IBM Developer
SKILLS NETWORK

# Winning Space Race with Data Science

Kelvin Mudita
16/10/2021

# Outline

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

# Executive Summary

## Summary of methodologies

- Data collection
- Data wrangling
- EDA with data visualization
- EDA with SQL
- Building an interactive map with Folium
- Building a Dashboard with Plotly Dash
- Predictive analysis (Classification)

## Summary of all results

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

# Introduction

- Project background and context

- The aim of the project is to determine if the Falcon 9 first stage will land successfully.
- The cost of launching the SpaceX Falcon 9 rocket is 62  million dollars.
- Alternative providers cost more than165 million dollars to launch.
- The results of  this project are usefull to alternative companies  who wants to bid against SpaceX for a rocket launch.

- Problems you want to find answers

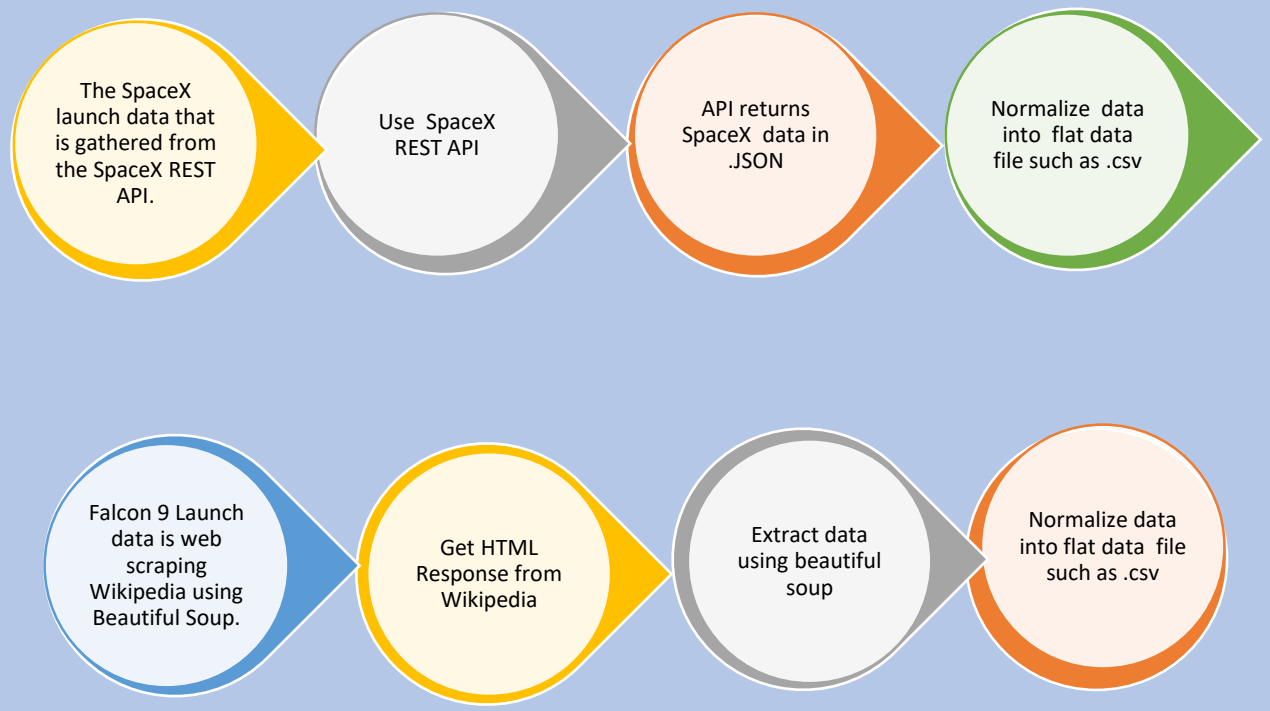
- What are the factors that determines successful launching?
- What is the impact of these rocket factors on the success
- What conditions does SpaceX have to achieve to get the best results  and ensure the best rocket success landing rate.

# Methodology

## Executive Summary

- **Data collection methodology**: SpaceX Rest API and Web Scrapping from Wikipedia
- **Performed Data Wrangling -** Encoding data fields for Machine Learning and dropping irrelevant columns
- **Performed EDA using visualization and SQ** - Plotting : Scatter Graphs, Bar Graphs to show relationships between variables to show patterns of data.
- **Performed interactive visual analytics using Folium and Plotly Dash**
- **Performed predictive analysis using classification models -** How to build, tune, evaluate classification models

# Data Collection



## Data collection –SpaceX API

*simplified flow chart*

### 1 .Getting Response from API

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
response = requests.get(spacex_url).json()
```

### 2. Converting Response to a .json file

```
response = requests.get(static_json_url).json()
data = pd.json_normalize(response)
```

### 3. Apply custom functions to clean data

```
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
```

```
getBoosterVersion(data)
```

### 4. Assign list to dictionary then dataframe

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
```

```
df = pd.DataFrame.from_dict(launch_dict)
```

### 5. Filter dataframe and export to flat file (.csv)

```
data_falcon9 = df.loc[df['BoosterVersion']!="Falcon 1"]
```

```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

*simplified flow chart*

## Data collection – Web Scrapping

### 1 .Getting Response from HTML

```
page = requests.get(static_url)
```

### 2. Creating BeautifulSoup Object

```
soup = BeautifulSoup(page.text, 'html.parser')
```

### 3. Finding tables

```
html_tables = soup.find_all('table')
```

### 4. Getting column names

```
column_names = []
temp = soup.find_all('th')
for x in range(len(temp)):
    try:
     name = extract_column_from_header(temp[x])
     if (name is not None and len(name) > 0):
        column_names.append(name)
    except:
     pass
```

### 5. Creation of dictionary

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']


launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

### 6. Appending data to keys (refer) to notebook block 12

```
In [12]: extracted_row = 0
         #Extract each table
         for table_number,table in enumerate(
            # get table row
             for rows in table.find_all("tr")
                 #check to see if first table
```

### 7. Converting dictionary to dataframe

