IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Ankita Anuradha
15 August 2024

Outline

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

Executive Summary

- **Summary of methodologies**
  - Data Collection through API
  - Data Collection with Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis with Data Visualization
  - Interactive Visual Analytics with Folium
  - Machine Learning Prediction
- **Summary of all results**
  - Exploratory Data Analysis result
  - Interactive analytics in screenshots
  - Predictive Analytics result

Introduction

- Project background and context

Space X advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because Space X can reuse the first stage. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against space X for a rocket launch. This goal of the project is to create a machine learning pipeline to predict if the first stage will land successfully.

- Problems you want to find answers

We need to find the various parameters responsible for the successful launch of the rocket and what needs to be done for a successful landing

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - We collected the data using SpaceX API and web scraping from Wikipedia.
- Perform data wrangling
  - For this one-hot encoding was applied to the categorical features
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Here we need to build, tune, evaluate classification models

Data Collection

- Data collection was done by using get request to the SpaceX API.
- We then decoded the response content as a Json using .json() function call and turn it into a pandas dataframe using .json_normalize().
- The data was cleaned, it was checked for missing values and filled when required
- We performed web scraping from Wikipedia for Falcon 9 launch records using BeautifulSoup.

The objective was to extract the launch records as HTML table and convert it to a pandas dataframe for future analysis.

Data Collection – SpaceX API

- Here we collect the data from the SpaceX REST API and clean the dataset to further work on it

- Here's the github url: https://github.com/Ankita510/IBM_CAPSTONEfinal/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

```python
# Takes the dataset and uses the cores column to call the API and append the data to the lists
def getCoreData(data):
    for core in data['cores']:
            if core['core'] != None:
                response = requests.get("https://api.spacexdata.com/v4/cores/"+core['core']).json()
                Block.append(response['block'])
                ReusedCount.append(response['reuse_count'])
                Serial.append(response['serial'])
            else:
                Block.append(None)
                ReusedCount.append(None)
                Serial.append(None)
            Outcome.append(str(core['landing_success'])+' '+str(core['landing_type']))
            Flights.append(core['flight'])
            GridFins.append(core['gridfins'])
            Reused.append(core['reused'])
            Legs.append(core['legs'])
            LandingPad.append(core['landpad'])
```

Python

Data Collection - Scraping

- We collected the data from the Wikipedia page of falcon9 launches, parsed it and made into a dataframe

- Here's the url:

https://github.com/Ankita510/IBM_CAPSTONEfinal/blob/main/jupyter-labs-webscraping.ipynb

```
def date_time(table_cells):
    """
    This function returns the data and time from the HTML  table cell
    Input: the  element of a table data cell extracts extra row
    """
    return [data_time.strip() for data_time in list(table_cells.strings)][0:2]

def booster_version(table_cells):
    """
    This function returns the booster version from the HTML  table cell
    Input: the  element of a table data cell extracts extra row
    """
    out=''.join([booster_version for i,booster_version in enumerate( table_cells.strings) if i%2==0][0:-1])
    return out

def landing_status(table_cells):
    """
    This function returns the landing status from the HTML table cell
    Input: the  element of a table data cell extracts extra row
    """
    out=[i for i in table_cells.strings][0]
    return out


def get_mass(table_cells):
    mass=unicodedata.normalize("NFKD", table_cells.text).strip()
    if mass:
        mass.find("kg")
        new_mass=mass[0:mass.find("kg")+2]
    else:
        new_mass=0
    return new_mass


def extract_column_from_header(row):
    """
    This function returns the landing status from the HTML table cell
    Input: the  element of a table data cell extracts extra row
    """
    if (row.br):
        row.br.extract()
    if row.a:
        row.a.extract()
    if row.sup:
        row.sup.extract()

    colunm_name = ' '.join(row.contents)

    # Filter the digit and empty names
    if not(colunm_name.strip().isdigit()):
        colunm_name = colunm_name.strip()
```

Data Wrangling

- The missing values were dealt and replaced by the mean of payload mass
- The url for the particular task is as follows:

https://github.com/Ankita510/IBM_CAPSTONEfinal/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

- We used scatterplot as well as the bar graph in order to understand the relationship between different parameters and success of launch.

Here's the github url:

https://github.com/Ankita510/IBM_CAPSTONEfinal/blob/main/edadataviz.ipynb

EDA with SQL

- First, we downloaded the dataset and connected the database
- We found out the unique launch sites used and the categorized them by their payloads
- We found the success and failure rates of the said launch sites
- The success rates were compared to the payloads as well
- Total payload and average payload were also calculated
- https://github.com/Ankita510/IBM_CAPSTONEfinal/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- To visualize the data in a better way, we used Folium as it engages the client and gives them a proper picture of the scenario.
- Circles and marker were used to specify the different lauch sites which have been successful in the past and they also specified the region.
- We could visualize the distance from railways and highways quite easily through this.
- https://github.com/Ankita510/IBM_CAPSTONEfinal/blob/main/edadataviz.ipynb

  Visit for more details

Build a Dashboard with Plotly Dash

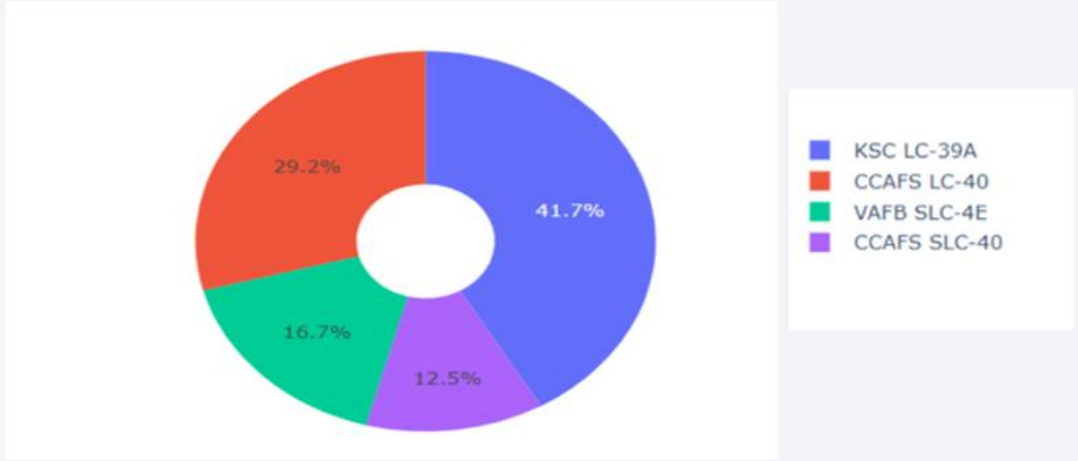
- We used dashdowns to make the dashboard more interactive
- For the given data, we used pie charts to compare between the given launch sites and the success rate for each one of them
- The plots were used to visualize the results of the given data in a better manner.
- https://github.com/Ankita510/IBM_CAPSTONEfinal/blob/main/spacex_dash_app

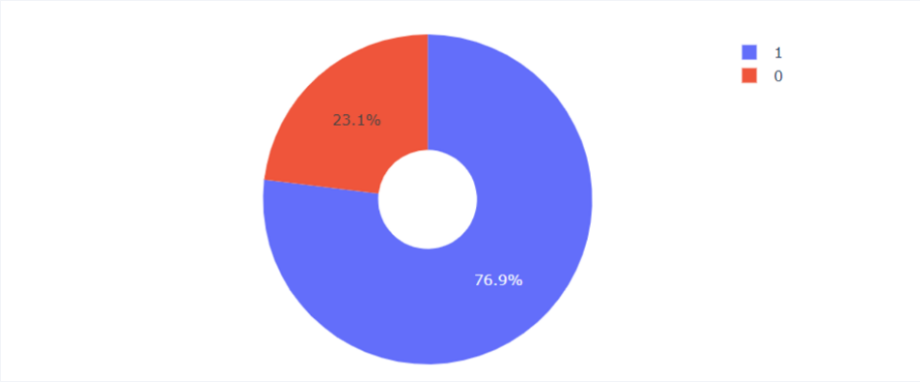
Predictive Analysis (Classification)

