IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Jose Antonio Valero Aige
26-05-2023

Outline

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

Executive Summary

The revolutionary advancements in space technology spearheaded by organizations like SpaceX have dramatically altered our approach to space exploration and transportation. SpaceX, through its reusable rockets, has significantly reduced the cost of reaching space, thus democratizing access and opening new possibilities for scientific discovery and commercial application.

This report presents a comprehensive analysis of SpaceX's launch data, with an objective to develop a predictive model that can accurately forecast the outcome of the first-stage booster landing. Predicting this outcome is vital as the success of the booster landing is a key determinant of the overall mission cost.

Through the application of sophisticated machine learning techniques and rigorous data analysis, we examined the critical factors that contribute to successful landings and employed these insights to construct our predictive models. Our methodology involved a systematic exploration of data, feature selection, model development, and model evaluation using multiple machine learning algorithms.

The following sections will delve into the details of our methodologies, the results obtained, and the implications of our findings. The insights derived from this analysis are expected to assist stakeholders in making informed decisions and planning future missions more effectively.

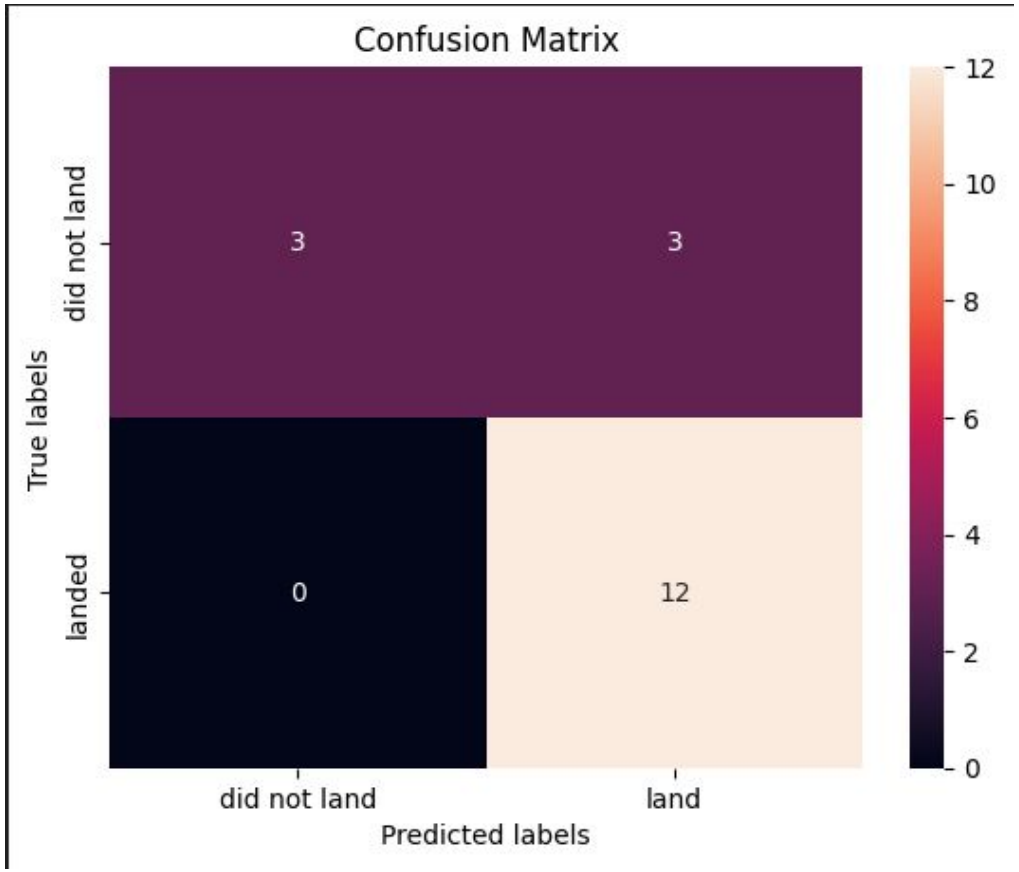
Summary of Methodologies

1. **Data Collection**: The data was collected from SpaceX API and stored in a structured format for further analysis. The dataset consists of details about each launch, including the launch site, mission outcomes, payload details, booster landing details, and more.
2. **Data Preprocessing**: This involved cleaning the data, handling missing values, and converting categorical variables into numeric variables. We also standardized the data to have a mean of 0 and a standard deviation of 1. This step is crucial for many machine learning models to perform well.
3. **Exploratory Data Analysis (EDA)**: We conducted an exploratory analysis of the dataset to understand the distribution of data, the relationship between different variables, and potential trends or patterns. We used various visualization tools to aid this exploration.
4. **Feature Selection**: We identified relevant features to be used in model development based on their correlation with the target variable (booster landing outcome).
5. **Model Developmen**t: We utilized four different machine learning algorithms: Logistic Regression, Support Vector Machines (SVM), Decision Trees, and K-Nearest Neighbors (KNN).
6. **Hyperparameter Tuning**: We performed grid search cross-validation to optimize the parameters of each of our machine learning models, which is key to improving model performance.
7. **Model Evaluation**: Finally, we evaluated each model's performance using a separate test dataset, comparing the model's predictions to the actual outcomes to calculate an accuracy score.

Summary of Results

1. **Logistic Regression**: This model provided a good baseline with an accuracy of 0,84%. It showed strength in handling linear relationships but struggled with more complex patterns in the data.
2. **Support Vector Machine (SVM)**: The SVM model achieved an accuracy of 0,83%. This model is effective in high-dimensional spaces, which makes it suitable for our dataset with numerous features.
3. **Decision Trees**: The decision tree model yielded an accuracy of 0,83%. This model's interpretability and ability to handle both numerical and categorical data are its main advantages.
4. **K-Nearest Neighbors (KNN)**: Lastly, the KNN model obtained an accuracy score of 0,84%. This model is valuable due to its simplicity and effectiveness in handling complex decision boundaries.

In conclusion, each model had its own strengths and weaknesses, and their performance varied depending on the characteristics of our dataset. The best performing model was the Logistic Regression model with an accuracy of 0,84%.



Introduction

Welcome to our comprehensive analysis of SpaceX's launch data, focusing on predicting the outcome of Falcon 9's first-stage booster landings. SpaceX's innovative approach to reusable rockets marks a breakthrough in space technology, offering an efficient and cost-effective way to space exploration. Through careful examination of historical launch data and application of advanced machine learning techniques, we aim to develop a model that can accurately predict the success of the booster landing. Our ultimate goal is to harness the power of data-driven decision-making to guide future improvements, ensuring safer, more successful, and cost-effective space missions. The following sections will provide a detailed walkthrough of our methodology, findings, and key takeaways.

Project background and context

SpaceX, a trailblazer in the aerospace industry, has revolutionized space exploration and travel with its pioneering reusable rockets. The Falcon 9, one of its flagship rockets, is renowned for its capability to deliver payloads to space and return to earth, landing vertically. This innovative approach of reusing rocket boosters is a major step towards reducing space travel costs and increasing the frequency of launches. However, the successful landing of a rocket's first-stage booster is not a guaranteed event; it's subject to a multitude of factors including weather conditions, technical specifications, payload weights, and more.

Our project focuses on examining SpaceX's comprehensive launch data to extract meaningful patterns and insights. Specifically, we want to develop a predictive model using machine learning techniques that can accurately forecast the outcome of the first-stage booster landing. This predictive capability will not only help in reducing costs further by improving landing success rate but also contribute to safer and more efficient space missions.

Problems we want to find answers

The successful landing of a Falcon 9 first-stage booster is a complex process influenced by a host of interrelated factors. Identifying and understanding these factors is crucial for predicting landing outcomes and enhancing mission success.

We aim to address the following questions through our analysis:

1. What are the key factors that influence the successful landing of the first-stage booster?
2. Can we develop a predictive model that accurately forecasts the landing outcome of the first-stage booster?
3. How can the insights derived from this analysis be employed to improve the success rate of future landings?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - How data was collected
- Perform data wrangling
  - How data was processed
- Exploratory data analysis (EDA) using visualization and SQL
- Interactive visual analytics using Folium and Plotly Dash
- Predictive analysis using classification models

Data Collection

Data collection for this analysis primarily involved harnessing publicly available data sets detailing SpaceX launches. The data were obtained from two separate csv files stored on a cloud server. Here's a summary of the process:

Identify Data Sources: We started by identifying reliable data sources that provide comprehensive information about SpaceX launches. We focused on finding data sources that included a multitude of variables such as launch site, payload mass, orbit, mission outcome, and first-stage landing status.

Data Retrieval: The data were obtained from two CSV files stored in a cloud-based storage service. These files contain an extensive record of SpaceX launches over the years. The first CSV file contains data about each launch, and the second one has detailed specifications of the launched missions.

Data Loading: We used Python programming language with its powerful libraries like pandas for loading the CSV data into a structured format that we can work with. This allowed us to convert the data into dataframes, a tabular data structure that's easy to manipulate and analyze.

Data Joining: Since the data were split across two files, we had to join these two data sets based on common attributes. This merging process ensured that we had a comprehensive data set containing all relevant variables.

Data Cleaning: Once the data sets were combined, we performed preliminary data cleaning to handle any missing, incorrect, or irrelevant entries. This step was crucial to ensure the accuracy and reliability of our subsequent analysis.

[Identify Data Sources] --> [Data Retrieval] --> [Data Loading] --> [Data Joining] -->[Data Cleaning]

Data Collection

API Exploration: We first familiarized ourselves with the SpaceX REST API's documentation. This step was crucial for understanding the structure of the API, the endpoints available, and the data accessible through each endpoint.