```
df = pd.DataFrame.from_dict(launch_dict)
```

### 8. Dataframe to .CSV

```
df.to_csv('spacex_web_scraped.csv', index=False)
```

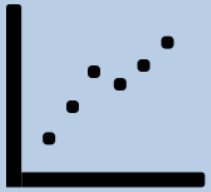
# EDA with data visualization

Scatter Graphs :

**Flight Number VS. Payload Mass Flight Number VS. Launch Site Payload VS. Launch Site**

**Orbit VS. Flight Number Payload VS. Orbit Type**

Scatter plots show how much one vvariable is affected by another.



**Bar Graph:**

**Mean VS. Orbit**

**A bar diagram helps to compare sets of data between different groups at a glance. The graph represents categories on one axis and a discrete value in the other.**

**Line Graph :**

**Success Rate VS. Year**

**Line graphs helps to show data variables and trends very clearly and can help to make predictions about the results of data not yet recorded**

# EDA with SQL

List of SQL queries

- SQL query to Display  the names of the unique launch sites in the space mission.
- SQL query to  Display  5 records where launch sites begin with the string 'KSC’
- SQL query to Display the total payload mass carried by boosters launched by NASA (CRS)
- SQL query to Display average payload mass carried by booster version F9 v1.1
- SQL query to List the date where the successful landing outcome in drone ship was achieved.
- SQL query to List the names of the boosters which have success in ground pad and have payload mass greater than 4000  but less than 6000
- SQL query to List the total number of successful and failure mission outcomes
- SQL query to List the names of the booster versions which have carried the maximum payload mass.
- SQL query to List the records which will display the month names, successful landing outcomes in ground pad ,booster  versions, launch_site for the months in year 2017
- SQL query to Rank the count of successful landing_outcomes

# Build an Interactive Map with Folium

- To visualize the Launch Data into an interactive map.
- Latitude and Longitude Coordinates at each launch site were used. A Circle Marker around each launch site with a label of the name of the launch site was also added.
