IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Vinay Ghosh
6 February 2025

Outline

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

Executive Summary

- Summary of methodologies

1. **Data Acquisition:** Data was collected using the SpaceX API, web scraping from Wikipedia, and stored in Python lists, dictionaries, Pandas DataFrames, CSV files, and an SQLite database.
2. **Data Preparation:** Data cleaning involved handling missing values, filtering, type conversions, and feature engineering. Data from different sources was integrated into a single DataFrame.
3. **Exploratory Analysis:** Descriptive statistics and visualizations (scatter plots, bar charts, line charts) were used to understand data patterns and relationships between variables like launch site, payload mass, orbit type, and success rate.
4. **Interactive Visualization:** Folium maps were used to visualize launch sites and outcomes geographically. A Dash dashboard was built for dynamic data exploration.
5. **Predictive Modeling:** Machine learning models (Logistic Regression, SVM, Decision Tree ,KNN) were trained on preprocessed data (scaled and one-hot encoded) to predict
6. launch outcomes. Model performance was evaluated using appropriate metrics.

- Summary of all results
  - Project revealed insights into factors impacting launch success and showed the potential of predictive modeling.
  - Further model evaluation is needed for comprehensive results and predictive capabilities.

Introduction

- Project background and context

The increasing demand for space exploration and commercial satellite deployment necessitates a deeper understanding of launch dynamics. By examining historical data, we aim to identify key trends and patterns associated with successful launches. This project utilizes data from various sources, including the SpaceX API and publicly available records, to create a robust dataset for analysis.

- The primary objectives of this project are:
    1. To investigate the relationships between launch site, payload mass, orbit type, and launch outcomes.
    2. To visualize these relationships and trends using interactive maps and dashboards.
    3. To develop machine learning models capable of predicting launch outcomes based on historical data

- Problems you want to find answers

    Data inconsistencies, missing values, and web scraping challenges hindered data preparation. Model selection, evaluation, and interpretation, along with computational limitations, posed difficulties during analysis and prediction.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - **Data was collected using the SpaceX API and web scraping from Wikipedia, then stored and processed using Python libraries.**
- Perform data wrangling
  - **Data was cleaned, transformed, and engineered using Pandas and scikit-learn for analysis & 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
  - Build by selecting features and training an algorithm; tune using hyperparameter optimization and cross-validation; evaluate with metrics like accuracy and a confusion matrix.

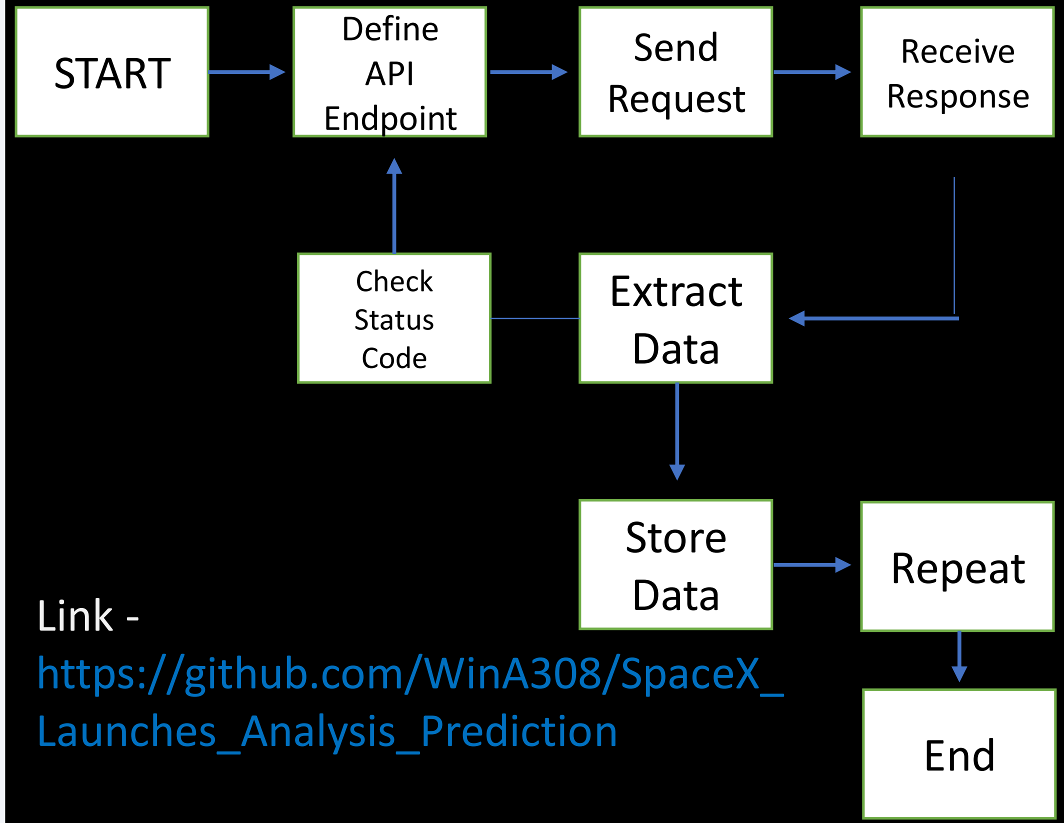
Data Collection

- Data collection involved leveraging the SpaceX API and web scraping techniques to obtain a comprehensive view of SpaceX launches. By integrating and augmenting data from multiple sources, a robust dataset was created, enabling subsequent analysis and model development.
- **SpaceX API Data Collection Flowchart**
    1. **Start**
    2. **Define API Endpoint**
    3. **Send Request.**
    4. **Receive Response**
    5. **Check Status Code**
    6. **Extract Data**
    7. **Store Data : Repeat (Optional)**
    8. **End**

Data Collection – SpaceX API

SpaceX API Data Collection Flowchart

1. **Start:** Begin data collection process.
2. **Define API Endpoint:** Specify the URL for the desired SpaceX API endpoint (e.g., "[redacted link]).
3. **Send Request:** Use the requests library in Python to send an HTTP GET request to the API endpoint.
4. **Receive Response:** Get the response from the API, containing launch data in JSON format.
5. **Check Status Code:** Verify if the request was successful (status code 200).
   - If successful, proceed to the next step.
   - If unsuccessful (e.g., status code 404), handle the error and potentially retry the request or choose a different endpoint.
6. **Extract Data:**
   - Parse the JSON response using the json() method of the response object.
   - Access the desired data fields (e.g., rocket, payload, launchpad, cores, flight number, date_utc).
7. **Store Data:** Store the extracted data in a suitable data structure, such as:
   - Python lists or dictionaries.
   - A Pandas DataFrame for easier manipulation and analysis.
8. **Repeat (Optional):** If you need to collect data from multiple API endpoints or pages, repeat steps 2-7 with the appropriate modifications.
9. **End:** Data collection completed.

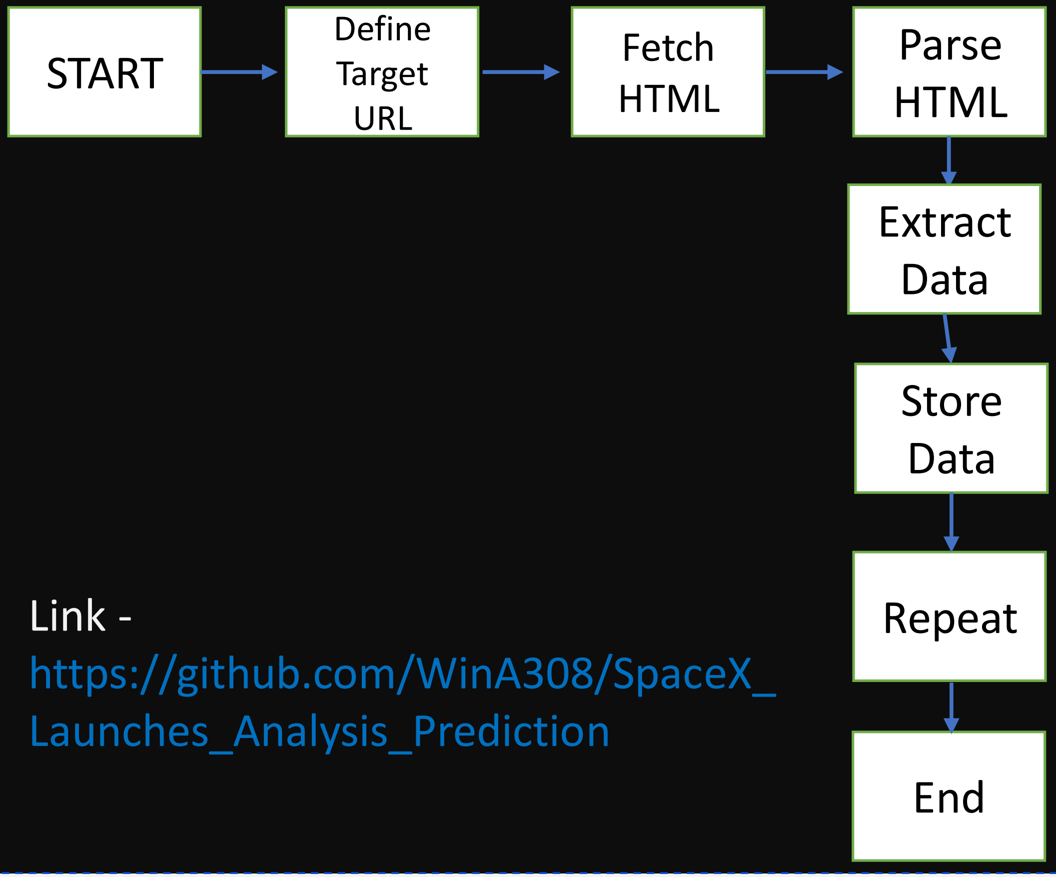


Link - https://github.com/WinA308/SpaceX_Launches_Analysis_Prediction

Data Collection - Scraping

Web Scraping Data Collection Flowchart

1. **Start:** Begin the web scraping process.
2. **Define Target URL:** Specify the URL of the Wikipedia page containing the SpaceX launch data (e.g., "[redacted link]).
3. **Fetch HTML Content:** Use the requests library in Python to send an HTTP GET request to the URL and retrieve the HTML content of the page.
4. **Parse HTML:**
   - Create a BeautifulSoup object using the HTML content to parse and navigate the document's structure.
   - Identify the specific HTML elements (e.g., tables, rows, cells) containing the desired data.
5. **Extract Data:**
   - Iterate through the relevant HTML elements using BeautifulSoup methods like find_all() or select().
   - Extract the text content or attribute values from the selected elements.
   - Apply any necessary data cleaning or formatting (e.g., removing whitespace, converting data types).
6. **Store Data:** Store the extracted data in a suitable data structure, such as:
   - Python lists or dictionaries.
   - A Pandas DataFrame for easier manipulation and analysis.
7. **Repeat (Optional):** If you need to extract data from multiple tables or sections of the webpage, repeat steps 4-6 with the appropriate modifications.
8. **End:** Web scraping completed.



