IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Ismail Zahraoui
18 September 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
  - Exploratory Data Analysis with Data Visualization
  - Interactive Visual Analytics with Folium
  - Machine Learning Prediction
- **Summary of all results**
  - Exploratory Data Analysis result
  - Interactive analytics in screenshots
  - Predictive Analytics result from Machine Learning Lab

Introduction

SpaceX stands as a groundbreaking company that has revolutionized the space industry by offering rocket launches, particularly with the Falcon 9, at a significantly lower cost, starting as low as 62 million dollars, compared to other providers whose prices soar to a staggering 165 million dollars per launch. Much of these cost savings are attributed to SpaceX's ingenious concept of reusing the first stage of the rocket by successfully re-landing it for use in subsequent missions. The iterative application of this process promises further cost reduction.

The project's focal challenges encompass:

- Identifying and comprehensively documenting all the variables that exert an influence on the landing outcome.
- Investigating the intricate relationships between these variables and understanding how they collectively impact the outcome.
- Discerning the optimal conditions necessary to enhance the likelihood of a successful landing.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Describe how data was collected
- Perform data wrangling
  - Describe how data was processed
- 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 involves the systematic process of acquiring and quantifying information related to specific variables within an established system. This process empowers individuals to address pertinent questions and assess the resulting outcomes. In this context, the dataset was compiled through two primary methods: REST API and web scraping from Wikipedia.

When utilizing the REST API, we initiated the process with a "get" request. Subsequently, we decoded the response content as JSON and transformed it into a pandas dataframe using the "json_normalize()" function. Following this step, we meticulously cleaned the data, scrutinized it for any missing values, and addressed these gaps as required.

In the case of web scraping, we employed BeautifulSoup to extract launch records presented in HTML tables. We proceeded to parse the table and convert its contents into a pandas dataframe, facilitating further in-depth analysis.

Data Collection – SpaceX API

Get request for rocket launch data using API

Use json_normalize method to convert json result to dataframe

Performed data cleaning and filling the missing value

```
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 a
nd 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 rocket
s 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 s
ingle 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 ex
tracting 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)]
```

Github:

https://github.com/Ismail-Z97/AppliedDataScienceCapstone/blob/main/1-%20data-collection-api.ipynb

Data Collection - Scraping

Request the Falcon9 Launch Wiki page from url

Create a BeautifulSoup from the HTML response

Extract all column/variable names from the HTML header

```
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 a
nd 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 rocket
s 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 s
ingle 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 ex
tracting 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)]
```

Github:

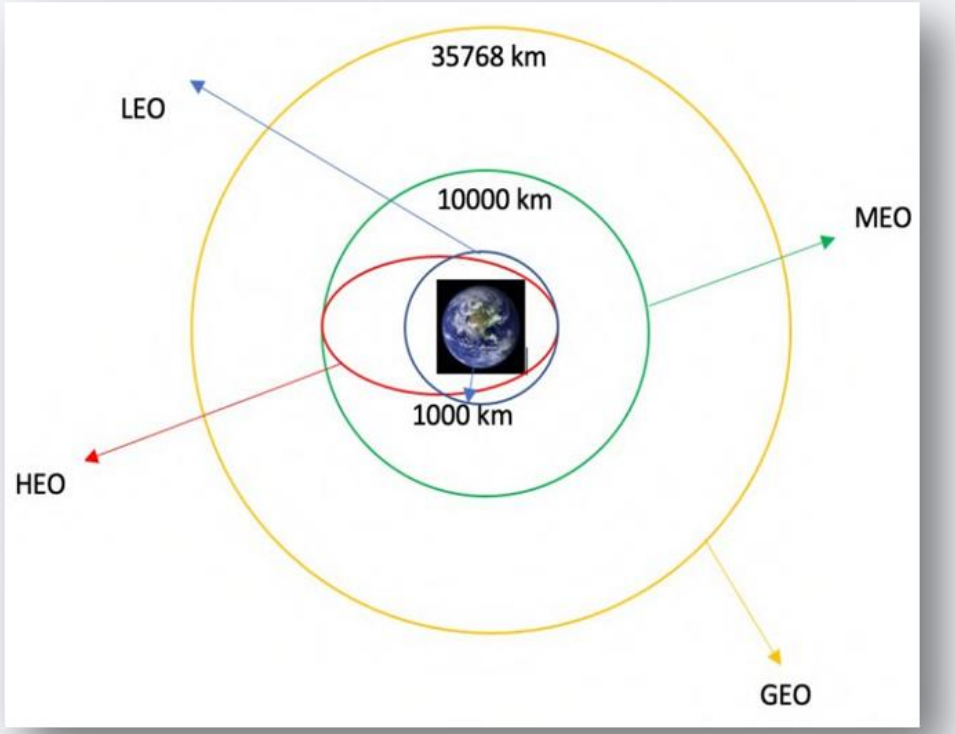
https://github.com/Ismail-Z97/AppliedDataScienceCapstone/blob/main/2-%20webscraping.ipynb

Data Wrangling

Data Wrangling is the process of cleaning and unifying messy and complex data sets for easy access and Exploratory Data Analysis (EDA).

We will first calculate the number of launches on each site, then calculate the number and occurrence of mission outcome per orbit type.

We then create a landing outcome label from the outcome column. This will make it easier for further analysis, visualization, and ML. Lastly, we will export the result to a CSV.

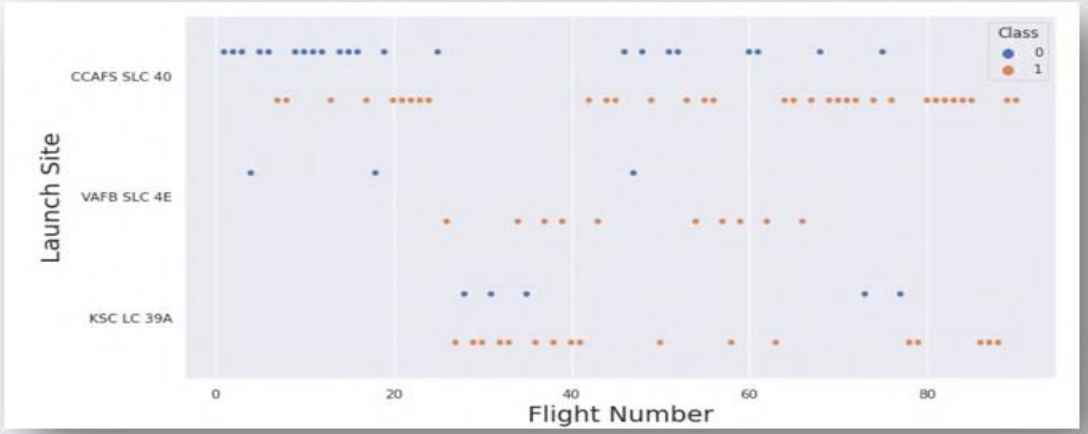


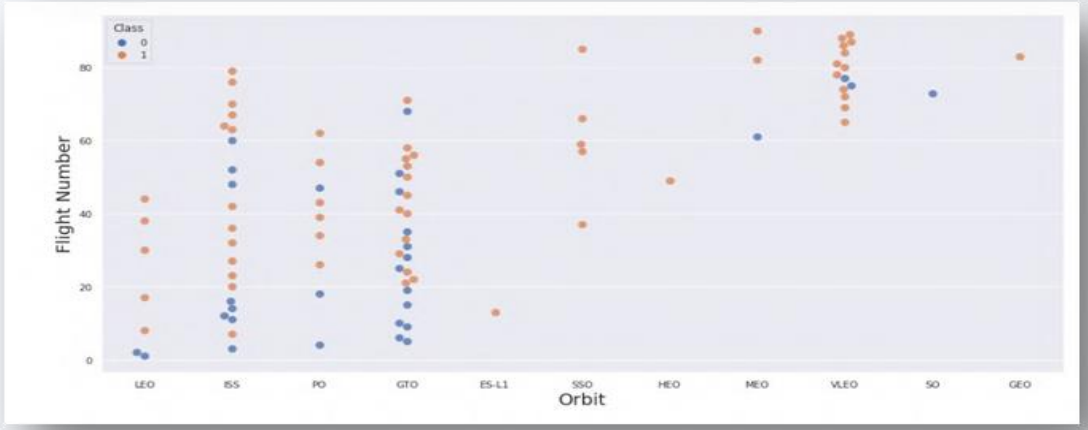
Github:

https://github.com/Ismail-Z97/AppliedDataScienceCapstone/blob/main/3-%20data_wrangling_jupyterlite.jupyterlite.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:

https://github.com/Ismail-Z97/AppliedDataScienceCapstone/blob/main/5-%20eda-dataviz.ipynb.jupyterlite.ipynb

