IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Robert Chase
October 7, 2023

OUTLINE

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

EXECUTIVE SUMMARY

- We've used a series of Data Science methodologies to analyze the data. A short list is as follows:
  - Data Collection, Sampling, and Wrangling
  - Webscraping and API
  - Beautiful Soup
  - Exploratory Analysis using SQL, Pandas, and Matplotlib
  - Visual Analytics and Dashboarding
  - Predictive Analysis
- Summary of all results
  - Exploratory Data Analysis results
  - Interactive analytics in screenshots
  - Predictive Analytics results

Background: In the race to space, companies are now aiming to make space travel more affordable and cost effective, ultimately successful. However, with only a few players in the industry able to produce rockets for passenger travel at scale, we need to find solutions to more successful and cost effective space travel. Enter Space Y.

- The Problem: We need to analyze the cost of each rocket launch and the success of the landing and predict the probability of reuse of the *first stage* of rocket propulsion.

Context: Space X has found a way to make space travel relatively cheap, however, at Space Y, we want to make it even cheaper and more effective.

INTRODUCTION

Section 1

Methodology

Executive Summary

Data collection methodology:

Data was collected using SpaceX API and web scraping 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

METHODOLOGY

DATA COLLECTION

- The data was collected using various methods
  - Data collection was done using get request to the SpaceX API.
  - Next, we decoded the response content as a Json using .json() function call in order to turn it into a pandas dataframe using the _normalize() function.
  - We then cleaned the data, checked for missing values and fill in missing values where necessary.
  - In addition, we performed web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.
  - The objective was to extract the launch records as HTML table, parse the table and convert it to a pandas dataframe for future analysis.

DATA COLLECTION – SPACEX API

1. request for rocket launch data using API

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

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

2. Use json_normalize method to convert json result to dataframe

```
[12]: # Use json_normalize method to convert the json result into a dataf

      # decode response content as json
      static_json_df = res.json()
```

```
[13]: # apply json_normalize
      data = pd.json_normalize(static_json_df)
```

3. We then performed data cleaning and filling in the missing values

```
[30]: rows = data_falcon9['PayloadMass'].values.tolist()[0]

      df_rows = pd.DataFrame(rows)
      df_rows = df_rows.replace(np.nan, PayloadMass)

      data_falcon9['PayloadMass'][0] = df_rows.values
      data_falcon9
```

- We used the requests.get() to collect the data from the SpaceX API, clean the requested data and did some basic data wrangling and formatting.
- The link to the notebook is https://github.com/robbthe/IBM_Data_Science_Capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

Data Collection - Scraping

We applied web scrapping to retrieve Falcon 9 launch records using the BeautifulSoup Method.

We parsed the table and converted it into a pandas dataframe.

The link to the notebook is https://github.com/robbthe/IBM_Data_Science_Capstone/blob/main/jupyter-labs-webscraping.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