Link - https://github.com/WinA308/SpaceX_Launches_Analysis_Prediction

Data Wrangling

Data Wrangling Flowchart :-

1.Start: Begin data wrangling with the raw data collected from the SpaceX API and web scraping.

2.Data Integration:

- If data was collected from multiple sources, combine them into a unified Pandas DataFrame.
- Ensure consistent column names and data types across all sources.

3.Data Cleaning:

- **Handle Missing Values:**
    - Identify columns with missing values (e.g., using df.isnull().sum()).
    - Decide on an imputation strategy:
        - Replace with mean, median, or mode for numerical data.
        - Replace with a constant value (e.g., "Unknown") for categorical data.
        - Drop rows or columns with excessive missing values.
- **Remove Duplicates:**
    - Identify and remove any duplicate rows (e.g., using df.drop_duplicates()).
- **Data Type Conversion:**
    - Convert columns to appropriate data types (e.g., using pd.to_datetime() for dates).

4.Data Transformation:

- **Feature Engineering:**
    - Create new features from existing ones (e.g., calculate success rate, extract year from date).
    - Apply transformations to improve model performance (e.g., scaling, encoding).
- **Data Filtering:**
    - Remove irrelevant or unnecessary data (e.g., filter by specific launch sites or date ranges).

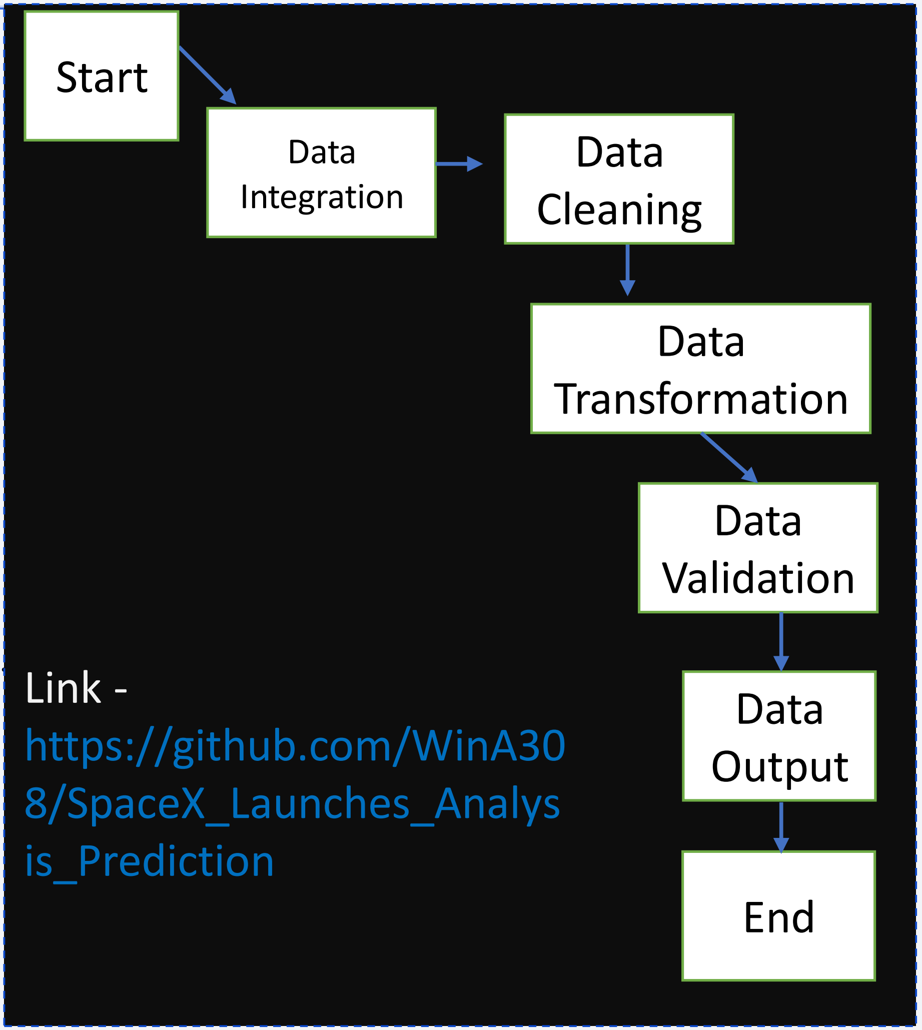
5.Data Validation:

- Check for data consistency and accuracy.
- Identify and correct any remaining errors or inconsistencies.

6.Data Output:

- Save the cleaned and transformed data to a new file (e.g., as a CSV file).

7.End: Data wrangling completed, ready for analysis and modeling.



EDA with Data Visualization

1. **Scatter Plots:**
    1. **Purpose:** To visualize the relationship between two numerical variables.
    2. **Examples:**
        1. Payload Mass vs. Flight Number: To see if there's a trend in payload mass over time or any correlation with launch success.
        2. Payload Mass vs. Launch Site: To explore if launch site has an influence on the types of payloads launched.
        3. Flight Number vs. Orbit: To analyze the relationship between flight number and the orbit type.
        4. Payload Mass vs. Orbit: To investigate the relationship between payload mass and the orbit type.
    3. **Why Scatter Plots?** They effectively show correlations, clusters, and outliers in data, which can help identify potential relationships between variables.

2. **Bar Charts:**
    1. **Purpose:** To compare the frequency or proportion of different categories.
    2. **Examples:**
        1. Success Rate by Orbit Type: To visualize the success rate of launches to different orbits.
    3. **Why Bar Charts?** They are suitable for categorical data and provide a clear visual comparison of the frequencies or proportions across different groups.

EDA with Data Visualization

1. Line Charts:

1. **Purpose:** To show trends or patterns over time.
2. **Examples:**
    1. Average Launch Success Trend Over the Years: To visualize the change in average success rate over the years.
3. **Why Line Charts?** They are ideal for displaying temporal data and visualizing trends, allowing you to see how a variable changes over a continuous period.

2. Pie Charts:

1. **Purpose:** To show the proportion of different categories in a whole.
2. **Examples:**
    1. Success Rate by Launch Site: To show the proportion of successful launches for each launch site.
    2. Success/Failure Distribution at a Specific Launch Site: To visualize the breakdown of successful and failed launches at a particular site.
3. **Why Pie Charts?** They are useful for displaying the relative size of different categories within a dataset, providing a quick visual representation of proportions.

Link - https://github.com/WinA308/SpaceX_Launches_Analysis_Prediction

EDA with SQL

SQL Command Summary

1.Data Loading:

- CREATE TABLE SPACEXTABLE AS SELECT * FROM SPACEXTBL WHERE Date IS NOT NULL: This command created a new table called SPACEXTABLE by selecting all columns and rows from the existing SPACEXTBL table where the Date column is not null. This effectively loaded your launch data into a new table for analysis.

2.Data Exploration:

- SELECT DISTINCT Launch_Site FROM SPACEXTABLE: This command retrieved the unique launch sites from the SPACEXTABLE table.
- SELECT * FROM SPACEXTABLE WHERE Launch_Site LIKE 'CCA%' LIMIT 5: This command selected all columns and rows from the SPACEXTABLE table where the Launch_Site column starts with 'CCA', limiting the results to the first 5 rows.
- SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Customer LIKE '%NASA%' AND Payload LIKE '%CRS%': This command calculated the total payload mass for launches where the customer was NASA and the payload was related to CRS (Commercial Resupply Services).
- SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Booster_Version = 'F9 v1.1': This command calculated the average payload mass for launches using the Falcon 9 v1.1 booster version.
- SELECT MIN(Date) FROM SPACEXTABLE WHERE Landing_Outcome LIKE '%Success (ground pad)%': This command retrieved the earliest launch date where the landing outcome was a success on the ground pad.
- SELECT DISTINCT Booster_Version FROM SPACEXTABLE WHERE Landing_Outcome LIKE '%Success (drone ship)%' AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000: This command selected the distinct booster versions used for launches that successfully landed on a drone ship with a payload mass between 4000 and 6000 kg.
- SELECT SUM(CASE WHEN Landing_Outcome LIKE '%Success%' THEN 1 ELSE 0 END) AS SuccessCount, SUM(CASE WHEN Landing_Outcome LIKE '%Failure%' THEN 1 ELSE 0 END) AS FailureCount FROM SPACEXTABLE: This command calculated the total number of successful and failed landings.
- SELECT Booster_Version FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE): This command retrieved the booster version used for the launch with the maximum payload mass.
- SELECT strftime('%Y-%m', Date) AS Month, Booster_Version, Launch_Site FROM SPACEXTABLE WHERE Landing_Outcome LIKE '%Failure (drone ship)%' AND strftime('%Y', Date) = '2015': This command selected the month, booster version, and launch site for launches that failed to land on a drone ship in 2015.

