



IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

AUGUSTIN RAJKUMAR S
20 MAY 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 from Machine Learning Lab

Introduction

SpaceX is a revolutionary company who has disrupted the space industry by offering a rocket launch specifically Falcon 9 as low as 62 million dollars; while other providers cost upward of 165 million dollars each. Most of this saving thanks to SpaceX's astounding idea to reuse the first stage of the launch by re-land the rocket to be used on the next mission. Repeating this process will make the price even further down. As a data scientist of a startup rivaling SpaceX, the goal of this project is to create the machine learning pipeline to predict the landing outcome of the first stage in the future. This project is crucial in identifying the right price to bid against SpaceX for a rocket launch.

The problems included:

- Identifying all factors that influence the landing outcome.
- The relationship between each variable and how it is affecting the outcome.
- The best condition needed to increase the probability of successful landing.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
 - Data was collected using SpaceX REST API and web-scraping from Wikipedia
- Perform data wrangling
 - Data was processed using one-hot encoding for categorical features
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
 - How to build, tune, evaluate classification models

Data Collection

Data collection is the process of gathering and measuring information on targeted variables in an established system, which then enables one to answer relevant questions and evaluate outcomes. As mentioned, the dataset was collected by REST API and Web Scrapping from Wikipedia

For REST API, its started by using the get request. Then, we decoded the response content as json and turn it into a pandas dataframe using `json_normalize()`. We then cleaned the data, checked for missing values and fill with whatever needed.

For web scrapping, we will use the BeautifulSoup to extract the launch records as HTML table, parse the table and convert it to a pandas dataframe for further analysis

Data Collection – SpaceX API

Get Request for rocket launch data using API

Use json_normalize method to convert json result to dataframe

Performed data cleaning and filling the missing value

From:

<https://github.com/AugustinRajkumar16/IBM-Project/blob/master/Data%20Collection%20API.ipynb>

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

```
response = requests.get(spacex_url)
```

```
# Use json_normalize meethod to convert the json result into a dataframe  
data = pd.json_normalize(response.json())
```

```
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.  
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]
```

```
# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and rows that have missing values  
data = data[data['cores'].map(len)!=1]  
data = data[data['payloads'].map(len)!=1]
```

```
# Since payloads and cores are lists of size 1 we will also extract the single value in the list and replace the feature.  
data['cores'] = data['cores'].map(lambda x : x[0])  
data['payloads'] = data['payloads'].map(lambda x : x[0])
```

```
# We also want to convert the date_utc to a datetime datatype and then extracting the date leaving the time  
data['date'] = pd.to_datetime(data['date_utc']).dt.date
```

```
# Using the date we will restrict the dates of the launches  
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```


Data Collection - Scraping

Request the Falcon9 Launch
Wiki page from URL

Create a BeautifulSoup from
the HTML Response

Extract all column/variable
names from the HTML header

From:

https://github.com/AugustinRajkumar16/IBM_Project/blob/master/Data%20Collection%20with%20Web%20Scraping%20lab.ipynb

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

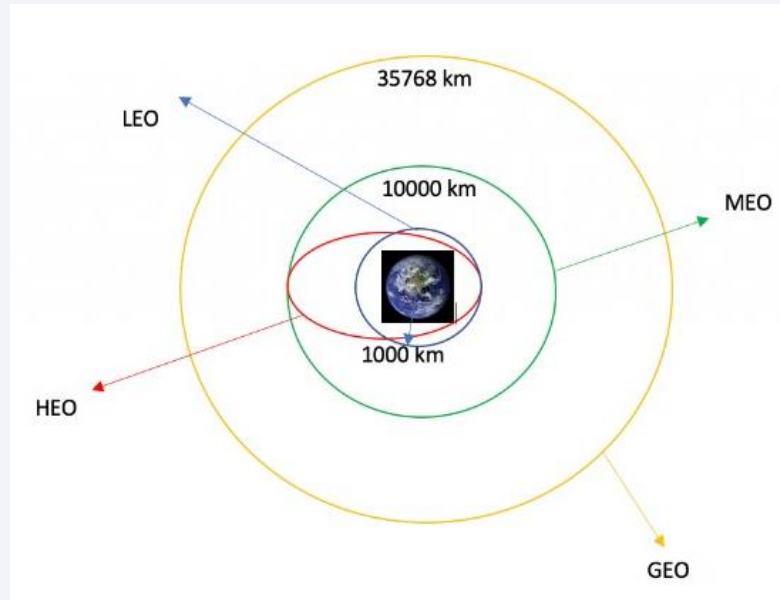
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response, 'html.parser')
```

```
extracted_row = 0
#Extract each table
for table_number, table in enumerate(soup.find_all('table', "wikitable plainrowheaders collapsible")):
    # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
            else:
                flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictionary
        if flag:
            extracted_row += 1
            # Flight Number value
            # TODO: Append the flight_number into Launch_dict with key `Flight No.`
            #print(flight_number)
            datatimelist=date_time(row[0])

            # Date value
            # TODO: Append the date into Launch_dict with key `Date`
            date = datatimelist[0].strip(',')
            #print(date)

            # Time value
            # TODO: Append the time into Launch_dict with key `Time`
            time = datatimelist[1]
            #print(time)
```

Data Wrangling



From:

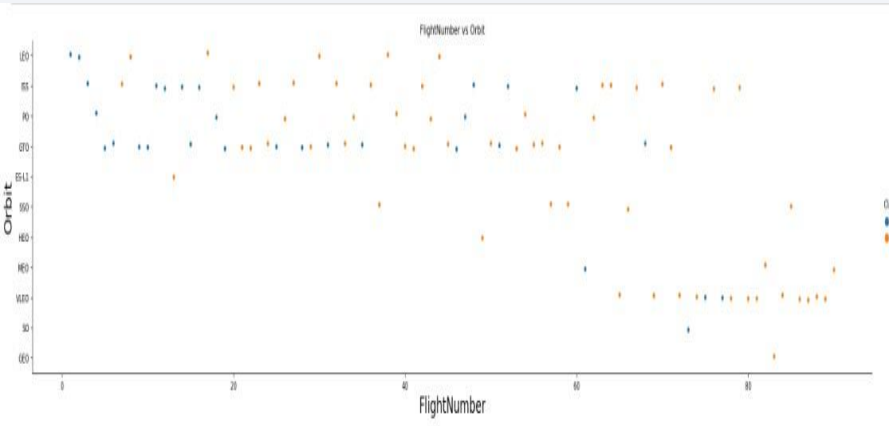
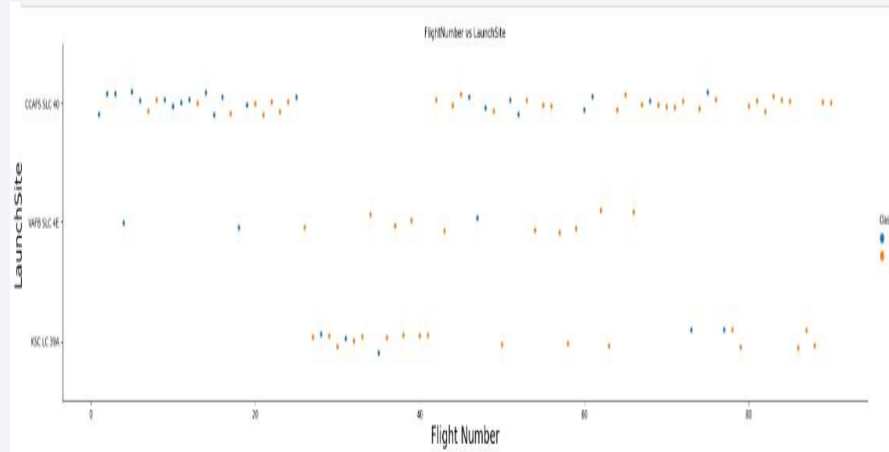
https://github.com/AugustinRajkumar16/IBM_Project/blob/master/DataWrangling.ipynb

Data Wrangling is the process of cleaning and unifying messy and complex data sets for easy access and Exploratory Data Analysis (EDA).

