IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Bakhshial Hajano
10-02-2024

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection: We collected launch data from an API and using web scraping, encompassing successful and failed landings.
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis with Data Visualization
  - Interactive Visual Analytics with Folium
  - Model Building: Implemented various models (Logistic Regression, SVM).
- Summary of all results
  - Exploratory Data Analysis result
  - Interactive analytics in screenshots
  - The predictive model demonstrated high accuracy in classifying Falcon 9 first stage landings as successful or unsuccessful.
  - Precision and recall metrics provided insights into minimizing false positives and false negatives

Introduction

- Project background and context

  Space X advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because Space X can reuse the first stage. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against space X for a rocket launch. This goal of the project is to create a machine learning pipeline to predict if the first stage will land successfully.

- Problems you want to find answers

  - What factors determine if the rocket will land successfully?
  - The interaction amongst various features that determine the success rate of a successful landing.
  - What operating conditions needs to be in place to ensure a successful landing program.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Describe how data was collected
- Perform data wrangling
  - Describe how data was processed
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

- The data was collected using various methods
  - Data collection was done using get request to the SpaceX API.
  - Next, we decoded the response content as a Json using .json() function call and turn it into a pandas dataframe using .json_normalize().
  - We then cleaned the data, checked for missing values and fill in missing values where necessary.
  - In addition, we performed web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.
  - The objective was to extract the launch records as HTML table, parse the table and convert it to a pandas dataframe for future analysis.

Data Collection – SpaceX API

- We used the get request to the SpaceX API to collect data, clean the requested data and did some basic data wrangling and formatting.
- The link to the notebook is https://github.com/Bakhshial/IBM_Data_Science/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

1. Get request for rocket launch data using API

In [6]:
```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

In [7]:
```
response = requests.get(spacex_url)
```

2. Use json_normalize method to convert json result to dataframe

In [12]:
```
# Use json_normalize method to convert the json result into a dataframe

# decode response content as json
static_json_df = res.json()
```

In [13]:
```
# apply json_normalize
data = pd.json_normalize(static_json_df)
```

3. We then performed data cleaning and filling in the missing values

In [30]:
```
rows = data_falcon9['PayloadMass'].values.tolist()[0]

df_rows = pd.DataFrame(rows)
df_rows = df_rows.replace(np.nan, PayloadMass)

data_falcon9['PayloadMass'][0] = df_rows.values
data_falcon9
```

Data Collection - Scraping

- We applied web scrapping to webscrap Falcon 9 launch records with BeautifulSoup
- We parsed the table
- The link to the notebook is https://github.com/Bakhshial/IBM_Data_Science/blob/main/jupyter-labs-webscraping.ipynb

To keep the lab tasks consistent, you will be asked to scrape the data from a snapshot of the `List of Falcon 9 and Falcon Heavy launches` Wikipage updated on `9th June 2021`

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

Next, request the HTML page from the above URL and get a `response` object

TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

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

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

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.text, 'html.parser')
```

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

```
# Use soup.title attribute
soup.title
```

```
Out[7]: <title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

TASK 2: Extract all column/variable names from the HTML table header

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please check the external reference link towards the end of this lab

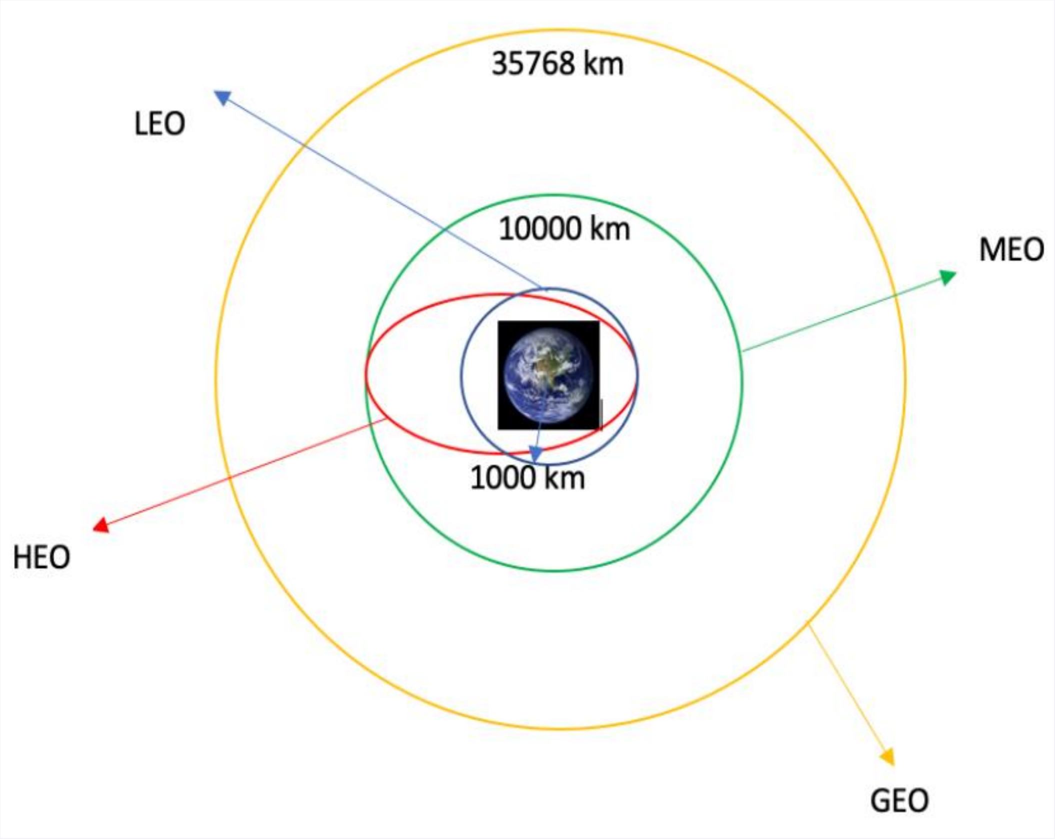
```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
 # Find all tables on the wiki page
html_tables = soup.find_all('table')

# Print the number of tables found
print(f"Number of tables found: {len(html_tables)}")
```

```
Number of tables found: 25
```

Starting from the third table is our target table contains the actual launch records.

