IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Dian Novita
January 23, 2025

Outline

Executive Summary

- Summary of methodologies
    - Data Collection through SpaceX API and Web Scraping
    - Data Wrangling to create outcome variable
    - Exploratory Data Analysis (EDA) with Data Visualization
    - Exploratory Data Analysis (EDA) using SQL
    - Interactive Visual Analytics using Folium and Plotly Dash
    - Machine Learning prediction
- Summary of all results
    - Exploratory Data Analysis (EDA)
    - Analytics & Visualization
    - Predictive Analysis

Introduction

SpaceX is a private aerospace manufacturer and space transportation services company. They have reached notable achievements in space exploration. A crucial factor in their success is the relatively low cost of rocket launches, particularly with their reusable Falcon 9 rocket. If we can ascertain the reusability of the first stage, we can also establish the launch price and determine the optimal conditions necessary to enhance the probability of successful landings.

This capstone project seeks to explore:

- The relationships between payload mass, launch site, number of flights, and orbital characteristics with first-stage landing success,
- Rate of successful landings over time, and
- Best Predictive model for a successful landing.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data collected Using the SpaceX API and web scraping to gather invaluable data efficiently and effectively.
- Perform data wrangling
  - The categorical features were processed using one-hot encoding.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Build classification models using libraries like scikit-learn, tune hyperparameters, and evaluate performance using metrics like or through cross-validation and evaluate classification models.

Data Collection

Data collection involves systematically gathering and measuring information about specific aspects within a defined system. As noted, the dataset was compiled using API requests from SpaceX REST and web scraping techniques from SpaceX Wikipedia. This method aims for clarity and conciseness while maintaining the original meaning.

- Define data requirements
- API integration
- Data extraction, cleaning and transforming
- Web scraping
- Data extraction, cleaning and transforming

Data Collection – SpaceX API

```
response=requests.get(static_json_url).json()
data=pd.json_normalize(response)
```

- `requests.get(static_json_url)`: This function fetches the data from the URL stored in the `static_json_url` variable.
- `.json()`: Parses the fetched data in JSON format and converts it into a Python dictionary or list.
- `.json_normalize()`: The data in the previous line gets "flattened" or normalises into a table-like structure

Request rocket launch data from SpaceX API

Decode response with `.json()` and convert to a dataframe using `.json_normalize()`

Request information about launches from API using custom functions

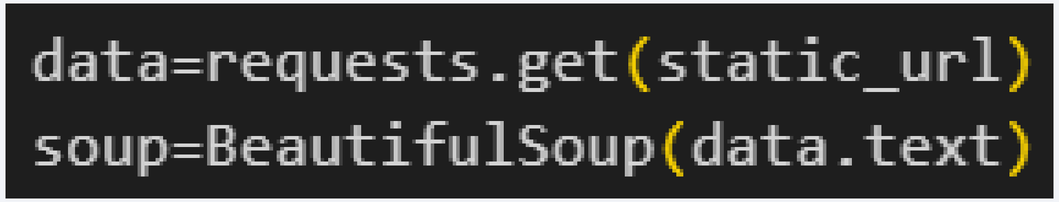
Create dictionary from rocket launch data

Create dataframe from dictionary

Clean the data and fill the missing values

`/ API data collection code`

Data Collection - Scraping



```
data=requests.get(static_url)
soup=BeautifulSoup(data.text)
```

- `requests.get(static_url)`: This function fetches the data from the URL stored in the `static_url` variable.
- `BeautifulSoup()`: This function creates an object which represents the parsed HTML content of the webpage

Request Falcon 9 launch data from Wikipedia

Create BeautifulSoup object from HTML response

Extract column names from HTML table header

Extract data from parsing HTML table

Create dictionary from Falcon 9 launch data

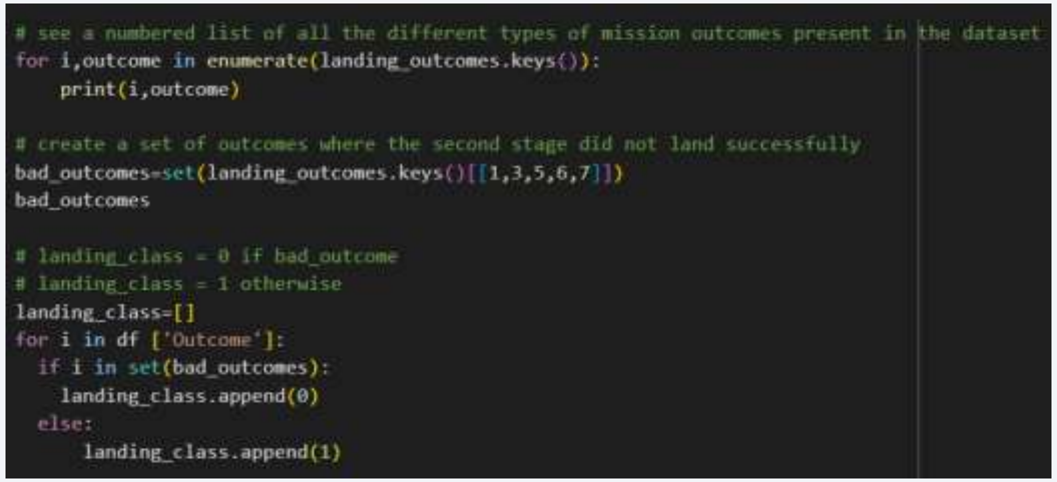
Create dataframe from dictionary

/ Web scraping data collection code

Data Wrangling

```
df['LaunchSite'].value_counts()
df['Orbit'].value_counts()
df['Outcome'].value_counts()
```

- `.value_counts()`: counts how many times each unique appears in a certain column.



```
# see a numbered list of all the different types of mission outcomes present in the dataset
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)

# create a set of outcomes where the second stage did not land successfully
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes

# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class=[]
for i in df ['Outcome']:
  if i in set(bad_outcomes):
    landing_class.append(0)
  else:
      landing_class.append(1)
```

- This code takes a column containing detailed landing outcomes and simplifies it into a binary classification label: `1` for successful landings and `0` for unsuccessful landings. This simplified `landing_class` label can then be used to train machine learning models to predict landing success

Perform EDA and determine training labels

Calculate:
Number of launches on each site
Number and occurance of orbit
Number and occurance of mission outcome per orbit

Set up a binary landing outcome

/ Data wrangling code

EDA with Data Visualization

- Scatter plot: displays the relationship between two variables by plotting data points as dots on a graph. We need to examine the relationship between Flight Number vs. Launch Site, Payload vs. Launch Site, Flight Number vs. Orbit Type, and Payload vs. Orbit Type.
- Bar chart: the rectangular bars represent data, where the length of each bar is proportional to the value it represents. This bar is Ideal for identifying which orbits have the highest probability of success.
- Line chart: A line chart connects data points with lines to show trends and changes over time. This kind of chart is suitable for identifying yearly launch success trends and patterns.

/ EDA with data visualization code

EDA with SQL

- `distinct:` This query will return a list of unique values in a column, removing any duplicates.
- `like '…%':` Used for pattern matching in string comparisons. For example, `like 'CCA%'`, the query will select data which starts with the letter `'CCA%'`.
- `limit:` Used to restrict the number of rows returned in a query result
- `where:` Filters the rows in a table based on a specified condition.
- `sum():` Calculates the sum of a column's values.
- `avg():` Calculates the average value of a column.
- `min():` Finds the minimum value in a column.
- `between … and …:` Selects values within a specified range. For example, `between 2000 and 6000`, the query will retrieve values in specific columns only within `2000-6000.`
- `count():` Counts the number of rows or non-NULL values in a column.
- `group by:` Groups rows that have the same values in specified columns. Often used with aggregate functions (`sum`, `avg`, `count`, etc.).
- `substr(Date,…,…):` Extracts a substring from a string. In this case, the data date is stored in string format, to extract it we need this query.
- `order by:` Sorts the result set in ascending or descending order.
- `Subquery:` A nested SELECT statement within another SQL statement.

/ EDA SQL code

Build an Interactive Map with Folium

- Markers: `folium.Marker` objects were used, specifying the coordinates (latitude and longitude of each launch) and icon properties. The markers can represent individual launch records with different colours indicating success (The '1' class in green icons) or failure (The '0' class in red icons).
- Circles: `folium.Circle` objects were used, specifying the centre coordinates (latitude and longitude of each launch site. In this case, the circle highlights the launch site locations.
- Mouse position: `folium.plugins.MousePosition` was added to display the latitude and longitude of the mouse cursor position on the top-right corner of the map. It helps us to easily find the coordinates of any point on the map by hovering the mouse over it.
- Distance markers: `folium.Marker` objects were used, with custom icons containing the distance value. These markers were placed at the proximity points. This marker will display the calculated distance between a launch site and its proximities.
- Line: `folium.PolyLine` objects were used, specifying the coordinates of the launch site and the proximity point. Used to visualize distances between a launch site and its proximities (coastline, highway, railroad, city).

`/ Interactive visualization with folium code`

Build a Dashboard with Plotly Dash

- Dropdown: A `dcc.Dropdown` is added for a specific category, with options dynamically generated from your data. Allows users to select a specific site selection from unique launch sites in a data frame
- Pie charts: `px.pie()` used to visualise the distribution of values within the selected category. It is used to display the total success launched by sites.
- Slider: A `dcc.RangeSlider` enables us to filter data based on a value range. This included filtering payload mass.
- Scatter plot: `px.scatter()` used to show the relationship between two for the selected category and value range. The scatter plot used to know the correlation between payload and success.
- Callback: The `@app.callback` functions will update the charts dynamically when the dropdown or slider values change. The function decorator links the dropdown and slider as inputs to the pie chart and scatter plot as outputs

/ Interactive visualization with Plotly Dash code

Predictive Analysis (Classification)

- Data Preparation.
- Model Selection and Hyperparameter Tuning. Four classification models were considered: Logistic Regression, Support Vector Machine (SVM), Decision Tree, and K-Nearest Neighbours (KNN).
- Model Evaluation. The performance of each model was evaluated using the following metrics: Accuracy, Jaccard score, and F1 score.
- Model Comparison and Selection.

