IBM Developer
SKILLS NETWORK

# Winning Space Race with Data Science


Zaw Aung

14/06/2025

# 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
  - Filtering data and dealing with missing values
  - 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
  - Building, tuning and evaluation of classification models to ensure the best results

# Data Collection – SpaceX API

1. We used the get request to the SpaceX API to collect data
2. Decoded response using **.json()** and convert into dataframe with **.json_normalize()**
3. Applied custom functions on data and made dictionary
4. Made dataframe from dictionary
5. Filtered dataframe to only have flacon 9 launches
6. Calculated mean of **PayloadMass** column
7. Replaced missing values from PayloadMass column with mean
8. Exported dataframe to CSV

- The link to the notebook is [Data Collection]

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 – Web Scraping

1. Used **get request** to obtain Falcon 9 launch data from Wikipedia
2. Created **BeautifulSoup** object from HTML response
3. Extracted all column names from HTML table header
4. Collected all data by parsing HTML tables and into a dictionary
5. Create dataframe from dictionary
6. Export dataframe to CSV

- The link to the notebook is [Web Scraping]

1. Apply HTTP Get method to request the Falcon 9 rocket launch page

In [4]:
```python
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

In [5]:
```python
# 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]:
```python
# 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]:
```python
# 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]:
```python
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

- Perform Exploratory Data Analysis and determine Training Labels Calculated the number of launches on each site
- Calculated the number and occurrence of each orbit
- Calculated the number and occurrence of mission outcome per orbit type
- Created a landing outcome label from Outcome column
- Exported the dataframe to CSV
- The link to the notebook is [Data Wrangling]()

```
# Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()
```

```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

```
# Apply value_counts on Orbit colum
df['Orbit'].value_counts()
```

```
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
SO        1
ES-L1     1
GEO       1
HEO       1
Name: Orbit, dtype: int64
```

```
# landing_outcomes = values on Outcome column
landing_outcomes=df['Outcome'].value_counts()
landing_outcomes
```

```
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
None ASDS       2
False Ocean     2
False RTLS      1
Name: Outcome, dtype: int64
```

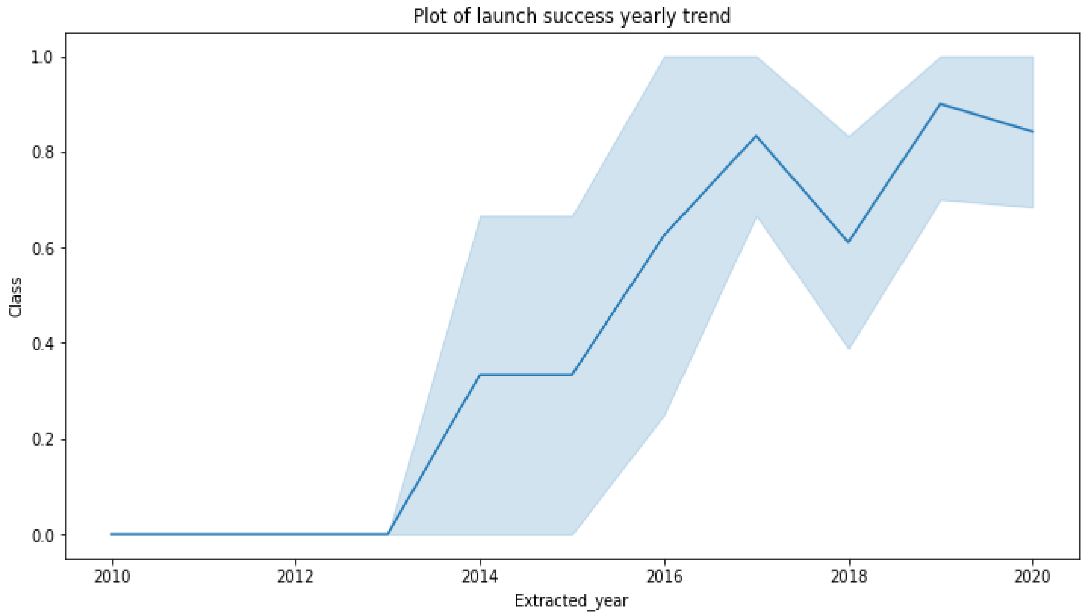
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for key, value in df['Outcome'].items():
    if value in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

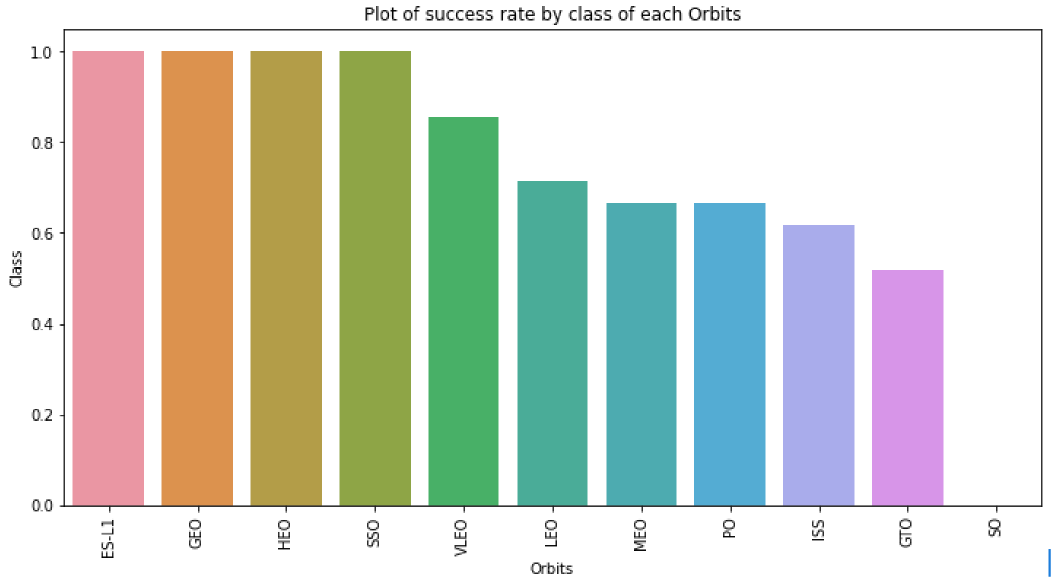
This variable will represent the classification va
land successfully; one means the first stage lan

```
df['Class']=landing_class
df[['Class']].head(8)
```

# EDA with Data Visualization

- Charts were plotted:
- Flight Number vs. Payload Mass, Flight Number vs. Launch Site, Payload Mass vs. Launch Site, Orbit Type vs. Success Rate, Flight Number vs. Orbit Type,Payload Mass vs Orbit Type and Success Rate Yearly Trend
- Scatter plots show the relationship between variables. If a relationship exists, they could be used in machine learning model.
- Bar charts show comparisons among discrete categories. The goal is to show the relationship between the specific categories being compared and a measured value.
- Line charts show trends in data over time (time series).
- Link to the notebook [Data Visualisation]



