IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Divya Damodaran

December 16, 2025

Outline

Executive Summary

Summary of methodologies

Data Collection

Data Wrangling

Exploratory Data Analysis with Data Visualization

Exploratory Data Analysis with SQL

Interactive Map building using Folium

Dashboard Build with Plotly Dash

Predictive Analysis(Classification)

Summary of all results

Exploratory Data Analysis Results

Interactive Analytics Demo Screenshots

Predictive Analysis Results

Introduction

Project background and context

Main focus of this project is to predict the successful landing of the Falcon 9 first stage. By accurately predicting landing success its easy to estimate launch costs and provide valuable insights for companies bidding against SpaceX.

Problems you want to find answers

How do variables such as payload mass, launch site, number of flights, and orbit type affect the success of the first stage landing?

How can we accurately predict the landing outcome using machine learning models

Which machine learning model performs best in predicting the landing success

Does the rate of success increase over the years?

Section 1

Methodology

Methodology

Data Collection: First stage of SpaceX Falcon9 was collected mainly from SpaceX API and Web Scraping from Wikipedia Article

Data Wrangling: This process involved handling missing values, standardizing format.

Key features were extracted and new features engineered to enrich the dataset

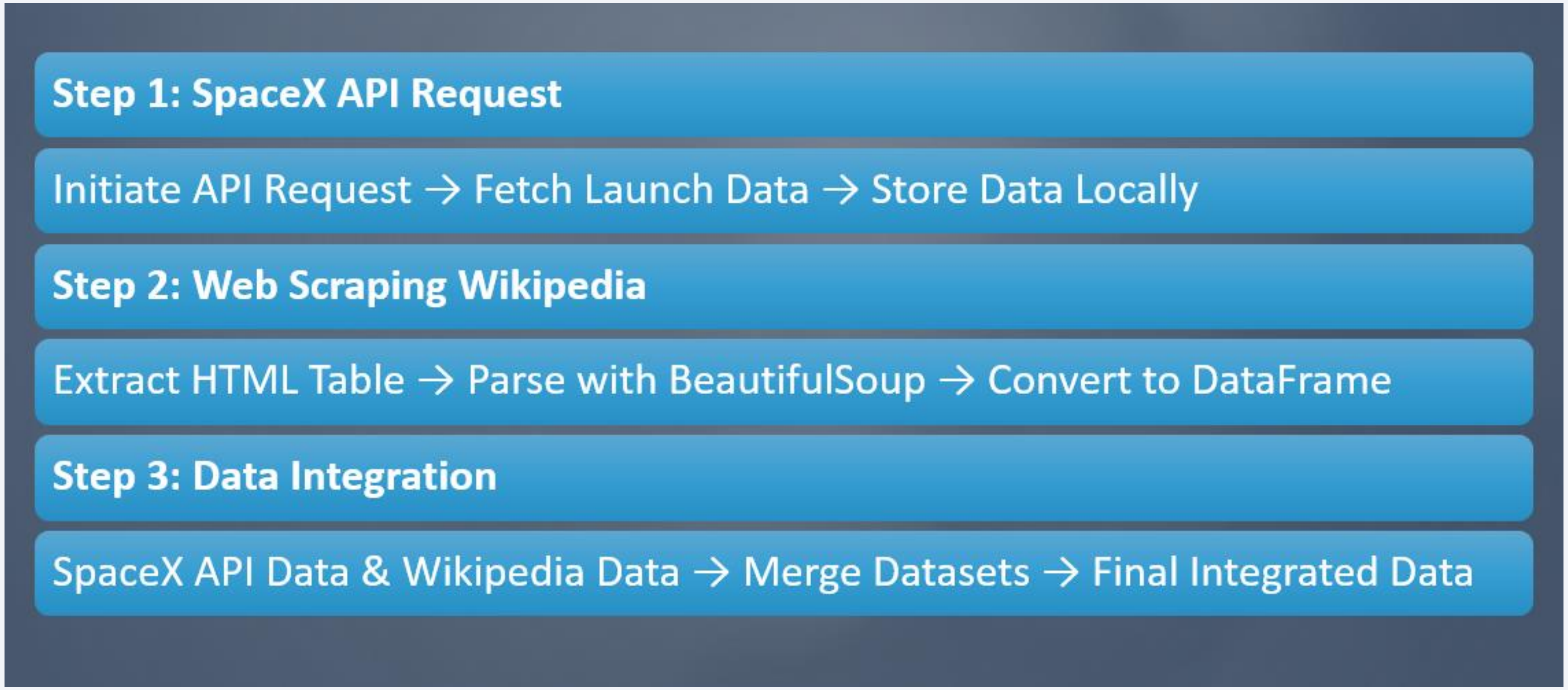
Exploratory Data Analysis(EDA) : This process was performed using Visualization and SQL

Interactive Visual Analytics: Visual diagrams and dashboard were created using Folium and Plotly DASH

Predictive Analysis: Analysis was performed using Classification Algorithms on ML models

GITHUB URL: **DDIVYA2025/IBM-APPLIED-DATA-SCIENCE-CAPSTONE-PROJECT**

Data Collection



Step 1: SpaceX API Request

Initiate API Request → Fetch Launch Data → Store Data Locally

Step 2: Web Scraping Wikipedia

Extract HTML Table → Parse with BeautifulSoup → Convert to DataFrame

Step 3: Data Integration

SpaceX API Data & Wikipedia Data → Merge Datasets → Final Integrated Data

Data Collection – SpaceX API

Step 1: Initiate API Request

- Use Python requests library to connect to SpaceX API.
- https://api.spacexdata.com/v4/launches (Source URL)

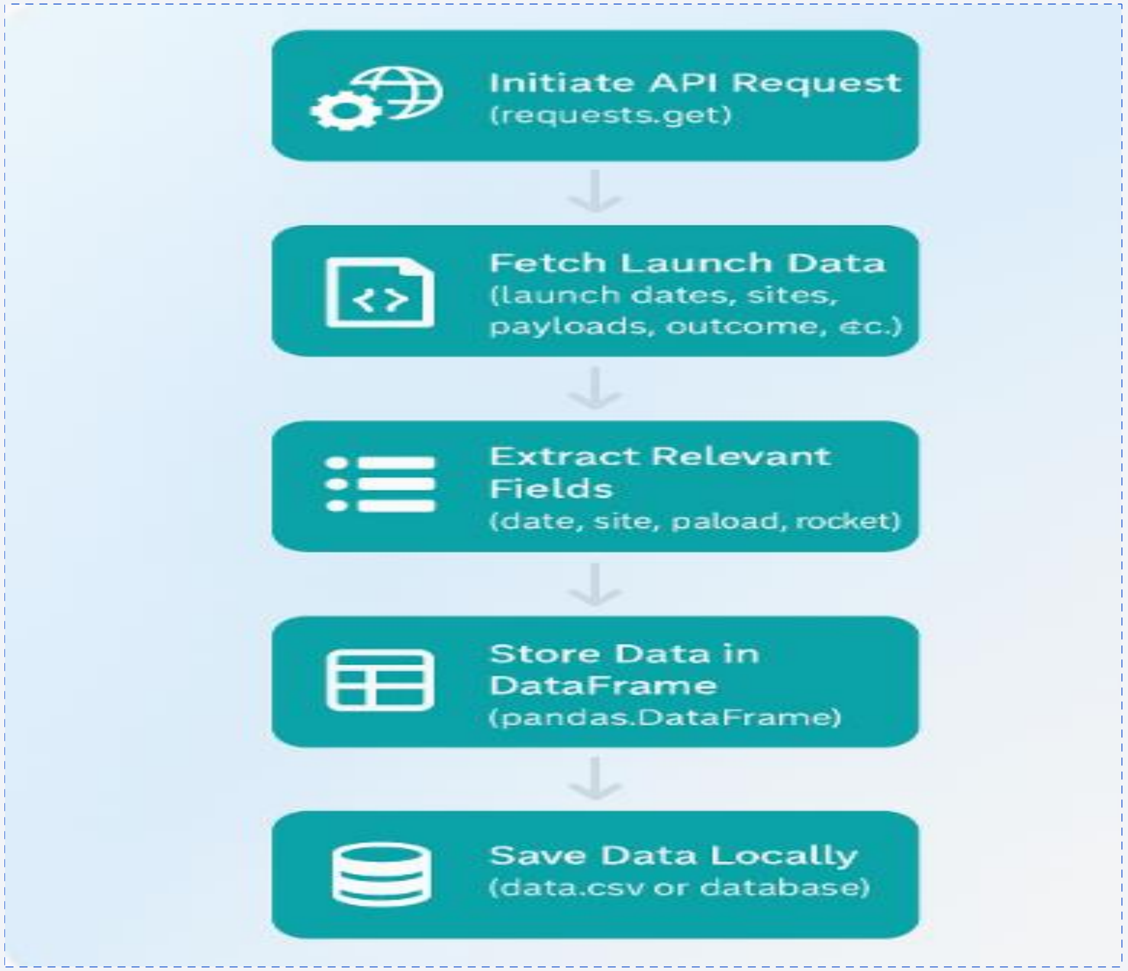
Step 2: Initiate API Request

- Convert JSON response to Python dictionary.Step 1: Initiate API Request
- Extract key fields like launch date/Launch Site /Payload Mass/Rockey Type,/outcome

Step 3: Initiate API Request

- Save data into a pandas DataFrame.
- Export as CSV or store in a database for further analysis.

GitHub URL : 1.Data Collection API.ipynb



Data Collection – Web Scraping

Step1:Initiate Webscraping

➢ Use Python’s request library to fetch HTML content from Wikipedia page

➢ Source url: https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches

Step 2: Parse HTML page info

➢ Use BeautifulSoup to parse the HTML content.

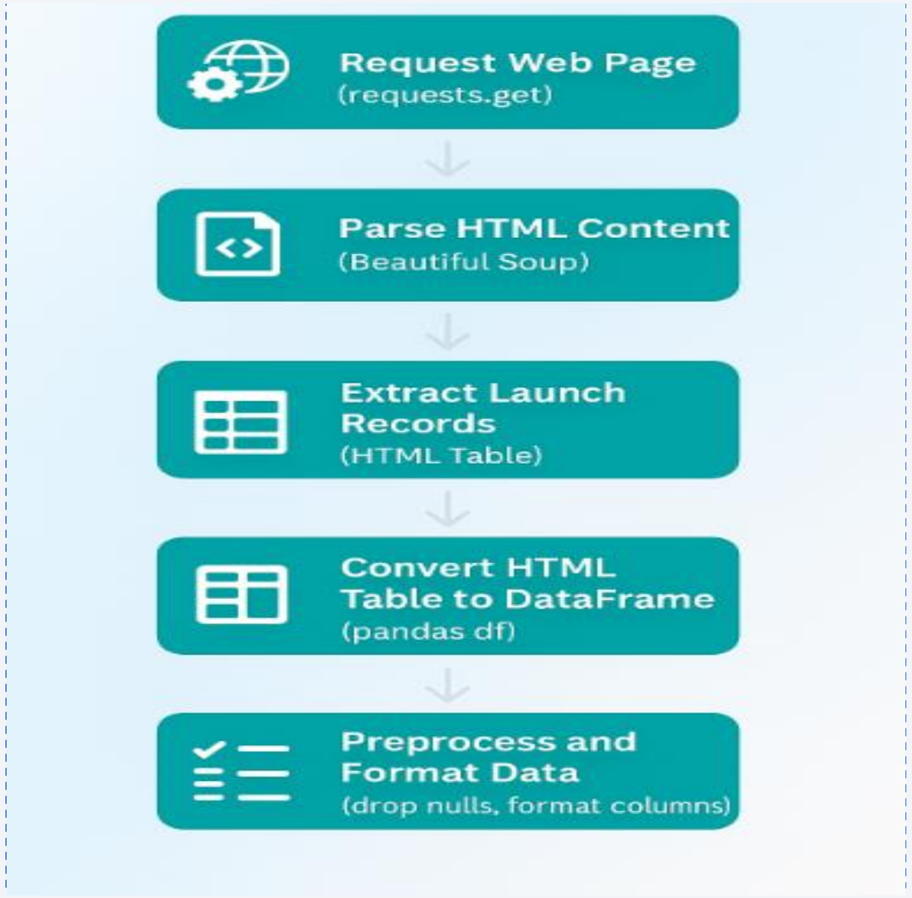
➢ Extract the HTML table containing Falcon 9 launch records.

Step 3: Convert to Pandas DataFrame

➢ Convert the extracted HTML table into a pandas DataFrame.

➢ Clean and format the DataFrame, ensuring data consistency.

GitHub URL:2. Webscraping.ipynb



Data Wrangling

Step1:Data Cleaning

- ➢ Identify and remove missing values from the dataset

Step2: Data Transformation

- ➢ Convert data types to appropriate format
- ➢ Create new features from existing data

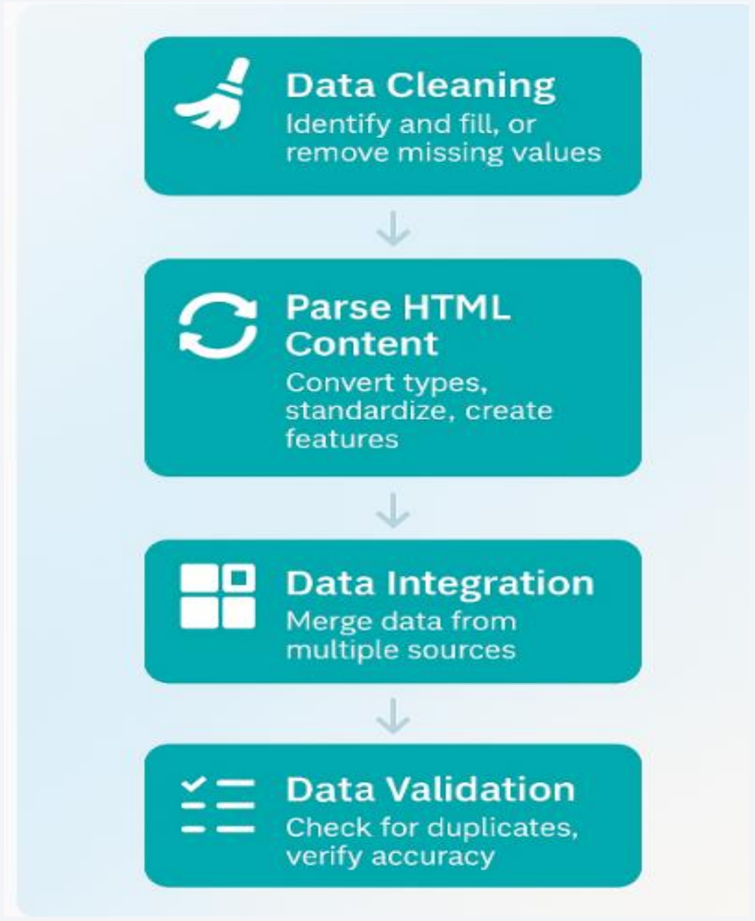
Step3:Data Integration

- ➢ Merge datasets collected from different sources like API
- ➢ Web scraping etc into single dataset

Step4:Data Validation

- ➢ Check for duplicates and remove if needed
- ➢ Verify consistency and accuracy

GitHub URL:3. Data wrangling.ipynb


Data Cleaning
Identify and fill, or remove missing values
Parse HTML Content
Convert types, standardize, create features
Data Integration
Merge data from multiple sources
Data Validation
Check for duplicates, verify accuracy

EDA with Data Visualization

- These are the charts used in this project
- **1. Histograms**

➢ Purpose: Used to visualize the distribution of numerical variables such as launch success rates, payload mass, and flight number.

