Winning Space Race with Data Science

Stella Kim
05.06.2023

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection through API
  - Data Collection with Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with SQL, Data Visualization
  - Interactive Visual Analytics with Folium
  - Machine Learning Prediction
- Summary of all results
  - The best Hyperparameters for Logistic Regression, SVM, Decision Tree, and KNN classifiers.
  - The method that performs best with test data

Introduction

- Project background and context
  - Space Y advertise Falcon 9 rocket launches on its website with a cost of 62 million dallars; other providers cost upward of 165 million dollars each, much of the savings is because Space Y can reuse the first stage. Thus, if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against Space Y for a rocket launch. This goal of the project is to create a machine learning pipeline to predict if the first stage will land successfully.
- Problems you want to find answers
  - What factors determine if the rocket will land successfully?
  - The interaction amongst various features that determine the success rate of a successful landing.
  - What operating conditions needs to be in place to ensure a successful landing program.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using Space Y API and web scraping.
- Perform data wrangling
  - One-hot encoding was applied to categorical features
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

- Describe how data sets were collected.
  - Data collection was done using get request to the Space Y API
  - Next, we decoded the response content as a Json using .json() function call and turn it into a pandas dataframe using .json_normalize().
  - Cleaned the data, checked for missing values and fill in missing values where necessary.
  - We performed web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.
  - The objective was to extract the launch records as HTML table, parse the table and convert it to a pandas data frame for future analysis.
- You need to present your data collection process use key phrases and flowcharts

Data Collection – SpaceX API

- Used the get request to the Space Y API to collect data, clean the requested data and did some basic data wrangling and formatting

- https://github.com/RHstudio/IBM_Applied-Data-Science-Capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

In [6]:
```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

In [7]:
```
response = requests.get(spacex_url)
```

[11]:
```
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

Using the dataframe `data` print the first 5 rows

[12]:
```
# Get the head of the dataframe
data.head(5)
```

In [31]:
```
# Calculate the mean value of PayloadMass column
Mean_PayloadMass = data_falcon9.PayloadMass.mean()
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, Mean_PayloadMass)

data_falcon9.isnull().sum()
```

Data Collection - Scraping

- Applied web scraping to web scrap Falcon 9 launch records with BeautifulSoup
- Parsed the table and converted it into a pandas dataframe.
- https://github.com/RHstudio/IBM_Applied-Data-Science-Capstone/blob/main/jupyter-labs-webscraping.ipynb

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

Next, request the HTML page from the above URL and get a `response` object

TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

```
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url).text
```

Create a `BeautifulSoup` object from the HTML `response`

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response, 'html.parser')
```

Print the page title to verify if the `BeautifulSoup` object was created properly

```
# Use soup.title attribute
print(soup.title)
```

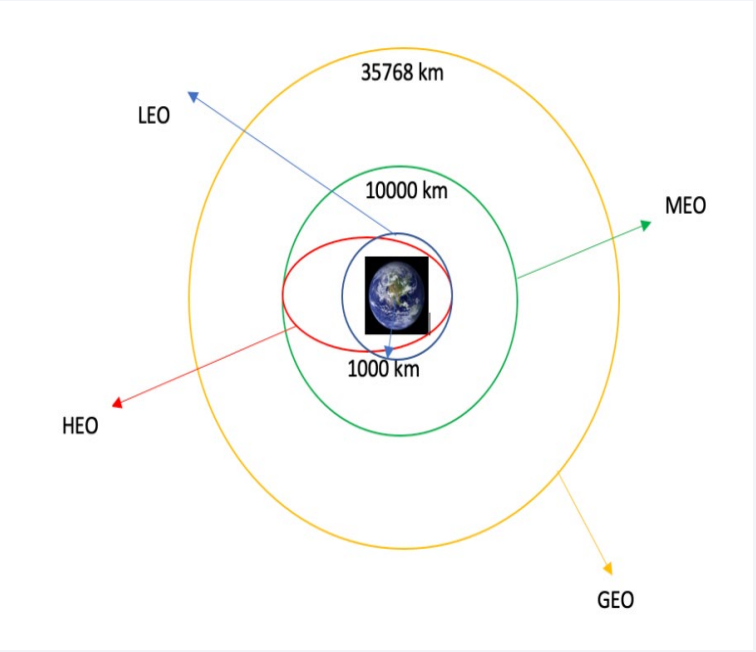
```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

In [17]:

```
headings = []
for key,values in dict(launch_dict).items():
    if key not in headings:
        headings.append(key)
    if values is None:
        del launch_dict[key]

def pad_dict_list(dict_list, padel):
    lmax = 0
    for lname in dict_list.keys():
        lmax = max(lmax, len(dict_list[lname]))
    for lname in dict_list.keys():
        ll = len(dict_list[lname])
        if  ll < lmax:
            dict_list[lname] += [padel] * (lmax - ll)
    return dict_list

pad_dict_list(launch_dict,0)

df = pd.DataFrame.from_dict(launch_dict)
df.head()
```

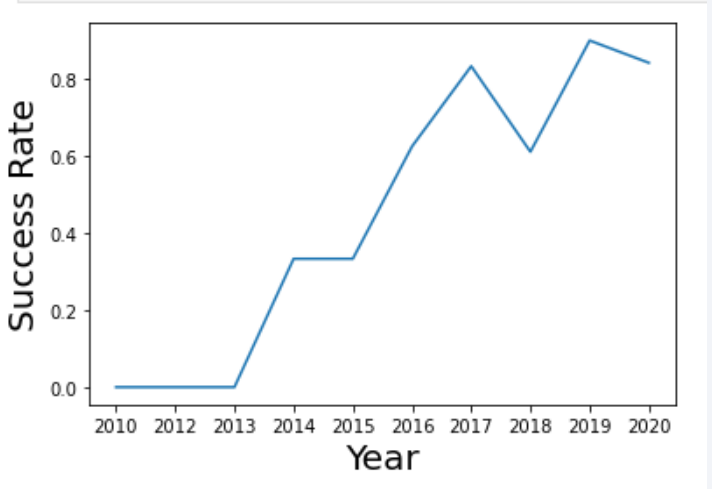
Data Wrangling

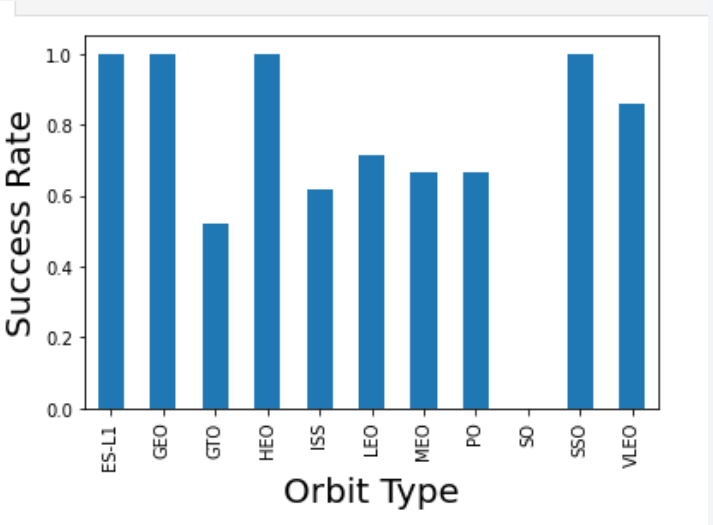
- Performed exploratory data analysis and determined the training labels.
- Calculated the number of launches at each site, and the number and occurrence of each orbits.
- Created landing outcome label from outcome column and exported the results to cvs.
- https://github.com/RHstudio/IBM_Applied-Data-Science-Capstone/blob/main/labs-jupyter-spacex-data_wrangling_jupyterlite.jupyterlite.ipynb



