IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Robert Maron
August 28, 2023

Outline

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

Executive Summary

- Methodologies
  - Data Collection via API's and Web Scraping
  - Exploratory Data Analysis (EDA) with Data Visualization
  - Exploratory Data Analysis (EDA) with SQL
  - Interactive Map with Folium
  - Dashboard with Plotly Dash
  - Predictive Analysis
- Results
  - Exploratory Data Analysis Results
  - Interactive Maps and Dashboard
  - Predictive Results

Introduction

- Background
  - In the rapidly evolving landscape of space technology, SpaceX has emerged as a trailblazer with its revolutionary advancements in rocketry. One of its groundbreaking achievements is the development of the Falcon 9 rocket, known for its reusability. This innovation has the potential to drastically reduce launch costs and reshape the economics of space travel. Central to this innovation is the successful landing of the Falcon 9's first stage after it completes its primary mission.
  - This capstone project aims to predict whether the Falcon 9's first stage will land successfully after launch. The ability to make accurate predictions about the first stage's landing success holds immense significance. Not only does a successful landing validate the reusability concept, saving millions of dollars per launch, but it also opens avenues for cost-effective space travel for various commercial and scientific purposes.
- Problems
  - What Factors Influence Landing Success?
  - Which Machine Learning Algorithm Performs Best?
  - Do Certain Payloads Affect Landing Success?
  - Can Landing Success be Predicted?

Section 1

Methodology

Methodology

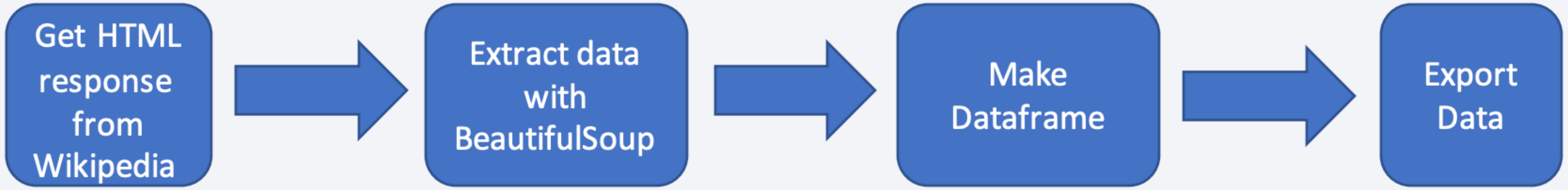
Executive Summary

- Data collection methodology:
  - SpaceX REST API
  - Web Scraping from Wikipedia
- Perform data wrangling
  - Dropping non-numerical columns and unnecessary data
  - One Hot Encoding for classification modeling
- 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

- Datasets are collected from Rest SpaceX API and Web Scraping Wikipedia
- The information obtained by the API includes Rocket Type, Launch Site, and Payload Mass.

- The information obtained by Web Scraping includes Launch Site, Outcome, and Payload Information.



Data Collection – SpaceX API

1. Getting Response from API

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
response = requests.get(spacex_url)
```

2. Convert Response to JSON File

```
data = response.json()
data = pd.json_normalize(data)
```

3. Transform data

```
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
getBoosterVersion(data)
```

4. Create dictionary with data

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
```

5. Create dataframe

```
data = pd.DataFrame.from_dict(launch_dict)
```

6. Filter dataframe

```
data_falcon9 = data[data['BoosterVersion']!='Falcon 1']
```

7. Export to file

```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

- https://github.com/roma-888/AppliedDataScienceCapstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

Data Collection – Scraping

1. Getting Response from HTML

```
response = requests.get(static_url)
```

2. Create BeautifulSoup Object

```
soup = BeautifulSoup(response.text, "html5lib")
```

3. Find all tables

```
html_tables = soup.findAll('table')
```

4. Get column names

```
for th in first_launch_table.find_all('th'):
    name = extract_column_from_header(th)
    if name is not None and len(name) > 0 :
        column_names.append(name)
```

5. Create dictionary

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

6. Add data to keys

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is a
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.stri
                flag=flight_number.isdigit()
```

See notebook for the rest of code

7. Create dataframe from dictionary

```
df=pd.DataFrame(launch_dict)
```

8. Export to file

```
df.to_csv('spacex_web_scraped.csv', index=False)
```

- https://github.com/roma-888/AppliedDataScienceCapstone/blob/main/jupyter-labs-webscraping.ipynb

Data Wrangling

1. **Import Required Packages:** The necessary Python packages (requests, BeautifulSoup, pandas, etc.) were imported to perform web scraping and data manipulation.
2. **HTTP GET Request:** An HTTP GET request was made to fetch the HTML content of the Wikipedia page using the provided static_url.
3. **BeautifulSoup Object Creation:** The fetched HTML content was converted into a BeautifulSoup object to facilitate parsing and navigation through the HTML structure.
4. **Extract Column/Variable Names:** All relevant column names for the launch records were extracted from the third table's header using the find_all function with the th element.
5. **Create an Empty Dictionary:** An empty dictionary called launch_dict was created with keys corresponding to the extracted column names.
6. **Iterate Through Table Rows:** For each table in the HTML (table element), and for each row in the table (tr element), the relevant launch record data was extracted and populated into the launch_dict.
7. **Data Extraction:** Various functions like date_time, booster_version, landing_status, and get_mass were used to extract specific data from the table cells (td elements).
8. **Populate launch_dict:** Extracted data like flight number, date, time, booster version, launch site, payload, payload mass, orbit, customer, launch outcome, and booster landing status were populated into the respective keys of the launch_dict.
9. **Create a DataFrame:** The populated launch_dict was converted into a pandas DataFrame for further analysis.
10. **Export to CSV:** The resulting DataFrame could be optionally exported to a CSV file for future use.

https://github.com/roma-888/AppliedDataScienceCapstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

1. **FlightNumber vs. PayloadMass Scatter Plot:** Observe how the FlightNumber and PayloadMass variables might affect the launch outcome (success or failure). The chart revealed that as FlightNumber increases, the success rate tends to increase, and heavier payloads are associated with lower success rates.
2. **FlightNumber vs. LaunchSite Scatter Plot:** Explore if there's any correlation between the launch site and the success of the first stage landing. The chart indicated differences in success rates across different launch sites.
3. **Payload vs. LaunchSite Scatter Plot:** Analyze whether the launch site has any correlation with the payload mass. The chart highlighted the lack of launches with heavy payloads at the VAFB-SLC launch site.
4. **Orbit Success Rate Bar Chart:** Identify which orbit types have higher success rates. The chart indicated varying success rates for different orbit types.
5. **FlightNumber vs. Orbit Type Scatter Plot:** Investigate whether there's a connection between FlightNumber and Orbit type in terms of launch success. The chart suggested that success in LEO orbit may be related to the number of flights.
6. **Payload vs. Orbit Type Scatter Plot:** Determine if PayloadMass is linked to Orbit type and its impact on success rates. The chart showed differences in success rates for different orbit types.
7. **Yearly Launch Success Rate Trend Line Chart:** Visualize the trend in launch success rates over the years. The chart revealed a positive trend in success rates since 2013.

https://github.com/roma-888/AppliedDataScienceCapstone/blob/main/jupyter-labs-eda-dataviz.ipynb

EDA with SQL

- **Task 1: Display Unique Launch Sites -** Query: Display distinct launch sites in the space mission.
- **Task 2: Display Records with Launch Sites Starting with 'CCA' -** Query: Display 5 records where launch sites begin with 'CCA'.
- **Task 3: Total Payload Mass Carried by NASA (CRS) Boosters -** Query: Display total payload mass carried by NASA (CRS) boosters.
- **Task 4: Average Payload Mass Carried by F9 v1.1 Boosters -** Query: Display average payload mass carried by F9 v1.1 boosters.
- **Task 5: Date of First Successful Landing on Ground Pad -** Query: Display date of first successful landing outcome on a ground pad.
- **Task 6: Booster Versions with Specific Landing Outcome and Payload Mass Range -** Query: Display booster versions with success in drone ship landing and payload mass within a range.
- **Task 7: Total Number of Successful and Failed Mission Outcomes -** Query: Display total number of successful and failed mission outcomes.
- **Task 8: Booster Versions with Maximum Payload Mass -** Query: Display booster versions with maximum payload mass using a subquery.
- **Task 9: Records with Month Names, Failure Landing Outcomes, and Other Details for the Year 2015 -** Query: Display records with month names, failure landing outcomes, booster versions, and launch sites for the year 2015.
- **Task 10: Ranking Landing Outcomes by Count -** Query: Rank landing outcomes' count between specific dates in descending order.

