IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

SAMARTH KAPOOR
03-07-2024

Outline

Executive Summary

- Summary of methodologies
  - Data collection through API
  - Data collection with web scraping
  - Data wrangling
  - Exploratory data analysis (EDA) with SQL
  - EDA with data visualization
  - Interactive visual analytics with Folium
  - Machine learning Predictions
- Summary of all results
  - EDA results
  - Interactive analytics in screenshots
  - Predictive analytics result from Machine learning lab session

Introduction

- Project background and 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. Repeating this process will make the price down even further. As a data scientist of a startup rivaling SpaceX, the goal of this project is to create the machine learning pipeline to predict the landing outcome of the first stage in the future. This project is crucial in identifying the right price to bid against SpaceX for a rocket launch.

- Problems you want to find answers
  - Identifying all factors that influence the landing outcome.
  - The relationship between each variables and how it is affecting the outcome.
  - The best condition needed to increase the probability of successful landing

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using SpaceX REST API and web scrapping from Wikipedia
- Perform data wrangling
  - Data was processed using one-hot encoding for 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

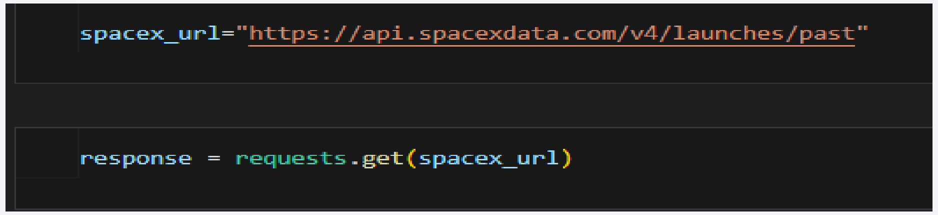
Data Collection

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. As mentioned, the dataset was collected by REST API and Web Scrapping from Wikipedia.

For REST API, its started by using the get request. Then, we decoded the response content as Json and turn it into a pandas dataframe using json_normalize(). We then cleaned the data, checked for missing values and fill with whatever needed.

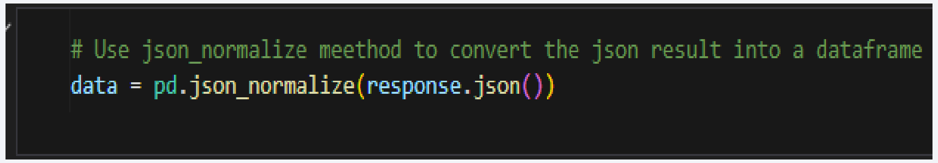
For web scrapping, we will use the BeautifulSoup to extract the launch records as HTML table, parse the table and convert it to a pandas dataframe for further analysis.

Data Collection – SpaceX API

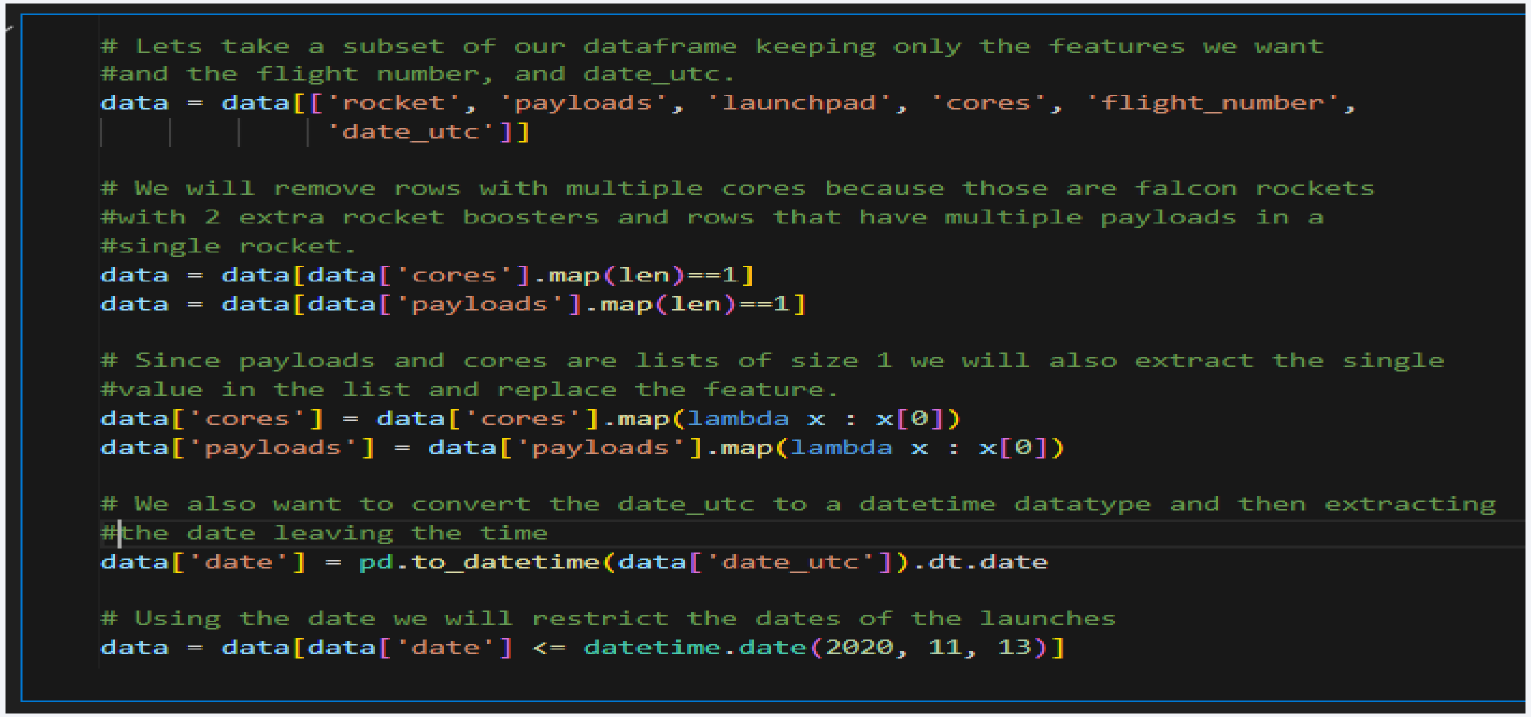


```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

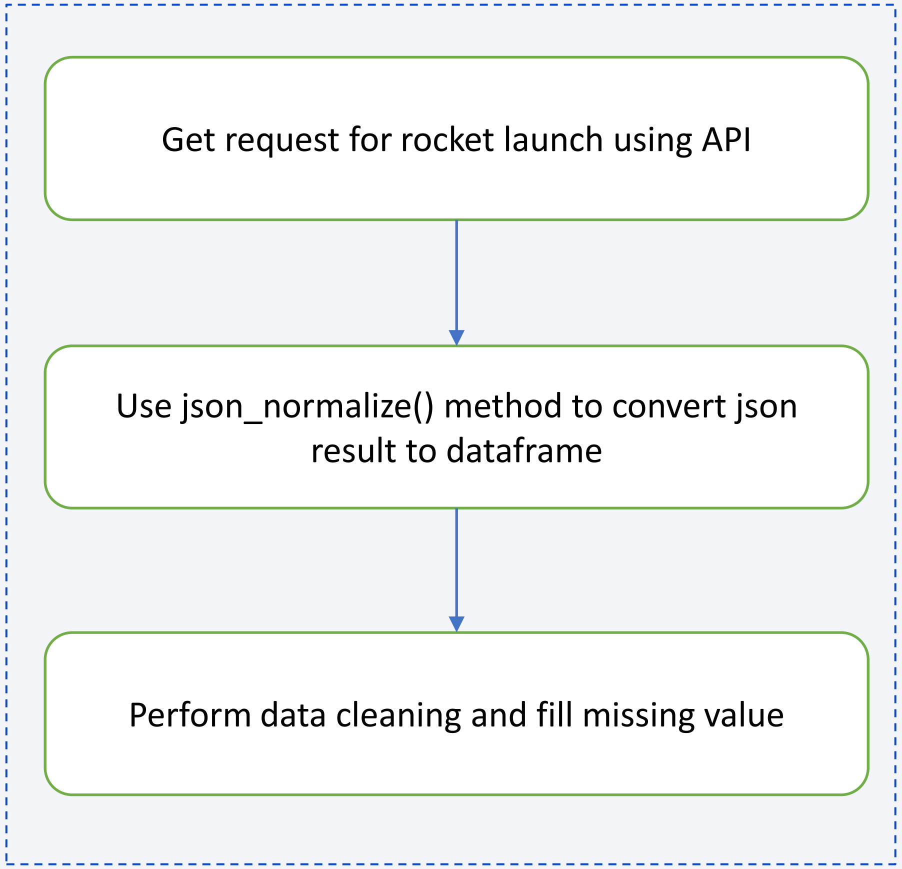
```
response = requests.get(spacex_url)
```



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

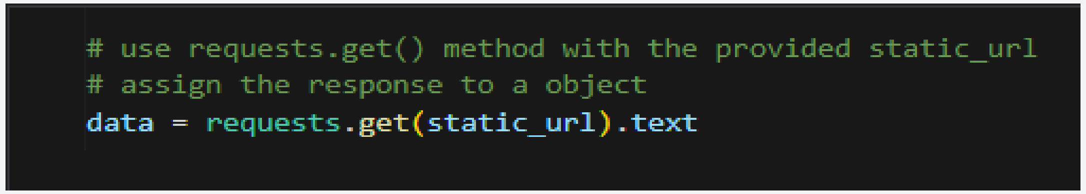


