IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Cristian SORIANO
19 January 2025

Outline

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

Executive Summary

- Summary of methodologies
    - Data collection with API
    - Data Collection with Web Scrapping
    - Data Wrangling
    - Exploratory Data Analysis (EDA) with SQL
    - EDA with Visualization
    - Interactive Visual Analytics with Folium
    - Build an Interactive Dashboard with Ploty Dash
    - Machine Learning Prediction
- Summary of all results

Introduction

- Project background and context

  The goal of this capstone project is to predict whether the Falcon 9 first stage will successfully land after launch. SpaceX has significantly reduced the cost of space missions by reusing the first stage of its Falcon 9 rocket, bringing launch costs down to $62 million compared to competitors charging over $165 million. By accurately predicting landing success, we can better estimate launch costs and provide insights for potential competitors in the space industry.

- Problems you want to find answers

- What factors influence the successful landing of the Falcon 9 first stage?
- Can we build a reliable predictive model to determine landing success?
- How does landing success impact the overall cost-effectiveness of a space launch?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using the SpaceX REST API and webscrapping from Wikkipedia
- Perform data wrangling
  - Data was processed through data wrangling, which involves cleaning, transforming, and organizing raw data into a usable format.
- 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 by preprocessing data, tuning hyperparameters, and evaluating performance using metrics like accuracy and confusion matrix.

Data Collection

Data sets were collected through two main methods:

- **SpaceX REST API**: By performing GET requests to the endpoint api.spacexdata.com/v4/launches/past, JSON formatted data about past launches was retrieved and normalized into a flat table.

- **Web Scraping**: The Python BeautifulSoup package was used to scrape HTML tables from related Wiki pages containing Falcon 9 launch records, which were the

Data Collection – SpaceX API

- Make get request to the SpaceX API + some basic data wrangling

- GitHub URL: https://github.com/cysorianoc/IBM_Data_Science/blob/main/CAPSTONE/CAPSTONE_P1_jupyter_labs_spacex_data_collection_api_v2.ipynb

Now let's start requesting rocket launch data from SpaceX API with the following URL:

```
[ ]  spacex_url="https://api.spacexdata.com/v4/launches/past"
```

```
[ ]  response = requests.get(spacex_url)
```

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

```
[ ]  # Use json_normalize meethod to convert the json result into a dataframe
     json_data = response.json()
     data = pd.json_normalize(json_data)
```

Finally we will remove the Falcon 1 launches keeping only the Falcon 9 launches. Filter the data dataframe using the `BoosterVersion` column to only keep the Falcon 9 launches. Save the filtered data to a new dataframe called `data_falcon9`.

```
[ ]  # Hint data['BoosterVersion']!='Falcon 1'
     data_falcon9 = data[data['BoosterVersion']!='Falcon 1']
```

Data Collection - Scraping

- Webscraping from a Wikipedia page using BeautifulSoup

- GitHub URL:

https://github.com/cysorianoc/IBM_Data_Science/blob/main/CAPSTONE/CAPSTON_P2_jupyter_labs_webscraping.ipynb

- Extract the column names from the HTML table headers using the extract_column_from_header() function.

- Create an empty dictionary with the extracted column names as keys.

- Iterate through the HTML table rows and extract the relevant data for each column.

- Fill in the dictionary with the extracted data for each row.

- Convert the dictionary to a Pandas dataframe using the pd.DataFrame() function.

```
import pandas as pd
from bs4 import BeautifulSoup

# Assume you have the HTML content in a variable called 'html'
soup = BeautifulSoup(html, 'html.parser')

# Extract column names from table headers
column_names = []
for th in soup.find_all('th'):
    column_names.append(extract_column_from_header(th))

# Create an empty dictionary with column names as keys
launch_dict = {column: [] for column in column_names}

# Iterate through table rows and extract data
for row in soup.find_all('tr'):
    # Extract data for each column
    data = []
    for td in row.find_all('td'):
        data.append(td.text.strip())

    # Fill in the dictionary with extracted data
    for i, column in enumerate(column_names):
        launch_dict[column].append(data[i])

# Convert dictionary to Pandas dataframe
df = pd.DataFrame(launch_dict)

# Export dataframe to CSV
df.to_csv('spacex_web_scraped.csv', index=False)
```

Data Wrangling

- Data processing

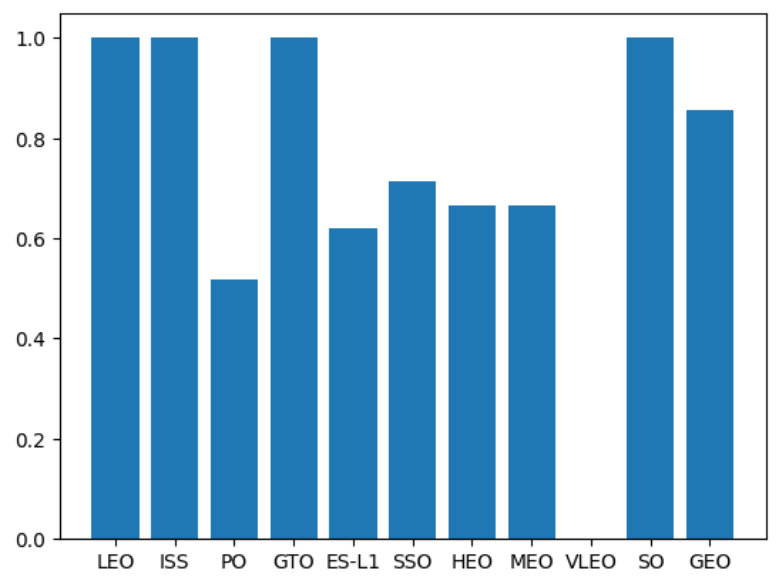
1. Calculate the number of launches using value_counts()
2. Calculate the number and occurrence of each orbit with value_counts()
3. Calculate the number and occurrence of mission outcomes with value_counts()
4. Create a landing outsome 0 or 1
5. Save as .csv

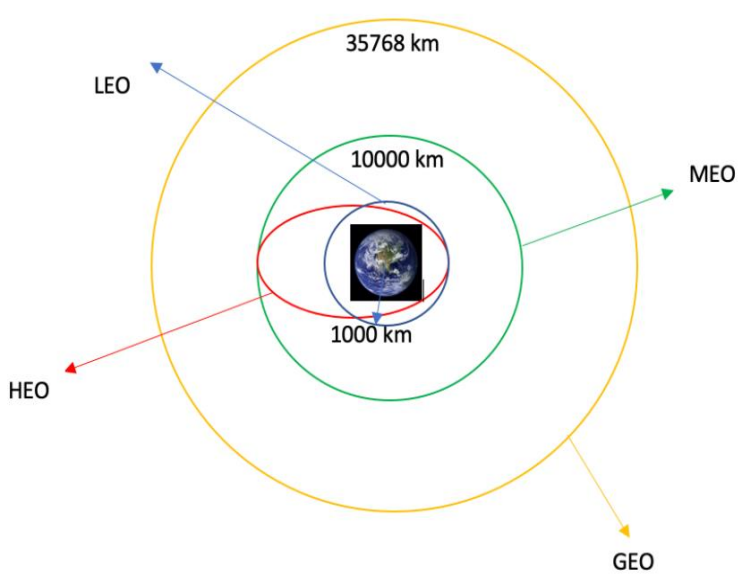
- GitHub URL: https://github.com/cysorianoc/IBM_Data_Science/blob/main/CAPSTONE/CAPSTONE_P3_spacex_Data_wrangling_v2.ipynb

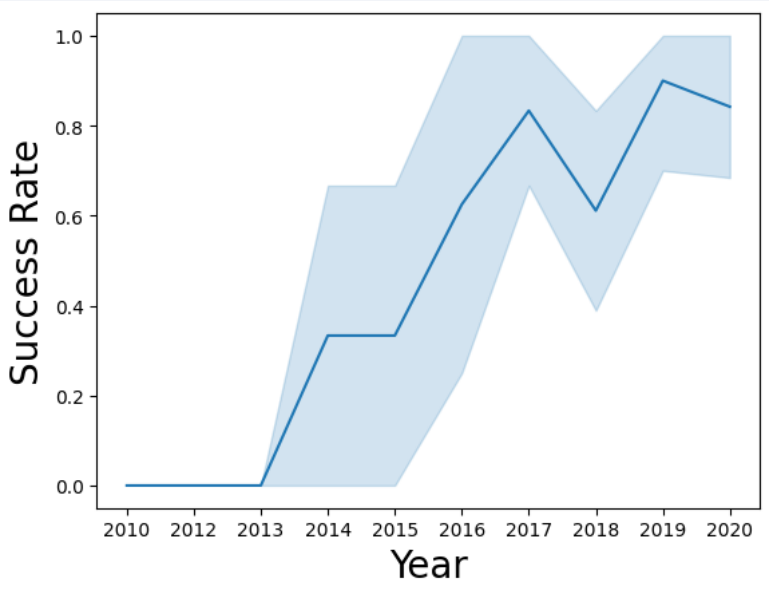
```python
import pandas as pd

# Load the dataset
df = pd.read_csv("your_dataset.csv")

# Calculate the percentage of missing values in each attribute
missing_values = df.isnull().sum()/len(df)*100
print(missing_values)

# Identify the data types of each column
print(df.dtypes)

# Calculate the number of launches on each site
launch_sites = df['LaunchSite'].value_counts()
print(launch_sites)

# Calculate the number and occurrence of each orbit
orbits = df['Orbit'].value_counts()
print(orbits)

# Calculate the number and occurrence of mission outcomes
landing_outcomes = df['Outcome'].value_counts()
print(landing_outcomes)

# Create a landing outcome label from the 'Outcome' column
bad_outcomes = set(landing_outcomes.keys()[[1, 3, 5, 6, 7]])
landing_class = []
for i in df['Outcome']:
    if i in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
df['Class'] = landing_class

# Calculate the success rate of landings
success_rate = df['Class'].mean()
print(success_rate)

# Export the dataset to a CSV file
df.to_csv("dataset_part_2.csv", index=False)
```

