IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Tan Dyi Perng>
<30/1/2022>

Outline

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

Executive Summary

- Summary of methodologies
  - Data collection
  - Data wrangling
  - EDA with data visualisation
  - EDA with SQL
  - Building an interactive map with Folium
  - Building a dashboard with Plotly Dash
  - Predictive analysis with machine learning
- Summary of all results
  - Exploratory data analysis
  - Interactive analytic demo
  - Predictive analysis

Introduction

- Project background and context

  In this project we will need to predict if Falcon 9 first stage will land successfully. SpaceX advertises that it will cost much lower than other provider because it can reuse the first stage. Therefore, if the landing successful rate can be determined, the cost of the launch can be determined as well.

- Problems you want to find answers

  - The factors that affect the success rate for the rocket landing
  - Relationship between the rocket variables that impact the success rate for rocket landing

Section 1

Methodology

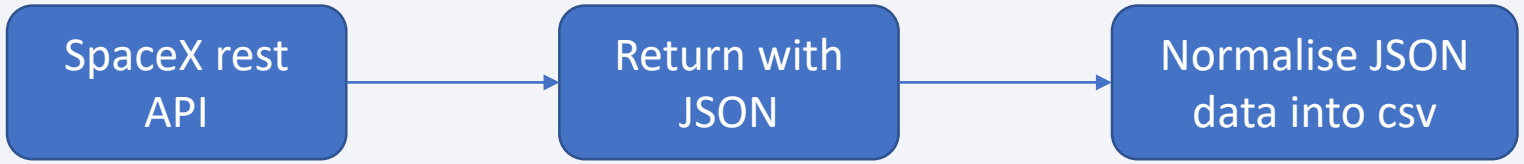
Methodology

Executive Summary

- Data collection methodology:
  - SpaceX Rest API and web scraping from Wikipedia
- Perform data wrangling
  - Remove irrelevant columns and use one hot encoding for data fields that will be used in machine learning
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

- Describe how data sets were collected.
  - Gather the data from SpaceX rest API
  - API will give data such as launches, information about rocket used, payload delivered, launch specifications, landing specifications and landing outcome
  - Use these data to predict the success rate for SpaceX to land
- You need to present your data collection process use key phrases and flowcharts



Data Collection – SpaceX API

1. Getting response from API

Task 1: Request and parse the SpaceX launch data using the GET request

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

```
In [9]: static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

2. Convert response to .json file

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

```
In [13]: # Use json_normalize meethod to convert the json result into a dataframe
         data = pd.json_normalize(response.json())
```

3. Apply custom functions

```
In [20]: # Call getLaunchSite
         getLaunchSite(data)

In [21]: # Call getPayloadData
         getPayloadData(data)

In [22]: # Call getCoreData
         getCoreData(data)
```

4. Assign list to dictionary

```
In [23]: 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}
```

5. Create data frame from dictionary

```
In [24]: # Create a data from launch_dict
         df = pd.DataFrame.from_dict(launch_dict)
```

[Data collection github link]

Data Collection - Scraping

1. Request Falcon9 launch Wiki page from URL

```
In [5]: # use requests.get() method with the provided static_url
        # assign the response to a object
        data  = requests.get(static_url).text
```

Create a BeautifulSoup object from the HTML response

```
In [6]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
        soup = BeautifulSoup(data, 'html5lib')
```

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

```
In [7]: # Use soup.title attribute
        print(soup.title)
```

```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

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

```
In [8]: # 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')
```

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

```
In [9]: # Let's print the third table and check its content
        first_launch_table = html_tables[2]
        print(first_launch_table)
```

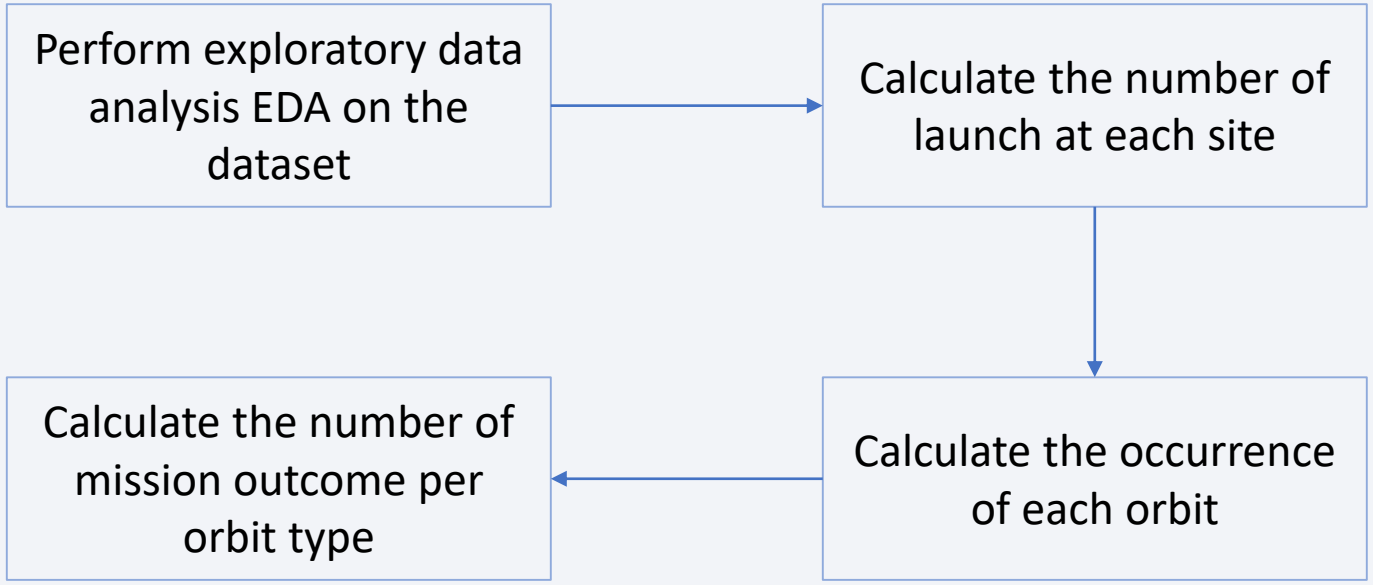
3. Create a data frame by parsing the launch HTML tables

```
In [26]: 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']=[]
```

```
In [27]: 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.`
                     launch_dict['Flight No.'].append(flight_number) #TODO-1
                     #print(flight_number)
                     datatimelist=date_time(row[0])

                     # Date value
                     # TODO: Append the date into launch_dict with key `Date`
                     date = datatimelist[0].strip(',')
                     launch_dict['Date'].append(date) #TODO-2
                     #print(date)
```

