IBM Developer
SKILLS NETWORK

# Winning Space Race with Data Science

Alejandra Miño S
05/28/2025

## Outline

## Executive Summary

- In this project, I aimed to predict the success of Falcon 9 first-stage landings using historical SpaceX data. I began by collecting datasets through APIs, web scraping, and CSV files, then performed data wrangling to clean and merge the information. I conducted Exploratory Data Analysis (EDA) using SQL, Python (Pandas, Matplotlib, Seaborn), and created interactive visualizations with Folium and Plotly to uncover relationships between launch outcomes, sites, payloads, and booster versions.

- To enable interactive analysis, I built a dashboard in Dash where users can filter data by launch site and payload range. For predictive modeling, I trained and tuned multiple classifiers including Logistic Regression, SVM, Decision Tree, and KNN, using GridSearchCV for hyperparameter optimization. Among them, the SVM with an RBF kernel achieved the highest accuracy on the test set. This project allowed me to apply the full data science pipeline—from data acquisition to deployment—and demonstrate how predictive analytics can support decision-making in aerospace operations.

# Introduction

As part of SpaceX's mission to reduce the cost of space exploration, they aim to make Falcon 9 first-stage boosters reusable by landing them after launch. The success or failure of these landings depends on multiple factors such as launch site, payload mass, booster version, and mission orbit.

In this project, I explored and analyzed SpaceX launch data to answer the following key questions:

- Which launch sites have the highest success rates?
- Is payload mass correlated with landing outcomes?
- Can we accurately predict the success of a booster landing using machine learning models?

By addressing these questions, I sought to identify operational patterns and build predictive models to support better launch planning and resource allocation

Section 1

## Methodology

# Methodology - Executive Summary

To complete this project, I followed a comprehensive data science workflow:

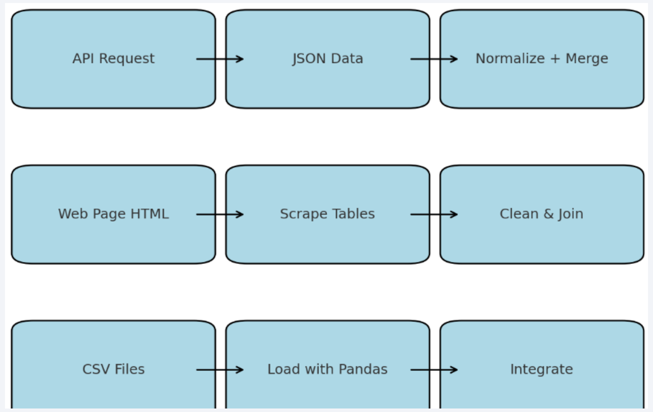
- Data Collection: Gathering SpaceX launch data from public APIs, performed web scraping, and downloaded datasets from cloud repositories.
- Data Wrangling: Cleaning and normalizing multiple data sources, handled missing values, merged datasets, and prepared structured tables for analysis.
- Data Processing: Transforming features, standardizing payload mass values, and engineered variables for classification tasks.
- Exploratory Data Analysis (EDA):I used Pandas, SQL, and visualization libraries to examine relationships among launch outcomes, payloads, and sites.
- Interactive Visual Analytics: I built an interactive map using Folium to explore launch site proximities and constructed a real-time dashboard using Plotly Dash for site and payload filtering.
- Predictive Analysis: I developed, tuned, and evaluated four classifiers (Logistic Regression, SVM, Decision Trees, k-NN) using GridSearchCV and confusion matrices to identify the best-performing model.

This structured methodology allowed me to combine analytical insight with interactive exploration and predictive intelligence.

# Data Collection

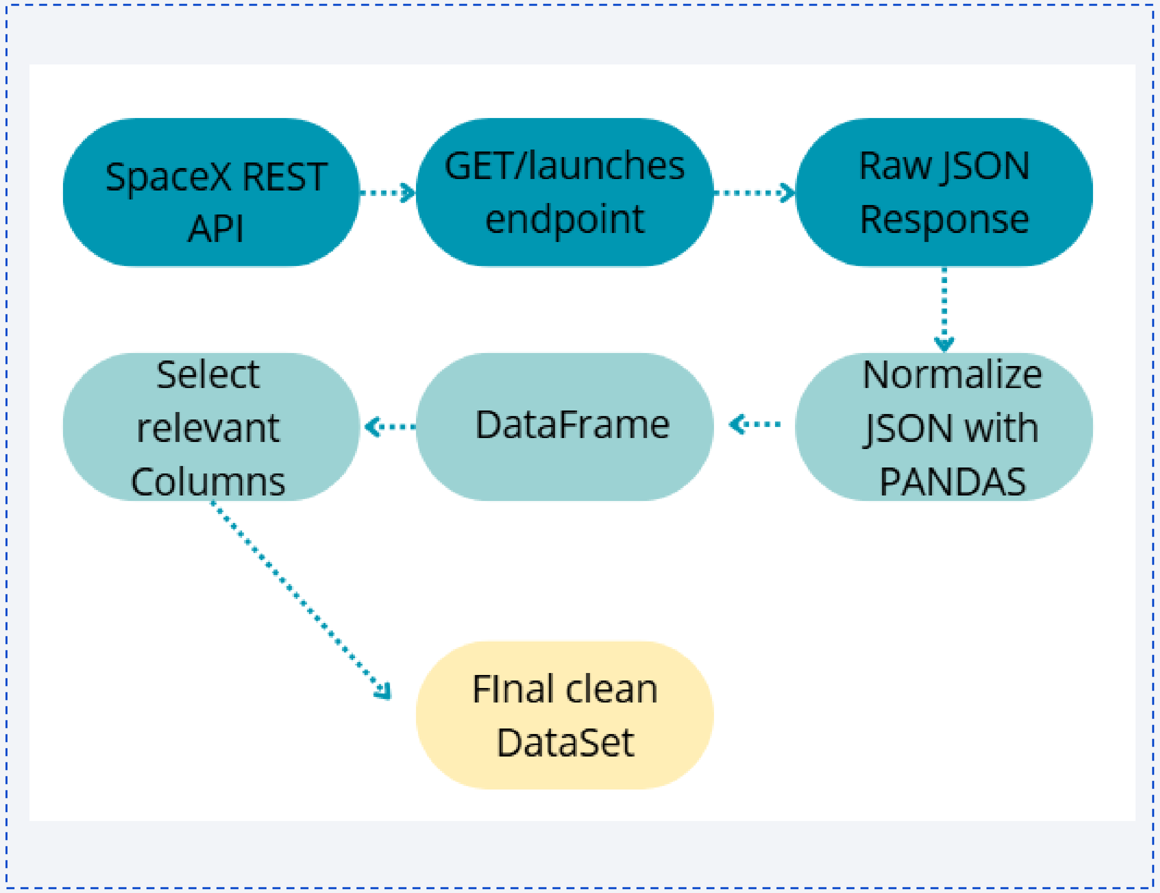
To build a comprehensive dataset for SpaceX launches, I followed a multi-source data collection strategy:

- Sources and Techniques:API Access: I used the SpaceX Launch API to retrieve real-time structured data on rocket launches, including payload, success outcome, and site info.
- Web Scraping: I scraped additional information about rocket boosters and payload classes using BeautifulSoup.
- CSV Datasets: I downloaded and imported supplementary datasets from IBM Skills Network cloud storage, such as payload metrics and launch site locations.



# Data Collection – SpaceX API

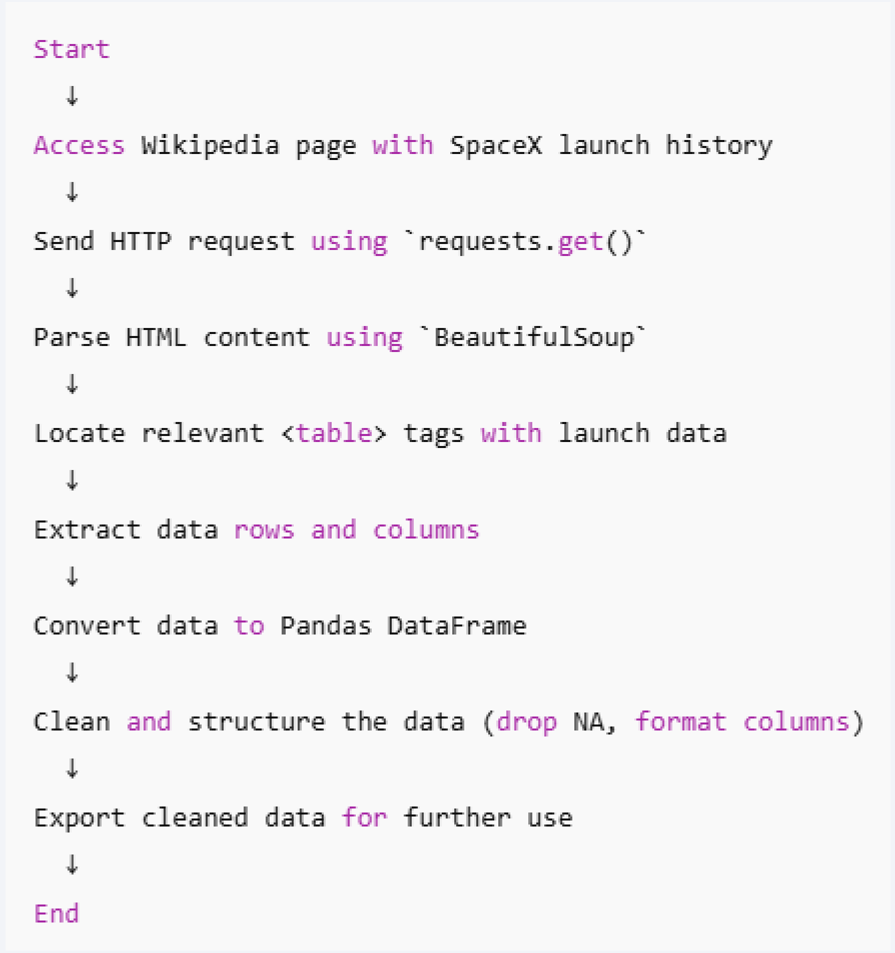
- https://github.com/AMinoSilva/NotebookUNAB/blob/main/jupyter-labs-spacex-data-collection-api.ipynb



# Data Collection - Scraping

Key Phrases – Web Scraping Process Extracted historical SpaceX launch data from Wikipedia.

- Used Python’s BeautifulSoup and requests libraries for HTML parsing.
- Targeted relevant HTML tables containing launch history.
- Converted HTML tables into Pandas DataFrames.
- Cleaned and structured scraped data for downstream analysis.
- Validated scraped data consistency with API data.
- https://github.com/AMinoSilva/NotebookUNAB/blob/main/jupyter-labs-webscraping.ipynb

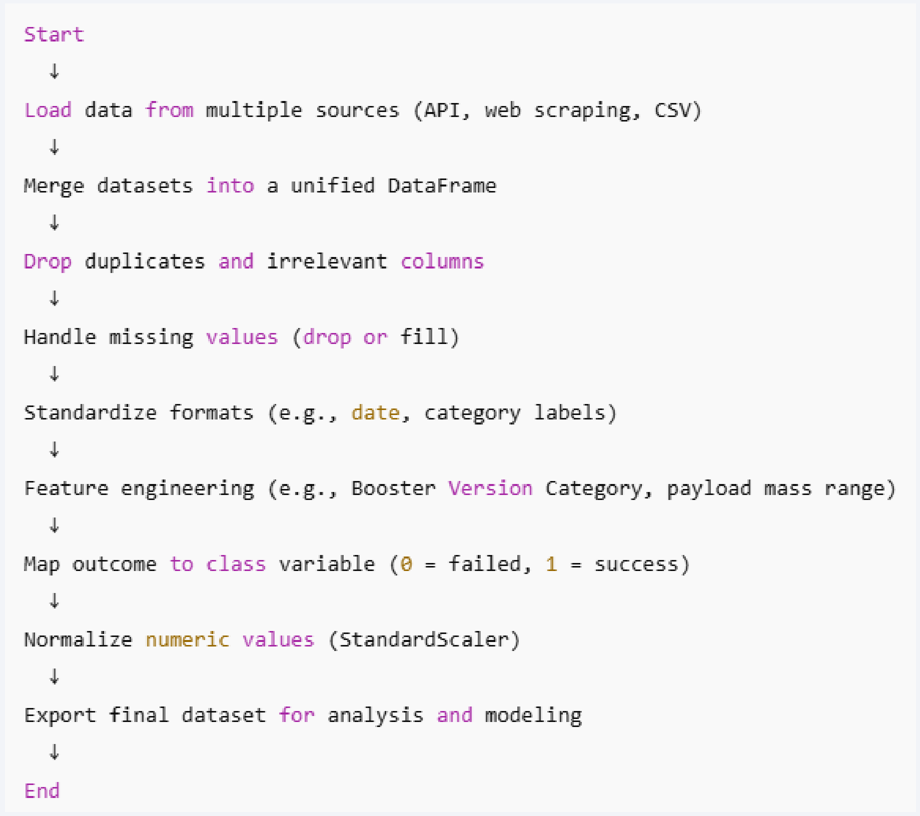


```
Start
  ↓
Access Wikipedia page with SpaceX launch history
  ↓
Send HTTP request using `requests.get()`
  ↓
Parse HTML content using `BeautifulSoup`
  ↓
Locate relevant <table> tags with launch data
  ↓
Extract data rows and columns
  ↓
Convert data to Pandas DataFrame
  ↓
Clean and structure the data (drop NA, format columns)
  ↓
Export cleaned data for further use
  ↓
End
```

# Data Wrangling

Key Phrases – Data Wrangling Process

- Merged datasets from multiple sources: API, web scraping, and CSV files.
- Removed duplicate entries and irrelevant columns.
- Handled missing values by imputation or deletion, depending on context.
- Renamed columns and standardized data formats (e.g., date, strings, numeric types).Mapped categorical values (e.g., success/failure) to numerical classes for ML models.
- Generated new features from raw data (e.g., Booster Version Category).
- Normalized and scaled numeric features using StandardScaler.
- Exported cleaned datasets for EDA, visualization, and machine learning
- https://github.com/AMinoSilva/NotebookUNAB/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb



```
Start
  ↓
Load data from multiple sources (API, web scraping, CSV)
  ↓
Merge datasets into a unified DataFrame
  ↓
Drop duplicates and irrelevant columns
  ↓
Handle missing values (drop or fill)
  ↓
Standardize formats (e.g., date, category labels)
  ↓
Feature engineering (e.g., Booster Version Category, payload mass range)
  ↓
Map outcome to class variable (0 = failed, 1 = success)
  ↓
Normalize numeric values (StandardScaler)
  ↓
Export final dataset for analysis and modeling
  ↓
End
```

# EDA with Data Visualization

To understand SpaceX launch performance and identify key trends, I created multiple visualizations using **Matplotlib**, **Seaborn**, and **Plotly Express**. Each chart was chosen to uncover specific insights:

Bar Charts Purpose: Compare total launches and successes by launch site.

