Winning Space Race with Data Science

Sharmistha Kundu

25th October,2023

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

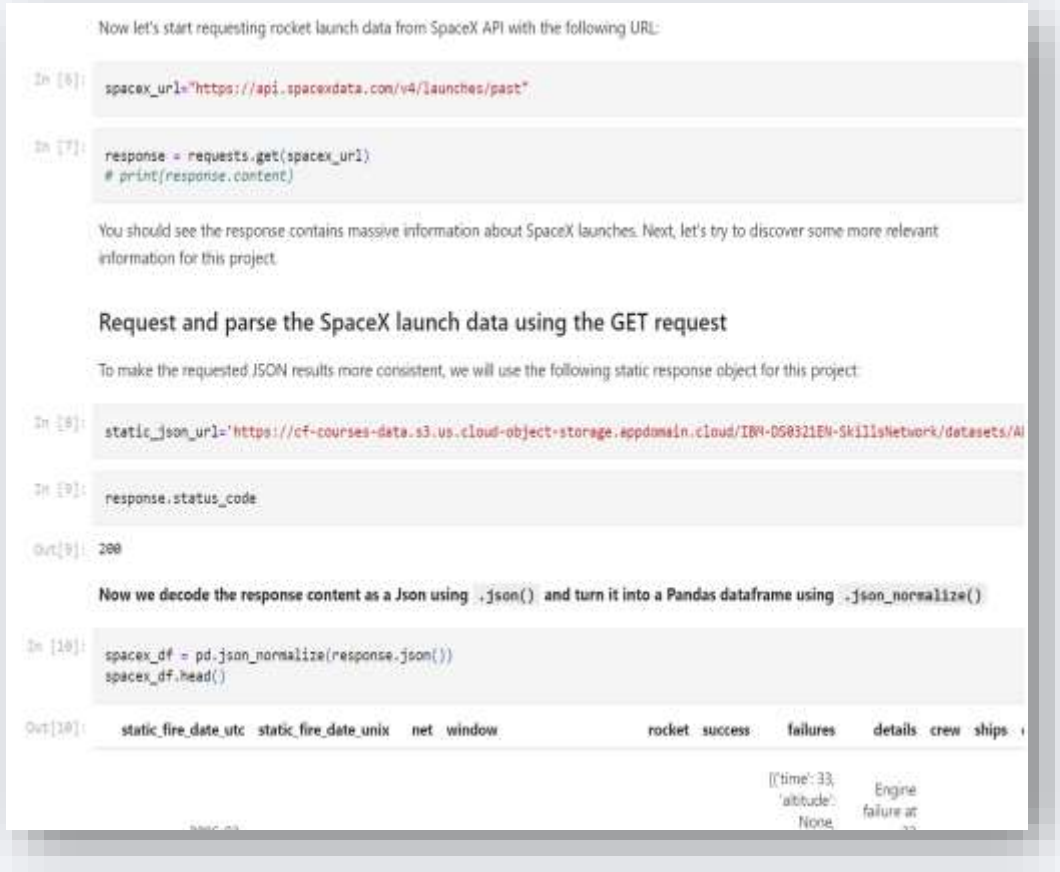
Data Collection

- **The data was collected using various methods**
  - Data collection was done using get request to the SpaceX API.
  - Next, I decoded the response content as a Json using .json() function call and turn it into a pandas dataframe using .json_normalize().
  - Then I cleaned the data, checked for missing values and fill in missing values where necessary.

Data Collection – SpaceX API

- First I 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/SharmisthaKundu98/Data_Science_portfolio/blob/main/Spacex_project/Data_collection_with_API.ipynb



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)
        # print(response.content)
```

You should see the response contains massive information about SpaceX launches. Next, let's try to discover some more relevant information for this project.

Request and parse the SpaceX launch data using the GET request

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

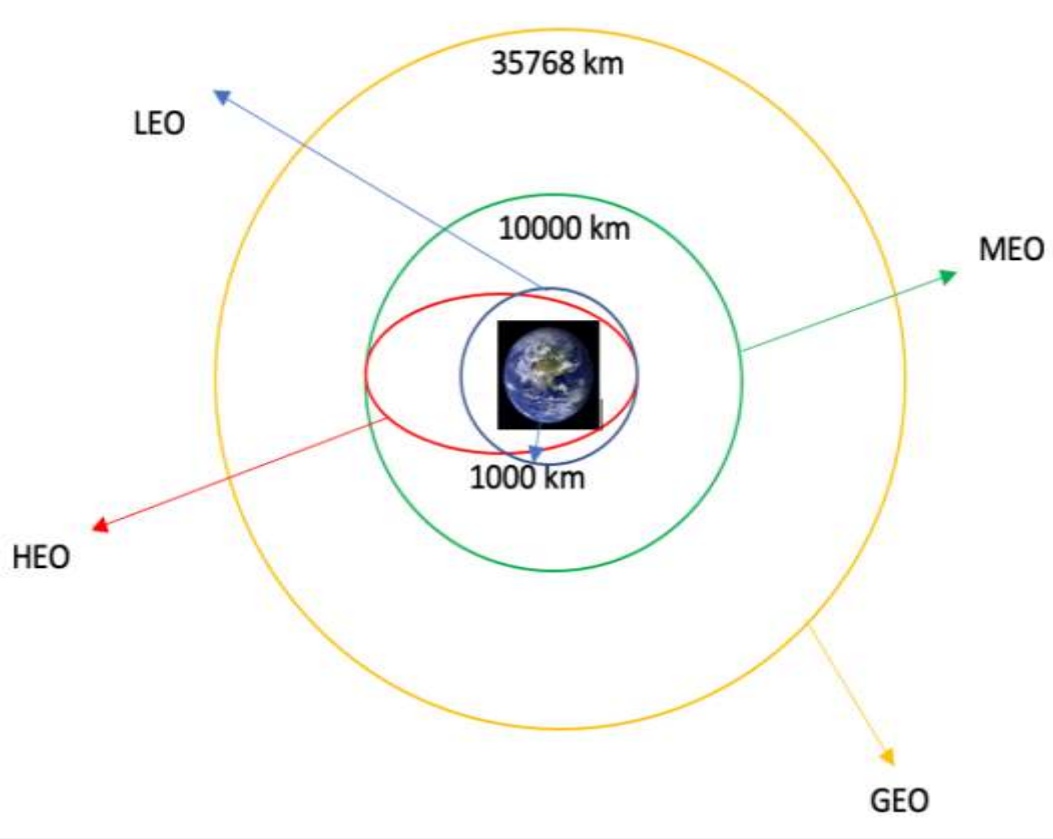
```
In [8]: static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/A

