IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Rekha Unni
15-jun-2024

Outline

Executive Summary

Introduction

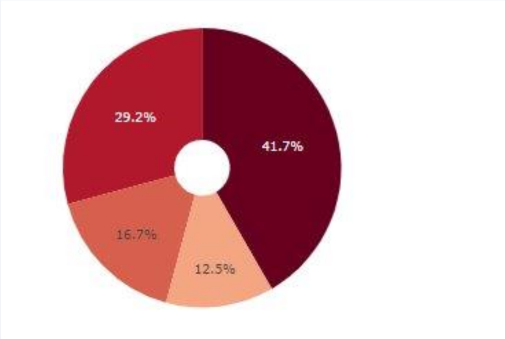
Methodology

Results

Conclusion

Appendix

Executive Summary



Summary of Methodologies

Data Collection - API , SQL and Web Scraping

Data Wrangling and Analysis

Interactive Maps with Folium

Exploratory Data Analysis with Python

Machine Learning Prediction

Summary of all Results

Exploratory Data Analysis result

Interactive analytics in screenshots -

Predictive Analytics result from Machine Learning Lab

Introduction

Project Background & Context

SpaceX is a revolutionary company who has disrupt the space industry by offering a rocket launches specifically Falcon 9 as low as 62 million dollars; while other providers cost upward of 165 million dollar each. Most of this saving thanks to SpaceX astounding idea to reuse the first stage of the launch by re-land the rocket to be used on the next mission. As a data scientist of a startup rivaling SpaceX, the goal of this crucial project is to use the information in identifying the right price to bid against SpaceX for a rocket launch

The problems for which solution is required:

- ❑Identifying all factors which influence the successful landing outcome.
- ❑The relationship between each variables and how it affects outcome
- ❑The best condition needed to increase the probability of successful landing

Section 1

Methodology

Methodology

Data Collection Methodology	SpaceX REST API Web Scraping from Wikipedia
Data Wrangling	One-hot encoding for categorical features (Transforming data for Machine Learning)
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 and Evaluate classification Models

Data Collection

- Data collection is the process of gathering and measuring information on targeted variables in an established system, which then enables one to answer relevant questions and evaluate outcomes.
- Dataset was collected by REST API and Web Scrapping from Wikipedia
- Response content as Json was processed with pandas dataframe
- Data Cleansing, Checks for missing values and replaced with meaningful data. Dataframe was filtered as per requirements for analysis
- Export data to flat file (CSV)

Data Collection – SpaceX API

GIT HUB - API Data Collection

Get request for rocket launch data using API

Convert Response to .json file

Data Cleansing and replace missing data

Create Data frame

Filter dataframe and export to Flat file

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

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

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

```
# Calculate the mean value of PayloadMass column
payloadmean = data_falcon9['PayloadMass'].mean()

# Replace the np.nan values with its mean value
data_falcon9.replace(np.nan,payloadmean)
```

```
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters anc
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single value in the list and replac
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting the date leaving the time
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

```
# Create a data from launch_dict
data = pd.DataFrame(launch_dict)
```

```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Data Collection - Scraping

GIT HUB Scaping – Data Collection

- Get response from HTML
- Creating BeautifulSoup Object
- Extract Column Names & details
- Create Dictionary and convert to dataframe
- Filter dataframe and export to Flat file

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

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

```
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('tr')

html_tables
```

```
column_names = []

# Apply find_all() function with `th` element on first_launch_table
temp = soup.find_all('th')
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
for x in range(len(temp)):
    try:
        name = extract_column_from_header(temp[x])
        if (name is not None and len(name)>0):
            column_names.append(name)
    except:
```

```
df=pd.DataFrame(launch_dict)
df.tail()
```

```
df.to_csv('spacex_web_scraped.csv', index=False)
```

Data Wrangling

GIT HUB: Data Wrangling

Data Wrangling is the process of cleaning and unifying messy and complex data sets for easy access and analysis
Here Outcomes are labelled: 1 as Successful and 0 as Unsuccessful landing

Calculate Number of Launches at each site

```
# Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()
```

Calculate Number of and occurrences at each orbit

```
# Apply value_counts on Orbit column
df['Orbit'].value_counts()
```

Calculate Number and occurrence of mission outcome per orbit type

```
# landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()

landing_outcomes
```

```
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
False Ocean     2
None ASDS       2
False RTLS      1
Name: Outcome, dtype: int64
```

Create landing outcome label from outcome column

```
landing_class = []
for key,value in df['Outcome'].items():
    if value in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

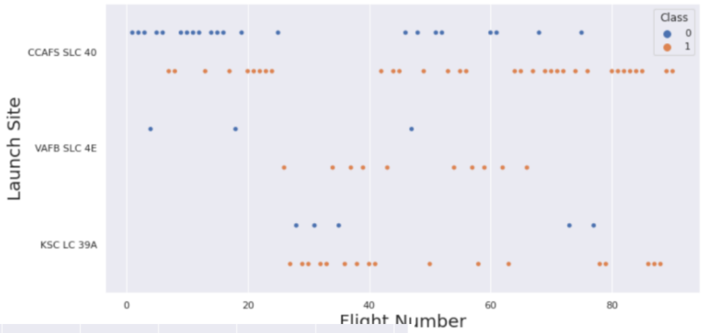
Export dataset as CSV

