Winning Space Race with Data Science

Billy Nguyen

30 June 2023

Outline

Executive Summary

Summary of methodologies	Data collection
	Data Wrangling
	EDA with data visualization
	EDA with SQL
	Interactive Map with Folium
	Dashboard with Plotly Dash
	Predictive Analysis (Classification)
Summary of all results	Exploratory Data Analysis (EDA) results
	Interactive Analytic Demo in Screenshots
	Predictive Analysis Results

Introduction

Project background and context summary:

- The commercial space industry has seen a surge in companies offering affordable space travel and satellite services. Among them, SpaceX stands out as a highly successful player, achieving significant milestones such as sending spacecraft to the International Space Station, launching the Starlink satellite internet constellation, and conducting manned missions to space. One of the key factors contributing to SpaceX's cost advantage is its ability to reuse the first stage of the Falcon 9 rocket. This report focuses on Space Y, a new rocket company founded by Billionaire industrialist Allon Musk, aiming to compete with SpaceX. The objective is to gather information about SpaceX, create informative dashboards, and predict whether SpaceX will reuse the first stage through machine learning models.

- In this project, we aim to address the following objectives and research questions:
- Objective: Price Determination for Each Launch
  - Research Question: How can we accurately determine the pricing of rocket launches, taking into consideration the impact of successful first stage landings?
- Objective: Predicting First Stage Reusability
  - Research Question: Can we develop a machine learning model using publicly available information to predict whether SpaceX will reuse the first stage of the Falcon 9 rocket?
- Focusing on these objectives and answering the corresponding research questions, we aim to provide Space Y with valuable insights and strategies to compete effectively with SpaceX in the commercial space industry.

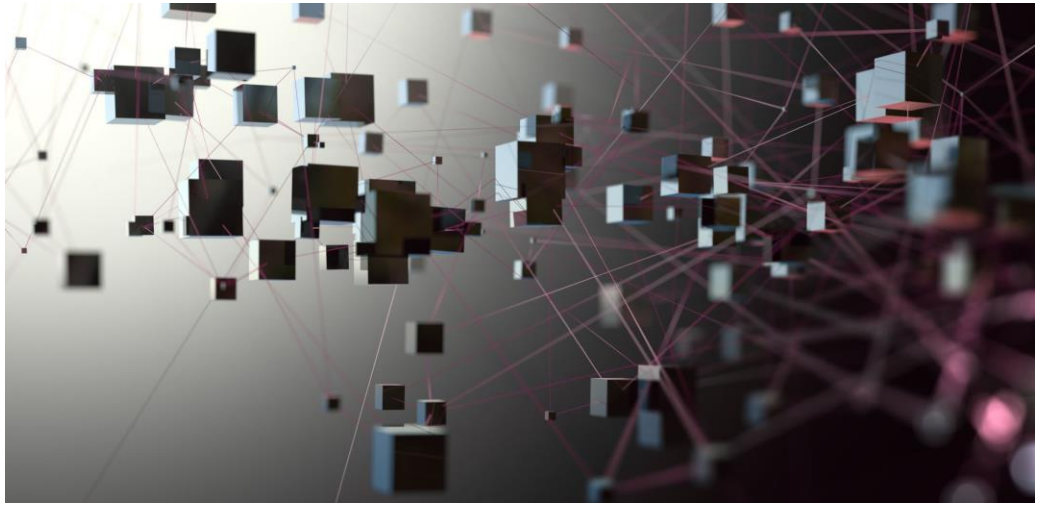
Section 1
Methodology

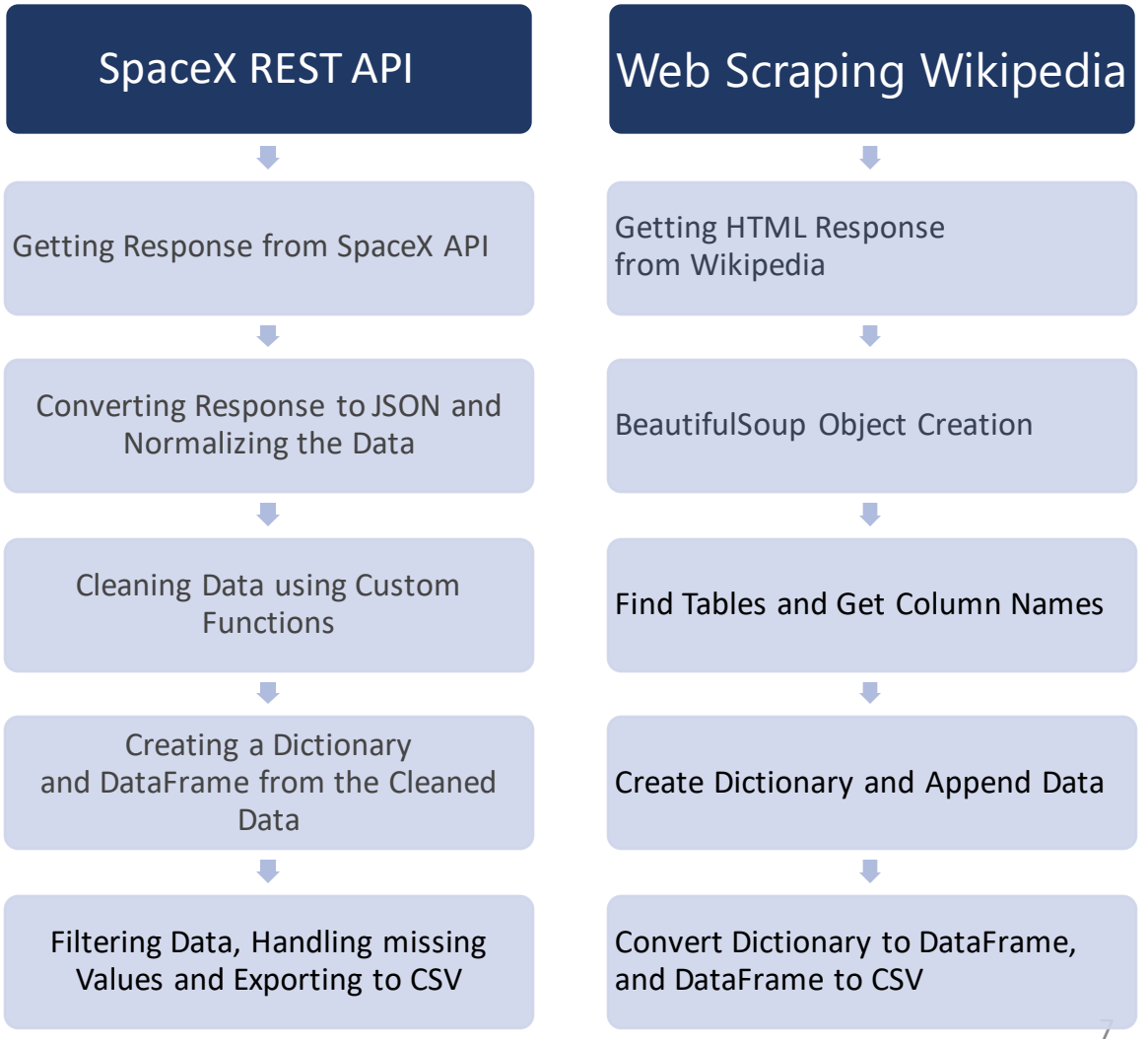
Methodology

Executive Summary	
Data collection methodology:	Utilized SpaceX REST API to gather detailed information about Falcon 9 launches, including launch outcomes and landing attempts.
	Conducted web scraping of Falcon Launch data from Wikipedia to supplement the API data and obtain additional launch details.
Perform data wrangling	Performed data wrangling techniques to normalize and clean the collected data.
	Identified missing data and applied appropriate strategies to handle missing values.
	Ensured consistency in data types and labeled the data appropriately for analysis.
Perform exploratory data analysis (EDA) using visualization and SQL	Utilized visualization techniques and SQL queries to conduct exploratory data analysis.
	Employed scatter plots and bar graphs to examine relationships between variables and identify data patterns.
Perform interactive visual analytics using Folium and Plotly Dash	Leveraged tools like Folium and Plotly Dash to create interactive visualizations for enhanced data exploration and analysis.
	Developed interactive maps and dynamic charts to visualize the launch sites, orbit types, and landing outcomes.
Perform predictive analysis using classification models	Implemented classification models to predict the landing outcome of Falcon 9 first stage.
	Built, tuned, and evaluated various classification models to identify the most accurate predictive model.
	Explored algorithms such as logistic regression, support vector machines (SVM), decision trees, and k-nearest neighbors (KNN).

Data Collection

The data collection process for this project involved gathering data from two primary sources: the SpaceX REST API and Web Scraping Wikipedia using BeautifulSoup. The following key phrases and flowchart provide an overview of the data collection process:





Data Collection – SpaceX REST API

GitHub URL to SpaceX REST API

Step 1

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

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

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

Step 2

```
response.status_code
```

Step 3

```
data = response.json()
df = pd.json_normalize(data)
```

Step 4

```
# Call getLaunchSite
getLaunchSite(df)

# Call getPayloadData
getPayloadData(df)

# Call getCoreData
getCoreData(df)

# Call getBoosterVersion
getBoosterVersion(df)
```

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}

# Create a data from launch_dict
df = pd.DataFrame(launch_dict)
```

Step 5

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = data[['BoosterVersion'] != 'Falcon 1']

#The .copy() method is used to create a new copy of the filtered data.
data_falcon9 = df[df['BoosterVersion'] != 'Falcon 1'].copy()

data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

```
data_falcon9.isnull().sum()

# Calculate the mean value of PayloadMass column
mean_value = data_falcon9['PayloadMass'].mean()
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'].fillna(value=mean_value, inplace=True)

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

1. Getting Response from SpaceX REST API.
2. Converting Response to JSON and Normalizing the Data.
3. Clean Data using Custom Functions.
4. Creating a Dictionary and DataFrame from the Cleaned Data.
5. Filtering Data, Handling missing Values and Exporting to CSV.

Data Collection – Web Scraping

GitHub URL to Web Scraping

Step 1

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

Step 2

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

Step 3

```
soup.find_all('table')
html_tables = soup.find_all('table')
```

```
column_names = []

th_list = first_launch_table.find_all('th')
```

```
for th in th_list:
    # Extract the column name from each <th> element
    name = extract_column_from_header(th)

    # Append non-empty column names to the list
    if name is not None and len(name) > 0:
        column_names.append(name)
```

Step 4

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with eac
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

```
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()
```

Step 5

```
#df=pd.DataFrame(launch_dict)
new_df = pd.DataFrame(launch_dict)
new_df["Launch outcome"]=new_df["Launch outcome"].str.replace("\n","")
new_df["Booster landing"]=new_df["Booster landing"].str.replace("\n","")
new_df.tail(10)
```

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

1. Getting HTML Response from WikiPedia.
2. BeautifulSoup Object Creation.
3. Find Tables and Get Column Names.
4. Create Dictionary and Append Data.
5. Convert Dictionary to DataFrame, and DataFrame to CSV.

