



IBM Developer  
SKILLS NETWORK

# Winning Space Race with Data Science

Melikşah Mercan  
01.02.2023



# 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:
  - Describe how data was collected
- Perform data wrangling
  - Describe how data was processed
- 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/allbosss/IBM-Spacex-Final-Project/blob/main/jpn/jupyter-labs-spacex-data-collection-api.ipynb>

```
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

- 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/allbosss/BM-Spacex-Final-Project/blob/main/jpn/jupyter-labs-webscraping.ipynb>

```
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 = []

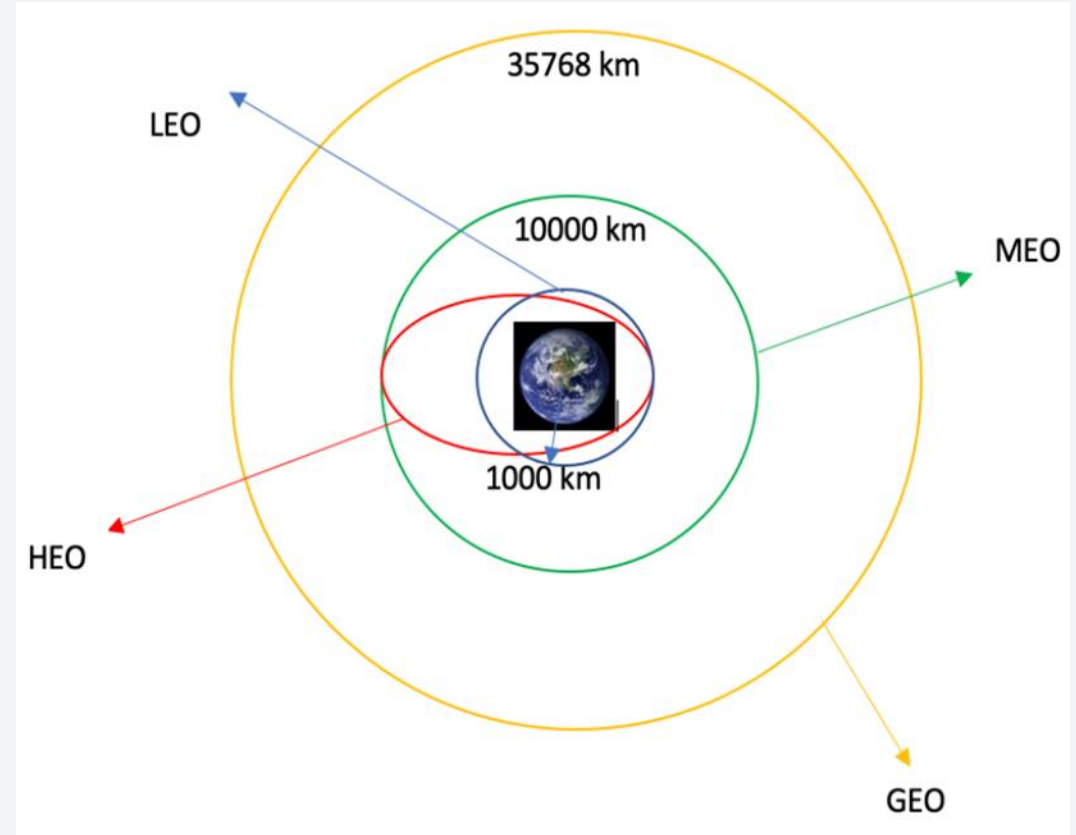
        # Apply find_all() function with 'th' element on first_launch_table
        # Iterate each th element and apply the provided extract_column_from_header() to get a column name
        # Append the Non-empty column name ('if name is not None and len(name) > 0') into a list called column_names