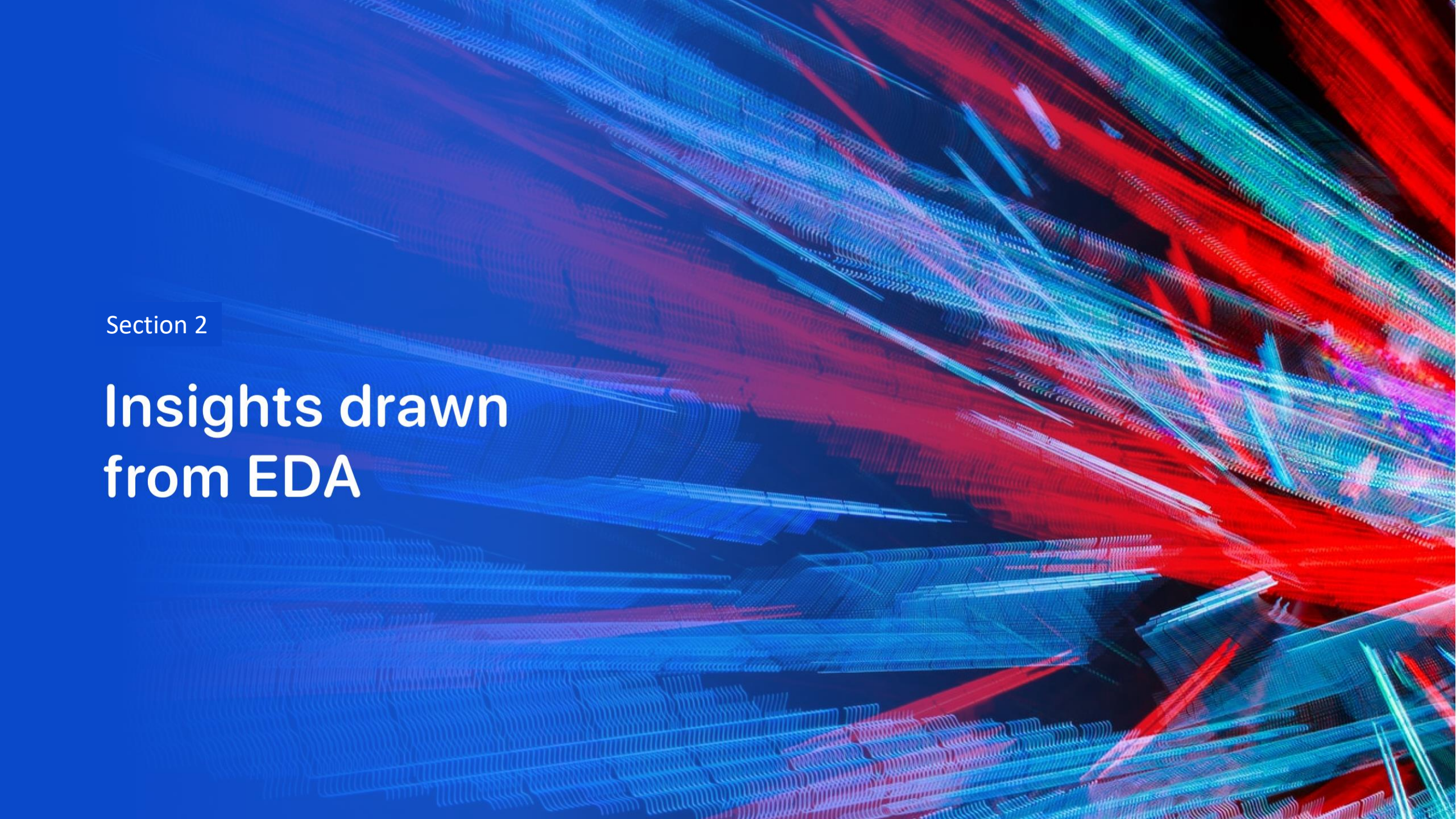
- Different models were used such as SVM, KNN, Logistic Regression, Linear Regression as well as Decision Trees
- The accuracy of each of these models were calculated as well as a confusion matrix for each was displayed
- We found that models like Decision tree and KNN are best suited for analysis here
- At the end we also displayed the accuracy of each model using bar graph
- https://github.com/Ankita510/IBM_CAPSTONEfinal/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Results

- Exploratory data analysis results: We found the various launch sites and the successs rates. Other parameters required for a successful launch were also determined
- Predictive analysis results: The decision tree tell us that we need to use KSC LC 39A Launch iste with low weighted payload for a successful launch