```
df.to_csv("dataset_part_2.csv", index=False)
```

EDA with Data Visualization

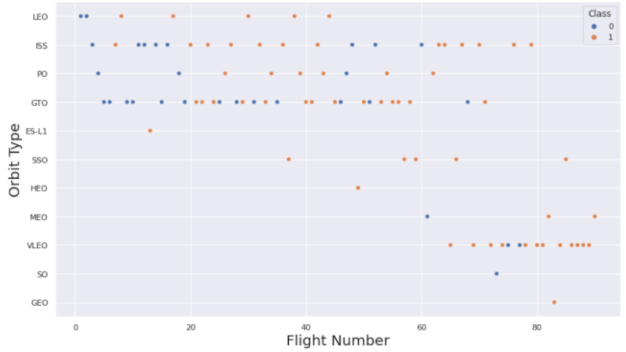
[Git HUB -EDA- Data Visualization]

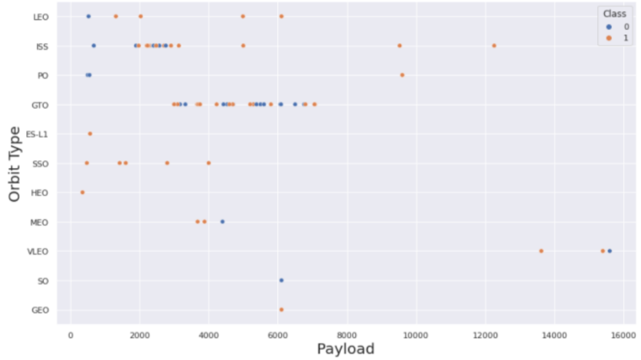
Exploratory Data Analysis is an approach of analysing data sets to summarise their main characteristics using statistical graphics and other visualization trends



Scatter graph was used to find the relationship/dependency between the attributes :

- Payload and Flight Number.
- Flight Number and Launch Site.
- Payload and Launch Site.
- Flight Number and Orbit Type.
- Payload and Orbit Type





Scatter plots show dependency of attributes on each other. Once a pattern is determined from the graphs. It's very easy to see which factors affecting the most to the success of the landing outcomes.

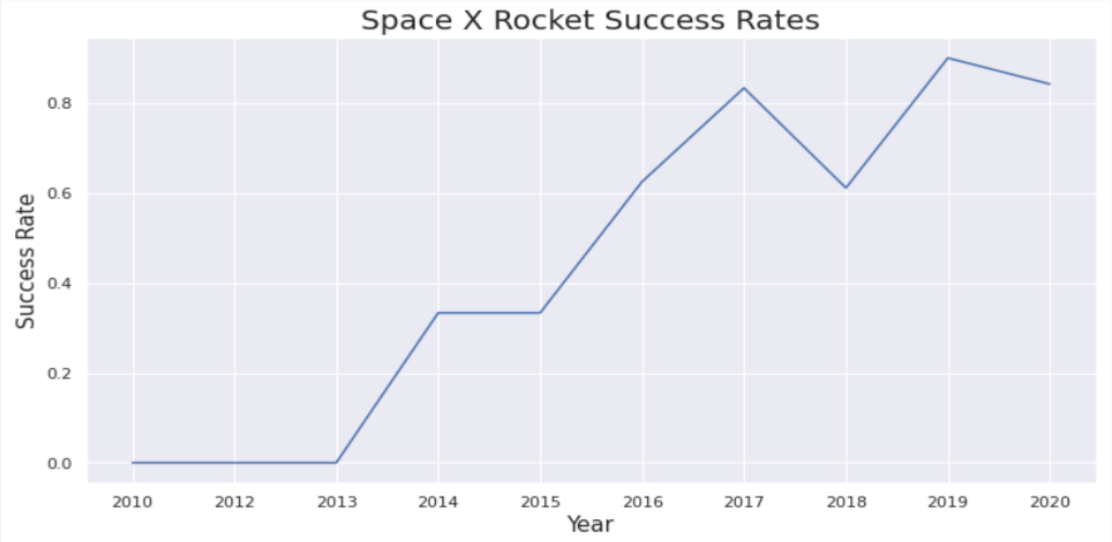
EDA with Data Visualization (contd…)

[Git HUB -EDA- Data Visualization]

Based on hints of the relationships using scatter plot - further visualization tools such as bar graph and line plots graph were used for further analysis.

1. Bar graphs is one of the easiest way to interpret the relationship between the attributes. In this case, we will use the bar graph to determine which orbits have the highest probability of success.

2. Line graph show a trends or pattern of the attribute over time which in this case, is used for see the launch success yearly trend.
3. Feature Engineering trend is to be used in success prediction in the future module by using the dummy variables for categorical columns.



EDA with SQL

GIT HUB: EDA with SQL

SQL is most powerful tool to analyse real world data stored in databases , it helps in analysing data and drawing useful insights

For this project, SQL queries were performed to gather information from Datasets:-

- Displaying the names of the launch sites
- Displaying 5 records where launch sites begin with the string ‘CCA’.
- Displaying the total payload mass carried by booster launched by NASA (CRS).
- Displaying the average payload mass carried by booster version F9 v1.1. –
- Listing the date when the first successful landing outcome in ground pad was achieved. –
- Listing the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000.
- Listing the total number of successful and failure mission outcomes.
- Listing the names of the booster_versions which have carried the maximum payload mass.
- Listing the failed landing_outcomes in drone ship, their booster versions, and launch sites names for in year 2015.
- Rank the count of landing outcomes or success between the date 2010-06-04 and 2017-03-20, in descending order

Build an Interactive Map with Folium

GIT HUB – MAP with Folium

Folium makes it easy To visualize the launch data into an interactive map.

We took the latitude and longitude coordinates at each launch site and added a circle marker around each launch site with a label of the name of the launch site. We then assigned the dataframe launch_outcomes(failure,success) to classes 0 and 1 with Red and Green markers on the map in MarkerCluster()

Haversine's formula was used to calculated the distance of the launch sites to various landmark to find answer to the questions of:

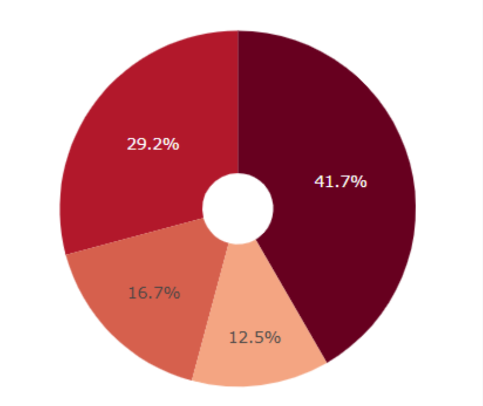
- How close the launch sites with railways, highways and coastlines?
- How close the launch sites with nearby cities?

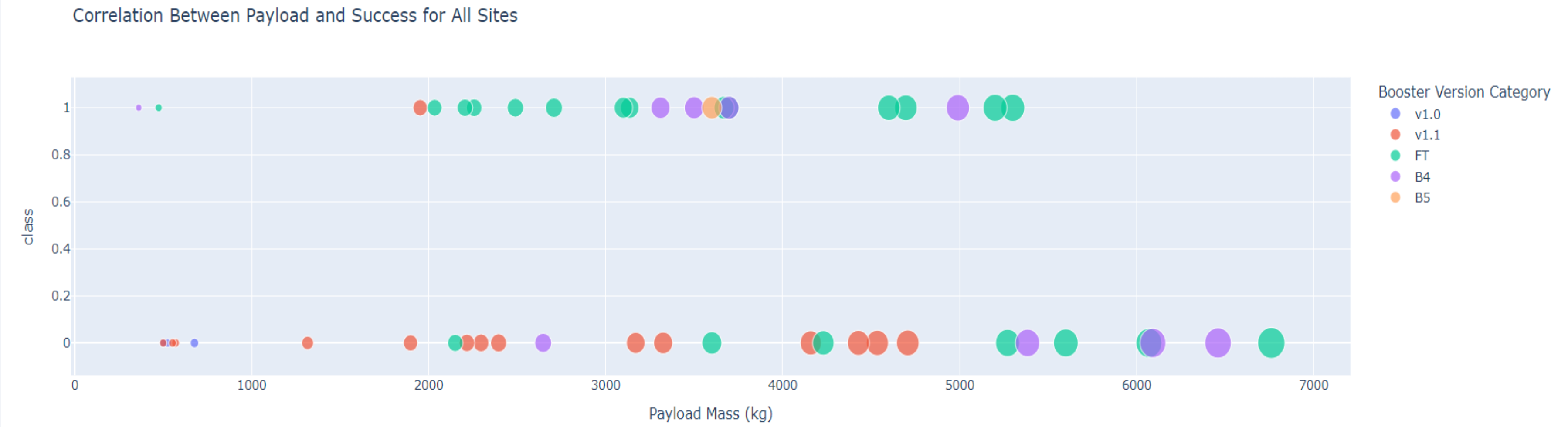
```
# Function to assign color to launch outcome
def assign_marker_color(launch_outcome):
    if launch_outcome == 1:
        return 'green'
    else:
        return 'red'

