IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Jorge Alberto Chavez Ponce
06/24/2024

Outline

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

Executive Summary

Methodologies

Data was gathered using the SpaceX REST API (api.spacexdata.com/v4/launches/past). Python's requests library was used to perform GET requests, and json_normalize converted JSON data to a Pandas DataFrame. Wikipedia pages related to Falcon 9 launches were scraped using BeautifulSoup. HTML tables were extracted and converted into Pandas DataFrames. Additional details were gathered from other API endpoints (Boosters, Launchpads, payloads, cores) and integrated into the dataset. The dataset was filtered to focus on Falcon 9 launches. Missing values in PayloadMass were replaced with column mean values. Target variable Class was extracted from the dataset, and features were standardized using StandardScaler. Data was split into training (80%) and testing (20%) sets using train_test_split. EDA was conducted using SQL and various visualization techniques to identify trends and patterns in the data. Interactive maps were created to visualize launch and landing sites. Dashboards were developed to enable dynamic data exploration. Logistic Regression, Support Vector Machine (SVM), Decision Tree, and K-Nearest Neighbors (KNN) models were built. GridSearchCV with 10-fold cross-validation was used for tuning. Model performance was evaluated using accuracy, precision, recall, and F1-score. Confusion matrices were plotted for visual inspection.

Executive Summary

Results

Successful launches were most frequent in the 0-4000 kg payload range, while failures were more spread out but occurred more frequently below 2000 kg and above 4000 kg. The FT and B5 booster versions had higher success rates, especially within the 1000-4000 kg range, while the v1.0 and v1.1 versions showed more scattered results.

Model Performance:

- Decision Tree: Achieved the highest classification accuracy.
- Confusion Matrix: For the best-performing model (Decision Tree), true positives (TP) and true negatives (TN) were accurately predicted, with few false positives (FP) and false negatives (FN).

Introduction

The commercial space age has arrived, with companies like Virgin Galactic, Rocket Lab, Blue Origin, and SpaceX making space travel more affordable. SpaceX has achieved notable successes, such as:

- Sending spacecraft to the International Space Station (ISS)
- Developing Starlink, a satellite internet constellation
- Conducting manned space missions

A key factor in SpaceX's cost-effectiveness is the reuse of the first stage of its Falcon 9 rockets, reducing launch costs to around $62 million compared to $165 million from other providers.

As a data scientist for Space Y, founded by billionaire Allon Musk, my tasks are:

- Determine the Price of Each Launch:
  - Gather and analyze information about SpaceX launches.
  - Create dashboards to visualize cost components.
- Predict the Reusability of the First Stage:
  - Use machine learning to predict if SpaceX will reuse the first stage based on public information.

These analyses will help Space Y compete with SpaceX by optimizing launch costs and strategies.

Section 1

Methodology

Methodology

Executive Summary

- Data Collection Methodology:
  - Gathered data from the SpaceX REST API (api.spacexdata.com/v4/launches/past).
  - Performed GET requests with the requests library to retrieve JSON data.
  - Used json_normalize to convert JSON data into a flat table.
  - Web scraped Wiki pages using BeautifulSoup to extract additional launch data.
- Data Wrangling:
  - Integrated detailed data from additional API endpoints (Booster, Launchpad, payload, core).
  - Filtered out Falcon 1 launches to focus on Falcon 9.
  - Handled missing values by replacing NULLs in PayloadMass with the column's mean value.

Methodology

Executive Summary

- Exploratory Data Analysis (EDA):
  - Conducted EDA using SQL and visualization techniques.
  - Generated summary statistics and visualizations to identify trends and patterns.
- Interactive Visual Analytics:
  - Created interactive maps with Folium to visualize launch and landing sites.
  - Developed interactive dashboards with Plotly Dash for dynamic data exploration.
- Predictive Analysis Using Classification Models:
  - Built and fine-tuned classification models (logistic regression, decision trees, svm) to predict first stage landings.
  - Evaluated models using accuracy, precision, recall, and F1-score.

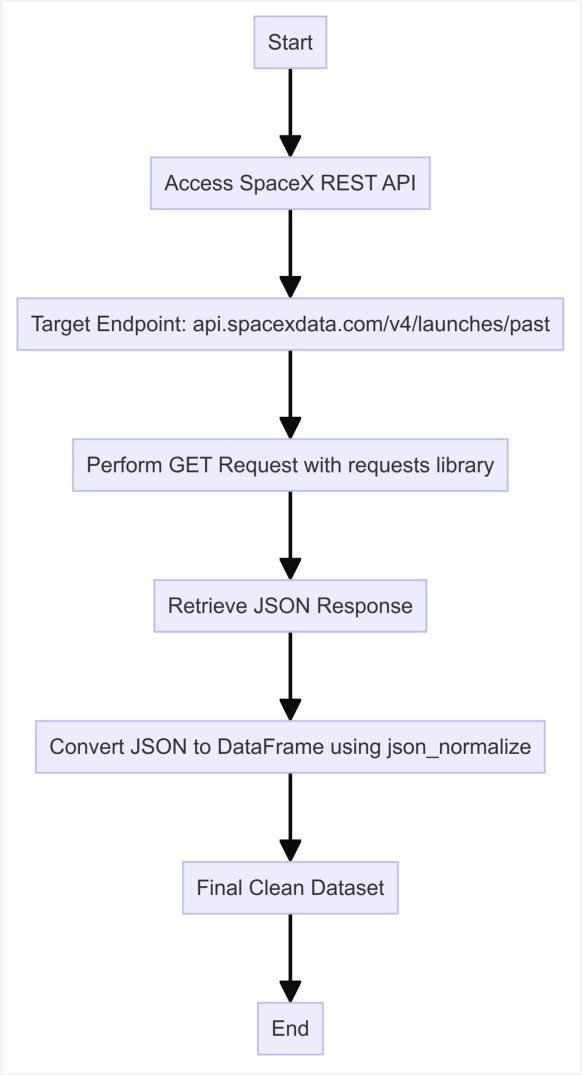
Data Collection

- SpaceX REST API
  - API Endpoint: api.spacexdata.com/v4/launches/past
  - Method: Used Python requests library to perform GET requests.
  - Data Format: JSON, converted to Pandas DataFrame using json_normalize.
- Web Scraping
  - Source: Wikipedia pages related to Falcon 9 launches.
  - Method: Used BeautifulSoup to extract HTML tables.
  - Data Processing: Parsed HTML tables and converted to Pandas DataFrame.
- Data Integration
  - Additional Endpoints: Used for detailed information on Boosters, Launchpads, payloads, and cores.
  - Method: Combined data from multiple endpoints to enrich dataset.
- Data Cleaning
  - Filtering: Removed Falcon 1 launches to focus on Falcon 9.
  - Handling Missing Values: Replaced NULL values in PayloadMass with mean values.
  - One-Hot Encoding: Prepared categorical columns for analysis.

Data Collection – SpaceX API

- Initiate Data Collection: Start by accessing the SpaceX REST API.
- API Endpoint: Target endpoint api.spacexdata.com/v4/launches/past for past launches.
- HTTP GET Request: Use Python requests library to perform GET requests.
- Retrieve JSON Response: Obtain data in JSON format.
- JSON to DataFrame: Convert JSON data to Pandas DataFrame using json_normalize

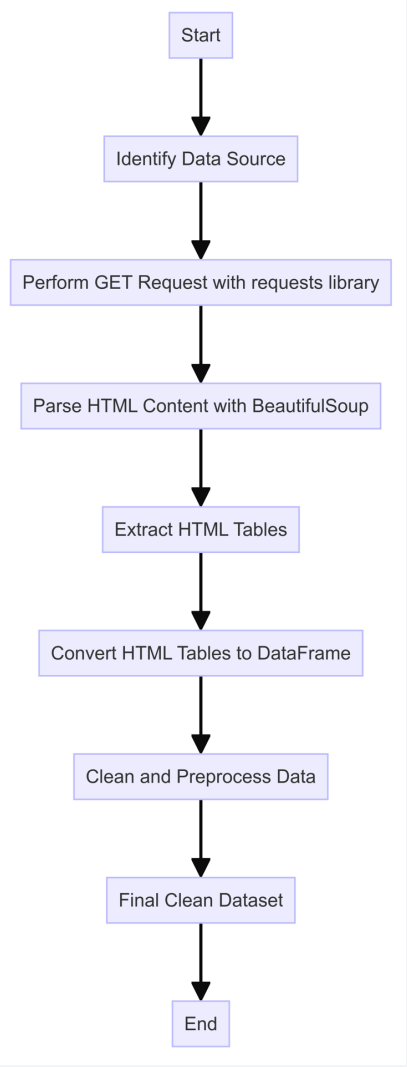
SpaceX API calls notebook GitHubURL



Data Collection - Scraping

- Identify Data Source: Locate relevant Wikipedia pages for Falcon 9 launch data.
- HTML Request: Use Python requests library to perform GET requests to the identified URLs.
- Parse HTML Content: Utilize BeautifulSoup to parse the HTML content of the web pages.
- Extract Data: Extract HTML tables containing launch records.
- Convert to DataFrame: Convert the extracted tables into Pandas DataFrames.
- Clean Data: Clean and preprocess the data for analysis.

