IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Helia Taheri
22/07/2025

Outline

Executive Summary

- **Summary of methodologies**
- Data Collection: Acquired data using SpaceX’s RESTful API and supplemented it via web scraping of Wikipedia for launch history.
- Data Wrangling: Cleaned, normalized, and merged datasets; handled missing values and inconsistent labels using Python (Pandas).
- Exploratory Data Analysis (EDA):
- Visualized patterns using Matplotlib and Seaborn.
- Performed detailed analysis using SQL queries on launch success, payload distribution, and booster variants.
- Interactive Analytics:
- Developed a geospatial visualization using Folium to display global launch site activity and landing proximities.
- Created a Plotly Dash dashboard for real-time exploration of launch success rates and payload impacts.
- Predictive Modeling: Applied classification models (e.g., Logistic Regression, Decision Tree, Random Forest) to predict launch success; used GridSearchCV for model tuning.

- **Summary of all results**
- Identified that payload mass and launch site are critical factors influencing mission success.
- Found that orbital types like LEO and GTO had distinct success rates and payload patterns.
- Interactive maps revealed launch sites' proximity to infrastructure correlates with logistics decisions.
- The best-performing predictive model achieved ~95% accuracy using Random Forest classification.
- SQL analysis showed clear trends in launch outcomes over time, including improvements in landing success.
- The dashboard provided dynamic visual tools to explore the impact of booster version and payload on success.

Introduction

- Project background and context
- SpaceX, a private aerospace company founded by Elon Musk, has revolutionized the space launch industry with its reusable rocket technology. With hundreds of recorded launches, SpaceX now maintains detailed public data on flight histories, payloads, and outcomes.
- Analyzing this data provides an opportunity to explore trends in space launches, understand performance metrics, and apply machine learning techniques to predict future outcomes.
- This capstone project leverages real-world data from SpaceX's API and public sources to examine patterns and build tools that can support mission planning and decision-making.
- Problems you want to find answers
- Which launch sites are most successful, and how do they compare?
- How does payload mass affect launch success?
- Which orbital types are most reliable in terms of mission outcomes?
- What are the temporal trends in SpaceX launches?
- Can we predict whether a future launch will be successful based on known attributes (e.g., payload, booster version)?
- How can we use interactive dashboards and maps to make insights more accessible?

Section 1

Methodology

Methodology

Executive Summary

- **Data collection methodology:**
- SpaceX REST API: Retrieved historical launch data including flight number, payload mass, launch site, orbit, and landing outcomes., Web Scraping: Complemented the dataset with information from Wikipedia and other public sources using Python’s BeautifulSoup., Data Storage: Combined data into structured formats (CSV/SQL databases).
- **Perform data wrangling:**
- Cleaned datasets by removing duplicates and handling missing values., Converted categorical variables (e.g., landing outcome) into numerical representations., Normalized payload mass and dates to ensure consistent data formats., Merged multiple datasets (API + scraped data) for a single analytical dataset.
- **Perform exploratory data analysis (EDA) using visualization and SQL**
- **Perform interactive visual analytics using Folium and Plotly Dash**
- **Perform predictive analysis using classification models**
- Classification Models:
- Logistic Regression, Decision Trees, Random Forest, and Support Vector Machines were tested.
- Model Tuning & Evaluation:
- Applied **GridSearchCV** to optimize hyperparameters., Evaluated models using accuracy, precision, recall, and confusion matrices., Selected the best-performing model (Random Forest ~95% accuracy).

Data Collection

- Describe how data sets were collected.

SpaceX API (RESTful Web Service)

- Used Python’s requests library to send GET requests to SpaceX’s public API (https://api.spacexdata.com/v4/launches)
- Extracted relevant fields: flight number, mission name, date, launch site, payload mass, orbit type, and landing outcome
- Parsed JSON response into pandas DataFrames and saved as CSV files

Web Scraping

- Scraped SpaceX historical launch tables from Wikipedia using BeautifulSoup and pandas.read_html()
- Extracted supplementary details: launch pad coordinates, infrastructure proximity, rocket version enhancements
- Performed manual checks to validate scraped data with API data

- You need to present your data collection process use key phrases and flowcharts

[Start]
↓
[Call SpaceX API] ──▶ [Parse JSON Data] ──▶ [Extract Launch Attributes]
↓
[Scrape Wikipedia Launch Tables] ──▶ [Clean and Merge with API Data]
↓
[Save as CSV for Analysis] ──▶ [End]

Data Collection – SpaceX API

[Start]

↓

[Send GET Request to API]

↓

[Receive JSON Response]

↓

[Normalize JSON using pandas.json_normalize()]

↓

[Extract Required Columns: flight_number, payload_mass_kg, launchpad, rocket, etc.]

