IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Amel Azizi

2024-04-14

Outline

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

Executive Summary

- Summary of methodologies
  - Data collection via API, Web scraping
  - Exploratory Data Analysis (EDA) with Data Visualization
  - EDA with SQL
  - Interactive map with Folium
  - Dashboard with Plotly Dash
  - Predictive Analysis
- Summary of all results
  - EDA Results
  - Interactive Maps and Dashboard
  - Predictive Results

Introduction

- Project background and context
  - This project aims to forecast the successful landing of the Falcon 9 first stage. The launch for a Falcon 9 is 62$ millions, significantly lower than other providers, the reason is the ability of SpaceX to reuse the launcher. With all the previous missions we've got a lot of data to analyze that can improve the success rate of the landing outcome and draw new insights to explore.
- Problems you want to find answers
  - Is there a relationship between all the data ? Can we define pattern from them ?
  - What distinguish the success or the fail of the landing ?
  - What are the optimal conditions to a success landing ?

Section 1

Methodology

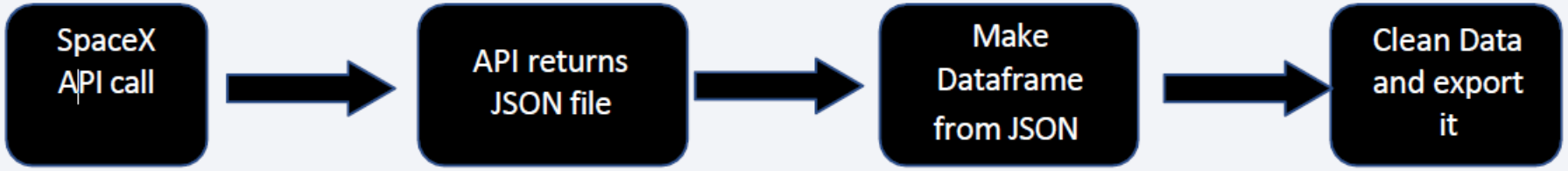
Methodology

Executive Summary

- Data collection methodology
  - SpaceX API, Web scraping from Wikipedia
- Perform data wrangling
  - Keeping the most useful data, one hot encoding for classification models
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - To define the insights and get answer from the data analysis

Data Collection

- Describe how data sets were collected.
  - The data retrieved from the API includes details about rockets, launches, and payload information



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

Data Collection – SpaceX API

1. Requesting rocket launch data from SpaceX API

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
response = requests.get(spacex_url)
```

2. Checking the content of the response

```
print(response.content)
```

3. Making the requested JSON results more consistent

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
response.status_code
data = pd.json_normalize(response.json())
data.head()
```

4. Transforming data from .json() into Pandas dataframe

```
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
getBoosterVersion(data)
```

5. Create dictionary with data

```
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}
```

6. Create dataframe from Launch dict

```
data = pd.DataFrame(launch_dict)
```

7. Filter dataframe to only include Falcon 9 launches

```
data_falcon9 = data[data.BoosterVersion == 'Falcon 9']
data_falcon9
```

8. Resetting the FlightNumber column

```
pd.options.mode.chained_assignment = None
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9
data_falcon9
```

Data Collection - Scraping

1. Requesting the Falcon9 launch Wiki page from its URL

```
html_data = requests.get(static_url)
html_data.status_code
```

2. Creating a BeautifulSoup Object from the HTML response

```
soup = BeautifulSoup(html_data.text, 'html.parser')
soup.title
```

3. Extracting all columns/variable names from the HTML table

```
html_tables = soup.find_all('table')
first_launch_table = html_tables[2]
print(first_launch_table)
```

4. Getting column names

```
element = soup.find_all('th')
for row in range(len(element)):
    try:
        name = extract_column_from_header(element[row])
        if (name is not None and len(name) > 0):
            column_names.append(name)
    except:
        pass
print(column_names)
```

5. Creating a dataframe by parsing the launch HTML tables

```
launch_dict= dict.fromkeys(column_names)
del launch_dict['Date and time ( )']
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'] = []
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

6. Adding data to keys

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
```

See link for the rest of code

7. Creating dataframe from dictionary

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

8. Exporting data file as CVS file

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

Data Wrangling

1. Calculating the number of launches on each site

```
df['LaunchSite'].value_counts()
```

```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

2. Calculating the number and occurence of each orbit

```
df['Orbit'].value_counts()
```

```
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
ES-L1     1
HEO       1
SO        1
GEO       1
Name: Orbit, dtype: int64
```

3. Calculating the number and occurrence of mission outcome of the orbits

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

```
True ASDS       41
None None       19
True RTLS       14
False ASDS       6
True Ocean       5
False Ocean      2
None ASDS        2
False RTLS       1
Name: Outcome, dtype: int64
```

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

4. Creating a landing outcome label from Outcome column

```
landing_class = df['Outcome'].replace({'False Ocean': 0, 'False ASDS': 0,
df['Outcome'] = df['Outcome'].astype(int)
df.info()
```

See link for the rest of code

```
df['Class']=landing_class
df[['Class']].head(8)
df.head(5)
df["Class"].mean()
```

5. Exporting data file as CVS file

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

EDA with Data Visualization

1. Importing libraries and defining auxiliary functions

```
import piplite
await piplite.install(['numpy'])
await piplite.install(['pandas'])
await piplite.install(['seaborn'])
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

2. Reading the SpaceX dataset into a Pandas dataframe and print its summary

```
from js import fetch
import io

URL = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_2.csv"
resp = await fetch(URL)
dataset_part_2_csv = io.BytesIO((await resp.arrayBuffer()).to_py())
df=pd.read_csv(dataset_part_2_csv)
df.head(5)
```

3. Plotting FlightNumber vs. PayloadMass

```
sns.catplot(y="PayloadMass", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Pay load Mass (kg)",fontsize=20)
plt.show()
```

4. Plotting FlightNumber vs. LaunchSite

```
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number", fontsize=20)
plt.ylabel("Launch Site", fontsize=20)
plt.show()
```

5. Plotting PayloadMass vs. LaunchSite

```
sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("Payload Mass (kg)", fontsize=20)
plt.ylabel("Launch Site", fontsize=20)
plt.show()
```

6. Visualizing success rate of each orbit type

```
plt.rcParams["figure.figsize"] = (20,10)
plt.bar(df_rank.index,df_rank["Outcome"])
plt.xlabel("Orbit", fontsize=15)
plt.ylabel("Outcome Count", fontsize=15)
```

7. Plotting FlightNumber vs. Orbit type

```
sns.catplot(x="FlightNumber", y="Orbit", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number", fontsize = 20)
plt.ylabel("Orbit", fontsize=20)
plt.show()
```

8. Plotting PayloadMass vs. Orbit type

```
sns.catplot(x="PayloadMass", y="Orbit", hue="Class", data=df, aspect = 5)
plt.xlabel("Payload (kg)", fontsize=20)
plt.ylabel("Orbit", fontsize=20)
plt.show()
```

9. Visualizing launch success yearly trend

```
sns.lineplot(x=df1.index, y=df1["Class"])
plt.xlabel("Year", fontsize=20)
plt.ylabel("Outcome", fontsize=20)
plt.show()
```

10. Features engineering

```
features = df[['FlightNumber', 'PayloadMass', 'Orbit',
features.head()
```

11. Creating dummy variables to categorize columns

```
features_one_hot = pd.get_dummies(features)
features_one_hot.head()
```

12. Casting all numeric columns to ‘float64’

```
features_one_hot.astype('float64')
```

EDA with SQL

1. Downloading the dataset

```
!pip install sqlalchemy==1.3.9
```

2. Connecting to the dataset

```
!pip install ibm_db_sa
!pip install ipython-sql
```

```
%load_ext sql
```

```
con = sqlite3.connect("my_data1.db")
cur = con.cursor()
```

```
!pip install -q pandas==1.1.5
```

```
%sql sqlite:///my_data1.db
```

```
df = pd.read_csv("https://cf-courses-data.s3.us.cloud
df.to_sql("SPACEXTBL", con, if_exists='replace', inde
```

```
%sql create table SPACEXTABLE as select * from SPACEXTBL where Date is not null
```