Build the model: Load and transform the data. Split into training and test datasets. Set up the parameters and algorithms

Evaluate the model: check the accuracy and set up the hyperparameters

Improve the model using Feature Engineering and Algorithm Tuning

Find the best model with the great accuracy score

Results

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

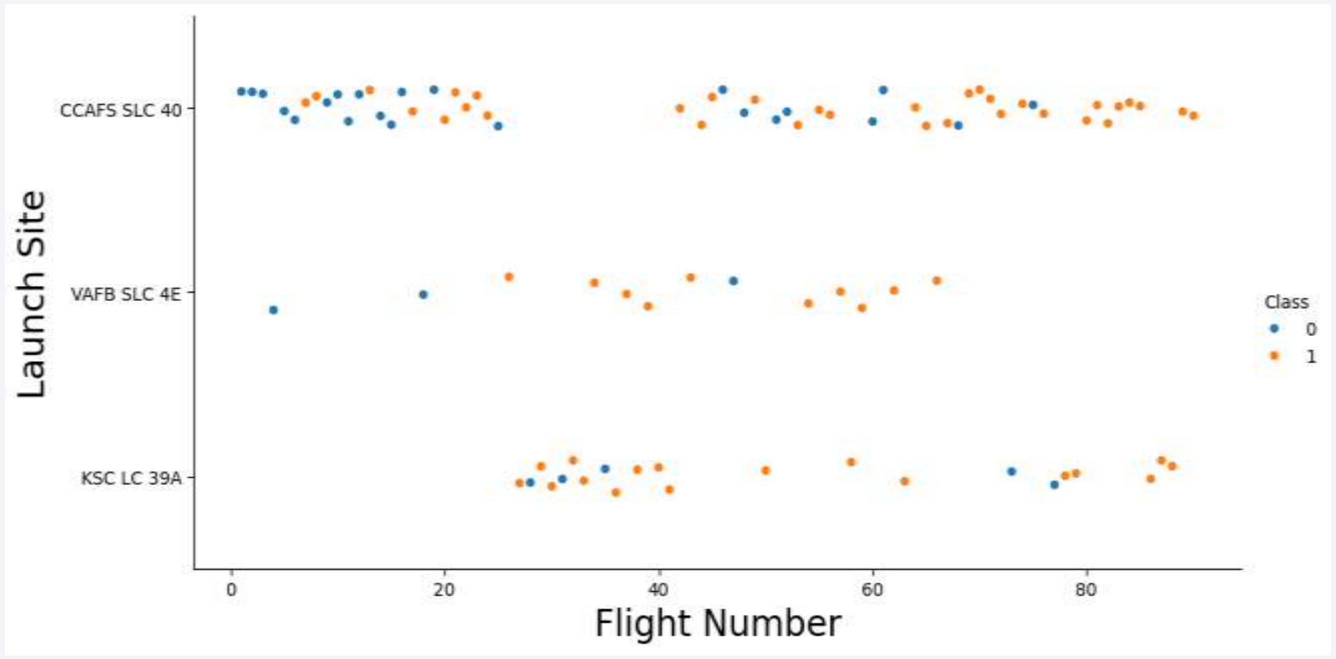
Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

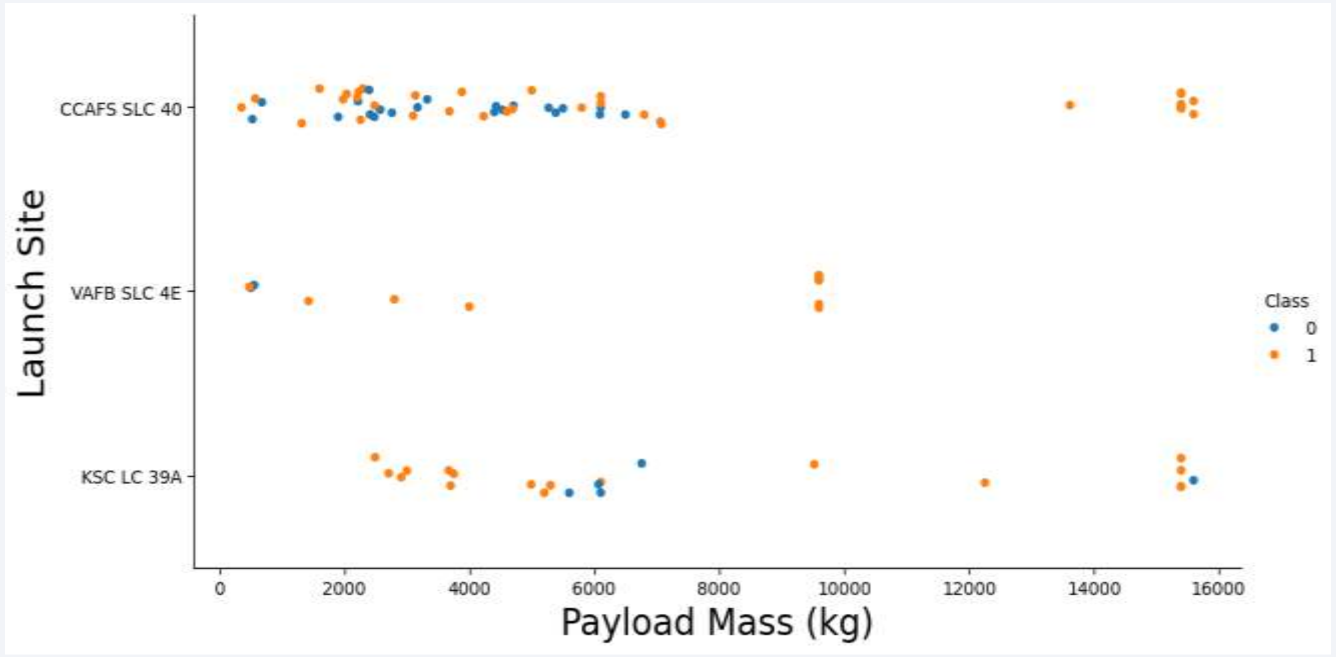
The CCAFS SLC 40 launch site has the highest number of launches, but there are a lot of failed launches (Class '0' is a failed in blue dots). However other launch sites, VAFB SLC 4E and KSC LC 39A have the least number of launches but have a higher success rate (Class '1' is a success in orange dots). The number of flights in the sites does not influence the success rate.

Earlier launches had many failures, but by the time the launches had a higher success rate. We can infer that the new launches have a higher chance of success.



Payload vs. Launch Site

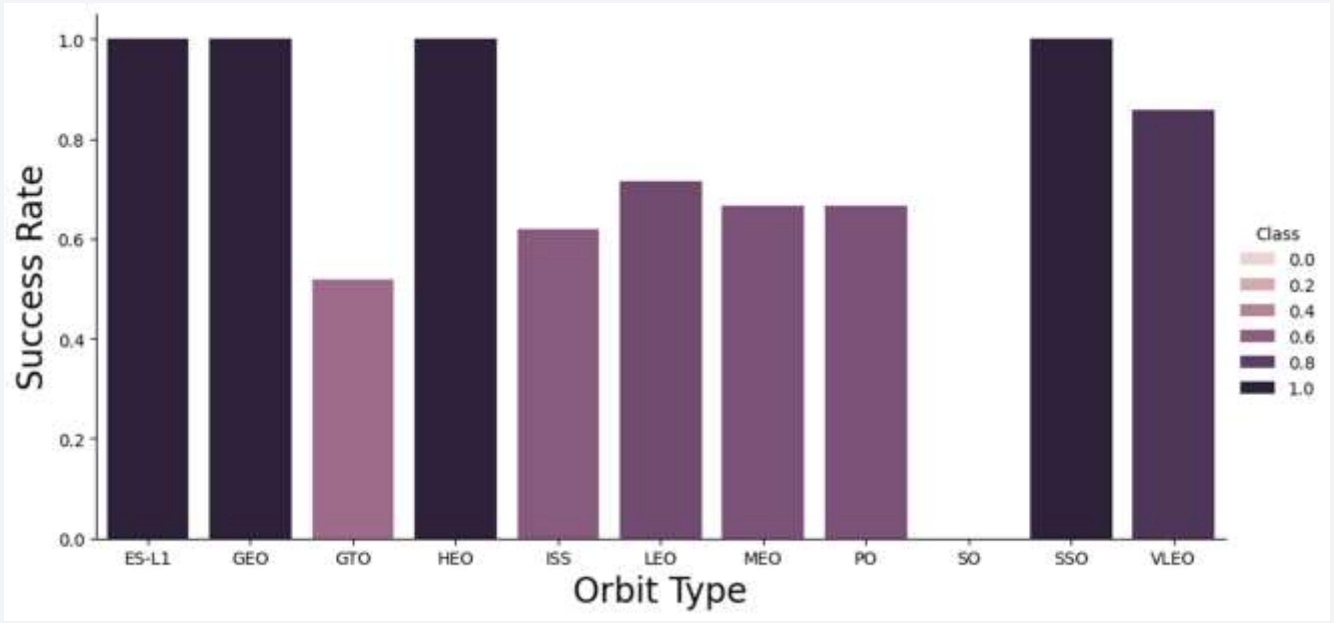
Looking at specific launch sites, KSC LC 39A boasts a perfect track record for launches carrying payloads under 5,500 kg. On the other hand, VAFB SKC 4E has not launched anything exceeding approximately 10,000 kg. Furthermore, the majority of launches carrying payloads heavier than 7,000 kg have been successful.



Success Rate vs. Orbit Type

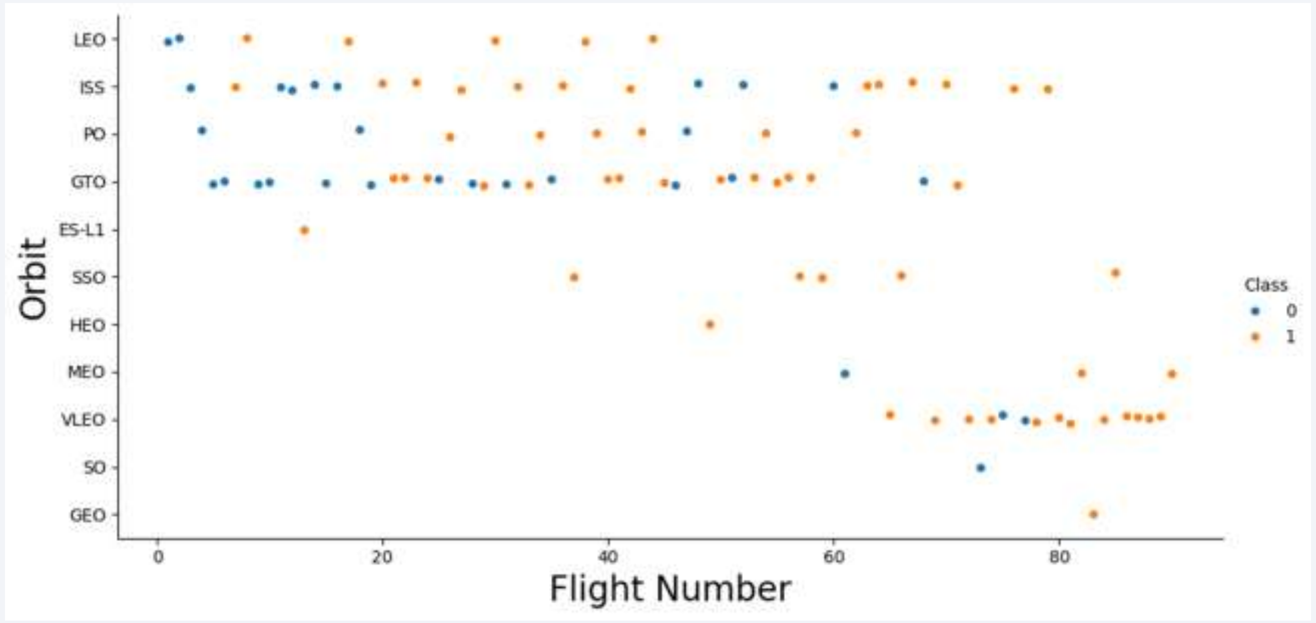
This launch provider demonstrates a 100% success rate for missions to ES L1, GEO, HEO and SSO. GTO, ISS, LEO, MEO, and PO orbits exhibit a success rate ranging from 50% to 80%. Notably, no successful launches have been achieved for SO.