```
# Lets take a subset of our dataframe keeping only the features we want
#and the flight number, and date_utc.
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number',
            'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets
#with 2 extra rocket boosters and rows that have multiple payloads in a
#single rocket.
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 replace the feature.
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)]
```

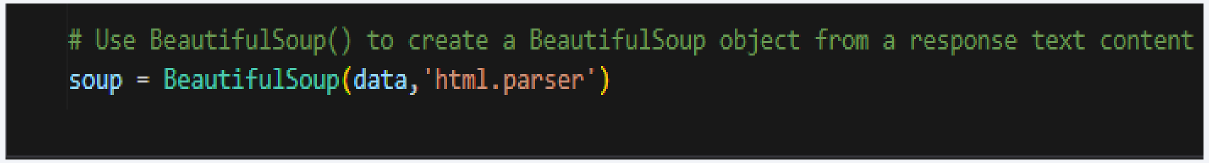

Get request for rocket launch using API
Use json_normalize() method to convert json result to dataframe
Perform data cleaning and fill missing value

Github link: https://github.com/samarth003/ibm-data-science-repo/blob/main/Python-Capstone/Data-collection-and-wrangling/jupyter-labs-spacex-data-collection-api.ipynb

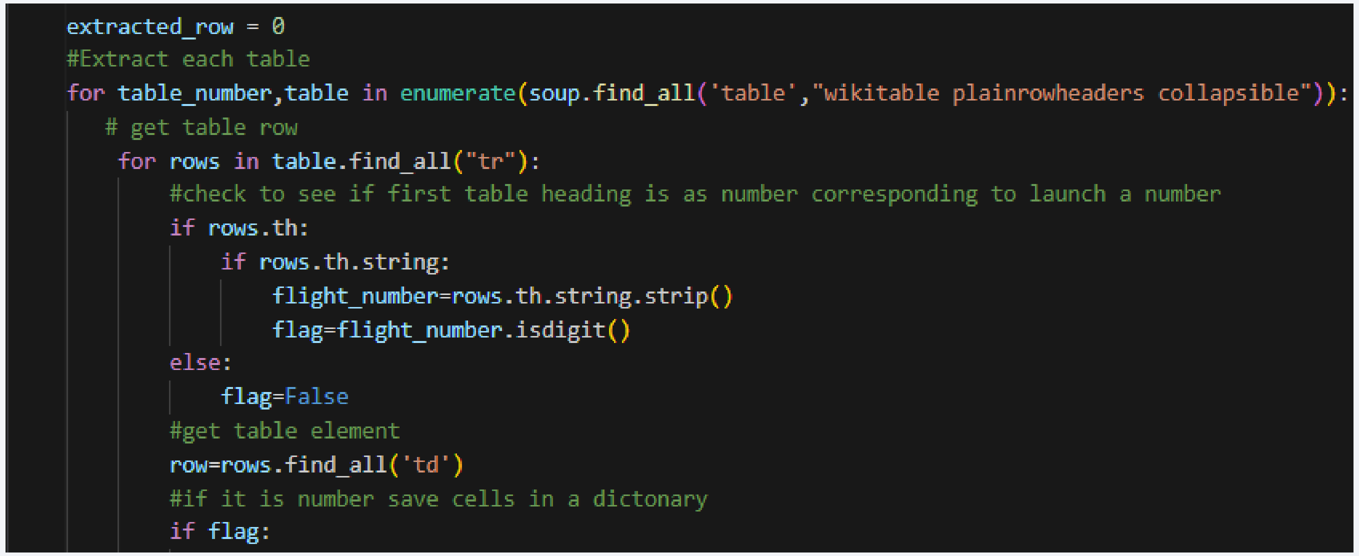
Data Collection - Scraping



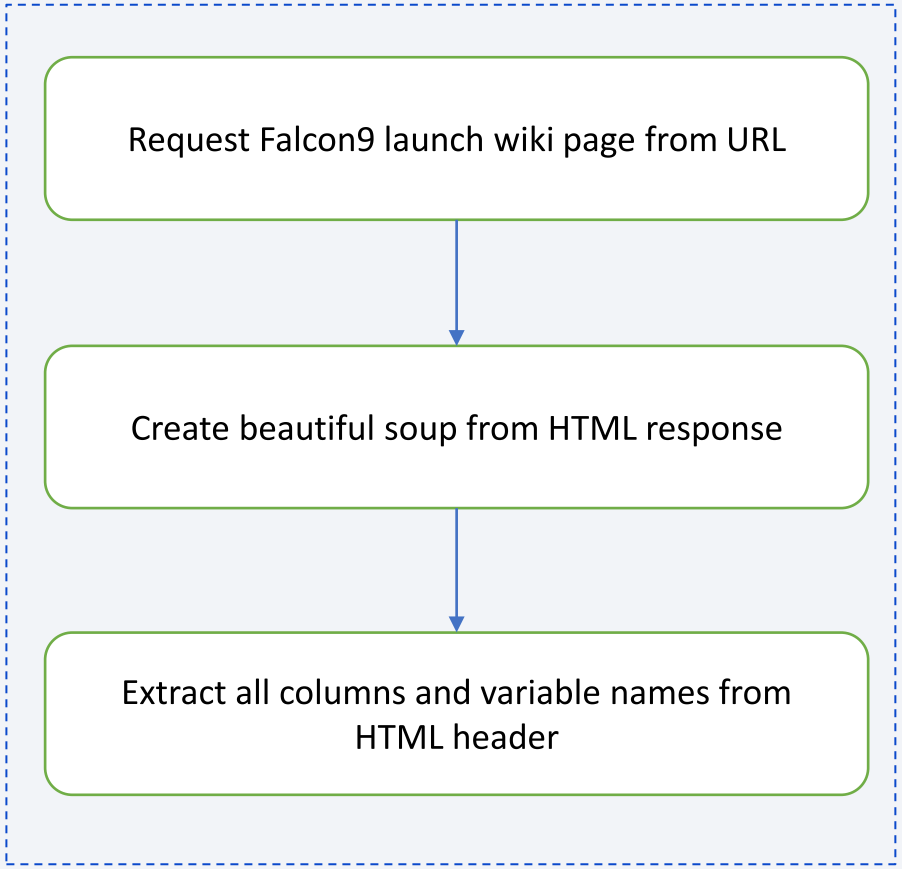
```
# use requests.get() method with the provided static_url
# assign the response to a object
data = requests.get(static_url).text
```



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



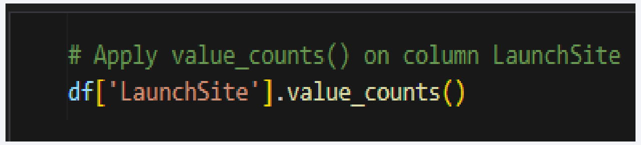
```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
```



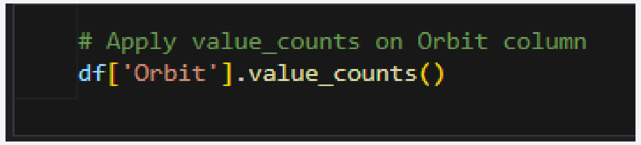
$$\begin{tikzcd}
\text{Request Falcon9 launch wiki page from URL} \arrow[d] \\
\text{Create beautiful soup from HTML response} \arrow[d] \\
\text{Extract all columns and variable names from HTML header}
\end{tikzcd}$$

Github link: https://github.com/samarth003/ibm-data-science-repo/blob/main/Python-Capstone/Data-collection-and-wrangling/jupyter-labs-spacex-data-collection-api.ipynb

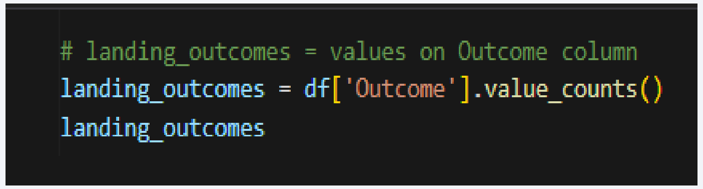
Data Wrangling



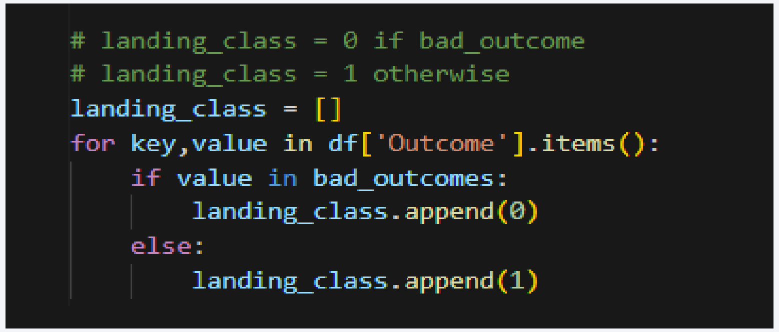
```
# Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()
```



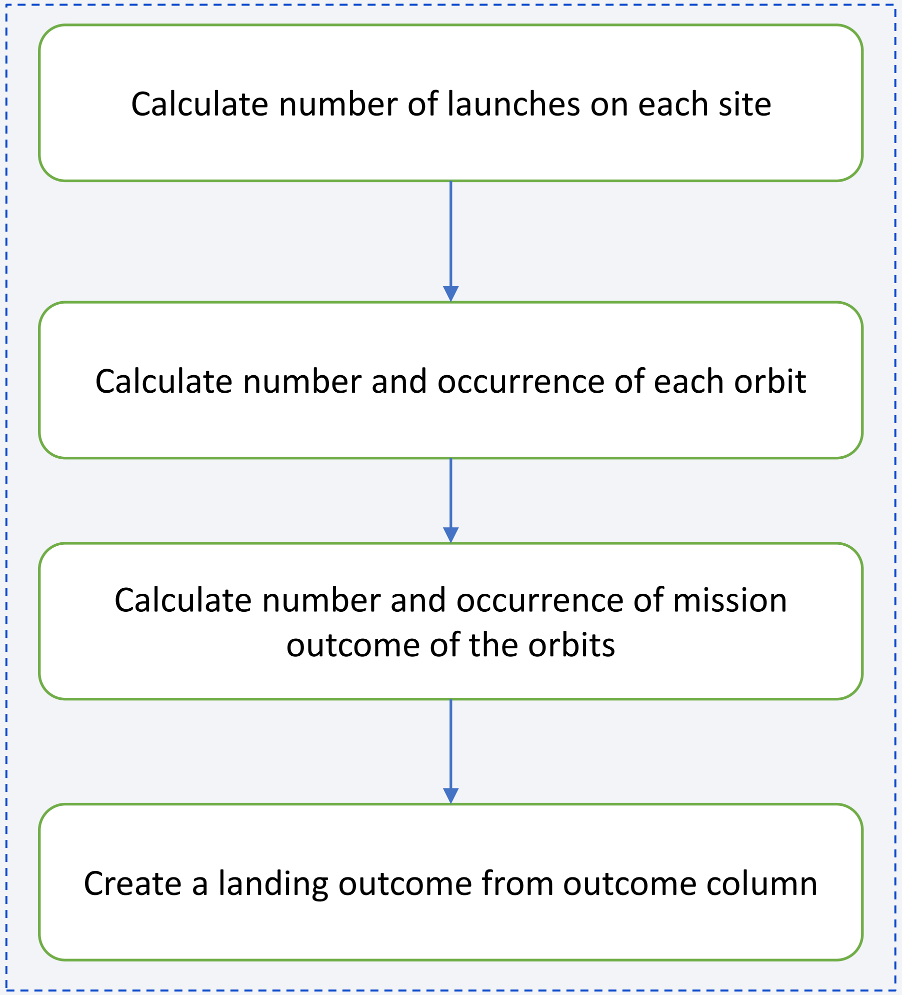
```
# Apply value_counts on Orbit column
df['Orbit'].value_counts()
```



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

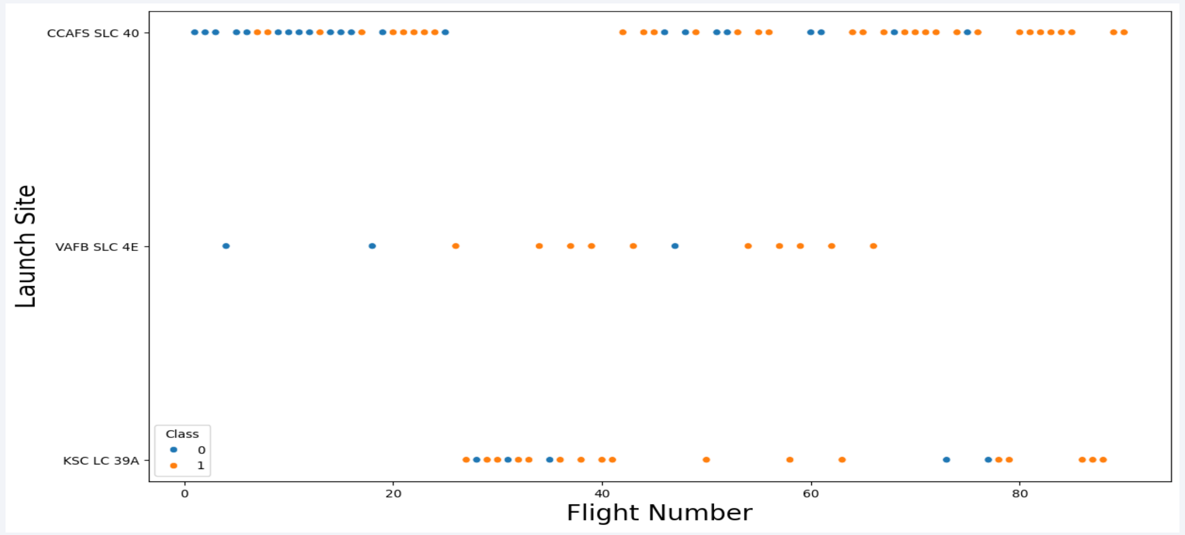


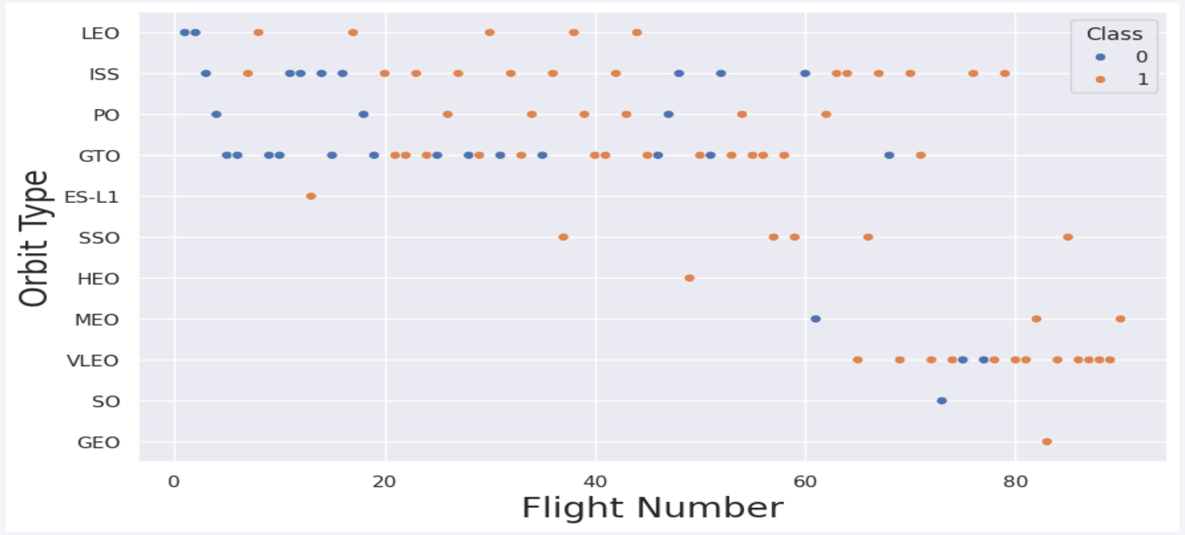
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for key,value in df['Outcome'].items():
    if value in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```


Calculate number of launches on each site
Calculate number and occurrence of each orbit
Calculate number and occurrence of mission outcome of the orbits