```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

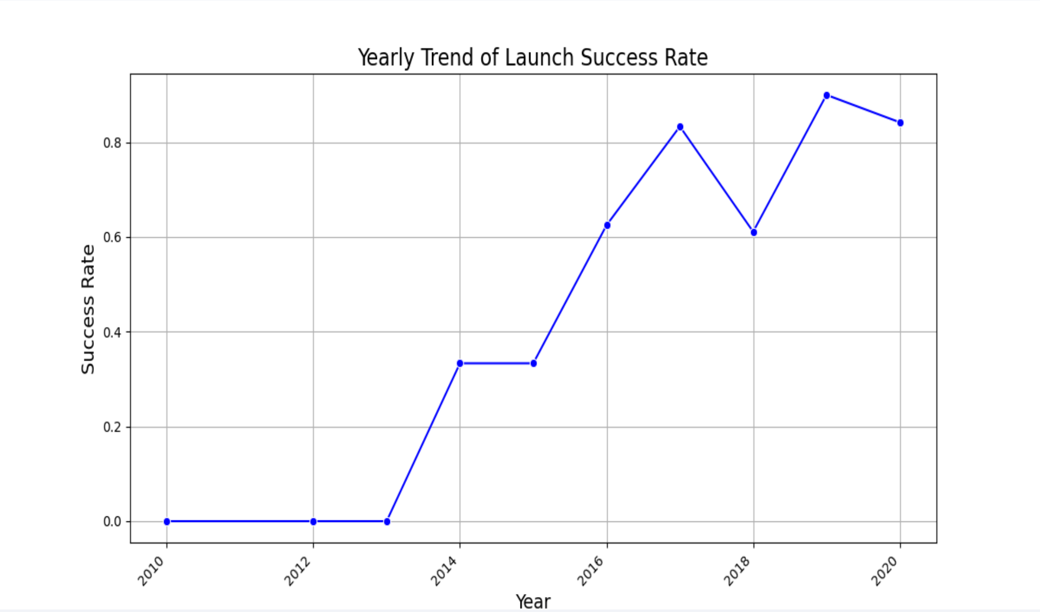
Data Wrangling

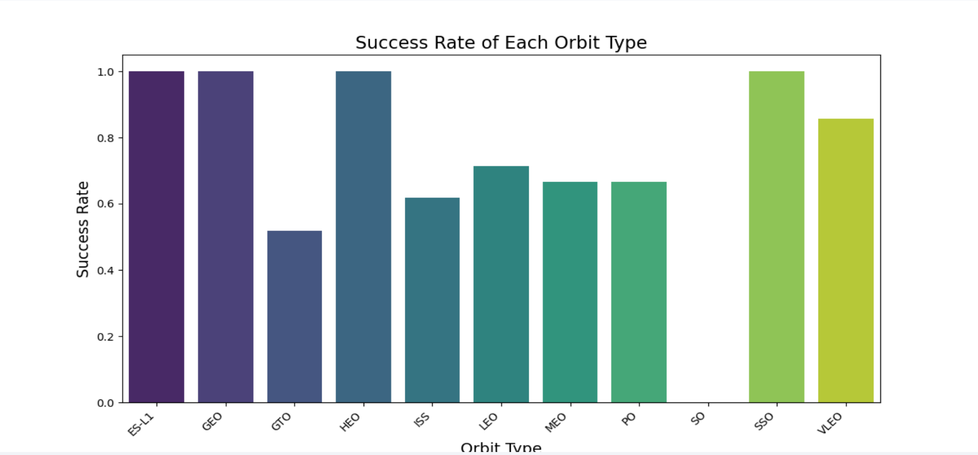


- We performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits
- We created landing outcome label from outcome column and exported the results to csv.
- The link to the notebook is https://github.com/Bakhshial/IBM_Data_Science/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

- We explored the data by visualizing the relationship between flight number and launch Site, payload and launch site, success rate of each orbit type, flight number and orbit type, the launch success yearly trend.





- The link to the notebook is https://github.com/Bakhshial/IBM_Data_Science/blob/main/jupyter-labs-eda-dataviz.ipynb

EDA with SQL

- We loaded the SpaceX dataset into a sqlite3 database without leaving the jupyter notebook.
- We applied EDA with SQL to get insight from the data. We wrote queries to find out for instance:
  - The names of unique launch sites in the space mission.
  - The total payload mass carried by boosters launched by NASA (CRS)
  - The average payload mass carried by booster version F9 v1.1
  - The total number of successful and failure mission outcomes
  - The failed landing outcomes in drone ship, their booster version and launch site names.
- The link to the notebook is https://github.com/Bakhshial/IBM_Data_Science/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- We marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- We assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.
- Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate.
- We calculated the distances between a launch site to its proximities. We answered some question for instance:
    - Are launch sites near railways, highways and coastlines.
    - Do launch sites keep certain distance away from cities.
- Notebook link: https://github.com/Bakhshial/IBM_Data_Science/blob/main/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly dash
- We plotted pie charts showing the total launches by a certain sites
- We plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.
- The link to the notebook is https://github.com/Bakhshial/IBM_Data_Science/blob/main/SpaceX_app_1.py

Predictive Analysis (Classification)

- We loaded the data using pandas, transformed the data, split our data into training and testing.
- We built different machine learning models and tune different hyperparameters using GridSearchCV.
- We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- We found the best performing classification model.
- The link to the notebook is https://github.com/Bakhshial/IBM_Data_Science/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Results

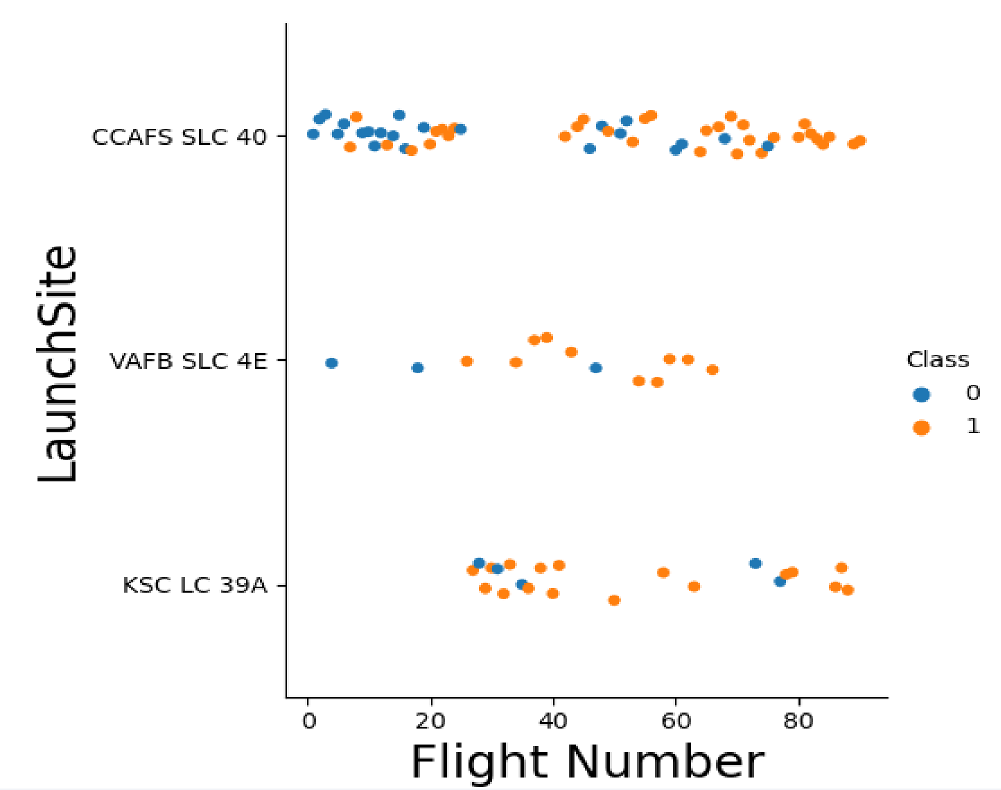
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