spacex_df['marker_color'] = spacex_df['class'].apply(assign_marker_color)
spacex_df.tail(10)
```

Build a Dashboard with Plotly Dash

[GIT HUB: Plotly Python Zip]

- Interactive dashboard was built with Plotly dash allowing the Analysts to play around with the data as they need.
- Pie charts were used to show the total launches by a certain sites.
- Scatter graph displayed the relationship with Outcome and Payload Mass (Kg) for the different booster version





Predictive Analysis (Classification)

GITHUB – Predictive Analysis

Building the Model

- Load the dataset into NumPy and Pandas
- Transform the data and split into training and test datasets
- Decide which ML should be used
- Set the parameters and algorithms to GridSearchCVand fit to dataset

Evaluating the Model

- Check the accuracy of each model
- Get tuned hyperparameters for each type of algorithms
- Plot the Confusion Matrix

Improving the Model

- Use Feature Engineering and Algorithm Tuning

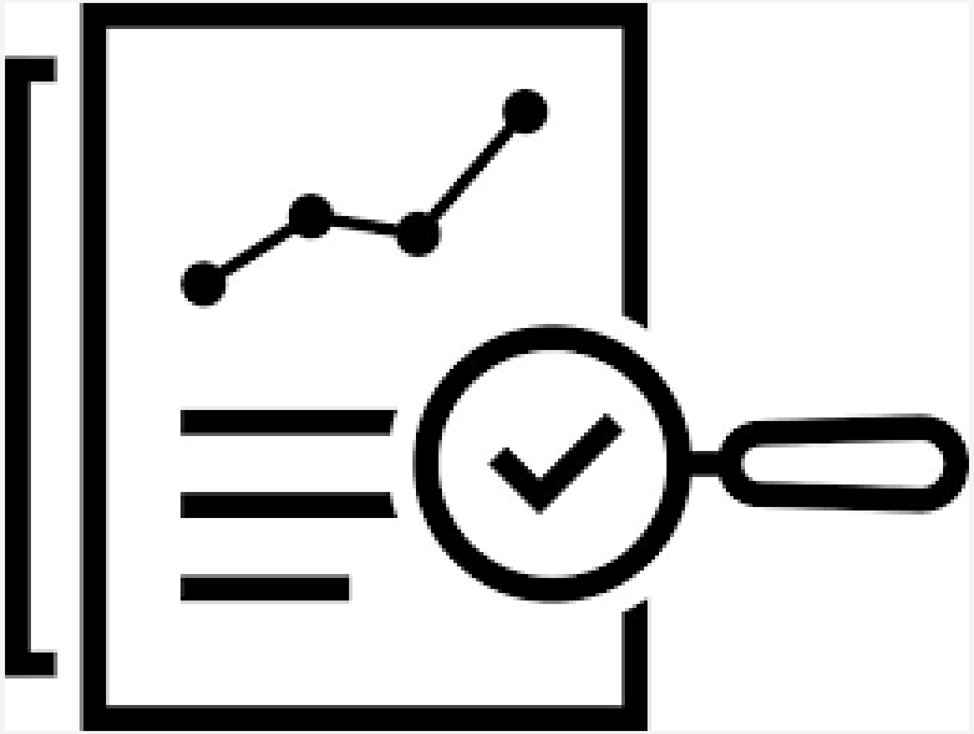
Finding the Best Model

- Model with best accuracy score will be best performing Model

Results

