Winning Space Race with Data Science

Kamil Chłosta

18nd October 2022

Outline

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

Executive Summary

- Summary of methodologies
  - Data collection (web scraping)
  - Data wrangling
  - EDA with data visualization (seaborn python module)
  - EDA with SQL (using sqlite)
  - Building an interactive map with Folium
  - Building a Dashboard with Plotly and Dash
  - Predictive analysis (classification problem)
- Summary of all results
  - EDA results
  - Interactive analytics
  - Predictive analysis

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
  - The project task is to predict whether the first stage of the SpaceX Falcon 9 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:
  - SpaceX Rest API
  - Web Scrapping from Wikipedia
- Perform data wrangling
  - Data cleaning, imputation and lastly One Hot Encoding have been performed
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Logistic Regression, KNN, SVM and Decision Trees models have been implemented and evaluated and compared to find the best classifier

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 and turn it into a pandas dataframe using .json_normalize().
  - 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

- 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 here.

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

- Falcon 9 launch records have been web-scrapped using BeautifulSoup Python module
- The table has been parsed and converted into pandas dataframe.
- The link to the notebook is here.

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

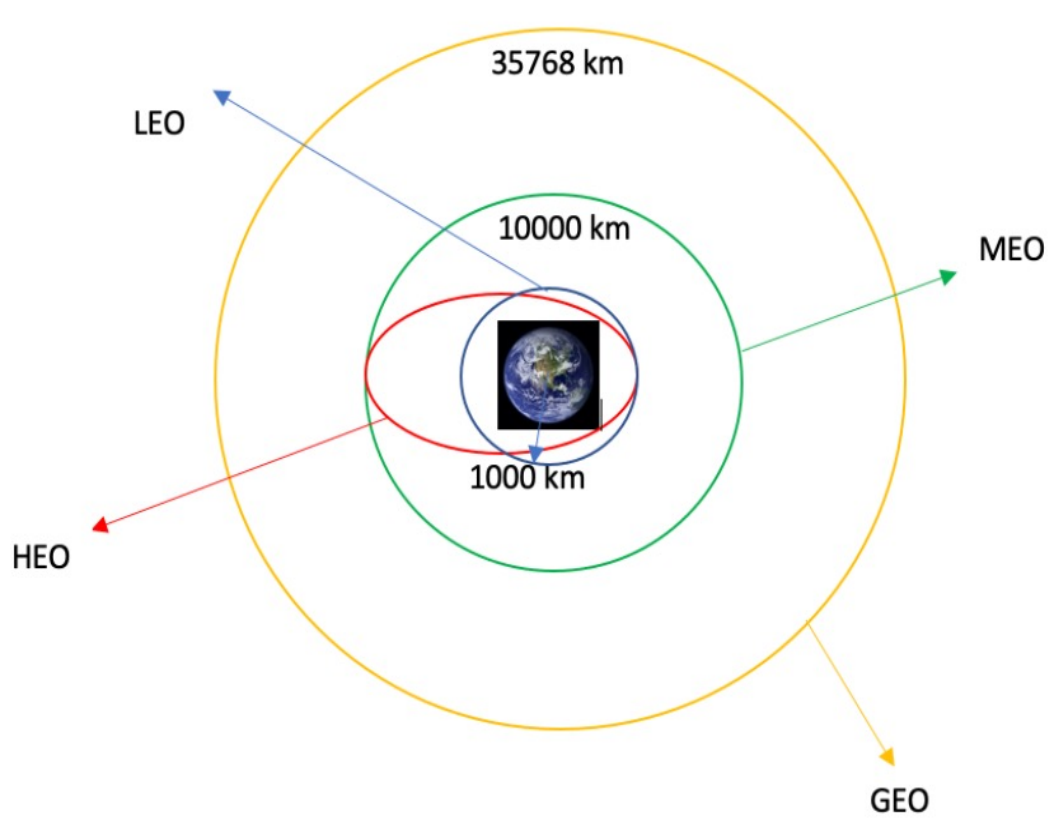
```
In [10]: column_names = []

         # 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

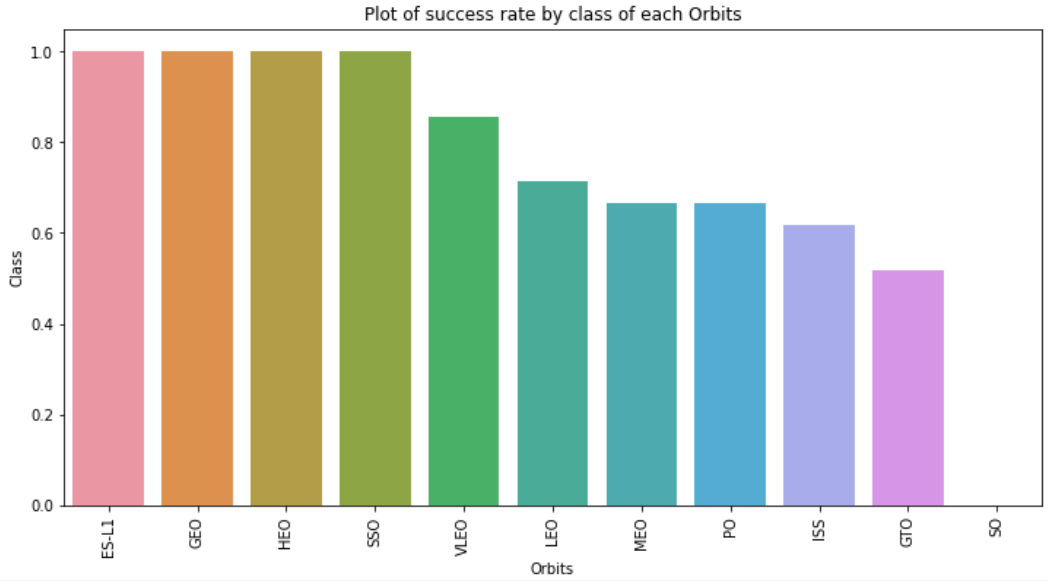
Data Wrangling

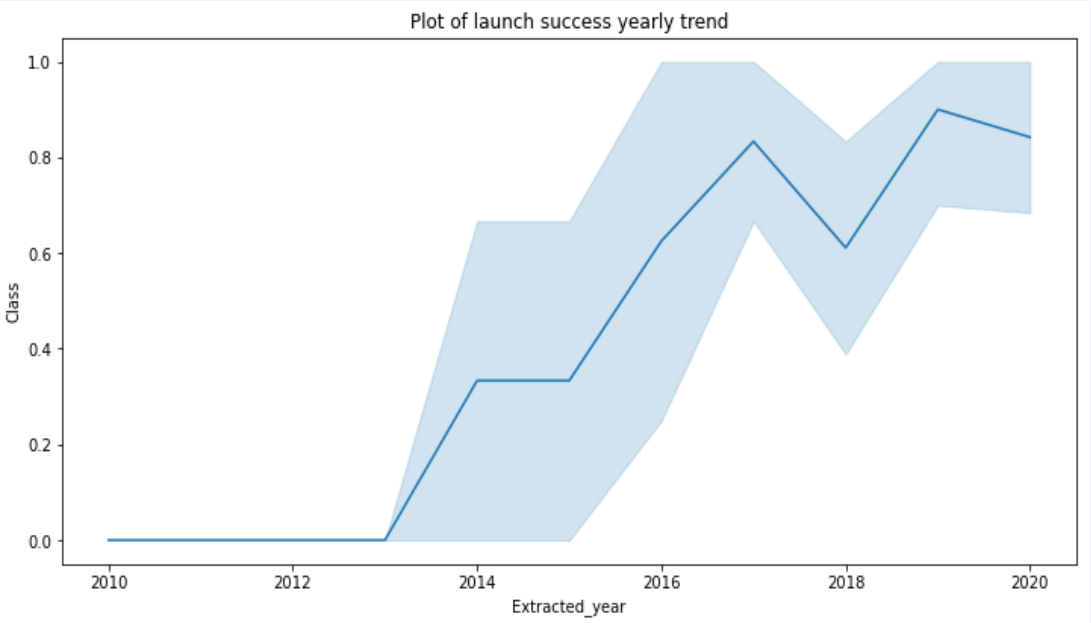


- Exploratory data analysis has been performed and the training labels have been determined
- The number of launches at each site and the number and occurrence of each orbits have been calculated
- Landing outcome label from outcome column have been determined and the results have been exported to csv.
- The link to the notebook is here.

EDA with Data Visualization

- The data has been explored 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/chuksoo/IBM-Data-Science-Capstone-SpaceX/blob/main/EDA%20with%20Data%20Visualization.ipynb