EDA with SQL

Using SQL, we performed various queries to gain a deeper understanding of the dataset, including:

- 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 boosters 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 on a ground pad was achieved.
- Listing the names of boosters that successfully landed on a drone ship with a payload mass greater than 4000 but less than 6000.
- Listing the total number of successful and failure mission outcomes.
- Listing the names of booster versions that carried the maximum payload mass.
- Listing the failed landing outcomes on a drone ship, including their booster versions and launch site names for the year 2015.
- Ranking the count of landing outcomes or success between the dates 2010-06-04 and 2017-03-20 in descending order.

Github:

https://github.com/Ismail-Z97/AppliedDataScienceCapstone/blob/main/4-%20eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

To create an interactive map visualizing the launch data, we followed these steps:

We extracted latitude and longitude coordinates for each launch site.

We added a circle marker around each launch site, accompanied by a label containing the launch site's name.

We categorized the launch outcomes (failure and success) in the dataframe into classes 0 and 1, representing them with Red and Green markers, respectively, using MarkerCluster().

We then applied Haversine's formula to calculate the distances between the launch sites and various landmarks. This allowed us to answer questions such as:

- The proximity of launch sites to railways, highways, and coastlines.
- The proximity of launch sites to nearby cities.

Github:

https://github.com/Ismail-Z97/AppliedDataScienceCapstone/blob/main/6-%20launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

We developed an interactive dashboard using Plotly Dash, enabling users to explore and manipulate the data according to their requirements. Within the dashboard:

- We generated pie charts to illustrate the total launches conducted at specific sites.
- We also created scatter plots to visualize the relationship between the outcome and payload mass (in kilograms) for various booster versions.

Github:

https://github.com/Ismail-Z97/AppliedDataScienceCapstone/blob/main/7-%20SpacexDashBoard.py

Predictive Analysis (Classification)

Building the Model

- Load the dataset into NumPy and Pandas
- Transform the data and then split into training and test datasets
- Decide which type of ML to use
- set the parameters and algorithms to GridSearchCV and fit it to dataset.

Evaluating the Model

- Check the accuracy for each model
- Get tuned hyperparameters for each type of algorithms.
- plot the confusion matrix.

Improving the Model

- Use Feature Engineering and Algorithm Tuning

Find the Best Model

- The model with the best accuracy score will be the best performing model.

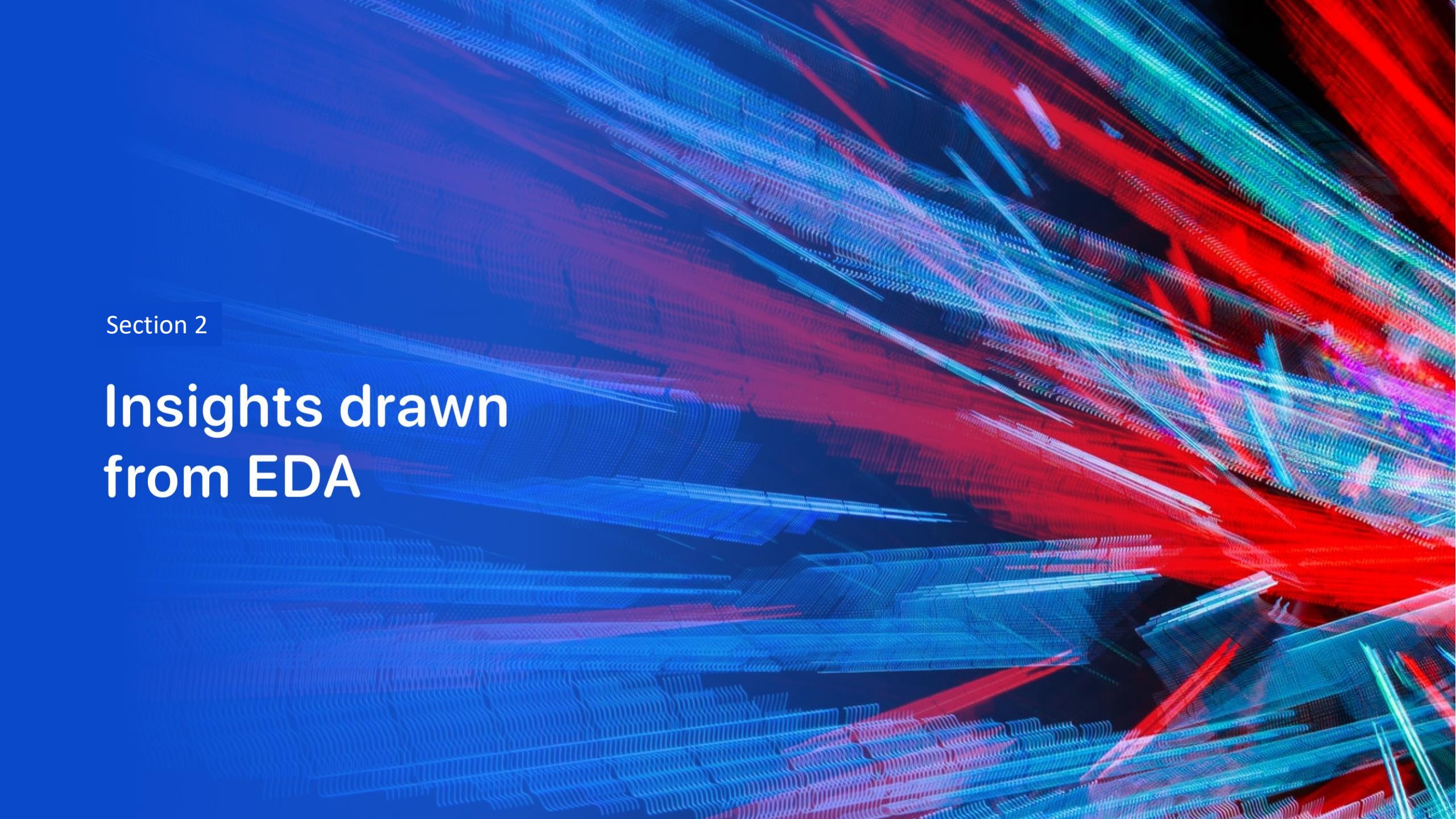
Github:

https://github.com/Ismail-Z97/AppliedDataScienceCapstone/blob/main/8-%20Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Results

