IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Name>
<Date>

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection using SQL, API, Web Scraping
  - Data Wrangling : Describe how data was processed
  - EDA with Data Visualization
  - EDA with SQL
  - Interactive Visual Analytics by creating maps with folium
  - Interactive Dashboard using Plotly
  - Predictive Analytics using Classification Algorithms
- Summary of all results
  - Exploratory Data Analysis Result
  - Interactive Analytics Demo
  - Predictive Analytics Result

Introduction

- Project background and context
  - The goal of this project is to predict if the first stage will land successfully.
  - SpaceX Falcon 9 rocket launches costs 62 million dollars; other providers cost upward of 165 million dollars each.
  - SpaceX is able to save this amount as it can reuse the first stage. Thus if we can predict if first stage will launch we can use this information to predict the cost of the launch.

- Problems you want to find answers
  - Which factors influence successful landing of rocket.
  - The relationship between each variable features and how it affects the outcome.
  - What conditions are needed to be achieved for increasing the probability of successful landing.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Via SpaceX API and Webscraping from Wikipedia
- Perform data wrangling:
  - Performed data cleaning processes like remove irrelevant columns.
  - 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:
  - Build, evaluate and tune classification models

Data Collection

- Data is gathered from SpaceX API which contains details about the rocket launch.
- It includes details rocket used, payload delivered, launch specifications, landing specifications, and landing outcome.
- Data is also gathered from Wikipedia with Webscraping using Beautifulsoup.
- For REST API, it starts with a 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 replaced those values with appropriate ones.
- For web scrapping, we will use the BeautifulSoup to extract the launch records as HTML table, parse the table and converted 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 Data Wrangling

```
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 a
nd 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 rocket
s with 2 extra rocket boosters and rows that have multiple payloads in a
single rocket.
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 s
ingle 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 ex
tracting 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 Wikipedia page from url

Create a BeautifulSoup object from the HTML response

Extract all column/variable names from the HTML header

Extract all column/variable names from the HTML header

Creation of dictionary and appending data to keys

Converting dictionary to Dataframe

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

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

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plai
nrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding t
o launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
```

Data Wrangling

Calculate number of launches at each site

Calculate number and occurrences of each orbit

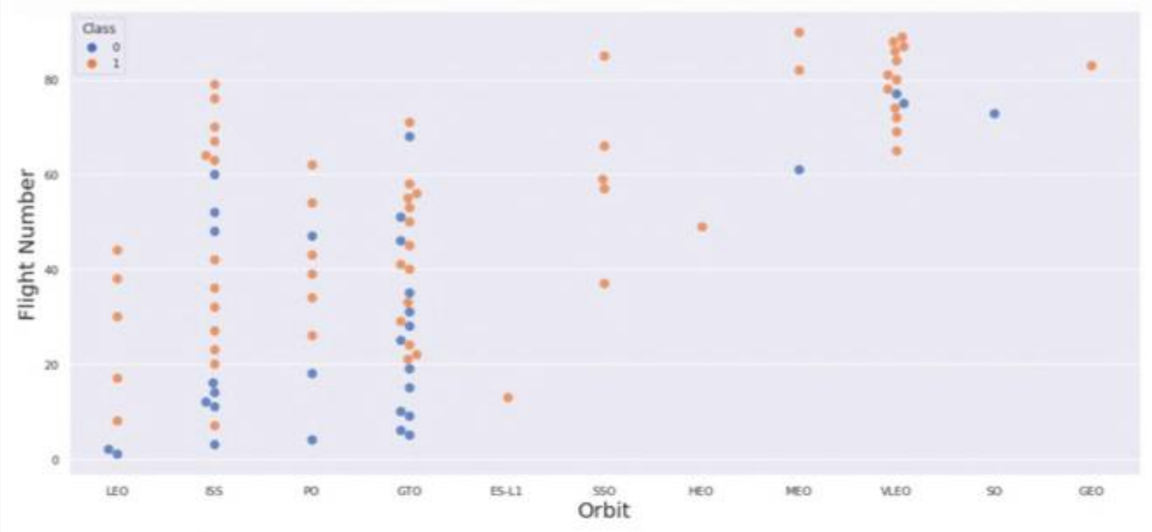
Calculate number and occurrences of mission outcome

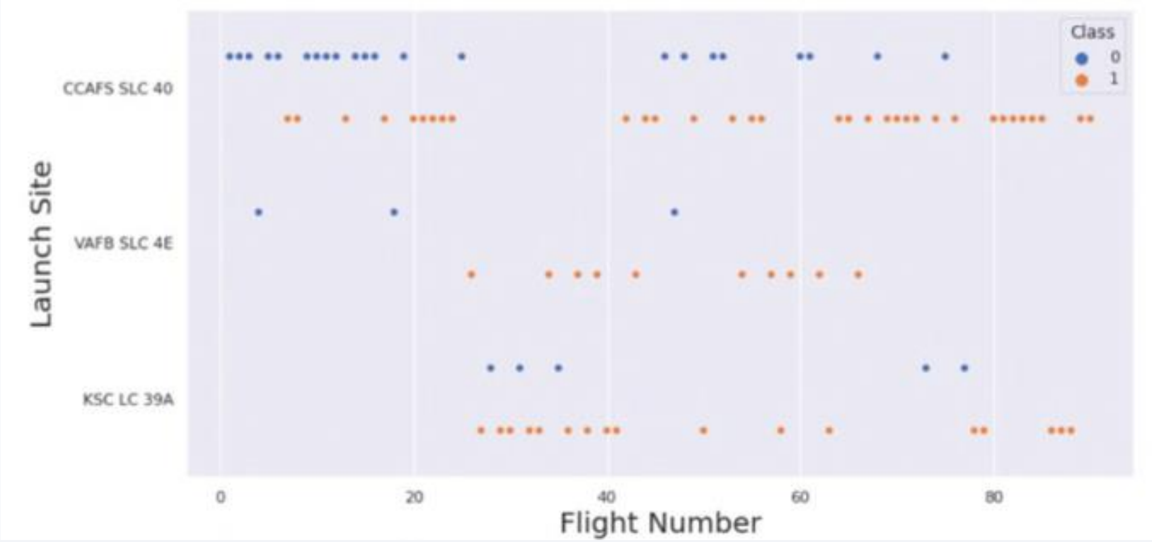
Create landing outcome label from outcome column

