IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Fulvio Pinna
3 September 2024

Outline

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

Executive Summary

- Methodologies
  - Data collection through API
  - Data collection with scraping
  - Data wrangling
  - EDA with SQL
  - EDA with data visualization
  - Visual analytics using Folium
  - Machine Learning Prediction
- Results
  - EDA result
  - Visual analytics in screenshots
  - Predictive analysis results

Introduction

- Project background and context

  SpaceX 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. The 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?

  - Interaction of the features that determine the success rate of a successful landing.

  - What conditions needs to be in place to ensure a successful landing program.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data has been collected using SpaceX API and web scraping from Wikipedia.
- Perform data wrangling
  - One hot encoding it has been 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

Data Collection

- Data has been collected using different methods

  - Data collection was performed using get request to the SpaceX API
  - Next, we transformed the response content as a Json and turn it into a pandas dataframe
  - Then data has been cleaned, filling missing values with the mean
  - After through web scraping we got data from Wikipedia for Falcon 9 launch using BeautifulSoup
  - The goal was to obtain the launch records as HTML table, parse the table and convert it to a dataframe

Data Collection – SpaceX API

- I used the get request to the SpaceX API to collect data, clean the requested data and did some basic data wrangling

Now let's start requesting rocket launch data from SpaceX API with the following URL:

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

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

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

In [11]:
```
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

- The link to the notebook is

[Data collection]

Calculate below the mean for the `PayloadMass` using the `.mean()`. Then use the mean and the `.replace()` function to replace `np.nan` values in the data with the mean you calculated.

In [55]:
```
# Calculate the mean value of PayloadMass column
media = data_falcon9['PayloadMass'].mean()
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'].replace(np.NaN, media, inplace=True)
```

Data Collection - Scraping

- Webscrap Falcon 9 launch records with BeautifulSoup from Wikipedia
- Parsed the table and converted it into a pandas dataframe.
- The link to the notebook is

[Web Scraping]

```
[4]: static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=10276
```

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

```
[5]: # use requests.get() method with the provided static_url
     # assign the response to a object
     response = requests.get(static_url)
```

```
[6]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
     soup = BeautifulSoup(response.text, 'html.parser')
```

Next, we just need to iterate through the `<th>` elements and apply the provided `extract_column_from_header()` to extract column name one by one