        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
```

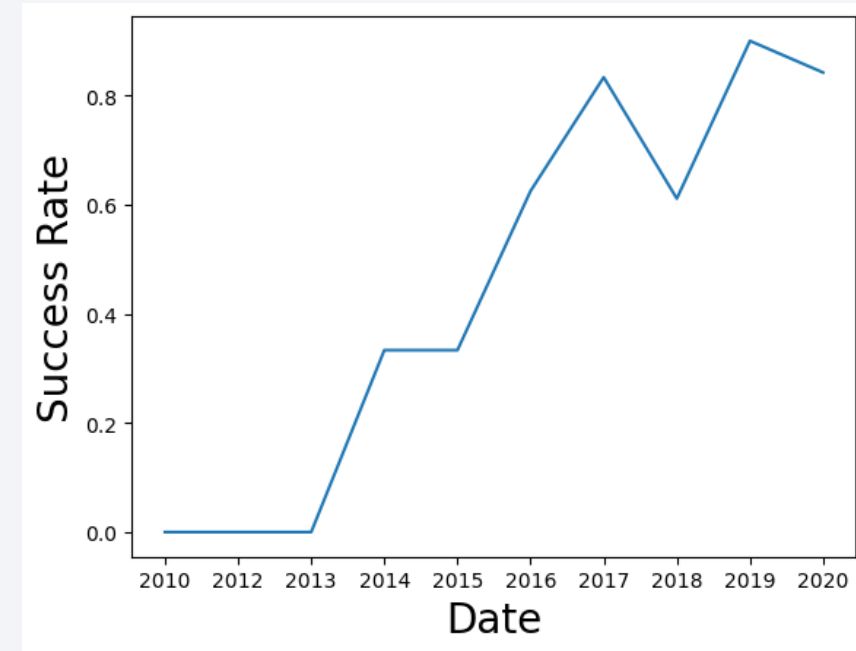
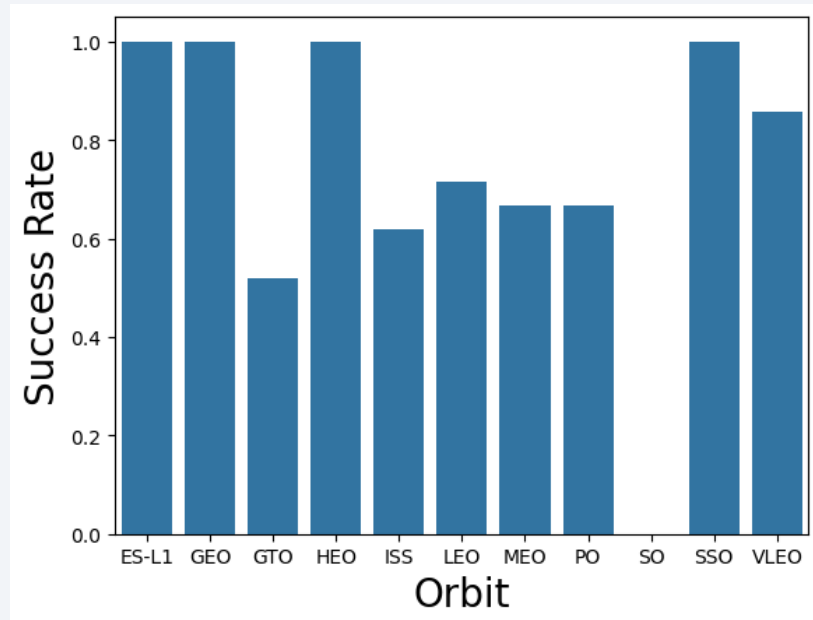
# Data Wrangling

- 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/allbosss/IBM-Spacex-Final-Project/blob/main/jpn/labs-jupyter-spacex-Data%20wrangling.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/allbosss/IBM-Spacex-Final-Project/blob/main/jpn/jupyter-labs-eda-dataviz.ipynb>

# EDA with SQL

---

- We loaded the SpaceX dataset into a PostgreSQL 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/allbosss/IBM-Spacex-Final-Project/blob/main/jpn/jupyter-labs-eda-sql-coursera\\_sqlite.ipynb](https://github.com/allbosss/IBM-Spacex-Final-Project/blob/main/jpn/jupyter-labs-eda-sql-coursera_sqlite.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