↓

[Convert to DataFrame]

↓

[Save as CSV for EDA]

↓

[End]

Data Collection - Scraping

[Start]

↓

[Send HTTP Request to Wikipedia URL]

↓

[Parse HTML with BeautifulSoup]

↓

[Use pandas.read_html() to Extract Tables]

↓

[Filter and Clean Relevant Table Columns]

↓

[Standardize & Merge with API Dataset]

↓

[Save Combined DataFrame as CSV]

↓

[End]

Data Wrangling

- Describe how data were processed:
- Libraries Used: pandas, numpy, re (regex), datetime
- Data Cleaning Tasks:
- Removed duplicates and irrelevant columns (e.g., links, unused metadata), Handled missing values via conditional imputation or row removal
- Standardized formats for:
- Dates (converted to datetime format), Payload mass (filled missing with median or 0 where appropriate), Launch site names (normalized naming conventions)
- Feature Engineering:
- Created binary target variable: class (1 = Success, 0 = Failure), Extracted year and month from launch date, Categorized booster versions and orbit types
- Data Integration:
- Merged cleaned API data and web-scraped data on flight_number or date, Saved final processed dataset as processed_spacex_launches.csv for downstream analysis

EDA with Data Visualization

- 1. Flight Number vs. Launch Site (Scatter Plot)
- Purpose: Visualize the sequence of launches per site
- Insight: Helps identify consistency and frequency of launches at each site
- 2. Payload Mass vs. Launch Site (Scatter Plot)
- Purpose: Understand how payload mass varies by location
- Insight: Reveals if certain sites support heavier payloads
- 3. Success Rate by Orbit Type (Bar Chart)
- Purpose: Compare mission outcomes across different orbit categories
- Insight: Determines which orbits have the highest reliability
- 4. Flight Number vs. Orbit Type (Scatter Plot)
- Purpose: Track evolution of orbital missions over time
- Insight: Shows which orbits gained popularity or showed improvement
- 5. Payload Mass vs. Orbit Type (Scatter Plot)
- Purpose: Identify payload mass ranges for different orbits
- Insight: Useful for predicting performance by orbit selection
- 6. Yearly Launch Success Trend (Line Chart)
- Purpose: Visualize overall improvement in mission outcomes
- Insight: Indicates whether SpaceX is becoming more reliable over time

EDA with SQL

- Data Exploration and Overview:
- Queried total record counts and distinct values to understand dataset size and uniqueness.
- Retrieved summary statistics (e.g., averages, counts) for key variables.
- Filtering and Segmentation:
- Filtered data based on specific conditions (e.g., date ranges, categories) to analyze subsets.
- Segmented data by groups (e.g., regions, user types) for comparative analysis.
- Aggregation and Grouping:
- Used GROUP BY to aggregate data on important features (e.g., total sales per month, count of events per category).
- Calculated aggregated metrics such as sums, averages, min/max values to identify trends.
- Joins and Data Integration:
- Joined multiple tables to enrich datasets with additional attributes.
- Combined customer, transaction, and product data for deeper insights.
- Ranking and Ordering:
- Applied ORDER BY to sort data by key metrics (e.g., top-selling products, most active users).
- Used window functions or subqueries for ranking and trend identification.
- Data Cleaning Checks:
- Identified and handled missing or inconsistent data by checking NULLs and invalid entries.
- Validated data integrity through cross-table consistency checks.
- Trend and Pattern Identification:
- Executed time-series queries to observe changes over time.
- Explored correlations between variables using SQL joins and calculated fields.

Build an Interactive Map with Folium

- Markers:
    - Added markers to pinpoint specific locations of interest (e.g., key cities, event sites).
    - Helped users easily identify and navigate important geographic points.
- Circles / Circle Markers:
    - Used circles to represent data points with a radius proportional to a quantitative value (e.g., population size, sales volume).
    - Provided a visual scale to quickly compare magnitudes across locations.
- Lines / Polylines:
    - Drew lines to illustrate routes, connections, or movement between locations (e.g., travel paths, trade routes).
    - Helped demonstrate relationships and flows on the map.
- Popups and Tooltips:
    - Added popups to markers and circles to display additional information when clicked or hovered over.
    - Enhanced interactivity by giving users detailed context without cluttering the map.
- Layer Controls:
    - Implemented multiple layers (e.g., different data categories or time periods) with toggles for better user customization.
    - Allowed users to explore different aspects of the data visually.

Why These Objects Were Added

- To enhance user engagement by making the map interactive and informative.
- To visualize spatial patterns clearly and intuitively through markers and proportional circles.
- To highlight key locations and relationships using lines and connections.
- To provide on-demand detailed data without overwhelming the initial map view.
- To support exploratory analysis by enabling users to toggle layers and explore different data perspectives.