Export as .csv

- Data Wrangling is the process of cleaning and unifying messy and complex data sets for easy access and Exploratory Data Analysis (EDA).
- We calculated the number of launches on each site, then calculated 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. Lastly, we will export the result to a CSV.

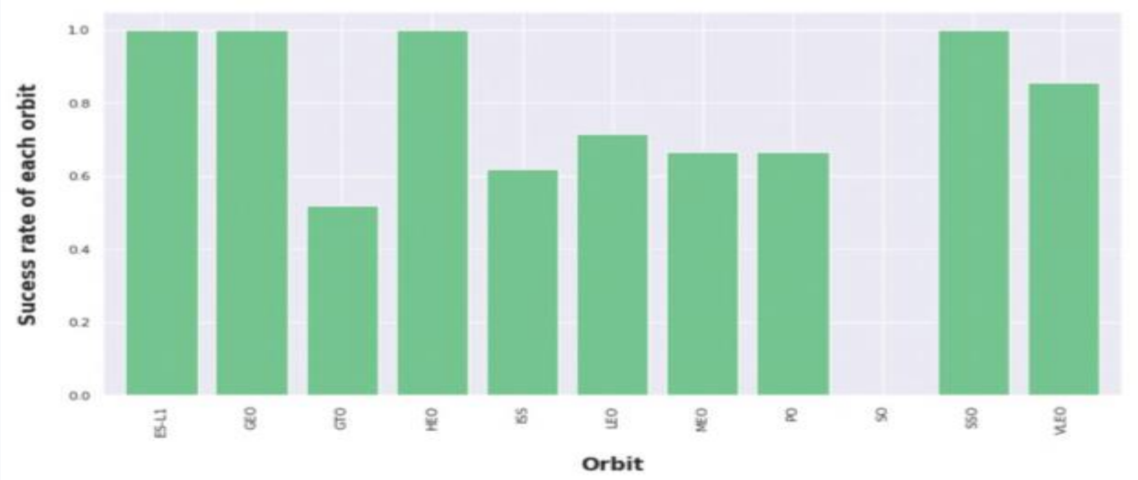
EDA with Data Visualization





- We use 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. It's very easy to see which factors affecting the most to the success of the landing outcomes.

EDA with Data Visualization



- We will then use further visualization tools such as bar graph and line plots graph for further analysis.
- Bar graph is used 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 and it is used for see the launch success yearly trend.

EDA with SQL

- Performed EDA with SQL to get insight from the data.
- Wrote SQL Queries to extract following Data:
  - The names of unique launch sites in the space mission.
  - The total payload mass carried by boosters launched by NASA (CRS)
  - The average payload mass carried by booster version F9 v1.1
  - The total number of successful and failure mission outcomes
  - The failed landing outcomes in drone ship, their booster version and launch site names.

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 assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.
- We calculated the distances between a launch site to its proximities. And looked for certain information:
  - Are launch sites near railways, highways and coastlines.
  - Do launch sites keep certain distance away from cities.

Build a Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly dash.
- We plotted pie charts showing the total launches by a certain site
- We then plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.

Predictive Analysis (Classification)

Building the model

- Load the dataset into NumPy and Pandas
- Transform the data and then split into training and test datasets
- Select ML Algorithms to use
- set the parameters and algorithms to GridSearchCV and fit it to dataset.

Evaluating the Model

- Check the accuracy for each model
- Perform Hyperparameter Tuning
- plot the confusion matrix.

Improving the Model

- Use Feature Engineering and Algorithm Tuning

Find the Best Model

- Comparing accuracy scores of different algorithms.