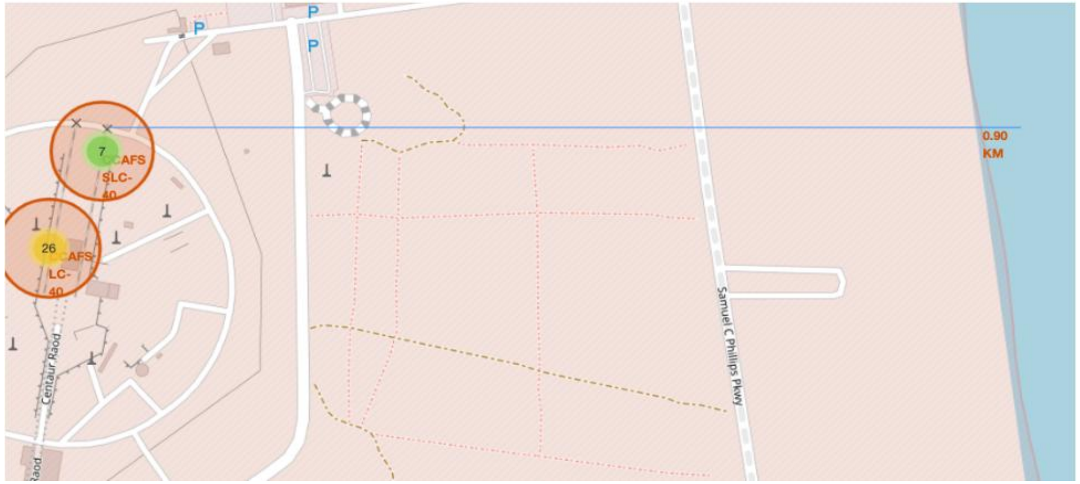
# EDA with SQL

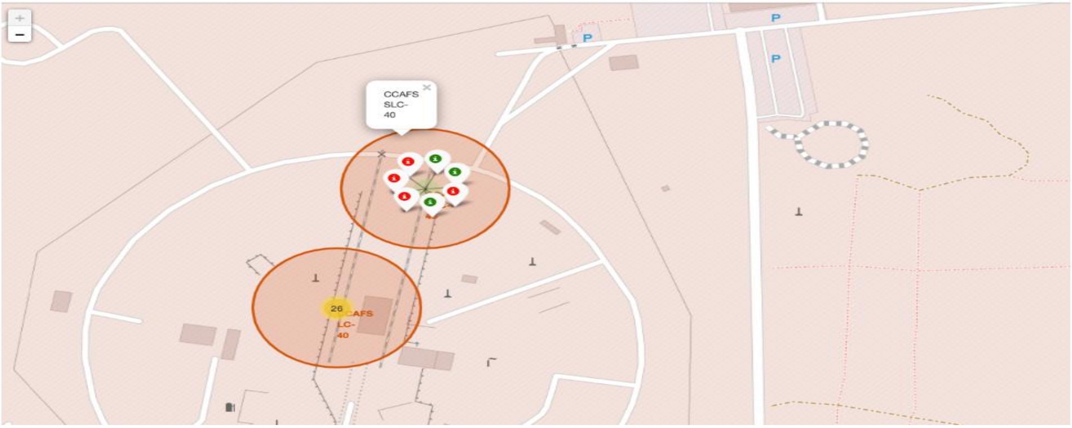
- We loaded the SpaceX dataset into a PostgreSQL database without leaving the jupyter notebook.
- We applied EDA with SQL to get insight from the data.
  - Displaying the names of the unique launch sites in the space mission
  - Displaying 5 records where launch sites begin with the string 'CCA'
  - Displaying the total payload mass carried by boosters launched by NASA (CRS)
  - Displaying average payload mass carried by booster version F9 v1.1
  - Listing the date when the first successful landing outcome in ground pad was achieved
  - Listing the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
  - Listing the total number of successful and failure mission outcomes
  - Listing the names of the booster versions which have carried the maximum payload mass
  - Listing the failed landing outcomes in drone ship, their booster versions and launch site names for the months in year 2015
  - Ranking 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.

- The link to the notebook is [EDA with SQL]

# 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.
- Link to the notebook [Interactive map with Folium]()



# Build a Dashboard with Plotly Dash

- Launch Sites Dropdown List: -

  Added a dropdown list to enable Launch Site selection.

- Pie Chart showing Success Launches (All Sites/Certain Site): -

  Added a pie chart to show the total successful launches count for all sites and the Success vs. Failed counts for the site, if a specific Launch Site was selected.

- Slider of Payload Mass Range: -

  Added a slider to select Payload range.

- Scatter Chart of Payload Mass vs. Success Rate for the different Booster Versions: -#

  Added a scatter chart to show the correlation between Payload and Launch Success.

Link to the notebook [Link to Plotly Dash App]

# Predictive Analysis (Classification)

1. Created a NumPy array from the column “Class” in data
2. Standardised the data with **StandardScaler**, then fitting and transforming it
3. Splitting the data into training and testing sets with **train_test_split** function
4. Created a **GridSearchCV** object with cv = 10 to find the best parameters
5. Applied **GridSearchCV on LogReg, SVM, Decision Tree, and KNN models**
6. Calculating the accuracy on the test data using the method **.score()** for all models
7. Examining the confusion matrix for all models
8. Finding the method performs best by examining the Jaccard_score and F1_score metrics

- The link to the notebook is [Machine Learning Prediction]