Why: Helps visually determine which sites are most active and most successful.

Pie Charts Purpose: Show the proportion of successful vs. failed launches.

Why: Clearly conveys overall mission success rate or breakdown per site.

Histogram / Distribution Plots Purpose: Analyze distribution of payload masses.

Why: Understand payload characteristics and potential outliers.

Scatter Plots Purpose: Examine the correlation between payload mass and launch outcome.

Why: Identify patterns that may impact success based on payload weight.

Box Plots Purpose: Compare launch success across booster version categories.

Why: Show variability in success and highlight performance differences by booster.

**https://github.com/AMinoSilva/NotebookUNAB/blob/main/edadataviz%20(1).ipynb**

## EDA with SQL

To perform exploratory data analysis at scale and with precision, I used SQL queries directly on the SpaceX dataset stored in an SQLite database. Key queries included:

- Count of total launches:
  - SELECT COUNT(*) FROM SPACEXDATASET
- Launch count by site:
  - SELECT Launch_Site, COUNT(*) AS Launch_Count FROM SPACEXDATASET GROUP BY Launch_Site
- Success count per site:
  - SELECT Launch_Site, SUM(Class) AS Successful_Launches FROM SPACEXDATASET GROUP BY Launch_Site
- Average payload mass by lauch site:
  - SELECT Launch_Site, AVG(PAYLOAD_MASS__KG_) AS Avg_Payload FROM SPACEXDATASET GROUP BY Launch_Site
- Success rate bybooster version
  - SELECT Booster_Version, AVG(Class) AS Success_Rate FROM SPACEXDATASET GROUP BY Booster_Version
- Filtering launches with payload > 4000kg and success:
  - SELECT * FROM SPACEXDATASET WHERE PAYLOAD_MASS__KG_ > 4000 AND Class = 1

## Build an Interactive Map with Folium

Using the Folium library, I developed an interactive map to visually analyze the spatial context of SpaceX launch sites and their proximity to important infrastructures

- **Markers**: Added to visually locate each launch site.
- **CircleMarkers**: Used to highlight key nearby features such as highways, railways, and coastlines
- **Polylines**: Drawn to connect the launch sites with nearby locations to calculate distances
- **MousePosition**: Enabled to capture coordinates while exploring the map interactively

https://github.com/AMinoSilva/NotebookUNAB/blob/main/lab_jupyter_launch_site_location.jupyterlite.ipynb

## Build a Dashboard with Plotly Dash

**Dropdown Menu**: Implemented to select a specific launch site or view all sites at once

**Pie Chart**: Dynamically displays success vs. failure rates per launch site based on selection.

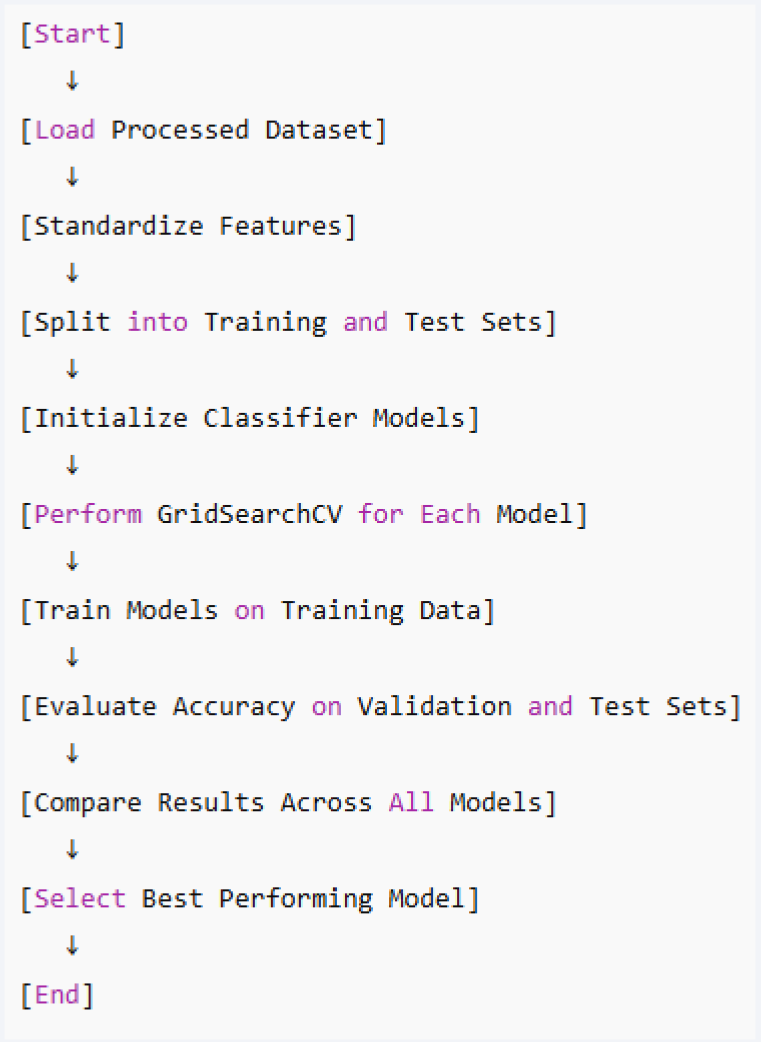
**Range Slider**: Allows users to filter data by payload mass to explore its correlation with success rates

**Scatter Plot**: Visualizes the relationship between payload mass and launch outcome, colored by booster version

## Predictive Analysis (Classification)

- Data Standardization using StandardScaler
- Train/Test Split using train_test_split (80/20)
- Model selection: Logistic Regression, SVM, Decision Tree, KNN
- Hyperparameter Tuning with GridSearchCV (cv=10)
- Performance Evaluation using Accuracy Score & Confusion Matrix
- Model Comparison on Test Set
- Best Model Selection based on test accuracy

https://github.com/AMinoSilva/NotebookUNAB/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb



```
[Start]
   ↓
[Load Processed Dataset]
   ↓
[Standardize Features]
   ↓
[Split into Training and Test Sets]
   ↓
[Initialize Classifier Models]
   ↓
[Perform GridSearchCV for Each Model]
   ↓
[Train Models on Training Data]
   ↓
[Evaluate Accuracy on Validation and Test Sets]
   ↓
[Compare Results Across All Models]
   ↓
[Select Best Performing Model]
   ↓
[End]
```

## Results

While performing this exploratory analysis, I uncovered key patterns in SpaceX launch successes by analyzing payload ranges, launch sites, and booster types. I visualized these findings using scatter plots, pie charts, and Folium maps to provide spatial insights. Interactive dashboards with Plotly Dash allowed dynamic filtering and exploration of mission outcomes. In the predictive phase, I built and tuned classification models—including Logistic Regression, SVM, Decision Tree, and KNN—selecting the best-performing model based on test accuracy. The Decision Tree model demonstrated the highest accuracy in predicting successful landings.

Section 2

## Insights drawn from EDA

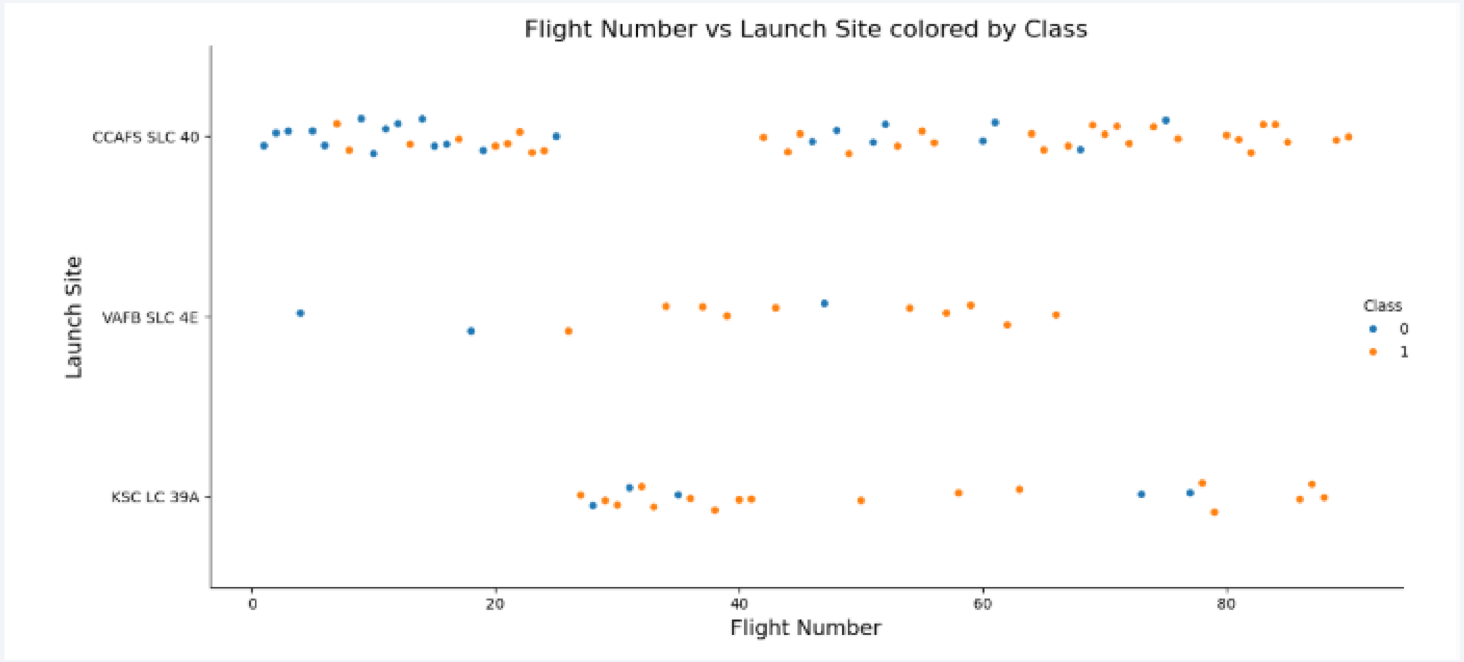
## Flight Number vs. Launch Site

CCAFS SLC 40 has the widest flight range, showing it's the most frequently used site.

KSC LC 39A appears later in the sequence with mostly successful launches, suggesting maturity.

VAFB SLC 4E has limited launches, indicating specialized or infrequent use.

General trend: Higher flight numbers show more successes, reflecting SpaceX's improvement over time



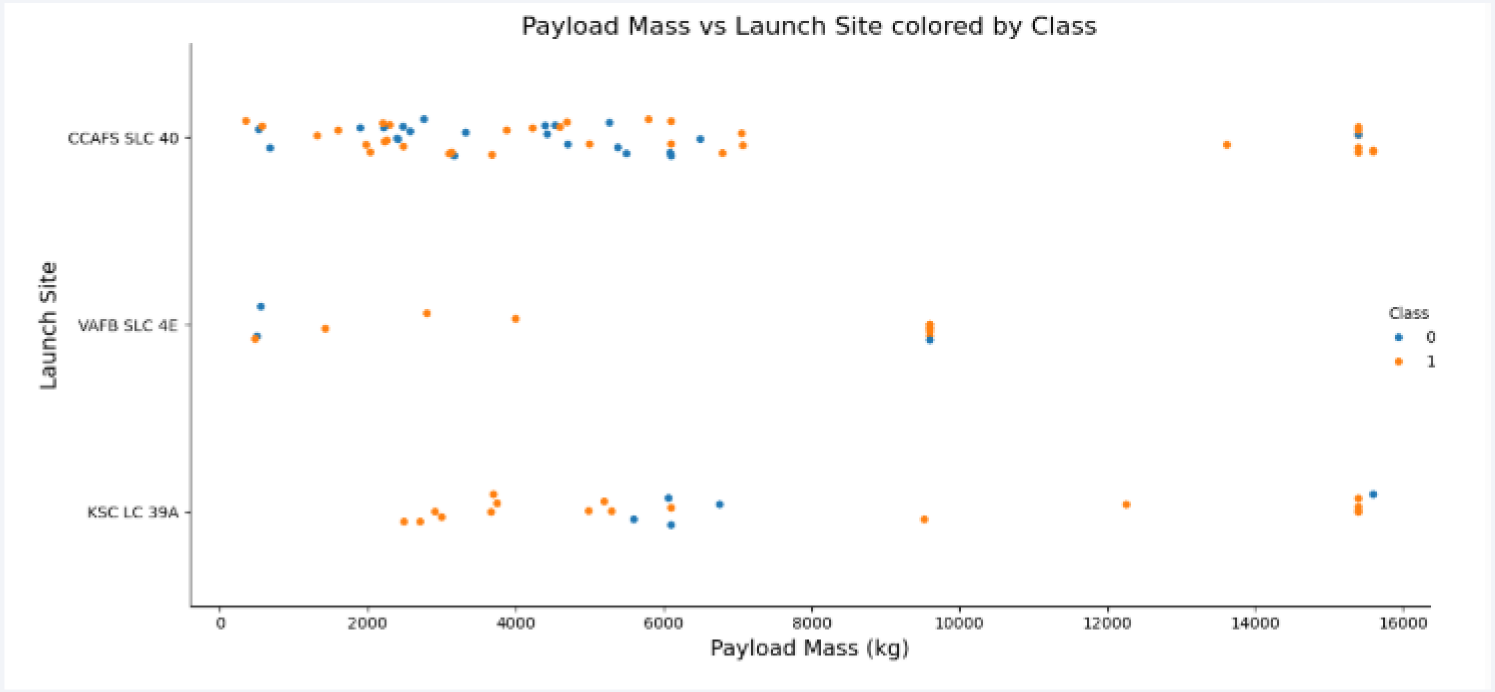
# Payload vs. Launch Site

Most launches fall within the 2,000–6,000 kg range.