[Data Scraping github link]

Data Wrangling



[data wrangling github link]

EDA with Data Visualization

- Scatter Plots
  - Flight Number vs Payload Mass
  - Flight Number vs Launch Site
  - Payload vs Launch Site
  - Orbit vs Flight Number
- Bar Graph
  - Mean vs Orbit
- Line Graph
  - Success Rate vs Year

Data visualization github link

EDA with SQL

SQL queries performed:

- Display the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string ‘CCA’
- Display the total payload mass carried by boosters launched by NASA (CRS)
- Display average payload mass carried by booster version F9 v1.1
- List the date where the successful landing outcome in drone ship was achieved
- List the names of the boosters which have success in “drone ship” and have payload between 4000 and 6000kg
- List the total number of successful and failure mission outcomes
- List the names of the booster versions 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
- 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

EDA with SQL github link

Build an Interactive Map with Folium

- To visualize the Launch Data into an interactive map, the Latitude and Longitude Coordinates at each launch site was used and added a Circle Marker around each launch site with a label of the name of the launch site.
- To visualize the success marker more clearly, we assigned the data frame launch outcomes(failures, successes) to 0 and 1 with Green and Red markers on the map in a MarkerCluster()
- To visualize the distance from the site, we used Haversine's formula and calculated the distance from the Launch Site to various landmarks to find various trends about what is around the Launch Site to measure patterns. Lines are drawn on the map to measure distance to landmarks

Interactive map with folium github link

Build a Dashboard with Plotly Dash

- Dashboard Graph
  - Pie Chart of success count for all launch sites
  - Pie Chart of each site’s total success launches
  - Plot graph of success counts on payload mass for all sites
  - Plot graph of success counts on payload mass for each site

plotly dash github link

Predictive Analysis (Classification)

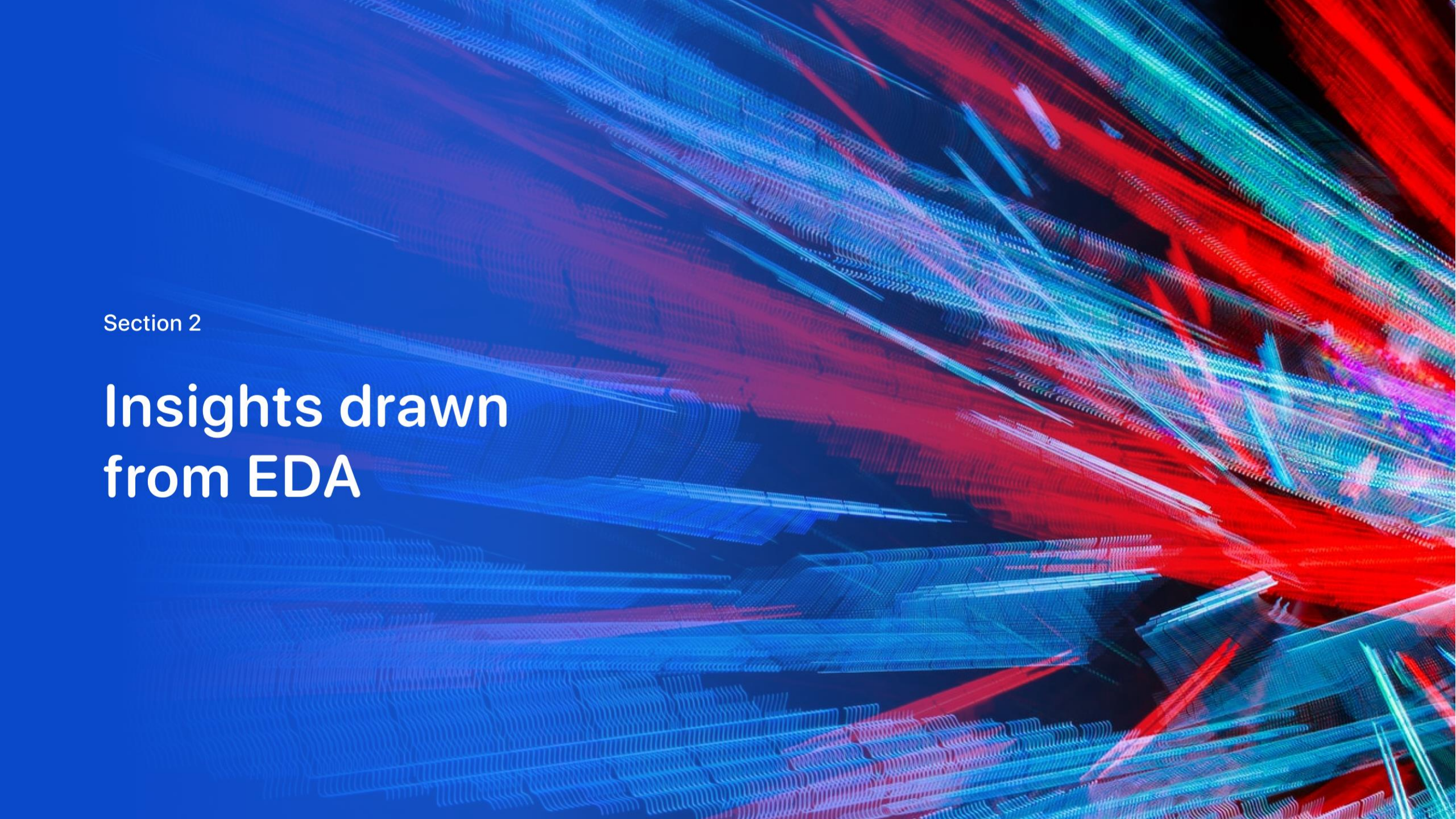
- BUILDING MODEL
  - Load dataset into NumPy and Pandas
  - Transform Data
  - Split the data into training and test data sets
  - Check the number of test samples
  - Decide the type of machine learning algorithms
  - Set the parameters and algorithms to GridSearchCV
  - Fit the datasets into the GridSearchCV objects and train the dataset
- EVALUATING MODEL
  - Check the accuracy for each model
  - Tune hyperparameters for each algorithms
  - Plot the Confusion Matrix
- IMPROVING MODEL
  - Feature Engineering
  - Algorithm Tuning
- FINDING THE BEST PERFORMING CLASSIFICATION MODEL
  - The model with the best accuracy score wins the best performing model