Link - https://github.com/WinA308/SpaceX_Launches_Analysis_Prediction

Build an Interactive Map with Folium

Rationale for Adding Map Objects

Markers:

- **Launch Site Visualization:** Markers were essential to pinpoint the locations of different SpaceX launch sites on the map. By placing markers at the coordinates of each launch site, you provided a visual representation of their geographical distribution.
- **Launch Outcome Representation:** Color-coded markers (green for success, red for failure) allowed you to quickly assess the outcomes of launches at specific locations. This visual cue helped identify patterns or potential correlations between launch site and success rate.

Circles:

- **Launch Site Emphasis:** Circles drawn around launch sites served to visually highlight these areas on the map. The orange color and fixed radius made them easily distinguishable from other map elements, drawing attention to the launch site locations.
- **Contextual Information:** The popups associated with the circles provided additional information about the launch site when clicked, enhancing the user's understanding of the map.

Lines (PolyLines):

- **Distance Visualization:** Lines were primarily used to represent distances between launch sites and other points of interest, such as coastlines or cities. By drawing lines between these locations, you could visually assess their proximity and potential influences on launch operations.
- **Geodesic Accuracy:** Using geodesic lines ensured that the lines followed the Earth's curvature, providing a more accurate representation of distances, especially for longer distances.

Marker Cluster:

- **Map Readability:** When dealing with a large number of launch locations, individual markers can clutter the map, making it difficult to interpret. Marker clusters addressed this issue by grouping nearby markers into a single icon, improving map readability and overall user experience.
- **Interactive Exploration:** The ability of marker clusters to expand and reveal individual markers when zoomed in allowed for interactive exploration of the data. Users could easily navigate between an overview of all launch locations and a detailed view of specific areas.

.

- Link - https://github.com/WinA308/SpaceX_Launches_Analysis_Prediction

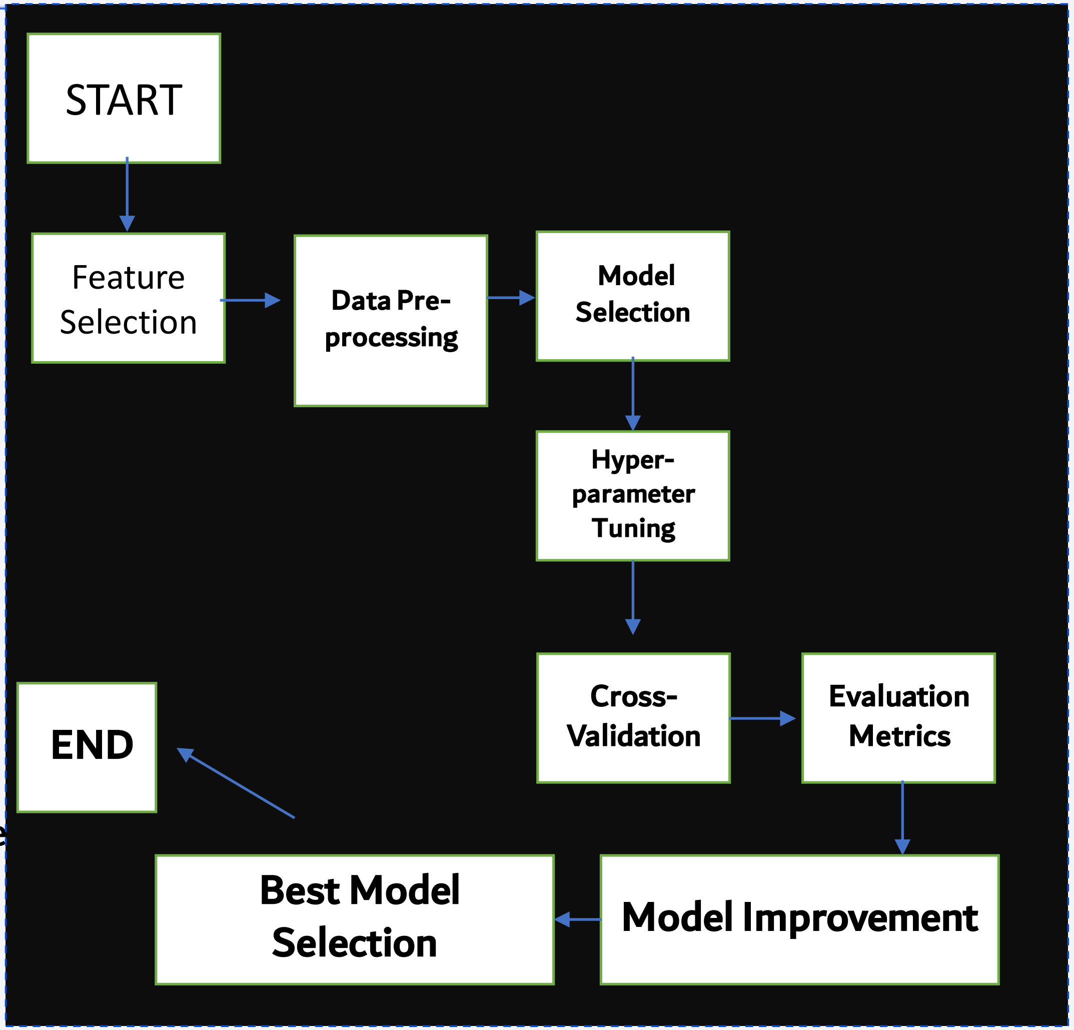
Build a Dashboard with Plotly Dash

- **Pie Chart:**
  1. **Purpose:** To show the proportion of successful and failed launches for a selected launch site or across all sites.
  2. **Interaction:** The pie chart dynamically updates based on the launch site selected in the dropdown menu.
  3. **Rationale:** Provides a quick visual overview of launch success rates, allowing users to compare outcomes across different sites or see the overall success/failure distribution.
- **Scatter Plot:**
  1. **Purpose:** To visualize the relationship between payload mass and launch outcome, colored by booster version.
  2. **Interactions:**
     1. Users can filter data by launch site using the dropdown menu.
     2. Users can adjust the range of payload mass using a slider.
  3. **Rationale:** Allows users to explore the correlation between payload mass and launch success, consider the impact of booster version, and focus on specific launch sites or payload ranges.
- **Dropdown Menu:**
  1. **Purpose:** To select a specific launch site for filtering data in both the pie chart and scatter plot.
  2. **Options:** Includes "All Sites" and individual launch site names.
  3. **Rationale:** Enables users to focus their analysis on particular launch sites or view overall trends across all sites.
- **Range Slider:**
  1. **Purpose:** To filter data by payload mass in the scatter plot.
  2. **Range:** Covers the minimum and maximum payload mass values in the dataset.
  3. **Rationale:** Allows users to investigate launch outcomes for specific payload ranges, identifying potential relationships between payload mass and success.

Link - https://github.com/WinA308/SpaceX_Launches_Analysis_Prediction

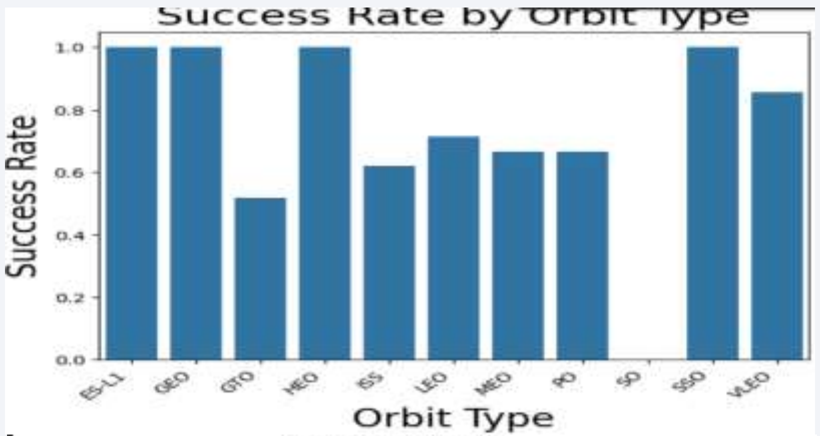
Predictive Analysis (Classification)

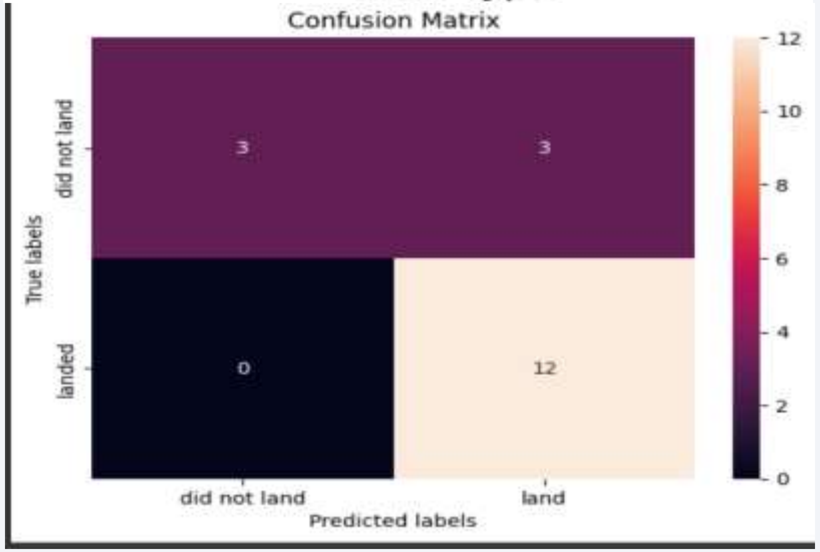
- **Feature Selection:** Identifying relevant features (FlightNumber, PayloadMass, Orbit, LaunchSite, etc.) for model training.
- **Data Preprocessing:** Transforming categorical features using one-hot encoding and scaling numerical features using StandardScaler.
- **Model Selection:** Evaluating Logistic Regression, Support Vector Machine, Decision Tree, and K-Nearest Neighbors classifiers.
- **Hyperparameter Tuning:** Using GridSearchCV to find optimal model parameters.
- **Cross-Validation:** Employing train-test split (80-20) and k-fold cross-validation to assess model performance on unseen data.
- **Evaluation Metrics:** Using accuracy, precision, recall, F1-score, and confusion matrix to compare model performance.
- **Model Improvement:** Iteratively adjusting model parameters, features, and algorithms to enhance performance.
- **Best Model Selection:** Identifying the model with the highest performance based on evaluation metrics and suitability for the task.