Create a landing outcome from outcome column

Github link: https://github.com/samarth003/ibm-data-science-repo/blob/main/Python-Capstone/Data-collection-and-wrangling/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization





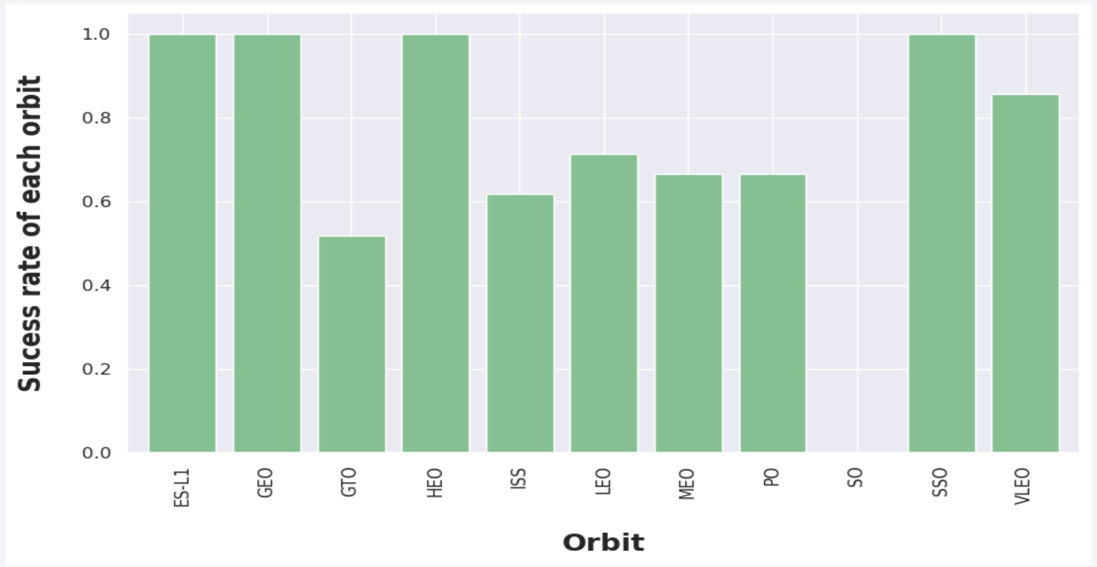
We first started by using scatter graph to find the relationship between the attributes such as between:

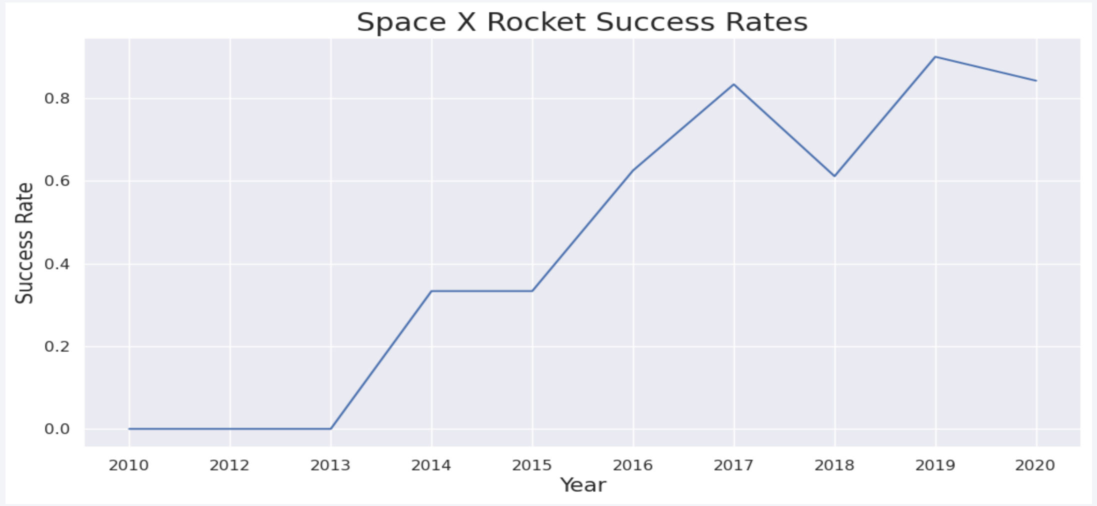
- Payload and Flight Number.
- Flight Number and Launch Site.
- Payload and Launch Site.
- Flight Number and Orbit Type.
- Payload and Orbit Type

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.

Github link: https://github.com/samarth003/ibm-data-science-repo/blob/main/Python-Capstone/EDA/edadataviz.ipynb

EDA with Data Visualization





Once we get a hint of the relationships using scatter plot, we will then use further visualization tools such as bar graph and line plots graph for further analysis.

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.

We then use the line graph to show a trends or pattern of the attribute over time which in this case, is used for see the launch success yearly trend.

We then use Feature Engineering to be used in success prediction in the future module by created the dummy variables to categorical columns.

Github link: https://github.com/samarth003/ibm-data-science-repo/blob/main/Python-Capstone/EDA/edadataviz.ipynb

EDA with SQL

Using SQL, we had performed many queries to get better understanding of the dataset, Eg:

- 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.

Github link: https://github.com/samarth003/ibm-data-science-repo/blob/main/Python-Capstone/EDA/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

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().

We then used the Haversine's formula 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?

Github link: https://github.com/samarth003/ibm-data-science-repo/blob/main/Python-Capstone/Interactive%20dashboards/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly dash which allowing the user to play around with the data as they need.
- We plotted pie charts showing the total launches by a certain sites.
- We then plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.

Github link: https://github.com/samarth003/ibm-data-science-repo/blob/main/Python-Capstone/Interactive%20dashboards/spacex_dash_app.py

