IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Harsh Rathod
20/05/2023

Outline

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

Executive Summary

Summary of methodologies

This project follows these steps:

- Data Collection
- Data Wrangling
- Exploratory Data Analysis
- Interactive Visual Analytics
- Predictive Analysis (Classification)

Summary of all results

This project produced the following outputs and visualizations:

1. Exploratory Data Analysis (EDA) results
2. Geospatial analytics
3. Interactive dashboard
4. Predictive analysis of classification models

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 data fields for ML and Dropping irrelevant columns.
- Perform exploratory data analysis (EDA) using visualization and SQL
  - Scatter and bar graphs to show patterns between data
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Build and evaluate classification models

Data Collection

- The data was collected using various methods
  - Data collection was done using get request 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 .json_normalize().
  - We then cleaned the data, checked for missing values and fill in missing values where necessary.
  - In addition, we performed web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.
  - 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. We then cleaned the data by removing any errors or inconsistencies. Finally, we did some basic data wrangling and formatting to make the data easier to use.
- https://github.com/harshhin/IBM-Data-Science-Capstone/blob/main/Data%20Collection%20API.ipynb

Now let's start requesting rocket launch data from SpaceX API with the following URL:

```
In [6]: spacex_url="https://api.spacexdata.com/v4/launches/past"
```

```
In [7]: response = requests.get(spacex_url)
```

Check the content of the response

```
In [8]: print(response.content)
```

Data Collection - Scraping

- We applied web scrapping to webscrap Falcon 9 launch records with BeautifulSoup
- We parsed the table and converted it into a pandas dataframe
- https://github.com/harshhin/IBM-Data-Science-Capstone/blob/main/Data%20Collection%20with%20Web%20Scraping.ipynb

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

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

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

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
!pip install html5lib
!pip install lxml

from bs4 import BeautifulSoup

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

```
Requirement already satisfied: html5lib in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (1.1)
Requirement already satisfied: six>=1.9 in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from html5lib) (1.1
6.0)
Requirement already satisfied: webencodings in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (from html5lib)
(0.5.1)
Requirement already satisfied: lxml in /home/jupyterlab/conda/envs/python/lib/python3.7/site-packages (4.9.2)
```

