DS Capstone Project

- Lisa Yvette Inyange
- 28/04/2025

IBM Developer
SKILLS NETWORK

Outline

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

Executive Summary

- This project focused on analyzing SpaceX launch data to understand and predict rocket landing success.

Executive Summary

Summary of methodologies:

- Data was collected from the public SpaceX API and the SpaceX Wikipedia page.
- A 'class' label was created to classify successful landings.
- Exploratory Data Analysis (EDA) was conducted using SQL, data visualization techniques, Folium maps, and Plotly dashboards.
- Relevant columns were selected as features, and categorical variables were transformed into binary variables using one-hot encoding.
- The data was standardized, and GridSearchCV was used to determine the optimal parameters for machine learning models.
- The accuracy scores of the models were visualized.

Executive Summary

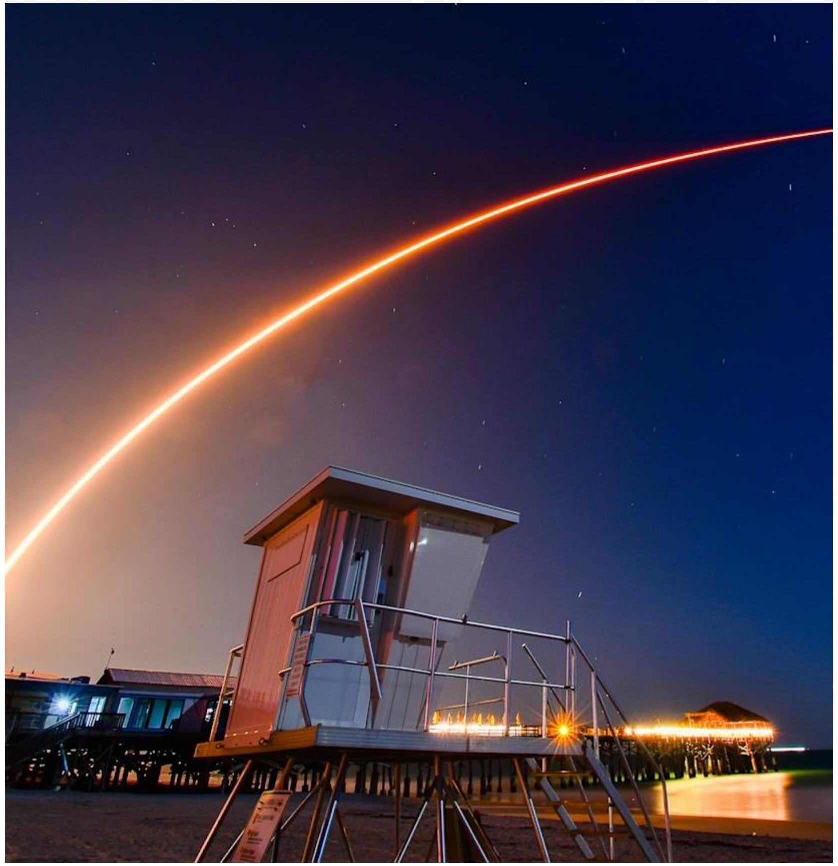
Summary of all results:

- Four machine learning models were developed: Logistic Regression, Support Vector Machine, Decision Tree Classifier, and K-Nearest Neighbors.
- All models achieved similar accuracy rates of approximately 83.33%.
- The models tended to over-predict successful landings.
- The analysis suggests that more data is needed to improve model accuracy and achieve better model differentiation.

Introduction

Project background and context i.e The Rise of Commercial Space

- Increased demand for launches (satellites, exploration, space tourism)
- SpaceX's innovation: Reusable rockets (Falcon 9, Starship)
- SpaceX cost advantage: $62M vs. $165M (industry average)
- Other players: Blue Origin, Rocket Lab, etc.



Introduction

Analysis Focus: Key Questions

- Key Questions:
  - What factors drive landing success? (e.g., weather, mission type, landing location)
  - Can we accurately predict outcomes? (using machine learning)
  - How can this optimize costs? (for SpaceX and competitors)
  - How can Space X further improve the costs? (e.g. refining landing procedures, improving hardware)
  - How can competitors position themselves against Space X? (e.g., highlighting different strengths, targeting specific markets)

Introduction

- **The Challenge of Landing Prediction**
  - Space Y's goal: Compete with SpaceX
  - Key factor: Predicting first-stage recovery (reduces launch cost)
  - Our task: Develop a predictive model (to aid Space Y's strategic planning)
  - Challenges: Data availability, model accuracy, real-time prediction
- **Value Proposition**
  - For SpaceX: Optimize recovery, reduce costs, improve mission reliability
  - For Competitors: Strategic benchmarking, market positioning, identify competitive advantages

Methodology

Methodology

- **Data Collection:**
  - Combined data from SpaceX API and Wikipedia.
  - SpaceX API for detailed launch data.
  - Wikipedia for supplementary information.
- **Data Wrangling:**
  - Processed data for analysis.
  - Classified landings as successful/unsuccessful.
  - Handled missing values, inconsistencies; transformed data.

Methodology

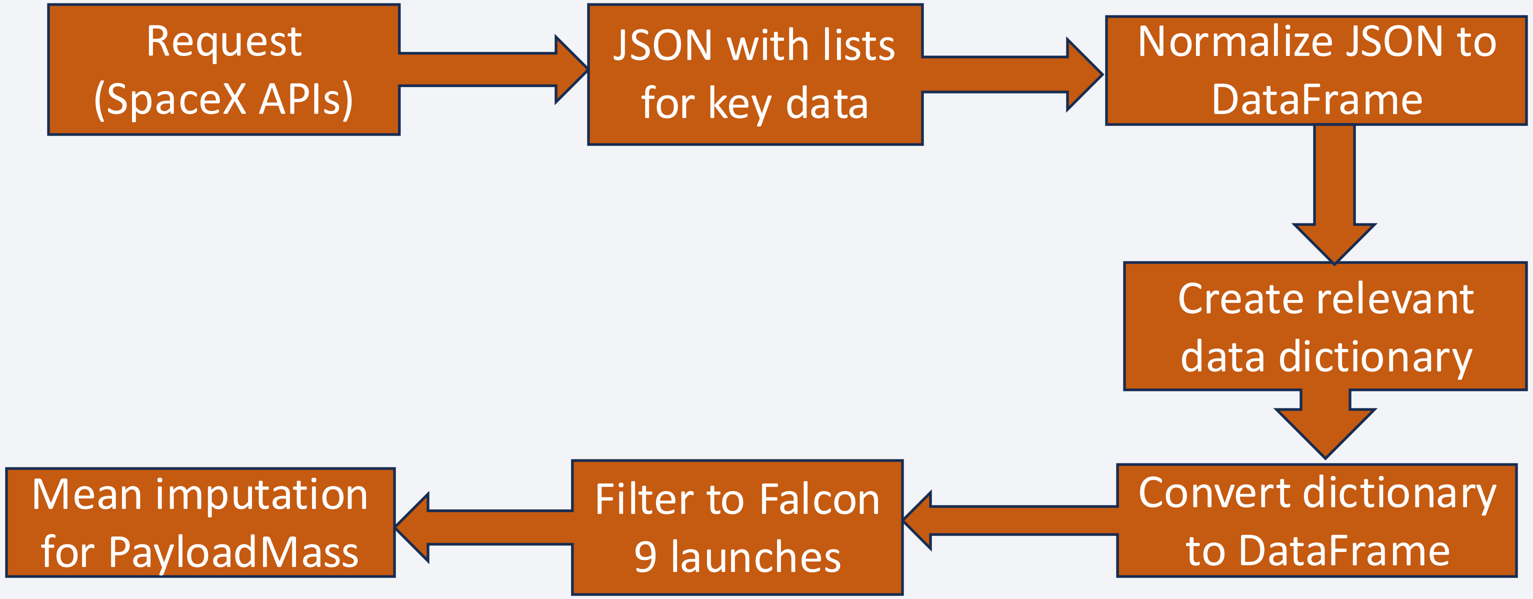
- **Exploratory Data Analysis (EDA):**
  - Visualization and SQL to identify key factors.
- **Interactive Visual Analytics:**
  - Folium for interactive maps of launch sites.
  - Plotly Dash for dynamic data exploration.
- **Predictive Analysis:**
  - Classification models (e.g., logistic regression, decision trees).
  - Model building, tuning (GridSearchCV), and evaluation.