3. Displaying the names of the unique launch sites in the space mission

```
%sql SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL;
```

4. Displaying 5 records where launch sites begin with the string 'CCA'

```
%sql SELECT LAUNCH_SITE from SPACEXTBL where (LAUNCH_SITE) LIKE 'CCA%' LIMIT 5;
```

5. Displaying the total payload mass carried by boosters launched by NASA (CRS)

```
%sql select sum(PAYLOAD_MASS__KG_) as "Sum of payload mass in kg" from SPACEXTBL;
```

6. Displaying the average payload mass carried by booster version F9 v1.1

```
%sql select avg(PAYLOAD_MASS__KG_) as "Avg of payload mass in kg" from SPACEXTBL;
```

7. Listing the date after first successful landing

```
%sql select min(DATE) from SPACEXTBL where "Landing_Outcome" like '%Success%'
```

8. Listing the names of the boosters with success

```
%sql select BOOSTER_VERSION from SPACEXTBL where LANDING_OUTCOME='Success (drone ship)' and PAYLOAD_MASS__KG_ BETWEEN 4000 and 6000;
```

9. Listing the total number of successful and failure mission outcomes

```
%sql select (select count("Mission_Outcome") from SPACEXTBL where "Mission_Outcome" like '%Success%') as Success, \
(select count("Mission_Outcome") from SPACEXTBL where "Mission_Outcome" like '%Failure%') as Failure
```

10. Listing the names of the booster_versions carrying the max payload mass

```
%sql select BOOSTER_VERSION as boosterversion from SPACEXTBL where PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_) from SPACEXTBL);
```

11. Listing the records displaying month names

```
%sql SELECT strftime('%m', DATE) AS Month, MISSION_OUTCOME, BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WHERE strftime('%Y', DATE)='2015';
```

12. Ranking the count of landing outcomes

```
%sql SELECT LANDING_OUTCOME FROM SPACEXTBL WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' ORDER BY DATE DESC;
```

EDA with SQL

- SQL queries was conducted to gather and understand data from dataset :
  - Displaying launch sites
  - Show 5 launch where the site begin by ‘CCA’
  - Total payload mass carried by boosters launch by NASA
  - Show the average payload mass carried by the booster F9 V1. 1.
  - List the first successful landing on the ground pas was achieved
  - List the names of the boosters that succeeded in landing on a drone ship and had a payload mass greater than 4000 but less than 6000 kg.
  - List the total number of successful and failed missions.
  - List the booster version that carried the maximum payload mass
  - List the records displaying the months, failed landing outcomes on a drone ship, boosters versions, and launch sites for the months in the year 2015.
  - Rank the count of successful landing outcomes between the dates 2010-06-04 and 2017-03-20.

Build an Interactive Map with Folium

- The Folium map object represents a map centered on the NASA Johnson Space Center in Houston, Texas. It includes:
  - A red circle at the NASA Johnson Space Center's coordinates, with a label showing its name (created using folium.Circle and folium.map.Marker).
  - Red circles at each launch site's coordinates, with labels showing the launch site names (implemented using folium.Circle, folium.map.Marker, and folium.features.DivIcon).
  - The grouping of points into clusters to display multiple pieces of information for the same coordinates (utilizing folium.plugins.MarkerCluster).
  - Markers to denote successful and unsuccessful landings, with green markers indicating successful landings and red markers indicating unsuccessful landings (constructed using folium.map.Marker and folium.Icon).
  - Markers to show the distance between launch sites and key locations such as railways, highways, coastways, and cities, with lines plotted between them (created using folium.map.Marker, folium.PolyLine, and folium.features.DivIcon).
- These objects are designed to provide a better understanding of the problem and the data. They allow for easy visualization of all launch sites, their surroundings, and the number of successful and unsuccessful landings.

Build a Dashboard with Plotly Dash

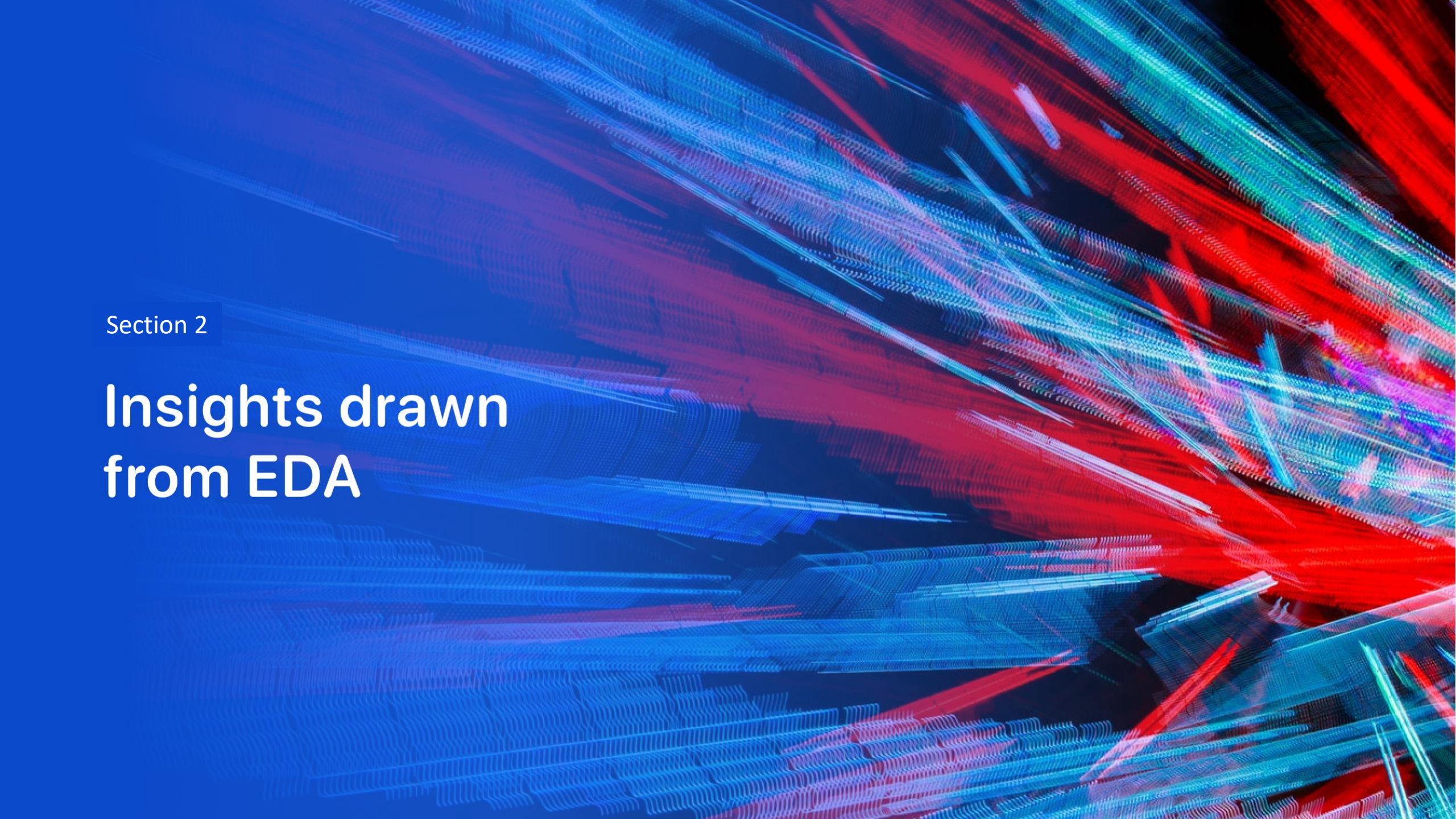
- The dashboard consists of several components:
  - A dropdown menu (dash_core_components.Dropdown) that enables users to select either a specific launch site or view data for all launch sites.
  - A pie chart (plotly.express.pie) depicting the total success and failure outcomes for the chosen launch site from the dropdown menu.
  - A rangeslider (dash_core_components.RangeSlider) that allows users to select a payload mass within a predefined range.
  - A scatter plot (plotly.express.scatter) illustrating the relationship between two variables, specifically Success vs Payload Mass.

