IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Sushil AB

5th August 2026

Outline

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

Executive Summary

- Summary of methodologies
- Summary of all results

The project successfully builds a machine learning pipeline to predict Falcon 9 first stage landing success, enabling Space Y to estimate launch costs. The methodologies involve data preprocessing, model training with hyperparameter tuning, and evaluation using accuracy and confusion matrices. All models achieve high test accuracies (around 83.33%–88.89%), with SVM (using rbf kernel) or logistic regression likely performing best. These predictions help Space Y compete with SpaceX by estimating launch costs based on first stage reusability.

Introduction

- Project background and context

The commercial space industry is rapidly evolving, with companies like SpaceX, Virgin Galactic, Rocket Lab, and Blue Origin making space travel more accessible and cost-effective. SpaceX, founded by Elon Musk, stands out due to its ability to reuse the first stage of its Falcon 9 rocket, significantly reducing launch costs to approximately $62 million compared to competitors' costs of up to $165 million. The ability to recover and reuse the first stage is a key factor in these savings. In this project, you assume the role of a data scientist at Space Y, a new rocket company aiming to compete with SpaceX. Your task is to determine launch costs by predicting whether SpaceX will reuse the Falcon 9 first stage, using machine learning models trained on public data rather than rocket science principles. This prediction helps estimate launch pricing for competitive bidding.

- Problems you want to find answers

The primary objective is to predict whether the Falcon 9 first stage will land successfully and be reused, which directly impacts the cost of a launch. Specific problems include:

Determining if the first stage will land successfully based on mission parameters.

Identifying the most effective machine learning model to predict landing success.

Analyzing the cost implications of reusing the first stage for Space Y's competitive strategy.

Creating dashboards to visualize data and support decision-making for Space Y's team.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Describe how data was collected
- Perform data wrangling
  - Describe how data was processed
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

- Describe how data sets were collected.
- You need to present your data collection process use key phrases and flowcharts

Data Collection – SpaceX API

Key Phrases

REST API: Fetching SpaceX launch data using HTTP GET requests from https://api.spacexdata.com/v4/launches/past.

Static JSON: Using a consistent dataset from API_call_spacex_api.json for reproducibility.

Data Filtering: Extracting Falcon 9 launches, removing Falcon 1 and multi-core/payload entries.

Helper Functions:

- getBoosterVersion: Retrieves rocket names.
- getLaunchSite: Extracts launch site, longitude, latitude.
- getPayloadData: Fetches payload mass and orbit.
- getCoreData: Collects landing outcome, flights, gridfins, reuse, legs, landing pad, block, serial.

Data Wrangling: Converting JSON to DataFrame, handling missing PayloadMass with mean replacement.

Output: Creating dataset_part_1.csv with 90 Falcon 9 launches up to Nov 13, 2020.

https://github.com/SUSHIL1102/Data-Science-Capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

Data Collection - Scraping

- **Web Scraping**: Extracting data from a website, specifically the Wikipedia page "List of Falcon 9 and Falcon Heavy launches."
- **HTTP GET Request**: Using the requests library to fetch the HTML content from a static URL.
- **BeautifulSoup**: Parsing the HTML content to navigate and extract elements using the html.parser.
- **HTML Table Extraction**: Identifying and extracting the target table (third table) containing launch records.
- **Column Names Extraction**: Iterating through <th> elements to extract and clean column names using a custom function (extract_column_from_header).
- **Data Parsing**: Looping through table rows (<tr>) to extract data for each launch, handling flight numbers, dates, times, booster versions, launch sites, payloads, payload masses, orbits, customers, launch outcomes, and booster landing statuses.
- **Helper Functions**: Using provided functions (date_time, booster_version, landing_status, get_mass) to clean and format extracted data.
- **Data Storage**: Storing parsed data in a dictionary (launch_dict) with keys matching column names.
- **Pandas DataFrame**: Converting the dictionary to a DataFrame for structured data storage.
- **CSV Export**: Saving the DataFrame to a CSV file (spacex_web_scraped.csv) for further analysis.
- **Error Handling**: Managing missing values, annotations, and inconsistent formatting in the HTML table.

https://github.com/SUSHIL1102/Data-Science-Capstone/blob/main/jupyter-labs-webscraping.ipynb

Data Wrangling

•**Dataset**: Loaded dataset_part_1.csv (90 Falcon 9 launches).
•**Missing Values**: LandingPad (28.89% missing, retained), no missing PayloadMass after prior mean replacement.
•**Data Types**: Numerical (e.g., FlightNumber, PayloadMass), Categorical (e.g., Orbit, Outcome).
•**EDA Tasks**:
•Counted launches per site: CCAFS SLC 40 (55), KSC LC 39A (22), VAFB SLC 4E (13).
•Orbit occurrences: GTO (27), ISS (21), VLEO (14), others.
•Outcome analysis: True ASDS, None None, etc., mapped to Class (1=success, 0=failure).
•**Label Creation**: Converted Outcome to Class (0 for bad_outcomes, 1 for successful landings).
•**Success Rate**: 66.67% successful landings.
•**Output**: Saved as dataset_part_2.csv.

https://github.com/SUSHIL1102/Data-Science-Capstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

- **Exploratory Data Analysis (EDA)**:
  - Visualized FlightNumber vs. PayloadMass with Class (success/failure) using Seaborn catplot.
  - Insight: Success rate increased from 2013 to 2020; higher payload masses still often result in successful landings.
- **Feature Engineering**:
  - Selected features: FlightNumber, PayloadMass, Orbit, LaunchSite, Flights, GridFins, Reused, Legs, LandingPad, Block, ReusedCount, Serial.
  - Task 7: Applied get_dummies to encode categorical columns (Orbit, LaunchSite, LandingPad, Serial) into features_one_hot (80 columns).
  - Task 8: Cast all columns in features_one_hot to float64 for consistency.
  - Exported: features_one_hot to dataset_part_3.csv for future analysis.
- **Libraries**: Pandas, NumPy, Matplotlib, Seaborn.

https://github.com/SUSHIL1102/Data-Science-Capstone/blob/main/edadataviz%20(copy).ipynb