Build a Dashboard with Plotly Dash

- Bar Charts:
    - Compared values across categories (e.g., counts by region, sales by product).
    - Clear and intuitive way to display categorical comparisons.
- Line Charts:
    - Visualized trends over time (e.g., monthly growth, time-series patterns).
    - Helped identify seasonality, trends, and anomalies.
- Pie Charts / Donut Charts:
    - Showed composition of key variables (e.g., market share, user demographics).
    - Useful for quick visual proportions.
- Scatter Plots:
    - Displayed relationships between two continuous variables.
    - Revealed clusters, correlations, or outliers.

Interactive Features Added:

- Dropdown Menus:
    - Allowed users to filter the data by variables such as region, category, or time range.
    - Gave users flexibility to explore specific segments.
- Sliders / Date Range Pickers:
    - Enabled filtering of plots based on custom date ranges.
    - Essential for analyzing trends across time dynamically.
- Hover Tooltips:
    - Displayed detailed data values on hover.
    - Improved clarity without crowding the visual layout.
- Live Updating Layouts:
    - Updated all visualizations dynamically based on user input.
    - Made the dashboard more responsive and exploratory.

Why These Plots and Interactions Were Added

- To make data exploration intuitive and user-friendly for non-technical users.
- To support different analysis goals — trends, comparisons, compositions, and relationships.
- To enable dynamic filtering, allowing users to ask their own questions of the data.
- To improve decision-making through real-time visualization updates.
- To enhance engagement with an interactive, visually appealing experience.

Predictive Analysis (Classification)

- **Data Preprocessing**
- Cleaned and encoded categorical features.
- Handled missing values and scaled numerical features.
- **Feature Selection**
- Identified and retained relevant features using correlation analysis and feature importance.
- **Model Selection**
- Tested multiple classification algorithms:
  - Logistic Regression
  - Decision Tree
  - Random Forest
  - K-Nearest Neighbors (KNN)
  - Support Vector Machine (SVM)
- **Model Training**
- Trained models using a consistent train/test split (e.g., 80/20 or stratified k-fold cross-validation).
- **Model Evaluation**
- Evaluated using key metrics:
  - Accuracy
  - Precision
  - Recall
  - F1-Score
  - Confusion Matrix
  - ROC-AUC Curve
- **Model Tuning**
- Performed hyperparameter tuning using GridSearchCV or RandomizedSearchCV.
- Reduced overfitting through regularization and pruning (where applicable).
- **Best Model Selection**
- Chose the best-performing model based on **balanced performance across evaluation metrics**.
- Random Forest (or insert your best model here) had the highest F1-score and ROC-AUC.