https://github.com/roma-888/AppliedDataScienceCapstone/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- Various map objects were added to the Folium map for visualization:
  1. **Markers**: Represented launch site locations.
  2. **Circles**: Showed proximity areas around launch sites.
  3. **Marker Clusters**: Grouped markers at the same location.
  4. **PolyLines**: Displayed distances between sites and points of interest.
  5. **MousePosition Plugin**: Provided coordinates on mouse hover.
  6. **DivIcon**: Customized marker icons with additional info.
- These objects visually conveyed launch site locations, outcomes, and proximity to key features, aiding in spatial pattern recognition.

https://github.com/roma-888/AppliedDataScienceCapstone/blob/main/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- **Plots/Graphs:**
  - **Pie Chart:** Displays the distribution of successful launches across all launch sites or for a selected launch site.
  - **Scatter Chart:** Plots payload mass against launch success, differentiated by booster version category.
- **Interactions:**
  - **Dropdown Menu:** Allows users to select different launch sites to view success distribution.
  - **Payload Range Slider:** Enables users to filter launches based on payload mass.
- **Purposes:**
  - **Pie Chart:** Provides a quick overview of launch success patterns across sites, aiding in comparative analysis.
  - **Scatter Chart:** Illustrates the relationship between payload mass and launch success, while also considering booster versions.
  - **Dropdown Menu:** Offers site-specific insights into launch success for deeper exploration.
  - **Payload Range Slider:** Allows users to focus on launches within specific payload ranges to identify trends.
- These interactive components together create an insightful and user-friendly dashboard for analyzing SpaceX launch data.

https://github.com/roma-888/AppliedDataScienceCapstone/blob/main/PLOTLY_INTERACTIVE_DASHBOARD.ipynb

Predictive Analysis (Classification)

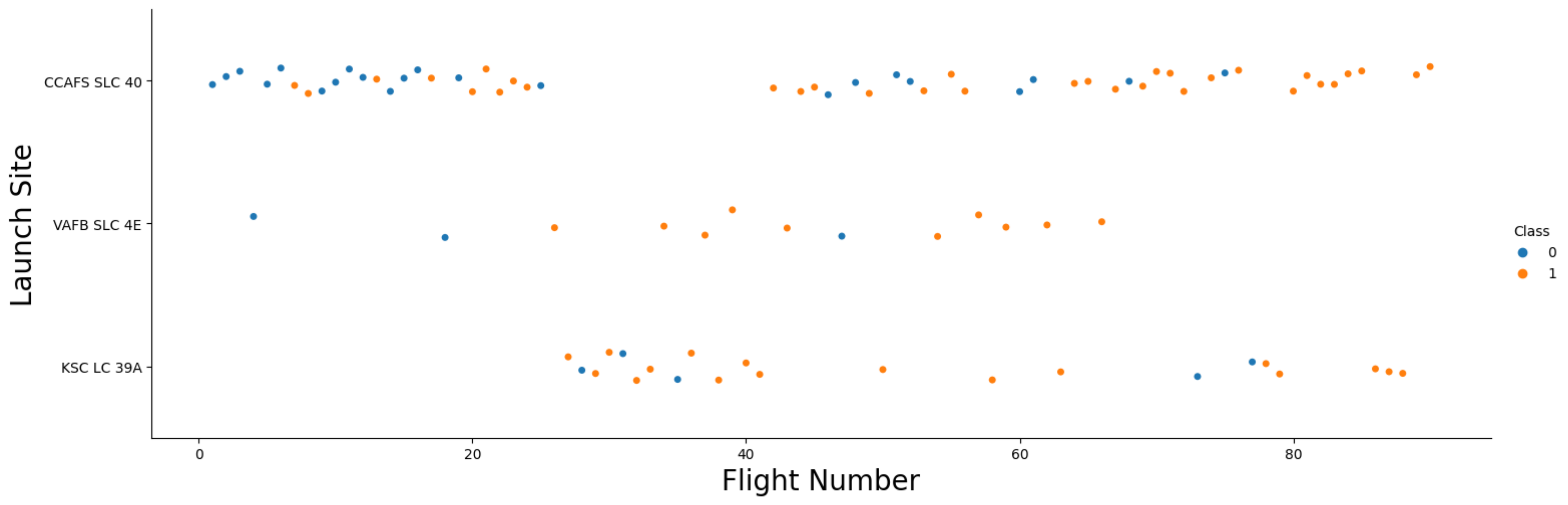
1. **Import Libraries and Load Data:** Begin by importing the necessary libraries for data manipulation and modeling. Load your dataset into a data structure like a DataFrame. This is the initial step in preparing your data for analysis.
2. **Data Preprocessing:** Clean the data by handling missing values and removing duplicates. Split the dataset into features (X) and the target variable (y). Further split the data into training, validation, and test sets to assess the model's performance. Scaling features, if needed, helps algorithms that are sensitive to varying scales.
3. **Model Training and Hyperparameter Tuning:** Choose a machine learning algorithm, such as a RandomForestClassifier, and train it on the training data. Adjust hyperparameters like the number of trees in the forest, the maximum depth of trees, and the minimum samples required to split nodes. These hyperparameters influence the model's performance.
4. **Model Evaluation and Iterative Improvement:** Assess the model's performance using metrics like accuracy, precision, recall, and F1-score on the validation set. This step helps identify potential areas of improvement. Consider altering feature engineering or trying different algorithms based on the evaluation results.
5. **Cross-Validation:** Implement cross-validation to gain a more robust understanding of your model's performance. This involves dividing the training data into multiple subsets (folds) and training/evaluating the model on different combinations of training and validation folds. The average performance provides a more accurate estimate of how the model might perform on unseen data.
6. **Final Evaluation and Deployment:** Evaluate the model on the test set, which simulates real-world performance. Assess various metrics to ensure the model meets your desired criteria. If satisfied, you can deploy the model for making predictions on new, unseen data.

https://github.com/roma-888/AppliedDataScienceCapstone/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Section 2