EDA with SQL

- The SpaceX dataset has been loaded into a PostgreSQL database inside of the jupyter notebook.
- EDA with SQL has been applied to get insight from the data. SQL queries have been executed to find out for example:
  - 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 here.

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
- answered some question for instance:
  - if launch sites are near railways, highways and coastlines.
  - Do launch sites keep certain distance away from cities?

Build a Dashboard with Plotly Dash

- an interactive dashboard with Plotly dash has been built
- pie charts showing the total launches by a certain sites have been plotted
- scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version have been plotted
- the link to the app code is here.

Predictive Analysis (Classification)

- loaded the data using numpy and pandas, transformed the data, split data into training and testing.
- built different machine learning models and tuned 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.
- The link to the notebook is here.

Results

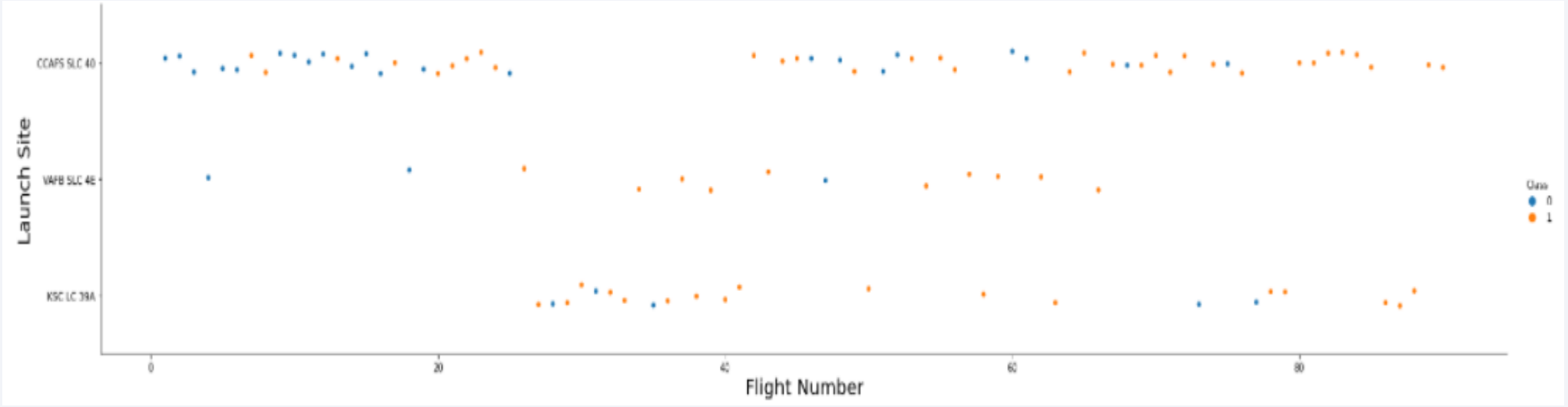