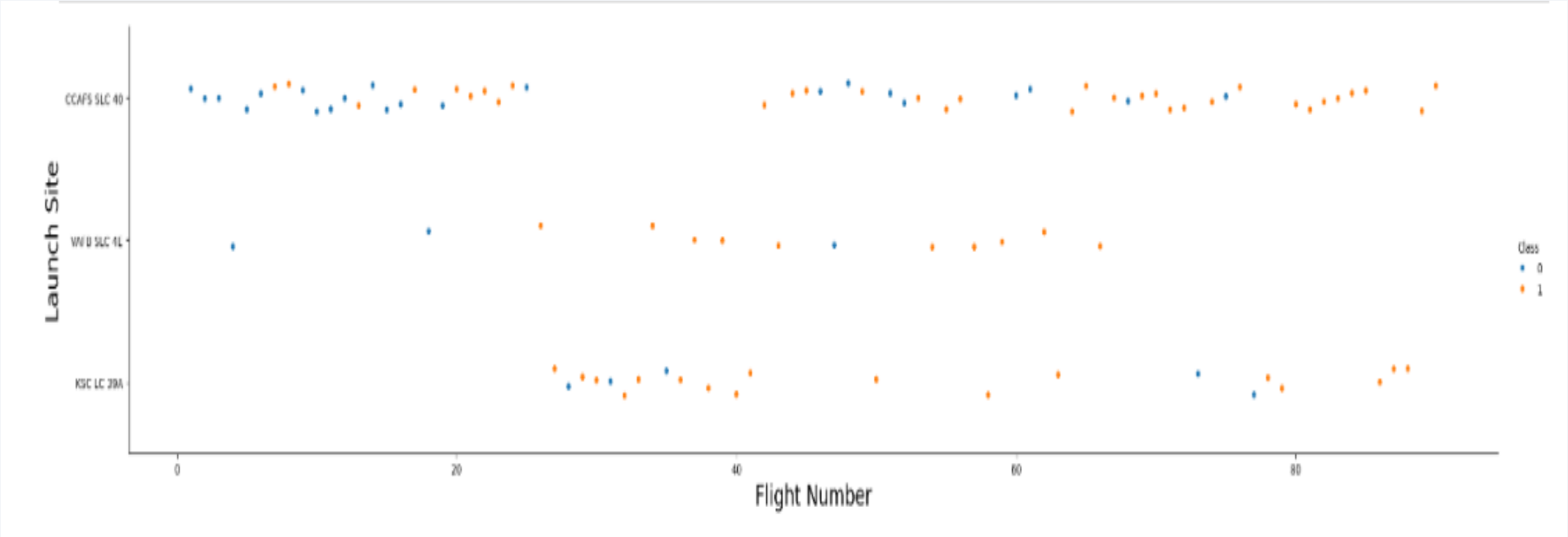





Section 2
Insights drawn from EDA

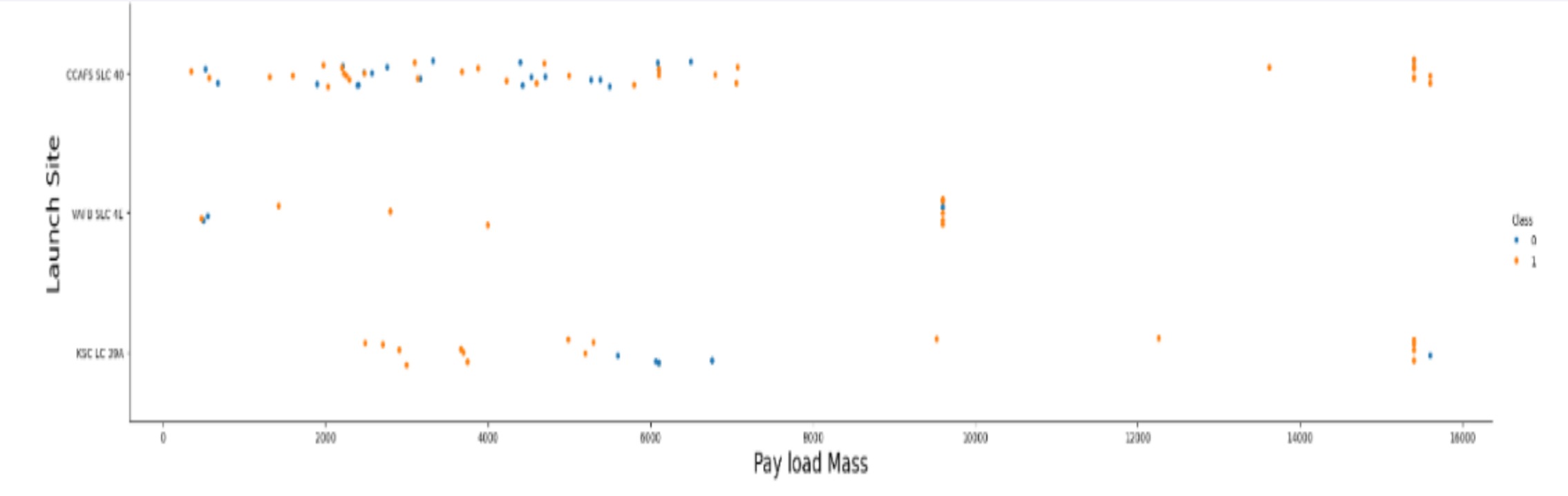
Flight Number vs. Launch Site

Here we can see the scatterplot attained to determine the relationship between flight number and launch site

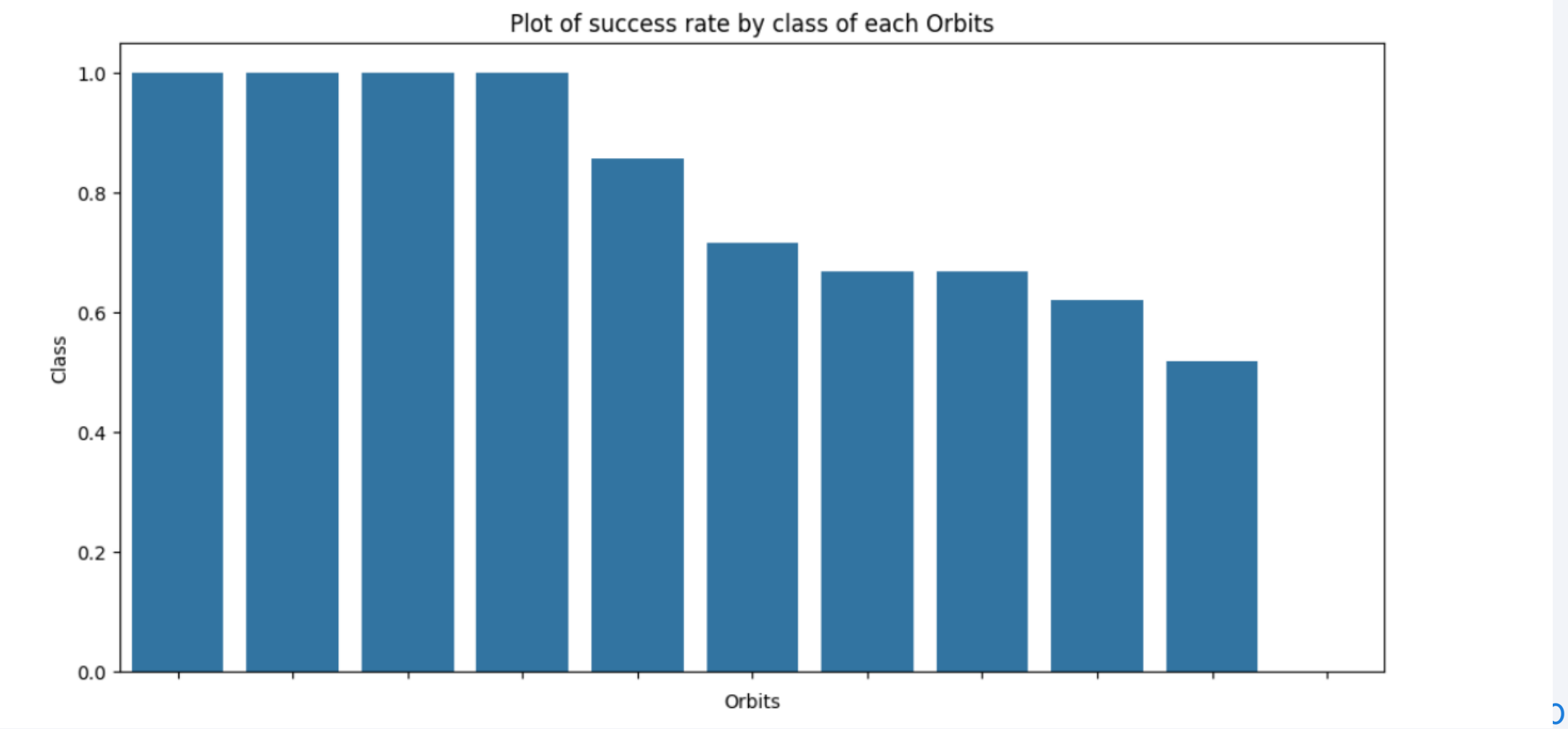


Payload vs. Launch Site

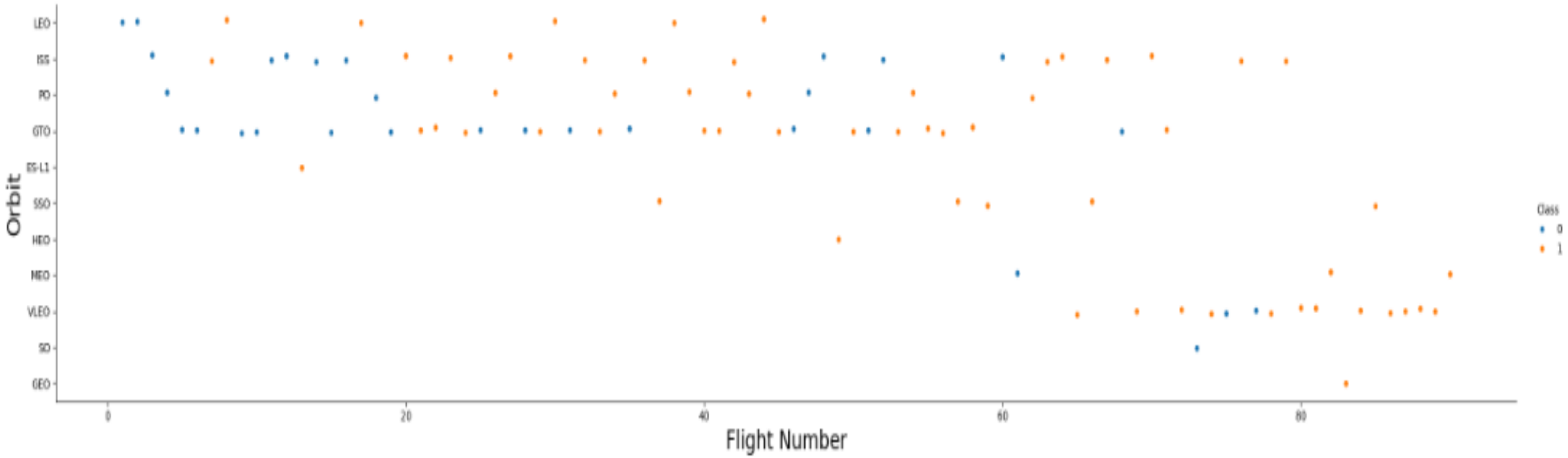
Scatter plot for Payload and Launch Site