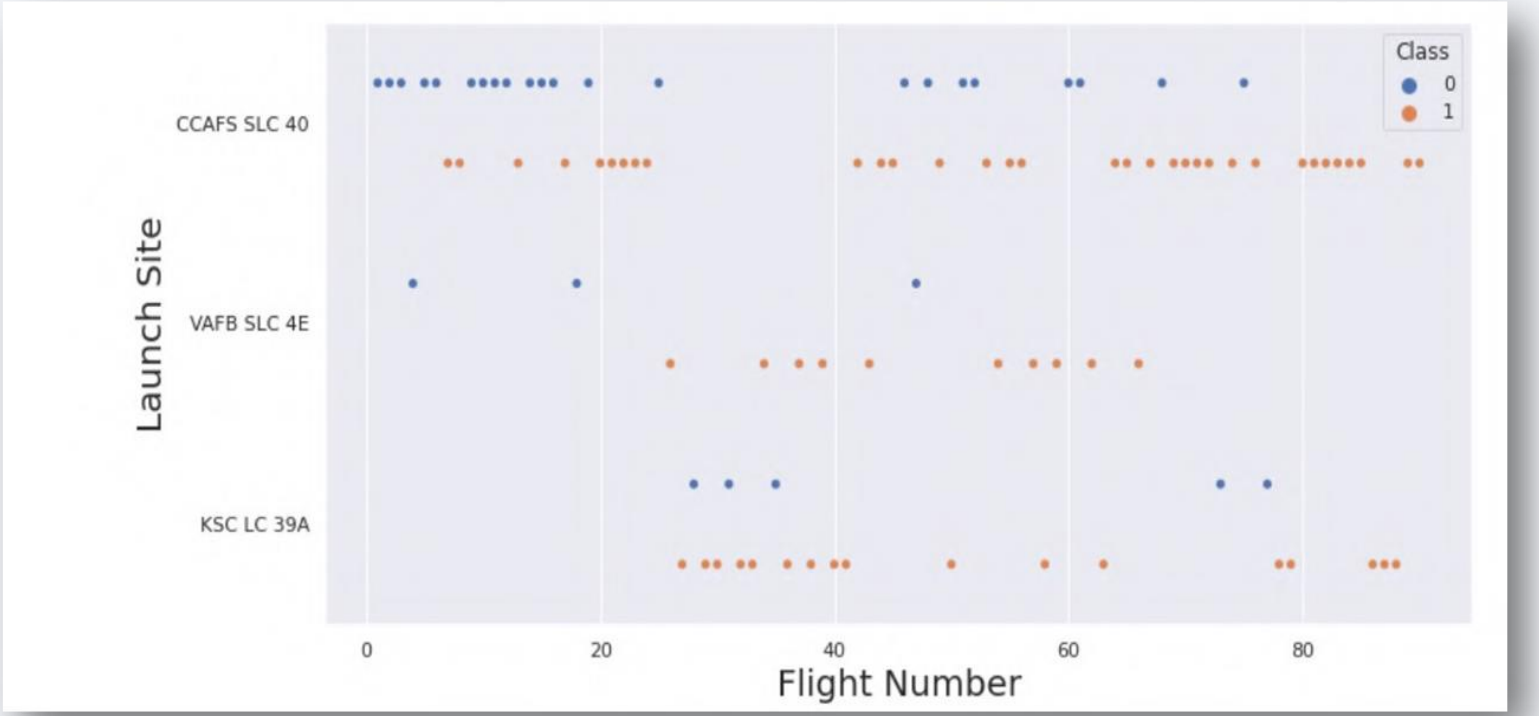
Our results will be explored in 3 main parts :

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


Section 2
Insights drawn from EDA

Flight Number vs. Launch Site

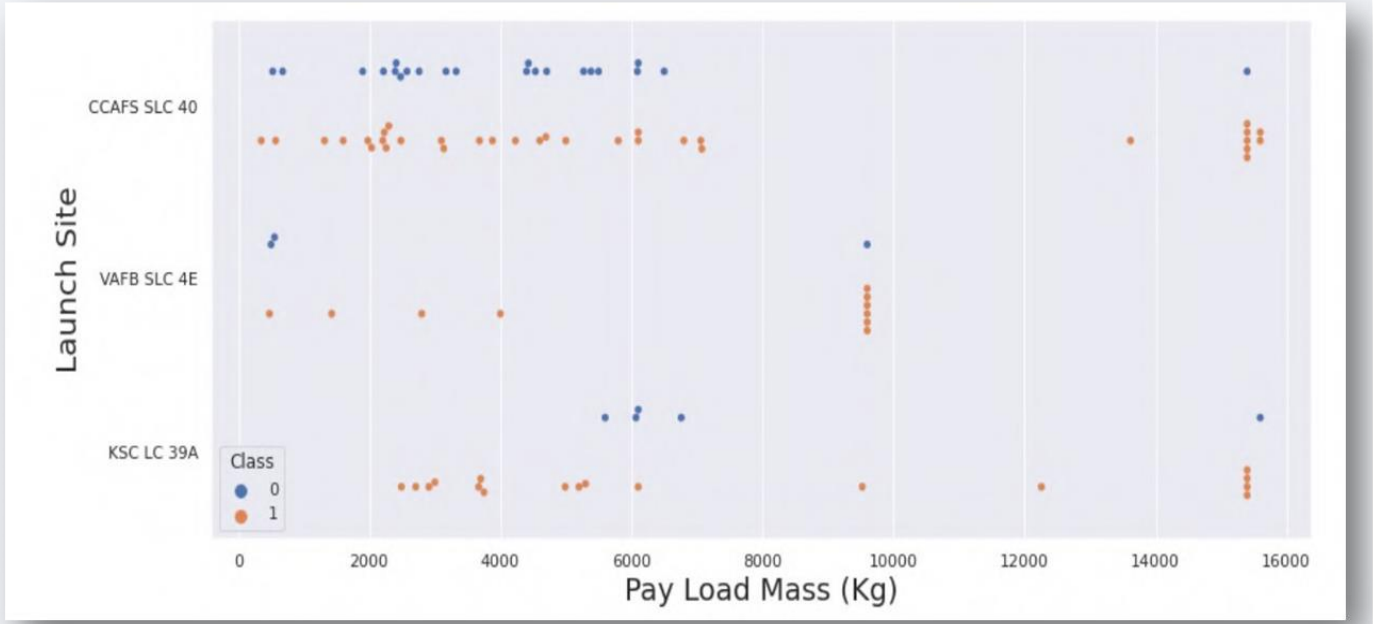
This scatter plot shows that the larger the flights amount of the launch site, the greater the the success rate will be. However, site CCAFS SLC40 shows the least pattern of this.



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.

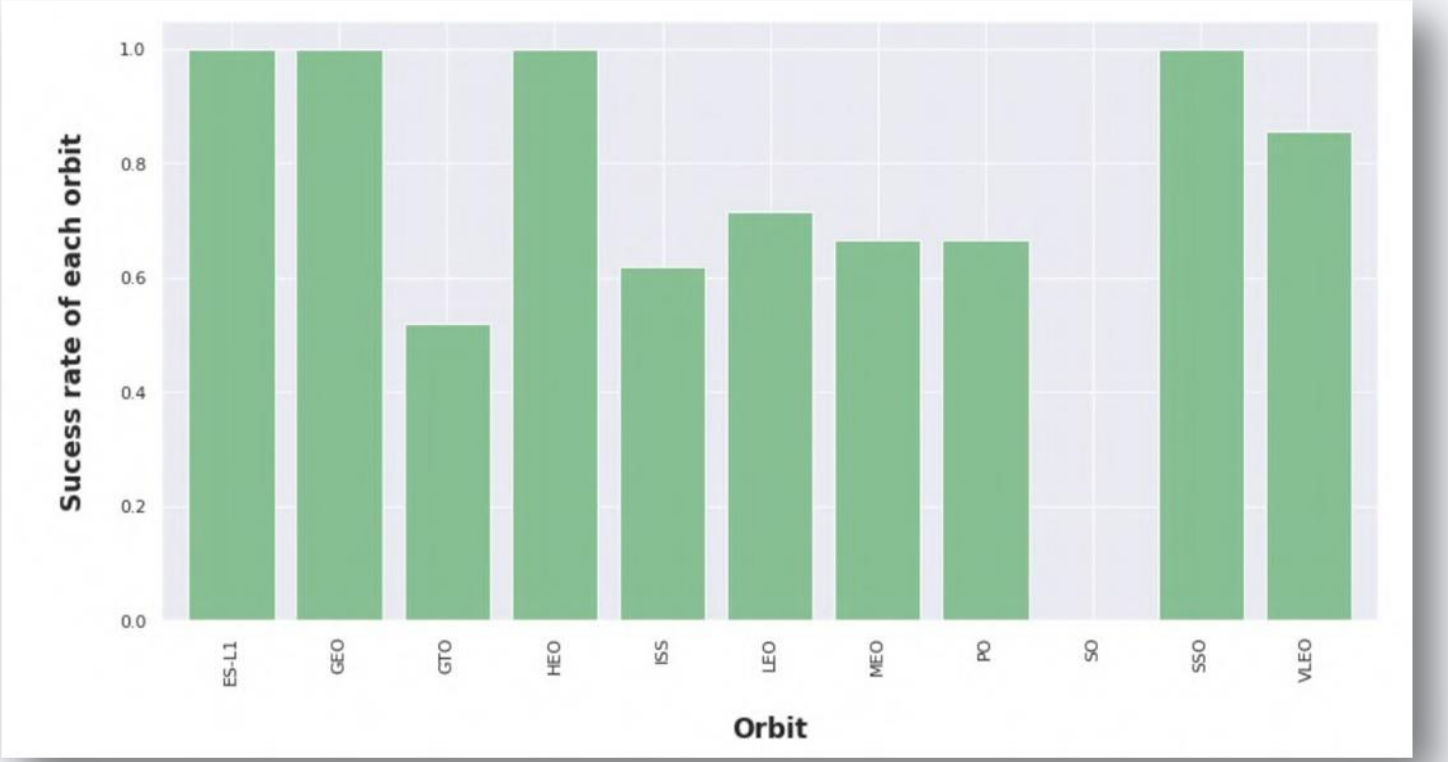
However, there is no clear pattern to say the launch site is dependent to the pay load mass for the success rate.



