IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Emanuel Fitta
16/03/2022

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

  - 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

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

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 https://github.com/jerf8010/DS_Projects/blob/main/Project%20space%20x/data_collection_api.ipynb

1. Get request for rocket launch data using API

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

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

2. Use json_normalize method to convert json result to dataframe

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

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

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

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

In [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
```

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 https://github.com/jerf8010/DS_Projects/blob/main/Project%20space%20x/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

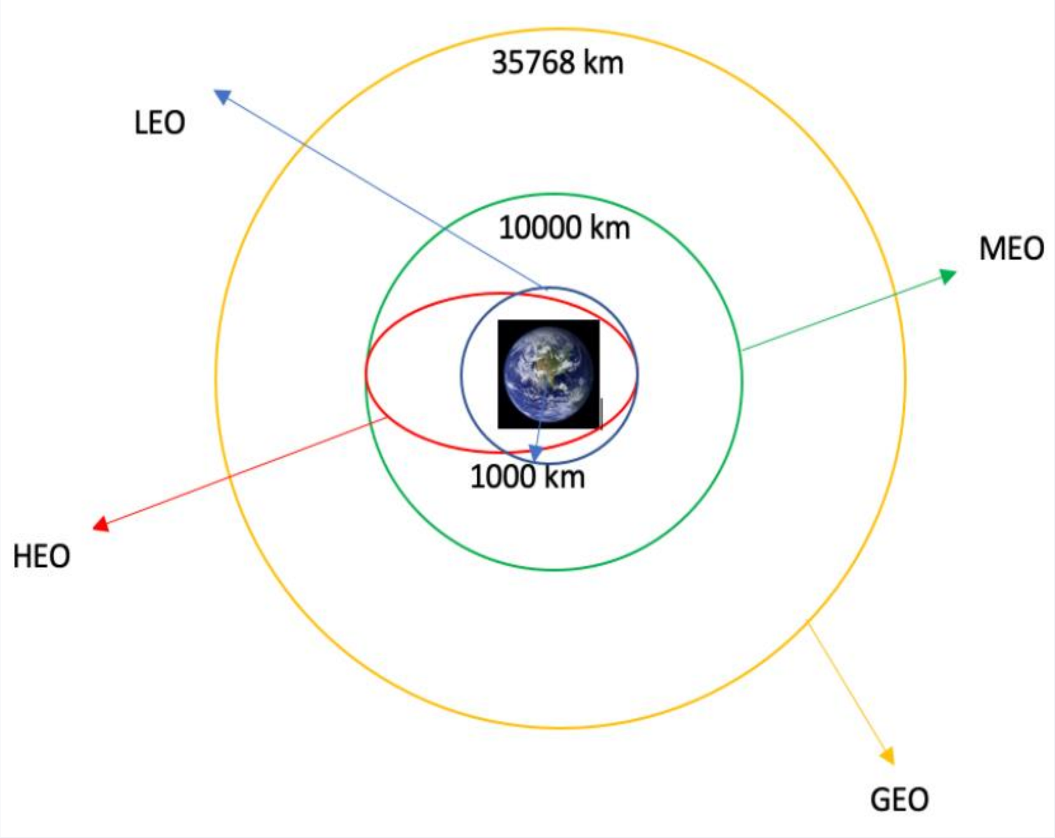
In [10]:
```
column_names = []