➢ Why: Helps in understanding the spread and central tendency of the data, identifying outliers, and assessing data skewness.

- **2. Bar Charts**

➢ Purpose: Used to compare categorical variables such as launch outcomes (success/failure) across different categories like launch sites or rocket types.

➢ Why: Provides a clear comparison of frequencies or proportions within categorical data, highlighting patterns or trends.

- **3. Line Charts**

➢ Purpose: Used to track trends over time, such as the success rate of Falcon 9 launches across different years.

➢ Why: Reveals temporal patterns and helps in understanding performance trends or changes over specific periods.

- GitHub URL:[4. EDA with Data Visualization.ipynb]

4.Scatter Plots

➢ Purpose: Used to explore relationships between two numerical variables, such as payload mass vs. launch success.

➢ Why: Identifies correlations or dependencies between variables, visualizing how one variable changes concerning another.

5. Heatmaps

➢ Purpose: Used to visualize correlation matrices between multiple numerical variables.

➢ Why: Helps in identifying strong correlations (positive or negative) between variables, aiding feature selection or understanding multicollinearity.

6. Box Plots

➢ Purpose: Used to display the distribution of numerical data through their quartiles.

➢ Why: Visualizes the spread and skewness of data, highlighting outliers and comparing distributions across different categories.

EDA with SQL

- **Aggregate Queries**
  - ➢ Calculated total number of launches.
  - ➢ Counted successful and failed launches.
  - ➢ Calculated success rates by launch site and rocket type.
- **Join Queries**
  - ➢ Joined tables to link launch records with additional data (e.g., rocket details).
  - ➢ Combined datasets for comprehensive analysis.
- **Filtering Queries**
  - ➢ Filtered data to focus on specific launch outcomes (success/failure).
  - ➢ Applied conditions to extract launches based on criteria like launch date or rocket configuration.
- **Sorting Queries**
  - ➢ Sorted data to identify trends or outliers.
  - ➢ Ordered launches by date or success rate for analysis.
- **Subqueries**
  - ➢ Nested queries to calculate derived metrics (e.g., average payload mass per launch site).
  - ➢ Subqueries used to perform detailed analysis within larger datasets.

GitHub URL:5. EDA with SQL.ipynb

Build an Interactive Map with Folium

- These Objects Used
- **Markers**
  - ➢ Indicate exact launch site locations
  - ➢ Each marker represents a real geographic point where SpaceX launches occurred
- **Circles**
  - ➢ Drawn around launch sites
  - ➢ Represent proximity or surrounding zones that may affect operations
- **Lines**
  - ➢ Connect launch sites to nearby or related locations
  - ➢ Provide spatial context between different points of interest
  - ➢ Explain why you added those objects

Reasons for Adding Them
Markers help pinpoint and identify launch locations geographically

Circles visualize potential impact areas, safety zones, or operational boundaries

Lines show spatial relationships, connections, or dependencies between locations

GitHub 6. Interactive Visual Analytics with Folium.ipynb

Build a Dashboard with Plotly Dash

- Dash
- Functions from Dash are used to generate an interactive website.
- Interactions added - Users can toggle inputs using:
    - Launch site dropdown menu
    - Range slider for Payload mass
- The interactive site uses:
    - A pie chart
    - A scatter plot
- The site displays:
    - The total number of successful launches from each launch site
    - The correlation between payload mass and mission outcome (success or failure) for each launch site
- GitHub URL:7. spacex dash app.py

Predictive Analysis (Classification)

Machine Learning Prediction Phase Steps:

- Standardizing the data
- Splitting the data into training and test sets
- Creating machine learning models, which include:

➢ Logistic Regression

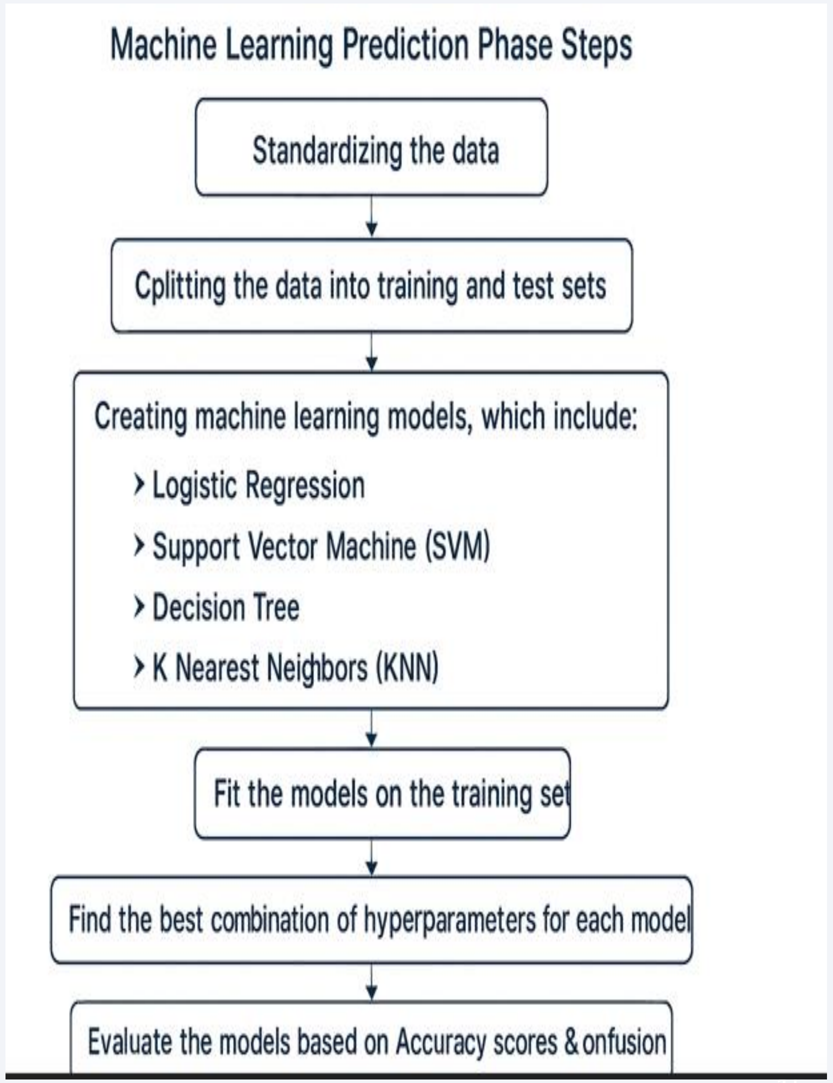
➢ Support Vector Machine (SVM)

➢ Decision Tree

➢ K Nearest Neighbors (KNN)

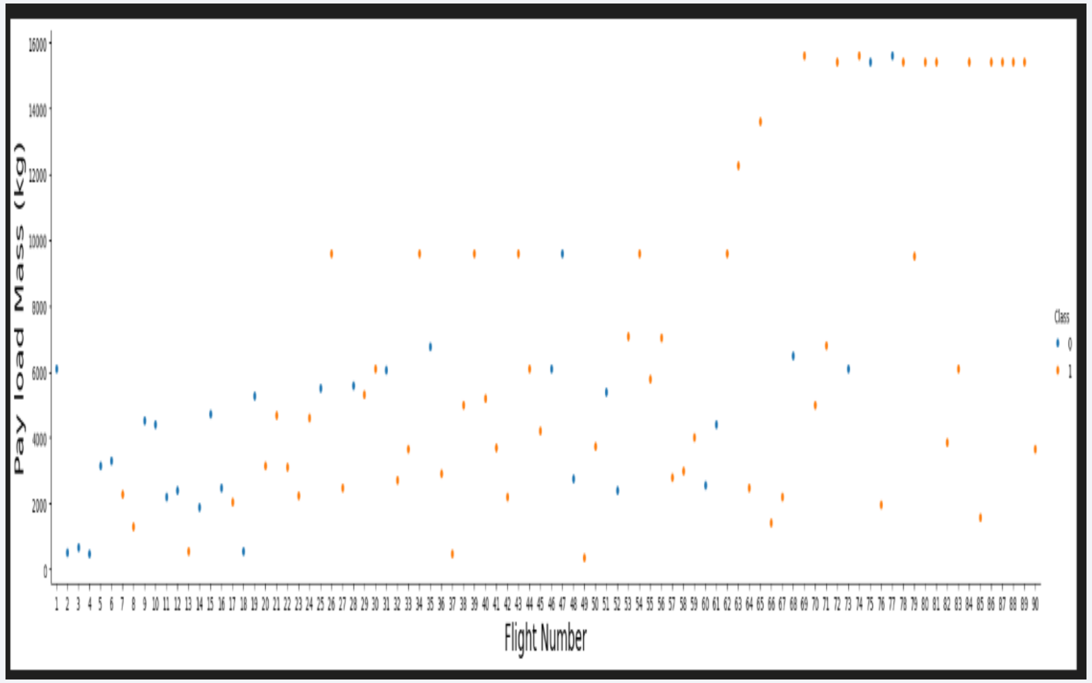
- Fit the models on the training set
- Find the best combination of hyperparameters for each model
- Evaluate the models based on Accuracy scores & Confusion matrix

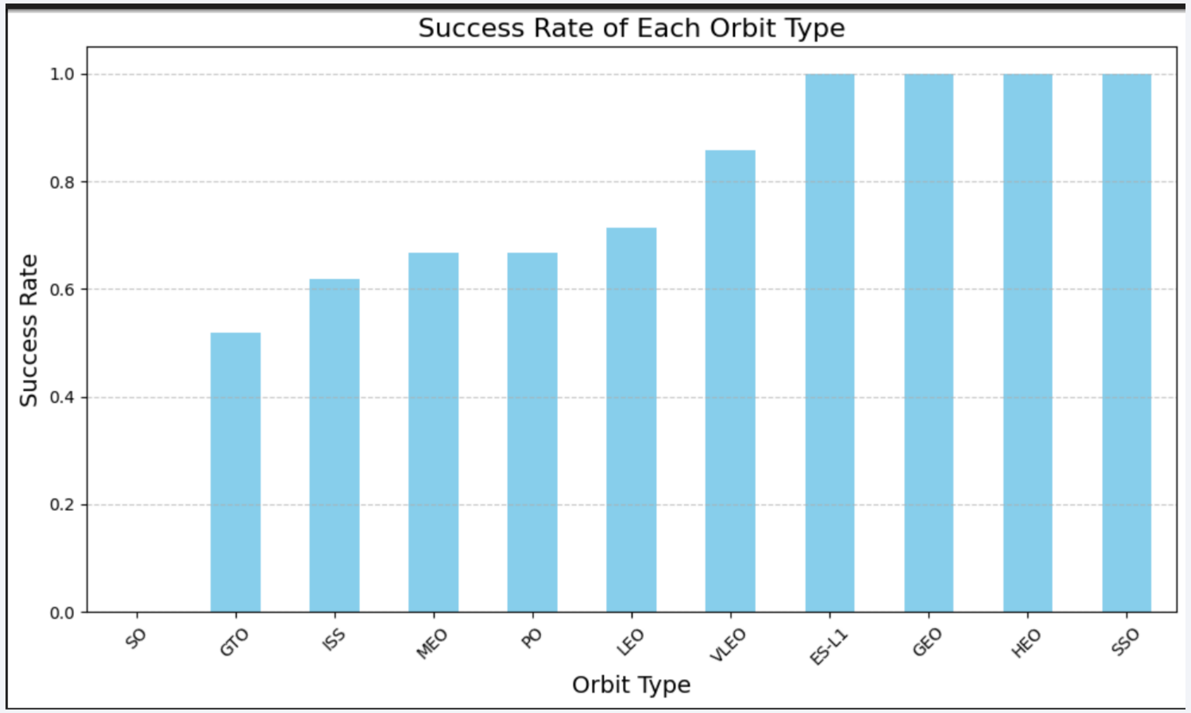
GitHub URL:8. Machine Learning Prediction.ipynb


Machine Learning Prediction Phase Steps
Standardizing the data
Cplitting the data into training and test sets
Creating machine learning models, which include:
› Logistic Regression
› Support Vector Machine (SVM)
› Decision Tree
› K Nearest Neighbors (KNN)
Fit the models on the training set
Find the best combination of hyperparameters for each model
Evaluate the models based on Accuracy scores & onfusion

