IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Maw Koon Khant
10-8-2025

Outline

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

Executive Summary

- Summary of methodologies
  * Data Collection (API & 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 Demo 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

* What factors determine if the rocket will land successfully?

* The interaction amongst various features that determine the success rate of a successful landing.

* What operating conditions needs to be in place to ensure a successful landing program.

Section 1

Methodology

Methodology

Data collection methodology:

* Data was collected using SpaceX API and Webscraping from Wikipedia.

Perform data wrangling

* One-hot encoding was applied to 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

* How to build, tune, evaluate classification models

Data Collection

The data was collected using various methods:

API Method

* Data collection was done using get request to the SpaceX API.

* Using .json() function call and turn it into a pandas dataframe and .json_normalize().

* After cleaning the data, checked for missing values and fill in missing values where necessary.

Web Scraping Method

* Using web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.

* Extract the launch records from HTML table, parse the table and convert it to a pandas dataframe for future analysis.

Data Collection – SpaceX API

Using the get request to the SpaceX API to collect data, clean the requested data and did some basic data wrangling and formatting.

The link to the notebook is https://github.com/mawkoon3/spacexcapstone/blob/main/Complete%20the%20Data%20Collection%20API%20Lab.ipynb

1 .Getting Response from API

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

2. Converting Response to a .json file

```
response = requests.get(static_json_url).json()
data = pd.json_normalize(response)
```

3. Apply custom functions to clean data

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

```
getBoosterVersion(data)
```

4. Assign list to dictionary then dataframe

```
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}
```

```
df = pd.DataFrame.from_dict(launch_dict)
```

5. Filter dataframe and export to flat file (.csv)

```
data_falcon9 = df.loc[df['BoosterVersion']!="Falcon 1"]
```

```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Data Collection - Scraping

Using web scraping to webscrap Falcon 9 launch records with BeautifulSoup

Parsing the table and converted it into a pandas dataframe.

The link to the notebook is https://github.com/mawkoon3/spacexcapstone/blob/main/2.Complete%20the%20Data%20Collection%20with%20Web%20Scraping%20lab.ipynb

1. Apply HTTP Get method to request the Falcon 9 rocket launch page

In [4]:
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

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

Out[5]:
```
200
```

2. Create a BeautifulSoup object from the HTML response

In [6]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(html_data.text, 'html.parser')
```

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

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

Out[7]:
```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

3. Extract all column names from the HTML table header

In [10]:
```
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

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
```

4. Create a dataframe by parsing the launch HTML tables
5. Export data to csv

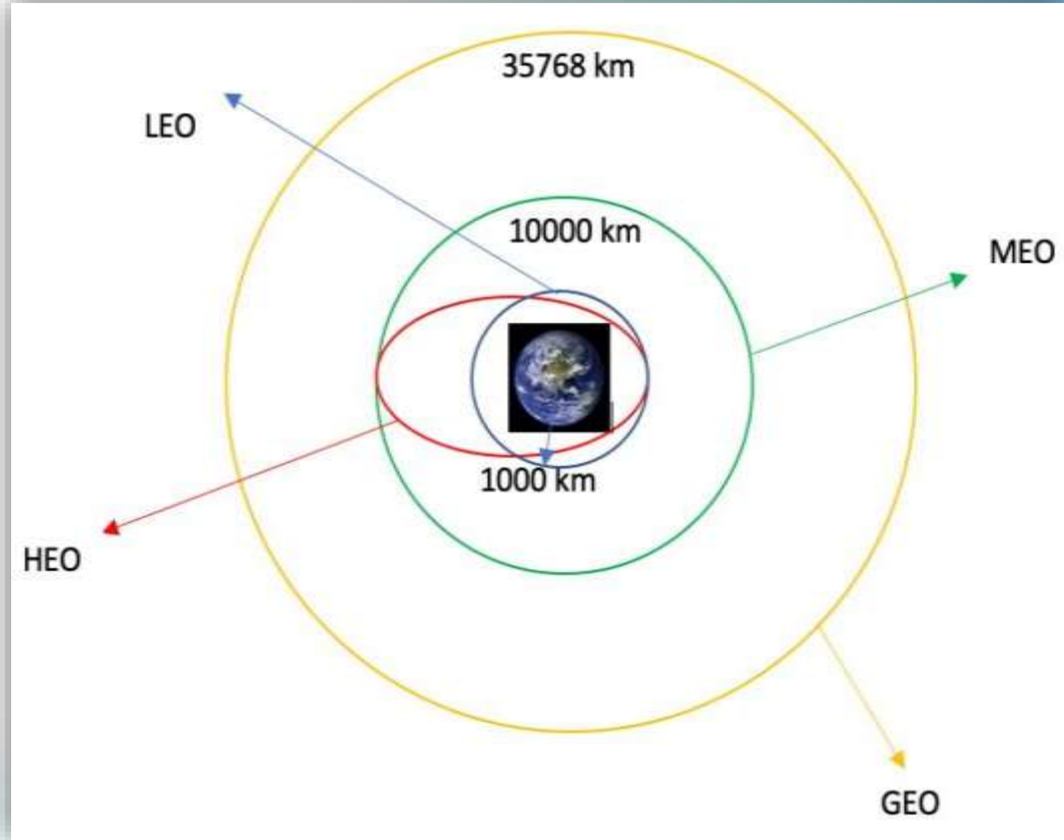
Data Wrangling

Performed EDA and determined the training labels.

Calculated the number of launches at each site, and the number and occurrence of each orbits

Created landing outcome label from outcome column and exported the results to csv.

The link to the notebook is https://github.com/mawkoon3/spacexcapstone/blob/main/3.Data%20Wrangling.ipynb



EDA with SQL

Loaded the SpaceX dataset into a DB2 database without leaving the notebook.

Applied EDA with SQL to get insight from the data. Execute queries to find out for instance:

* The names of unique launch sites in the space mission.

* The total payload mass carried by boosters launched by NASA (CRS)

* The average payload mass carried by booster version F9 v1.1

* The total number of successful and failure mission outcomes

* The failed landing outcomes in drone ship, their booster version and launch site names.

https://github.com/mawkoon3/spacexcapstone/blob/main/4.%20the%20EDA%20with%20SQL.ipynb

```
In [8]: %sql select * from SPACEXTBL where LAUNCH_SITE like 'CCA%' limit 5
```

* ibm_db_sa://[illegible]/bludb
Done.

Out[8]:

DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt

```
In [16]: %sql select * from SPACEXTBL where Landing__Outcome like 'Success%' and (DATE between '2010-06-04' and '2017-03-20') order by da
te desc
```

* ibm_db_sa://yhf02717:***@815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30367/bludb
Done.

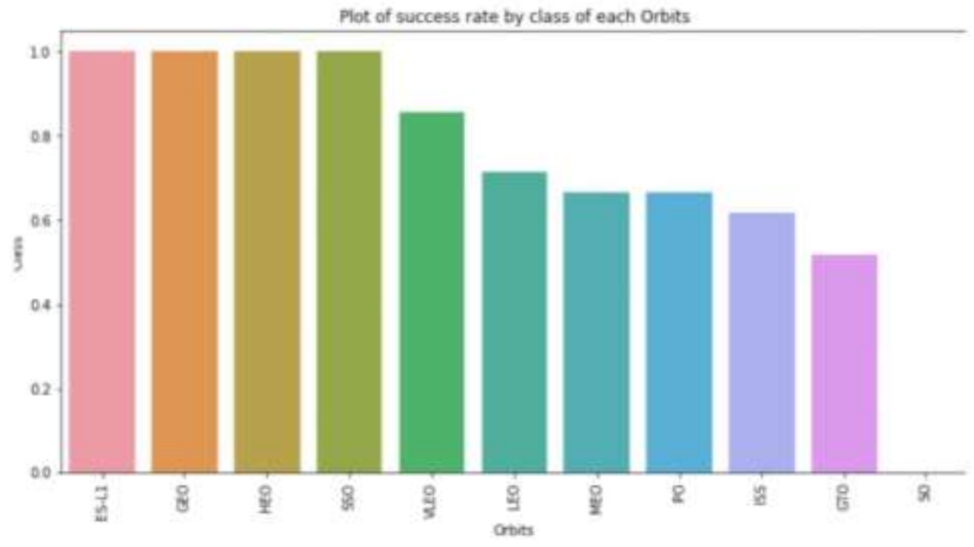
Out[16]:

DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2017-02-19	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
2017-01-14	17:54:00	F9 FT B1029.1	VAFB SLC-4E	Iridium NEXT 1	9600	Polar LEO	Iridium Communications	Success	Success (drone ship)
2016-08-14	05:26:00	F9 FT B1026	CCAFS LC-40	JCSAT-16	4600	GTO	SKY Perfect JSAT Group	Success	Success (drone ship)
2016-07-18	04:45:00	F9 FT B1025.1	CCAFS LC-40	SpaceX CRS-9	2257	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
2016-05-27	21:39:00	F9 FT B1023.1	CCAFS LC-40	Thaicom 8	3100	GTO	Thaicom	Success	Success (drone ship)
2016-05-06	05:21:00	F9 FT B1022	CCAFS LC-40	JCSAT-14	4696	GTO	SKY Perfect JSAT Group	Success	Success (drone ship)
2016-04-08	20:43:00	F9 FT B1021.1	CCAFS LC-40	SpaceX CRS-8	3136	LEO (ISS)	NASA (CRS)	Success	Success (drone ship)
2015-12-22	01:29:00	F9 FT B1019	CCAFS LC-40	OG2 Mission 2 11 Orbcomm-OG2 satellites	2034	LEO	Orbcomm	Success	Success (ground pad)

EDA with Data Visualization

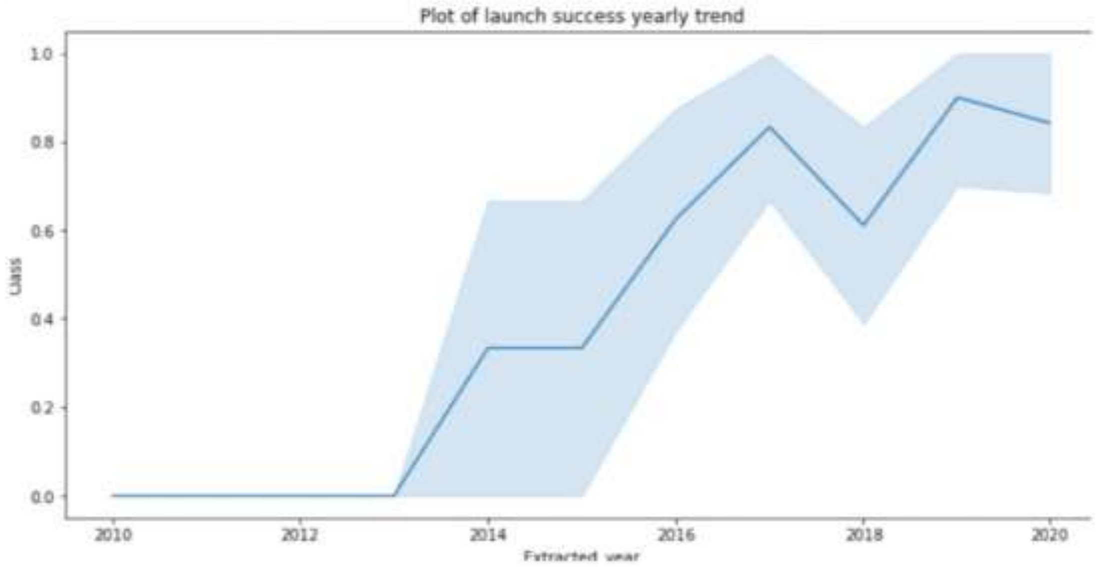
Explored the data by visualizing the relationship between flight number and launch Site, payload and launch site, success rate of each orbit type, flight number and orbit type, the launch success yearly trend.

```
# Use groupby method on Orbit column and get the mean of Class column
grouped_orbits = df.groupby(by=['Orbit'])['Class'].mean().sort_values(ascending=Fals
fig, ax=plt.subplots(figsize=(12,6))
ax = sns.barplot(x = 'Orbit', y = 'Class', data=grouped_orbits)
ax.set_title('Plot of success rate by class of each Orbits', fontdict={'size':12})
ax.set_ylabel('Class', fontsize = 10)
ax.set_xlabel('Orbits', fontsize = 10)
ax.set_xticklabels(ax.get_xticklabels(), fontsize = 10, rotation=90);
```




```
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
df_copy = df.copy()
df_copy['Extracted_year'] = pd.DatetimeIndex(df['Date']).year

