IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Martin Gael Negrete Rangel>
<12-01-2025>

Outline

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

Executive Summary

- Summary of methodologies
  - Data collection
  - Data Wrangling Exploratory
  - Data Analysis (EDA)
  - EDA with Data Visualization
  - EDA with SQL
  - Building an interactive Map with Folium
  - Building a dashboard with Plotly Dash
  - Predictive Analysis (Classification)
- Summary of all results
  - Inter Exploratory Data Analysis result
  - Active analytics in screenshots
  - Predictive Analytics result

Introduction

- Project background and context

  SpaceX markets Falcon 9 rocket launches on its website at a cost of $62 million, significantly less than the $165 million charged by other providers. A large portion of this cost reduction is attributed to SpaceX's ability to reuse the rocket's first stage. Consequently, by determining whether the first stage will successfully land, the cost of a launch can also be estimated. This insight could be valuable for competing companies aiming to bid against SpaceX for rocket launches. The objective of this project is to develop a machine learning pipeline capable of predicting whether the first stage will land successfully..

- Problems you want to find answers

  What are the key factors influencing the rocket's ability to land successfully?

  How do different variables interact to impact the likelihood of a successful landing?

  What operational conditions are required to guarantee the success of a rocket landing program?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:

  SpaceX Rest API· (Web Scrapping) from Wikipedia

  SpaceX was obtained from 2 sources

- Perform data wrangling

  One Hot Encoding data fields for Machine Learning and dropping irrelevant columns

- Perform exploratory data analysis (EDA) using visualization and SQL

  Visualizations: Create scatter plots and bar graphs to illustrate relationships between variables and identify patterns within the data.

- Perform interactive visual analytics using Folium and Plotly Dash

- Perform predictive analysis using classification models

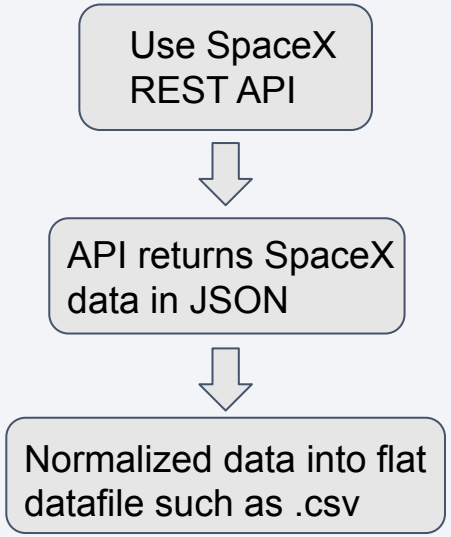
  How to build, tune, evaluate classification models

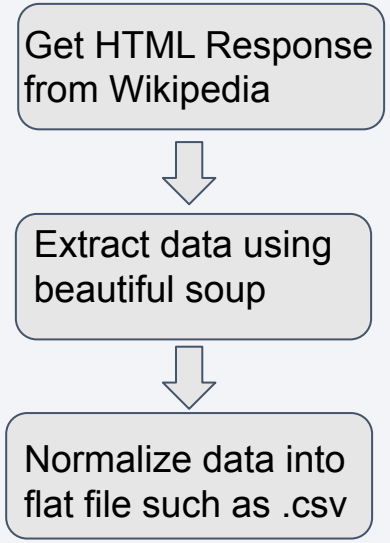
Data Collection

- Describe how data sets were collected.

  The dataset used in this project was sourced from SpaceX launch data obtained via the SpaceX REST API.This API provides comprehensive details about launches, including information on the rockets used, payloads delivered, launch parameters, landing conditions, and the outcomes of landings.The primary goal of the analysis is to predict whether SpaceX will attempt to land a rocket or not using the provided data.The SpaceX REST API can be accessed through endpoints that begin with api.spacexdata.com/v4/.Another commonly used method to acquire Falcon 9 launch data is by web scraping Wikipedia using the BeautifulSoup library.

- You need to present your data collection process use key phrases and flowcharts





Data Collection – SpaceX API

- Retrieve data from the html

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

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

- Converting response to a .json file

```
[15]: # Use json_normalize meethod to convert the json result into a dataframe
      response = requests.get(static_json_url).json()
      data = pd.json_normalize(response)
```

- Apply custom functions to preprocess the data

Now, let's apply `getBoosterVersion` function method to get the booster version

```
[20]: # Call getBoosterVersion
      getBoosterVersion(data)
```

the list has now been update

```
[21]: BoosterVersion[0:5]
```

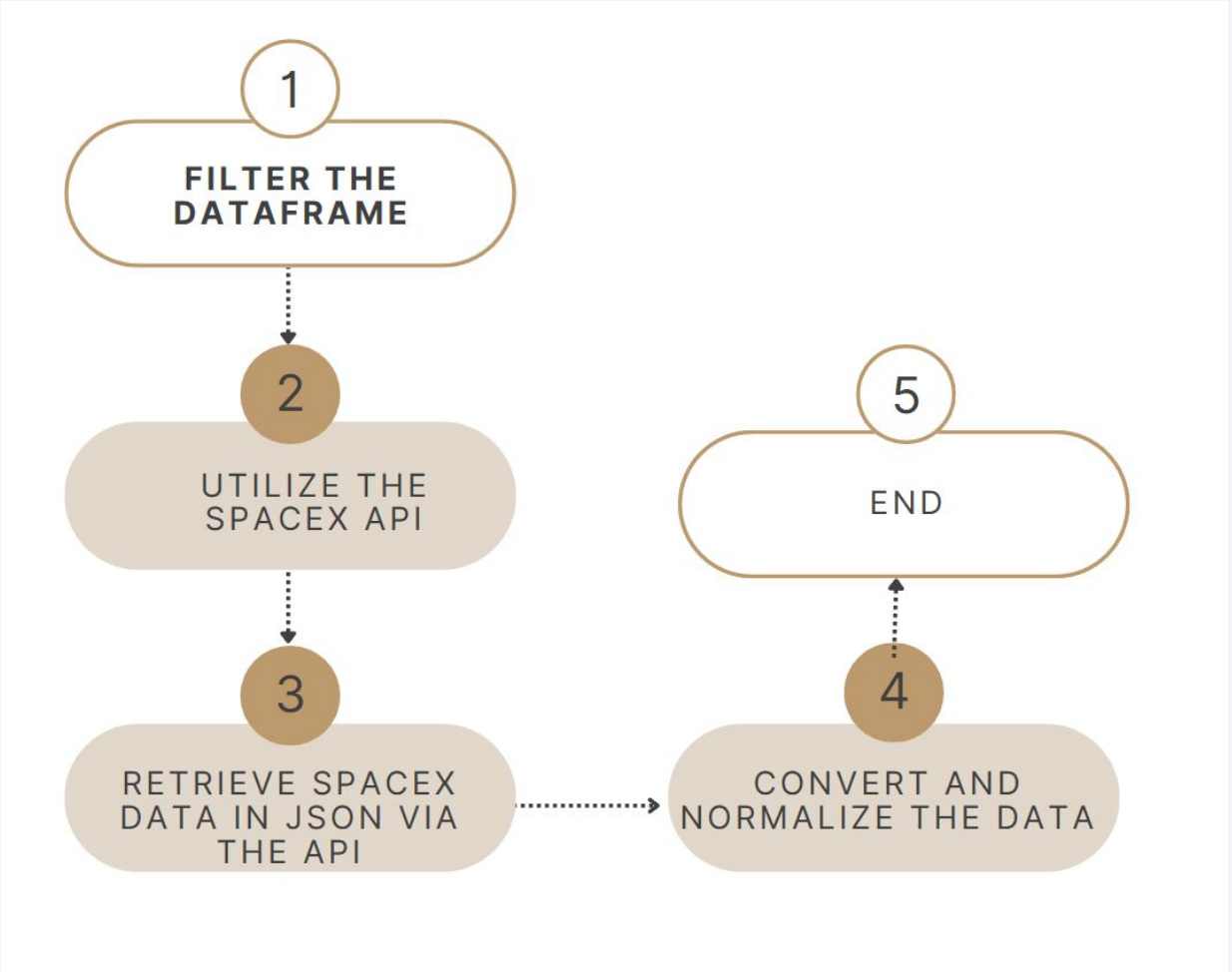
```
[22]: # Call getLaunchSite
      getLaunchSite(data)

[23]: # Call getPayLoadData
      getPayloadData(data)

[24]: # Call getCoreData
      getCoreData(data)
```

- Map lists to a dictionary and convert it into a DataFrame

```
[25]: 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}
```

- Filter the DataFrame and export it as a flat file .csv

```
[33]: data_falcon9.to_csv('dataset_part_1.csv', index=False)
```



GITHUB

Data Collection - Scraping

- Retrieve data from the html

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

- Initialize a BeautifulSoup object to parse the HTML content.