In [9]: response.status_code

Out[9]: 200
```

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

```
In [10]: spacex_df = pd.json_normalize(response.json())
         spacex_df.head()
```

	static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures	details	crew	ships
							[{'time': 33, 'altitude': None,	Engine failure at		

Data Wrangling

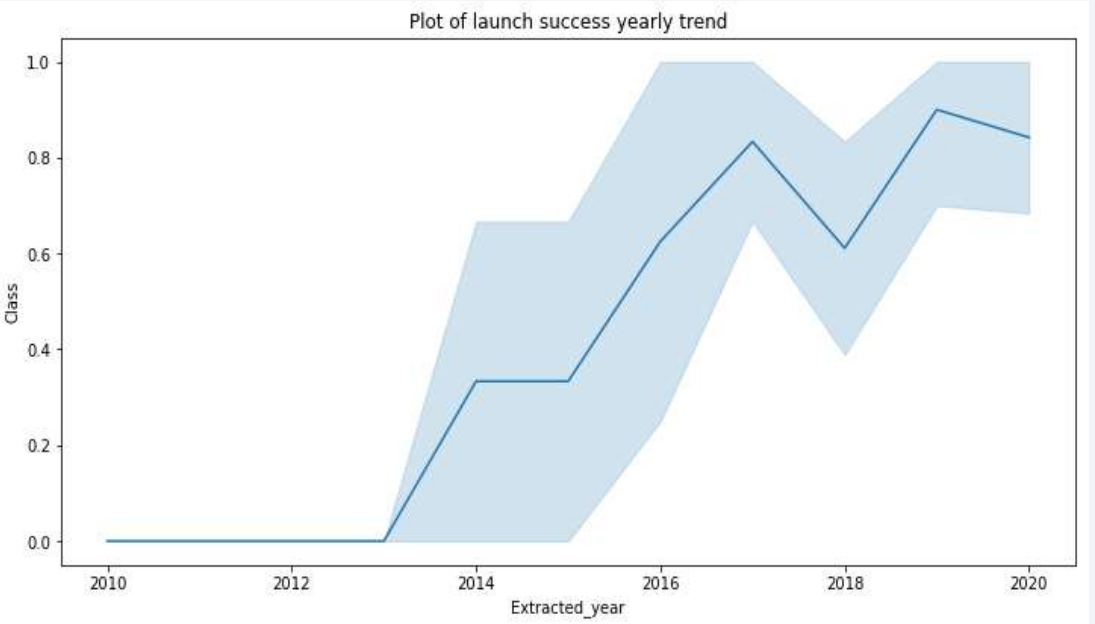


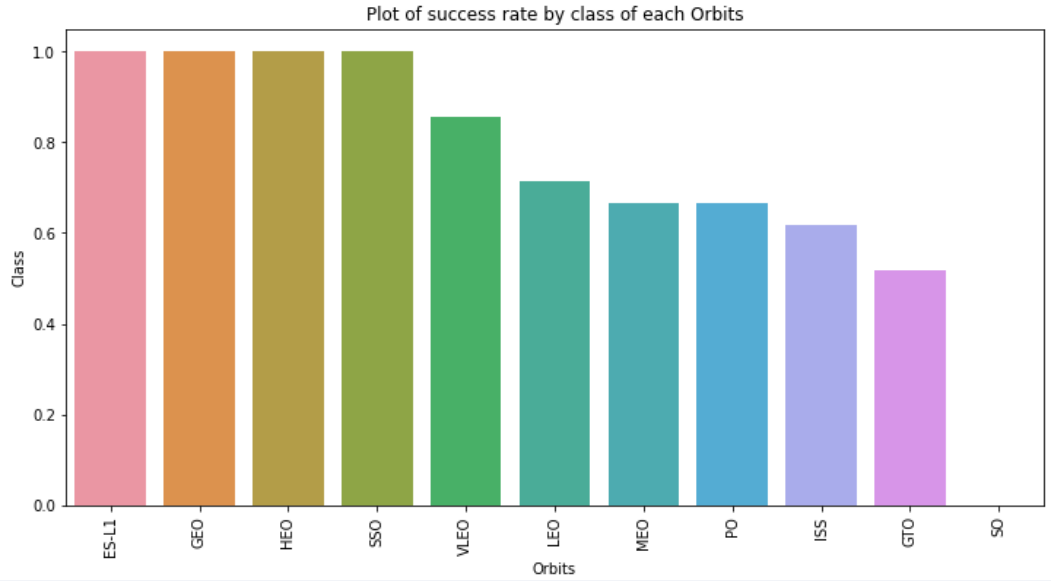
- I performed exploratory data analysis and determined the training labels.
- Then I calculated the number of launches at each site, and the number and occurrence of each orbits
- Then I created landing outcome label from outcome column and exported the results to csv.
- The link to the notebook is -

  https://github.com/SharmisthaKundu98/Data_Science_portfolio/blob/main/Spacex_project/Data_wrangling.ipynb

EDA with Data Visualization

- I 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/SharmisthaKundu98/Data_Science_portfolio/blob/main/Spacex_project/eda_with_visualization.ipynb

EDA with SQL

- First I loaded the SQL extension and establish a connection with sqlite3 database and then loaded the SpaceX dataset into the database without leaving the jupyter notebook.
- I 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/SharmisthaKundu98/Data_Science_portfolio/blob/main/Spacex_project/EDA_with_SQL.ipynb