# plot line chart
fig, ax=plt.subplots(figsize=(12,6))
sns.lineplot(data=df_copy, x='Extracted_year', y='Class')
plt.title('Plot of launch success yearly trend');
plt.show()
```



https://github.com/mawkoon3/spacexcapstone/blob/main/5.EDA%20with%20Visualization%20Lab.ipynb

Build an Interactive Map with Folium

Marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.

Assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure in RED, and 1 for success in GREEN.

Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate.

Florida Launch Sites

Green Marker shows successful Launches and Red Marker shows Failures

California Launch Site

We can see that the SpaceX launch sites are in the United States of America coasts. Florida and California

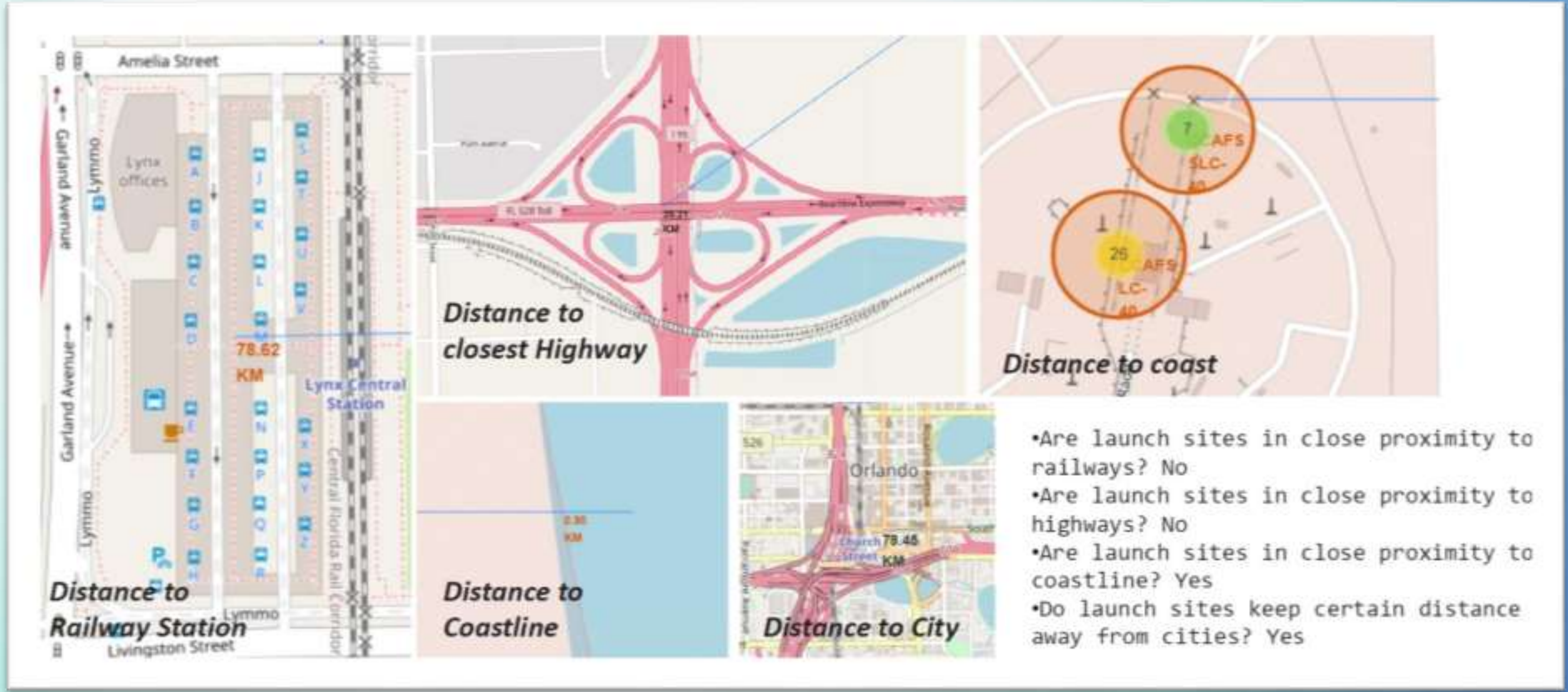
Build an Interactive Map with Folium

Calculated the distances between a launch site to its proximities. And find:-

* Are launch sites near railways, highways and coastlines.

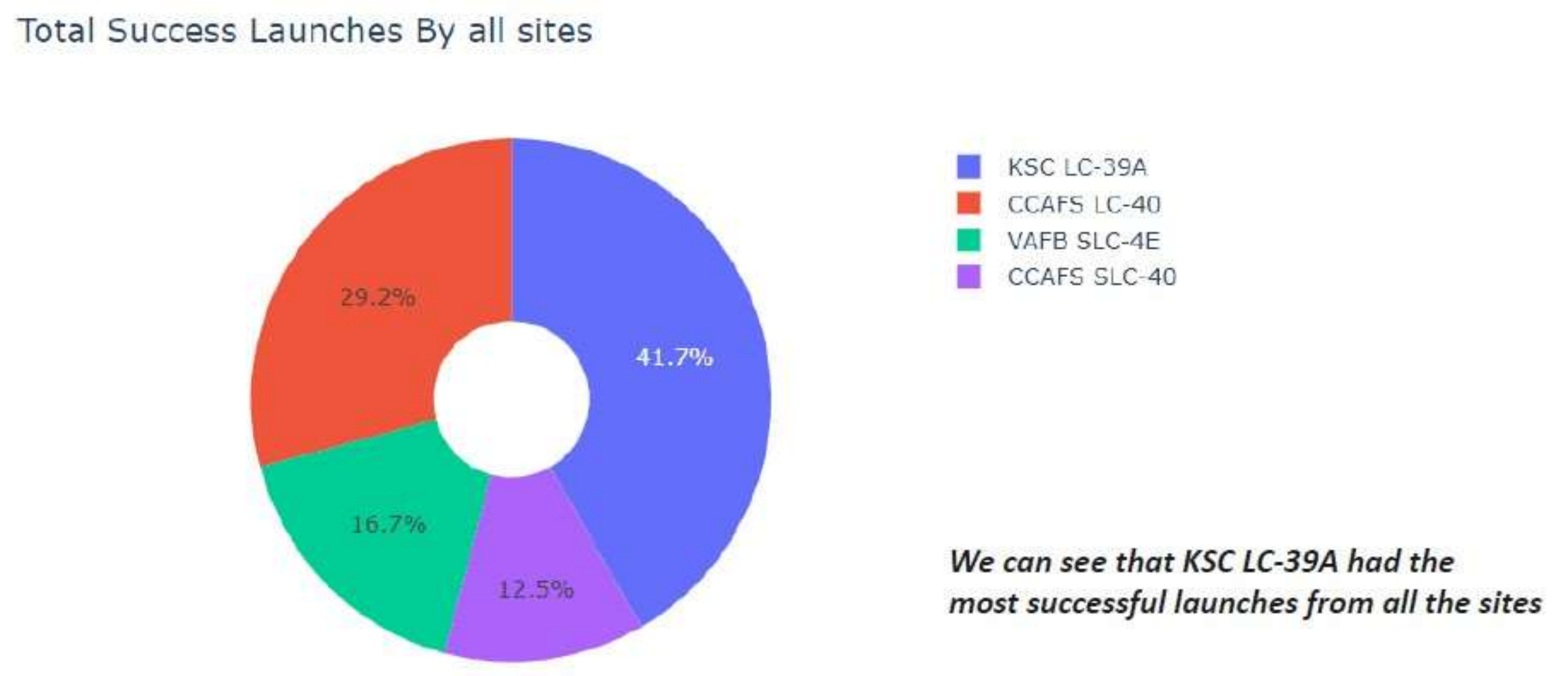
* Do launch sites keep certain distance away from cities.

The link to the notebook is https://github.com/mawkoon3/spacexcapstone/blob/main/6.Interactive%20Visual%20Analytics%20with%20Folium%20lab.ipynb



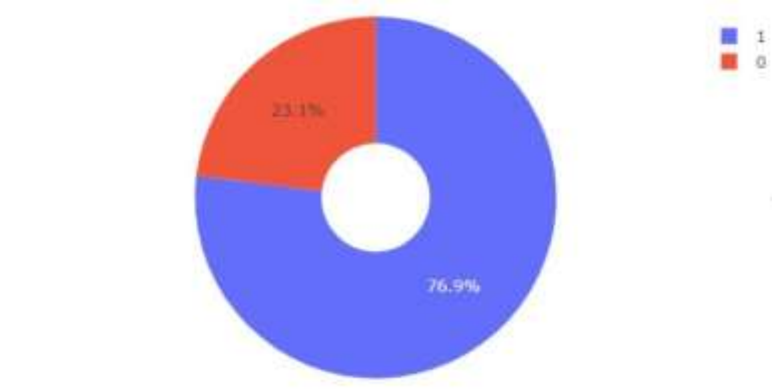
Build a Dashboard with Plotly Dash