```
[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

      ths = first_launch_table.find_all('th')
      for th in ths:
          col = extract_column_from_header(th)
          if col is not None and len(col) > 0:
              column_names.append(col)
```

Data Wrangling

- I performed exploratory data analysis and determined the training labels.
- Calculated the number of launches for each site, and the number and occurrence of each orbits.
- Created landing outcome label from outcome column and exported the results to csv.
- The link to the notebook is Data Wrangling

EDA with Data Visualization

- I explored the data and show 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 Exploratory Data Analysis

EDA with SQL

- I applied EDA with SQL to get insight from the data and wrote 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.
- The link to the notebook is EDA with SQL

Build an Interactive Map with Folium

- I 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 (failure) and 1 (success).
- Calculated the distances between a launch site to its proximities and answered some question like:
  - Are launch sites near railways, highways and coastlines.
  - Do launch sites keep certain distance away from cities.
- The link to the notebook is Visual analytic with Folium

Build a Dashboard with Plotly Dash

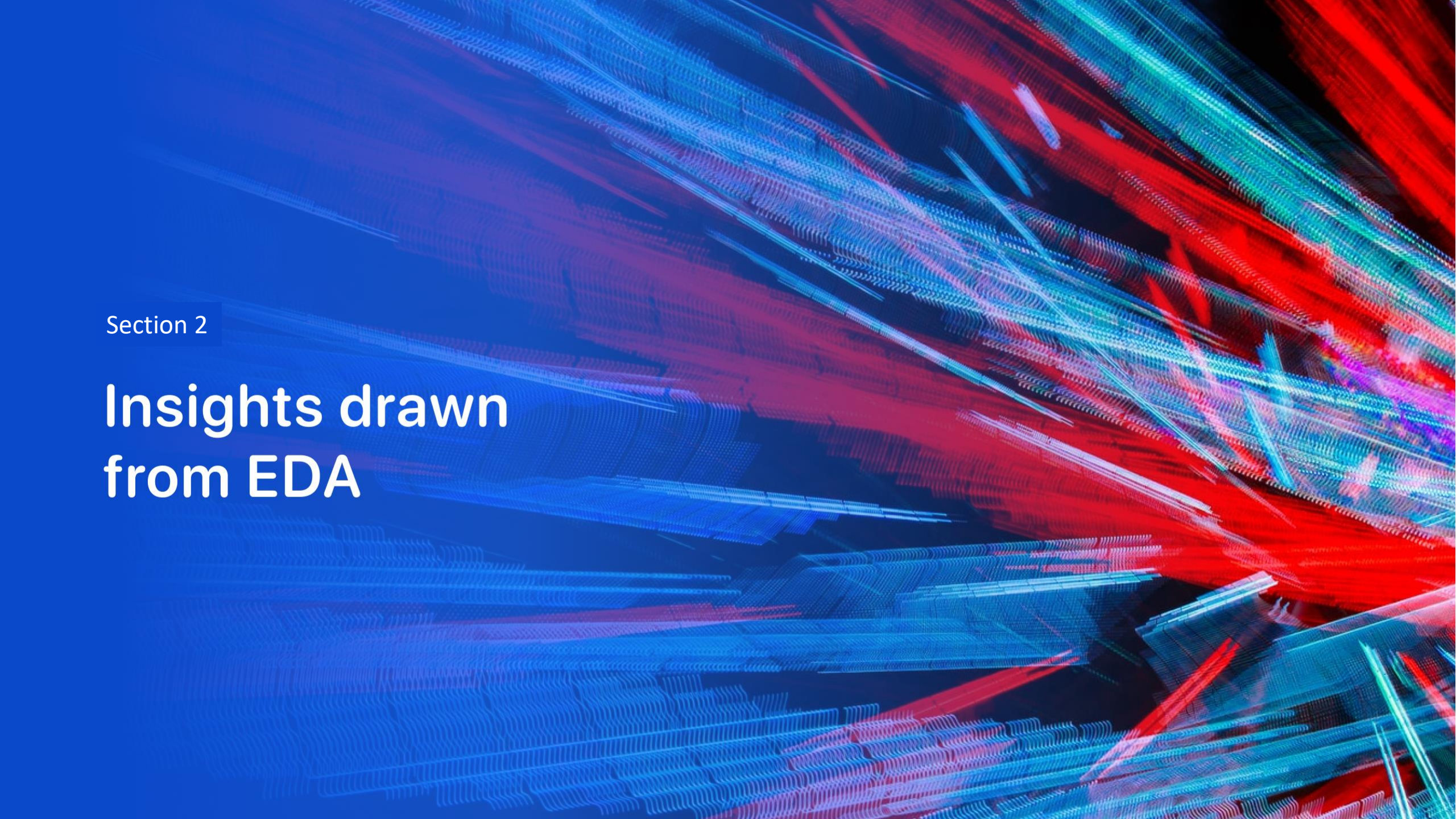
- I built an interactive dashboard with Plotly dash.
- Plotted pie charts showing the total launches by a certain sites.
- Plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.
- The link to the notebook is Dashboard

Predictive Analysis (Classification)