Results

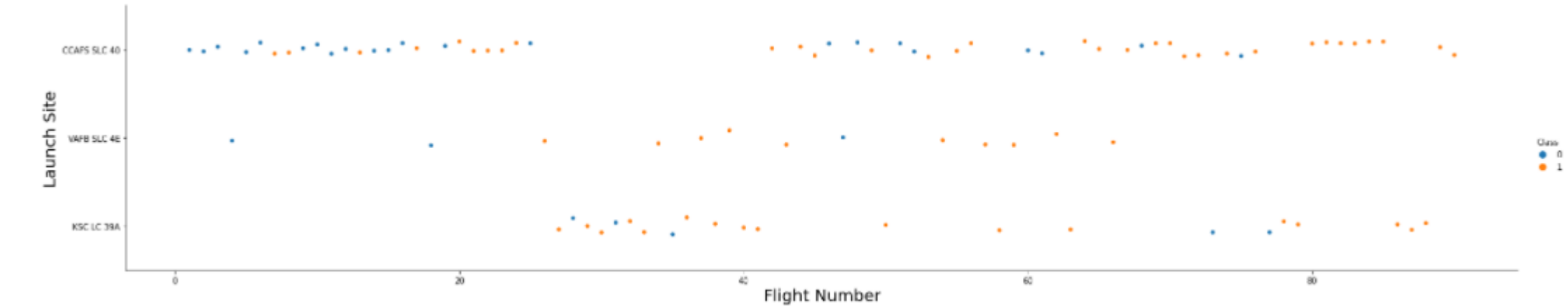
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

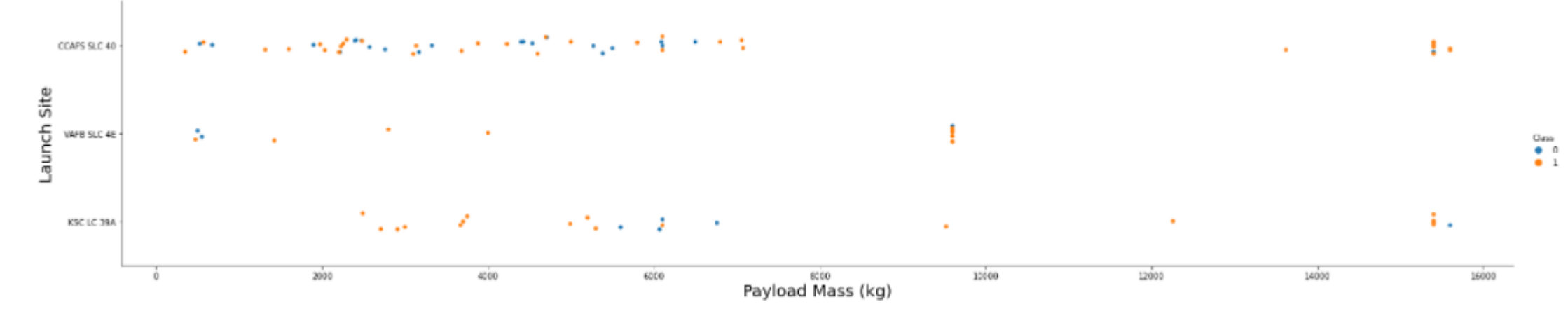


Flight Number vs. Launch Site



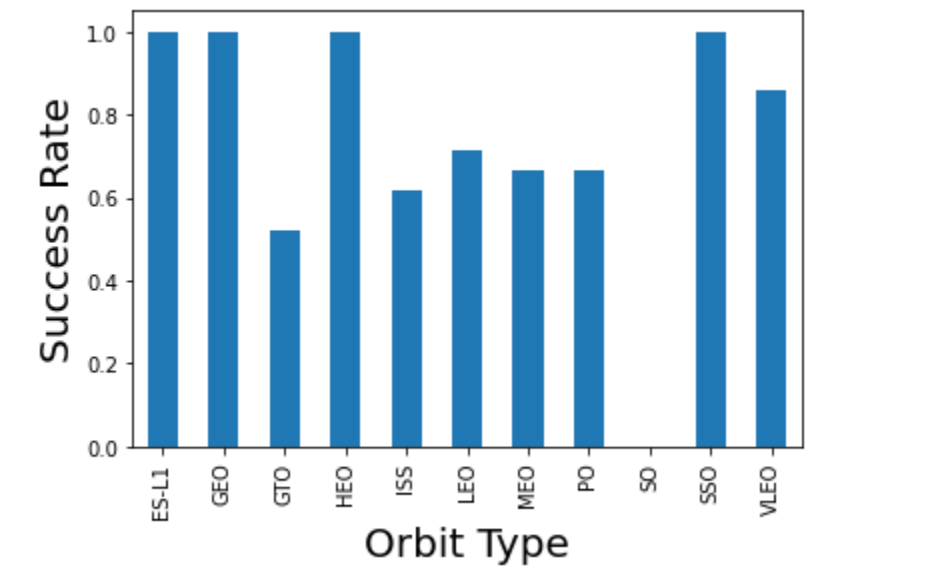
The success rate increased when there is more amount of flights at a launch site

Payload vs. Launch Site



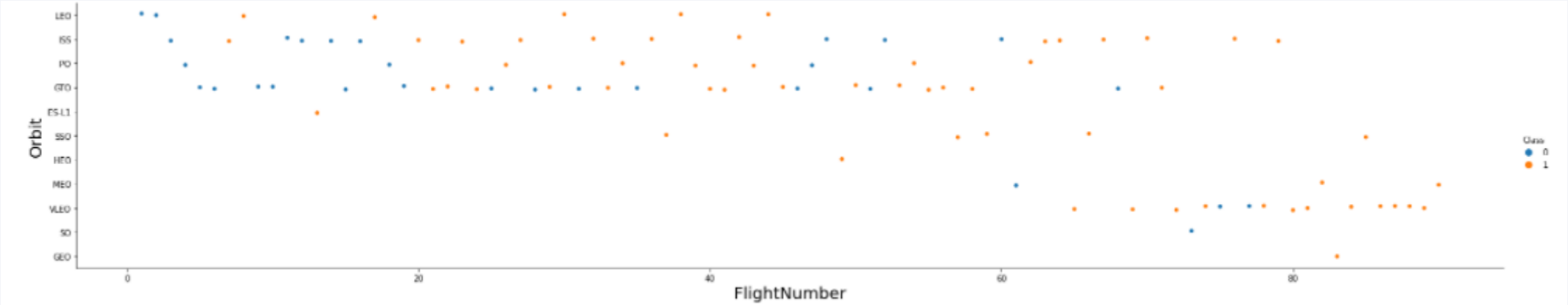
The greater the payload mass for Launch Site CCAFS SLC 40 the higher the success rate for the Rocket.
There is no clear pattern using this visualization to make a decision if the Launch Site is dependent on Pay Load Mass for a success launch.

