IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Chukwuemeka Okoli
2nd December 2021

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

  On its website, Space X promotes Falcon 9 rocket launches at a price of 62 million dollars; in comparison, other suppliers charge up to 165 million dollars per launch; a large portion of the cost savings are attributable to Space X's ability to reuse the first stage. Thus, we can calculate the cost of a launch if we can ascertain if the first stage will land. This data may be utilized should another business wish to submit a bid for a rocket launch against Space X. The project's objective is to build a machine learning pipeline that can forecast whether or not the initial stage will land successfully.

- Problems you want to find answers

  - What elements influence the rocket's likelihood of a successful landing?
  - The way different features interact to determine the likelihood of a successful landing.
  - What operational circumstances must exist in order to guarantee the success of the 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

- The data was collected using various methods
  - Utilising a receive call to the SpaceX API, data was gathered.
  - Next, we used the.json() function call to decode the response content as JSON and the.json_normalize() method to convert it into a pandas Dataframe.
  - After that, we cleaned the data, looked for any missing values, and, if needed, filled them in.
  - Additionally, we used BeautifulSoup to perform web scraping of Wikipedia to find launch records for the Falcon 9.
  - The goal was to take the HTML table containing the launch records, parse it, and then transform it into a pandas Dataframe so that it could be examined further.

Data Collection – SpaceX API

- We used 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 [Notebook]

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

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

Python

We should see that the request was successfull with the 200 status response code

```python
response.status_code
```

Python

```
200
```

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

```python
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

Python

Data Collection - Scraping

- We applied web scrapping to webscrap Falcon 9 launch records with BeautifulSoup
- We parsed the table and converted it into a pandas dataframe.
- The link to the notebook is [Notebook]()

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

Python

Next, request the HTML page from the above URL and get a `response` object

TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

```python
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url)
```

Python

Create a `BeautifulSoup` object from the HTML `response`

```python
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.content, "html.parser")
```

Python

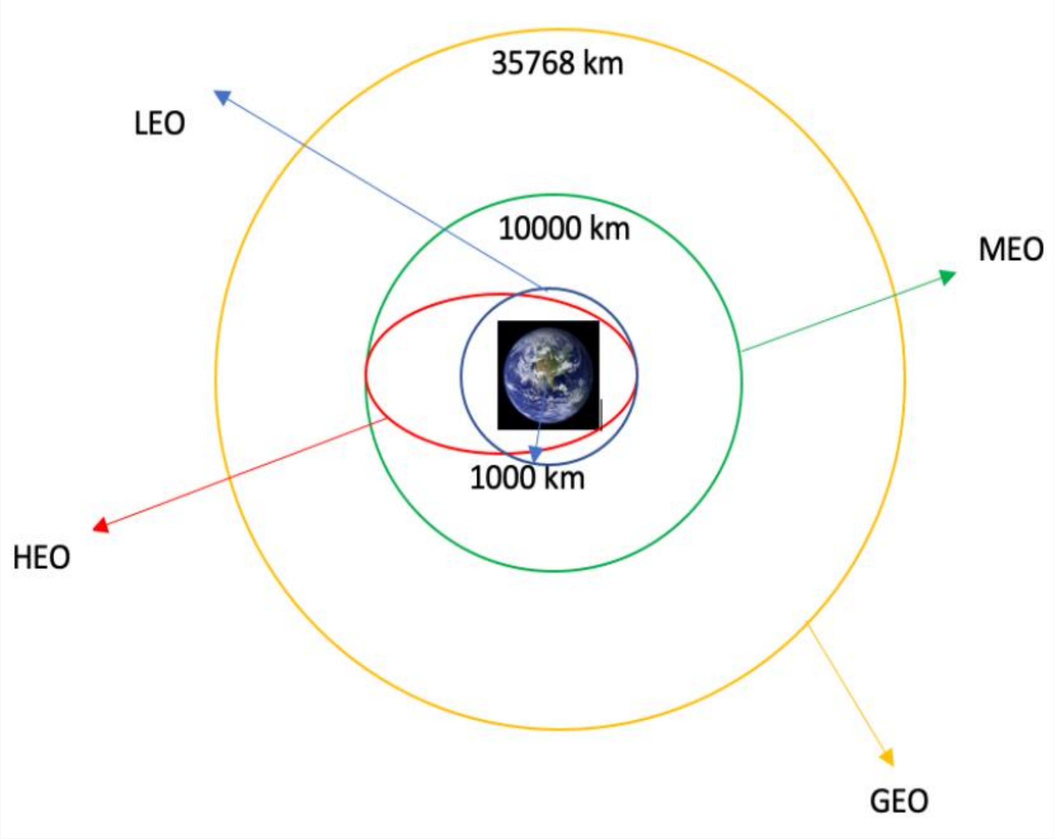
Print the page title to verify if the `BeautifulSoup` object was created properly

```python
# Use soup.title attribute
title_of_webpage = soup.title.string
print(f"Title of the webpage is: {title_of_webpage}")
```

Python

