IBM Developer
SKILLS NETWORK

# Winning Space Race with Data Science


Ujjwal Tyagi

4th February 2024

# 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, 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 https://github.com/ujjwaltyagi2000/data-science-captsone-ibm/blob/main/spacex-data-collection-api.ipynb

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

[36]
```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

[37]
```
response = requests.get(spacex_url)
```

Check the content of the response

[38]
```
print(response.content)
```

b'[{"fairings":{"reused":false,"recovery_attempt":false,"recovered":false,"ships":[]},"link

You should see the response contains massive information about SpaceX launches. Next, let's try

# Data Collection - Scraping

- We applied web scrapping to webscrap Falcon 9 launch records with BeautifulSoup
- We parsed the table and converted it into a pandas dataframe.
- The link to the notebook is https://github.com/ujjwaltyagi2000/data-science-captsone-ibm/blob/main/webscraping.ipynb

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

```
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url)
print(response.status_code)
```

[4] ✓ 2.1s

```
200
```

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

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

[5] ✓ 1.8s

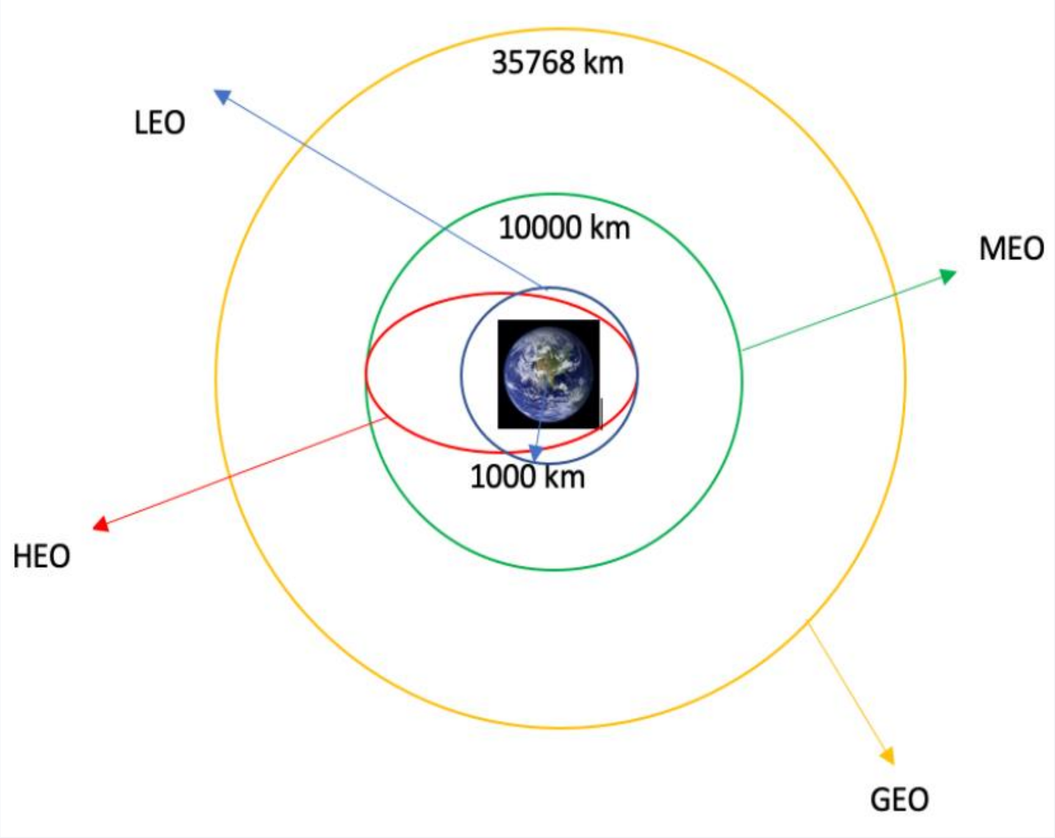
Print the page title to verify if the `BeautifulSoup` object was created properly