Predictive Analysis (Classification)

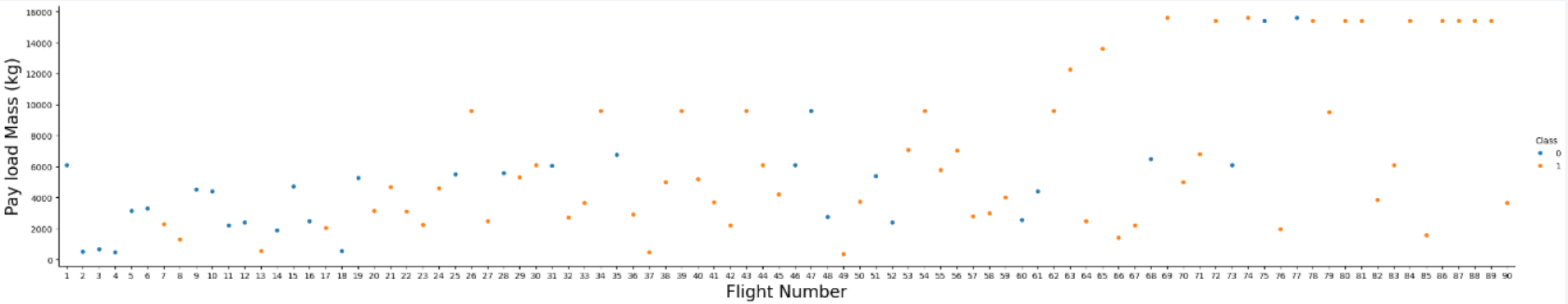
- Data preprocessing:
  - Dataset loading
  - Data normalization
  - Splitting data into training and test sets
- Model setup:
  - Selection of machine learning algorithms
  - Setting parameters for each algorithm using GridSearchCV
  - Training GridSearchCV models with the training dataset
- Model evaluation:
  - Obtaining the best hyperparameters for each model type
  - Computing accuracy for each model using the test dataset
  - Plotting the Confusion Matrix
- Model comparison:
  - Comparing models based on their accuracy
  - Selecting the model with the highest accuracy (refer to Notebook for results)

Results

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

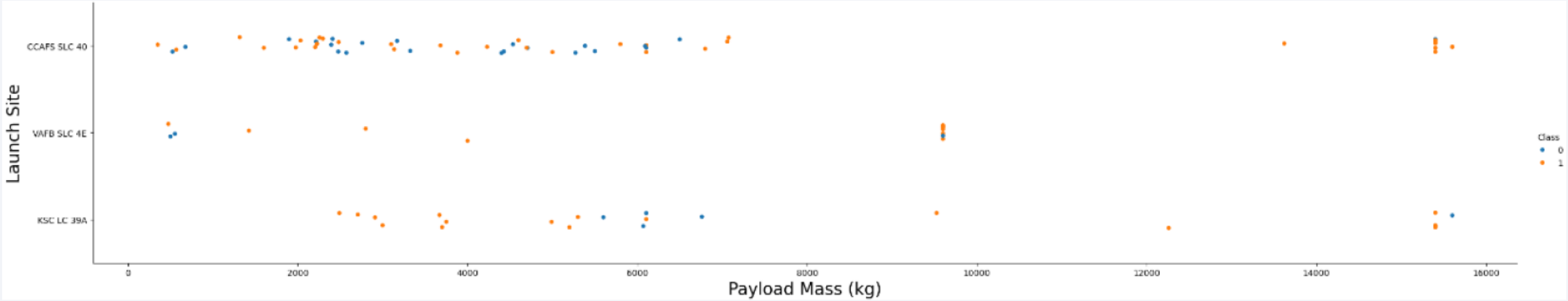

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



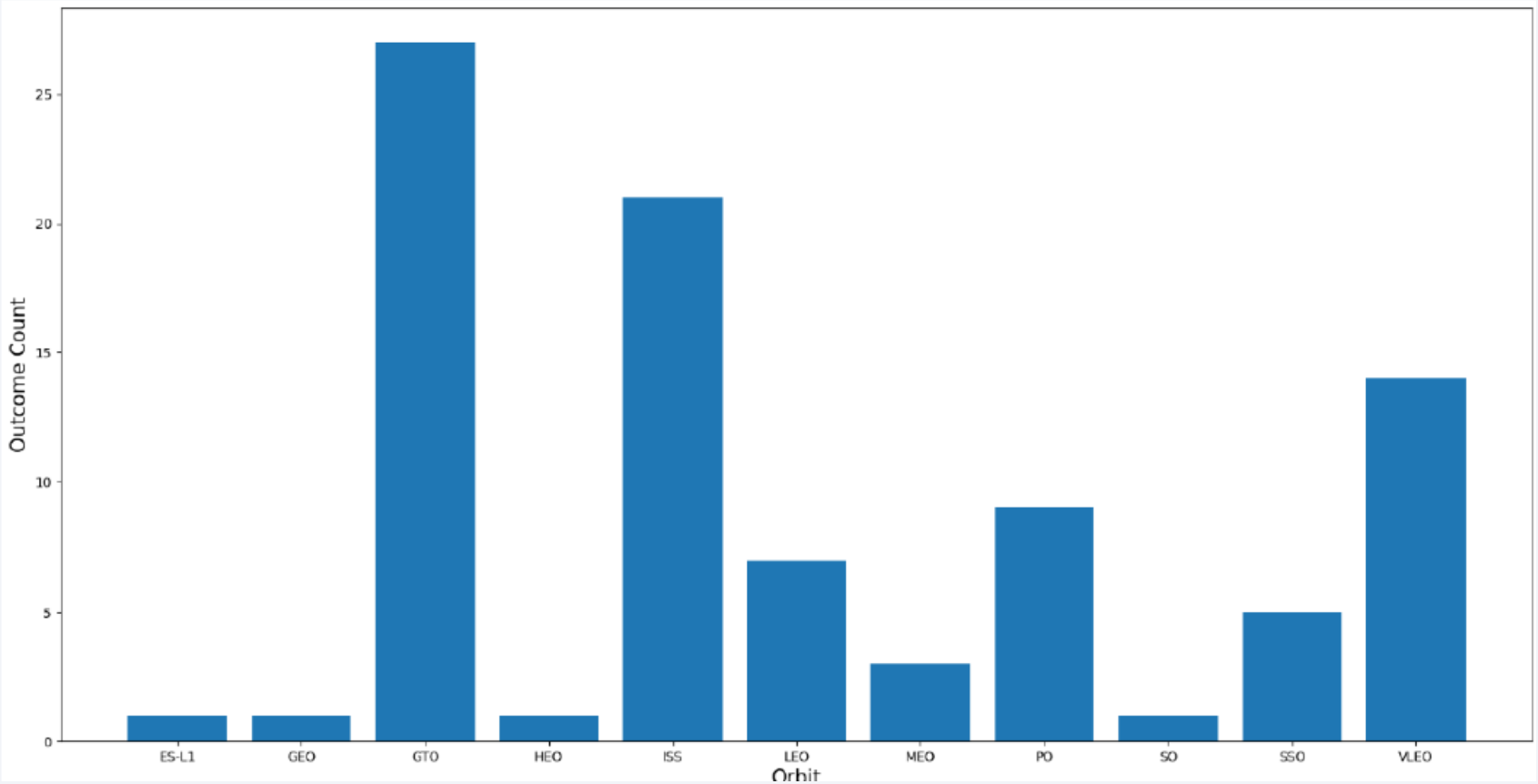
Insight: As the flight increases, the payload mass increases