- I 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.
- The link to the notebook is Machine Learning Prediction

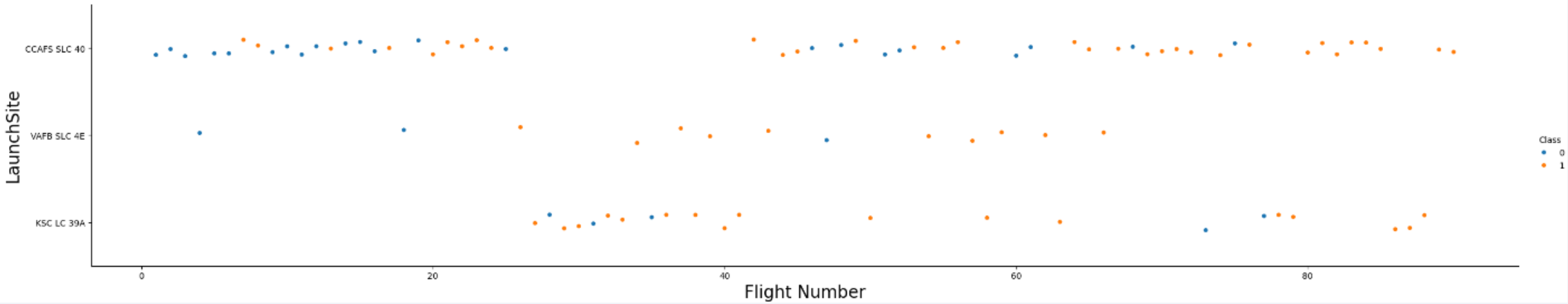
Results

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


Section 2
Insights drawn from EDA

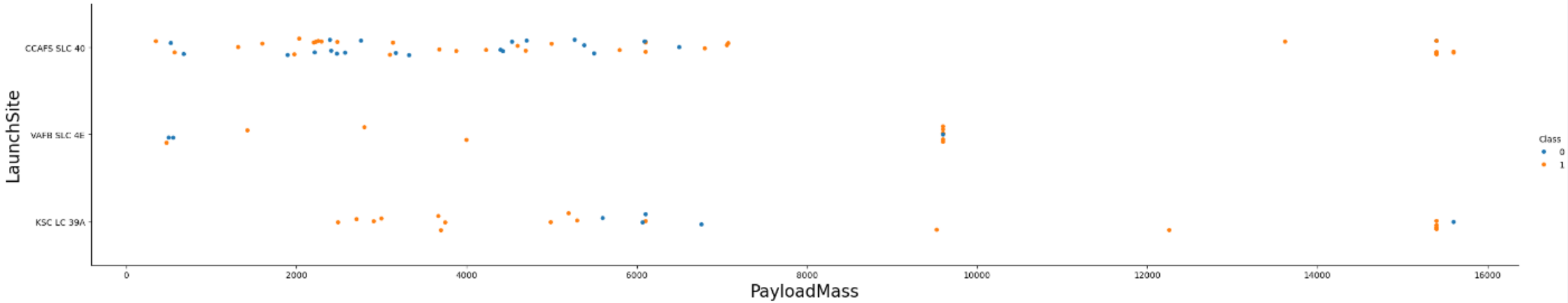
Flight Number vs. Launch Site

- From the plot, you can see that the larger is the number of flight at a launch site, the greater is the success rate for that launch site.



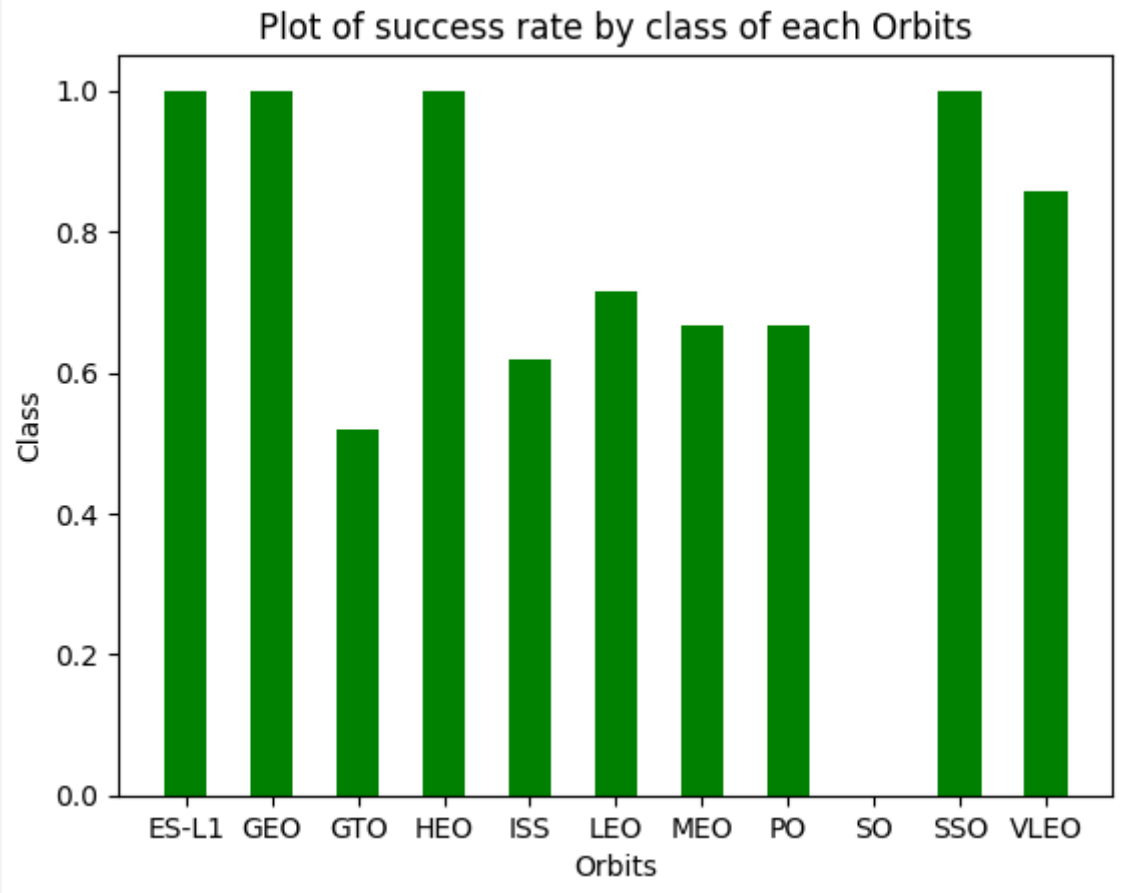
Payload vs. Launch Site

- Positive correlation between the payload mass for launch site with the success rate for the rocket.



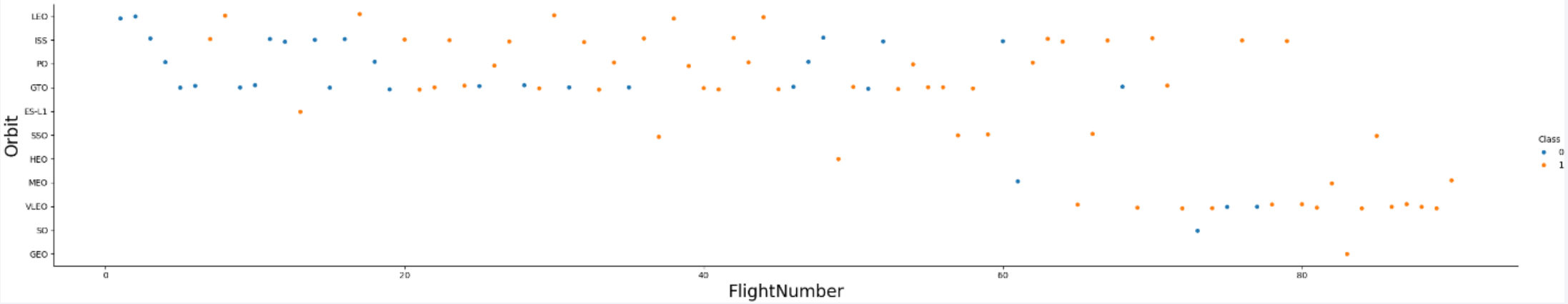
Success Rate vs. Orbit Type

- From the plot, is possible to find out that ES-L1, GEO, HEO, SSO, VLEO had the most success rate.



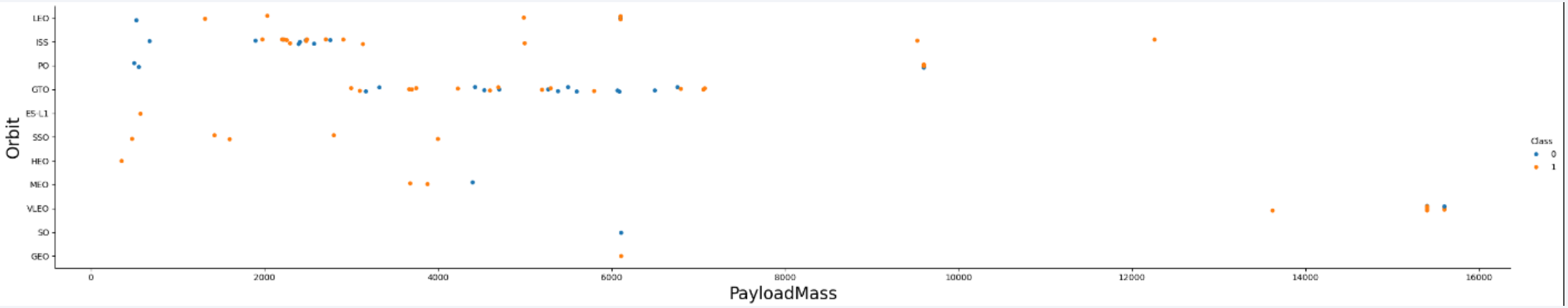
Flight Number vs. Orbit Type

- Is possible to generalize that for all the orbits the success for the next launch is higher as the number of attempts grow.



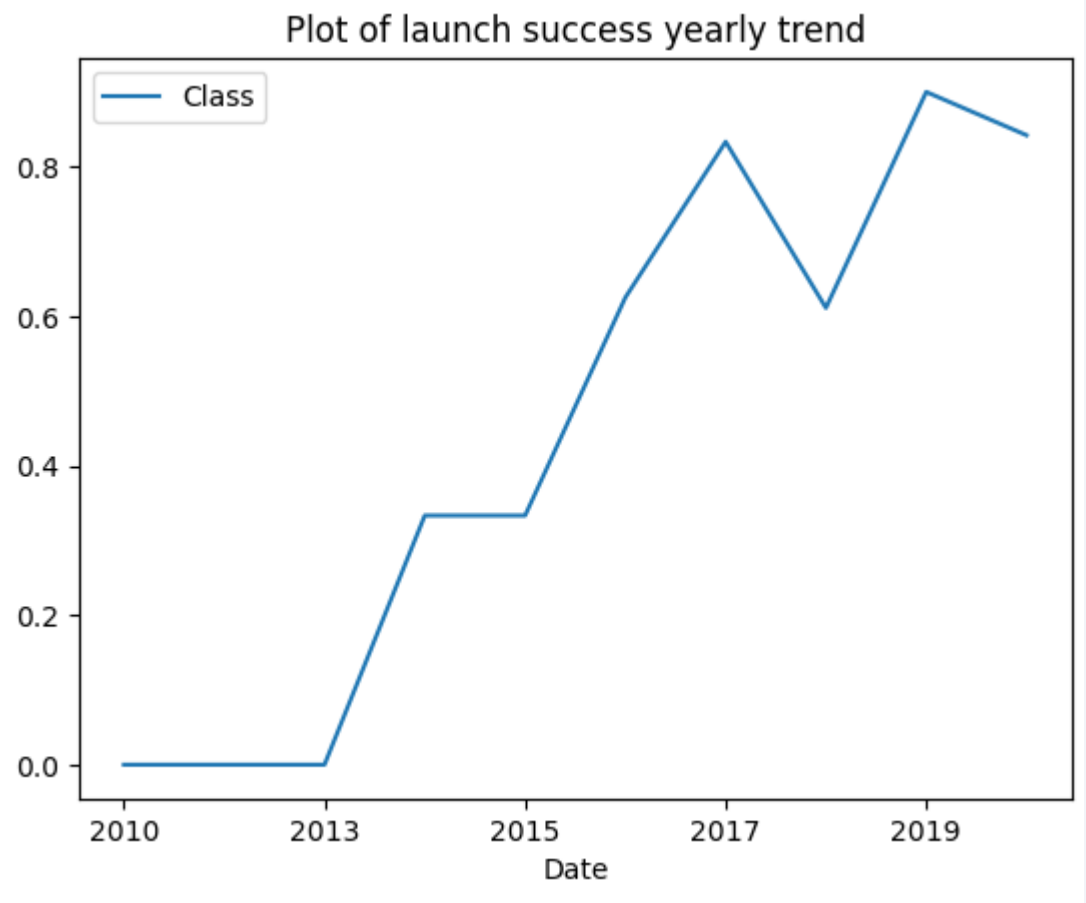
Payload vs. Orbit Type

- Only for the orbits PO, VLEO and ISS the payload mass are grater than 8000, and the majority of them shows a success.



Launch Success Yearly Trend

- The plot, show that the success rate since 2013 kept on increasing till 2020.



All Launch Site Names

- The command DISTINCT is been used to show only unique launch sites from the 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'

- The query above display the first 5 records of the database where launch sites contains `CCA`

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