CCAFS SLC 40 supports a wide payload range, including several heavy missions.

KSC LC 39A handled heavier payloads, often with successful outcomes.

No strong correlation between payload mass and success, indicating reliability across payload sizes

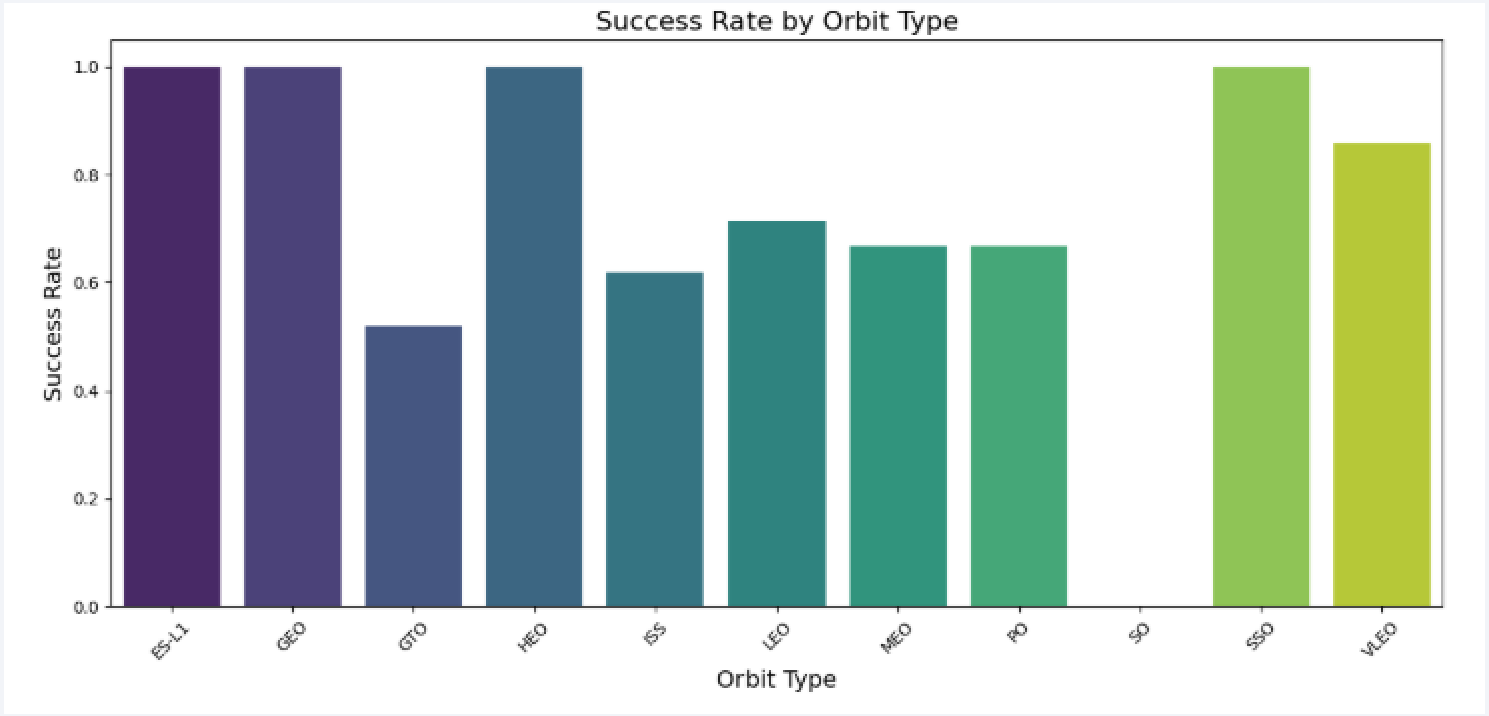


# Success Rate vs. Orbit Type

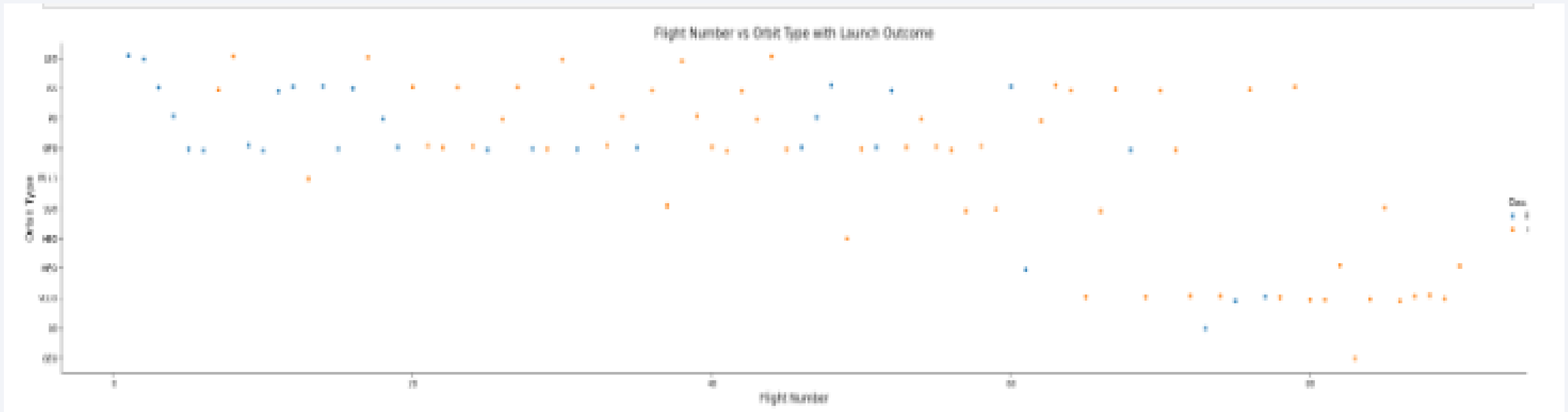
Orbits like ES-L1, GEO, HEO, and SSO show near or full success rates.

GTO and ES have noticeably lower success, indicating higher risk.

Overall, SpaceX performs well across most orbit types, with variation likely due to mission complexity

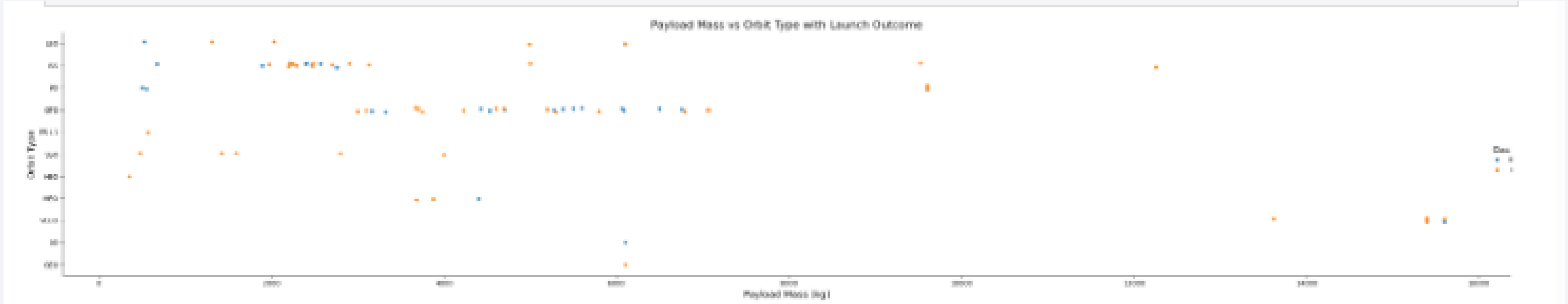


# Flight Number vs. Orbit Type



You can observe that in the LEO orbit, success seems to be related to the number of flights. Conversely, in the GTO orbit, there appears to be no relationship between flight number and success.

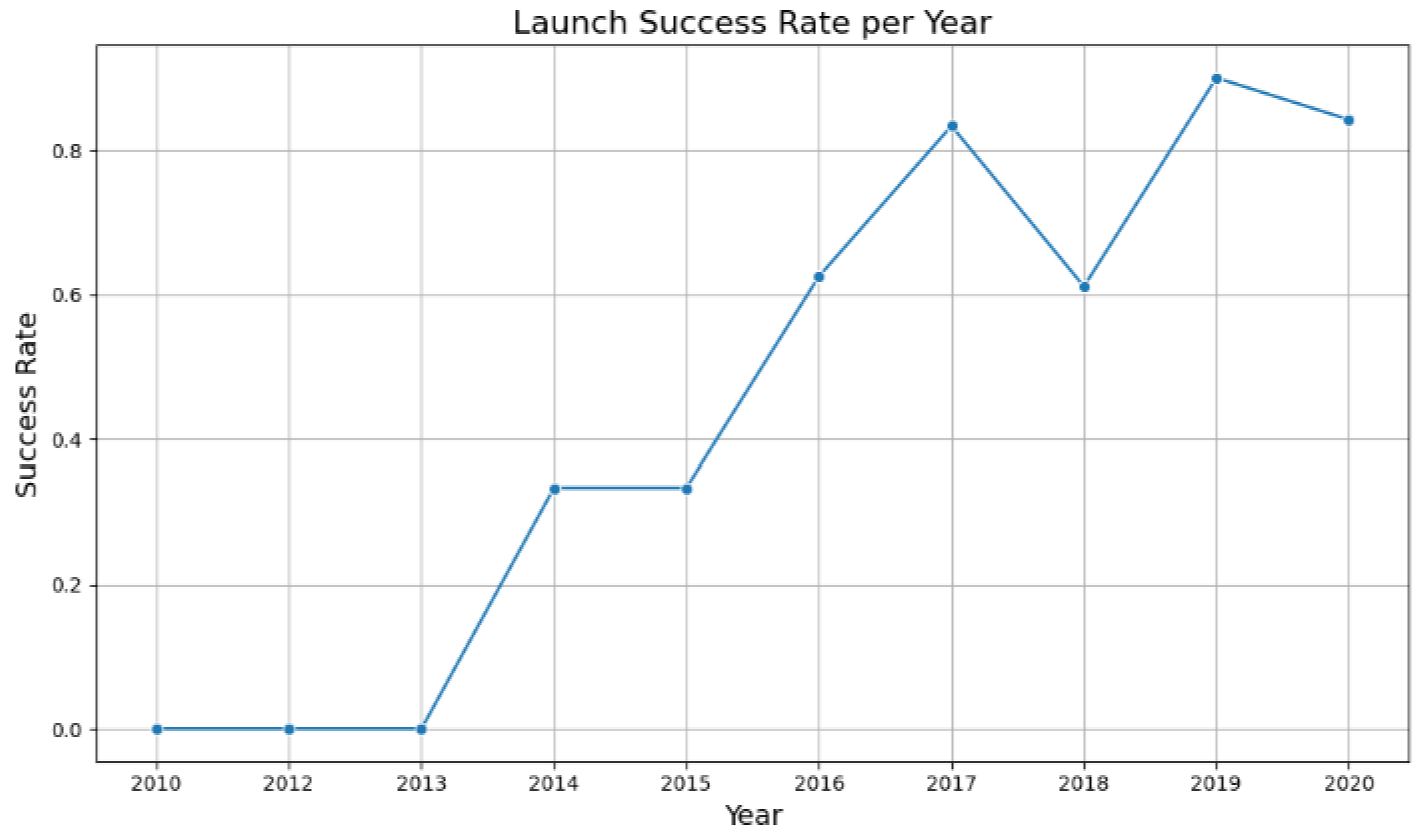
## Payload vs. Orbit Type



With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

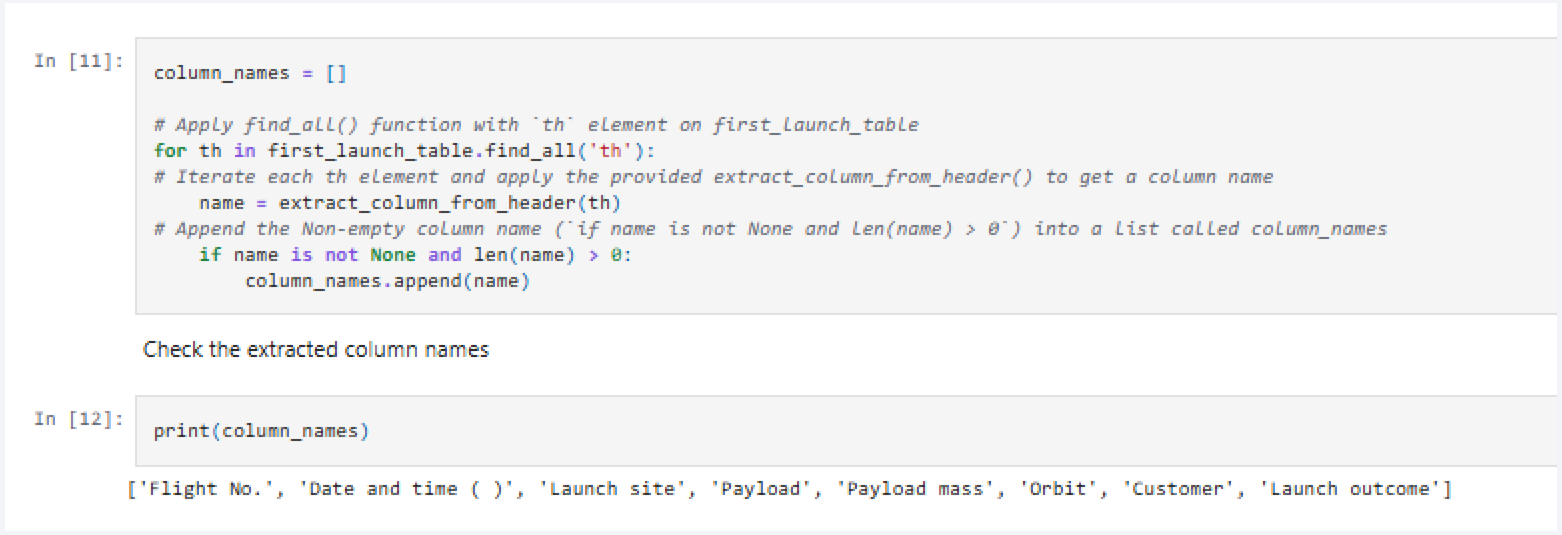
However, for GTO, it's difficult to distinguish between successful and unsuccessful landings as both outcomes are present.