Further analysis reveals that certain orbits, such as GEO, SO, HEO, and ES L1, have only been attempted once. With such limited data for these orbits, it is challenging to discern any meaningful patterns or trends regarding landing success rates. More extensive datasets are required to draw reliable conclusions for these specific orbital destinations.



Flight Number vs. Orbit Type

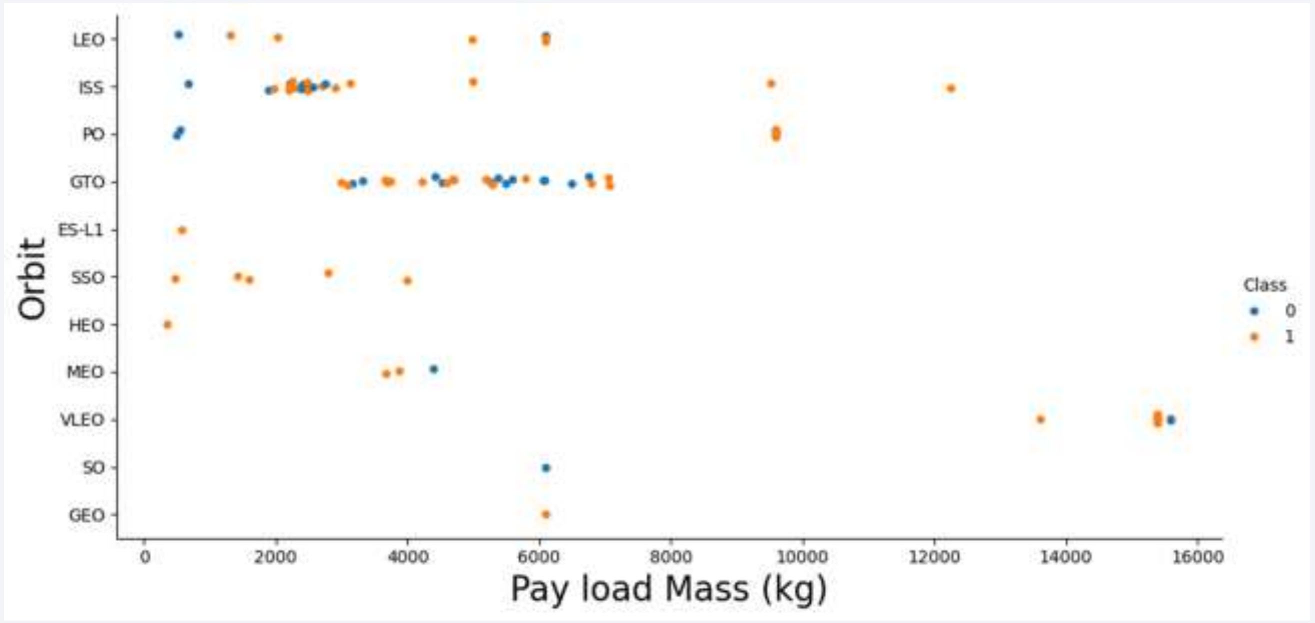
This scatter plot reveals that as the number of flights increases for a particular orbit, the success rate tends to improve, particularly for LEO missions. However, this trend is not observed for GTO missions, where no apparent relationship exists between the number of flights and the success rate.



Payload vs. Orbit Type

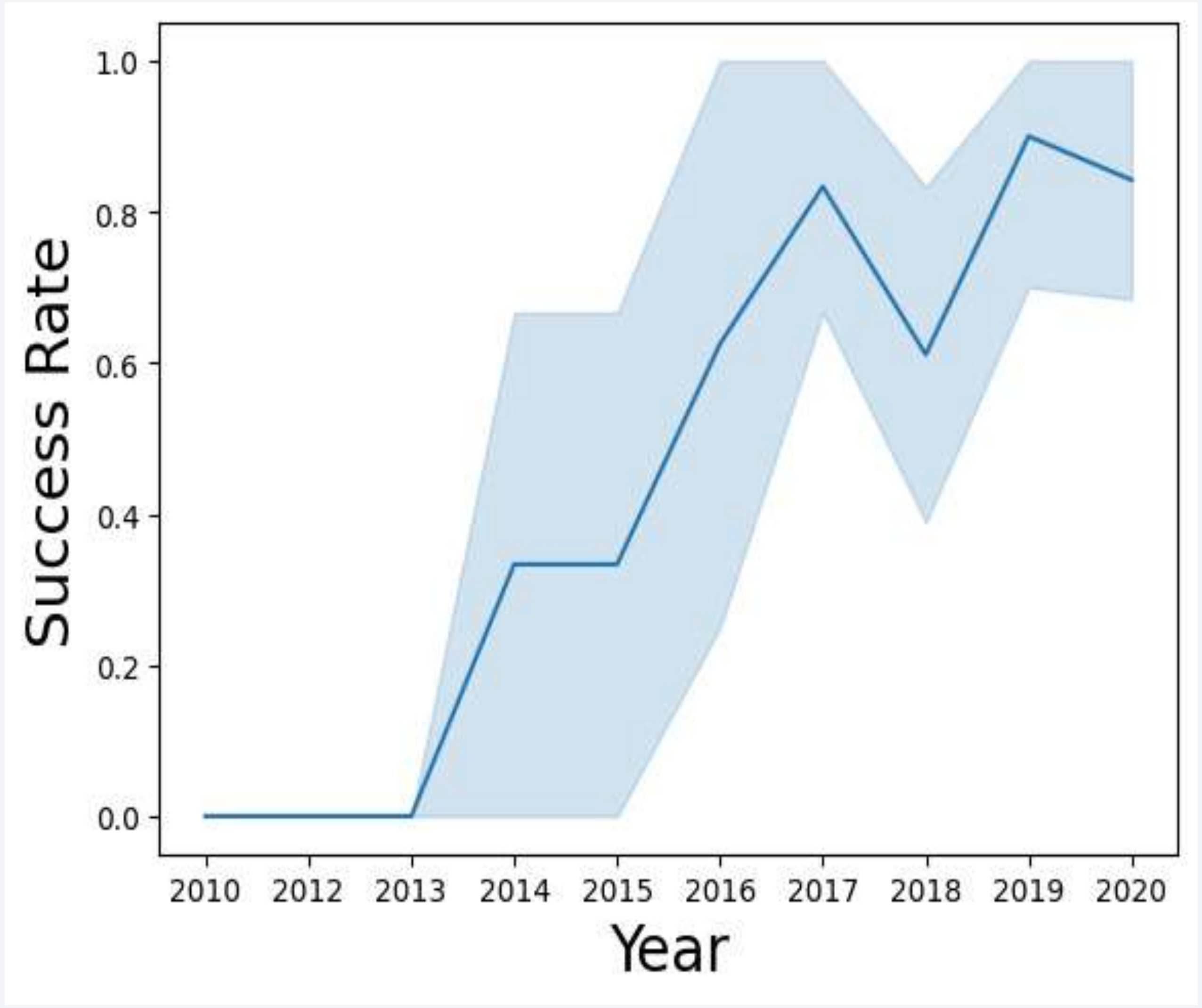
Heavier payloads appear to have a beneficial influence on the success of missions to LEO, ISS, and PO. In these orbital regimes, launching larger payloads seems to correlate with higher success rates.

Conversely, heavier payloads appear to negatively impact the success of missions to MEO and VLEO. In these cases, increasing the payload mass may lead to a decrease in mission success. Notably, for GTO missions, the relationship between payload mass and success rate is less clear, with no discernible pattern emerging from the data.



Launch Success Yearly Trend

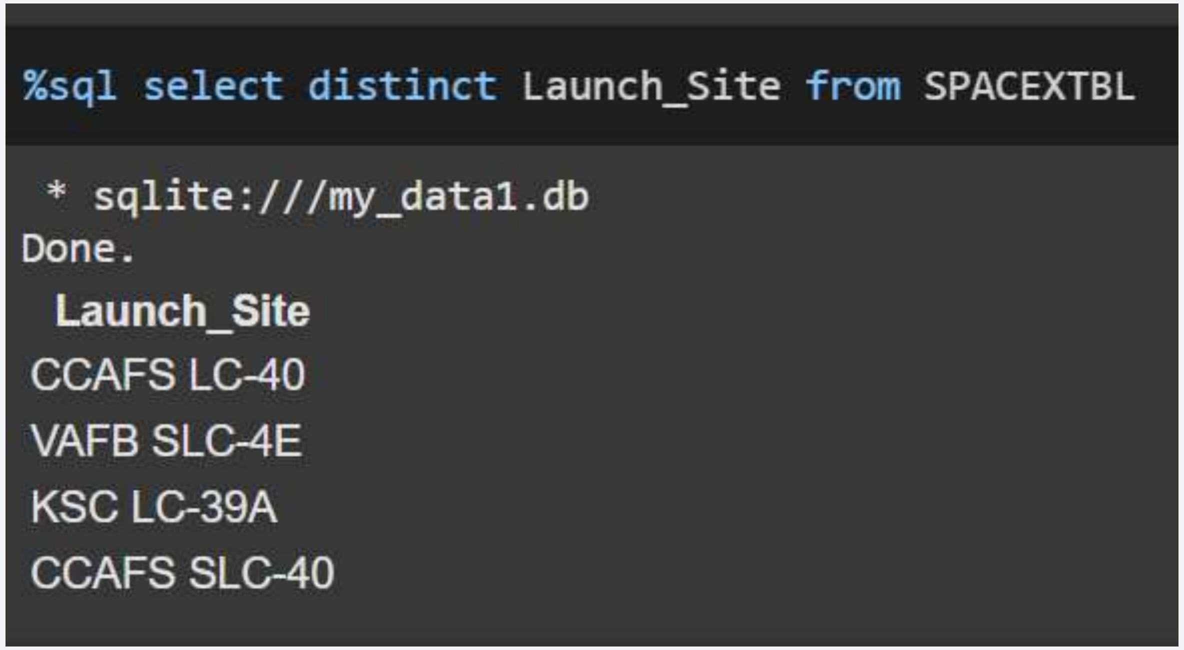
These figures indicate a rising trend from 2013 to 2020. If this trajectory continues, we can expect the success rate to increase and ultimately reach a 100% success rate steadily.



