IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Tomasz Mulka
05.06.2025

Outline

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

Executive Summary

Methodologies Used

•**Data Collection & Cleaning**: Acquired launch records from SpaceX and NASA APIs, cleaned and preprocessed the data.

•**EDA & Visualization**:

- Used **pandas** and **seaborn** for correlation analysis.
- Plotted launch sites and success rates on interactive **Folium** maps.

•**Feature Engineering**:

- Encoded categorical features like launch site, booster version.
- Normalized numerical values like payload mass.

•**Modeling**:

- Implemented and compared multiple classification models:
  - Logistic Regression
  - Support Vector Machine (SVM)
  - Decision Tree
  - K-Nearest Neighbors (KNN)

•**Model Evaluation**:

- Used metrics like accuracy
- Tuned hyperparameters using GridSearchCV.

Introduction

🌍 Project Context

SpaceX offers Falcon 9 launches at a competitive price of $62 million per mission. A key factor in this cost-efficiency is the reusability of the first stage booster. In contrast, other providers can charge upwards of $165 million per launch. Predicting whether the first stage will land successfully is critical for estimating mission costs and enhancing commercial viability.

❓ Problems & Research Questions

Will the Falcon 9 first stage land successfully?

→ Predict landing outcome based on historical launch data.

What factors influence the success of a rocket landing?

→ Explore features such as launch site, payload mass, booster version, and orbit type.

Can we support bidding decisions for alternate providers?

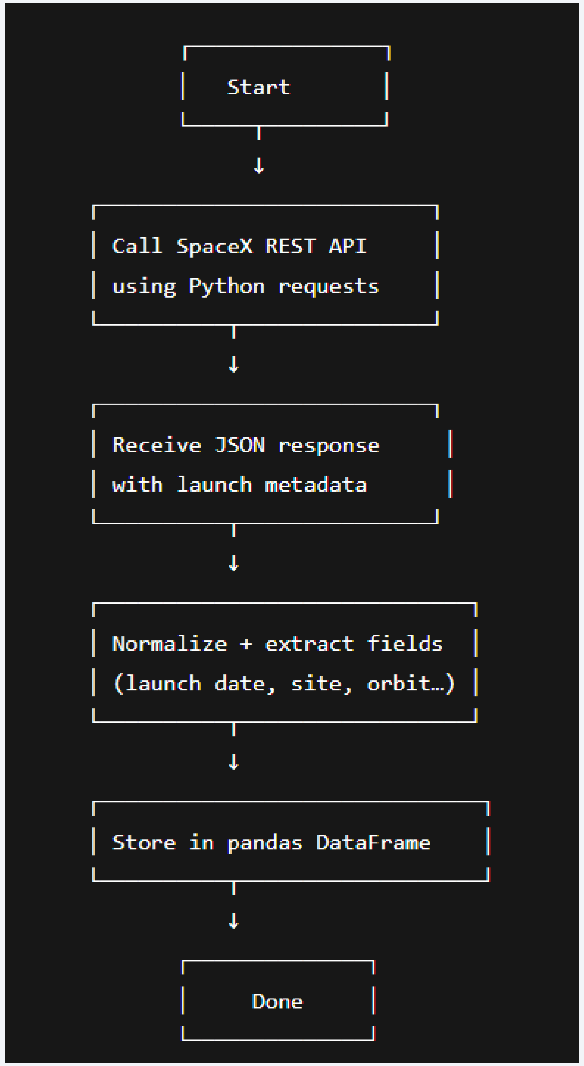
→ Provide actionable insights for organizations competing with SpaceX or planning launches.

Section 1

Methodology

Data Collection – SpaceX API

Main source for launch metadata:
booster version, payload mass, orbit.

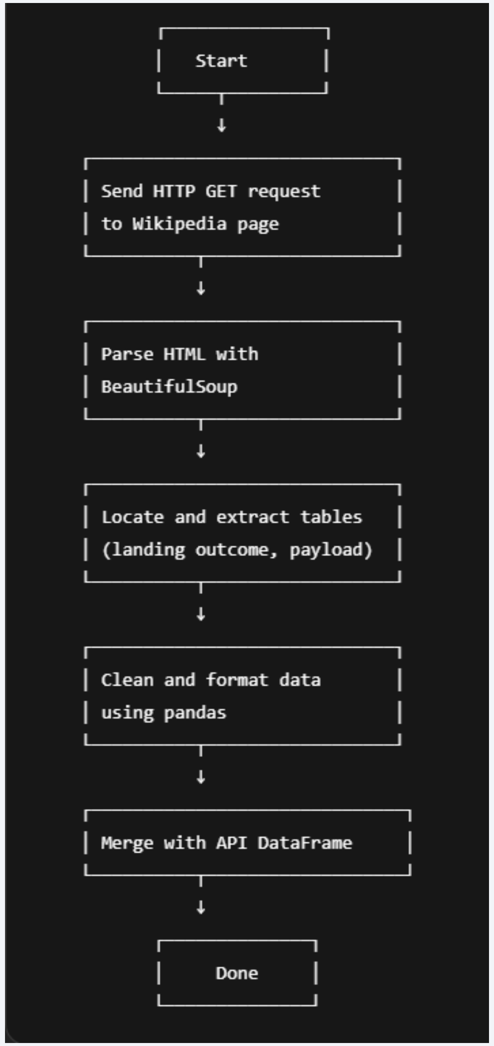


Link to github with corresponding jupyter notebook:

https://github.com/Tomulll/Space-X---Data-Science-Project/blob/main/Data_Collection_API.ipynb

Data Collection - Scraping

Supplementary data: landing outcomes, detailed payload info.



Link to github with corresponding jupyter notebook:

https://github.com/Tomulll/Space-X---Data-Science-Project/blob/main/Webscraping.ipynb

Data Wrangling

Key Processing Steps

Data Cleaning:

- Removed null/irrelevant entries (e.g. launches with no payload or outcome).
- Replaced None/NaN with proper defaults or removed when necessary.

Data Type Conversion:

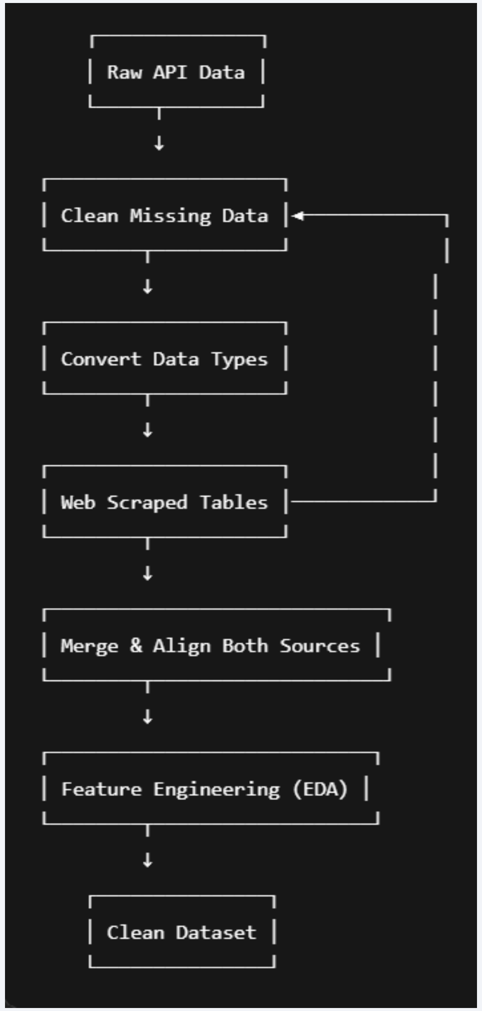
- Converted timestamp strings to datetime format.
- Ensured consistent numeric types for payload mass and flight numbers.