Results

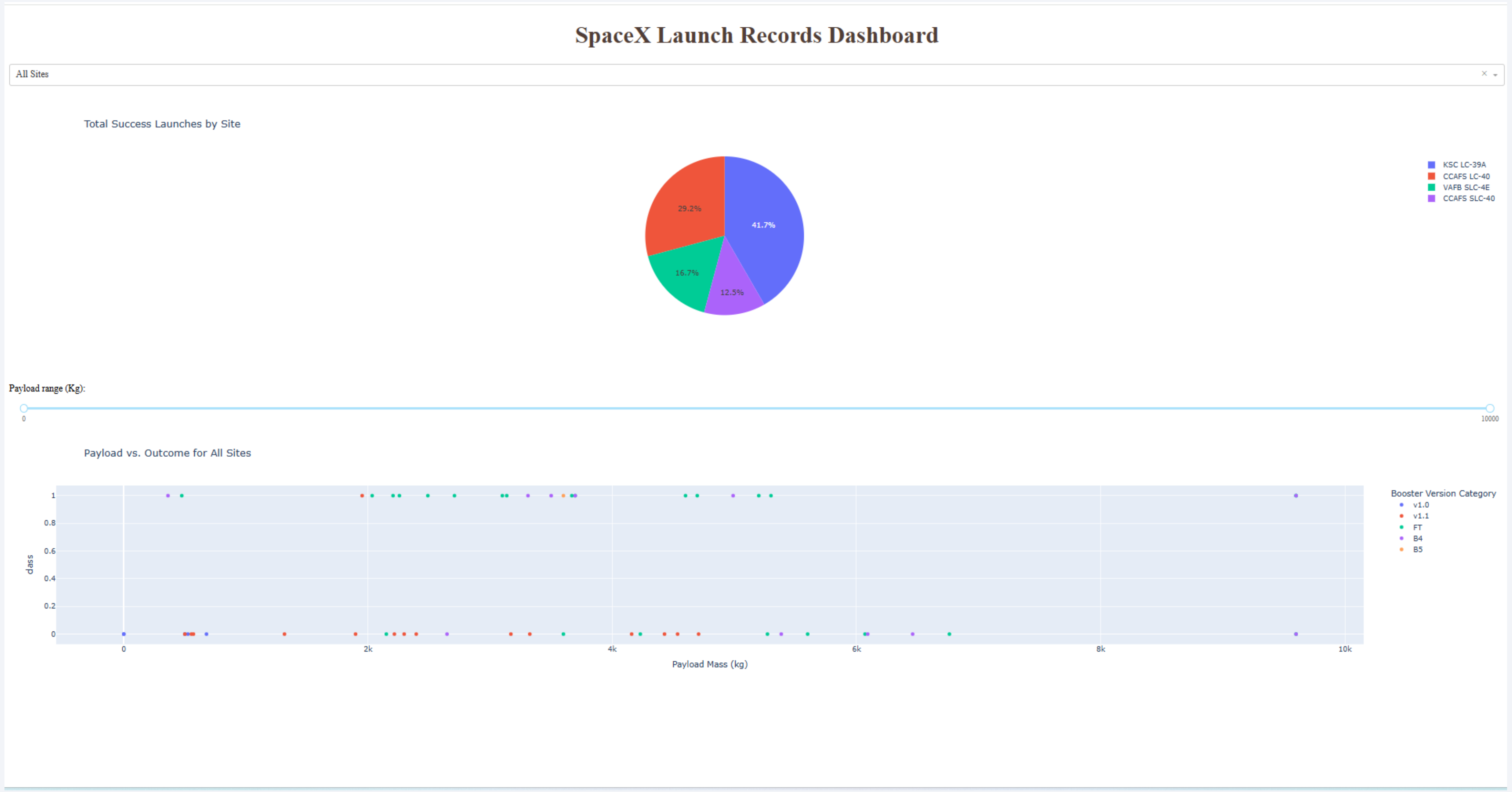
Exploratory data analysis results





Results cont’d

Interactive analytics demo in screenshots



Results cont'd

Predictive analysis results:

- Libraries or modules used:

  pandas, numpy, matplotlib.pyplot, seaborn, sklearn
- The machine learning prediction phase include the following steps:

➢ Standardizing the data using the preprocessing.StandardScaler() function from sklearn

➢ Splitting the data into training and test data using the train_test_split function from sklearn.model_selection

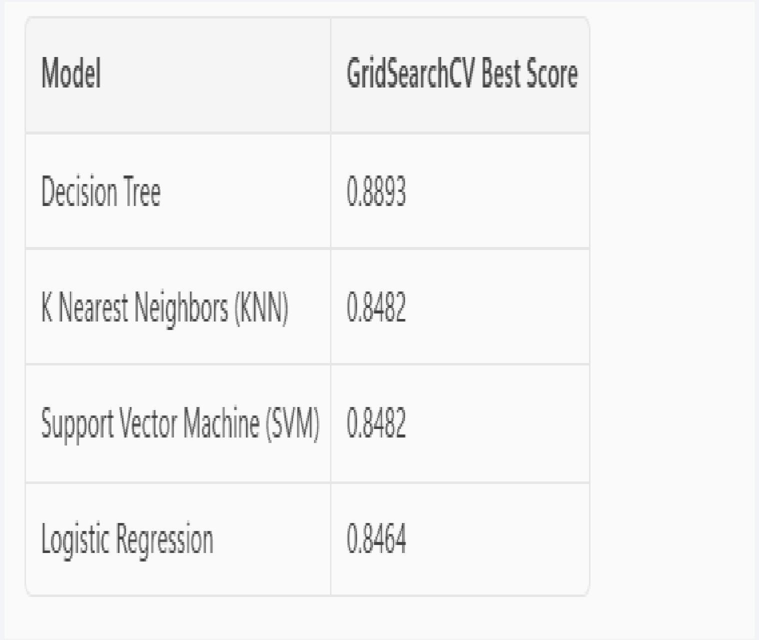
Creating machine learning models, which include:

- Logistic regression using LogisticRegression from sklearn.linear_model
- Support vector machine (SVM) using SVC from sklearn.svm
- Decision tree using DecisionTreeClassifier from sklearn.tree
- K nearest neighbors (KNN) using KNeighborsClassifier from sklearn.neighbors
- Fit the models on the training set

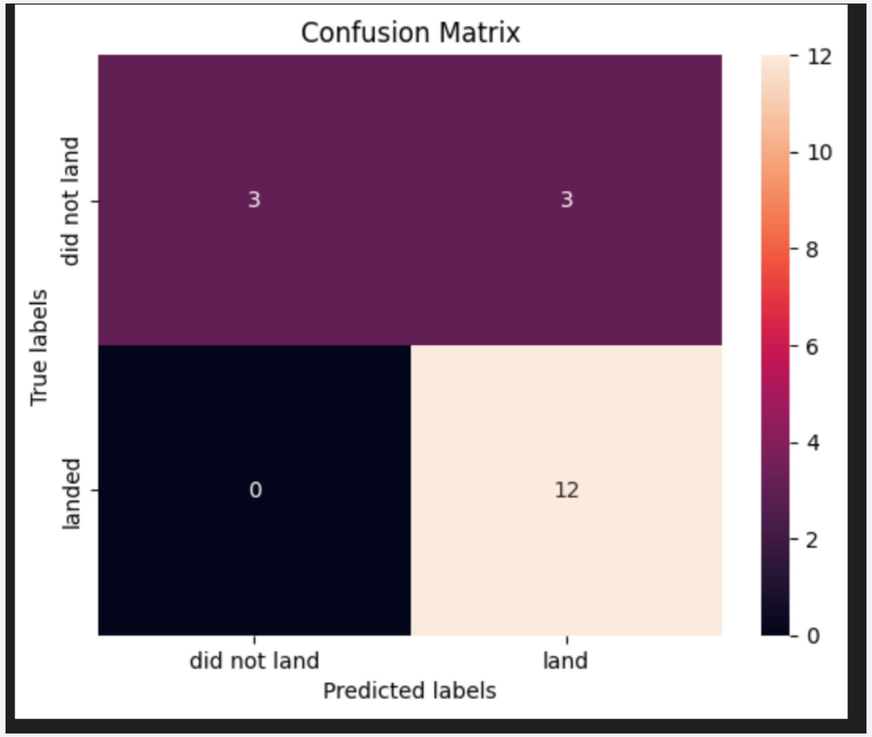
Results cont’d

Predictive analysis results:

- Find the best combination of hyperparameters for each model using GridSearchCV from sklearn.model_selection
- Evaluate the models based on their accuracy scores and confusion matrix using the score() function and confusion_matrix from sklearn.metrics
- Putting the results of all 4 models side by side, we can see that they all share the same accuracy score and confusion matrix when tested on the test set. Based on the GridSearchCV best scores, the models are ranked in the following order with the first being the best and the last one being the worst:



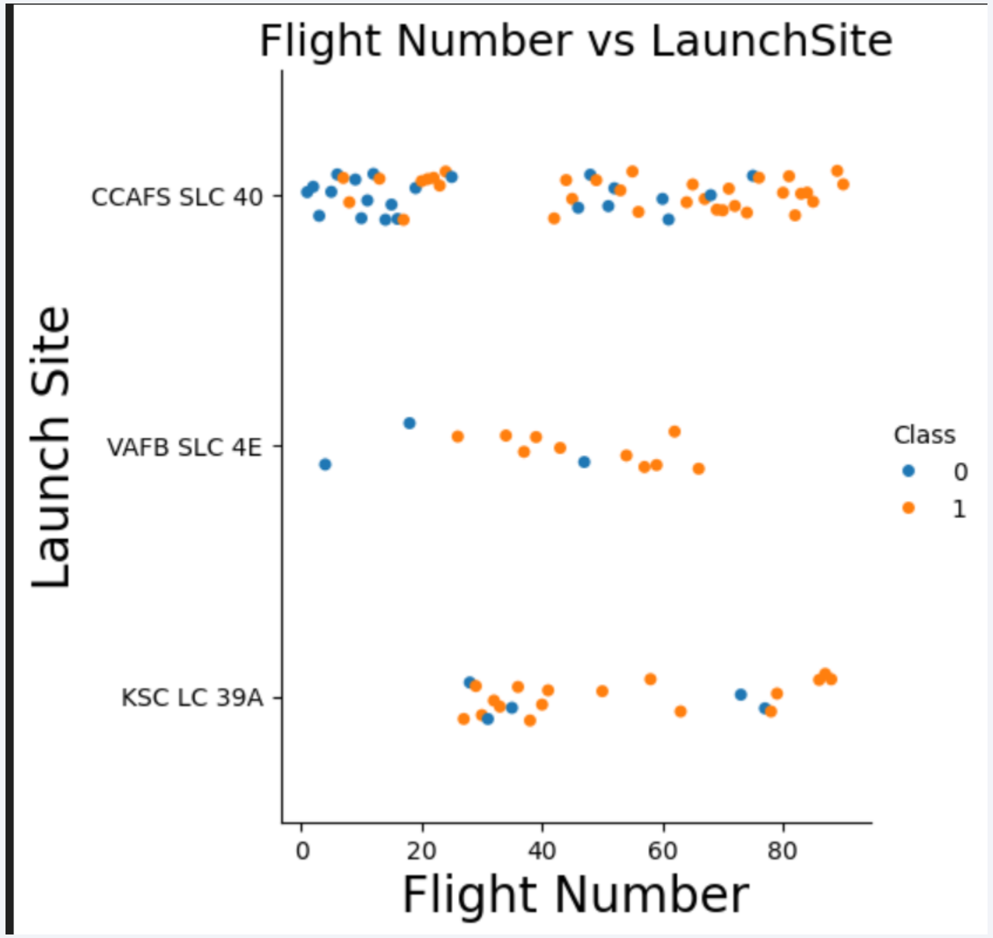
Model	GridSearchCV Best Score
Decision Tree	0.8893
K Nearest Neighbors (KNN)	0.8482
Support Vector Machine (SVM)	0.8482
Logistic Regression	0.8464



Section 2

Insights drawn from EDA

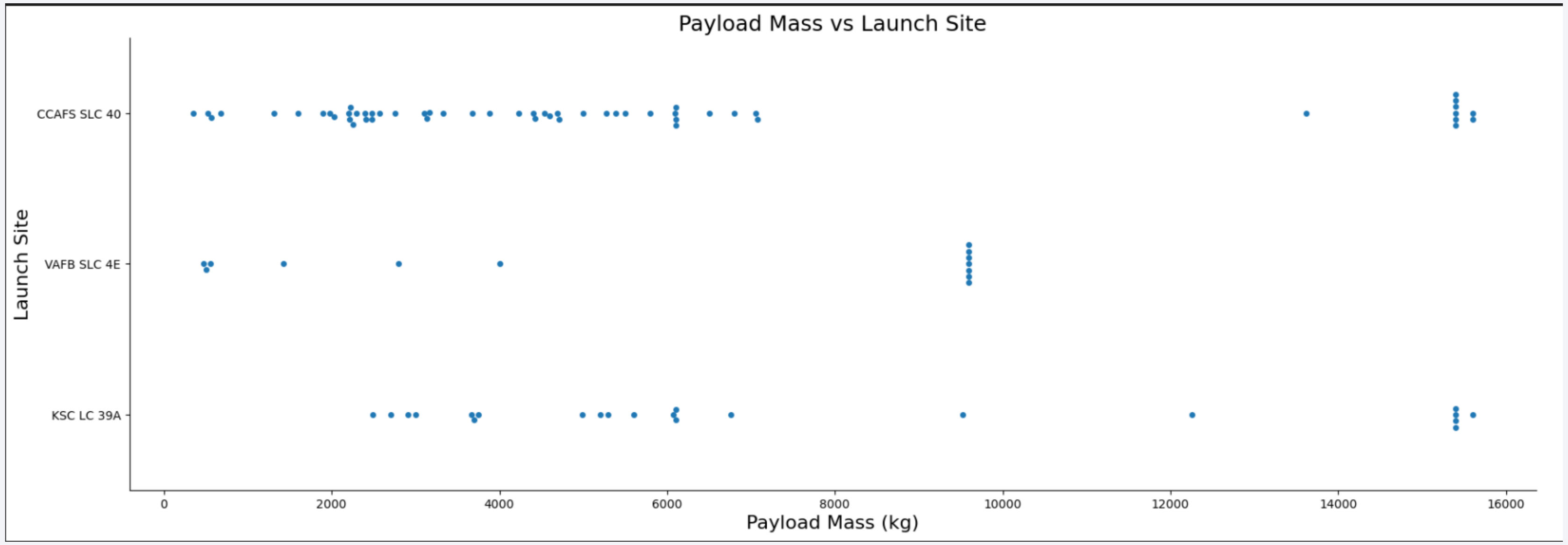
Flight Number vs. Launch Site



Observations

- Success rate (class=1) increases as number of flights increases
- Launch site 'KSC LC 39A' it took around 25 launches before the 1st success

