IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

TRINH BAO TOAN
11/08/2023

Outline

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

Executive Summary

- Summary of methodologies
  - Gathering Data through API Integration
  - Extracting Data through Web Scraping
  - Data Preparation and Cleaning
  - Initial Data Analysis using SQL
  - Initial Data Analysis through Visualizations
  - Engaging Visual Analysis with Folium
  - Utilizing Machine Learning for Predictive Insights
- Summary of all results
  - Findings from Initial Data Analysis
  - Screenshots showcasing Interactive Analytics
  - Outcomes of Predictive Analytics

Introduction

- Project background and context

  SpaceX offers Falcon 9 rocket launches at a significantly lower cost of 62 million dollars compared to other providers, whose prices start at 165 million dollars. This cost advantage is largely attributed to SpaceX's ability to reuse the first stage of the rocket. Therefore, the ability to predict whether the first stage will successfully land is crucial in estimating the overall launch cost. This information can be invaluable for companies considering bidding against SpaceX for a rocket launch contract. The primary objective of this project is to establish a machine learning pipeline capable of forecasting the successful landing of the first stage.

- Problems you want to find answers

  - What are the determining factors for a successful rocket landing?
  - How do various features interact to influence the success rate of a landing?
  - What operational conditions are essential for ensuring a successful landing program?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Utilized SpaceX API and conducted web scraping from Wikipedia for data acquisition.
- Perform data wrangling
  - Implemented data wrangling techniques to ensure data quality and consistency.
  - Applied one-hot encoding to handle 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
  - How to build, tune, evaluate classification models

Data Collection

- **Data Collection Approach:**
- The data gathering process encompassed a diverse set of techniques:
- **SpaceX API Integration:**
  - Data retrieval was initiated through a series of HTTP GET requests to the SpaceX API.
- **Response Decoding and DataFrame Creation:**
  - The response content was decoded as JSON using the .json() function call, and subsequently transformed into a structured pandas dataframe via .json_normalize().
- **Data Cleaning and Imputation:**
  - Rigorous cleaning procedures were implemented, including the identification and handling of missing values where applicable.
- **Web Scraping with BeautifulSoup:**
  - We conducted web scraping activities on Wikipedia to extract Falcon 9 launch records. This involved the extraction of launch records from an HTML table, parsing the content, and converting it into a pandas dataframe for future analytical purposes.

Data Collection – SpaceX API

- The data acquisition process commenced with the utilization of HTTP GET requests directed towards the SpaceX API. Subsequently, the obtained data underwent a thorough cleaning process, which encompassed basic data wrangling and formatting procedures.
- https://github.com/sonyway1704/IBM-Data-Science-Professional-Certificate/blob/main/IBM%20Data%20Science%20Capstone%20Project%20-%20SpaceX/Data%20Collection%20API.ipynb

1. Get request for rocket launch data using API

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

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

2. Use json_normalize method to convert json result to dataframe

In [12]:
```
# Use json_normalize method to convert the json result into a dataframe

# decode response content as json
static_json_df = res.json()
```

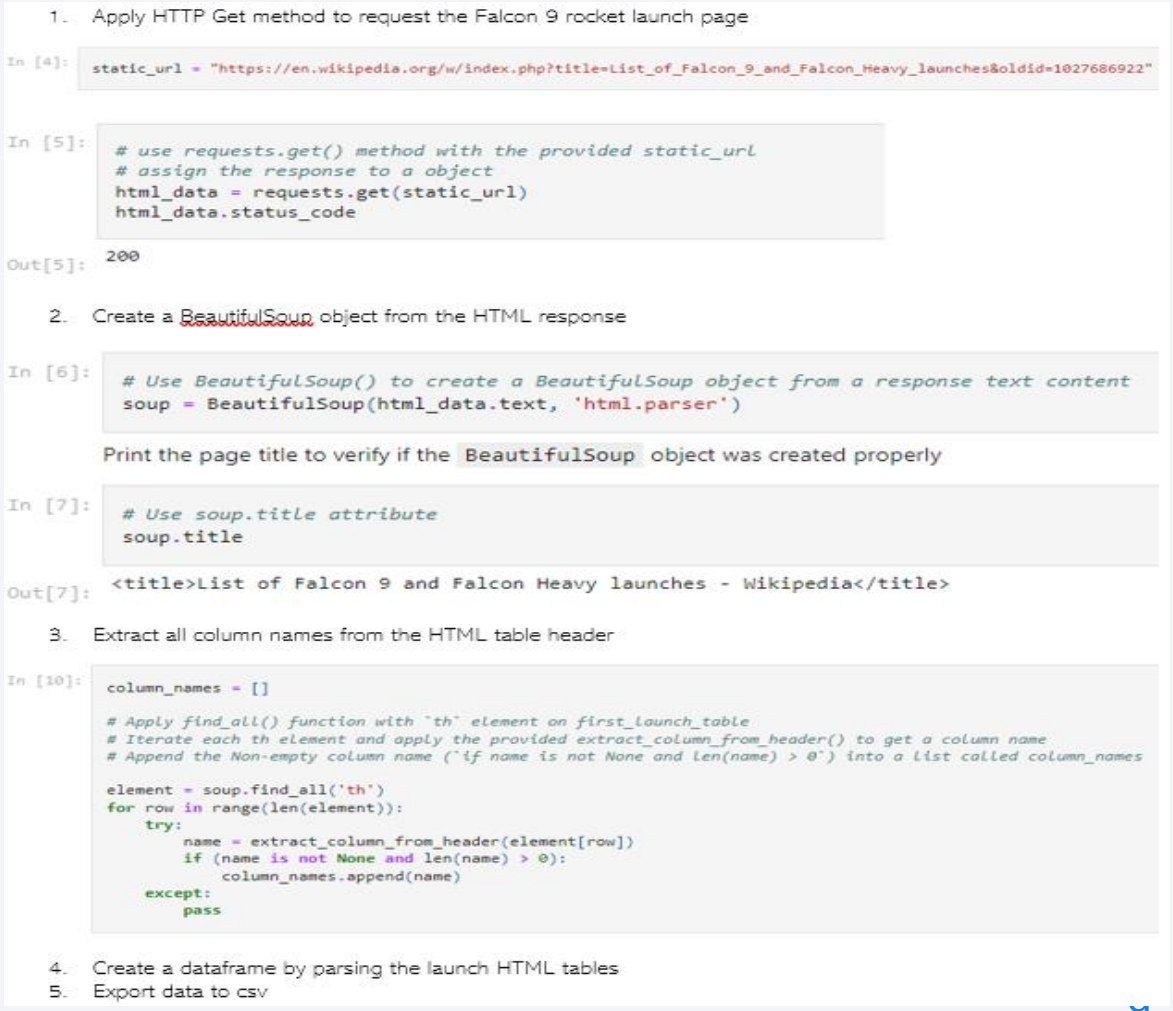
In [13]:
```
# apply json_normalize
data = pd.json_normalize(static_json_df)
```

