IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Ashwini Dadasaheb Khandave
31/01/2024

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion

Executive Summary

This project aims to predict the successful landing of the Falcon 9 first stage, crucial for estimating launch costs and competitive bidding.

Key Methods:

- Data Manipulation and Analysis
- Data Collection and Wrangling
- Interactive Dashboard Creation
- Machine Learning Modeling

Key Results:

- Successful data manipulation and analysis provided valuable insights.
- Cleaned Falcon 9 landing data enabled accurate analysis.
- Developed an intuitive interactive dashboard for launch analysis.
- Machine learning models achieved high accuracy in predicting landing success.

Overall, this project aims to optimize rocket launch operations, contributing to more efficient and cost-effective space missions.

Introduction

Project Background:

- **SpaceX & Falcon 9**: SpaceX, led by Elon Musk, is a key player in aerospace, with the Falcon 9 rocket being pivotal in space transport.
- **First-Stage Landing**: Successful first-stage landing of Falcon 9 reduces launch costs significantly, making accurate prediction crucial for cost estimation and competitive bidding.

Key Problems:

- **Predicting Landing Success**: Project focuses on accurately predicting Falcon 9 first-stage landing to improve cost estimation and bidding decisions.
- **Data-Driven Approach**: Utilizing data analysis and machine learning to make informed predictions based on historical launch data.
- **Operational Optimization**: Optimizing rocket launch operations for increased efficiency and cost-effectiveness.

Section 1

Methodology

Methodology

Executive Summary

- **Data Collection**:
    - Data collected from various sources, including SpaceX API and web scraping techniques.
- **Data Wrangling**:
    - Processed data to ensure consistency and accuracy, including handling missing values and formatting issues.
- **Exploratory Data Analysis (EDA)**:
    - Utilized visualization techniques and SQL queries to explore data patterns, trends, and correlations.
- **Interactive Visual Analytics**:
    - Implemented interactive visual analytics using Folium for geographic analysis and Plotly Dash for dynamic dashboard creation.
- **Predictive Analysis**:
    - Developed classification models using machine learning techniques.
- **Model Building and Tuning**:
    - Built and fine-tuned classification models to improve accuracy and performance.

Data Collection

The data sets were collected using various methods:

- **SpaceX API**: A GET request was made to the SpaceX API to retrieve relevant data.
- **Data Wrangling and Formatting**: The response content was decoded as JSON using the .json() function and converted into a pandas DataFrame using .json_normalize().
- **Data Cleaning**: The collected data underwent cleaning processes to address missing values, ensuring data integrity.
- **Web Scraping**: Falcon 9 launch records were extracted from Wikipedia using BeautifulSoup. This involved parsing the HTML table containing launch records and converting it into a pandas DataFrame for further analysis.

Data Collection – SpaceX API

- Utilized a GET request to the SpaceX API for data collection, followed by cleaning and basic data wrangling to ensure data quality and formatting.
- Link - https://github.com/AshuKhandave/Machine_Learning_Project/blob/main/Code_files/spacex-data-collection-api.ipynb

Requesting rocket launch data from SpaceX API with the following URL:

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

[7]
```python
response = requests.get(spacex_url)
```
Python

[8]
```python
print(response.content)
```
Python

b'[{"fairings":{"reused":false,"recovery_attempt":false,"recovered":false,"ships":[]},"links":{"p

[11]
```python
# json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```
Python

```python
# Missing values in the LaunchSite
# Calculate the mean value of PayloadMass column
mean_payload = data_falcon9['PayloadMass'].mean()
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'].replace(np.nan, mean_payload, inplace=True)
```
Python

Data Collection - Scraping

- Applied web scrapping to webscrap Falcon 9 launch records with **BeautifulSoup**
- Parsed the table and converted it into a pandas dataframe.
- Link - https://github.com/AshuKhandave/Machine_Learning_Project/blob/main/Code_files/webscraping.ipynb

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

```python
# use requests.get() method with the provided static_url
response = requests.get(static_url)
```

```python
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.text, 'html.parser')
```

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

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

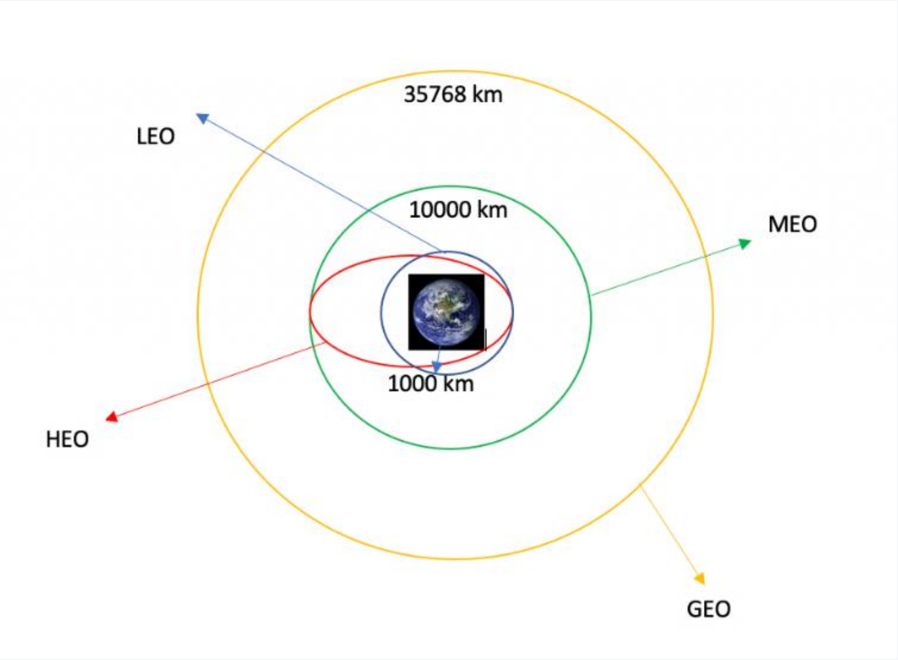
```python
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
html_tables
```

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

