IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Goutham Udayagiri
01-12-2024

Outline

Executive Summary

- Collected data from public SpaceX API and SpaceX Wikipedia page. Created labels column ‘class’ which classifies successful landings. Explored data using SQL, visualization, folium maps, and dashboards. Gathered relevant columns to be used as features. Changed all categorical variables to binary using one hot encoding. Standardized data and used Research to find best parameters for machine learning models. Visualize accuracy score of all models.

- Four machine learning models were produced: Logistic Regression, Support Vector Machine, Decision Tree Classifier, and K Nearest Neighbors. All produced similar results with accuracy rate of about 83.33%. All models over predicted successful landings. More data is needed for better model determination and accuracy.

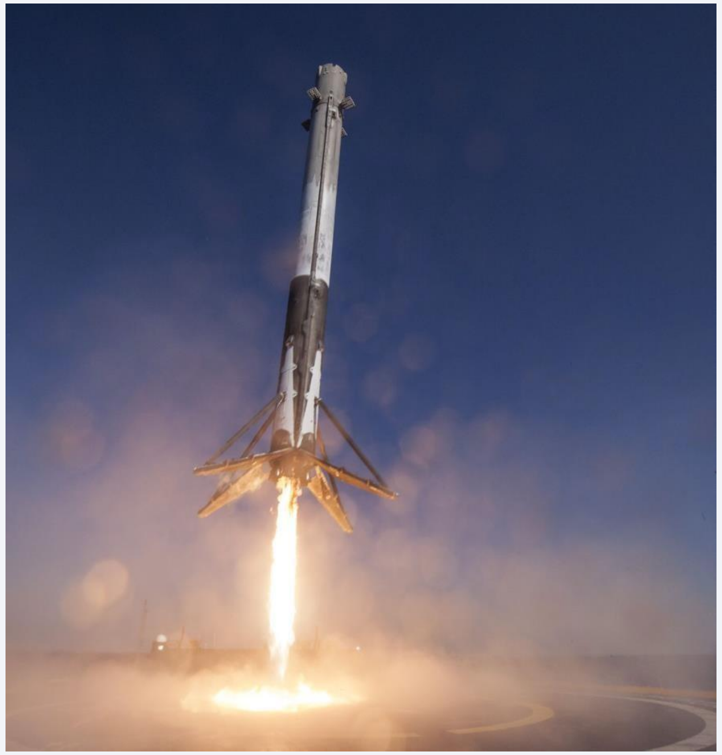
Introduction

Background:

- Commercial Space Age is Here
- Space X has best pricing ($62 million vs. $165 million USD)
- Largely due to ability to recover part of rocket (Stage 1)
- Space Y wants to compete with Space X

Problem:

- Space Y tasks us to train a machine learning model to predict successful Stage 1 recovery



Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Utilize SpaceX REST API (api.spacexdata.com/v4/launches/past) to gather past launch data and Response format: JSON (list of JSON objects).
- Perform data wrangling
  - Convert JSON data to flat table using json normalize and Handle missing data in PayloadMass by replacing NULLs with the column mean
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Select classification algorithm (e.g., Logistic Regression, Decision Trees, Random Forests), Use cross-validation for robust model evaluation.

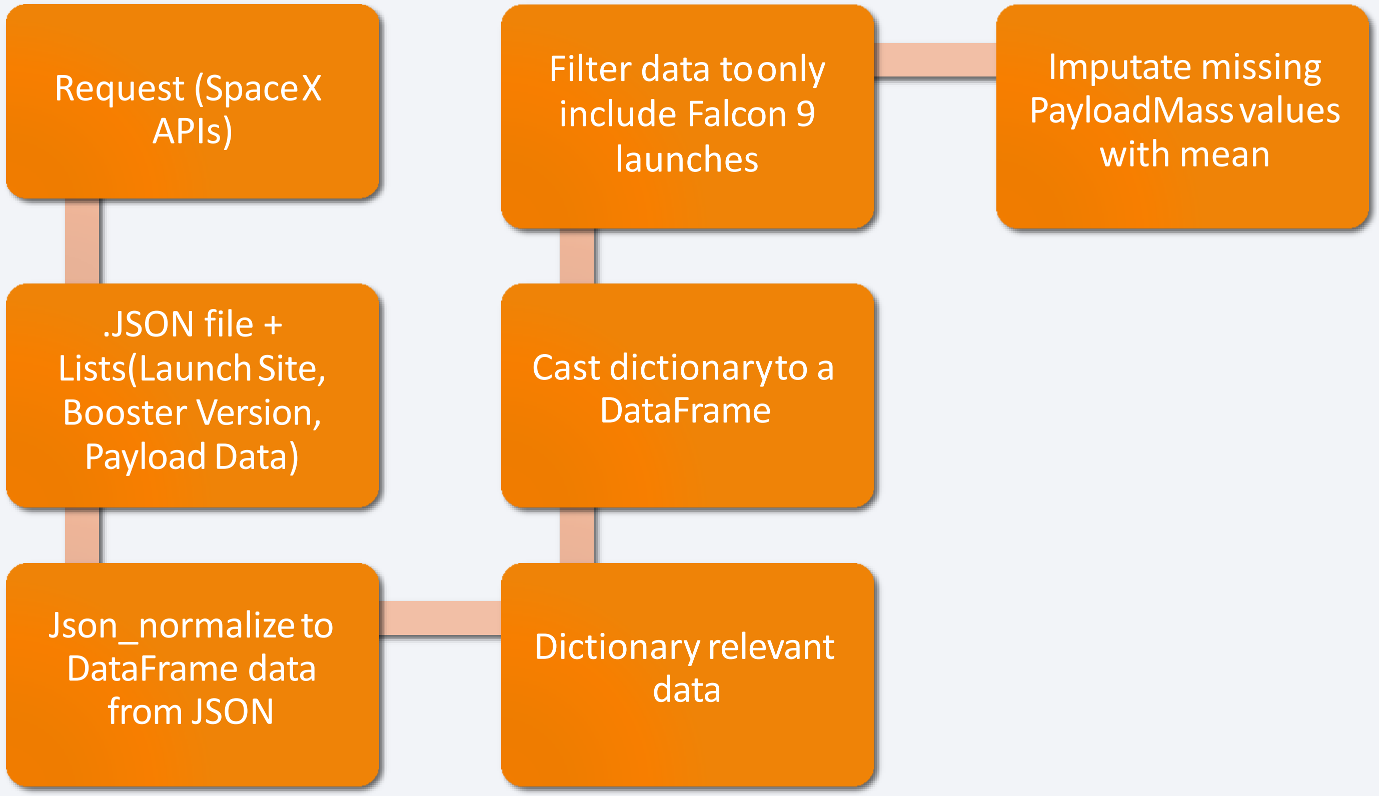
Data Collection

- Utilize SpaceX REST API (api.spacexdata.com/v4/launches/past) to gather past launch data , Response format: JSON (list of JSON objects).
- Convert JSON data to a flat table using json_normalize and Filter data to exclude Falcon 1 launches.
- Handle missing data in PayloadMass by replacing NULLs with the column mean.
- Scrape Falcon 9 launch records from Wiki pages using BeautifulSoup,Parse HTML tables and convert to Pandas DataFrame.
- Fetch additional details for columns like Booster, Launchpad, Payload, and Core using API endpoints and Replace PayloadMass NULLs with the mean and Retain NULLs in LandingPad for one-hot encoding.

Data Collection – SpaceX API

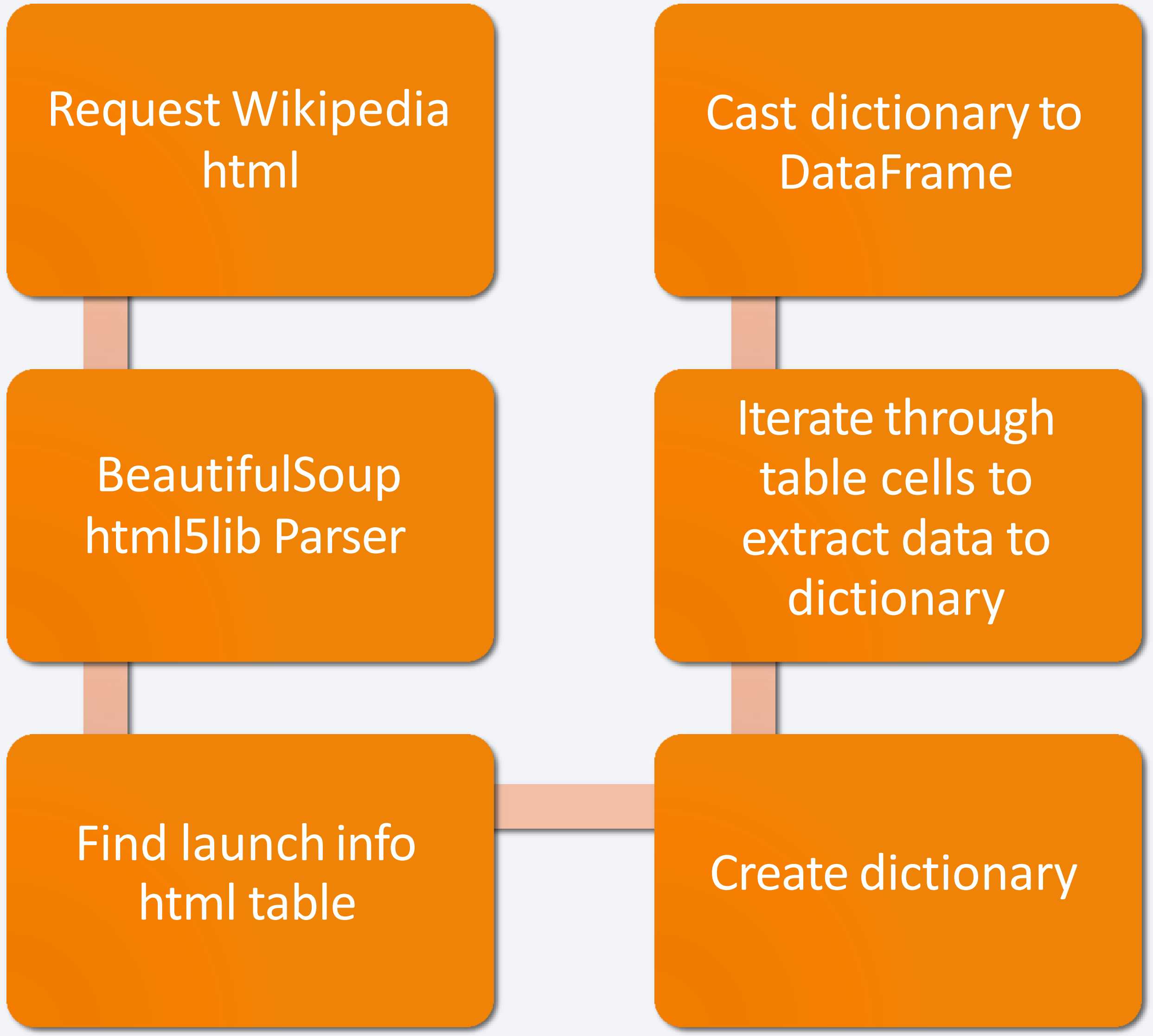
GitHub URL :

[Applied-Data-Science-Capstone/jupyter-labs-spacex-data-collection-api.ipynb at main · Gouthamudayagiri/Applied-Data-Science-Capstone]


Request (SpaceX APIs)
.JSON file + Lists(Launch Site, Booster Version, Payload Data)
Json_normalize to DataFrame data from JSON
Dictionary relevant data
Cast dictionary to a DataFrame
Filter data to only include Falcon 9 launches
Imputate missing PayloadMass values with mean

Data Collection - Scraping

GitHub URL:
[Applied-Data-Science-Capstone/jupyter-labs-webscraping.ipynb at main · Gouthamudayagiri/Applied-Data-Science-Capstone]


Request Wikipedia html
BeautifulSoup html5lib Parser
Find launch info html table
Create dictionary
Iterate through table cells to extract data to dictionary
Cast dictionary to DataFrame

Data Wrangling

- Gather data using SpaceX REST API (api.spacexdata.com/v4/launches/past).
- Web-scrape relevant HTML tables using BeautifulSoup (e.g., Falcon 9 data from Wiki).
- Convert JSON responses into flat tables using json_normalize().
- Parse HTML tables into Pandas Data Frames.
- Filter data to include only Falcon 9 launches, exclude Falcon 1.
- Handle NULL values in PayloadMass: Replace with mean value.
- One-hot encode categorical variables, such as LandingPad.
- Merge additional API data for IDs like Booster, Launchpad, Payload.
- Create a unified dataset ready for analysis.