Data Wrangling

GitHub URL to Data Wrangling

Step 1

```
df=pd.read_csv(dataset_part_1_csv)
```

Step 2

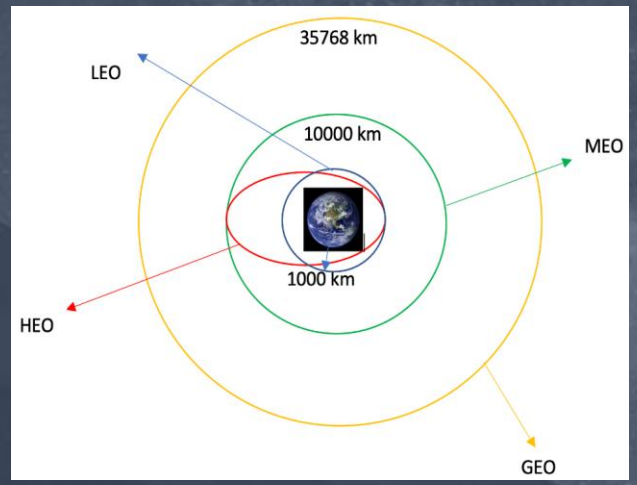
```
df.head(10)
df.isnull().sum()/df.shape[0]*100
df.types
```

Step 3

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

Step 4

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



Step 5

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

```
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
```

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

Step 4

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

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

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

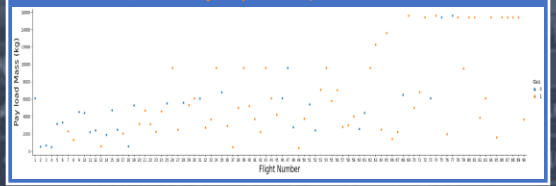
1. Load the dataset into a suitable data structure for analysis, such as a Pandas DataFrame.
2. Explore the data by examining its characteristics, structure, and key attributes.
3. Identify the number of launches from each launch site to understand the distribution and frequency of launches.
4. Calculate the number of launches for each orbit type to gain insights into the prevalence of different orbit types.
5. Assess Mission Outcomes by examining the count and occurrence of mission outcomes to understand the distribution of successful and unsuccessful landing outcomes.
6. Derive Training Labels, which represents the classification variable indicating the success or failure of the first stage landing.

Exploratory Data Analysis (EDA) with Data Visualization

[GitHub URL to EDA with Visualization]

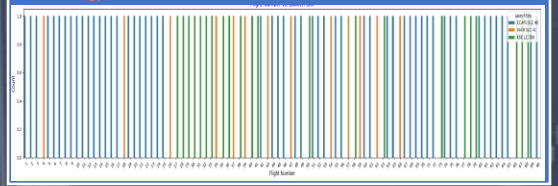
Flight number Vs Payload Mass Scatter Plot:

- *Purpose: To Analyse relationship between Flight Number and Payload Mass and how they affect launch outcomes.*
- *Code: sns.catplot(y="PayloadMass", x="FlightNumber", hue="Class", data=df, aspect = 5)*



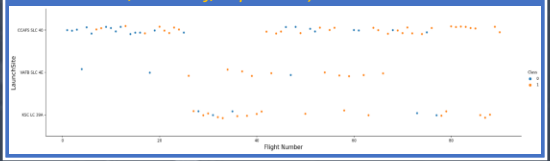
Flight Number Vs Launch Site Count Plot:

- *Purpose: To visualize the count of launch sites for each Flight Number.*
- *Code: sns.countplot(x="FlightNumber", hue="LaunchSite", data=df)*



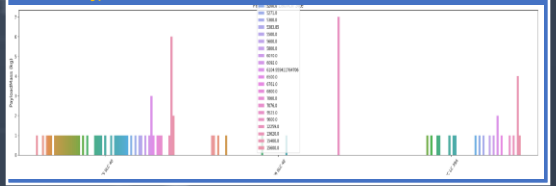
Flight Number Vs Launch Site Scatter Point Plot:

- *Purpose: To visualize the relationship between Flight Number and Launch Site.*
- *Code: sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 4)*



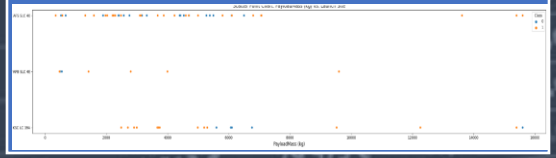
Launch Site Vs Payload Mass Count Plot:

- *Purpose: To visualize the count of launch sites for each Payload Mass.*
- *Code: sns.countplot(x="LaunchSite", hue="PayloadMass", data=df)*



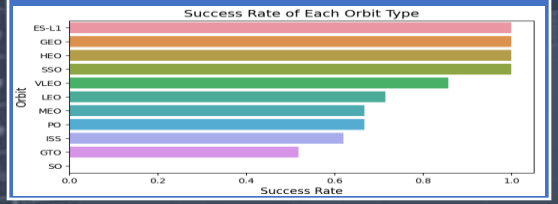
Payload Mass Vs Launch Site Scatter point Plot:

- *Purpose: To analyze the relationship between Payload Mass and Launch Site.*
- *Code: sns.scatterplot(x="PayloadMass", y="LaunchSite", hue="Class", data=df)*



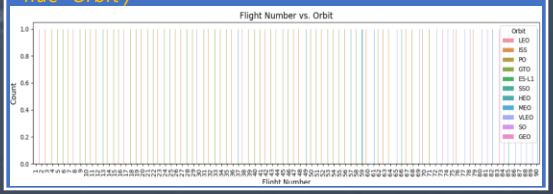
Success Rate of Each Orbit Type Bar plot:

- *Purpose: To visualize the success rate for each Orbit type.*
- *Code: sns.barplot(x='Class', y='Orbit', data=success_rate, color='blue')*



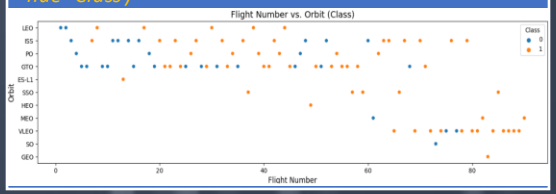
Flight Number Vs Orbit Count plot:

- *Purpose: To observe the relationship between Flight Number and Orbit type.*
- *Code: sns.countplot(data=df, x='FlightNumber', hue='Orbit')*



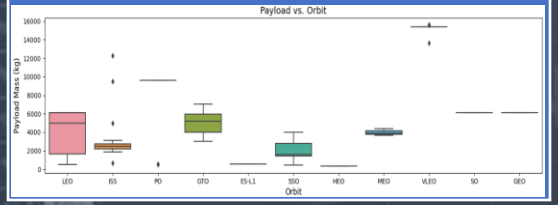
Flight Number Vs Orbit Scatter Point Plot:

- *Purpose: To analyze the relationship between Flight Number and Orbit type.*
- *Code: sns.scatterplot(data=df, x='FlightNumber', y='Orbit', hue='Class')*



Payload Vs Orbit Box plot:

- *Purpose: To visualize the relationship between Payload and Orbit type.*
- *Code: sns.boxplot(data=df, x='Orbit', y='PayloadMass')*



Success rate by Year Bar Chart:

- *Purpose: To observe the success rate trend over the years.*
- *Code: plt.bar(success_rate_by_year.index, success_rate_by_year.values)*

Exploratory Data Analysis (EDA) with SQL

Github URL to EDA with SQL

Display	Display the names of the unique launch sites in the space mission.
Display	Display 5 records where launch sites begin with the string 'CCA'
Display	Display the total payload mass carried by boosters launched by NASA (CRS)
Display	Display the average payload mass carried by booster version F9 v1.1
List	List the date when the first successful landing outcome in ground pad was achieved
List	List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
List	List the total number of successful and failure mission outcomes
List	List the names of the booster versions which have carried the maximum payload mass using a subquery
List	List the month, landing_outcome, booster_version, and launch_site for failed missions in 2015 with a landing outcome of 'Failure (drone ship)'
Rank	Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order

GitHub URL to Interactive Interactive Map with Folium with Folium

Build an Interactive Map with Folium

- The following objects were created and added to folium map to enhance the visualization and provide useful information about the launch site's proximity to various features on the map:
    - Markers were added to give representation of the launch sites location on the map.
    - Distance Markers were added to represent the distances between the launch sites and specific points of interest, such as coastline, highway, railways, and city. Distance Markers provide information about the proximity of the launch site to these features and the exact distance between them.
    - PolyLines were added to draw lines between the launch site and the closest points of interest, such as coastline, highway, railways, and city. PolyLines provide visualization of the spatial relationships and distances between the launch site and these features. They help provide the direct route from the launch site to the features and the exact distances between them.
- Adding these objects to the map provides a comprehensive view of the launch site's location relative to railways, highways, coastlines, and cities. These visual elements aid in analyzing the spatial relationships and distances between the launch site and the surrounding infrastructure, contributing to a better understanding of the launch site's accessibility and proximity to key facilities.

Build a Dashboard with Plotly Dash

GitHub URL to Dashboard with Plotly Dashboard with Plotly Dash

- In the dashboard, the following plots/graphs and interactions were added:
  - Launch Site Drop-down – This menu component allows users to select different launch sites. This interactions enables users to focus on specific launch sites and compare their success rates.
  - Success Pie Chart – Pie chart provide a visual on success counts for each launch site. This chart provides an overview of the success distribution among different sites and helps identify the site with the largest successful launches.
  - Payload Range Slider – Range slider component allows users to select different payload ranges. This interactions enables users to analyze the relationship between payload and launch success rates.
  - Success Payload Scatter Chart - Scatter chart plots the payload mass (x-axis) against the launch outcomes (y-axis). Scatter points are colour-labeled based on the F9 Booster version. This chart provide a visual representations of how payload mass and booster version correlate with launch success.
- These plots and interactions were added to facilitate interactive visual analytics on the SpaceX launch data. Drop-down menu allows users to select specific launch sites, while pie chart provides an overview off success counts. Range slider enables users to explore different payload ranges and observe their impacts on launch success. Finally, scatter plot chart combines multiple variables (payload, launch outcome, booster version) to provide a comprehensive view of their relationships.
- Incorporating these plots and interactions, users can easily analyze and gain insights from SpaceX launch data, such as identifying the most successful launch site, understanding the impacts of payload range on success rates, and examining the performance of different F9 booster versions.

Predictive Analysis (Classification)

GitHub URL to Machine Learning PredictionMachine Learning Prediction

Following these steps iteratively and refining the models based on evaluation results, the aim is to find the most accurate classification model with the best combination of hyperparameters for the given task

Data Preparation	• Convert the target column into a NumPy array and standardize the feature data using the *StandardScaler* to ensure all features have the same scale.
Train-Test Split	• Split the data into training and test sets, 80% for training and 20% of data reserved for evaluation or test sets.
Logistic Regression	• Create a logistic regression model and perform hyperparameter tuning using cross-validation. Find the best combination of hyperparameters (*C*, *penalty*, and *solver*) that maximizes the model's performance.
Support Vector Machine (SVM)	• Create an SVM model and perform hyperparameter tuning using cross-validation. Explore different options for *kernel*, *C*, and *gamma* to find the best performing configuration.
Decision Tree	• Create a decision tree classifier and perform hyperparameter tuning using cross-validation. Search for optimal values of *criterion*, *splitter*, *max_depth*, *max_features*, *min_samples_leaf*, and *min_samples_split* to improve the model's accuracy.
K-Nearest Neighbors (KNN)	• Create a KNN model and perform hyperparameter tuning using cross-validation. Experiment with various values of *n_neighbors*, *weights*, and *p* to identify the optimal settings for the model.
Confusion Matrix	• Generate predictions using each model on the test data and create a confusion matrix to evaluate the model's performance. This matrix provides insights into the number of true positives, true negatives, false positives, and false negatives.
Model Comparison	• Compare the accuracy scores of different models and select the one with the highest accuracy as the best performing model. Visualize the accuracy of each model using a bar graph to make it easier to compare their performance.