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.



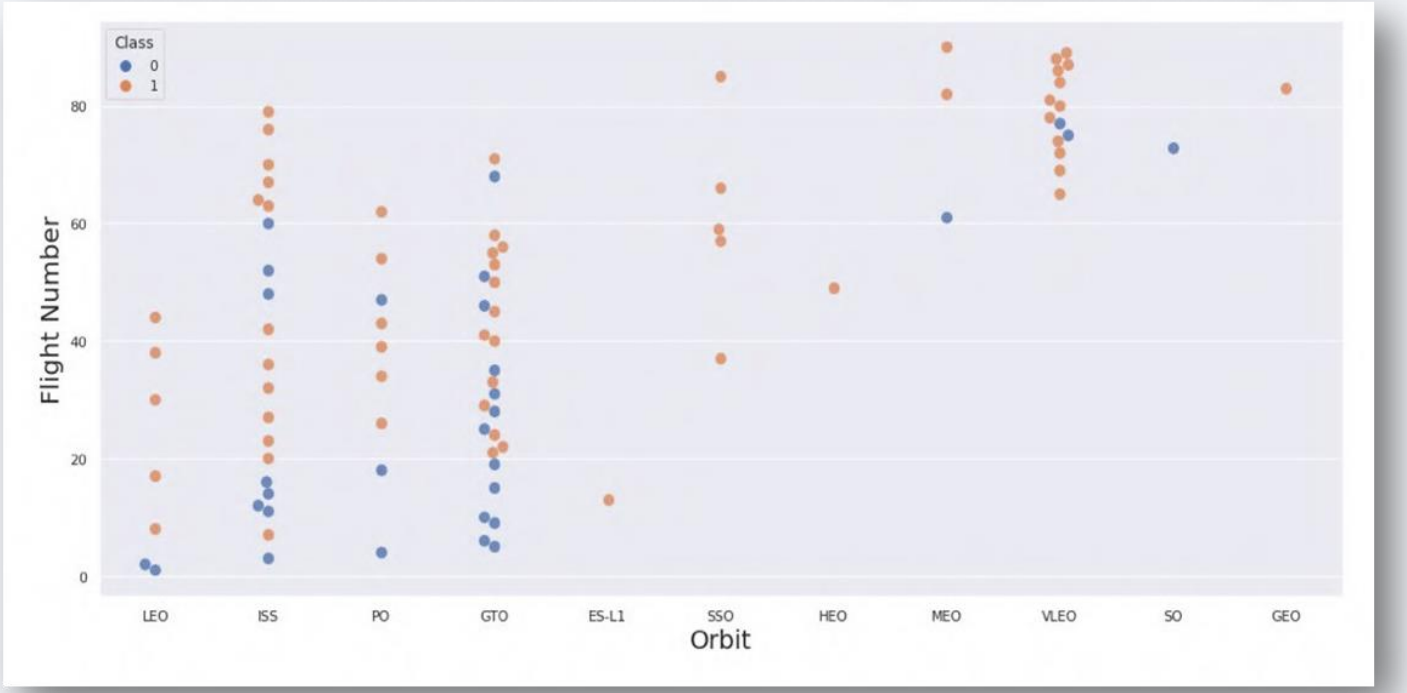
Flight Number vs. Orbit Type

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

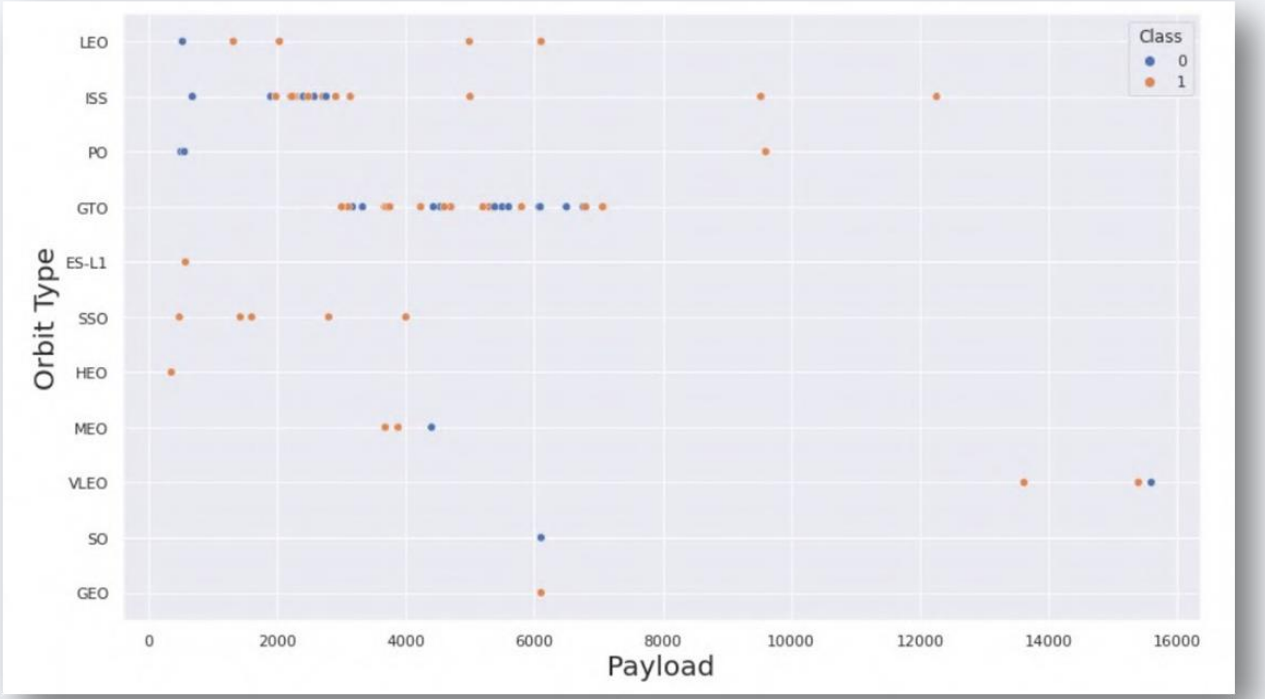


Payload vs. Orbit Type

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.