```
# Use soup.title attribute
print(soup.title)
```

[6] ✓ 0.0s

```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

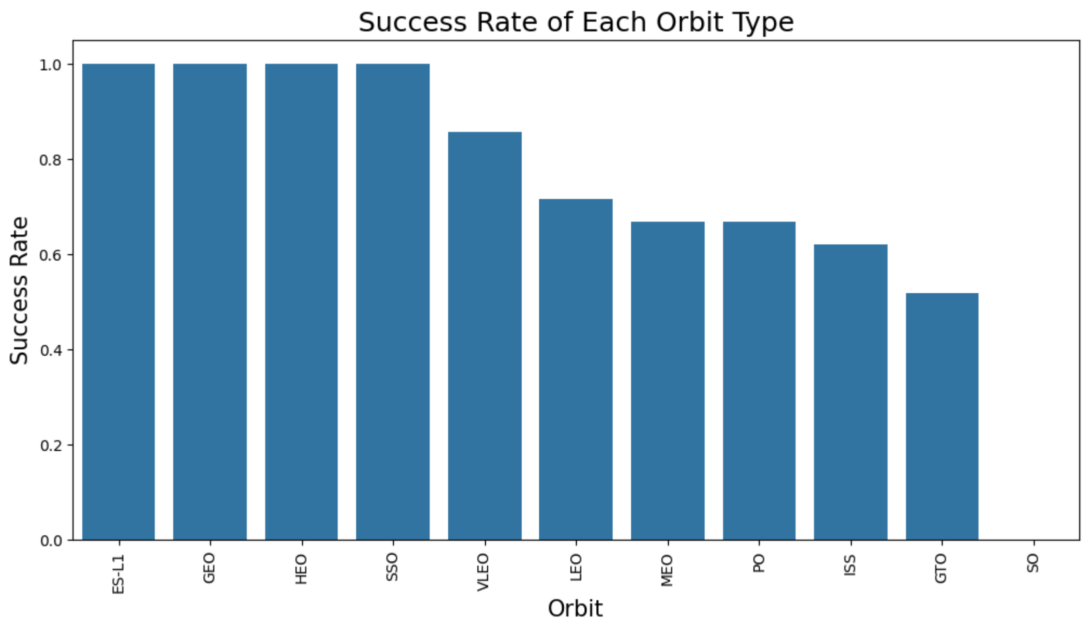
# Data Wrangling

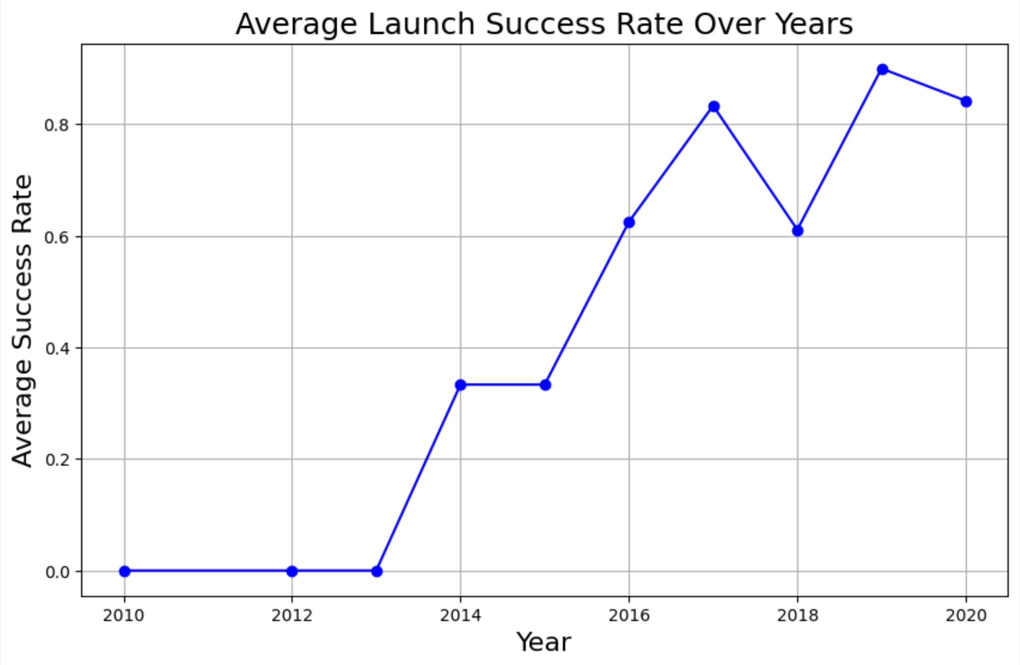



- We performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits
- We created landing outcome label from outcome column and exported the results to csv.
- The link to the notebook is https://github.com/ujjwaltyagi2000/data-science-captsone-ibm/blob/main/data-wrangling.ipynb

# EDA with Data Visualization

- We explored the data by visualizing the relationship between flight number and launch Site, payload and launch site, success rate of each orbit type, flight number and orbit type, the launch success yearly trend.




- The link to the notebook is https://github.com/ujjwaltyagi2000/data-science-captsone-ibm/blob/main/eda.ipynb

# EDA with SQL

- We loaded the SpaceX dataset into a sqllite database without leaving the jupyter notebook.
- We applied EDA with SQL to get insight from the data. We wrote queries to find out for instance:
  - The names of unique launch sites in the space mission.
  - The total payload mass carried by boosters launched by NASA (CRS)
  - The average payload mass carried by booster version F9 v1.1
  - The total number of successful and failure mission outcomes
  - The failed landing outcomes in drone ship, their booster version and launch site names.
- The link to the notebook is https://github.com/ujjwaltyagi2000/data-science-captsone-ibm/blob/main/sqllite_queries.ipynb

# Build an Interactive Map with Folium

- We marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- We assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.
- Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate.
- We calculated the distances between a launch site to its proximities. We answered some question for instance:
  - Are launch sites near railways, highways and coastlines.
  - Do launch sites keep certain distance away from cities.

# Build a Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly dash
- We plotted pie charts showing the total launches by a certain sites
- We plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.
- The link to the notebook is https://github.com/ujjwaltyagi2000/data-science-captsone-ibm/blob/main/dash_app.py

# Predictive Analysis (Classification)

- We loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- We built different machine learning models and tune different hyperparameters using GridSearchCV.
- We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- We found the best performing classification model.
- The link to the notebook is https://github.com/ujjwaltyagi2000/data-science-captsone-ibm/blob/main/SpaceX_Machine_Learning_Prediction.ipynb

# Results

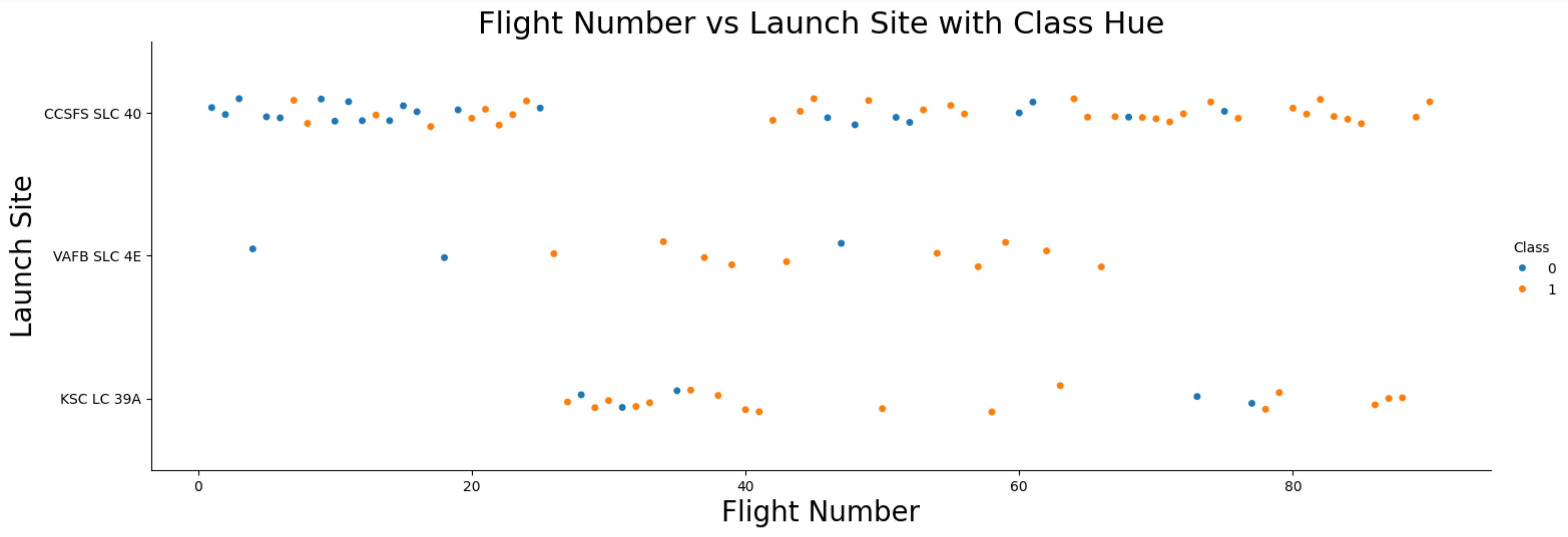
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

