Joseph Ogijoh

Winning Space Race with Data Science

Joseph Ogijoh
February 5, 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

Data collection methodology: Data was collected from Space X API ensuring completeness and accuracy. The process involved sending a get request to the API.

Perform data wrangling: The data was cleaned by handling missing values, removing duplicates and handling errors. We went further to transform the data by creating a landing outcome label from the outcome column.

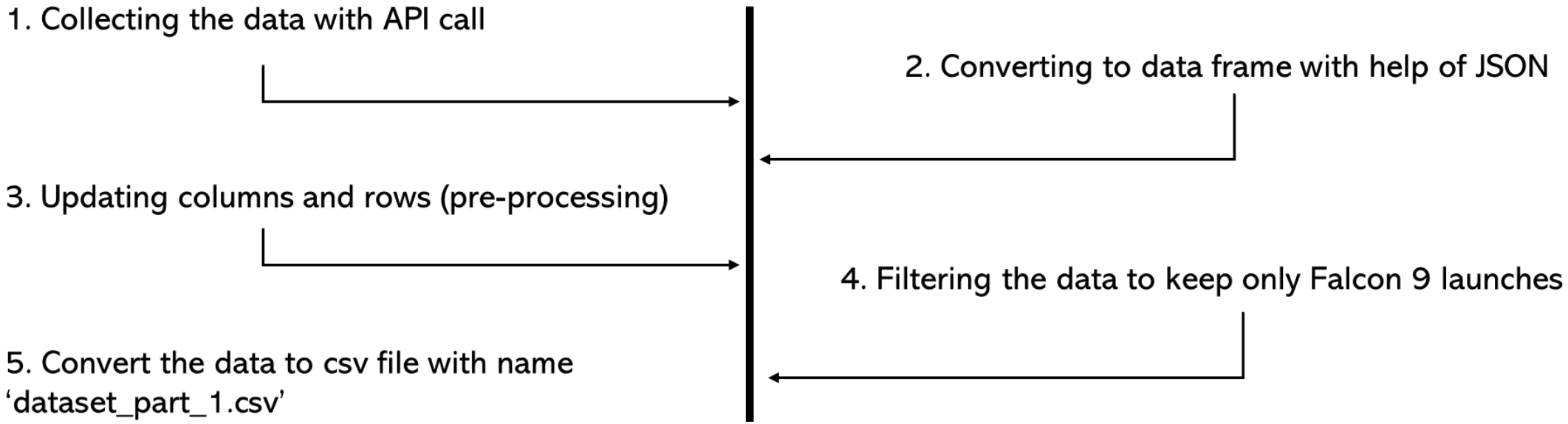
Perform exploratory data analysis (EDA) using visualization and SQL to discover trends and patterns

Perform interactive visual analytics using Folium and Plotly Dash

Perform predictive analysis using classification models: Here, we perform exploratory data analysis to determine training labels, create a column for the class, standardize the data, split the data into training data and test data, find best Hyperparameter for SVM, Classification Trees, KNN and logistic regression and also find the method that performs best using the 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.



Data Collection – SpaceX API

1. Collecting the data with API call

```
In [2]: # 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']:
                response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
                BoosterVersion.append(response['name'])
```

2. Converting to data frame with help of JSON

```
# Use json_normalize meethod to convert the json result into a dataframe
jlist = requests.get(static_json_url).json()
df = pd.json_normalize(jlist)
df.head()
```

	static_fire_date_utc	static_fire_date_unix	tbd	net	window	rocket	success	details	crew	ships	capsules	
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	False	0.0	5e9d0d95eda69955f709d1eb	False	Engine failure at 33 seconds and loss of	[]	[]	[]	[5eb0e

3. Updating columns and rows (pre-processing)

```
# 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 multi
ple 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)]
```

4. Filtering the data to keep only Falcon 9 launches

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9.drop(data_falcon9[data_falcon9['BoosterVersion']!='Falcon 9'].index, inplace = True)
```

5. Convert the data to csv file with name ‘dataset_part_1.csv’

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

GitHub repo: https://github.com/Manibe745/Data-science/blob/main/SpaceX%20Falcon%209%20first%20stage%20Landing%20Prediction%20(1).ipynb?short_path=f84f5ed

Data Collection - Scraping

GitHub repo: https://github.com/Manibe745/Data-science/blob/main/Space%20X%20Falcon%209%20First%20Stage%20Landing%20Prediction.ipynb?short_path=a0fc8fb

1. Creating the BeautifulSoup object

```
# Extract the HTML content from the response
html_content = response.text
```

Create a `BeautifulSoup` object from the HTML `response`

```
soup = BeautifulSoup(html_content, 'html.parser')
```

2. Getting column names

```
column_names = []
column_names = []

# Assuming first_launch_table is your BeautifulSoup object containing the table
th_elements = first_launch_table.find_all('th')

for th in th_elements:
    name = extract_column_from_header(th)
    if name is not None and len(name) > 0:
        column_names.append(name)

# Now, column_names should contain all the extracted column names


print(column_names)
```

3. Creating the launch_dict

```
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.`
            #print(flight_number)
            datatimelist=date_time(row[0])
            launch_dict['Flight No.'].append(flight_number)
```

4. Converting to final data frame

	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 v1.0B0003.18	Failure	4 June 2010	18:45
1	2	CCAFS	Dragon	0	LEO	NASA (COTS)\nNRO	Success	F9 v1.0B0004.18	Failure	8 December 2010	15:43
2	3	CCAFS	Dragon	525 kg	LEO	NASA (COTS)	Success	F9 v1.0B0005.18	No attempt\n	22 May 2012	07:44
3	4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA (CRS)	Success\n	F9 v1.0B0006.18	No attempt	8 October 2012	00:35
4	5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA (CRS)	Success\n	F9 v1.0B0007.18	No attempt\n	1 March 2013	15:10

5. Convert the data to csv file with name ‘spacex_web_scraped.csv’

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

Data Wrangling

GitHub repo: https://github.com/Manibe745/Data-science/blob/main/Space%20X%20Falcon%209%20First%20Stage%20Landing%20Prediction_Data_wrangling%20(1).ipynb?short_path=014fbd1

1. Loading the data set

```
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/da
art_1.csv")
df.head(10)
```

2. Creating landing outcomes

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

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

3. Finding the bad outcomes

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

```
{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

4. Presenting outcomes as 0 and 1

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

5. Determining the success outcome

```
df["Class"].mean()
```

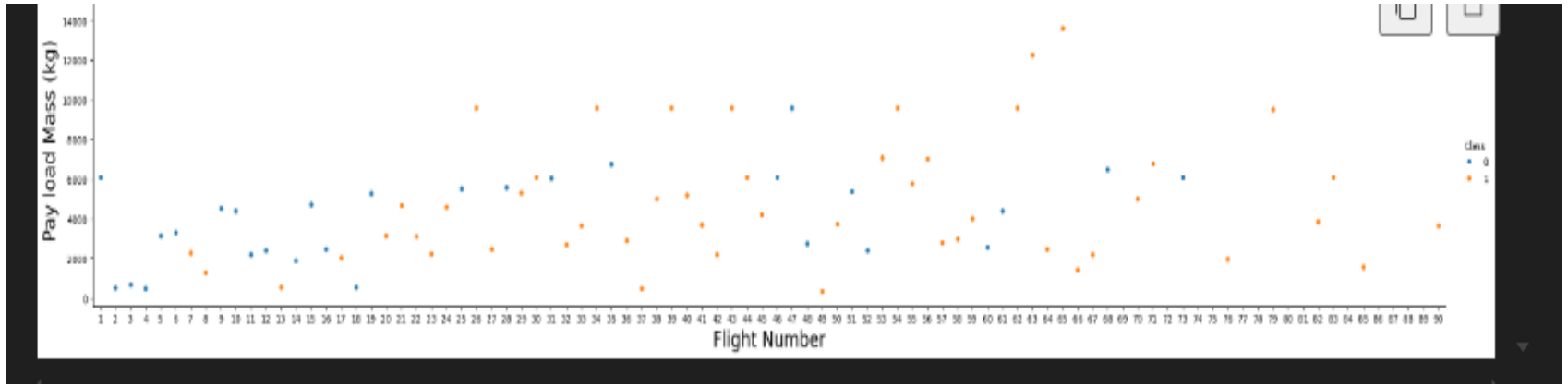
```
0.6666666666666666
```

6. Convert the data to csv file with name 'dataset_part_2.csv'

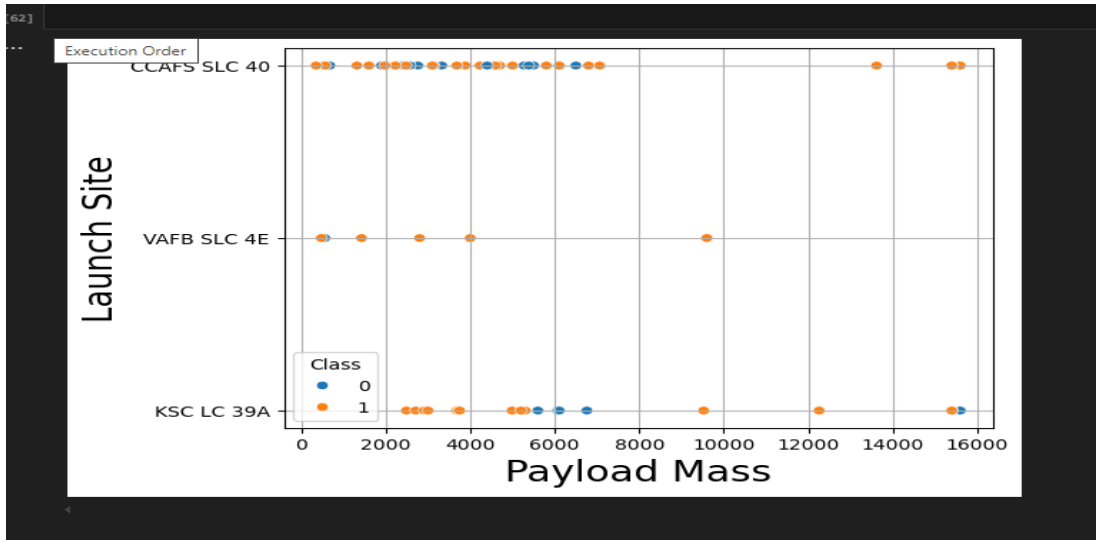
```
df.to_csv("dataset_part_2.csv", index=False)
```

