Winning Space Race with Data Science

Sathish
01MAR2024

Outline

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

GitHub: https://github.com/sandysmiles/Python-first-project

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

  SpaceX is a leading company in the commercial space age, offering affordable space travel through reusable rockets, most notably the Falcon 9. The Falcon 9's first stage is crucial, as it performs most of the work and can be reused, significantly reducing launch costs. To compete with SpaceX, determine launch prices and predict first-stage reusability by analyzing SpaceX data and training a machine learning model on public information. The goal is to compete with SpaceX by offering affordable launches while minimizing risks associated with first-stage reusability.

- Problems you want to find answers
  - Determine which features impact 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

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

Data Collection

How data sets were collected.

1. Data collection was done using get request to the SpaceX API.
2. Next, we decoded the response content as a Json using .json() function call and turn it into a pandas dataframe using .json_normalize().
3. We then cleaned the data, checked for missing values and fill in missing values where necessary.
4. In addition, we performed web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.
5. 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

We used the get request to the SpaceX API to collect data.

Cleaned the requested data by filtering the dataframe.

Performed Basic data wrangling and formatting by dropping no-value columns.

GitHub: https://github.com/sandysmiles/Python-first-project/blob/main/week1-spacex-data-collection-api%20(2).ipynb

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

```
Use json_normalize meethod to convert the json result into a dataframe
resp = requests.get(static_json_url)
data = pd.json_normalize(resp.json())
data.shape
```

Filter the dataframe to only include Falcon 9 launches:

```
Assuming 'rocket_name' is the column containing the rocket names
data_falcon9 = df[df['BoosterVersion'] == 'Falcon 9']
data_falcon9

data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

Data Wrangling: Dealing with Missing Values:

```
Calculate the mean value of PayloadMass column
pay_mean = data_falcon9['PayloadMass'].mean()

Replace the np.nan values with its mean value
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, pay_mean)
data_falcon9.isnull().sum()
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Data Collection - Scraping

Web scrapping “Falcon” launch records with BeautifulSoup.

Parsed the table and converted it into a pandas dataframe.

GitHub URL:https://github.com/sandysmiles/Python-first-project/blob/main/week1-spacex-Data%20wrangling.ipynb

Create a BeautifulSoup object from the HTML response

```
Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(html_content, 'html.parser')
```

Extract all column/variable names from the HTML table header

```
Use the find_all function in the BeautifulSoup object, with element type `table`
html_tables = soup.find_all('table')
```

Create a data frame by parsing the launch HTML tables:

```
First create an empty dictionary with keys from the extracted column names in the previous task.
launch_dict= dict.fromkeys(column_names)
```

Parsing through the html:

```
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"):
```

The created dictionary will be converted into a Pandas dataframe:

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

Data Wrangling

Performed exploratory data analysis 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.

GitHub URL:https://github.com/sandysmiles/Python-first-project/blob/main/Week1-spacex-Data%20wrangling%20(2).ipynb

Calculate the number of launches on each site:

```
# Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()
```

Calculate the number and occurrence of each orbit:

```
# Apply value_counts on Orbit column
df['Orbit'].value_counts()
```

Calculate the number and occurence of mission outcome of the orbits:

```
# landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()
landing_outcomes
```

Create a landing outcome label from Outcome column:

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = df['Outcome'].map(lambda x: 0 if x in bad_outcomes else 1)
landing_class

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

Success Rate:

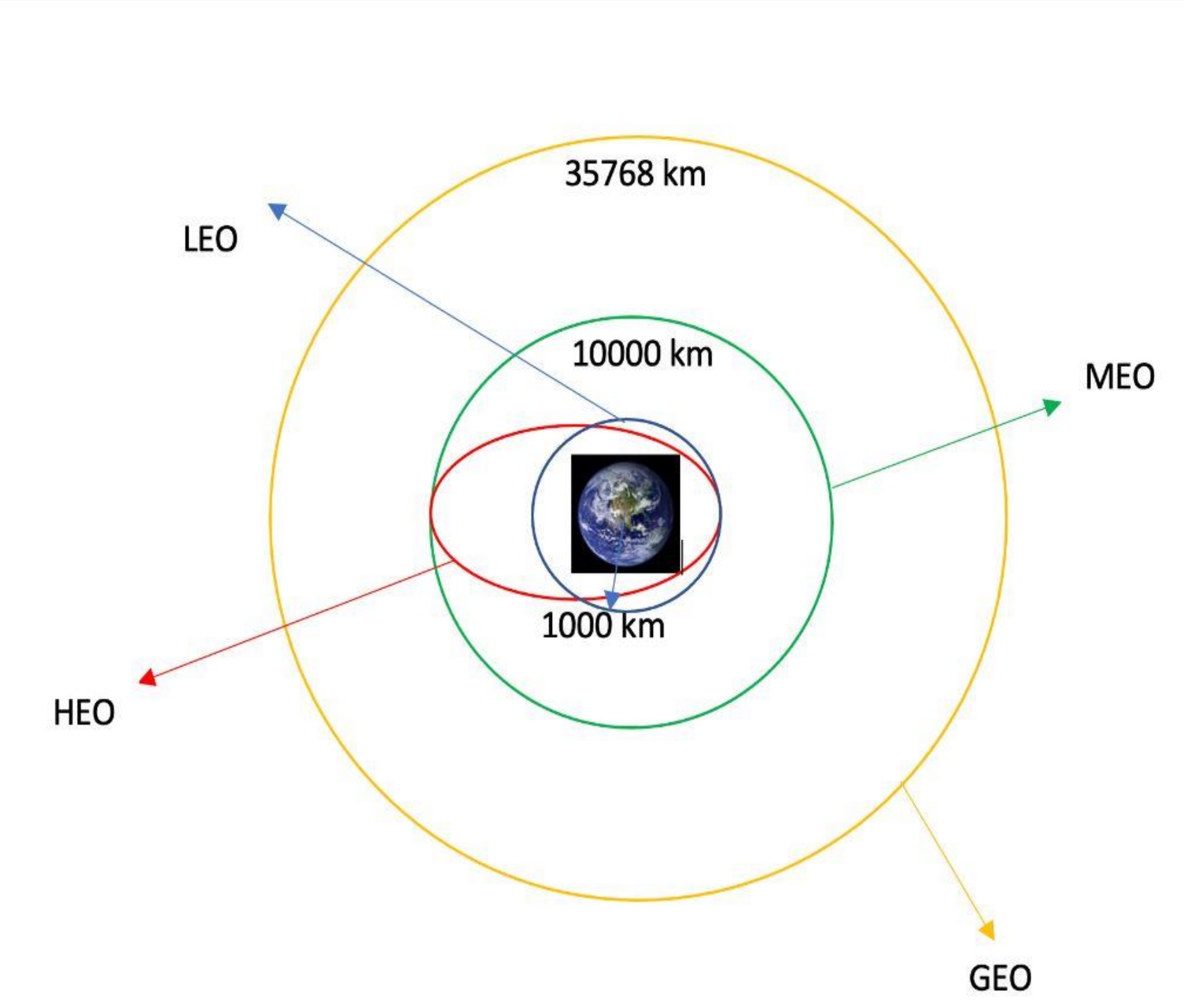
```
df["Class"].mean()
0.6666666666666666
```

