# IBM DATA SCIENCE CAPSTONE PROJECT

SpaceX Falcon 9 Landing and Cost Analysis

Jonathan De la O

# Outline

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

# Executive Summary

- **Summary of methodologies**

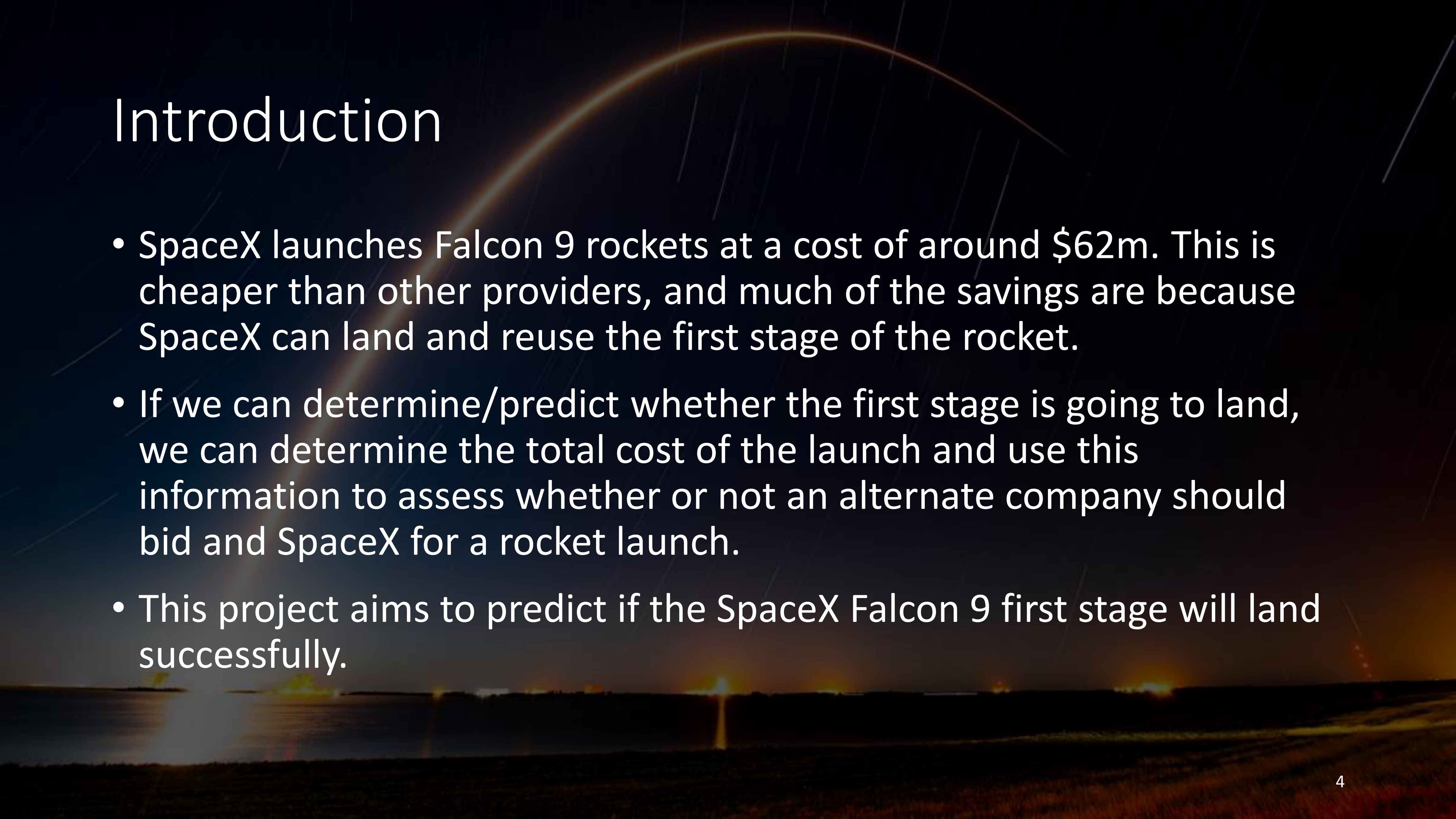
This project follow the next steps:

1. Data Collection
2. Data Wrangling
3. Exploratory Data Analysis
4. Interactive Visual Analytics
5. Predictive Analysis

- **Summary of all results**

This project outcomes were:

1. Exploratory Data Analysis Results
2. Geospatial analytics
3. Interactive dashboard
4. Predictive analysis of classification models



# Introduction

- SpaceX launches Falcon 9 rockets at a cost of around $62m. This is cheaper than other providers, and much of the savings are because SpaceX can land and reuse the first stage of the rocket.
- If we can determine/predict whether the first stage is going to land, we can determine the total cost of the launch and use this information to assess whether or not an alternate company should bid and SpaceX for a rocket launch.
- This project aims to predict if the SpaceX Falcon 9 first stage will land successfully.

# Section 1

## Methodology

# Methodology

## Executive Summary

- Data collection methodology:
  - Making GET requests to the SpaceX REST API.
  - Web Scraping
- Perform data wrangling
  - Using the .fillna() method to remove NaN values.
  - Using the .value_counts() to determine the number of launches on each site, number and occurrence of each orbit and the number and occurrence of mission outcome per orbit type.
  - Creating a landing outcome label that shows when the booster didn’t land successfully and when it did land successfully.

# Methodology

## Executive Summary

- Perform exploratory data analysis (EDA) using visualization and SQL
  - Using SQL queries to manipulate and evaluate the SpaceX dataset
  - Using Pandas and Matplotlib to visualize relationships between variables, and determine patterns
- Perform interactive visual analytics using Folium and Plotly Dash
  - Geospatial analytics using Folium
  - Creating an interactive dashboard using Plotly Dash
- Perform predictive analysis using classification models

  Using Scikit-Learn to:

  ○ Pre-process (standardize) the data

  ○ Split the data into training and testing data using train_test_split

  ○ Train different classification models

  ○ Find hyperparameters using GridSearchCV

  Plotting confusion matrices for each classification model

  Assessing the accuracy of each classification model

# Data Collection

- Using the SpaceX API to retrieve data about launches, including information about the rocket used, payload delivered, launch specifications, landing specifications, and landing outcome.
- The next slide will show the flowchart of data collection from API and the one after will show  the flowchart of data collection from webscraping.
- Space X API Data Columns:

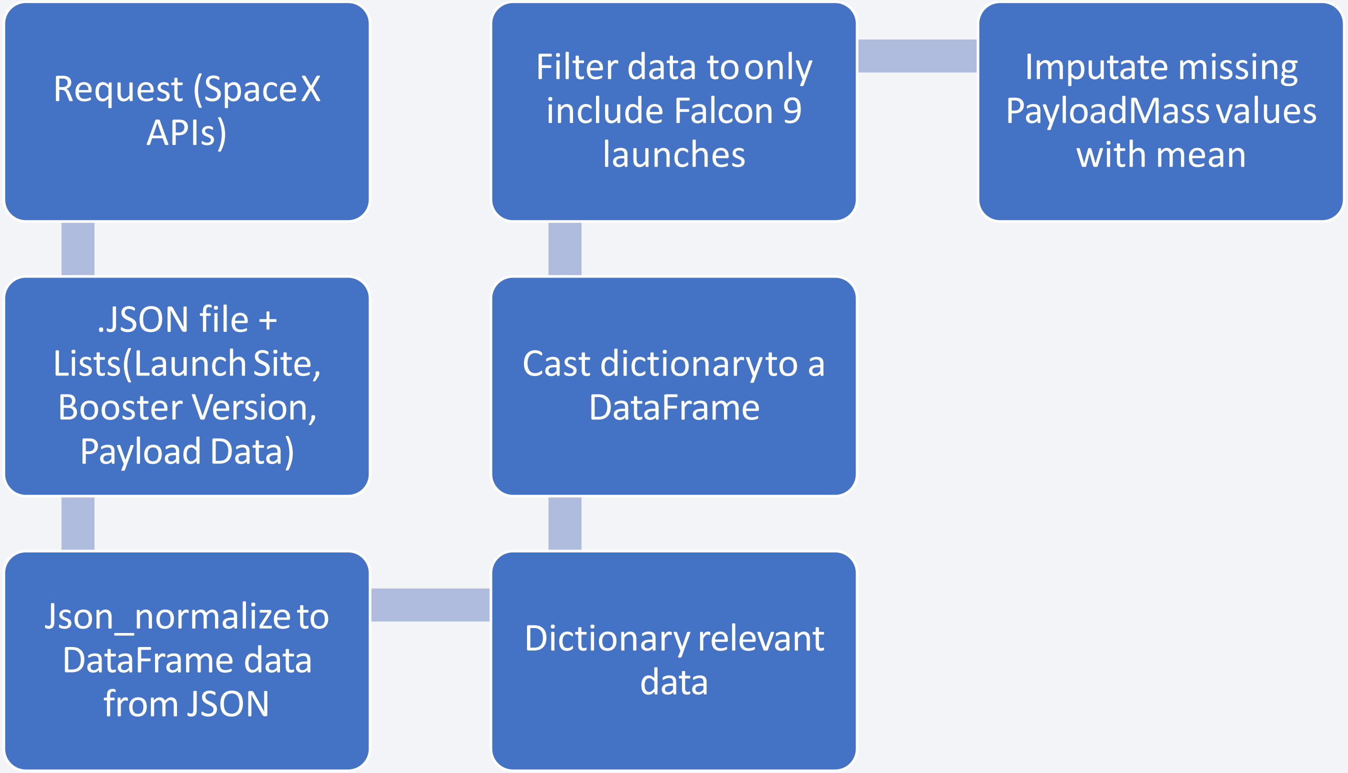
FlightNumber, Date, BoosterVersion, PayloadMass, Orbit, LaunchSite, Outcome, Flights, GridFins, Reused, Legs, LandingPad, Block, ReusedCount, Serial, Longitude, Latitude

- Wikipedia Webscrape Data Columns:

Flight No., Launch site, Payload, PayloadMass, Orbit, Customer, Launch outcome, Version  Booster, Booster landing, Date, Time

# Data Collection – SpaceX API

GitHub Link: https://github.com/Jonathandelao10/Data-Science-Certificate/blob/eaad19ec43e83d8ba677e8acf32e766dd5706599/Capstone%20Project/jupyter-labs-spacex-data-collection-api.ipynb



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

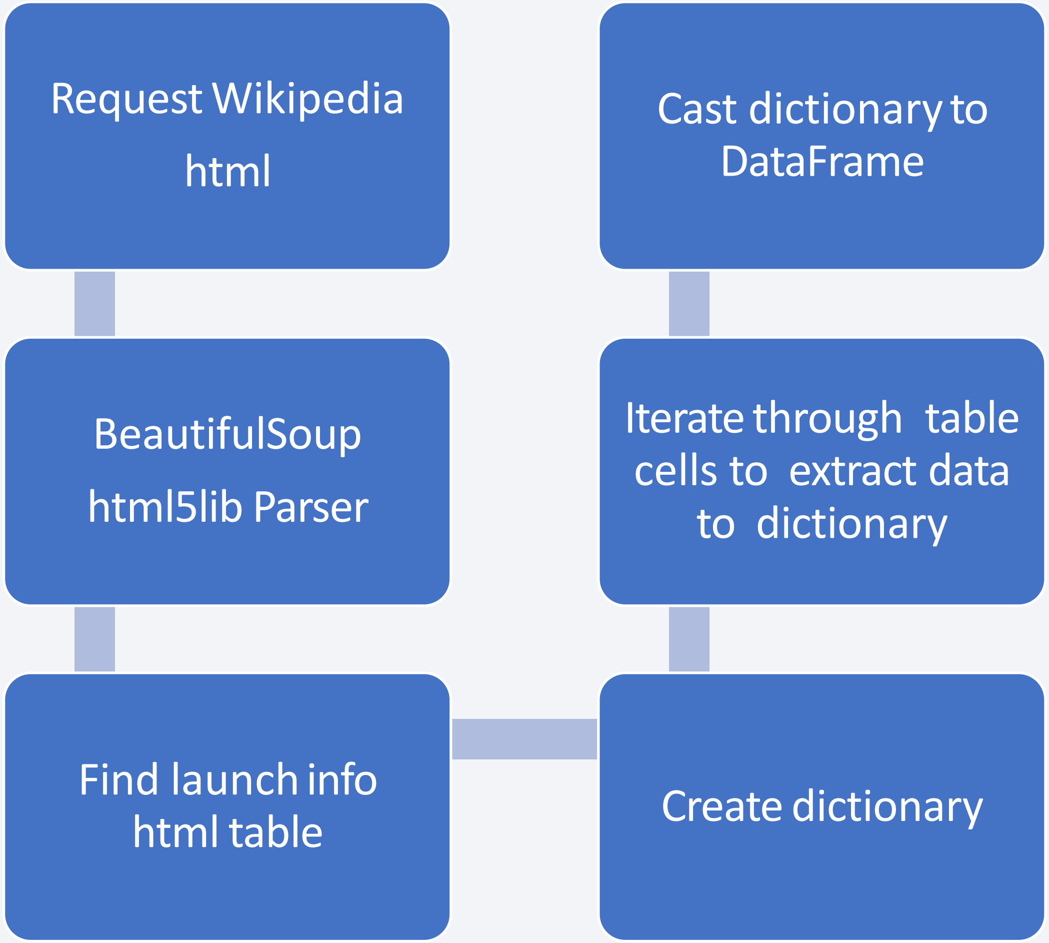
Filter data to only include Falcon 9 launches

Imputate missing PayloadMass values with mean

# Data Collection - Scraping

- GitHub Link:

https://github.com/Jonathandelao10/Data-Science-Certificate/blob/69b94f3f7d40c91bf259191d260f842f6f485a54/Capstone%20Project/jupyter-labs-webscraping.ipynb