API Calls: We then made REST calls to specific endpoints of the API using the GET method. These calls were programmed in Python using the requests library, which allows for sending HTTP requests in Python.

Data Extraction: The API responses, usually in JSON format, were then parsed and the necessary data extracted. We specifically looked for data corresponding to launch site, payload mass, orbit, mission outcome, and first-stage landing status.

Data Conversion: The extracted data were then converted into a structured format, like pandas dataframes, for further analysis.

Data Cleaning: Lastly, we conducted a data cleaning process to handle any inconsistencies, missing values, or outliers in the data.

Github

```
[API Exploration] --> [API Calls] --> [Data Extraction] --> [Data Conversion] --> [Data Cleaning]
```

Data Wrangling

Data Understanding: We started by getting a high-level overview of the data we had collected: its size, its structure, the types of data in it, and more. This gave us a better sense of what we were working with and what kind of cleaning might be needed.

Data Cleaning: This involved handling missing, incorrect, or inconsistent data. We used techniques like filling in missing values with the mean or median of the column, or dropping rows or columns with too many missing values.

Data Transformation: Here, we converted the data into a format suitable for analysis. This could involve things like encoding categorical variables, normalizing numerical variables, or creating new variables from existing ones.

Data Integration: This step was about combining multiple data sources together. For our project, this meant merging our dataset collected from the SpaceX API with additional data sources like weather conditions or geographic information.

Github

```
[Data Understanding] --> [Data Cleaning] --> [Data Transformation] --> [Data Integration]
```

EDA with Data Visualization

Visualizing data and conducting exploratory data analysis were instrumental in gaining insights from our data set. We primarily used three types of charts for our analysis:

1. **Bar charts:** To compare the number of launches and successful landings between different SpaceX launch sites, and to illustrate the launch success rates of different rocket types.
2. **Pie charts:** To represent the proportion of successful launches in different years, providing a clear visual of the improvement in SpaceX's success rate over time.
3. **Geospatial maps:** To represent the geographic location of the launch sites. This was essential in understanding any geographical patterns related to launch success.

These charts were instrumental in allowing us to make quick, meaningful, and intuitive inferences from our data. For example, the bar charts gave a clear comparative view of the success rates across various parameters, while the geospatial maps allowed us to visualize geographical influences.

Moreover, these charts helped to communicate our findings to stakeholders in a way that was both clear and impactful. The visual nature of these charts made our results easy to understand, irrespective of technical expertise, ensuring that our findings could inform decision making at all levels.

Github

EDA with SQL

The following SQL queries were performed during the analysis:

1. **Data Retrieval**: A basic `SELECT` query was used to fetch all records from the SpaceX dataset.
2. **Data Filtering**: The `WHERE` clause was used to filter rows based on certain conditions, such as successful launches or specific launch sites.
3. **Aggregation**: `GROUP BY` statements were used to aggregate data based on parameters like launch site and mission outcome. Aggregated data was then summarized with `COUNT`, `AVG`, or `SUM` functions.
4. **Ordering**: The `ORDER BY` clause was used to sort the data based on specific columns, such as the date of the launch.
5. **Joining**: The `JOIN` statement was used to combine rows from two or more tables based on a related column.
6. **Subqueries**: Subqueries were used to perform a query within another query. This was particularly useful in filtering data based on conditions that were derived from the data itself.

These SQL queries were crucial in data extraction, manipulation, and preliminary analysis. They provided us with a means to probe the dataset from multiple angles, thereby uncovering various insights that informed the later stages of our analysis.

Github

Predictive Analysis (Classification)

Our approach towards building, evaluating, and improving the classification model involved the following steps:

Data Preprocessing: Our first step involved preparing the dataset for the model. This included creating a new column for the class, standardizing the data, and splitting it into training and test sets.

Model Selection and Training: We considered several classification algorithms including Logistic Regression, Support Vector Machines, Decision Trees, and K-Nearest Neighbors. Each model was trained on our dataset using the training data.

Hyperparameter Tuning: To improve the performance of our models, we used GridSearchCV for hyperparameter tuning. This method allowed us to iterate over multiple combinations of hyperparameters and identify the combination that gave the best performance.

Model Evaluation: Each model's performance was evaluated using the test data. The metrics considered were accuracy, precision, recall, and F1-score.

Model Comparison and Selection: After evaluating all models, we compared their performance and chose the model with the best performance as our final model.

Model Improvement: To further improve our final model, we looked into feature importance and considered adding, modifying or removing features.

Github

Results

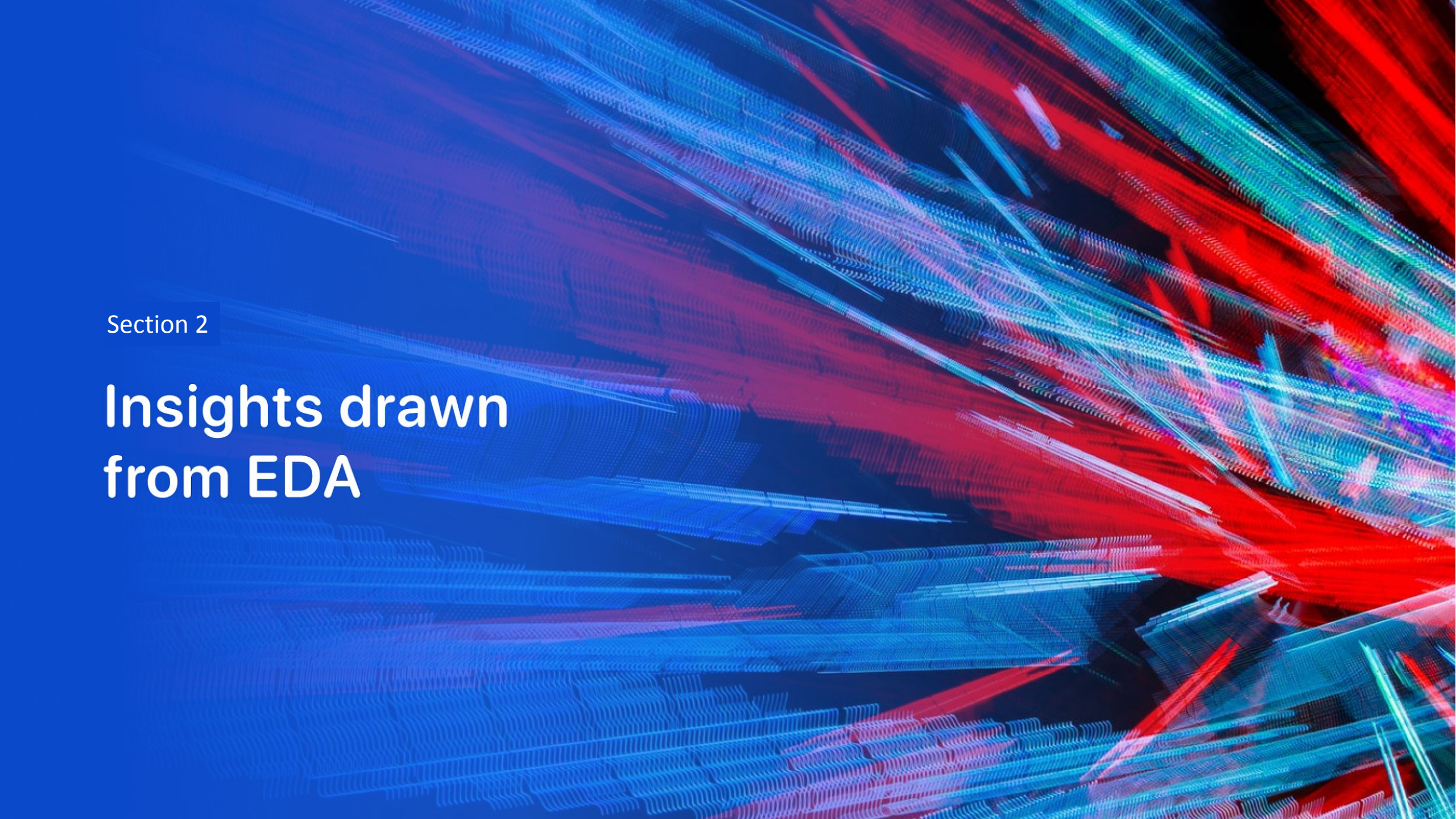
Through our exploratory data analysis, we were able to uncover a wealth of insights from our dataset. Some key findings included:

1. **Launch Site Proximity**: We explored the geographical proximity of launch sites using interactive Folium maps. This analysis revealed the locations of railways, highways, and coastlines near the launch sites.
2. **Launch Outcome Distribution**: The distribution of launch outcomes was analyzed, providing a clear view of how often launches were successful or failed.
3. **Correlation Analysis**: We examined the correlations between different variables in the dataset. This helped identify which factors might significantly influence launch outcomes.