Results

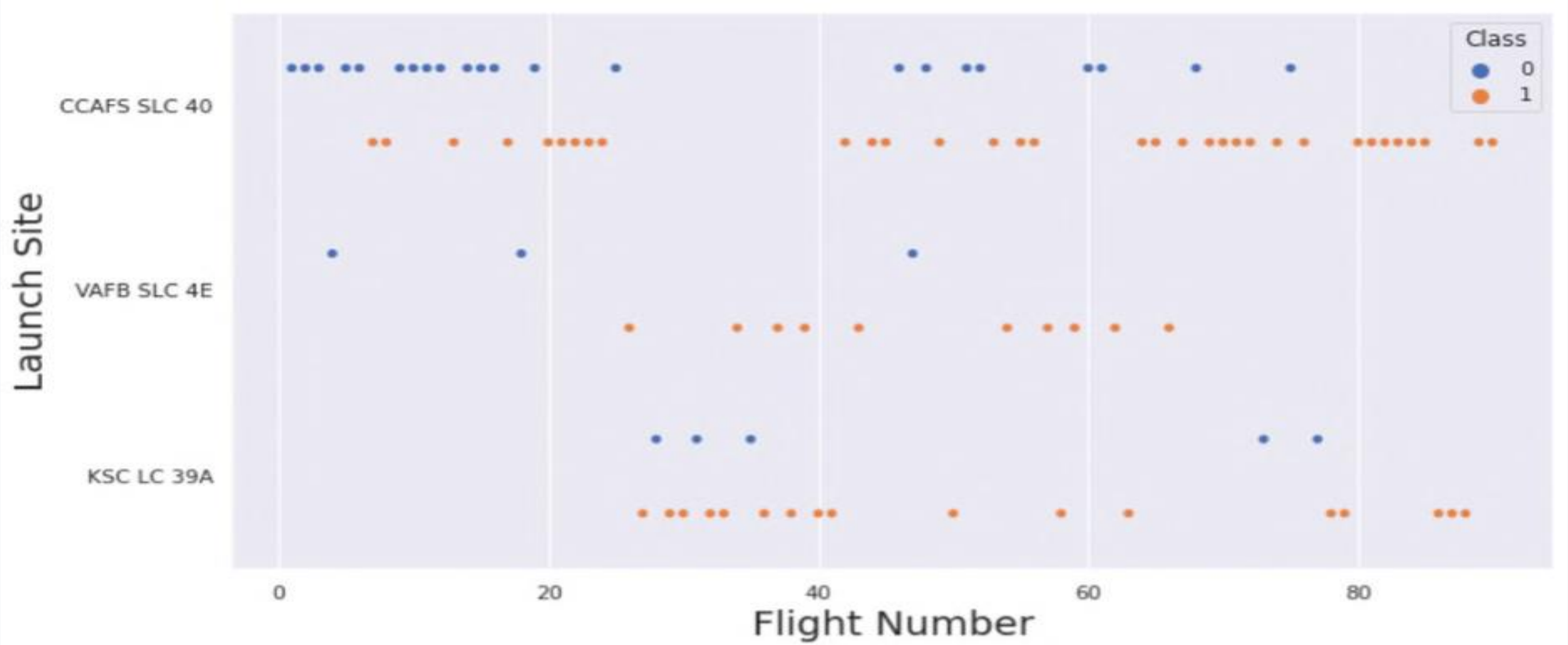
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

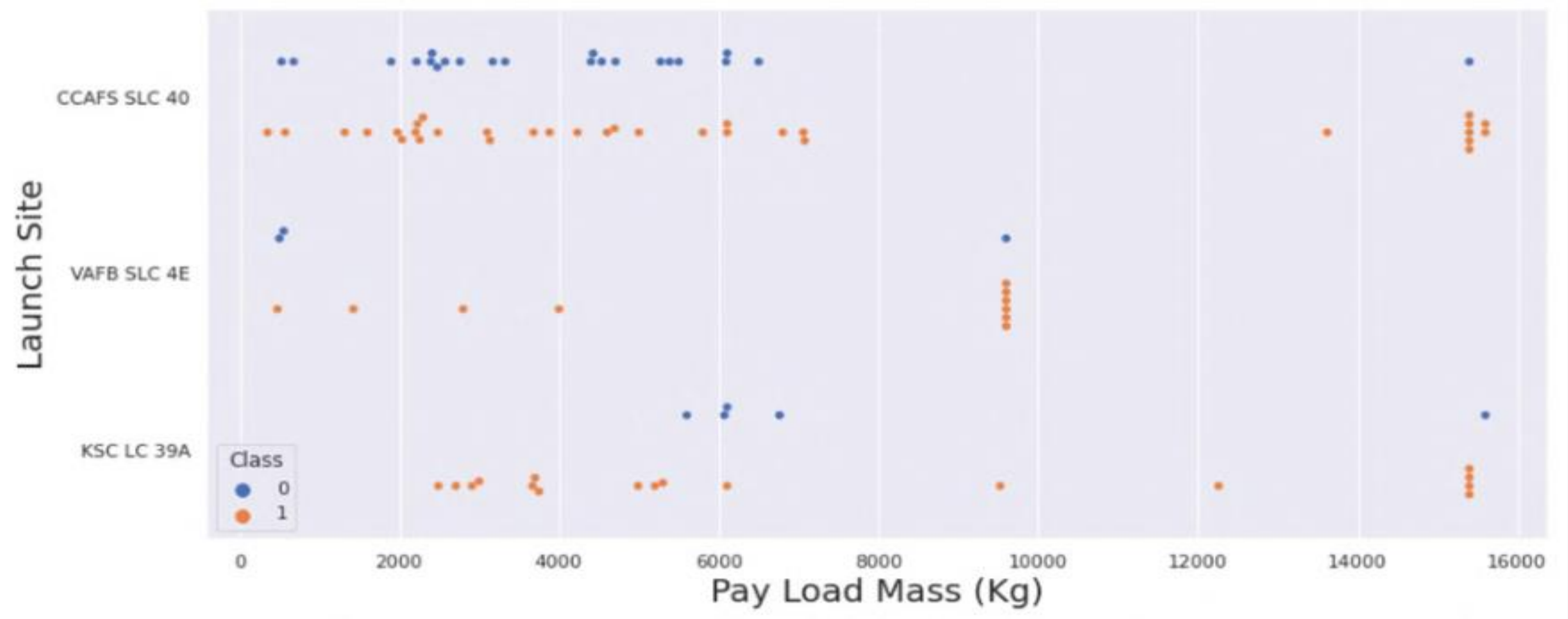
Flight Number vs. Launch Site

- With higher flight number the success rate for rocket is increasing.
- But this pattern can't be seen for CCAFS SLC 40.