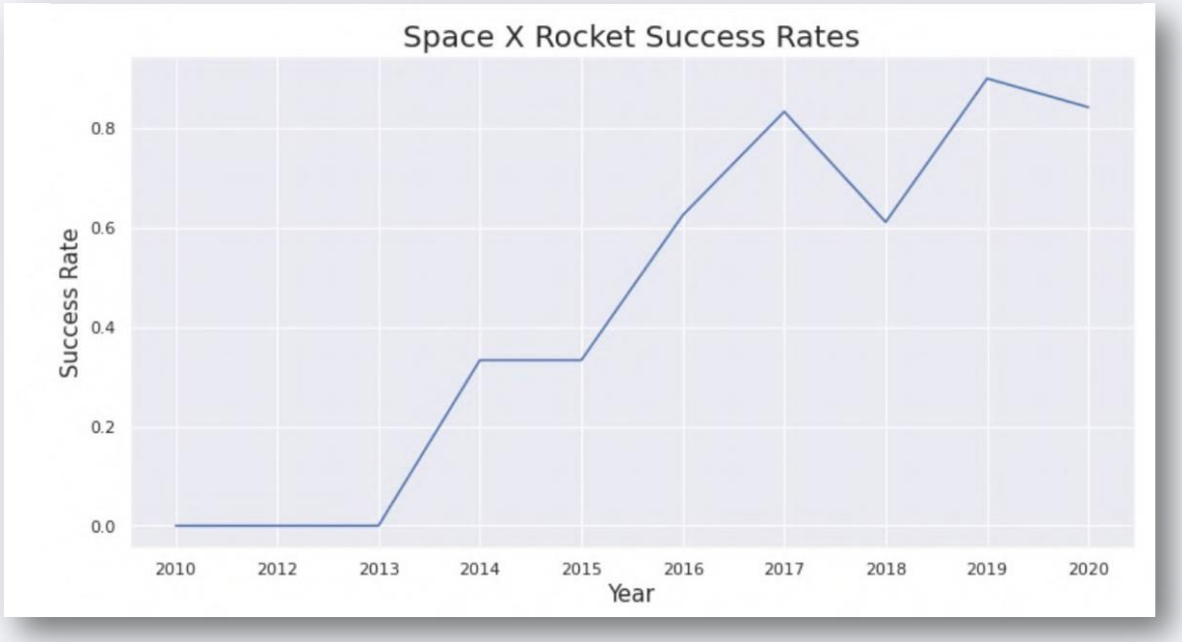
Meanwhile, again, SO, GEO and HEO orbit need more dataset to see any pattern or trend.



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.

```
In [5]: %sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEX;
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3
sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

Out[5]:

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

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'

In [11]:
```
task_2 = '''
        SELECT *
        FROM SpaceX
        WHERE LaunchSite LIKE 'CCA%'
        LIMIT 5
        '''
create_pandas_df(task_2, database=conn)
```

Out[11]:

	date	time	boosterversion	launchsite	payload	payloadmasskg	orbit	customer	missionoutcome	landingoutcome
0	2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	2010-08-12	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2	2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
3	2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

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

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

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS "Total Payload Mass by NASA (CRS)
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3
sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

Total Payload Mass by NASA (CRS)
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

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS "Average Payload Mass by Booster
WHERE BOOSTER_VERSION = 'F9 v1.1';
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3
sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

Average Payload Mass by Booster Version F9 v1.1
2928

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 Succesful Landing Outcome in Ground Pad
WHERE LANDING__OUTCOME = 'Success (ground pad)';
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3
sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

First Succesful Landing Outcome in Ground Pad
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 SPACEX WHERE LANDING__OUTCOME = 'Success (drone ship)' \
AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
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.

List the total number of successful and failure mission outcomes

```
%sql SELECT COUNT(MISSION_OUTCOME) AS "Successful Mission" FROM SPACEX WHERE MISSION_OUTCOME LIKE 'Success%';
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

Successful Mission
100

```
%sql SELECT COUNT(MISSION_OUTCOME) AS "Failure Mission" FROM SPACEX WHERE MISSION_OUTCOME LIKE 'Failure%';
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
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 SPACEX
WHERE PAYLOAD_MASS__KG_ =(SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEX);
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.clou
d:32731/bludb
Done.
```

Booster Versions which carried the Maximum Payload Mass
F9 B5 B1048.4
F9 B5 B1048.5
F9 B5 B1049.4
F9 B5 B1049.5
F9 B5 B1049.7
F9 B5 B1051.3
F9 B5 B1051.4
F9 B5 B1051.6
F9 B5 B1056.4
F9 B5 B1058.3
F9 B5 B1060.2
F9 B5 B1060.3

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 SPACEX WHERE DATE LIKE '2015-%' AND \
LANDING__OUTCOME = 'Failure (drone ship)';
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.
databases.appdomain.cloud:32731/bludb
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 SPACEX \
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' \
GROUP BY  LANDING__OUTCOME \
ORDER BY COUNT(LANDING__OUTCOME) DESC ;
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

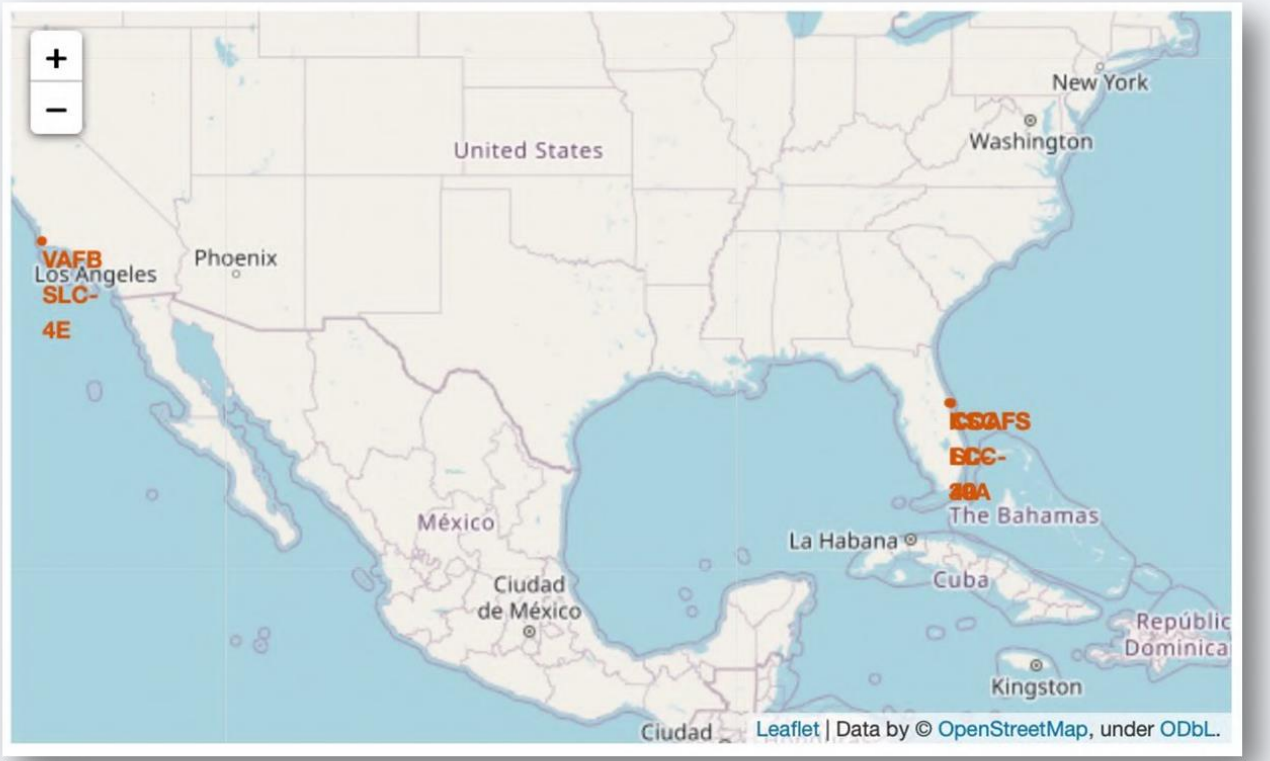
Landing Outcome	Total Count
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Failure (parachute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