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

- Identify and extract tables from the HTML document

```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table', {'class': 'wikitable'})
```

- Extract column names from the tables.

```
column_names = []

# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names
def extract_column_from_header(th):
    return ' '.join(th.stripped_strings) if th else None

for th in first_launch_table.find_all('th'):
    name = extract_column_from_header(th)
    if name:
        column_names.append(name)
```

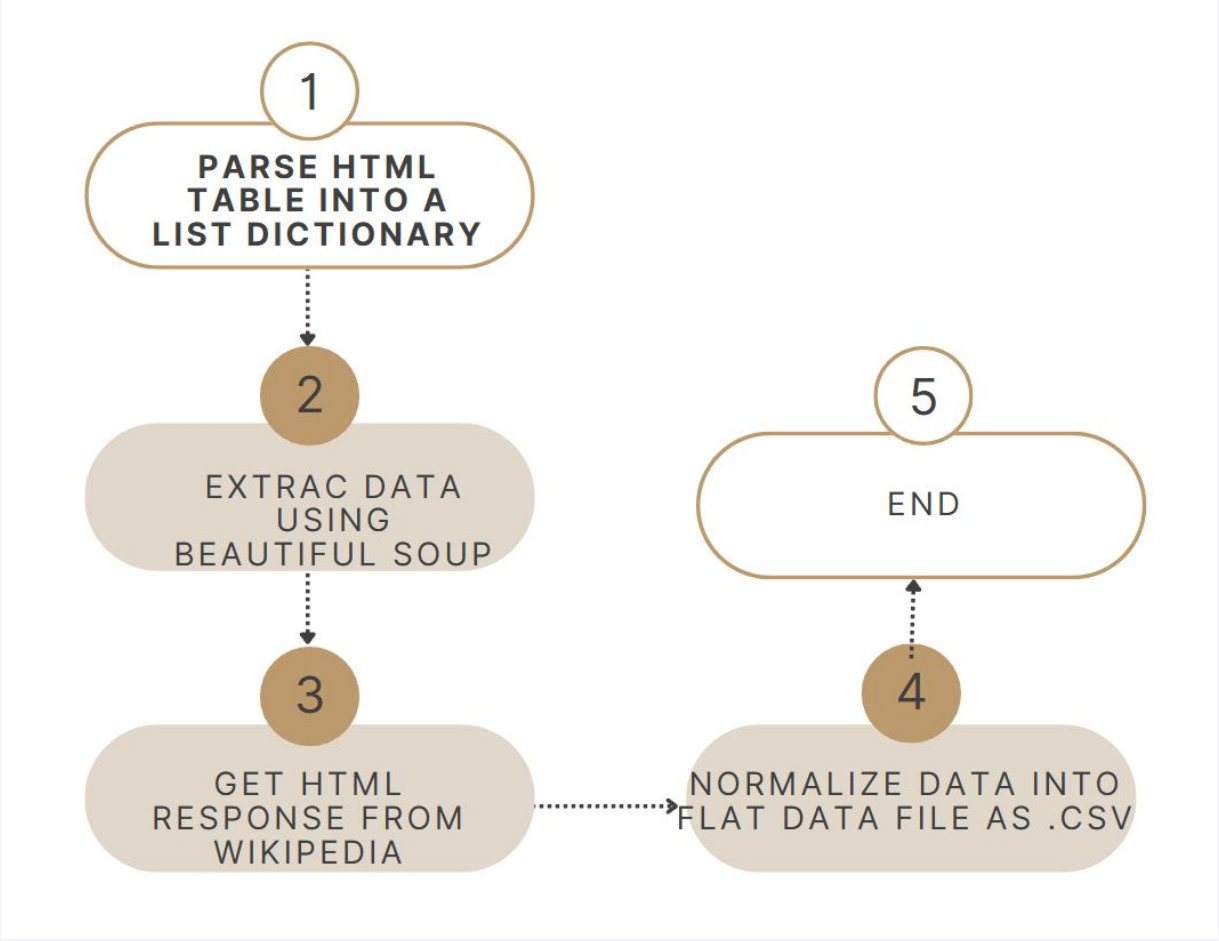
```
launch_dict= dict.fromkeys(column_names, [])

# Remove an irrelvant column
date_time_key = 'Date and time ( )'
if date_time_key in launch_dict:
    del launch_dict[date_time_key]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

- Append additional data to dictionary keys

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
    # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            #print(flight_number)
            datatimelist=date_time(row[0])
```

- Convert the dictionary into a DataFrame and export the dataframe as .csv file

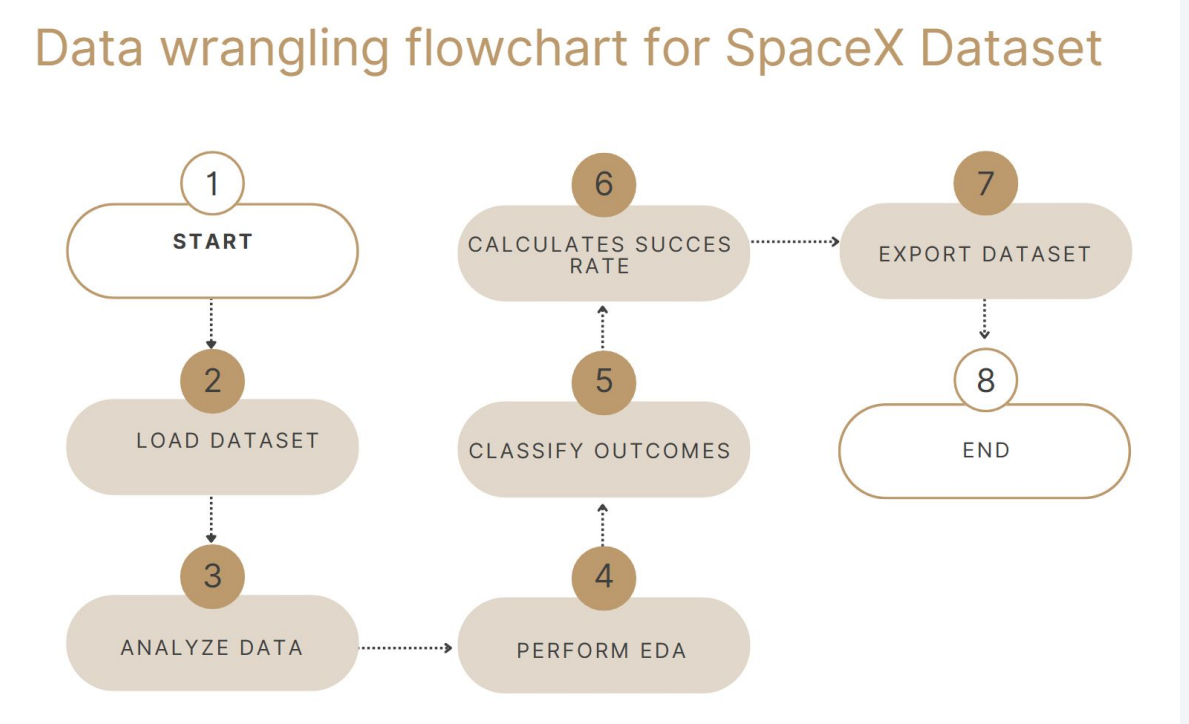
```
df = pd.DataFrame({key: pd.Series(value) for key, value in launch_dict.items()})
df.to_csv('spacex_web_scraped.csv', index=False)
print("DataFrame created and exported to 'spacex_web_scraped.csv'")
```