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

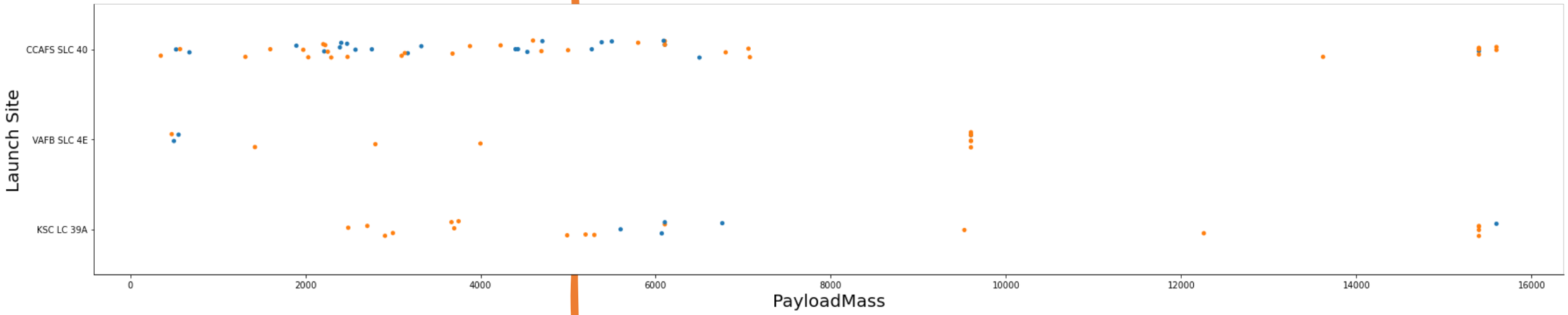
Flight Number vs. Launch Site

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



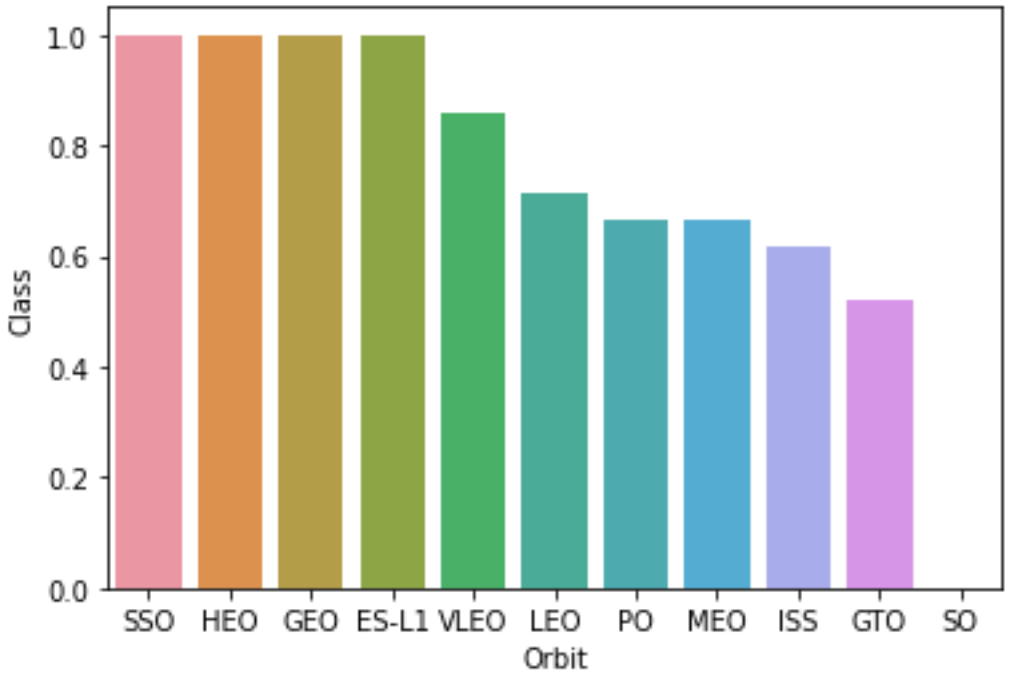
Payload vs. Launch Site

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



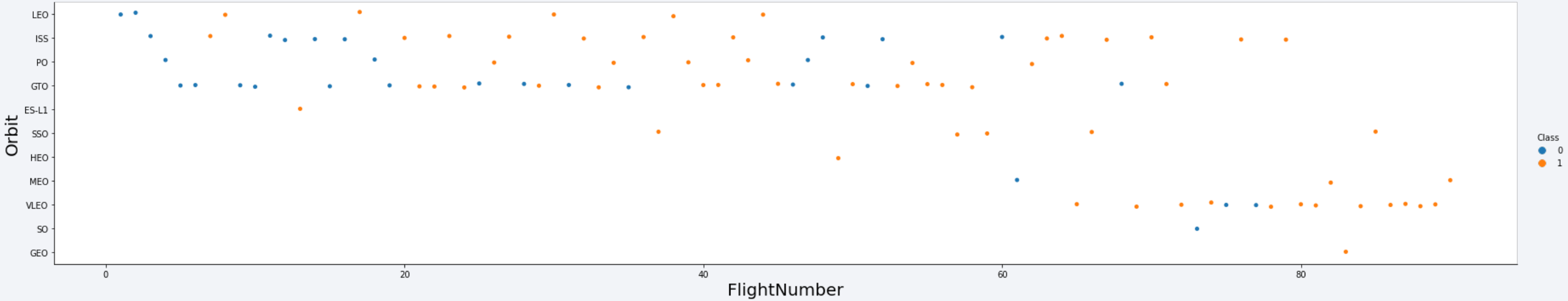
Success Rate vs. Orbit Type

- ES-L1, GEO, HEO, SSO, VLEO orbits have the best success rate.
- The SO orbit has the worst success rate (only failures)



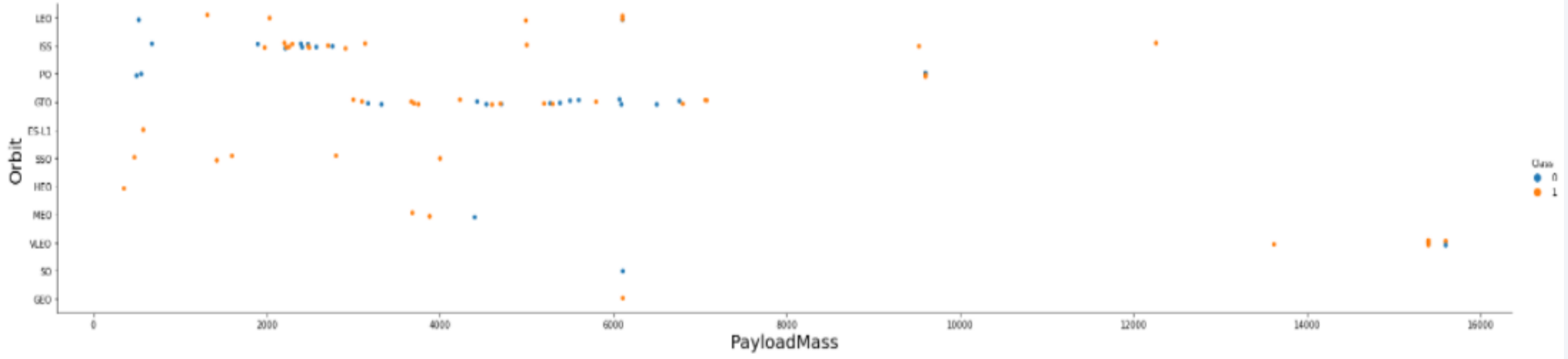
Flight Number vs. Orbit Type

- in the LEO orbit, success is related to the number of flights
- in the GTO orbit, there is no relationship between flight number and the success rate



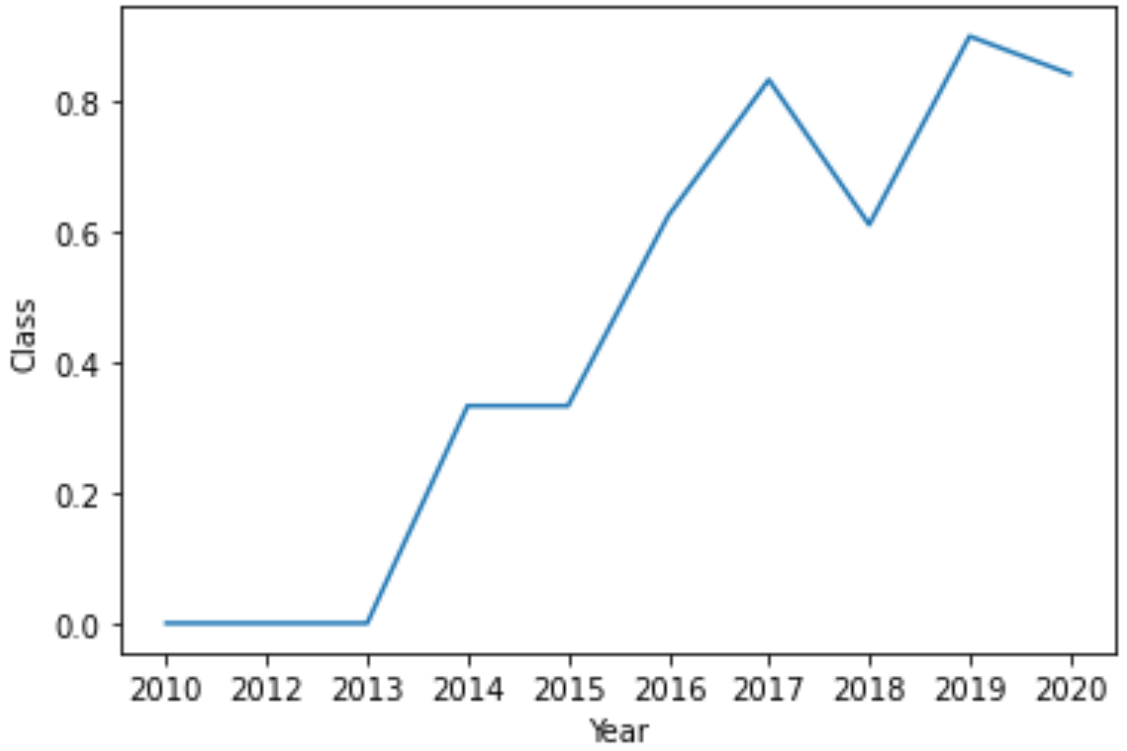
Payload vs. Orbit Type

- with heavy payloads, the successful landings are more likely to happen for PO, LEO and ISS orbits.



Launch Success Yearly Trend

- success rate kept on increasing since 2013 till 2020 apart from slight setback in 2018



All Launch Site Names

- Used SQL **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'

10 records where launch sites begin with `CCA`