EDA with Data Visualization

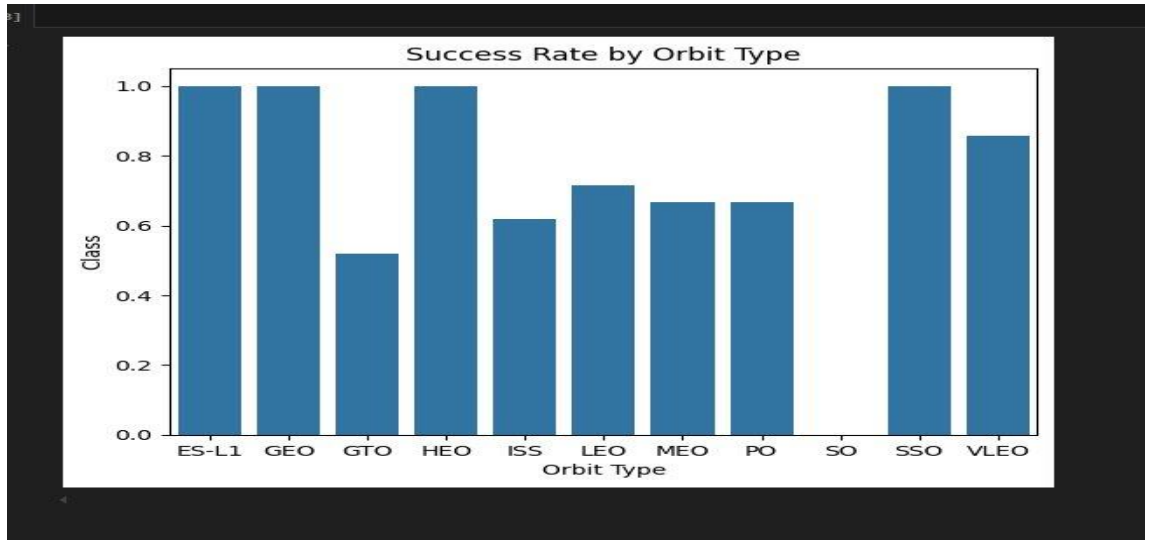
GitHub repo: https://github.com/Manibe745/Data-science/blob/main/SpaceX_Exploring_and_preparing_data.ipynb?short_path=8787f8c



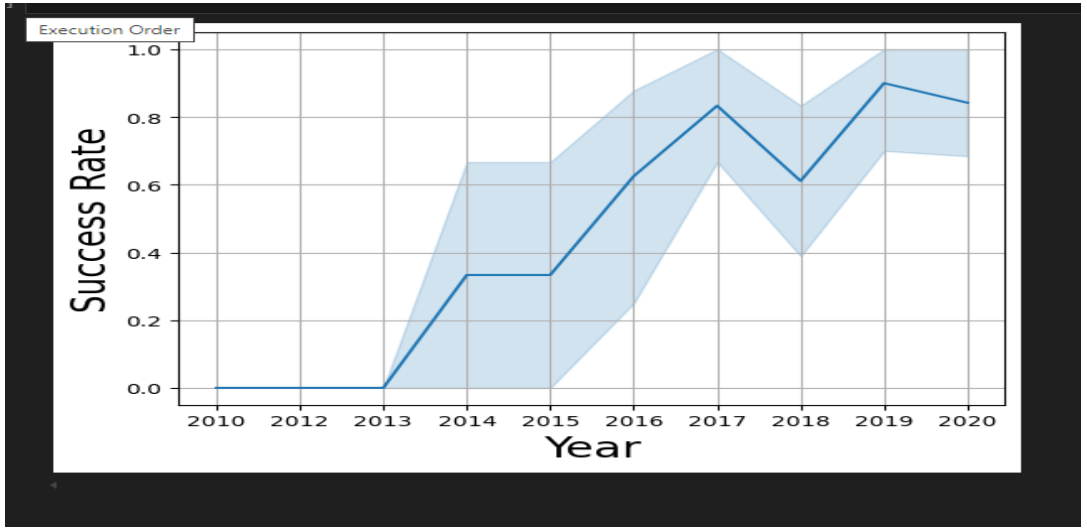
Categorial plot between Pay load mass (kg) and Flight number



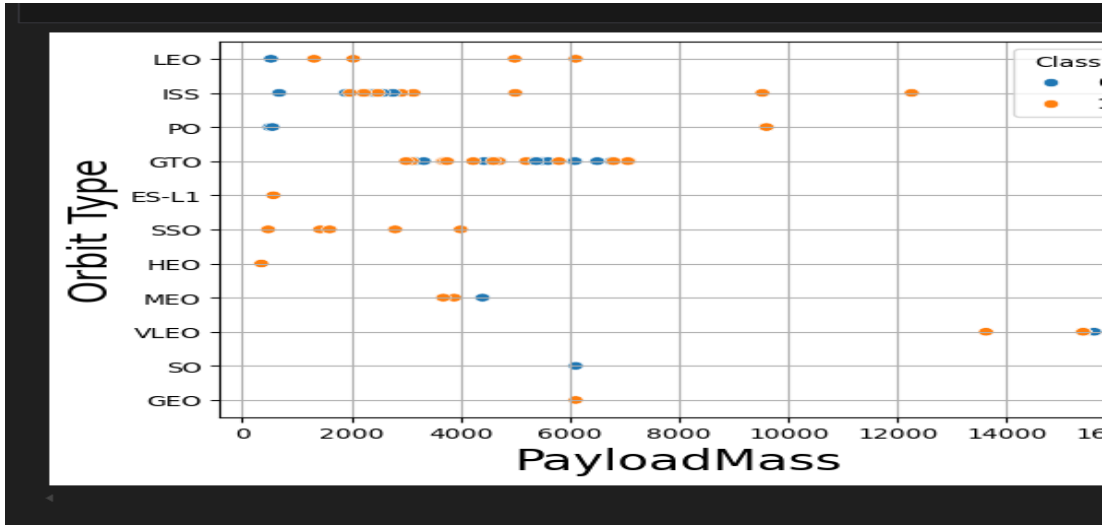
Scatter plot between Launch site and Payload mass



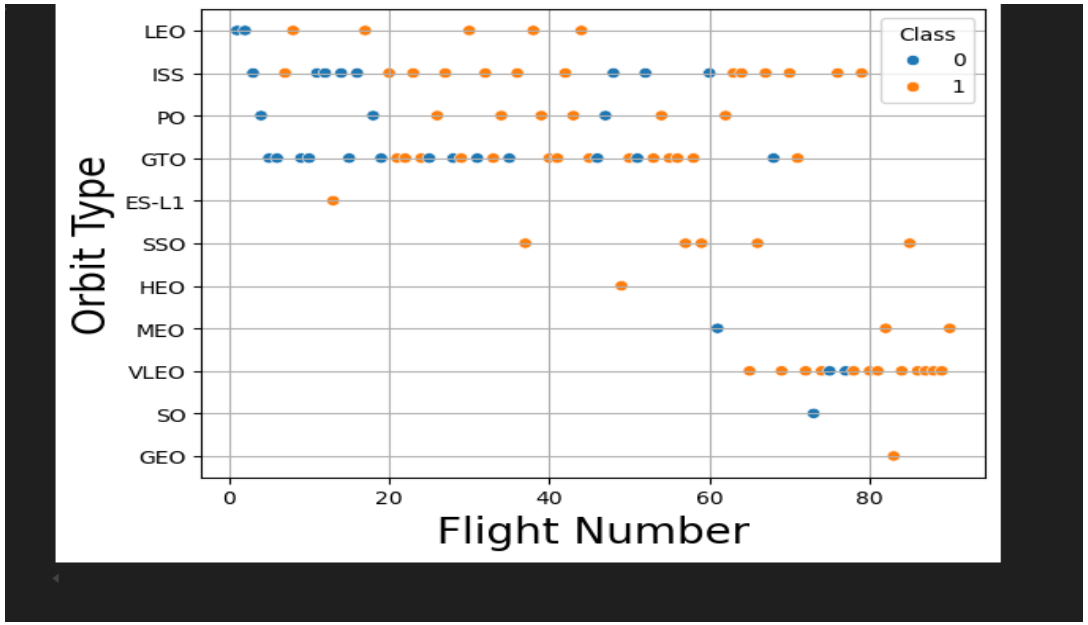
Bar chart of success rate of each orbit type



