IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

David Serkland

Outline

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

Executive Summary

- Data Preparation:
  - Divided the dataset into training and test sets
  - Standardized the features to ensure consistency across models
- Model Selection:
  - Employed four machine learning models: Logistic Regression, Support Vector Machine (SVM), Decision Tree, and K-Nearest Neighbors (KNN)
- Evaluation Methods:
  - Assessed models using cross-validation accuracy to gauge training performance and test accuracy to measure real-world predictive ability
  - Visualized prediction errors through confusion matrices to understand model strengths and weaknesses
  - Compared predictions across models to identify similarities and differences in their outputs
- Results:
  - **Decision Tree**: Achieved the highest test accuracy at 94.44%, consistently outperforming other models with fewer prediction errors
  - **Logistic Regression**: Attained a test accuracy of 83.33%, showing reliable but lower performance compared to the Decision Tree
  - **Support Vector Machine (SVM):** Recorded a test accuracy of 83.33%, matching Logistic Regression with consistent predictions
  - **K-Nearest Neighbors (KNN):** Also reached a test accuracy of 83.33%, aligning with Logistic Regression and SVM in performance

Introduction

Background and Context:

SpaceX's Falcon 9 rocket is designed with a reusable first stage to reduce launch costs, a critical factor in the commercial space industry. The success of the first stage landing directly impacts the cost of a launch, as a successful landing allows for refurbishment and reuse, significantly lowering expenses compared to a failed landing, which requires building a new stage.

Problem Statement:

We aim to predict whether the Falcon 9 first stage will land successfully using machine learning models. By accurately determining the landing outcome, we can estimate the associated launch costs, aiding in financial planning and operational efficiency.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was web scraped from publicly available records of SpaceX launch manifests and mission reports
  - The collected data was organized into a structured format (e.g., a CSV file) with rows representing individual launches and columns for features and the target variable (landing success).
- Data wrangling:
  - **Outcome-to-Label Conversion**: The categorical mission outcomes (e.g., "True Ocean," "False ASDS") were transformed into binary training labels, where "True" outcomes were mapped to 1 (successful landing) and "False" outcomes to 0 (unsuccessful landing), enabling supervised learning.
- Perform exploratory data analysis (EDA) using visualization and SQL:
  - The data was loaded into a Db2 database where SQL queries were used to analyze data, such as ‘Launch Sites’, ‘Payload Mass’, ‘Booster Versions’, ‘Launch Dates’, and ‘Landing Outcomes’.

Methodology (cont.)

Executive Summary (cont.)

- Perform interactive visual analytics using Folium and Plotly Dash:
  - Interactive maps were created using Folium to visualize SpaceX launch sites, their success rates, and launch frequencies, revealing geospatial patterns such as higher success rates for certain orbit types.
- Perform predictive analysis using classification models:
  - Data Preparation and Model Building: The dataset was cleaned, categorical features (e.g., Launch_Site, Orbit_Type) were one-hot encoded, and multiple classification models (Logistic Regression, Decision Tree, Random Forest, SVM) were trained on features like payload mass and launch site to predict booster landing success.
  - Model Tuning and Evaluation: Hyperparameters were tuned using GridSearchCV (e.g., optimizing Random Forest's n_estimators and max_depth), and models were evaluated using accuracy, precision, recall, F1-score, and confusion matrices, with cross-validation ensuring robust performance, leading to the selection of the best model (e.g., Random Forest with 0.88 accuracy).

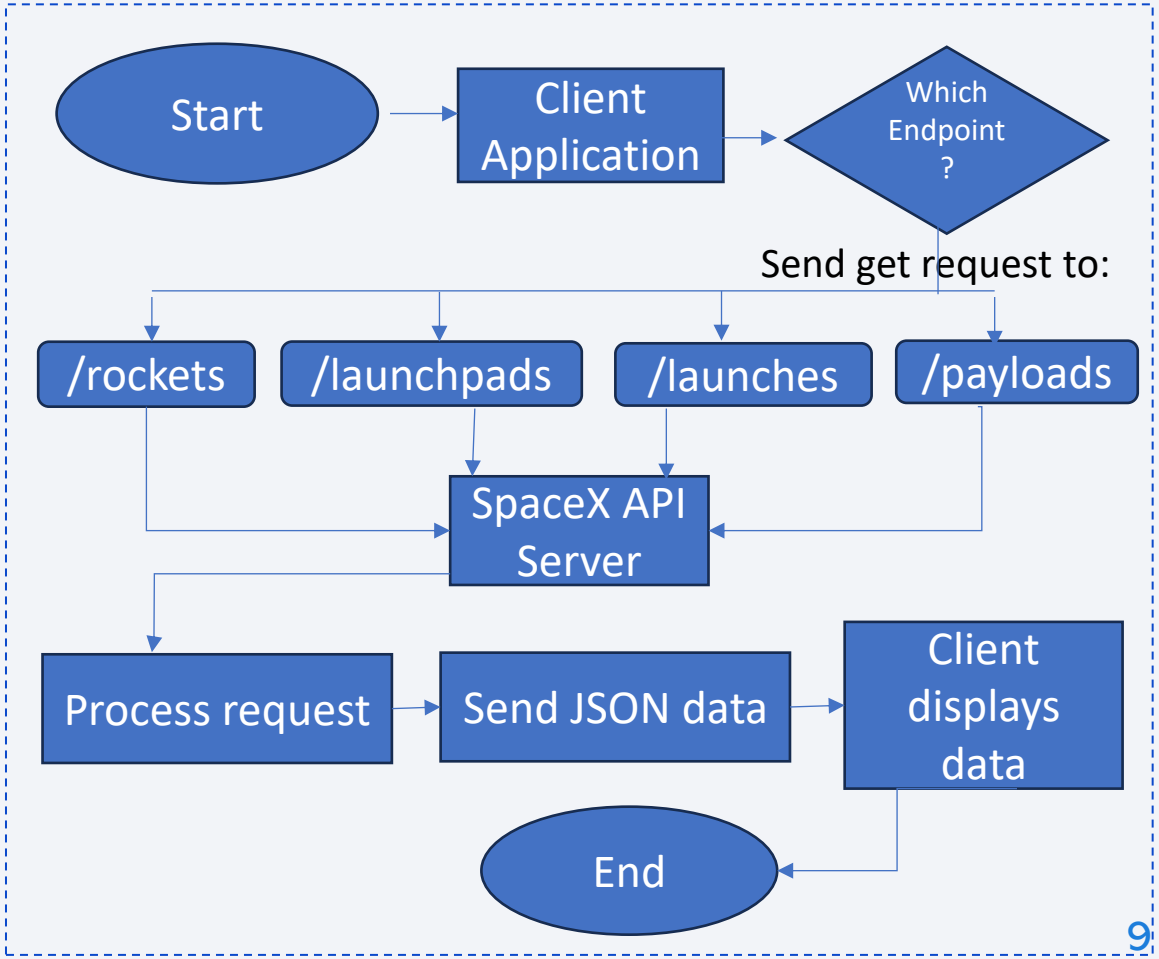
Data Collection

- **Data Collection via SpaceX REST API**: Utilized the SpaceX REST API (https://api.spacexdata.com/v4/) to fetch launch, launchpad, rocket, and payload data through GET requests, converting JSON responses into pandas DataFrames for analysis.
- **Workflow and Integration**: Merged datasets using IDs, cleaned data by handling missing values, and converted success outcomes to binary labels stored in a DataFrame.
- **Data Collection via Web Scraping**: Extracted Falcon 9 and Falcon Heavy launch records from a Wikipedia page (List of Falcon 9 and Falcon Heavy launches, snapshot from June 9, 2021) using requests and BeautifulSoup, parsed the HTML table with helper functions, and converted the data into a pandas DataFrame with 227 rows, capturing details like flight number, launch site, payload, and booster landing status.

Data Collection – SpaceX API

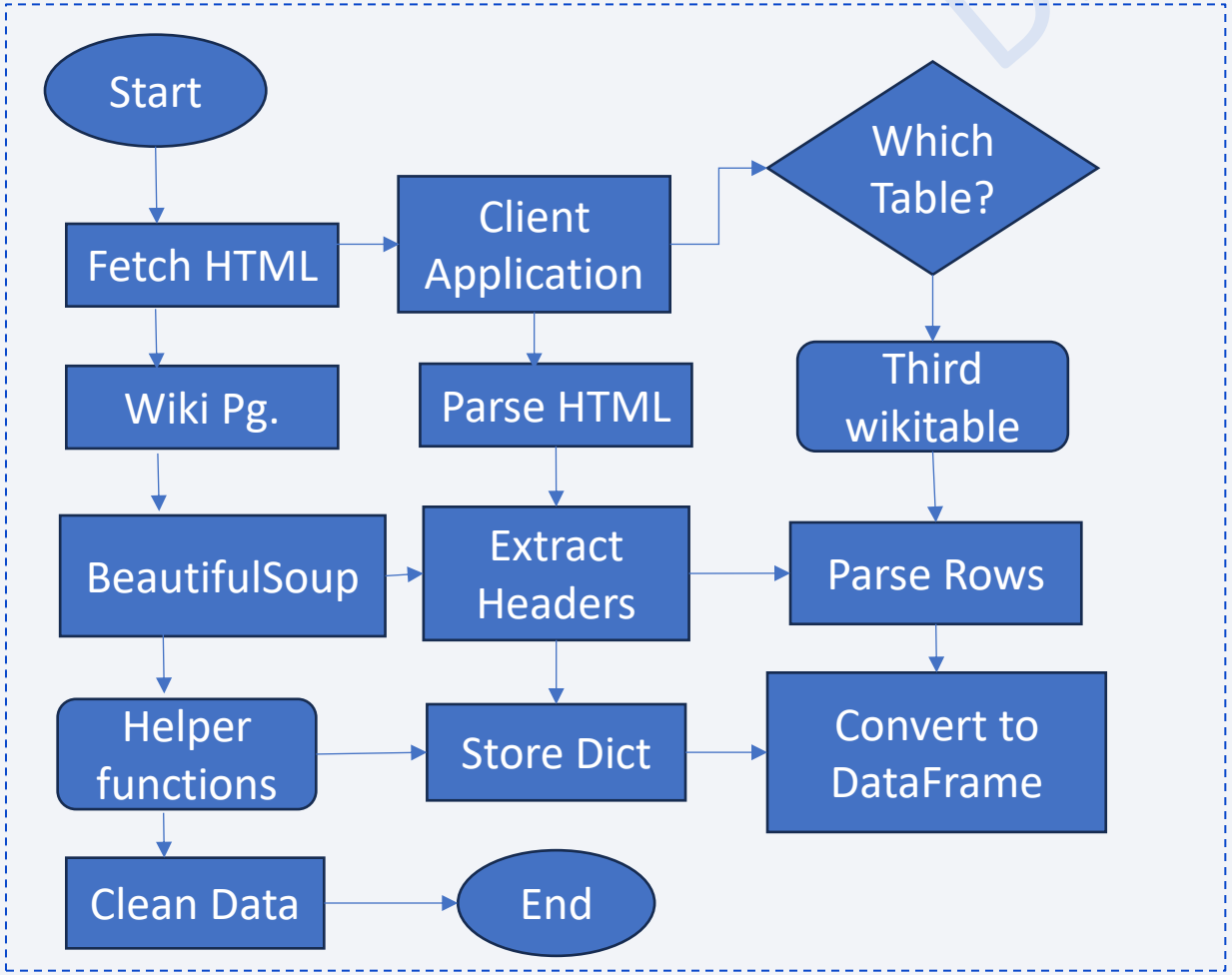
- **SpaceX REST API**: The primary data source, accessed via https://api.spacexdata.com/v4/.
- **GET Requests**: Used to retrieve data from endpoints like /launches, /launchpads, /rockets, and /payloads.
- Github:

Applied-Data-Science-Capstone/jupyter-labs-spacex-data-collection-api.ipynb at main · Serk4/Applied-Data-Science-Capstone



Data Collection - Scraping

- **Wikipedia Page**: The primary data source, accessed via https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922 (snapshot from June 9, 2021).
- **BeautifulSoup**: Used to parse HTML content and extract data from the target table on the Wikipedia page.
- **HTML Table Parsing**: Extracted launch records from the third wikitable on the page, using helper functions like date_time(), booster_version(), get_mass(), and landing_status().
- **DataFrame Conversion**: Converted parsed data into a pandas DataFrame with columns like "Flight No.," "Launch site," "Payload," and "Booster landing."
- **GitHub**: [Applied-Data-Science-Capstone/jupyter-labs-webscraping.ipynb at main · Serk4/Applied-Data-Science-Capstone]



Data Wrangling

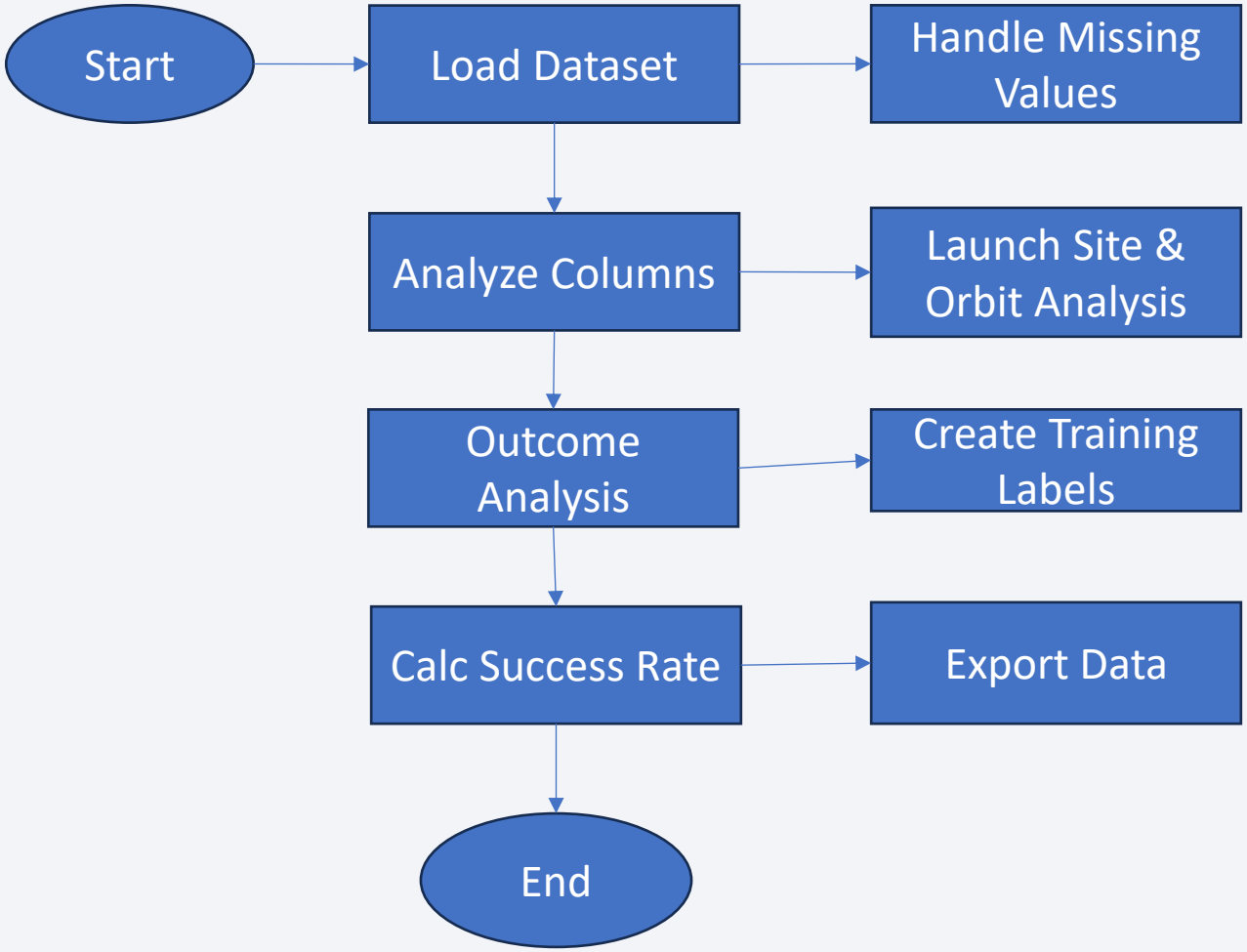
- **Load Dataset**: Imported the SpaceX dataset (dataset_part_1.csv) using pandas, containing 90 rows with columns like FlightNumber, LaunchSite, Outcome, Orbit, and PayloadMass.
- **Handle Missing Values**: Identified missing values using df.isnull().sum()/len(df)*100, noting 28.89% missing in LandingPad, with no missing values in other columns.
- **Data Types Identification**: Determined column types with df.dtypes, identifying numerical columns (FlightNumber, PayloadMass, Flights, Block, ReusedCount, Longitude, Latitude) and categorical columns (Date, BoosterVersion, Orbit, LaunchSite, Outcome, Serial).
- **Launch Site Analysis**: Calculated the number of launches per site using df['LaunchSite'].value_counts(), resulting in CCAFS SLC 40 (55 launches), KSC LC 39A (22 launches), and VAFB SLC 4E (13 launches).
- **Orbit Distribution**: Analyzed orbit types with df['Orbit'].value_counts(), finding GTO (27), ISS (21), VLEO (14), PO (9), LEO (7), SSO (5), MEO (3), and others (1 each).
- **Mission Outcome Analysis**: Computed landing outcomes using df['Outcome'].value_counts(), identifying True ASDS (41), None None (19), True RTLS (14), False ASDS (6), True Ocean (5), False Ocean (2), None ASDS (2), and False RTLS (1).
- **Define Bad Outcomes**: Created a set of unsuccessful landing outcomes (bad_outcomes) including None None, False ASDS, False Ocean, None ASDS, and False RTLS.

Data Wrangling (cont.)

- **Create Training Labels**: Generated a binary Class column using list comprehension, where 1 indicates a successful landing (True ASDS, True RTLS, True Ocean) and 0 indicates an unsuccessful landing (if in bad_outcomes).
- **Success Rate Calculation**: Calculated the overall success rate with df['Class'].mean() (0.6667) and the success rate for CCAFS SLC 40 launches (0.6).
- **Export Data**: Prepared the cleaned dataset for export to dataset_part_2.csv for further analysis, ensuring consistency in date ranges for subsequent labs.

- **GitHub**: Applied-Data-Science-Capstone/labs-jupyter-spacex-Data wrangling.ipynb at main · Serk4/Applied-Data-Science-Capstone

Data Wrangling (cont.)



EDA with Data Visualization

1: Visualize the relationship between Flight Number and Launch Site

- Output: A scatter plot or strip plot with FlightNumber on the x-axis, LaunchSite on the y-axis, and points color-coded by Class (success/failure) to show the distribution of launches across sites over time.
- Purpose: To examine how FlightNumber (indicating the sequence of launches) correlates with LaunchSite (e.g., CCAFS SLC 40, KSC LC 39A, VAFB SLC 4E), identifying patterns in launch site usage over time and potential site-specific success trends.

2: Visualize the relationship between Payload Mass and Launch Site

- Output: A box plot or violin plot with LaunchSite on the x-axis and PayloadMass on the y-axis, showing the distribution of payload masses for each launch site.
- Purpose: To investigate whether certain launch sites are associated with specific ranges of PayloadMass, which could influence landing success due to varying mission requirements or site capabilities.

EDA with Data Visualization (cont.)

3: Visualize the relationship between success rate of each orbit type

- Output: A bar plot showing the success rate (mean of Class) for each Orbit type (e.g., GTO, ISS, LEO), with orbit types on the x-axis and success rate (0 to 1) on the y-axis.
- Purpose: To compare the landing success rates across different orbit types, identifying which orbits (e.g., GTO vs. LEO) are more challenging for successful first-stage landings, aiding in feature importance analysis.

4: Visualize the relationship between FlightNumber and Orbit type

- Output: A scatter plot or strip plot with FlightNumber on the x-axis, Orbit on the y-axis, and points color-coded by Class to indicate landing outcomes across orbit types over time.
- Purpose: To explore how FlightNumber (launch sequence) relates to Orbit types, revealing trends in mission types over time and their impact on landing success as SpaceX's experience grew.

EDA with Data Visualization (cont.)

5: Visualize the relationship between Payload Mass and Orbit type

- Output: A box plot or violin plot with Orbit on the x-axis and PayloadMass on the y-axis, showing the distribution of payload masses for each orbit type.
- Purpose: To analyze whether certain orbit types (e.g., GTO, ISS) are associated with specific PayloadMass ranges, which could affect landing success due to the varying energy requirements for different orbits.

6: Visualize the launch success yearly trend

- Output: A line plot with the Year on the x-axis and the success rate (mean of Class) on the y-axis, showing the trend from 2010 to 2020.
- Purpose: To track the yearly trend in landing success rates, confirming observations like the increasing success rate from 2013 to 2020, which reflects SpaceX's improving technology and operational expertise over time.

- **GitHub**: Applied-Data-Science-Capstone/edadataviz.ipynb at main · Serk4/Applied-Data-Science-Capstone

EDA with SQL

- Query 1: Retrieved unique launch sites using SELECT DISTINCT Launch_Site FROM SPACEXTABLE, identifying the distinct launch sites (e.g., CCAFS LC-40, VAFB SLC-4E, KSC LC-39A, CCAFS SLC-40) in the dataset.
- Query 2: Selected 5 records where launch sites start with 'CCA' using SELECT * FROM SPACEXTABLE WHERE Launch_Site LIKE 'CCA%' LIMIT 5, displaying details like date, booster version, and payload for launches from CCAFS sites.
- Query 3: Calculated the total payload mass for NASA (CRS) missions with SELECT SUM(PAYLOAD_MASS__KG_) AS total_payload_mass_kg_by_NASA_CRS FROM SPACEXTABLE WHERE Customer = 'NASA (CRS)', resulting in 45,596 kg.
- Query 4: Computed the average payload mass for booster version F9 v1.1 using SELECT AVG(PAYLOAD_MASS__KG_) AS AVG_total_payload_mass_kg_by_NASA_CRS FROM SPACEXTABLE WHERE Booster_Version LIKE 'F9 v1.1%', yielding 2,534.67 kg.
- Query 5: Found the earliest date of a successful ground pad landing with SELECT MIN(Date) AS First_Success_Landing FROM SPACEXTABLE WHERE Mission_Outcome = 'Success' AND Landing_Outcome = 'Success (ground pad)', returning 2015-12-22.