EDA with Data Visualization

- Success rates per orbit type and trends od success rate with time







- GitHub URL: https://github.com/cysorianoc/IBM_Data_Science/blob/main/CAPSTONE/CAPSTONE_P5_labs_eda_dataviz_v2.ipynb

EDA with SQL

SQL Queries Conducted

- Unique Launch Sites, Launch Sites Starting with 'CCA', Total Payload Mass for NASA (CRS), Average Payload Mass for Booster Version F9 v1.1, First Successful Ground Pad Landing, Boosters with Successful Drone Ship Landings, Count of Successful and Failed Missions, Booster Versions with Maximum Payload Mass, Failure Landing Outcomes on Drone Ships, Ranking of Landing Outcomes

- GitHub URL: https://github.com/cysorianoc/IBM_Data_Science/blob/main/CAPSTONE/CAPSTONE_P4_eda_sql_coursera_sqllite.ipynb

Build an Interactive Map with Folium

- We mapped all launch sites and added visual markers like circles and lines on the Folium map to indicate the success or failure of launches at each site.
- We categorized launch outcomes (failure or success) into two classes: 0 for failure and 1 for success.
- By analyzing color-coded marker clusters, we determined which launch sites had higher success rates.
- We measured the distances between launch sites and nearby features. This helped us answer questions such as:

  Are launch sites located near railways, highways, or coastlines?

  Do launch sites maintain a specific distance from urban areas?

- GitHub URL: https://github.com/cysorianoc/IBM_Data_Science/blob/main/CAPSTONE/CAPSTONE_P6_launch_site_location_v2.ipynb

Build a Dashboard with Plotly Dash

Plots and Interactions in the Dashboard

- **Pie Chart (Launch Success Rate per Site)**
  - Shows the proportion of successful vs. failed launches for all sites or a selected site.
  - Helps visualize overall success rates and compare launch sites.
- **Scatter Plot (Payload vs. Launch Success)**
  - Displays the relationship between payload mass and launch success.
  - Allows users to analyze trends and identify optimal payload ranges.
- **Interactive Dropdown (Launch Site Selection)**
  - Users can filter data by a specific launch site or view all sites.
  - Enables focused analysis on individual locations.
- **Payload Range Slider**
  - Allows filtering launches based on payload mass.
  - Helps examine how payload size impacts success probability.

- GitHub URL: https://github.com/cysorianoc/IBM_Data_Science/blob/main/CAPSTONE/Capstone_P7_dash.ipynb

Predictive Analysis (Classification)

1.Building the Model:
Data Preprocessing.
Train-Test Split
Model Selection

2. Evaluating the Model:
Training the Model
Hyperparameter Tuning
Performance Metrics

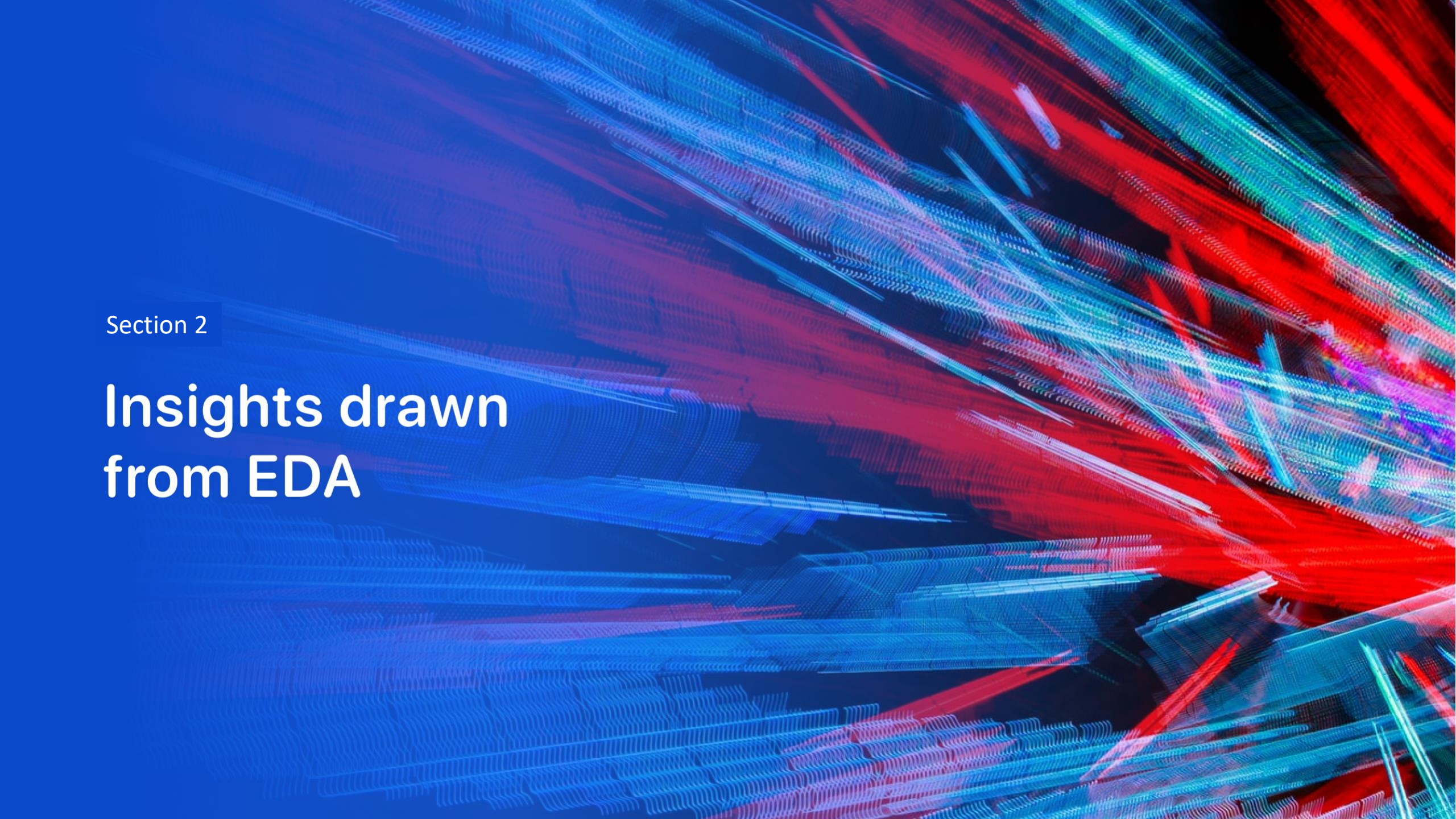
3. Improving the Model:
Feature Engineering
Model Comparison

4. Finding the Best Performing Model:
Final Evaluation
Confusion Matrix

- GitHub URL: https://github.com/cysorianoc/IBM_Data_Science/blob/main/CAPSTONE/CAPSTONE_P8_SpaceX_Machine_Learning_Prediction.ipynb