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

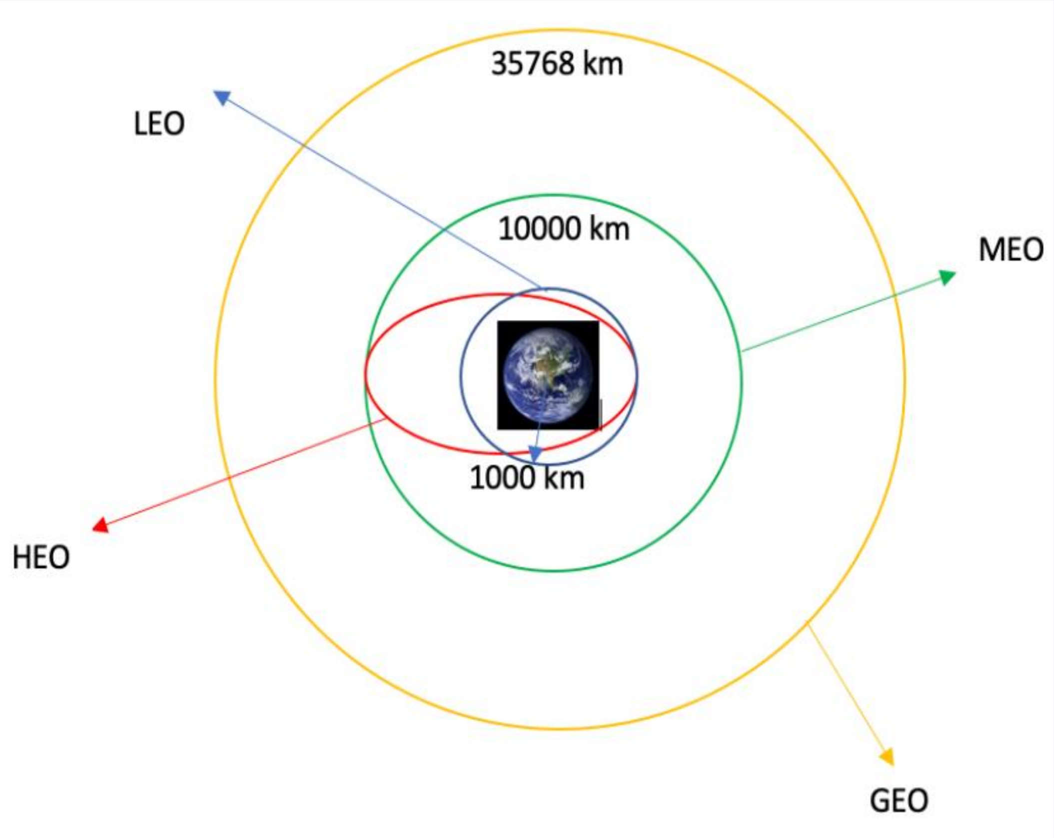
```
# Use soup.title attribute
print(soup.title)
```

```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

Data Wrangling

- We conducted exploratory data analysis and derived the training labels based on the outcomes. We analyzed the frequency of launches at each site as well as the number and occurrence of different orbits. We transformed the outcome column into a landing outcome label and exported the results to a CSV file.

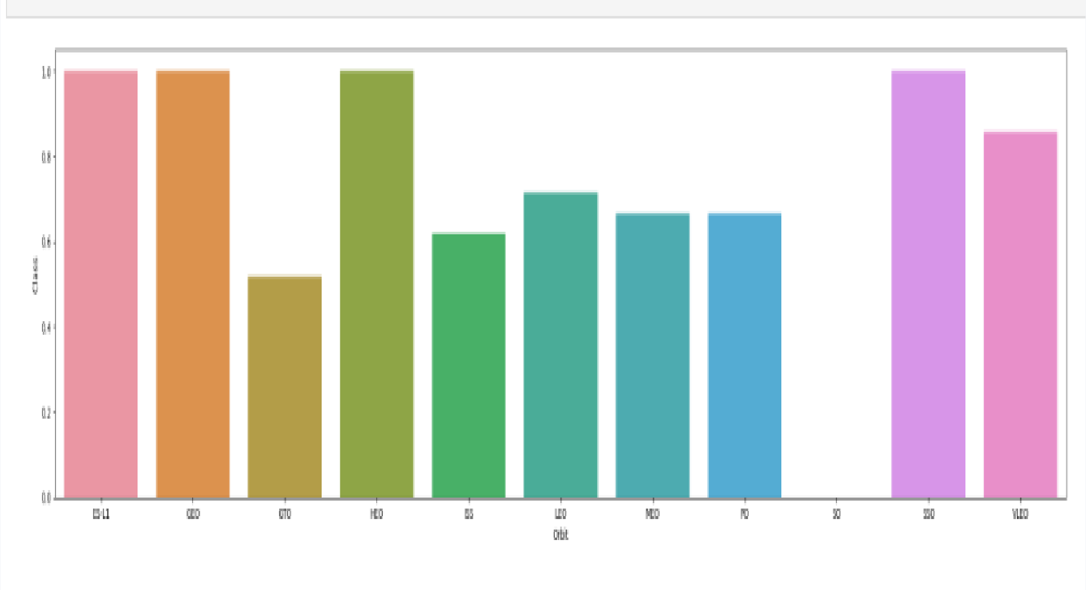
- https://github.com/harshhin/IBM-Data-Science-Capstone/blob/main/Data%20Wrangling.ipynb



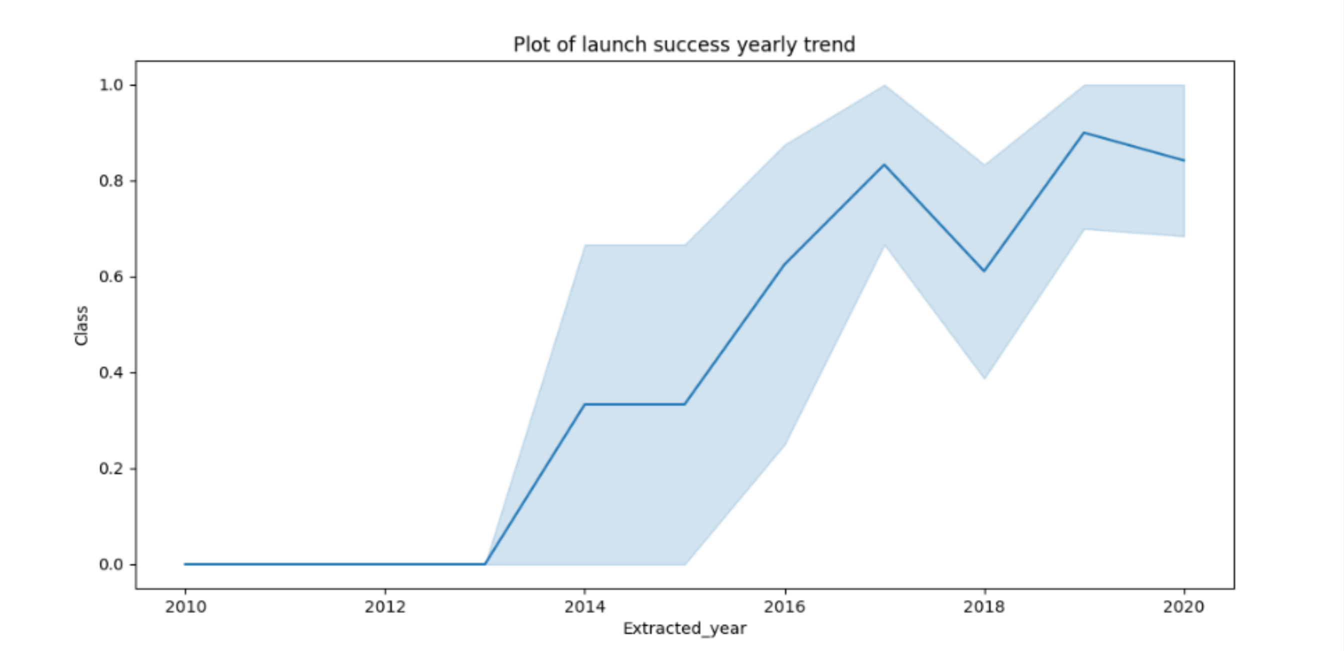
EDA with Data Visualization

- https://github.com/harshhin/IBM-Data-Science-Capstone/blob/main/EDA%20with%20Data%20Visualization.ipynb

Bar Graph



Line Drawn



EDA with SQL

- The SpaceX dataset was loaded into a PostgreSQL database seamlessly within the Jupyter Notebook environment. Through the utilization of SQL queries, exploratory data analysis (EDA) was performed to extract valuable insights from the data. The following specific queries were executed to unveil important information:
  - The names of unique launch sites in the space mission.
  - The total payload mass carried by boosters launched by NASA (CRS)
  - The average payload mass carried by booster version F9 v1.1
  - The 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 https://github.com/harshhin/IBM-Data-Science-Capstone/blob/main/SQL%20Notebook%20for%20Peer%20Assignment.ipynb