GITHUB

Data Wrangling

This dataset includes various scenarios where the booster failed to land successfully. In some cases, landing attempts were made but resulted in failure due to accidents. For instance, True Ocean indicates that the mission successfully landed in a designated ocean area, whereas False Ocean signifies that the landing attempt in that ocean region was unsuccessful. Similarly, True RTLS (Return to Launch Site) refers to a successful landing on a ground pad, while False RTLS indicates that the landing on the ground pad failed. In the same way, True ASDS (Autonomous Spaceport Drone Ship) represents a successful landing on a drone ship, while False ASDS denotes an unsuccessful attempt to land on the drone ship.b The main goal is to transform these outcomes into training labels for supervised machine learning. A label of 1 will indicate a successful booster landing, while 0 will represent an unsuccessful landing. These labels will be used for further analysis and model development



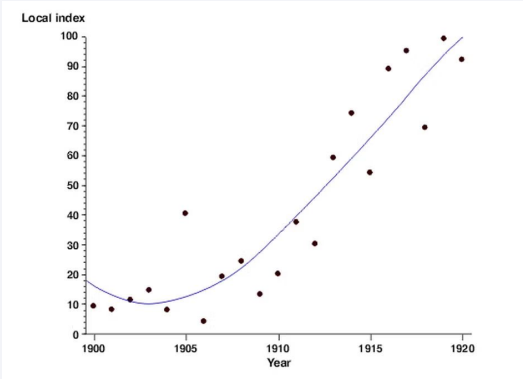
GITHUB

EDA with Data Visualization

Scatter Plots:

- Relationship between Flight Number and Payload Mass.
- Relationship between Flight Number and Launch Site.
- Relationship between Payload and Launch Site.
- Relationship between Orbit and Flight Number.
- Relationship between Payload and Orbit Type.
- Relationship between Orbit and Payload Mass.

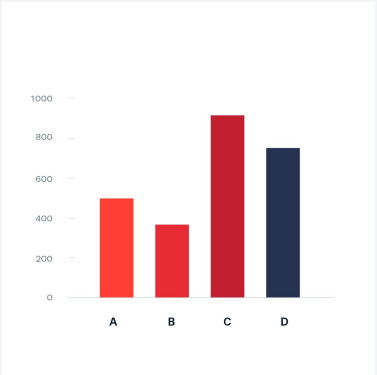
Scatter plots are effective tools to illustrate how one variable influences another, revealing their correlation. These plots are particularly useful for analyzing large datasets.



Bar Charts:

•Average values based on Orbit.

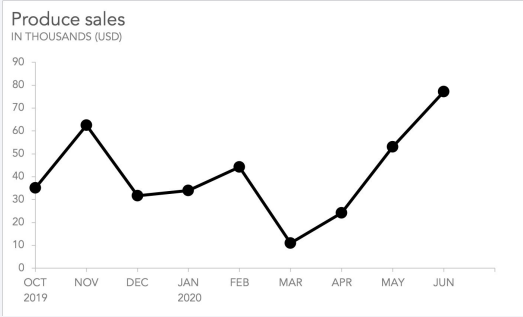
Bar charts provide a clear comparison of data across different categories. These visualizations represent categories on one axis and corresponding discrete values on the other, effectively showcasing relationships and significant changes in the data.



Line Graphs:

•Success Rate over Years.

Line graphs are ideal for presenting trends and variations in data over time. They are instrumental in predicting outcomes based on existing data patterns.



GITHUB

EDA with SQL

Data Exploration Using SQL

- SQL queries were utilized to analyze and extract meaningful insights from the dataset.
- To address specific questions regarding the data, the following queries were performed:
- Identifying the unique launch site names associated with space missions.
- Retrieving five records where the launch site name includes the substring "KSC."
- Calculating the total payload mass carried by boosters launched for NASA's CRS missions.
- Determining the average payload mass transported by boosters of type F9 v1.1.
- Finding the date when a successful landing outcome was achieved on a drone ship.
- Listing the names of boosters with successful ground pad landings and payload masses between 4000 and 6000.
- Calculating the total number of mission outcomes, distinguishing between successful and failed attempts.
- Identifying the booster versions that transported the maximum payload mass.
- Displaying records showing the month names, successful landing outcomes on ground pads, booster versions, and launch sites for the year 2017.
- Ranking the counts of successful landing outcomes between June 4, 2010, and March 20, 2017, in descending order.

GITHUB

Build an Interactive Map with Folium

- To visualize the launch data, an interactive map was created using Folium. The latitude and longitude coordinates of each launch site were used to place circle markers around the sites, each labeled with the corresponding launch site name.The dataset of launch outcomes (successes and failures) was categorized into two classes: 0 for failures and 1 for successes. These were represented on the map using red and green markers, respectively, within a MarkerCluster().The Haversine formula was applied to calculate the distances from each launch site to various landmarks. This allowed for the identification of trends and patterns related to the launch site locations. Lines were drawn on the map to represent these distances, providing additional insights. Some notable observations include:

Example of some trends in which the Launch Site is situated in..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

GITHUB

Build a Dashboard with Plotly Dash

- Hosted on PythonAnywhere for Live Access to the Website
- The dashboard is developed using the Flask and Dash web frameworks.
- Graphs
- Pie Chart showing the total launches by a specific site or across all sites

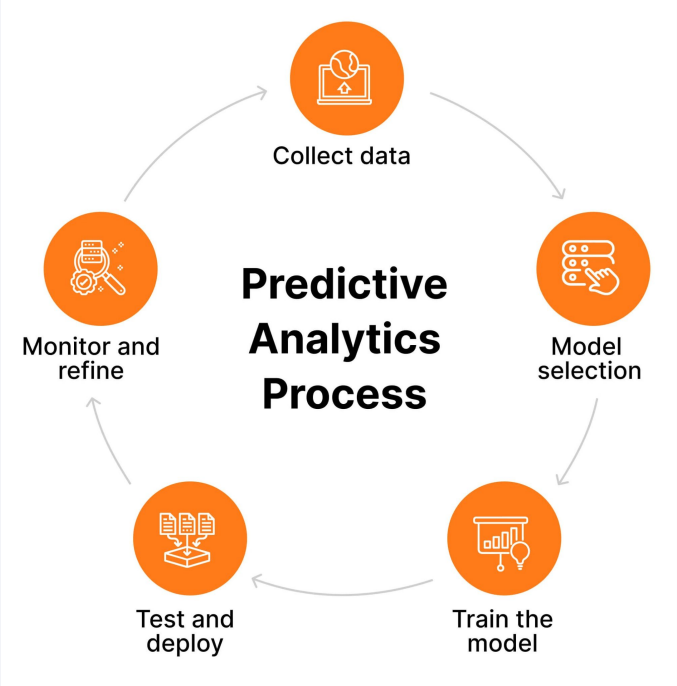
  Displays the relative proportions of different classes of data.

  The size of the circle can be adjusted proportionally to represent the total quantity.
- Scatter Graph illustrating the relationship between Outcome and Payload Mass (Kg) for various Booster Versions
- Represents the relationship between two variables.
- Ideal for identifying non-linear patterns in the data.
- Enables determination of the data range (minimum and maximum values).
- Straightforward to observe and interpret.

GITHUB

Predictive Analysis (Classification)

- Building the Model
- Load the dataset into NumPy and Pandas.
- Transform and prepare the data.
- Split the dataset into training and testing subsets.
- Determine the number of test samples available.
- Choose the machine learning algorithms to apply.
- Set parameters and algorithms using GridSearchCV.
- Train the dataset using GridSearchCV objects.
- Evaluating the Model
- Assess accuracy for each model.
- Obtain fine-tuned hyperparameters for each algorithm.
- Visualize results using a Confusion Matrix.
- Improving the Model
- Perform feature engineering.
- Optimize algorithms.
- Finding the Best Performing Classification Model
- The model with the highest accuracy score is considered the best.
- A dictionary of algorithms and their scores is included in the notebook for reference.