Predictive Analysis (Classification)

Building the Model

- Load dataset into pandas and numpy
- Transform data and then split it into training and test data sets
- Decide which type of ML to apply
- Set parameters and algorithms to GridSearchCV and fit the data set

Evaluating the Model

- Check accuracy for each model
- Get tuned hyper-parameters 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 is the best performing model

Github link: https://github.com/samarth003/ibm-data-science-repo/blob/main/Python-Capstone/Confusion%20Matrices/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Results

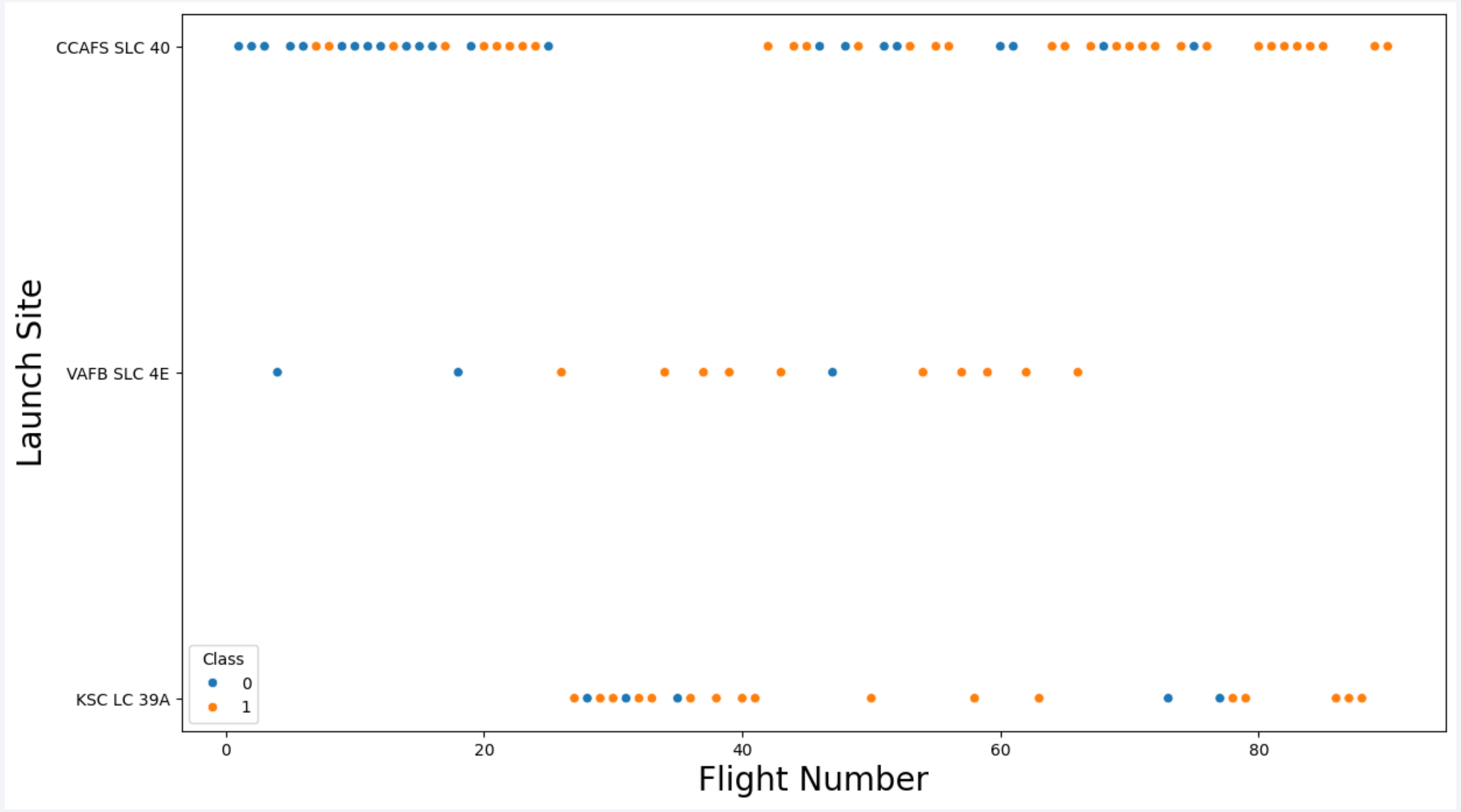
The results will be categorized to 3 main results:

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

Section 2

Insights drawn from EDA

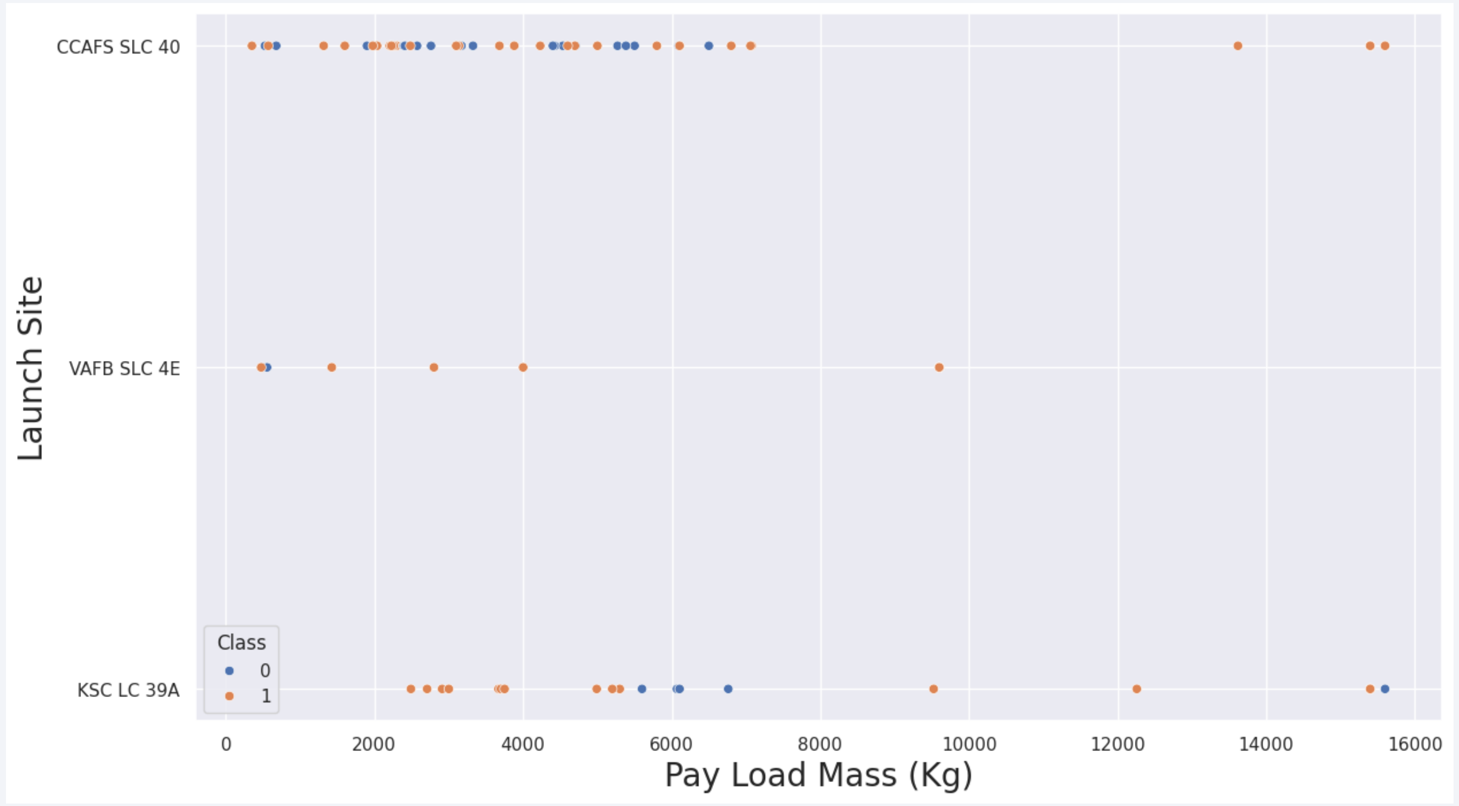
Flight Number vs. Launch Site



This scatter plot shows that the larger the flights amount of the launch site, the greater the success rate will be.

However, site CCAFS SLC40 shows the least pattern of this.

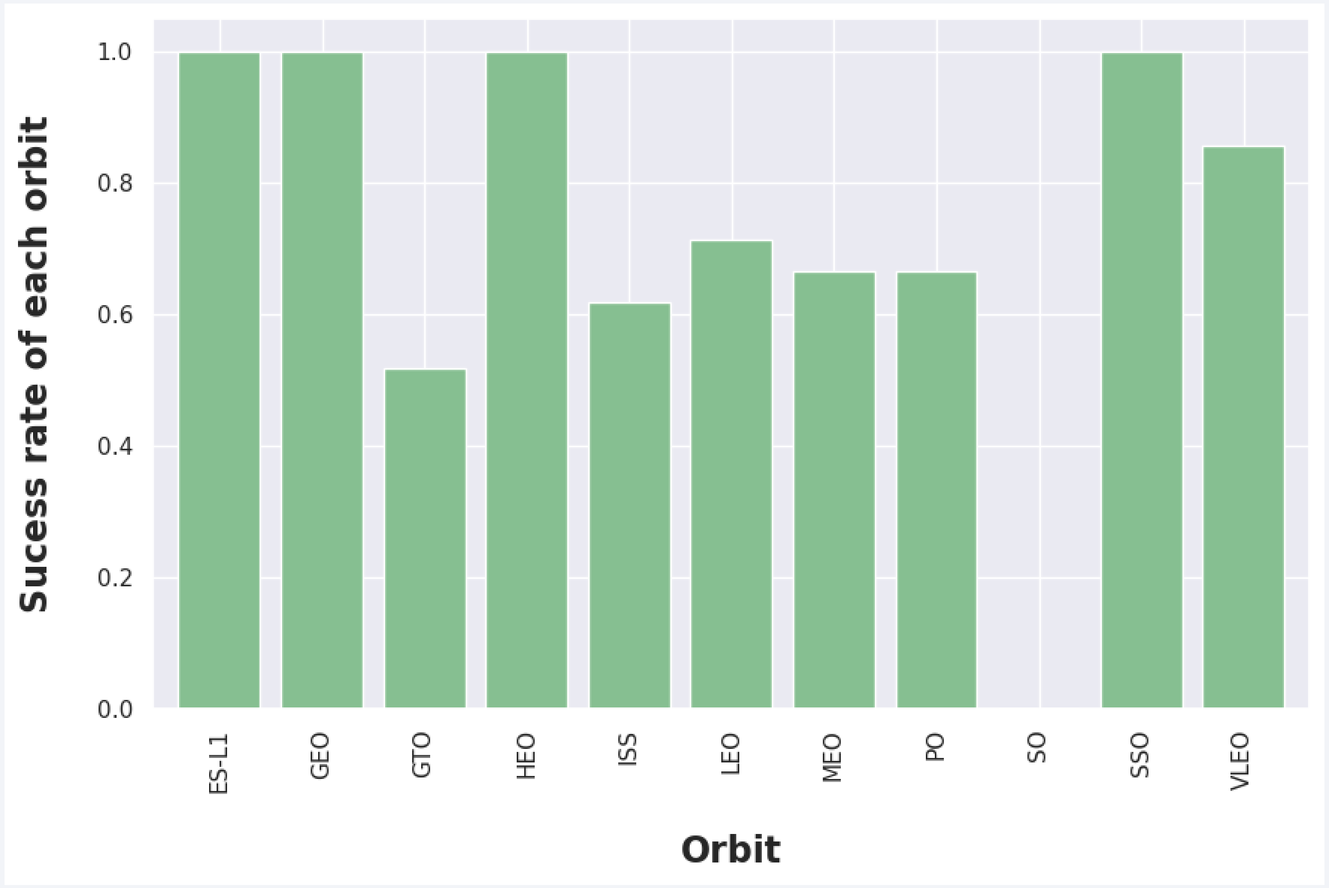
Payload vs. Launch Site



This scatter plot shows once the pay load mass is greater than 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.

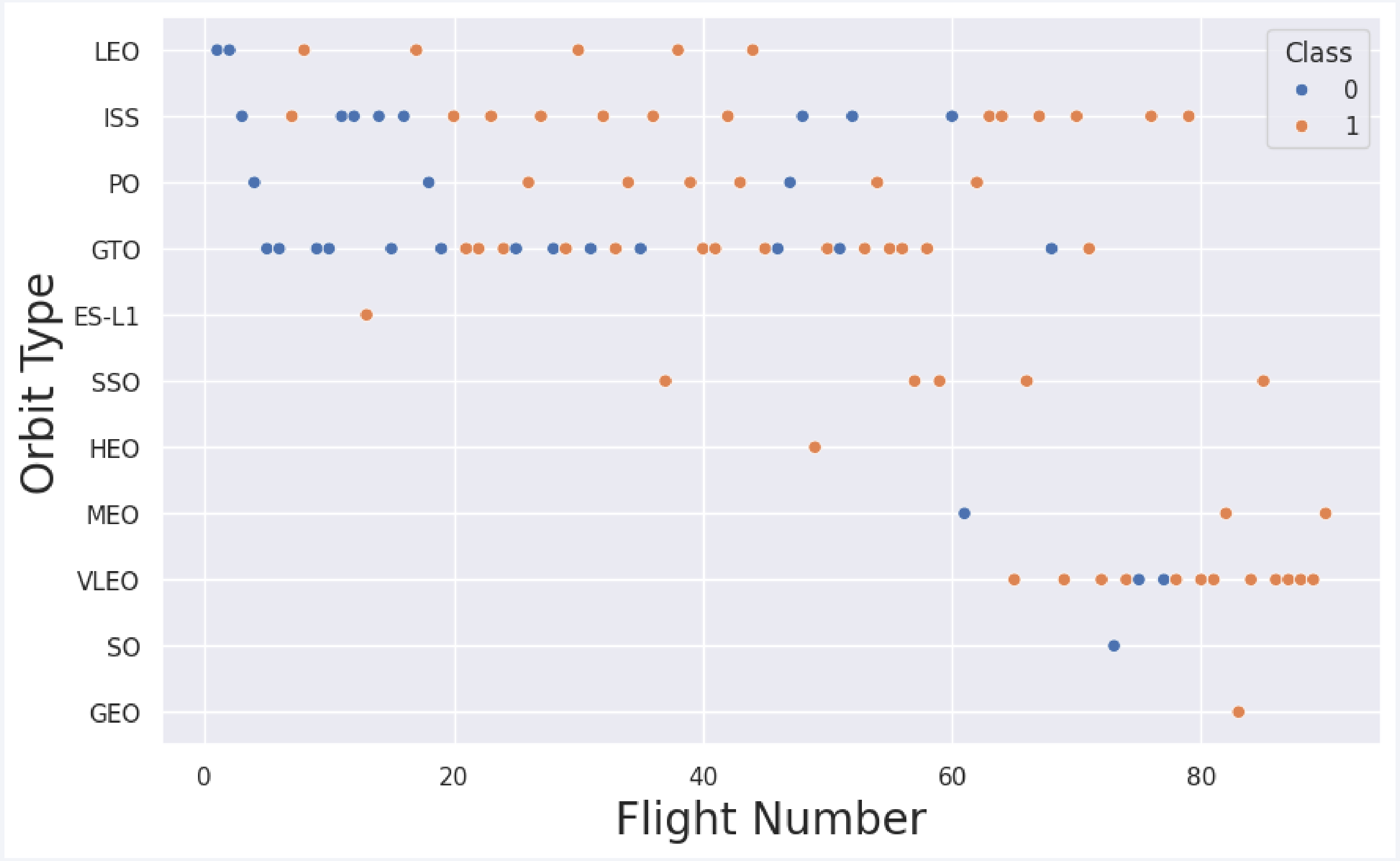
Success Rate vs. Orbit Type



This figure depicted the possibility of the orbits to influences 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.

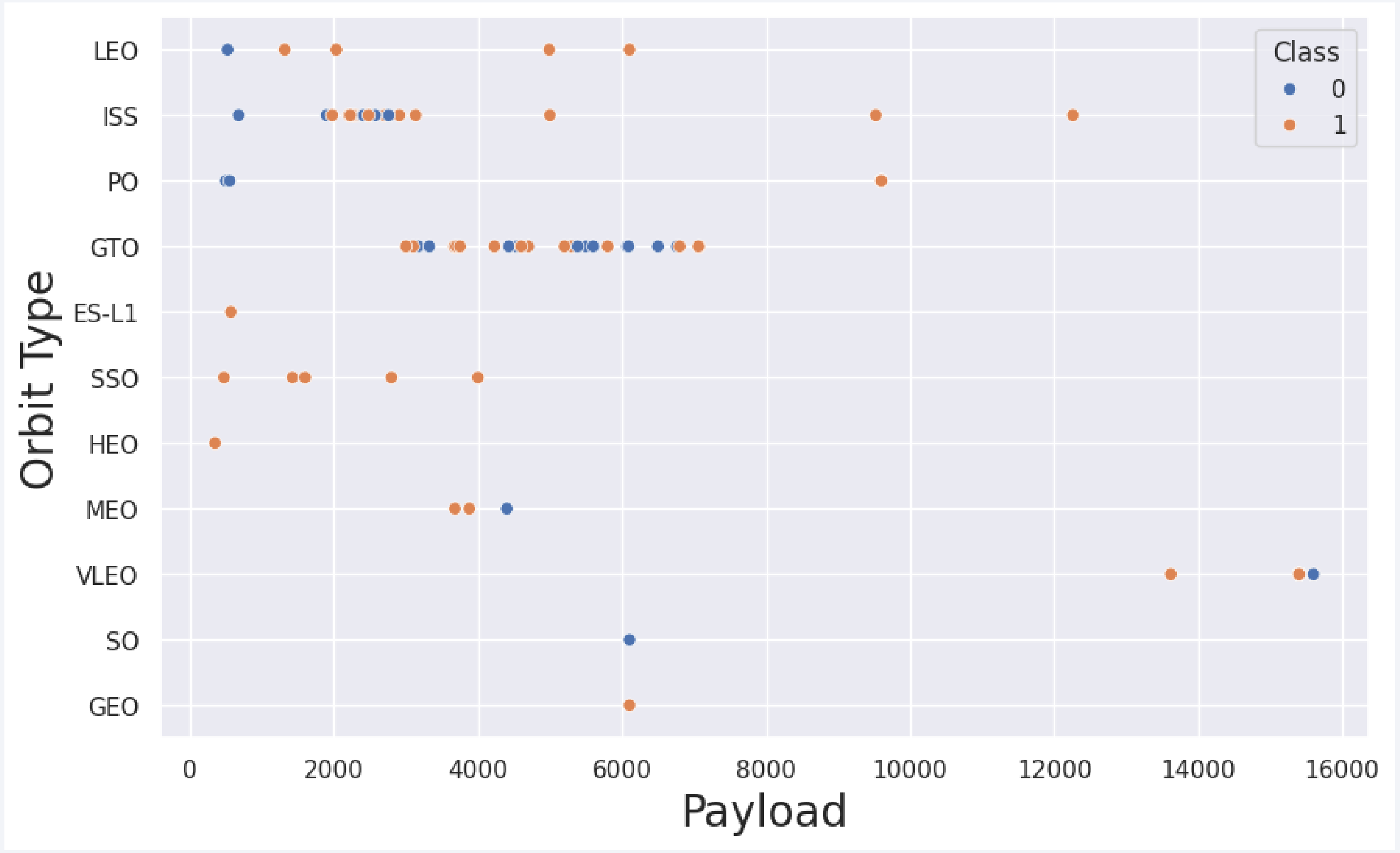
Flight Number vs. Orbit Type



This scatter plot shows that generally, the larger 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

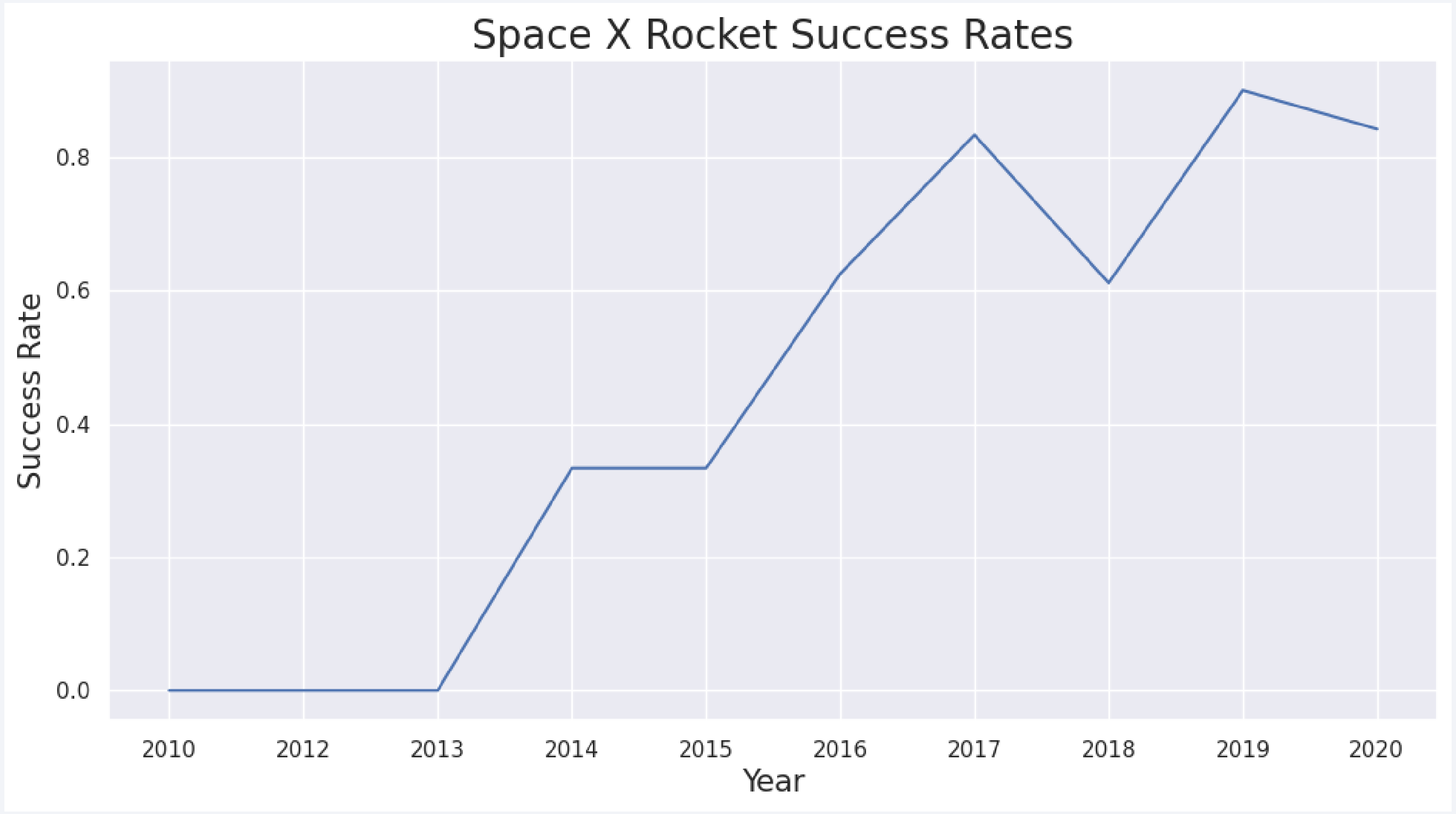


Heavier payload has positive impact on LEO, ISS and P0 orbit. However, it has negative impact on MEO and VLEO orbit.

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

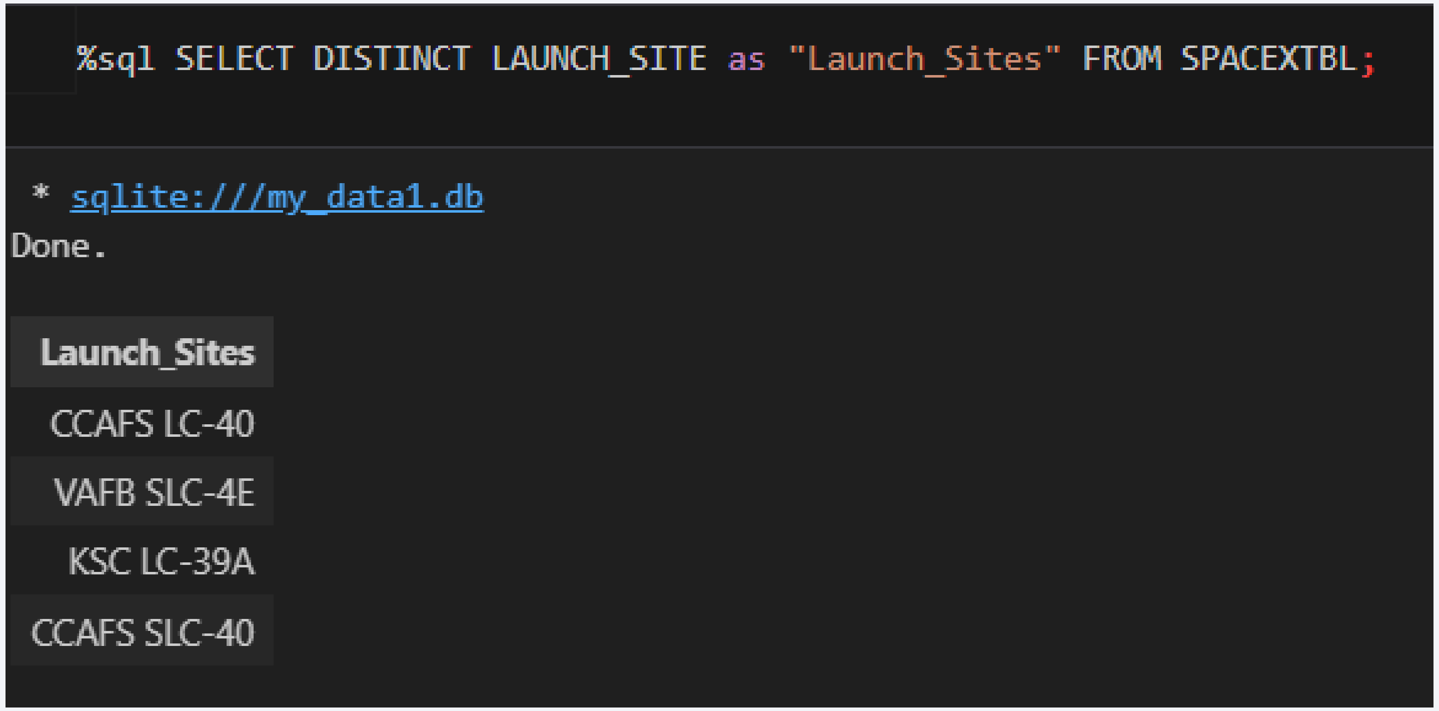


This figures clearly depicted and increasing trend from the year 2013 until 2020.

If this trend continue for the next year onward. The 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 data.



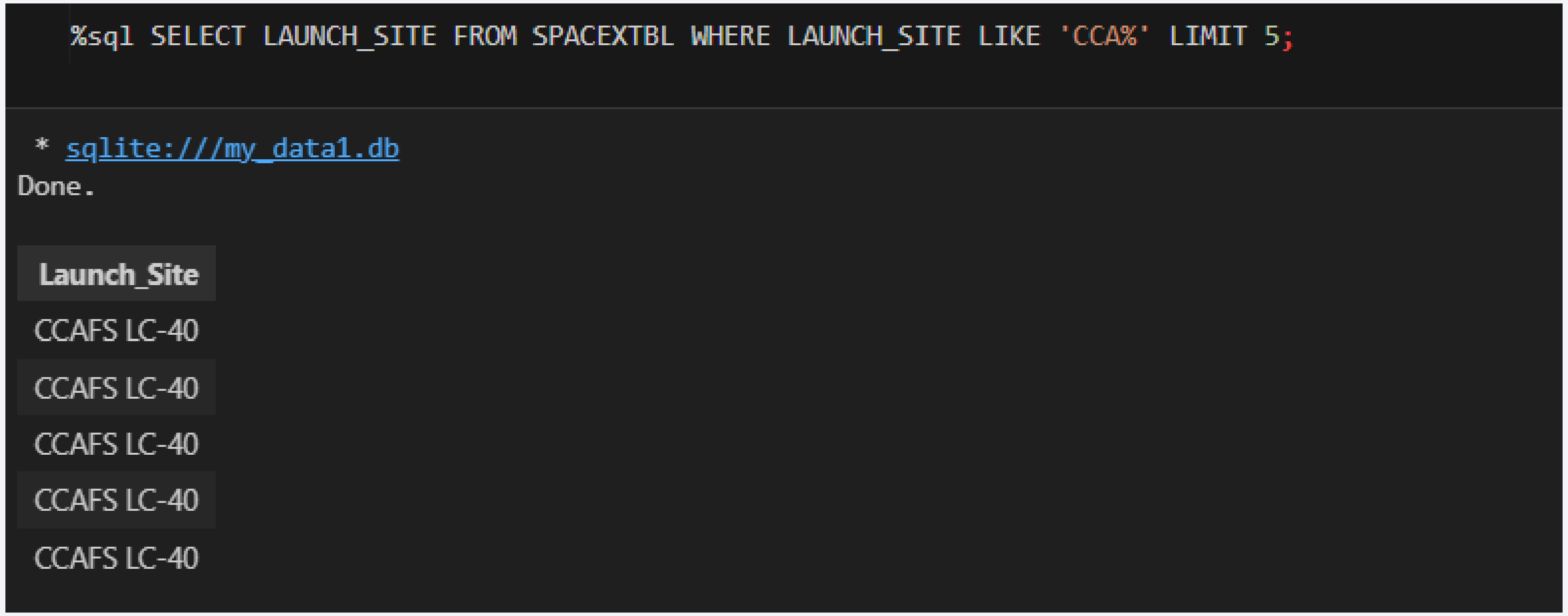
```
%sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTBL;
```

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

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`



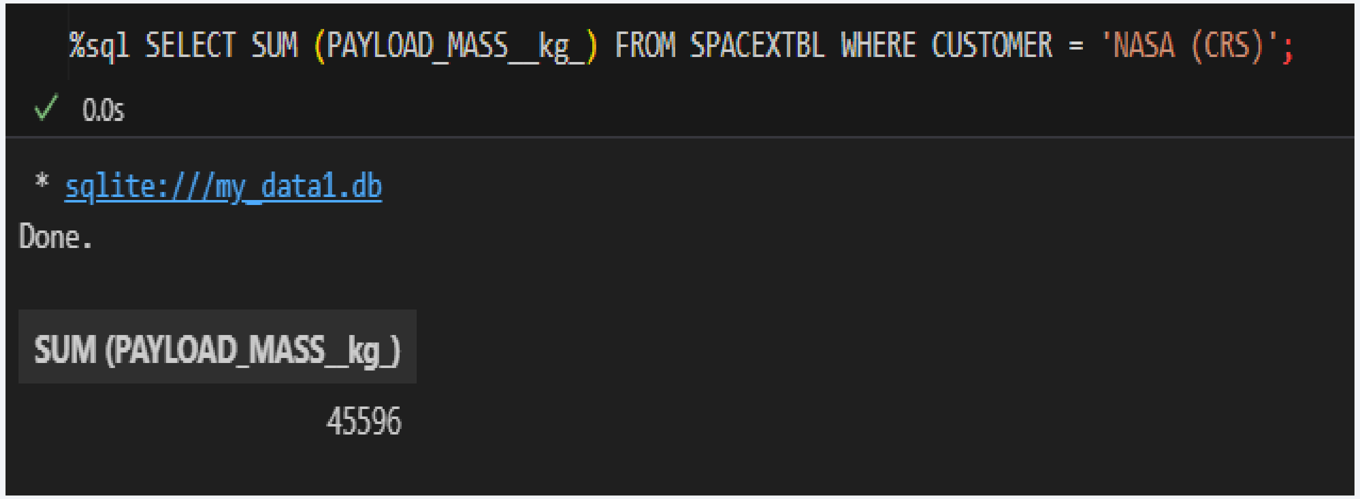
```
%sql SELECT LAUNCH_SITE FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
```