## Launch Success Yearly Trend



you can observe that the sucess rate since 2013 kept increasing till 2020

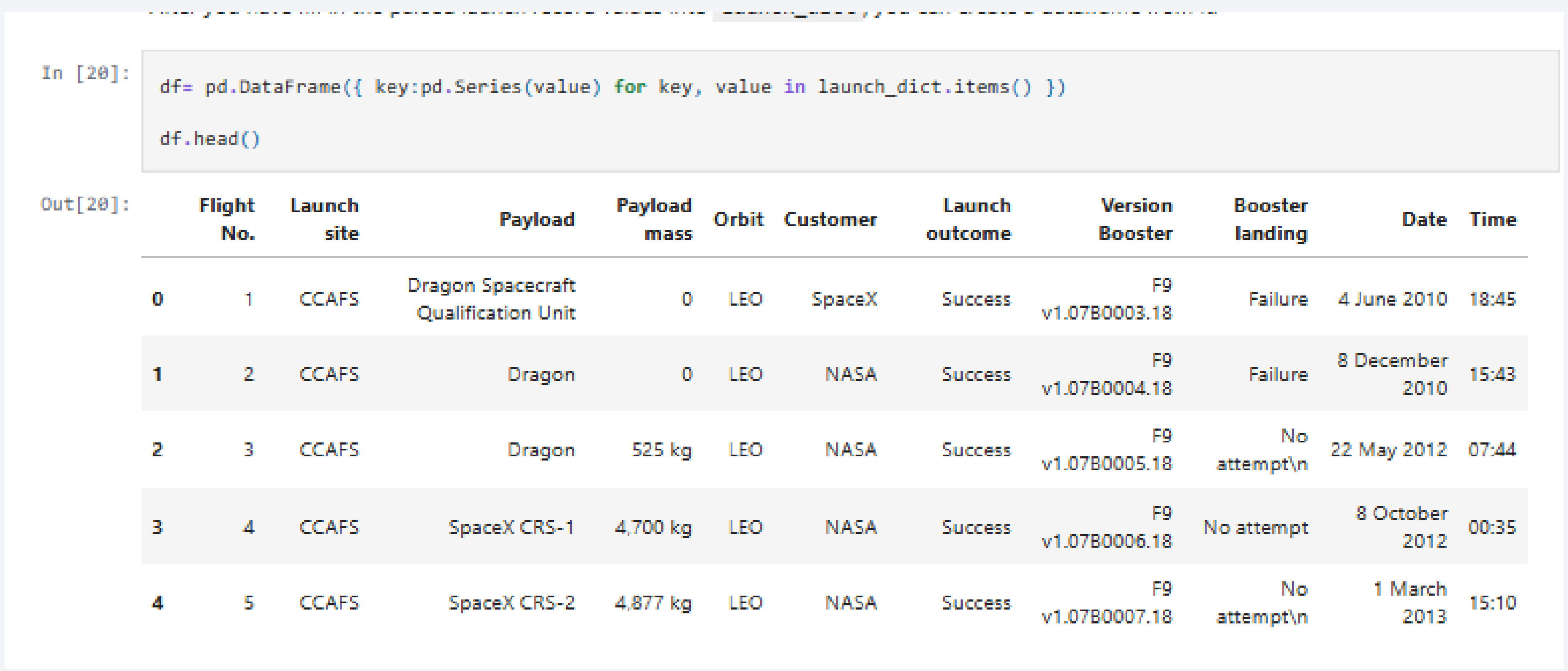
## All Launch Site Names


```
In [11]:
column_names = []

# Apply find_all() function with `th` element on first_launch_table
for th in first_launch_table.find_all('th'):
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
    name = extract_column_from_header(th)
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names
    if name is not None and len(name) > 0:
        column_names.append(name)
```

Check the extracted column names

```
In [12]:
print(column_names)

['Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome']
```

# Launch Site Names Begin with 'CCA'



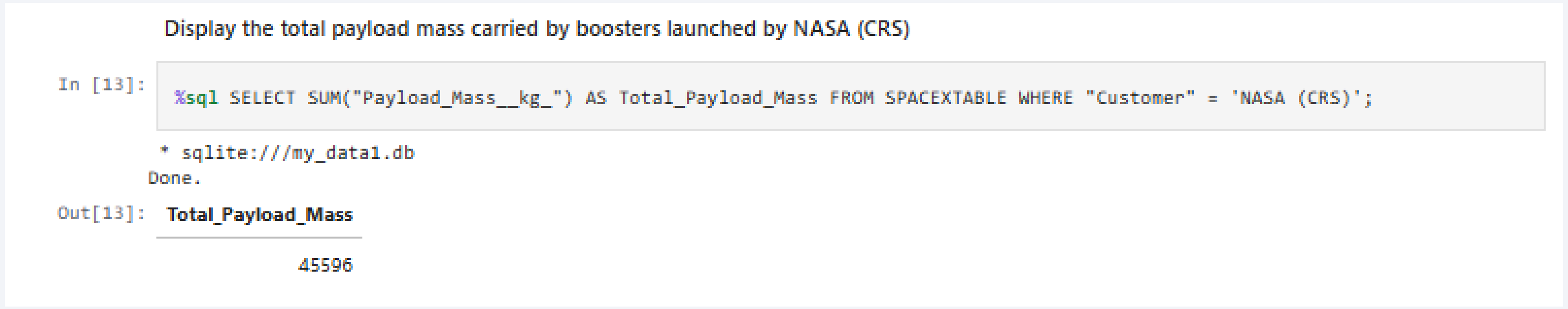
```
In [20]: df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })

         df.head()
```

Out[20]:

| | Flight No. | Launch site | Payload | Payload mass | Orbit | Customer | Launch outcome | Version Booster | Booster landing | Date | Time |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | CCAFS | Dragon Spacecraft Qualification Unit | 0 | LEO | SpaceX | Success | F9 v1.07B0003.18 | Failure | 4 June 2010 | 18:45 |
| 1 | 2 | CCAFS | Dragon | 0 | LEO | NASA | Success | F9 v1.07B0004.18 | Failure | 8 December 2010 | 15:43 |
| 2 | 3 | CCAFS | Dragon | 525 kg | LEO | NASA | Success | F9 v1.07B0005.18 | No attempt\n | 22 May 2012 | 07:44 |
| 3 | 4 | CCAFS | SpaceX CRS-1 | 4,700 kg | LEO | NASA | Success | F9 v1.07B0006.18 | No attempt | 8 October 2012 | 00:35 |
| 4 | 5 | CCAFS | SpaceX CRS-2 | 4,877 kg | LEO | NASA | Success | F9 v1.07B0007.18 | No attempt\n | 1 March 2013 | 15:10 |

# Total Payload Mass

- Using SQL, I calculated the total payload mass launched by NASA under the CRS (Commercial Resupply Services) program. The result shows a total of 45,596 kg, reflecting NASA's significant contribution to ISS resupply missions via SpaceX boosters. This confirms the operational scale of NASA's partnership with SpaceX.

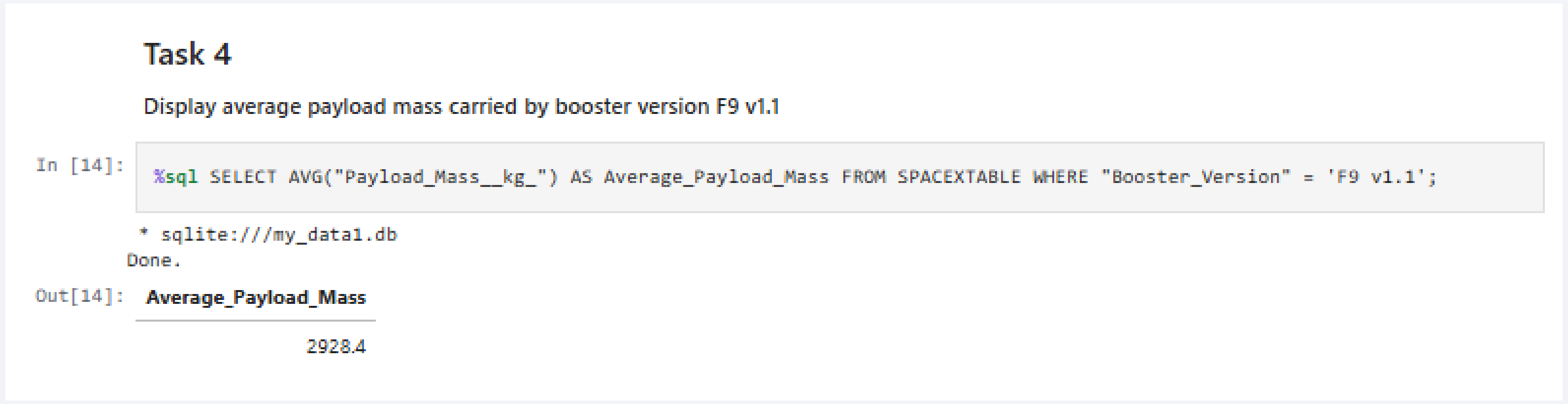


Display the total payload mass carried by boosters launched by NASA (CRS)

```
In [13]: %sql SELECT SUM("Payload_Mass__kg_") AS Total_Payload_Mass FROM SPACEXTABLE WHERE "Customer" = 'NASA (CRS)';

 * sqlite:///my_data1.db
Done.
Out[13]: Total_Payload_Mass
                     45596
```

## Average Payload Mass by F9 v1.1

I calculated the average payload mass carried by the Falcon 9 v1.1 booster version using SQL. The result shows an average of 2,928.4 kg, highlighting the typical payload capacity handled during this booster's operational phase.



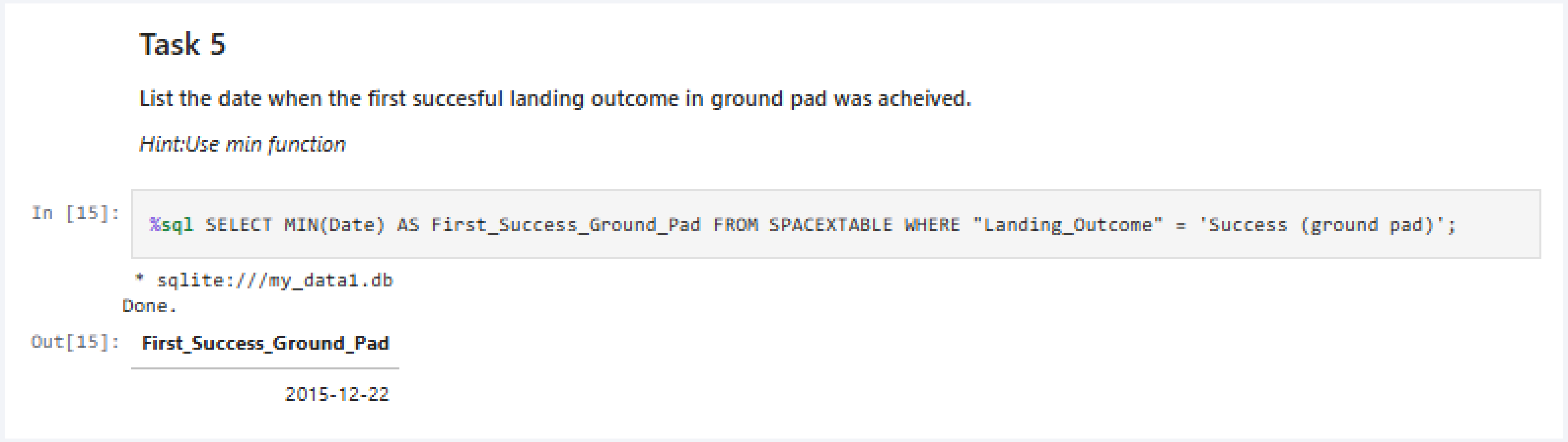
Task 4

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

```
In [14]: %sql SELECT AVG("Payload_Mass__kg_") AS Average_Payload_Mass FROM SPACEXTABLE WHERE "Booster_Version" = 'F9 v1.1';

 * sqlite:///my_data1.db
Done.

Out[14]: Average_Payload_Mass
                       2928.4
```

## First Successful Ground Landing Date

- Using SQL and the MIN() function, I identified that the first successful Falcon 9 landing on a ground pad occurred on December 22, 2015. This marked a major milestone in SpaceX's reusability efforts.