```python
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 colu
thead = first_launch_table.find_all('th')
column_names = [extract_column_from_header(row) for row in thead]
```

Data Wrangling

- Calculated launches per site using **value_counts()** on **LaunchSite**.
- Counted orbit occurrences with **value_counts()** on **Orbit**.
- Determined landing outcomes frequency with **value_counts()** on **Outcome**.
- Created **landing_class** list: 0 if **Outcome** in **bad_outcome**, 1 otherwise.
- Link - https://github.com/AshuKhandave/Machine_Learning_Project/blob/main/Code_files/spacex-data%20wrangling.ipynb



EDA with Data Visualization

- We visualized:
  - Relationship between flight number and launch site
  - Payload and launch site correlation
  - Success rate of each orbit type
  - Flight number and orbit type correlation
  - Launch success yearly trend
- Link - https://github.com/AshuKhandave/Machine_Learning_Project/blob/main/Code_files/EDA_with_python.ipynb

EDA with SQL

- We loaded the SpaceX dataset into a PostgreSQL database directly from the Jupyter notebook.
- Then, we performed Exploratory Data Analysis (EDA) with SQL to extract insights from the data. Some of the queries we executed included:
  - Finding the names of unique launch sites in the space mission.
  - Calculating the total payload mass carried by boosters launched by NASA (CRS).
  - Determining the average payload mass carried by booster version F9 v1.1.
  - Analyzing the total number of successful and failure mission outcomes.
  - Identifying failed landing outcomes in the drone ship, including their booster version and launch site names.
- Link - https://github.com/AshuKhandave/Machine_Learning_Project/blob/main/Code_files/EDA_SQL.ipynb

Interactive Map with Folium

- Utilized **Folium** for map creation and visualization, pinpointing launch site locations with **markers** and **circles**.
- Employed **MarkerClusters** to categorize launch outcomes and **MousePosition** for real-time coordinate display.
- Implemented **PolyLines** to calculate and depict distances between launch sites and nearby features.
- Link - https://github.com/AshuKhandave/Machine_Learning_Project/blob/main/Code_files/launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- Developed an interactive dashboard using **Plotly Dash**.
