IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Jawad Ahmad>
<13/01/2025>

Outline

Introduction

- **Project Background and Context**
- SpaceX offers Falcon 9 rocket launches at a significantly lower cost—$62 million—compared to other providers charging upwards of $165 million. This cost advantage is largely due to SpaceX's ability to reuse the first stage of the rocket. Accurately predicting whether the first stage will land successfully can provide crucial insights into launch costs. Such predictions are valuable for competing companies looking to bid against SpaceX. The primary goal of this project is to build a machine learning pipeline capable of forecasting the likelihood of a successful first-stage landing.
- **Key Problems to Address**

1. **Critical Factors for Successful Landing**
   What are the key factors that influence the success of a rocket's first-stage landing?
2. **Feature Interactions**
   How do different features interact to impact the success rate of first-stage landings?
3. **Operational Requirements**
   What specific operating conditions are necessary to ensure a successful landing program?

Executive Summary

- **Methodology Overview**

1. **Data Collection via API**
   Collection of data through APIs to ensure real-time and structured information retrieval.
2. **Web Scraping for Data Extraction**
   Leveraging web scraping techniques to extract valuable data from online sources.
3. **Data Wrangling and Preprocessing**
   Cleaning and organizing raw data to make it ready for analysis.
4. **Exploratory Data Analysis (EDA) with SQL**
   Using SQL queries for an in-depth exploration and understanding of the dataset.
5. **Exploratory Data Analysis through Data Visualization**
   Visualization techniques to uncover insights and trends from the dataset.
6. **Interactive Visual Analytics with Folium**
   Integration of Folium for interactive maps and geographical data analysis.
7. **Machine Learning Model for Prediction**
   Applying machine learning techniques to develop predictive models.

- **Results Overview**

1. **Exploratory Data Analysis Results**
   Key findings from the data exploration process, presented in both numerical and visual formats.
2. **Screenshots of Interactive Analytics**
   Visual snapshots of interactive analytics to showcase data-driven insights.
3. **Predictive Analytics Results**
   The outcomes of the predictive models and their effectiveness in forecasting.

Section 1

Methodology

Methodology

Executive Summary

- **Data Collection Methodology:**
  Data was gathered through the SpaceX API for structured information and web scraping from Wikipedia for additional data points.
- **Data Wrangling:**
  Data preprocessing techniques, such as one-hot encoding, were applied to categorical features to prepare the dataset for analysis.
- **Exploratory Data Analysis (EDA):**
  EDA was conducted using a combination of data visualization techniques and SQL queries to explore trends, correlations, and patterns within the data.
- **Interactive Visual Analytics:**
  Interactive visualizations were developed using Folium for geographical analysis and Plotly Dash for engaging, dynamic dashboards.
- **Predictive Analysis:**
  Classification models were implemented to predict outcomes based on the dataset. The models were tuned for optimal performance and evaluated for accuracy and robustness.
- **Model Development, Tuning, and Evaluation:**
  The process of building, fine-tuning, and evaluating classification models involved using techniques such as cross-validation and hyperparameter optimization to ensure the highest predictive accuracy.

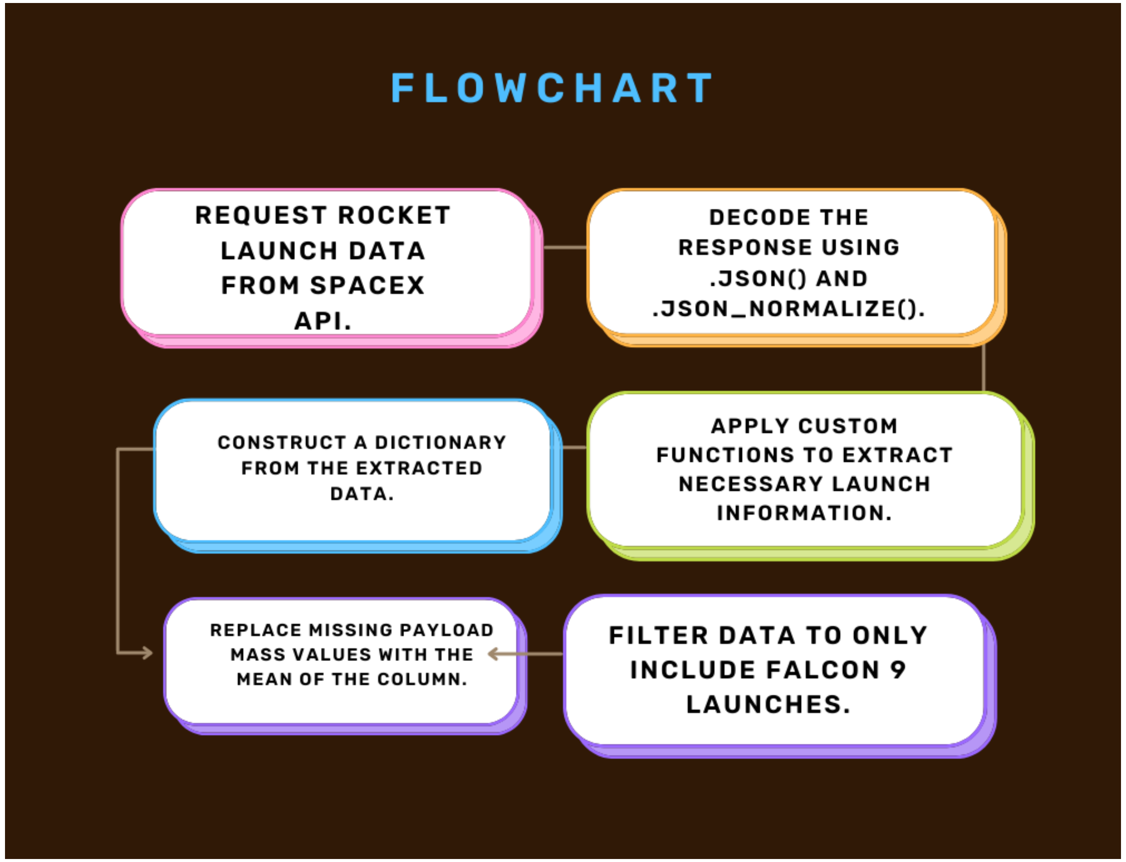
Data Collection

- The data was collected using a combination of techniques:

1. **SpaceX API Collection:**
   We initiated a GET request to the SpaceX API to retrieve relevant data. The response was decoded using the .json() function call and then converted into a pandas DataFrame using .json_normalize() for structured analysis.
2. **Data Cleaning:**
   After obtaining the data, we conducted a thorough cleaning process, checking for missing values and filling in gaps where necessary to ensure the dataset was complete and ready for analysis.
3. **Web Scraping from Wikipedia:**
   We performed web scraping using BeautifulSoup to extract Falcon 9 launch records from Wikipedia. The goal was to retrieve the data in the form of an HTML table, parse the table, and convert it into a pandas DataFrame for further analysis.