# 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

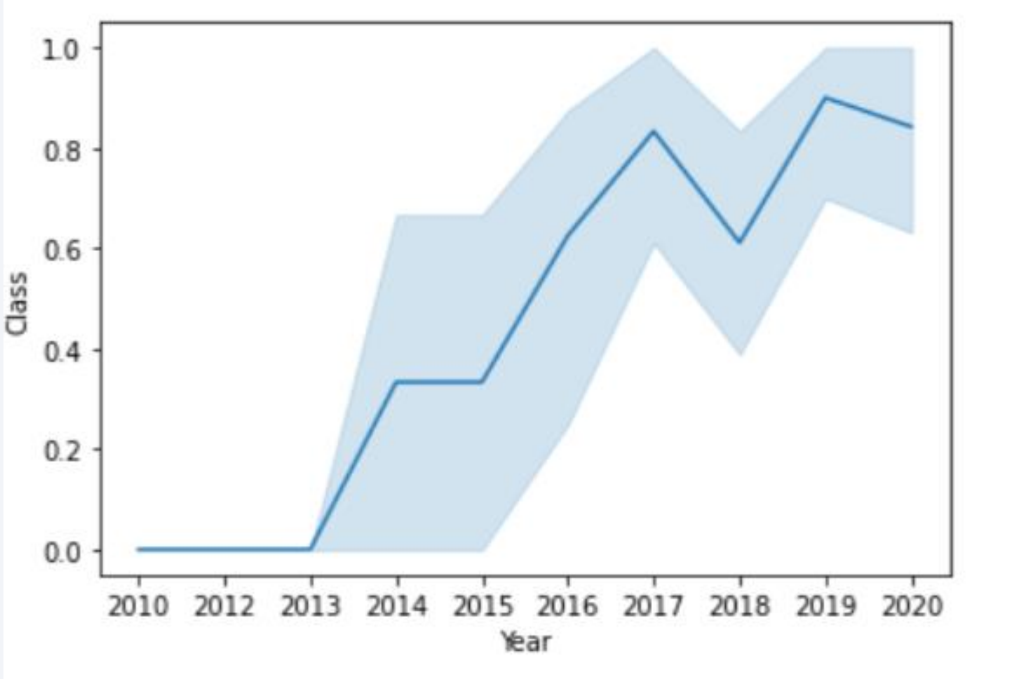
Data Wrangling

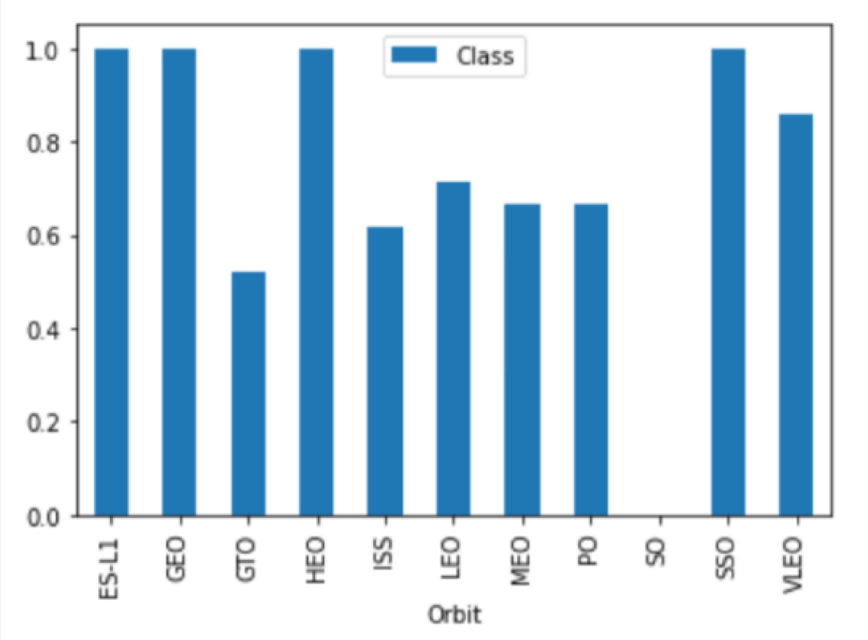


- We performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits
- We created landing outcome label from outcome column and exported the results to csv.
- The link to the notebook is https://github.com/jerf8010/DS_Projects/blob/main/Project%20space%20x/Data_wrangling.ipynb

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 https://github.com/jerf8010/DS_Projects/blob/main/Project%20space%20x/eda_dataviz.ipynb

EDA with SQL

- We loaded the SpaceX dataset into a PostgreSQL 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 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.
- The link to the notebook is https://github.com/jerf8010/DS_Projects/blob/main/Project%20space%20x/eda-sql.ipynb

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.
- 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.

https://github.com/jerf8010/DS_Projects/blob/main/Project%20space%20x/launch_site_location.ipynb

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 notebook is https://github.com/jerf8010/IBM-Data-Science-Capstone-SpaceX/blob/main/app.py

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 https://github.com/jerf8010/DS_Projects/blob/main/Project%20space%20x/SpaceX_Machine_Learning_Prediction_Part_5.ipynb

Results

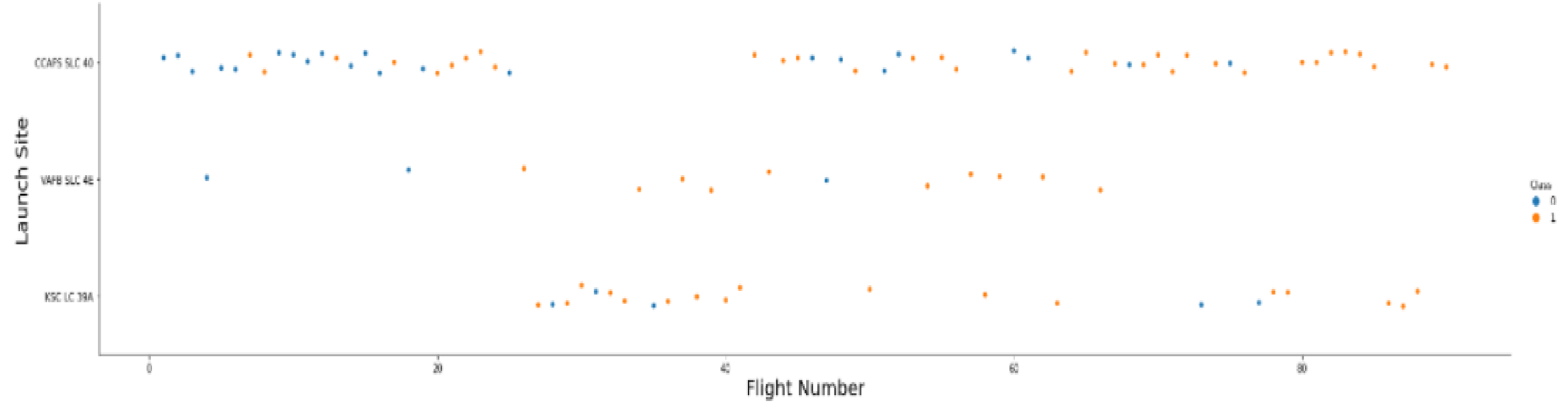
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