Data Collection Overview

Data collection process involved combining data from the SpaceX public API and scraping data from a table on SpaceX's Wikipedia page.

The process ensured a comprehensive dataset for analysis, capturing both structured API data and tabular web data.

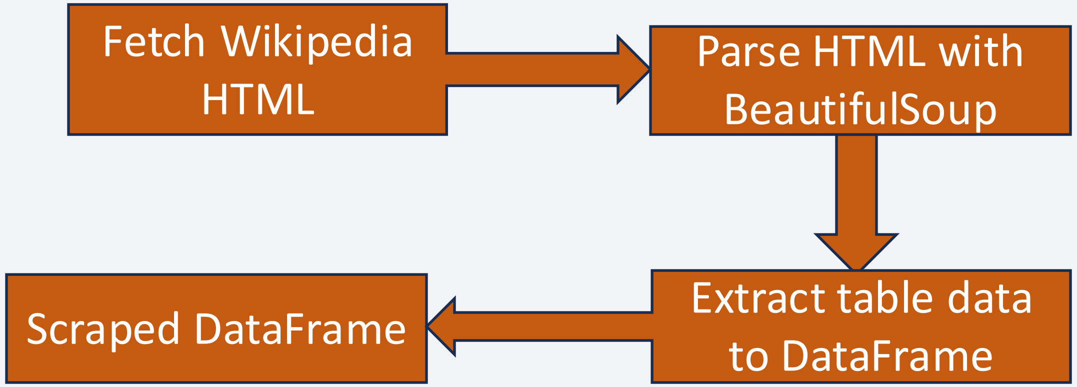
Data Collection – SpaceX API



Flowchart of SpaceX API calls

Link to GitHub URL: https://github.com/ILisa250/IBM_Data_Science/blob/main/jupyter-labs-spacex-data-collection-api%20(1).ipynb

Data Collection – Scraping



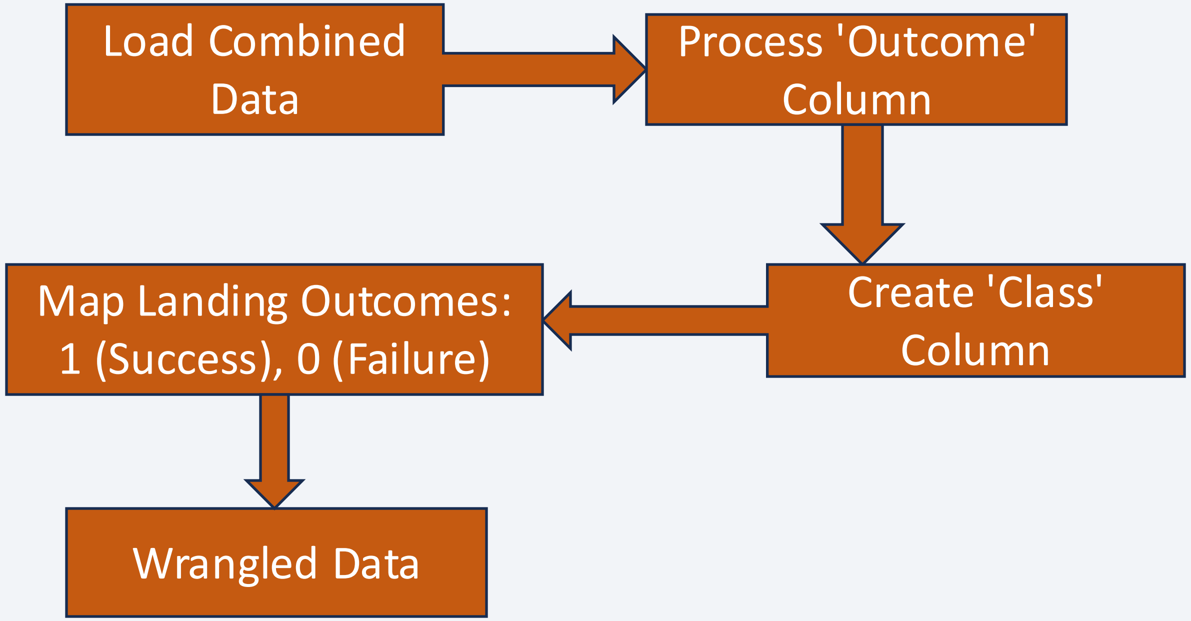
flowchart of web scraping

Link to GitHub URL: https://github.com/ILisa250/IBM_Data_Science/blob/main/jupyter-labs-webscraping%20(3).ipynb

Data Wrangling

Data wrangling involved transforming the collected data into a suitable format for analysis and machine learning

- Value mapping: Successful Landings ('class' = 1): True ASDS, True RTLS, True Ocean
- Unsuccessful Landings ('class' = 0): None None, False ASDS, None ASDS, False Ocean, False RTLS