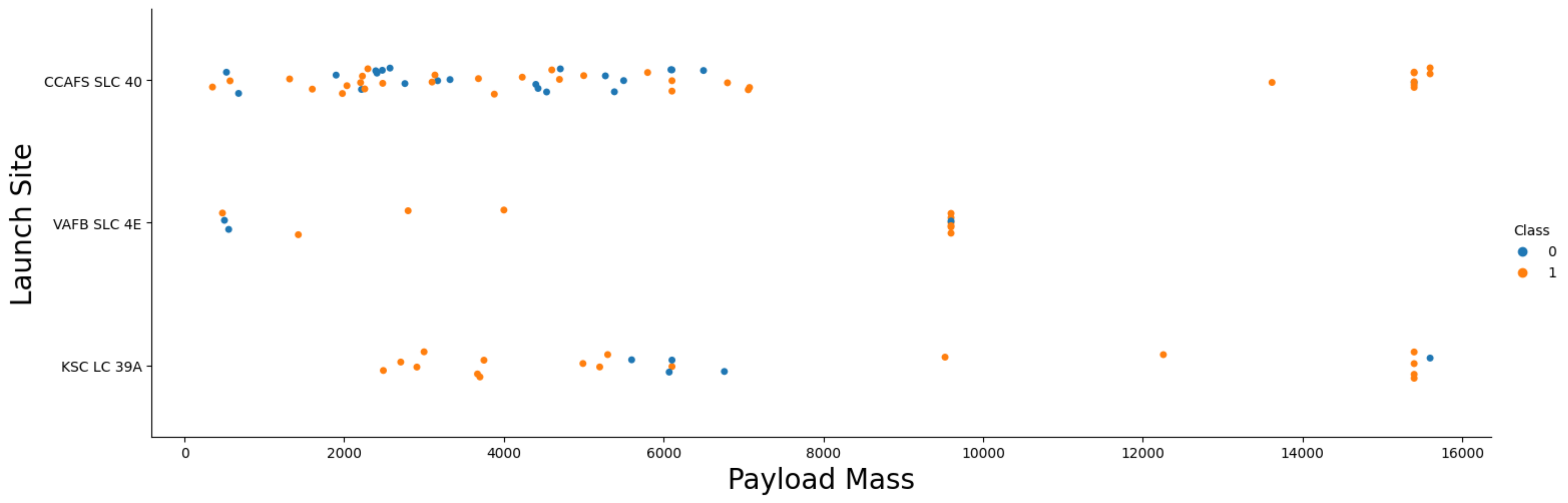
Insights drawn from EDA

Flight Number vs. Launch Site



- For each site, the success rate increases as the Flight Number increases.
- Flight Number also has *some* direct correlation to the Launch Site

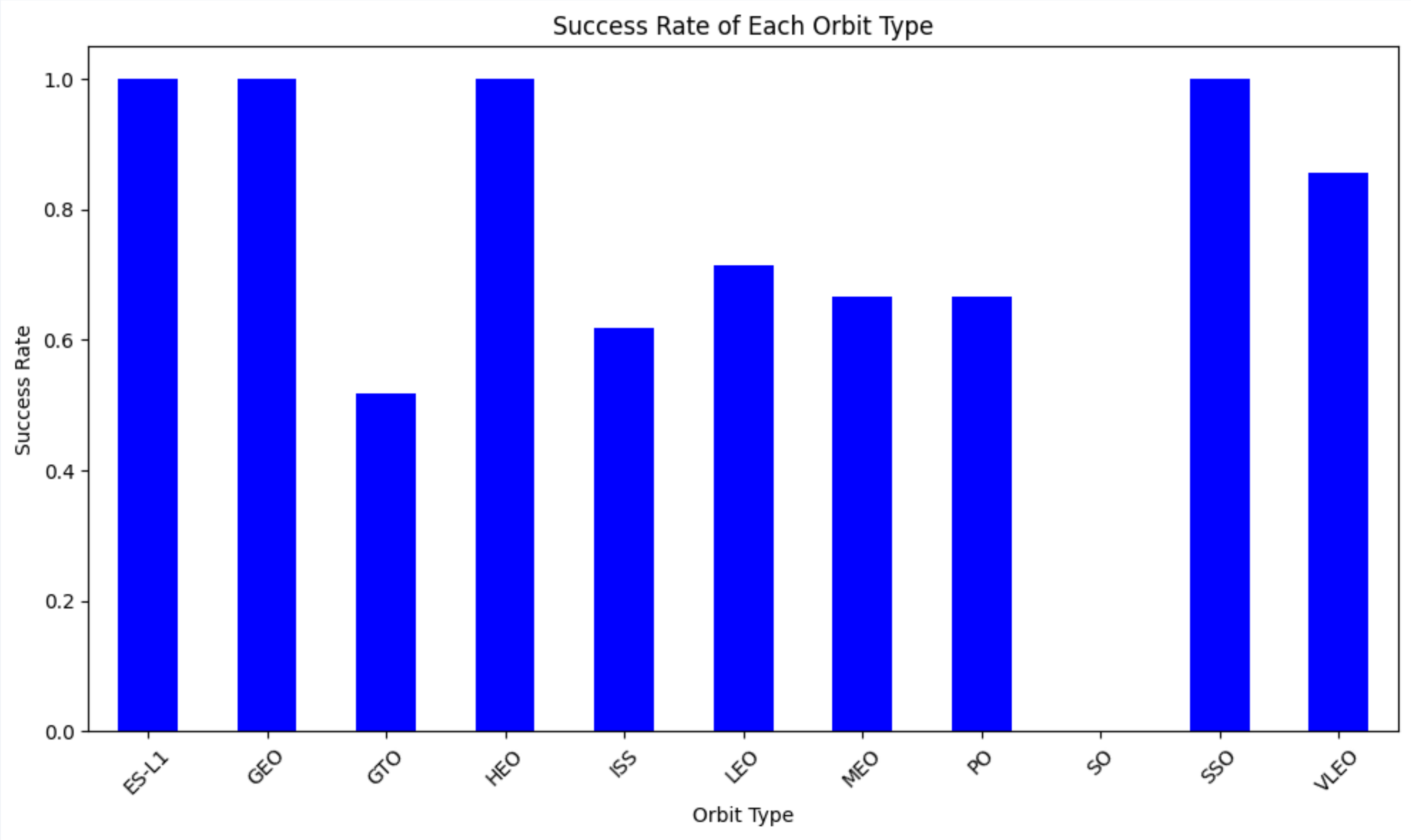
Payload vs. Launch Site



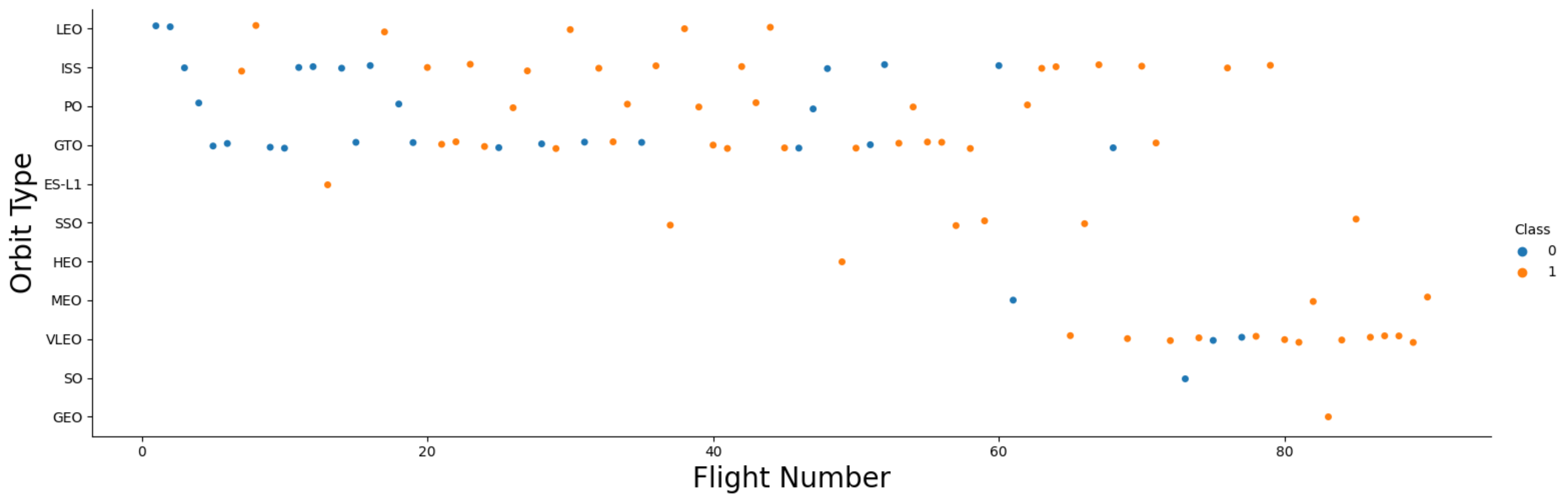
- The success of a landing might depend on the launch site, and in some cases, a heavier payload could contribute to a higher likelihood of success. However, it's important to note that an excessively heavy payload could also lead to a failed landing attempt.