Results

- **EDA Results**
  1. **Launch Success Rate Over Time:** Upward trend, improved technology & operations.
  2. **Payload Mass vs. Outcome:** Possible correlation with success.
  3. **Success Rate by Launch Site:** Performance varies by site.
  4. **Success Rate by Orbit:** Mission complexity matters.
  5. **Launch Site Locations:** Geographical distribution & context.
- **Predictive Model**
  1. **Confusion Matrix:** Visualizes model's prediction accuracy.
  2. **Feature Importance:** Identifies key factors influencing launch success.
  3. **Model Performance Metrics:** Accuracy, precision, recall, F1-score reported.
  4. **ROC Curve:** Illustrates model's ability to discriminate between classes.
  5. **Prediction Results on Test Data:** Shows model performance on unseen data.





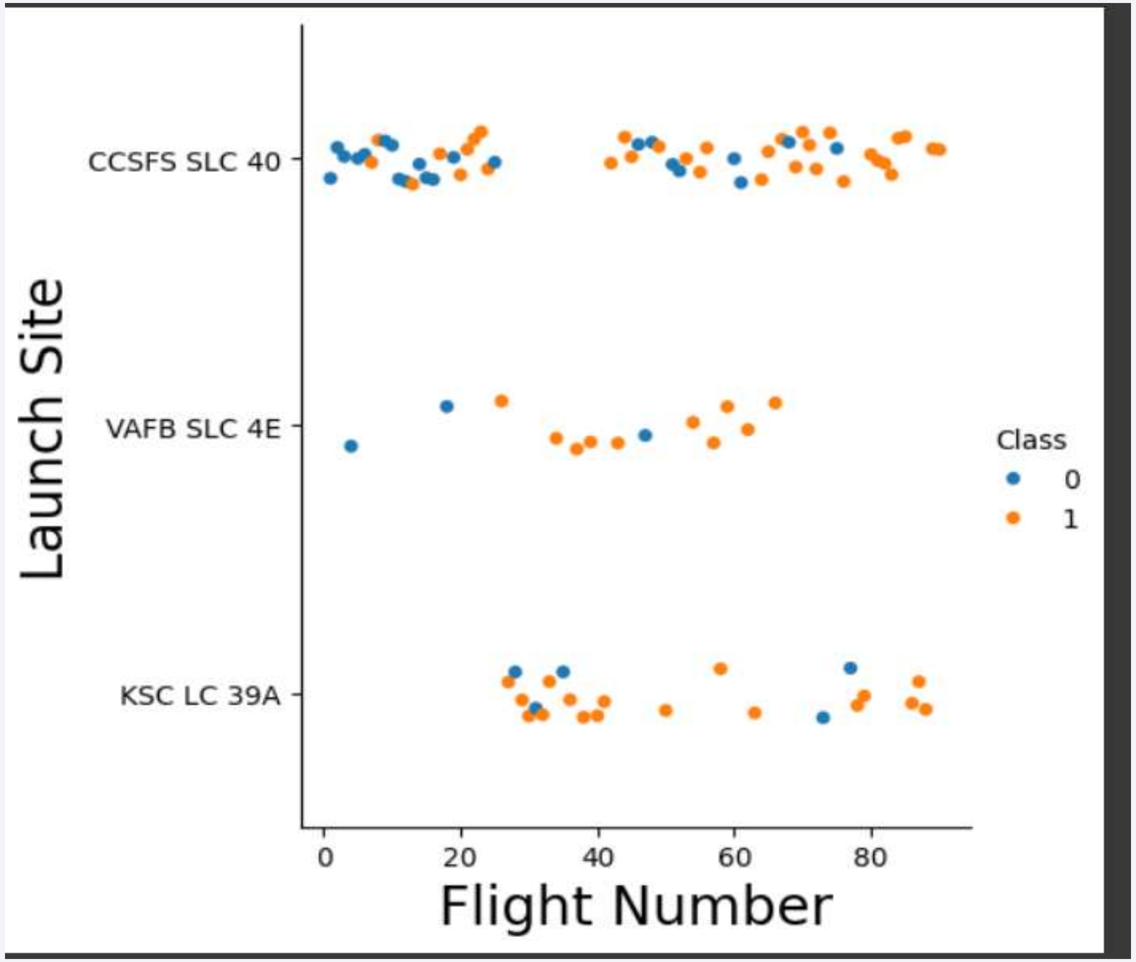
Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

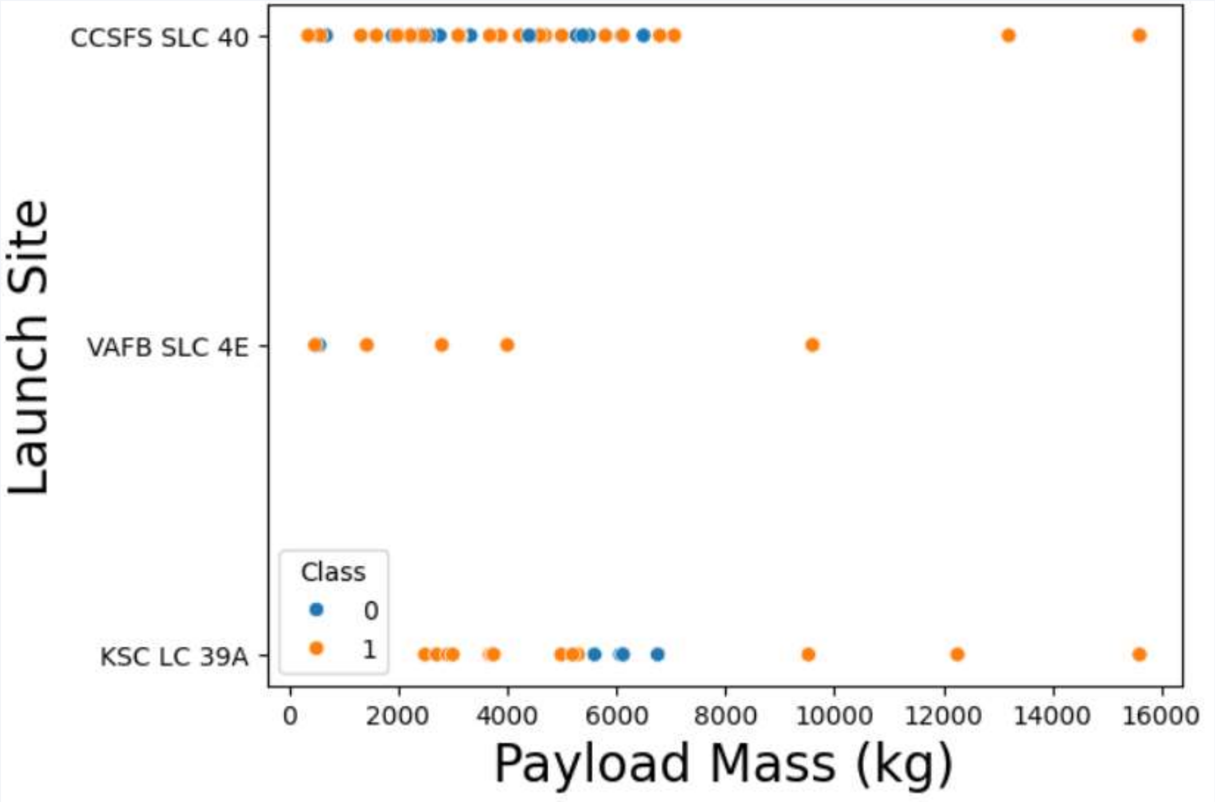
- Show a scatter plot of Flight Number vs. Launch Site

In essence, visualizes how launch outcomes (success or failure) are distributed across different launch sites and flight numbers.