# Insights drawn from EDA

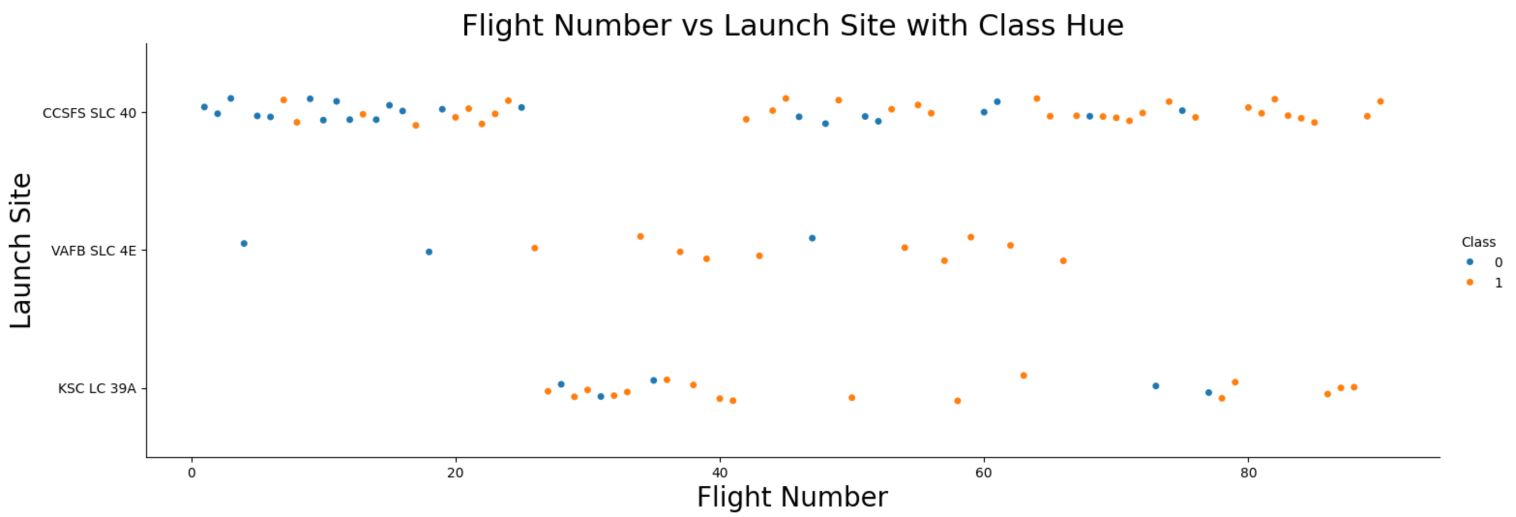
# Flight Number vs. Launch Site

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



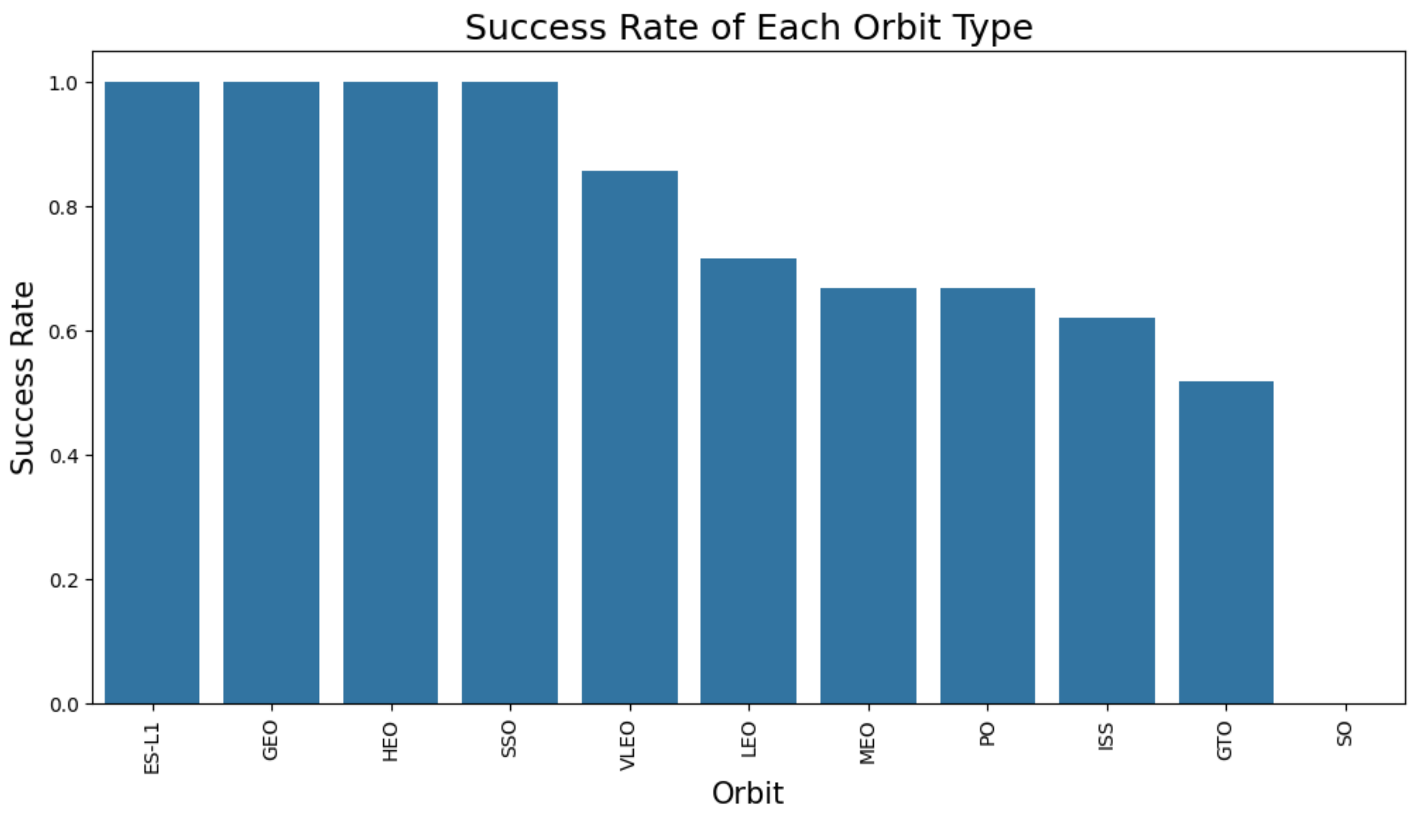
# Payload vs. Launch Site

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



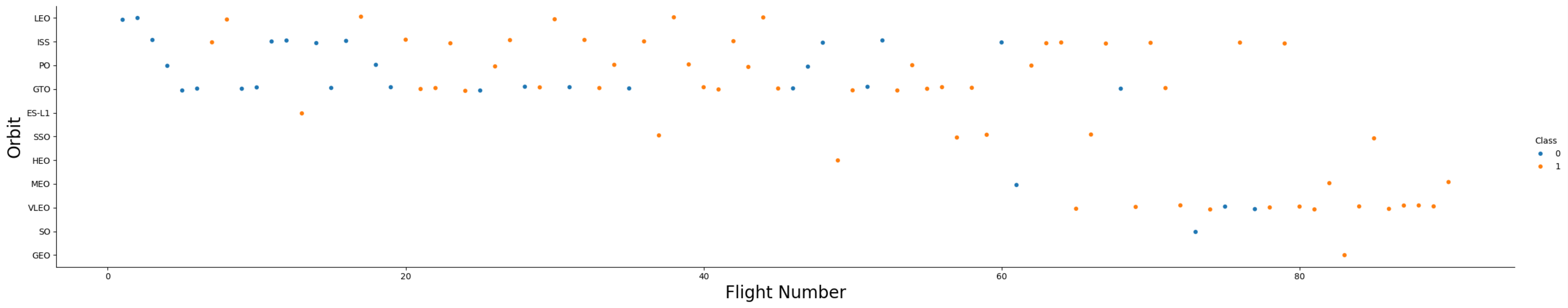