# Results

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

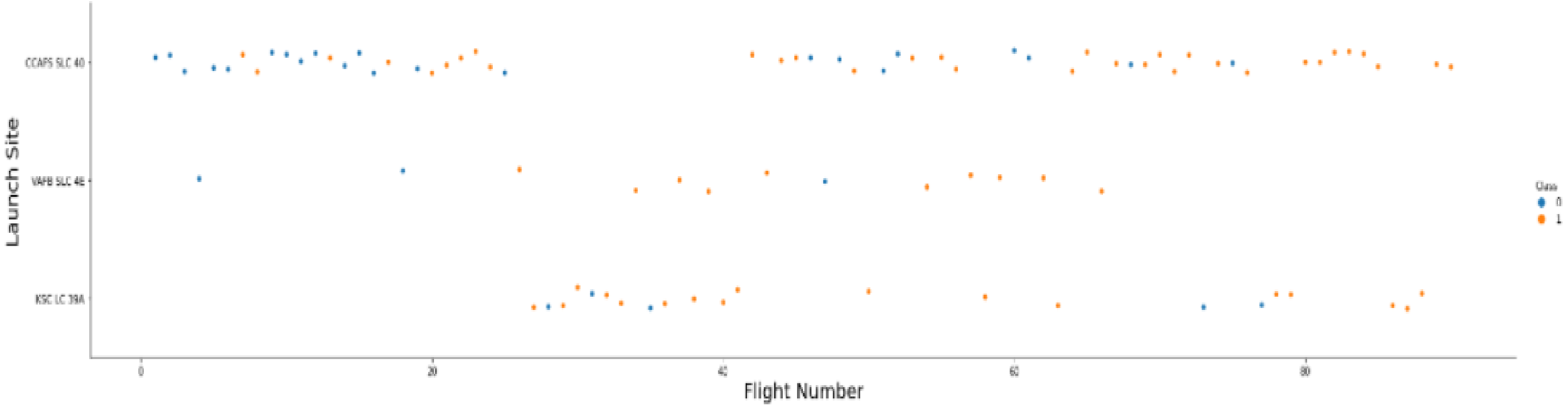
Section 2

# Insights drawn from EDA

# EDA with Visualization

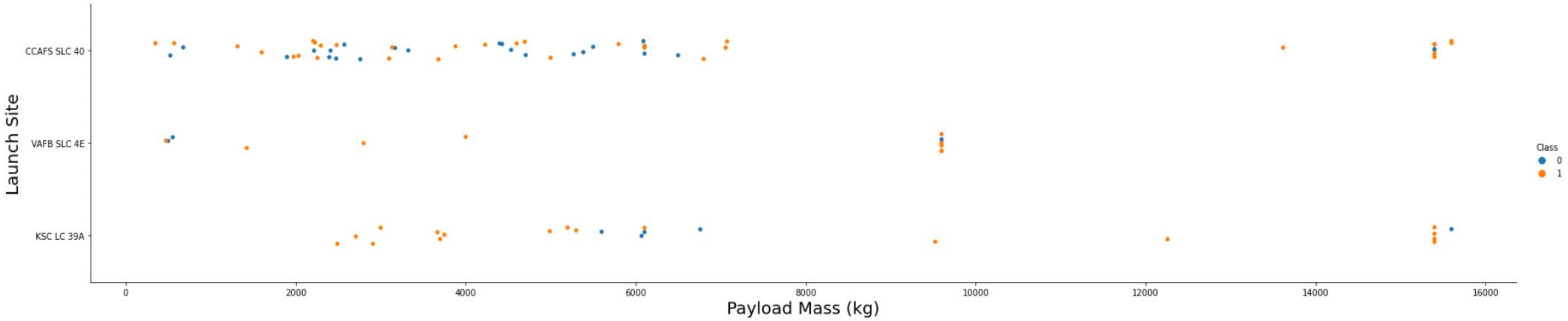
# Flight Number vs. Launch Site

- • The earliest flights all failed while the latest flights all succeeded.
- • The CCAFS SLC 40 launch site has about a half of all launches.
- • VAFB SLC 4E and KSC LC 39A have higher success rates.
- • It can be assumed that each new launch has a higher rate of success.



# Payload vs. Launch Site

- For every launch site the higher the payload mass, the higher the success rate.
- Most of the launches with payload mass over 7000 kg were successful.
- KSC LC 39A has a 100% success rate for payload mass under 5500 kg too.

