IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Dr Shalini Sankar
23rd September 2024

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion

Executive Summary

Summary of methodologies

- Data collection through API
- Data collection through Web Scraping
- Data wrangling
- Exploratory data analysis using SQL
- Exploratory data analysis with data visualisation techniques
- Interactive visual analytics using Folium
- Building a dashboard with folium
- Machine learning predictive analytics

Summary of all results

- Results from exploratory data analysis
- Results from interactive analysis as screenshots
- Results from machine learning predictive analytics

Introduction

- SpaceX 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 SpaceX can reuse the first stage.
- The aim of this capstone project was to predict if the Falcon 9 first stage will land successfully. By doing so, we can determine the cost of a launch.
- In this capstone, I took role of a data scientist working for a new rocket company, Space Y, that would like to compete with SpaceX founded by billionaire industrialist Allon Musk. My job was to determine the price of each launch. I gathered information about Space X and created dashboards. I also determined if SpaceX could reuse the first stage. Instead of using rocket science to determine if the first stage will land successfully, I trained a machine learning model and used public information to predict if SpaceX could reuse the first stage.

Points to address:

- *Identifying all factors that influence whether a landing will be successful or not.*
- *The relationships between different variables and how it affects the landing outcome.*
- *Best conditions required to increase the likelihood of a successful landing.*

Section 1

Methodology

Methodology

- Performed Data collection
  - Performed by using SpaceX API and Wikipedia Web Scraping.
- Performed Data wrangling
  - Data was processed using one-hot encoding for categorical features. Null values were replaced with mean and unwanted columns were cleaned.
- Performed exploratory data analysis (EDA) using visualization libraries and SQL
- Performed interactive visual analytics using Folium and Plotly Dash
- Performed predictive analysis using classification models
  - Building, tuning and evaluating classification models

Data Collection – SpaceX API

Now let's start requesting rocket launch data from SpaceX API with the following URL:

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

response = requests.get(spacex_url)
```

To make the requested JSON results more consistent, we will use the following static response object for this project:

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

We should see that the request was successfull with the 200 status response code

```
response.status_code
```

```
200
```

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

Using the dataframe `data` print the first 5 rows

```
# Decode the JSON data
data = response.json()

# Convert the JSON data to a DataFrame
data = pd.json_normalize(data)

# Display the first few rows of the DataFrame
data.head()
```

```
# Filter the DataFrame to exclude Falcon 1 launches and keep only Falcon 9 launches
data_falcon9 = df[df['BoosterVersion'] != 'Falcon 1']
# Calculate the mean value of PayloadMass column
payload_mass_mean = data_falcon9['PayloadMass'].mean()

# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'].replace(np.nan, payload_mass_mean, inplace=True)
# Export the cleaned DataFrame to a CSV file
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

- ❑ Data collection involved gathering and measuring information on specific variables within a system, which allowed important questions to be answered and outcomes to be evaluated. The dataset was collected using a REST API and web scraping from Wikipedia.
- ❑ For the REST API, a GET request was sent, and the response was decoded as JSON and converted into a pandas dataframe using json_normalize().
- ❑ The data was then cleaned, missing values were replaced with mean, and any gaps were filled as needed.
- ❑ Finally, the data was exported into a csv file.

GitHub URL for notebook: https://github.com/shalinimoorthy29/IBM-Data-Science-Capstone/blob/main/071_SpaceX_Data%20Collection%20API.ipynb

Data Collection - Scraping

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

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content

html_content = response.text  # Get the text content of the response

# Create a BeautifulSoup object using the HTML parser
soup = BeautifulSoup(html_content, 'html.parser')
```

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

```
column_names = []

# Apply find_all() function with `th` element on first_launch_table
columns = first_launch_table.find_all('th')

# Iterate each th element and apply the provided extract_column_from_header() to get a column name
for column in columns:
    name = extract_column_from_header(column)  # Extract the column name
    if name and len(name) > 0:  # Check if the column name is non-empty
        column_names.append(name)  # Append the column name to the list
```

```
launch_dict= dict.fromkeys(column_names)

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

# Let's initial the launch_dict with each value to be an empty list
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"):
```

```
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
```

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

- ❑ Performed a HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.
- ❑ Created a BeautifulSoup object using the HTML parser
- ❑ Extracted all column/variable names from the HTML table header
- ❑ Created an empty dictionary with keys from the extracted column names.
- ❑ This dictionary was then converted into a Pandas dataframe and data was exported to csv file.

GitHub URL for notebook: https://github.com/shalinimoorthy29/IBM-Data-Science-Capstone/blob/main/072_SpaceX_Webscraping.ipynb

Data Wrangling

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

```
df.isnull().sum()/len(df)*100
```

```
df.dtypes
```

```
# Calculate the number of launches for each launch site
launch_counts = df['LaunchSite'].value_counts()

# Display the number of launches for each site
print(launch_counts)
```

```
orbit_counts = df['Orbit'].value_counts()
print(orbit_counts)
```

```
landing_outcomes = df['Outcome'].value_counts()
print(landing_outcomes)
```

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

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

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise

df['Class'] = df['Outcome'].apply(lambda x: 0 if x in bad_outcomes else 1)
df['Class'].value_counts()
```

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

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

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

- ❑ Exploratory data analysis (EDA) was performed to identify patterns in the data and determine the label for training supervised models. Firstly, the Space X dataset was loaded.
- ❑ Calculated the number of launches on each site, the data frame number and occurrence of each orbit as well as the number and occurrence of mission outcome of the orbits.
- ❑ Outcomes were converted into training labels, where 1 indicates the booster successfully landed and 0 means it was unsuccessful.
- ❑ Data was then exported into a csv file.

GitHub URL for notebook: https://github.com/shalinimoorthy29/IBM-Data-Science-Capstone/blob/main/073_SpaceX_Data%20Wrangling.ipynb

EDA with Data Visualization

Scatter graphs were first used to find relationships between attributes such as:

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

Scatter plots help show how these attributes depend on each other. Once patterns were identified from the graphs, it became easier to see which factors had the most impact on the success of the landing outcomes.

Libraries used to perform analysis

 # pandas is a software library written for the Python programming language for data manipulation and analysis.
import pandas as pd

#NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays
import numpy as np

 # Matplotlib is a plotting library for python and pyplot gives us a MATLAB like plotting framework. We will use this in our plotter function to plot data.
import matplotlib.pyplot as plt

#Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics
import seaborn as sns

GitHub URL for notebook: https://github.com/shalinimoorthy29/IBM-Data-Science-Capstone/blob/main/075_SpaceX_EDA%20with%20Visualization%20Lab.ipynb

EDA with SQL

Using SQL, various queries were performed to gain a better understanding of the dataset, including:

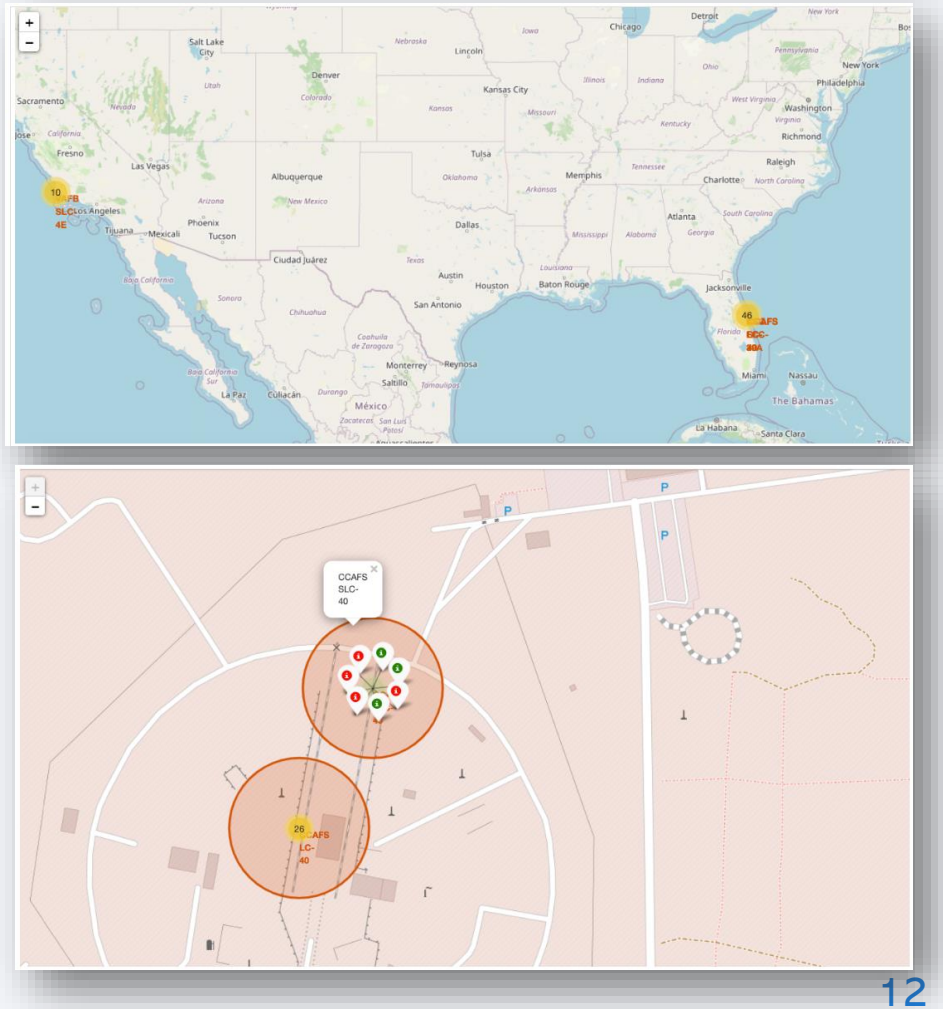
- ❑ Displaying the names of the launch sites.
- ❑ Displaying 5 records where launch sites begin with the string 'CCA'.
- ❑ Displaying the total payload mass carried by boosters launched by NASA (CRS).
- ❑ Displaying the average payload mass carried by booster version F9 v1.1.
- ❑ Listing the date of the first successful landing outcome on a ground pad.
- ❑ Listing the names of boosters that successfully landed on a drone ship and carried a payload mass greater than 4000 but less than 6000.
- ❑ Listing the total number of successful and failed mission outcomes.
- ❑ Listing the booster versions that carried the maximum payload mass.
- ❑ Listing the failed drone ship landing outcomes, their booster versions, and launch site names for the year 2015.
- ❑ Ranking the count of landing outcome successes between the dates 2010-06-04 and 2017-03-20 in descending order.

GitHub URL for notebook: https://github.com/shalinimoorthy29/IBM-Data-Science-Capstone/blob/main/074_SpaceX_EDA%20with%20SQL.ipynb

Build an Interactive Map with Folium

Interactive visual analytics were performed using Folium to explore the SpaceX launch sites:

- ❑ Folium circles and markers were created and added for each launch site on the map to allow zooming in and out of the marked areas.
- ❑ Successes and failed launches for each site were marked on the map.
- ❑ The distances between each launch site and nearby locations were calculated.
- ❑ A Mouse Position feature was added to display the coordinates (latitude and longitude) when hovering over the map.
- ❑ A marker was created to show the distance to the nearest city, railway, highway, etc.
- ❑ A line was drawn between the marker and the launch site to visualise the proximity.



GitHub URL for notebook: https://github.com/shalinimoorthy29/IBM-Data-Science-Capstone/blob/main/076_SpaceX_Launch%20Site%20Location%20-%20Folium.ipynb

Build a Dashboard with Plotly Dash

```
# Import required libraries
import pandas as pd
import dash
import dash_html_components as html
import dash_core_components as dcc
from dash.dependencies import Input, Output
import plotly.express as px

# Read the airline data into pandas dataframe
spacex_df = pd.read_csv("spacex_launch_dash.csv")
max_payload = spacex_df['Payload Mass (kg)'].max()
min_payload = spacex_df['Payload Mass (kg)'].min()

# Create a dash application
app = dash.Dash(__name__)

# Create an app layout
app.layout = html.Div(children=[html.H1('SpaceX Launch Records Dashboard',
                                        style={'textAlign': 'center', 'color': '#503D36',
                                               'font-size': 40}),
                                # TASK 1: Add a dropdown list to enable Launch Site selection
                                # The default select value is for ALL sites
                                # dcc.Dropdown(id='site-dropdown',...)
                                html.Br(),

                                # TASK 2: Add a pie chart to show the total successful launches count for all sites
                                # If a specific launch site was selected, show the Success vs. Failed counts for the site
                                html.Div(dcc.Graph(id='success-pie-chart')),
                                html.Br(),

                                html.P("Payload range (Kg):"),
                                # TASK 3: Add a slider to select payload range
                                #dcc.RangeSlider(id='payload-slider',...)

                                # TASK 4: Add a scatter chart to show the correlation between payload and launch success
                                html.Div(dcc.Graph(id='success-payload-scatter-chart')),
                                ])

# TASK 2:
# Add a callback function for `site-dropdown` as input, `success-pie-chart` as output

# TASK 4:
# Add a callback function for `site-dropdown` and `payload-slider` as inputs, `success-payload-scatter-chart` as output


# Run the app
if __name__ == '__main__':
    app.run_server()
```

For this part of the project, a Plotly Dash application was built for users to perform interactive visual analytics on SpaceX launch data in real-time.

- ❑ Added a Launch Site Drop-down Input Component
- ❑ Added a callback function to render success-pie-chart based on selected site dropdown
- ❑ Added a Range Slider to Select Payload
- ❑ Added a callback function to render the success-payload-scatter-chart scatter plot

GitHub URL for notebook: https://github.com/shalinimoorthy29/IBM-Data-Science-Capstone/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

- ❑ The data was loaded using numpy and pandas, then transformed and split into training and testing sets.
- ❑ Different machine learning models were built.
- ❑ Hyperparameters were tuned using GridSearchCV.
- ❑ Accuracy was used as the evaluation metric for the models.
- ❑ The model was further improved through feature engineering and algorithm tuning.
- ❑ The best-performing classification model was identified after testing and tuning (K-Nearest Neighbors , DecisionTree, LogisticRegression, Support Vector Machine).

Libraries used to perform analysis

```
import piplite
await piplite.install(['numpy'])
await piplite.install(['pandas'])
await piplite.install(['seaborn'])
```

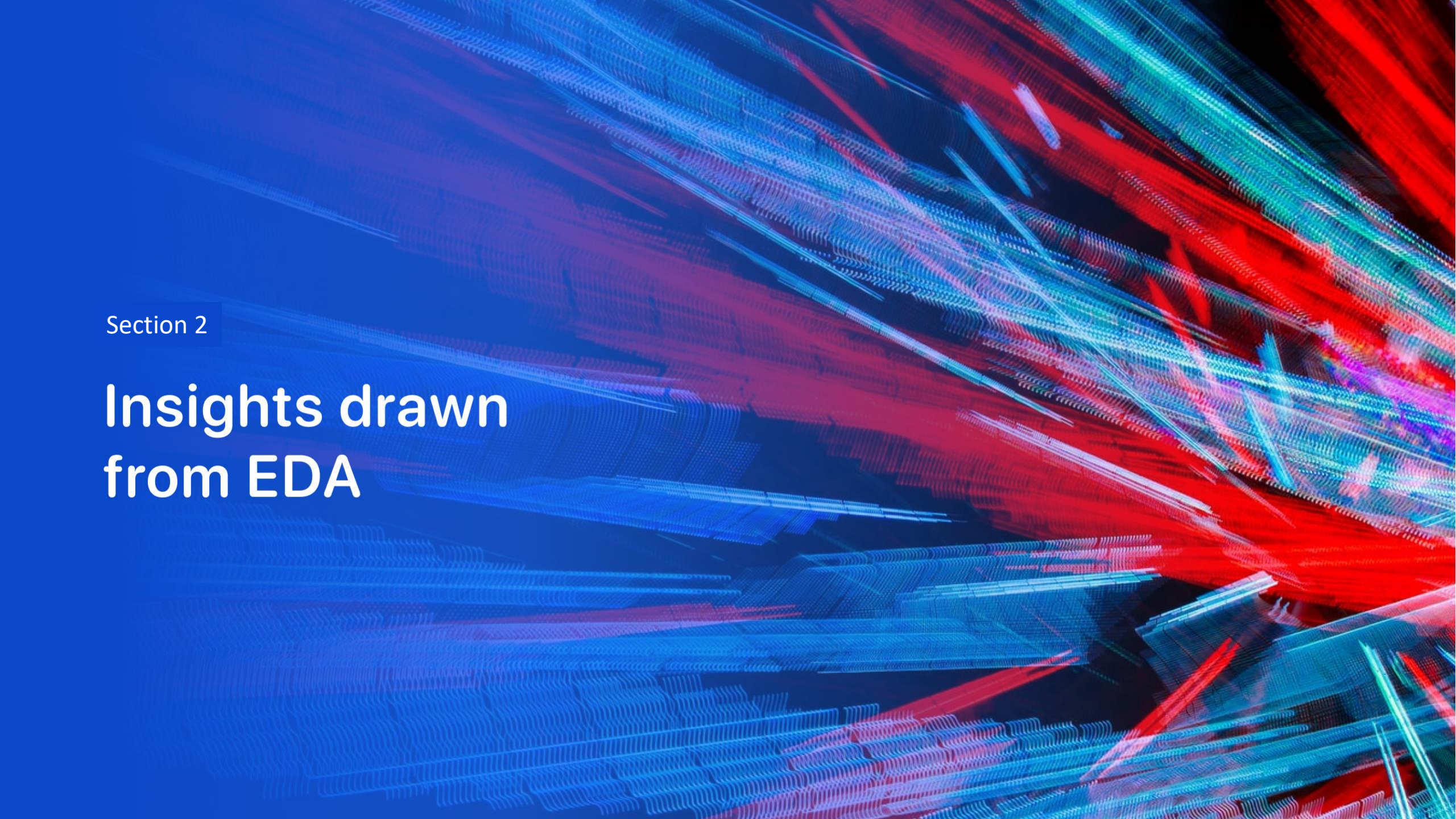
We will import the following libraries for the lab

```
# Pandas is a software library written for the Python programming language for data
import pandas as pd
# NumPy is a library for the Python programming language, adding support for large,
import numpy as np
# Matplotlib is a plotting library for python and pyplot gives us a MatLab like plot
import matplotlib.pyplot as plt
#Seaborn is a Python data visualization library based on matplotlib. It provides a h
import seaborn as sns
# Preprocessing allows us to standarsize our data
from sklearn import preprocessing
# Allows us to split our data into training and testing data
from sklearn.model_selection import train_test_split
# Allows us to test parameters of classification algorithms and find the best one
from sklearn.model_selection import GridSearchCV
# Logistic Regression classification algorithm
from sklearn.linear_model import LogisticRegression
# Support Vector Machine classification algorithm
from sklearn.svm import SVC
# Decision Tree classification algorithm
from sklearn.tree import DecisionTreeClassifier
# K Nearest Neighbors classification algorithm
from sklearn.neighbors import KNeighborsClassifier
```