EDA with Data Visualization

- Explored the data by visualizing the relationship between flight number and launch site, payload and launch site, success rate of each orbit type, flight number and orbit type, the launch success yearly trend.
- https://github.com/RHstudio/IBM_Applied-Data-Science-Capstone/blob/main/5.%20eda-dataviz.ipynb





EDA with SQL

- Applied EDA with SQL to get insight from the data. We wrote queries to find out for instance:
  - The names of unique launch sites in the space mission
  - The total payload mass carried by boosters launched by NASA (CRS)
  - The average payload mass carried by booster version F9
  - The total number of successful and failure mission outcomes
  - The failed landing outcomes in drone ship, their booster version and launch site names.
- https://github.com/RHstudio/IBM_Applied-Data-Science-Capstone/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- Marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- Assigned the feature launch outcomes (failure or success) to class 0 (failure) and 1 (success)
- Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate.
- Calculated the distances between a launch site to its proximities. Answered some question for instance”
  - Are launch sites near railways, highways and coastlines.
  - Do launch sites keep certain distance away from cities.
- https://github.com/RHstudio/IBM_Applied-Data-Science-Capstone/blob/main/Week3_Interactive%20Visual%20Analytics%20with%20Folium%20lab(1).ipynb

Build a Dashboard with Plotly Dash

- Built an interactive dashboard with Plotly dash.
- Plotted pie charts showing the total launches by a certain sites.
- Plotted scatter graph showing the relationship with outcome and payload mass (Kg) for the different booster version
- https://github.com/RHstudio/IBM_Applied-Data-Science-Capstone/blob/main/Wee3_Build%20an%20Interactive.ipynb%20Dashboard%20with%20Ploty%20Dash

Predictive Analysis (Classification)

- Loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- Built different machine learning models and tune different hyperparameters using GridSearchCV.
- Used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- Found the best performing classification model
- https://github.com/RHstudio/IBM_Applied-Data-Science-Capstone/blob/main/Week4_Complete%20the%20Machine%20Learning%20Prediction%20lab.ipynb

Results

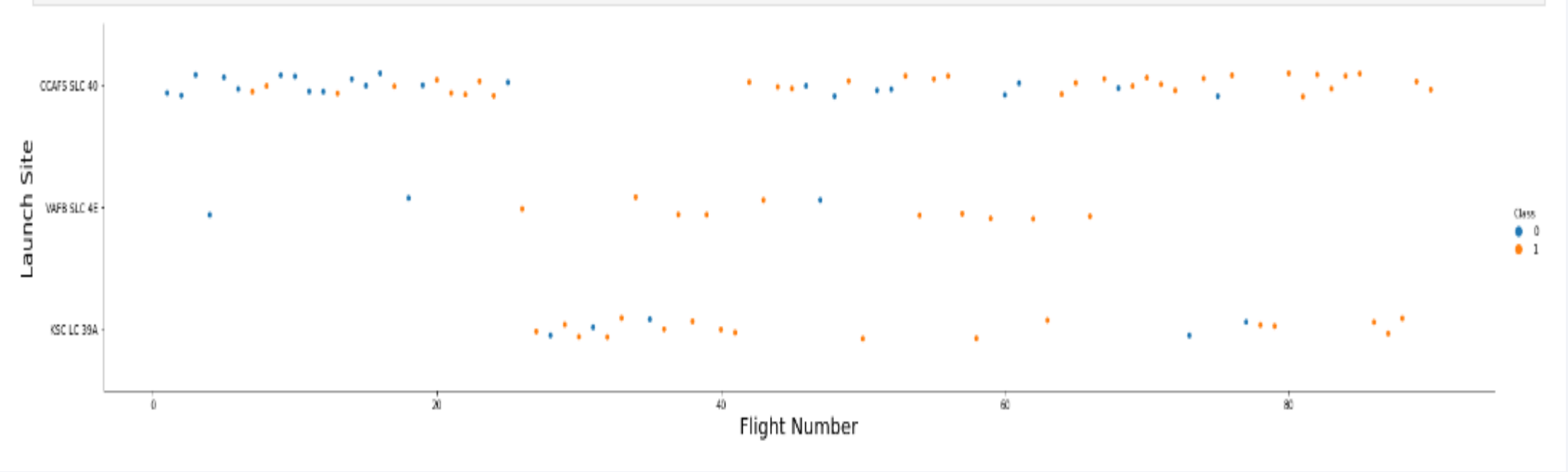
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

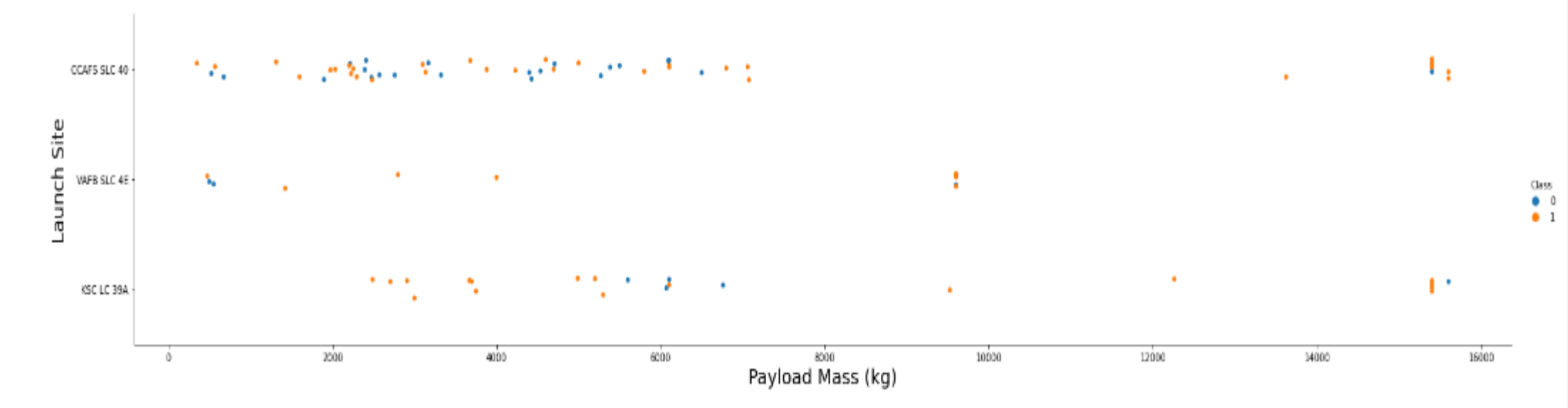
Flight Number vs. Launch Site

- Show a scatter plot of Flight Number vs. Launch Site
- The larger the flight amount at a launch site, the greater the success rate at a launch site