Request Wikipedia html

BeautifulSoup html5lib Parser

Find launch info html table

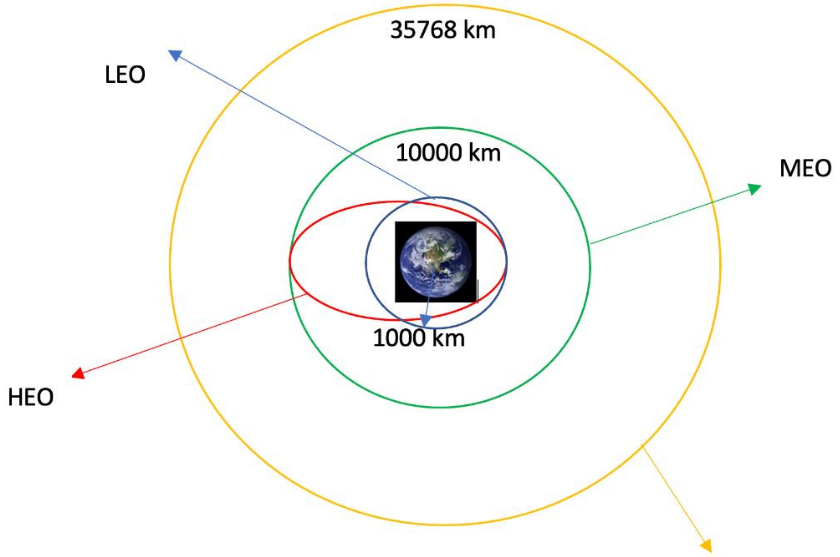
Create dictionary

Iterate through table cells to extract data to dictionary

Cast dictionary to DataFrame

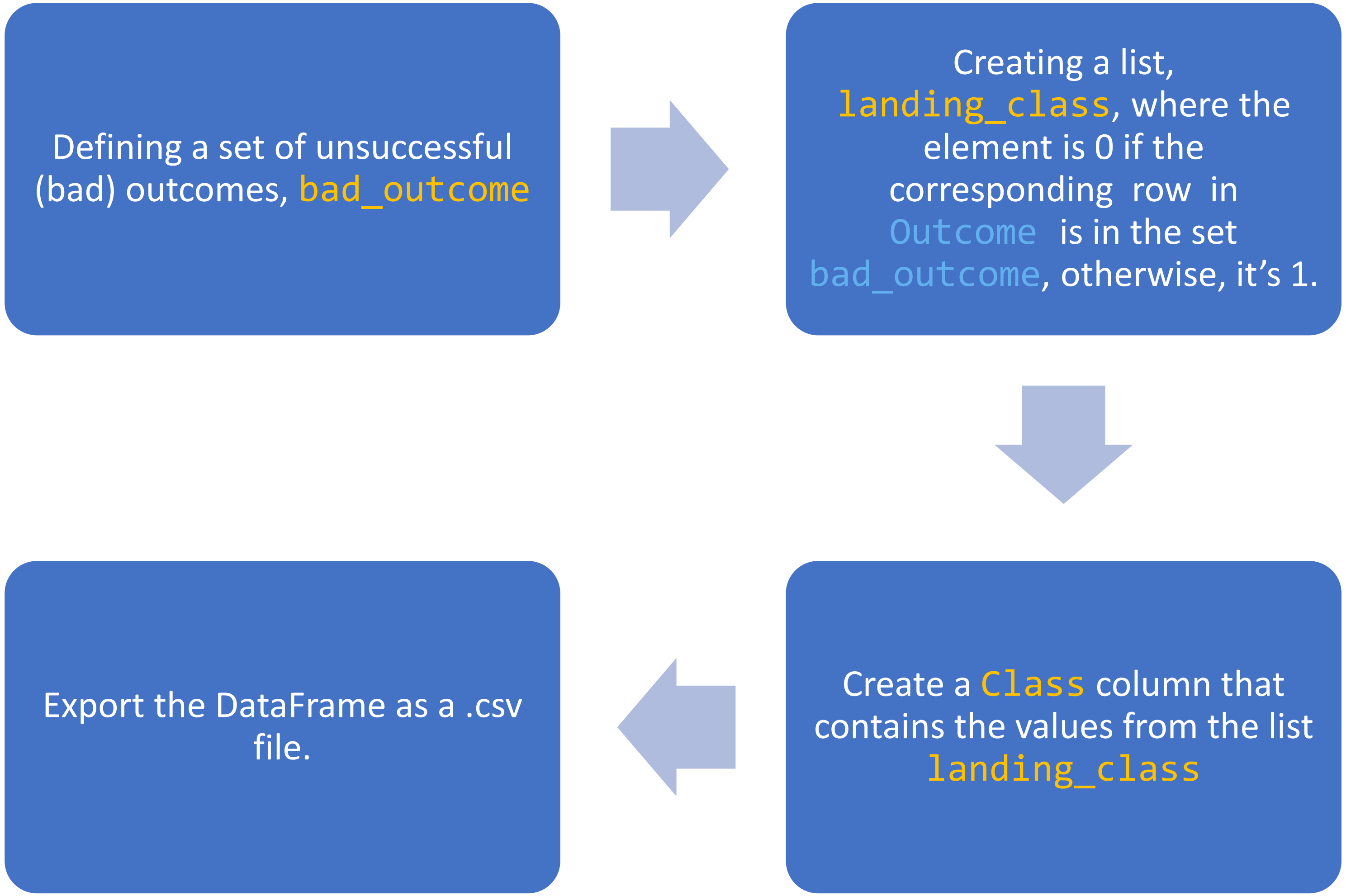
# Data Wrangling

- Context:
  - The SpaceX dataset contains several Space X launch facilities, and each location is in the LaunchSite column.
  - Each launch aims to a dedicated orbit, and some of the common orbit types are shown in the figure below. The orbit type is in the Orbit column.
  - Initial Data Exploration:
  - Using the .value_counts() method to determine the following:
  - Number of launches on each site
  - Number and occurrence of each orbit
  - Number and occurrence of landing outcome per orbit type



# Data Wrangling

- To determine whether a booster will successfully land, it is best to have a binary column, i.e., where the value is 1 or 0, representing the success of the landing.
- GitHub Link: https://github.com/Jonathandelao10/Data-Science-Certificate/blob/3229cdf29fc081d77331a4a2783840ec4389fecc/Capstone%20Project/labs-jupyter-spacex-Data%20wrangling.ipynb



Defining a set of unsuccessful (bad) outcomes, `bad_outcome`

Creating a list, `landing_class`, where the element is 0 if the corresponding row in `Outcome` is in the set `bad_outcome`, otherwise, it’s 1.

Create a `Class` column that contains the values from the list `landing_class`

Export the DataFrame as a .csv file.

# 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 Link: https://github.com/Jonathandelao10/Data-Science-Certificate/blob/c3f99190838d37ea4f1c3bf47abaa49e31a2a85b/Capstone%20Project/Module%202/EDA%20with%20Visualization%20Lab.ipynb

# EDA with SQL

- To gather some information about the dataset, some SQL queries were performed.
- **The SQL queries performed on the data set were used to:**

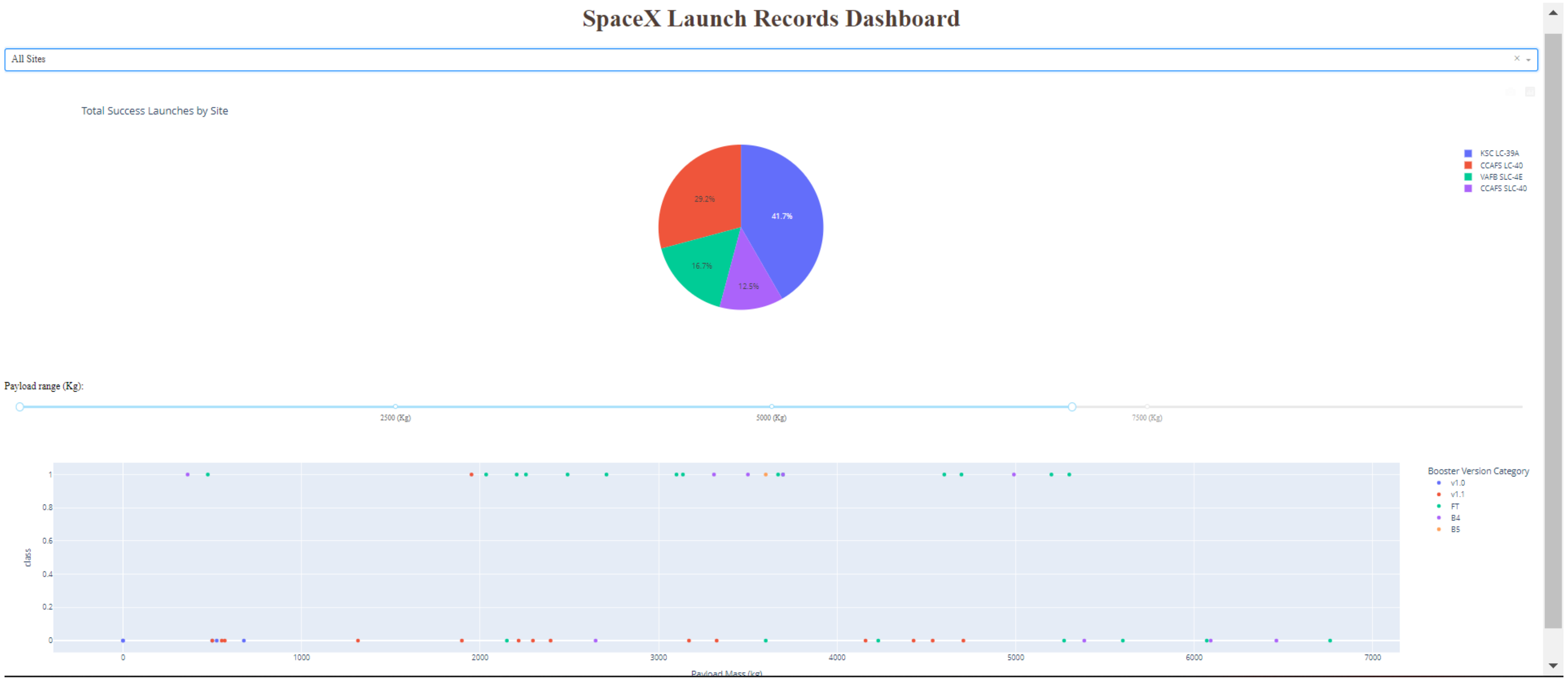
1. Display the names of the unique launch sites in the space mission
2. Display 5 records where launch sites begin with the string ‘CCA’
3. Display the total payload mass carried by boosters launched by NASA (CRS)
4. Display the average payload mass carried by booster version F9 v1.1
5. List the date when the first successful landing outcome on a ground pad was achieved
6. List the names of the boosters which had success on a drone ship and a payload mass between 4000 and 6000 kg
7. List the total number of successful and failed mission outcomes
8. List the names of the booster versions which have carried the maximum payload mass
9. List the failed landing outcomes on drone ships, their booster versions, and launch site names for 2015
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

GitHub Link:
https://github.com/Jonathandelao10/Data-Science-Certificate/blob/7782649d4c6764fd895377076fca75c3d1dfcb6b/Capstone%20Project/Module%202/jupyter-labs-eda-sql-coursera_sqllite.ipynb

# Build an Interactive Map with Folium

- Folium maps mark Launch Sites, successful and unsuccessful landings, and a proximity example to key locations: Railway, Highway, Coast, and City.
- This allows us to understand why launch sites may be located where they are. Also visualizes successful landings relative to location.

- GitHub Link: https://github.com/Jonathandelao10/Data-Science-Certificate/blob/b9a181ff03e25e6251161f28d968e4929bb3d6e1/Capstone%20Project/Module%203/lab_jupyter_launch_site_location.ipynb



SpaceX Launch Records Dashboard

All Sites

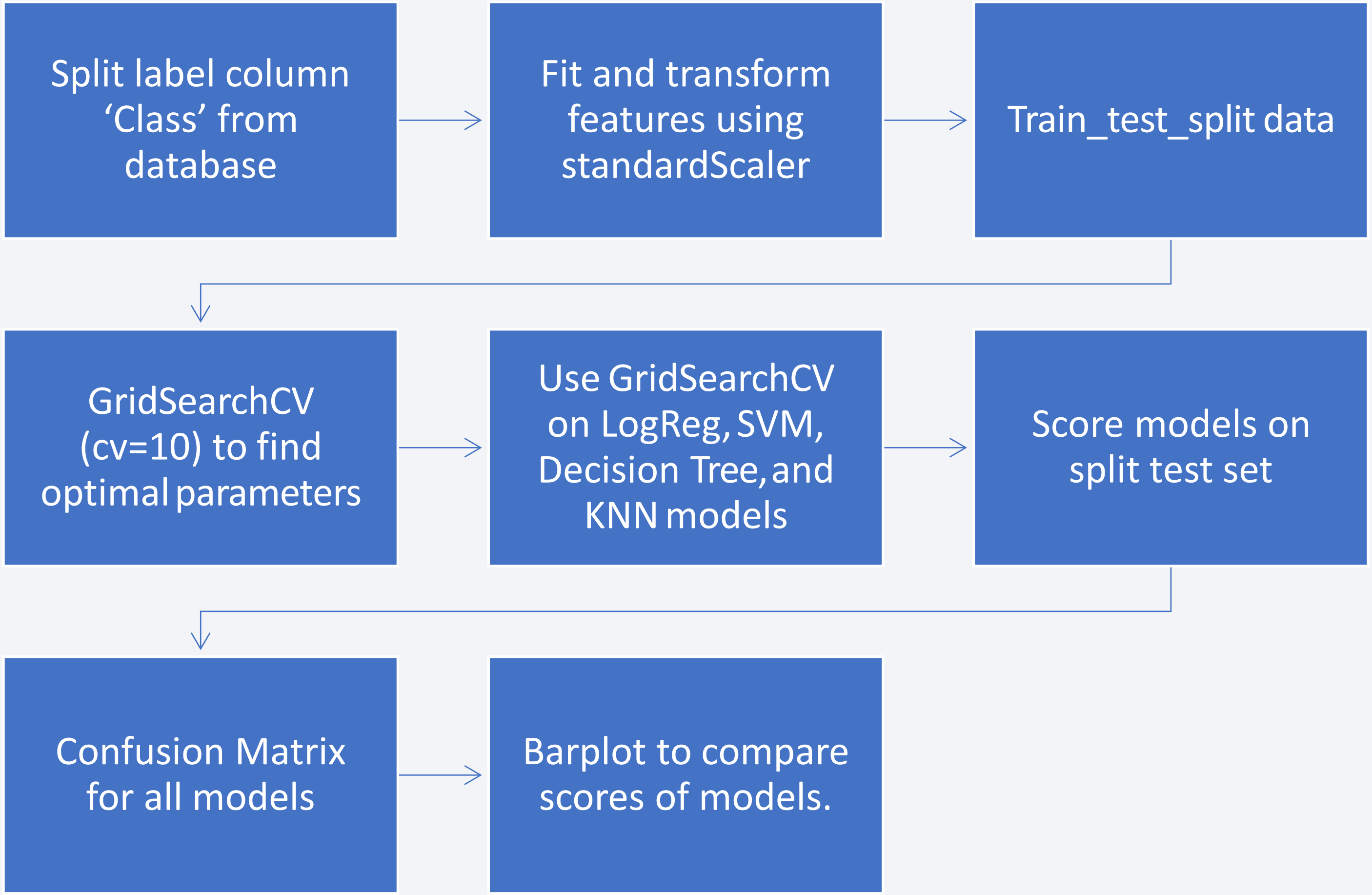
Payload range (Kg):

# Results

This is a preview of the Plotly dashboard.