Results will be categorized as:

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

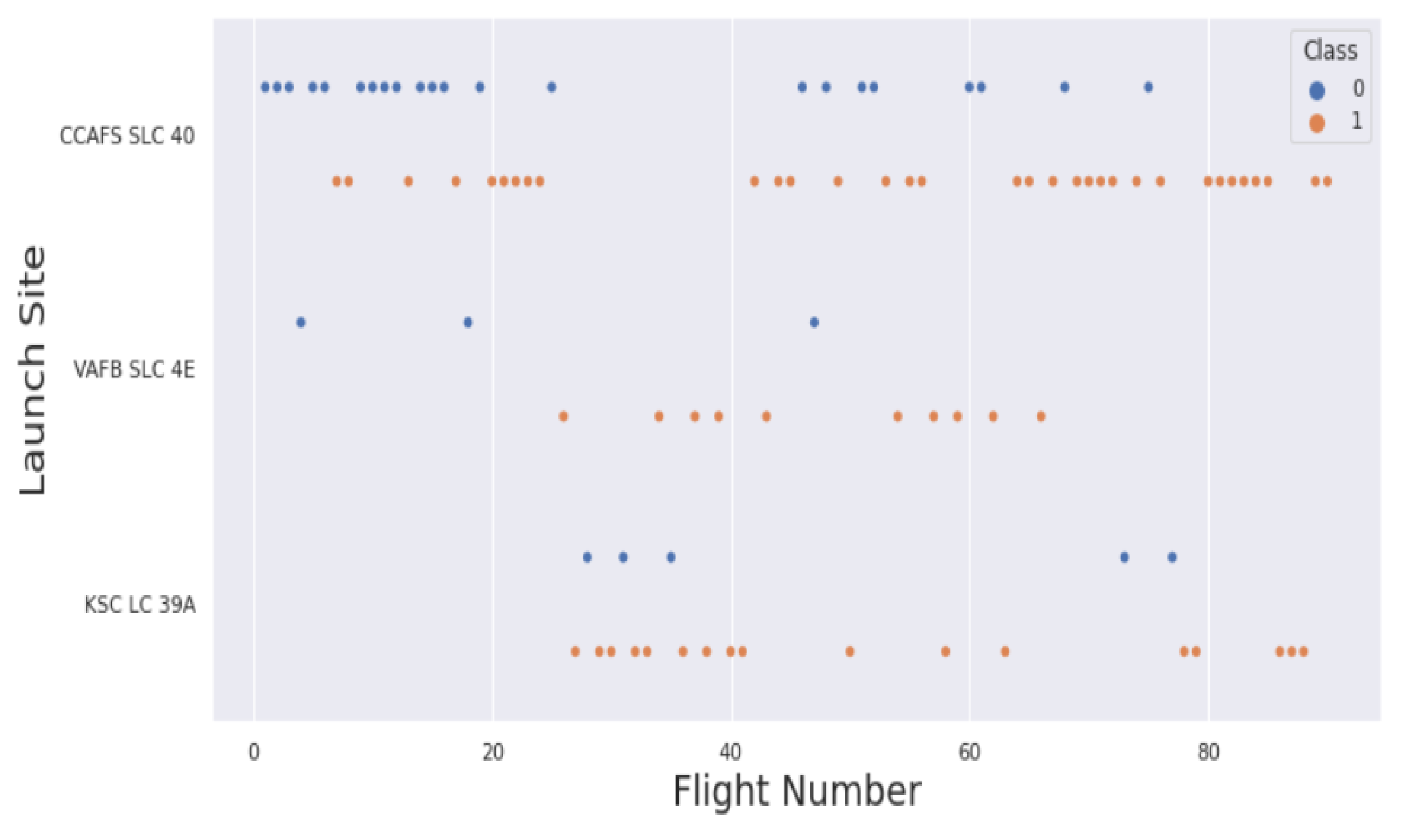


Section 2

Insights drawn from EDA

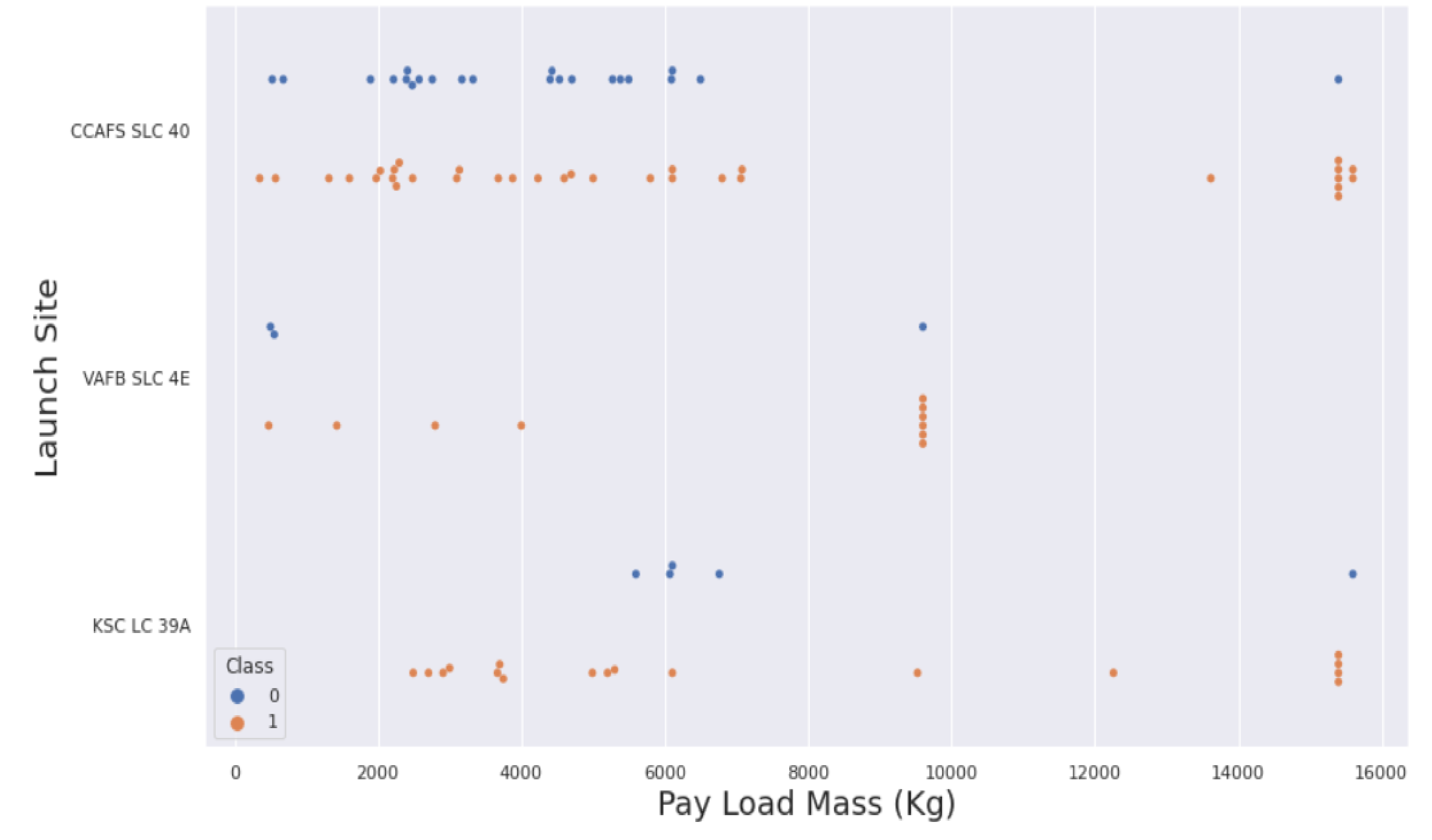
Flight Number vs. Launch Site

This scatter plot shows that the higher the flight numbers of the launch site, the greater the success rate will be. However, site CCAFS SLC40 shows the least pattern of this.



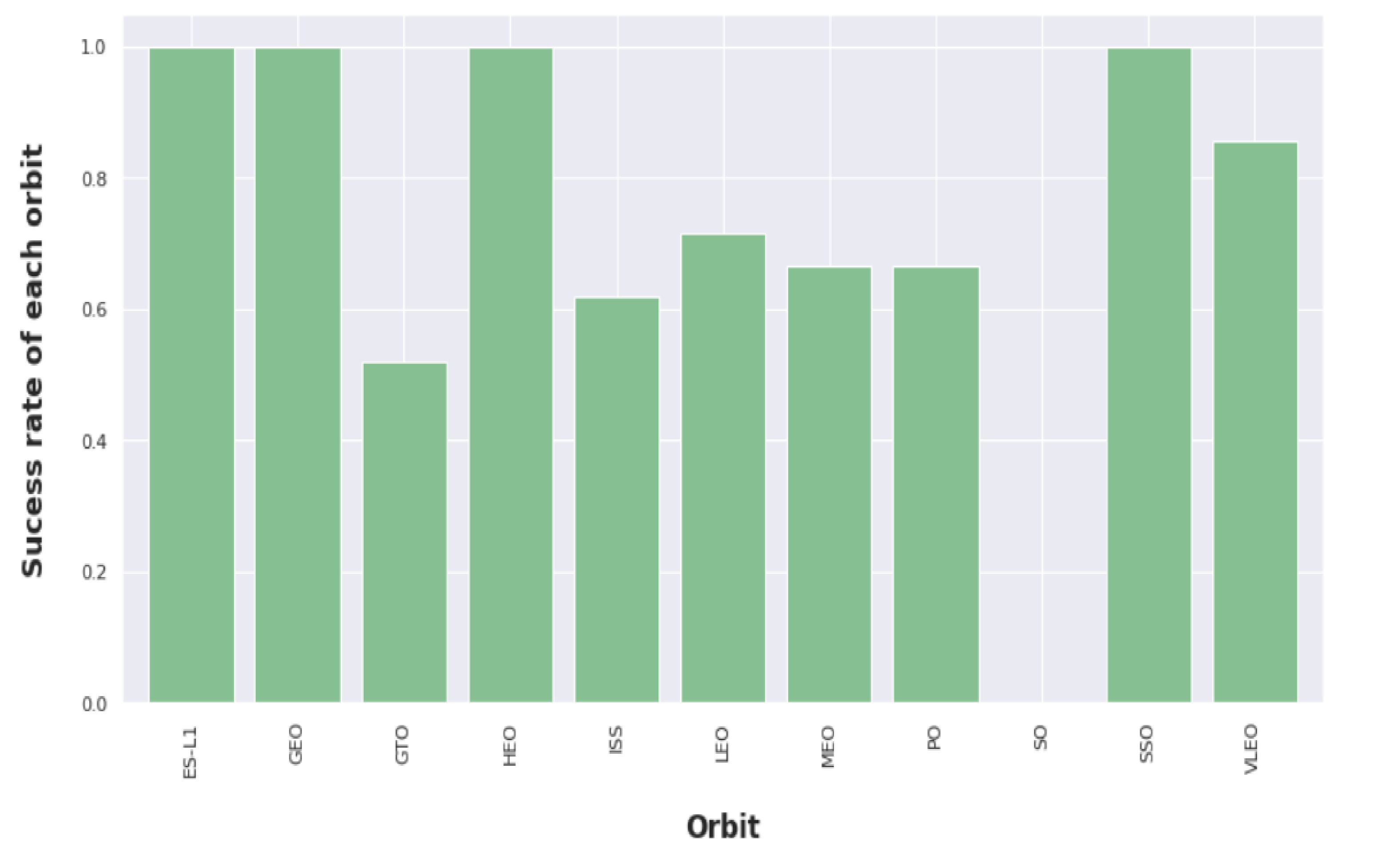
Payload vs. Launch Site

This scatter plot shows that the greater pay load mass (>7000kg), the probability of the success rate will be highly increased. However, there is no clear pattern to say the launch site is dependent to the pay load mass for the success rate.



Success Rate vs. Orbit Type

This plot depicts the possibility of the orbits to influence the landing outcomes as some orbits has 100% success rate such as SSO, HEO, GEO AND ES-L1 while SO orbit produced 0% rate of success. However, deeper analysis show that some of this orbits has only 1 occurrence such as GEO, SO, HEO and ES-L1 which mean this data need more dataset to see pattern or trend before we draw any conclusion.



