IBM Developer
SKILLS NETWORK

# Winning Space Race with Data Science

Shivam Kumar
08.07.2023

# Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion

# Executive Summary

- Summary of methodologies
  - Data collection through API and Web Scraping
  - Exploratory Data Analysis (EDA) with SQL and Python Libraries
  - Interactive visual analytics with Folium
  - Machine Learning
- Summary of all results
  - Exploratory Data Analysis results
  - Interactive Visual images
  - Machine Learning predictions

# Introduction

## Project background and context

Space X offers Falcon 9 rocket launches at a significantly lower cost than other providers, primarily due to its ability to reuse the first stage. To be able to compete with Space X, other companies need to know if their first stage will land successfully, and the goal of the project is to create a machine learning pipeline that can predict this outcome.

## Issues to resolve

-Identify the factors that determine if the rocket's first stage will land successfully

-Understand the interaction among features.

-Determine necessary operating conditions that must be in place to ensure a successful landing program.

Section 1

# Methodology

# Methodology

## Executive Summary

- Data collection methodology:
  - Data was collected through SpaceX APIs and Web Scraping methods
- Perform data wrangling
  - Data was cleaned and was made ready to analyze with feature engineering. This process includes dealing with null values and applying one-hot encoding to categorical variables.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Machine Learning models are optimized, tuned and evaluated based on their accuracy performance.

# Data Collection

- Datasets were collected with API requests from SpaceX API. Also, BeautifulSoup library was used for web scraping relevant data.
- For API requests, json formatted data was normalized. For BeautifulSoup web scraping, html parsing was used.
- Data was organized as tabular format.
- Thanks to feature engineering, data was cleaned and became ready for analysis.

# Data Collection - SpaceX API

- To gather the necessary data, we utilized a get request to access the SpaceX API. After collecting the data, we performed some basic data cleaning and formatting, as well as data wrangling to ensure that the data was suitable for analysis.
- The link to the relevant notebook is [https://github.com/brkksp/IBM_Capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb](https://github.com/brkksp/IBM_Capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb)

In [7]:
```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

In [8]:
```
response = requests.get(spacex_url)
```

Check the content of the response

In [10]:
```
#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.

### Task 1: 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 [22]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_ap
```

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

In [23]:
```
response.status_code
```

Out[23]: 200

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

In [36]:
```
# Use json_normalize meethod to convert the json result into a dataframe
response = requests.get(static_json_url).json()
data = pd.json_normalize(response)
```

Using the dataframe `data` print the first 5 rows

In [37]:
```
# Get the head of the dataframe
data.head()
```

# Data Collection - Scraping

- We utilized web scraping with BeautifulSoup to obtain Falcon 9 launch records. We parsed the information in the table and converted it into a pandas dataframe, which we then used for further analysis.

- The relevant notebook link is: https://github.com/brkksp/IBM_Capstone/blob/main/jupyter-labs-webscraping.ipynb

In [92]:
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

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.

In [93]:
```
# use requests.get() method with the provided static_url
response = requests.get(static_url)
# assign the response to a object
```

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

In [94]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.content,'html.parser')
```

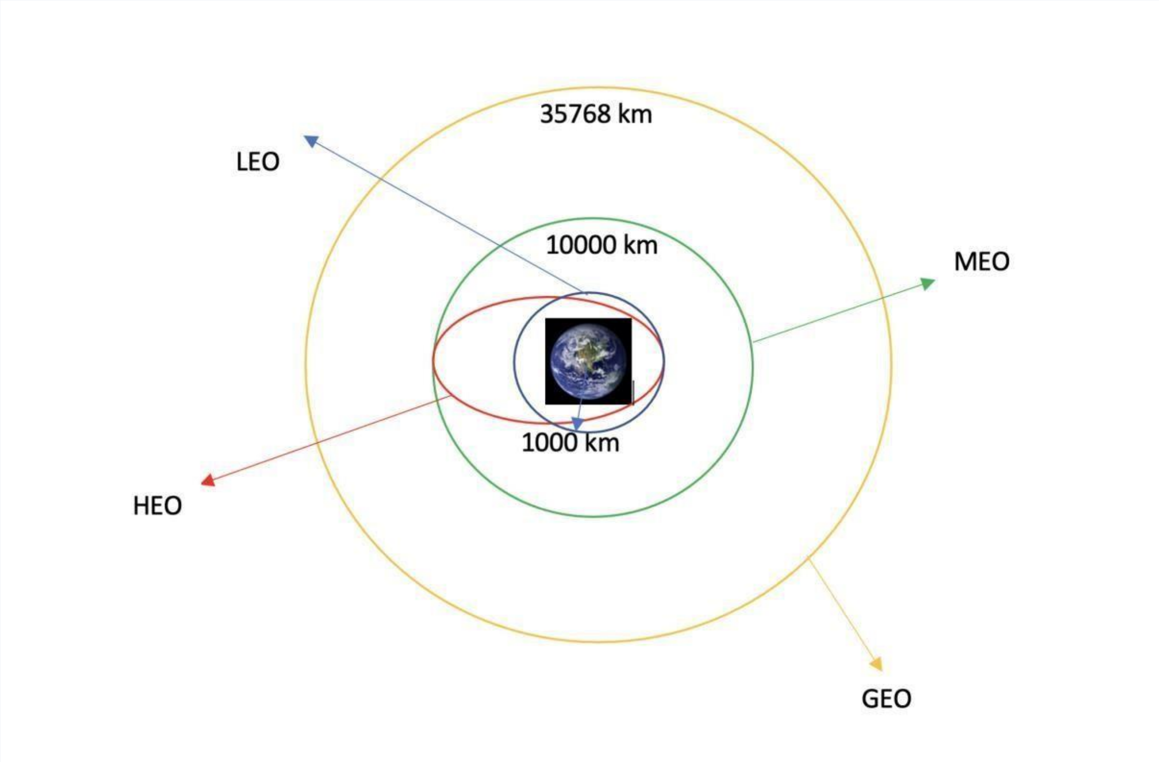
Print the page title to verify if the `BeautifulSoup` object was created properly

In [95]:
```
# Use soup.title attribute
soup.title.string
```

Out[95]: 'List of Falcon 9 and Falcon Heavy launches - Wikipedia'