# Predictive Analysis (Classification)

GitHub Link: https://github.com/Jonathandelao10/Data-Science-Certificate/blob/1050810d2dc79ea4f54217ab896e2c4c584d0689/Capstone%20Project/Module%204/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb



Split label column 'Class' from database

Fit and transform features using standardScaler

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

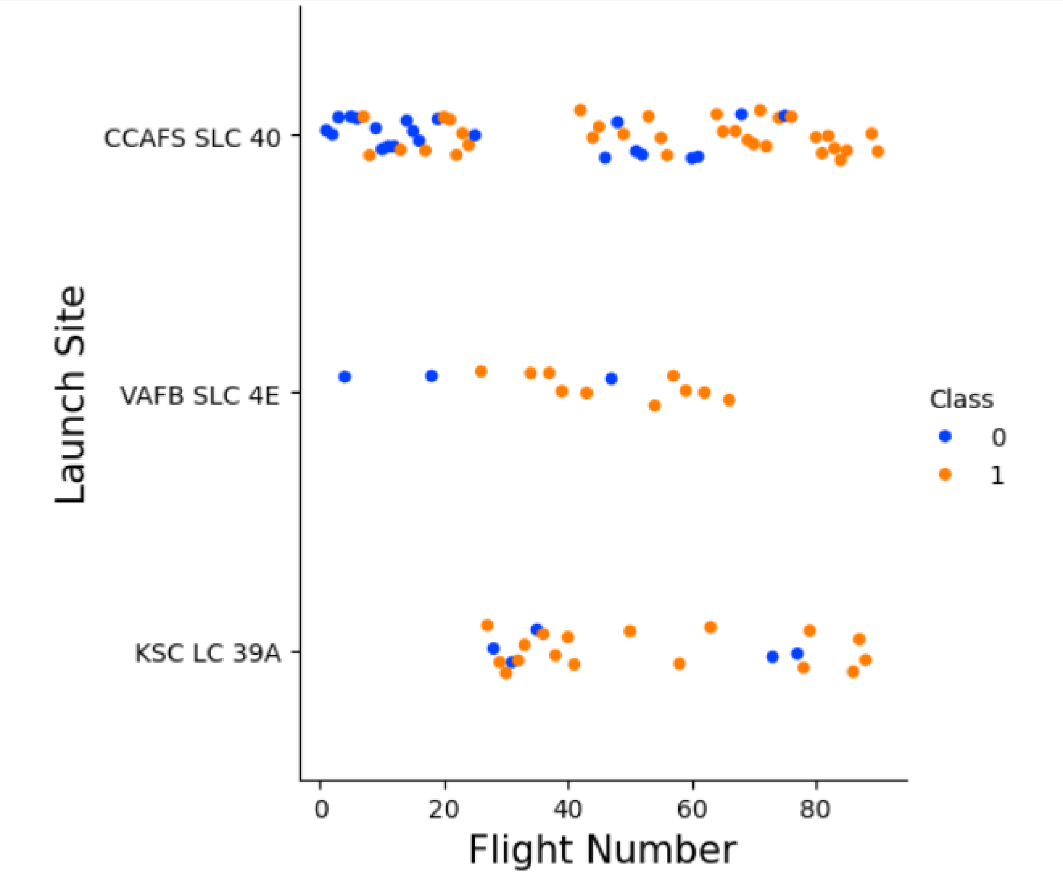
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

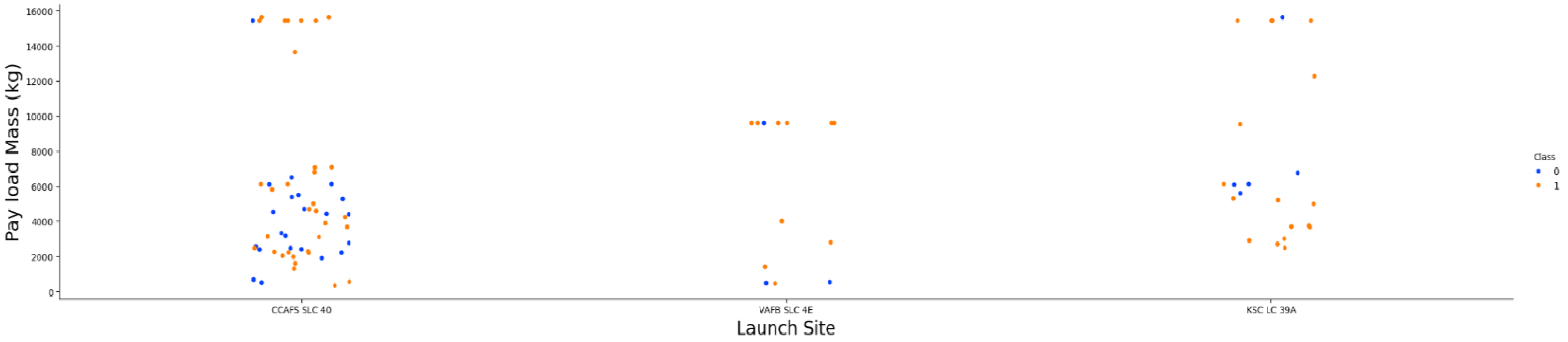
Section 2

# Insights drawn from EDA

# Flight Number vs. Launch Site

- The scatter plot of Launch Site vs. Flight Number shows that:
- No early flights were launched from KSC LC 39A, so the launches from this site are more successful.
- Above a flight number of around 30, there are significantly more successful landings (Class = 1).
- As the number of flights increases, the rate of success at a launch site increases.
- Most of the early flights (flight numbers < 30) were launched from CCAFS SLC 40, and were generally unsuccessful.
- The flights from VAFB SLC 4E also show this trend, that earlier flights were less successful.





# Payload vs. Launch Site

- Orange indicates successful launch; Blue indicates unsuccessful launch.
- Payload mass appears to fall mostly between 0-7,000 kg. Different launch sites also seem to use different payload mass.

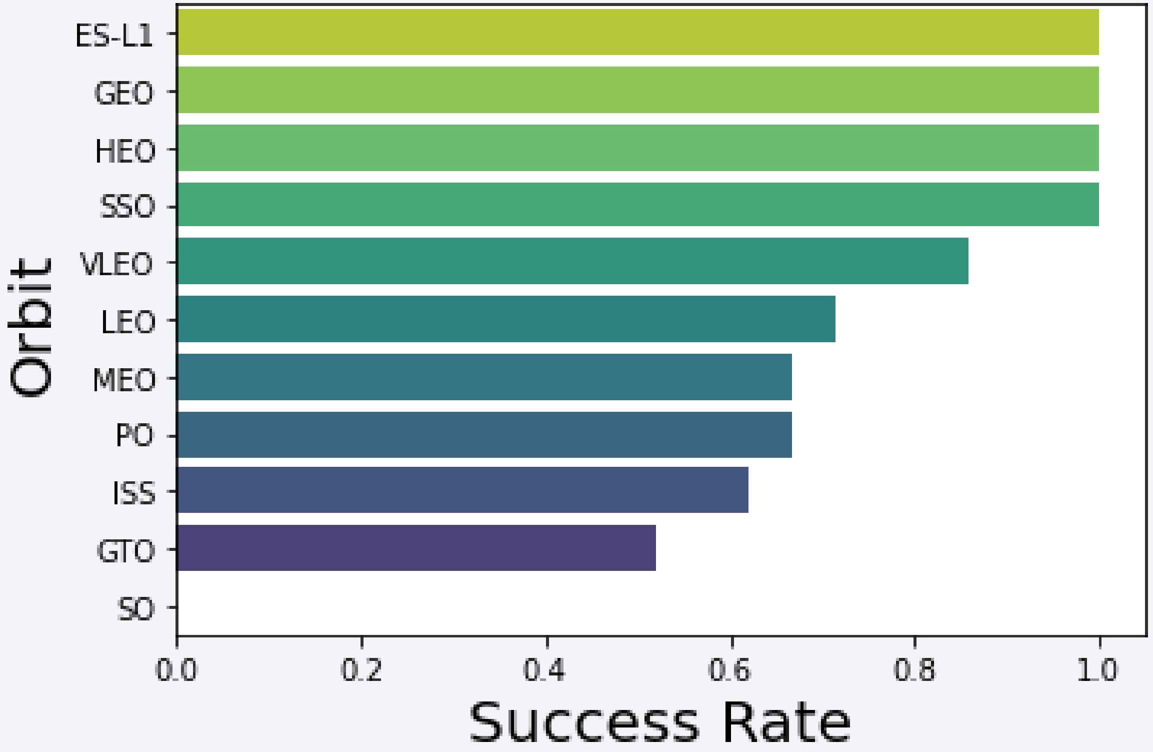
# Success Rate vs. Orbit Type

- Success Rate Scale with

0 as 0%

0.6 as 60%

1 as 100%



ES-L1 (1), GEO (1), HEO (1) have 100% success rate (sample sizes in parenthesis) SSO (5) has 100% success rate

VLEO (14) has decent success rate and attempts

SO (1) has 0% success rate

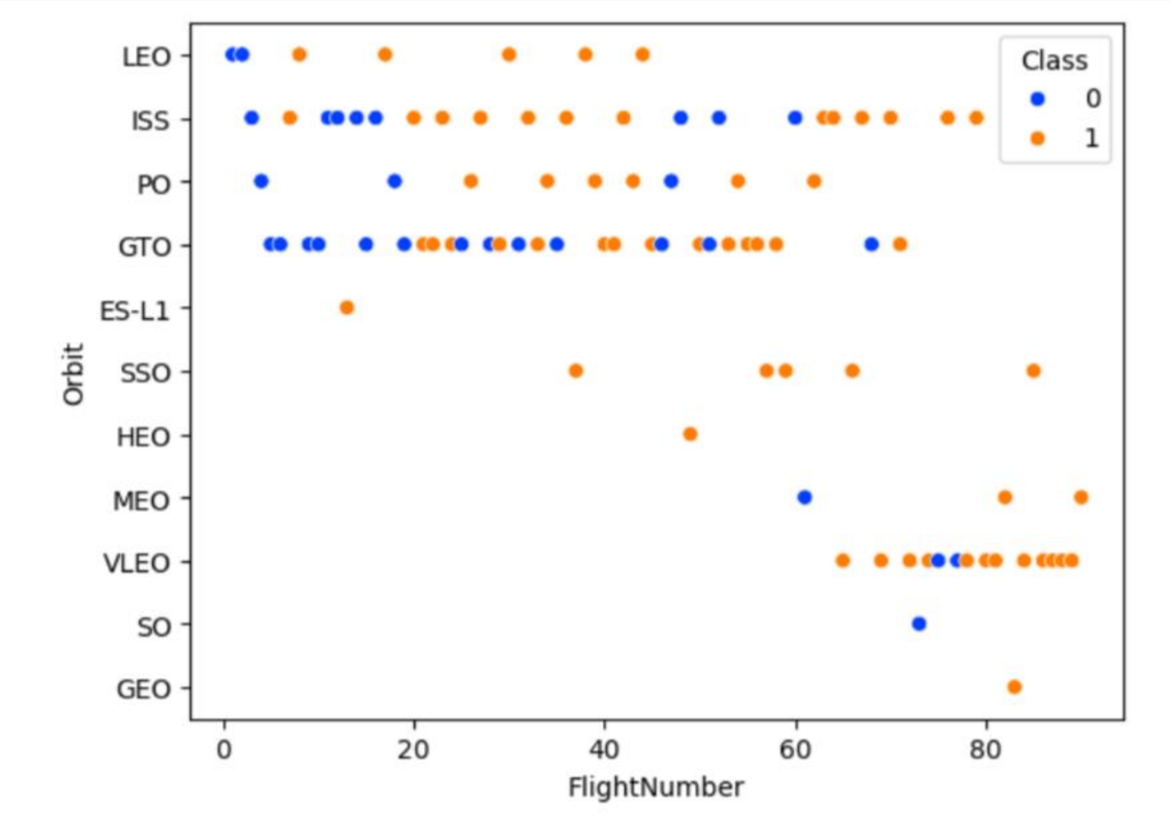
GTO (27) has the around 50% success rate but largest sample

# Flight Number vs. Orbit Type

- Orange indicates successful launch; Blue indicates unsuccessful launch.