```
Title of the webpage is: List of Falcon 9 and Falcon Heavy launches - Wikipedia
```

Data Wrangling

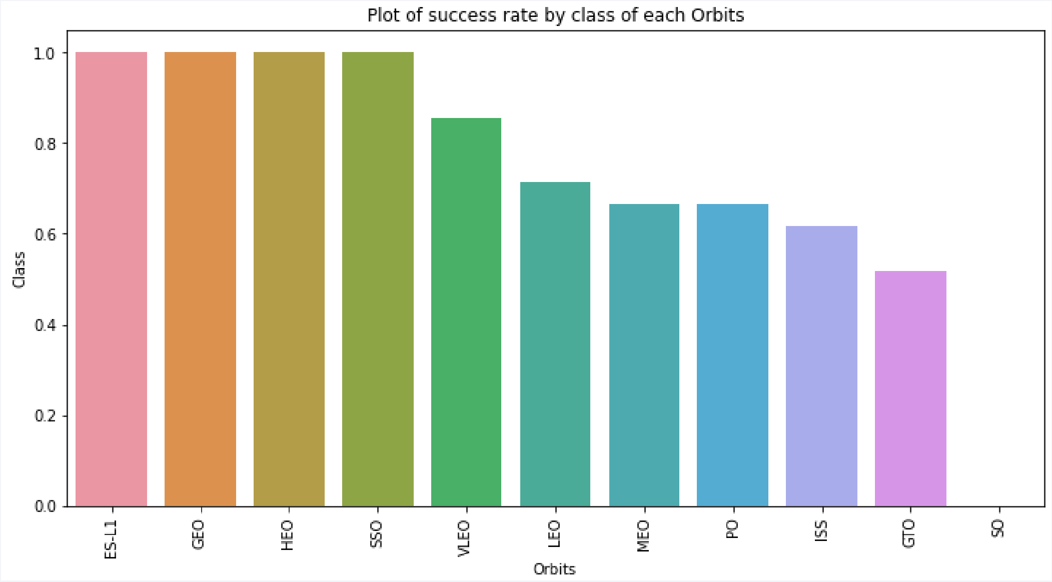


- We performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits
- The link to the notebook is [Notebook]

EDA with Data Visualization

- We 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.

- The link to the notebook is [Notebook]



EDA with SQL

- We loaded the SpaceX dataset into a SQL database without leaving the jupyter notebook.
- We applied EDA with SQL to get insight from the data. We wrote 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
  - The total number of successful and failure mission outcomes
  - The failed landing outcomes in drone ship, their booster version and launch site names.