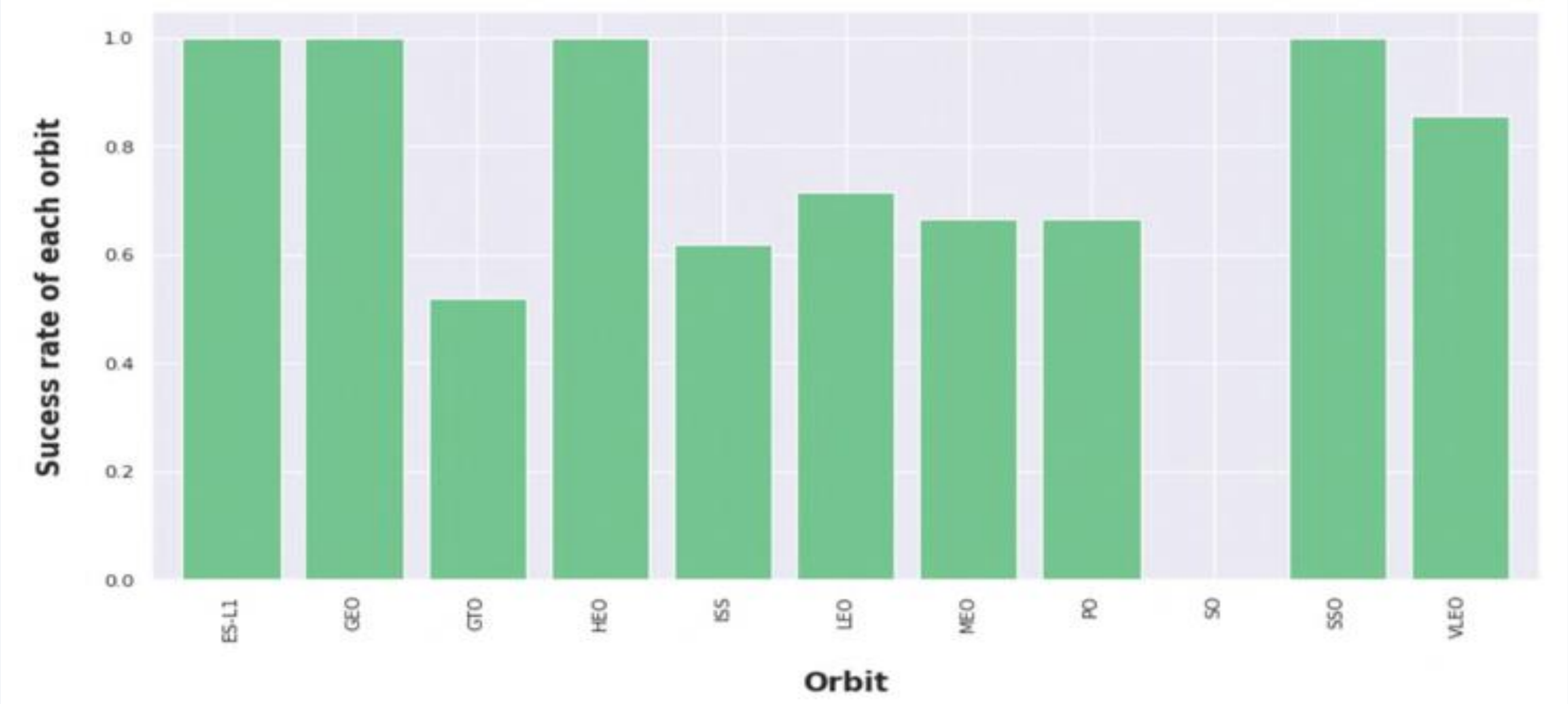
Payload vs. Launch Site

- The greater the payload mass for Launch Site the higher the success rate for the Rocket.
- There is not quite a clear pattern to be found using this visualization to make a decision if the Launch Site is dependent on Pay Load Mass for a success launch.