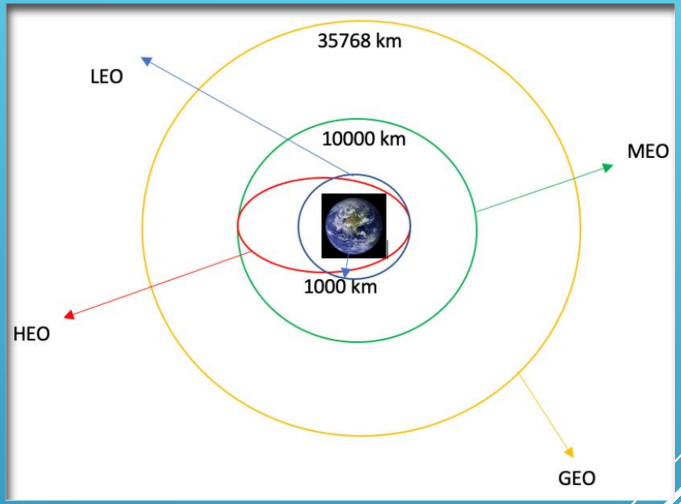
In [10]:
```
column_names = []

# 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

- We performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits
- We created landing outcome label from outcome column and exported the results to csv.
- The link to the notebook is https://github.com/robbthe/IBM_Data_Science_Capstone/blob/main/labs-jupyter-spacex-data_wrangling_jupyterlite.jupyterlite.ipynb

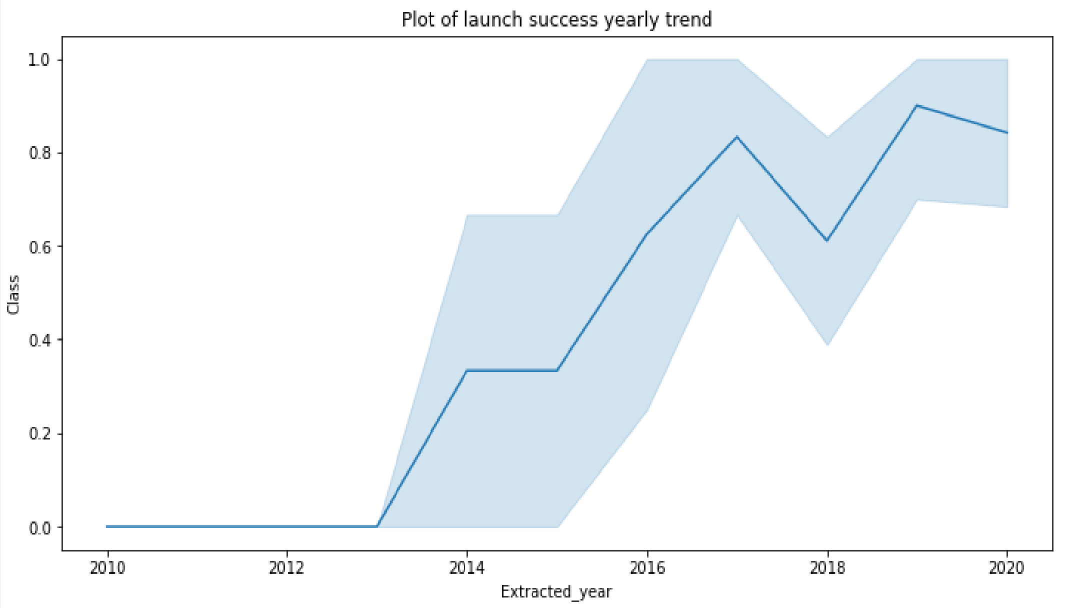


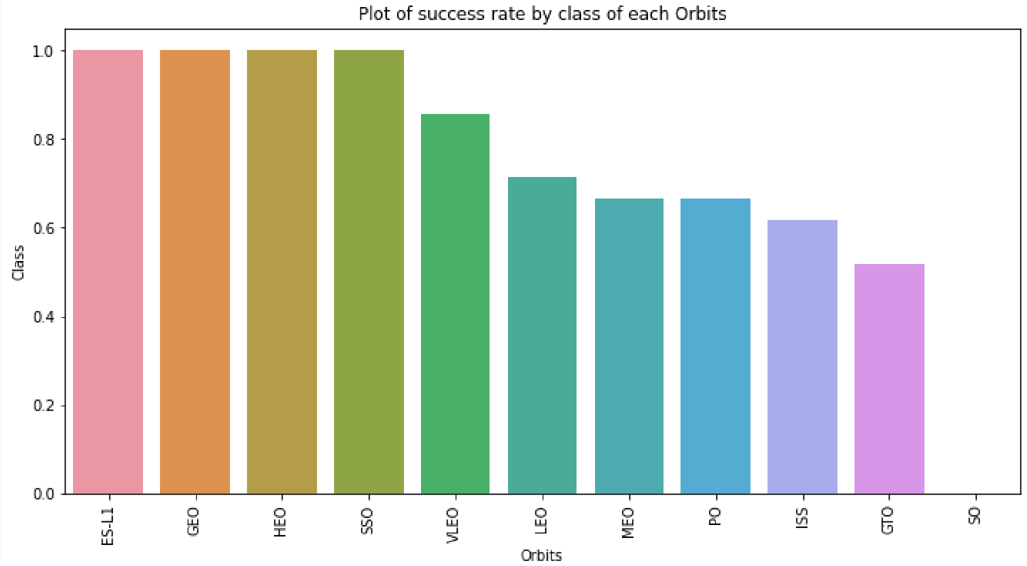
DATA WRANGLING

EDA with Data Visualization

We explored the data by visualizing the relationship between:

- Flight Number vs. Launch Site,
- Payload vs. Launch site,
- The success rate of each Orbit Type
- Flight number vs. Orbit
- The launch success yearly trend.





https://github.com/robbthe/IBM_Data_Science_Capstone/blob/main/jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