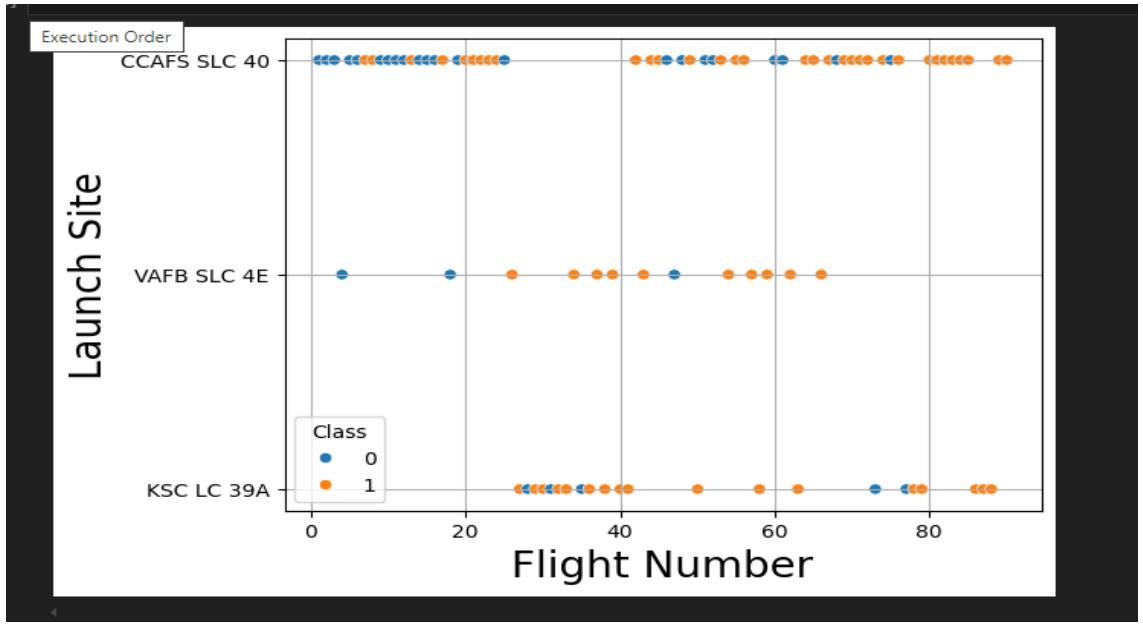
Line chart between success rate and Year



Scatter plot between Orbit type and Payload mass



Scatter plot between Orbit type and Flight number



Scatter plot between Launch site and Flight number

EDA with SQL

GitHub repo: https://github.com/Manibe745/Data-science/blob/main/Exploratory_analysis_of_spaceX_Falcon9_Landing_prediction%20(1).ipynb

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

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/Manibe745/Data-science/blob/main/Interactive%20Visual%20Analytics%20with%20Folium_on%20falcon9_prediction.ipynb?short_path=fd7dd88

Build a 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/Manibe745/Data-science/blob/main/spacex_dash_app_updated.py

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

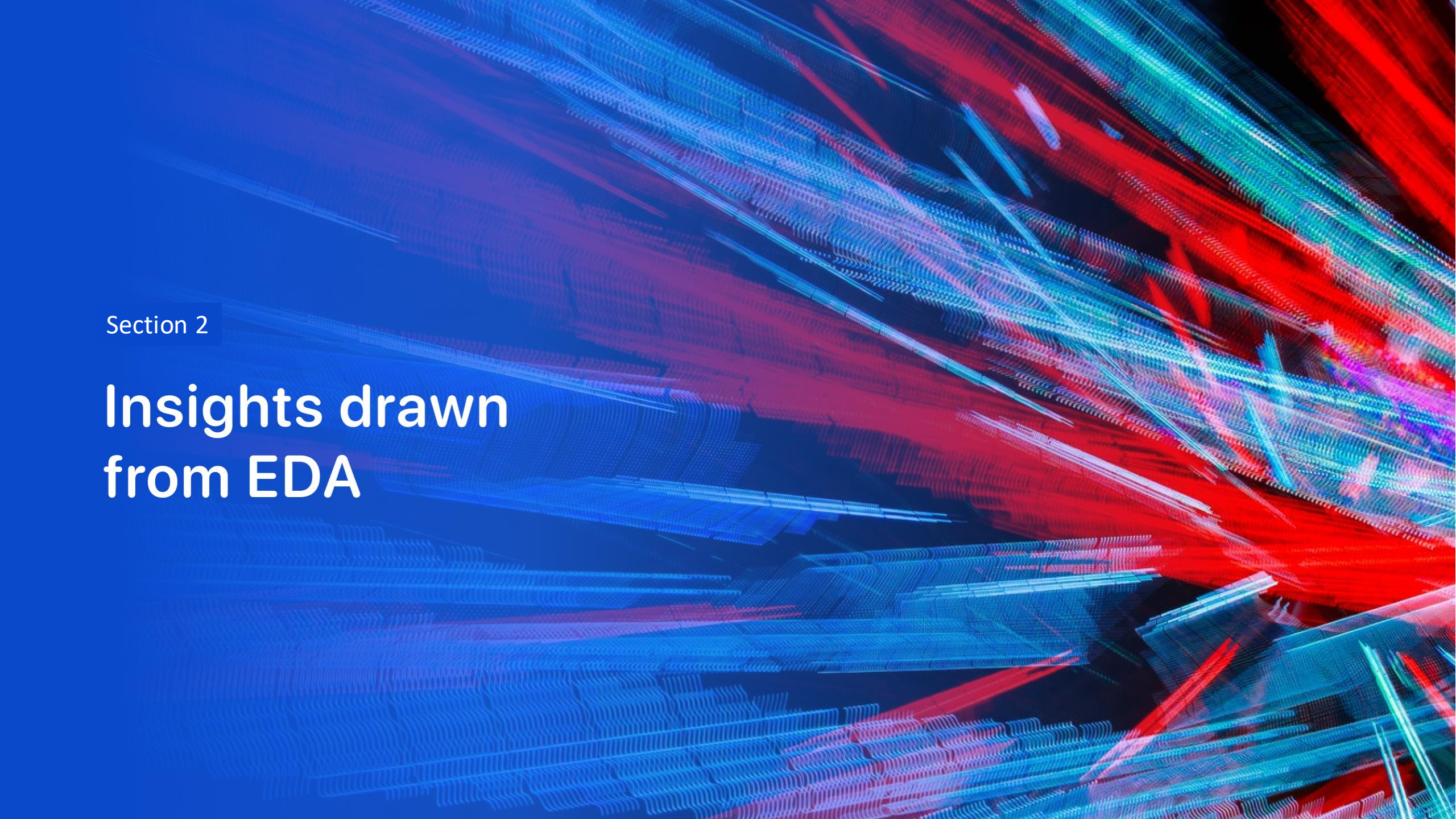
Find the best model with highest accuracy

Confirm the optimal model

GitHub repo: https://github.com/Manibe745/Data-science/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb%20(1)_updated2.ipynb

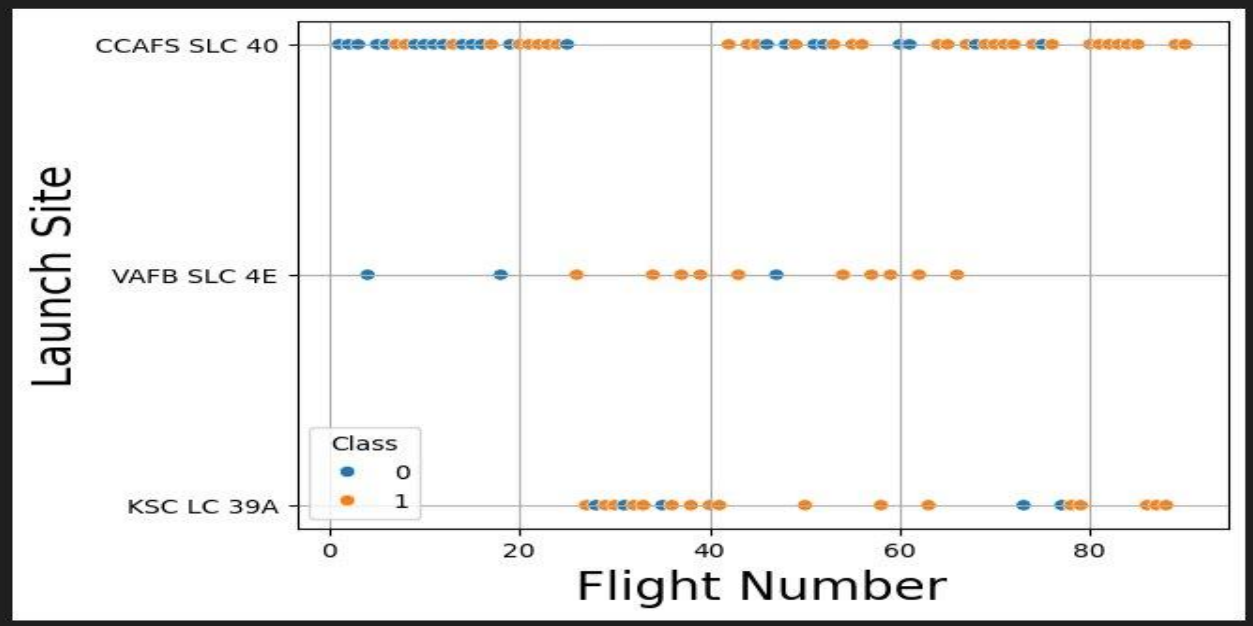
Results

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


Section 2
Insights drawn from EDA

Flight Number vs. Launch Site

```
sns.scatterplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.grid(True)
plt.show()
```