- The link to the notebook is Notebook

Build an Interactive Map with Folium

- 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.
- The link to the notebook is [Notebook]
- 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 a Dashboard with Plotly Dash

- 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 python app is [App]

Predictive Analysis (Classification)

- We loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- We built different machine learning models and tune different hyperparameters using GridSearchCV.
- We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- We found the best performing classification model.
- The link to the notebook is Notebook

Results

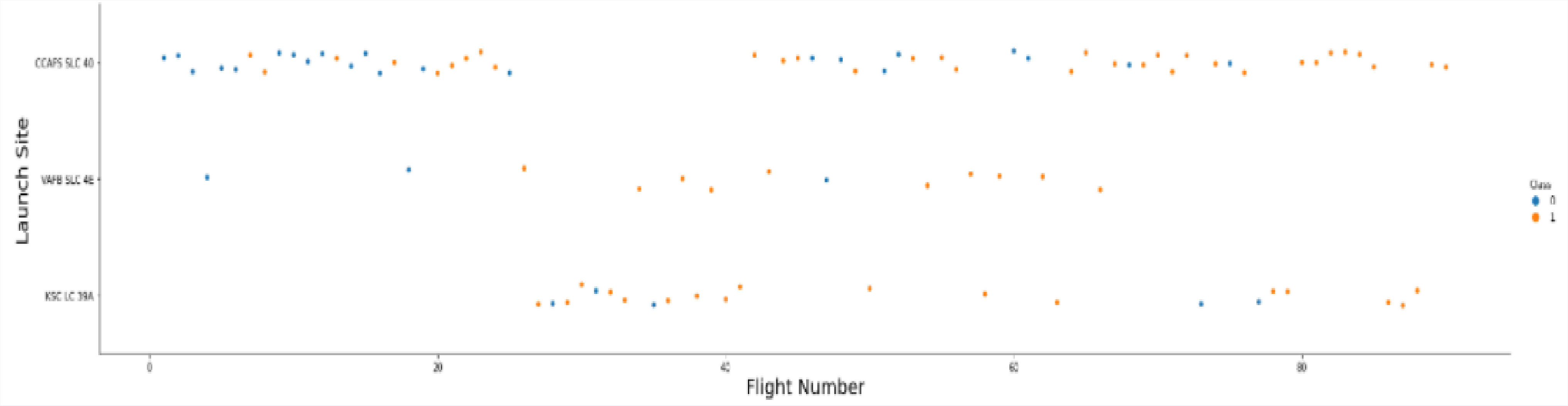
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

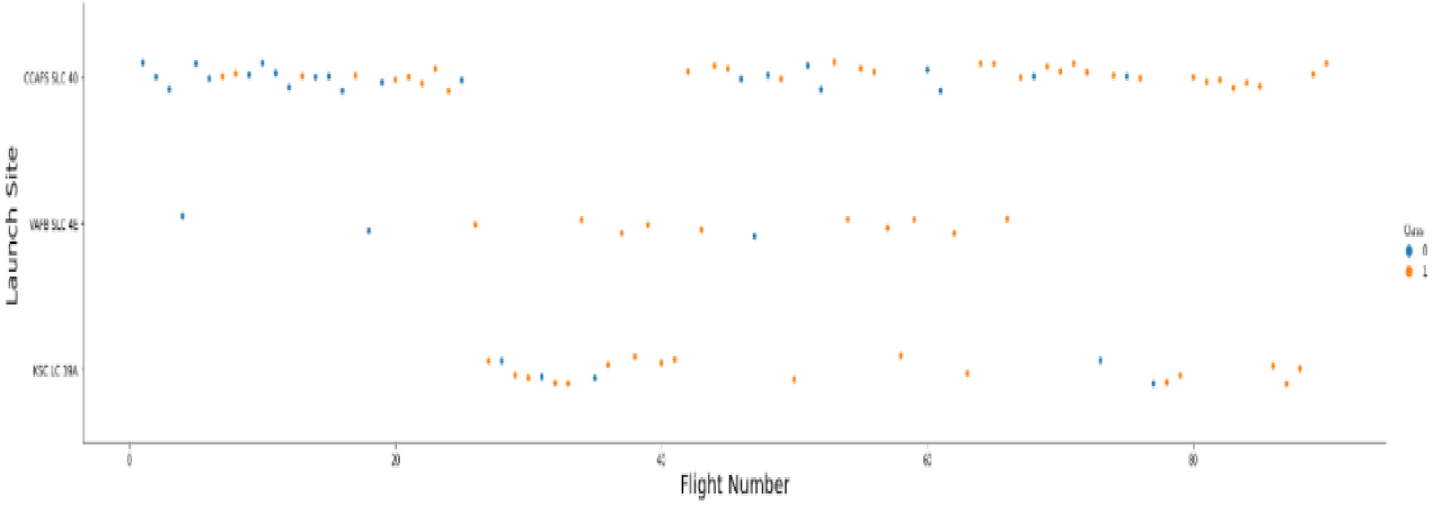
Flight Number vs. Launch Site

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



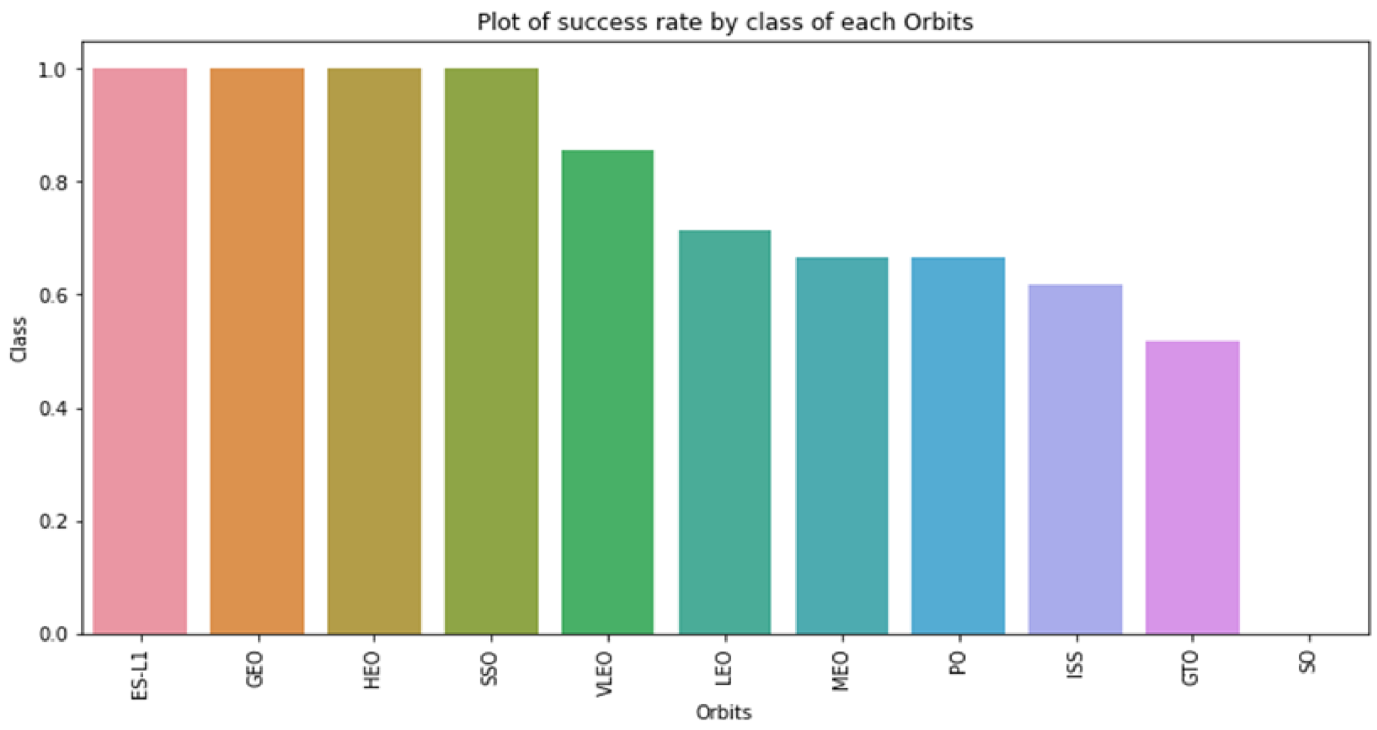
Payload vs. Launch Site

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



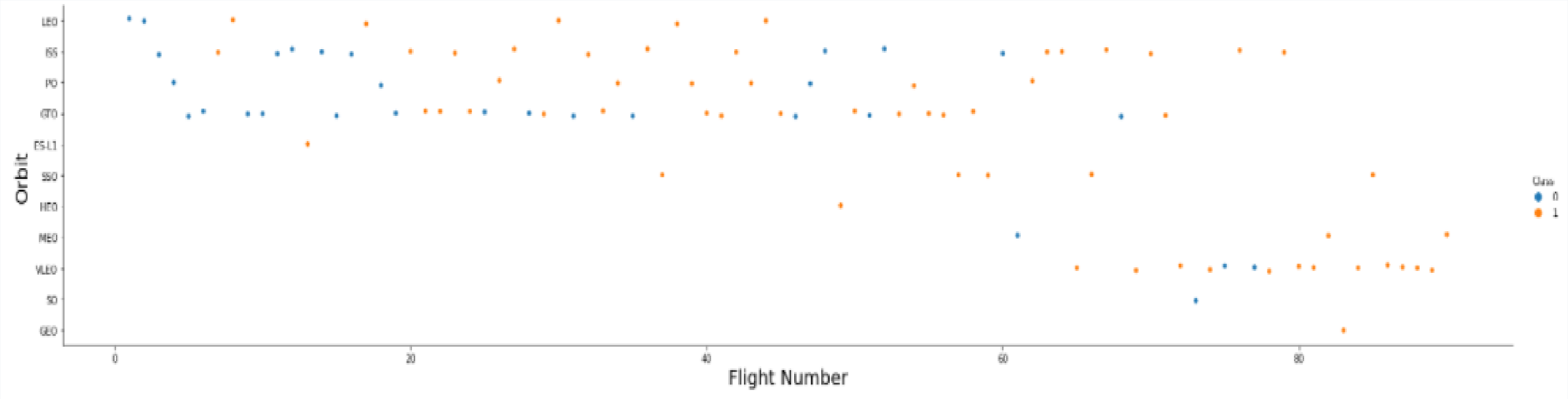
Success Rate vs. Orbit Type

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



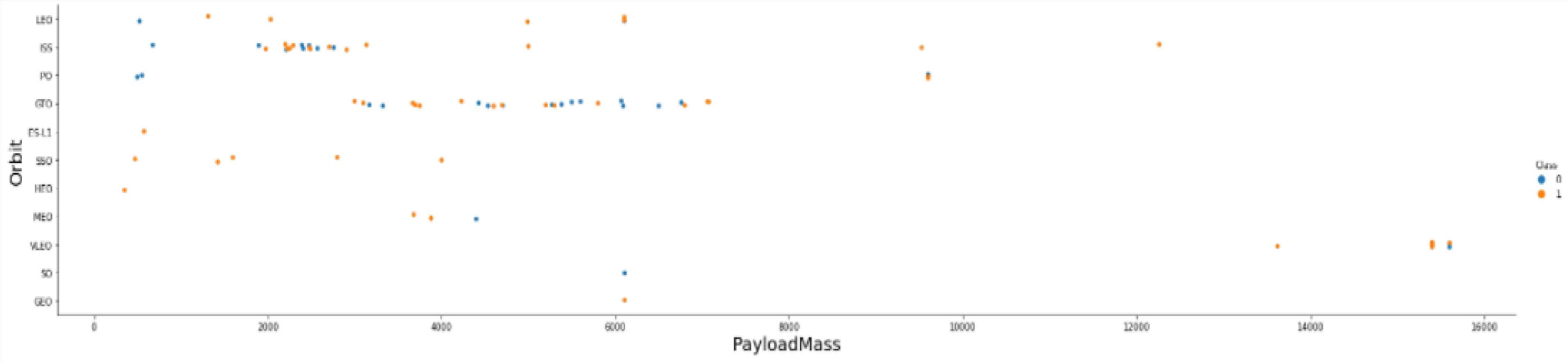
Flight Number vs. Orbit Type

- 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.



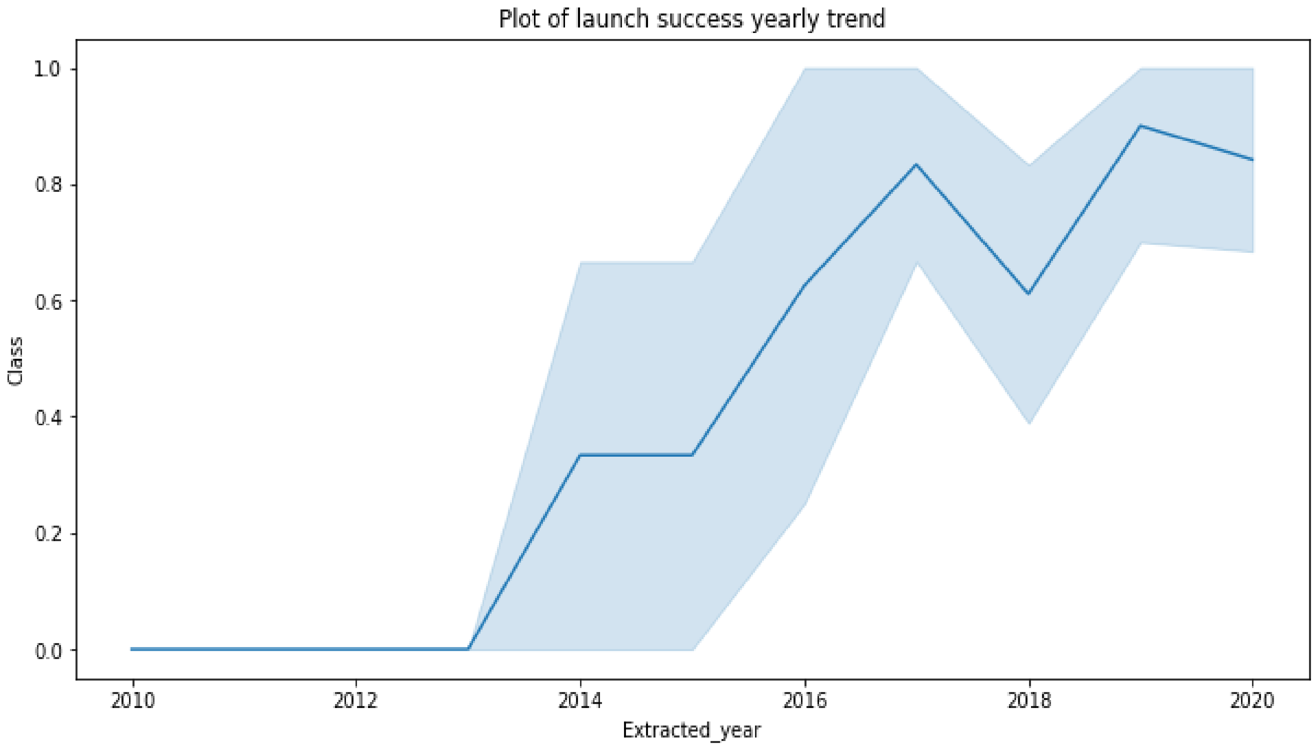
Payload vs. Orbit Type

- 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