EDA WITH SQL

We loaded the SpaceX dataset into a PostgreSQL database. Then we applied EDA with SQL to get insights from the data. We queried the data to discover different attributes related to the launches such as Payload Mass, Launch Site, Orbit Types, etc.

The link to the notebook https://github.com/robbthe/IBM_Data_Science_Capstone/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

We 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.

We assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.

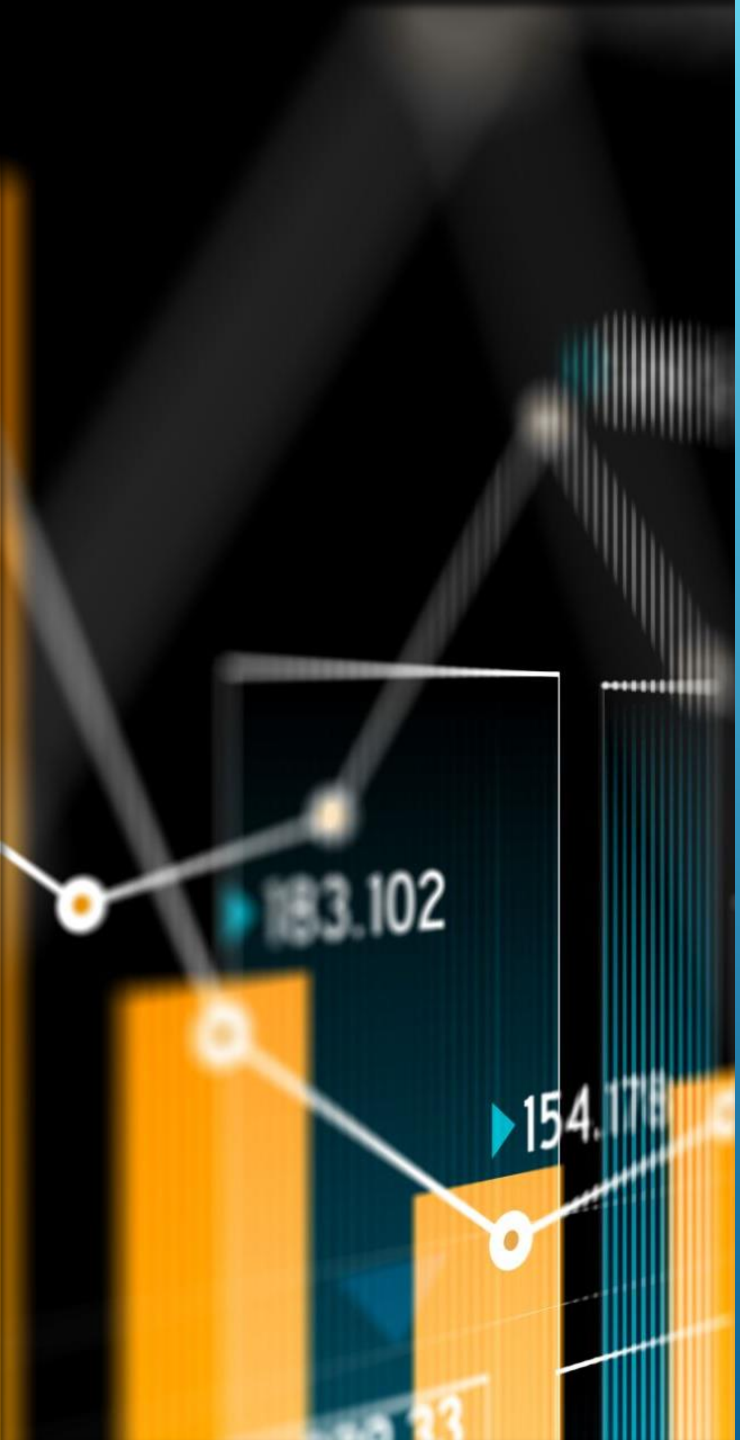
Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate.

We calculated the distances between a launch site to its proximities. We answered some question for instance:

Are launch sites near railways, highways and coastlines.

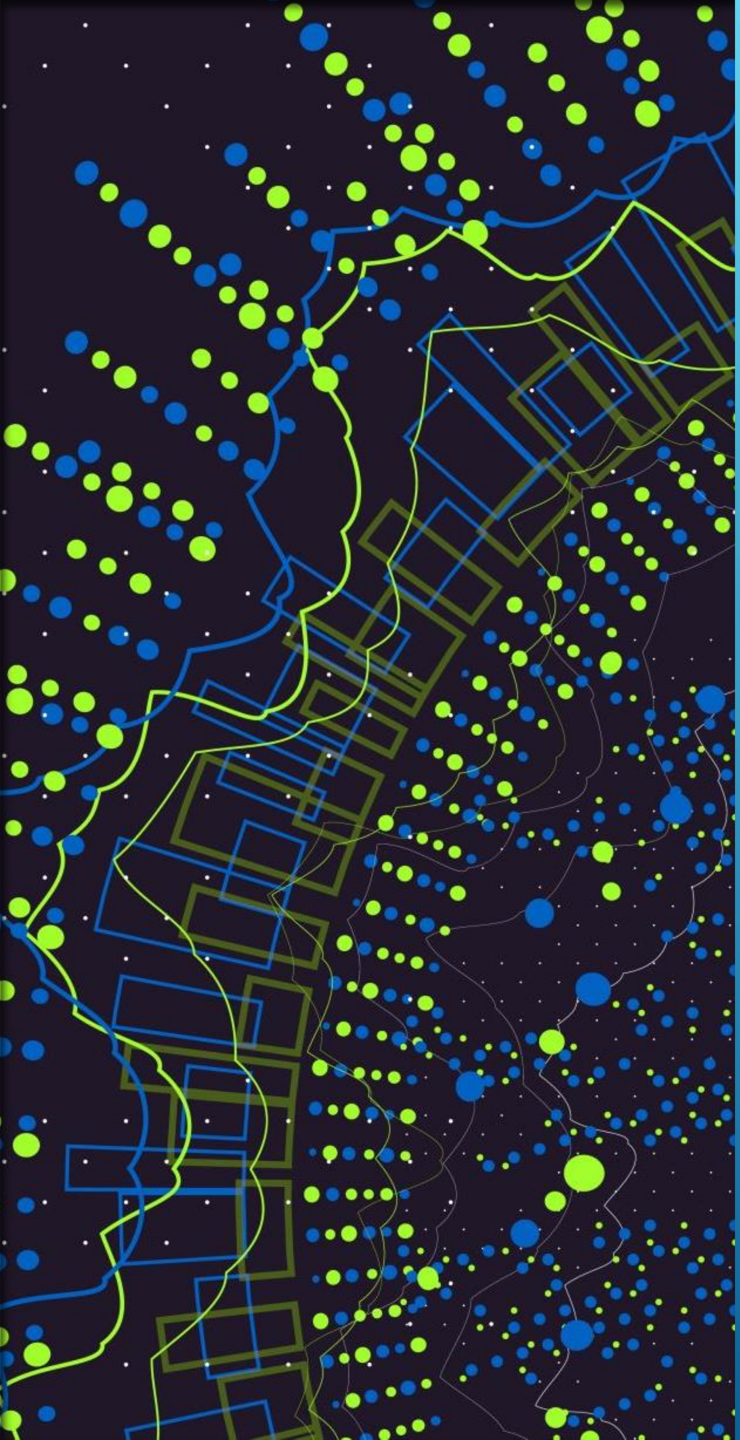
Do launch sites keep certain distance away from cities.

BUILD AN INTERACTIVE MAP WITH FOLIUM