# Success Rate vs. Orbit Type

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



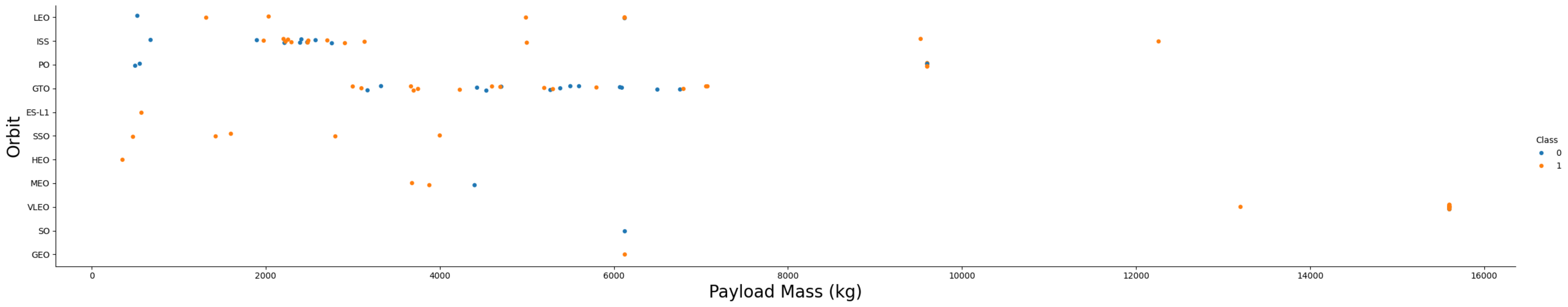
# Flight Number vs. Orbit Type

- The plot below shows the Flight Number vs. Orbit type. We observe that in the LEO orbit, success is related to the number of flights whereas in the GTO orbit, there is no relationship between flight number and the orbit.