We will first calculate the number of launches on each site, then calculate the number and occurrence of mission outcome per orbit type.

We then create a landing outcome label from the outcome column. This will make it easier for further analysis, visualization, and ML. Lastly, we will export the result to a CSV

EDA with Data Visualization



We first started by using scatter graph to find the relationship between the attributes such as between:

- Payload and Flight Number.
- Flight Number and Launch Site.
- Payload and Launch Site.
- Flight Number and Orbit Type.
- Payload and Orbit Type.

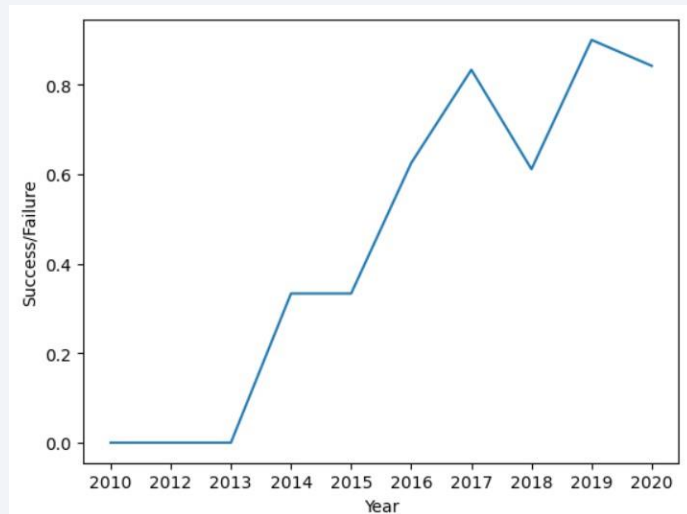
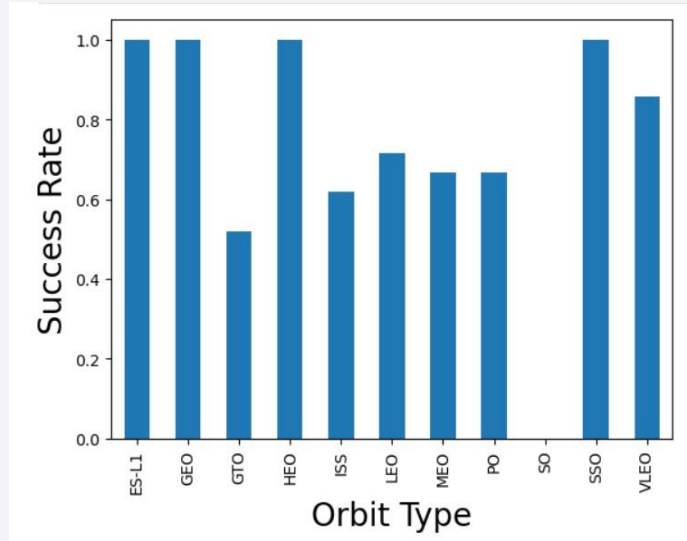
Scatter plots show dependency of attributes on each other.

Once a pattern is determined from the graphs. It's very easy to see which factors affecting the most to the success of the landing outcomes

From:

https://github.com/AugustinRajkumar16/IBM_Project/blob/master/EDA%20with%20Visualization%20lab.ipynb

EDA with Data Visualization



- Once we get a hint of the relationships using scatter plot. We will then use further visualization tools such as bar graph and line plots graph for further analysis.
- Bar graphs is one of the easiest way to interpret the relationship between the attributes. In this case, we will use the bar graph to determine which orbits have the highest probability of success.
- We then use the line graph to show a trends or pattern of the attribute over time which in this case, is used for see the launch success yearly trend.
- We then use Feature Engineering to be used in success prediction in the future module by created the dummy variables to categorical columns.

From:

https://github.com/AugustinRajkumar16/IBM_Project/blob/master/EDA%20with%20Visualization%20lab.ipynb

EDA with SQL

Using SQL, we had performed many queries to get better understanding of the dataset, Ex:

- Displaying the names of the launch sites.
- Displaying 5 records where launch sites begin with the string 'CCA'.
- Displaying the total payload mass carried by booster launched by NASA (CRS).
- Displaying the 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 sites names for in year 2015.
- Rank the count of landing outcomes or success between the date 2010-06-04 and 2017-03-20, in descending order

From:

https://github.com/AugustinRajkumar16/IBM_Project/blob/master/EDA_WITH_SQL.ipynb

Build an Interactive Map with Folium

To visualize the launch data into an interactive map. We took the latitude and longitude coordinates at each launch site and added a circle marker around each launch site with a label of the name of the launch site.

We then assigned the dataframe launch_outcomes (failure,success) to classes 0 and 1 with **Red** and **Green** markers on the map in MarkerCluster().

We then used the Haversine's formula to calculate the distance of the launch sites to various landmarks to find answers to the questions of:

- How close the launch sites with railways, highways and coastlines?
- How close the launch sites with nearby cities?

From:

https://github.com/AugustinRajkumar16/IBM_Project/blob/master/Interactive%20Visual%20Analytics%20with%20Folium%20lab.ipynb

Build a Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly dash which allowing the user to play around with the data as they need.
- We plotted pie charts showing the total launches by a certain sites.
- We then plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.

From:

https://github.com/AugustinRajkumar16/IBM_Project/blob/master/spacex_dash_app.py

Predictive Analysis (Classification)

Building the Model

- Load the dataset into NumPy and Pandas
- Transform the data and then split into training and test datasets
- Decide which type of ML to use
- Set the parameters and algorithms to GridSearchCV and fit it to dataset.

Evaluating the Model

- Check the accuracy for each model
- Get tuned hyperparameters for each type of algorithms.
- Plot the confusion matrix.

Improving the Model

- Use Feature Engineering and Algorithm Tuning

Find the Best Model

- The model with the best accuracy score will be the best performing model.