- Visualized total launches by specific sites using pie charts.
- Examined the relationship between **Outcome** and **Payload Mass (Kg)** for various booster versions using scatter graphs.
- Link - https://github.com/AshuKhandave/Machine_Learning_Project/blob/main/Code_files/spacex_dash_app.py

Predictive Analysis (Classification)

Following are the steps taken to find the best performing classification model:

1. **Data Loading and Transformation**: Loaded data with NumPy and Pandas. Transformed and preprocessed the data
2. **Data Splitting**: Split the data into training and testing sets.
3. **Model Building**: Developed machine learning models.
4. **Hyperparameter Tuning**: Used GridSearchCV to tune hyperparameters.
5. **Model Evaluation**: Evaluated models based on accuracy.
6. **Model Improvement**: Improved models with feature engineering and tuning.
7. **Best Performing Model**: Selected the top-performing classification model.

Link - https://github.com/AshuKhandave/Machine_Learning_Project/blob/main/Code_files/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Results

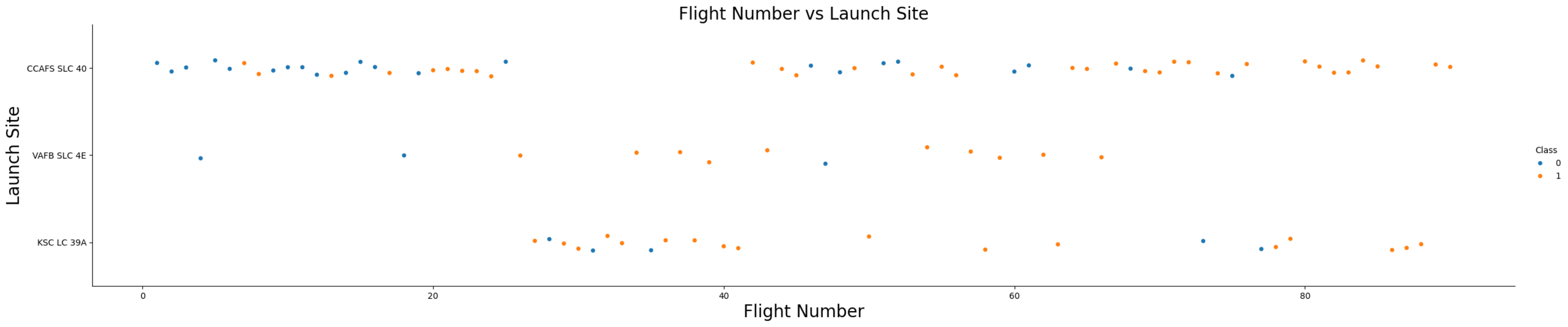
- Exploratory data analysis results
- Interactive analytics demo
- Predictive analysis results

Section 2

Insights drawn from EDA

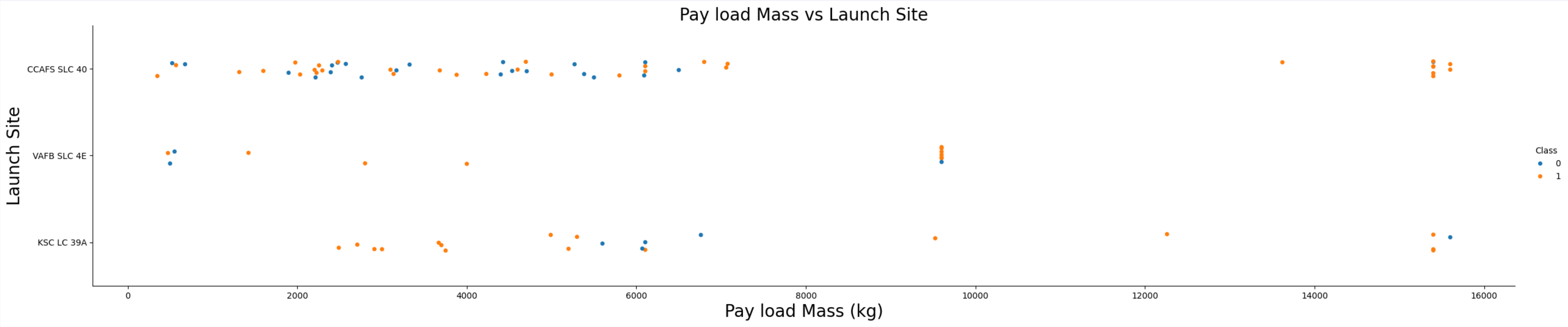
Flight Number vs. Launch Site

The analysis revealed that launch sites with higher flight counts tended to have higher success rates.



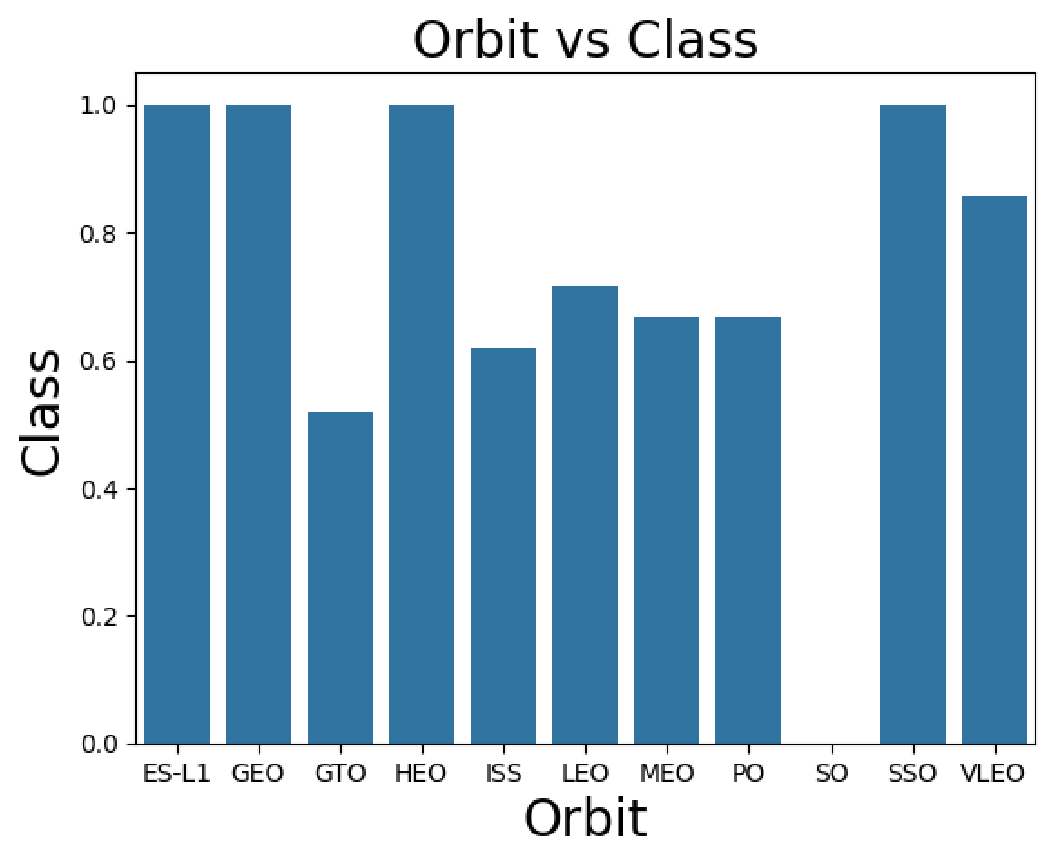
Payload vs. Launch Site

At the VAFB-SLC launch site, there were no rockets launched with a payload mass exceeding 10,000 kg.



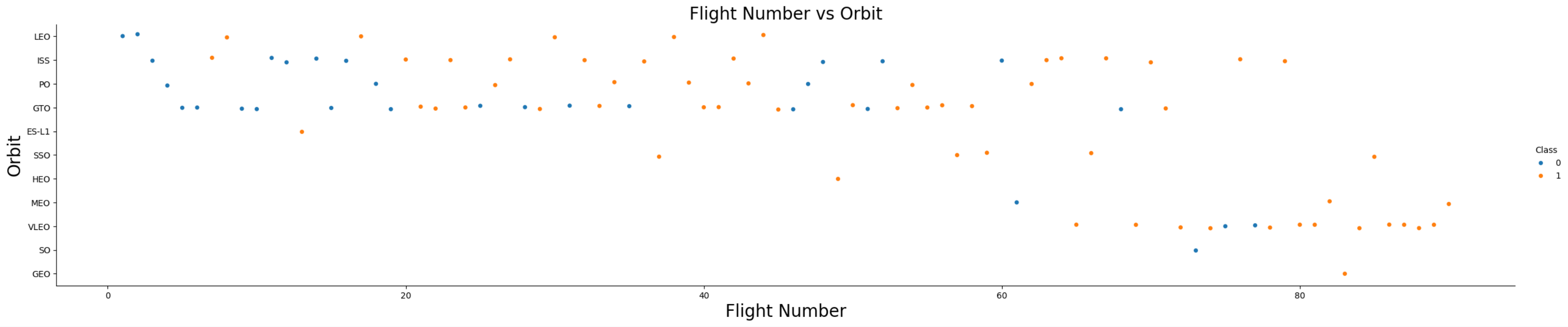
Success Rate vs. Orbit Type

From the plot, we can see that ES-L1, GEO, HEO, SSO, VLEO had the most success rate.



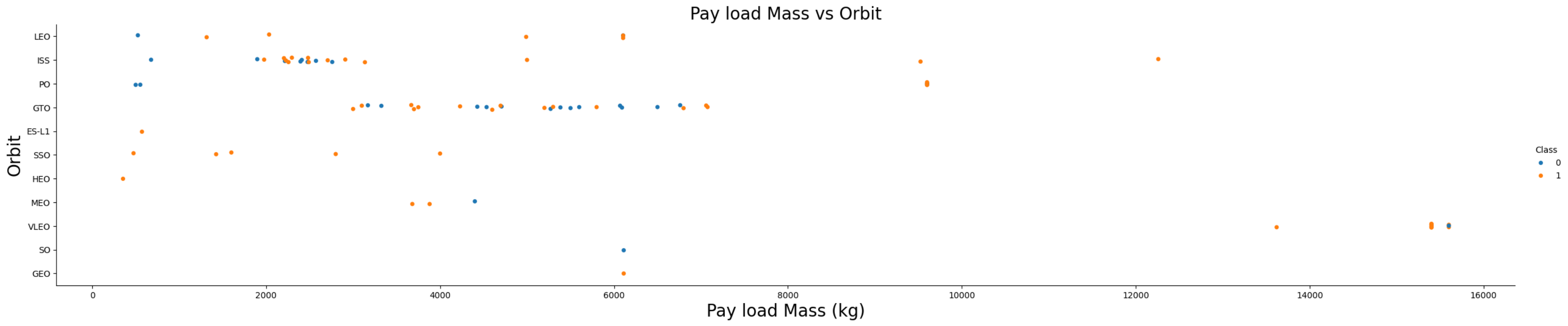
Flight Number vs. Orbit Type

In the LEO orbit, success appears to be related to the number of flights, whereas in the GTO orbit, there seems to be no relationship between flight number and success.



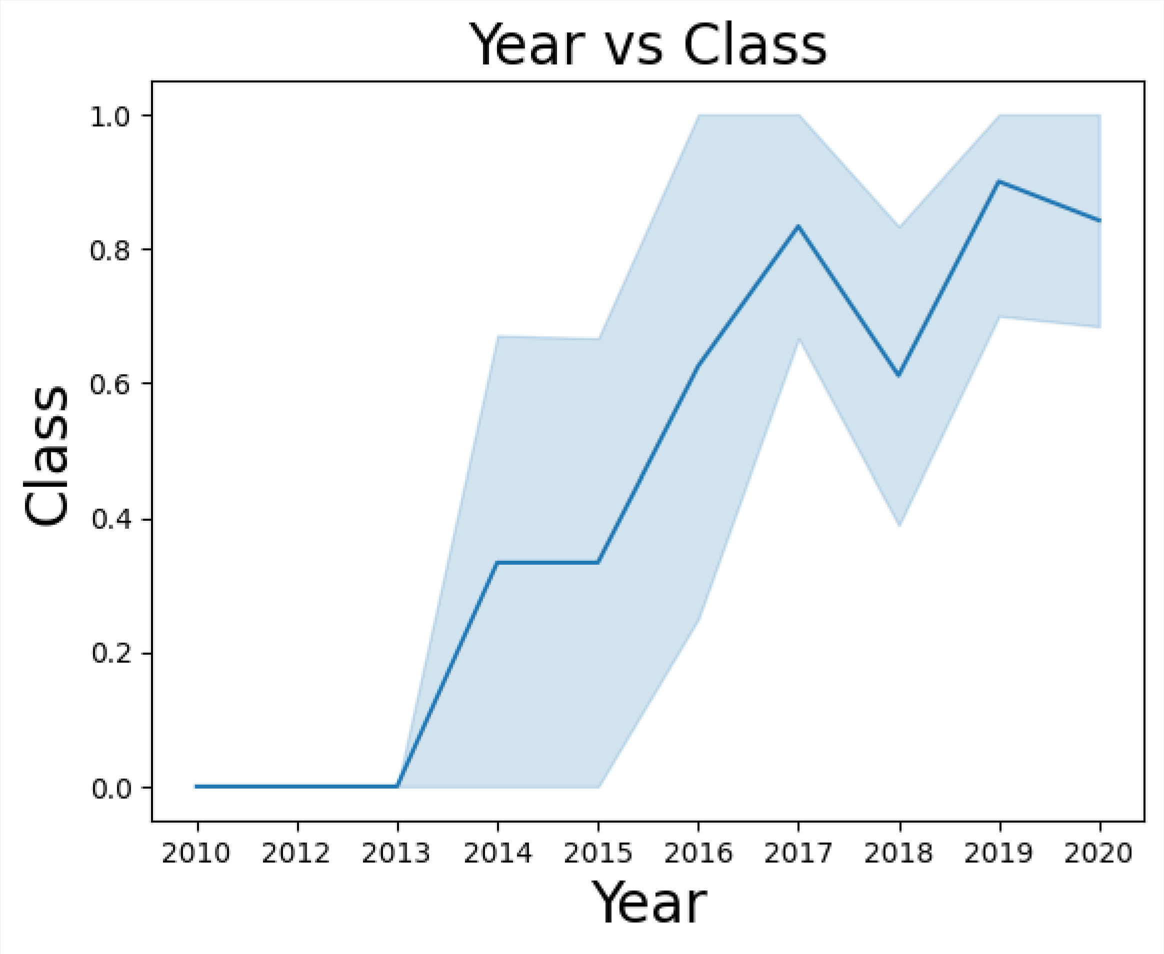
Payload vs. Orbit Type

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.



All Launch Site Names

Used the **DISTINCT** keyword in SQL to display unique launch sites from the SpaceX data.

[10]