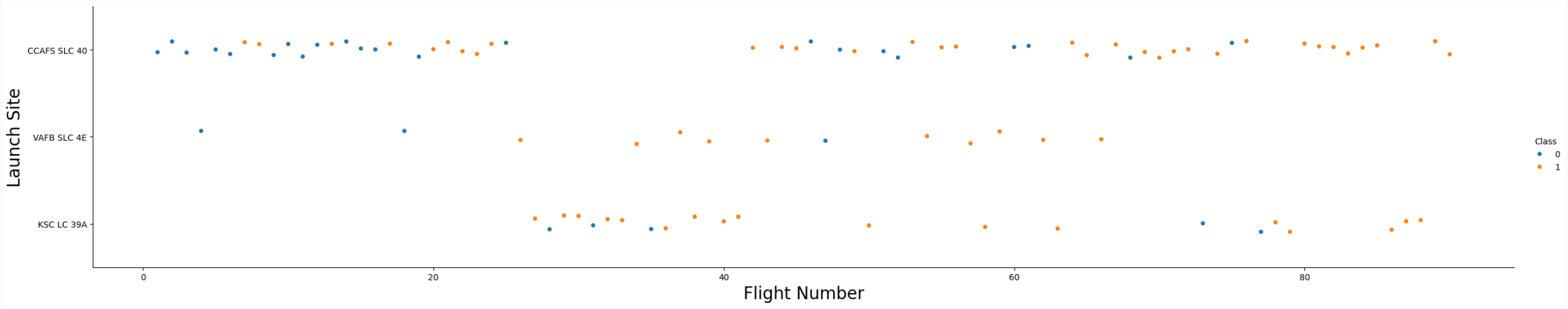
Results

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


Section 2
Insights drawn from EDA

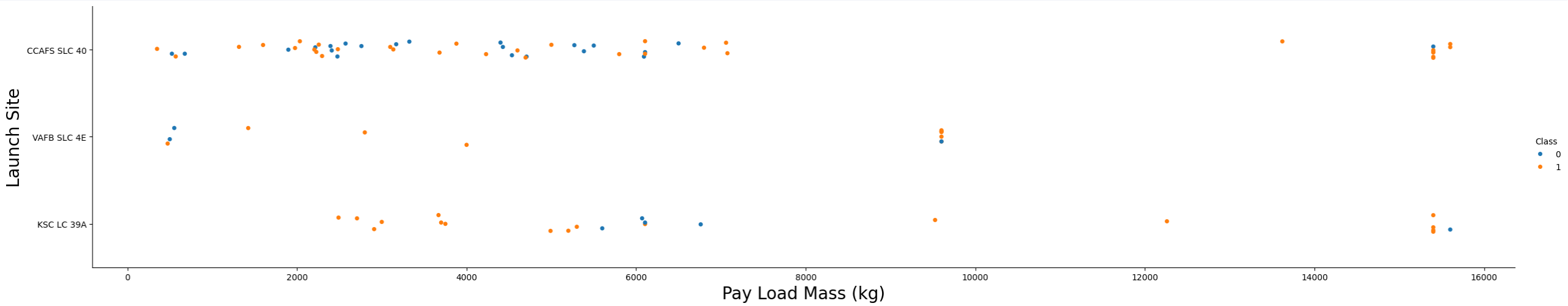
Flight Number vs. Launch Site

- As the number of flights increase, the success rate has a trend to 1.0 which means success flights.



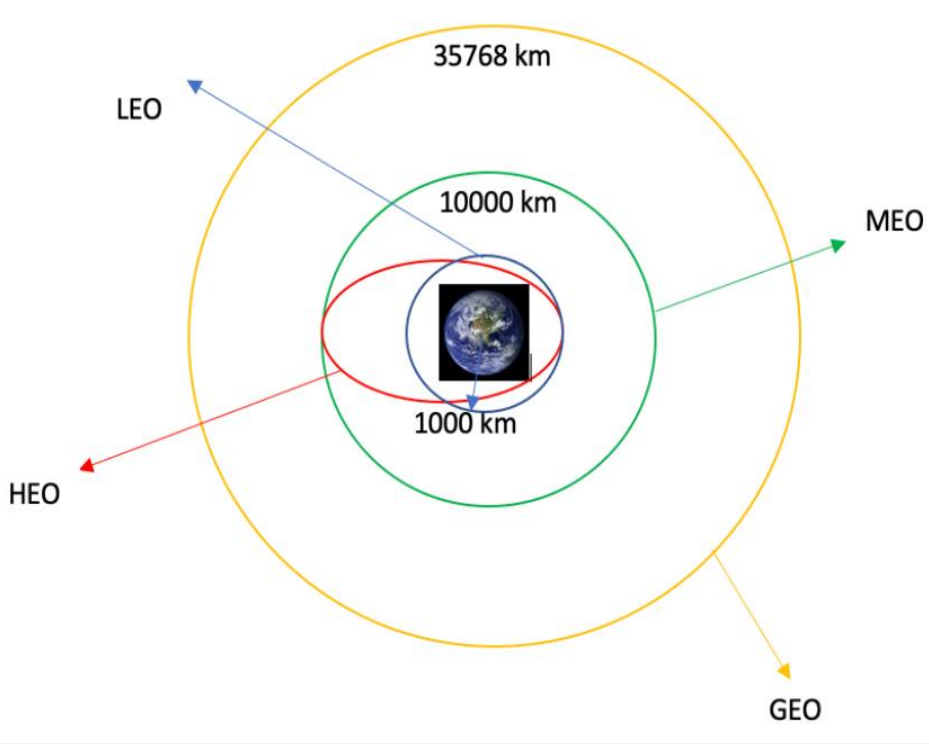
Payload vs. Launch Site

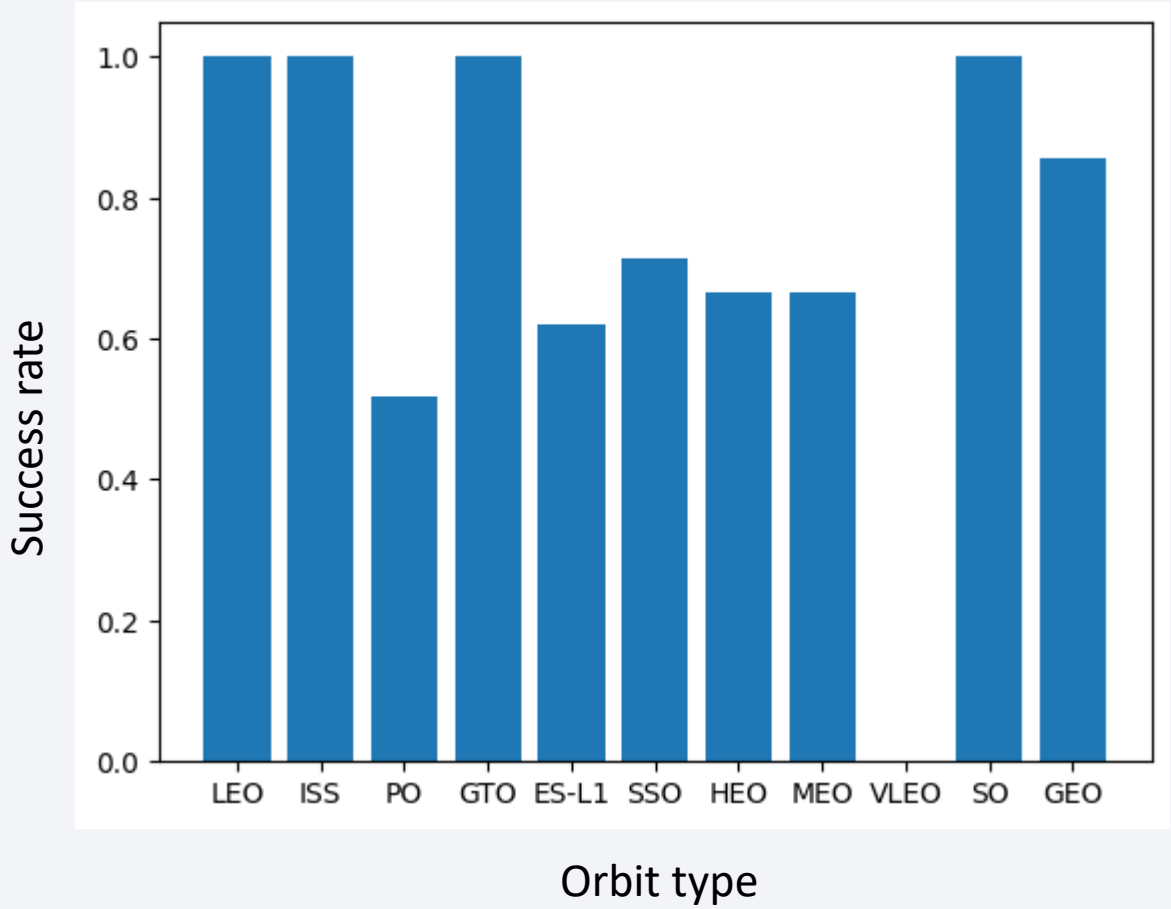
- If you observe Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).