All Launch Site Names

Find the names of the unique launch sites

In the query, we use the `distinct` to show the unique value and get the following launch sites. It gets four `Launch_Site`

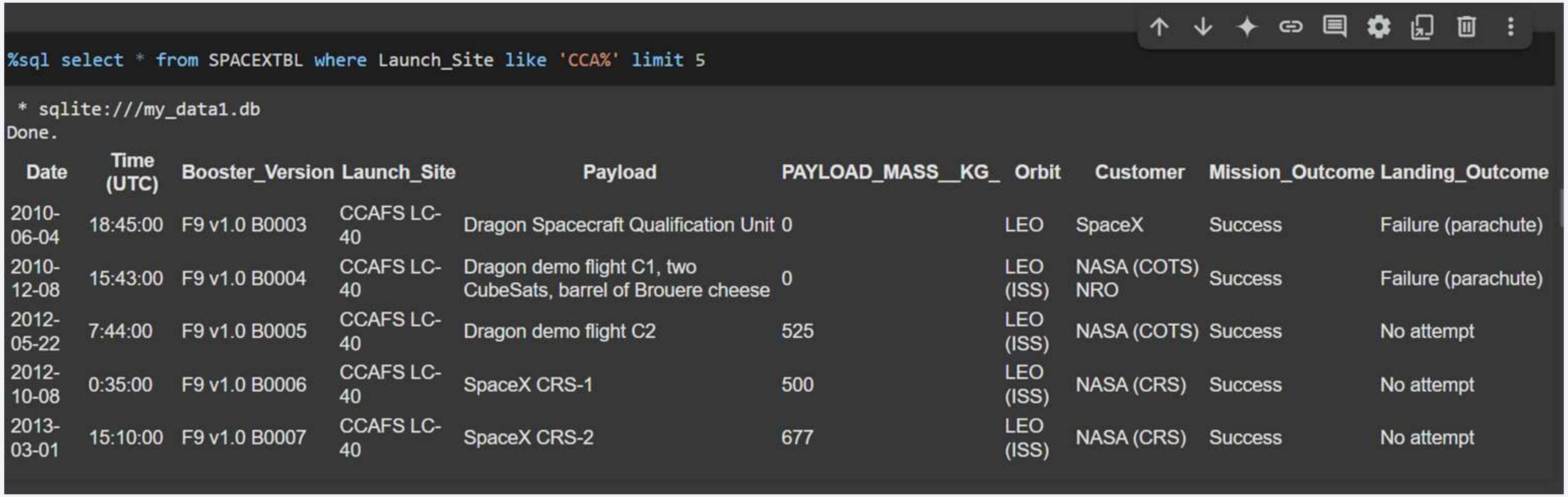


```
%sql select distinct Launch_Site from SPACEXTBL

 * 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'

Find 5 records where launch sites begin with `CCA`

Use a `like ‘CCA%’` and `limit` query to retrieve the data. Here are the results of the records.



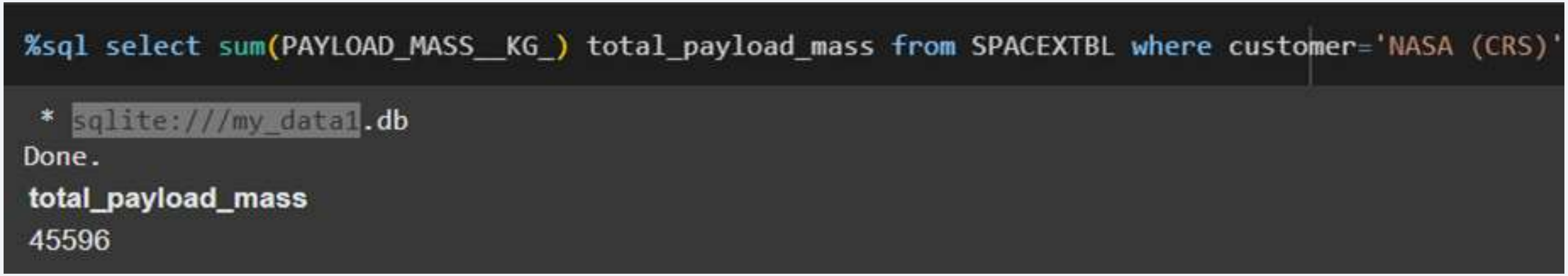
```
%sql select * from SPACEXTBL where Launch_Site like 'CCA%' limit 5

 * 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

Calculate the total payload carried by boosters from NASA

In the query, we use the `sum` to calculate the total and `where` to filter the data only shows the `'Nasa (CSR)'`. From the calculation, we get `45596` total payload mass launched by NASA (CSR).



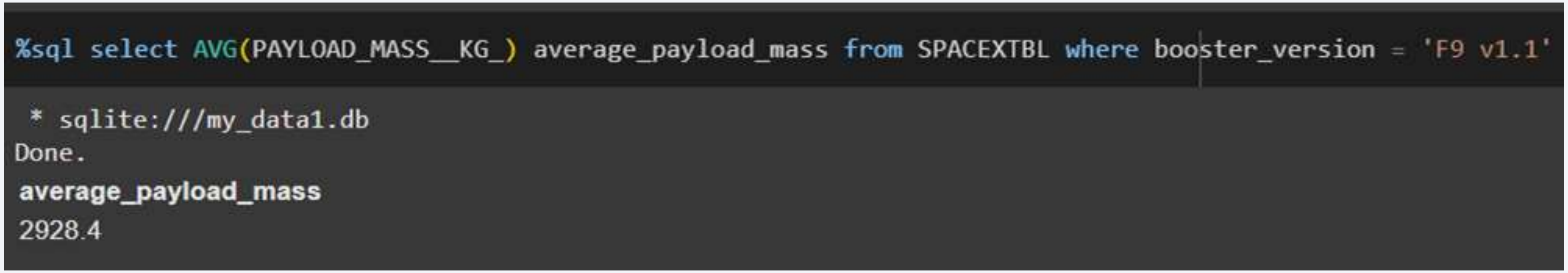
```
%sql select sum(PAYLOAD_MASS__KG_) total_payload_mass from SPACEXTBL where customer='NASA (CRS)'
```

```
 * sqlite:///my_data1.db
Done.
total_payload_mass
45596
```

Average Payload Mass by F9 v1.1

Calculate the average payload mass carried by booster version F9 v1.1

Calculate the average value with the `avg` and `where` query to filter the data only shows the ‘`F9 v1.1`’ booster version. We get `2928,4` as an average value.

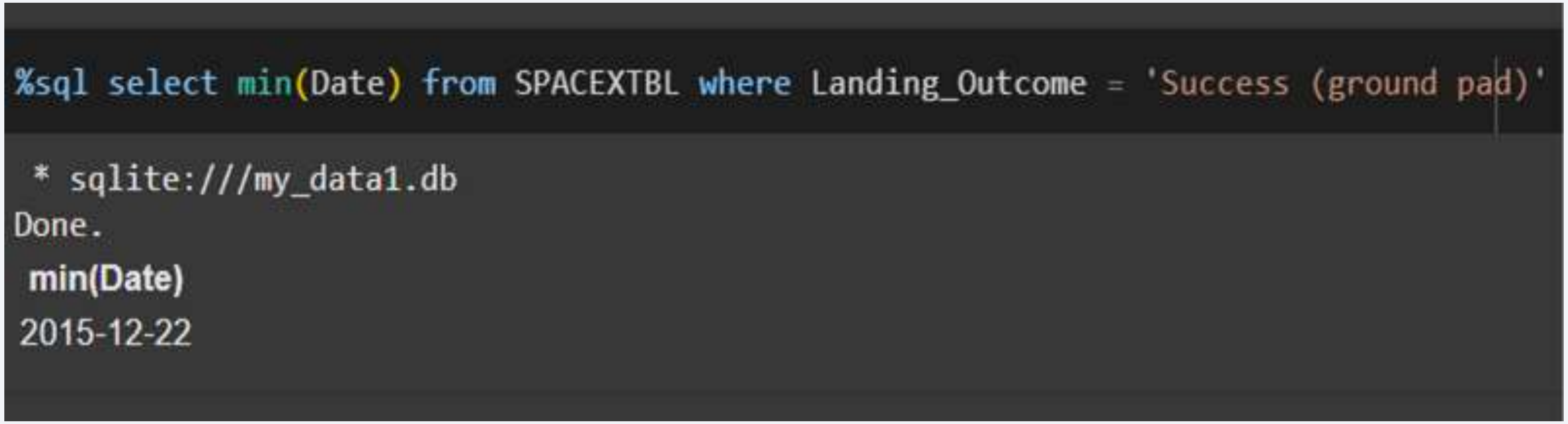


```
%sql select AVG(PAYLOAD_MASS__KG_) average_payload_mass from SPACEXTBL where booster_version = 'F9 v1.1'

 * sqlite:///my_data1.db
Done.
average_payload_mass
2928.4
```

First Successful Ground Landing Date

Find the dates of the first successful landing outcome on the ground pad

the `min` used to retrieve the oldest date and where to filter the data only shows the `‘Success (ground pad)’`. The result shows that the very first successful landing on the ground pad is December 22, 2015.

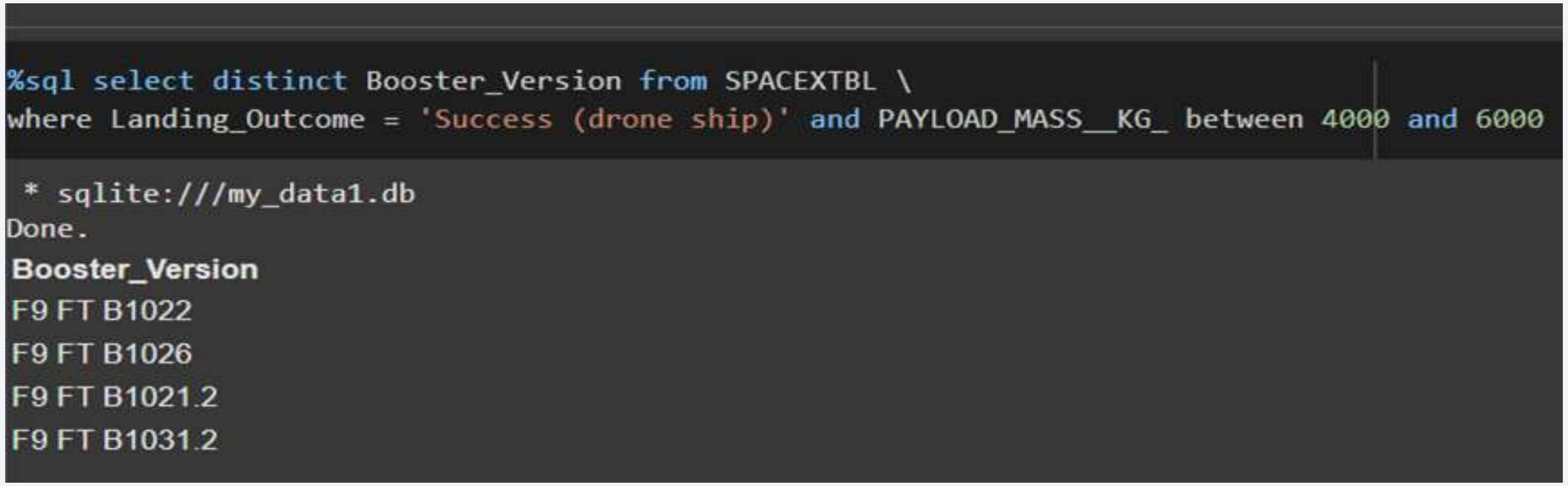


```
%sql select min(Date) from SPACEXTBL where Landing_Outcome = 'Success (ground pad)'

 * sqlite:///my_data1.db
Done.
 min(Date)
 2015-12-22
```

Successful Drone Ship Landing with Payload between 4000 and 6000

List the names of boosters which have successfully landed on drone ships and had payload mass greater than 4000 but less than 6000

The `distinct` used to get a list of unique values and `where` to filter the data only shows the exact condition. In this case, we need `between … and …` to filter `payload_mass_kg between 4000 and 6000`. Or we can use the math operation like this `payload_mass_kg > = 4000 < = 6000`