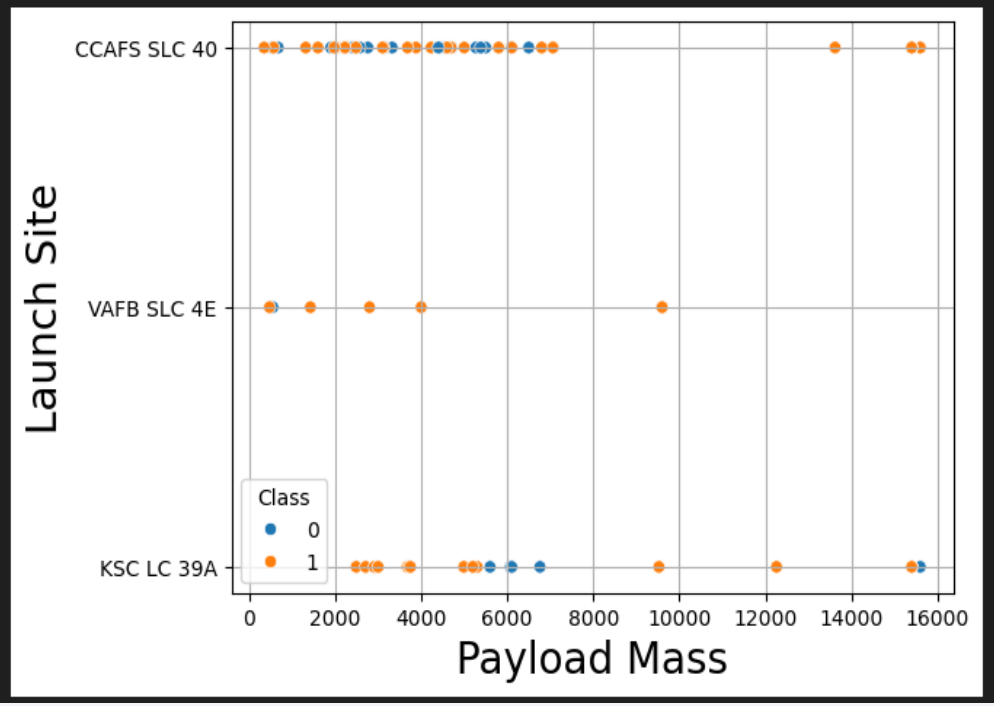


From the plot, we find for the VAFB-SLC launchsite there are no rockets launched for high flight number (greater than 70). However, we can see an overall increase in success rate for all Launch sites with increasing flight number.

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 site, and hue to be

sns.scatterplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df)
plt.xlabel("Payload Mass",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.grid(True)
plt.show()
```

[62] Python



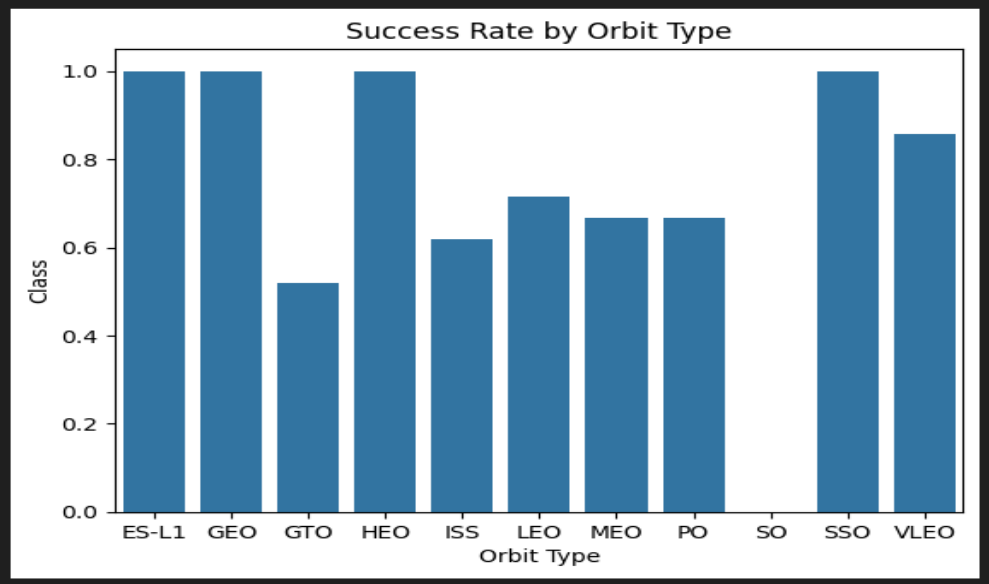
If you observe the Payload Mass Vs. Launch Site scatter point chart you will find for the VAFB-SLC launch site there are no rockets launched for heavy payload mass(greater than 10000).

However, there is an observed increase in success rate for all Launch sites with increasing payload mass.

Success Rate vs. Orbit Type

```
# Calculate success rate for each orbit type
success_rate = df.groupby('Orbit')['Class'].mean().reset_index()

# Create a bar plot
sns.barplot(x='Orbit', y='Class', data=success_rate)

# Add labels and title
plt.xlabel('Orbit Type')
plt.ylabel('Class')
plt.title('Success Rate by Orbit Type')

# Show the plot
plt.show()
```

[63]

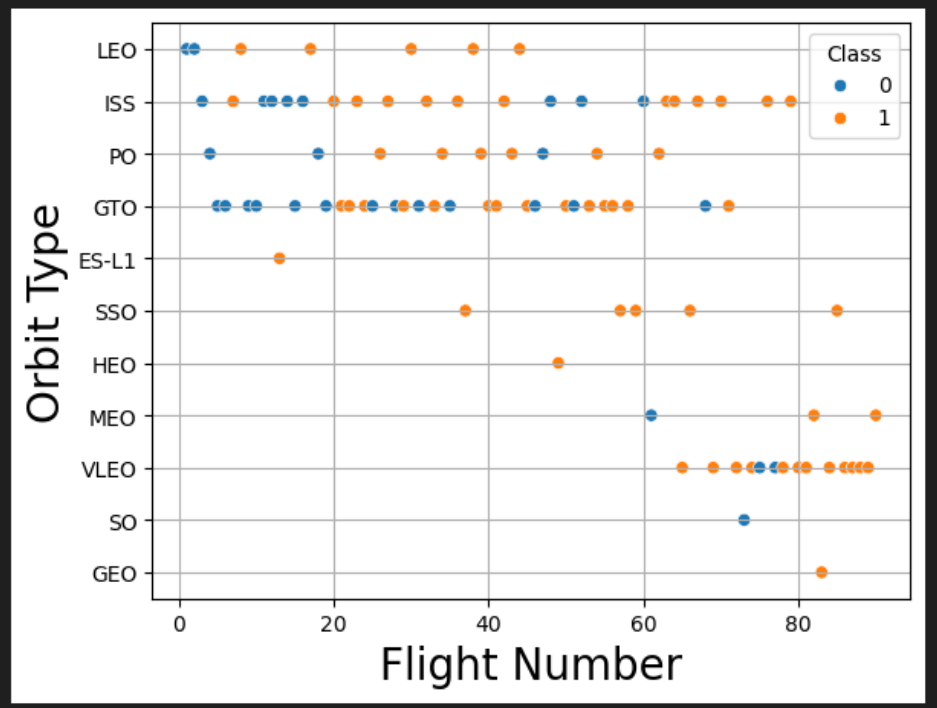


From the plot above, the ES-L1, GEO, HEO and SSO Orbits have 100% success rate while the SO orbit has 0% success rate.

Flight Number vs. Orbit Type

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

[64] Python

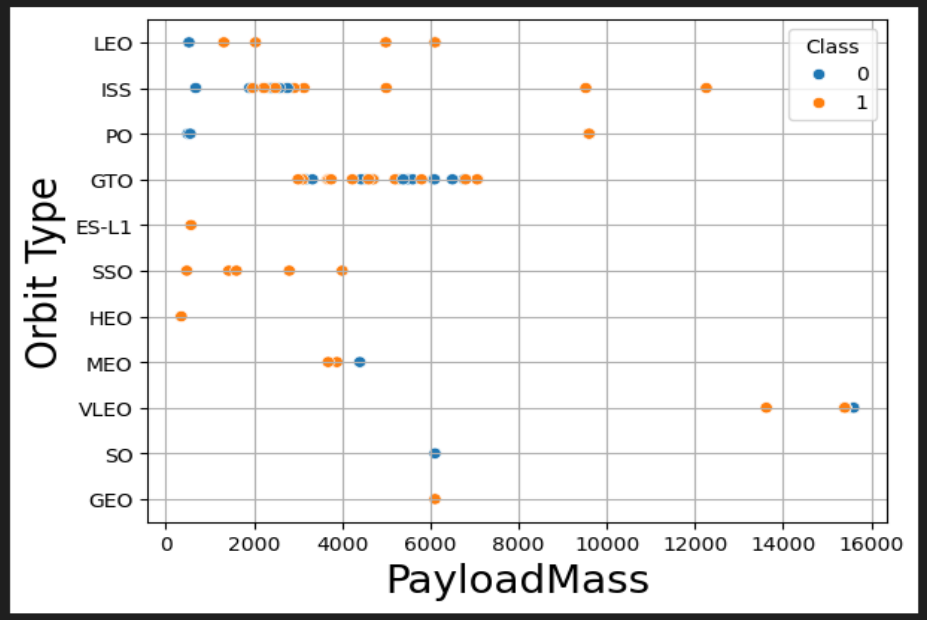


You can observe that in the LEO orbit, success seems to be related to the number of flights. Conversely, in the GTO orbit, there appears to be no relationship between flight number and success.

Payload vs. Orbit Type