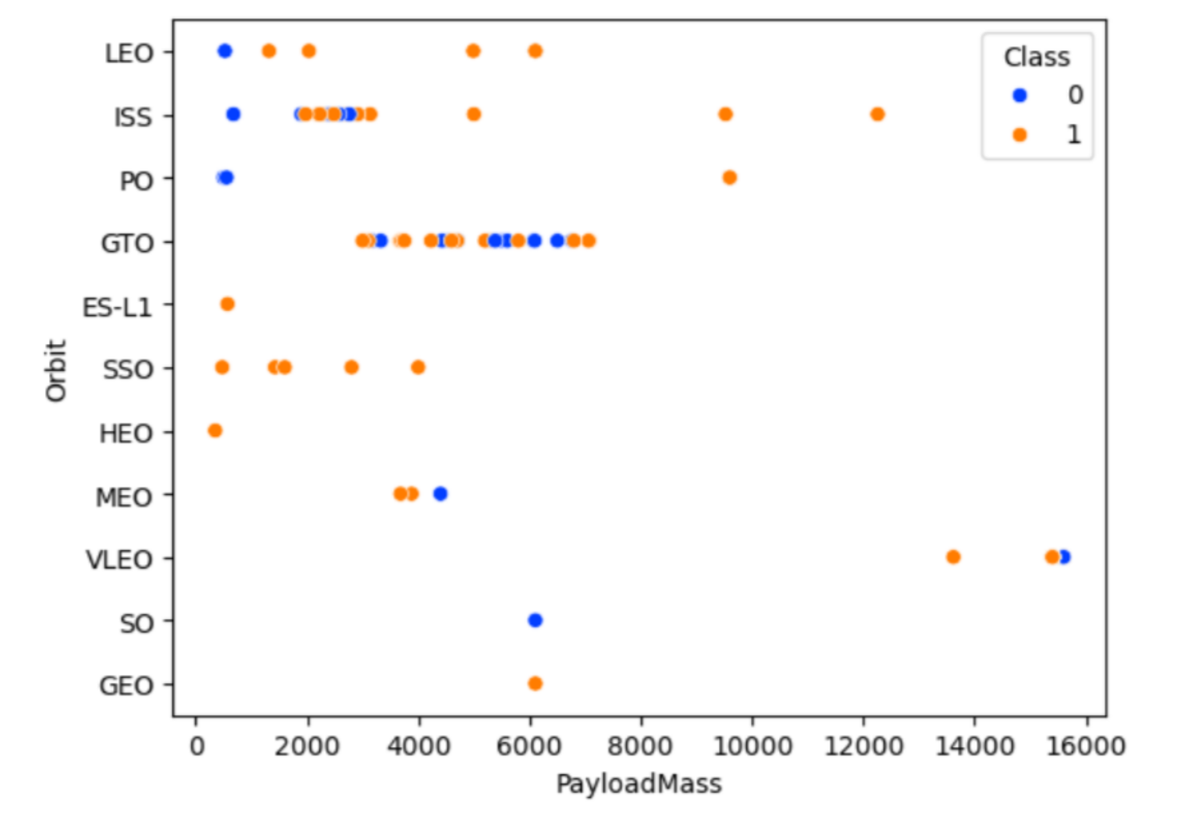
This plot, shows the following:

- The 100% success rate of GEO, HEO, and ES-L1 orbits can be explained by only having 1 flight into the respective orbits.
- The 100% success rate in SSO is more impressive, with 5 successful flights.
- There is little relationship between Flight Number and Success Rate for GTO.
- Generally, as Flight Number increases, the success rate increases. This is most extreme for LEO, where unsuccessful landings only occurred for the low flight numbers (early launches).



# Payload vs. Orbit Type

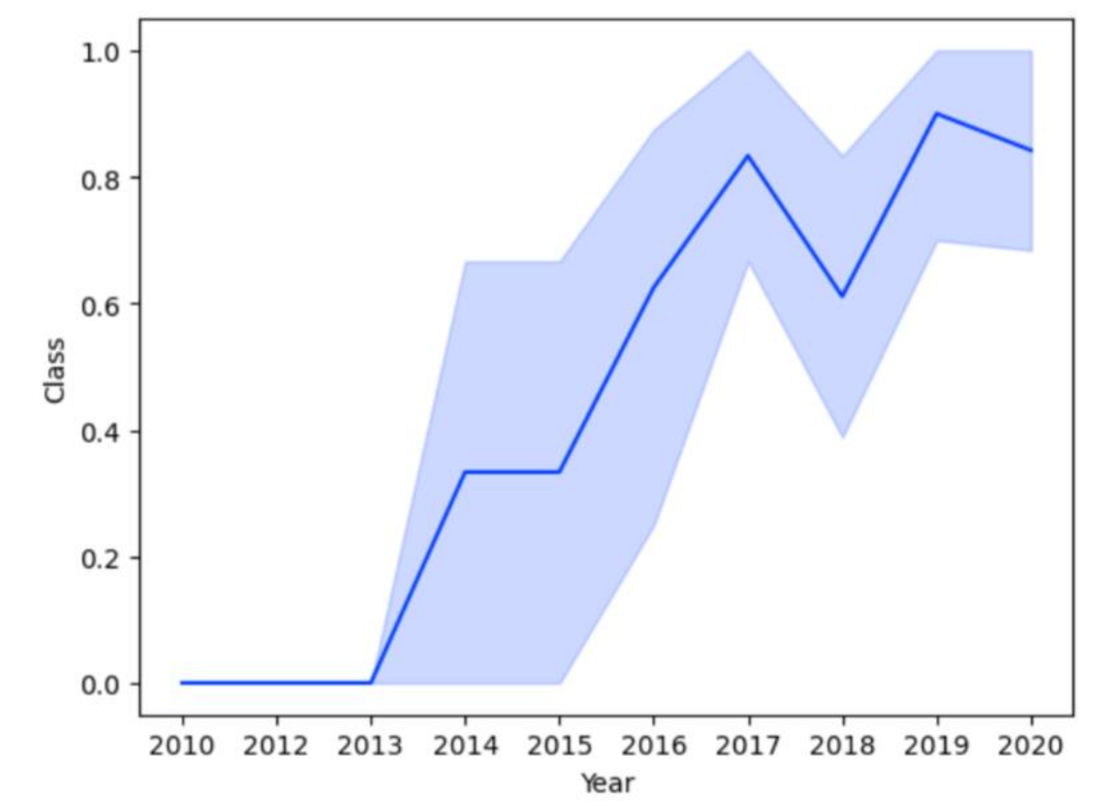
- Orange indicates successful launch; Blue indicates unsuccessful launch.
- This scatter plot of Orbit Type vs. Payload Mass shows that:
- The following orbit types have more success with heavy payloads:
  - PO (although the number of data points is small)
  - ISS
  - LEO
- For GTO, the relationship between payload mass and success rate is unclear.
- VLEO (Very Low Earth Orbit) launches are associated with heavier payloads, which makes intuitive sense.



# Launch Success Yearly Trend

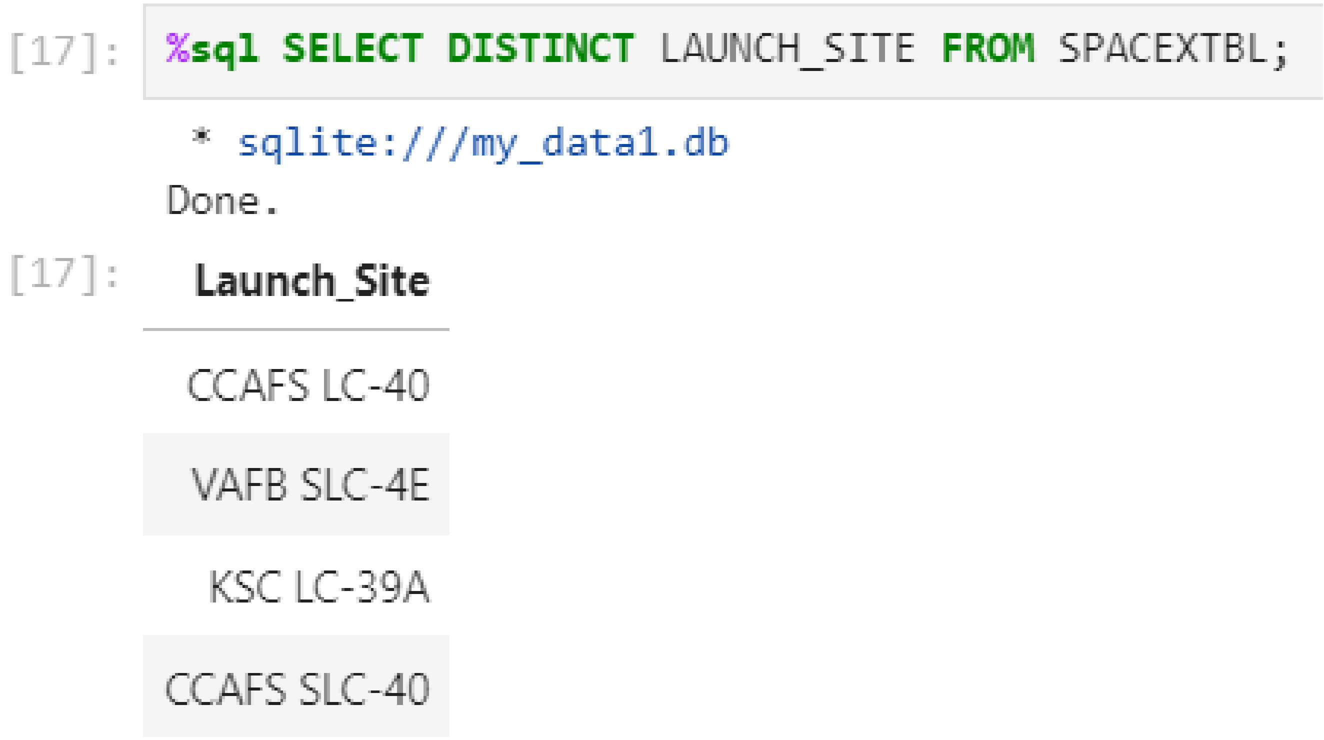
The line chart of yearly average success rate shows that:

- Between 2010 and 2013, all landings were unsuccessful (as the success rate is 0).
- After 2013, the success rate generally increased, despite small dips in 2018 and 2020.
- After 2016, there was always a greater than 50% chance of success.



# All Launch Site Names

Find the names of the unique launch sites



```
[17]: %sql SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL;
```

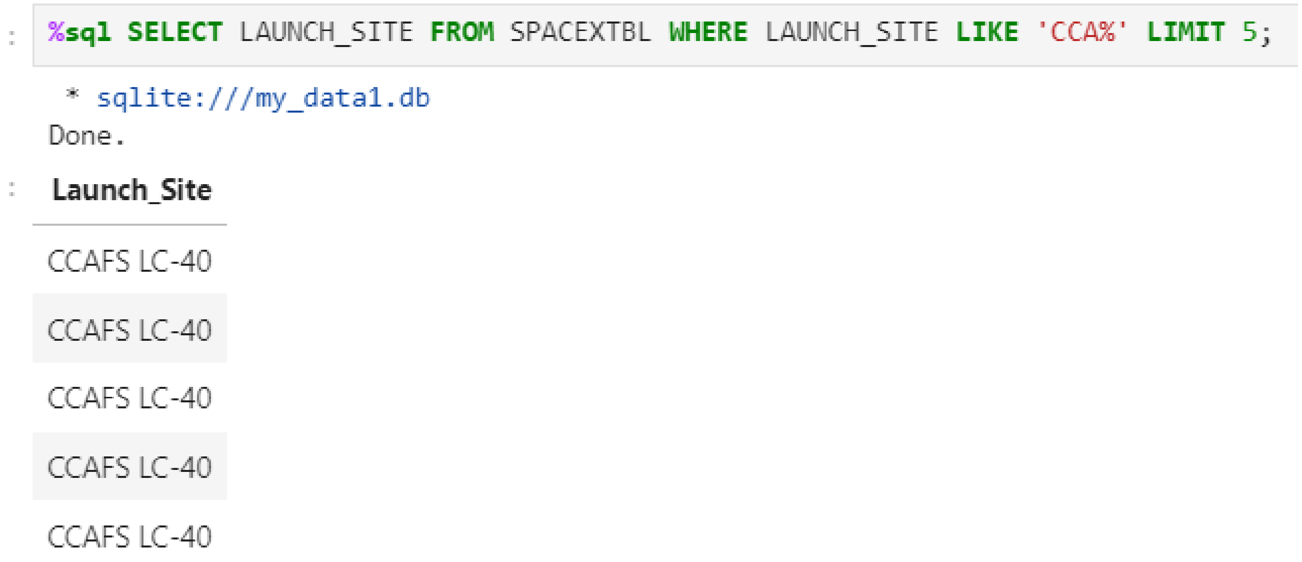
```
 * sqlite:///my_data1.db
Done.
```

[17]:

| 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`
- LIMIT 5 fetches only 5 records, and the LIKE keyword is used with the wild card ‘CCA%’ to retrieve string values beginning with ‘CCA’.



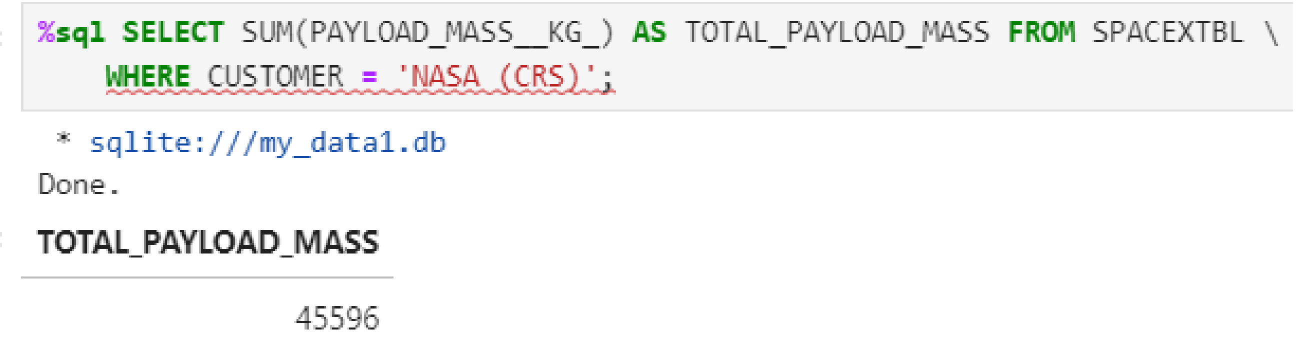
```
%sql SELECT LAUNCH_SITE FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
```

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

| Launch_Site |
|---|
| CCAFS LC-40 |
| CCAFS LC-40 |
| CCAFS LC-40 |
| CCAFS LC-40 |
| CCAFS LC-40 |

# Total Payload Mass

- Calculate the total payload carried by boosters from NASA
- The SUM keyword is used to calculate the total of the LAUNCH column, and the SUM keyword (and the associated condition) filters the results to only boosters from NASA (CRS).