Success Rate vs. Orbit Type

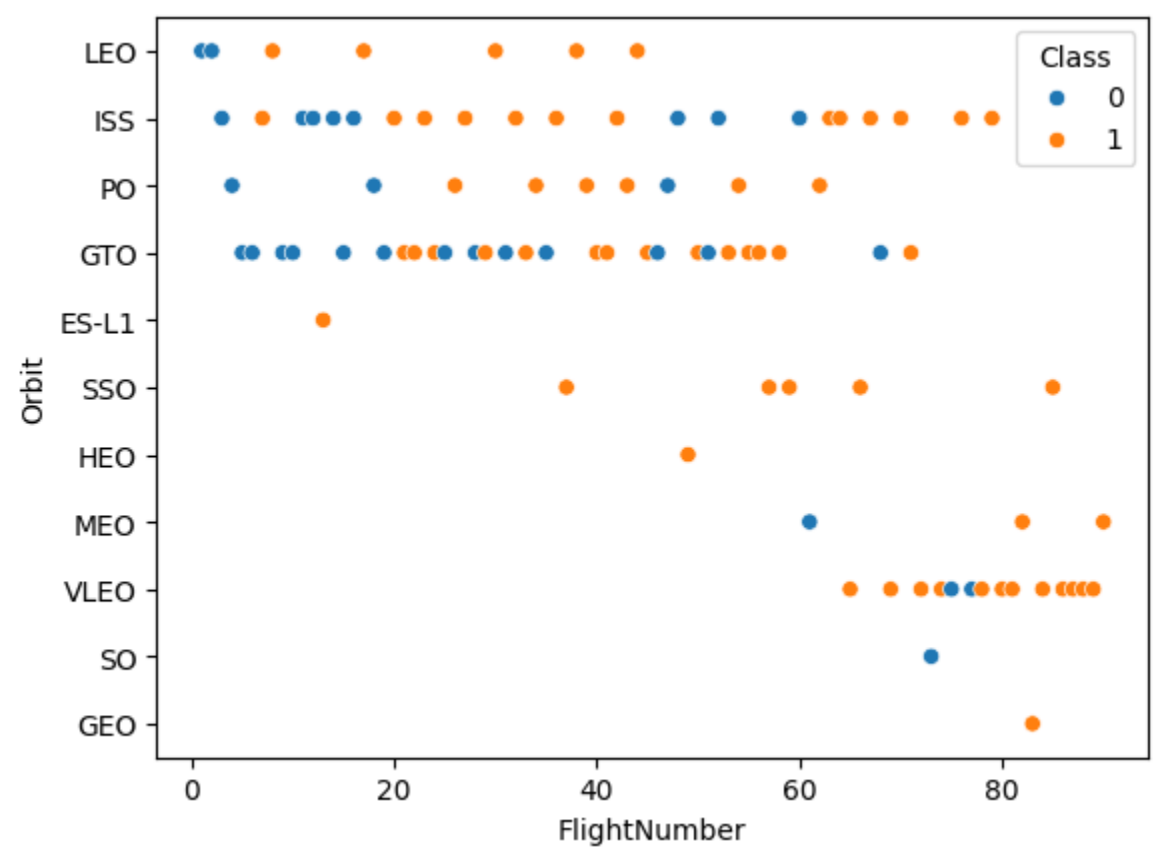
- LEO, ISS, GTO and SO have the highest success rates





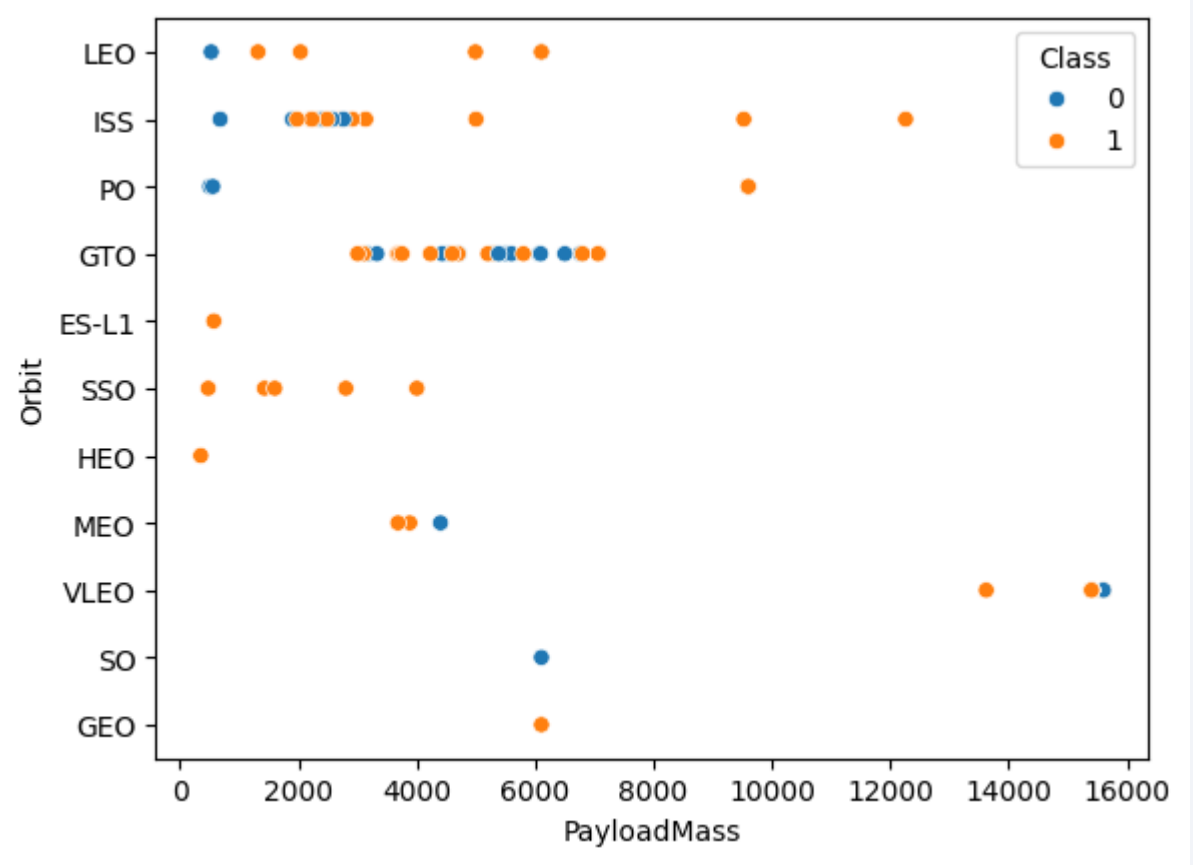
Flight Number vs. Orbit Type

- In the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.



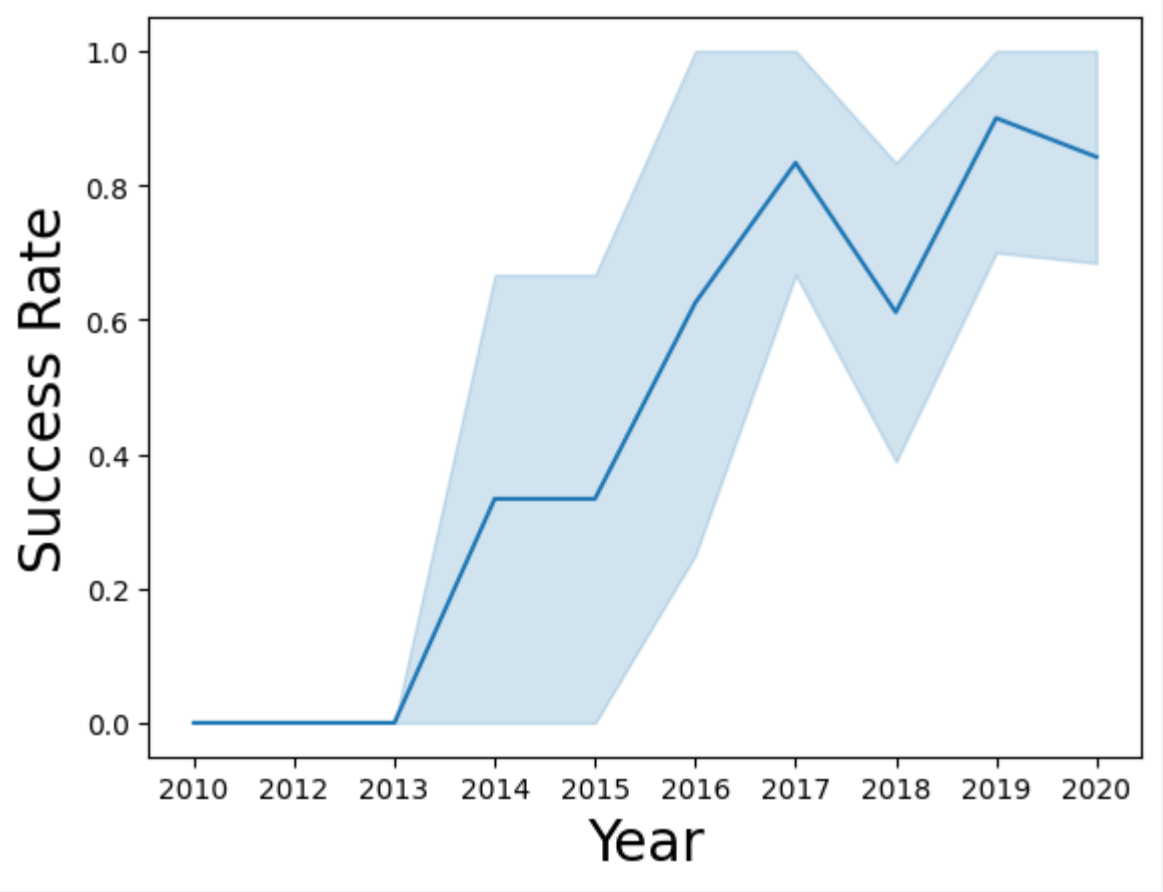
Payload vs. Orbit Type

- With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.
- However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission).



Launch Success Yearly Trend

- You can observe that the success rate since 2013 kept increasing till 2017 (stable in 2014) and after 2015 it started increasing.