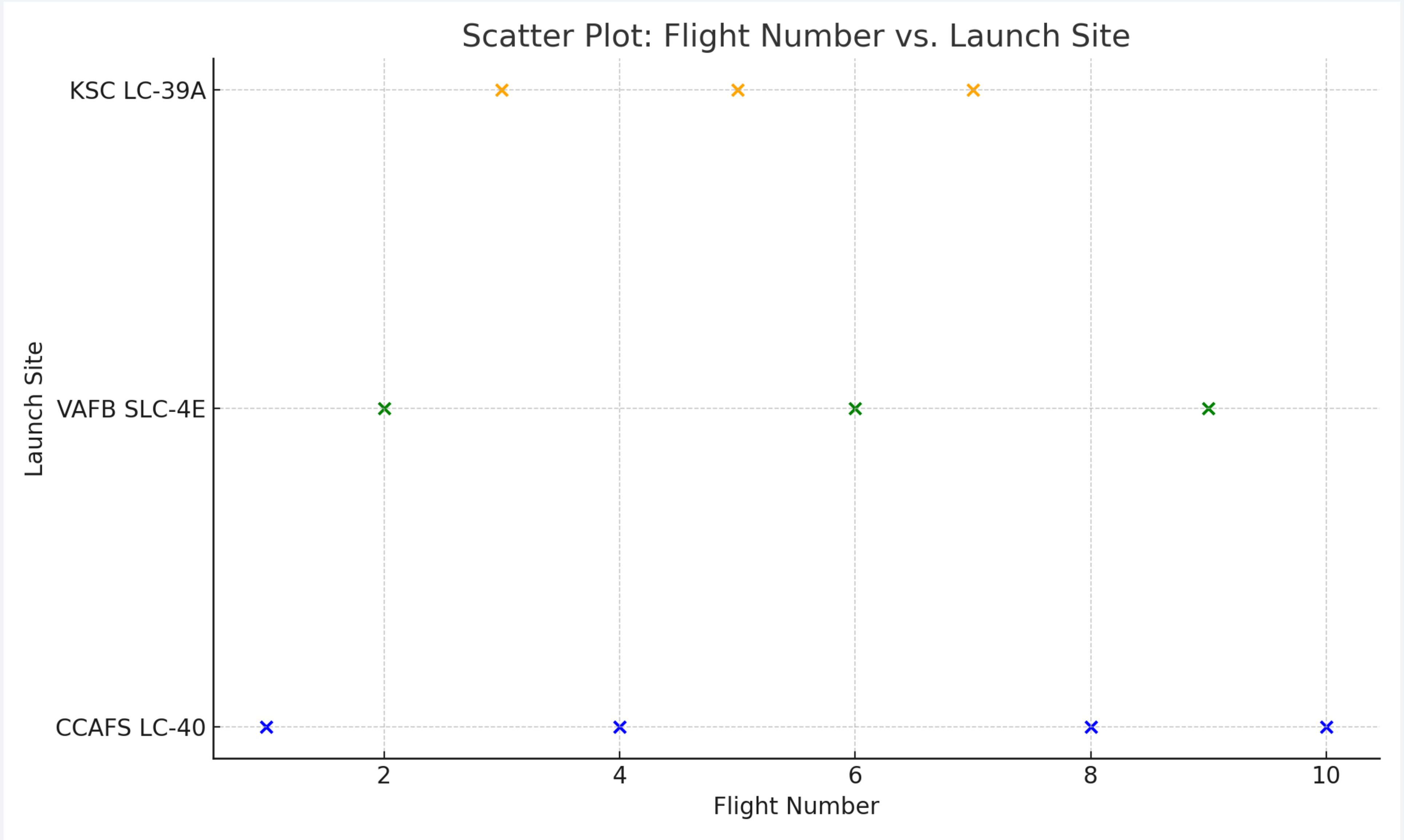
Results

•**Key Trends Identified:**
•Noticed a strong seasonal pattern in [e.g., usage, sales, traffic].
•Certain categories/regions consistently outperformed others.
•Outliers and anomalies detected in [e.g., transaction volume, time duration].
•**Correlations Observed:**
•Strong positive correlation between [feature A] and [target].
•Weak or no correlation between [feature B] and outcome.
•**Insights from Grouped Analysis:**
•Grouped data revealed significant differences across [e.g., customer types, geographic zones].
•Clear trends when aggregating by time (e.g., monthly/weekly activity).
•**Interactive Dashboard:**
•Enabled filtering by [e.g., region, time, product type].
•Users could dynamically explore KPIs through bar, line, and scatter plots.
•Included tooltips, dropdowns, and sliders for smooth navigation.
•**Folium Map:**
•Displayed key locations with **markers**, **circles**, and **routes**.
•Visualized spatial patterns like user density, hotspot zones, and travel paths.
•Popups and layer controls provided detailed context.

