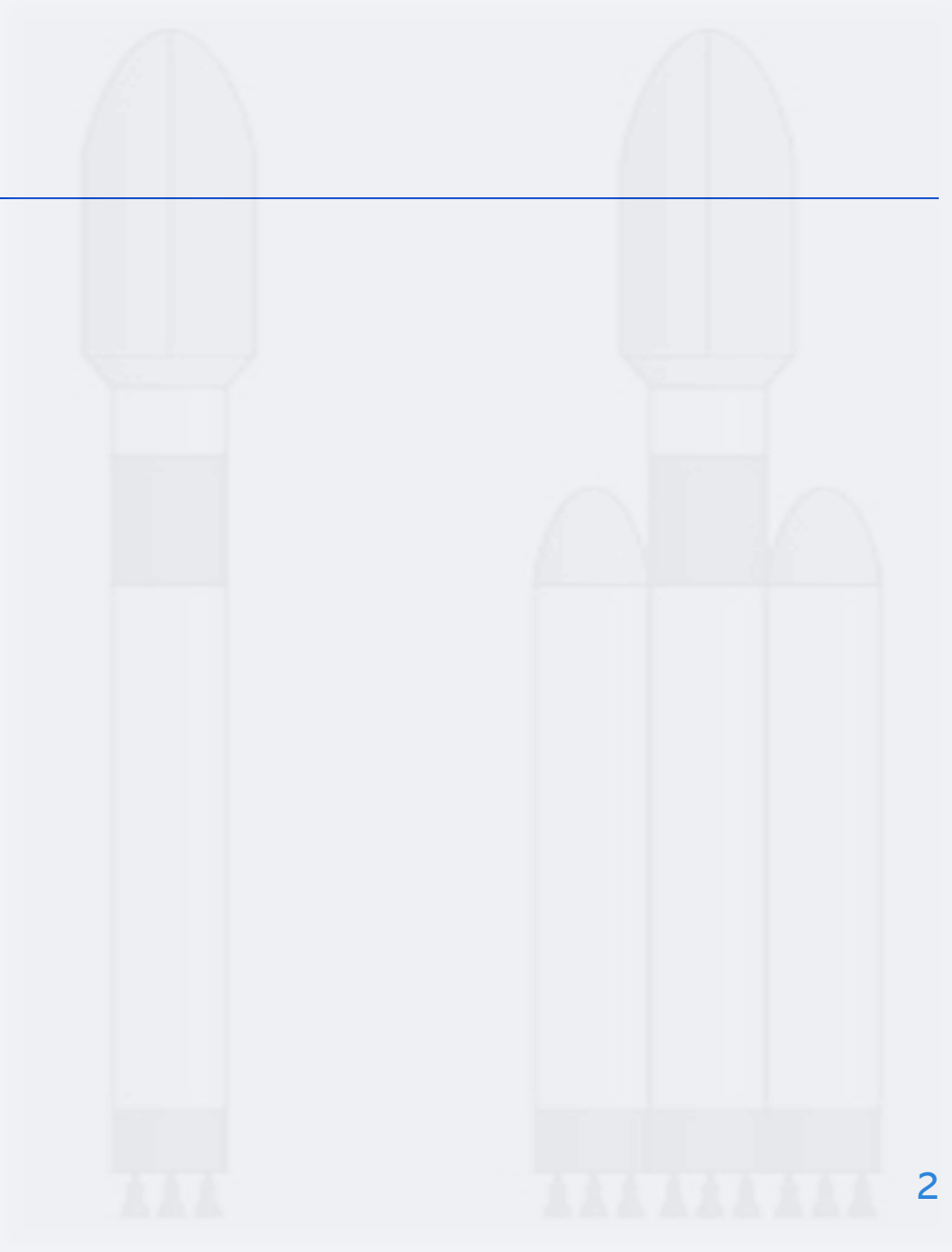
IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Belén López-Montenegro Ordoñez
01/Octubre/2024

Outline

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



Executive Summary

❖ **Summary of methodologies**

* Data Collection
* Data Wrangling
* Exploratory Data Analysis (EDA)
* Interactive Visual Analytics
* Predictive Analysis

❖ **Summary of all results**

* Exploratory Data Analysis
* Interactive Visualizations
* Predictive Modeling

Introduction

❖Project Background and Context:

- **SpaceX** is the first private company to return a spacecraft from low-Earth orbit and has revolutionized the space industry by significantly reducing costs through the reuse of the **Falcon 9** first stage.

- Each Falcon 9 launch costs **62 million dollars**, while other companies may charge up to **165 million dollars**. The savings come from SpaceX's ability to reuse the first stage, which depends on whether it successfully lands.

- **Project Objective**: If we can predict whether the Falcon 9's first stage will successfully land, we can estimate the launch cost and potentially allow other companies to compete with SpaceX on future missions.

Introduction

❖Problems You Want to Find Answers:

- **What factors determine the success of the first stage landing?**
  - Identify the most important variables, such as **payload mass**, **flight number**, **orbit type**, and **launch site**, that influence the likelihood of a successful landing.

- **How does component reuse impact the success rate of launches?**
  - Assess whether the number of times a rocket is reused significantly affects the likelihood of a successful landing.

- **Which predictive model is the most effective for forecasting successful landings?**
  - Compare various machine learning algorithms (Logistic Regression, SVM, Decision Trees, KNN) to identify the best method for predicting landing success.

- **Can future launches be optimized based on historical data?**
  - Use the analysis results to optimize launch planning and further reduce SpaceX's costs or allow competing companies to improve their launch offerings.

Section 1

Methodology

Methodology

Executive Summary

- **Data collection methodology:**
  - **Data Collection**: Data was collected using the SpaceX REST API and web scraping techniques from Wikipedia. The API provided detailed information on past SpaceX launches, while web scraping was used to gather additional launch records.
    - API URL: https://api.spacexdata.com/v4/launches/past
    - Web scraping using Python's BeautifulSoup.
- **Perform data wrangling**
  - **Cleaning the Data**: The dataset was cleaned by removing irrelevant columns, filtering for Falcon 9 launches, and handling missing values (e.g., filling in missing payload mass values with the mean).
- **Data Processing:**
  - The data was normalized and structured into a Pandas DataFrame to enable efficient manipulation and analysis.

Methodology

Executive Summary

- **Perform exploratory data analysis (EDA) using visualization and SQL**
  - **Visualization**: EDA was conducted using Python libraries like matplotlib and seaborn, exploring relationships between launch success and key variables (e.g., launch site, payload mass).
  - **SQL Queries**: SQL was used to query and analyze specific patterns, such as the success rates by orbit type and payload mass carried by each booster.

- **Perform interactive visual analytics using Folium and Plotly Dash**
  - **Folium**: Created an interactive map to visualize SpaceX launch sites and their proximities.
  - **Plotly Dash**: Developed an interactive dashboard with plots showing launch success rates, payload vs. launch outcome, and success per launch site.

Methodology

Executive Summary

- **Perform predictive analysis using classification models**

Various classification models (e.g., Logistic Regression, Support Vector Machines, Decision Trees) were used to predict whether the Falcon 9's first stage would successfully land.

- **Model tuning**: Hyperparameters were tuned using grid search to optimize model performance.
- **Evaluation**: Models were evaluated using accuracy, precision, recall, and confusion matrices to select the best performing model.

Data Collection

- **Data Collection – SpaceX REST API**
  - **Objective**: Collect data on past SpaceX launches using their public REST API.
  - **API URL**: https://api.spacexdata.com/v4/launches/past
  - **Method**: An HTTP GET request was used to retrieve launch data.
  - **Response**: The API returned a **JSON** response containing detailed information about each launch, including:

Rocket used Payload Launch date Launch site Landing outcome

Data Collection – SpaceX API

- **Flowchart Overview:**

- **Step 1**: Request to SpaceX REST API
- **Step 2**: JSON Response
- **Step 3**: Convert the JSON response into a **Pandas DataFrame**
- **Step 4**: Filter and clean data (Falcon 9 only)
- **Step 5**: Handle missing values
- **Step 6**: Store cleaned dataset for further analysis
- **GitHub URL**: Data collection

Step 1: Make a **GET** request to the SpaceX API

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```
✓ 0.0s Python

▷ Initialize Reactive Jupyter | Sync all Stale code
```
response = requests.get(spacex_url)
```
✓ 0.4s Python

Step 2: Convert the JSON response into a Pandas DataFrame

```
# Use json_normalize meethod to convert the json result into a dataframe
data=pd.json_normalize(response.json())
```
✓ 0.0s Python

Step 3: Filter the dataset to include only Falcon 9 launches.

▷ Initialize Reactive Jupyter | Sync all Stale code
```
# Hint data['BoosterVersion']!='Falcon 1'
# Filtrar la columna BoosterVersion para mantener solo los lanzamientos de Falcon 9
data_falcon9 = data_dict[data_dict['BoosterVersion'].str.contains('Falcon 9', regex=False)]

```
✓ 0.0s Python

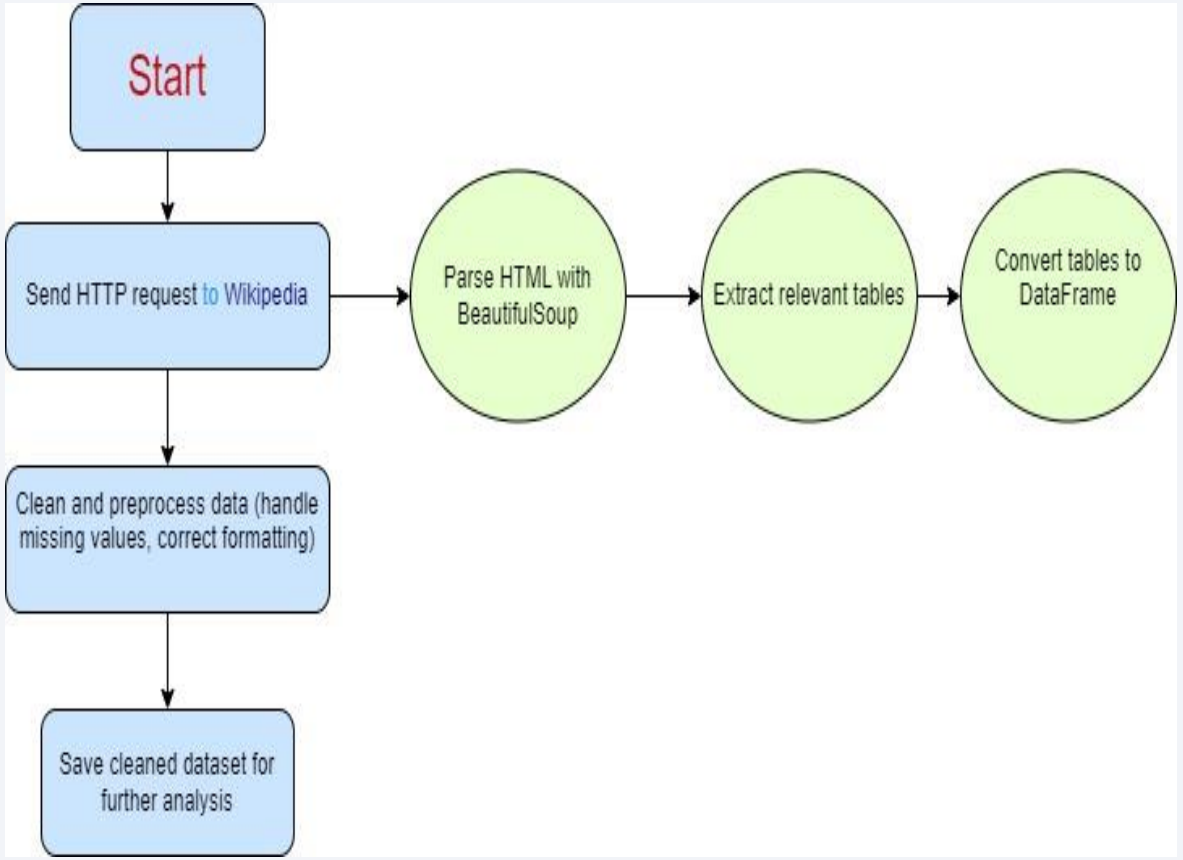
Step 4: Handle missing values, such as filling in missing payload mass values with the mean.

```
# Reemplazar los valores NaN en la columna 'PayloadMass' por la media calculada de manera segura
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].fillna(mean_payload_mass)
```

Data Collection - Scraping

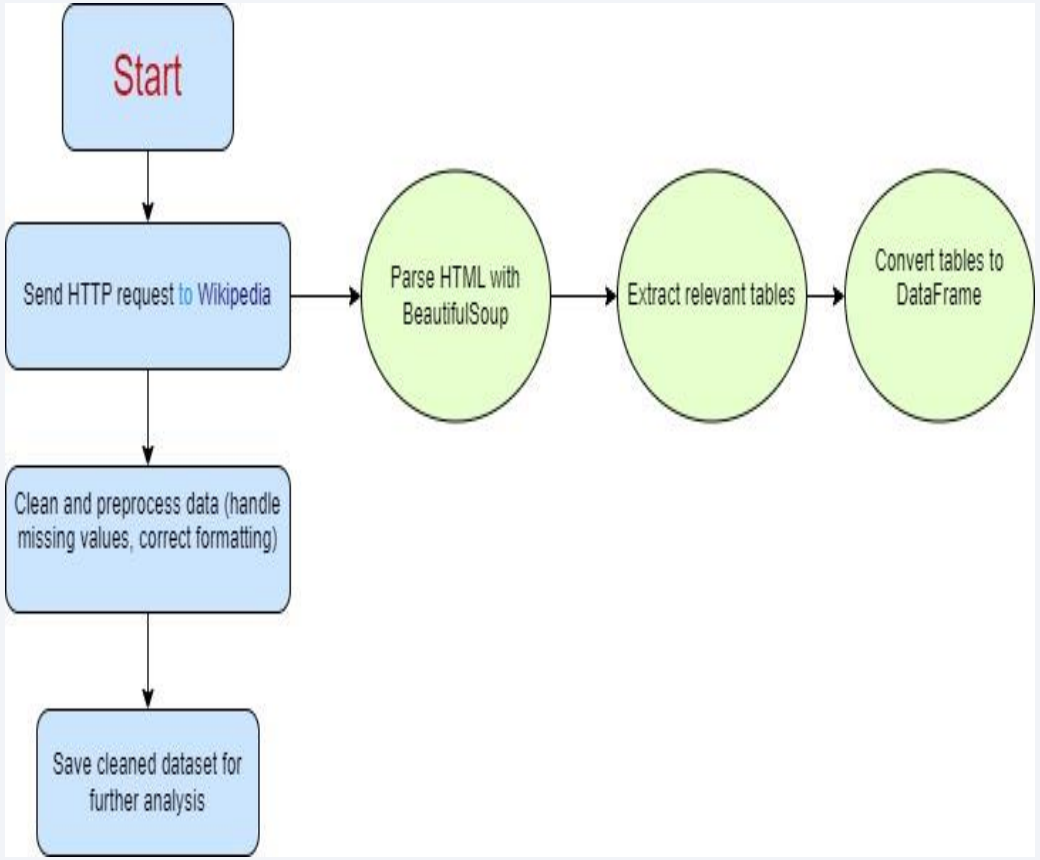
First Box: Present your web scraping process using key phrases and flowcharts

- **Objective**: Extract launch data from Wikipedia to complement the data retrieved via the SpaceX API.
- **Methodology**:
  - **Library Used**: BeautifulSoup and requests in Python.
  - **Target URL**: Web pages containing historical Falcon 9 launch data in table format.
- **Process**:
  - **Step 1**: Send an HTTP request to the Wikipedia page containing the Falcon 9 launch records.
  - **Step 2**: Parse the HTML content using BeautifulSoup to locate the relevant HTML tables.
  - **Step 3**: Extract the tables and convert them into a Pandas DataFrame.
  - **Step 4**: Clean and preprocess the data, handling any missing values and inconsistent formatting.
- **Final Data**: The cleaned data was merged with the API dataset for a more comprehensive analysis of Falcon 9 launches.



Data Collection - Scraping

First Box: Present your web scraping process using key phrases and flowcharts




```
# use requests.get() method with the provided static_url
# assign the response to a object
# URL de la página estática de Wikipedia sobre los lanzamientos de Falcon 9 y Falcon Heavy
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"

# Realizar una solicitud HTTP GET para obtener el contenido HTML de la página
response = requests.get(static_url)
```

```
soup = BeautifulSoup(response.content, 'html.parser')
```

```
html_tables = soup.find_all('table', {"class": "wikitable"})
```

```
df = df=pd.DataFrame({key: pd.Series(value, dtype='object') for key, value in
launch_dict.items()})
```

GITHUB: webscraping.