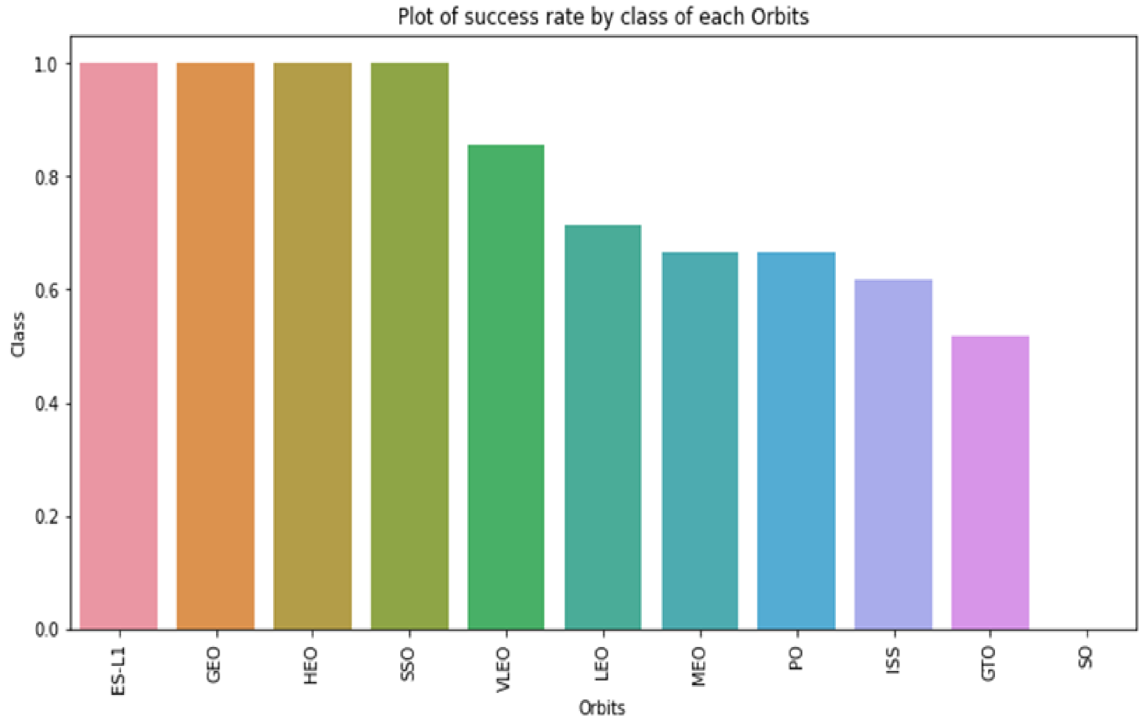


# Success Rate vs. Orbit Type

- Orbits with 100% success rate: - ES-L1, GEO, HEO, SSO

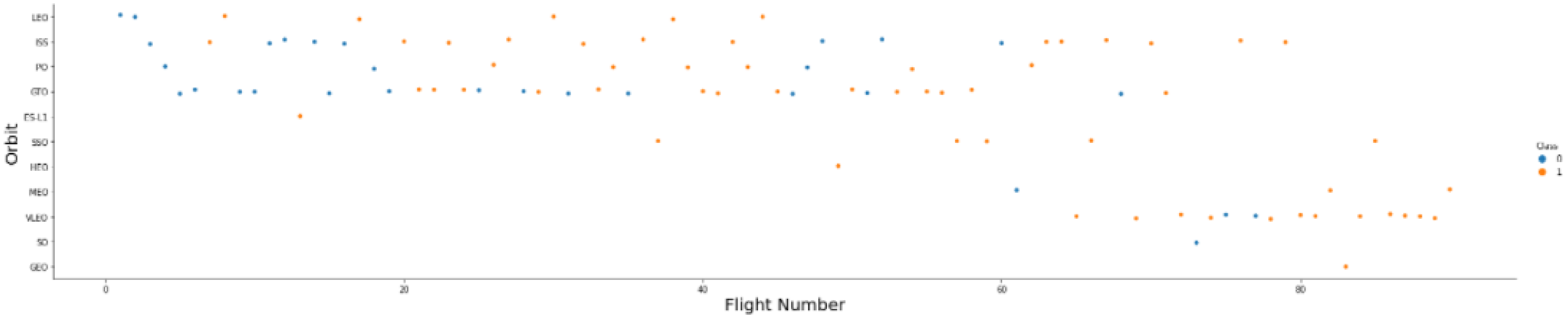
- Orbits with 0% success rate: - SO

- Orbits with success rate between 50% and 85%: - GTO, ISS, LEO, MEO, PO



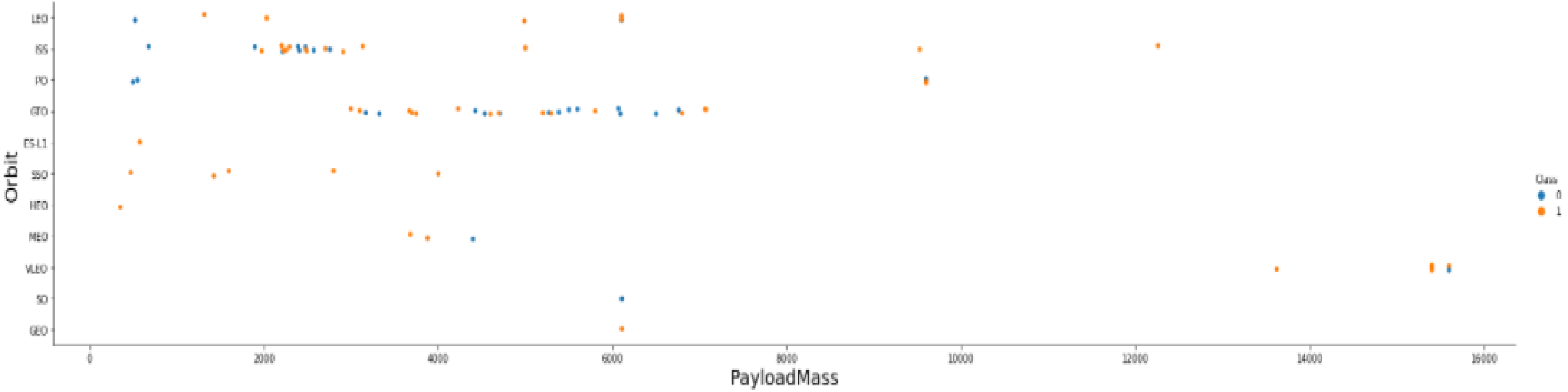
# Flight Number vs. Orbit Type

- 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.



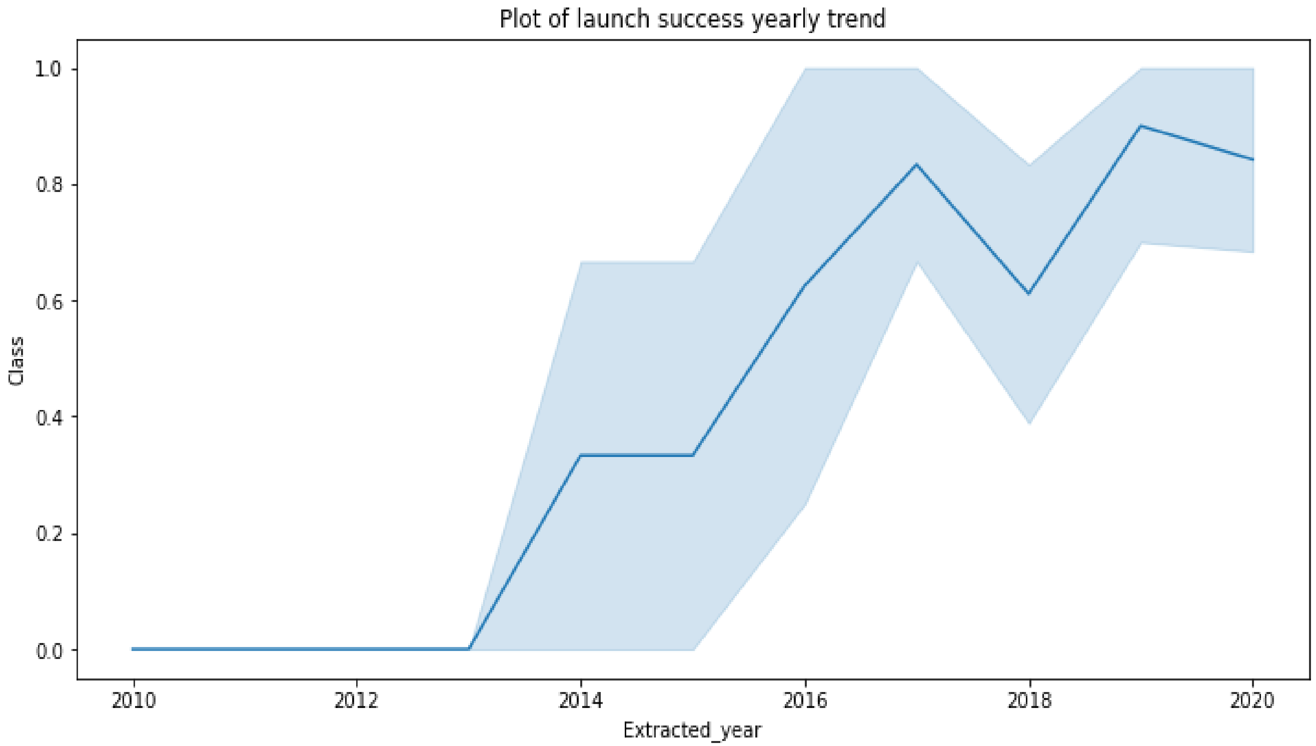
# Payload Mass vs. Orbit Type

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



# Launch Success Yearly Trend

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



# EDA with SQL

# All Launch Site Names

- Displaying the names of the unique launch sites in the space mission.

- We 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

In [10]:

```
task_1 = '''
        SELECT DISTINCT LaunchSite
        FROM SpaceX
'''
create_pandas_df(task_1, database=conn)
```

Out[10]:

| | launchsite |
|---|---|
| 0 | KSC LC-39A |
| 1 | CCAFS LC-40 |
| 2 | CCAFS SLC-40 |
| 3 | VAFB SLC-4E |

# Launch Site Names Begin with 'CCA'

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

In [11]:
```
task_2 = '''
        SELECT *
        FROM SpaceX
        WHERE LaunchSite LIKE 'CCA%'
        LIMIT 5
        '''
create_pandas_df(task_2, database=conn)
```

Out[11]:

| | date | time | boosterversion | launchsite | payload | payloadmasskg | orbit | customer | missionoutcome | landingoutcome |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2010-04-06 | 18:45:00 | F9 v1.0 B0003 | CCAFS LC-40 | Dragon Spacecraft Qualification Unit | 0 | LEO | SpaceX | Success | Failure (parachute) |
| 1 | 2010-08-12 | 15:43:00 | F9 v1.0 B0004 | CCAFS LC-40 | Dragon demo flight C1, two CubeSats, barrel of... | 0 | LEO (ISS) | NASA (COTS) NRO | Success | Failure (parachute) |
| 2 | 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 |
| 3 | 2012-08-10 | 00:35:00 | F9 v1.0 B0006 | CCAFS LC-40 | SpaceX CRS-1 | 500 | LEO (ISS) | NASA (CRS) | Success | No attempt |
| 4 | 2013-01-03 | 15:10:00 | F9 v1.0 B0007 | CCAFS LC-40 | SpaceX CRS-2 | 677 | LEO (ISS) | NASA (CRS) | Success | No attempt |