Success Rate vs. Orbit Type



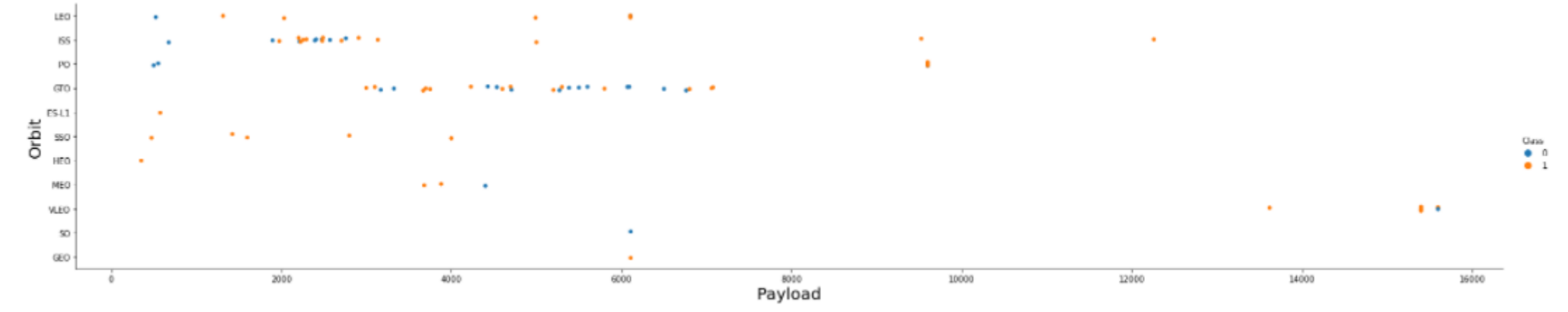
Orbit GEO,HEO,SSO,ES-L1 has the highest success rate

Flight Number vs. Orbit Type



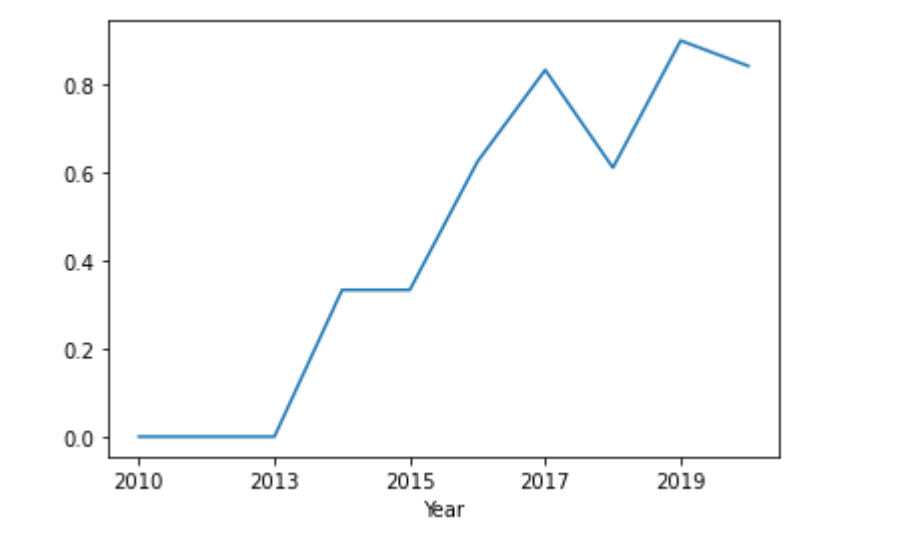
For LEO orbit the success rate appears to be related to the number of flights

Payload vs. Orbit Type



With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS. For GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.

Launch Success Yearly Trend



The success rate since 2013 kept increasing till 2020

All Launch Site Names

Out[11]:

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

- select DISTINCT Launch_Site from SPACEXTBL
- QUERY EXPLAINATION
  - Using DISTINCT in the query means that it will only show Unique values in the Launch_Site column from SPACEXTBL

Launch Site Names Begin with 'CCA'

Out[24]:

launch_site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

- Select TOP 5 from SPACEXTBL WHERE Launch_Site LIKE ‘CCA%’
- QUERY EXPLAINATION
  - Using the word TOP 5 in the query to show 5 records from SPACEXTBL and LIKE keyword has a wild card with the words. The percentage in the end suggests that the Launch_Site name must start with CCA.

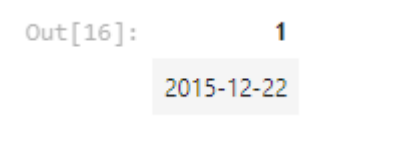
Total Payload Mass

- select SUM(PAYLOAD_MASS__KG_) from SPACEXTBL where LAUNCH_SITE LIKE 'CCA%‘
- QUERY EXPLAINATION
  - Using the function SUM to calculate the total mass in the column PAYLOAD_MASS_KG
  - The WHERE clause filters the dataset to only perform calculations on launch site name is CCA

Average Payload Mass by F9 v1.1

- select AVG(PAYLOAD_MASS__KG_) from SPACEXTBL where BOOSTER_VERSION LIKE 'F9 v1.1‘
- QUERY EXPLAINATION
  - Using the function AVG to calculate the average in the column PAYLOAD_MASS_KG_
  - The WHERE clause filters the dataset to only perform calculations on Booster_version F9 v1.1