Success Rate vs. Orbit Type

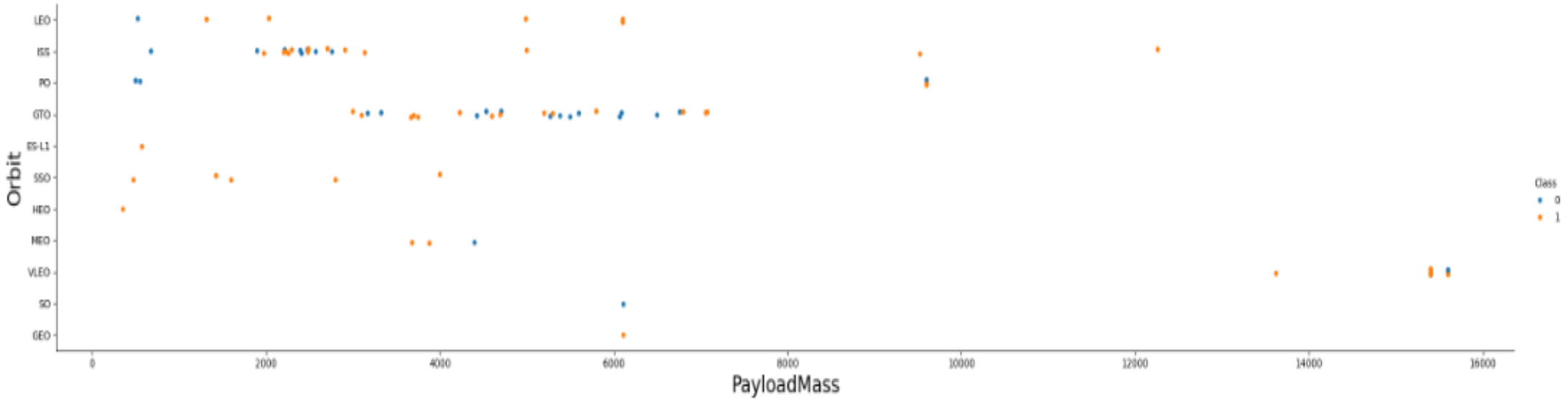


Flight Number vs. Orbit Type



You can observe that in the LEO orbit, success seems to be related to the number of flights. Conversely, in the GTO orbit, there appears to be no relationship between flight number and success.

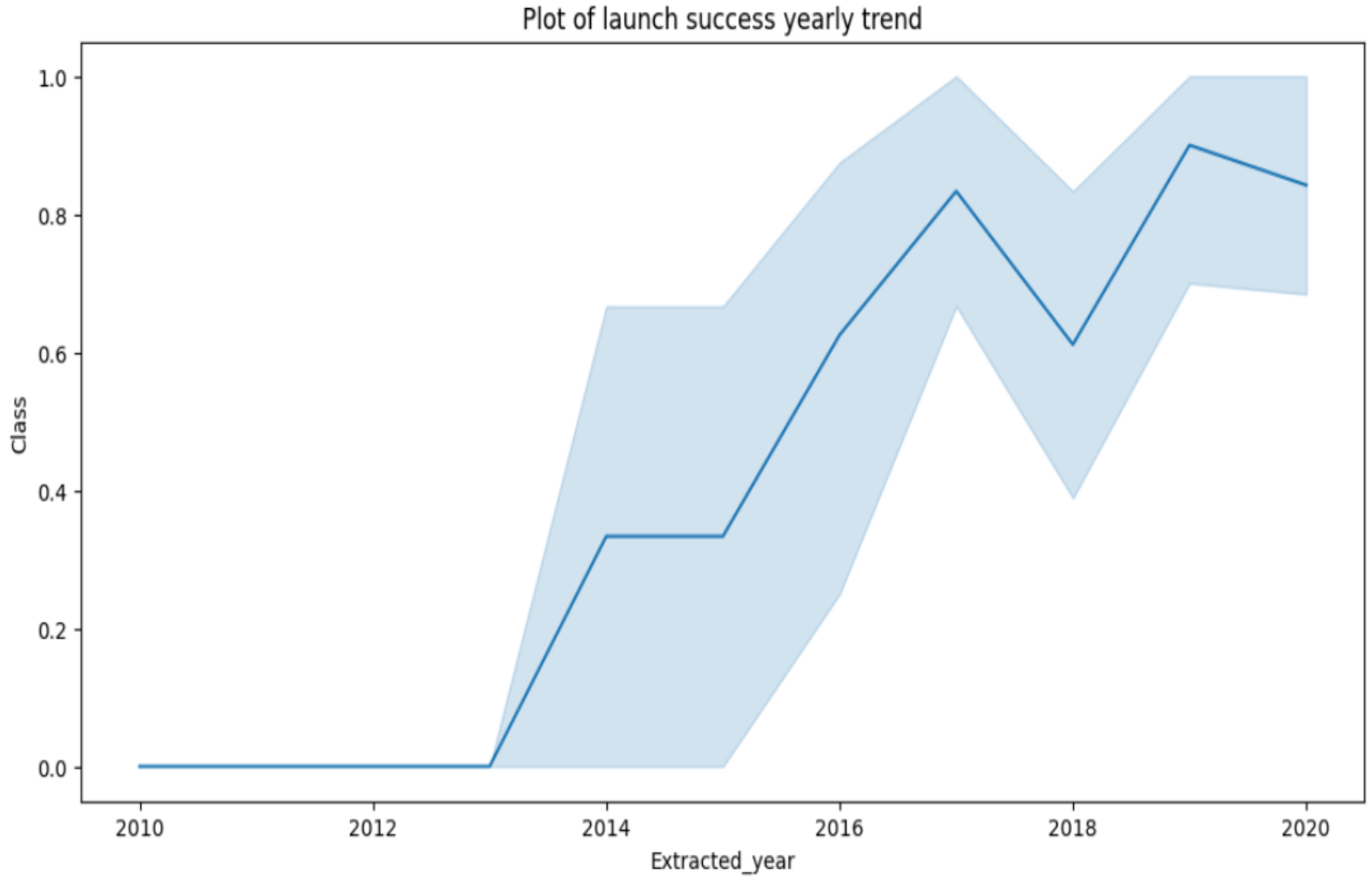
Payload vs. Orbit Type



With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

However, for GTO, it's difficult to distinguish between successful and unsuccessful landings as both outcomes are present.