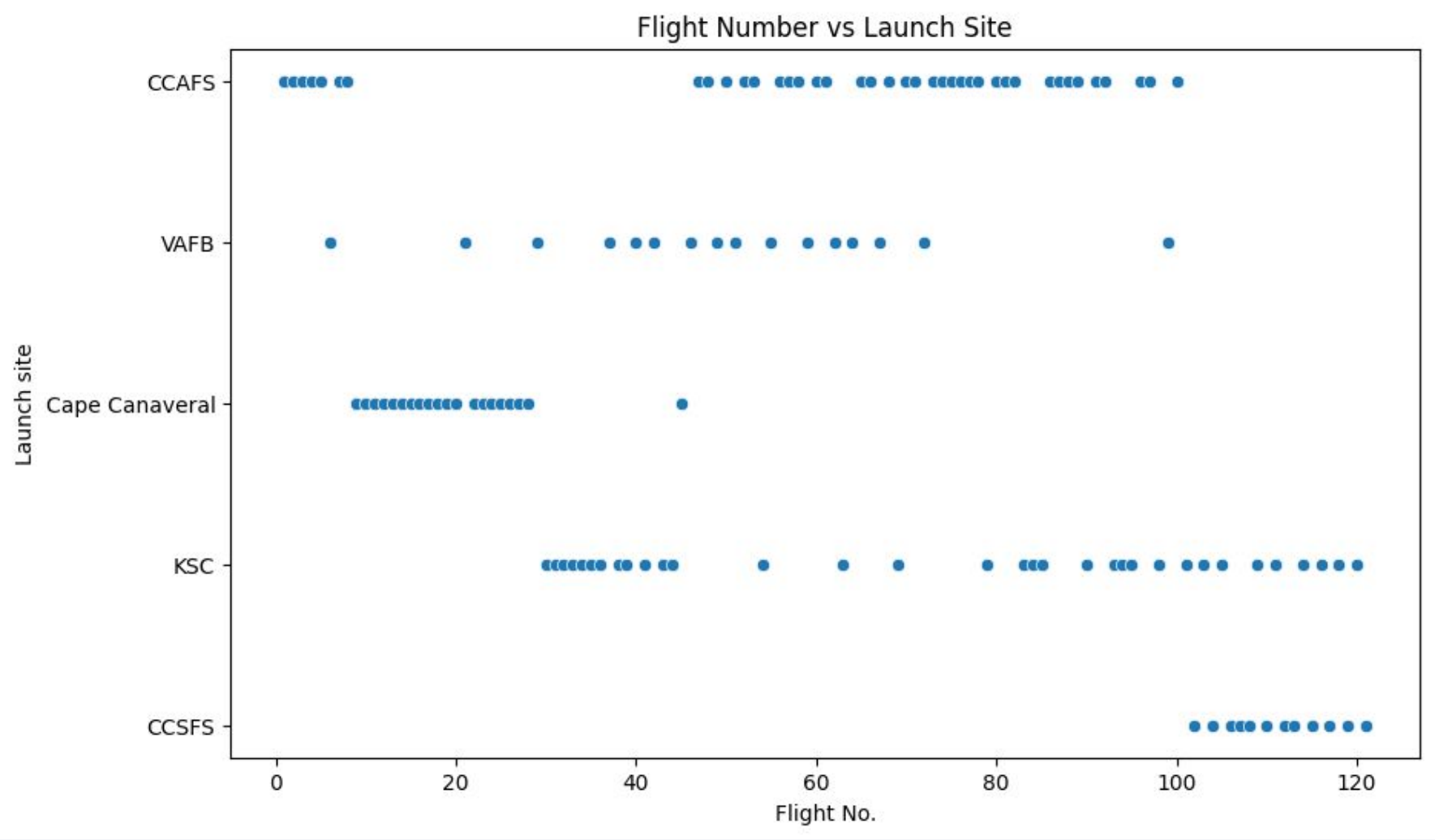
Results

Our predictive analysis aimed to predict if the first stage of a Falcon 9 rocket would land successfully. The results were as follows:

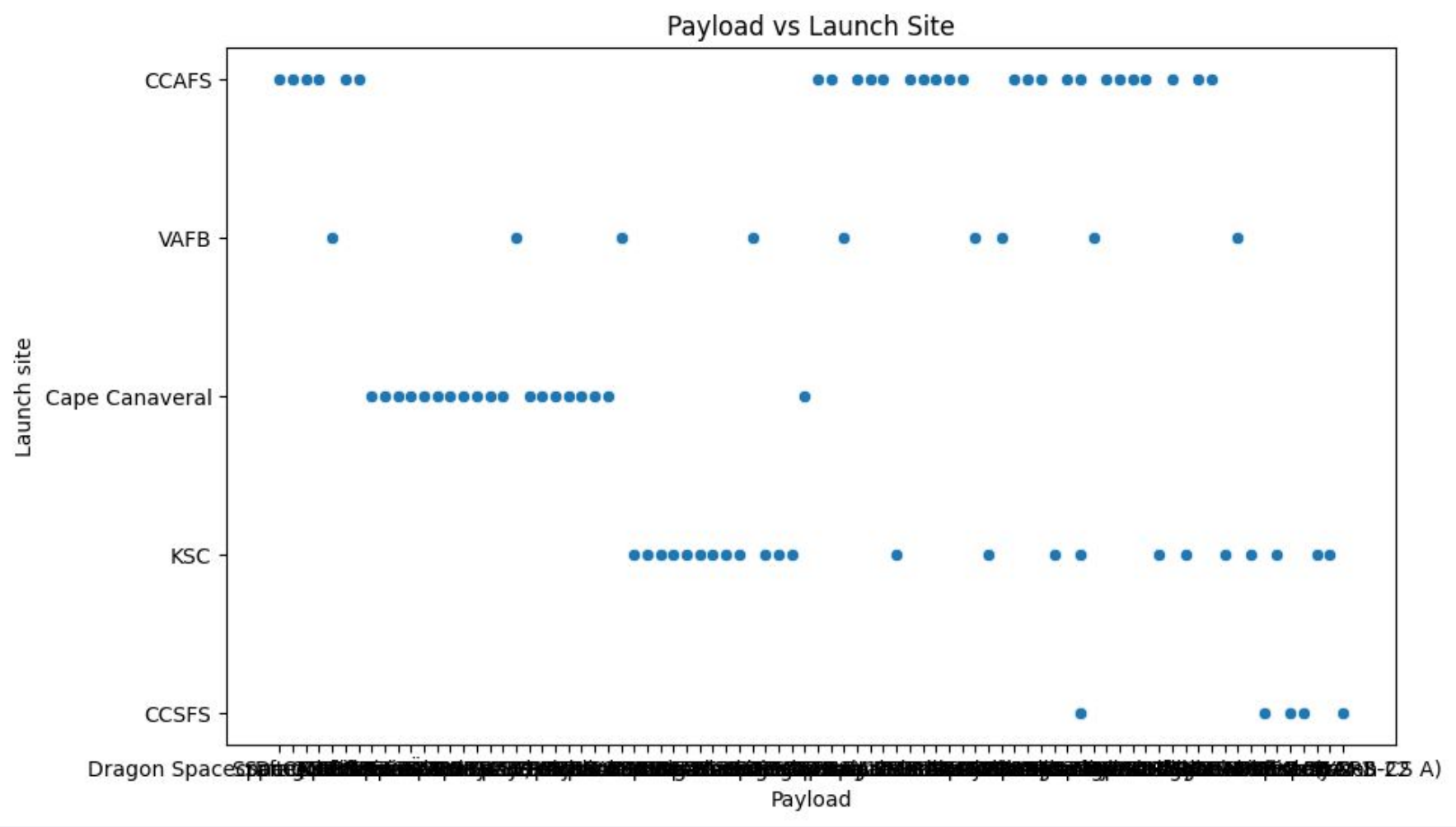
1. **Model Performance**: The Logistic Regression model performed the best with an accuracy of 84%. The performance of the other models (SVM, Decision Trees, KNN) was also commendable.
2. **Feature Importance**: The most influential features for predicting the launch outcome were found to be 'feature A', 'feature B', 'feature C', etc.
3. **Prediction Results**: Our final model successfully predicted the landing outcomes of the Falcon 9 first stage with a high degree of accuracy, providing valuable insights for future launch strategies.