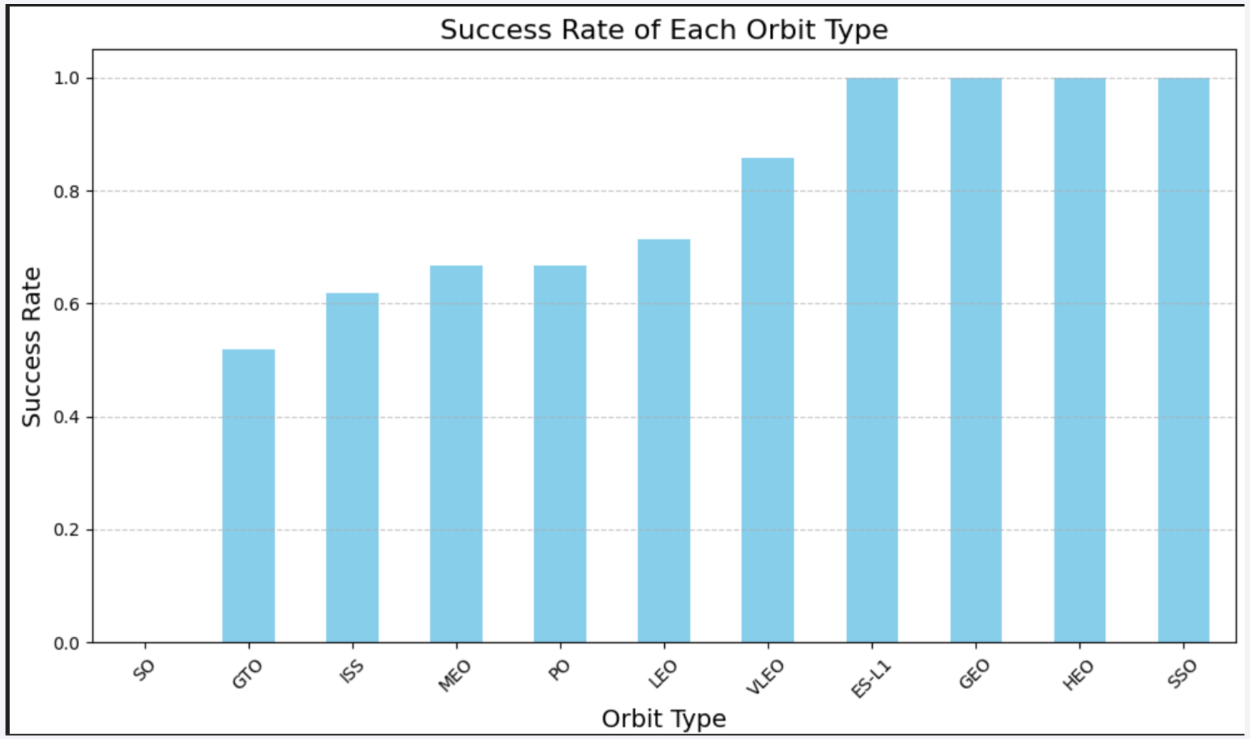
Payload vs. Launch Site



Observations

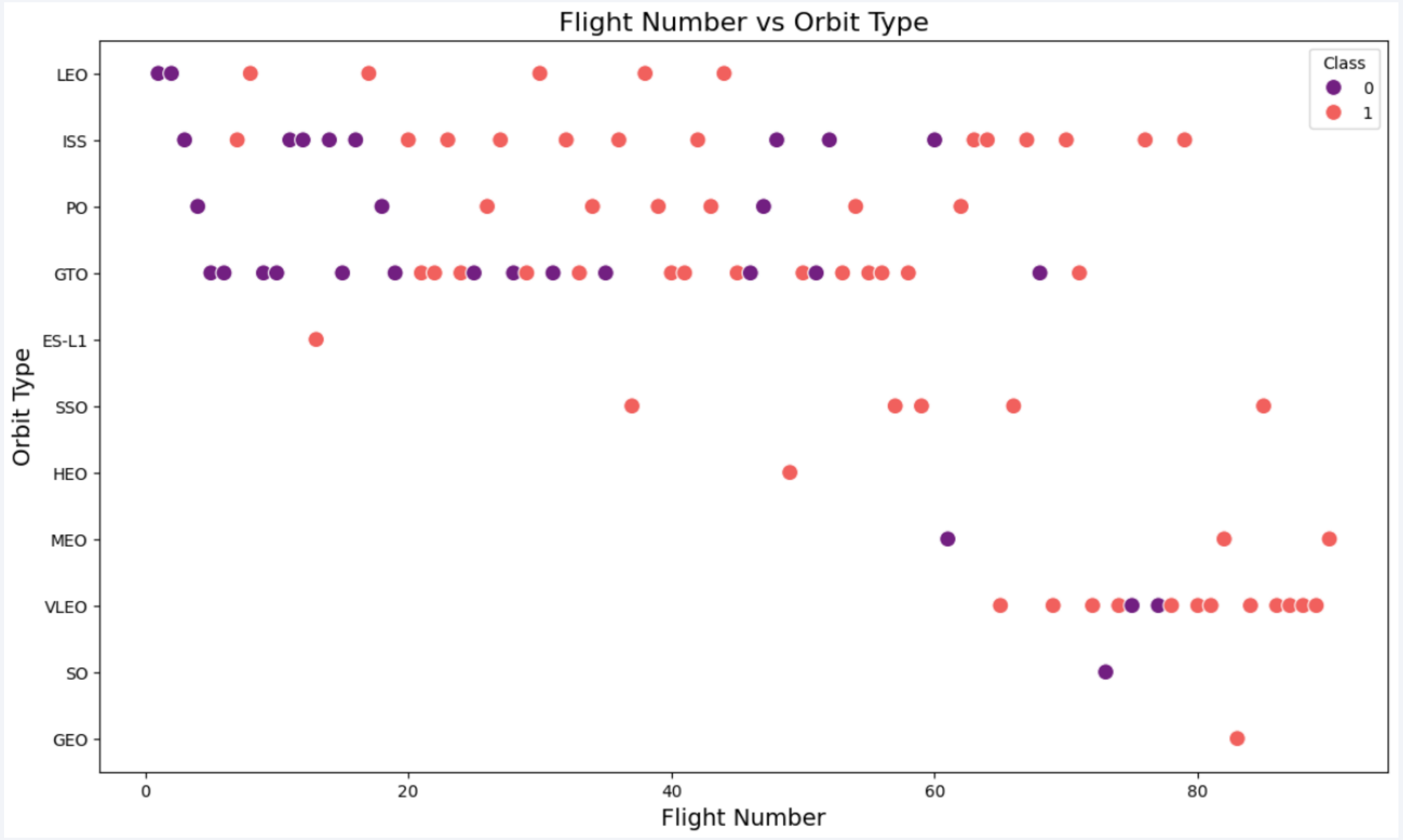
- For ‘VAFB SLC 4E’ there were no rockets launched for payload greater than 10,000kg
- There is no clear correlation /pattern between launch site and payload mass

Success Rate vs. Orbit Type



- Observations
  - Orbit types ES-L1, GEO, HEO, and SSO have high success rates
  - GTO orbit type has the lowest success rate

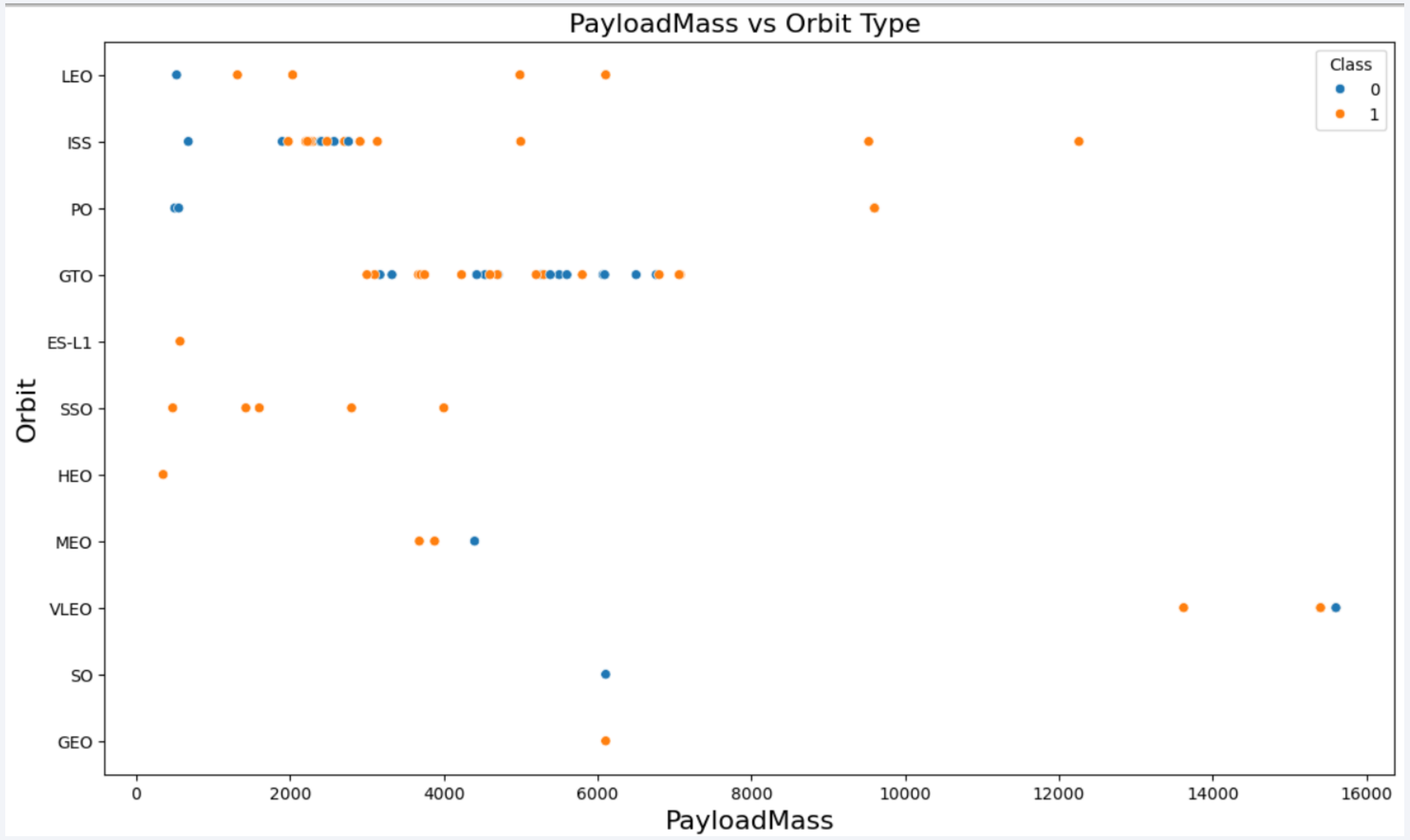
Flight Number vs. Orbit Type



Observations

- Leo orbit success tie to number of flights
- GTO Orbit has no relationship between flight number and success

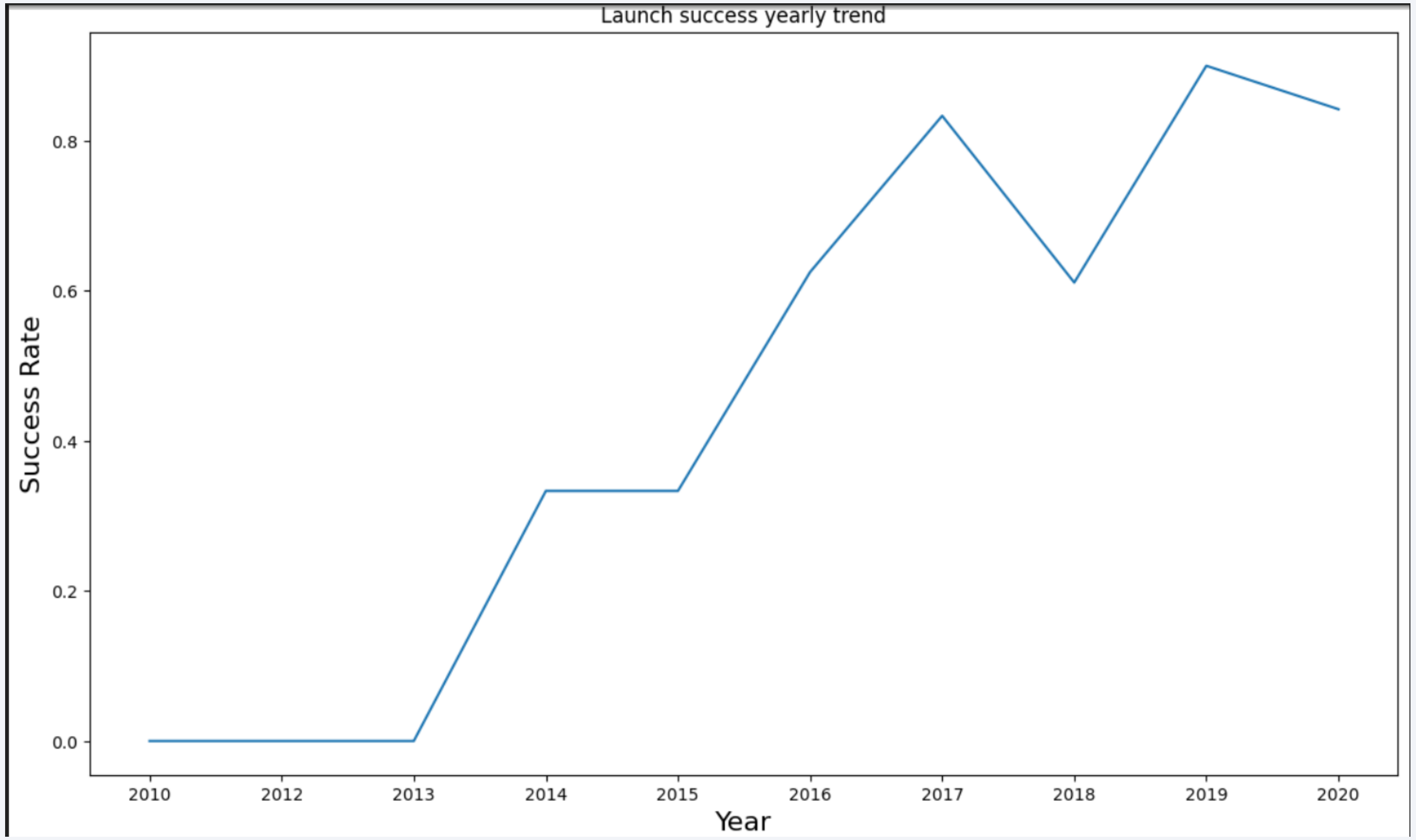
Payload vs. Orbit Type



Observations

- Successful landing/positive landing rate more for Polar, LEO and ISS
- For GTO its difficult to distinguish between successful and unsuccessful landings as both outcomes are present

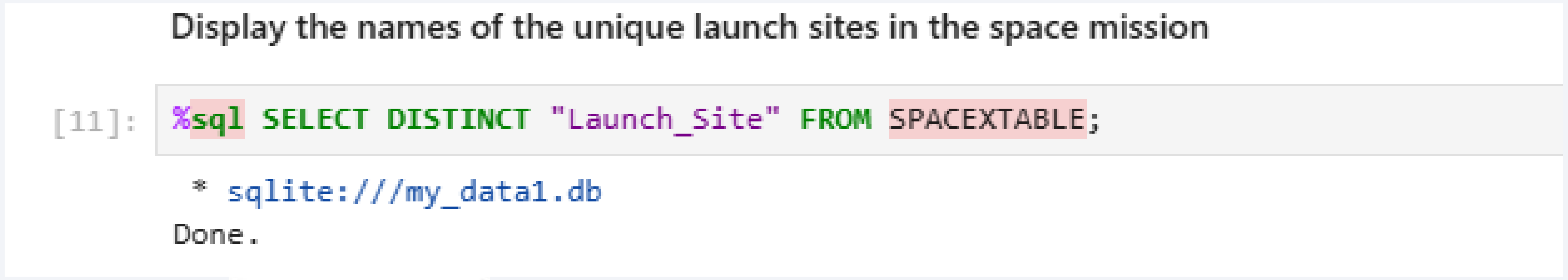
Launch Success Yearly Trend



- Observations
- Success rate keep increasing between 2013 and 2020
- Success rate remained same between 2010 and 2013 /2014 and 2015
- Success rate decreased between 2017 and 2018/2019 and 2020