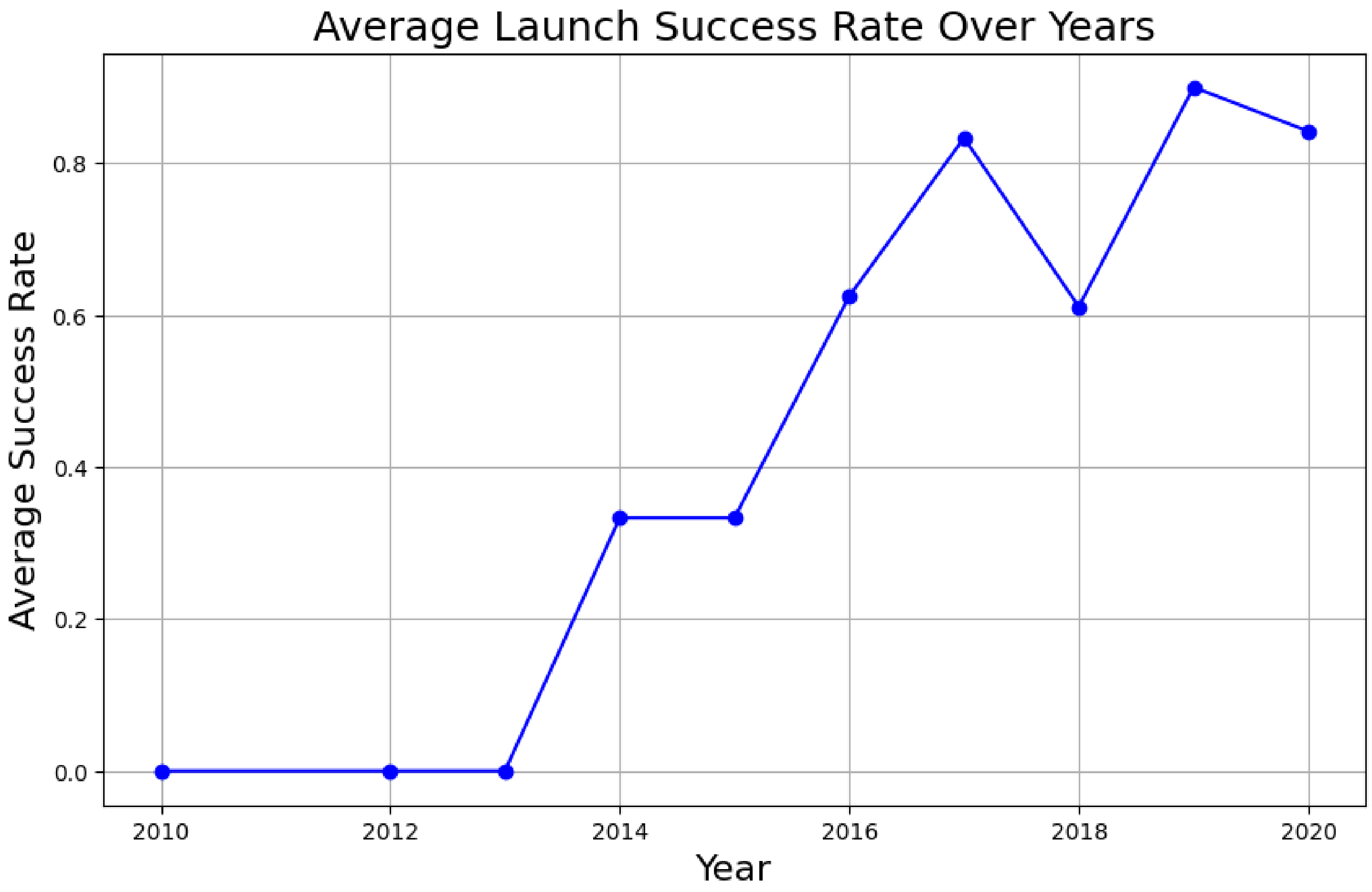
# Payload vs. Orbit Type

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



# Launch Success Yearly Trend

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



# All Launch Site Names

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

Task 1

Display the names of the unique launch sites in the space mission

```
[7]  %sql select distinct Launch_Site from SPACEXTBL;
```

```
 * sqlite:///data\spacex.db
Done.
```

| Launch_Site |
|---|
| CCAFS LC-40 |
| VAFB SLC-4E |
| KSC LC-39A |
| CCAFS SLC-40 |

# Launch Site Names Begin with 'CCA'

Task 2

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

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

Python

* sqlite:///data\spacex.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 |

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

# Total Payload Mass

- We calculated the total payload carried by boosters from NASA as 45596 using the query below

## Task 3

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

```
# %sql SELECT SUM(PAYLOAD_MASS__KG_) AS "Total Payload Mass (kg)" FROM SPACEXTABLE WHERE "Customer" LIKE 'NASA (CRS)';
%sql SELECT SUM(PAYLOAD_MASS__KG_) as 'Total Payload Mass (kg)' FROM SPACEXTABLE WHERE Customer LIKE 'NASA (CRS)';
```

[9]

```
 * sqlite:///data\spacex.db
Done.
```

| Total Payload Mass (kg) |
|---|
| 45596 |

# Average Payload Mass by F9 v1.1

- We calculated the average payload mass carried by booster version F9 v1.1 as 2928.4

## Task 4

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

```
%sql SELECT Booster_Version, AVG(PAYLOAD_MASS__KG_) AS 'Average Payload Mas
```

[10]

```
* sqlite:///data\spacex.db
Done.
```

| Booster_Version | Average Payload Mass (kg) |
|---|---|
| F9 v1.1 | 2928.4 |

# First Successful Ground Landing Date

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

## Task 5

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

*Hint:Use min function*

```
[11]
%sql SELECT MIN(Date) AS 'First Successful Landing Date' FROM SPACEXTABLE WHERE Landing_Outcome LIKE '%Success (ground pad)%';
# %sql SELECT Date FROM SPACEXTABLE WHERE Landing_Outcome LIKE 'Success (ground pad)';
```

```
 * sqlite:///data\spacex.db
Done.
```

| First Successful Landing Date |
|---|
| 2015-12-22 |

# Successful Drone Ship Landing with Payload between 4000 and 6000

- We used the **WHERE** clause to filter for boosters which have successfully landed on drone ship and applied the **AND** condition to determine successful landing with payload mass greater than 4000 but less than 6000

## Task 6

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

[12]

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

```
 * sqlite:///data\spacex.db
Done.
```

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

# Total Number of Successful and Failure Mission Outcomes

- We used GROUP BY to filter for **WHERE** MissionOutcome was a success or a failure.
- Number of Successful Flights = 98+1+1= 100
- Number of Unsuccessful Flights = 1

## Task 7

List the total number of successful and failure mission outcomes

[13]
```
%sql SELECT Mission_Outcome, COUNT(*) AS 'Total' FROM SPACEXTABLE GROUP BY Mission_Outcome;
```

```
 * sqlite:///data\spacex.db
Done.
```

| Mission_Outcome | Total |
|---|---|
| Failure (in flight) | 1 |
| Success | 98 |
| Success | 1 |
| Success (payload status unclear) | 1 |

# Boosters Carried Maximum Payload

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

## Task 8

List the names of the booster_versions which have carried th

```
[14]
%sql SELECT DISTINCT Booster_Version, PAYLOAD_MASS__KG_ FROM SPAC
```

```
 * sqlite:///data\spacex.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

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

Task 9

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 strftime('%m', Date) AS Month, Landing_Outcome, Booster_Version, Launch_Site FROM SPACEXTABLE WHERE Landing_Outcome LIKE '%Fail%' AND substr("Date", 0, 5)
# %sql SELECT DISTINCT Launch_Site FROM SPACEXTABLE ;
```