```
%sql select * from SPACEXTBL WHERE ("Launch_Site" LIKE '%CCA%') LIMIT 5
```

```
 * 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

Total Payload Mass

- The total payload carried by boosters from NASA (CRS) is been calculated using the SUM function over PAYLOAD_MASS variable, filtering the database by customer.

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

```
%sql select SUM("PAYLOAD_MASS__KG_") from SPACEXTBL WHERE ("Customer" LIKE 'NASA (CRS)')
```

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

SUM("PAYLOAD_MASS__KG_")
45596

Average Payload Mass by F9 v1.1

- The average payload mass carried by booster version F9 v1.1 is 2928.4 and was calculated using the following command

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

```
%sql select AVG("PAYLOAD_MASS__KG_") from SPACEXTBL WHERE ("Booster_Version" LIKE 'F9 v1.1')
```

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

AVG("PAYLOAD_MASS__KG_")
2928.4

First Successful Ground Landing Date

- The date of the first successful landing outcome on ground pad was is been found using the MIN function and a simple filter on the Landing Outcome field.

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

Hint:Use min function

```
%sql select MIN(Date) from SPACEXTBL WHERE ("Landing_Outcome" LIKE 'Success (ground pad)')
```

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

MIN(Date)
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- Using WHERE function and filtering applying the AND condition for Landing Outcome and Payload Mass fields the result is show below.

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, "PAYLOAD_MASS__KG_", "Landing_Outcome" from SPACEXTBL WHERE ("Landing_Outcome" LIKE 'Success (drone ship)' AND "PAYLOAD_MASS__KG_" BETWEEN 4000 AND 6000)
```

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

Booster_Version	PAYLOAD_MASS__KG_	Landing_Outcome
F9 FT B1022	4696	Success (drone ship)
F9 FT B1026	4600	Success (drone ship)
F9 FT B1021.2	5300	Success (drone ship)
F9 FT B1031.2	5200	Success (drone ship)

Total Number of Successful and Failure Mission Outcomes

- Is possible to find out the number of success and failure using the simple function COUNT() and grouping the dataset for each distinct element of Landing Outcome.

List the total number of successful and failure mission outcomes

```
%sql select "Landing_Outcome", COUNT() from SPACEXTBL GROUP BY "Landing_Outcome"
```

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

Landing_Outcome	COUNT()
Controlled (ocean)	5
Failure	3
Failure (drone ship)	5
Failure (parachute)	2
No attempt	21
No attempt	1
Precluded (drone ship)	1
Success	38
Success (drone ship)	14
Success (ground pad)	9
Uncontrolled (ocean)	2

Boosters Carried Maximum Payload

- The booster that have carried the maximum payload is been selected 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

```
%sql select "Booster_Version", "PAYLOAD_MASS__KG_" from SPACEXTBL WHERE "PAYLOAD_MASS__KG_" LIKE (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

- Using a combinations of the WHERE, LIKE, AND, and the function “substr” to filter for year 2015 and failed landing outcomes in drone ship. The result is show below.

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.

```
%sql select substr(Date, 6,2) as Month, "Landing_Outcome", "Booster_Version", "Launch_Site" from SPACEXTBL WHERE ("Landing_Outcome" LIKE "%Failure (drone ship)%" AND substr(Date,0,5)='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

- Rank the count of landing outcomes between the date 2010-06-04 and 2017-03-20, in descending order.

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 "Landing_Outcome", COUNT() as Conto from SPACEXTBL WHERE (DATE BETWEEN "2010-06-04" AND "2017-03-20") GROUP BY "Landing_Outcome" ORDER BY Conto DESC
```