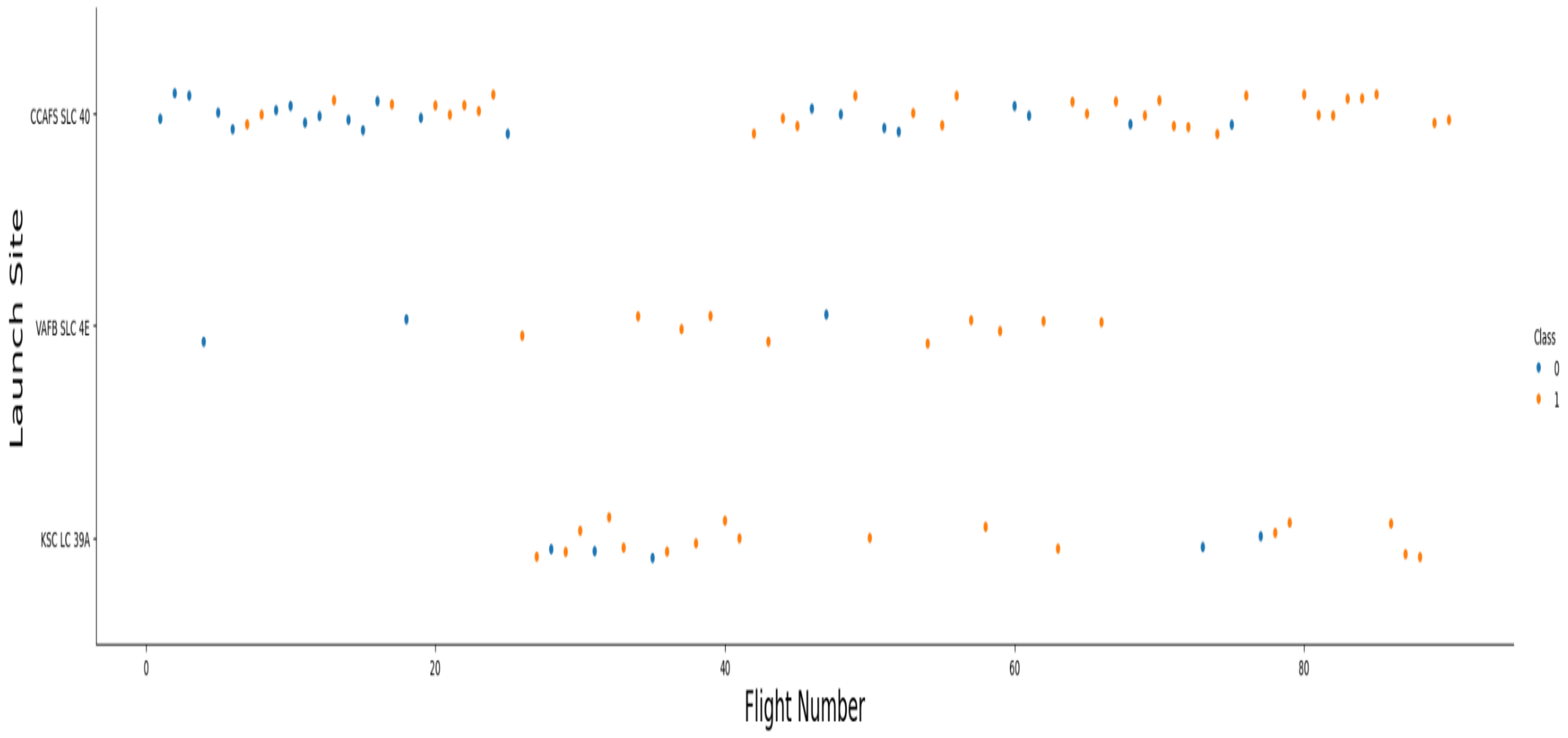


flowchart of Data Wrangling

Link to GitHub URL: https://github.com/ILisa250/IBM_Data_Science/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

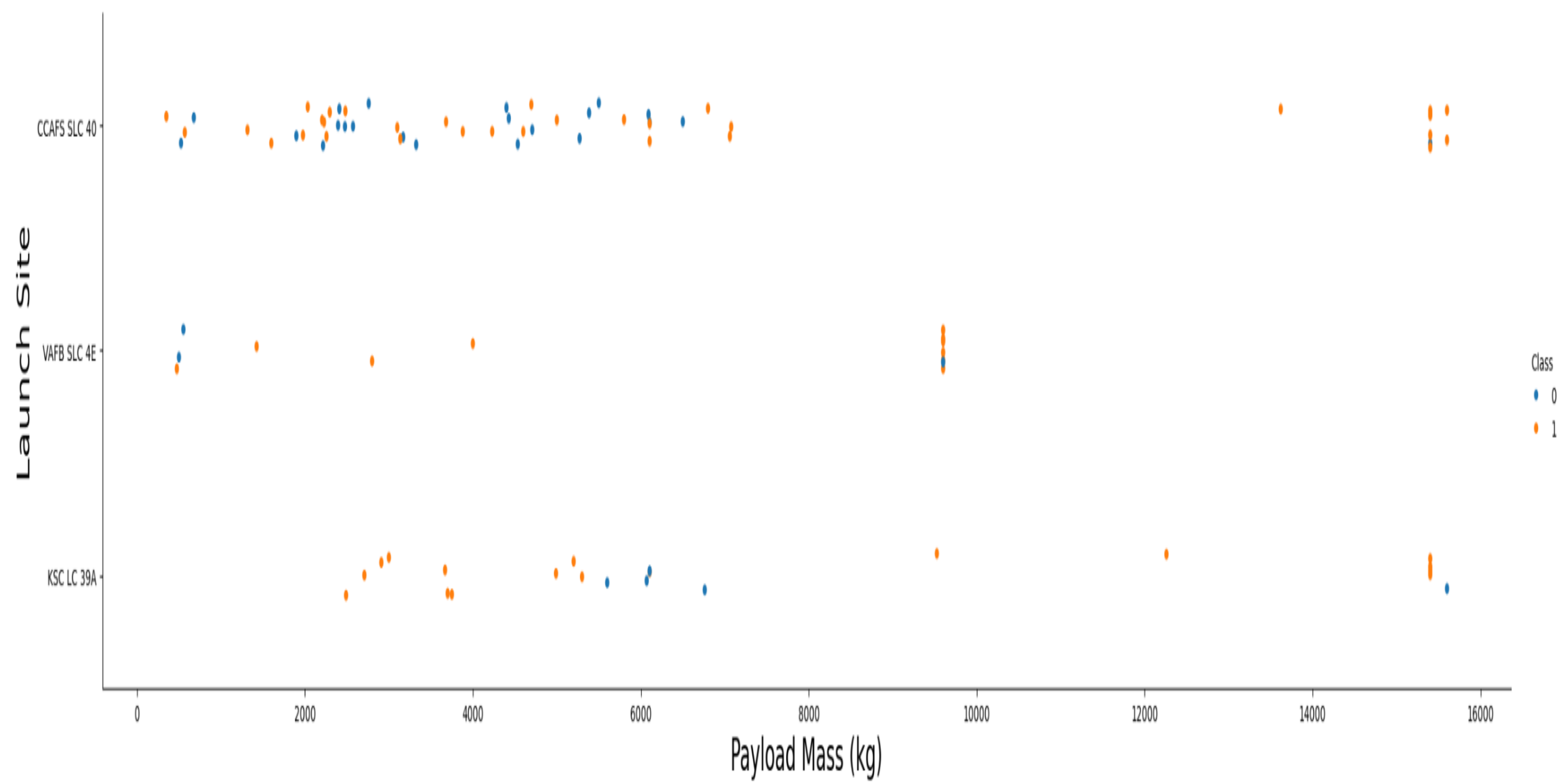
Relationship between Flight Number and Launch Site