- GitHub URL:

Applied-Data-Science-Capstone/labs-jupyter-spacex-Data wrangling.ipynb at main · Gouthamudayagiri/Applied-Data-Science-Capstone

EDA with Data Visualization

- Exploratory Data Analysis performed on variables Flight Number, Payload Mass, Launch Site, Orbit, Class and Year.
- Plots Used:
- Flight Number vs. Payload Mass, Flight Number vs. Launch Site, Payload Mass vs. Launch Site, Orbit vs. Success Rate, Flight Number vs. Orbit, Payload vs Orbit, and Success Yearly Trend
- Scatter plots, line charts, and bar plots were used to compare relationships between variables to
- decide if a relationship exists so that they could be used in training the machine learning model

GitHub URL:

Applied-Data-Science-Capstone/edadataviz.ipynb at main · Gouthamudayagiri/Applied-Data-Science-Capstone

EDA with SQL

- Data Filtering: Selected records based on specific conditions (e.g., successful launches, year range).
- Aggregation: Calculated total launches, success rates, and average payload mass.
- Joins: Combined tables to link launch details with booster performance data.
- Grouping: Grouped data by year and booster type for trend analysis.
- Sorting: Ordered results to rank top-performing boosters.

GitHub URL:

Applied-Data-Science-Capstone/jupyter-labs-eda-sql-coursera_sqllite.ipynb at main · Gouthamudayagiri/Applied-Data-Science-Capstone

Build an Interactive Map with Folium

- Added markers for locations, circles for coverage areas, and lines for launch paths. Included popups for extra details and customized tiles for improved visualization.
- Markers: To highlight key points of interest on the map.
- Circles: To visually represent geographical boundaries or coverage zones.
- Lines: To show trajectories or connections between different locations.
- Popups: To provide extra details when clicking on markers or objects.
- Tiles: To enhance map aesthetics and context-specific visual appeal.

GitHub URL:

Applied-Data-Science-Capstone/lab_jupyter_launch_site_location.ipynb at main · Gouthamudayagiri/Applied-Data-Science-Capstone

Build a Dashboard with Plotly Dash

- Dashboard includes a pie chart and a scatter plot.
- Pie chart can be selected to show distribution of successful landings across all launch sites and can be selected to show individual launch site success rates.
- Scatter plot takes two inputs: All sites or individual site and payload mass on a slider between 0 and 10000 kg.
- The pie chart is used to visualize launch site success rate.
- The scatter plot can help us see how success varies across launch sites, payload mass, and
- booster version category.

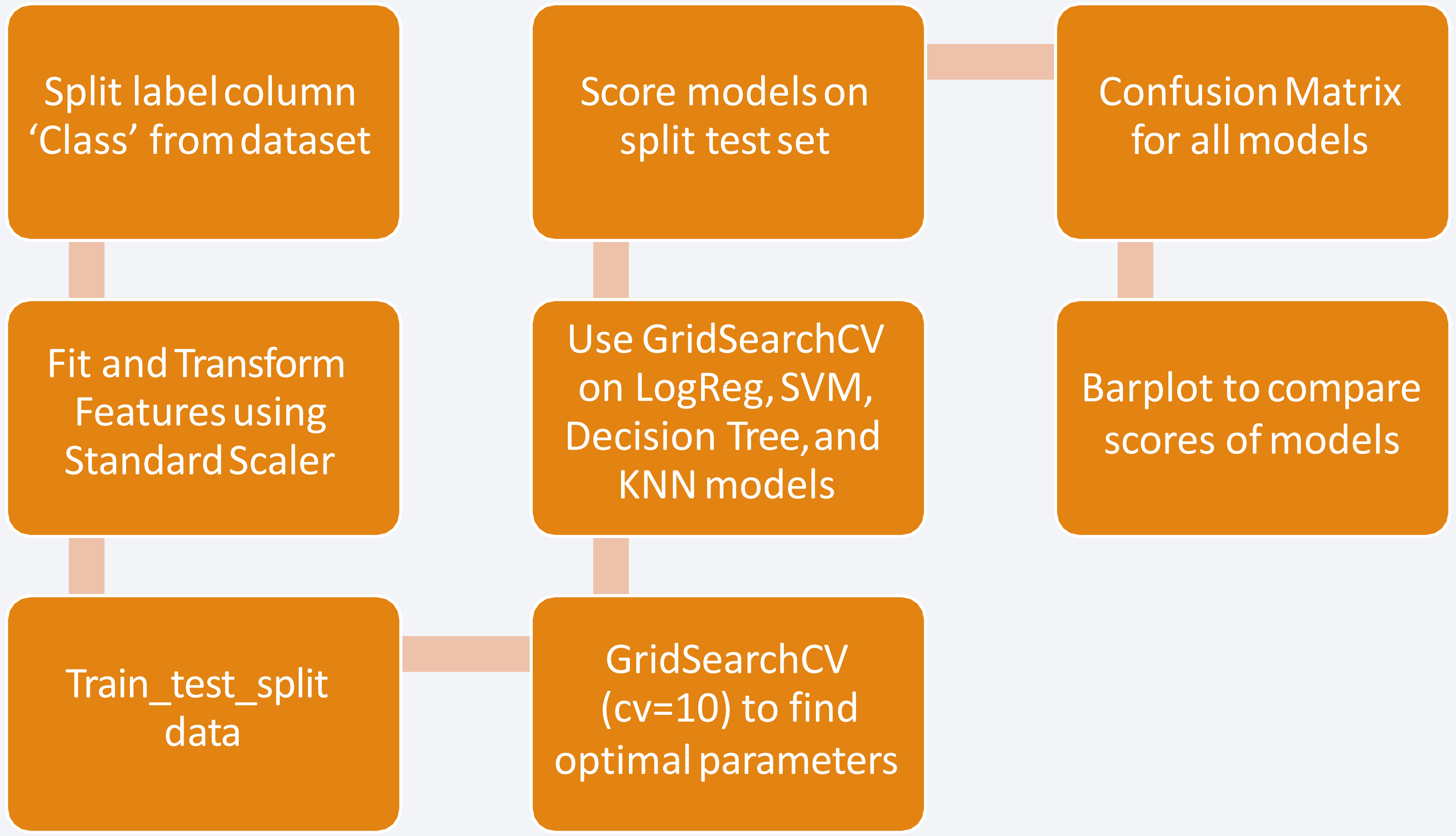
GitHub URL :

Applied-Data-Science-Capstone/spacex_dash_app.py at main · Gouthamudayagiri/Applied-Data-Science-Capstone

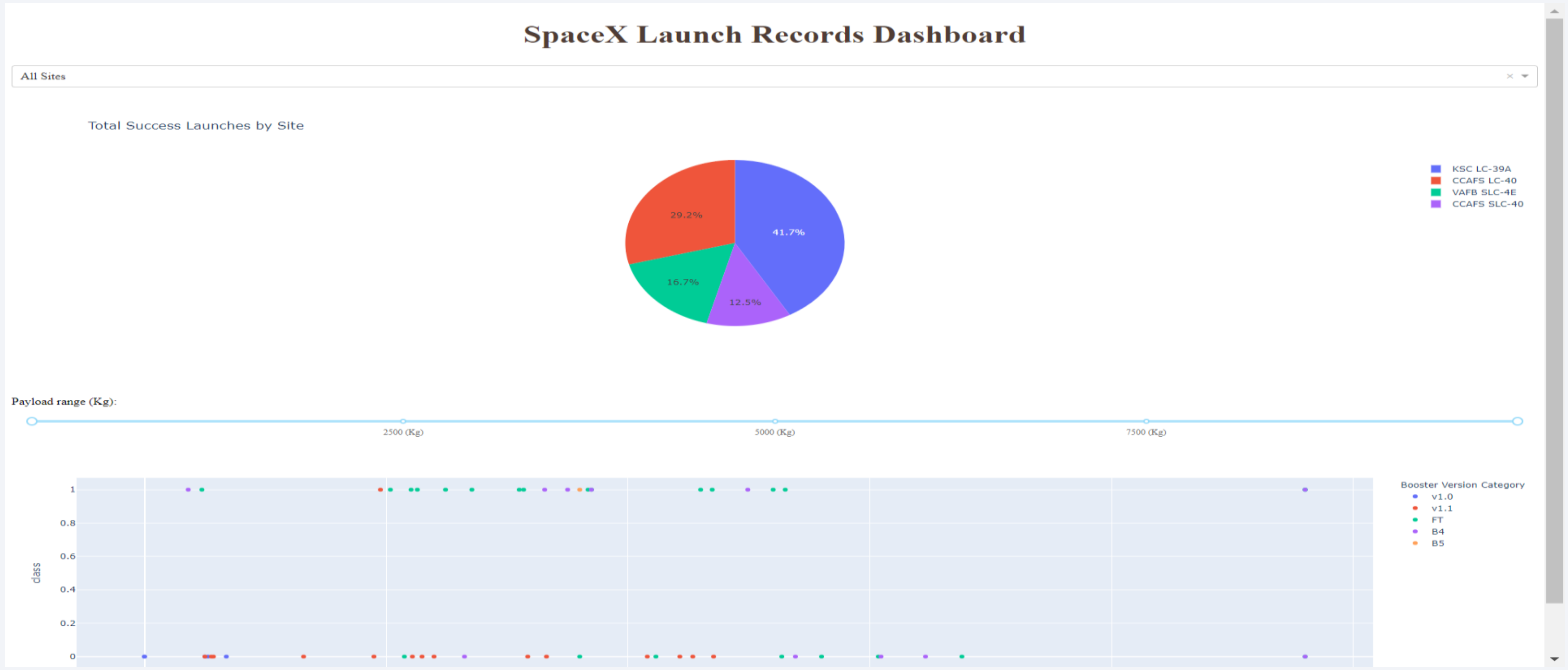
Predictive Analysis (Classification)

GitHub URL:

Applied-Data-Science-Capstone/SpaceX_Machine Learning Prediction_Part_5.ipynb at main · Gouthamudayagiri/Applied-Data-Science-Capstone


Split label column 'Class' from dataset
Fit and Transform Features using Standard Scaler
Train_test_split data
GridSearchCV (cv=10) to find optimal parameters
Use GridSearchCV on LogReg, SVM, Decision Tree, and KNN models
Score models on split test set
Confusion Matrix for all models
Barplot to compare scores of models

Results



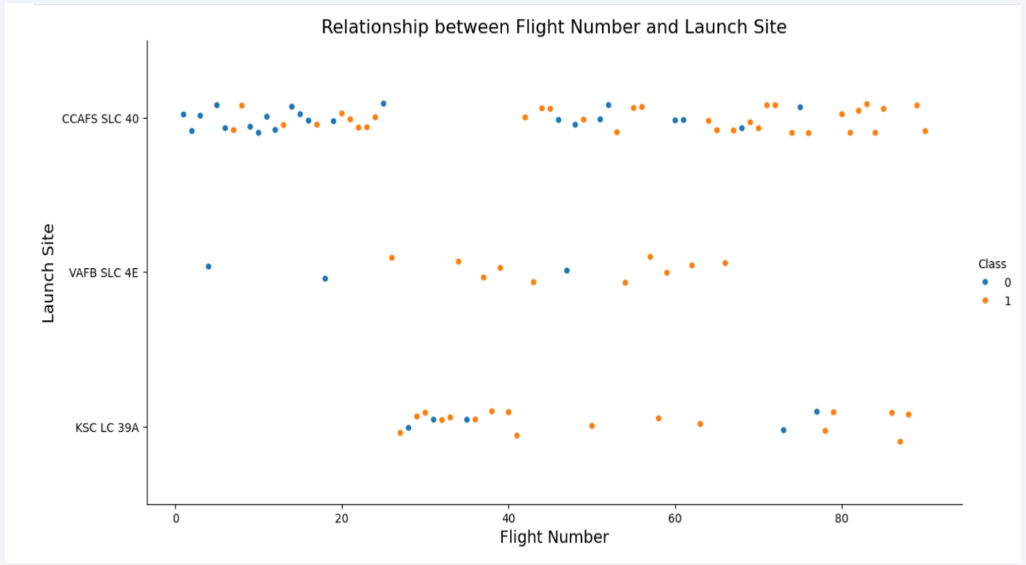
This is a preview of the Plotly dashboard. The following sides will show the results of EDA with visualization, EDA with SQL, Interactive Map with Folium, and finally the results of our model with about 83% accuracy.