Payload vs. Launch Site

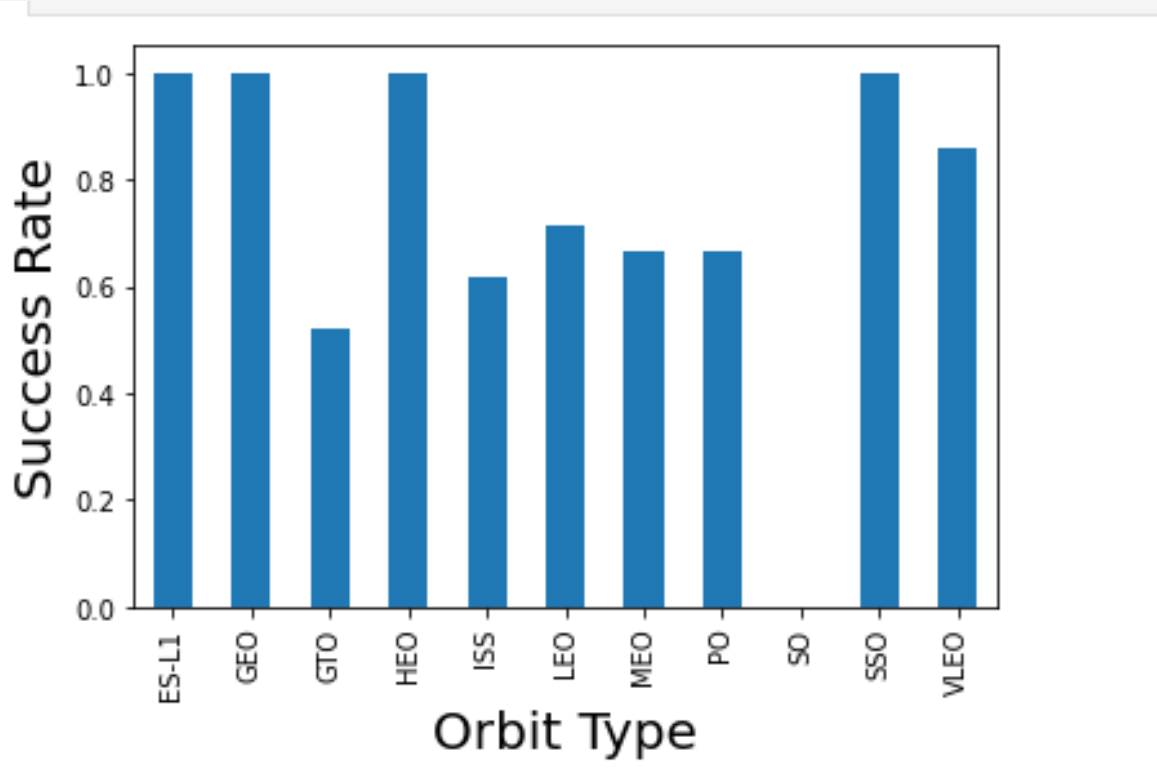
- Show a scatter plot of Payload vs. Launch Site
- The greater the payload mass for launch site CCAFS SLC 40 the higher the success rate for the rocket.



Success Rate vs. Orbit Type

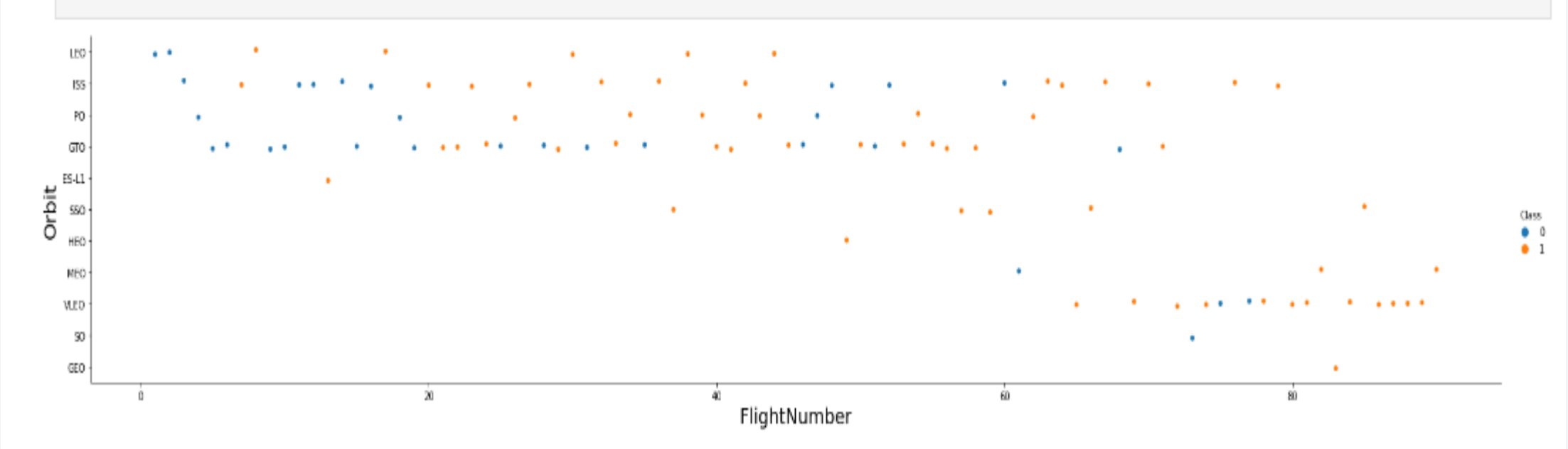
- Show a bar chart for the success rate of each orbit type

- ES-L1, GEO, HEO, SSO, VLEO had the most success rate.