Results

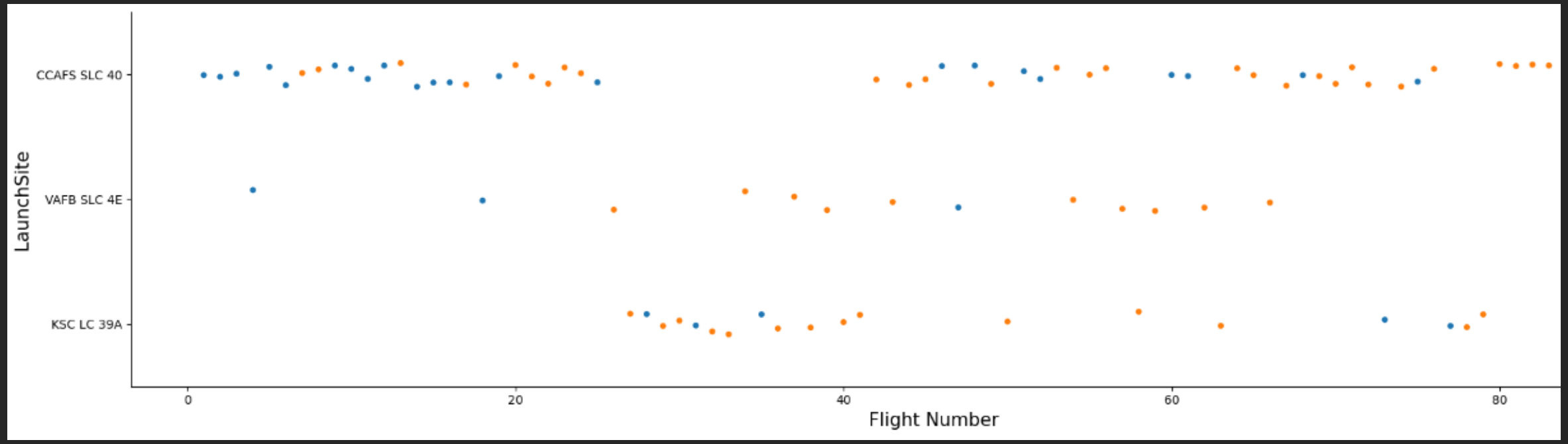
EXPLORATORY DATA ANALYSIS RESULTS

INTERACTIVE ANALYTICS DEMO IN SCREENSHOTS

PREDICTIVE ANALYSIS RESULTS

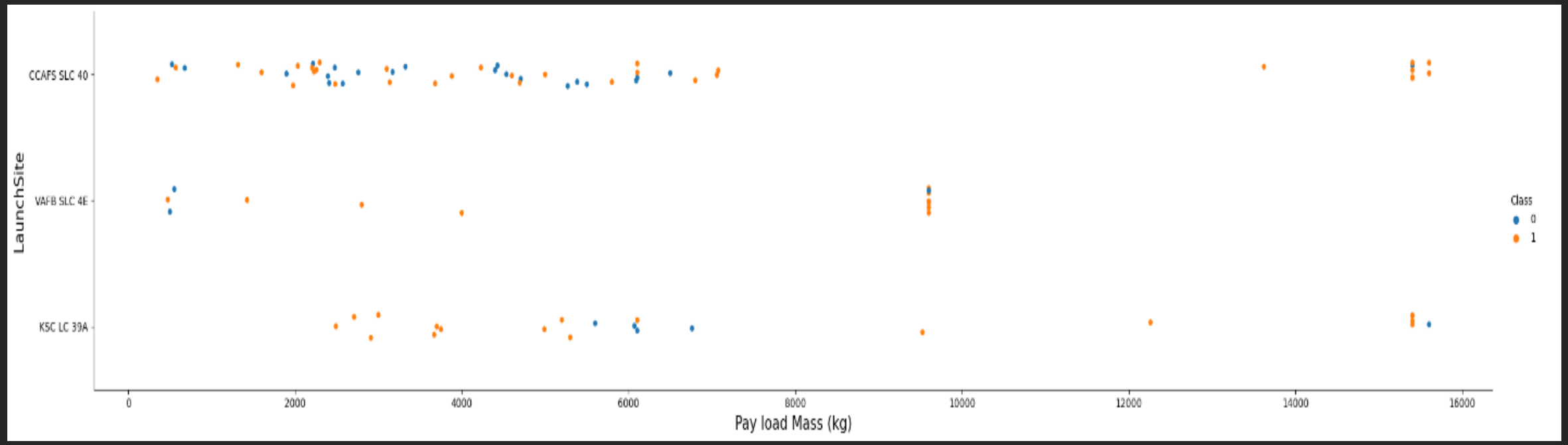
Section 2

Insights drawn from EDA



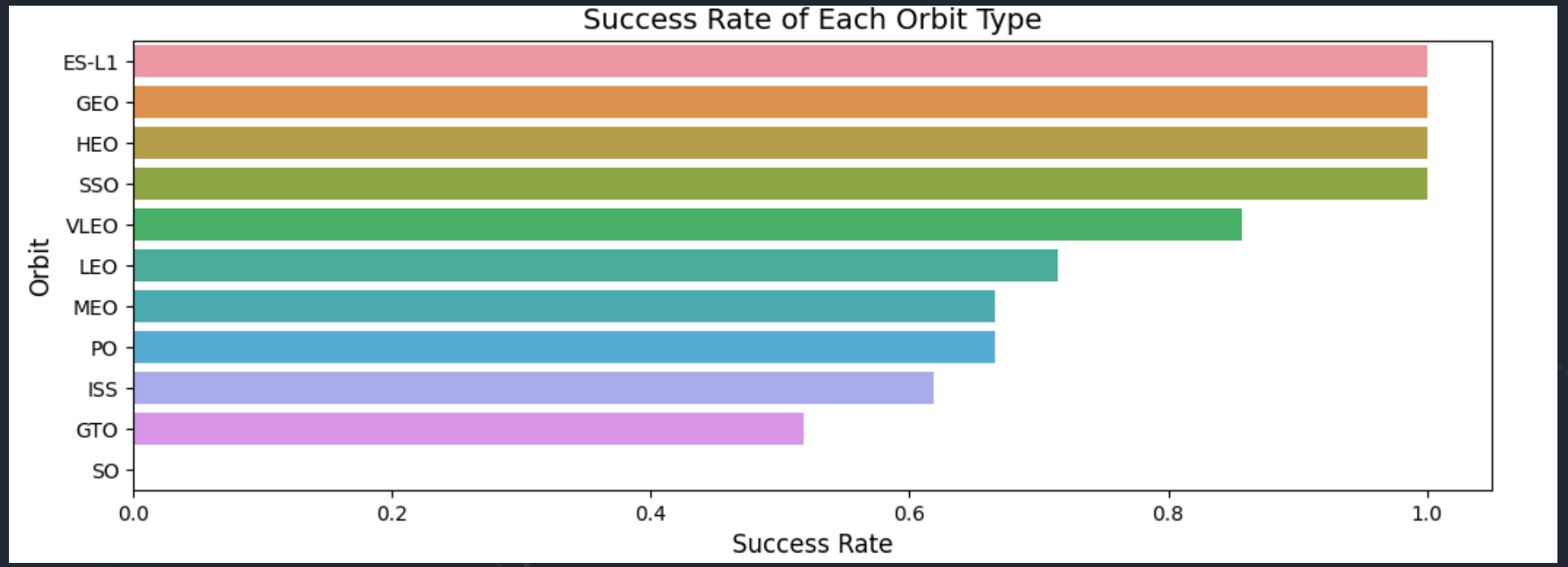
Flight Number vs. Launch Site

- The LaunchSite Vs. Flight Number Scatter Plot shows a positive correlation between Launchsite and Flight Number.
- CCAFS SLC 40 achieved 100% success rate when flight numbers increased above 80.
- VAFB SLC 4E achieved 100% success rate when flight numbers increased above 50.
- KSC LC 39A achieved 100 % success rate when flight numbers is between 35 - 65.



Payload vs. Launch Site

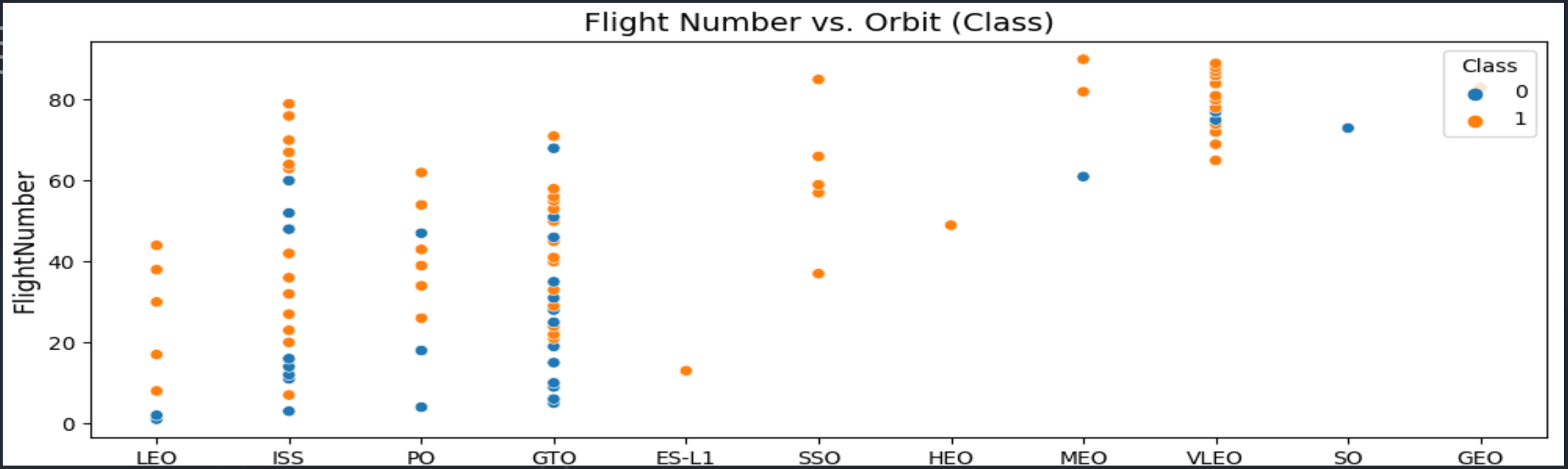
- The Payload Vs. LaunchSite Scatter Plot shows there is a negative correlation between Payload and LaunchSite for rocket Launch at launchsite for certain payload mass.
- Success rate at CCAFS SLC 40 LaunchSite only when payload mass less than 8000kg and greater than 14000kg.
- Success rate at VAFB SLC 4E Launchsite only for payload less than 10000kg.
- Success rate at KSC LC 39A is more frequently when payload is less than 8000kg and greater 14000kg.