Data Wrangling

- **Describe how data were processed**:
  - **Data Cleaning**: Missing values were checked and handled to ensure data completeness. For example, no significant percentage of null values were detected in the dataset.
  - **Label Creation**: The Outcome column, which contained different descriptions of landing results, was transformed into a binary classification label (Class). Successful landings were labeled as 1 and unsuccessful landings as 0.
    - Example outcomes: True Ocean → success (1), False Ocean → failure (0).
  - **Feature Selection**: Key features such as LaunchSite, Orbit, and Outcome were analyzed to create insights and labels for predictive modeling.

Data wrangling

Data Wrangling

Data Wrangling Process (Key Phrases and Flowchart):

- **Process Flow:**
  - **Step 1**: Load the dataset and check for missing values.

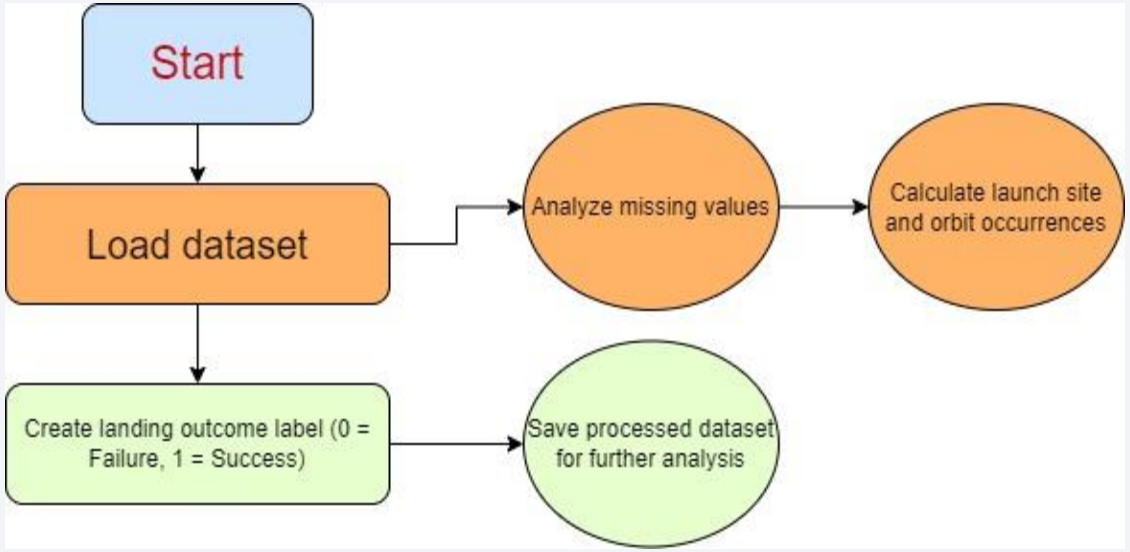
```
df.isnull().sum()/len(df)*100
```

  - **Step 2**: Analyze and count the occurrences of key features like LaunchSite and Orbit.

```
launch_site_counts=df['LaunchSite'].value_counts()
```

  - **Step 3**: Create a binary Class label from the Outcome column to represent successful (1) and unsuccessful (0) landings.

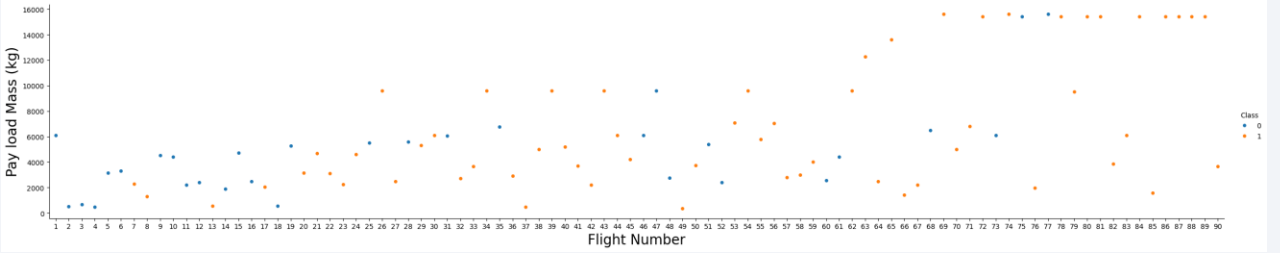
```
landing_outcomes = df['Outcome'].value_counts()
print(landing_outcomes)
```

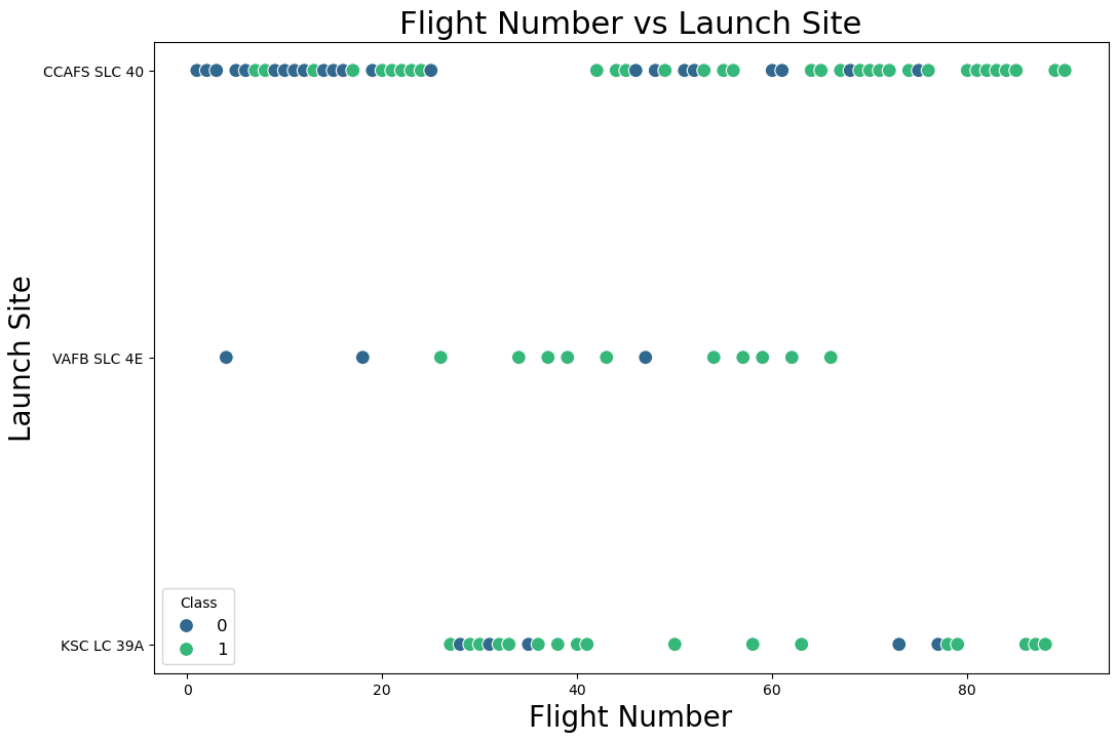


Data wrangling

EDA with Data Visualization

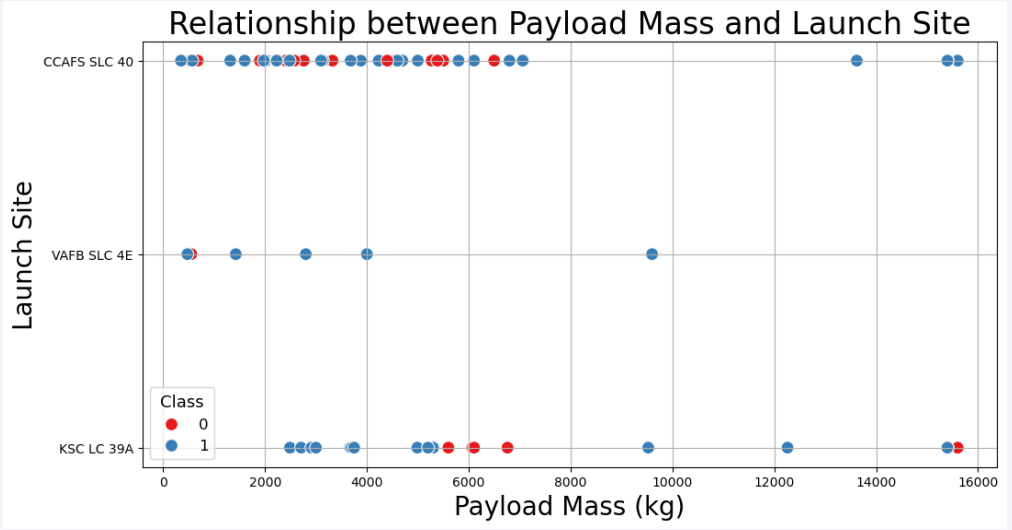
- **Summary of Plotted Charts**:
- **Flight Number vs. Payload Mass**:
  - **Chart Type**: Scatter Plot
  - **Purpose**: To visualize the relationship between the number of flights and payload mass. This chart helps to identify trends, showing that as the flight number increases, the likelihood of a successful landing also increases, while higher payload masses generally correlate with lower success rates.
- **Flight Number vs. Launch Site**:
  - **Chart Type**: Scatter Plot
  - **Purpose**: To assess the frequency of launches from different sites over time. This visualization reveals preferences for certain launch sites and helps understand how launch frequency has evolved.



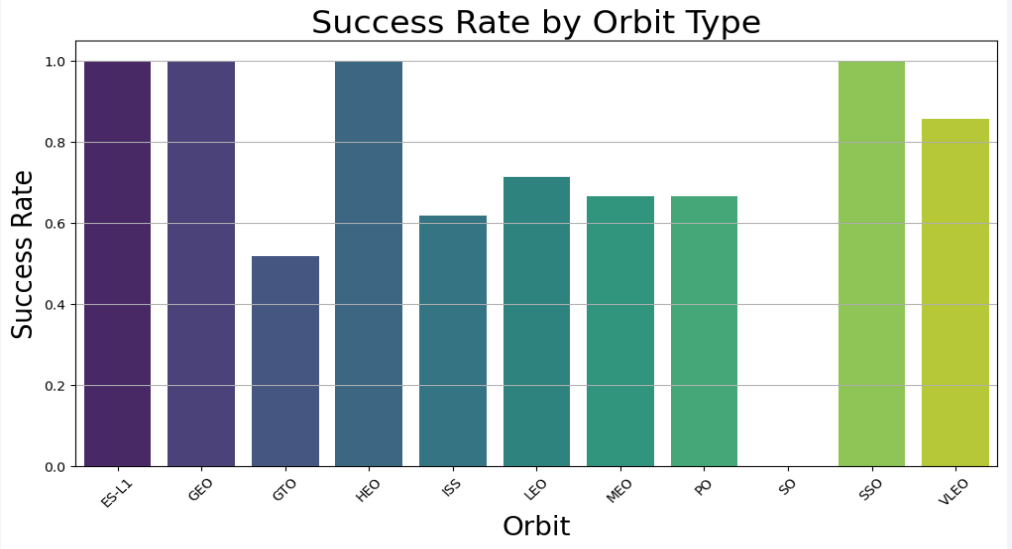


EDA with Data Visualization

- **Payload Mass vs. Launch Site**:
  - **Chart Type**: Scatter Plot
  - **Purpose**: To explore the correlation between payload mass and the launch site. It shows that certain sites (e.g., VAFB SLC-4E) do not host launches for heavy payloads (>10,000 kg).



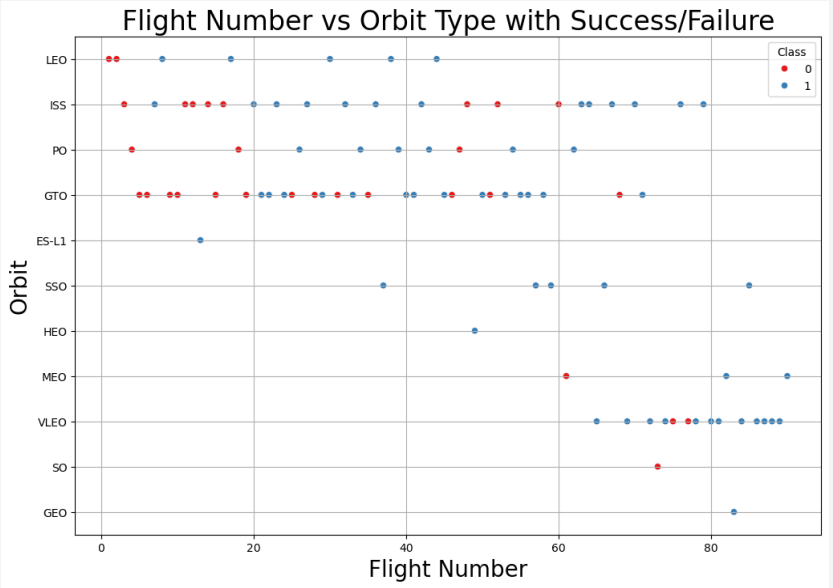
- **Success Rate by Orbit Type**:
  - **Chart Type**: Bar Plot
  - **Purpose**: To display the average success rate of launches based on the orbit type. This chart is crucial for determining which orbits have historically been more successful, guiding future mission planning.

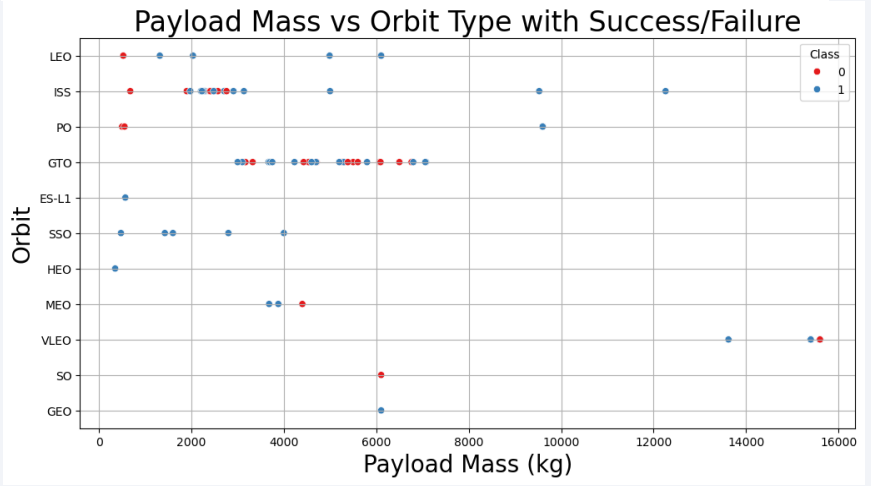


EDA with Data Visualization

- **Flight Number vs. Orbit Type**:
  - **Chart Type**: Scatter Plot
  - **Purpose**: To examine the relationship between flight number and orbit type. It highlights that in LEO orbits, success rates tend to improve with higher flight numbers, while GTO orbits show no clear relationship.

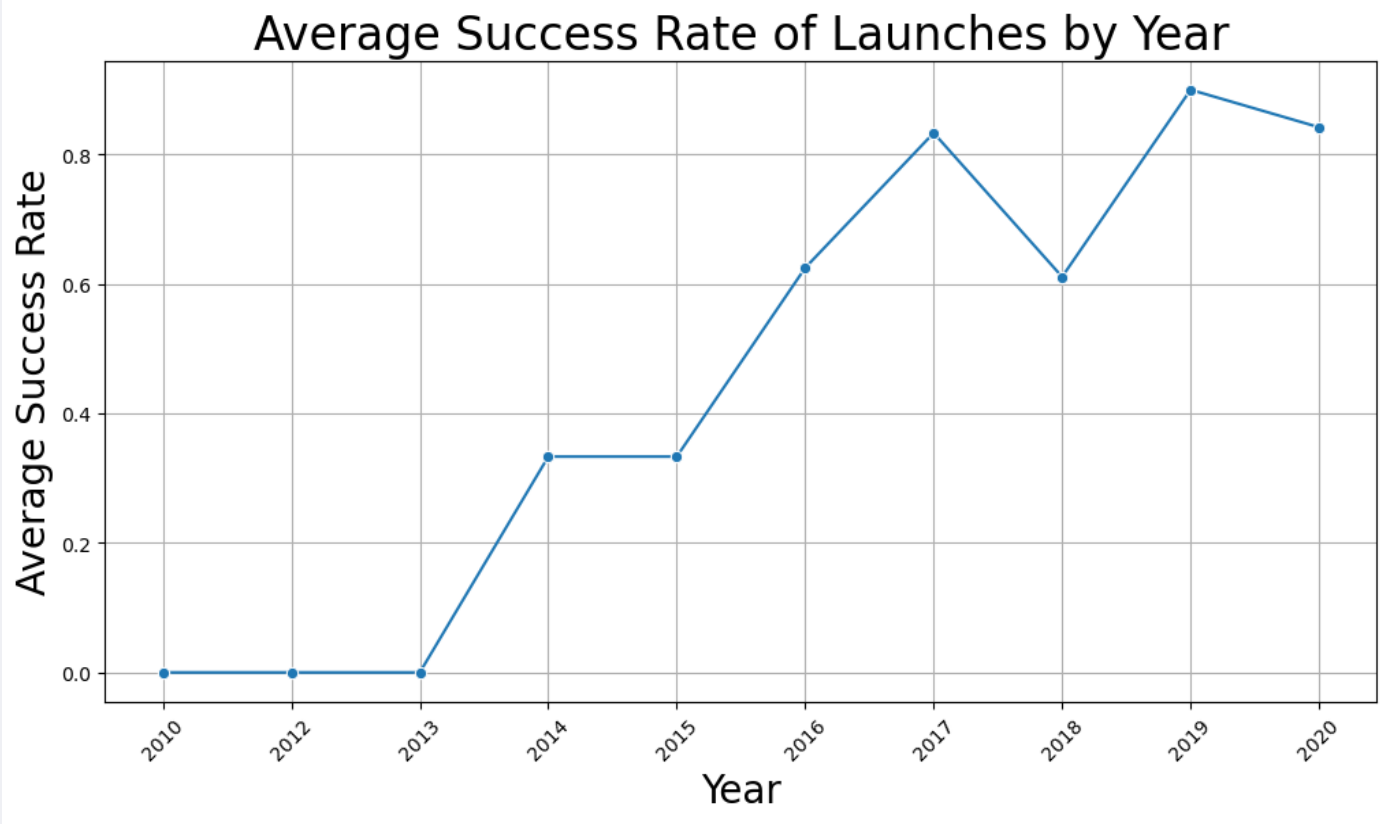
- **Payload Mass vs. Orbit Type**:
  - **Chart Type**: Scatter Plot
  - **Purpose**: To assess the relationship between payload mass and orbit type. It indicates that successful landings are more frequent in orbits like Polar and LEO with heavier payloads, while GTO shows mixed results.