Launch Success Yearly Trend



you can observe that the sucess rate since 2013 kept increasing till 2020

All Launch Site Names

- The Launching sites obtained were:

KSC LC-39A

CCA FS LC-40

CCA FS SLC-40

VA FB SLC-4E

Launch Site Names Begin with 'CCA'

	date	time	boosterversion	launchsite	payload	payloadmasskg	orbit	customer	missionoutcome
0	2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success
1	2010-08-12	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
2	2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success
3	2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success
4	2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success

Total Payload Mass

Total Payload Mass Carried by NASA was found to be 45596

We got this result by using the following code

```
dis_3 = '''
        SELECT SUM("PayloadMassKG") AS Total_PayloadMass
        FROM SPACEXTABLE
        WHERE Customer LIKE 'NASA (CRS)'
        '''
df=pd.read_sql(dis_3,con)
```

Average Payload Mass by F9 v1.1

The average payload mass was found ton be 2928.4 and we found it using the following code

```
dis_4 = '''
        SELECT AVG(PayloadMassKG) AS Avg_PayloadMass
        FROM SPACEXTABLE
        WHERE BoosterVersion = 'F9 v1.1'
        '''
df=pd.read_sqldf(dis_4,con)
```

First Successful Ground Landing Date

The first successful landing date was 2015-12-22 and we found the results by using the following code

```
dis_5 = '''
        SELECT MIN(Date) AS FirstSuccessfull_landing_date
        FROM SPACEXTABLE
        WHERE LandingOutcome LIKE 'Success (ground pad)'
        '''
df=pd.read_sqldf(dis_4,con)
```

Successful Drone Ship Landing with Payload between 4000 and 6000

This is the list the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

0	F9 FT B1022
1	F9 FT B1026
2	F9 FT B1021.2
3	F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

The total number of successful outcomes were 100

The total number of failures was only 1

```
task_7a = '''
        SELECT COUNT(MissionOutcome) AS SuccessOutcome
        FROM SPACEXTABLE
        WHERE MissionOutcome LIKE 'Success%'
        '''

task_7b = '''
        SELECT COUNT(MissionOutcome) AS FailureOutcome
        FROM SPACEXTABLE
        WHERE MissionOutcome LIKE 'Failure%'
        '''
print('The total number of successful mission outcome is:')
df=pd.read_sqldf(task_7a, database=conn))
print()
print('The total number of failed mission outcome is:')
df=pd.read_sqldf(task_7b, database=conn)
```

Boosters Carried Maximum Payload

Here's a list the names of the booster which have carried the maximum payload mass

boosterversion	payloadmasskg	
0	F9 B5 B1048.4	15600
1	F9 B5 B1048.5	15600
2	F9 B5 B1049.4	15600
3	F9 B5 B1049.5	15600
4	F9 B5 B1049.7	15600
5	F9 B5 B1051.3	15600
6	F9 B5 B1051.4	15600
7	F9 B5 B1051.6	15600
8	F9 B5 B1056.4	15600
9	F9 B5 B1058.3	15600
10	F9 B5 B1060.2	15600
11	F9 B5 B1060.3	15600

2015 Launch Records

- The list of failed landing outcomes in drone ship, their booster versions, and launch site names for in year 2015

boosterversion	launchsite	landingoutcome	
0	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
1	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

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

- Ranking 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 we get

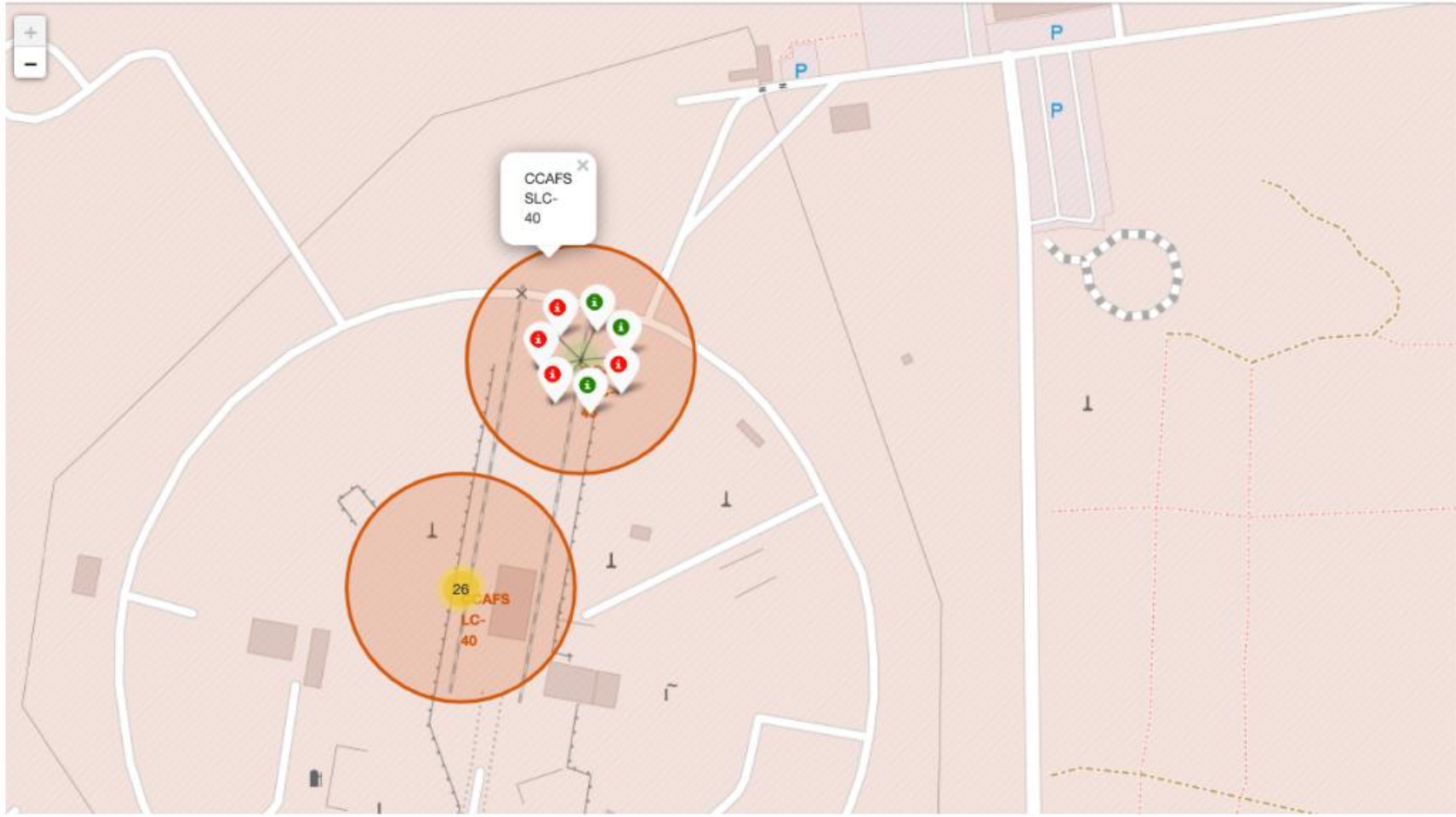
landingoutcome	count	
0	No attempt	10
1	Success (drone ship)	6
2	Failure (drone ship)	5
3	Success (ground pad)	5
4	Controlled (ocean)	3
5	Uncontrolled (ocean)	2
6	Precluded (drone ship)	1
7	Failure (parachute)	1