Build an Interactive Map with Folium

- First 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.
- Then I 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.

Build a Dashboard with Plotly Dash

- I built an interactive dashboard with Plotly dash
- Then I plotted pie charts showing the total launches by a certain sites
- I 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/SharmisthaKundu98/Data_Science_portfolio/blob/main/Spacex_project/visalization_with_plotly_dash.ipynb

Predictive Analysis (Classification)

- I loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- I built different machine learning models and tune different hyperparameters using GridSearchCV.
- I used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- Then I found the best performing classification model.
- The link to the notebook is - https://github.com/SharmisthaKundu98/Data_Science_portfolio/blob/main/Spacex_project/machine_Learning_Prediction.ipynb

Results

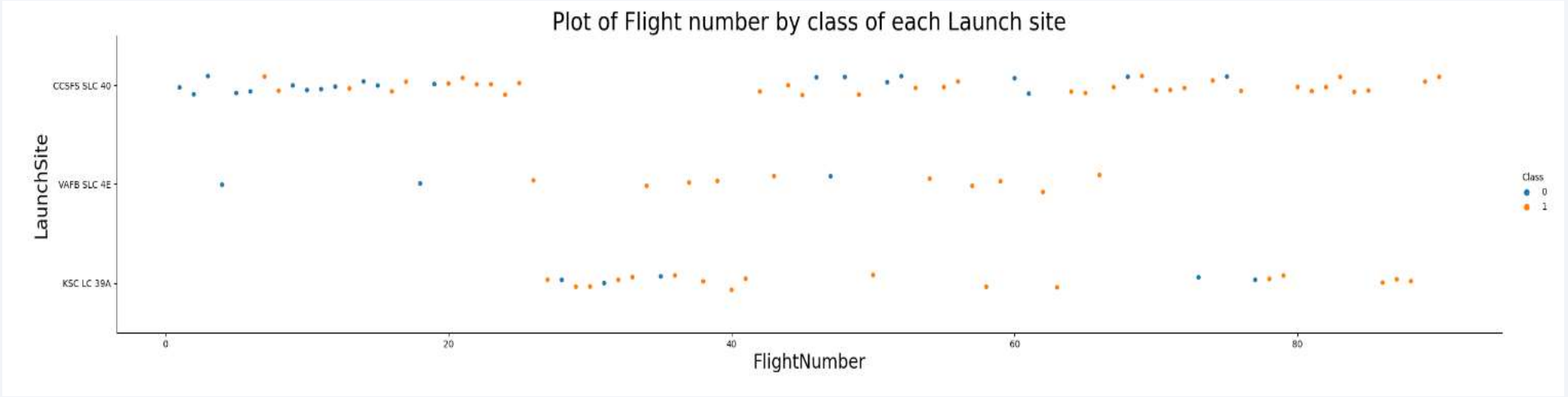
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

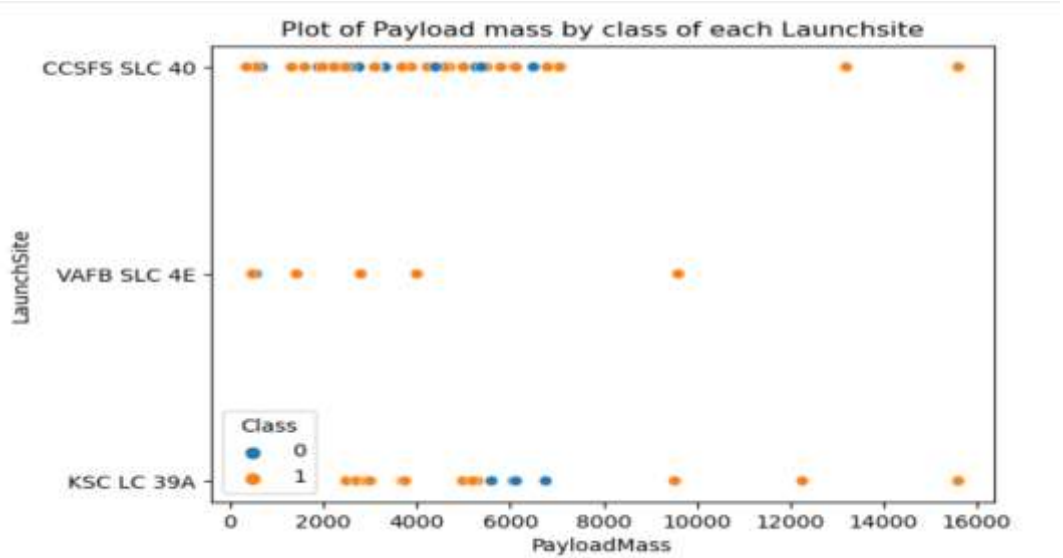
Flight Number vs. Launch Site

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



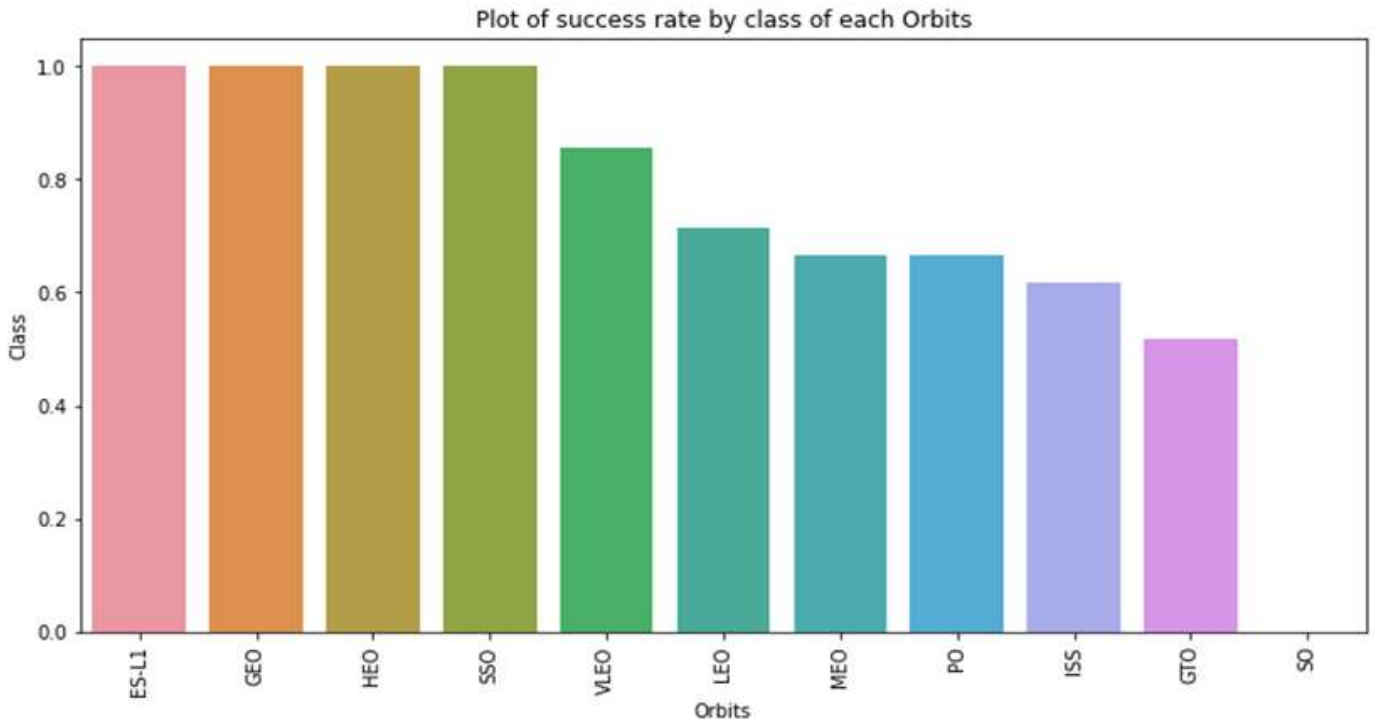
Payload vs. Launch Site