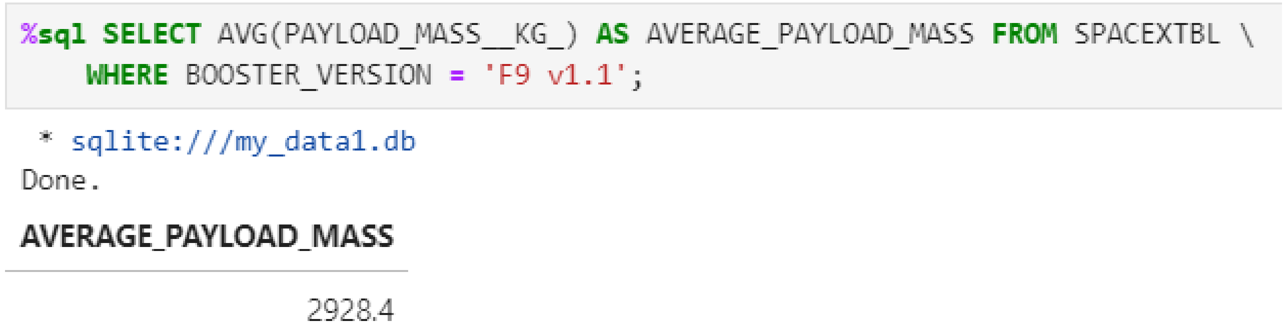


```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS TOTAL_PAYLOAD_MASS FROM SPACEXTBL \
    WHERE CUSTOMER = 'NASA (CRS)';
```

```
 * 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
- The AVG keyword is used to calculate the average of the PAYLOAD_MASS__KG_ column, and the WHERE keyword (and the associated condition) filters the results to only the F9 v1.1 booster version.



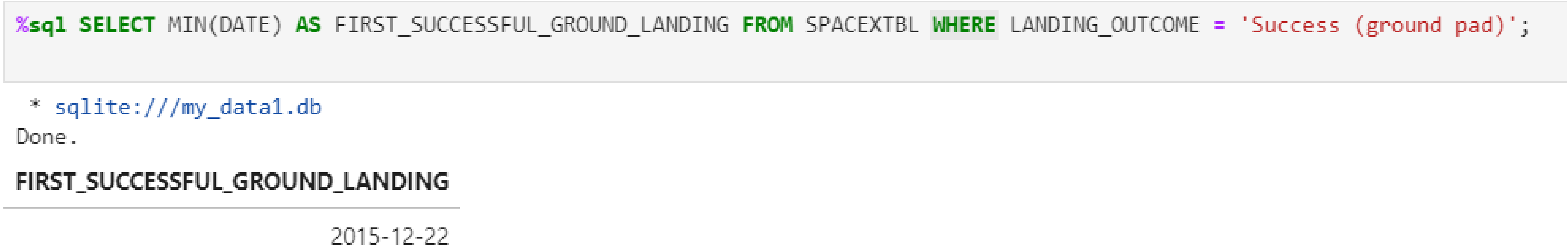
```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS AVERAGE_PAYLOAD_MASS FROM SPACEXTBL \
    WHERE BOOSTER_VERSION = 'F9 v1.1';
```

```
 * 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 ground pad
- The MIN keyword is used to calculate the minimum of the DATE column, i.e. the first date, and the WHERE keyword (and the associated condition) filters the results to only the successful ground pad landings.



```
%sql SELECT MIN(DATE) AS FIRST_SUCCESSFUL_GROUND_LANDING FROM SPACEXTBL WHERE LANDING_OUTCOME = 'Success (ground pad)';
```

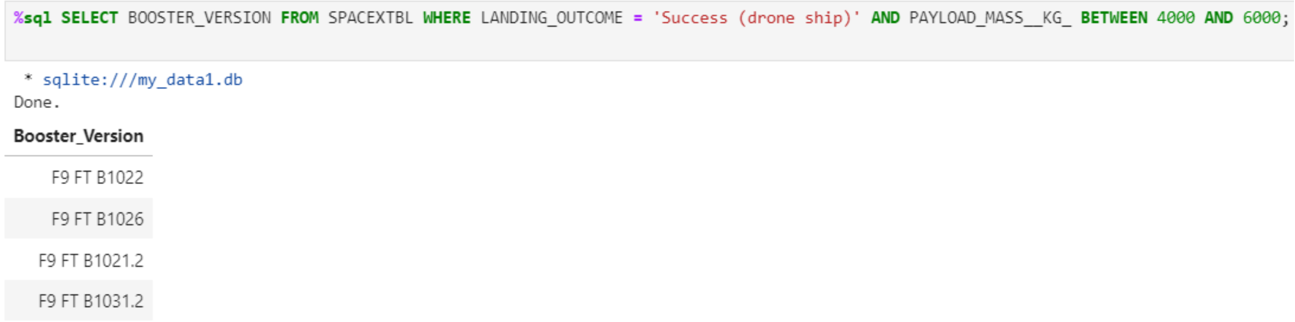
```
 * sqlite:///my_data1.db
Done.
```

| FIRST_SUCCESSFUL_GROUND_LANDING |
|---:|
| 2015-12-22 |

# Successful Drone Ship Landing with Payload between 4000 and 6000

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

- This query returns the four booster versions that had successful drone ship landings and a payload mass between 4000 and 6000 noninclusively.



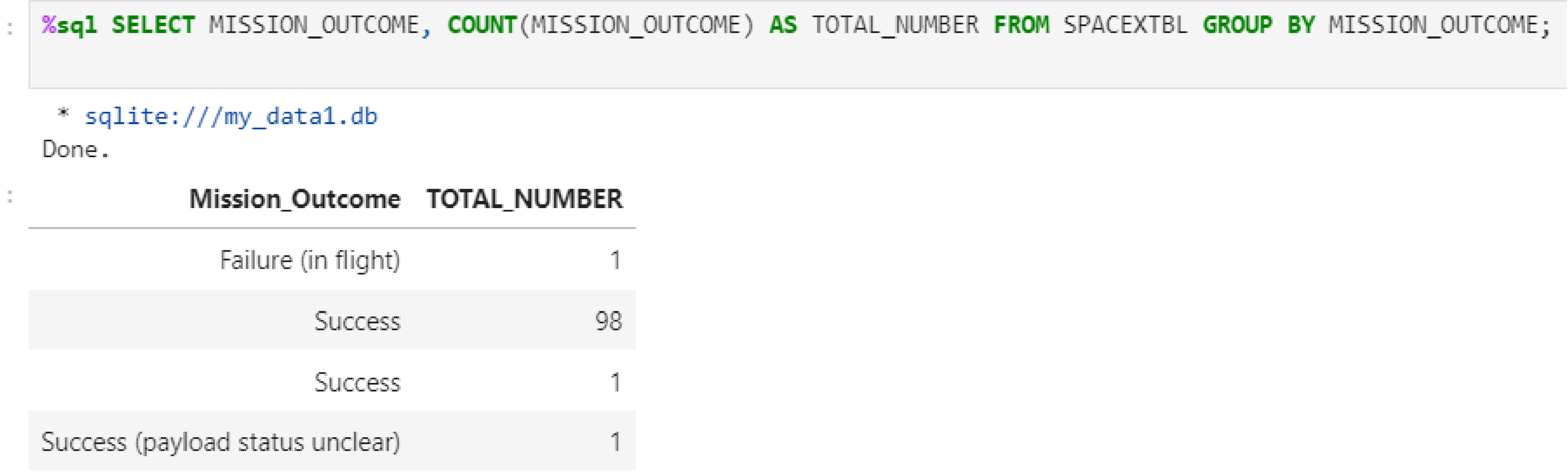
```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE LANDING_OUTCOME = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000;
```

```
 * 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 failure mission outcomes
- The COUNT keyword is used to calculate the total number of mission outcomes, and the GROUPBY keyword is also used to group these results by the type of mission outcome.



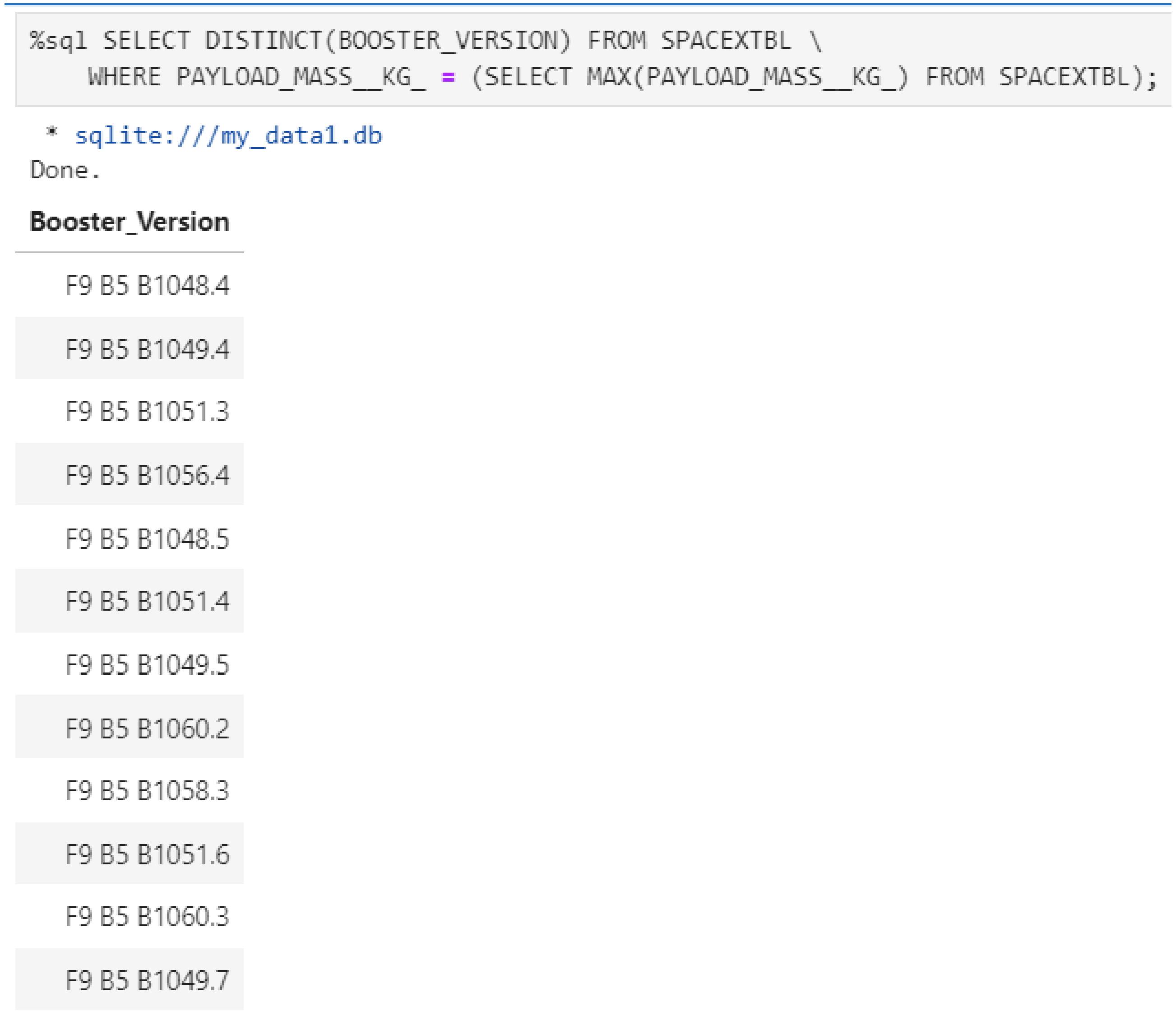
```
%sql SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) AS TOTAL_NUMBER FROM SPACEXTBL GROUP BY MISSION_OUTCOME;
```

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

| Mission_Outcome | TOTAL_NUMBER |
|---:|---:|
| Failure (in flight) | 1 |
| Success | 98 |
| Success | 1 |
| Success (payload status unclear) | 1 |

# Boosters Carried Maximum Payload

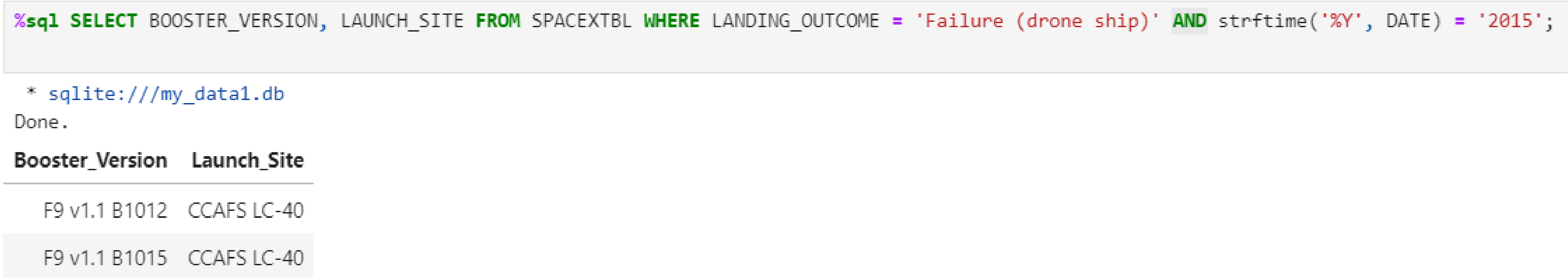
- List the names of the booster which have carried the maximum payload mass
- A subquery is used here. The SELECT statement within the brackets finds the maximum payload, and this value is used in the WHERE condition. The DISTINCT keyword is then used to retrieve only distinct /unique booster versions.