EDA with SQL

- **Dataset**: Loaded Spacex.csv (101 launches) into SQLite database (SPACEXTBL).
- **Setup**: Installed sqlalchemy, ipython-sql; connected to my_data1.db.
- **Data Cleaning**: Removed blank rows, created SPACEXTABLE.
- **SQL Queries**:
- **Task 1**: Unique launch sites: CCAFS LC-40, VAFB SLC-4E, KSC LC-39A, CCAFS SLC-40.
- **Task 2**: 5 records with launch sites starting with 'CCA'.
- **Task 3**: Total NASA (CRS) payload: 45,596 kg.
- **Task 4**: Avg. payload for F9 v1.1: 2,534.67 kg.
- **Task 5**: First successful ground pad landing: 2015-12-22.
- **Task 6**: Boosters with successful drone ship landings, payload 4,000–6,000 kg: F9 FT B1022, B1026, B1021.2, B1031.2.
- **Task 7**: Mission outcomes: Success (98), Success (unclear payload) (1), Success (space) (1), Failure (1).
- **Task 8**: Boosters with max payload (15,600 kg): F9 B5 variants.
- **Task 9**: 2015 drone ship failures: Jan (F9 v1.1 B1012), Apr (F9 v1.1 B1015), CCAFS LC-40.
- **Task 10**: Pending (landing outcome counts, 2010-06-04 to 2017-03-20).

https://github.com/SUSHIL1102/Data-Science-Capstone/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- Summarize what map objects such as markers, circles, lines, etc. you created and added to a folium map
- Explain why you added those objects
- Add the GitHub URL of your completed interactive map with Folium map, as an external reference and peer-review purpose

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

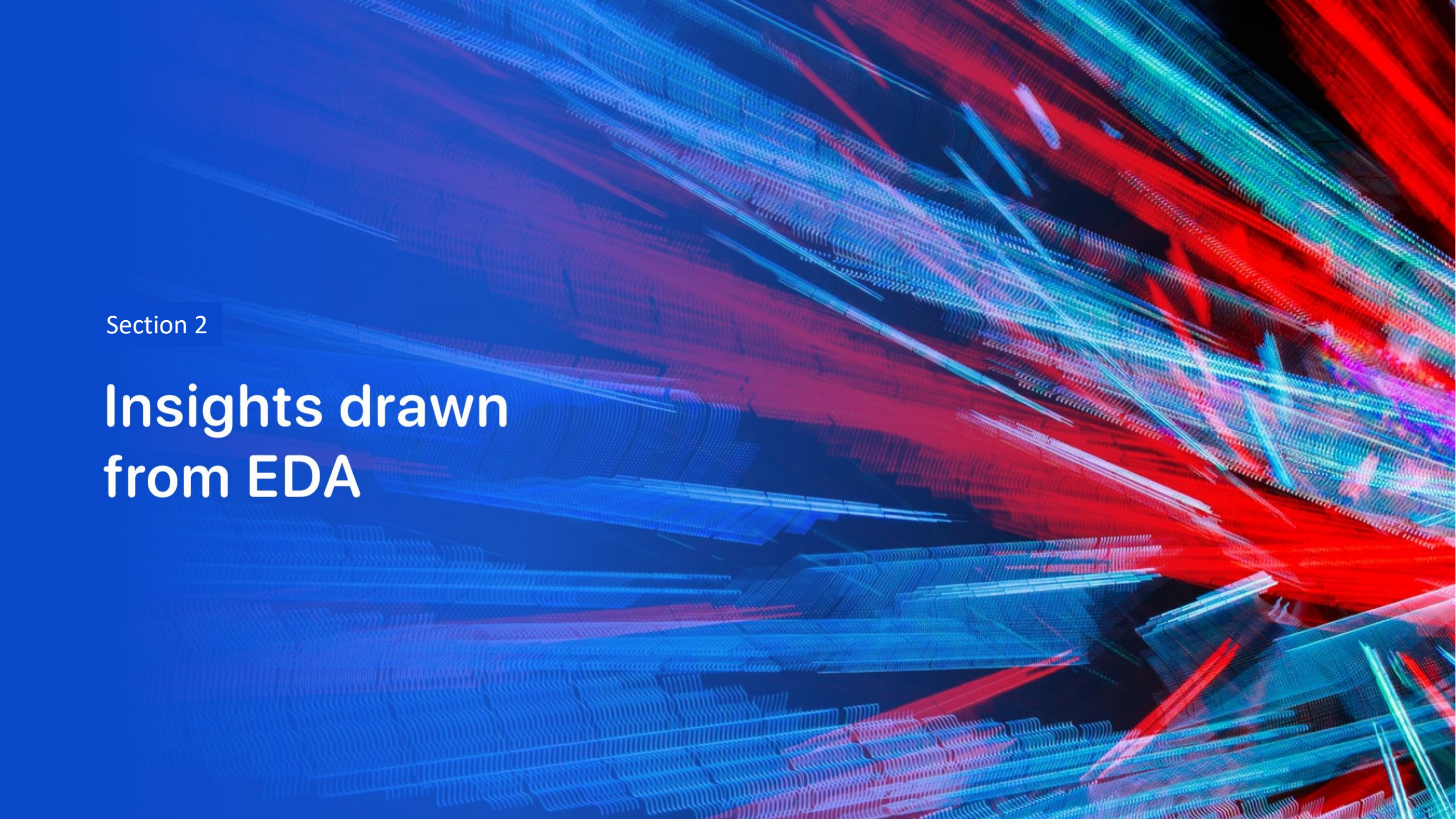
Predictive Analysis (Classification)

- **Pipeline**: **Task 1**: Extract Class column as NumPy array (Y) using to_numpy().
- **Task 2**: Standardize features (X) using StandardScaler.
- **Task 3**: Split data into training (80%) and test (20%) sets (X_train, X_test, Y_train, Y_test) with random_state=2 (18 test samples).
- **Tasks 4–11**: Hyperparameter tuning with GridSearchCV (cv=10) for:
  - **Logistic Regression (Task 4–5)**: Parameters {C: [0.01, 0.1, 1], penalty: ['l2'], solver: ['lbfgs']}.
  - **SVM (Task 6–7)**: Parameters {kernel: ['linear', 'rbf', 'poly', 'sigmoid'], C: logspace(-3, 3, 5), gamma: logspace(-3, 3, 5)}.
  - **Decision Tree (Task 8–9)**: Parameters {criterion: ['gini', 'entropy'], splitter: ['best', 'random'], max_depth: [2, 4, ..., 18], max_features: ['auto', 'sqrt'], min_samples_leaf: [1, 2, 4], min_samples_split: [2, 5, 10]}.
  - **KNN (Task 10–11)**: Parameters {n_neighbors: [1, 2, ..., 10], algorithm: ['auto', 'ball_tree', 'kd_tree', 'brute'], p: [1, 2]}.
- **Evaluation**:
  - Calculate test accuracy using score method.
  - Plot confusion matrices to analyze true positives (landed) vs. false positives (not landed).
  - Logistic Regression issue: False positives (3 cases).
- **Task 12**: Compare model performance to identify the best method.