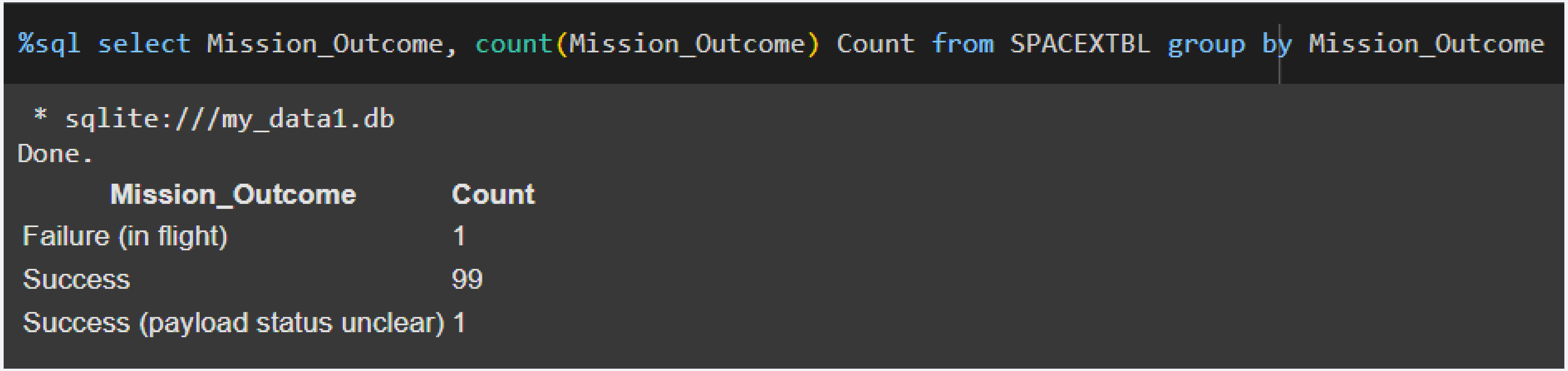


```
%sql select distinct Booster_Version from SPACEXTBL \
where Landing_Outcome = 'Success (drone ship)' and PAYLOAD_MASS__KG_ between 4000 and 6000

 * 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

Calculate the total number of successful and failed mission outcomes

The `count` query to count the number of entries. Then add `group by` query to divide the result-set into groups based on the `mission_outcome`. So, we get these results, mostly it is a successful mission (99 counts).



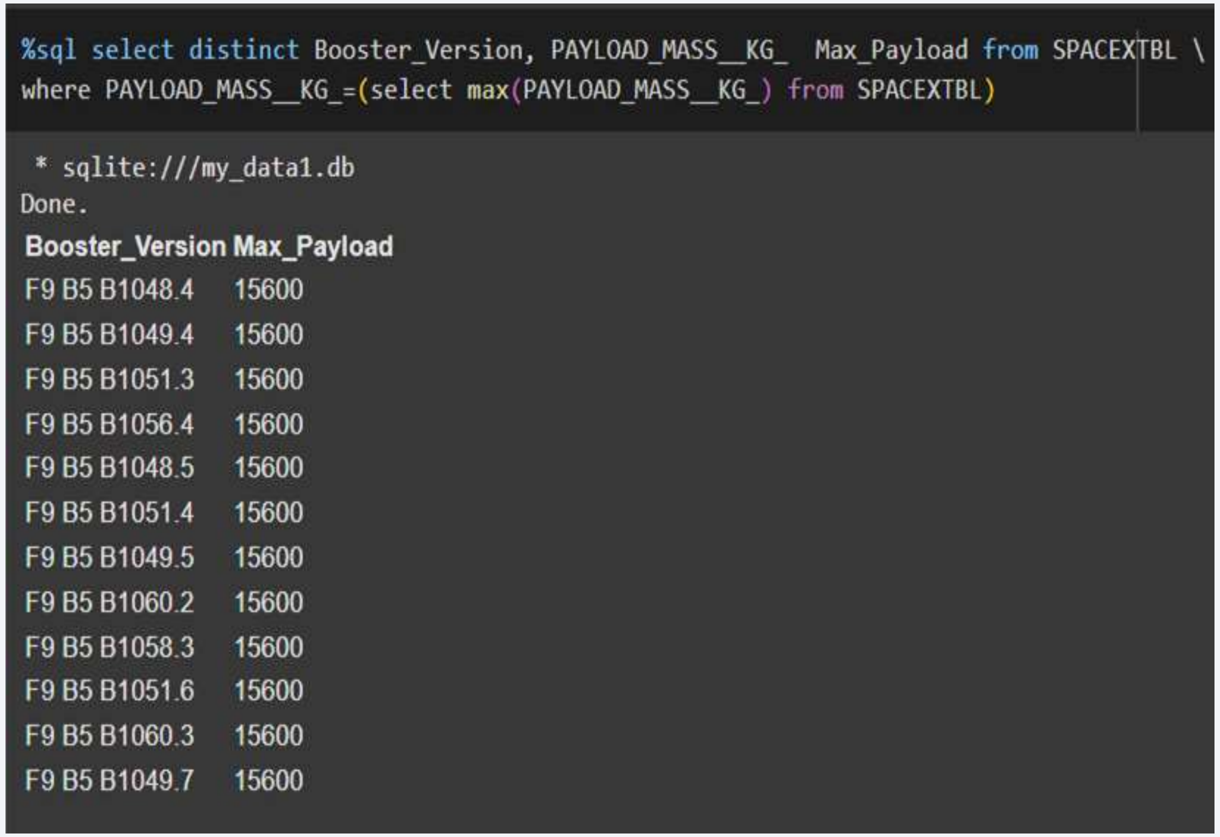
```
%sql select Mission_Outcome, count(Mission_Outcome) Count from SPACEXTBL group by Mission_Outcome

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

Mission_Outcome	Count
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

Boosters Carried Maximum Payload

List the names of the boosters which have carried the maximum payload mass

It finds the maximum payload mass `payload_mass__kg_` cross all records, the result is 15600 payload. Then select the `booster_version` and `payload_mass__kg_` for those records where the `payload_mass__kg_` matches the maximum value found in the first step. 12 booster versions can carry the maximum payload.

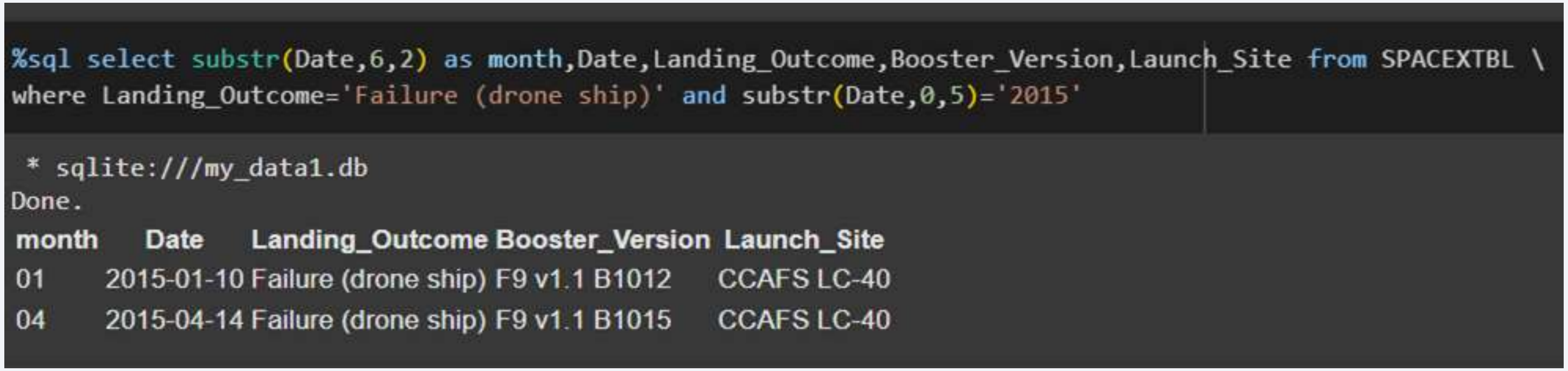


```
%sql select distinct Booster_Version, PAYLOAD_MASS__KG_  Max_Payload from SPACEXTBL \
where PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_) from SPACEXTBL)

 * sqlite:///my_data1.db
Done.
Booster_Version Max_Payload
F9 B5 B1048.4   15600
F9 B5 B1049.4   15600
F9 B5 B1051.3   15600
F9 B5 B1056.4   15600
F9 B5 B1048.5   15600
F9 B5 B1051.4   15600
F9 B5 B1049.5   15600
F9 B5 B1060.2   15600
F9 B5 B1058.3   15600
F9 B5 B1051.6   15600
F9 B5 B1060.3   15600
F9 B5 B1049.7   15600
```

2015 Launch Records

List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

`substr(Column,Start,Lenght)` part extracts the value from string. `substr(Date,6,2)` from the `Date` column, `6` specifies the starting position for extraction (6th character), and `2` specifies the length of the substring to extract (2 characters). As well as `substr(Date,0,5)` part extracts the year and ensures the value equals `'2015'`. There are two results shown.

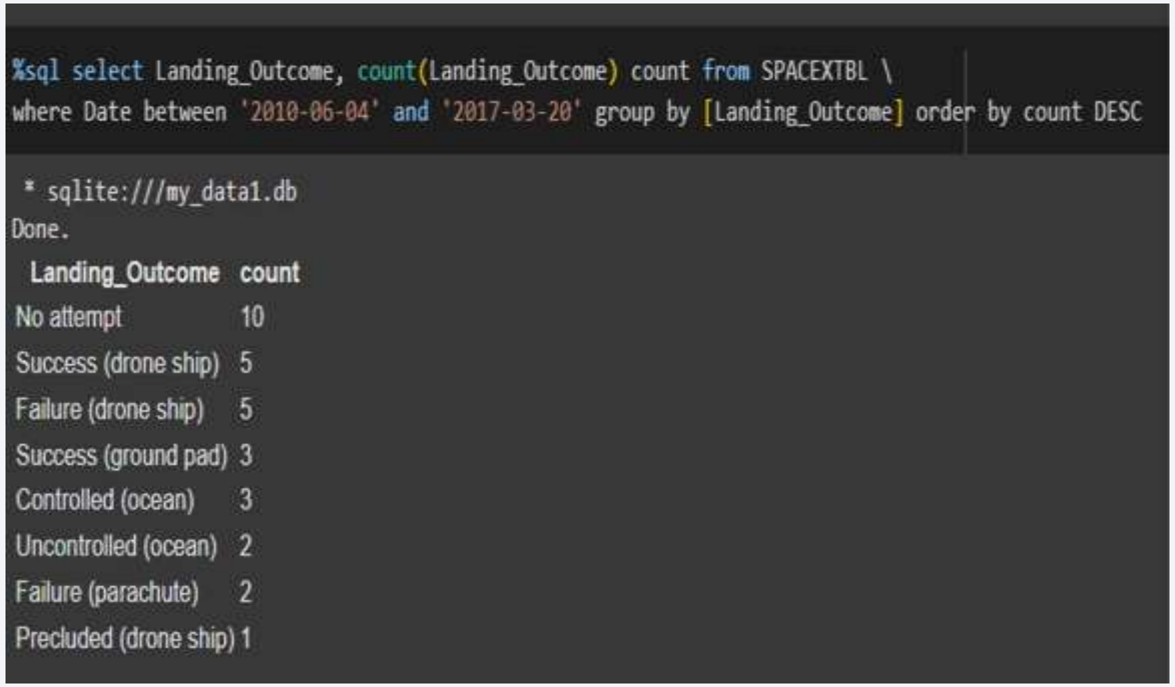


```
%sql select substr(Date,6,2) as month,Date,Landing_Outcome,Booster_Version,Launch_Site from SPACEXTBL \
where Landing_Outcome='Failure (drone ship)' and substr(Date,0,5)='2015'

 * sqlite:///my_data1.db
Done.
month   Date       Landing_Outcome      Booster_Version  Launch_Site
01      2015-01-10 Failure (drone ship) F9 v1.1 B1012    CCAFS LC-40
04      2015-04-14 Failure (drone ship) F9 v1.1 B1015    CCAFS LC-40
```