Success Rate vs. Orbit Type

The bar chart shows the following to be the best successful orbits:

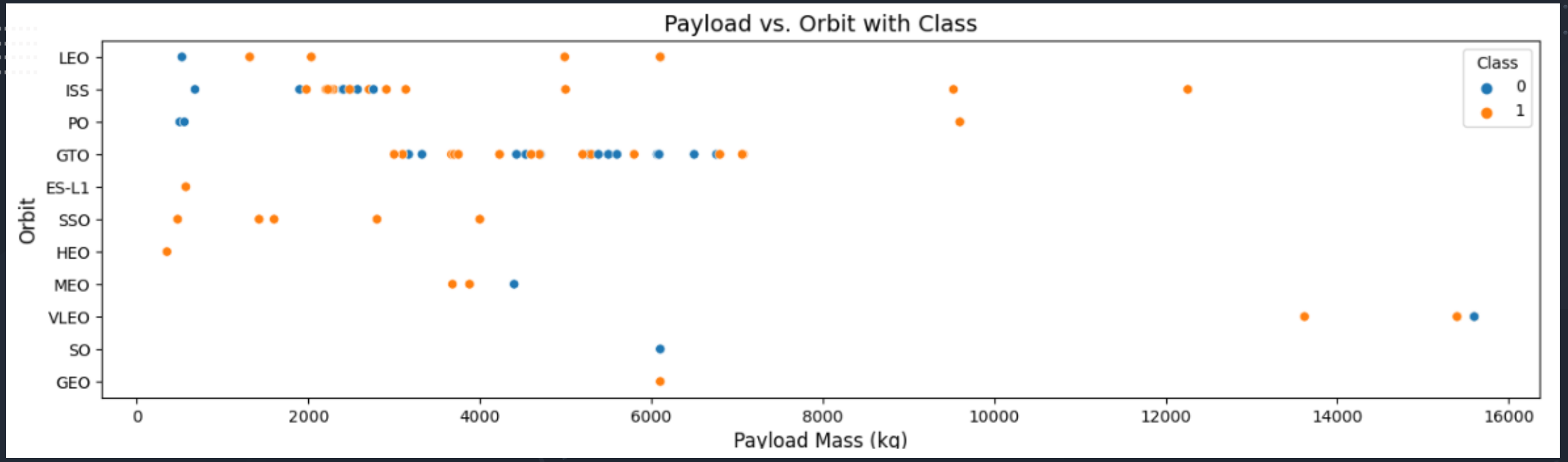
- ✓ ES-L1
- ✓ GEO
- ✓ HEO
- ✓ SSO



Payload vs. Orbit Type

The following can be visualized in the Flight Number Vs. Orbit scatter plot:

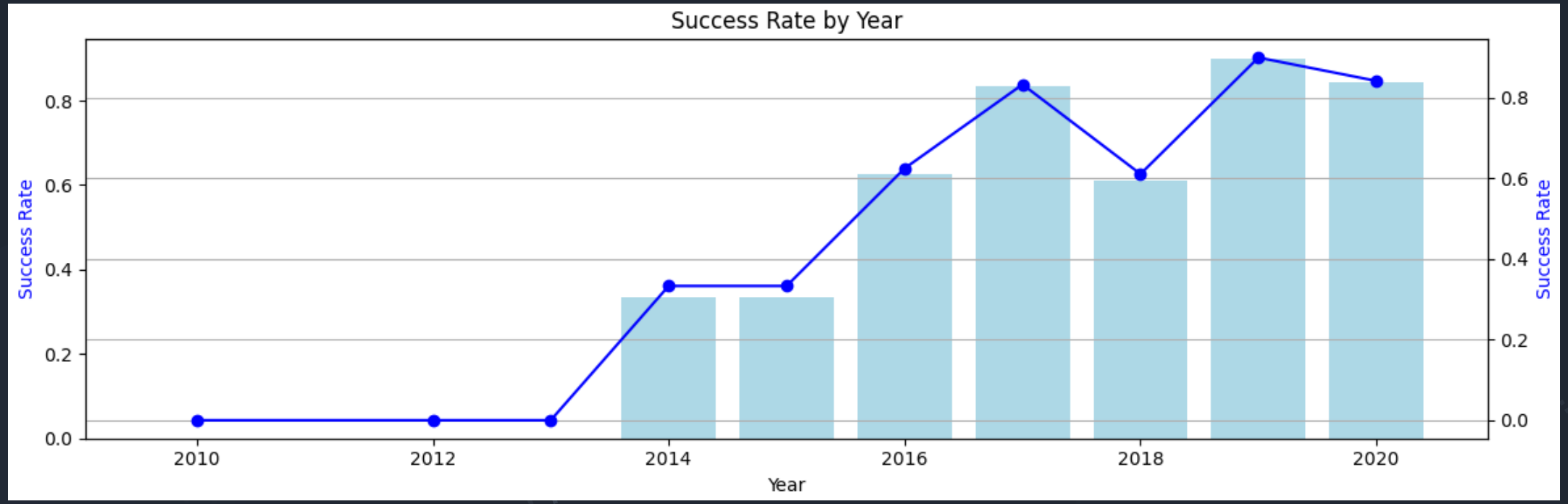
- Success appears to be related to flight number in LEO, MEO, and SSO orbit, when flight numbers are above 10, 60, and 40 respectively.
- Success not related to flight number in the ISS, PO, GTO, and SO orbits.



Payload vs. Orbit Type

The following can be visualized in the Flight Number Vs. Orbit scatter plot:

- Success appears to be related to flight number in LEO, MEO, and SSO orbit, when flight numbers are above 10, 60, and 40 respectively.
- Success not related to flight number in the ISS, PO, GTO, and SO orbits.



Launch Success Yearly Trend

The success rate shows an increasing trends from 2013 to 2020.

All Launch Site Names

This SQL command is used to retrieves all the unique/distinct values from the "launch_site" column of the "SPACEXTBL" table. The following is a breakdown of this command.

- SELECT DISTINCT : Select only the unique/distinct values from the column.
- (launch_site) : The column to retrieve the distinct values from.
- FROM SPACEXTBL : Specifies the table("SPACEXTBL") from which the data is retrieved.

```
%sql select distinct(launch_site) from SPACEXTBL;
```

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

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

```
%sql select * from SPACEXTBL where launch_site like 'CCA%' limit 5;
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
06/04/2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0.0	LEO	SpaceX	Success	Failure (parachute)
12/08/2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0.0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
22/05/2012	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525.0	LEO (ISS)	NASA (COTS)	Success	No attempt
10/08/2012	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500.0	LEO (ISS)	NASA (CRS)	Success	No attempt
03/01/2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677.0	LEO (ISS)	NASA (CRS)	Success	No attempt

Launch Site Names Begin with 'CCA'

This SQL command is used to fetches the first 5 records from the "SPACEXTBL" table where the launch site starts with 'CCA'.

- SELECT * : Queries/Retrieve data from database (my_data1.db). * is a wildcard character to represents all column in the table.
- FROM SPACEXTBL : Specifies the table("SPACEXTBL") from which the data is retrieved.
- WHERE launch_site LIKE 'CCA%' : Filter the row where the value in the "launch_site" column start with "CCA."
- LIMIT 5 : Only display the first 5 rows of the matched results.

```
%sql select sum(payload_mass__kg_) from SPACEXTBL where customer = 'NASA (CRS)';
```

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

sum(payload_mass__kg_)
45596.0

Total Payload Mass

This SQL command is used to query the "SPACEXTBL" table and calculating the sum of the 'payload_mass__kg_' columns for rows where the 'CUSTOMER' is 'NASA (CRS)'. The return will be a single value, which is the sum of payload masses for NASA's CRS missions.

- SELECT SUM(payload_mass__kg_) : Specifies the sum of all the values in columns payload_mass__kg_ from the table.
- FROM SPACEXTBL : Specifies the table("SPACEXTBL") from which the data is retrieved.
- WHERE CUSTOMER = 'NASA (CRS)' : Filter rows based on the value in the "CUSTOMER" column. Only those rows where 'CUSTOMER' is equal to 'NASA (CRS)' will be considered for calculation.

```
%sql select avg(payload_mass__kg_) from SPACEXTBL where booster_version like 'F9 v1.1%';
```

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

avg(payload_mass__kg_)
2534.6666666666665

Average Payload Mass by F9 v.1.1

This SQL command is used to retrieve the average value of the 'payload_mass__kg' column from the 'SPACEXTBL' table, specifically for rows where the 'booster_version' column starts with F9 v1.1.

- SELECT AVG(payload_mass__kg_) : Retrieve the average value of all data from database in the 'payload_mass__kg_' column.
- FROM SPACEXTBL : Specifies the table("SPACEXTBL") from which the data is retrieved.
- WHERE BOOSTER_VERSION like 'F9 v1.1%' : Filter rows based on the condition that the values in the 'BOOSTER_VERSION' column starts with 'F9 v1.1'.

```
%sql select min(DATE) from SPACEXTBL where landing_outcome = 'Success (ground pad)';
 * sqlite:///my_data1.db
Done.
```

min(DATE)
01/08/2018

First Successful Ground Landing Data

This SQL command is used to retrieve the minimum date value from the 'DATE' column in the 'SPACEXTBL' table, but only for rows where the 'landing_outcome' column has the value 'Success (ground pad)'.

- SELECT MIN(DATE) : retrieve the minimum value of the 'DATE' column in the result set.
- FROM SPACEXTBL : specifies the table("SPACEXTBL") from which the data is retrieved.
- WHERE LANDING_OUTCOME = 'Success (ground pad)' : Filter rows in the 'LANDING_OUTCOME' column that has a value 'Success (ground pad)'

```
%sql select distinct(booster_version) from SPACEXTBL where landing_outcome = 'Success (drone ship)' AND (payload_mass__kg_ > 4000 AND payload_mass__kg_ < 6000);
```

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

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

Successful Drone Ship Landing with payload between 4000 and 6000.

This SQL command is used to retrieves the distinct values of the booster_version column from the SPACEXTBL table for rows that meet the specified conditions of a successful landing outcome and a payload mass between 4000 and 6000 kilograms.

- SELECT DISTINCT(BOOSTER_VERSION) : Retrieve distinct value from the 'BOOSTER_VERSION' column. The DISTINCT keyword ensures that only unique values are returned, eliminating any duplicates.
- FROM SPACEXTBL : specifies the table("SPACEXTBL") from which the data is retrieved.
- WHERE LANDING_OUTCOME = 'Success (ground pad)' : Filter rows in the 'LANDING_OUTCOME' column that has a value 'Success (ground pad)'

```
%sql select  mission_outcome,count(*) as Counter from SPACEXTBL group by mission_outcome;
 * sqlite:///my_data1.db
Done.
```

Mission_Outcome	Counter
None	898
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Total Number of Successful and Failure Mission Outcomes.

This SQL command is used to retrieves data from the SPACEXTBL table and provides a count of records for each unique value in the mission_outcome column. The result will include distinct mission_outcome values and their corresponding counts, allowing for the analysis of the distribution of mission outcomes in the dataset.

- SELECT MISSION_OUTCOME, COUNT(*) as COUNTER : Specifies the count of records in the result set in the 'MISSION OUTCOME' columnn.
- FROM SPACEXTBL : specifies the table("SPACEXTBL") from which the data is retrieved.
- GROUP BY MISSION_OUTCOME : Group records based on 'MISSION_OUTCOME' column. Ensure count is performed for each unique value of 'MISSION_OUTCOME'.