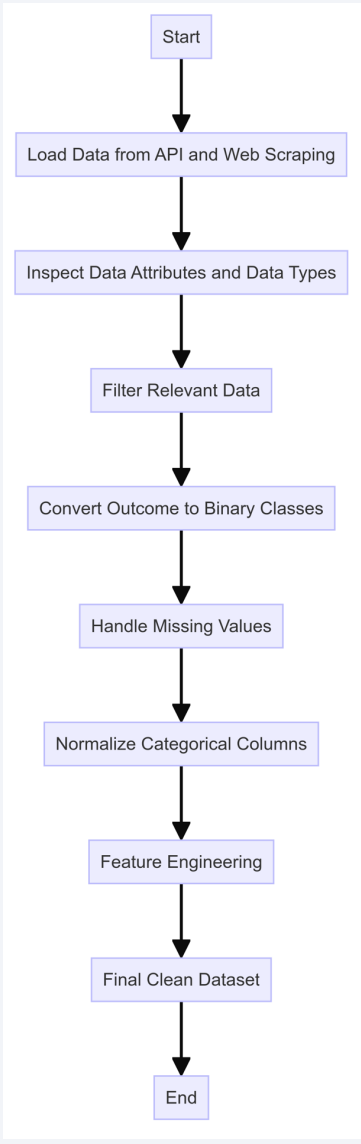
[Web scraping notebook GitHub URL]



Data Wrangling

- Data Attributes: Flight Number, Date, Booster Version, Payload Mass, Orbit, Launch Site, Outcome, Flights, Grid Fins, Reused, Legs, Landing Pad, Block, Reused Count, Serial, Longitude, Latitude.
- Load Data: Import data from the SpaceX API and web scraping sources into a Pandas DataFrame.
- Inspect Data: Review attributes and data types.
- Filter Relevant Data: Focus on Falcon 9 launches and relevant columns.
- Convert Outcome to Classes: Transform landing outcomes to binary classes (0 or 1).
- Handle Missing Values: Replace NULL values in Payload Mass with the mean value.
- Normalize Data: Use one-hot encoding for categorical columns like Launch Site and Orbits.
- Feature Engineering: Create new features or modify existing ones to improve model performance.
- Final Clean Dataset: Ensure data is clean and ready for analysis and modeling.

[Data wrangling notebook GitHub URL]



EDA with Data Visualization

- FlightNumber vs. PayloadMass with Launch Outcome Overlay:
  - Chart Type: Scatter Plot
  - Purpose: To show how flight experience (number of flights) and payload mass impact the success of the first stage landing.
- FlightNumber vs. LaunchSite with Outcome Hue:
  - Chart Type: Categorical Plot (Catplot)
  - Purpose: To identify if certain launch sites have higher success rates based on flight number and outcome.
- PayloadMass vs. LaunchSite:
  - Chart Type: Scatter Plot
  - Purpose: To highlight the distribution of payload masses at different launch sites and identify sites that do not handle heavy payloads.

EDA with Data Visualization

- Success Rate by Orbit Type:
  - Chart Type: Bar Chart
  - Purpose: To compare the success rates across different orbit types and identify which orbits have higher or lower success rates.
- FlightNumber vs. Orbit Type:
  - Chart Type: Line Chart
  - Purpose: To analyze how repeated launches to specific orbits affect success rates over time.
- Year vs. Average Success Rate:
  - Chart Type: Line Chart
  - Purpose: To observe the trend in launch success rates over the years, indicating improvements or changes in success rates over time.

EDA with data visualization notebook GitHub URL

EDA with SQL

- Display unique launch sites
  - Queried the distinct values in the Launch_Site column to identify unique launch sites.
- Display total payload mass for NASA (CRS) launches
  - Queried the sum of PAYLOAD_MASS__KG_ for launches with Customer as 'NASA (CRS)'.
- Display average payload mass for booster version F9 v1.1
  - Queried the average of PAYLOAD_MASS__KG_ for launches with Booster_Version as 'F9 v1.1'.
- List date of first successful landing on ground pad
  - Queried the minimum Date where Landing_Outcome was 'Success'.
- List boosters with successful drone ship landings and specific payload mass range
  - Queried distinct Booster_Version values where Landing_Outcome was 'Success' and PAYLOAD_MASS__KG_ was between 4000 and 6000 kg.

EDA with SQL

- Display total number of successful and failed mission outcomes
  - Queried the count of records for Mission_Outcome 'Success'.
  - Queried the count of records for Mission_Outcome starting with 'Failure'.
- List boosters carrying the maximum payload mass
  - Queried the Booster_Version with the maximum value of PAYLOAD_MASS__KG_.
- List records with failure outcomes in 2015 with specific details
  - Queried the Month, Booster_Version, Launch_Site, and Landing_Outcome for records in 2015 with Landing_Outcome as 'Failure (drone ship)'.
- Rank landing outcomes between specific dates
  - Queried the count of Landing_Outcome values grouped by outcome type, between '2010-06-04' and '2017-03-20', ordered by count in descending order.

EDA with SQL notebook GitHub URL

Build an Interactive Map with Folium

- NASA Johnson Space Center Marker and Circle:
  - Marker: Located at NASA Johnson Space Center, Houston, Texas.
  - Circle: A circle with a radius of 1000 meters around the center.
  - Reason: To set a reference point for the map and mark NASA's Johnson Space Center.
- Launch Sites Markers and Circles:
  - Markers and Circles: Added for each SpaceX launch site with coordinates from the launch_sites_df dataframe.
  - Reason: To visually mark the locations of all SpaceX launch sites on the map.
- Launch Outcomes Markers (MarkerCluster):
  - Markers: Green markers for successful launches and red markers for failed launches.
  - Reason: To show the outcome of each launch at the respective launch sites, making it easy to identify success rates.

Build an Interactive Map with Folium

- Distance to Coastline Marker and Line:
  - Marker: Shows the distance from the launch site to the closest coastline.
  - Line: A line drawn between the launch site and the coastline point.
  - Reason: To measure and visualize the proximity of the launch site to the coastline.
- Distance to Railway Marker and Line:
  - Marker: Shows the distance from the launch site to the closest railway.
  - Line: A line drawn between the launch site and the railway point.
  - Reason: To measure and visualize the proximity of the launch site to a railway.

Completed interactive map with Folium map GitHub URL

Build a Dashboard with Plotly Dash

- Dropdown for Launch Site Selection:
  - Element: Dropdown menu.
  - Purpose: Allows the user to select a specific launch site or view data for all launch sites.
  - Interaction: When a user selects a launch site, the pie chart and scatter plot update to reflect the data for the selected site.
- Pie Chart for Launch Success:
  - Element: Pie chart.
  - Purpose: Displays the total count of successful and failed launches.
  - Interaction: Updates based on the launch site selected in the dropdown. If "All Sites" is selected, it shows the success vs. failure counts for all sites combined. If a specific site is selected, it shows the success vs. failure counts for that site.

Build a Dashboard with Plotly Dash

- Payload Range Slider:
  - Element: Range slider.
  - Purpose: Allows the user to filter the data based on the payload mass range.
  - Interaction: Updates the scatter plot to display data within the selected payload range.
- Scatter Plot for Payload vs. Launch Success:
  - Element: Scatter plot.
  - Purpose: Shows the correlation between payload mass and launch success.
  - Interaction: Updates based on the selected launch site and the payload range. If "All Sites" is selected, it displays the correlation for all sites. If a specific site is selected, it shows the correlation for that site only.for all sites combined. If a specific site is selected, it shows the success vs. failure counts for that site.

[Plotly Dash lab GitHub URL]

Predictive Analysis (Classification)

1. Loading and Preparing Data

- Load Data: Loaded datasets (dataset_part_2.csv and dataset_part_3.csv) into pandas DataFrames.
- Extract Features and Labels: Extracted the target variable (Class) from the first dataset and the features from the second dataset.
- Standardize Features: Standardized the features using StandardScaler from scikit-learn.

2. Splitting Data

- Train-Test Split: Split the data into training (80%) and testing (20%) sets using train_test_split.

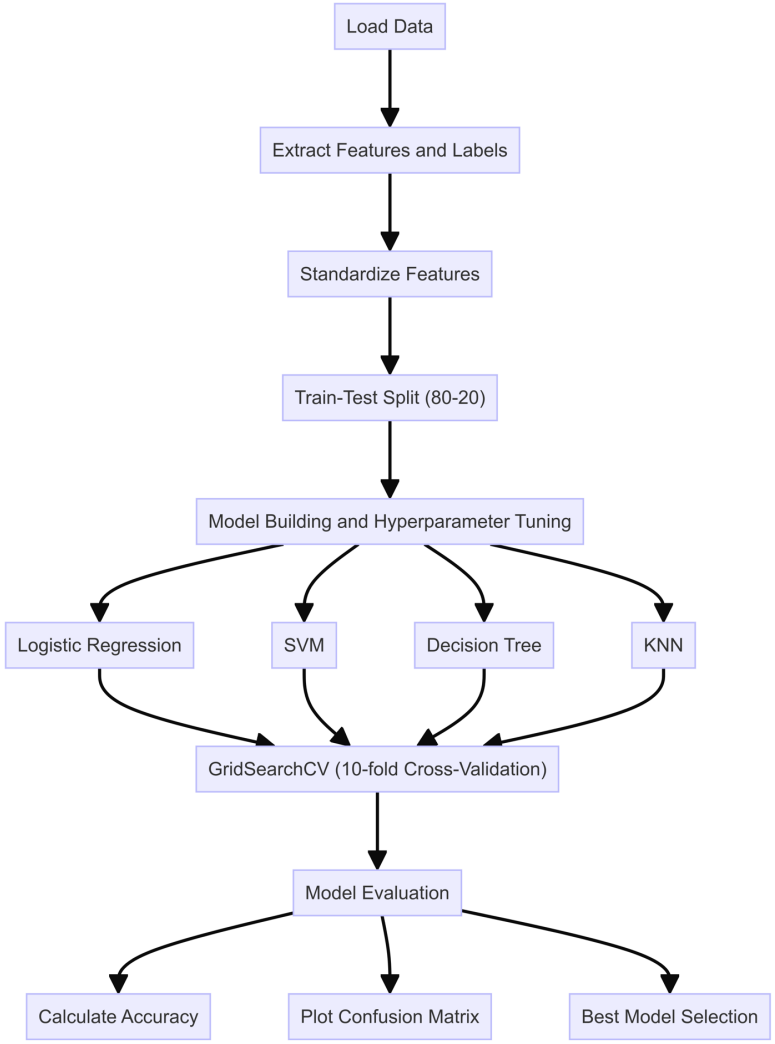
Predictive Analysis (Classification)