```
%sql SELECT DISTINCT(BOOSTER_VERSION) FROM SPACEXTBL \
    WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```

```
 * 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 |



```
%sql SELECT BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WHERE LANDING_OUTCOME = 'Failure (drone ship)' AND strftime('%Y', DATE) = '2015';
```

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

| Booster_Version | Launch_Site |
|---|---|
| F9 v1.1 B1012 | CCAFS LC-40 |
| F9 v1.1 B1015 | CCAFS LC-40 |

# 2015 Launch Records

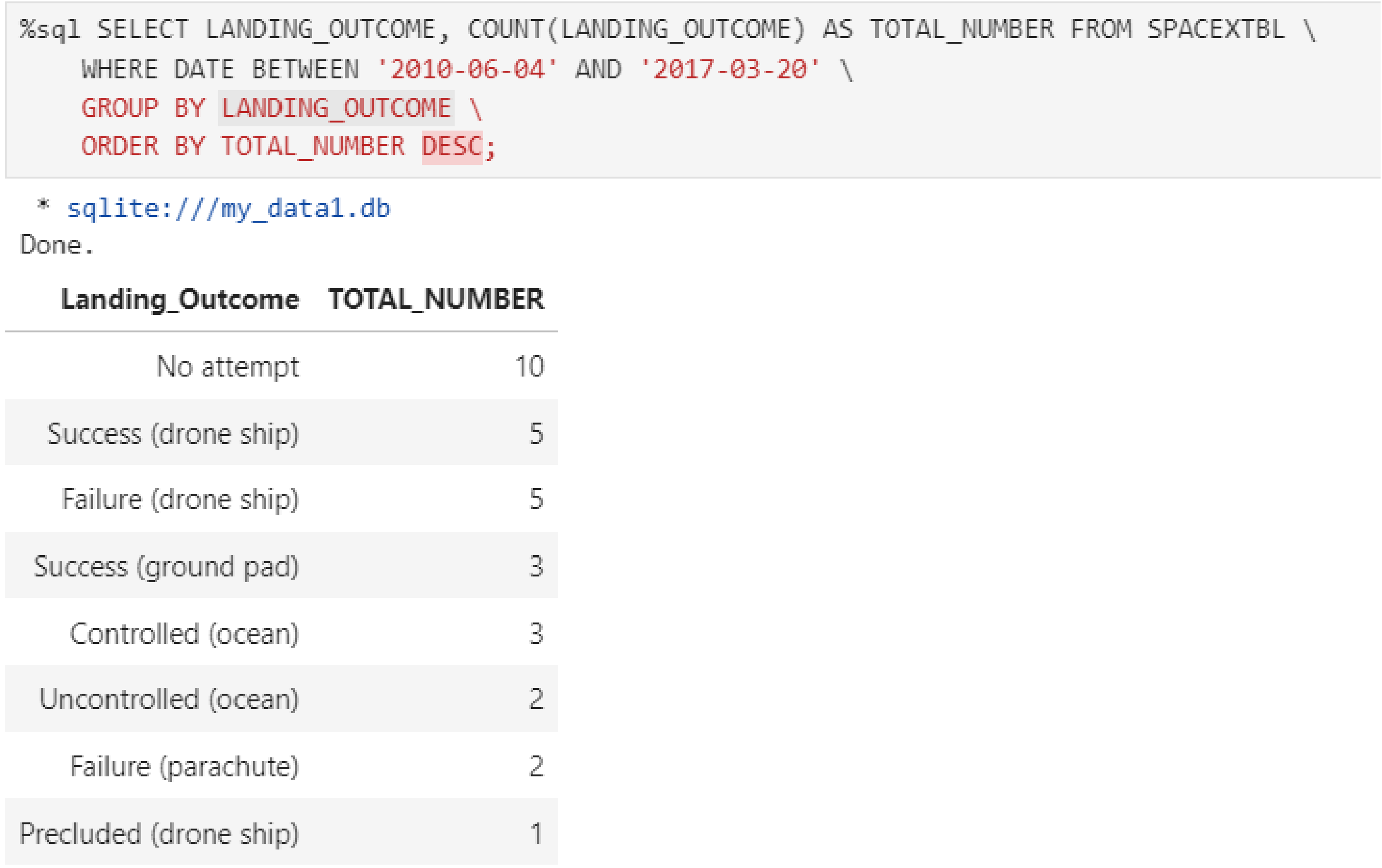
- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

- The WHERE keyword is used to filter the results for only failed landing outcomes, AND only for the year of 2015.

# 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

- The WHERE keyword is used with the BETWEEN keyword to filter the results to dates only within those specified. The results are then grouped and ordered, using the keywords GROUP BY and ORDER BY, respectively, where DESC is used to specify the descending order.