Success Rate vs. Orbit Type

- This plot illustrates the success rates for various orbit types. Notably, ES-L1, GEO, HEO, and SSO orbits exhibit the highest rates of success.

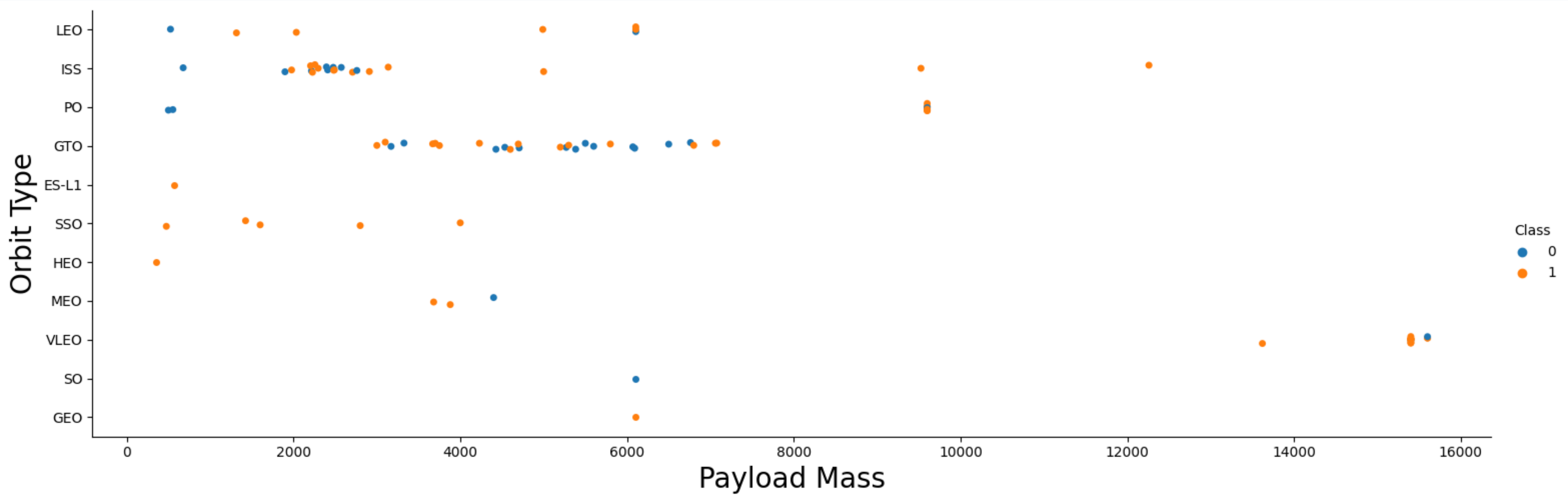


Flight Number vs. Orbit Type



- It's evident that the success rate in the LEO orbit rises as the number of flights increases. Conversely, in orbits such as GTO, the success rate appears unaffected by flight numbers. However, we can hypothesize that the notable success rates in orbits like SSO or HEO might be attributed to insights gained from previous launches in different orbits.

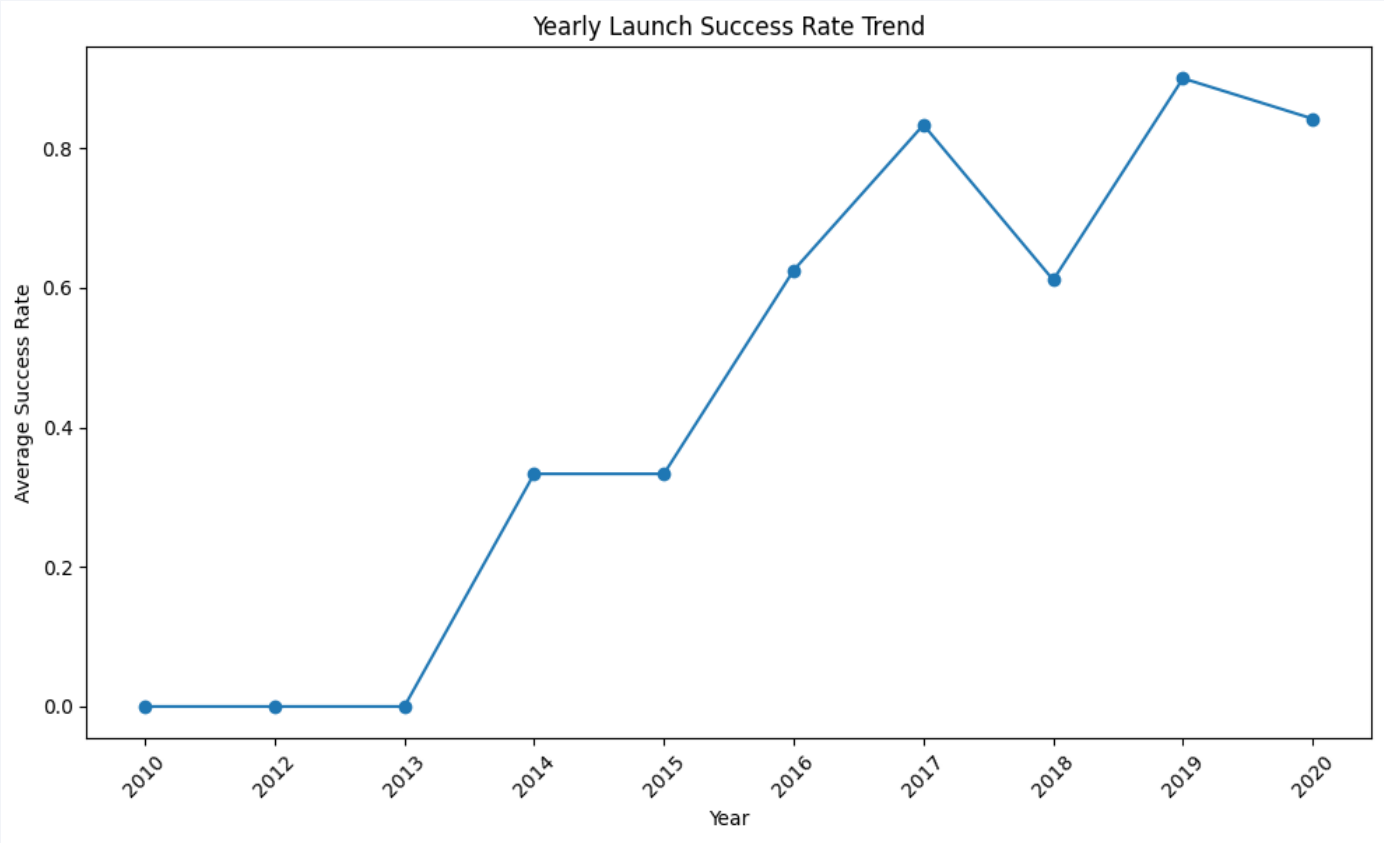
Payload vs. Orbit Type



- Payload weight significantly impacts the success rate of launches within specific orbits. Notably, in the LEO orbit, heavier payloads correlate with higher success rates. Conversely, for the GTO orbit, reducing payload weight enhances the likelihood of a successful launch.

Launch Success Yearly Trend

- Starting from 2013, there is a discernible upward trend in the success rate of SpaceX rocket launches.



All Launch Site Names

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

```
%%sql
SELECT DISTINCT Launch_Site
FROM SPACEXTABLE;
```

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

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

- This query retrieves distinct launch site names from the dataset.

Launch Site Names Begin with 'CCA'

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

```
%%sql
SELECT *
FROM SPACEXTABLE
WHERE Launch_Site LIKE 'CCA%'
LIMIT 5;
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX
2010-08-12	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
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)
2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)
2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)

- This query retrieves records from the dataset where the launch site starts with 'CCA', limited to 5 records.

Total Payload Mass

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

```
%%sql
SELECT SUM(PAYLOAD_MASS__KG_) AS 'NASA (CRS) Total Payload Mass (kg)'
FROM SPACEXTABLE
WHERE Customer LIKE 'NASA (CRS)'
```

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

NASA (CRS) Total Payload Mass (kg)
45596

- This query calculates the sum of payload mass for missions with the customer 'NASA (CRS)'.

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 'F9 v1.1 Average Payload Mass (kg)'
FROM SPACEXTABLE
WHERE Booster_Version LIKE 'F9 v1.1'
```

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

F9 v1.1 Average Payload Mass (kg)
2928.4

- This query calculates the average payload mass for missions with the booster version 'F9 v1.1'.