```
%sql select distinct "Launch_site" from SPACEXTABLE
```

Python

* sqlite:///my_data1.db
Done.

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

Launch Site Names Begin with 'CCA'

```
%sql select * from SPACEXTABLE where "Launch_site" like 'CCA%' LIMIT 5
```

Python

* sqlite:///my_data1.db
Done.

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success
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
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success

Used the query above to display 5 records where launch sites begin with `CCA`

Total Payload Mass

We calculated the total payload carried by boosters from NASA as 48213 using the query below:

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

```
%sql select sum("PAYLOAD_MASS__KG_") AS "total_payload_mass" from SPACEXTABLE where "Customer" like 'NASA (CRS)%'
```

Python

```
 * sqlite:///my_data1.db
Done.
```

total_payload_mass
48213

Average Payload Mass by F9 v1.1

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

```
%sql select AVG("PAYLOAD_MASS__KG_") AS "mean_payload_mass" from SPACEXTABLE where "Booster_Version" like 'F9 v1.1%'
```

Python

```
 * sqlite:///my_data1.db
Done.
```

mean_payload_mass
2534.6666666666665

The average payload mass carried by booster version F9 v1.1 as 2534.66.

First Successful Ground Landing Date

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

```
%sql select "Date" from SPACEXTABLE where "Mission_Outcome" like 'Success%' order by "Date" desc
LIMIT 1
```

Python

```
 * sqlite:///my_data1.db
Done.
```

Date
2020-12-06

Observed that the dates of the first successful landing outcome on ground pad was 6th December 2015.

Successful Drone Ship Landing with Payload between 4000 and 6000

- Applied the **WHERE** clause to filter boosters that successfully landed on a drone ship.
- Used the **AND** condition to determine successful landings with payload mass greater than 4000 but less than 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 distinct "Booster_Version" from SPACEXTABLE where "PAYLOAD_MASS__KG_" > 4000 and "PAYLOAD_MASS__KG_" < 6000
```

```
 * sqlite:///my_data1.db
Done.
```

Booster_Version
F9 v1.1
F9 v1.1 B1011
F9 v1.1 B1014
F9 v1.1 B1016
F9 FT B1020
F9 FT B1022
F9 FT B1026
F9 FT B1030
F9 FT B1021.2
F9 FT B1032.1
F9 B4 B1040.1
F9 FT B1031.2
F9 B4 B1043.1
F9 FT B1032.2
F9 B4 B1040.2
F9 B5 B1046.2
F9 B5 B1047.2
F9 B5B1054
F9 B5 B1048.3
F9 B5 B1051.2
F9 B5B1060.1
F9 B5 B1058.2
F9 B5B1062.1

Total Number of Successful and Failure Mission Outcomes

List the total number of successful and failure mission outcomes

```
%sql SELECT "Mission_Outcome", COUNT(*) AS "Total" FROM SPACEXTABLE GROUP
BY "Mission_Outcome"
```

```
 * sqlite:///my_data1.db
Done.
```

Mission_Outcome	Total
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Retrieved the total number of successful and failed mission outcomes using SQL.