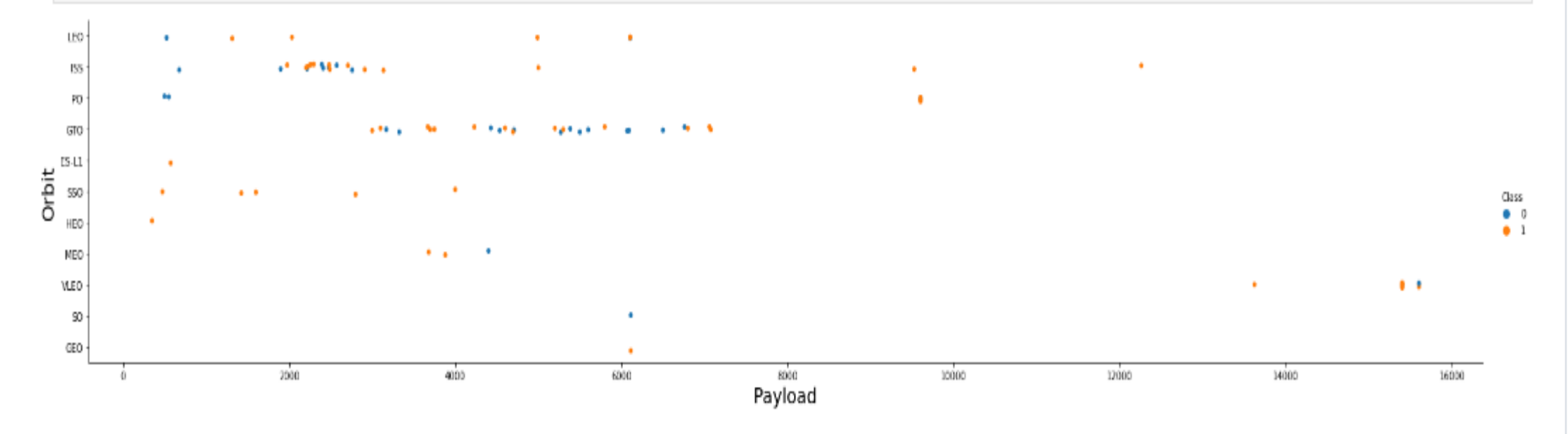
Flight Number vs. Orbit Type

- Show a scatter point of Flight number vs. Orbit type
- Observe that in the LEO orbit, success is related to the number of flights whereas in the GTO orbit, there is no relationship between flight number and the orbit.



Payload vs. Orbit Type

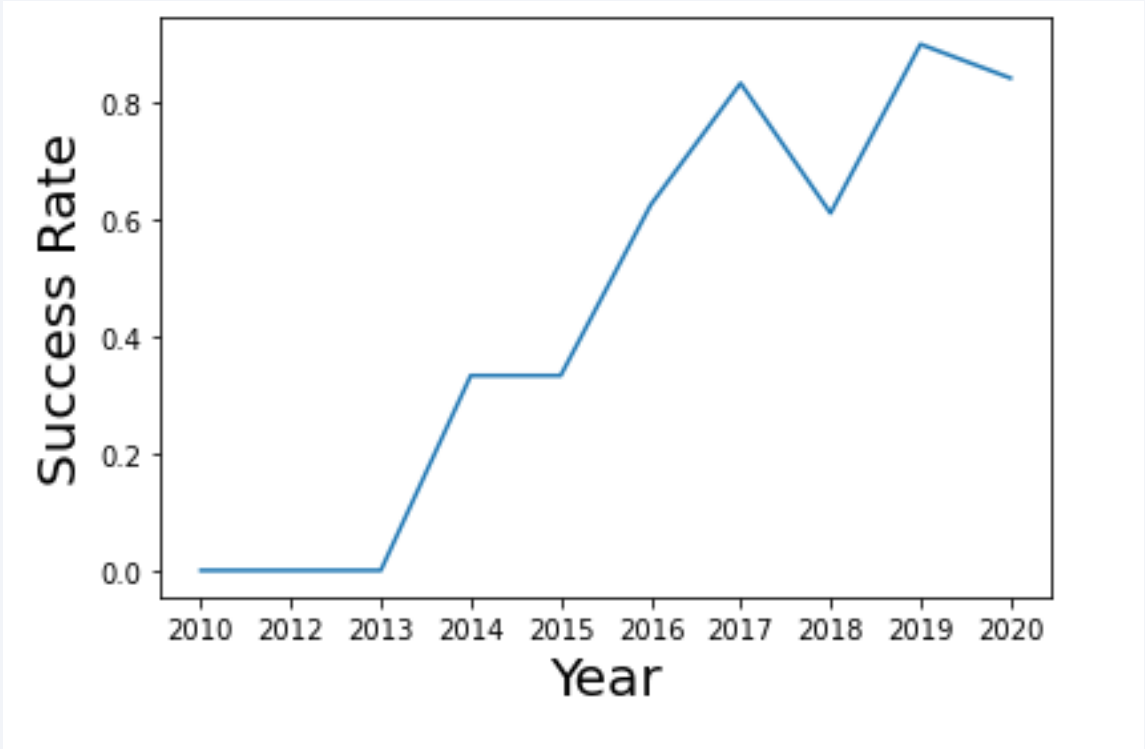
- Show a scatter point of payload vs. orbit type
- The successful landing are more for PO, LEO, and ISS orbits.



Launch Success Yearly Trend

- Show a line chart of yearly average success rate

- Can see the success rate since 2023 kept on increasing till 2020.



All Launch Site Names

- Find the names of the unique launch sites
- Used DISTINCT to show only unique launch site from the Space Y data.

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

```
%sql SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL;
```

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

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

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`
- We used the query above to display 5 records where launch sites begin with CCA.

Display 5 records where launch sites begin with the string '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
- From below results, the result is 45596.

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

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE Customer = 'NASA (CRS)';
```

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

SUM(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.0

- Calculate the average payload mass carried by booster version F9 v1.0

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

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE Booster_Version LIKE 'F9 v1.0%';
```

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

AVG(PAYLOAD_MASS__KG_)
340.4

First Successful Ground Landing Date

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

```
%sql SELECT MIN(Date) FROM SPACEXTBL WHERE Landing__Outcome = 'Success (ground pad)';
```

```
 * sqlite:///my_data1.db
(sqlite3.OperationalError) no such column: Landing__Outcome
[SQL: SELECT MIN(Date) FROM SPACEXTBL WHERE Landing__Outcome = 'Success (ground pad)';]
(Background on this error at: http://sqlalche.me/e/e3q8)
```

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

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE LANDING__OUTCOME = 'Success (drone ship)' AND 4000 < PAYLOAD_MASS__KG_ < 6000
```

```
* sqlite:///my_data1.db
sqlite3.OperationalError) no such column: LANDING__OUTCOME
SQL: SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE LANDING__OUTCOME = 'Success (drone ship)' AND 4000 < PAYLOAD_MASS__KG_ < 6000]
Background on this error at: http://sqlalche.me/e/e3q8)
```

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes

```
[13]: %sql select count(MISSION_OUTCOME) from SPACEXTBL where MISSION_OUTCOME = 'Success' or MISSION_OUTCOME = 'Failure (in flight)'
```

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

t[13]:

count(MISSION_OUTCOME)
99

Boosters Carried Maximum Payload

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

```
[14]: %sql select BOOSTER_VERSION from SPACEXTBL where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTBL)
```

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

[14]:

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

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, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.