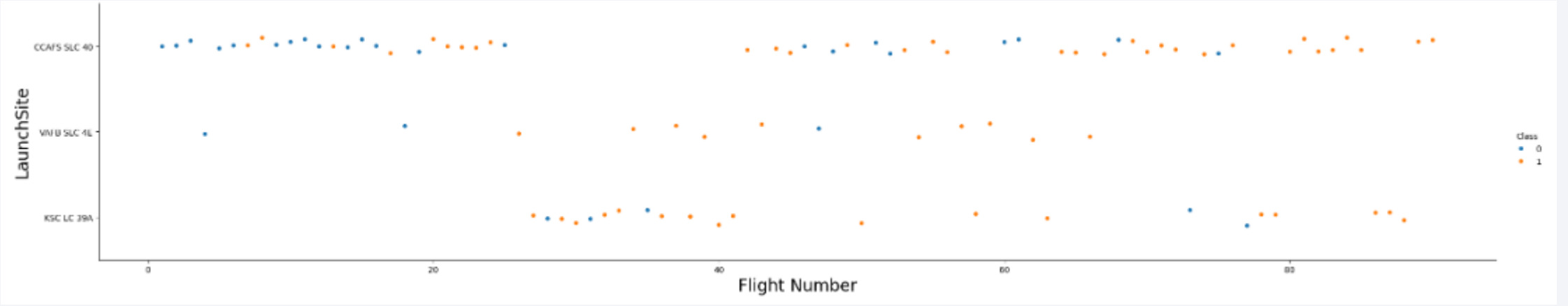

GitHub URL for notebook: https://github.com/shalinimoorthy29/IBM-Data-Science-Capstone/blob/main/077_SpaceX_Machine%20Learning%20Prediction.ipynb

Results

- Results from data analysis with visualization libraries
- Results from data analysis with SQL
- Screenshots from interactive data analysis with Folium
- Screenshots from dashboard using Plotly dash
- Results from predictive analysis

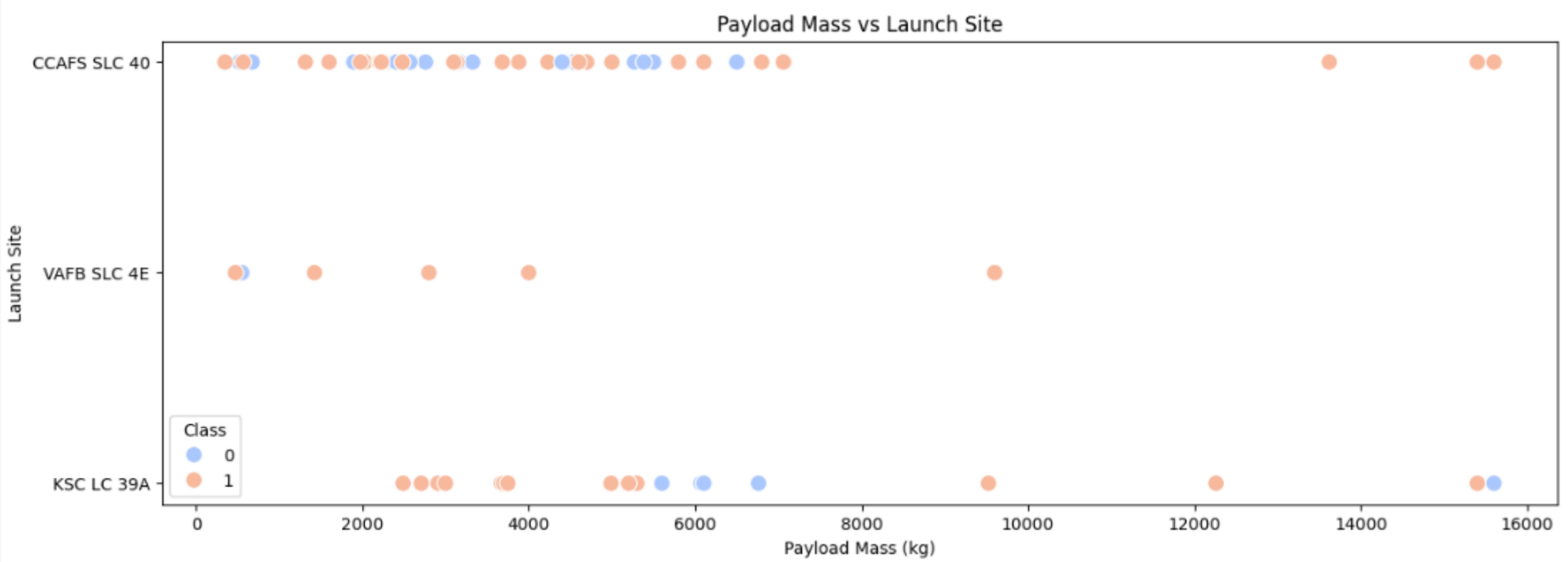

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



Launches from the site CCAFS SLC 40 are much higher than the other two sites

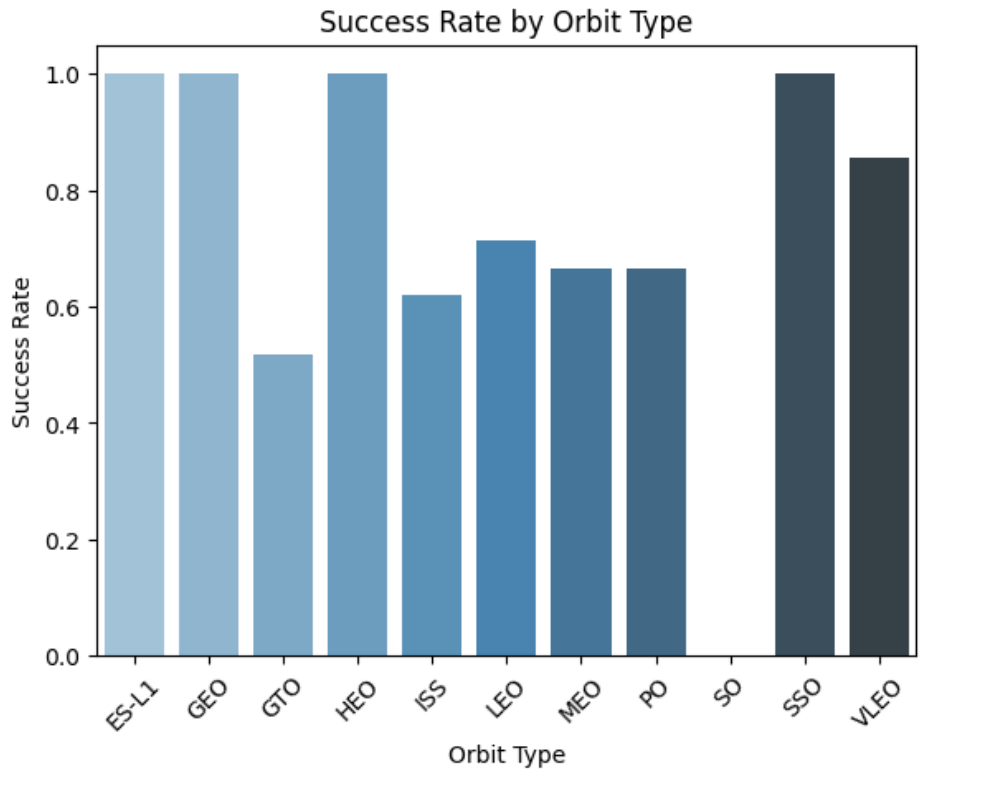
Payload vs. Launch Site



Majority of flights with payload mass less than 10000kg have been launched from site CCAFS SLC 40.

