IBM Developer
SKILLS NETWORK

# Winning Space Race with Data Science

Ibrahim Muhammed
22|05|2023

# Outline

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

# Executive Summary

- Aim: To predict landing success rate of spaceX falcon 9
- Webscrapping of data from the source.
- Exploratory data analysis (EDA) of first stage.
- SQL queries were used for EDA
- Machine learning models were used for predicting the success landing rate

# Introduction

- The project emphasized on landing successes rate of Space X falcon 9. Space X is a company that advertised falcon 9 rocket launch on it website.
- Our aim is to predict the success of the launch by manipulating the data using python and constructing machine learning model.

Section 1

# Methodology

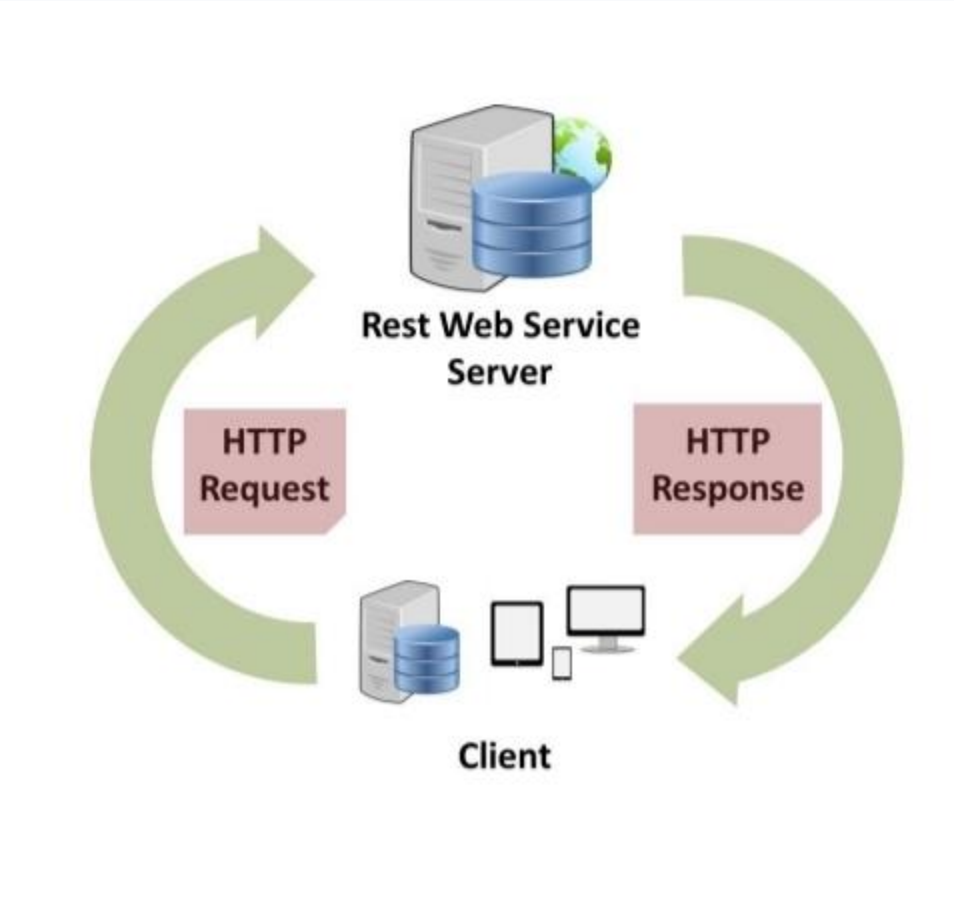
# Methodology

## Executive Summary

- Data collection methodology:
    - Web scrapping was used to collect data from an article on Wikipedia about spaceX rocket launch.
- Perform data wrangling
    - Python packages were used to manipulate the data on jupyter notebook.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
    - Python packages were imported which were used to build,tune and evaluate machine learning models

# Data Collection

- BeautifulSoup was used to webscrap falcon 9 launch record.
- GET method was used to request falcon9 HTML page,as an HTML response



# Data Collection – SpaceX API

- Python code was used to generate a GET response
- Beautiful soup was used to extract content from the data.
- https://github.com/iro062/Applied-Data-Science-Capstone-IBM/blob/main/jupyter-labs-webscraping.ipynb

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

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

# Data Collection - Scraping

- BeautifulSoup function was used to create a beautifulsoup object.
- Title attribute was used to print out the tittle
- find_all function was used to generate table
- https://github.com/iro062/Applied-Data-Science-Capstone-IBM/blob/main/jupyter-labs-webscraping.ipynb

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

```
# Use soup.title attribute
title = soup.title.string
print("Page Title:", title)
```

```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
```

# Data Wrangling

- Python packages were used to find pattern and manipulate data
- The necessary packages were imported
- Value_count function was used to calculate the number of particular variables;
    - Number of launch on site
    - the number and occurrence of each orbit
    - number and occurence of mission outcome per orbit type
- A column was created for classification of the outcome column
- The average of the column class was calculated with a value of 0.666

https://github.com/iro062/Applied-Data-Science-Capstone-IBM/blob/main/spacex-data_wrangling_jupyterlite.jupyterlite.ipynb