[15] Python

```
 * sqlite:///data\spacex.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

## Task 10

Rank the count of landing outcomes (such as Failure (drone ship) o
descending order.

[16]
```
%sql SELECT Landing_Outcome, COUNT(*) AS Count FROM SPACEXTABLE WHERE Dat
```

```
 * sqlite:///data\spacex.db
Done.
```

| Landing_Outcome | Count |
|---|---|
| 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 |

- We selected Landing outcomes and the **COUNT** of landing outcomes from the data and used the **WHERE** clause to filter for landing outcomes **BETWEEN** 2010-06-04 to 2010-03-20.
- We applied the **GROUP BY** clause to group the landing outcomes and the **ORDER BY** clause to order the grouped landing outcome in descending order.

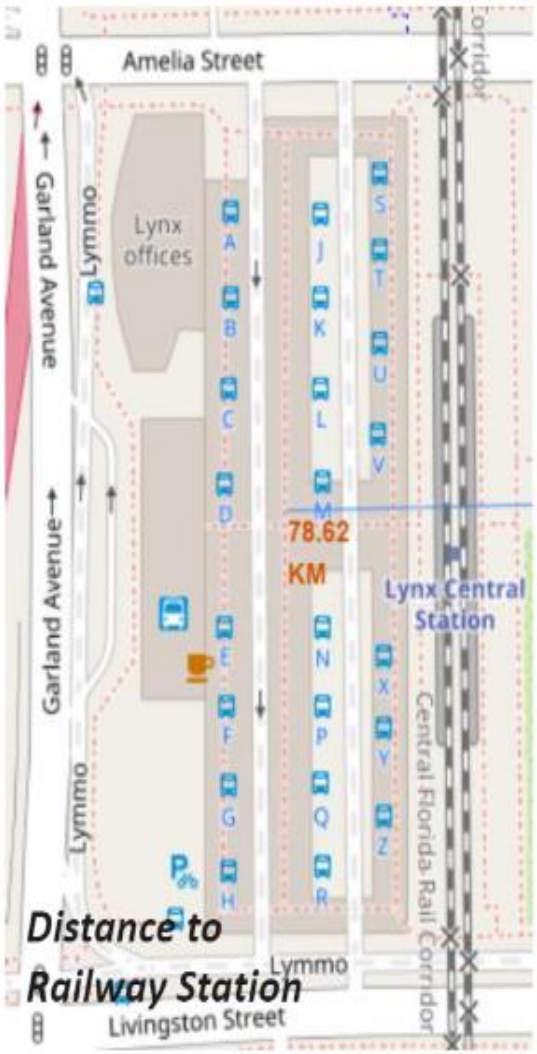
Section 4

# Launch Sites Proximities Analysis

# All launch sites global map markers

# Markers showing launch sites with color labels

# Launch Site distance to landmarks





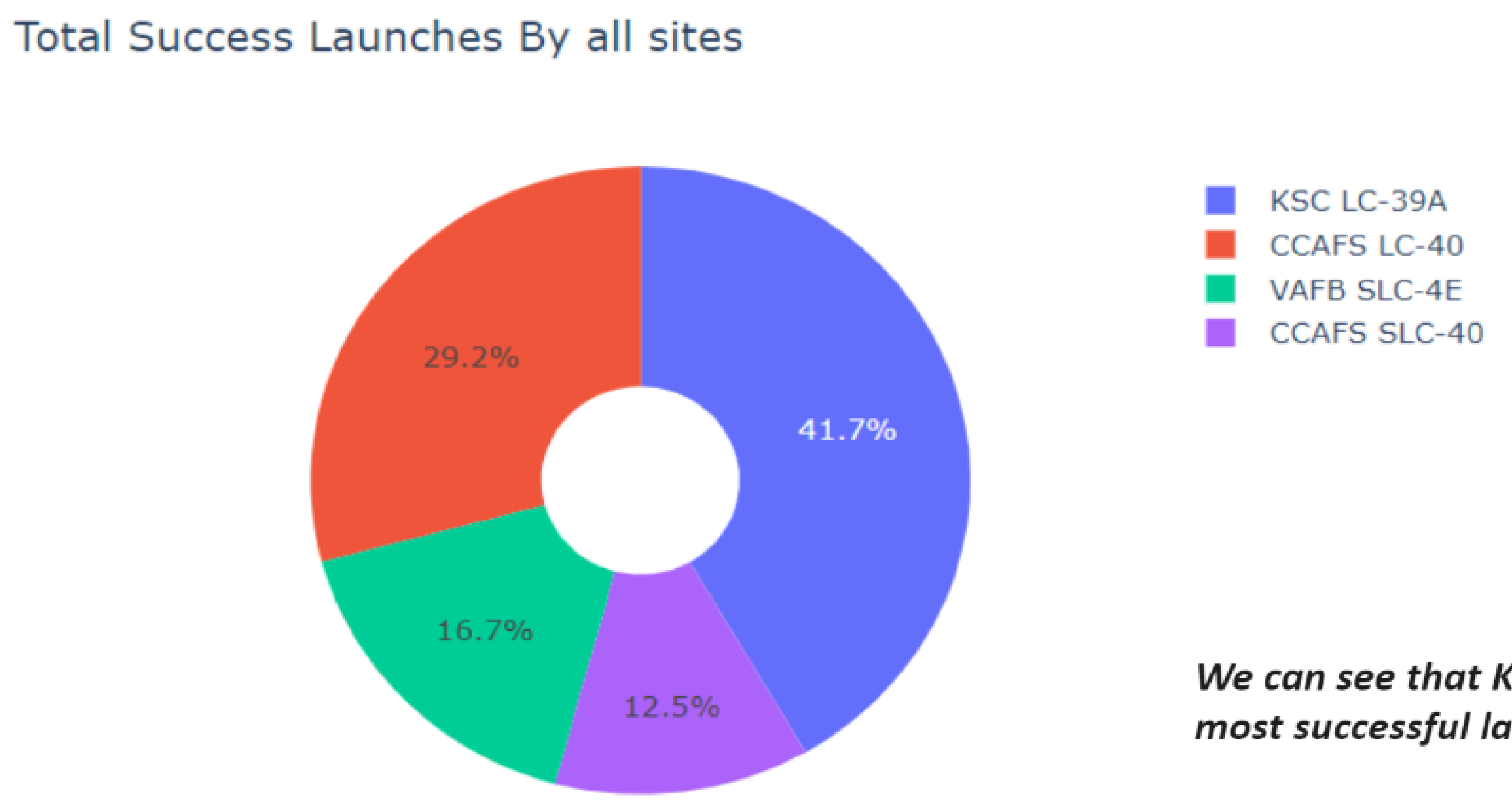



- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

Section 5

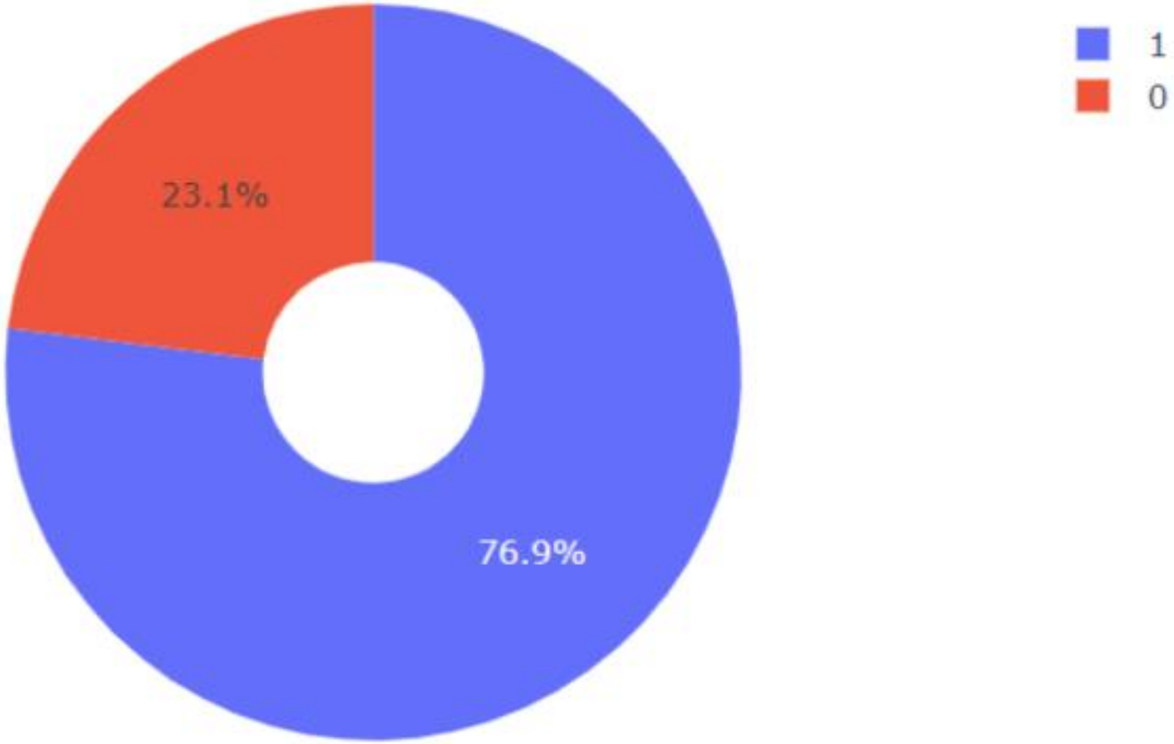
# Build a Dashboard with Plotly Dash

# Pie chart showing the success percentage achieved by each launch site



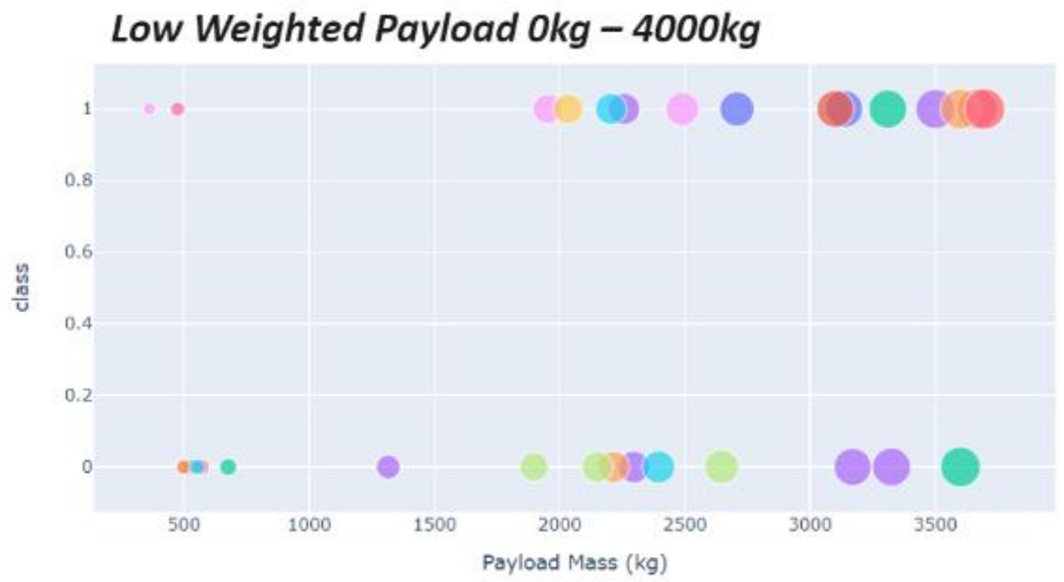

***We can see that KSC LC-39A had the most successful launches from all the sites***

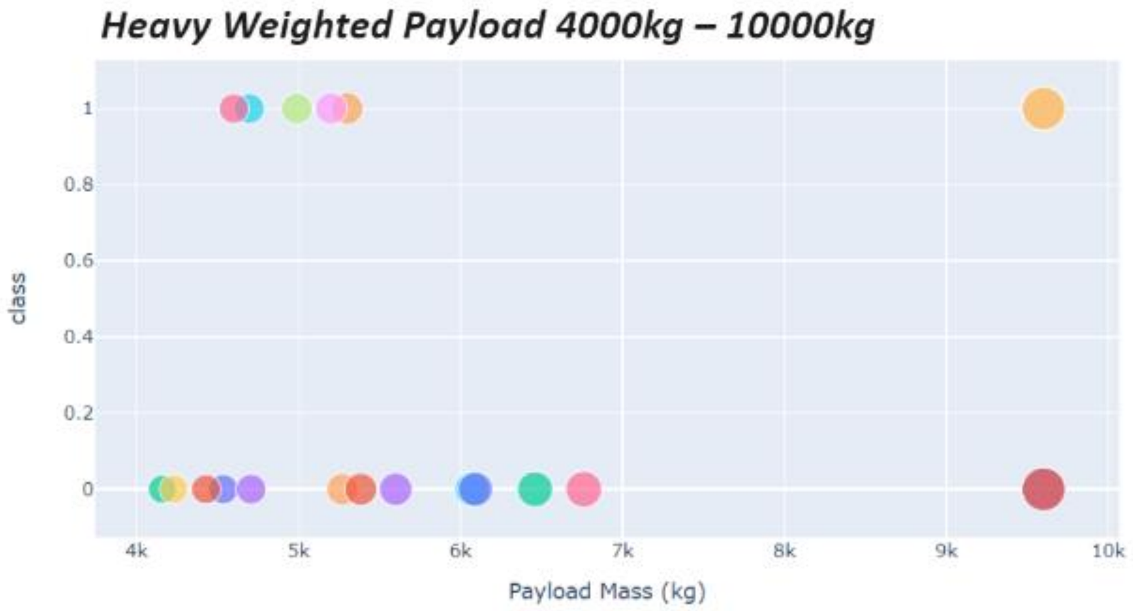
# Pie chart showing the Launch site with the highest launch success ratio




***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