All Launch Site Names

- From the SQL Notebook

https://github.com/cysorianoc/IBM_Data_Science/blob/main/CAPSTONE/CAPSTONE_P4_eda_sql_coursera_sqllite.ipynb

We can perform the following query

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

```
[ ]  sql SELECT DISTINCT Launch_Site FROM SPACEXTABLE;
```

```
 * 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'

- From the SQL Notebook

https://github.com/cysorianoc/IBM_Data_Science/blob/main/CAPSTONE/CAPSTONE_P4_eda_sql_coursera_sqllite.ipynb

We can perform the following query

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

```
[ ]  sql SELECT * FROM SPACEXTABLE 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

- From the SQL Notebook

https://github.com/cysorianoc/IBM_Data_Science/blob/main/CAPSTONE/CAPSTONE_P4_eda_sql_coursera_sqllite.ipynb

We can perform the following query

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

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

```
 * sqlite:///my_data1.db
Done.
 SUM(PAYLOAD_MASS__KG_)
 45596
```

Average Payload Mass by F9 v1.1

- From the SQL Notebook

https://github.com/cysorianoc/IBM_Data_Science/blob/main/CAPSTONE/CAPSTONE_P4_eda_sql_coursera_sqllite.ipynb

We can perform the following query

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

 * sqlite:///my_data1.db
Done.
AVG(PAYLOAD_MASS__KG_)
2928.4
```

First Successful Ground Landing Date

- From the SQL Notebook

https://github.com/cysorianoc/IBM_Data_Science/blob/main/CAPSTONE/CAPSTONE_P4_eda_sql_coursera_sqllite.ipynb

We can perform the following query

```
[ ]  sql SELECT MIN(Date) FROM SPACEXTABLE 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

- From the SQL Notebook

https://github.com/cysorianoc/IBM_Data_Science/blob/main/CAPSTONE/CAPSTONE_P4_eda_sql_coursera_sqllite.ipynb

We can perform the following query

```
[ ] sql SELECT 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.
```

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

Total Number of Successful and Failure Mission Outcomes

- From the SQL Notebook

https://github.com/cysorianoc/IBM_Data_Science/blob/main/CAPSTONE/CAPSTONE_P4_eda_sql_coursera_sqllite.ipynb

We can perform the following query

```
[ ]  sql SELECT Mission_Outcome, COUNT(*) FROM SPACEXTABLE GROUP BY Mission_Outcome;
```

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

Mission_Outcome	COUNT(*)
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Boosters Carried Maximum Payload

- We determined the booster that have carried the maximum payload using a subquery in the **WHERE** clause and the **MAX()** function

```
sql SELECT Booster_Version, PAYLOAD_MASS__KG_  FROM SPACEXTABLE  WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE);
```

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

Booster_Version	PAYLOAD_MASS__KG_
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

- We utilized a combination of the WHERE clause, LIKE, AND, and BETWEEN conditions to filter failed landing outcomes on drone ships, along with their booster versions and launch site names for the year 2015.

```
[ ] %sql SELECT substr(Date, 6, 2) AS Month, Landing_Outcome, Booster_Version, Launch_Site FROM SPACEXTABLE WHERE substr(Date, 0, 5) = '2015' AND Landing_Outcome = 'Failure (drone ship)';
 * sqlite:///my_data1.db
Done.
```

Month	Landing_Outcome	Booster_Version	Launch_Site
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

- This SQL query retrieves the number of occurrences of each **Landing_Outcome** from the **SPACEXTABLE** for launches between **June 4, 2010, and March 20, 2017**. It groups the results by **Landing_Outcome** and counts how many times each outcome appears. Finally, it orders the results in **descending order** based on the count to show the most frequent outcomes first.

```
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;
```