3. We then performed data cleaning and filling in the missing values

In [30]:
```
rows = data_falcon9['PayloadMass'].values.tolist()[0]

df_rows = pd.DataFrame(rows)
df_rows = df_rows.replace(np.nan, PayloadMass)

data_falcon9['PayloadMass'][0] = df_rows.values
data_falcon9
```

Data Collection - Scraping

- We utilized BeautifulSoup for web scraping activities targeted at Falcon 9 launch records. This involved parsing the extracted table and subsequently transforming it into a structured pandas dataframe.
- https://github.com/sonyway1704/IBM-Data-Science-Professional-Certificate/blob/main/IBM%20Data%20Science%20Capstone%20Project%20-%20SpaceX/Data%20Collection%20with%20Web%20Scraping.ipynb



1. Apply HTTP Get method to request the Falcon 9 rocket launch page

In [4]:
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

In [5]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
html_data = requests.get(static_url)
html_data.status_code
```

Out[5]: 200

2. Create a BeautifulSoup object from the HTML response

In [6]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(html_data.text, 'html.parser')
```

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

In [7]:
```
# Use soup.title attribute
soup.title
```

Out[7]: <title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>

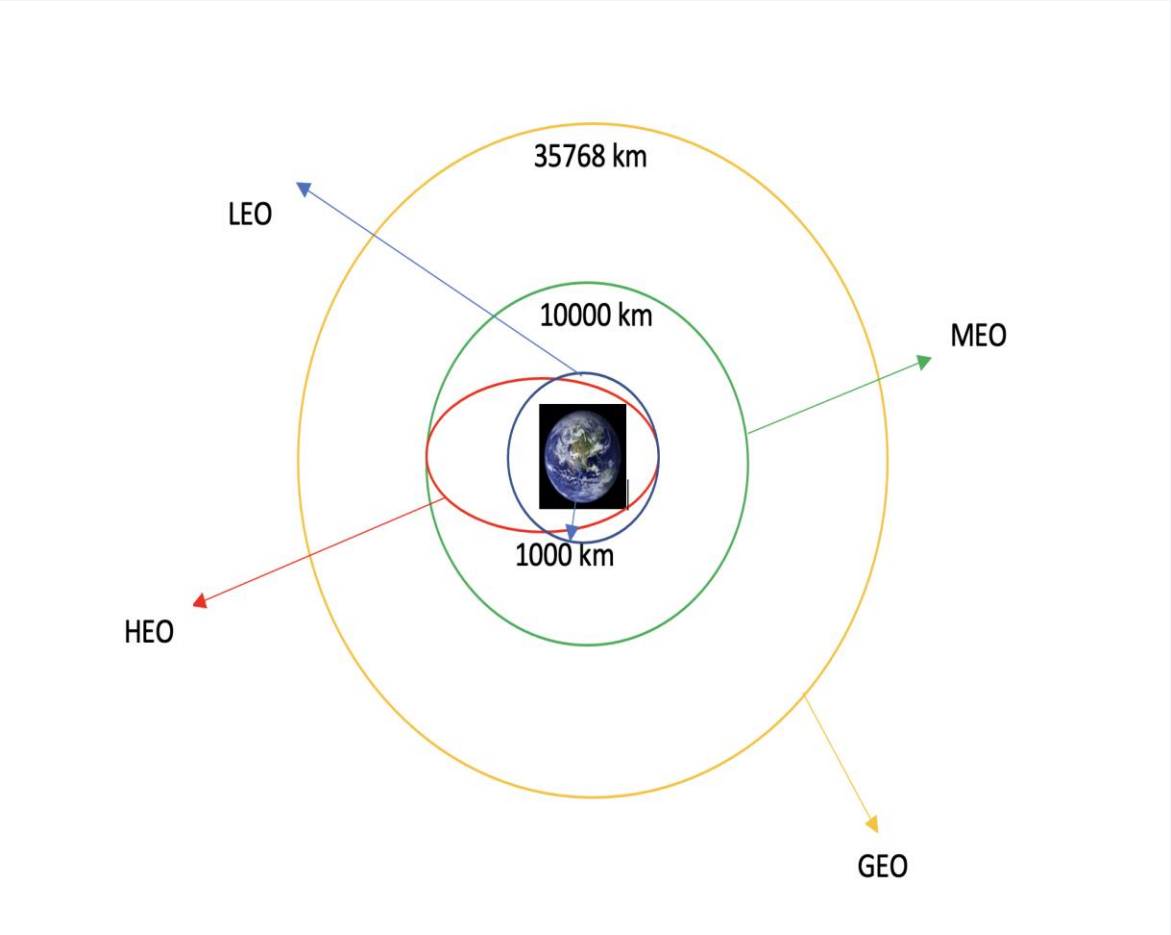
3. Extract all column names from the HTML table header

In [10]:
```
column_names = []

# 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

element = soup.find_all('th')
for row in range(len(element)):
    try:
        name = extract_column_from_header(element[row])
        if (name is not None and len(name) > 0):
            column_names.append(name)
    except:
        pass
```