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

Landing_Outcome	Conto
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

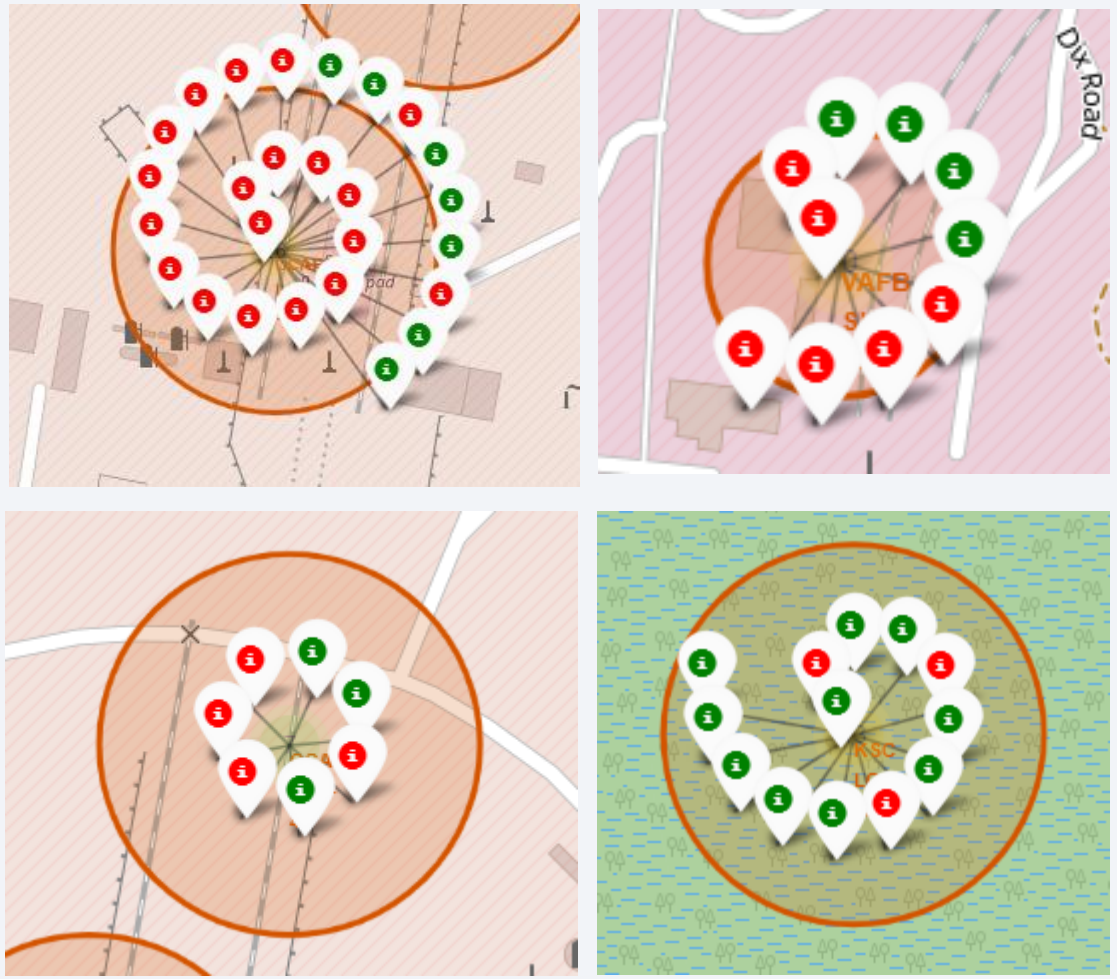
Section 3

Launch Sites Proximities Analysis

Lauch sites marked on global map

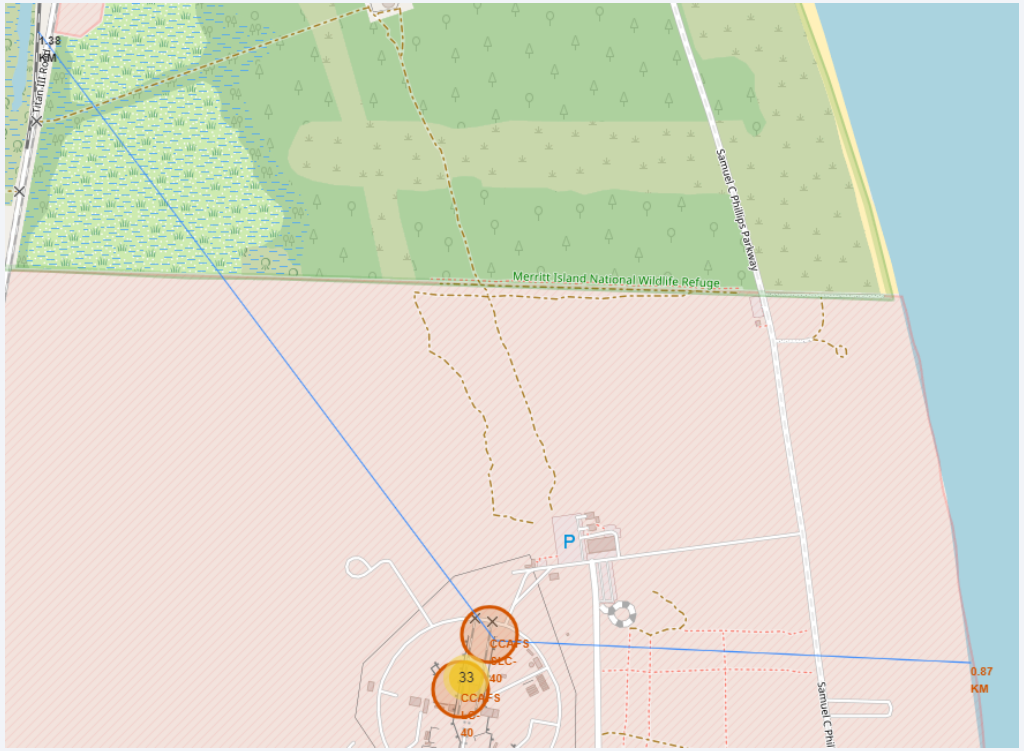
Launch sites with labels

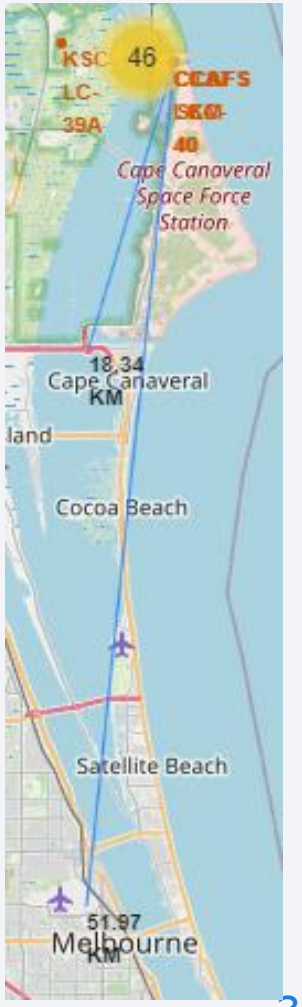
- Green markers shows when a launch is been a Success, otherwise the red ones represent the Failures.