Payload vs. Launch Site



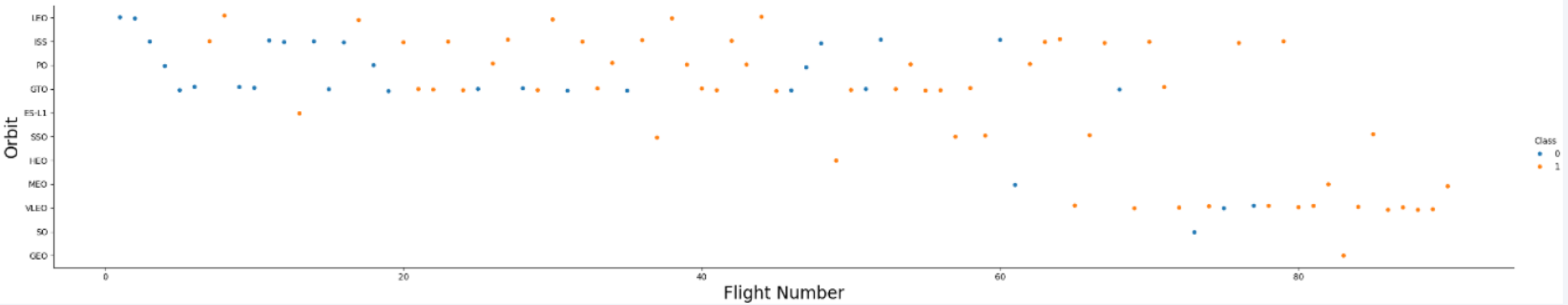
Insight: Launch site seems associated with various payloads

Success Rate vs. Orbit Type



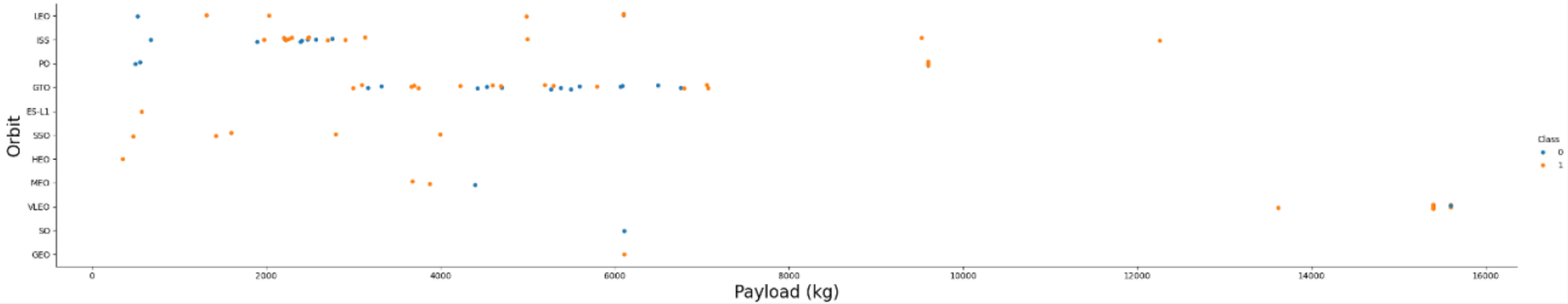
Insight: GTO, ISS, VLEO have the best success rate

Flight Number vs. Orbit Type



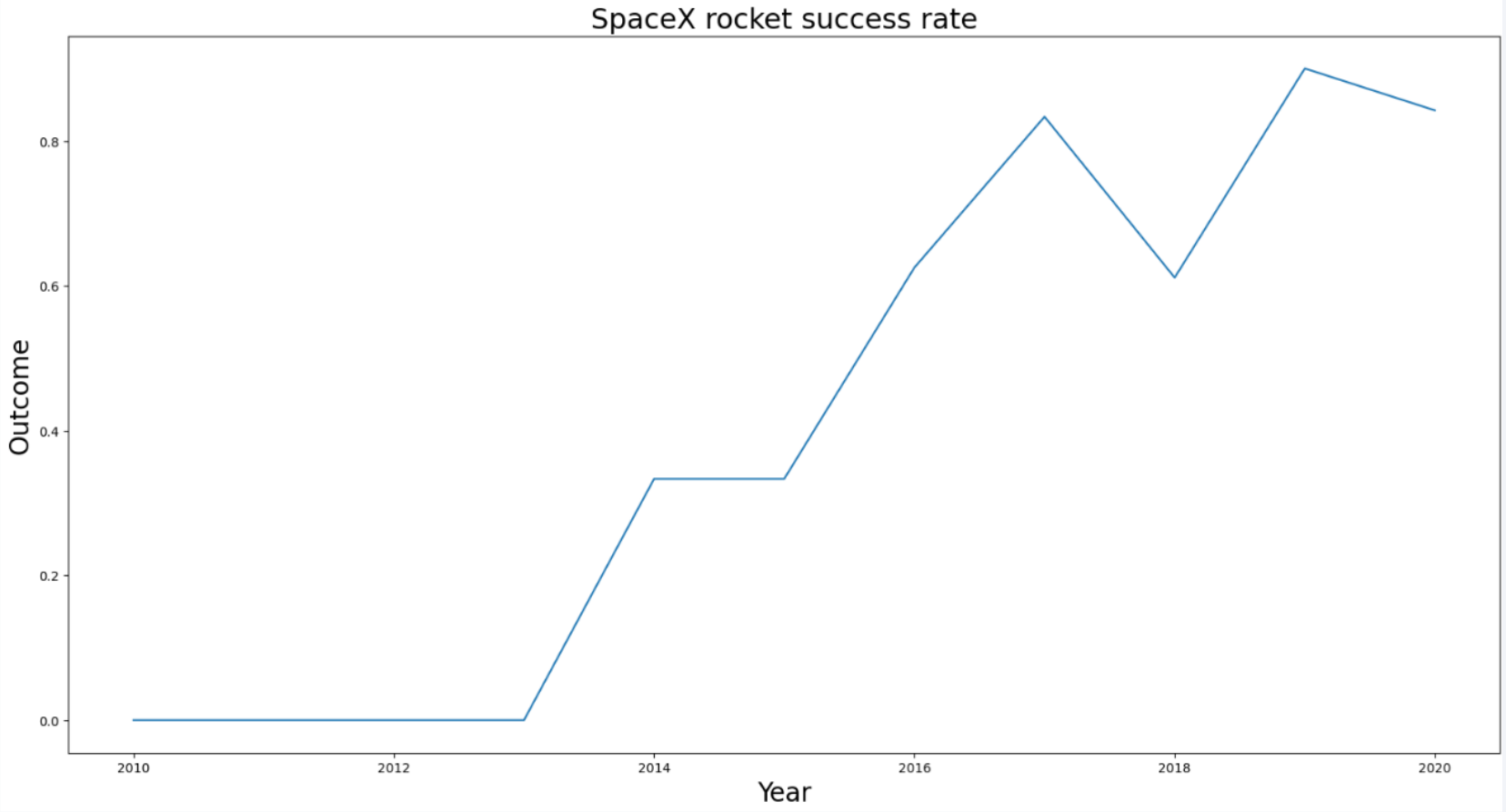
Insight: LEO, ISS, PO and GTO are mainly associated with 1-80 flight numbers while the rest are linked with > 60 flight numbers

Payload vs. Orbit Type



Insight: Orbit type is influenced by the payload mass

Launch Success Yearly Trend



Insight: Success rate since 2013 kept increasing till 2020

All Launch Site Names

SQL Query

```
%sql SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL;
```

Results

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

Insight: The use of DISTINCT in the query allows to remove duplicate LAUNCH_SITE

Launch Site Names Begin with 'CCA'

SQL Query

```
%sql SELECT LAUNCH_SITE from SPACEXTBL where (LAUNCH_SITE) LIKE 'CCA%' LIMIT 5;
```

Results

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

Insight: The WHERE clause followed by LIKE clause filters launch sites that contain the substring CCA.
LIMIT 5 shows 5 record from filtering

Total Payload Mass

SQL Query

```
%sql select sum(PAYLOAD_MASS__KG_) as payloadmass from SPACEXTBL;
```

Results

payloadmass
619967

Insight: This query returns the sum of all paylaod masses where the customer is NASA (CRS)

Average Payload Mass by F9 v1.1

SQL Query

```
%sql select avg(PAYLOAD_MASS__KG_) as payloadmass from SPACEXTBL;
```

Results

payloadmass
6138.287128712871

Insight: This query returns the average of all payload masses where the booster version contains the substring F9 V1. 1.