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for outcome in df["Outcome"]:
  if outcome in bad_outcomes:
    landing_class.append(0)
  else:
    landing_class.append(1)
```

Fig: classification of the outcome column into binary 0 and 1

# EDA with Data Visualization

- The visualisation of some features or columns were plotted to check for correlation
- Yearly success rate of launch was visualized
- It indicated that there was a rise from 2013
- https://github.com/iro062/Applied-Data-Science-Capstone-IBM/blob/main/jupyter_labs_eda_dataviz.ipynb

# EDA with SQL

- Firstly, sql extention was created and connection was established from a database
- Sql query was performed for the following task;
  - the names of the unique launch sites in the space mission
  - Display 5 records where launch sites begin with the string 'CCA'
  - Display the total payload mass carried by boosters launched by NASA (CRS)
  - Display average payload mass carried by booster version F9 v1.1
  - List the date when the first succesful landing outcome in ground pad was acheived.
  - List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
  - List the total number of successful and failure mission outcomes
  - List the names of the booster_versions which have carried the maximum payload mass.
  - 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
  - Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.
- https://github.com/iro062/Applied-Data-Science-Capstone-IBM/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

# Build an Interactive Map with Folium

- Folium packages were imported and installed to build an Interactive map
- Markers,circles and lines were used to mark the success or failed launches for each site on the map
- The distances between a launch site to its proximities was calculated
- https://github.com/iro062/Applied-Data-Science-Capstone-IBM/blob/main/lab_jupyter_launch_site_location.ipynb

# Build a Dashboard with Plotly Dash

- Pie chart and scatter plot were used
- SpaceX launch site
- Pie chart and Scatter plot were used to showcase the following
  - Site with the largest launches
  - Site with the highest launch success rate
  - Payload with highest and lowest success rate
  - F9 booster version with highest lauch success rate