From:

https://github.com/AugustinRajkumar16/IBM_Project/blob/master/space_dash_app.py

Results

The results will be categorized to 3 main results which is :

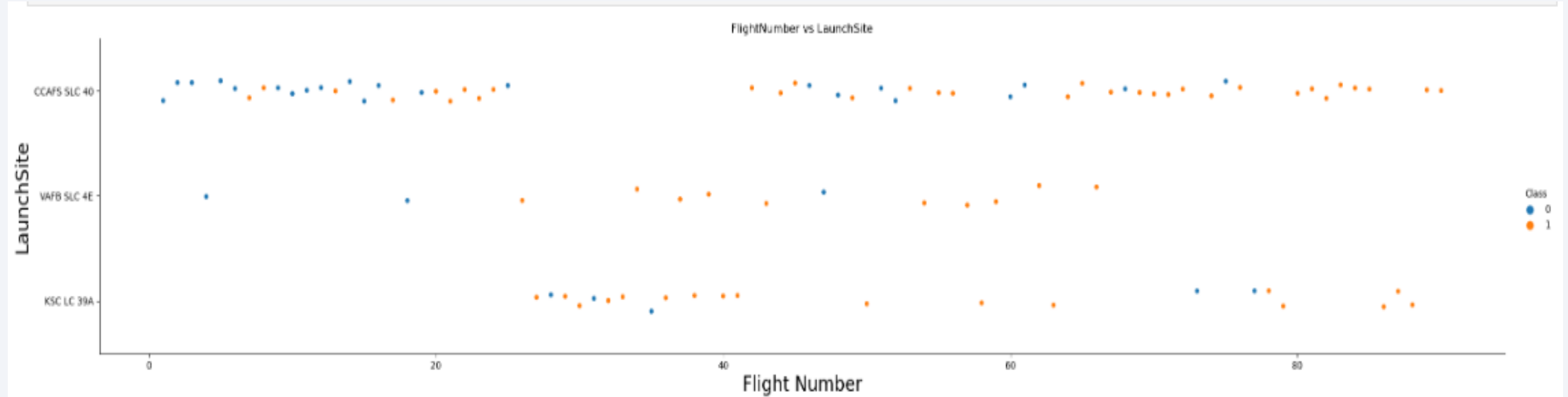
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

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

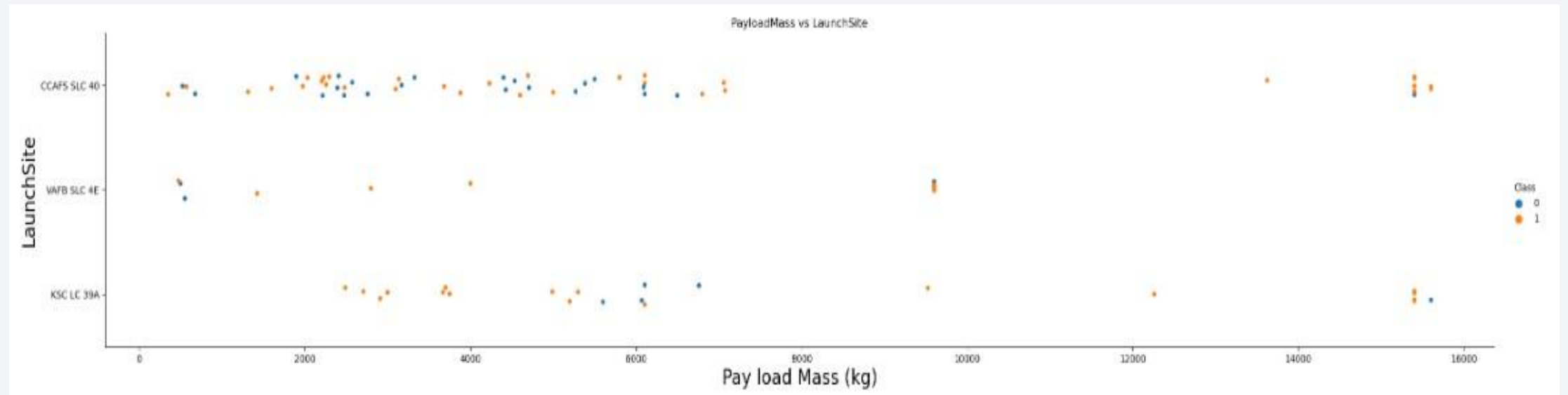
Insights drawn from EDA

Flight Number vs. Launch Site



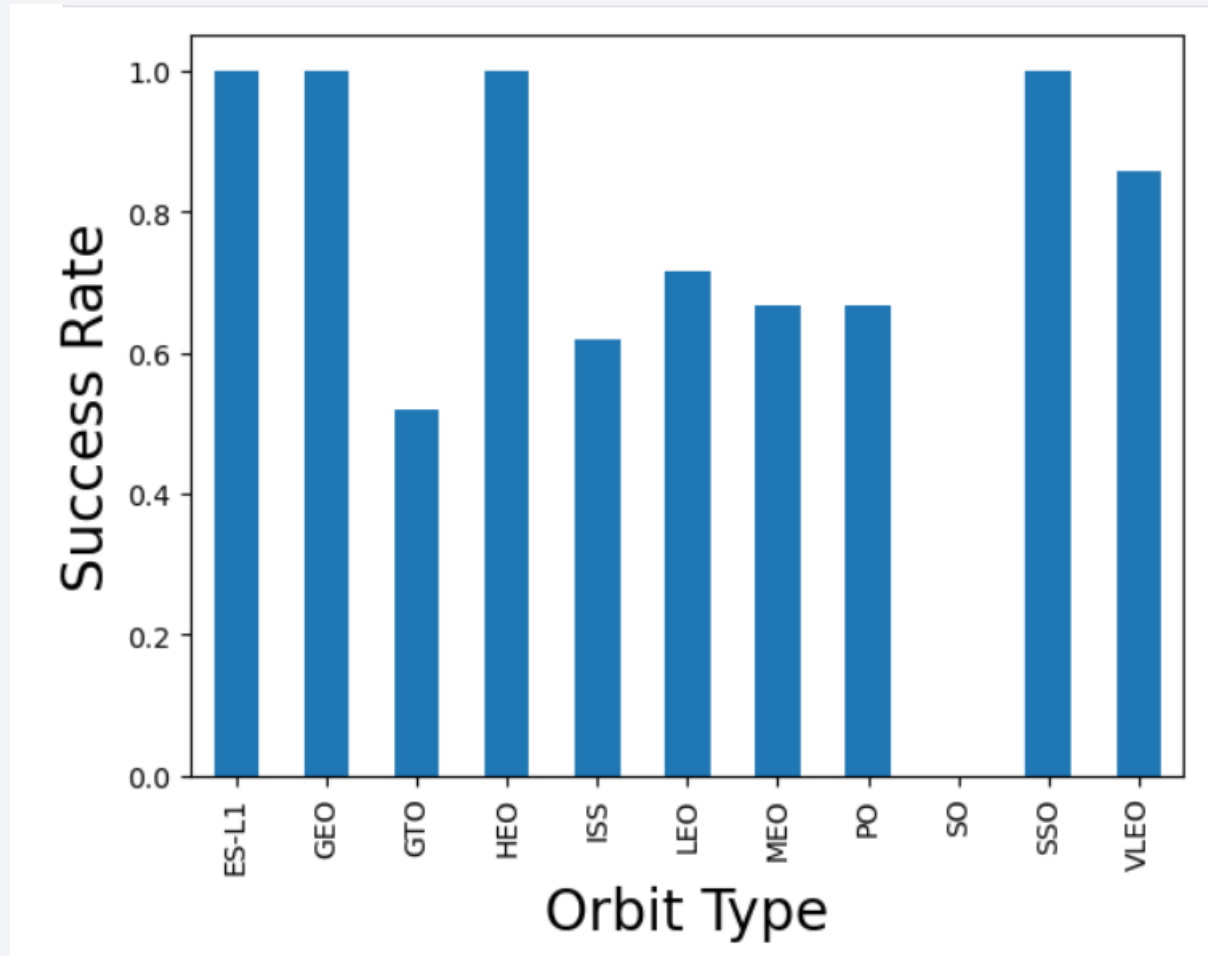
- This scatter plot shows that the larger the flights amount of the launch site, the greater the success rate will be.
- However, site CCAFS SLC40 shows the least pattern of this.