- The dataframe launch outcomes(failures, successes) was assigned to classes 0 and 1 with Green and Red markers on the map in a MarkerCluster()
- Using Haversine's formula the distance from the Launch Site to various landmarks was calculated to find various trends about what is around the Launch Site to measure patterns.
- Lines are drawn on the map to measure distance to landmarks.

# Build a Dashboard with Plotly Dash

- **The dashboard is built with Flask and Dash web framework.**
  - **Pie Chart showing the total launches by a certain site/all sites**
    - *display relative proportions of multiple classes of data.*
    - *size of the circle can be made proportional to the total quantity it represents.*

**Scatter Graph showing the relationship with Outcome and Payload Mass (Kg) for the different Booster Versions**

- It shows the relationship between two variables.
- It is the best method to show you a non-linear pattern.
- The range of data flow, i.e. maximum and minimum value, can be determined.
- Observation and reading are straightforward.

# Predictive analysis Classification

**BUILDING MODEL**

- **L**oad our dataset into NumPy and Pandas
- Transform Data
- Split our data into training and test data sets
- Check how many test samples we have
- Decide which type of machine learning algorithms we want to use
- Set our parameters and algorithms to GridSearchCV
- Fit our datasets into the GridSearchCV objects and train our dataset.

**EVALUATING MODEL**

- Check accuracy for each model
- Get tuned hyperparameters for each type of algorithms
- Plot Confusion Matrix

**IMPROVING MODEL**

- Feature Engineering
- Algorithm Tuning

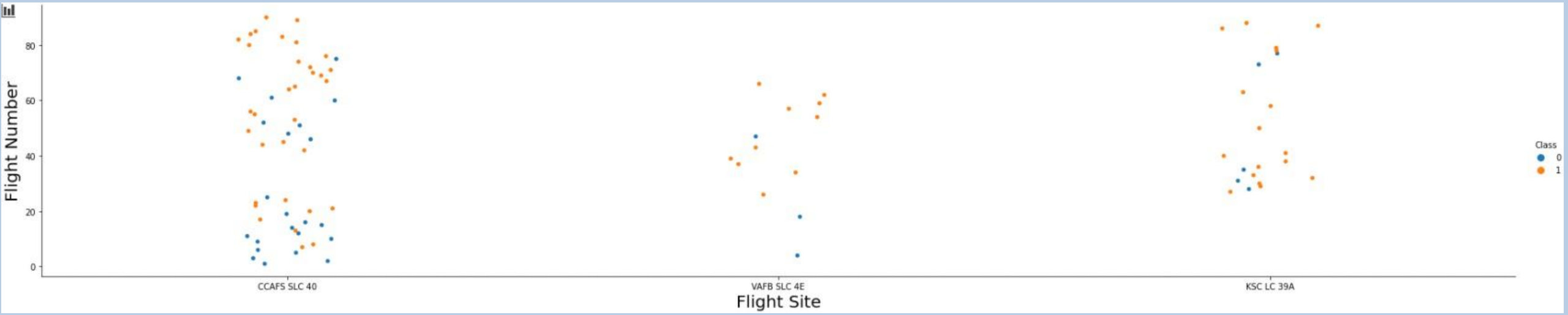
**FINDING THE BEST PERFORMING CLASSIFICATION MODEL**

- The model with the best accuracy score wins the best performing model
- In the notebook there is a dictionary of algorithms with scores at the bottom of the notebook.

# Results

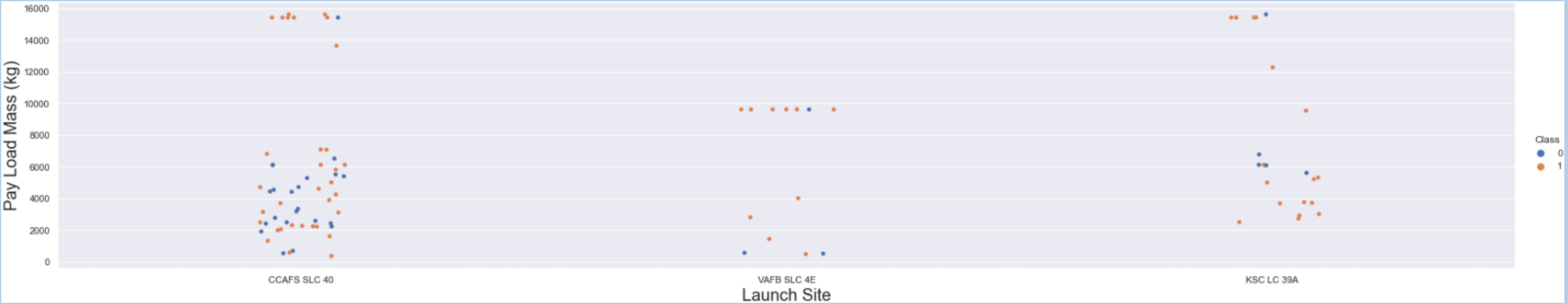
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

# Flight Number vs. Flight Site



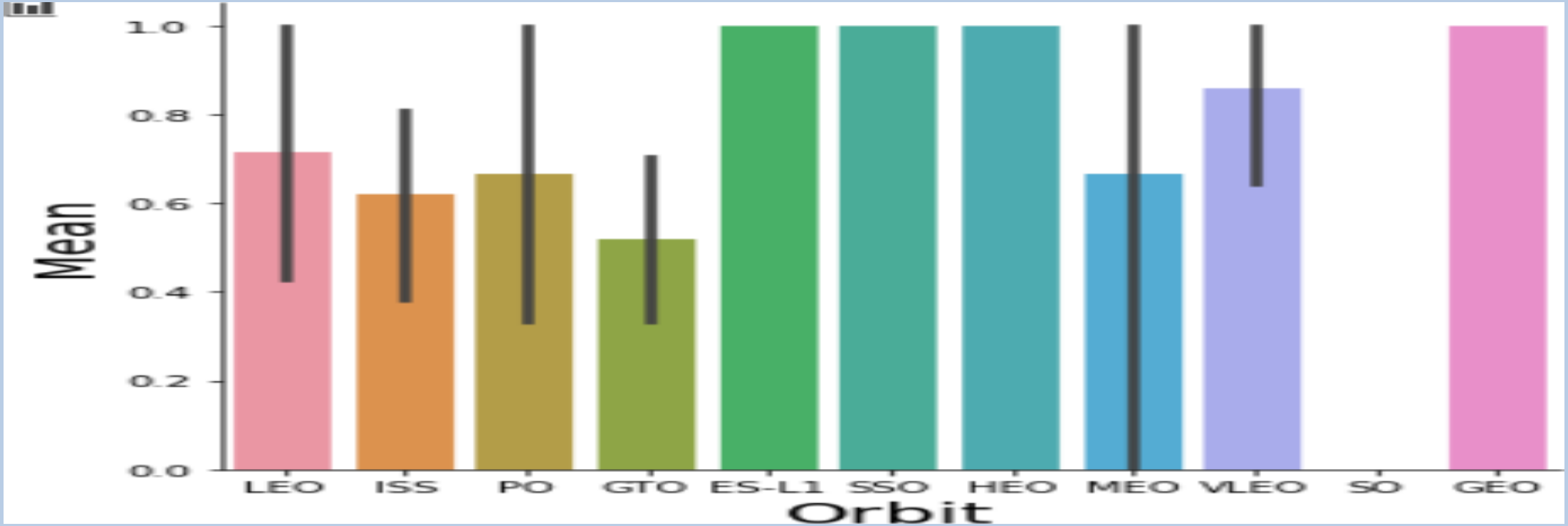

The more amount of flights at a launch site the greater the success rate at a launch site.

