IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Kumaresh Manoharan
June 23, 2025

OUTLINE

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

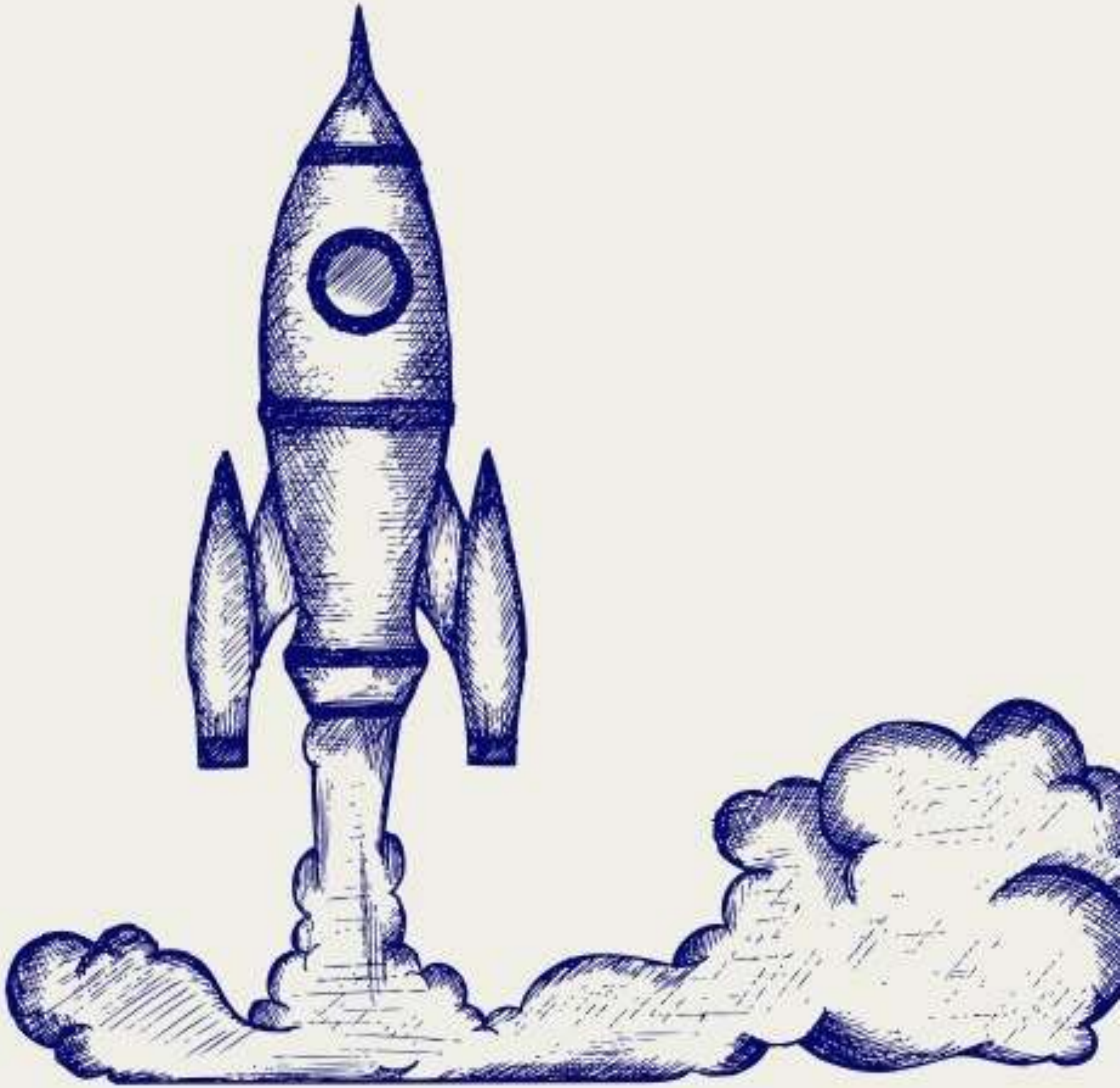
EXECUTIVE SUMMARY

- This project seeks to predict whether or not the first stage of Space X Falcon 9 will land successfully. Data was collected from Space X API ensuring completeness and accuracy. The process involved sending a get request to the API. This approach gathered relevant datapoints to provide a holistic view of the subject matter.
- The analysis revealed several key trends and patterns within the data. It was observed from various plots that ES-LI, GEO and HEO orbits have higher probability of successful landing with increasing flight number. It was also noted that Polar, LEO and ISS orbits with high payload mass and increasing flight number are more likely to land successfully. Moreover, flight number was seen to be highly correlated with successful landing.

INTRODUCTION

• **Project background and context:** working in SpaceX company, I want to predict the success or failure landing of Falcon 9 first stage. It’s important to know if the rockets will land successfully or not because the failure will cost the company much resources.

•**Statement of the problem:** this project seeks to predict whether or not the Falcon 9 first stage will land successfully



Section 1

Methodology

METHODOLOGY

EXECUTIVE SUMMARY

- **Data collection methodology:**
  Data was collected from Space X API ensuring completeness and accuracy. The process involved sending a GET request to the API.
- **Data wrangling:**
  The data was cleaned by handling missing values, removing duplicates, and correcting errors. A new column for landing outcome was also created from the existing outcome column.
- **Exploratory Data Analysis (EDA):**
  Used visualization and SQL to discover patterns and trends within the dataset.
- **Interactive visual analytics:**
  Built using **Folium** for mapping and **Plotly Dash** for interactive dashboard creation.
- **Predictive analysis using classification models:**
  Conducted by:

1. Creating a column for classification
2. Standardizing the data
3. Splitting into training and testing sets
4. Finding best hyperparameters for **SVM**, **Decision Trees**, **KNN**, and **Logistic Regression**
5. Evaluating model performance on test data

Data Collection

Data sets were collected using the API call from several websites, I collected rocket, launchpad, payloads, and cores data from https://api.spacexdata.com/v4 website.

1. Collecting the data with API call

2. Converting to data frame with help of JSON

3. Updating columns and rows (pre-processing)

4. Filtering the data to keep only Falcon 9 launches

5. Convert the data to csv file with name 'dataset_part_1.csv'

Data Collection – SpaceX API

1.Collecting the Data with API call

In [3]:
```
# Takes the dataset and uses the rocket column to call the API and append the data to the list
def getBoosterVersion(data):
    for x in data['rocket']:
       if x:
        response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
        BoosterVersion.append(response['name'])
```

2.Converting into DataFrame with JSON

In [13]:
```
# Use json_normalize meethod to convert the json result into a dataframe
data = response.json()
data = pd.json_normalize(data)
```

3. Updating Rows and Columns (pre-processing)

In [15]:
```
# 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
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)]
```

4. Filtering the data to keep only Falcon 9 launches

In [26]:
```
# Hint data['BoosterVersion']!='Falcon 1'
indexnames = launch_df.loc[launch_df['BoosterVersion'] != 'Falcon 9'].index
launch_df.drop(indexnames, inplace = True)
data_falcon9 = data
```

5. Convert the data to csv file with name

In [ ]:
```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

GitHub repo https://github.com/Kumaresh07-git/DataScience-Capstone-project

Data Collection – Scraping

1.Creating BeautifulSoap object

In [9]:
```python
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(html_content, 'html.parser')
```

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

In [10]:
```python
# Use soup.title attribute
print(soup.title.string)
```

2.Getting column names

In [16]:
```python
column_names = []

# Apply find_all() function with `th` element on first_launch_table
th_elements = first_launch_table.find_all('th')
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
for th in th_elements:
    name = extract_column_from_header(th)  # Extract the column name
    if name is not None and len(name) > 0:  # Check if the name is non-empty
        column_names.append(name)
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names
```

Check the extracted column names

In [17]:
```python
print(column_names)
```

```
['Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome']
```

3. Creating the launch_dict

In [22]:
```python
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:
            extracted_row += 1
            # Flight Number value
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            launch_dict['Flight No.'].append(flight_number) #jgTODO-1
            #print(flight_number)
            datatimelist=date_time(row[0])
```

4.Converting to final Data Frame

In [23]:
```python
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
df
```

Out[23]:

	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date	Time
0	1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success\n	F9 v1.07B0003.18	Failure	4 June 2010	18:45
1	2	CCAFS	Dragon	0	LEO	NASA (COTS)\nNRO	Success	F9 v1.07B0004.18	Failure	8 December 2010	15:43
2	3	CCAFS	Dragon	525 kg	LEO	NASA (COTS)	Success	F9 v1.07B0005.18	No attempt\n	22 May 2012	07:44
3	4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA (CRS)	Success\n	F9 v1.07B0006.18	No attempt	8 October 2012	00:35
4	5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA (CRS)	Success\n	F9 v1.07B0007.18	No attempt\n	1 March 2013	15:10

GitHub repo https://github.com/Kumaresh07-git/DataScience-Capstone-project

Data Wrangling

1. Loading the Data set

In [3]:
```
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/datas
df.head(10)
```

2.Creating Landing outcomes

In [9]:
```
# landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()
landing_outcomes
```

Out[9]:
```
Outcome
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: count, dtype: int64
```

3. Finding Bad outcomes

In [11]:
```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