- https://github.com/iro062/Applied-Data-Science-Capstone-IBM/blob/main/Interactive%20Dashboard%20with%20Ploty%20Dash

# Predictive Analysis (Classification)

- The required packages machine learning packages were imported
- The data was splitted into train and test sets
- Various classification models were carried out such as logistic regression and SVM
- The models were evaluated and improved
- The score and accuracy for each model was calculated and compared to choose the best performing model
- https://github.com/iro062/Applied-Data-Science-Capstone-IBM/blob/main/SpaceX_Machine_Learning_Prediction_Part_5_jupyterlite%20(3).ipynb

# Results

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

Section 2

# Insights drawn from EDA

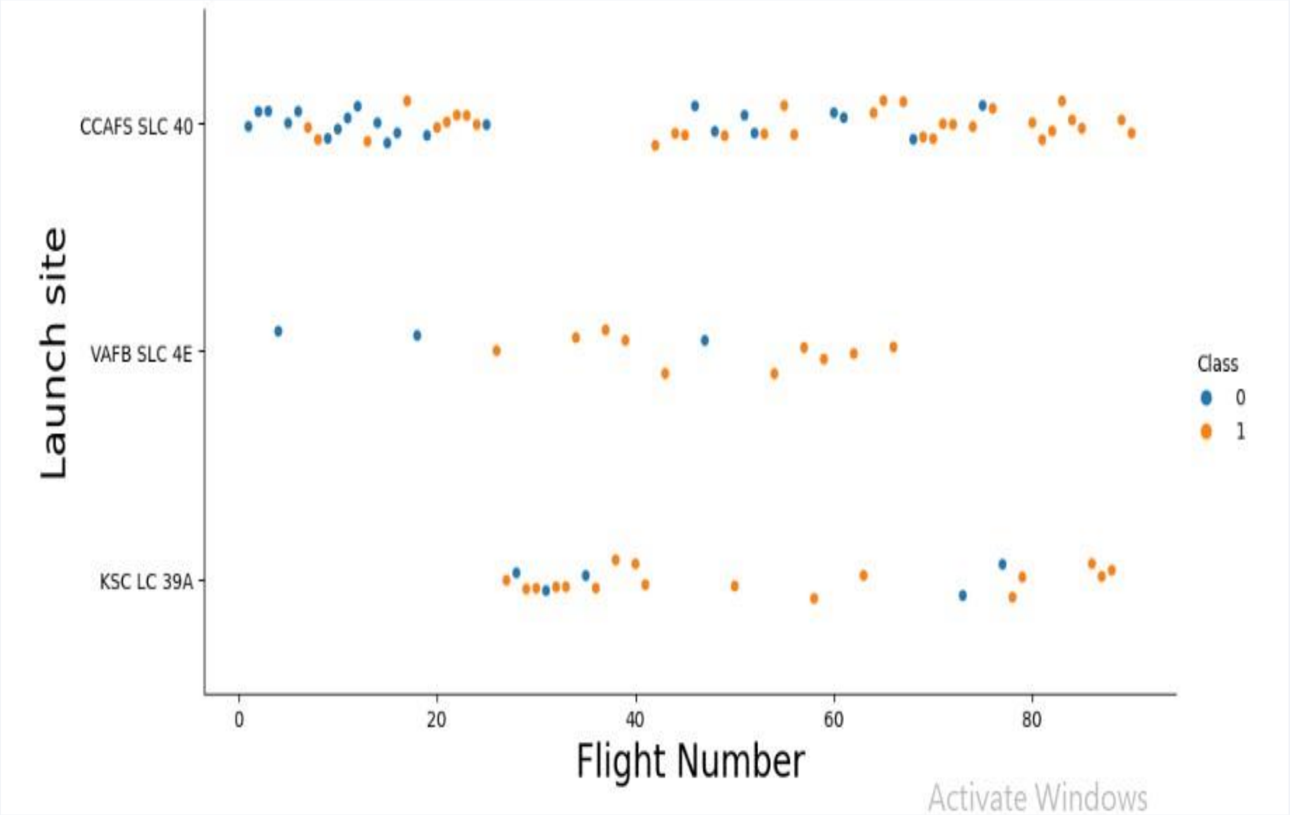
# Flight Number vs. Launch Site

- A scatter plot of Flight Number vs. Launch Site

- CCAFS LC-40 at the range of 0 to 20 flight number has high rate of low success rate and high success rate at the range of 60 to 80

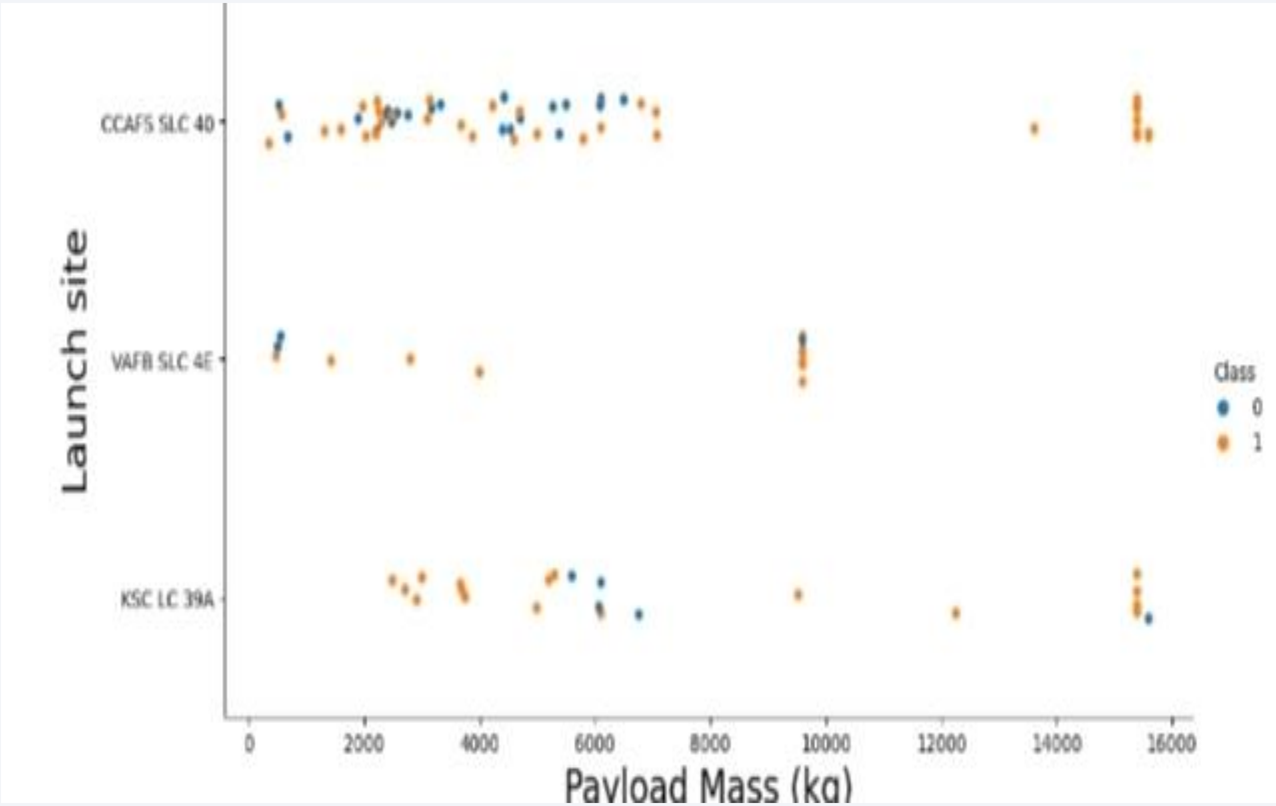
- VAFB SLC 4E has low flight number and a high success rate

- KSC LC 39A had no flight number output at the range of 0 to 20 with few numbers at the range of 40 to 6p but had high success rate



# Payload vs. Launch Site

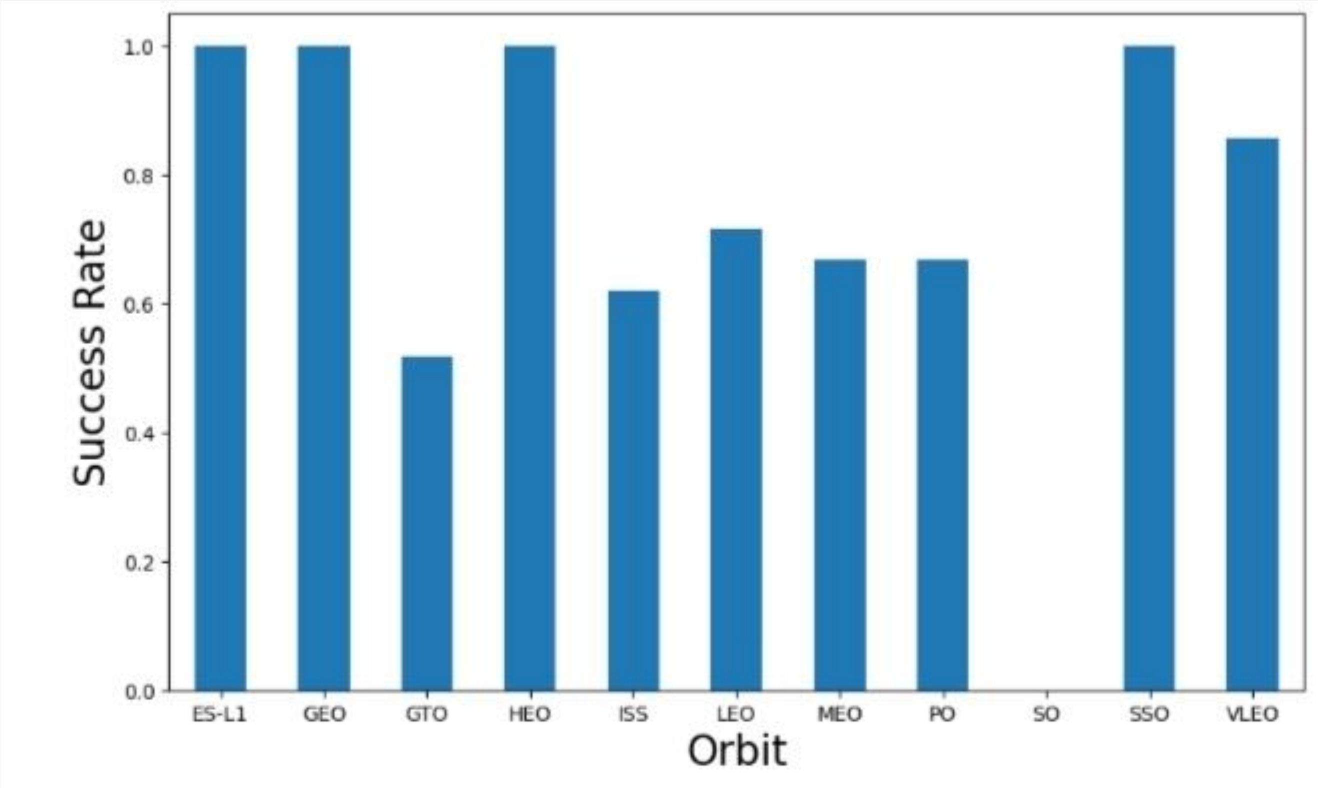
- A scatter plot of Payload vs. Launch Site
- For the VAFB-SLC launchsite ,there are no rockets launched for heavypayload mass(greater than 10000).



# Success Rate vs. Orbit Type

- A bar chart for the success rate of each orbit type

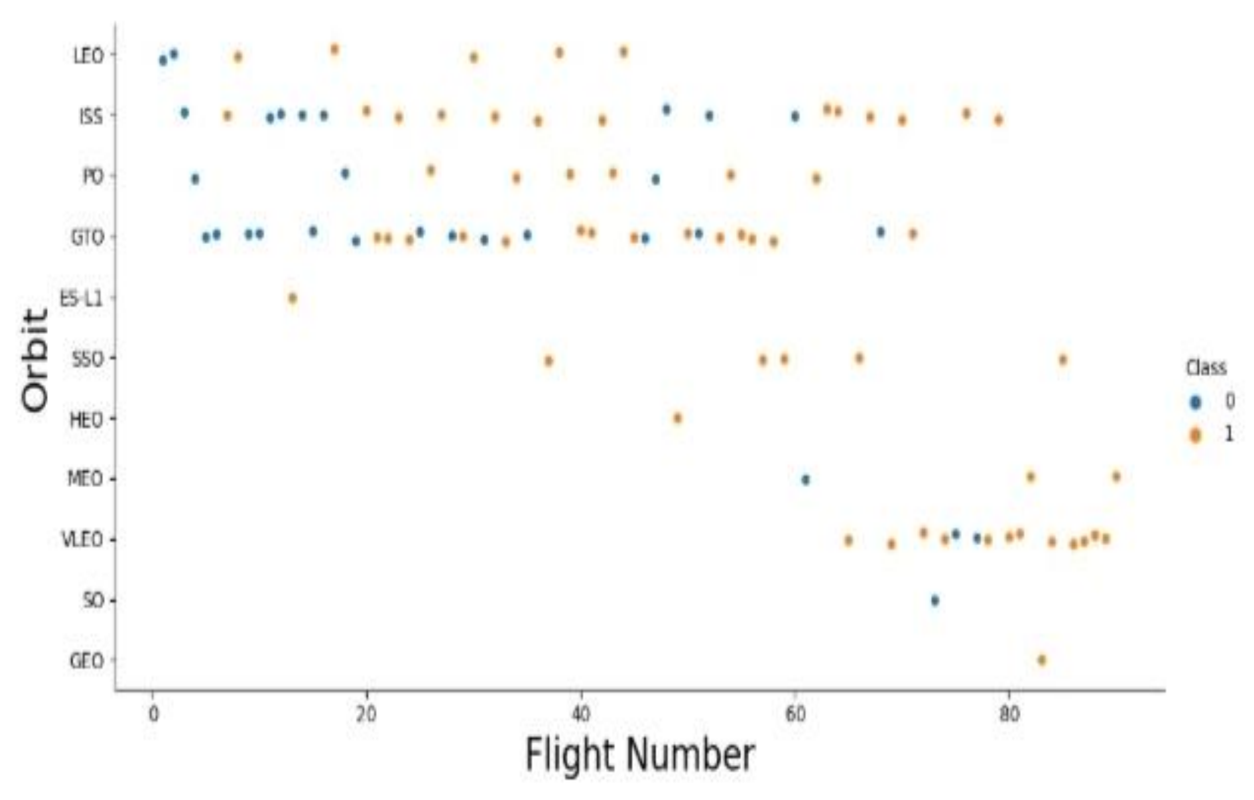
- Orbit types ES-L1, SSO, and GEO have the highest success rate with GTO having low success rate



# Flight Number vs. Orbit Type

- A scatter point of Flight number vs. Orbit type

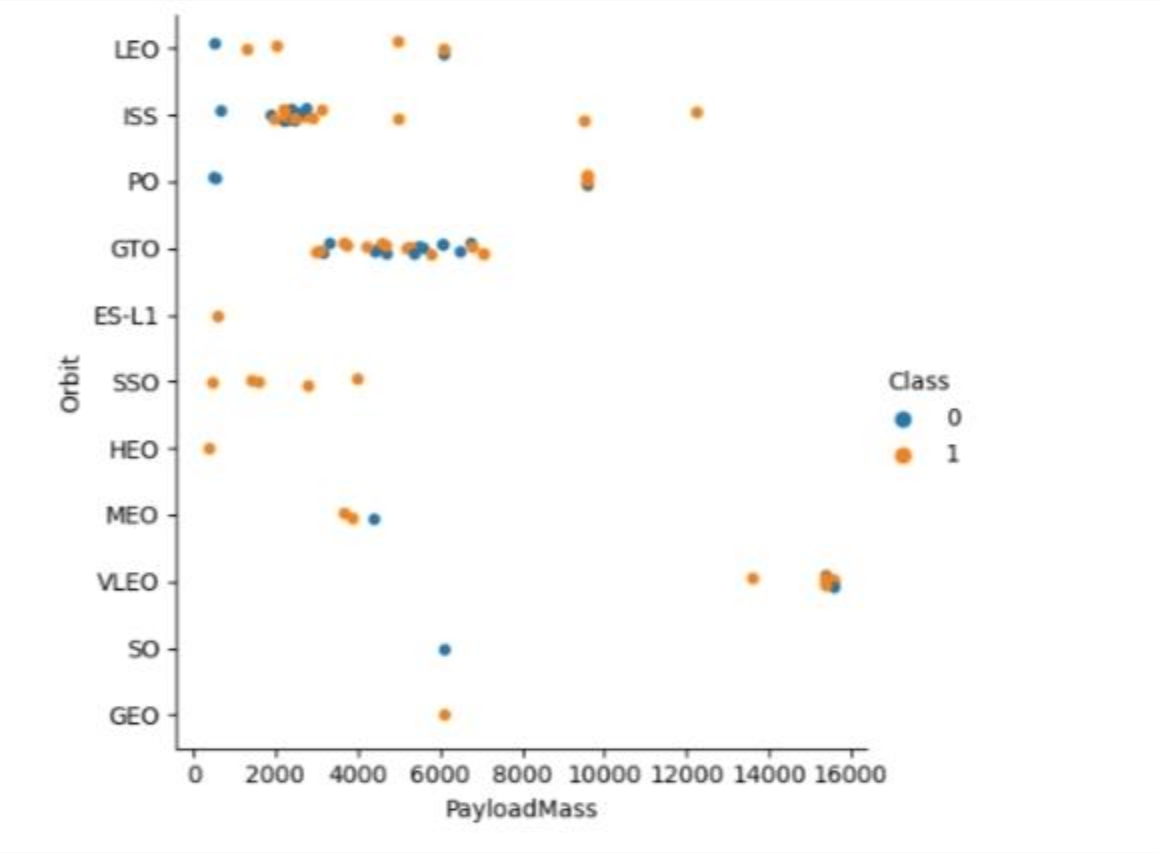
- In the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.



# Payload vs. Orbit Type

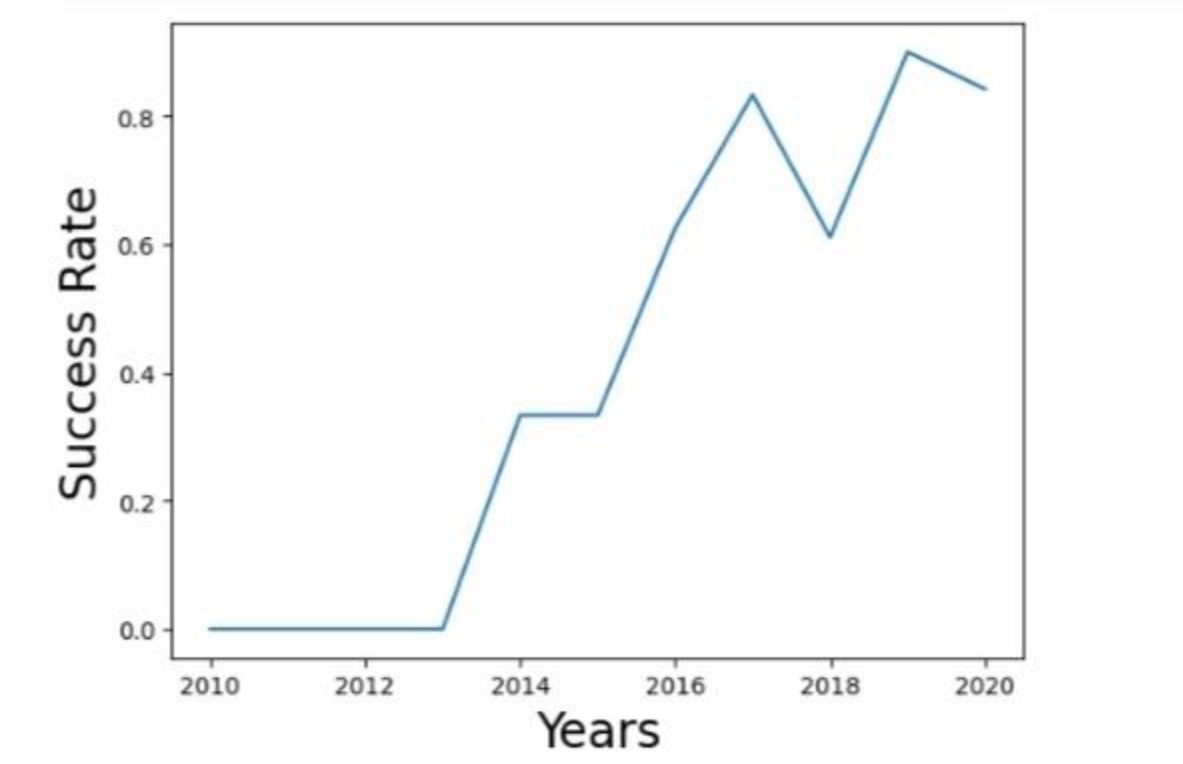
- A scatter point of payload vs. orbit type

- Heavy payloads have a negative influence on GTO orbits and positive on GTO and Polar LEO (ISS) orbits.



# Launch Success Yearly Trend

- A line chart of yearly average success rate
- The sucess rate since 2013 kept increasing till 2020



# All Launch Site Names

- The names of the unique launch sites

### Task 1

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

In [8]:
```