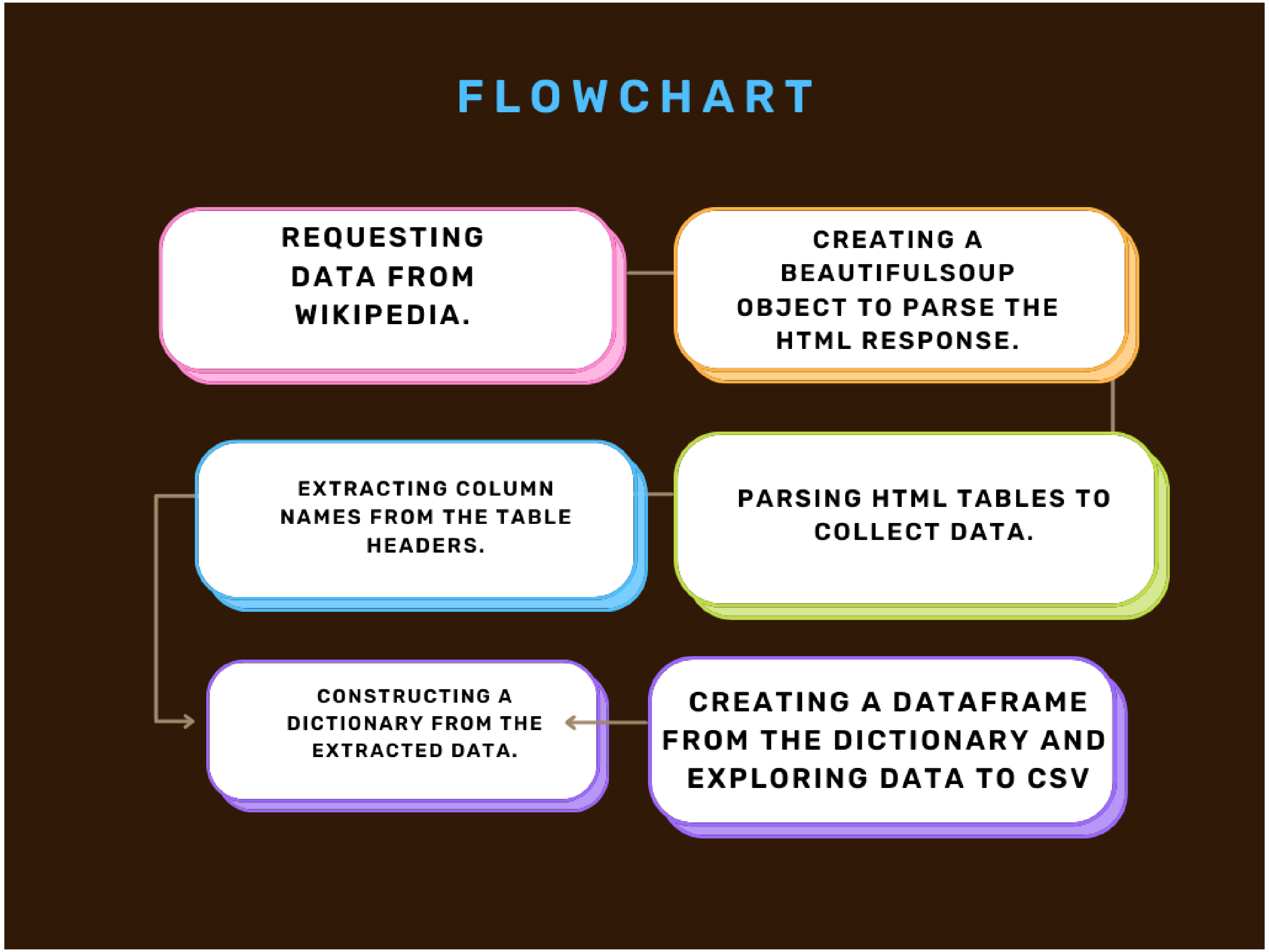
Data Collection – SpaceX API

- Add the GitHub URL: https://github.com/ahmadJawad13/Ibm-Data-Science-Capstone-Project/blob/main/jupyter-labs-spacex-data-collection-api.ipynb



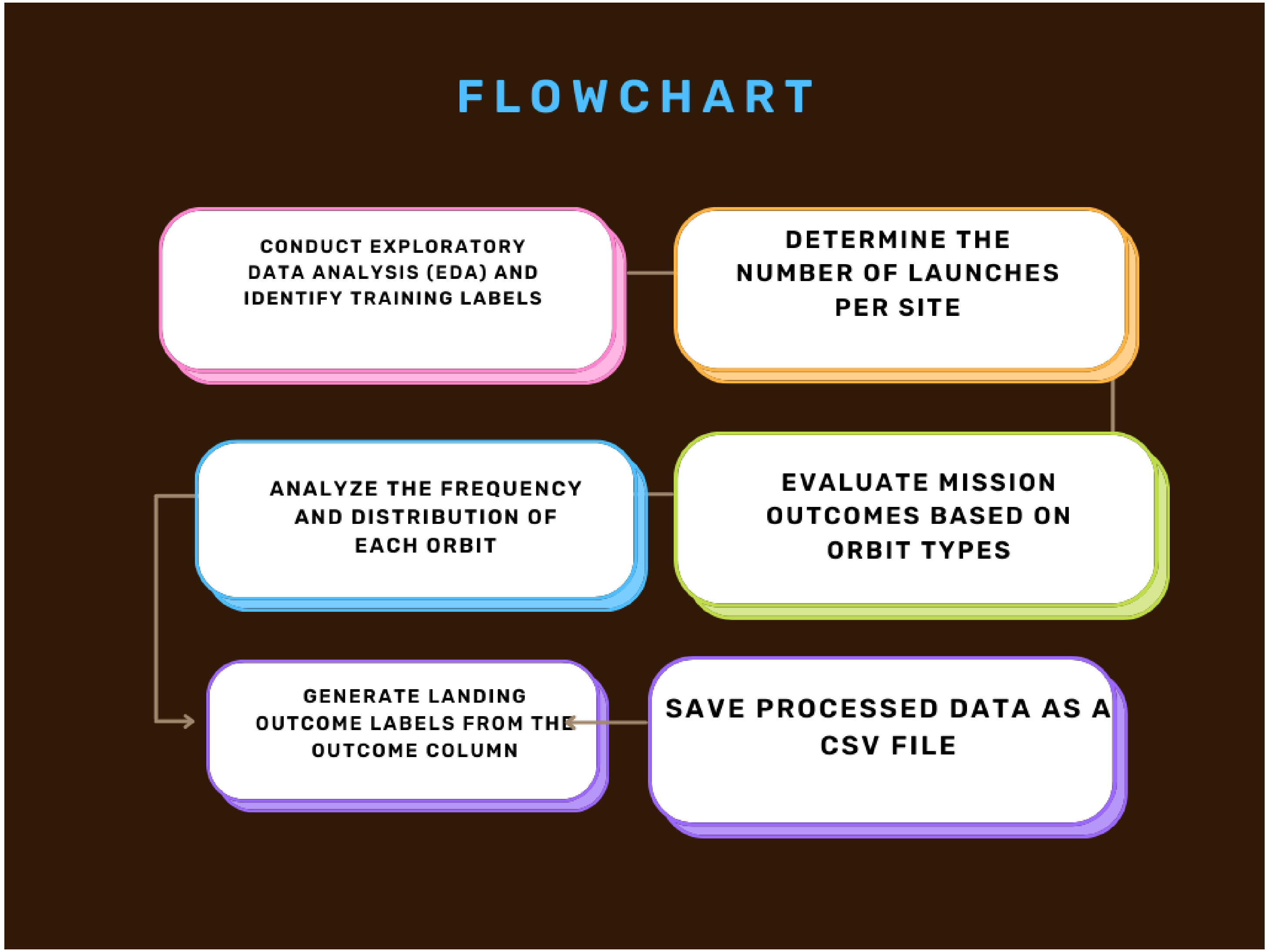
Data Collection - Scraping

- Github url:
- https://github.com/ahmadJawad13/Ibm-Data-Science-Capstone-Project/blob/main/jupyter-labs-webscraping.ipynb



