IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

BRIJESH A
8.3.2024

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 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.

The problems included:

- 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:
  - SpaceX websites or space agencies' databases that provide information on past launches. SpaceX's own website often provides detailed information on their launches.
  - If there are no APIs available, you can scrape data from websites that publish information about SpaceX launches. Be sure to review the website's terms of service and respect any rate limits or restrictions they have in place.
- Perform data wrangling
  - t involves tasks such as handling missing values, correcting data types, performing feature engineering, and integrating data from multiple sources. Data wrangling is a critical step in the data analysis pipeline, as it ensures that the data is accurate, consistent, and well-structured for further analysis and visualization.
  - Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models

Data Collection

- **Official Sources: Visit official SpaceX websites or space agencies' databases that provide information on past launches. SpaceX's own website often provides detailed information on their launches.**
- **APIs: Check if there are any APIs available that provide access to SpaceX launch data. Some space-related APIs or general data APIs may offer endpoints for retrieving information about SpaceX launches.**
- **Web Scraping: If there are no APIs available, you can scrape data from websites that publish information about SpaceX launches. Be sure to review the website's terms of service and respect any rate limits or restrictions they have in place.**
- **Public Datasets: Look for public datasets available on platforms like Kaggle, GitHub, or government databases that include SpaceX launch data. These datasets may already be cleaned and formatted for analysis.**
- **Manual Entry: If necessary, you could manually enter data from reliable sources into a spreadsheet or database for use in your dashboard.**

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

From:
https://github.com/16Brijesh10/APPLIED_DATA_SCIENCE_CAPSTONE/blob/main/week%201/spacex%20data%20collection/data%20collection%20api/1%20jupyter-labs-spacex-data-collection-api%20(1).ipynb

```
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)]
```

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

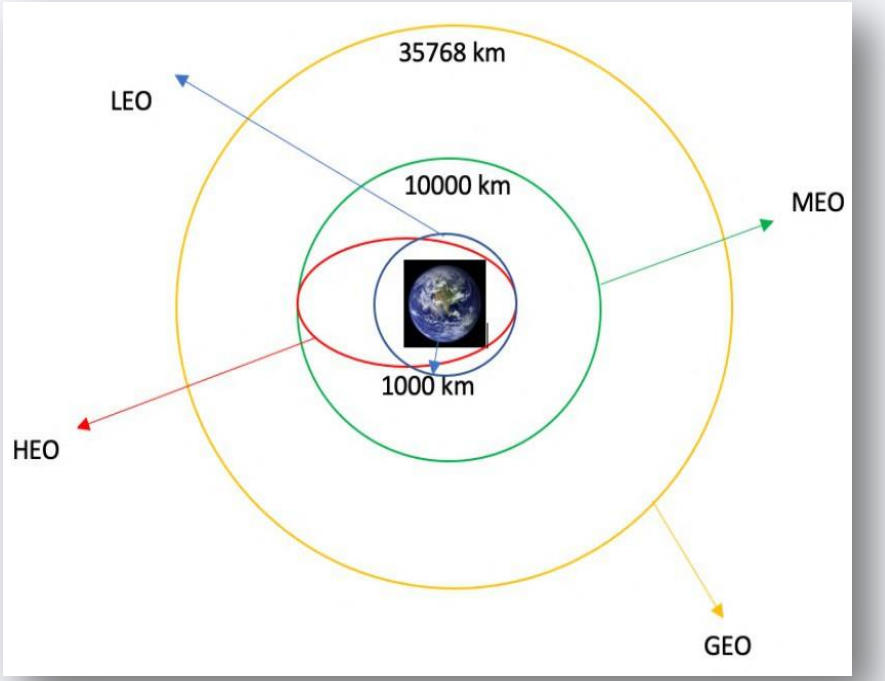
- From: https://github.com/16Brijesh10/APPLIED_DATA_SCIENCE_CAPSTONE/blob/main/week%201/spacex%20data%20collection/web%20scrapping%20data/2%20jupyter-labs-webscraping.ipynb

```
# 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 te
xt content
soup = BeautifulSoup(data,'html.parser')
```