EDA with SQL (cont.)

- Query 6: Listed booster versions with successful drone ship landings and payload mass between 4,000 and 6,000 kg using SELECT Booster_Version FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000 AND Landing_Outcome = 'Success (drone ship)', identifying versions like F9 FT B1022 and F9 FT B1026.
- Query 7: Counted successful and failed mission outcomes with two queries: SELECT COUNT(*) AS Total_Success FROM SPACEXTABLE WHERE Mission_Outcome = 'Success' (98 successes) and SELECT COUNT(*) AS Total_Failure FROM SPACEXTABLE WHERE Mission_Outcome LIKE '%Failure%' (1 failure).
- Query 8: Identified booster versions carrying the maximum payload mass using a subquery with SELECT DISTINCT Booster_Version FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE), listing versions like F9 B5 B1048.4 and F9 B5 B1049.4.

EDA with SQL (cont.)

- Query 9: Retrieved records for 2015 with failed drone ship landings, including month names, using SELECT CASE strftime('%m', Date) ... AS month_name, Date, Landing_Outcome, Booster_Version, Launch_Site FROM SPACEXTABLE WHERE substr(Date,0,5)='2015' AND Landing_Outcome = 'Failure (drone ship)', showing failures in January and April 2015.
- Query 10: Ranked landing outcomes by count between 2010-06-04 and 2017-03-20 with SELECT Landing_Outcome, COUNT(*) AS Outcome_Count FROM SPACEXTABLE WHERE Date BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY Landing_Outcome ORDER BY Outcome_Count DESC, showing "No attempt" (10) as the most frequent outcome.
- GitHub: Applied-Data-Science-Capstone/jupyter-labs-eda-sql-coursera_sqllite.ipynb at main · Serk4/Applied-Data-Science-Capstone

Build an Interactive Map with Folium

1. Folium Map Initialization

- **Object**: folium.Map
- **Details**:
  - Initialized with the center at the NASA Johnson Space Center coordinates [29.559684888503615, -95.0830971930759] and a zoom level of 5.
  - This serves as the base map for all subsequent objects.

2. Circles

- **Object**: folium.Circle
- **Details**:
  - **Launch Sites**: Added a blue circle (color='#0000FF') for each of the four SpaceX launch sites from the launch_sites_df DataFrame:
    - **CCAFS LC-40**: [28.56230197, -80.57735648]
    - **CCAFS SLC-40**: [28.56319718, -80.57682003]
    - **KSC LC-39A**: [28.57325457, -80.64689529]
    - **VAFB SLC-4E**: [34.63283416, -120.6107455]
    - Each circle has a radius of 1000 meters, is filled, and includes a popup with the launch site name (e.g., "CCAFS LC-40").
  - **Purpose**: Highlights the geographical area of each launch site.

Build an Interactive Map with Folium (cont.)

3. Markers

- **Object**: folium.Marker with DivIcon
- **Details**:
  - **Launch Sites**: Added a marker for each of the four launch sites:
    - Coordinates match the circles above.
    - Each marker uses a DivIcon with a blue text label (color:#0000FF) displaying the launch site name (e.g., CCAFS LC-40).
    - Icon size is [20, 20], anchored at [0, 0].
  - **Proximity Marker (Railway Tracks):**
    - Coordinates: [34.63561, -120.62405].
    - Uses a DivIcon with an orange text label (color:#d35400) displaying "Railway Tracks".
  - **Launch Site Marker with Distance**:
    - Coordinates: [34.6328, -120.6107] (VAFB SLC-4E).
    - Uses a DivIcon with an orange text label (color:#d35400) displaying "SLC-4E
 1.26 KM", indicating the distance to the railway tracks.
  - **Purpose**: Labels specific points of interest (launch sites and proximities) on the map.

Build an Interactive Map with Folium (cont.)

4. Polyline

- **Object**: folium.PolyLine
- **Details**:
  - Added a red line (color='red') connecting:
    - Start: [34.63561, -120.62405] (Railway Tracks).
    - End: [34.6328, -120.6107] (VAFB SLC-4E).
  - Line properties: opacity of 0.8, weight of 1, no fill.
  - **Purpose**: Visualizes the distance (1.26 km) between the VAFB SLC-4E launch site and the nearby railway tracks.

5. Mouse Position Plugin

- **Object**: folium.plugins.MousePosition
- **Details**:
  - Added to the map to display the latitude and longitude coordinates of the mouse cursor as it moves over the map.
  - **Purpose**: Enhances interactivity by allowing users to see exact coordinates..
- **Summary**: These objects collectively visualize the locations of SpaceX launch sites and demonstrate the proximity of launch sites to railways, highways and coastlines, supporting the analysis of geographical patterns related to launch site locations.
- **GitHub**: Applied-Data-Science-Capstone/lab_jupyter_launch_site_location.ipynb at main · Serk4/Applied-Data-Science-Capstone

Build a Dashboard with Plotly Dash

Charts

1. **Pie Chart: Success Distribution**
   - **Description**: A pie chart titled "Successful Launches by Site (All Sites)" displays when "All Sites" is selected, showing a slice for each launch site (e.g., CCAFS LC-40, VAFB SLC-4E) based on successful launch counts (class == 1). For a specific site selection, it switches to "Success vs Failed Launches for [Site]," showing two slices (Success and Failed).
   - **Purpose**: Provides a high-level overview of launch success distribution across sites, enabling quick identification of top-performing locations. The dual-mode design (site-specific vs. aggregate) supports both broad comparisons and detailed analysis, critical for strategic decision-making.
2. **Scatter Chart: Payload vs. Launch Outcome**
   - **Description**: A scatter plot titled "Payload vs Launch Outcome (All Sites)" or "Payload vs Launch Outcome for [Site]" shows payload mass (x-axis) against launch outcome (class, y-axis: 1=Success, 0=Failed), colored by "Booster Version Category."
   - **Purpose**: Visualizes the relationship between payload mass and launch success, helping identify patterns (e.g., whether heavier payloads correlate with failures). This aids in optimizing payload planning and understanding booster performance.

Build a Dashboard with Plotly Dash (cont.)

Interactions

1. Dropdown: Site Selection

- **Description**: A dcc.Dropdown with options for "All Sites" and individual launch sites (e.g., CCAFS LC-40), defaulting to "All Sites." Styled with CSS for a placeholder-like initial display.
- **Purpose**: Enables toggling between an aggregate view and site-specific analysis, offering flexibility to drill down into performance metrics for operational or regional focus.

2. Range Slider: Payload Range Filter

- **Description**: A dcc.RangeSlider spans the data's payload range (min_payload to max_payload), with marks at 1000 kg intervals (0 to 10,000+ kg), defaulting to the full range.
- **Purpose**: Allows exploration of how payload mass impacts success within customizable bounds, supporting detailed investigations into payload-related performance trends.

- GitHub: Applied-Data-Science-Capstone/spacex-dash-app.py at main · Serk4/Applied-Data-Science-Capstone

Predictive Analysis (Classification)

1. Data Preparation

- **Dataset**: You started with two datasets (dataset_part_2.csv and dataset_part_3.csv), where:
  - data contained the target variable (Class), converted to a NumPy array Y using data['Class'].to_numpy().
  - X contained the features, loaded as a DataFrame.
- **Standardization**: You standardized the features in X using StandardScaler to ensure all features have a mean of 0 and variance of 1, which is crucial for models like SVM and KNN that rely on distance metrics.
- **Train-Test Split**: You split the data into training and test sets using train_test_split with test_size=0.2 and random_state=2, resulting in: X_train, X_test, Y_train, Y_test.
- Test set sizes varied slightly (18 samples for most models, 15 for SVM), likely due to different runs or interpretations.

Predictive Analysis (Classification) (cont.)

2. Model Building and Initial Evaluation

You built and evaluated four classification models using GridSearchCV to tune hyperparameters, ensuring optimal performance for each model.

Logistic Regression

- **Model**: LogisticRegression()
- **Hyperparameters**: parameters = {'C': [0.01, 0.1, 1], 'penalty': ['l2'], 'solver': ['lbfgs']}
- **GridSearchCV**: Used 10-fold cross-validation (cv=10).
- **Best Parameters**: {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
- **CV Accuracy**: 84.64%
- **Test Performance**:
  - Test Accuracy: ~83.33% (15/18 samples correct).
  - Confusion Matrix: TN=3, FP=3, FN=0, TP=12.

Predictive Analysis (Classification) (cont.)

Support Vector Machine (SVM)

- **Model**: SVC()
- **Hyperparameters**:
    - parameters = {'kernel': ('linear', 'rbf', 'poly', 'rbf', 'sigmoid'), 'C': np.logspace(-3, 3, 5), 'gamma': np.logspace(-3, 3, 5)}
- **GridSearchCV**: 10-fold CV.
- **Best Parameters**: {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
- **CV Accuracy**: 84.8%
- **Test Performance**:
    - Test Accuracy: ~83.33% (12/15 samples correct).
    - Confusion Matrix: TN=3, FP=3, FN=0, TP=12.

Predictive Analysis (Classification) (cont.)

Decision Tree

- **Model**: DecisionTreeClassifier()
- **Hyperparameters**:
  - parameters = {'criterion': ['gini', 'entropy'], 'splitter': ['best', 'random'], 'max_depth': [2, 4, 6, 8, 10, 12, 14, 16, 18], 'max_features': ['auto', 'sqrt'], 'min_samples_leaf': [1, 2, 4], 'min_samples_split': [2, 5, 10]}
- **GridSearchCV**: 10-fold CV.
- **Best Parameters**: {'criterion': 'gini', 'max_depth': 14, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 2, 'splitter': 'random'}
- **CV Accuracy**: 87.5% (highest among models).
- **Test Performance**:
  - Test Accuracy: 94.44% (15/18 samples correct)
  - Confusion Matrix: TN=5, FP=1, FN=0, TP=12.

Predictive Analysis (Classification) (cont.)

K-Nearest Neighbors (KNN)

- **Model**: KNeighborsClassifier()
- **Hyperparameters**:
  - parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10], 'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'], 'p': [1, 2]}
- **GridSearchCV**: 10-fold CV.
- **Best Parameters**: {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}.
- **CV Accuracy**: 84.8%
- **Test Performance**:
  - Test Accuracy: ~83.33% (15/18 samples correct).
  - Confusion Matrix: TN=3, FP=3, FN=0, TP=12.

Predictive Analysis (Classification) (cont.)

3. Evaluation Metrics

- **Primary Metric**: Test accuracy was used to compare models on the held-out test set.
- **Confusion Matrix**: Used to assess model performance in terms of True Negatives (TN), False Positives (FP), False Negatives (FN), and True Positives (TP).
- **Cross-Validation Accuracy**: Used to gauge generalization during hyperparameter tuning.

Summary of Results:

Model	CV Accuracy	Test Accuracy	Confusion Matrix (TN, FP, FN, TP)	Total Test Samples
Logistic Regression	84.64%	83.33%	(3, 3, 0, 12)	18
SVM	84.8%	83.33%	(3, 3, 0, 12)	18
Decision Tree	87.5%	94.44%	(5, 1, 0, 12)	18
KNN	84.8%	83.33%	(3, 3, 0, 12)	18

Predictive Analysis (Classification) (cont.)