```
]:  %sql SELECT EXTRACT(MONTH, select min(DATE) from SPACEXTBL where Landing__Outcome = 'Success (ground pad)')
```

```
 * sqlite:///my_data1.db
(sqlite3.OperationalError) near "select": syntax error
[SQL: SELECT EXTRACT(MONTH, select min(DATE) from SPACEXTBL where Landing__Outcome = 'Success (ground pad)')]
(Background on this error at: http://sqlalche.me/e/e3q8)
```

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

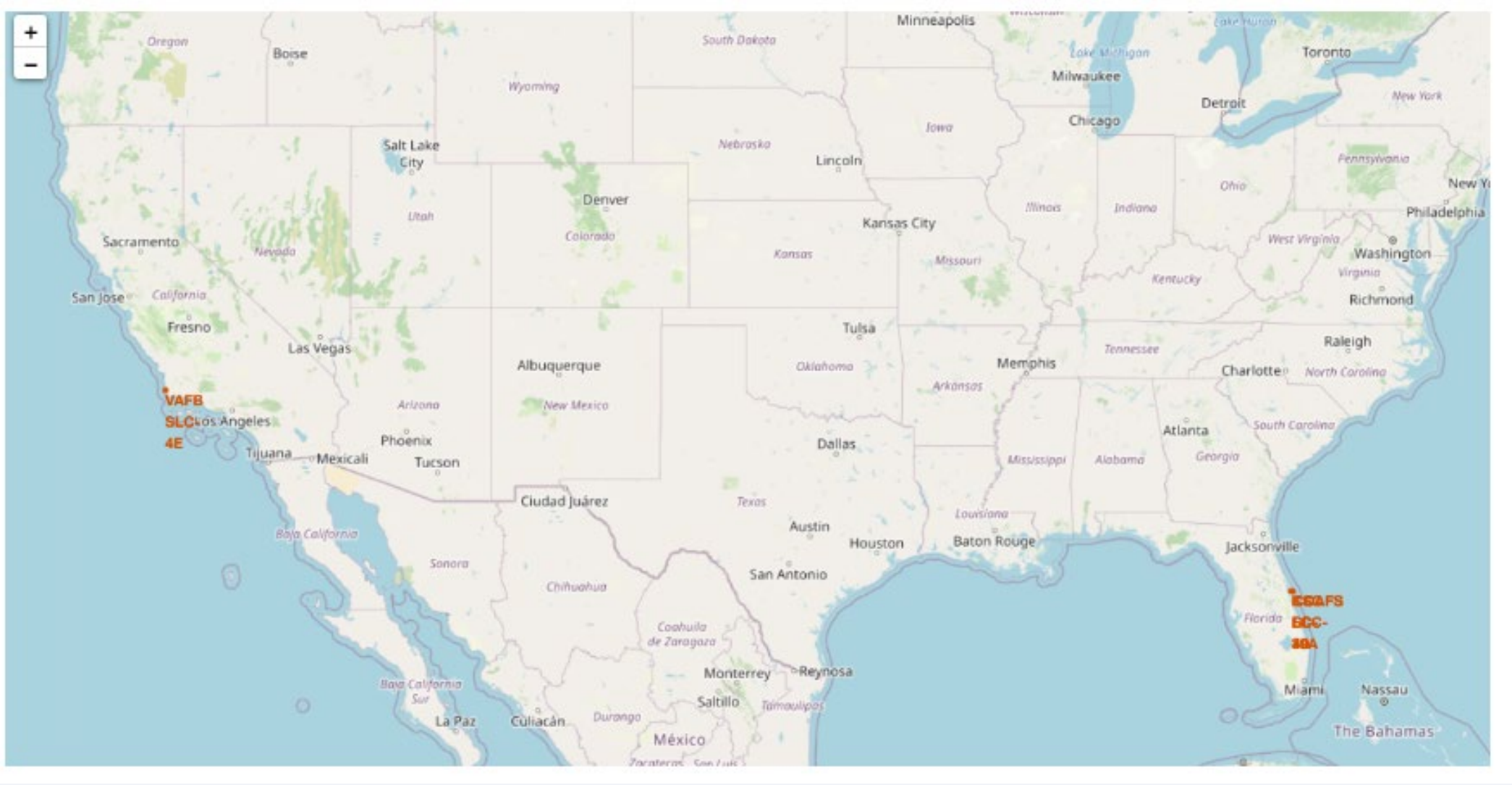
- Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

```
%sql select * from SPACEXTBL where Landing__Outcome like 'Success%' and (DATE between '2010-06-04' and '2017-03-20') order by date desc
```