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



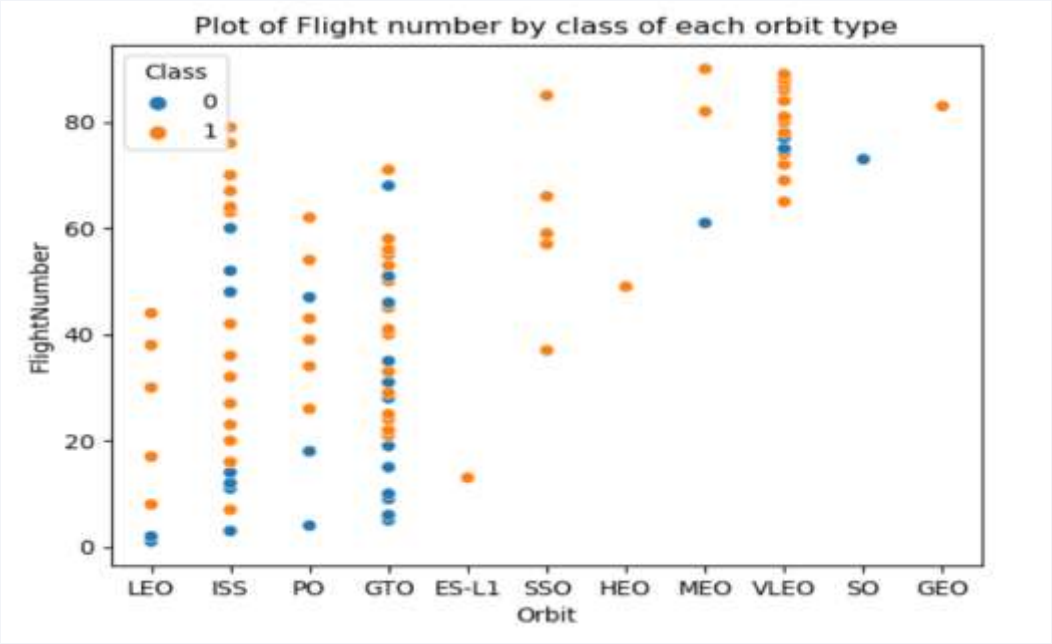
Success Rate vs. Orbit Type

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



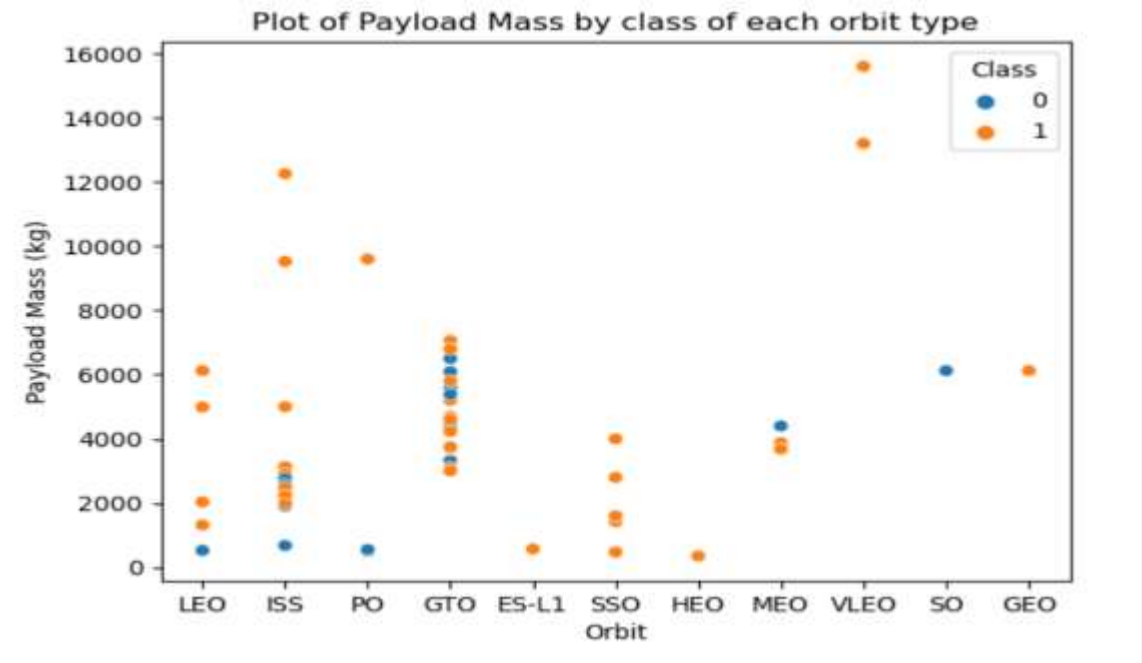
Flight Number vs. Orbit Type

- The plot below shows the Flight Number vs. Orbit type. I 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.



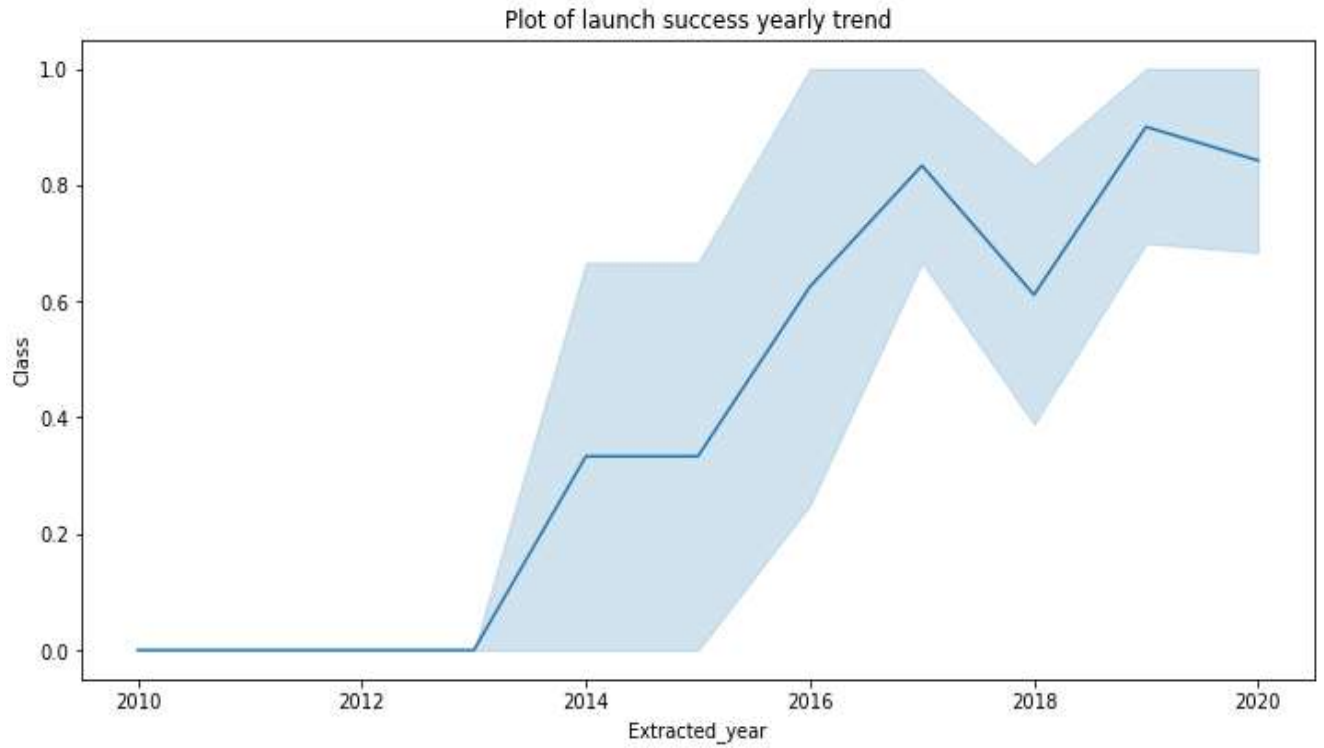
Payload vs. Orbit Type

- Here I can observe that with heavy payloads, the successful landing are more for LEO, ISS and PO orbits.



Launch Success Yearly Trend

- From the plot, I can observe that success rate since 2013 kept on increasing till 2020.



All Launch Site Names

- Here I 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 SPACEXTABLE
```

```
 * 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 * from SPACEXTABLE 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
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-08-12	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	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

- Here I used the query above to display 5 records where launch sites begin with `CCA`

Total Payload Mass

- I 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 from SPACEXTABLE where Customer = 'NASA (CRS)'
```