4. Create a dataframe by parsing the launch HTML tables
5. Export data to csv

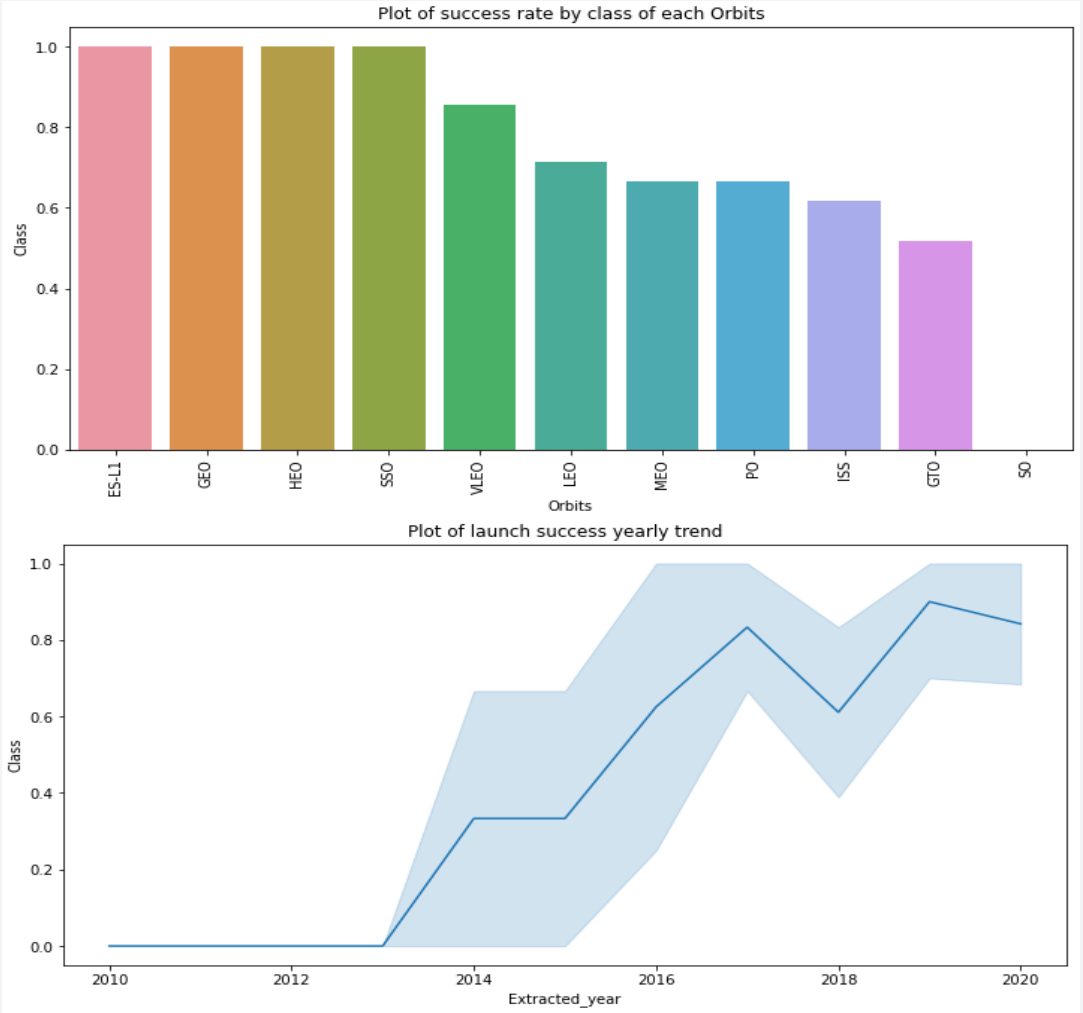
Data Wrangling

- We conducted exploratory data analysis (EDA) to gain insights into the dataset and establish training labels. This included computations for the count of launches at individual sites, as well as the frequency and occurrence of distinct orbits.
- Furthermore, we derived a landing outcome label based on the information provided in the outcome column, and subsequently exported the outcomes to a CSV file.
- https://github.com/sonyway1704/IBM-Data-Science-Professional-Certificate/blob/main/IBM%20Data%20Science%20Capstone%20Project%20-%20SpaceX/Data%20Wrangling.ipynb



EDA with Data Visualization

- **Data Exploration through Visualization:**
- We delved into the dataset by employing visualizations to examine various relationships:
- Relationship between Flight Number and Launch Site
- Correlation between Payload and Launch Site
- Success Rate Analysis for Each Orbit Type
- Association between Flight Number and Orbit Type
- Yearly Trend Analysis of Launch Success
- https://github.com/sonyway1704/IBM-Data-Science-Professional-Certificate/blob/main/IBM%20Data%20Science%20Capstone%20Project%20-%20SpaceX/EDA%20with%20Data%20Visualization.ipynb



EDA with SQL

- **Database Integration and SQL-based EDA:**
- We seamlessly integrated the SpaceX dataset into a PostgreSQL database directly from the Jupyter notebook environment.
- We then applied Exploratory Data Analysis (EDA) utilizing SQL queries to extract insightful information from the dataset. Some of the key queries included:

1. Obtaining the names of distinct launch sites involved in the space missions.
2. Calculating the total payload mass carried by boosters launched under NASA's CRS missions.
3. Determining the average payload mass carried by booster version F9 v1.1.
4. Enumerating the total count of both successful and failed mission outcomes.
5. Identifying instances of failed landings on drone ships, including details about the associated booster version and launch site names.

- https://github.com/sonyway1704/IBM-Data-Science-Professional-Certificate/blob/main/IBM%20Data%20Science%20Capstone%20Project%20-%20SpaceX/EDA%20with%20SQL.ipynb

Build an Interactive Map with Folium

- **Map Annotation and Analysis:**
- We enriched the map interface by tagging all launch sites and incorporating map elements like markers, circles, and lines. These visual cues were employed to signify the outcomes (success or failure) of each launch at the respective sites within the Folium map.
- Furthermore, we categorized launch outcomes into classes, labeling failures as 0 and successes as 1.
- Leveraging color-coded marker clusters, we discerned launch sites with notably high success rates.
- Additionally, we conducted distance computations between launch sites and their immediate surroundings. This led us to address pertinent questions, such as:
- Proximity to railways, highways, and coastlines
- The degree of separation from nearby cities
- https://github.com/sonyway1704/IBM-Data-Science-Professional-Certificate/blob/main/IBM%20Data%20Science%20Capstone%20Project%20-%20SpaceX/Interactive%20Visual%20Analytics%20with%20Folium.ipynb

Build a Dashboard with Plotly Dash

- **Interactive Dashboard with Plotly Dash:**
- We constructed a dynamic and interactive dashboard using Plotly Dash, providing an engaging interface for data exploration.
- Within the dashboard, we integrated pie charts to visually represent the distribution of total launches across specific launch sites.
- Additionally, we incorporated scatter plots to illustrate the correlation between Outcome and Payload Mass (in kilograms) for distinct booster versions.
- https://github.com/sonyway1704/IBM-Data-Science-Professional-Certificate/blob/main/IBM%20Data%20Science%20Capstone%20Project%20-%20SpaceX/app.py

Predictive Analysis (Classification)

- **Data Loading, Transformation, and Model Building:**
- We employed the powerful combination of NumPy and Pandas to efficiently load and manipulate the dataset. Subsequently, the data was divided into training and testing sets to facilitate model development.
- We constructed a diverse range of machine learning models and fine-tuned their hyperparameters using GridSearchCV, a systematic approach for optimizing model performance.
- Our metric of choice for model evaluation was accuracy. We iteratively refined the model by implementing feature engineering techniques and fine-tuning the algorithms.
- Through rigorous experimentation, we identified the classification model that exhibited the highest level of performance.
- https://github.com/sonyway1704/IBM-Data-Science-Professional-Certificate/blob/main/IBM%20Data%20Science%20Capstone%20Project%20-%20SpaceX/Machine%20Learning%20Prediction.ipynb

Results

Find the method performs best:

```
]:
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_)
```

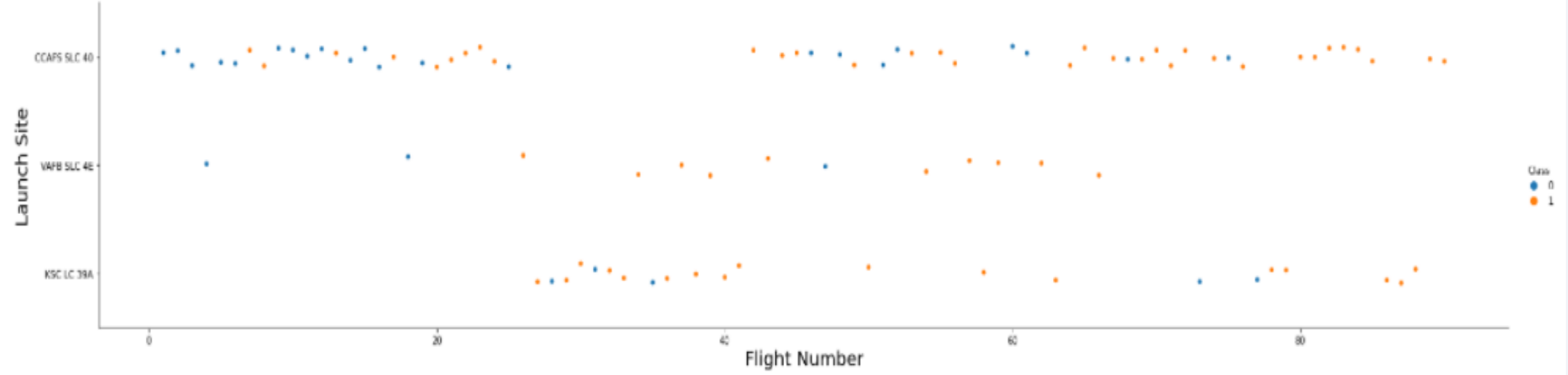
```
Best model is DecisionTree with a score of 0.8732142857142856
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split':
5, 'splitter': 'random'}
```