- We built an interactive dashboard with Plotly dash
- We plotted pie charts showing the total launches by a certain sites
- We plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.
- The link to the notebook is https://github.com/chuksoo/IBM-Data-Science-Capstone-SpaceX/blob/main/app.py

BUILD A DASHBOARD WITH PLOTLY DASH



PREDICTIVE ANALYSIS (CLASSIFICATION)

- We loaded the data using numpy and pandas, transformed the data, split our data into two groups for training and testing.
- We built different machine learning models in order to tune different hyperparameters using GridSearchCV.
- We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- As a result, after comparative analysis, we found the best classification model.
- The link to the notebook is https://github.com/robbthe/IBM_Data_Science_Capstone/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

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

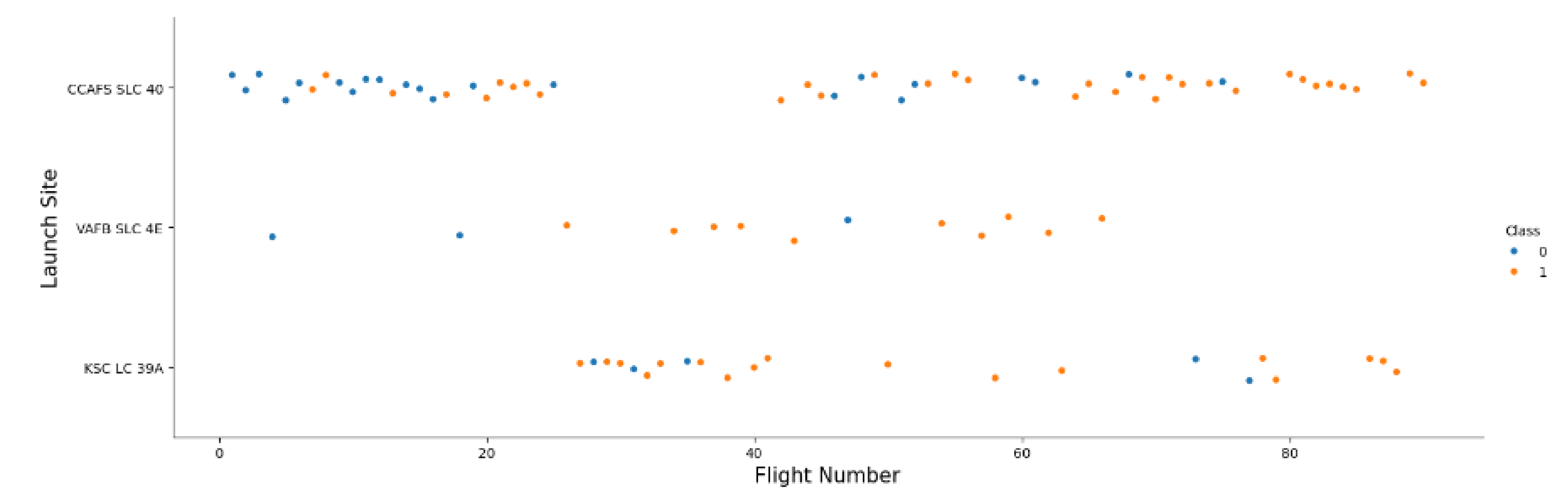
RESULTS

Section 2

Insights drawn from EDA

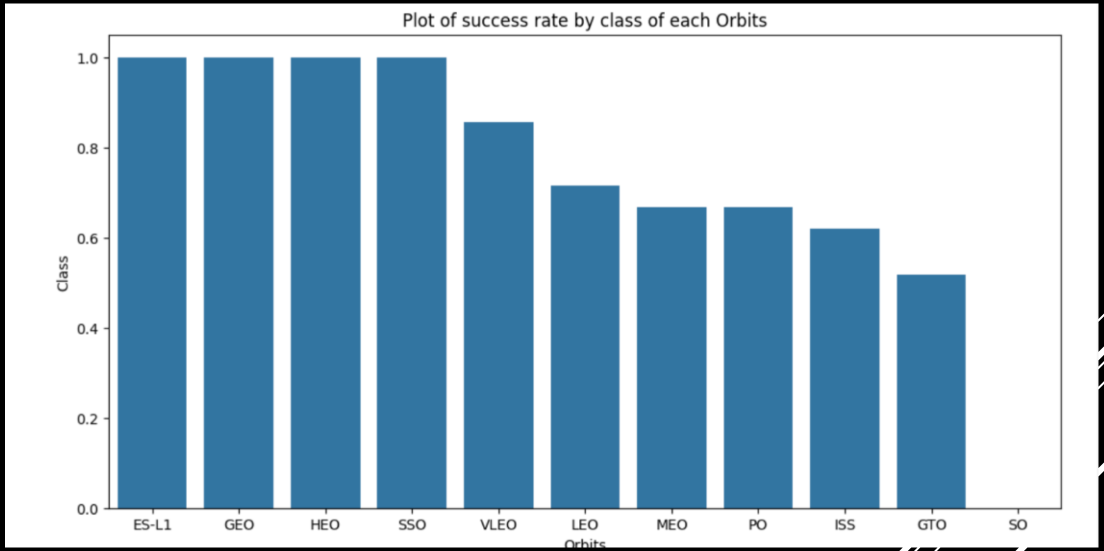
Flight Number vs. Launch Site

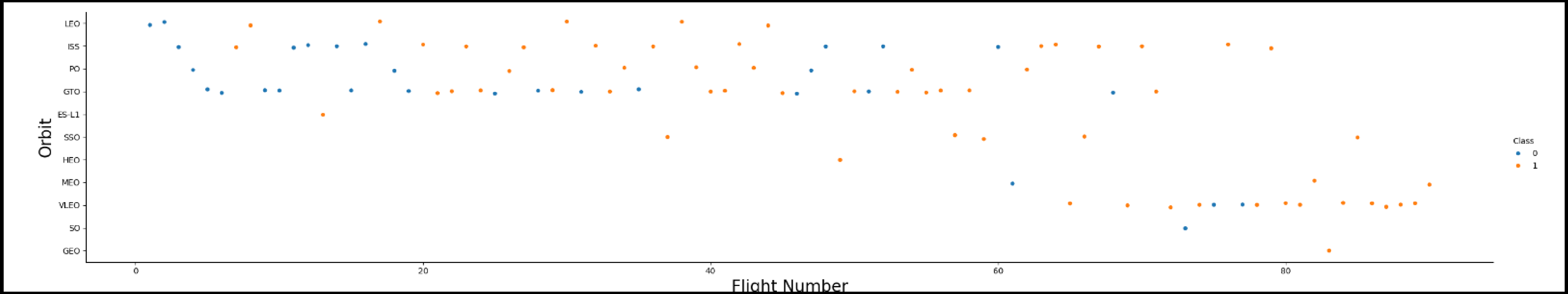
- From the plot, we found that the larger the flight amount at a launch site, the greater the success rate at a launch site.



Success Rate vs. Orbit Type

- From the plot, we can see that ES-L1, GEO, HEO, and SSO had the best success rates.



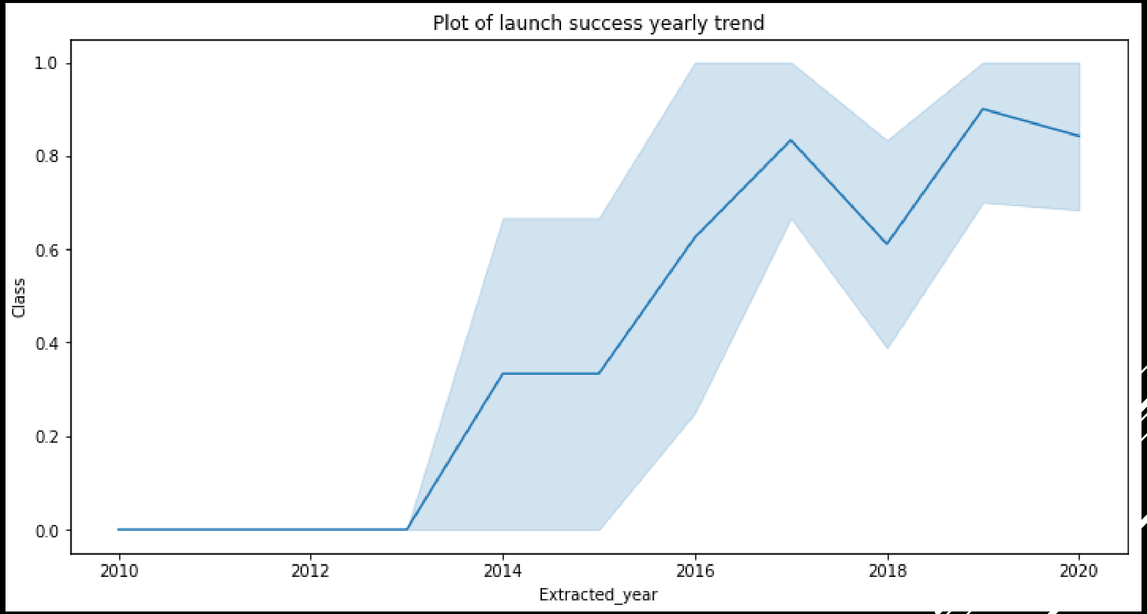


Flight Number vs. Orbit Type

- The plot 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.

Launch Success Yearly Trend

- From the plot, we can observe that there has been a continuous increase in the launch success, with the exception for a dip around 2019, likely due to COVID and the resulting economic strain.



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

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)
```