Build an Interactive Map with Folium

- Launch site mapping: All launch sites were marked on a Folium map, and map objects like markers, circles, and lines were added to represent the success or failure of launches at each site.
- Classifying launch outcomes: The launch outcomes, categorized as failure or success, were assigned class labels of 0 and 1, respectively. This classification allowed for easier analysis and visualization of the data.
- Color-labeled marker clusters: Marker clusters on the map were color-coded based on the launch outcomes (success or failure). This visualization technique helped identify launch sites with relatively high success rates by observing the clustering patterns.
- Calculation of distances: Distances between a launch site and its neighboring locations were calculated. This allowed for further analysis and exploration of spatial relationships between launch sites and their proximities.

https://github.com/harshhin/IBM-Data-Science-Capstone/blob/main/Interactive%20Visual%20Analytics%20with%20Folium%20lab.ipynb

Build a Dashboard with Plotly Dash

- An interactive dashboard was created using Plotly Dash to visualize the data. The dashboard includes the following plots:
- Pie charts: Pie charts were used to display the total launches for specific launch sites, providing a visual representation of the distribution of launches across different sites.
- Scatter graph: A scatter graph was plotted to examine the relationship between the launch outcome and payload mass (in kilograms) for different booster versions. This plot allows for the identification of any patterns or trends between the outcome and payload mass.

GitHub link is https://github.com/harshhin/IBM-Data-Science-Capstone/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

- The data was loaded into the notebook using the numpy and pandas libraries. Afterward, the data was transformed, and a split was performed to separate it into training and testing sets.
- Different machine learning models were built, and hyperparameters were tuned using GridSearchCV, a method for systematically searching the hyperparameter space. The accuracy metric was utilized to evaluate the performance of the models.
- To enhance the models, feature engineering techniques and algorithm tuning were employed. These steps aimed to improve the predictive capabilities of the models by selecting relevant features and optimizing the algorithms' settings.
- Through the evaluation process, the best performing classification model was identified based on its accuracy score.

GitHub link is https://github.com/harshhin/IBM-Data-Science-Capstone/blob/main/Machine%20Learning%20Prediction%20lab.ipynb

Results

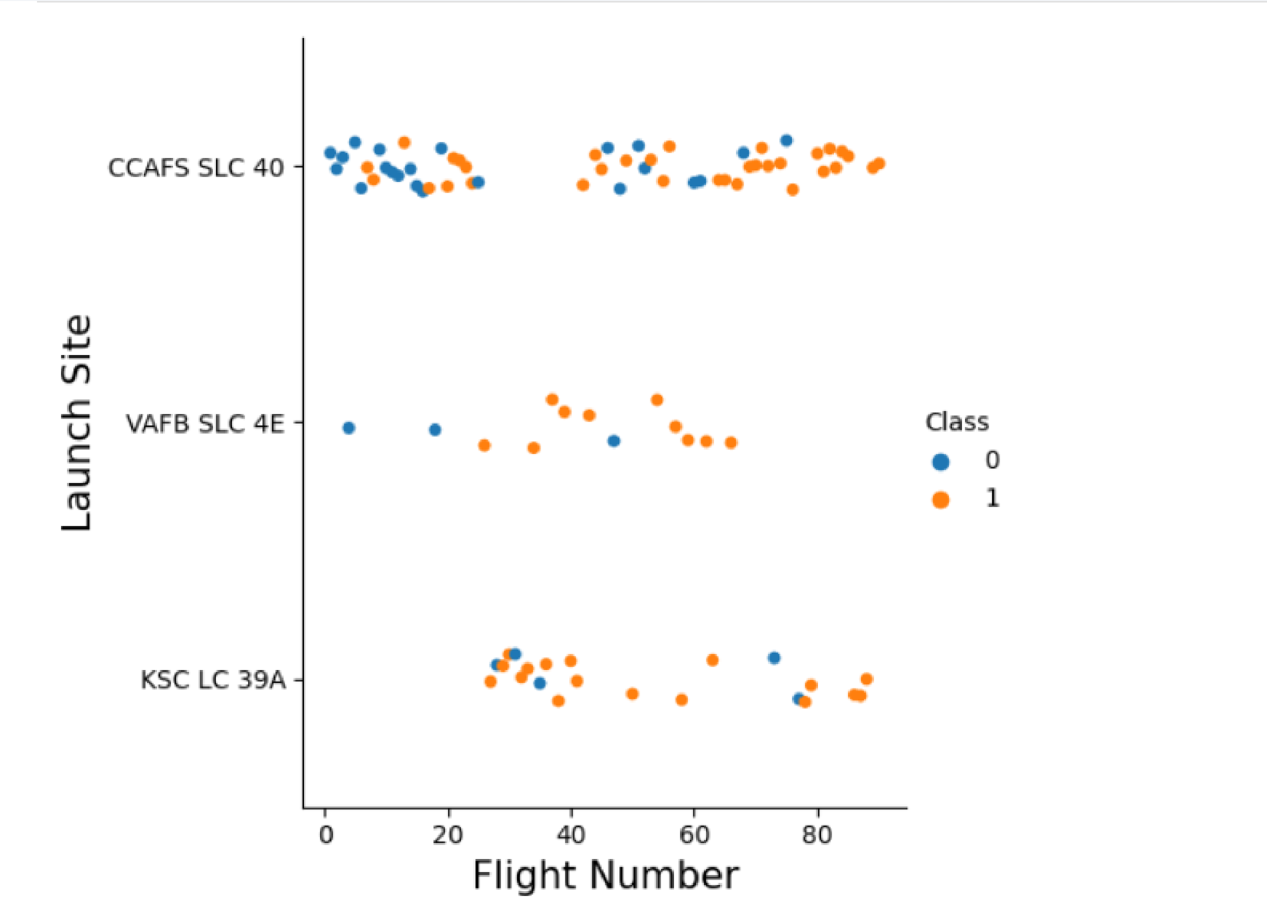
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

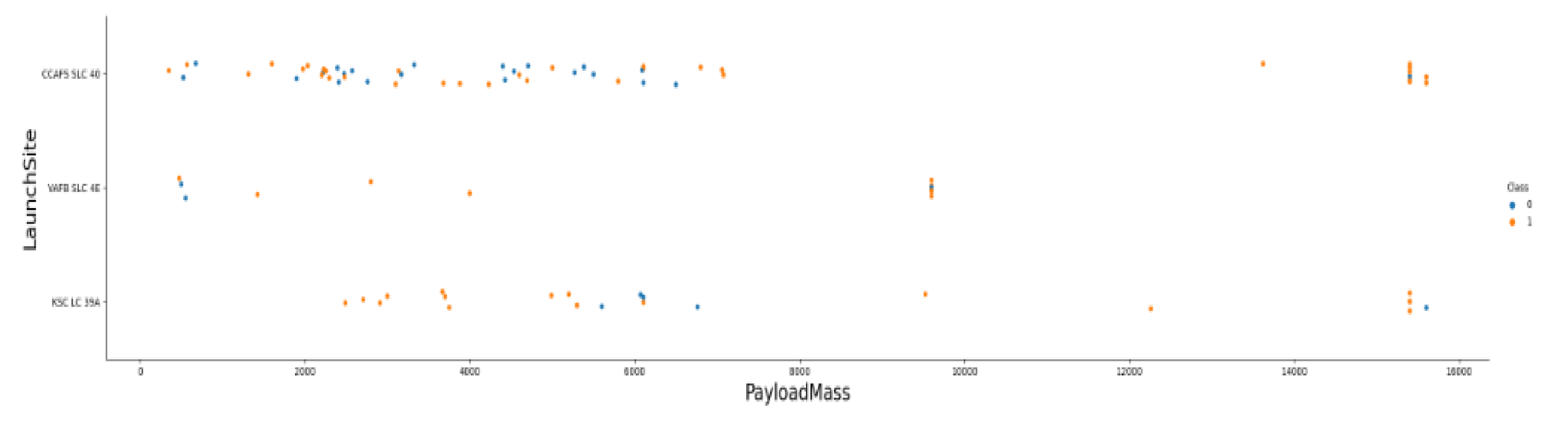
Flight Number vs. Launch Site

- **Number vs. Launch Site**



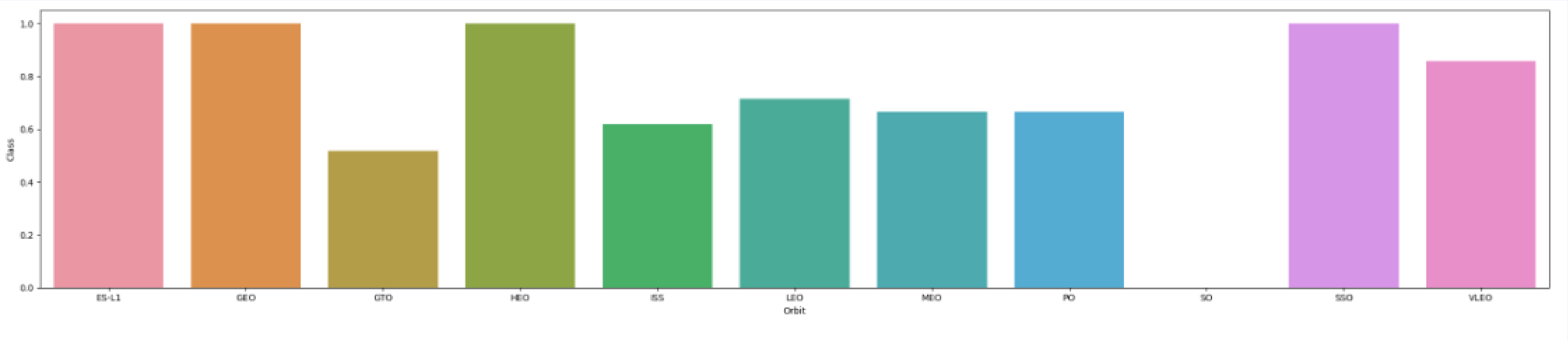
Payload vs. Launch Site

- **Payload vs. Launch Site**