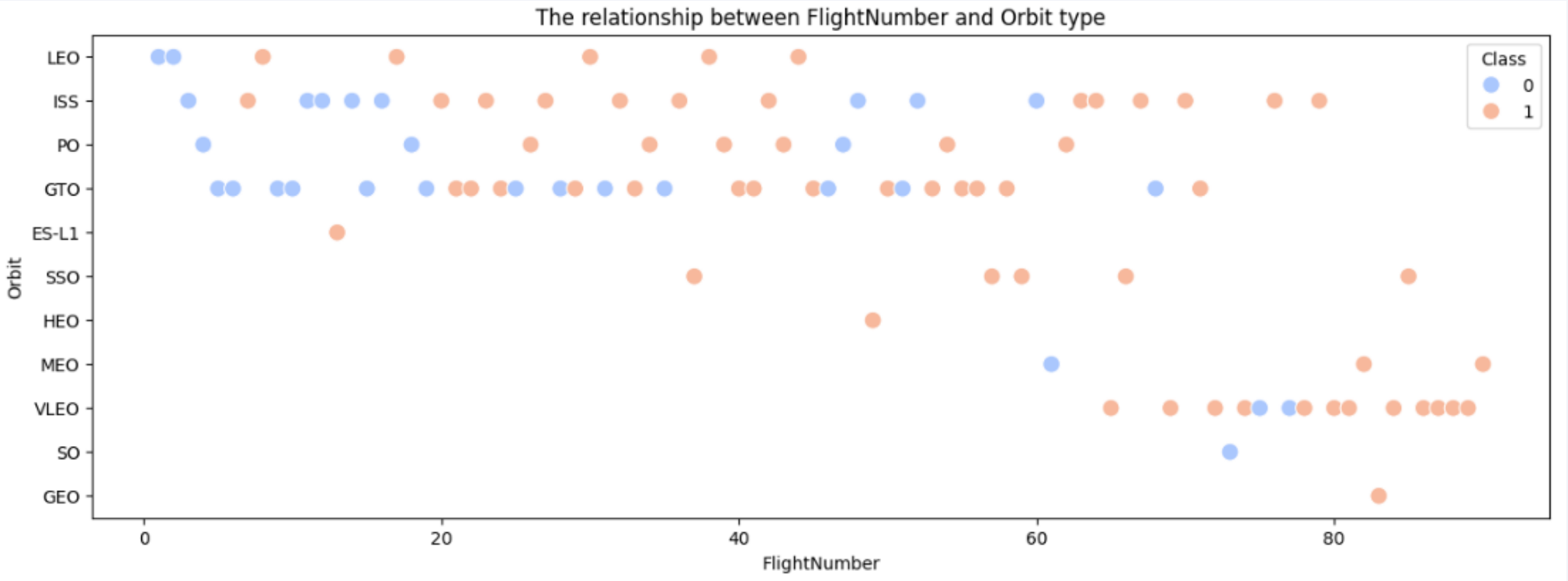
At the VAFB-SLC launch site there are no flights launched for heavy payload mass (greater than 10000).

Success Rate vs. Orbit Type



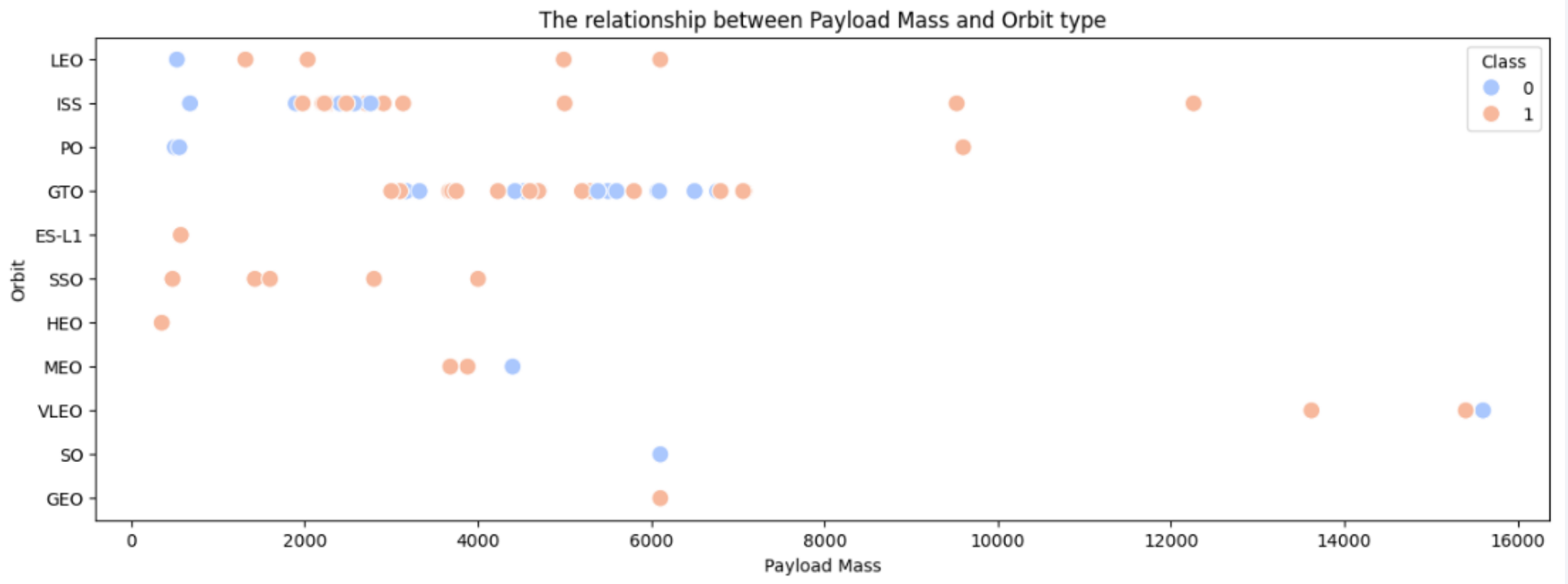
The best success rates are for orbit types ES-L1, GEO, HEO and SSO

