Winning Space Race with Data Science

Tetiana Rospopchuk
17.11.2024

Outline

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

Executive Summary

- Summary of methodologies:
  - API requests, Python accompanied with different packages such as pandas, BeautifulSoup were for data gathering, manipulation and analysis
- . Summary of all results:
  - Discovered the success rate, number of unique values and missing values, dealing with inconsistent data defined within the project

Introduction

- Project background and context

  The commercial space industry is growing rapidly, with companies like Virgin Galactic, Rocket Lab, Blue Origin, and SpaceX leading the way. SpaceX, in particular, has revolutionized space travel by making rocket launches significantly more affordable through the reuse of its Falcon 9 first stage, which can land back on Earth after launch. This cost-saving strategy has made SpaceX a major player in space exploration, providing satellite services like Starlink and sending missions to the International Space Station.

- Problems you want to find answers

  The primary challenge is to predict the cost of a SpaceX launch, which largely depends on whether the Falcon 9 first stage will land successfully. Determining the likelihood of a successful landing, based on mission parameters and public data, could allow Space Y to competitively price its own launches. Additionally, by training a machine learning model, the goal is to predict when the first stage will land or crash, improving decision-making and cost predictions for future launches.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected from the website using API requests
- Perform data wrangling
  - Using pandas & numpy
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - create a column for the class, standardize the data, split into training data and test data, find best Hyperparameter for SVM, Classification Trees and Logistic Regression

Data Collection

1. **Define Objective**: The first step is to clearly define the goals of the data collection, such as predicting launch costs or determining the likelihood of successful first stage landings.
2. **Identify Data Sources**: Sources might include websites that track rocket launches, public APIs from space organizations like SpaceX, and historical datasets related to rocket missions.
3. **Select Data Collection Methods**: Depending on the sources, methods like web scraping (to gather data from websites), APIs (to get structured data), or manually curated datasets (if available) are chosen.
4. **Data Cleaning & Preprocessing**: This involves transforming raw data into a usable format, including tasks like filling in missing values, normalizing units, and removing duplicates.
5. **Data Integration**: If multiple sources of data are used, they need to be integrated (e.g., combining web scraped data with official launch records).
6. **Data Storage**: The collected and cleaned data is stored in an appropriate format, such as a CSV file, database, or cloud storage, for easy access and processing.
7. **Data Validation**: Validation ensures the data is accurate and relevant to the research question, including checking for outliers or errors in the data collection process.
8. **Ongoing Data Collection**: In some cases, data needs to be continuously collected, such as tracking real-time rocket launches or updating mission data.

Data Collection – SpaceX API

- Present your data collection with SpaceX REST calls using key phrases and flowcharts

- Add the GitHub URL

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

# assign the response to a object
response =requests.get(static_url)
response.status_code
# response.json()
```

```
200
```

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

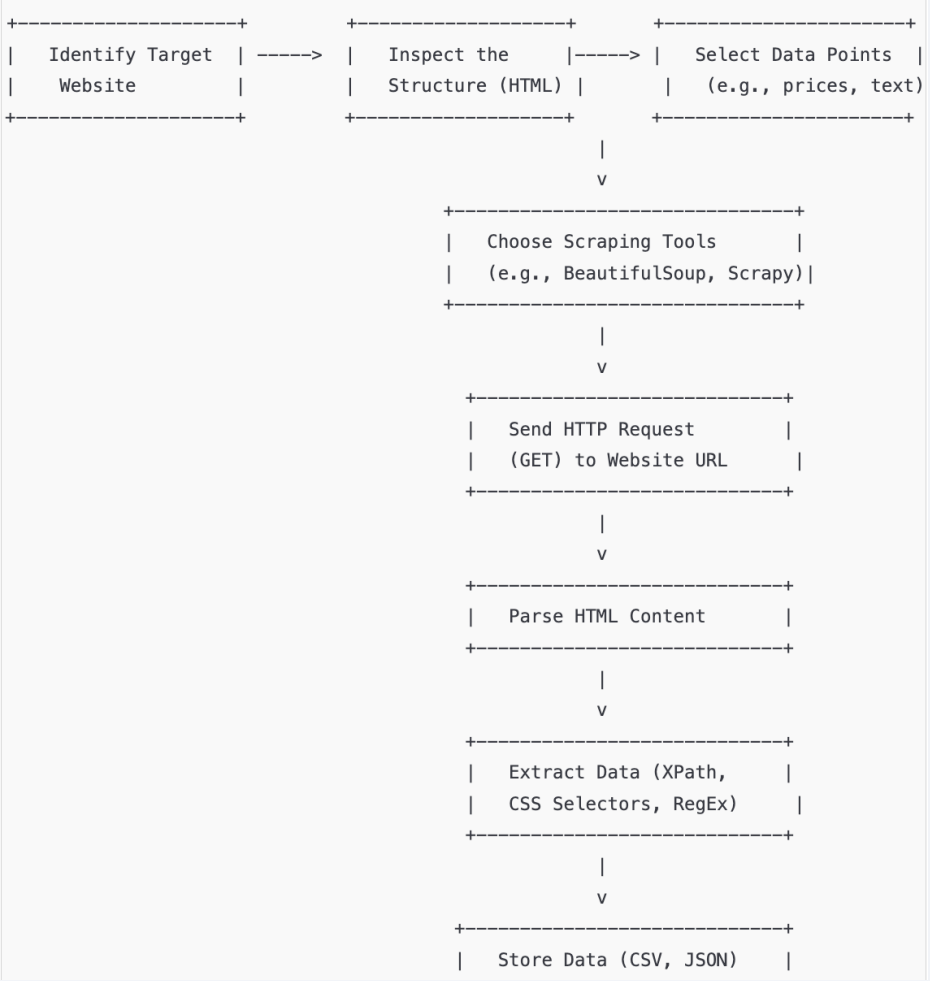
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.text, 'html.parser')

# Step 3: Print or work with the parsed HTML
print(soup.prettify())
```

```
<!DOCTYPE html>
<html class="client-nojs vector-feature-language-in-header-enabled vector-feature-language-in-main-page-header-disabled
vector-feature-sticky-header-disabled vector-feature-page-tools-pinned-disabled vector-feature-toc-pinned-clientpref-1
vector-feature-main-menu-pinned-disabled vector-feature-limited-width-clientpref-1 vector-feature-limited-width-content
```

Data Collection - Scraping

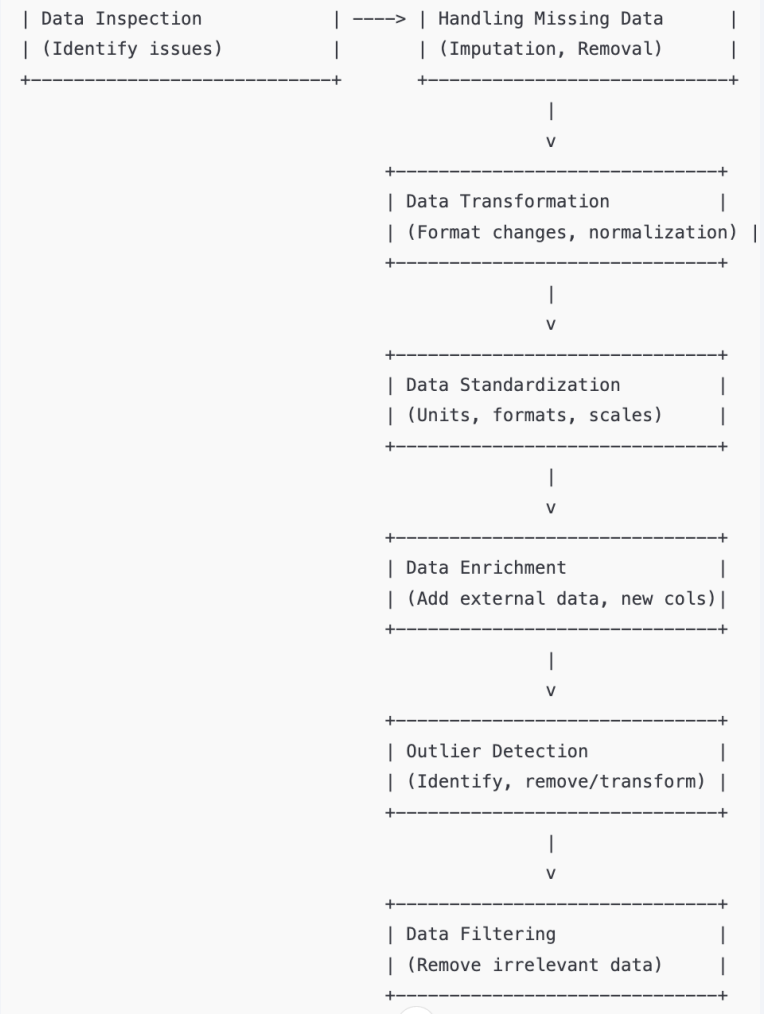
- Present your web scraping process using key phrases and flowcharts

- Add the GitHub URL



Data Wrangling

1. **Data Inspection** – Examine the raw data to identify issues such as missing values, inconsistencies, or outliers.
2. **Handling Missing Data** – Address any gaps in the data through methods like imputation or removal.
3. **Data Transformation** – Apply necessary transformations to make the data compatible with the analysis (e.g., changing formats, normalization).
4. **Data Standardization** – Standardize units, formats, and scales (e.g., ensuring all dates are in the same format, prices are in the same currency).
5. **Data Enrichment** – Add additional data (e.g., adding new columns based on existing data or combining external data sources).
6. **Outlier Detection** – Identify and handle outliers that could skew the analysis (e.g., removing extreme values or using transformations).
7. **Data Filtering** – Remove irrelevant data that doesn't contribute to the analysis (e.g., filtering out certain columns or rows).
8. **Feature Engineering** – Create new features from existing data that might improve model performance (e.g., calculating ratios, aggregating data).
9. **Data Aggregation** – Summarize data at the required level of granularity (e.g., group by launch date or mission type).
10. **Data Validation** – Ensure the processed data is accurate and ready for analysis (e.g., checking for duplicates or verifying consistency).