All Launch Site Names

- Query for launch sites list

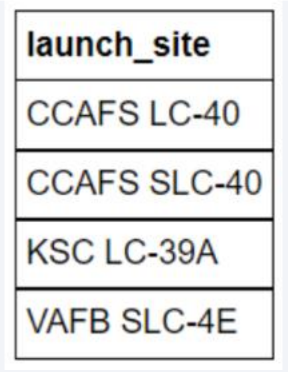


Display the names of the unique launch sites in the space mission

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

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

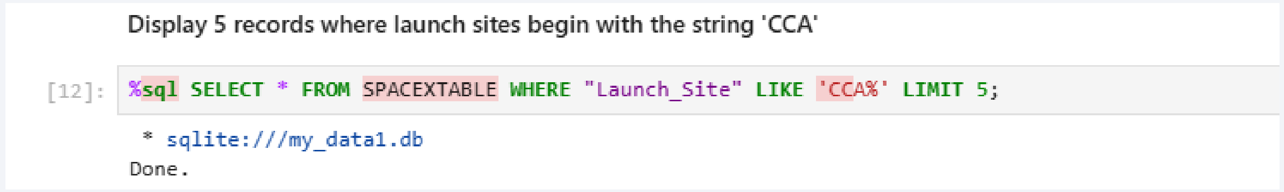
- Result:



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

Launch Site Names Begin with 'CCA'

Query for launch site begins with ‘CCA’

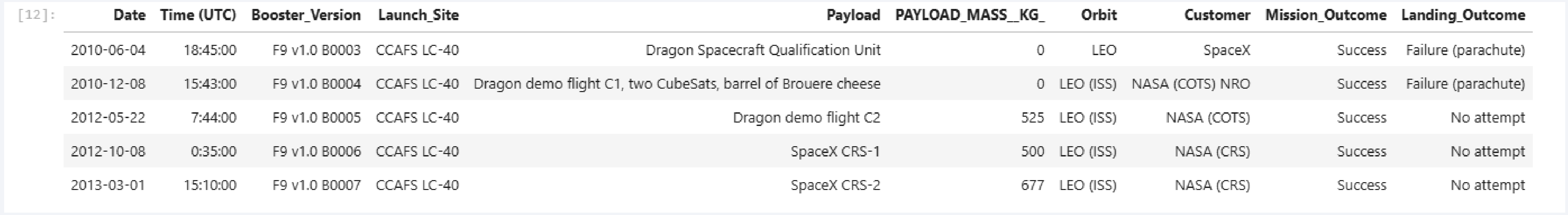


Display 5 records where launch sites begin with the string 'CCA'

```
[12]: %sql SELECT * FROM SPACEXTABLE WHERE "Launch_Site" LIKE 'CCA%' LIMIT 5;

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

- Result:

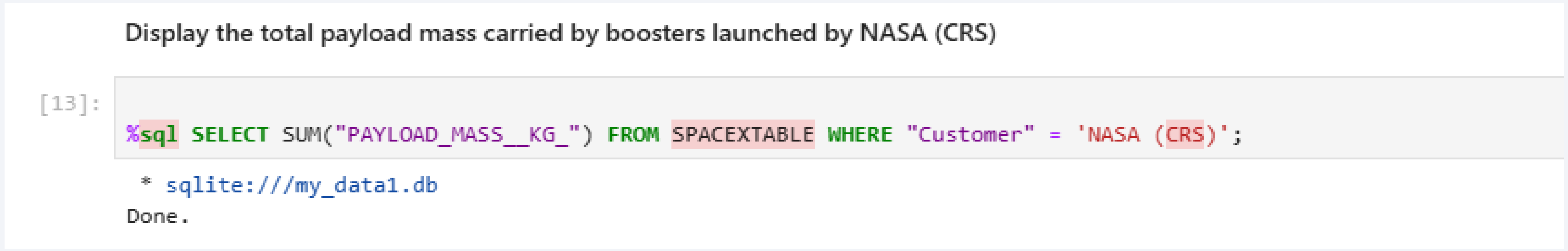


[12]:

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

- Query for payload mass launched by NASA

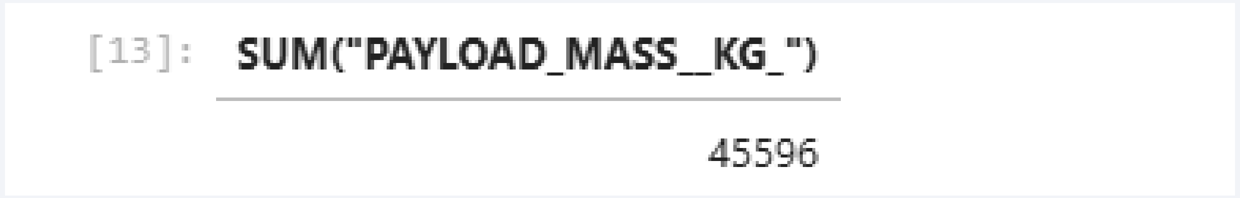


Display the total payload mass carried by boosters launched by NASA (CRS)

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

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

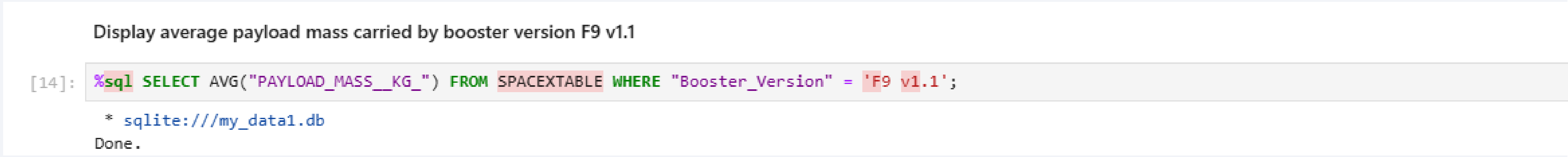
- Result:



```
[13]:  SUM("PAYLOAD_MASS__KG_")
                          45596
```

Average Payload Mass by F9 v1.1

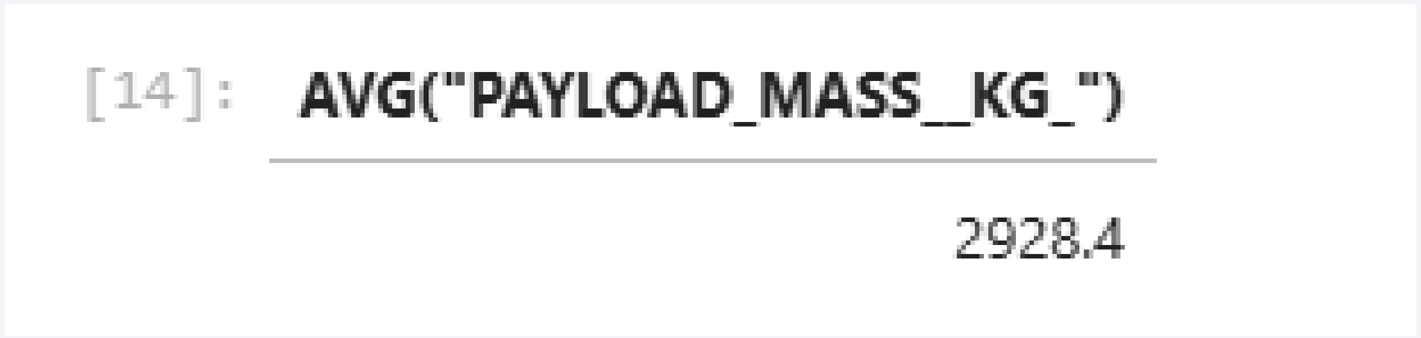
- Query for average payload mass carried by booster version F9 v1.1


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

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

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

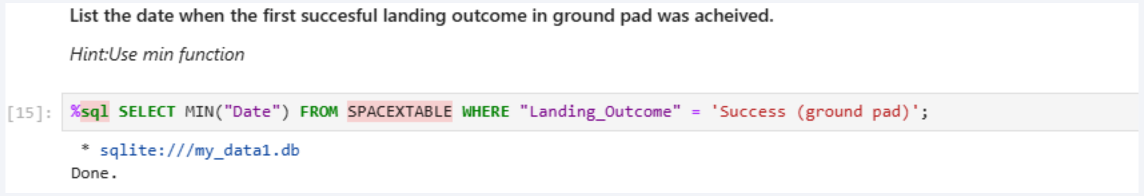
- Result:


[14]:

AVG("PAYLOAD_MASS__KG_")
2928.4

First Successful Ground Landing Date

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



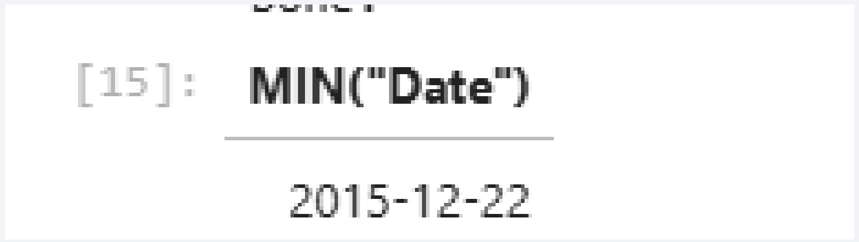
List the date when the first succesful landing outcome in ground pad was acheived.

Hint:Use min function

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

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

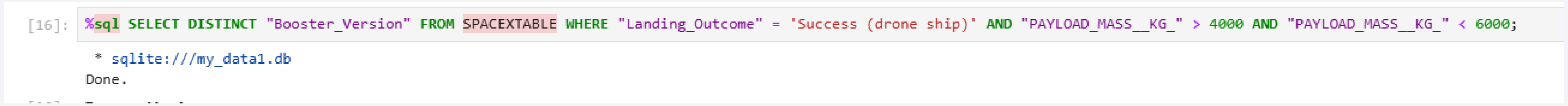
- Result:



```
[15]: MIN("Date")
      2015-12-22
```

Successful Drone Ship Landing with Payload between 4000 and 6000

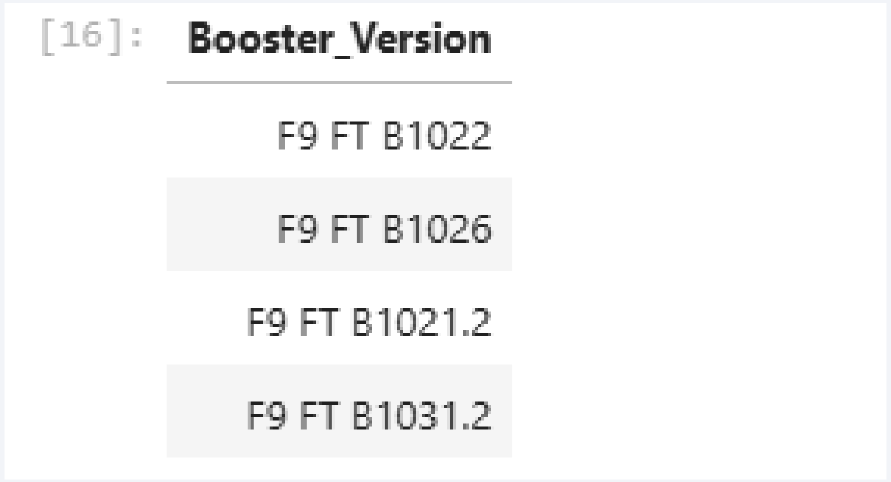
- Query :List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000



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

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

- Result:

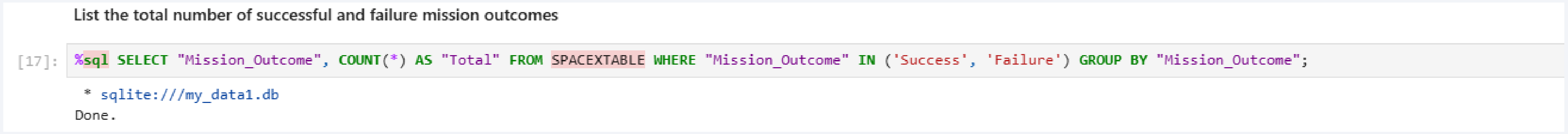


[16]:

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

Total Number of Successful and Failure Mission Outcomes

- Query:Calculate the total number of successful and failure mission outcomes

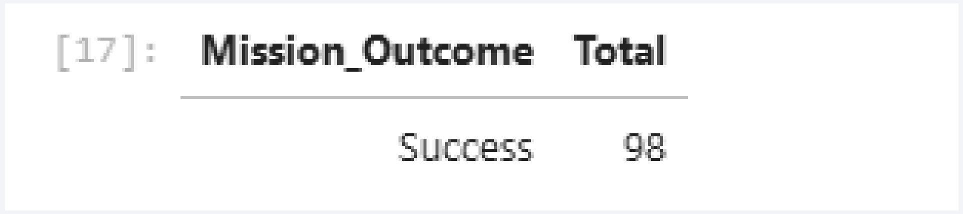


List the total number of successful and failure mission outcomes

```
[17]: %sql SELECT "Mission_Outcome", COUNT(*) AS "Total" FROM SPACEXTABLE WHERE "Mission_Outcome" IN ('Success', 'Failure') GROUP BY "Mission_Outcome";
```

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

- Result:

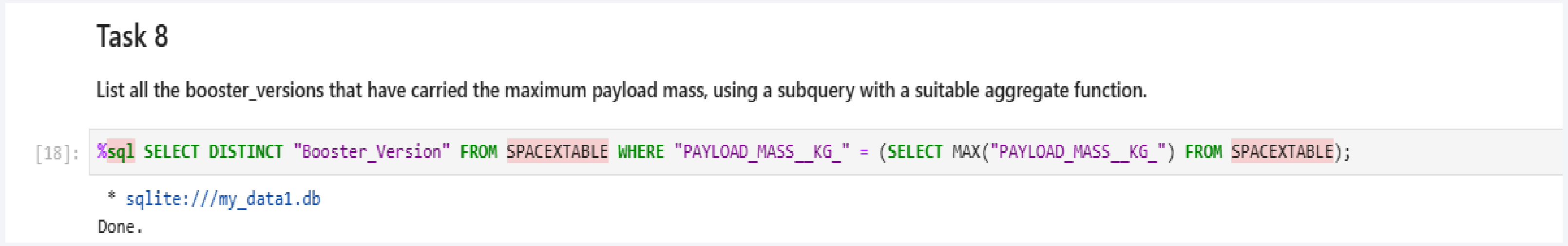


[17]:

Mission_Outcome	Total
Success	98

Boosters Carried Maximum Payload

- Query: List the names of the booster which have carried the maximum payload mass



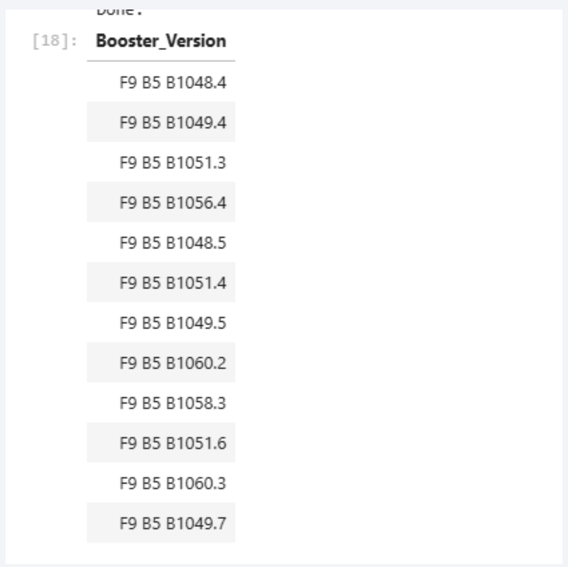
Task 8

List all the booster_versions that have carried the maximum payload mass, using a subquery with a suitable aggregate function.