%sql SELECT DISTINCT Launch_Site FROM SPACEXTBL;
```

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

Out[8]:

| Launch_Site |
|---|
| CCAFS LC-40 |
| VAFB SLC-4E |
| KSC LC-39A |
| CCAFS SLC-40 |
| None |

# Launch Site Names Begin with 'CCA'

- 5 records where launch sites begin with `CCA`

## Task 2

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

In [9]:
```
%%sql SELECT *
FROM SPACEXTBL
WHERE Launch_Site LIKE 'CCA%'
LIMIT 5;
```

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

Out[9]:

| Date | Time (UTC) | Booster_Version | Launch_Site | Payload | PAYLOAD_MASS__KG_ | Orbit | Customer | Mission_Outcome | La |
|---|---|---|---|---|---|---|---|---|---|
| 06/04/2010 | 18:45:00 | F9 v1.0 B0003 | CCAFS LC-40 | Dragon Spacecraft Qualification Unit | 0.0 | LEO | SpaceX | Success | |
| 12/08/2010 | 15:43:00 | F9 v1.0 B0004 | CCAFS LC-40 | Dragon demo flight C1, two CubeSats, barrel of Brouere cheese | 0.0 | LEO (ISS) | NASA (COTS) NRO | Success | |
| 22/05/2012 | 7:44:00 | F9 v1.0 B0005 | CCAFS LC-40 | Dragon demo flight C2 | 525.0 | LEO (ISS) | NASA (COTS) | Success | |
| 10/08/2012 | 0:35:00 | F9 v1.0 B0006 | CCAFS LC-40 | SpaceX CRS-1 | 500.0 | LEO (ISS) | NASA (CRS) | Success | |
| 03/01/2013 | 15:10:00 | F9 v1.0 B0007 | CCAFS LC-40 | SpaceX CRS-2 | 677.0 | LEO (ISS) | NASA (CRS) | Success | |

# Total Payload Mass

- The total payload carried by boosters from NASA

### Task 3

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

In [10]:
```
%%sql
SELECT SUM(PAYLOAD_MASS__KG_) AS TotalPayloadMass
FROM SPACEXTBL
WHERE CUSTOMER = 'NASA (CRS)';
```

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

Out[10]:

| TotalPayloadMass |
|---|
| 45596.0 |

# Average Payload Mass by F9 v1.1

- The average payload mass carried by booster version F9 v1.1

### Task 4

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

In [11]:
```
%%sql
SELECT AVG(PAYLOAD_MASS__KG_) AS AveragePayloadMass
FROM SPACEXTBL
WHERE Booster_Version LIKE 'F9 v1.1%';
```

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

Out[11]:

| AveragePayloadMass |
| --- |
| 2534.6666666666665 |

# First Successful Ground Landing Date

the dates of the first successful landing outcome on ground pad was achieved

### Task 5

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

*Hint:Use min function*

In [14]:

```
%%sql
SELECT MIN(Date) AS FirstSuccessfulLandingDate
FROM SPACEXTBL
WHERE Landing_Outcome = 'Success'
```

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

Out[14]:

| FirstSuccessfulLandingDate |
| --- |
| 01/07/2020 |

# Successful Drone Ship Landing with Payload between 4000 and 6000

- The names of boosters which have successfully landed on drone ship and had 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 [16]:

```
%%sql
SELECT Booster_Version
FROM SPACEXTBL
WHERE Landing_Outcome = 'Success'
AND PAYLOAD_MASS__KG_ > 4000
AND PAYLOAD_MASS__KG_ < 6000;
```

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

Out[16]:

| Booster_Version |
|---|
| F9 B5 B1046.2 |
| F9 B5 B1047.2 |
| F9 B5 B1048.3 |
| F9 B5 B1051.2 |
| F9 B5B1060.1 |
| F9 B5 B1058.2 |
| F9 B5B1062.1 |

# Total Number of Successful and Failure Mission Outcomes

- the total number of successful and failure mission outcomes

### Task 7

List the total number of successful and failure mission outcomes

In [23]:

```
%%sql SELECT Mission_Outcome, COUNT(*) AS TotalMissions
FROM SPACEXTBL
GROUP BY Mission_Outcome;
```

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

Out[23]:

| Mission_Outcome | TotalMissions |
|---|---|
| None | 898 |
| Failure (in flight) | 1 |
| Success | 98 |
| Success | 1 |
| Success (payload status unclear) | 1 |

# Boosters Carried Maximum Payload

the names of the booster which have carried the maximum payload mass

### Task 8

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

In [26]:

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

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

Out[26]:

| 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

the failed landing_outcomes in drone ship, their booster versions, and launch site names for in 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, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.**

In [31]:

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

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

Out[31]:

| Month | Landing_Outcome | Booster_Version | Launch_Site |
|---|---|---|---|
| None | Failure (drone ship) | F9 v1.1 B1012 | CCAFS LC-40 |
| None | Failure (drone ship) | F9 v1.1 B1015 | CCAFS LC-40 |

# Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

- 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

### 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***

```
%%sql
SELECT landing__outcome,COUNT(landing__outcome) as OccurenceValue FROM SPACEXTBL
where SPACEXTBL.Date between '2010-06-04' and '2017-03-20'
GROUP BY landing__outcome ORDER BY OccurenceValue desc;
```