```
# Plot a scatter point chart with x axis to be Payload Mass and y axis to be the Orbit, and hue to be the class va

sns.scatterplot(y="Orbit", x="PayloadMass", hue="Class", data=df)
plt.xlabel("PayloadMass",fontsize=20)
plt.ylabel("Orbit Type",fontsize=20)
plt.grid(True)
plt.show()
```

[65] Python

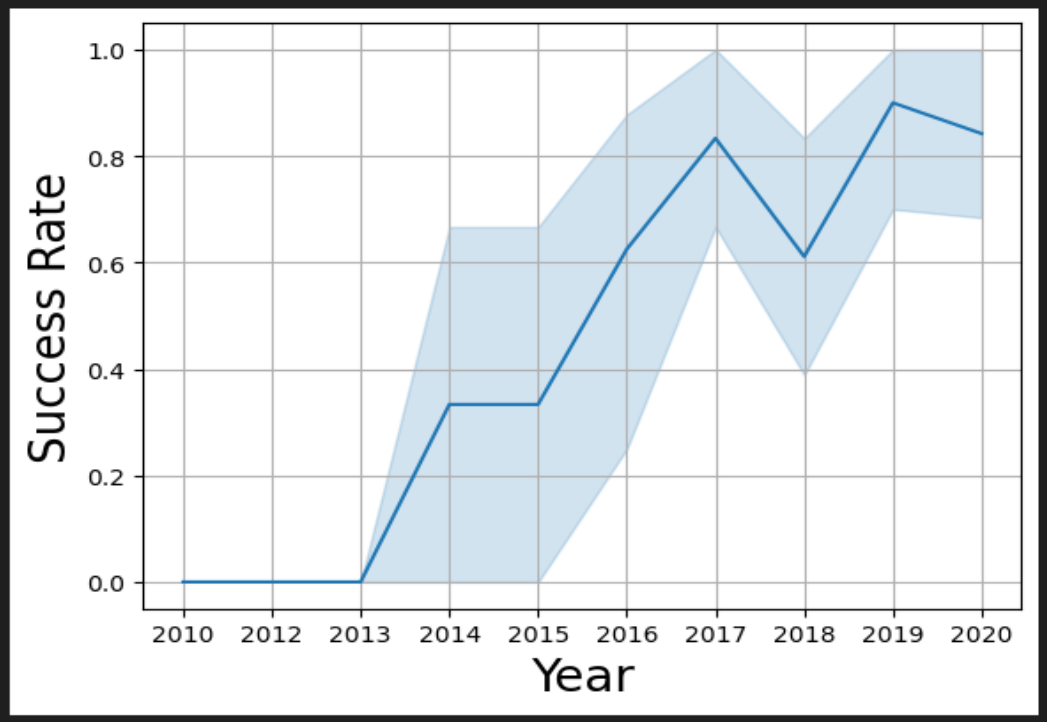


With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS.

However, for GTO, it's difficult to distinguish between successful and unsuccessful landings as both outcomes are present.

Launch Success Yearly Trend

```
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 kept increasing till 2020

All Launch Site Names

```
%sql SELECT DISTINCT "Launch_Site" FROM SPACEXTABLE;
```

[10]

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

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

We can get the unique values by using "DISTINCT"

Launch Site Names Begin with 'CCA'

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

[10] Python

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure
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
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	

We use "LIMIT" to get only 5 rows.

Total Payload Mass

```
%sql SELECT SUM("Payload_Mass__kg_") AS Total_Payload_Mass FROM SPACEXTABLE WHERE "Customer" = 'NASA (CRS)';
```

[15] Python

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

Total_Payload_Mass
45596

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

Average Payload Mass by F9 v1.1

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

[16] Python

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

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

```
%sql SELECT MIN(Date) FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (ground pad)';
```

[17]

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

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_
```

[17] Python

```
 * 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 outcome" FROM SPACEXTABLE WHERE landing_outcome LIKE "Success";
```

[19] Python

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

Successful outcome
38

```
%sql SELECT COUNT(landing_outcome) AS "Failure outcome" FROM SPACEXTABLE WHERE landing_outcome LIKE "Failure";
```

[20] Python

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

Failure outcome
3

I used “COUNT” and LIKE “Success” to get the number of all successful landing outcome, and “COUNT” and LIKE “Failure” for the number of all failed landing outcome

Boosters Carried Maximum Payload

```
%%sql

SELECT booster_version
FROM SPACEXTABLE
WHERE payload_mass__KG_ = (
    SELECT MAX(payload_mass__KG_)
    FROM spacextable
);
```

[21]