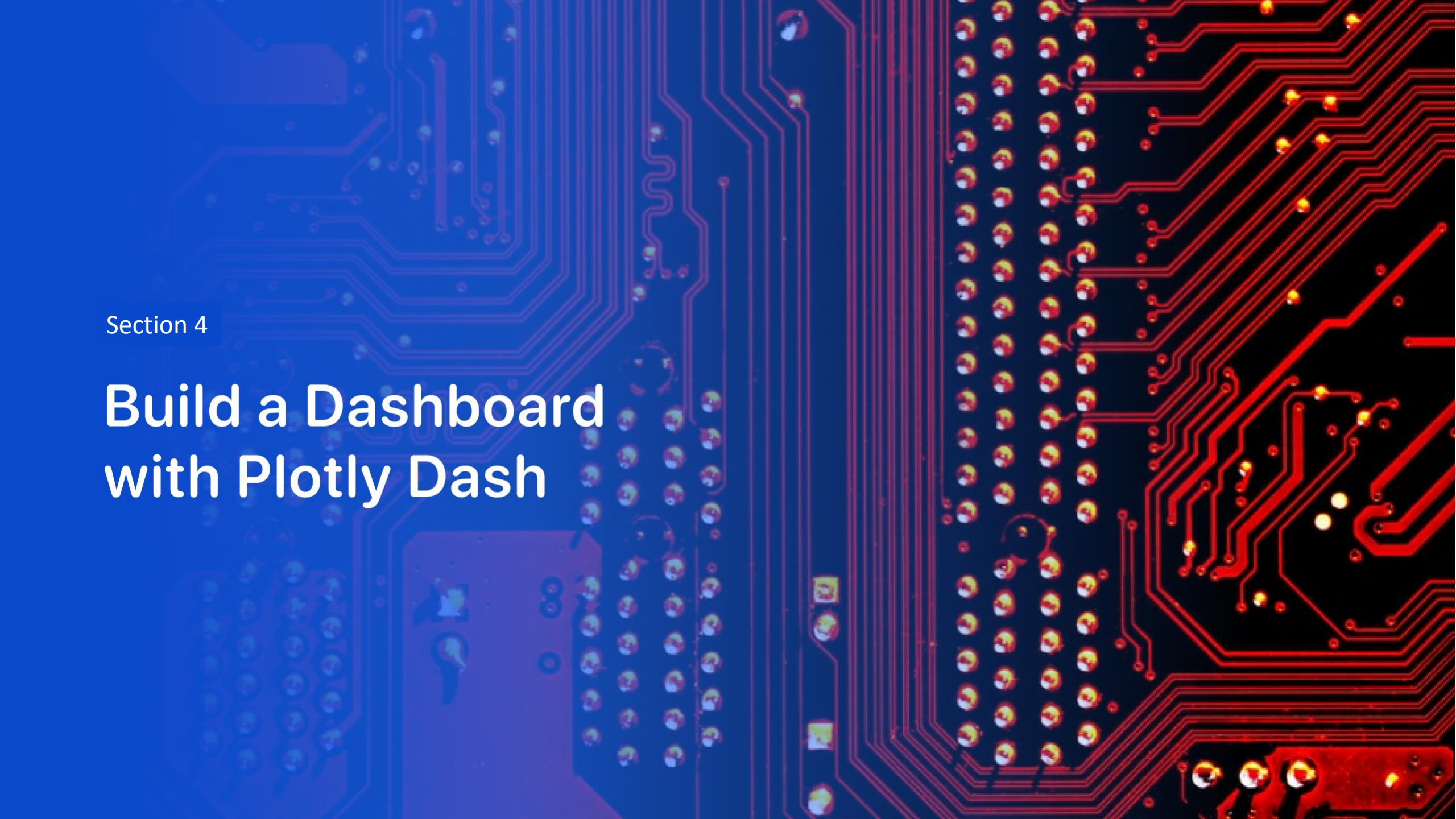


Distance to coast, city, railway and highway

- Is possible to notice that Florida's launch sites are closer to the coast (0.97 km) and the Railway (1.38 km), compare to the highway (18.34 km) and Melbourne (51.07 km).