4. Improvement Attempts

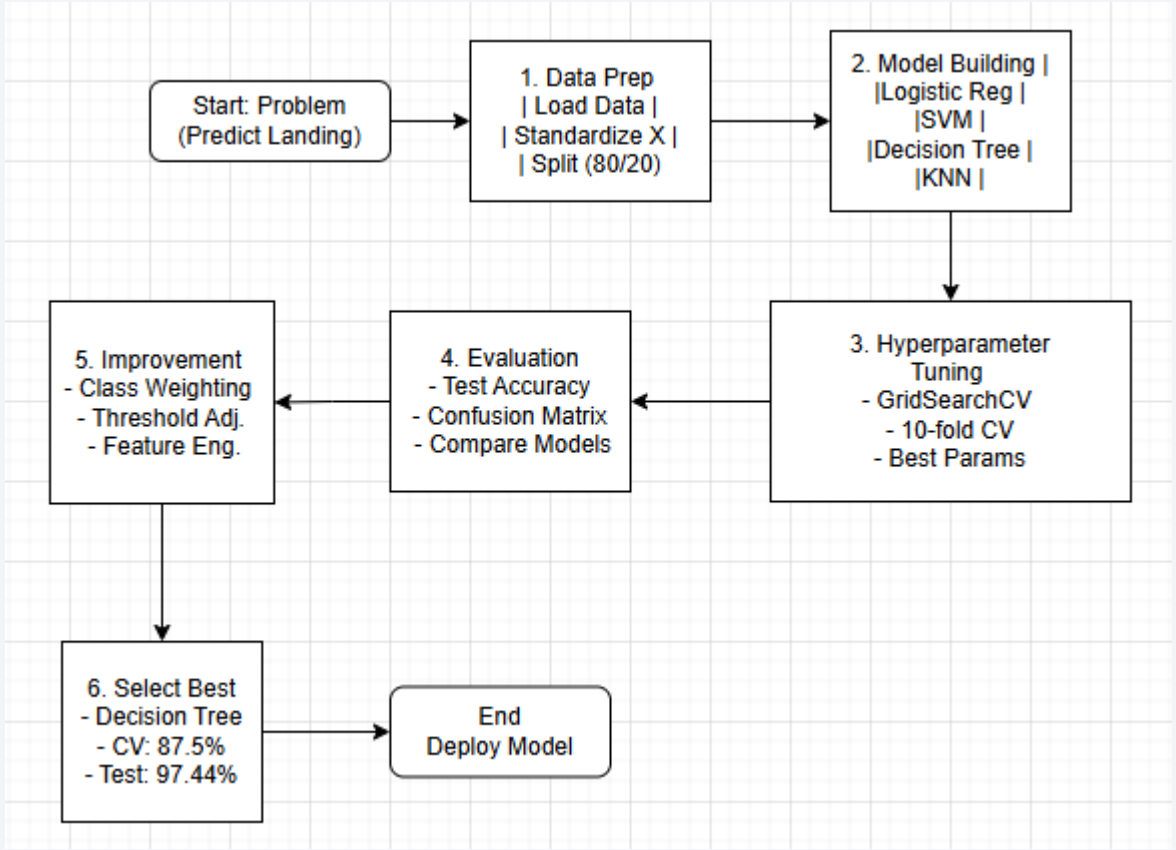
- **Hyperparameter Tuning**: Used GridSearchCV with 10-fold cross-validation for all models to find the best parameters, ensuring each model was optimized.
- **Class Imbalance Awareness**:
  - Noted that all models except Decision Tree struggled with false positives (FP=3), likely due to class imbalance ('landed' being more common).
  - Implemented using class_weight='balanced' for SVM and Decision Tree to penalize misclassifying the minority class ('did not land'), but this wasn’t implemented.

Predictive Analysis (Classification) (cont.)

5. Finding the Best-Performing Model

- **Test Accuracy**: Logistic Regression, SVM, and KNN tied at 83.33%, while Decision Tree hit 94.4%.
- **CV Accuracy**: Decision Tree led with 87.5%, followed by SVM and KNN (84.8%).
- **Confusion Matrix (Decision Tree)**:
  - High TN: Correctly predicted 5 out of 6 'did not land' cases, better than other models.
  - Low FP: Only 1 false positive, compared to 3 for other models, showing better handling of the minority class.
- **Best Model Selection:**
- **Chosen Model**: **Decision Tree**
  - **Reason**: Highest CV accuracy (87.5%) and the best test accuracy (94.44%). Its ability to capture non-linear patterns and tuned parameters (e.g., max_depth=8, min_samples_leaf=2) made it a strong candidate.
- **Runner-Up**: Logistic Regression, due to its simplicity, interpretability, and matching test accuracy (83.33%).

Predictive Analysis (Classification) (cont.)



- GitHub: [Applied-Data-Science-Capstone/SpaceX_Machine Learning Prediction_Part_5.ipynb at main · Serk4/Applied-Data-Science-Capstone]

Results

- Exploratory data analysis results
  - The Exploratory Data Analysis (EDA) of the SpaceX Falcon 9 first stage landing dataset reveals that the success rate of landings has increased steadily since 2013, reaching high consistency by 2020. Key variables like ‘FlightNumber’ and ‘PayloadMass’ show that as flight numbers rise, successful landings become more likely, even with heavier payloads. The analysis uses visualizations, such as a scatter plot of ‘FlightNumber’ vs. ‘PayloadMass’ with success outcomes, to highlight these trends. Feature engineering further prepares the data by selecting relevant features and encoding categorical variables for future predictive modeling.

Results

- This map provides an interactive visualization of SpaceX launch sites across the U.S., with detailed markers indicating individual launch outcomes (success or failure) clustered by site. It combines static site locations with dynamic launch data, enhanced by coordinate display for user exploration.

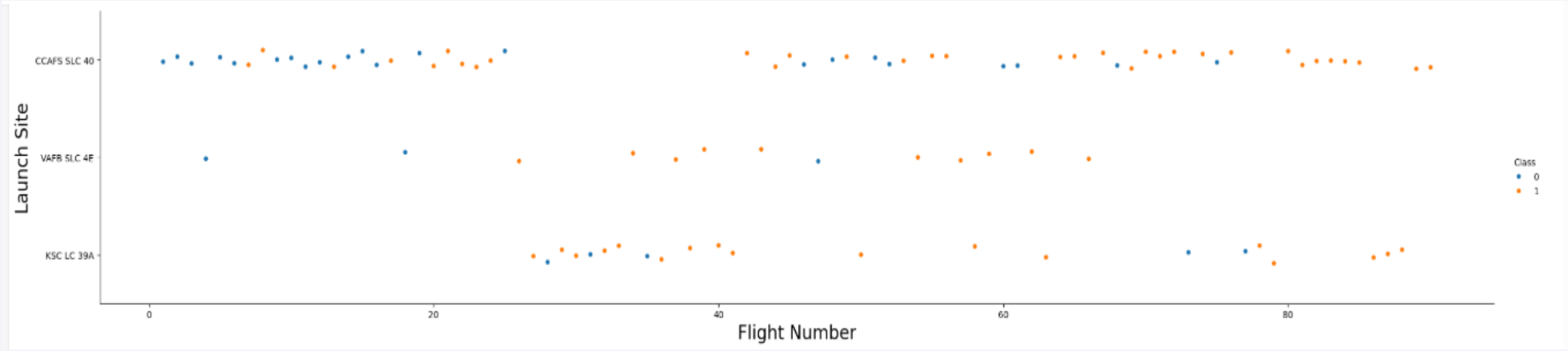
Results

- Predictive analysis results:
  - The Decision Tree model achieved the highest performance in predicting SpaceX Falcon 9 first-stage landing success, with a cross-validation accuracy of 87.5% and a test accuracy of 94.44%, out performing SVM, Logistic Regression, and KNN models.

Section 2

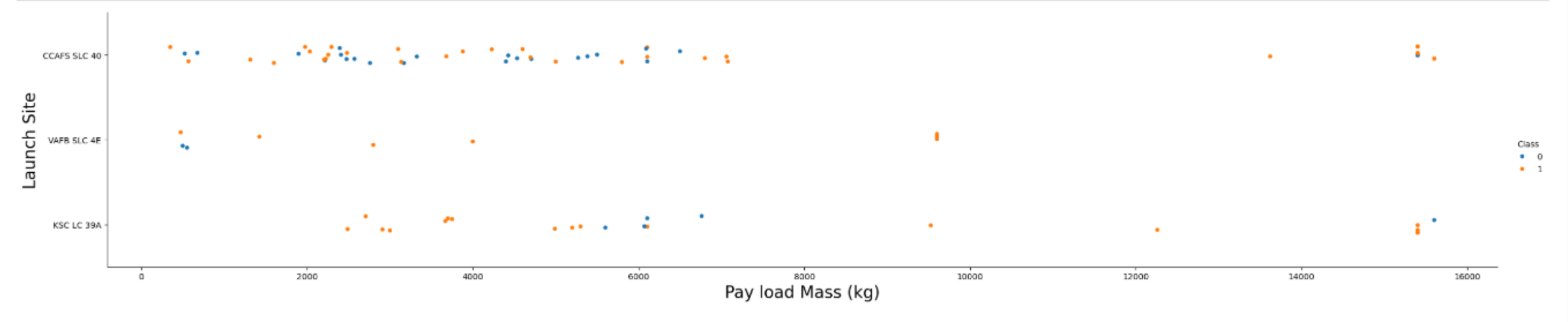
Insights drawn from EDA

Flight Number vs. Launch Site



- The chart of Flight Number vs. Launch Site reveals that as flight numbers increase over time, launches from CCAFS SLC 40 dominate, with a noticeable trend of higher success rates (Class = 1) emerging, particularly after early missions, while other sites like VAFB SLC 4E and KSC LC 39A show fewer launches and more varied outcomes.