EDA with Data Visualization

Explored the data by visualizing the relationship:

- 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.

GitHub URL: https://github.com/sandysmiles/Python-first-project/blob/main/Week2-eda-dataviz.ipynb.jupyterlite.ipynb



EDA with SQL

Loaded the SpaceX dataset into a PostgreSQL database without leaving the jupyter notebook.

We applied EDA with SQL to get insight from the data. Execute queries to find out:

- 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.

Add the GitHub URL: https://github.com/sandysmiles/Python-first-project/blob/main/week2-eda-sql-coursera_sqllite.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, and 1 for success.

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

Calculated the distances between a launch site to its proximities.

Found nearby distance: -

Are launch sites near railways, highways and coastlines.

Do launch sites keep certain distance away from cities.

Add the GitHub URL: https://github.com/sandysmiles/Python-first project/blob/main/Week3_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

Built an interactive dashboard with Plotly dash.

Plotted pie charts showing the total launches by a certain sites.

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

Add the GitHub URL: https://github.com/sandysmiles/Python-first-project/blob/main/spacex_dash_app.py

https://sathishsmile-8050.theiadockernext-0-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/

Predictive Analysis (Classification)

- Loaded 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.
- Used accuracy as the metric for our model, improved the model using Feature engineering and algorithm tuning.
- Found the best performing classification model.

Add the GitHub URL: https://github.com/sandysmiles/Python-first-project/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Results

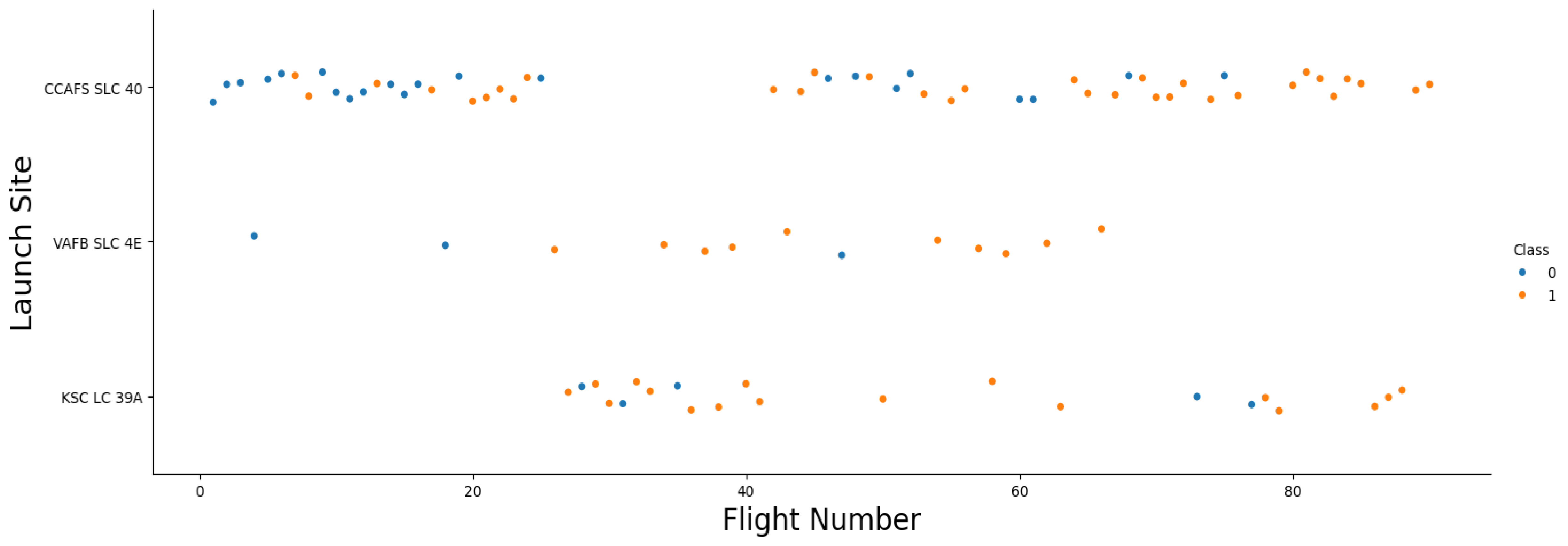
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