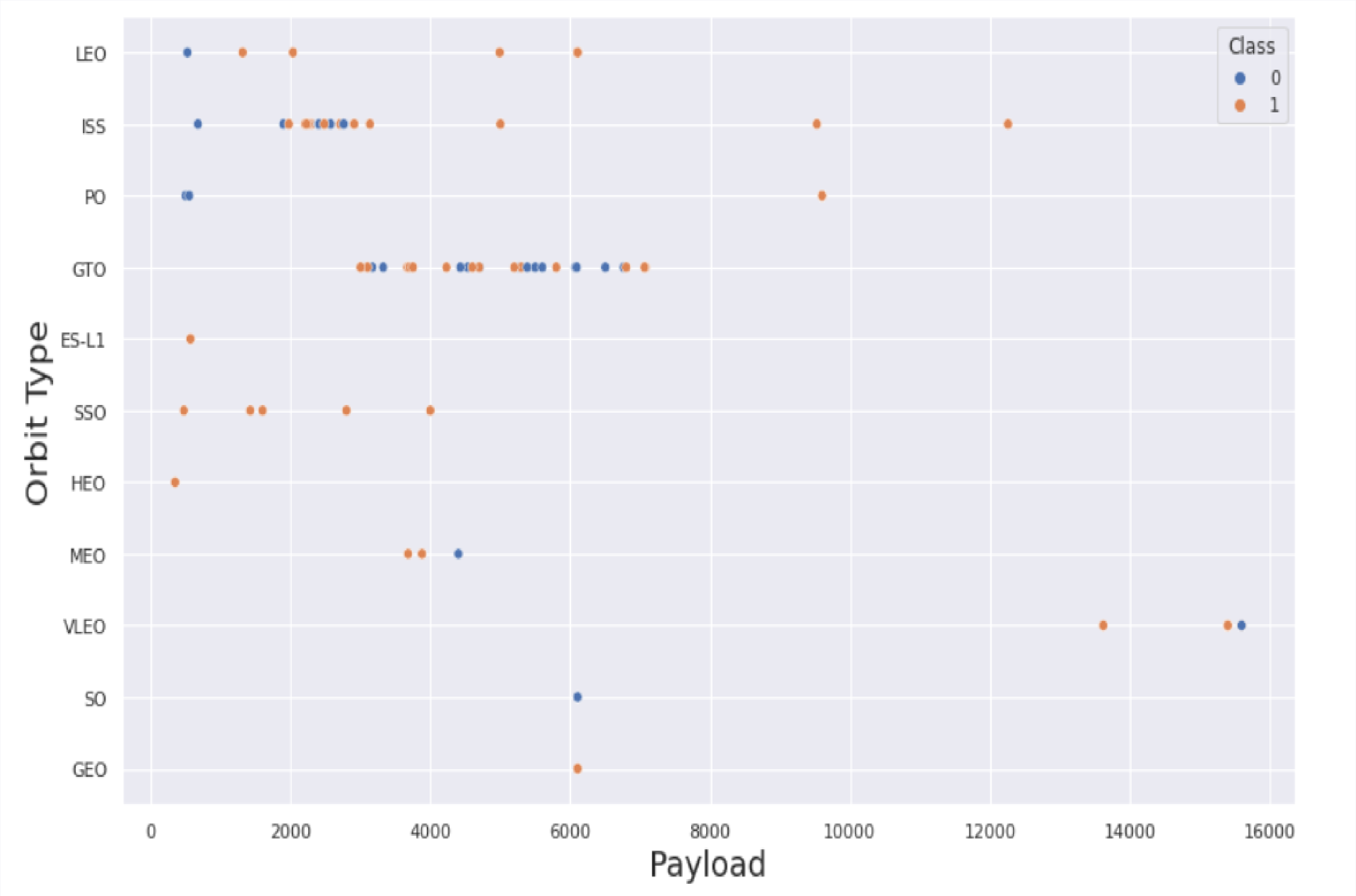
Flight Number vs. Orbit Type

This scatter plot shows that generally, the higher the flight number on each orbits, the greater the success rate (especially LEO orbit) except for GTO orbit which depicts no relationship between both attributes. Orbit that only has 1 occurrence should also be excluded from above statement as it's needed more dataset.

Payload vs. Orbit Type

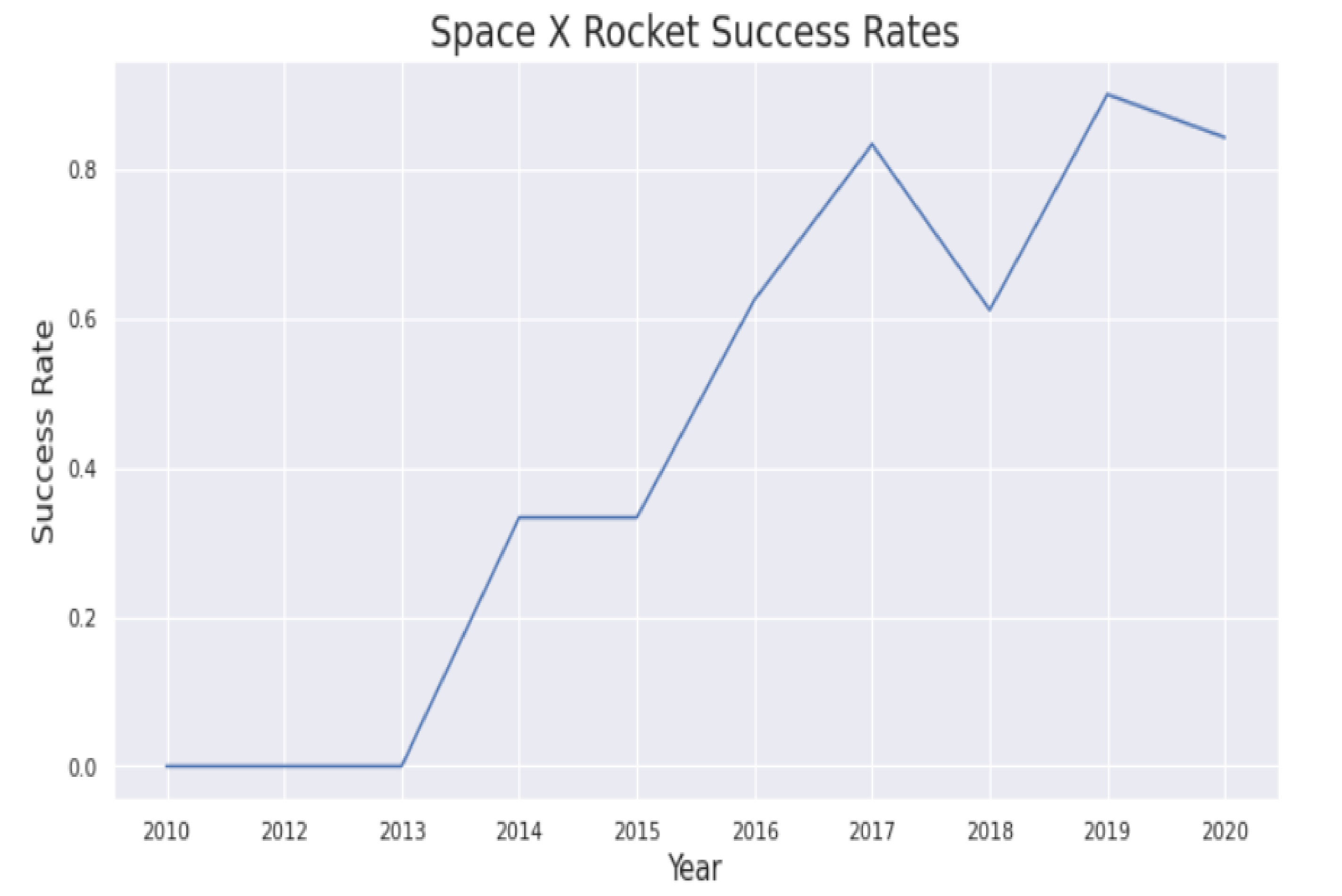
Heavier payload has positive impact on LEO, ISS and P0 orbit. However, it has negative impact on MEO and VLEO orbit.

GTO orbit seem to depict no relation between the attributes. Meanwhile, again, SO, GEO and HEO orbit need more dataset to see any pattern or trend



Launch Success Yearly Trend

This figures clearly depicted and increasing trend from the year 2013 until 2020 with dip in 2019. If trend continues onward, success rate will steadily increase until reaching 1/100% success rate.



All Launch Site Names

We used the key word DISTINCT to show only unique launch sites from the SpaceX Table data