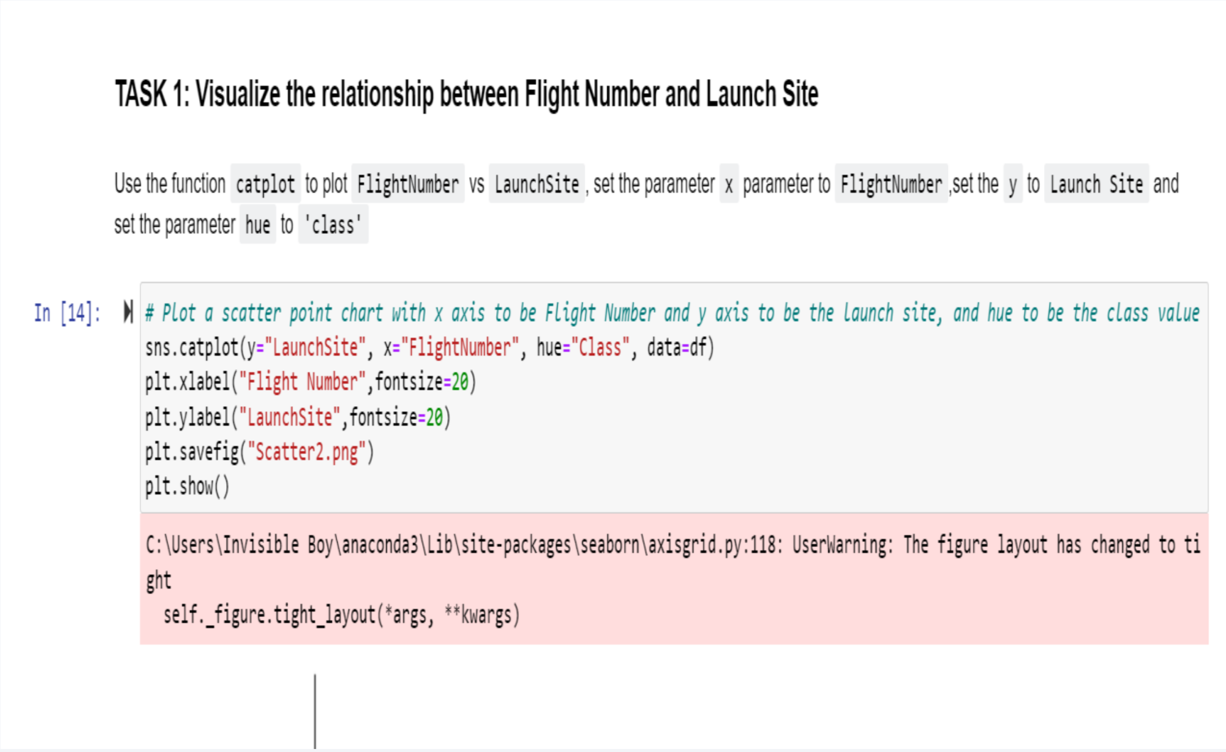
Insights drawn from EDA

Flight Number vs. Launch Site

- Show a scatter plot of Flight Number vs. Launch Site



- Show the screenshot of the scatter plot with explanations



TASK 1: Visualize the relationship between Flight Number and Launch Site

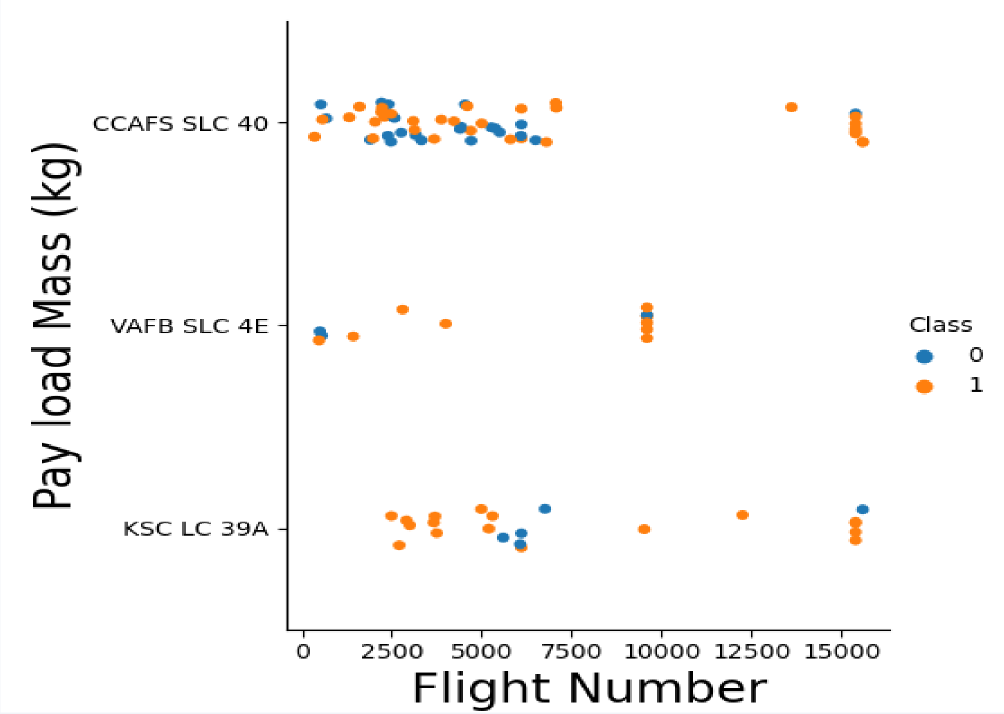
Use the function `catplot` to plot `FlightNumber` vs `LaunchSite`, set the parameter `x` parameter to `FlightNumber`,set the `y` to `Launch Site` and set the parameter `hue` to `'class'`

In [14]:
```
# 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 value
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("LaunchSite",fontsize=20)
plt.savefig("Scatter2.png")
plt.show()
```

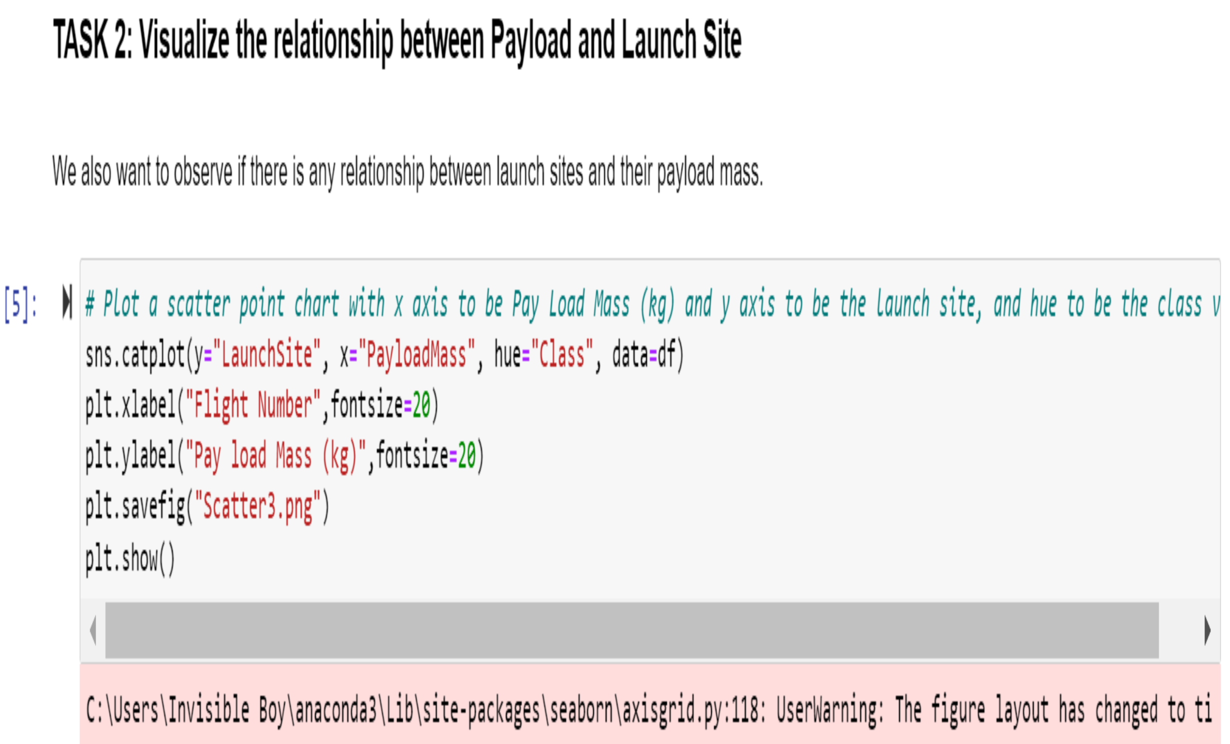
```
C:\Users\Invisible Boy\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed to ti
ght
  self._figure.tight_layout(*args, **kwargs)
```