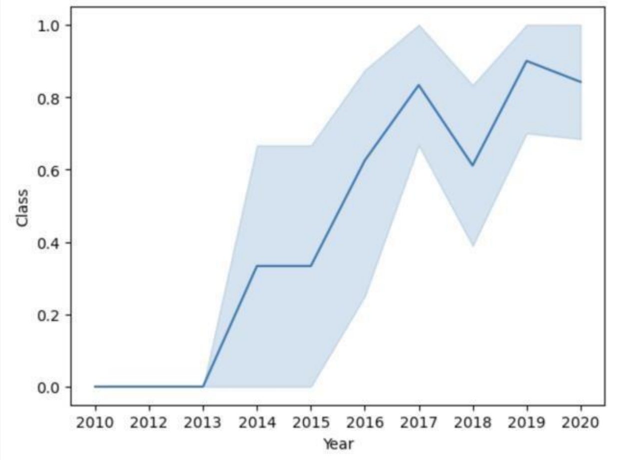
# Data Wrangling

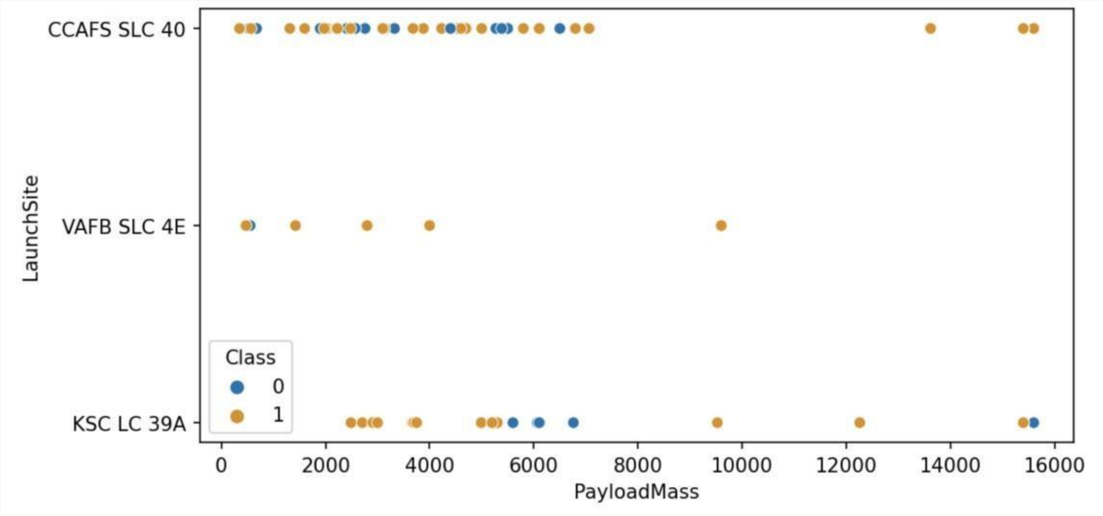
- Exploratory data analysis was conducted, and appropriate training labels were determined. Specifically, the number of launches occurred at each site was calculated, as well as the frequency and occurrence of each orbit.
- The relevant notebook link is: https://github.com/brkksp/IBM_Capstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb



# EDA with Data Visualization

- We explored the relation between launch sites and other features. We aimed to seek any relation in terms of success/failure of launches.
- We also observed success rate by year.
- The relevant notebook link is: https://github.com/brkksp/IBM_Capstone/blob/main/jupyter-labs-eda-dataviz.ipynb



# EDA with SQL

- Utilizing SQL queries, we performed exploratory data analysis on the data to gain further insight.
- We extracted information such as the unique names of launch sites used in space missions, the total payload mass carried by boosters launched by NASA's CRS program, the average payload mass carried by the F9 v1.1 booster version, and the total number of successful and failed mission outcomes.
- The relevant notebook link is: https://github.com/brkksp/IBM_Capstone/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

# Build an Interactive Map with Folium

- Markers, circles, lines are added to the Interactive Folium Map.
- These added features enabled us to observe the success rate of launches.
- Launch sites were investigated in terms of their proximity to coastlines, railways and highways.
- The relevant notebook link is: https://github.com/brkksp/IBM_Capstone/blob/main/IBM-DS0321EN-SkillsNetwork_labs_module_3_lab_jupyter_launch_site_location.jupyterlite.ipynb

# Build a Dashboard with Plotly Dash

- We used Plotly Dash to build interactive visuals.
- We included scatterplot to observe the relation between Outcome and Payload Mass (kg) in terms of distinct booster versions.
- We included pie charts to observe total launches per launch sites.
- The relevant notebook link is: https://github.com/brkksp/IBM_Capstone/blob/main/dash.py

# Predictive Analysis (Classification)

- We used Scikit-Learn Library for predictive analysis.
- We split the dataset as train-test-split. Then we tried distinct predictive models.
- For each predictive model, we used Grid Search Cross Validation to choose best parameters.
- Then we plotted confusion matrix and calculated the accuracy per model.
- We then opted the most accurate model.
- The relevant notebook link is: https://github.com/brkksp/IBM_Capstone/blob/main/IBM-DS0321EN-SkillsNetwork_labs_module_4_SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

# Results

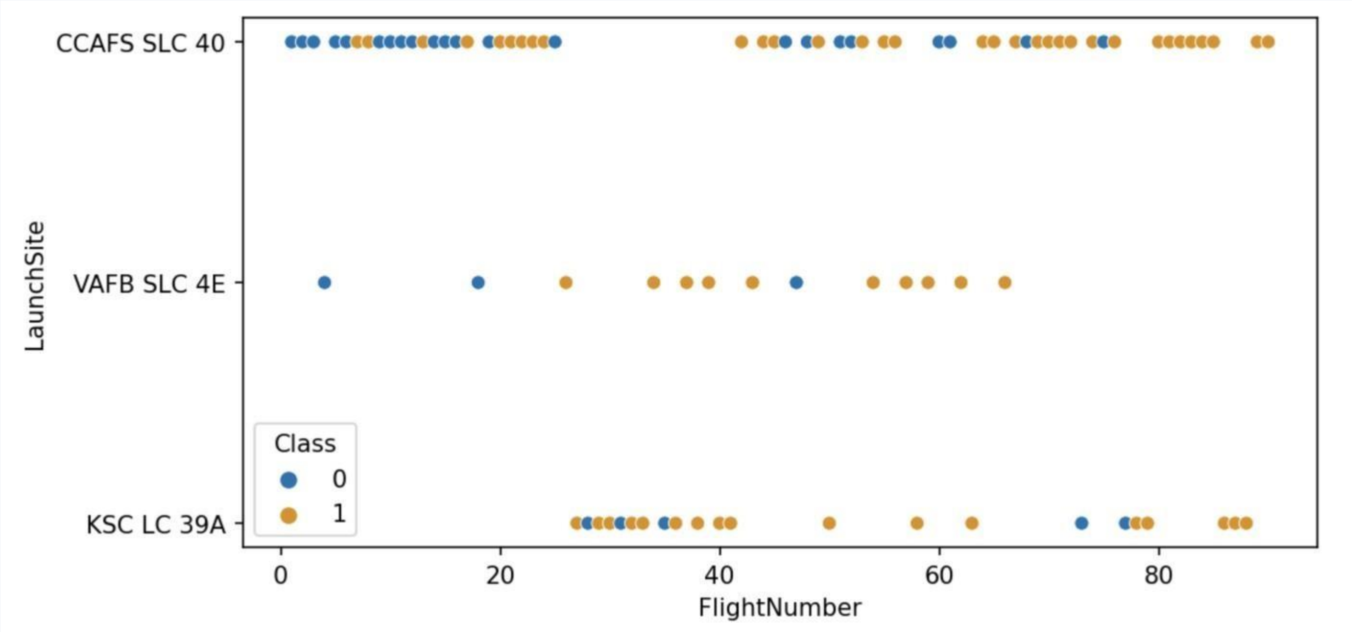
- After cleaning and organizing the data, we observed that best fitted model for our data is Decision Tree.