```
[18]: %sql SELECT DISTINCT "Booster_Version" FROM SPACEXTABLE WHERE "PAYLOAD_MASS__KG_" = (SELECT MAX("PAYLOAD_MASS__KG_") FROM SPACEXTABLE);

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

- Result

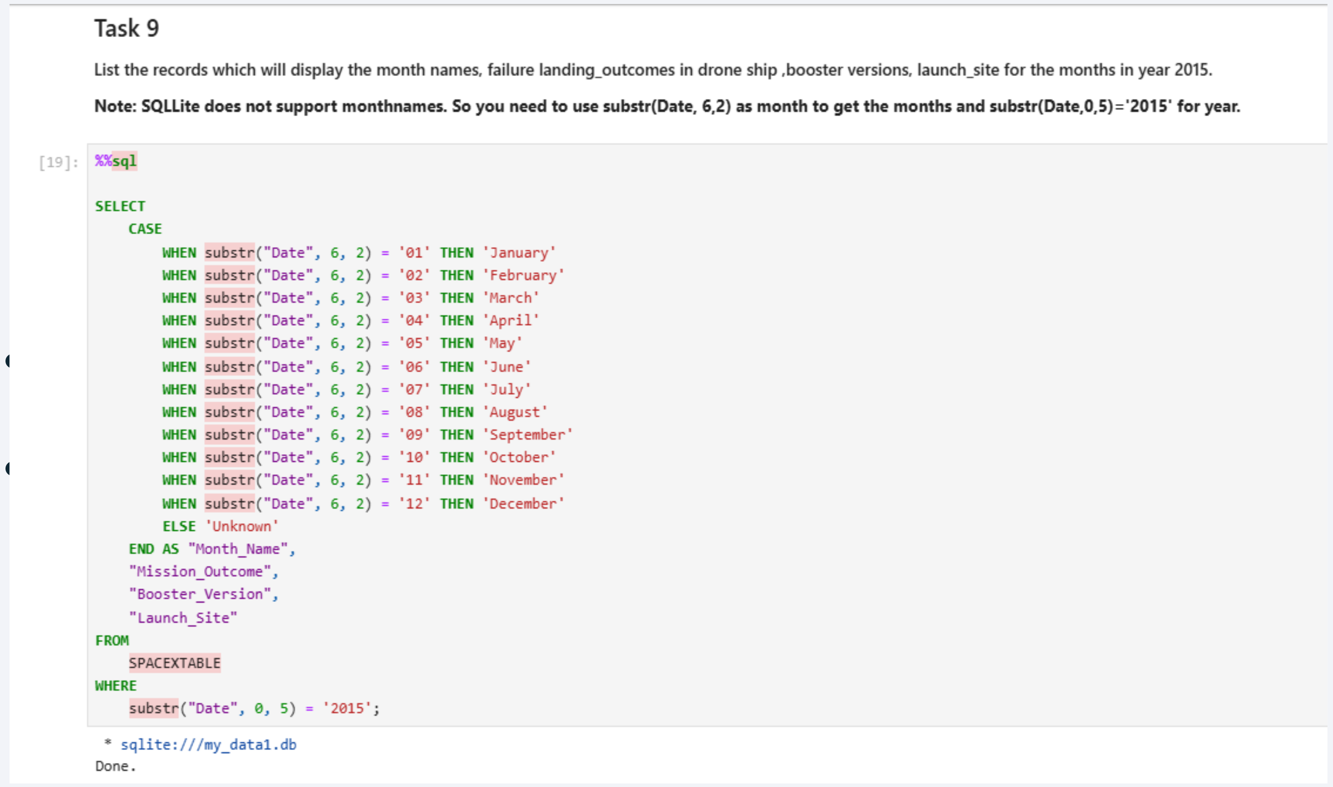


[18]:

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

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



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.

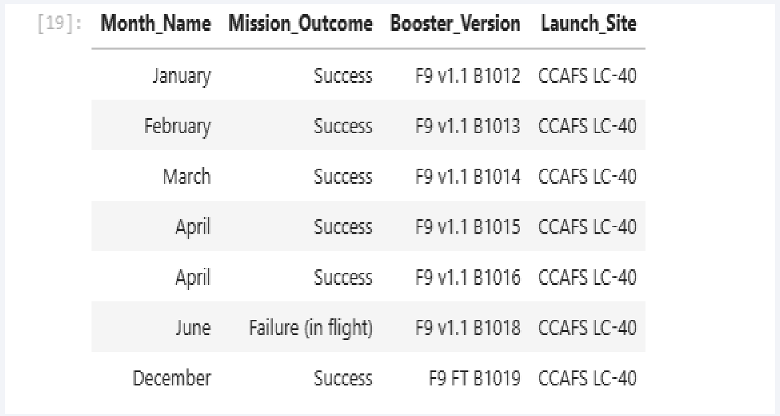
Note: SQLLite does not support monthnames. So you need to use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

```
[19]: %%sql

SELECT
    CASE
        WHEN substr("Date", 6, 2) = '01' THEN 'January'
        WHEN substr("Date", 6, 2) = '02' THEN 'February'
        WHEN substr("Date", 6, 2) = '03' THEN 'March'
        WHEN substr("Date", 6, 2) = '04' THEN 'April'
        WHEN substr("Date", 6, 2) = '05' THEN 'May'
        WHEN substr("Date", 6, 2) = '06' THEN 'June'
        WHEN substr("Date", 6, 2) = '07' THEN 'July'
        WHEN substr("Date", 6, 2) = '08' THEN 'August'
        WHEN substr("Date", 6, 2) = '09' THEN 'September'
        WHEN substr("Date", 6, 2) = '10' THEN 'October'
        WHEN substr("Date", 6, 2) = '11' THEN 'November'
        WHEN substr("Date", 6, 2) = '12' THEN 'December'
        ELSE 'Unknown'
    END AS "Month_Name",
    "Mission_Outcome",
    "Booster_Version",
    "Launch_Site"
FROM
    SPACEXTABLE
WHERE
    substr("Date", 0, 5) = '2015';
```

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

Result :

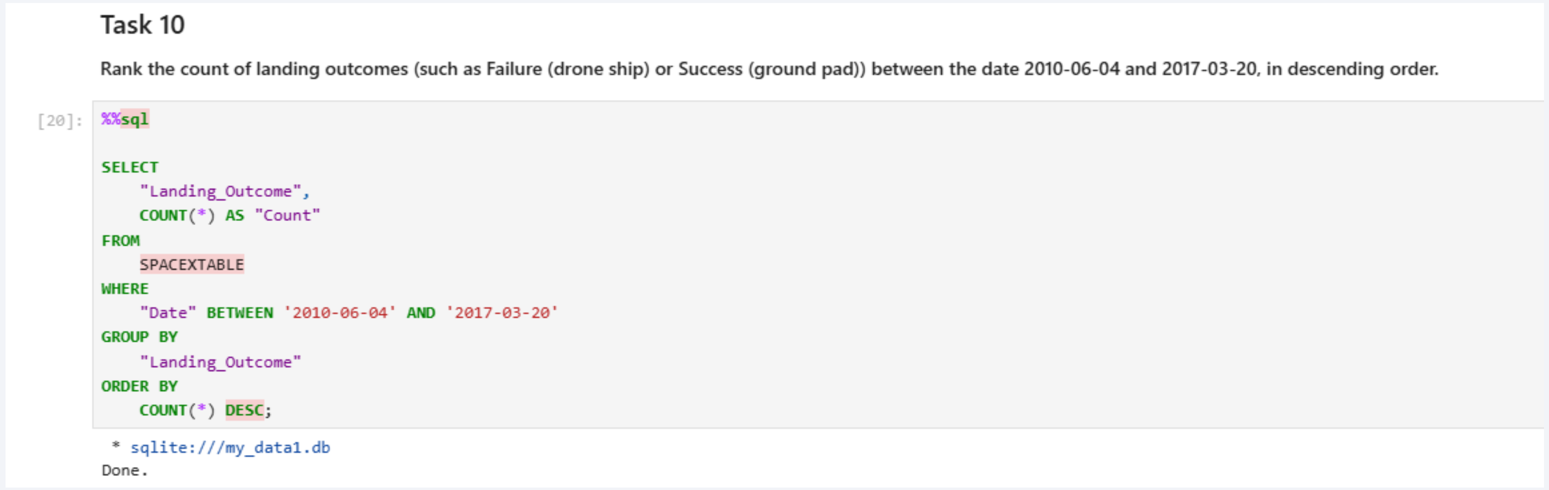


[19]:

Month_Name	Mission_Outcome	Booster_Version	Launch_Site
January	Success	F9 v1.1 B1012	CCAFS LC-40
February	Success	F9 v1.1 B1013	CCAFS LC-40
March	Success	F9 v1.1 B1014	CCAFS LC-40
April	Success	F9 v1.1 B1015	CCAFS LC-40
April	Success	F9 v1.1 B1016	CCAFS LC-40
June	Failure (in flight)	F9 v1.1 B1018	CCAFS LC-40
December	Success	F9 FT B1019	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