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

TOTAL
45596

Average Payload Mass by F9 v1.1

- I 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 Avg_of_payload_mass from SPACEXTABLE where Booster_Version = 'F9 v1.1'
```

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

Avg_of_payload_mass
2928.4

First Successful Ground Landing Date

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

The first succesful ground landing date -

```
%sql select min(Date) as First_successful_landing_date from SPACEXTABLE where Landing_Outcome = 'Success (ground pad)'
```

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

First_successful_landing_date
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 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 Distinct Booster_Version from SPACEXTABLE \
where Landing_Outcome = 'Success (drone ship)' and (PAYLOAD_MASS__KG_ > 4000 and PAYLOAD_MASS__KG_ < 6000)
```

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

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

- Here I 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

Total Number of Successful and Failure Mission Outcomes

List the total number of successful and failure mission outcomes

```
%sql select Mission_Outcome,count(*) as total_outcomes from SPACEXTABLE group by Mission_Outcome
```

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

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

```
%sql select count(Mission_Outcome) as total_success_outcomes from SPACEXTABLE where Mission_Outcome like 'Success%'
```

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

total_success_outcomes
100

```
%sql select count(Mission_Outcome) as total_failure_outcomes from SPACEXTABLE where Mission_Outcome like 'Failure%'
```

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

total_failure_outcomes
1

- I used wildcard like '%' to filter for **WHERE** Mission Outcome was a success or a failure.

Boosters Carried Maximum Payload

- I determined the booster that have carried the maximum payload 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.

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

```
 * 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

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

List 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 we need to use `substr(Date, 6, 2)` as month to get the months and SUBSTR(Date, 1, 4)='2015' for year.

SUBSTR(Date_string, starting position, for how many charaters)

```
%sql select substr(Date,6,2) as Month, Landing_Outcome, Booster_Version, Launch_Site from SPACEXTABLE \
where Landing_Outcome = 'Failure (drone ship)' and SUBSTR(Date, 1, 4) = '2015'
```

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

Month	Landing_Outcome	Booster_Version	Launch_Site
10	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