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plai
nrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding t
o launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
```

Data Wrangling



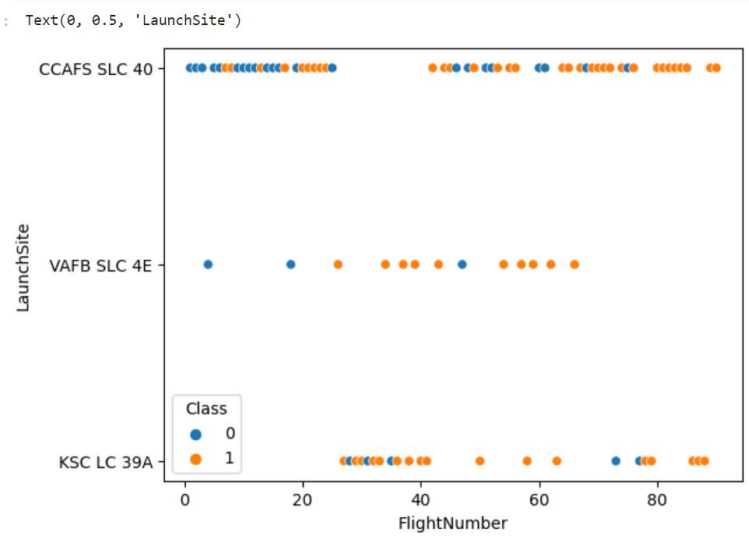
From:
https://github.com/16Brijesh10/APPLIED_DATA_SCIENCE_CAPSTONE/blob/main/week%201/Data%20wrangling/3%20labs-jupyter-spacex-Data%20wrangling.ipynb

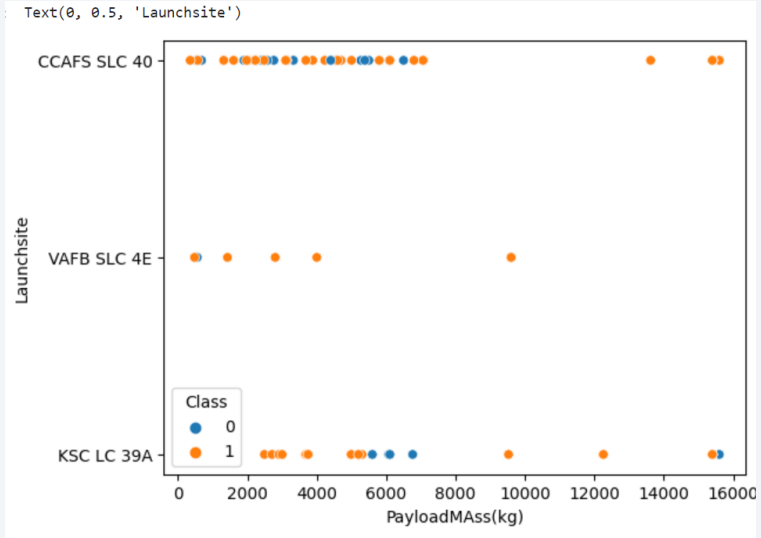
Data Wrangling is the process of cleaning and unifying messy and complex data sets for easy access and Exploratory Data Analysis (EDA).

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.

EDA with Data Visualization





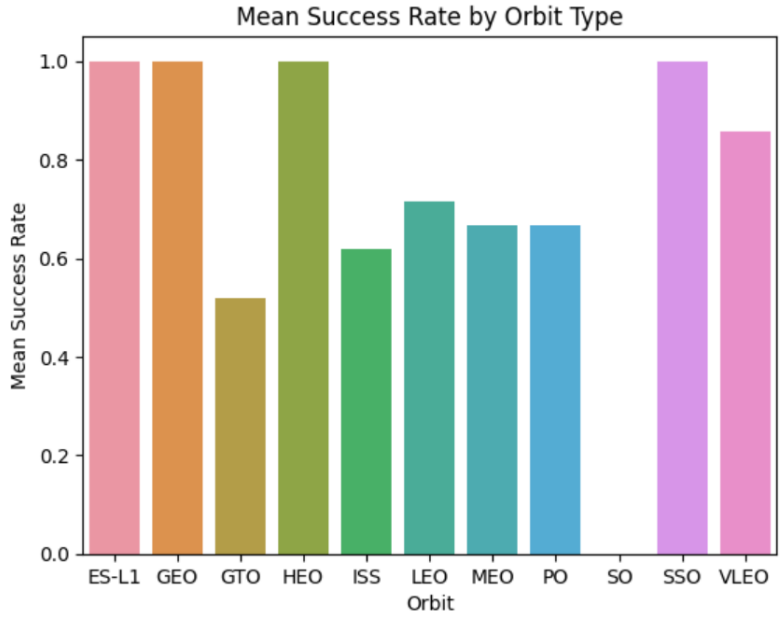
We first started by using scatter graph to find the relationship between the attributes such as between:

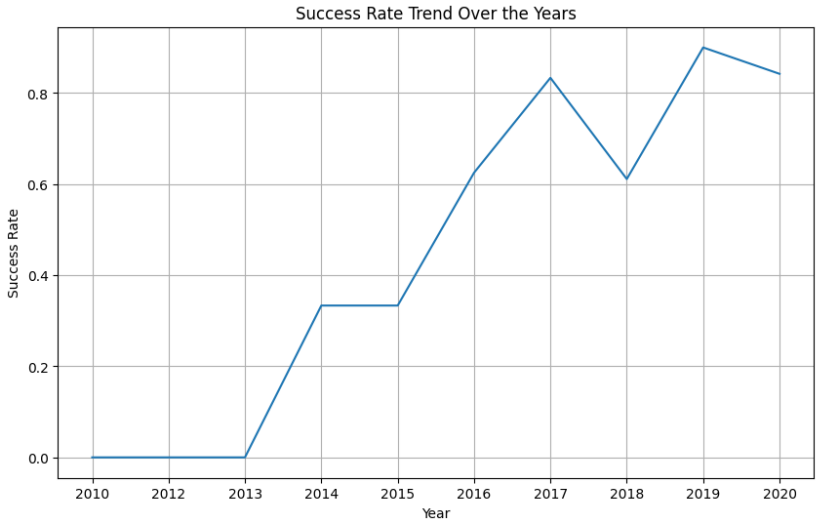
- Payload and Flight Number.
- Flight Number and Launch Site.
- Payload and Launch Site.
- Flight Number and Orbit Type.
- Payload and Orbit Type.

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.

https://github.com/16Brijesh10/APPLIED_DATA_SCIENCE_CAPSTONE/blob/main/week%202/Data%20visualization/5%20jupyter-labs-EDA-dataviz.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.

https://github.com/16Brijesh10/APPLIED_DATA_SCIENCE_CAPSTONE/blob/main/week%202/Data%20visualization/5%20jupyter-labs-EDA-dataviz.ipynb

EDA with SQL

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

- 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.

https://github.com/16Brijesh10/APPLIED_DATA_SCIENCE_CAPSTONE/blob/main/week%202/EDA%20with%20sql%20lite/4%20jupyter-labs-EDA-sql-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?