Flight Number vs. Orbit Type



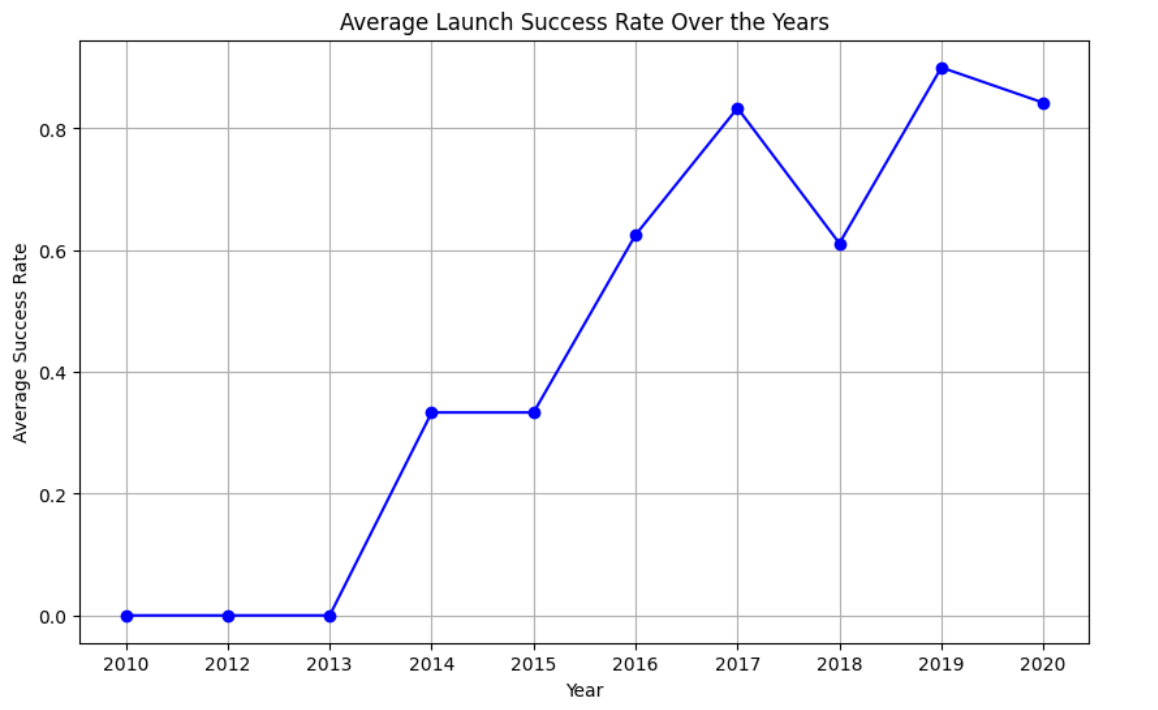
It can be observed that in the Low Earth Orbit (LEO), the success of a mission seems to correlate with the number of flights. However, in the Geostationary Transfer Orbit (GTO), there appears to be no clear relationship between the flight number and the success of the mission.

Payload vs. Orbit Type



With heavy payloads, the successful landing rate tends to be higher for missions to Polar, Low Earth Orbit (LEO), and the International Space Station (ISS). However, for Geostationary Transfer Orbit (GTO), it is harder to differentiate between successful and unsuccessful landings, as both outcomes are frequently observed.

Launch Success Yearly Trend



The success rate of launches has steadily increased since 2013, continuing to rise through to 2020. This suggests ongoing improvements in technology and operational efficiency over this period.

All Launch Site Names

This SQL query retrieves all the unique launch sites from the **SPACEXTABLE** by selecting distinct values from the "Launch_Site" column. It eliminates duplicates to provide a list of unique launch locations.

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

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

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

Launch Site Names Begin with 'CCA'

This SQL query retrieves all columns (`*`) from the **SPACEXTABLE** where the launch site name starts with 'CCA'. The `LIKE 'CCA%'` condition filters records for launch sites whose names begin with 'CCA', and the query is limited to return only the first five results.

```
%sql SELECT * FROM 'SPACEXTABLE' 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
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

This query calculates the total payload mass (in kilograms) launched by NASA under the CRS (Commercial Resupply Services) program. It sums up the `PAYLOAD_MASS__KG_` for all records where the `Customer` is 'NASA (CRS)', displaying the total payload mass along with the customer name.

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS "Total Payload Mass(Kgs)", Customer FROM 'SPACEXTABLE' WHERE Customer = 'NASA (CRS)';
```

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

Total Payload Mass(Kgs)	Customer
45596	NASA (CRS)

Average Payload Mass by F9 v1.1

This query computes the average payload mass (in kilograms) for launches using the 'F9 v1.1' booster version. It filters the results to include only records where the `Booster_Version` begins with 'F9 v1.1%', and returns the average payload mass, along with the corresponding customer and booster version.

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS "Average Payload Mass Kgs", Customer, Booster_Version FROM 'SPACEXTABLE' WHERE Booster_Version LIKE 'F9 v1.1%';
```

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

Average Payload Mass Kgs	Customer	Booster_Version
2534.6666666666665	MDA	F9 v1.1 B1003

First Successful Ground Landing Date

This query retrieves the earliest date (`MIN(Date)`) when a successful landing occurred on a ground pad. It filters the records to include only those where the `Landing_Outcome` is 'Success (ground pad)', identifying the first successful ground landing in the dataset.

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

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

First_Successful_Landing
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

This query retrieves the distinct booster versions that successfully landed on a drone ship, with a payload mass between 4000 kg and 6000 kg. It filters the results to only include rows where the `Landing_Outcome` is 'Success (drone ship)' and the `PAYLOAD_MASS__KG_` falls within the specified range.

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

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

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

Total Number of Successful and Failure Mission Outcomes

This query groups the data by the `Mission_Outcome` column and counts how many times each unique outcome occurs. The result shows the total number of missions for each possible mission outcome in the `SPACEXTABLE`.

```
%sql SELECT "Mission_Outcome", COUNT(*) AS Total FROM SPACEXTABLE GROUP BY "Mission_Outcome";
```

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

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

Boosters Carried Maximum Payload

This query retrieves the `Booster_Version` associated with the maximum payload mass recorded in the `SPACEXTABLE`. It uses a subquery to first find the highest payload mass and then selects the corresponding booster version that achieved this maximum payload.

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

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

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

2015 Launch Records

This query extracts the month, landing outcome, booster version, and launch site from the `SPACEXTABLE` for records in the year 2015 where the landing outcome was a failure on a drone ship. The `CASE` statement converts the month number (extracted from the "Date" column) into its corresponding name, allowing for clearer interpretation of the results.