First Successful Ground Landing Date

SQL Query

```
%sql select min(DATE) from SPACEXTBL;
```

Results

min(DATE)
2010-06-04

Insight: With this query, we select the oldest succesful landing.
The WHERE clause filters dataset in order to keep only records where landing was successful. With the MIN function, we select the record with the oldest date.

Successful Drone Ship Landing with Payload between 4000 and 6000

SQL Query

```
%sql select BOOSTER_VERSION from SPACEXTBL where LANDING_OUTCOME='Success (drone ship)' and PAYLOAD_MASS__KG_ BETWEEN 4000 and 6000;
```

Results

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

Insight: This query returns the booster version where landing was successful and payload mass is between 4000 and 6000 kg. The WHERE and AND clauses filter the dataset

Total Number of Successful and Failure Mission Outcomes

SQL Query

```
%sql select (select count("Mission_Outcome") from SPACEXTBL where "Mission_Outcome" like '%Success%') as Success, \
(select count("Mission_Outcome") from SPACEXTBL where "Mission_Outcome" like '%Failure%') as Failure
```

Results

Success	Failure
100	1

Insight: With the first SELECT, we show the subqueries that return results. The first subquery counts the successful mission. The second subquery counts the unsuccessful mission. The WHERE clause followed by LIKE clause filters mission outcome. The COUNT function counts records filtered

Boosters Carried Maximum Payload

SQL Query

```
%sql select BOOSTER_VERSION as boosterversion from SPACEXTBL where PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_) from SPACEXTBL);
```

Results

boosterversion
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

Insight: We used a subquery to filter data by returning only the heaviest payload mass with MAX function. The main query uses subquery results and returns unique booster version (SELECT DISTINCT) with the heaviest payload mass

2015 Launch Records

SQL Query

```
%sql SELECT strftime('%m', DATE) AS Month, MISSION_OUTCOME, BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WHERE strftime('%Y', DATE)='2015';
```

Results

Month	Mission_Outcome	Booster_Version	Launch_Site
01	Success	F9 v1.1 B1012	CCAFS LC-40
02	Success	F9 v1.1 B1013	CCAFS LC-40
03	Success	F9 v1.1 B1014	CCAFS LC-40
04	Success	F9 v1.1 B1015	CCAFS LC-40
04	Success	F9 v1.1 B1016	CCAFS LC-40
06	Failure (in flight)	F9 v1.1 B1018	CCAFS LC-40
12	Success	F9 FT B1019	CCAFS LC-40

Insight: This query returns month, booster version, launch site where landing was unsuccessful and landing date took place in 2015. Substr function process date in order to take month or year.

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

SQL Query

```
%sql SELECT LANDING_OUTCOME FROM SPACEXTBL WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' ORDER BY DATE DESC;
```

Results

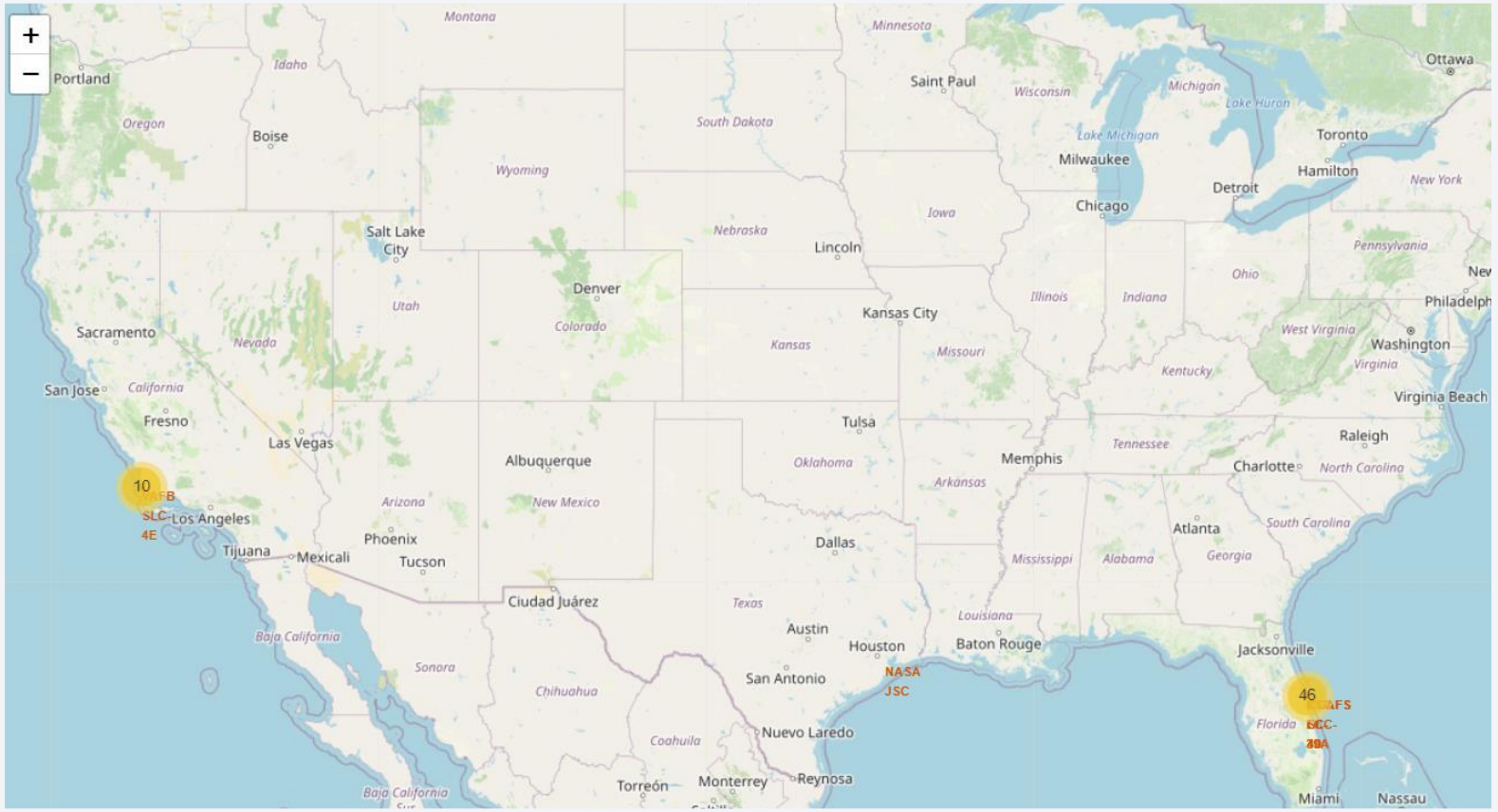
Landing_Outcome
No attempt
Success (ground pad)
Success (drone ship)
Success (drone ship)
Success (ground pad)
Failure (drone ship)
Success (drone ship)
Success (drone ship)
Success (drone ship)
Failure (drone ship)
Failure (drone ship)
Success (ground pad)
Precluded (drone ship)
No attempt