EDA with Data Visualization

- **Yearly Launch Success Trend**:
  - **Chart Type**: Line Plot
  - **Purpose**: To visualize the trend of launch success rates over the years. This chart shows the overall increase in success rates, particularly from 2013 onwards, and stabilizes around 2014.



- [EDA with Data Visualization Notebook]

EDA with SQL

- **Summary of SQL Queries Performed:**
  - **Unique Launch Sites:**
    - **Query:**
      ```
      cur.execute('''SELECT DISTINCT "Launch_Site" FROM SPACEXTABLE;''')
      ```
    - **Purpose:** To identify all unique launch sites used in SpaceX missions.
  - **Launch Sites Starting with 'CCA':**
    - **Query:**
      ```
      cur.execute('''SELECT * FROM SPACEXTABLE WHERE "Launch_Site" LIKE 'CCA%' LIMIT 5;''')
      ```
    - **Purpose:** To retrieve records of launches from sites that start with 'CCA', analyzing site usage.
  - **Total Payload Mass for NASA (CRS):**
    - **Query:**
      ```
      SELECT SUM("PAYLOAD_MASS__KG_")
      FROM SPACEXTABLE
      WHERE "Customer" LIKE '%NASA (CRS)%';
      ```
    - **Purpose:** To calculate the total payload mass carried by SpaceX missions for NASA's CRS program.

EDA with SQL

- **Average Payload Mass for F9 v1.1**:
  - **Query**:
```
SELECT AVG("PAYLOAD_MASS__KG_")
FROM SPACEXTABLE
WHERE "Booster_Version" = 'F9 v1.1';
```
  - **Purpose**: To determine the average payload mass carried by the Falcon 9 version 1.1 boosters.
- **First Successful Landing Date on Ground Pad**:
  - **Query**:
```
SELECT MIN(Date)
FROM SPACEXTABLE
WHERE "Landing_Outcome" = 'Success (ground pad)';
```
  - **Purpose**: To find the date of the first successful landing on a ground pad.
- **Boosters with Success on Drone Ships**:
  - **Query**:
```
SELECT DISTINCT "Booster_Version"
FROM SPACEXTABLE
WHERE "Landing_Outcome" IN ('Success (drone ship)', 'Success')
AND "PAYLOAD_MASS__KG_" > 4000
AND "PAYLOAD_MASS__KG_" < 6000;
```
  - **Purpose**: To identify which booster versions successfully landed on drone ships while carrying payloads between 4000 kg and 6000 kg.

EDA with SQL

- **Mission Outcome Counts**:
  - **Query**:

```
cur.execute('''
    SELECT "Mission_Outcome", COUNT(*) AS Total
    FROM SPACEXTABLE
    GROUP BY "Mission_Outcome";
''')
```

  - **Purpose**: To summarize the total number of successful and failed mission outcomes.

- **Boosters with Maximum Payload Mass**:
  - **Query**:

```
cur.execute('''
    SELECT DISTINCT "Booster_Version"
    FROM SPACEXTABLE
    WHERE "PAYLOAD_MASS__KG_" = (
        SELECT MAX("PAYLOAD_MASS__KG_")
        FROM SPACEXTABLE
    );
''')
```

  - **Purpose**: To find the booster versions that have carried the maximum payload mass.

EDA with SQL

- **Monthly Failure Outcomes in 2015**:
  - **Query**:

```
SELECT
    CASE
        WHEN substr("Date", 6, 2) = '01' THEN 'January'
        WHEN substr("Date", 6, 2) = '02' THEN 'February'
        WHEN substr("Date", 6, 2) = '03' THEN 'March'
        WHEN substr("Date", 6, 2) = '04' THEN 'April'
        WHEN substr("Date", 6, 2) = '05' THEN 'May'
        WHEN substr("Date", 6, 2) = '06' THEN 'June'
        WHEN substr("Date", 6, 2) = '07' THEN 'July'
        WHEN substr("Date", 6, 2) = '08' THEN 'August'
        WHEN substr("Date", 6, 2) = '09' THEN 'September'
        WHEN substr("Date", 6, 2) = '10' THEN 'October'
        WHEN substr("Date", 6, 2) = '11' THEN 'November'
        WHEN substr("Date", 6, 2) = '12' THEN 'December'
    END AS "Month",
    "Landing_Outcome",
    "Booster_Version",
    "Launch_Site"
FROM SPACEXTABLE
WHERE "Landing_Outcome" LIKE 'Failure%'
  AND substr("Date", 0, 5) = '2015';
```

  - **Purpose**: To list records of failure outcomes by month for the year 2015.

EDA with SQL Notebook

EDA with SQL

- **Landing Outcome Count from 2010-06-04 to 2017-03-20:**
  - **Query:**

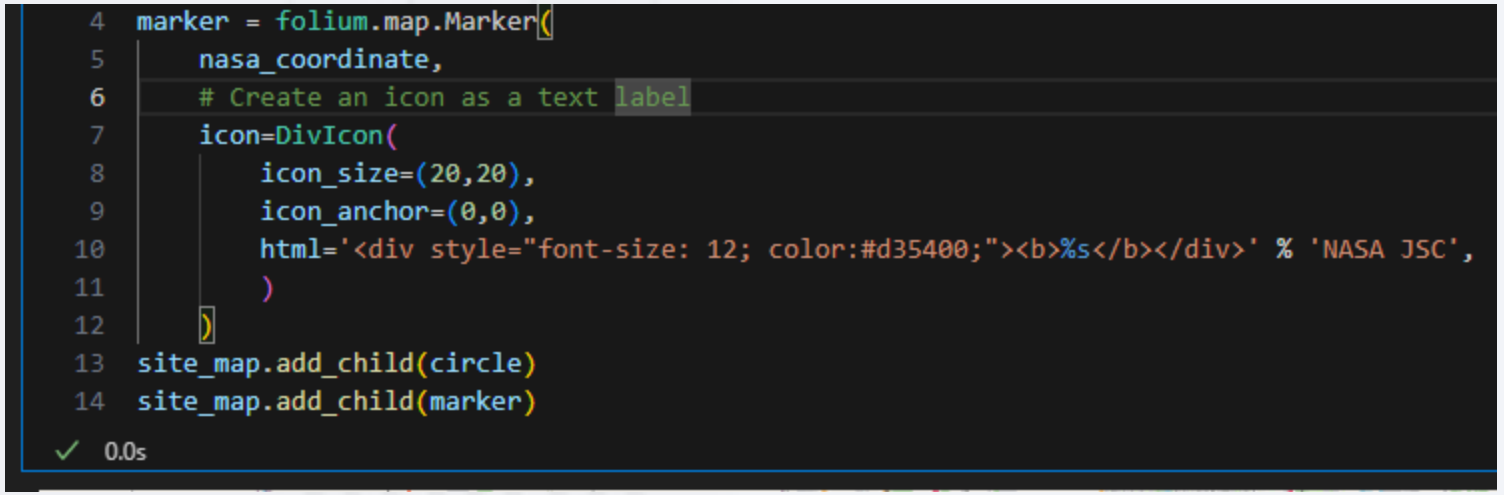
```
# Ejecutar la consulta SQL para contar los resultados de aterrizaje en el rango de fechas especificado
cur.execute('''
    SELECT
        CASE
            WHEN "Landing_Outcome" IN ('Success (ground pad)', 'Success (drone ship)', 'Success') THEN 'Éxito (plataforma terrestre)'
            WHEN "Landing_Outcome" LIKE 'Failure%' THEN 'Fracaso (nave no tripulada)'
            ELSE 'Otros'
        END AS "Resultado",
        COUNT(*) AS "Total"
    FROM SPACEXTABLE
    WHERE "Date" BETWEEN '2010-06-04' AND '2017-03-20'
    GROUP BY "Resultado"
    ORDER BY "Total" DESC;
''')
```

  - **Purpose**: To categorize and count landing outcomes within a specified date range.

EDA with SQL Notebook

Build an Interactive Map with Folium

- **Map Objects Added**:
- **Launch Site Markers**:
  - **Object**: `folium.Marker` with custom icon and popup.
  - **Purpose**: To mark the location of each SpaceX launch site and provide relevant details about the site (e.g., site name). This allows users to easily identify and explore the different launch locations.



Build an Interactive Map with Folium

❖**Launch Site Circles**:

* **Object**: folium.Circle with a radius around each launch site.

```
marker = folium.map.Marker(
    nasa_coordinate,
    # Create an icon as a text label
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % 'NASA JSC',
        )
    )
```

* **Purpose**: To visually highlight the area surrounding each launch site. This gives users an understanding of the spatial extent of each site's proximity.

Build an Interactive Map with Folium

- **Success/Failure Markers**:
  - **Object**: Color-coded markers based on launch outcomes.
  - **Purpose**: To distinguish successful landings (green) from failed landings (red).

```
    # Crear un marcador para el resultado del lanzamiento
    launch_marker = folium.Marker(
        location=launch_coordinate,
        icon=folium.Icon(color=row['marker_color'], icon='rocket'),
        popup=f"Launch Site: {row['Launch Site']}<br>Outcome: {'Success' if row['class'] == 1 else 'Failure'}"
    )

    # Añadir el marcador al cluster de marcadores
    marker_cluster.add_child(launch_marker)

# Añadir el marker_cluster al mapa
site_map.add_child(marker_cluster)
```

Build an Interactive Map with Folium

- **Distance Lines**:
  - **Object**: folium.PolyLine.
  - **Purpose**: To visually represent the distance between launch sites and key geographic features (e.g., coastlines).

```
3  coordinates = [[launch_site_lat, launch_site_lon], [coastline_lat, coastline_lon]]
4  lines = folium.PolyLine(locations=coordinates, weight=2, color='blue')
5
6  # lines=folium.PolyLine(locations=coordinates, weight=1)
7  site_map.add_child(lines)
```

✓ 0.1s

Build an Interactive Map with Folium

- **Mouse Position Tracker**:
  - **Object**: MousePosition.
  - **Purpose**: To track the latitude and longitude of the cursor as it moves across the map.

```
14  site_map.add_child(mouse_position)
15  site_map
```

✓ 0.1s

[Interactive Map with Folium]

Build an Interactive Map with Folium

- **Why These Objects Were Added**: Interactive Map with Folium
  - **Launch Site Markers**: To make it easy for users to identify the launch sites.
  - **Circles**: To emphasize the launch site locations and their surrounding areas.
  - **Success/Failure Markers**: To visualize the outcome of each launch, providing immediate insights into success rates.
  - **Distance Lines**: To show the geographic proximity between launch sites and important features like coastlines.
  - **Mouse Position Tracker**: To help users explore the map interactively and obtain coordinates.

Build a Dashboard with Plotly Dash

- Summarize what plots/graphs and interactions you have added to a dashboard
- Explain why you added those plots and interactions
- Add the GitHub URL of your completed Plotly Dash lab, as an external reference and peer-review purpose

Predictive Analysis (Classification)

- **Model Development Process Summary**:
  - Data Preparation
  - Model Selection
  - Hyperparameter Tuning
  - Model Evaluation
  - Best Performing Model

Predictive Analysis (Classification)

- **Model Development Process Summary**:
  - **Data Preparation**:
    - **Label Creation**: Created a binary target variable indicating whether the Falcon 9 first stage landed successfully.
    - **Feature Selection**: Selected key features including FlightNumber, PayloadMass, Orbit, LaunchSite, and others relevant for prediction.
    - **Standardization**: Standardized the feature values to ensure optimal performance for algorithms sensitive to feature scaling.
  - **Model Selection**:
    - Evaluated several classification algorithms:
      - **Logistic Regression**
      - **Support Vector Machine (SVM)**
      - **Decision Tree Classifier**
      - **K-Nearest Neighbors (KNN)**

Predictive Analysis (Classification)

- **Hyperparameter Tuning**:
  - Used GridSearchCV to find the best hyperparameters for each model, improving performance through cross-validation.
  - Selected hyperparameter grids specific to each model type (e.g., C, penalty for Logistic Regression, kernel, gamma for SVM).
- **Model Evaluation:**
  - Accuracy Measurement: Calculated accuracy on test data for each model to compare performance.
  - Confusion Matrix: Analyzed confusion matrices to assess true positives, false positives, true negatives, and false negatives.

Predictive Analysis (Classification)

- **Best Performing Model:**
  - Compared the accuracy of all models on the test data to identify the best performing model.
  - Logistic Regression and SVM showed similar test accuracy (~83.3%), with Decision Trees having a higher training accuracy but similar test results, indicating potential overfitting.

- Add the GitHub URL of your completed predictive analysis lab, as an external reference and peer-review purpose

Predictive Analysis (Classification)

Flowchart Overview:

- **Model Development Process Flow:**

1.Data Preparation

- Create target variable
- Select features
- Standardize features

2.Model Selection

- Choose algorithms: Logistic Regression, SVM, Decision Tree, KNN

3.Hyperparameter Tuning

- Use GridSearchCV for optimal parameters

4.Model Evaluation

- Calculate accuracy
- Analyze confusion matrices

5.Select Best Model

- Compare accuracy across models
- Choose the best performing model

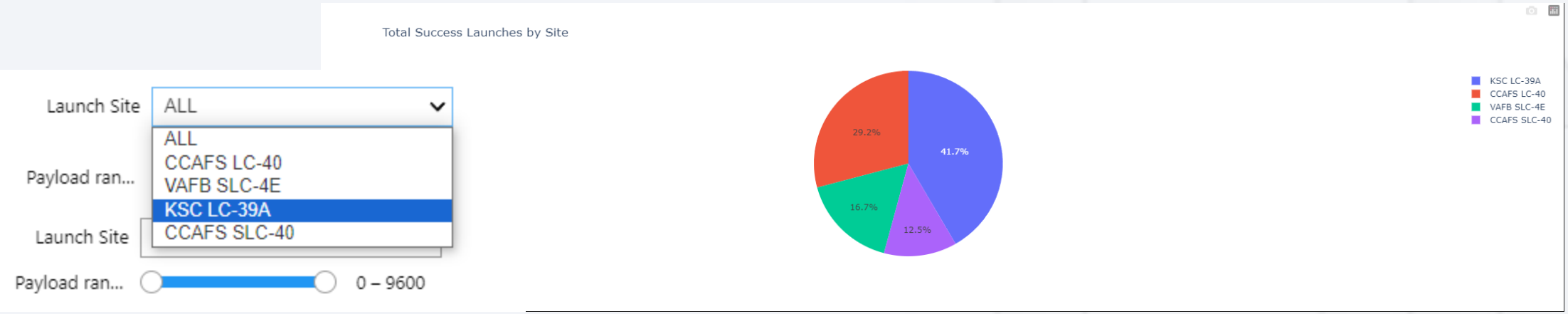
Predictive Analysis Notebook

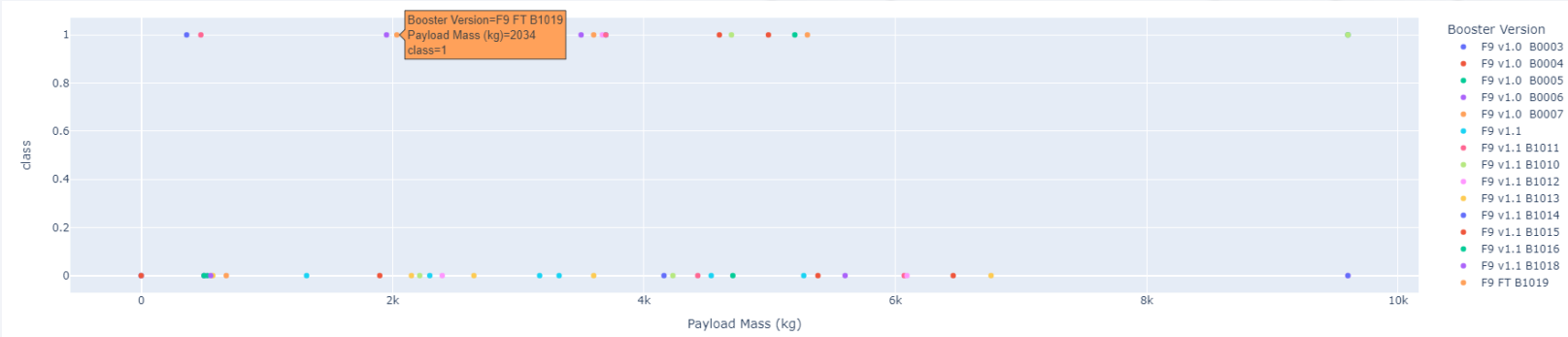
Results

- **Exploratory Data Analysis Results:**
- **Key Findings**:
  - **Flight Number vs. Payload Mass**: Increased flight numbers correlate with higher success rates, while higher payload masses generally lead to lower success.
  - **Launch Site Performance**: Certain sites, such as CCAFS LC-40, exhibit higher launch frequencies and success rates.
  - **Orbit Type Success Rates**: LEO orbits demonstrate significantly higher success rates compared to GTO orbits.