Feature Engineering:

- Extracted binary target variable: LandingSuccess (1 = landed, 0 = failed).
- Created new categorical variables like LaunchSiteCategory.

Merging Sources:

- Merged SpaceX API data and scraped data using FlightNumber and Date.
- Aligned schema and resolved format mismatches.



EDA with Data Visualization

Purpose of Exploratory Data Analysis (EDA)

To explore underlying patterns, detect relationships between features, and gain actionable insights to support model selection and feature engineering for predicting Falcon 9 first stage landing success.

Link to github with corresponding jupyter notebook:

https://github.com/Tomulll/Space-X---Data-Science-Project/blob/main/Visualizations.ipynb

EDA with SQL

Insights Discovered via SQL

•Certain orbits (like LEO and ISS) had **significantly higher success rates**.

•Some launch sites showed **greater consistency in successful landings**.

•Launch activity and success rate **steadily improved each year**, confirming learning effects and technological maturity.

Link to github with corresponding jupyter notebook:

https://github.com/Tomulll/Space-X---Data-Science-Project/blob/main/SQL_Lite.ipynb

Build an Interactive Map with Folium

Map Components Created

•**Markers**

- Placed at each **SpaceX launch site** with custom popups showing site name and coordinates.
- Helped identify launch location distribution across the U.S.

•**Circle Markers**

- Added to highlight proximity to nearby cities and features.
- Radius scaled based on payload mass or distance for visual clarity.

•**Lines (Polylines)**

- Connected launch sites to nearest major cities.
- Visualized potential logistical considerations (e.g., transport or supply paths).

Link to github with corresponding jupyter notebook:

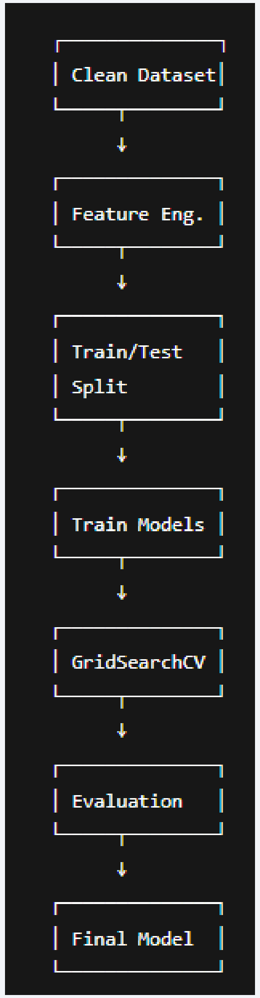
https://github.com/Tomulll/Space-X---Data-Science-Project/blob/main/Launch_Site_Location_Folium.ipynb

Predictive Analysis (Classification)

To predict the landing success of Falcon 9's first stage, I trained multiple classification models including Logistic Regression, Decision Tree, SVM, and K-Nearest Neighbors. The dataset was preprocessed with feature selection and one-hot encoding and then split into training and test sets. I performed hyperparameter tuning using GridSearchCV to optimize model performance. Among all models, the **Decision Tree Classifier** achieved the highest accuracy and interpretability, making it the final choice for deployment.

Link to github with corresponding jupyter notebook:

https://github.com/Tomulll/Space-X---Data-Science-Project/blob/main/Machine_Learning_Predictions.ipynb