```
 * 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

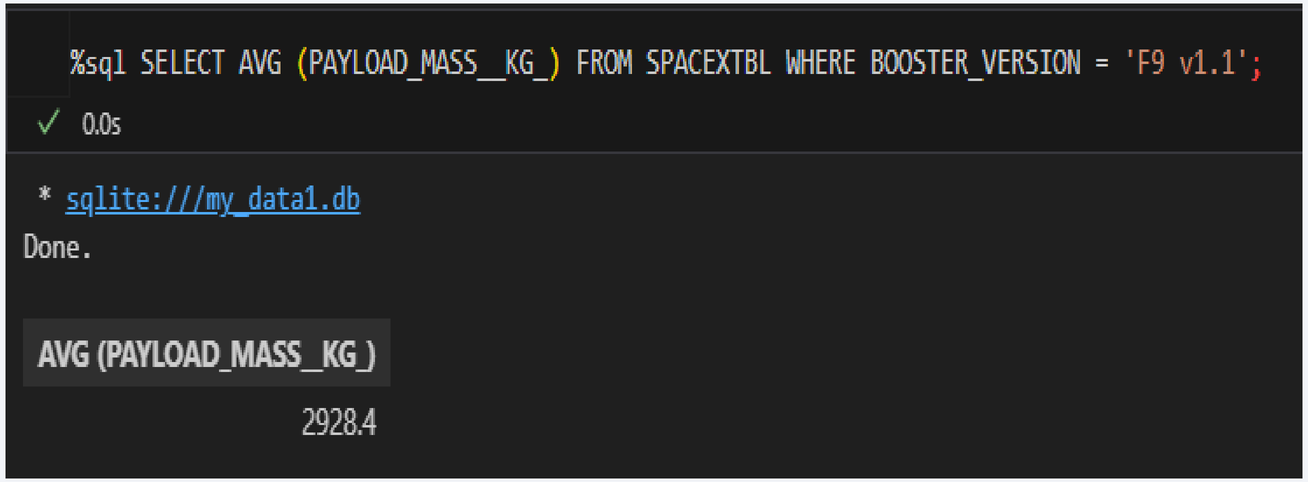
We calculated the total payload carried by boosters from NASA as 45596 using the query below



```
%sql SELECT SUM (PAYLOAD_MASS__kg_) FROM SPACEXTBL WHERE CUSTOMER = 'NASA (CRS)';
✓ 0.0s

 * sqlite:///my_data1.db
Done.

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

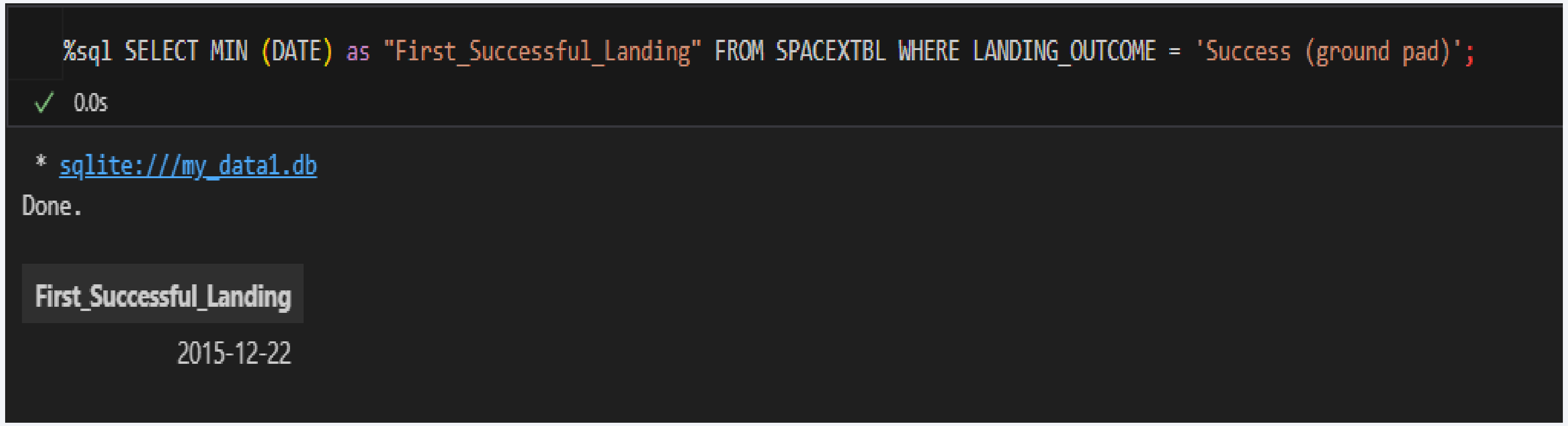


```
%sql SELECT AVG (PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE BOOSTER_VERSION = 'F9 v1.1';
✓ 0.0s

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

AVG (PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

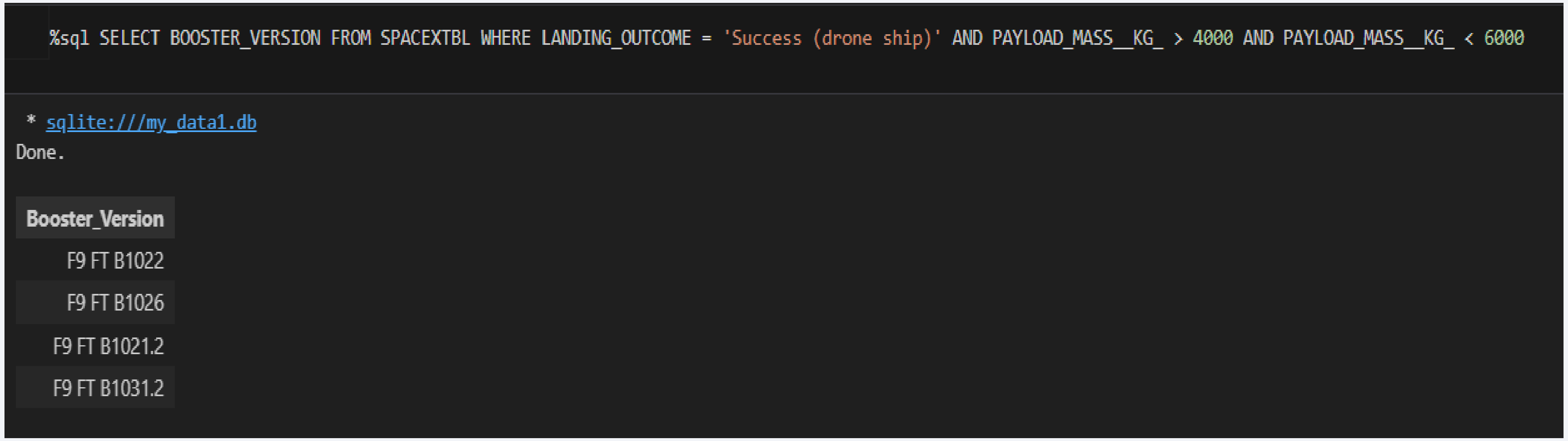
We use the min() function to find the result We observed that the dates of the first successful landing outcome on ground pad was 22nd December 2015



```
%sql SELECT MIN (DATE) as "First_Successful_Landing" FROM SPACEXTBL WHERE LANDING_OUTCOME = 'Success (ground pad)';
✓ 0.0s

 * sqlite:///my_data1.db
Done.

First_Successful_Landing
              2015-12-22
```

Successful Drone Ship Landing with Payload between 4000 and 6000

We used the WHERE clause 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



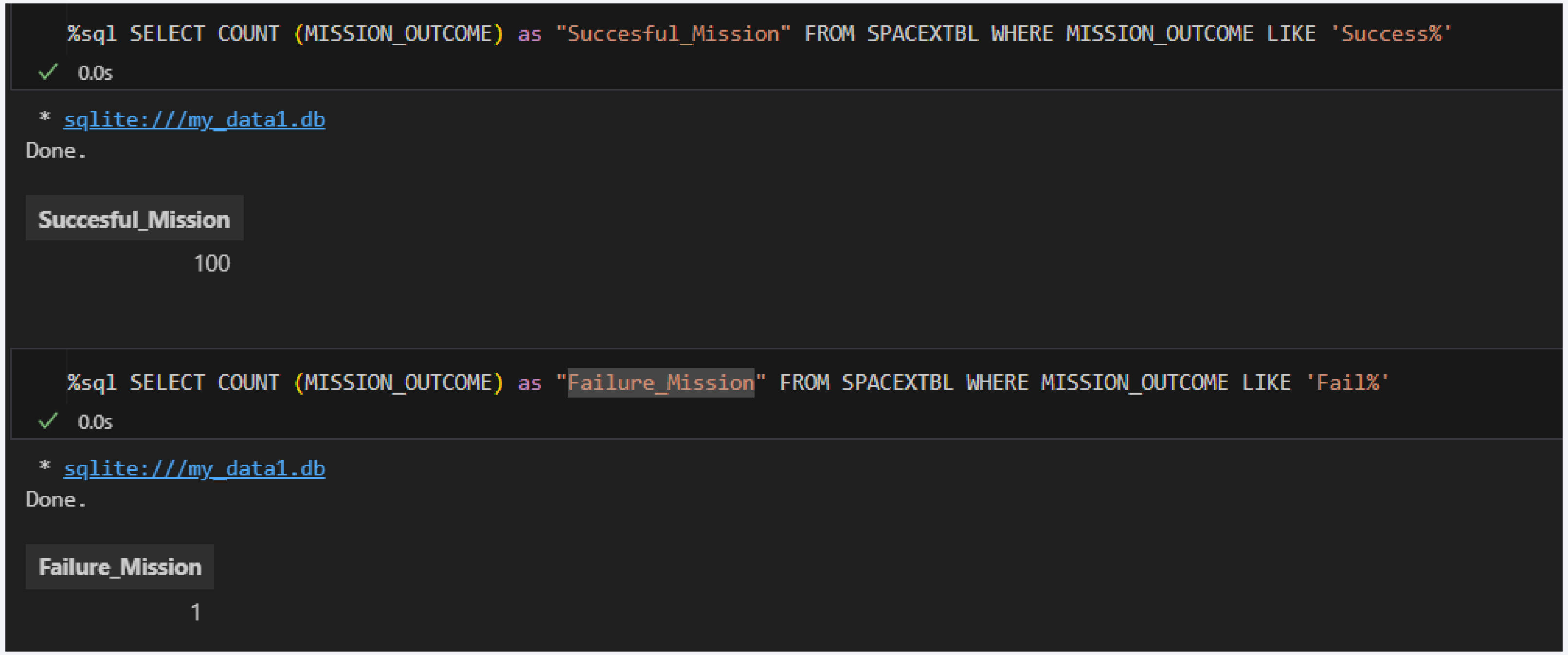
```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE LANDING_OUTCOME = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000
```

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

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 MissionOutcome was a success or a failure.