# Payload Mass vs. Launch Site



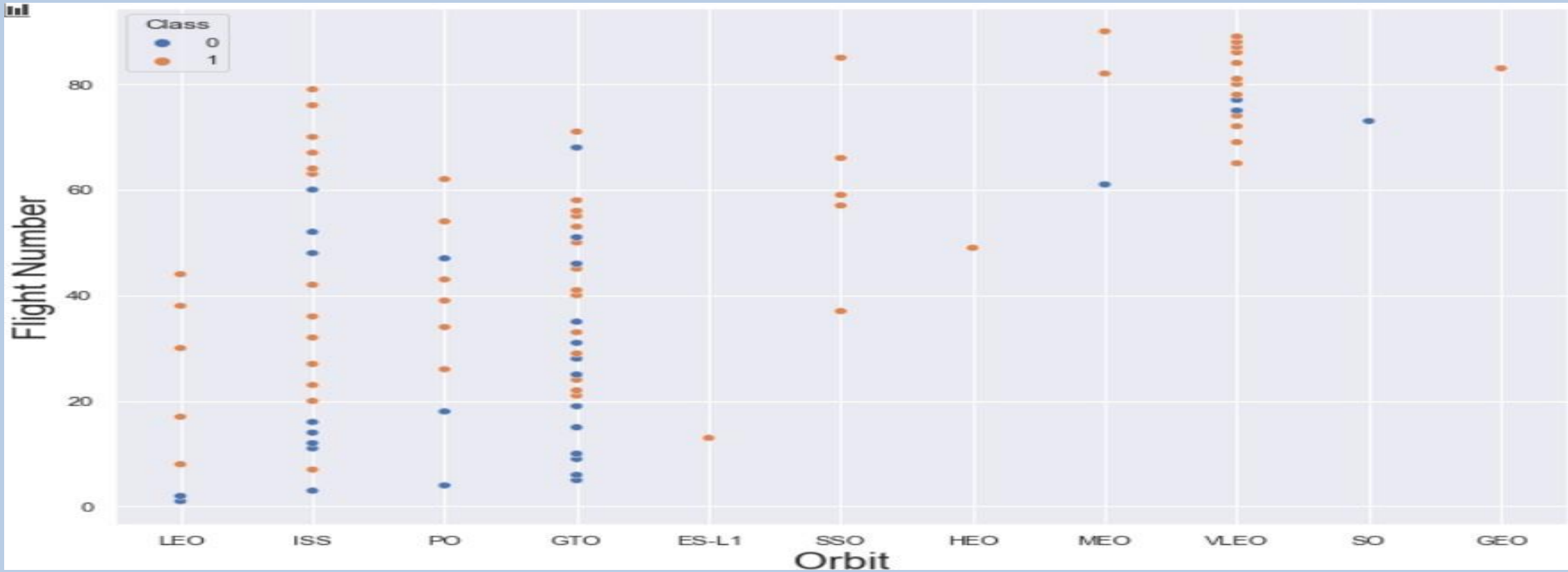

The greater the payload mass for Launch Site CCAFS SLC 40 the higher the success rate for the Rocket.

# Success rate vs. Orbit type



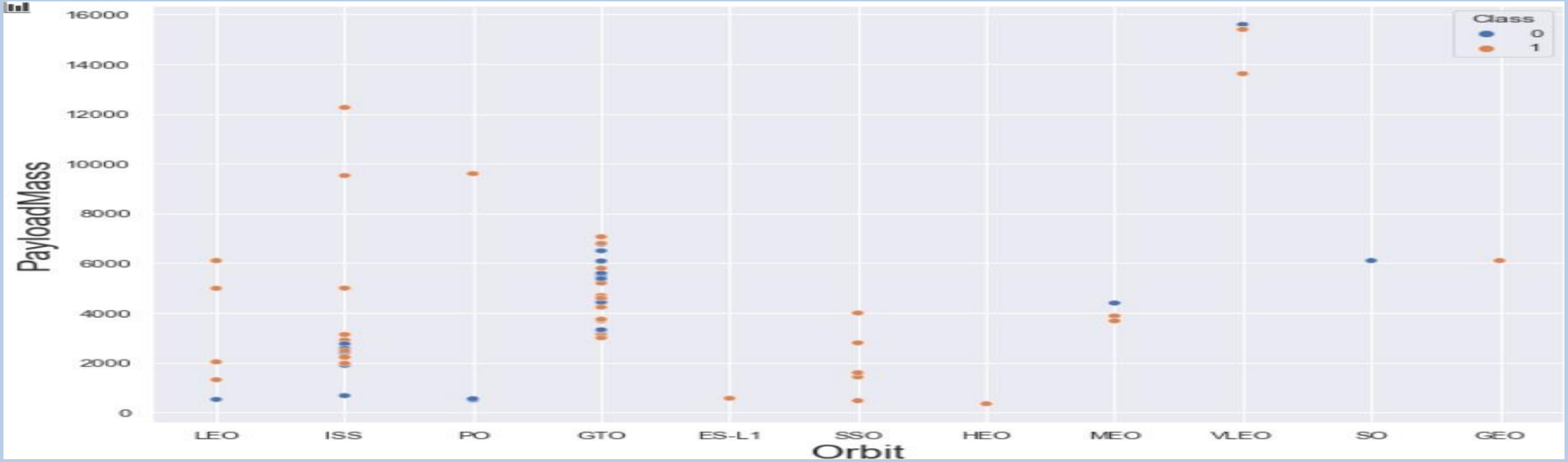

Orbit GEO,HEO,SSO,ES-L1 has the best Success Rate

# Flight Number vs. Orbit type



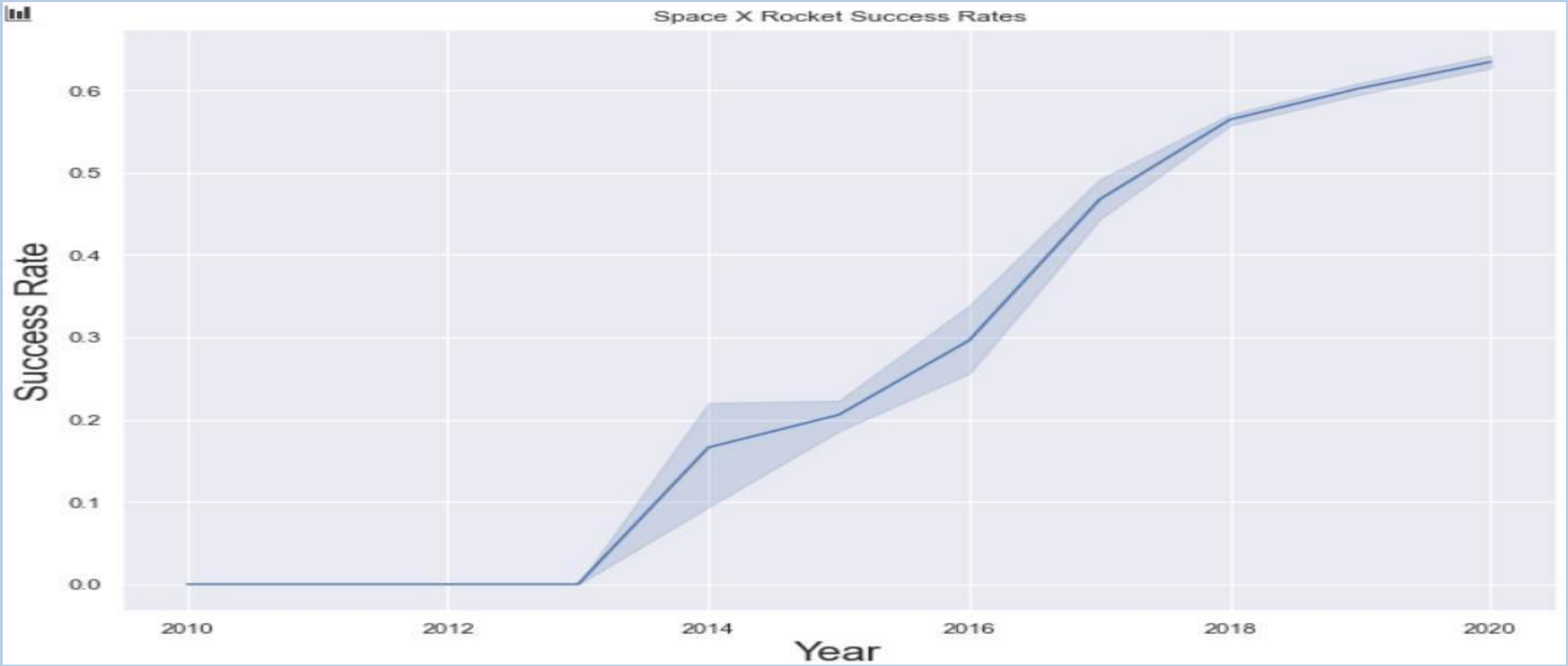

The LEO orbit the Success appears related to the number of flights

# 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




The success rate since  2013 kept increasing till 2020

# All launch site names

select DISTINCT Launch_Site from tblSpaceX

| Unique Launch Sites |
| --- |
| CCAFS LC-40 |
| CCAFS SLC-40 |
| CCAFS SLC-40 |
| KSC LC-39A |
| VAFB SLC-4E |

**Using the word *DISTINCT* in the query means that it will only show Unique values in the *Launch_Site* column from *tblSpaceX***

# Launch site names begin with `CCA`

**select TOP 5 * from tblSpaceX WHERE Launch_Site LIKE 'KSC%'**

| | Date | Time_UTC | Booster_Version | Launch_Site | Payload | PAYLOAD_MASS_KG_ | Orbit | Customer | Mission_Outcome | Landing_Outcome |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 19-02-2017 | 2021-07-02 14:39:00.0000000 | F9 FT B1031.1 | KSC LC-39A | SpaceX CRS-10 | 2490 | LEO (ISS) | NASA (CRS) | Success | Success (ground pad) |
| 1 | 16-03-2017 | 2021-07-02 06:00:00.0000000 | F9 FT B1030 | KSC LC-39A | EchoStar 23 | 5600 | GTO | EchoStar | Success | No attempt |
| 2 | 30-03-2017 | 2021-07-02 22:27:00.0000000 | F9 FT B1021.2 | KSC LC-39A | SES-10 | 5300 | GTO | SES | Success | Success (drone ship) |
| 3 | 01-05-2017 | 2021-07-02 11:15:00.0000000 | F9 FT B1032.1 | KSC LC-39A | NROL-76 | 5300 | LEO | NRO | Success | Success (ground pad) |
| 4 | 15-05-2017 | 2021-07-02 23:21:00.0000000 | F9 FT B1034 | KSC LC-39A | Inmarsat-5 F4 | 6070 | GTO | Inmarsat | Success | No attempt |

Using the word ***TOP 5*** in the query means that it will only show 5 records from ***tblSpaceX*** and ***LIKE*** keyword has a wild card with the words ***'KSC%'*** the percentage in the end suggests that the Launch_Site name must start with KSC.

# Total payload mass