- Used the query above to display 5 records where launch sites begin with `CCA`

# Total Payload Mass

- Displaying the total payload mass carried by boosters launched by NASA (CRS).
- The total payload carried by boosters from NASA calculated as 45596 using the query below

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

```
In [12]:
task_3 = '''
        SELECT SUM(PayloadMassKG) AS Total_PayloadMass
        FROM SpaceX
        WHERE Customer LIKE 'NASA (CRS)'
        '''
create_pandas_df(task_3, database=conn)
```

Out[12]:

| | total_payloadmass |
|---|---|
| 0 | 45596 |

# Average Payload Mass by F9 v1.1

- Calculated the average payload mass carried by booster version F9 v1.1 as 2534

```
In [7]: %sql select avg(payload_mass__kg_) as average_payload_mass from SPACEXDATASET where booster_version like '%F9 v1.1%';

 * ibm_db_sa://wzf08322:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

Out[7]:

| average_payload_mass |
|---|
| 2534 |

# First Successful Ground Landing Date

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

In [14]:
```
task_5 = '''
        SELECT MIN(Date) AS FirstSuccessfull_landing_date
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Success (ground pad)'
        '''
create_pandas_df(task_5, database=conn)
```

Out[14]:

| | firstsuccessfull_landing_date |
|---|---|
| 0 | 2015-12-22 |

# Successful Drone Ship Landing with Payload between 4000 and 6000

In [15]:

```
task_6 = '''
        SELECT BoosterVersion
        FROM SpaceX
        WHERE LandingOutcome = 'Success (drone ship)'
            AND PayloadMassKG > 4000
            AND PayloadMassKG < 6000
        '''
create_pandas_df(task_6, database=conn)
```

Out[15]:

| | boosterversion |
|---|---|
| 0 | F9 FT B1022 |
| 1 | F9 FT B1026 |
| 2 | F9 FT B1021.2 |
| 3 | F9 FT B1031.2 |

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

- 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

- Used wildcard **LIKE** ‘%’ to filter for **WHERE** MissionOutcome was a success or a failure.

List the total number of successful and failure mission outcomes

In [16]:
```
task_7a = '''
        SELECT COUNT(MissionOutcome) AS SuccessOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Success%'
        '''

task_7b = '''
        SELECT COUNT(MissionOutcome) AS FailureOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Failure%'
        '''

print('The total number of successful mission outcome is:')
display(create_pandas_df(task_7a, database=conn))
print()
print('The total number of failed mission outcome is:')
create_pandas_df(task_7b, database=conn)
```

```
The total number of successful mission outcome is:
```

| | successoutcome |
|---|---|
| **0** | 100 |

```
The total number of failed mission outcome is:
```

Out[16]:

| | failureoutcome |
|---|---|
| **0** | 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.

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

In [17]:
```
task_8 = '''
        SELECT BoosterVersion, PayloadMassKG
        FROM SpaceX
        WHERE PayloadMassKG = (
                                SELECT MAX(PayloadMassKG)
                                FROM SpaceX
                                )
        ORDER BY BoosterVersion
        '''
create_pandas_df(task_8, database=conn)
```

Out[17]:

| | boosterversion | payloadmasskg |
|---|---|---|
| 0 | F9 B5 B1048.4 | 15600 |
| 1 | F9 B5 B1048.5 | 15600 |
| 2 | F9 B5 B1049.4 | 15600 |
| 3 | F9 B5 B1049.5 | 15600 |
| 4 | F9 B5 B1049.7 | 15600 |
| 5 | F9 B5 B1051.3 | 15600 |
| 6 | F9 B5 B1051.4 | 15600 |
| 7 | F9 B5 B1051.6 | 15600 |
| 8 | F9 B5 B1056.4 | 15600 |
| 9 | F9 B5 B1058.3 | 15600 |
| 10 | F9 B5 B1060.2 | 15600 |
| 11 | F9 B5 B1060.3 | 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

List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

In [18]:
```
task_9 = '''
        SELECT BoosterVersion, LaunchSite, LandingOutcome
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Failure (drone ship)'
            AND Date BETWEEN '2015-01-01' AND '2015-12-31'
        '''
create_pandas_df(task_9, database=conn)
```

Out[18]:

| | boosterversion | launchsite | landingoutcome |
|---|---|---|---|
| 0 | F9 v1.1 B1012 | CCAFS LC-40 | Failure (drone ship) |
| 1 | F9 v1.1 B1015 | CCAFS LC-40 | Failure (drone ship) |

# Rank success count between 2010-06-04 and 2017-03-20

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad))

In [19]:

```
task_10 = '''
        SELECT LandingOutcome, COUNT(LandingOutcome)
        FROM SpaceX
        WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
        GROUP BY LandingOutcome
        ORDER BY COUNT(LandingOutcome) DESC
        '''
create_pandas_df(task_10, database=conn)
```

Out[19]:

| | landingoutcome | count |
|---|---|---|
| 0 | No attempt | 10 |
| 1 | Success (drone ship) | 6 |
| 2 | Failure (drone ship) | 5 |
| 3 | Success (ground pad) | 5 |
| 4 | Controlled (ocean) | 3 |
| 5 | Uncontrolled (ocean) | 2 |
| 6 | Precluded (drone ship) | 1 |
| 7 | Failure (parachute) | 1 |

- 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.
- 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 3

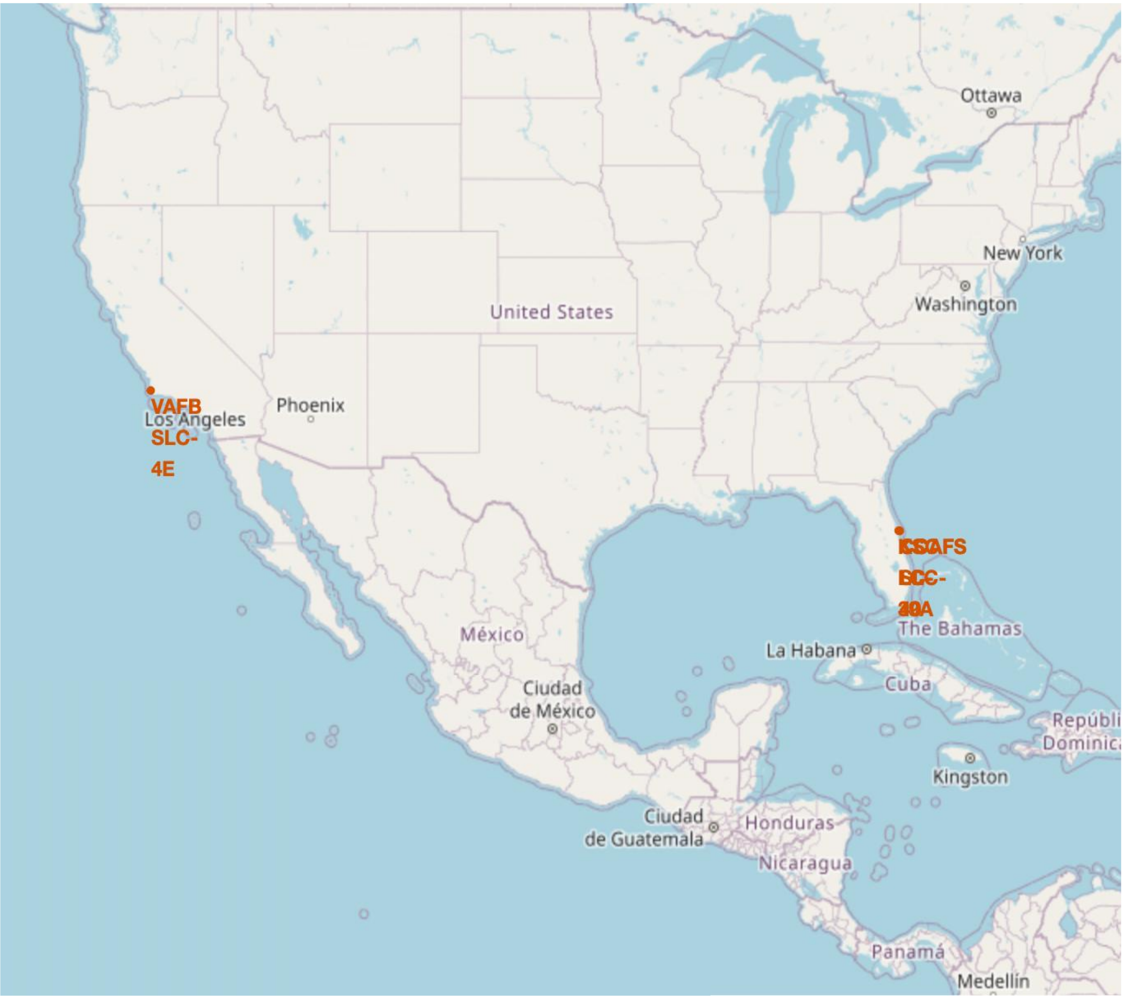
# Launch Sites Proximities Analysis

# Interactive map with Folium

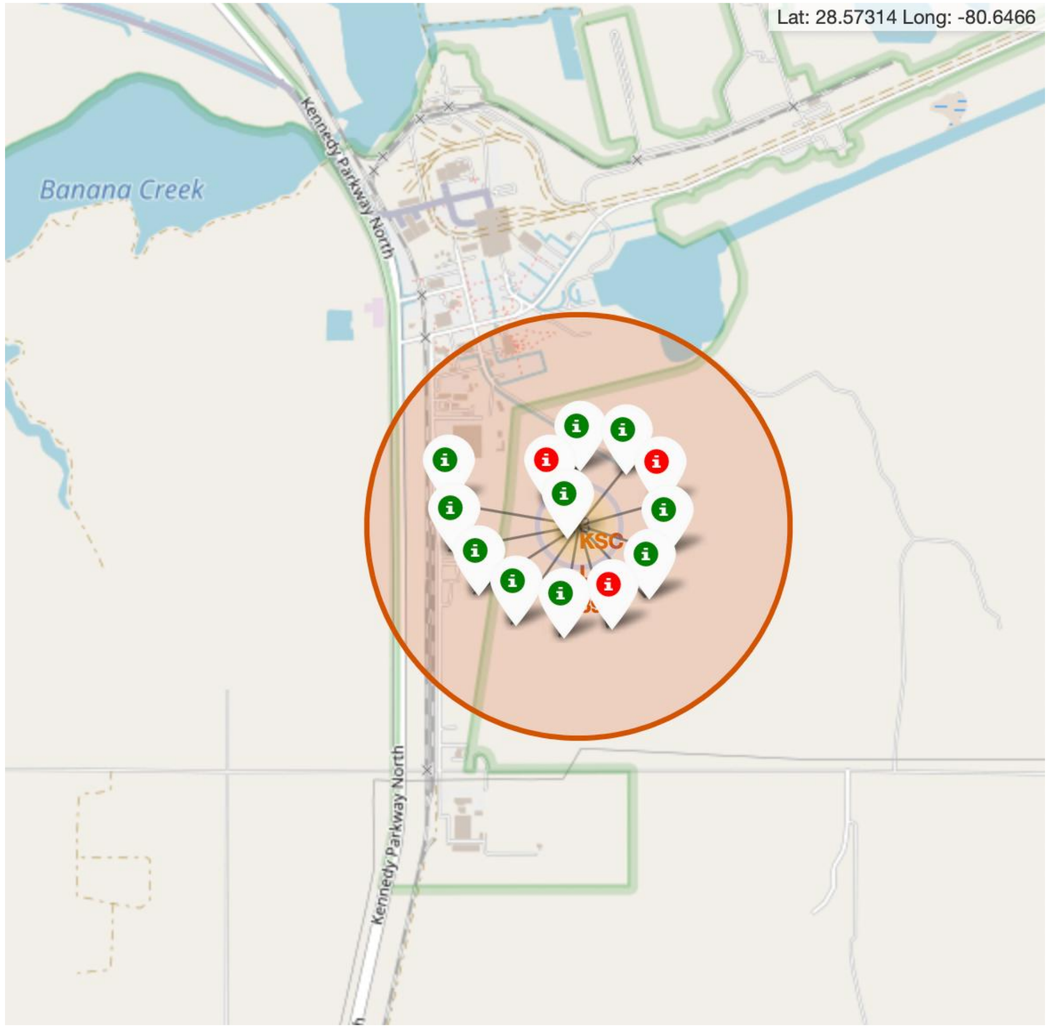
# All launch sites global map markers

• Most of Launch sites are in proximity to the Equator line. The land is moving faster at the equator than any other place on the surface of the Earth. Anything on the surface of the Earth at the equator is already moving at 1670 km/hour. This speed will help the spacecraft keep up a good enough speed to stay in orbit.

• All launch sites are in very close proximity to the coast, while launching rockets towards the ocean it minimises the risk of having any debris dropping or exploding near people.



# Markers showing launch sites with color labels

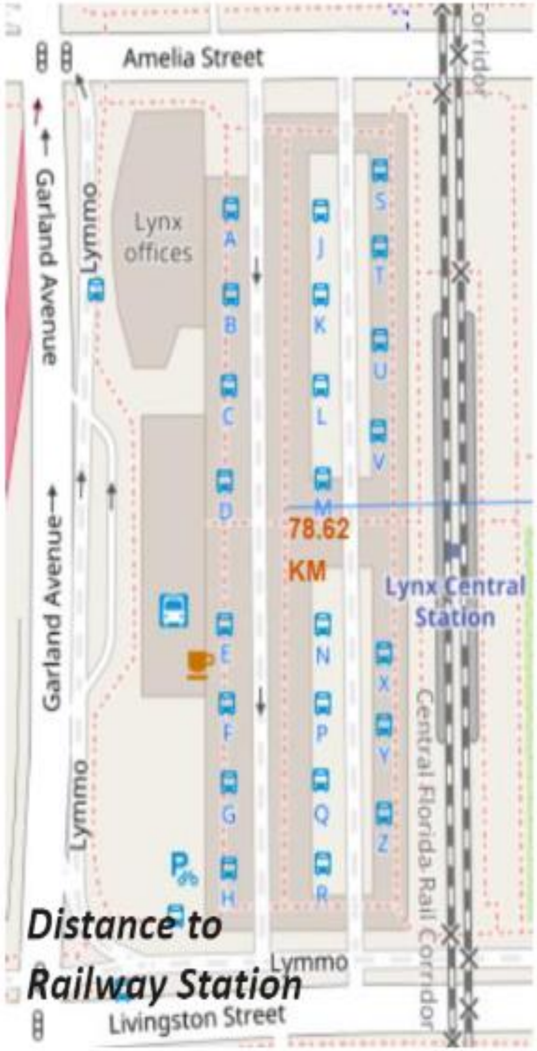