  - [https://github.com/allbosss/IBM-Spacex-Final-Project/blob/main/jpn/lab\\_jupyter\\_launch\\_site\\_location.jupyterlite.ipynb](https://github.com/allbosss/IBM-Spacex-Final-Project/blob/main/jpn/lab_jupyter_launch_site_location.jupyterlite.ipynb)



# 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/allbosss/IBM-Spacex-Final-Project/blob/main/spacex\\_dash\\_app.py](https://github.com/allbosss/IBM-Spacex-Final-Project/blob/main/spacex_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/allbosss/IBM-Spacex-Final-Project/blob/main/jpn/SpaceX\\_Machine\\_Learning\\_Prediction\\_Part\\_5.jupyterlite.ipynb](https://github.com/allbosss/IBM-Spacex-Final-Project/blob/main/jpn/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb)

# Results

---

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



The background of the slide is an abstract composition. It features a dark blue base color. Overlaid on this are numerous diagonal streaks in shades of red and cyan. A faint, light blue grid pattern is also visible, particularly in the lower-left quadrant. The overall effect is dynamic and technological.

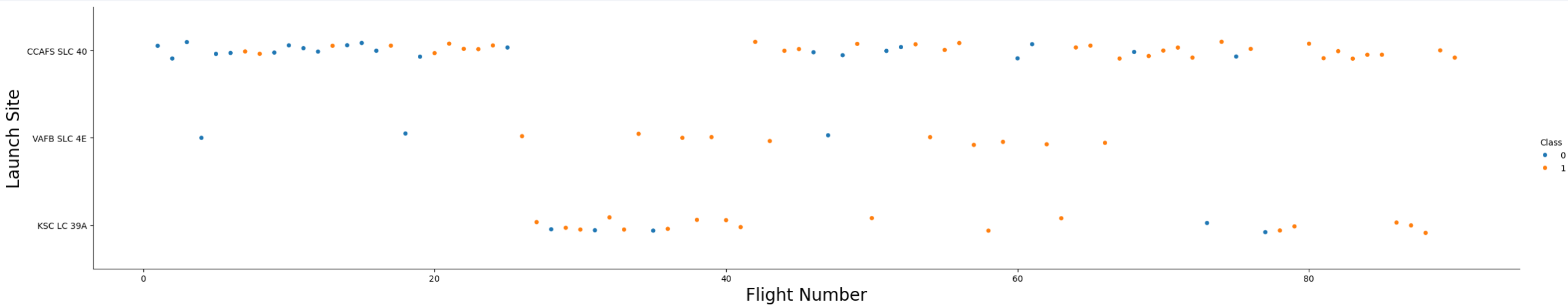
Section 2

# Insights drawn from EDA



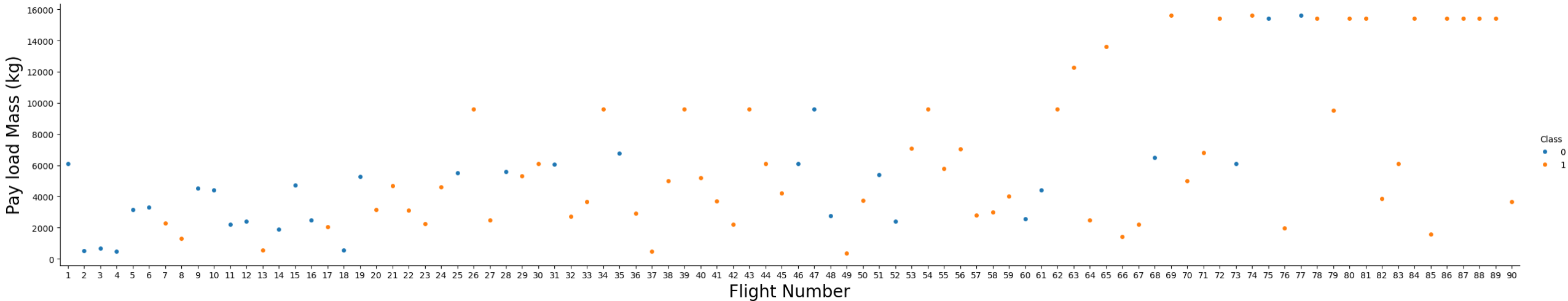
# Flight Number vs. Launch Site

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





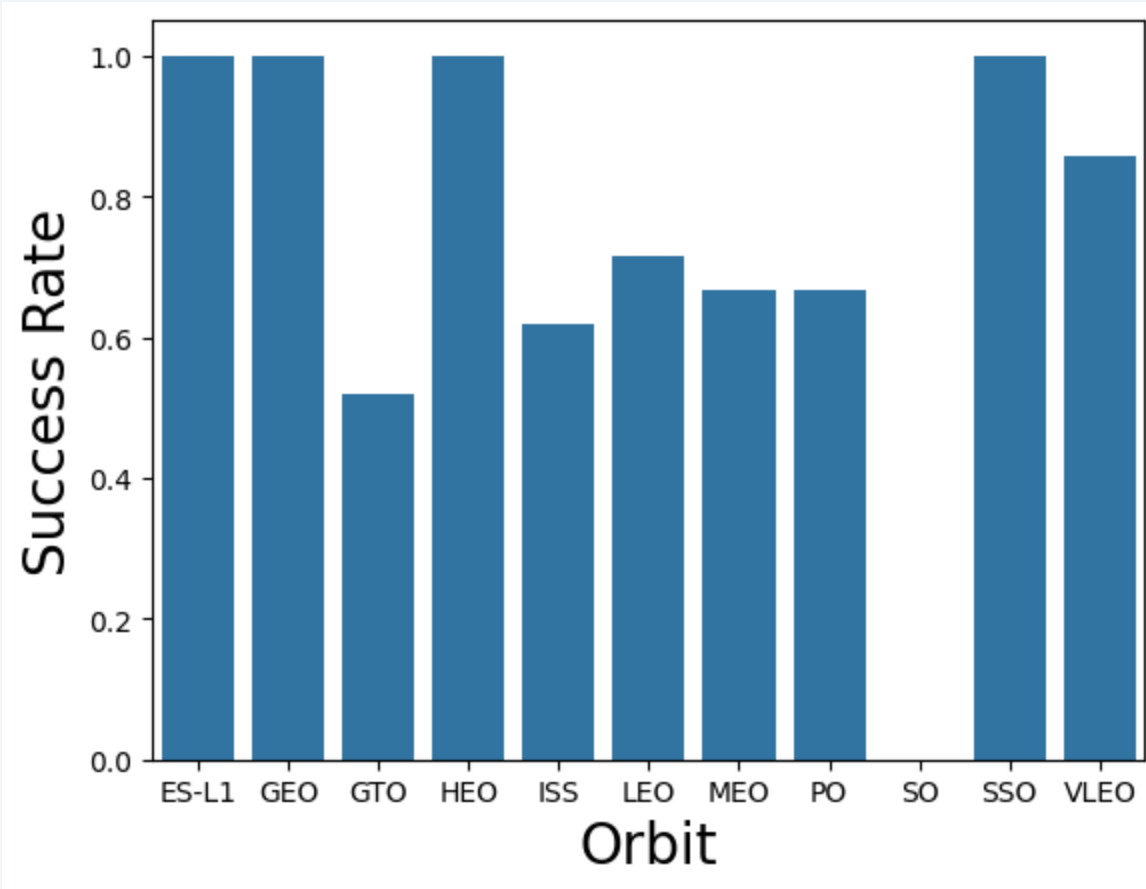
# Payload vs. Launch Site



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

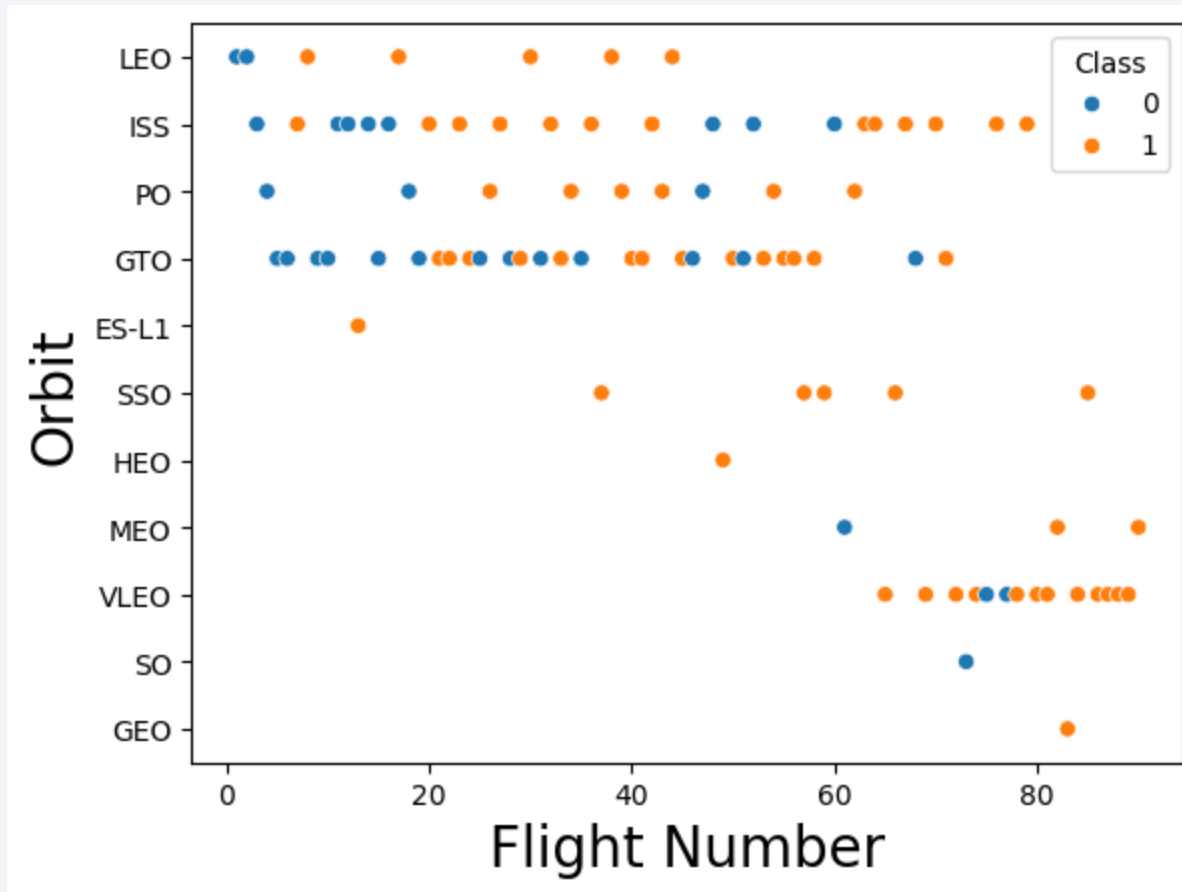
# Success Rate vs. Orbit Type

---



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

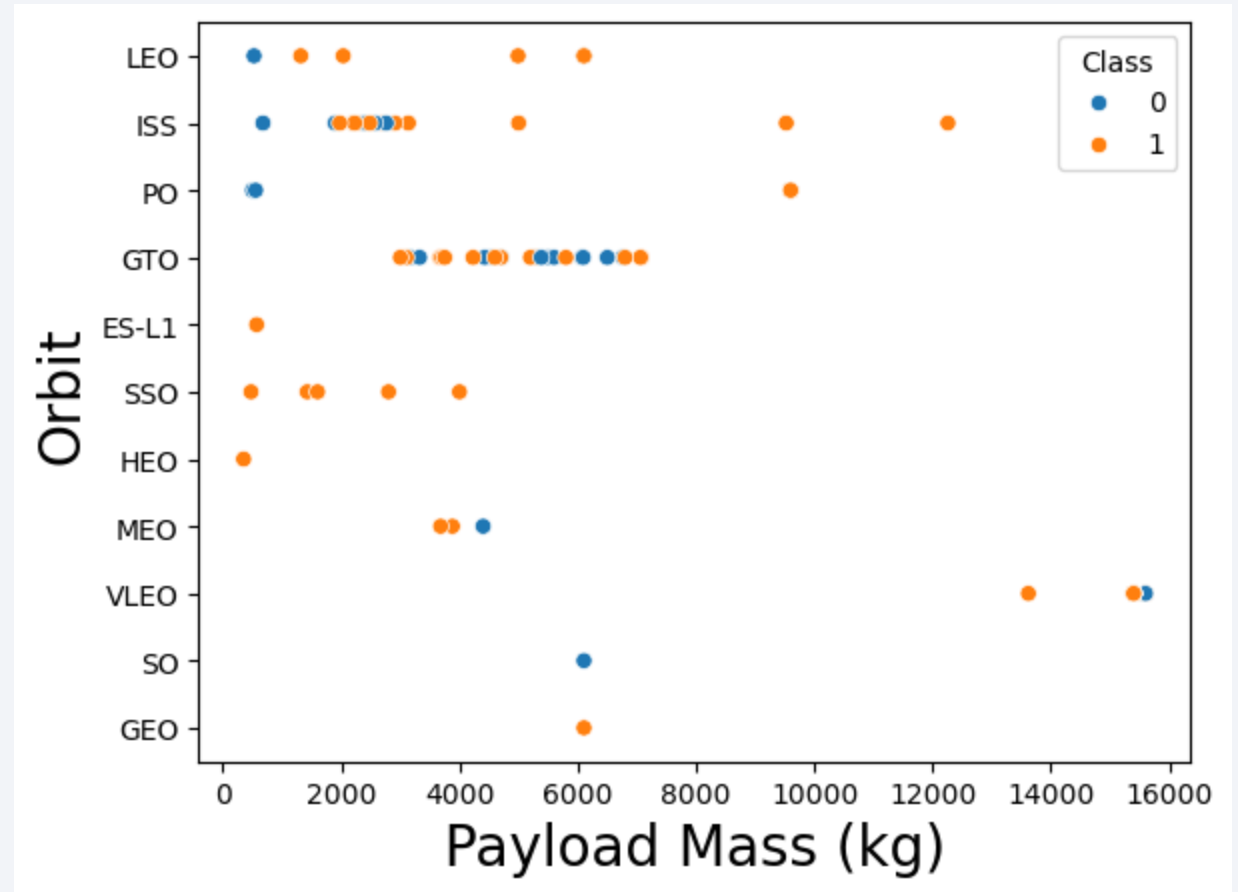
# Flight Number vs. Orbit Type



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.

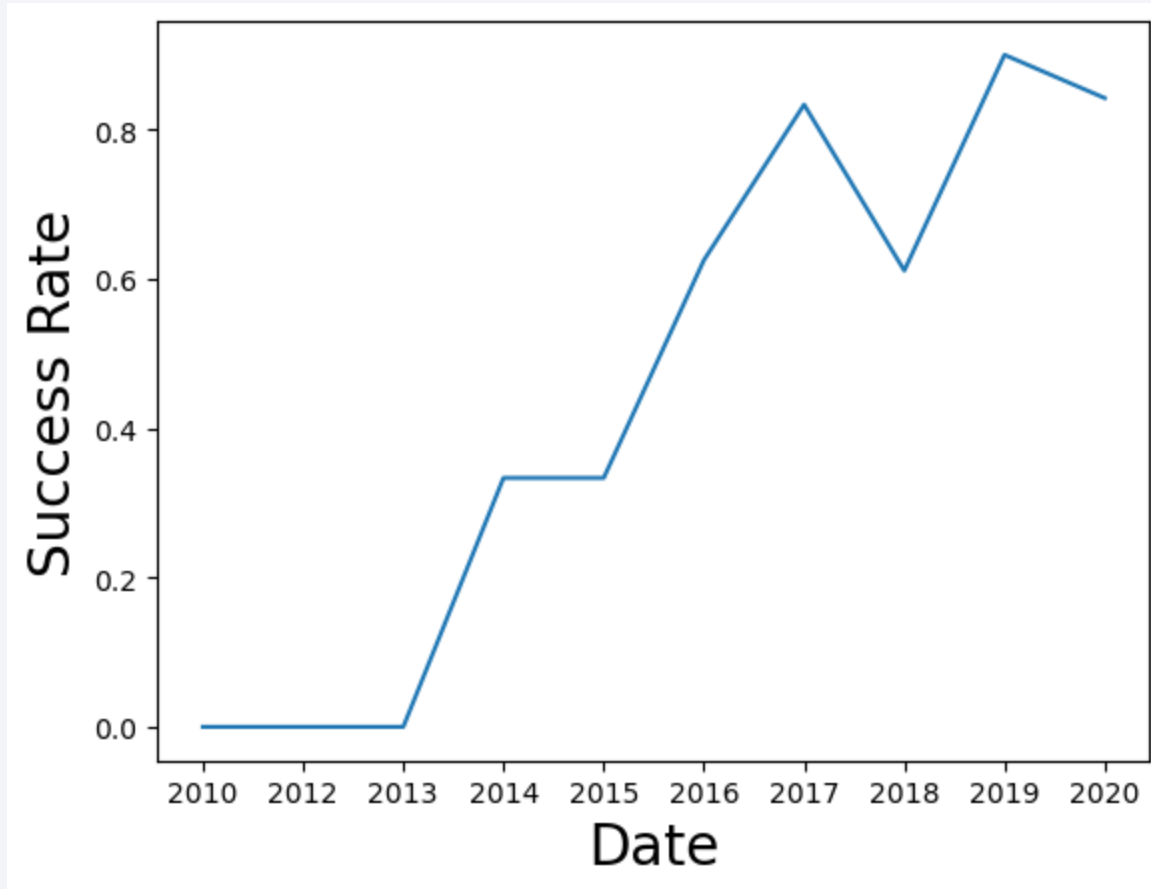
# Payload vs. Orbit Type

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