Section 3

Launch Sites Proximities Analysis

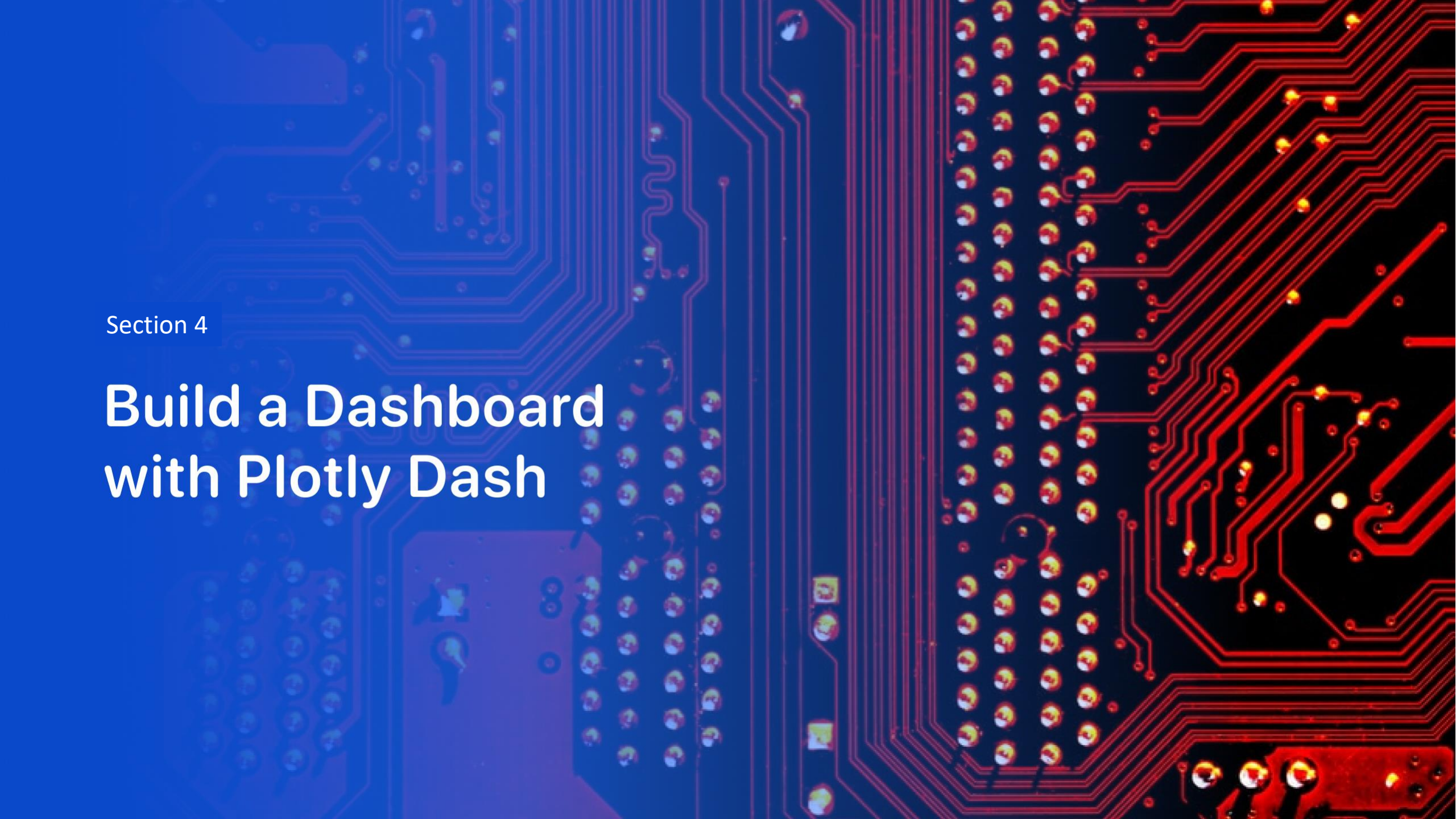
Global map showing all the launch sites

Here's a more detailed view

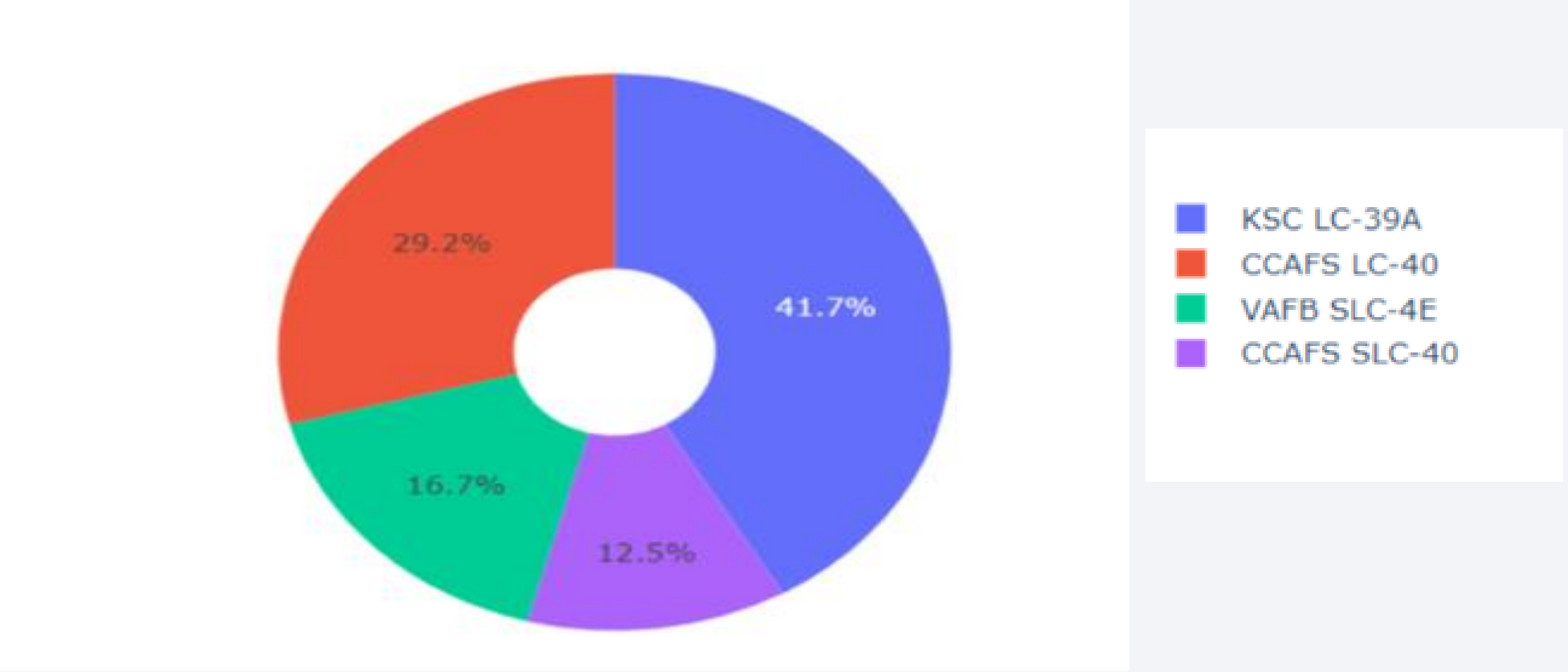


From the color-labeled markers in marker clusters, you should be able to easily identify which launch sites have relatively high success rates.