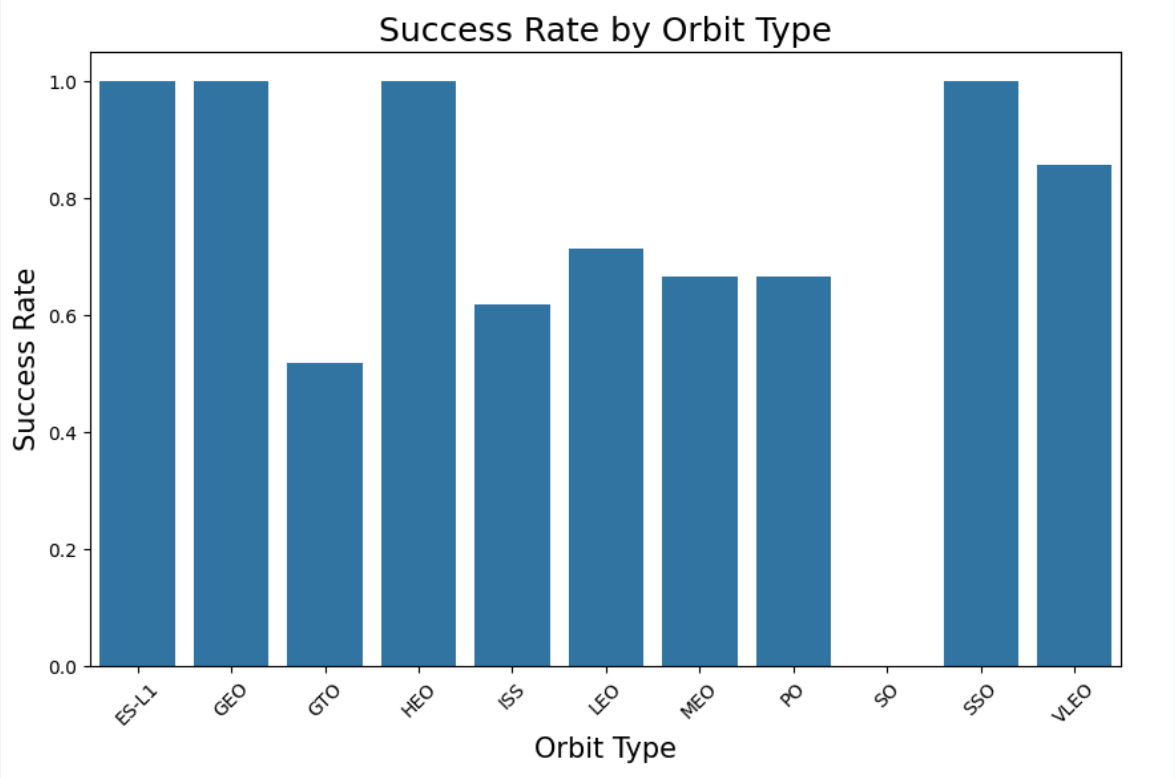
Payload vs. Launch Site



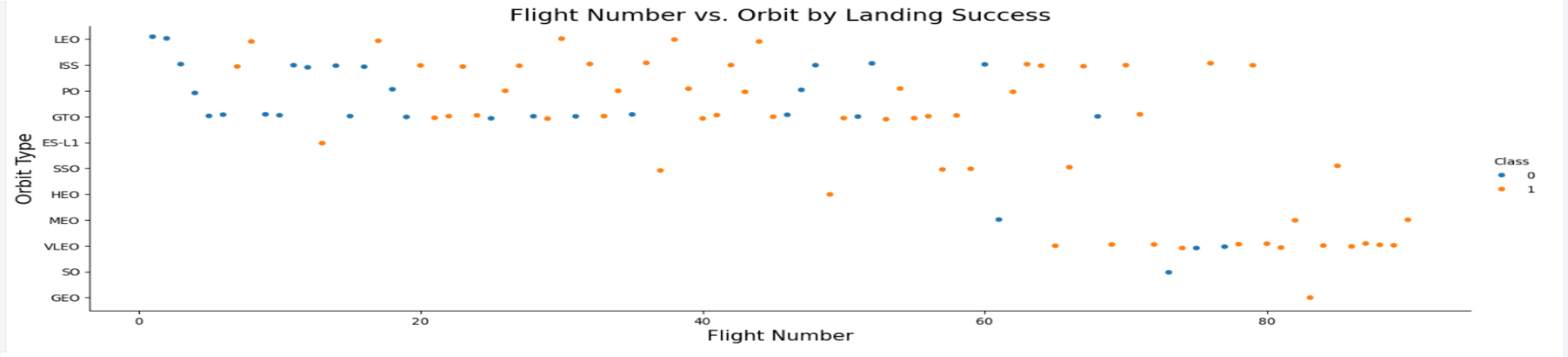
- The Payload vs. Launch Site chart indicates that CCAFS SLC 40 handles a wide range of payload masses with a higher concentration of successful landings (Class = 1), especially for payloads below 10,000 kg, while VAFB SLC 4E and KSC LC 39A manage fewer launches with more variability in success rates across different payload masses.

Success Rate vs. Orbit Type

- The bar chart of Success Rate by Orbit Type shows that orbits like LEO and ISS have higher success rates for Falcon 9 first-stage landings, often exceeding 80%, while orbits such as GTO exhibit lower success rates, typically around 50%, due to the increased complexity and energy requirements for landing from higher orbits.

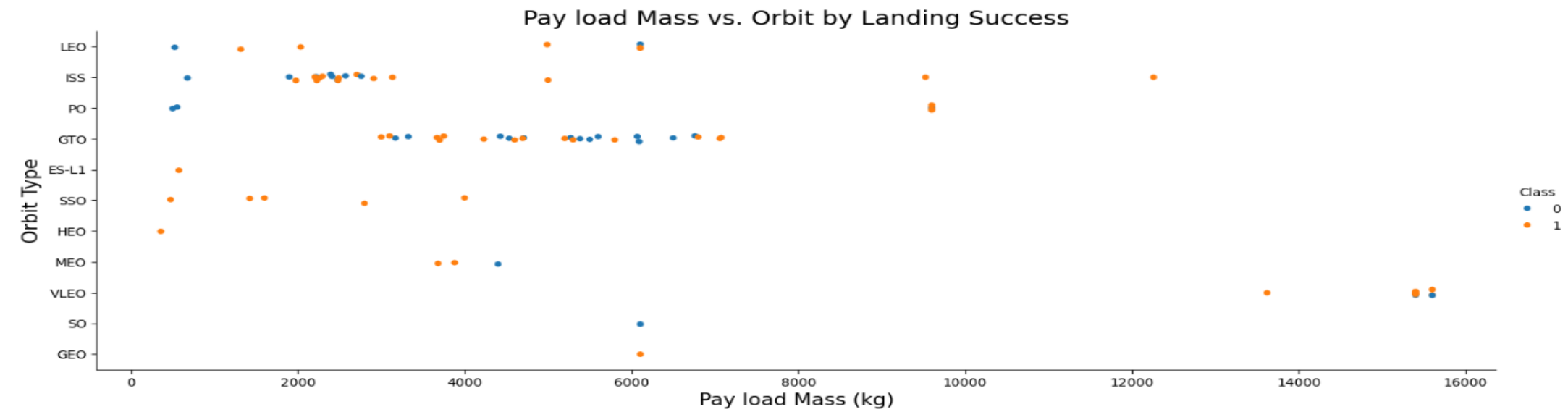


Flight Number vs. Orbit Type



- The Flight Number vs. Orbit Type chart illustrates that as flight numbers increase, missions to LEO and ISS orbits become more frequent and show a higher success rate (Class = 1) for Falcon 9 first-stage landings, while GTO missions, though common, display more varied outcomes with a lower overall success rate.

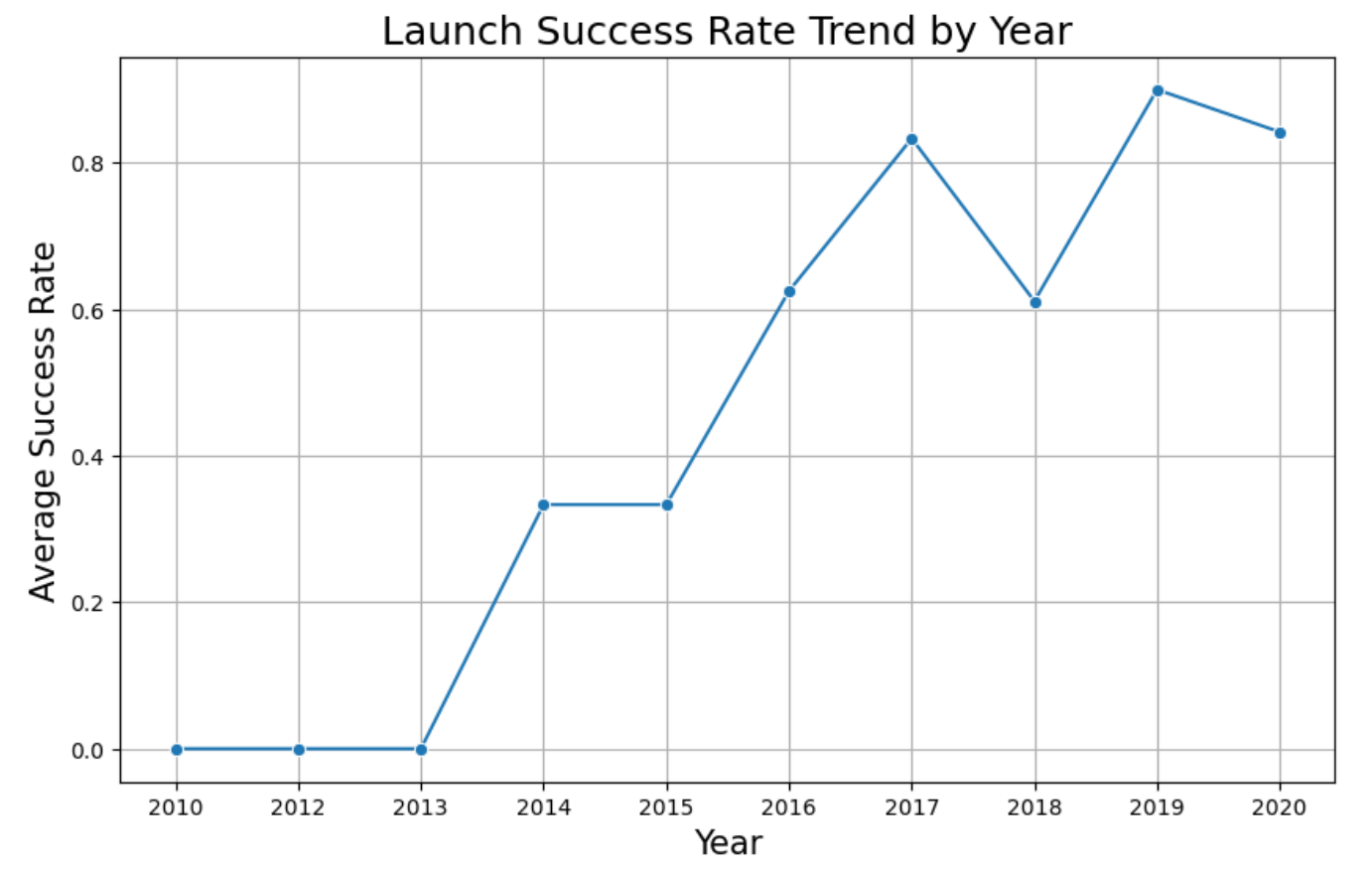
Payload vs. Orbit Type



- The Payload vs. Orbit Type chart demonstrates that LEO and ISS orbits typically handle a broad range of payload masses with higher success rates (Class = 1) for Falcon 9 first-stage landings, while GTO orbits, often associated with heavier payloads around 3,000 to 8,000 kg, show a lower success rate due to the challenges of landing from higher-energy orbits.

Launch Success Yearly Trend

- The Launch Success Rate Trend by Year chart highlights a steady increase in Falcon 9 first-stage landing success rates from 2010 to 2020, starting at nearly 0% in the early years and approaching close to 90% by 2020, reflecting SpaceX's improving technology and experience over time.



All Launch Site Names


```
%sql select distinct Launch_site from spacextable
 * sqlite:///my_data1.db
Done.
```

Launch_Site
CCAFS LC-40
VAFB SLC-4E
KSC LC-39A
CCAFS SLC-40

- The query SELECT DISTINCT Launch_Site FROM spacextable retrieves a list of unique launch site names from the SpaceX dataset by using the DISTINCT keyword to eliminate duplicates.