```
In [25]: %sql SELECT DISTINCT Launch_Site FROM SPACEXTABLE;
```

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

```
Done.
```

```
Out[25]: Launch_Site
```

```
CCAFS LC-40
```

```
VAFB SLC-4E
```

```
KSC LC-39A
```

```
CCAFS SLC-40
```

# Launch Site Names Begin with 'CCA'

We used the query above to display 5 records where launch sites begin with 'CCA'

## Task 2

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

In [31]: `%sql SELECT * FROM SPACEXTABLE WHERE Launch_Site LIKE 'CCA%' LIMIT 5;`

\* sqlite:///my\_data1.db  
Done.

Out[31]:

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

---

We calculated the total payload carried by boosters from NASA as 48213 using the query below:

## Task 3

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

```
In [32]: %sql SELECT SUM(Payload_Mass__kg_) AS Total_Payload_Mass FROM SPACEXTABLE WHERE Customer LIKE '%NASA (CRS)%';
```

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

```
Out[32]: Total_Payload_Mass  
         48213
```

# 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

```
In [34]: %sql SELECT AVG(Payload_Mass__kg_) AS Average_Payload_Mass FROM SPACEXTABLE WHERE Booster_Version = 'F9 v1.1';
```

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

```
Out[34]: Average_Payload_Mass  
         2928.4
```

# First Successful Ground Landing Date

---

## Task 5

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

*Hint: Use min function*

```
In [35]: %sql SELECT MIN(Date) AS First_Successful_Landing_Date FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (ground pad)';
```

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

```
Out[35]: First_Successful_Landing_Date  
                2015-12-22
```

- We observed that the dates of the first successful landing outcome on ground pad was 22<sup>nd</sup> December 2015.



## 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

```
In [36]: %sql SELECT 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.
```

```
Out[36]: 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 wildcard like '%' to filter for **WHERE** MissionOutcome was a success or a failure.