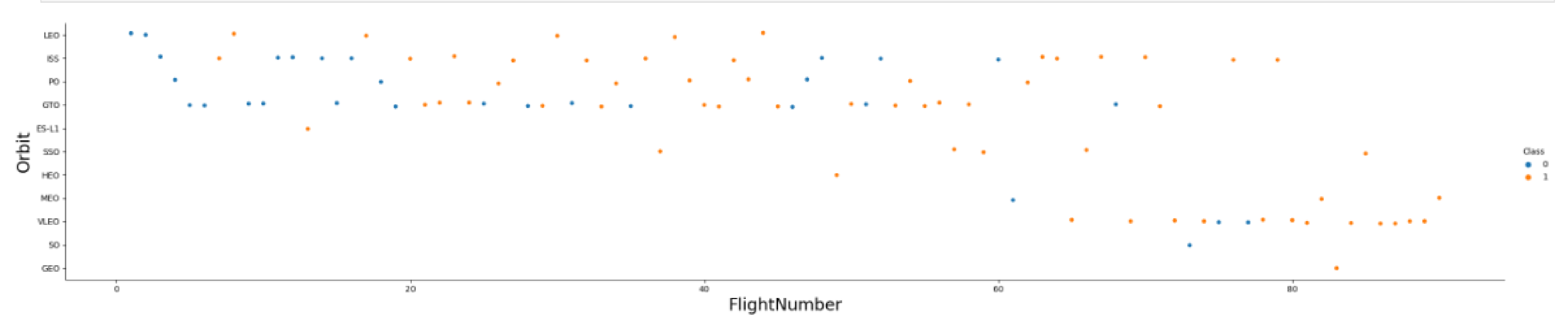
Success Rate vs. Orbit Type

- From the plot we can see Success rate of all type of Orbit.



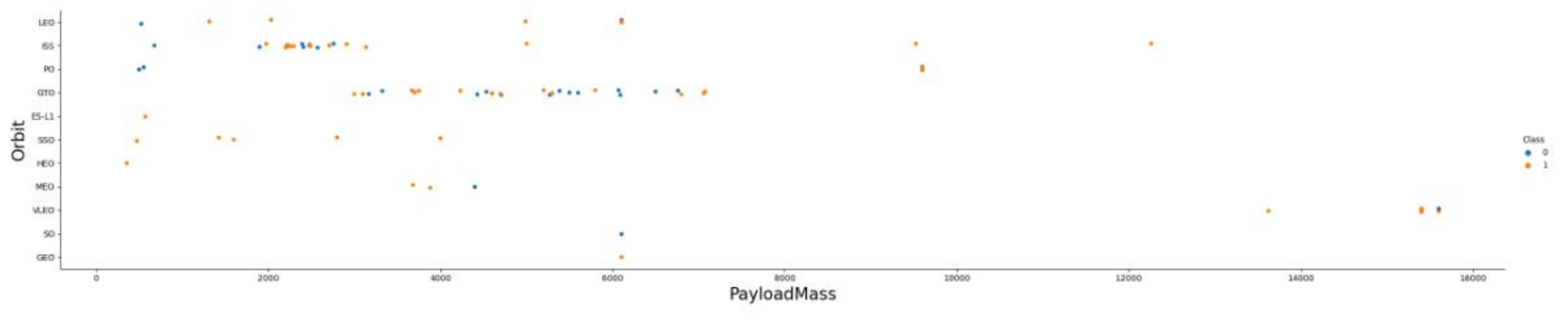
Flight Number vs. Orbit Type

- The plot below shows the Flight Number vs. Orbit type. We observe that in the LEO orbit, success is related to the number of flights whereas in the GTO orbit, there is no relationship between flight number and the orbit.



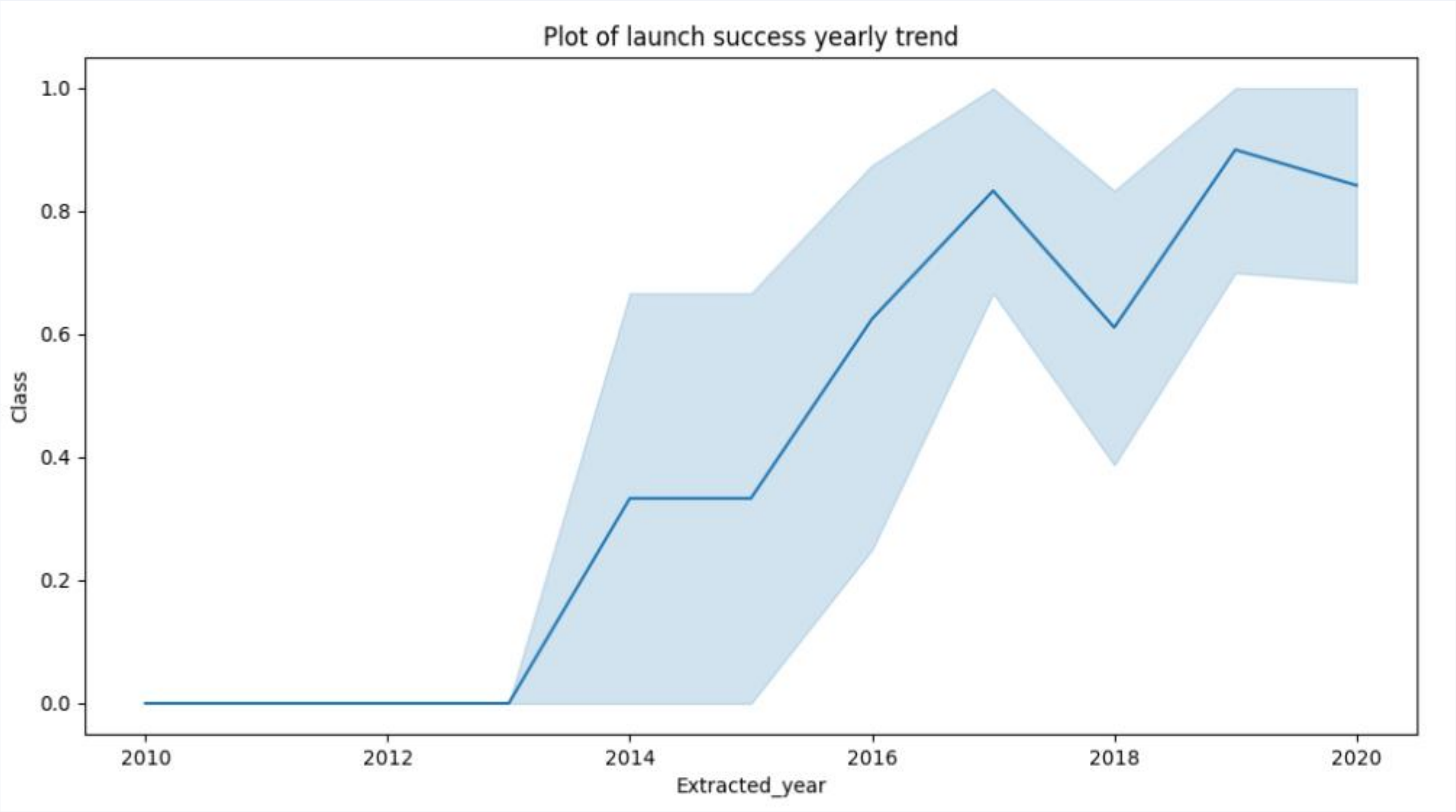
Payload vs. Orbit Type

- We can observe that with heavy payloads, the successful landing are more for PO, LEO and ISS orbits.



Launch Success Yearly Trend

- From the plot, we can observe that success rate since 2013 kept on increasing till 2020. While there is slightly Decline between 2017 to 2018.



All Launch Site Names

- We used the key word **DISTINCT** 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 below to display 5 records where launch sites begin with `CCA`

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

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 as 2928.4

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 observed that the dates of the first successful landing outcome on ground pad was 22nd December 2015

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

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 wildcard like '%' to filter for **WHERE** MissionOutcome was a success or a failure.

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)
```

```
The total number of successful mission outcome is:
```