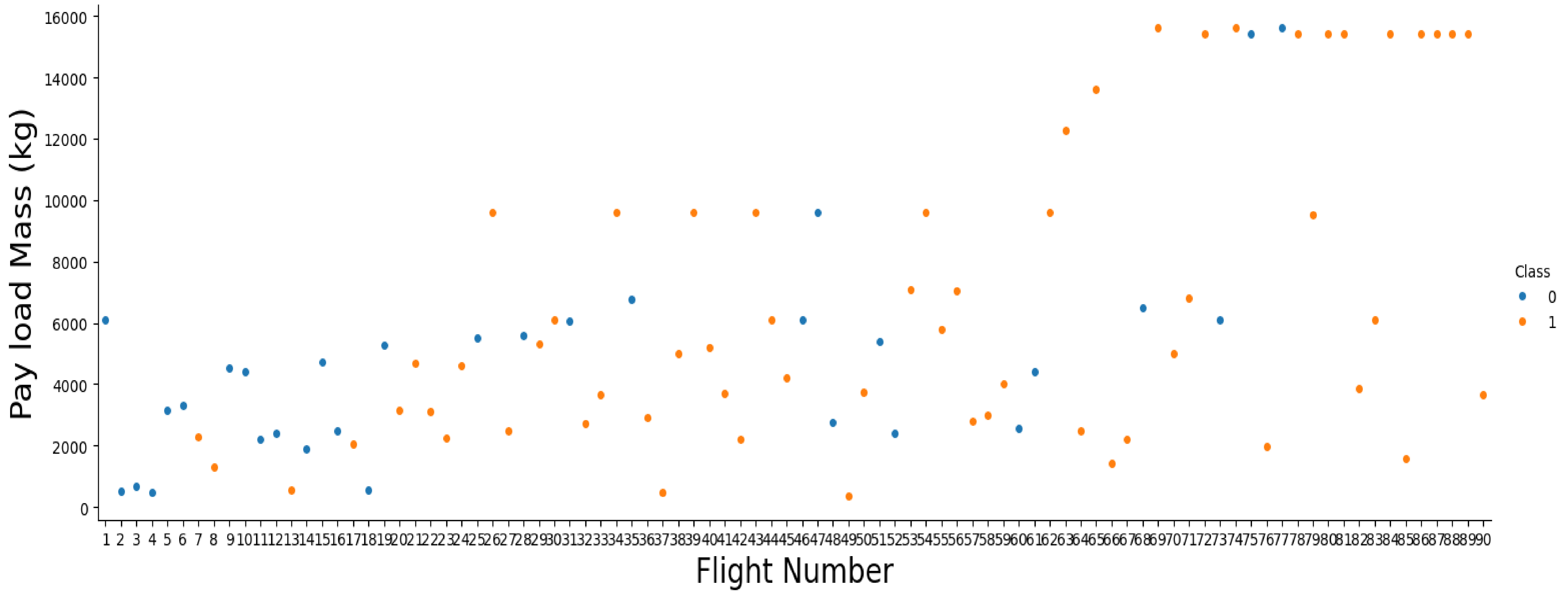
Flight Number vs. Launch Site

- From the scatter 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

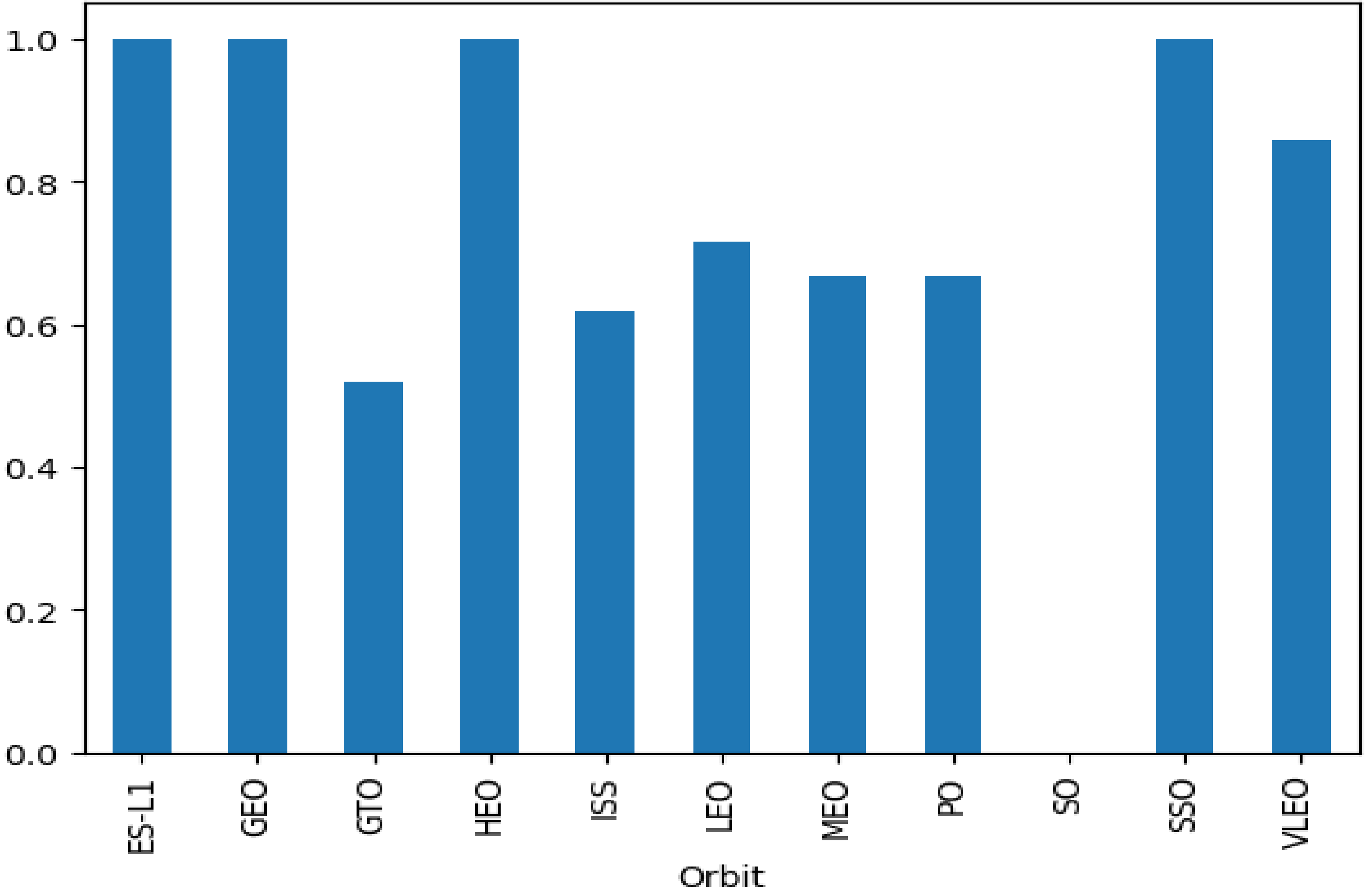
We see that different launch sites have different success rates. CCAFS LC-40 has a success rate of 60 %, while KSC LC-39A and VAFB SLC 4E has a success rate of 77%.