```
 * 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 the 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 'March'
        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,
    booster_version,
    launch_site
FROM
    SPACEXTABLE
WHERE
    substr(Date, 0, 5) = '2015'
    AND landing_outcome = 'Failure (drone ship)';
```

[23] Python

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

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

[24]

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

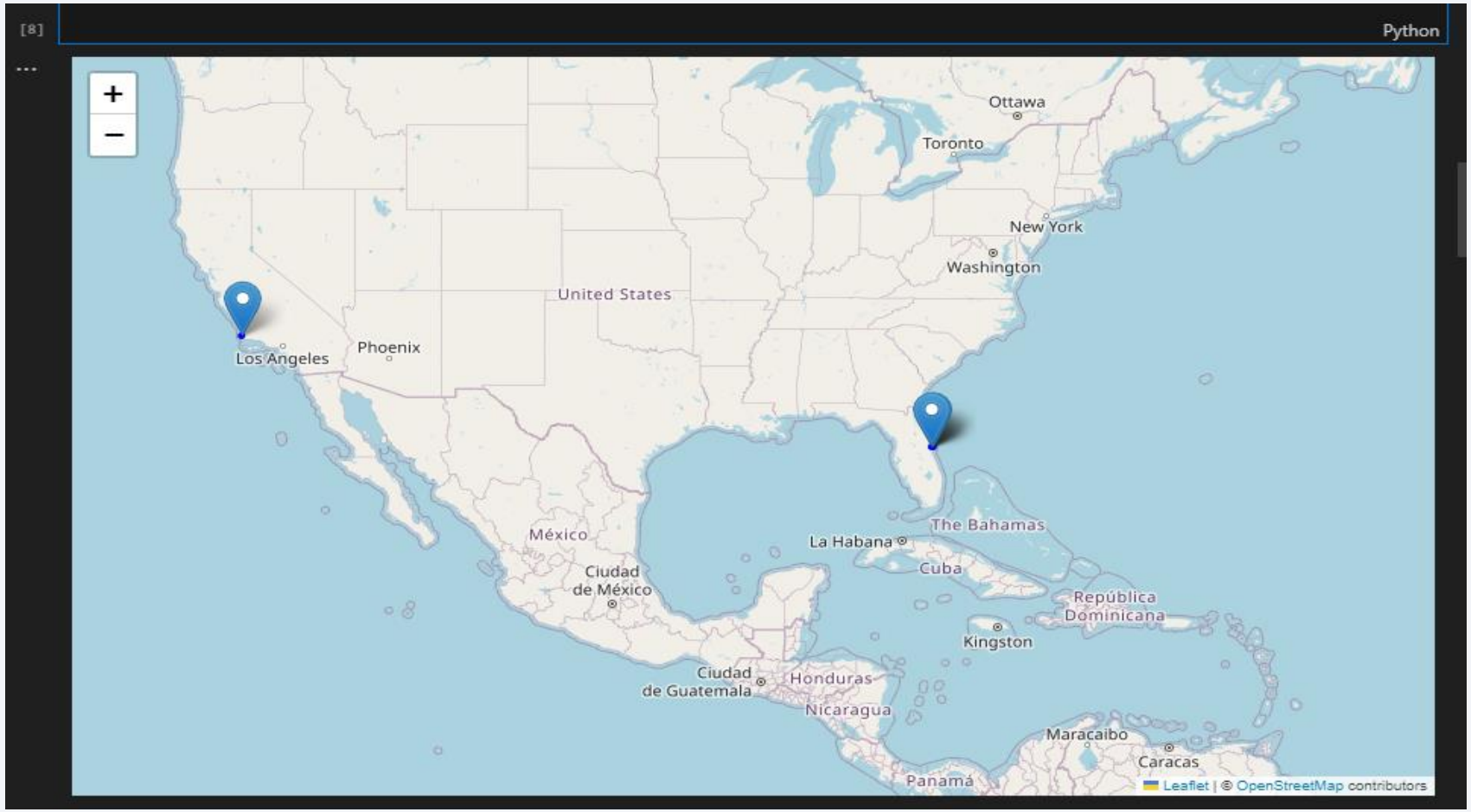
Landing_Outcome	outcome_count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

I used COUNT to count the number of occurrences, GROUP BY to arrange the data by a column and ORDER BY and DESC to arrange the count in descending order

Section 3

Launch Sites Proximities Analysis

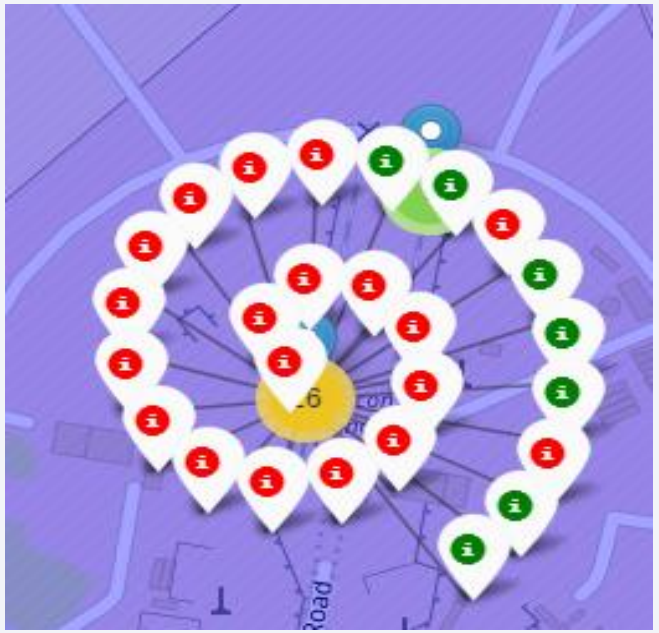
All Launch Site Location Markers



Color-labeled Launch Outcomes

VAFB SLC- 4E

CCAFSLC-40

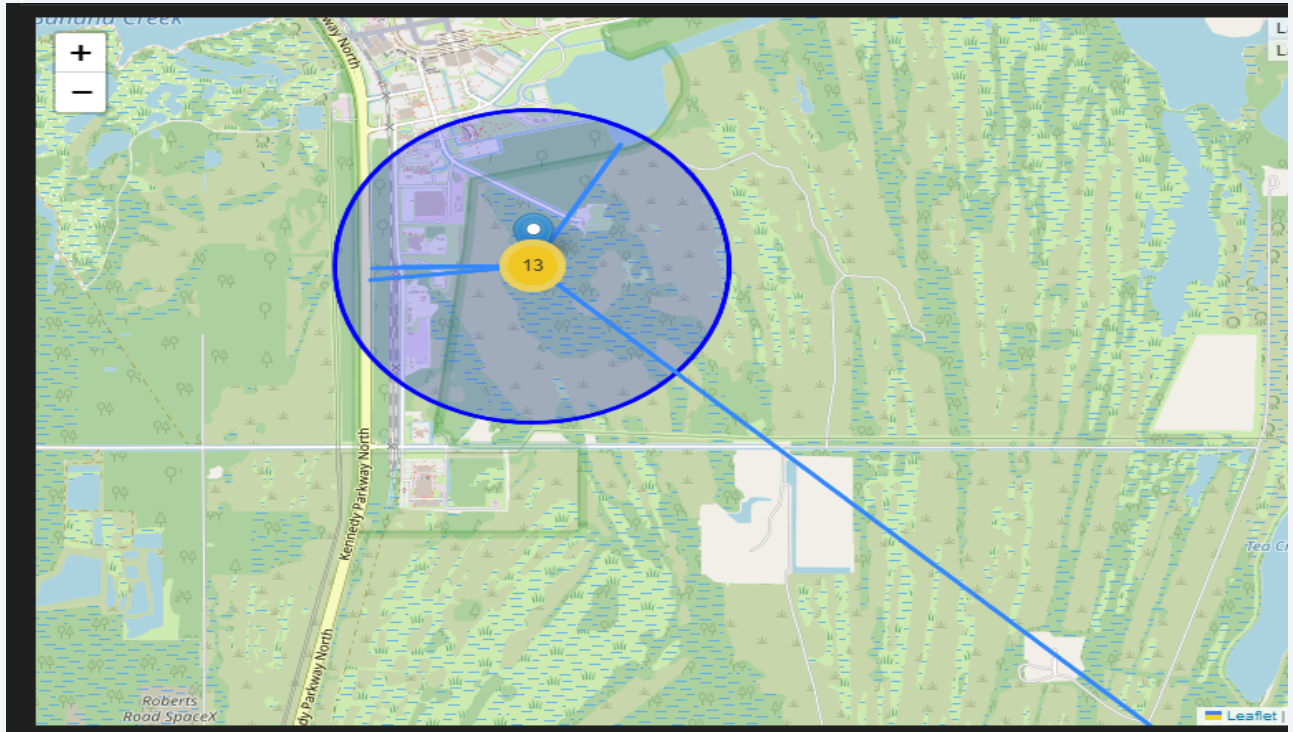


Key:

Green = Successful

Red = Failure

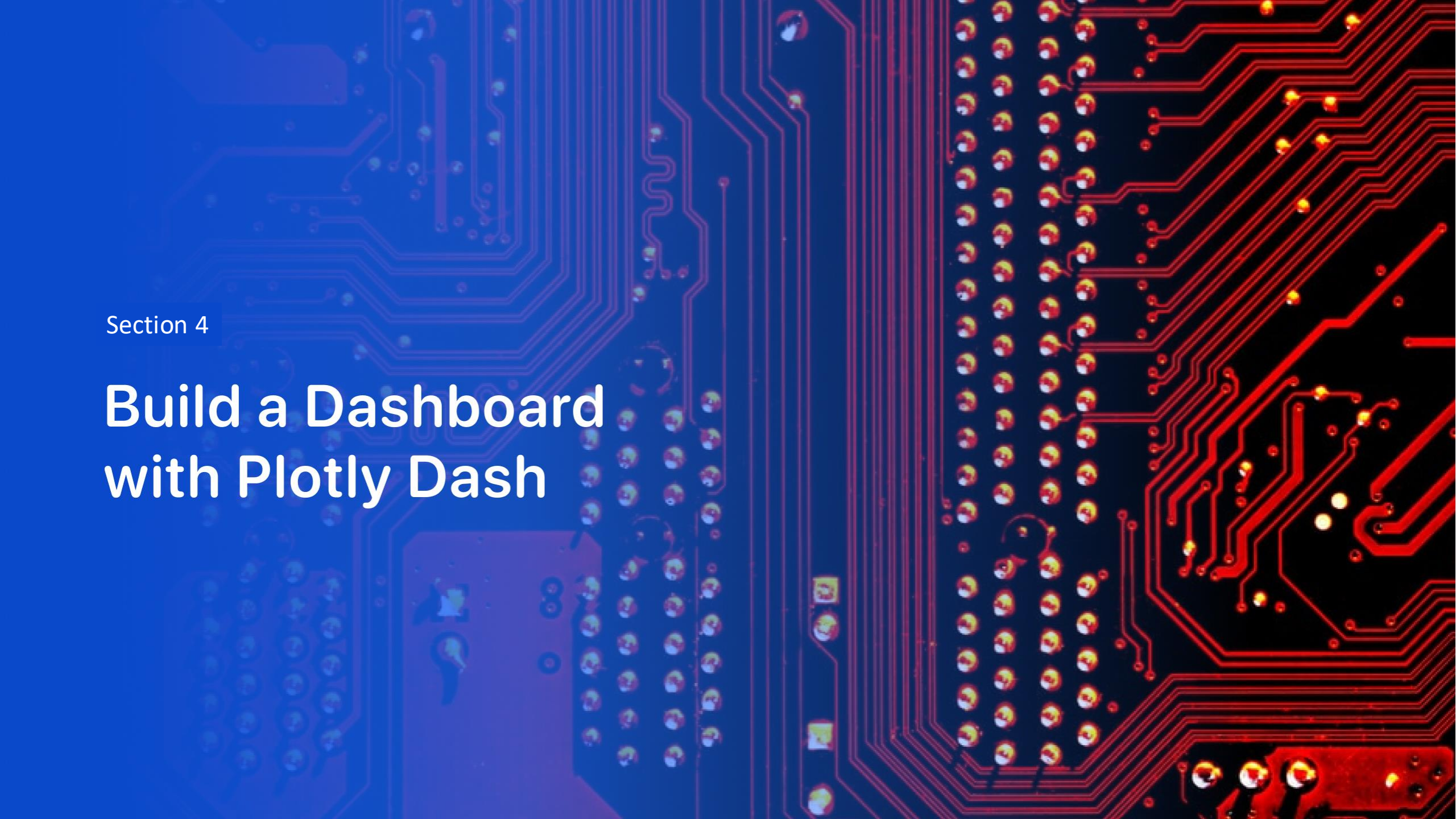
<Folium Map Screenshot 3>



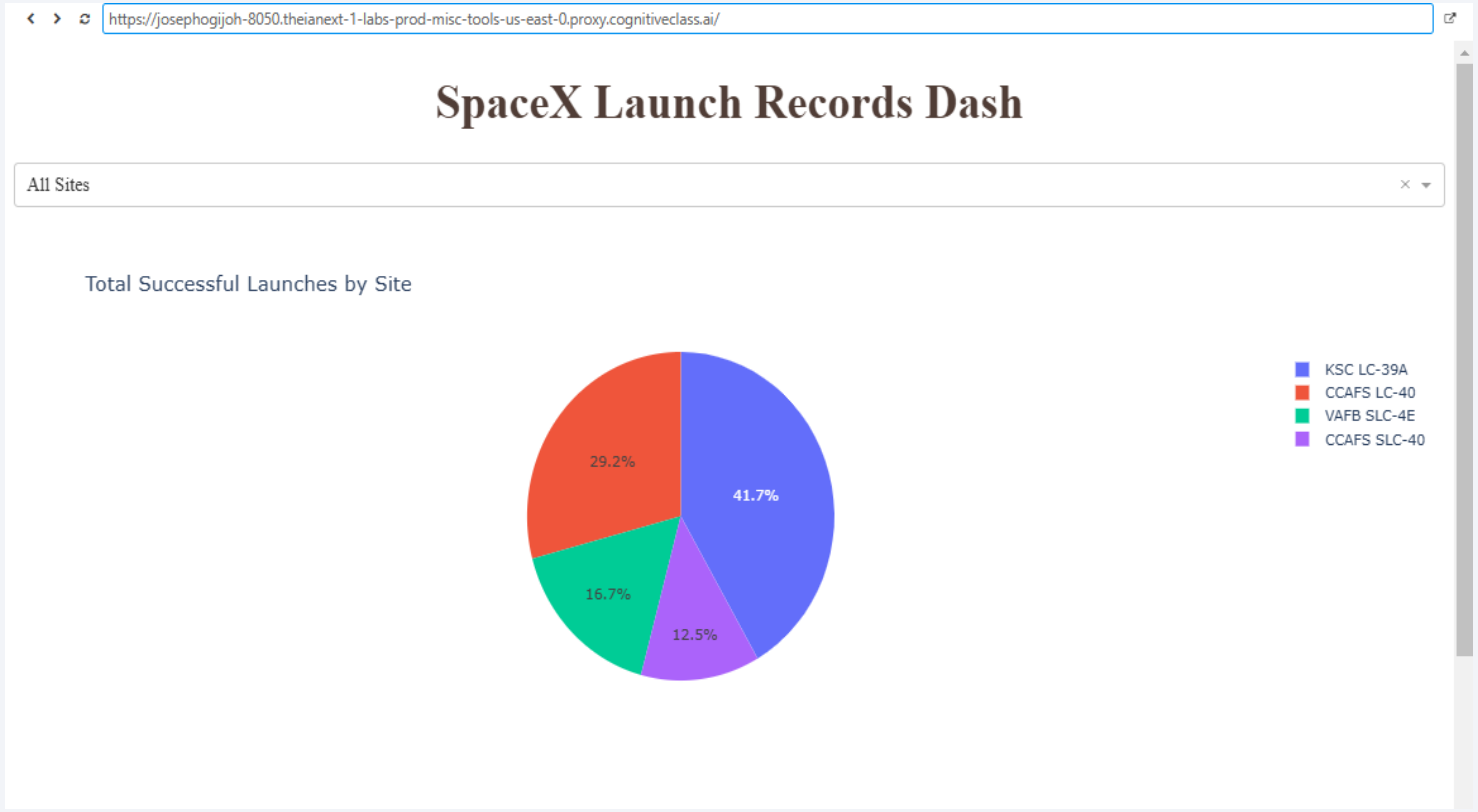