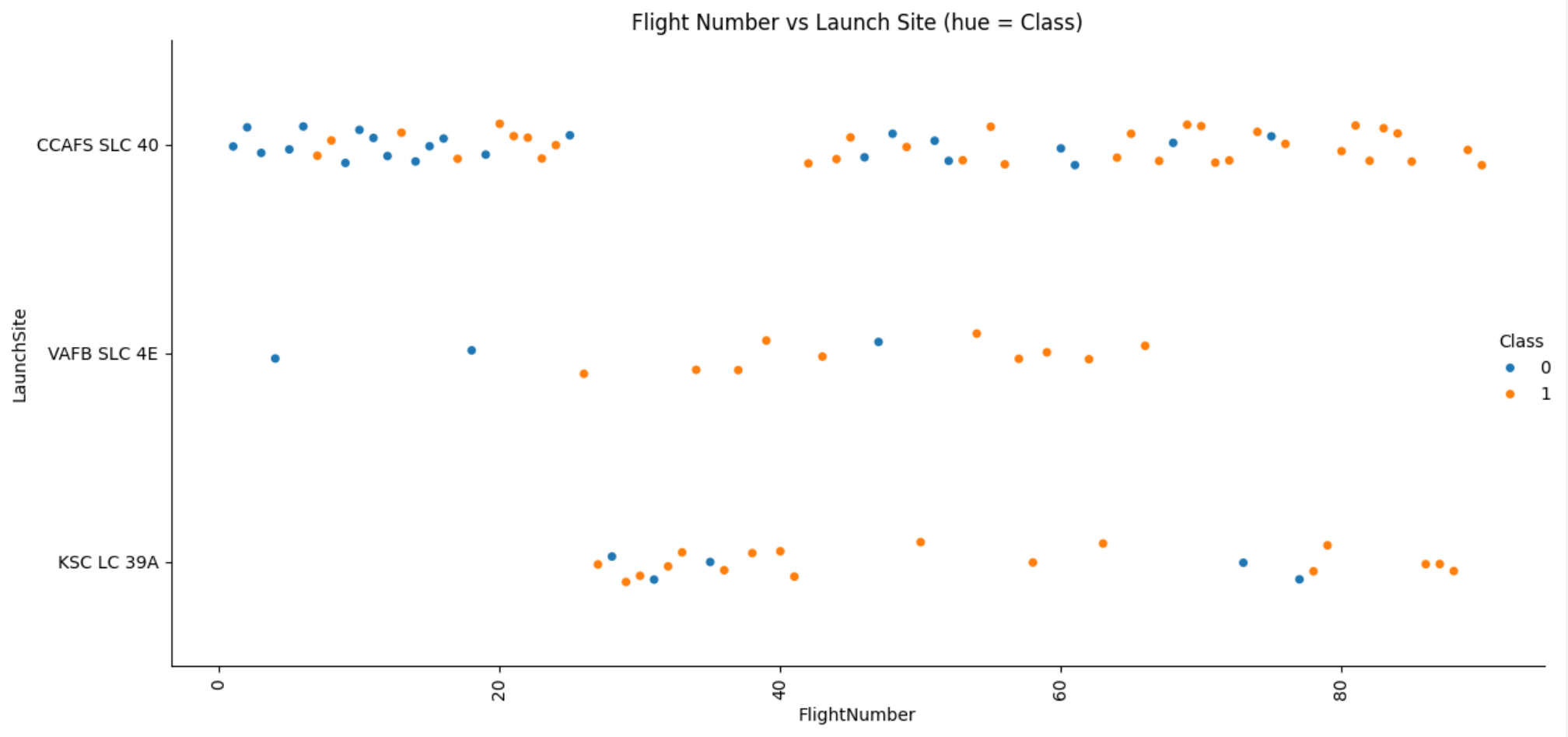
https://github.com/SUSHIL1102/Data-Science-Capstone/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Results