Out[11]: `{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}`

4. Presenting outcomes as 0 and 1

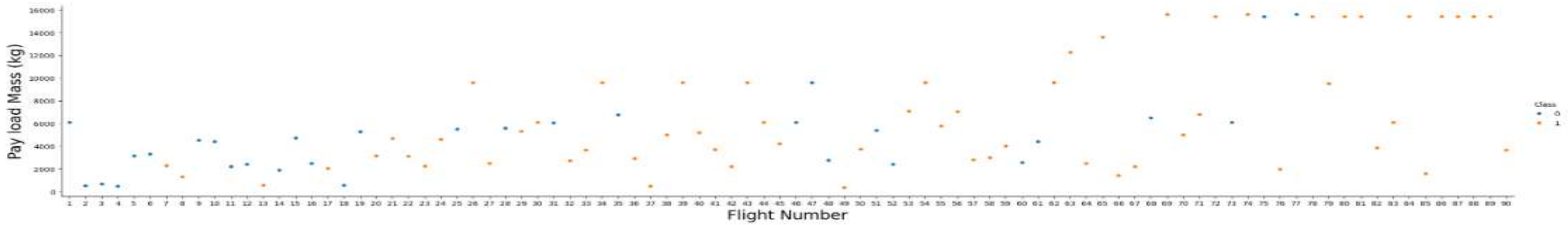
In [13]:
```
df['Class']=landing_class
df[['Class']].head(8)
```

Out[13]:

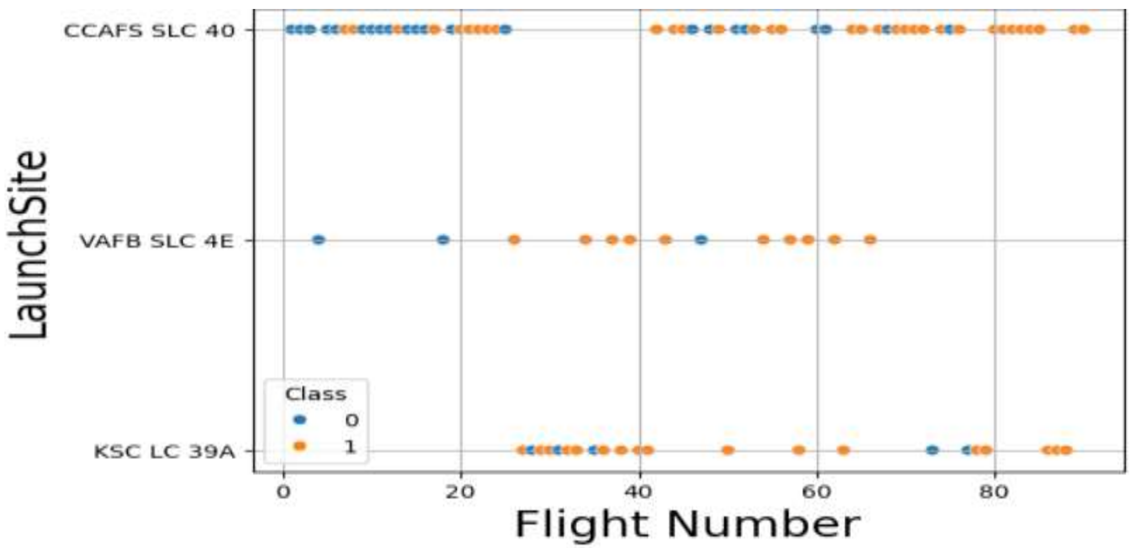
	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0

GitHub repo https://github.com/Kumaresh07-git/DataScience-Capstone-project

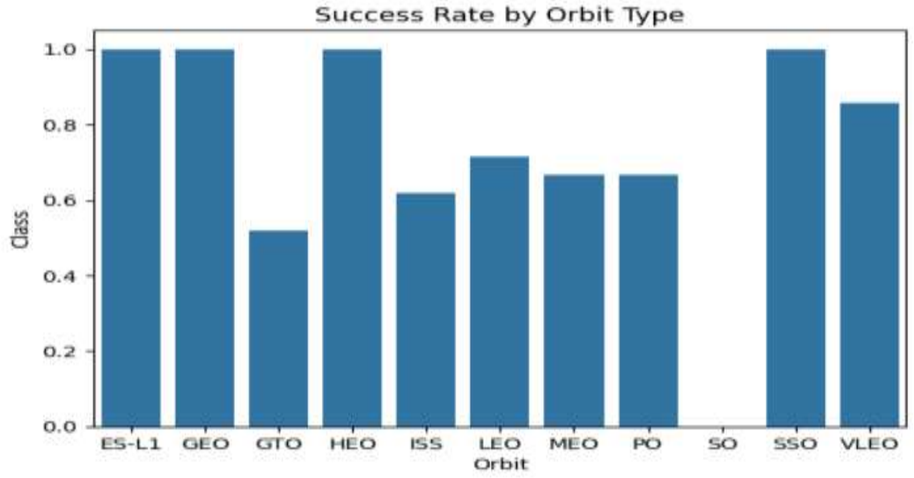
EDA with Visualization



Categorial plot between Pay load mass (kg) and Flight number



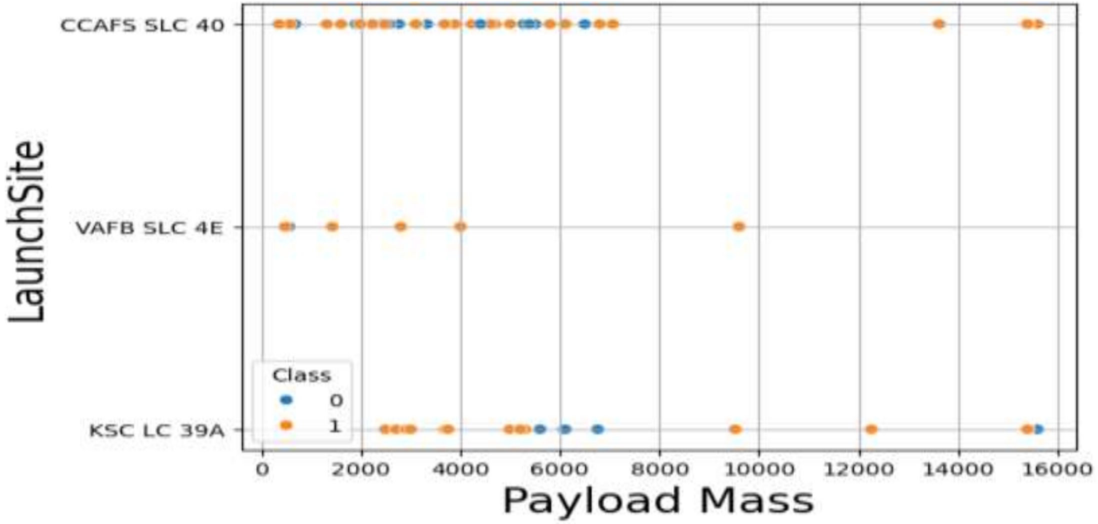
Scatter plot between Launch site and Flight number



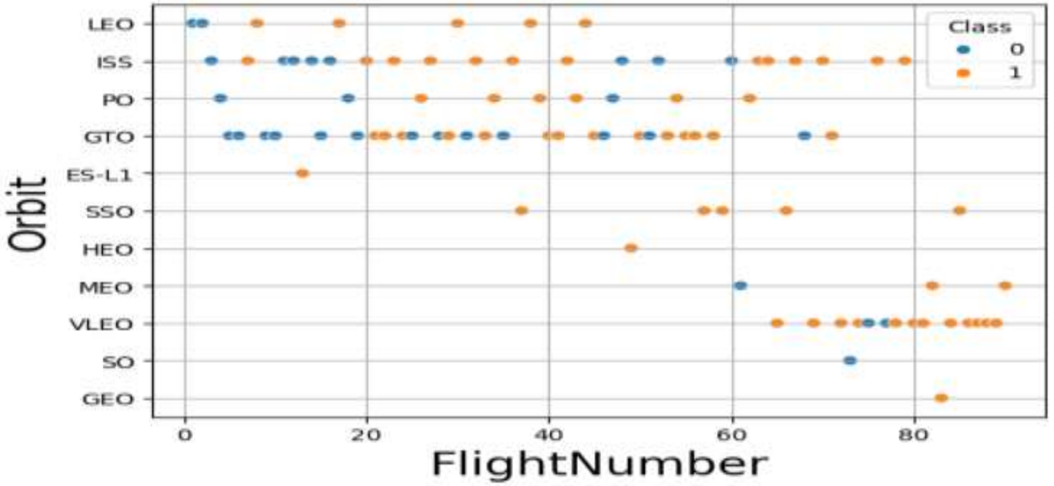
Bar chart of success rate of each orbit type