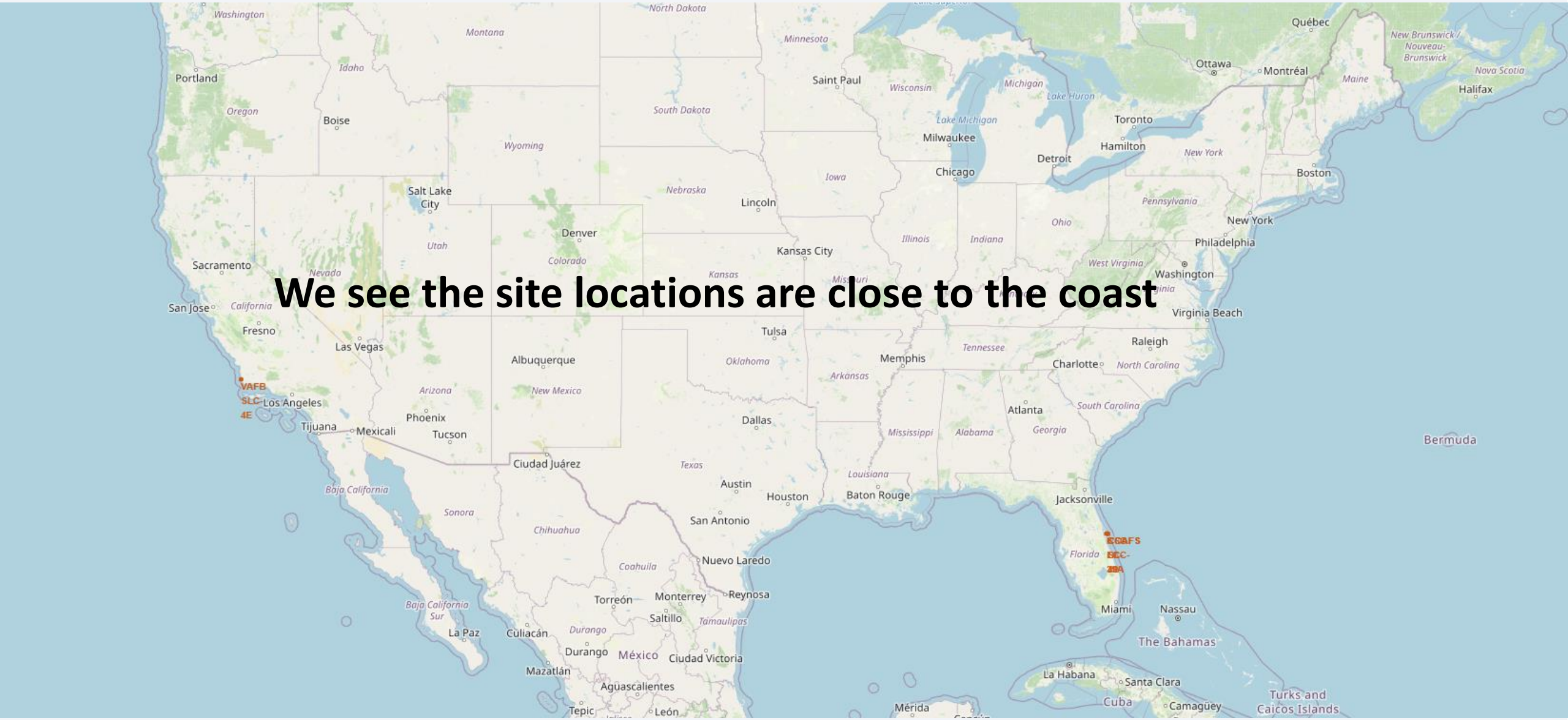
```
 * sqlite:///my_data1.db
Done.
```

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

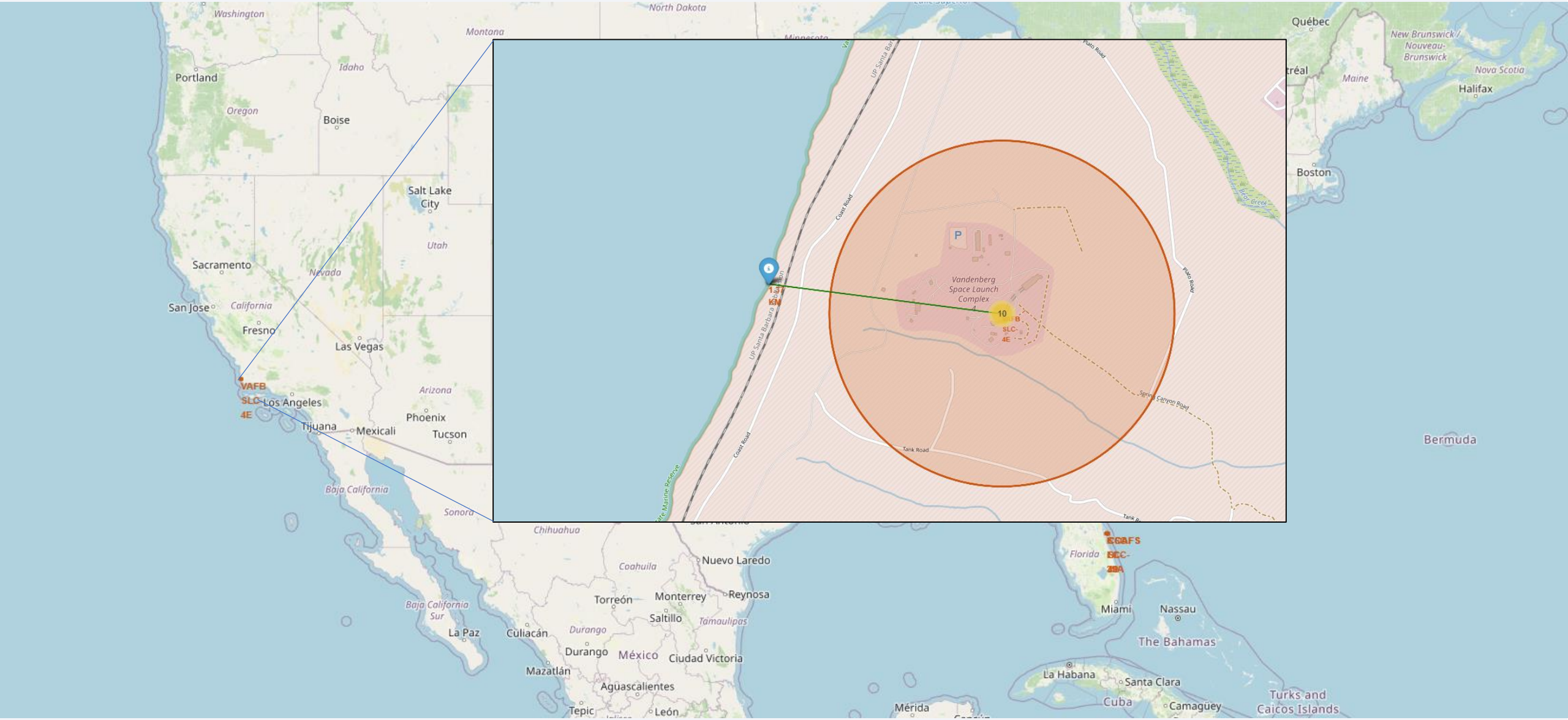
Section 3

Launch Sites Proximities Analysis

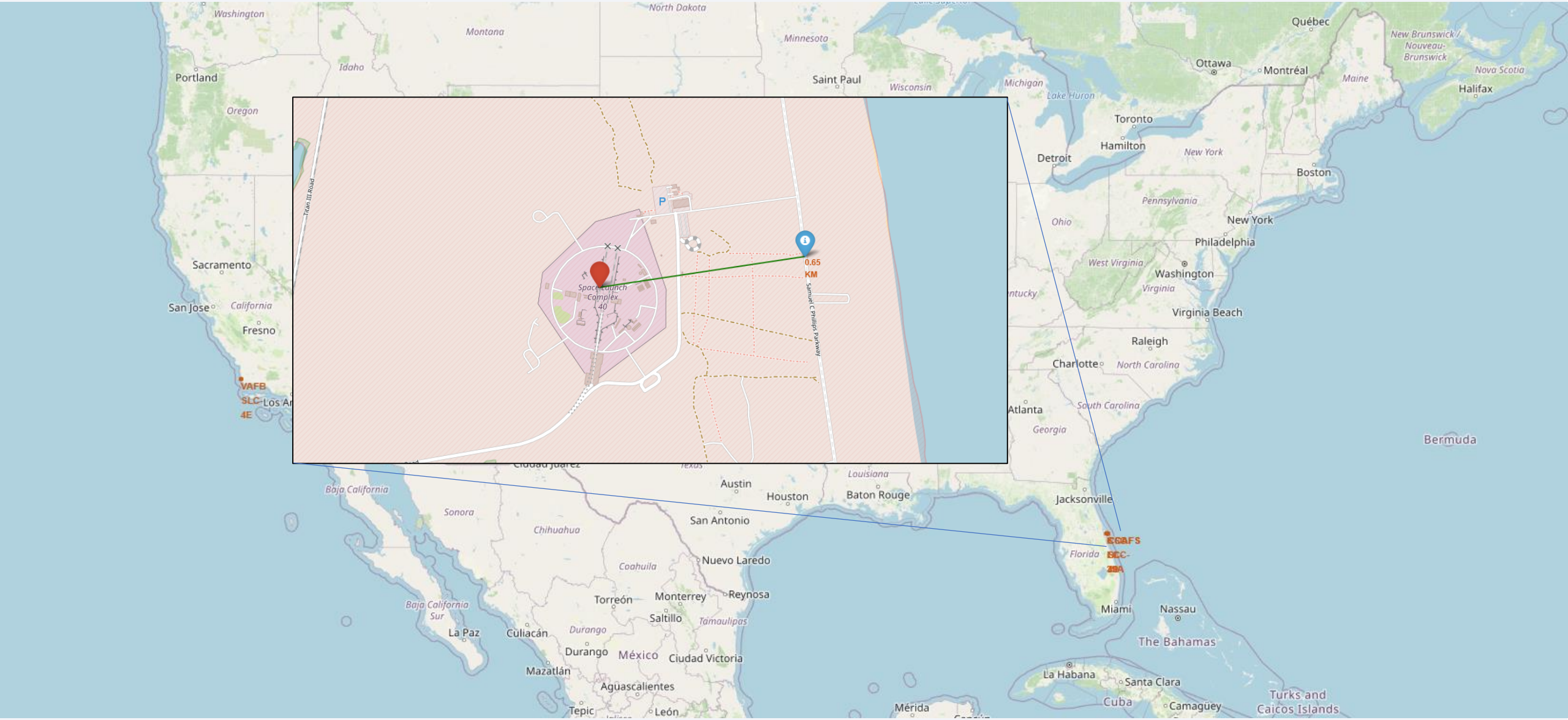
Map of Launch Sites locations



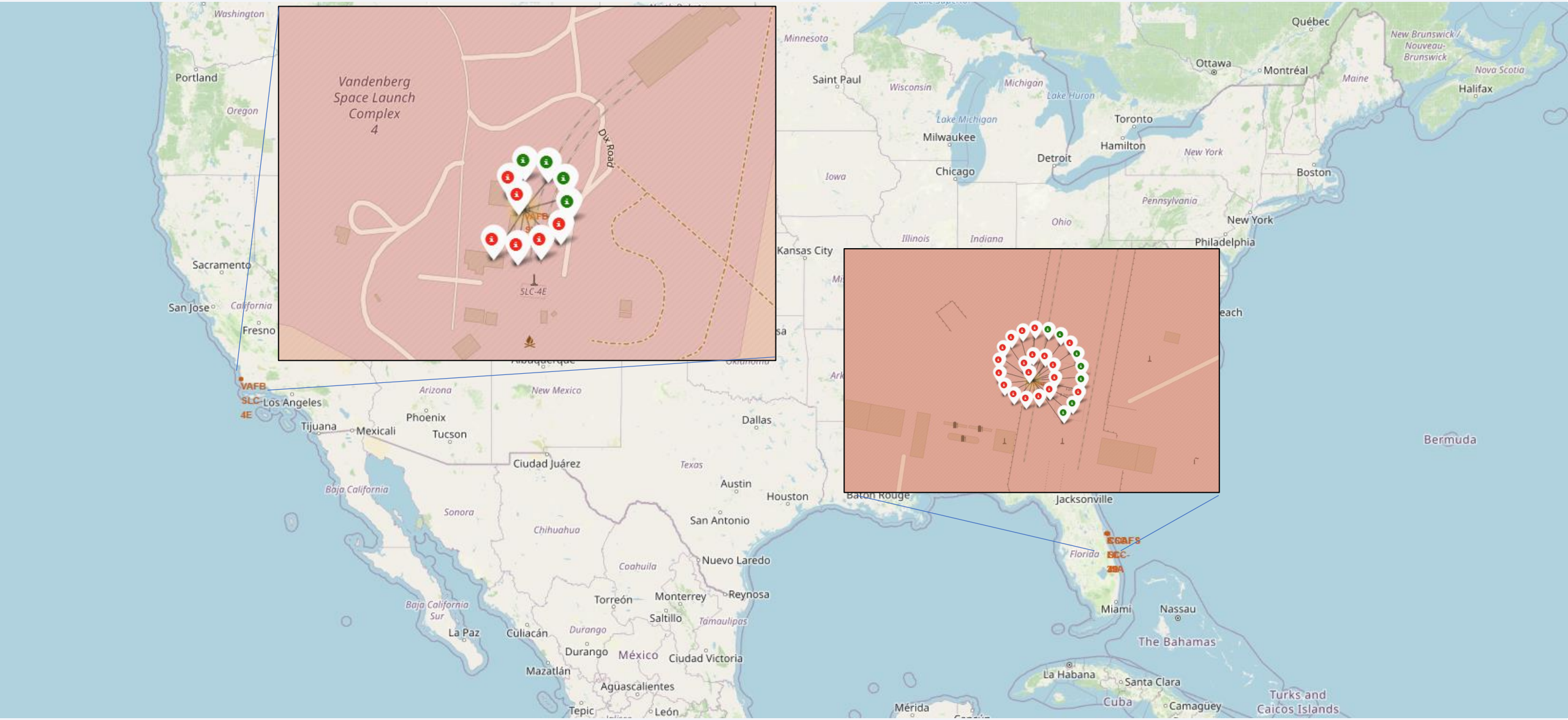
An example of distance to the coast line