From:
https://github.com/16Brijesh10/APPLIED_DATA_SCIENCE_CAPSTONE/blob/main/week%203/Interactive%20Visual%20Analytics%20with%20Folium%20lab/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.

The link of the app.py::
https://github.com/16Brijesh10/APPLIED_DATA_SCIENCE_CAPSTONE/blob/main/week%203/spaceX%20dashboard/spacex_dash_plot.ipynb

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

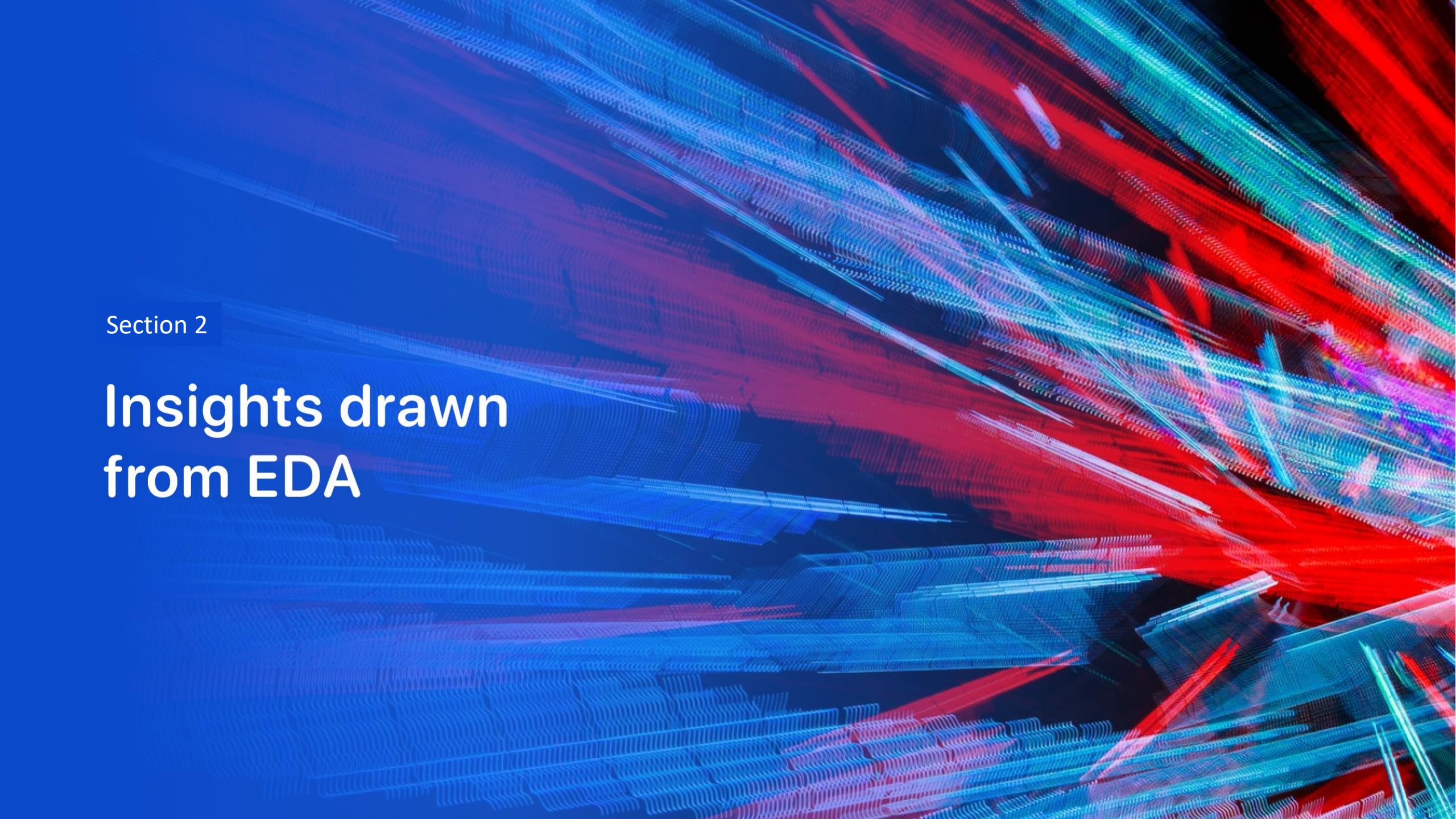
- The model with the best accuracy score will be the best performing model.

From:
https://github.com/16Brijesh10/APPLIED_DATA_SCIENCE_CAPSTONE/blob/main/week%204/paceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

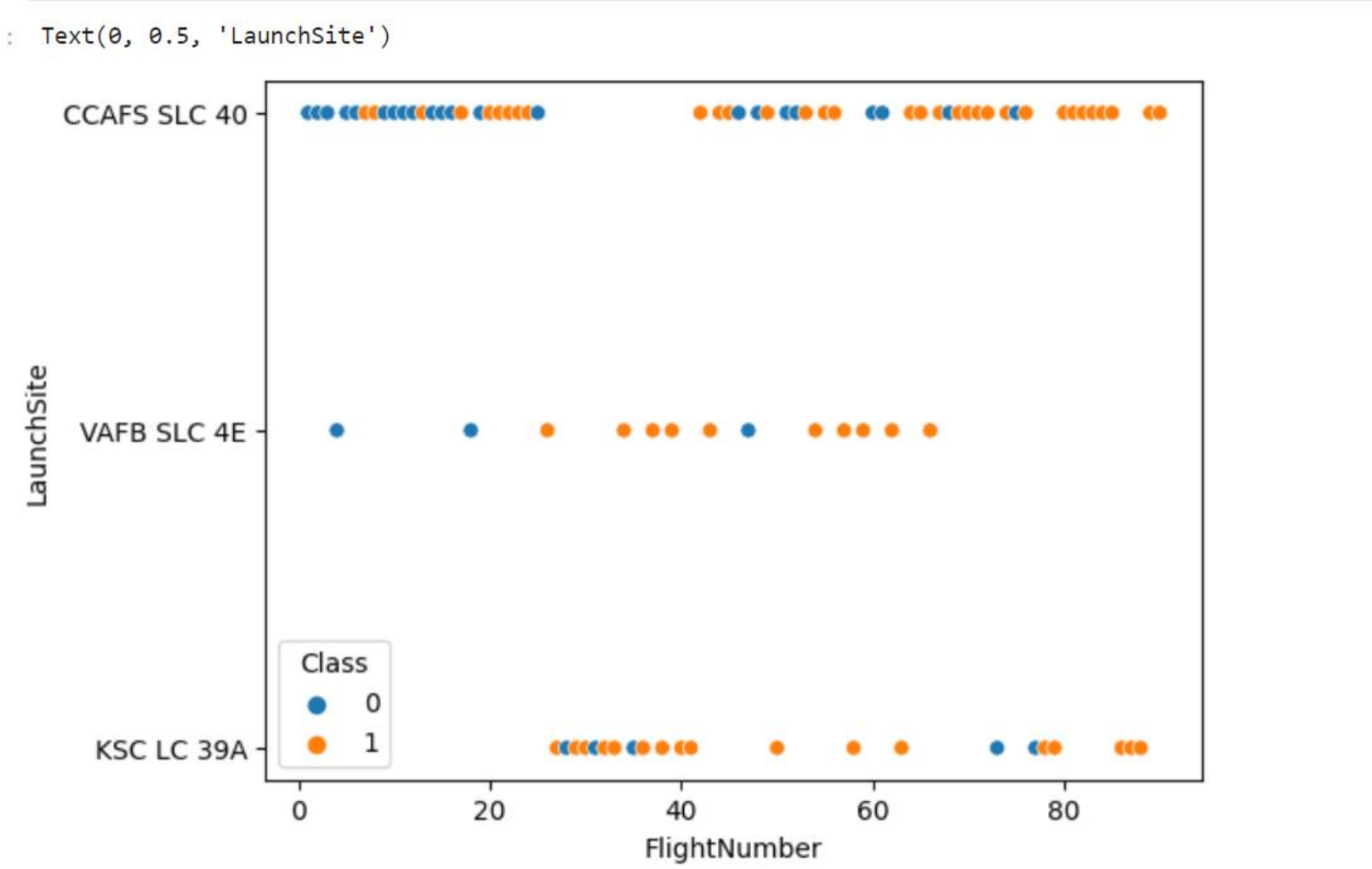
Results

The results will be categorized to 3 main results which is:

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


Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



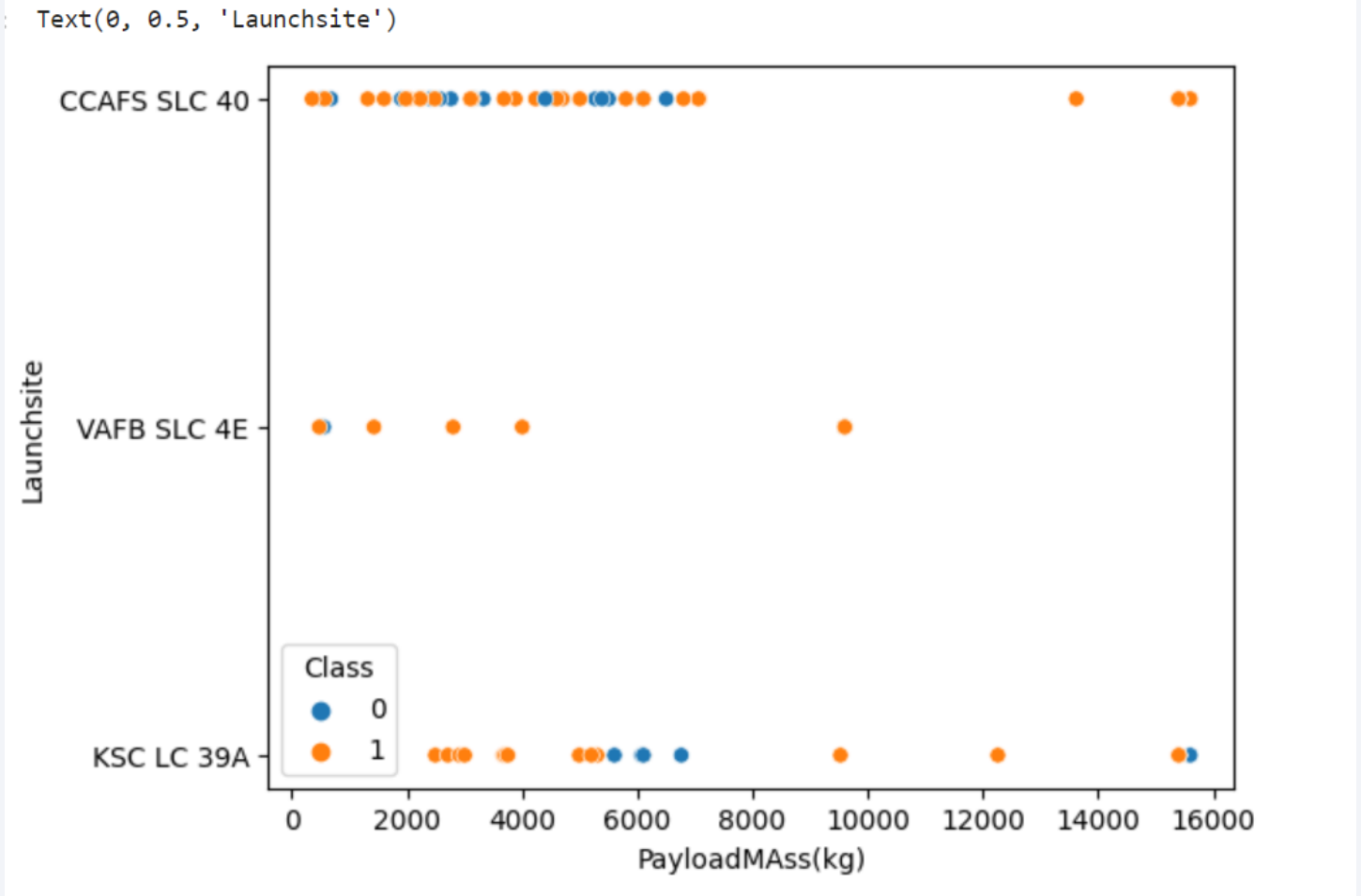
This scatter plot shows that the larger the flights amount of the launch site, the greater the the success rate will be.

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

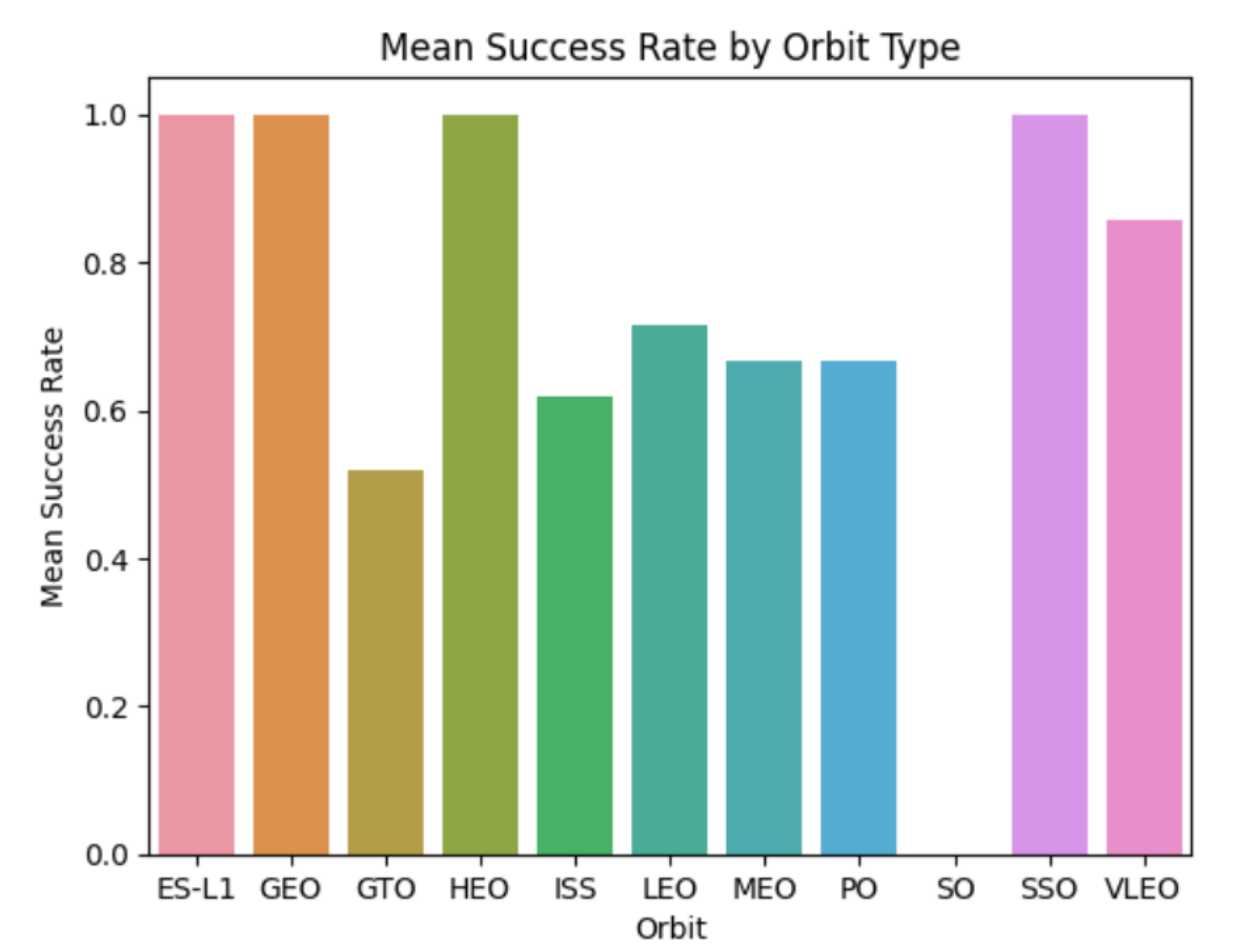
Payload vs. Launch Site

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



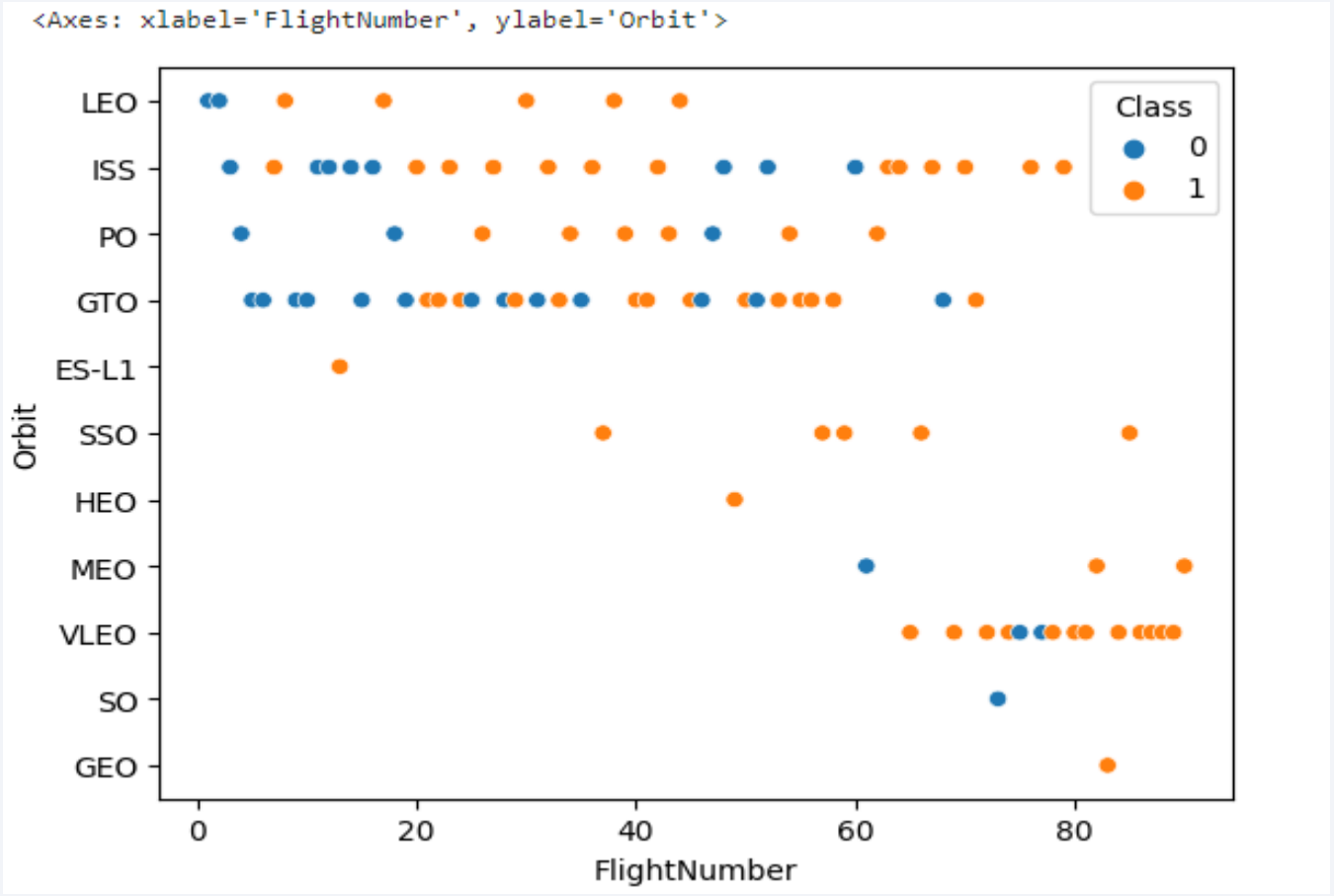
This figure depicted the possibility of the orbits to influences the landing outcomes as some orbits has 100% success rate such as SSO, HEO, GEO AND ES-L1 while SO orbit produced 0% rate of success.

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.

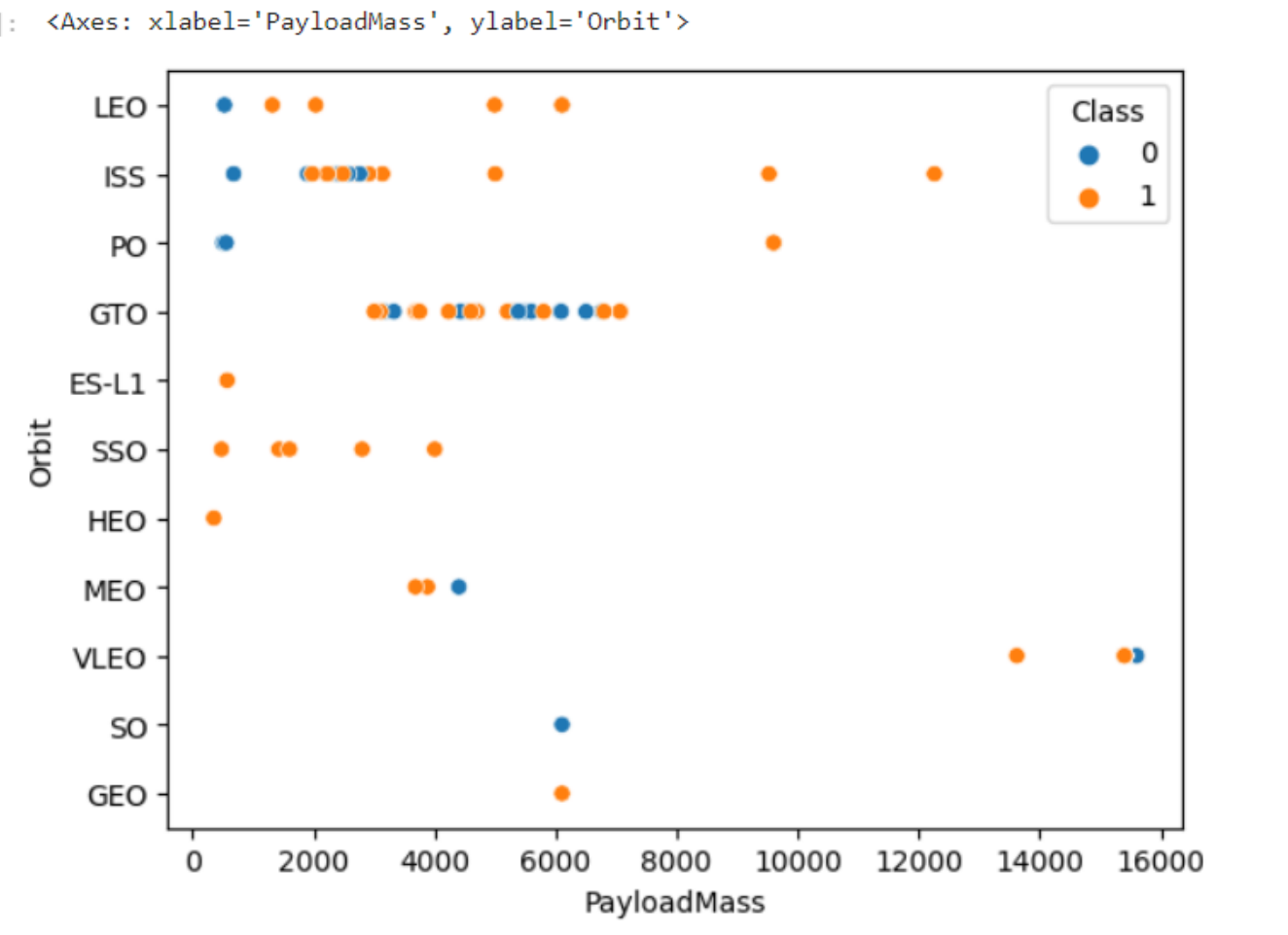