Results

- Interactive Analytics Demo:
  - Screenshots of Dashboard

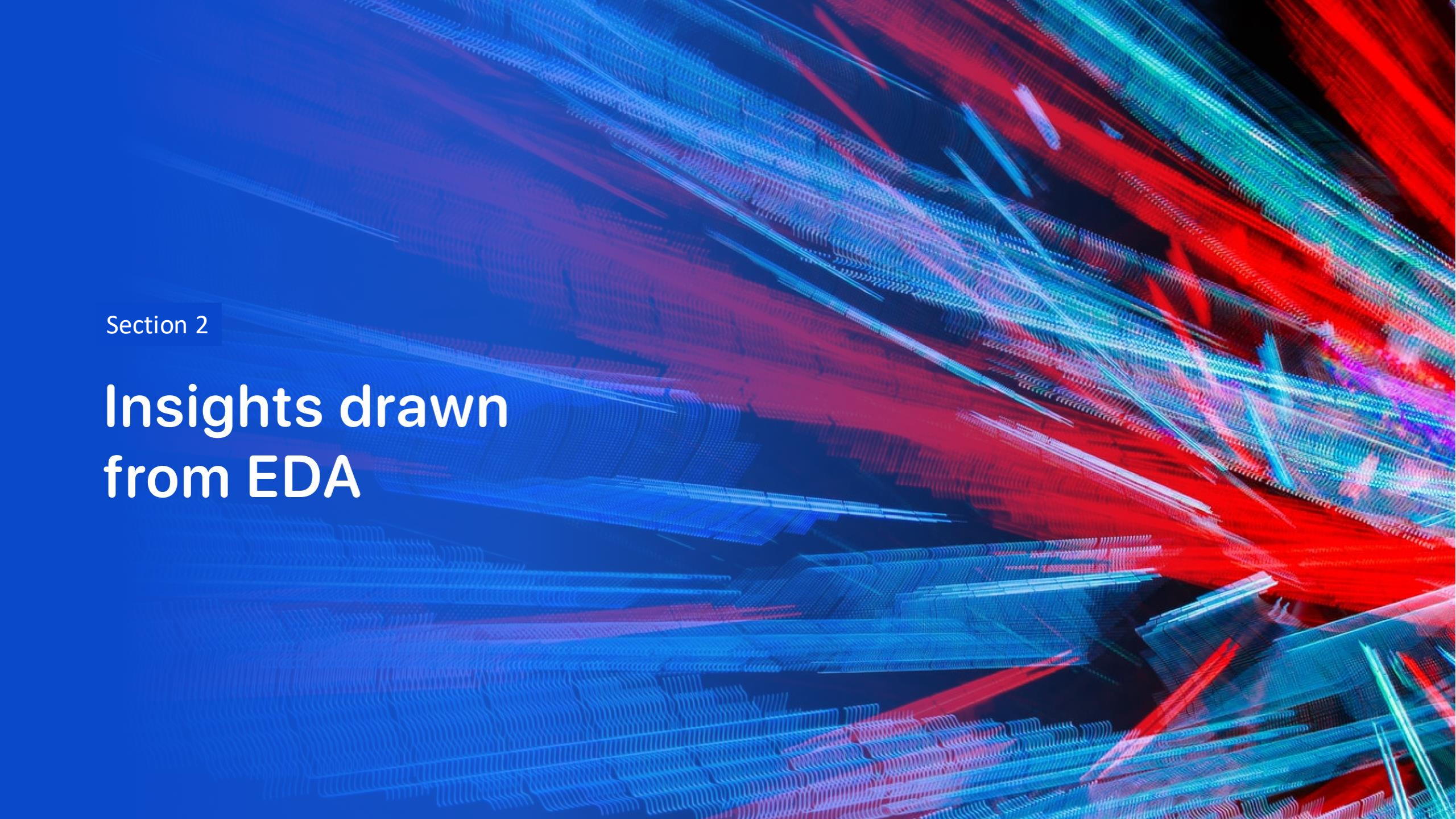




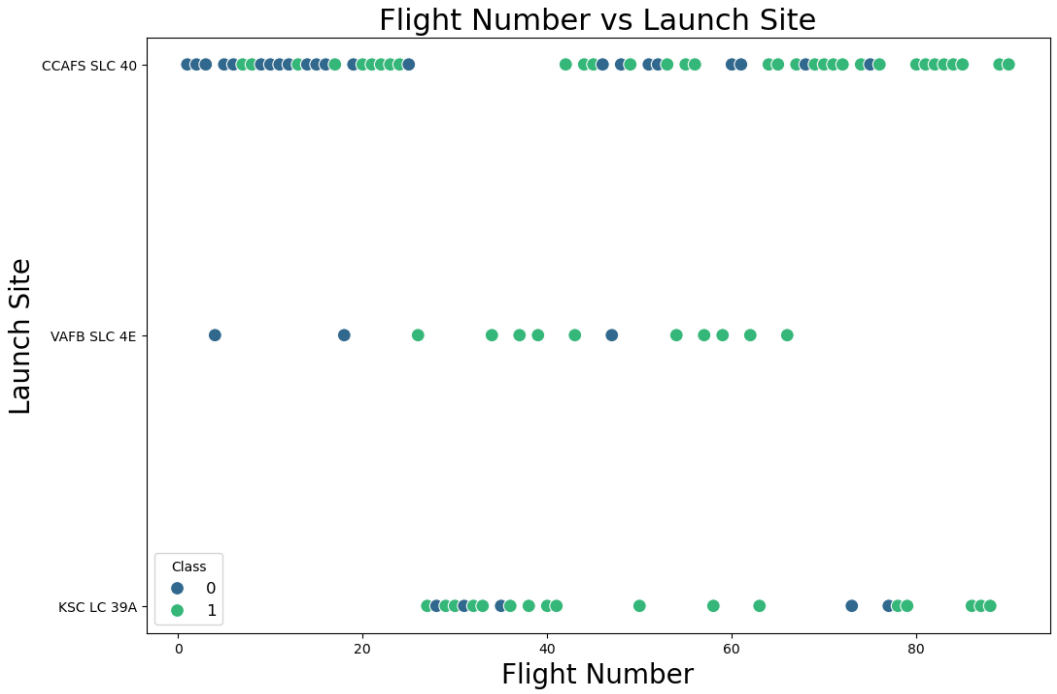
Results

- **Predictive Analysis Results:**
- **Model Performance**:
  - **Logistic Regression**: Accuracy = 83.33%
  - **SVM**: Accuracy = 83.33%
  - **Decision Tree**: Accuracy = 83.33%
  - **KNN**: Accuracy = 83.33%
- **Best Performing Model**: Logistic Regression and SVM showed the best balance of simplicity and accuracy.

The predictive analysis involved a systematic approach to building, evaluating, and improving classification models to forecast the success of Falcon 9 landings. By employing various algorithms and tuning their hyperparameters, the analysis revealed the most effective model for predicting landing outcomes, contributing valuable insights for future missions.


Section 2
Insights drawn from EDA

Flight Number vs. Launch Site




```
# Create a scatter plot
plt.figure(figsize=(12, 8))  # Set the figure size
sns.scatterplot(x="FlightNumber", y="LaunchSite", hue="Class", data=df, palette="viridis", s=100)

# Add labels and title
plt.xlabel("Flight Number", fontsize=20)
plt.ylabel("Launch Site", fontsize=20)
plt.title("Flight Number vs Launch Site", fontsize=22)

# Show the plot
plt.legend(title='Class', fontsize=12)
plt.show()

✓ 0.2s
```

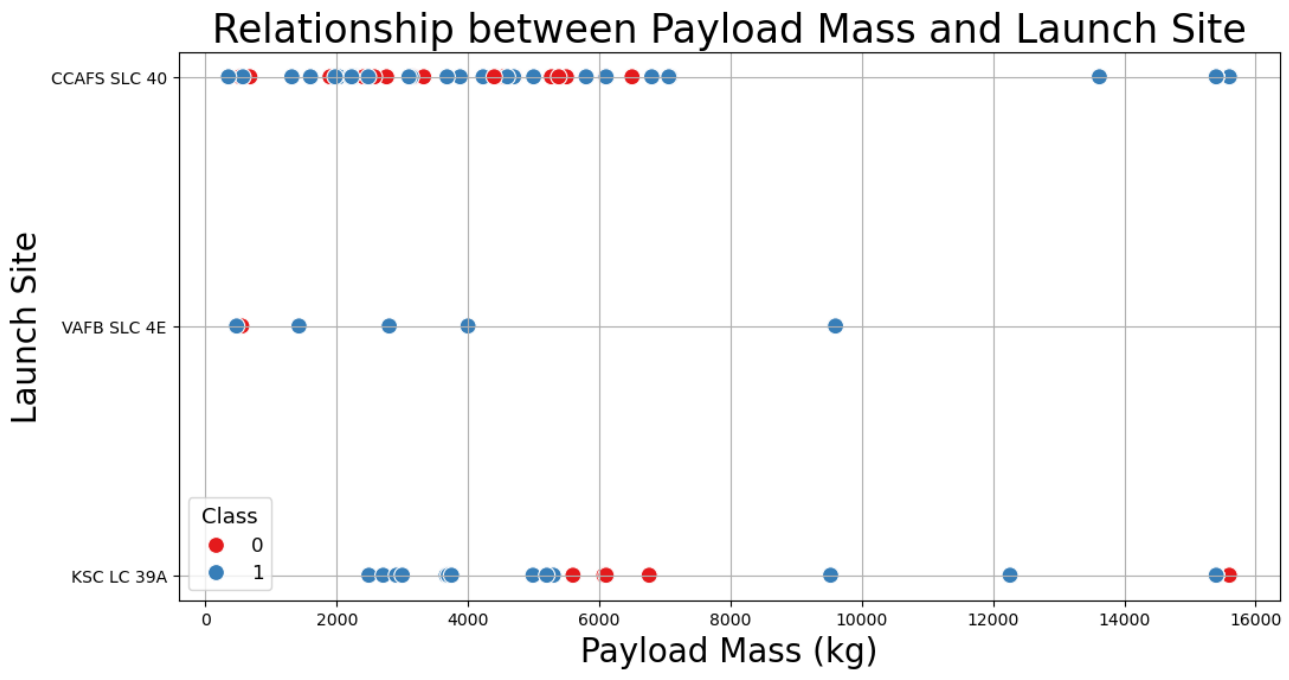
Explanation of the scatter plot:

- This scatter plot shows the correlation between **Flight Number** (which represents the sequence of launches) and the different **Launch Sites** (e.g., CCAFS SLC-40, KSC LC-39A).
- The scatter plot helps to identify patterns regarding which launch sites are used more frequently as the number of launches increases.
- For example, certain launch sites, such as **CCAFS SLC-40**, might show higher flight numbers, indicating frequent use, while others may have fewer launches.

Payload vs. Launch Site

Explanation of the scatter plot:

- The plot shows how different launch sites handle various payload masses.
- It helps identify if certain launch sites tend to carry heavier or lighter payloads.
- It also analyzes whether the success of the launches varies based on the launch site and payload mass.




```
plt.figure(figsize=(12, 6))
sns.scatterplot(x="PayloadMass", y="LaunchSite", hue="Class", data=df, palette="Set1", s=100)

# Añadir etiquetas y título
plt.xlabel("Payload Mass (kg)", fontsize=20)
plt.ylabel("Launch Site", fontsize=20)
plt.title("Relationship between Payload Mass and Launch Site", fontsize=24)
plt.legend(title='Class', fontsize=12, title_fontsize='13')
plt.grid(True)
plt.show()

✓ 0.2s
```

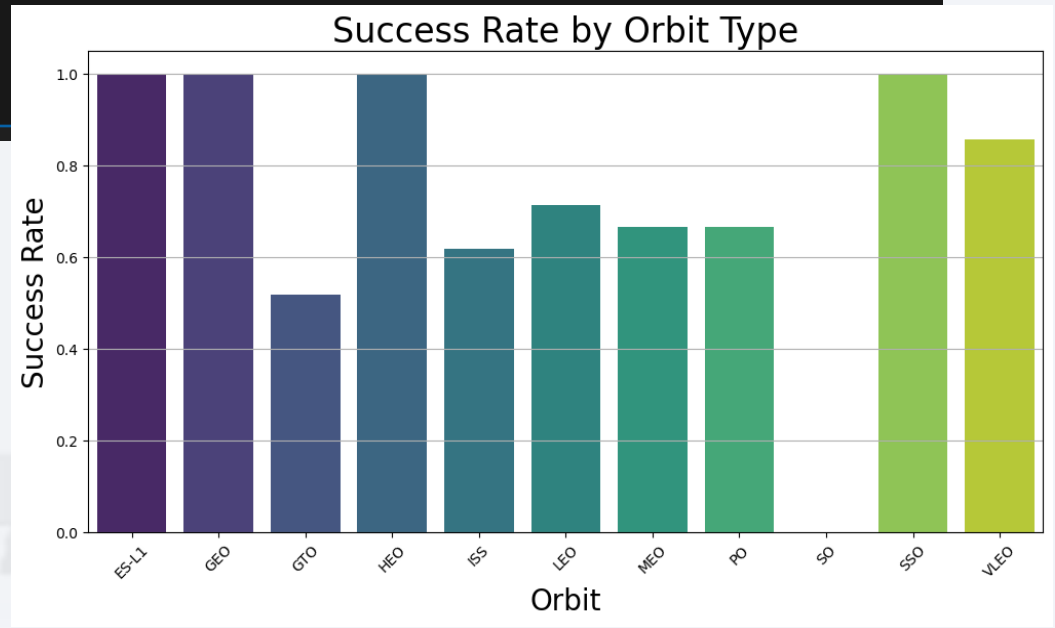
Success Rate vs. Orbit Type

```python
# Agrupar por 'Orbit' y calcular la tasa de éxito
success_rate = df.groupby('Orbit')['Class'].mean().reset_index()

# Trazar el gráfico de barras
plt.figure(figsize=(12, 6))
sns.barplot(x='Orbit', y='Class', data=success_rate, palette='viridis')

# Añadir etiquetas y título
plt.xlabel('Orbit', fontsize=20)
plt.ylabel('Success Rate', fontsize=20)
plt.title('Success Rate by Orbit Type', fontsize=24)
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.show()
```

✓ 0.3s



Explanation of the bar chart:

- The bar chart visualizes the success rate of launches across various orbit types (LEO, GTO, etc.).
- It allows the user to see which orbit types have higher success rates and which ones are more challenging.
- For instance, **LEO orbits** may show a higher success rate compared to **GTO orbits**, revealing important patterns for launch planning.