EDA with Data Visualization

- **1. Bar Chart -** Comparing quantities across different categories.

Example: Comparing the accuracy of different machine learning models.

- **2. Line Chart -** Showing trends over time or continuous data.

Example: Showing change of success rate YoY.

- **3. Pie Chart**

Example: Success vs False launch structure.

- **4. Scatter Plot**

Example: Plotting the relationship between features like Orbit type & PayloadMass

EDA with SQL

- Find the names of the unique launch sites
- Find 5 records where launch sites begin with `CCA`
- Calculate the total payload carried by boosters from NASA
- Calculate the average payload mass carried by booster version F9 v1.1
- Find the dates of the first successful landing outcome on ground pad
- Successful Drone Ship Landing with Payload between 4000 and 6000
- Total Number of Successful and Failure Mission Outcomes
- List the names of the booster which have carried the maximum payload mass
- List the failed landing outcomes in drone ship, their booster versions, and launch site names for in year 2015

Build an Interactive Map with Folium

- Marker (Launch Sites) : To represent the launch sites on the map.
  - Details:
    - Added markers for each launch site from the `spacex_df` DataFrame.
    - Customized the marker color based on the launch outcome (green for success, red for failure).
    - Added a popup to display the launch site name and class (success or failure).

- MarkerCluster: To group multiple markers (e.g., launch sites) into clusters for easier navigation when zoomed out.
  - Details:
    - A `MarkerCluster` object was used to add all the launch site markers.
    - It improves the map's performance and usability by grouping markers at similar zoom levels.

- Point of Interest Markers: To represent various points of interest (city, railway, highway).
  - Details:
    - Each point of interest has its own popup displaying its name.

- Launch Site Marker: To represent the launch site itself.
  - Details:
    - A dedicated marker was added for the launch site (e.g., "CCAFS LC-40").
    - The marker has an orange color and a cloud icon for visibility.

- Polyline (Connecting Launch Site and Closest Point): To visually connect the launch site to the closest point of interest.
  - Details:
    - A polyline was drawn between the launch site and the closest point of interest

Build a Dashboard with Plotly Dash

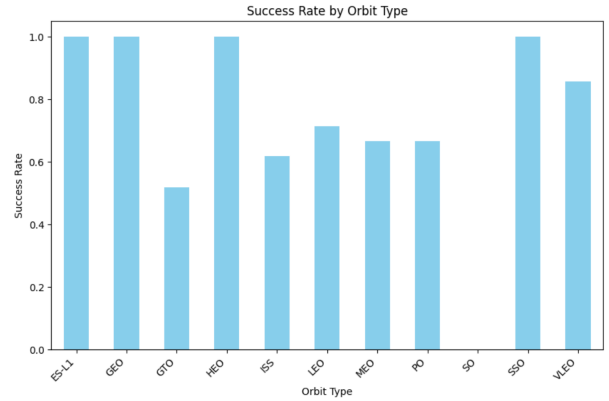
- Dropdown for Launch Site Selection (Task 1): The dropdown will allow users to select a specific launch site or view data for all sites.
- Pie Chart for Success and Failure Counts (Task 2): A pie chart will display the total successful vs failed launches based on the selected launch site.
- Slider for Payload Range (Task 3): A slider will allow users to filter the data by payload range.
- Scatter Chart for Payload vs Launch Success (Task 4): A scatter plot will show the correlation between payload mass and launch success, which will update based on both the selected launch site and payload range.

Predictive Analysis (Classification)

- Data Collection → Gather data
- Data Cleaning → Remove missing values, outliers
- Feature Engineering → Create new features, transform existing ones
- Feature Scaling → Normalize numerical data
- Model Selection → Logistic Regression, KNN, SVM, Decision Trees
- Train-Test Split → Split dataset into training/testing sets
- Hyperparameter Tuning → Use GridSearchCV to optimize parameters
- Cross-Validation → Ensure robust model performance
- Model Evaluation → Evaluate using accuracy or other metrics
- Model Improvement → Tune features, use ensemble methods
- Best Model Selection → Choose model with highest performance
- Final Evaluation → Final test and model validation

Results

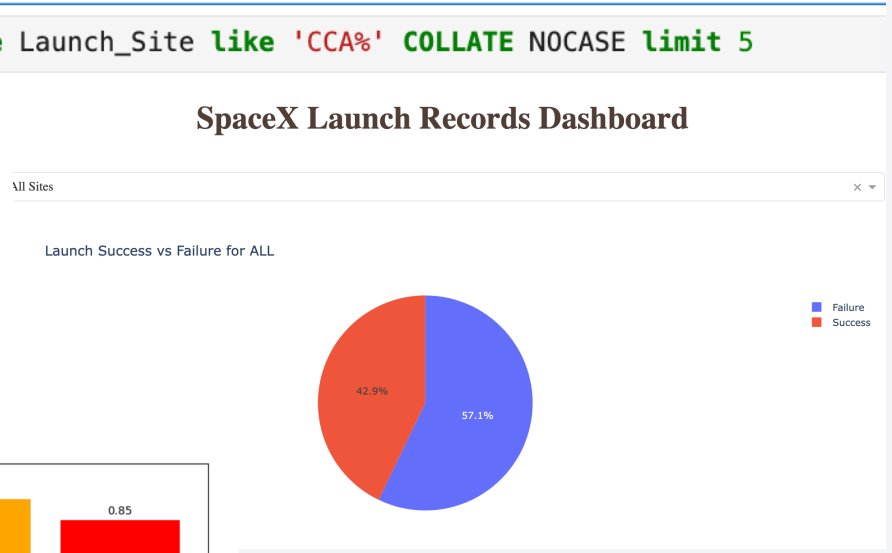
```
#Orbit success rate
orbit_success_rate = df.groupby('Orbit')["Class"].mean()
# Plot the bar chart
plt.figure(figsize=(10, 6))
orbit_success_rate.plot(kind='bar', color='skyblue')

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

# Show the plot
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for readability
plt.show()
```

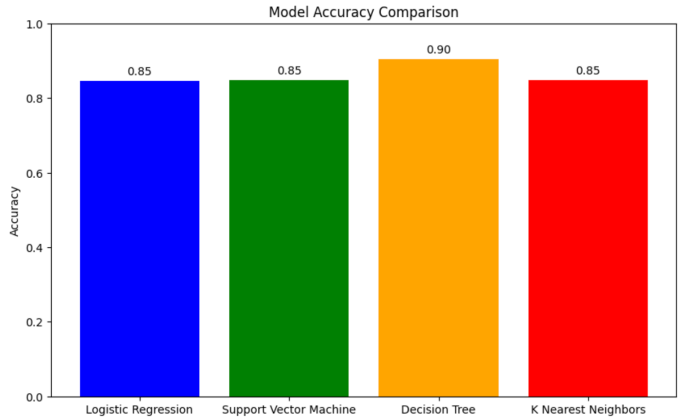



```
%sql select Launch_Site from SPACEXTBL where Launch_Site like 'CCA%' COLLATE NOCASE limit 5
```