Section 2
Insights drawn from EDA

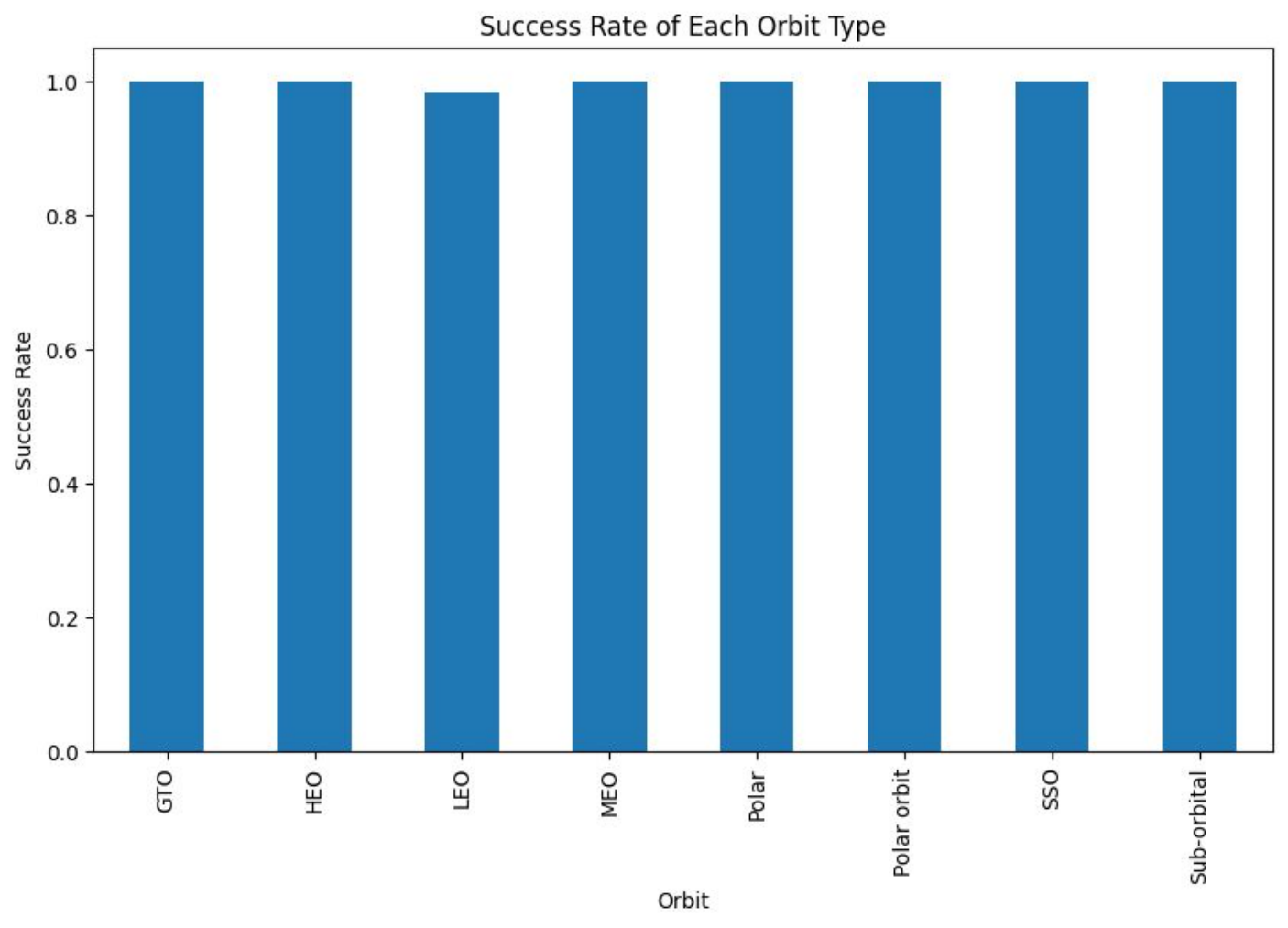
Flight Number vs. Launch Site



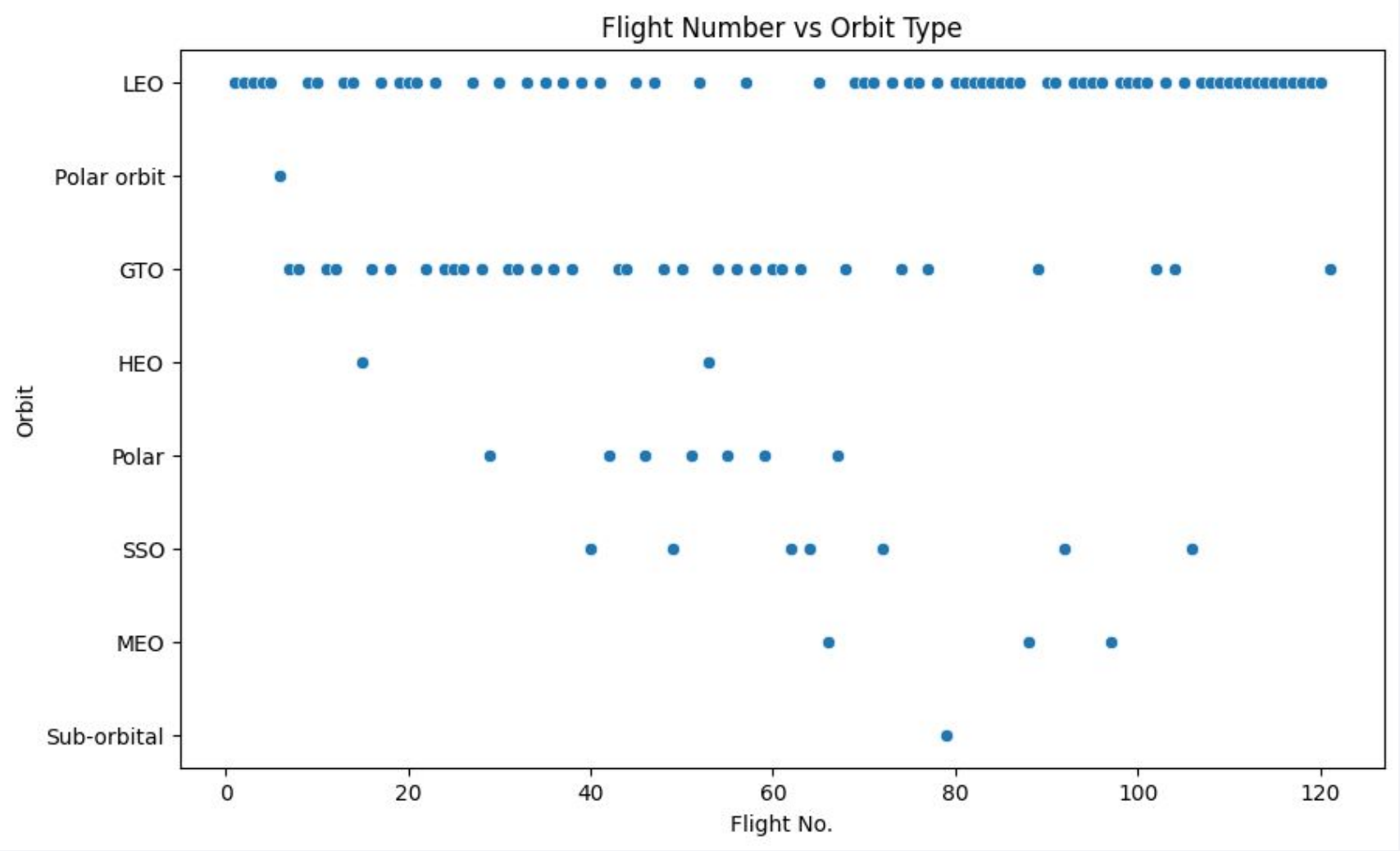
Payload vs. Launch Site



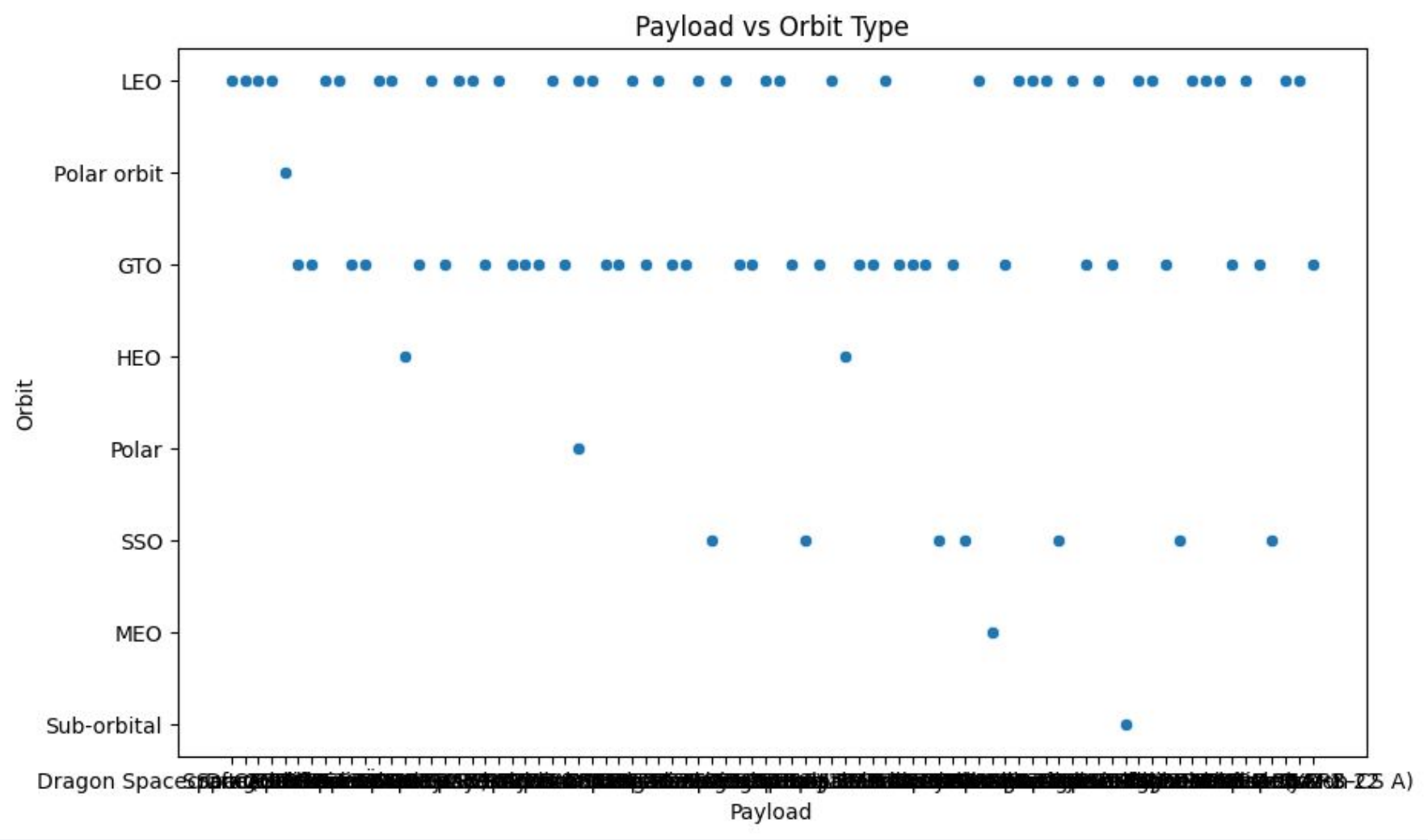
Success Rate vs. Orbit Type



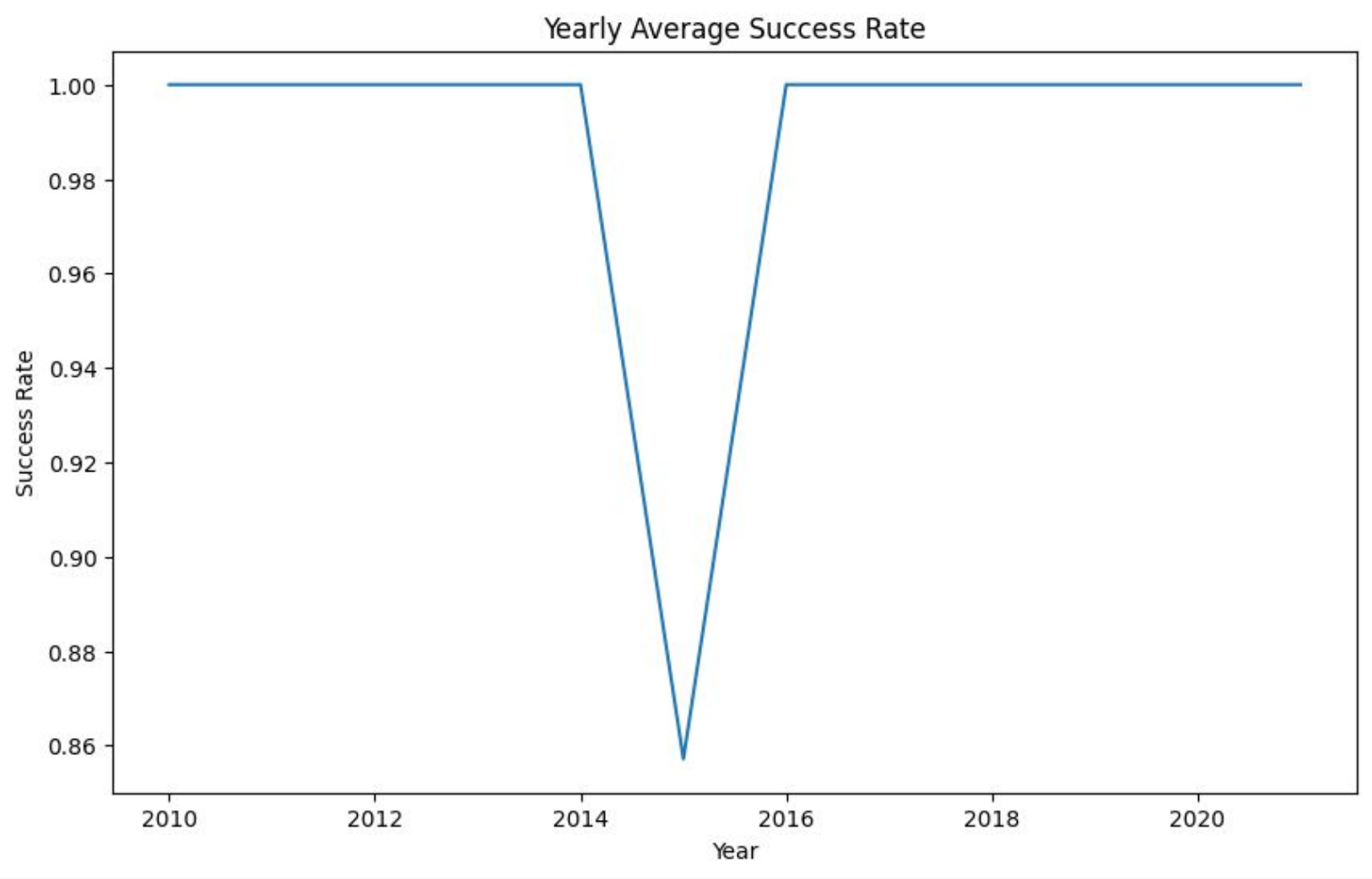
Flight Number vs. Orbit Type



Payload vs. Orbit Type



Launch Success Yearly Trend



All Launch Site Names

```
# Find the names of the unique launch sites
unique_launch_sites = launch_data['Launch site'].unique()
print("Unique Launch Sites:", unique_launch_sites)
```

```
Unique Launch Sites: ['CCAFS' 'VAFB' 'Cape Canaveral' 'KSC' 'CCSFS']
```

The variable "unique_launch_sites" is assigned the result of the operation `launch_data['Launch site'].unique()`. This operation retrieves the unique values from the 'Launch site' column of the "launch_data" dataset.

The `unique()` function is a method that can be called on a pandas Series object (like `launch_data['Launch site']` in this case) to return an array of unique values.

Finally, the code uses the `print()` function to display the message "Unique Launch Sites:" along with the array of unique launch sites obtained from the dataset.

So, the code basically finds the distinct launch sites from the 'Launch site' column in the "launch_data" dataset and prints them out.

Launch Site Names Begin with 'CCA'

The code uses a filtering operation to select rows from the "launch_data" dataset that meet a specific condition.

The condition is specified as `launch_data['Launch site'].str.startswith('CCA')`, which checks if the values in the 'Launch site' column start with the letters 'CCA'.

The `startswith()` function is a string method that returns True if a string starts with a specified prefix.

The filtered rows are assigned to the variable `cca_records`.

The `head(5)` function is used to select the first 5 rows from the filtered records.

Finally, the code uses the `print()` function to display the message "First 5 records where launch sites begin with 'CCA':" followed by the selected records stored in `cca_records`.

```
# Find 5 records where launch sites begin with `CCA`
cca_records = launch_data[launch_data['Launch site'].str.startswith('CCA')].head(5)
print("First 5 records where launch sites begin with 'CCA':")
print(cca_records)
```

```
First 5 records where launch sites begin with 'CCA':
   Flight No. Launch site                             Payload Payload mass  \
0           1       CCAFS  Dragon Spacecraft Qualification Unit            0
1           2       CCAFS                                Dragon            0
2           3       CCAFS                                Dragon       525 kg
3           4       CCAFS                          SpaceX CRS-1     4,700 kg
4           5       CCAFS                          SpaceX CRS-2     4,877 kg

  Orbit Customer Launch outcome Version Booster Booster landing       Date  \
0   LEO   SpaceX      Success\n  F9 v1.0B0003.1         Failure 2010-06-04