## Task 7

List the total number of successful and failure mission outcomes

```
In [37]: %sql SELECT Mission_Outcome, COUNT(*) AS Total FROM SPACEXTABLE GROUP BY Mission_Outcome;
```

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

```
Out[37]:
```

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 the maximum payload mass. Use a subquery

```
In [39]: %sql SELECT Booster_Version FROM SPACEXTABLE WHERE Payload_Mass__kg_ = ( SELECT MAX(Payload_Mass__kg_) FROM SPACEXTABLE );
```

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

```
Out[39]: Booster_Version
```

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

# 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: SQLite 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.**

```
In [46]: %sql SELECT \
        CASE substr(Date, 6, 2) \
        WHEN '01' THEN 'January' \
        WHEN '02' THEN 'February' \
        WHEN '03' THEN 'March' \
        WHEN '04' THEN 'April' \
        WHEN '05' THEN 'May' \
        WHEN '06' THEN 'June' \
        WHEN '07' THEN 'July' \
        WHEN '08' THEN 'August' \
        WHEN '09' THEN 'September' \
        WHEN '10' THEN 'October' \
        WHEN '11' THEN 'November' \
        WHEN '12' THEN 'December' \
        END AS Month_Name, \
        Landing_Outcome, \
        Booster_Version, \
        Launch_Site \
        FROM SPACEXTABLE \
        WHERE substr(Date, 0, 5) = '2015' \
        AND Landing_Outcome = 'Failure (drone ship)';
```

\* sqlite:///my\_data1.db  
Done.

```
Out[46]:
```

Month_Name	Landing_Outcome	Booster_Version	Launch_Site
January	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
April	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) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

```
In [47]: %sql SELECT Landing_Outcome, COUNT(*) AS Total, RANK() OVER (ORDER BY COUNT(*) DESC) AS Rank FROM SPACEXTABLE \
WHERE Date BETWEEN '2010-06-04' AND '2017-03-20' \
GROUP BY Landing_Outcome \
ORDER BY Total DESC;
```

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

```
Out[47]:
```

Landing_Outcome	Total	Rank
No attempt	10	1
Success (drone ship)	5	2
Failure (drone ship)	5	2
Success (ground pad)	3	4
Controlled (ocean)	3	4
Uncontrolled (ocean)	2	6
Failure (parachute)	2	6
Precluded (drone ship)	1	8

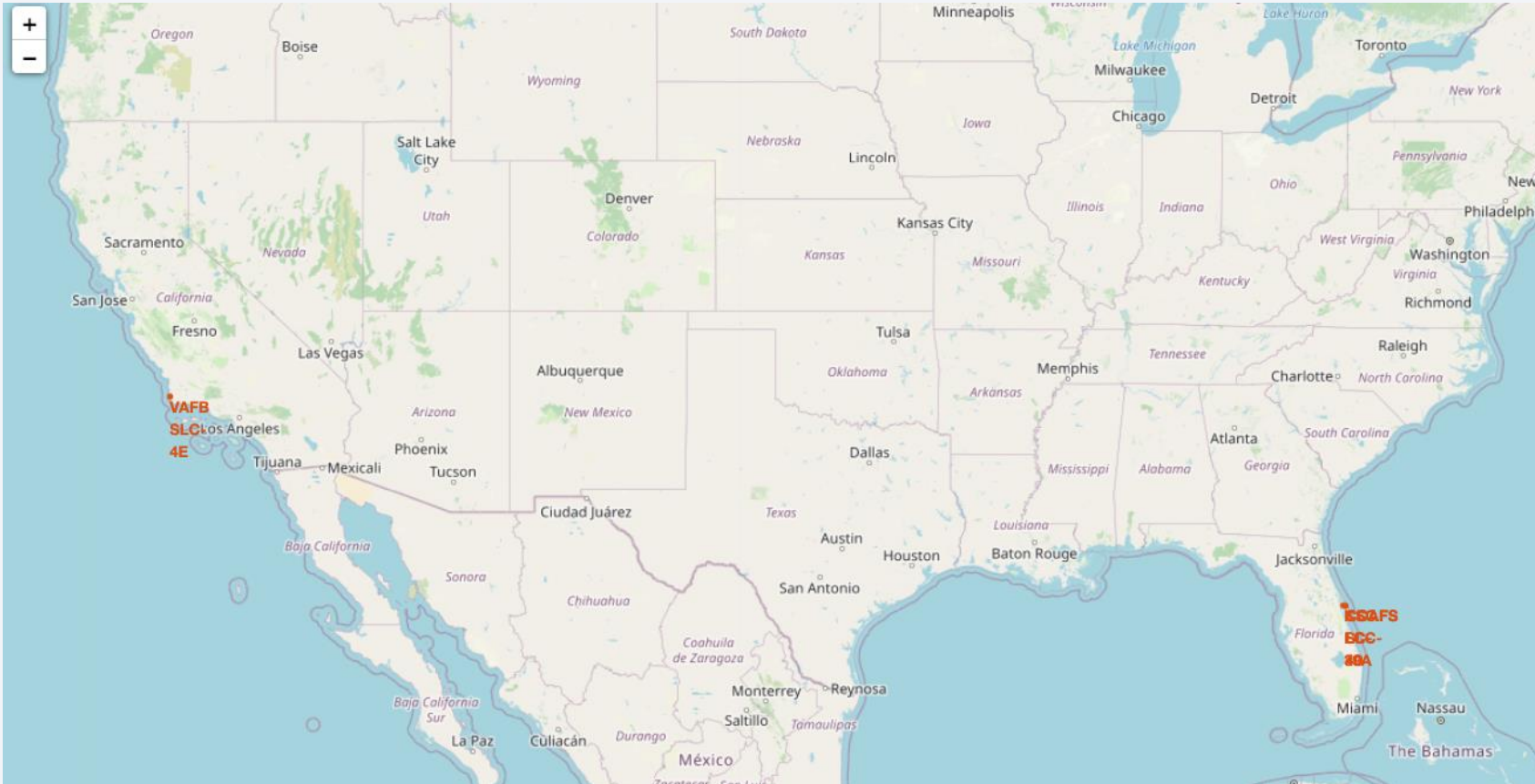
- We selected Landing outcomes and the **COUNT** of landing outcomes from the data and used the **WHERE** clause to filter for landing outcomes **BETWEEN** 2010-06-04 to 2017-03-20.
- We applied the **GROUP BY** clause to group the landing outcomes and the **ORDER BY** clause to order the grouped landing outcome in descending order.