First Successful Ground Landing Date



- Select MIN(Date) from SPACEXTBL where Landing_Outcome ==”Success (drone ship)”
- QUERY EXPLAINATION
  - Using the function MIN to calculate the minimum date in the column Date
  - The WHERE clause filters the dataset to perform calculations on Landing_Outcome Success (drone ship) only

Successful Drone Ship Landing with Payload between 4000 and 6000

Out[18]:

booster_version	payload_mass__kg_
F9 FT B1022	4696
F9 FT B1026	4600
F9 FT B1021.2	5300
F9 FT B1031.2	5200

- select BOOSTER_VERSION, PAYLOAD_MASS__KG_ from SPACEXTBL where (LANDING__OUTCOME LIKE 'Success (drone ship)') AND (PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000)QUERY EXPLAINATION
  - Selecting only Booster_Version
  - The WHERE clause filters the dataset to Landing_Outcome = Success (drone ship)
  - The AND clause specifies additional filter conditions Payload_MASS_KG 4000 AND Payload_MASS_KG 6000

Total Number of Successful and Failure Mission Outcomes

- SELECT(SELECT Count(Mission_Outcome from SPACEXTBL where Mission_Outcome LIKE '%Success%') as Successful_Mission_Outcomes (SELECT Count(Mission_Outcome from SPACEXTBL where Mission_Outcome LIKE ''%Failure%') as Failure_Mission _Coutcomes
- QUERY EXPLAINATION
  - Using multiple wildcard filter the criteria of the mission outcome so that only successful and failure mission outcomes are selected

Boosters Carried Maximum Payload

Out[20]:

booster_version	payload_mass__kg_
F9 B5 B1048.4	15600
F9 B5 B1048.5	15600
F9 B5 B1049.4	15600
F9 B5 B1049.5	15600
F9 B5 B1049.7	15600
F9 B5 B1051.3	15600
F9 B5 B1051.4	15600
F9 B5 B1051.6	15600
F9 B5 B1056.4	15600
F9 B5 B1058.3	15600
F9 B5 B1060.2	15600
F9 B5 B1060.3	15600

- SELECT DISTINCT(BOOSTER_VERSION), PAYLOAD_MASS__KG_ FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL)
- QUERY EXPLAINATION
  - DISTINCT in the query will only show Unique values in the Booster_Version column from SPACEXTBL GROUP BY puts the list in order set to a certain condition

2015 Launch Records

Out[21]:

booster_version	launch_site	DATE	landing__outcome
F9 v1.1 B1012	CCAFS LC-40	2015-01-10	Failure (drone ship)
F9 v1.1 B1015	CCAFS LC-40	2015-04-14	Failure (drone ship)

- SELECT BOOSTER_VERSION, LAUNCH_SITE, DATE, LANDING__OUTCOME FROM SPACEXTBL WHERE LANDING__OUTCOME LIKE 'Failure (drone ship)' AND YEAR(DATE)=2015
- QUERY EXPLAINATION
  - Using AND function to limit the year to 2015 and present the data only for 2015

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

Out[22]:

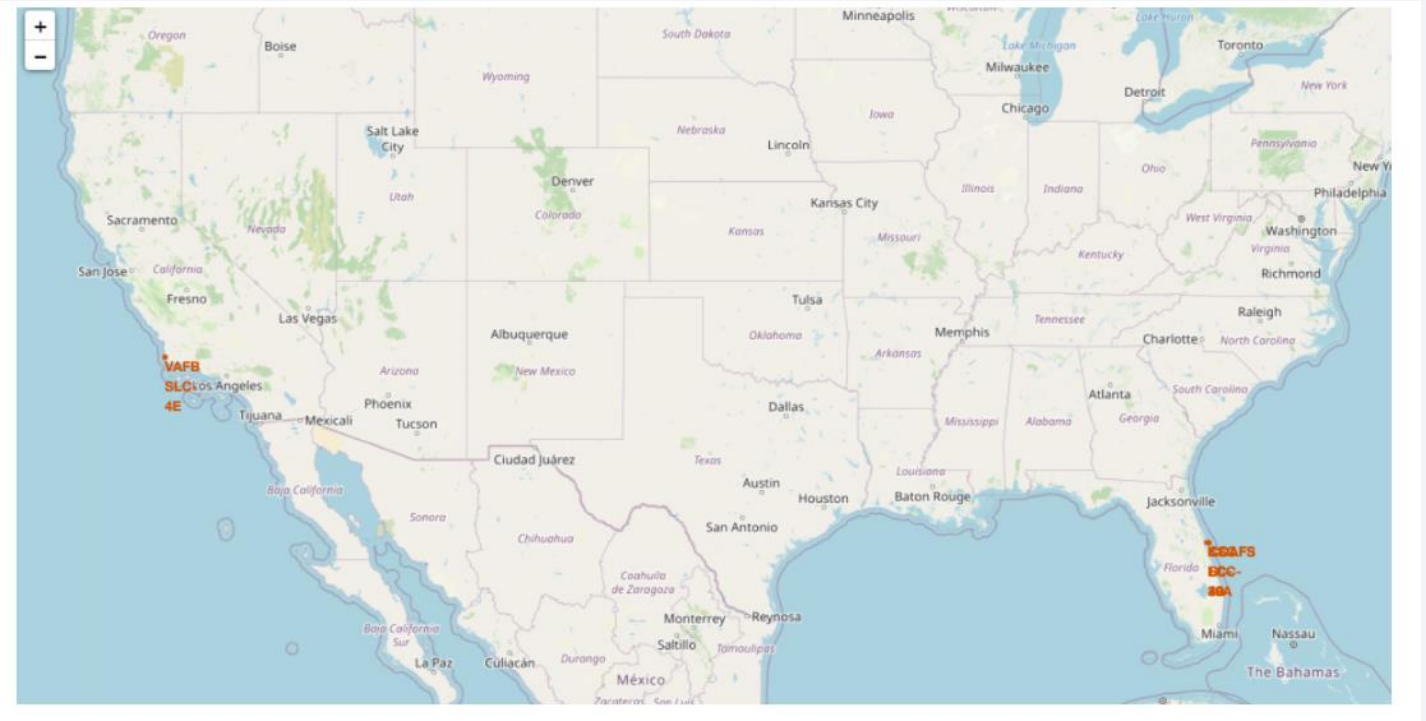
landing_outcome	COUNT
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Failure (parachute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