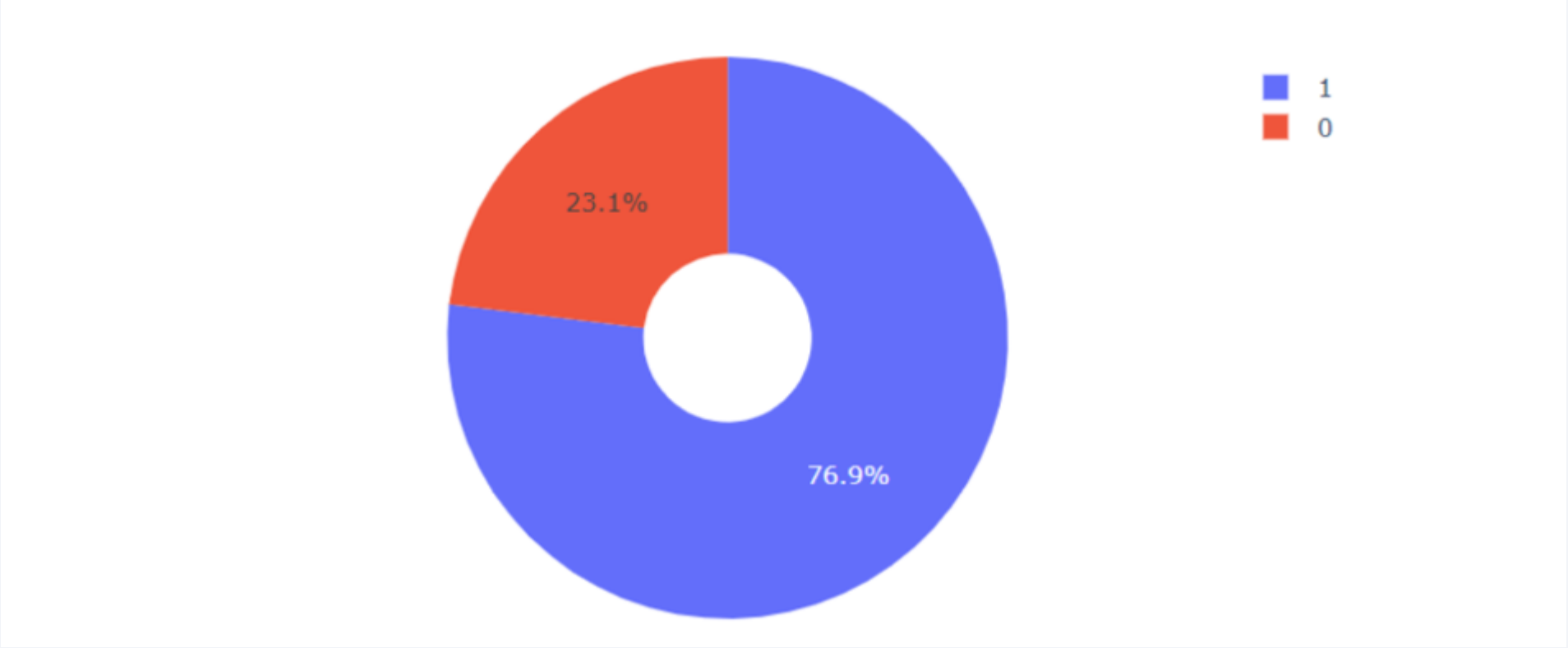
Map showing proximity to railway, highways, etc


Section 4
Build a Dashboard with Plotly Dash

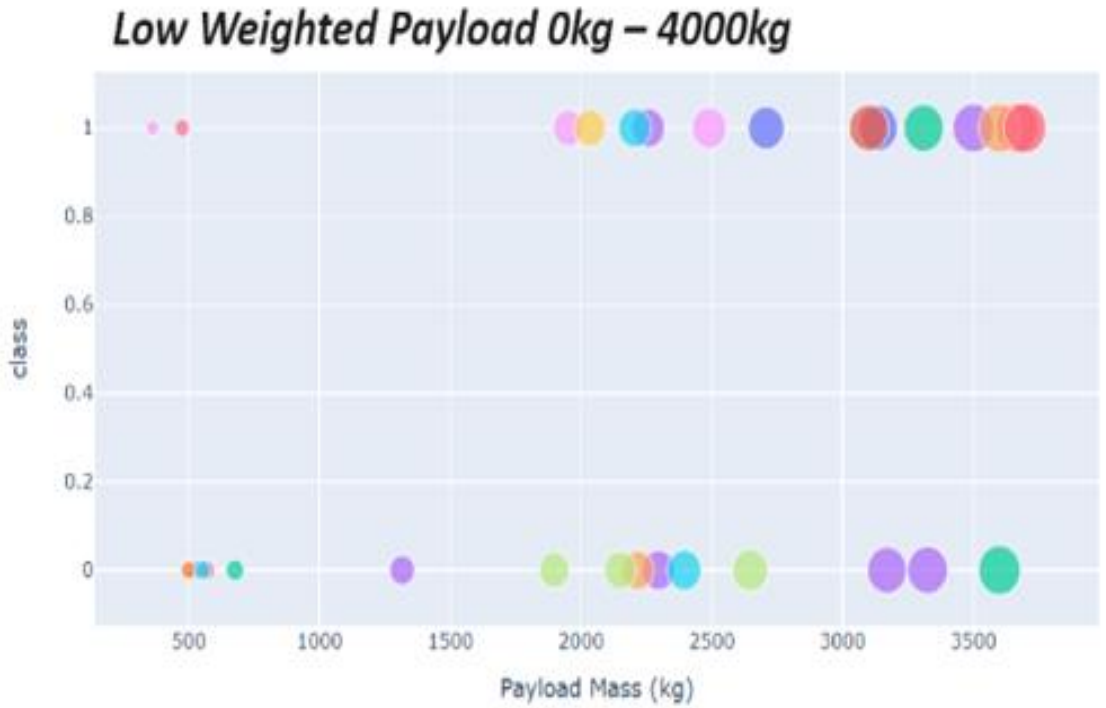
Percentage of Success achieved by the LaunchSites