3. Model Building and Hyperparameter Tuning

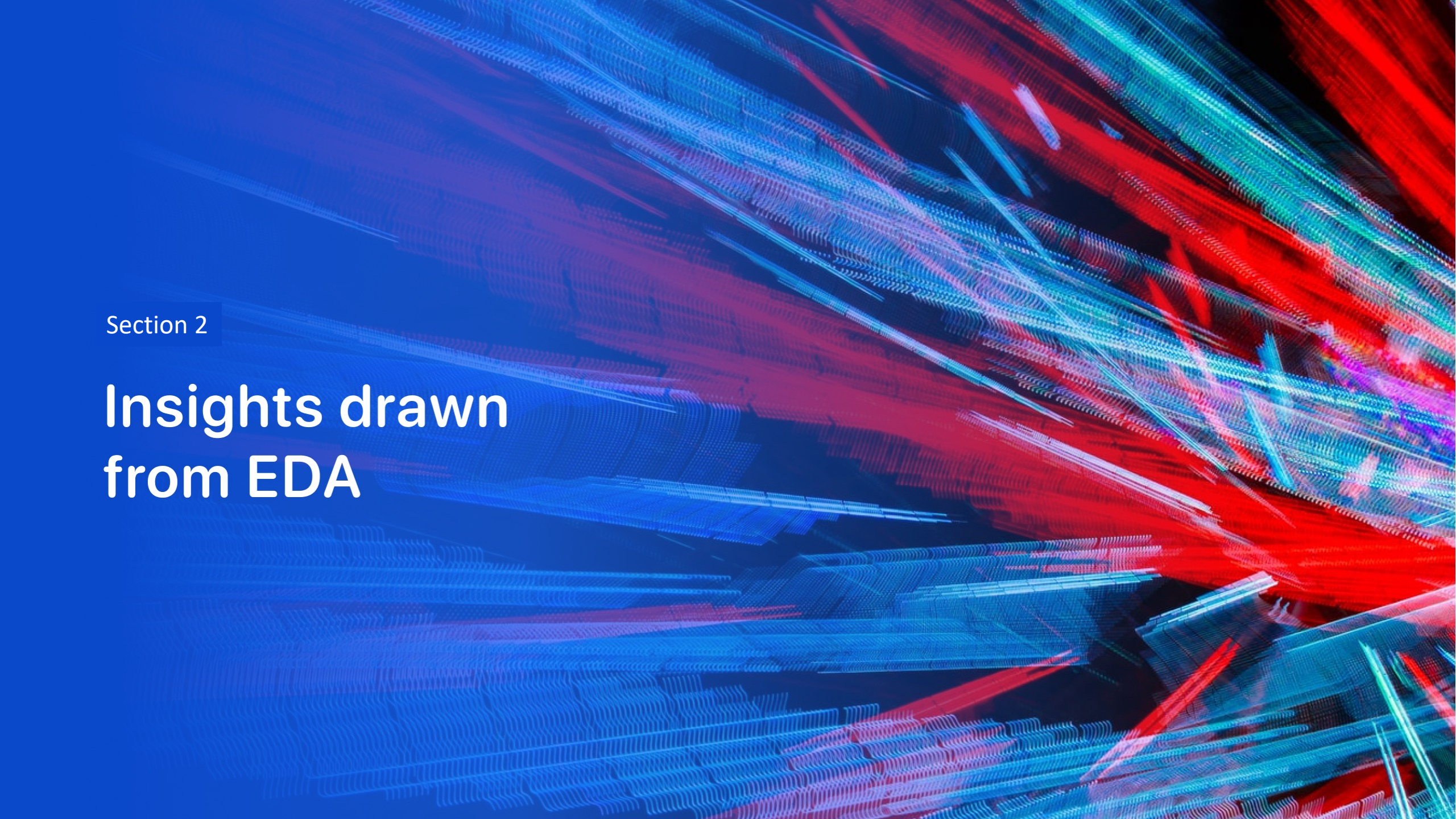
- Models: Logistic Regression, Support Vector Machine (SVM), Decision Tree, K-Nearest Neighbors (KNN)
- Hyperparameter Tuning: Used GridSearchCV with 10-fold cross-validation to find the best parameters.
- Evaluation: Evaluated model performance on test data and visualized results using a confusion matrix.

4. Model Evaluation and Selection

- Accuracy Calculation: Calculated the accuracy for each model on the test set.
- Confusion Matrix: Plotted confusion matrices for visual inspection of each model's performance.
- Best Model Selection: Compared the accuracies and selected the model with the highest accuracy as the best performing model.



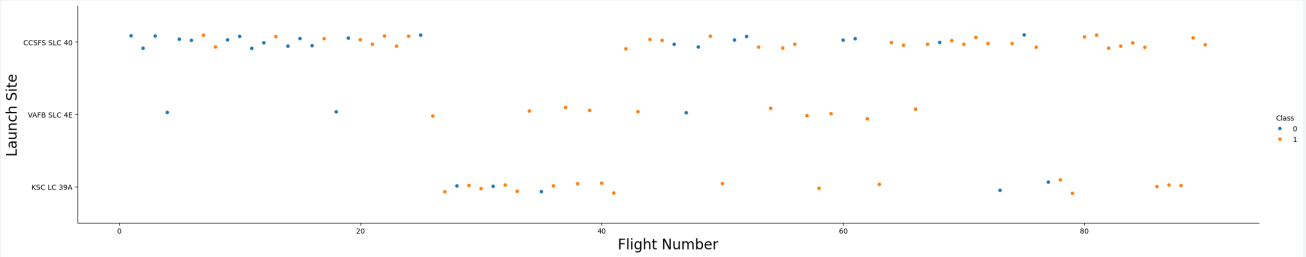
[Predictive Analysis lab GitHub URL]


Section 2
Insights drawn from EDA

Flight Number vs. Launch Site

This scatter plot visualizes the relationship between flight numbers and launch sites. It can help identify patterns or trends in launch success rates at different sites over time.

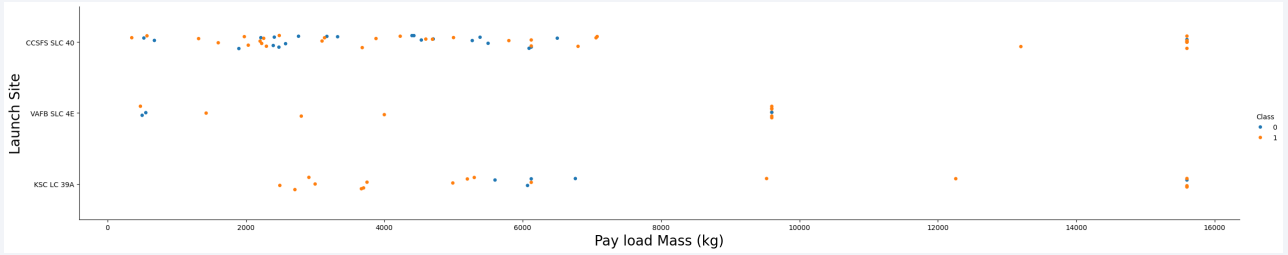
- Flight Number: Represents the sequence of launches.
- Launch Site: Indicates where the launch took place.
- Class (Hue): Differentiates successful and unsuccessful launches.
- Insight: Observing the scatter plot might reveal if certain launch sites have a higher frequency of successful flights over time.



Payload vs. Launch Site

This scatter plot shows the relationship between the payload mass and the launch site, which can indicate if specific sites are used for heavier payloads.