First Successful Ground Landing Date

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

Hint:Use min function

```
%%sql
SELECT MIN(Date)
FROM SPACEXTABLE
WHERE Mission_Outcome LIKE 'Success' AND Landing_Outcome LIKE '%ground pad%
```

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

MIN(Date)
2015-12-22

- This query retrieves the minimum date from missions with a successful landing outcome on a ground pad.

Successful Drone Ship Landing with Payload between 4000 and 6000

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

```
%%sql
SELECT Booster_Version
FROM SPACEXTABLE
WHERE Landing_Outcome LIKE 'Success (drone ship)' AND PAYLOAD_MASS__KG_ > 4
```

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

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

- List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

Total Number of Successful and Failure Mission Outcomes

List the total number of successful and failure mission outcomes

```
%%sql
SELECT
    SUM(CASE WHEN Landing_Outcome LIKE '%Success%' THEN 1 ELSE 0 END) AS Su
    SUM(CASE WHEN Landing_Outcome LIKE '%Failure%' THEN 1 ELSE 0 END) AS Fa
FROM SPACEXTABLE
```

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

Successful_Landings	Failed_Landings
61	10

- Calculate the total number of successful and failure mission outcomes

Boosters Carried Maximum Payload

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

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

```
 * 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

- List the names of the booster which have carried the maximum payload mass

2015 Launch Records

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.