```
%sql select distinct(booster_version) from SPACEXTBL where payload_mass__kg_  = (select max(payload_mass__kg_) from SPACEXTBL);
 * 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

Booster Carried Maximum Payload.

This SQL command retrieves the distinct values of the "booster_version" column from the "SPACEXTBL" table where the payload mass is equal to the maximum payload mass in the table.

- SELECT DISTINCT(booster_version) : Selects the unique values from the "booster_version" column. The DISTINCT keyword ensures that only distinct values are returned, removing any duplicates..
- FROM SPACEXTBL : specifies the table("SPACEXTBL") from which the data is retrieved.
- WHERE payload_mass__kg_ = (SELECT MAX(payload_mass__kg_) FROM SPACEXTBL) : Filters the rows based on the payload mass. Compares the value of the "payload_mass__kg_" column with the maximum payload mass value obtained from the subquery.
    - The subquery (SELECT MAX(payload_mass__kg_) FROM SPACEXTBL) retrieves the maximum payload mass value from the same table..

```
%sql select substr(Date, 4, 2) As month, landing_outcome, booster_version, launch_site from SPACEXTBL where substr(Date, 7, 4) = '2015' AND landing_outcome = 'Failure (drone ship)'
```

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

month	Landing_Outcome	Booster_Version	Launch_Site
10	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

2015 Launch Records.

This SQL command selects the month, landing outcome, booster version, and launch site columns from the "SPACEXTBL" table, specifically for the year 2015 and where the landing outcome is marked as 'Failure (drone ship)'.

- SELECT SUBSTR(Date, 4, 2) As month, landing_outcome, booster_version, landing_site : Specifies the columns to retrieve from the table.
  - substr(Date, 4, 2) AS month: Extracts the month from the "Date" column and labels it as "month".
  - landing_outcome: Retrieves the "landing_outcome" column.
  - booster_version: Retrieves the "booster_version" column.
  - launch_site: Retrieves the "launch_site" column.
- FROM SPACEXTBL : specifies the table("SPACEXTBL") from which the data is retrieved.
- WHERE SUBSTR(Date, 7, 4) = '2015' AND landing_outcome = 'Failure (drone ship)' : Filters the rows based on specified conditions.
  - substr(Date, 7, 4) = '2015': Selects rows where the year part of the "Date" column is '2015'.
  - landing_outcome = 'Failure (drone ship)': Selects rows where the "landing_outcome" column has the value 'Failure (drone ship)'.

```
%sql SELECT *, COUNT(Landing_Outcome) FROM SPACEXTBL WHERE DATE BETWEEN '04-06-2010' AND '20-03-2017' GROUP BY Landing_Outcome ORDER BY COUNT(Landing_Outcome) DESC
 * sqlite:///my_data1.db
Done.
```

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome	COUNT(Landing_Outcome)
08/07/2018	5:18:00	F9 B5 B1046.2	CCAFS SLC-40	Merah Putih	5800.0	GTO	Telkom Indonesia	Success	Success	20
10/08/2012	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500.0	LEO (ISS)	NASA (CRS)	Success	No attempt	10
04/08/2016	20:43:00	F9 FT B1021.1	CCAFS LC-40	SpaceX CRS-8	3136.0	LEO (ISS)	NASA (CRS)	Success	Success (drone ship)	8
18/07/2016	4:45:00	F9 FT B1025.1	CCAFS LC-40	SpaceX CRS-9	2257.0	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)	7
14/04/2015	20:10:00	F9 v1.1 B1015	CCAFS LC-40	SpaceX CRS-6	1898.0	LEO (ISS)	NASA (CRS)	Success	Failure (drone ship)	3
12/05/2018	18:16:00	F9 B5B1050	CCAFS SLC-40	SpaceX CRS-16	2500.0	LEO (ISS)	NASA (CRS)	Success	Failure	3
06/04/2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0.0	LEO	SpaceX	Success	Failure (parachute)	2
18/04/2014	19:25:00	F9 v1.1	CCAFS LC-40	SpaceX CRS-3	2296.0	LEO (ISS)	NASA (CRS)	Success	Controlled (ocean)	2
08/06/2019	23:23:00	F9 B5 B1047.3	CCAFS SLC-40	AMOS-17, Starlink 1 v1.0	6500.0	GTO	Spacecom	Success	No attempt	1

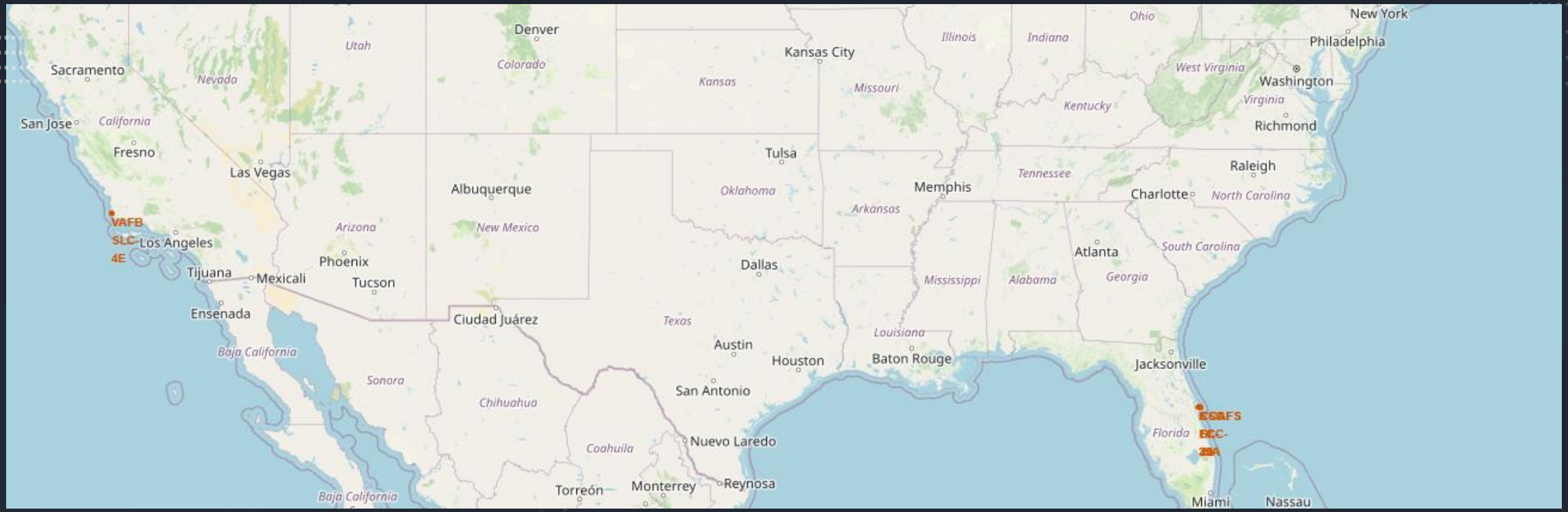
Rank Landing Outcomes Between 2010-06-04 and 2017-03-20.

This SQL command retrieve data from the table "SPACEXTBL" between date 04-06-2010 & 20-03-2017, group by the Landing_Outcome column, and ordered in descending order based on the count of occurrences of each landing outcome. It provide a breakdown of the landing outcomes and their respective counts during this period.

- SELECT *, COUNT(landing_Outcome) from SPACEXTBL : Retrieves all columns from the table "SPACEXTBL" and calculates the count of occurrences of the Landing_Outcome column for each row in the table.
- FROM SPACEXTBL : specifies the table("SPACEXTBL") from which the data is retrieved.
- WHERE DATE BETWEEN '04-06-2010' AND '20-03-2017' GROUP BY Landing_Outcome ORDER BY COUNT(Landing_Outcome) DESC : filters the data based on the specified date range, the GROUP BY clause groups the filtered data by Landing_Outcome, and the ORDER BY clause orders the resulting groups based on the count of each Landing_Outcome in descending order.

Section 3

Launch Sites Proximities Analysis



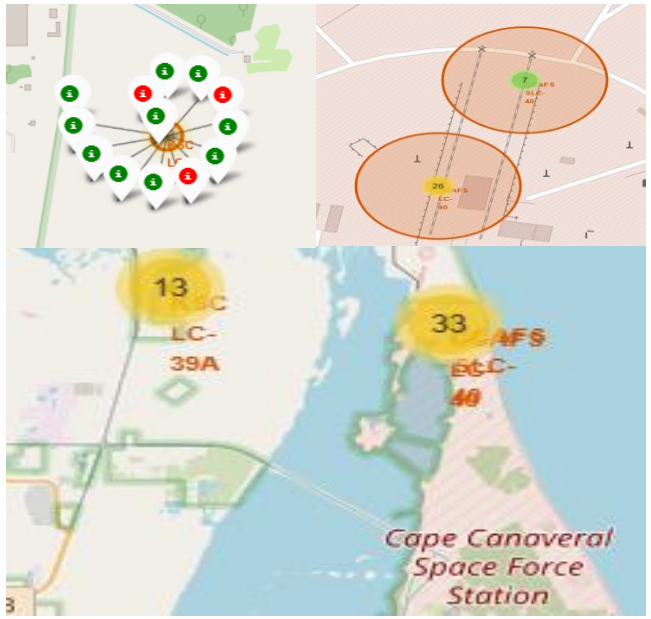
NASA Launch Sites Map

The following are the locations of the four launch sites shown on the map:

- CCAFS LC-40 (Cape Canaveral Air Force Station, Launch Complex 40): ***Florida***
- CCAFS SLC-40 (Cape Canaveral Space Launch Complex 40): ***Florida***
- KSC LC-39A (Kennedy Space Center, Launch Complex 39A): ***Florida***
- VAFB SLC-4E (Vandenberg Air Force Base, Space Launch Complex 4E): ***California***

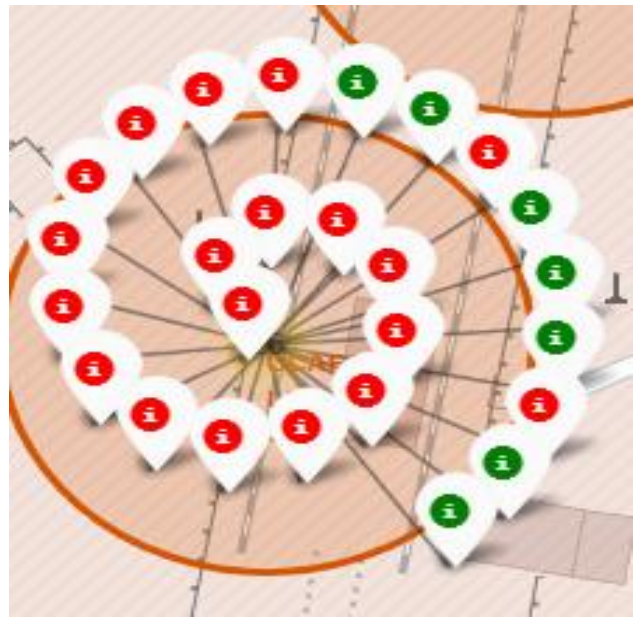
Total 10 launches at VAFB SL – 4E site California is.