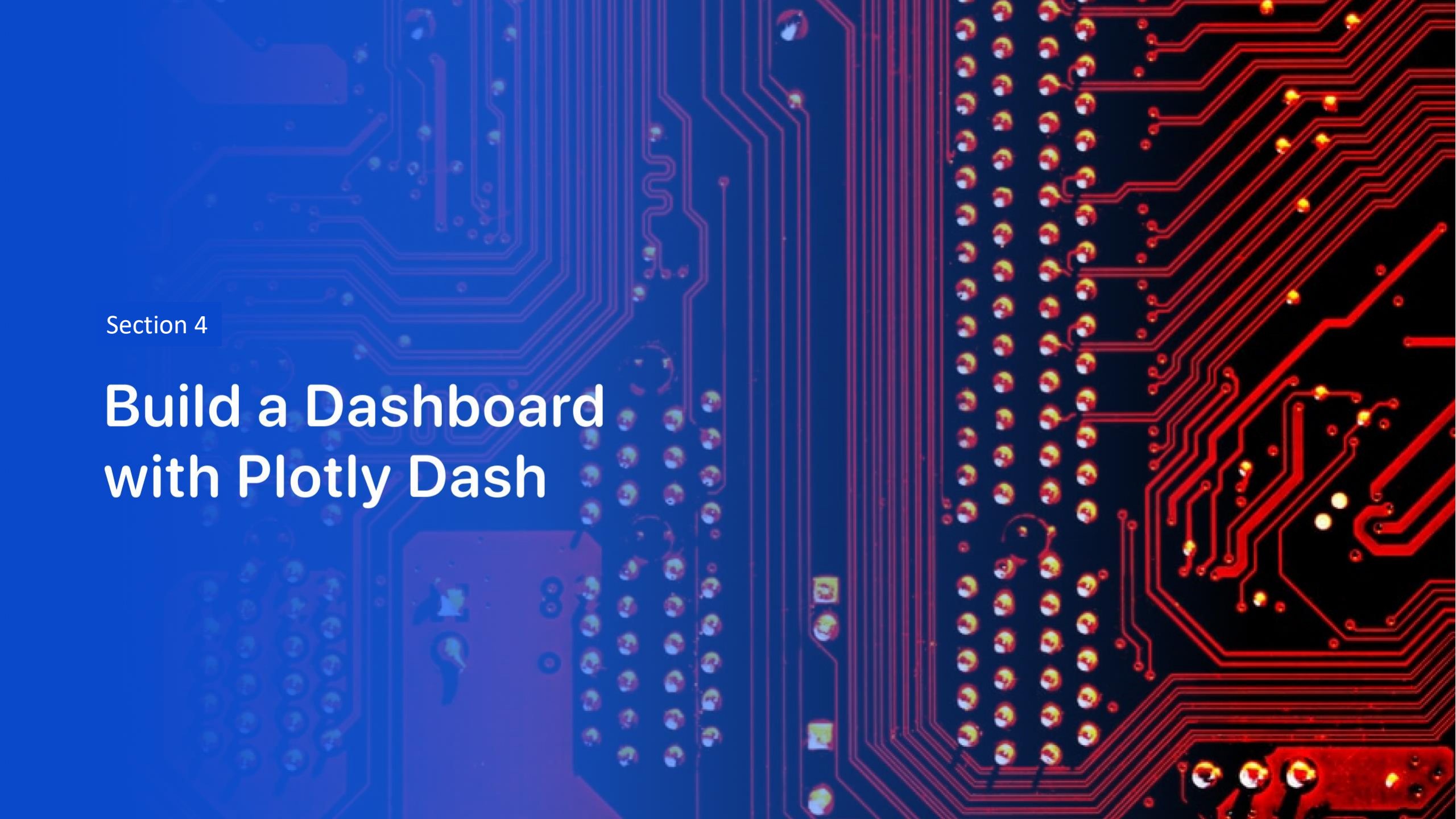


An example of distance to infrastructure

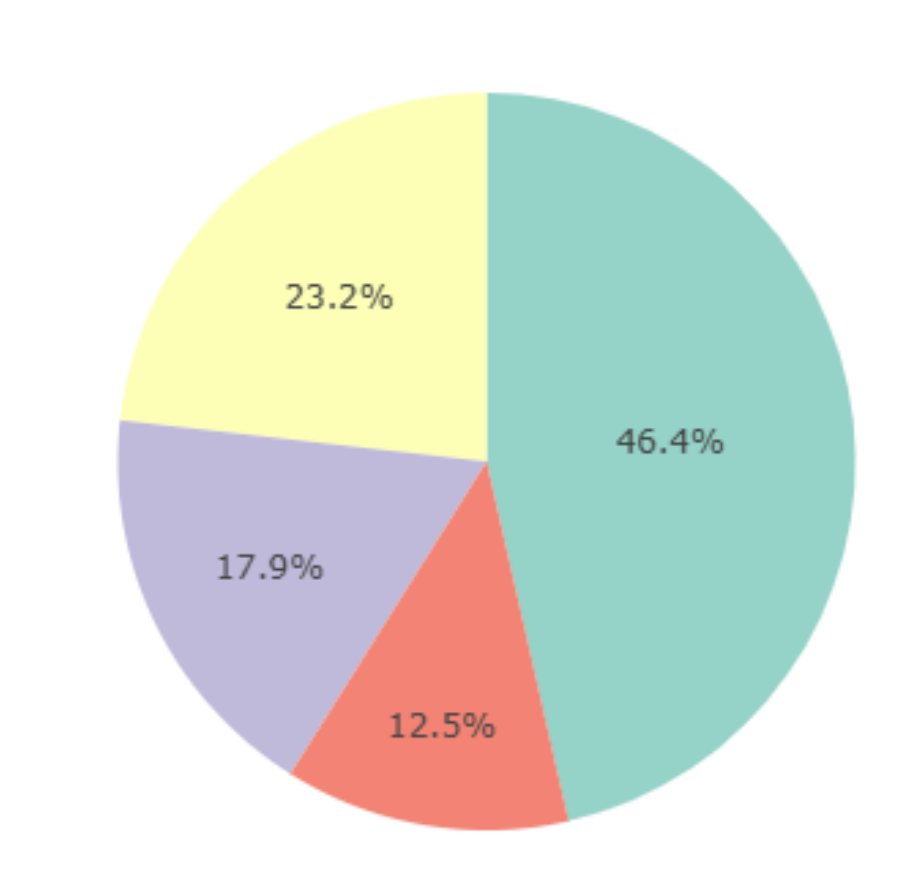


We can see the outcome of a launch with color labels




Section 4
Build a Dashboard with Plotly Dash

Summary pie chart

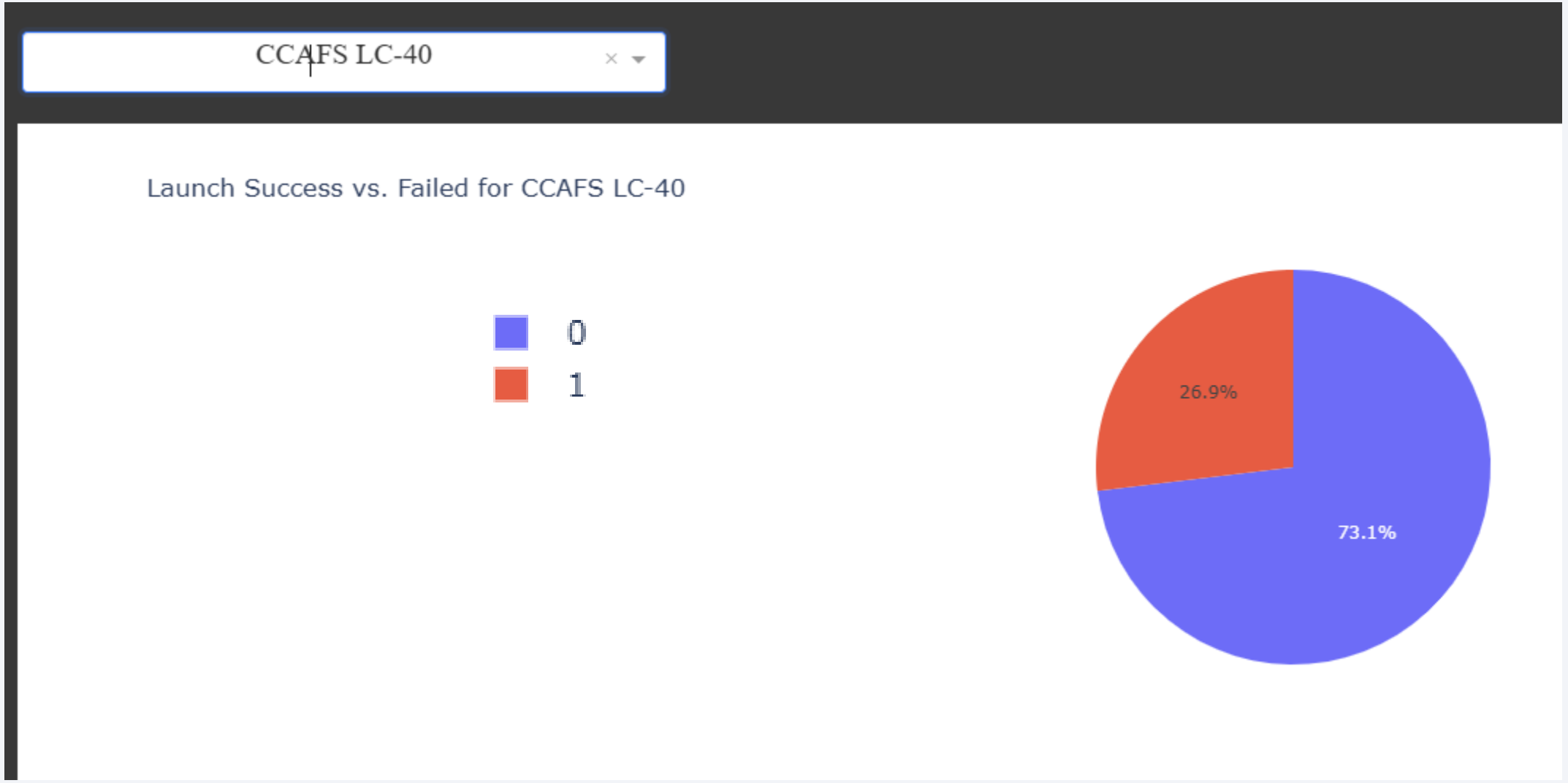


CCAFS LC-40
KSC LC-39A
VAFB SLC-4E
CCAFS SLC-40

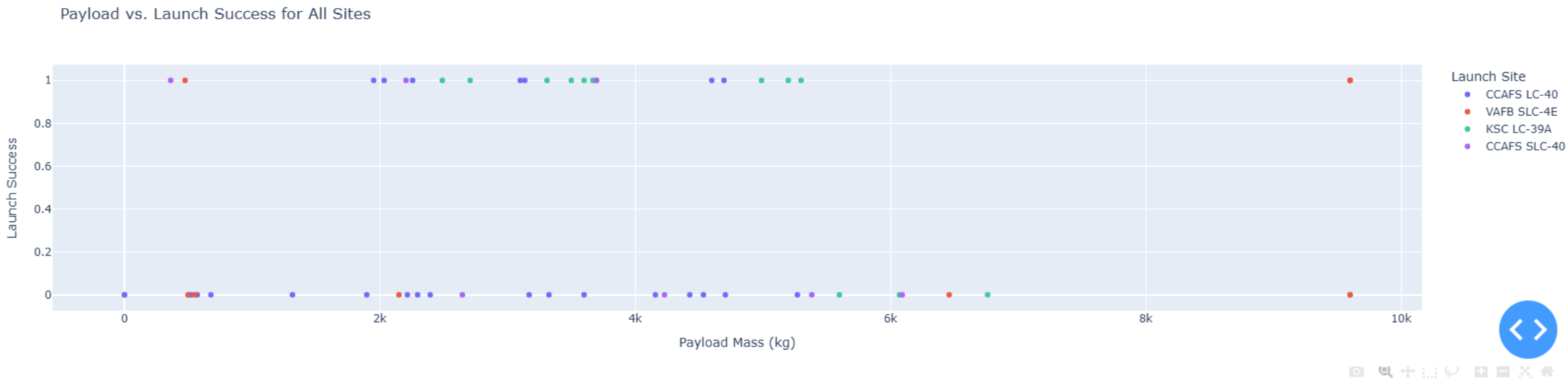
Dashboards can be done in Jupyter and Colab

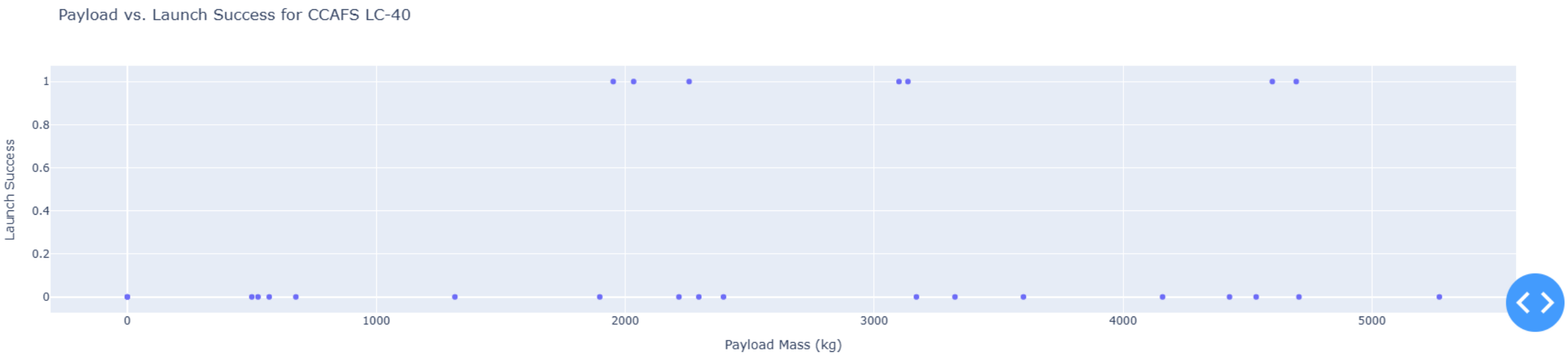
```
[1] pip install jupyter-dash -q
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.8/7.8 MB 23.6 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 101.7/101.7 kB 2.0 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.6/1.6 MB 37.1 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 228.0/228.0 kB 9.0 MB/s eta 0:00:00

[2] !pip3 install wget
    Collecting wget
      Downloading wget-3.2.zip (10 kB)
      Preparing metadata (setup.py) ... done
    Building wheels for collected packages: wget
      Building wheel for wget (setup.py) ... done
      Created wheel for wget: filename=wget-3.2-py3-none-any.whl size=9656 sha256=c208efc6
      Stored in directory: /root/.cache/pip/wheels/40/b3/0f/a40dbd1c6861731779f62cc4babcb2
    Successfully built wget
    Installing collected packages: wget
    Successfully installed wget-3.2

[3] import wget