```
%sql select DISTINCT("Launch_Site") from "SPACEXTBL"
```

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

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

Launch Site Names Begin with 'CCA'

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

```
%sql select "Launch_Site" from "SPACEXTBL" where "Launch_Site" like 'CCA%' limit 5
```

Python

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

Launch_Site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

- 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)

```
%sql select sum("PAYLOAD_MASS__KG_") AS 'Total Payload Mass carried by boosters launched by Nasa (CRS)'  from "SPACEXTBL" where "Customer" Like 'N
```

Python

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

Total Payload Mass carried by boosters launched by Nasa (CRS)
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

```
%sql select avg("PAYLOAD_MASS__KG_") as 'Average Payload mass carried by F9 v1.1' from SPACEXTBL where "Booster_Version" like 'F9 v1.1';
```

Python

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

Average Payload mass carried by F9 v1.1
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

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

Hint:Use min function

```
%sql select "Date" as 'First Successful landing outcome in ground pad date' from SPACEXTBL where "Landing_Outcome" like 'Success (ground pad)' lim
```

Python

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

First Successful landing outcome in ground pad date
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

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

```
%%sql select "Booster_Version" as 'boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000'
    from spacextbl
    where "Landing_Outcome" like 'Success (drone ship)' and ("PAYLOAD_MASS__KG_" < 6000 and "PAYLOAD_MASS__KG_" > 4000);
```

Python

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

boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

- We used group by

List the total number of successful and failure mission outcomes

```
%%sql select "Mission_Outcome", count("Mission_Outcome") as 'Total Number'
    from spacextbl group by "Mission_Outcome";
```

Python

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

Mission_Outcome	Total Number
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	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.

Task 8

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

```
%%sql select "Booster_Version" as 'Booster Versions with max payload mass' from spacextbl
        where "PAYLOAD_MASS__KG_" = (select max("PAYLOAD_MASS__KG_") from spacextbl)
```

Python

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

Booster Versions with max payload mass
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

2015 Launch Records

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

```
%%sql SELECT
    substr("Date", 0, 5) as 'Year'
    ,CASE substr("Date", 6, 2)
        WHEN '01' THEN 'January'
        WHEN '02' THEN 'February'
        WHEN '03' THEN 'March'
        WHEN '04' THEN 'April'
        WHEN '05' THEN 'May'
        WHEN '06' THEN 'June'
        WHEN '07' THEN 'July'
        WHEN '08' THEN 'August'
        WHEN '09' THEN 'September'
        WHEN '10' THEN 'October'
        WHEN '11' THEN 'November'
        WHEN '12' THEN 'December'
        ELSE 'Invalid Month'
    END AS "Month Names", "Landing_Outcome" as "Landing Outcomes in drone ship", "Booster_Version", "Launch_Site"
FROM spacextbl
Where "Landing_Outcome" like "Failure (drone ship)" and (substr("Date", 0, 5) like "2015");
```

Python

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

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

- We selected Landing outcomes first 15

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.