```
%sql SELECT LANDING_OUTCOME, COUNT(LANDING_OUTCOME) AS TOTAL_NUMBER FROM SPACEXTBL \
    WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' \
    GROUP BY LANDING_OUTCOME \
    ORDER BY TOTAL_NUMBER DESC;
```

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

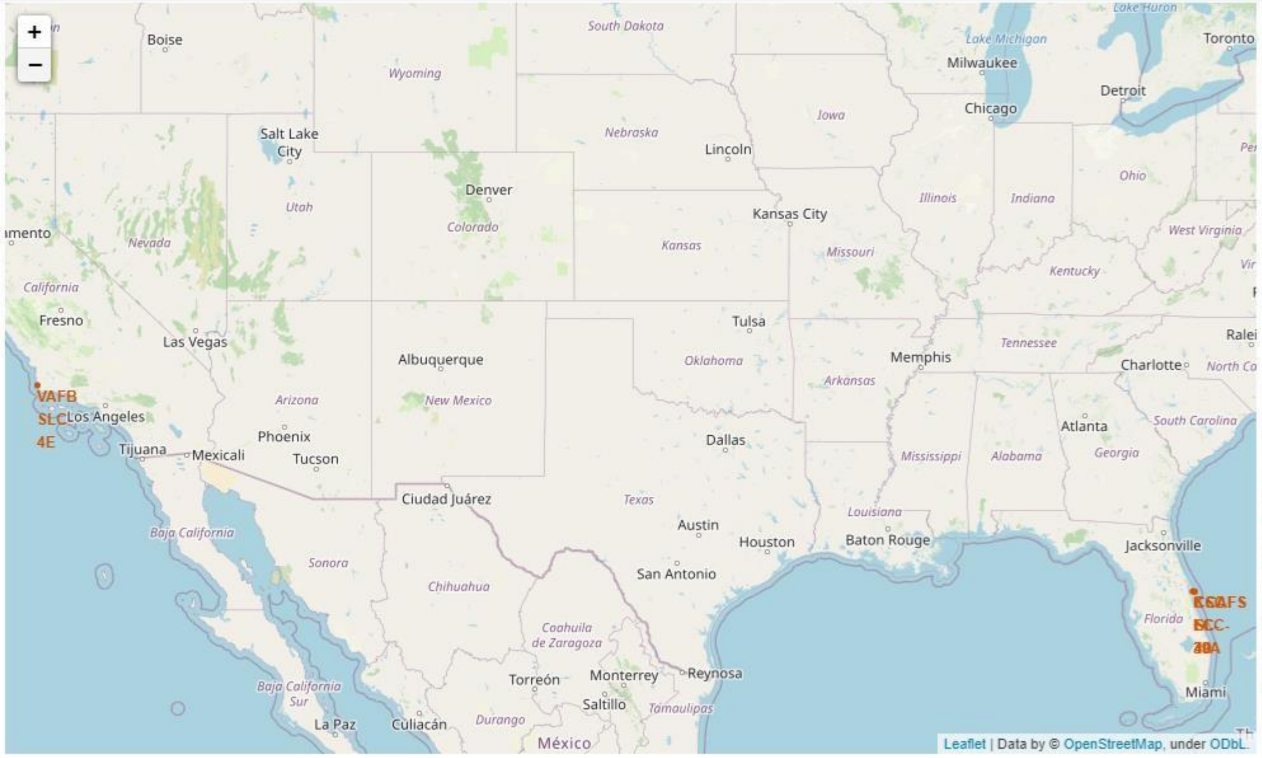
| Landing_Outcome | TOTAL_NUMBER |
|---:|---:|
| No attempt | 10 |
| Success (drone ship) | 5 |
| Failure (drone ship) | 5 |
| Success (ground pad) | 3 |
| Controlled (ocean) | 3 |
| Uncontrolled (ocean) | 2 |
| Failure (parachute) | 2 |
| Precluded (drone ship) | 1 |

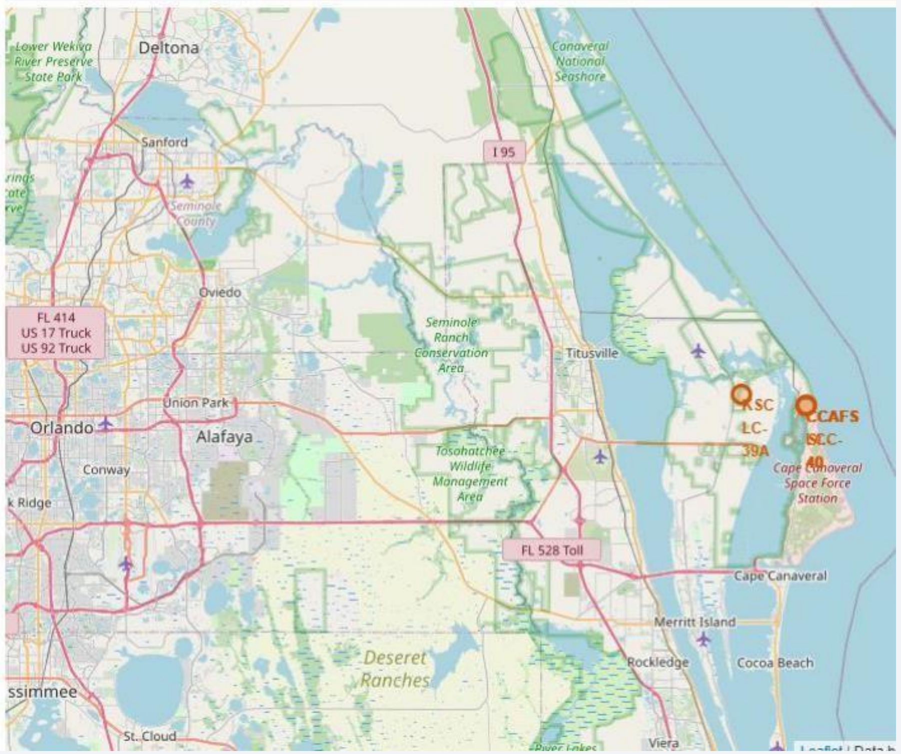
Section 3

# Launch Sites Proximities Analysis

# Launch Site Locations

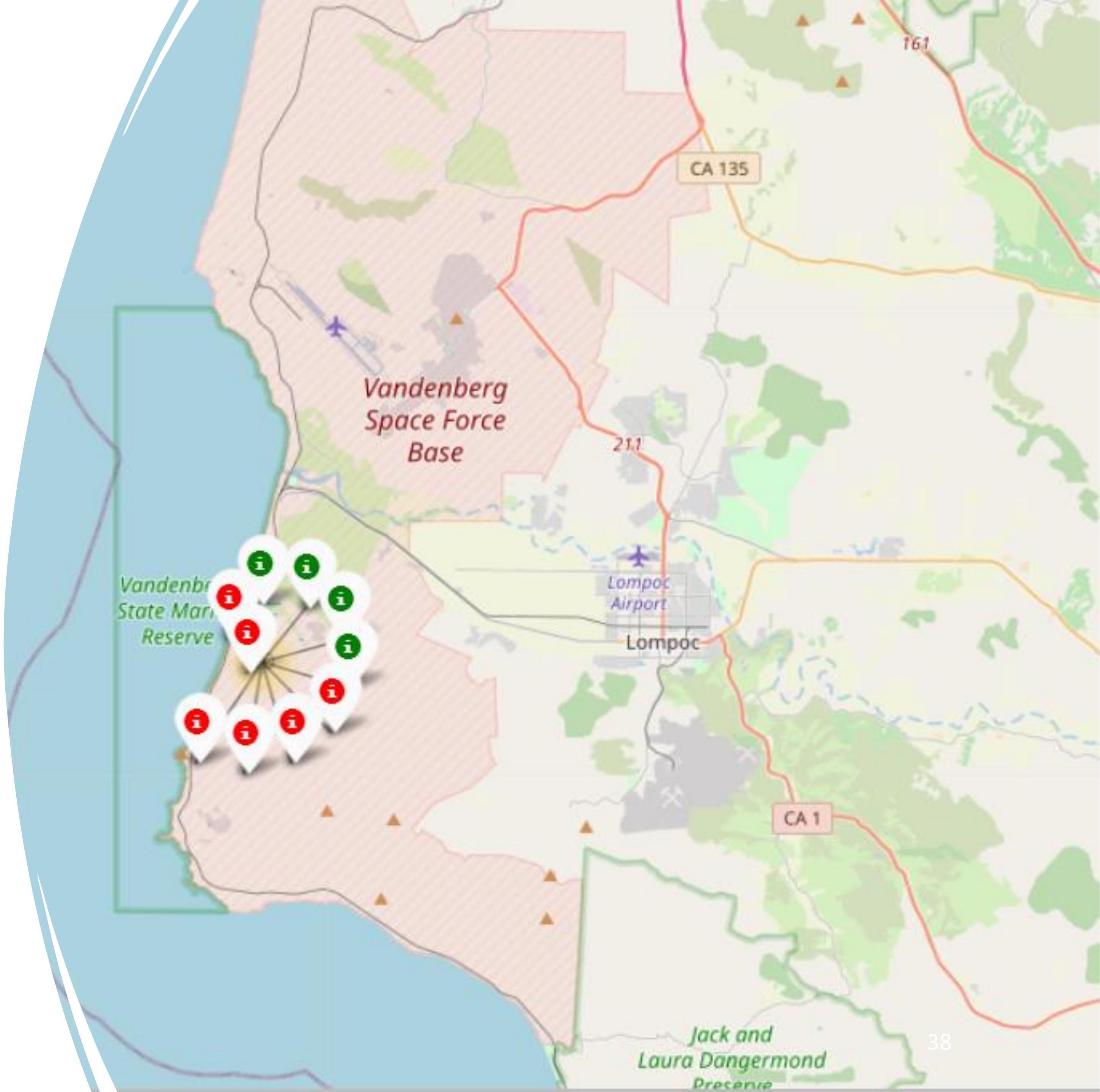
- The left map shows all launch sites relative US map. The right map shows the two Florida launch  sites since they are very close to each other. All launch sites are near the ocean.





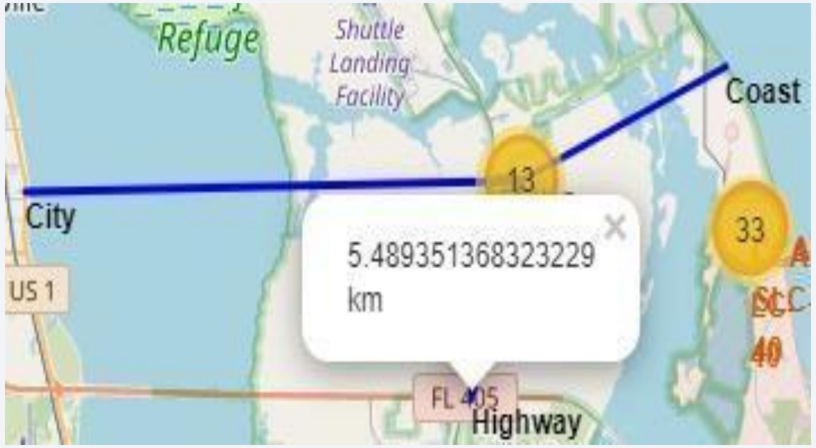
# Color Coded Launch Markers

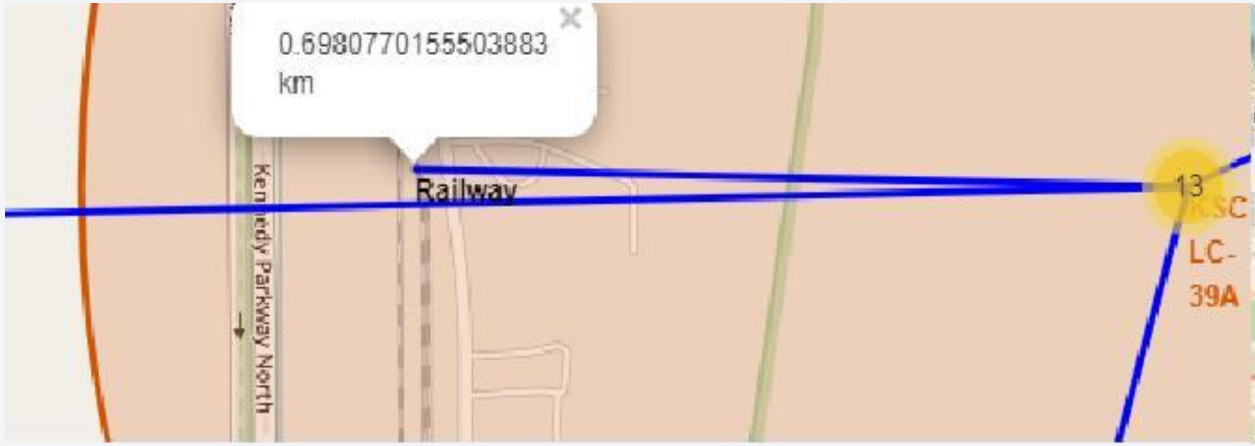
- Clusters on Folium map can be clicked on to display each successful landing (green icon) and failed landing (red icon). In this example VAFB SLC-4E shows 4 successful landings and 6 failed landings.

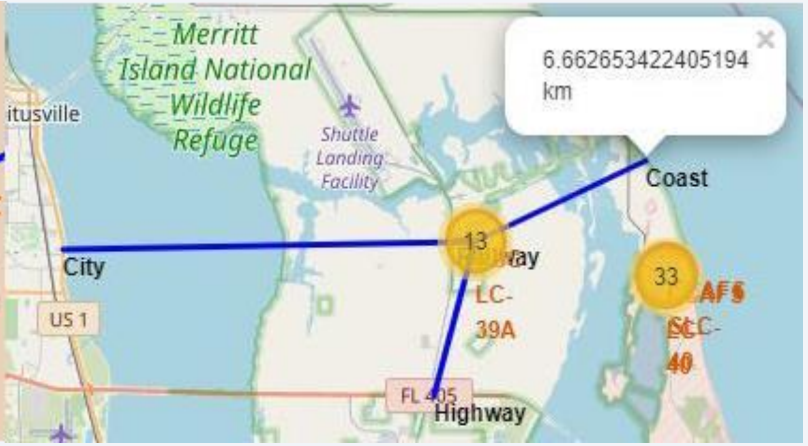


# PROXIMITY OF LAUNCH SITES TO OTHER POINTS OF INTEREST

- 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.

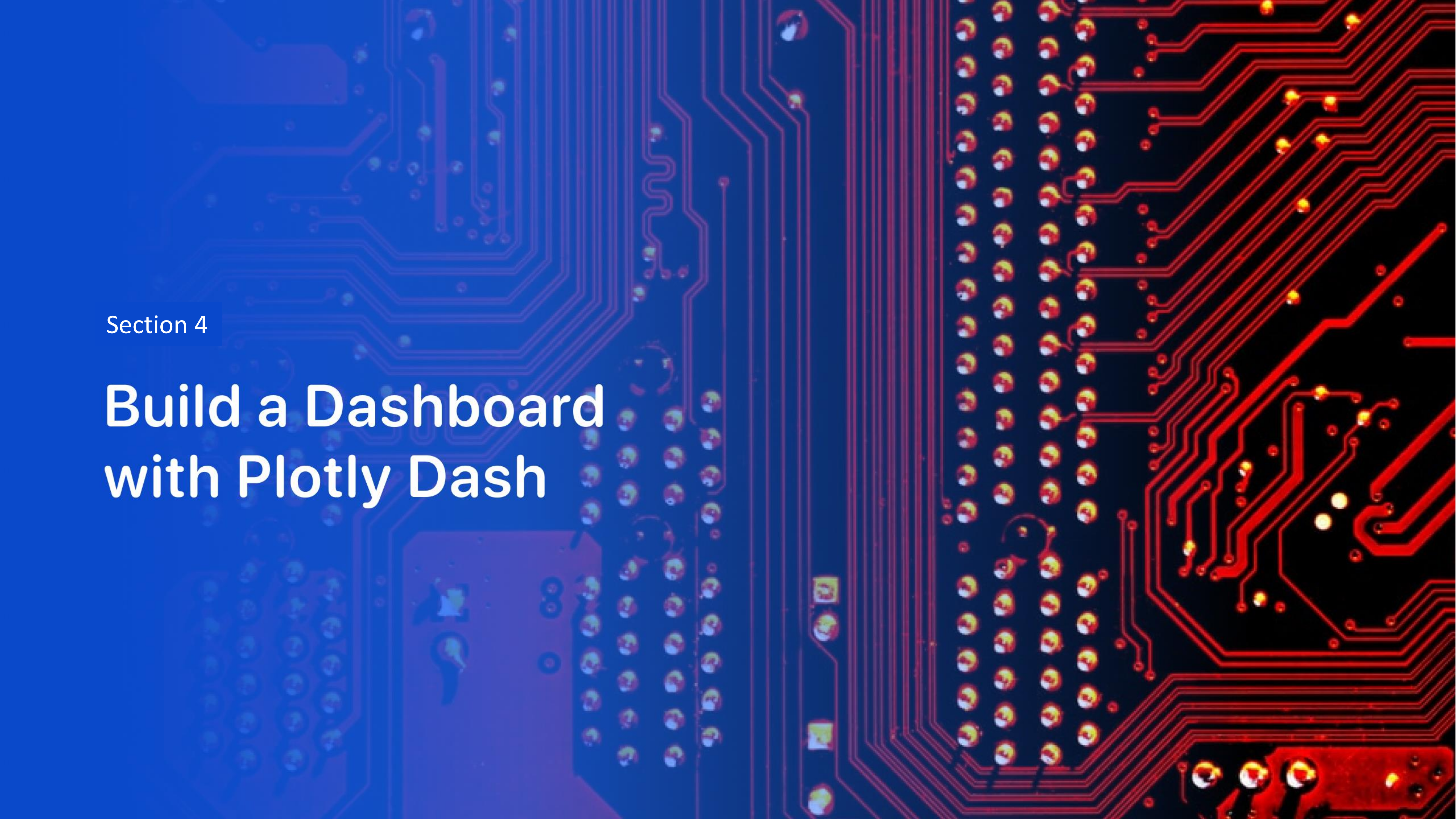






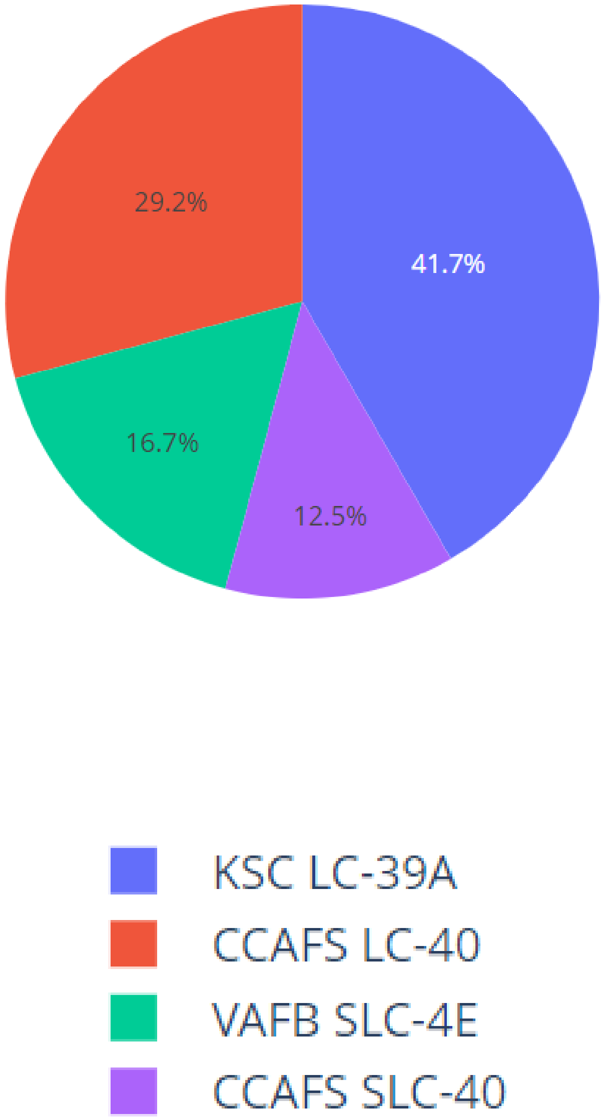
Section 4

# Build a Dashboard with Plotly Dash



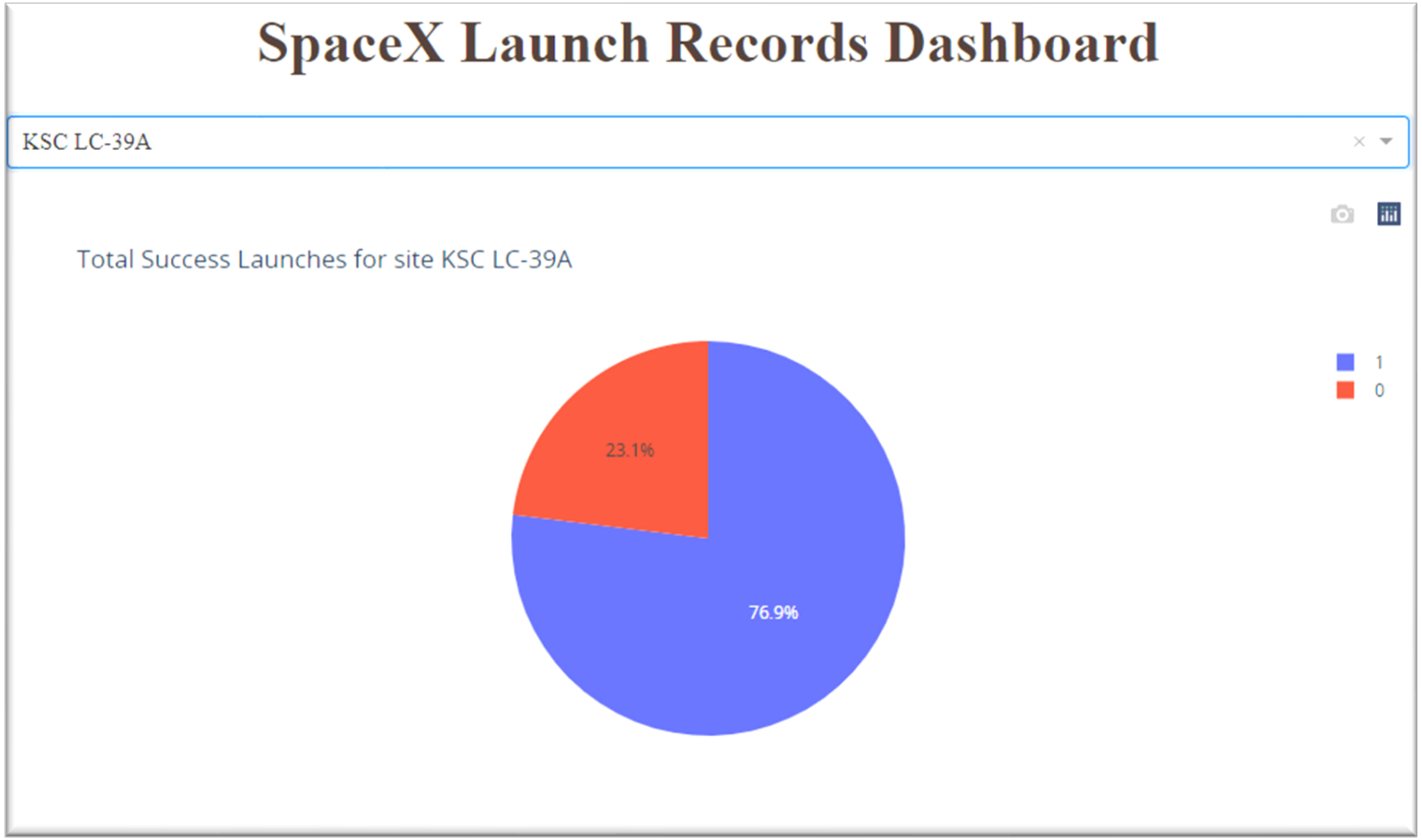
# Successful Launches Across Launch Sites

- 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.



# Pie chart for the launch site with highest launch success ratio

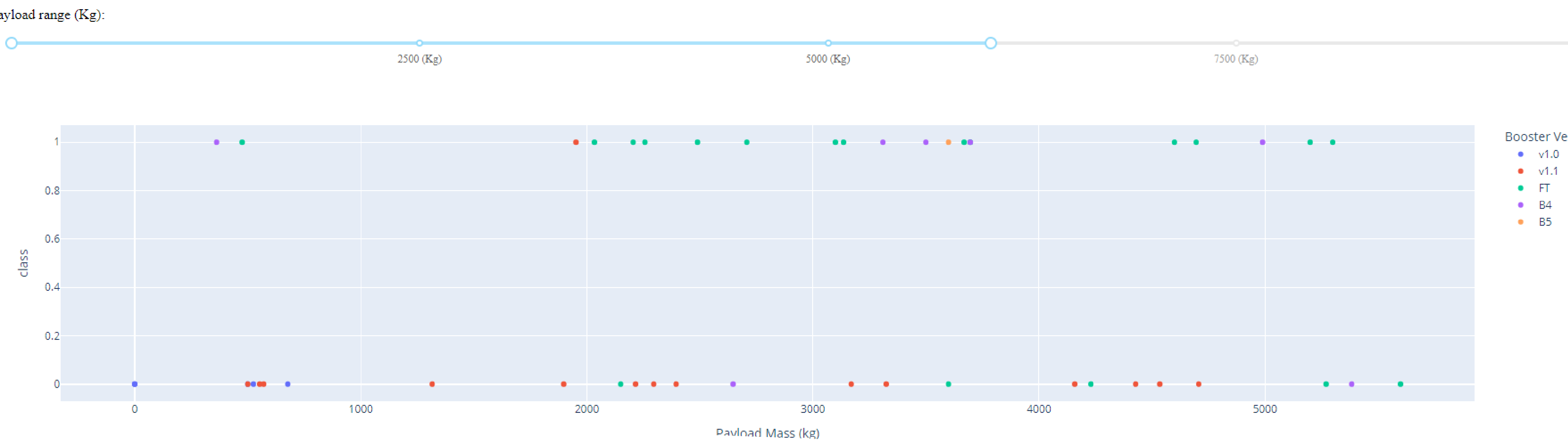
- The launch site KSC LC-39 A also had the highest rate of successful launches, with a 76.9% success rate.



SpaceX Launch Records Dashboard

KSC LC-39A

Total Success Launches for site KSC LC-39A



# Payload Mass vs. Success vs. Booster Version Category

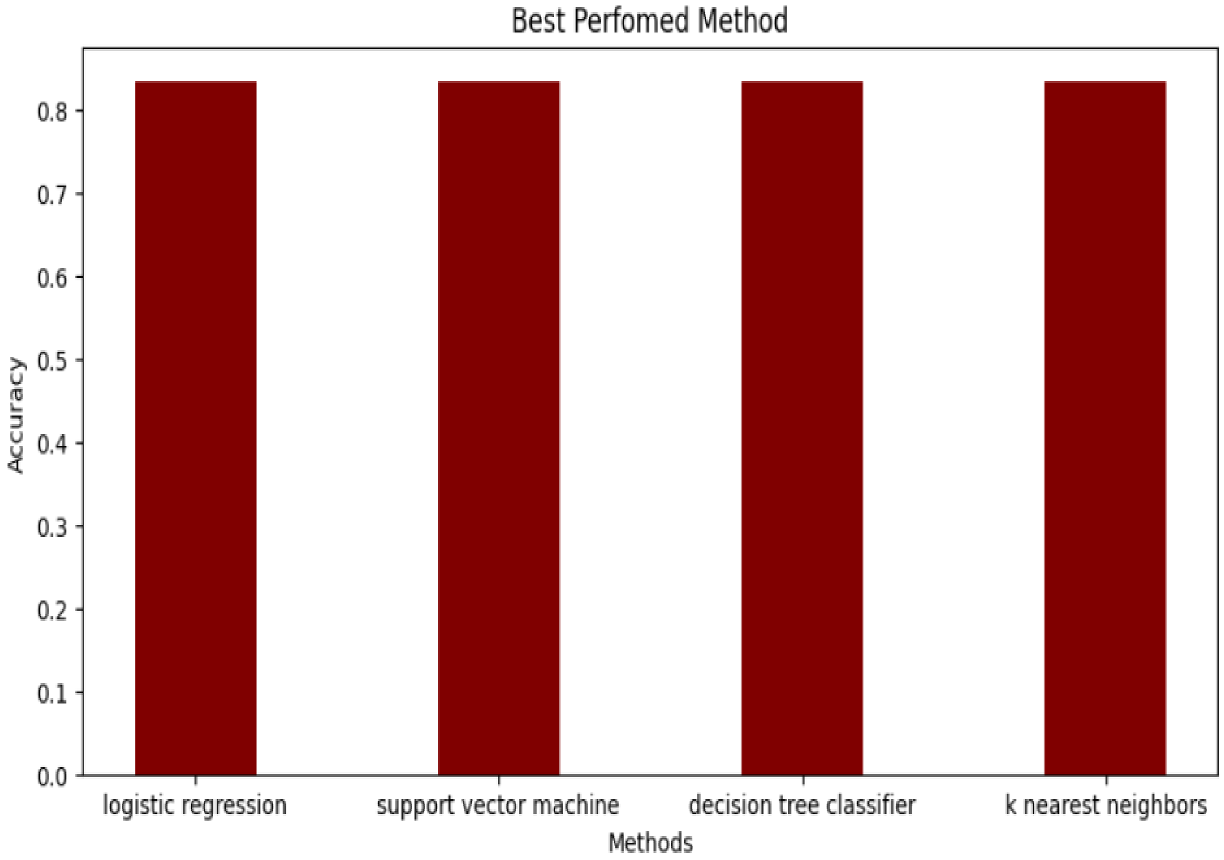
- Plotly dashboard has a Payload range selector. However, this is set from 0-10000 instead of the max Payload of 15600. Class indicates 1 for successful landing and 0 for failure. Scatter plot also accounts for booster version category in color and number of launches in point size. In this particular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

Section 5

# Predictive Analysis (Classification)

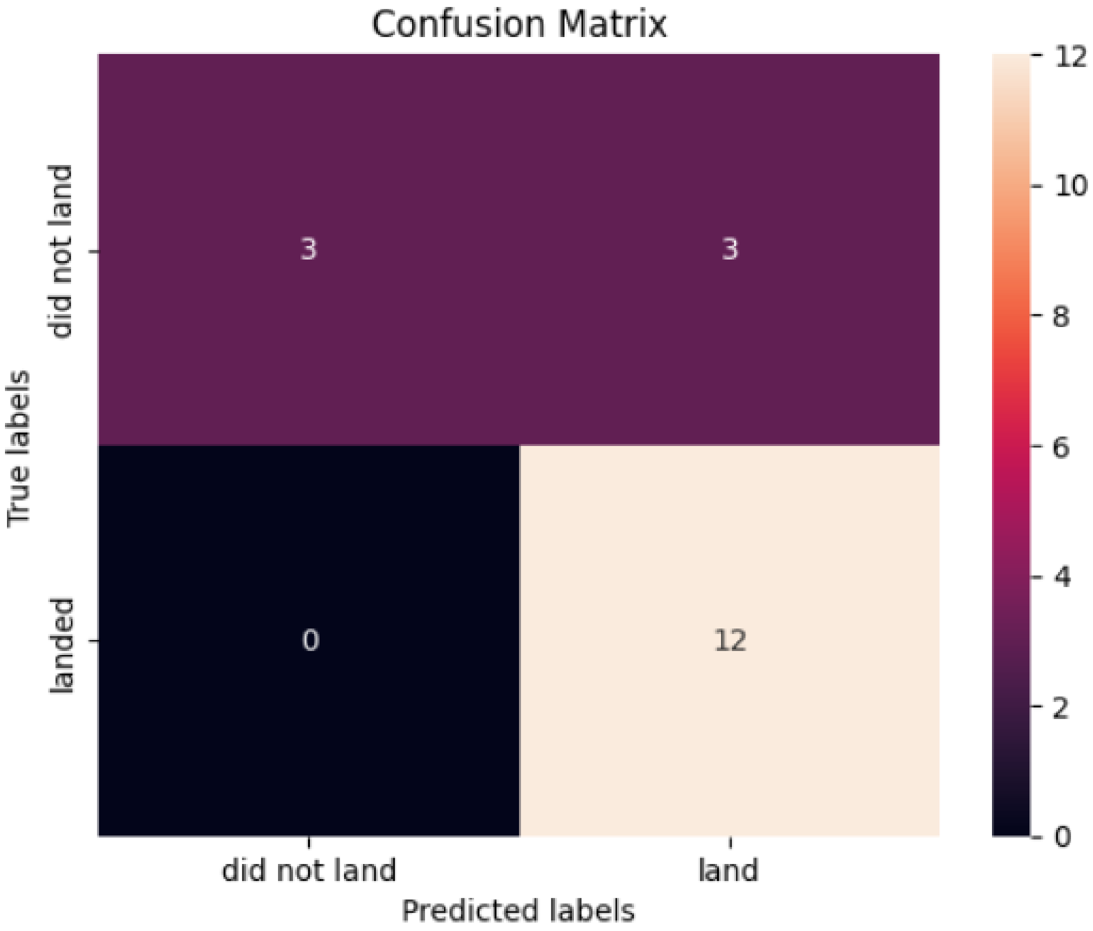
# Classification Accuracy

- 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

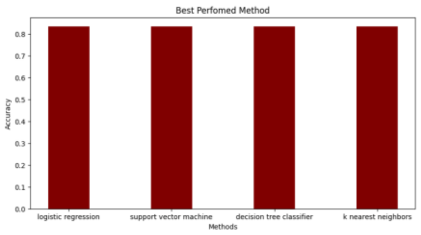
- Since all models performed the same for the test set, the confusion matrix is the same across all models. The models predicted 12 successful landings when the true label  was successful landing.
- The models predicted 3 unsuccessful landings when the true label was unsuccessful  landing.
- The models predicted 3 successful landings when the true label was unsuccessful landings (false positives). Our models over predict successful landings.

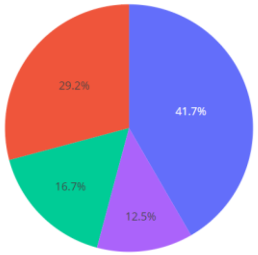


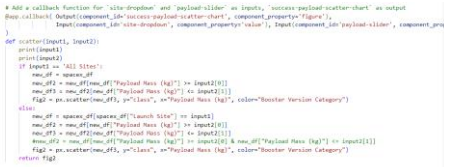
# Conclusions

- As the number of flights increases, the success rate at a launch site increases, with most early flights being unsuccessful. I.e. with more experience, the success rate increases.
- Orbit types ES-L1, GEO, HEO, and SSO, have the highest (100%) success rate.
- The launch site KSC LC-39 A had the most successful launches, with 41.7% of the total successful launches, and also the highest rate of successful launches, with a 76.9% success rate.
- 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
- The success for massive payloads (over 4000kg) is lower than that for low payloads.

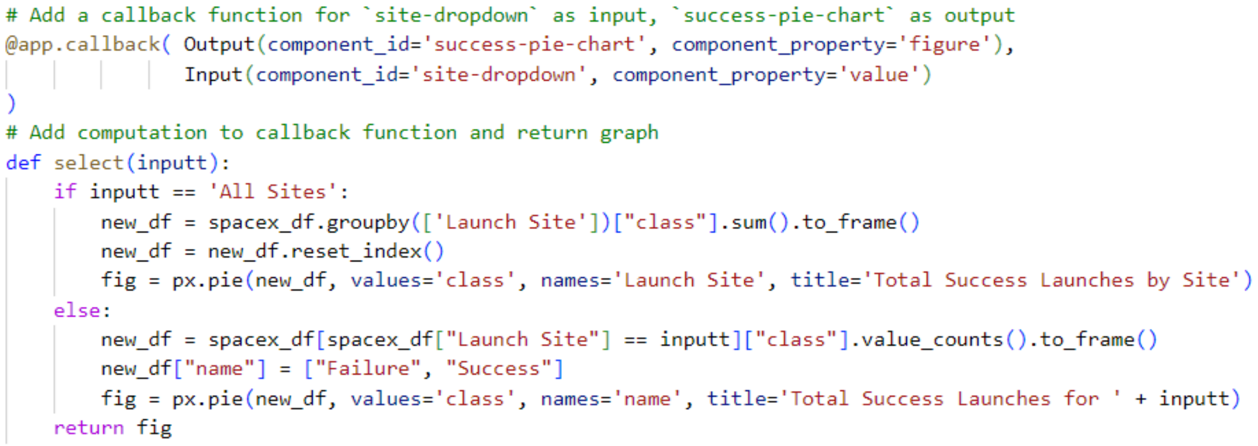
# Appendix