```
if best_method == 'Decision Tree':
    print('Best params is :', tree_cv.best_params_)
elif best_method == 'K-Nearest Neighbors':
    print('Best params is :', knn_cv.best_params_)
elif best_method == 'Logistic Regression':
    print('Best params is :', logreg_cv.best_params_)
if best_method == 'SVM':
    print('Best params is :', svm_cv.best_params_)
```

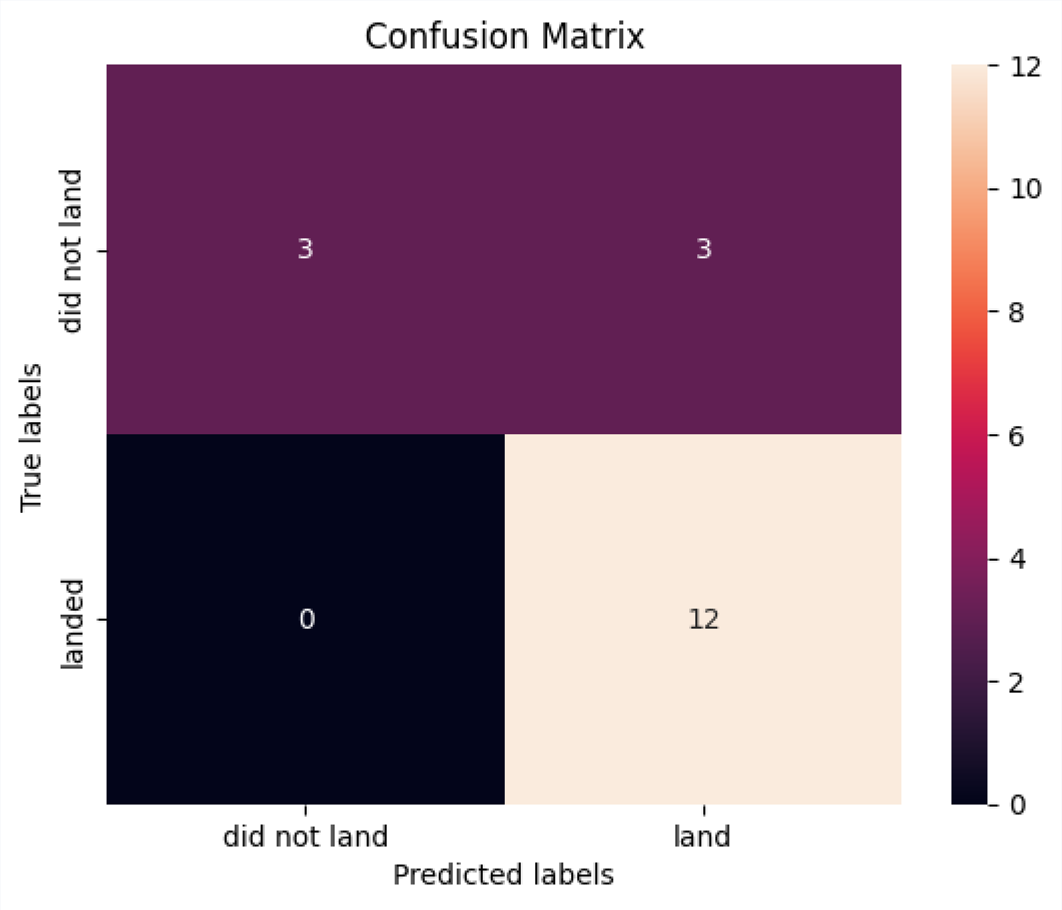
[73]

```
Accuracy Scores:
Logistic Regression: 0.8464285714285713
SVM: 0.8482142857142856
Decision Tree: 0.8732142857142856
K-Nearest Neighbors: 0.8482142857142858

Best-Performing Method:
Decision Tree with accuracy: 0.8732142857142856
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```

# Confusion Matrix

- The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes. The major problem is the false positives .i.e., unsuccessful landing marked as successful landing by the classifier.



# Conclusions

We can conclude that:

- The larger the flight amount at a launch site, the greater the success rate at a launch site.
- Launch success rate started to increase in 2013 till 2020.
- Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
- KSC LC-39A had the most successful launches of any sites.
- The Decision tree classifier is the best machine learning algorithm for this task.

# Thank you!