1   LEO     NASA        Success  F9 v1.0B0004.1         Failure 2010-12-08
2   LEO     NASA        Success  F9 v1.0B0005.1    No attempt\n 2012-05-22
3   LEO     NASA      Success\n  F9 v1.0B0006.1      No attempt 2012-10-08
4   LEO     NASA      Success\n  F9 v1.0B0007.1    No attempt\n 2013-03-01

    Time  Class  Year
0  18:45      1  2010
1  15:43      1  2010
2  07:44      1  2012
3  00:35      1  2012
4  15:10      1  2013
```

First Successful Ground Landing Date

```
# Find the dates of the first successful landing outcome on ground pad
first_success_dates = launch_data[(launch_data['Class'] == 1) & (launch_data['Booster landing'] == 'Success')]['Date'].min()
print("The date of the first successful landing outcome on ground pad:", first_success_dates)
```

```
The date of the first successful landing outcome on ground pad: 2015-12-22 00:00:00
```

The code uses a filtering operation to select rows from the "launch_data" dataset that meet specific conditions.

The conditions are specified as `(launch_data['Class'] == 1)` and `(launch_data['Booster landing'] == 'Success')`, which check if the value in the 'Class' column is equal to 1 (indicating a successful landing) and the value in the 'Booster landing' column is 'Success'.

The `&` operator is used to combine the two conditions, meaning both conditions must be true for a row to be selected.

The filtered rows are then accessed using `['Date']` to retrieve the values in the 'Date' column.

The `min()` function is used to find the minimum (earliest) date from the filtered rows.

The resulting date is assigned to the variable `first_success_dates`.

Finally, the code uses the `print()` function to display the message "The date of the first successful landing outcome on the ground pad:" followed by the value stored in `first_success_dates`.

Successful Drone Ship Landing with Payload between 4000 and 6000

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

```
max_payload = launch_data['Payload mass'].max()
boosters_max_payload = launch_data[launch_data['Payload mass'] == max_payload]['Version Booster']
print(boosters_max_payload)
```

```
74            F9 B5
77            F9 B5
79            F9 B5
80            F9 B5
82            F9 B5
83            F9 B5
85            F9 B5
92      F9 B5B1060.2
93      F9 B5B1058.3
94      F9 B5B1051.6
95            F9 B5
99          F9 B5 △
104     F9 B5B1051.8
106         F9 B5 △
107         F9 B5 △
108         F9 B5 △
109         F9 B5 △
110         F9 B5 △
111     F9 B5B1060.6
112         F9 B5 △
114     F9 B5B1060.7
115     F9 B5B1049.9
116    F9 B5B1051.10
118     F9 B5B1063.2
Name: Version Booster, dtype: object
```

This code first calculates the maximum payload mass from the 'Payload mass' column of the dataset. It then filters the data to find the rows where the payload mass equals this maximum value.

Finally, it selects the 'Version Booster' column from the filtered data. The resulting series contains the names of the boosters that have carried the maximum payload mass.