Task 10

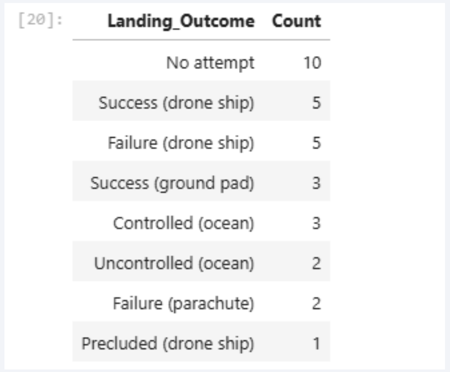
Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

```
[20]: %%sql

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

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

- Result:



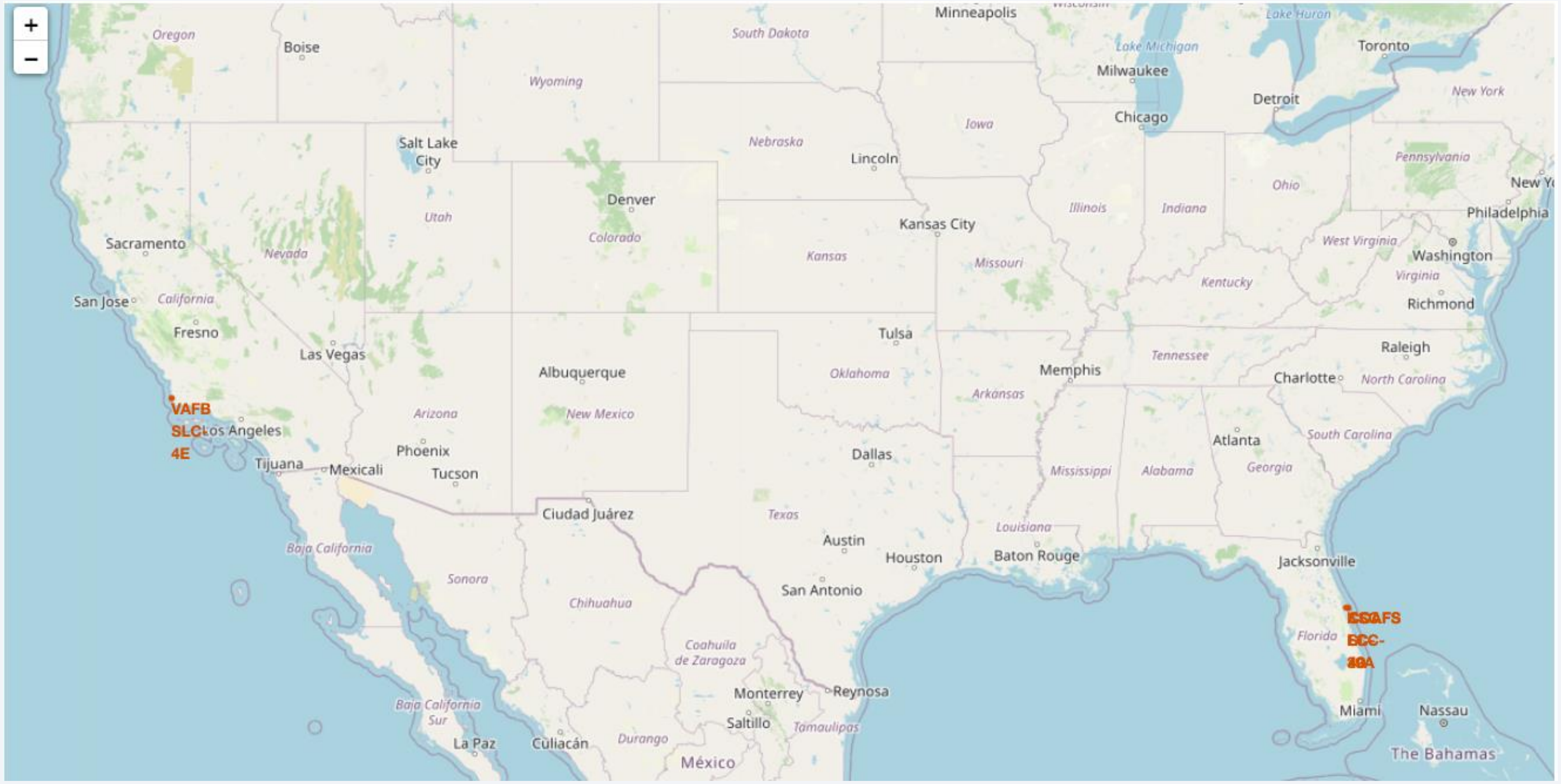
[20]:

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

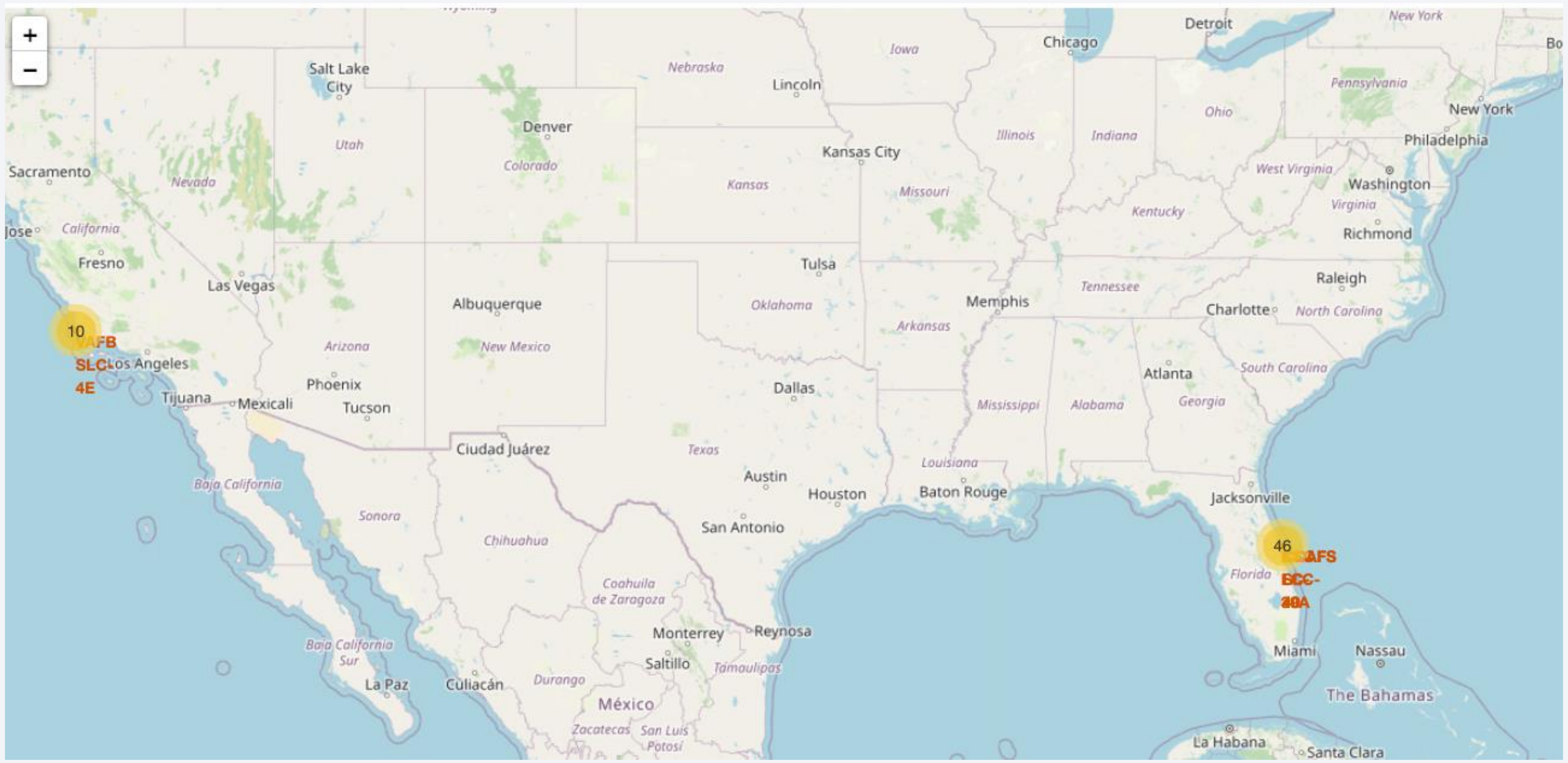
Launch Sites Proximities Analysis

SpaceX Falcon9 Launch Sites Map

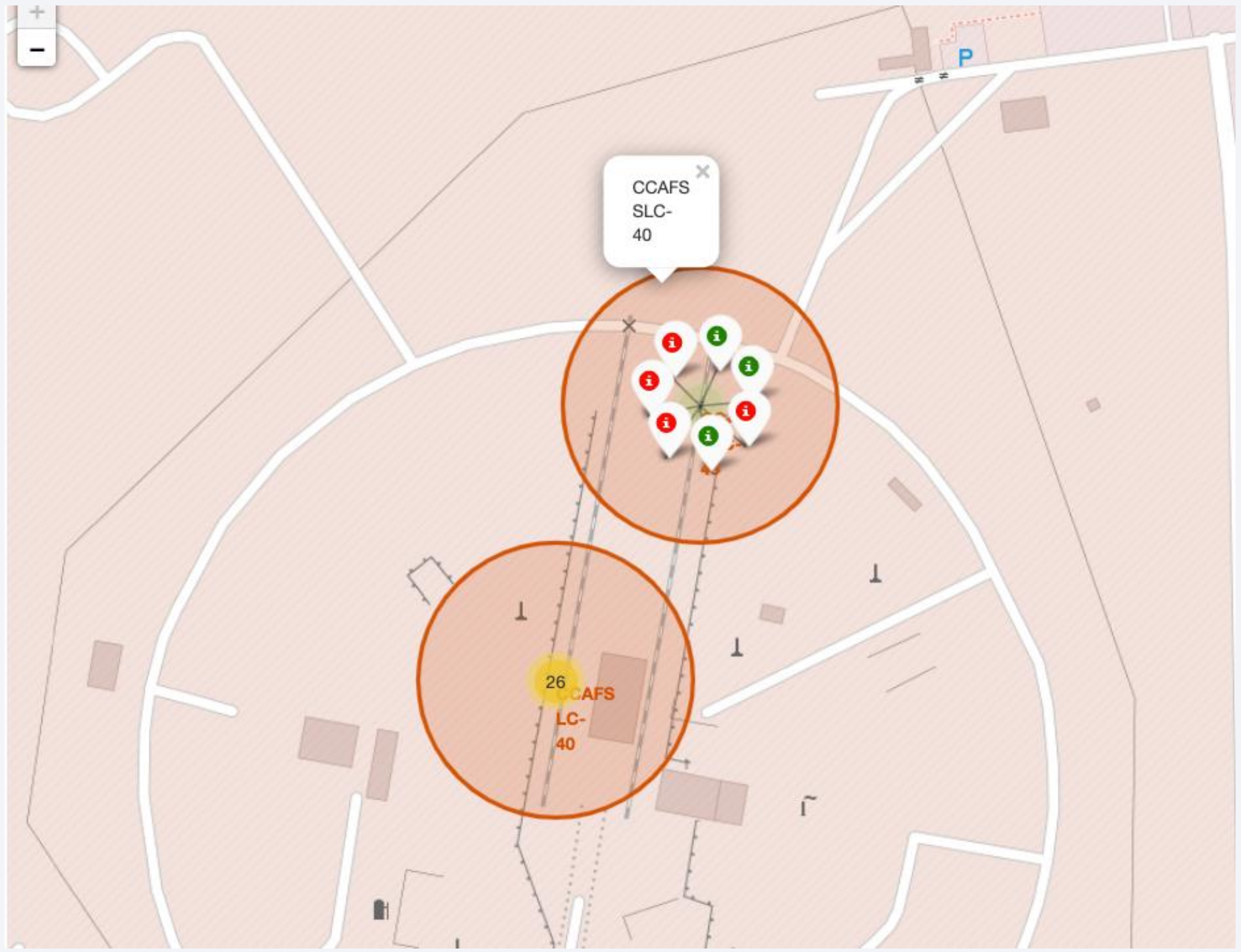


Observation: Figure displays global map with Falcon9 launch sites that are located in California & Florida. Each site highlights the location and name of the launch site.

SpaceX Falcon9 – Success/Failure Launch site Map

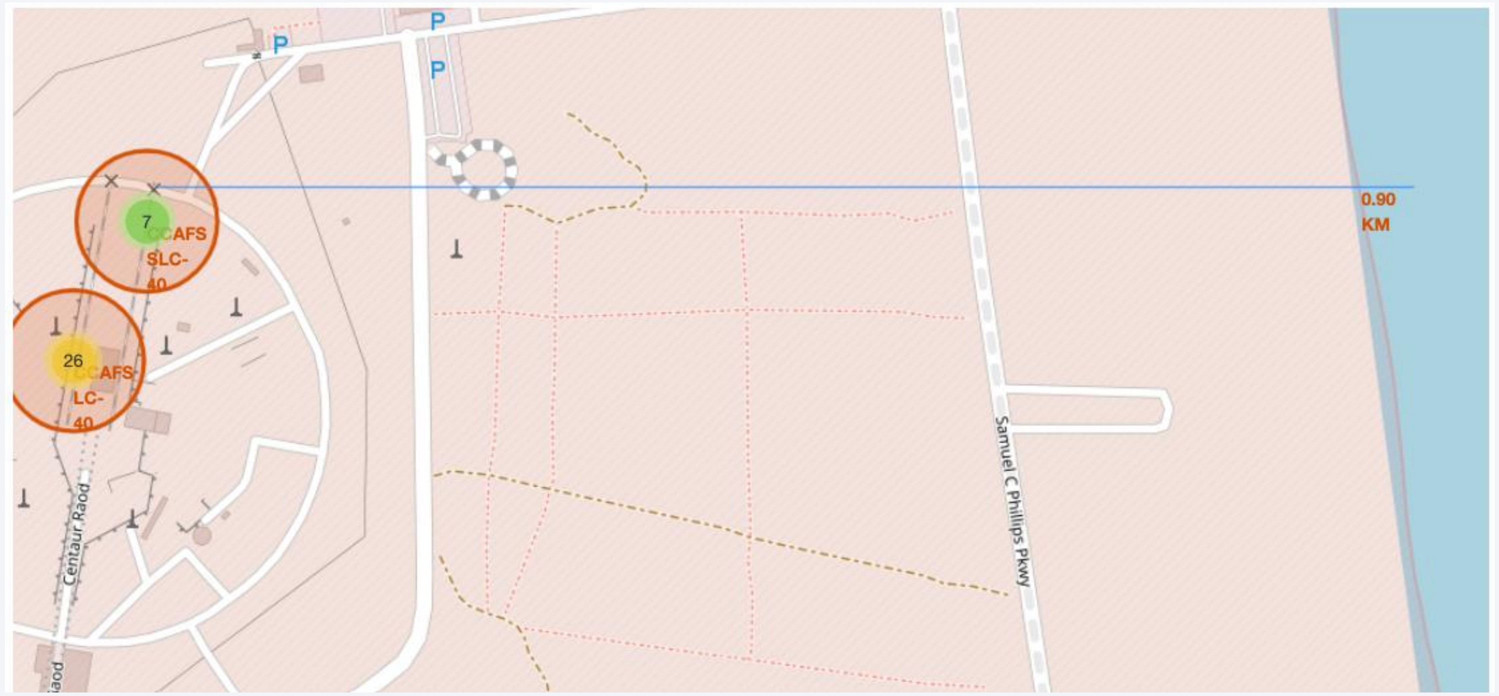


SpaceX Falcon9 – Success/Failure Launch Map –Cont’d

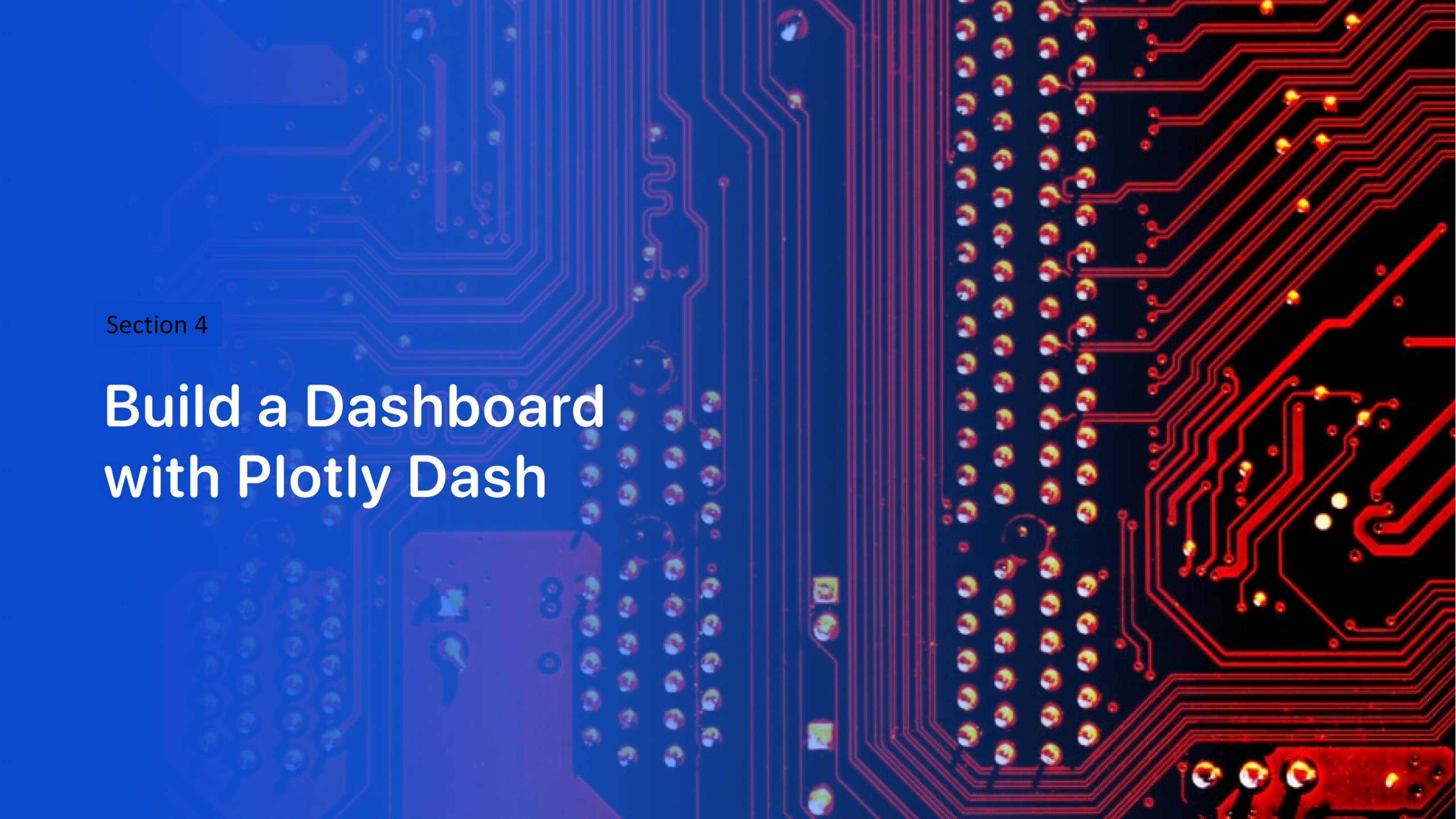


Observation : Zooming a specific launch site we can see green and red tags. Each green tag mean successful launch where red one represents failed launch

SpaceX Falcon9 – Launch Site Proximity Map



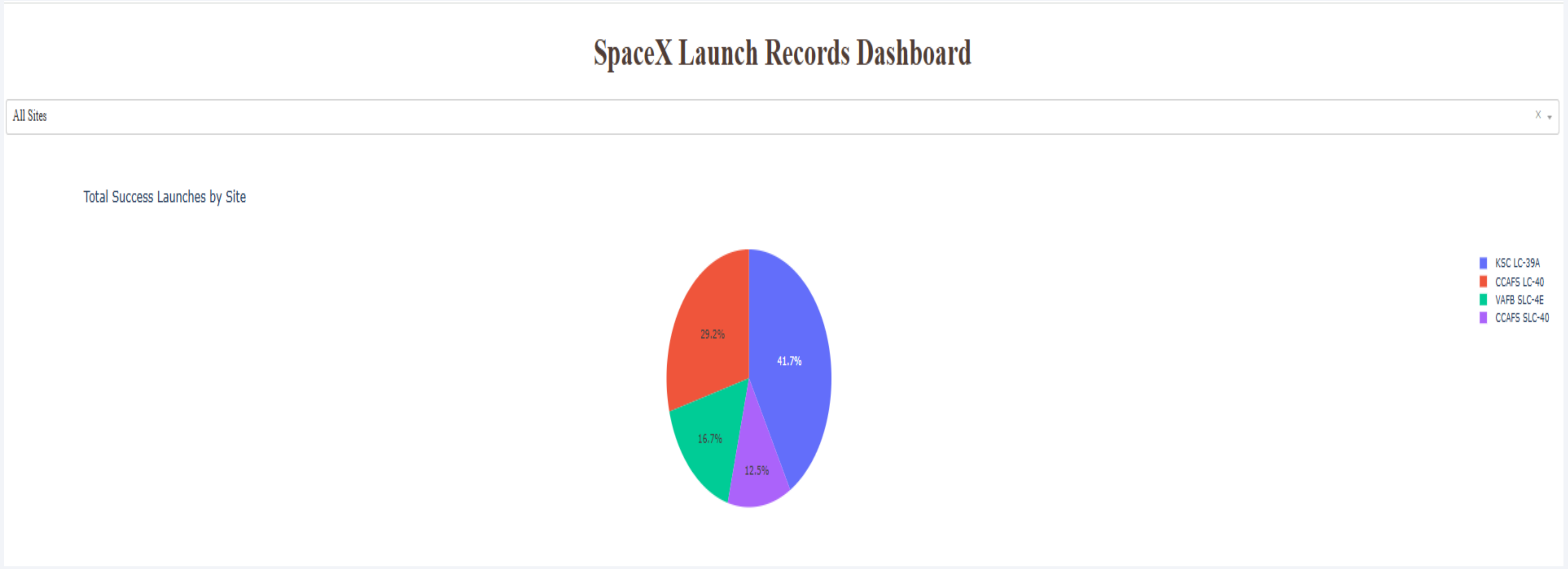
Observation:Figure shows distance between launch site and nearest coastline



Section 4

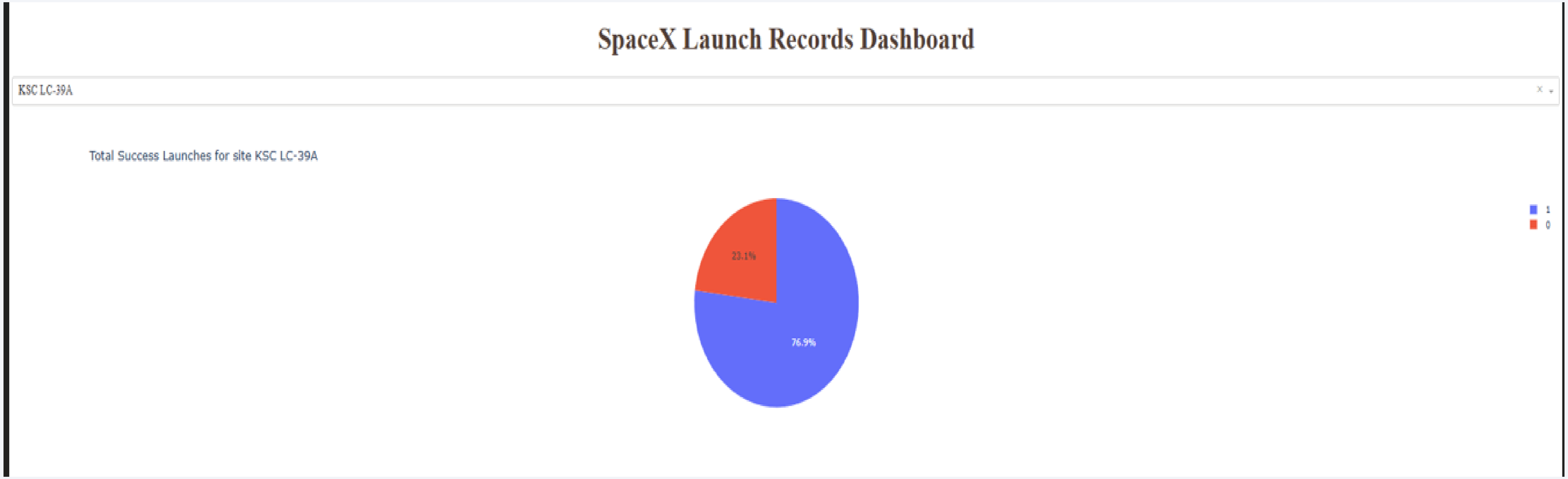
Build a Dashboard with Plotly Dash

All Sites – Launch Success



- Launch site 'KSC LC-39A' has the highest launch success rate
- Launch site 'CCAFS SLC-40' has the lowest launch success rate

Launch Site – Highest Launch Success Rate

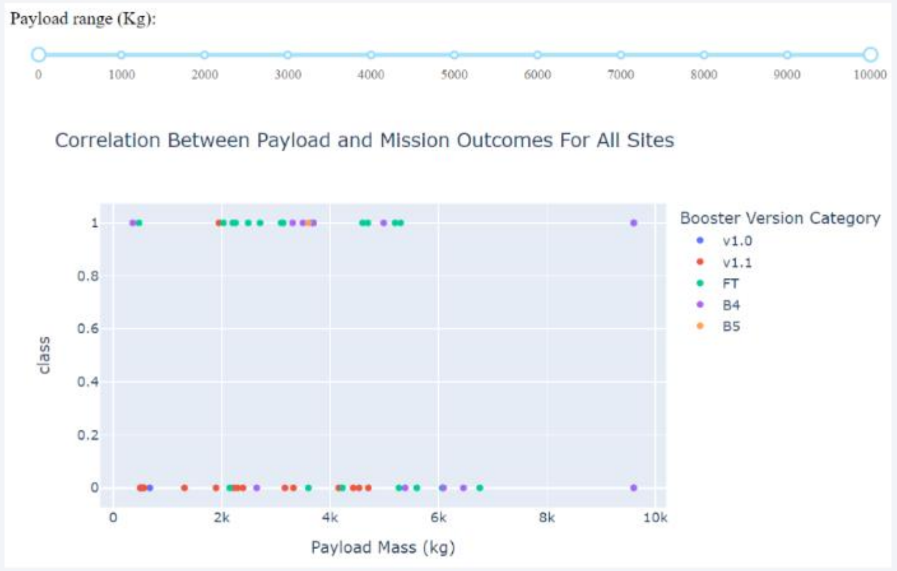