GitHub repo https://github.com/Kumaresh07-git/DataScience-Capstone-project

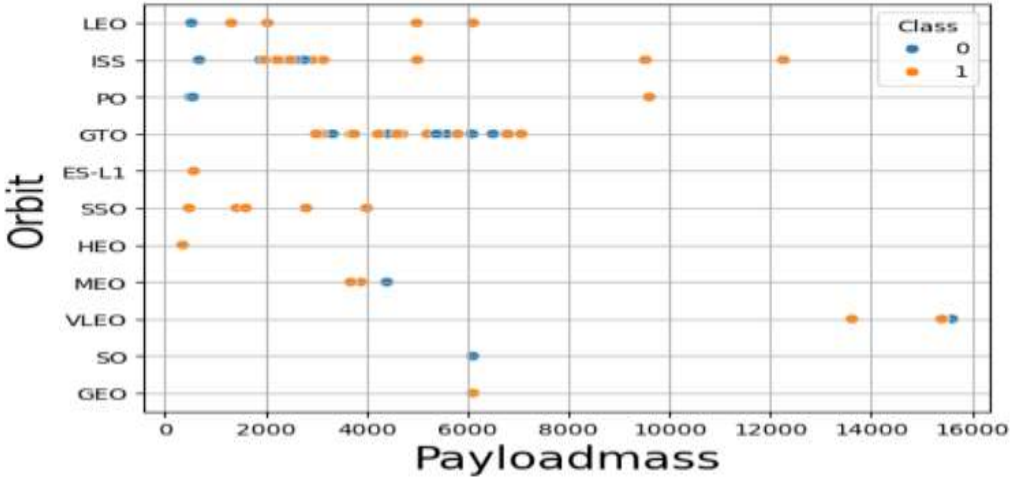
EDA with Visualization

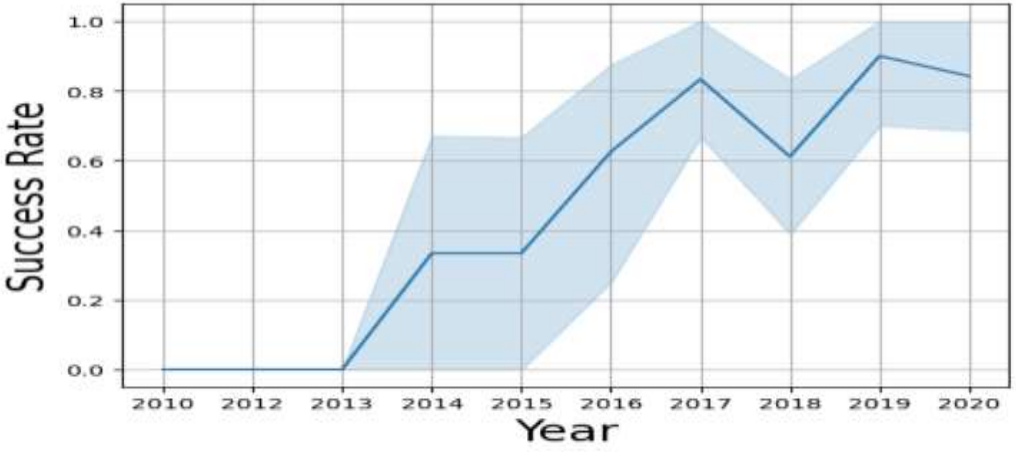


Scatter plot between Launch site and Payload mas



Scatter plot between Orbit type and Flight number





Line chart between success rate and Year

Scatter plot between Orbit type and Payload mass

GitHub repo https://github.com/Kumaresh07-git/DataScience-Capstone-project

EDA with SQL

I used SQL queries to answer the following questions:

- Display the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string 'CCA'
- Display the total payload mass carried by boosters launched by NASA (CRS)
- Display average payload mass carried by booster version F9 v1.1
- List the date when the first successful landing outcome in-ground pad was achieved
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- List the total number of successful and failure mission outcomes
- List the names of the booster versions which have carried the maximum payload mass. Use a subquery
- List the failed landing outcomes in drone ship, their booster versions, and launch site names for the months in year 2015
- 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

GitHub repo https://github.com/Kumaresh07-git/DataScience-Capstone-project

Build an Interactive Map with Folium

- folium.Marker() was used to create marks on the maps.
- folium.Circle() was used to create a circles above markers on the map.
- folium.PolyLine() was used to create polynomial line between the points.
- folium.plugins.AntPath() was used to create animated line between the points.
- markerCluster() was used to simplify the maps which contain several markers with identical coordinates
- Calculated the distances between a launch site to its Proximities

GitHub repo https://github.com/Kumaresh07-git/DataScience-Capstone-project

Build an Dashboard with Plotly Dash

- Dash and html components were used as almost everything depends on them, such as graphs, tables, dropdowns, etc.
- Pandas was used to simplify the work by creating dataframe.
- Plotly was used to plot the graphs.
- Pie chart and scatter chart were used for plotting purposes.
- Rangeslider was used for payload mass range selection.
- Dropdown was used for launch sites selection.

GitHub repo https://github.com/Kumaresh07-git/DataScience-Capstone-project

Predictive Analysis (Classification)

1. Building the model

Create column for the class

Standardize the data

Split the data info train and test sets

Build GridSearchCV model and fit the data

2. Evaluating the model

Calculating the accuracies

Calculating the confusion matrixes

Plot the results

3. Finding the optimal model

Find the best hyperparameters for the models

Find the best model with highest accuracy

Confirm the optimal model

GitHub repo https://github.com/Kumaresh07-git/DataScience-Capstone-project

Results

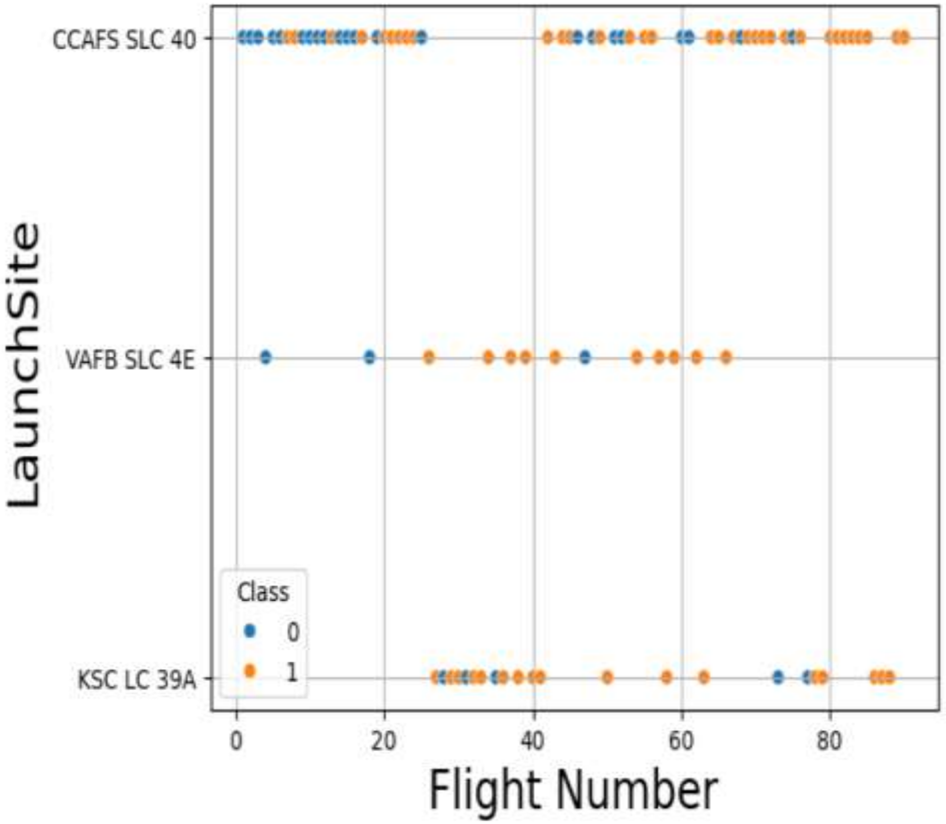
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

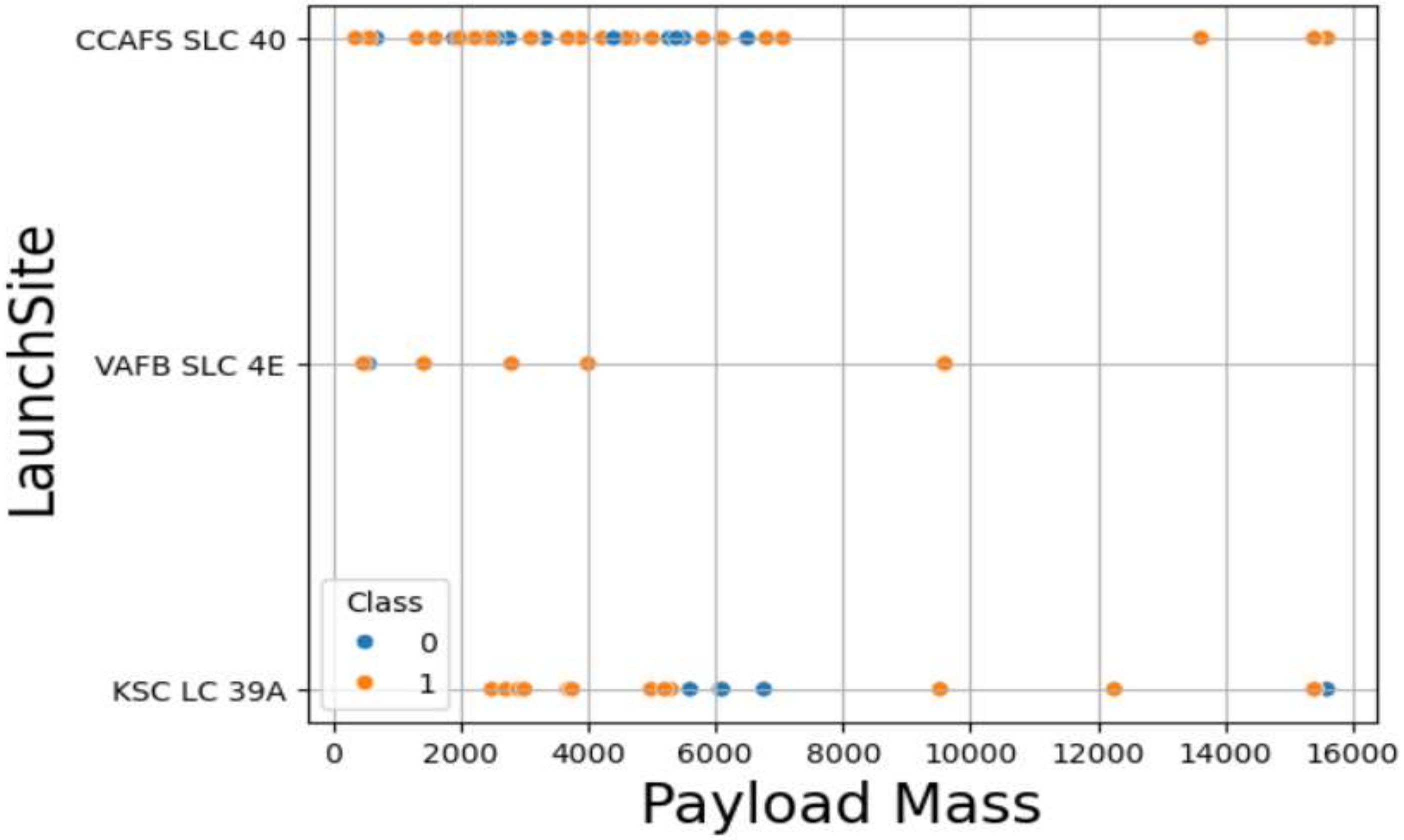
```
In [9]: # Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class va
        sns.scatterplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df)
        plt.xlabel("Flight Number",fontsize=20)
        plt.ylabel("LaunchSite",fontsize=20)
        plt.grid(True)
        plt.show()
```