Success Rate vs. Orbit Type

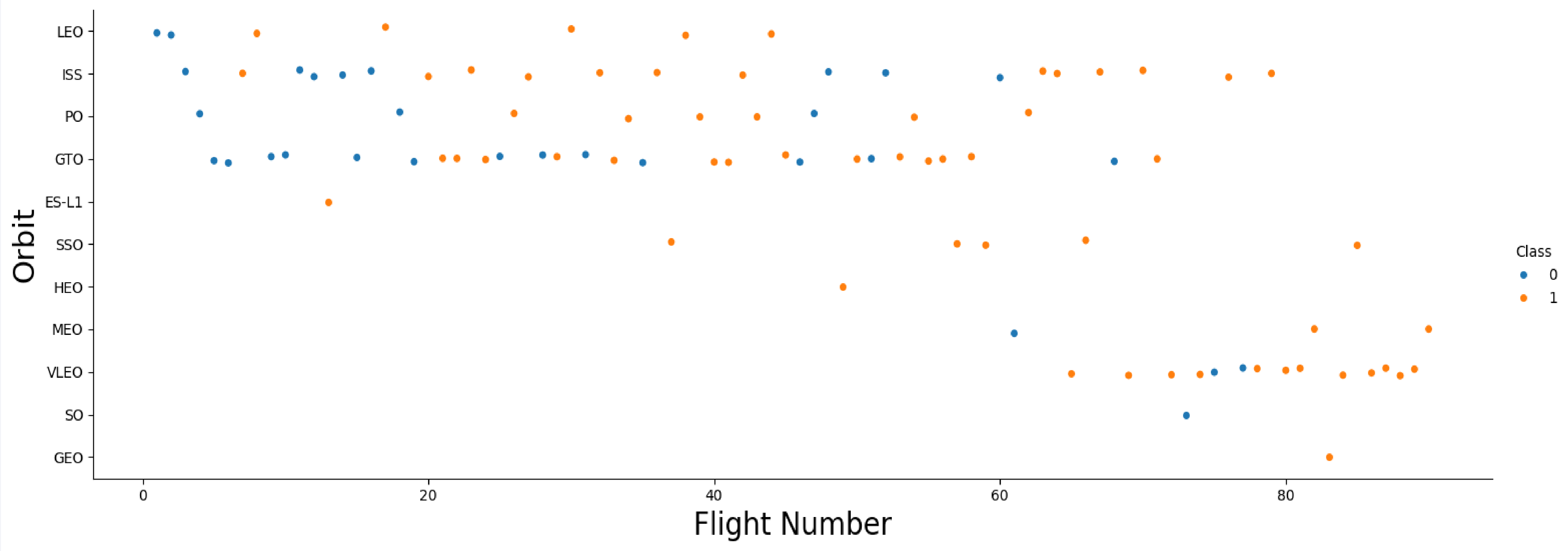
The most success rate:

- ES-L1
- GEO
- HEO
- SSO
- VLEO



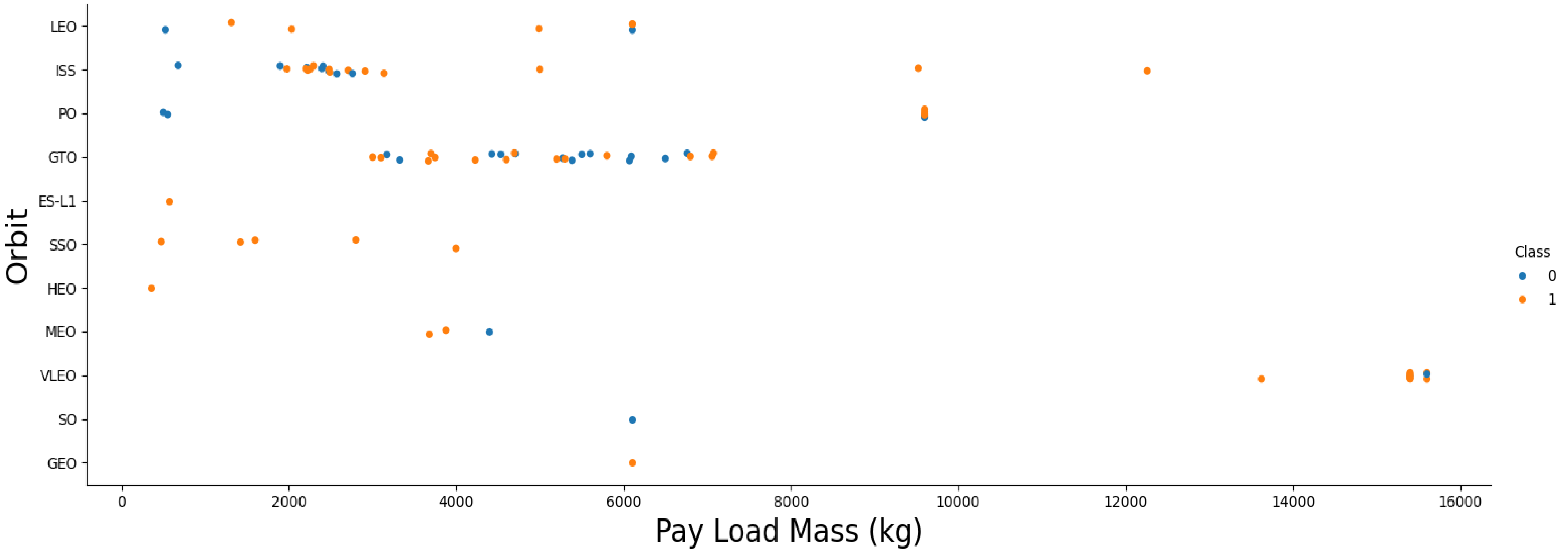
Flight Number vs. Orbit Type

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.