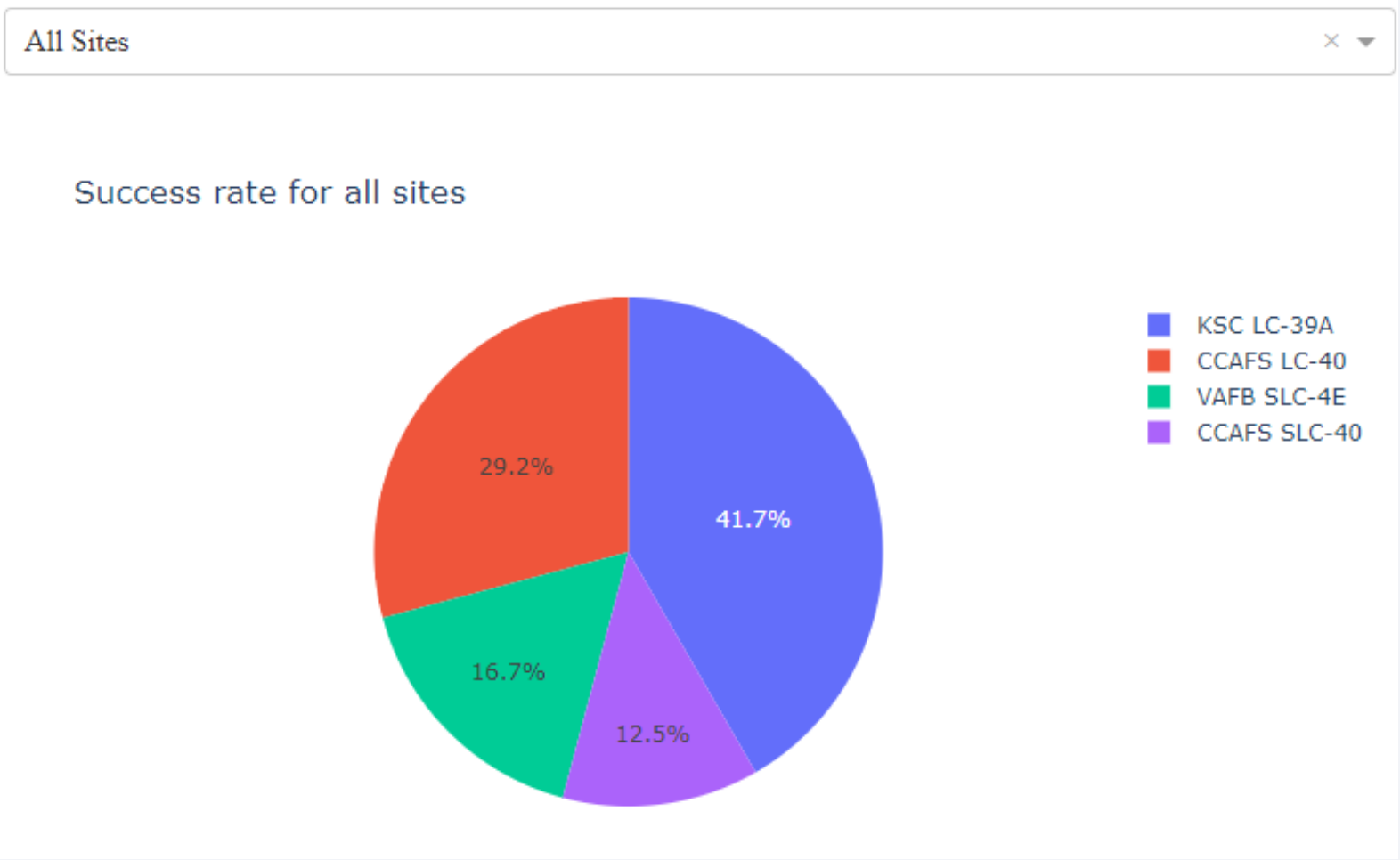





Section 4
Build a Dashboard with Plotly Dash

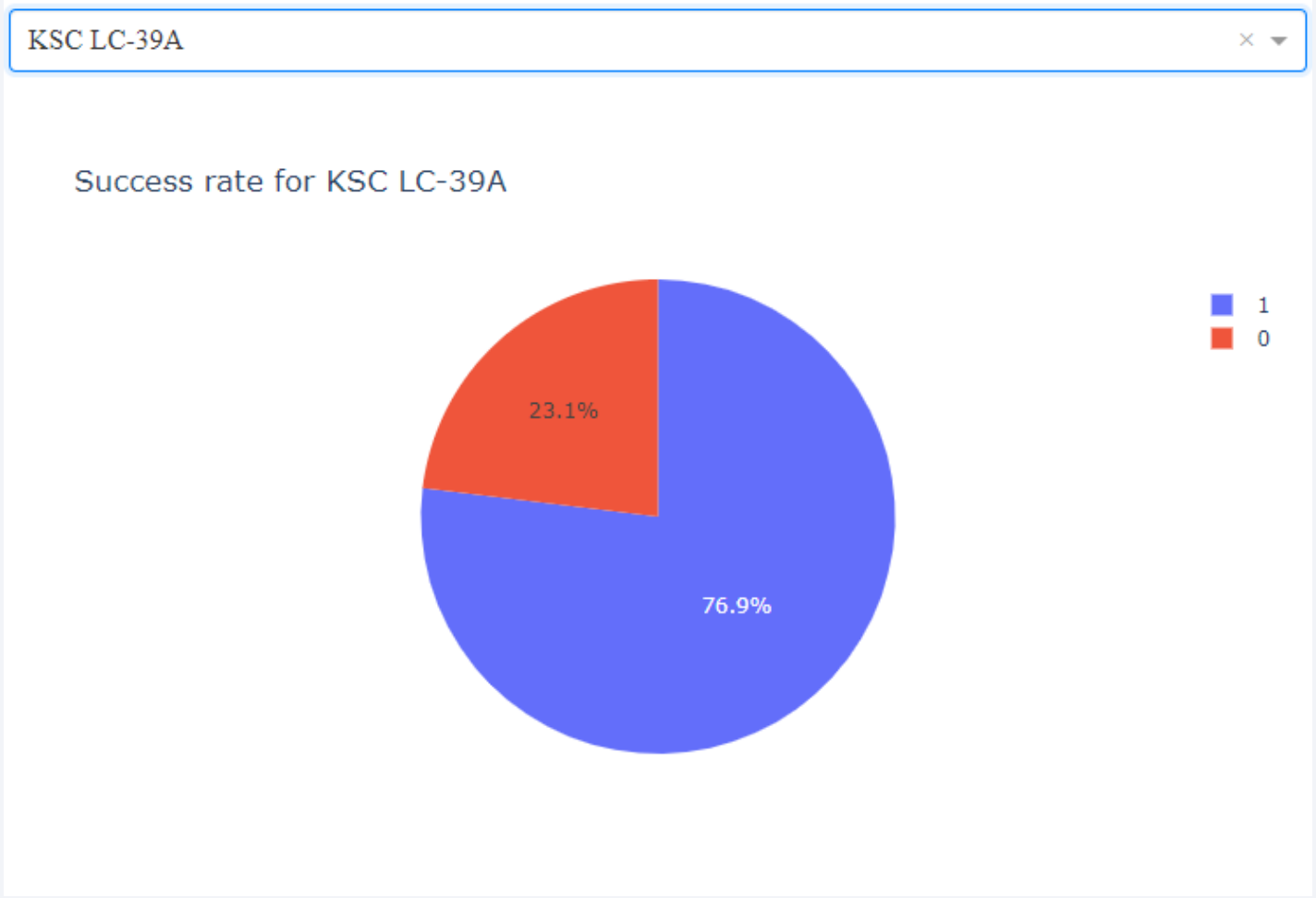
Success ratio by launch sites

- Find out that KSC LC has the most successful launches



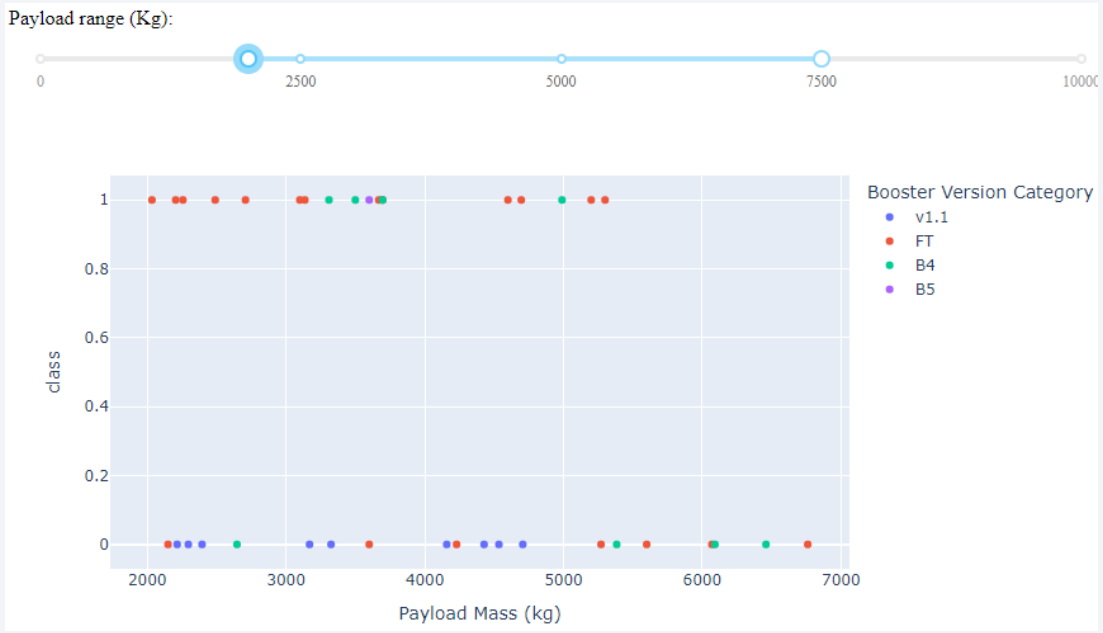
Success ratio for KSC LC

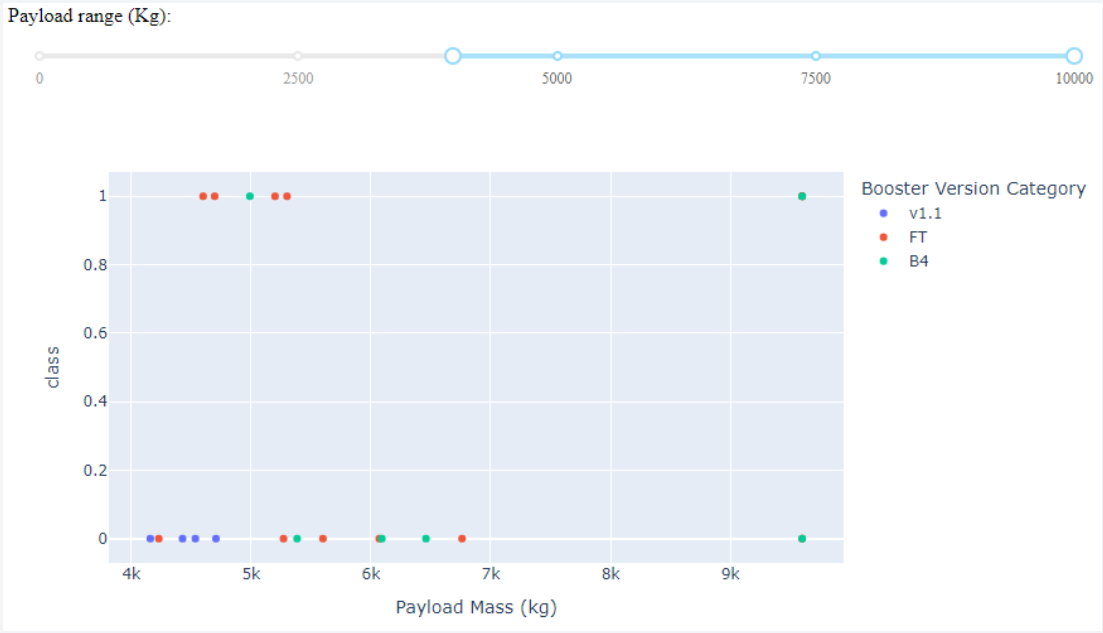
- KSC LC at the moment show a percentage of success of almost 77%



Scatter plot of Payload vs Launch Outcome for all sites

- Success rate for lower weighted payload is higher compared to the heavy weighted.