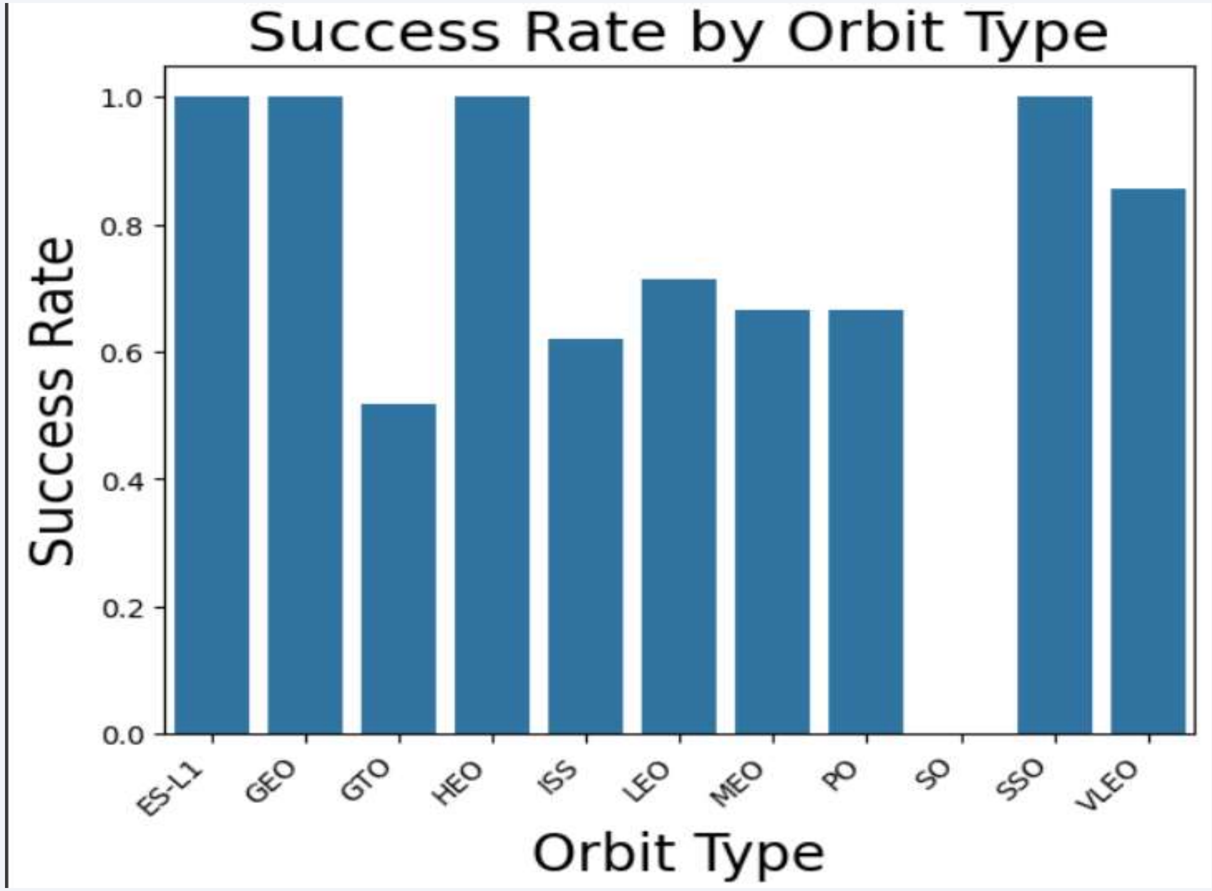
Payload vs. Launch Site

- Show a scatter plot of Payload vs. Launch Site
- **In essence, a scatter plot that visualizes the relationship between "Payload Mass" and "Launch Site" for SpaceX launches. Different launch outcomes (represented by the "Class" column) are distinguished by color, allowing for easier interpretation of the data.**



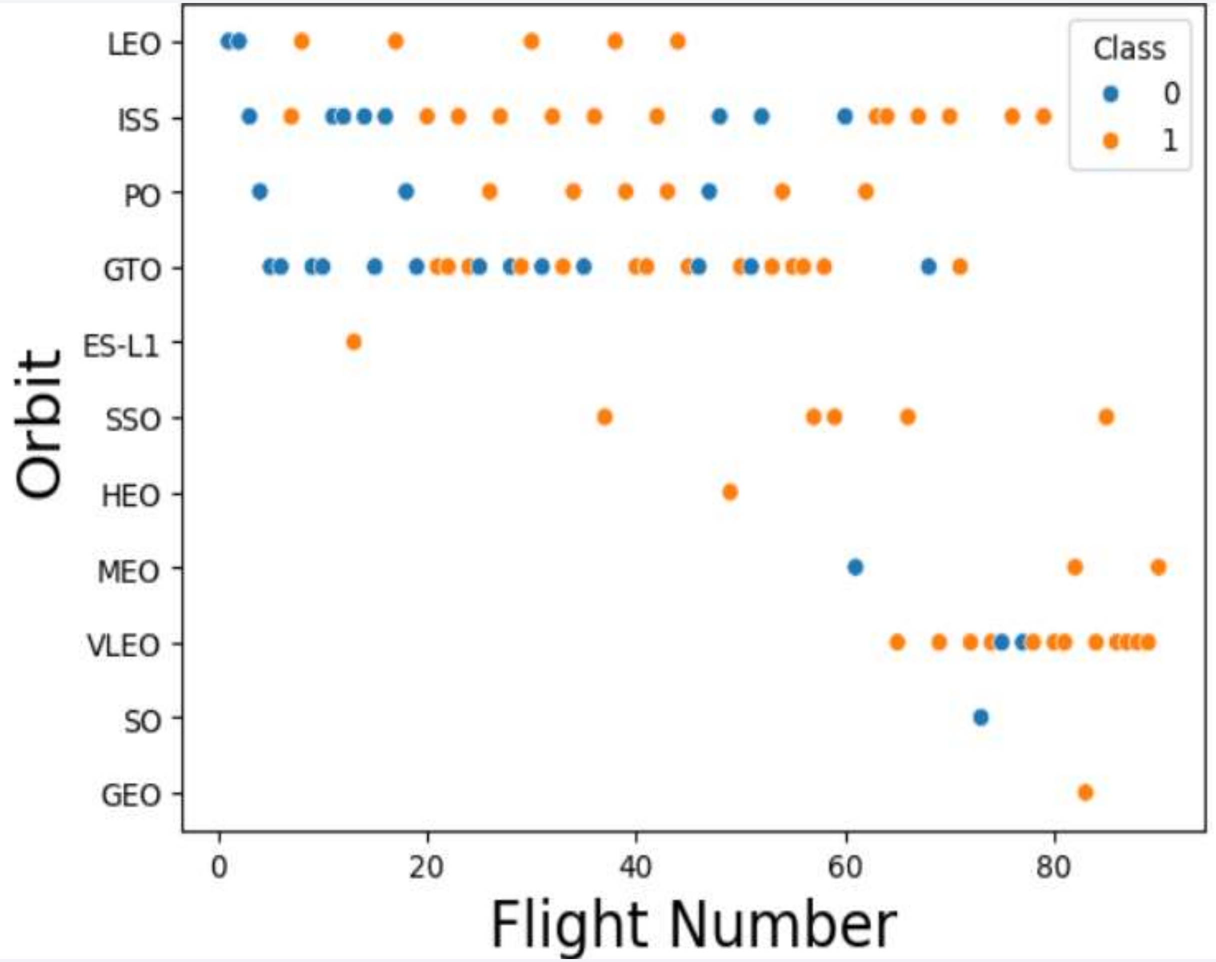
Success Rate vs. Orbit Type

- Show a bar chart for the success rate of each orbit type
- **DataFrame of SpaceX launch data, groups it by orbit type, calculates the success rate for each orbit, and then creates a bar chart to visualize these success rates. The chart is further enhanced with labels, a title, and rotated x-axis labels for better presentation.**



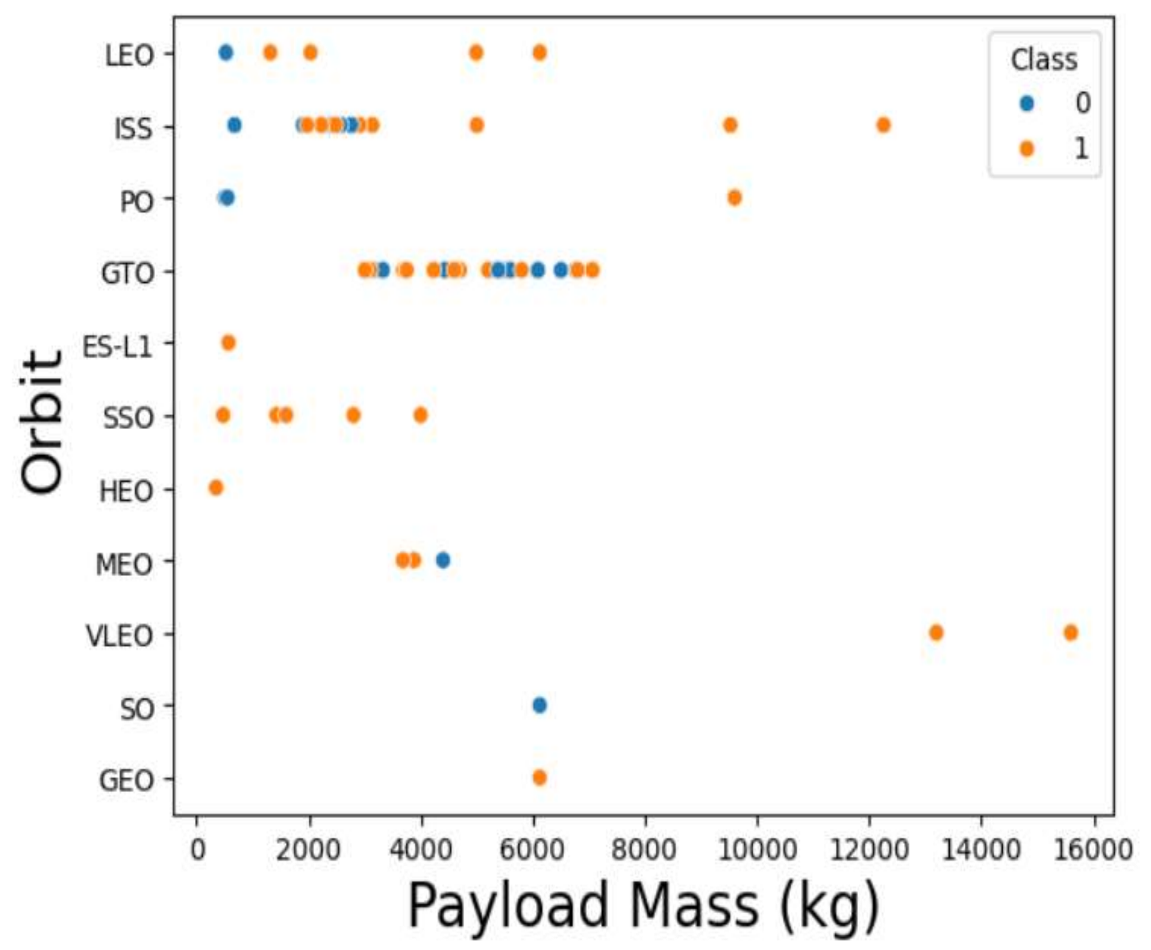
Flight Number vs. Orbit Type

- Show a scatter point of Flight number vs. Orbit type
- **A scatter plot that visualizes the relationship between Flight Number and Orbit type, with points color-coded to indicate whether the launch was a Success or Failure. This visualization can help to identify any patterns or trends in the data, such as whether certain orbits are more likely to result in successful launches or if the success rate changes over time (as indicated by the flight number).**