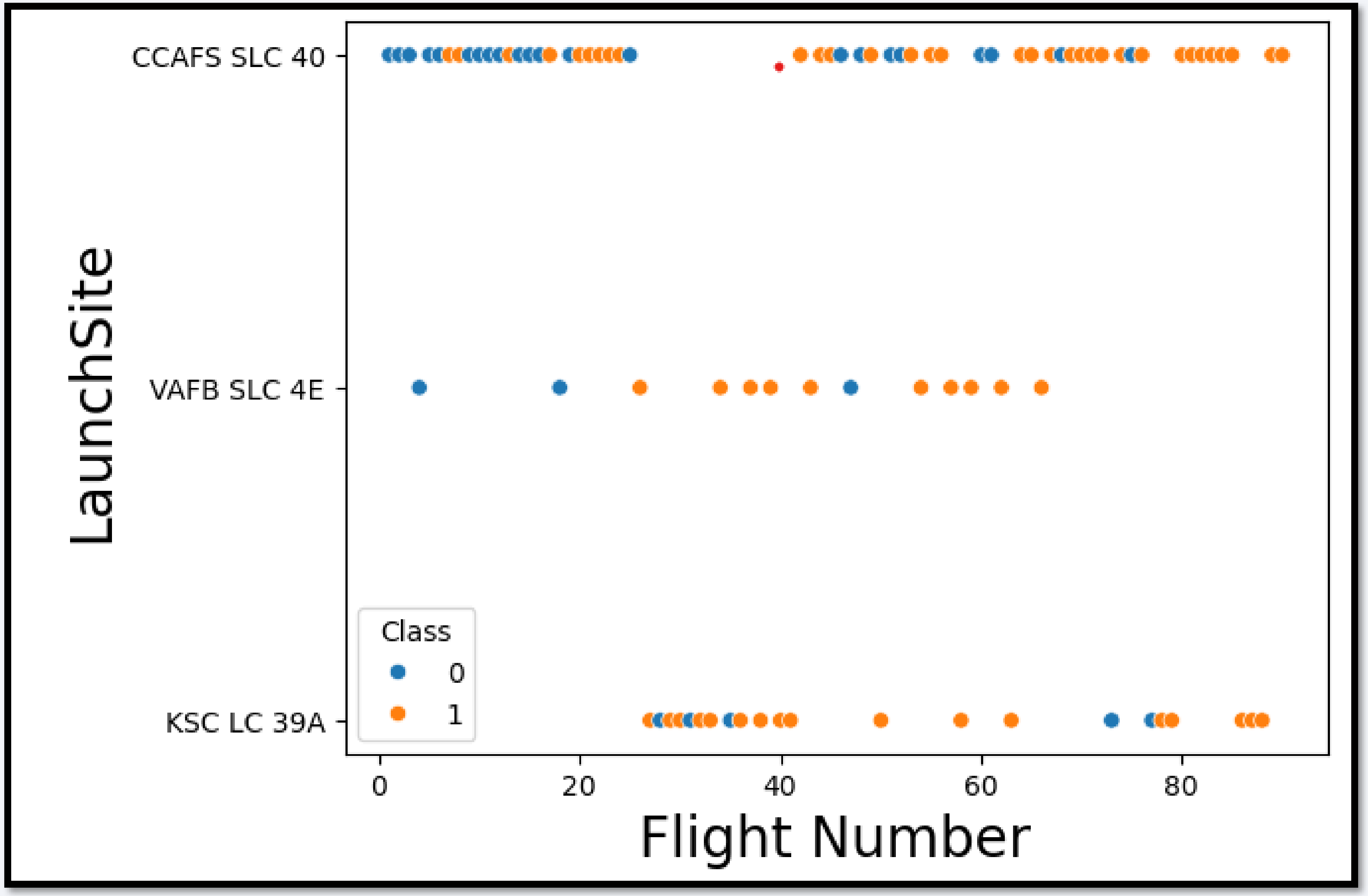
Results

Exploratory data analysis revealed that missions to Low Earth Orbit and moderate payload masses had the highest landing success rates. The success rate has also steadily improved over the years, reflecting SpaceX's growing reliability. An interactive Folium map was created to visualize launch sites and their proximity to key cities, enhancing geographic context. Among the classification models tested, the Decision Tree Classifier achieved the best performance with approximately 85–90% accuracy. Key predictors identified by the model included payload mass, orbit type, and launch site.