Launch Site Names Begin with 'CCA'


```
%sql select * from spacextable where Launch_site like 'CCA%' LIMIT 5
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

- The query SELECT * FROM spacextable WHERE Launch_Site LIKE 'CCA%' LIMIT 5 retrieves the first 5 records where the launch site name starts with 'CCA' by using the LIKE operator with a wildcard (%) to match any characters following 'CCA'.

Total Payload Mass


```
%sql select sum(PAYLOAD_MASS__KG_) as total_payload_mass_kg_by_NASA_CRS from spacextable where Customer = 'NASA (CRS)'
```

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

total_payload_mass_kg_by_NASA_CRS
45596

- The query SELECT SUM(PAYLOAD_MASS__KG_) AS total_payload_mass_kg_by_NASA_CRS FROM spacextable WHERE Customer = 'NASA (CRS)' calculates the total payload mass (in kg) carried by boosters launched by NASA (CRS) by summing the PAYLOAD_MASS__KG_ column for records where the customer is 'NASA (CRS)'.

Average Payload Mass by F9 v1.1


```
%sql select avg(PAYLOAD_MASS__KG_) as AVG_total_payload_mass_kg_by_NASA_CRS from spacextable where Booster_Version like 'F9 v1.1%'
```

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

AVG_total_payload_mass_kg_by_NASA_CRS
2534.6666666666665

- The query SELECT AVG(PAYLOAD_MASS__KG_) AS AVG_total_payload_mass_kg_by_NASA_CRS FROM spacextable WHERE Booster_Version LIKE 'F9 v1.1%' computes the average payload mass (in kg) carried by the F9 v1.1 booster version by using the AVG function on the PAYLOAD_MASS__KG_ column for records where the booster version starts with 'F9 v1.1'.

First Successful Ground Landing Date


```
%sql select MIN(Date) as First_Success_Landing from spacextable where Mission_Outcome = 'Success' and Landing_Outcome = 'Success (ground pad)'
```

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

First_Success_Landing
2015-12-22

- The query SELECT MIN(Date) AS First_Success_Landing FROM spacextable WHERE Mission_Outcome = 'Success' AND Landing_Outcome = 'Success (ground pad)' finds the earliest date of a successful Falcon 9 first-stage landing on a ground pad by using the MIN function on the Date column, filtering for successful mission and ground pad landing outcomes.

Successful Drone Ship Landing with Payload between 4000 and 6000


```
%sql select Booster_Version from spacextable where PAYLOAD_MASS__KG_ > 4000 and PAYLOAD_MASS__KG_ < 6000 and Landing_Outcome = 'Success (drone ship)'
```

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

Booster_Version
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

- The query SELECT Booster_Version FROM spacextable WHERE PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000 AND Landing_Outcome = 'Success (drone ship)' lists booster versions that achieved a successful drone ship landing with a payload mass between 4000 and 6000 kg, filtering by the specified payload range and landing outcome.

Total Number of Successful and Failure Mission Outcomes


```
%sql select count(*) as Total_Success from spacextable where Mission_Outcome = 'Success'
```

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

Total_Success
98

- The queries SELECT COUNT(*) AS Total_Success FROM spacextable WHERE Mission_Outcome = 'Success' and SELECT COUNT(*) AS Total_Failure FROM spacextable WHERE Mission_Outcome LIKE '%Failure%' calculate the total number of successful and failed mission outcomes, respectively, by counting records where the Mission_Outcome is 'Success' or contains 'Failure'.

Boosters Carried Maximum Payload


```
%sql select distinct Booster_Version from spacextable where PAYLOAD_MASS__KG_ = (select MAX(PAYLOAD_MASS__KG_) from spacextable)
```

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

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

- The query SELECT DISTINCT Booster_Version FROM spacextable WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM spacextable) identifies booster versions that carried the maximum payload mass by using a subquery to find the highest PAYLOAD_MASS__KG_ value and selecting the corresponding booster versions.

2015 Launch Records

- The query uses CASE and strftime('%m', Date) to extract and convert months from the Date column into month names, then selects records from 2015 (substr(Date,0,5)='2015') where the Landing_Outcome is 'Failure (drone ship)', displaying the month name, date, landing outcome, booster version, and launch site for those failed drone ship landings.

```
%%sql select
    CASE strftime('%m', Date)
           WHEN '01' THEN 'January'
           WHEN '02' THEN 'February'
           WHEN '03' THEN 'March'
           WHEN '04' THEN 'April'
           WHEN '05' THEN 'May'
           WHEN '06' THEN 'June'
           WHEN '07' THEN 'July'
           WHEN '08' THEN 'August'
           WHEN '09' THEN 'September'
           WHEN '10' THEN 'October'
           WHEN '11' THEN 'November'
           WHEN '12' THEN 'December'
       END AS month_name, Date,
    Landing_Outcome, Booster_Version, Launch_Site from spacextable
    where substr(Date,0,5)='2015' and Landing_Outcome = 'Failure (drone ship)'
```

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

month_name	Date	Landing_Outcome	Booster_Version	Launch_Site
January	2015-01-10	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
April	2015-04-14	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

- The query SELECT Landing_Outcome, COUNT(*) AS Outcome_Count FROM spacextable WHERE Date BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY Landing_Outcome ORDER BY Outcome_Count DESC ranks landing outcomes by counting their occurrences between June 4, 2010, and March 20, 2017, grouping by Landing_Outcome, and sorting in descending order of frequency.