```
# Add a callback function for `site-dropdown` and `payload-slider` as inputs, `success-payload-scatter-chart` as output
@app.callback( Output(component_id='success-payload-scatter-chart', component_property='figure'),
               Input(component_id='site-dropdown', component_property='value'), Input(component_id='payload-slider', component_pro
)
def scatter(input1, input2):
    print(input1)
    print(input2)
    if input1 == 'All Sites':
        new_df = spacex_df
        new_df2 = new_df[new_df["Payload Mass (kg)"] >= input2[0]]
        new_df3 = new_df2[new_df["Payload Mass (kg)"] <= input2[1]]
        fig2 = px.scatter(new_df3, y="class", x="Payload Mass (kg)", color="Booster Version Category")
    else:
        new_df = spacex_df[spacex_df["Launch Site"] == input1]
        new_df2 = new_df[new_df["Payload Mass (kg)"] >= input2[0]]
        new_df3 = new_df2[new_df["Payload Mass (kg)"] <= input2[1]]
        #new_df2 = new_df[new_df["Payload Mass (kg)"] >= input2[0] & new_df["Payload Mass (kg)"] <= input2[1]]
        fig2 = px.scatter(new_df3, y="class", x="Payload Mass (kg)", color="Booster Version Category")
    return fig2
```



```
# Add a callback function for `site-dropdown` as input, `success-pie-chart` as output
@app.callback( Output(component_id='success-pie-chart', component_property='figure'),
               Input(component_id='site-dropdown', component_property='value')
)
# Add computation to callback function and return graph
def select(inputt):
    if inputt == 'All Sites':
        new_df = spacex_df.groupby(['Launch Site'])["class"].sum().to_frame()
        new_df = new_df.reset_index()
        fig = px.pie(new_df, values='class', names='Launch Site', title='Total Success Launches by Site')
    else:
        new_df = spacex_df[spacex_df["Launch Site"] == inputt]["class"].value_counts().to_frame()
        new_df["name"] = ["Failure", "Success"]
        fig = px.pie(new_df, values='class', names='name', title='Total Success Launches for ' + inputt)
    return fig
```

# Thank you!