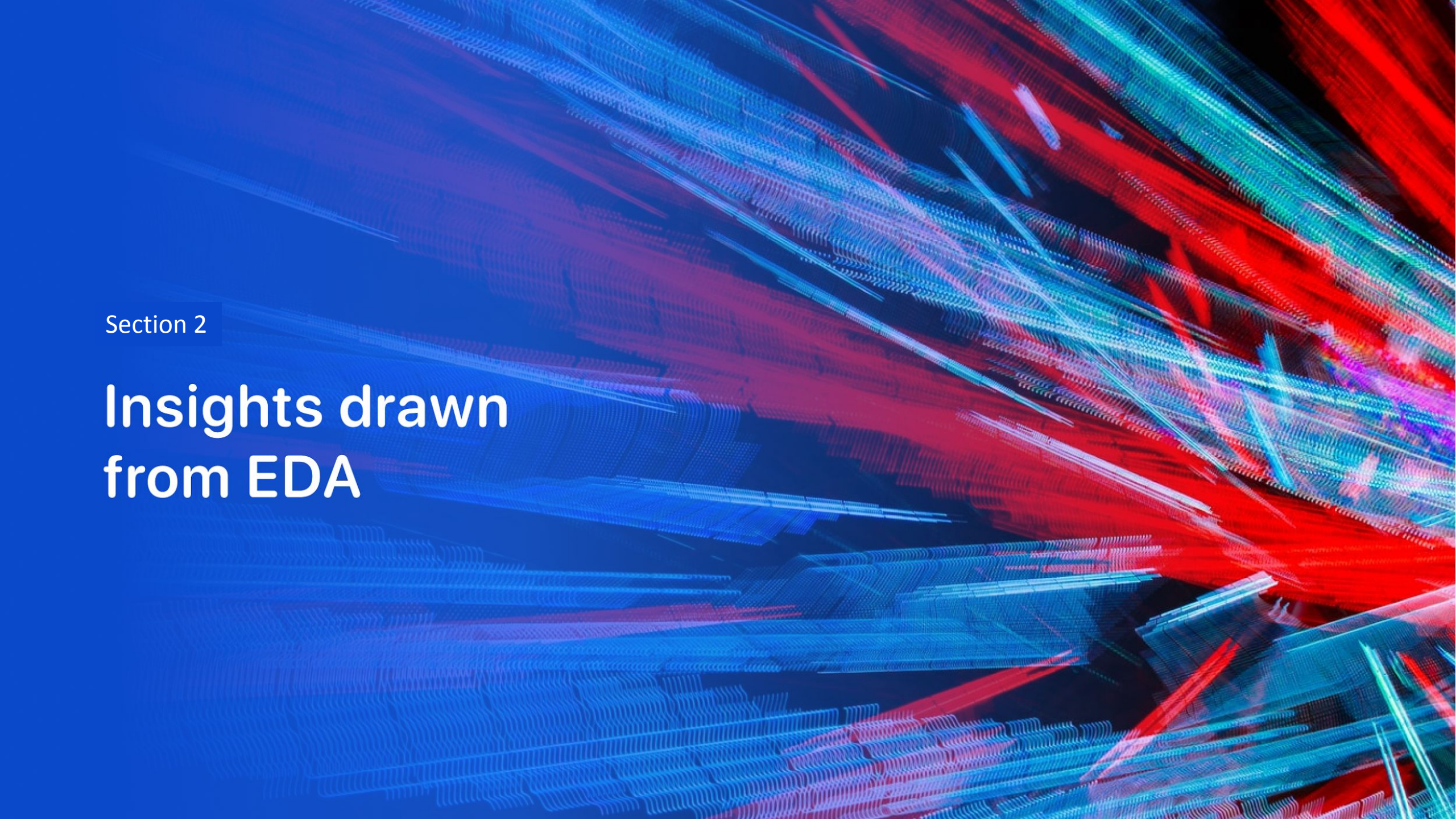
GITHUB

Results

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

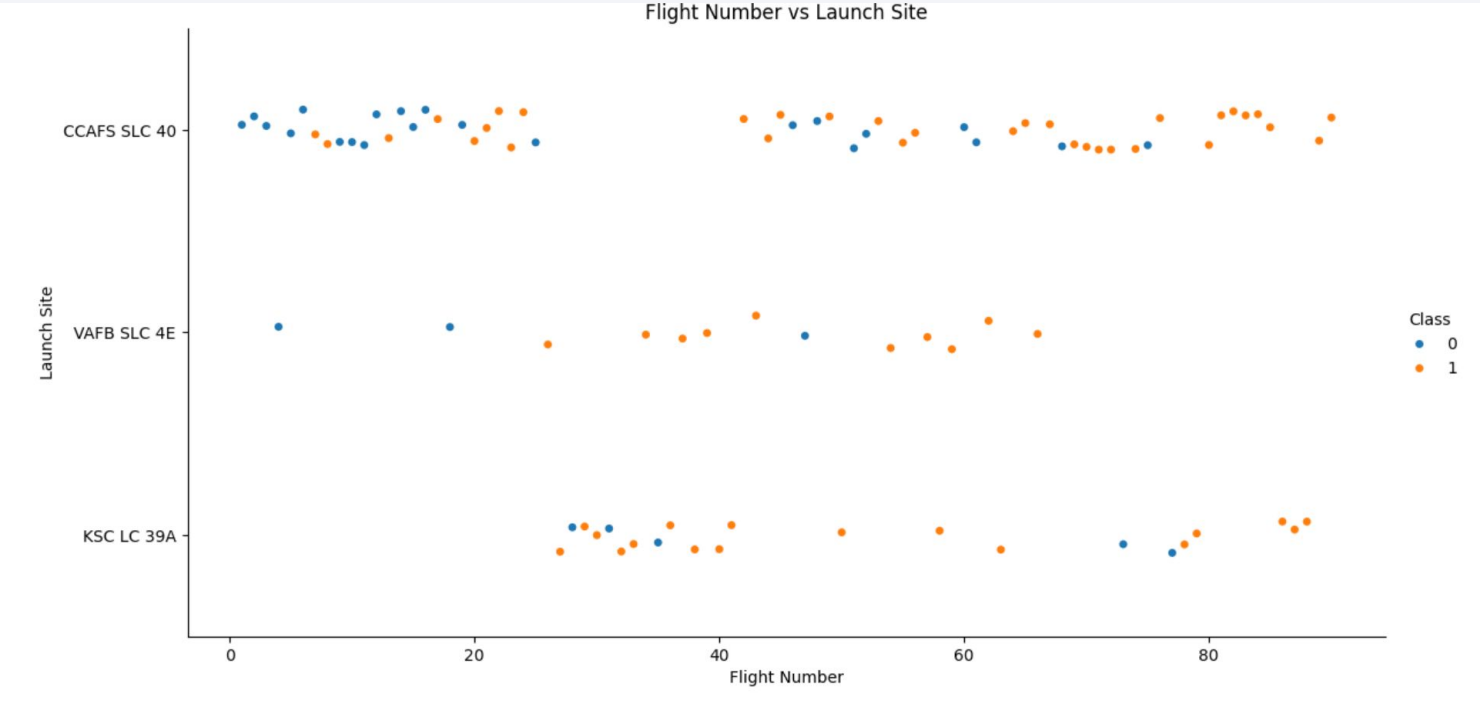
Section 2

Insights drawn from EDA



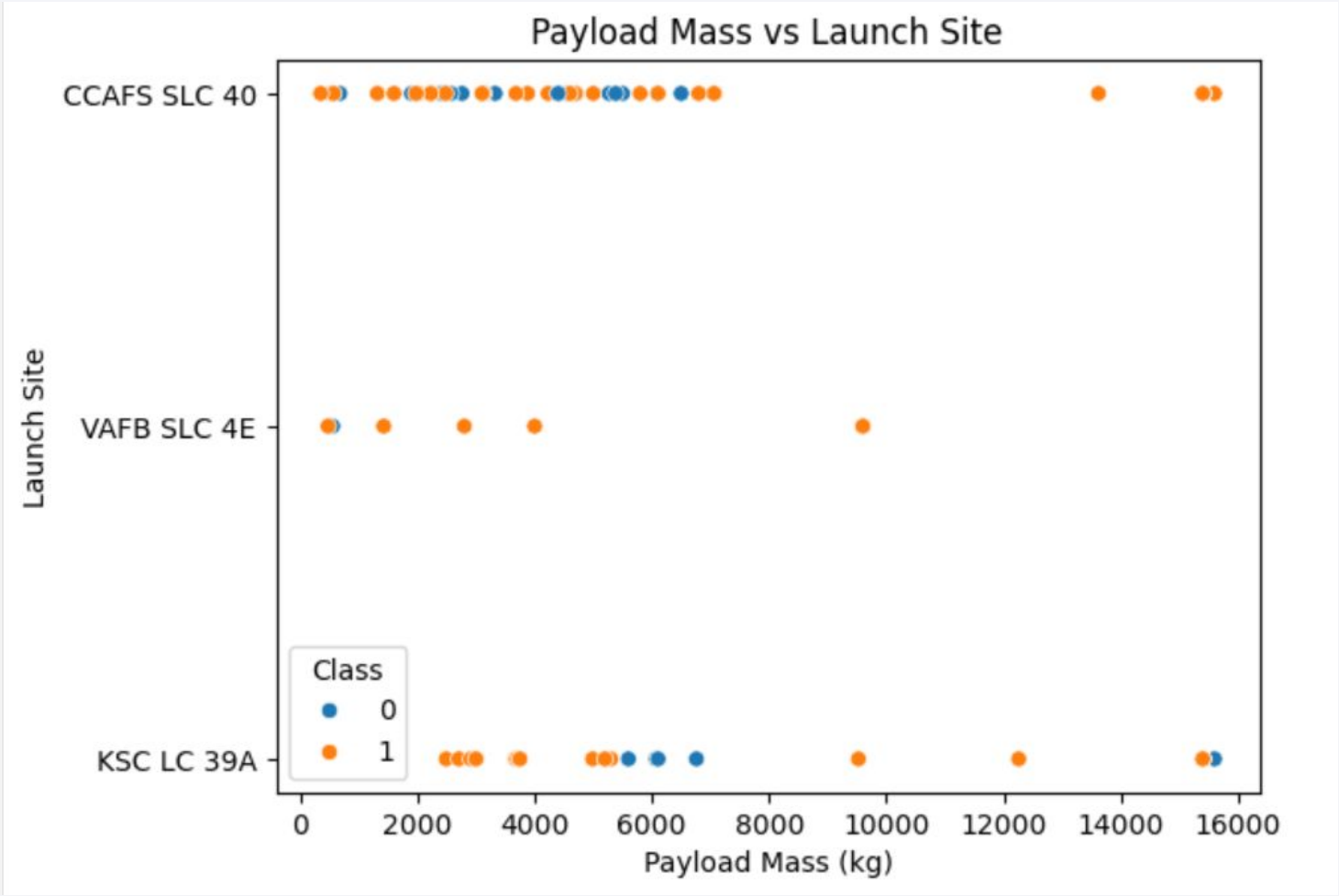
Flight Number vs. Launch Site