Boosters Carried Maximum Payload

Used a **subquery** within the **WHERE** clause to retrieve the booster version with the highest

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

```
%sql SELECT Booster_Version FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = ( SELECT MAX
(PAYLOAD_MASS__KG_) FROM SPACEXTABLE )
```

Python

```
 * sqlite:///my_data1.db
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

```
%sql SELECT substr(Date, 6, 2) AS "Month", "Landing_Outcome", "Booster_Version", "Launch_Site" FROM
SPACEXTABLE WHERE Landing_Outcome LIKE 'Failure (drone ship)%' and substr(Date,0,5) = '2015'
```

Python

```
 * sqlite:///my_data1.db
Done.
```

Month	Landing_Outcome	Booster_Version	Launch_Site
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

Used a subquery within the **WHERE** clause to filter the data based on a specific condition. In this case, I retrieved the month, landing outcome, booster version, and launch site from the SPACEXTABLE where the landing outcome was specified as "Failure (drone ship)" and the year was 2015.

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

```
%sql SELECT "Landing_Outcome", count(*) AS "count" FROM SPACEXTABLE WHERE "Date" BETWEEN
'2010-06-04' AND '2017-03-20' GROUP BY "Landing_Outcome" ORDER BY "COUNT" DESC;
```

```
 * sqlite:///my_data1.db
Done.
```

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

This SQL query retrieves the count of different landing outcomes from the SPACEXTABLE between the dates '2010-06-04' and '2017-03-20'. The results are grouped by the landing outcome and sorted in descending order based on the count.

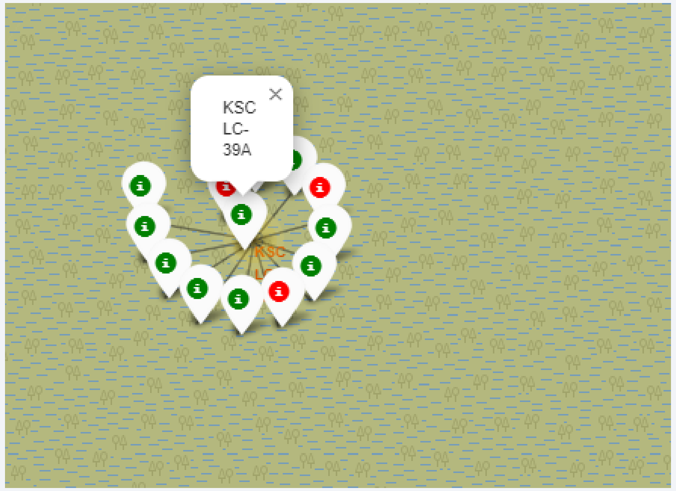
Section 3

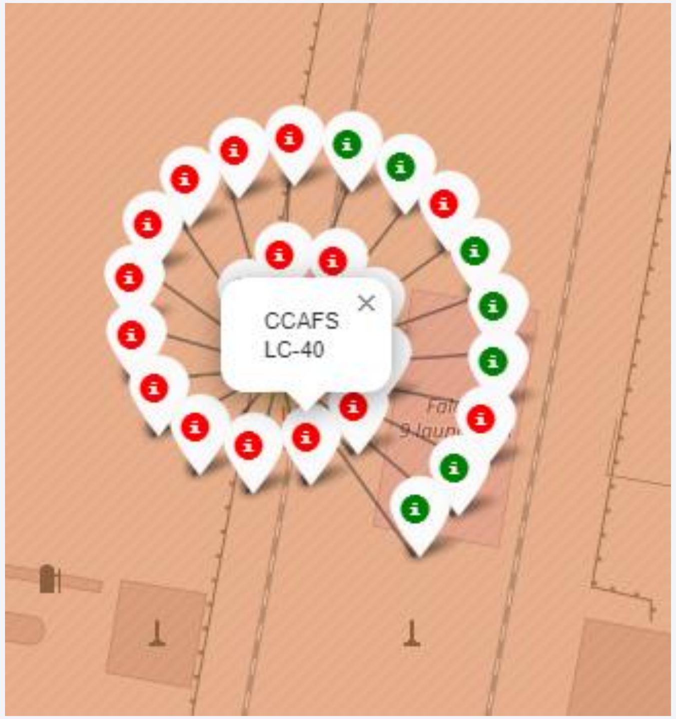
Launch Sites Proximities Analysis

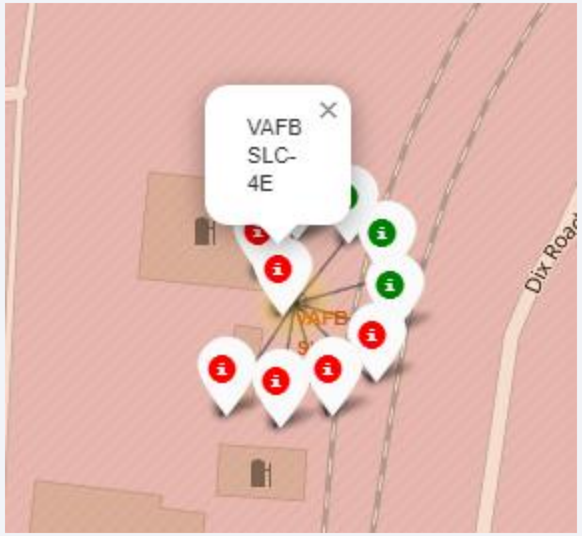
All Launch Sites Global Map Markers

From the map we can see that the SpaceX launch sites are in the USA coasts, Florida and California.