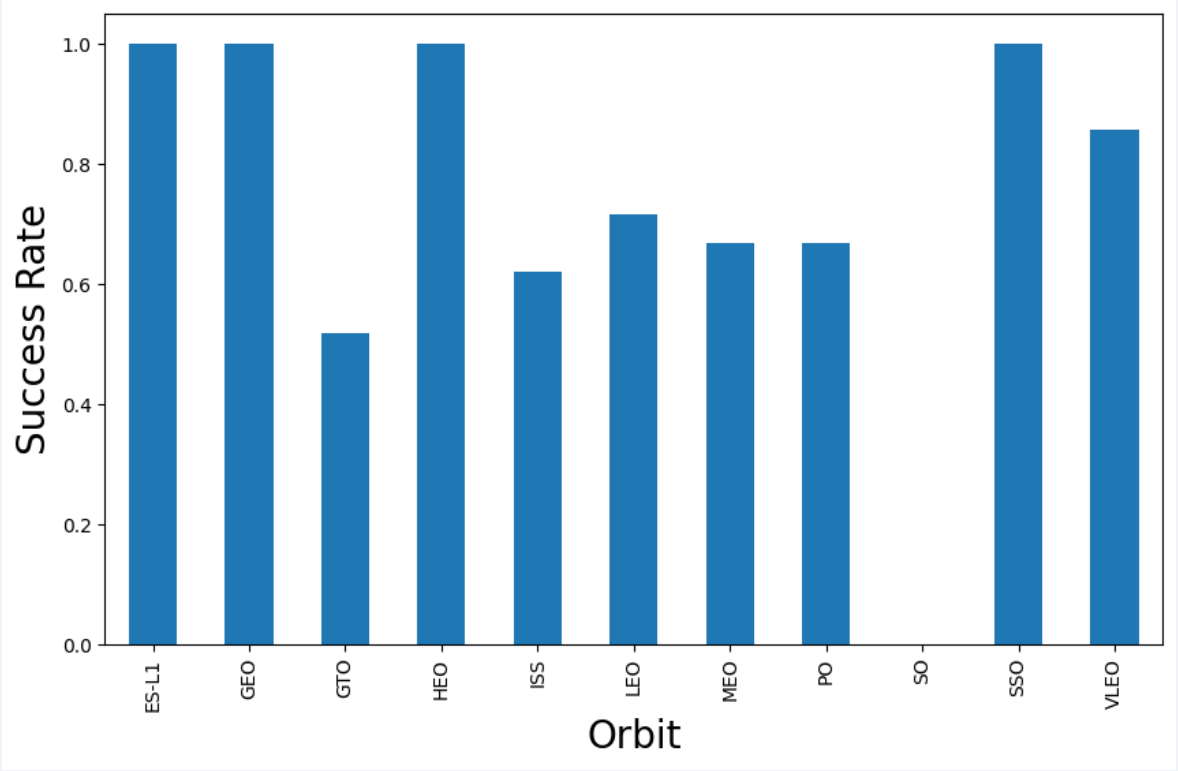
- Payload Mass: The mass of the payload being launched.
- Launch Site: Indicates where the launch took place.
- Class (Hue): Differentiates successful and unsuccessful launches.
- Insight: This plot might show if certain launch sites handle heavier payloads and their corresponding success rates.



Success Rate vs. Orbit Type

This bar chart visualizes the success rates of different orbit types, helping to understand which orbits have higher chances of successful launches.

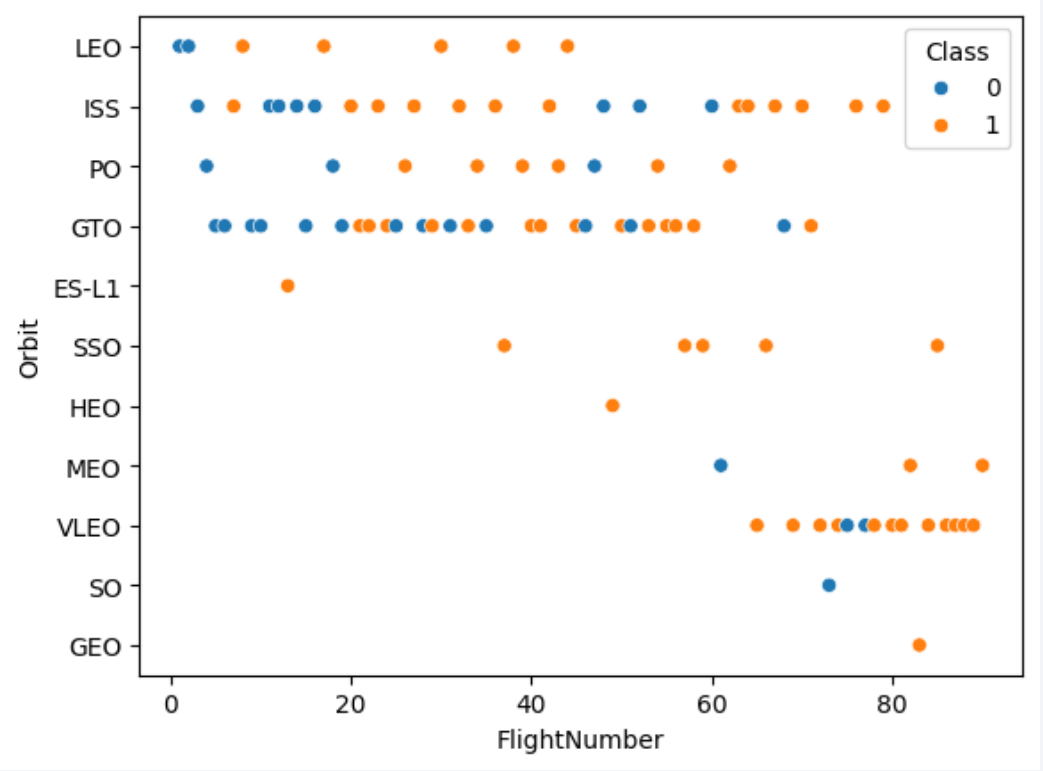
- Orbit: Different types of orbits (e.g., LEO, GTO).
- Success Rate: The average success rate for each orbit type.
- Insight: This chart can highlight which orbits are more reliable for successful launches.



Flight Number vs. Orbit Type

This scatter plot shows the relationship between flight numbers and orbit types, which can reveal trends in launch success over time for different orbits.

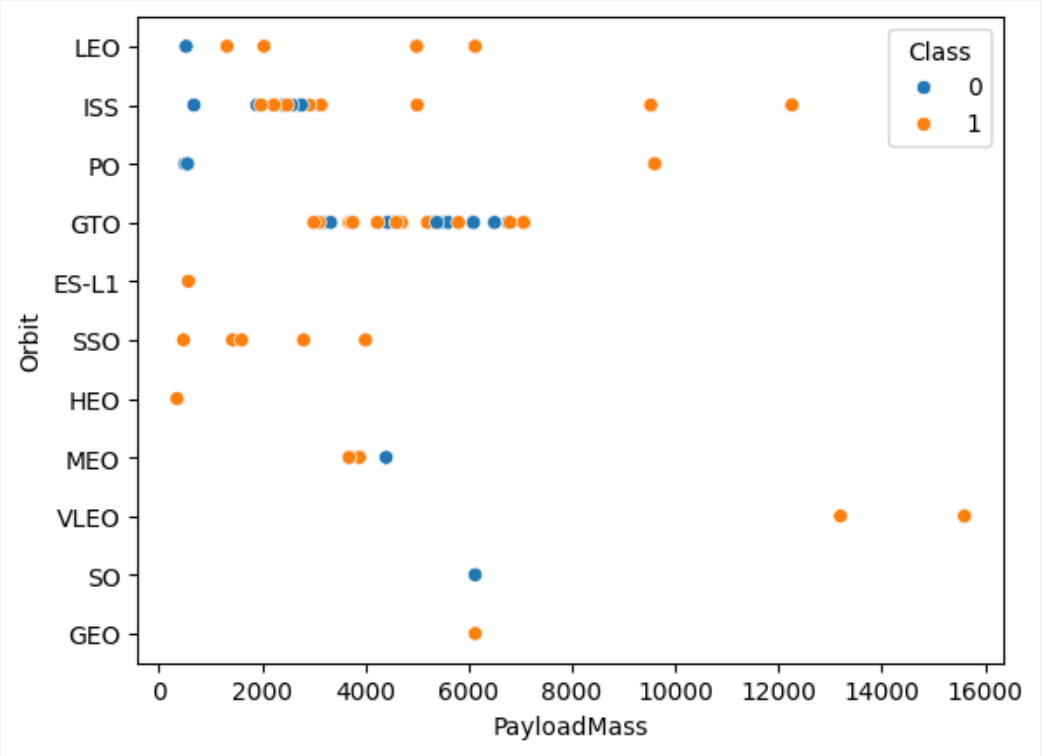
- Flight Number: Represents the sequence of launches.
- Orbit: Different types of orbits.
- Class (Hue): Differentiates successful and unsuccessful launches.
- Insight: This plot might show how success rates vary across different orbit types over time.



Payload vs. Orbit Type

This scatter plot visualizes the relationship between payload mass and orbit type, helping to identify trends in payload distribution across different orbits.