```
select SUM(PAYLOAD_MASS_KG_) TotalPayloadMass from tblSpaceX where Customer = 'NASA (CRS)'",'TotalPayloadMass
```

| | Total Payload Mass |
|---|---|
| 0 | 45596 |

Using the function ***SUM*** summates the total in the column ***PAYLOAD_MASS_KG_***
The ***WHERE*** clause filters the dataset to only perform calculations on ***Customer NASA (CRS)***

# Average Payload Mass carried by booster version F9 v1.1

```
select AVG(PAYLOAD_MASS_KG_) AveragePayloadMass from tblSpaceX where Booster_Version = 'F9 v1.1'
```

| | Average Payload Mass |
|---|---|
| 0 | 2928 |

Using the function ***AVG*** works out the average in the column ***PAYLOAD_MASS_KG_***

The ***WHERE*** clause filters the dataset to only perform calculations on ***Booster_version F9 v1.1***

# First successful ground landing date

select MIN(Date) SLO from tblSpaceX where Landing_Outcome = ”Success (drone ship)”

| | Date which first Successful landing outcome in drone ship was acheived. |
|---|---|
| 0 | 06-05-2016 |

Using the function ***MIN*** works out the minimum date in the column ***Date***

The ***WHERE*** clause filters the dataset to only perform calculations on ***Landing_Outcome Success (drone ship)***

# Successful drone ship landing with payload between 4000 and 6000

select Booster_Version from tblSpaceX where Landing_Outcome = 'Success (ground pad)’ AND Payload_MASS_KG_ > 4000 AND Payload_MASS_KG_ < 6000

| | Date which first Successful landing outcome in drone ship was acheived. |
|---|---|
| 0 | F9 FT B1032.1 |
| 1 | F9 B4 B1040.1 |
| 2 | F9 B4 B1043.1 |

The WHERE clause filters the dataset to Landing_Outcome =Success (drone ship)

The AND clause specifies additional filter conditions Payload_MASS_KG_ > 4000 AND Payload_MASS_KG_ < 6000