### Task 5

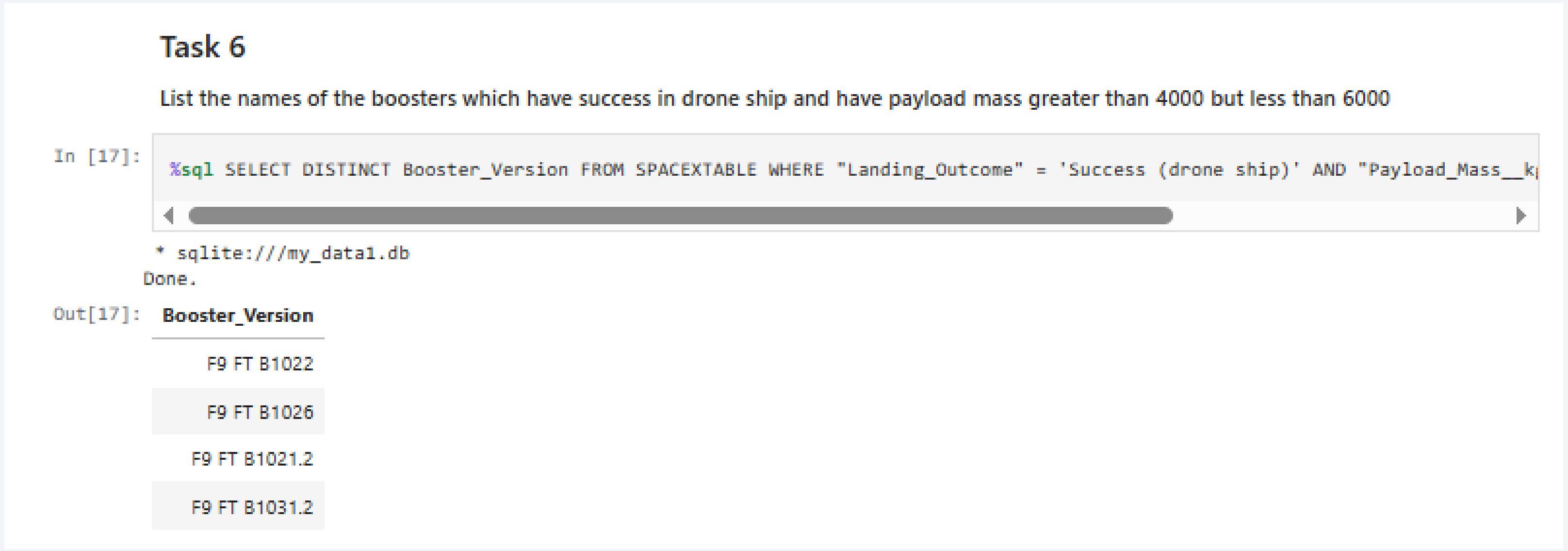
List the date when the first succesful landing outcome in ground pad was acheived.

*Hint:Use min function*

```
In [15]: %sql SELECT MIN(Date) AS First_Success_Ground_Pad FROM SPACEXTABLE WHERE "Landing_Outcome" = 'Success (ground pad)';

 * sqlite:///my_data1.db
Done.
Out[15]: First_Success_Ground_Pad
                       2015-12-22
```

## Successful Drone Ship Landing with Payload between 4000 and 6000

- Using SQL filters, I identified four booster versions that successfully landed on drone ships while carrying payloads between 4,000 and 6,000 kg. These include variants of F9 FT, demonstrating consistent performance within this weight range.

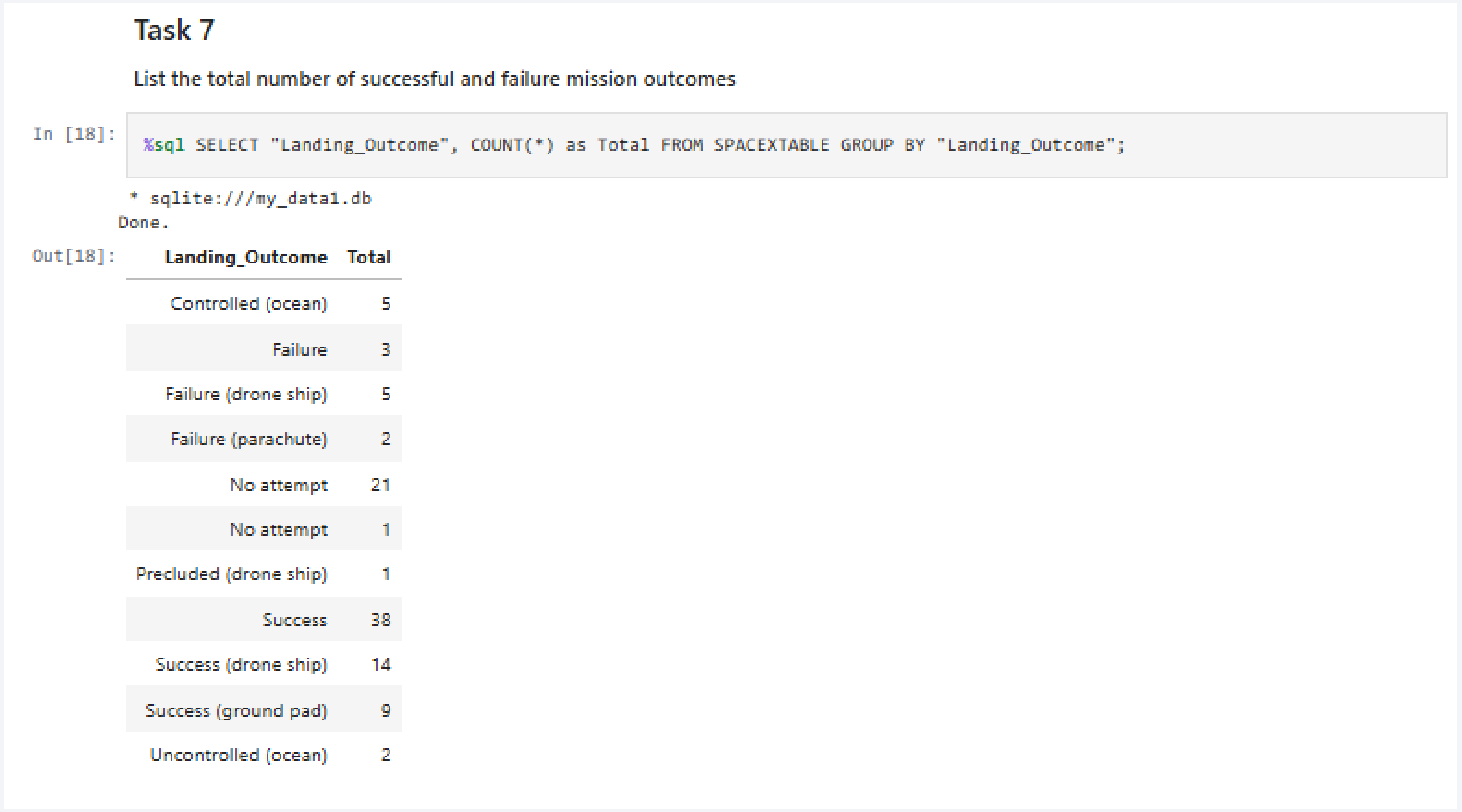


Task 6

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

```
In [17]:  %sql SELECT DISTINCT Booster_Version FROM SPACEXTABLE WHERE "Landing_Outcome" = 'Success (drone ship)' AND "Payload_Mass__k

 * sqlite:///my_data1.db
Done.
Out[17]:  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

```
In [18]: %sql SELECT "Landing_Outcome", COUNT(*) as Total FROM SPACEXTABLE GROUP BY "Landing_Outcome";

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

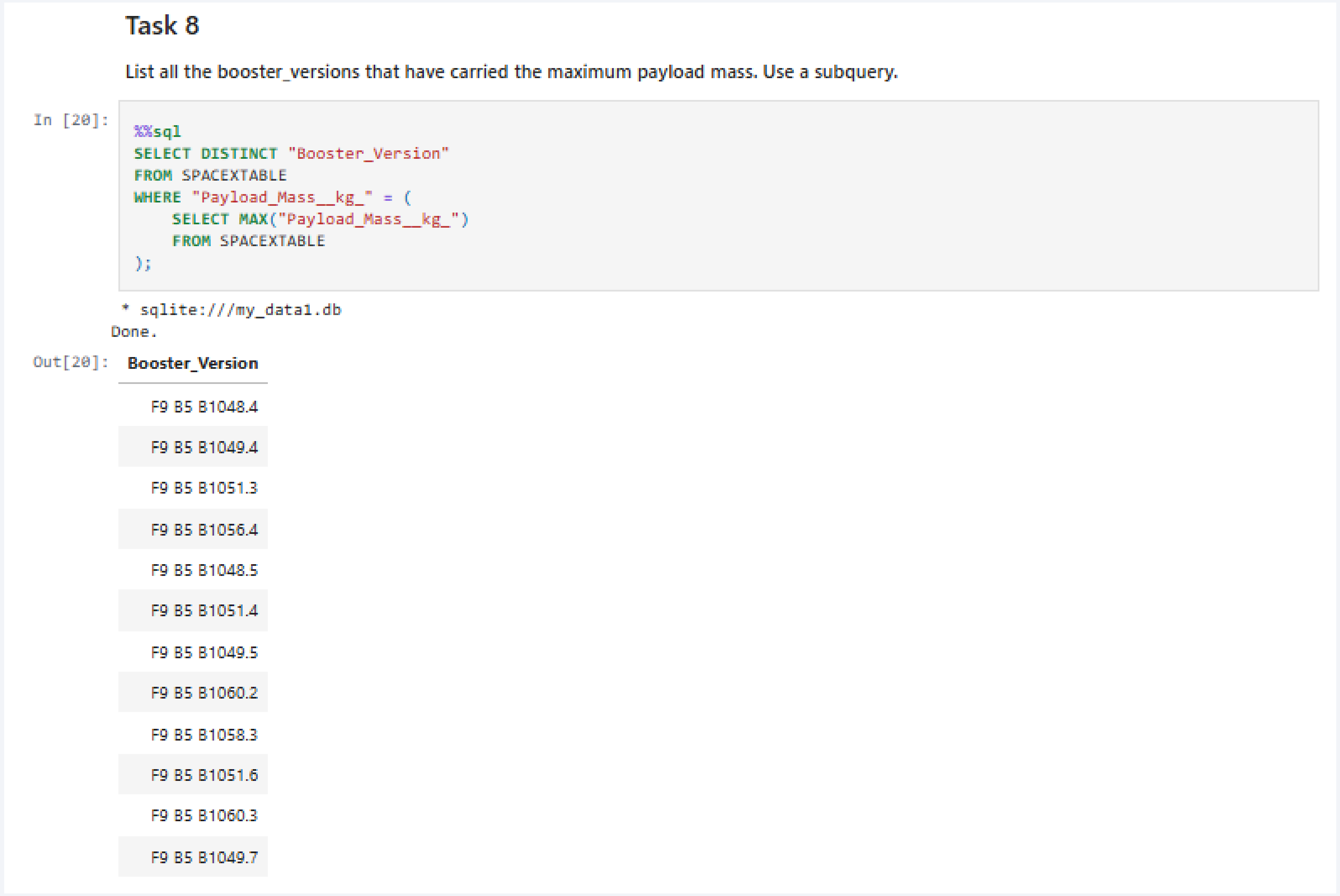
Out[18]:

| Landing_Outcome | Total |
|---|---|
| Controlled (ocean) | 5 |
| Failure | 3 |
| Failure (drone ship) | 5 |
| Failure (parachute) | 2 |
| No attempt | 21 |
| No attempt | 1 |
| Precluded (drone ship) | 1 |
| Success | 38 |
| Success (drone ship) | 14 |
| Success (ground pad) | 9 |
| Uncontrolled (ocean) | 2 |

- 38 generic successes, plus 23 specific ones (ground pad and drone ship).Several failed landings and 21 with no attempt, reflecting early mission phases or testing.
- This highlights SpaceX’s progress toward consistent successful recoveries.

## Boosters Carried Maximum Payload

- Using a SQL subquery, I retrieved all booster versions that carried the maximum payload mass recorded in the dataset. These versions—mainly F9 B5 variants—represent SpaceX's most powerful and capable launches to date.



### Task 8

List all the booster_versions that have carried the maximum payload mass. Use a subquery.

```
In [20]:
%%sql
SELECT DISTINCT "Booster_Version"
FROM SPACEXTABLE
WHERE "Payload_Mass__kg_" = (
    SELECT MAX("Payload_Mass__kg_")
    FROM SPACEXTABLE
);
```

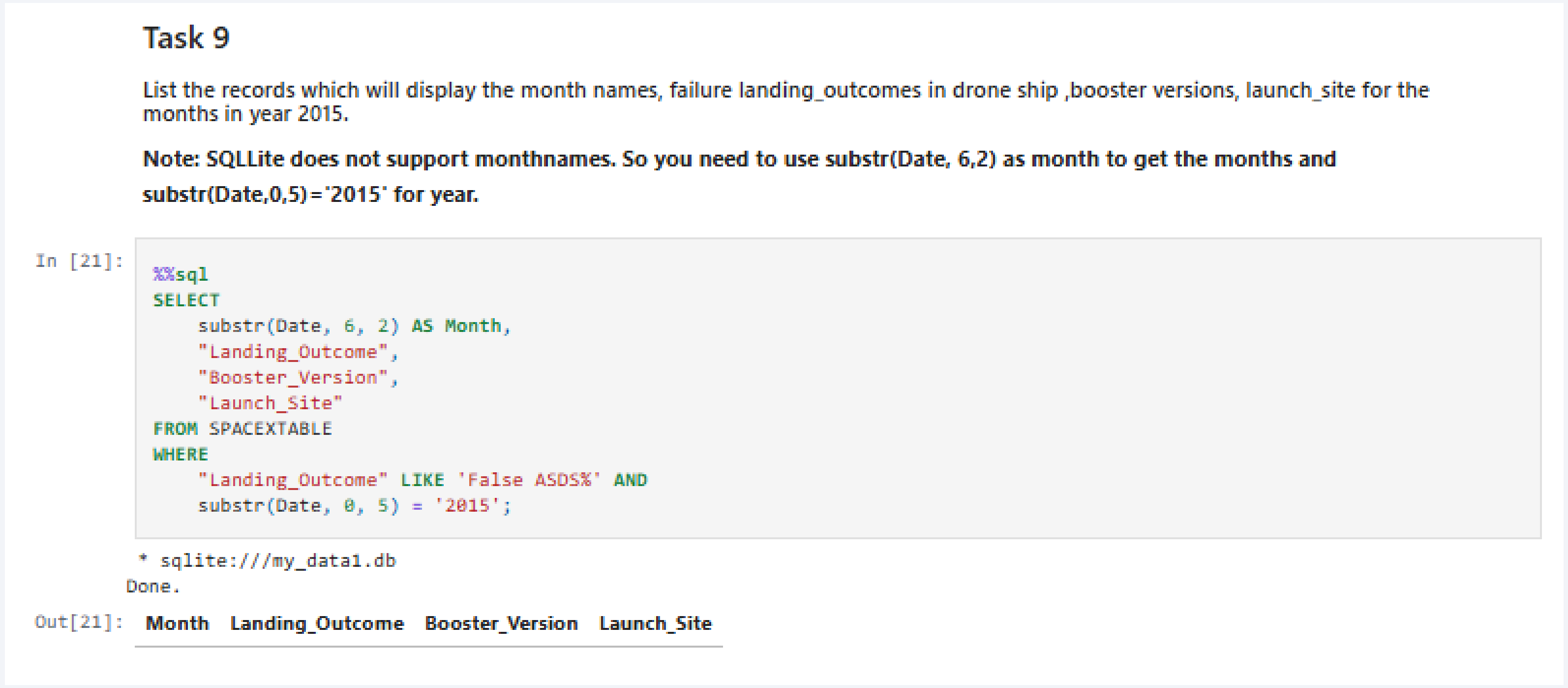
```
 * sqlite:///my_data1.db
Done.
```

Out[20]:

| 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

- I used SQL string functions (substr) to extract the month and year from the launch date, focusing on failed drone ship landings in 2015. The result highlights the booster versions and launch sites associated with those failed missions.