Payload vs. Launch Site



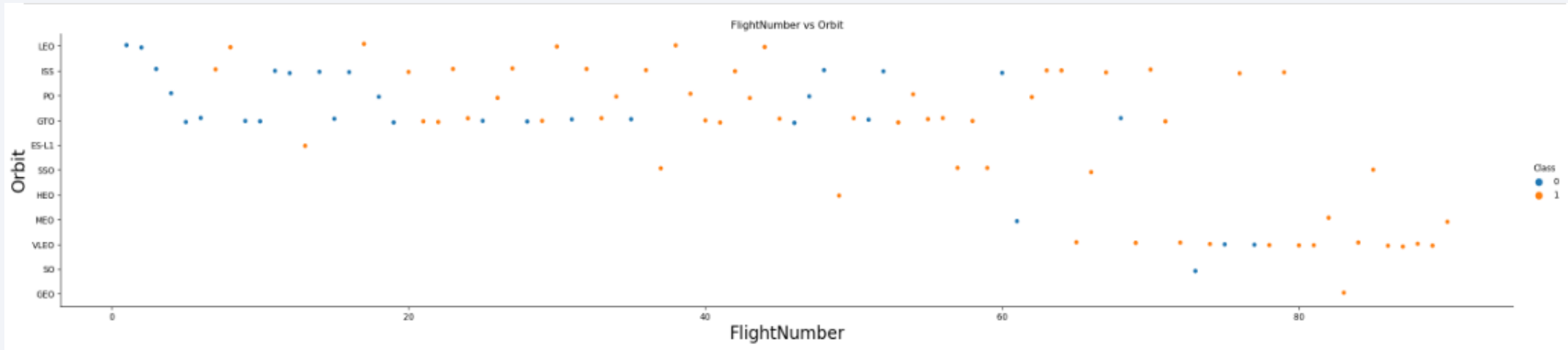
- This scatter plot shows once the payload mass is greater than 7000kg, the probability of the success rate will be highly increased.
- However, there is no clear pattern to say the launch site is dependent to the payload mass for the success rate.

Success Rate vs. Orbit Type



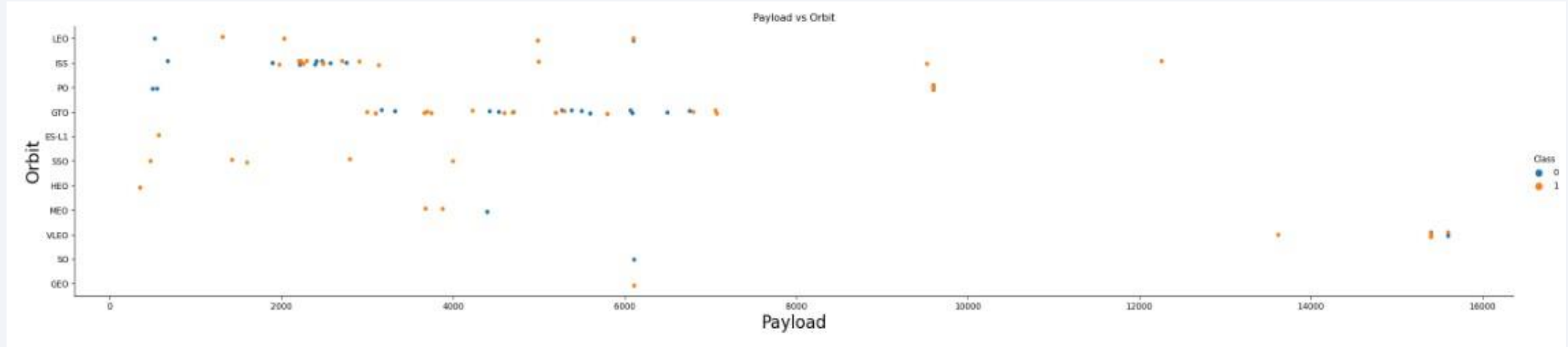
- This figure depicted the possibility of the orbits to influence the landing outcomes as some orbits have 100% success rate such as SSO, HEO, GEO AND ES-L1 while SO orbit produced 0% rate of success.
- However, deeper analysis shows that some of these orbits have only 1 occurrence such as GEO, SO, HEO and ES-L1 which means this data needs more dataset to see pattern or trend before we draw any conclusion.

Flight Number vs. Orbit Type



- This scatter plot shows that generally, the larger the flight number on each orbits, the greater the success rate (especially LEO orbit) except for GTO orbit which depicts no relationship between both attributes.
- Orbit that only has 1 occurrence should also be excluded from above statement as it's needed more dataset.