Section 2

# Insights drawn from EDA

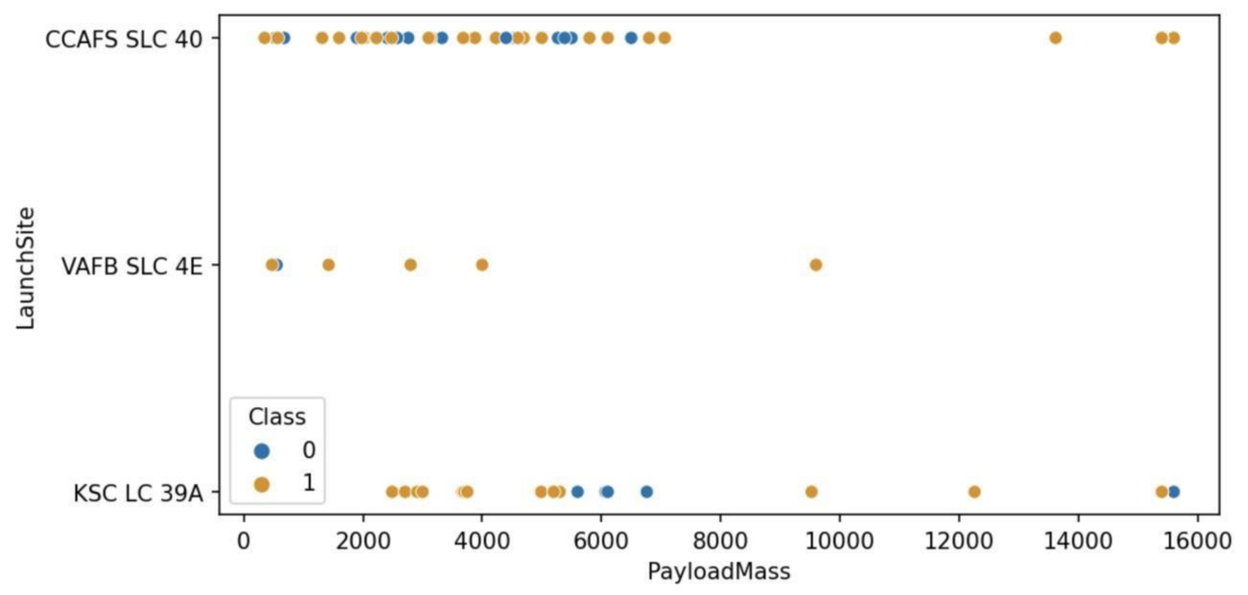
# Flight Number vs. Launch Site

- Based on the plot, it can be observed that as the number of flights increases at a given launch site, the likelihood of success also tends to increase.



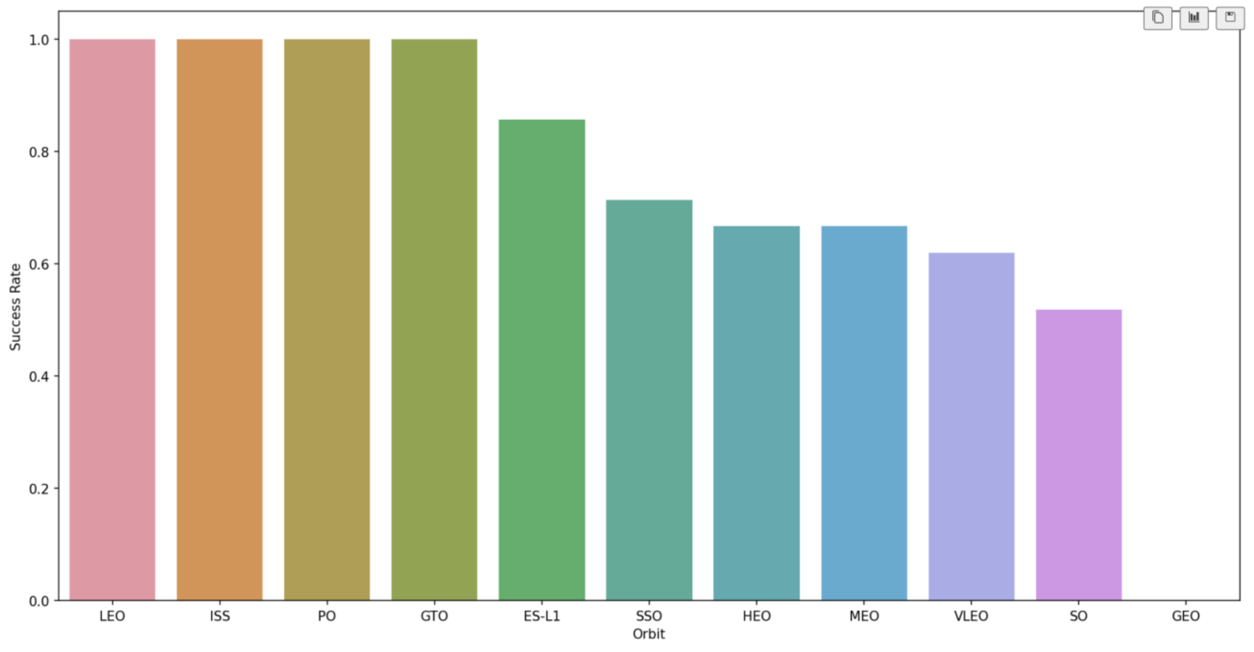
# Payload vs. Launch Site

- A correlation can be observed between the success rate of the rocket at launch site CCAFS SLC 40 and the mass of its payload, where a higher payload mass is associated with a higher success rate.