Data Wrangling

- https://github.com/ahmadJawad13/Ibm-Data-Science-Capstone-Project/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb



EDA with Data Visualization

- We analyzed the data by visualizing various relationships to uncover meaningful patterns and trends. This included examining how flight numbers were distributed across different launch sites and exploring the relationship between payload mass and launch sites.
- We also assessed the success rates associated with each orbit type and investigated how flight numbers correlated with specific orbit types. Additionally, we analyzed the yearly trend of launch successes to identify patterns over time and gain insights into the overall performance trajectory. These visualizations provided a clearer understanding of the key factors influencing launch outcomes.
- **GitHub**: https://github.com/ahmadJawad13/Ibm-Data-Science-Capstone-Project/blob/main/eda%20with%20visualization%20vi(1).ipynb

EDA with SQL

- We utilized the SpaceX dataset and integrated it into a PostgreSQL database directly within the Jupyter Notebook environment for streamlined data management and analysis.
- By leveraging SQL for exploratory data analysis, we gained valuable insights from the dataset. Some of the key findings included:

• Listing all distinct launch sites used in SpaceX missions.

• Summing up the payload mass transported by NASA (CRS) booster missions.

• Calculating the average payload mass for the booster version **F9 v1.1**.

• Counting the number of missions that were successful versus those that failed.

• Identifying details of failed landings on drone ships, such as the booster versions involved and the associated launch sites.

- **GitHub** :https://github.com/ahmadJawad13/Ibm-Data-Science-Capstone-Project/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- We visualized all the launch sites on a Folium map and incorporated various map elements, such as markers, circles, and lines, to represent the success or failure of launches at each location.

- The launch outcomes were categorized into two classes: 0 for failure and 1 for success. These outcomes were represented using color-coded marker clusters, enabling us to identify which launch sites demonstrated higher success rates.

- Additionally, we measured the distances between each launch site and its surrounding features. This allowed us to explore questions such as:
  - Whether launch sites are located near railways, highways, and coastlines.
  - Whether launch sites maintain a certain distance from cities.

GitHub:

- https://github.com/ahmadJawad13/Ibm-Data-Science-Capstone-Project/blob/main/lab_jupyter_launch_site_location.ipynb

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)

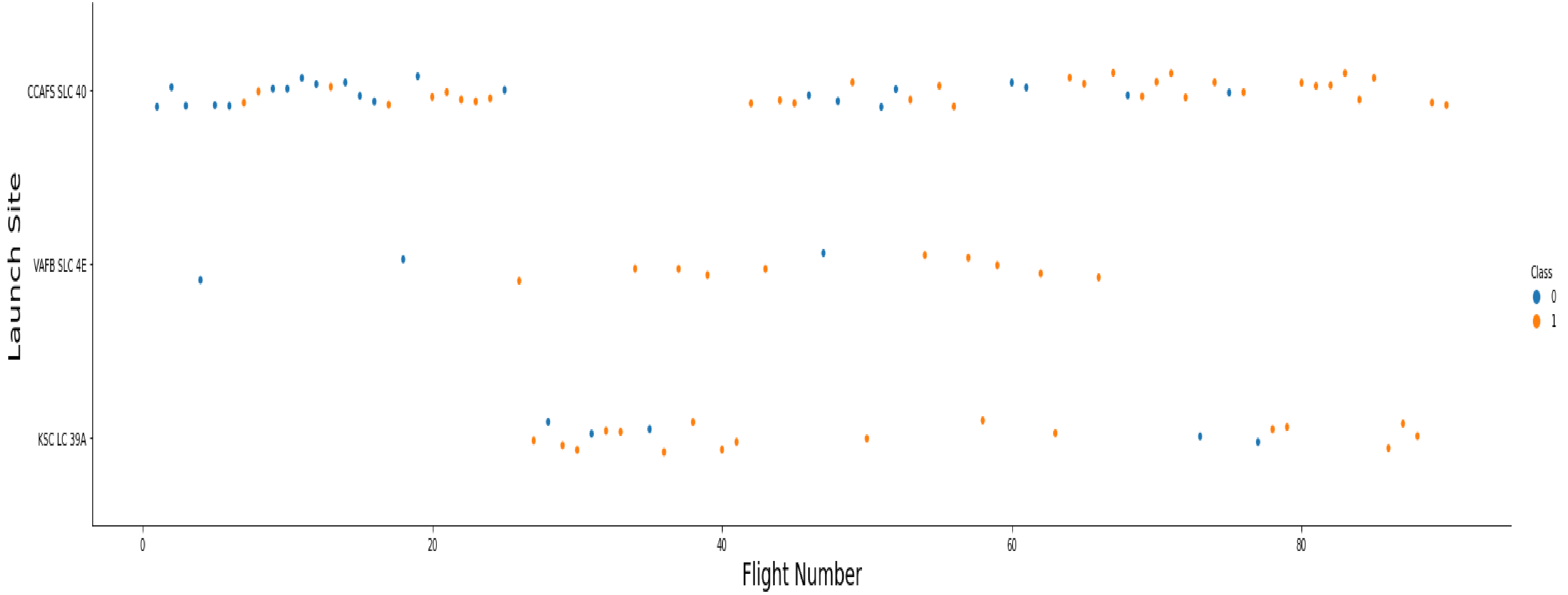
- We began by loading the dataset using NumPy and Pandas for efficient data manipulation. The data was then transformed and preprocessed before splitting it into training and testing subsets.
- Next, we developed multiple machine learning models and fine-tuned their hyperparameters using GridSearchCV to identify optimal configurations.
- Accuracy was chosen as the primary evaluation metric for model performance. To further enhance the results, we applied feature engineering and fine-tuned the algorithms.
- Finally, we identified the best-performing classification model based on its performance metrics.
- **GitHub** :https://github.com/ahmadJawad13/Ibm-Data-Science-Capstone-Project/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5%20(1).ipynb