Link to GitHub URL: https://github.com/ILisa250/IBM_Data_Science/blob/main/edadataviz%20(1).ipynb

EDA with Data Visualization

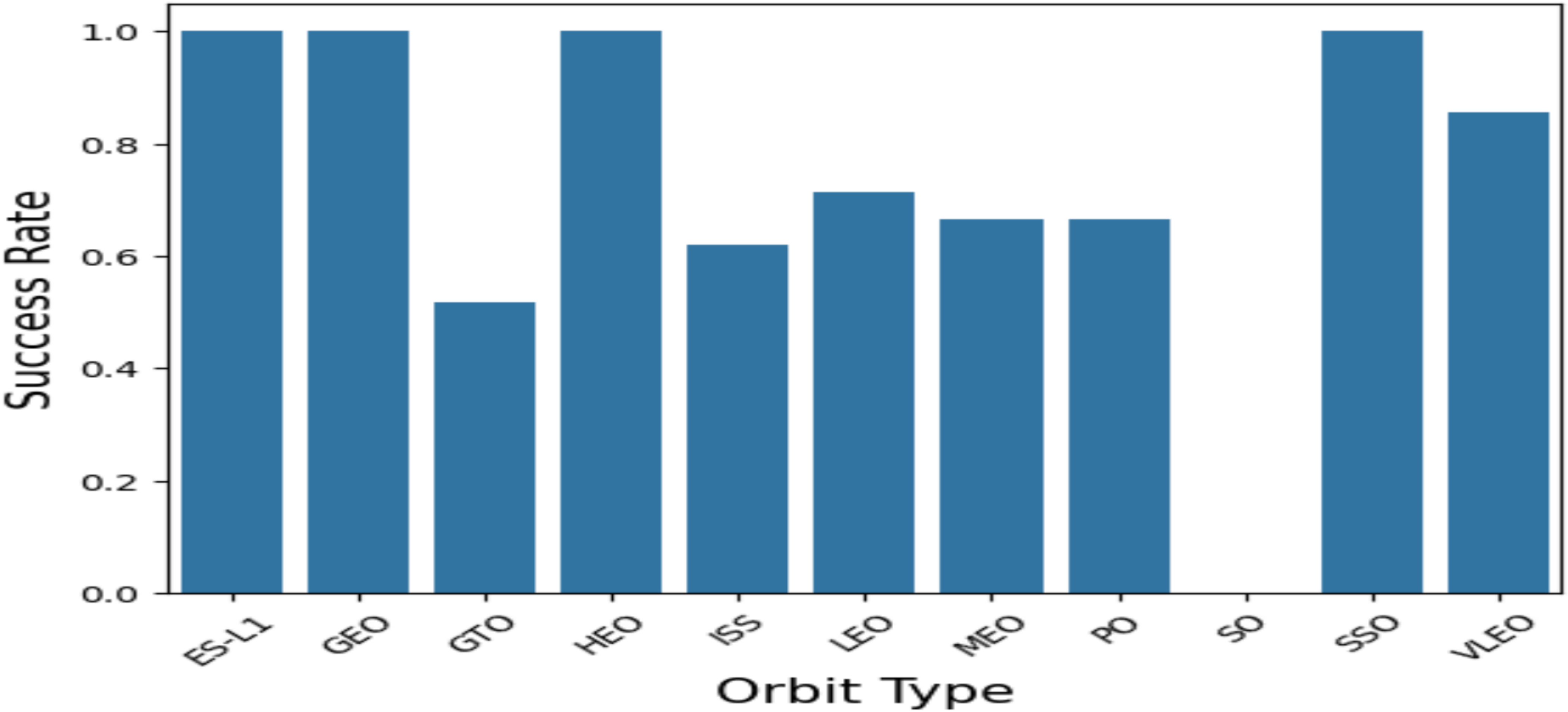
Relationship between Payload Mass and Launch Site



Link to GitHub URL: https://github.com/ILisa250/IBM_Data_Science/blob/main/edadataviz%20(1).ipynb

EDA with Data Visualization

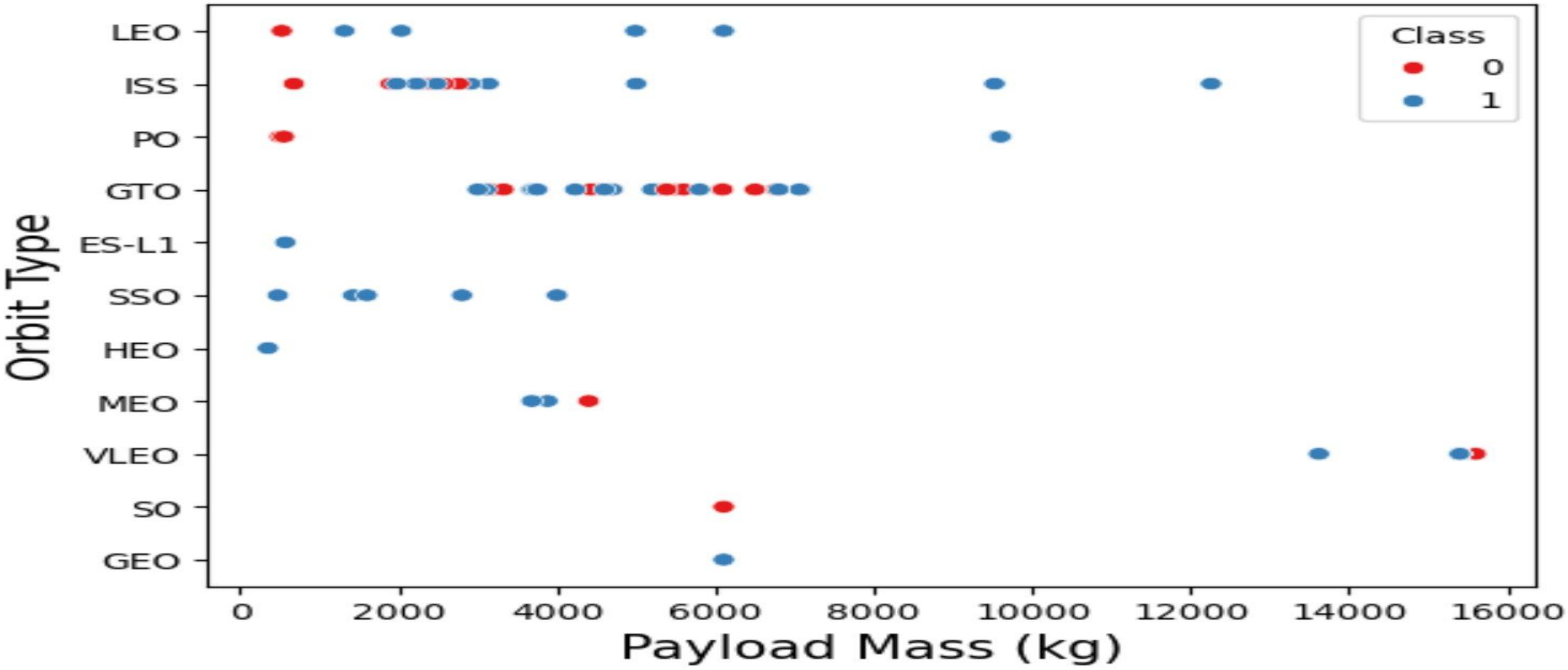
Relationship between success rate of each orbit type