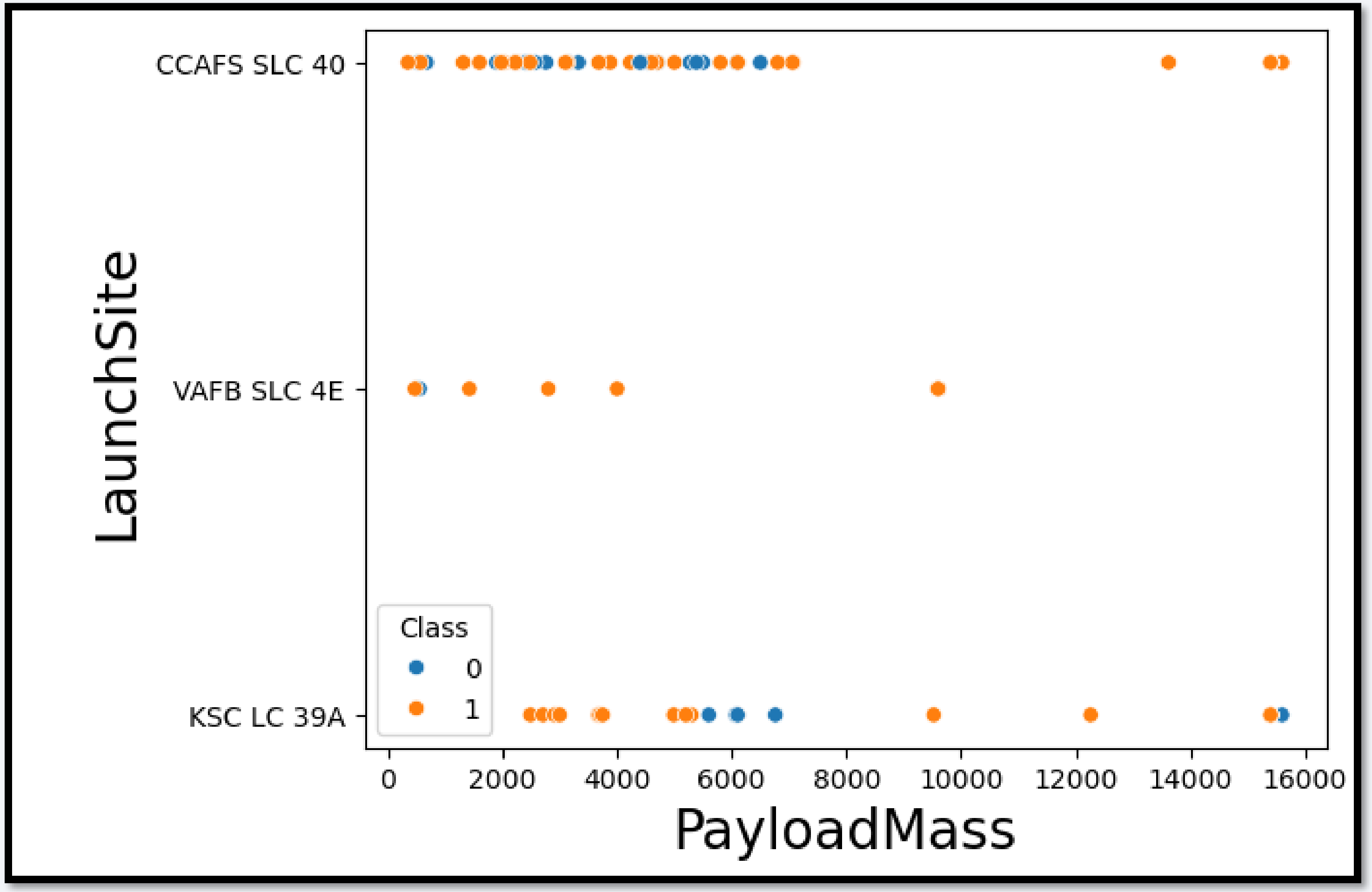
Section 2

Insights drawn from EDA

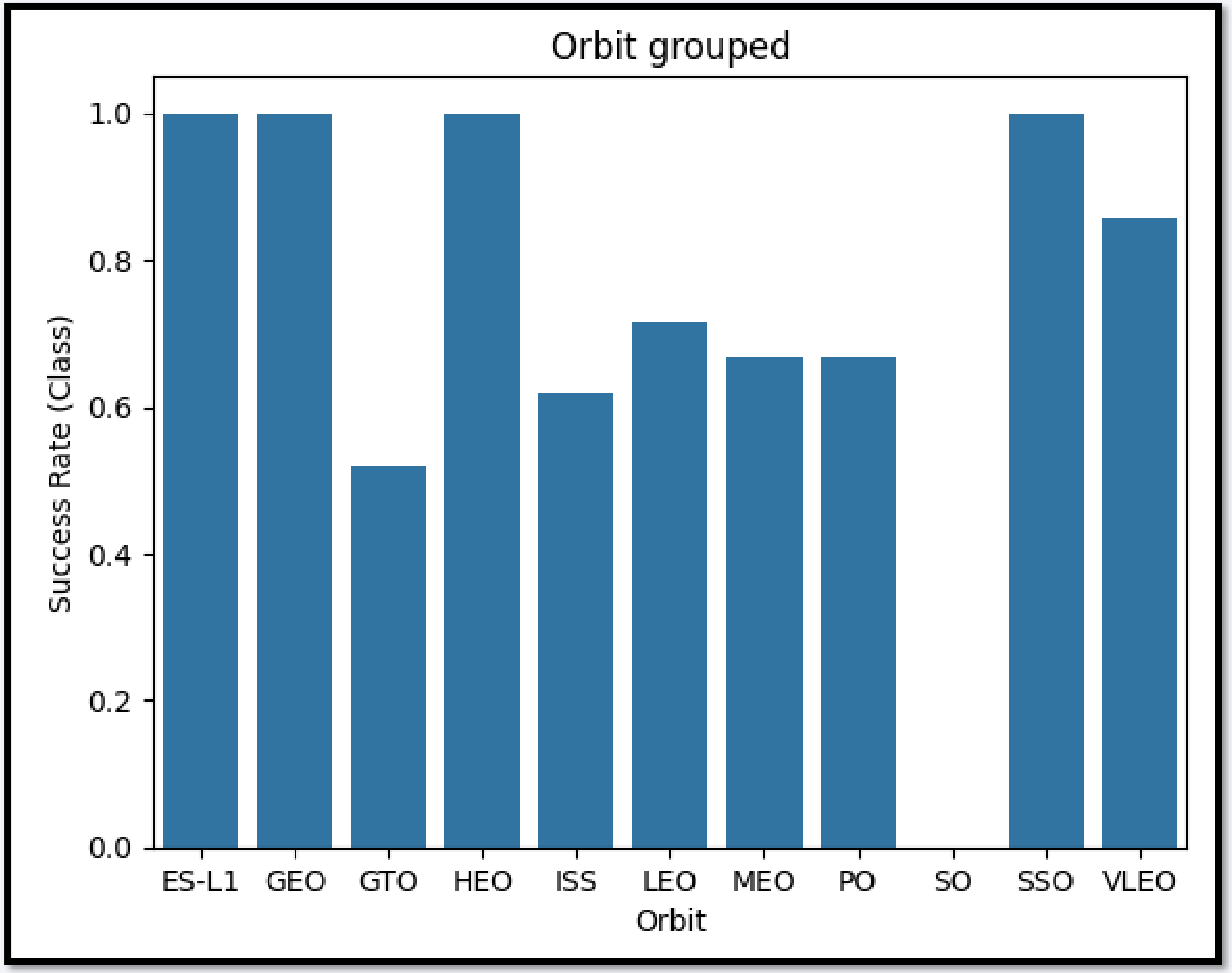
Flight Number vs. Launch Site



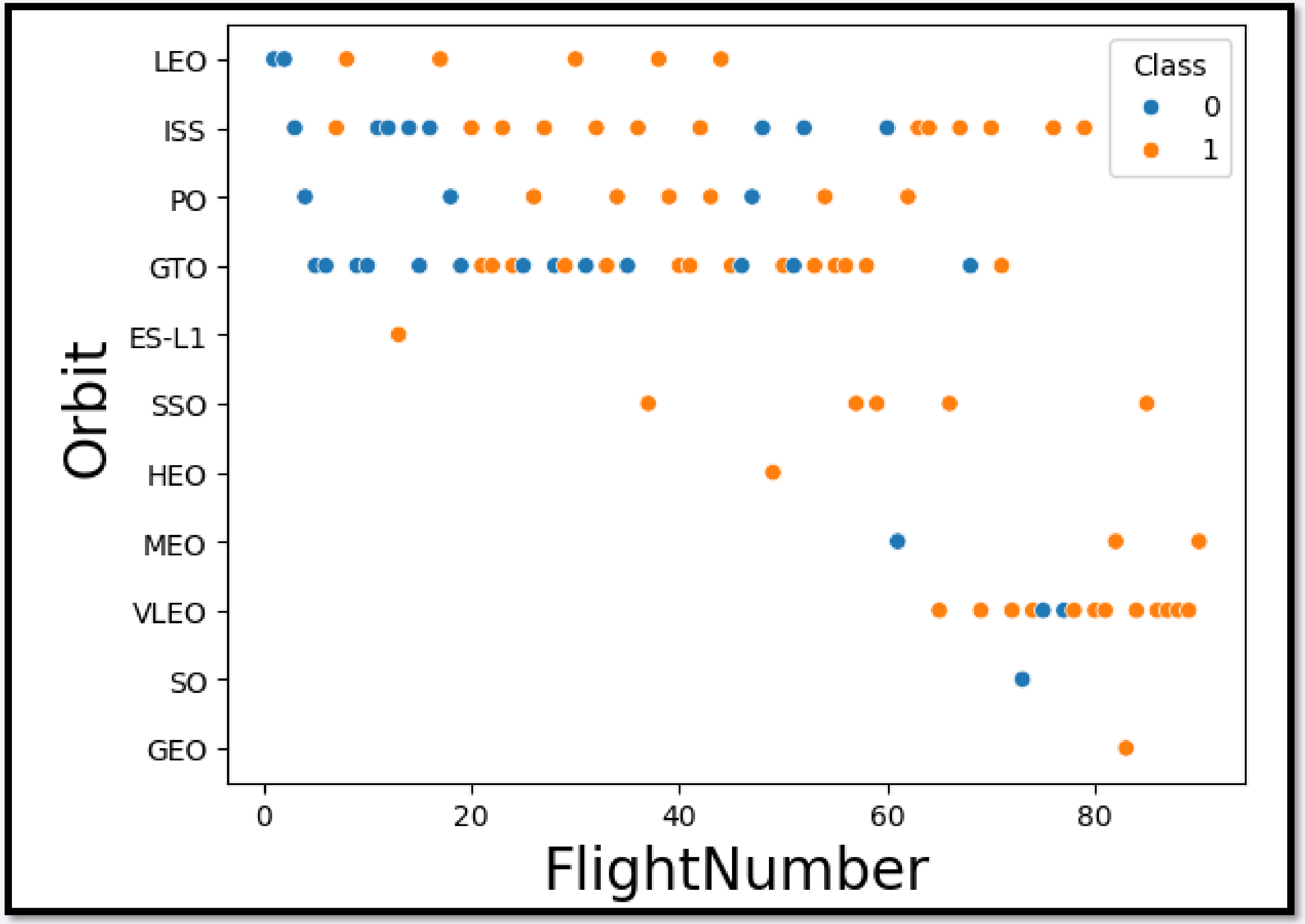
Payload vs. Launch Site



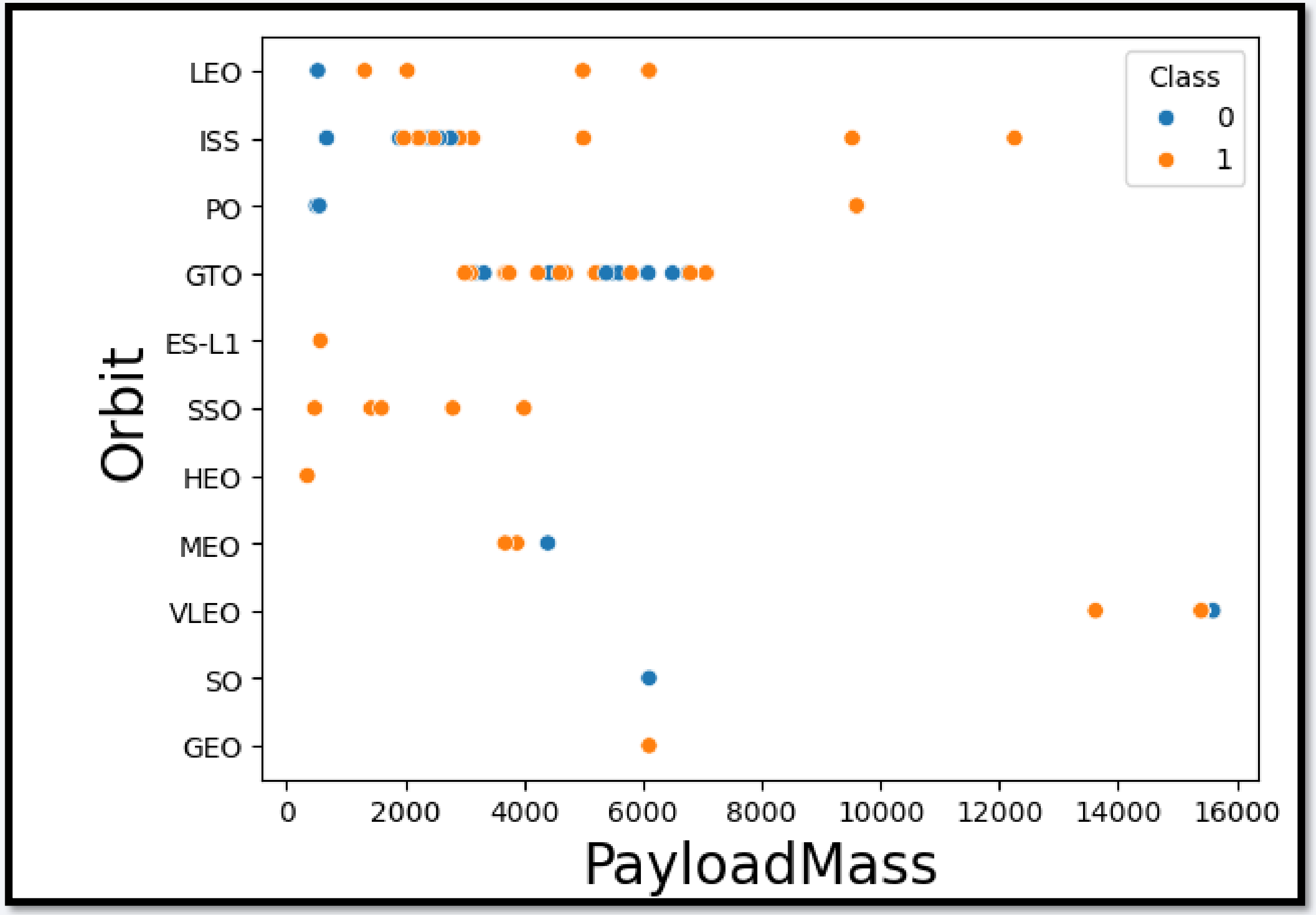
Success Rate vs. Orbit Type



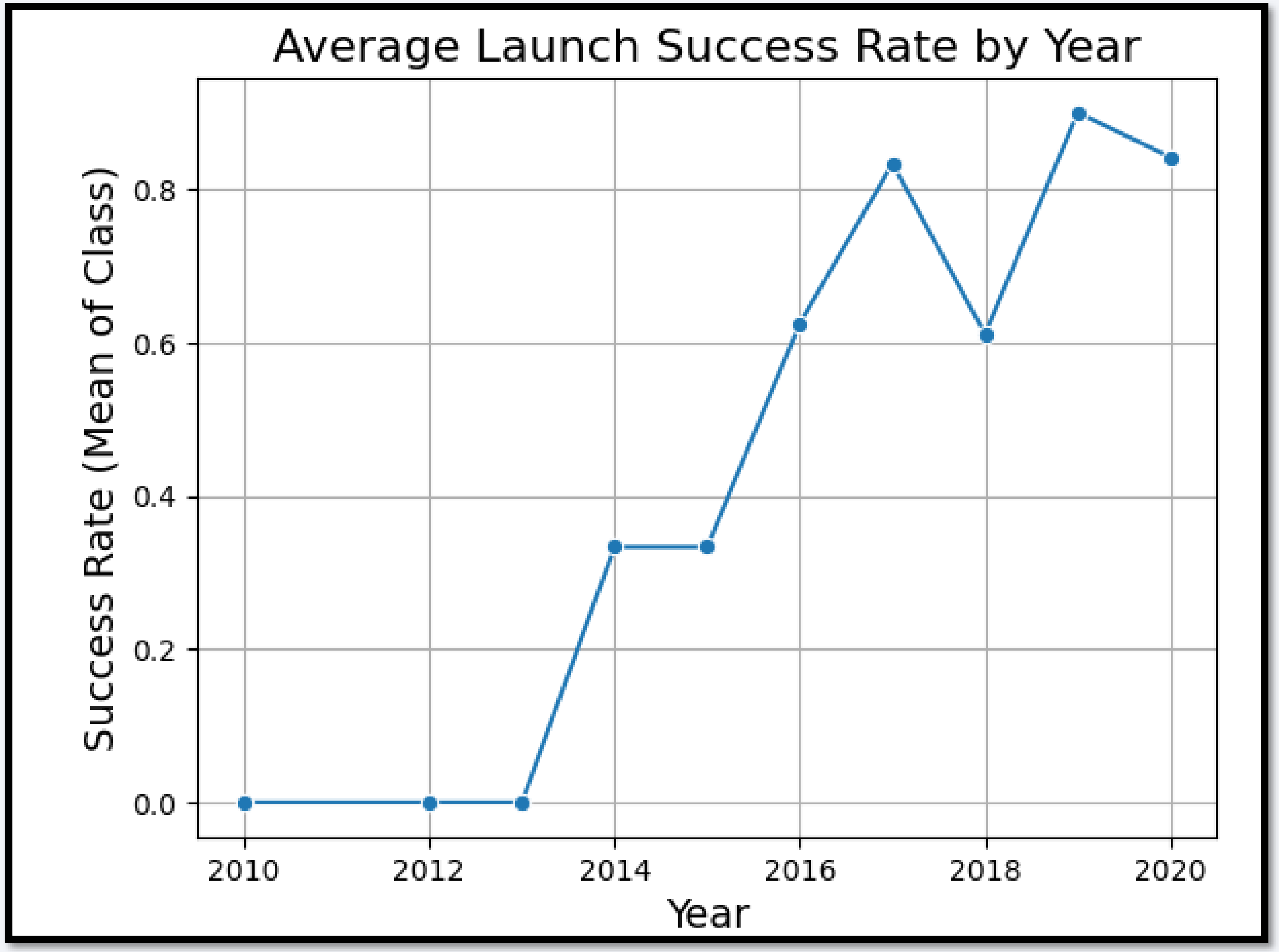
Flight Number vs. Orbit Type



Payload vs. Orbit Type



Launch Success Yearly Trend



All Launch Site Names

```
%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

Launch Site Names Begin with 'CCA'

Task 2

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

Total Payload Mass

Task 3

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