Launch Sites with Colored Markers





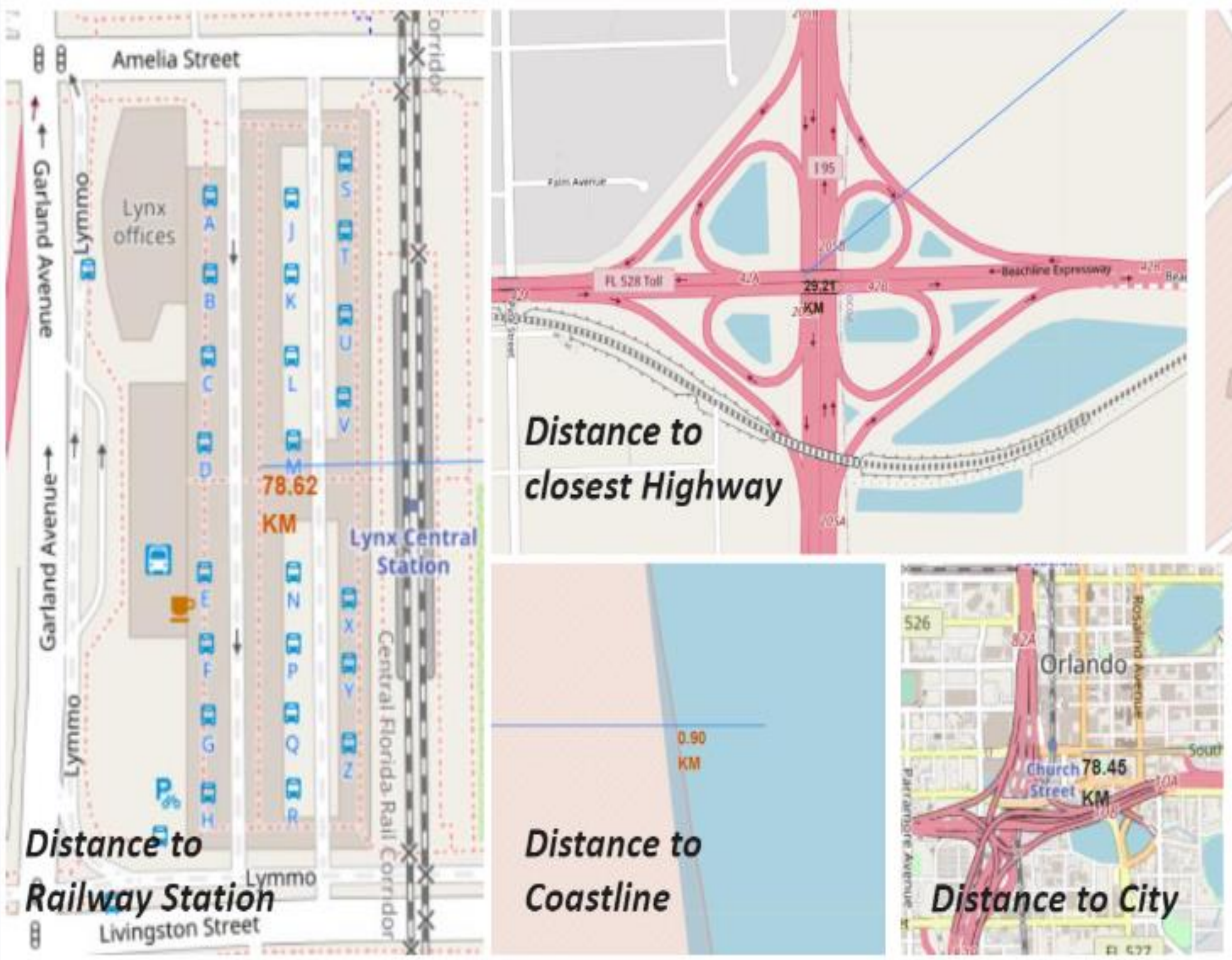


California Launch Sites

Florida Launch Sites

Green Marker shows successful Launches and Red Marker shows failures.

Launch Site Distance to Landmarks

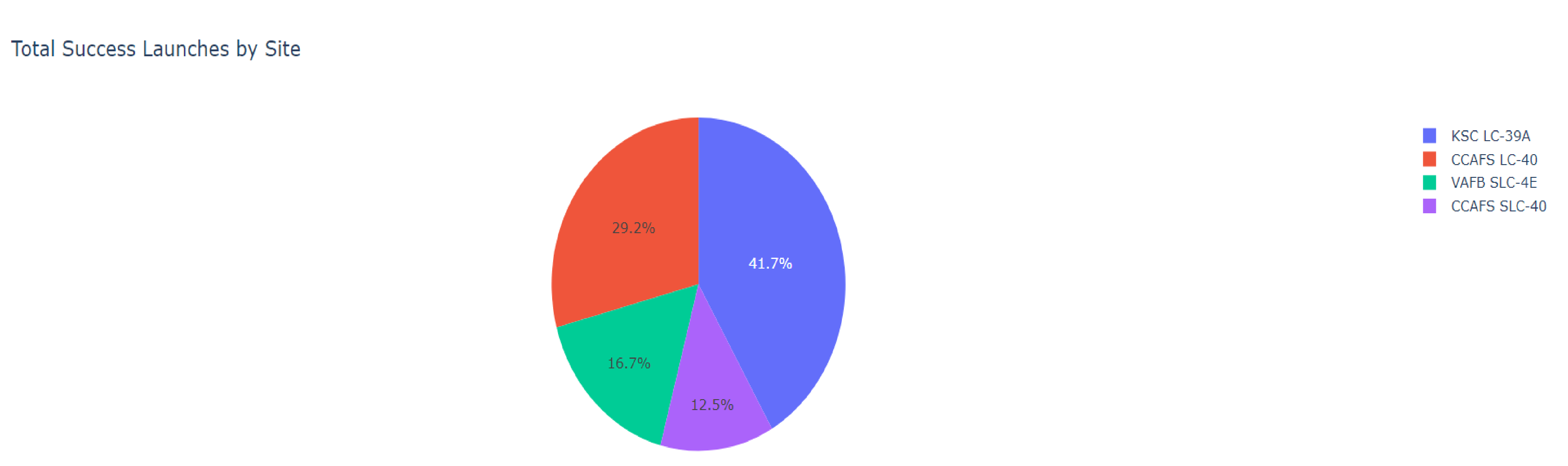


- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

Section 4

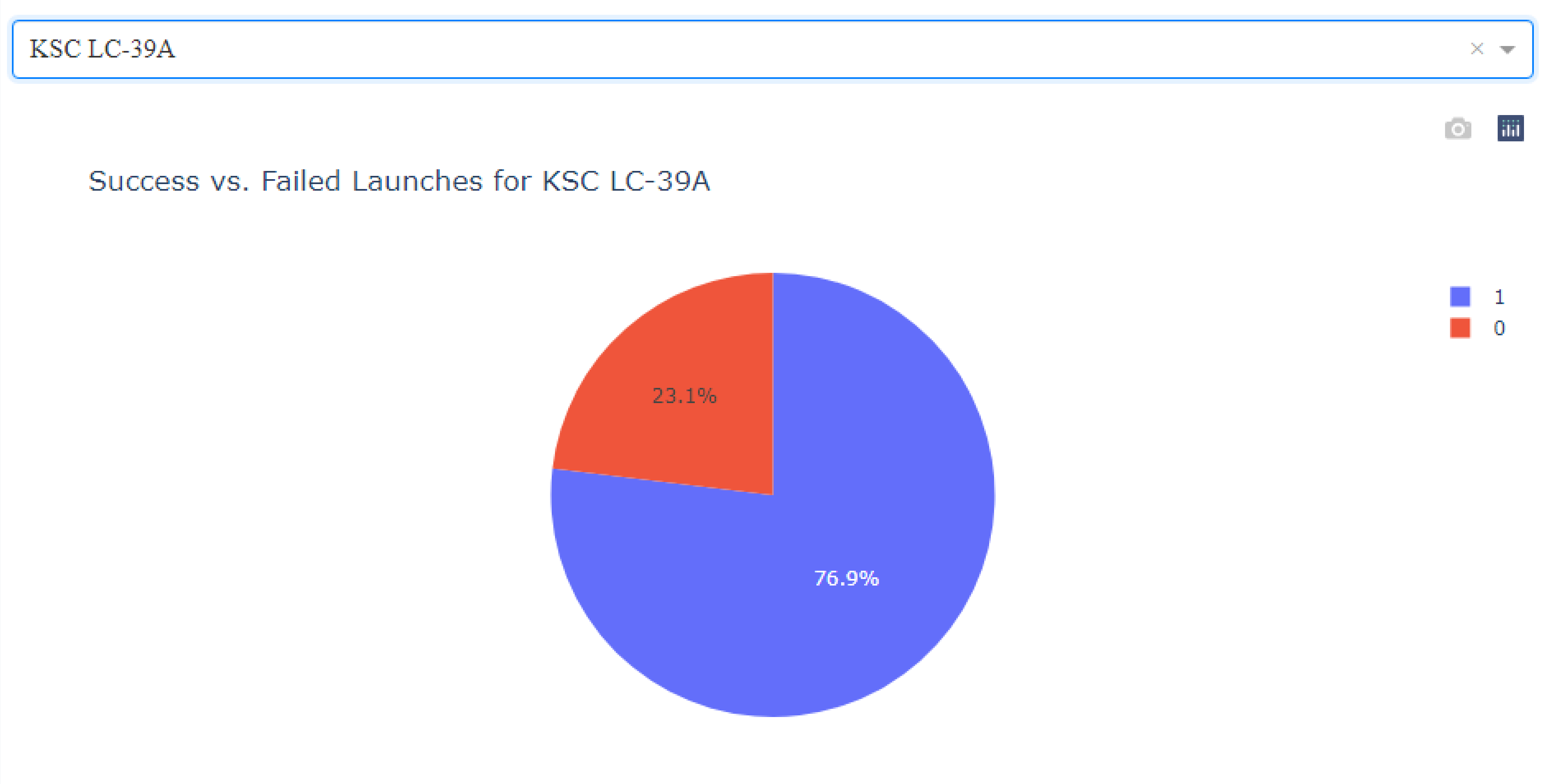
Build a Dashboard with Plotly Dash

Success Percentage Achieved by each Launch Site



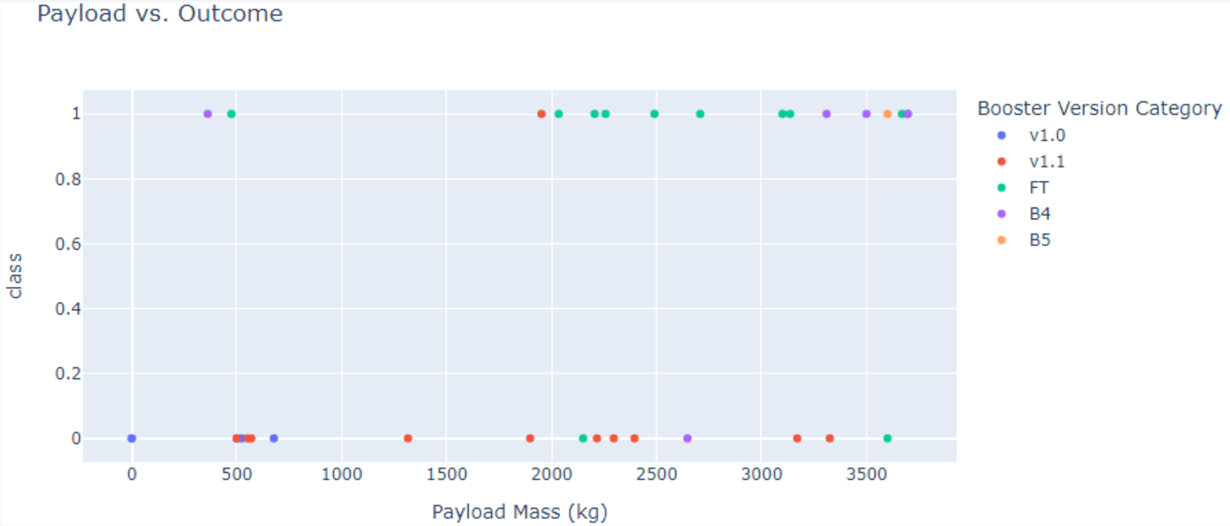
KSC LC-39A had the most successful launches amongst all the sites.

Launch Site with the Highest Launch Success Ratio

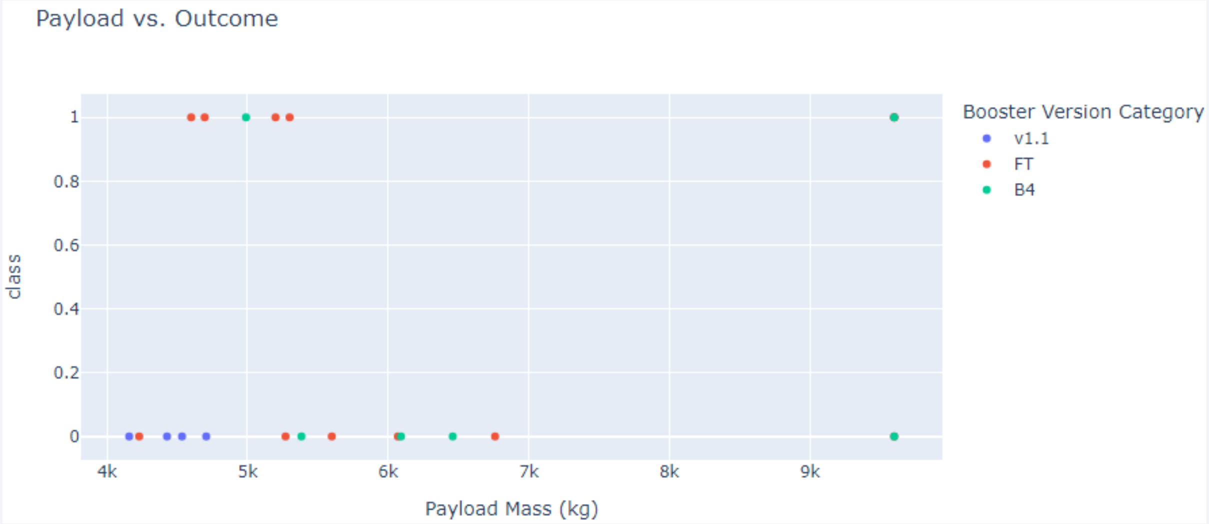


<Dashboard Screenshot 3>

Low Payload (0kg - 4000kg)



High Payload (4000kg – 10000kg)



The success rate is higher for low weighted payloads than high weighted payload.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