Flight Number vs. Orbit Type

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.

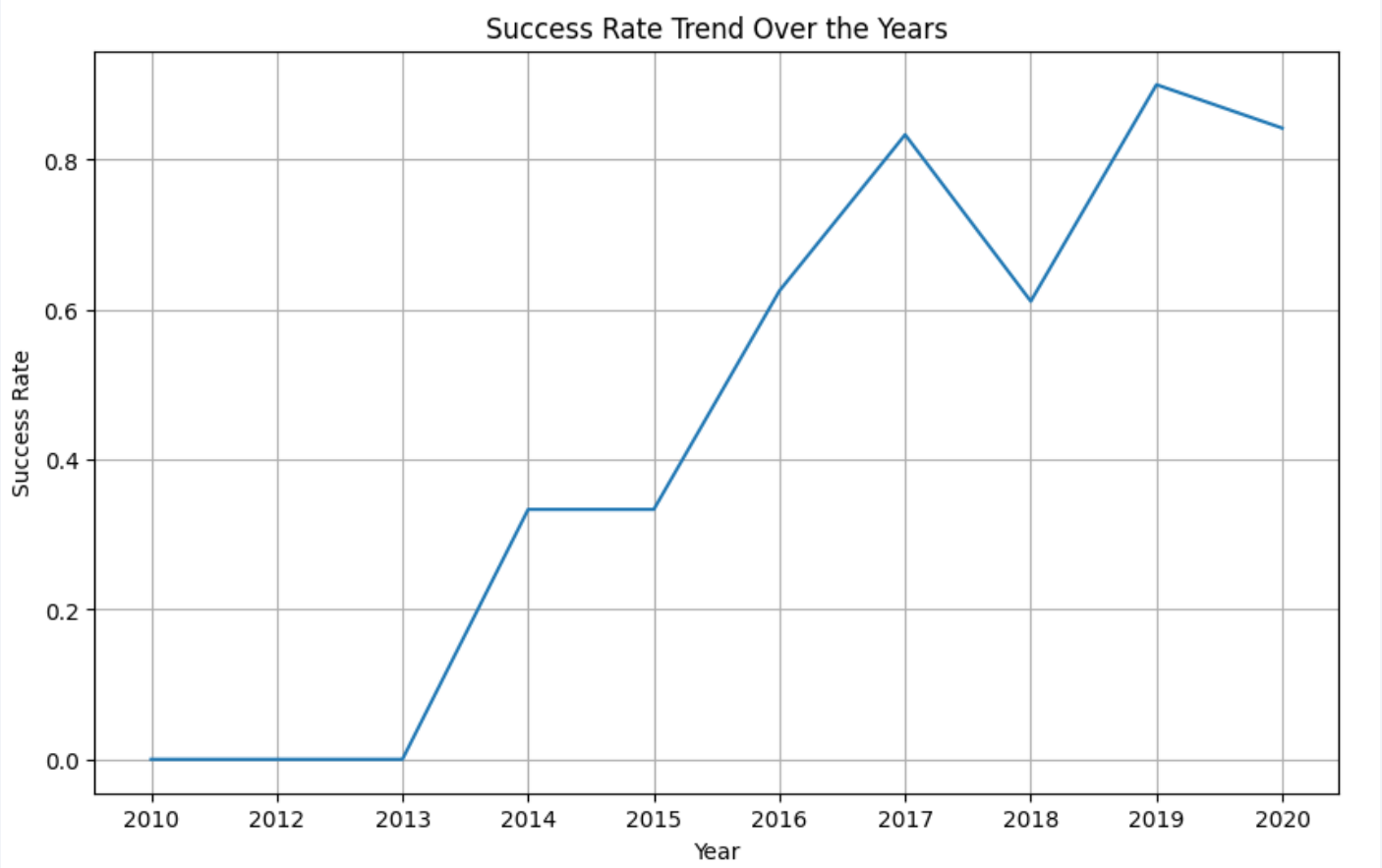
Meanwhile, again, SO, GEO and HEO orbit need more dataset to see any pattern or trend.

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 [32]:
```
pd.read_sql_query("select DISTINCT Launch_Site from SPACEXTABLE",con = con)
```

Out[32]:

	Launch_Site
0	CCAFS LC-40
1	VAFB SLC-4E
2	KSC LC-39A
3	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'

In [37]:
```
pd.read_sql_query("SELECT * FROM SPACEXTBL WHERE Launch_Site LIKE 'CCA%' LIMIT 5;",con=con)
```

Out[37]:

	Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcor
0	2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachu
1	2010-12-08	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 (parachu
2	2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attem
3	2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attem
4	2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attem

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)

In [40]:
```
pd.read_sql_query("SELECT SUM(PAYLOAD_MASS__KG_) AS TotalPayloadMass FROM SPACEXTBL WHERE Customer = 'NASA (CRS)'",con=con)
```

Out[40]:

	TotalPayloadMass
0	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

In [42]:
```
pd.read_sql_query("SELECT AVG(PAYLOAD_MASS__KG_) AS AVGPAYLOAD FROM SPACEXTBL WHERE Booster_Version = 'F9 v1.1'",con = con)
```

Out[42]:

	AVGPAYLOAD
0	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

List the date when the first succesful landing outcome in ground pad was acheived.

Hint:Use min function

In [44]:

```
pd.read_sql_query("""
    SELECT MIN(Date) AS EarliestSuccessfulLandingDate
    FROM SPACEXTBL
    WHERE Landing_Outcome = 'Success (ground pad)';
""",con=con)
```

Out[44]:

	EarliestSuccessfulLandingDate
0	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

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

In [49]:

```
pd.read_sql_query("""
    SELECT Booster_Version
    FROM SPACEXTBL
    WHERE Landing_Outcome = 'Success (drone ship)'
    AND PAYLOAD_MASS__KG_ > 4000
    AND PAYLOAD_MASS__KG_ < 6000;
""",con=con)
```

Out[49]:

	Booster_Version
0	F9 FT B1022
1	F9 FT B1026
2	F9 FT B1021.2
3	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
Successful = 100
Unsuccessful = 1.

List the total number of successful and failure mission outcomes

In [53]:
```
pd.read_sql_query("""
    SELECT Mission_Outcome, COUNT(*) AS Total
    FROM SPACEXTBL
    GROUP BY Mission_Outcome;
""",con=con)
```

Out[53]:

	Mission_Outcome	Total
0	Failure (in flight)	1
1	Success	98
2	Success	1
3	Success (payload status unclear)	1

Boosters Carried Maximum Payload

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

In [55]:
```
pd.read_sql_query("""
    SELECT Booster_Version
    FROM SPACEXTBL
    WHERE PAYLOAD_MASS__KG_ = (
        SELECT MAX(PAYLOAD_MASS__KG_)
        FROM SPACEXTBL
    );
""",con=con)
```

Out[55]:

	Booster_Version
0	F9 B5 B1048.4
1	F9 B5 B1049.4
2	F9 B5 B1051.3
3	F9 B5 B1056.4
4	F9 B5 B1048.5
5	F9 B5 B1051.4
6	F9 B5 B1049.5
7	F9 B5 B1060.2
8	F9 B5 B1058.3
9	F9 B5 B1051.6
10	F9 B5 B1060.3
11	F9 B5 B1049.7

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

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

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, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

In [57]:

```
pd.read_sql_query("""
    SELECT
        CASE
            WHEN substr(Date, 6, 2) = '01' THEN 'January'
            WHEN substr(Date, 6, 2) = '02' THEN 'February'
            WHEN substr(Date, 6, 2) = '03' THEN 'March'
            WHEN substr(Date, 6, 2) = '04' THEN 'April'
            WHEN substr(Date, 6, 2) = '05' THEN 'May'
            WHEN substr(Date, 6, 2) = '06' THEN 'June'
            WHEN substr(Date, 6, 2) = '07' THEN 'July'
            WHEN substr(Date, 6, 2) = '08' THEN 'August'
            WHEN substr(Date, 6, 2) = '09' THEN 'September'
            WHEN substr(Date, 6, 2) = '10' THEN 'October'
            WHEN substr(Date, 6, 2) = '11' THEN 'November'
            WHEN substr(Date, 6, 2) = '12' THEN 'December'
        END AS Month,
        Landing_Outcome,
        Booster_Version,
        Launch_Site
    FROM SPACEXTBL
    WHERE substr(Date, 0, 5) = '2015'
    AND Landing_Outcome LIKE 'Failure%'
    AND Landing_Outcome LIKE '%drone ship%';
""",con=con)
```

Out[57]:

	Month	Landing_Outcome	Booster_Version	Launch_Site
0	January	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
1	April	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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.

In [59]:
```
pd.read_sql_query("""