From the plot, we find for the VAFB-SLC launchsite there are no rockets launched for high flight number (greater than 70). However, we can see an overall increase in success rate for all Launch sites with increasing flight number.

Payload vs. Launch Site

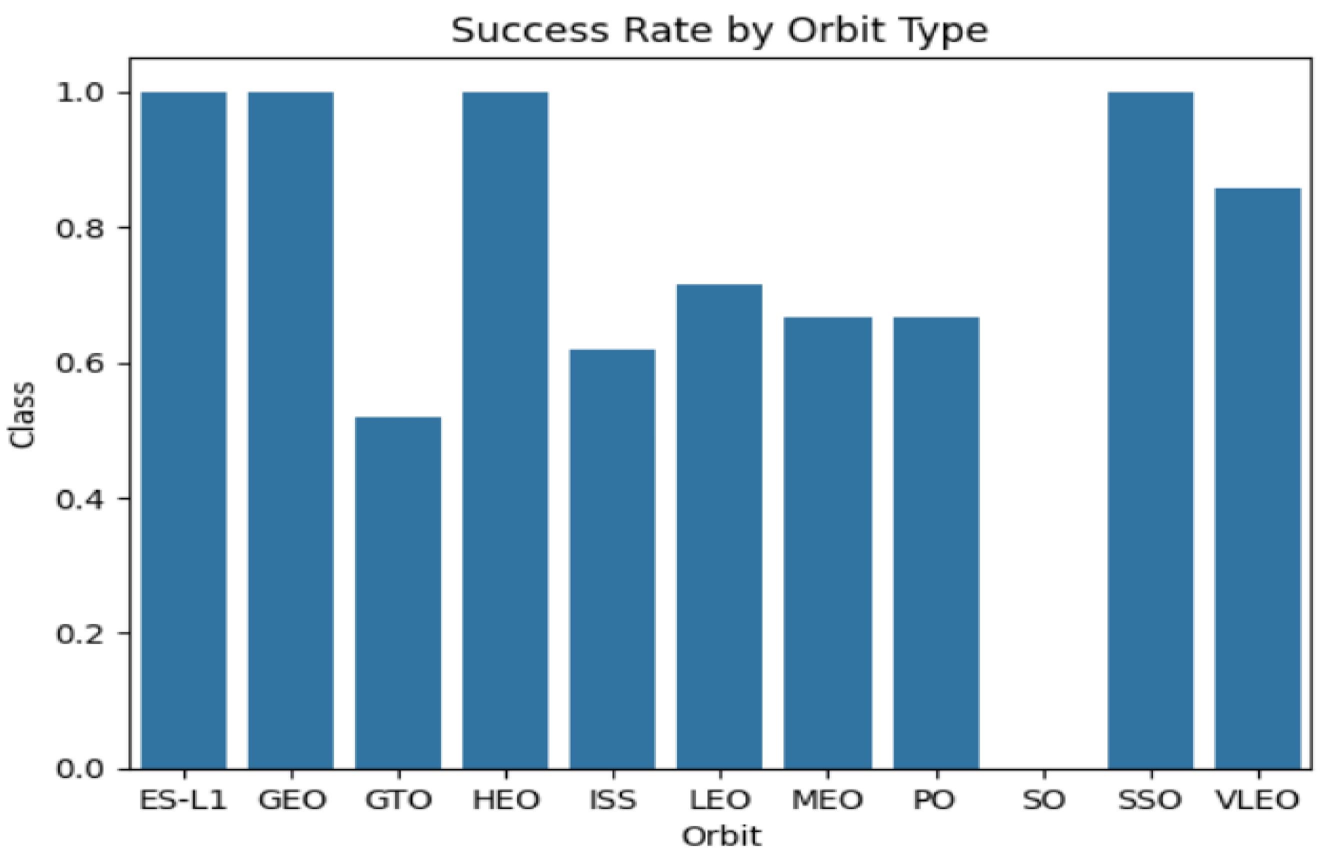
```
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch s
sns.scatterplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df)
plt.xlabel("Payload Mass",fontsize=20)
plt.ylabel("LaunchSite",fontsize=20)
plt.grid(True)
plt.show()
```



If you observe the Payload Mass Vs. Launch Site scatter point chart you will find for the VAFB-SLC launch site there are no rockets launched for heavy payload mass (greater than 10000).
However, there is an observed increase in success rate for all Launch sites with increasing payload mass.

Success Rate vs. Orbit Type

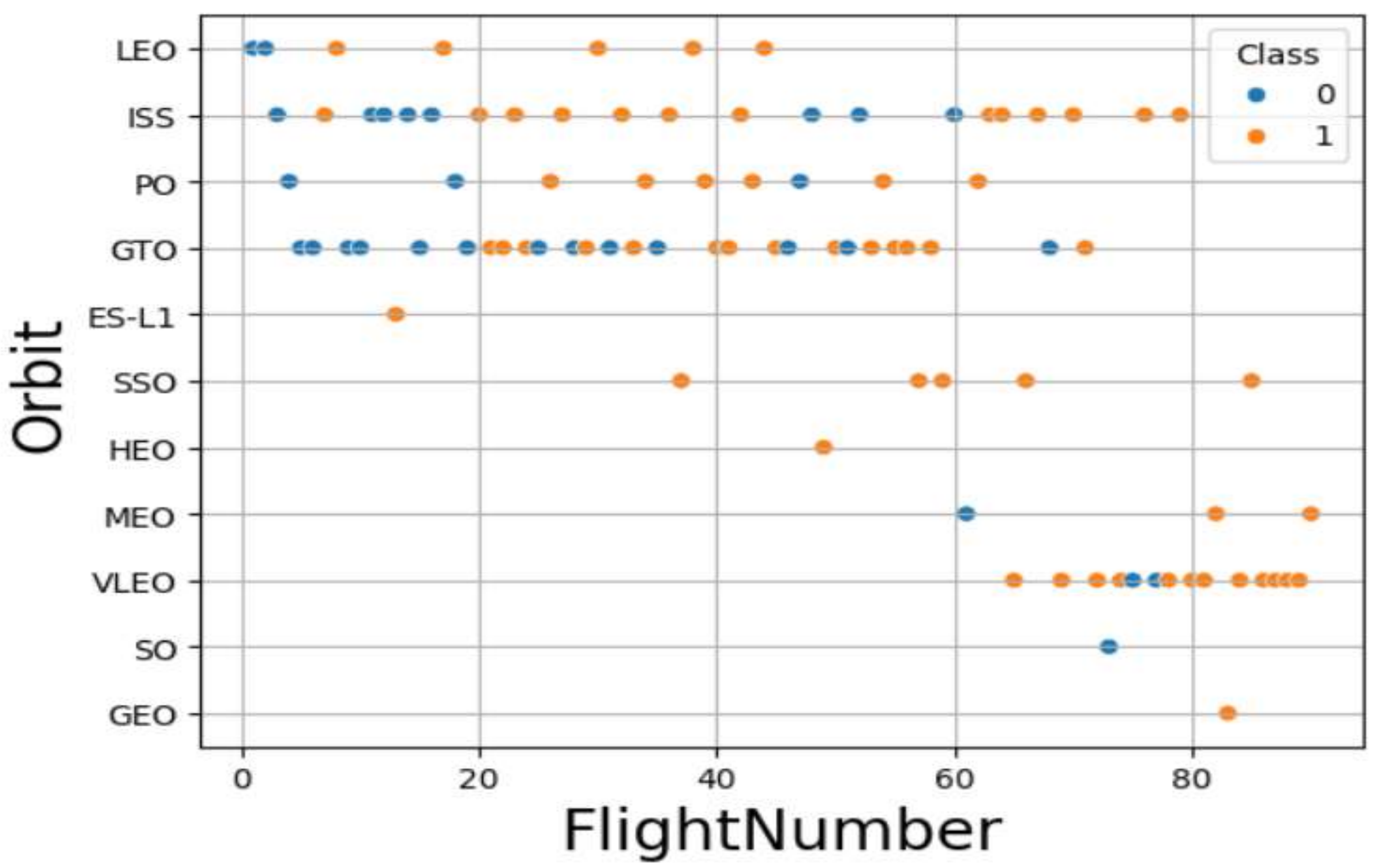
```
# HINT use groupby method on Orbit column and get the mean of Class column
success_rate = df.groupby('Orbit')['Class'].mean().reset_index()

sns.barplot(y="Class", x="Orbit", data=success_rate)
plt.xlabel("Orbit")
plt.ylabel("Class")
plt.title('Success Rate by Orbit Type')
plt.show()
```



From the plot above, the ES-L1, GEO, HEO and SSO Orbits have 100% success rate while the SO orbit has 0% success rate.