- From the colour-labeled markers we should be able to easily identify which launch sites have relatively high success rates.

  - Green Marker = Successful Launch
  - Red Marker = Failed Launch

- Launch Site KSC LC-39A has a very high Success Rate.

# Launch Site distance to landmarks







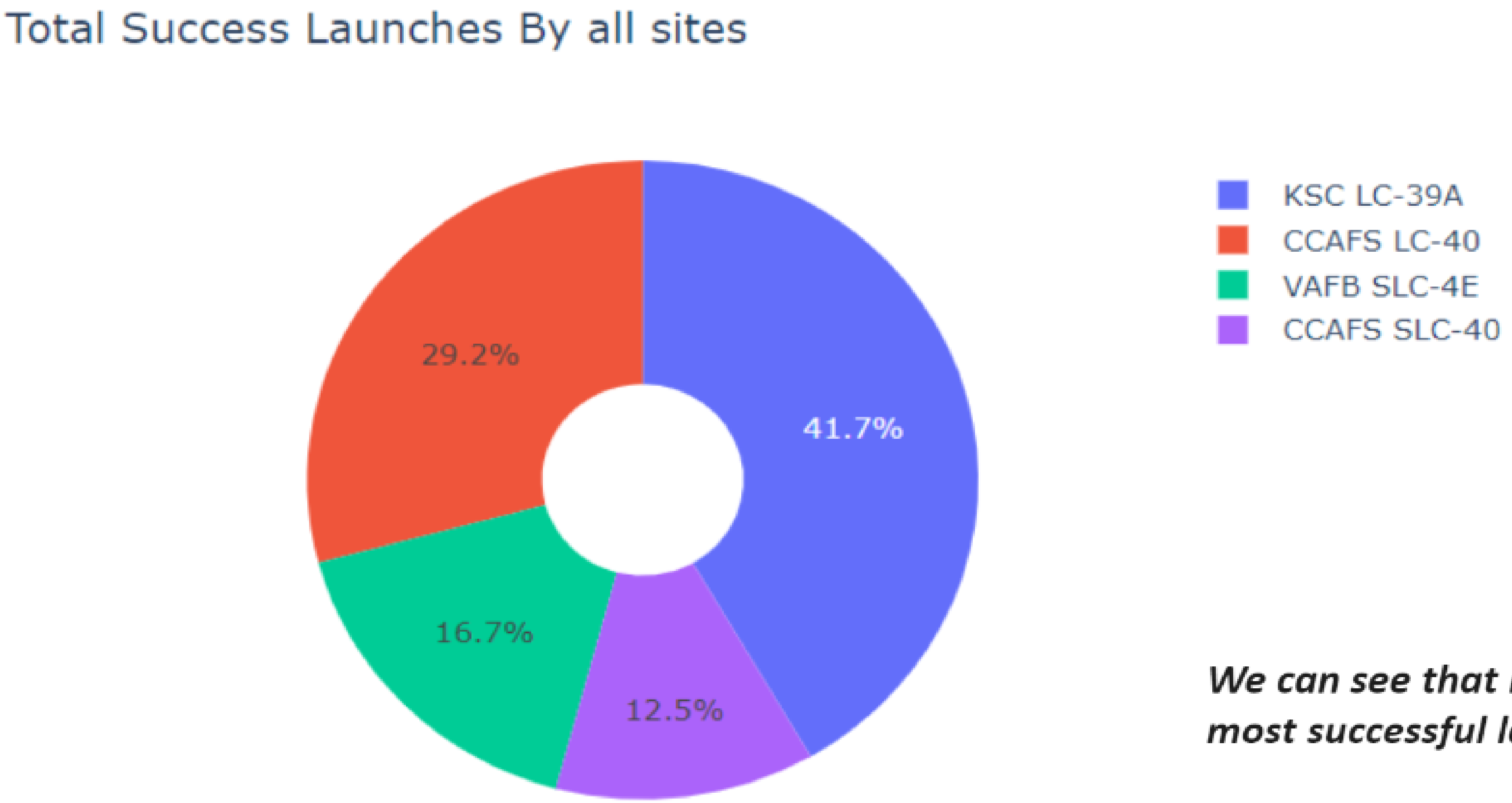

•Are launch sites in close proximity to railways? No
•Are launch sites in close proximity to highways? No
•Are launch sites in close proximity to coastline? Yes
•Do launch sites keep certain distance away from cities? Yes

Section 4

# Build a Dashboard with Plotly Dash

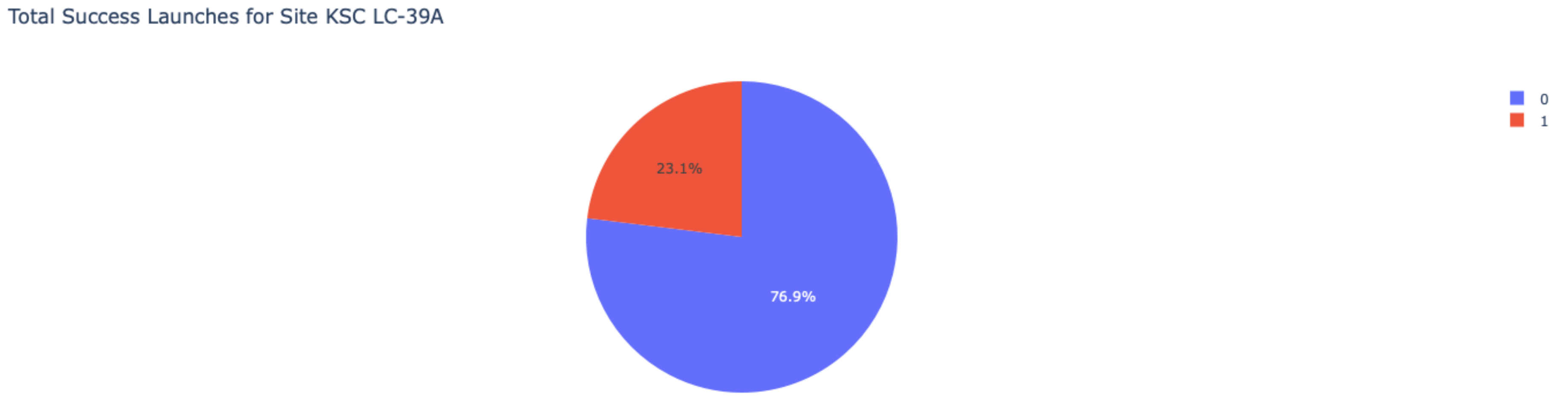
# Build a Dashboard with Plotly Dash

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



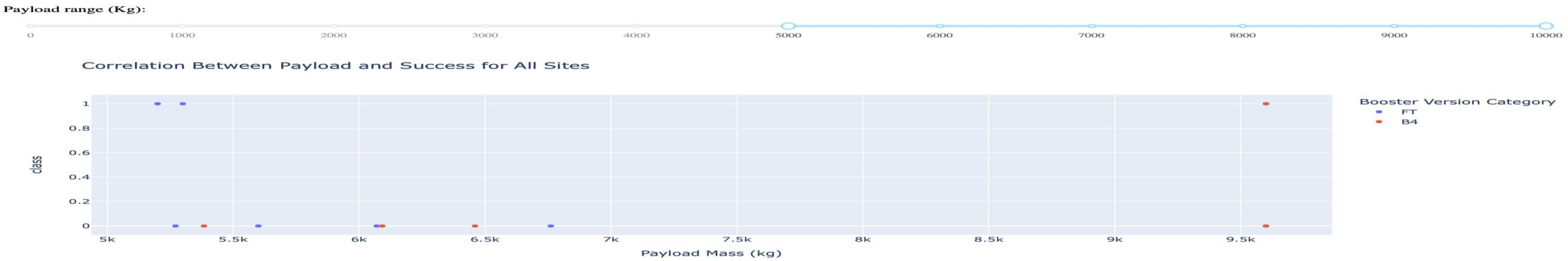

***We can see that KSC LC-39A had the most successful launches from all the sites***

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




KSC LC-39A has the highest launch success rate (76.9%) with 10 successful and only 3 failed landings.

# Payload Mass vs. Launch Outcome for all sites


The charts show that payloads between 2000 and 5500 kg have the highest success rate.

Section 5

# Predictive Analysis (Classification)

# Predictive analysis (Classification)

# Classification Accuracy

- Based on the scores of the Test Set, we can not confirm which method performs best.

- Same Test Set scores may be due to the small test sample size (18 samples). Therefore, we tested all methods based on the whole Dataset.