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

- Purpose: Shows the relationship between Payload Mass (kg) and Launch Sites, with mission
- success (Class = 1) or failure (Class = 0) indicated by color.
- X-Axis: Payload Mass in kilograms.
- Y-Axis: Launch Sites (CCAFS SLC 40, VAFB SLC 4E, KSC LC 39A).
- Orange: Successful launches (Class = 1).
- Blue: Failed launches (Class = 0).
- CCAFS SLC 40 handles heavier payloads with more successes.
- KSC LC 39A has high success rates for mid-range payloads.
- VAFB SLC 4E shows fewer launches and mixed results.
- Success rates vary by site and payload weight. CCAFS and KSC
- show higher reliability for heavier payloads.



Payload vs. Launch Site

Axes:

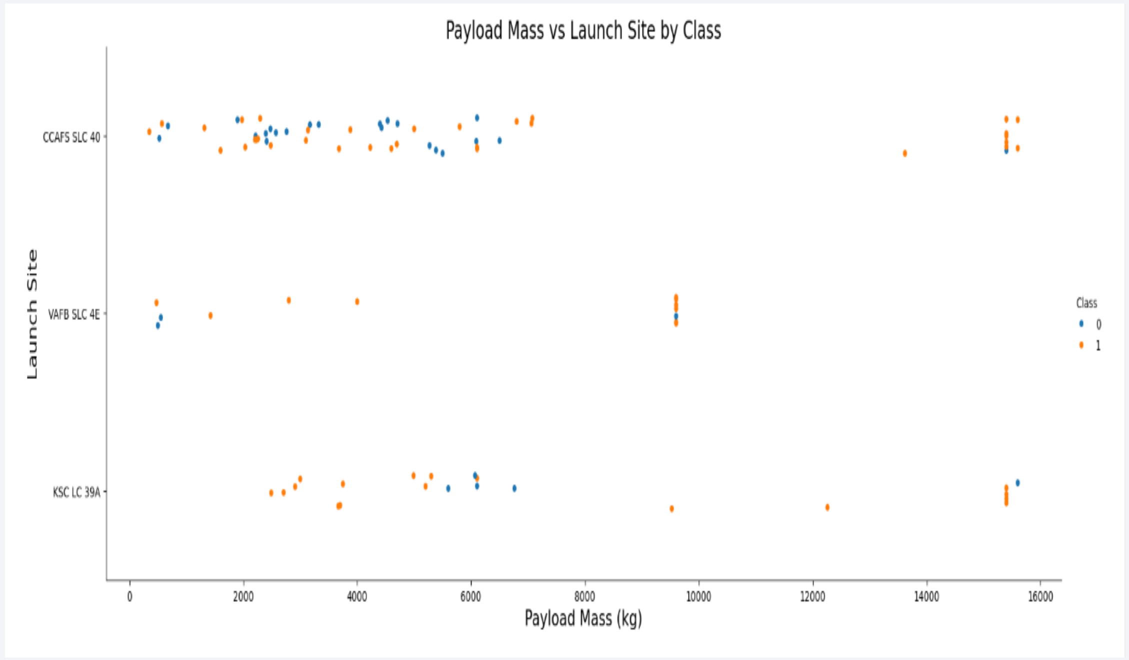
- The x-axis represents the Payload Mass in kilograms.
- The y-axis lists the Launch Sites (e.g., CCAFS SLC-40, KSC LC-39A).

Data Points:

- Each dot corresponds to a launch, with payload mass plotted against the launch site.
- The Class attribute (0 or 1) is indicated by different colors (e.g., success or failure).

Insights:

- Some launch sites (like CCAFS SLC-40) have a wider payload mass range.
- KSC LC-39A launches mostly include higher payloads compared to others.



Success Rate vs. Orbit Type

Axes:

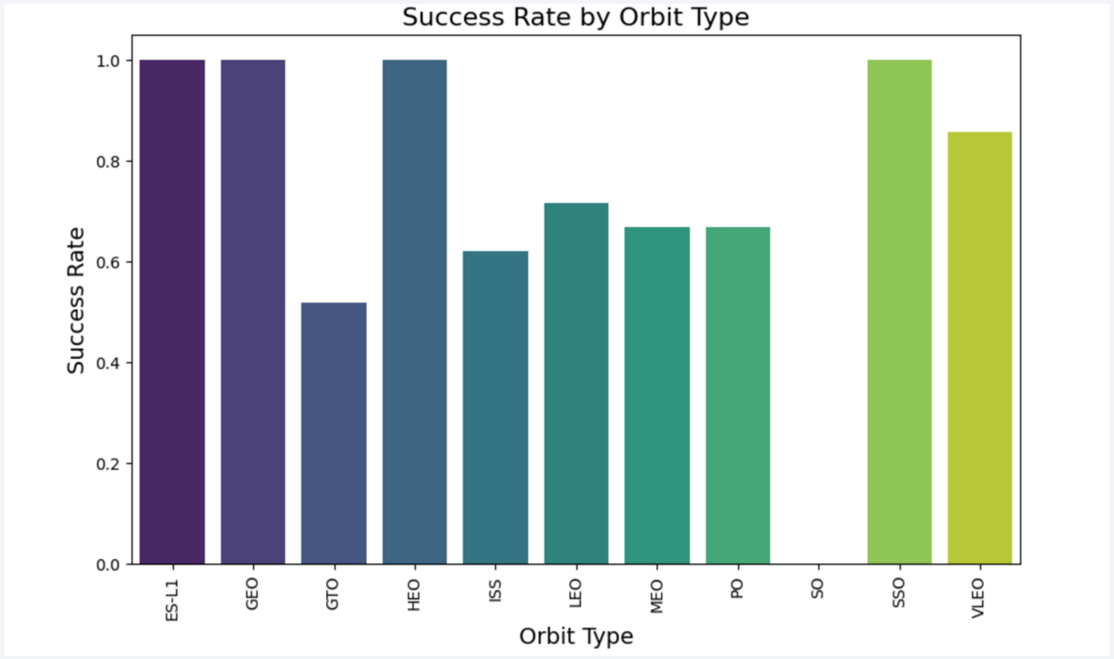
- The x-axis shows various orbit types, such as GEO, GTO, HEO, LEO, SSO, and VLEO.
- The y-axis indicates the success rate on a scale from 0 to 1 (or 0% to 100%).

Bar Heights:

- Orbit types like ES-L1, GEO, and HEO show a 100% success rate, indicating perfect performance.
- GTO and LEO have moderate success rates, with bars around the 50-70% range.
- SSO and VLEO perform well with success rates close to 90% or higher.

Insights:

- Orbit types with high success rates, such as SSO and VLEO, may reflect advanced reliability in those specific mission types.
- Lower success rates could highlight challenges or risks associated with certain orbits like GTO.



Flight Number vs. Orbit Type

Axes:

- The x-axis represents the flight numbers, which increase sequentially.
- The y-axis lists the orbit types (e.g., LEO, GTO, GEO, SSO, etc.).

Data Points:

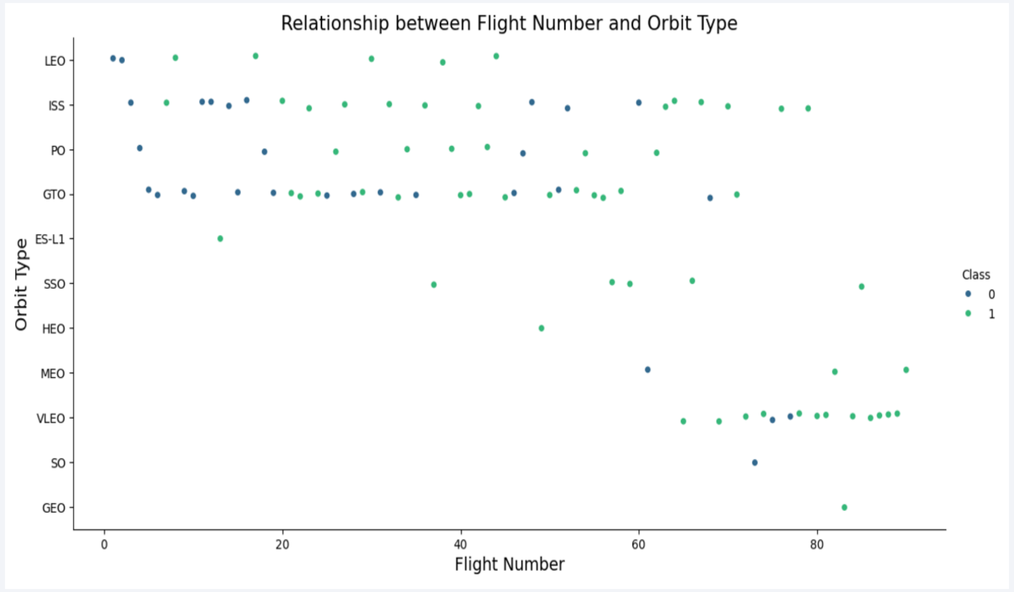
- Each point represents a specific flight.
- The color coding or markers denote the class, likely indicating success (1) or failure (0).

Trends:

- For orbits like LEO and GTO, there is a consistent spread across the range of flight numbers, suggesting frequent missions targeting these orbits.
- SSO and VLEO seem concentrated at higher flight numbers, possibly indicating these orbits were utilized more often in later missions.
- The failure points (class 0) seem fewer overall and may cluster at earlier flight numbers for certain orbits, implying improvement in mission success over time.