Flight Number vs. Orbit Type

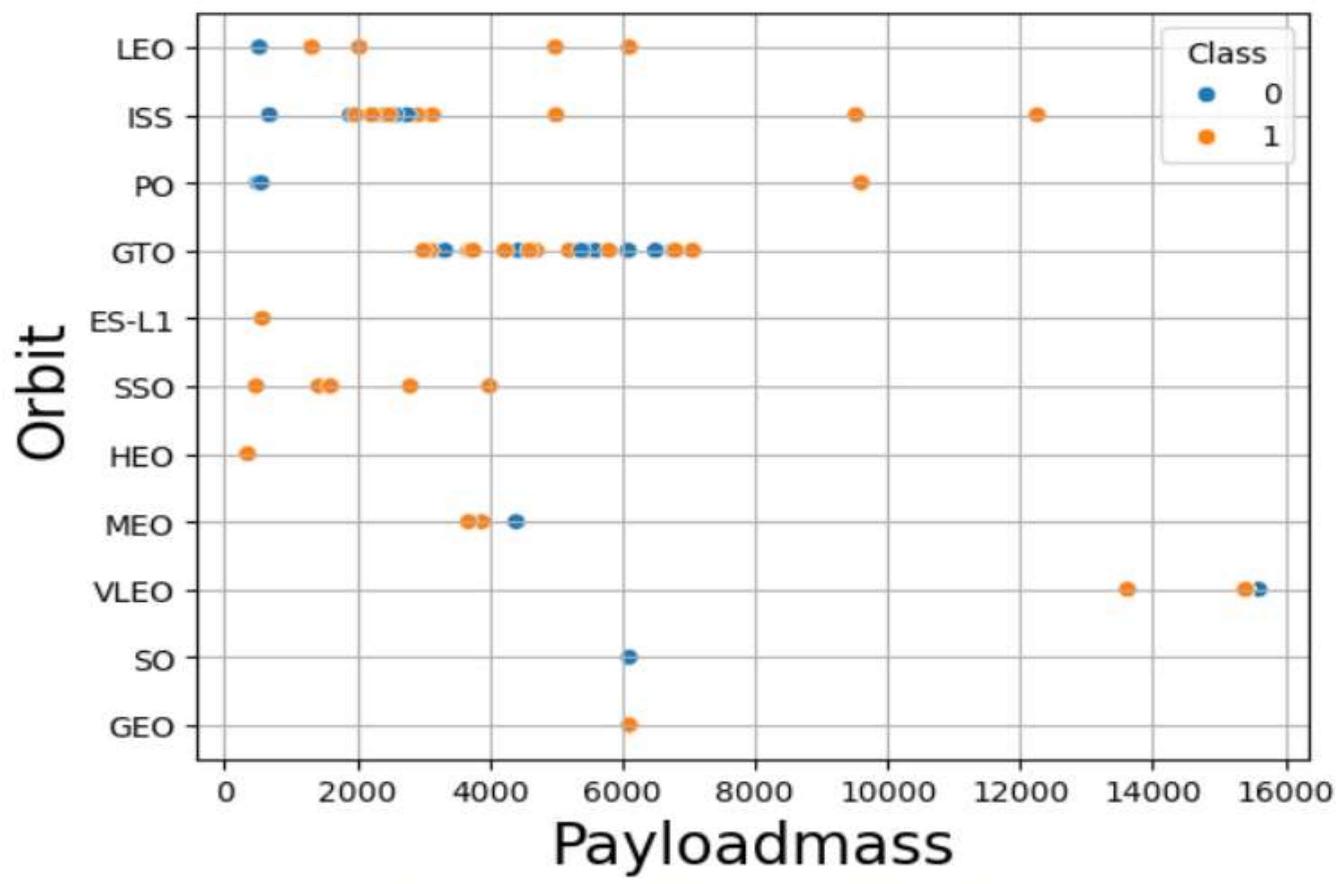
```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orb
sns.scatterplot(y="Orbit", x="FlightNumber", hue="Class", data=df)
plt.xlabel("FlightNumber",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.grid(True)
plt.show()
```



You can observe that in the LEO orbit, success seems to be related to the number of flights. Conversely, in the GTO orbit, there appears to be no relationship between flight number and success.

Payload vs. Orbit Type

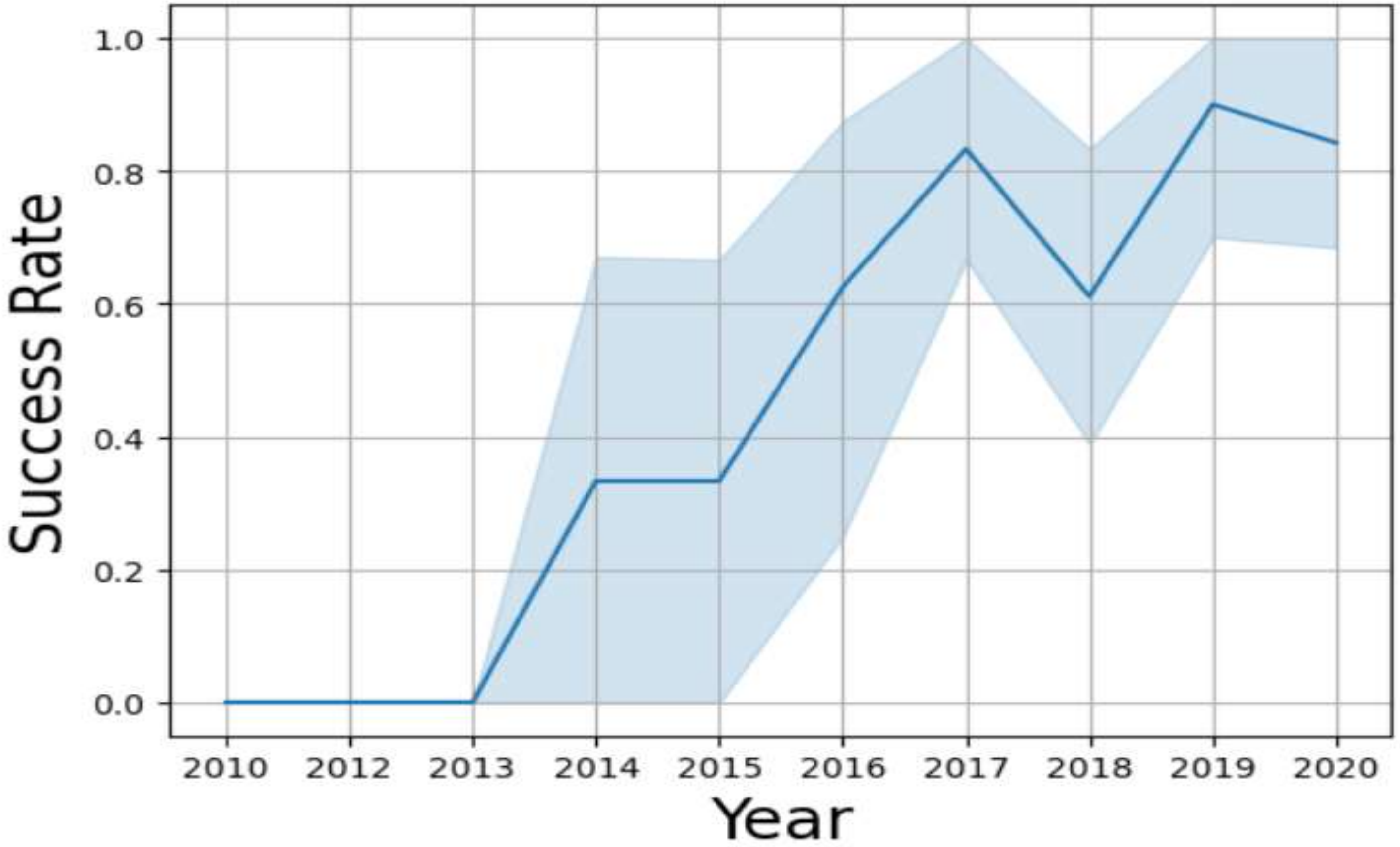
```
4]:  # Plot a scatter point chart with x axis to be Payload Mass and y axis to be the (
sns.scatterplot(y="Orbit", x="PayloadMass", hue="Class", data=df)
plt.xlabel("Payloadmass",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.grid(True)
plt.show()
```



With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS.
However, for GTO, it's difficult to distinguish between successful and unsuccessful landings as both outcomes are present.