Payload vs. Launch Site

- Show a scatter plot of Payload vs. Launch Site



Show the screenshot of the scatter plot with explanations



TASK 2: Visualize the relationship between Payload and Launch Site

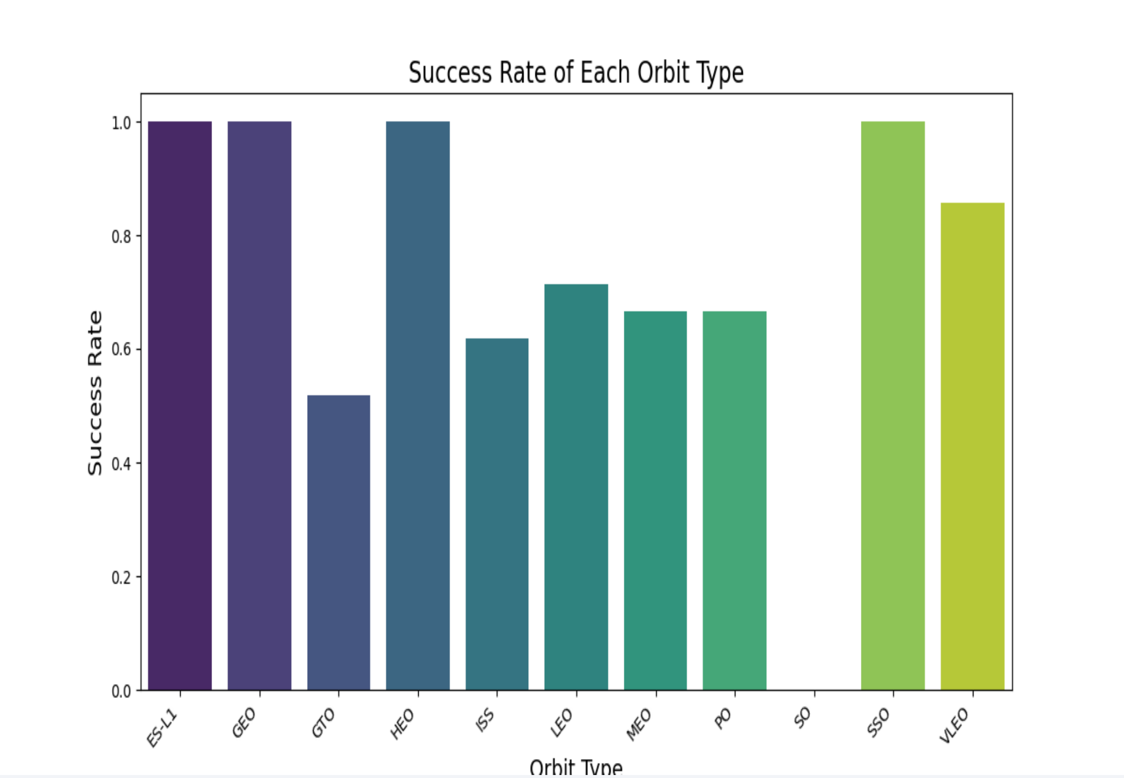
We also want to observe if there is any relationship between launch sites and their payload mass.

```
[5]: # 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 the class v
sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Pay load Mass (kg)",fontsize=20)
plt.savefig("Scatter3.png")
plt.show()
```

```
C:\Users\Invisible Boy\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed to ti
```

Success Rate vs. Orbit Type

- Show a bar chart for the success rate of each orbit type



- Show the screenshot of the scatter plot with explanations

TASK 3: Visualize the relationship between success rate of each orbit type

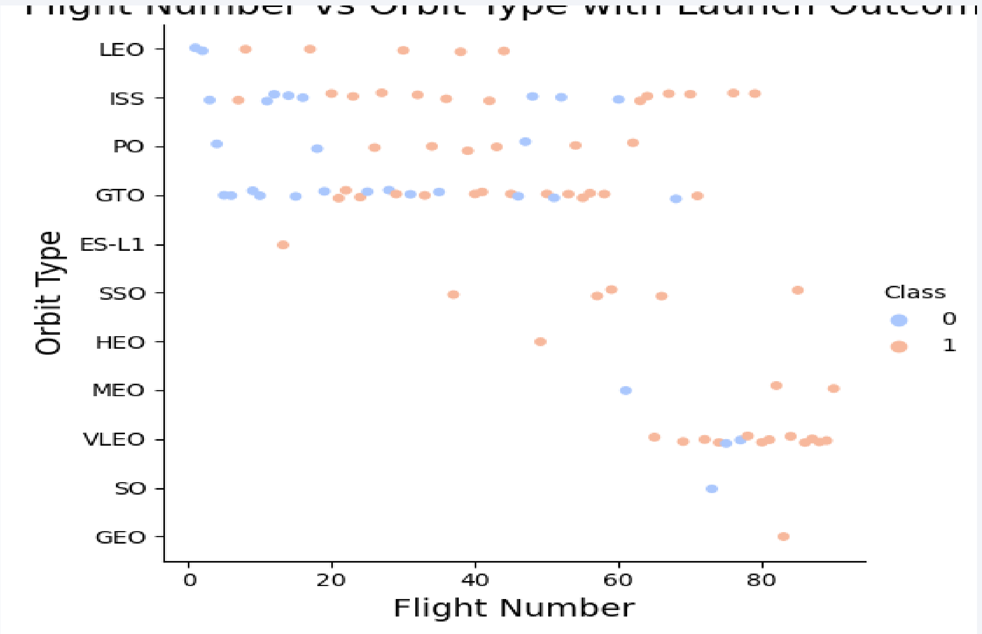
Next, we want to visually check if there are any relationship between success rate and orbit type.

Let's create a `bar chart` for the sucess rate of each orbit

```python
# HINT use groupby method on Orbit column and get the mean of Class column
# Assuming your DataFrame is named 'df'
orbit_success_rate = df.groupby('Orbit')['Class'].mean().reset_index()

# Plotting the bar chart
plt.figure(figsize=(12, 6))
sns.barplot(x='Orbit', y='Class', data=orbit_success_rate, palette='viridis')
plt.xlabel("Orbit Type", fontsize=14)
plt.ylabel("Success Rate", fontsize=14)
plt.title("Success Rate of Each Orbit Type", fontsize=16)
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better visibility
plt.savefig("barchart1.png")
plt.show()
```

Flight Number vs. Orbit Type

- Show a scatter point of Flight number vs. Orbit type



Show the screenshot of the scatter plot with explanations

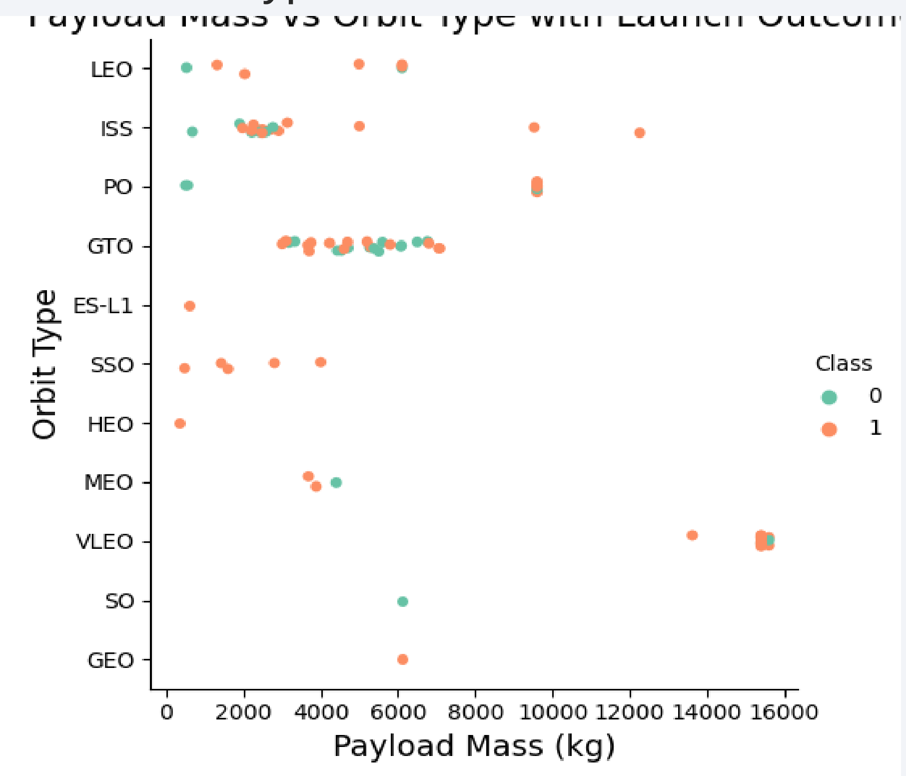
TASK 4: Visualize the relationship between FlightNumber and Orbit type

For each orbit, we want to see if there is any relationship between FlightNumber and Orbit type.