```
]: %sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Customer == 'NASA (CRS)'
```

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

]:

SUM(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

Task 4

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

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Booster_Version == 'F9 v1.1'
```

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

AVG(PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

Task 5

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

Hint:Use min function

```
%sql SELECT MIN(Date) FROM SPACEXTABLE WHERE Landing_Outcome == 'Success (ground pad)'
```

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

MIN(Date)
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

```
%sql SELECT Booster_Version FROM SPACEXTABLE WHERE 6000 > PAYLOAD_MASS__KG_ AND PAYLOAD_MASS__KG_ > 4000 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

Total Number of Successful and Failure Mission Outcomes

Task 7

List the total number of successful and failure mission outcomes

```
%sql SELECT (SELECT COUNT(*) FROM SPACEXTABLE WHERE Landing_Outcome LIKE 'Success%') AS Success_Count, (SELECT COUNT(*) FROM SPACEXTABLE WHERE Landing_Outcome LIKE 'Failure%') AS Failure_Count
```

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

Success_Count	Failure_Count
61	10

Boosters Carried Maximum Payload

Task 8

List all the booster_versions that 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

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.

```
%sql SELECT substr(Date, 6,2) as month, Landing_Outcome, Booster_Version, Launch_Site FROM SPACEXTABLE WHERE Landing_Outcome LIKE 'Failure (drone ship)' AND substr(Date,0,5)='2015'
```

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

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

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

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 (SELECT COUNT(Landing_Outcome) FROM SPACEXTABLE WHERE Landing_Outcome LIKE 'Failure (drone ship)' AND Date BETWEEN '2010-06-04' and '2017-03-20') AS Failure_Count,  (SELECT COUNT(Lan
```