```
 * ibm_db_sa://nmf32271:***@54a2f15b-5c0f-46df-8954-7e38e612c2bd.clogj3sd0tgtu0lqde00.databases.appdomain.cloud:32733/bludb
Done.
```

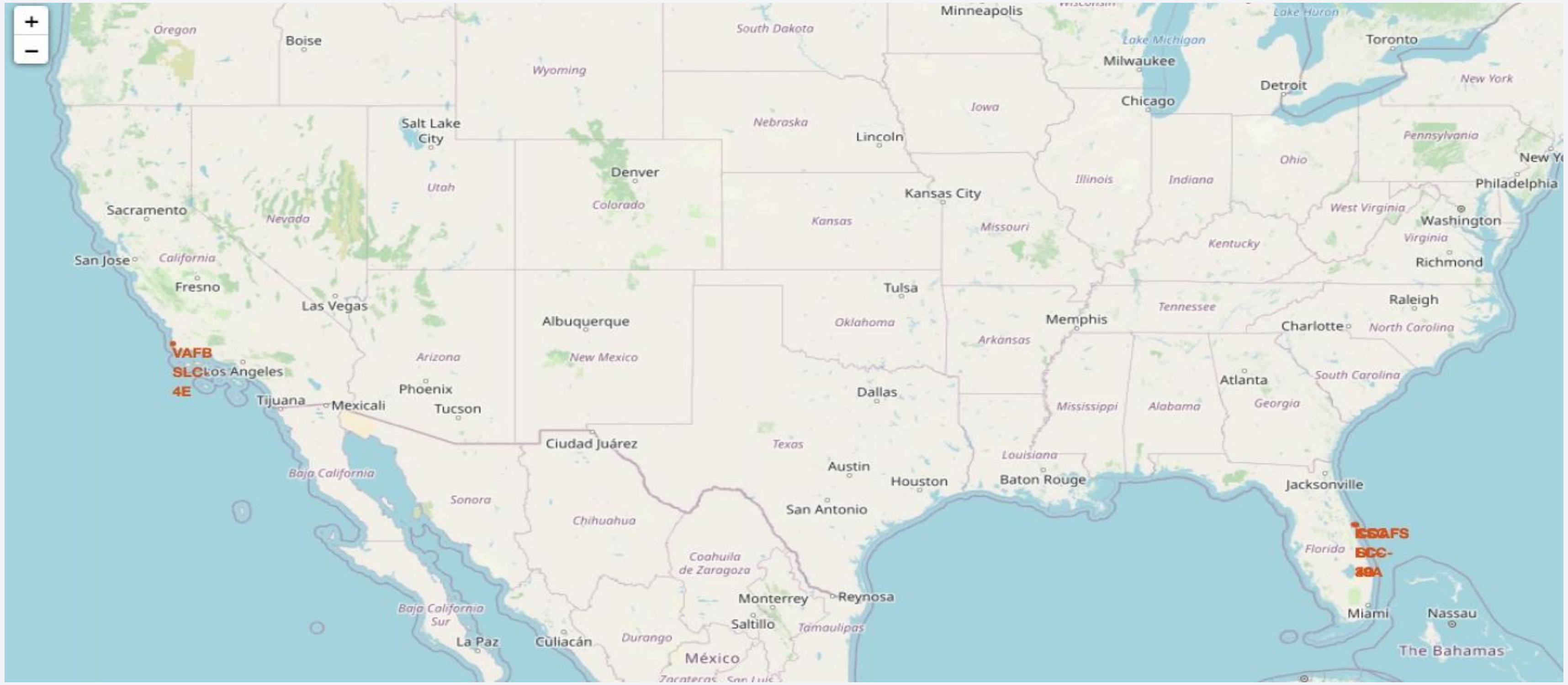
| landing__outcome | occurencevalue |
|---|---|
| No attempt | 10 |
| Failure (drone ship) | 5 |
| Success (drone ship) | 5 |
| Controlled (ocean) | 3 |
| Success (ground pad) | 3 |
| Failure (parachute) | 2 |
| Uncontrolled (ocean) | 2 |
| Precluded (drone ship) | 1 |

Section 3

# Launch Sites Proximities Analysis

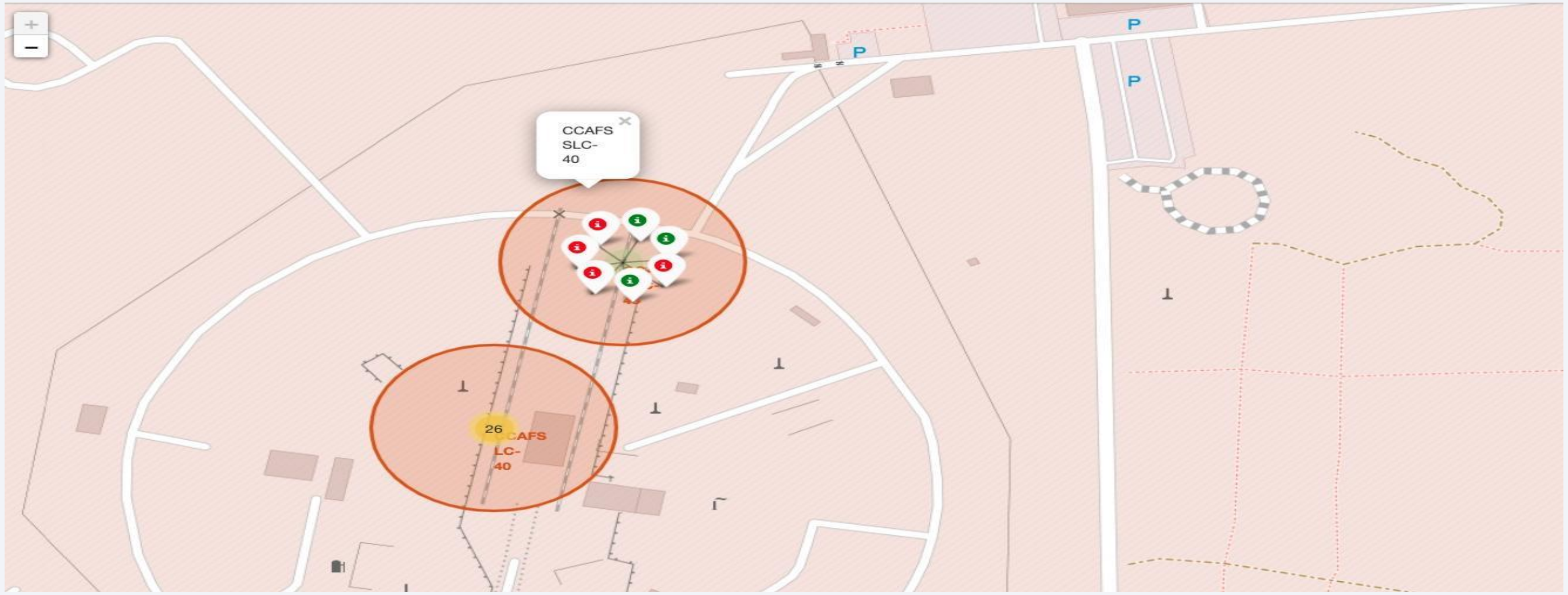
# Launch sites on a map

- Generated folium map with all launch site locations



# Color labeled Outcome

- the folium map with the color-labeled launch outcomes on the map



# Folium Map launch site and it proximities

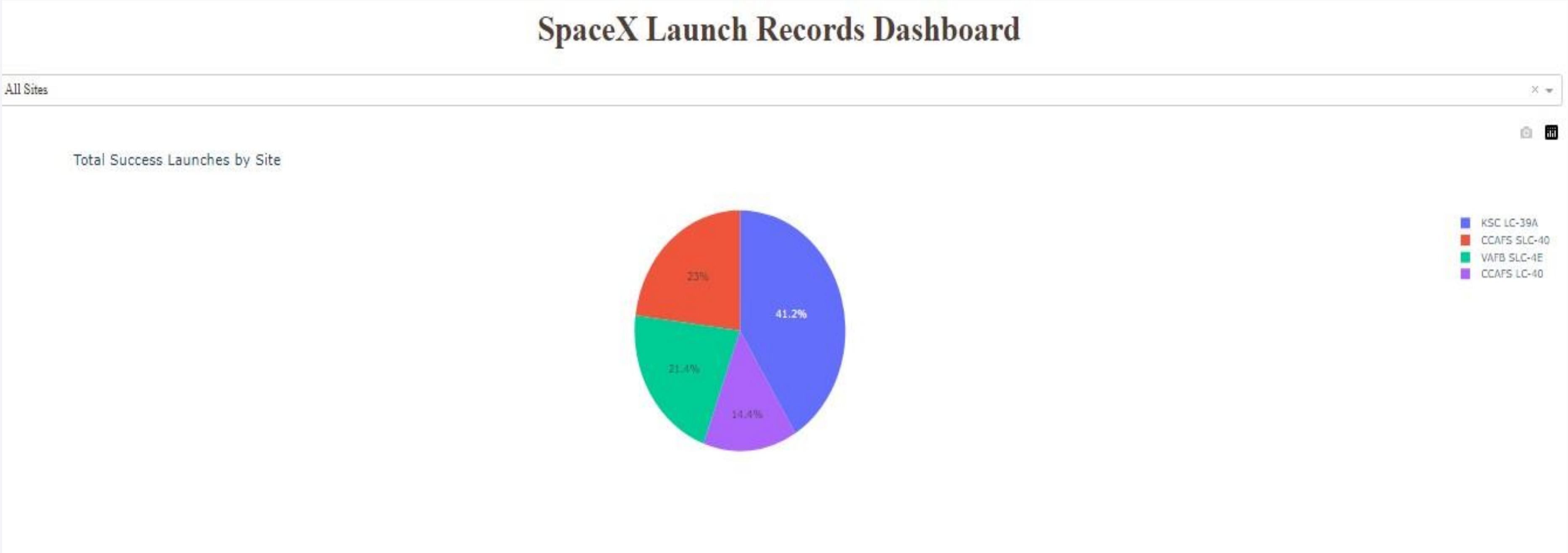
- The generated folium map and selected launch site to its proximities such as railway, highway, coastline, with distance calculated and displayed

Section 4