Built an interactive dashboard with Plotly dash. The link to the notebook is https://github.com/mawkoon3/spacexcapstone/blob/main/7.Build%20an%20Interactive%20Dashboard%20with%20Plotly%20Dash.py Plotted pie charts showing the total launches by a certain sites



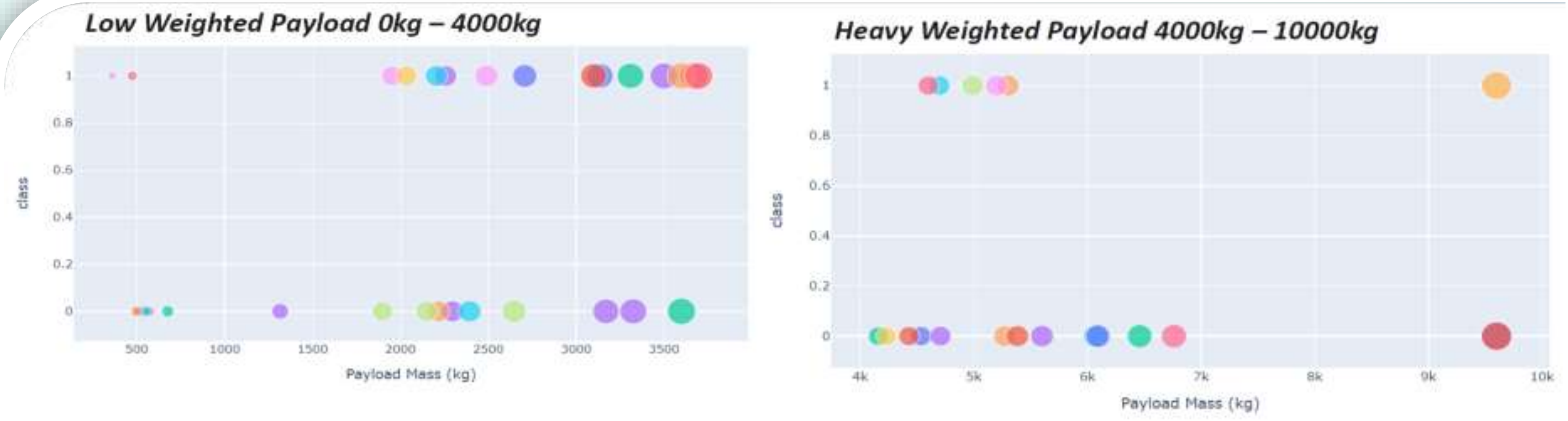
Build a Dashboard with Plotly Dash

Plotted pie charts showing the site of KSC LC-39A achieved most success rate



KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

Plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.



We can see the success rates for low weighted payloads is higher than the heavy weighted payloads

Predictive Analysis (Classification)

Load the data using numpy and pandas, transformed the data, split our data into training and testing.

Built different machine learning models and tune different hyperparameters using GridSearchCV.

Use accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.

Found the best performing classification model.

The link to the notebook is https://github.com/mawkoon3/spacexcapstone/blob/main/8.Complete%20the%20Machine%20Learning%20Prediction%20lab.ipynb

Results of Predictive Analysis

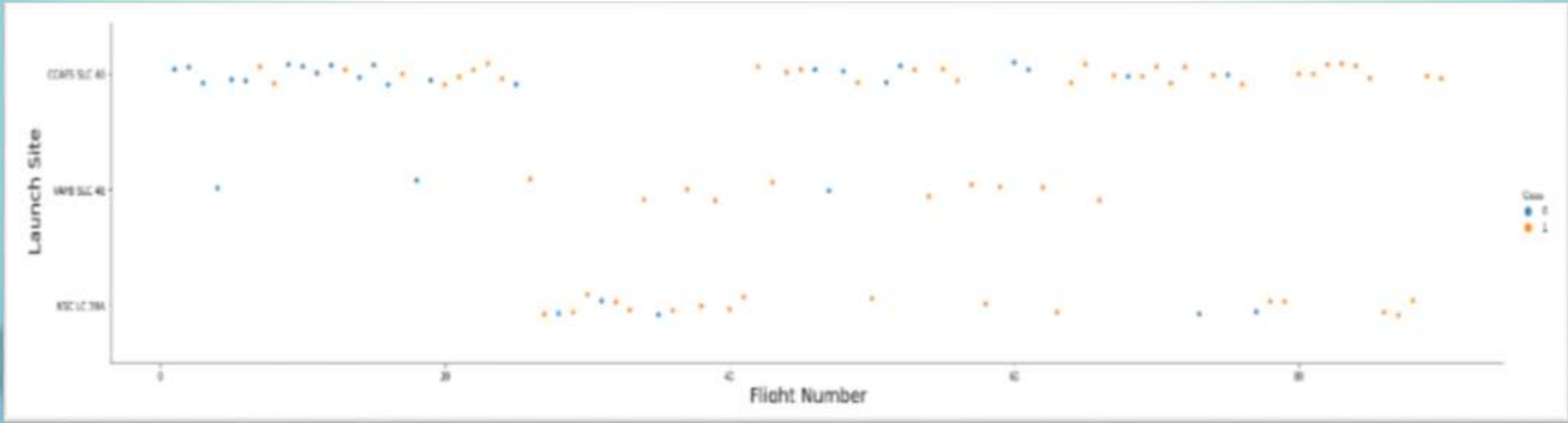
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2 (RESULT)

Insights drawn from EDA