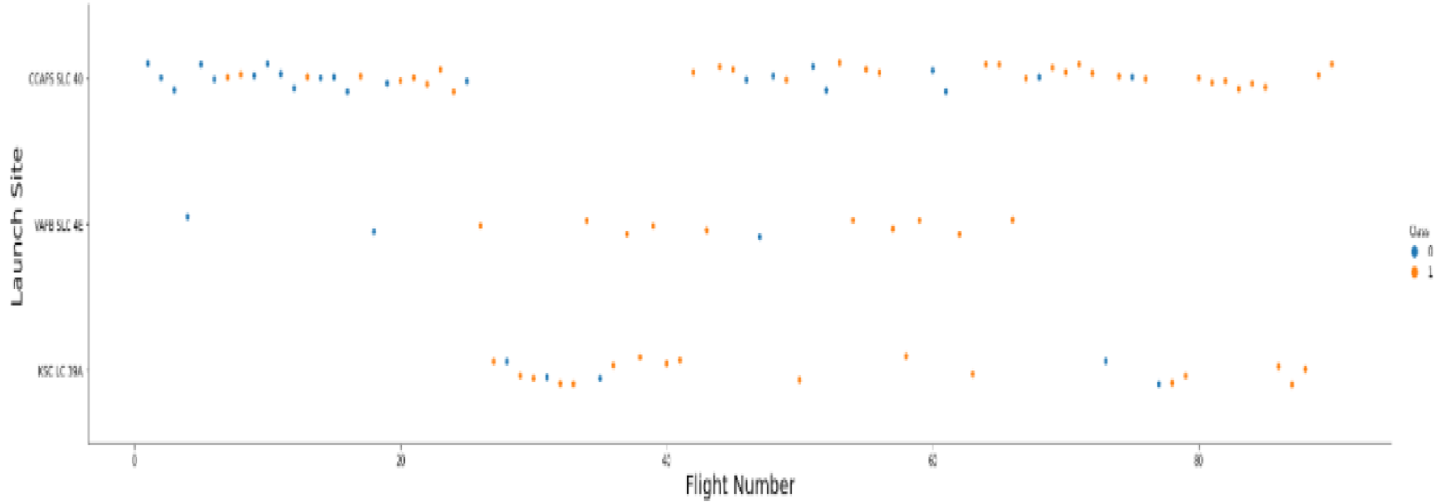
Insights drawn from EDA

Flight Number vs. Launch Site

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

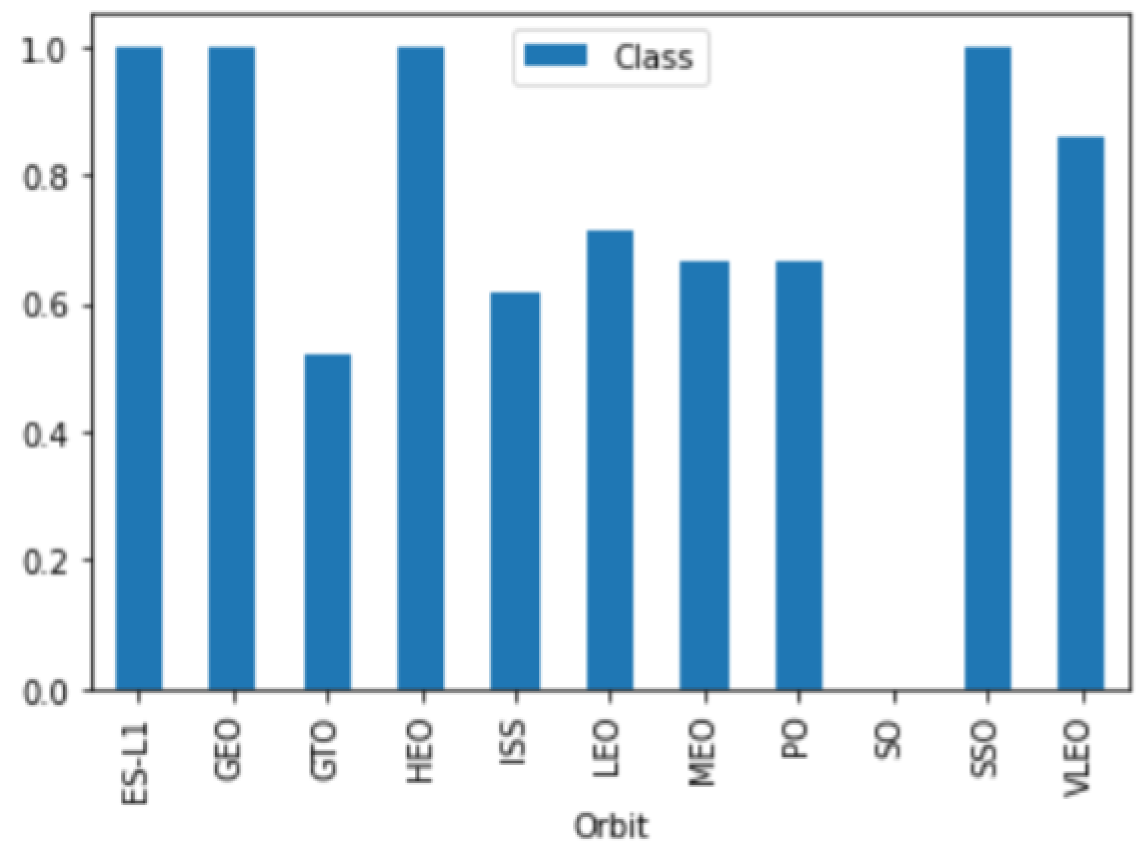


Payload vs. Launch Site