Launch Success Yearly Trend

```
# Plot a line chart with x axis to be the extracted year and y axis to be the su
sns.lineplot(y="Class", x="Date", data=df)
plt.xlabel("Year",fontsize=20)
plt.ylabel("Success Rate",fontsize=20)
plt.grid(True)
plt.show()
```



It is observed that the success rate since 2013 has been increasing till 2020

All Launch Site Names

```
17]:  %sql SELECT DISTINCT Launch_Site FROM SPACEXTABLE;

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

17]:

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

All Launch Site names , we get this by using “DISTINCT” method

Launch Site Names Begin with 'CCA'

18]:

```
%sql SELECT * FROM SPACEXTABLE WHERE Launch_Site LIKE 'CCA%' LIMIT 5;
```

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

18]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

We find name begin with “CCA” with using LIKE and LIMIT 5

Total Payload Mass

```
[13]: %sql SELECT SUM("PAYLOAD_MASS__KG_") AS TotalPayload_Mass FROM SPACEXTABLE WHERE "Customer" LIKE  'NASA (CRS)';

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

[13]:

TotalPayload_Mass
45596

We use 'SUM' to get the total payload mass of all launches by NASA

Average Payload Mass by F9 v1.1

```
2]:  %sql SELECT AVG("Payload_Mass__kg_") AS Average_Payload_Mass FROM SPACEXTABLE WHERE "Booster_Version" = 'F9 v1.1';

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

2]:

Average_Payload_Mass
2928.4

We use 'AVG' to get the average
payload mass by booster version F9 v1.1

First Successful Ground Landing Date

```
23]:  %sql SELECT MIN(Date) FROM SPACEXTABLE WHERE Landing_Outcome LIKE '%pad%';

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

23]:

MIN(Date)
2015-12-22

I used “MIN” to get the date when the first successful landing outcome in ground pad was achieved because the first date is same with the minimum date

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql SELECT Booster_Version FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ > 4000 ANI
```

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

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

The payload mass data was taken between 4000 and 6000 only, and
the landing outcome was determined to be “success drone ship”

Total Number of Successful and Failure Mission Outcomes

```
]: %sql SELECT COUNT(Landing_Outcome) AS "Successful Outcomes" FROM SPACEXTABLE WHERE landing_outcome LIKE "%Success%"

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

Successful Outcomes
61

```
]: %sql SELECT COUNT(Landing_Outcome) AS "Failure Outcomes" FROM SPACEXTABLE WHERE landing_outcome LIKE "%Fail%"

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

Failure Outcomes
10

I used COUNT and Like “success ” to get all successful outcomes

And used COUNT and like “Fail” to get all failure outcomes

Boosters Carried Maximum Payload

```
%sql SELECT Booster_Version FROM SPACEXTABLE WHERE payload_mass__kg_ = (SELECT MAX(payload_mass__kg_) FROM SPACEXTABLE);
```

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

Booster_Version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

We get Maximum Payload masses by using “MAX”

2015 Launch Records

```
%%sql SELECT
    CASE Substr(Date, 6, 2)
        WHEN '01' THEN 'January'
        WHEN '02' THEN 'FebruAry'
        WHEN '03' THEN 'Marvh'
        WHEN '04' THEN 'April'
        WHEN '05' THEN 'May'
        WHEN '06' THEN 'June'
        WHEN '07' THEN 'July'
        WHEN '08' THEN 'August'
        WHEN '09' THEN 'September'
        WHEN '10' THEN 'October'
        WHEN '11' THEN 'November'
        WHEN '12' THEN 'December'
    END AS month_name,
    Landing_Outcome,
    Launch_Site
FROM
    SPACEXTABLE
WHERE
    substr(Date, 0, 5) = '2015'
    AND Landing_Outcome = 'Failure (drone ship)';
```

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

month_name	Landing_Outcome	Launch_Site
January	Failure (drone ship)	CCAFS LC-40
April	Failure (drone ship)	CCAFS LC-40

I used substr (Date, 6, 2) as month to get the months and Substr (Date, 0, 5) = ‘2015’ for year

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