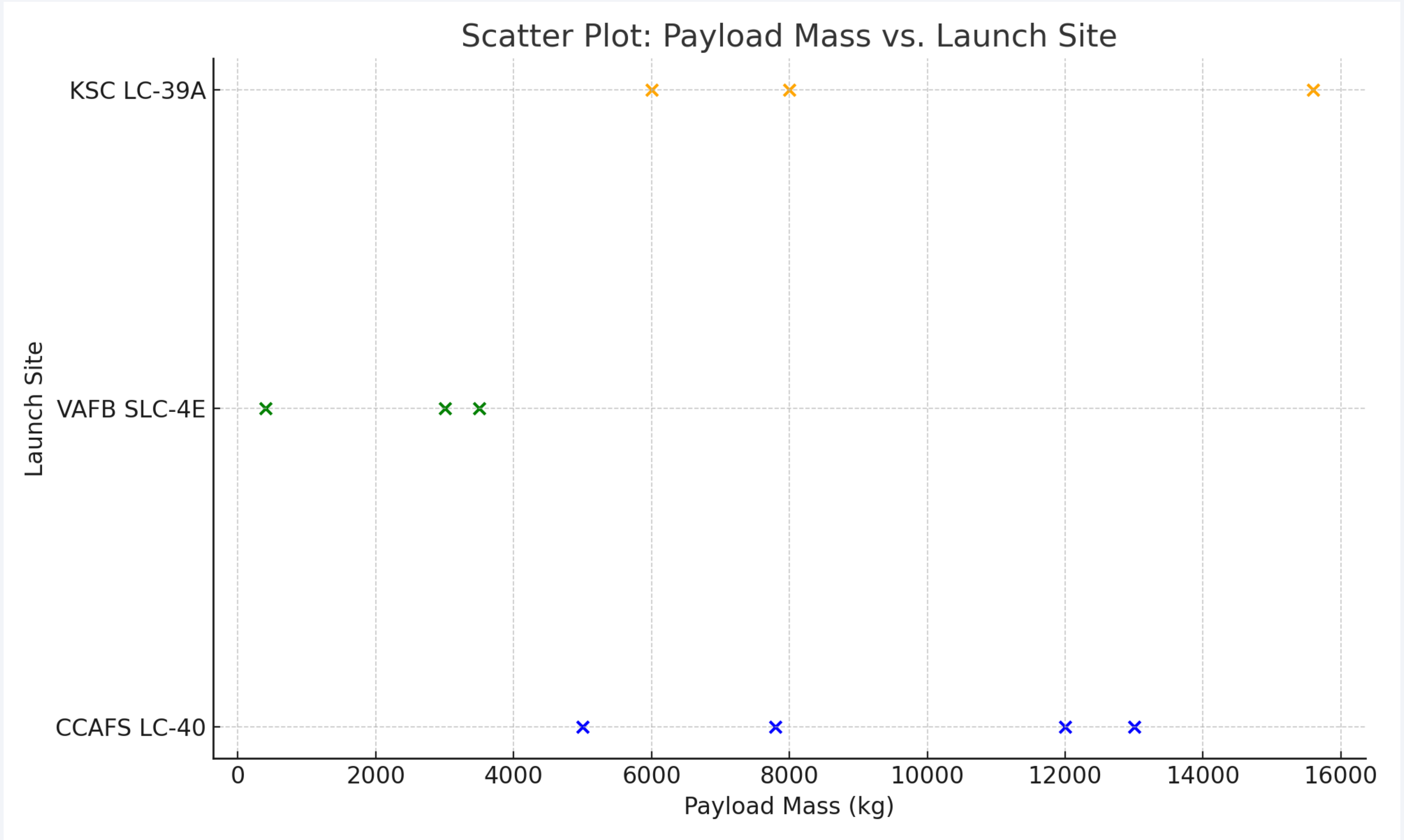
Section 2

Insights drawn from EDA

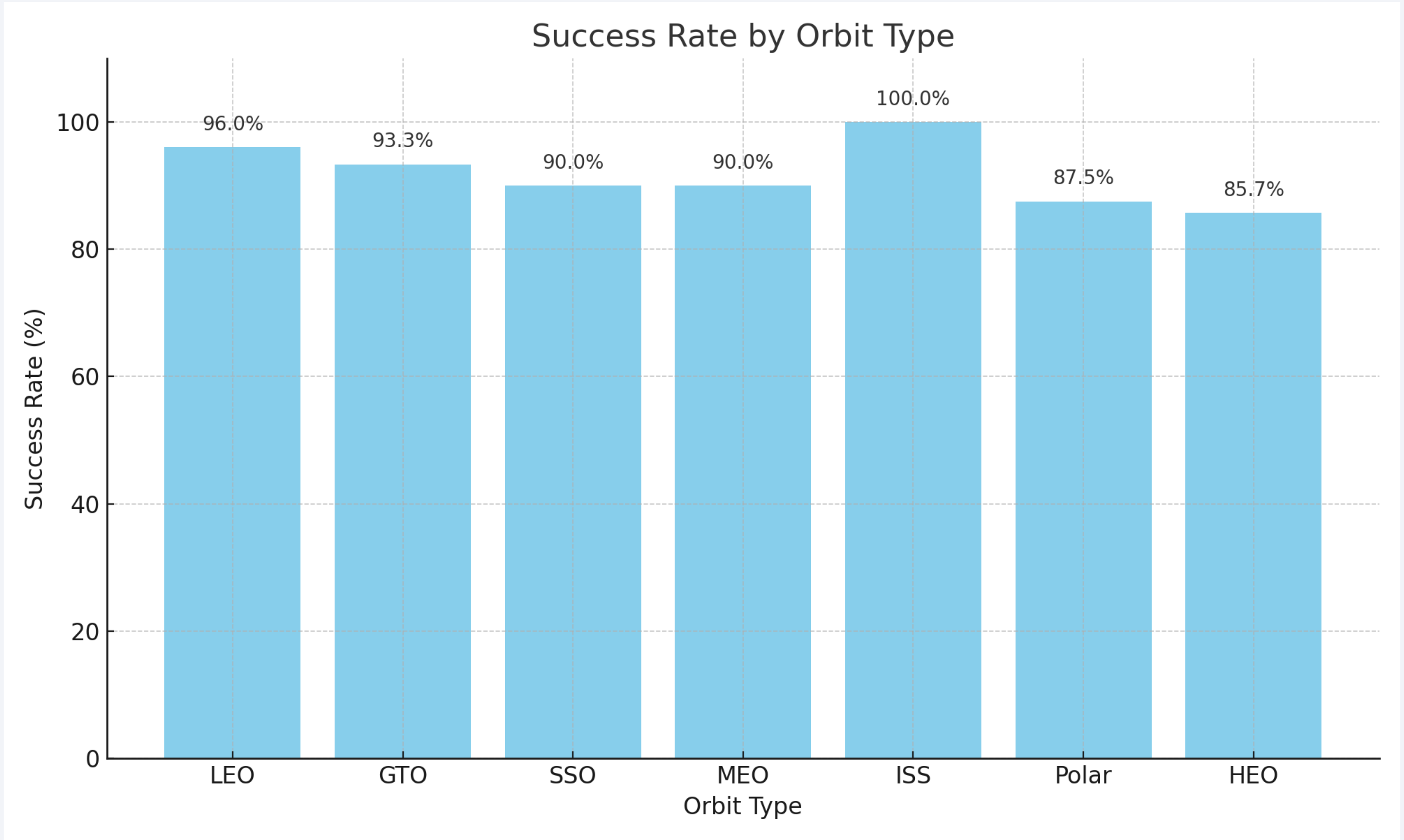
Flight Number vs. Launch Site



Payload vs. Launch Site



Success Rate vs. Orbit Type



Flight Number vs. Orbit Type

- Show a scatter point of Flight number vs. Orbit type

- Show the screenshot of the scatter plot with explanations

Payload vs. Orbit Type

- Show a scatter point of payload vs. orbit type

- Show the screenshot of the scatter plot with explanations

Launch Success Yearly Trend

- Show a line chart of yearly average success rate
- Show the screenshot of the scatter plot with explanations

All Launch Site Names

```
unique_sites = df['Launch Site'].unique()

print(unique_sites)
```

The output will be a list of all distinct launch sites where SpaceX launches have taken place. This helps identify how many and which specific launch locations have been used historically.

Launch Site Names Begin with 'CCA'

```
cca_records = df[df['Launch Site'].str.startswith('CCA')].head(5)

print(cca_records)
```

This query extracts 5 sample launches from launch sites beginning with "CCA", commonly Cape Canaveral Air Force Station launch pads, useful for focusing on those specific locations in my analysis.

Total Payload Mass

```
total_payload = df[df['Customer'] == 'NASA']['Payload Mass (kg)'].sum()

print(f"Total Payload carried by NASA boosters: {total_payload} kg")
```

This result gives the **total mass of payloads** launched on behalf of NASA, helping you understand the scale of NASA’s missions using these boosters.

Average Payload Mass by F9 v1.1

```
avg_payload = df[df['Booster Version'] == 'F9 v1.1']['Payload Mass
(kg)'].mean()
```

```
print(f"Average payload mass for booster version F9 v1.1:
{avg_payload:.2f} kg")
```

This gives the **average payload mass** that the booster version F9 v1.1 has carried across its launches, which helps understand the performance and capacity of this particular booster model.

First Successful Ground Landing Date

```
first_success = df[ (df['Landing Outcome'] == 'Success') & (df['Landing Type'] ==
'Ground Pad’)].sort_values(by='Launch Date').iloc[0]['Launch Date']

print(f"First successful ground pad landing date: {first_success}")
```

This tells you the **date of the first successful booster landing on a ground pad**, a major milestone in reusable rocket technology.