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

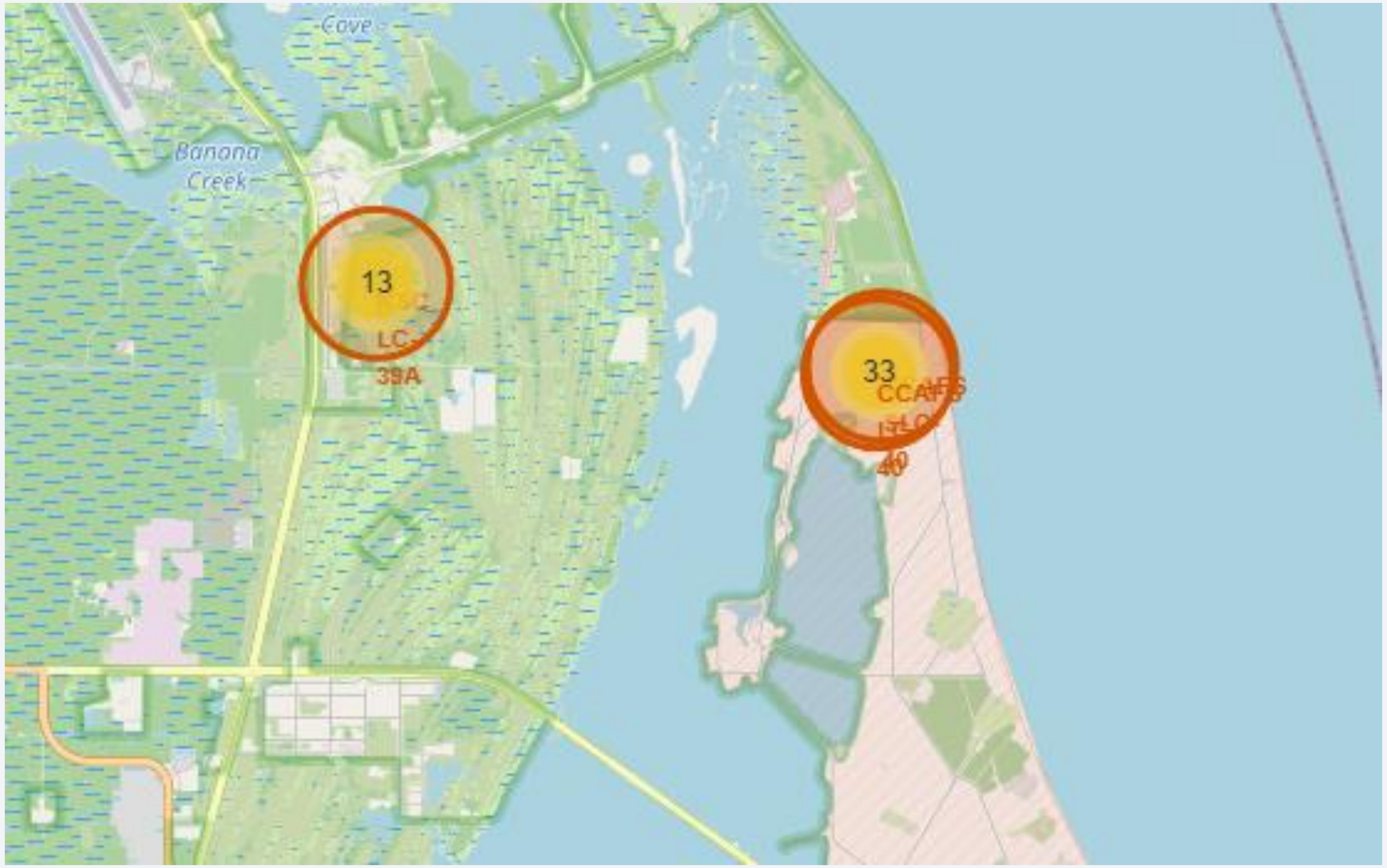
Failure_Count	Success_count
5	3

Section 3

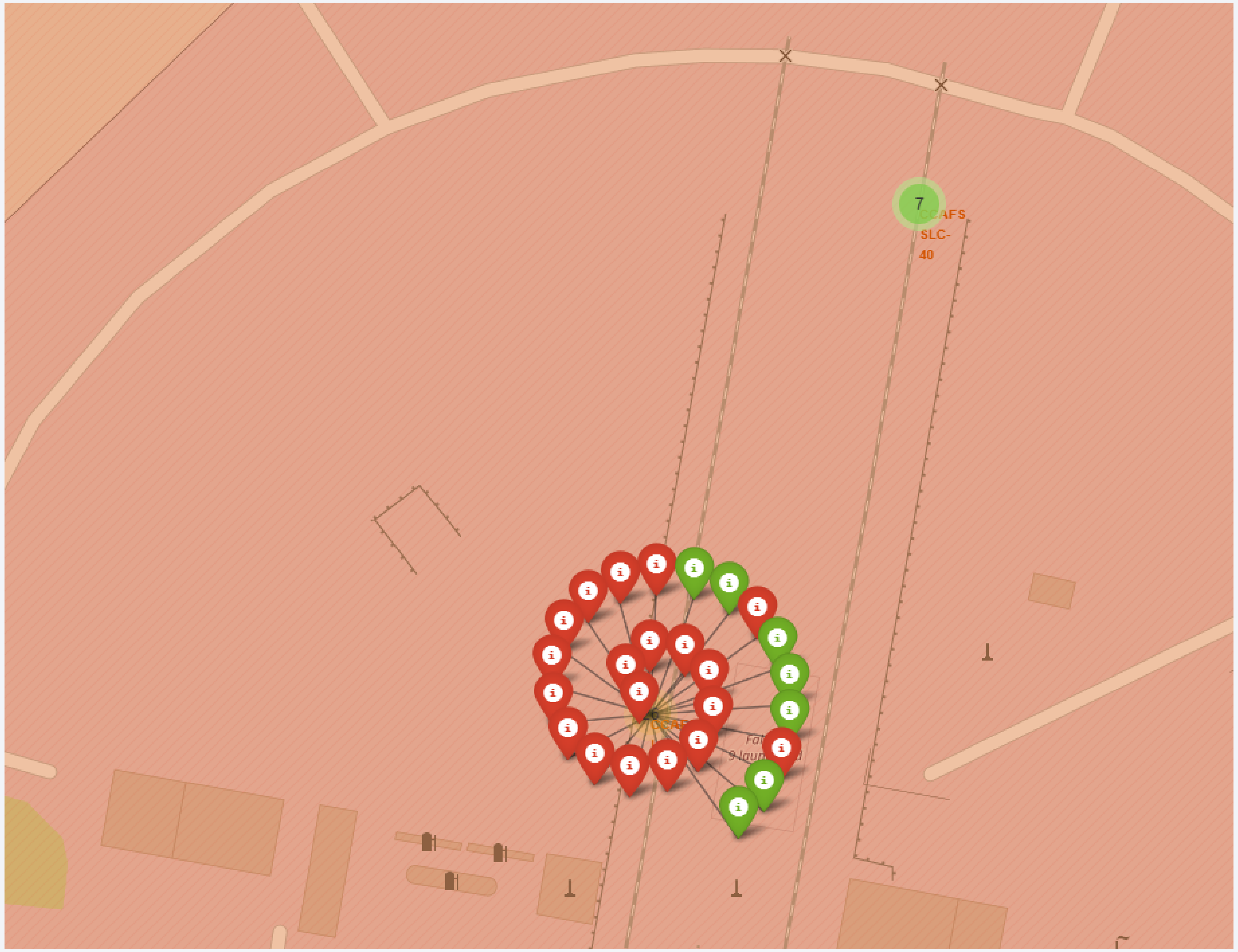
Launch Sites Proximities Analysis

Launch Sites on map by Folium

Launch Sites with outcomes/attempts



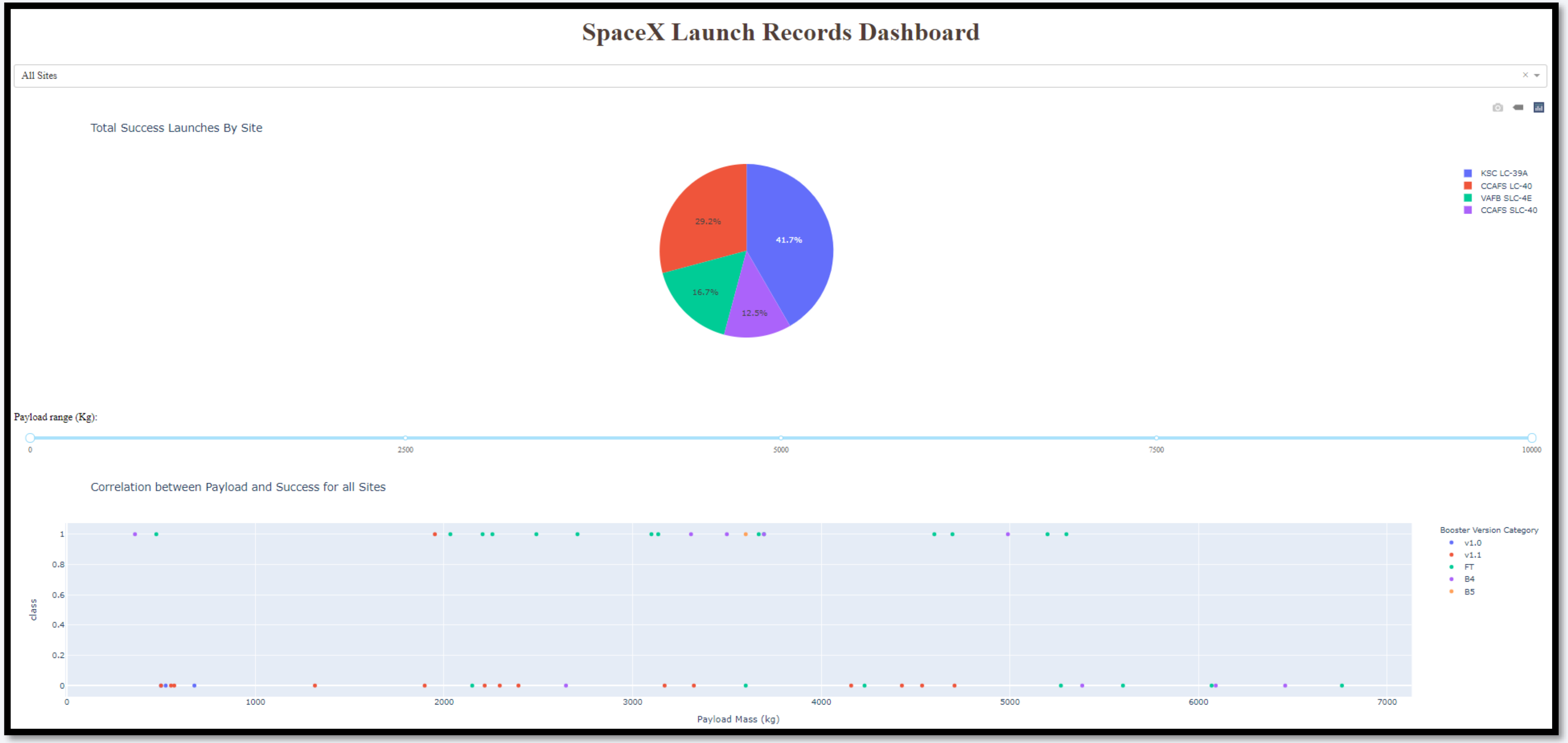
Markers for succesful/failure attempts by Folium