Results

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

Section 2

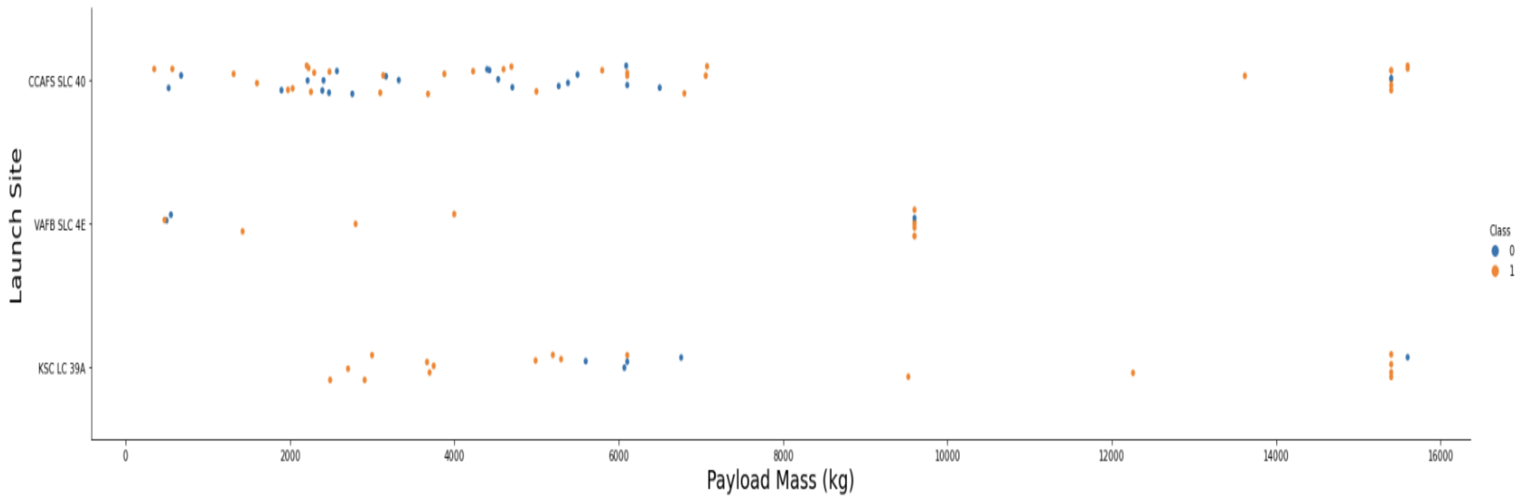
Insights drawn from EDA



Flight Number vs. Launch Site

- From the plot, we observe that as the flight number increases at a launch site, the success rate (Class 1) tends to improve. This suggests that with more launches at a given site, the likelihood of successful outcomes increases over time.

Payload vs. Launch Site

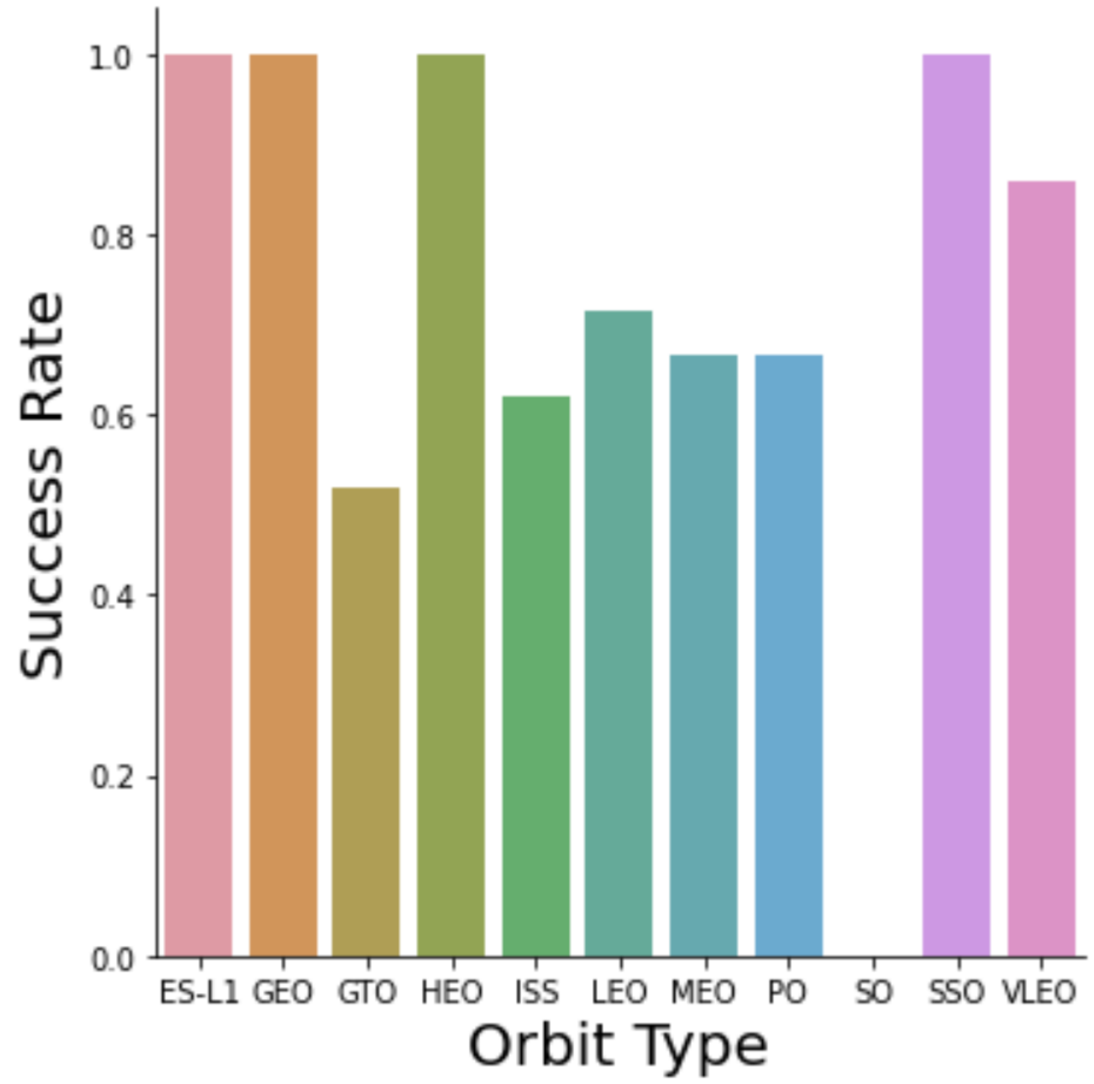


Now try to explain any patterns you found in the Payload Vs. Launch Site scatter point chart.