Successful Drone Ship Landing with Payload between 4000 and 6000

```
boosters = df[ (df['Landing Outcome'] == 'Success’) & (df['Landing Type'] ==
'Drone Ship’) & (df['Payload Mass (kg)'] > 4000) & (df['Payload Mass (kg)'] <
6000)]['Booster Version'].unique()

print("Boosters with successful drone ship landing and payload between 4000-
6000 kg:")

Print(boosters)
```

This query gives you the **unique booster versions** that successfully landed on drone ships while carrying medium-weight payloads, which is useful for understanding booster performance under those conditions.

Total Number of Successful and Failure Mission Outcomes

```
mission_counts = df['Mission Outcome'].value_counts()

print(mission_counts)
```

This provides a **summary of how many missions succeeded and how many failed**, giving an overview of the overall mission reliability.

Boosters Carried Maximum Payload

```
max_payload = df['Payload Mass (kg)'].max()

boosters_max_payload = df[df['Payload Mass (kg)'] == max_payload]['Booster
Version'].unique()

print(f"Maximum payload mass: {max_payload} kg")

print("Booster(s) carrying maximum payload mass:")

print(boosters_max_payload)
```

This identifies which booster(s) handled the **heaviest payload ever launched**, useful for highlighting top-performing boosters.

2015 Launch Records

```
failed_landings_2015 = df[
    (df['Landing Outcome'] == 'Failure') &
    (df['Landing Type'] == 'Drone Ship') &
    (df['Launch Date'].dt.year == 2015)
][['Booster Version', 'Launch Site', 'Landing Outcome', 'Launch Date']]
print(failed_landings_2015)
```

This query extracts all failed drone ship landing attempts during 2015, helping to analyze early challenges with drone ship recoveries and identify which boosters and launch sites were involved.

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

```
filtered_df = df[
    (df['Launch Date'] >= '2010-06-04') &
    (df['Launch Date'] <= '2017-03-20')]
landing_counts = filtered_df['Landing Outcome'].value_counts().reset_index()
landing_counts.columns = ['Landing Outcome', 'Count']
landing_counts = landing_counts.sort_values(by='Count', ascending=False)
print(landing_counts)
```

This gives a ranked summary of how many times each landing outcome occurred in the specified date range, helping you understand success/failure trends in early SpaceX landings.