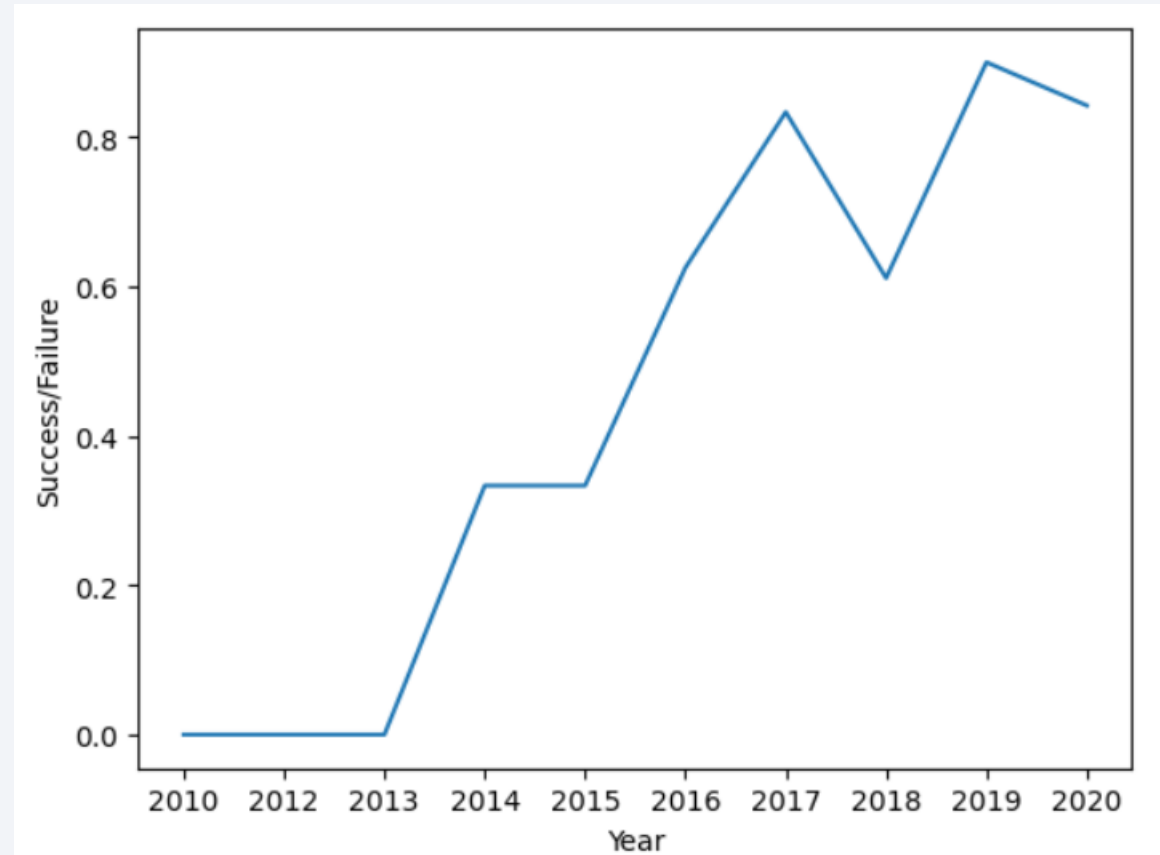
Payload vs. Orbit Type



- Heavier payload has positive impact on LEO, ISS and PO orbit. However, it has negative impact on MEO and VLEO orbit.
- GTO orbit seem to depict no relation between the attributes.
- Meanwhile, again, SO, GEO and HEO orbit need more dataset to see any pattern or trend.

Launch Success Yearly Trend

- This figures clearly depicted and increasing trend from the year 2013 until 2020.
- If this trend continue for the next year onward. The success rate will steadily increase until reaching 1/100% success rate.



All Launch Site Names

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

```
In [5]: %sql select distinct(launch_site) from spacex
```

```
* ibm_db_sa://fhh66868:***@824dfd4d-99de-440d-9991-629c01b3832d.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30119/bludb  
Done.
```

```
Out[5]: launch_site
```

```
CCAFS LC-40
```

```
CCAFS SLC-40
```

```
KSC LC-39A
```

```
VAFB SLC-4E
```

Launch Site Names Begin with 'CCA'

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

```
In [6]: %sql select * from spacex where launch_site like 'CCA%' limit 5
```

* ibm_db_sa://fhb66868:***@824dfd4d-99de-440d-9991-629c01b3832d.bs2io90108kqb1od8l1cg.databases.appdomain.cloud:30119/bludb Done.

```
Out[6]:
```

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 45596 using the query below.

```
In [7]: %sql select sum(payload_mass_kg_) as Total_payload_mass from spacex where customer = 'NASA (CRS)'
```

* ibm_db_sa://fhh66868:***@824dfd4d-99de-440d-9991-629c01b3832d.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30119/bludb
Done.

```
Out[7]: total_payload_mass
```

45596

Average Payload Mass by F9 v1.1

We calculate the average payload mass carried by booster version F9 v1.1 as 2928.

```
In [8]: %sql select avg(payload_mass_kg_) from spacex where booster_version = 'F9 v1.1'
* ibm_db_sa://fhh66868:***@824dfd4d-99de-440d-9991-629c01b3832d.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30119/bludb
Done.
Out[8]: 1
2928
```


First Successful Ground Landing Date

We use the min() function to find the result.

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

```
In [9]: %sql select min(date) from spacex where landing_outcome = 'Success (ground pad)'
```

```
* ibm_db_sa://fhb66868:***@824dfd4d-99de-440d-9991-629c01b3832d.bs2io90108kqb1od8lcg.databases.appdomain.cloud:30119/bludb  
Done.
```

```
Out[9]:      1  
        2015-12-22
```

Successful Drone Ship Landing with Payload between 4000 and 6000

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

```
In [10]: %sql select booster_version from spacex where payload_mass_kg_ between 4000 and 6000 and landing_outcome='Success (drone shi
* ibm_db_sa://fhb66868:***@824dfd4d-99de-440d-9991-629c01b3832d.bs2io90108kqb1od8lcg.databases.appdomain.cloud:30119/bludb
Done.
Out[10]: 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 Mission_Outcome was a success or a failure.

```
In [11]: %sql select mission_outcome, count(mission_outcome) as count_of_mission from spacex group by mission_outcome

* ibm_db_sa://fhb66868:***@824dfd4d-99de-440d-9991-629c01b3832d.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:30119/bludb
Done.
```

```
Out[11]:
```

mission_outcome	count_of_mission
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

Boosters Carried Maximum Payload

```
In [12]: %sql select booster_version, payload_mass_kg_ from spacex where payload_mass_kg_ = (select max(payload_mass_kg_) from spacex)
```

```
* ibm_db_sa://fhh66868:***@824dfd4d-99de-440d-9991-629c01b3832d.bs2io90108kqb1od8l1cg.databases.appdomain.cloud:30119/bludb
Done.
```

```
Out[12]:
```

booster_version	payload_mass_kg_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

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

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 in year 2015.

```
In [13]: %sql select date, booster_version, launch_site , landing_outcome from spacex where landing_outcome = 'Failure (drone ship)'
```

* ibm_db_sa://fhb66868:***@824dfd4d-99de-440d-9991-629c01b3832d.bs2io90108kqb1od8lcg.databases.appdomain.cloud:30119/bludb
Done.

```
Out[13]:
```

DATE	booster_version	launch_site	landing_outcome
2015-01-10	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
2015-04-14	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

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

```
In [14]: %sql SELECT LANDING_OUTCOME, COUNT(landing_outcome) AS COUNT_LAUNCHES FROM SPACEX WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
```

* ibm_db_sa://f9b66868:***@824dfd4d-99de-440d-9991-629c01b3832d.bs2io90108kqb1od8lcg.databases.appdomain.cloud:30119/bludb
Done.