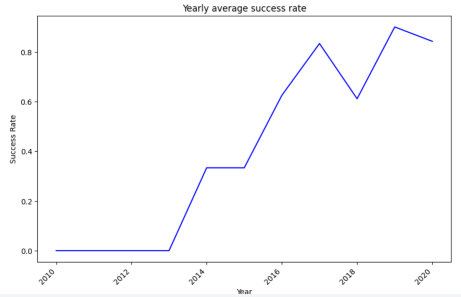
```
 * sqlite:///my_data1.db
Done.
```

Launch_Site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

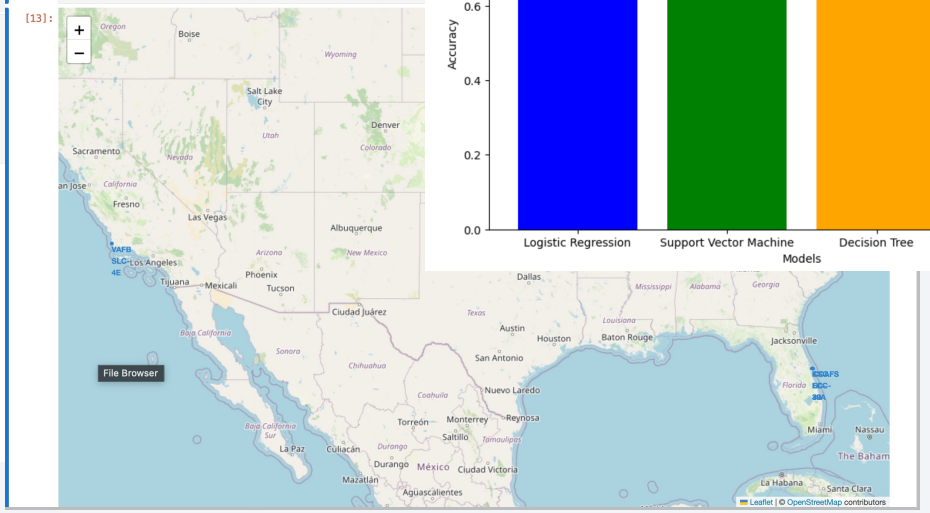


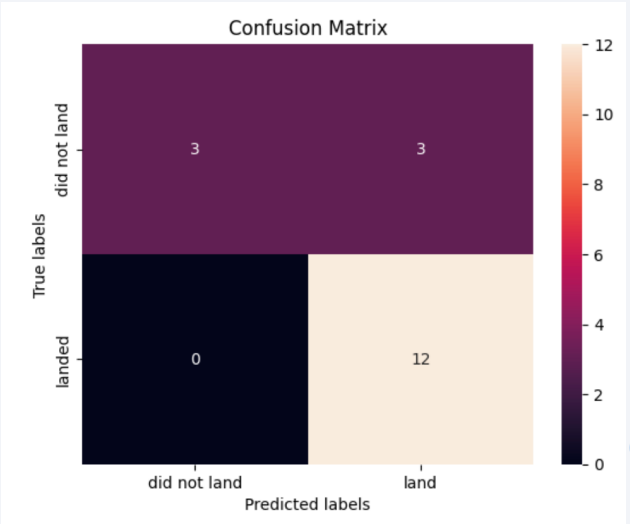


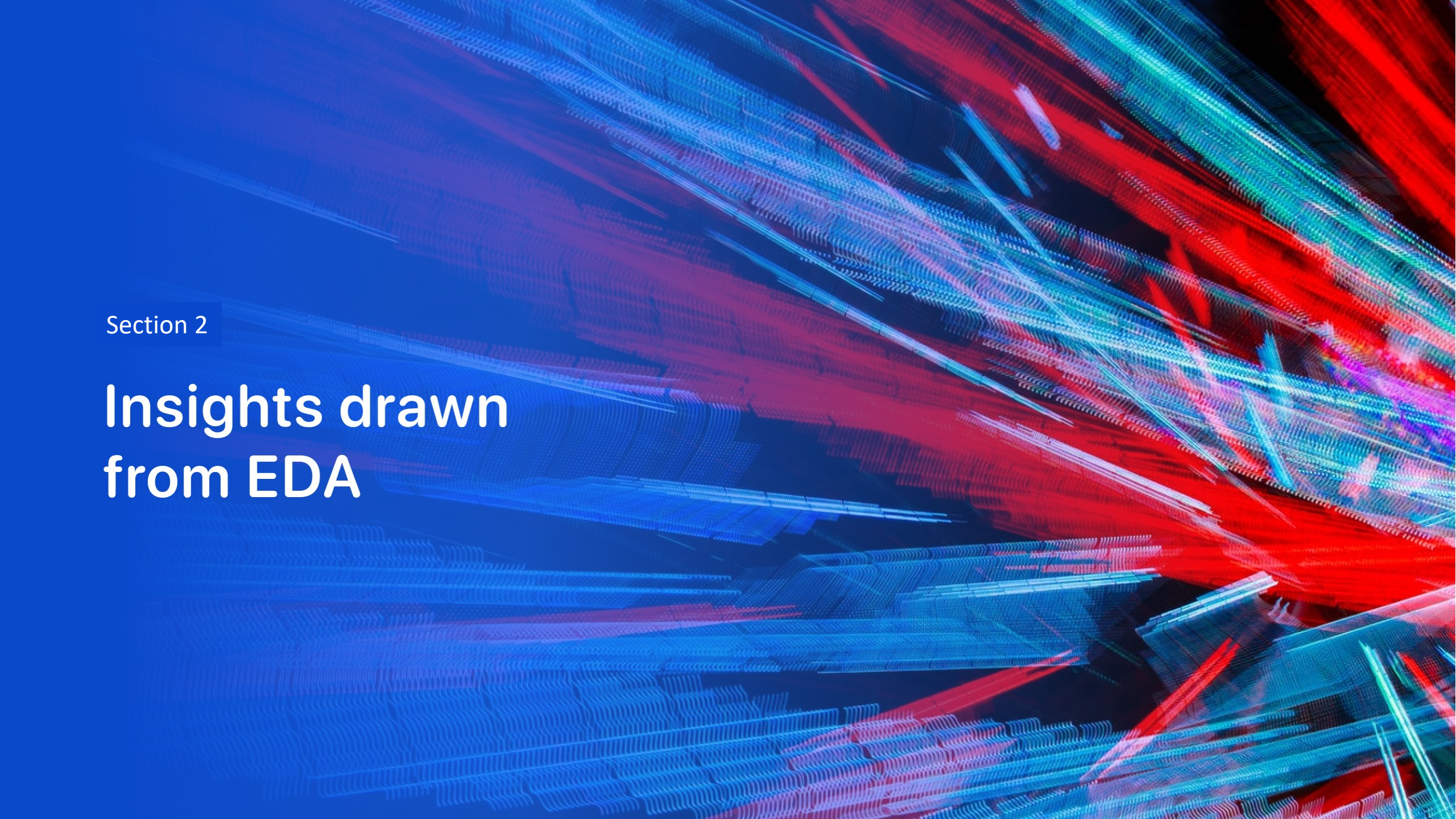

```
df['Year'] = pd.to_datetime(df['Date']).dt.year
#success rate
yearly_success_rate = df.groupby('Year')["Class"].mean()
# Plot the bar chart
plt.figure(figsize=(10, 6))
yearly_success_rate.plot(kind='line', color='blue')

# Add labels and title
plt.xlabel('Year')
plt.ylabel('Success Rate')
plt.title('Yearly average success rate')

# Show the plot
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for readability
plt.show()
```




```
[13]:
```