- SELECT landing__outcome, COUNT(*) AS count FROM SPACEXTBL WHERE DATE BETWEEN '2010-06-04'and'2017-03-20‘ GROUP BY landing__outcome ORDER BY COUNT(landing__outcome) DESC
- QUERY EXPLAINATION
  - DESC function will order the data in descending order

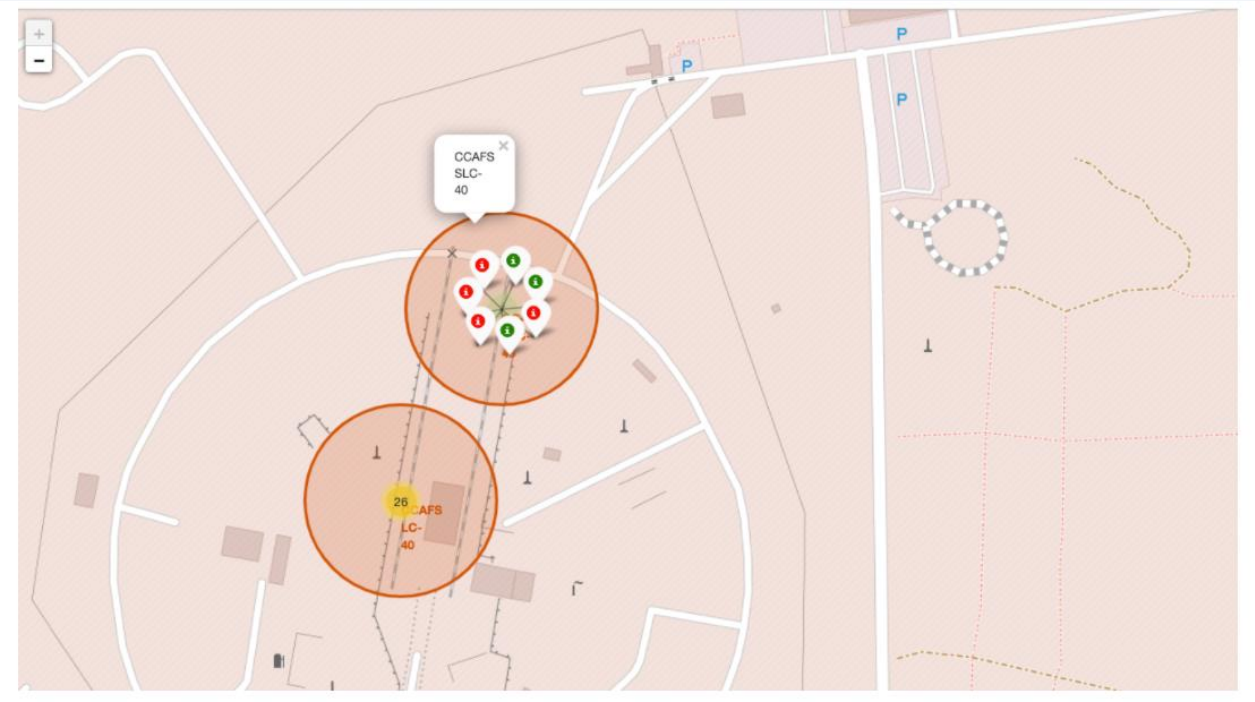
Section 4

Launch Sites Proximities Analysis

<Folium Map Screenshot 1>



<Folium Map Screenshot 2>



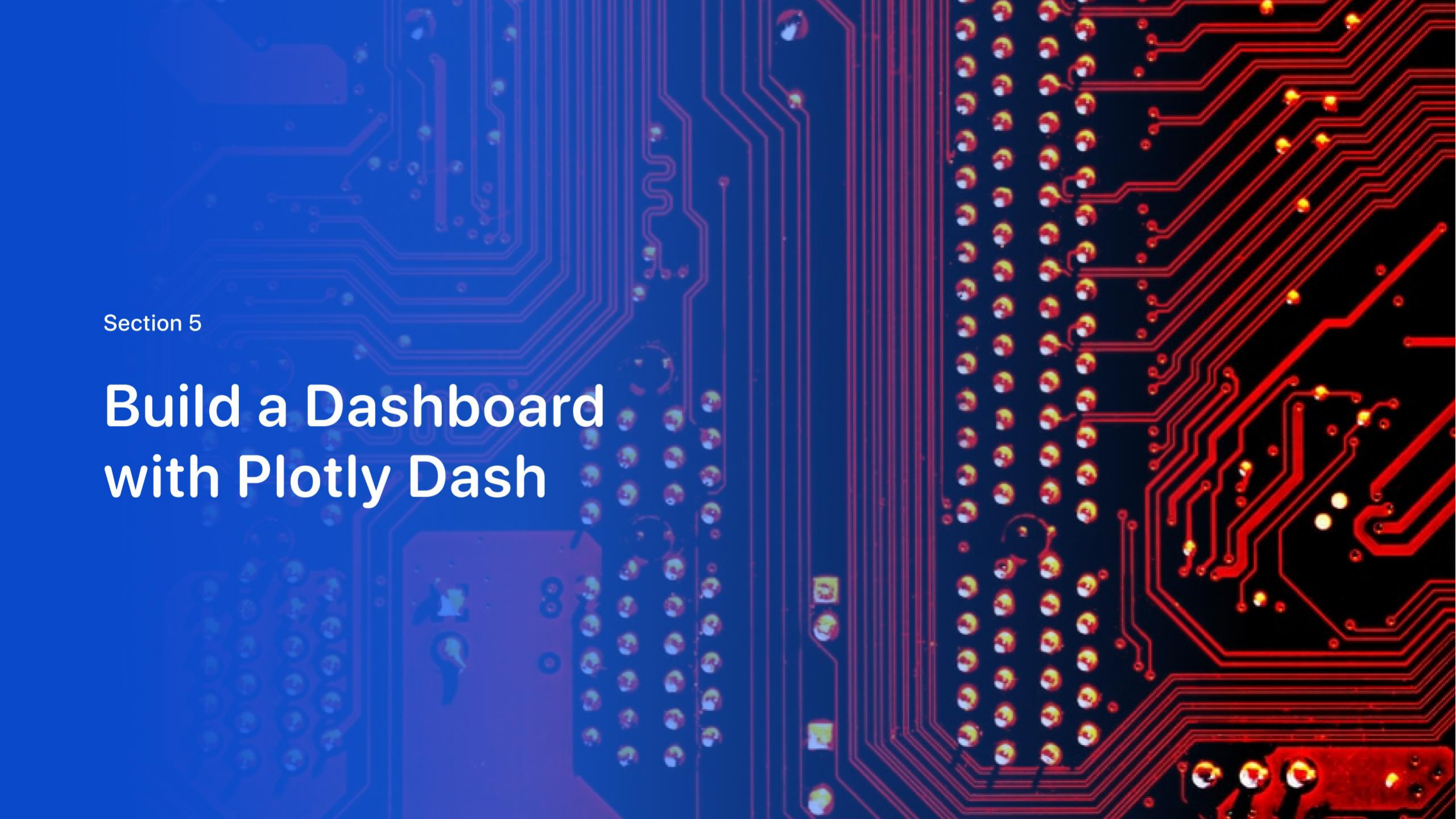
From the color-labeled markers in marker clusters, you should be able to easily identify which launch sites have relatively high success rates