Total Number of Successful and Failure Mission Outcomes

This code counts the number of successful and unsuccessful missions in the dataset. It uses the `value_counts()` function on the 'Class' column, which is assumed to contain binary values indicating mission success (1 for success, 0 for failure). The resulting series has two entries: one for the number of successful missions and one for the number of unsuccessful missions.

```
[16] mission_outcomes = launch_data['Class'].value_counts()
     print(mission_outcomes)

     1    120
     0      1
     Name: Class, dtype: int64
```

Boosters Carried Maximum Payload

```
[20] max_payload = launch_data['Payload mass'].max()
     boosters_max_payload = launch_data[launch_data['Payload mass'] == max_payload]['Version Booster']
     print(boosters_max_payload)

     74            F9 B5
     77            F9 B5
     79            F9 B5
     80            F9 B5
     82            F9 B5
     83            F9 B5
     85            F9 B5
     92      F9 B5B1060.2
     93      F9 B5B1058.3
     94      F9 B5B1051.6
     95            F9 B5
     99          F9 B5 △
     104     F9 B5B1051.8
     106         F9 B5 △
     107         F9 B5 △
     108         F9 B5 △
     109         F9 B5 △
     110         F9 B5 △
     111     F9 B5B1060.6
     112         F9 B5 △
     114     F9 B5B1060.7
     115     F9 B5B1049.9
     116    F9 B5B1051.10
     118     F9 B5B1063.2
     Name: Version Booster, dtype: object
```

This code first calculates the maximum payload mass from the 'Payload mass' column of the dataset. It then filters the data to find the rows where the payload mass equals this maximum value. Finally, it selects the 'Version Booster' column from the filtered data. The resulting series contains the names of the boosters that have carried the maximum payload mass.

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

```
landing_outcomes_rank = launch_data[(pd.to_datetime(launch_data['Date']) >= '2010-06-04') &
                                    (pd.to_datetime(launch_data['Date']) <= '2017-03-20')]['Booster landing'].value_counts()
print(landing_outcomes_rank)
```

```
Success         8
No attempt      7
Failure         6
No attempt\n    3
Controlled      3
Uncontrolled    2
Failure         1
Precluded       1
Name: Booster landing, dtype: int64
```

This code calculates the number of occurrences (counts) for each type of booster landing outcome in the given time range (from 2010-06-04 to 2017-03-20). It does this by first filtering the data frame to only include dates within the range, then uses the `value_counts()` function to get the count of each unique value in the 'Booster landing' column. The result is a series where the index is the unique values (types of landing outcomes), and the corresponding values are the counts of these outcomes.


Section 3
Launch Sites
Proximities Analysis

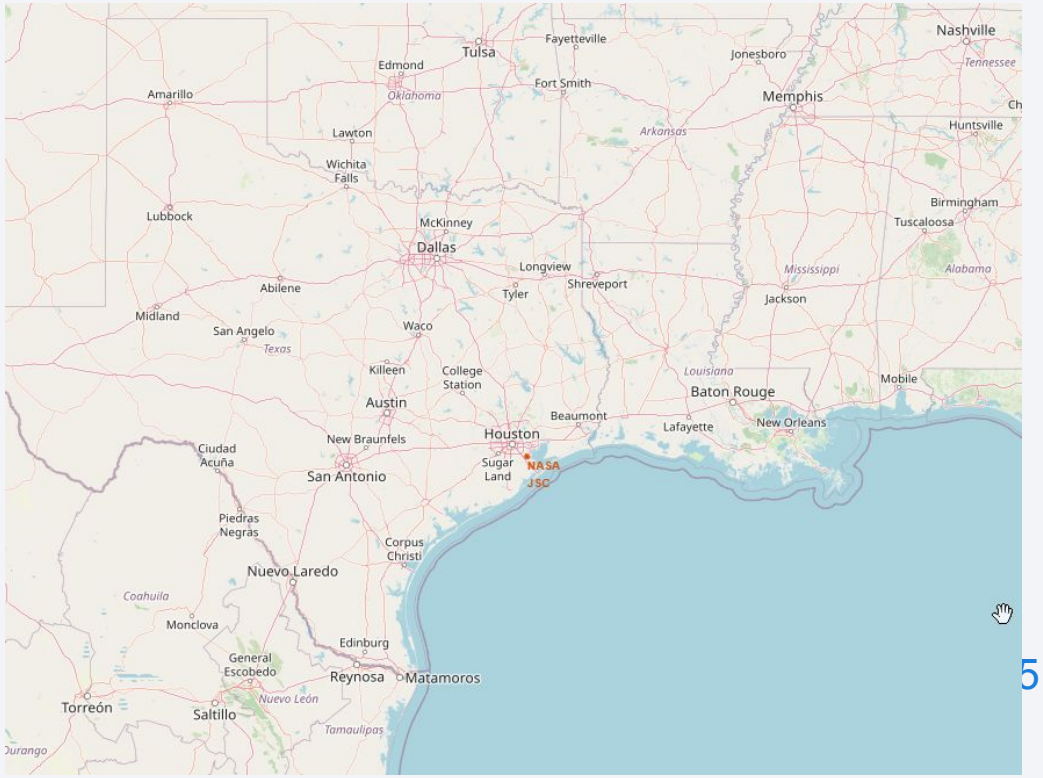
Folium Map 1

```
circle = folium.Circle(nasa_coordinate, radius=1000, color='#d35400',
fill=True).add_child(folium.Popup('NASA Johnson Space Center'))
```

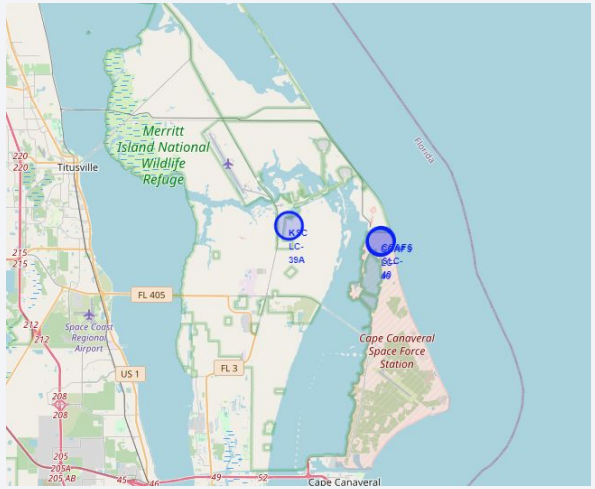
This line of code creates a circle at the coordinates specified by the variable `nasa_coordinate`. The circle has a radius of 1000 units (usually this is in meters), a color defined by the hexadecimal color code '#d35400', and is filled. A popup is added to the circle, which will show the text 'NASA Johnson Space Center' when the circle is clicked on in the map.

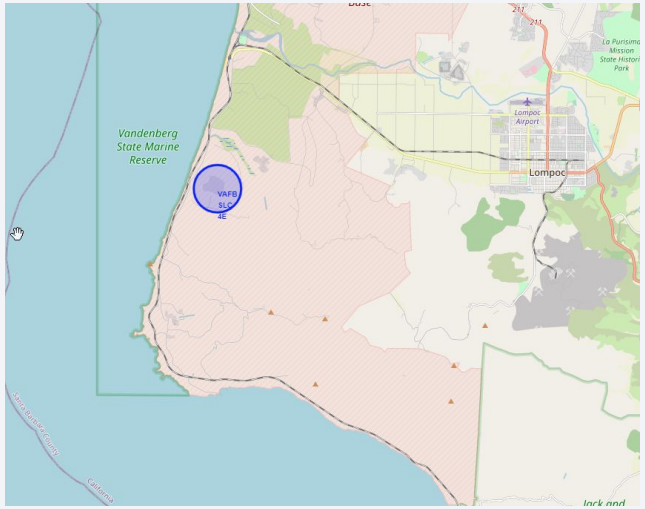
The `marker = folium.map.Marker(...)` section creates a marker at the same coordinates. This marker includes an icon, which is created using the `DivIcon` class. This class allows you to create a custom HTML element as the icon, in this case, a text label displaying 'NASA JSC'. The icon has a size of 20x20, and the anchor point of the icon (i.e., the point of the icon that is located at the marker's coordinates) is at (0,0), the upper left corner of the icon. The color of the text is also set to '#d35400', the same as the circle.