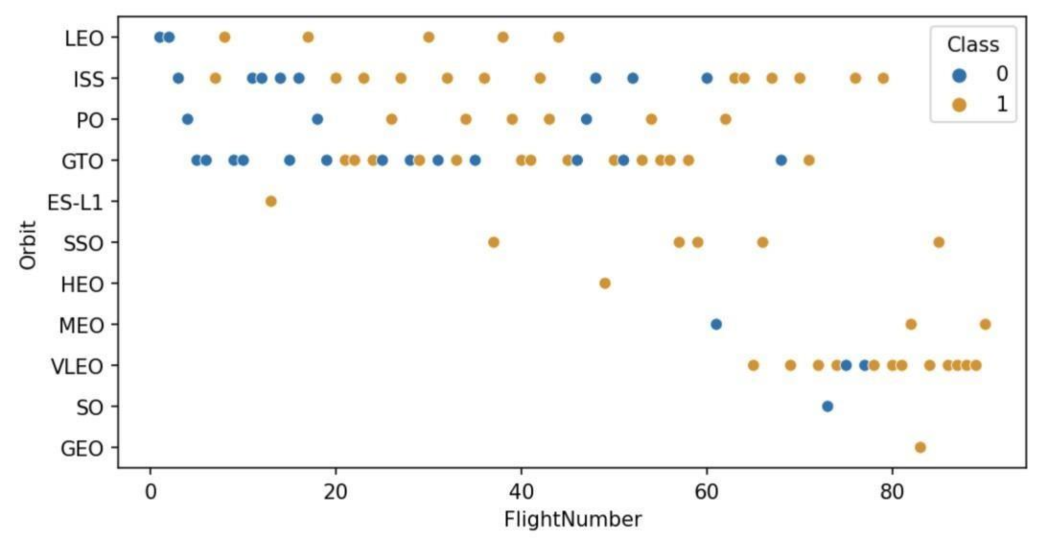
# Success Rate vs. Orbit Type

- Based on the plot, it can be inferred that LEO, ISS, PO, GTO, and ES-L1 exhibited the highest success rates.



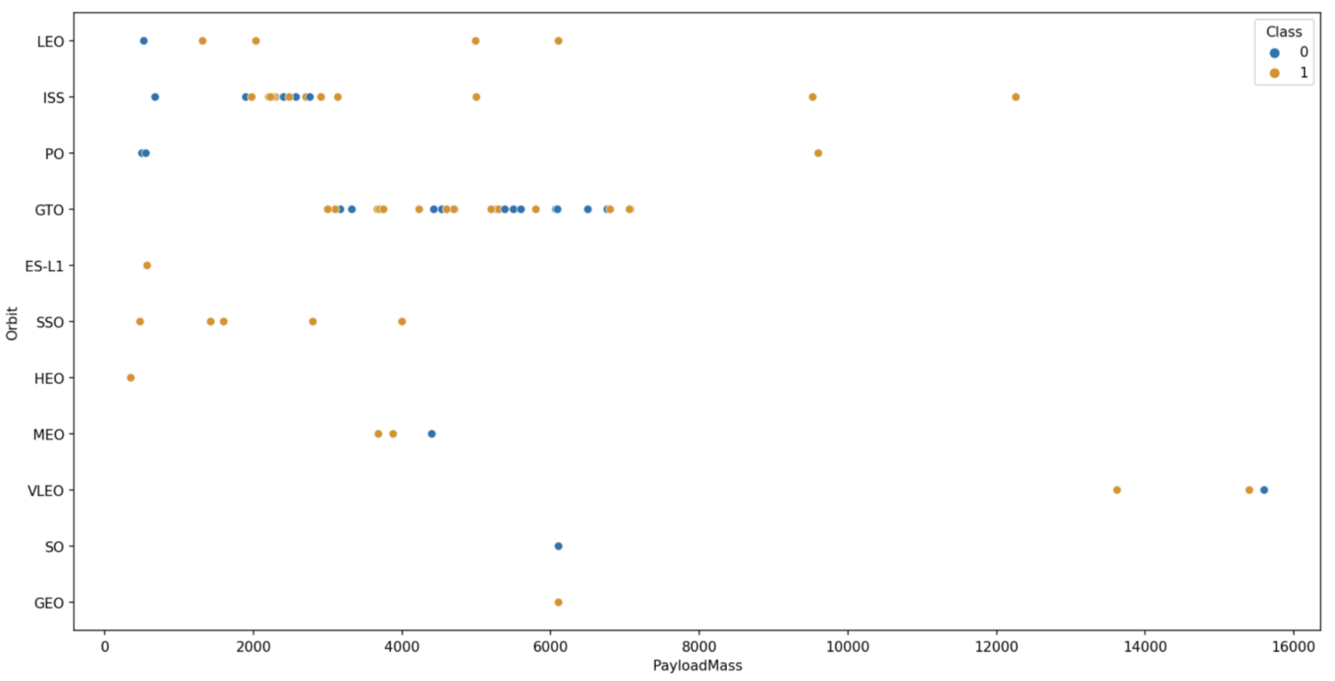
# Flight Number vs. Orbit Type

- The following plot illustrates the correlation between Flight Number and Orbit type. It can be observed that in the LEO orbit, the success rate appears to be linked to the number of flights, while in the GTO orbit, there seems to be no such association between the success rate and flight number.



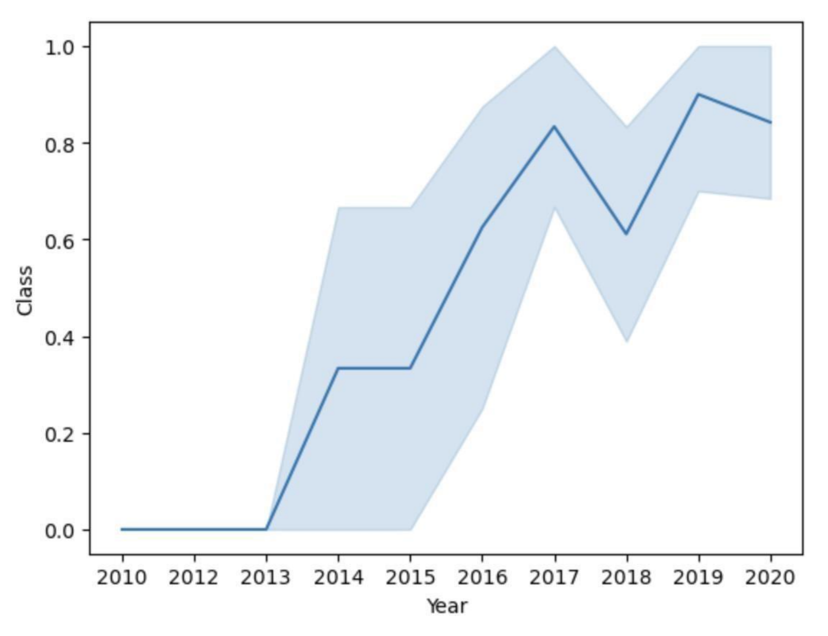
# Payload vs. Orbit Type

- It can be observed that for orbits such as PO, LEO, and ISS, a higher success rate in terms of landing is associated with heavy payloads.