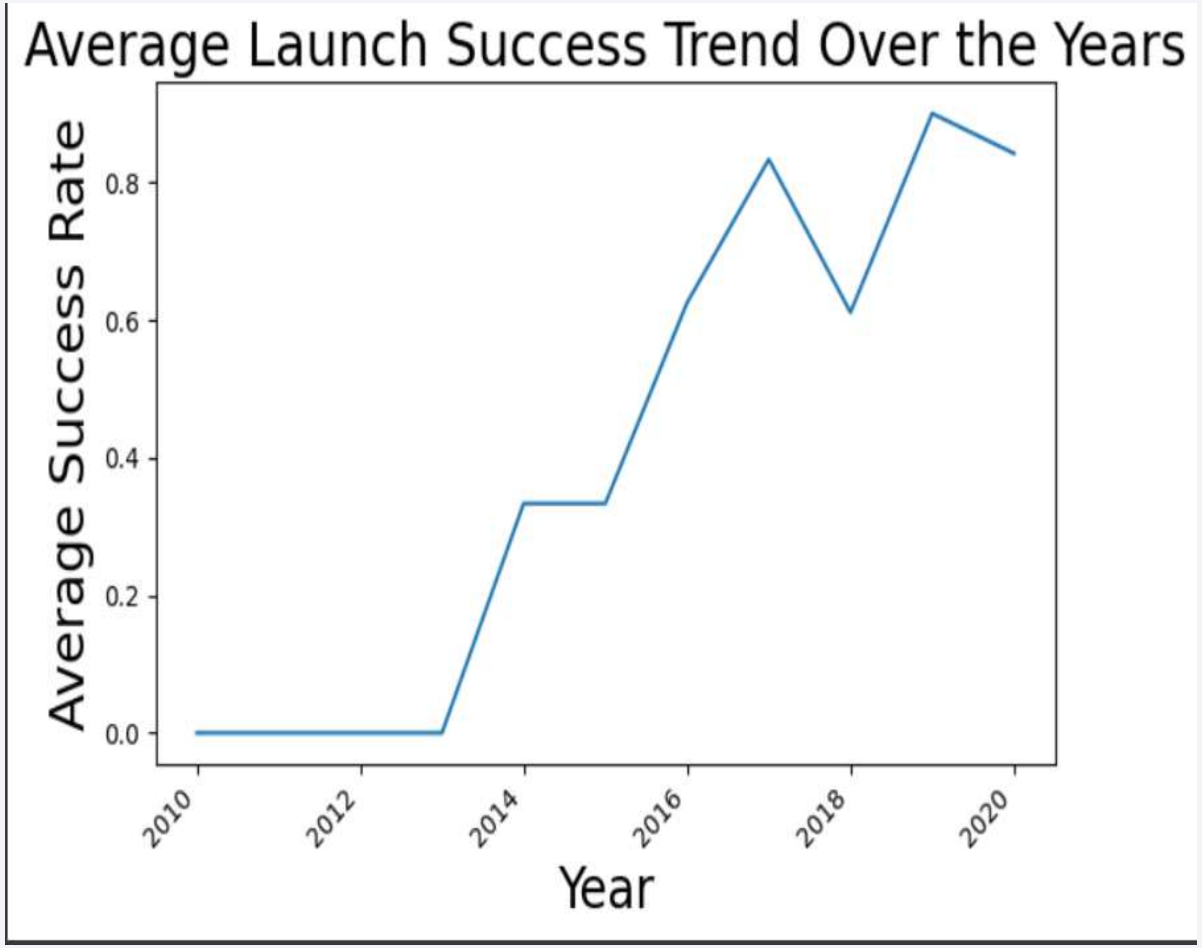
Payload vs. Orbit Type

- Show a scatter point of payload vs. orbit type
- A scatter plot to explore how the payload mass and target orbit of a SpaceX rocket might relate to the success or failure of the launch. It uses color to visually distinguish between successful and unsuccessful launches. The plot is then labeled for clarity and finally displayed to the user.



Launch Success Yearly Trend

- Show a line chart of yearly average success rate
- **SpaceX's launch success rates over time. It achieves this by first extracting the year from the launch date data within a DataFrame. Subsequently, it calculates the average success rate for each year by grouping the data and computing the mean. Using the extracted year and average success rate, the code generates a line chart, with the year plotted on the x-axis and the corresponding average success rate on the y-axis.**



this query will be a list of unique launch site names present in the SPACEXTABLE. This is useful to get an overview of all the places where launches have occurred, without repetition.

All Launch Site Names

- Find the names of the unique launch sites
- **This query will be a list of unique launch site names present in the** SPACEXTABLE. **This is useful to get an overview of all the places where launches have occurred, without repetition.**

```
# Now you should be able to run your query:
%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

Launch Site Names Begin with '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	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
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	Failure (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

- Find 5 records where launch sites begin with `CCA`
- **All the data from the** SPACEXTABLE **where the** Launch_Site **column starts with 'CCA', but only show me the first 5 rows that meet this condition."**

Total Payload Mass

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Customer LIKE '%NASA%' AND Payload LIKE '%CRS%';

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

SUM(PAYLOAD_MASS__KG_)
60268

- Calculate the total payload carried by boosters from NASA
- **This code is querying a database table named** SPACEXTABLE**. It is finding all the rows where the customer is NASA and the payload contains "CRS". Then, it calculates the total payload mass (in kilograms) for those specific missions by summing up the** PAYLOAD_MASS__KG_ **values from those selected rows.**

Average Payload Mass by F9 v1.1

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Booster_Version = 'F9 v1.1';

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

AVG(PAYLOAD_MASS__KG_)
2928.4

- Calculate the average payload mass carried by booster version F9 v1.1
- **It will calculate the average payload mass only for those specific rows that meet the condition in the** WHERE **clause**.

First Successful Ground Landing Date

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

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

MIN(Date)
2015-12-22

- Find the dates of the first successful landing outcome on ground pad
- **It is essentially searching for the date of the first successful landing on a ground pad within the dataset stored in** SPACEXTABLE**.**

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql SELECT DISTINCT Booster_Version FROM SPACEXTABLE WHERE Landing_Outcome LIKE '%Success (drone ship)%' AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000;

 * 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
- **a list of all the unique booster rocket versions used in SpaceX launches that landed successfully on a drone ship and carried a payload weighing between 4000 and 6000 kilograms.**

Total Number of Successful and Failure Mission Outcomes

```
%sql SELECT SUM(CASE WHEN Landing_Outcome LIKE '%Success%' THEN 1 ELSE 0 END) AS SuccessCount, SUM(CASE WHEN Landing_Outcome LIKE '%Failure%' THEN 1 ELSE 0 END) AS FailureCount FROM SPACEXTABLE;

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

SuccessCount	FailureCount
61	10

- Calculate the total number of successful and failure mission outcomes
- **This query goes through the** SPACEXTABLE**, examines the** Landing_Outcome **column for each row. If the outcome contains "Success", it adds 1 to the** SuccessCount**. If the outcome contains "Failure", it adds 1 to the** FailureCount**. Finally, it displays these two calculated totals. Essentially, it's counting the number of successful and failed landings recorded in the table.**

Boosters Carried Maximum Payload

```
%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
- **This SQL query is designed to find the** Booster_Version **associated with the launch that had the heaviest payload (maximum** PAYLOAD_MASS__KG_**). It does this by first finding the maximum payload mass and then selecting the booster version corresponding to that specific launch.**

The query retrieves the month, booster version, and launch site for all launches from the SPACEXTABLE that resulted in a "Failure (drone ship)" and occurred in the year 2015.

2015 Launch Records