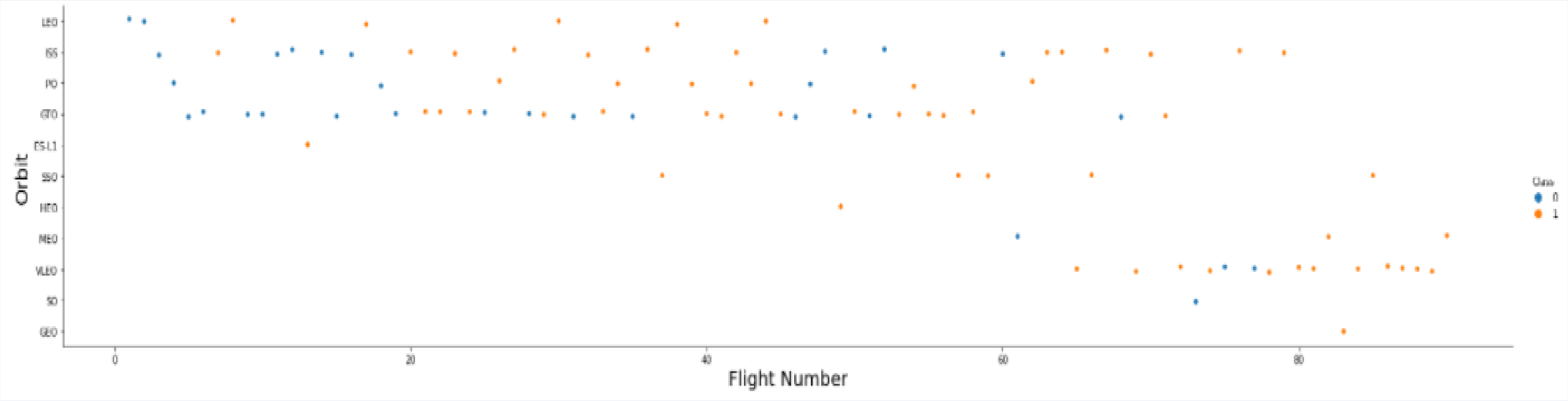
Success Rate vs. Orbit Type

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



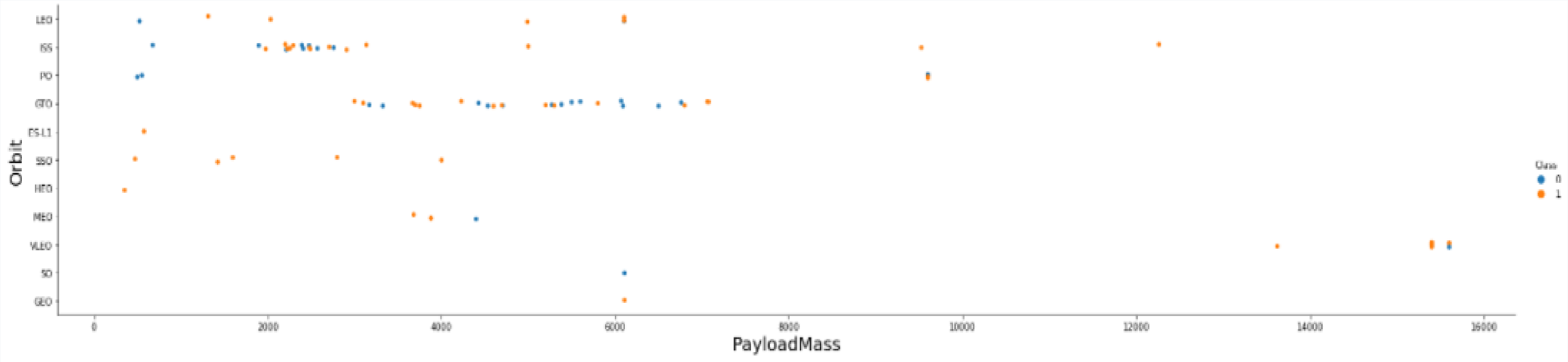
Flight Number vs. Orbit Type

- 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.