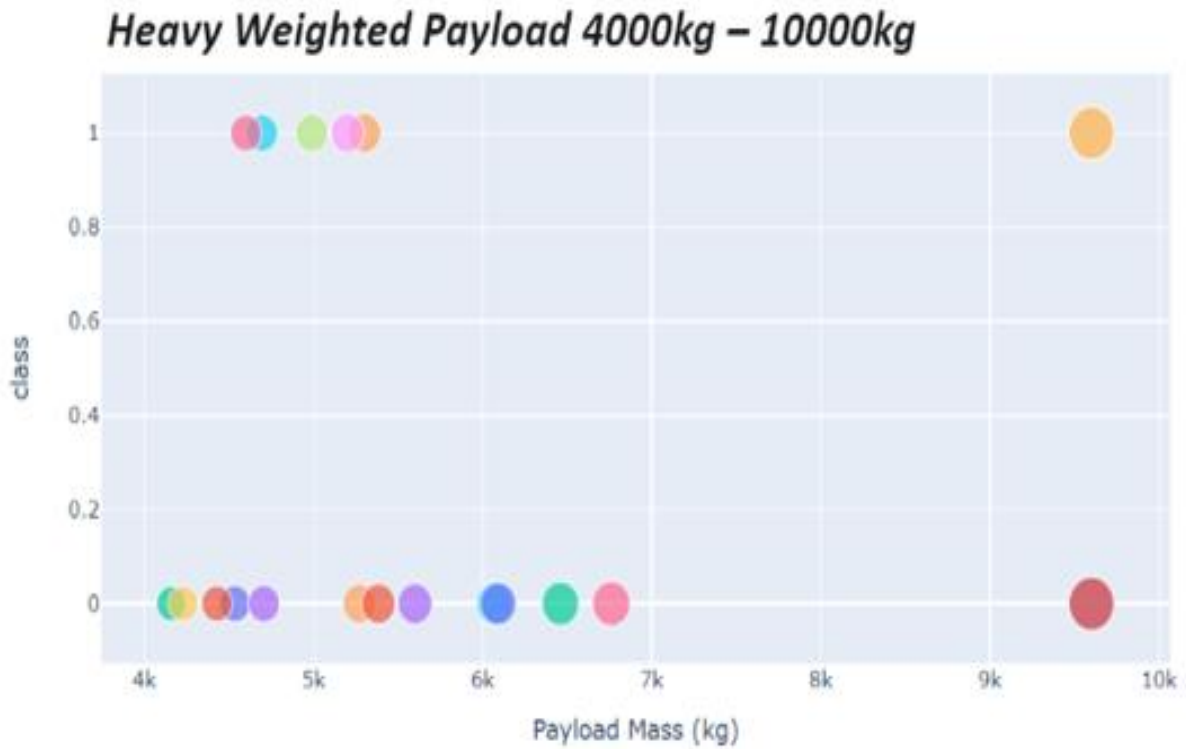


Launch Site with Highest Success Ratio: KSC LC-39A



Payload vs Launch Outcome For All the Sites



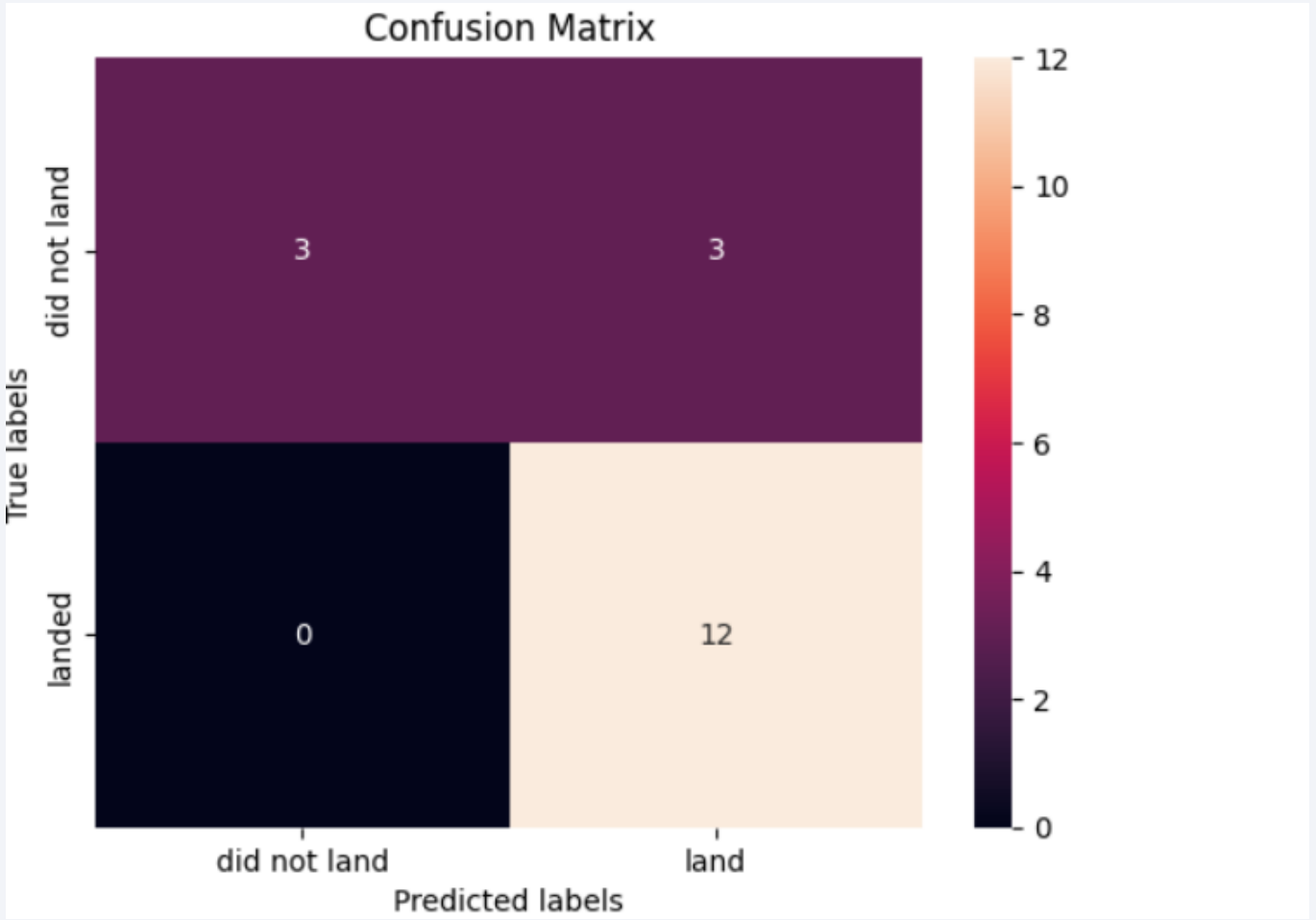


Section 5

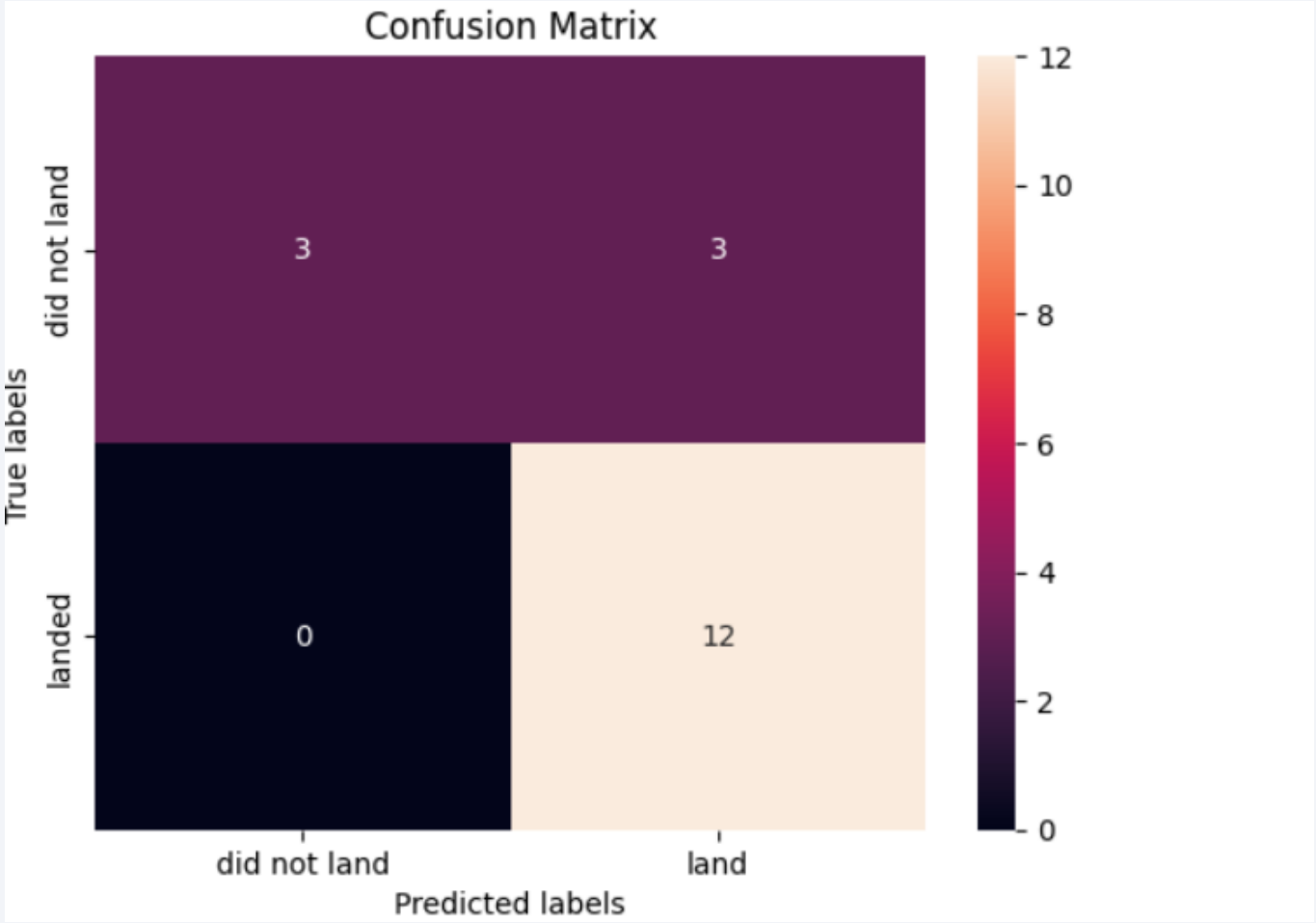
Predictive Analysis (Classification)

Classification Accuracy

It was found that the model with highest accuracy was DECISION TREE



Confusion Matrix of best performing model



Conclusions

With the completion of this project, we conclude that:

- Though this data, KSC LC-39A site seems to have most successful launches out of all the listed launch sites
- Orbit GEO,HEO,SSO as well as ES have the best success rates compared to others
- It was found that the launches where the payloads were low weighted, performed better than the ones with heavier weighted payloads
- For prediction and analysis purposes, machine learning models such as Decision tree and KNN are best suited

Appendix

- Refer to the github url provided in this project for more details

Thank you!