# Launch Success Yearly Trend

- Based on the plot, it can be observed that the success rate has consistently increased from 2013 to 2020.



# All Launch Site Names

- We used the key word DISTINCT to show only unique launch sites from the SpaceX data.

```
%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 was utilized to retrieve 5 records from a dataset where the launch sites begin with 'CCA'.

```
%sql SELECT * FROM SPACEXTBL WHERE Launch_Site LIKE 'CCA%' LIMIT 5;
```

Python

```
* 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 |

# Total Payload Mass

- By executing the query provided below, we were able to calculate that the total payload carried by NASA's boosters is 45596.

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS Total_Payload_Mass FROM SPACEXTBL WHERE Customer LIKE 'NASA (CRS)';
```

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

| Total_Payload_Mass |
|---|
| 45596 |

# Average Payload Mass by F9 v1.1

- By performing the following query, it was determined that the average payload mass carried by booster version F9 v1.1 is 2928.4.

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS Average_Payload_Mass FROM SPACEXTBL WHERE Booster_Version LIKE 'F9 v1.1'
```

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

| Average_Payload_Mass |
|---|
| 2928.4 |

# First Successful Ground Landing Date

- Based on our observations, the date on which the first successful landing occurred on a ground pad was December 22nd, 2015.

```
%sql SELECT MIN(Date) AS Min_Date FROM SPACEXTBL WHERE Landing_Outcome LIKE 'Success (ground pad)';
✓ 0.0s
* sqlite:///my_data1.db
Done.
```

| Min_Date |
|---|
| 22-12-2015 |

# Successful Drone Ship Landing with Payload between 4000 and 6000

- We deployed the WHERE clause to narrow down the dataset to boosters that had accomplished a successful landing on a drone ship. Subsequently, we applied an AND condition to further refine the results and only include boosters that had a payload mass greater than 4000 but less than 6000 at the time of the successful landing.

```
%sql SELECT * FROM SPACEXTBL WHERE Landing_Outcome LIKE '%Success (drone ship)%' AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000;
```

✓ 0.0s

```
* 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

- UNION method is used to merge both results in the same column. It can be observed that total number of:
  - Success outcome is 100
  - Failure outcome is 1

```
%sql SELECT COUNT(*) FROM SPACEXTBL WHERE Mission_Outcome LIKE '%Success%' UNION SELECT COUNT(*) Failure FROM SPACEXTBL WHERE Mission_Outcome LIKE '%Failure%';
```

✓ 0.0s Python

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

| COUNT(*) |
|---|
| 1 |
| 100 |

# Boosters Carried Maximum Payload

- Using a subquery within the WHERE clause and the MAX() function, we were able to identify the booster that carried the maximum payload.

```
%sql SELECT * FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```

✓ 0.0s Python

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

| Date | Time (UTC) | Booster_Version | Launch_Site | Payload | PAYLOAD_MASS__KG_ | Orbit | Customer | Mission_Outcome | Landing_Outcome |
|---|---|---|---|---|---|---|---|---|---|
| 11-11-2019 | 14:56:00 | F9 B5 B1048.4 | CCAFS SLC-40 | Starlink 1 v1.0, SpaceX CRS-19 | 15600 | LEO | SpaceX | Success | Success |
| 07-01-2020 | 02:33:00 | F9 B5 B1049.4 | CCAFS SLC-40 | Starlink 2 v1.0, Crew Dragon in-flight abort test | 15600 | LEO | SpaceX | Success | Success |
| 29-01-2020 | 14:07:00 | F9 B5 B1051.3 | CCAFS SLC-40 | Starlink 3 v1.0, Starlink 4 v1.0 | 15600 | LEO | SpaceX | Success | Success |
| 17-02-2020 | 15:05:00 | F9 B5 B1056.4 | CCAFS SLC-40 | Starlink 4 v1.0, SpaceX CRS-20 | 15600 | LEO | SpaceX | Success | Failure |
| 18-03-2020 | 12:16:00 | F9 B5 B1048.5 | KSC LC-39A | Starlink 5 v1.0, Starlink 6 v1.0 | 15600 | LEO | SpaceX | Success | Failure |
| 22-04-2020 | 19:30:00 | F9 B5 B1051.4 | KSC LC-39A | Starlink 6 v1.0, Crew Dragon Demo-2 | 15600 | LEO | SpaceX | Success | Success |
| 04-06-2020 | 01:25:00 | F9 B5 B1049.5 | CCAFS SLC-40 | Starlink 7 v1.0, Starlink 8 v1.0 | 15600 | LEO | SpaceX, Planet Labs | Success | Success |
| 03-09-2020 | 12:46:14 | F9 B5 B1060.2 | KSC LC-39A | Starlink 11 v1.0, Starlink 12 v1.0 | 15600 | LEO | SpaceX | Success | Success |
| 06-10-2020 | 11:29:34 | F9 B5 B1058.3 | KSC LC-39A | Starlink 12 v1.0, Starlink 13 v1.0 | 15600 | LEO | SpaceX | Success | Success |
| 18-10-2020 | 12:25:57 | F9 B5 B1051.6 | KSC LC-39A | Starlink 13 v1.0, Starlink 14 v1.0 | 15600 | LEO | SpaceX | Success | Success |
| 24-10-2020 | 15:31:34 | F9 B5 B1060.3 | CCAFS SLC-40 | Starlink 14 v1.0, GPS III-04 | 15600 | LEO | SpaceX | Success | Success |
| 25-11-2020 | 02:13:00 | F9 B5 B1049.7 | CCAFS SLC-40 | Starlink 15 v1.0, SpaceX CRS-21 | 15600 | LEO | SpaceX | Success | Success |

# 2015 Launch Records

- To filter the dataset for failed landing outcomes on a drone ship, along with their corresponding booster versions and launch site names for the year 2015, we utilized a combination of the WHERE clause, LIKE operator, AND condition.

```
%sql SELECT substr(Date,4,2) AS Month, Landing_Outcome, Booster_Version, Launch_Site
FROM SPACEXTBL
WHERE Landing_Outcome LIKE '%Failure (drone ship)%' AND substr(Date,7,4) = '2015';
```

✓ 0.0s

```
* 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

- We selected Landing outcomes and the COUNT of landing outcomes. We then used the WHERE clause to filter the data for landing outcomes that fell between the dates 2010-06-04 to 2010-03-20.
- Next, we applied the GROUP BY clause to group the landing outcomes and used the ORDER BY clause to arrange the grouped landing outcomes in descending order.