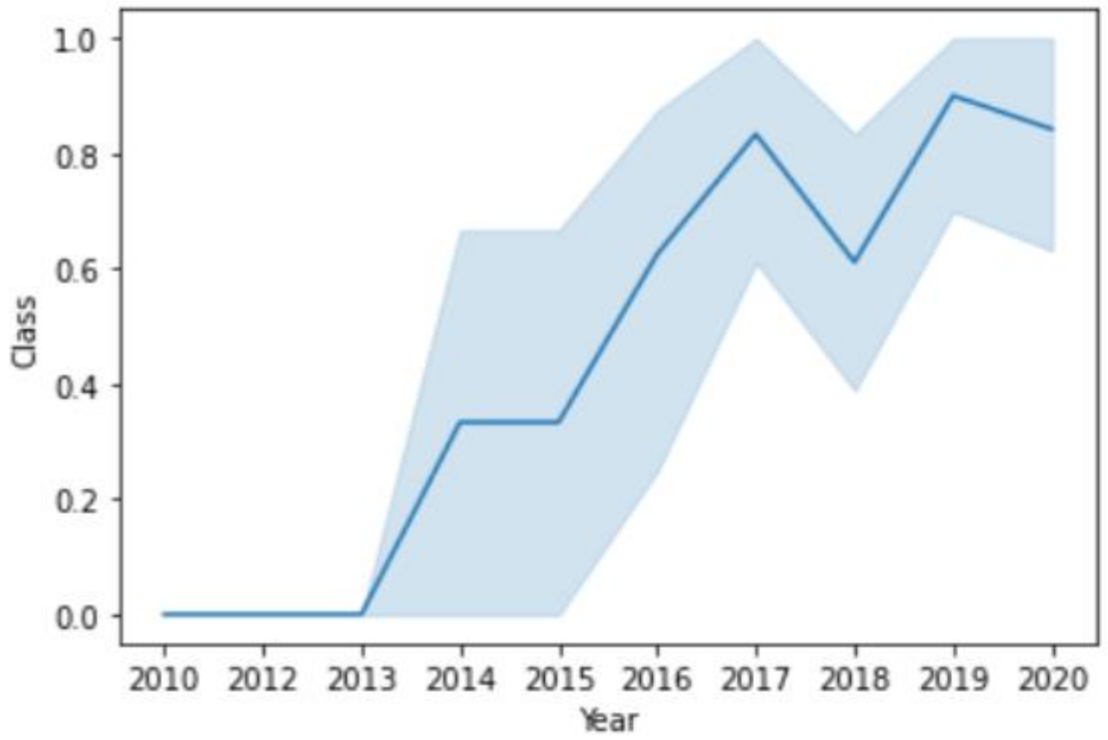
Payload vs. Orbit Type

- 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.

```
spark.sql('SELECT DISTINCT Launch_Site FROM spacex').toPandas()
```

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

Launch Site Names Begin with 'CCA'

```
spark.sql('SELECT * FROM spacex WHERE Launch_Site LIKE "CCA%"').limit(5).toPandas()
```

	Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
0	04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	08-12-2010	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	22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
3	08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	01-03-2013	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

```
spark.sql('SELECT SUM(PAYLOAD_MASS__KG_)  FROM spacex WHERE Customer = "NASA (CRS)"').toPandas()
```

	sum(PAYLOAD_MASS__KG_)
0	45596

Average Payload Mass by F9 v1.1

- We calculated the average payload mass carried by booster version F9 v1.1 as 2534.66

```
]: spark.sql('SELECT AVG(PAYLOAD_MASS__KG_)  FROM spacex WHERE Booster_Version LIKE "F9 v1.1%"').toPandas()
```

]:

	avg(PAYLOAD_MASS__KG_)
0	2534.666667

First Successful Ground Landing Date

- We observed that the dates of the first successful landing outcome on ground pad was 01-05-2017

```
spark.sql('SELECT MIN(Date) FROM spacex WHERE Landing_Outcome = "Success (ground pad)"').toPandas()
```

	min(Date)
0	01-05-2017

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

```
spark.sql('SELECT Payload FROM spacex WHERE Landing_Outcome = "Success (drone ship)" AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000').toPandas()
```

	Payload
0	JCSAT-14
1	JCSAT-16
2	SES-10
3	SES-11 / EchoStar 105

Total Number of Successful and Failure Mission Outcomes

Task 7

List the total number of successful and failure mission outcomes

16]:
```
spark.sql('SELECT Mission_Outcome, COUNT(*) FROM spacex GROUP BY Mission_Outcome').toPandas()
```

16]:

	Mission_Outcome	count(1)
0	Success	98
1	Success (payload status unclear)	1
2	Failure (in flight)	1
3	Success	1

- We group by Mission Outcome and count every register.

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.

```
spark.sql('SELECT Booster_Version, PAYLOAD_MASS__KG_ FROM spacex WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM spacex) ').toPandas()
```

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

```
[19]: spark.sql('SELECT Booster_Version, Launch_Site, date, Landing_Outcome FROM spacex WHERE Landing_Outcome = "Success (drone ship)" AND Date LIKE "%2017"'
```

[19]:

	Booster_Version	Launch_Site	date	Landing_Outcome
0	F9 FT B1029.1	VAFB SLC-4E	14-01-2017	Success (drone ship)
1	F9 FT B1021.2	KSC LC-39A	30-03-2017	Success (drone ship)
2	F9 FT B1029.2	KSC LC-39A	23-06-2017	Success (drone ship)
3	F9 FT B1036.1	VAFB SLC-4E	25-06-2017	Success (drone ship)
4	F9 FT B1038.1	VAFB SLC-4E	24-08-2017	Success (drone ship)
5	F9 B4 B1041.1	VAFB SLC-4E	09-10-2017	Success (drone ship)
6	F9 FT B1031.2	KSC LC-39A	11-10-2017	Success (drone ship)
7	F9 B4 B1042.1	KSC LC-39A	30-10-2017	Success (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.

```
spark.sql('SELECT Landing_Outcome, COUNT(*) FROM spacex WHERE Date BETWEEN "04-06-2010" AND "20-03-2017" GROUP BY Landing_Outcome').toPandas()
```