Link to GitHub URL: https://github.com/ILisa250/IBM_Data_Science/blob/main/edadataviz%20(1).ipynb

EDA with Data Visualization

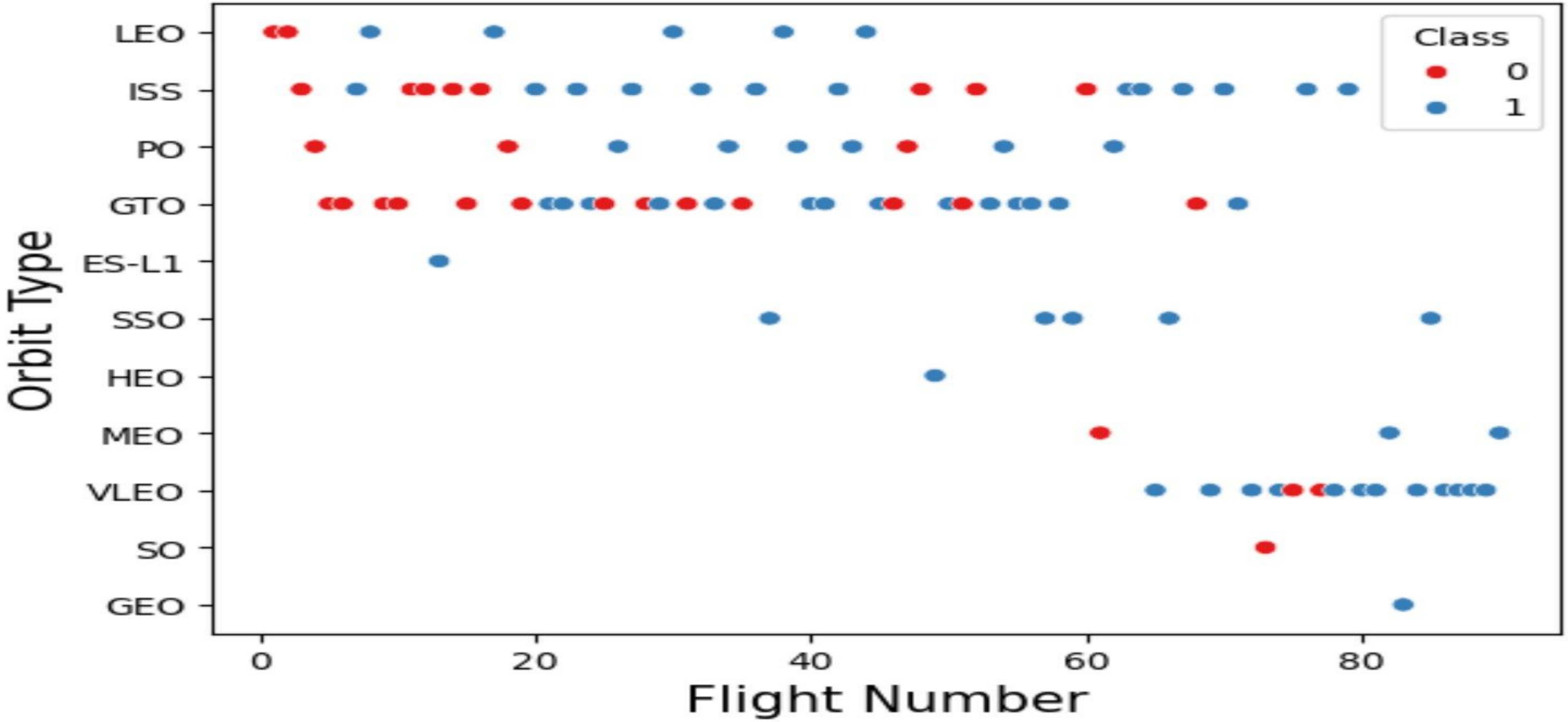
Relationship between Payload Mass and Orbit type



Link to GitHub URL: https://github.com/ILisa250/IBM_Data_Science/blob/main/edadataviz%20(1).ipynb

EDA with Data Visualization

Relationship between Flight Number and Orbit type



Link to GitHub URL: https://github.com/ILisa250/IBM_Data_Science/blob/main/edadataviz%20(1).ipynb

EDA with Data Visualization

Launch success yearly trend

Link to GitHub URL: https://github.com/ILisa250/IBM_Data_Science/blob/main/edadataviz%20(1).ipynb

EDA with SQL

SQL queries were used to explore the SpaceX dataset and gain insights. The following are example queries that were executed:

- Displaying unique launch site names.
- Showing 5 records where launch sites begin with 'CCA'."
- "Calculating total payload mass for NASA (CRS) launches.
- "List boosters with successful drone ship landings and payload mass between 4000 and 6000."
- Listing booster versions with the maximum payload mass (using a subquery).

Link to GitHub URL:
https://github.com/ILisa250/IBM_Data_Science/blob/main/jupyter-labs-eda-sql-coursera_sqllite%20(1).ipynb

Launch Site Names Begin with 'CCA'

5 records where launch sites begin with `CCA`

```
('2010-06-04', '18:45:00', 'F9 v1.0  B0003', 'CCAFS LC-40', 'Dragon Spacecraft Qualification Unit', 0, 'LEO', 'SpaceX', 'Suc
cess', '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)', 'Succes
s', '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 at
tempt')
('2013-03-01', '15:10:00', 'F9 v1.0  B0007', 'CCAFS LC-40', 'SpaceX CRS-2', 677, 'LEO (ISS)', 'NASA (CRS)', 'Success', 'No a
ttempt')
```

Total Payload Mass i.e Carried by boosters from NASA

```
cursor.execute('''
SELECT SUM("Payload_Mass_(kg)") AS total_payload_mass
FROM SPACEXTABLE
WHERE "Launch_Provider" = 'NASA (CRS)';
''')

# Fetch the result
total_payload_mass = cursor.fetchone()[0]

print(f"Total Payload Mass carried by boosters launched by NASA (CRS): {total_payload_mass} kg")

# Close the connection
conn.close()
```

First Successful Ground Landing Date

Date of the first successful landing outcome on ground pad

```
conn = sqlite3.connect('my_data1.db')
cursor = conn.cursor()

cursor.execute('''
SELECT MIN("Date") AS first_successful_landing_date
FROM SPACEXTABLE
WHERE "Landing_Outcome" = 'Success (ground pad)';
''')

# Fetch the result
first_successful_landing_date = cursor.fetchone()[0]

print(f"First successful landing date on ground pad: {first_successful_landing_date}")

conn.close()
```

```
First successful landing date on ground pad: 2015-12-22
```