Insight: This query returns landing outcomes and their count where mission was successful and date is between 2010-06-04 and 2017-03-20. The GROUPBY clause groups results by landing outcome and ORDER BY COUNT DESC shows results in decreasing order

Section 3

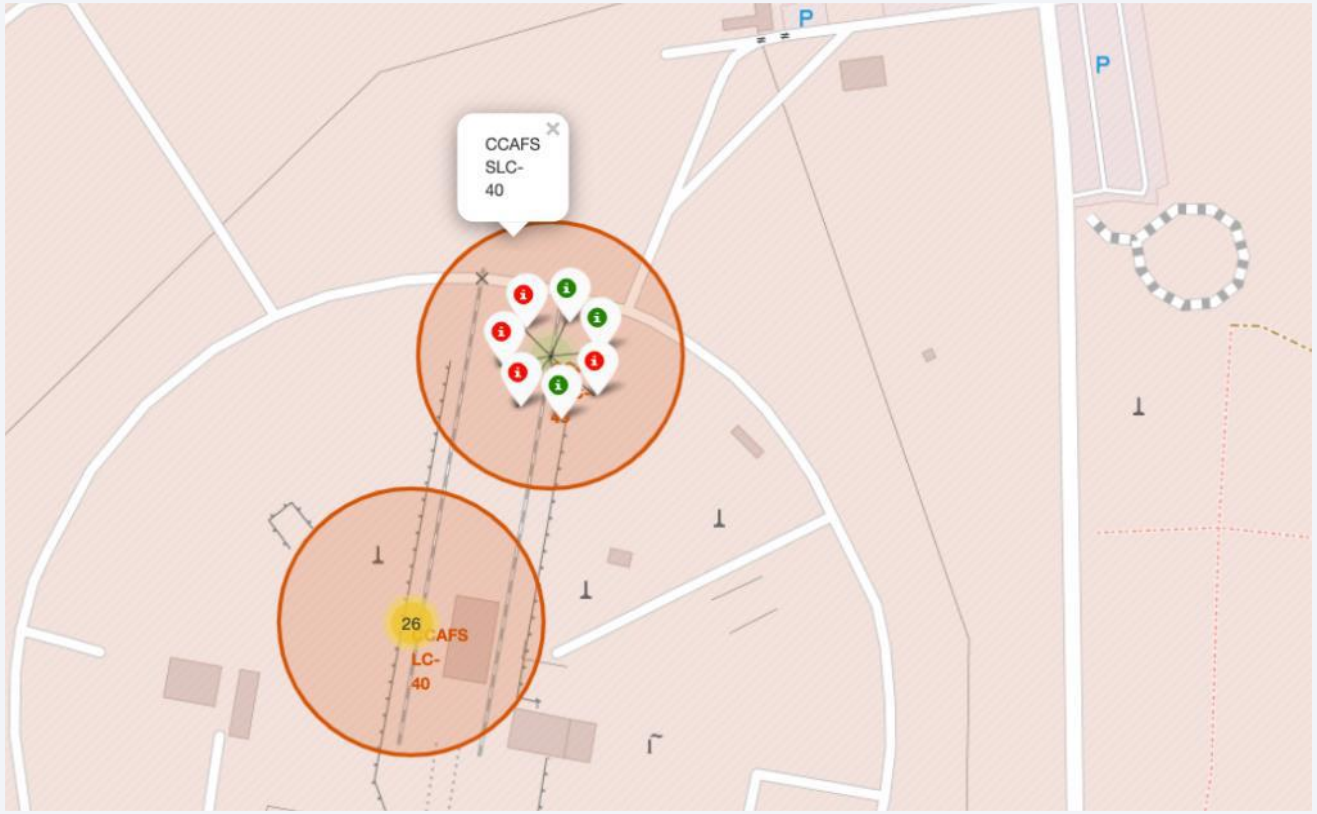
Launch Sites Proximities Analysis

<Folium Map Screenshot 1>



We see that the SpaceX launch site are on the coast of the United States and near the equator