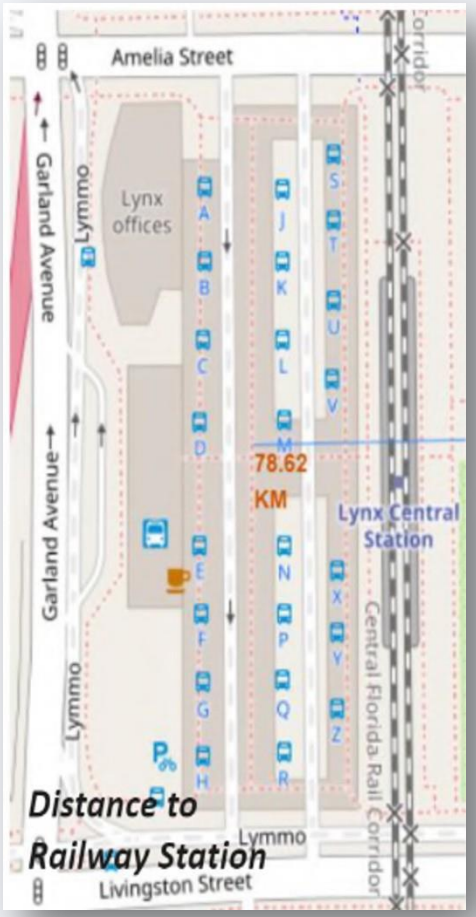
<Folium Map Screenshot 1>

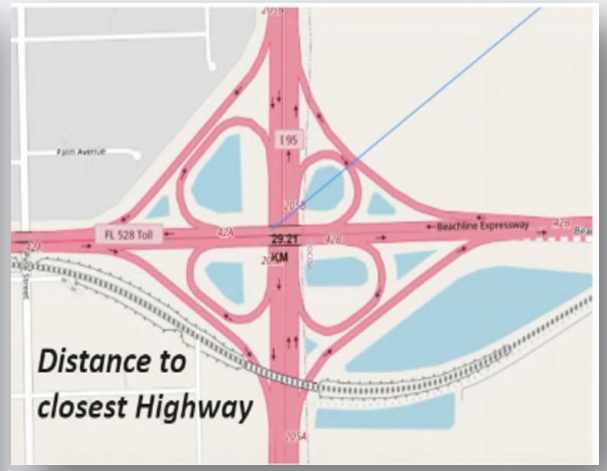
We can see that all the SpaceX launch sites are located inside the United States