Section 2

Insights drawn from EDA

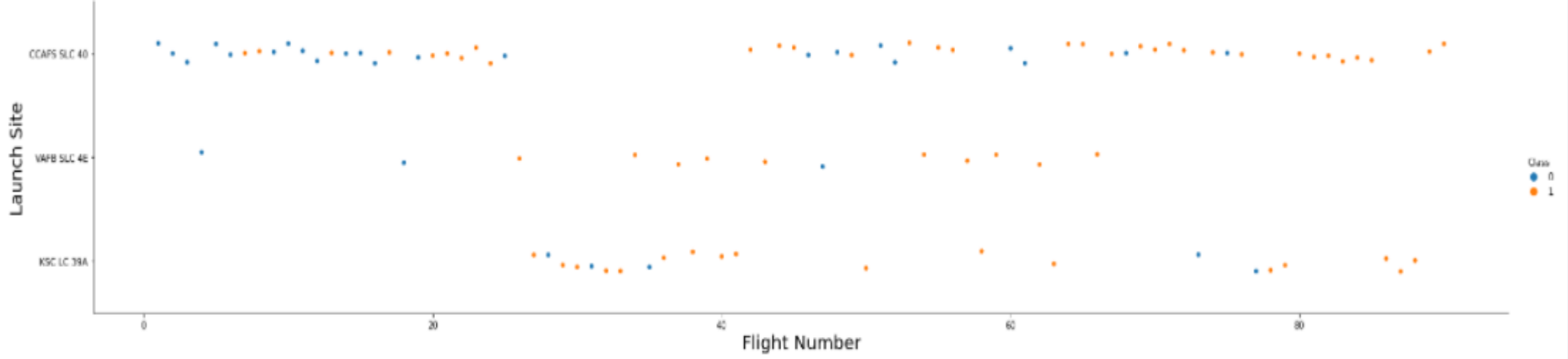
Flight Number vs. Launch Site

- Show a scatter plot of Flight Number vs. Launch Site



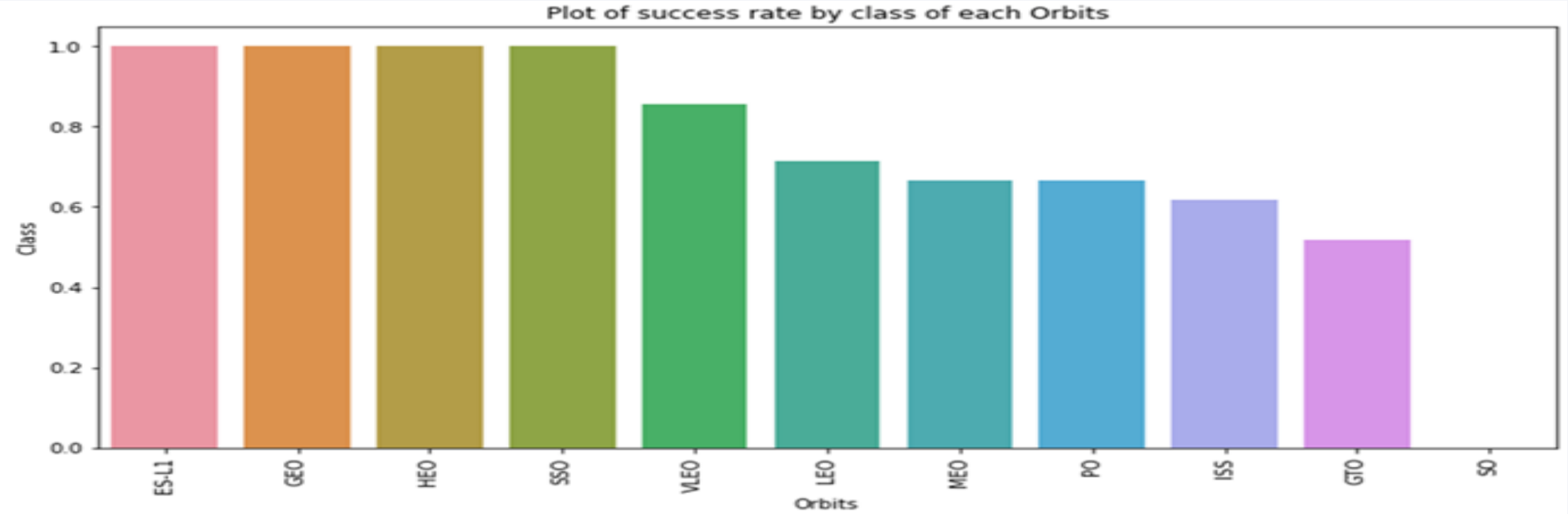
Payload vs. Launch Site

The greater the payload mass for launch site CCAFS SLC 40 the higher the success rate for the rocket.



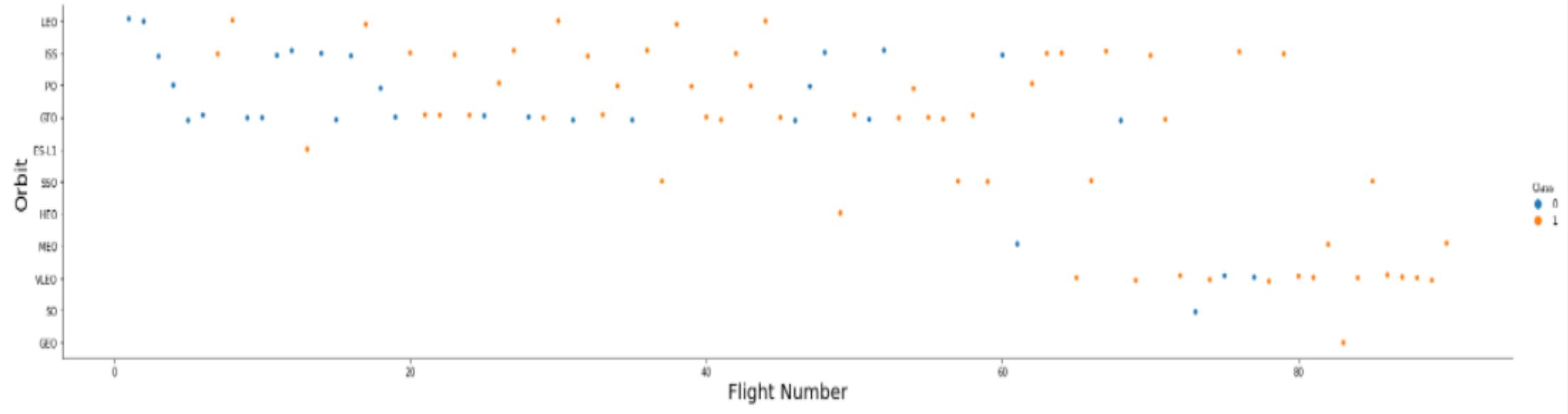
Success Rate vs. Orbit Type

The plot clearly indicates that ES-L1, GEO, HEO, SSO, and VLEO orbits demonstrated the highest success rates among the launch sites.