```
%%sql SELECT
    Landing_Outcome,
    COUNT(*) AS Outcome_Count
FROM
    SPACEXTABLE
WHERE
    Date BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY
    Landing_Outcome
ORDER BY
    Outcome_Count DESC;
```

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

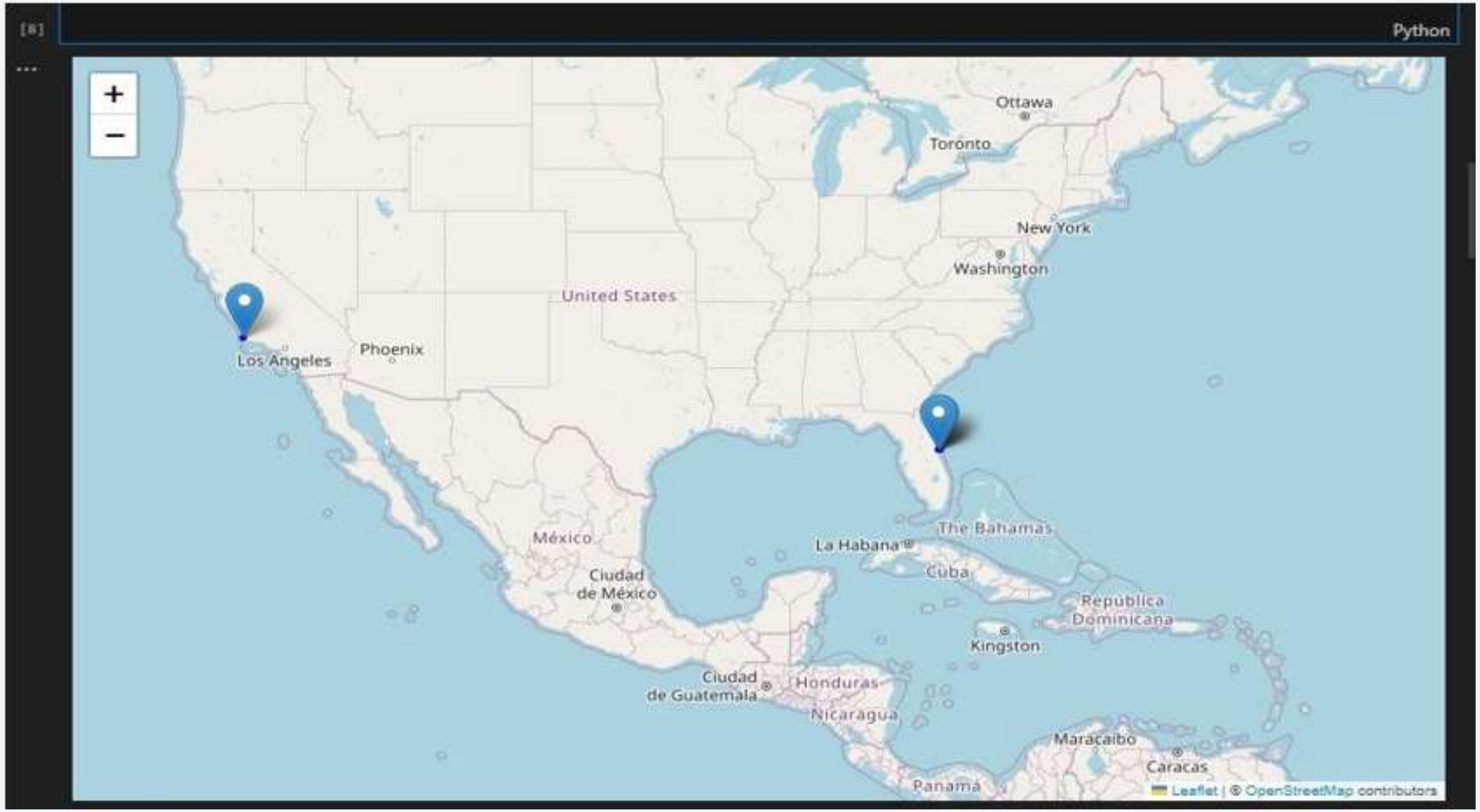
Landing_Outcome	Outcome_Count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

I used COUNT to count the occurances GROUP BY to arrange the data by column and ORDER BY and DESC to arrange the count in descending order

Section 3

Launch Sites Proximities Analysis

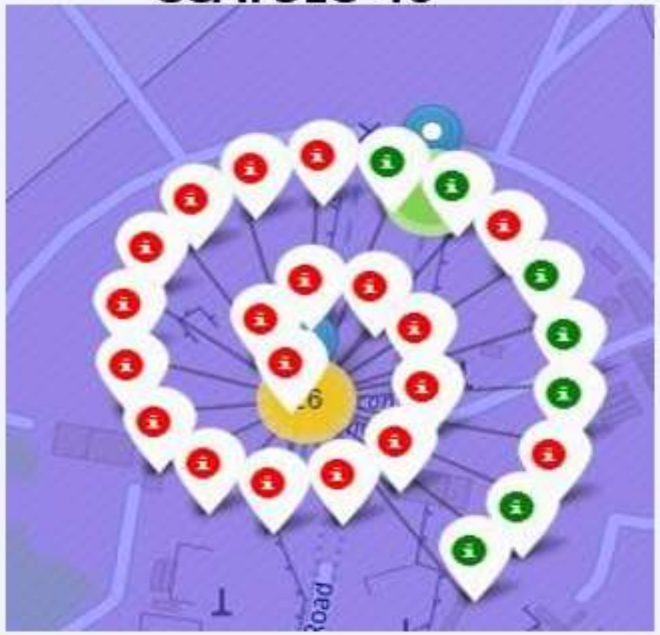
All Launch Site Location Markers



Color - labeled Launch Outcomes

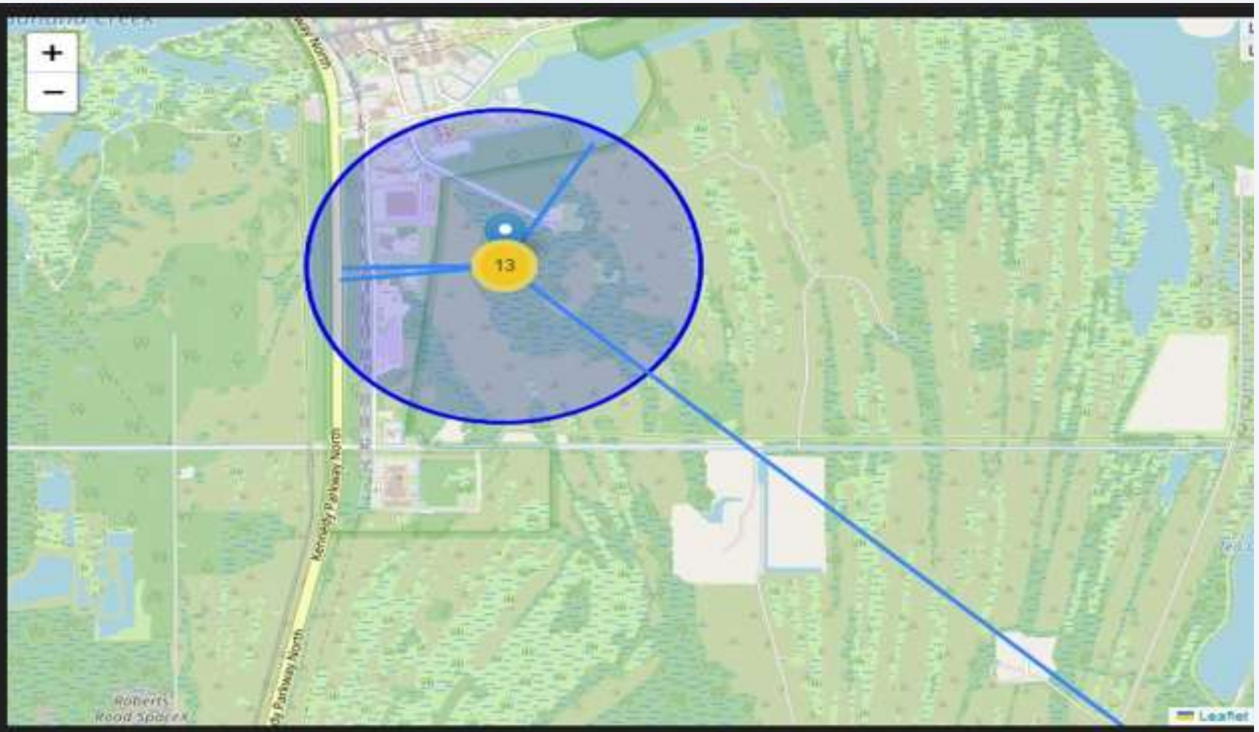
VAFB SLC- 4E

CCAFSLC-40



Key:
GREEN = SUCCESSFUL
RED = FAILURE

Co-ordinate to the Closest City



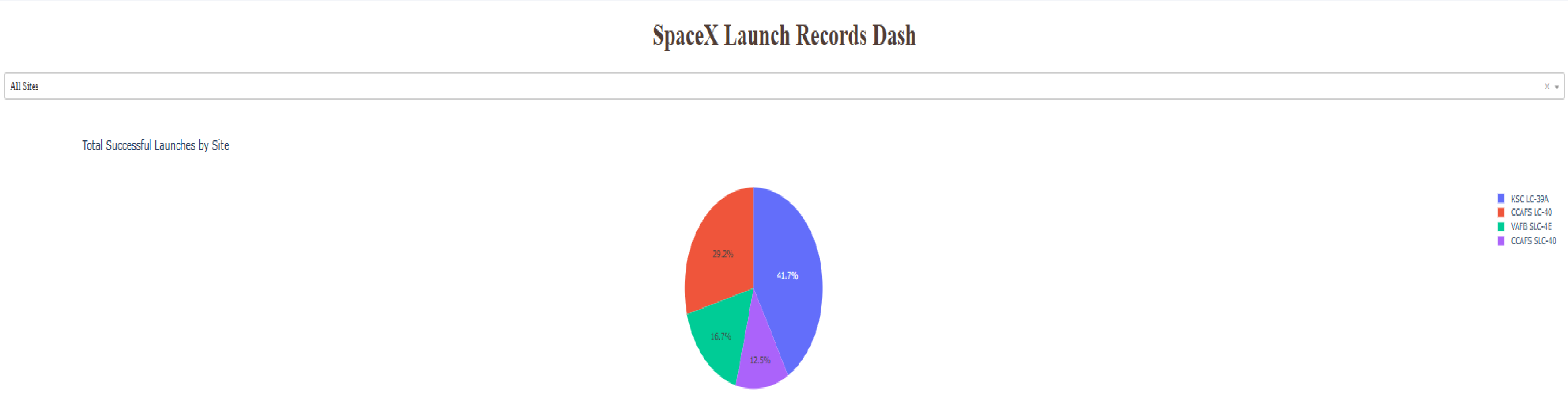

```
# find coordinate of the closet city
# e.g.,: Lat: 28.53989  Lon: -80.60645
city_lat = 28.57258
city_lon = -80.65149

distance_city = calculate_distance(launch_site_lat, launch_site_lon, city_lat, city_lon)
distance_city
```