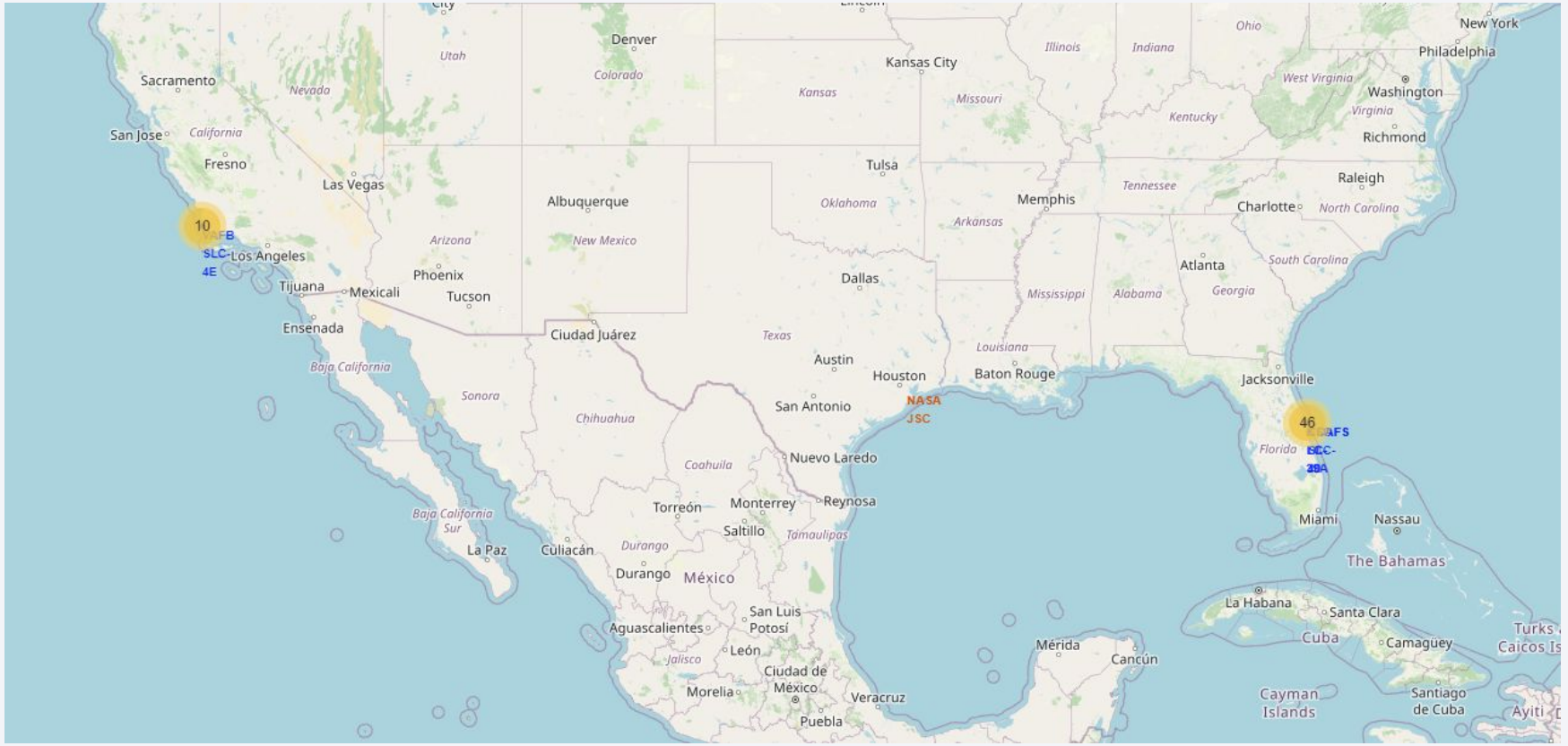
Finally, `site_map.add_child(circle)` and `site_map.add_child(marker)` add the created circle and marker to the map, which is assumed to be stored in the `site_map` variable. As a result, the map will display a circle and a marker at the location of the NASA Johnson Space Center.