Explanation:

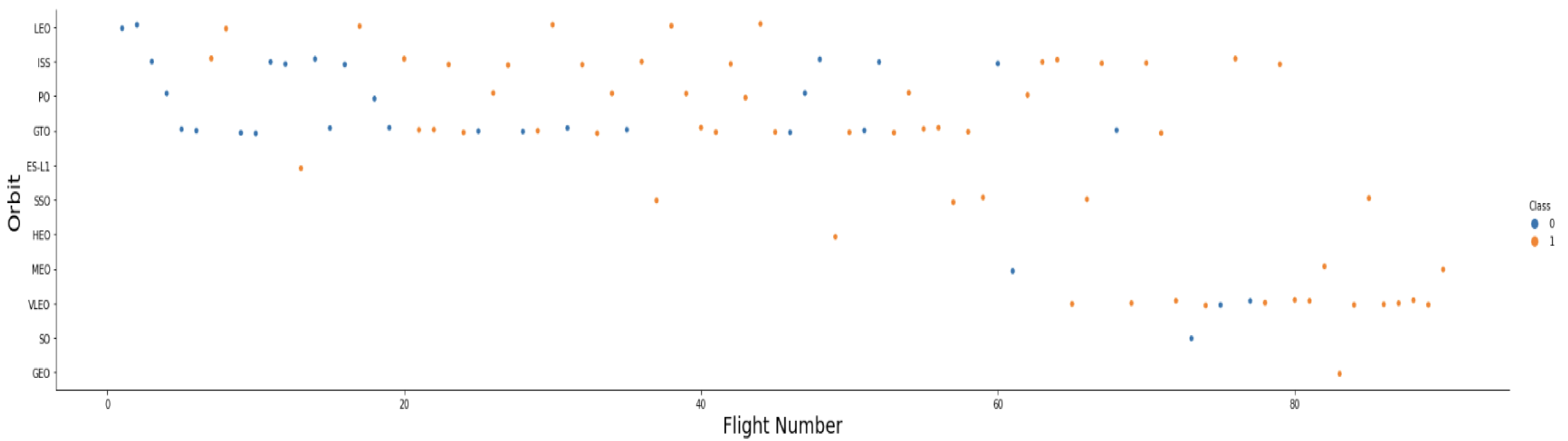
- For every launch site the higher the payload mass, the higher the success rate.
- Most of the launches with payload mass over 7000 kg were successfull.
- KSC LC 39A has a 100% success rate for payload mass under 5500 kg too.

Success Rate vs. Orbit Type



Explanation:

- **100% success rate orbits**: ES-L1, GEO, HEO, SSO
- **0% success rate orbit**: SO
- **Success rates between 50% and 85%**: GTO, ISS, LEO, MEO, PO

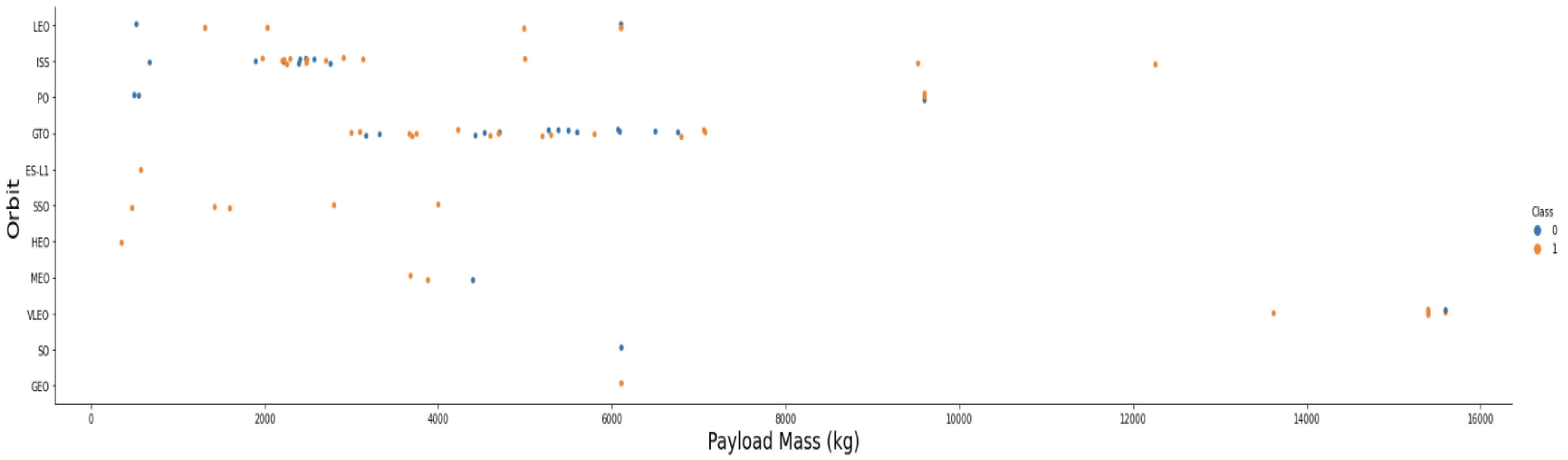


Flight Number vs. Orbit Type

- The plot below illustrates the relationship between Flight Number and Orbit type. It reveals that for LEO orbit, success rates tend to improve with an increasing number of flights. However, for GTO orbit, no clear correlation between flight number and success is observed.

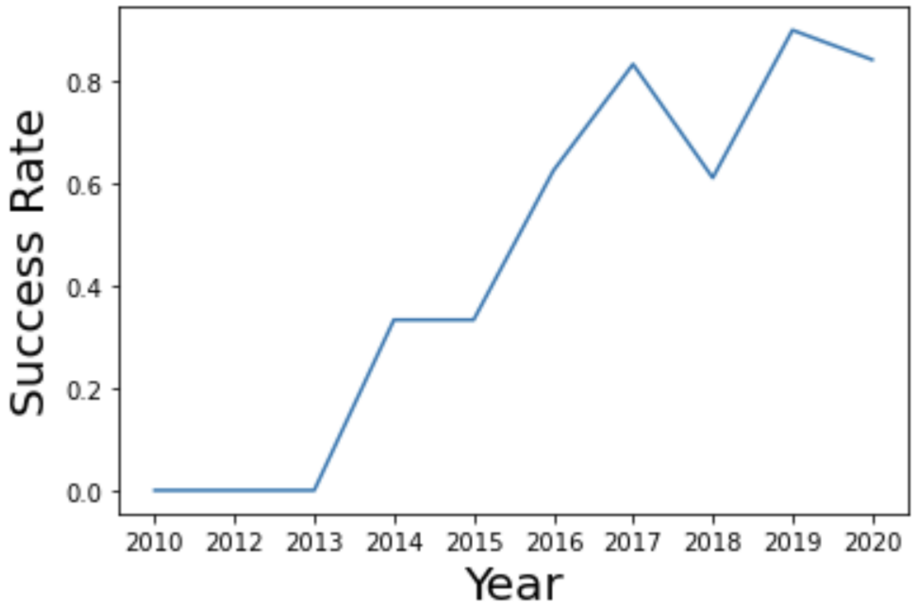
Payload vs. Orbit Type

- It can be observed that for heavy payloads, successful landings are more frequent in PO, LEO, and ISS orbits.



Launch Success Yearly Trend

- The plot shows that the success rate steadily increased from 2013 to 2020.



All Launch Site Names

- We used the keyword **DISTINCT** in the SQL query to retrieve only the unique launch site names from the SpaceX dataset. This helps eliminate duplicate entries and provides a clear list of all launch sites used.

Task 1

Display the names of the unique launch sites in the space mission

```
%sql select distinct launch_site from SPACEXDATASET;
```

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

Display 5 records where launch sites begin with the string '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

Launch Site Names Begin with 'CCA'

- This query uses the LIKE operator to find rows where the Launch_Site column begins with “CCA” and limits the output to the first 5 matching records.

