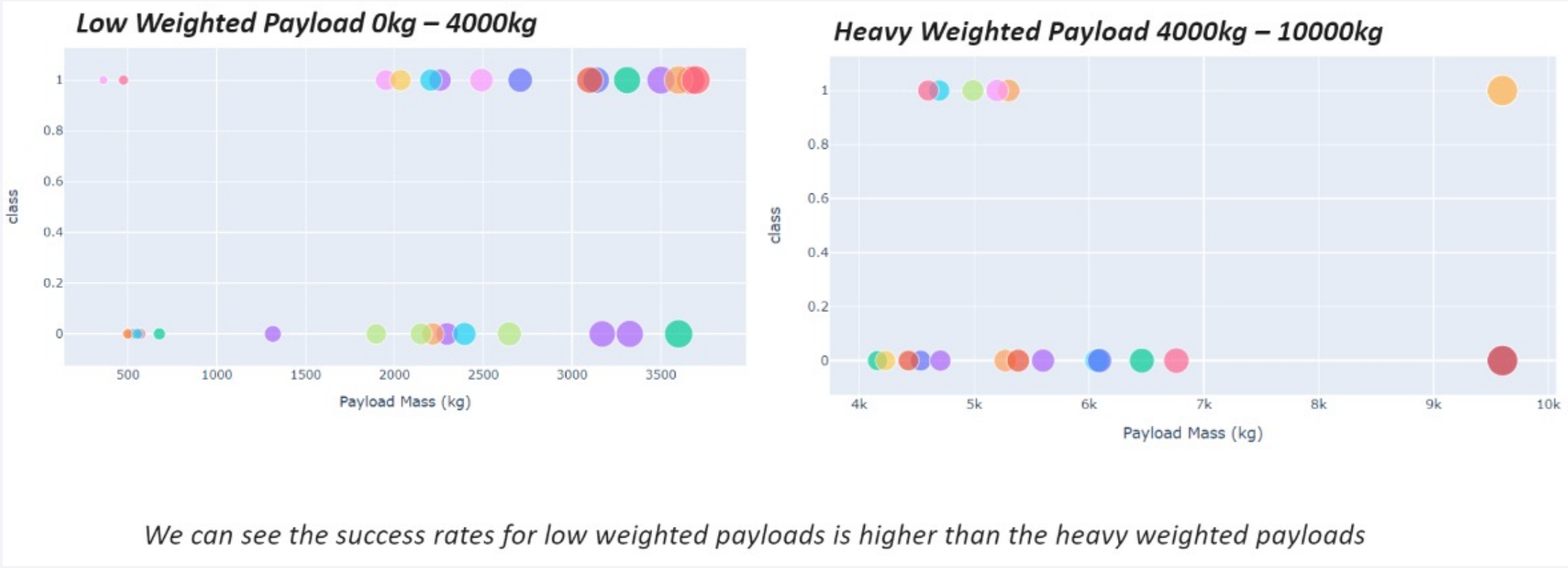
We can see that KSC LC-39A had the most successful launches from all the sites

Pie chart showing the Launch site with the highest success ratio



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

Scatter plot of Payload vs Launch Outcome 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

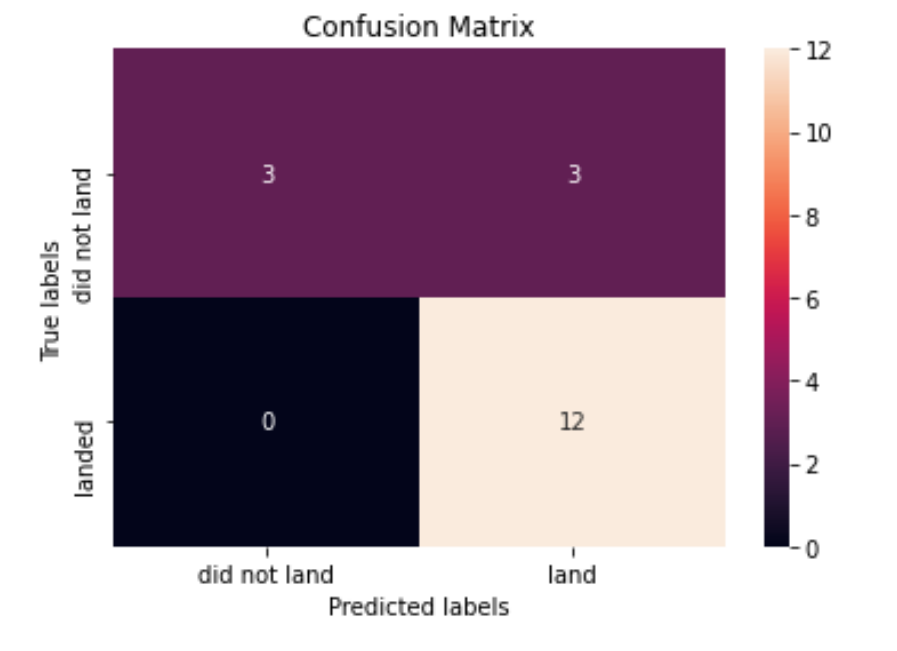
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)
```

```
Best model is DecisionTree with a score of 0.8732142857142856
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```

The Decision Tree classifier is the model with the highest classification accuracy

Confusion Matrix



The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes. The major problem is the false positives .i.e., 3 unsuccessful landing marked as successful landing by the classifier. While this is small number but it is half of the testing data.

Conclusions

With all the analysis that have been done, we can essentially conclude that :

- The larger the flight amount at a launch site, the greater the success rate at a launch site.
- Launch success rate started to increase in 2013 till 2020.
- Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
- KSC LC-39A site, had the most successful launches of all.
- The Decision Tree classifier is the best machine learning algorithm that can be deployed on this ask.

Thank you!