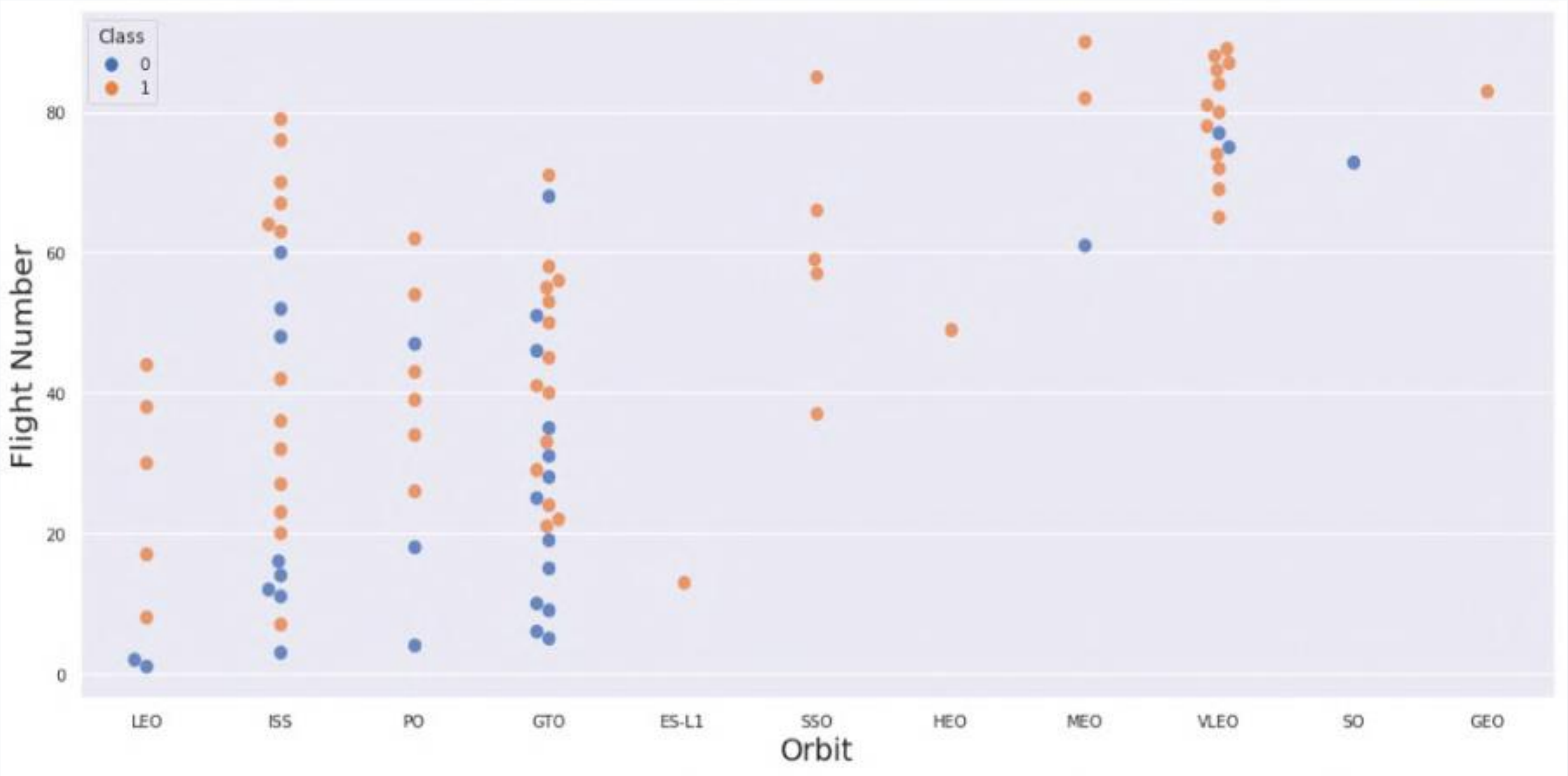
Success Rate vs. Orbit Type

- We can see that landing outcomes for some orbits has 100% success rate such as SSO, HEO, GEO AND ES-L1 while SO orbit produced 0% rate of success.