Total13 launches at KSC LC-39A, and 33 launches at CCAFS LC-40/CCAFS SLC-40 sites in Florida.

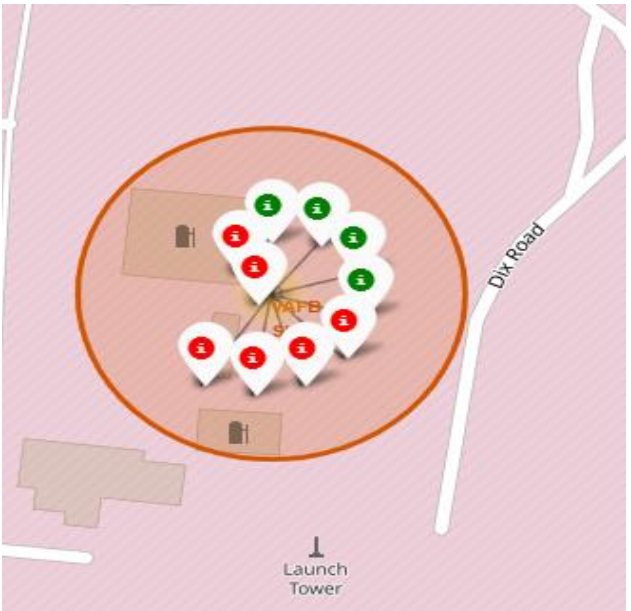


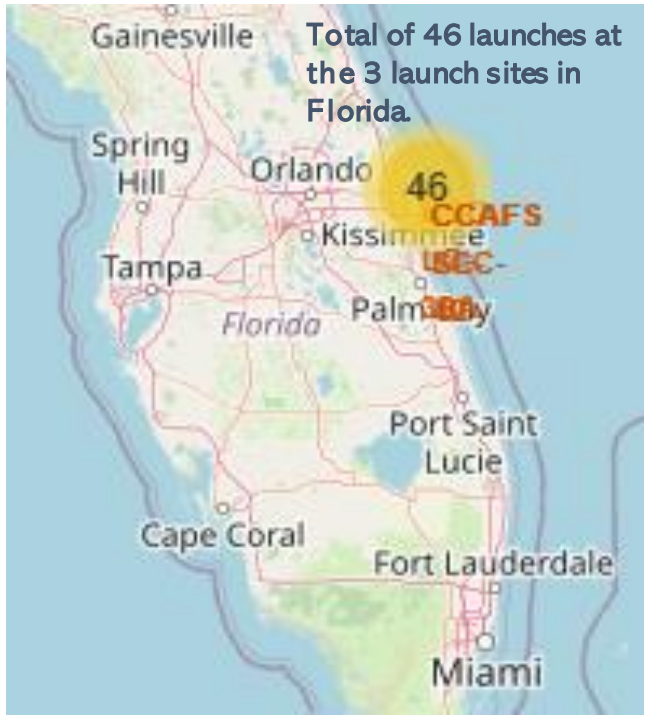
Success/Failure launches at CCAFS LC – 40:
Success – 3 success Failures – 4 Failures

Success/Failure launches at CCAFS SLC – 40:
Success – 7 success Failures – 19 Failures



Success/Failure launches at VAFB SL – 4E:
Success – 4 success Failures – 6 Failures

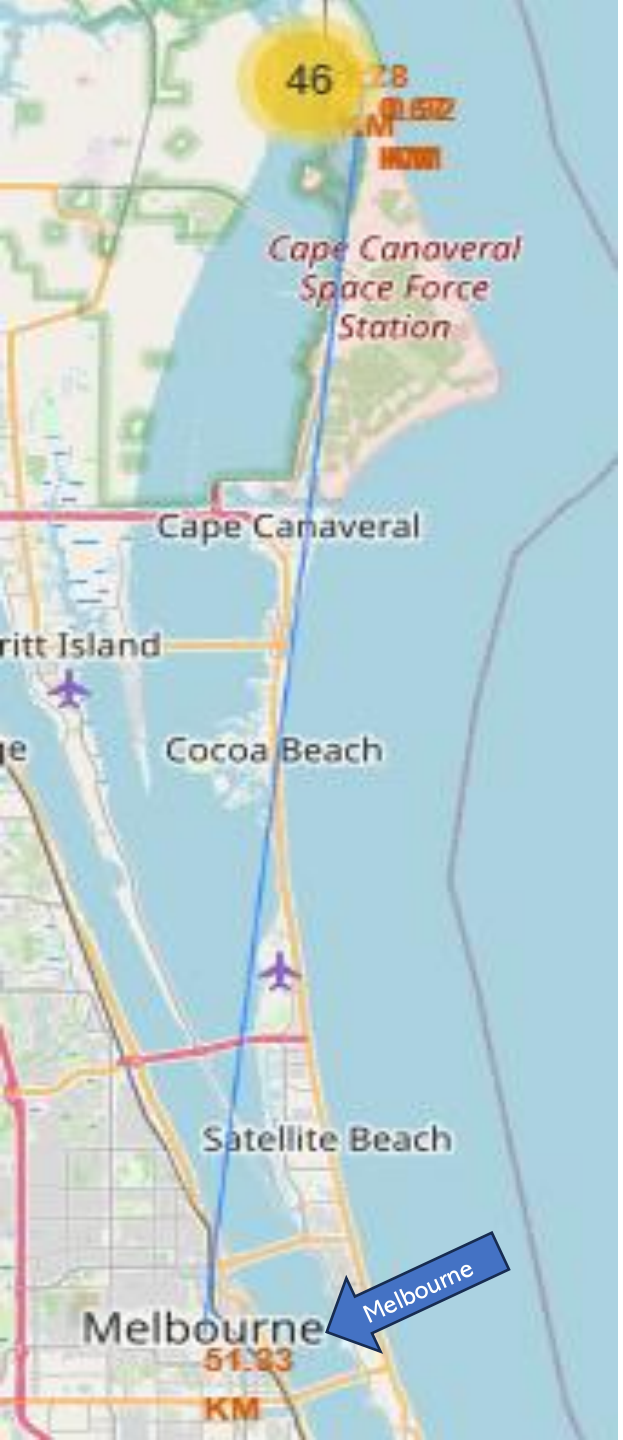


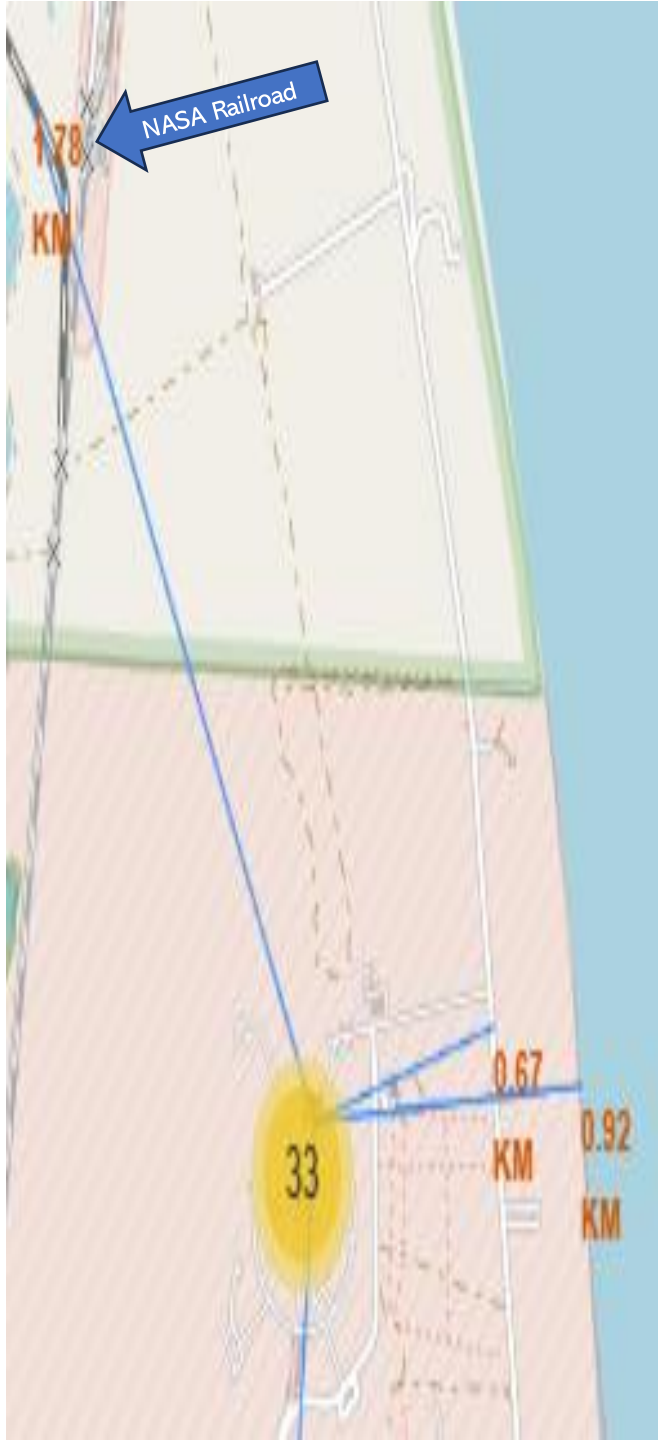


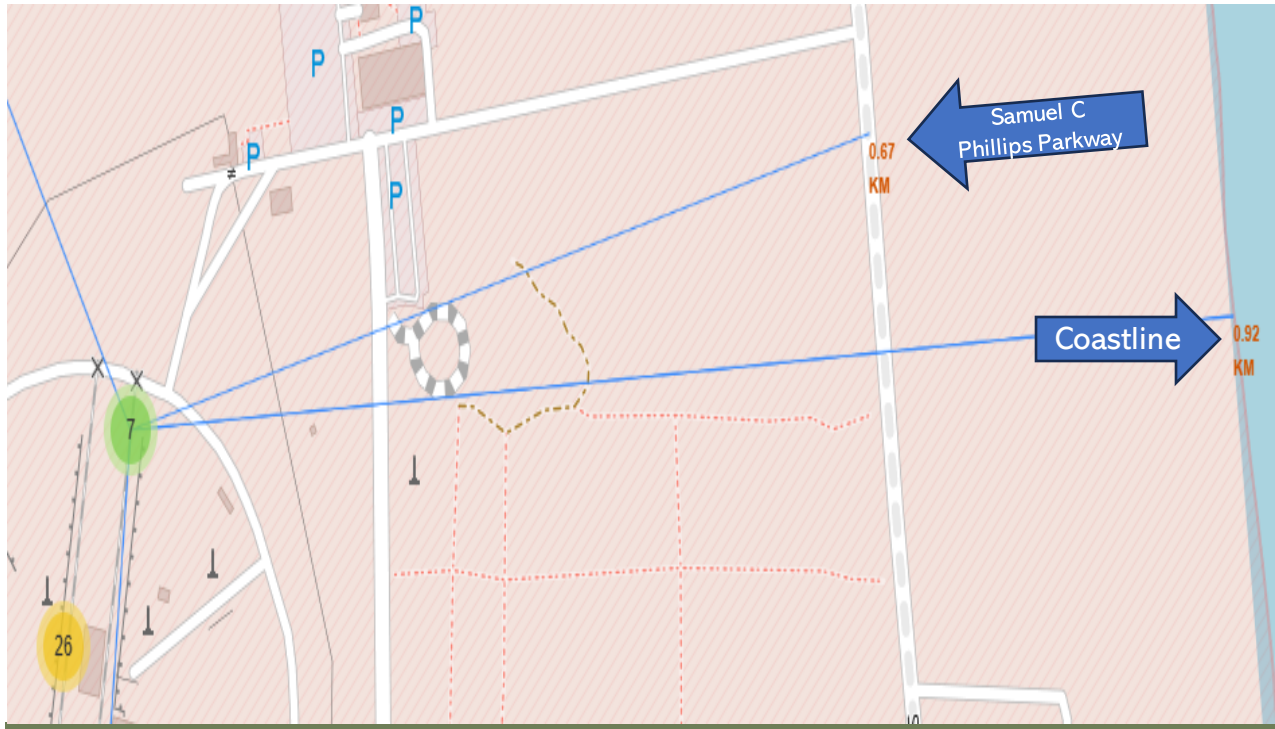
NASA Launch Sites Map with Marker Labels