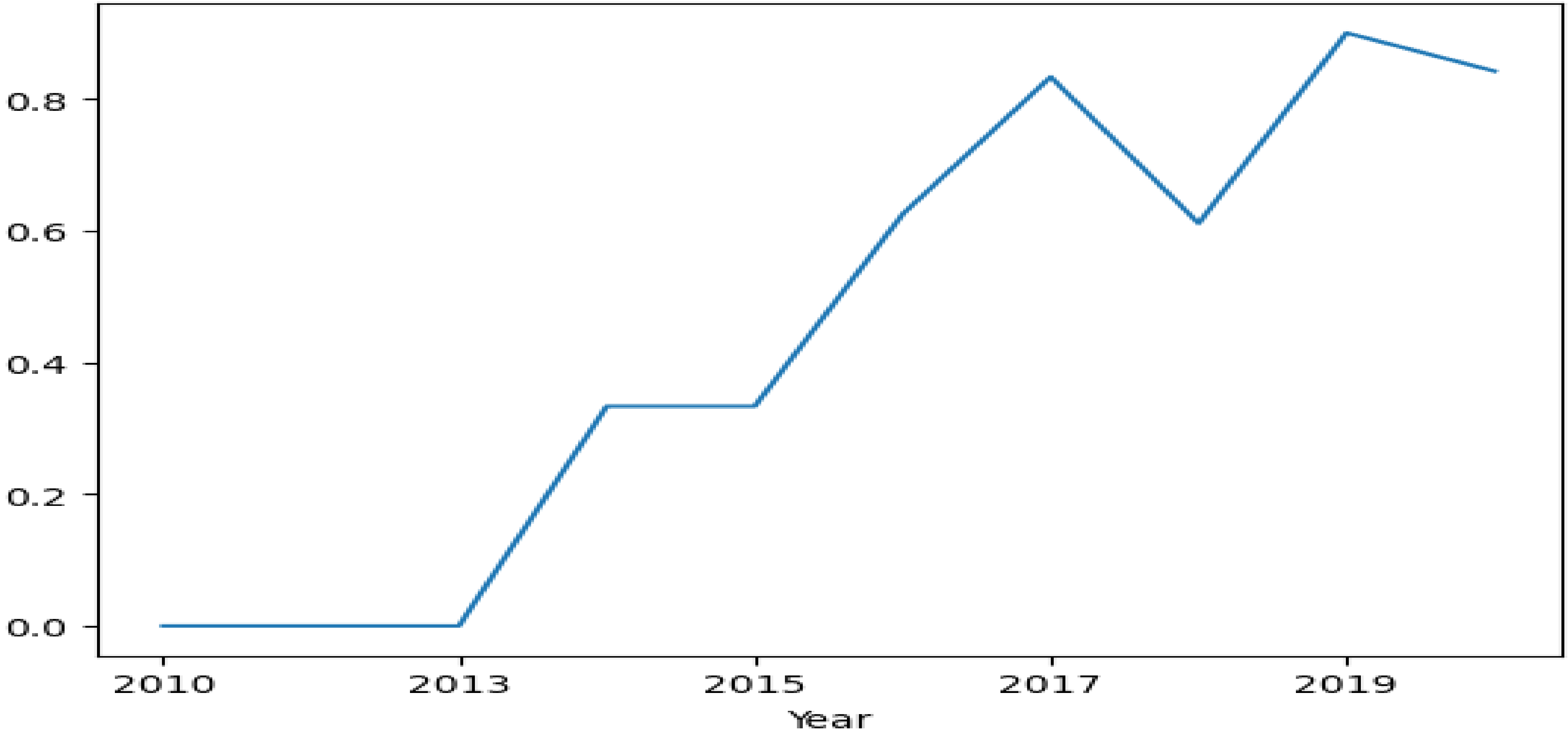
Payload vs. Orbit Type

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 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.

Task 1

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

In [9]:
```
%sql SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL ORDER BY 1;
```

```
 * sqlite:///my_data1.db
Done.
```

Out[9]:

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

Launch Site Names Begin with 'CCA'

Query to display 5 records where launch sites begin with `CCA`

Task 2

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

In [11]:
```
%sql SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
```

```
 * sqlite:///my_data1.db
Done.
```

Out[11]:

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	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

Calculated the total payload carried by boosters from NASA (CRS) as 111268 using the query below

Task 3

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

```
[12]: %sql SELECT SUM(PAYLOAD_MASS__KG_) AS TOTAL_PAYLOAD FROM SPACEXTBL WHERE PAYLOAD  LIKE '%CRS%';
```

```
 * sqlite:///my_data1.db
Done.
```

[12]:

TOTAL_PAYLOAD
111268

Average Payload Mass by F9 v1.1

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

111268

Task 4

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

In [13]:
```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS AVG_PAYLOAD FROM SPACEXTBL WHERE BOOSTER_VERSION = 'F9 v1.1';
```

```
 * sqlite:///my_data1.db
Done.
```

Out[13]:

AVG_PAYLOAD
2928.4

First Successful Ground Landing Date

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

Task 5

List the date when the first succesful landing outcome in ground pad was acheived.

Hint:Use min function

In [14]:
```
%sql SELECT MIN(DATE) AS FIRST_SUCCESS_GP FROM SPACEXTBL WHERE LANDING_OUTCOME = 'Success (ground pad)';
```

```
 * sqlite:///my_data1.db
Done.
```

Out[14]:

FIRST_SUCCESS_GP
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

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

Task 6

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

In [15]:

```
%sql SELECT DISTINCT BOOSTER_VERSION FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000 AND LANDING_OUTCOME = 'Suc
```

```
 * sqlite:///my_data1.db
Done.
```

Out[15]:

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

Total Number of Successful and Failure Mission Outcomes

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

Task 7

List the total number of successful and failure mission outcomes

[15]:
```
%sql SELECT COUNT(MISSION_OUTCOME) AS Success_Outcome FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE '%Success%';
```

```
 * sqlite:///my_data1.db
Done.
```

[15]:

Success_Outcome
100

[16]:
```
%sql SELECT COUNT(MISSION_OUTCOME) AS Failure_Outcome FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE '%Failure%';
```

```
 * sqlite:///my_data1.db
Done.
```

[16]:

Failure_Outcome
1

Boosters Carried Maximum Payload

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

Task 8

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

```
[23]: %sql SELECT DISTINCT BOOSTER_VERSION FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL) ORDER BY
```

```
 * sqlite:///my_data1.db
Done.
```

[23]:

Booster_Version
F9 B5 B1048.4
F9 B5 B1048.5
F9 B5 B1049.4
F9 B5 B1049.5
F9 B5 B1049.7
F9 B5 B1051.3
F9 B5 B1051.4
F9 B5 B1051.6
F9 B5 B1056.4
F9 B5 B1058.3
F9 B5 B1060.2
F9 B5 B1060.3

2015 Launch Records

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

Task 9

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

[20]:
```
%sql SELECT substr(Date, 6, 2) AS Month, BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WHERE LANDING_OUTCOME = 'Failure (drone ship)' AN
```

```
 * sqlite:///my_data1.db
Done.
```

[20]:

Month	Booster_Version	Launch_Site
01	F9 v1.1 B1012	CCAFS LC-40
04	F9 v1.1 B1015	CCAFS LC-40

Task 10

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

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.

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

Task 10

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.