	successoutcome
0	100

```
The total number of failed mission outcome is:
```

Out[16]:

	failureoutcome
0	1

Boosters Carried Maximum Payload

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

- We determined the booster that have carried the maximum payload using a subquery in the **WHERE** clause and the **MAX()** function.

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

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

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

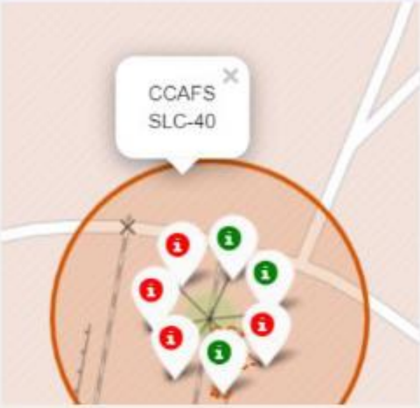
- Selection: Landing outcomes and the count of landing outcomes were chosen from the dataset.
- Filtering: The WHERE clause was used to filter the landing outcomes for the period between 2010-06-04 and 2010-03-20.
- Grouping: The GROUP BY clause was applied to group the landing outcomes based on their values.
- Ordering: The ORDER BY clause was used to sort the grouped landing outcomes in descending order.

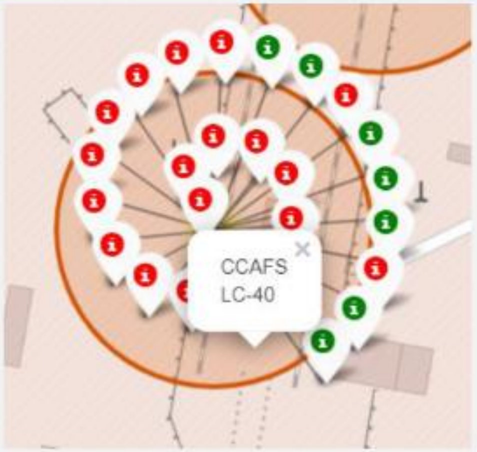
Section 3

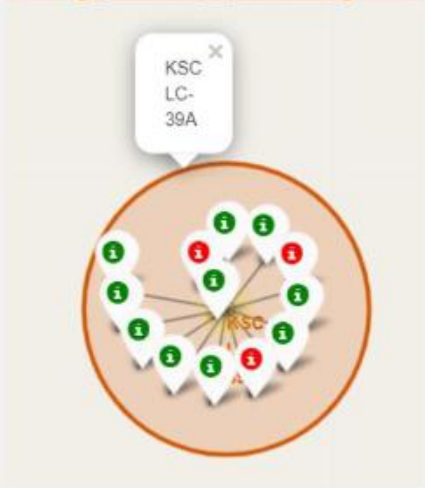
Launch Sites Proximities Analysis

All launch sites global map markers

Markers showing launch sites with color labels







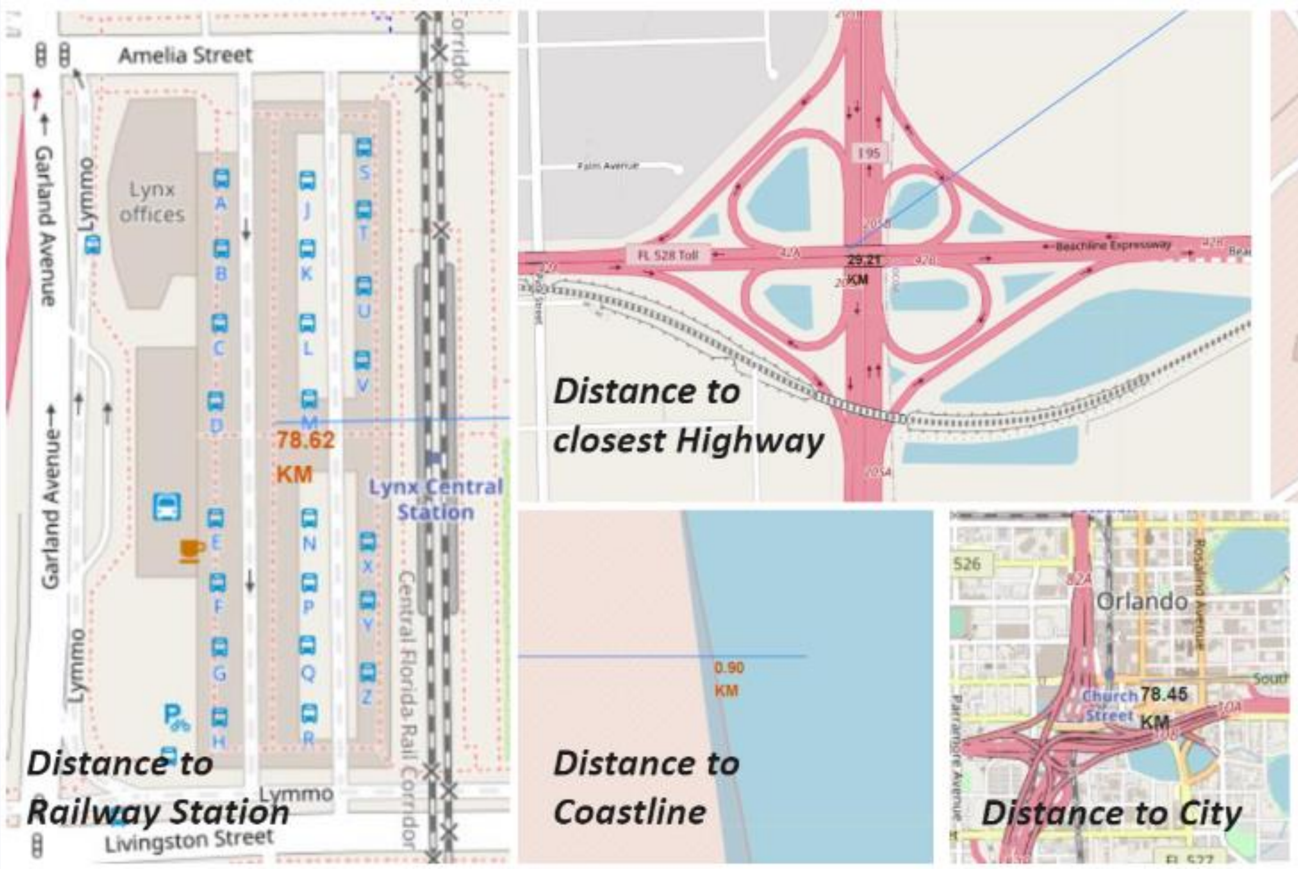
Florida Launch Sites

Green Marker shows successful Launches and Red Marker shows Failures

California Launch Site

37

Launch Site distance to landmarks

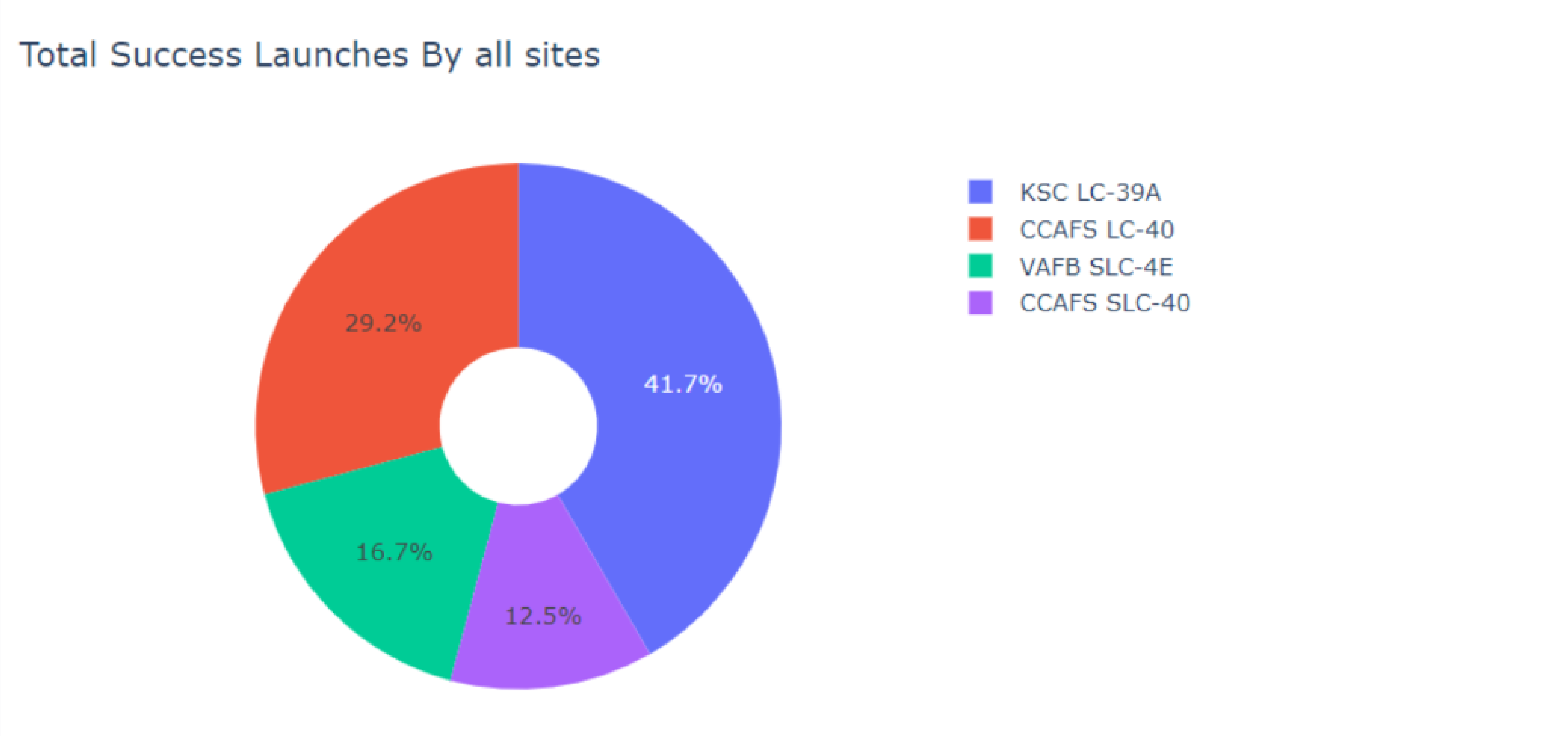


•Are launch sites in close proximity to railways? No
•Are launch sites in close proximity to highways? No
•Are launch sites in close proximity to coastline? Yes
•Do launch sites keep certain distance away from cities? Yes