```
%sql select * from SPACEXTBL limit 15;
```

Python

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

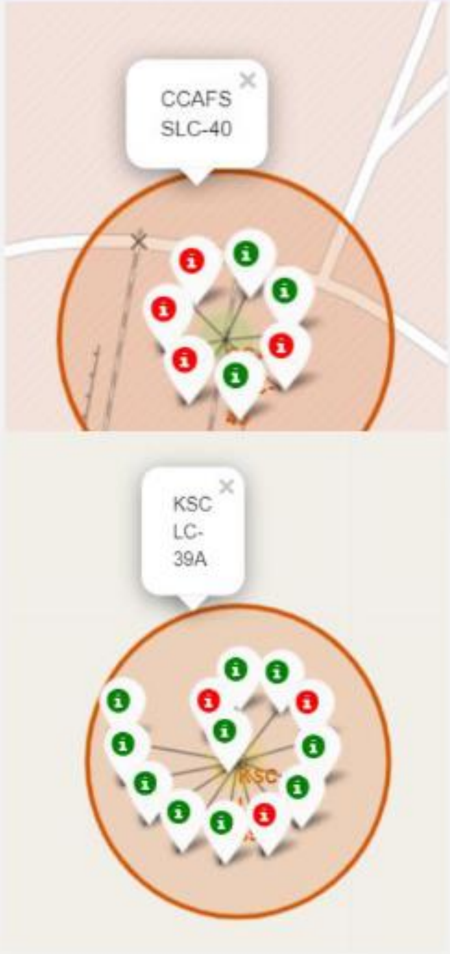
Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight 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
2013-09-29	16:00:00	F9 v1.1 B1003	VAFB SLC-4E	CASSIOPE	500	Polar LEO	MDA	Success	Uncontrolled (ocean)

Section 4

Launch Sites Proximities Analysis

All launch sites global map markers

Markers showing launch sites with color labels



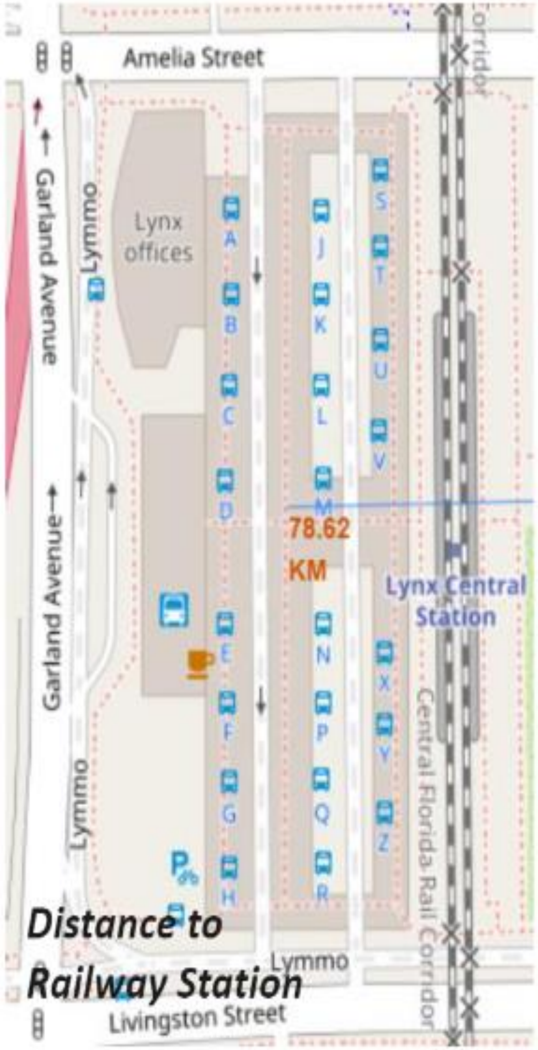
Florida Launch Sites

Green Marker shows successful Launches and Red Marker shows Failures

California Launch Site