[24]:
```
%sql SELECT LANDING_OUTCOME, COUNT(LANDING_OUTCOME) AS QTY FROM SPACEXTBL WHERE LANDING_OUTCOME LIKE '%Success%' AND Date BETWEEN '201
```

```
 * sqlite:///my_data1.db
Done.
```

[24]:

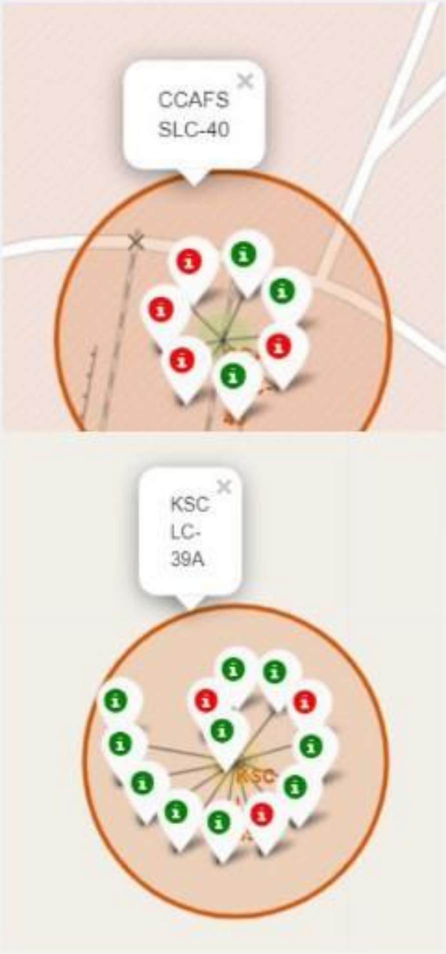
Landing_Outcome	QTY
Success (drone ship)	5
Success (ground pad)	3