Section 4

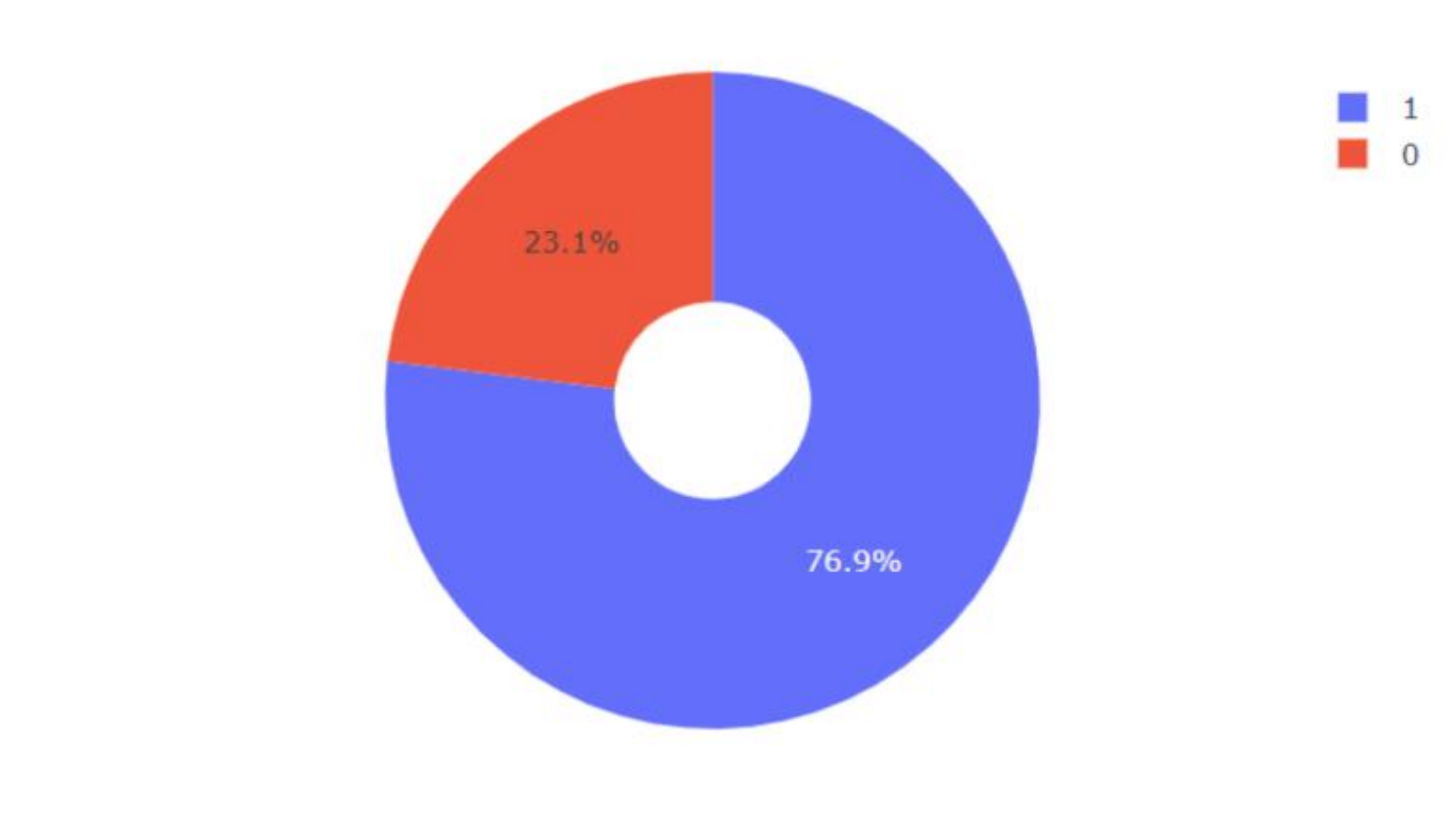
Build a Dashboard with Plotly Dash

Launch success count for all sites, in a pie chart



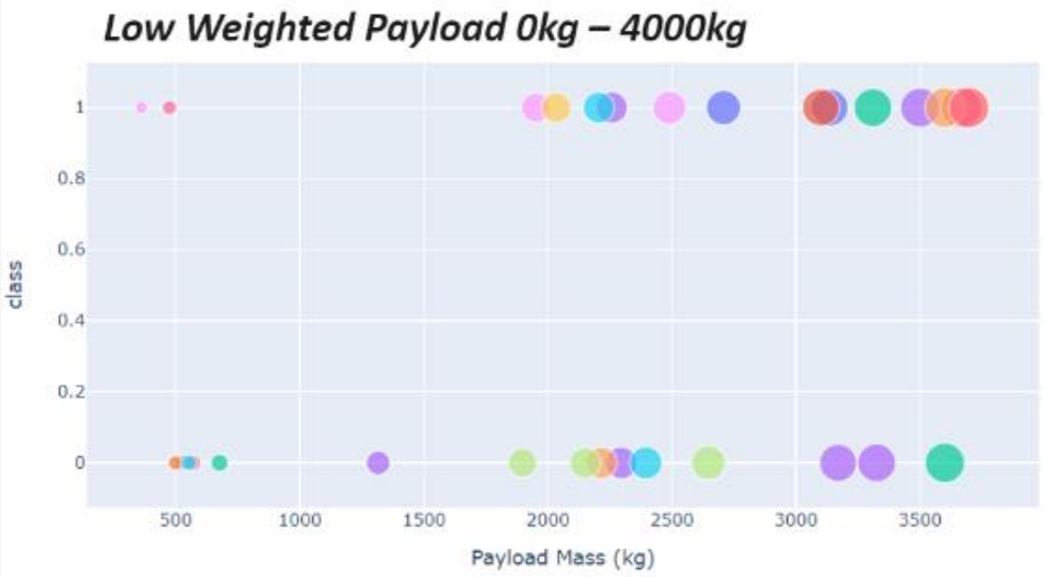
We can see that KSC LC-39A had the most successful launches from all the sites

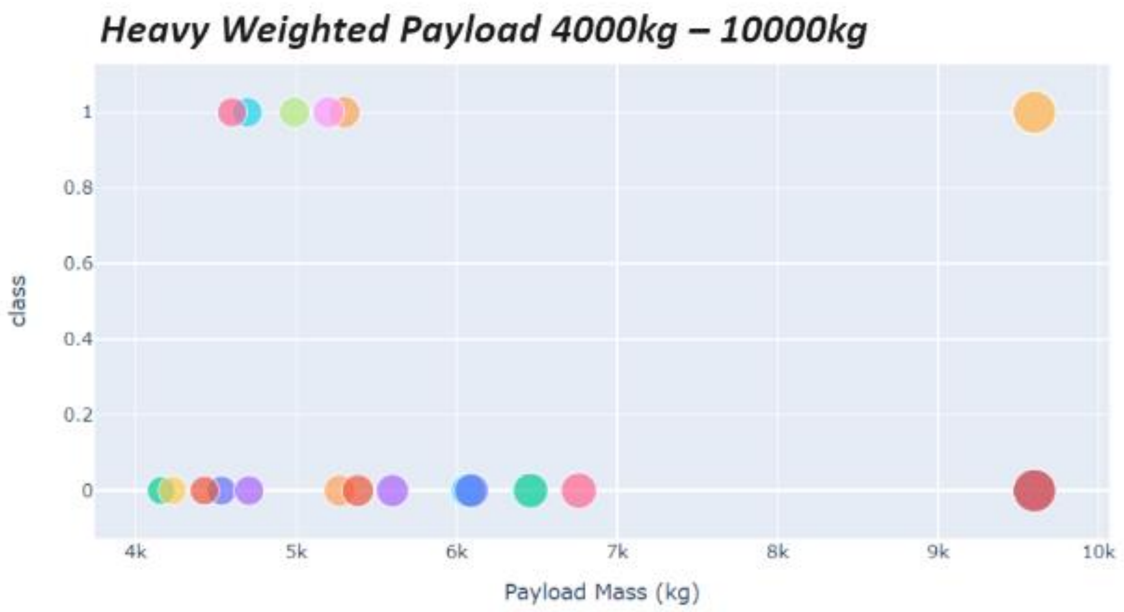
Pie chart showing the Launch site with the highest launch success ratio