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.**

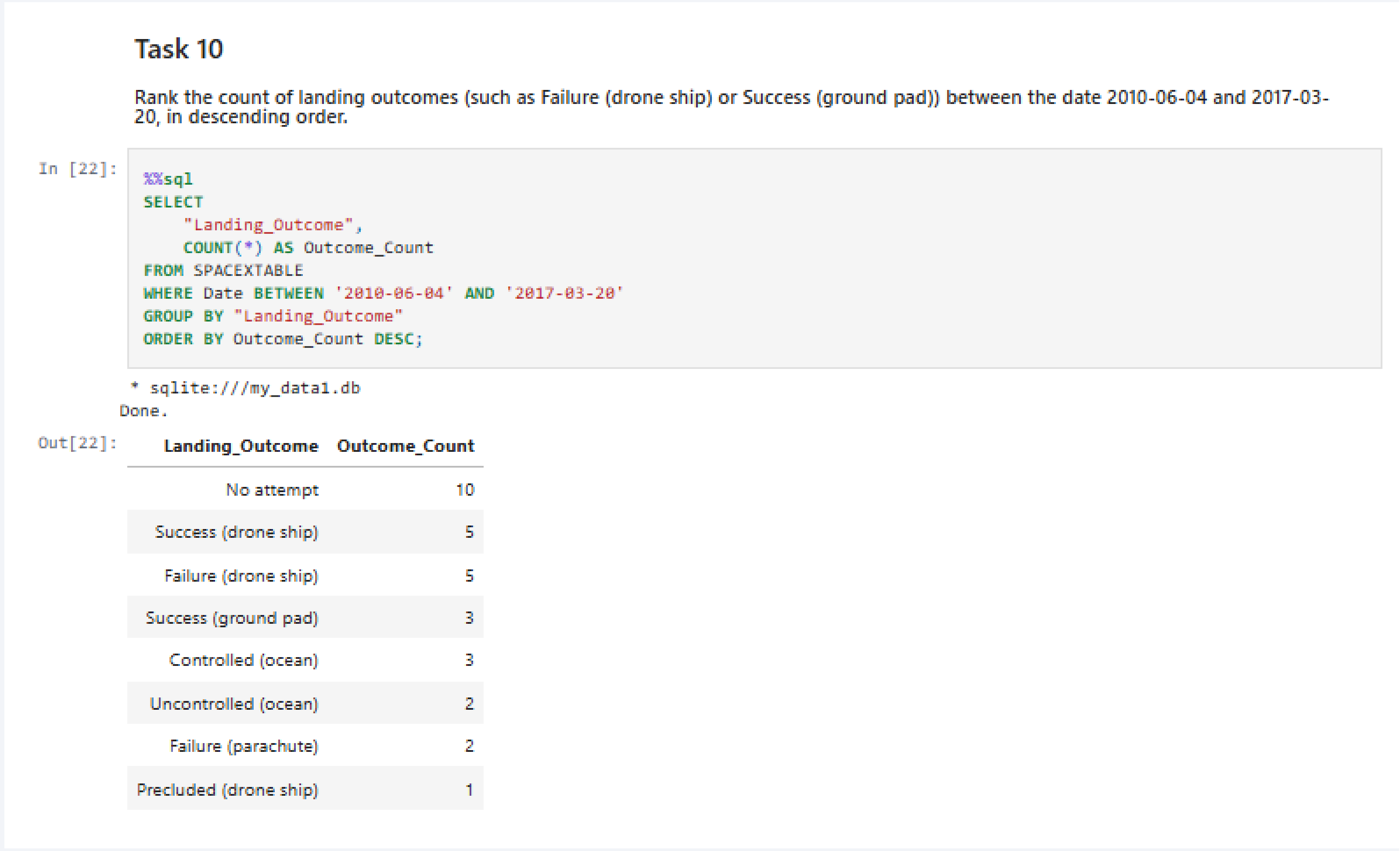
```
In [21]: %%sql
         SELECT
             substr(Date, 6, 2) AS Month,
             "Landing_Outcome",
             "Booster_Version",
             "Launch_Site"
         FROM SPACEXTABLE
         WHERE
             "Landing_Outcome" LIKE 'False ASDS%' AND
             substr(Date, 0, 5) = '2015';

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

Out[21]:

| Month | Landing_Outcome | Booster_Version | Launch_Site |
|---|---|---|---|

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

- I filtered mission data between June 2010 and March 2017, grouped by Landing_Outcome, and ranked them in descending order. The most common outcome was “No attempt”, followed by equal counts of successes and failures on drone ships, highlighting the developmental stage of reusability during this period.



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.

In [22]:
```
%%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.
```

Out[22]:

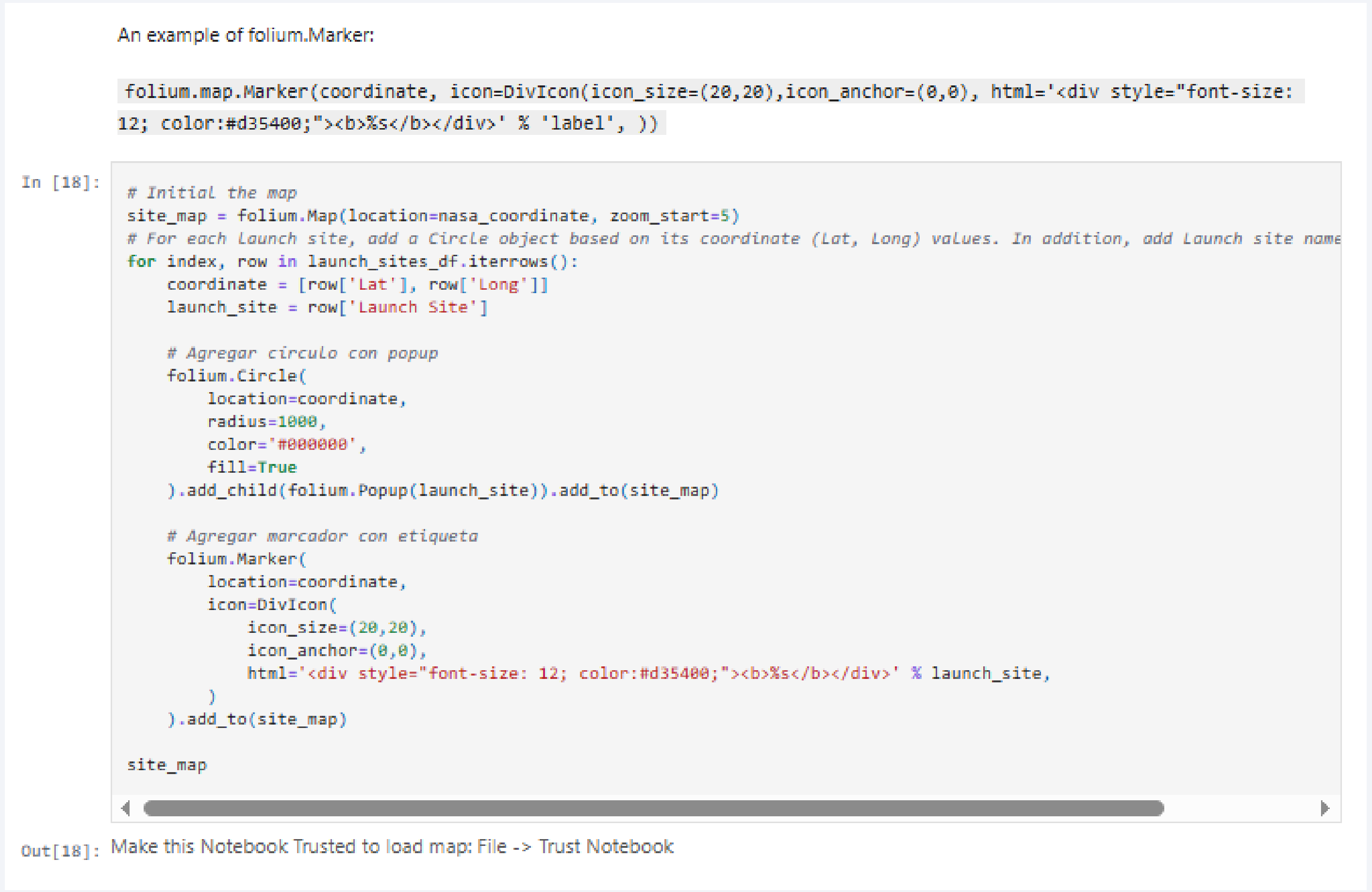
| Landing_Outcome | Outcome_Count |
|---|---|
| No attempt | 10 |
| Success (drone ship) | 5 |
| Failure (drone ship) | 5 |
| Success (ground pad) | 3 |
| Controlled (ocean) | 3 |
| Uncontrolled (ocean) | 2 |
| Failure (parachute) | 2 |
| Precluded (drone ship) | 1 |

Section 3

## Launch Sites Proximities Analysis

# <Launch Sites exploration>

- I couldn't show the maps because of a problem on the web extensions, it shows a message that says: Make this notebook Trusted to Load: File->Trust Notebook, but the online Lab doesn’t have that option.



An example of folium.Marker:

`folium.map.Marker(coordinate, icon=DivIcon(icon_size=(20,20),icon_anchor=(0,0), html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % 'label', ))`

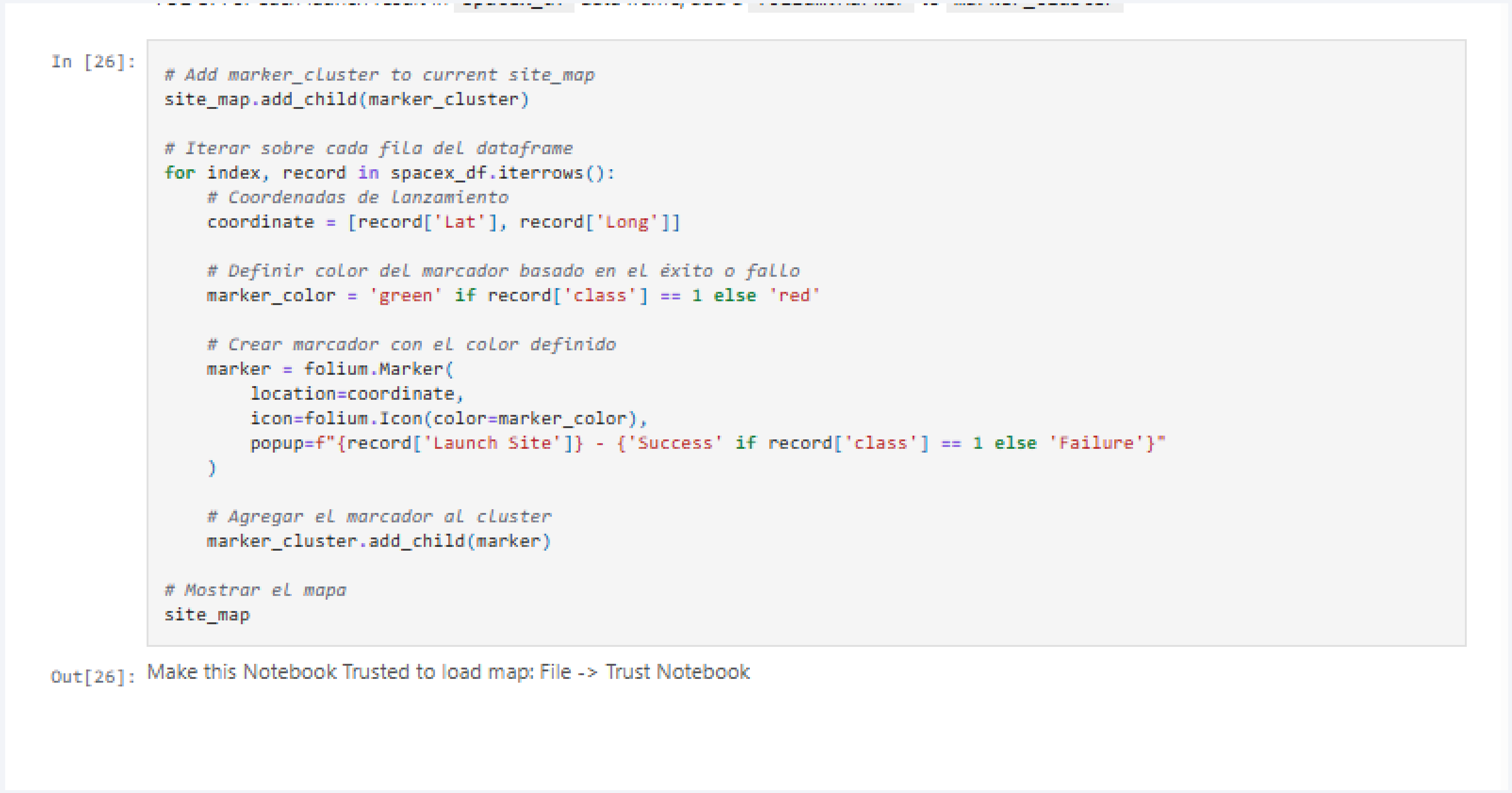
In [18]:

```
# Initial the map
site_map = folium.Map(location=nasa_coordinate, zoom_start=5)
# For each launch site, add a Circle object based on its coordinate (Lat, Long) values. In addition, add Launch site name
for index, row in launch_sites_df.iterrows():
    coordinate = [row['Lat'], row['Long']]
    launch_site = row['Launch Site']