Section 4

Launch Sites Proximities Analysis

All launch sites global map markers

Markers showing launch sites with color labels



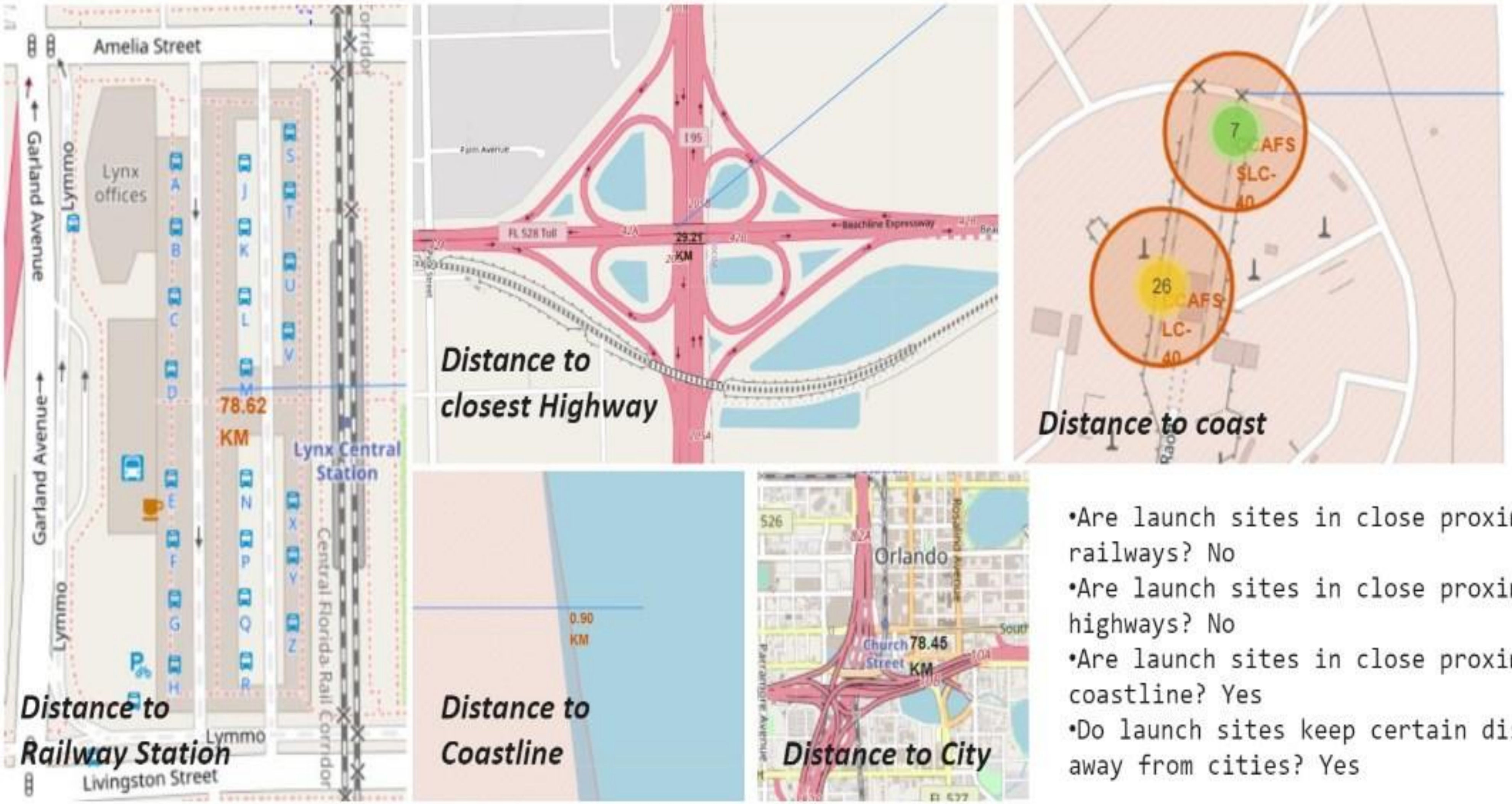
Florida Launch Sites

Green Marker shows successful Launches and Red Marker shows Failures

California Launch Site

37

Launch Site distance to landmarks

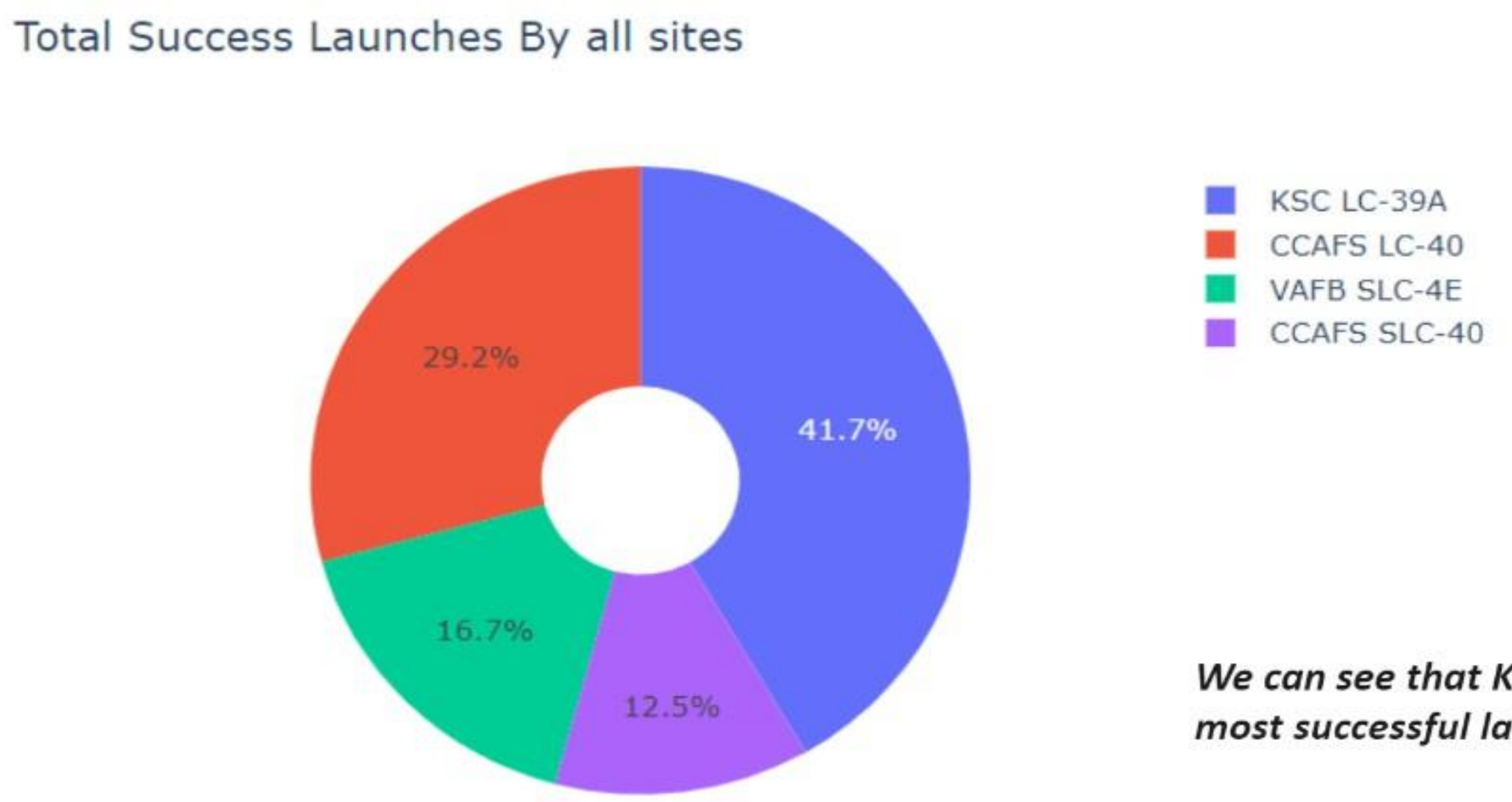


- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

Section 5

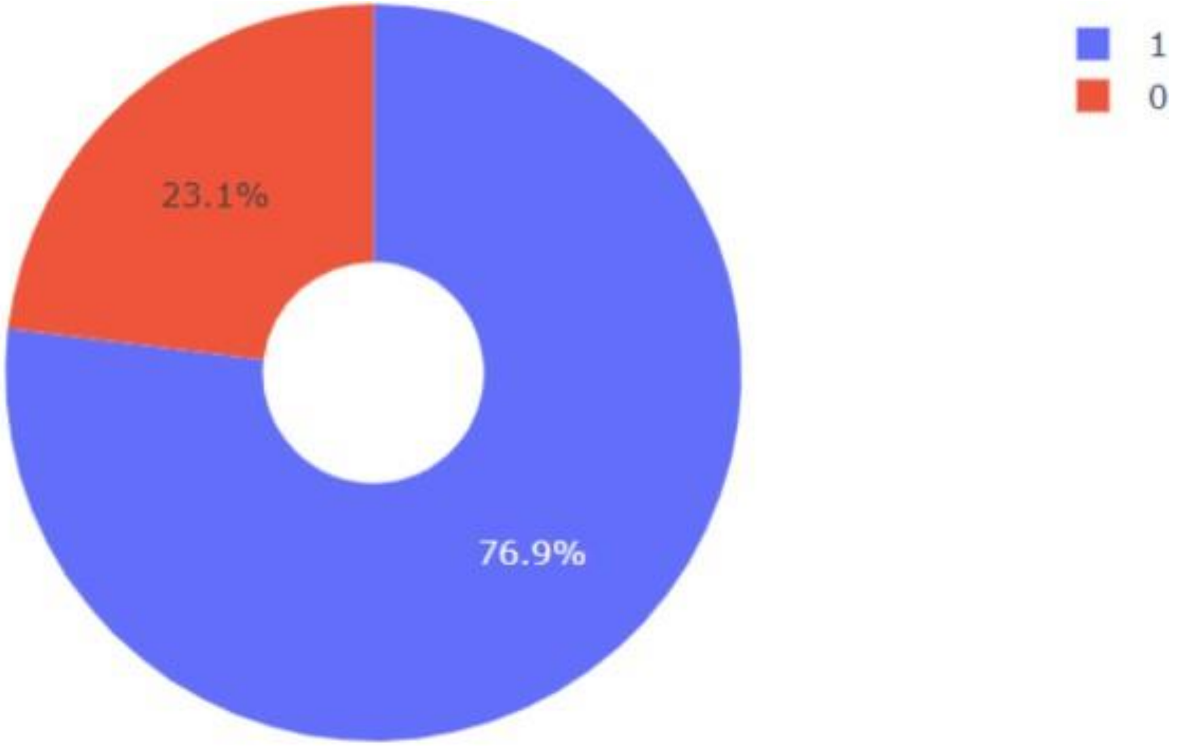
Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site