```
%%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,
    "Landing_Outcome",
    "Booster_Version",
    "Launch_Site"
FROM SPACEXTABLE
WHERE substr("Date",0, 5) = '2015'
    AND "Landing_Outcome" = 'Failure (drone ship)';
```

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

Month	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

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

This query counts the number of launches categorized by their landing outcomes from the `SPACEXTABLE` between the specified dates (June 4, 2010, to March 20, 2017). The results are grouped by landing outcome, and the `ORDER BY` clause sorts the outcomes in descending order of their counts, allowing for quick identification of the most common landing outcomes within that date range.

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

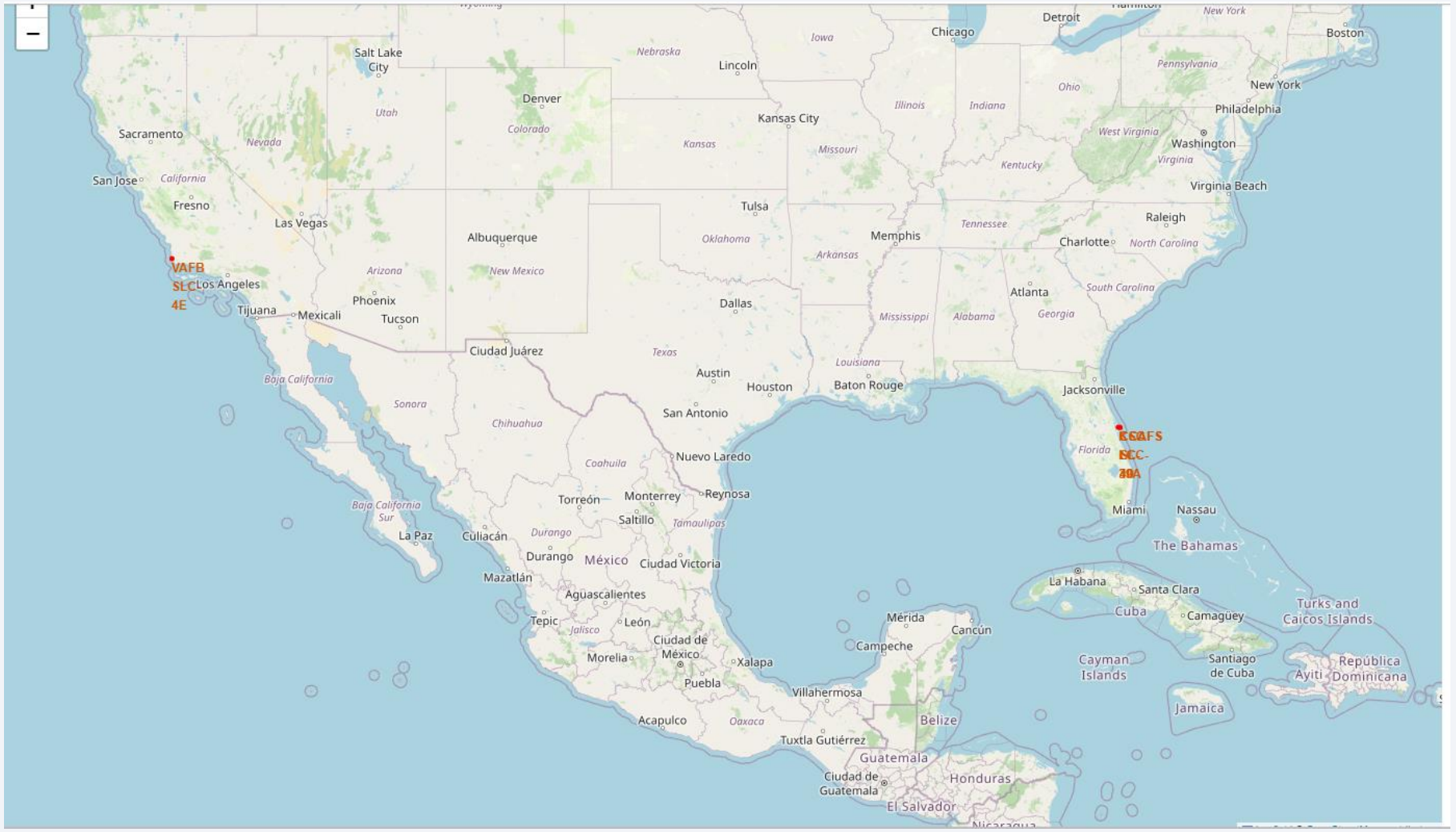
```
 * sqlite:///my_data1.db
Done.
```

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

Section 3

Launch Sites Proximities Analysis

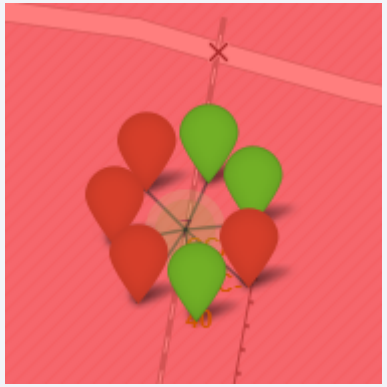
Marking of all launch sites



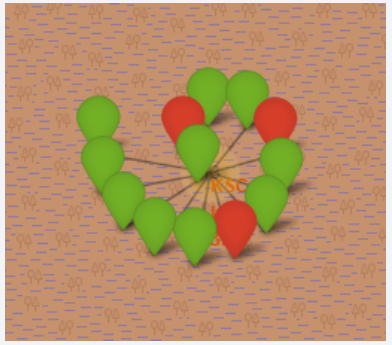
Coloured markings on launch outcomes at various sites

CCAFS LC-40

CCAFS SLC-40



KSC LC-39A



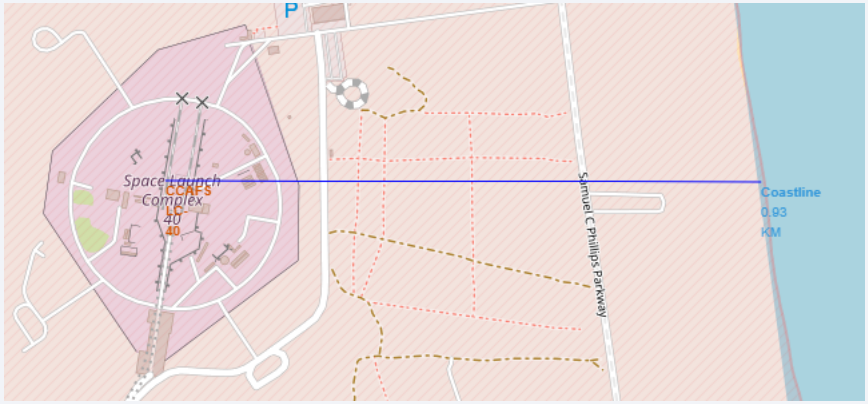
VAFB SLC-4E

Green = Successful