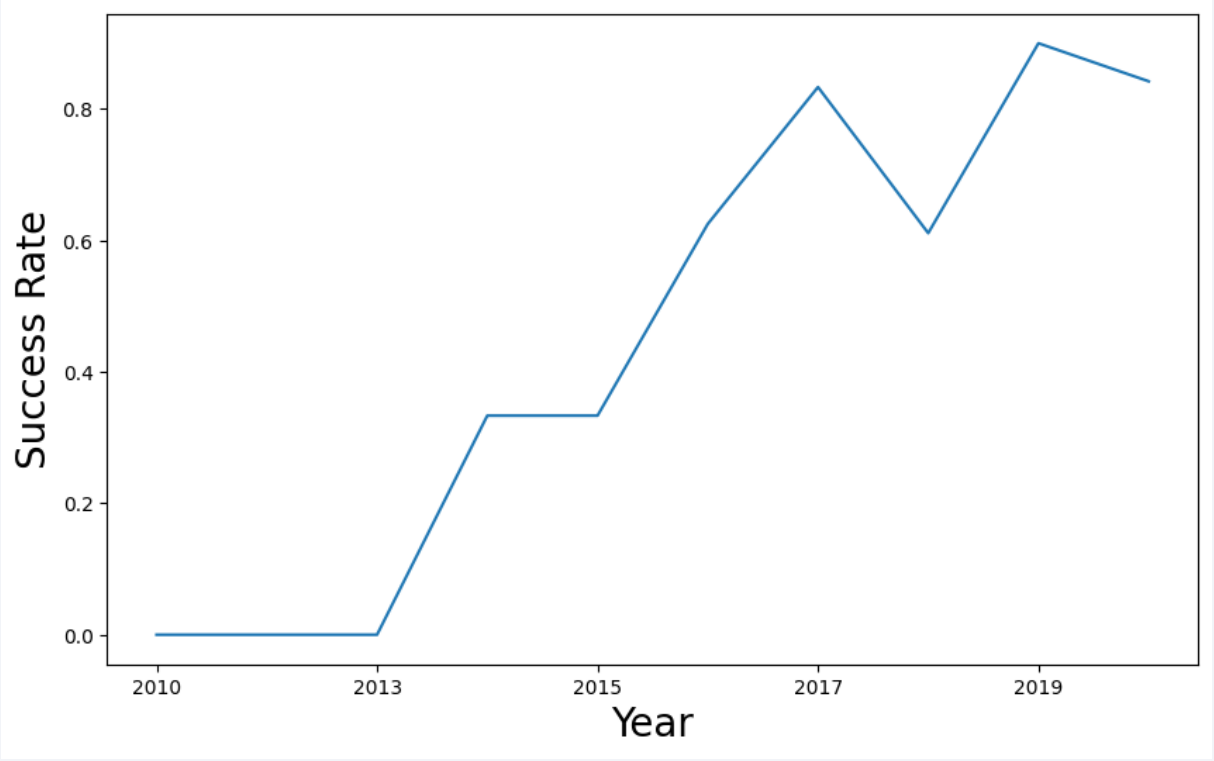
- Payload Mass: The mass of the payload being launched.
- Orbit: Different types of orbits.
- Class (Hue): Differentiates successful and unsuccessful launches.
- Insight: This plot might indicate which orbits are preferred for heavier or lighter payloads and their success rates.



Launch Success Yearly Trend

This line chart shows the trend of average success rates over the years, providing a historical perspective on the improvement or decline in launch success.

- Year: The year of the launch.
- Success Rate: The average success rate for each year.
- Insight: This chart can show the trend in launch success rates over time, indicating improvements or challenges in launch technology and operations.



All Launch Site Names

- Query:

```
%sql SELECT DISTINCT `Launch_Site` FROM SPACEXTABLE
✓ 0.0s
```

- A list of all unique launch sites from the SPACEXTABLE. This shows the various locations from which launches have been conducted.

Launch Site Names Begin with 'CCA'

- Query:

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

- A list of launch sites from the SPACEXTABLE table that start with "CCA". This typically includes all launch sites located at Cape Canaveral.

Launch_Site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

Total Payload Mass

- Query:

```
%sql SELECT SUM(`PAYLOAD_MASS__KG_`) FROM SPACEXTABLE WHERE `Customer` = 'NASA (CRS)'
```

- The total payload mass (in kilograms) of all launches where the customer is 'NASA (CRS)'. This provides an aggregate figure for the amount of payload launched for NASA's Commercial Resupply

SUM(`PAYLOAD_MASS__KG_`)
45596

Average Payload Mass by F9 v1.1

- Query:

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

- The average payload mass (in kilograms) for launches using the 'F9 v1.1' booster version. This indicates the typical payload capacity for this specific version of the booster.

AVG(`PAYLOAD_MASS__KG_`)
2928.4

First Successful Ground Landing Date

- Query:

```
%sql SELECT MIN(`Date`) as First_Successful_Landing FROM SPACEXTABLE WHERE `Landing_Outcome` = 'Success';
```

- The date of the first successful landing. This indicates when the first successful landing occurred in the dataset.

First_Successful_Landing
2018-07-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- Query:

```
%sql SELECT DISTINCT `Booster_Version` FROM SPACEXTABLE WHERE
`Landing_Outcome` = 'Success' AND `PAYLOAD_MASS__KG_` > 4000 AND
`PAYLOAD_MASS__KG_` < 6000;
```

- A list of unique booster versions that have successfully landed and carried a payload mass between 4000 kg and 6000 kg. This identifies which booster versions are capable of handling medium payloads and landing successfully.

Booster_Version
F9 B5 B1046.2
F9 B5 B1047.2
F9 B5 B1048.3
F9 B5 B1051.2
F9 B5B1060.1
F9 B5 B1058.2
F9 B5B1062.1

Total Number of Successful and Failure Mission Outcomes

- Query:

```
%sql SELECT COUNT(*) FROM SPACEXTABLE WHERE `Mission_Outcome` = 'Success' OR `Mission_Outcome` LIKE 'Failure%';
```

- The total number of launches that resulted in either a success or some form of failure. This provides a count of all recorded mission outcomes that were not indeterminate.

COUNT(*)
99

Boosters Carried Maximum Payload

- Query:

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

- The booster version that was used for the launch with the maximum payload mass. This identifies the specific booster version associated with the heaviest payload in the dataset.

Booster_Version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

2015 Launch Records

- Query:

```
%sql SELECT substr(`Date`, 6, 2) as Month, `Booster_Version`,
`Launch_Site`, `Landing_Outcome` FROM SPACEXTABLE WHERE
`Landing_Outcome` = 'Failure (drone ship)' AND substr(`Date`, 0,
5) = '2015';
```

- A list of launches that failed on a drone ship in 2015, showing the month, booster version, launch site, and landing outcome. This provides details on specific failed landings on drone ships for the year 2015.

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

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

- Query:

```
%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;
```

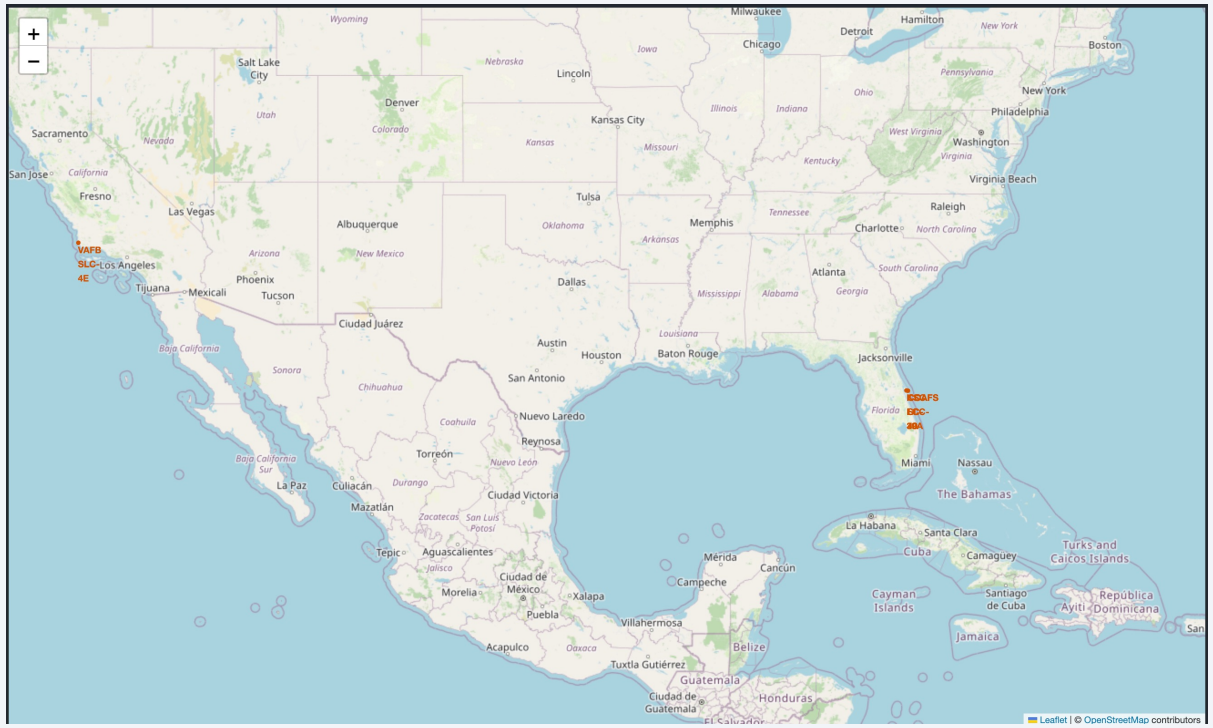
Landing_Outcome	Count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

- The count of each landing outcome between June 4, 2010, and March 20, 2017, ordered by the number of occurrences in descending order. This shows the frequency of different landing outcomes over the specified period.