    # Agregar circulo con popup
    folium.Circle(
        location=coordinate,
        radius=1000,
        color='#000000',
        fill=True
    ).add_child(folium.Popup(launch_site)).add_to(site_map)

    # Agregar marcador con etiqueta
    folium.Marker(
        location=coordinate,
        icon=DivIcon(
            icon_size=(20,20),
            icon_anchor=(0,0),
            html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % launch_site,
        )
    ).add_to(site_map)

site_map
```

Out[18]: Make this Notebook Trusted to load map: File -> Trust Notebook

## <Colored Launch site marks>

- I couldn't show the maps because of a problem on the web extensions, it shows a message that says: Make this notebook Trusted to Load: File->Trust Notebook, but the online Lab doesn’t have that option.

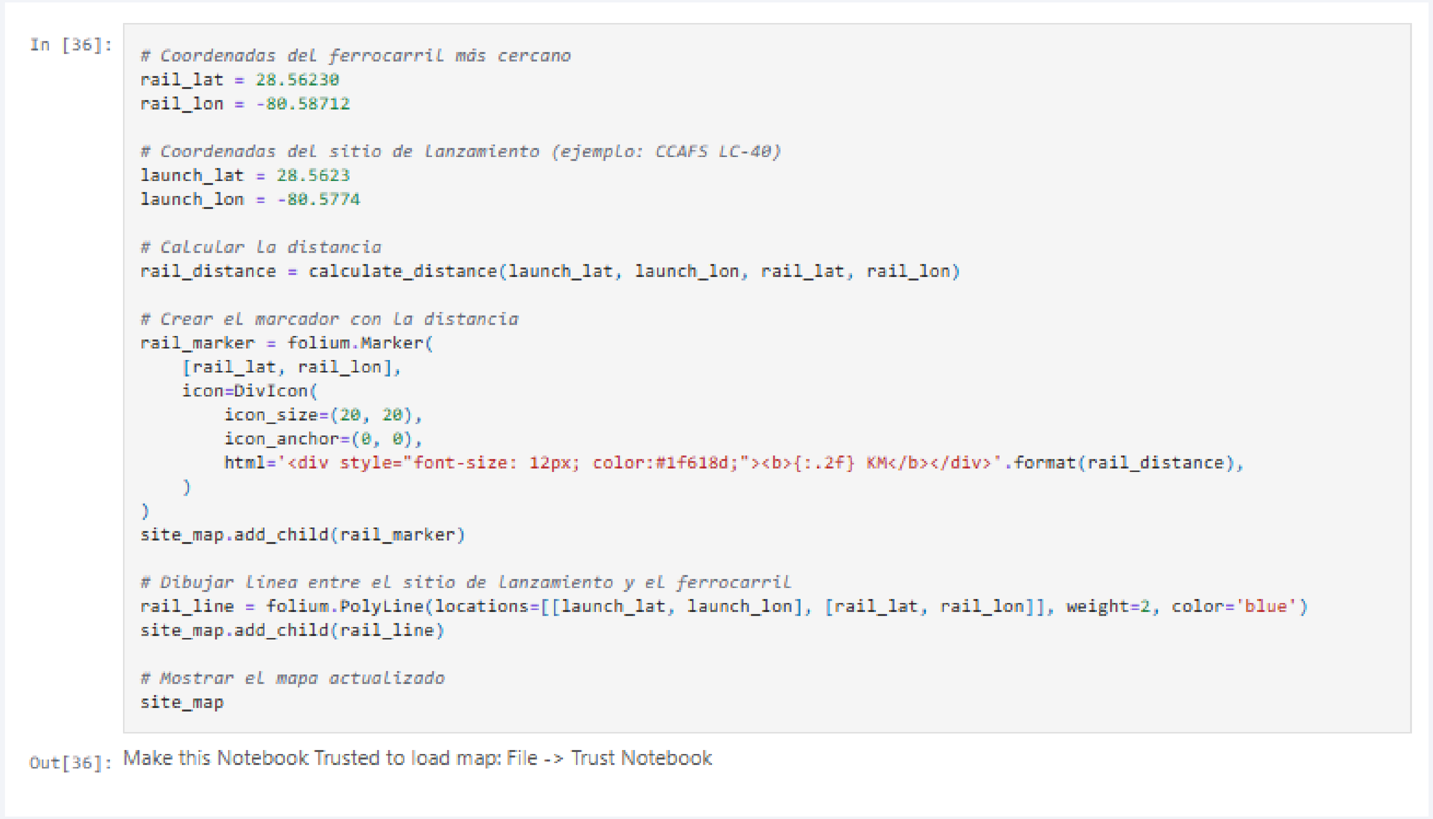


In [26]:
```python
# Add marker_cluster to current site_map
site_map.add_child(marker_cluster)

# Iterar sobre cada fila del dataframe
for index, record in spacex_df.iterrows():
    # Coordenadas de lanzamiento
    coordinate = [record['Lat'], record['Long']]

    # Definir color del marcador basado en el éxito o fallo
    marker_color = 'green' if record['class'] == 1 else 'red'

    # Crear marcador con el color definido
    marker = folium.Marker(
        location=coordinate,
        icon=folium.Icon(color=marker_color),
        popup=f"{record['Launch Site']} - {'Success' if record['class'] == 1 else 'Failure'}"
    )

    # Agregar el marcador al cluster
    marker_cluster.add_child(marker)

# Mostrar el mapa
site_map
```

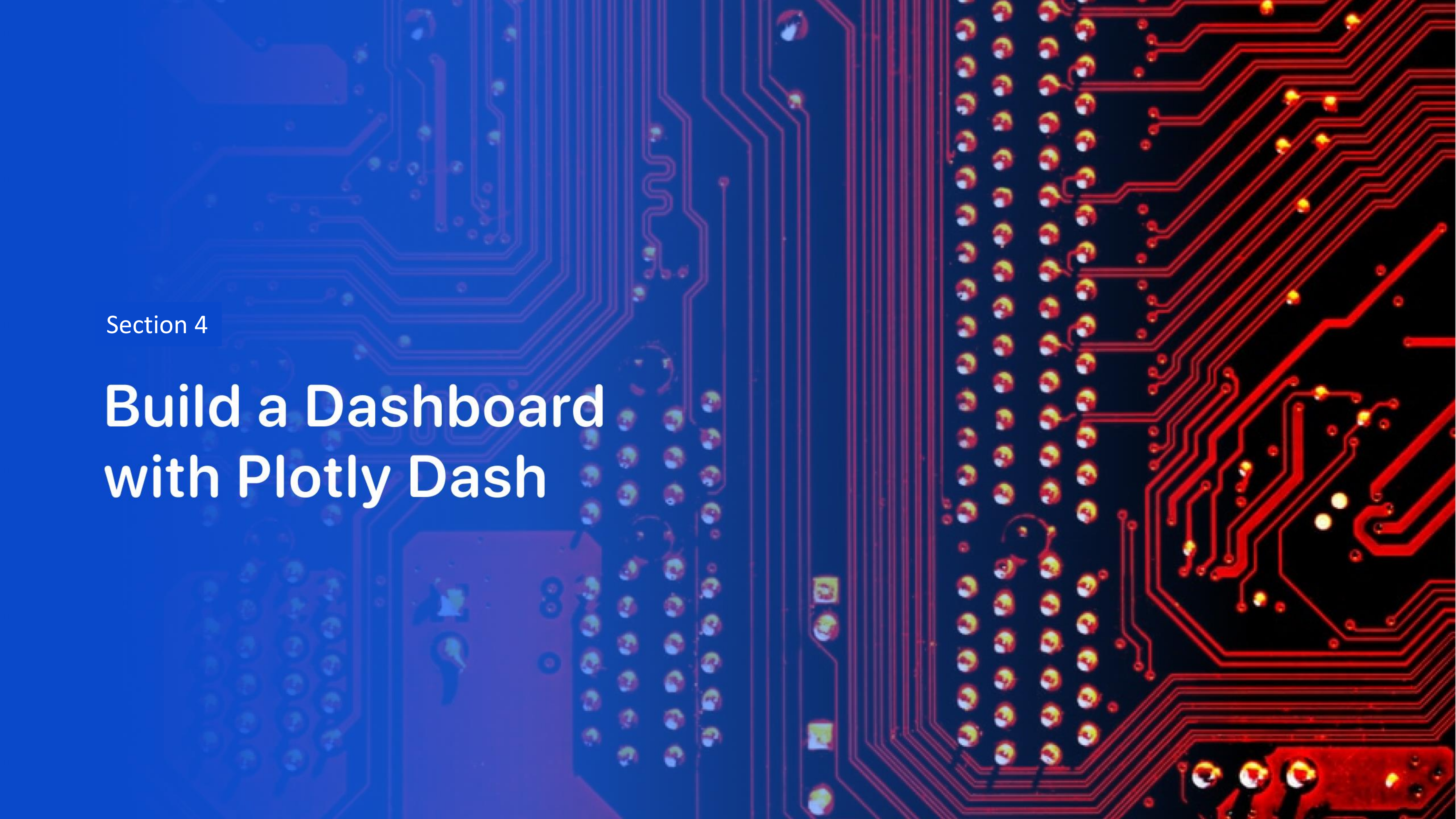
Out[26]:
```
Make this Notebook Trusted to load map: File -> Trust Notebook
```

# <Proximities interest points map>

- I couldn't show the maps because of a problem on the web extensions, it shows a message that says: Make this notebook Trusted to Load: File->Trust Notebook, but the online Lab doesn't have that option.



```
In [36]:
# Coordenadas del ferrocarril más cercano
rail_lat = 28.56230
rail_lon = -80.58712

# Coordenadas del sitio de lanzamiento (ejemplo: CCAFS LC-40)
launch_lat = 28.5623
launch_lon = -80.5774

# Calcular la distancia
rail_distance = calculate_distance(launch_lat, launch_lon, rail_lat, rail_lon)

# Crear el marcador con la distancia
rail_marker = folium.Marker(
    [rail_lat, rail_lon],
    icon=DivIcon(
        icon_size=(20, 20),
        icon_anchor=(0, 0),
        html='<div style="font-size: 12px; color:#1f618d;"><b>{:.2f} KM</b></div>'.format(rail_distance),
    )
)
site_map.add_child(rail_marker)

# Dibujar línea entre el sitio de lanzamiento y el ferrocarril
rail_line = folium.PolyLine(locations=[[launch_lat, launch_lon], [rail_lat, rail_lon]], weight=2, color='blue')
site_map.add_child(rail_line)