```python
models = {'KNeighbors':knn_cv.score(X_test, Y_test),
              'DecisionTree':tree_cv.score(X_test, Y_test),
              'LogisticRegression':logreg_cv.score(X_test, Y_test),
              'SupportVector': svm_cv.score(X_test, Y_test)}

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_)
```

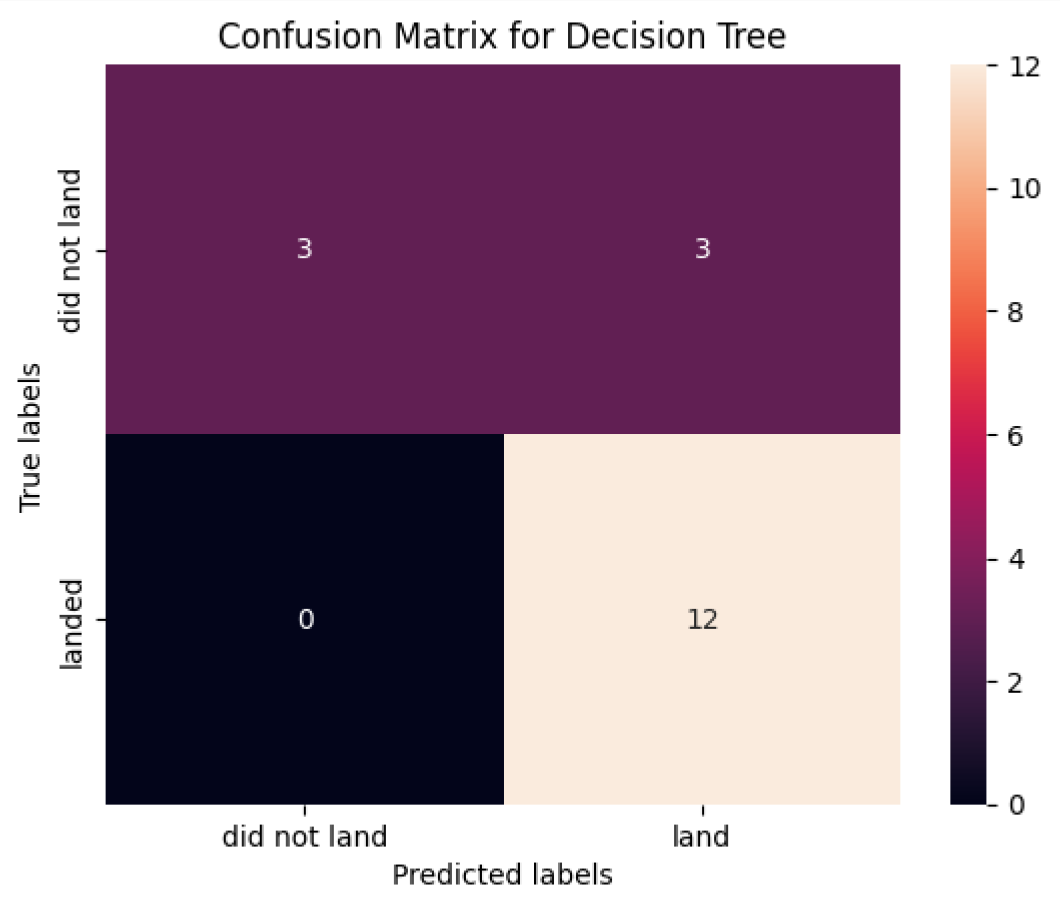
[93] ✓ 0.0s Python

```
Best model is DecisionTree with a score of 0.8333333333333334
Best params is : {'criterion': 'gini', 'max_depth': 2, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_sa
```

Decision tree performs the best amongst all the models with an accuracy of 83.33%.

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

Summary of Conclusions:

- Higher flight numbers at a launch site correlate with higher success rates.
- Launch success rates have shown an increasing trend from 2013 to 2020.
- Orbits ES-L1, GEO, HEO, SSO, and VLEO exhibit the highest success rates.
- KSC LC-39A stands out with the highest number of successful launches among all sites.
- The Decision Tree Classifier emerges as the optimal machine learning algorithm for this analysis.

Thank you!