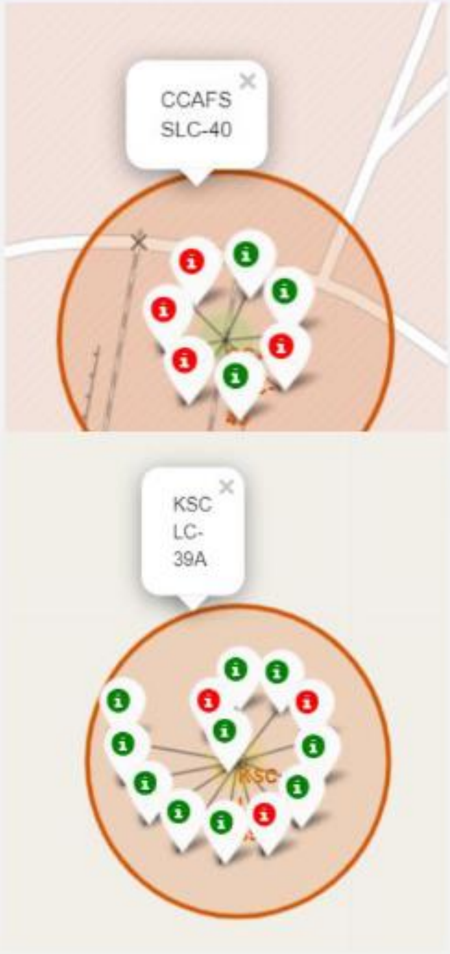
	Landing_Outcome	count(1)
0	Failure (drone ship)	4
1	Success	20
2	Failure	3
3	No attempt	1
4	Success (ground pad)	6
5	No attempt	10
6	Failure (parachute)	2
7	Success (drone ship)	8
8	Controlled (ocean)	3