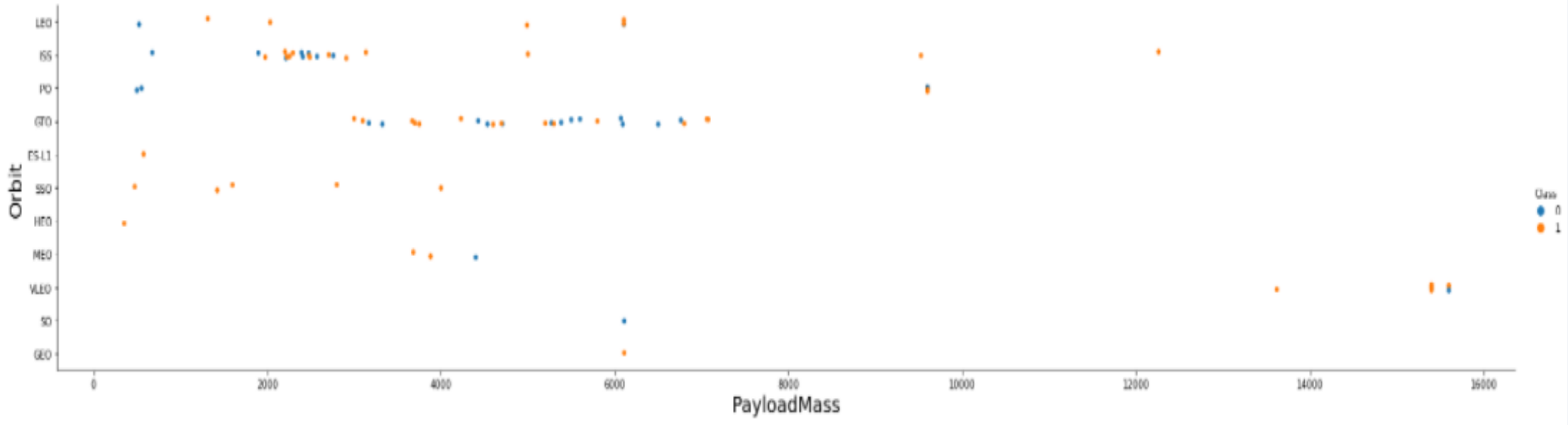
Flight Number vs. Orbit Type

- In the plot depicting Flight Number versus Orbit type, notable patterns emerge. Specifically, within the LEO (Low Earth Orbit) category, there appears to be a discernible connection between the number of flights and the success rate. Conversely, in the GTO (Geostationary Transfer Orbit) category, no significant correlation is evident between flight count and the orbit type.



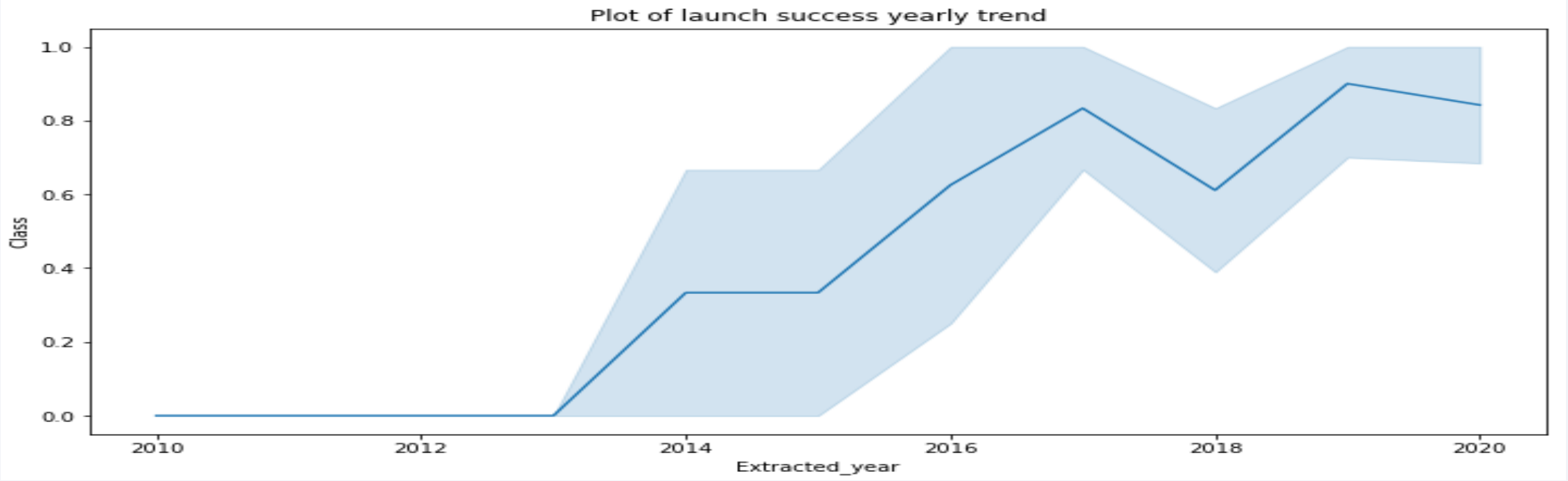
Payload vs. Orbit Type

Upon examination, it becomes apparent that orbits associated with heavy payloads, such as PO (Polar Orbit), LEO (Low Earth Orbit), and ISS (International Space Station), exhibit a higher frequency of successful landings.



Launch Success Yearly Trend

- The plot reveals a notable trend. Specifically, it demonstrates that the success rate has exhibited a consistent upward trajectory from 2013 to 2020.



All Launch Site Names

Utilizing DISTINCT Keyword for Unique Launch Sites:

To extract only the unique launch sites from the SpaceX dataset, we employed the DISTINCT keyword.

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'

Query for Displaying Records with Specific Launch Sites:

The provided query was utilized to retrieve and display five records where the launch sites commence with the identifier ‘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

Calculation of Total Payload for NASA Boosters:

By executing the provided query, we determined that the total payload carried by boosters from NASA amounted to 45,596 units.

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

Average Payload Mass for Booster Version F9 v1.1:
Through our computations, we established that the average payload mass carried by booster version F9 v1.1 amounted to 2928.4 units.