Task 3

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

```
%sql SELECT SUM("PAYLOAD_MASS__KG_") AS total_payload_mass FROM SPACEXTABLE WHERE "Customer" = 'NASA (CRS)';
```

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

total_payload_mass
45596

Total Payload Mass

The total payload mass carried by the boosters from NASA is 45569

Task 4

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

```
4]: %sql SELECT AVG(PAYLOAD_MASS__KG_) \
        FROM SPACEXTBL \
        WHERE BOOSTER_VERSION = 'F9 v1.1';
```

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

4]:

AVG(PAYLOAD_MASS__KG_)
2928.4

Average Payload Mass by F9 v1.1

- The average payload mass transported by the booster version **F9 v1.1** was determined to be **2928.4.**
- The query calculates the average payload mass for launches where the booster version contains **“F9 v1.1”**. It uses the AVG function to find the mean value of the "PAYLOAD_MASS__KG_" column, filtering rows with a LIKE clause for booster versions matching %F9 v1.1%.

First Successful Ground Landing Date

- The first successful landing on a ground pad occurred on **22nd December 2015**. The query finds this date using MIN("Date"), filtering for rows where "Landing_Outcome" equals 'Success (ground pad)'.

Task 5

List the date when the first succesful landing outcome in ground pad was acheived.

Hint:Use min function

```
]: %sql SELECT MIN("Date") AS first_successful_landing_date FROM SPACEXTABLE WHERE "Landing_Outcome" = 'Success (ground pad)';
```

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

]:

first_successful_landing_date
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- The query filters for boosters that successfully landed on a drone ship ("Landing_Outcome" = 'Success (drone ship)') and carried a payload mass between 4000 and 6000 kilograms (BETWEEN 4000 AND 6000). The DISTINCT clause ensures that only unique booster versions are returned.

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 ¶

```
[15]: %sql SELECT DISTINCT "Booster_Version" FROM SPACEXTABLE WHERE "Landing_Outcome" = 'Success (drone ship)' AND "PAYLOAD_MASS__KG_" BETWEEN 4000 AND 6000;
```

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

[15]:

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

Total Number of Successful and Failure Mission Outcomes

- This query counts the number of occurrences for each unique mission outcome in the "Mission_Outcome" column of the SPACEXTABLE. The COUNT(*) function counts the rows, and the GROUP BY "Mission_Outcome" groups the results by each unique mission outcome value.

Task 7

List the total number of successful and failure mission outcomes

```
%sql SELECT "Mission_Outcome", COUNT(*) AS count FROM SPACEXTABLE GROUP BY "Mission_Outcome";
```

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

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

Task 8

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

```
%sql SELECT "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

Boosters Carried Maximum Payload

- This query finds the **Booster_Version** associated with the maximum payload mass:
- 1. **Subquery**: SELECT MAX("PAYLOAD_MASS__KG_") FROM SPACEXTABLE identifies the highest payload mass value in the table.
- 2. **Main Query**: Retrieves the Booster_Version where the "PAYLOAD_MASS__KG_" matches this maximum value.

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.

```
%%sql
SELECT substr(Date, 1, 4) AS Year, substr(Date, 6, 2) AS Month, Landing_Outcome, Booster_Version, Launch_Site
FROM SPACEXTABLE
WHERE substr(Date, 1, 4) = '2015' AND Landing_Outcome = 'Failure (drone ship)';
```

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

Year	Month	Landing_Outcome	Booster_Version	Launch_Site
2015	01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
2015	04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

2015 Launch Records

- The query fetches data for failed drone ship landings in 2015. It extracts the year and month from the Date column and filters for records where Landing_Outcome is 'Failure (drone ship)'.

Task 10

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