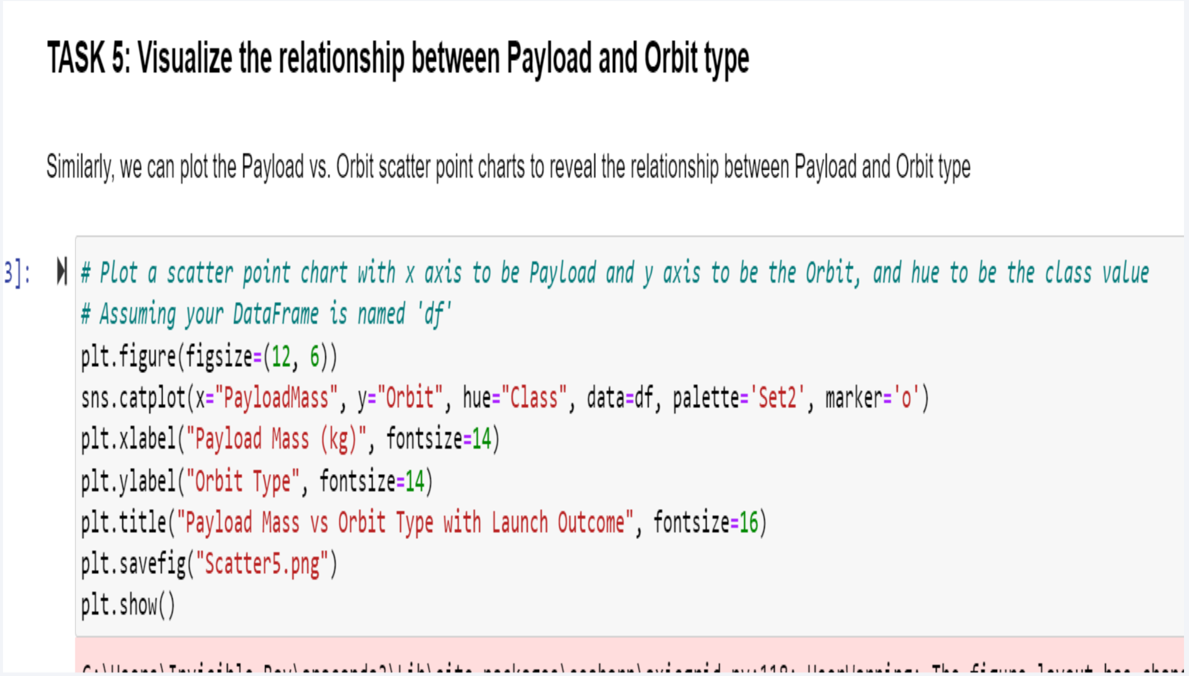
```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value
# Assuming your DataFrame is named 'df'
plt.figure(figsize=(12, 6))
sns.catplot(x="FlightNumber", y="Orbit", hue="Class", data=df, palette='coolwarm', marker='o')
plt.xlabel("Flight Number", fontsize=14)
plt.ylabel("Orbit Type", fontsize=14)
plt.title("Flight Number vs Orbit Type with Launch Outcome", fontsize=16)
plt.savefig("Scatter4.png")
plt.show()
```

Payload vs. Orbit Type

- Show a scatter point of payload vs. orbit type



- Show the screenshot of the scatter plot with explanations



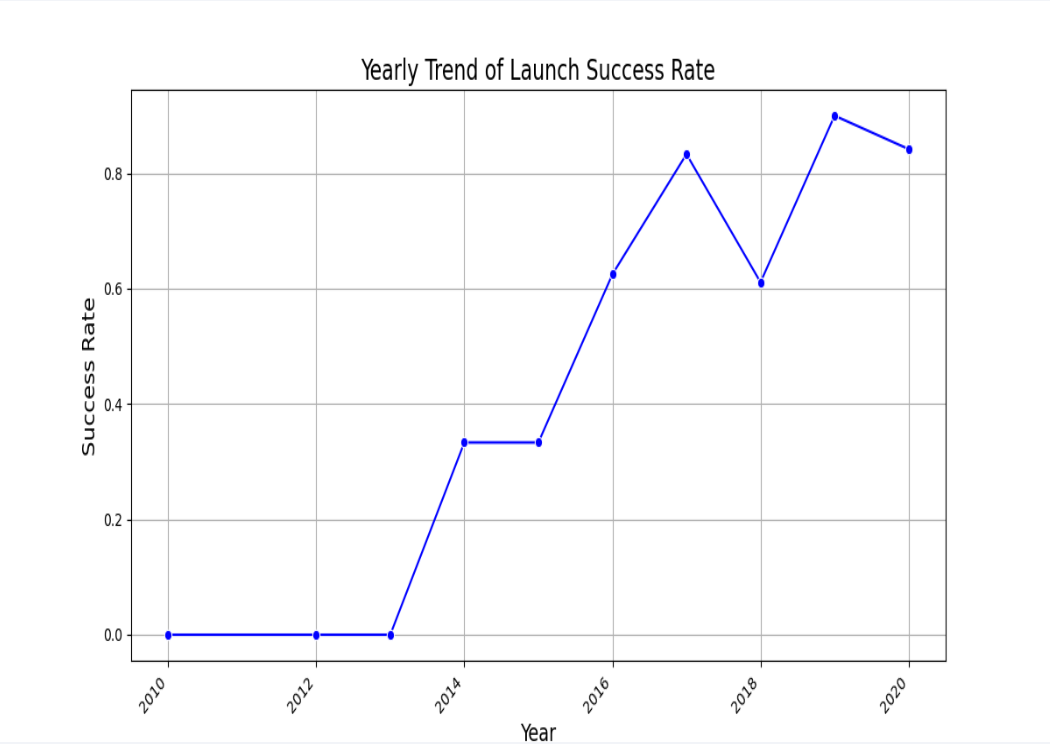
TASK 5: Visualize the relationship between Payload and Orbit type

Similarly, we can plot the Payload vs. Orbit scatter point charts to reveal the relationship between Payload and Orbit type

```python
# Plot a scatter point chart with x axis to be Payload and y axis to be the Orbit, and hue to be the class value
# Assuming your DataFrame is named 'df'
plt.figure(figsize=(12, 6))
sns.catplot(x="PayloadMass", y="Orbit", hue="Class", data=df, palette='Set2', marker='o')
plt.xlabel("Payload Mass (kg)", fontsize=14)
plt.ylabel("Orbit Type", fontsize=14)
plt.title("Payload Mass vs Orbit Type with Launch Outcome", fontsize=16)
plt.savefig("Scatter5.png")
plt.show()
```

Launch Success Yearly Trend

- Show a line chart of yearly average success rate



- Show the screenshot of the scatter plot with explanations

```
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate

# Convert 'Class' column to numeric for calculating mean


#Calculate the average success rate per year
yearly_success_rate = df.groupby('Year')['Class'].mean().reset_index()

# Plotting the line chart
plt.figure(figsize=(12, 6))
sns.lineplot(x="Year", y="Class", data=yearly_success_rate, marker='o', color='blue')
plt.xlabel("Year", fontsize=14)
plt.ylabel("Success Rate", fontsize=14)
plt.title("Yearly Trend of Launch Success Rate", fontsize=16)
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better visibility
plt.grid(True)  # Add grid lines for better readability
plt.savefig("lineplot.png")
plt.show()
```

All Launch Site Names

- We used the key word **DISTINCT** to show only unique launch sites from the SpaceX data.