```
# find coordinate of the closet city
# e.g.,: Lat: 28.53989  Lon: -80.60645
city_lat = 28.57258
city_lon = -80.65149

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

```
0.4550896858875425
```

The launch station is closest to a city followed by a railway, then a highway and farthest from the coastline.

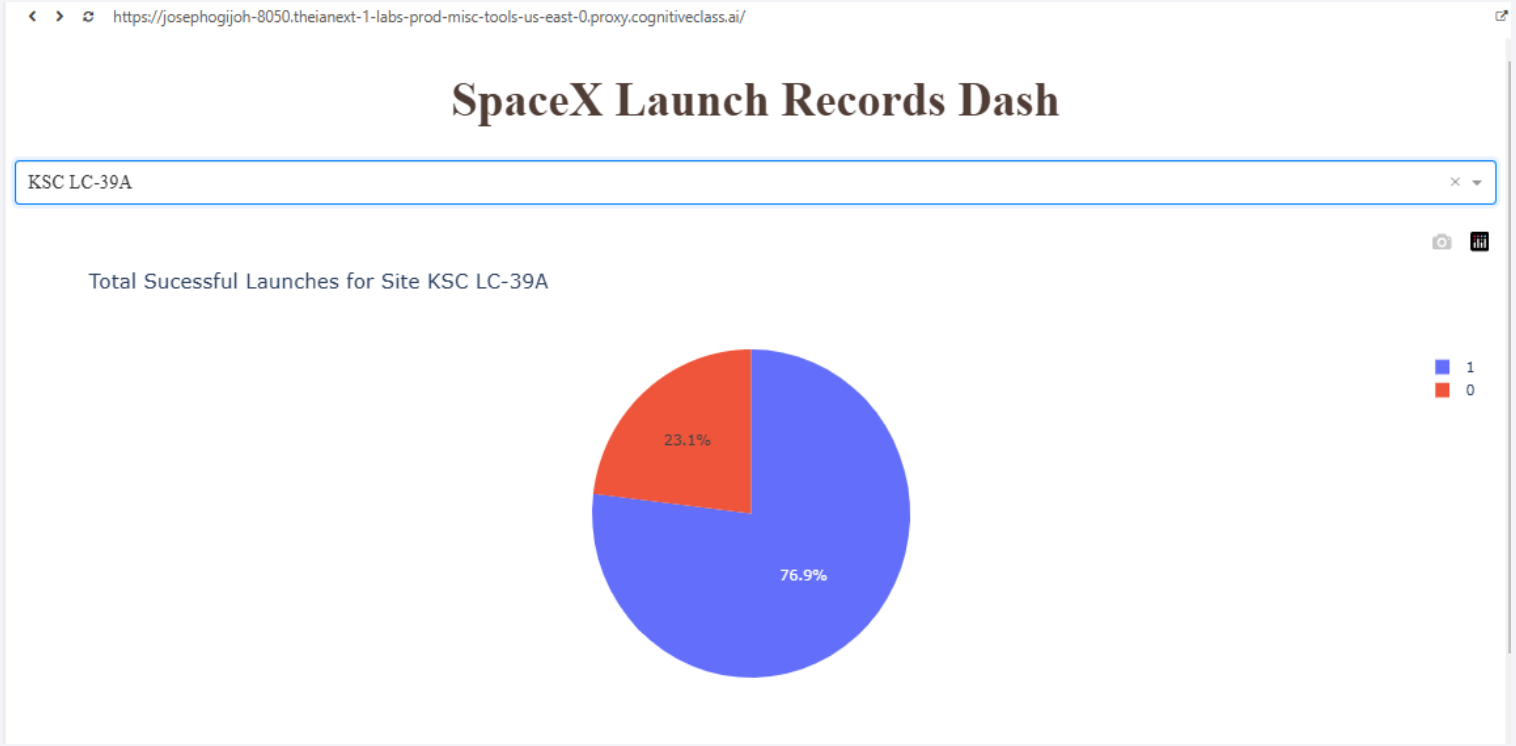

Section 4
Build a Dashboard with Plotly Dash

Launch Success Count for all sites



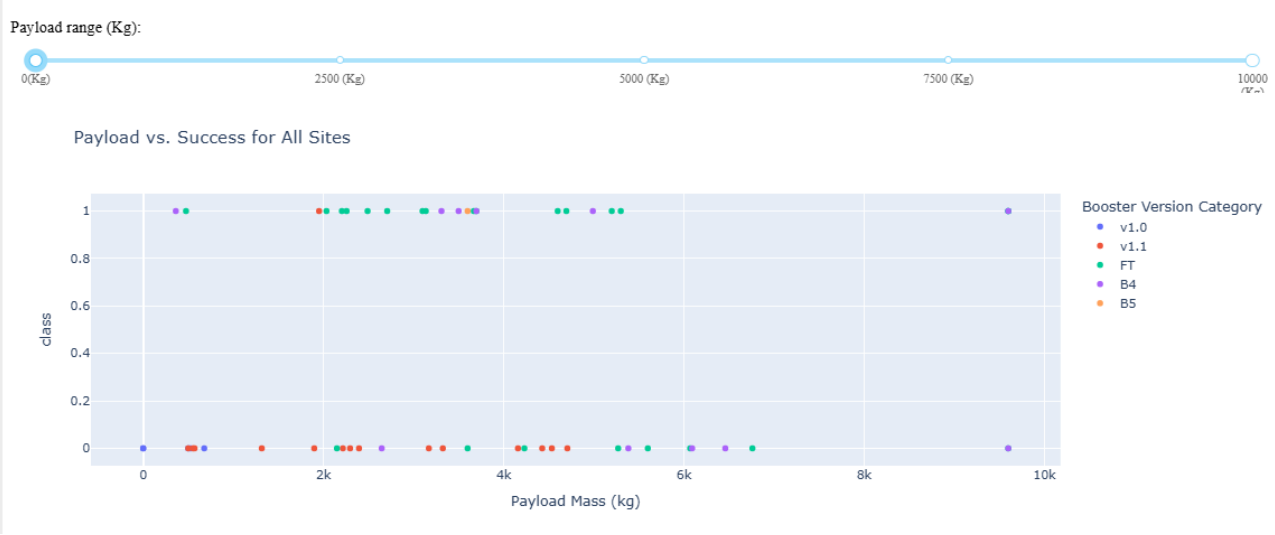
KSC LC-39A has the highest success score with 41.7%. This is followed by CCAFS LC-40 with 29.2% success rate and then VAFB SLC-4E and CCAFS SLC-40 with 16.7% and 12.5% respectively.