A satellite view of Earth from space, showing the curvature of the planet and city lights at night. The image is a composite of a solid blue background on the left and a satellite photograph of Earth on the right. The Earth's surface is dark, with numerous bright yellow and orange lights representing cities and urban areas. The horizon of the Earth is visible as a curved line separating the dark surface from the deep blue of space.

Section 3

# Launch Sites Proximities Analysis

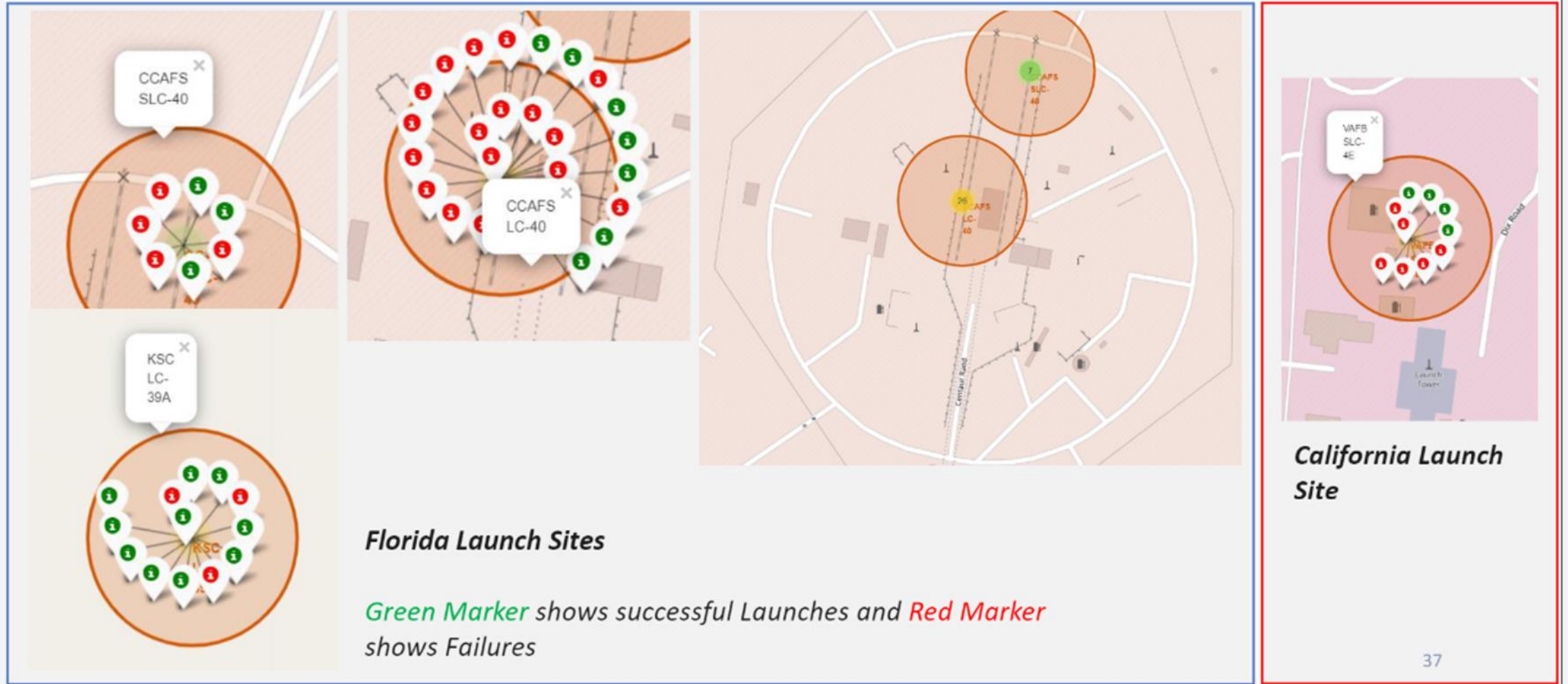
# All launch sites global map markers



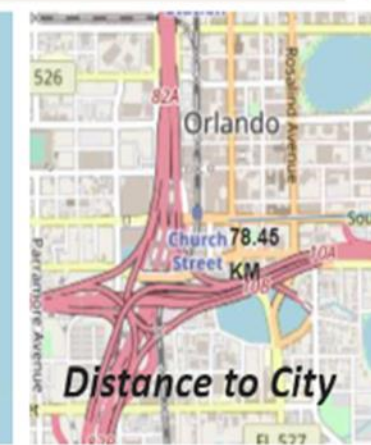
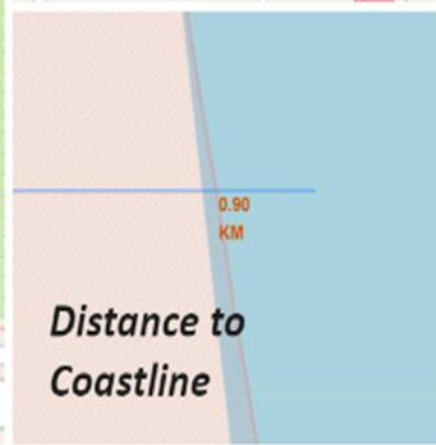
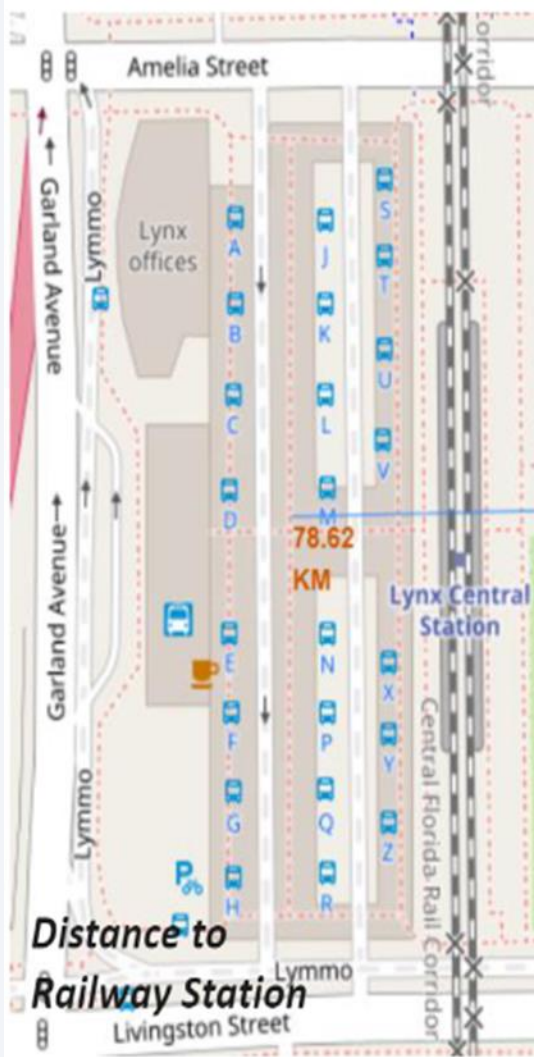
We can see that the SpaceX launch sites are in the USA coasts. (Florida and California)