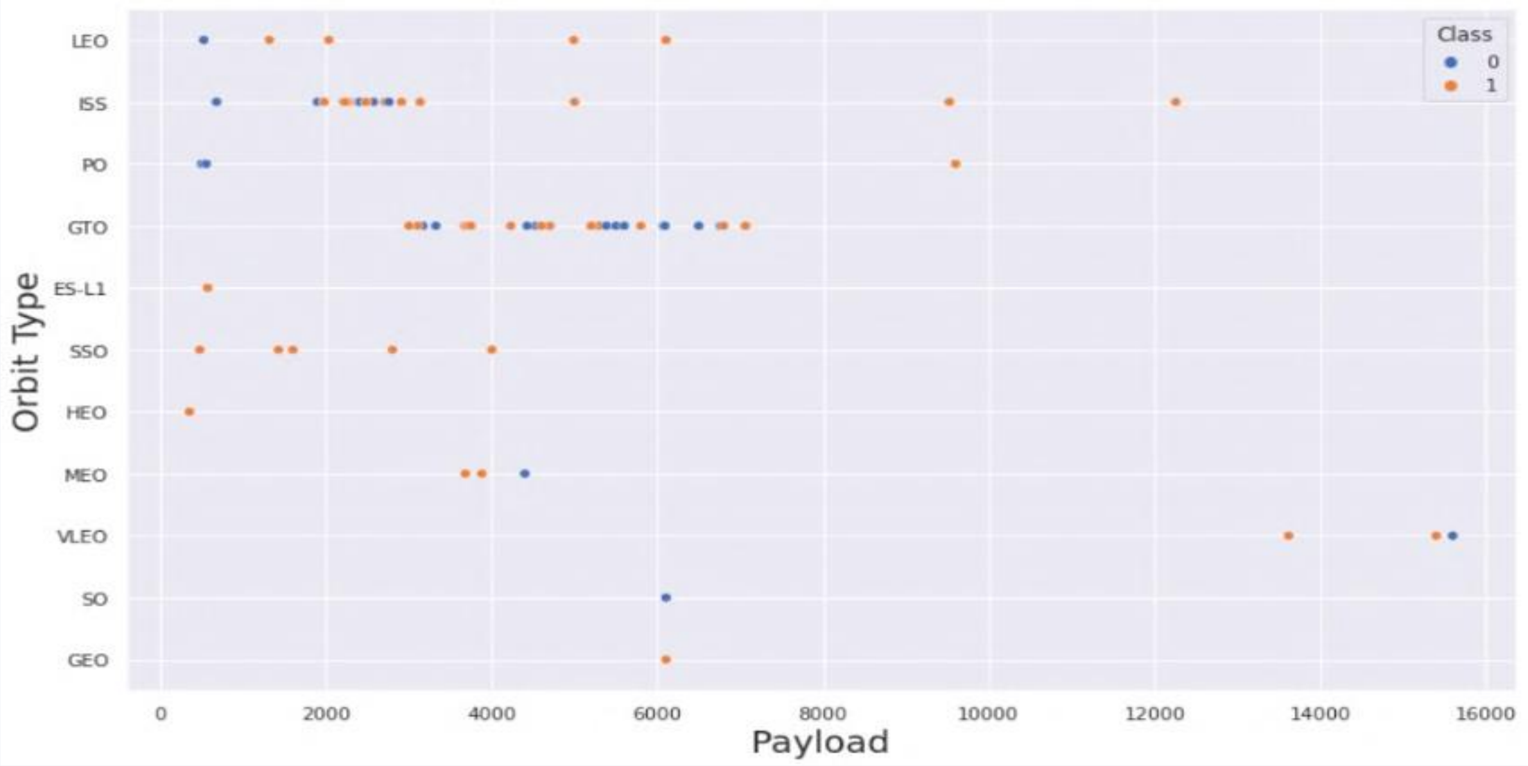
Flight Number vs. Orbit Type

- We can see that 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.



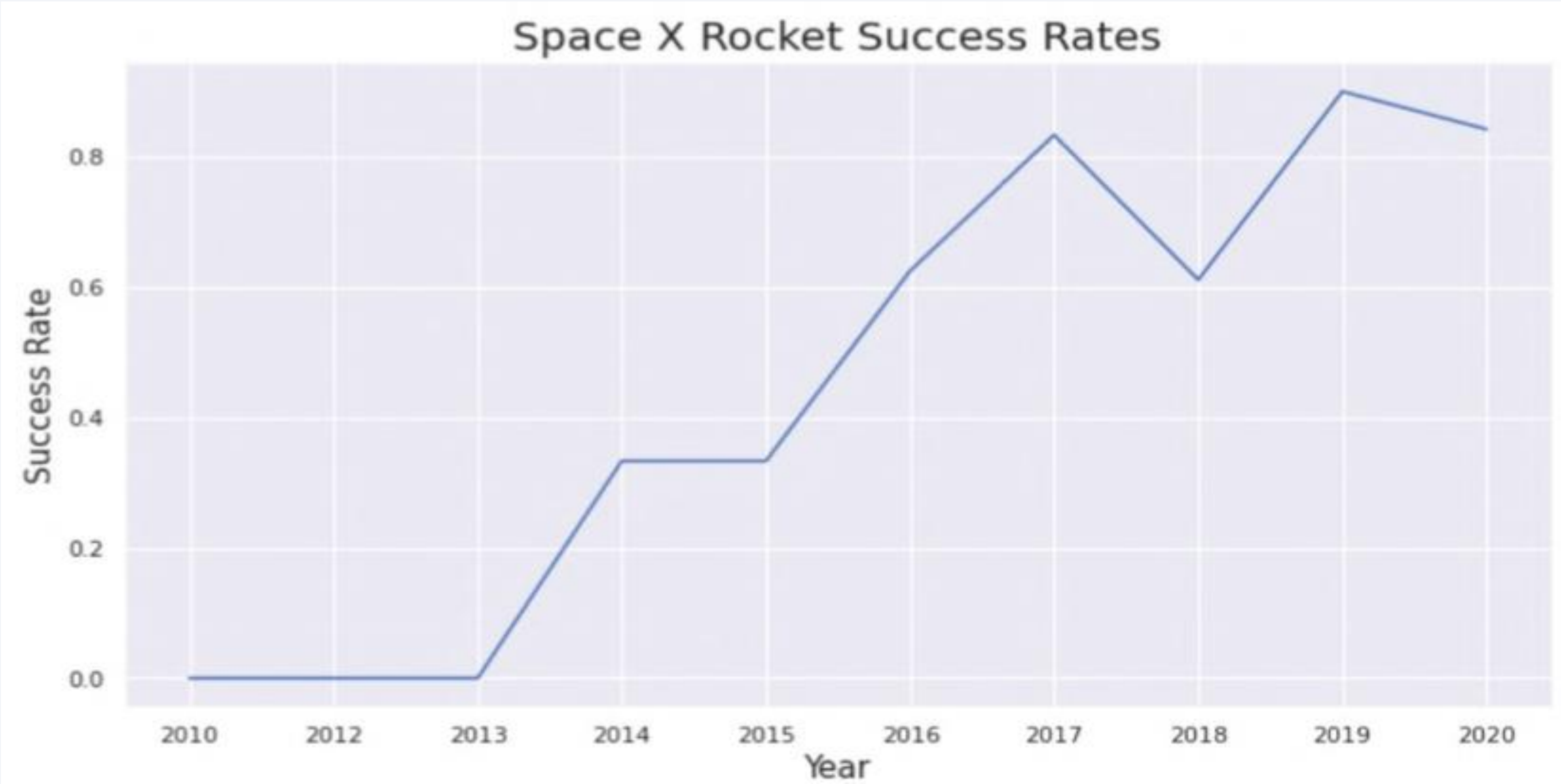
Payload vs. Orbit Type

- As the payload increases it has a positive impact on LEO, ISS and P0 orbit, whereas it has a negative impact on MEO and VLEO orbit.
- SO, GEO and HEO orbit need more dataset to see any pattern or trend.



Launch Success Yearly Trend

- It can be seen that there has been an increasing trend from the year 2013 till 2020.