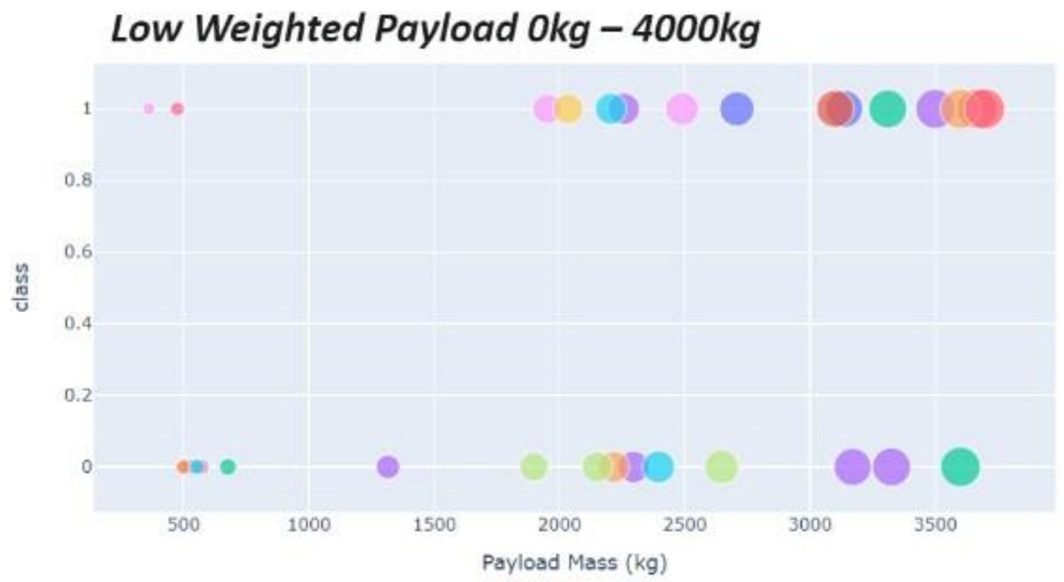
We can see that KSC LC-39A had the most successful launches from all the sites

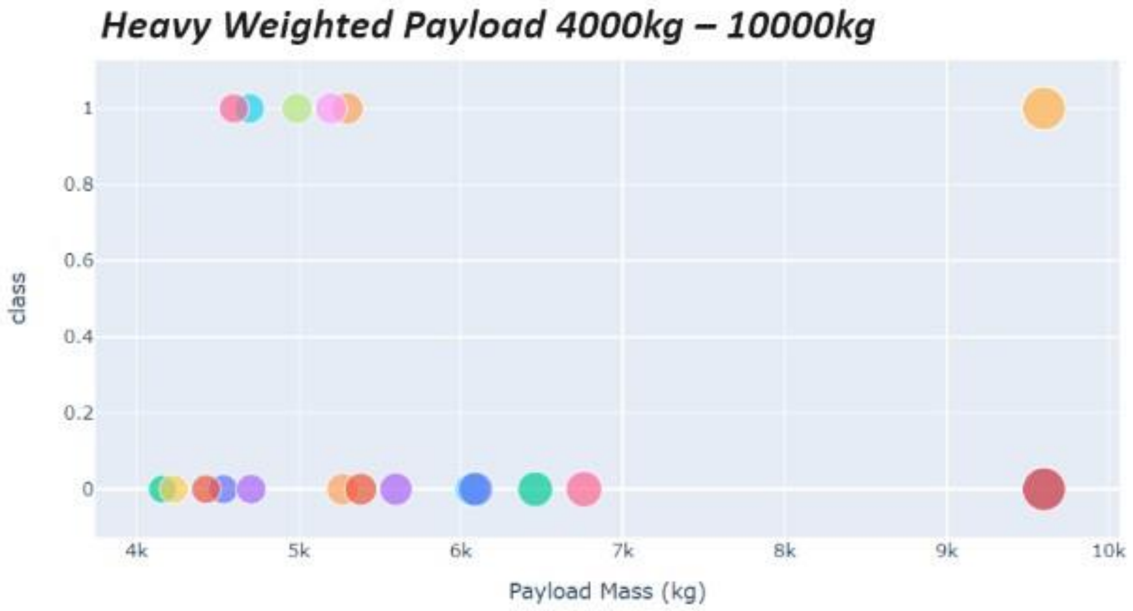
Pie chart showing the Launch site with the highest launch success ratio



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

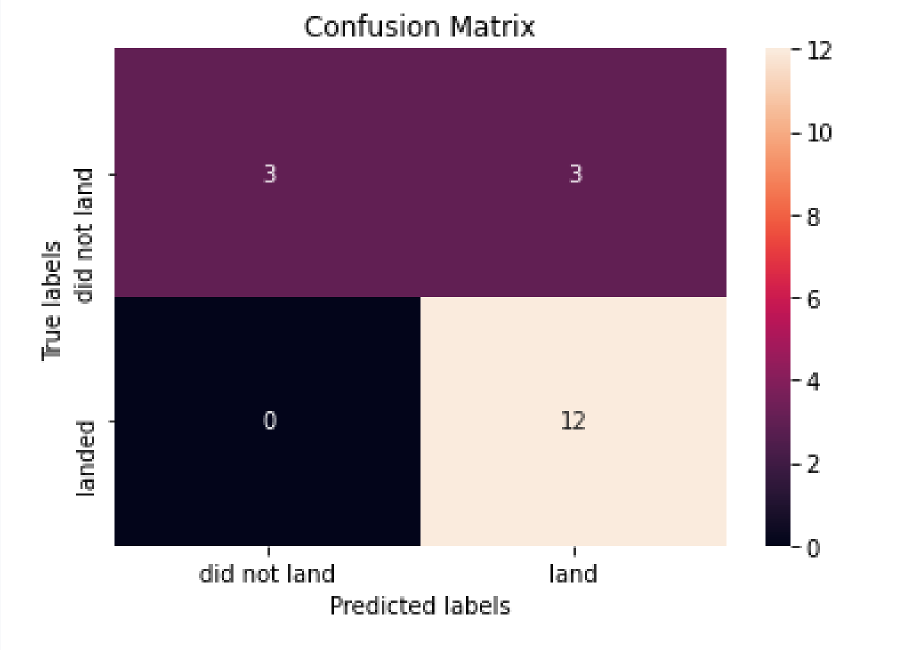
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

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.



Conclusions

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.

Thank you!