```
 * sqlite:///my_data1.db
(sqlite3.OperationalError) no such column: Landing__Outcome
[SQL: select * from SPACEXTBL where Landing__Outcome like 'Success%' and (DATE between '2010-06-04' and '2017-03-20') order by date desc]
(Background on this error at: http://sqlalche.me/e/e3q8)
```

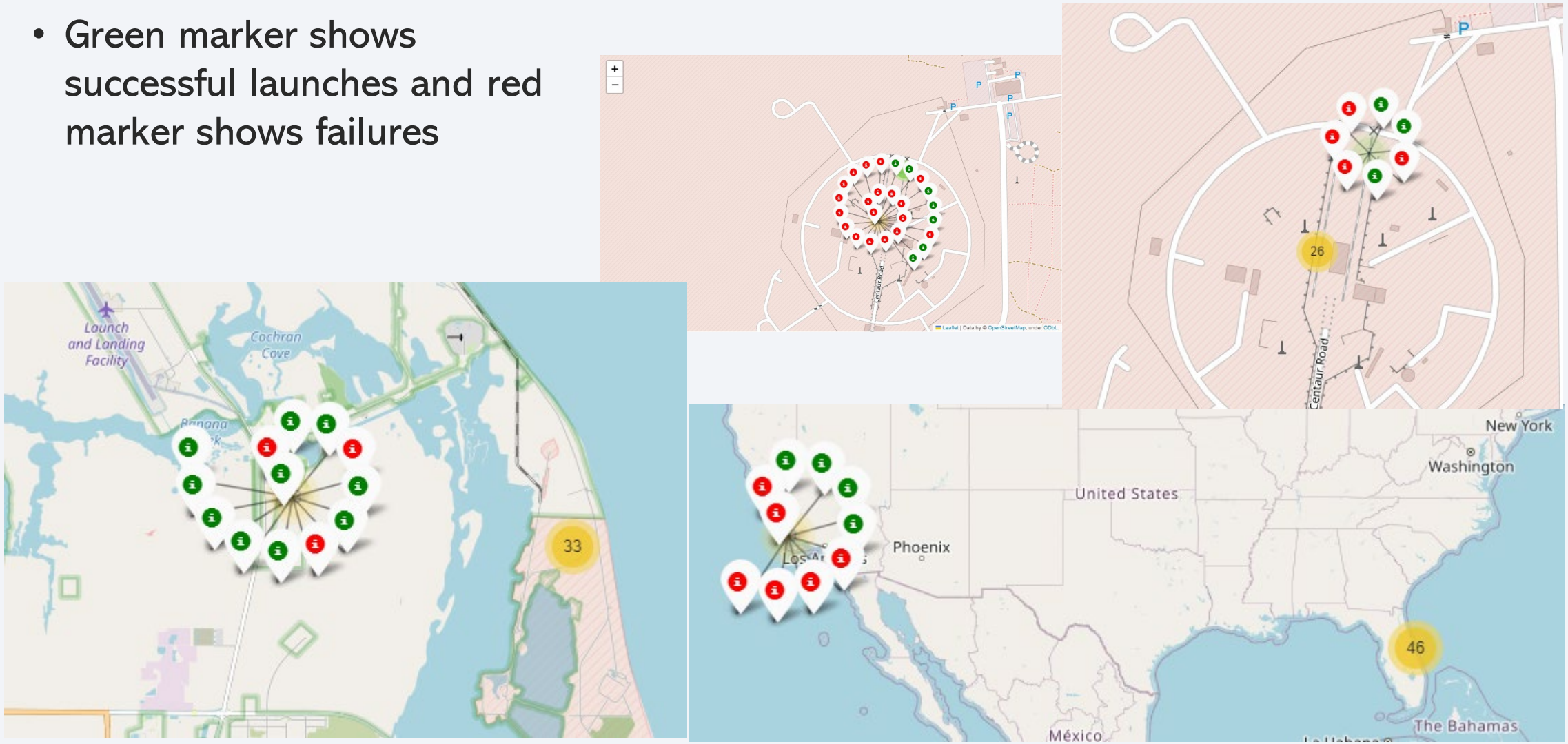

Section 3
Launch Sites
Proximities Analysis

<Folium Map Screenshot 1>

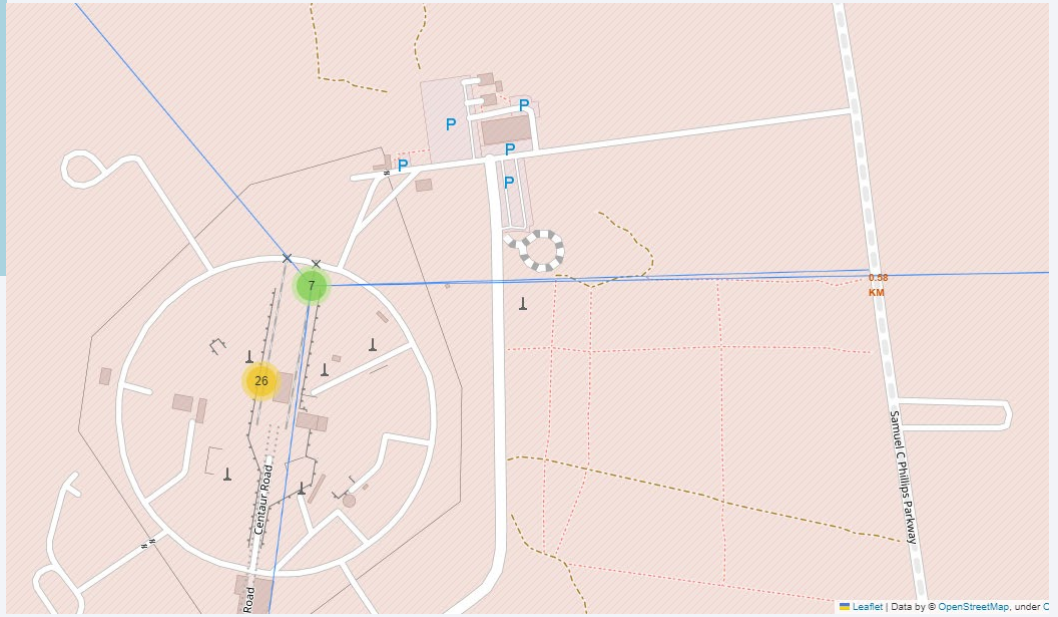


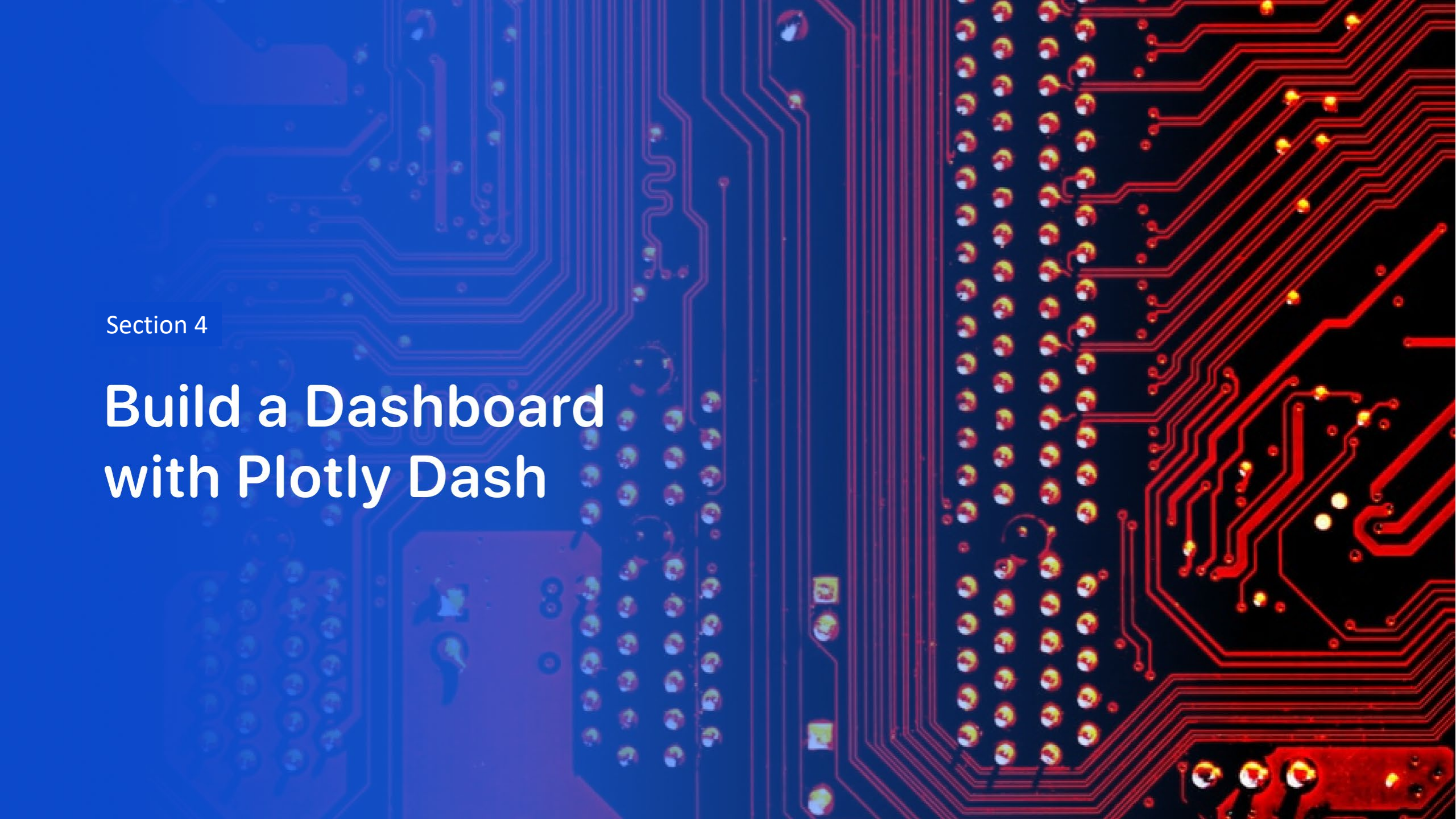
<Folium Map Screenshot 2>

- Green marker shows successful launches and red marker shows failures

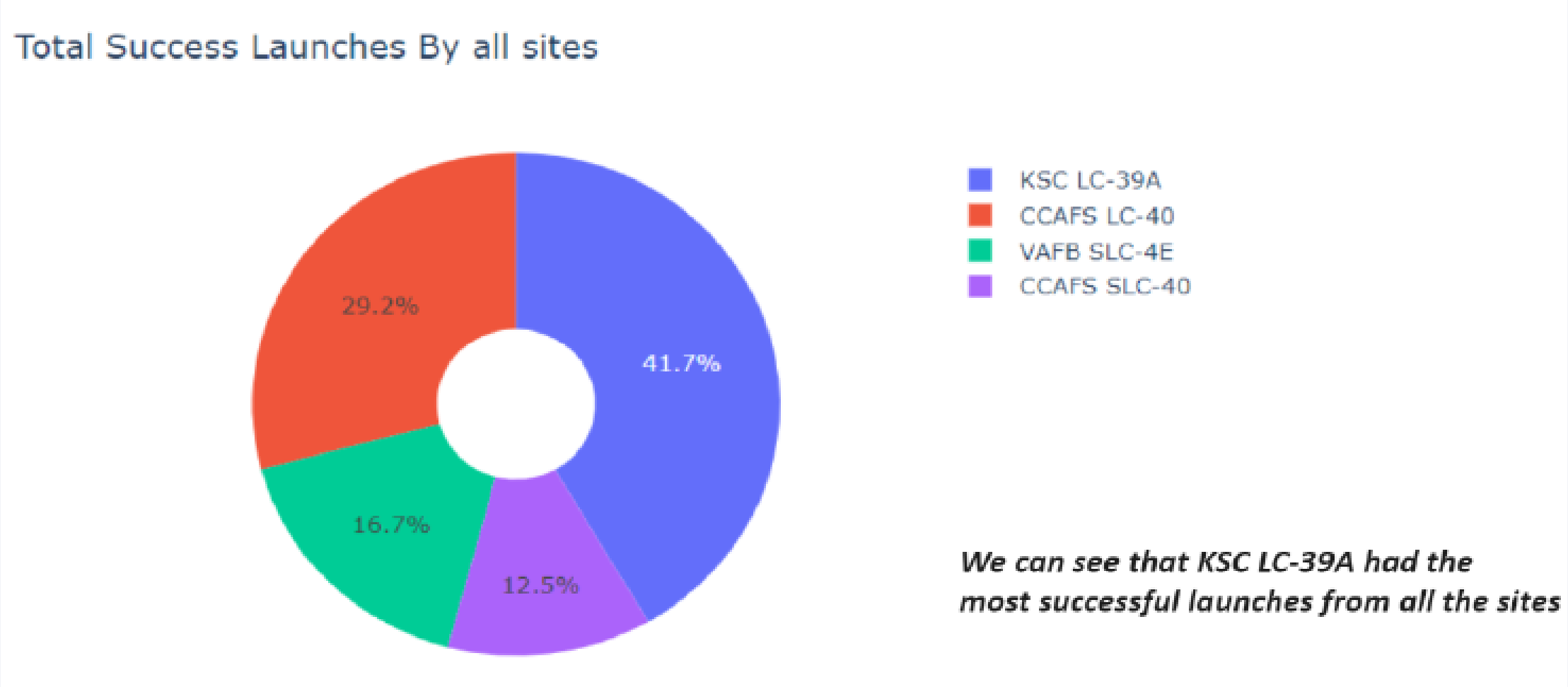


<Folium Map Screenshot 3>