Section 3
Launch Sites
Proximities Analysis

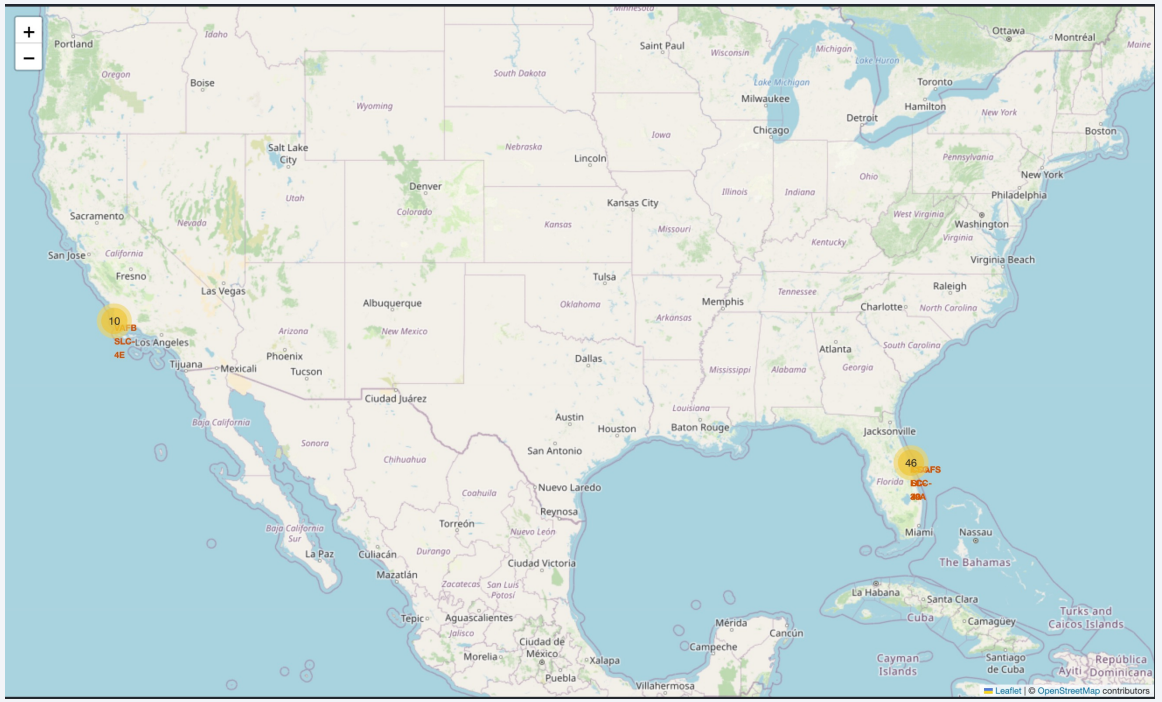
Marking Launch Sites on the Map

- Objective: Identify the geographical locations of SpaceX launch sites.
- Findings: The launch sites are located at:
  - CCAFS LC-40
  - CCAFS SLC-40
  - KSC LC-39A
  - VAFB SLC-4E
- Observations: Most launch sites are close to the coast and relatively near the equator, which is advantageous for launching rockets into orbit due to the Earth's rotational speed.



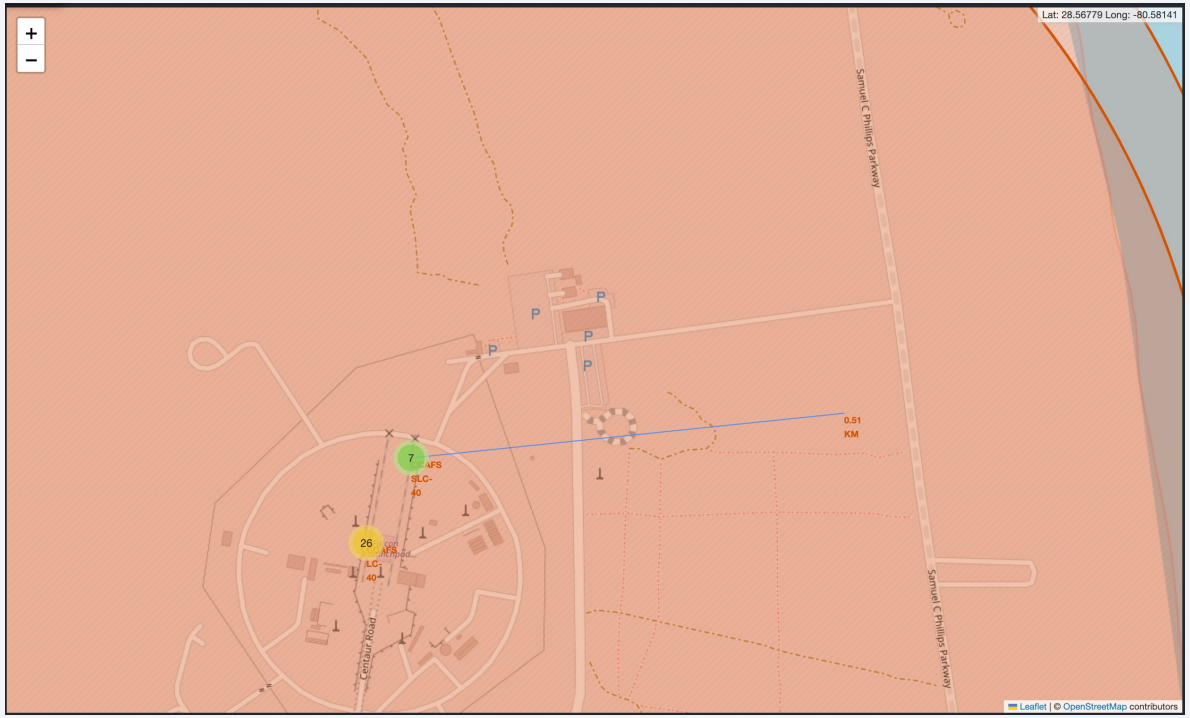
Marking Success/Failed Launches

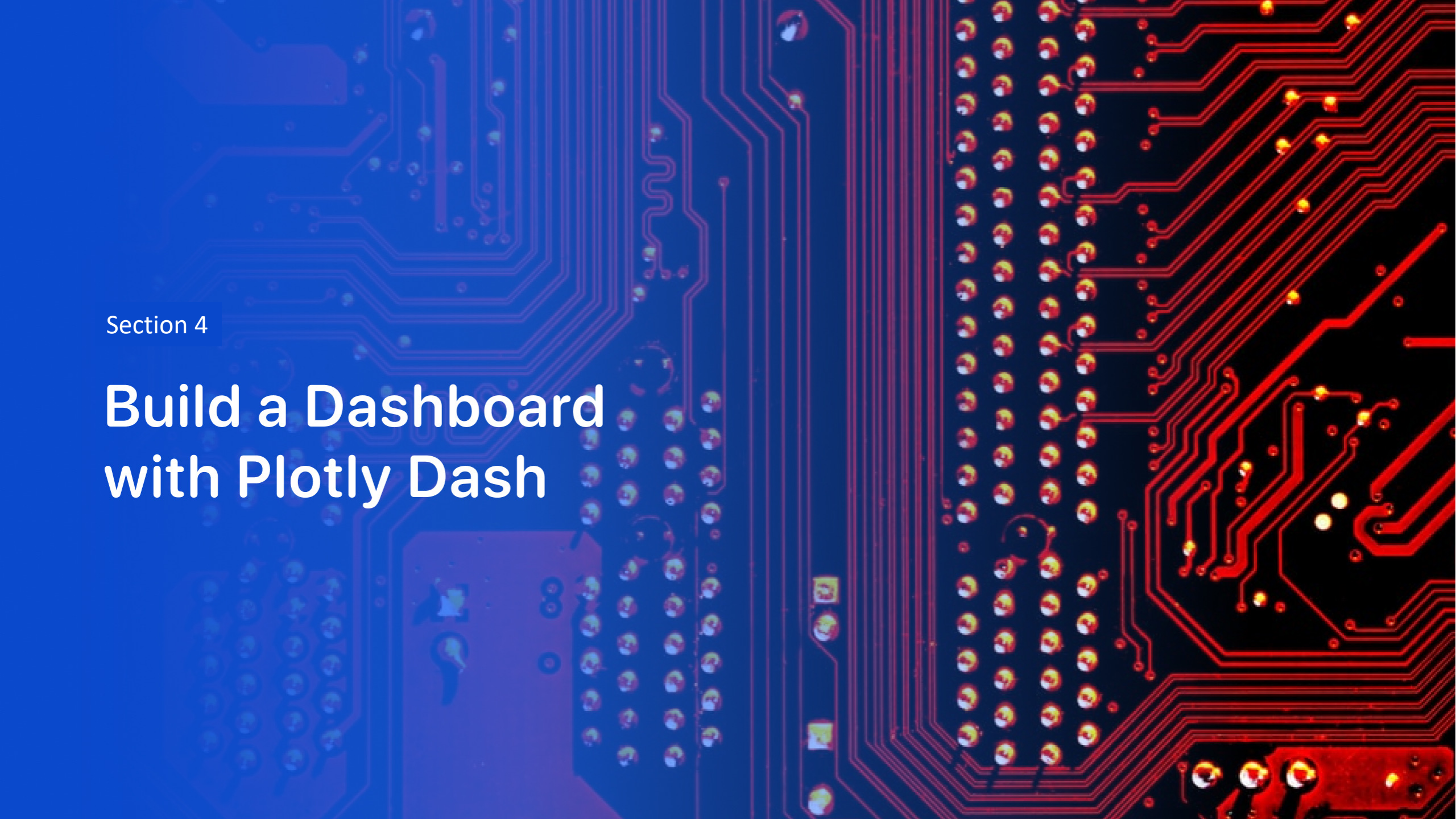
- Objective: Visualize the outcomes (success or failure) of launches at each site.
- Findings: Each site is marked with green markers for successful launches and red markers for failed launches.
- Observations: This visualization helps in identifying sites with higher success rates. For instance, if a site predominantly has green markers, it indicates a higher success rate compared to a site with more red markers.



Calculating Distances to Proximities

- Objective: Measure the distances between launch sites and their nearest significant features (e.g., coastlines, highways, railways).
- Findings: The distance between CCAFS SLC-40 and its closest coastline is approximately 0.51 km.
- Observations: Proximity to coastlines is crucial for safety and logistical reasons. The short distance to the coastline at CCAFS SLC-40 ensures quick transportation and minimal risk to populated areas.