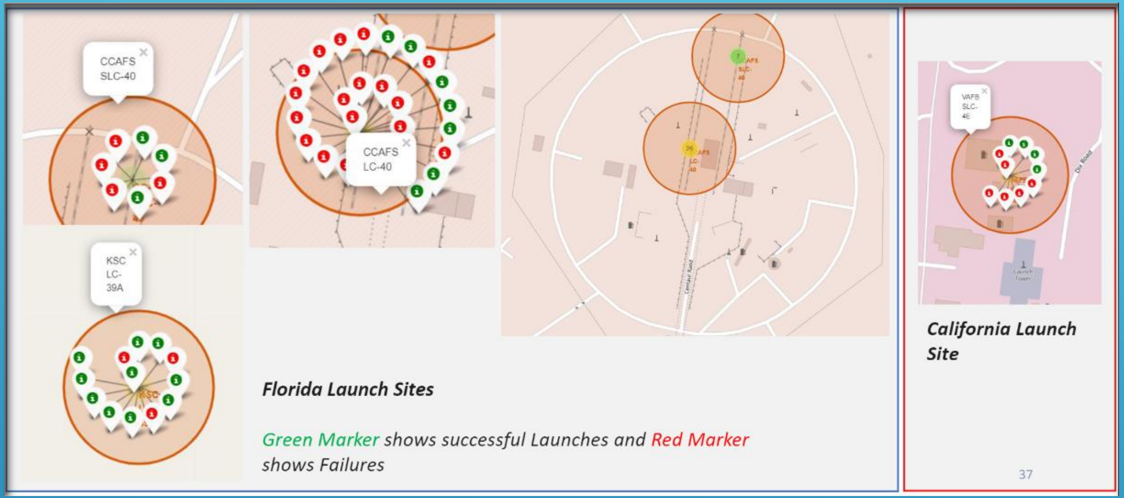
Out[19]:

	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

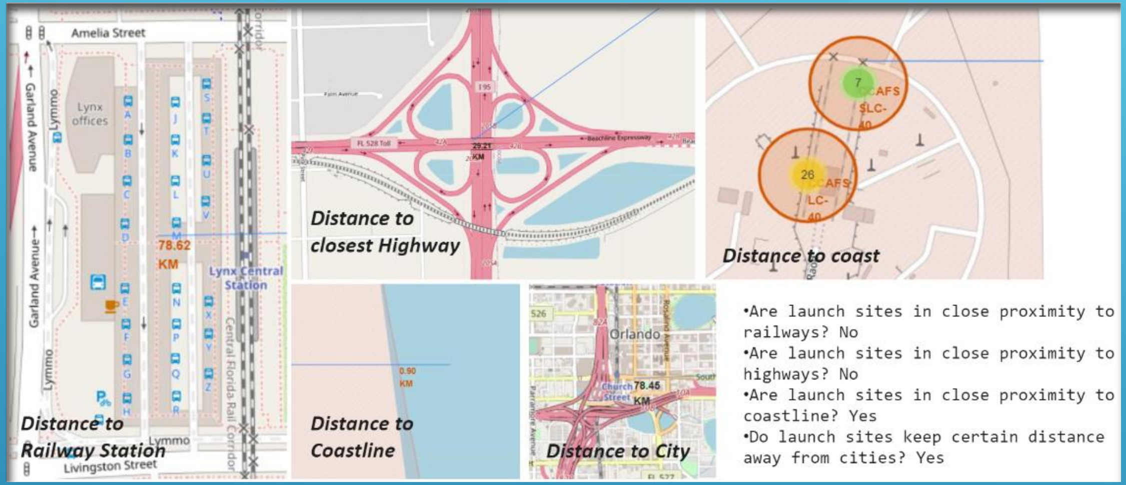
Section 4

Launch Sites Proximities Analysis

ALL LAUNCH SITES GLOBAL MAP MARKERS



MARKERS SHOWING LAUNCH SITES WITH COLOR LABELS

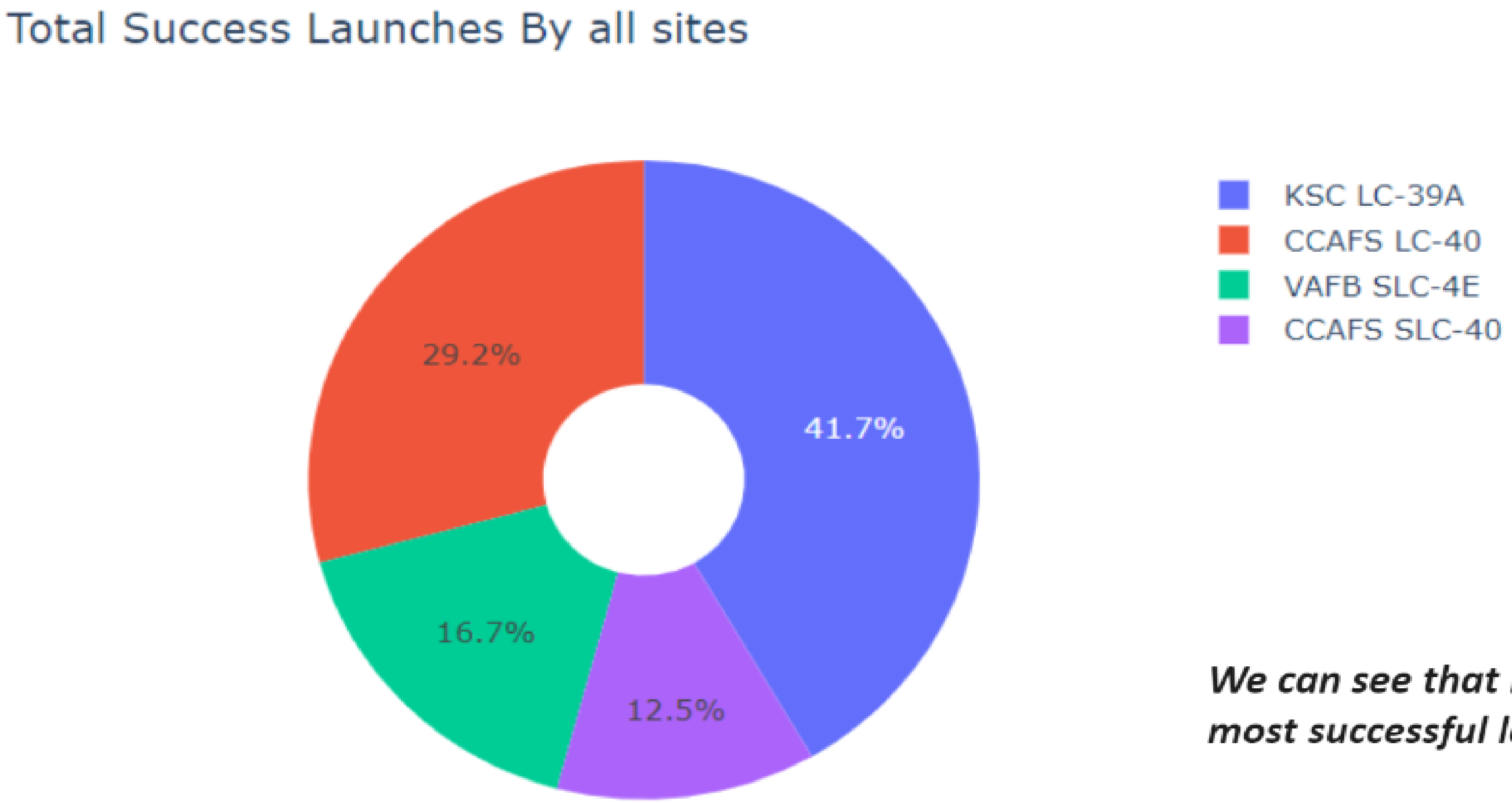


LAUNCH SITE DISTANCE TO LANDMARKS

Section 5

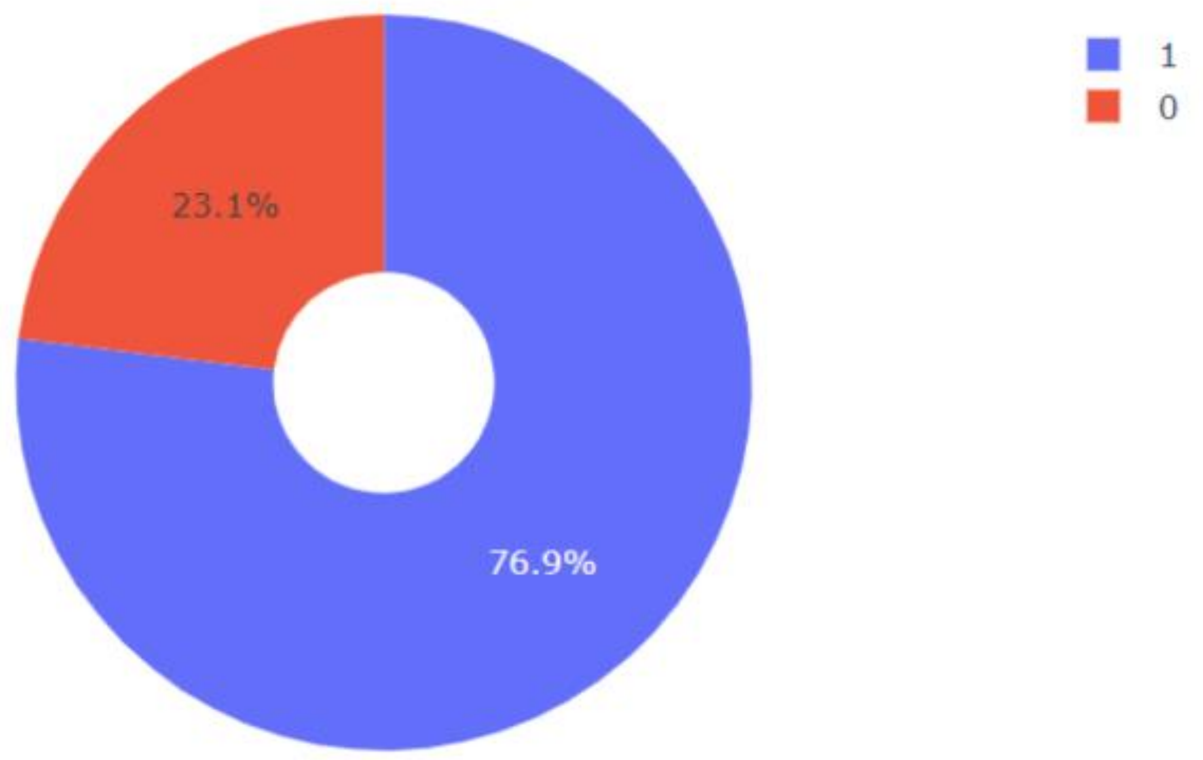
Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site



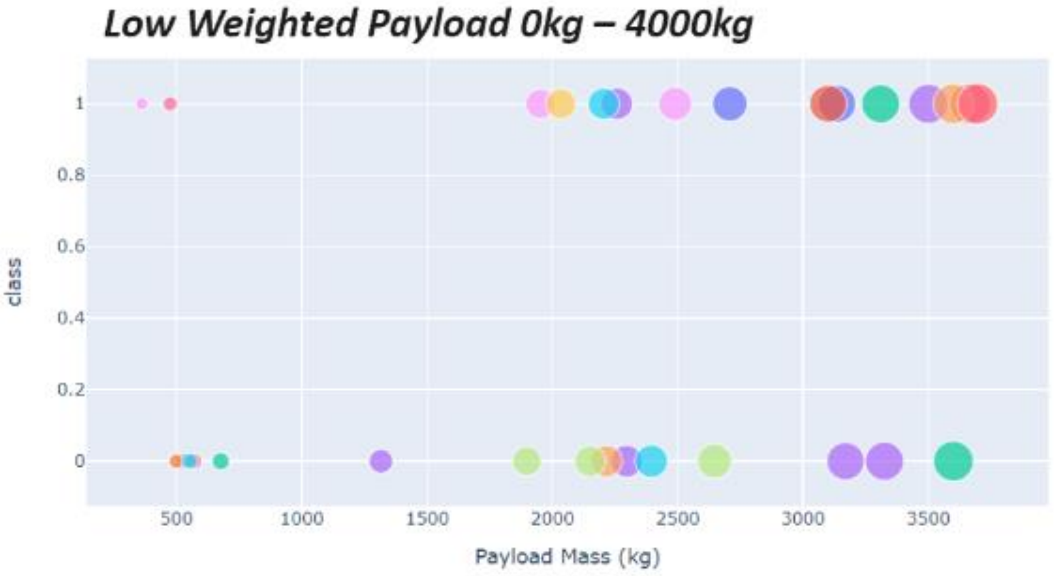
We can see that KSC LC-39A had the most successful launches from all the sites