```
%sql select * from SPACEXTBL where Launch_Site like 'CCA%' limit 10
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	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)
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
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt
03-12-2013	22:41:00	F9 v1.1	CCAFS LC-40	SES-8	3170	GTO	SES	Success	No attempt
06-01-2014	22:06:00	F9 v1.1	CCAFS LC-40	Thaicom 6	3325	GTO	Thaicom	Success	No attempt
18-04-2014	19:25:00	F9 v1.1	CCAFS LC-40	SpaceX CRS-3	2296	LEO (ISS)	NASA (CRS)	Success	Controlled (ocean)
14-07-2014	15:15:00	F9 v1.1	CCAFS LC-40	OG2 Mission 1 6 Orbcomm-OG2 satellites	1316	LEO	Orbcomm	Success	Controlled (ocean)
05-08-2014	08:00:00	F9 v1.1	CCAFS LC-40	AsiaSat 8	4535	GTO	AsiaSat	Success	No attempt

Total Payload Mass

- calculated the total payload carried by boosters from NASA as 99980 kg

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

```
%sql select SUM(PAYLOAD_MASS__KG_) from SPACEXTBL where Customer like 'NASA%'
```

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

SUM(PAYLOAD_MASS__KG_)
99980

Average Payload Mass by F9 v1.1

- the average payload mass carried by booster version F9 v1.1: 2928.4 kg

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

- the date of the first successful landing outcome on ground pad was 01st May 2017

```
%sql select min(Date), "Landing _Outcome" from SPACEXTBL where "Landing _Outcome" = 'Success (ground pad)'
```

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

min(Date)	Landing _Outcome
01-05-2017	Success (ground pad)

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

```
%sql select * from SPACEXTBL where (PAYLOAD_MASS__KG_ between 4000 and 6000) and "Landing _Outcome" = 'Success (drone ship)'
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
06-05-2016	05:21:00	F9 FT B1022	CCAFS LC-40	JCSAT-14	4696	GTO	SKY Perfect JSAT Group	Success	Success (drone ship)
14-08-2016	05:26:00	F9 FT B1026	CCAFS LC-40	JCSAT-16	4600	GTO	SKY Perfect JSAT Group	Success	Success (drone ship)
30-03-2017	22:27:00	F9 FT B1021.2	KSC LC-39A	SES-10	5300	GTO	SES	Success	Success (drone ship)
11-10-2017	22:53:00	F9 FT B1031.2	KSC LC-39A	SES-11 / EchoStar 105	5200	GTO	SES EchoStar	Success	Success (drone ship)

Total Number of Successful and Failure Mission Outcomes

- used **Group By** the Mission Outcome to see number of successes and failures.

```
%sql select count(Mission_Outcome), Mission_Outcome from SPACEXTBL group by Mission_Outcome
```

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

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

Boosters Carried Maximum Payload

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

```
%sql select Booster_Version, PAYLOAD_MASS__KG_ from SPACEXTBL where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTBL)
```

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

Booster_Version	PAYLOAD_MASS__KG_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

2015 Launch Records

- Listed the records which will display the month names, failure landing outcomes in drone ship, booster versions, launch site for the months in year 2015

```
%sql select substr(Date, 4, 2) as month, "Landing _Outcome", Booster_Version, Launch_Site from SPACEXTBL where substr(Date, 7, 4) = '2015
```

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

month	Landing _Outcome	Booster_Version	Launch_Site
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

- **Ranked the count of successful landing outcomes between the date 04-06-2010 and 20-03-2017 in descending order.**