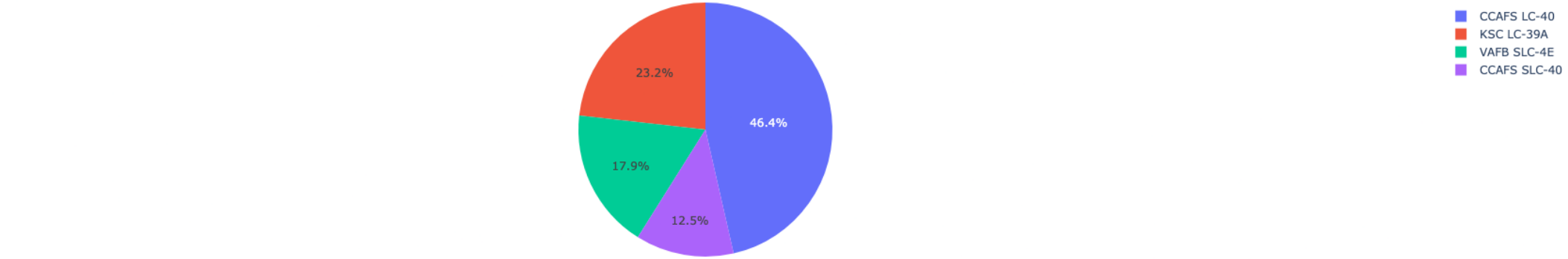



Section 4
Build a Dashboard with Plotly Dash

Success vs Failed Launches for All Sites

Indicates the data presented is a comparison of successful and failed launches across different launch sites.

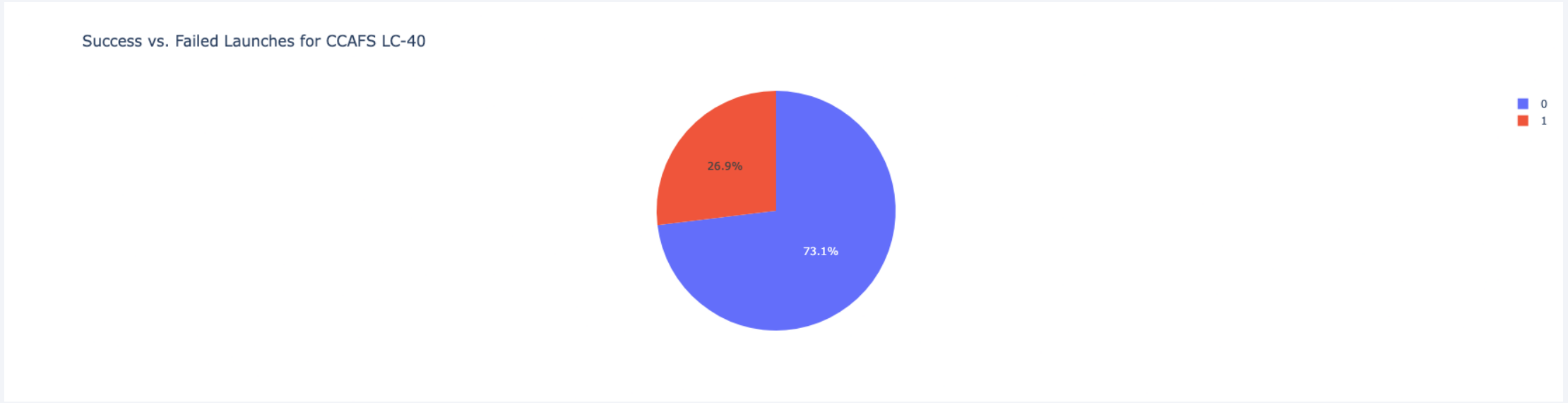
- CCAFS LC-40 has the highest number of launches, making up 46.4% of the total.
- KSC LC-39A follows with 23.2%.
- VAFB SLC-4E accounts for 17.9%.
- CCAFS SLC-40 has the fewest launches at 12.5%.



Success vs. Failed Launches for CCAFS LC-40

Focuses specifically on the success and failure rates of launches from the CCAFS LC-40 site.

- Most launches from CCAFS LC-40 are successful, with a success rate of 73.1%.
- The failure rate is 26.9%, indicating that just over a quarter of the launches failed.



Correlation between Payload and Success for all Sites

Examines the relationship between payload mass and launch success across all sites.

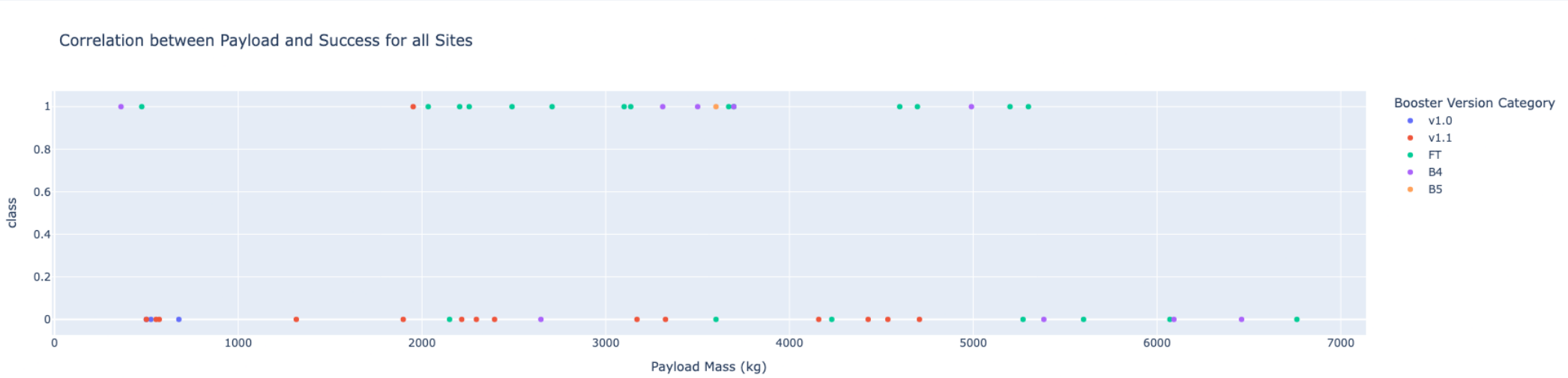
Success Rates by Payload Range:

- Most successful launches (class = 1) cluster in the 0-4000 kg payload range.
- There are fewer successful launches for payloads above 4000 kg.

Failure Rates by Payload Range:

- Launch failures (class = 0) are more spread out but tend to occur with payloads both below 2000 kg and above 4000 kg.

- Booster Version Success:
  - FT and B5 versions appear to have higher success rates, especially within the 1000-4000 kg range.
  - v1.0 and v1.1 versions show more scattered successes and failures.
  - B4 has limited data points but shows success with payloads around 5000 kg.

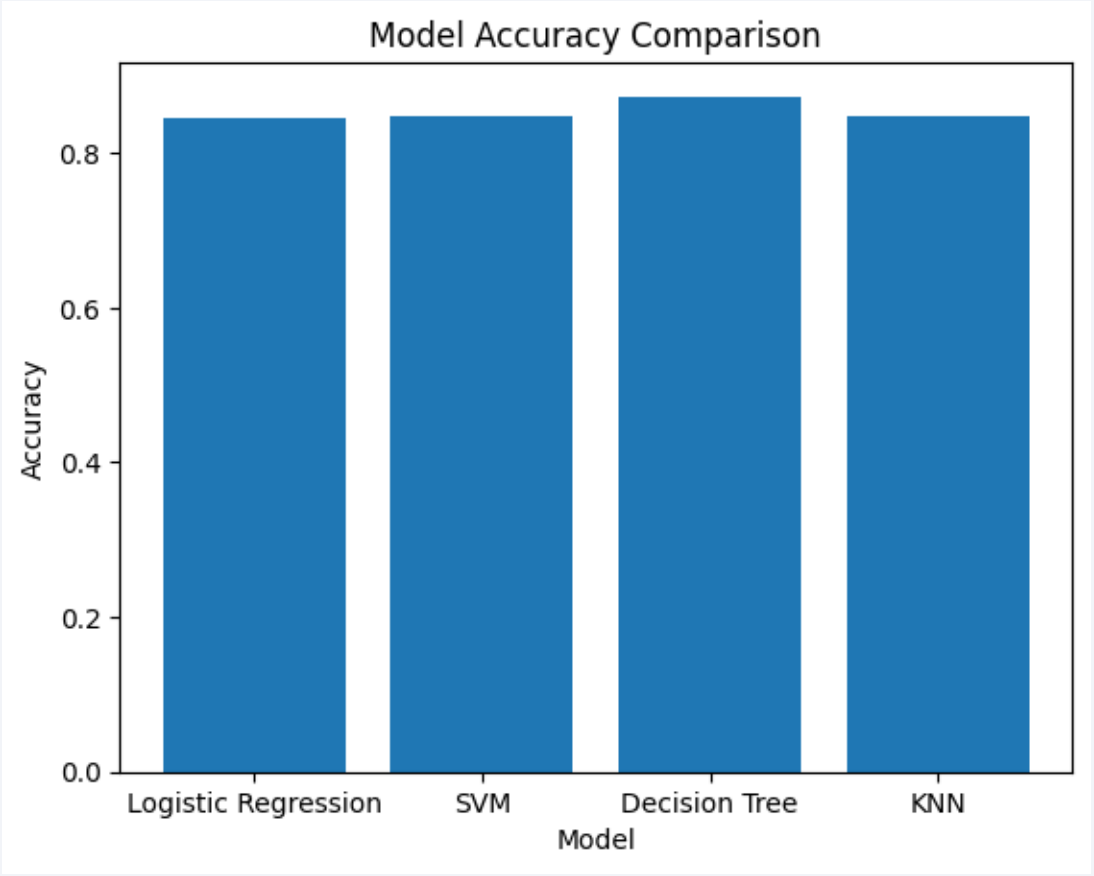


Section 5

Predictive Analysis (Classification)