# Total Number of Successful and Failure Mission Outcomes

SELECT(SELECT Count(Mission_Outcome) from tblSpaceX where Mission_Outcome LIKE '%Success%') as Successful_Mission_Outcomes,
(SELECT Count(Mission_Outcome) from tblSpaceX where Mission_Outcome LIKE '%Failure%') as Failure_Mission_Coutcomes

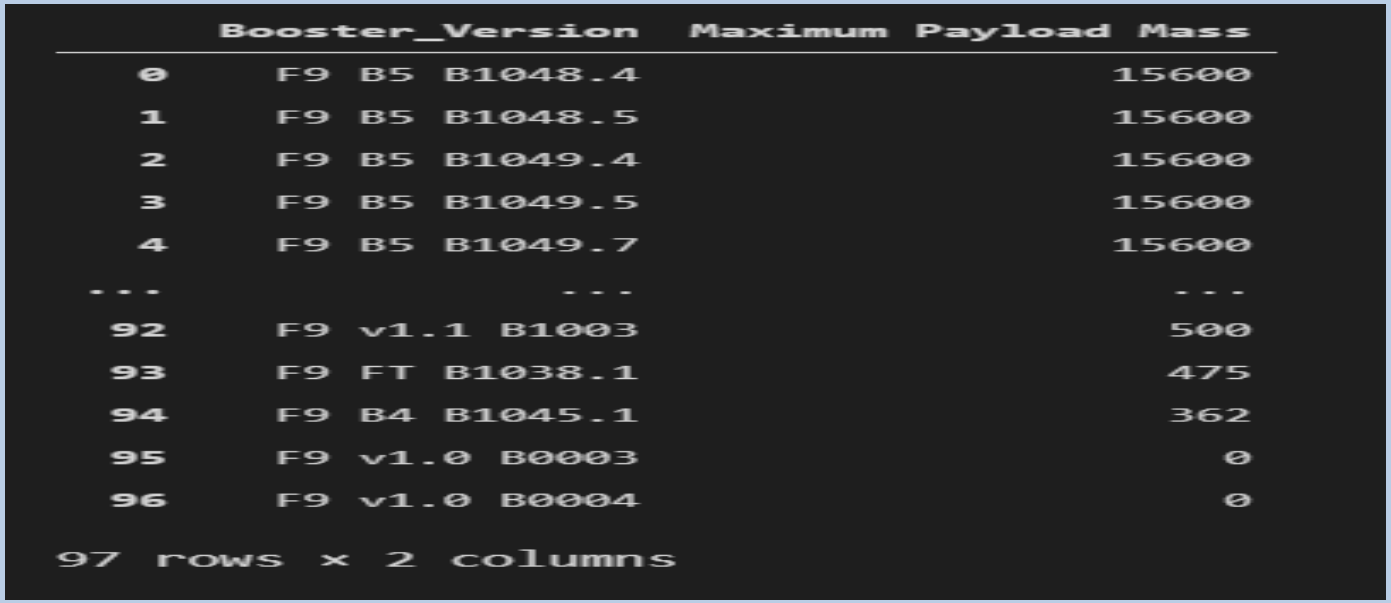
| | Successful_Mission_Outcomes | Failure_Mission_Outcomes |
|---|---|---|
| 0 | 100 | 1 |

The LIKE '%foo%' wildcard shows that in the record the foo phrase is in any part of the string in the records for example.

PHRASE "(Drone Ship was a Success)" LIKE '%Success%' Word 'Success' is in the phrase the filter will include it in the dataset

# Boosters  maximum payload

SELECT DISTINCT Booster_Version, MAX(PAYLOAD_MASS _KG_) AS [Maximum Payload Mass] FROM tblSpaceX GROUP BY Booster_Version ORDER BY [Maximum Payload Mass] DESC



| | Booster_Version | Maximum Payload Mass |
|---|---|---|
| 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 |
| ... | ... | ... |
| 92 | F9 v1.1 B1003 | 500 |
| 93 | F9 FT B1038.1 | 475 |
| 94 | F9 B4 B1045.1 | 362 |
| 95 | F9 v1.0 B0003 | 0 |
| 96 | F9 v1.0 B0004 | 0 |

97 rows x 2 columns

Using the word DISTINCT in the query means that it will only  show Unique values in the Booster_Version column from  tblSpaceX GROUP BY puts the list in order set to a certain condition. DESC means its arranging the dataset into descending order

# 2015 launch records

SELECT DATENAME(month, DATEADD(month, MONTH(CONVERT(date, Date, 105)), 0) - 1) AS Month, Booster_Version, Launch_Site, Landing_Outcome FROM tblSpaceX WHERE (Landing_Outcome LIKE N'%Success%') AND (YEAR(CONVERT(date, Date, 105)) = '2015')

| Month | Booster_Version | Launch_Site | Landing_Outcome |
|---|---|---|---|
| January | F9 FT B1029.1 | VAFB SLC-4E | Success (drone ship) |
| February | F9 FT B1031.1 | KSC LC-39A | Success (ground pad) |