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

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

`order by count desc` this is the `order by` clause, which sorts the results. It sorts the groups in descending order based on the count column, meaning the landing outcome with the highest count will appear first. It shows that between those times, no attempt has the highest landing outcome.



```
%sql select Landing_Outcome, count(Landing_Outcome) count from SPACEXTBL \
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
```

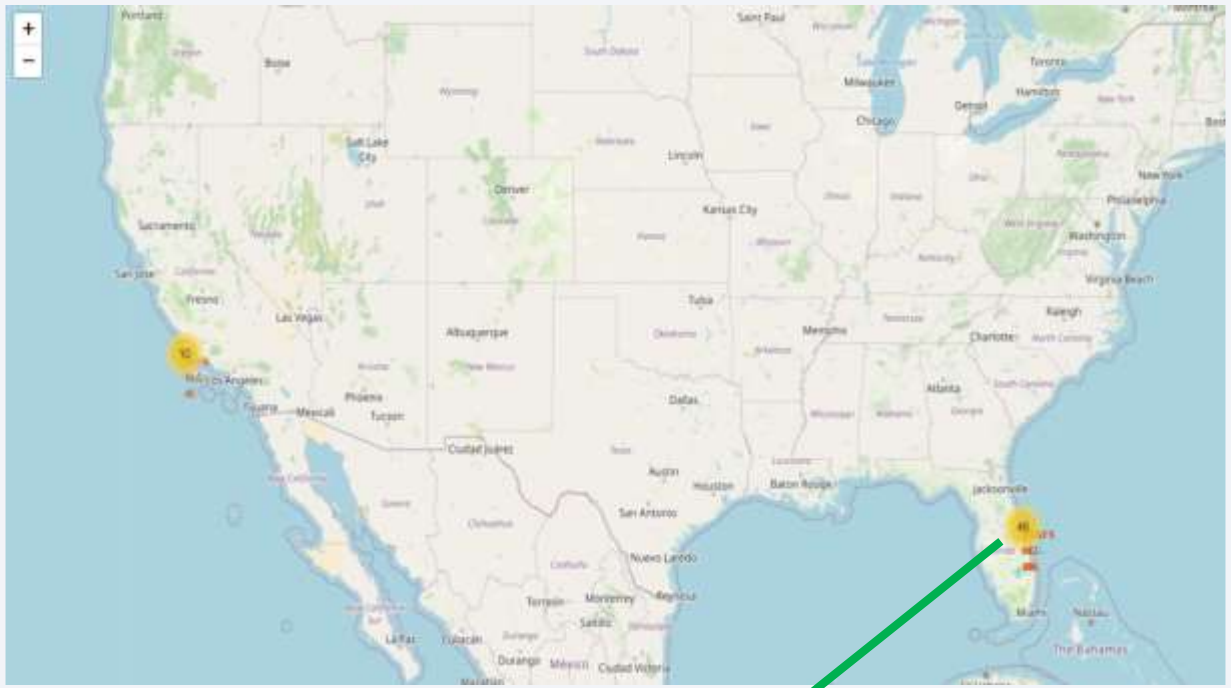
Section 3

Launch Sites Proximities Analysis

Mark All Launch Sites on Map

Rocket launch sites are strategically located near coastlines. This deliberate positioning aims to mitigate the potential dangers associated with rocket launches, such as:

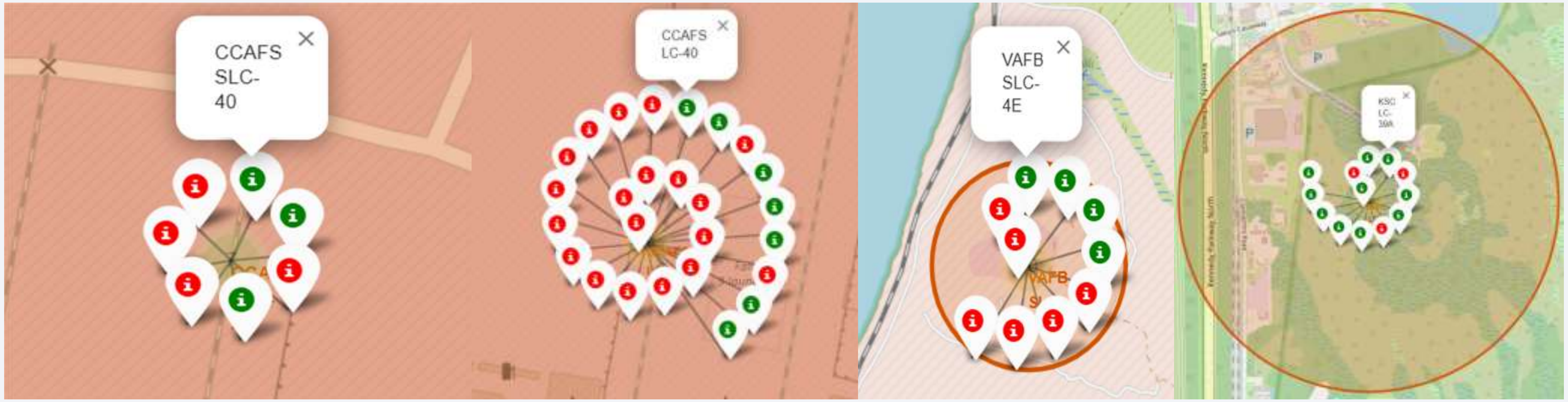
- Launching rockets over the ocean significantly reduces the likelihood of falling debris or explosive events impacting populated areas and
- Minimising the risk of harm to the public in the event of a launch failure or unexpected incident.



Jacksonville
46
Florida

Mark the success/failed launches for each site on the map

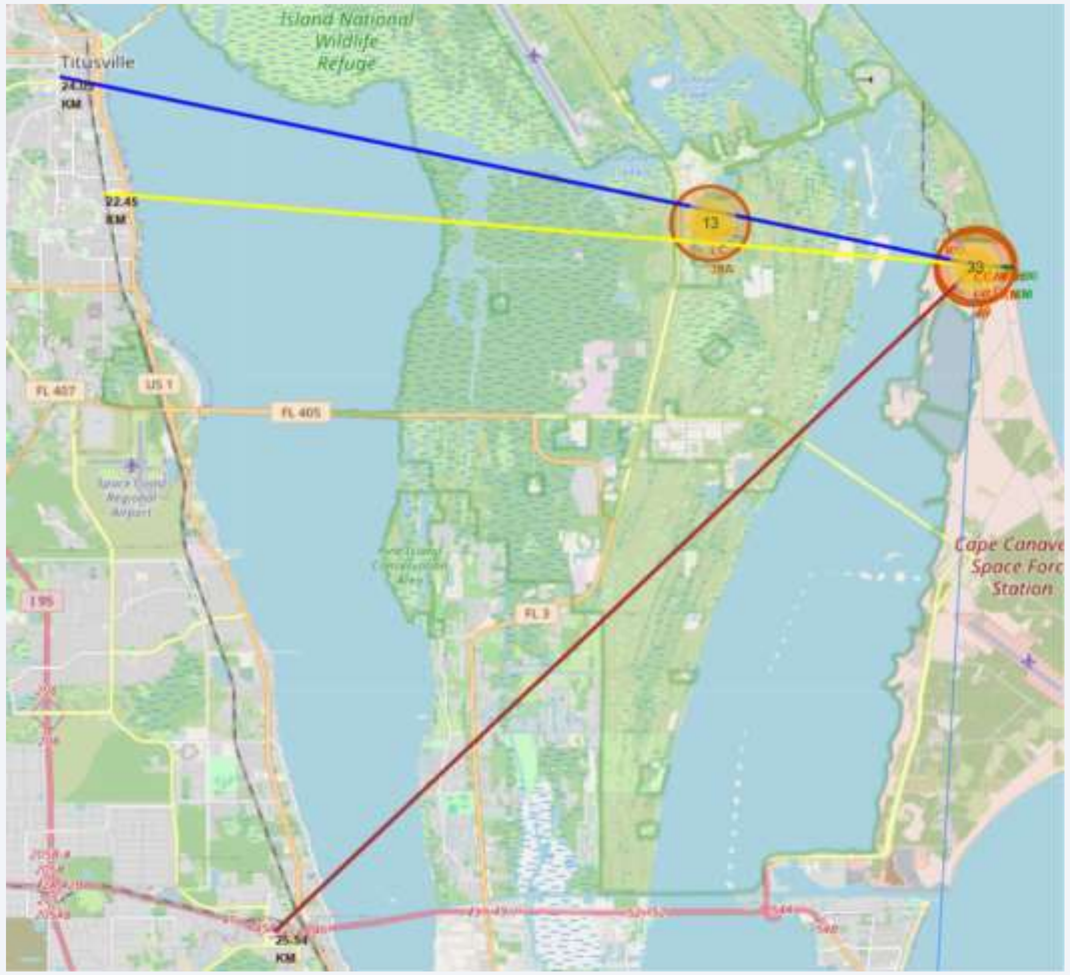
The colour-coded markers provide a visual representation of launch site success rates. Green markers signify successful launches, while red markers indicate failed launches. This visual analysis allows for a quick and intuitive understanding of the relative performance of various launch sites. For example, KSC LC-39A, characterised by a preponderance of green markers, demonstrates a remarkably high success rate in its launch history.



Calculate The Distances and Create Polyline Between Launch Site to Its Proximities

This visualisation emphasizes the launch site's proximity to the coast and its distance from potential sources of disturbance. The launch site is situated in a relatively isolated location, offering a degree of seclusion and tranquillity. Here are the details:

- The nearest coastline is a mere 0.90 km away.
- The nearest railway line is 22.45 km.
- The nearest city is 24.05 km away.
- The nearest highway is 25.54 km away.