2015 Launch Records

List of failed landing outcomes in drone ship, their booster versions, and launch site names for in year 2015

```
FROM SPACEXTABLE
WHERE substr("Date", 0, 5) = '2015'
  AND "Landing_Outcome" = 'Failure (drone ship)';
''')
records = cursor.fetchall()
print("Records of failure landing outcomes on drone ship in 2015:")
for record in records:
    print(f"Month: {record[0]}, Booster Version: {record[1]}, Launch Site: {record[2]}")

conn.close()
```

```
Records of failure landing outcomes on drone ship in 2015:
Month: January, Booster Version: Failure (drone ship), Launch Site: F9 v1.1 B1012
Month: April, Booster Version: Failure (drone ship), Launch Site: F9 v1.1 B1015
```

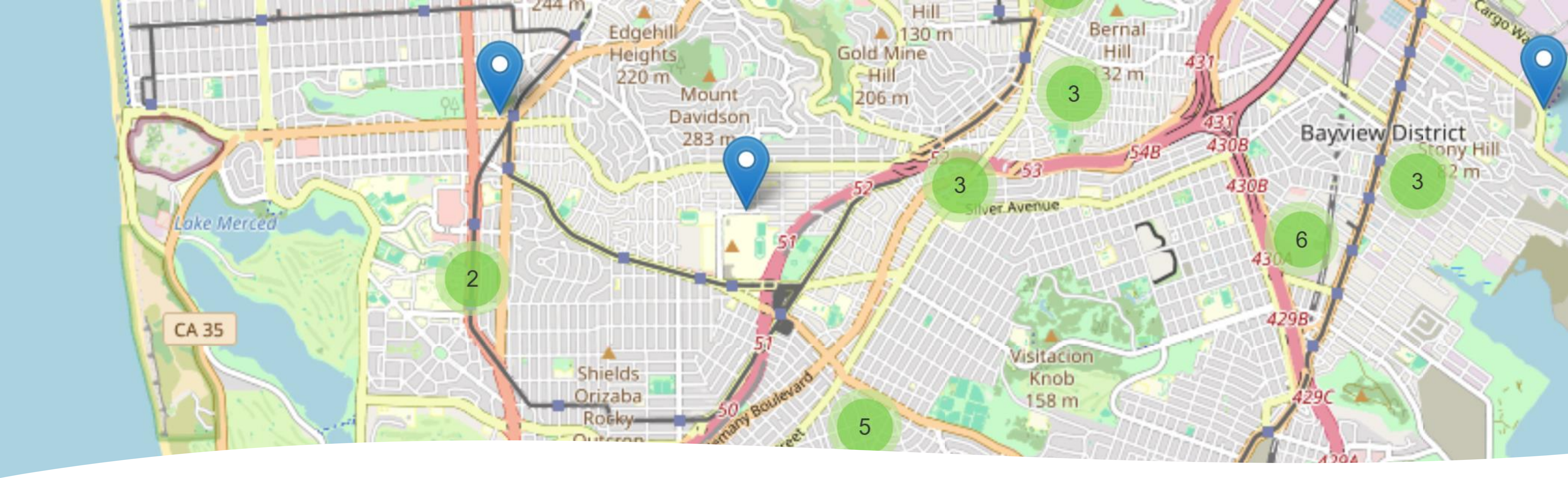
Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

- There are two types of successful landing outcomes: drone ship and ground pad
- There were 8 successful landings in total during this time period

```
cursor.execute('''
SELECT "Landing_Outcome", COUNT(*) AS total_count
FROM SPACEXTABLE
WHERE "Date" BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY "Landing_Outcome"
ORDER BY total_count DESC;
''')
outcomes = cursor.fetchall()
print("Ranked landing outcomes between 2010-06-04 and 2017-03-20:")
for outcome in outcomes:
    print(f"{outcome[0]}: {outcome[1]} occurrences")

conn.close()
```

```
Ranked landing outcomes between 2010-06-04 and 2017-03-20:
No attempt: 10 occurrences
Success (drone ship): 5 occurrences
Failure (drone ship): 5 occurrences
Success (ground pad): 3 occurrences
Controlled (ocean): 3 occurrences
Uncontrolled (ocean): 2 occurrences
Failure (parachute): 2 occurrences
Precluded (drone ship): 1 occurrences
```



Summary of Map Objects in Folium

1. **CircleMarker:**
   - **What it is:** Circular markers used to represent points on the map.
   - **Why added:** To visually indicate incident locations with customizable size, color, and opacity
2. **Marker (with Pop-up):**
   - **What it is:** Map markers that can display additional information when clicked.
   - **Why added:** To show details (e.g., incident category) when clicking on the markers.

Link to GitHub URL: https://github.com/ILisa250/IBM_Data_Science/blob/main/DV0101EN-Exercise-Generating-Maps-in-Python.ipynb

Summary of Map Objects in Folium

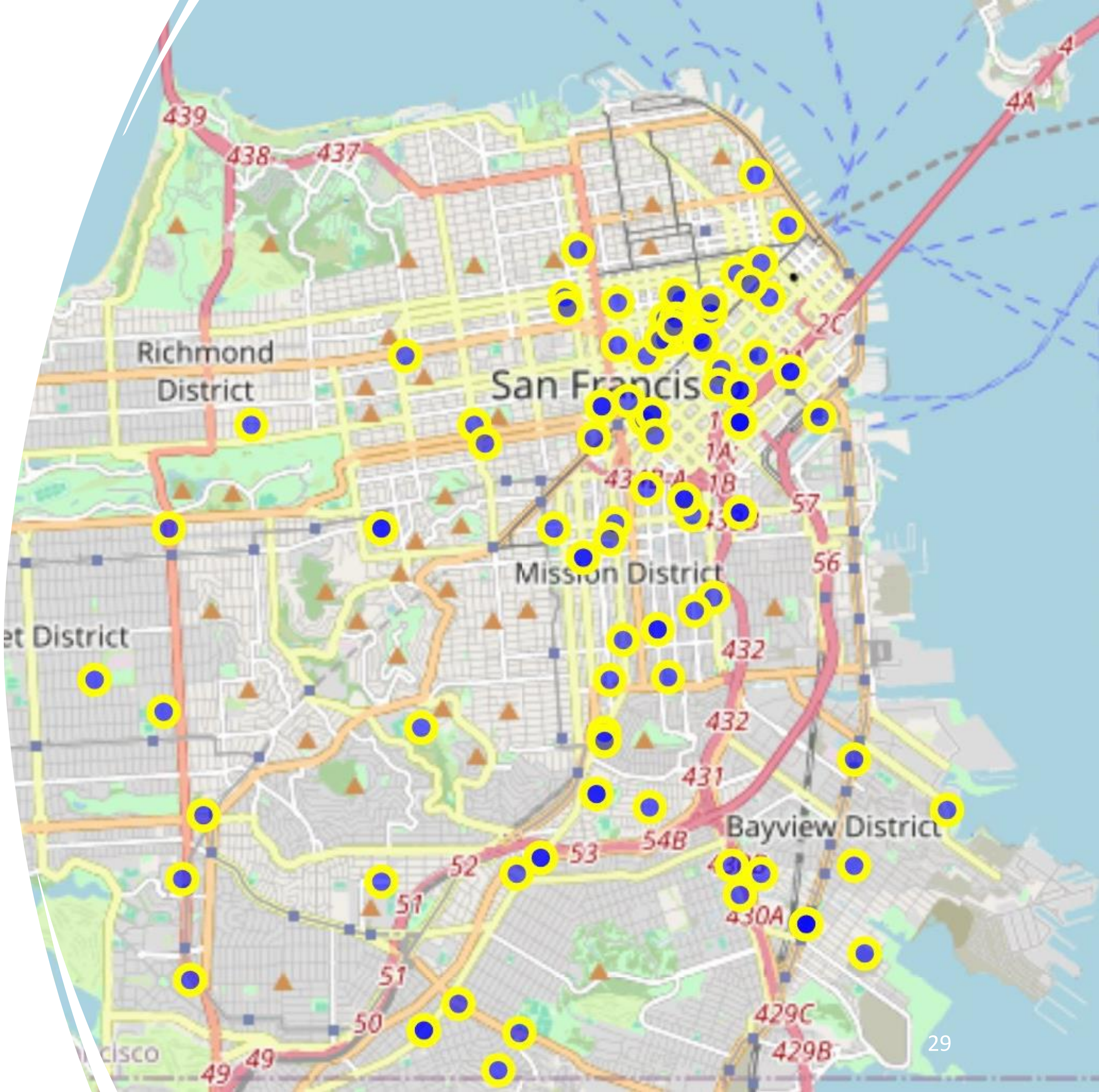
3. FeatureGroup:

- **What it is:** A group to organise multiple map elements
- **Why added:** To manage and style the collection of incident markers as a single unit

4. MarkerCluster:

- **What it is:** A plugin that groups nearby markers into clusters
- **Why added:** To avoid map clutter and improve navigation by clustering markers at lower zoom levels

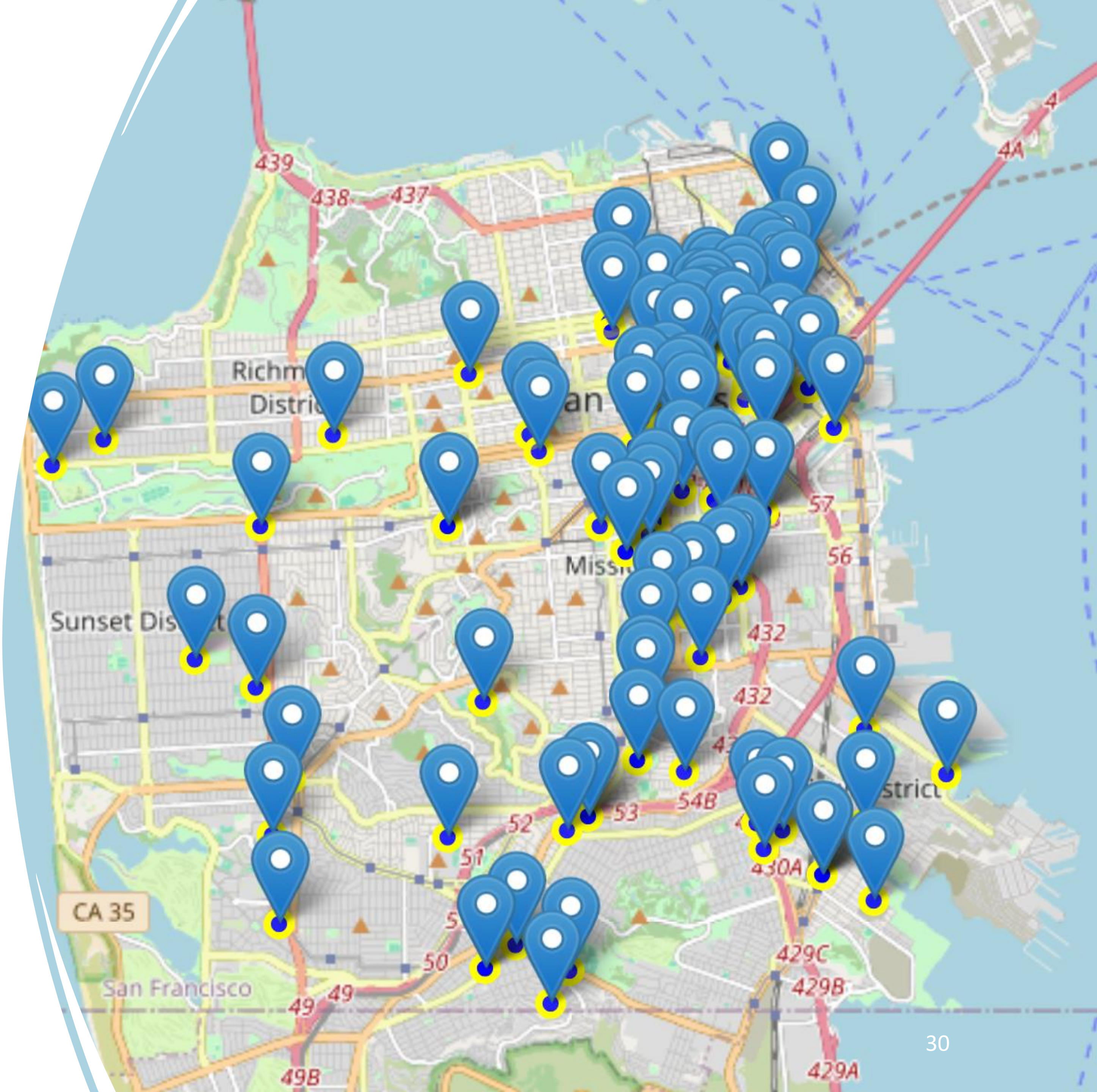
Link to GitHub URL:
https://github.com/ILisa250/IBM_Data_Science/blob/main/DV0101EN-Exercise-Generating-Maps-in-Python.ipynb