Red = Failed

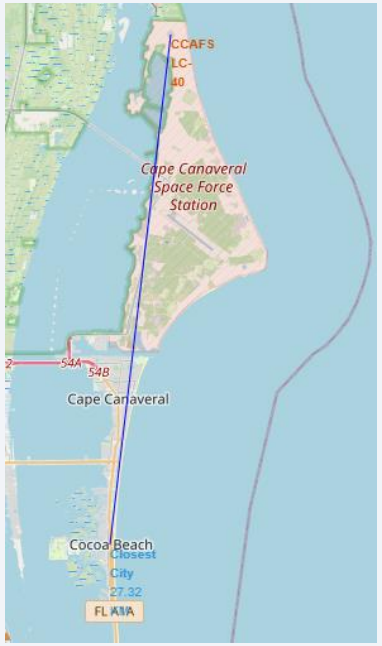
Distance from selected launch site to closest coastline, railroad, city and highway

Proximity to coastline

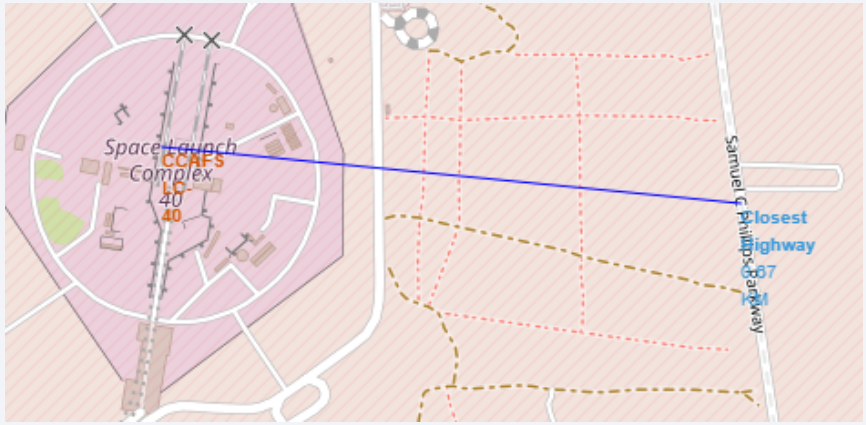


Proximity to railroad

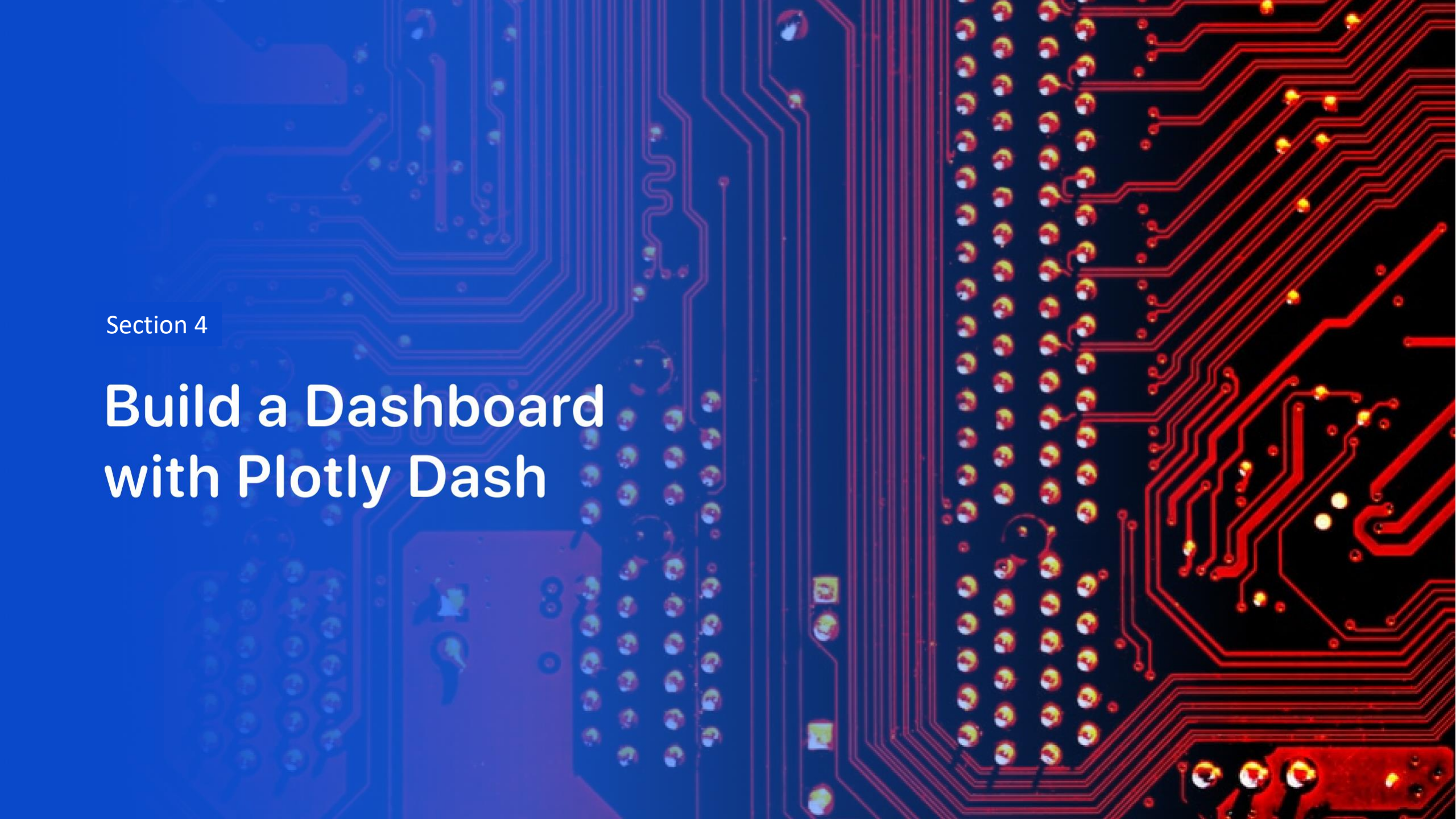
Proximity to city



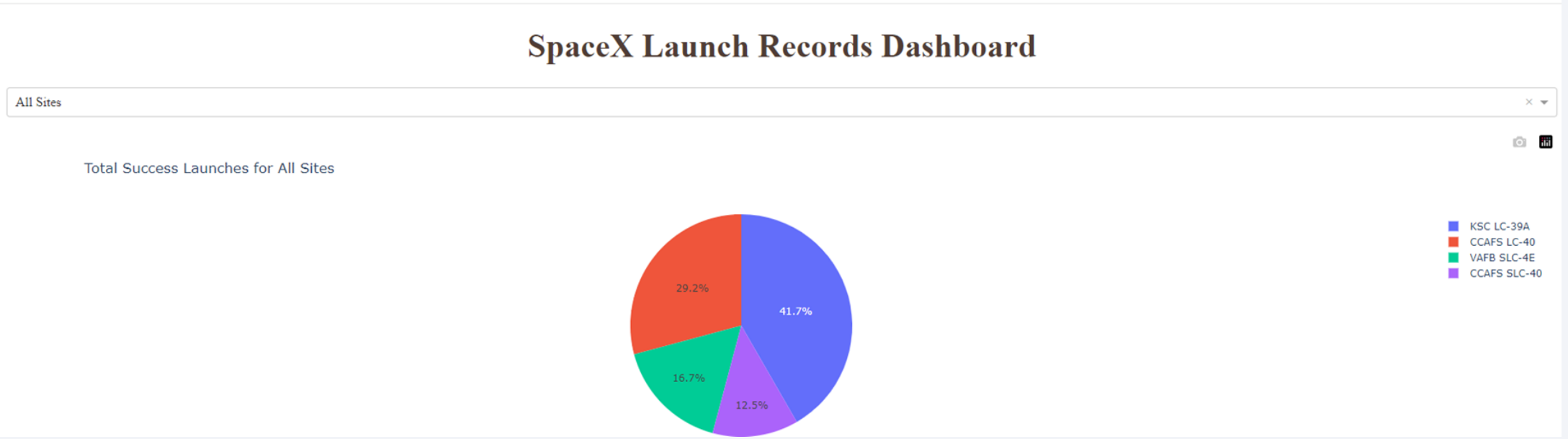
Proximity to highway



- Launch site is not particularly close to railroads, highways or cities. This could be due to safety reasons.
- Launch site is closer to coastline compared to the other landmarks.

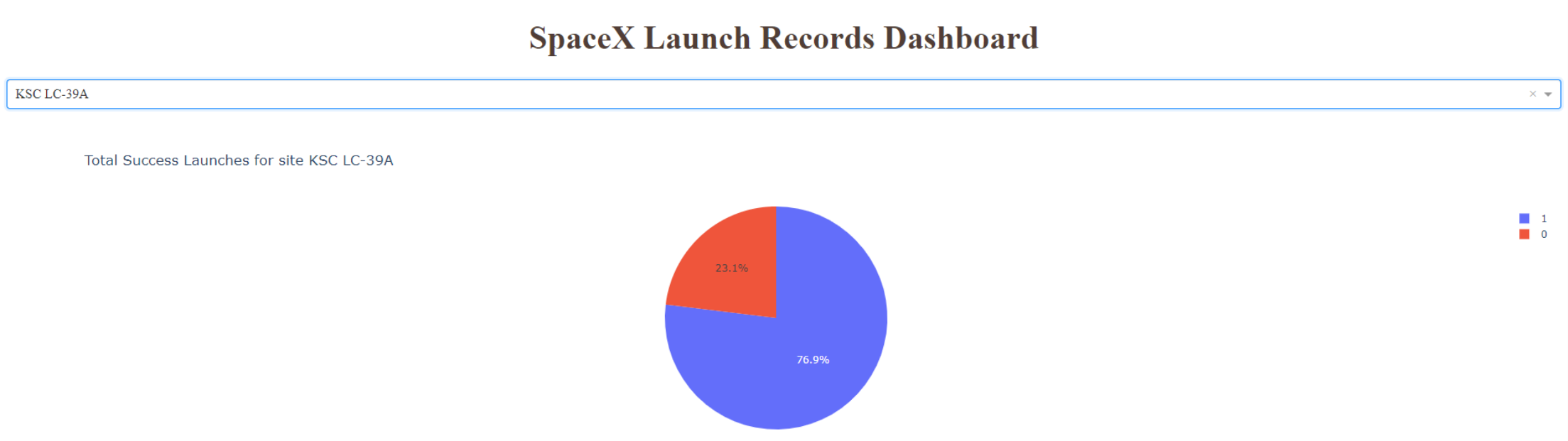

Section 4
Build a Dashboard with Plotly Dash

Launch success count



KSC LC -39A has had the most successful launches

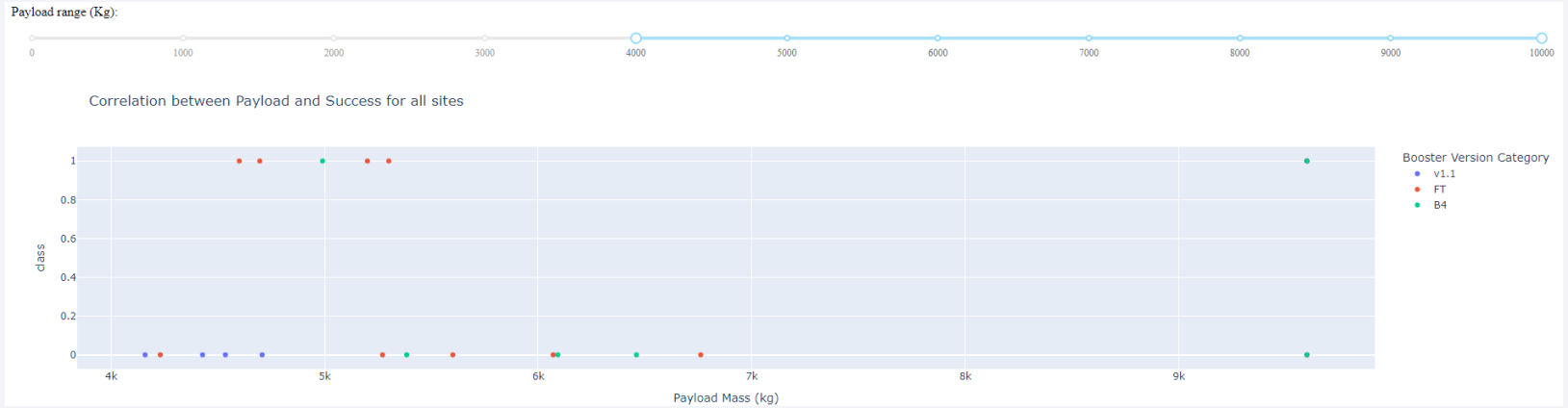
Launch site with the highest launch success ratio



KSC LC -39A has had highest launch success ratio

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

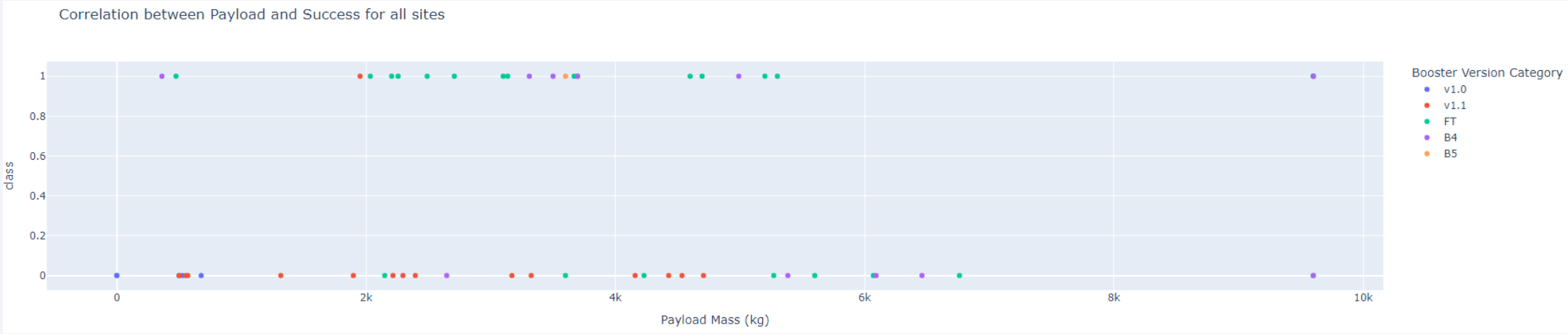
0 – 4000kg

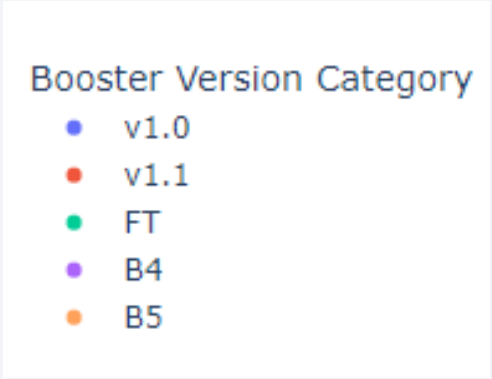


0 – 10000kg

Success rate is better overall for lower payload mass range compared to the higher range.

F9 Booster version and launch success rate





The FT booster version has the highest launch success rate

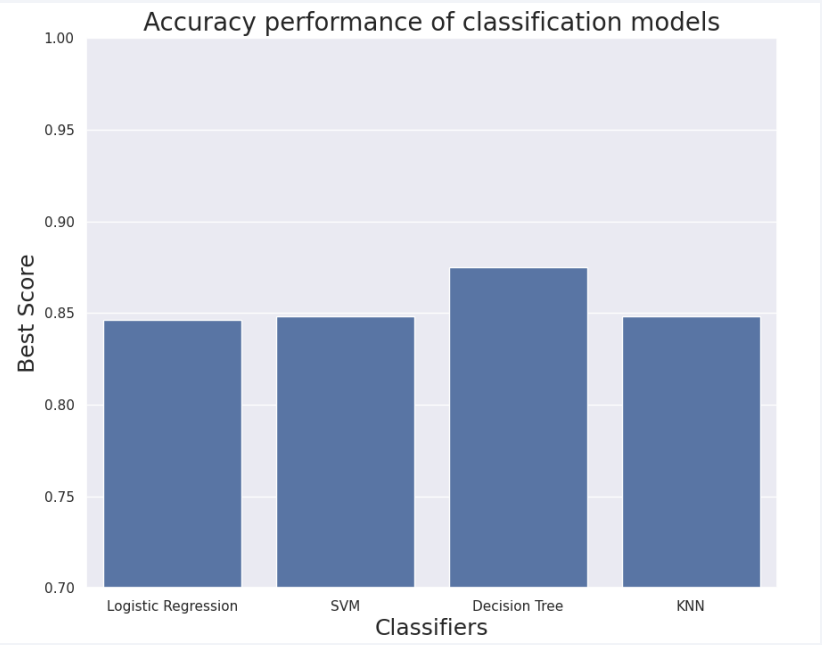
Section 5

Predictive Analysis (Classification)

Classification Accuracy