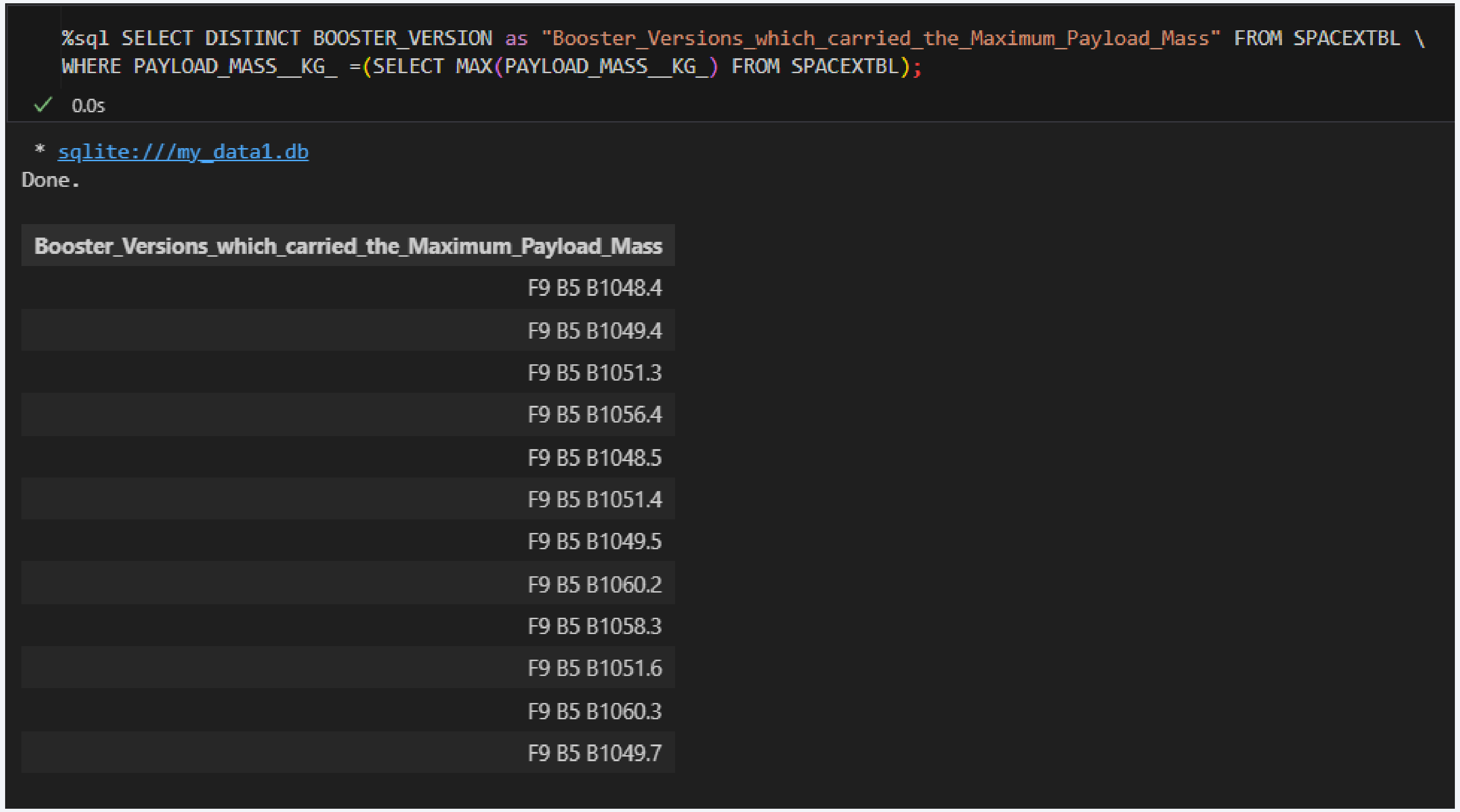


```
%sql SELECT COUNT (MISSION_OUTCOME) as "Succesful_Mission" FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE 'Success%'
✓ 0.0s

 * sqlite:///my_data1.db
Done.

Succesful_Mission
              100
```

```
%sql SELECT COUNT (MISSION_OUTCOME) as "Failure_Mission" FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE 'Fail%'
✓ 0.0s

 * sqlite:///my_data1.db
Done.

Failure_Mission
            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 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);
```

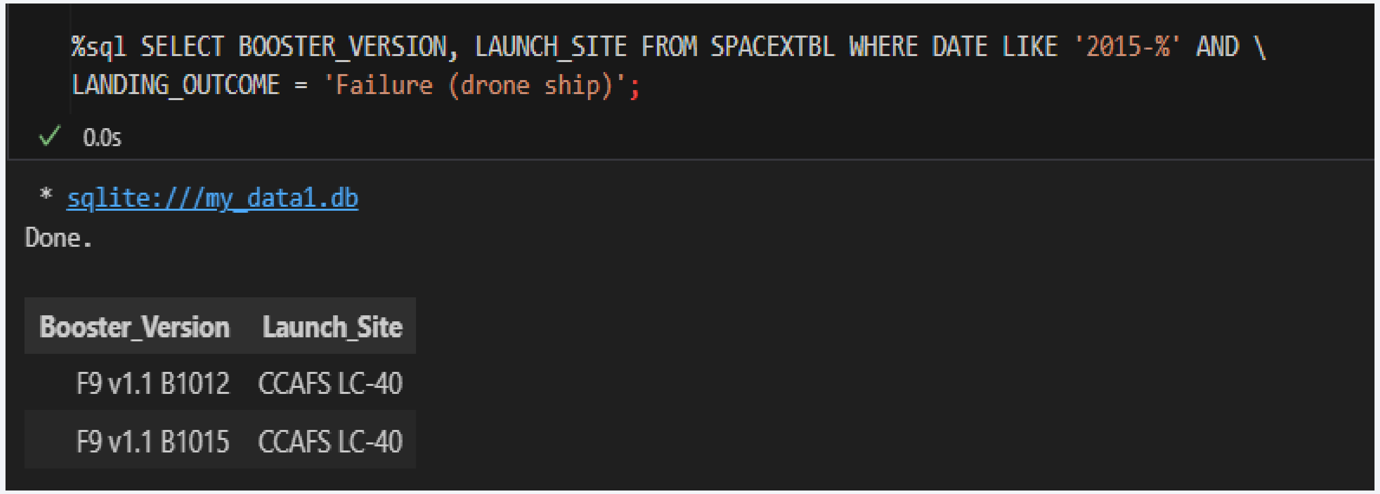
✓ 0.0s

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

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, LIKE, AND, and BETWEEN conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015



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

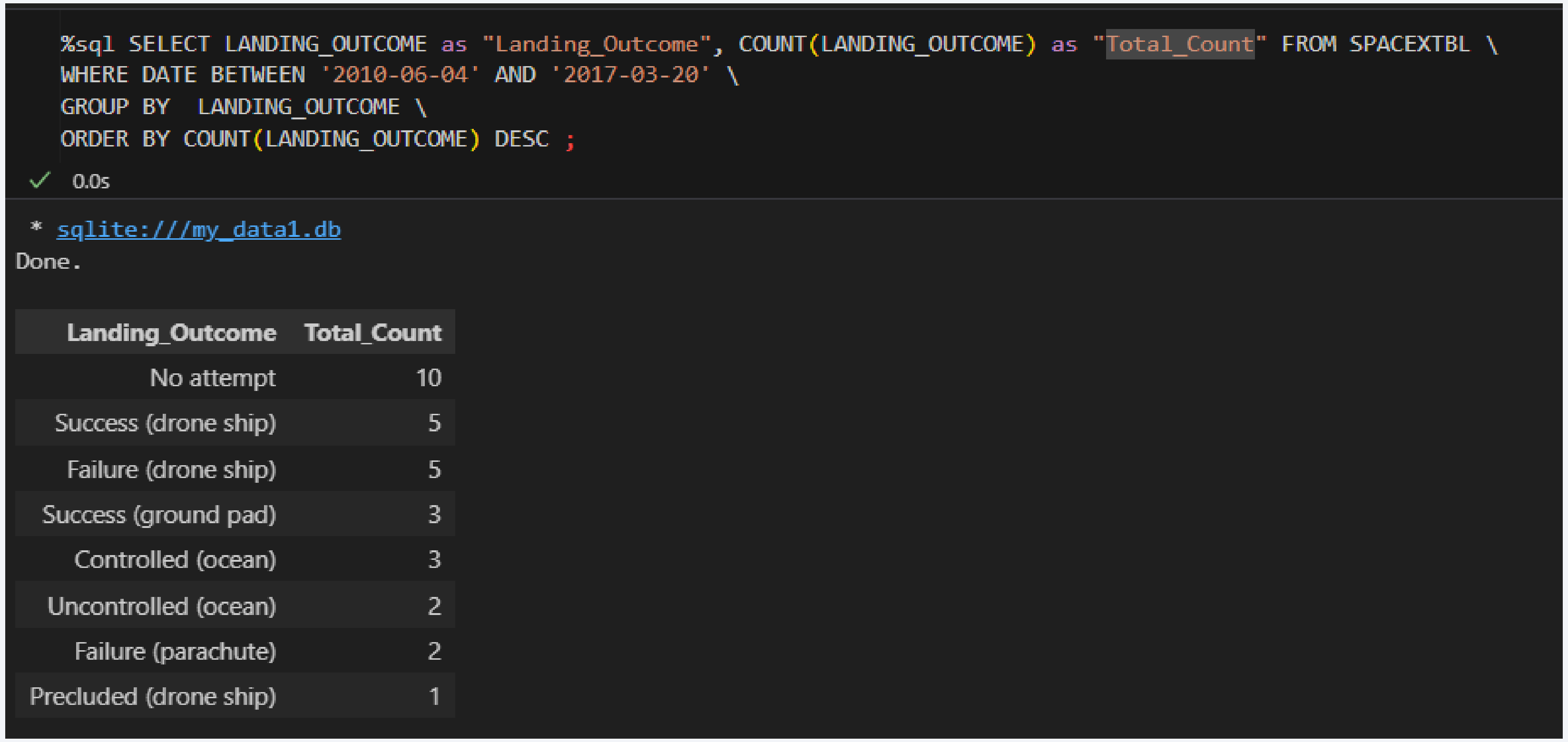
✓ 0.0s

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

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

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 2010-03-20. 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.



```
%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 ;
```

✓ 0.0s

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

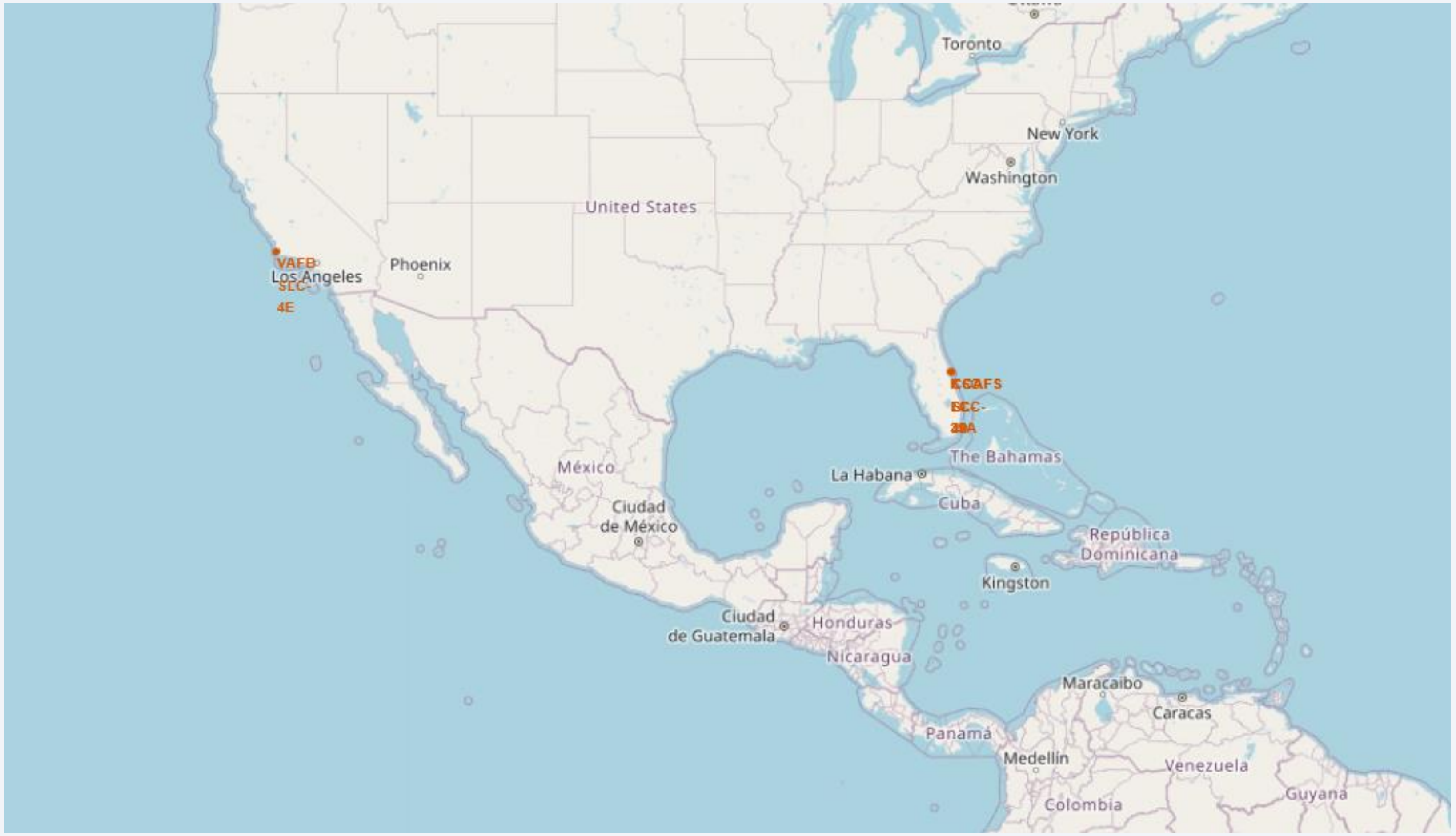
Landing_Outcome	Total_Count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