# Mostrar el mapa actualizado
site_map

Out[36]: Make this Notebook Trusted to load map: File -> Trust Notebook
```

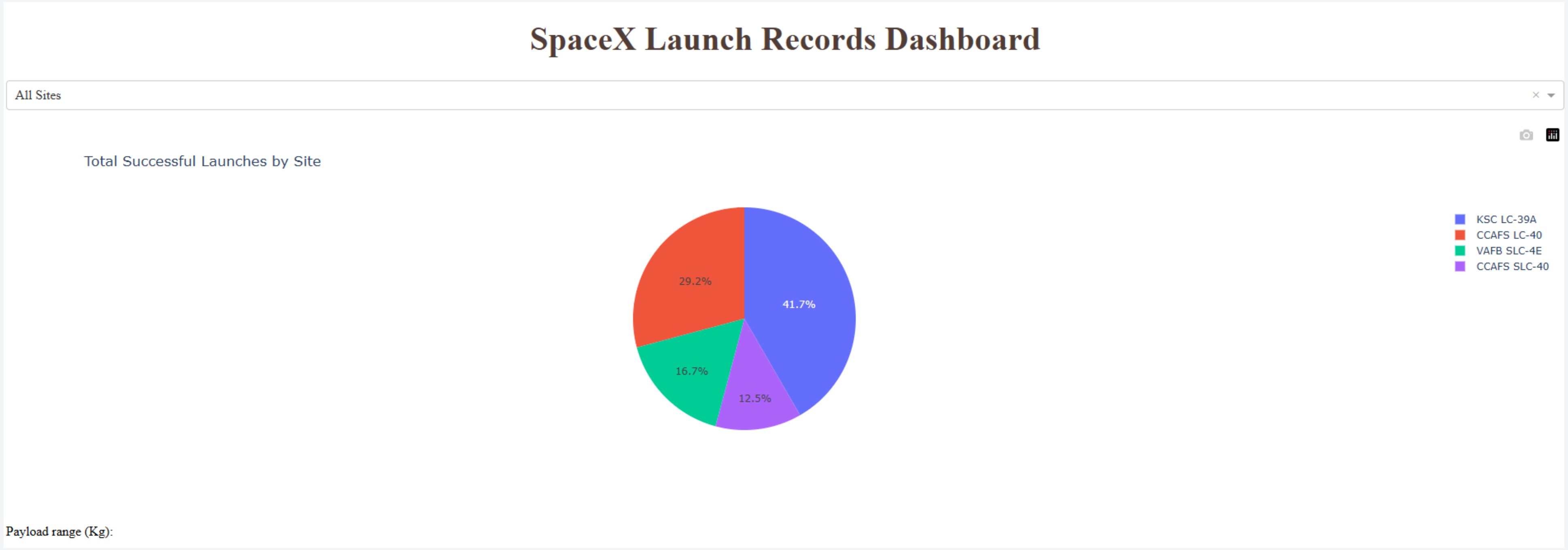


Section 4

## Build a Dashboard with Plotly Dash

# <SpaceX Launch Records Dashboard>

KSC LC-39A accounts for the highest success rate with 41.7% of successful launches. CCAFS LC-40 follows with 29.2%, while VAFB SLC-4E and CCAFS SLC-40 have comparatively fewer successful launches.This visualization helps identify which sites contributed most to SpaceX's successful missions.

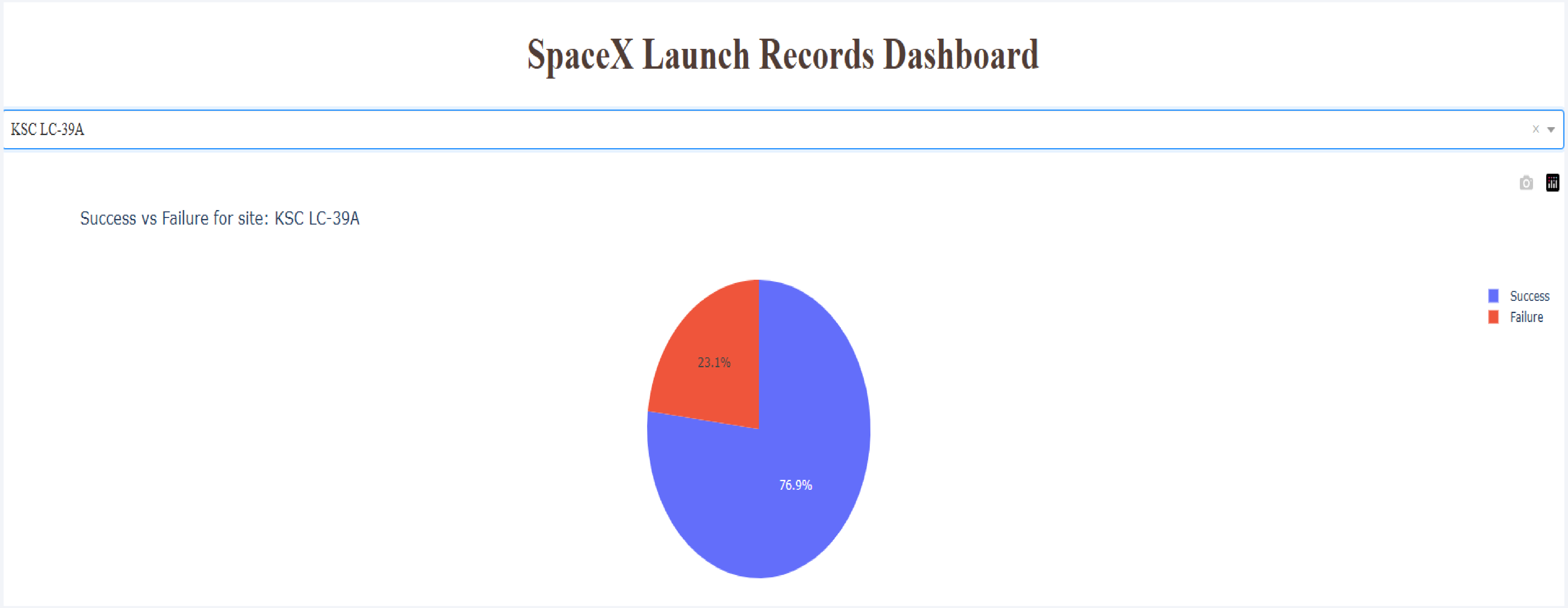


SpaceX Launch Records Dashboard

All Sites

Payload range (Kg):

## <Highest Launch success ratio>

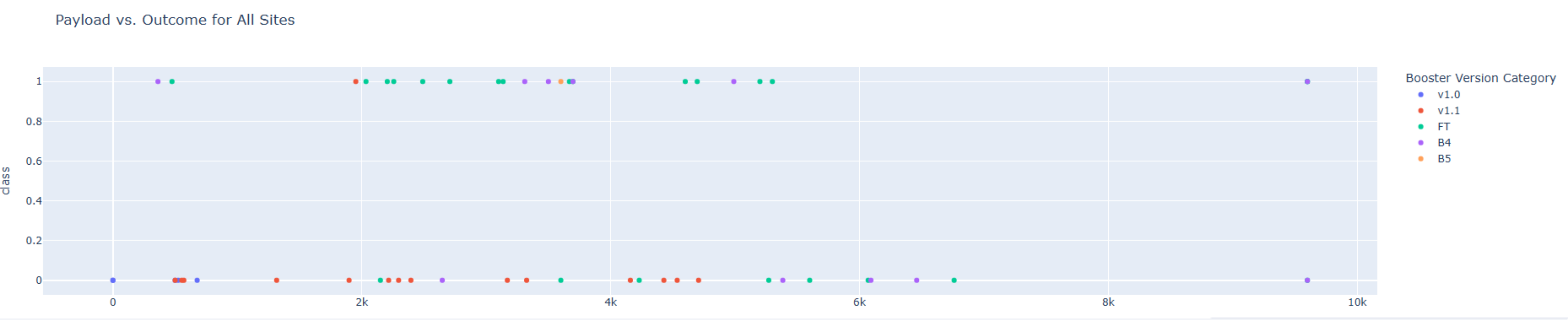


SpaceX Launch Records Dashboard

KSC LC-39A

Success vs Failure for site: KSC LC-39A

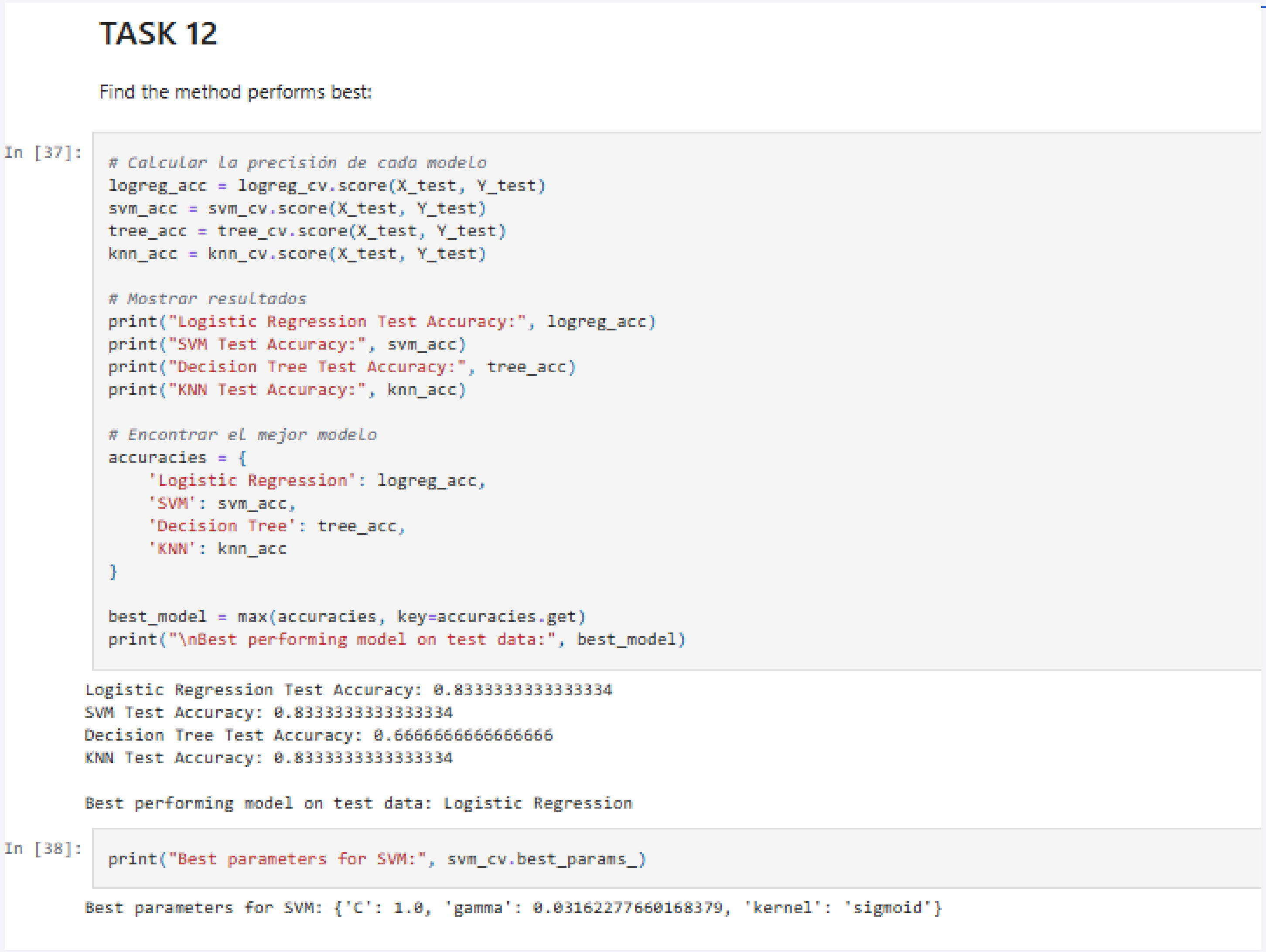
# <Payload vs. Outcome for all sites>


Payload vs. Outcome for All Sites
class
1
0.8
0.6
0.4
0.2
0
0
2k
4k
6k
8k
10k
Booster Version Category
v1.0
v1.1
FT
B4
B5

Section 5

## Predictive Analysis (Classification)

# Classification Accuracy



## TASK 12

Find the method performs best:

In [37]:

```
# Calcular la precisión de cada modelo
logreg_acc = logreg_cv.score(X_test, Y_test)
svm_acc = svm_cv.score(X_test, Y_test)
tree_acc = tree_cv.score(X_test, Y_test)
knn_acc = knn_cv.score(X_test, Y_test)

# Mostrar resultados
print("Logistic Regression Test Accuracy:", logreg_acc)
print("SVM Test Accuracy:", svm_acc)
print("Decision Tree Test Accuracy:", tree_acc)
print("KNN Test Accuracy:", knn_acc)

# Encontrar el mejor modelo
accuracies = {
    'Logistic Regression': logreg_acc,
    'SVM': svm_acc,
    'Decision Tree': tree_acc,
    'KNN': knn_acc
}

best_model = max(accuracies, key=accuracies.get)
print("\nBest performing model on test data:", best_model)
```

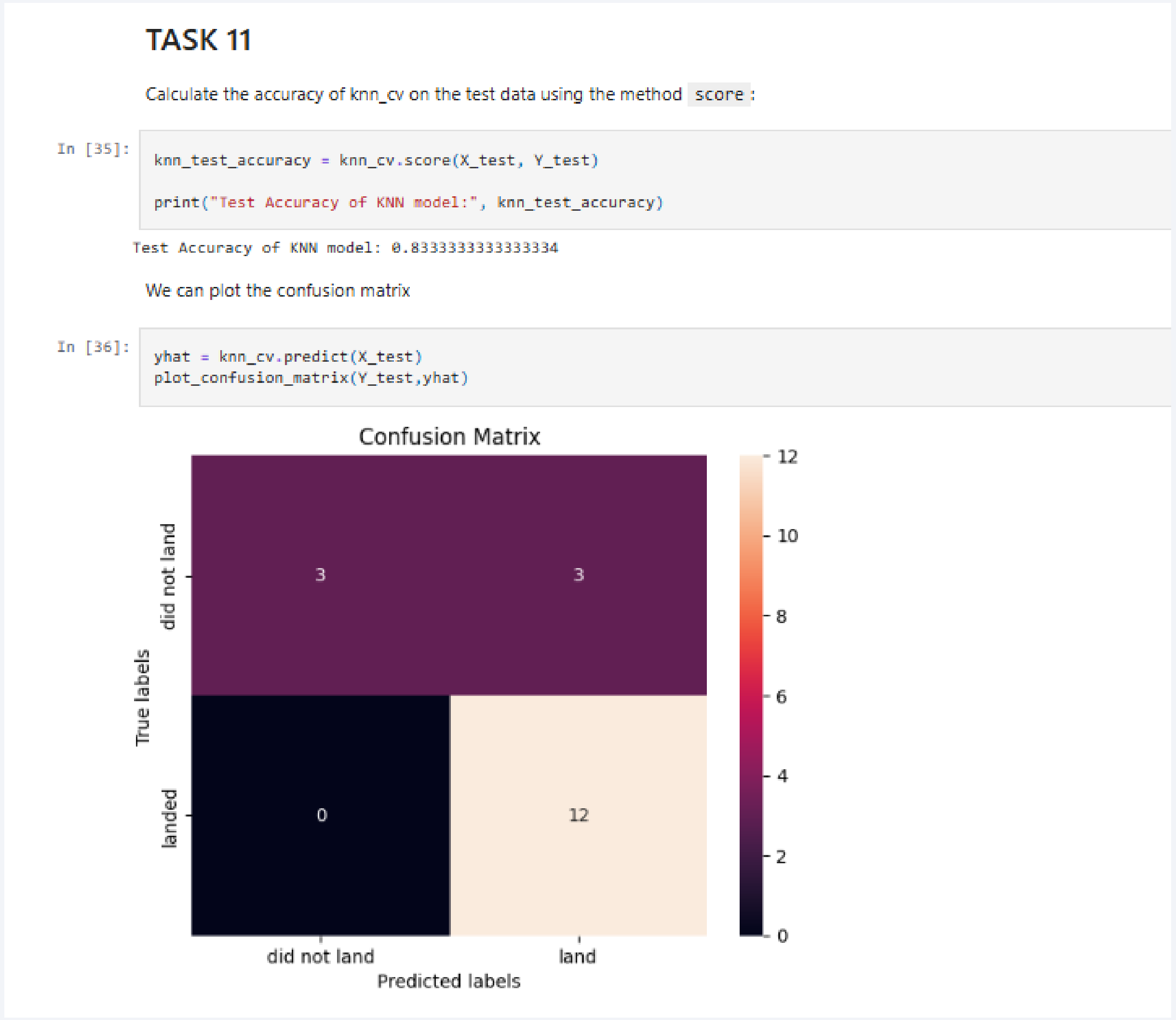
```
Logistic Regression Test Accuracy: 0.8333333333333334
SVM Test Accuracy: 0.8333333333333334
Decision Tree Test Accuracy: 0.6666666666666666
KNN Test Accuracy: 0.8333333333333334

Best performing model on test data: Logistic Regression
```

In [38]:

```
print("Best parameters for SVM:", svm_cv.best_params_)
```

```
Best parameters for SVM: {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
```

# Confusion Matrix



## TASK 11

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

In [35]:
```
knn_test_accuracy = knn_cv.score(X_test, Y_test)

print("Test Accuracy of KNN model:", knn_test_accuracy)
```

```
Test Accuracy of KNN model: 0.8333333333333334
```

We can plot the confusion matrix

In [36]:
```
yhat = knn_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```

Thank you!