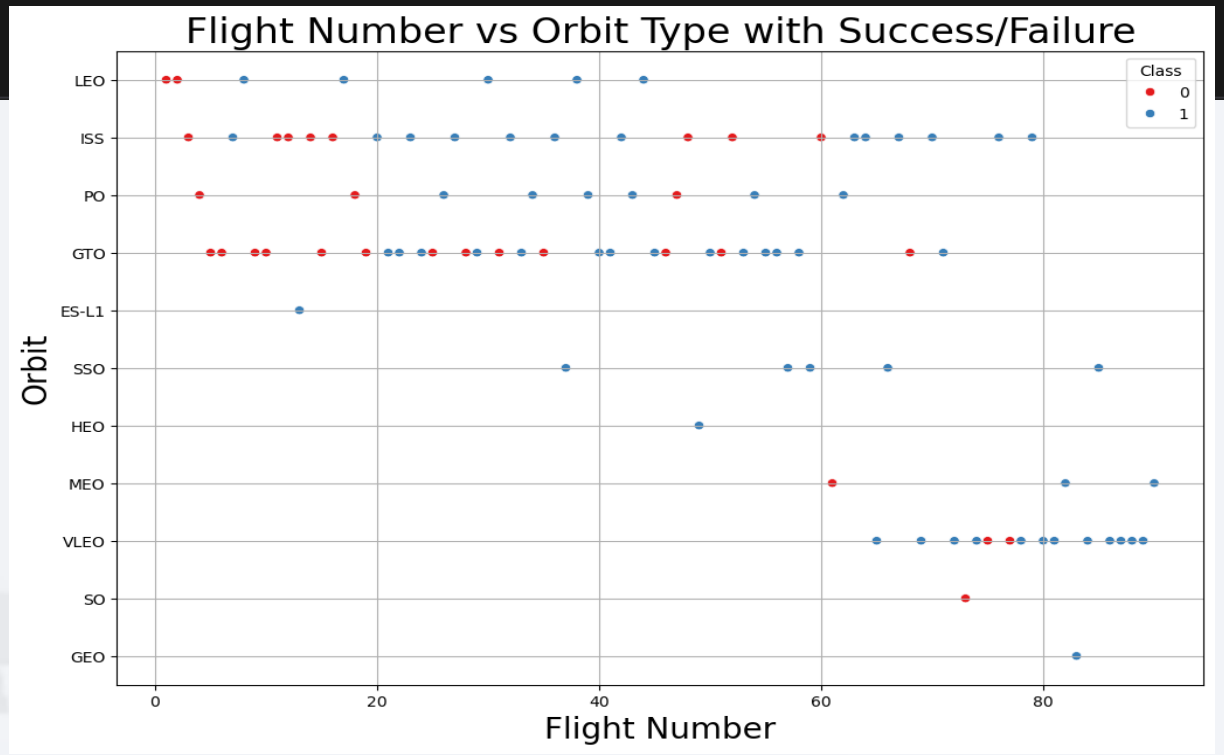
Flight Number vs. Orbit Type

```python
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value

# Traza un gráfico de dispersión
plt.figure(figsize=(12, 8))
sns.scatterplot(x='FlightNumber', y='Orbit', hue='Class', data=df, palette='Set1')

# Añadir etiquetas y título
plt.xlabel('Flight Number', fontsize=20)
plt.ylabel('Orbit', fontsize=20)
plt.title('Flight Number vs Orbit Type with Success/Failure', fontsize=24)
plt.legend(title='Class', loc='upper right')
plt.grid()
plt.show()
```

✓ 0.3s



Explanation of the scatter plot:

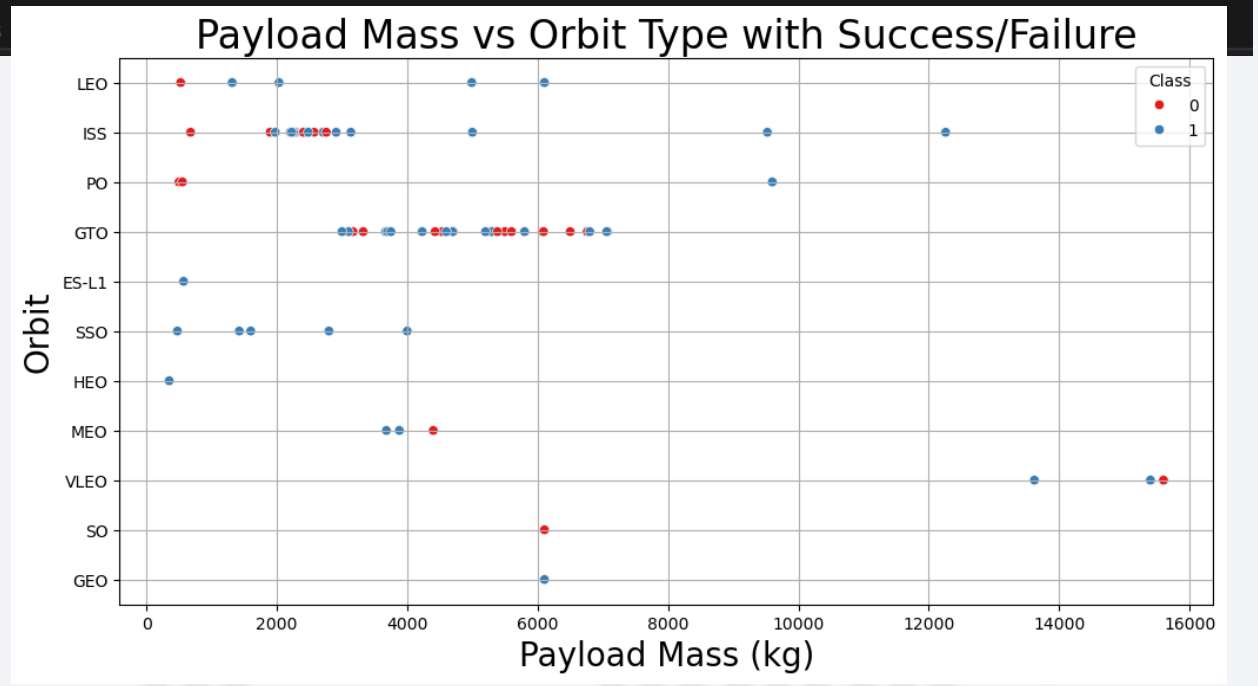
- This scatter plot shows how **Flight Numbers** (representing the sequence of launches) relate to **Orbit Types** (e.g., LEO, GTO) and their success rates.
- The plot helps visualize if there is any correlation between the number of flights and the success rate for specific orbits.
- For example, **LEO orbits** may show an increasing trend in success with higher flight numbers, while **GTO orbits** may not follow this pattern.

Payload vs. Orbit Type

- **Explanation of the scatter plot**:
  - This scatter plot visualizes how **Payload Mass** is distributed across different **Orbit Types** (e.g., LEO, GTO, Polar), and whether the launch was successful.
  - It helps analyze whether certain orbits handle heavier payloads better and if payload size impacts the likelihood of a successful landing.
  - For instance, **LEO orbits** might show a higher success rate with larger payloads, while **GTO orbits** might display mixed results for heavier payloads.

```
4  plt.figure(figsize=(12, 6))
5  sns.scatterplot(x='PayloadMass', y='Orbit', hue='Class', data=df, palette='Set1')
6
7  # Añadir etiquetas y título
8  plt.xlabel('Payload Mass (kg)', fontsize=20)
9  plt.ylabel('Orbit', fontsize=20)
10 plt.title('Payload Mass vs Orbit Type with Success/Failure', fontsize=24)
11 plt.legend(title='Class')
12 plt.grid()
13 plt.show()
```

✓ 0.2s



Launch Success Yearly Trend

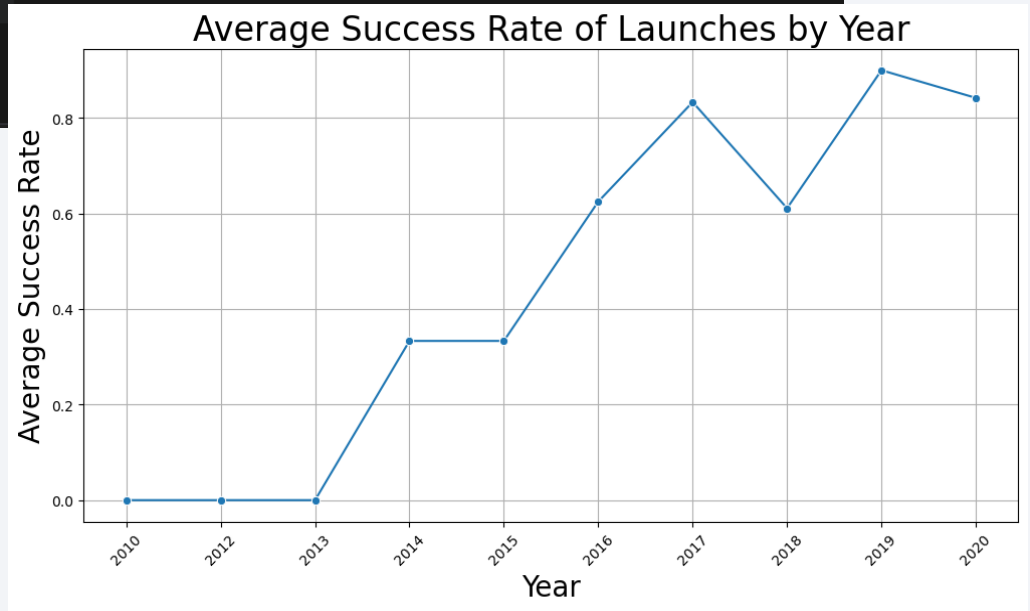
```python
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
# Extraer los años
df['Year'] = Extract_year(df['Date'])

# Calcular la tasa de éxito media por año
success_rate_by_year = df.groupby('Year')['Class'].mean().reset_index()

# Traza el gráfico de líneas
plt.figure(figsize=(12, 6))
sns.lineplot(x='Year', y='Class', data=success_rate_by_year, marker='o')

# Añadir etiquetas y título
plt.xlabel('Year', fontsize=20)
plt.ylabel('Average Success Rate', fontsize=20)
plt.title('Average Success Rate of Launches by Year', fontsize=24)
plt.xticks(rotation=45)
plt.grid()
plt.show()
```

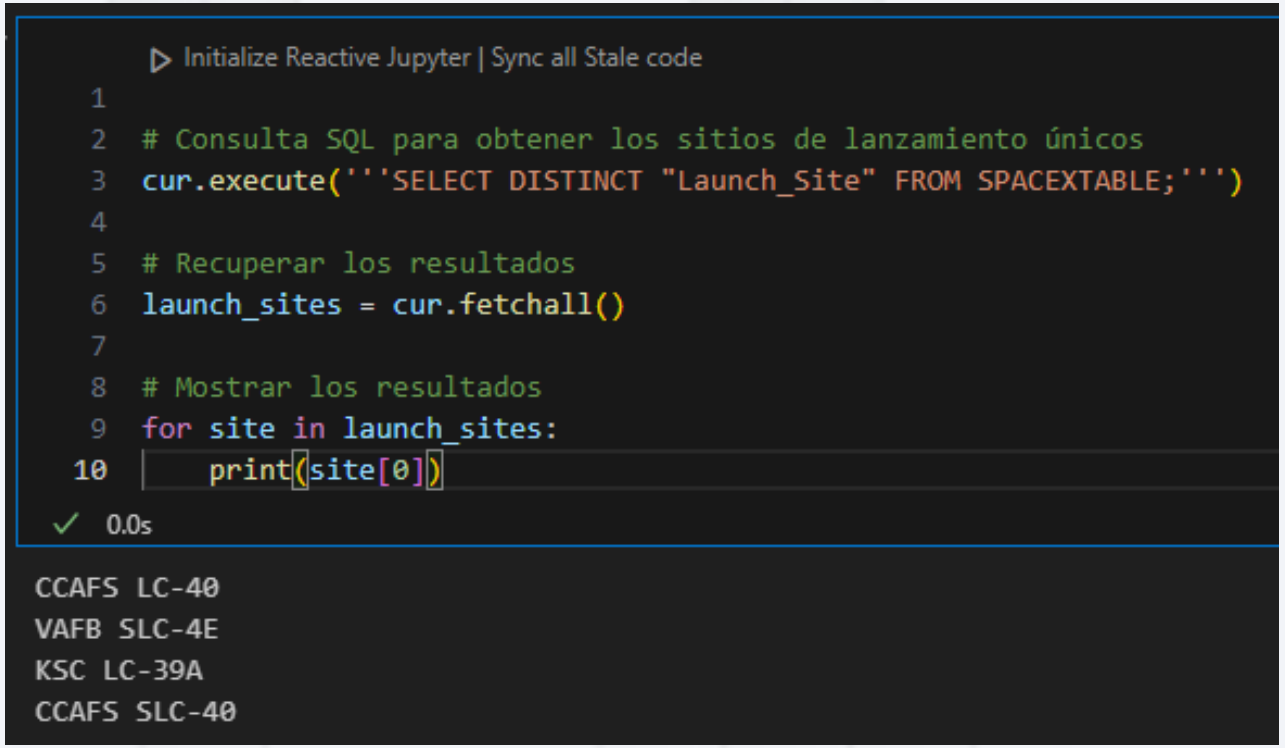
✓ 0.2s



Explanation of the line chart:

- This line chart visualizes the **average success rate** of Falcon 9 launches over the years.
- It allows you to see trends, showing how SpaceX's success rate has improved over time.
- For instance, the success rate may have remained stable or even slightly declined in the early years, but starting around 2015, the success rate steadily increased, reflecting improvements in technology and experience.

All Launch Site Names



Explanation:

1. The query successfully retrieved the unique launch site names used by SpaceX. These sites are critical for analyzing launch performance, as different sites may have varying success rates and payload capabilities.

2. **CCAFS SLC-40** and **KSC LC-39A** are frequently used sites, while **VAFB SLC-4E** is located on the West Coast for polar orbit launches.

Launch Site Names Begin with 'CCA'

- **Explanation**:
  - The query retrieves records of launch sites beginning with **'CCA'**, reflecting the various launch complexes at **Cape Canaveral Air Force Station**. Understanding these sites helps analyze launch performance and frequency, which is vital for SpaceX's operational strategy.

```
Initialize Reactive Jupyter | Sync all Stale code
# Consulta SQL para obtener 5 registros donde los sitios de lanzamiento comienzan con 'CCA'
cur.execute('''SELECT * FROM SPACEXTABLE WHERE "Launch_Site" LIKE 'CCA%' LIMIT 5;''')

# Recuperar los resultados
records = cur.fetchall()

# Mostrar los resultados
for record in records:
    print(record)
✓ 0.0s
('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')
```

Total Payload Mass

Explanation:

- The query calculates the total payload mass carried by SpaceX boosters for NASA's Commercial Resupply Services. This total payload is essential for understanding the scale of SpaceX's contributions to NASA's missions and their impact on cargo transportation to the ISS.

```
# Consulta SQL para sumar la masa total de las cargas útiles de las misiones "NASA (CRS)"
cur.execute('''
    SELECT SUM("PAYLOAD_MASS__KG_")
    FROM SPACEXTABLE
    WHERE "Customer" LIKE '%NASA (CRS)%';
''')

# Recuperar el resultado
total_mass = cur.fetchone()[0]

# Mostrar el resultado
print(f"Masa total de la carga útil transportada por los propulsores lanzados por la NASA (CRS): {total_mass} kg")

```

✓ 0.0s

```
Masa total de la carga útil transportada por los propulsores lanzados por la NASA (CRS): 48213 kg
```

Average Payload Mass by F9 v1.1

- **Explanation**:
  - This query successfully calculates the average payload mass carried by the Falcon 9 version 1.1 boosters. Understanding the average payload mass is crucial for evaluating the capacity and efficiency of the F9 v1.1 in comparison to other booster versions.
  - This analysis provides insights into the operational capabilities of SpaceX's Falcon 9 rockets and their ability to meet mission requirements.

```
# Consulta SQL para obtener la masa media de la carga útil de la versión de refuerzo "F9 v1.1"
cur.execute('''
    SELECT AVG("PAYLOAD_MASS__KG_")
    FROM SPACEXTABLE
    WHERE "Booster_Version" = 'F9 v1.1';
''')

# Recuperar el resultado
average_mass = cur.fetchone()[0]

# Mostrar el resultado
print(f"Masa media de la carga útil transportada por la versión de refuerzo F9 v1.1: {average_mass} kg")

✓ 0.0s
Masa media de la carga útil transportada por la versión de refuerzo F9 v1.1: 2928.4 kg
```

First Successful Ground Landing Date

- **Explanation**:

  The query retrieves the date of the first successful landing outcome on a ground pad. This date is significant as it highlights SpaceX's achievement in developing reusable rocket technology, paving the way for future advancements in spaceflight and cost reduction.

```
# Consulta SQL para obtener la fecha del primer aterrizaje exitoso en la plataforma de tierra (RTLS)
cur.execute('''
    SELECT MIN(Date)
    FROM SPACEXTABLE
    WHERE "Landing_Outcome" = 'Success (ground pad)';
''')

# Recuperar el resultado
first_successful_rtls_date = cur.fetchone()[0]

# Mostrar el resultado
print(f"Fecha del primer aterrizaje exitoso en la plataforma de tierra: {first_successful_rtls_date}")

✓ 0.0s
Fecha del primer aterrizaje exitoso en la plataforma de tierra: 2015-12-22
```

Successful Drone Ship Landing with Payload between 4000 and 6000

Explanation:

The query lists the names of boosters that successfully landed on drone ships while carrying payloads between 4000 kg and 6000 kg. This analysis helps evaluate the performance and reliability of specific booster versions, informing operational decisions for future missions.

```
Initialize Reactive Jupyter | Sync all Stale code
cur.execute('''
    SELECT DISTINCT "Booster_Version"
    FROM SPACEXTABLE
    WHERE "Landing_Outcome" IN ('Success (drone ship)', 'Success')
    AND "PAYLOAD_MASS__KG_" > 4000
    AND "PAYLOAD_MASS__KG_" < 6000;
''')
# Recuperar el resultado
nombres_propulsores = cur.fetchall()

# Mostrar el resultado
print(f"Nombres de los propulsores que tienen éxito y tienen una masa de carga útil >4000 pero <6000: {nombres_propulsores}")
✓ 0.0s
Nombres de los propulsores que tienen éxito y tienen una masa de carga útil >4000 pero <6000: [('F9 FT B1022',), ('F9 FT B1026',), ('
```

Total Number of Successful and Failure Mission Outcomes

- **Explanation**:

The query calculates the total number of successful and failed mission outcomes from the SpaceX dataset. This analysis is essential for assessing SpaceX's performance in launching missions and can help identify trends or patterns in mission success over time.

```
Initialize Reactive Jupyter | Sync all Stale code
# Ejecutar la consulta SQL para contar los resultados de misiones exitosas y fallidas
cur.execute('''
    SELECT "Mission_Outcome", COUNT(*) AS Total
    FROM SPACEXTABLE
    GROUP BY "Mission_Outcome";
''')

# Recuperar todos los resultados
resultados = cur.fetchall()

# Mostrar el resultado
print("Número total de resultados de misiones exitosas y fallidas:")
for resultado in resultados:
    print(f"{resultado[0]}: {resultado[1]}")
```

✓ 0.0s

```
Número total de resultados de misiones exitosas y fallidas:
Failure (in flight): 1
Success: 98
Success : 1
Success (payload status unclear): 1
```

Boosters Carried Maximum Payload

- **Explanation:**

The query lists the names of boosters that have carried the maximum payload mass. This analysis is important for understanding the payload capabilities of different booster versions and their applicability for various mission types.

```
Initialize Reactive Jupyter | Sync all Stale code
# Ejecutar la consulta SQL utilizando una subconsulta para encontrar los nombres de los booster_versions
cur.execute('''
    SELECT DISTINCT "Booster_Version"
    FROM SPACEXTABLE
    WHERE "PAYLOAD_MASS__KG_" = (
        SELECT MAX("PAYLOAD_MASS__KG_")
        FROM SPACEXTABLE
    );
''')

# Recuperar todos los resultados
nombres_boosters = cur.fetchall()

# Mostrar el resultado
print("Nombres de las versiones de booster que han transportado la masa máxima de carga útil:")
for booster in nombres_boosters:
    print(booster[0])
```

✓ 0.0s

```
Nombres de las versiones de booster que han transportado la masa máxima de carga útil:
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

- **Explanation**:

The query lists the failed landing outcomes on drone ships for the year 2015, along with their corresponding booster versions and launch site names. This analysis is essential for understanding the challenges faced during landing attempts and for improving the reliability of future missions.

```
# Ejecutar la consulta SQL para obtener los registros requeridos
cur.execute('''
    SELECT
        CASE
            WHEN substr("Date", 6, 2) = '01' THEN 'January'
            WHEN substr("Date", 6, 2) = '02' THEN 'February'
            WHEN substr("Date", 6, 2) = '03' THEN 'March'
            WHEN substr("Date", 6, 2) = '04' THEN 'April'
            WHEN substr("Date", 6, 2) = '05' THEN 'May'
            WHEN substr("Date", 6, 2) = '06' THEN 'June'
            WHEN substr("Date", 6, 2) = '07' THEN 'July'
            WHEN substr("Date", 6, 2) = '08' THEN 'August'
            WHEN substr("Date", 6, 2) = '09' THEN 'September'
            WHEN substr("Date", 6, 2) = '10' THEN 'October'
            WHEN substr("Date", 6, 2) = '11' THEN 'November'
            WHEN substr("Date", 6, 2) = '12' THEN 'December'
        END AS "Month",
        "Landing_Outcome",
        "Booster_Version",
        "Launch_Site"
    FROM SPACEXTABLE
    WHERE "Landing_Outcome" LIKE 'Failure%'
      AND substr("Date", 0, 5) = '2015';
''')

# Recuperar todos los resultados
resultados = cur.fetchall()

# Mostrar el resultado
print("Registros de fallas en la nave no tripulada para el año 2015:")
for resultado in resultados:
    print(f"Month: {resultado[0]}, Landing Outcome: {resultado[1]}, Booster Version: {resultado[2]}, Launch Site: {resultado[3]}")

✓ 0.0s
Registros de fallas en la nave no tripulada para el año 2015:
Month: January, Landing Outcome: Failure (drone ship), Booster Version: F9 v1.1 B1012, Launch Site: CCAFS LC-40
Month: April, Landing Outcome: Failure (drone ship), Booster Version: F9 v1.1 B1015, Launch Site: CCAFS LC-40
```

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

- **Explanation**:

The query ranks the count of landing outcomes between specified dates, showing the distribution of successes and failures. This analysis is critical for understanding SpaceX's performance trends and areas that may require improvement.

```
# Ejecutar la consulta SQL para contar los resultados de aterrizaje en el rango de fechas especificado
cur.execute('''
    SELECT
        CASE
            WHEN "Landing_Outcome" IN ('Success (ground pad)', 'Success (drone ship)', 'Success') THEN 'Éxito (plataforma terrestre)'
            WHEN "Landing_Outcome" LIKE 'Failure%' THEN 'Fracaso (nave no tripulada)'
            ELSE 'Otros'
        END AS "Resultado",
        COUNT(*) AS "Total"
    FROM SPACEXTABLE
    WHERE "Date" BETWEEN '2010-06-04' AND '2017-03-20'
    GROUP BY "Resultado"
    ORDER BY "Total" DESC;
''')

# Recuperar todos los resultados
resultados = cur.fetchall()

# Mostrar el resultado
print("Recuento de resultados de aterrizaje entre 2010-06-04 y 2017-03-20:")
for resultado in resultados:
    print(f"Resultado: {resultado[0]}, Total: {resultado[1]}")