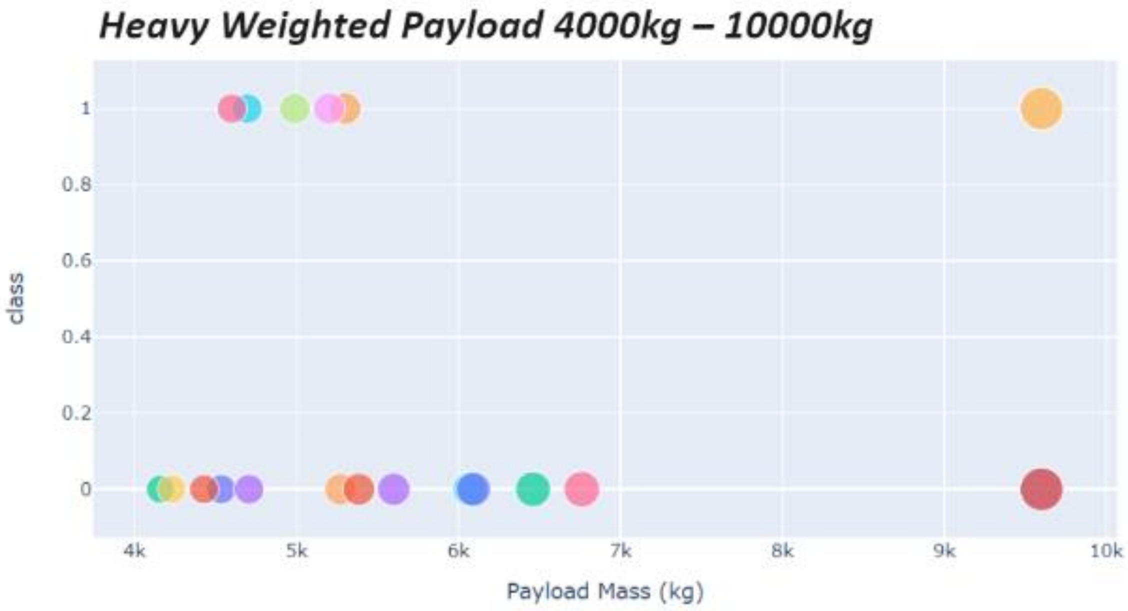
Pie chart showing the Launch site with the highest launch success ratio



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 6

Predictive Analysis (Classification)

Classification Accuracy

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

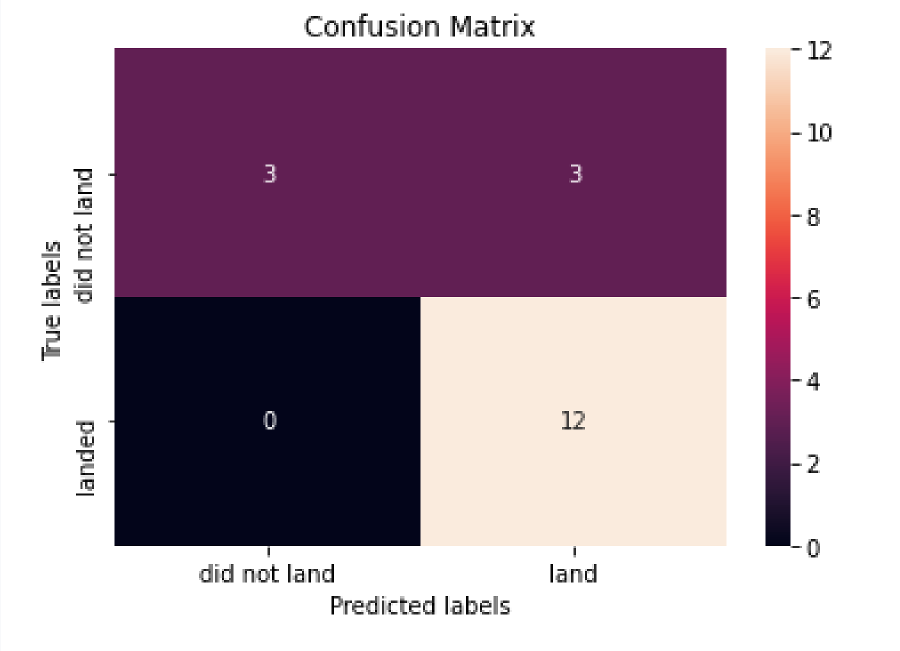
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

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.



- We can conclude 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.

CONCLUSIONS

Thank you!