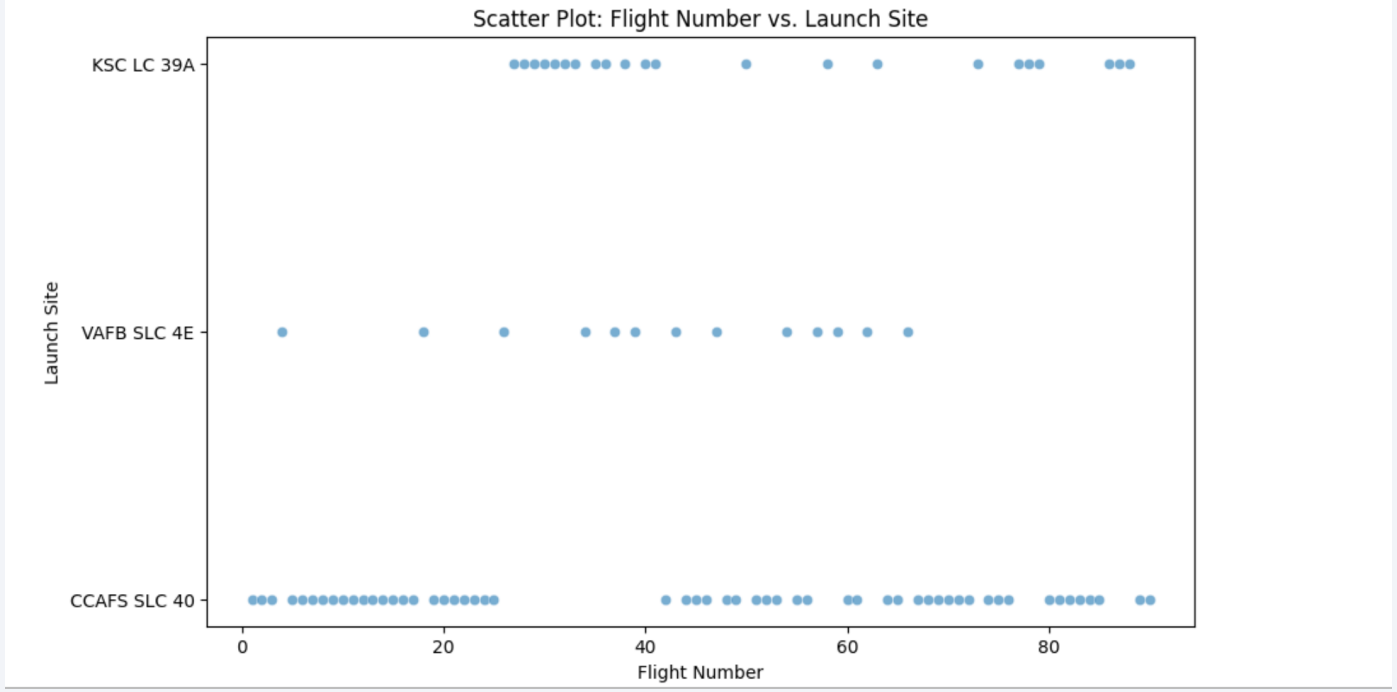



Section 2
Insights drawn from EDA

Flight Number vs. Launch Site

- CCAFS SLC 40 had the most number of flights

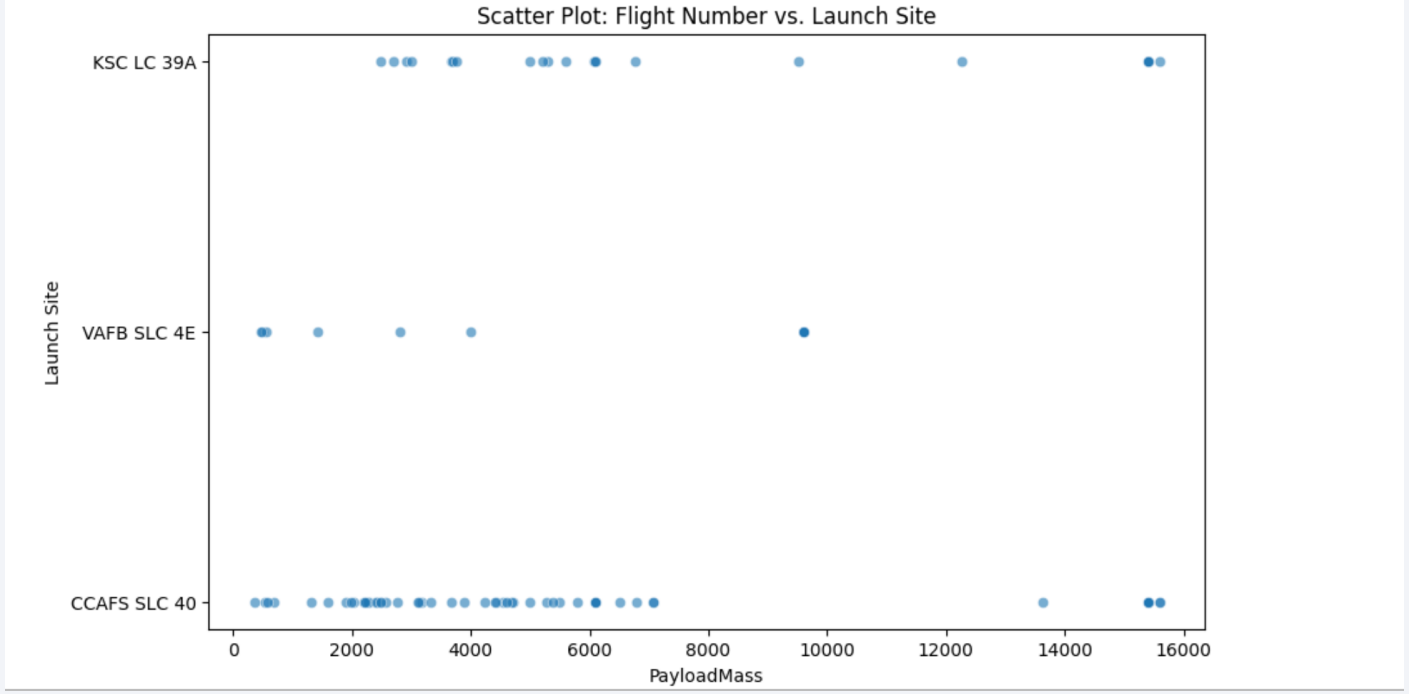
```
#Define parameters for plot
plt.figure(figsize=(10, 6))  # Set the figure size
plt.scatter(df['FlightNumber'], df['LaunchSite'], alpha=0.6, edgecolors="w", linewidth=0.5)

# Add labels and title
plt.xlabel('Flight Number')
plt.ylabel('Launch Site')
plt.title('Scatter Plot: Flight Number vs. Launch Site')

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



Payload vs. Launch Site

- CCAFS SLC 40 used for smaller payload mass

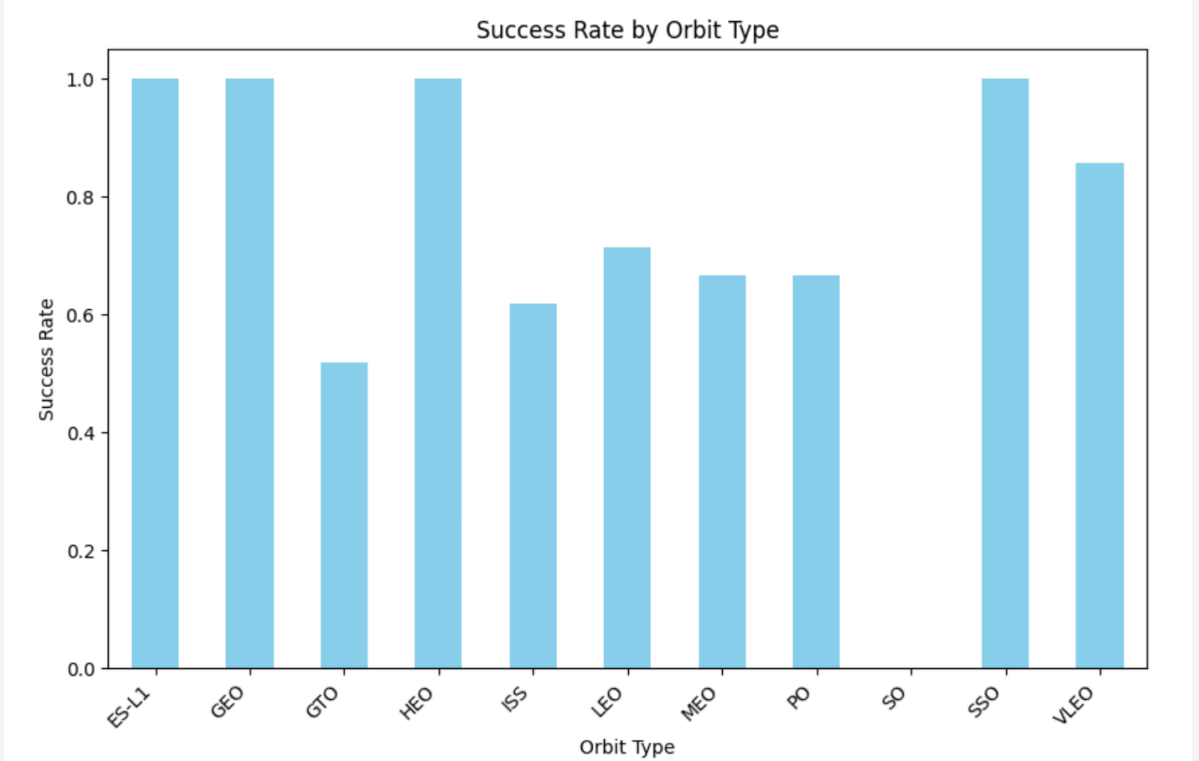
```
#Define parameters for plot
plt.figure(figsize=(10, 6))  # Set the figure size
plt.scatter(df['PayloadMass'], df['LaunchSite'], alpha=0.6, edgecolors="w", linewidth=0.5)

# Add labels and title
plt.xlabel('PayloadMass')
plt.ylabel('Launch Site')
plt.title('Scatter Plot: PayloadMass vs. Launch Site')

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



Success Rate vs. Orbit Type

- 3 out of 10 has 100% success rate, and only one below 50%

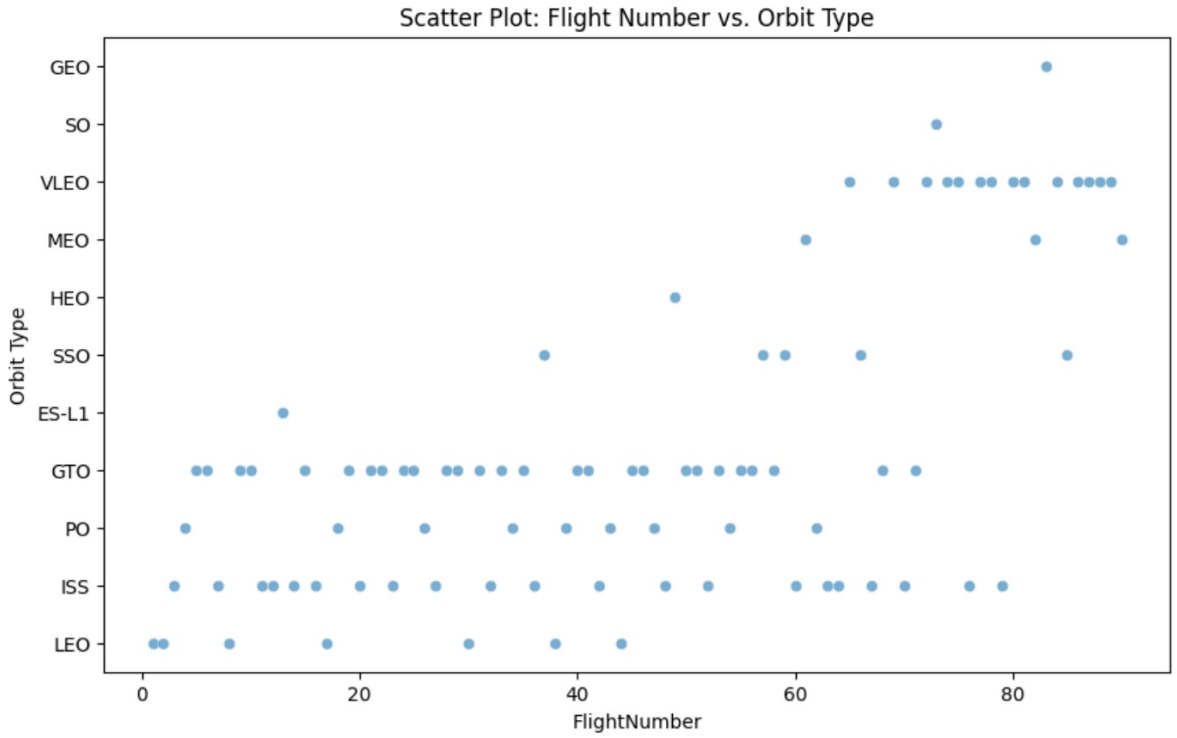
```
:  #Orbit success rate
   orbit_success_rate = df.groupby('Orbit')["Class"].mean()
   # Plot the bar chart
   plt.figure(figsize=(10, 6))
   orbit_success_rate.plot(kind='bar', color='skyblue')

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

   # Show the plot
   plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for readability
   plt.show()
```