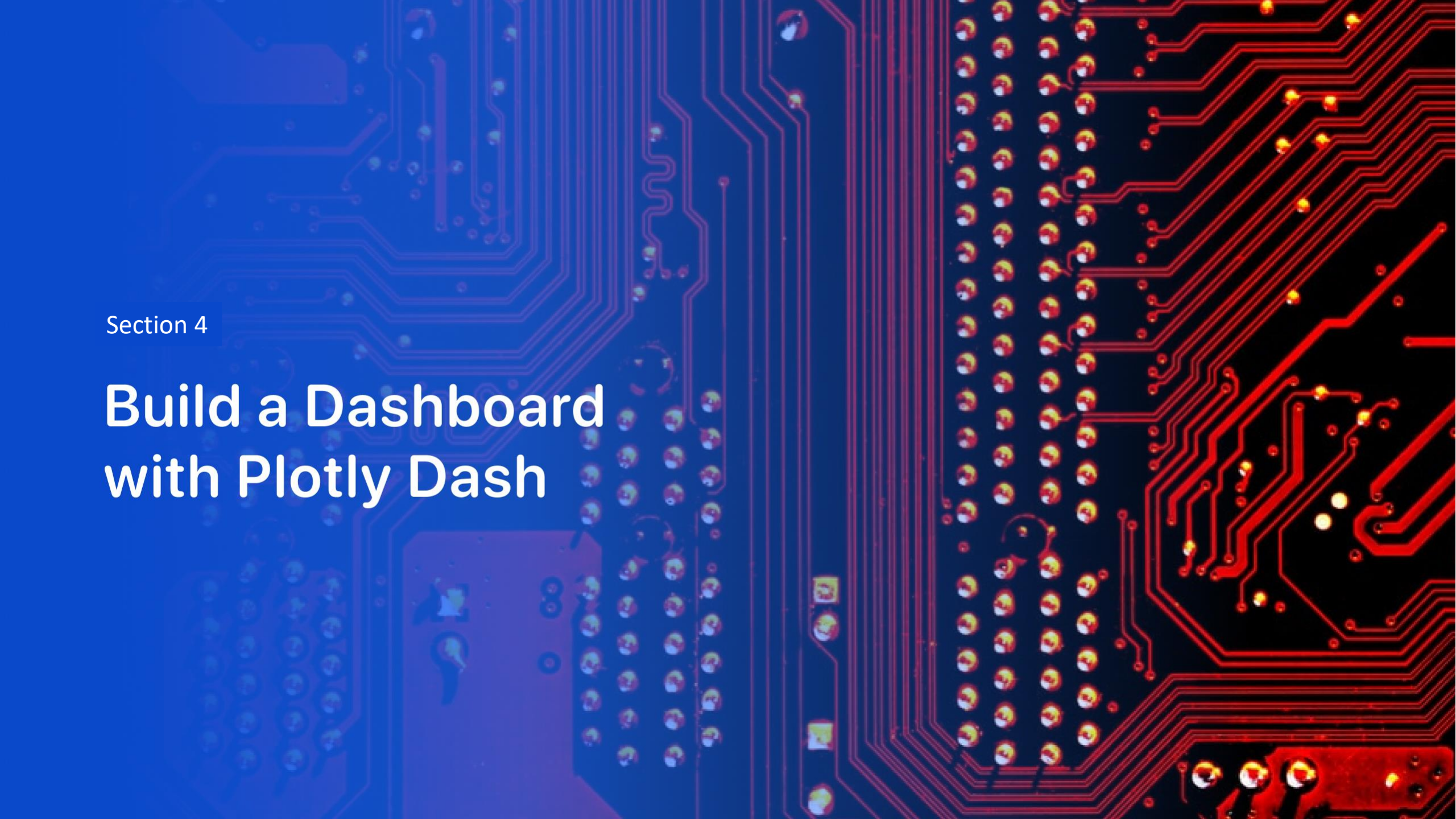
<Folium Map Screenshot 2>



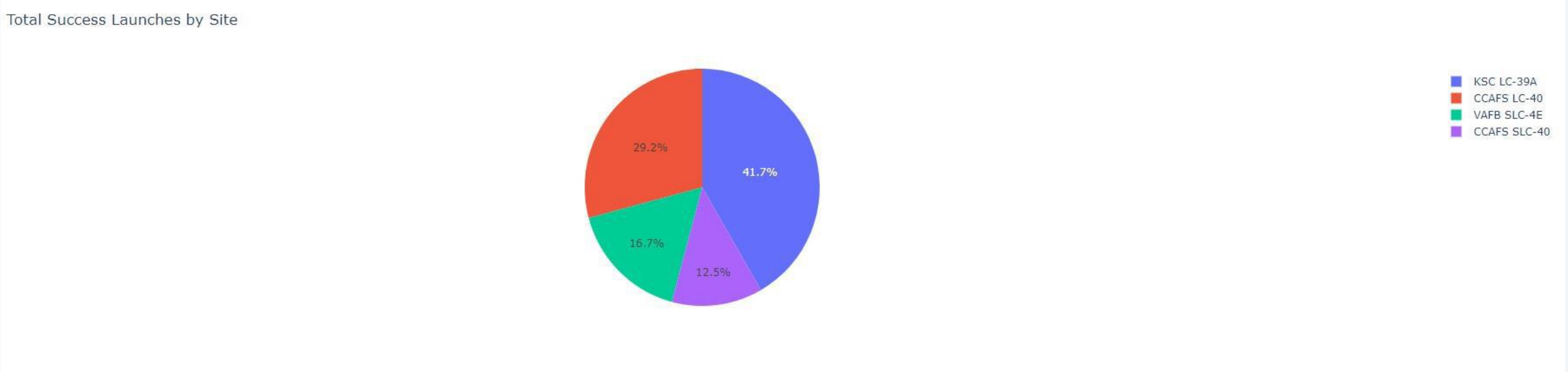
Green marker represents successful launches while Red marker represnts unsuccessful launches

<Folium Map Screenshot 3>

CCFS SLC-40 is in close proximity with railways, highways and coastlines aswell close to cities

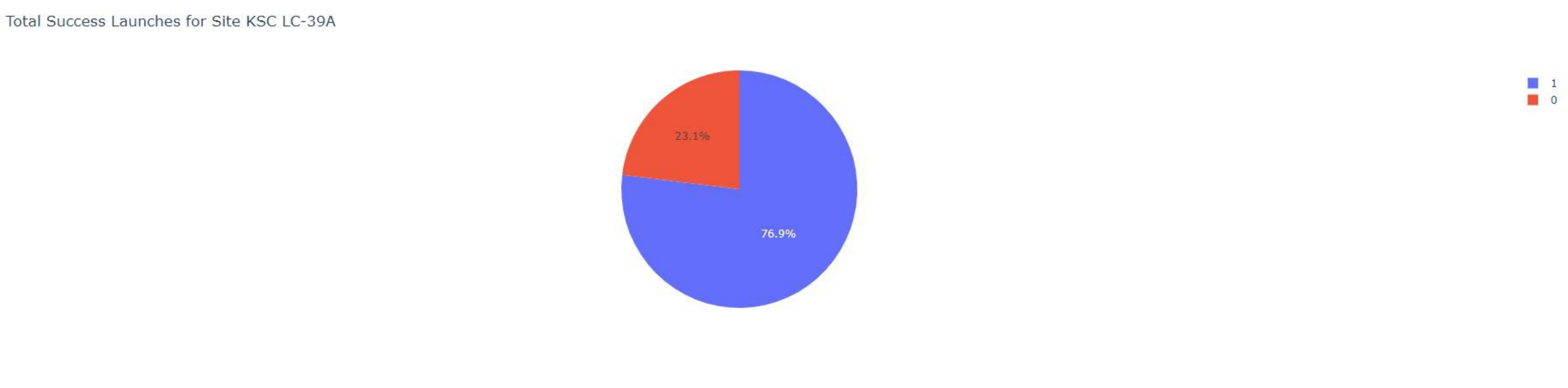

Section 4
Build a Dashboard with Plotly Dash

<Dashboard Screenshot 1>



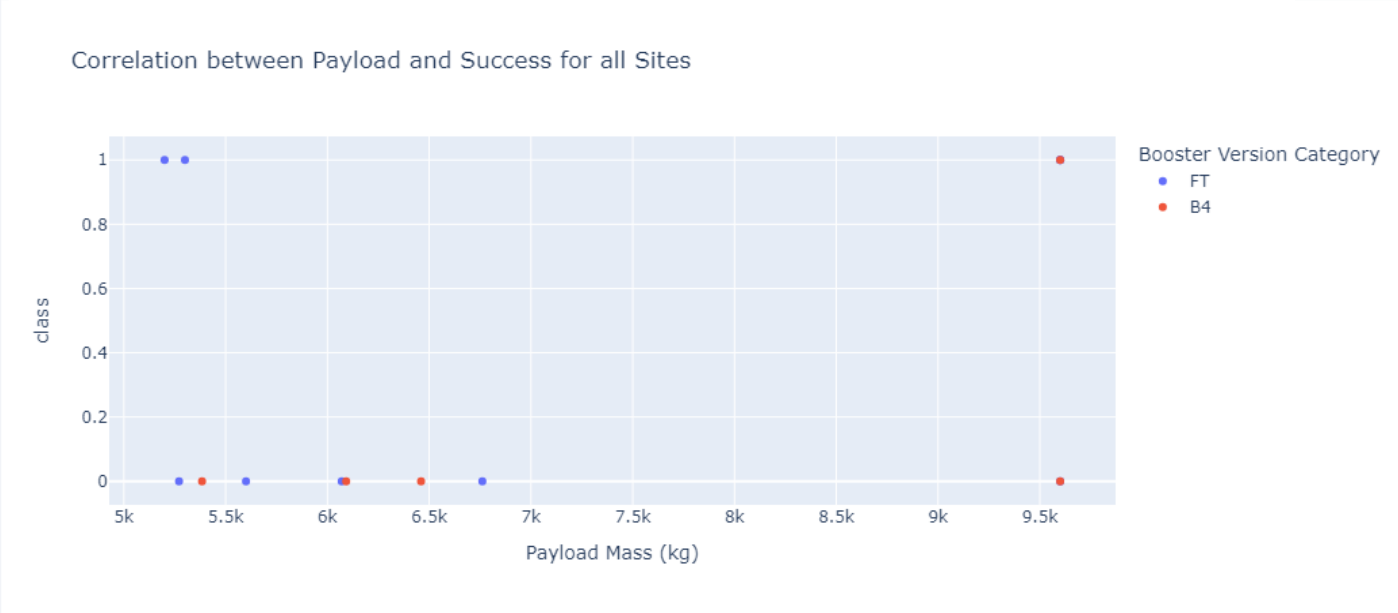
KSC LC-39A has the best success rate (41,7%) of launches while CCAFS SLC-40 has the least chance of success rate (12,5%)