| March | F9 FT B1021.2 | KSC LC-39A | Success (drone ship) |
| May | F9 FT B1032.1 | KSC LC-39A | Success (ground pad) |
| June | F9 FT B1035.1 | KSC LC-39A | Success (ground pad) |
| June | F9 FT B1029.2 | KSC LC-39A | Success (drone ship) |
| June | F9 FT B1036.1 | VAFB SLC-4E | Success (drone ship) |
| August | F9 B4 B1039.1 | KSC LC-39A | Success (ground pad) |
| August | F9 FT B1038.1 | VAFB SLC-4E | Success (drone ship) |
| September | F9 B4 B1040.1 | KSC LC-39A | Success (ground pad) |
| October | F9 B4 B1041.1 | VAFB SLC-4E | Success (drone ship) |
| October | F9 FT B1031.2 | KSC LC-39A | Success (drone ship) |
| October | F9 B4 B1042.1 | KSC LC-39A | Success (drone ship) |
| December | F9 FT B1035.2 | CCAFS SLC-40 | Success (ground pad) |

The function CONVERT converts NVARCHAR to Date. WHERE clause filters Year to be 2015

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

SELECT COUNT(Landing_Outcome) FROM tblSpaceX WHERE (Landing_Outcome LIKE '%Success%') AND (Date > '04-06-2010') AND (Date < '20-03-2017')

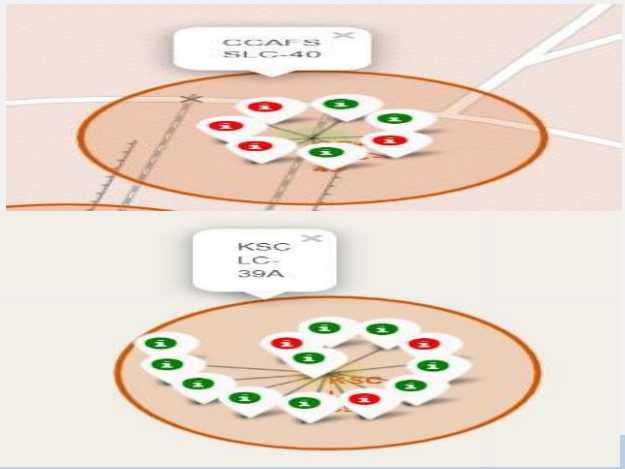
| | Successful Landing Outcomes Between 2010-06-04 and 2017-03-20 |
|---|---|
| 0 | 34 |

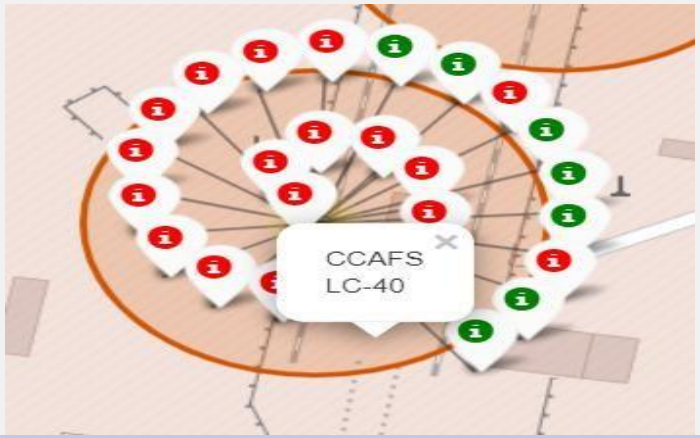
Function COUNT counts records in column WHERE filters data LIKE (wildcard) AND (conditions) AND (conditions)

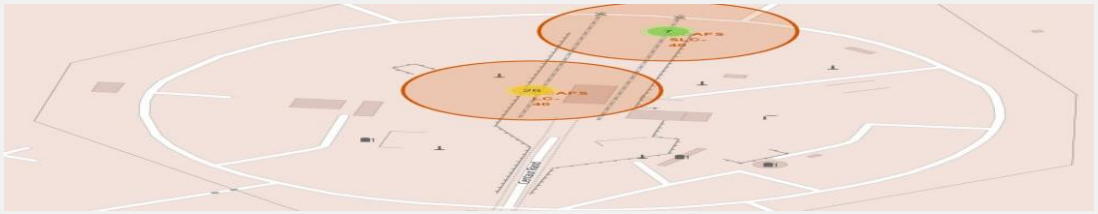
# <Folium map screenshot 1>


We can see that the SpaceX launch sites are in the United States of America coasts. Florida and California

# <Folium Map Screenshot 2>

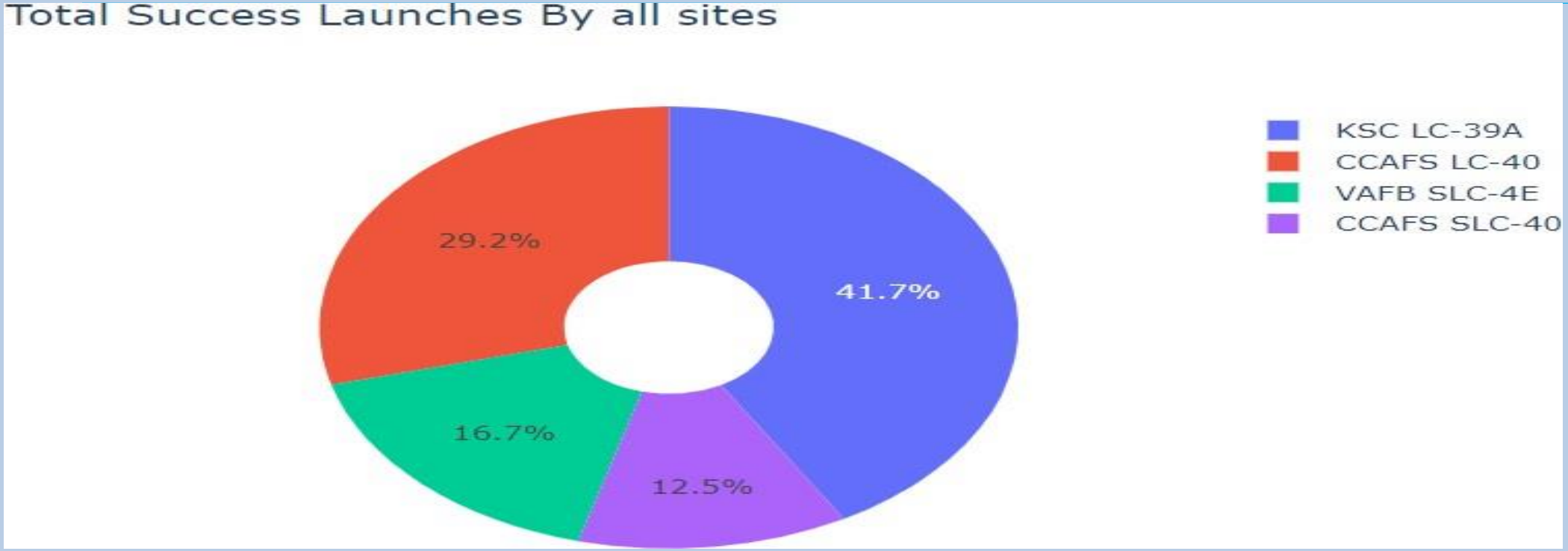


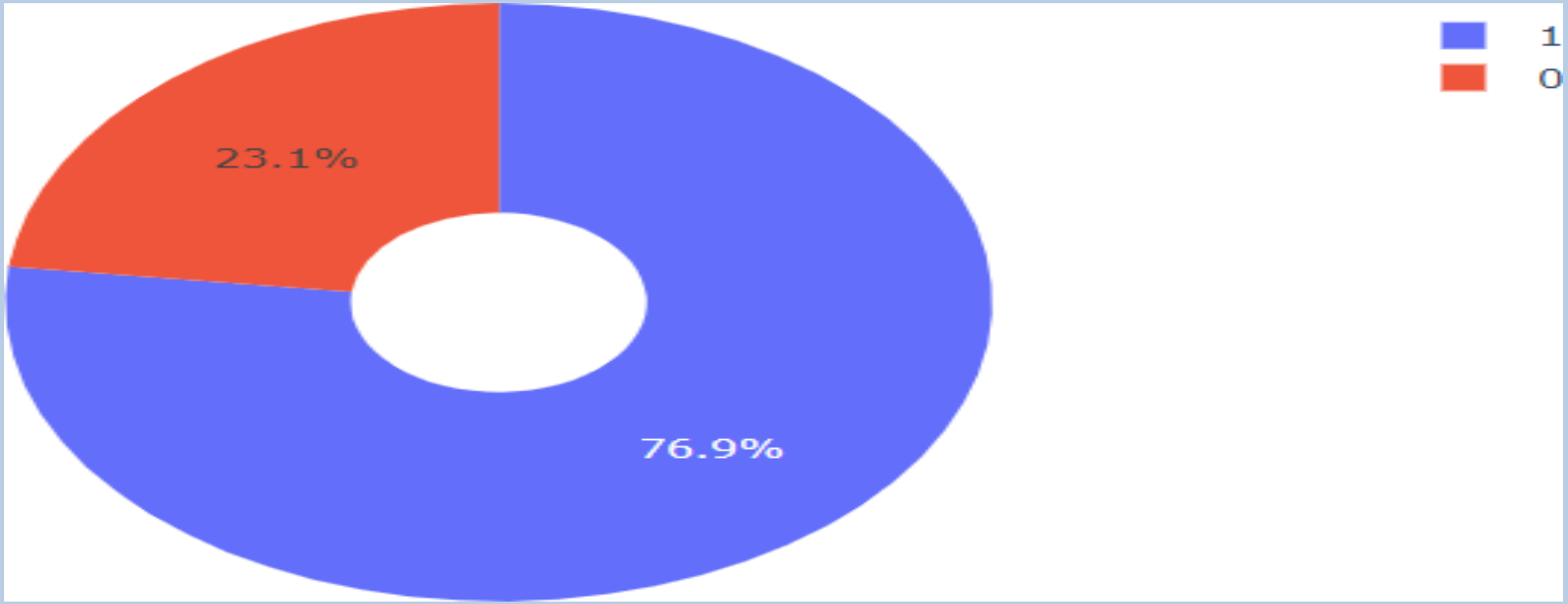
*Green Marker shows successful Launches and Red Marker shows Failures*

# <Dashboard screenshot 1>




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