```
Out[14]:
```

landing_outcome	count_launches
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

We selected Landing outcomes (such as Failure (drone ship) or Success (ground pad) from the data and used the WHERE clause to filter for landing outcomes between the date 2010-06-04 and 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 on descending order.

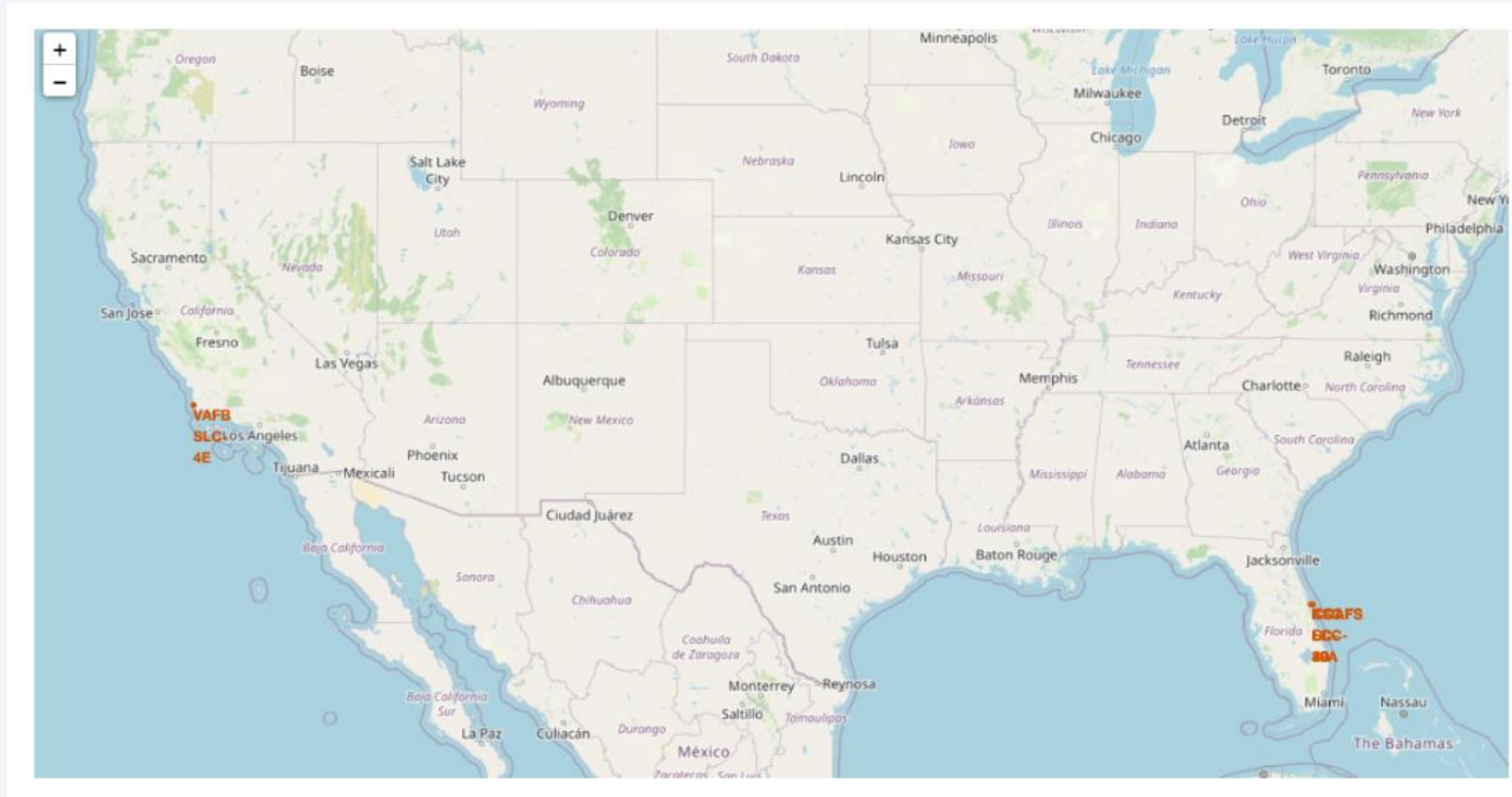
A satellite view of Earth from space, showing the curvature of the planet and city lights at night. The background is a deep blue gradient.

Section 3

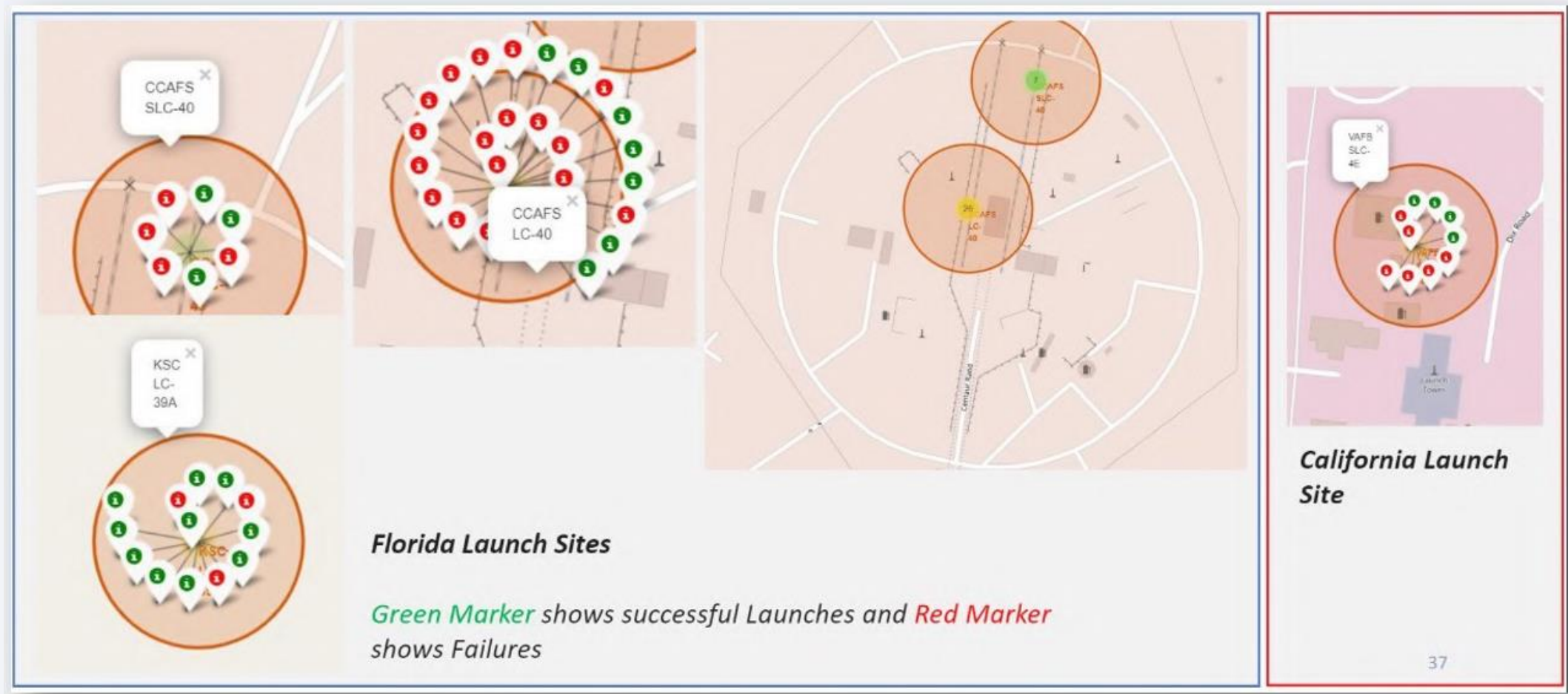
Launch Sites Proximities Analysis

Location of all the Launch Sites

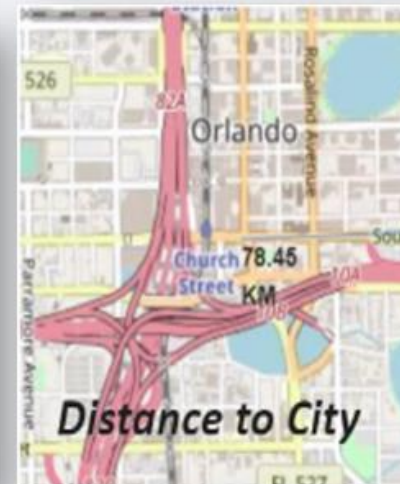
We can see that all the SpaceX launch sites are located inside the United States



Markers showing launch sites with color labels



Launch Sites Distance to Landmarks



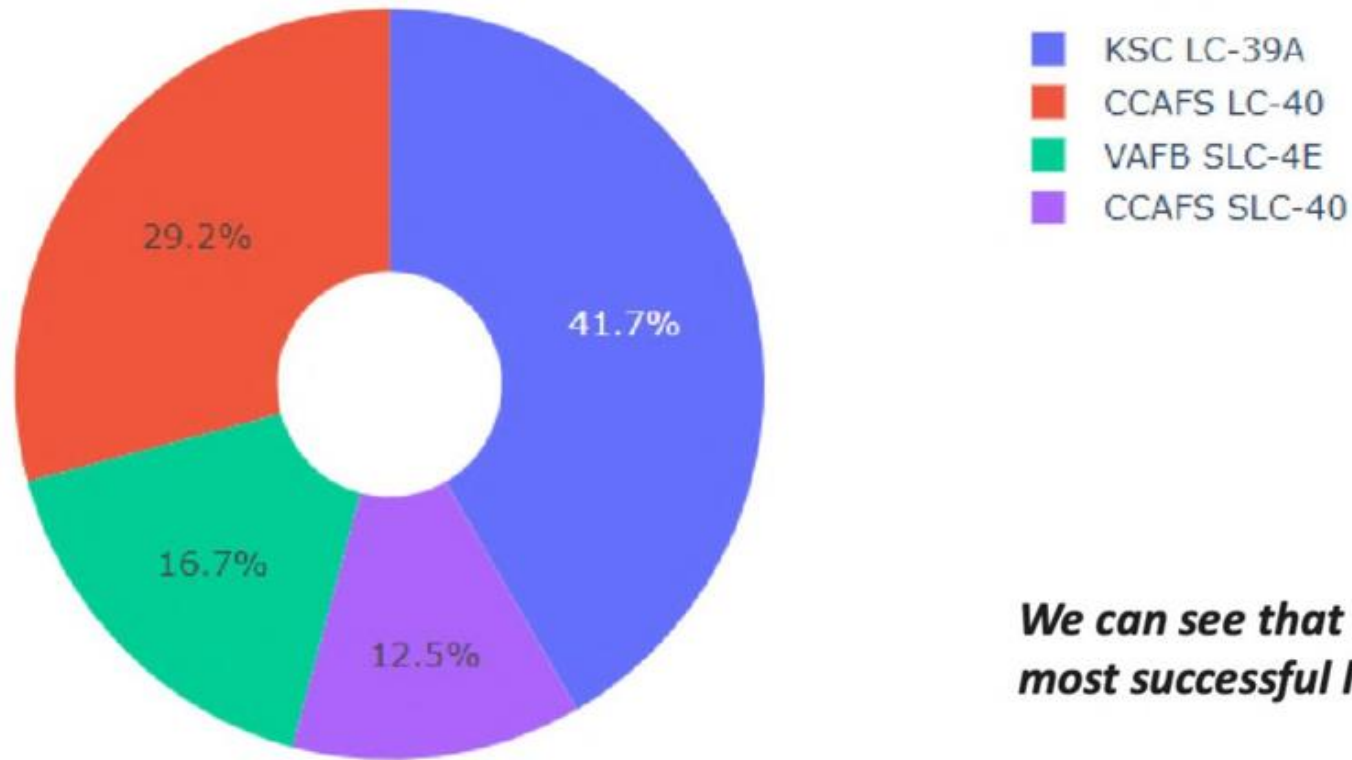
- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes



Section 4

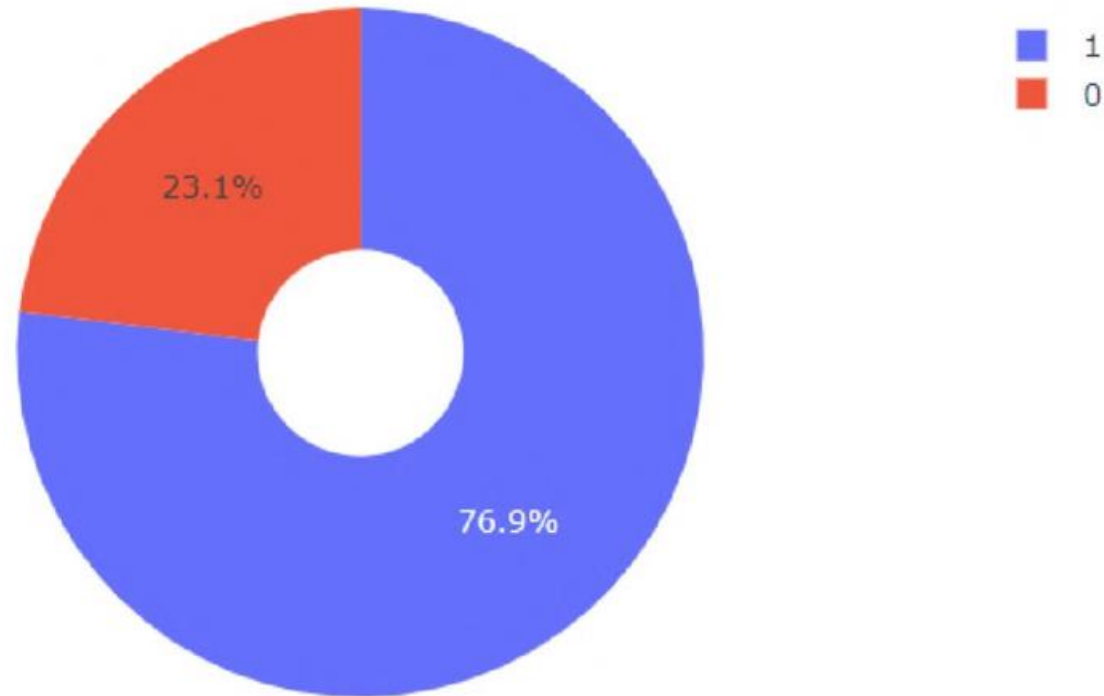