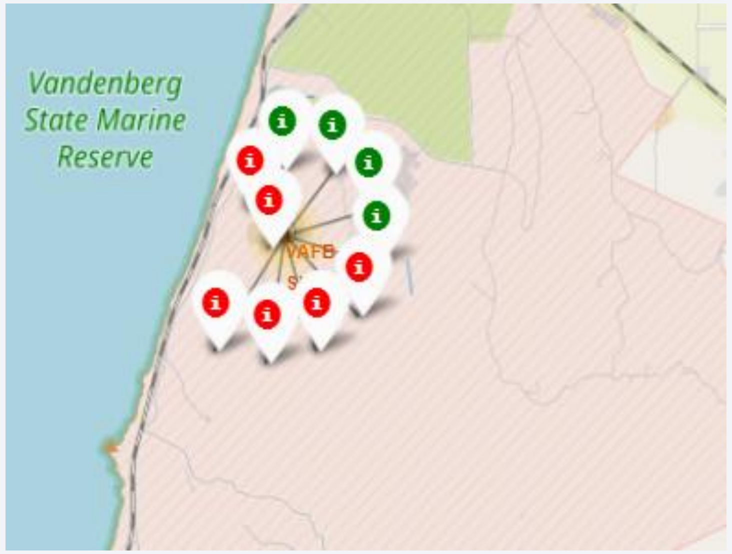
Launch Sites Proximities Analysis

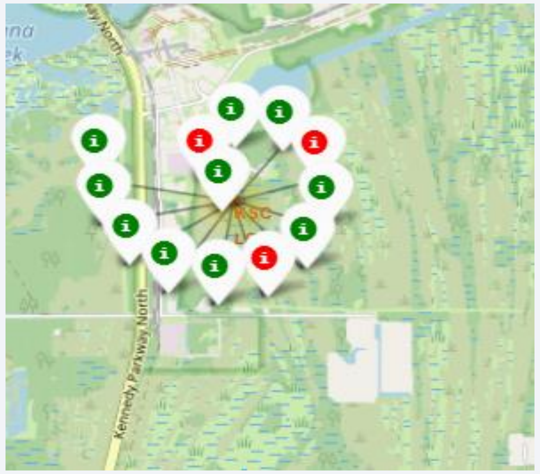
Launch Sites Locations

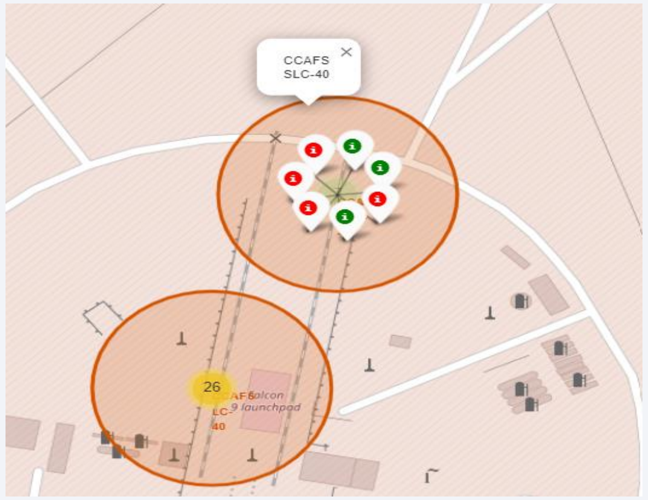
We can see below that all the SpaceX launch sites are located inside the United States

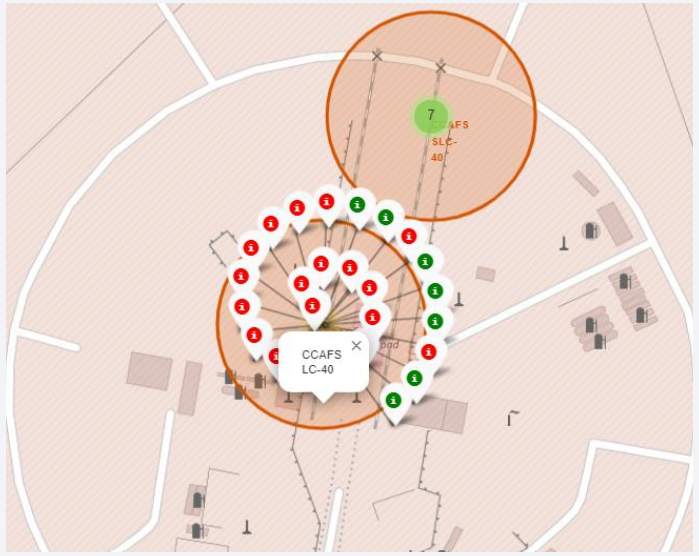


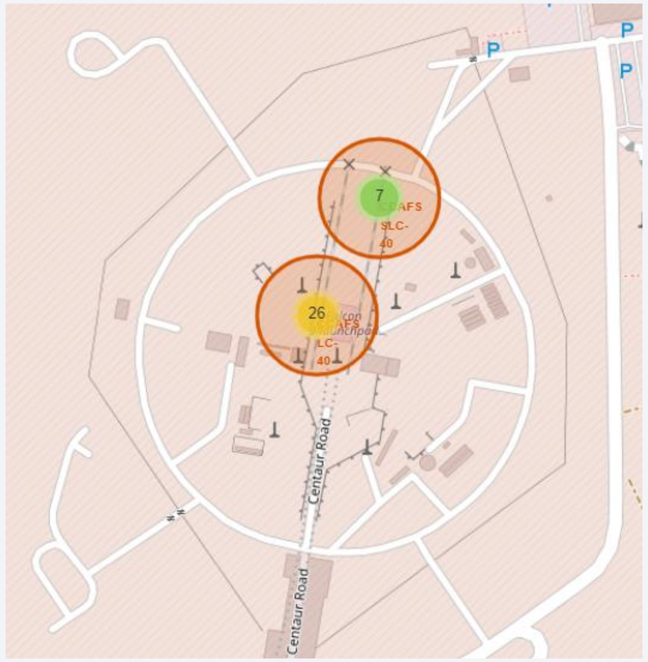
Launch sites markers with color labels





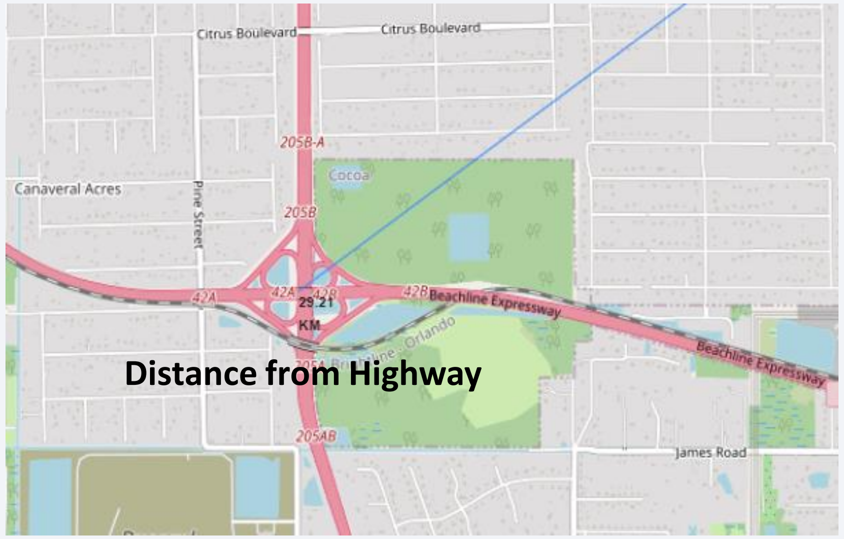


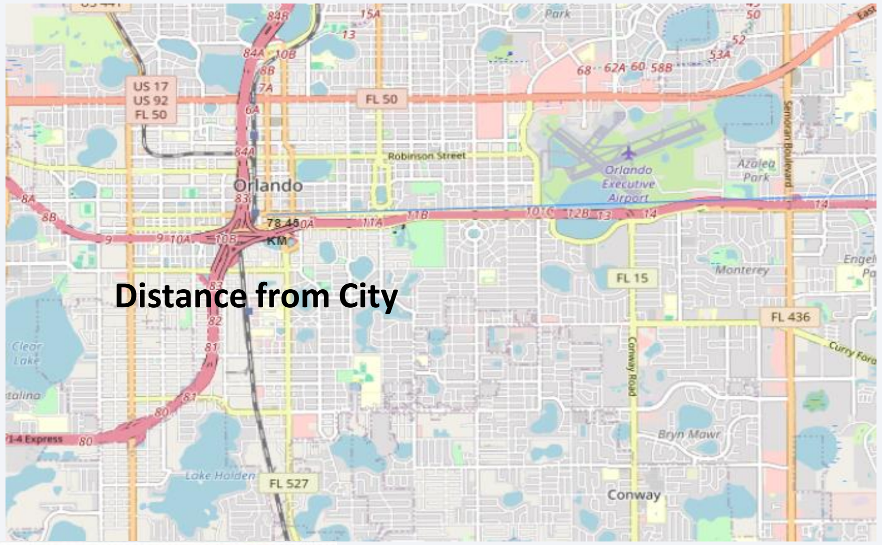


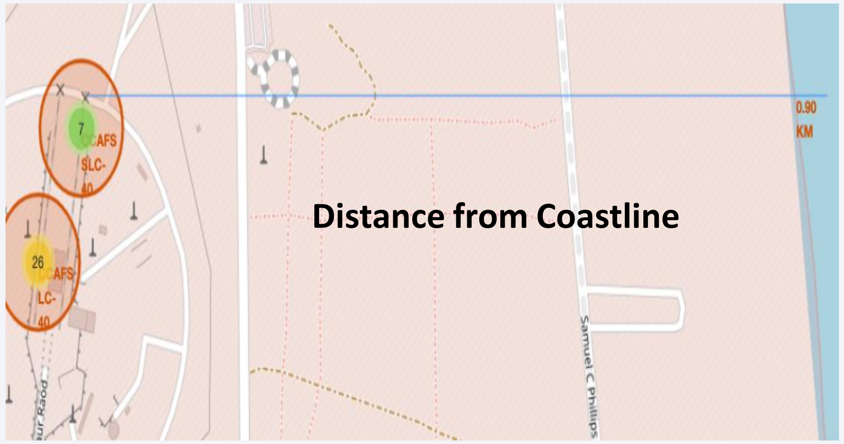


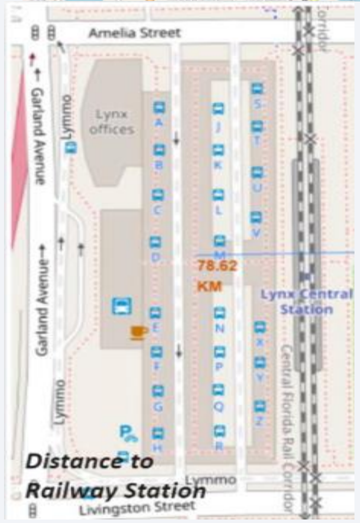
Green marker shows successful launches while Red
Marker shows Failures

Launch sites distance from landmarks









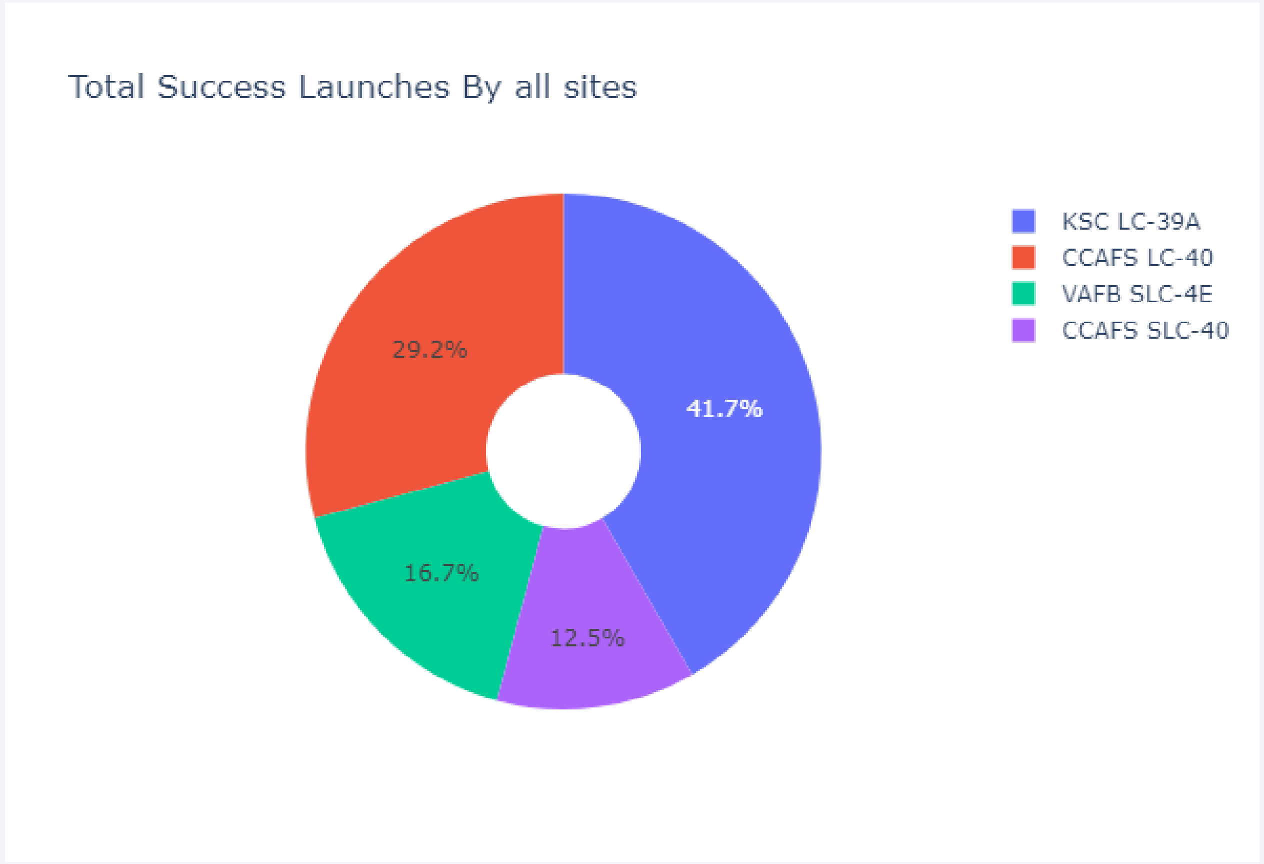
- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