```
%sql SELECT Landing_Outcome,COUNT(*) FROM SPACEXTBL WHERE Date between '04-06-2010' and '20-03-2017' GROUP BY Landing_Outcome ORDER BY COUNT(*) DESC;
```

✓ 0.0s

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

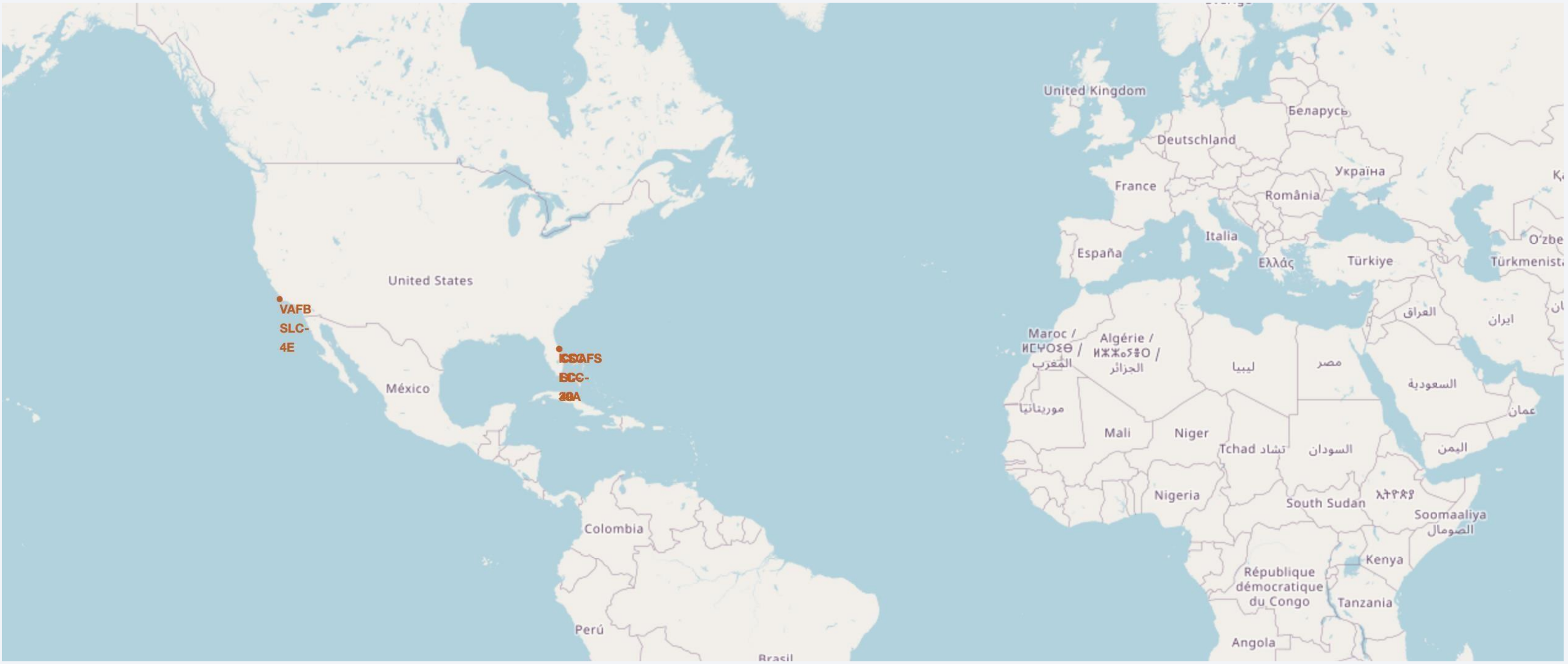
| Landing_Outcome | COUNT(*) |
|---|---|
| Success | 20 |
| No attempt | 10 |
| Success (drone ship) | 8 |
| Success (ground pad) | 6 |
| Failure (drone ship) | 4 |
| Failure | 3 |
| Controlled (ocean) | 3 |
| Failure (parachute) | 2 |
| No attempt | 1 |

Section 3

# Launch Sites Proximities Analysis

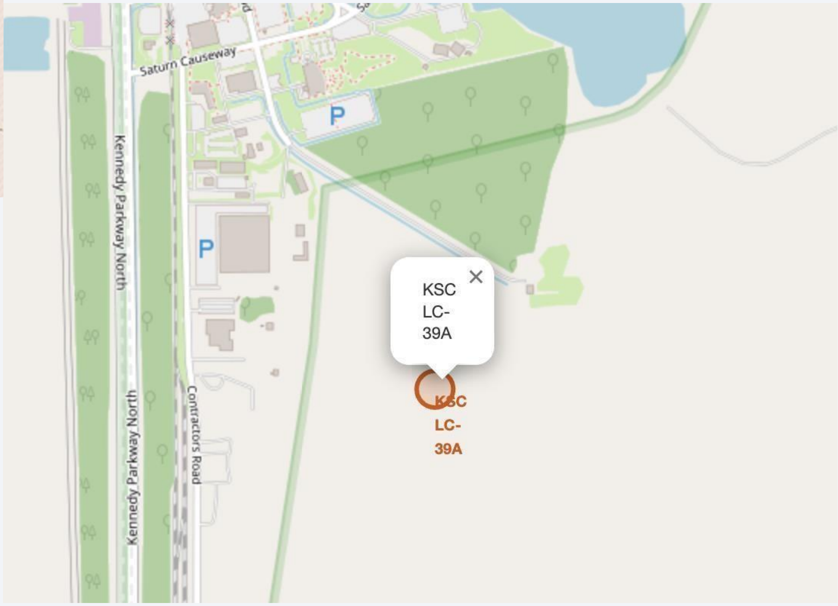
# Launch Sites of SpaceX

- As can be seen from the map below, all launch sites of SpaceX are in the United States, mostly around two locations.



# Launch Sites with Labels


- Green is for success
- Red is for failure.

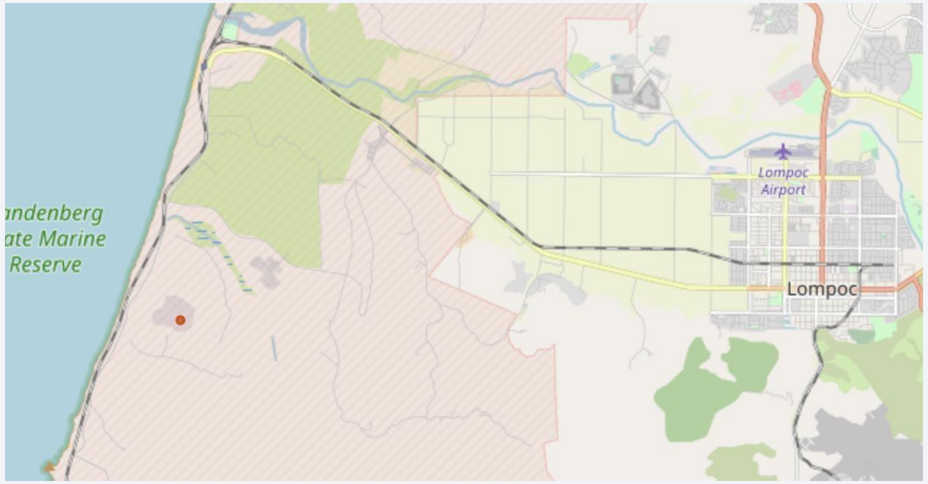