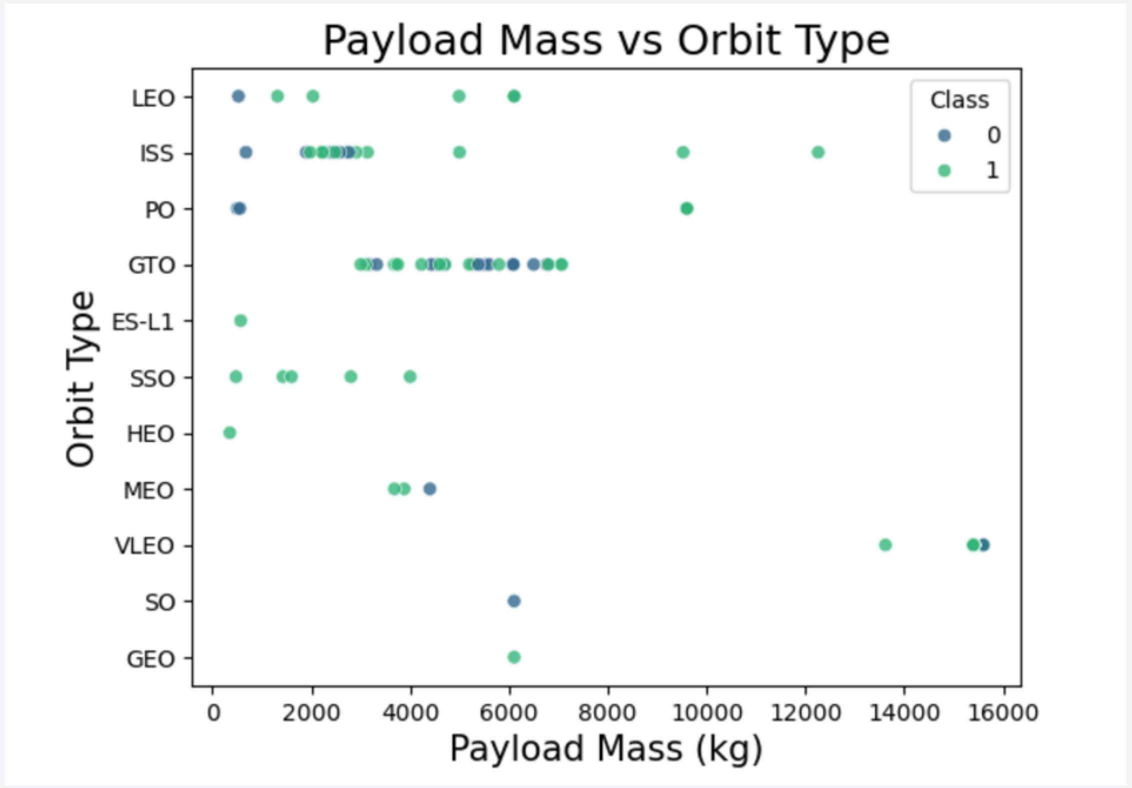
Insights:

- Orbit types like LEO and SSO have a higher density of successful missions (class 1) across the flight history.
- Failure trends might be orbit-specific, offering insight into risk factors for different mission profiles.



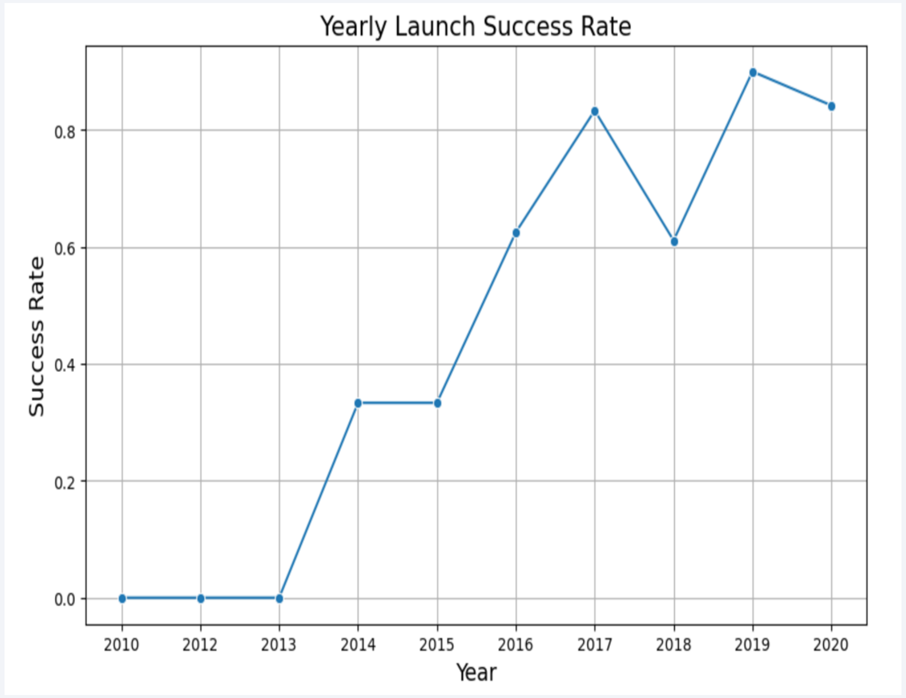
Payload vs. Orbit Type

- The scatter plot visually represents the relationship between payload mass and orbit type. By analyzing this plot, one can observe how payload masses vary across different orbit types. This information is valuable for:
- Space Mission Planning: Understanding the typical payload masses for various orbit types can help in designing and optimizing missions.
- Trend Analysis: Identifying patterns in payload deployment can provide insights into current trends and future directions in space exploration and satellite technology.
- Resource Allocation: By knowing the payload mass distribution, resources can be allocated more efficiently to meet the requirements of different orbit types.

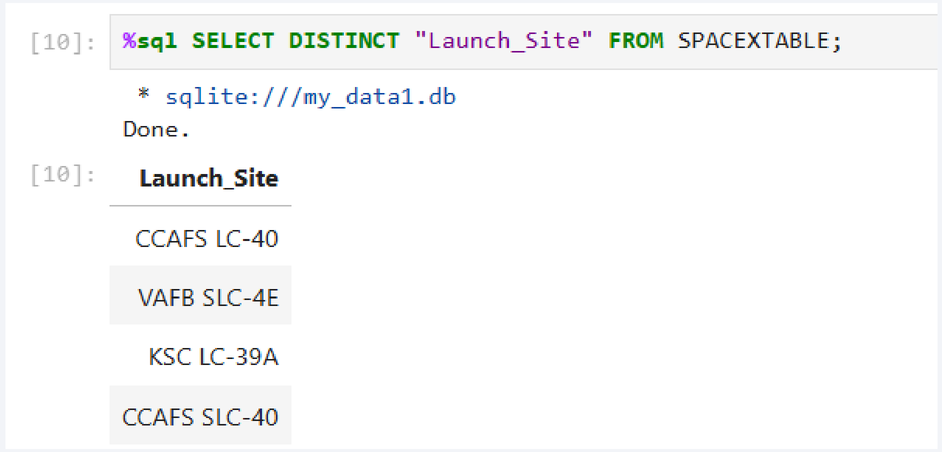


Launch Success Yearly Trend

- The chart visually depicts the trend in launch success rates from 2010 to 2020. Here's a breakdown of what the chart reveals:
- 2010 to 2013: The success rate was very low or almost zero, indicating challenges and perhaps frequent failures in launch attempts during these years.
- 2014: A noticeable increase in success rate marks a significant improvement in launch technology and processes.
- 2015 to 2016: The success rate continues to rise steadily, reflecting ongoing advancements and successful missions.
- 2017: The peak in the success rate suggests a high level of reliability and a milestone in achieving near-perfect launch success.
- 2018: There is a slight drop in the success rate, indicating some challenges or anomalies in that year.
- 2019 to 2020: The success rate rises again, showing recovery and further improvements, ending the decade on a high note.



All Launch Site Names



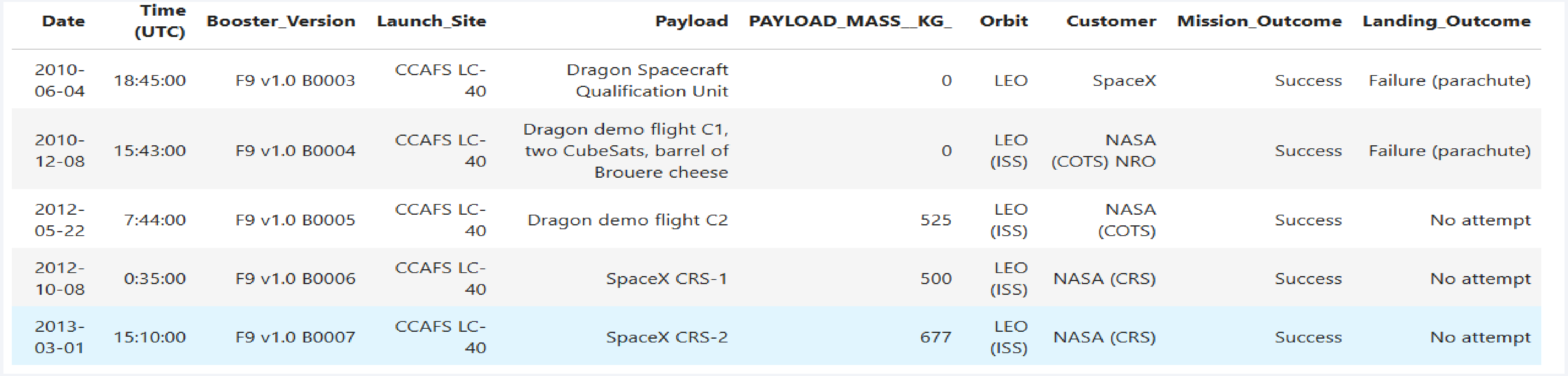
```
[10]: %sql SELECT DISTINCT "Launch_Site" FROM SPACEXTABLE;

 * sqlite:///my_data1.db
Done.
[10]:  Launch_Site
       CCAFS LC-40
       VAFB SLC-4E
       KSC LC-39A
       CCAFS SLC-40
```

- The query SELECT DISTINCT "Launch_Site" FROM SPACEXTABLE; is used to retrieve
- the unique names of the launch sites from the Launch_Site column in the SPACEXTABLE.
- The DISTINCT keyword ensures that only distinct (non-duplicate) launch site names are returned.
- The result shows four unique launch sites used in the dataset:

Launch Site Names Begin with 'CCA'

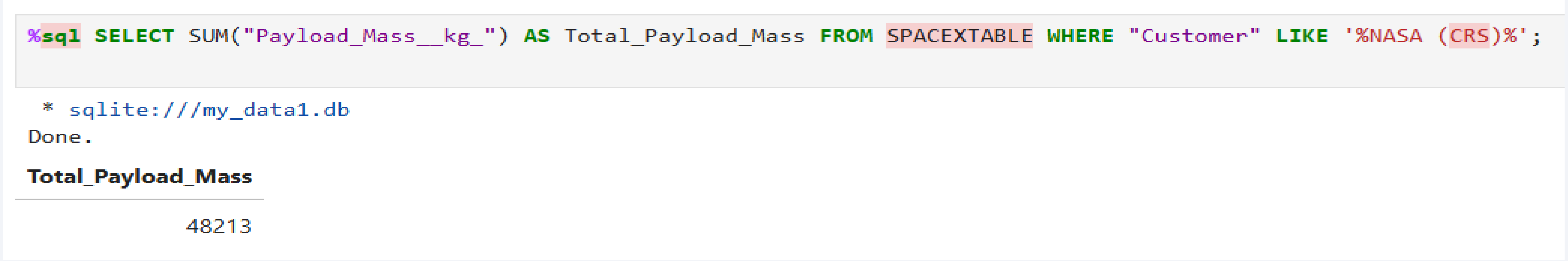
Find 5 records where launch sites begin with `CCA`

- The query retrieves the first 5 records from the SPACEXTABLE where the Launch_Site column starts with the string "CCA".
- The LIKE 'CCA%' condition is used to filter the launch sites that begin with "CCA".
- This query limits the result to 5 records using LIMIT 5. The CCA prefix indicates the launch sites
- that are associated with Cape Canaveral, Florida, a prominent launch location for SpaceX missions