All Launch Site Names

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

```
[7]: %sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTBL;

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

[7]:

Launch_Sites
CCAFS LC-40
VAFB SLC-4E
KSC LC-39A
CCAFS SLC-40

Launch Site Names Begin with 'CCA'

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

Task 2

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

```
[8]: %sql SELECT * FROM SPACEXTBL where Launch_Site like 'CCA%' limit 5
```

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

[8]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	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)
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt

Total Payload Mass

- We calculated the total payload carried by boosters using the query below

Task 3

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

```
[9]: %sql SELECT SUM(PAYLOAD_MASS__KG_) As "Total Payload Mass Carried" from SPACEXTBL where Customer = "NASA (CRS)"
```

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

[9]:

Total Payload Mass Carried
45596

Average Payload Mass by F9 v1.1

- We calculated the average payload mass carried by booster version F9 v1.1

Task 4

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

```
[10]: %sql SELECT AVG(PAYLOAD_MASS__KG_) as "Average Payload Mass" from  SPACEXTBL where Booster_Version = "F9 v1.1"

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

[10]:

Average Payload Mass
2928.4

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 1st May 2017

Task 5

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

Hint:Use min function

```
[35]: %sql Select Min("Date") as "First Successful Landing Outcome" from SPACEXTBL \
      where "Landing _Outcome" = 'Success (ground pad)'

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

[35]:

First Successful Landing Outcome
01-05-2017

Successful Drone Ship Landing with Payload between 4000 and 6000

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

Task 6

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

```
[37]: %sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE "Landing _Outcome" = 'Success (drone ship)' \
AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;
```

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

[37]:

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 LIKE ‘%’ to filter for **WHERE** Mission_Outcome was a success or a failure.

Task 7

List the total number of successful and failure mission outcomes

[47]:
```
%sql SELECT COUNT(MISSION_OUTCOME) AS "Successful Mission" FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE 'Success%';
```

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

[47]:

Successful Mission
100

[48]:
```
%sql SELECT COUNT(MISSION_OUTCOME) AS "Successful Mission" FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE 'Failure%';
```

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

[48]:

Successful Mission
1

Boosters Carried Maximum Payload

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

Task 8

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

[28]:
```
%sql SELECT Distinct Booster_Version from SPACEXTBL where PAYLOAD_MASS__KG_ = (Select max(PAYLOAD_MASS__KG_) from SPACEXTBL)
```

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

[28]:

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

2015 Launch Records

- We used a combinations of the **WHERE** clause, **LIKE**, **AND**, and **BETWEEN** conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015

Task 9

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Note: 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.

[67]:
```
%sql Select Booster_Version, Launch_site from SPACEXTBL where "Date" LIKE '%-2015' \
and "Landing _Outcome" = "Failure (drone ship)"
```

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

[67]:

Booster_Version	Launch_Site
F9 v1.1 B1012	CCAFS LC-40
F9 v1.1 B1015	CCAFS LC-40

Rank Landing Outcomes Between 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 in descending order.

Task 10

Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