[4] data = wget.download( "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cl
```

Example of pie chart for a launch site



Scatter chart for all sites and for a site





Section 5

Predictive Analysis (Classification)

Classification Accuracy

- The decision tree classifier is the model with the highest classification accuracy

```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)
```

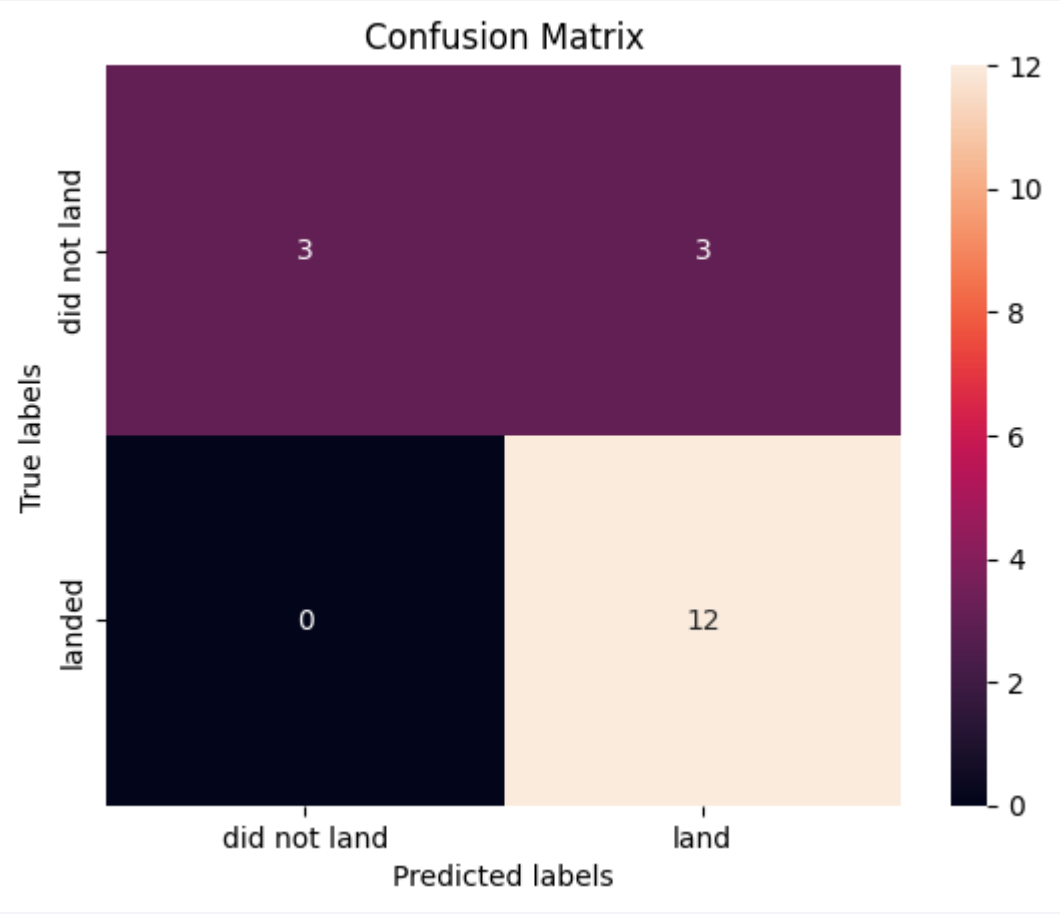
```
Best model is DecisionTree with a score of 0.8732142857142856
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```

Confusion Matrix

- Examining the confusion matrix, we see that the problem is false positives.

Confusion Matrix

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)



Conclusions

- A higher number of flights at a launch site corresponds to a higher success rate.
- Launch success rates began rising from 2013 to 2020.
- Orbits ES-L1, GEO, HEO, SSO, and VLEO had the highest success rates.
- KSC LC-39A recorded the most successful launches among all sites.
- The Decision Tree classifier is the most effective machine learning algorithm for this task.

Appendix

- I did the dashboards in Google Colab without using the online environment, so is worth to have a look to this alternative to use dash.

https://github.com/cysorianoc/IBM_Data_Science/blob/main/CAPSTONE/Capstone_P7_dash.ipynb

Thank you!