```
%%sql
SELECT substr(Date, 6, 2) AS Month, Landing_Outcome, Booster_Version, Launc
FROM SPACEXTABLE
WHERE Landing_Outcome LIKE 'Failure (drone ship)' AND Date LIKE '%2015%'
```

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

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

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

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)) between the date 2010-06-04 and 2017-03-20, in descending order.

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

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

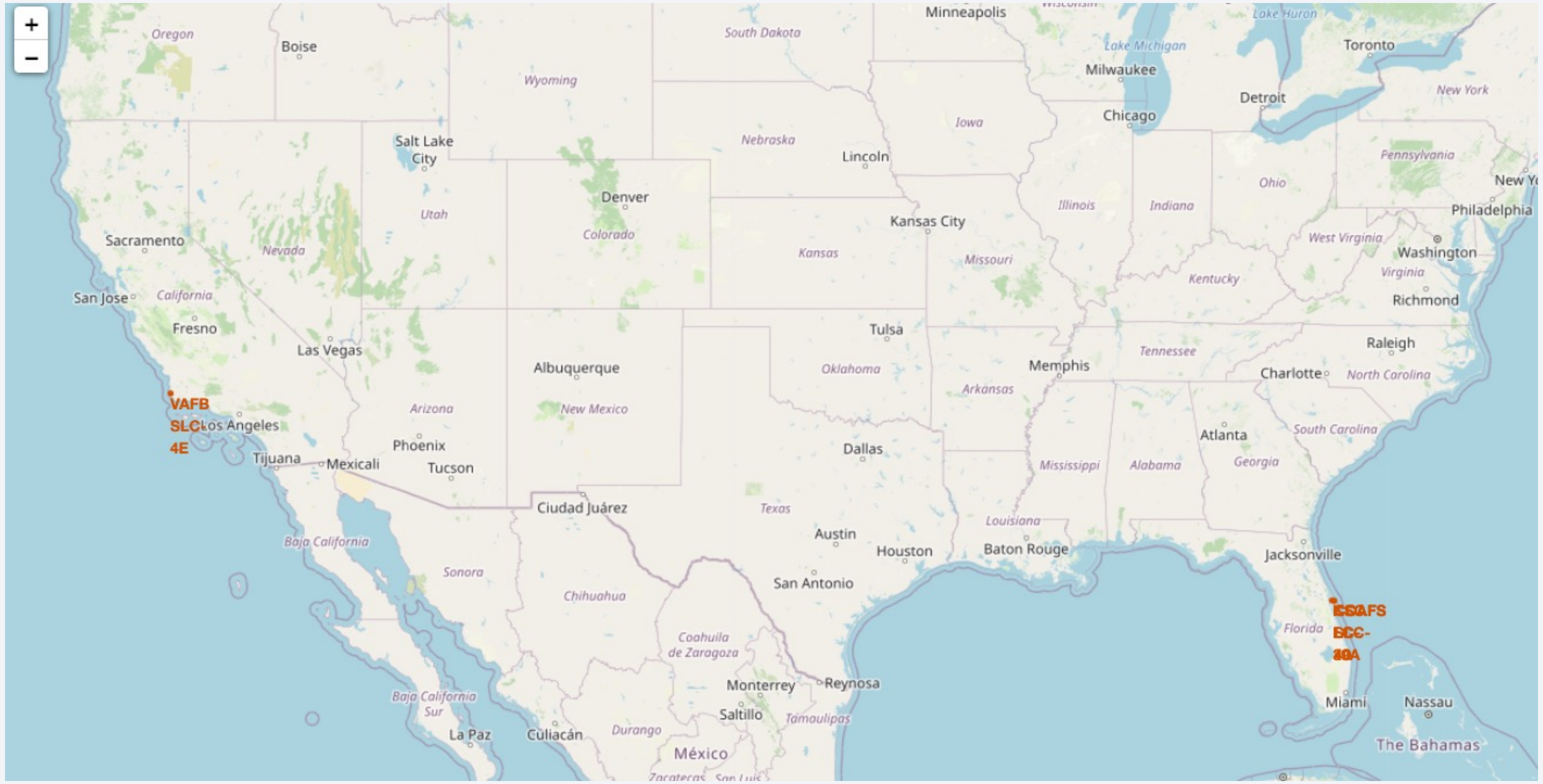
Landing_Outcome	Outcome_Count
No attempt	10
Success (ground pad)	5
Success (drone ship)	5
Failure (drone ship)	5
Controlled (ocean)	3
Uncontrolled (ocean)	2
Precluded (drone ship)	1
Failure (parachute)	1

- Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

Section 3

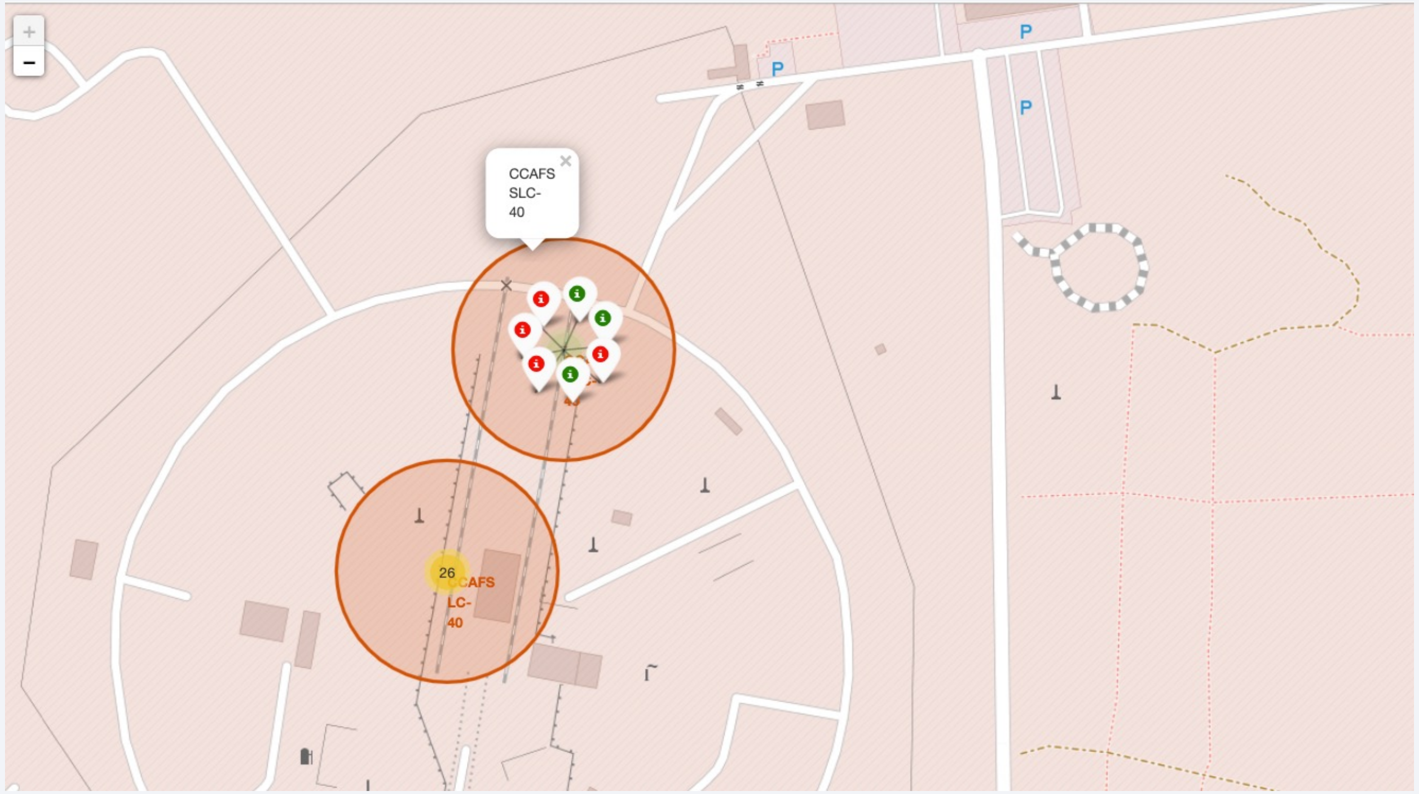
Launch Sites Proximities Analysis

Folium Map – Ground Stations



- It's evident that SpaceX launch sites are situated along the coastal regions of the United States.

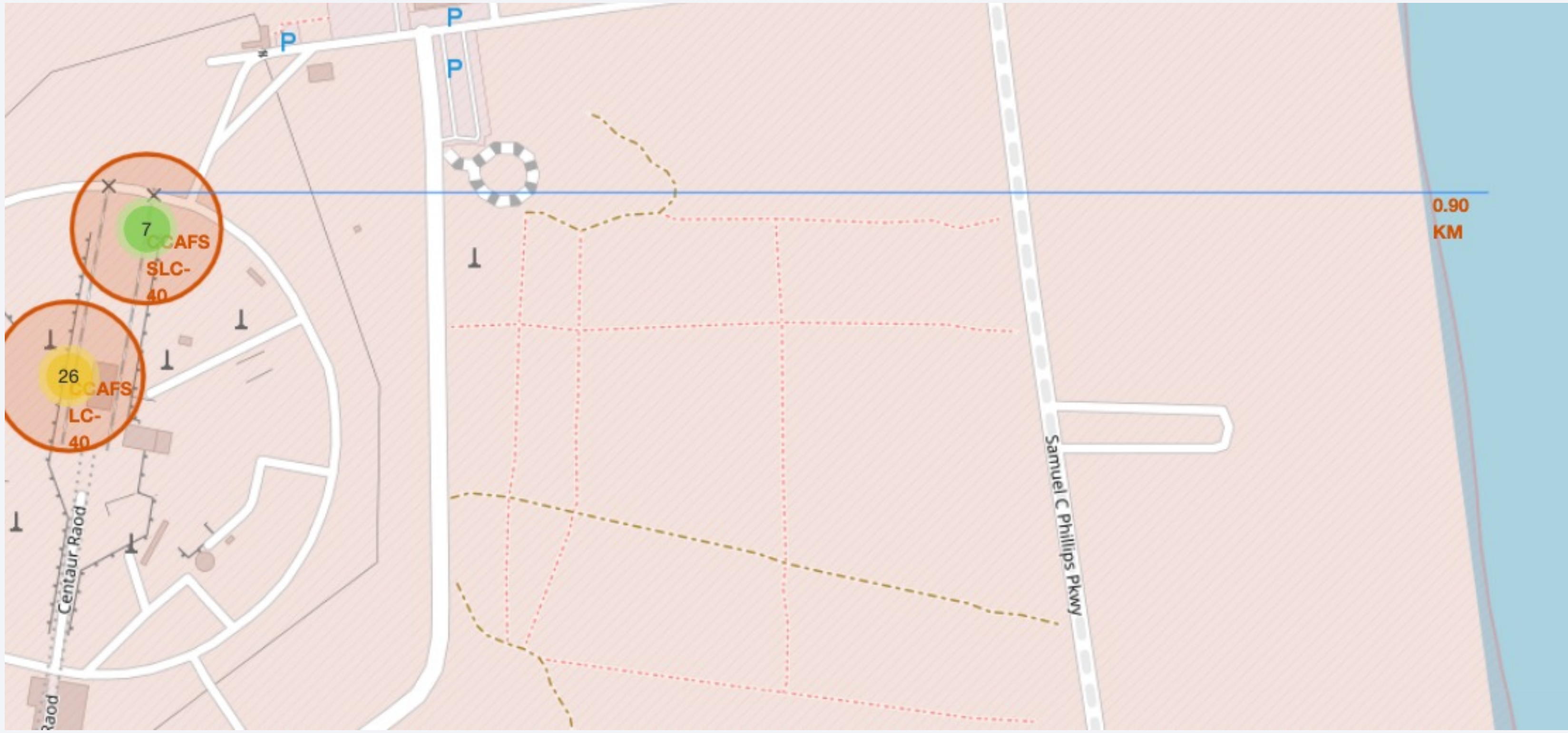
Folium Map – Colored Markers