Section 4

Build a Dashboard with Plotly Dash

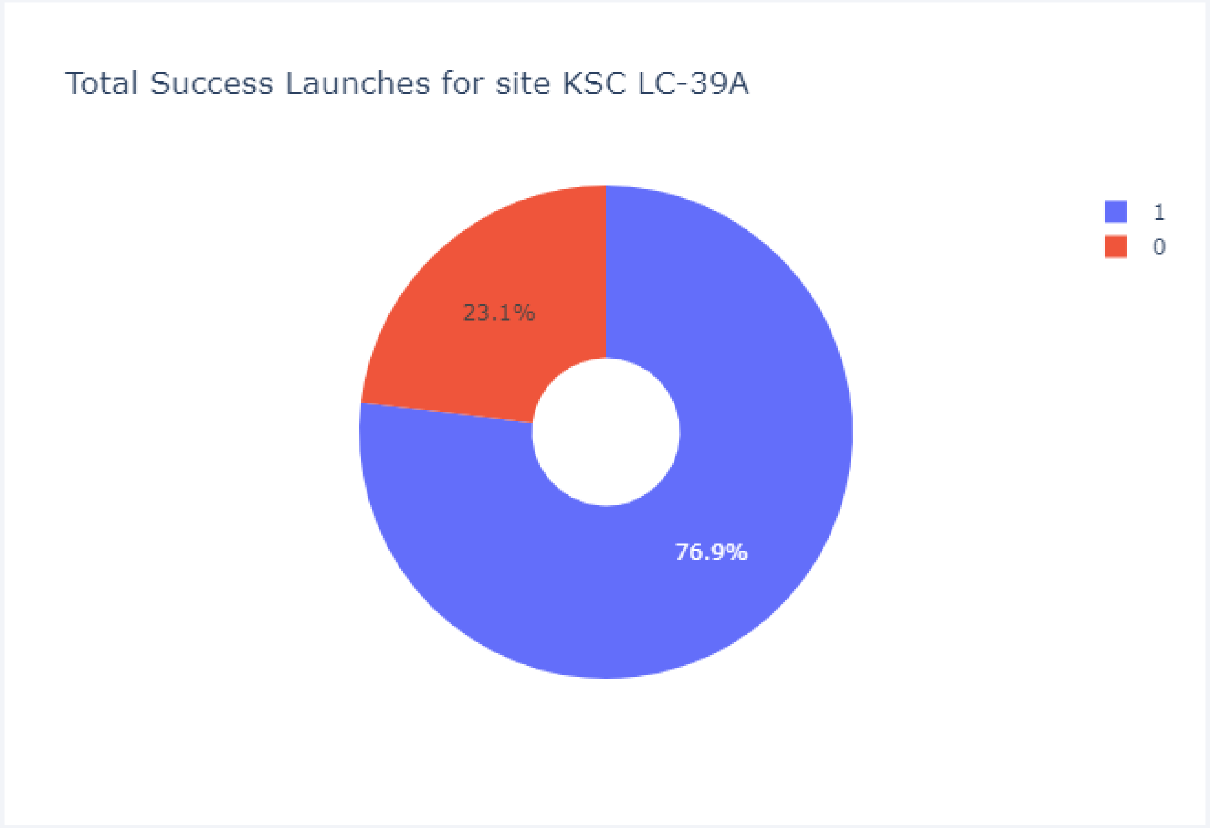
Success launches from each site

KSC LC-39A had the most successful launches from all the sites



Total success launches for KSC LC-39A

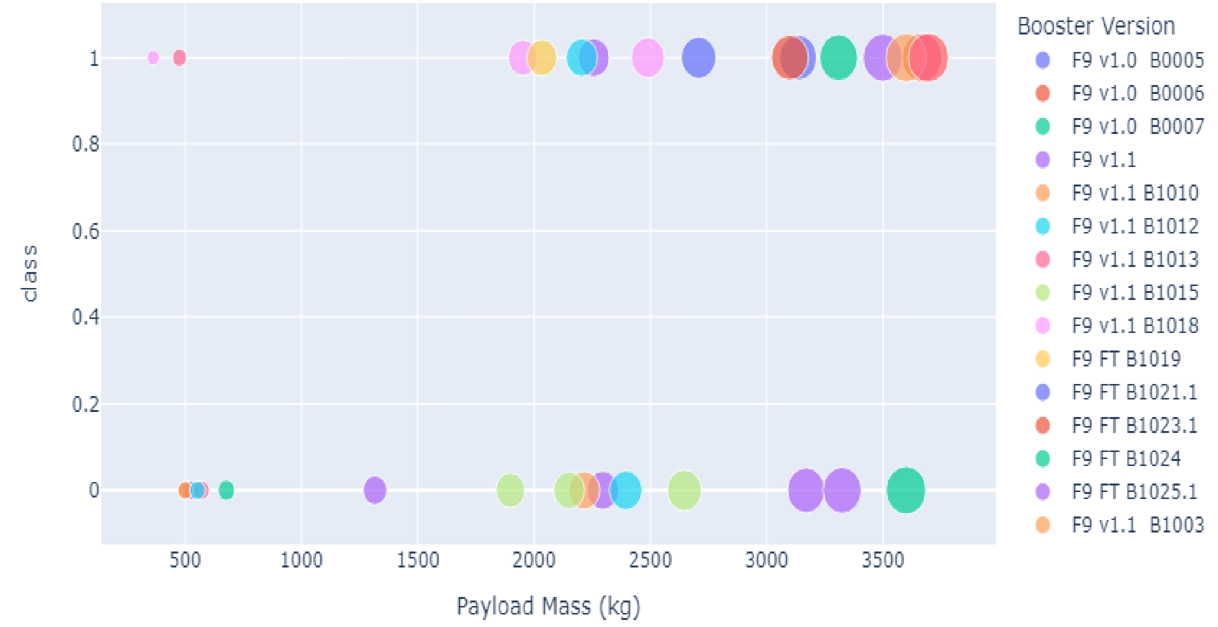
This site had a mammoth 76.9% successful launches



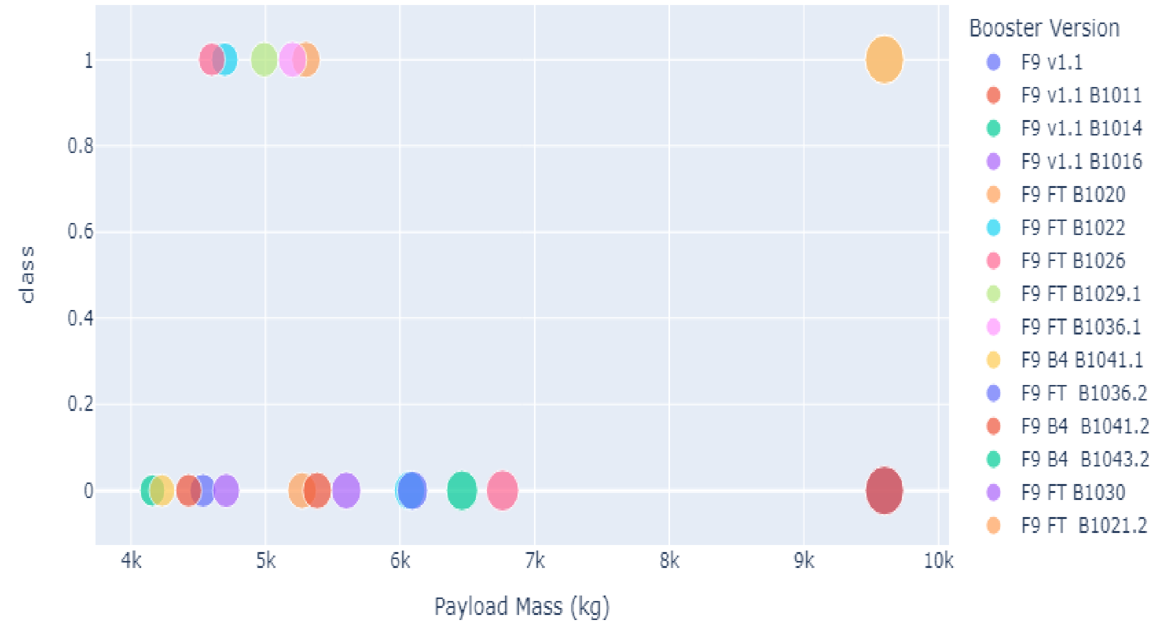
Payload vs Launch outcome scatter plot

We can see that all the success rate for low weighted payload is higher than heavy weighted payload

Low weighted payload 0kg-4000 kg



Heavy weighted payload 4000kg-10000 kg

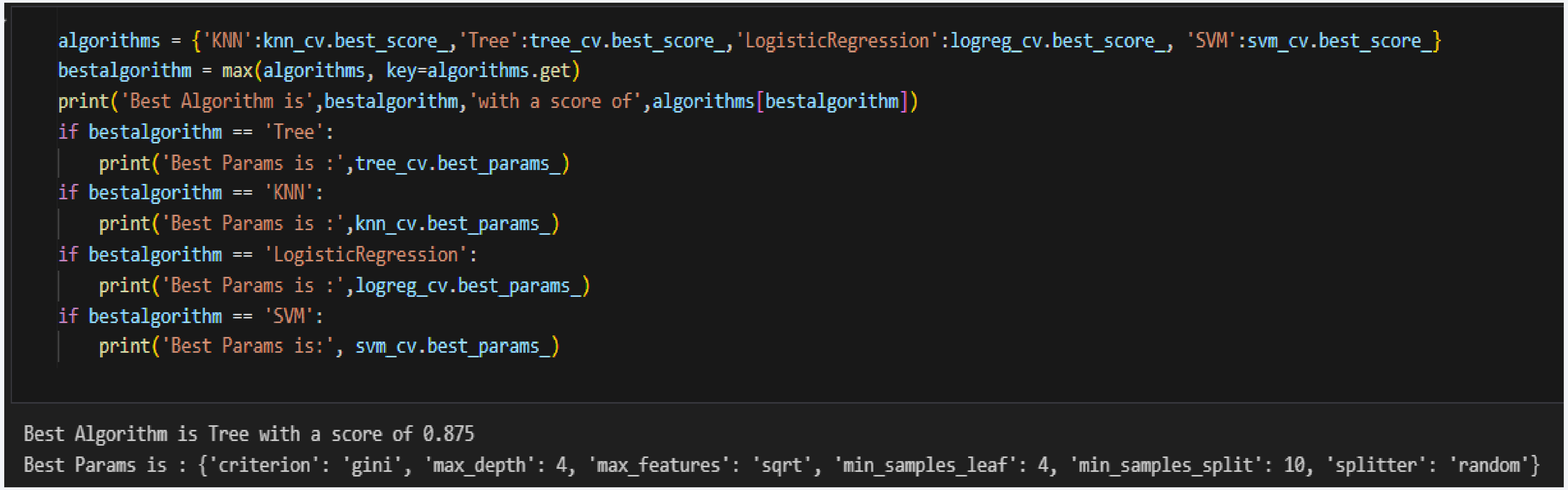


Section 5

Predictive Analysis (Classification)

Classification Accuracy

As we can see, by using the code as below: we could identify that the best algorithm to be the Tree Algorithm which have the highest classification accuracy.

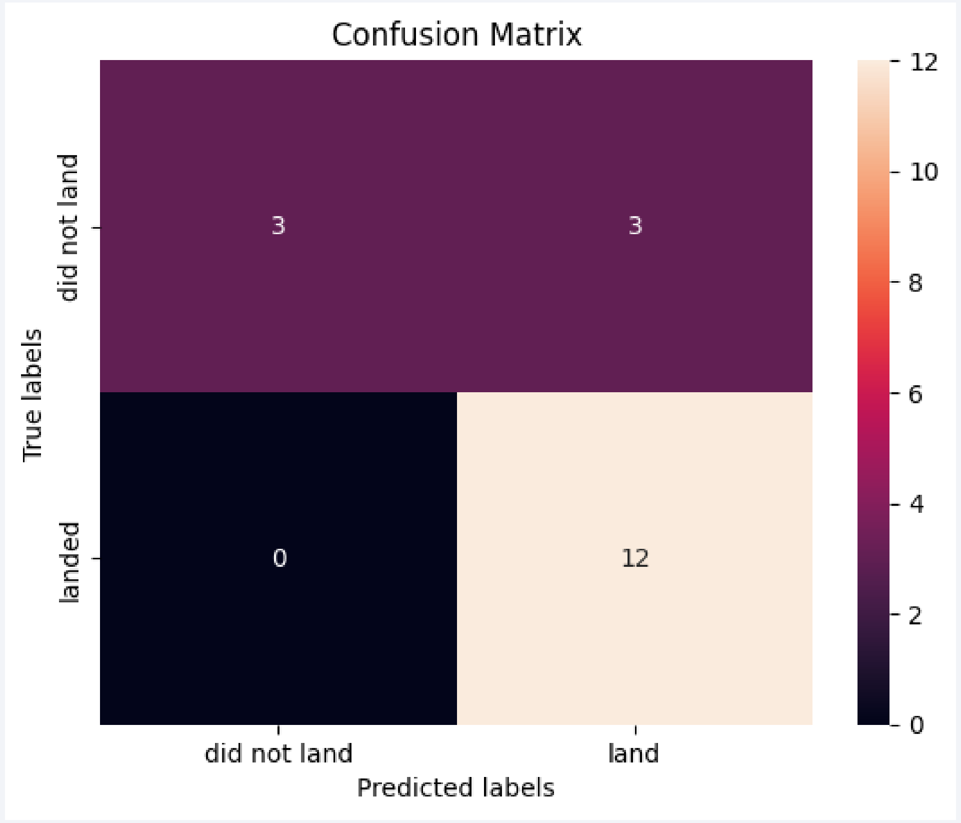


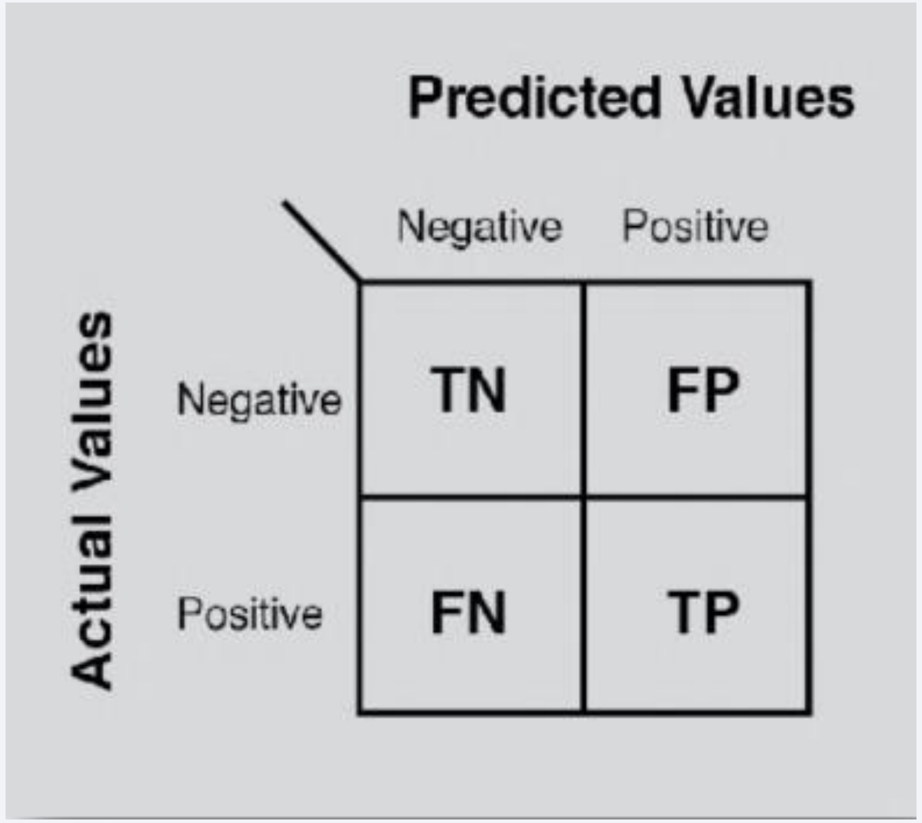
```
algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_, 'SVM':svm_cv.best_score_}
bestalgorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
if bestalgorithm == 'Tree':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

```
Best Algorithm is Tree with a score of 0.875
Best Params is : {'criterion': 'gini', 'max_depth': 4, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 10, '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 positives .i.e., unsuccessful landing marked as successful landing by the classifier.





Conclusions

We can conclude that:

- The Tree Classifier Algorithm is the best Machine Learning approach for this dataset.
- The low weighted payloads (which define as 4000kg and below) performed better than the heavy weighted payloads.
- 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.
- KSC LC-39A have the most successful launches of any sites; 76.9%
- SSO orbit have the most success rate; 100% and more than 1 occurrence.

Thank you!