```
# Assuming you've already set up the database and loaded data

%sql SELECT strftime('%Y-%m', Date) AS Month, Booster_Version, Launch_Site FROM SPACEXTABLE WHERE Landing_Outcome LIKE '%Failure (drone ship)%' AND strftime('%Y', Date) = '2015';
```

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

Month	Booster_Version	Launch_Site
2015-01	F9 v1.1 B1012	CCAFS LC-40
2015-04	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
- **The query retrieves the month, booster version, and launch site for all launches from the** SPACEXTABLE **that resulted in a "Failure (drone ship)" and occurred in the year 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

- 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
- **The query is designed to find out how many times each landing outcome occurred for SpaceX launches between June 4, 2010, and March 20, 2017, and then presents the results with the most frequent outcomes at the top.**

Section 3

Launch Sites Proximities Analysis

SpaceX Launch Site Analysis

- **In essence, through launch data, creates a marker for each launch at its specific location, assigns a color to the marker based on the launch outcome, and then adds the marker to the map, allowing for a visual representation of launch successes and failures.**

Visualizing SpaceX Mission Outcomes and Launch Site Locations

- **In an interactive map with markers representing SpaceX launch locations, color-coded to show launch success or failure, and grouped using clusters for a cleaner visualization.**

Visualizing SpaceX Launch Site Proximities

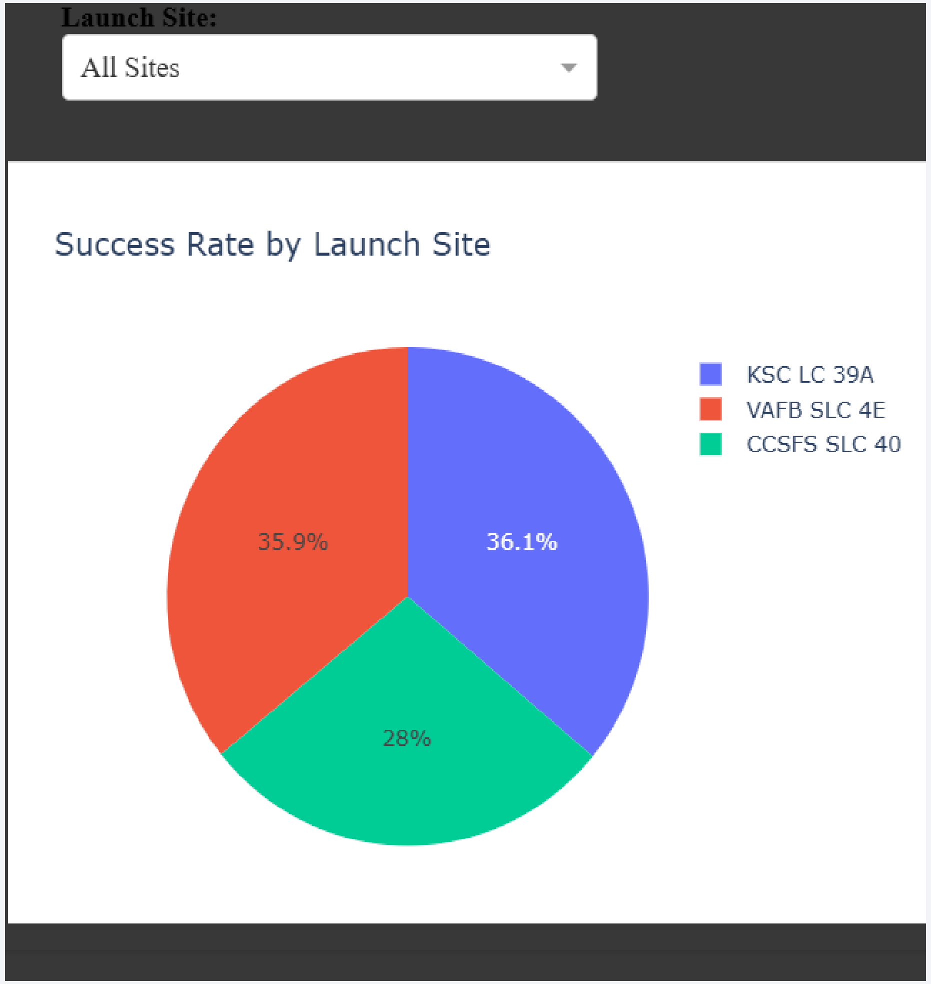
- **Provide a more informative map by highlighting the location of the nearest city** , railway, highway, coastline **to a launch site and their relative distance**.

Section 4

Build a Dashboard with Plotly Dash

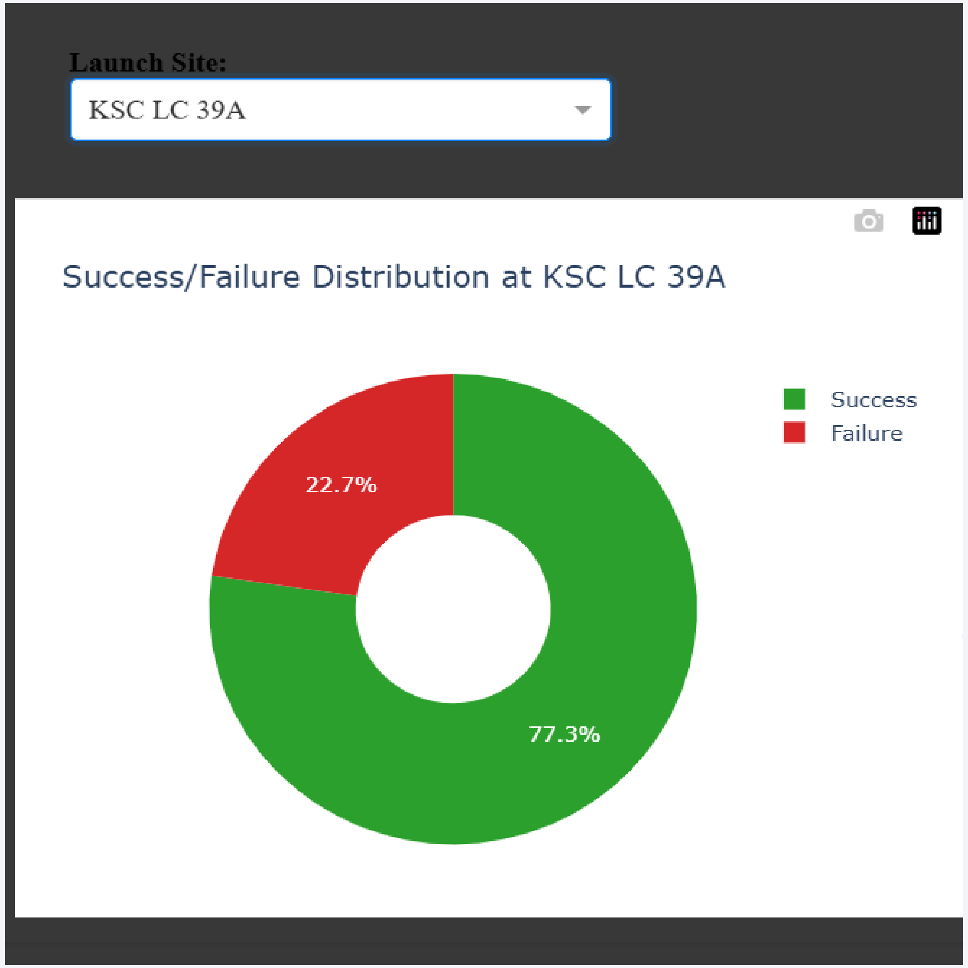
All Site Success Rate Piechart

- **When the user selects "All Sites" in the dropdown, the pie chart visualizes the success rate of launches for each individual launch site. Each slice of the pie represents a launch site, and the size of the slice corresponds to its success rate**



Specific Site Success Rate Piechart Using Dropdown

- **When a user selects a specific launch site from the dropdown, the pie chart updates to show the distribution of successful and failed launches only for that selected site. The chart is color-coded (green for success, red for failure) and displays the proportion of each outcome for the chosen launch site.**



SpaceX Launch Analytics Dashboard

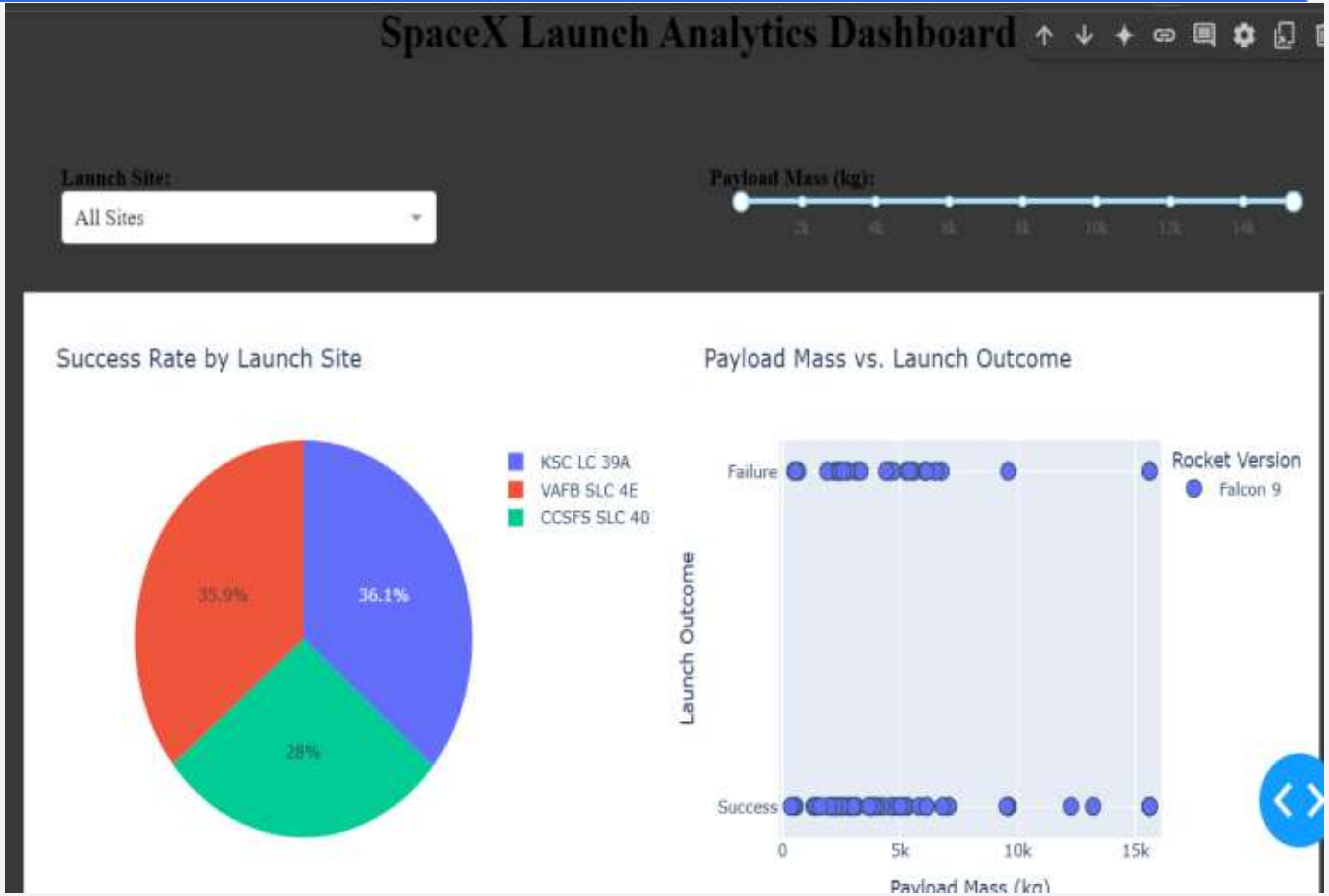
The dashboard allows users to interactively explore SpaceX launch data. It has:

1. Controls:

1. A dropdown to select a specific launch site or view data for all sites.
2. A slider to filter launches based on payload mass.

2. Visualizations:

1. A pie chart displaying success rate by launch site (or success/failure breakdown for a specific site).
2. A scatter plot showing the relationship between payload mass and launch outcome, colored by booster version.

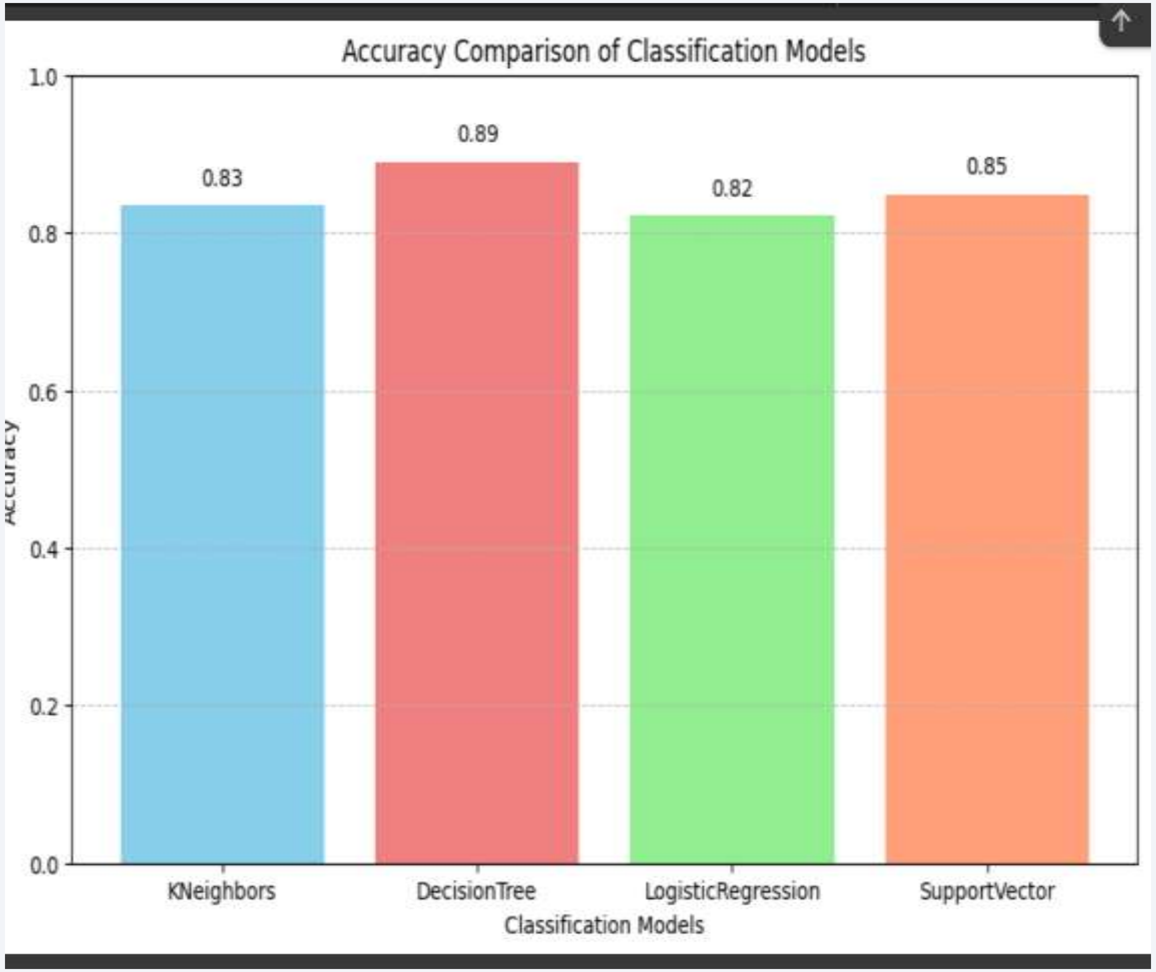


Section 5

Predictive Analysis (Classification)

Classification Accuracy

Decision Tree model has the highest classification accuracy with score of 89% Accuracy rate as shown in Bar Chart.

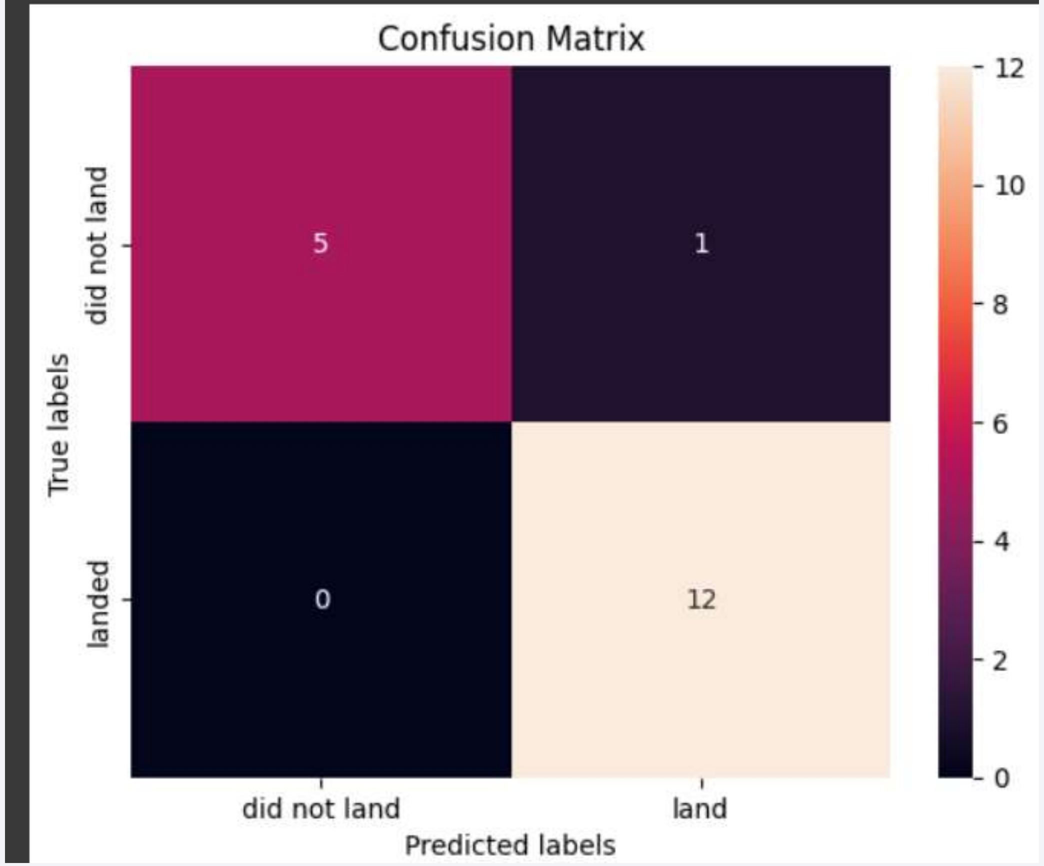


Confusion Matrix

The confusion matrix helps you understand the types of errors your Decision Tree is making. This is crucial for:

- **Identifying areas for improvement:** For instance, if you have a high number of false negatives, you might need to adjust the model to be more sensitive to the positive class.
- **Comparing different models:** You can compare the confusion matrices of different Decision Tree models (or other classification models) to see which one performs better overall or for specific classes.
- **Understanding the trade-offs:** There's often a trade-off between precision and recall. The confusion matrix helps you visualize this and choose the model that best suits your specific needs.