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



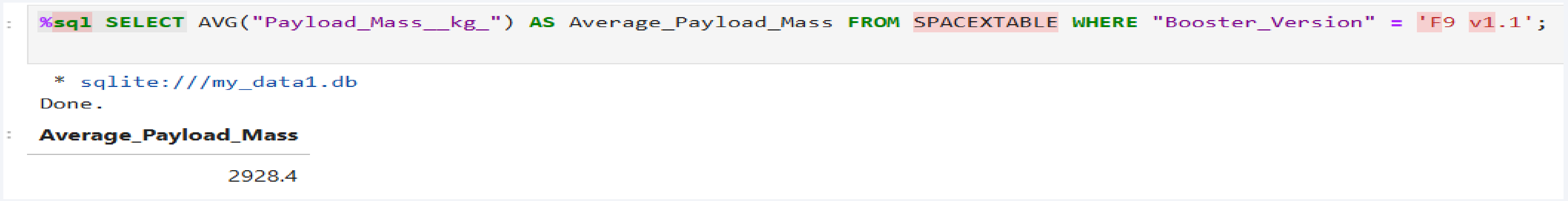
```
%sql SELECT SUM("Payload_Mass__kg_") AS Total_Payload_Mass FROM SPACEXTABLE WHERE "Customer" LIKE '%NASA (CRS)%';
```

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

Total_Payload_Mass
48213

- This SQL query calculates the total payload mass (in kilograms) carried by SpaceX boosters for NASA
- (specifically for the NASA CRS - Commercial Resupply Services missions). The SUM("Payload_Mass__kg_")
- function is used to compute the total payload mass, and the WHERE "Customer" LIKE '%NASA (CRS)%'
- clause filters the records to include only those where the Customer column contains the phrase "NASA (CRS)".
- The query returns a total payload mass of 48,213 kg

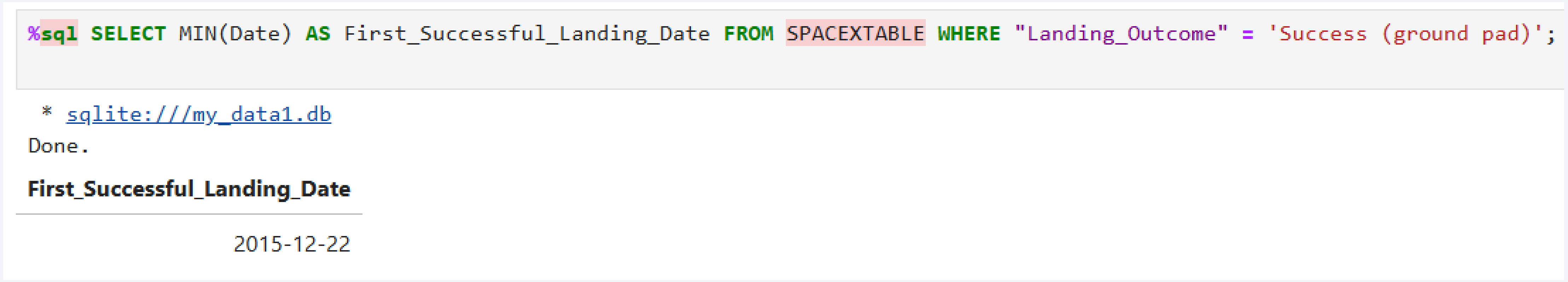
Average Payload Mass by F9 v1.1



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

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

- This SQL query calculates the average payload mass (in kilograms) carried by the SpaceX boosters of version F9 v1.1.
- The AVG("Payload_Mass__kg_") function computes the average value of the payload mass from the records where
- the Booster_Version is equal to 'F9 v1.1'. The query returns an average payload mass of 2,928.4 kg

First Successful Ground Landing Date



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

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

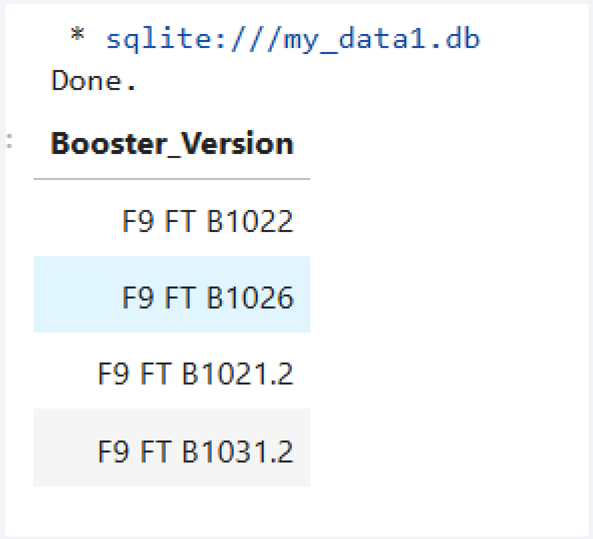
First_Successful_Landing_Date
2015-12-22

- This SQL query retrieves the first successful landing date of a SpaceX mission where the landing outcome was 'Success (ground pad)'.
- The MIN(Date) function returns the earliest date from the records that match this condition.
- The query result shows the date 2015-12-22, which indicates that the first successful landing
- on a ground pad occurred on December 22, 2015.

Successful Drone Ship Landing with Payload between 4000 and 6000

- This SQL query retrieves the unique booster versions that successfully landed on a drone ship
- and had a payload mass between 4000 kg and 6000 kg.
- The query uses the condition "Landing_Outcome" = 'Success (drone ship)' to filter for successful
- landings on the drone ship, and the Payload_Mass__kg_ > 4000 AND Payload_Mass__kg_ < 6000
- condition to narrow down the payload mass.

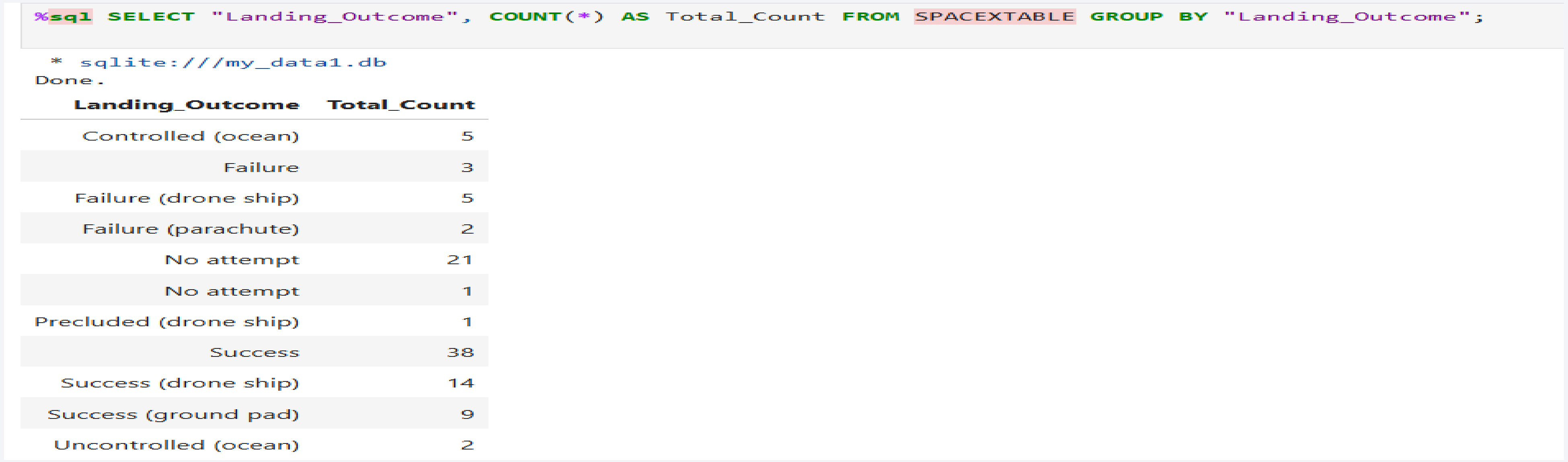
- The query result lists the following booster versions:
- F9 FT B1022
- F9 FT B1026
- F9 FT B1021.2
- F9 FT B1031.2



```
 * 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



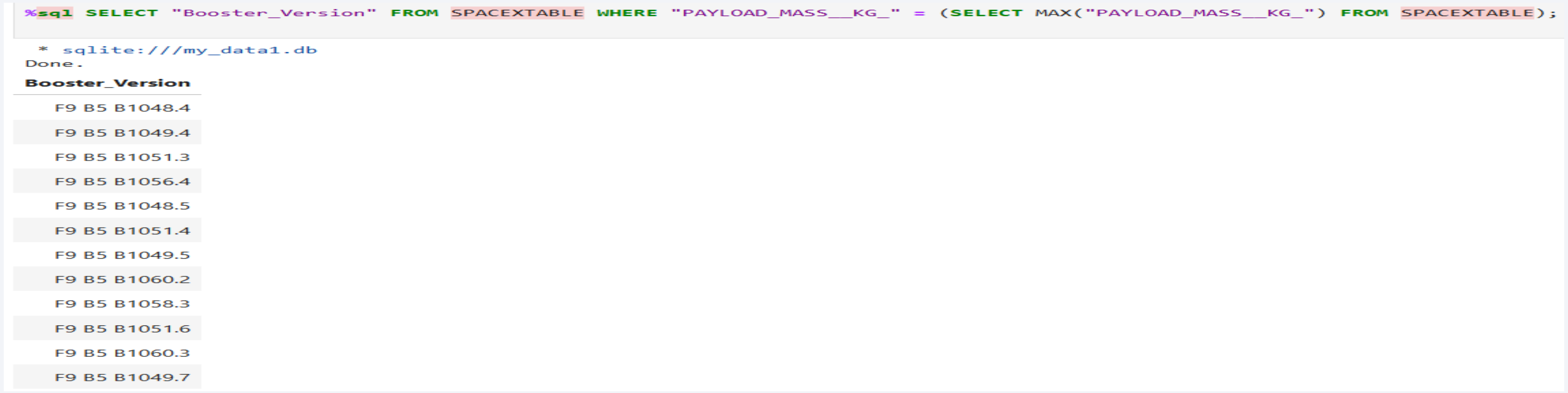
```
%sql SELECT "Landing_Outcome", COUNT(*) AS Total_Count FROM SPACEXTABLE GROUP BY "Landing_Outcome";
```

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

Landing_Outcome	Total_Count
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

- The SQL query groups the data by the Landing_Outcome column and counts the occurrences of each outcome.
- Based on the results, we can calculate the total number of successful and failure mission outcomes.

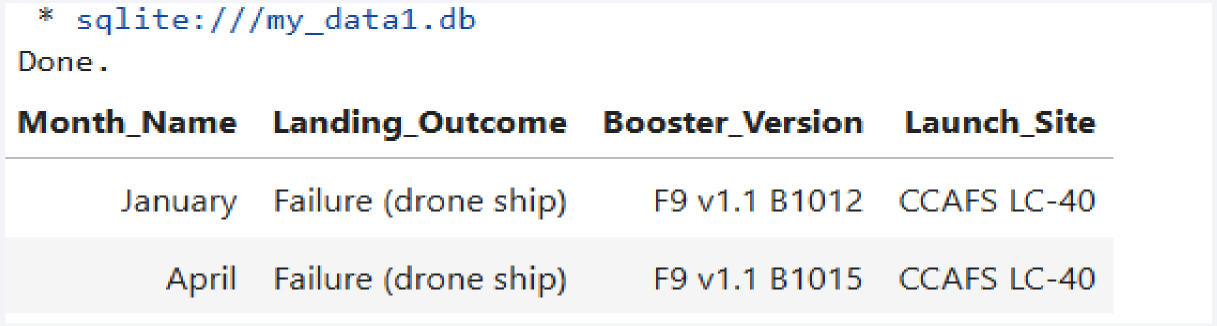
Boosters Carried Maximum Payload