```
%sql select Landing_Outcome, count(*) as Total_outcome,\
RANK() Over(order by count(*) DESC) as outcome_rank \
from SPACEXTABLE \
where Date >= '2010-06-04' AND Date <= '2017-03-20' \
Group by Landing_Outcome
```

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

Landing_Outcome	Total_outcome	outcome_rank
No attempt	10	1
Success (ground pad)	5	2
Success (drone ship)	5	2
Failure (drone ship)	5	2
Controlled (ocean)	3	5
Uncontrolled (ocean)	2	6
Precluded (drone ship)	1	7
Failure (parachute)	1	7

- Here I selected Landing outcomes and the **COUNT** of all landing outcomes from the data and used **OVER** the **ORDER BY** clause to order the grouped landing outcome in descending order **and RANK()** to rank the outcomes.
- Then I used the **WHERE** clause to filter for landing outcomes from 2010-06-04 to 2010-03-20.
- I applied the **GROUP BY** clause to group the landing outcomes .

Section 4

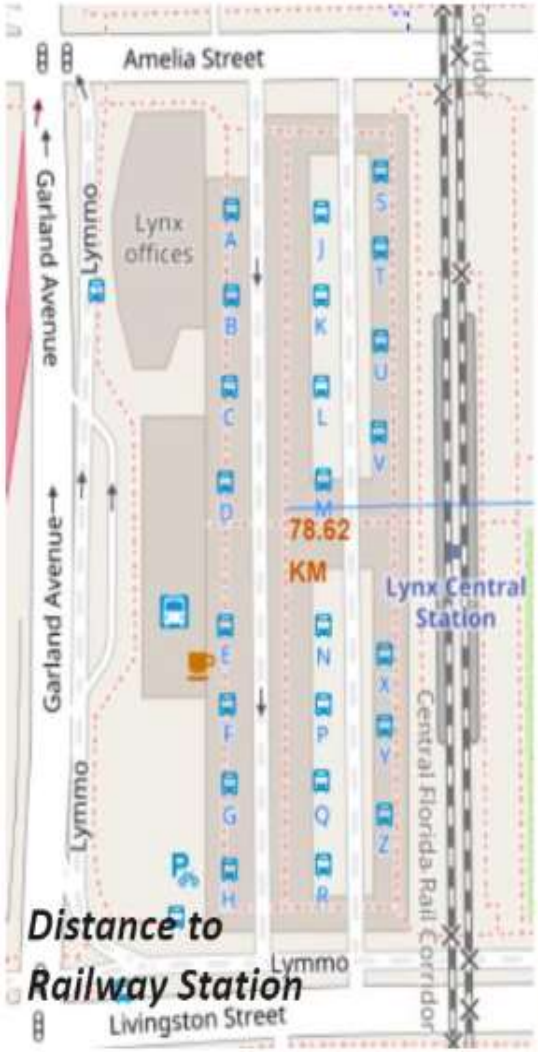
Launch Sites Proximities Analysis

All launch sites global map markers

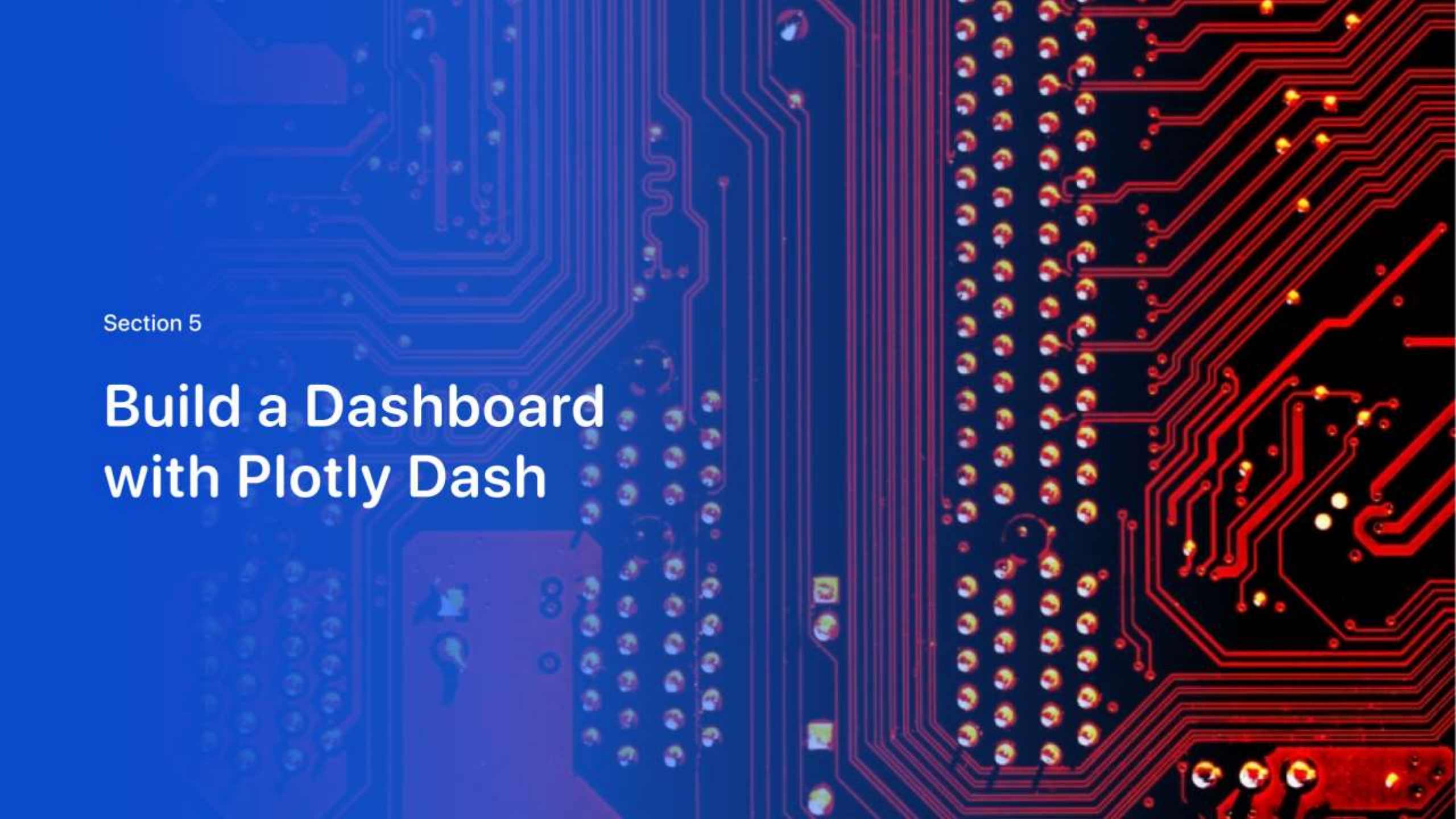
We can see that the SpaceX launch sites are in the United States of America coasts. Florida and California

Markers showing launch sites with color labels

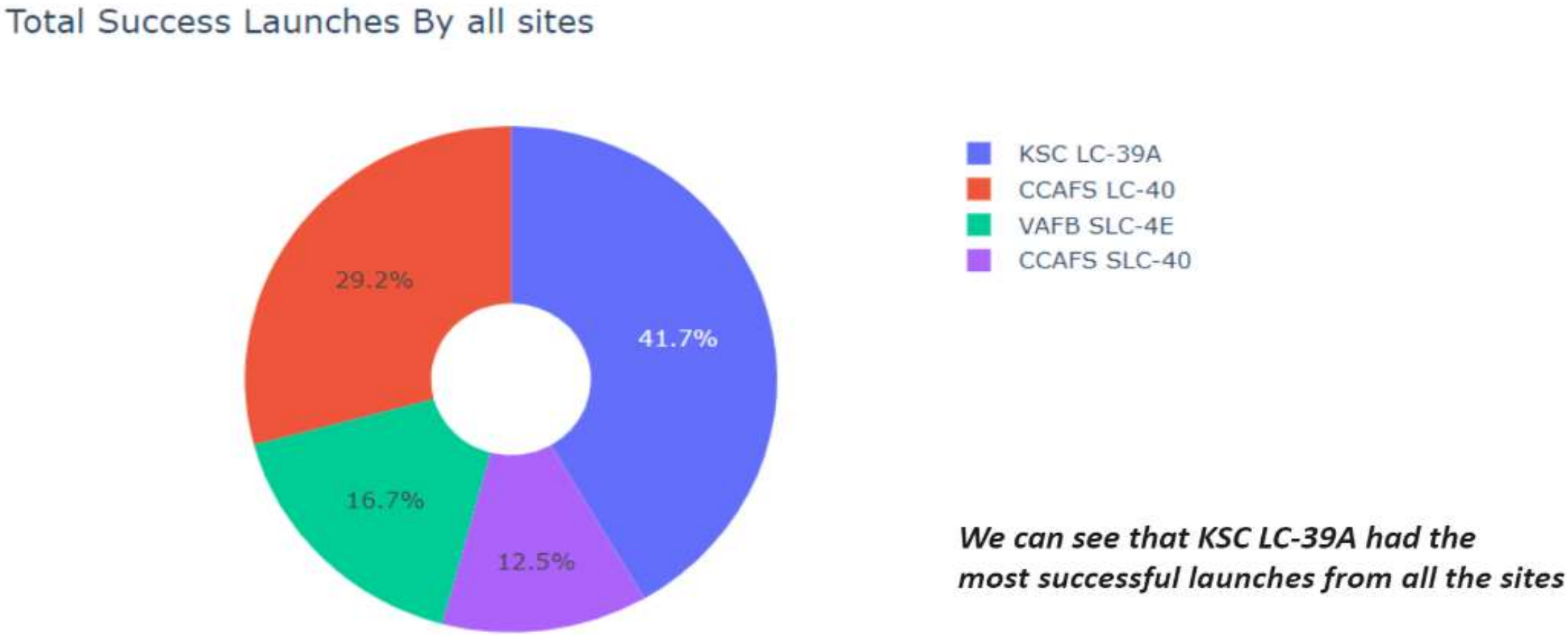
Launch Site distance to landmarks



- 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

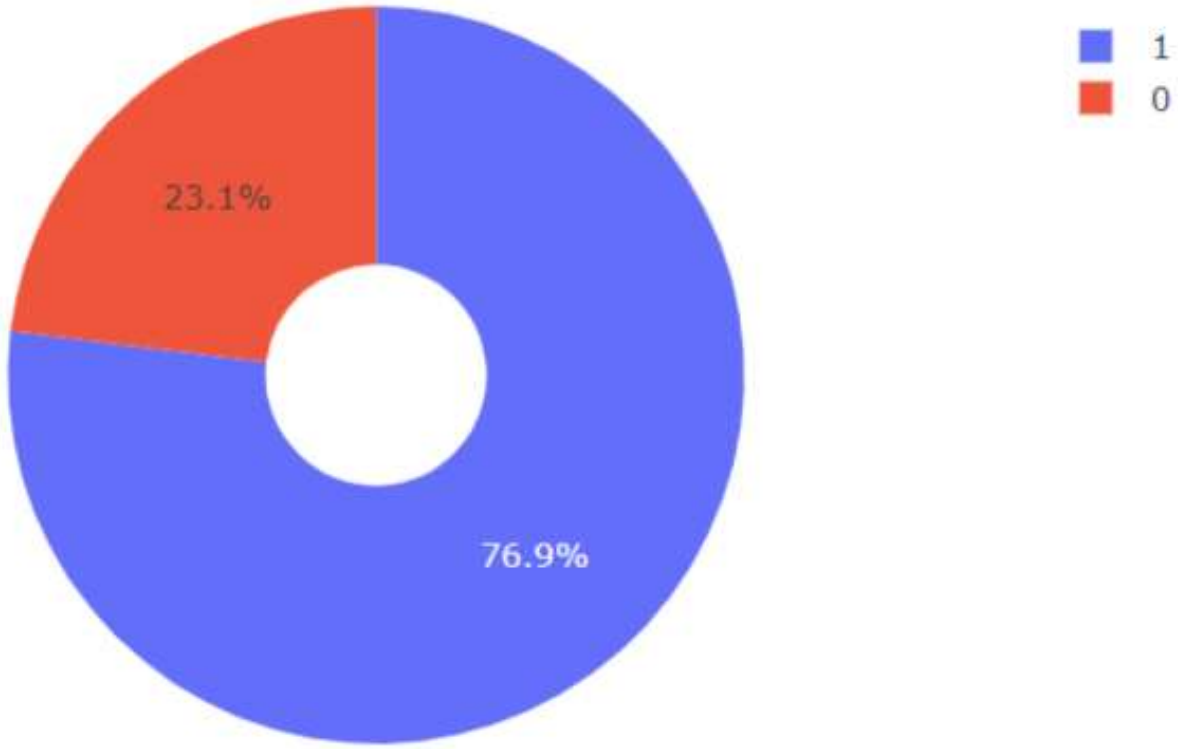

Section 5
Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site



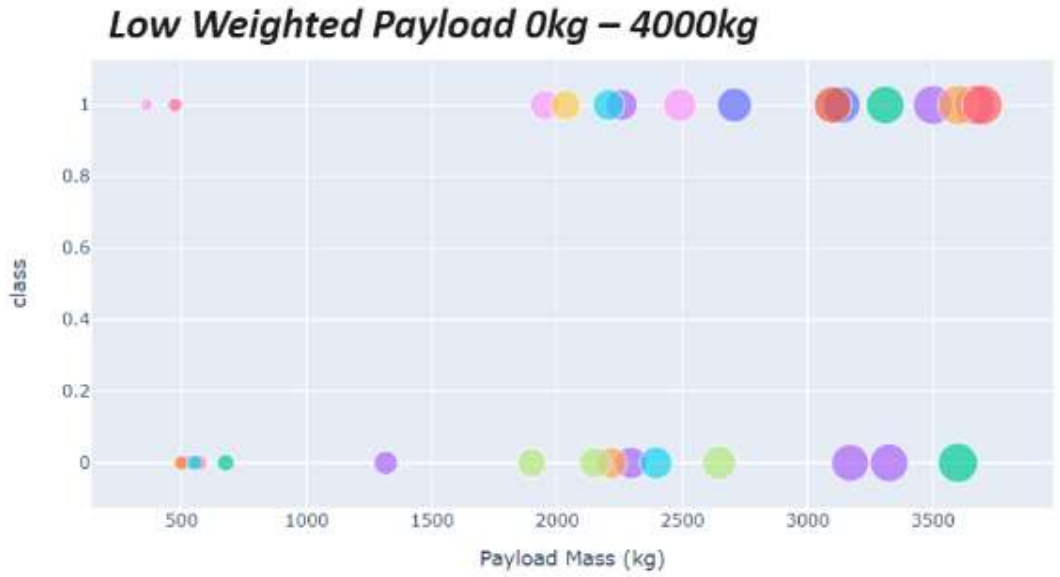
We can see that KSC LC-39A had the most successful launches from all the sites

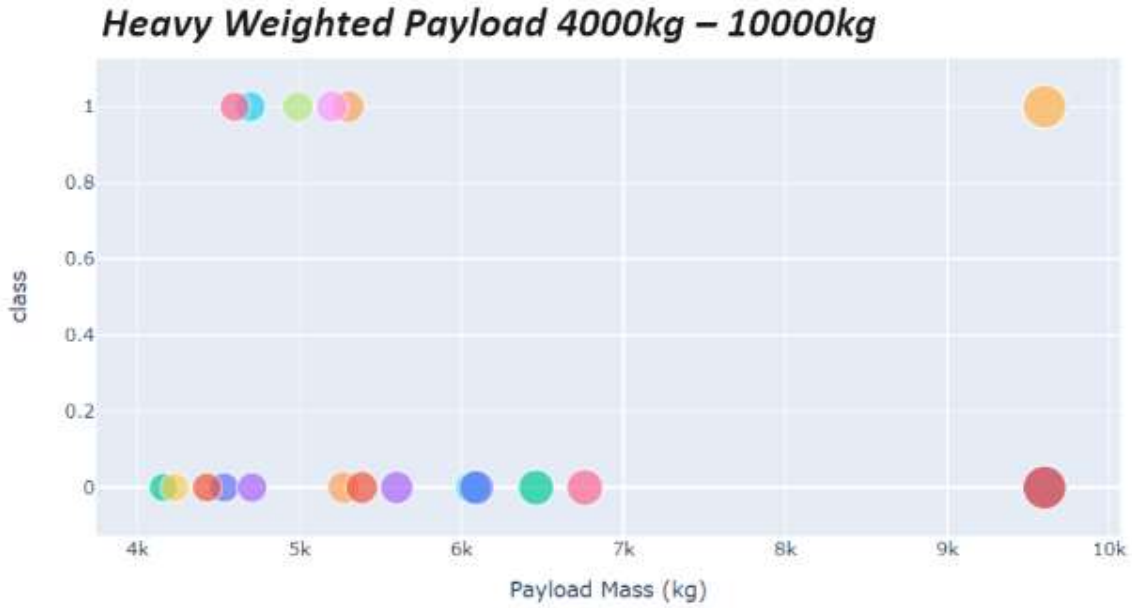
Pie chart showing the Launch site with the highest launch success ratio