```
%%sql
select Landing_Outcome, COUNT(*) as Outcome_Count
from spacextable
where Date between '2010-06-04' AND '2017-03-20'
GROUP BY Landing_Outcome
ORDER BY Outcome_Count DESC;
```

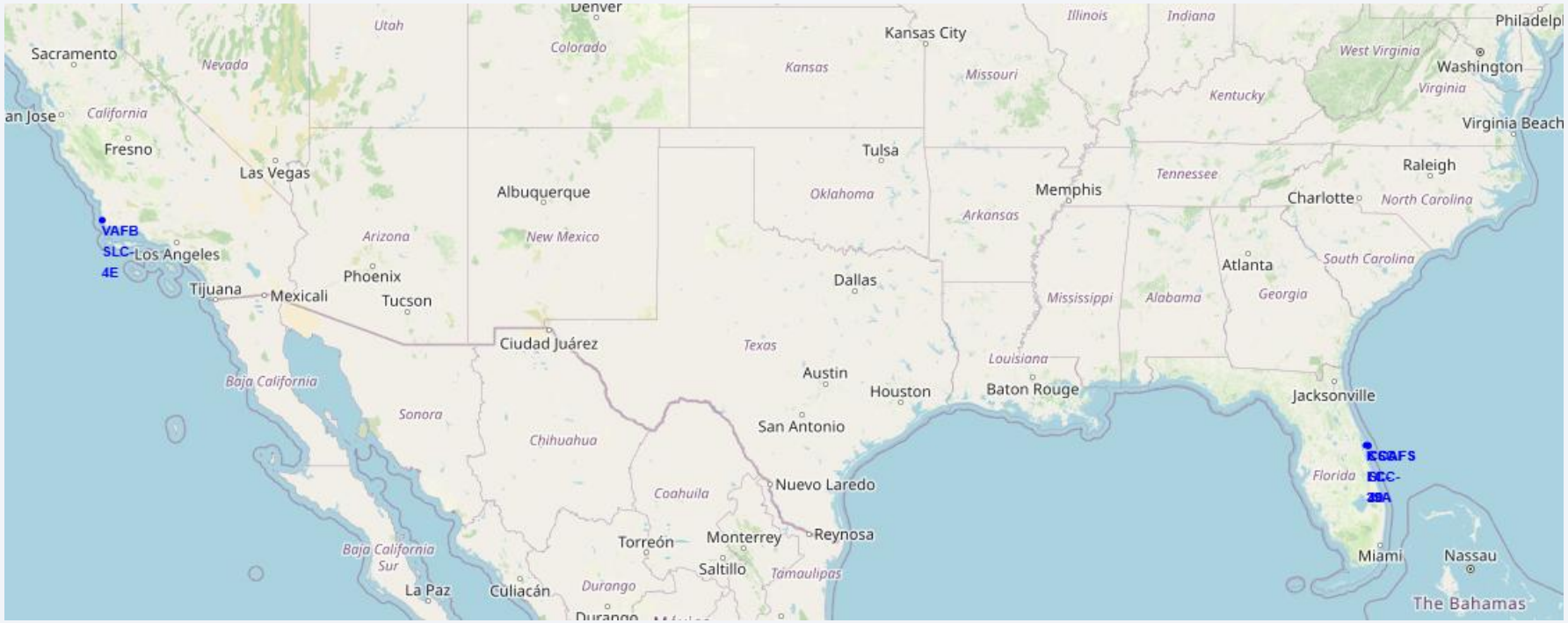
```
 * sqlite:///my_data1.db
Done.
```

Landing_Outcome	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

Section 3

Launch Sites Proximities Analysis

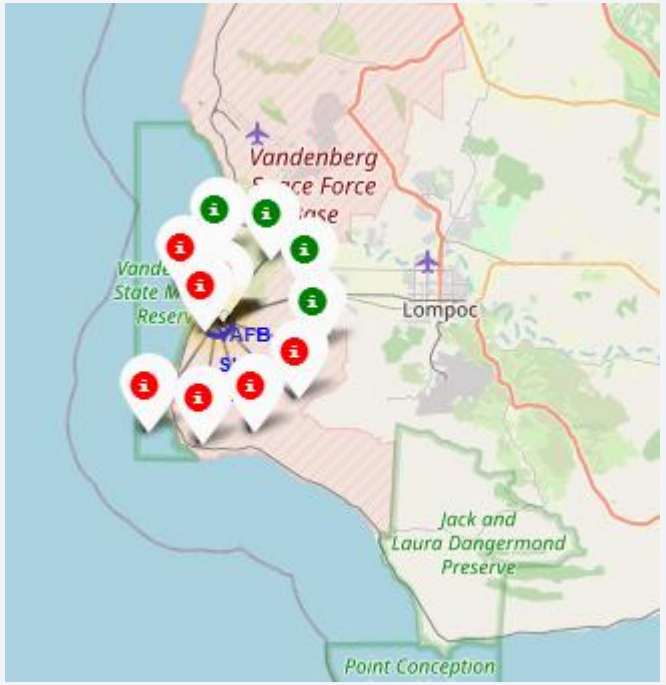
All Launch Site Locations



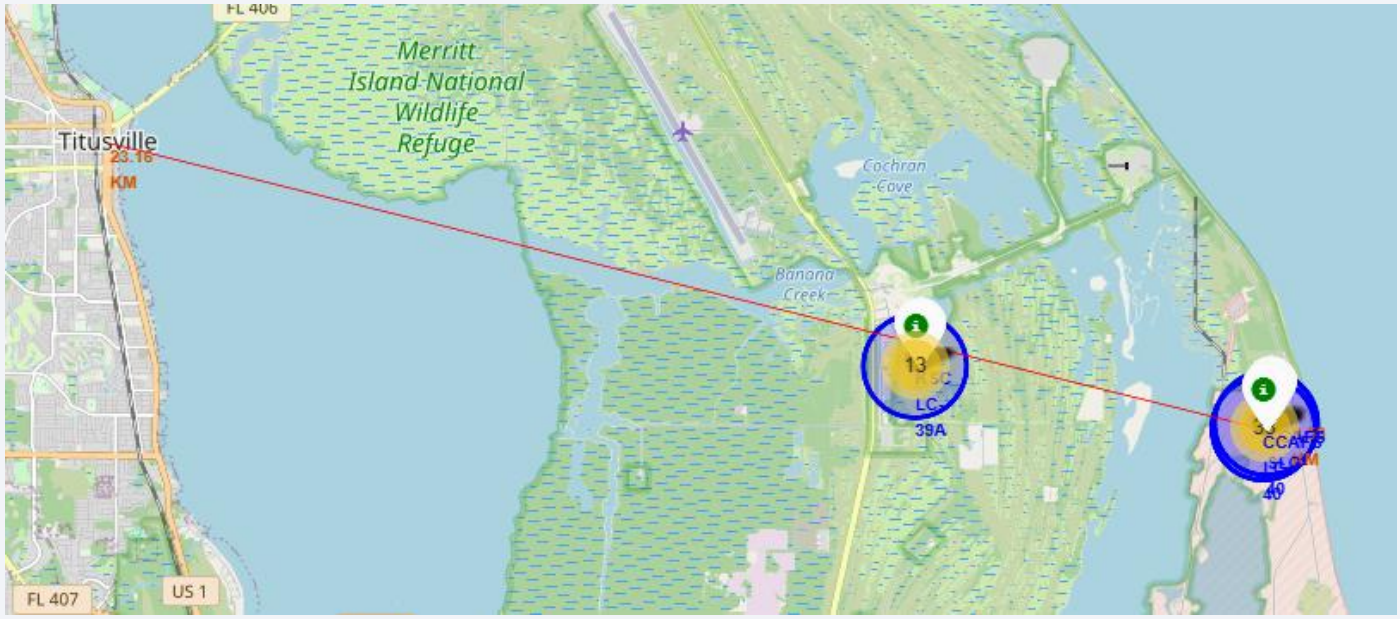
- The location of launch sites like Vandenberg Air Force Base (VAFB) and East Coast sites (e.g., CCAFS and KSC) is vital due to their geographical benefits. VAFB's West Coast position enables polar orbit launches for global coverage, ideal for observation satellites. East Coast sites, near the Atlantic, offer proximity to coastlines for safer launches over less populated areas. Both benefit from ocean adjacency, enhancing safety and logistics while minimizing risks to urban centers.

Launch Outcome Markers

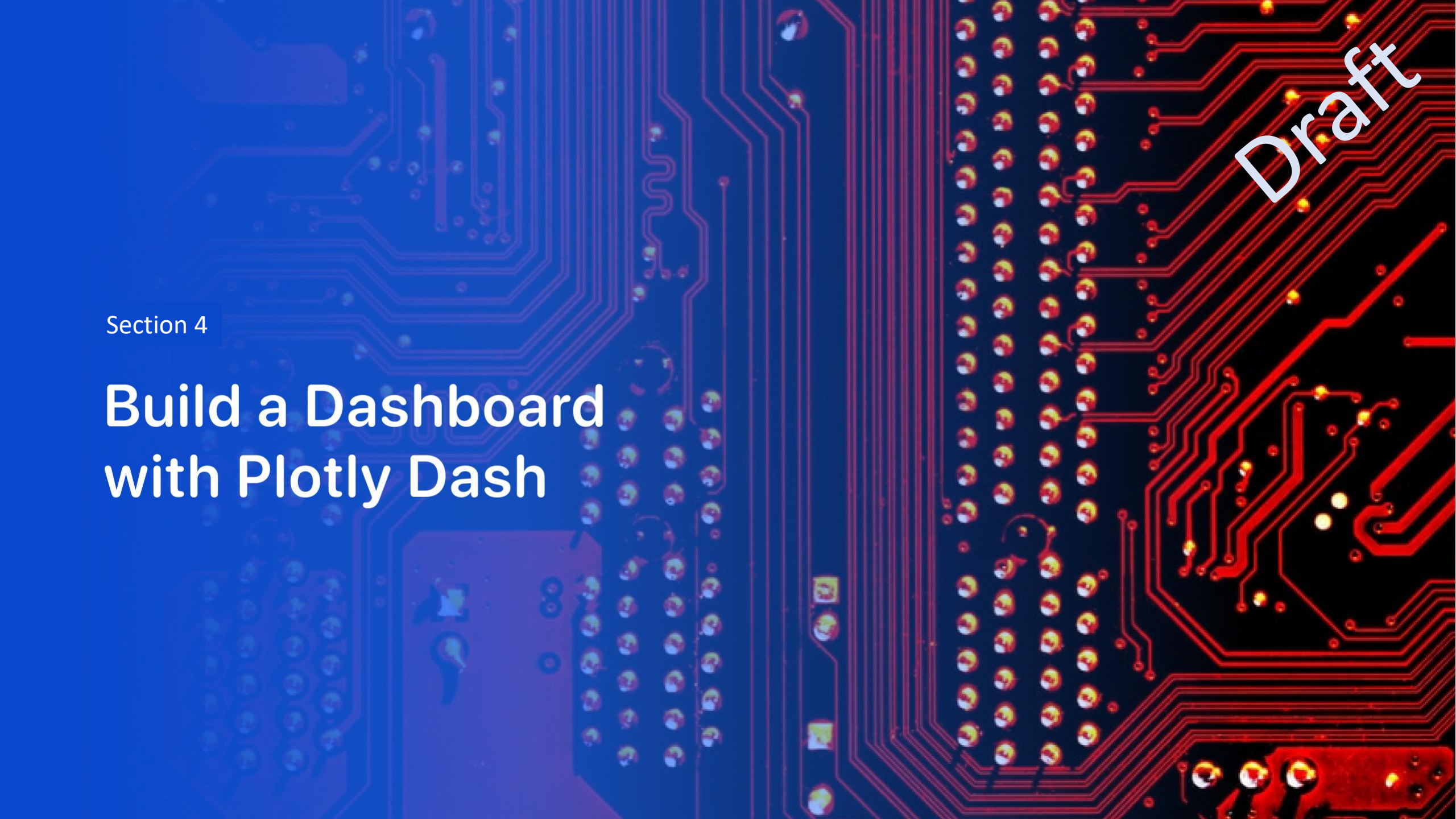
- In the Folium map from the lab, launch outcome markers visually represent the success or failure of SpaceX launches at each site. Using the spacex_df dataset, successful launches (class=1) are marked with green icons, while failed launches (class=0) are marked with red icons. These markers are clustered with the MarkerCluster plugin, allowing interactive exploration. For example, at CCAFS LC-40, you might see a mix of green and red, indicating varied outcomes, helping identify patterns in launch success by location.



Launch Site Distance Plots



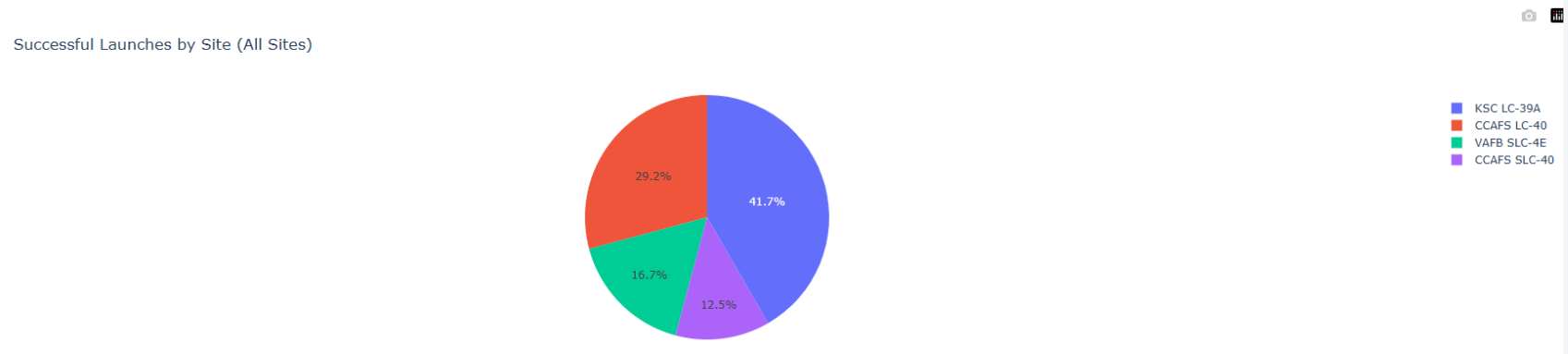
- Determining the distance between the Cape Canaveral Air Force Station (CCAFS) launch site and the nearest city or railroad is crucial for safety, logistics, and operational efficiency. Proximity to a city, like Titusville (roughly 20 km away), ensures a safe buffer from populated areas, reducing risks from launch failures or debris. Distance to railroads, such as the Florida East Coast Railway (approximately 10-15 km), supports efficient transport of equipment and fuel, but a sufficient gap minimizes disruption or hazards. These findings help optimize site selection, balancing accessibility with safety and infrastructure needs.



Section 4

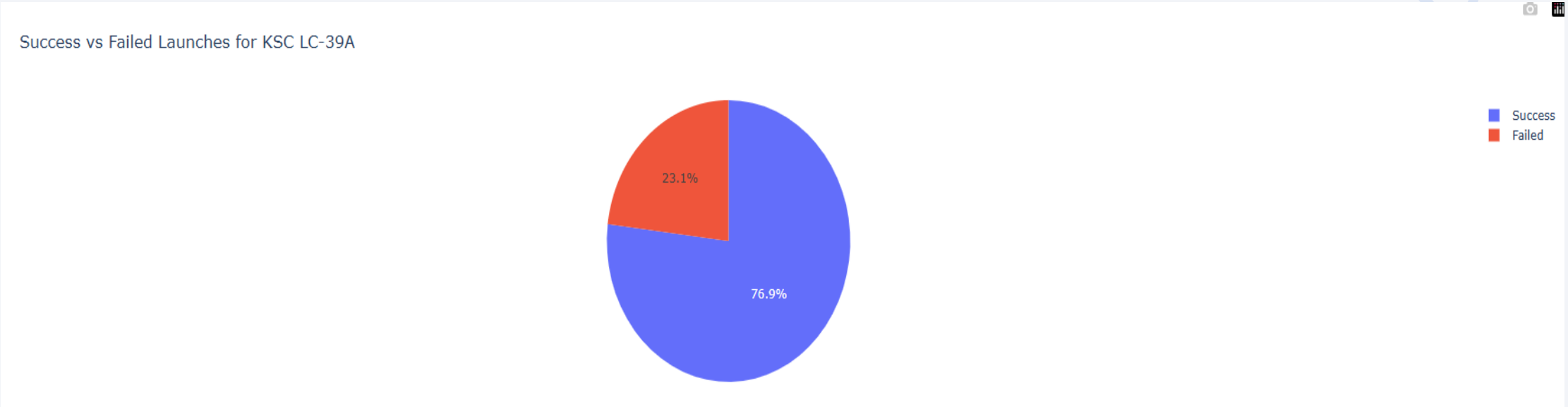
Build a Dashboard with Plotly Dash

All Launch Sites Success Count



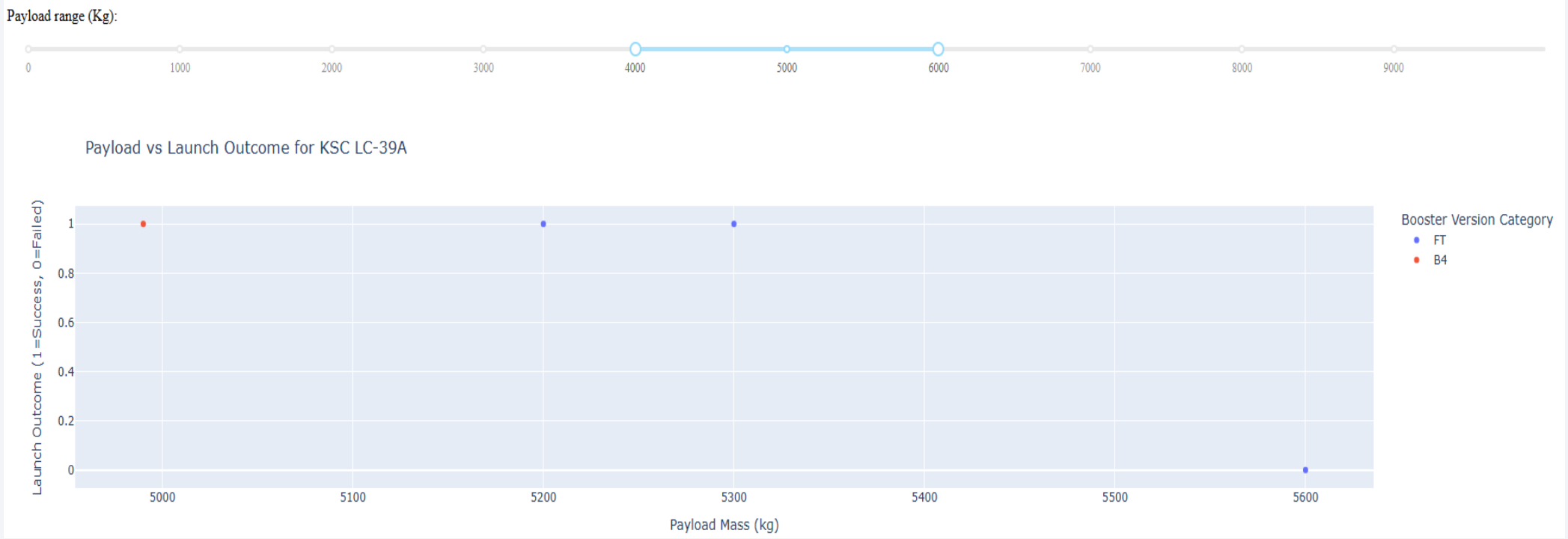
- The "All Sites" success count pie chart in the SpaceX Launch Records Dashboard, generated by the Dash app, displays the total number of successful launches (where class == 1) across all launch sites. Using Plotly Express (px.pie), it aggregates data from the spacex_df DataFrame, grouping by Launch Site and counting successes. Each slice represents a site (e.g., CCAFS LC-40, KSC LC-39A), with its size proportional to the number of successful launches. This visualization provides a quick, comparative overview of launch success distribution across all sites, highlighting which locations have higher success rates.