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

Observation of First Successful Ground Pad Landing Date:

Based on our observation, we noted that the inaugural instance of a successful landing on a ground pad occurred on December 22, 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

Filtering with WHERE Clause for Successful Drone Ship Landings:

We applied the WHERE clause to specifically target boosters that have achieved successful landings on drone ships. In addition, we utilized the AND condition to further refine our search, ensuring that the payload mass falls within the range of 4000 to 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

Utilizing Wildcards with WHERE Clause:

In our query, we incorporated the wildcard symbol '%' with the WHERE clause to selectively retrieve instances where the ‘MissionOutcome’ indicated either 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)
```

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

Identifying Booster with Maximum Payload using Subquery and MAX() Function:

We employed a subquery within the WHERE clause, coupled with the MAX() function, to ascertain the specific booster that had transported the highest 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

2015 Launch Records

Applying WHERE, LIKE, AND, and BETWEEN Conditions to Filter Drone Ship Failures in 2015:

We utilized a combination of the WHERE clause, LIKE, AND, and BETWEEN conditions to selectively extract information pertaining to failed landing outcomes on drone ships. This included details about the associated booster versions and launch site names, specifically for the 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

Selection, Filtering, Grouping, and Ordering of Landing Outcomes:

- We executed a query to retrieve both the landing outcomes and their corresponding counts from the dataset. Using the WHERE clause, we filtered the results to include only outcomes between June 4, 2010, and March 20, 2010.
- Next, we applied the GROUP BY clause to aggregate the landing outcomes. Finally, we utilized the ORDER BY clause to arrange the grouped outcomes in descending order.

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

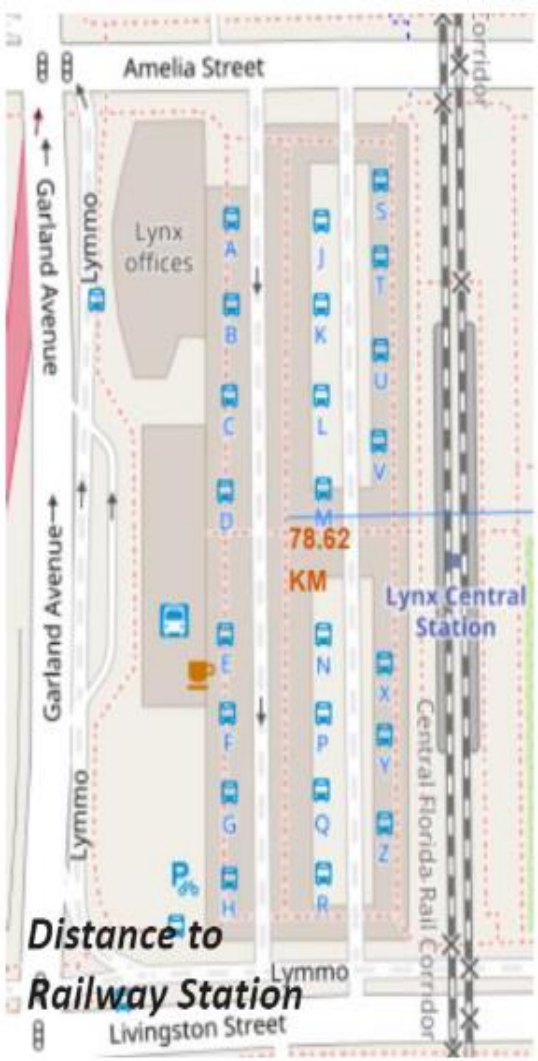
Launch Sites Proximities Analysis

LAUNCH SITES MAP

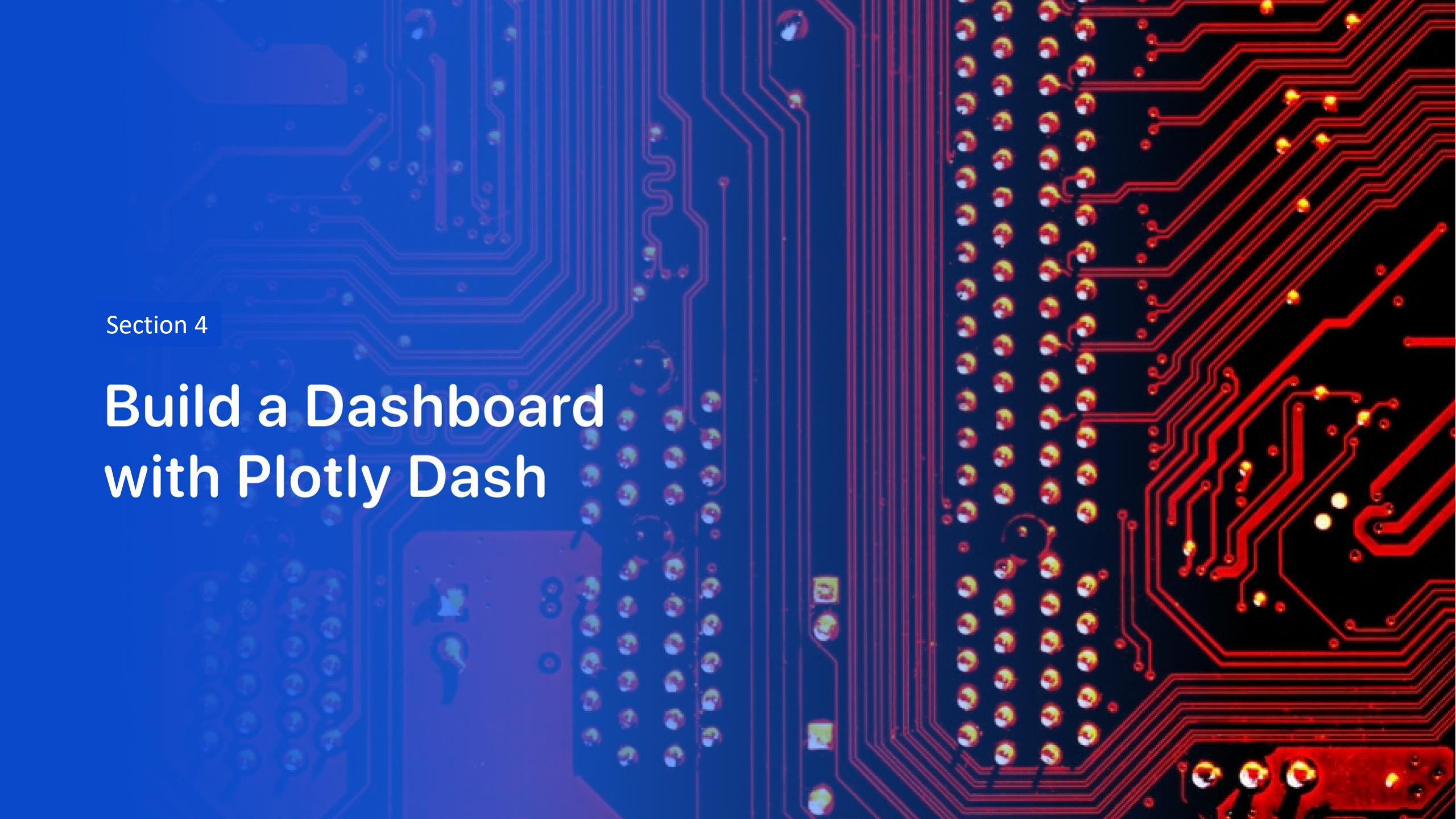
We can see that the SpaceX launch sites are in the United States of America coasts. Florida and California

LAUNCH SITES MAP WITH LABELS

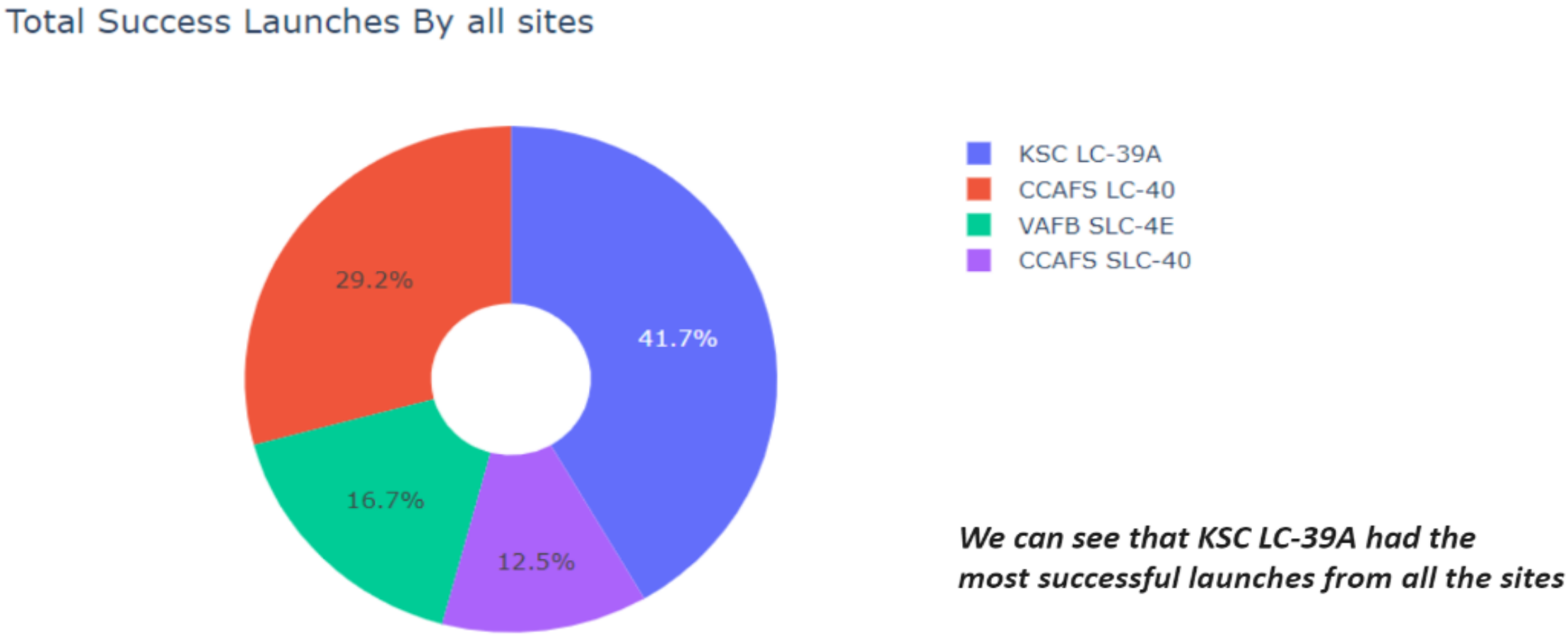
LAUNCH SITES MAP DISTANCE TO LANDMARKS



- 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

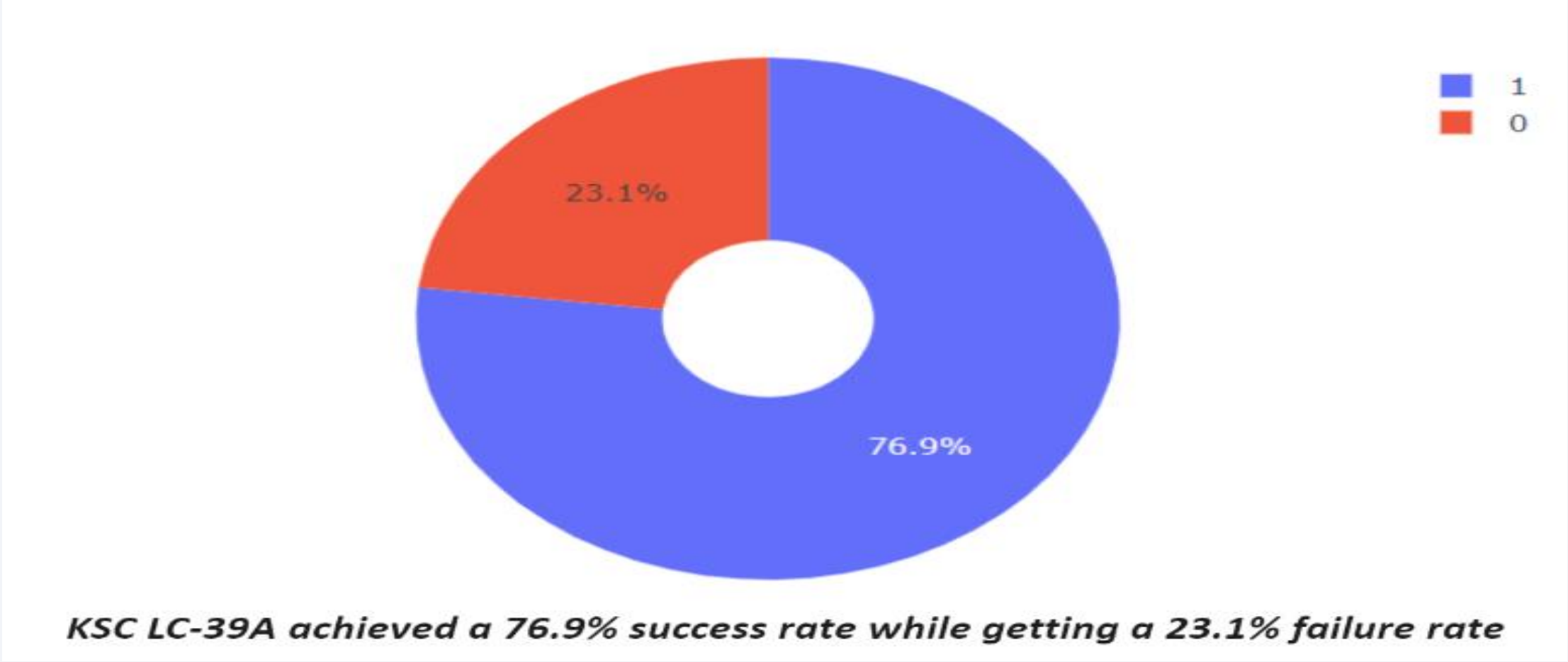

Section 4
Build a Dashboard with Plotly Dash

The success percentage achieved by each launch site



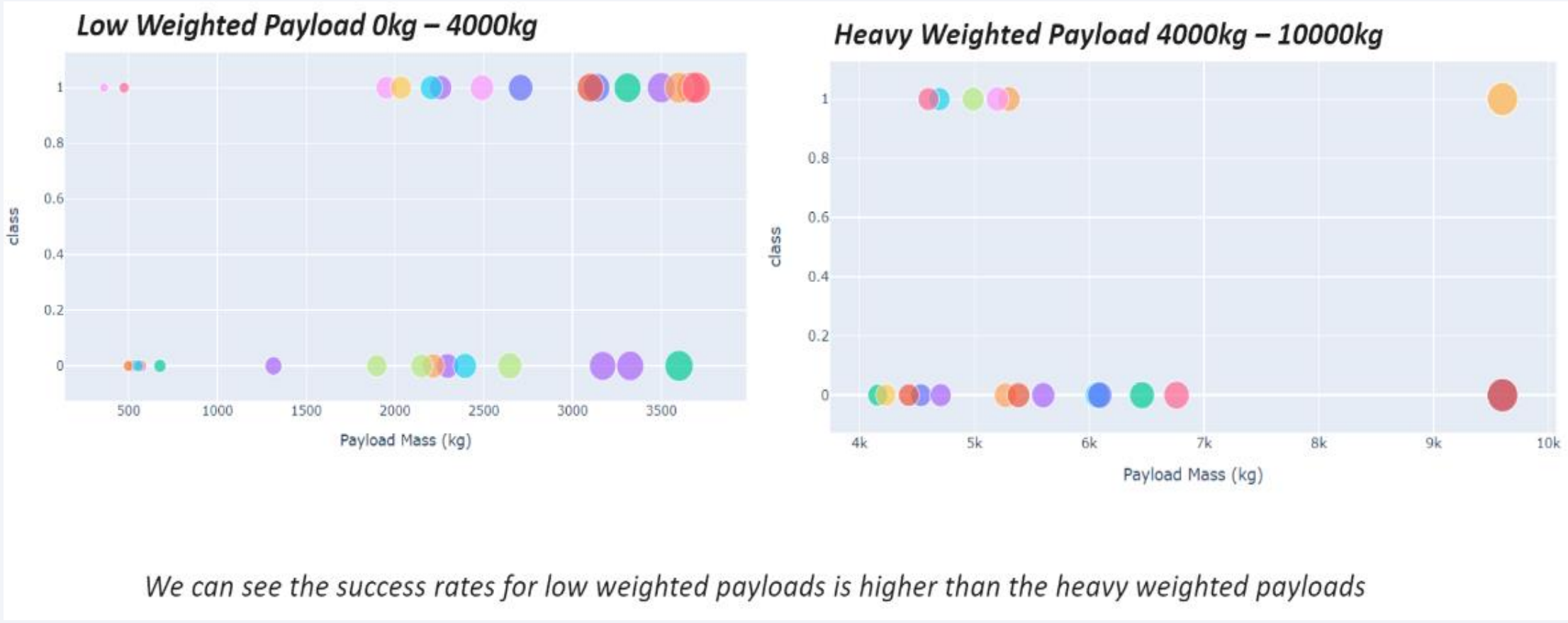
We can see that KSC LC-39A had the most successful launches from all the sites

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

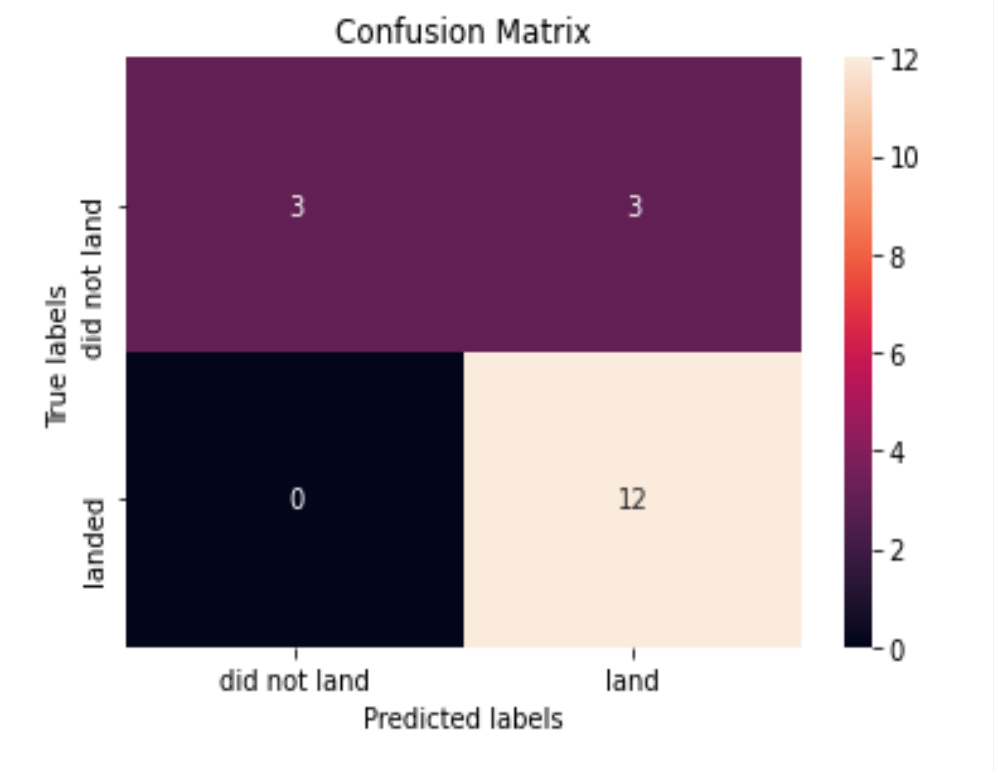
The Decision Tree Classifier demonstrates the highest level of accuracy in classifying the data.

```
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'}
```

Confusion Matrix

The confusion matrix of the decision tree classifier demonstrates its capacity to discern between different classes. However, a noteworthy concern arises from the occurrence of false positives, where unsuccessful landings are erroneously categorized as successful by the classifier.



Conclusions

- The analysis of the SpaceX dataset for rocket launches has yielded valuable insights. The data was collected using various methods, including SpaceX API integration and web scraping from Wikipedia. Exploratory Data Analysis (EDA) and data visualization techniques were employed to gain a comprehensive understanding of the dataset.
- The machine learning pipeline was implemented to predict the success of the first stage landing. Several models were built and evaluated, with the Decision Tree Classifier emerging as the most accurate in classification.
- Furthermore, the examination of specific features and visualizations unveiled interesting patterns. Notably, the success rate of launches showed a positive correlation with the number of flights at a launch site, particularly in the Low Earth Orbit (LEO) category. Additionally, specific orbits, such as ES-L1, GEO, HEO, SSO, and VLEO, exhibited higher success rates.
- Overall, the project successfully leveraged data collection, preprocessing, exploratory analysis, and machine learning techniques to make predictions regarding first stage landings. This information can prove invaluable for strategic decision-making in the context of rocket launches.

Thank you!