```
%%sql
SELECT Landing_Outcome, COUNT(*) AS count
FROM SPACEXTABLE
WHERE Date BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY Landing_Outcome
ORDER BY count DESC;
```

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

Landing_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

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

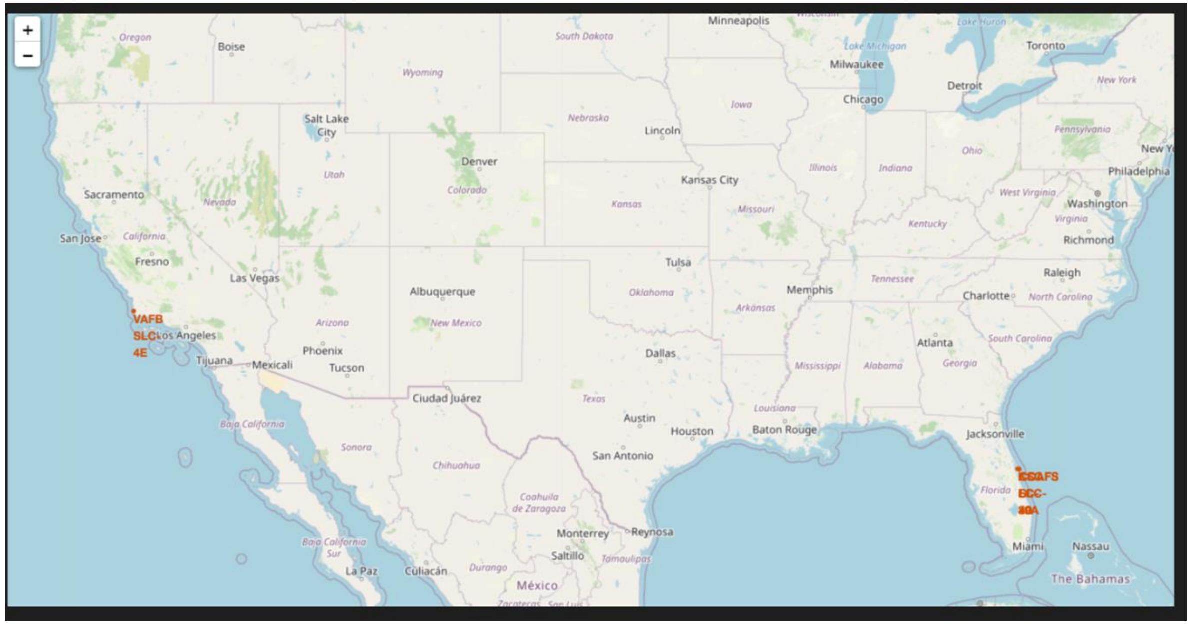
- This query counts the occurrences of each landing outcome between **June 4, 2010**, and **March 20, 2017**:
- 1. **Filter**: WHERE Date BETWEEN '2010-06-04' AND '2017-03-20' selects rows within the specified date range.
- 2. **Group and Count**: GROUP BY Landing_Outcome groups results by each unique landing outcome, and COUNT(*) tallies the number of occurrences for each.
- 3. **Order**: ORDER BY count DESC sorts the results in descending order of counts.

Section 3

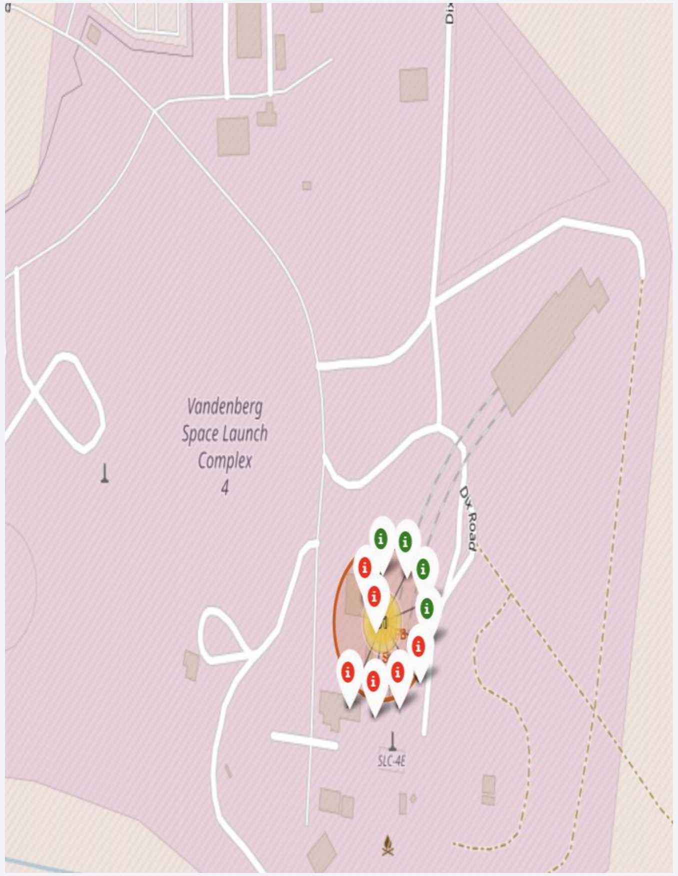
Launch Sites Proximities Analysis

<Global sites for all lanuches>

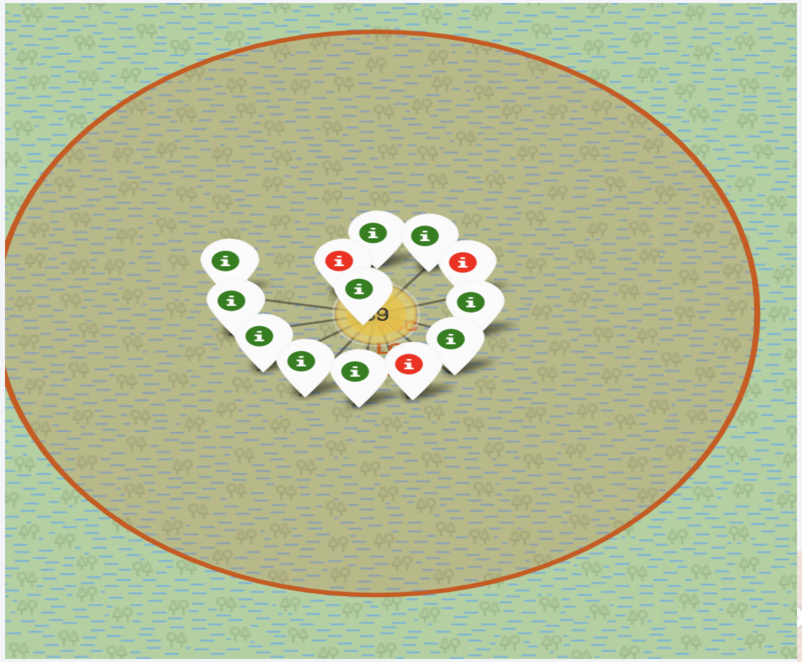
- We can see that all launches sites from SpaceX are on the edge of **America** coastal area like **Florida** and **California**

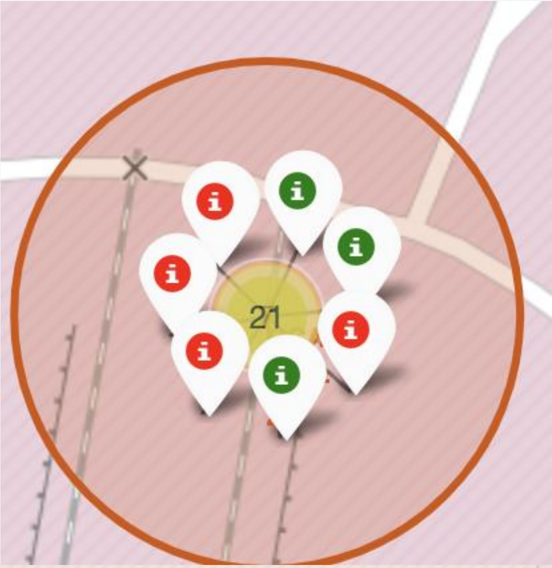


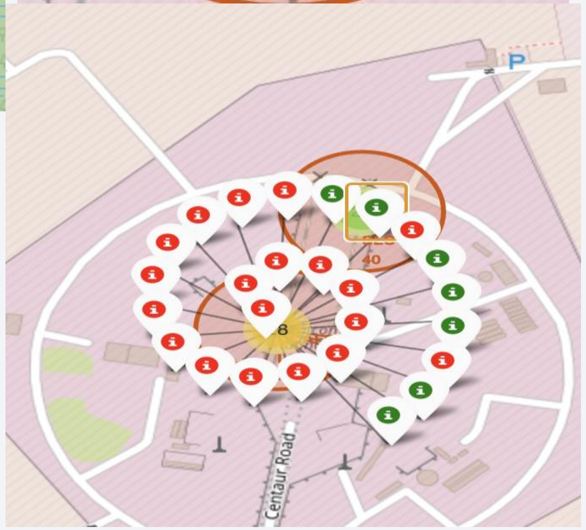
Markers showing launch sites with color labels



California launch

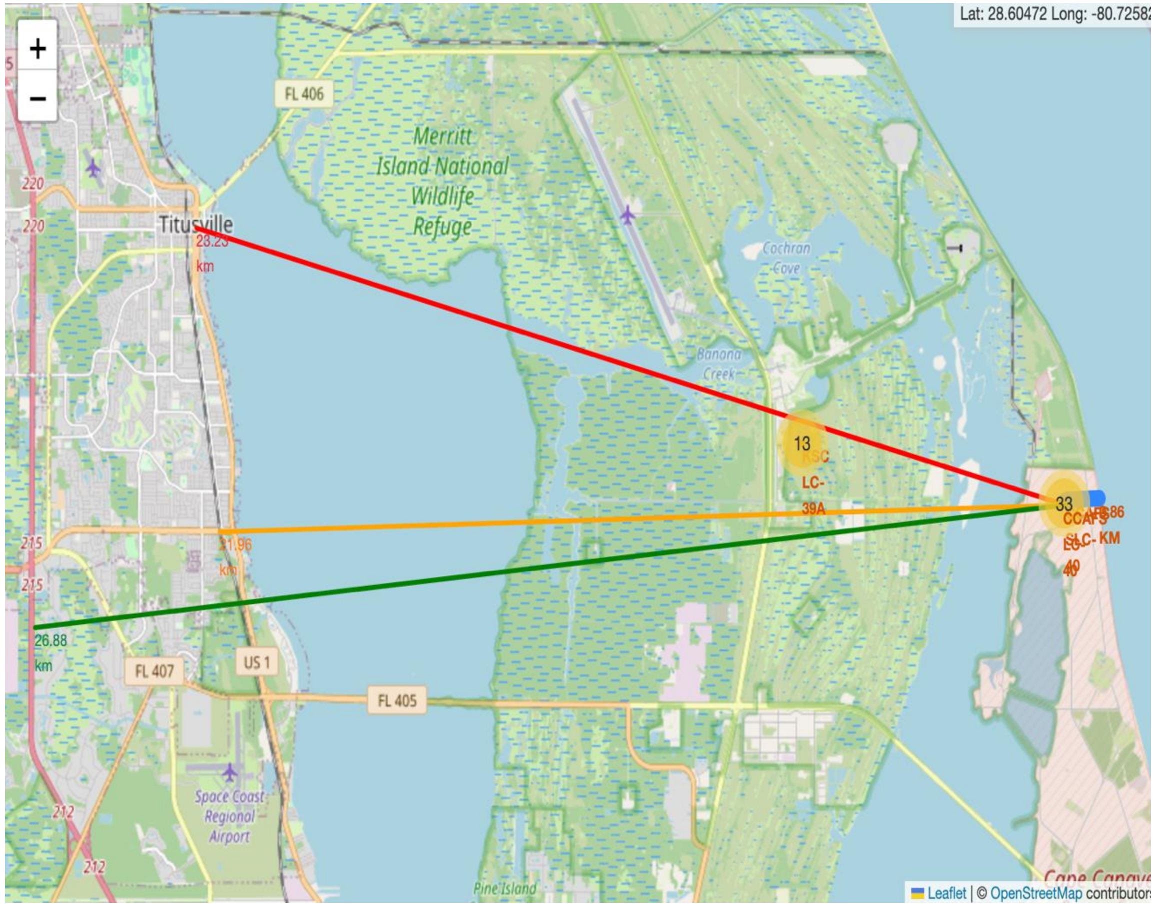

Florida launches





<Analysis of SpaceX Landing Spot: KSC LC-39A>

- **Proximity to Key Infrastructure:**
- **Railway**: Relatively close to a railway (~26.88 km), which could be affected in case of debris impact.
- **Highway**: Situated near a major highway (~21.96 km), posing potential risks for transportation and logistics.
- **Coastline**: The site is in close proximity to the coastline, facilitating marine recovery operations but also increasing risks in case of failures.
- **Closest City**: Titusville (~23.23 km away) is the nearest urban area and could be affected in scenarios involving high-speed debris.
- **Potential Hazards:**
- **Rocket Failures**: A failed rocket or booster at high speed could cover distances of 15–25 km in seconds, posing risks to:
    - **Populated Areas**: Titusville and nearby communities.
    - **Critical Infrastructure**: Highways, railways, and coastal operations.
- **Environmental Risks**: Proximity to the Merritt Island National Wildlife Refuge raises concerns about potential environmental damage in the event of debris or fuel leaks.



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

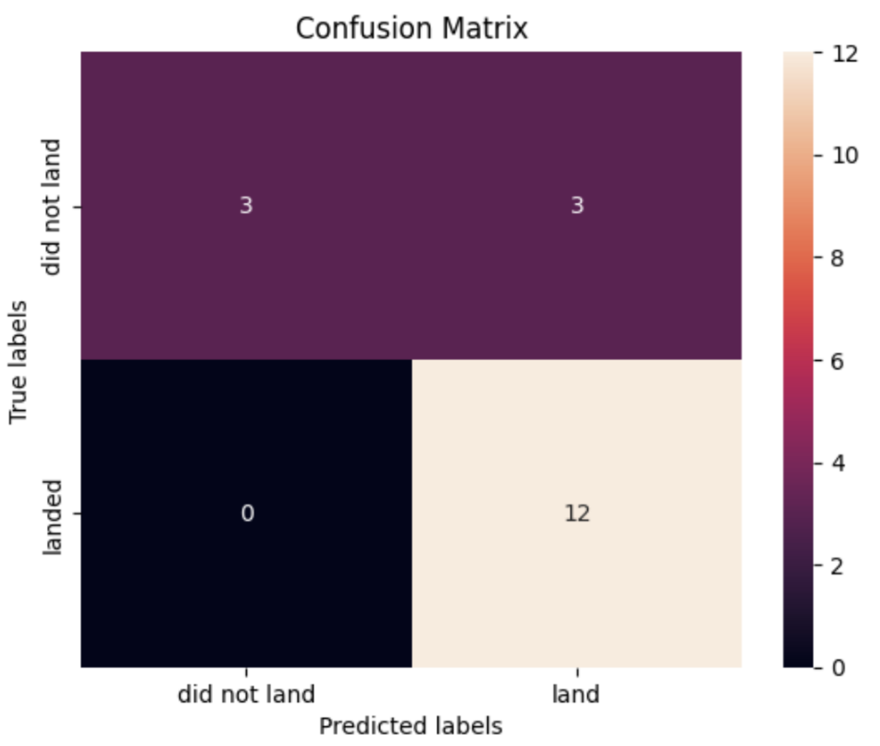
- The **decision tree classifier** achieved the **highest classification accuracy** among all models evaluated, making it the most effective for the given dataset and problem. Its ability to handle complex decision boundaries and interpretability likely contributed to its superior performance.