```
[9]: %sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTBL;

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

[9]:

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

Launch Site Names 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'

```
[10]: %sql SELECT LAUNCH_SITE FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
```

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

[10]:

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

Total Payload Mass

We calculated the total payload carried by boosters from NASA as 45596 using the query below

```
[13]: %sql SELECT SUM (PAYLOAD_MASS__kg_) FROM SPACEXTBL WHERE CUSTOMER like 'NASA (CRS)' ;

 * sqlite:///my_data1.db
Done.

[13]:
```

SUM (PAYLOAD_MASS__kg_)
45596

Average Payload Mass by F9 v1.1

We calculated the average payload mass carried by booster version F9 v1.1 as 2928.4

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

```
[14]: %sql SELECT AVG(PAYLOAD_MASS__KG_) AS "Average Payload Mass by Booster Version F9 v1.1" FROM SPACEXTBL \
WHERE BOOSTER_VERSION = 'F9 v1.1';
```

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

[14]:

Average Payload Mass by Booster Version F9 v1.1
2928.4

First Successful Ground Landing Date

min() function was used to find the result
We observed that the dates of the first successful landing outcome on ground pad was 22nd December 2015

```
[16]: %sql SELECT MIN(DATE) AS "First Succesful Landing Outcome in Ground Pad" FROM SPACEXTBL \
      WHERE LANDING_OUTCOME = 'Success (ground pad)';

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

[16]:

First Succesful Landing Outcome in Ground Pad
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

WHERE clause was used to filter for boosters which have successfully landed on drone ship and applied the AND condition to determine successful landing with payload mass greater than 4000 but less than 6000

```
[17]: %sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE LANDING_OUTCOME = 'Success (drone ship)' \
AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;

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

[17]:

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

Total Number of Successful and Failure Mission Outcomes

We used wildcard like '%' to filter for WHERE Mission_Outcome was a success or a failure.

```
[23]: %sql SELECT COUNT(MISSION_OUTCOME) AS "Successful Mission" FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE 'Success%' ;

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

[23]:

Successful Mission
100

```
[25]: %sql SELECT COUNT(MISSION_OUTCOME) AS "Failure Mission" FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE 'Failure%';

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

[25]:

Failure Mission
1

```
[27]: %sql SELECT COUNT(MISSION_OUTCOME) AS "Total Number of Successful and Failure Mission" FROM SPACEXTBL \
WHERE MISSION_OUTCOME LIKE 'Success%' OR MISSION_OUTCOME LIKE 'Failure%';

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

[27]:

Total Number of Successful and Failure Mission
101

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

```
[28]: %sql SELECT DISTINCT BOOSTER_VERSION AS "Booster Versions which carried the Maximum Payload Mass" FROM SPACEXTBL \
WHERE PAYLOAD_MASS__KG_ =(SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);

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

[28]:

Booster Versions which carried the Maximum Payload Mass
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

We used a combinations of the WHERE clause, SUBSTR, AND conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015

```
[30]: %sql SELECT substr(DATE,6,2) as Month, BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WHERE substr(DATE,0,5) = '2015' AND \
LANDING_OUTCOME = 'Failure (drone ship)';

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

[30]:

Month	Booster_Version	Launch_Site
01	F9 v1.1 B1012	CCAFS LC-40
04	F9 v1.1 B1015	CCAFS LC-40

*SQL Lite has limitation as it does not support month names, so Lab recommends using Substr to achieve the result

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

We selected Landing outcomes and the COUNT of landing outcomes from the data and used the WHERE clause to filter for landing outcomes BETWEEN 2010-06-04 to 2017-03-20. We applied the GROUP BY clause to group the landing outcomes and the ORDER BY clause to order the grouped landing outcome in descending order.