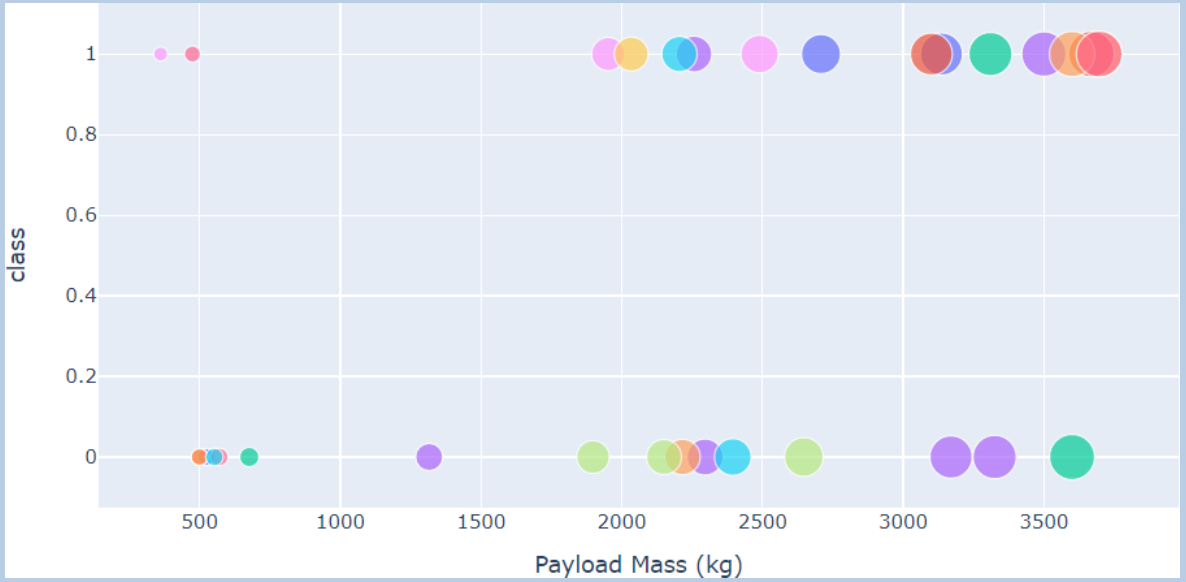
# <Dashboard Screenshot 2>



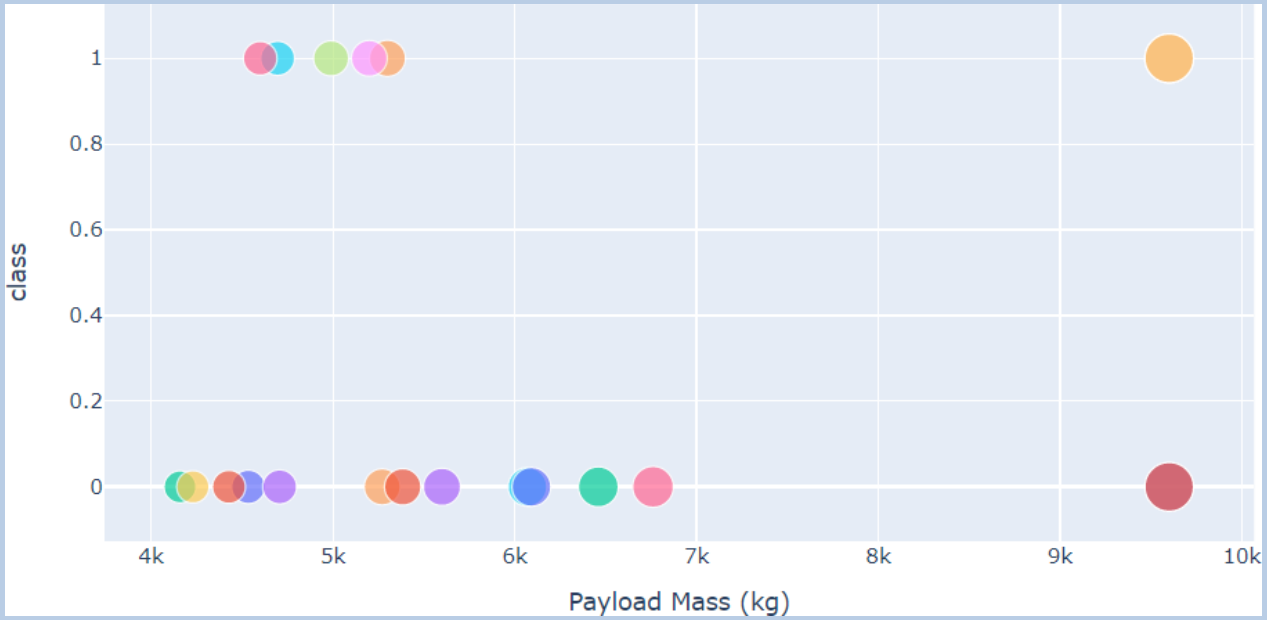

***KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate***

# <Dashboard screenshot 3>

***Low Weighted Payload 0kg – 4000kg***




***Heavy Weighted Payload 4000kg – 10000kg***



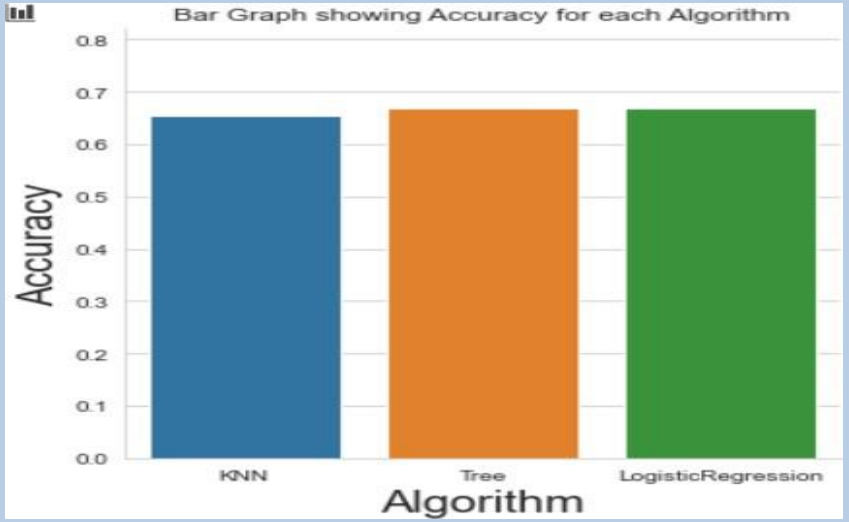

*We can see the success rates for low weighted payloads is higher than the heavy weighted payloads*

# Classification Accuracy using training data

As *you can see our accuracy is extremely close but we  do have a winner its down to decimal places! using this function*

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

| | Accuracy | Algorithm |
|---|---|---|
| 0 | 0.653571 | KNN |
| 1 | 0.667857 | Tree |
| 2 | 0.667857 | LogisticRegression |

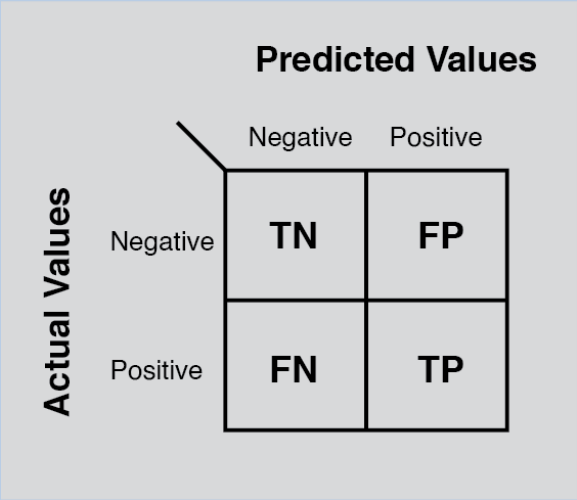


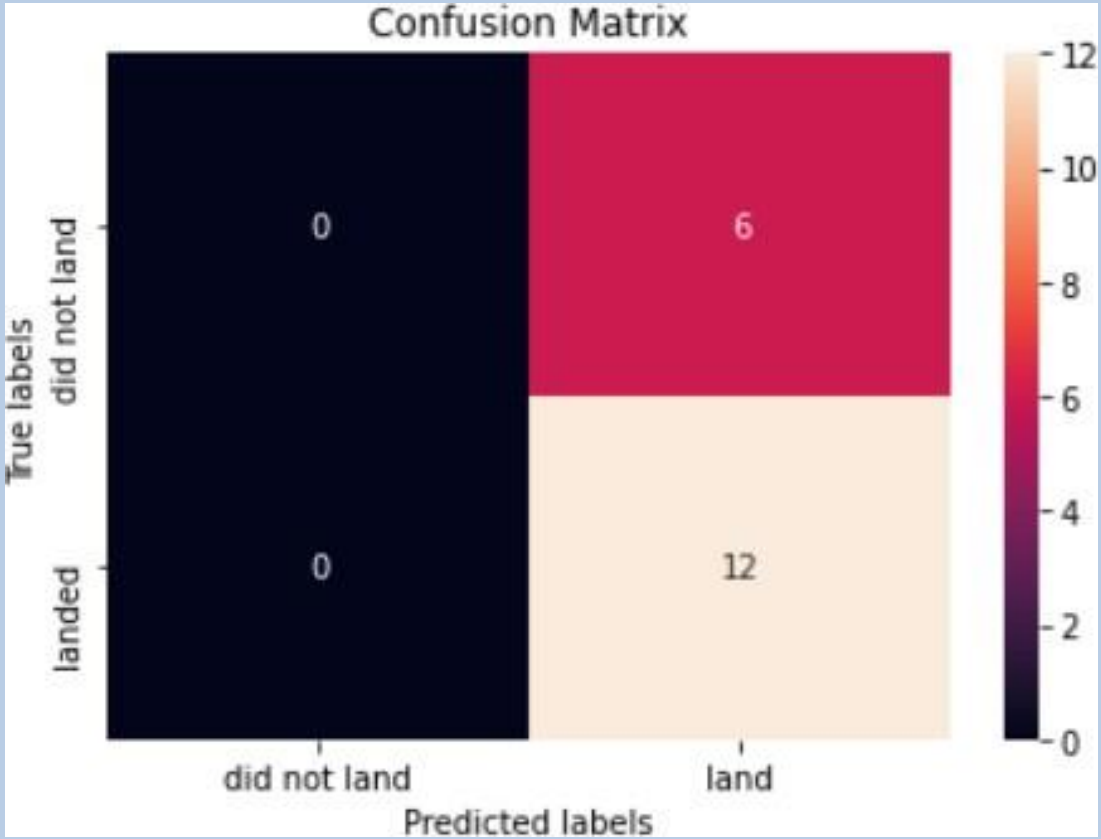

```
Best Algorithm is Tree with a score of 0.6678571428571429
Best Params is : {'criterion': 'gini', 'max_depth': 2, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 2, 'splitter': 'best'}
```

After selecting the best hyperparameters for the decision tree classifier using the validation data, we achieved 83.33% accuracy on the test data.

# Confusion Matrix for the Tree

Examining the confusion matrix, we see that Tree can distinguish between the different classes. We see that the major problem is false positives.



# CONCLUSIONS

1. The Tree Classifier Algorithm is the best for Machine Learning for this dataset
2. Low weighted payloads perform better than the heavier payloads
3. The success rates for SpaceX launches is  directly proportional time in years they will eventually perfect the launches.
4. KSC LC-39A had the most successful launches from all the sites
5. Orbit GEO,HEO,SSO,ES-L1 has the best Success Rate