# 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



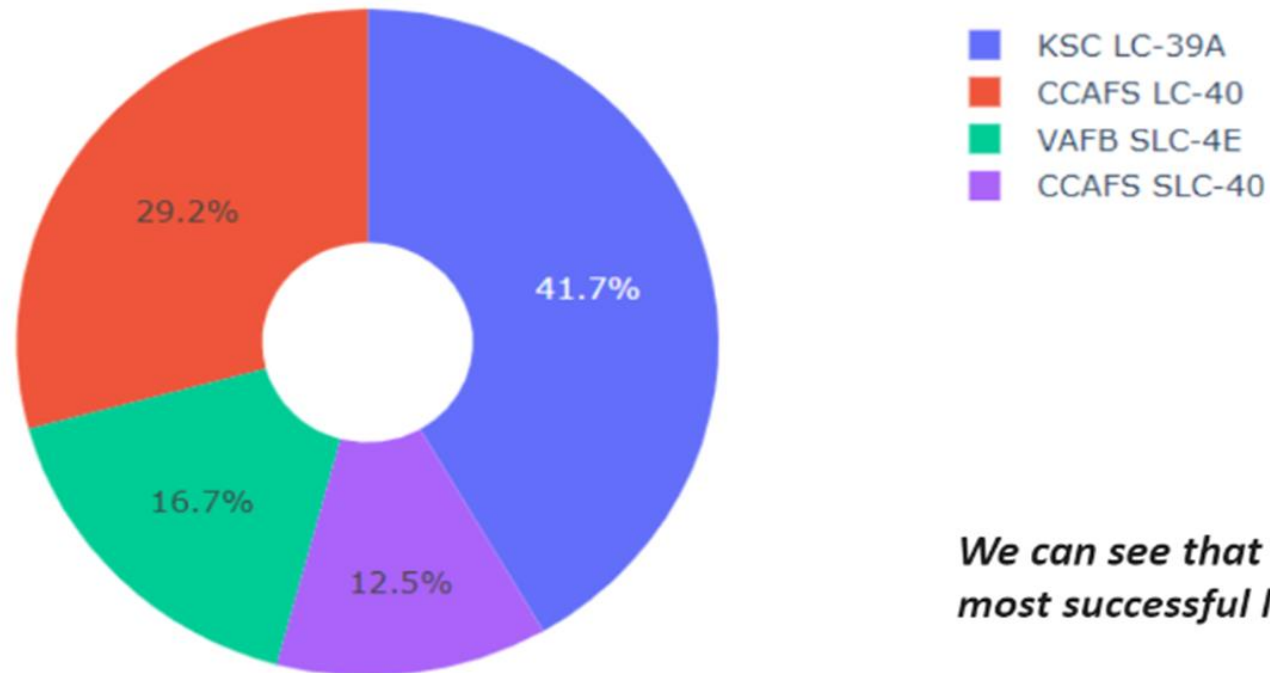


Section 4

# Build a Dashboard with Plotly Dash

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

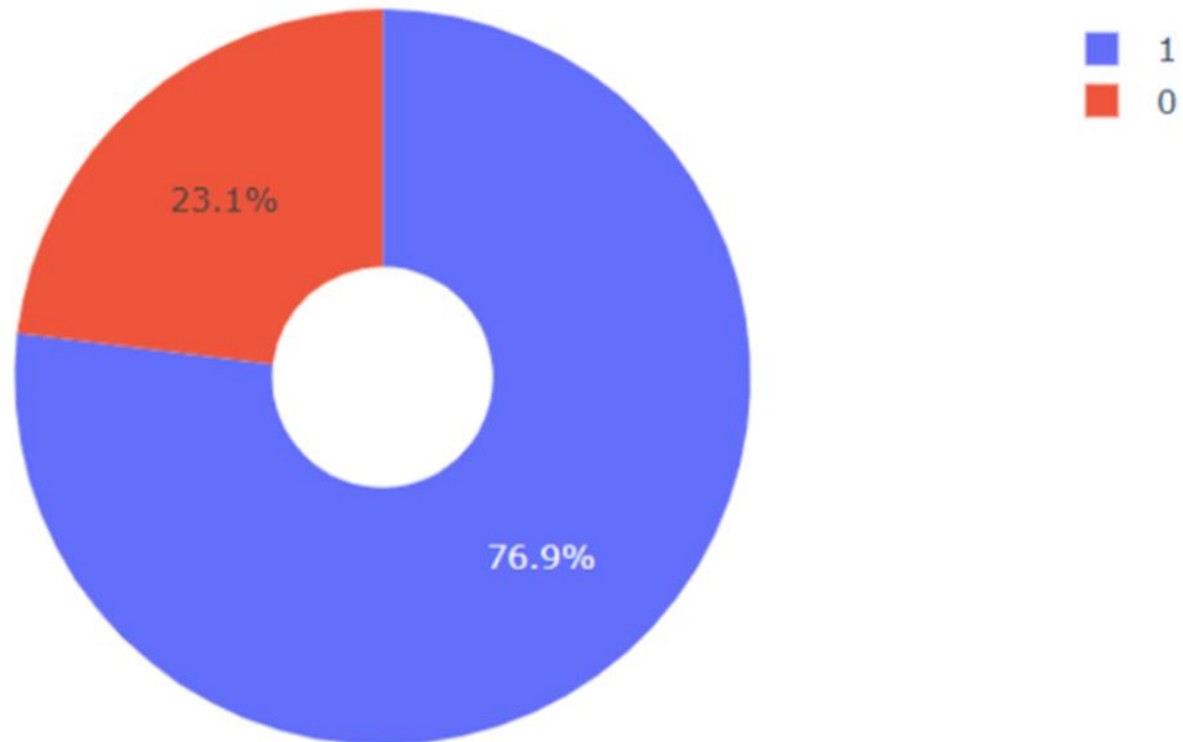
Total Success Launches By all sites



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

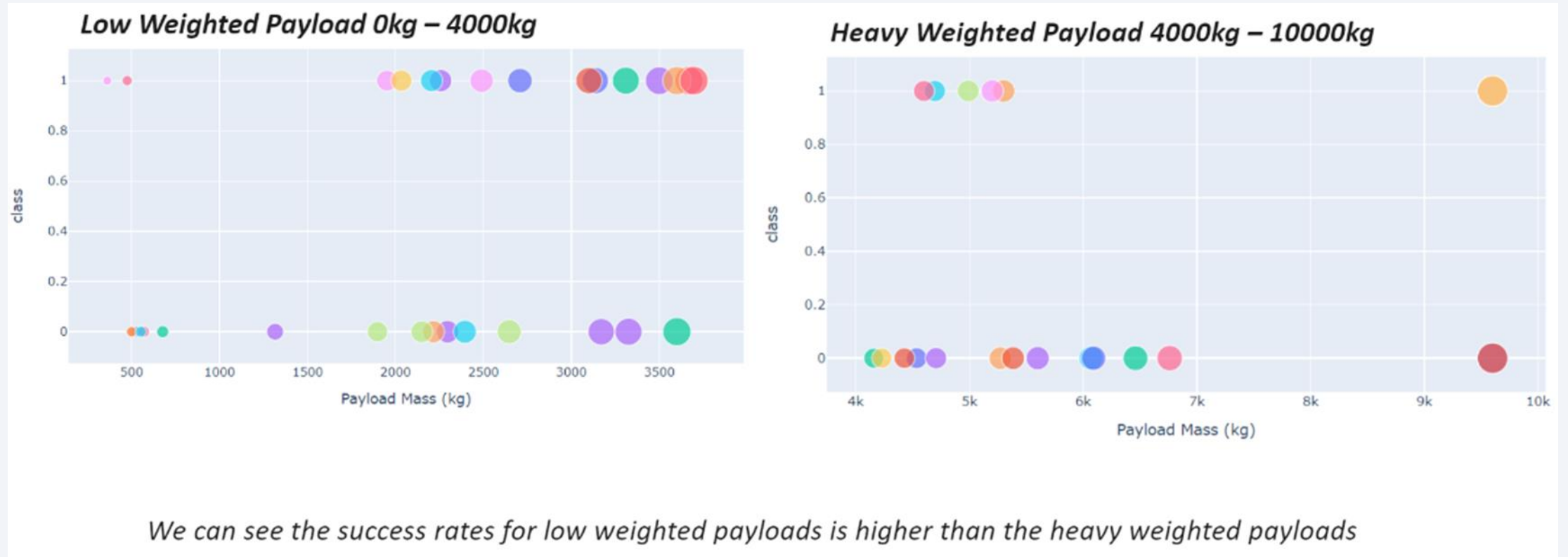
## Pie chart showing the Launch site with the highest launch success ratio

---



***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







Section 5

# Predictive Analysis (Classification)

# Classification Accuracy

---

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

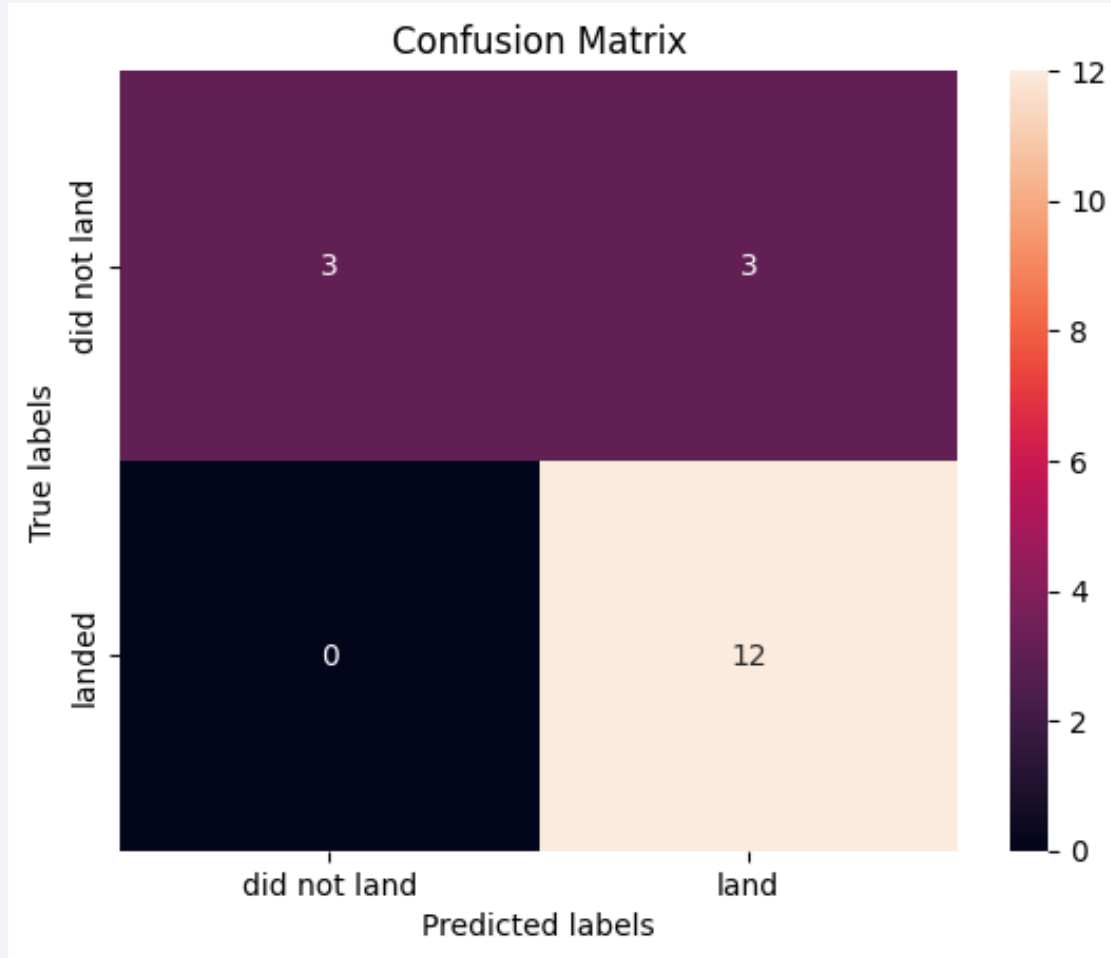
```
models = {'KNeighbors': knn_cv.best_score_,
          'DecisionTree': tree_cv.best_score_,
          'LogisticRegression': logreg_cv.best_score_,
          'SupportVector': svm_cv.best_score_}

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

---

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.

# Appendix

---

- My Github link: <https://github.com/allbosss/IBM-Spacex-Final-Project/tree/main>





Thank you!