<Folium Map Screenshot 3>

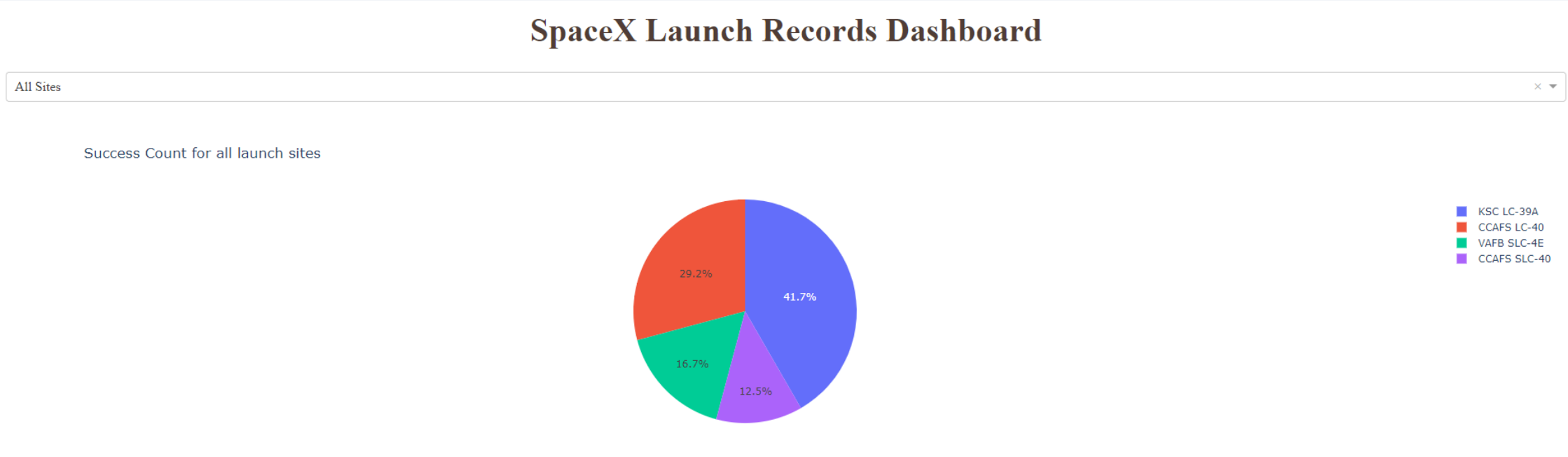
Screenshot showing distances from the CCAFS LC-40 to the coastline (0.88 km) to the east and a railway (1.37 km) to the northwest.