Flight Number vs. Orbit Type

- Distribution of flights are not equal over Orbit types

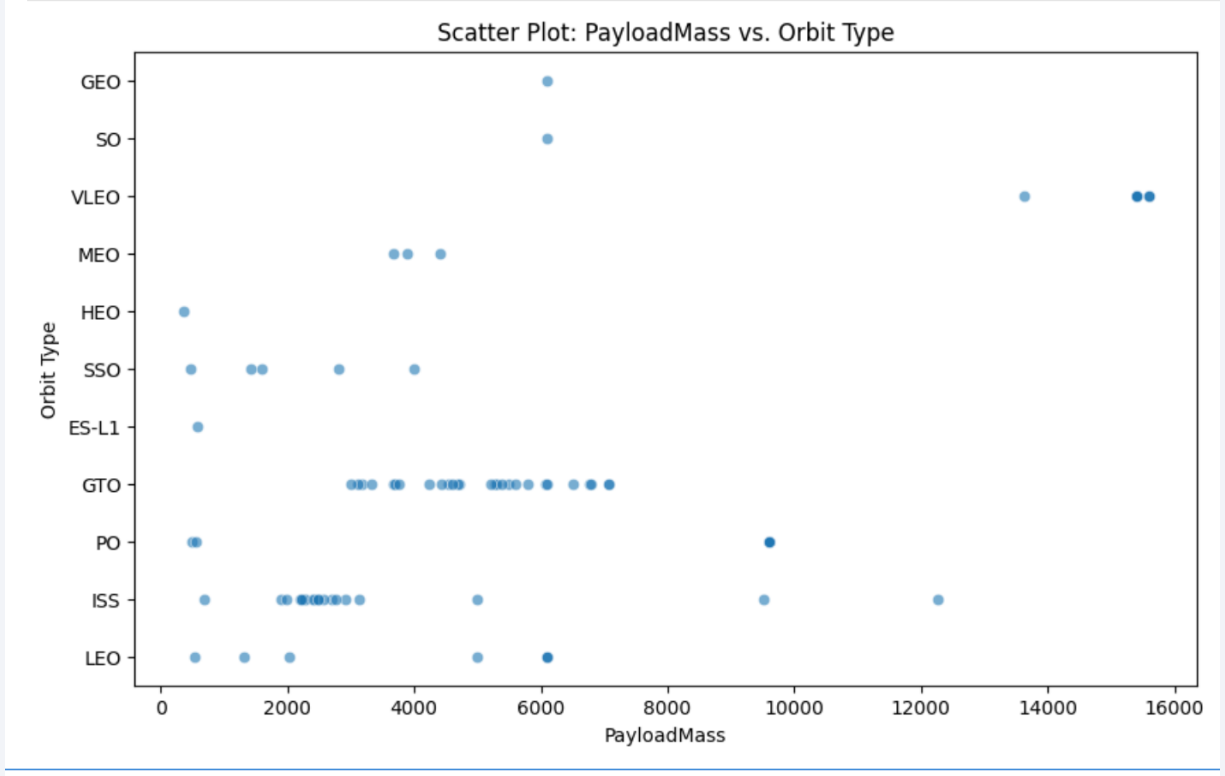
```
#Define parameters for plot
plt.figure(figsize=(10, 6))  # Set the figure size
plt.scatter(df['FlightNumber'], df['Orbit'], alpha=0.6, edgecolors="w", linewidth=0.5)

# Add labels and title
plt.xlabel('FlightNumber')
plt.ylabel('Orbit Type')
plt.title('Scatter Plot: Flight Number vs. Orbit Type')

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



Payload vs. Orbit Type

- Distribution of Payload Mass are not equal over Orbit types

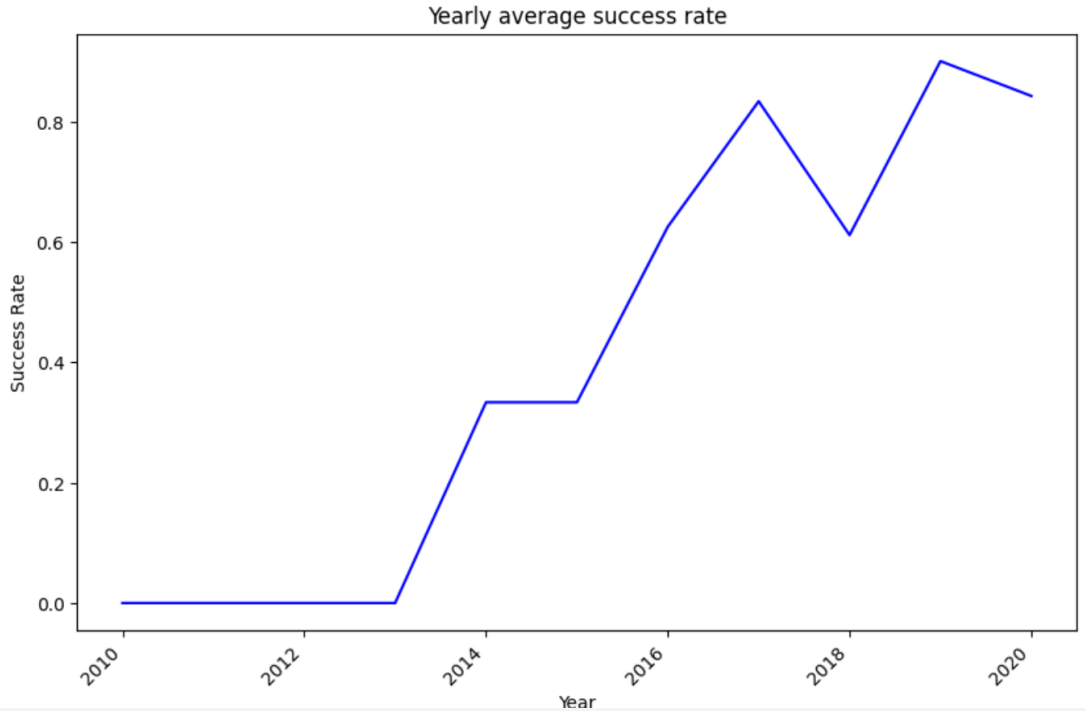
```
#Define parameters for plot
plt.figure(figsize=(10, 6))  # Set the figure size
plt.scatter(df['PayloadMass'], df['Orbit'], alpha=0.6, edgecolors="w", linewidth=0.5)

# Add labels and title
plt.xlabel('PayloadMass')
plt.ylabel('Orbit Type')
plt.title('Scatter Plot: PayloadMass vs. Orbit Type')

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



Launch Success Yearly Trend

- YoY average success rate increases

```
df['Year'] = pd.to_datetime(df['Date']).dt.year
#success_rate
yearly_success_rate = df.groupby('Year')["Class"].mean()
# Plot the bar chart
plt.figure(figsize=(10, 6))
yearly_success_rate.plot(kind='line', color='blue')

# Add labels and title
plt.xlabel('Year')
plt.ylabel('Success Rate')
plt.title('Yearly average success rate')

# Show the plot
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for readability
plt.show()
```



All Launch Site Names

- Find the names of the unique launch sites

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

```
%sql select Launch_Site from SPACEXTBL group by 1
```

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

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

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`

```
%sql select Launch_Site from SPACEXTBL where Launch_Site like 'CCA%' COLLATE NOCASE limit 5
```

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

Launch_Site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

Total Payload Mass

- Calculate the total payload carried by boosters from NASA
- Present your query result with a short explanation here

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

```
: %sql select sum(PAYLOAD_MASS__KG_) from SPACEXTBL where customer = 'NASA (CRS)'
 * sqlite:///my_data1.db
Done.
: sum(PAYLOAD_MASS__KG_)
45596
```

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1

```
: %sql select avg(PAYLOAD_MASS__KG_) from SPACEXTBL where booster_version  = 'F9 v1.1'
 * sqlite:///my_data1.db
Done.
: avg(PAYLOAD_MASS__KG_)
2928.4
```

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad

```
%sql select min(Date) from SPACEXTBL where Landing_Outcome = 'Success (ground pad)'
```

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

min(Date)
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

```
[33]: %sql select Booster_Version from SPACEXTBL where Mission_Outcome = 'Success (drone ship)' and PAYLOAD_MASS__KG_ >4000 and PAYLOAD_MASS__KG_<6000
 * sqlite:///my_data1.db
Done.
[33]: Booster_Version
```

Total Number of Successful and Failure Mission Outcomes

```
[35]: %sql select Mission_Outcome, Count(*) from SPACEXTBL group by 1
```

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

[35]:

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

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass

```
[45]: %sql select Booster_Version from SPACEXTBL where PAYLOAD_MASS__KG_ = (Select max(PAYLOAD_MASS__KG_) as max1 from SPACEXTBL)
 * sqlite:///my_data1.db
Done.
```

[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

- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

```
%sql select Booster_Version, Launch_Site from SPACEXTBL where Landing_Outcome = 'Failure (drone ship)'  and substr(Date,0,5)='2015'
```