```
yhat = tree_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



Conclusions

- Project analyzed SpaceX launch data using various techniques.
- Data wrangling cleaned and preprocessed the data.
- Exploratory analysis revealed trends in launch outcomes.
- Visualizations highlighted key relationships.
- Interactive maps displayed geographical distributions.
- Predictive models forecasted launch success.
- Logistic Regression, SVM, Decision Tree, and KNN were compared.
- Project provided insights into SpaceX launch history.
- Analysis identified success-related factors.
- Interactive elements enhanced understanding.
- Project contributed to space exploration knowledge.

Appendix

A. Data Sources

- List datasets used, source URLs, and brief descriptions.
- Example: SpaceX Launch Data, Wikipedia Web Scraped Data.

B. Code Snippets

- Include relevant code sections with explanations.
- Focus on critical functions or analysis steps.
- Organize by project sections (e.g., wrangling, analysis).

C. Supplementary Visualizations

- Add extra charts or tables supporting findings.
- Provide concise captions explaining relevance.

D. External Libraries and Tools

- List libraries and tools with versions and purpose.
- Example: pandas, NumPy, matplotlib, seaborn, scikit-learn.

E. Glossary of Terms

- Define technical terms or acronyms for clarity.

Tools:-

- Coursera Python code snippets,
- SQL queries
- Python Libraries (Numpy, Pandas , Seaborn, etc)
- Google Colab
- Gemini (For project Report and Debugging)

Thank you!