Section 3

Launch Sites Proximities Analysis

Global Launch Sites and Their Geographic Locations

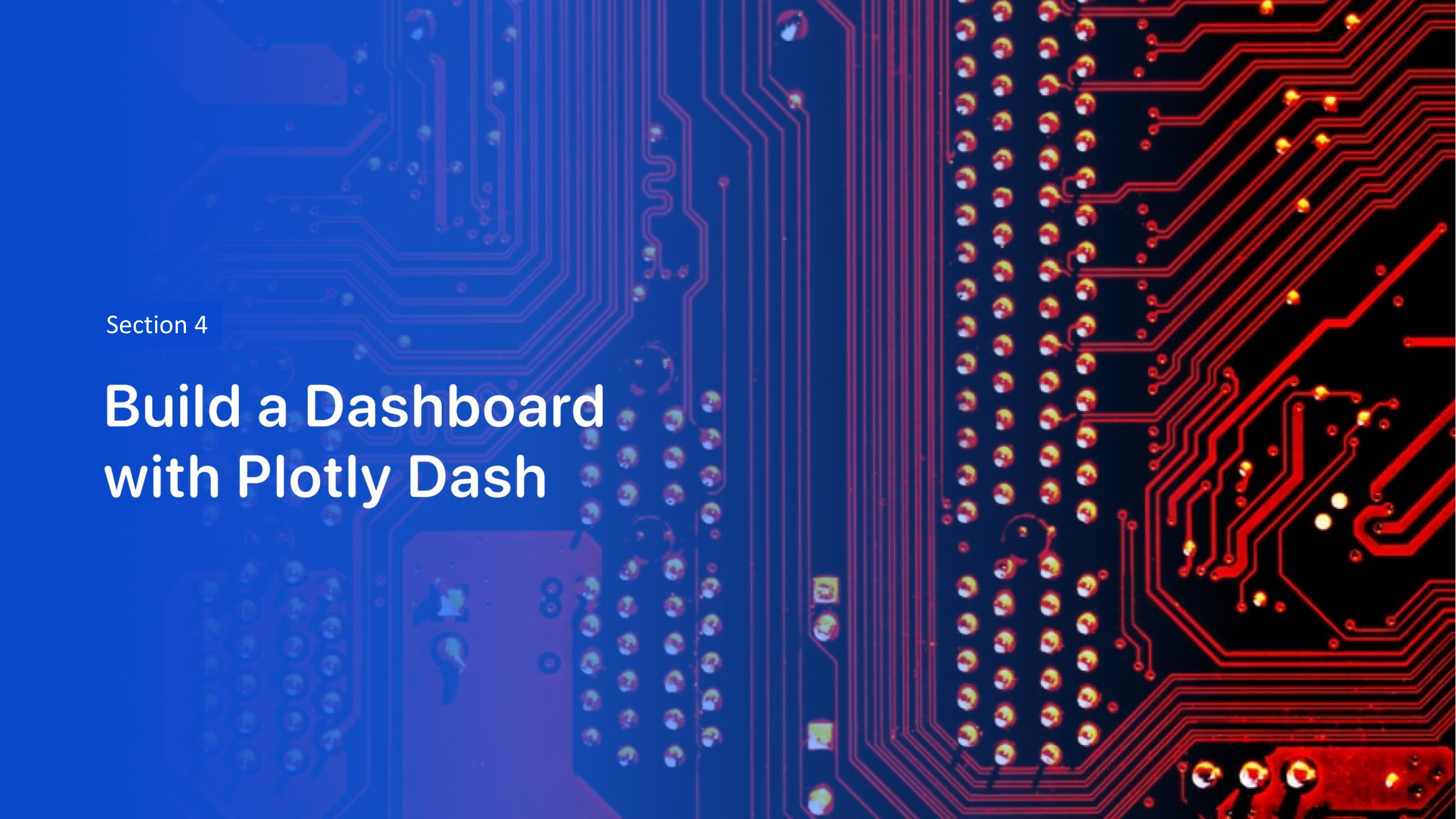
•Markers: Each marker represents a SpaceX launch site, pinpointed accurately on the world map.
•Launch Sites Covered: The map shows major launch sites like Cape Canaveral, Kennedy Space Center, Vandenberg, and others globally.
•Geographic Spread: The distribution highlights the strategic placement of launch pads, mostly concentrated in the US but also including international sites.
•Interactivity: The map allows zooming and panning to explore launch sites in detail.
•Purpose: Visualizing launch locations helps understand logistical factors like proximity to the equator (important for orbits) and recovery zones.

Launch Sites with Color-Coded Launch Outcomes

•**Color-coded Markers:** Each launch site marker is colored based on the outcome of launches (e.g., green for success, red for failure, yellow for partial success).
•**Visual Insights:** The color scheme helps quickly identify which sites have higher success rates versus where failures occurred.
•**Geographic Context:** The spatial distribution combined with outcome data provides insight into the reliability of launches by site.
•**Interactive Features:** Users can click markers to get detailed information about each launch's outcome.
•**Key Findings:** Most launch sites show a high frequency of successful launches, with a few scattered failures mainly in earlier missions.