Display the names of the unique launch sites in the space mission

In [10]:
```
task_1 = '''
        SELECT DISTINCT LaunchSite
        FROM SpaceX
'''
create_pandas_df(task_1, database=conn)
```

Out[10]:

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

Launch Site Names Begin with 'CCA'

Display 5 records where launch sites begin with the string 'CCA'

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

Out[11]:

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

- We used the query above to display 5 records where launch sites begin with \`CCA\`

Total Payload Mass

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

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

In [12]:
```
task_3 = '''
        SELECT SUM(PayloadMassKG) AS Total_PayloadMass
        FROM SpaceX
        WHERE Customer LIKE 'NASA (CRS)'
        '''
create_pandas_df(task_3, database=conn)
```

Out[12]:

	total_payloadmass
0	45596

Average Payload Mass by F9 v1.1

- We calculated the average payload mass carried by booster version F9 v1.1 as 2928.4

Display average payload mass carried by booster version F9 v1.1

In [13]:
```
task_4 = '''
        SELECT AVG(PayloadMassKG) AS Avg_PayloadMass
        FROM SpaceX
        WHERE BoosterVersion = 'F9 v1.1'
        '''
create_pandas_df(task_4, database=conn)
```

Out[13]:

	avg_payloadmass
0	2928.4

First Successful Ground Landing Date

- We observed that the dates of the first successful landing outcome on ground pad was 22nd December 2015

In [14]:

```
task_5 = '''
        SELECT MIN(Date) AS FirstSuccessfull_landing_date
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Success (ground pad)'
        '''
create_pandas_df(task_5, database=conn)
```

Out[14]:

	firstsuccessfull_landing_date
0	2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

In [15]:

```
task_6 = '''
        SELECT BoosterVersion
        FROM SpaceX
        WHERE LandingOutcome = 'Success (drone ship)'
            AND PayloadMassKG > 4000
            AND PayloadMassKG < 6000
        '''
create_pandas_df(task_6, database=conn)
```

Out[15]:

	boosterversion
0	F9 FT B1022
1	F9 FT B1026
2	F9 FT B1021.2
3	F9 FT B1031.2

- We used the **WHERE** clause to filter for boosters which have successfully landed on drone ship and applied the **AND** condition to determine successful landing with payload mass greater than 4000 but less than 6000

Total Number of Successful and Failure Mission Outcomes

List the total number of successful and failure mission outcomes

In [16]:
```
task_7a = '''
        SELECT COUNT(MissionOutcome) AS SuccessOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Success%'
        '''

task_7b = '''
        SELECT COUNT(MissionOutcome) AS FailureOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Failure%'
        '''
print('The total number of successful mission outcome is:')
display(create_pandas_df(task_7a, database=conn))
print()
print('The total number of failed mission outcome is:')
create_pandas_df(task_7b, database=conn)
```

```
The total number of successful mission outcome is:
```

	successoutcome
0	100

```
The total number of failed mission outcome is:
```

Out[16]:

	failureoutcome
0	1

- We used wildcard like '%' to filter for **WHERE** MissionOutcome was a success or a failure.

Boosters Carried Maximum Payload

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

Task 8

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

```
5]: %sql SELECT Booster_Version FROM SPACEXTBL WHERE "PAYLOAD_MASS__KG_" = (SELECT MAX("PAYLOAD_MASS__KG_") FROM SPACEXTBL);
```

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

ut[45]:

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

2015 Launch Records

- We used a combinations of the **WHERE** clause, **LIKE**, **AND**, and **BETWEEN** conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015

Task 9

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

32]:

```
%%sql
SELECT
  substr(Date, 6, 2) AS "Month",
  "Landing_Outcome",
  "Booster_Version",
  "Launch_Site"
FROM SPACEXTBL
WHERE "Landing_Outcome" like'Failure (drone ship)' and Date between "2015-01-01" and "2015-12-31";
```

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

Out[32]:

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

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

- Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

Task 10

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.

3]:

```
%%sql
SELECT
  "Landing_Outcome",
  COUNT(*) AS count_outcomes,
  DENSE_RANK() OVER (ORDER BY COUNT(*) DESC) AS rank
FROM SPACEXTBL
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY "Landing_Outcome"
ORDER BY rank;
```

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

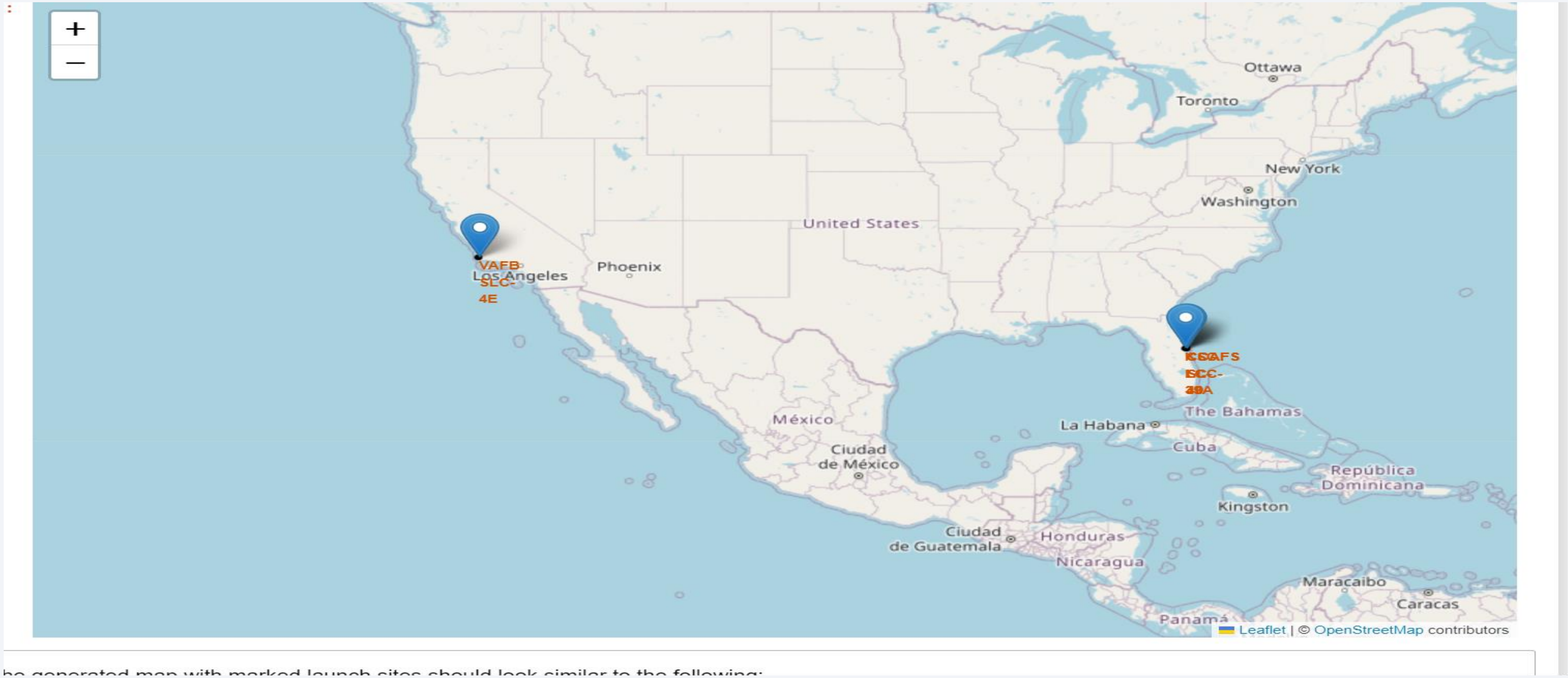
ut[33]:

Landing_Outcome	count_outcomes	rank
No attempt	10	1
Success (drone ship)	5	2
Failure (drone ship)	5	2
Success (ground pad)	3	3
Controlled (ocean)	3	3
Uncontrolled (ocean)	2	4
Failure (parachute)	2	4
Precluded (drone ship)	1	5