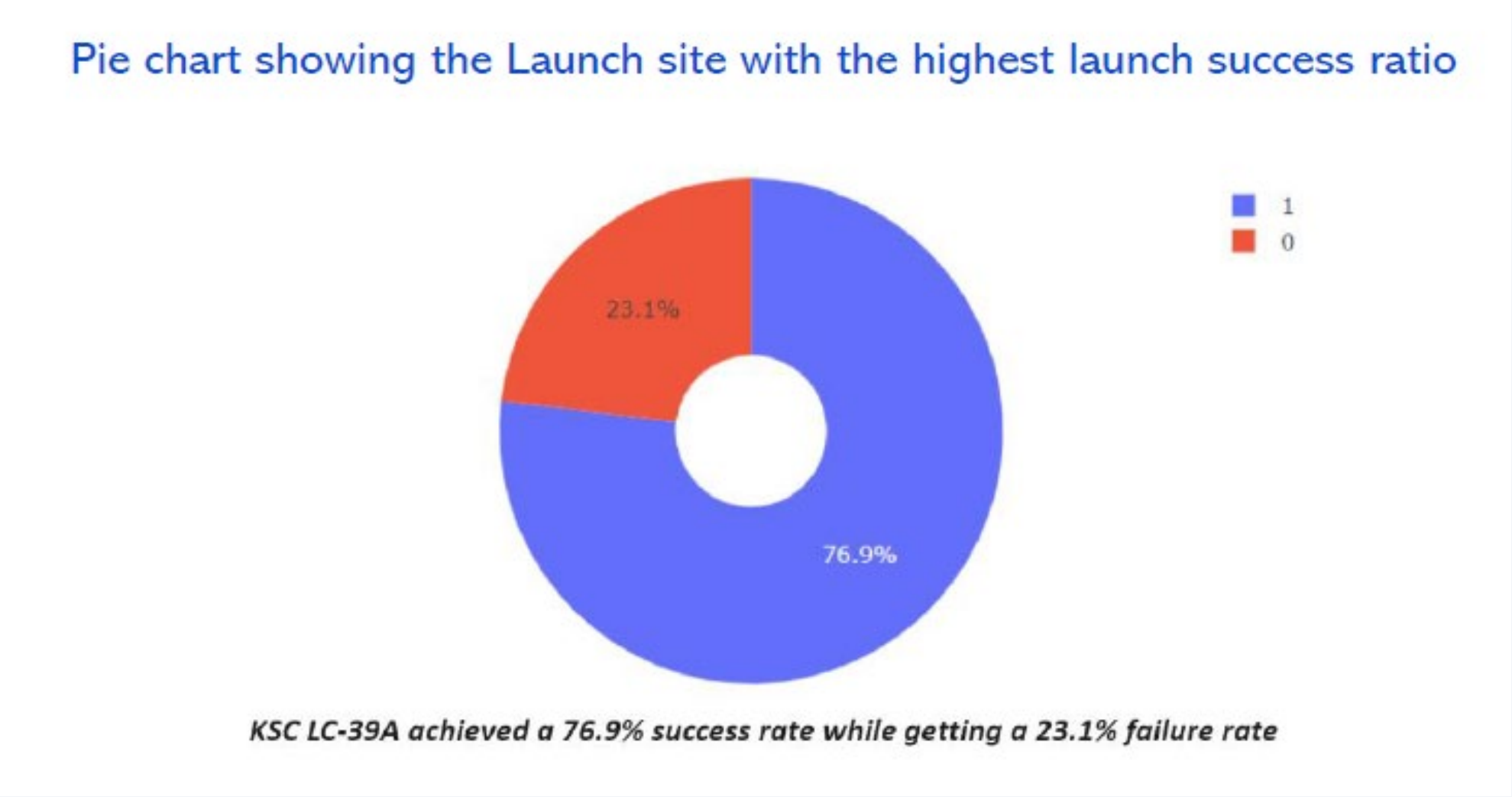



Section 4
Build a Dashboard with Plotly Dash

<Dashboard Screenshot 1>

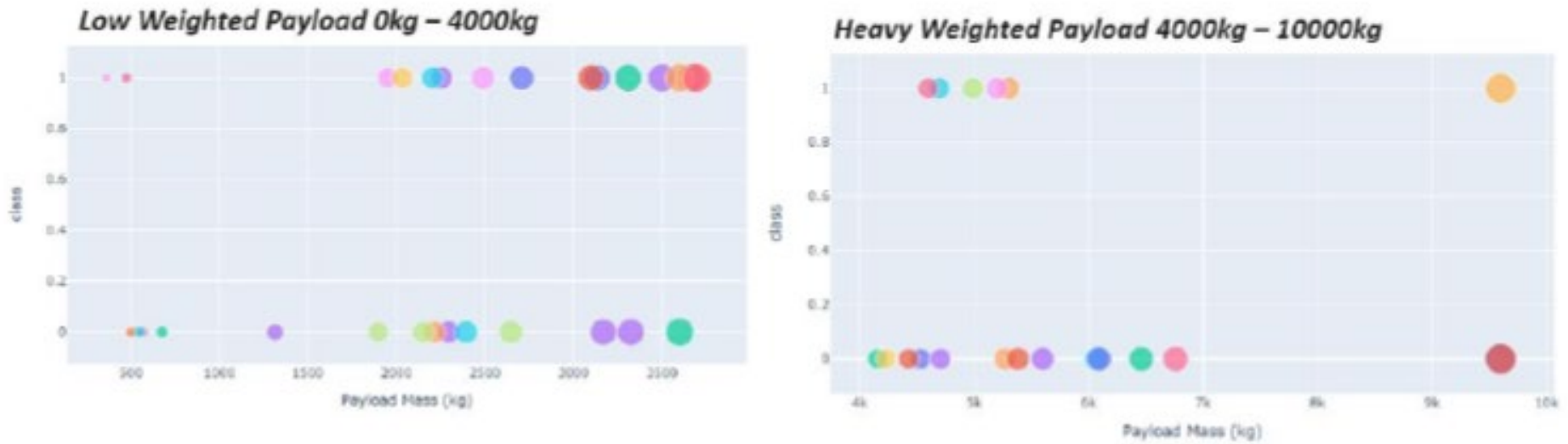


<Dashboard Screenshot 2>



<Dashboard Screenshot 3>

Scatter plot of Payload vs Launch Outcome for all sites, with different payload selected in the range slider



We can see the success rates for low weighted payloads is higher than the heavy weighted payloads

Section 5

Predictive Analysis (Classification)

Classification Accuracy

- Visualize the built model accuracy for all built classification models, in a bar chart

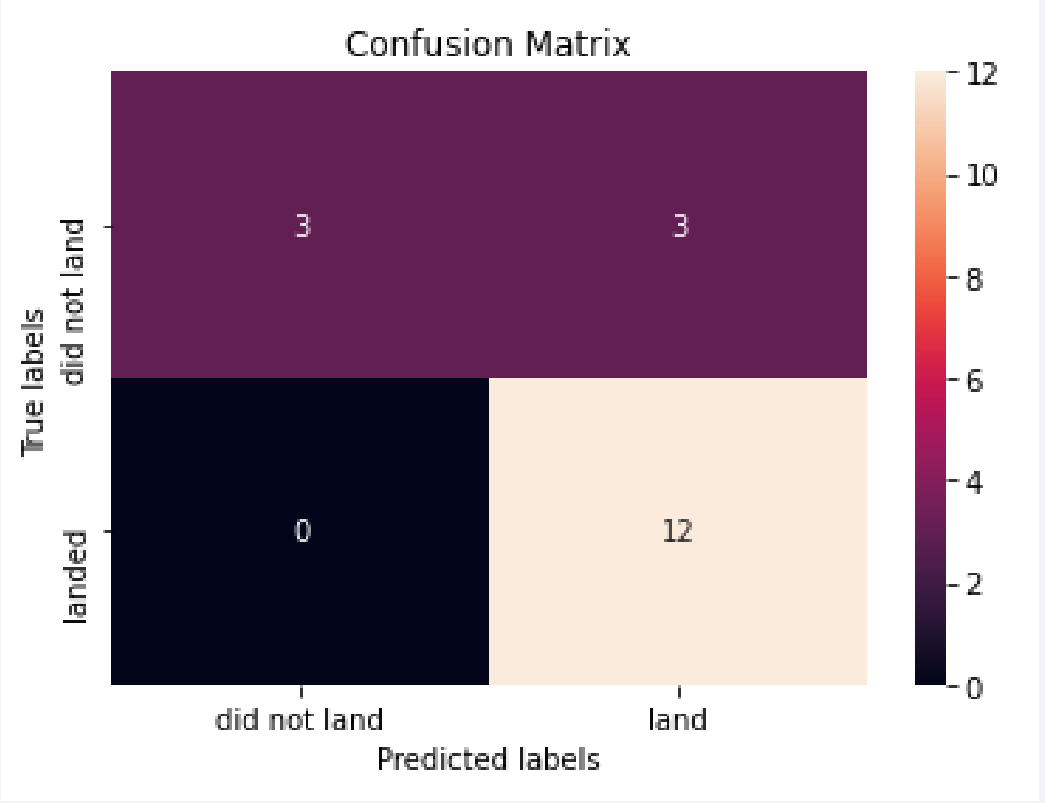
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

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

- The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes. The major problem is the false positive.

i.e., unsuccessful landing marked as successful landing by the classifier.



Conclusions

- The larger the flight amount at a launch site, the greater the success rate at a launch site.
- Launch success rate started to increase in 2023 till 2020.
- Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
- KSC LC-39A had the most successful launches of any sites.
- The Decision tree classifier is the best machine learning algorithm for this task.

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!