Section 5

Build a Dashboard with Plotly Dash

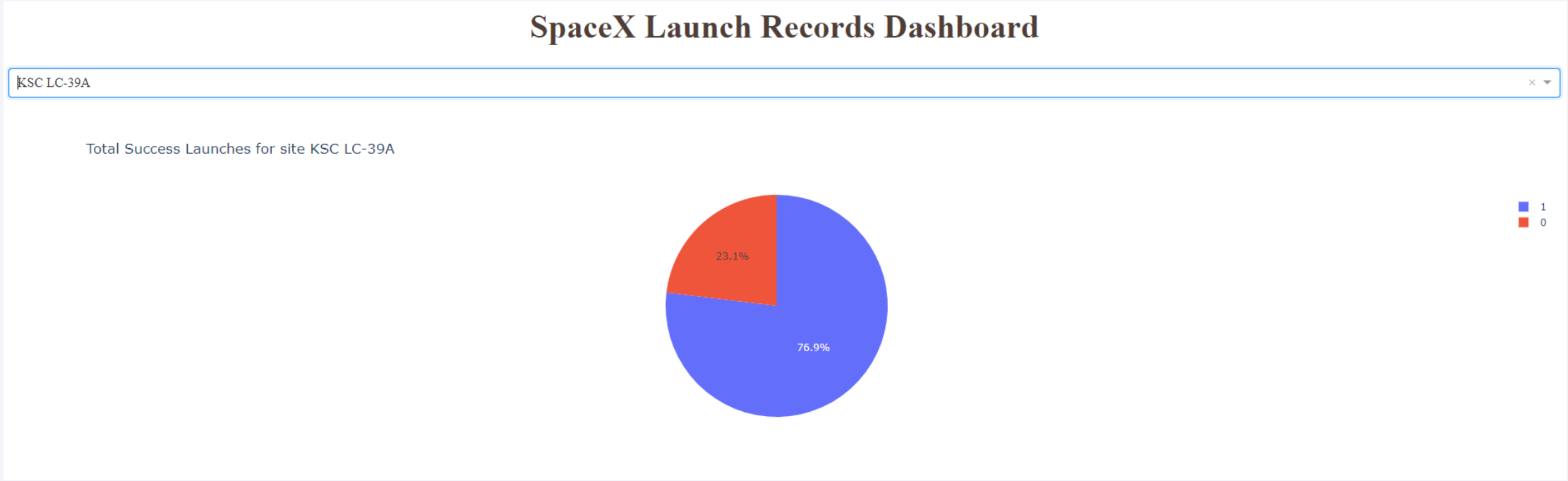


<Launch success count for all sites>



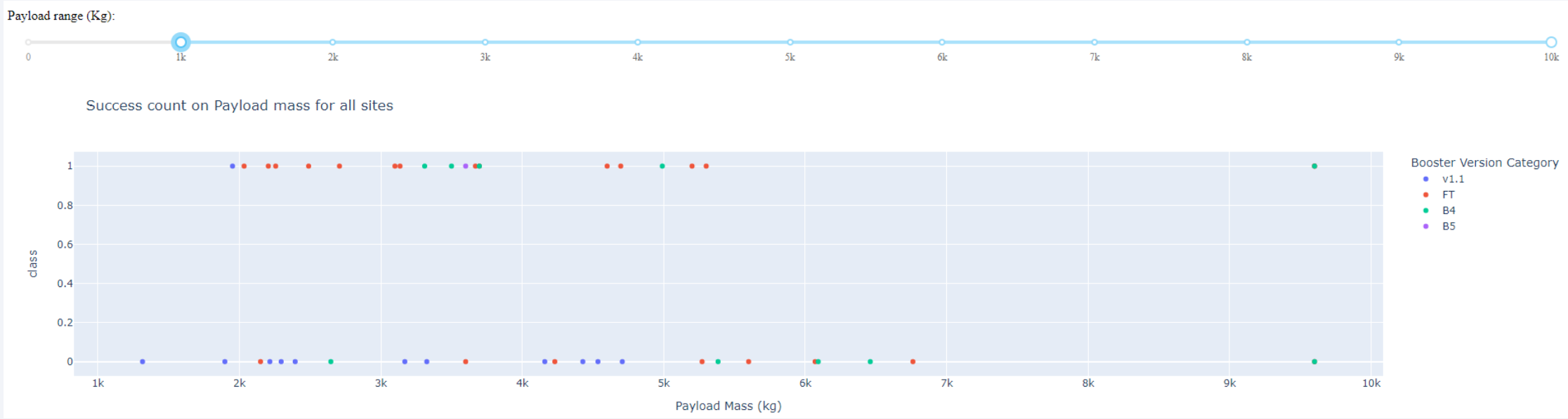
The pie chart shows the success count for all launch sites. From the pie chart it clearly shows that for KSC LC-39A launch site has the highest success rate

<launch site with highest launch success ratio>



The pie chart shows that the success launches for site KSC LC-39A is 76.9%

<Payload vs launch outcome scatter plot>

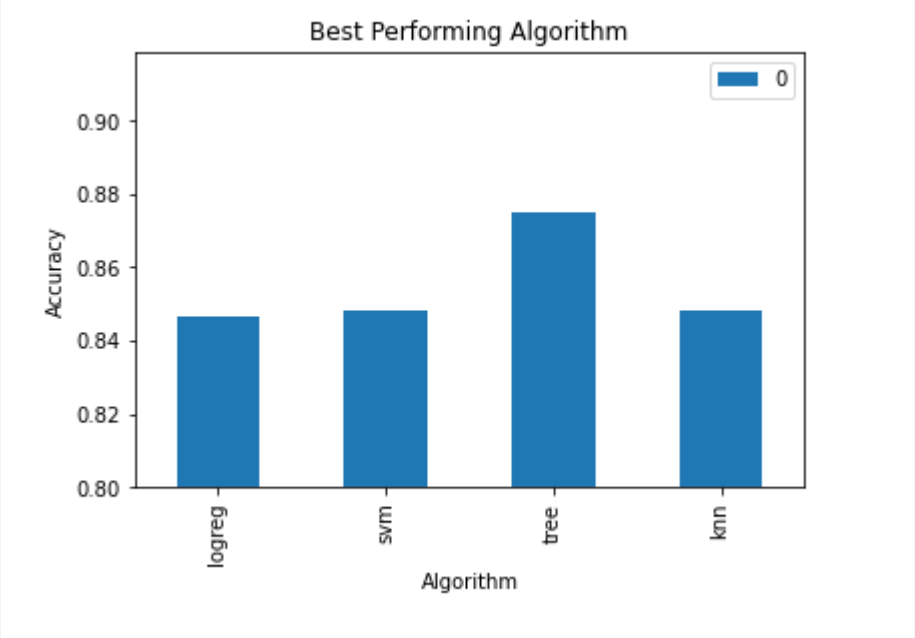


The payload range is adjustable so that we can filter the range with the highest success count

Section 6

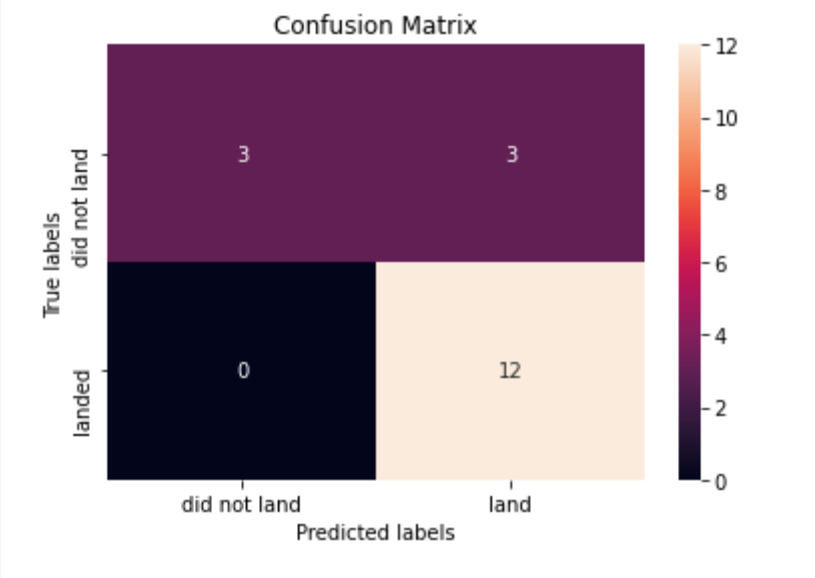
Predictive Analysis (Classification)

Classification Accuracy



From the bar chart, we can see that tree algorithm perform the best in terms of accuracy

Confusion Matrix



From the confusion matrix, we can see that Tree can distinguish between the different classes

Conclusions

- The Tree Classifier Algorithm perform the best for this dataset
- Low weighted payloads perform better than the heavier payloads
- The success rates for SpaceX launches is directly proportional to time in years
- KSC LC 39A had the most successful launches from all the sites
- Orbit GEO,HEO,SSO,ES L1 has the best Success Rate

Thank you!