Summary of Map Objects in Folium

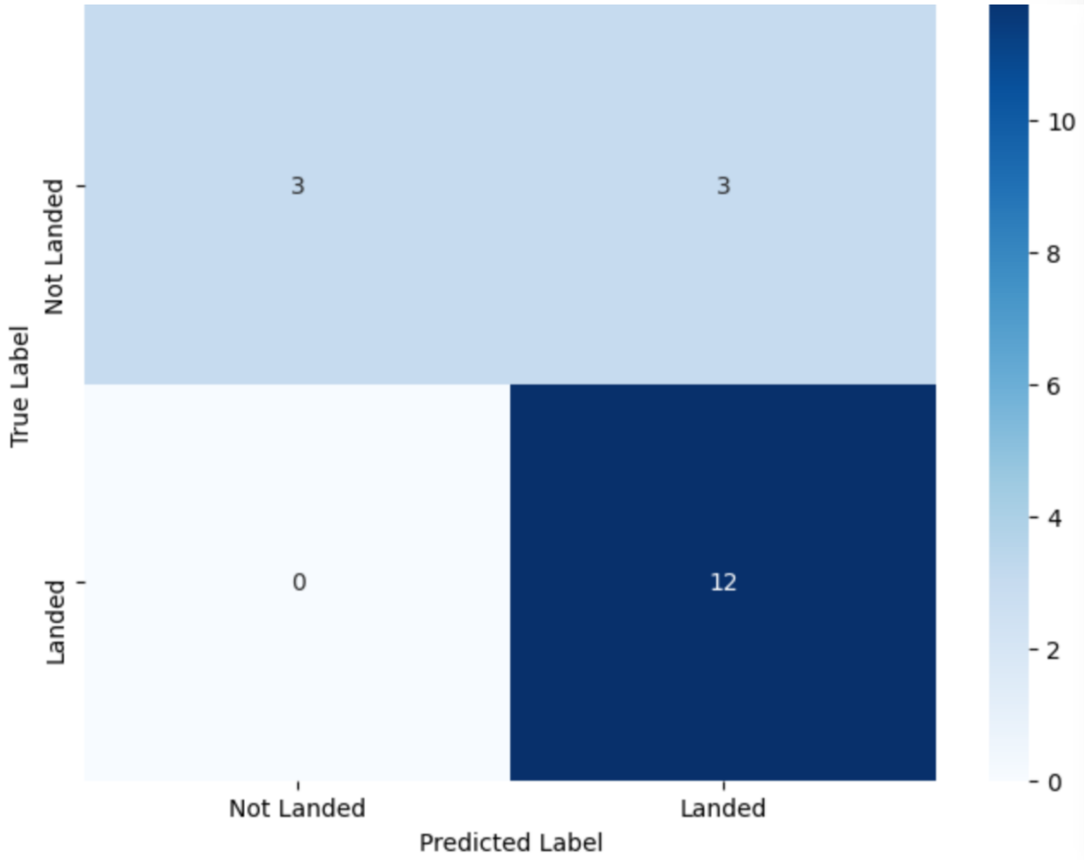
5. Map:

- **What it is:** The base object that holds all map elements
- **Why added:** To define the center location and zoom level for the map's initial view.



Link to GitHub URL:
https://github.com/ILisa250/IBM_Data_Science/blob/main/DV0101EN-Exercise-Generating-Maps-in-Python.ipynb

Predictive Analysis (Classification)




```python
svm = SVC()

# Define the parameter grid for GridSearchCV
parameters = {
    'kernel': ['linear', 'rbf', 'poly', 'sigmoid'],
    'C': np.logspace(-3, 3, 5),
    'gamma': np.logspace(-3, 3, 5)
}

# Create the GridSearchCV object with 10-fold cross-validation
svm_cv = GridSearchCV(svm, parameters, cv=10)

# Fit the SVM model using the training data (X_train and Y_train)
svm_cv.fit(X_train, Y_train)

# Print the best parameters found by GridSearchCV
print("Tuned hyperparameters (best parameters):", svm_cv.best_params_)
print("Accuracy:", svm_cv.best_score_)
```

```
Tuned hyperparameters (best parameters): {'C': 1.0, 'gamma': 0.03162277660168379,
Accuracy: 0.8482142857142856
```

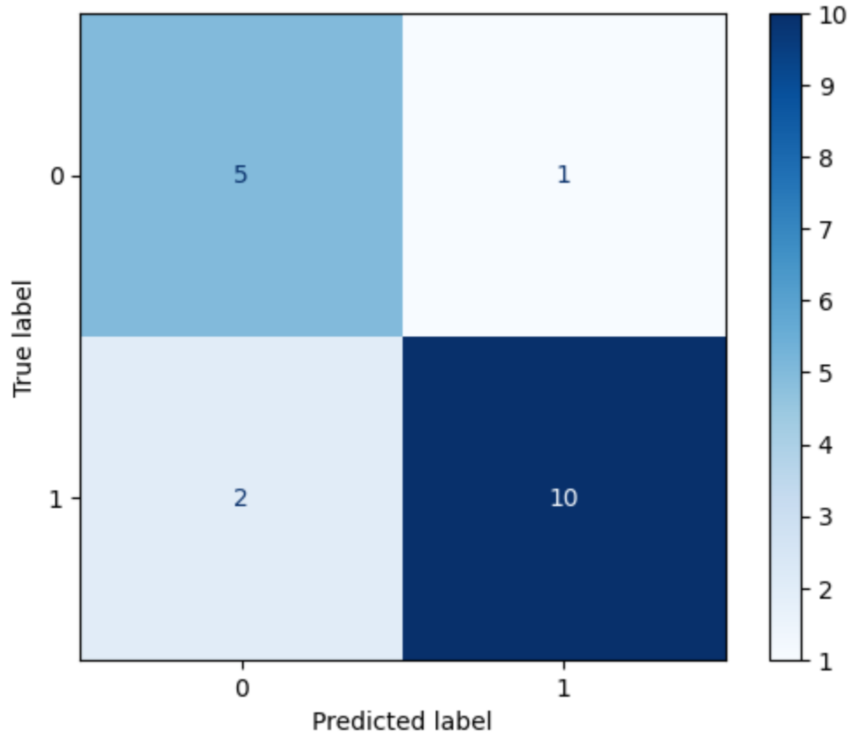
Predictive Analysis (Classification)

```python
test_accuracy = tree_cv.score(X_test, Y_test)
print("Accuracy on test data: ", test_accuracy)

yhat = tree_cv.predict(X_test)
cm = confusion_matrix(Y_test, yhat)

disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=tree_cv.classes_)
disp.plot(cmap='Blues')
plt.show()
```

```
Accuracy on test data:  0.8333333333333334
```



Key Findings

- All models had virtually the same accuracy on the test set at **83.33% accuracy**.
- **SpaceX** has shown a consistent improvement in launch success rates over the years.
- Orbits such as **ES-L1, GEO, HEO, and SSO** exhibit the highest success rates.
- The test size is small at only sample size of 18, which can cause large variance in accuracy results.

Conclusion

- The model, with an **83% accuracy**, can assist SpaceY in making informed decisions on whether a launch should proceed, based on the likelihood of a successful Stage 1 landing.
- **Future Data Collection** is crucial to refine the model and enhance accuracy for better predictions.

By leveraging this model, SpaceY can significantly improve decision-making, reduce costs, and optimize its launch operations!

Aknowledgements

Instructors:

Rav Ahuja, Alex Aklson, Aije Egwaikhide, Svetlana Levitan, Romeo Kienzler, Polong Lin, Joseph Santarcangelo, Azim Hirjani, Hima Vasudevan, Saishruthi Swaminathan, Saeed Aghabozorgi, Yan Luo


Thank you!