The following are the success/failure launches carried out at both California and Florida launch sites shown by marker labels on Folium map:

- CCAFS SLC-40 (Cape Canaveral Air Force Station, Launch Complex 40) (***Florida) :***
  - ***Success – 3***
  - ***Failures – 4***
- CCAFS LC-40 (Cape Canaveral Space Launch Complex 40) (***Florida) :***
  - ***Success – 7***
  - ***Failure – 19***
- KSC LC-39A (Kennedy Space Center, Launch Complex 39A) (***Florida) :***
  - ***Success – 10***
  - ***Failures – 3***
- VAFB SLC-4E (Vandenberg Air Force Base, Space Launch Complex 4E) (***California) :***
  - ***Success – 4***
  - ***Failures – 6***







NASA Launch Sites Proximities to Landmarks Map

Close proximities means within 500 feet(or 0.1524km) on a straight line commencing at the property lines nearest to each other. Therefore the proximities of the following landmarks can be established once the Polylines is drawn on the Folium map:

- Are launch sites in close proximity to railways? No, closest railways is NASA railroad, which is about 1.35km away from CCAFS SLC-40 launch site.
- Are launch sites in close proximity to highways? No, closest highways is Samuel C Phillips Parkway, which is about 0.67km from CCAFS SLC-40 launch site.
- Are launch sites in close proximity to coastline? No, closest coastline is Florida coast, which is about 0.92km from CCAFS SLC-40 launch site.
- Do launch sites keep certain distance away from cities? Yes, farthest city is Melbourne, which is about 51.33km away from CCAFS SLC-40 launch site.

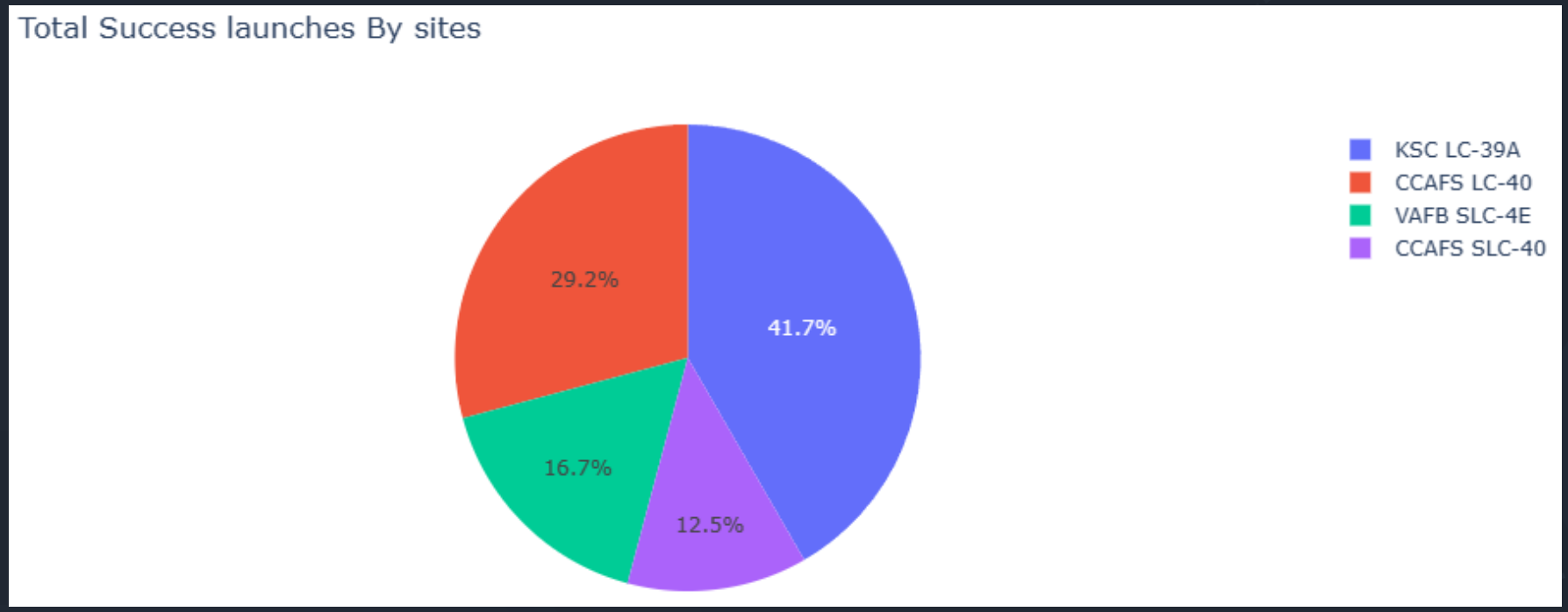
Section 4

Build a Dashboard with Plotly Dash

Pie Chart – Distribution of successful Launches by Site.

The following are total of successful launches by Sites shown on a pie chart in descending order:

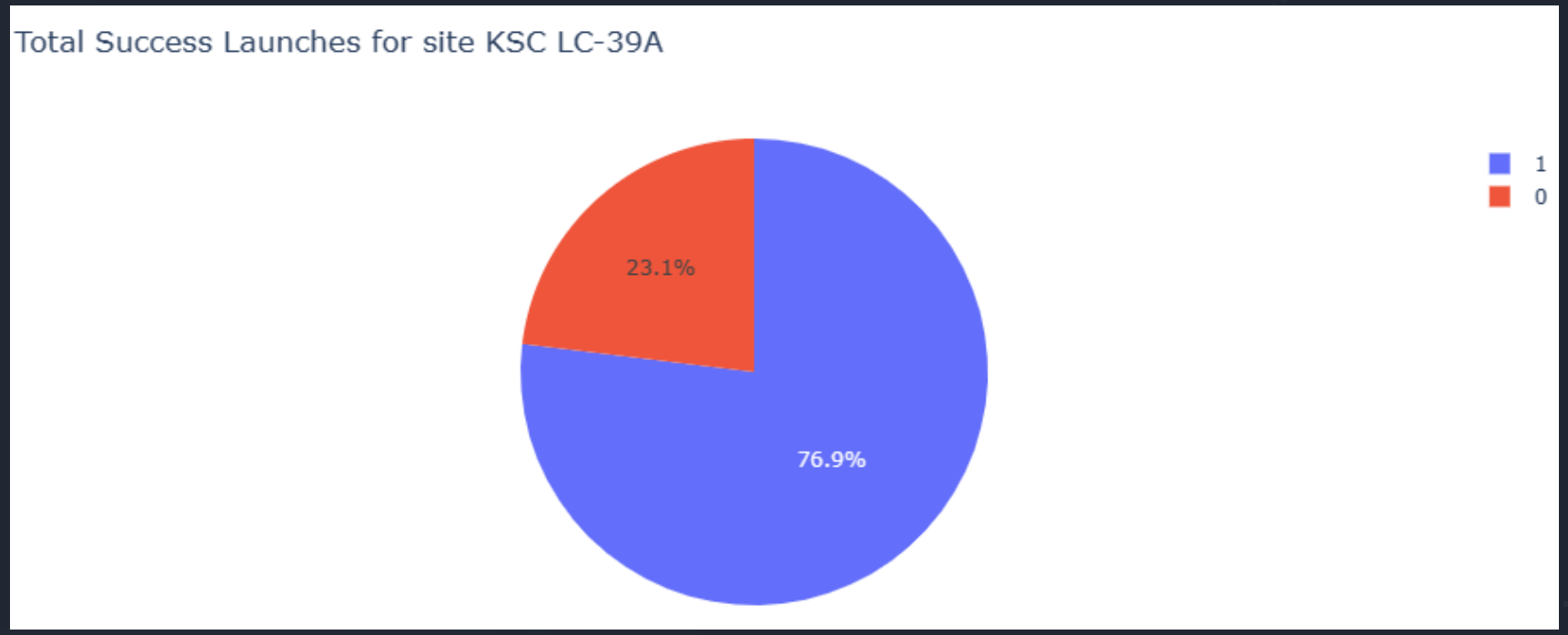
- ***41.7% of the overall successful launches c at KSC LC-39A.***
- ***29.2% of the overall successful launches are launched at CCAFS LC-40.***
- ***16.7% of the overall successful launches are launched at VAFB SLC-4E.***
- ***12.5% of the overall successful launches is are launched at CCAFS SLC-40.***