    SELECT Landing_Outcome, COUNT(*) AS OutcomeCount
    FROM SPACEXTBL
    WHERE Date >= '2010-06-04' AND Date <= '2017-03-20'
    GROUP BY Landing_Outcome
    ORDER BY OutcomeCount DESC;
""",con=con)
```

Out[59]:

	Landing_Outcome	OutcomeCount
0	No attempt	10
1	Success (drone ship)	5
2	Failure (drone ship)	5
3	Success (ground pad)	3
4	Controlled (ocean)	3
5	Uncontrolled (ocean)	2
6	Failure (parachute)	2
7	Precluded (drone ship)	1

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.

Section 3

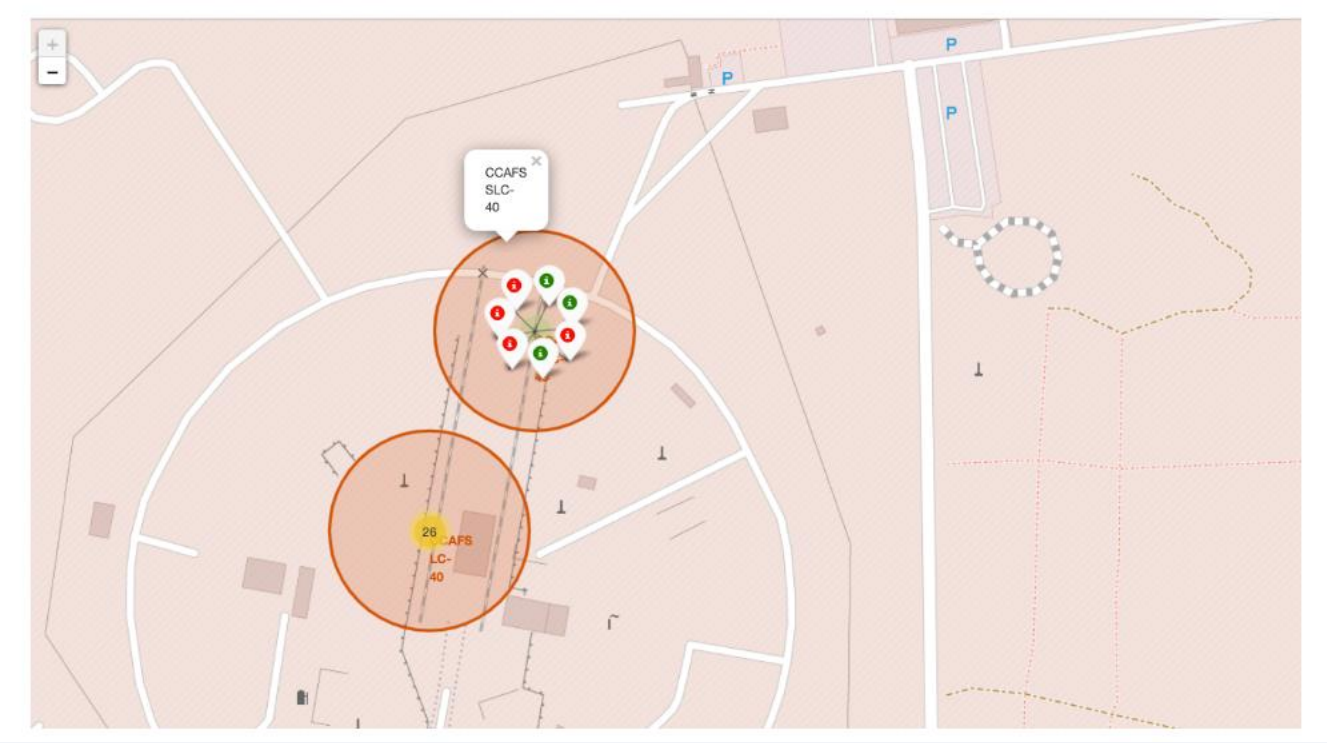
Launch Sites Proximities Analysis

Location of all the Launch Sites

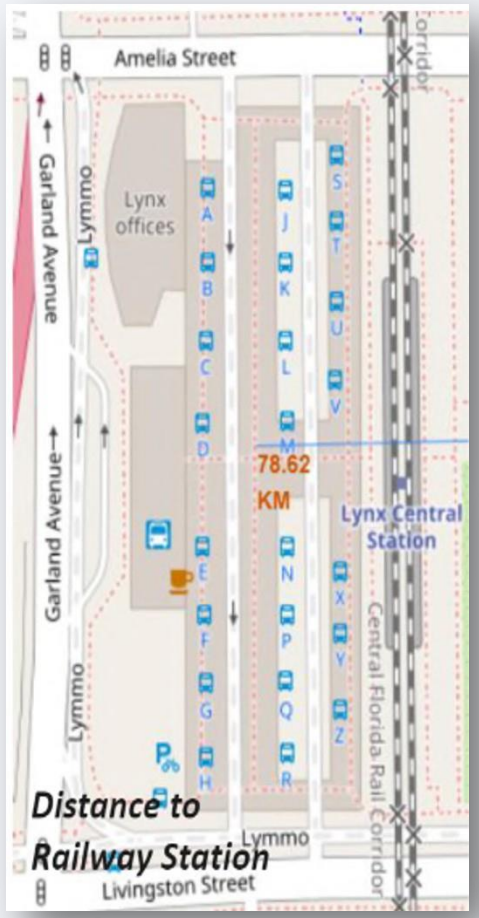
The generated map with marked launch sites should look similar to the following:

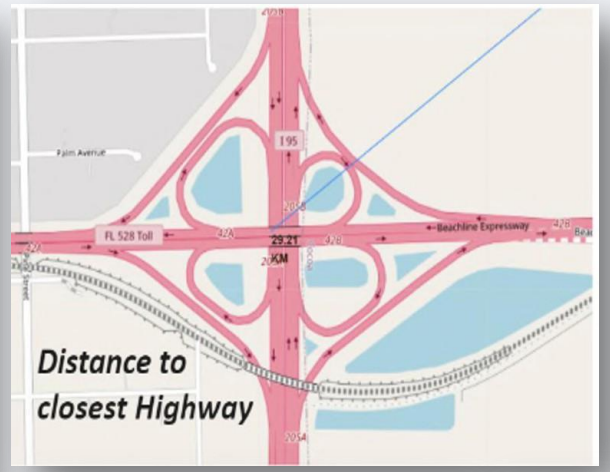
We can see that all the SpaceX launch sites are located inside the United States

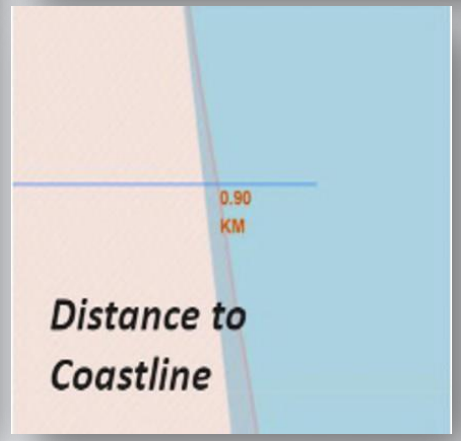
Markers showing launch sites with color labels



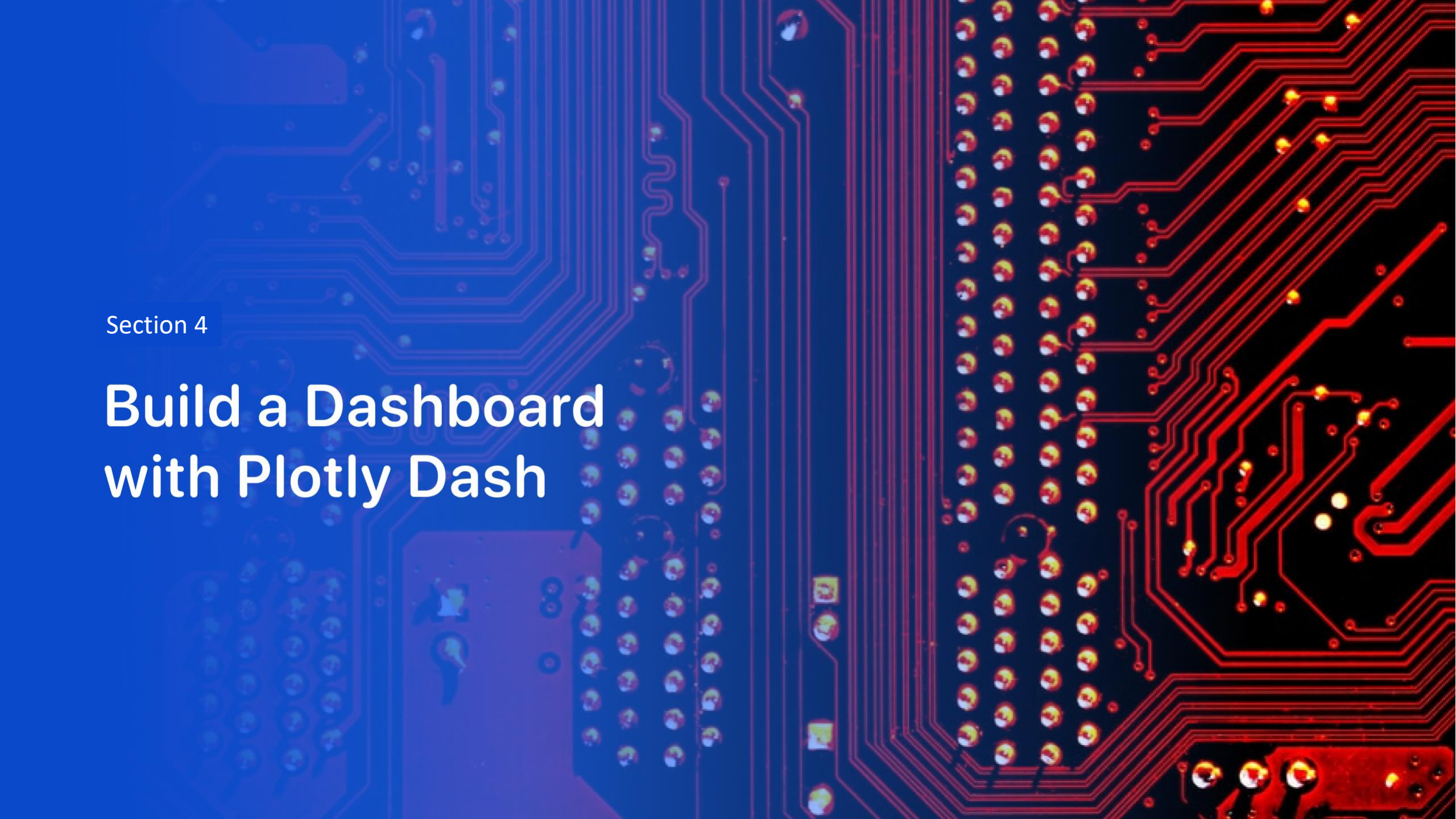
Launch Sites Distance to Landmarks



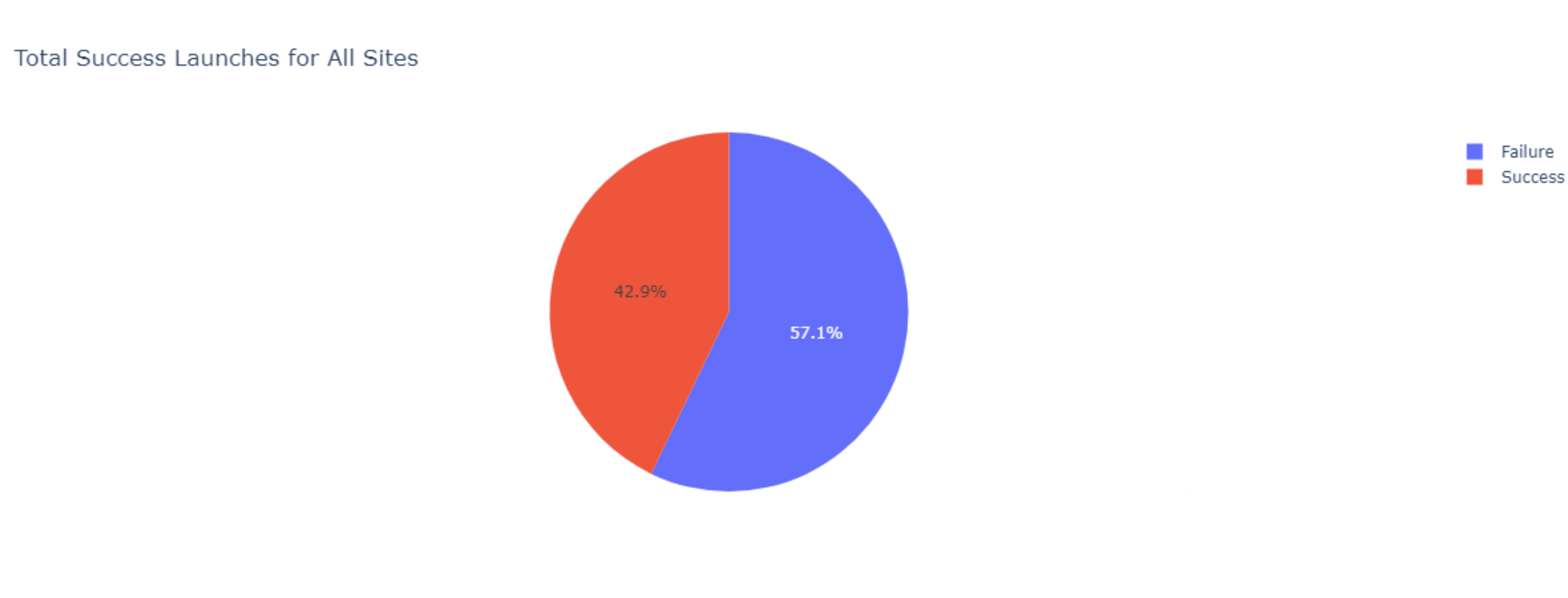




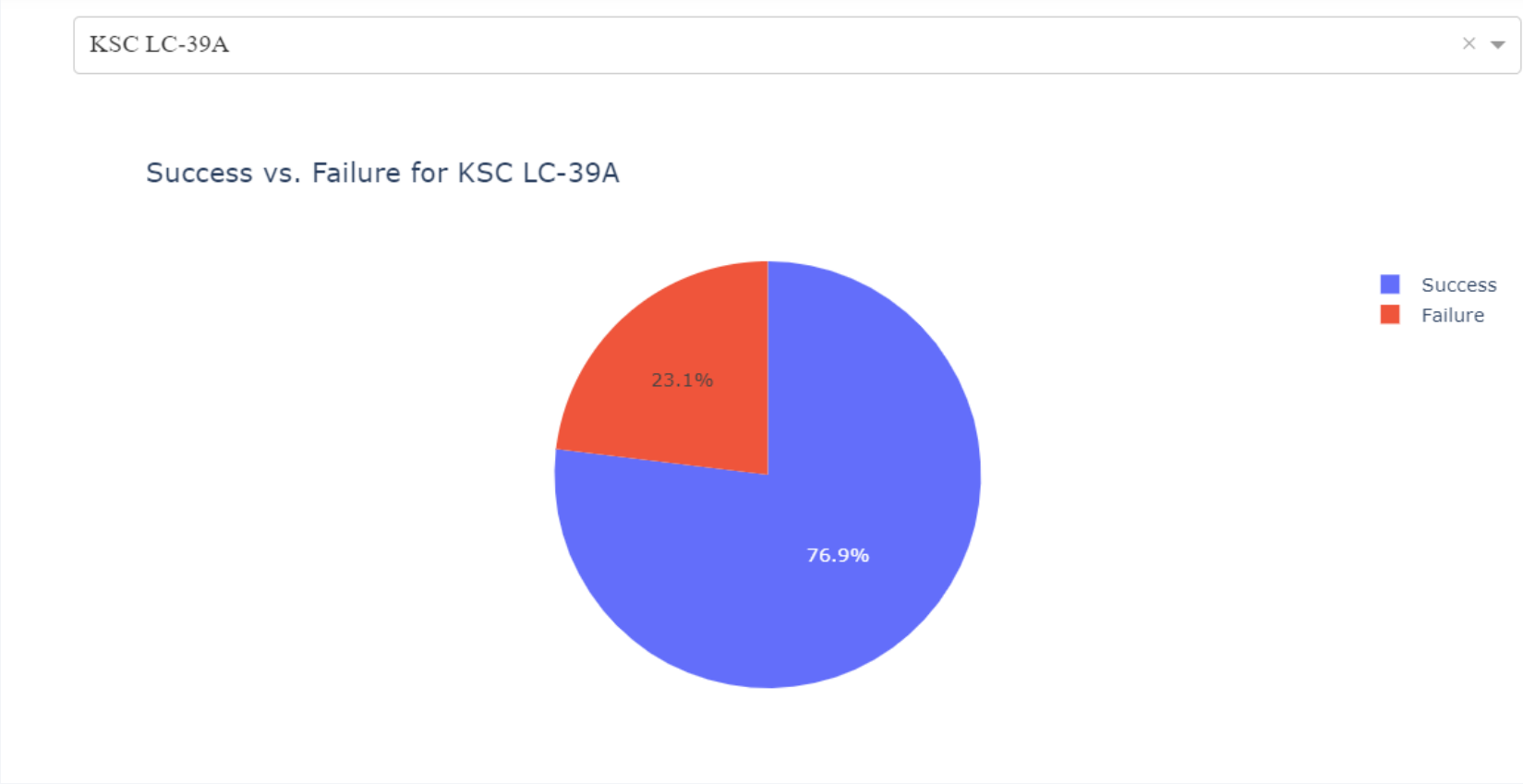
- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes


Section 4
Build a Dashboard with Plotly Dash

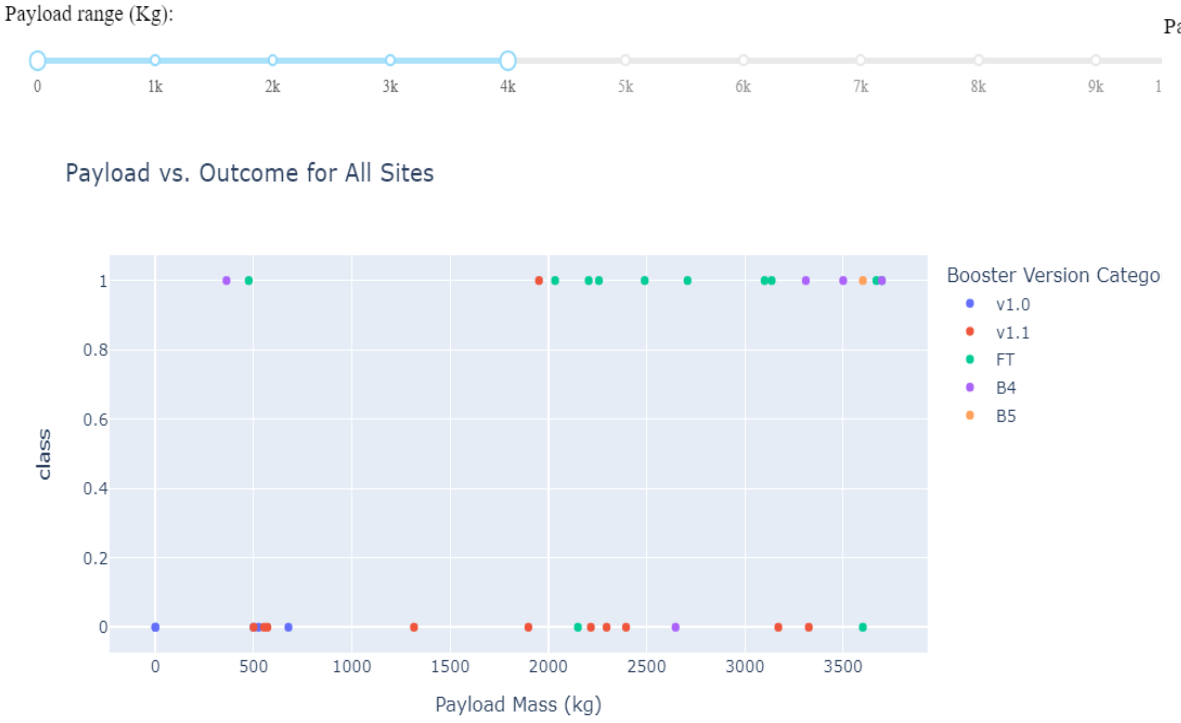
The success percentage by each sites.