```
%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
```

- The SQL query is designed to return the booster versions that have carried the maximum payload mass.
- It works in two parts:
- The subquery (SELECT MAX("PAYLOAD_MASS__KG_") FROM SPACEXTABLE) finds the highest payload mass in the table.
- The main query fetches the names of the booster versions that carried that maximum payload mass.

2015 Launch Records



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

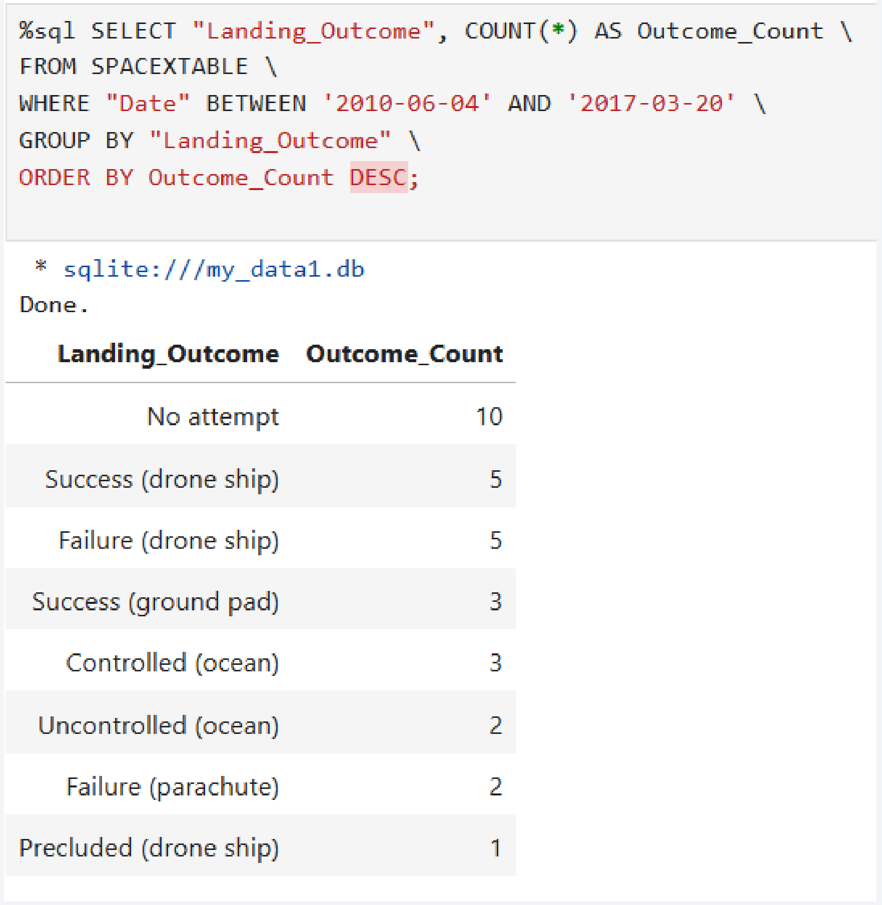
Month_Name	Landing_Outcome	Booster_Version	Launch_Site
January	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
April	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

- Date Handling: The query extracts the month from the Date column (which is in the format 'YYYY-MM-DD')
- and converts the month number into the full month name (e.g., '01' to 'January').
- Conditions: It filters the data for the year 2015 (WHERE SUBSTR("Date", 1, 4) = '2015')
- and selects records where the Landing_Outcome includes "Failure (drone ship)".
- Query Result: The result lists two failed landing outcomes on the drone ship in 2015:

- January: Failure (drone ship) by Booster Version F9 v1.1 B1012 launched from CCAFS LC-40.
- April: Failure (drone ship) by Booster Version F9 v1.1 B1015 launched from CCAFS LC-40.

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

- Most Frequent Outcome: "No attempt" occurred 10 times, making it the most frequent landing outcome between June 4, 2010, and March 20, 2017.
- Drone Ship Success/Failure: Both "Success (drone ship)" and "Failure (drone ship)" each occurred 5 times, indicating a notable number of attempts on drone ships.
- Ground and Ocean Success: "Success (ground pad)" and "Controlled (ocean)" both occurred 3 times each.
- Less Frequent Outcomes: "Uncontrolled (ocean)" and "Failure (parachute)" occurred 2 times each.
- Rare Outcome: "Precluded (drone ship)" was the least frequent, occurring only once.