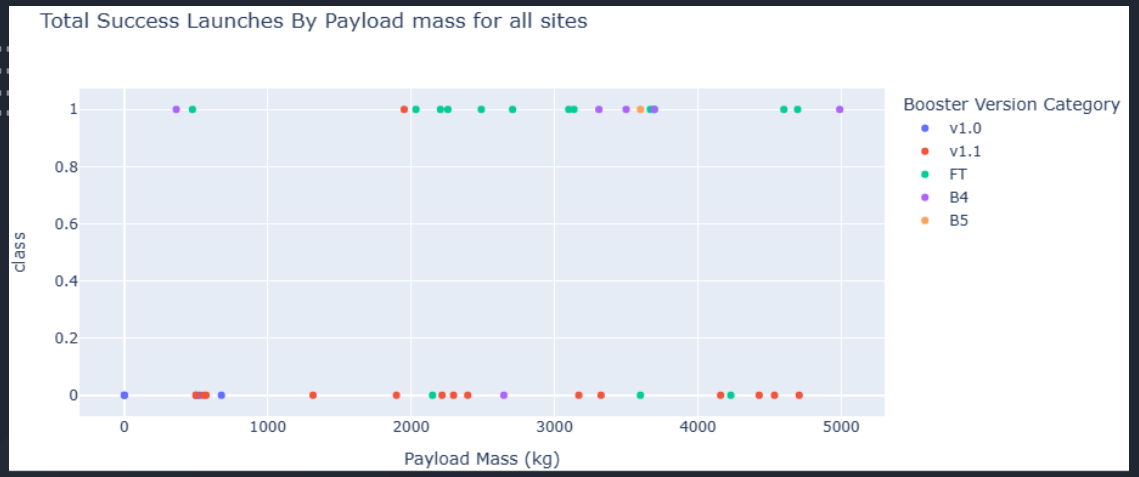


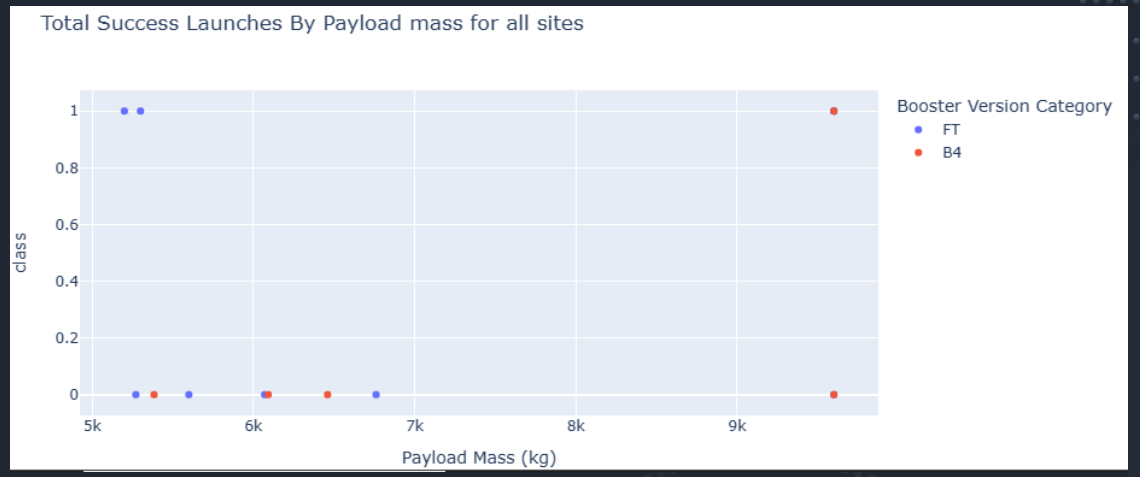
Pie Chart – Distribution of success/failure Launches at KSC LC-39A.

The following are the ratio of success/failure launches at KSC LC-39A. The pie chart shows the success rate at KSC LC-39A to be more than 3 times that of failure rate.

- ***76.9% of all launches at KSC LC-39A are successful launches.***
- ***23.1% of all launches at CCAFS LC-40 are failure launches.***





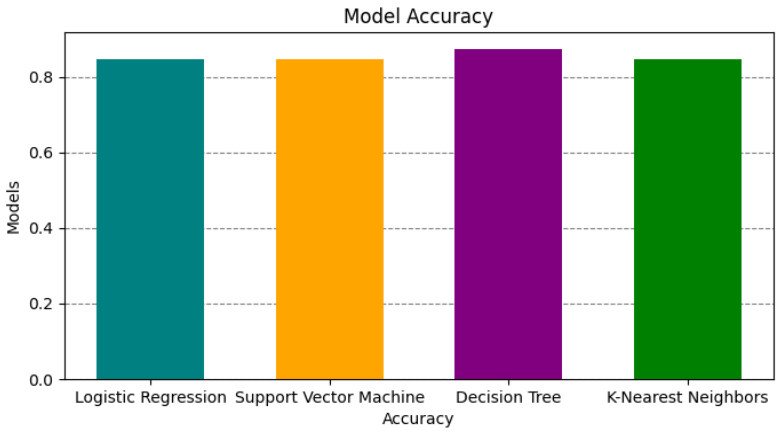


Scatter plot of Payload Vs. Launch Outcome

- The scatter plot of Payload Vs. Launch Outcome shows:
  - Shows low weighted payload has more success with Booster Version FT and less success with Booster Version v.1.1.
  - There are no success rate on heavy weighted payload other than Booster Version FT and B4, which only have 2 and 1 success respectively.

Section 5

Predictive Analysis (classification)



Support Vector Machine (SVM):
Best parameters: {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
Accuracy: 0.8482142857142856

```
print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
print("Support Vector Machine Accuracy :", svm_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
Support Vector Machine Accuracy : 0.8482142857142856
```

Decision Tree Classification:
Best parameters: {'criterion': 'entropy', 'max_depth': 2, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 2, 'splitter': 'best'}
Accuracy: 0.8749999999999998

```
print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
print("Decision Tree Classification Accuracy :", tree_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 2, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 2, 'splitter': 'best'}
Decision Tree Classification Accuracy : 0.8749999999999998
```

Logistic Regression:
Best parameters: {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
Accuracy: 0.8464285714285713

```
# Print the best parameters and best score
print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
print("Logistic Regression Accuracy :", logreg_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
Logistic Regression Accuracy : 0.8464285714285713
```

K Nearest Neighbors (KNN):
Best parameters: {'n_neighbors': 10, 'p': 1, 'weights': 'uniform'}
Accuracy: 0.8482142857142858

```
print("tuned hpyerparameters :(best parameters) ",knn_cv.best_params_)
print("K Nearest Neighbors Accuracy :", knn_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'n_neighbors': 10, 'p': 1, 'weights': 'uniform'}
K Nearest Neighbors Accuracy : 0.8482142857142858
```

Classification Accuracy

After performing the GridSearchCV on each classification model, we obtained the best parameters and corresponding accuracy scores for each model are:

- Logistic Regression: 0.8464285714285713
- Support Vector Machine (SVM): 0.8482142857142856
- Decision Tree Classification: 0.8749999999999998
- K Nearest Neighbors (KNN): 0.8482142857142858

From these results, we can see that the Decision Tree Classification model achieved the highest accuracy of 0.875, followed closely by SVM and KNN with accuracies of 0.848 each. Logistic Regression had a slightly lower accuracy of 0.846.

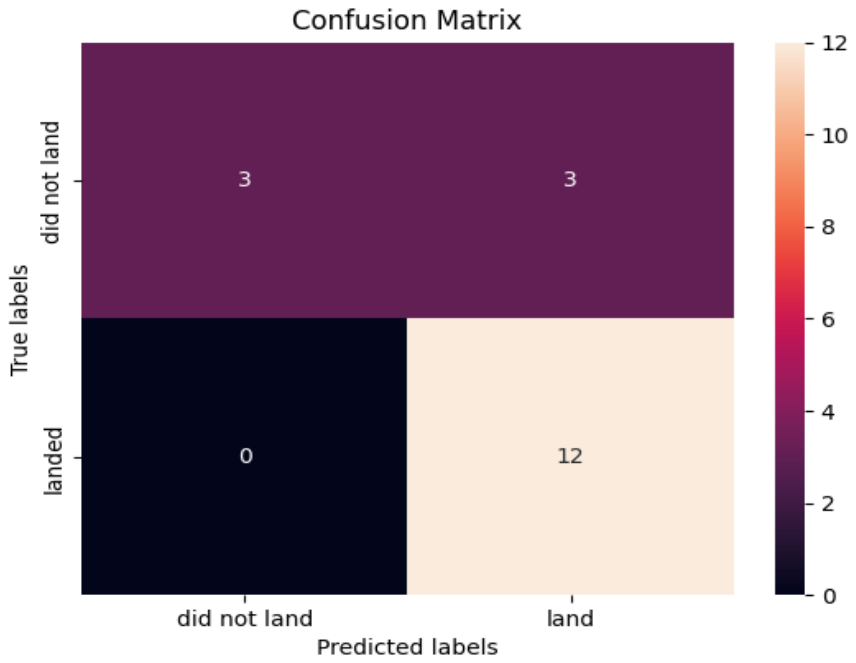
Therefore, based on the accuracy scores, the Decision Tree Classification model performed the best among the four models.

```
# Create a dictionary to store the model names and accuracy scores
model_scores = {
    'Logistic Regression': logreg_cv.best_score_,
    'Support Vector Machine': svm_cv.best_score_,
    'Decision Tree Classification': tree_cv.best_score_,
    'K Nearest Neighbors': knn_cv.best_score_
}

# Find the model with the highest accuracy
best_model = max(model_scores, key=model_scores.get)
best_accuracy = model_scores[best_model]

# Print the best performing model and its accuracy
print("Best Performing Model:", best_model)
print("Accuracy:", best_accuracy)
```

```
Best Performing Model: Decision Tree Classification
Accuracy: 0.8749999999999998
```

Confusion Matrix

- The confusion matrix reveals the following insights:
  - ➢ *True Positives (TP): This indicates that the model correctly predicted 12 instances as positive (class 1) when they were actually positive.*
  - ➢ *True Negatives (TN): The model correctly predicted 3 instances as negative (class 0) when they were actually negative.*
  - ➢ *False Positives (FP): The model incorrectly predicted 3 instances as positive when they were actually negative.*
  - ➢ *False Negatives (FN): The model did not predict any negative instances as positive incorrectly.*
- The decision tree achieved an accuracy score of approximately 88.93% on the training data, which suggests that it is able to accurately classify the majority of the instances in the dataset.
- The decision tree model has a relatively low number of false negatives (FN = 0), which means it rarely misclassifies negative instances as positive.
- The decision tree model is considered the best model based on the accuracy score obtained during the grid search




```
# Create a dictionary to store the model names and accuracy scores
model_scores = {
    'Logistic Regression': logreg_cv.best_score_,
    'Support Vector Machine': svm_cv.best_score_,
    'Decision Tree Classification': tree_cv.best_score_,
    'K Nearest Neighbors': knn_cv.best_score_
}

# Find the model with the highest accuracy
best_model = max(model_scores, key=model_scores.get)
best_accuracy = model_scores[best_model]

# Print the best performing model and its accuracy
print("Best Performing Model:", best_model)
print("Accuracy:", best_accuracy)
```

```
Best Performing Model: Decision Tree Classification
Accuracy: 0.8892857142857145
```

		Actual	
		NO	YES
Predicted	NO	True Negative (TN)	False Negative (FN)
	YES	False Positive (FP)	True Positive (TP)

Conclusions

- Launch Site Success Rate – If flight numbers are high (above 80 or 50) or if the payload mass falls within the specified ranges, then CCAFS SLC 40 or KSC LC 39A could be considered as having the best success rates
- Payload and Launch Site relationship – The success rates of launch sites can be influenced by the payload mass.
- Best Successful Orbits – The best successful orbits based on the analysis are ES-L1 (Earth-Sun L1), GEO (Geostationary Orbit), HEO (Highly Elliptical Orbit), and SSO (Sun-Synchronous Orbit).
- Flight Number and Orbit Relationship – The success rate is related to flight number in LEO(above 10), MEO(above 60), and SSO(above 40) and not significant in other orbits.
- Success Rate trends – There is a positive trends in success rate launches from 2013 to 2020.
- Model Performance – The best performing model for predicting the outcomes of SpaceX launches is the Decision Tree Classification model with an accuracy of approximately 0.8893.
- Launch sites location – Launch sites are not in proximities of landmarks, and are kept a certain distance away from cities.


Thank You!