```
# Dictionary of model names and their best scores
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

# Determine the best model
bestalgorithm = max(models, key=models.get)

# Print the best model and its score
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])

# Print the best parameters for the best model
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)
```

```
Best model is DecisionTree with a score of 0.875

Best params is : {'criterion': 'gini', 'max_depth': 8, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 5, 'splitter': 'best'}
```

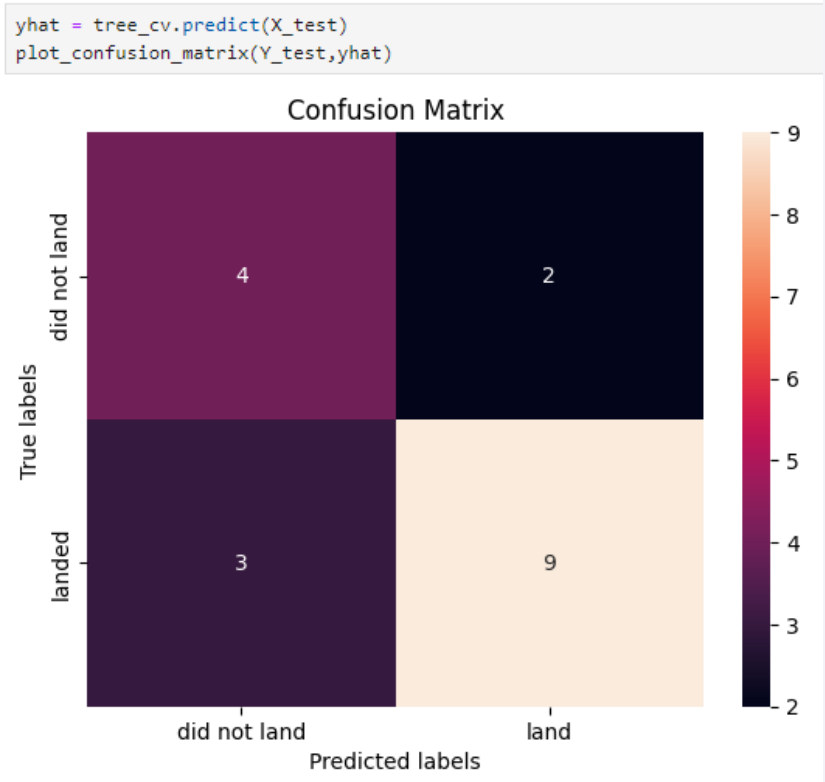


DecisionTree model performs the best with a score of 0.875

Confusion Matrix

DecisionTree

```
yhat = tree_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



The confusion matrix provides insights into the performance of your classification model, which predicts whether a Falcon 9 booster landed or did not land.

True Negatives (Top-Left, 4): The model correctly predicted 4 instances where the booster did not land.

False Positives (Top-Right, 2): The model incorrectly predicted that 2 boosters landed, but they did not.

False Negatives (Bottom-Left, 3): The model incorrectly predicted that 3 boosters did not land, but they actually did.

True Positives (Bottom-Right, 9): The model correctly predicted 9 instances where the booster landed.

This confusion matrix shows that your model is performing reasonably well but tends to misclassify a few landings and non-landings. If the main goal is to maximize the prediction of successful landings, focusing on improving recall could help reduce the number of false negatives.

Accuracy of test data

```
# Make predictions using the test data
logreg_test_accuracy = logreg_cv.score(X_test, Y_test)
svm_test_accuracy = svm_cv.score(X_test, Y_test)
tree_test_accuracy = tree_cv.score(X_test, Y_test)
knn_test_accuracy = knn_cv.score(X_test, Y_test)

print("Logistic Regression Test Accuracy: ", logreg_test_accuracy)
print("SVM Test Accuracy: ", svm_test_accuracy)
print("Decision Tree Test Accuracy: ", tree_test_accuracy)
print("KNN Test Accuracy: ", knn_test_accuracy)
```

```
Logistic Regression Test Accuracy:  0.8333333333333334

SVM Test Accuracy:  0.8333333333333334

Decision Tree Test Accuracy:  0.8333333333333334

KNN Test Accuracy:  0.8333333333333334
```

Conclusions

- **Flight Amount and Success Rate** – A positive correlation was identified between the number of launches at a site and the success rate, which suggests operational experience may contribute to better outcomes.
- **Launch Success Rate Over Time** - The trend from 2013 to 2020 shows an increase in success rate, likely due to technological improvements, experience, and operational refinements over the years.
- **Successful Orbits** - Specific orbits like ES-L1, GEO, HEO, SSO, and VLEO seem to have a higher success rate, possibly due to the familiarity with these missions and refined launch procedures for those orbits.
- **Most Successful Launch Site** - KSC LC-39A being the most successful site is an important operational insight, as it indicates that this site is optimal for launches, possibly due to infrastructure, location, or other factors.
- **Best Machine Learning Algorithm** - After evaluating the different classifiers, it appears the Decision Tree classifier performed the best, suggesting that it is well-suited for predicting launch success based on the available features.

These conclusions can be used to guide further exploration of launch site optimization, orbital success factors, and future machine learning tasks with a focus on improving predictive accuracy.

Thank you!