```
0.4550896858875425
```

Section 4

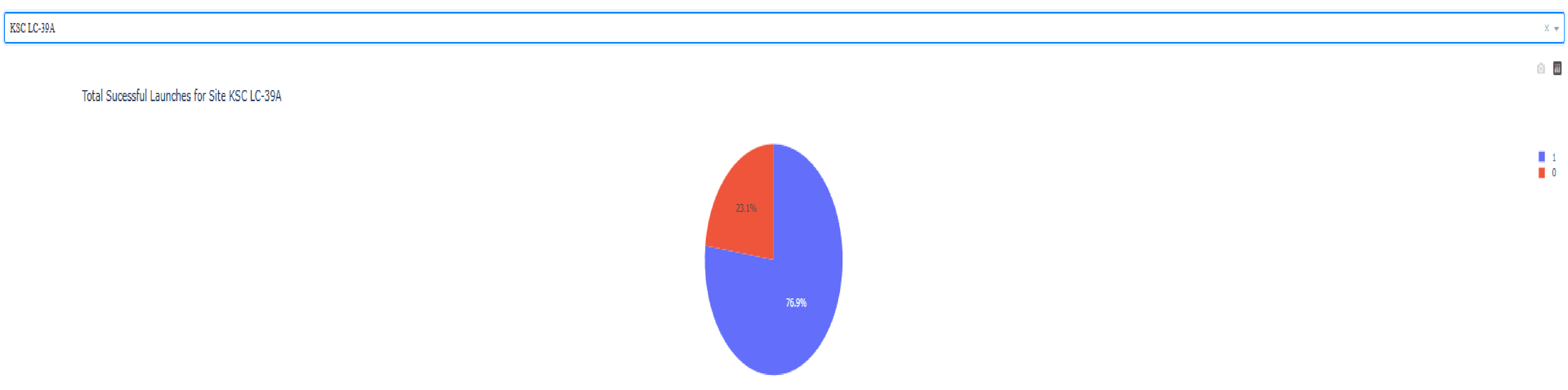
Build a Dashboard with Plotly Dash

Launch Success Count for all sites



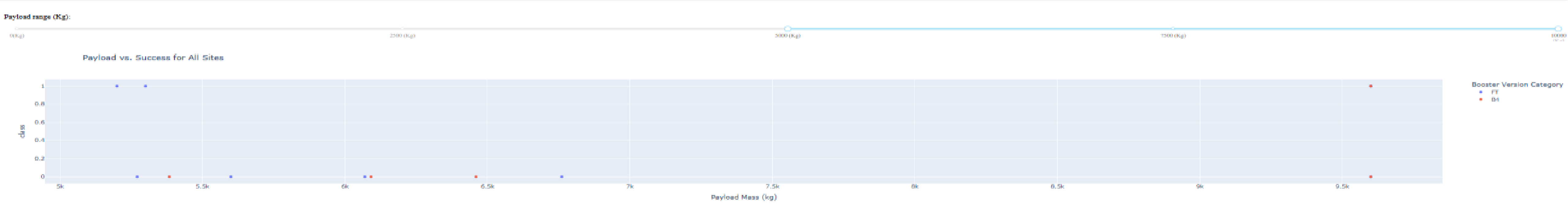
KSC LC-39A has the highest success score with 41.7%. This is followed by CCAFS LC-40 with 29.2% success rate and then VAFB SLC-4E and CCAFS SLC-40 with 16.7% and 12.5% respectively

Launch site with the highest launch success ratio



KSC LC 39A Has the highest success Ratio of 76.9%

Payload vs. Launch Outcome

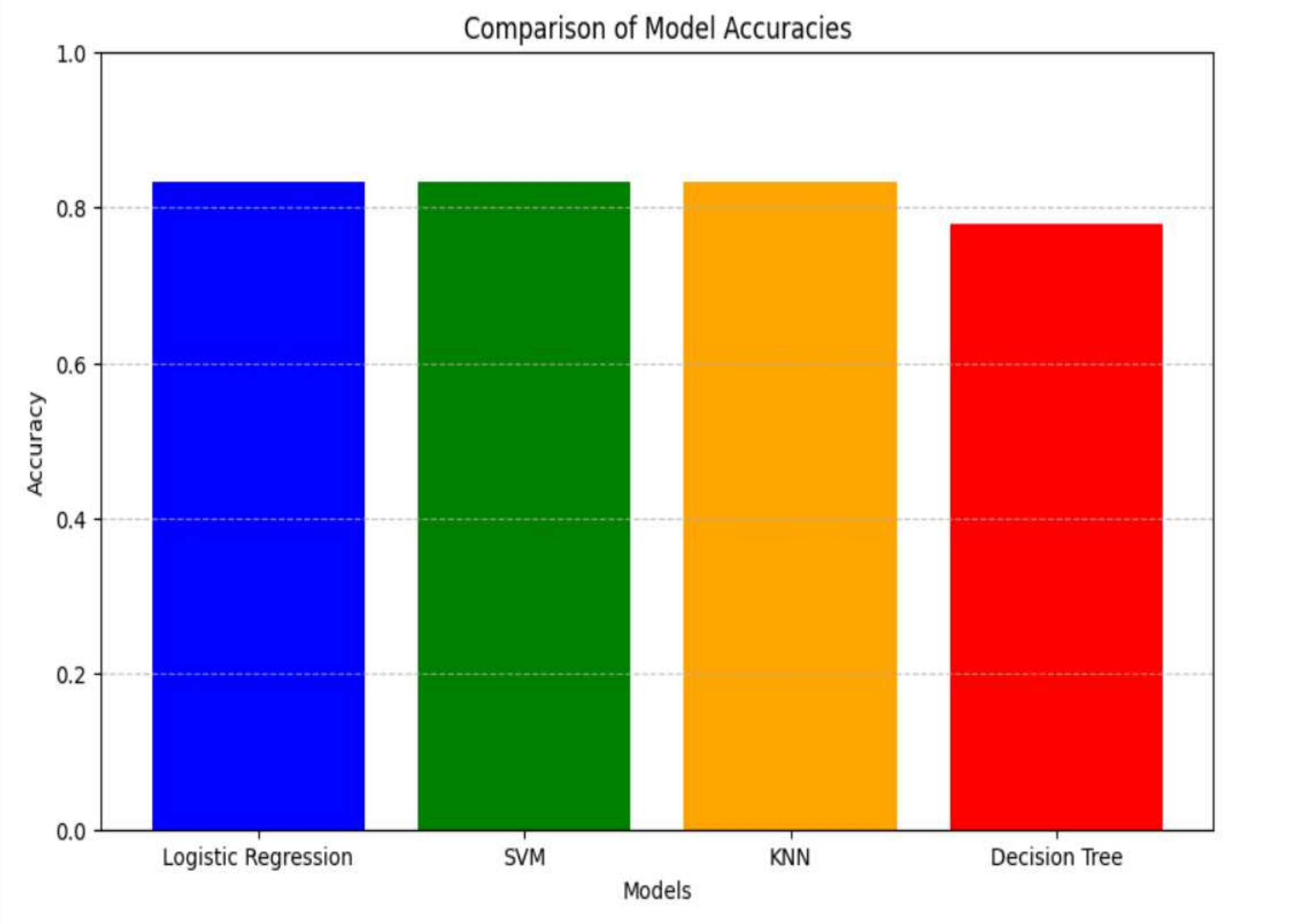


Payload range (0 – 10,000) kg for all sites. It is observed that booster version B4, FT and B5 has a high success rate at this range. Conversely, for the range (5,000 – 10,000)kg, both FT and B4 are seen to have low success rate.

Section 5

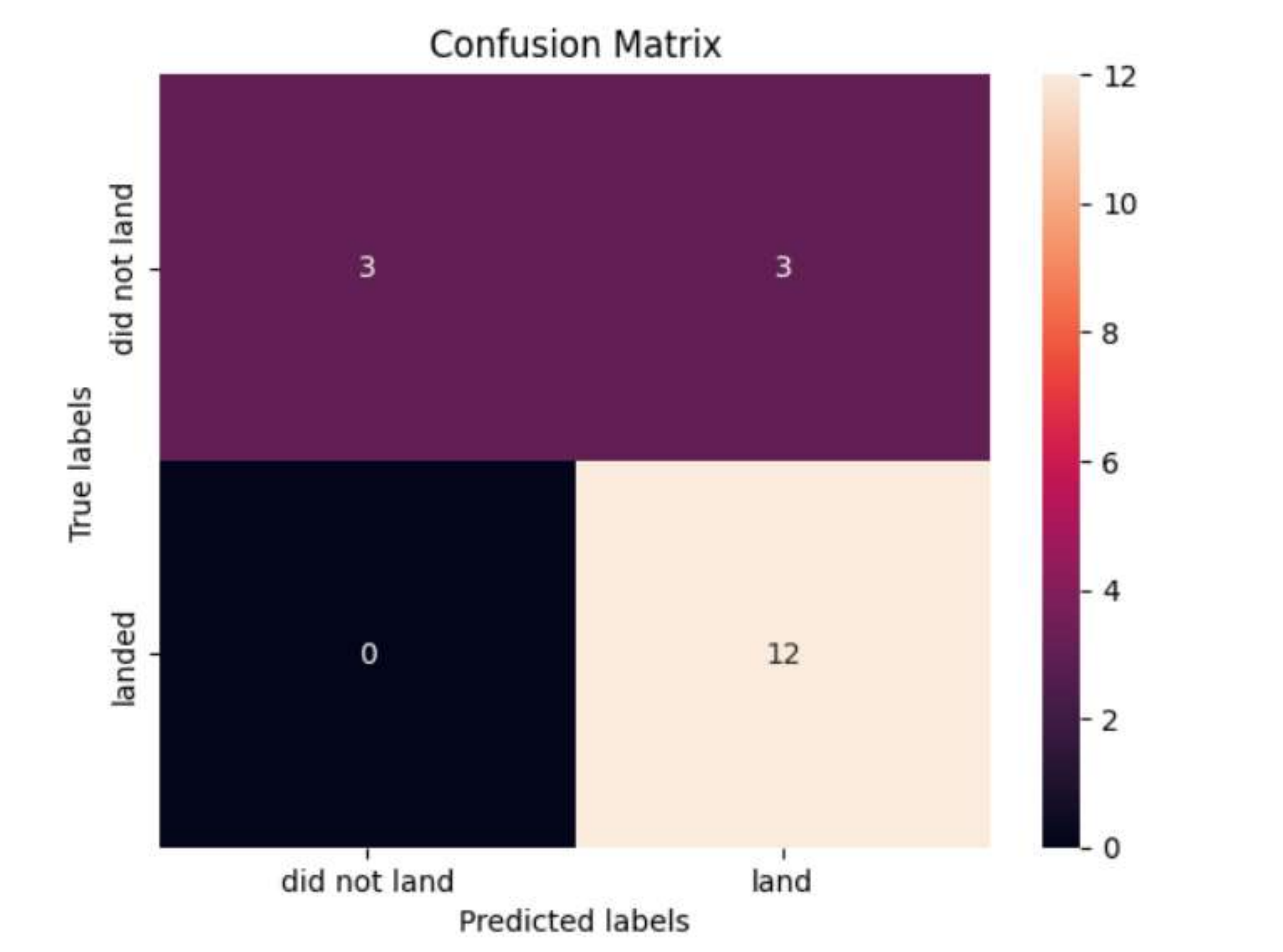
Predictive Analysis (Classification)

Classification Accuracy



Logistic model , SVM, KNN has the same accuracy rate where Decision Tree has the lowest accuracy in my assignment

Confusion Matrix



- **Accuracy** = (TP + TN) / Total = (12 + 3) / 18 = **0.8333**
- **Precision** (for "landed") = TP / (TP + FP) = 12 / (12 + 3) = **0.8**
- **Recall** (for "landed") = TP / (TP + FN) = 12 / (12 + 0) = **1.0**
- **F1 Score** = 2 × (Precision × Recall) / (Precision + Recall) ≈ **0.8889**

Conclusions

- We found that KSC LC-39A has the Highest Success rate.
- Payload Range of 0 – 10,000 gave the Better success rate than range 5000 – 10,000
- Based on the performance evaluation of four machine learning models — Logistic Regression, SVM, KNN, and Decision Tree — we observe that **Logistic Regression**, **SVM**, and **KNN** each achieved a high accuracy of approximately **83.33%**, as visualized in the bar chart. Among them, **Logistic Regression** was selected as the best performing model due to its consistent results and slightly more favorable characteristics
- The launch sites were found to be within short distances to railways, coastline, highways and cities
- A positive relationship was observed between flight number and success rate

Appendix

Please refer to my GitHub for my all Codes

GitHub:https://github.com/Kumaresh07-git/DataScience-Capstone-project

Thank you!