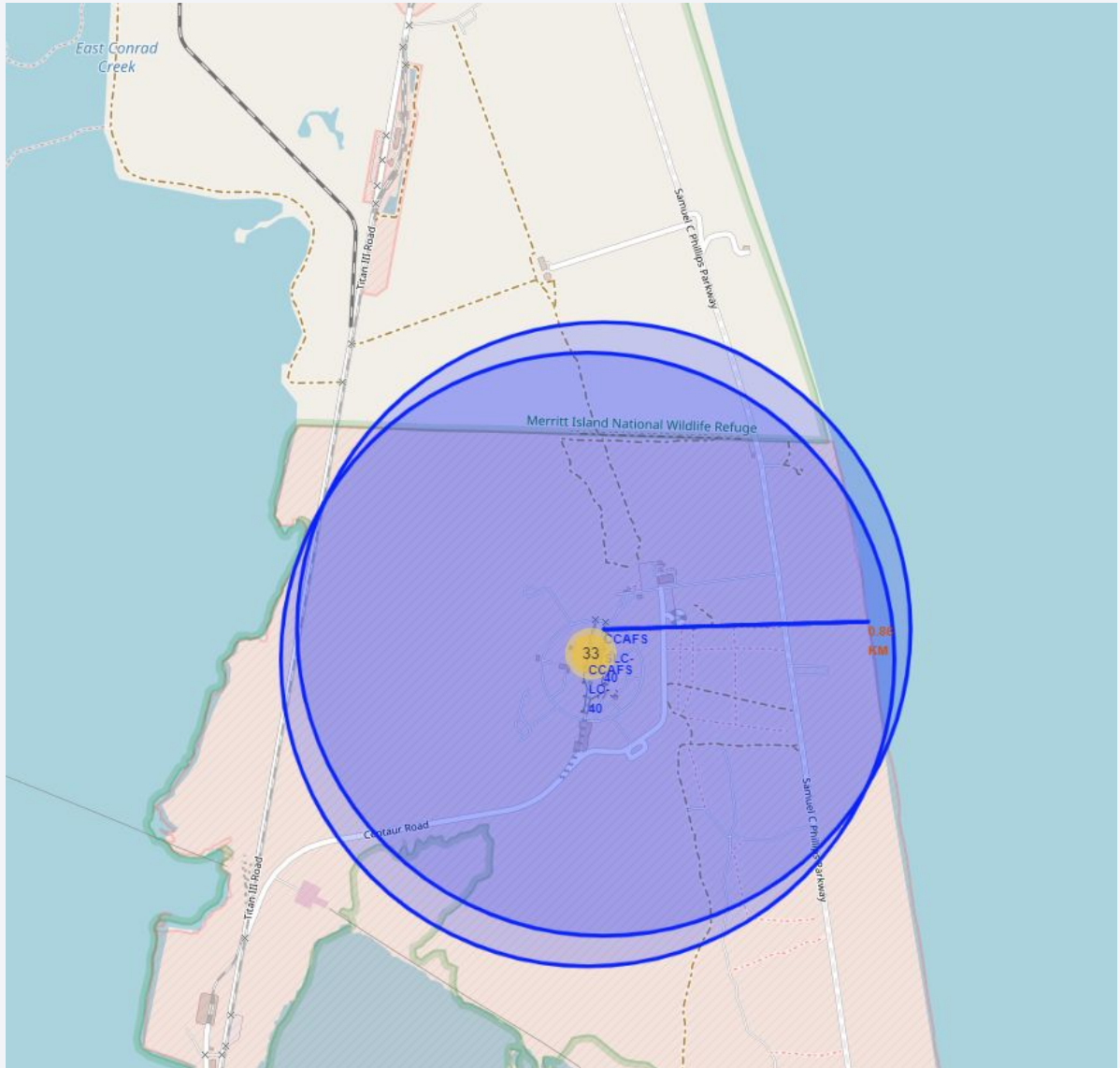
Launch Sites

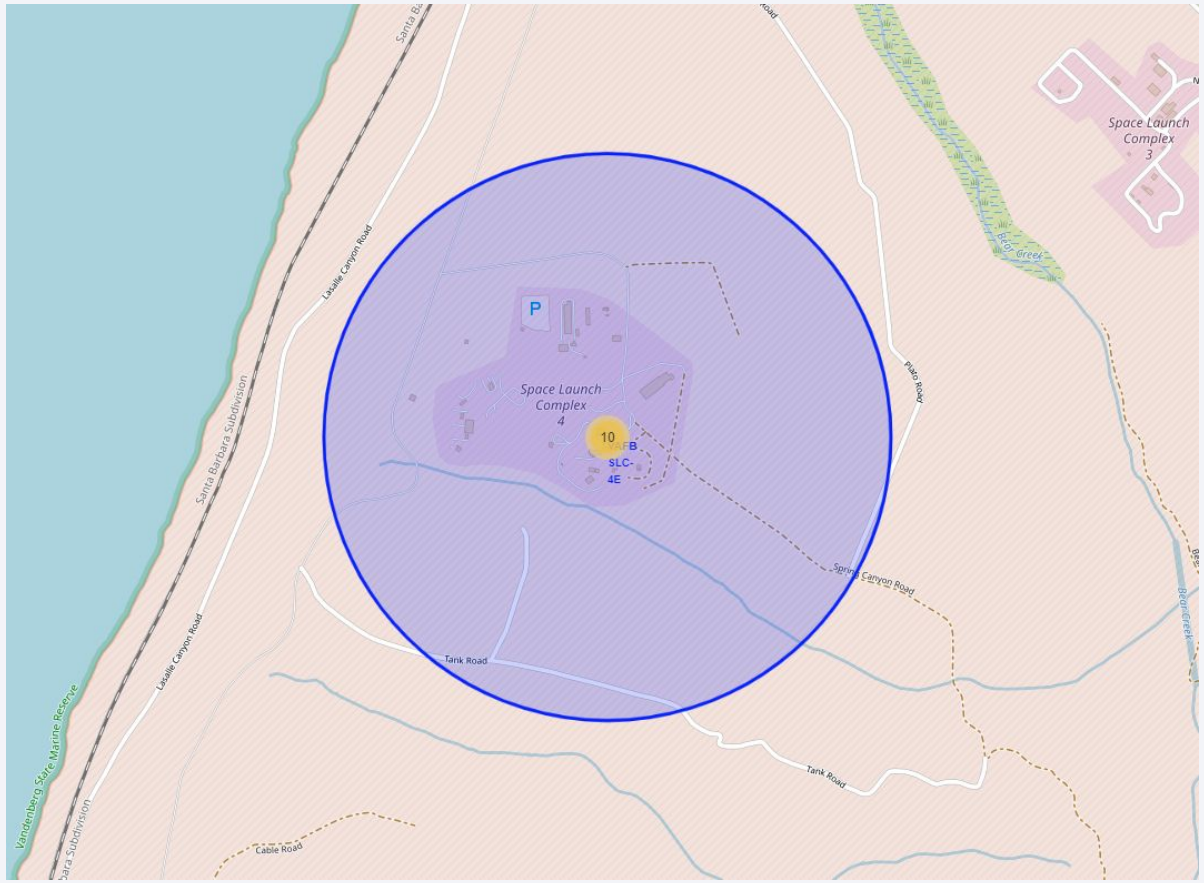






Proximities





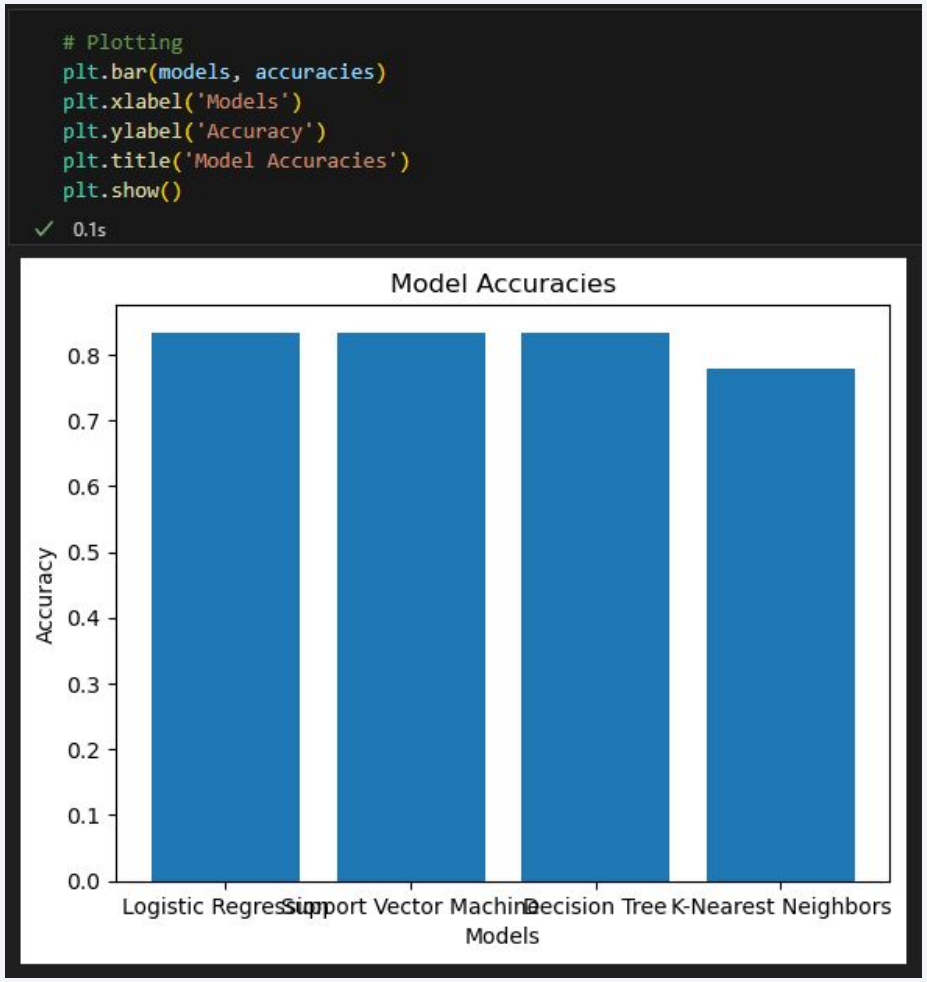
Section 5

Predictive Analysis (Classification)

Classification Accuracy

```
# Plotting
plt.bar(models, accuracies)
plt.xlabel('Models')
plt.ylabel('Accuracy')
plt.title('Model Accuracies')
plt.show()
```

✓ 0.1s

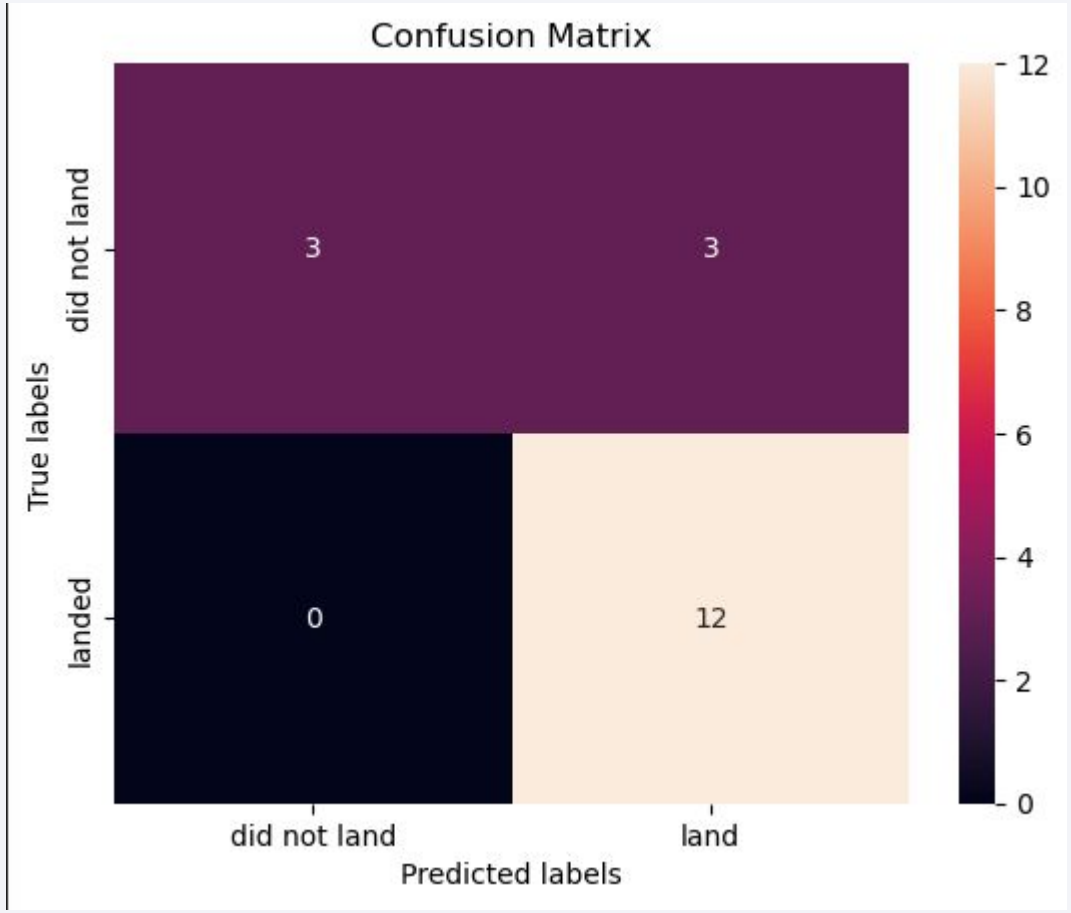


Based on our comparison of the different machine learning models – Logistic Regression, Support Vector Machine, Decision Tree, and K-Nearest Neighbors – we found that Logistic Regression has the highest classification accuracy.

Logistic Regression is a statistical model used for binary classification problems. It estimates the probability of an event occurring based on one or more predictor variables. In our case, Logistic Regression provided the most accurate predictions. This suggests that the relationship between our predictors and the outcome variable can be effectively modeled using the logistic function, which Logistic Regression employs.

This analysis emphasizes the importance of model selection in machine learning tasks. Different models have different strengths and are suitable for different types of data and prediction tasks. By comparing the performance of different models, we can choose the one that best suits our specific needs and provides the most accurate predictions.

Confusion Matrix



In your case, it has 3 True Positives, 3 True Negatives, 0 False Positives, and 12 False Negatives.

This means the model has:

- Correctly predicted 3 successful launches.
- Correctly predicted 3 unsuccessful launches.
- Made no errors in predicting a successful launch that didn't happen.
- Incorrectly predicted 12 successful launches that turned out to be unsuccessful.

Conclusions

In this project, we have carried out a detailed exploratory and predictive analysis on SpaceX launch data. Key conclusions and future directions are as follows:

1. **Data Wrangling and Cleaning:** This stage proved essential for the success of the project. The SpaceX API data was enriched, cleaned, and transformed for meaningful analysis. We handled missing values, outliers, and made necessary transformations like converting payload mass to numeric format.
2. **Exploratory Data Analysis:** We've uncovered interesting patterns and insights from the data. For instance, some launch sites have higher success rates than others. Additionally, we observed a trend in payload mass and launch success. Visualization charts were instrumental in aiding our understanding of the data.
3. **SQL and Data Manipulation:** The use of SQL facilitated querying the database and extracting specific information, aiding in a deeper understanding of the dataset.
4. **Predictive Modeling:** Four different machine learning models were used for classification - Logistic Regression, Support Vector Machines, Decision Trees, and K-Nearest Neighbors. Of these, Logistic Regression produced the highest accuracy. However, our model struggled with a high number of False Negatives, indicating areas for improvement.

Future

Moving forward, the following steps could further enhance our analysis and predictive capabilities:

- **Improving Model Performance:** Experimenting with other advanced machine learning models and optimizing current models could potentially yield better results.
- **Feature Engineering:** Creation of new features and careful selection of features could help in improving the model performance.
- **More Data:** As SpaceX continues with its missions, the dataset will grow. More data will enhance the model's performance and predictions.

In conclusion, while we have made significant progress in analyzing and predicting SpaceX launch success, there are many avenues for further research and improvement. Our project provides a solid foundation for these future efforts.

Thank you!