```
%sql select "Landing _Outcome", count(*) as landing_outcome_count from SPACEXTBL where "Landing _Outcome" like 'Success%' group by "Landi
```

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

Landing _Outcome	landing_outcome_count
Success	38
Success (drone ship)	14
Success (ground pad)	9

Section 4

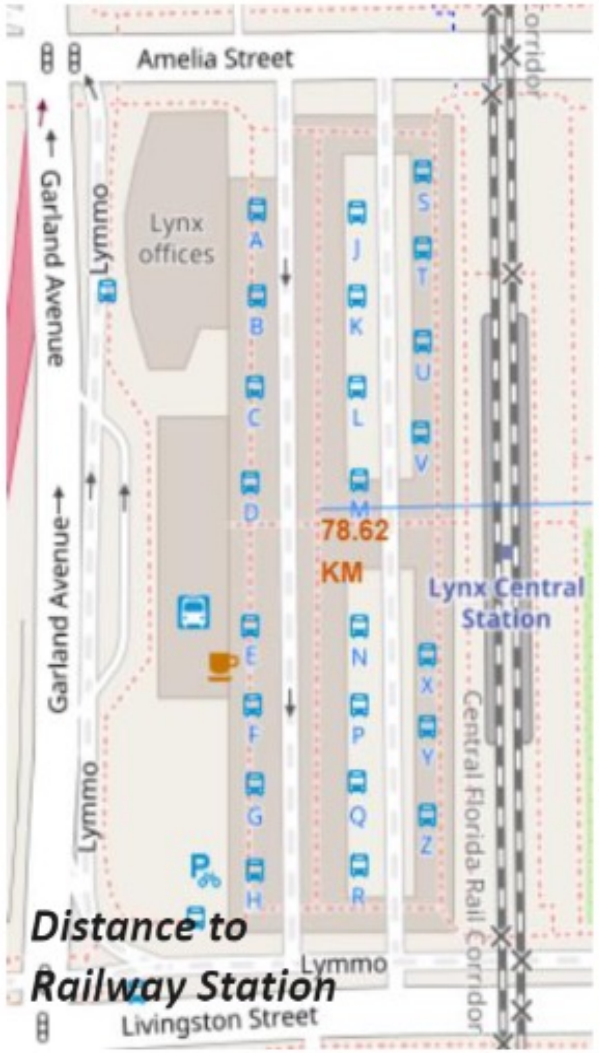
Launch Sites Proximities Analysis

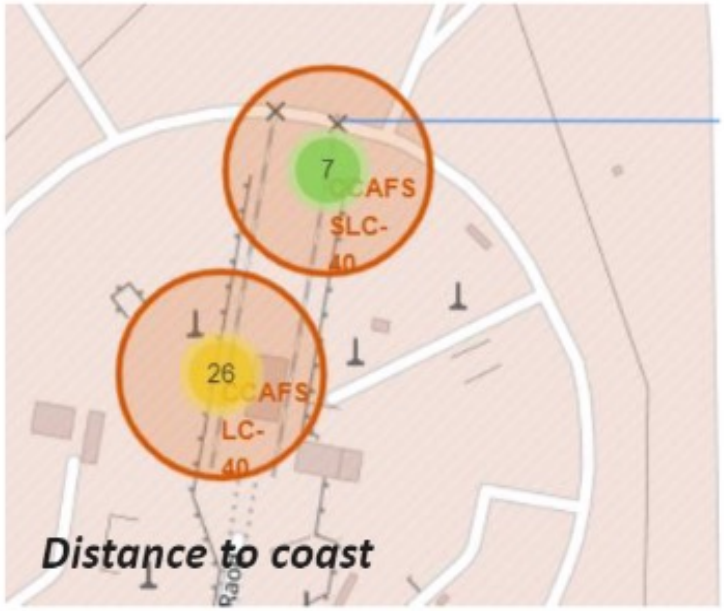
All launch sites global map markers

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

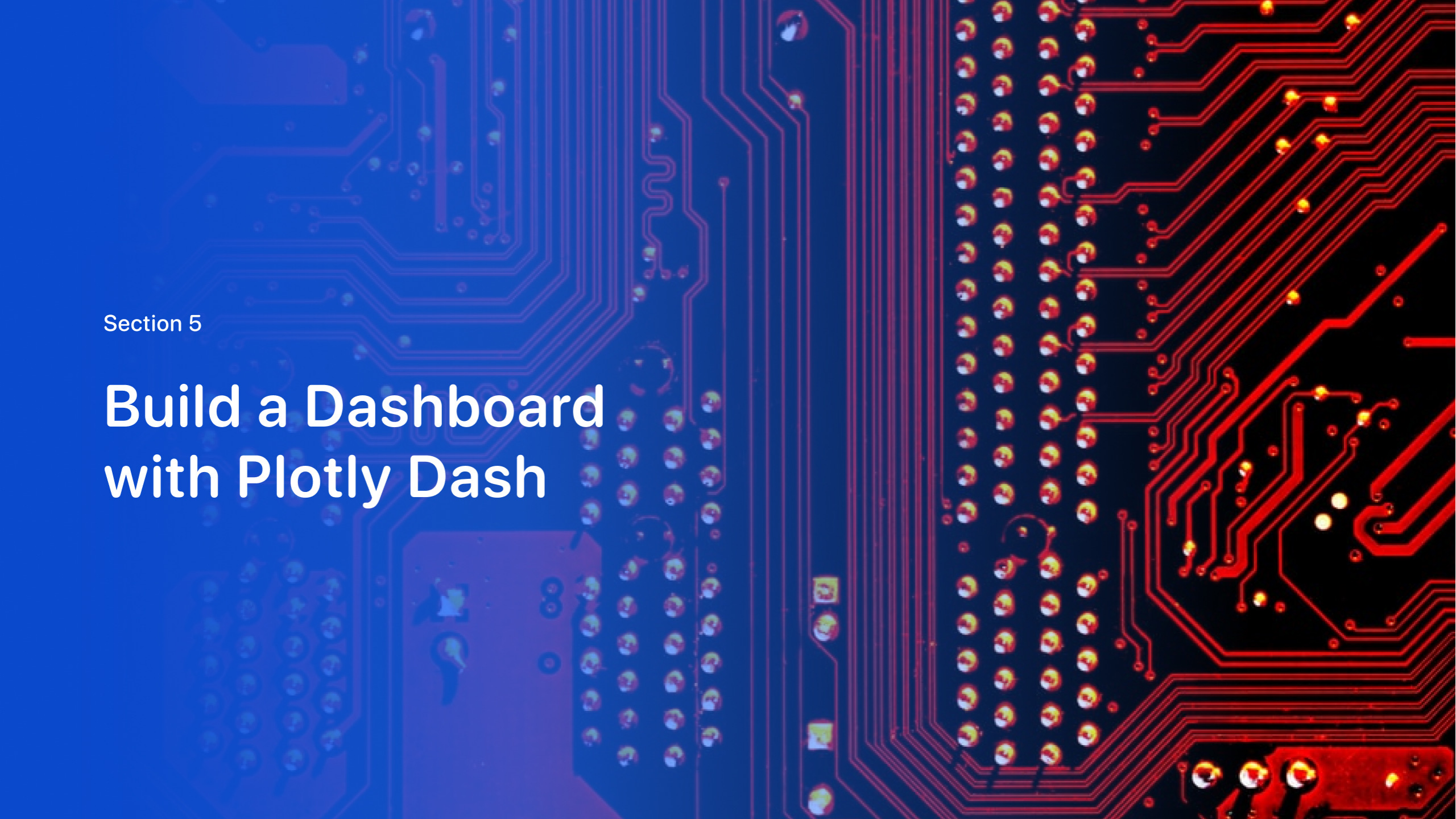
Markers showing launch sites with color labels