```
[33]: %sql SELECT LANDING_OUTCOME as "Landing Outcome", COUNT(LANDING_OUTCOME) AS "Total Count" FROM SPACEXTBL \
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' \
GROUP BY  LANDING_OUTCOME \
ORDER BY COUNT(LANDING_OUTCOME) DESC ;
```

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

[33]:

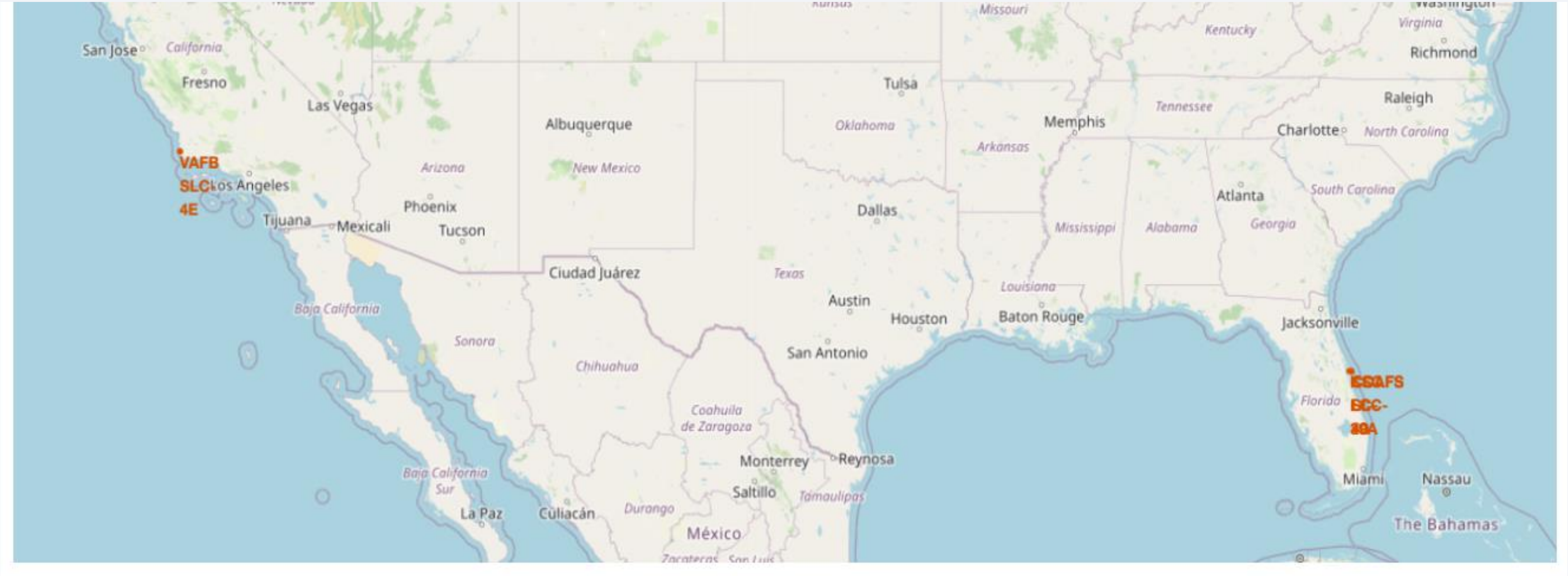
Landing Outcome	Total Count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

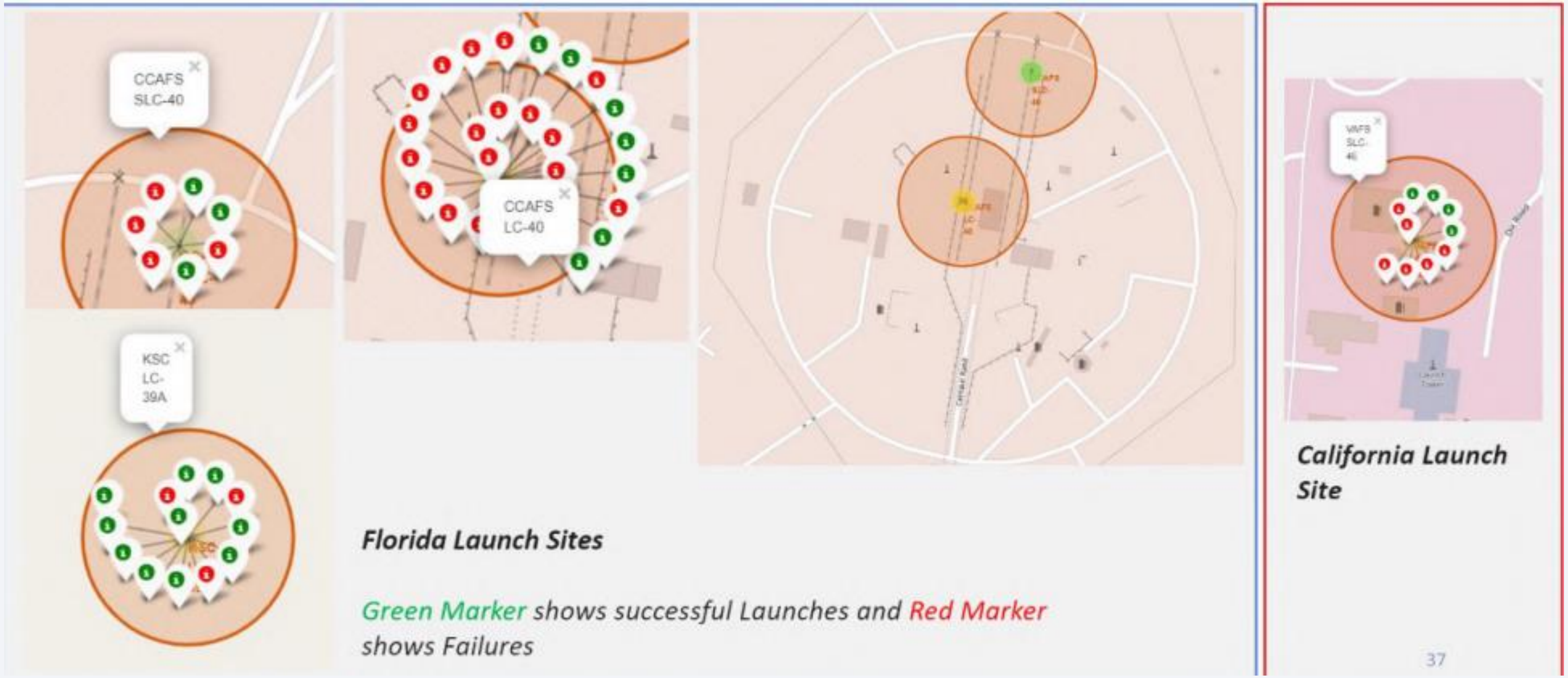
Launch Sites Proximities Analysis

Launch Sites on Folium Map

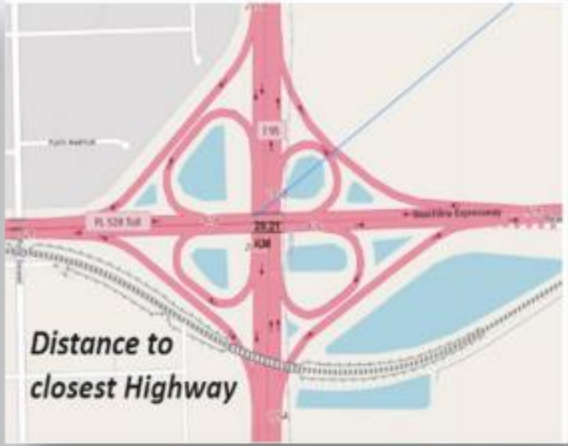
We can see that all the SpaceX launch sites are in and around United States



Marker with Launch Sites with Colored Labels



Launch Site Distances to Landmarks & Railways



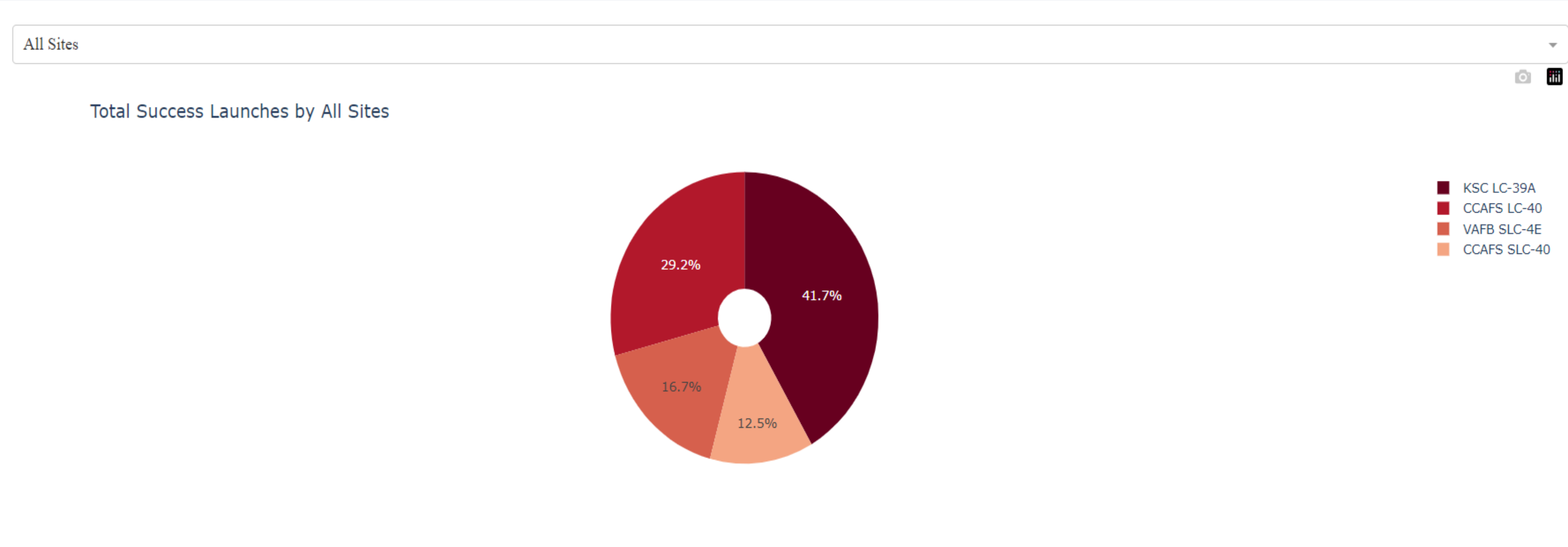
- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

Section 4

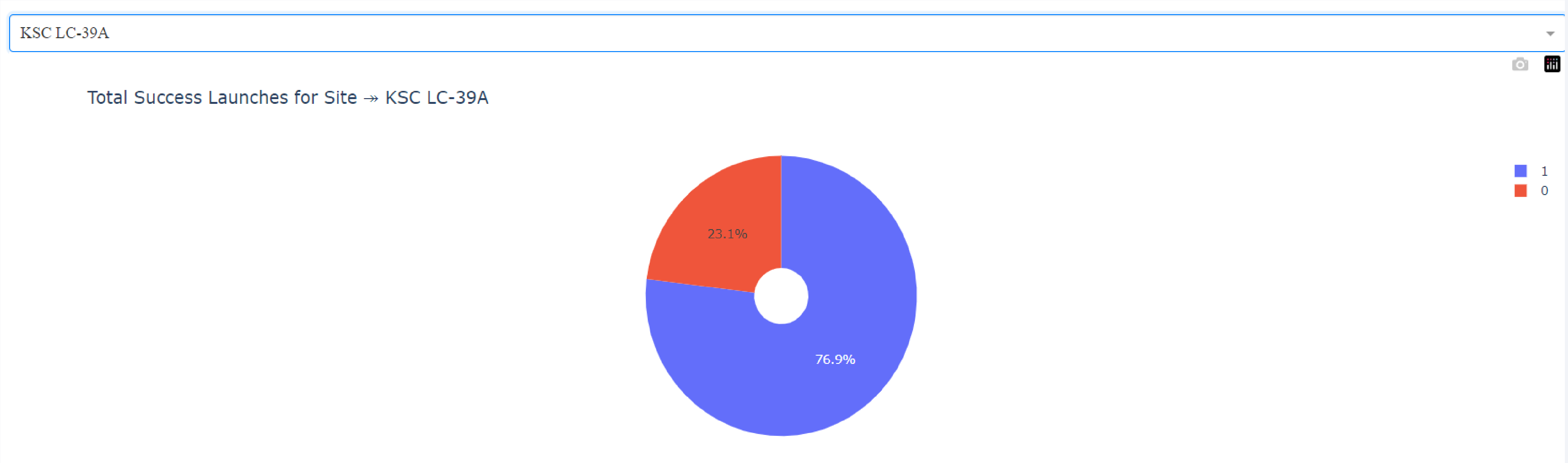
Build a Dashboard with Plotly Dash

Launch Success Site for All Sites

We can see that KSC LC-39A had the most successful launches from all sites



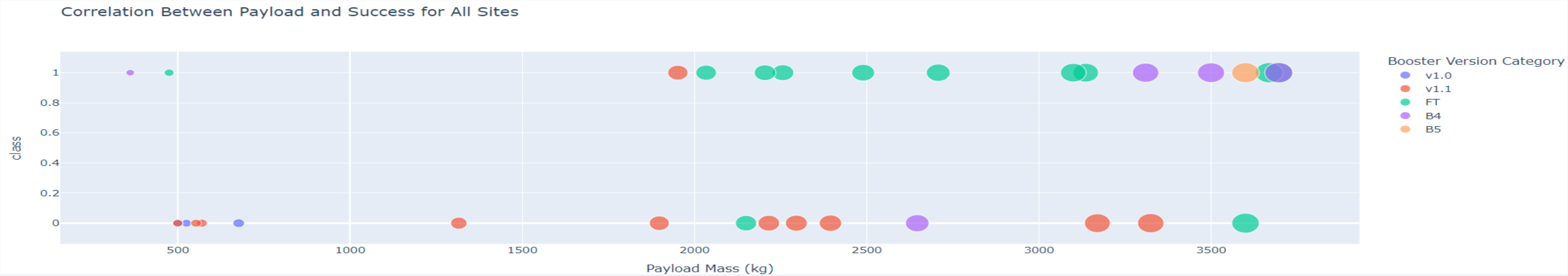
Launch Site with Highest launch Success Ratio



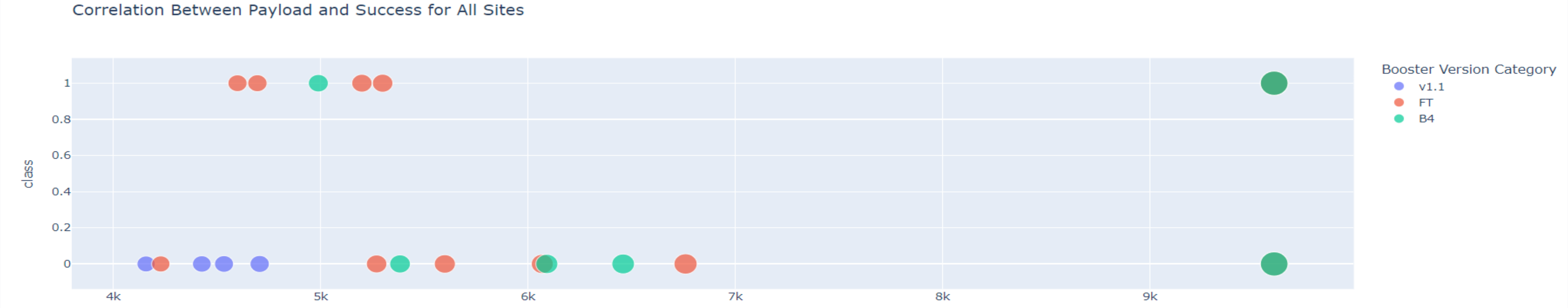
KSC LC-39A has highest launch Success Rate with 76.9% success and 23.1% failure rate and FT Booster version of F9 has highest launch success rate

Payload Vs Launch Outcome for All Sites

We can see that success rate for Low weighted Payloads is higher than heavy weighted Payloads



Weighted Payloads from 0 to 4000 Kg



Weighted Payloads from 4000 – 10,000 Kg

Section 5

Predictive Analysis (Classification)

Classification Accuracy