<Dashboard Screenshot 2>



KSC LC-39A has achieved a 76,9% success rate with a 23,1% failure rate

<Dashboard Screenshot 3>



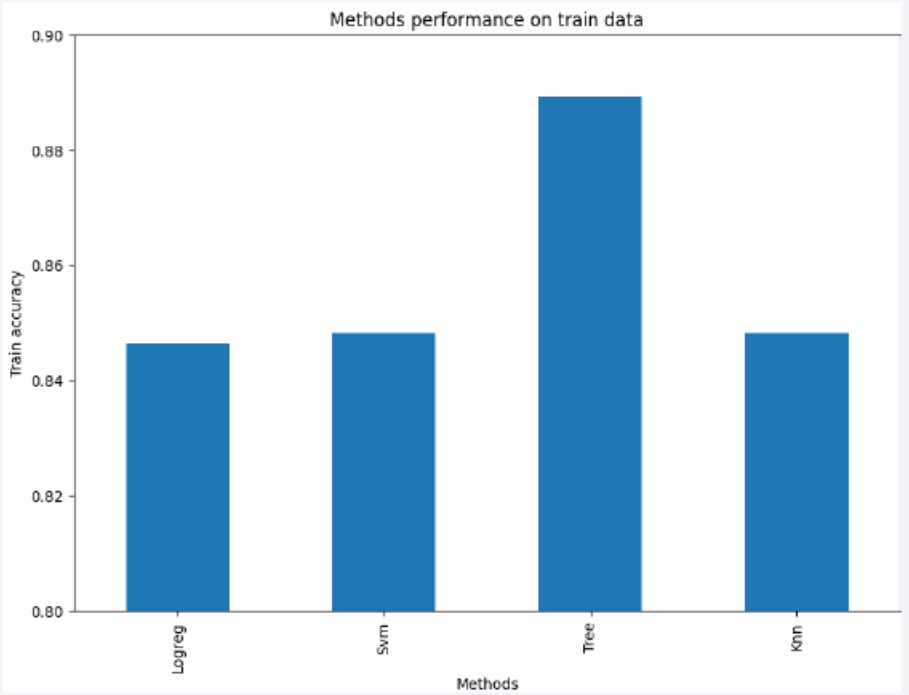
Low weighted payloads have a better success rate than the heavy weighted payloads

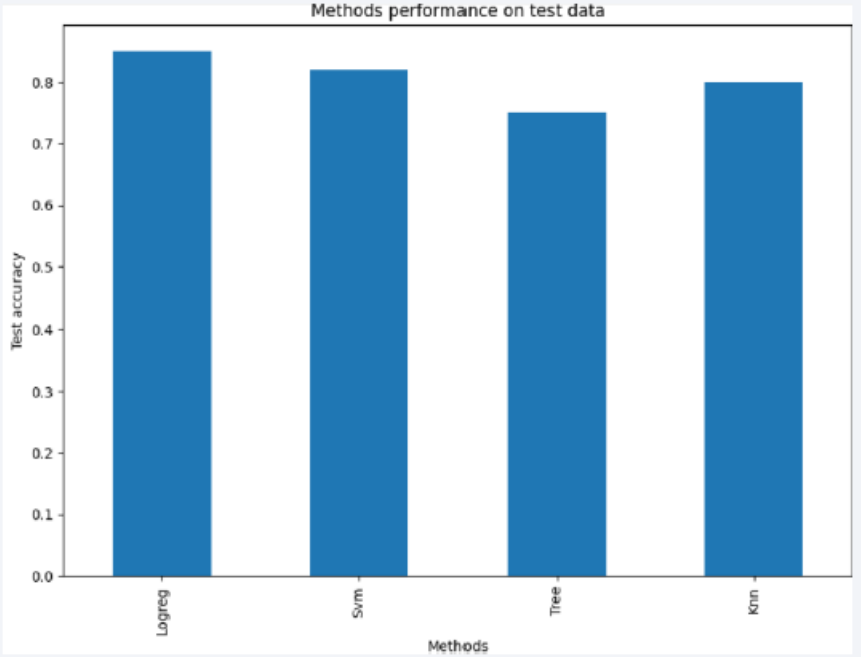
Section 5

Predictive Analysis (Classification)

Classification Accuracy

	Accuracy Train	Accuracy Test
Tree	0.889286	0.75
Knn	0.848214	0.80
Svm	0.848214	0.82
Logreg	0.846429	0.85

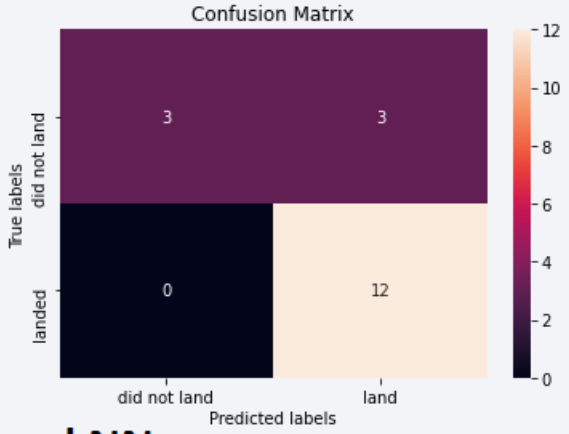




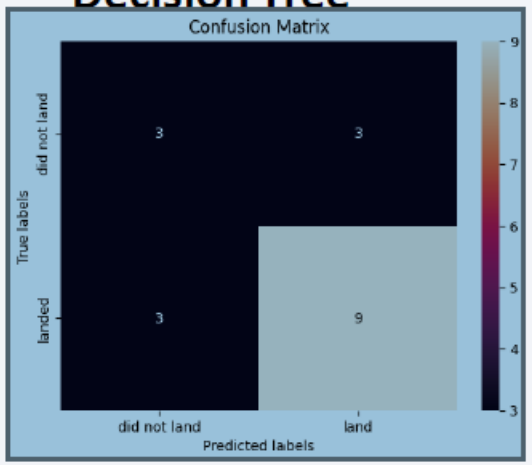
Logreg is the one with the best accuracy

Confusion Matrix

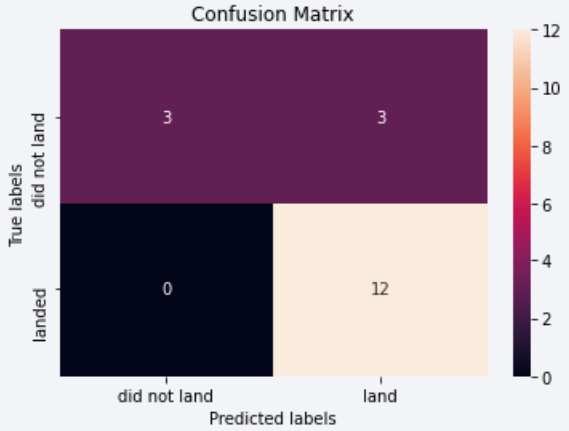
Logistic regression



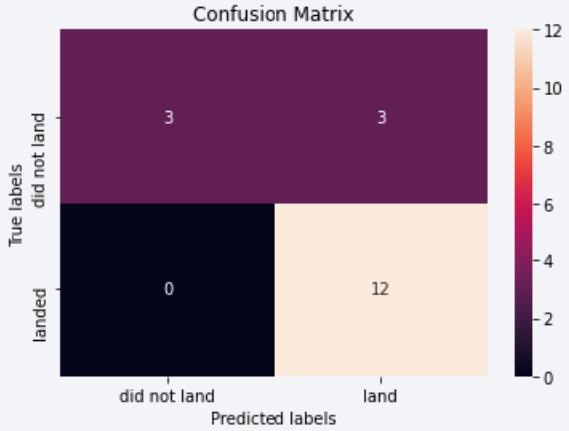
Decision Tree



kNN



SVM



Decision tree's performance was mixed

Conclusions

- The success of a mission can be attributed to various factors, including orbit, payload mass, launch site, etc… Over the large amount of data we can cross the analysis to define the best scenarios for launch.

- The payload mass play a big role in the success rate of the mission, this can change with différents conditions but in general this a key attribute to analyse.

- Orbits have very diversify success rate, the most successful one are GTO, ISS, VLEO.

- ALL tools use in this analyse provide better understanding of raw data from SpaceX, to go a step further it will be necessary to have more data to really draw pattern that happen with the launch of the Falcon 9.

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!