- The more amount of flights at a launch site the greater the succes rate at a launch site



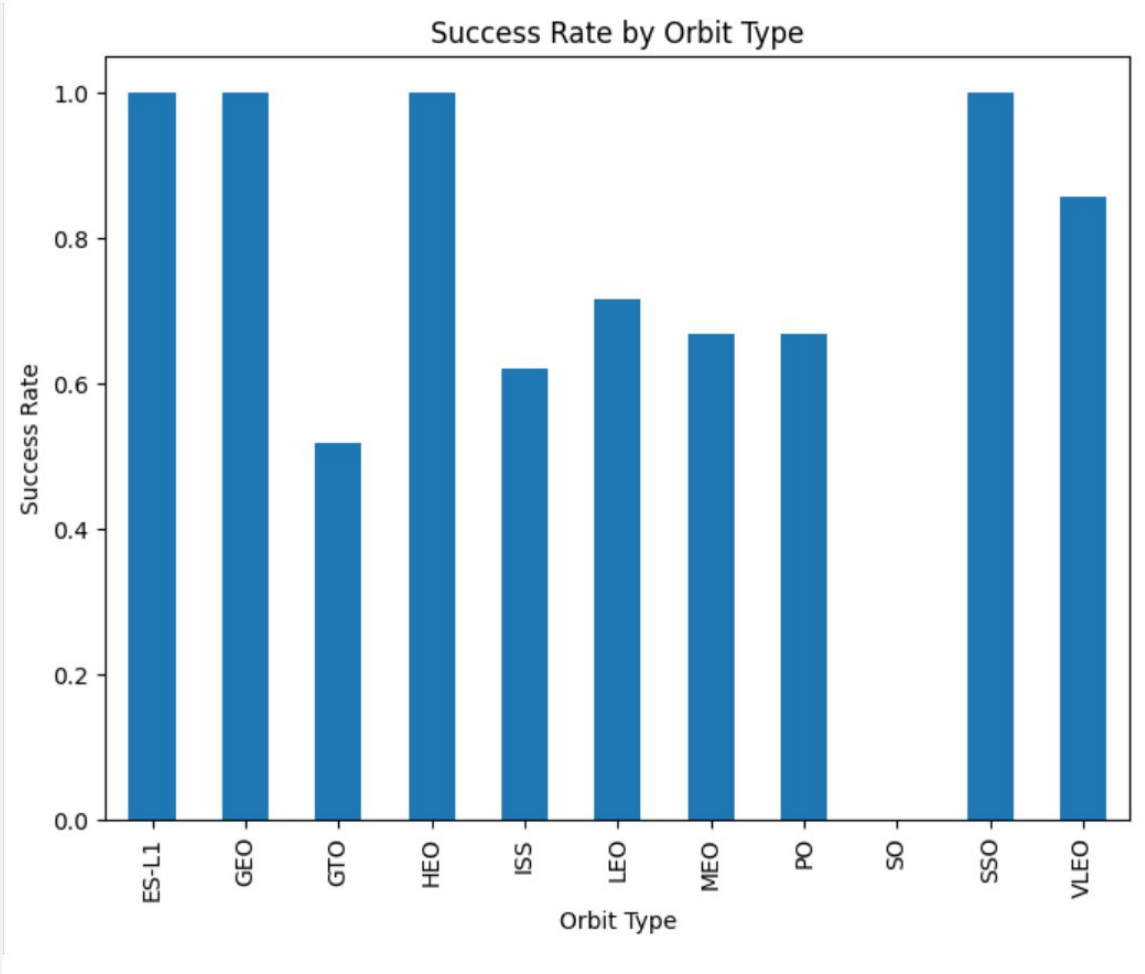
Payload vs. Launch Site

- For the VAFB-SLC launchsite there are no rockets launched for heavy payload mass, o greater than 10000