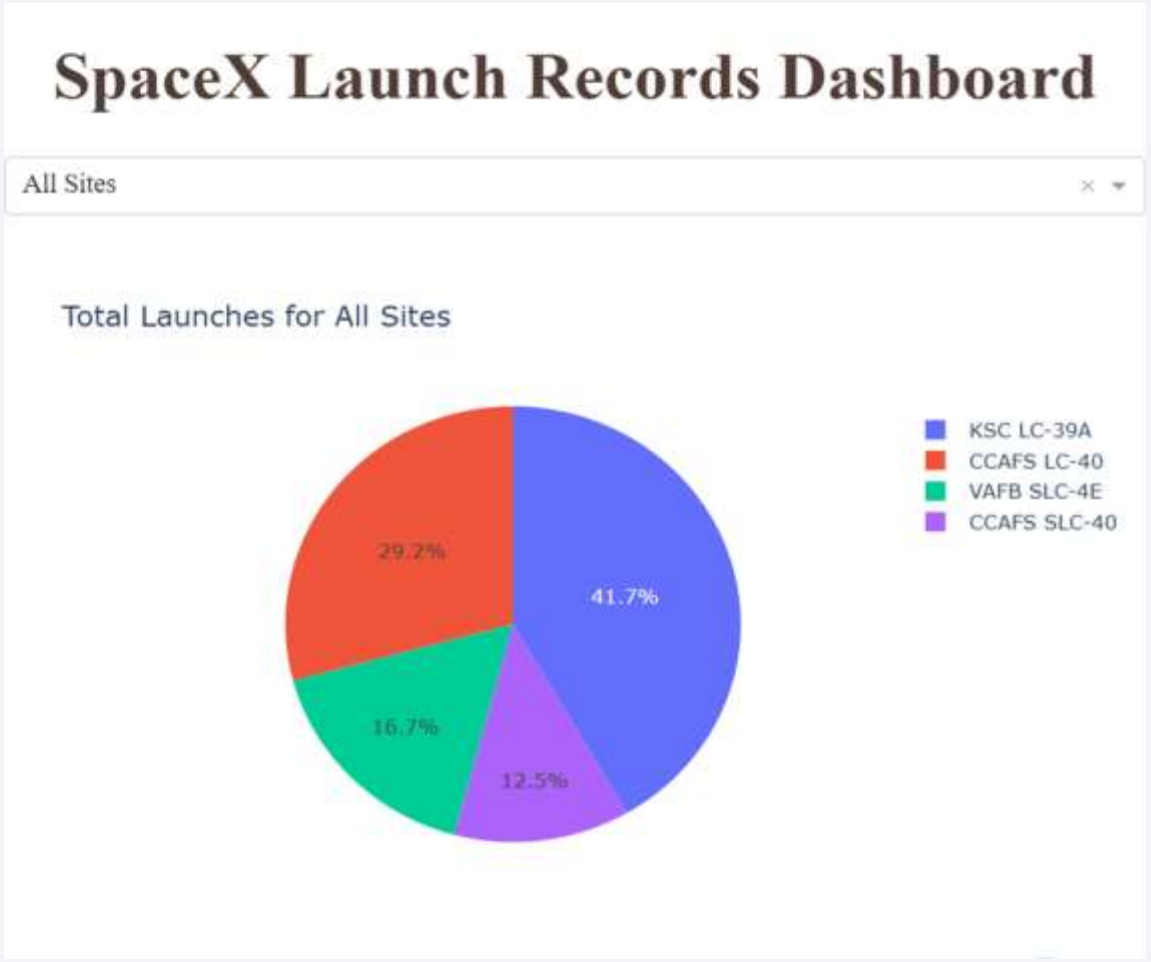
Island National Wildlife Refuge
Titusville
24.05 KM
22.45 KM
13
33
Cape Canave Space Forc Station
25.54 KM

Section 4

Build a Dashboard with Plotly Dash

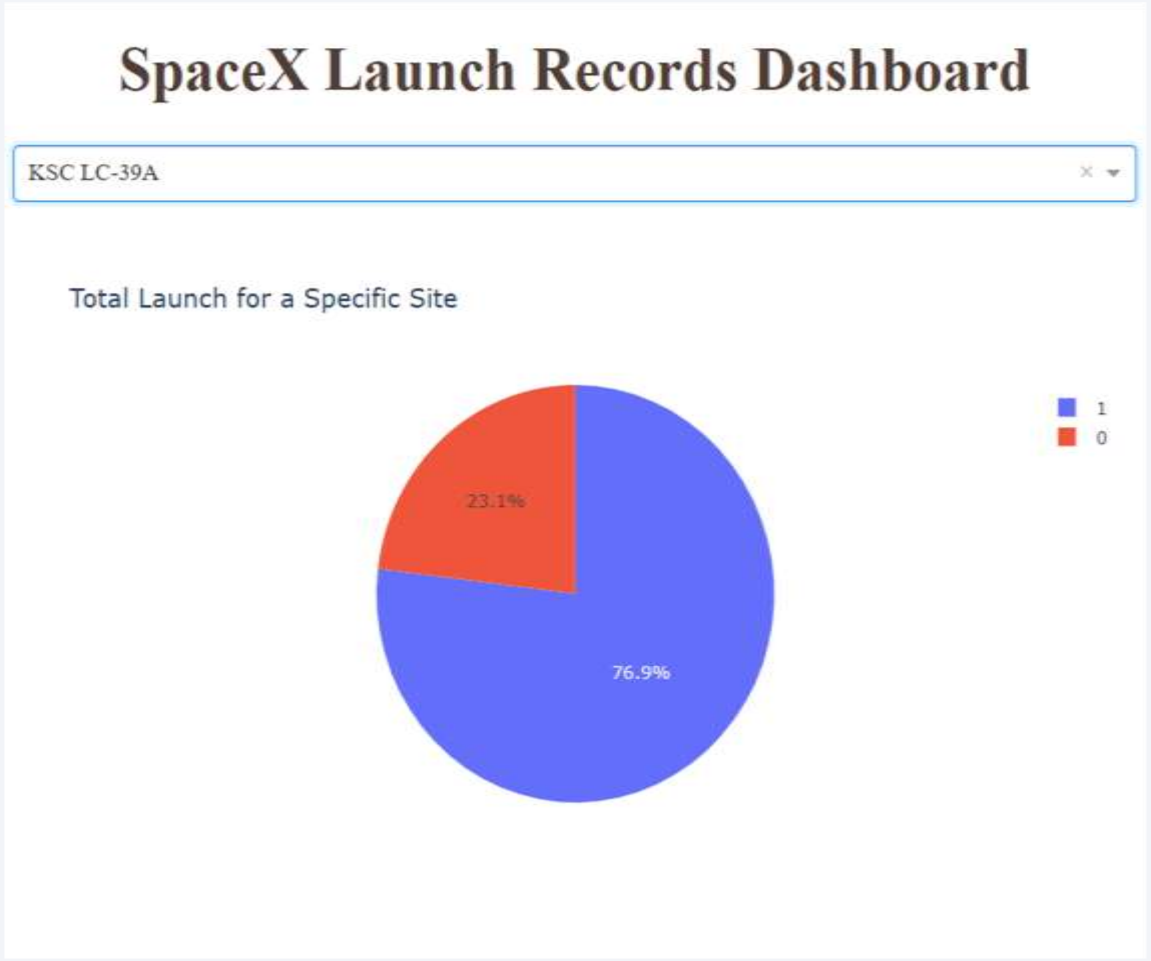
Pie Chart to Show The Total Successful Launches Count

KSC LC-39A stands out as a leader in launch success, boasting an impressive 41.7% success rate among all launch sites. This remarkable statistic highlights the site's exceptional track record and its consistent ability to deliver successful missions. In the second palce is CCAFS LC-40 with 29.2% launch success.



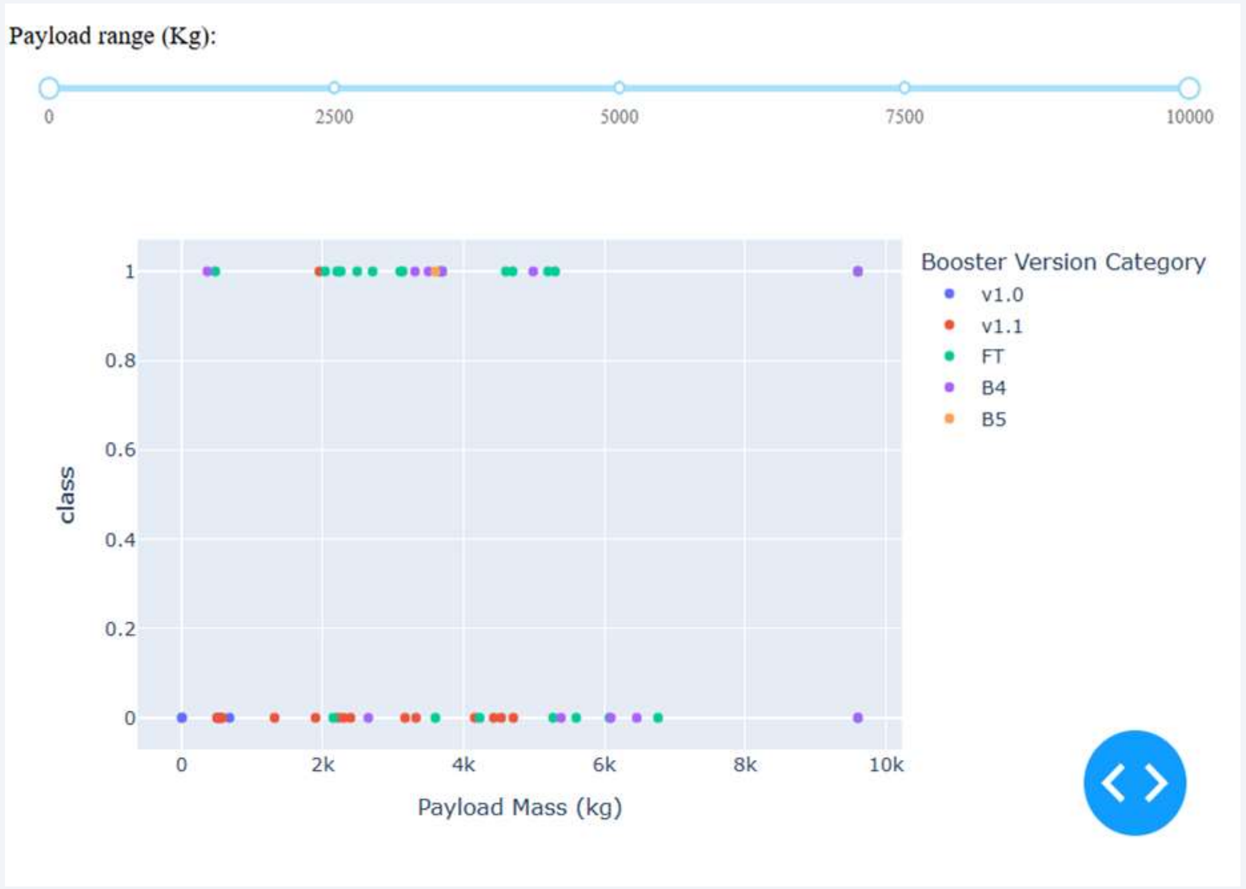
Launch Site with The Highest Success Rate

More details about the KSC LC-39A launch site. A significant 76.9% of launches from this site have been successful. Otherwise, 23.1% of launches from this site have resulted in failures.



Payload vs. Launch Success

The cluster of successful launches (represented by dots at '1' on the y-axis) is densest between the 2,500 kg and 5,000 kg mark on the x-axis. This suggests a strong correlation between this payload range and successful landings.