Classification Accuracy

- The model has the highest classification accuracy is Decision Tree

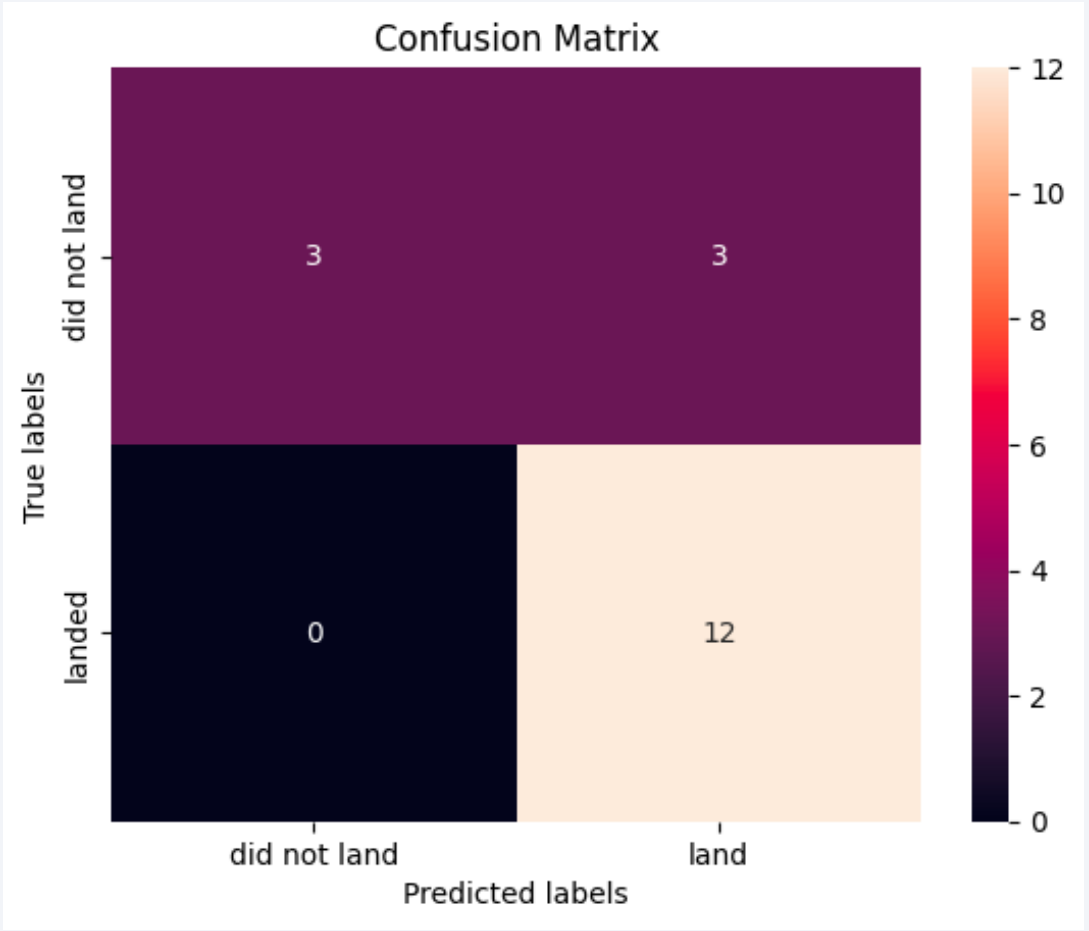


Confusion Matrix

A confusion matrix is a table used to evaluate the performance of a classification model. Each row of the matrix represents the instances in an actual class, while each column represents the instances in a predicted class.

Specific Values:

- Top-left (3): These are the true negatives (TN). The model correctly predicted that the rocket did not land for 3 instances.
- Top-right (3): These are the false positives (FP). The model incorrectly predicted that the rocket landed for 3 instances that actually did not land.
- Bottom-left (0): These are the false negatives (FN). The model incorrectly predicted that the rocket did not land for 0 instances that actually landed.
- Bottom-right (12): These are the true positives (TP). The model correctly predicted that the rocket landed for 12 instances.



Conclusions

- Most successful launches, indicated by a classification of 1, tend to fall within the payload range of 0-4000 kg. Conversely, there are fewer successful launches for payloads exceeding 4000 kg
- Among various classification models evaluated, including Logistic Regression, SVM, and KNN, the Decision Tree model achieved the highest classification accuracy for predicting the success of rocket landings
- The FT and B5 booster versions demonstrate higher success rates, particularly for payloads within the 1000-4000 kg range. In contrast, the v1.0 and v1.1 versions show more scattered results, with successes and failures distributed more evenly
- The analysis found that the distance to the nearest coastline is a significant factor, with the CCAFS SLC-40 launch site being only approximately 0.51 km from the coastline. This proximity is vital for both safety and logistical reasons
- The best model accurately predicted 12 successful landings out of 12 actual successful landings, indicating strong performance in identifying when a rocket will land.

Appendix

- Python code for the dashboard with Plotly

```python
import pandas as pd
import dash
import dash_html_components as html
import dash_core_components as dcc
from dash.dependencies import Input, Output
import plotly.express as px

# Read the airline data into pandas dataframe
spacex_df = pd.read_csv("spacex_launch_dash.csv")
max_payload = spacex_df["Payload Mass (kg)"].max()
min_payload = spacex_df["Payload Mass (kg)"].min()

# Create a dash application
app = dash.Dash(__name__)

# Create an app layout
app.layout = html.Div(
    children=[
        html.H1(
            "SpaceX Launch Records Dashboard",
            style={"textAlign": "center", "color": "#503D36", "font-size": 40},
        ),
        # TASK 1: Add a dropdown list to enable Launch Site selection
        # The default select value is for ALL sites
        html.Div(
            [
                html.Label("Select a Launch Site: "),
                dcc.Dropdown(
                    id="site-dropdown",
                    options=[
                        {"label": "All Sites", "value": "ALL"},
                        {"label": "CCAFS LC-40", "value": "CCAFS LC-40"},
                        {"label": "VAFB SLC-4E", "value": "VAFB SLC-4E"},
                        {"label": "KSC LC-39A", "value": "KSC LC-39A"},
                        {"label": "CCAFS SLC-40", "value": "CCAFS SLC-40"},
                    ],
                    value="ALL",
                    placeholder="Select a Launch Site here",
                    searchable=True,
                ),
            ]
        ),
        html.Br(),
        # TASK 2: Add a pie chart to show the total successful launches count for all sites
        # If a specific launch site was selected, show the Success vs. Failed counts for the site
        html.Div(dcc.Graph(id="success-pie-chart")),
        html.Br(),
        html.P("Payload range (Kg):"),
        # TASK 3: Add a slider to select payload range
        dcc.RangeSlider(
            id="payload-slider",
            min=0,
            max=10000,
            step=1000,
            marks={
                0: "0",
                100: "100",
            },
            value=[min_payload, max_payload],
        ),
        # TASK 4: Add a scatter chart to show the correlation between payload and launch success
        html.Div(dcc.Graph(id="success-payload-scatter-chart")),
    ]
)
```

```python
@app.callback(
    Output(component_id="success-pie-chart", component_property="figure"),
    Input(component_id="site-dropdown", component_property="value"),
)
def get_pie_chart(entered_site):
    filtered_df = spacex_df
    if entered_site == "ALL":
        # return the outcomes piechart for all sites
        data = (
            filtered_df.groupby(["Launch Site"]).size().reset_index(name="class count")
        )
        data["pie chart names"] = data["Launch Site"]
        title = "Success vs. Failed Launches for All Sites"
        fig = px.pie(data, values="class count", names="pie chart names", title=title)
        return fig
    else:
        # return the outcomes piechart for a selected site
        data = filtered_df[filtered_df["Launch Site"] == entered_site]
        data = data.groupby(["class"]).size().reset_index(name="class count")
        data["pie chart names"] = data["class"]
        title = f"Success vs. Failed Launches for {entered_site}"
        fig = px.pie(data, values="class count", names="pie chart names", title=title)
        return fig


# TASK 4:
# Add a callback function for `site-dropdown` and `payload-slider` as inputs, `success-payload-scatter-chart` as output
@app.callback(
    Output(component_id="success-payload-scatter-chart", component_property="figure"),
    [
        Input(component_id="site-dropdown", component_property="value"),
        Input(component_id="payload-slider", component_property="value"),
    ],
)
def get_scatter_chart(entered_site, payload_range):
    low, high = payload_range
    mask = (spacex_df["Payload Mass (kg)"] > low) & (
        spacex_df["Payload Mass (kg)"] < high
    )
    filtered_df = spacex_df[mask]
    if entered_site == "ALL":
        fig = px.scatter(
            filtered_df,
            x="Payload Mass (kg)",
            y="class",
            color="Booster Version Category",
            title="Correlation between Payload and Success for all Sites",
        )
        return fig
    else:
        # return the outcomes scatter chart for a selected site
        data = filtered_df[filtered_df["Launch Site"] == entered_site]
        fig = px.scatter(
            data,
            x="Payload Mass (kg)",
            y="class",
            color="Booster Version Category",
            title=f"Correlation between Payload and Success for {entered_site}",
        )
        return fig


# Run the app
if __name__ == "__main__":
    app.run_server(port=8030)
```

Thank you!