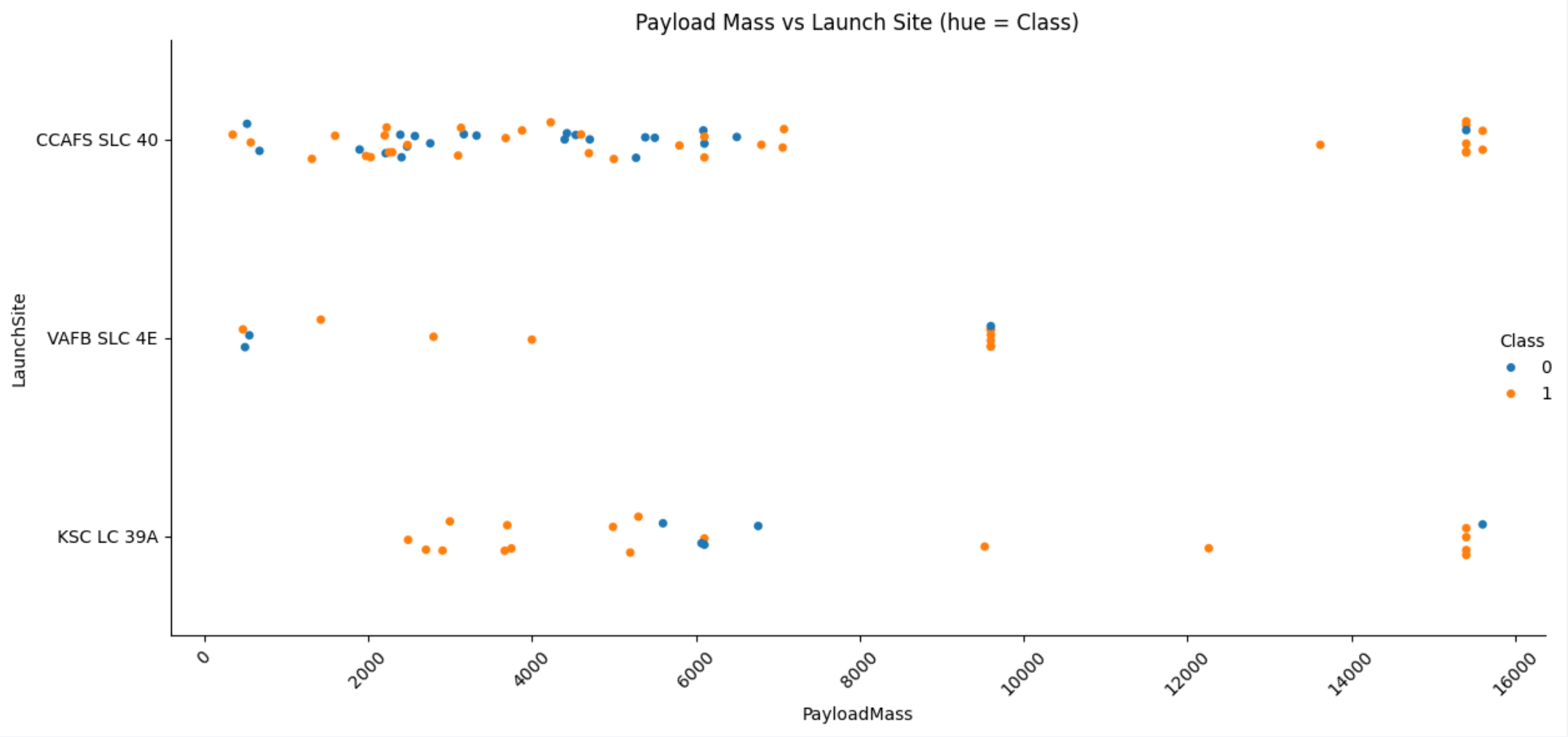
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results