Section 3

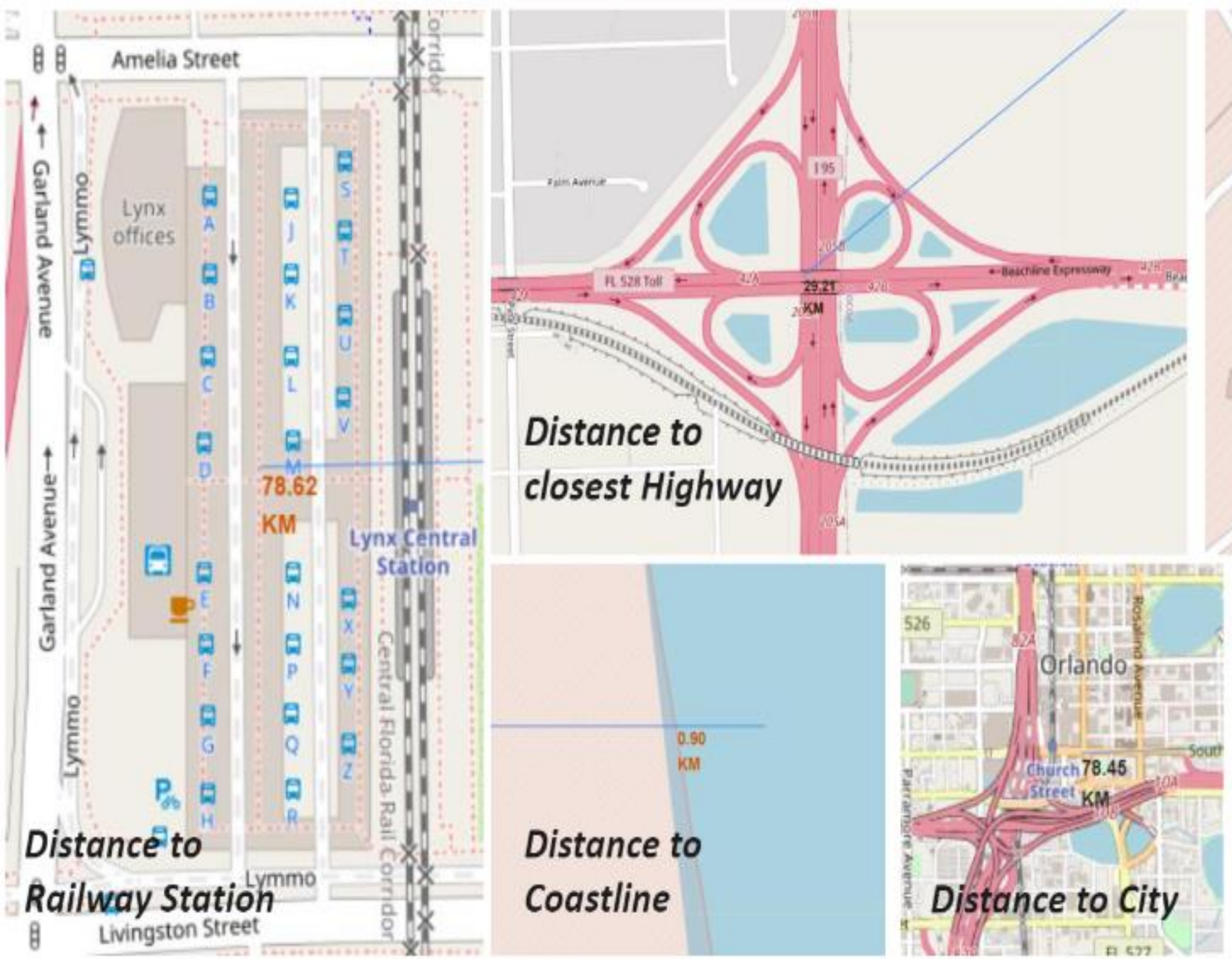
Launch Sites Proximities Analysis

All launch sites global map markers



Markers showing launch sites with color labels

Launch Site distance to landmarks

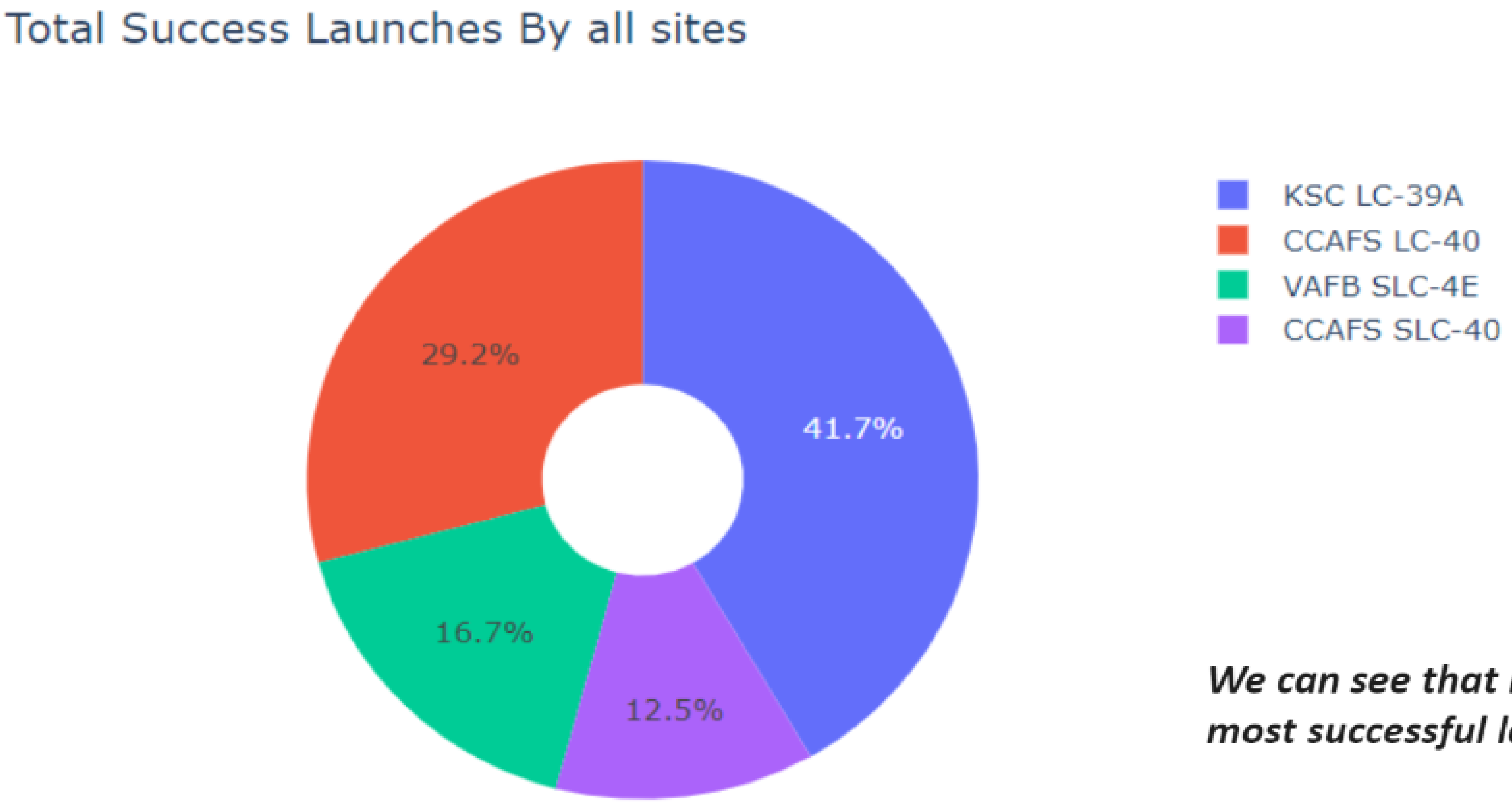


- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

Section 4

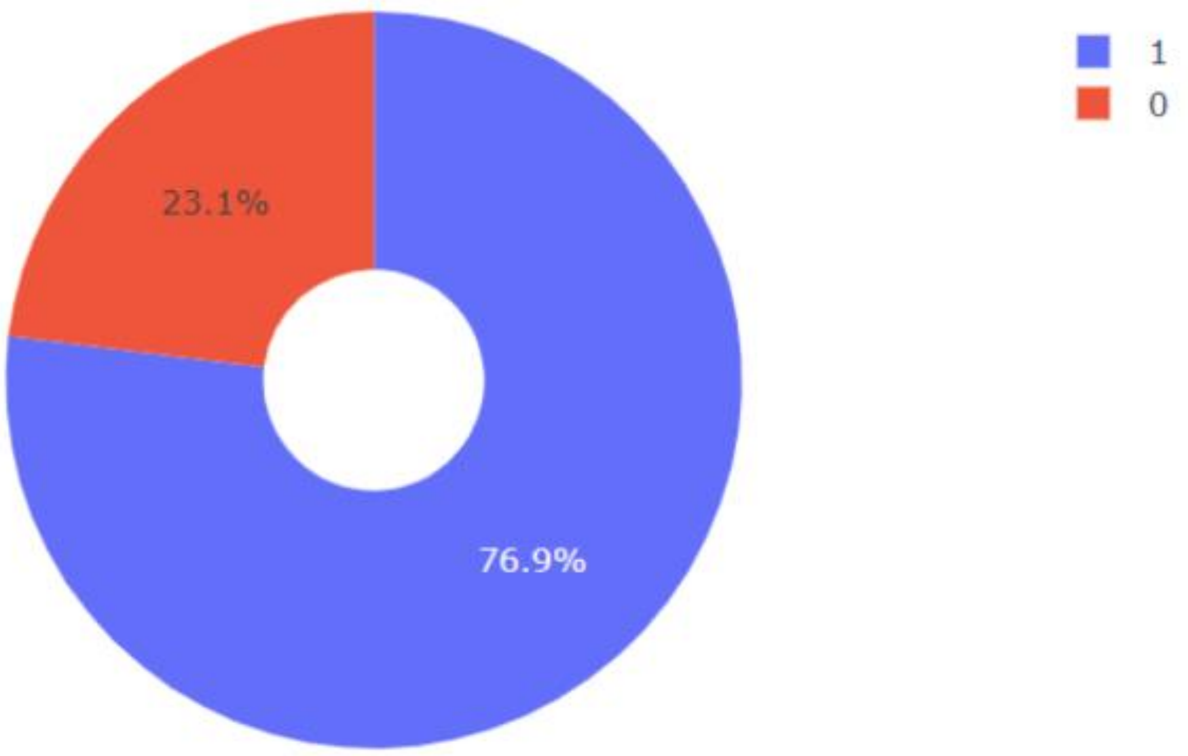
Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site



We can see that KSC LC-39A had the most successful launches from all the sites

Pie chart showing the Launch site with the highest launch success ratio



KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

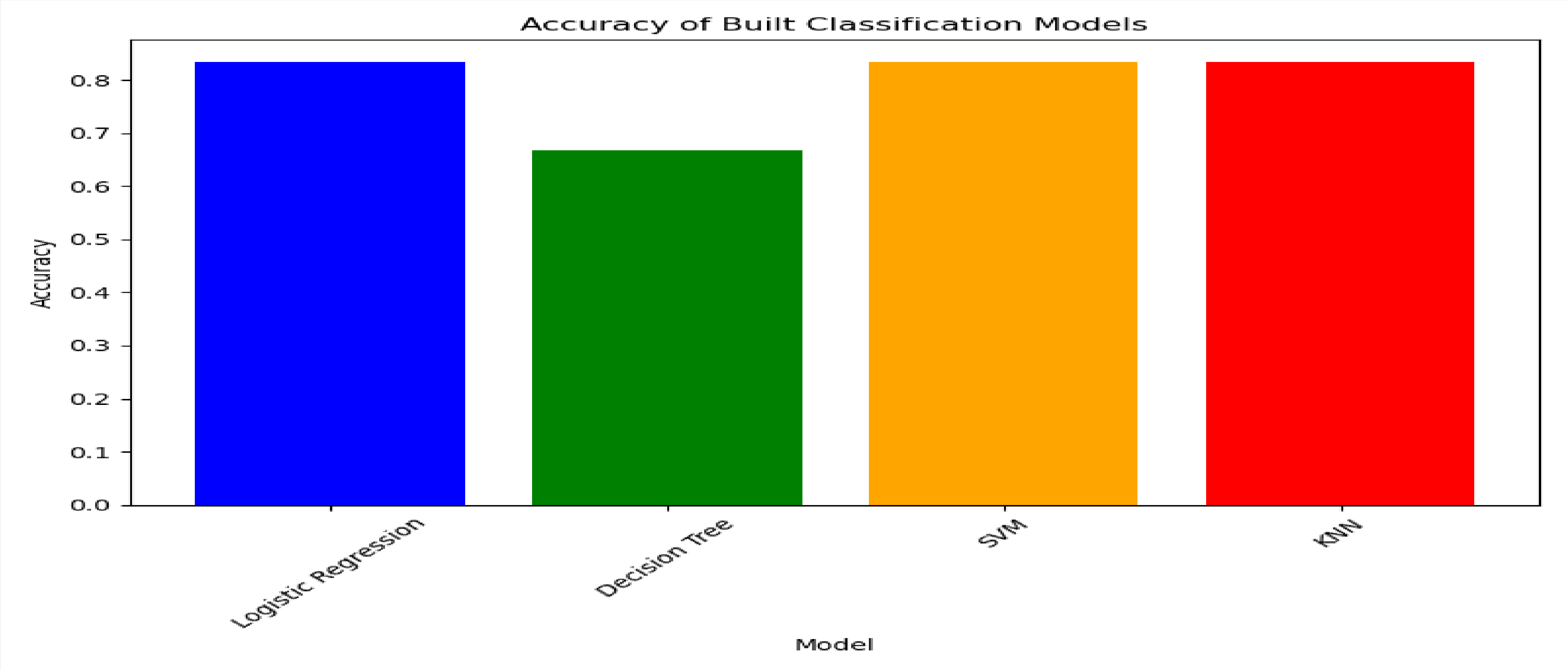
Scatter plot of Payload vs Launch Outcome for all sites, with different payload selected in the range slider

Section 5