# Launch Site Distance to Landmarks

- It can be observed that eastern launch sites are quite close to landmarks meanwhile western one is distant.

Eastern


Western

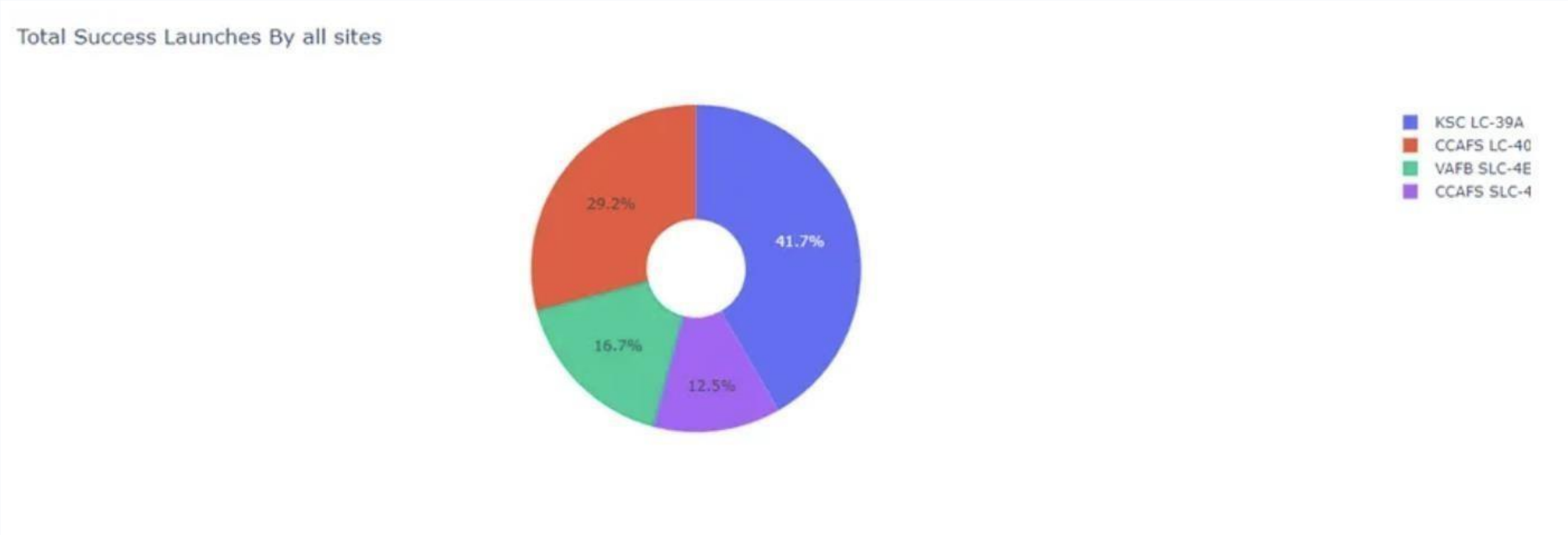


Section 4

# Build a Dashboard with Plotly Dash

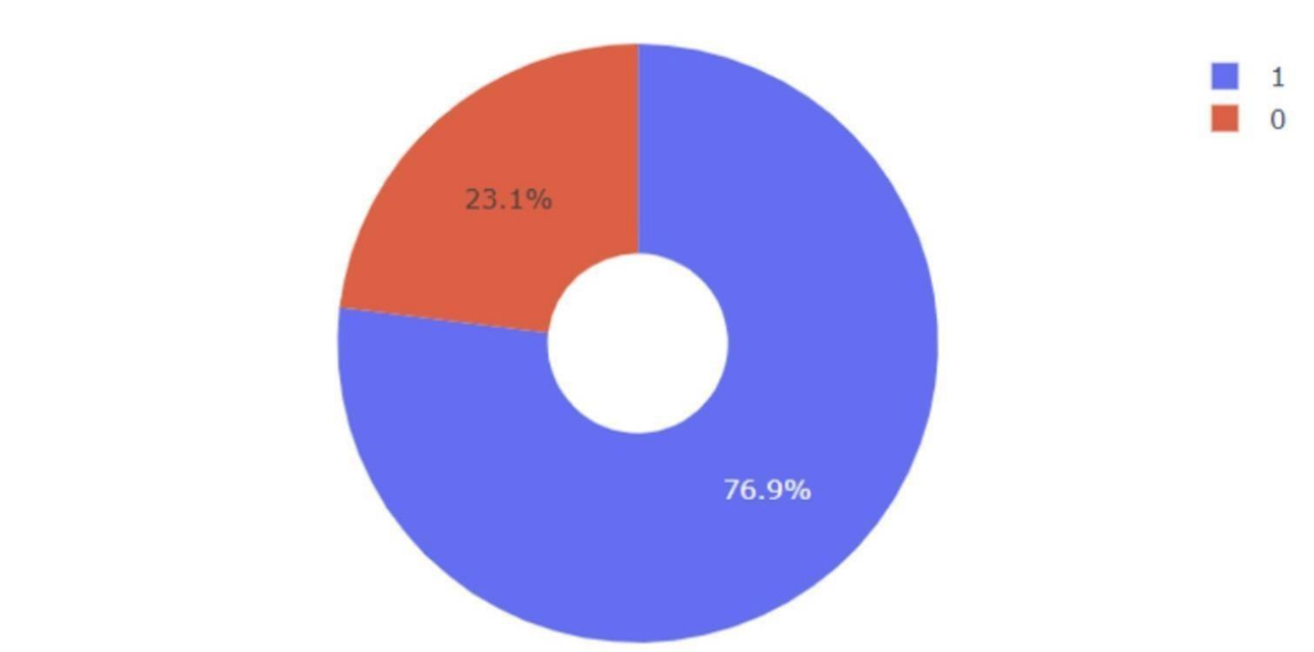
# Total Success Launches by All Sites

- As can be seen below, KSC LC-39A is the most successful launch of all launches.



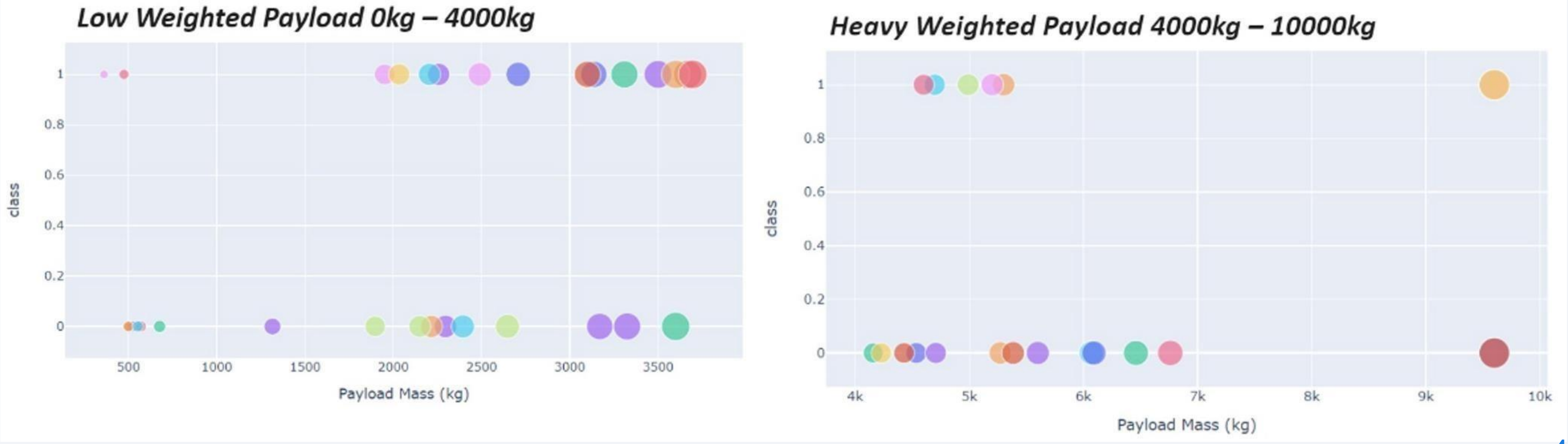
# KSC LC-39A Success Rate

- As can be seen from the image, success rate of KSC LC-39A is 76.9%.



# Relationship between Payload and Launch Outcome across all launch sites

- One could observe that the success rates are higher for payloads with lower weight compared to those with heavier weight.



Section 5

# Predictive Analysis (Classification)

# Classification Accuracy

- The decision tree classifier is considered to have the highest classification accuracy among the models used.