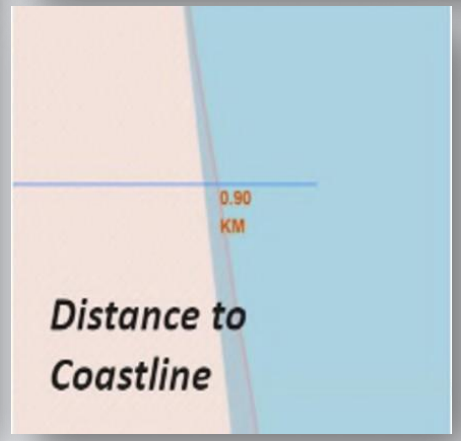


<Folium Map Screenshot 2>

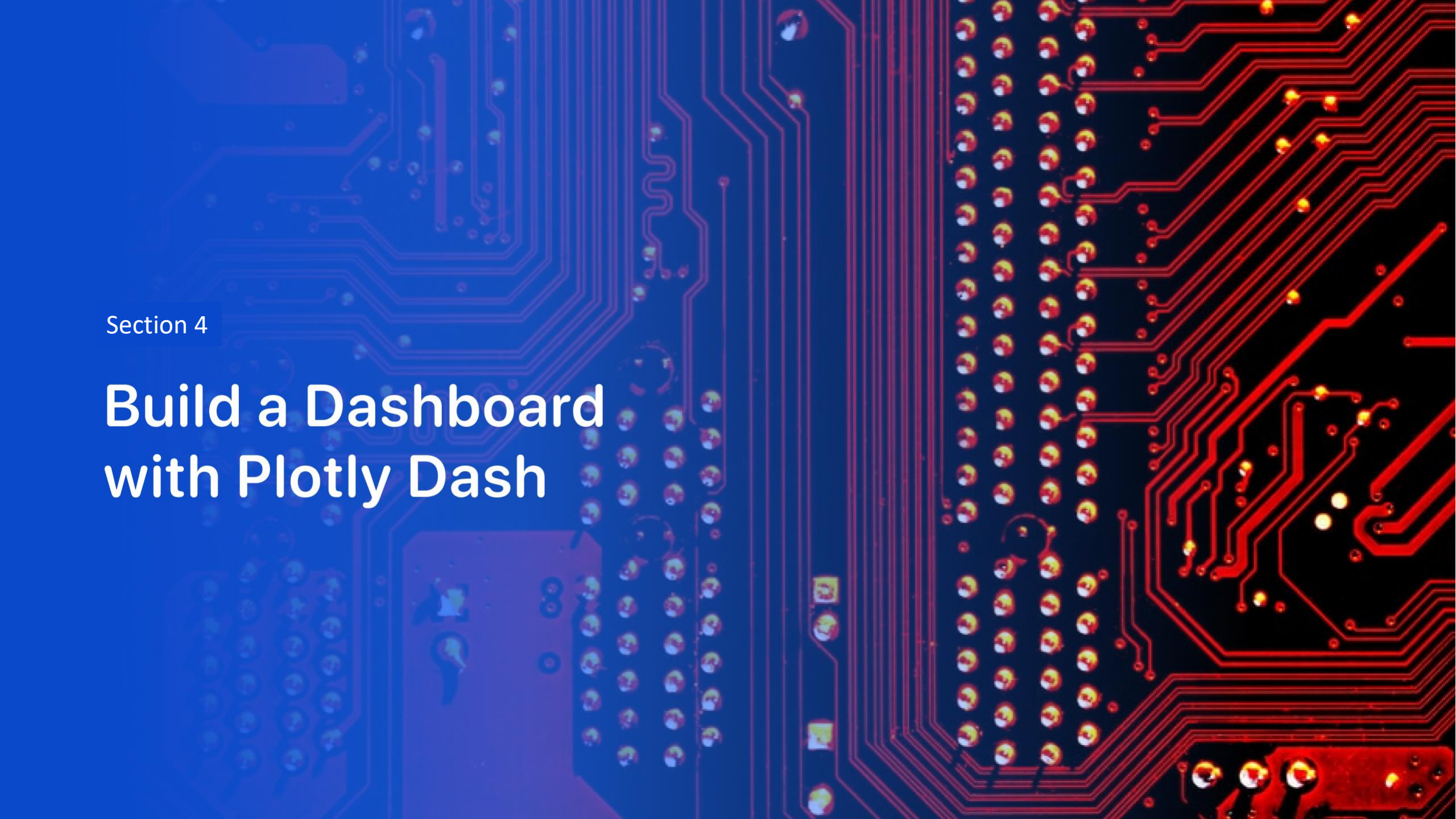
<Folium Map Screenshot 3>



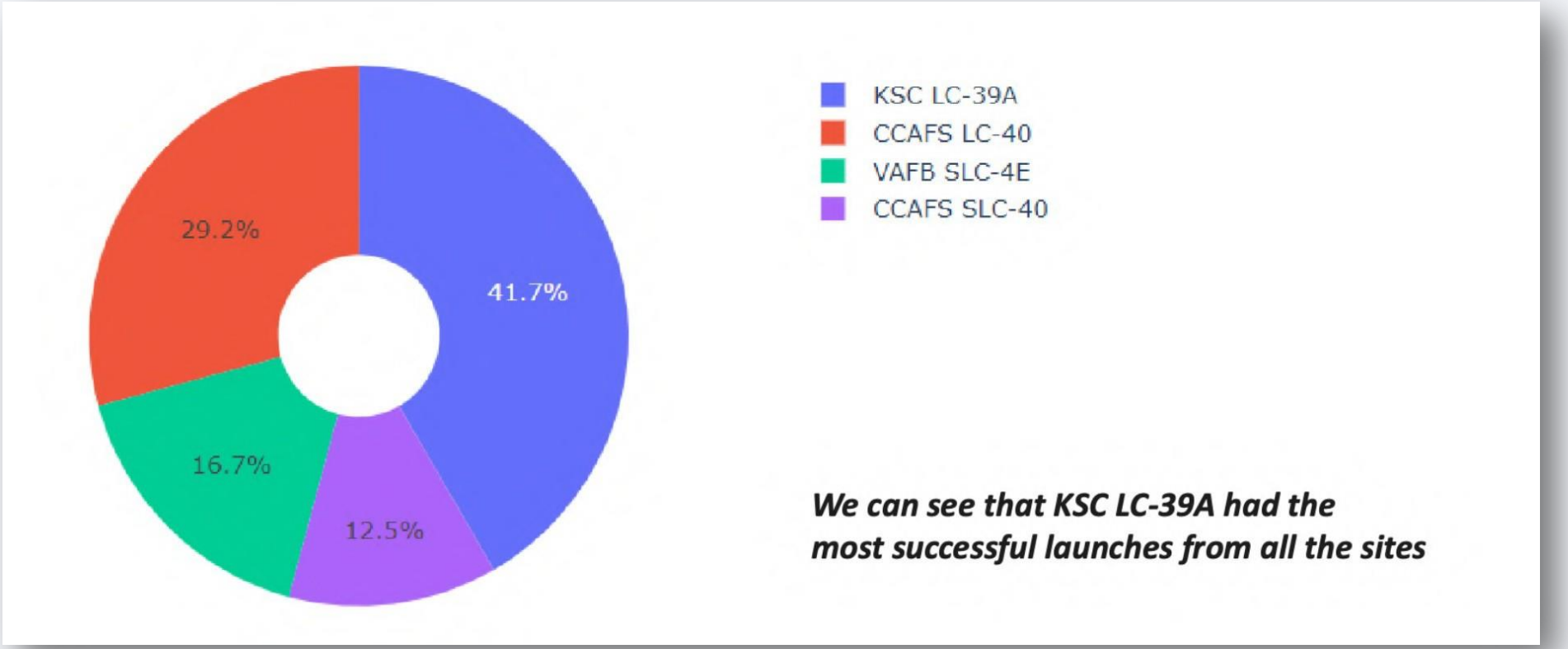




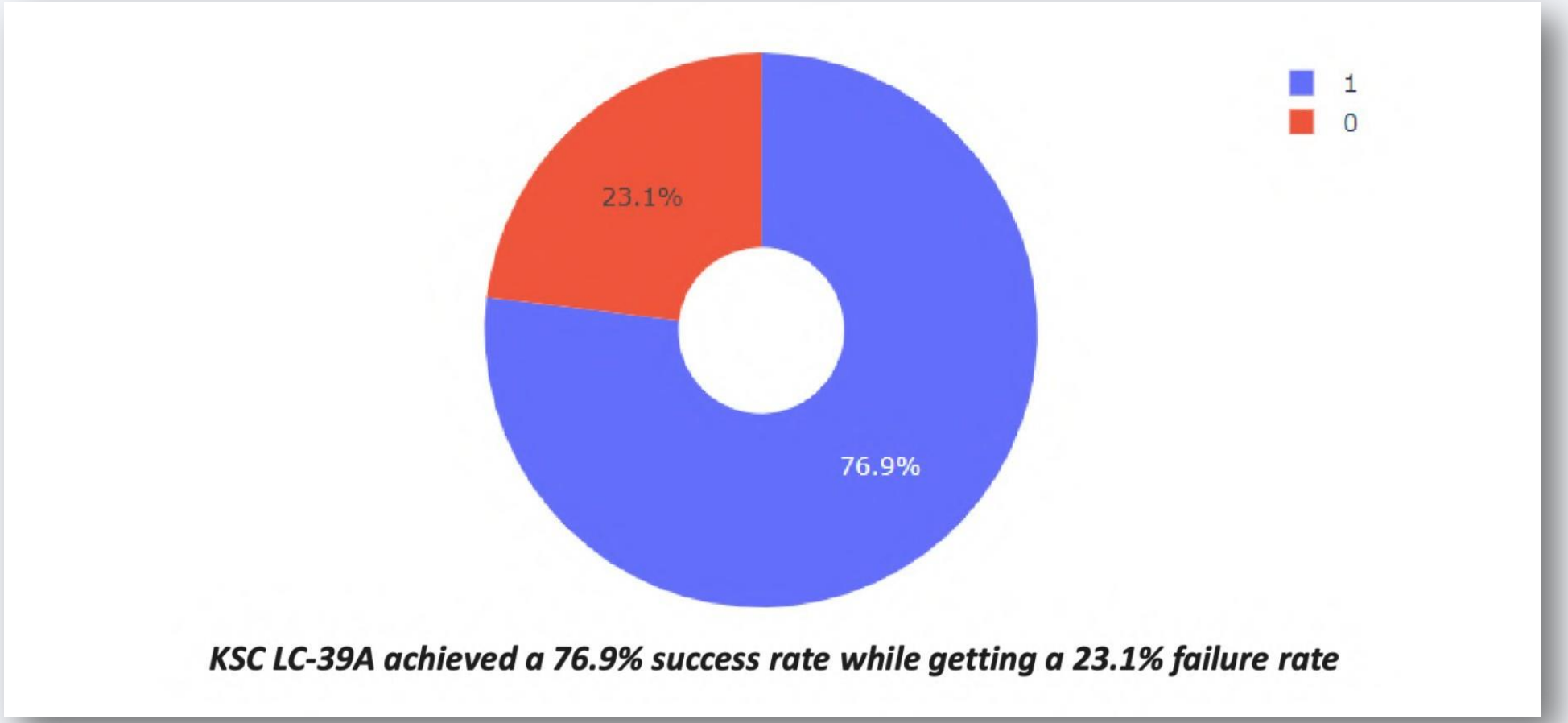
- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes


Section 4
Build a Dashboard with Plotly Dash

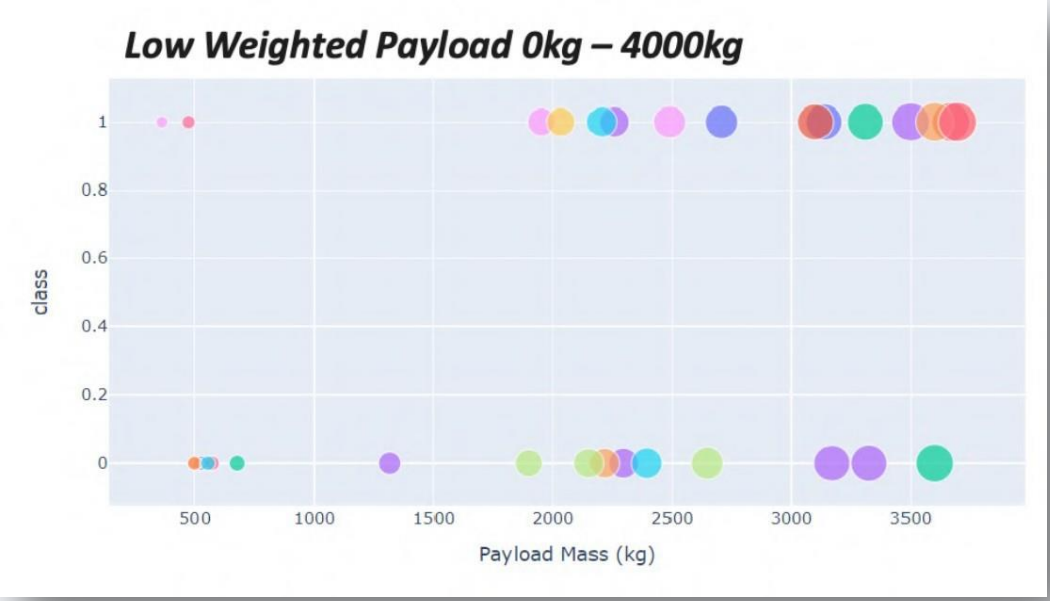
<Dashboard Screenshot 1>

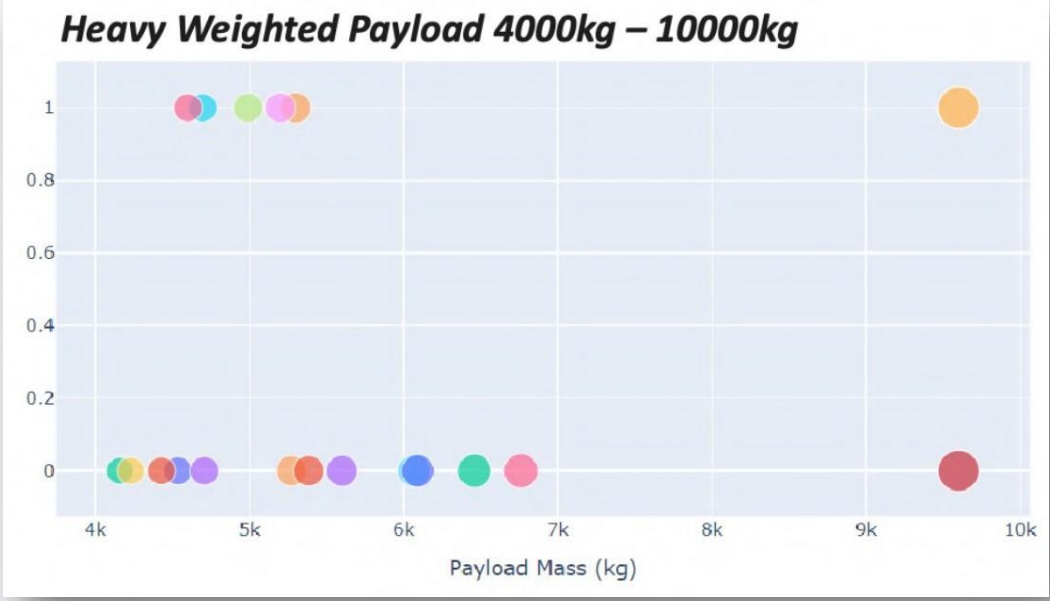


<Dashboard Screenshot 2>



<Dashboard Screenshot 3>





Section 5

Predictive Analysis (Classification)

Classification Accuracy

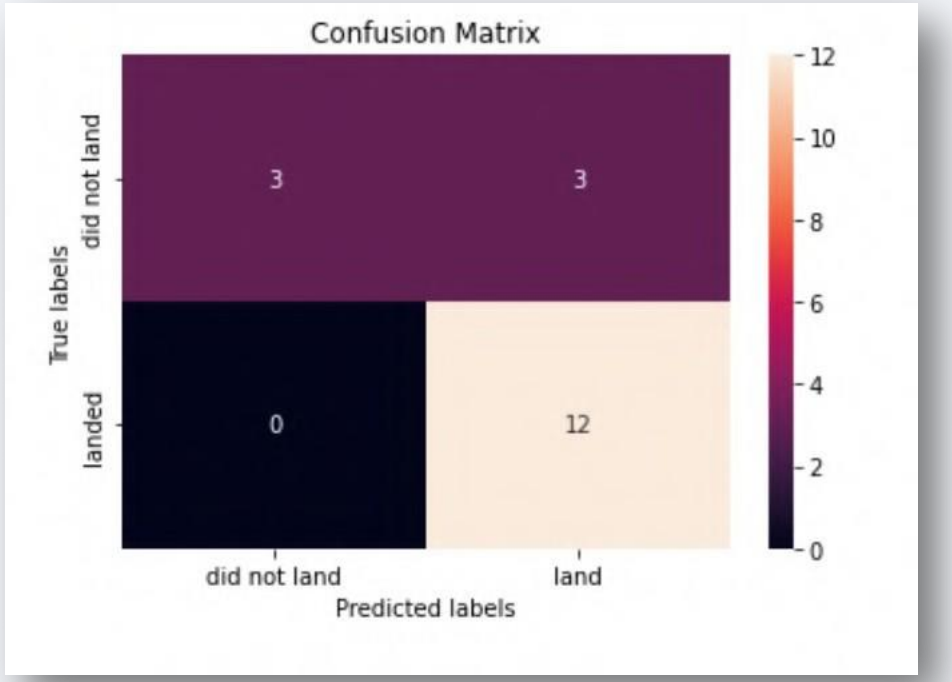
As we can see, by using the code as below: we could identify that the best algorithm to be the Tree Algorithm which have the highest classification accuracy.

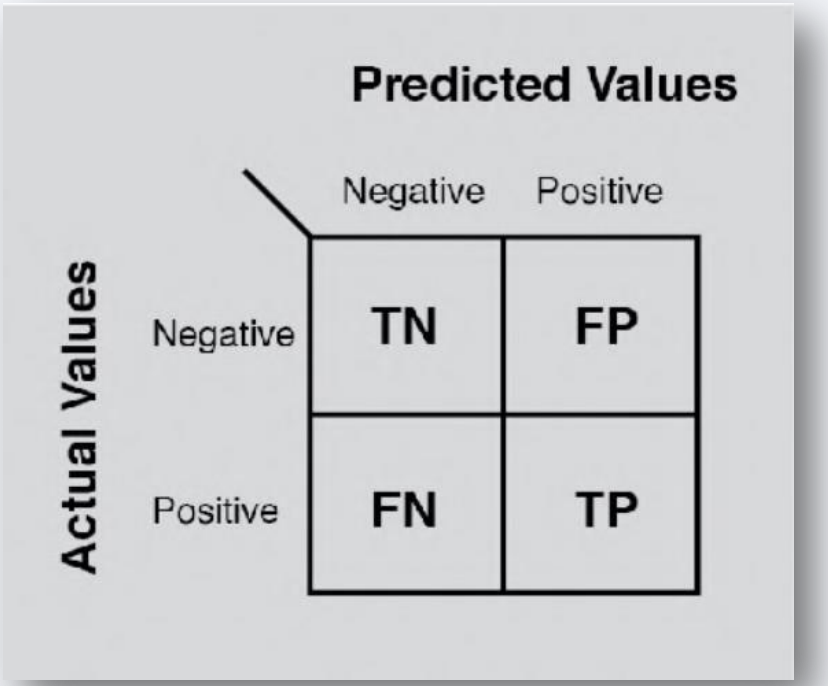
```
algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_}
bestalgorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
if bestalgorithm == 'Tree':
    print('Best Params 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 0.9017857142857142
Best Params is : {'criterion': 'entropy', 'max_depth': 10, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_sampl
es_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; unsuccessful landing marked as successful landing by the classifier.





Conclusions

- The Tree Classifier Algorithm stands out as the most suitable Machine Learning approach for this dataset.
- Payloads with weights below 4000kg exhibited superior performance compared to their heavier counterparts.
- Commencing from 2013, the success rate of SpaceX launches has consistently increased in direct proportion to the years, with the expectation of achieving perfect launches in the future up to 2020.
- Among all launch sites, KSC LC-39A boasts the highest success rate at 76.9%.
- The SSO orbit displays an exceptional success rate of 100% with multiple successful occurrences.

Appendix

Github Links for Created Datasets :

- https://github.com/Ismail-Z97/AppliedDataScienceCapstone/blob/main/1-%20dataset_part_1.csv
- https://github.com/Ismail-Z97/AppliedDataScienceCapstone/blob/main/2-%20dataset_part_2.csv
- https://github.com/Ismail-Z97/AppliedDataScienceCapstone/blob/main/3-%20jupyter%20spacex_web_scraped.csv
- https://github.com/Ismail-Z97/AppliedDataScienceCapstone/blob/main/5-%20jupyter%20dataset_part_3.csv

Thank you!