✓ 0.0s
```

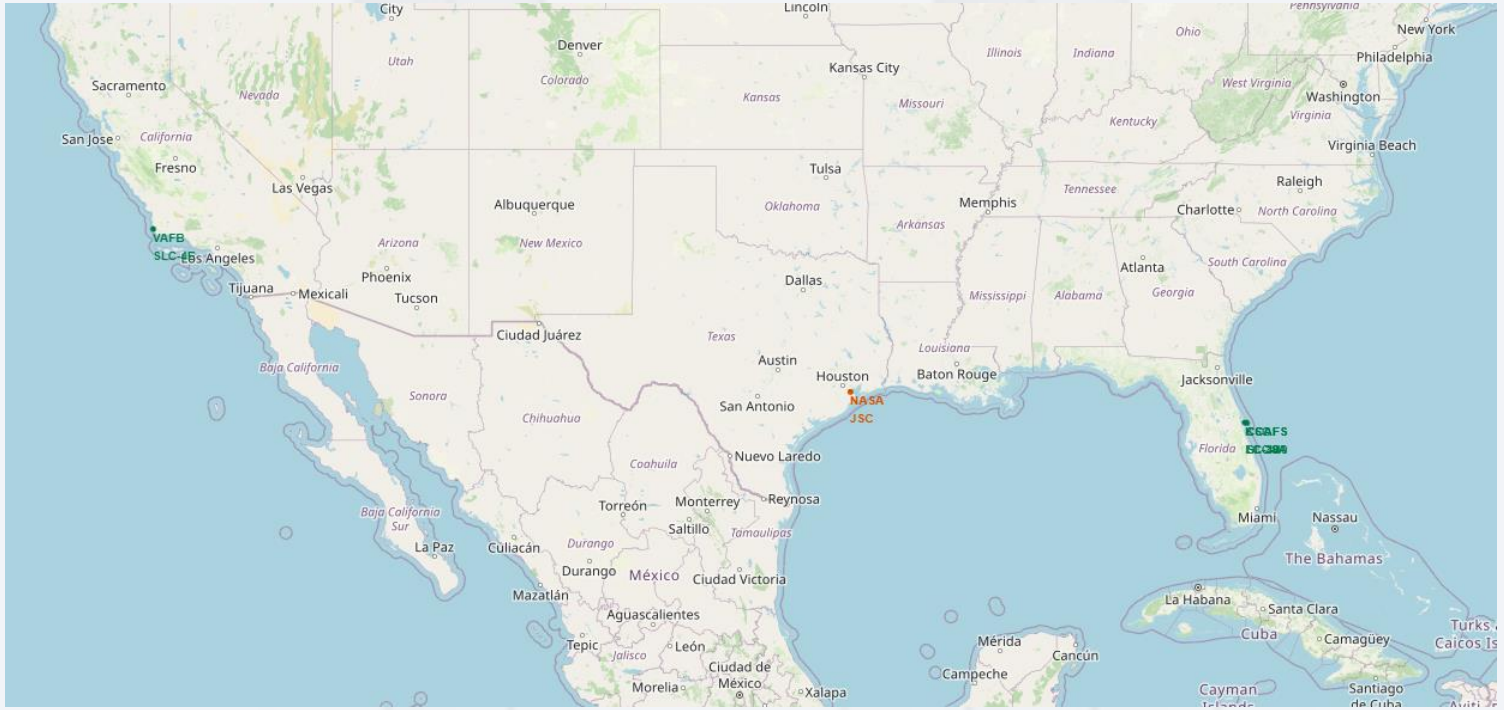
```
Recuento de resultados de aterrizaje entre 2010-06-04 y 2017-03-20:
Resultado: Otros, Total: 16
Resultado: Éxito (plataforma terrestre), Total: 8
Resultado: Fracaso (nave no tripulada), Total: 7
```

Section 3

Launch Sites Proximities Analysis

Global Launch Sites Map with SpaceX Markers

- **Explanation**:
- The map displays the global locations of SpaceX launch sites, marked with distinctive icons. Each marker indicates a launch site, such as Cape Canaveral, Kennedy Space Center, and Vandenberg Space Force Base.
- The geographical distribution illustrates that all sites are positioned near coastlines, enhancing safety during launches. This strategic placement is critical for optimizing launch trajectories and supporting recovery operations.



Launch Outcomes Visualization on Folium Map

- **Explanation**:

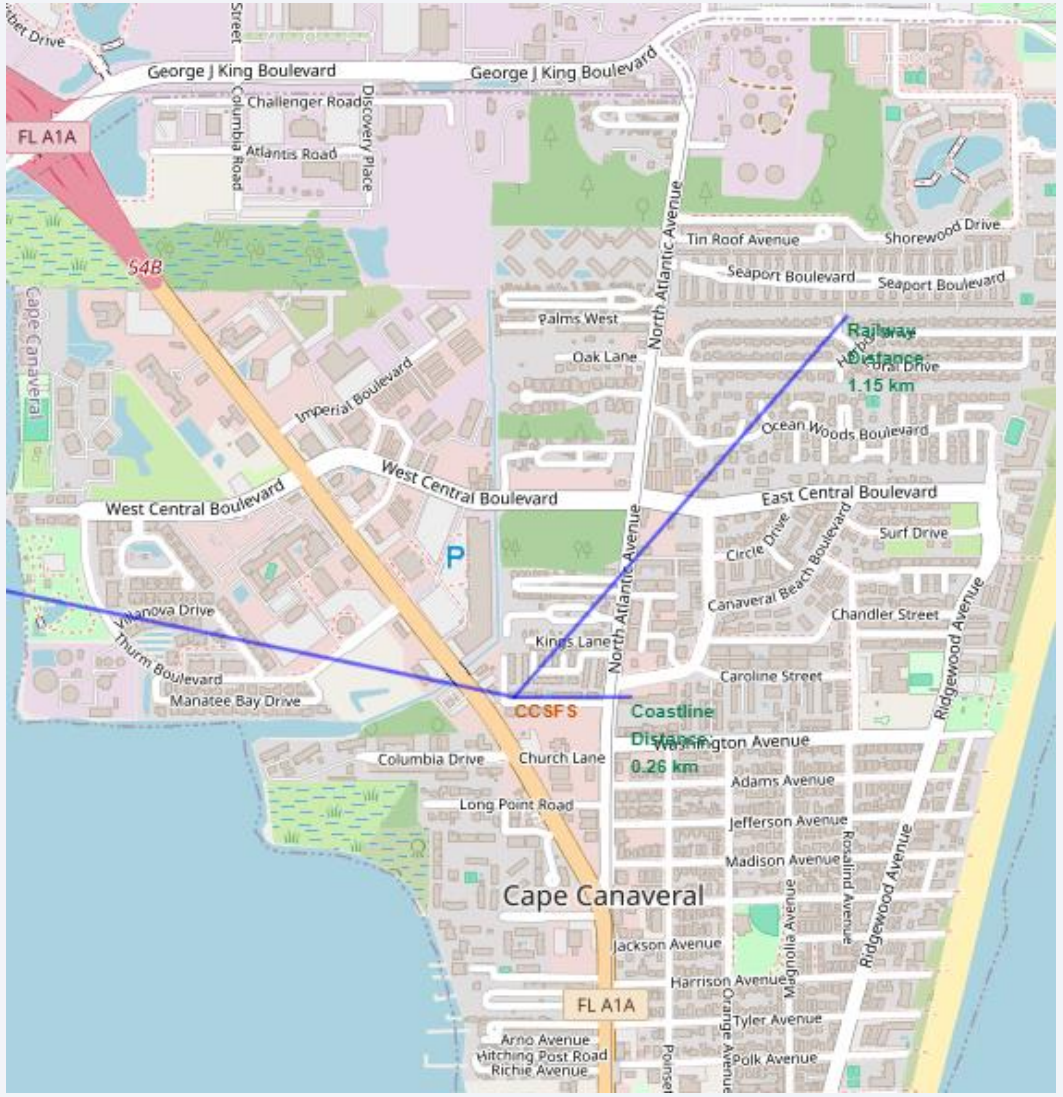
  *The map displays color-labeled markers representing the outcomes of SpaceX launches. Green markers indicate successful landings, while red markers represent failures.

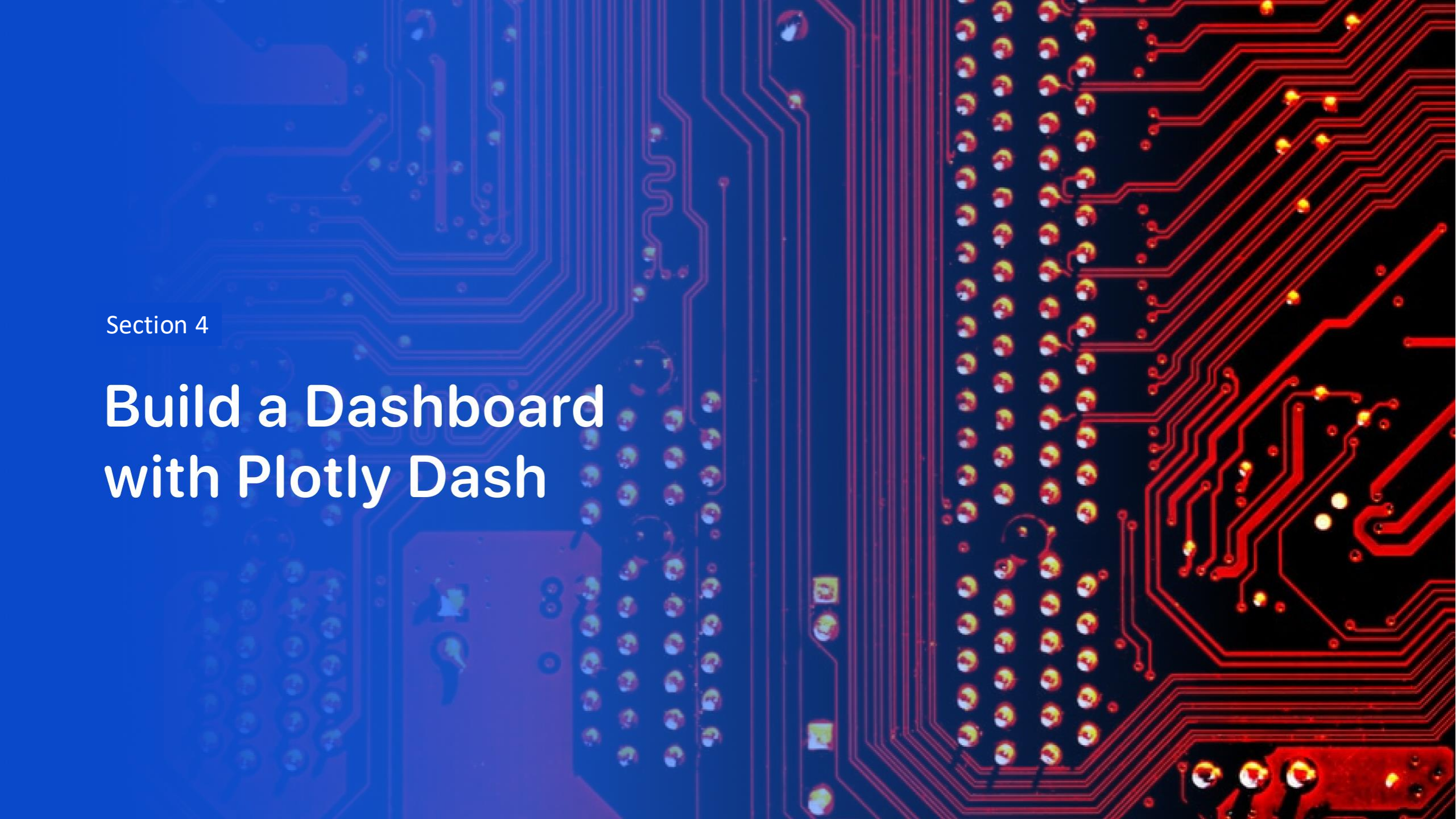
  *This visualization allows for an immediate assessment of launch success rates across different sites and highlights patterns in operational performance.

  *Notably, certain sites may show a higher concentration of failures, which can inform strategies for improving future launches.

Proximity Analysis of Selected Launch Site on Folium Map

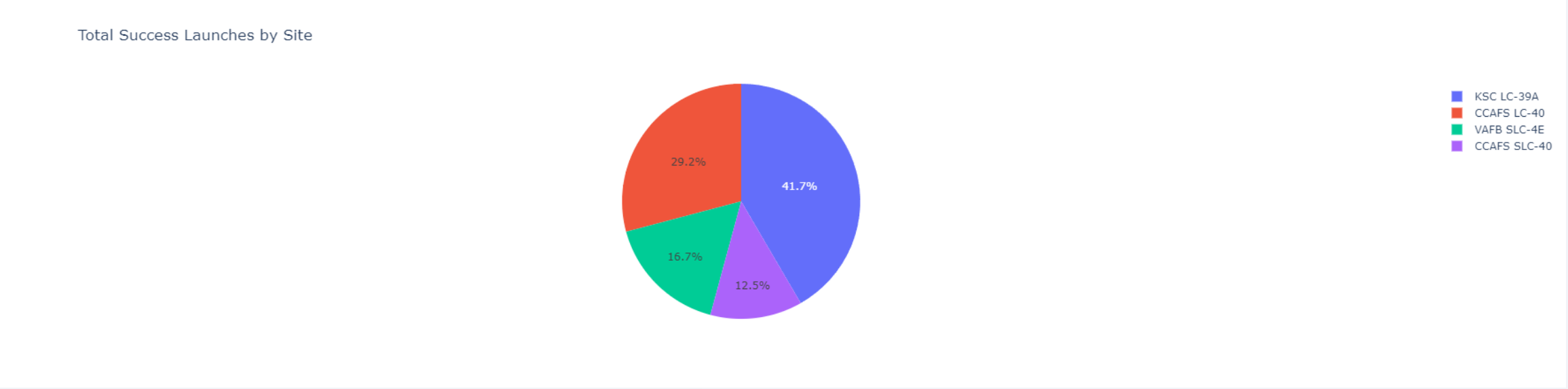
- **Explanation**:
  - The map highlights the proximity of **Cape Canaveral Space Force Station** to critical infrastructure such as railways, highways, and the coastline.
  - The distance lines demonstrate how these features relate to the launch site, with distances labeled for clarity (e.g., "Distance to Coastline: 5.2 km").
  - This analysis reveals that the site is strategically located, enhancing operational logistics and safety considerations for SpaceX launches.




Section 4
Build a Dashboard with Plotly Dash

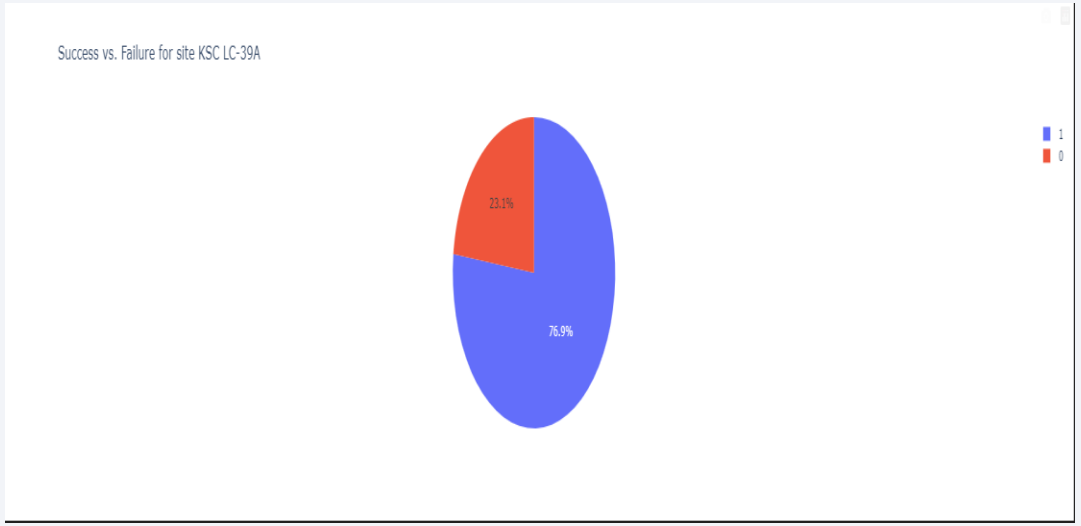
Launch Success Count for All Sites (Pie Chart)

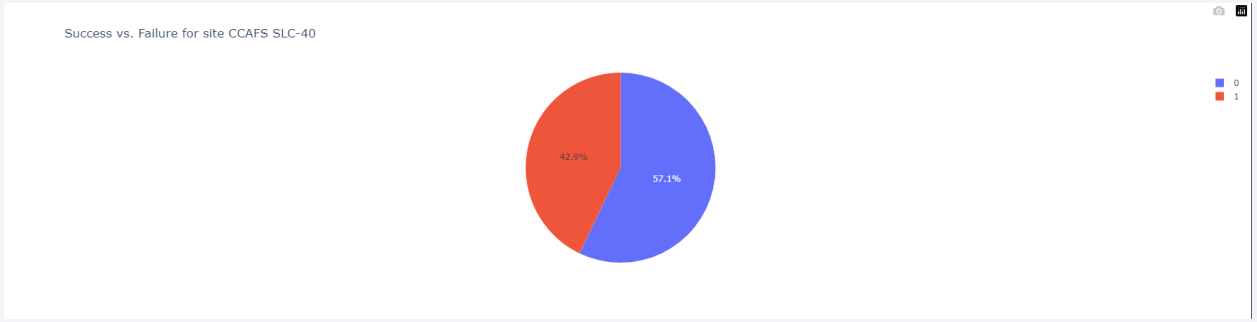
- **Explanation**:
  - **Importance**: This visualization helps identify which launch sites are most successful, which is critical for planning future missions and optimizing resources.
  - **Insights**: From the chart, you can conclude whether certain sites are more reliable for successful landings. This can also highlight operational efficiency at specific locations.



Launch Site with Highest Success Ratio (Pie Chart)

- **Importance**: This pie chart highlights the reliability of a particular launch site, **KSC LC-39A** or **CCAFS SLC-40,** which could be essential for mission planning and site optimization.
- **Insights**: By analyzing this launch site, SpaceX can allocate more resources to high-performing locations or investigate why other sites have lower success rates. The site with the highest success ratio may indicate operational efficiency or favorable conditions for launches.





Payload vs. Launch Outcome Scatter Plot

- **Key Elements**:
  - The x-axis represents **Payload Mass (kg)**.
  - The y-axis shows **Launch Outcomes** (Success or Failure).
  - Different colors represent various **Booster Versions**.
- **Findings**:
  - The analysis indicates that payloads between **[insert specific range]** tend to have higher success rates, while those above **[insert specific threshold]** often face more failures.
  - Certain booster versions, such as **[insert booster version]**, show consistent success across varying payloads, suggesting they are well-optimized for heavier launches.

Section 5

Predictive Analysis (Classification)

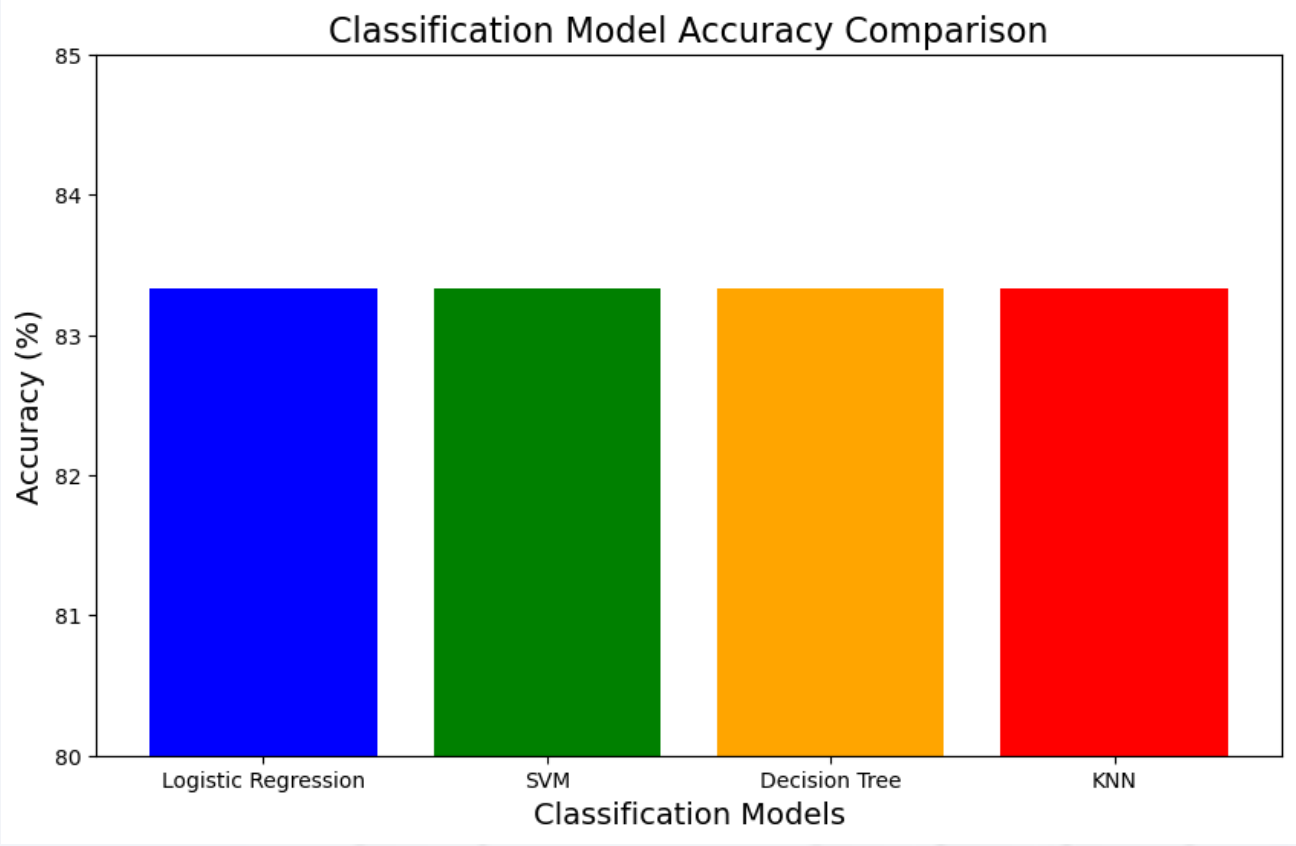
Classification Accuracy

- **Explanation**:

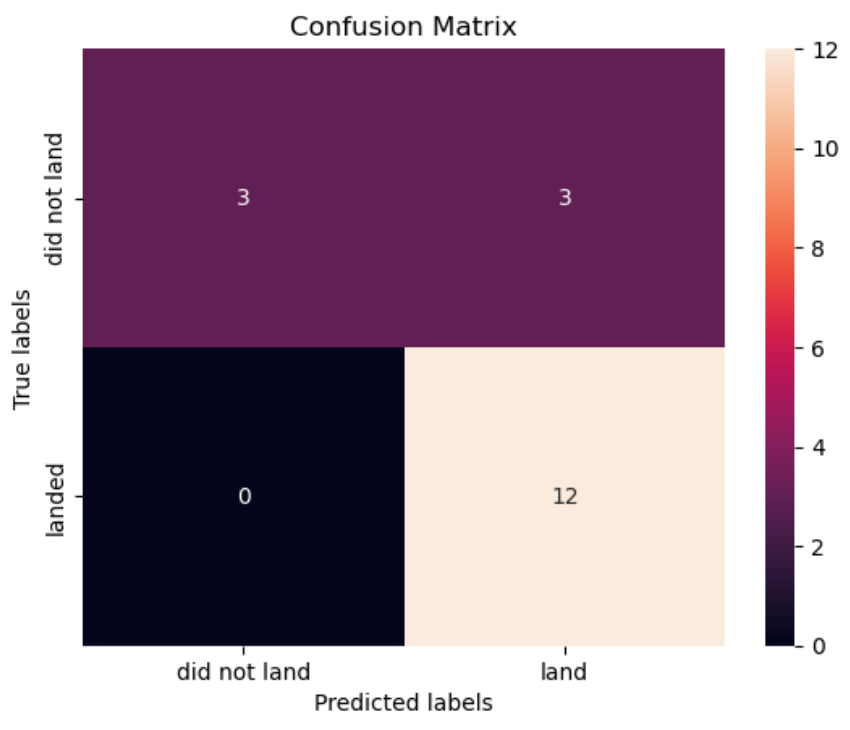
This bar chart compares the accuracies of the classification models built to predict SpaceX Falcon 9 landing success.

The models, including **Logistic Regression**, **SVM**, **Decision Tree**, and **KNN**, all achieved an accuracy of approximately **83.33%**.

Since the accuracy is identical across the models, other metrics (e.g., precision, recall) or model performance in different scenarios may be considered to select the best-performing model for specific use cases.



Confusion Matrix



The confusion matrix is identical regardless of the model chosen as well as its performance

Accuracy of Logistic Regression: 0.8333333333333334

Accuracy of SVM: 0.8333333333333334

Accuracy of Decision Tree: 0.8333333333333334

Accuracy of KNN: 0.8333333333333334

Conclusions

- **Effective Data Collection via API and Web Scraping**:

  Data was successfully collected from both the SpaceX API and Wikipedia through web scraping. This allowed for a more complete dataset, integrating both launch records and landing outcomes for Falcon 9 missions.

- **Exploratory Data Analysis Revealed Key Patterns**:

  Through exploratory data analysis (EDA), we discovered important patterns, such as a correlation between payload mass and launch outcomes, as well as the impact of orbit type and launch site on success rates.

- **Machine Learning Models Achieved Comparable Accuracy**:

  All classification models (Logistic Regression, SVM, Decision Tree, KNN) achieved an accuracy of **83.33%** on the test data. However, additional metrics such as precision and recall could further differentiate their performance.

- **Strategic Insights for SpaceX Launch Sites**:

  The interactive map built with Folium highlighted the proximity of launch sites to key infrastructures like coastlines, highways, and railways. This geographical analysis suggests that site location plays a crucial role in launch success and logistics.

- **Future Model Improvements**:

  Despite achieving consistent accuracy, further improvements could be made by optimizing hyperparameters, introducing more complex models (e.g., random forests), or including additional features related to environmental factors or real-time data.

Appendix

- **1. Python Code Snippets**
  - **API Data Collection**:
  - **Web Scraping using BeautifulSoup**:

```python
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

0.0s Python

Initialize Reactive Jupyter | Sync all Stale code

```python
response = requests.get(spacex_url)
```

0.4s Python

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

Initialize Reactive Jupyter | Sync all Stale code

```python
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
# Verificar si la solicitud fue exitosa (código de estado 200)
if response.status_code == 200:
    # Parsear el contenido HTML usando BeautifulSoup
    soup = BeautifulSoup(response.content, 'html.parser')
else:
    print(f"Error al cargar la página. Código de estado: {response.status_code}")
```

1.3s Python

Appendix

- **1. Python Code Snippets**
  - **Machine Learning Model Accuracy Comparison**:

```python
# Model names and their corresponding accuracies
models = ['Logistic Regression', 'SVM', 'Decision Tree', 'KNN']
accuracies = [accuracy_logreg * 100, accuracy_svm * 100, accuracy_tree * 100, accuracy_knn * 100]  # C

# Create the bar chart
plt.figure(figsize=(10, 6))
plt.bar(models, accuracies, color=['blue', 'green', 'orange', 'red'])

# Add labels and title
plt.xlabel('Classification Models', fontsize=14)
plt.ylabel('Accuracy (%)', fontsize=14)
plt.title('Classification Model Accuracy Comparison', fontsize=16)
plt.ylim([80, 85])  # Adjust y-axis to focus on accuracy values
plt.show()
```