```
methods = {'KNN':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SVM':svm_cv.best_score_}
```

```
Best_Method = max(algorithms,key=algorithms.get)
Best_Method
```

```
'DecisionTree'
```

```
tree_cv = GridSearchCV(estimator=tree, param_grid=parameters, cv=10)
tree_cv.fit(X_train,Y_train)
```

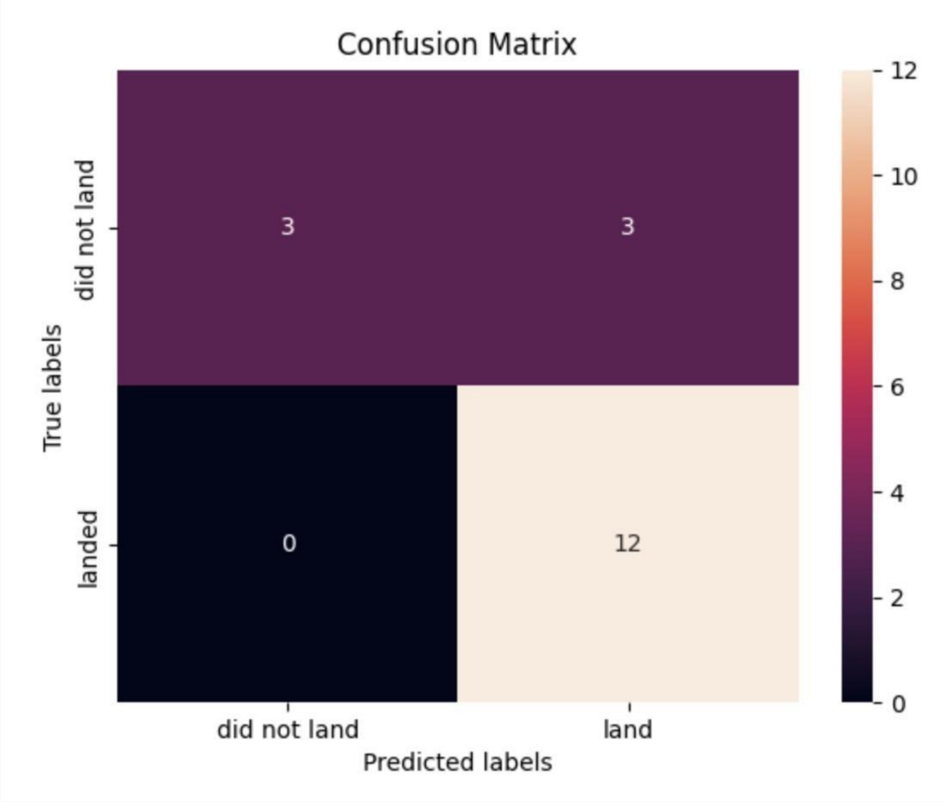
```
GridSearchCV(cv=10, estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [2, 4, 6, 8, 10, 12, 14, 16, 18],
                         'max_features': ['auto', 'sqrt'],
                         'min_samples_leaf': [1, 2, 4],
                         'min_samples_split': [2, 5, 10],
                         'splitter': ['best', 'random']})
```

```
print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
print("accuracy :",tree_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 6
accuracy : 0.9035714285714287
```

# Confusion Matrix

- Based on the confusion matrix of the decision tree classifier, it can be inferred that the classifier is capable of accurately distinguishing between different classes. However, the major issue appears to be the occurrence of false positives, which implies that the classifier sometimes labels unsuccessful landings as successful ones.



# Conclusions

Based on the analysis, we can draw the following conclusions:

- There is a positive correlation between the flight amount at a launch site and the success rate at the site.
- Launch success rate has been increasing steadily from 2013 to 2020.
- KSC LC-39A is the launch site with the highest number of successful launches.
- The orbits ES-L1, GEO, HEO, SSO, VLEO have the highest success rates.
- The decision tree classifier is the most suitable machine learning algorithm for this task based on its performance.

Overall, these findings provide useful insights into the factors that contribute to launch success and can be used to improve future launches.

# Thank you!