SpaceX Launch Records Dashboard

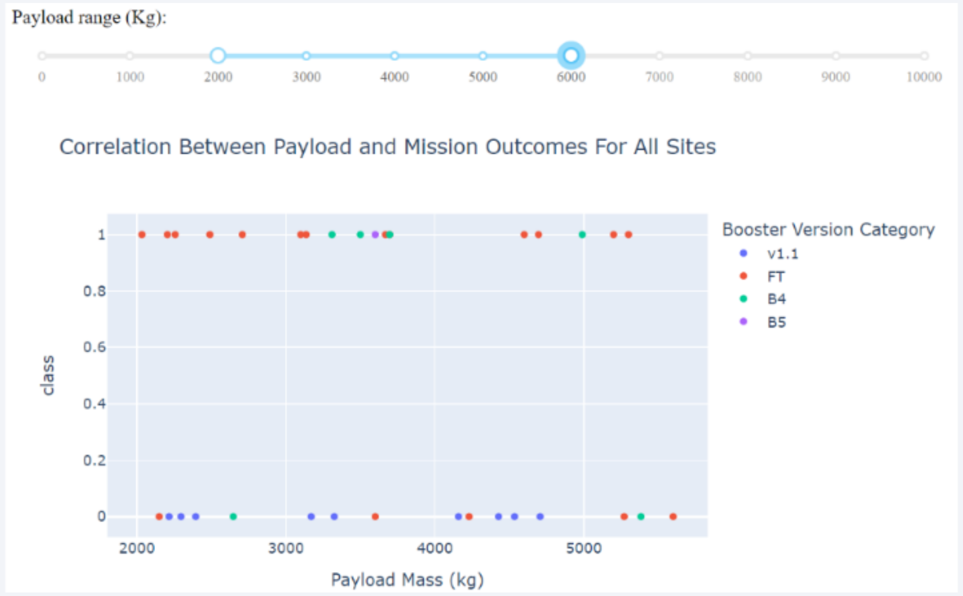
KSC LC-39A

Total Success Launches for site KSC LC-39A

- Launch site ‘KSC LC-39A’ has highest success rate 76.9% and failure rate 23.1%

Payload Vs Launch Outcome



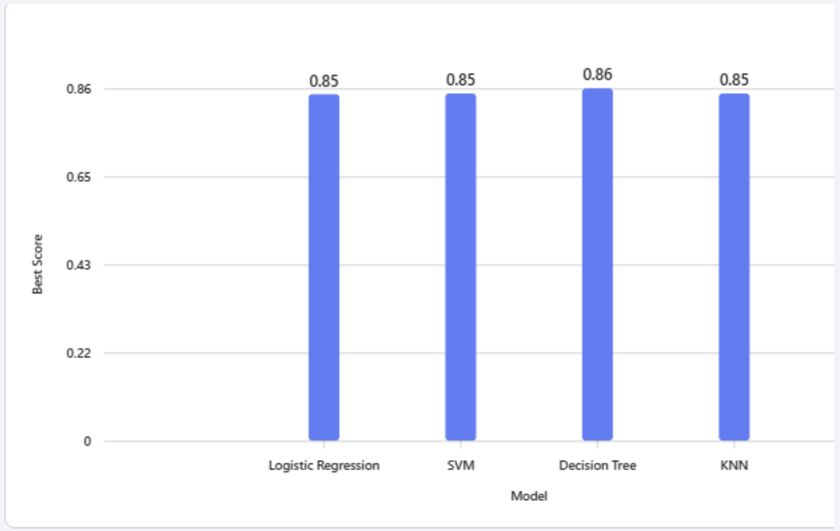


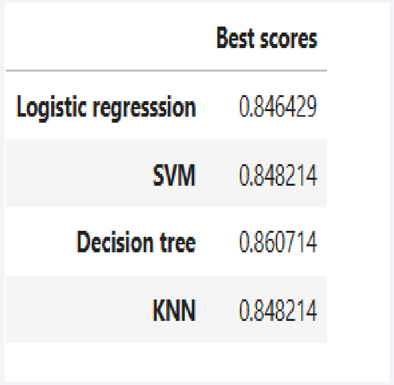
- **Observation:** Successful launches falls under payload mass range 2000 - 5500

Section 5

Predictive Analysis (Classification)

Classification Accuracy

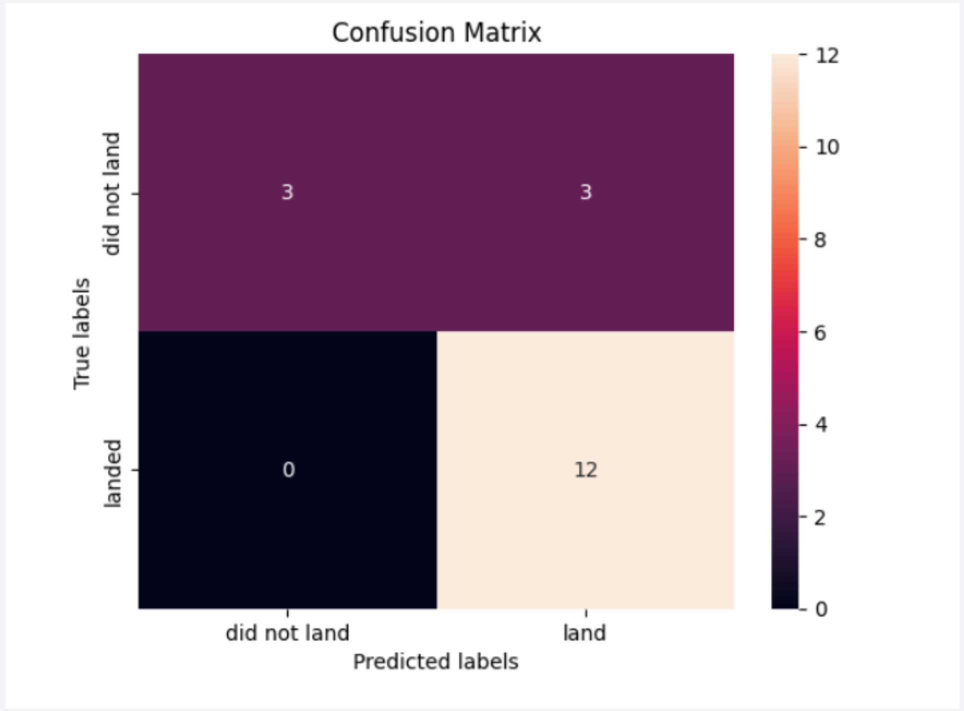




	Best scores
Logistic regresssion	0.846429
SVM	0.848214
Decision tree	0.860714
KNN	0.848214

- Based on the accuracy score and evidence from bar chart ‘Decision Tree algorithm’ has highest classification score

Confusion Matrix



- In general classifier is correctly predicted for about 83.4% of the time (True positive + True Negative /Total = 12 + 3 /18) and error rate 16.6% (False Positive + False Negative /Total = 3+0 /18)

Conclusions

- **Insight #1**: The study successfully predicted Falcon 9 first-stage landing outcomes using launch and mission features, helping estimate launch costs.
- **Insight#2** :Landing and launch success rates improved significantly over time (≈80% from 2013–2020)
- **Insight#3**:Success varies by launch site and orbit type, while payload mass shows no strong correlation with landing success.
- **Insight#4**:The Decision Tree model performed best among all models, achieving ~83% accuracy on test data, with potential for improvement using more data.

Thank you!