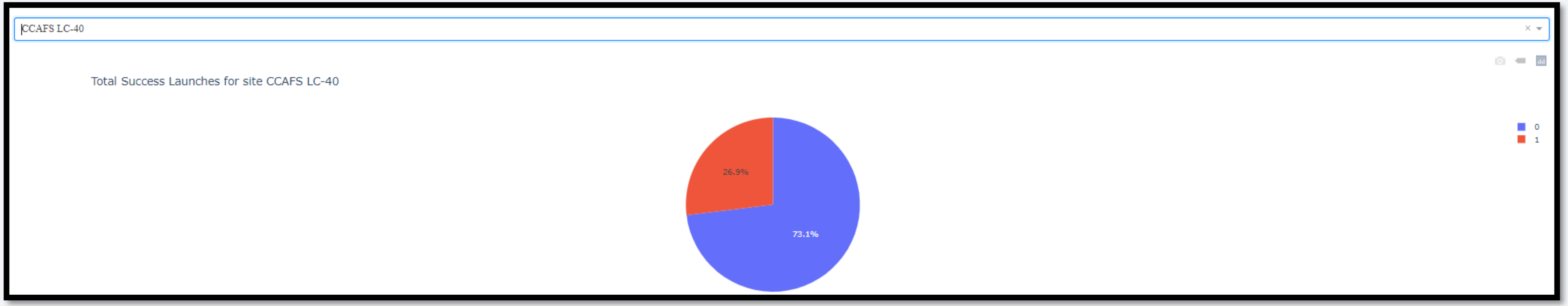
Section 4

Build a Dashboard with Plotly Dash

Dashboard by Plotly



Dashboard - piechart



Section 5

Predictive Analysis (Classification)

Classification Accuracy - Knn

Create a k nearest neighbors object then create a `GridSearchCV` object `knn_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

[40]:
```python
parameters_knn = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
              'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
              'p': [1,2]}

KNN = KNeighborsClassifier()
```

[41]:
```python
knn_cv = GridSearchCV(estimator=KNN, param_grid=parameters_knn, scoring='accuracy', cv=10, return_train_score=True)
knn_cv.fit(X_train, y_train)
```

[41]:
```
▸ GridSearchCV
▸ estimator: KNeighborsClassifier
    ▸ KNeighborsClassifier
```

[42]:
```python
print("tuned hpyerparameters :(best parameters) ",knn_cv.best_params_)
print("accuracy :",knn_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
accuracy : 0.8482142857142858
```

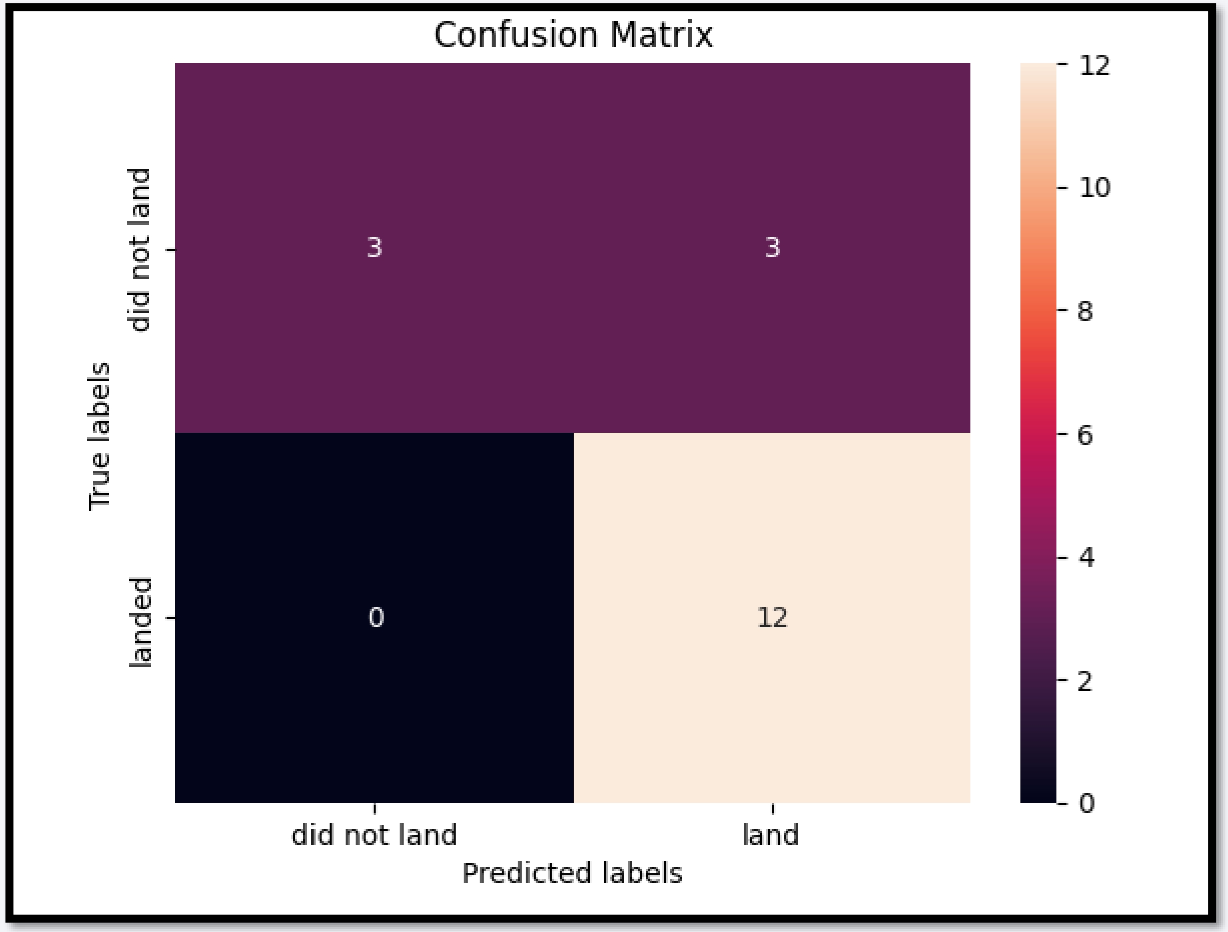
TASK 11

Calculate the accuracy of knn_cv on the test data using the method `score`:

[43]:
```python
accuracy_knn = knn_cv.score(X_test, y_test)
print(f"Accuracy for knn is {accuracy_knn}")
```

```
Accuracy for knn is 0.8333333333333334
```

Confusion Matrix of best performing model



Thank you!