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

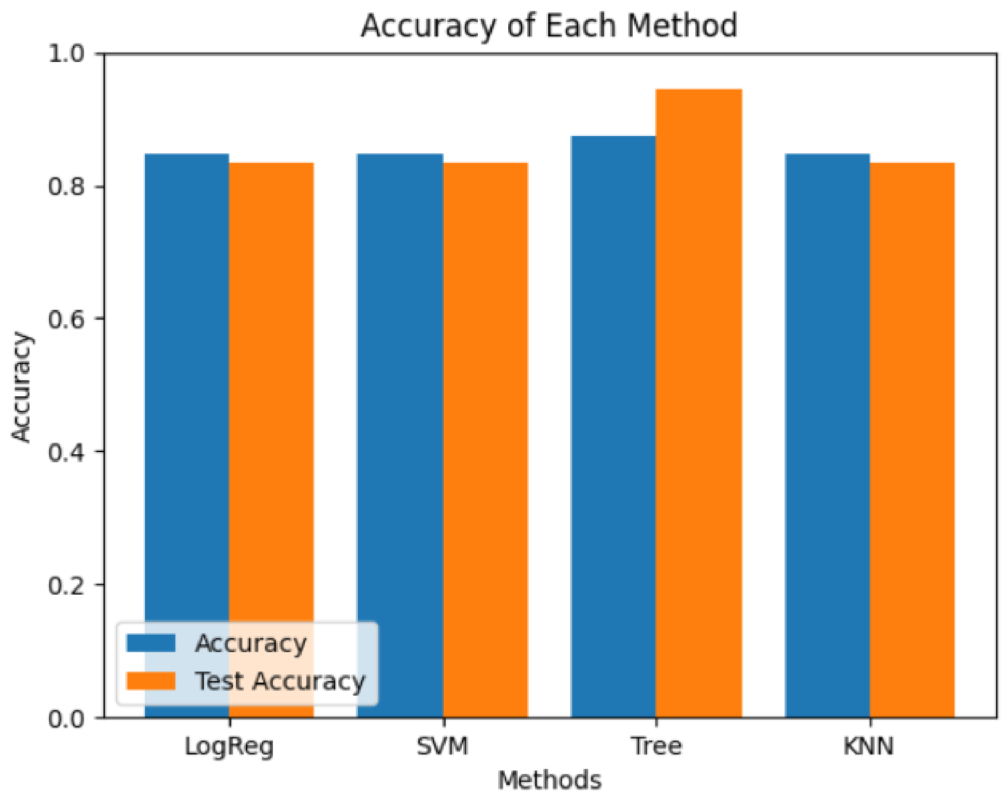
In [36]:

```
print("Model\t\tAccuracy\tTestAccuracy")#,logreg_cv.best_score_)
print("LogReg\t\t{}\t\t{}".format((logreg_cv.best_score_).round(5), logreg_cv.score(X_test, Y_test).round(5)))
print("SVM\t\t{}\t\t{}".format((svm_cv.best_score_).round(5), svm_cv.score(X_test, Y_test).round(5)))
print("Tree\t\t{}\t\t{}".format((tree_cv.best_score_).round(5), tree_cv.score(X_test, Y_test).round(5)))
print("KNN\t\t{}\t\t{}".format((knn_cv.best_score_).round(5), knn_cv.score(X_test, Y_test).round(5)))

comparison = {}

comparison['LogReg'] = {'Accuracy': logreg_cv.best_score_.round(5), 'TestAccuracy': logreg_cv.score(X_test, Y_test).round(5)
comparison['SVM'] = {'Accuracy': svm_cv.best_score_.round(5), 'TestAccuracy': svm_cv.score(X_test, Y_test).round(5)}
comparison['Tree'] = {'Accuracy': tree_cv.best_score_.round(5), 'TestAccuracy': tree_cv.score(X_test, Y_test).round(5)}
comparison['KNN'] = {'Accuracy': knn_cv.best_score_.round(5), 'TestAccuracy': knn_cv.score(X_test, Y_test).round(5)}
x = []
y1 = []
y2 = []
for meth in comparison.keys():
    x.append(meth)
    y1.append(comparison[meth]['Accuracy'])
    y2.append(comparison[meth]['TestAccuracy'])


x_axis = np.arange(len(x))

plt.bar(x_axis - 0.2, y1, 0.4, label = 'Accuracy')
plt.bar(x_axis + 0.2, y2, 0.4, label = 'Test Accuracy')

plt.ylim([0,1])
plt.xticks(x_axis, x)

plt.xlabel("Methods")
plt.ylabel("Accuracy")
plt.title("Accuracy of Each Method")
plt.legend(loc='lower left')
plt.show()
```

Result is on next slide

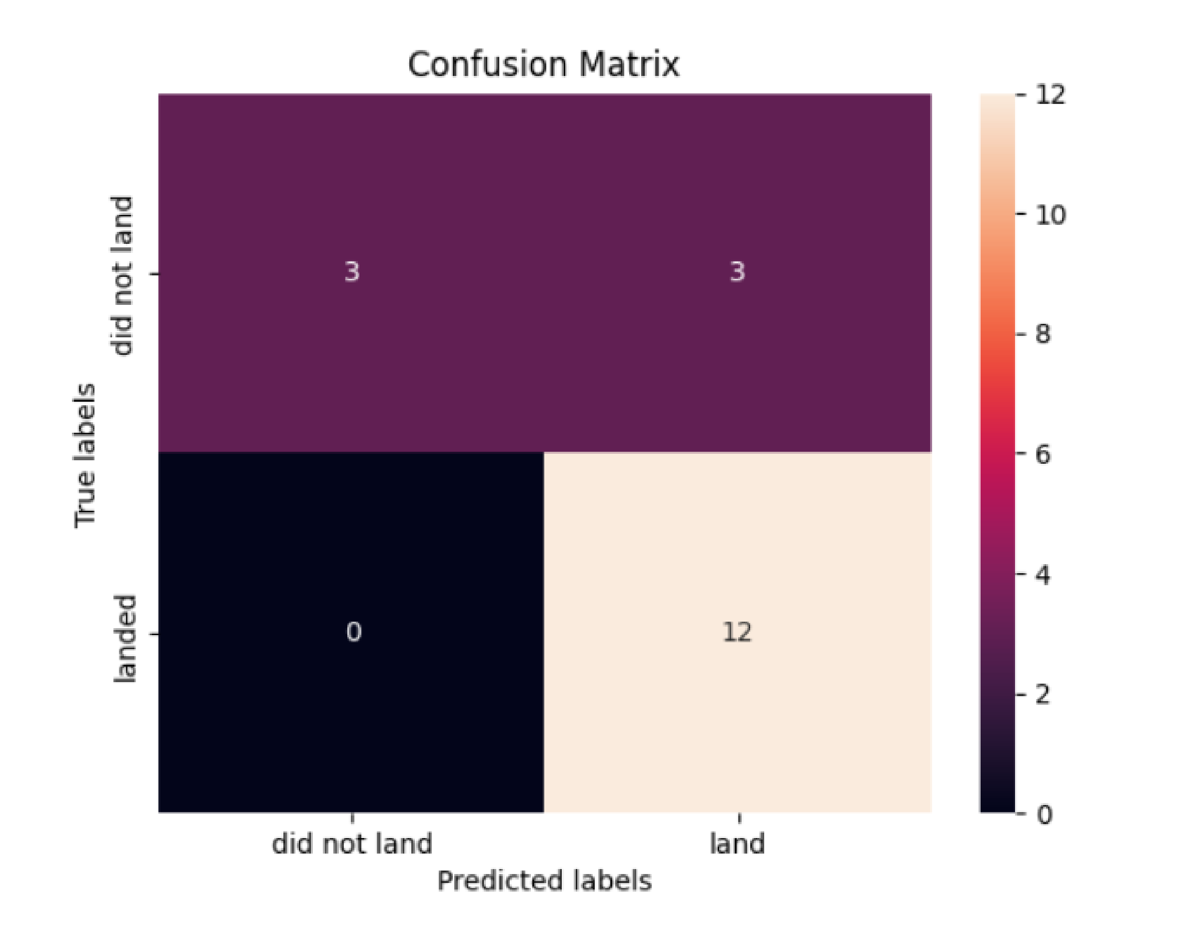
Classification Accuracy

Model	Accuracy	TestAccuracy
LogReg	0.84643	0.83333
SVM	0.84821	0.83333
Tree	0.875	0.94444
KNN	0.84821	0.83333



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., unsuccessful landing marked as successful landing by the classifier.



Conclusions

We conclude that

- Flight amount and success rate: There is a positive correlation between the flight amount at a launch site and the success rate. Sites with larger flight amounts tend to have higher success rates.
- Increase in success rate: From 2013 to 2020, there was a noticeable increase in the overall launch success rate.
- Successful orbits: Orbits such as ES-L1, GEO, HEO, SSO, and VLEO exhibited the highest success rates among all the orbits.
- Successful launches at KSC LC-39A: KSC LC-39A was identified as the launch site with the highest number of successful launches compared to other sites.
- Best machine learning algorithm: The Decision Tree Classifier was determined to be the most effective machine learning algorithm for this specific task.

Thank you!