```
%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;
```

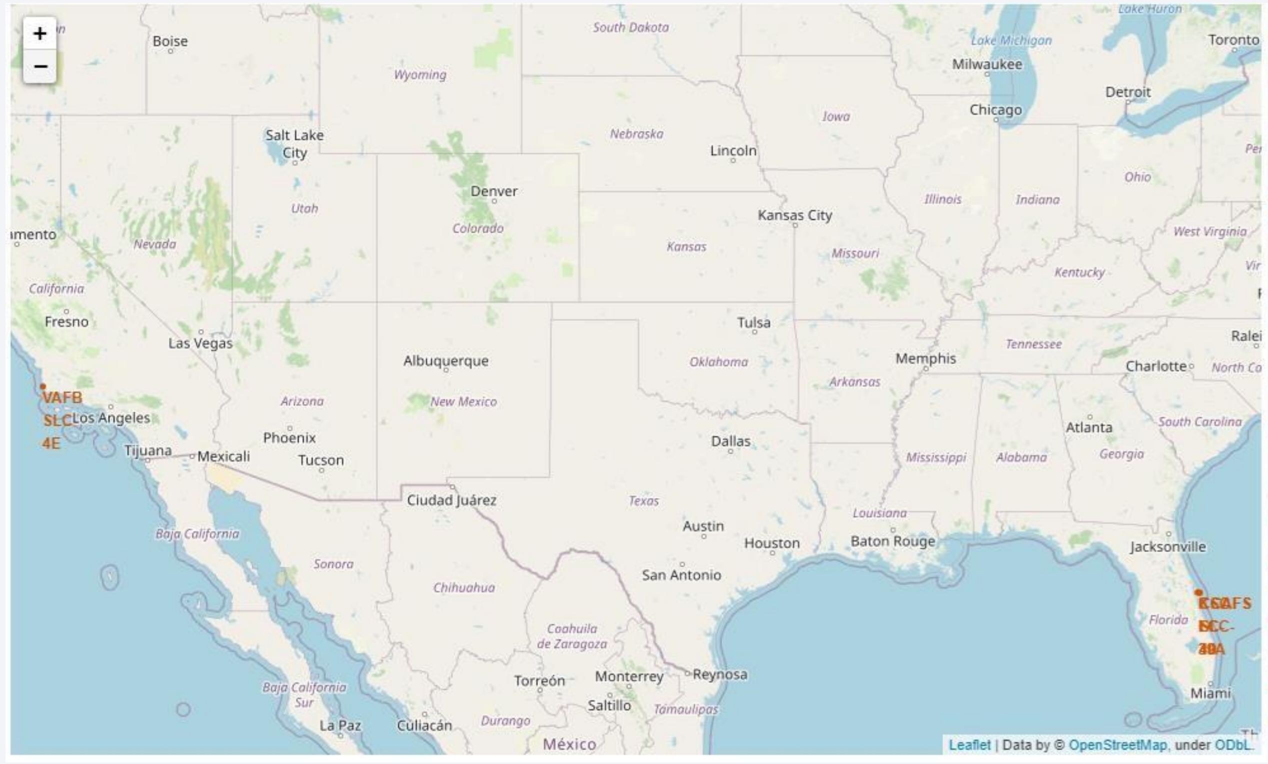
```
 * sqlite:///my_data1.db
Done.
```

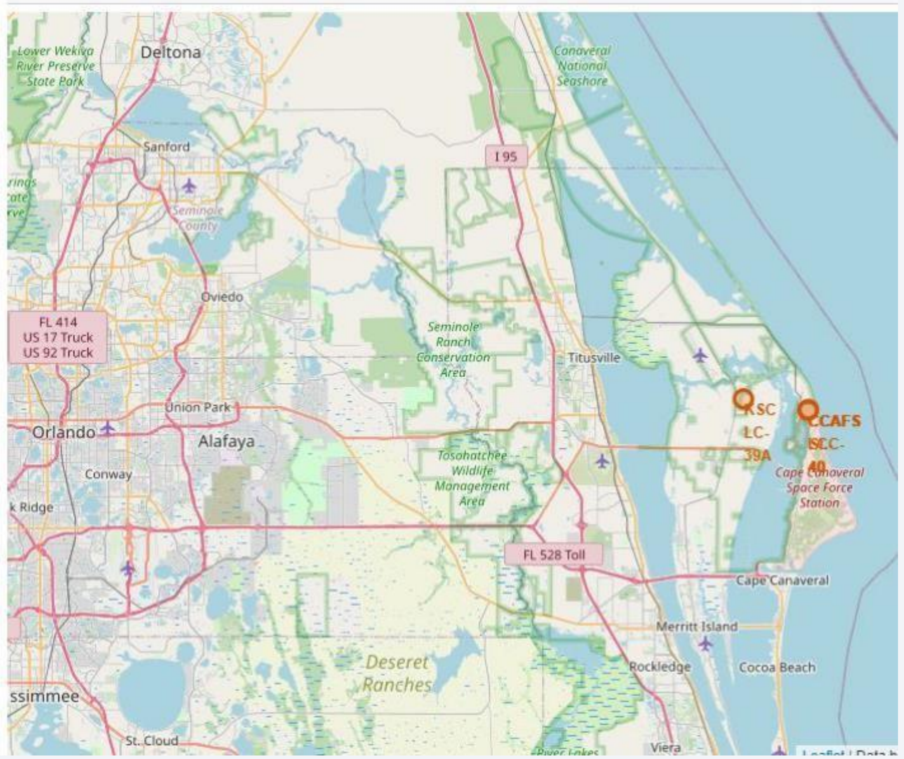
Landing_Outcome	Outcome_Count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

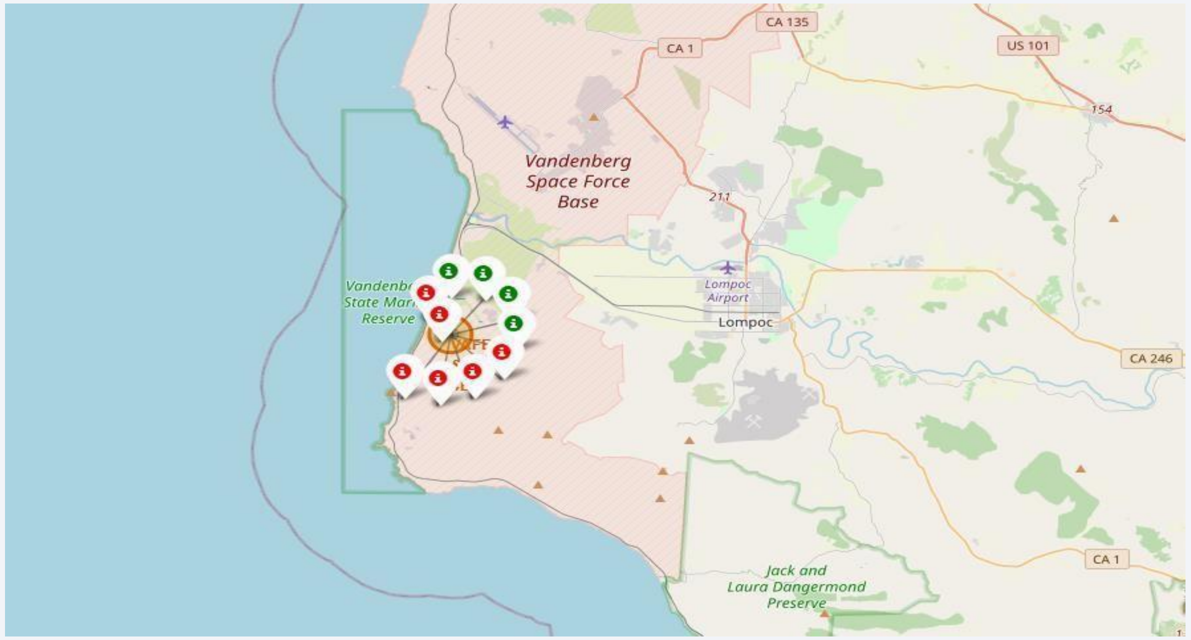
Geographical Distribution of Launch Sites





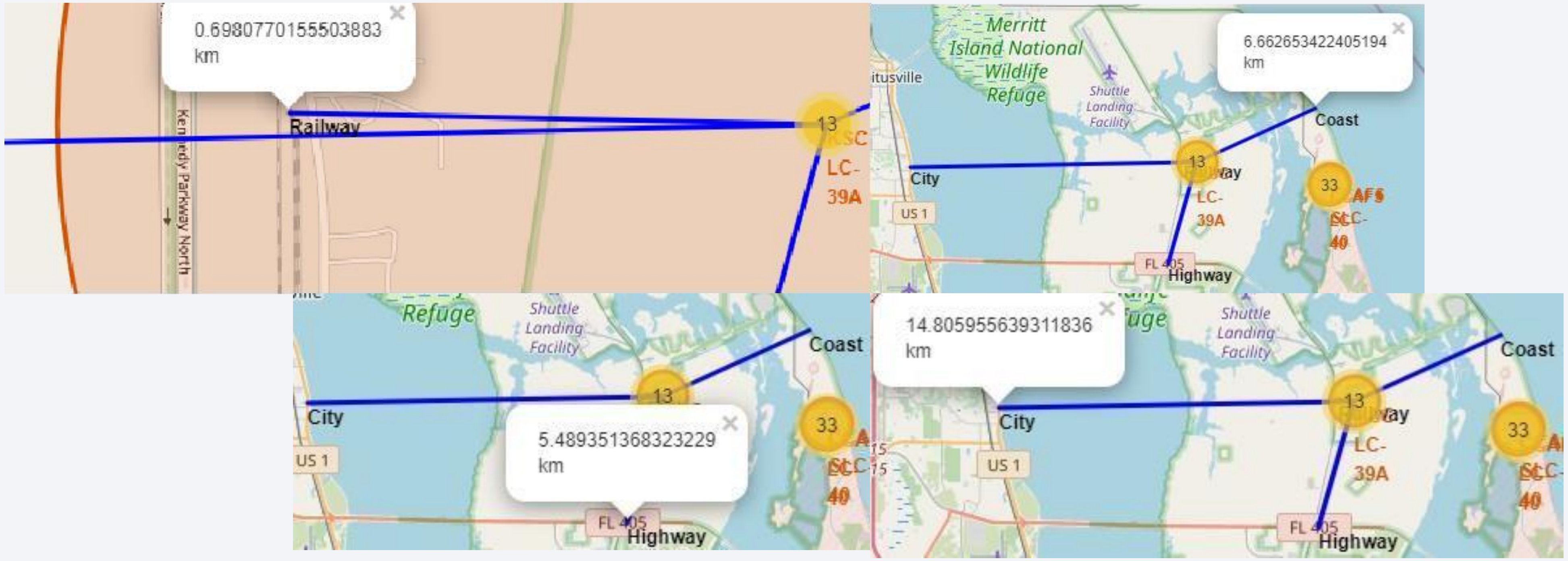
- The left map shows all launch sites relative US map. The right map shows the two Florida launch sites since they are very close to each other. All launch sites are near the ocean.

Launch Outcomes on Folium Map


CA 135
CA 1
US 101
154
Vandenberg Space Force Base
211
Lompoc Airport
Lompoc
CA 246
CA 1
Jack and Laura Dangermond Preserve

- Clusters on Folium map can be clicked on to display each successful landing (green icon) and failed
- landing (red icon). In this example VAFB SLC-4E shows 4 successful landings and 6 failed landings.

Insights on U.S. Launch Site Distribution

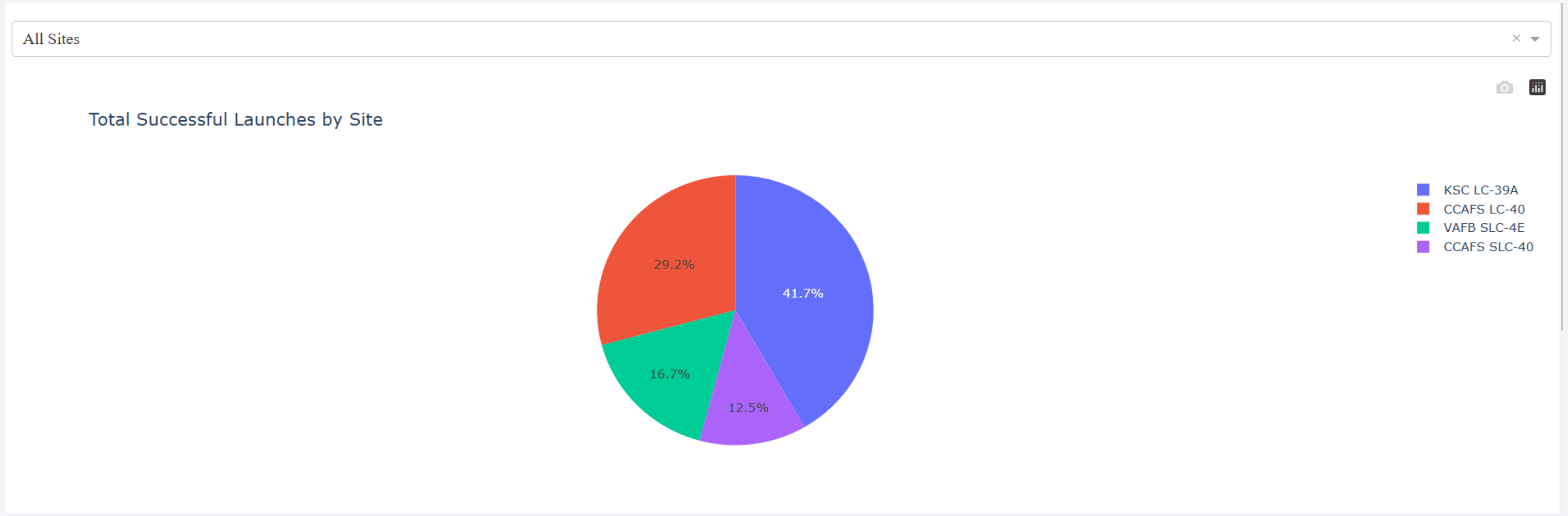


Using KSC LC-39A as an example, launch sites are very close to railways for large part and supply transportation. Launch sites are close to highways for human and supply transport. Launch sites are also close to coasts and relatively far from cities so that launch failures can land in the sea to avoid rockets falling on densely populated areas.

Section 4

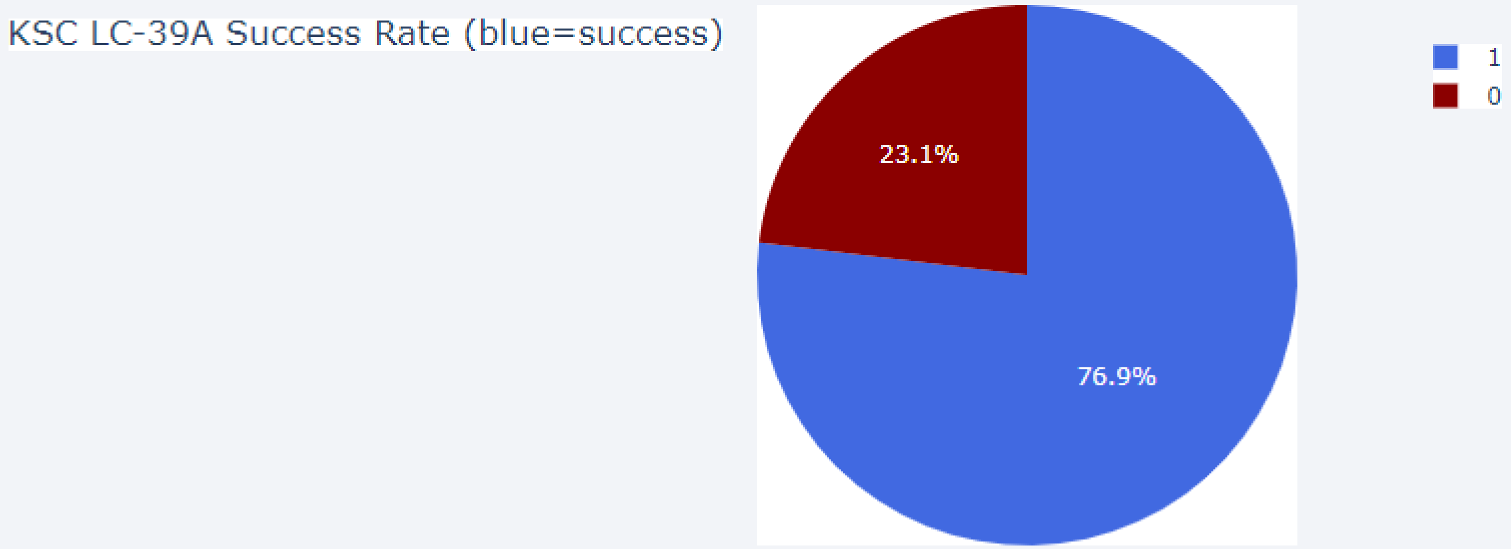
Build a Dashboard with Plotly Dash

Total Successful Launches by Site



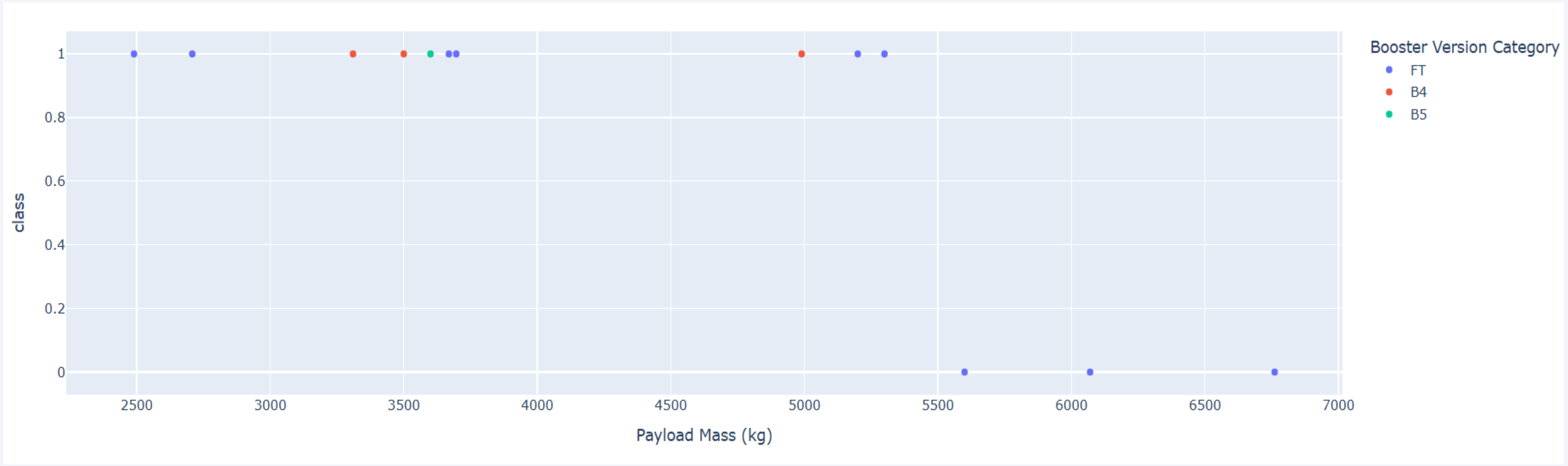
This is the distribution of successful landings across all launch sites. CCAFS LC-40 is the old name of CCAFS SLC-40 so CCAFS and KSC have the same amount of successful landings, but a majority of the successful landings where performed before the name change. VAFB has the smallest share of successful landings. This may be due to smaller sample and increase in difficulty of launching in the west coast.

Launch Success and Failure Rates for CCAFS LC-40



KSC LC-39A has the highest success rate with 10 successful landings and 3 failed landings.

Correlation between Payload Mass and Launch Success for All Sites

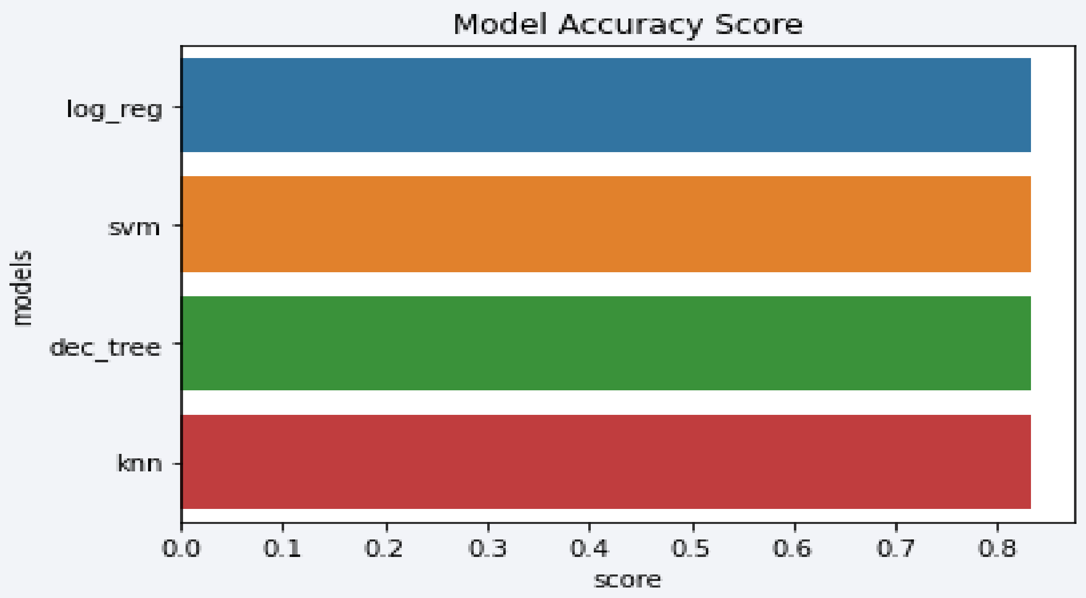


Plotly dashboard has a Payload range selector. However, this is set from 0-10000 instead of the max Payload of 15600. Class indicates 1 for successful landing and 0 for failure. Scatter plot also accounts for booster version category in color and number of launches in point size. In this particular range of 0-6000, interestingly there are two failed landings with payloads of zero kg.