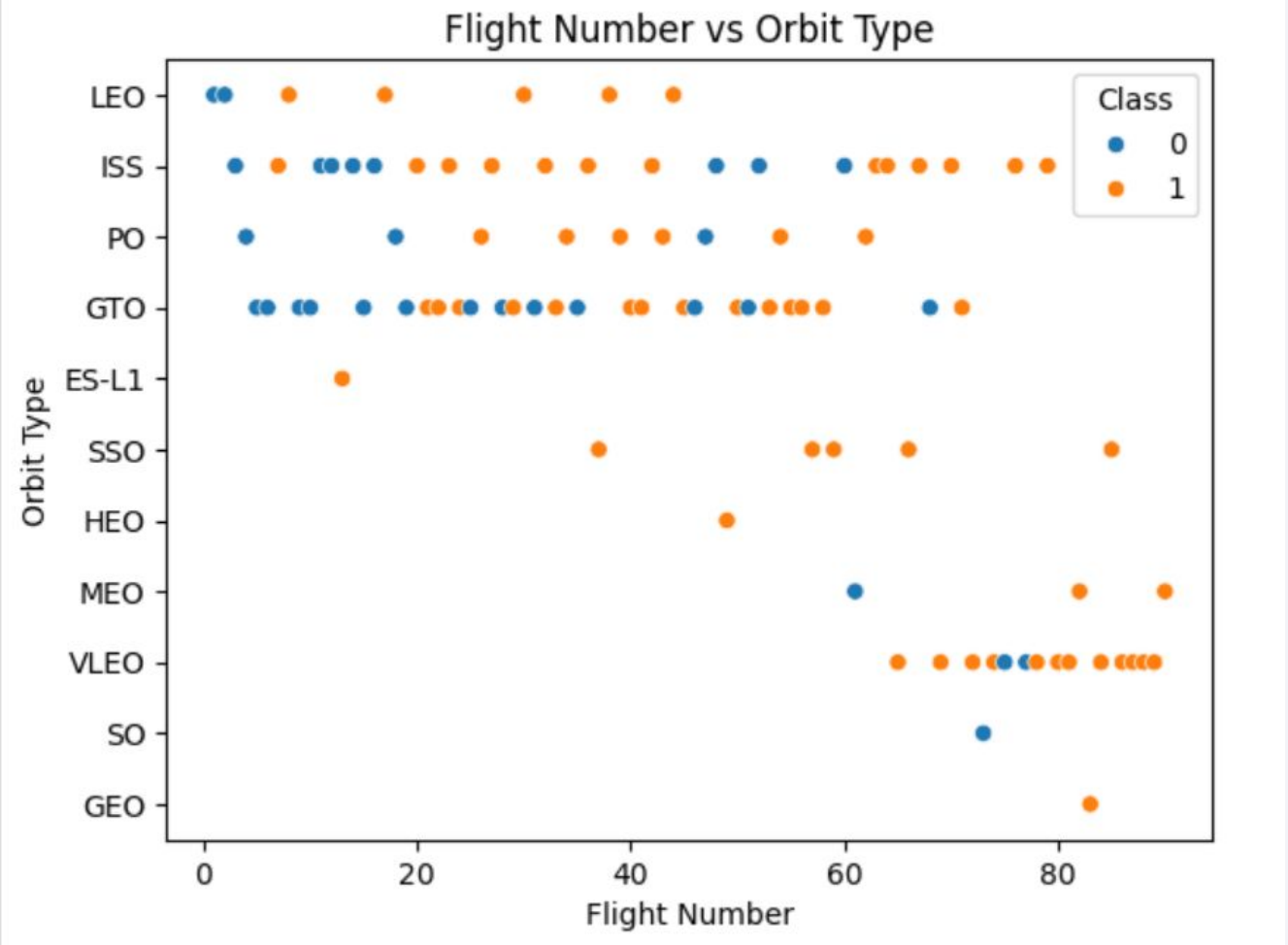
Success Rate vs. Orbit Type

- You can identified which orbits have the highest success rates when analyzes the bar charts



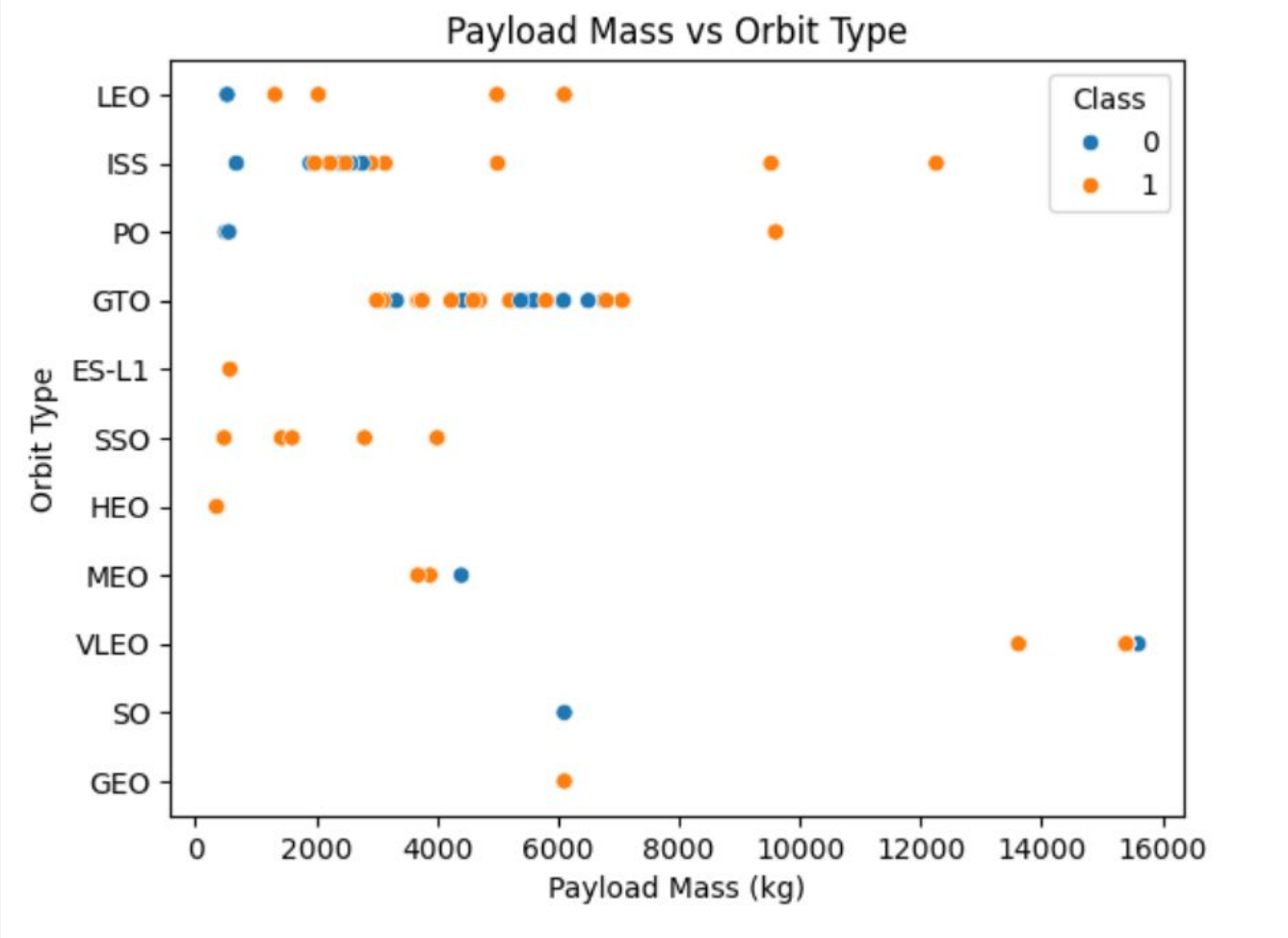
Flight Number vs. Orbit Type

- In the LEO orbit, success seems to be related to the number of flights, while in GTO there does not seem to be a relationship between the number of flights and success.