Section 2
Insights drawn from EDA

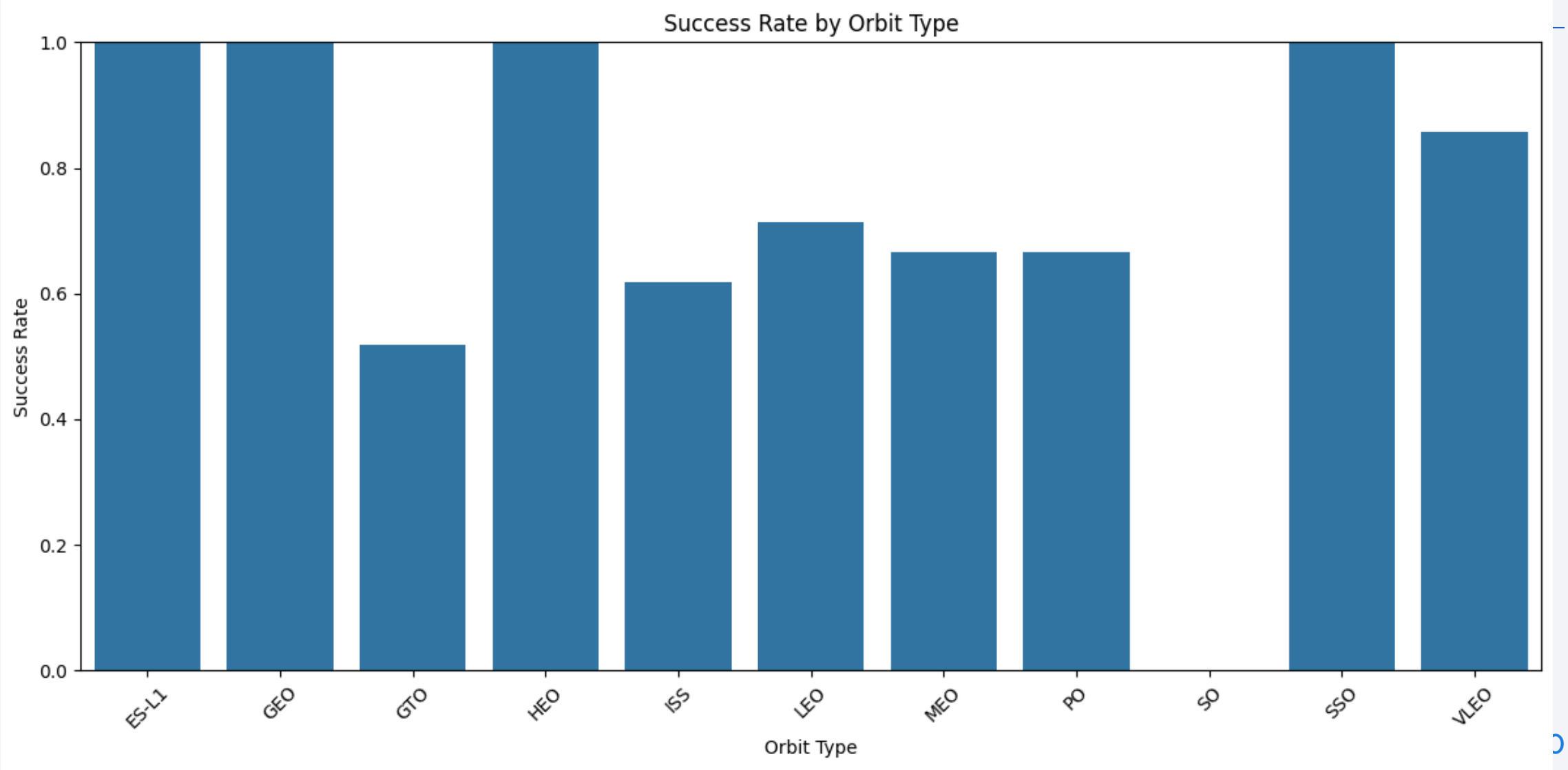
Flight Number vs. Launch Site



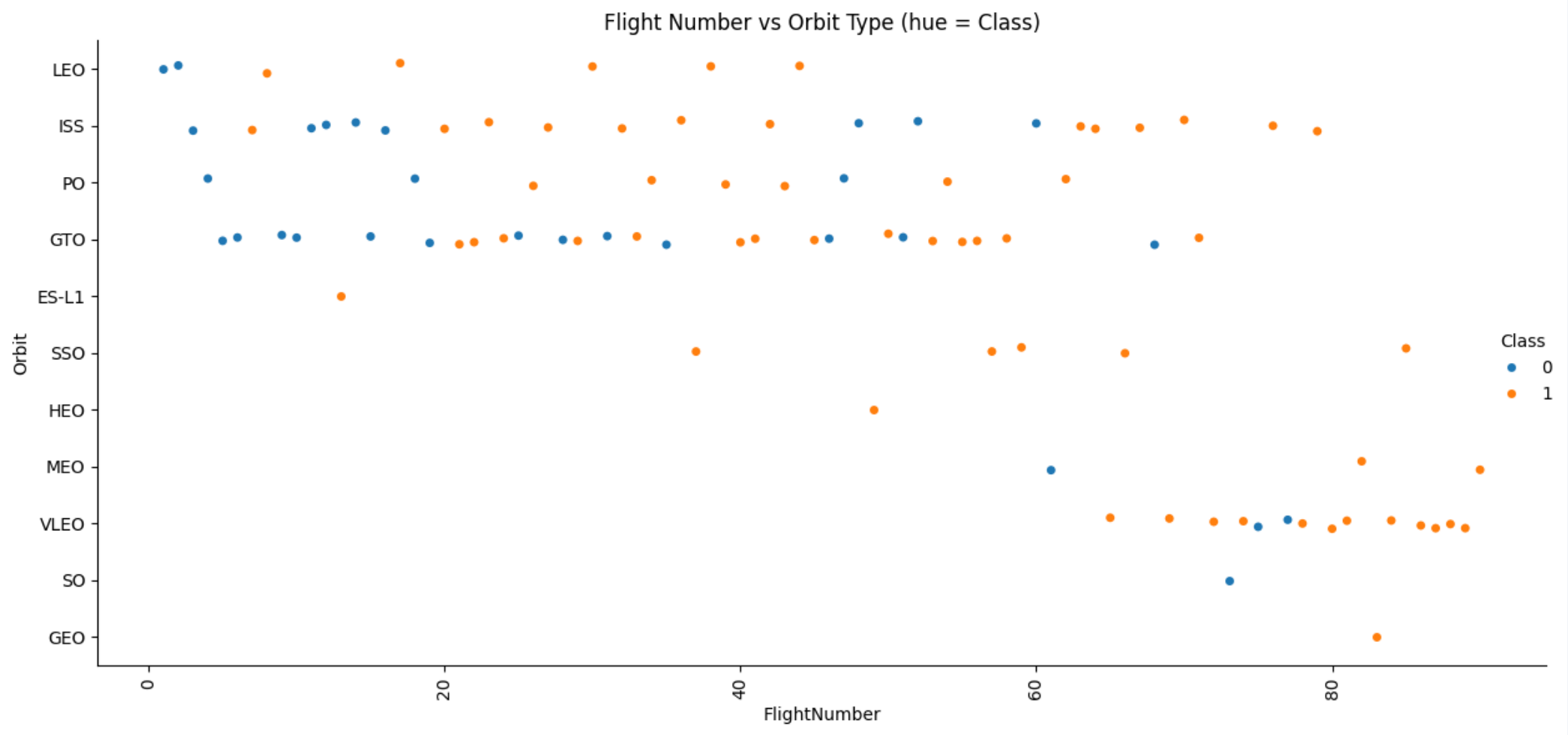
Payload vs. Launch Site



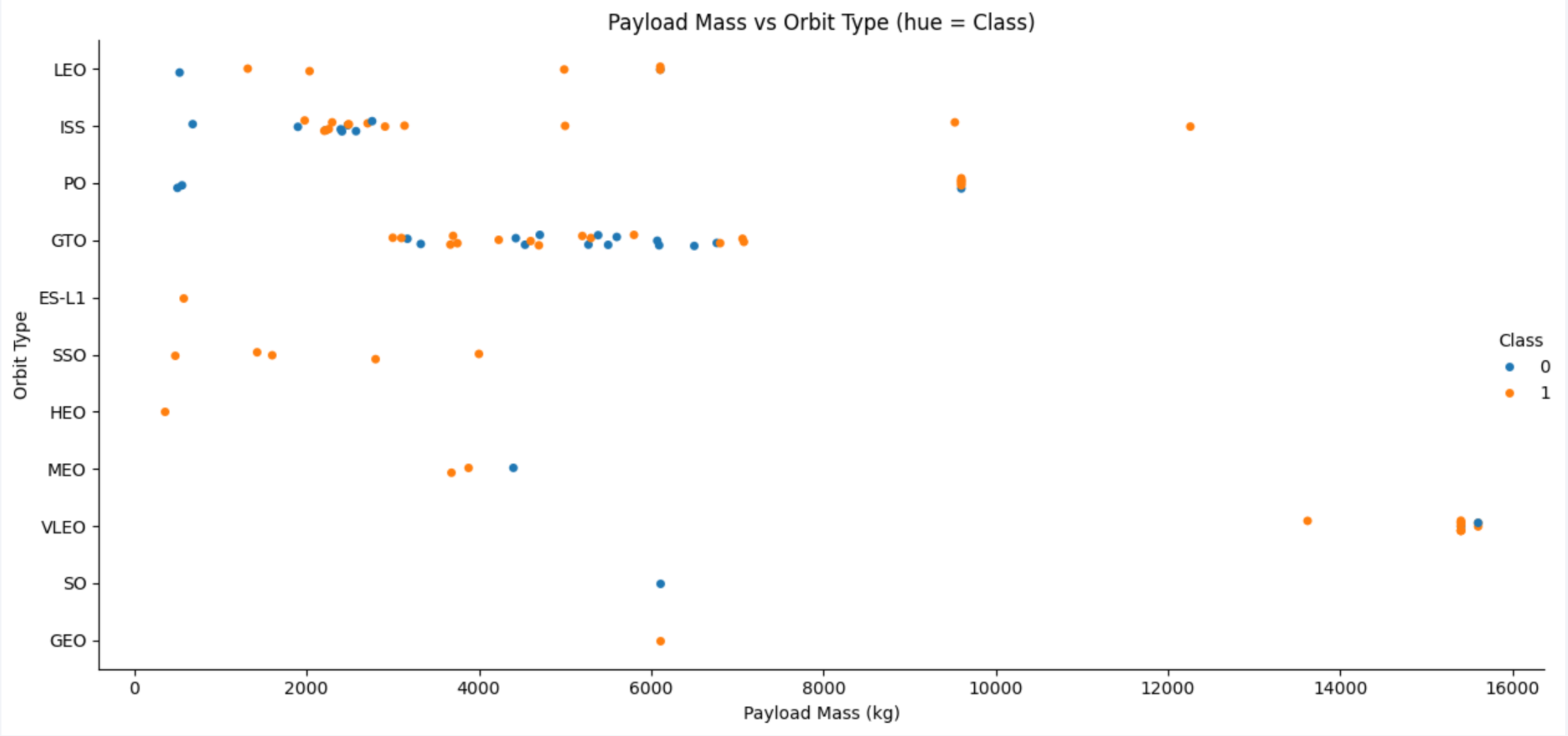
Success Rate vs. Orbit Type



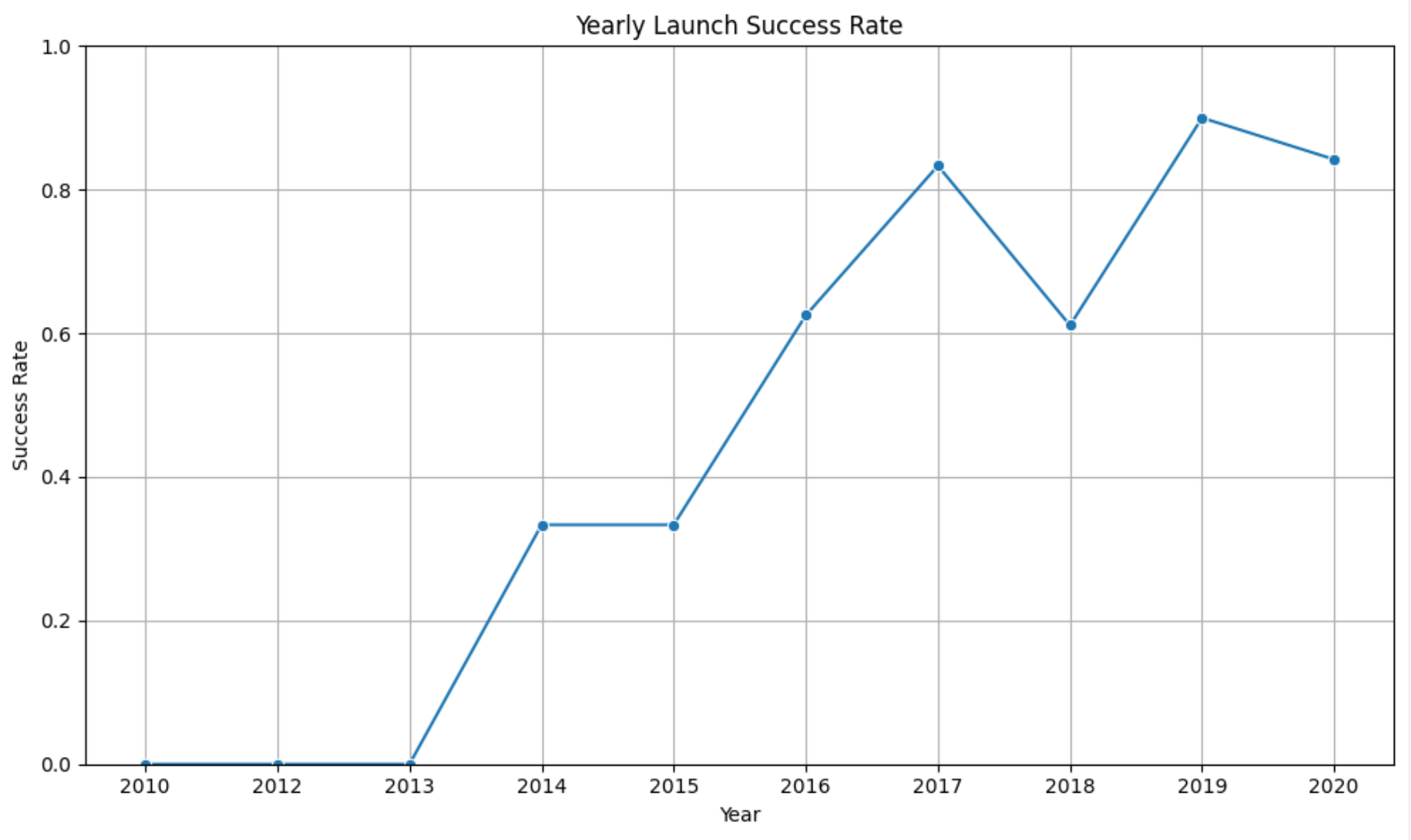
Flight Number vs. Orbit Type



Payload vs. Orbit Type



Launch Success Yearly Trend



All Launch Site Names

```
[14]: %sql SELECT DISTINCT Launch_Site FROM SPACEXTBL
 * sqlite:///my_data1.db
Done.
```

[14]:

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

Launch Site Names Begin with 'CCA'

Task 2

Display 5 records where launch sites begin with the string 'CCA'

```
[5]: %sql SELECT * FROM SPACEXTBL WHERE Launch_Site LIKE 'CCA%' LIMIT 5
```

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

[5]:

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

Total Payload Mass

Task 3

Display the total payload mass carried by boosters launched by NASA (CRS)