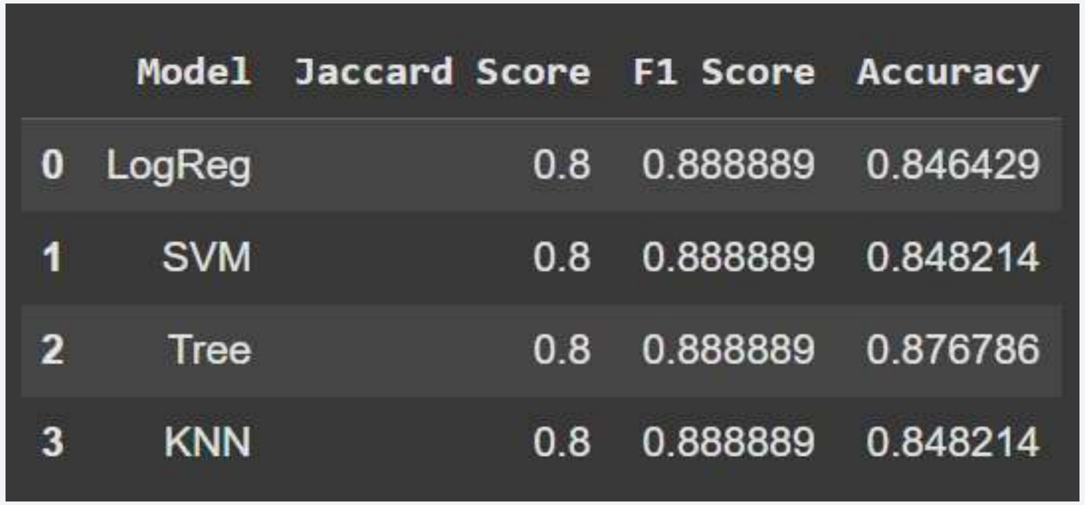


Section 5

Predictive Analysis (Classification)

Classification Accuracy

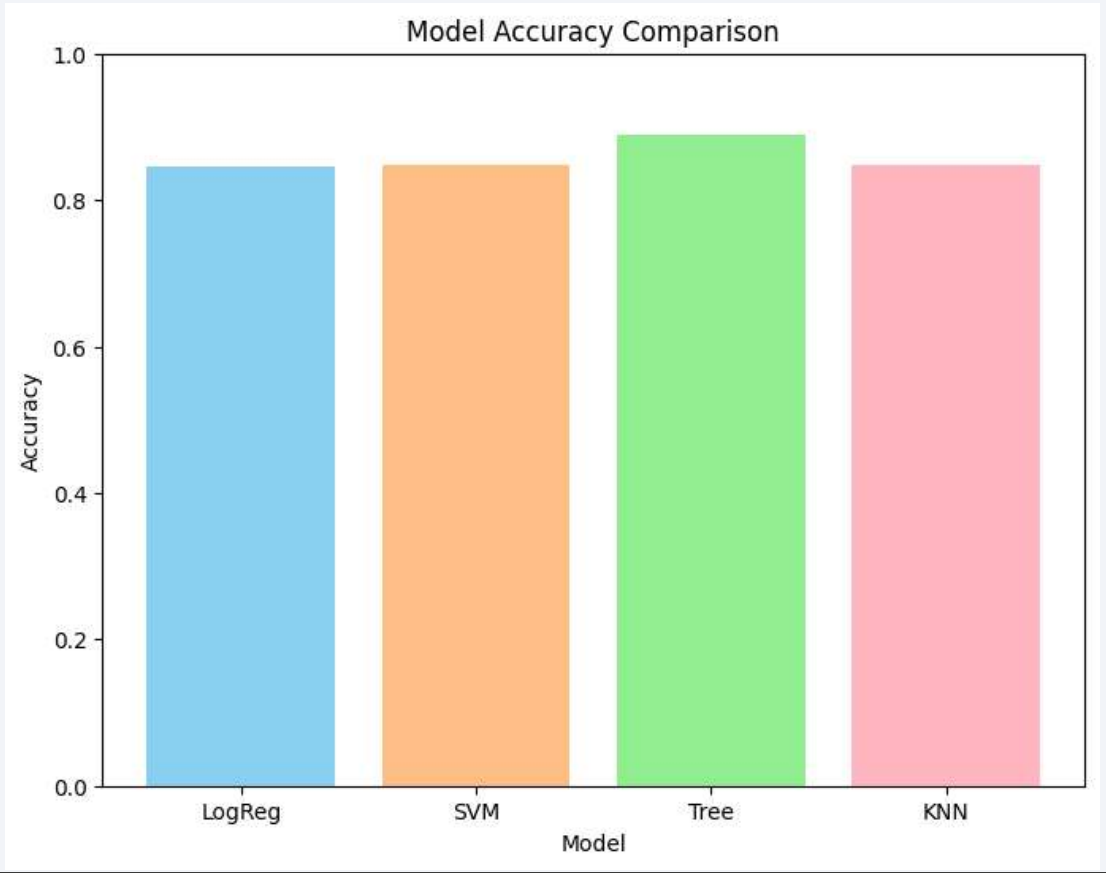
When evaluating the performance of several machine learning models on a relatively small dataset, it was observed that all models exhibited comparable performance levels. This similarity in performance across models is likely attributable to the training data's limited size and insufficient diversity. It shows in this table that the Jaccard and F1 Score has same value. The Accuracy score shows only minimal variation.

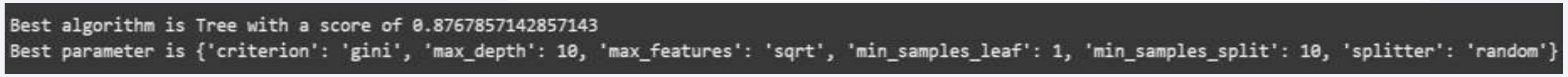


	Model	Jaccard Score	F1 Score	Accuracy
0	LogReg	0.8	0.888889	0.846429
1	SVM	0.8	0.888889	0.848214
2	Tree	0.8	0.888889	0.876786
3	KNN	0.8	0.888889	0.848214

Classification Accuracy

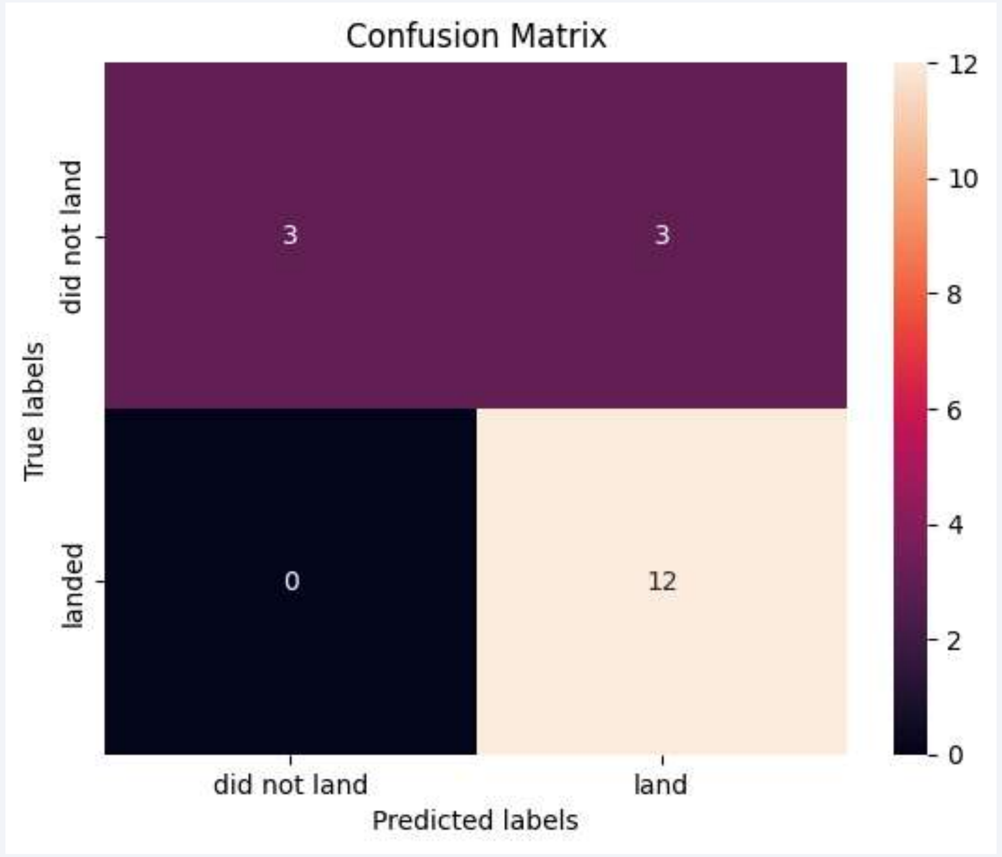
However, a closer examination of the `.best_score_ metric`, which often reflects the model's performance on a held-out validation set during hyperparameter tuning, indicated a slight edge for the Decision Tree model. It has an accuracy of 0.876786. The decision Tree model might have found a slightly better configuration of its hyperparameters, potentially leading to improved generalization performance

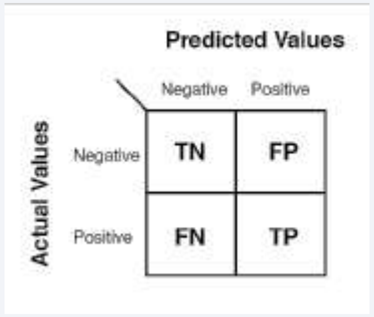




```
Best algorithm is Tree with a score of 0.8767857142857143
Best parameter is {'criterion': 'gini', 'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 10, 'splitter': 'random'}
```

Confusion Matrix





- The confusion matrix for the decision tree classifier demonstrates some success in differentiating between the various landing outcome classes. This suggests that the model has learned some meaningful patterns in the data to categorise landings with a certain level of accuracy.
- However, we need to pay attention to the fact that this model still has flaws. The presence of a high number of false positives. This means the classifier erroneously predicts unsuccessful landings as successful.

Conclusions

- Launch Site Proximity: All launch sites are situated near the coastline.
- SpaceX Launch Success: Since 2013, SpaceX has demonstrated a consistent increase in launch success rates. This positive trend suggests a continuous improvement in launch capabilities, potentially leading to near-perfect launch reliability in the future.
- Launch Sites Performance: Among all launch sites, KSC LC-39A exhibits the highest success rate. Notably, it boasts a 100% success record for launches carrying payloads under 5,500 kilograms.

Conclusions

- **Orbital Success Rates:** Orbits such as ES-L1, GEO, HEO, and SSO have achieved 100% success rates. SSO stands out with the highest success rate, exceeding 100% with more than 1 occurrence..
- **Payload Mass and Success:** A positive correlation exists between payload mass and launch success across all launch sites. Generally, heavier payloads tend to be associated with higher success rates.
- **Best Machine Learning Approach:** The Decision Tree Classifier algorithm has been identified as the most suitable Machine Learning approach for analyzing this dataset.

Conclusions

- **Dataset Limitations and Future Improvements:** The current dataset's limited size may restrict the generalizability of the predictive analytics results. To enhance the robustness of the findings, a larger dataset is crucial.

- **Feature Engineering:** Incorporating additional feature analysis or employing techniques like Principal Component Analysis (PCA) could potentially improve the accuracy and insights derived from the dataset.

Appendix

- Coursera
- IBM Data Science GitHub

Thank you!