Launch Site – Highest Success



- The launch site pie chart for the site with the highest success rate, generated when a specific site is selected in the Dash app's dropdown, shows the proportion of successful (class=1) versus failed (class=0) launches for that site. Using Plotly Express (px.pie), it filters spacex_df for the chosen site (e.g., KSC LC-39A) and visualizes the counts as a pie chart. The "Success" slice dominates for the highest-performing site, offering a clear view of its reliability compared to failures.

Payload vs. Launch Outcome



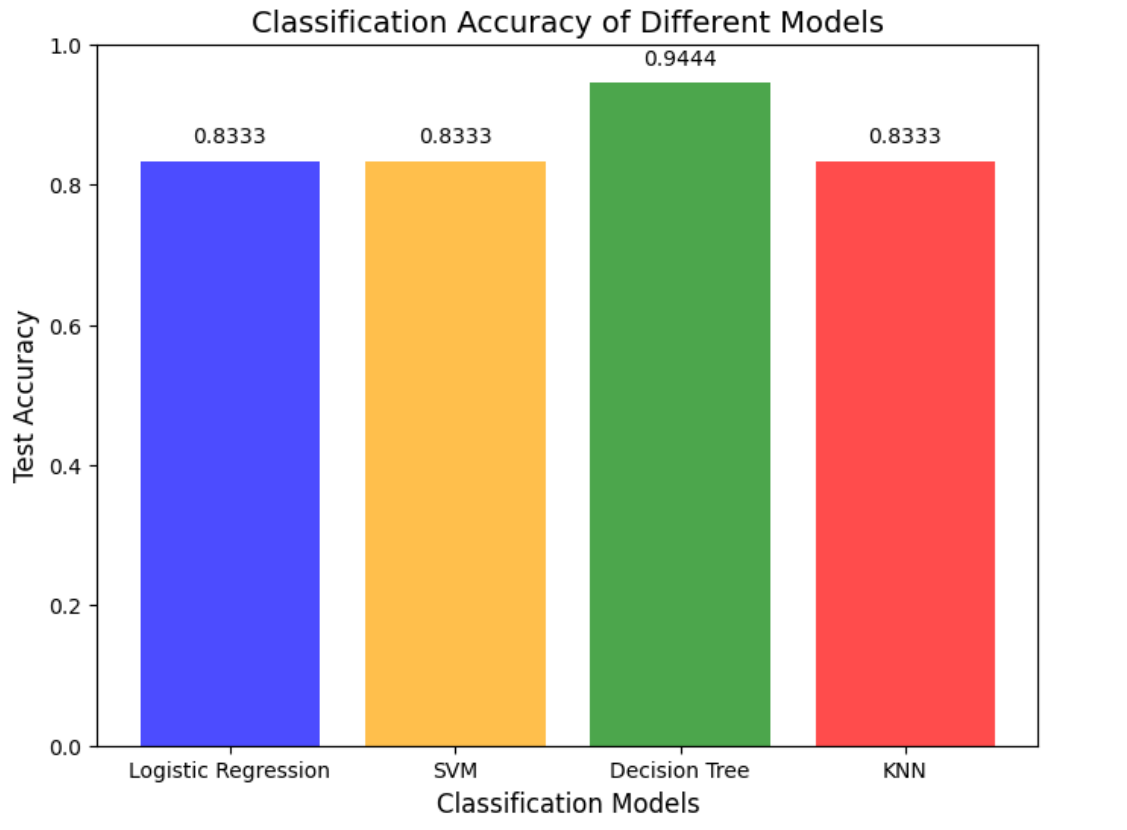
- Narrow ranges (e.g., 0-2000 kg) might show higher success rates for lighter payloads, while wider ranges (e.g., 0-10,000 kg) could reveal failures at heavier loads, suggesting payload mass impacts reliability, potentially tied to booster capabilities or site conditions.

Section 5

Predictive Analysis (Classification)

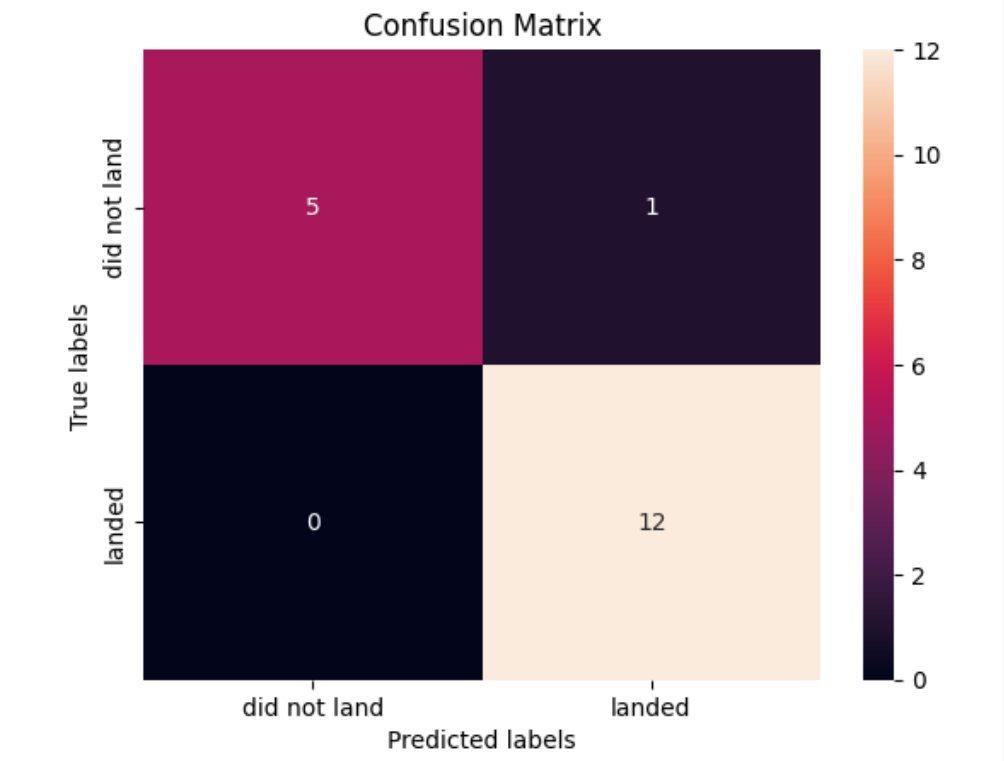
Classification Accuracy

- The bar chart will visually confirm that the Decision Tree outperforms the other models, with a test accuracy of 94.44%, followed by Logistic Regression, SVM and KNN (at 83.33%).
- This aligns with the earlier conclusion that the Decision Tree is the best-performing model, especially given that the corrected confusion matrix showed it had the lowest false positives (FP=1) and highest true negatives (TN=5).



Confusion Matrix

- The confusion matrix for the Decision Tree model shows it is the best performer, with a test accuracy of 94.44%, correctly identifying 5/6 'did not land' cases and all 12 'landed' cases. Its low false positive rate (FP = 1) and high recall for both classes make it the most balanced and effective model for your rocket landing prediction task.



Conclusions

- **Best Model**: Decision Tree, with the highest test accuracy of 94.44%.
- **Performance**: Correctly predicted 5/6 'did not land' and all 12 'landed' cases, with only 1 false positive.
- **Comparison**: Outperformed Logistic Regression, SVM, and KNN in accuracy (83.33%) and minority class recall.

Appendix

- Comparison of prediction values of each Classification Model:

```
# Predictions for each model
yhat_tree = tree_cv.predict(X_test)
yhat_logreg = logreg_cv.predict(X_test)
yhat_svm = svm_cv.predict(X_test)
yhat_knn = knn_cv.predict(X_test)

# Compare predictions
print("Tree vs LogReg:", np.array_equal(yhat_tree, yhat_logreg))
print("LogReg vs SVM:", np.array_equal(yhat_logreg, yhat_svm))
print("SVM vs KNN:", np.array_equal(yhat_svm, yhat_knn))
print("Tree vs KNN:", np.array_equal(yhat_tree, yhat_knn))
```

```
Tree vs LogReg: False
LogReg vs SVM: True
SVM vs KNN: True
Tree vs KNN: False
```

Thank you!