37

Launch Site distance to landmarks

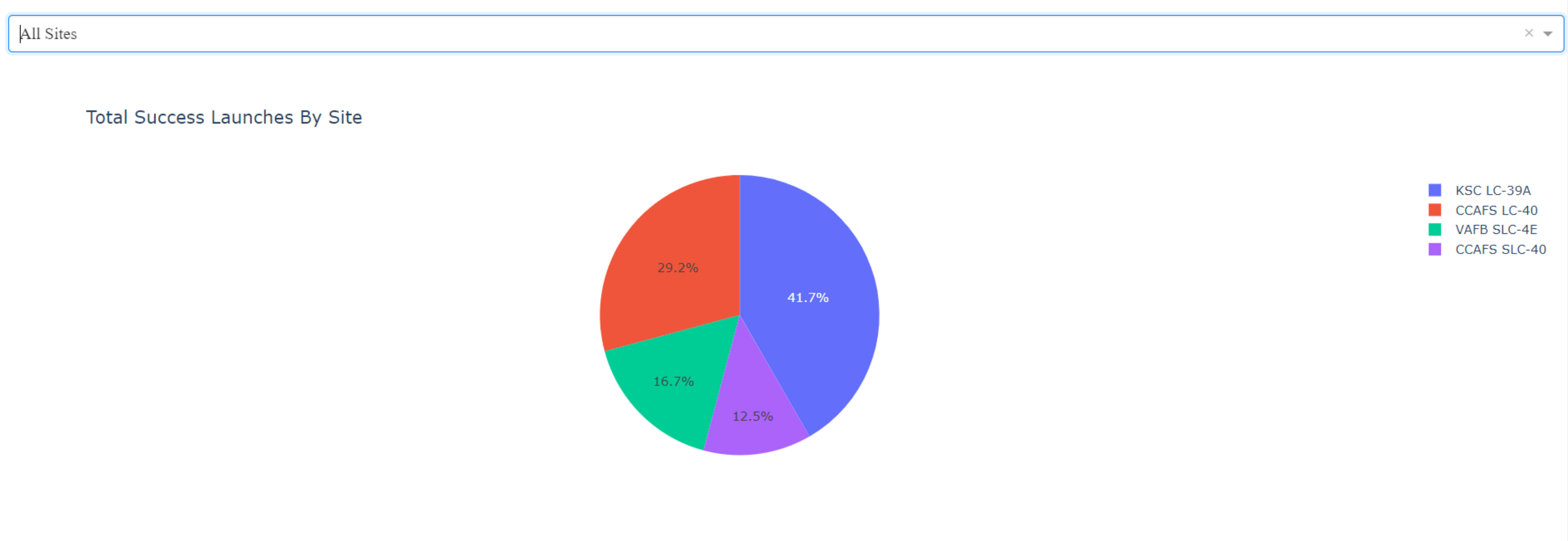


- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

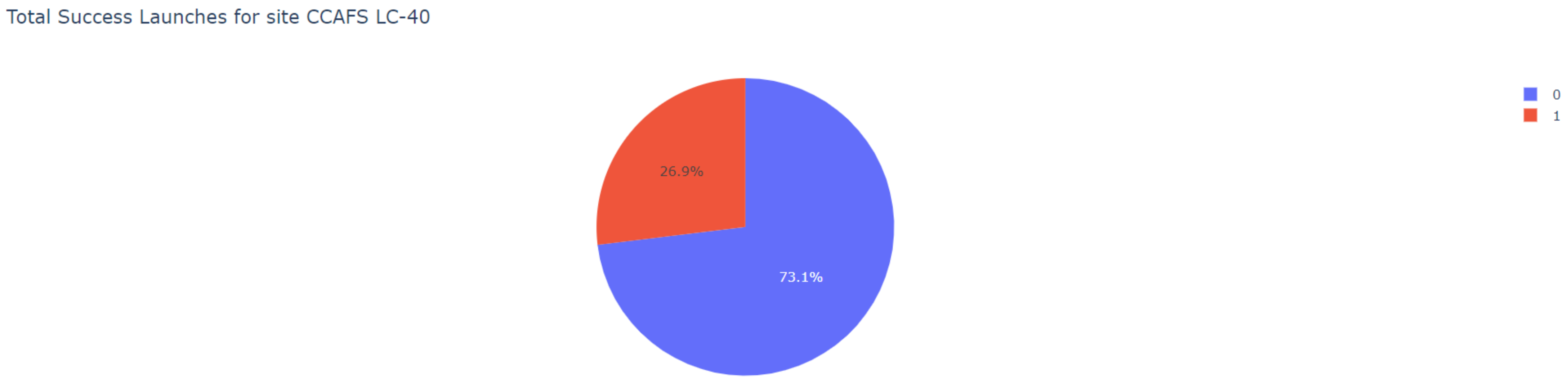
Section 5

Build a Dashboard with Plotly Dash

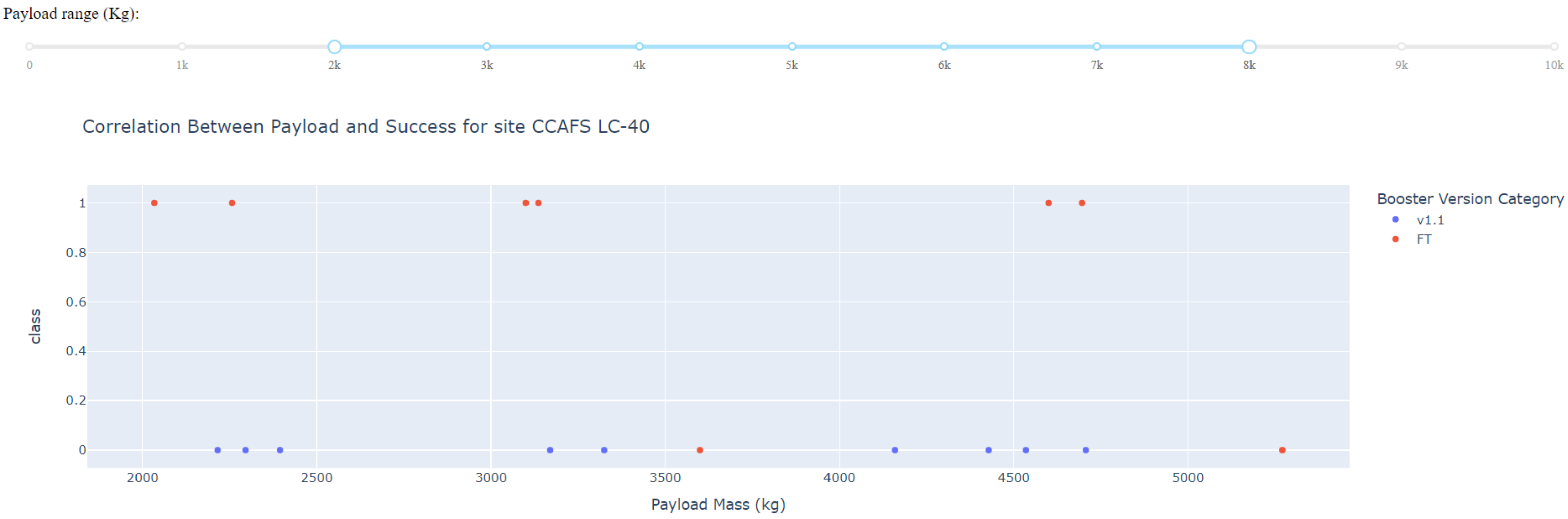
Pie chart showing the success percentage achieved by each launch site



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



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



Section 6

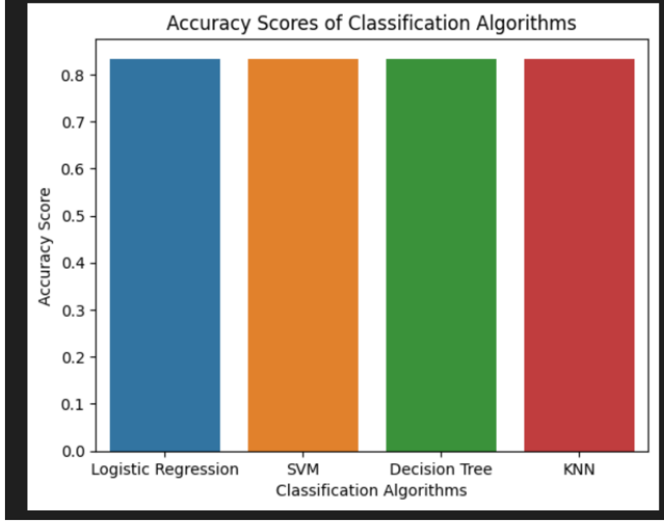
Predictive Analysis (Classification)

Classification Accuracy

- They got the same accuracy

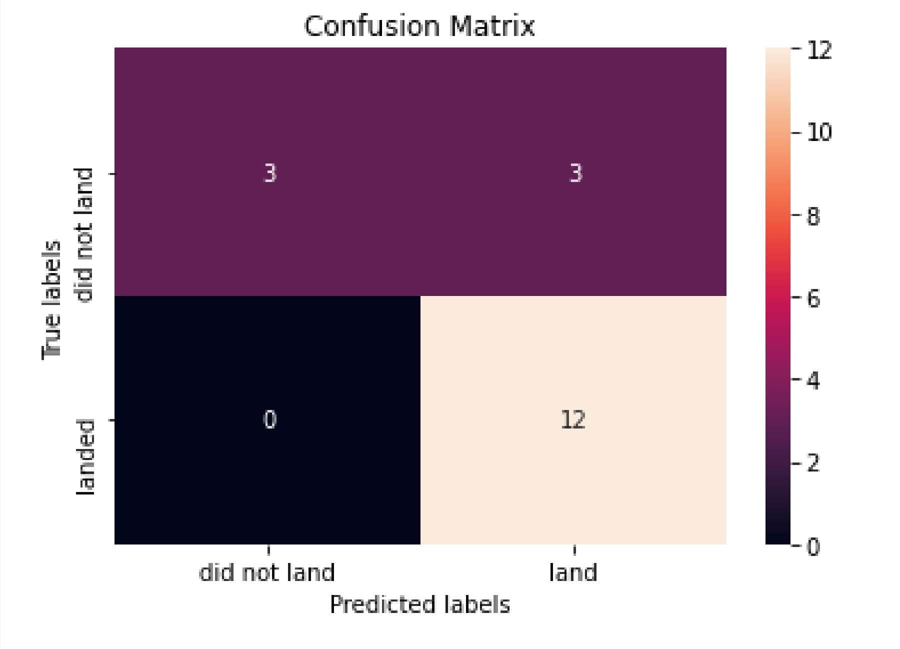
```python
Accuracy = {'Logistic Regression': [logreg_cv.score(X_test, Y_test)],
        'SVM': [svm_cv.score(X_test, Y_test)],
        'Decision Tree': [tree_cv.score(X_test, Y_test)],
        'KNN': [knn_cv.score(X_test, Y_test)]}

accuracy_df = pd.DataFrame(Accuracy, index =['Accuracy'])
accuracy_df

sns.barplot(data=accuracy_df)
plt.title("Accuracy Scores of Classification Algorithms")
plt.xlabel("Classification Algorithms")
plt.ylabel("Accuracy Score")
plt.show()
```

Python



Confusion Matrix

- The confusion matrix for the SVM 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 4 models are considered good for this task.

Appendix

- GitHub Repository

Thank you!