# Build a Dashboard with Plotly Dash

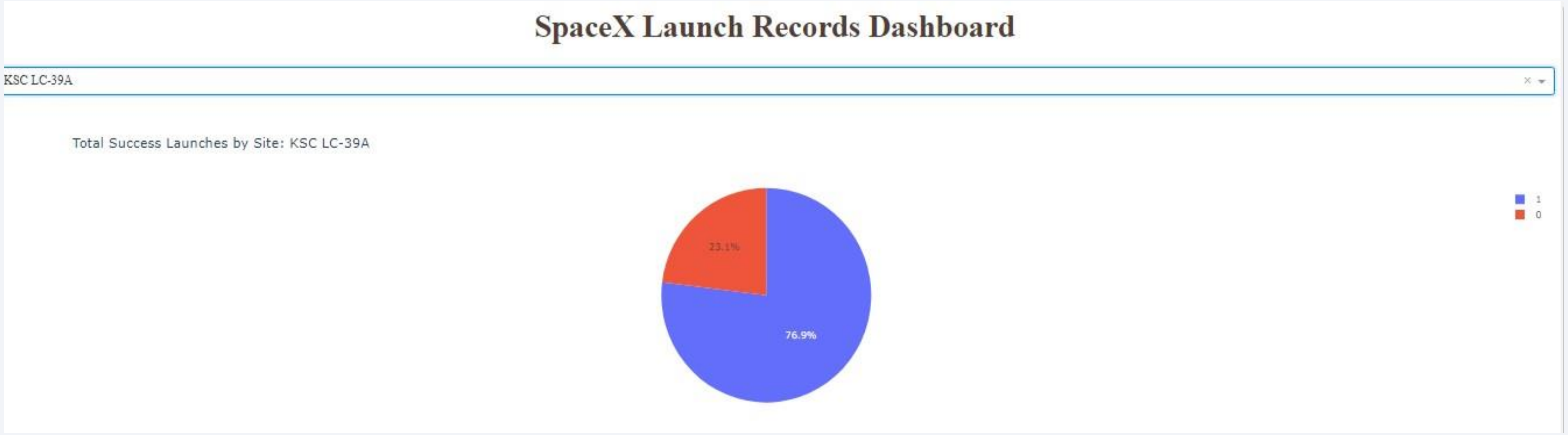
# Dashboard of launch success count for all sites

- Successful launches distributed by launch sites
- Most successful rate (41.7%) belongs to the KSC LC-39A site



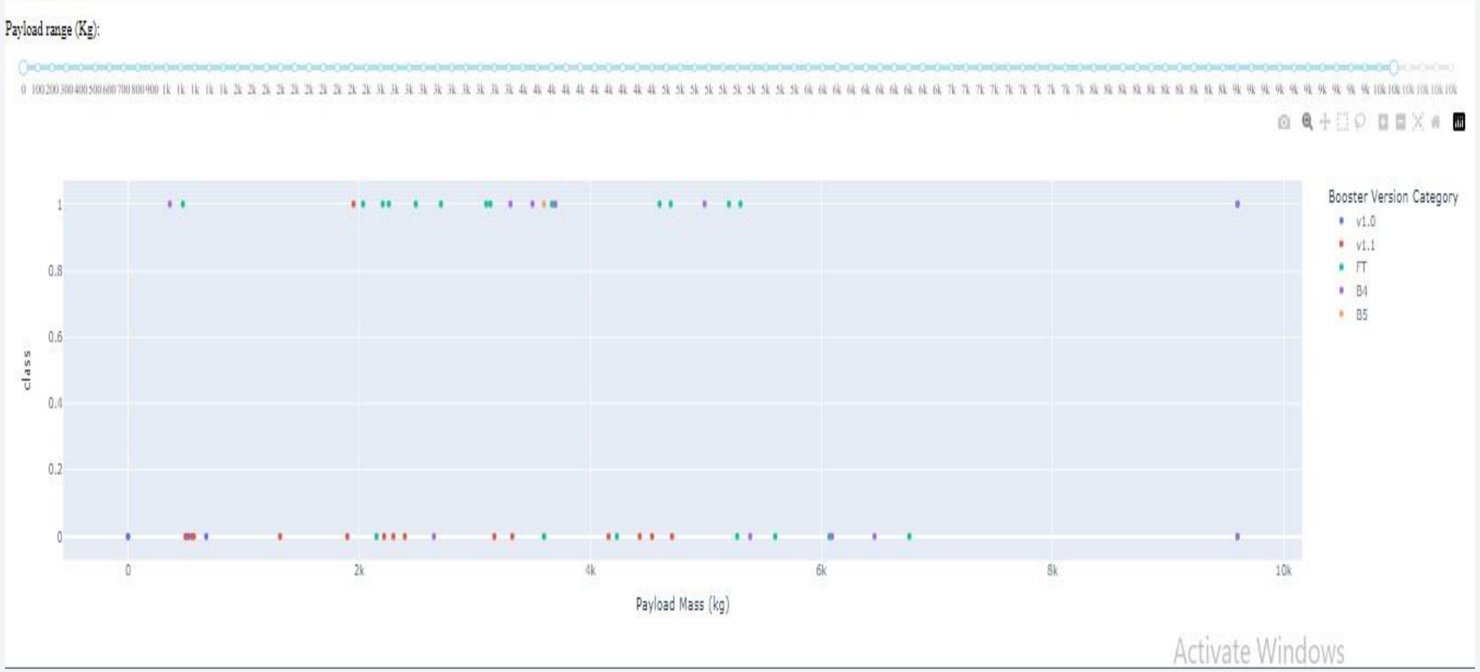
# Dashboard for Pie chart for highest success rate

- The pie chart for the launch site with highest launch success ratio
- It has a success rate of 76.9%



# Dashboard for Payload vs Launch Outcome

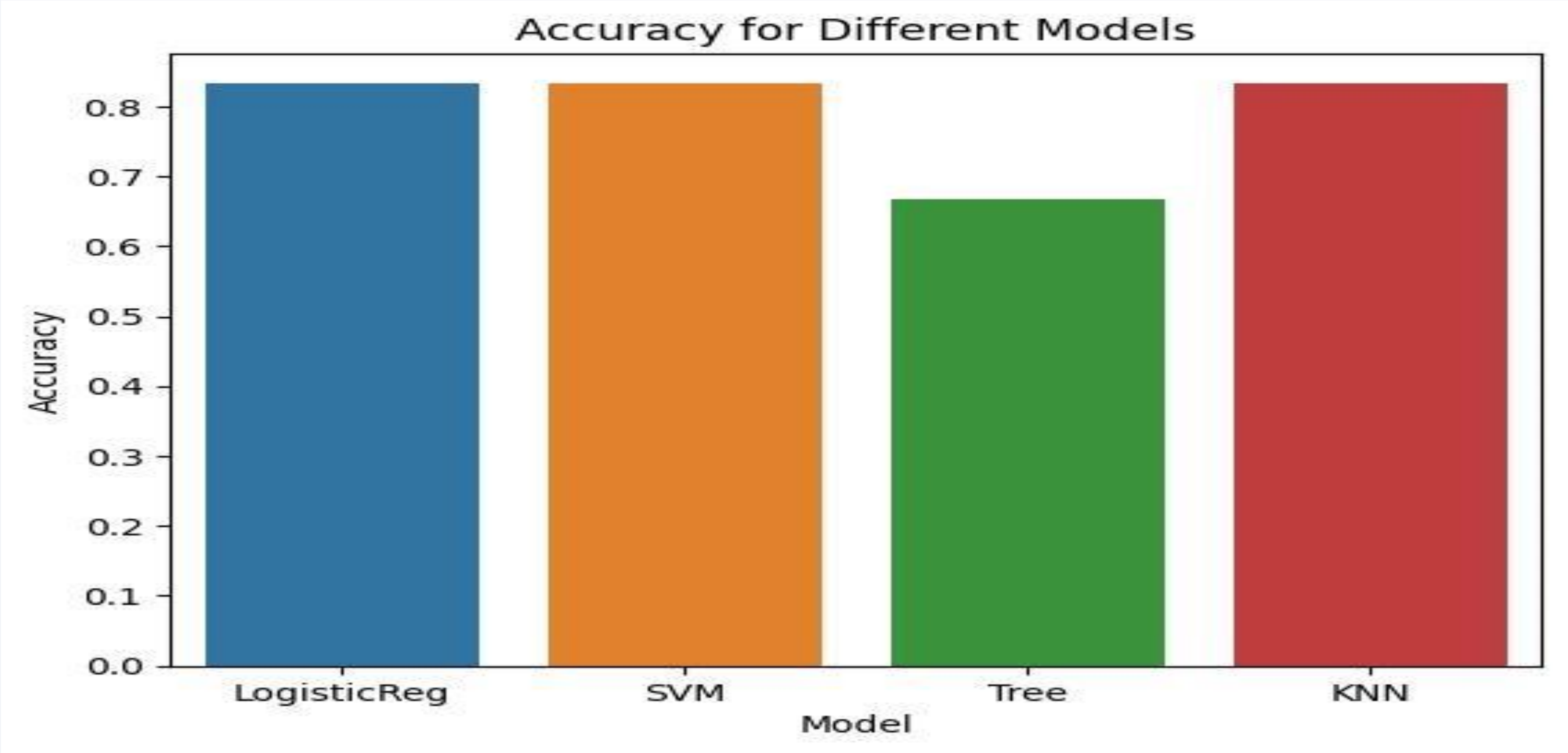
- Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider



Section 5

# Predictive Analysis (Classification)

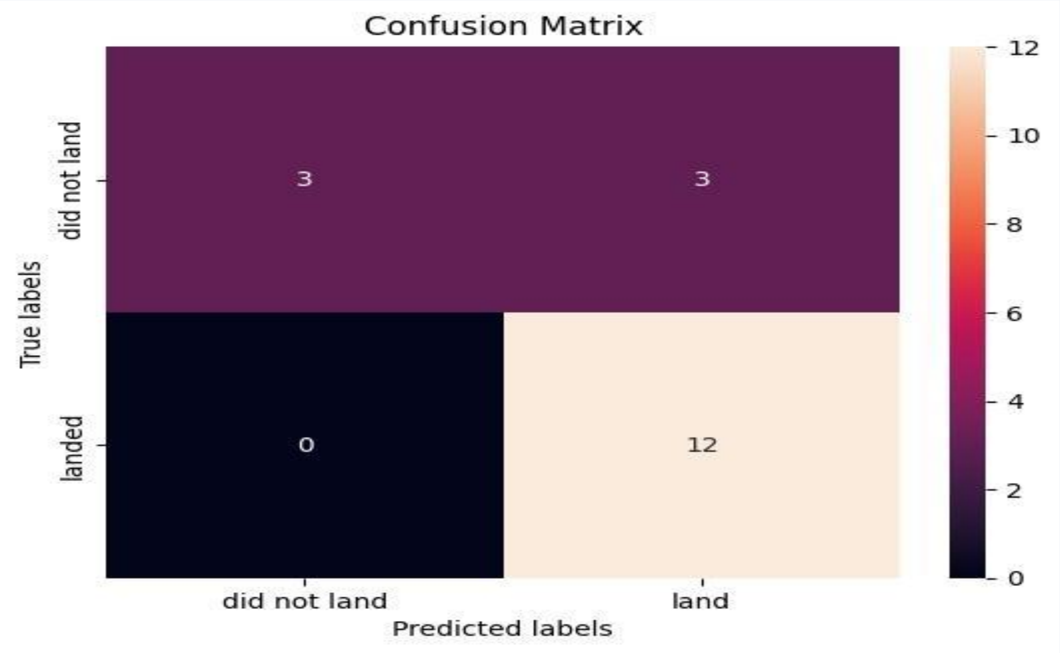
# Classification Accuracy




All models have high accuracy with the exception of Tree which has the lowest accuracy among all.

# Confusion Matrix

- All 4 models (LogReg, SVM, Tree, and KNN) show similar behavior and performance
- The Tree model has the lowest and F1 score which maybe due false positive




Out[ ]:

| | LogisticReg | SVM | Tree | KNN |
|---|---|---|---|---|
| F1_Score | 0.814815 | 0.814815 | 0.666667 | 0.814815 |
| Accuracy | 0.833333 | 0.833333 | 0.666667 | 0.833333 |

# Conclusions

- The features shows relationship when plotted against each other.
- There is an increase of success rate from 2013 till 2020
- KSC LC-39A has the highest success rate among the launching site
- All the constructed models showed similar accuracy except tree model
- The false positive part of the confusion matrix is the main source of error

# Appendix

- import pandas as pd
- import seaborn as sns
- Import matplotlib.pyplot as plt
- data = {     'Model': ['LogisticReg', 'SVM', 'Tree', 'KNN'],     'Accuracy': [0.833333, 0.833333, 0.666667, 0.833333]}
- df = pd.DataFrame(data)sns.barplot(x='Model', y='Accuracy', data=df)
- plt.title('Accuracy for Different Models’)
- plt.xlabel('Model’)
- plt.ylabel('Accuracy')plt.show()

# Thank you!