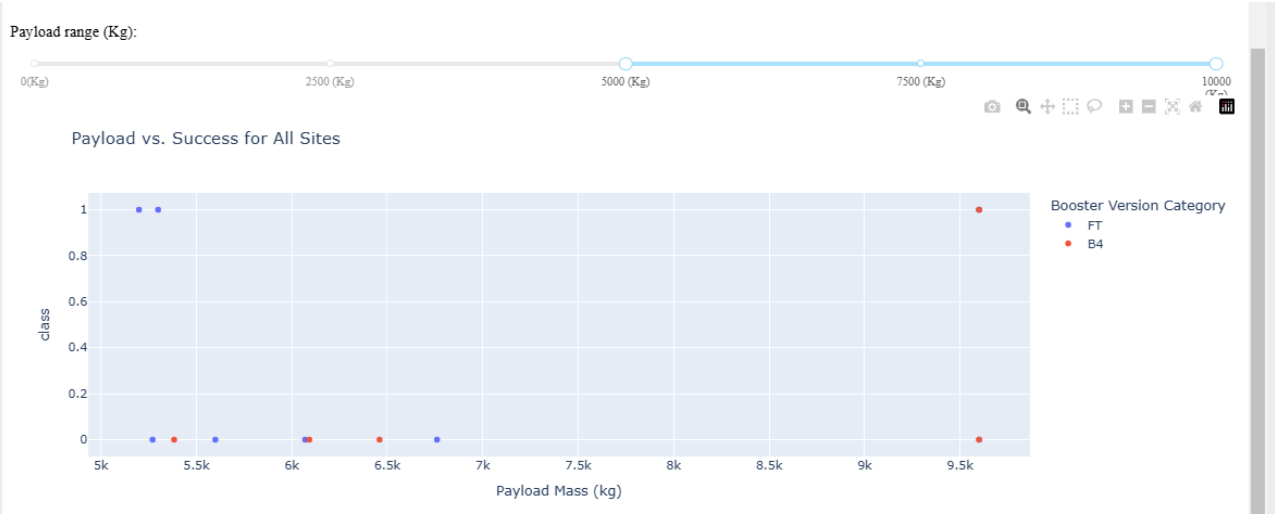
Launch site with the highest launch success ratio



KSC LC-39A has a high success ratio of 76.9%

Payload vs. Launch Outcome



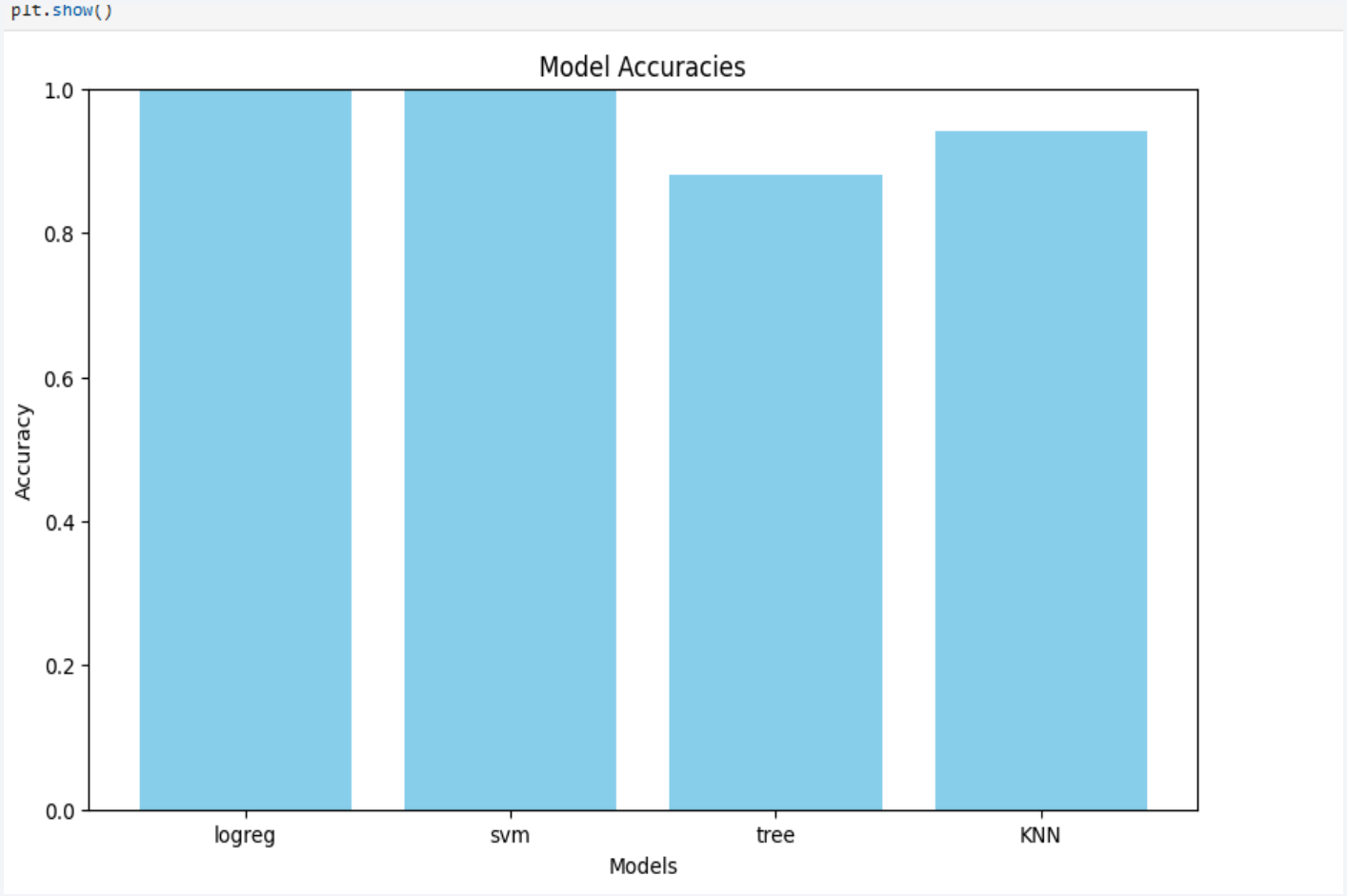


Payload range (0 – 10,000) kg for all sites. It is observed that booster version B4, FT and B5 has a high success rate at this range. Conversely, for the range (5,000 – 10,000)kg, both FT and B4 are seen to have low success rate.

Section 5

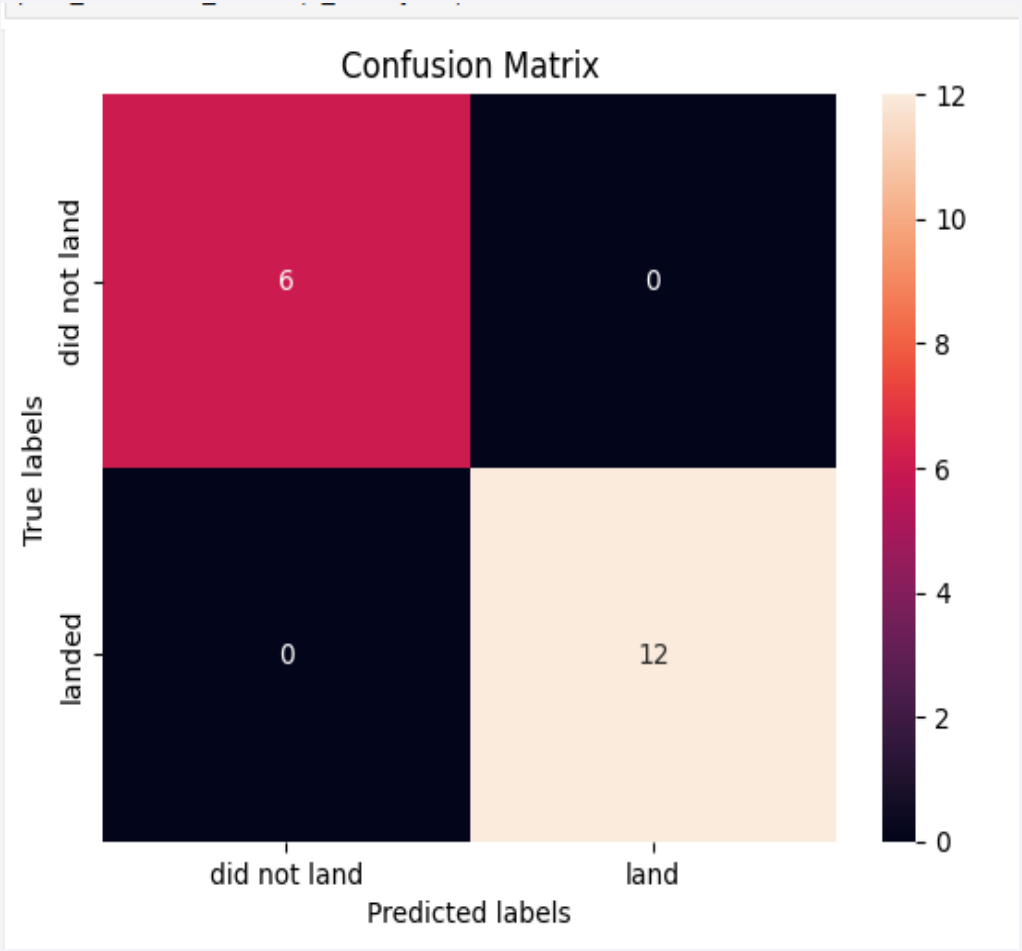
Predictive Analysis (Classification)

Classification Accuracy



Logistic regression and SVM models have the best accuracy of 100% followed by KNN with an accuracy of 94% and the model with the least accuracy is Decision tree with an accuracy of 88%

Confusion Matrix



Sensitivity = 1.00, formula: TPR = TP / (TP + FN)
Specificity = 1.00, formula: SPC = TN / (FP + TN)
Precision = 1.00, formula: PPV = TP / (TP + FP)
Accuracy = 1.00, formula: ACC = (TP + TN) / (TP + TN + FP + FN)
F1 Score = 1.00, formula: F1 = 2TP / (2TP + FP + FN)

Conclusions

- We found that the KSC LC-39A launch site has the highest success rate.
- Payload range of 0 – 5000 kg gave a better success and failure distribution information than 6000 – 10000 kg range.
- Logistic regression and SVM where found to be the best performing models with a perfect accuracy of 1.00.
- 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 all codes.

https://github.com/Manibe745/Data-science

Thank you!