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

Booster_Version	Launch_Site
F9 v1.1 B1012	CCAFS LC-40
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

```
5]: %sql select Landing_Outcome, count(*) from SPACEXTBL where Landing_Outcome in ('Failure (drone ship)',  'Success (ground pad)') and date>='2010-06-04' and date <='20

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

5]:

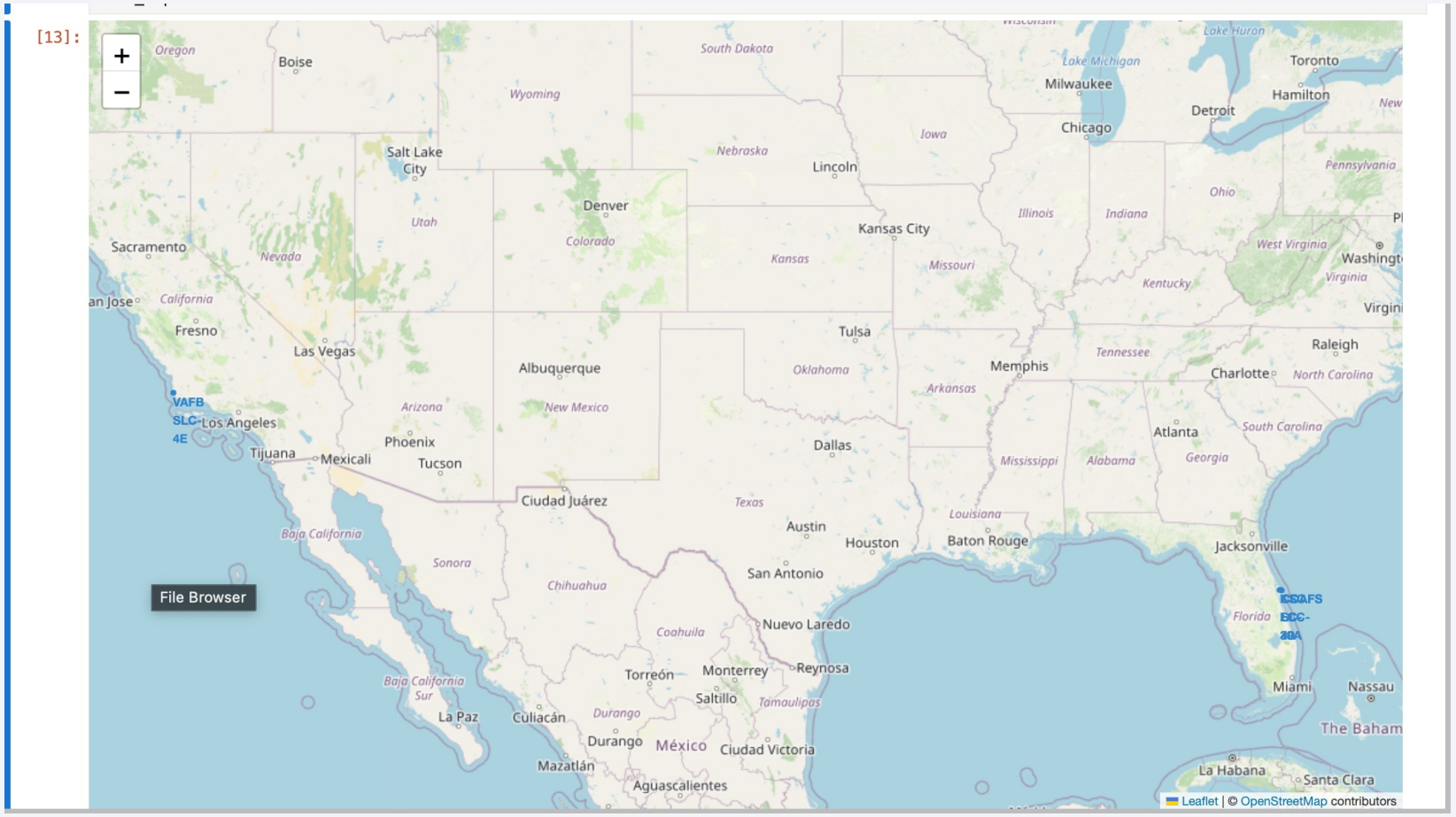
Landing_Outcome	count(*)
Failure (drone ship)	5
Success (ground pad)	3


Section 3
Launch Sites
Proximities Analysis

Launch sites on a map

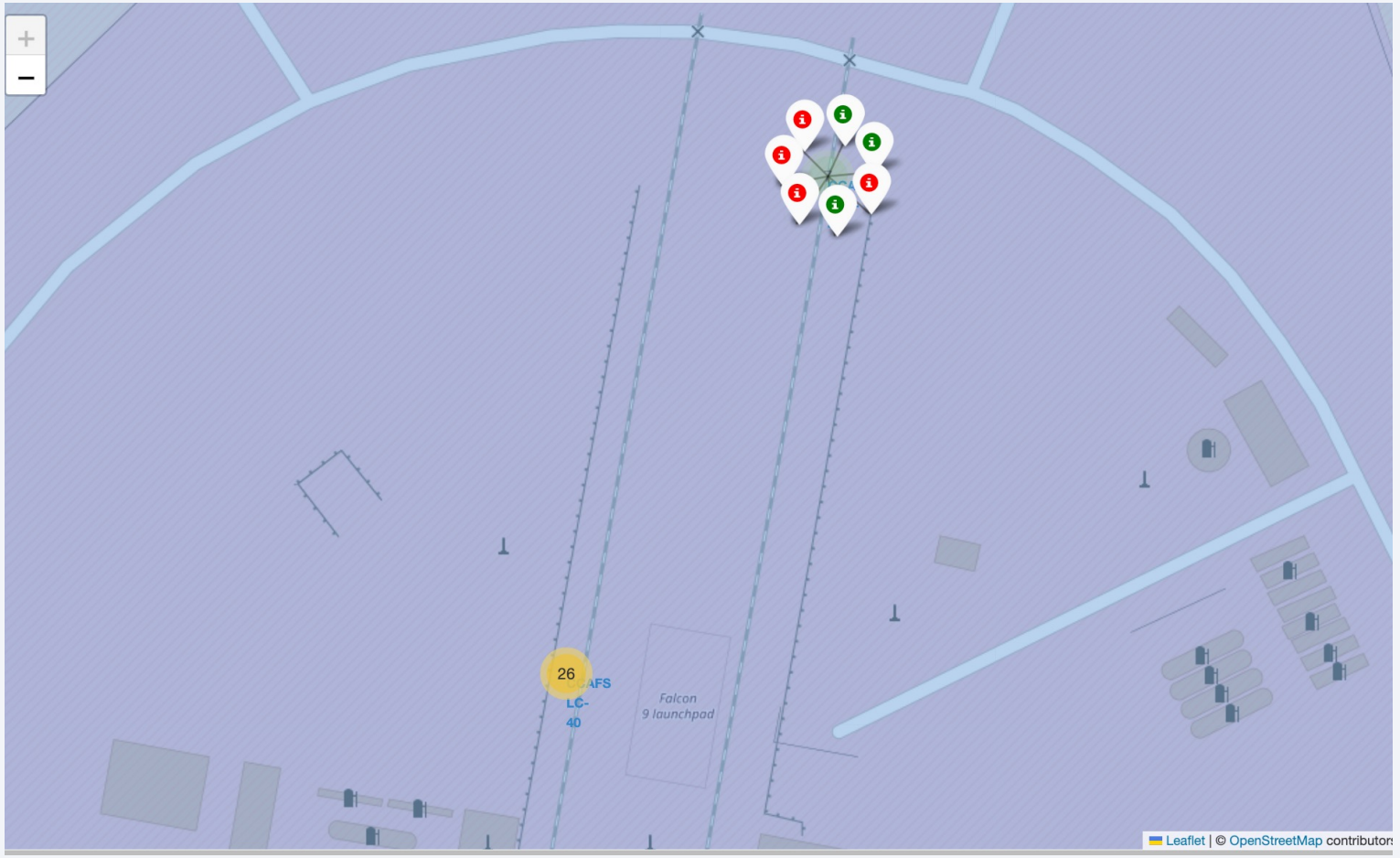
All are very close to the ocean and are located closer to the Equator line.

Reasons for this:
The Earth rotates fastest at the Equator. Rockets launched here gain a "boost" from this rotational velocity, reducing the amount of fuel needed to reach orbit.

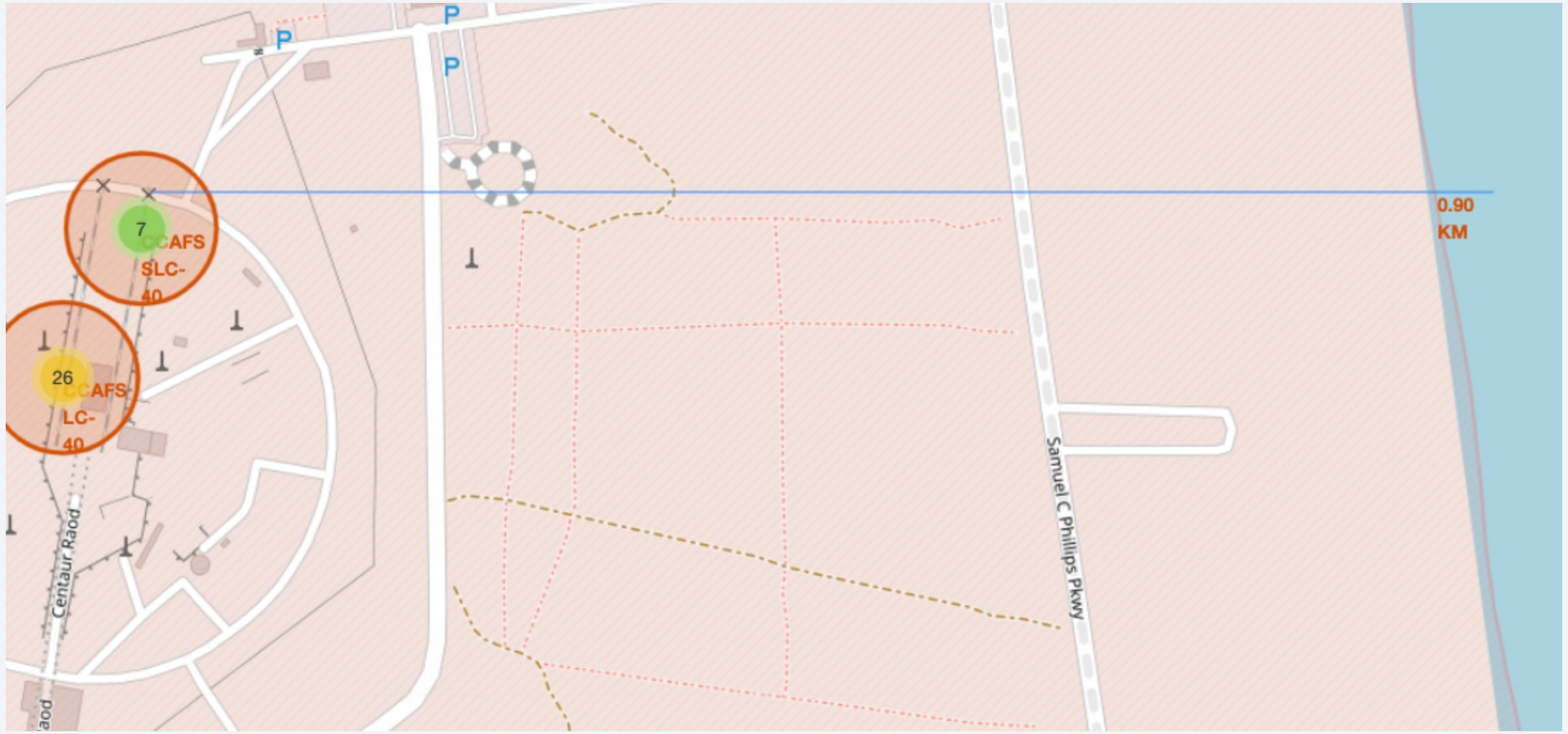


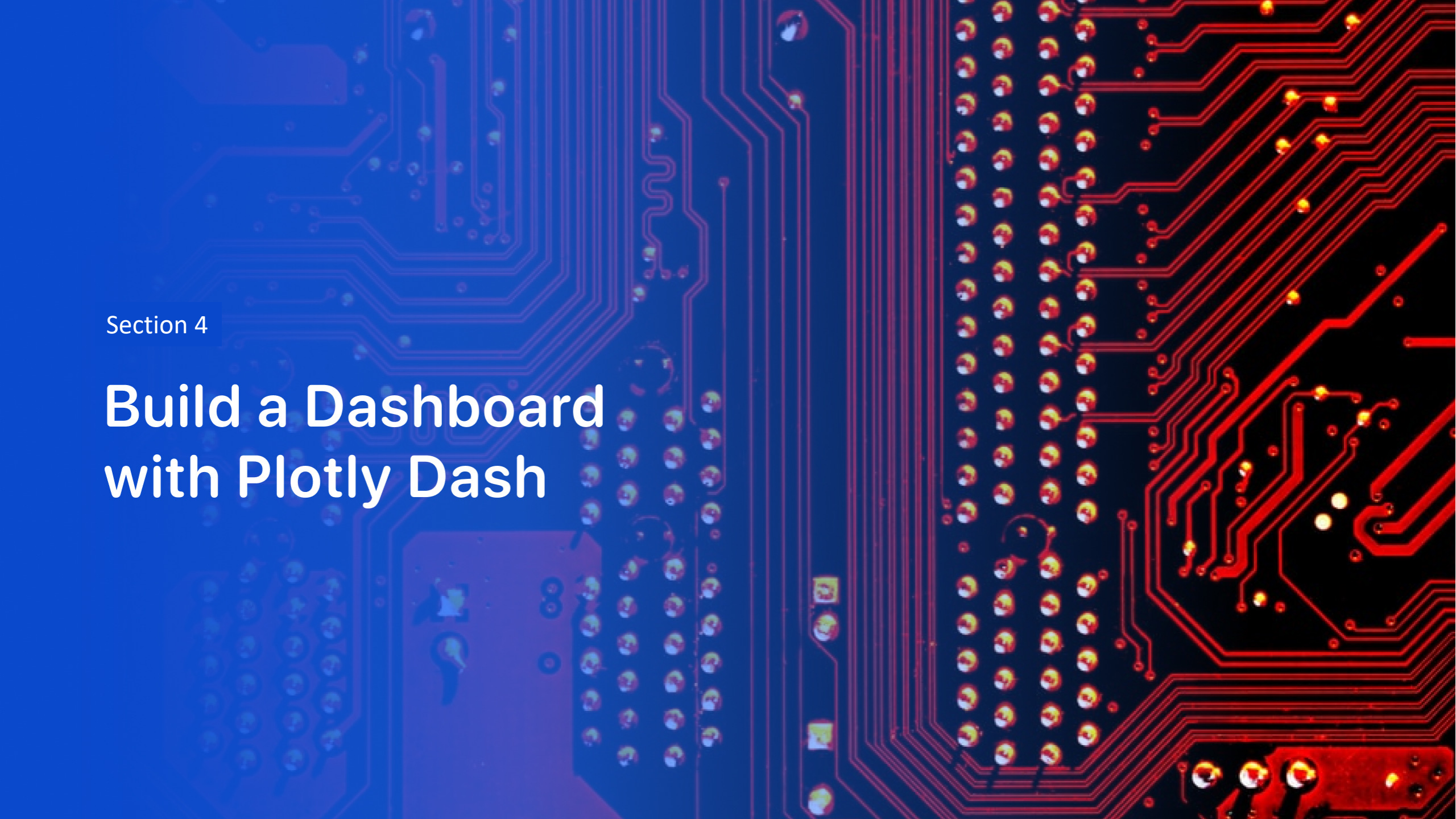
Clusters all launch records

At some places more launches were successful then on other



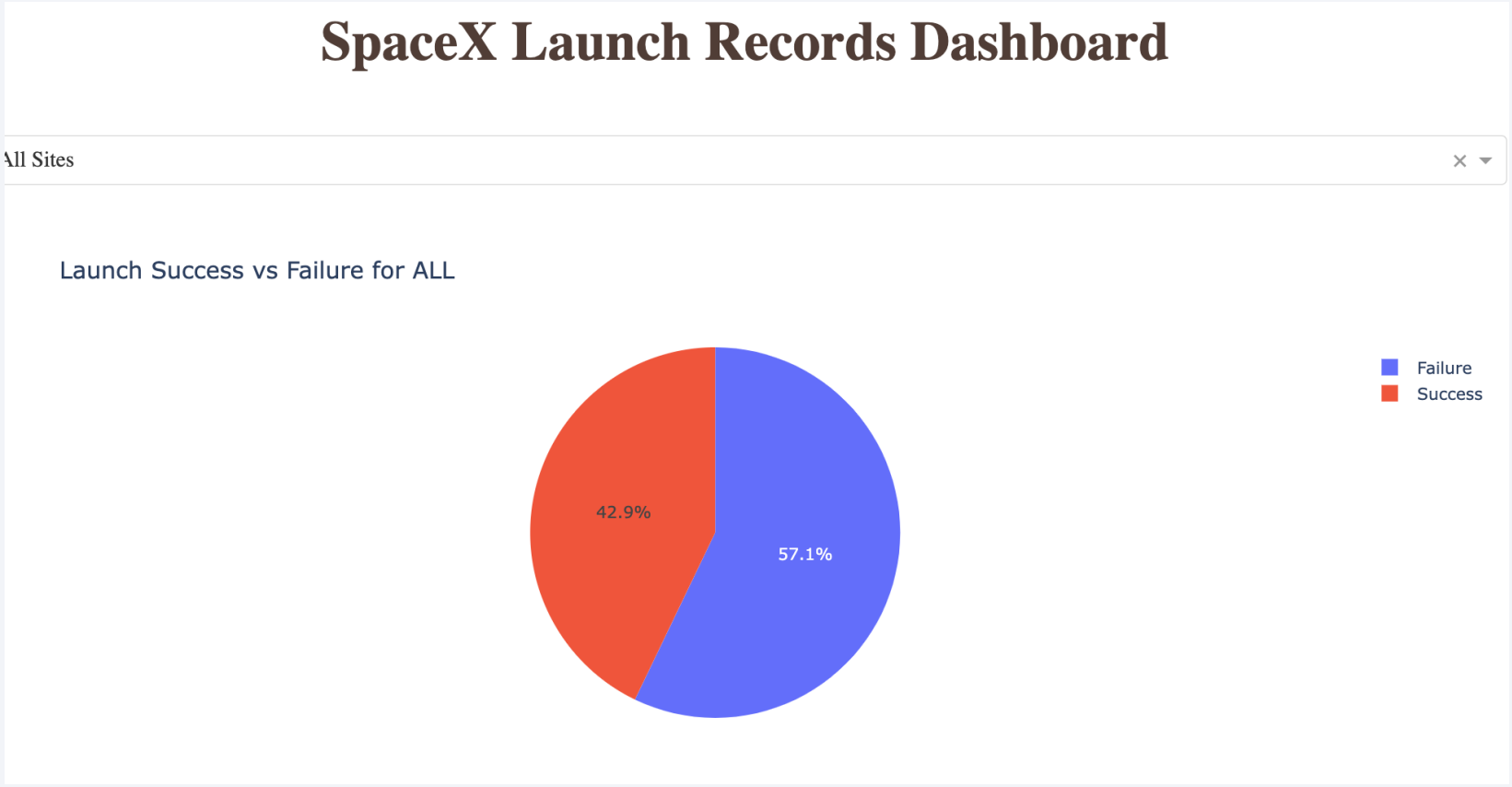
Distance to nearest coastline



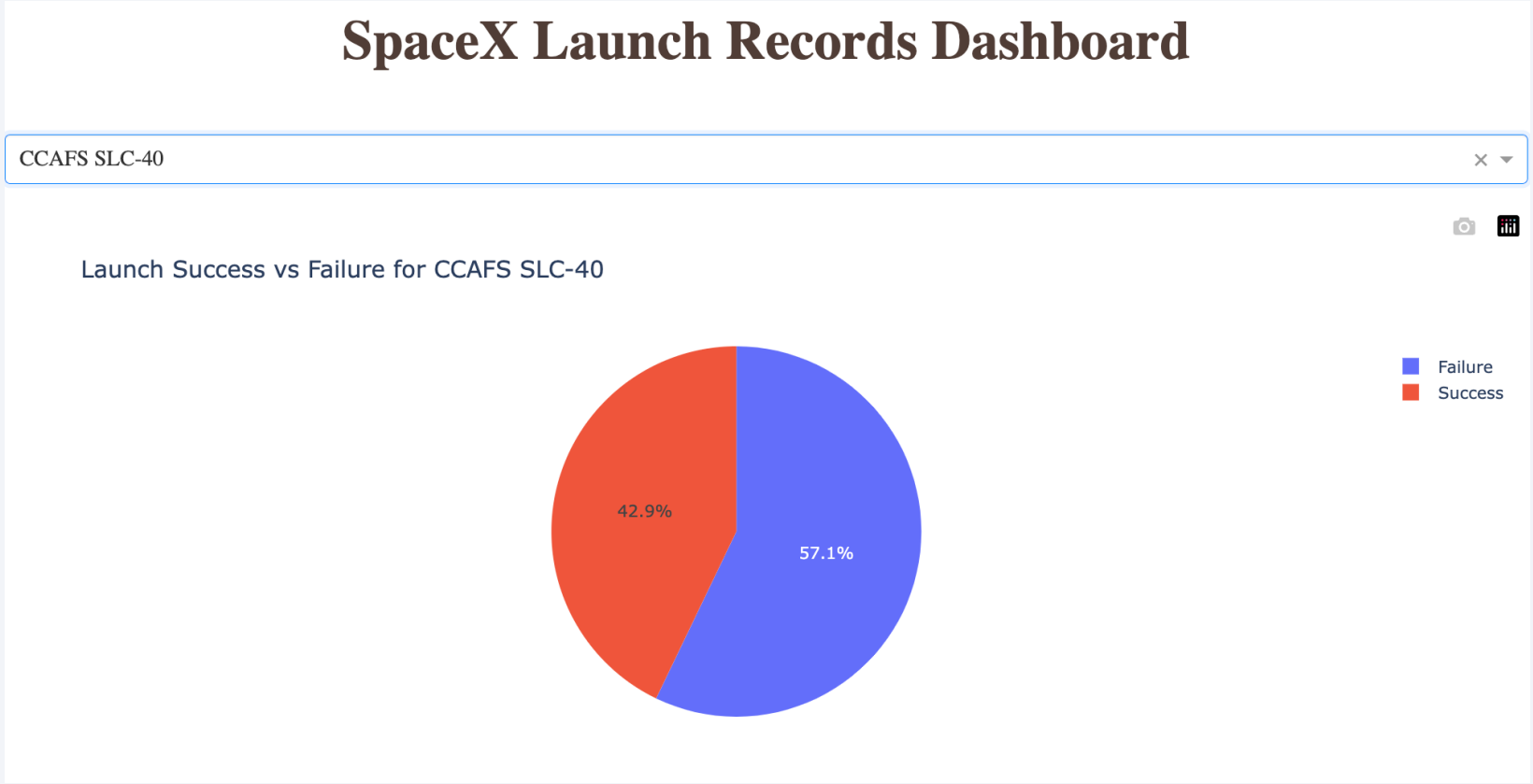

Section 4
Build a Dashboard with Plotly Dash

Launch Success & Failure for All

- Success rate in total is only 42.9%



Highest Launch Success



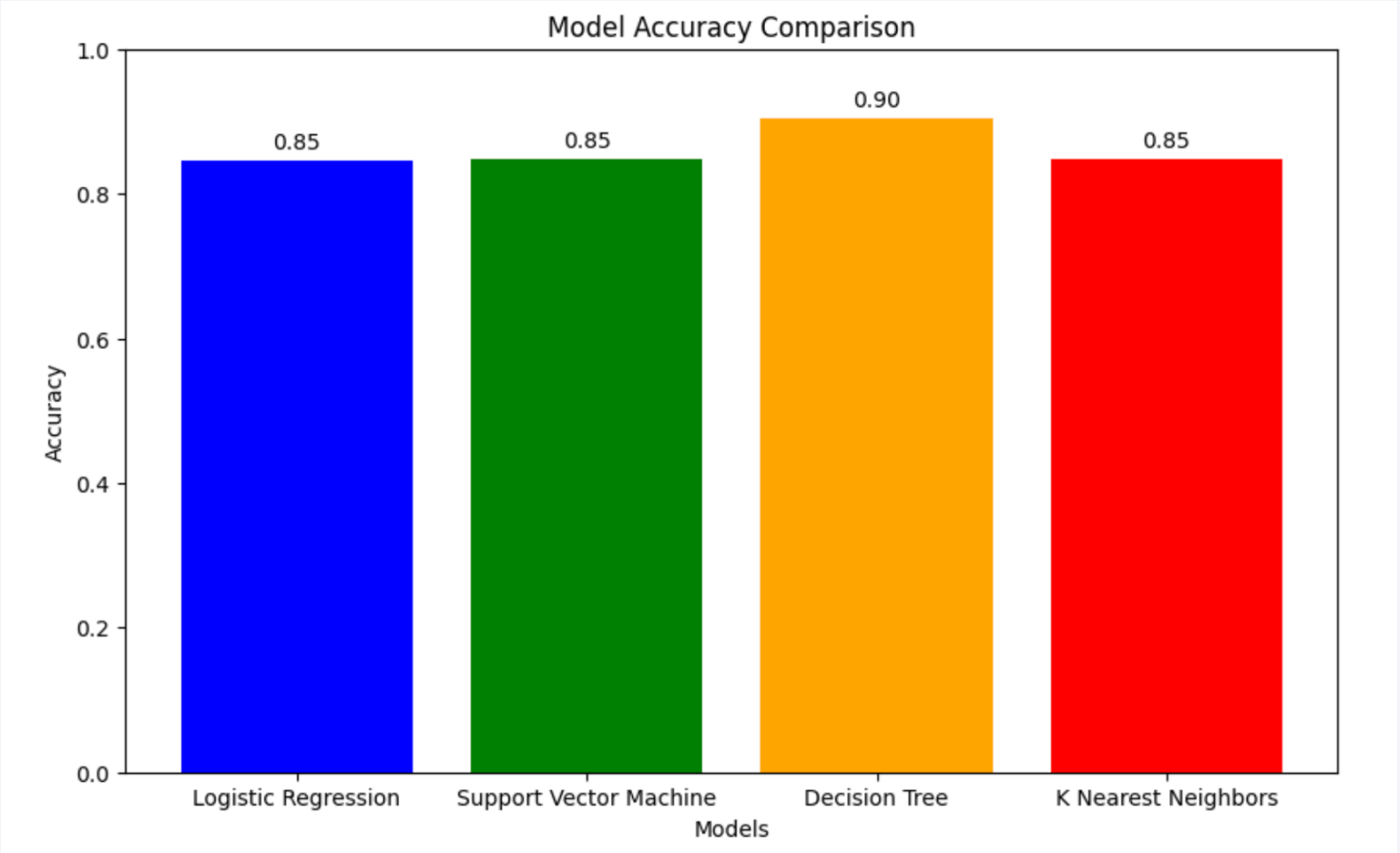
Scatter plot for all launch sites

Section 5

Predictive Analysis (Classification)