Section 5

Predictive Analysis (Classification)

Classification Accuracy



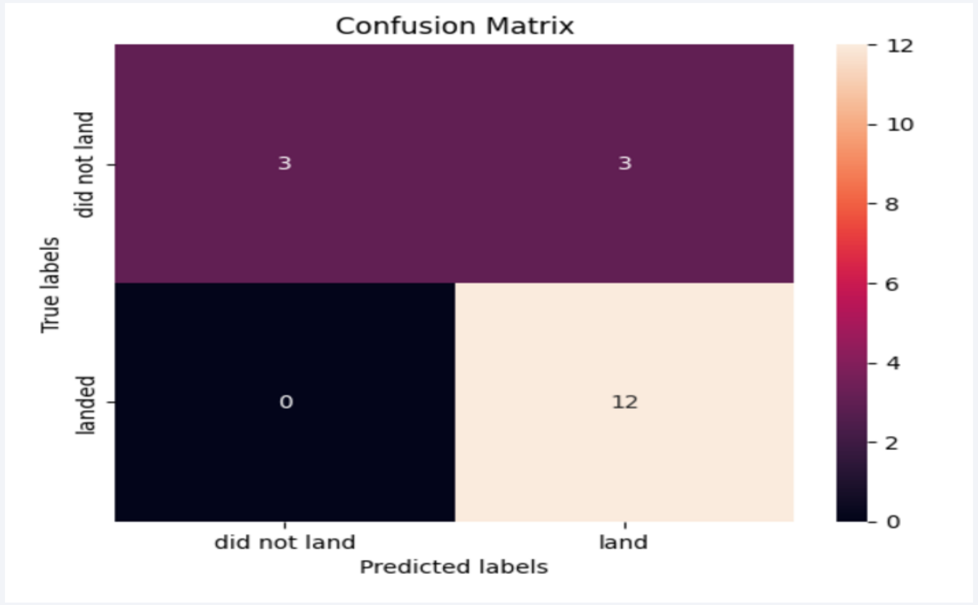
All models had virtually the same accuracy on the test set at 83.33% accuracy. It should be noted that test size is small at only sample size of 18.

This can cause large variance in accuracy results, such as those in Decision Tree Classifier model in repeated runs.

We likely need more data to determine the best model.

Confusion Matrix

- The confusion matrix provides a detailed evaluation of the model's performance by comparing the true labels with the predicted labels. The model shows:
- A high number of true positives (12), indicating that it is very accurate in predicting successful landings.
- A reasonable number of true negatives (3), showing it correctly identifies non-landings.
- A few false positives (3), where the model incorrectly predicted a landing.
- No false negatives (0), indicating that the model does not miss any actual landings.



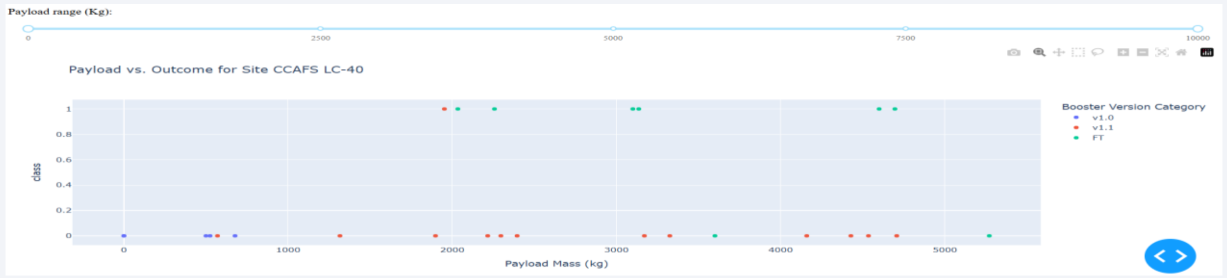
Conclusions

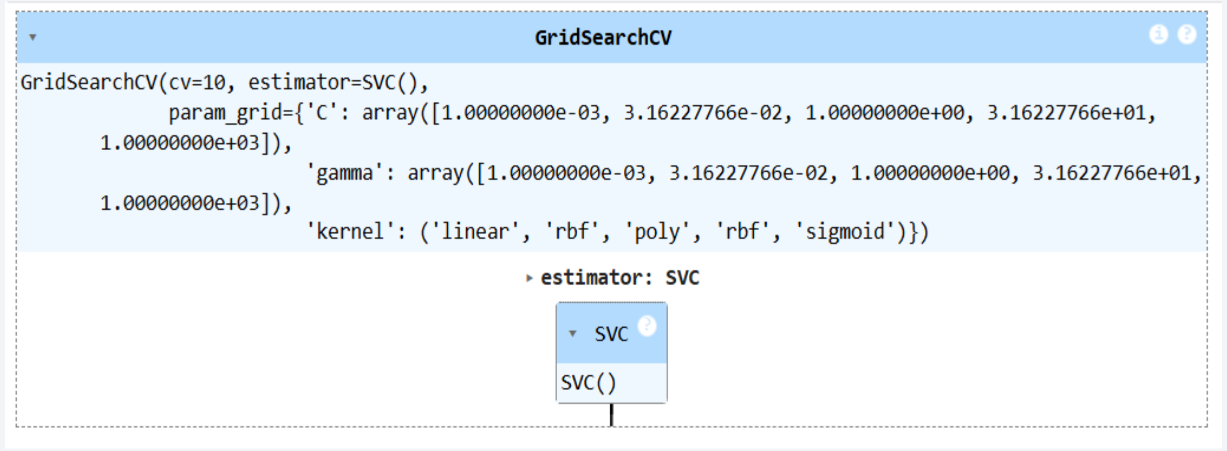
- Our task: to develop a machine learning model for Space Y who wants to bid against SpaceX
- The goal of model is to predict when Stage 1 will successfully land to save ~$100 million USD
- Used data from a public SpaceX API and web scraping SpaceX Wikipedia page
- Created data labels and stored data into a DB2 SQL database
- Created a dashboard for visualization
- We created a machine learning model with an accuracy of 83%
- Allon Mask of SpaceY can use this model to predict with relatively high accuracy whether a launch will have a successful Stage 1 landing before launch to determine whether the launch should be made or not
- If possible more data should be collected to better determine the best machine learning model and improve accuracy

Appendix

GitHub repository url:

[Gouthamudayagiri/Applied-Data-Science-Capstone]





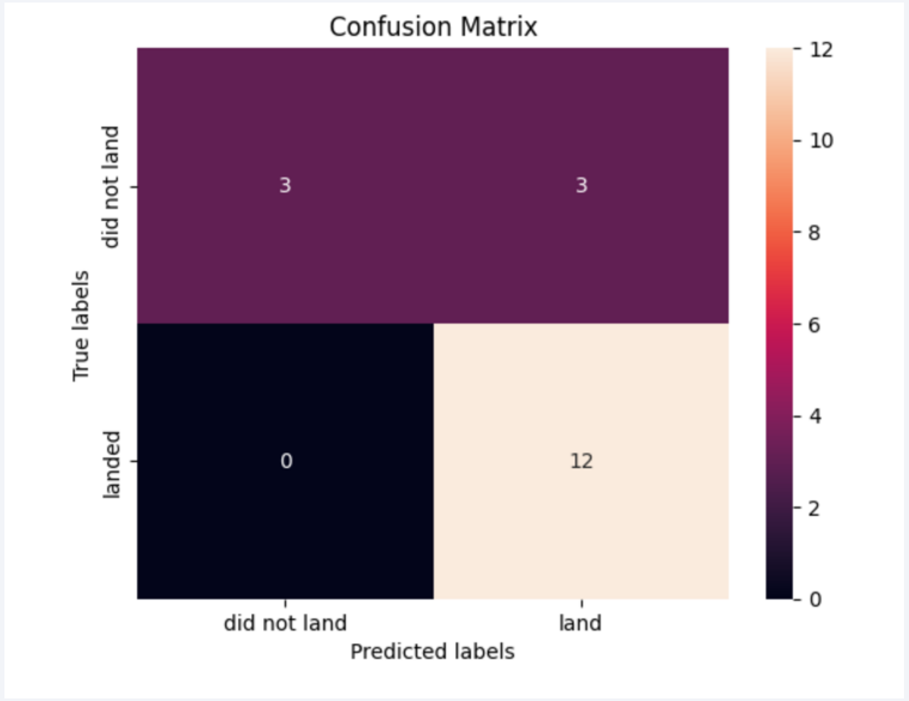
GridSearchCV

```
GridSearchCV(cv=10, estimator=SVC(),
             param_grid={'C': array([1.00000000e-03, 3.16227766e-02, 1.00000000e+00, 3.16227766e+01,
       1.00000000e+03]),
                         'gamma': array([1.00000000e-03, 3.16227766e-02, 1.00000000e+00, 3.16227766e+01,
       1.00000000e+03]),
                         'kernel': ('linear', 'rbf', 'poly', 'rbf', 'sigmoid')})
```

▸ **estimator: SVC**

▾ SVC

SVC()



Thank you!