Section 5

Predictive Analysis (Classification)

Classification Accuracy

- Decision tree is the model with the best classification score

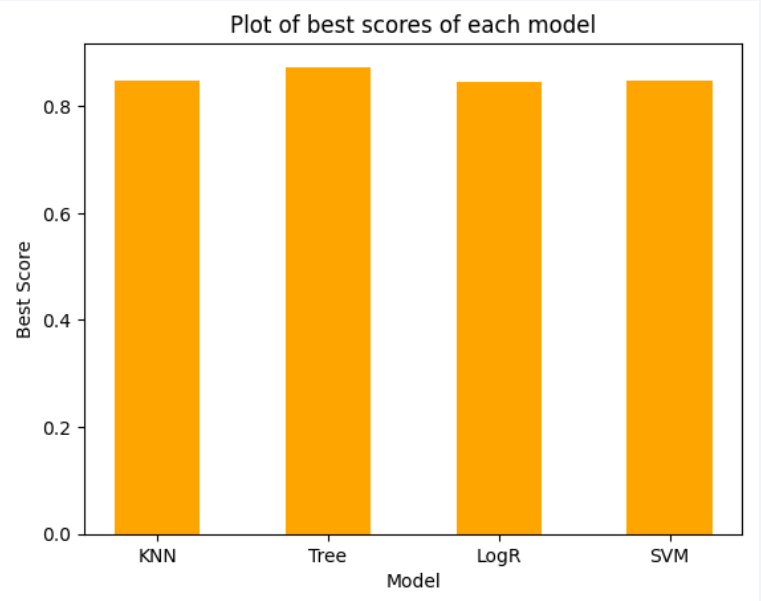
```
algo = {'KNN':(knn_cv.best_score_, knn_cv.best_params_), 'Tree':(tree_cv.best_score_, tree_cv.best_params_),
        'LogR':(logreg_cv.best_score_, logreg_cv.best_params_), 'SVM': (svm_cv.best_score_, svm_cv.best_params_)}

bestalgo = max(algo, key=algo.get)
print('Best model is', bestalgo,'with a score of', algo[bestalgo][0], '\n and parameters: ', algo[bestalgo][1])

y = (list(x[0] for x in algo.values()))
x = (list(algo.keys()))

plt.bar(x, y, color='orange', width=0.5)
plt.title('Plot of best scores of each model', fontdict={'size':12})
plt.ylabel('Best Score', fontsize = 10)
plt.xlabel('Model', fontsize = 10)
```

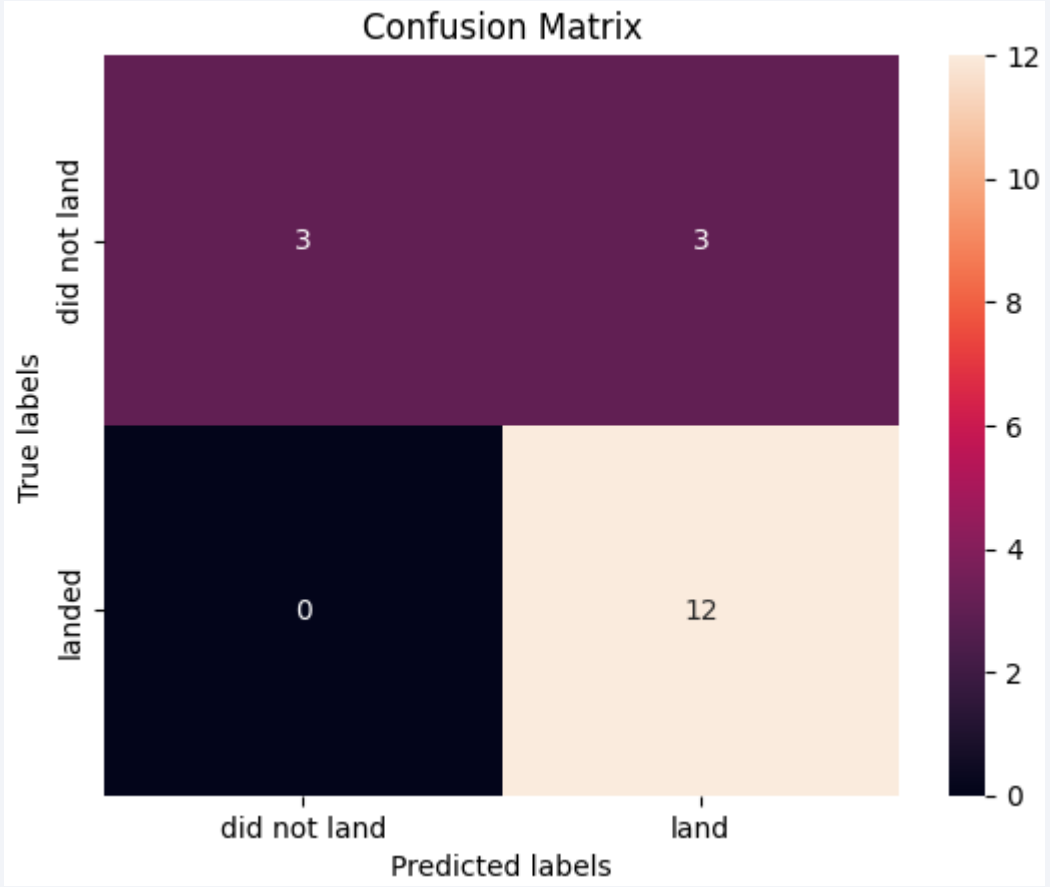
```
Best model is Tree with a score of 0.8732142857142856
 and parameters:  {'criterion': 'gini', 'max_depth': 6, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```



Confusion Matrix

- Confusion matrix for the decision tree classifier shows that the model can distinguish between the different classes.

- The main error seems to be the false positives, 3 unsuccessful launch are labeled as successful by the classifier.



Conclusions

- The higher are the number of flight for a launch site, the better is the success rate for that launch site.
- Launch success rate show a strong increment from 2013 till today.
- Orbits ES-L1, GEO, HEO, SSO, VLEO has the most success rate.
- KSC LC-39A has the most successful launches of any sites.
- The Decision tree classifier is the best machine learning algorithm for this project.

Thank you!