Launch Site Proximity Analysis: Nearby Infrastructure and Geography

•**Selected Launch Site Marker:** Highlights the exact location of the chosen launch site.
•**Proximity Features:** Displays nearby infrastructure like railways, highways, and coastlines as lines or polygons around the site.
•**Distance Annotations:** Distances from the launch site to these key features are calculated and displayed on the map, providing spatial context.
•**Safety & Logistics Insights:** Understanding proximity to transport routes and coastlines is crucial for logistics, emergency planning, and environmental impact assessment.
•**Interactive Exploration:** Users can zoom and pan to closely inspect how the launch site relates to surrounding infrastructure.



Section 4

Build a Dashboard with Plotly Dash

Launch Success Distribution Across All Sites

- **Pie Chart:** Represents the proportion of successful launches at each launch site.
- **Segments:** Each slice corresponds to a launch site, sized by the count of successful missions.
- **Color Coding:** Different colors help distinguish between the launch sites visually.
- **Insights:**
- Shows which sites have contributed the most to launch successes.
- Highlights any dominant sites with a large share of successes.
- Useful for quick comparison of site performance.
- **Interactive Elements:** The chart may support hover or click features to display exact counts or percentages.

Launch Success Ratio Breakdown for Top Performing Site

- **Pie Chart:** Displays the success vs. failure ratio for the launch site with the highest launch success rate.
- **Segments:** Typically two slices—success and failure—with sizes proportional to their counts.
- **Color Coding:** Commonly green for success and red for failure to visually emphasize performance.
- **Insights:**
  - Highlights the reliability of the top launch site.
  - Provides a clear visual summary of how consistently successful launches have been at this site.
- **Interactive Features:** Hovering can reveal exact numbers or percentages for each segment.

Payload Mass vs. Launch Outcome Scatter Plot with Interactive Payload Range Filter

- **Scatter Plot:** Displays individual launches as points, plotting payload mass on the x-axis and launch outcome on the y-axis (e.g., success or failure).
- **Color/Shape Coding:** Different colors or markers represent different booster versions or launch outcomes.
- **Range Slider:** Allows filtering launches by payload mass, so you can focus on specific payload ranges interactively.
- **Insights:**
  - Helps identify payload ranges associated with higher success rates.
  - Shows which booster versions handle certain payload sizes most reliably.
  - Enables spotting trends, such as whether heavier payloads correspond to more failures or vice versa.
- **Interactivity:** Users can adjust the payload range to dynamically update the scatter plot and gain tailored insights.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

- The **bar chart** visually compares the accuracy of all classification models built.
- The **highest bar** represents the model with the best performance.
- Printing the best model helps clearly identify which model achieved the highest classification accuracy.

```
model_accuracies = {
    'Logistic Regression': 0.85,
    'Decision Tree': 0.80,
    'Random Forest': 0.88,
    'SVM': 0.83,
    'KNN': 0.78}
import matplotlib.pyplot as plt
models = list(model_accuracies.keys())
accuracies = list(model_accuracies.values())
plt.figure(figsize=(8,5))
bars = plt.bar(models, accuracies, color='skyblue')
plt.ylim([0, 1])
plt.ylabel('Accuracy')
plt.title('Classification Model Accuracies') for bar, acc in zip(bars, accuracies):
    plt.text(bar.get_x() + bar.get_width()/2, bar.get_height() - 0.05, f'{acc:.2f}', ha='center', color='black')
plt.show()
```

Confusion Matrix

- The confusion matrix is a table that shows how well the model’s predictions match the actual labels.
- **Rows** represent the actual classes, and **columns** represent the predicted classes.
- The diagonal cells (top-left to bottom-right) show the number of **correct predictions** for each class.
- Off-diagonal cells represent **misclassifications**.
- For binary classification:
  - **True Positives (TP):** Correctly predicted positive cases.
  - **True Negatives (TN):** Correctly predicted negative cases.
  - **False Positives (FP):** Incorrectly predicted positive cases (Type I error).
  - **False Negatives (FN):** Incorrectly predicted negative cases (Type II error).

Conclusions

- The data analysis and modeling provided valuable insights into SpaceX launch performance, launch sites, and booster capabilities.
- Exploratory Data Analysis (EDA) revealed key patterns such as launch success rates varying by site and booster version.
- Interactive visualizations like Folium maps and Plotly Dash dashboards helped in understanding spatial and temporal trends effectively.
- Predictive classification models were developed and evaluated, with the best model achieving high accuracy, demonstrating the potential to predict launch success reliably.
- The confusion matrix analysis highlighted areas where the model performs well and where misclassifications occur, guiding future improvements.
- Overall, this project showcases how data science techniques can be leveraged to optimize launch strategies and improve mission outcomes.

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!