The highest launch-success ratio: KSC LC-39A



Payload vs Launch Outcome Scatter Plot



Section 5

Predictive Analysis (Classification)

Classification Accuracy

Find the method performs best:

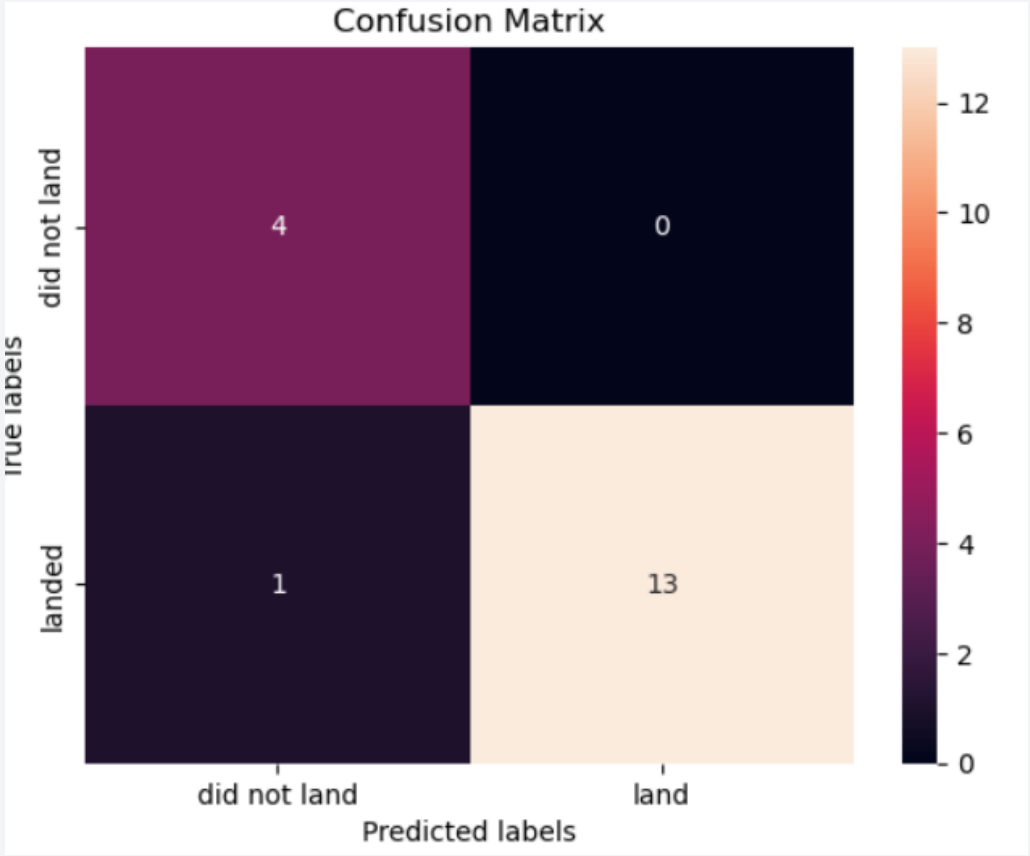
[46]:

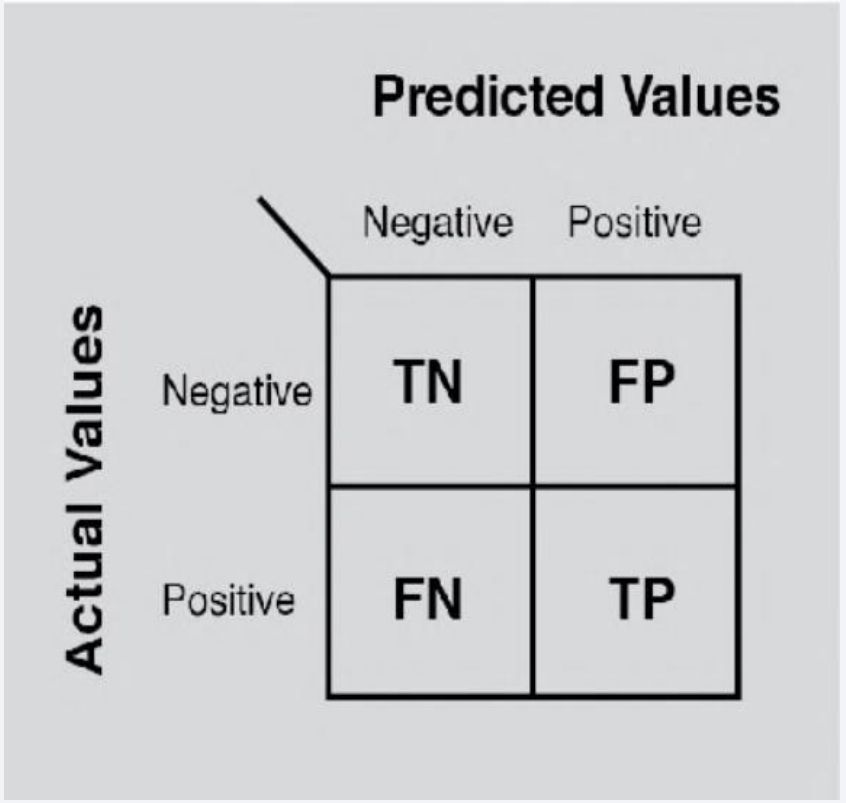
```
# Create a dictionary to store accuracy scores
accuracy_scores = {
    'Logistic Regression': accu,
    'Support Vector Machine': accc,
    'Decision Tree': acc,
    'K-Nearest Neighbors': acc
}

# Find the classifier with the highest accuracy
best_classifier = max(accuracy_scores, key=accuracy_scores.get)
best_accuracy = accuracy_scores[best_classifier]

# Print the results
print("Accuracy scores:")
for classifier, score in accuracy_scores.items():
    print(f"{classifier}: {score}")

print("\nBest performing classifier:")
print(f"{best_classifier}: {best_accuracy}")
```

```
Accuracy scores:
Logistic Regression: 0.8888888888888888
Support Vector Machine: 0.7222222222222222
Decision Tree: 0.9444444444444444
K-Nearest Neighbors: 0.9444444444444444

Best performing classifier:
Decision Tree: 0.9444444444444444
```

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.





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

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!