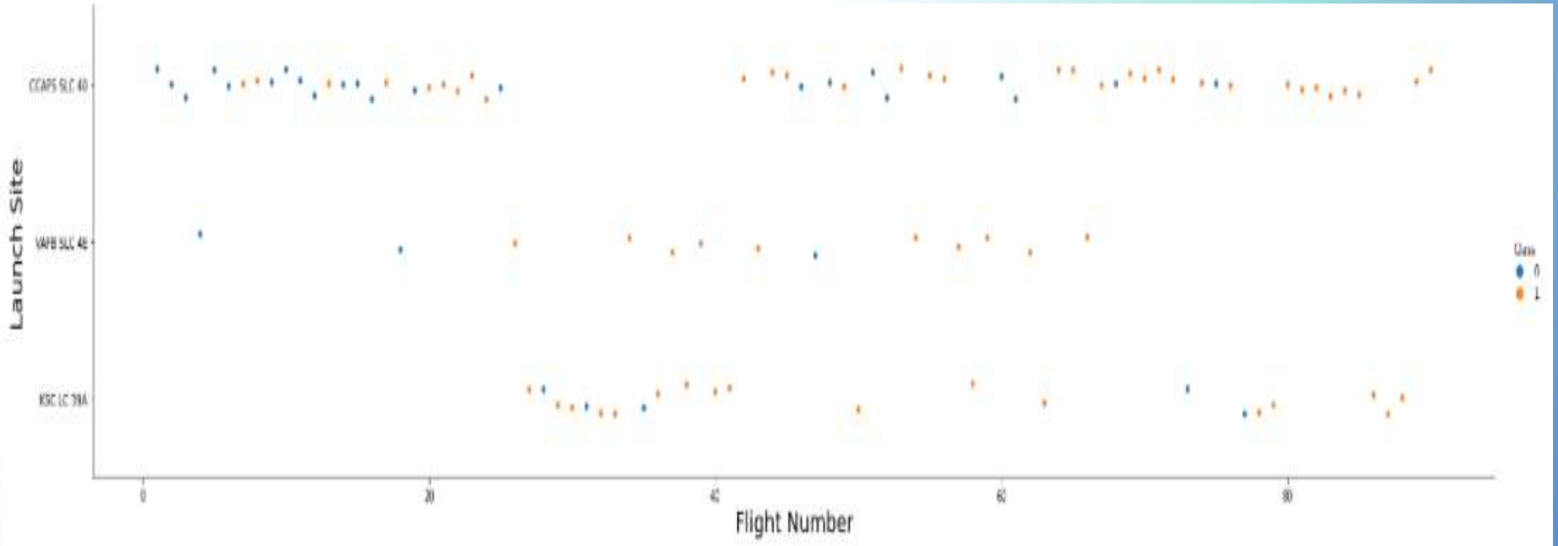
Flight Number vs. Launch Site

From the plot, we found that the larger the flight amount at a launch site, the greater the success rate at a launch site.



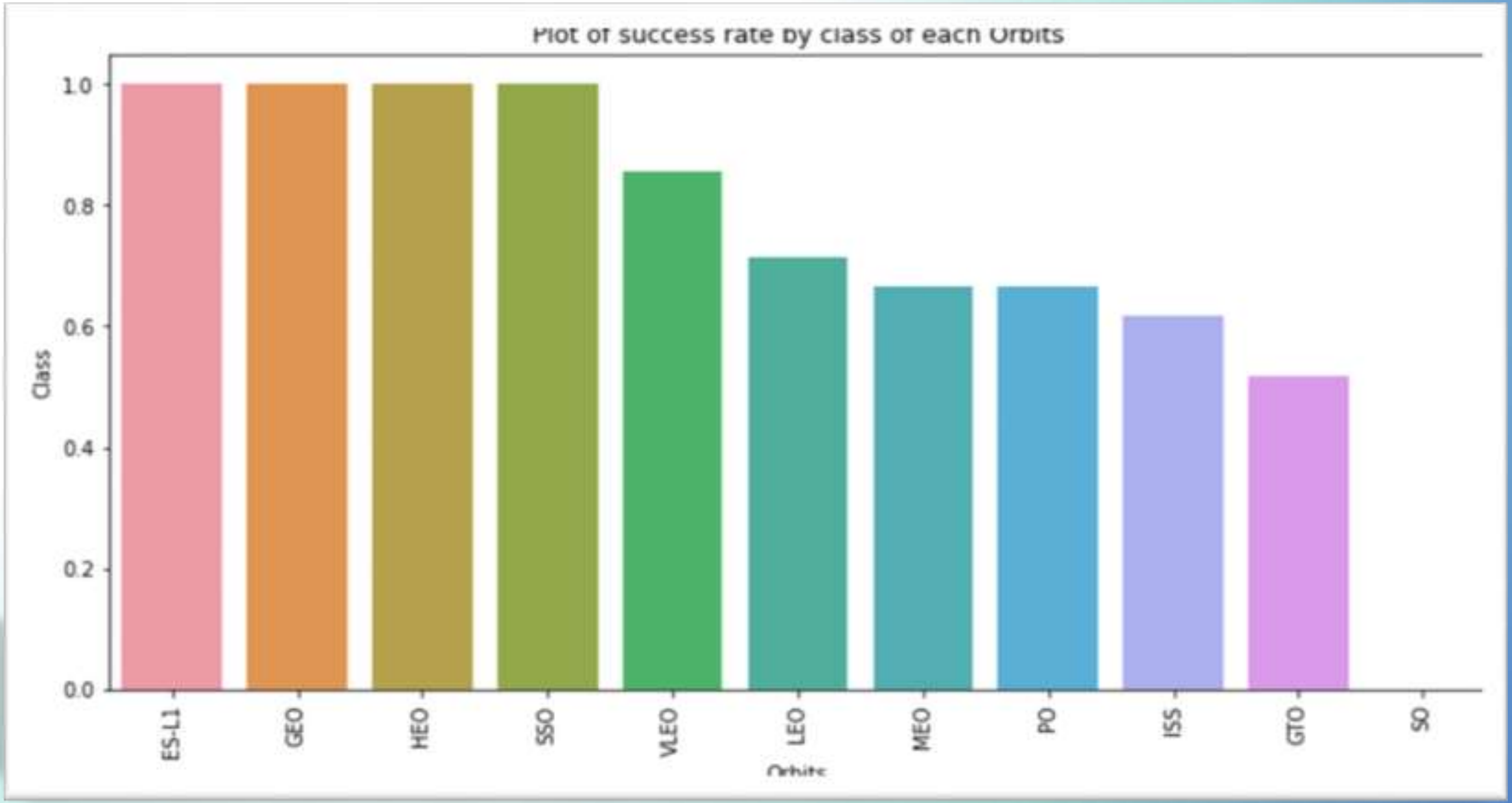
Payload vs. Launch Site

The greater the payload mass for launch site CCAFS SLC 40 the higher the success rate for the rocket.



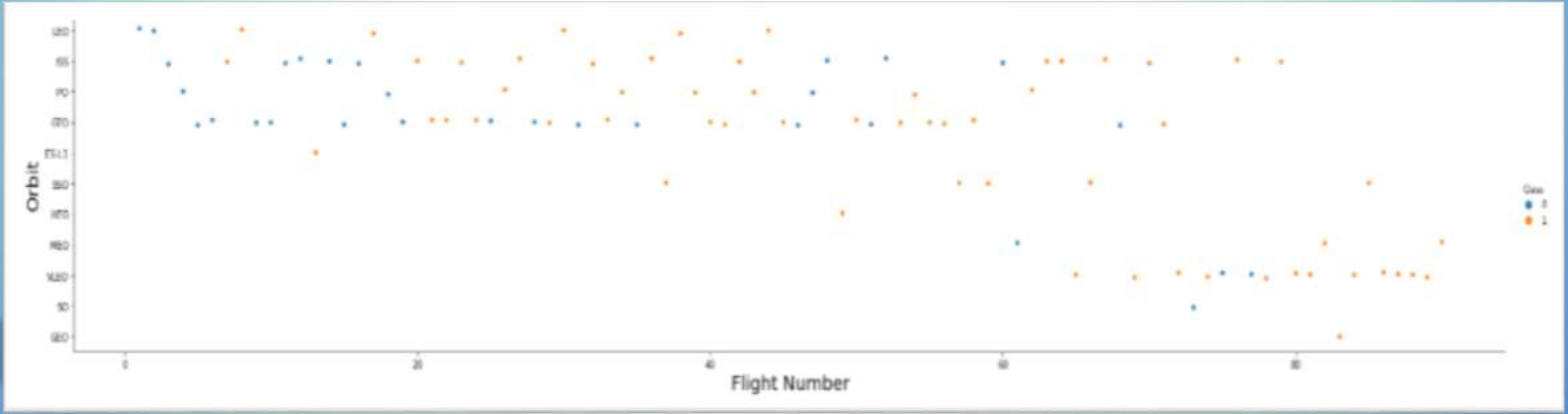
Success Rate vs. Orbit Type

From the plot, we can see that ES-L1, GEO, HEO, SSO, VLEO had the most success rate.



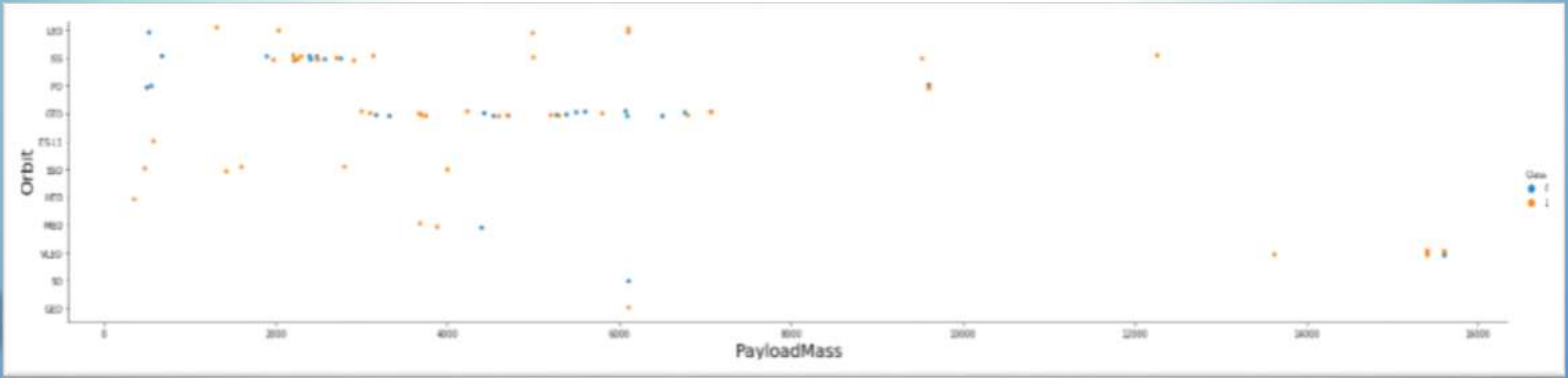
Flight Number vs. Orbit Type

The plot below shows the Flight Number vs. Orbit type. We observe that in the LEO orbit, success is related to the number of flights whereas in the GTO orbit, there is no relationship between flight number and the orbit.



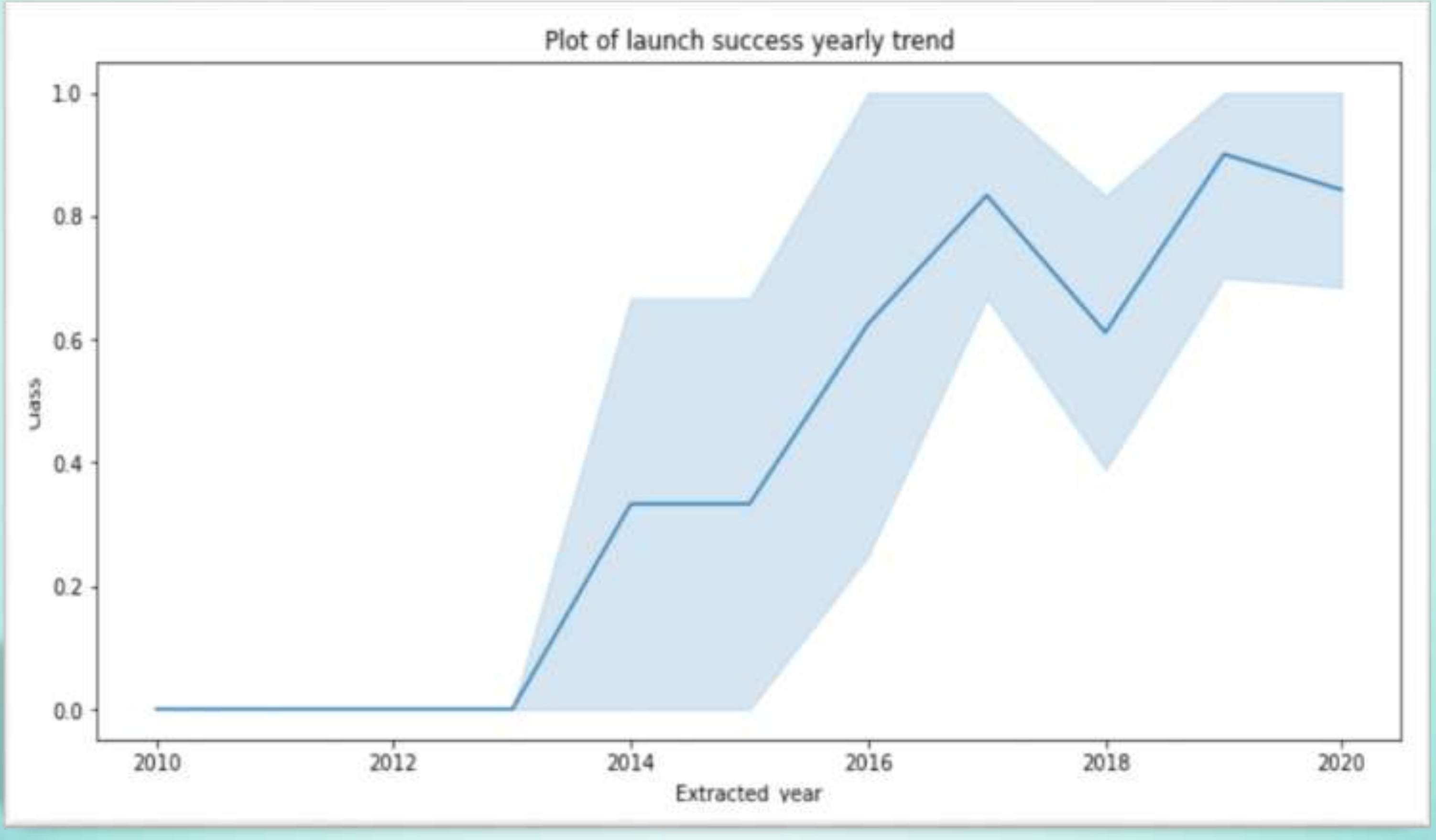
Payload vs. Orbit Type

We can observe that with heavy payloads, the successful landing are more for PO, LEO and ISS orbits.



Launch Success Yearly Trend

From the plot, we can observe that success rate since 2013 kept on increasing till 2020.



All Launch Site Names

We used the key word **DISTINCT** to show only unique launch sites from the SpaceX data.

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

In [10]:

```
task_1 = '''
        SELECT DISTINCT LaunchSite
        FROM SpaceX
'''
create_pandas_df(task_1, database=conn)
```

Out[10]:

	launchsite
0	KSC LC-39A
1	CCAFS LC-40
2	CCAFS SLC-40
3	VAFB SLC-4E

Launch Site Names Begin with 'CCA'

Display 5 records where launch sites begin with the string 'CCA'

In [11]:

```
task_2 = '''
        SELECT *
        FROM SpaceX
        WHERE LaunchSite LIKE 'CCA%'
        LIMIT 5
        '''
create_pandas_df(task_2, database=conn)
```

Out[11]:

	date	time	boosterversion	launchsite	payload	payloadmasskg	orbit	customer	missionoutcome	landingoutcome
0	2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
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	Failure (parachute)
2	2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
3	2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

We used the query above to display 5 records where launch sites begin with \`CCA\`

Total Payload Mass

We calculated the total payload carried by boosters from NASA as 45596 using the query below

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

In [12]:
```
task_3 = '''
        SELECT SUM(PayloadMassKG) AS Total_PayloadMass
        FROM SpaceX
        WHERE Customer LIKE 'NASA (CRS)'
        '''
create_pandas_df(task_3, database=conn)
```

Out[12]:

	total_payloadmass
0	45596

Average Payload Mass by F9 v1.1

We calculated the average payload mass carried by booster version F9 v1.1 as 2928.4

Display average payload mass carried by booster version F9 v1.1

In [13]:
```
task_4 = '''
        SELECT AVG(PayloadMassKG) AS Avg_PayloadMass
        FROM SpaceX
        WHERE BoosterVersion = 'F9 v1.1'
        '''
create_pandas_df(task_4, database=conn)
```

Out[13]:

	avg_payloadmass
0	2928.4

First Successful Ground Landing Date

We observed that the dates of the first successful landing outcome on ground pad was 22nd December 2015

In [14]:
```
task_5 = '''
        SELECT MIN(Date) AS FirstSuccessfull_landing_date
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Success (ground pad)'
        '''
create_pandas_df(task_5, database=conn)
```

Out[14]:

	firstsuccessfull_landing_date
0	2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

We used the **WHERE** clause to filter for boosters which have successfully landed on drone ship and applied the **AND** condition to determine successful landing with payload mass greater than 4000 but less than 6000

In [15]:

```
task_6 = '''
        SELECT BoosterVersion
        FROM SpaceX
        WHERE LandingOutcome = 'Success (drone ship)'
            AND PayloadMassKG > 4000
            AND PayloadMassKG < 6000
        '''
create_pandas_df(task_6, database=conn)
```

Out[15]:

	boosterversion
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

We used wildcard like ‘%’ to filter for **WHERE** MissionOutcome was a success or a failure.

List the total number of successful and failure mission outcomes

In [16]:
```
task_7a = '''
        SELECT COUNT(MissionOutcome) AS SuccessOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Success%'
        '''

task_7b = '''
        SELECT COUNT(MissionOutcome) AS FailureOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Failure%'
        '''
print('The total number of successful mission outcome is:')
display(create_pandas_df(task_7a, database=conn))
print()
print('The total number of failed mission outcome is:')
create_pandas_df(task_7b, database=conn)
```

The total number of successful mission outcome is:

	successoutcome
0	100

The total number of failed mission outcome is:

Out[16]:

	failureoutcome
0	1

Boosters Carried Maximum Payload

We determined the booster that have carried the maximum payload using a subquery in the **WHERE** clause and the **MAX()** function.

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

In [17]:

```
task_8 = '''
        SELECT BoosterVersion, PayloadMassKG
        FROM SpaceX
        WHERE PayloadMassKG = (
                                SELECT MAX(PayloadMassKG)
                                FROM SpaceX
                                )
        ORDER BY BoosterVersion
        '''
create_pandas_df(task_8, database=conn)
```

Out[17]:

	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

We used a combinations of the **WHERE** clause, **LIKE**, **AND**, and **BETWEEN** conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015

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

In [18]:

```
task_9 = '''
        SELECT BoosterVersion, LaunchSite, LandingOutcome
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Failure (drone ship)'
            AND Date BETWEEN '2015-01-01' AND '2015-12-31'
        '''
create_pandas_df(task_9, database=conn)
```

Out[18]:

	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

We selected Landing outcomes and the **COUNT** of landing outcomes from the data and used the **WHERE** clause to filter for landing outcomes **BETWEEN** 2010-06-04 to 2010-03-20.

We applied the **GROUP BY** clause to group the landing outcomes and the **ORDER BY** clause to order the grouped landing outcome in descending order.

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad))

In [19]:

```
task_10 = '''
        SELECT LandingOutcome, COUNT(LandingOutcome)
        FROM SpaceX
        WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
        GROUP BY LandingOutcome
        ORDER BY COUNT(LandingOutcome) DESC
        '''
create_pandas_df(task_10, database=conn)
```

Out[19]:

	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 (RESULT)

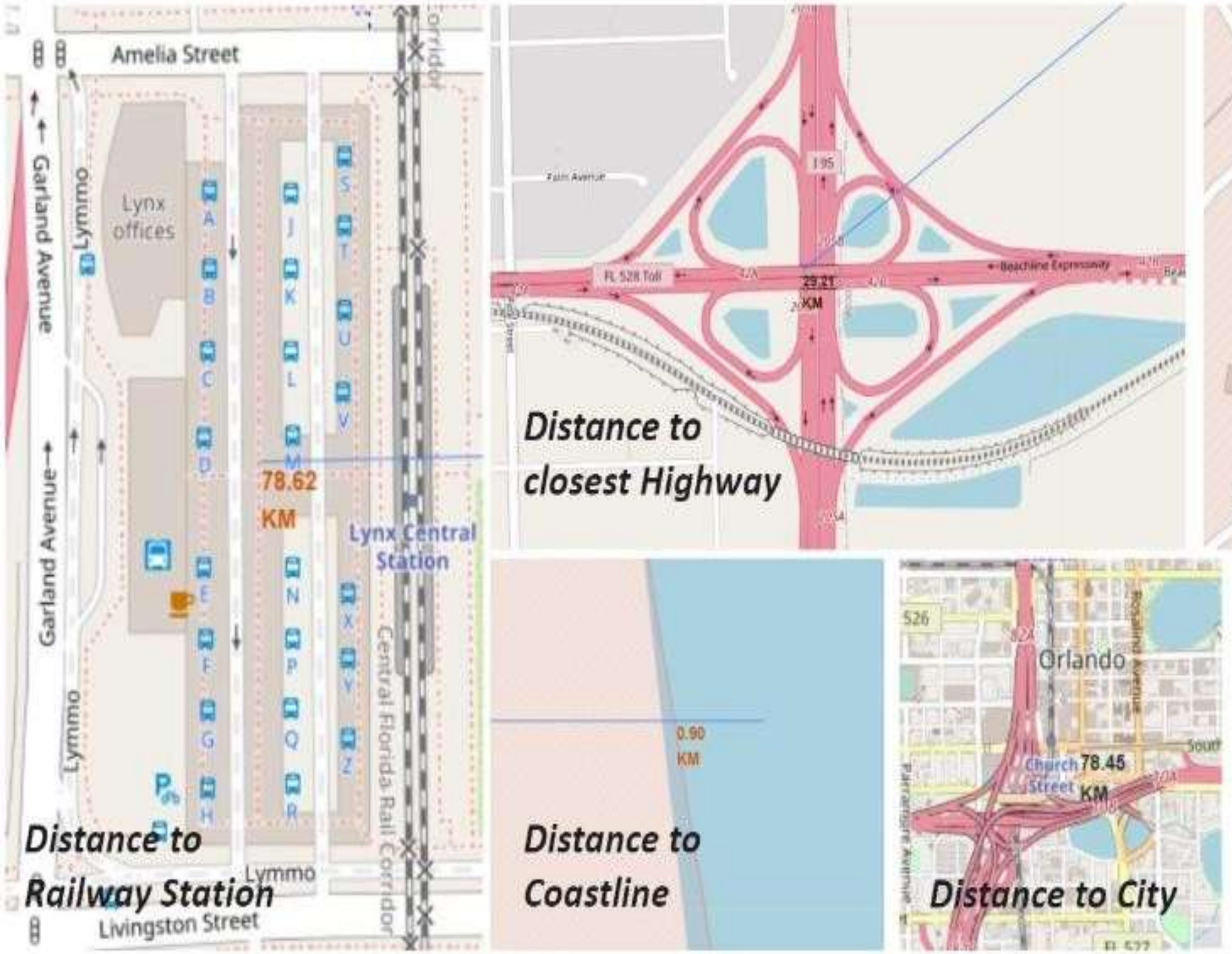
Launch Sites Proximities Analysis

All launch sites global map markers

Markers showing launch sites with color labels

Launch Outcomes for Two Sites

Launch Site distance to landmarks

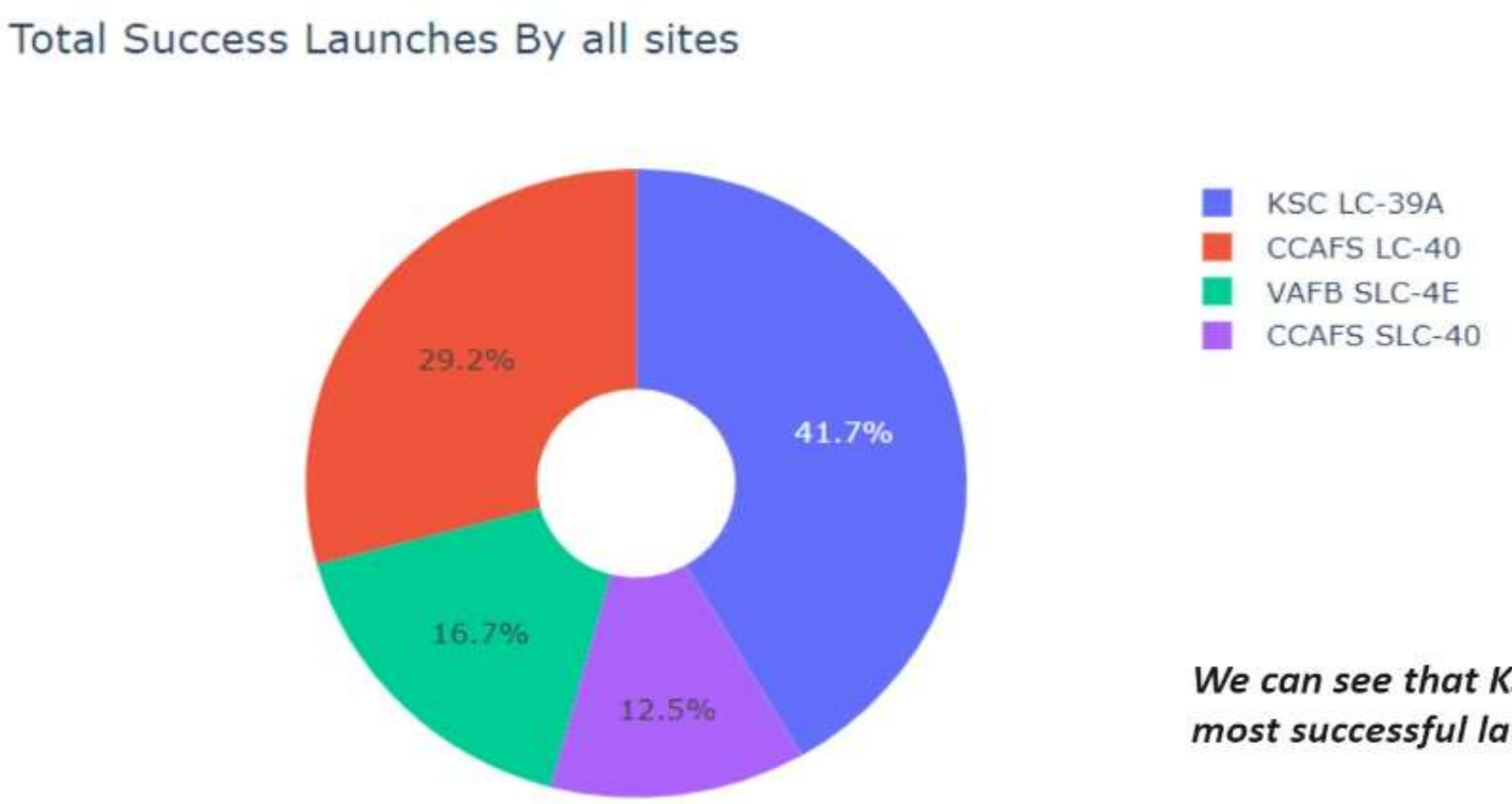


- •Are launch sites in close proximity to railways? No
- •Are launch sites in close proximity to highways? No
- •Are launch sites in close proximity to coastline? Yes
- •Do launch sites keep certain distance away from cities? Yes

Section 4 (RESULT)

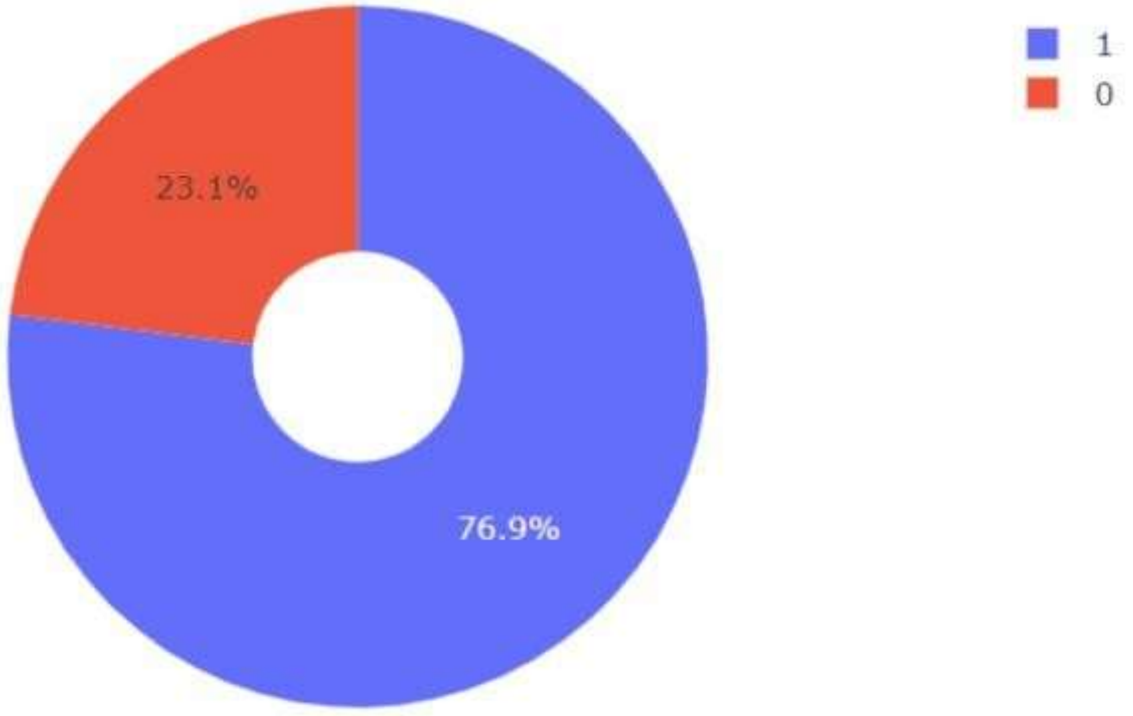
Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site



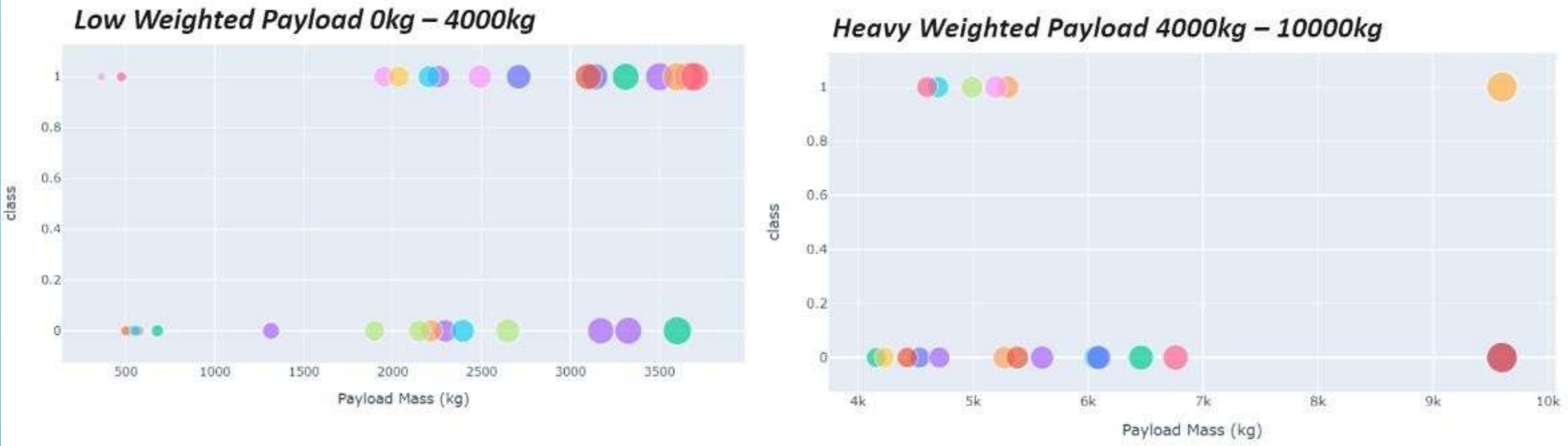
We can see that KSC LC-39A had the most successful launches from all the sites

Pie chart showing the Launch site with the highest launch success ratio



KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

Scatter plot of Payload vs Launch Outcome for all sites, with different payload selected in the range slider



We can see the success rates for low weighted payloads is higher than the heavy weighted payloads

Section 5 (RESULT)

Predictive Analysis (Classification)

Classification Accuracy

The decision tree classifier is the model with the highest classification accuracy

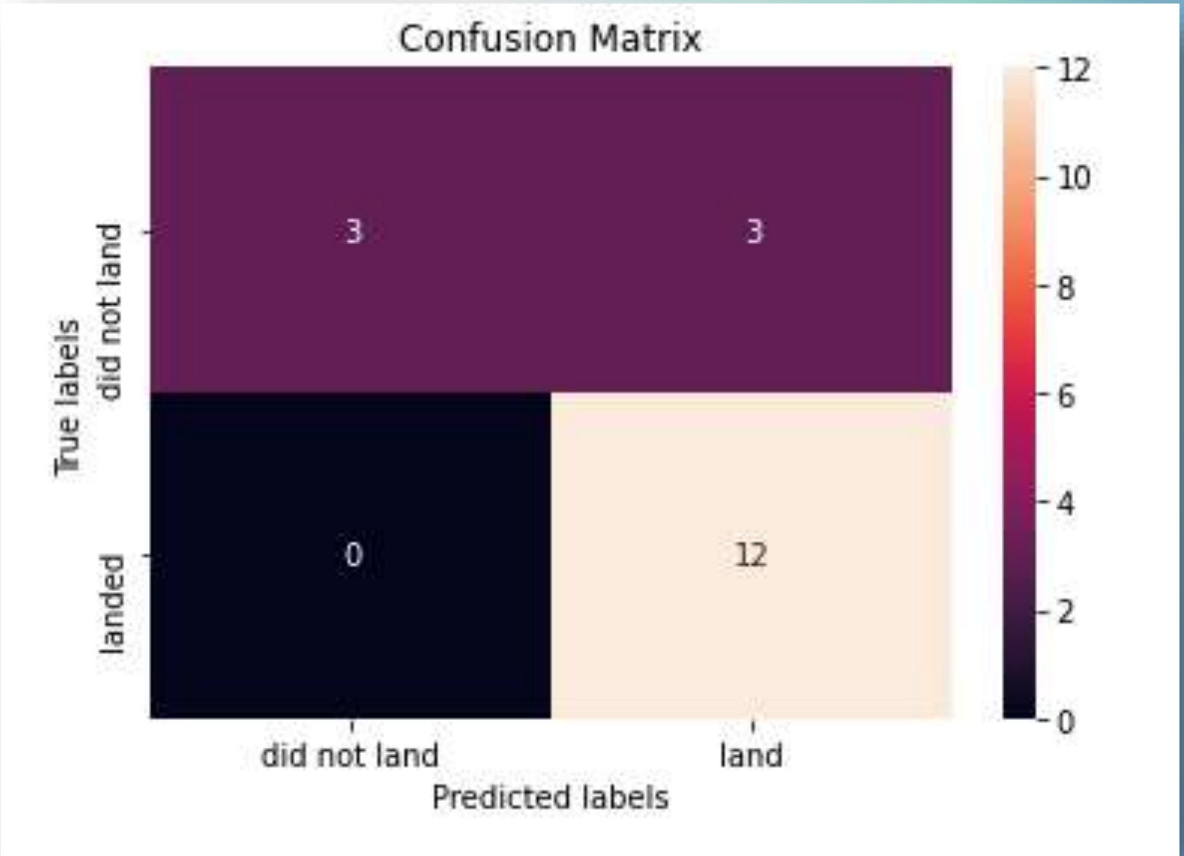
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)
```

```
Best model is DecisionTree with a score of 0.8732142857142856
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```

Confusion Matrix

The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes. The major problem is the false positives .i.e., unsuccessful landing marked as successful landing by the classifier.



Conclusions

It is concluded that:

The larger the flight amount at a launch site, the greater the success rate at a launch site.

Launch success rate started to increase in 2013 till 2020.

Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate.

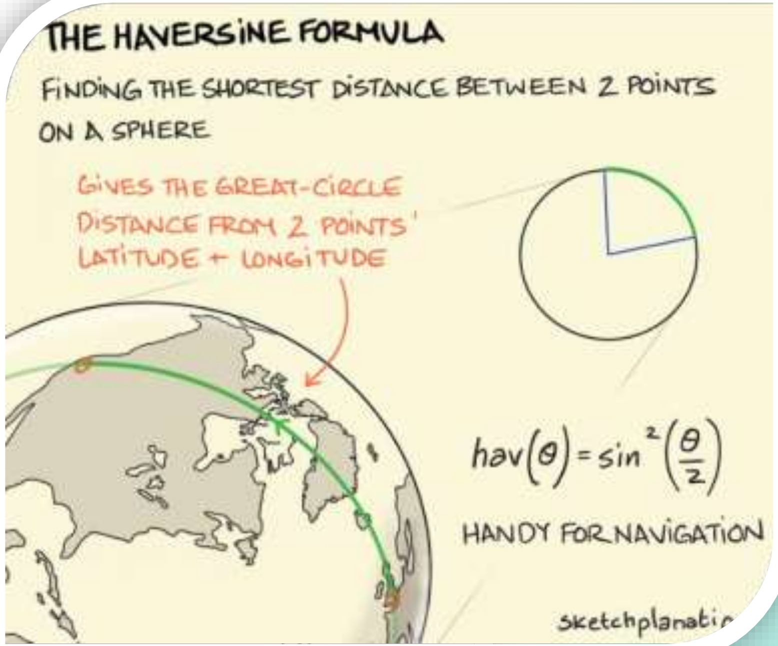
KSC LC-39A had the most successful launches of any sites.

The Decision tree classifier is the best machine learning algorithm for this task.

Appendix

(1) Haversine Formula

https://plus.maths.org/content/lost-lovely-haversine



$$\sin^2\left(\frac{d}{2R}\right) = \sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos\phi_1 \cos\phi_2 \sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right),$$

(where the angles are measured in radians).

Solving for d gives

$$d = 2R\sin^{-1}\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos\phi_1 \cos\phi_2 \sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right).$$

You'll admit that this isn't the simplest of formulae. If you were are a seafarer hundreds of years ago, armed only with sine and cosine tables to help you, working out the distance d would prove pretty cumbersome. There's a square root to take, as well as the inverse of the sine function .... argh!

Appendix

(2) ADGGoogleMaps Module

```
import gmaps
import gmaps.datasets

# Use google maps api
gmaps.configure(api_key=api_key) # Fill in with your API key

# Get the dataset
earthquake_df = gmaps.datasets.load_dataset_as_df('earthquakes')

#Get the locations from the data set
locations = earthquake_df[['latitude', 'longitude']]

#Get the magnitude from the data
weights = earthquake_df['magnitude']

#Set up your map
fig = gmaps.figure()
fig.add_layer(gmaps.heatmap_layer(locations, weights=weights))
fig
```

Thank you!