Section 4

Launch Sites Proximities Analysis

All launch sites global map markers

Markers showing launch sites with color labels



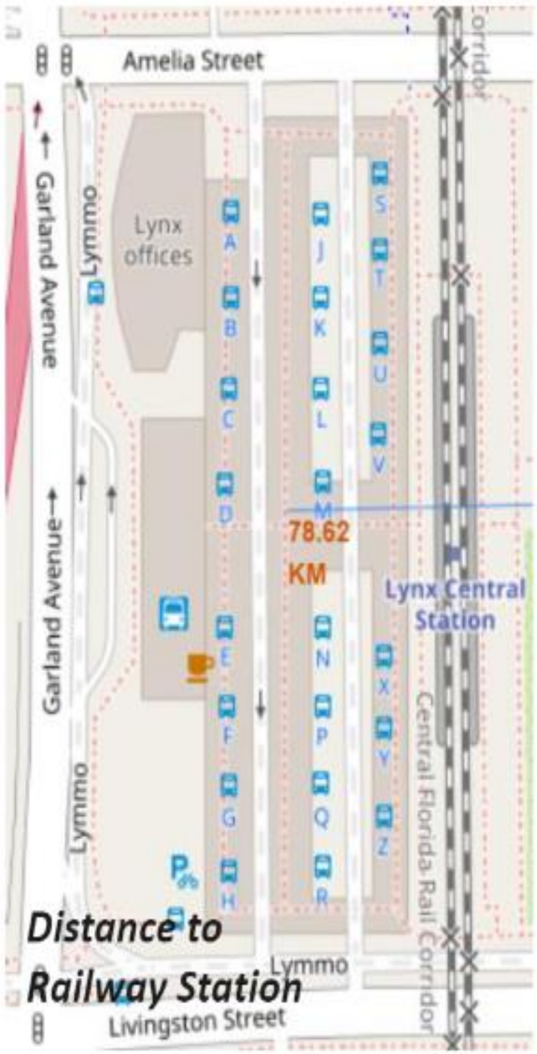
Florida Launch Sites

Green Marker shows successful Launches and Red Marker shows Failures

California Launch Site

37

Launch Site distance to landmarks

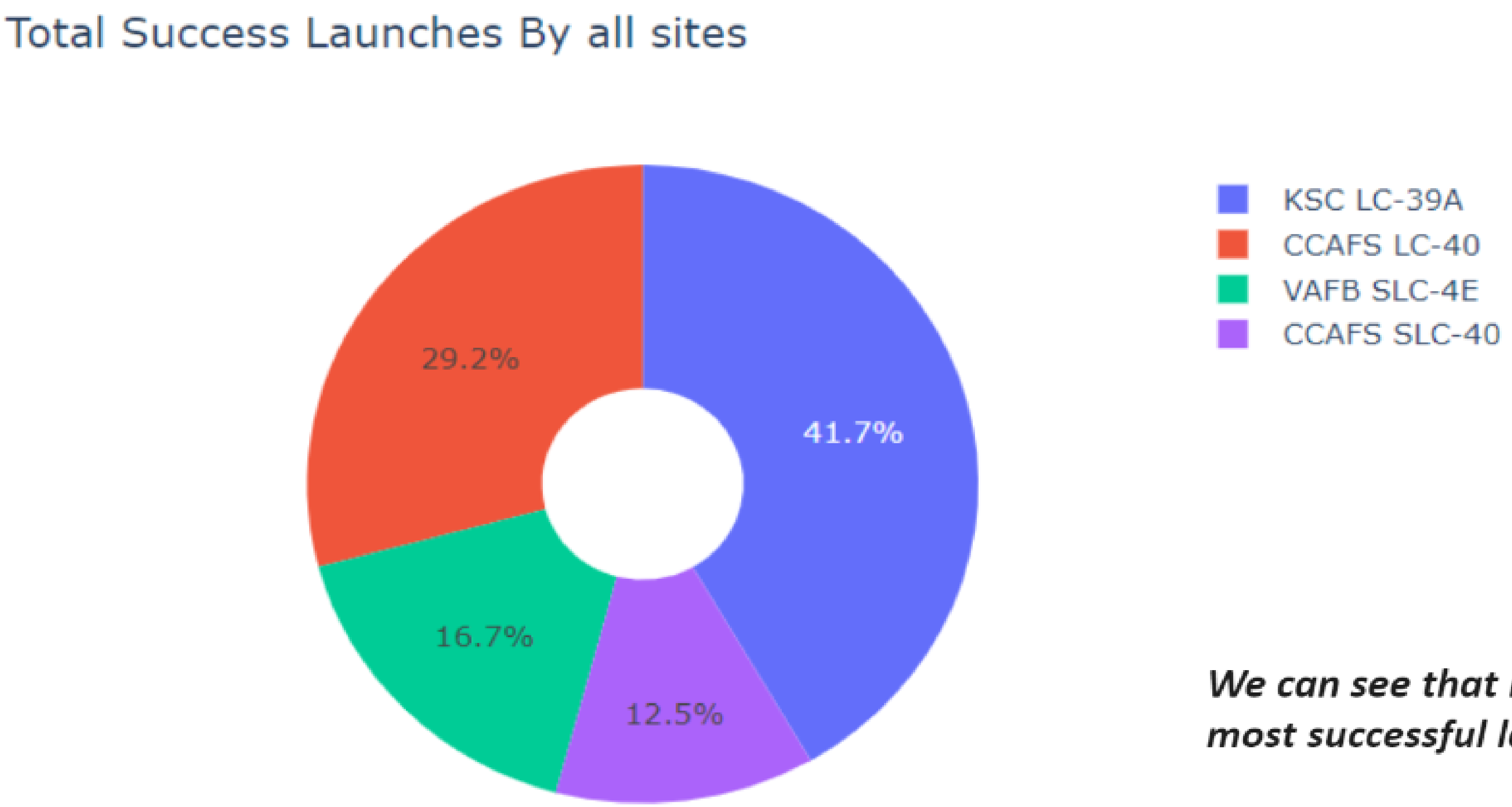


- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

Section 5

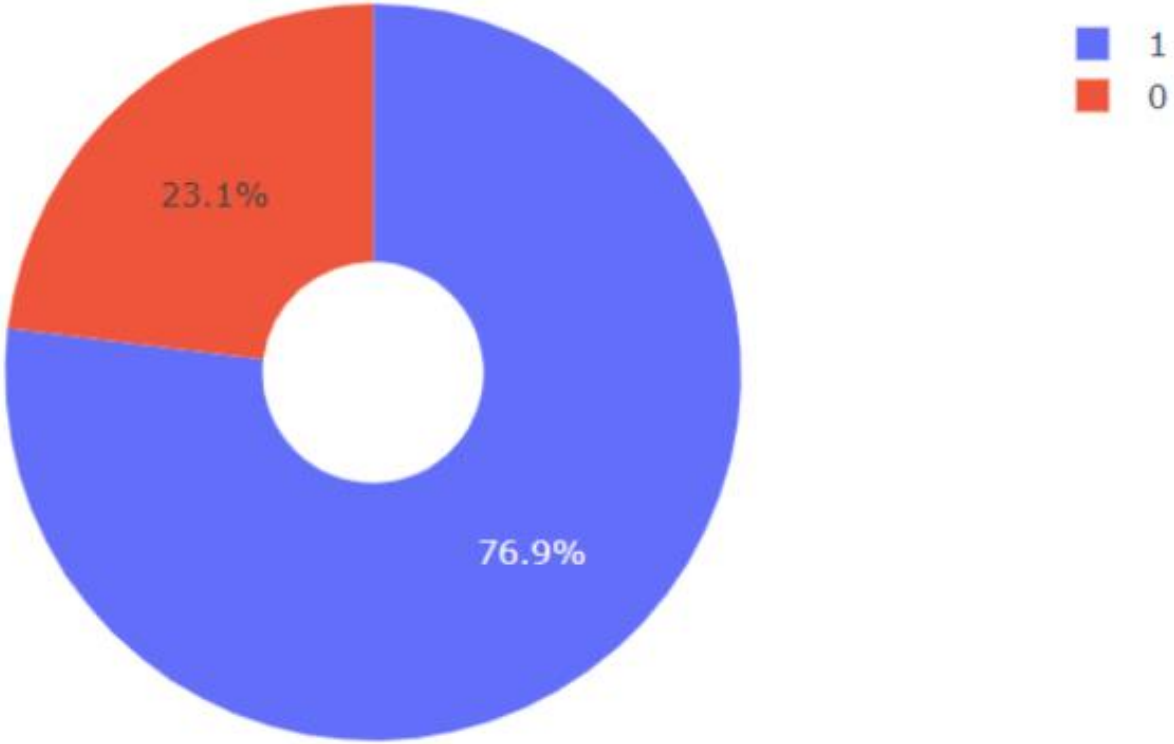
Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site