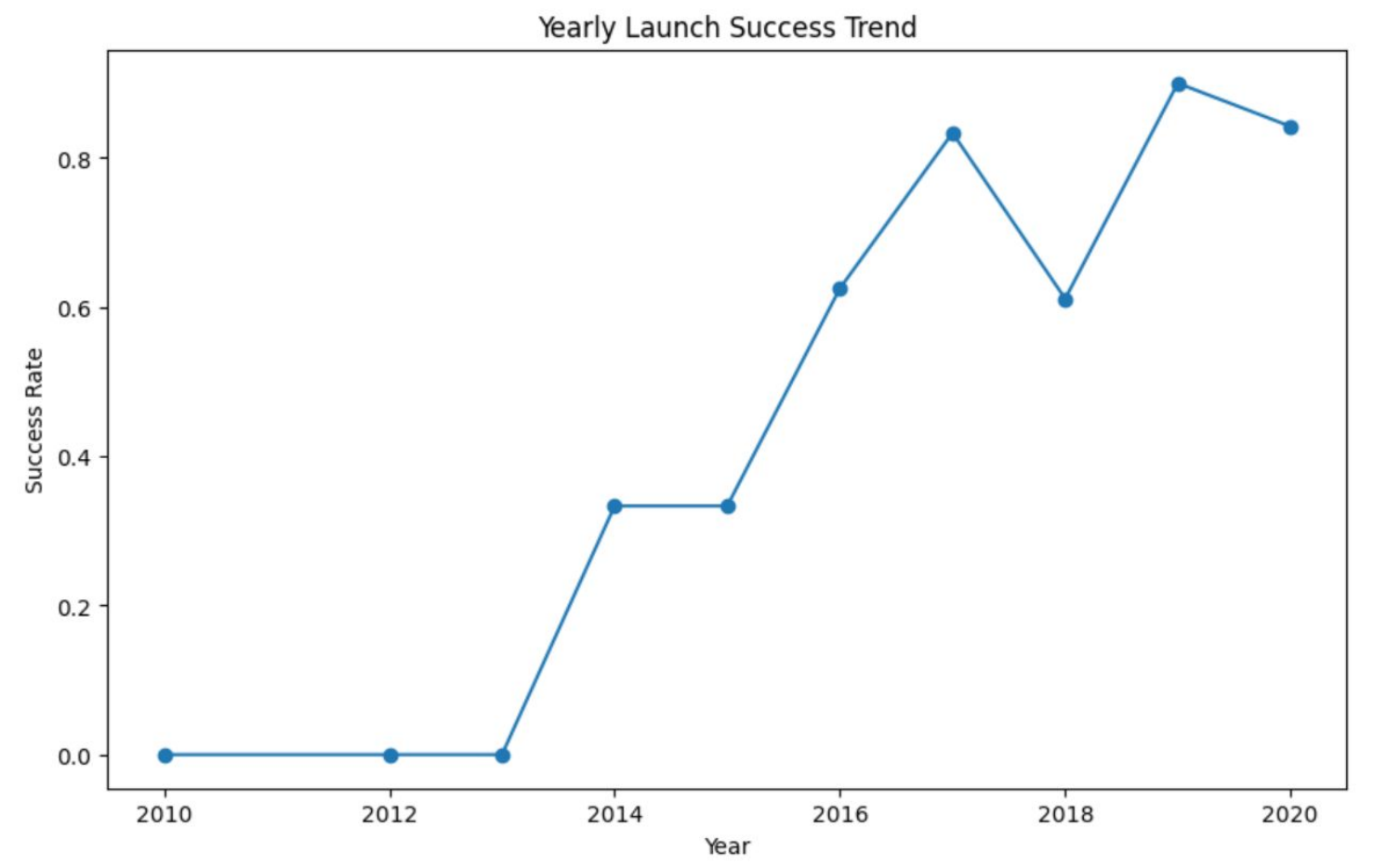
Payload vs. Orbit Type

- With heavy payloads the successful landing or positive landing rate is higher for POLAR, LEO and EEI



Launch Success Yearly Trend

- The succes rate since 2013 kept increasing till 2020



All Launch Site Names

- %sql select distinct(LAUNCH_SITE) from SPACEXTBL

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

Launch Site Names Begin with 'CCA'

```
con = sqlite3.connect("my_data1.db")
df =pd.read_sql_query('SELECT* FROM SPACEXTABLE', con)
filtered_df=df[df['Launch_Site'].str.startswith('CCA',na=False)]
filtered_df.head(5)
```

- With this part of code the connection to the SQLite database is established and if it does not exist it creates one with the name my_data1.db, executes the query on the database, and obtains the data from SPACEXTABLE which is saved in a Pandas DataFrame, only the rows where the column begins with the name CCA are selected and filters the nan values, finally it shows those first 5 rows with the DataFrame already filtered.

	Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
0	2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	2010-12-08	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	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
3	2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	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

- The dataframe is filtered to select only the rows where the customer column contains the string CRS, a new dataframe is created with the PAYLOAD_MASS__KG_ and Booster_Version columns, the total sum of the payload mass kg column in the dataframe is calculated, the calculated value is displayed and the columns.

```
df_CRS = df[df['Customer'].str.contains('CRS', na=False)]
df_Mass = df_CRS[['PAYLOAD_MASS__KG_', 'Booster_Version']]
total_payload_mass= df_Mass['PAYLOAD_MASS__KG_'].sum()
print("Total Payload Mass:", total_payload_mass)
df_Mass
```

```
Total Payload Mass: 48213
```

	PAYLOAD_MASS__KG_	Booster_Version
3	500	F9 v1.0 B0006
4	677	F9 v1.0 B0007
8	2296	F9 v1.1
12	2216	F9 v1.1 B1010
13	2395	F9 v1.1 B1012
16	1898	F9 v1.1 B1015
18	1952	F9 v1.1 B1018
22	3136	F9 FT B1021.1
26	2257	F9 FT B1025.1
29	2490	F9 FT B1031.1
34	2708	F9 FT B1035.1
38	3310	F9 B4 B1039.1
44	2205	F9 FT B1035.2
51	2647	F9 B4 B1039.2
56	2697	F9 B4 B1045.2
64	2500	F9 B5B1050
69	2495	F9 B5B1056.1
72	2268	F9 B5 B1056.2
75	2617	F9 B5B1059.1
81	1977	F9 B5 B1059.2
100	2972	F9 B5 B1058.4

Average Payload Mass by F9 v1.1

- Filter the data, calculate the average and display the average

```
df_F9=df[df['Booster_Version'].str.contains('F9 v1.1')]
average_payload_mass =df_F9['PAYLOAD_MASS__KG_'].mean()
print(f"Average payload mass carried by booster version F9 v1.1: {average_payload_mass:.2f} kg")
```

```
Average payload mass carried by booster version F9 v1.1: 2534.67 kg
```

First Successful Ground Landing Date

- Filter the data, search the first successful landing date and display the date

```
filtered_df=df[(df['Landing_Outcome'] == 'Success (ground pad)')]
first_success_date = filtered_df['Date'].min()
print("The first successful landing on a ground pad was on:", first_success_date)
```

```
The first successful landing on a ground pad was on: 2015-12-22
```

Successful Drone Ship Landing with Payload between 4000 and 6000

- Filters the dataframe data to obtain launches that meet the conditions and then displays the booster versions used in those launches

```
df_boost = df[
(df['PAYLOAD_MASS__KG_'] >4000) &
(df['PAYLOAD_MASS__KG_'] < 6000) &
(df['Landing_Outcome'] == 'Success (drone ship)')
]
boosters = df_boost['Booster_Version'].unique()
print(boosters)
```

```
['F9 FT B1022' 'F9 FT B1026' 'F9 FT  B1021.2' 'F9 FT  B1031.2']
```

Total Number of Successful and Failure Mission Outcomes

- Analyze data from a space mission to count how many missions were successful and how many were unsuccessful.

```
df_success = df[df['Mission_Outcome'].str.contains('Success',na=False)]
num_success= len(df_success)
df_failure= df[df['Mission_Outcome'].str.contains('Failure', na=False)]
num_failure = len(df_failure)
num_success, num_failure
```

```
(100, 1)
```

Boosters Carried Maximum Payload

- The code looks for the launch with the largest payload, obtains the samples of the versions of the propellants used and the result is the versions that were used in launches with maximum payload

```
max_payload=df['PAYLOAD_MASS__KG_'].max()
df_boosters=df[df['PAYLOAD_MASS__KG_'] == max_payload]['Booster_Version'].unique()
df_boosters
```

```
array(['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 '], dtype=object)
```

2015 Launch Records

- The code filters the launches that occurred in 2015 that had unsuccessful results on a floating platform (drone ship) and shows the month

```
df['Month'] = pd.to_datetime(df['Date']).dt.month_name()
df[df['Date'].str.startswith('2015') & df['Landing_Outcome'].str.contains('Failure', na=False)
& df['Landing_Outcome'].str.contains('drone ship', na=False)][['Month', 'Landing_Outcome', 'Booster_Version', 'Launch_Site']]
```

	Month	Landing_Outcome	Booster_Version	Launch_Site
13	January	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
16	April	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

- Filters launch data between dates, counts and shows the distribution of landing results

```
df_filtered=df[(df['Date']>='2010-06-04')&(df['Date'] <= '2017-03-20')]
df_filtered['Landing_Outcome'].value_counts().sort_values(ascending=False)
```

```
Landing_Outcome
No attempt               10
Failure (drone ship)      5
Success (drone ship)      5
Controlled (ocean)        3
Success (ground pad)      3
Failure (parachute)       2
Uncontrolled (ocean)      2
Precluded (drone ship)    1
Name: count, dtype: int64
```

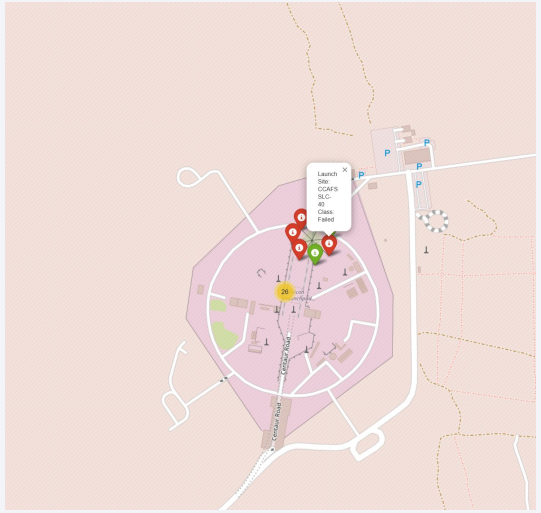

Section 3
Launch Sites
Proximities Analysis

Global markers for all launch sites

- SpaceX launch sites are located along the coasts

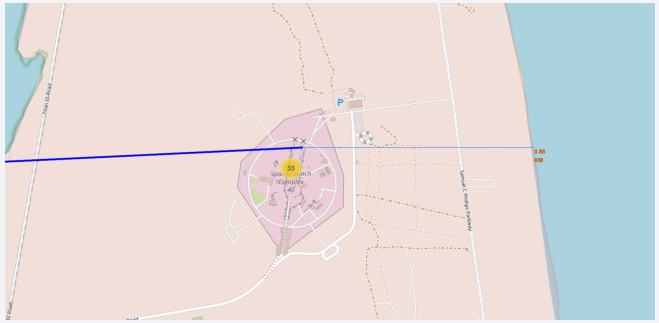
Markers

- In the screenshots you can see the launch sites, as well as if you zoom in on the image, you can see the green and red markers that indicate successful and unsuccessful launches.