- Successful launches are denoted by green markers, while unsuccessful launches are indicated by red markers.

Folium Map – Proximity Distances

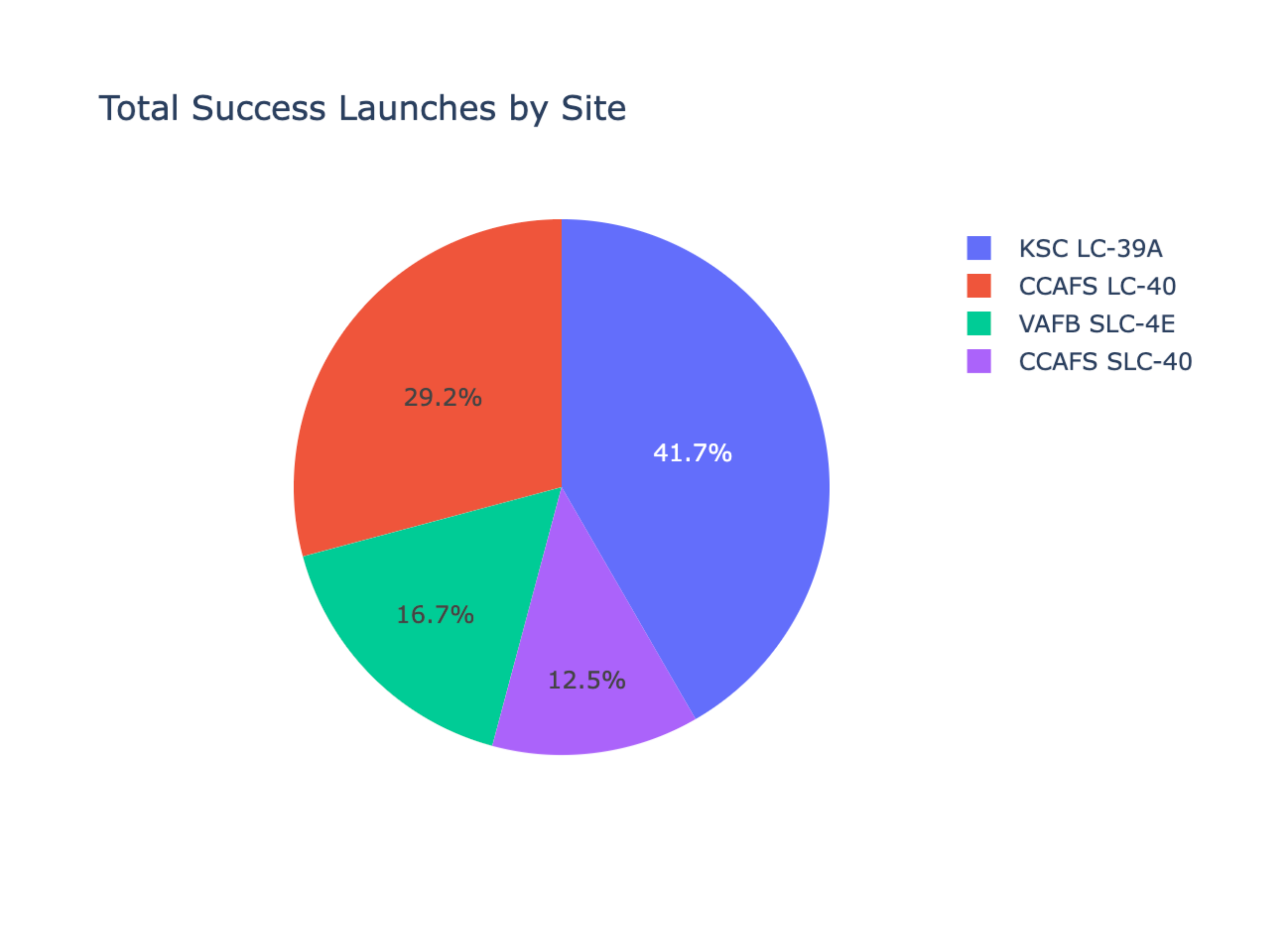
- Are launch sites in close proximity to railways?
  - Yes
- Are launch sites in close proximity to highways?
  - Yes
- Are launch sites in close proximity to coastline?
  - Yes
- Do launch sites keep certain distance away from cities?
- No



Section 4

Build a Dashboard with Plotly Dash

<Dashboard – Total Success Launches by Site

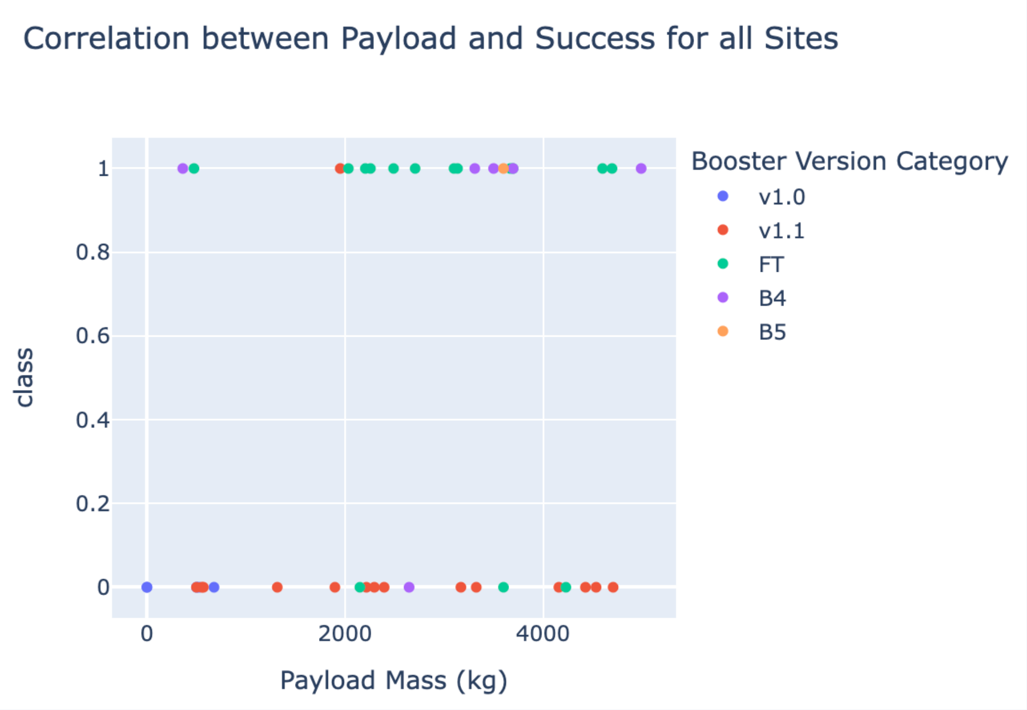


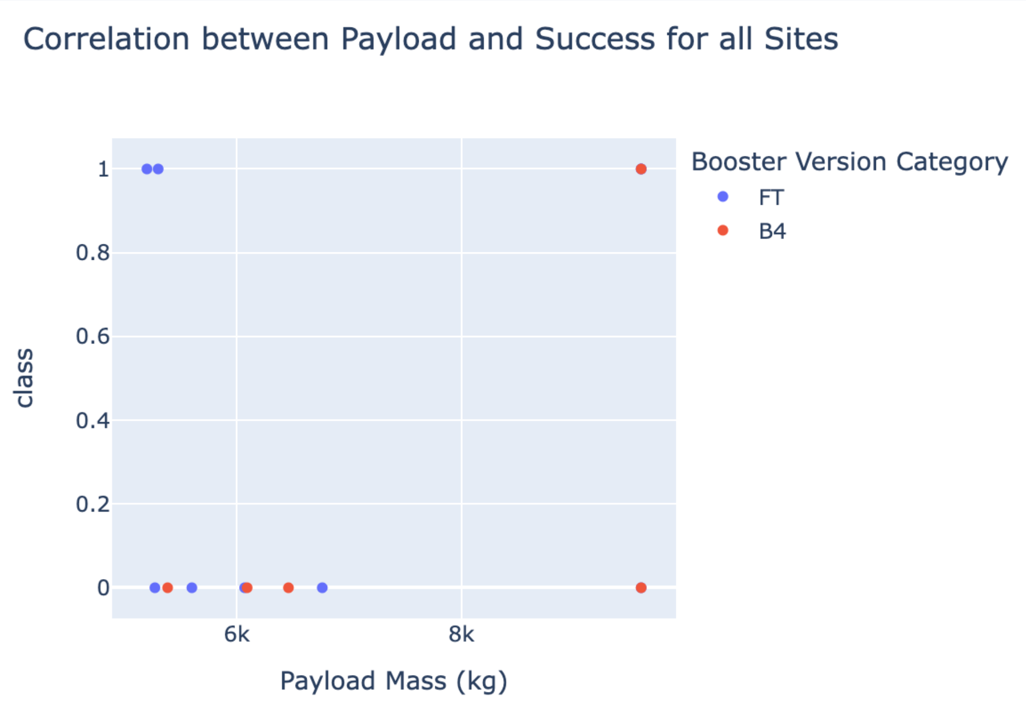
- We see that KSC LC-39A has the best success rate of launches.

<Dashboard Screenshot 2>

- We see that KSC LA-39A had the highest percentage of successful launches at 76.9%.

<Dashboard Screenshot 3>



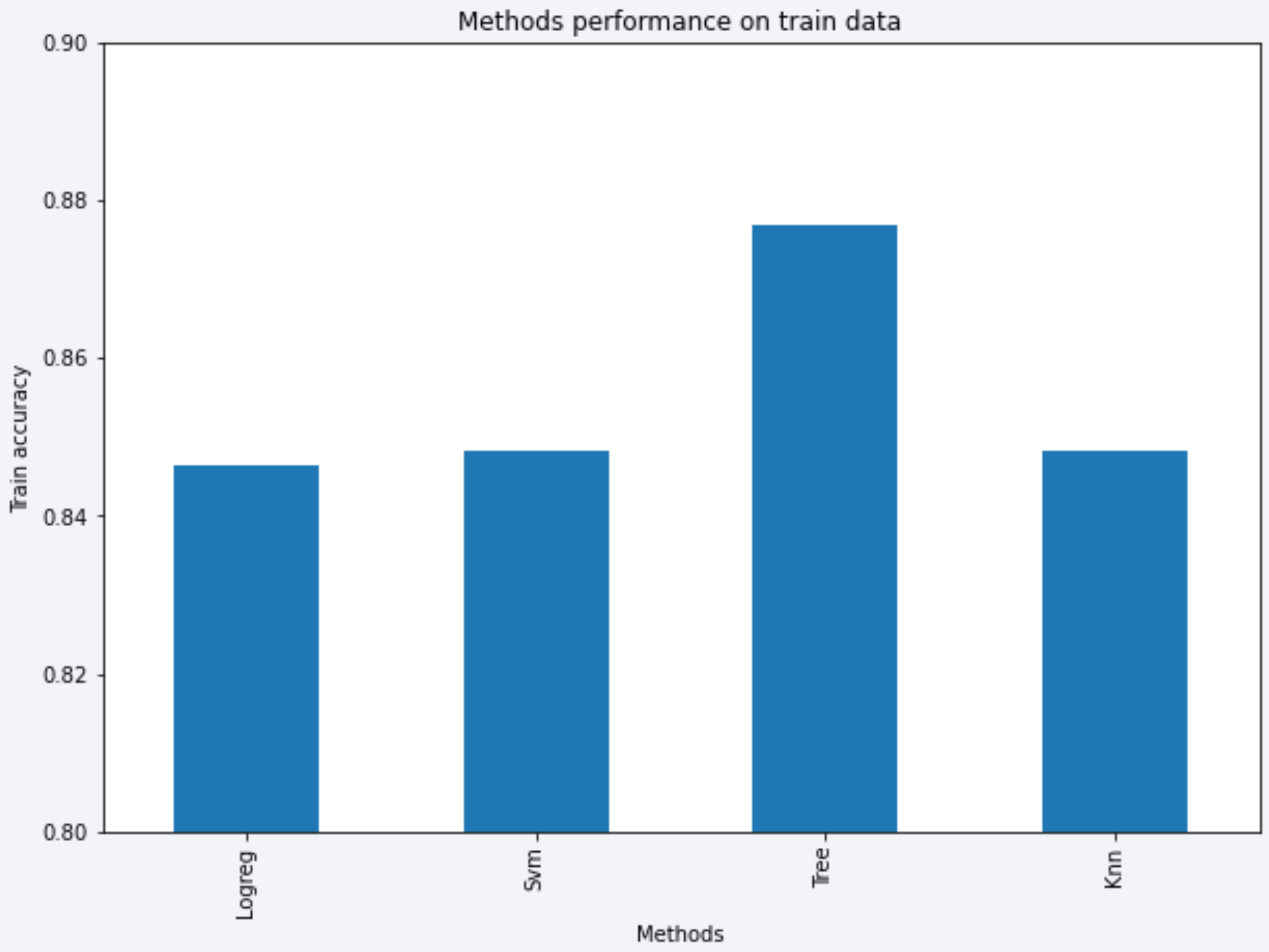


- Low weighted payloads have a better success rate than the heavy weighted payloads.

Section 5

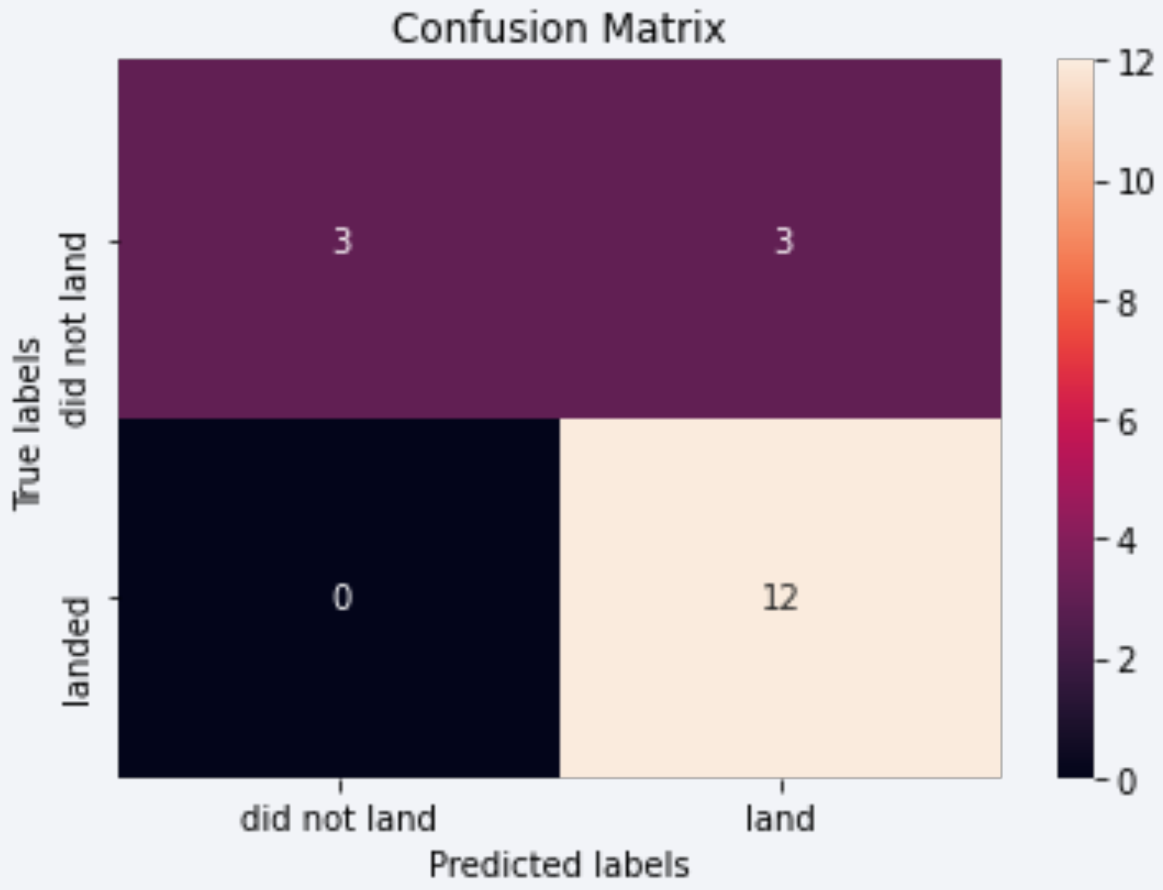
Predictive Analysis (Classification)

Classification Accuracy



- The decision tree seems to perfrom the best, having a higher train accuracy than LogReg, SVM, and KNN.

Confusion Matrix



- All models produced the same Confusion Matrix, showing there is no best model.

Conclusions

- Mission success can be attributed to various factors, including the launch site, the chosen orbit, and notably, the historical count of previous launches. It is reasonable to assume that knowledge gained over successive launches has contributed to transitioning from launch failures to successful ones.
- Among the orbits, GEO, HEO, SSO, and ES-L1 stand out with the highest success rates. Payload mass also plays a role in mission success, varying depending on the specific orbit. Certain orbits necessitate light or heavy payloads, with lighter payloads generally exhibiting better success rates than heavier counterparts.
- While the dataset doesn't provide insight into why certain launch sites perform better than others, KSC LC-39A stands out as the most successful. Addressing this issue would require additional data, such as atmospheric conditions.
- In the context of this dataset, the Decision Tree Algorithm emerged as the preferred model for predicting success. Although the test accuracy across all models is identical, the Decision Tree Algorithm was favored due to its superior training accuracy.

Thank you!