Launch Site distance to landmarks





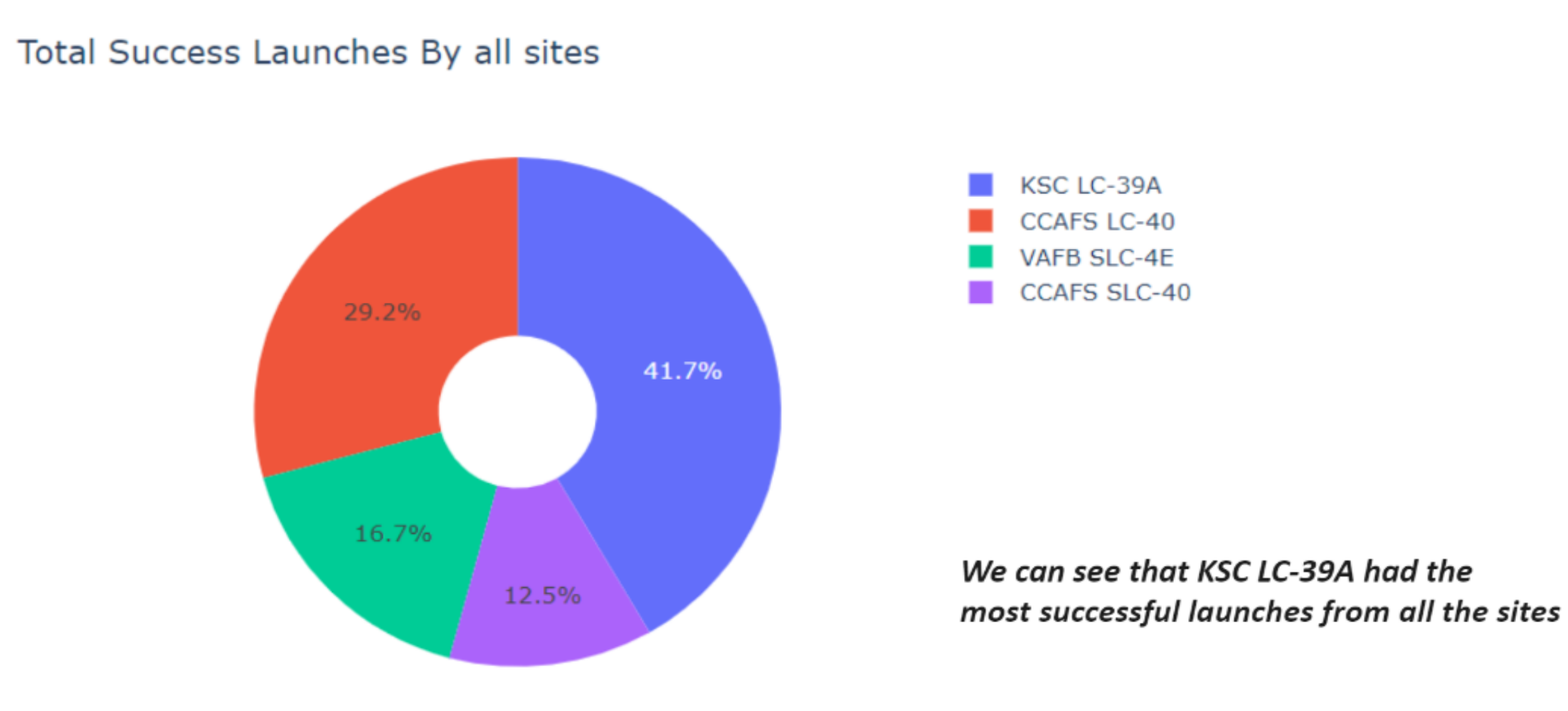
- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes



Section 5

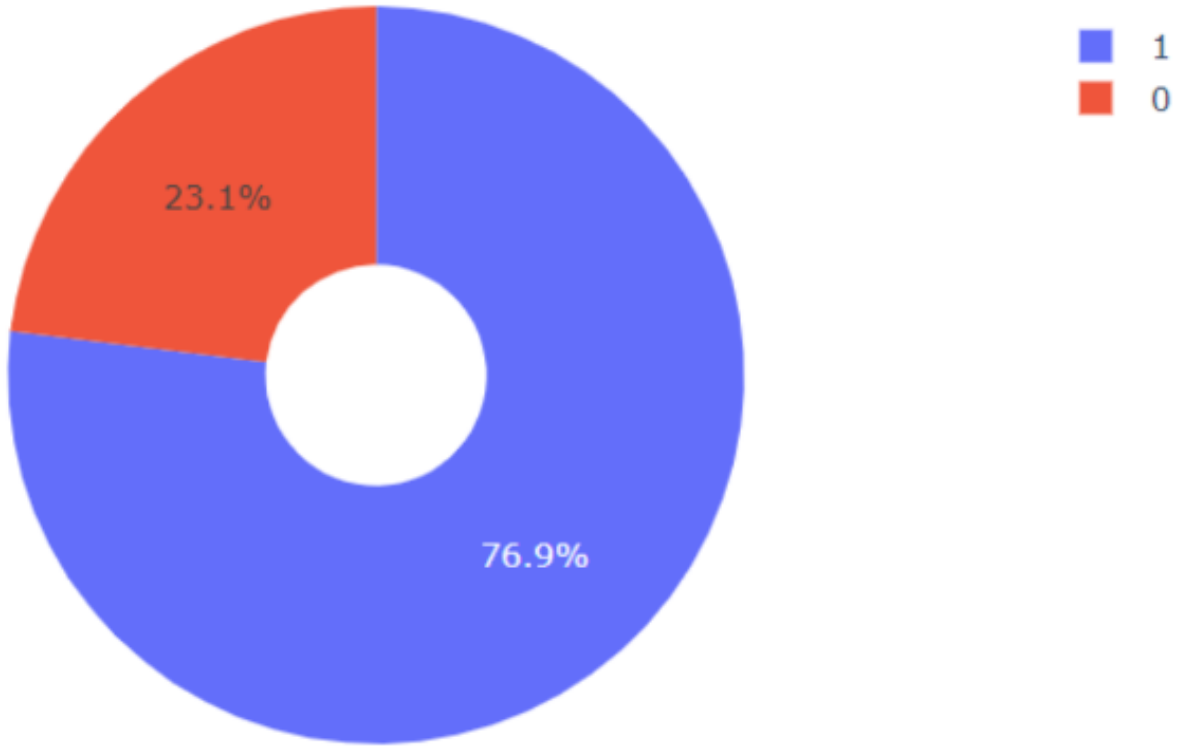
Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site



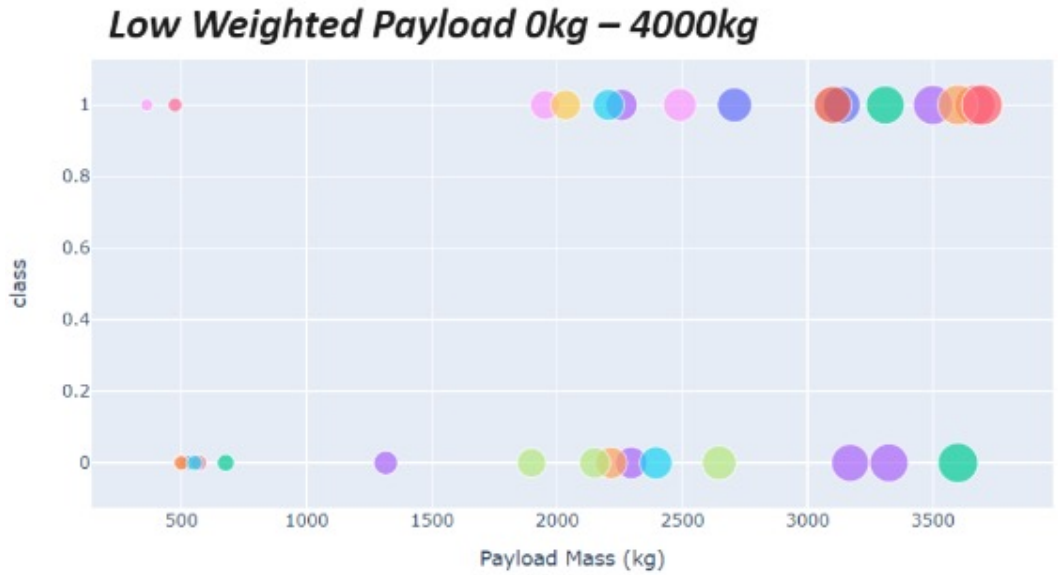
We can see that KSC LC-39A had the most successful launches from all the sites

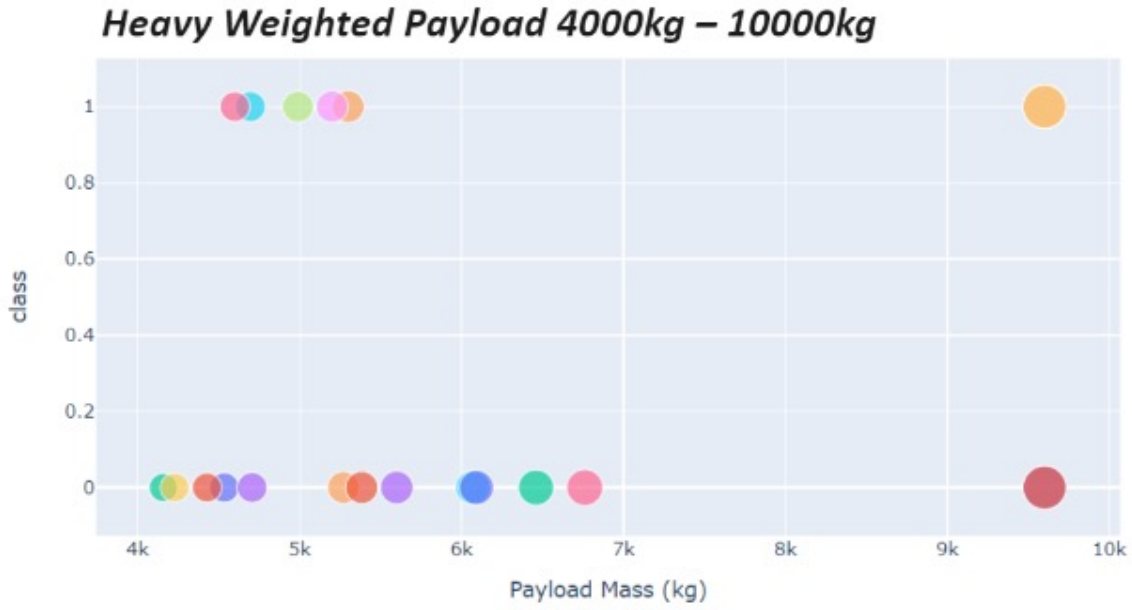
Pie chart showing the Launch site with the highest launch success ratio



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

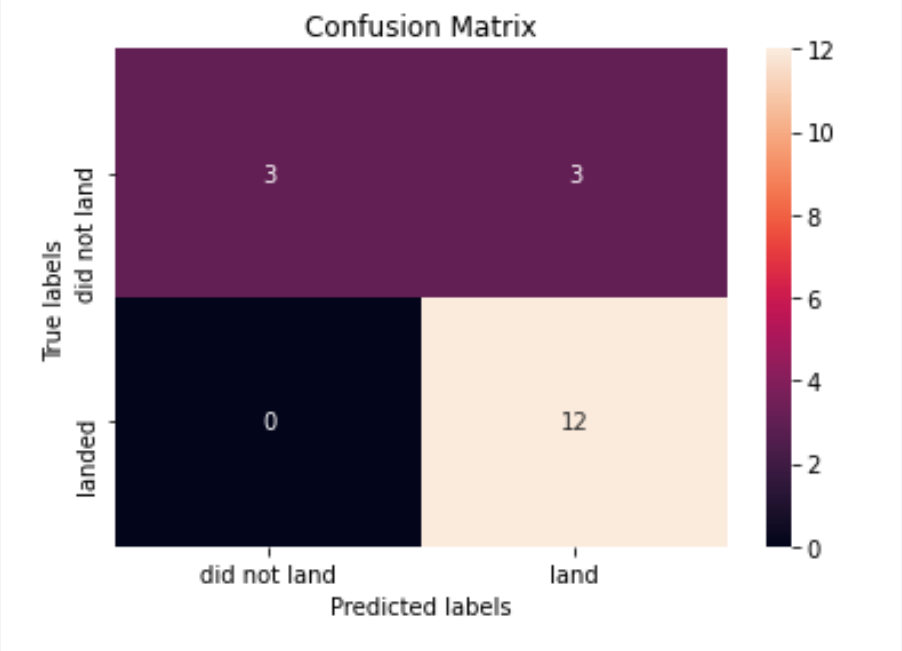
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

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

- The larger the flight amount at a launch site, the greater the success rate at a launch site.
- Launch success rate started to increase since 2013 till 2020 with slight setback in 2018.
- Orbits ES-L1, GEO, HEO, SSO, VLEO showed the best success rate.
- KSC LC-39A location had the biggest number of successful launches of any sites.
- The Decision tree classifier was the best machine learning algorithm for predicting failures and successes of SpaceX rocket launches.

Thank you!