Build a Dashboard with Plotly Dash

The Success Percentage by each sites



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

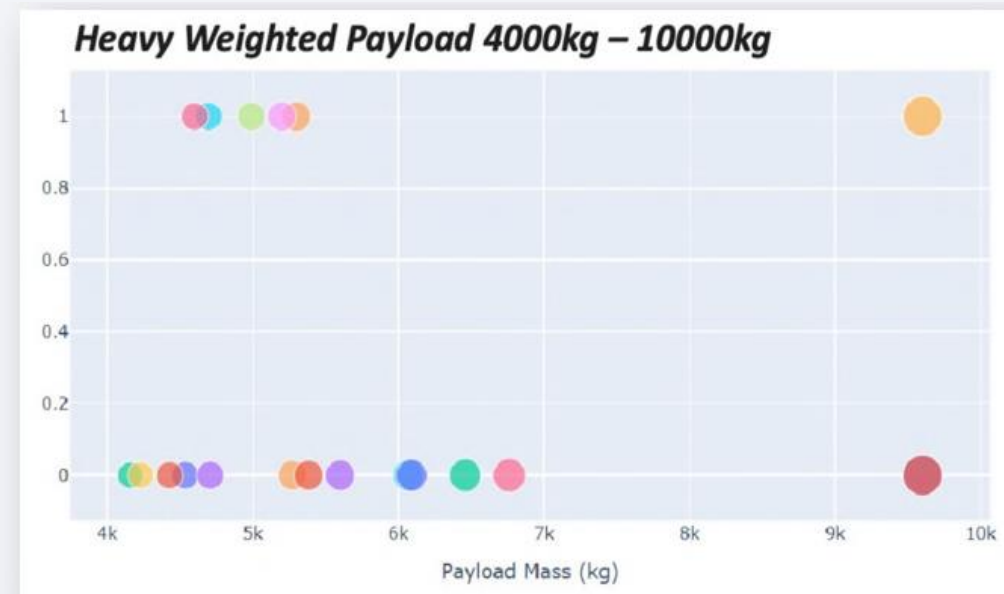
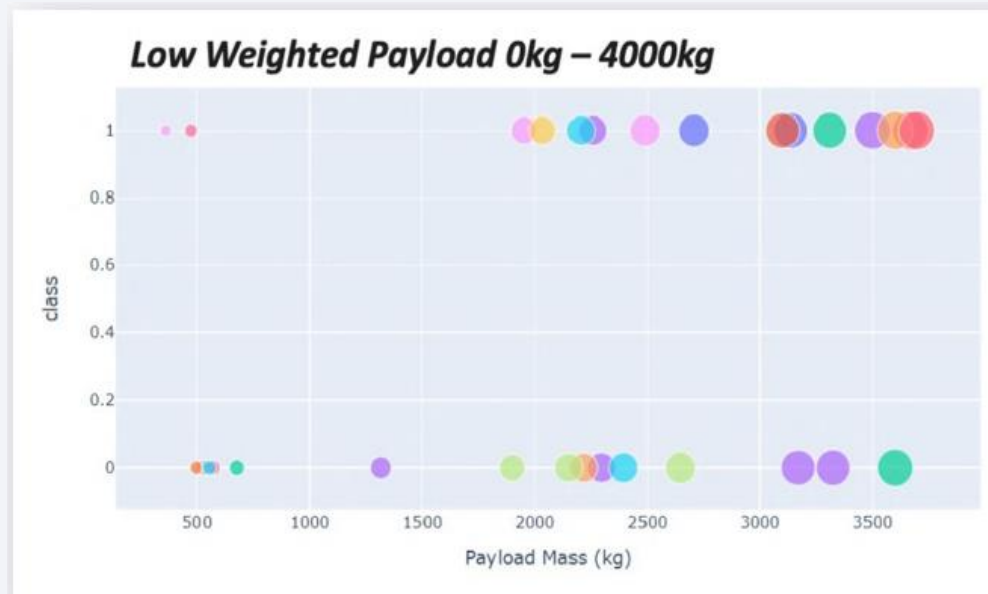
The Highest Launch-Success ratio : KSC LC-39A



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

Payload vs Launch Outcome Scatter Plot

We can see that all the success rate for low weighted payload is higher than heavy weighted payload





Section 5

Predictive Analysis (Classification)

Classification Accuracy

As we can see, by using the code as below: We could identify that the best algorithm to be the Tree Algorithm which have the highest classification accuracy.

In [30]:

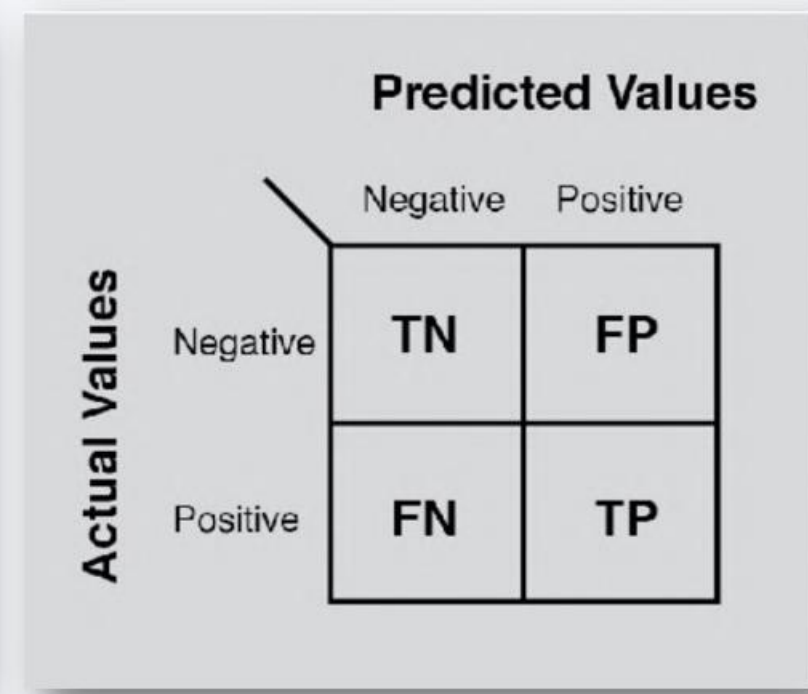
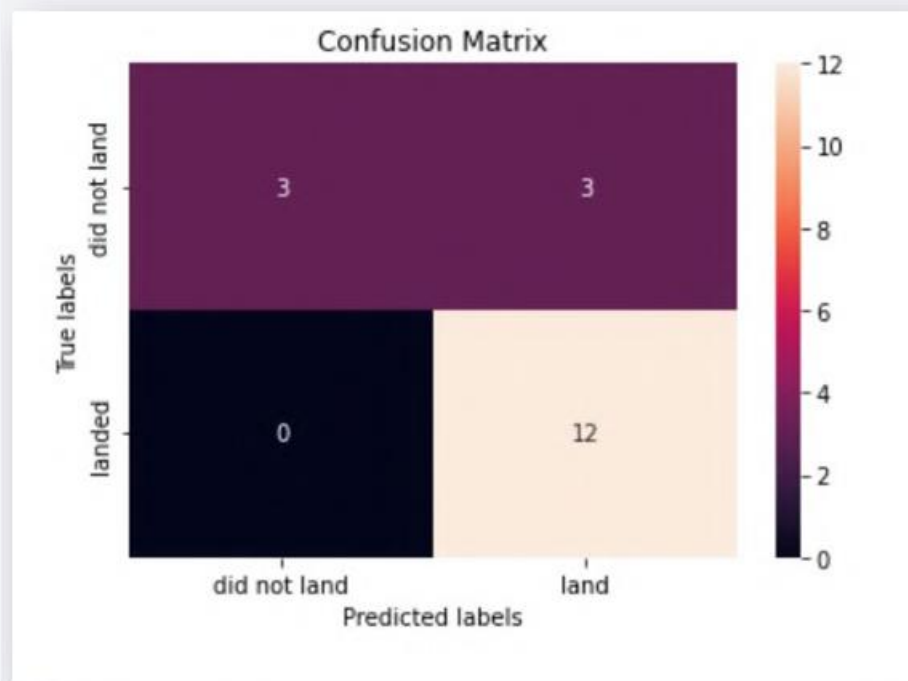
```
algorithms = {'KNN':knn_cv.best_score_, 'Tree':tree_cv.best_score_, 'LogisticRegression':logreg_cv.best_score_}
bestalgorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
if bestalgorithm == 'Tree':
    print('Best Params is :',tree_cv.best_params_)
if bestalgorithm == 'KNN':
    print('Best Params is :',knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best Params is :',logreg_cv.best_params_)
```

Best Algorithm is Tree with a score of 0.8857142857142856

Best Params is : {'criterion': 'gini', 'max_depth': 18, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 5, 'splitter': 'best'}

Confusion Matrix

The confusion matrix of 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 Tree Classifier Algorithm is the best Machine Learning approach for this dataset.
- The low weighted payloads (which define as 4000kg and below) performed better than the heavy weighted payloads.
- Starting from the year 2013, the success rate for SpaceX launches is increased, directly proportional time in years to 2020, which it will eventually perfect the launches in the future.
- KSC LC-39A have the most successful launches of any sites; 76.9%
- SSO orbit have the most success rate; 100% and more than 1 occurrence.

Thank you!