- The scores of the whole Dataset confirm that the best model is the Decision Tree Model. This model has not only higher scores, but also the highest accuracy.

Scores and Accuracy of the Test Set

| | LogReg | SVM | Tree | KNN |
|---|---|---|---|---|
| **Jaccard_Score** | 0.800000 | 0.800000 | 0.800000 | 0.800000 |
| **F1_Score** | 0.888889 | 0.888889 | 0.888889 | 0.888889 |
| **Accuracy** | 0.833333 | 0.833333 | 0.833333 | 0.833333 |

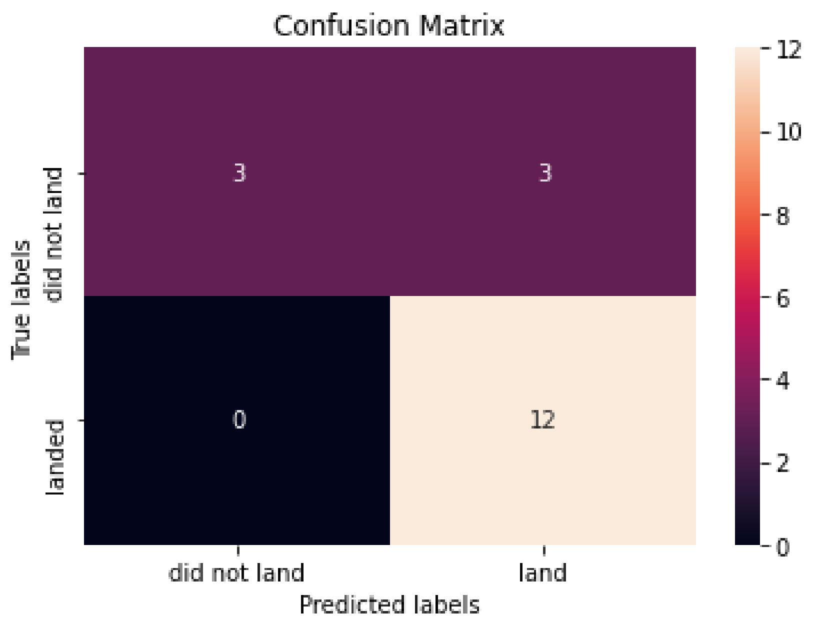
Scores and Accuracy of the Entire Data Set

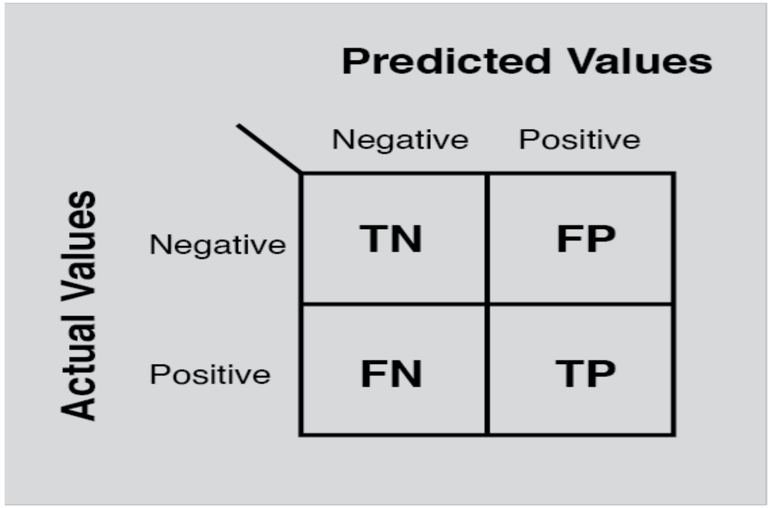
| | LogReg | SVM | Tree | KNN |
|---|---|---|---|---|
| **Jaccard_Score** | 0.833333 | 0.845070 | 0.882353 | 0.819444 |
| **F1_Score** | 0.909091 | 0.916031 | 0.937500 | 0.900763 |
| **Accuracy** | 0.866667 | 0.877778 | 0.911111 | 0.855556 |

# 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

- Decision Tree Model is the best algorithm for this dataset.
- Launches with a low payload mass show better results than launches with a larger payload mass.
- Most of launch sites are in proximity to the Equator line and all the sites are in very close proximity to the coast.
- The success rate of launches increases over the years.
- KSC LC-39A has the highest success rate of the launches from all the sites.
- Orbits ES-L1, GEO, HEO and SSO have 100% success rate.

# Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

# Thank you!