✓ 0.2s Python

Appendix

- **2. SQL Queries**
  - **Unique Launch Sites**:

```
# Consulta SQL para obtener los sitios de lanzamiento únicos
cur.execute('''SELECT DISTINCT "Launch_Site" FROM SPACEXTABLE;''')

```

  - **Total Payload for NASA (CRS) Missions**:

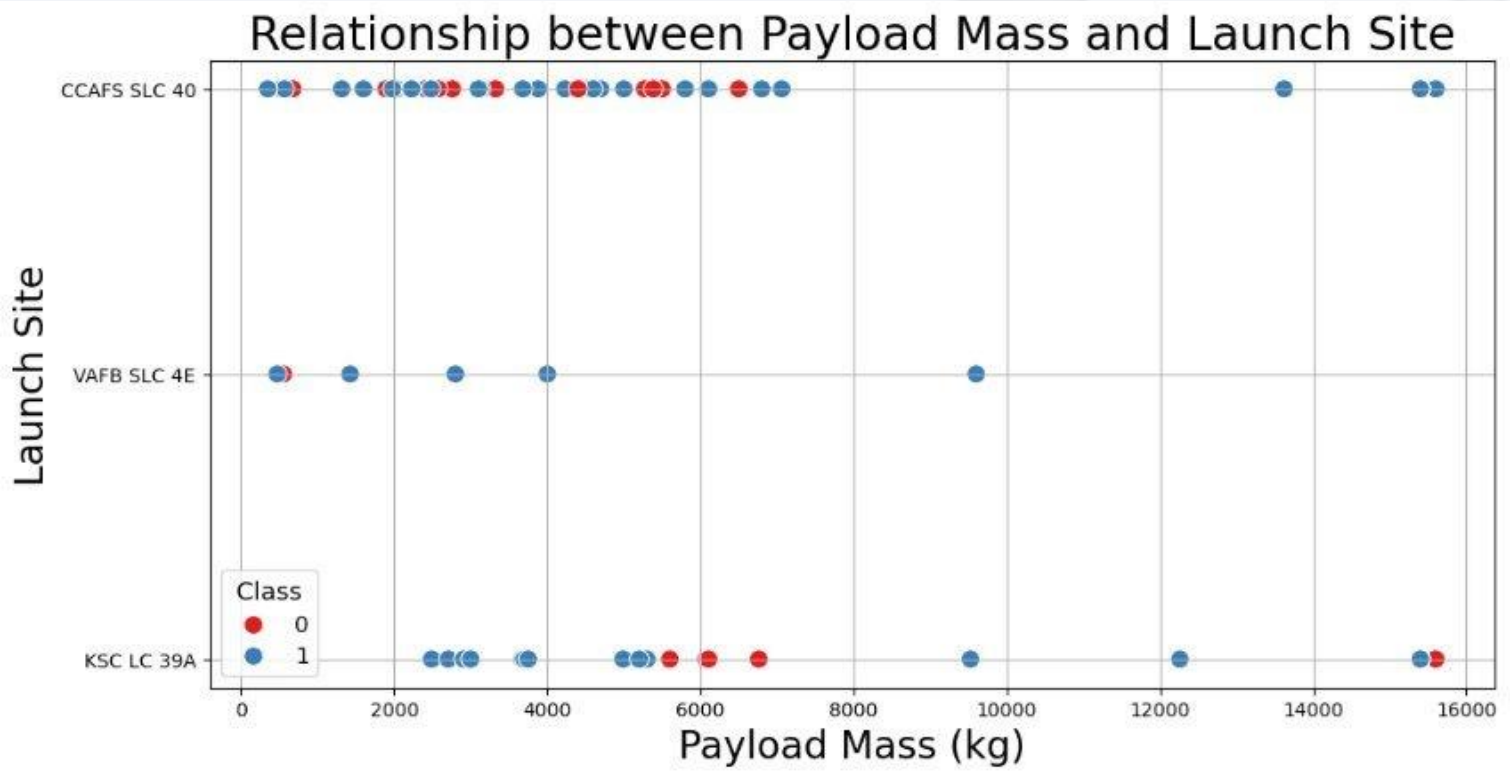
```
# Consulta SQL para sumar la masa total de las cargas útiles de las misiones "NASA (CRS)"
cur.execute('''
    SELECT SUM("PAYLOAD_MASS__KG_")
    FROM SPACEXTABLE
    WHERE "Customer" LIKE '%NASA (CRS)%';
''')
```

  - **Landing Outcomes Count**:

```
# Ejecutar la consulta SQL para contar los resultados de misiones exitosas y fallidas
cur.execute('''
    SELECT "Mission_Outcome", COUNT(*) AS Total
    FROM SPACEXTABLE
    GROUP BY "Mission_Outcome";
''')
```

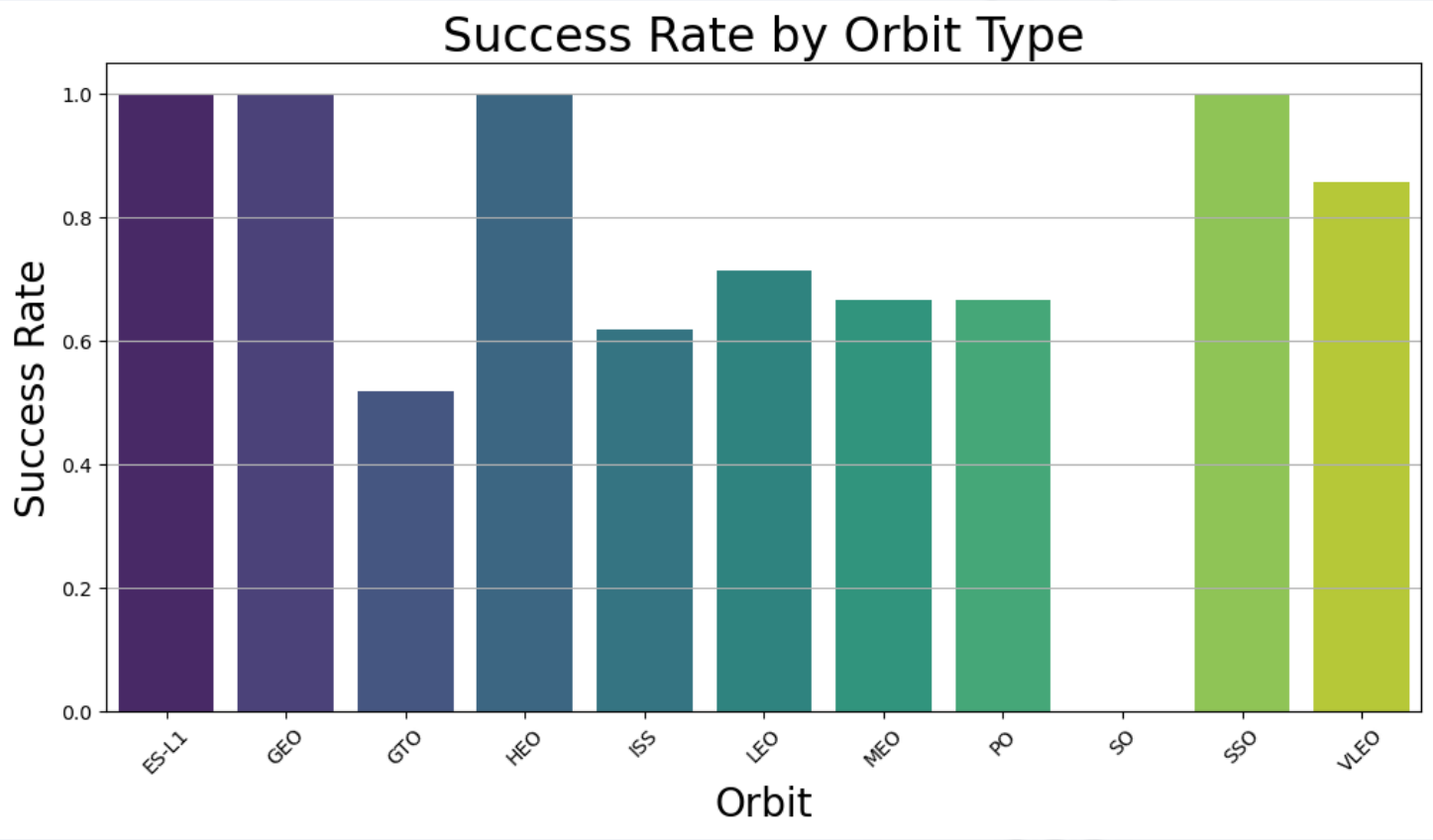
Appendix

- **3. Charts and Visualizations**
  - **Scatter Plot: Payload vs. Launch Site**:



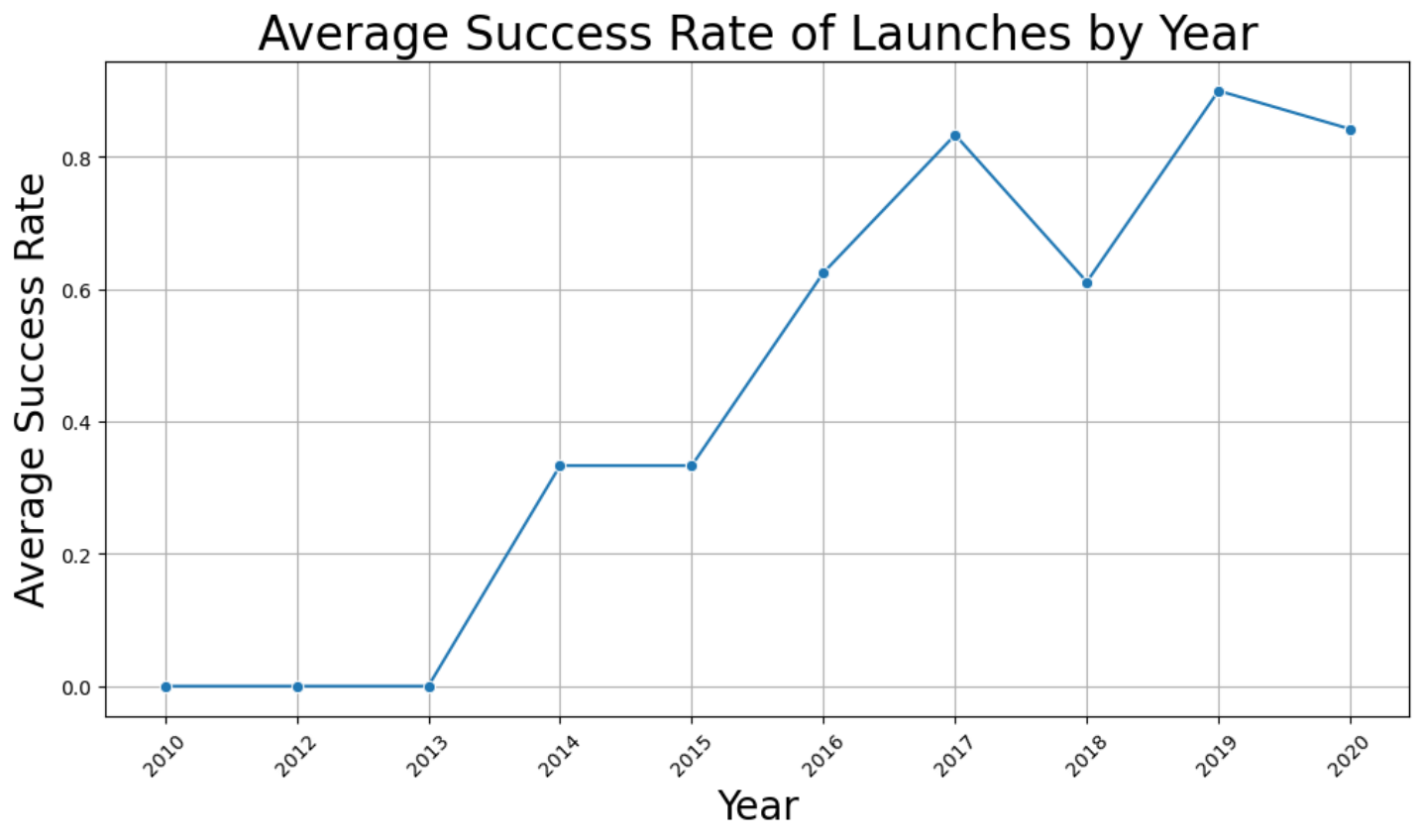
Appendix

- **Bar Chart: Success Rate by Orbit Type**: *(Include screenshot of the bar chart here)*



Appendix

- **Line Chart: Yearly Average Success Rate**:



Appendix

- **4. Notebook Outputs**
  - **Notebook Screenshots:**

DataFrame from API:

```
1  df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_1.csv")
2  df.head(10)
```

✓ 1.1s

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003	-80.577366	28.561857
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005	-80.577366	28.561857
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007	-80.577366	28.561857
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003	-120.610829	34.632093
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004	-80.577366	28.561857
5	6	2014-01-06	Falcon 9	3325.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1005	-80.577366	28.561857
6	7	2014-04-18	Falcon 9	2296.000000	ISS	CCAFS SLC 40	True Ocean	1	False	False	True	NaN	1.0	0	B1006	-80.577366	28.561857
7	8	2014-07-14	Falcon 9	1316.000000	LEO	CCAFS SLC 40	True Ocean	1	False	False	True	NaN	1.0	0	B1007	-80.577366	28.561857
8	9	2014-08-05	Falcon 9	4535.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1008	-80.577366	28.561857
9	10	2014-09-07	Falcon 9	4428.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1011	-80.577366	28.561857

Appendix

- **4. Notebook Outputs**
  - **Notebook Screenshots**:

    Count of failed Landing :

```
1  # Contar cuántas veces aparece 0 en la columna 'Class'
2  number_of_failed_landings = df['Class'].value_counts().get(0, 0)
3
4  print(f"Número de aterrizajes fallidos: {number_of_failed_landings}")
✓ 0.0s
Número de aterrizajes fallidos: 30
```

Appendix

Model Performance:

- **Logistic Regression**:

```
Initialize Reactive Jupyter | Sync all Stale code
1  parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
2  lr=LogisticRegression()
3  logreg_cv = GridSearchCV(lr, parameters, cv=10)
4  logreg_cv.fit(X_train, Y_train)
✓ 0.1s
```

```
▸ GridSearchCV ⓘ ?
▸ estimator: LogisticRegression
  ▸ LogisticRegression ?
```

```
1  print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
2  print("accuracy :",logreg_cv.best_score_)
✓ 0.0s
tuned hpyerparameters :(best parameters)  {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
accuracy : 0.8464285714285713
```

Appendix

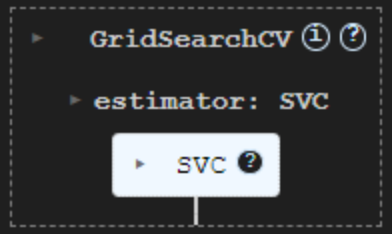
Model Performance:

- **SVM**:

```python
# Crear el objeto GridSearchCV con validación cruzada de 10 pliegues (cv=10)
svm_cv = GridSearchCV(svm, parameters, cv=10)

# Ajustar el objeto GridSearchCV con los datos de entrenamiento
svm_cv.fit(X_train, Y_train)
```

✓ 2.9s




```python
# Mostrar los mejores hiperparámetros y la mejor precisión
print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
print("accuracy :",svm_cv.best_score_)
```

✓ 0.0s

```
tuned hpyerparameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
accuracy : 0.8482142857142856
```

Appendix

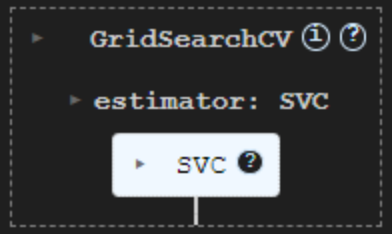
Model Performance:

- **SVM**:

```python
# Crear el objeto GridSearchCV con validación cruzada de 10 pliegues (cv=10)
svm_cv = GridSearchCV(svm, parameters, cv=10)

# Ajustar el objeto GridSearchCV con los datos de entrenamiento
svm_cv.fit(X_train, Y_train)
```

✓ 2.9s




```python
# Mostrar los mejores hiperparámetros y la mejor precisión
print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
print("accuracy :",svm_cv.best_score_)
```

✓ 0.0s

```
tuned hpyerparameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
accuracy : 0.8482142857142856
```

Appendix

- **5. Datasets**

  - spacex_launch_data.csv: Data collected from SpaceX API and web scraping.
  - dataset_part_1.csv: Processed dataset for EDA and visualization.
  - dataset_part_2.csv: Features for machine learning models.

Appendix

- **Additional Notes:**

  - **GitHub URLs:** Testrepo

Thank you!