Predictive Analysis (Classification)

Classification Accuracy

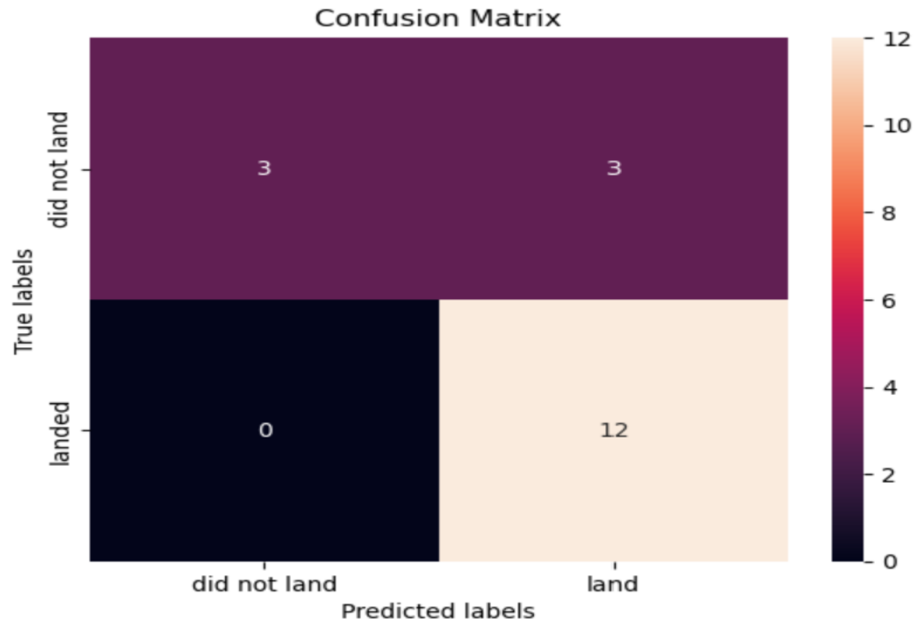
- The model with the highest accuracy on the test data is Logistic Regression. This model achieved an accuracy of 83.33%



Confusion Matrix

- Show the confusion matrix of the best performing model with an explanation

```
In [27]:  yhat=logreg_cv.predict(X_test)
          plot_confusion_matrix(Y_test,yhat)
```



Conclusions

We can conclude that:

- The larger the flight amount at a launch site, the greater the success rate at a launch site.
- Launch success rate started to increase in 2013 till 2020.
- Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
- KSC LC-39A had the most successful launches of any sites.
- The Decision tree classifier is the best machine learning algorithm for this task.

Thank you!