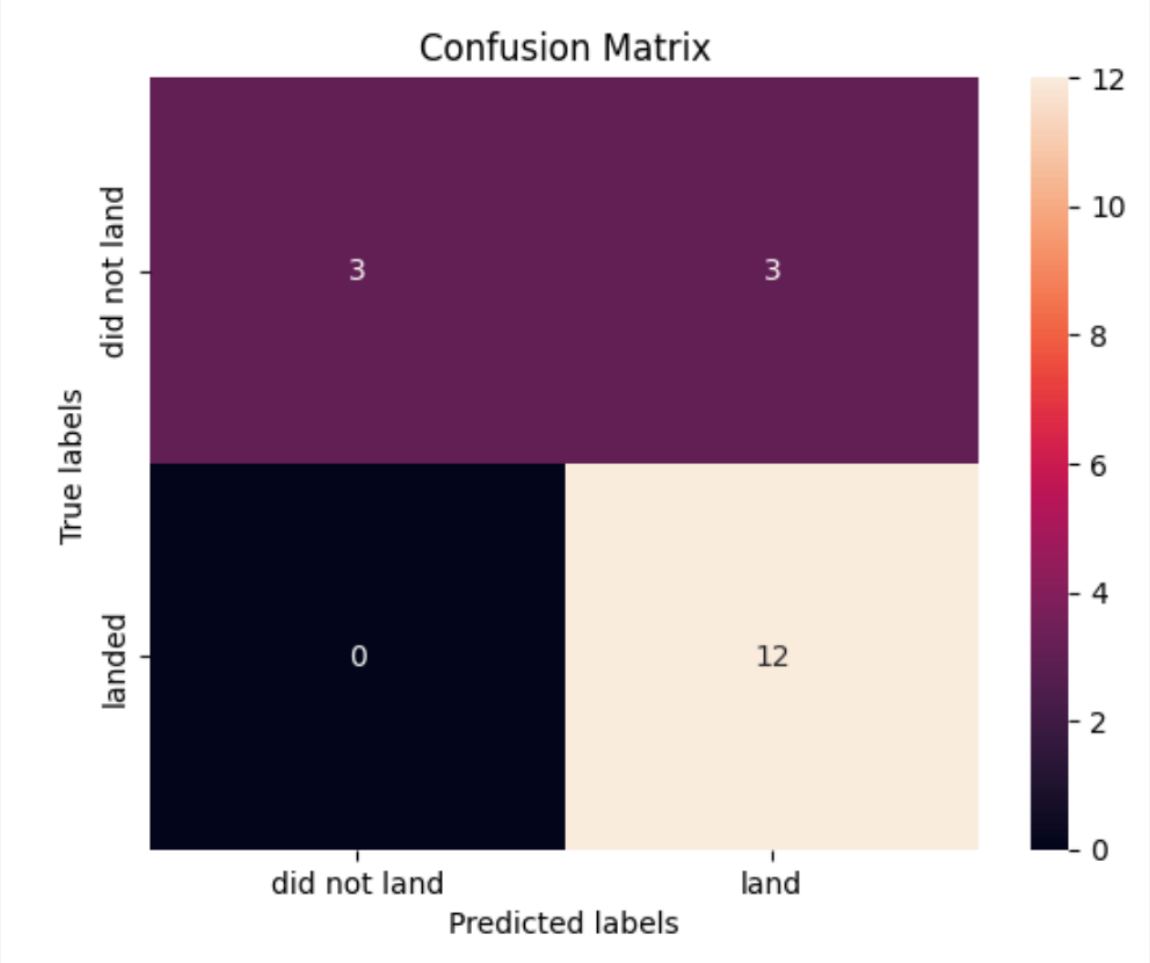
Classification Accuracy

- Decision Tree has the best score accuracy



Confusion Matrix

- Model define correctly all of the landed
- 3 were false predicted to be landed while they were no



Conclusions

- Various packages helps a lot in researching on data. From SpaceX dataset was discovered next:
- CCAFS SLC 40 had the most number of flights, used for smaller payload mass
- Distribution of Payload Mass are not equal over Orbit types
- YoY average success rate increases
- Folium helps a lot with clustering and visualization data on map
- Dash helps with interactive charts building
- Among the predictive models the best performance had decision tree

Thank you!