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

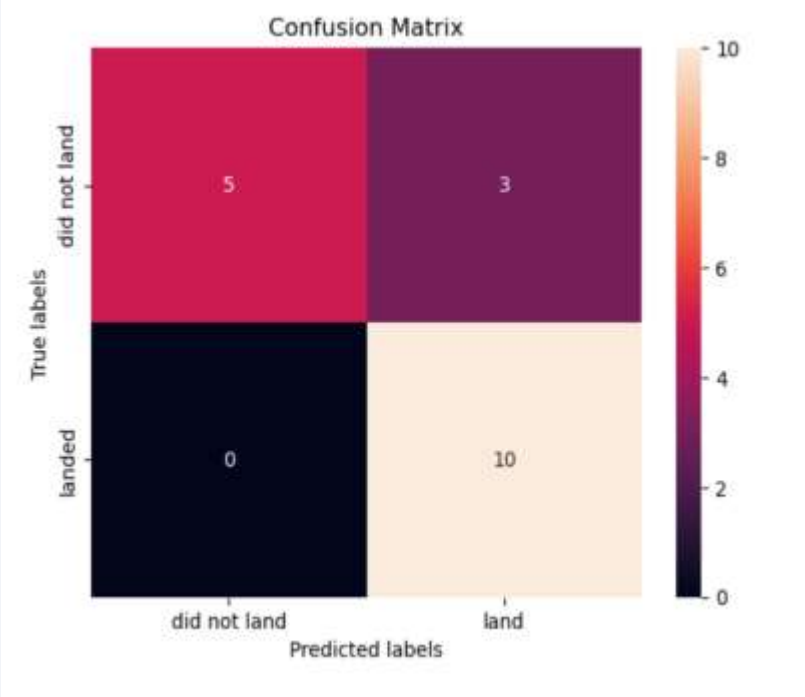
- The decision tree classifier is the model with the highest classification accuracy

```
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 score of ',models[bestalgorithm])

if bestalgorithm == 'KNeighbors':
    print('Best parameter is ', knn_cv.best_params_)
if bestalgorithm == 'DecisionTree':
    print('Best parameter is ', tree_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best parameter is ', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best parameter is ', svm_cv.best_params_)
```

```
Best model is  DecisionTree with score of  0.8803571428571428
Best parameter is  {'criterion': 'gini', 'max_depth': 4, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 2, 'splitte
r': '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!