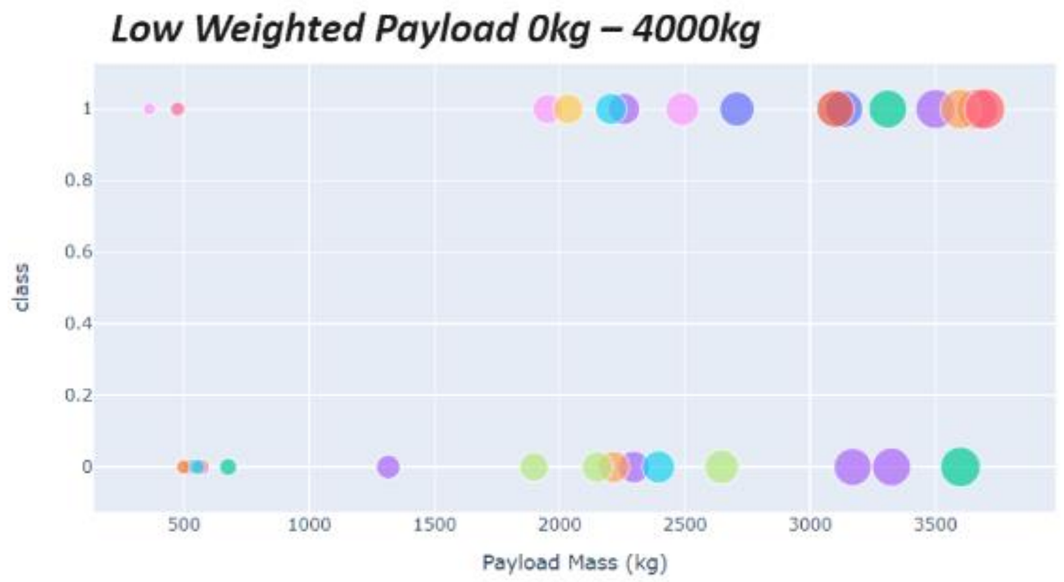
We can see that KSC LC-39A had the most successful launches from all the sites

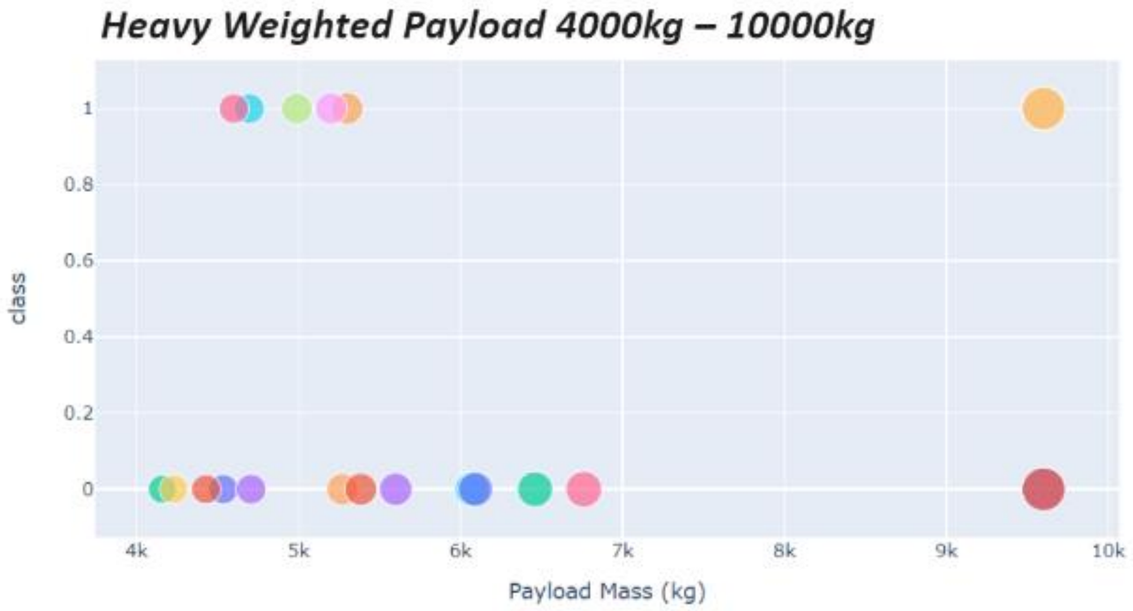
Pie chart showing the Launch site with the highest launch success ratio



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

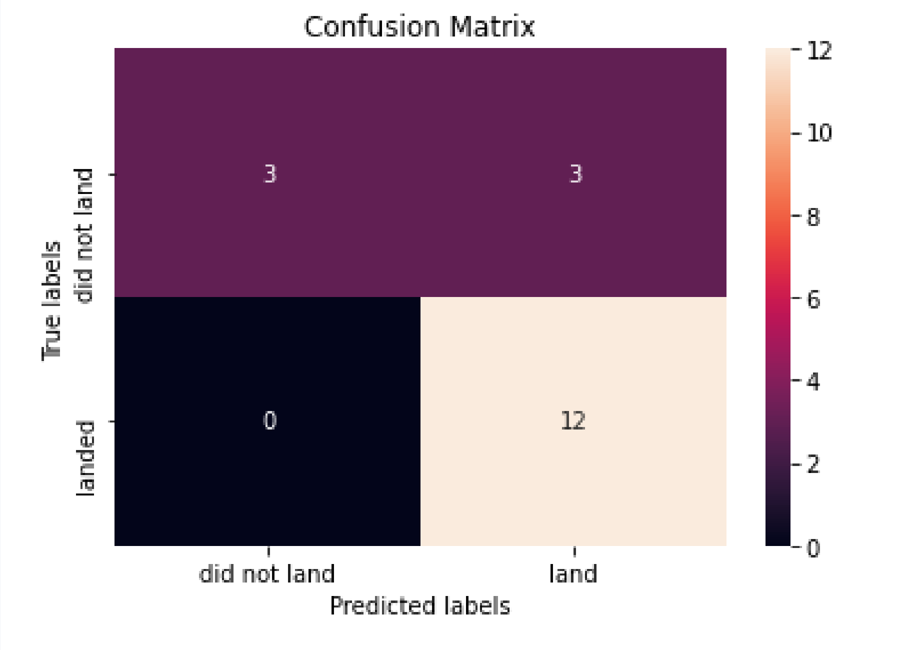
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

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!