```python
# Create a dictionary to store the best scores of different models
model_scores = {
    'KNeighbors': knn_cv.best_score_,
    'DecisionTree': tree_cv.best_score_,
    'LogisticRegression': logreg_cv.best_score_,
    'SupportVector': svm_cv.best_score_
}

# Find the model with the highest score
best_model = max(model_scores, key=model_scores.get)

# Print the best model and its score
print(f"The best model is {best_model} with a score of {model_scores[best_model]}")

# Check which model has the best score and print its hyperparameters
if best_model == 'DecisionTree':
    print('Best parameters are:', tree_cv.best_params_)
elif best_model == 'KNeighbors':
    print('Best parameters are:', knn_cv.best_params_)
elif best_model == 'LogisticRegression':
    print('Best parameters are:', logreg_cv.best_params_)
elif best_model == 'SupportVector':
    print('Best parameters are:', svm_cv.best_params_)
```

```
The best model is DecisionTree with a score of 0.8875
Best parameters are: {'criterion': 'entropy', 'max_depth': 8, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_
samples_split': 2, 'splitter': 'random'}
```

Confusion Matrix

- The confusion matrix for the decision tree classifier indicates it can effectively differentiate between the classes. However, the main issue lies in **false positives**, where **unsuccessful landings are incorrectly predicted as successful landings**. This misclassification could impact decision-making processes, particularly in scenarios where accurate identification of unsuccessful landings is critical.



Conclusions

- **Conclusion Summary:**
- 1. **Flight Volume and Success Rate**: Launch sites with a higher number of flights tend to have greater success rates.
- 2. **Improved Success Over Time**: The launch success rate showed consistent growth from **2013 to 2020**.
- 3. **Orbit Success Rates**: Orbits such as **ES-L1, GEO, HEO, SSO, and VLEO** demonstrated the highest success rates.
- 4. **Top Performing Launch Site**: **KSC LC-39A** had the highest number of successful launches among all sites.
- 5. **Best Model**: The **Decision Tree Classifier** proved to be the most effective machine learning model for this classification task.

Thank you!