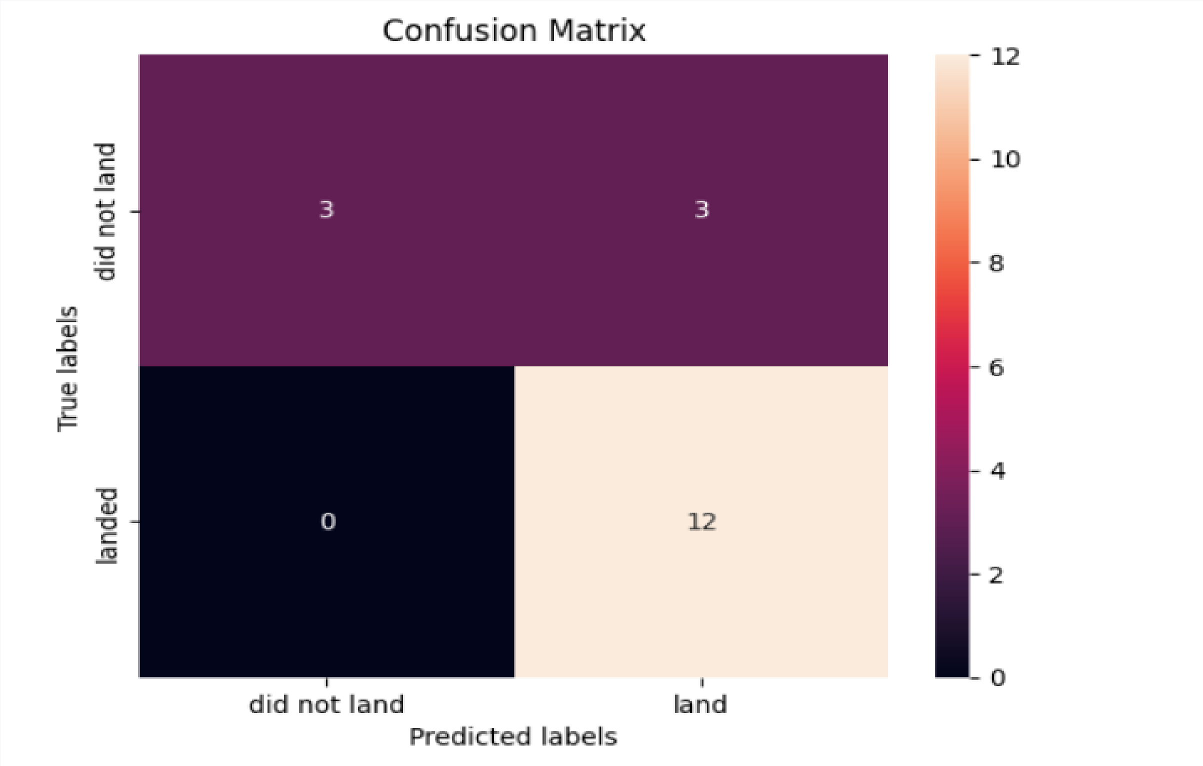
As we can see below: we could identify that the best algorithm is to be the Tree Algorithm which have the highest classification accuracy

```
algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_}
bestalgorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
if bestalgorithm == 'Tree':
    print('Best Params is :',tree_cv.best_params_)
if bestalgorithm == 'KNN':
    print('Best Params is :',knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best Params is :',logreg_cv.best_params_)
```

```
Best Algorithm is Tree with a score of 0.8732142857142857
Best Params is : {'criterion': 'entropy', 'max_depth': 4, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'best'}
```

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



		Predicted Values: No	Predicted Values: Yes
Actual Values	No	**True Negative**	**False Positive**
	Yes	False Negative	False Positive

Conclusions

1. The Tree Classifier Algorithm is the best Machine Learning approach for this dataset.
2. The low weighted payloads (which define as 4000kg and below) performed better than the heavy weighted payloads.
3. Starting from the year 2013, the success rate for SpaceX launches is increased, directly proportional time in years to 2020, which it will eventually perfect the launches in the future.
4. KSC LC-39A have the most successful launches of any sites - 76.9%
5. SSO orbit have the most success rate; 100% and more than 1 occurrence. GTO has lowest success rate
6. As Number of flights increase, the first stage is more likely to land successfully

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!