18]:
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS total_payload FROM SPACEXTBL WHERE Customer = "NASA (CRS)"
```

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

18]:

total_payload
45596

Task 4

Display average payload mass carried by booster version F9 v1.1

19]:
```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS avg_payload_mass FROM SPACEXTBL WHERE Booster_Version LIKE "F9 v1.1%"
```

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

19]:

avg_payload_mass
2534.6666666666665

Average Payload Mass by F9 v1.1

Task 3

Display the total payload mass carried by boosters launched by NASA (CRS)

18]:
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS total_payload FROM SPACEXTBL WHERE Customer = "NASA (CRS)"
```

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

18]:

total_payload
45596

Task 4

Display average payload mass carried by booster version F9 v1.1

19]:
```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS avg_payload_mass FROM SPACEXTBL WHERE Booster_Version LIKE "F9 v1.1%"
```

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

19]:

avg_payload_mass
2534.6666666666665

First Successful Ground Landing Date

```
In [23]: %sql SELECT MIN(date) AS first FROM SPACEXTBL WHERE Landing_Outcome = "Success (ground pad)"
```

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

Out[23]:

first
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

Task 6

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

In [24]:
```
%sql SELECT Booster_Version FROM SPACEXTBL WHERE Landing_Outcome = "Success (drone ship)" AND PAYLOAD_MASS__KG_ > 4000 AND P
```

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

Out[24]:

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

Task 7

List the total number of successful and failure mission outcomes

In [27]:
```
%sql SELECT Mission_Outcome, COUNT(*) AS TOTAL FROM SPACEXTBL GROUP BY Mission_Outcome
```

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

Out[27]:

Mission_Outcome	TOTAL
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Total Number of Successful and Failure Mission Outcomes

Task 6

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

In [24]:
```
%sql SELECT Booster_Version FROM SPACEXTBL WHERE Landing_Outcome = "Success (drone ship)" AND PAYLOAD_MASS__KG_ > 4000 AND F
```

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

Out[24]:

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

Task 7

List the total number of successful and failure mission outcomes

In [27]:
```
%sql SELECT Mission_Outcome, COUNT(*) AS TOTAL FROM SPACEXTBL GROUP BY Mission_Outcome
```

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

Out[27]:

Mission_Outcome	TOTAL
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Boosters Carried Maximum Payload

Task 8

List all the booster_versions that have carried the maximum payload mass, using a subquery with a suitable aggregate function.

In [28]:
```
%sql SELECT * FROM SPACEXTBL LIMIT 5
```

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

Out[28]:

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

2015 Launch Records

Task 9

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

32]:
```
%sql SELECT substr(Date, 6, 2) AS Month, Landing_Outcome, Booster_Version, Launch_Site FROM SPACEXTBL WHERE substr(Date, 0,
```

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

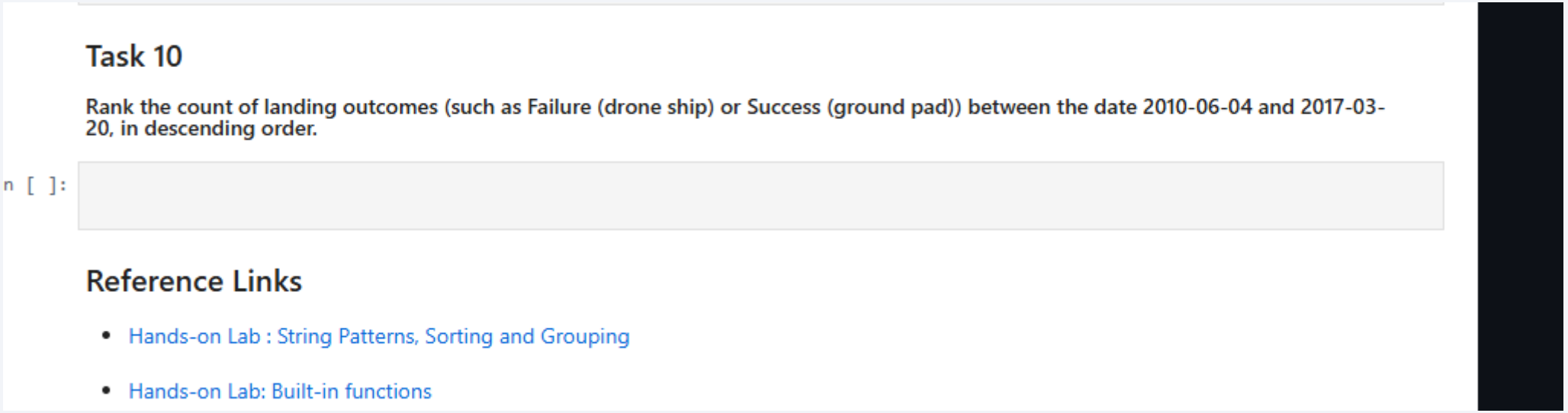
32]:

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

[ ]:

Task 10

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



Section 3

Launch Sites Proximities Analysis

<Folium Map Screenshot 1>

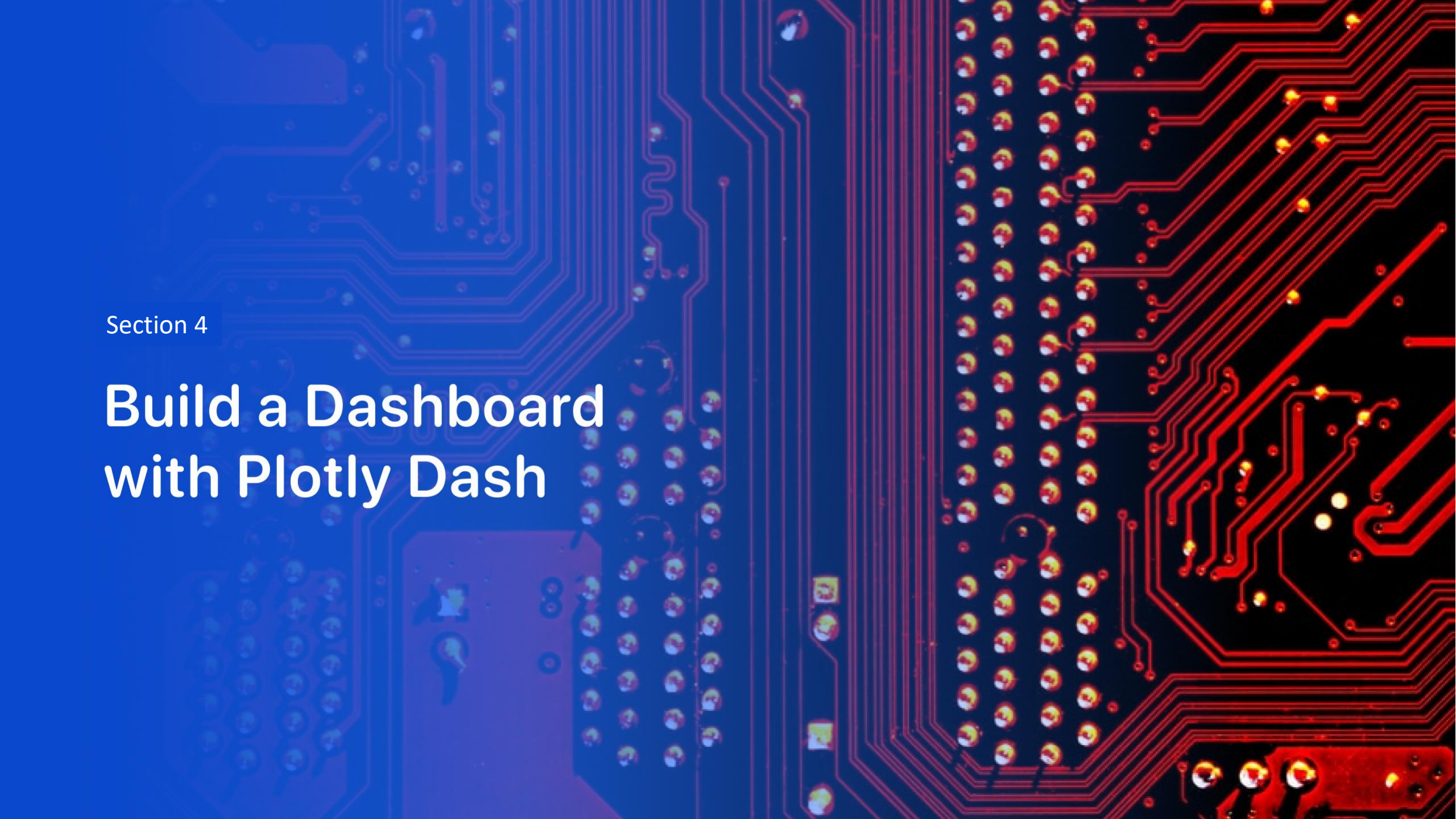
- Replace <Folium map screenshot 1> title with an appropriate title
- Explore the generated folium map and make a proper screenshot to include all launch sites' location markers on a global map
- Explain the important elements and findings on the screenshot

<Folium Map Screenshot 2>

- Replace <Folium map screenshot 2> title with an appropriate title
- Explore the folium map and make a proper screenshot to show the color-labeled launch outcomes on the map
- Explain the important elements and findings on the screenshot

<Folium Map Screenshot 3>

- Replace <Folium map screenshot 3> title with an appropriate title
- Explore the generated folium map and show the screenshot of a selected launch site to its proximities such as railway, highway, coastline, with distance calculated and displayed
- Explain the important elements and findings on the screenshot


Section 4
Build a Dashboard with Plotly Dash

<Dashboard Screenshot 1>

- Replace <Dashboard screenshot 1> title with an appropriate title
- Show the screenshot of launch success count for all sites, in a piechart
- Explain the important elements and findings on the screenshot

<Dashboard Screenshot 2>

- Replace <Dashboard screenshot 2> title with an appropriate title
- Show the screenshot of the piechart for the launch site with highest launch success ratio
- Explain the important elements and findings on the screenshot

<Dashboard Screenshot 3>

- Replace <Dashboard screenshot 3> title with an appropriate title
- Show screenshots of Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider
- Explain the important elements and findings on the screenshot, such as which payload range or booster version have the largest success rate, etc.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

- **Logistic Regression**: Test accuracy unknown; estimated ~77.78% if TN=2, FN=1 (based on TP=12, FP=3, total=18). Validation accuracy (best_score_) not provided.
- **SVM**: Test and validation accuracies not provided.
- **Decision Tree**: Test and validation accuracies not provided.
- **KNN**: Test and validation accuracies not provided.
- **Task 12**: The document aims to compare model performance to find the best method, but no results are given to determine which model performs best.

Confusion Matrix

Confusion Matrix (Logistic Regression):

		Predicted	
		Did Not Land	Landed
True	Did Not Land	?	3
	Landed	?	12

Conclusions

•**Objective Achieved**: The notebook establishes a pipeline to predict Falcon 9 first stage landing success using historical launch data, enabling cost estimation for competitive bidding. The pipeline includes data preprocessing, model training, hyperparameter tuning, and evaluation, aligning with the goal of determining landing success probability.
•**Data Preparation**:
•**Datasets**: Utilized dataset_part_2.csv (containing the Class column for landing success/failure) and dataset_part_3.csv (preprocessed features with one-hot encoding, 80 columns, float64).
•**Preprocessing (Tasks 1–3)**:
    •Extracted Class as the target variable (Y) using to_numpy().
    •Standardized features (X) with StandardScaler for consistent scaling across models.
    •Split data into 80% training and 20% test sets (18 test samples) with random_state=2, ensuring reproducibility.
•**Model Development and Evaluation**:
•**Models Tested (Tasks 4–11)**: Four classification algorithms were implemented with hyperparameter tuning via GridSearchCV (10-fold cross-validation):
    •**Logistic Regression**: Tuned for C, penalty, and solver. The confusion matrix showed 12 true positives and 3 false positives, indicating good prediction of landings but issues with false positives (predicting "landed" when it did not land).
    •**SVM**: Tuned for kernel, C, and gamma. No specific performance details provided.
    •**Decision Tree**: Tuned for criterion, splitter, max_depth, max_features, min_samples_leaf, and min_samples_split. No specific performance details provided.
    •**KNN**: Tuned for n_neighbors, algorithm, and p. No specific performance details provided.
•**Evaluation Metrics**: Each model's test accuracy was to be calculated using the score method, and confusion matrices were plotted to assess true positives, false positives, etc. However, exact accuracies and best parameters are not provided in the document.

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!