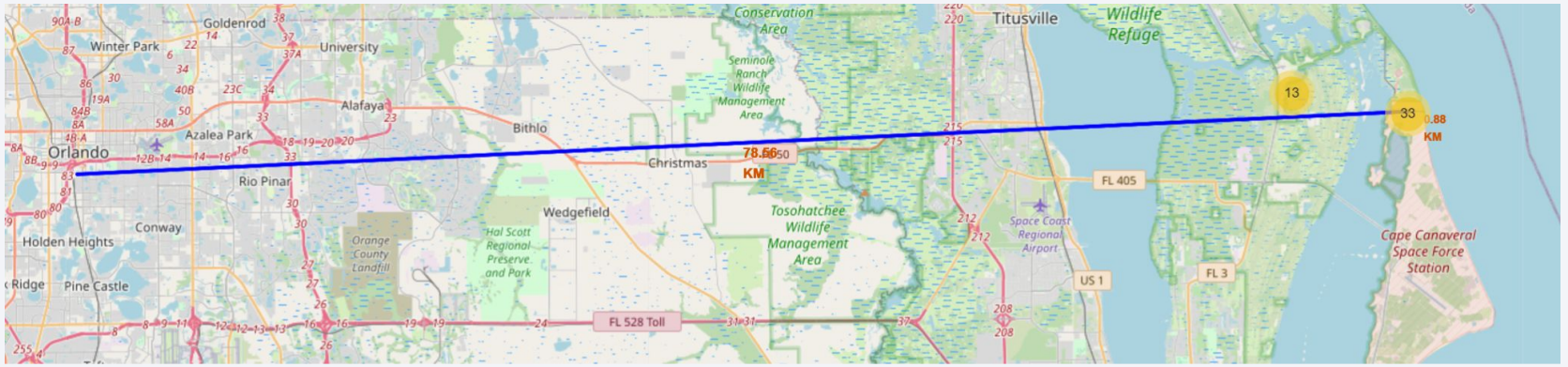


Distance between launch sites and landmarks using the site CCAFS-SLC-40 as reference

Distance to coastline

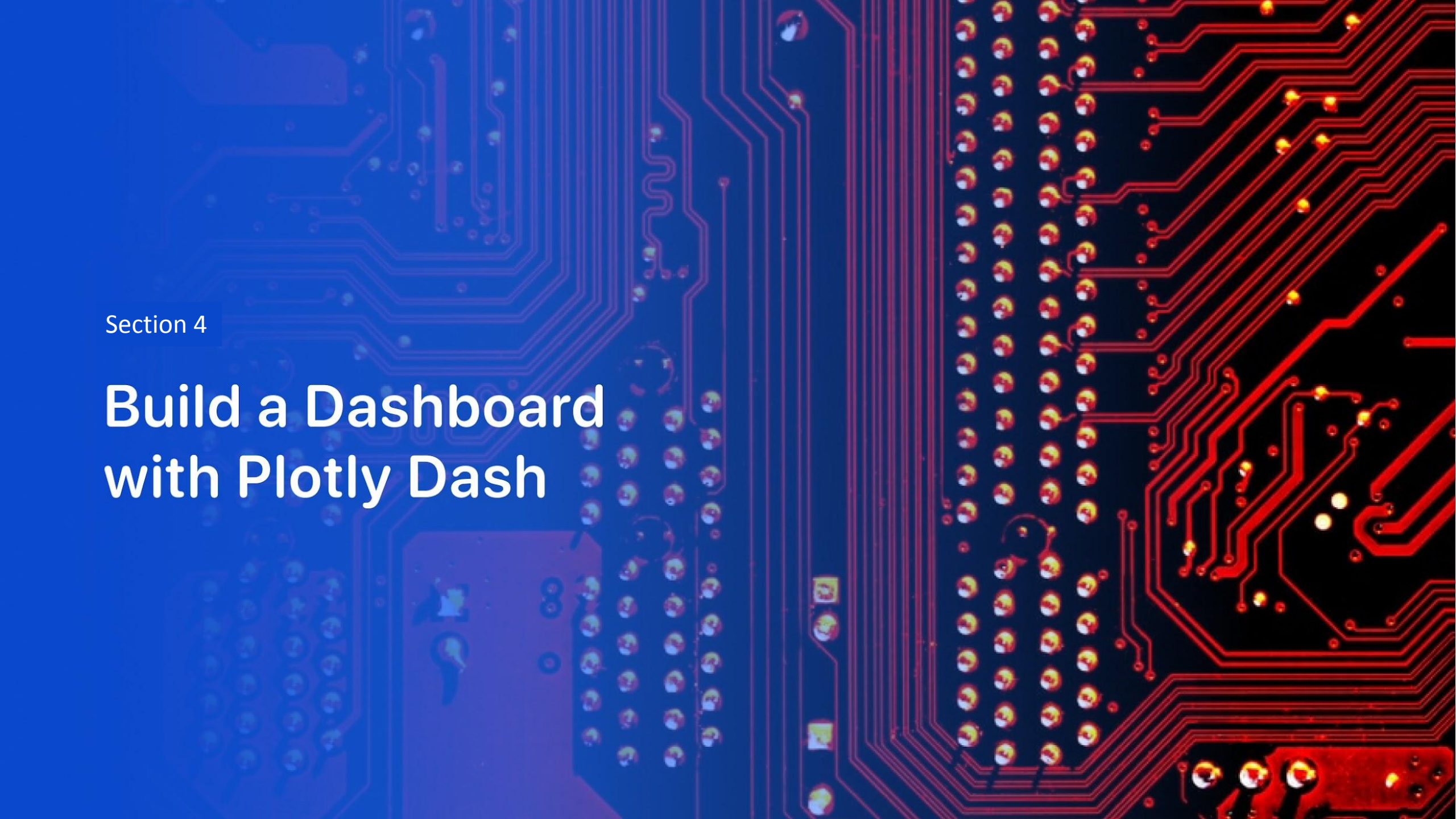


Distance to city



After you plot distance lines to the proximities, you can answer the following questions easily:

- 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, they are located near the coastline
- Do launch sites keep certain distance away from cities?Yes, they are quite far from the cities.


Section 4
Build a Dashboard with Plotly Dash

<Dashboard Screenshot 1>

- Replace <Dashboard screenshot 1> title with an appropriate title
- Show the screenshot of launch success count for all sites, in a piechart
- Explain the important elements and findings on the screenshot

<Dashboard Screenshot 2>

- Replace <Dashboard screenshot 2> title with an appropriate title
- Show the screenshot of the piechart for the launch site with highest launch success ratio
- Explain the important elements and findings on the screenshot

<Dashboard Screenshot 3>

- Replace <Dashboard screenshot 3> title with an appropriate title
- Show screenshots of Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider
- Explain the important elements and findings on the screenshot, such as which payload range or booster version have the largest success rate, etc.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

- Visualize the built model accuracy for all built classification models, in a bar chart

- Find which model has the highest classification accuracy

Confusion Matrix

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

Conclusions

- Point 1
- Point 2
- Point 3
- Point 4
- …

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!