```
[73]: %sql SELECT "Landing _Outcome", count("Landing _Outcome") as 'Total Count' from SPACEXTBL Where Date between '04-06-2010' \
and '20-03-2017' group by "Landing _outcome" ORDER by count("Landing _Outcome") DESC
```

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

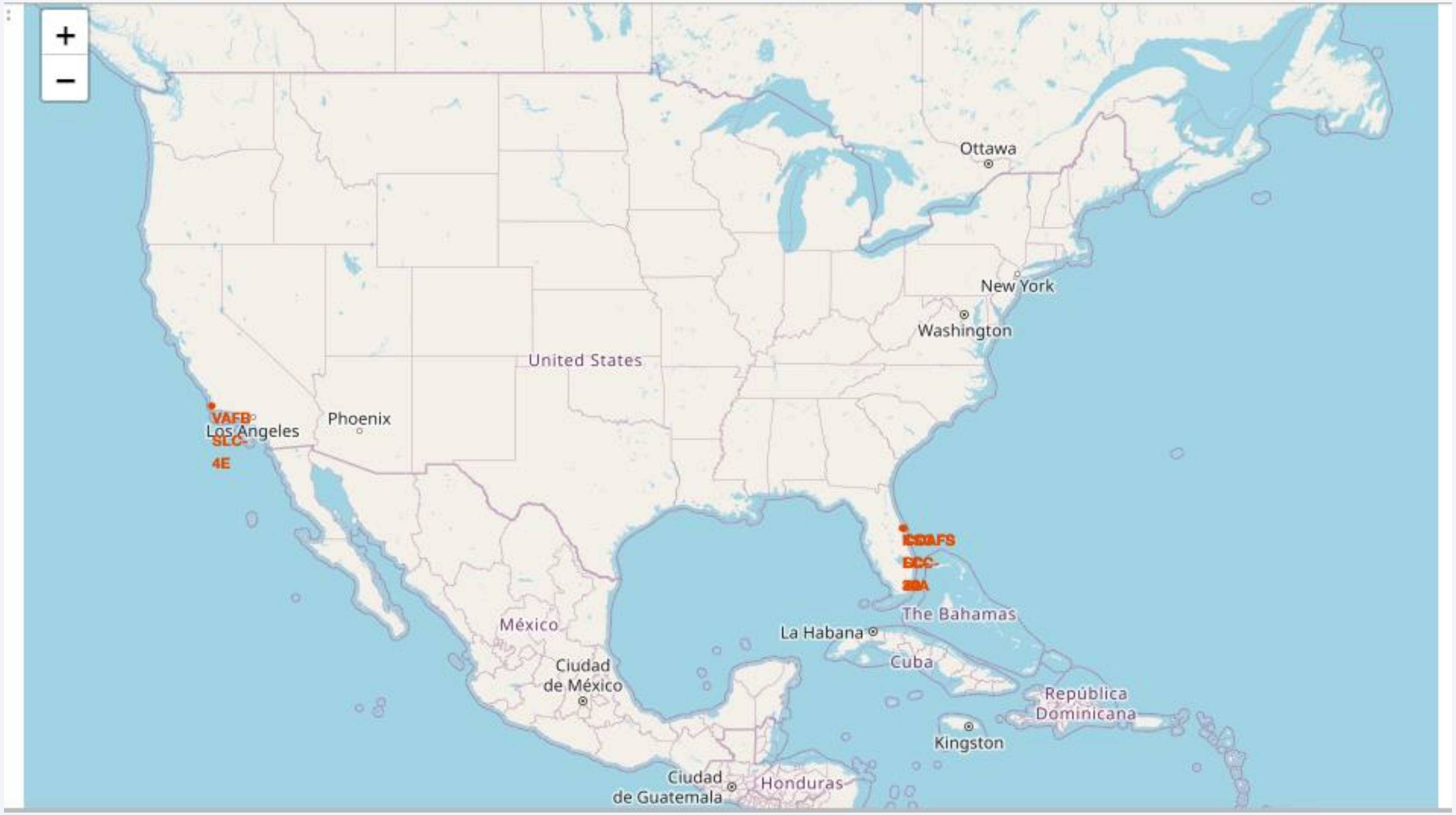
[73]:

Landing _Outcome	Total Count
Success	20
No attempt	10
Success (drone ship)	8
Success (ground pad)	6
Failure (drone ship)	4
Failure	3
Controlled (ocean)	3
Failure (parachute)	2
No attempt	1

Section 3

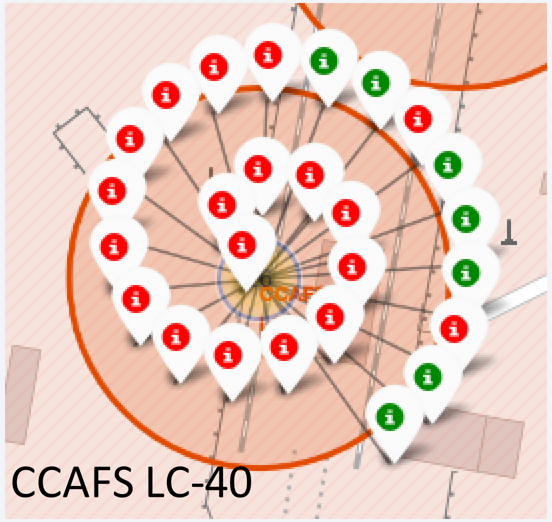
Launch Sites Proximities Analysis

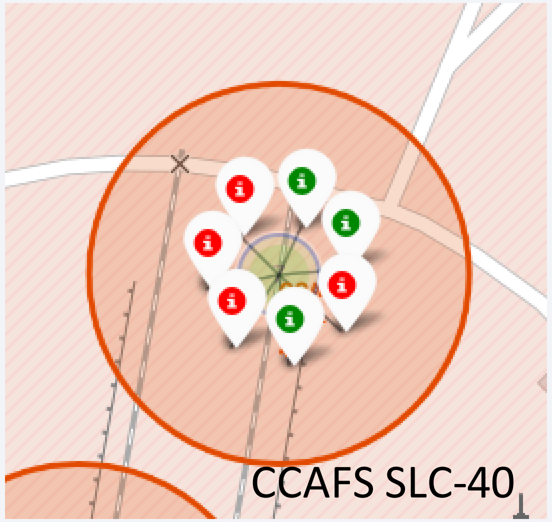
Location of all Launch Sites

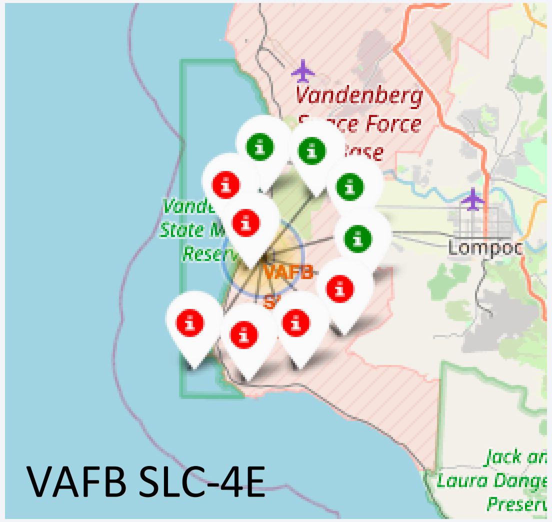


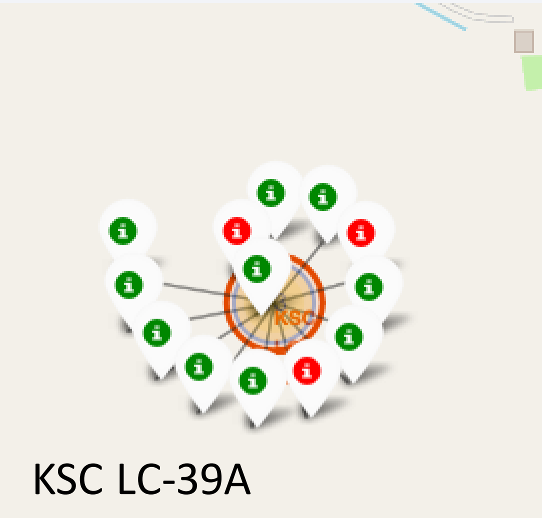
- We can see that all the SpaceX launch sites are located inside the United States

Markers showing Launch Sites with Color Labels







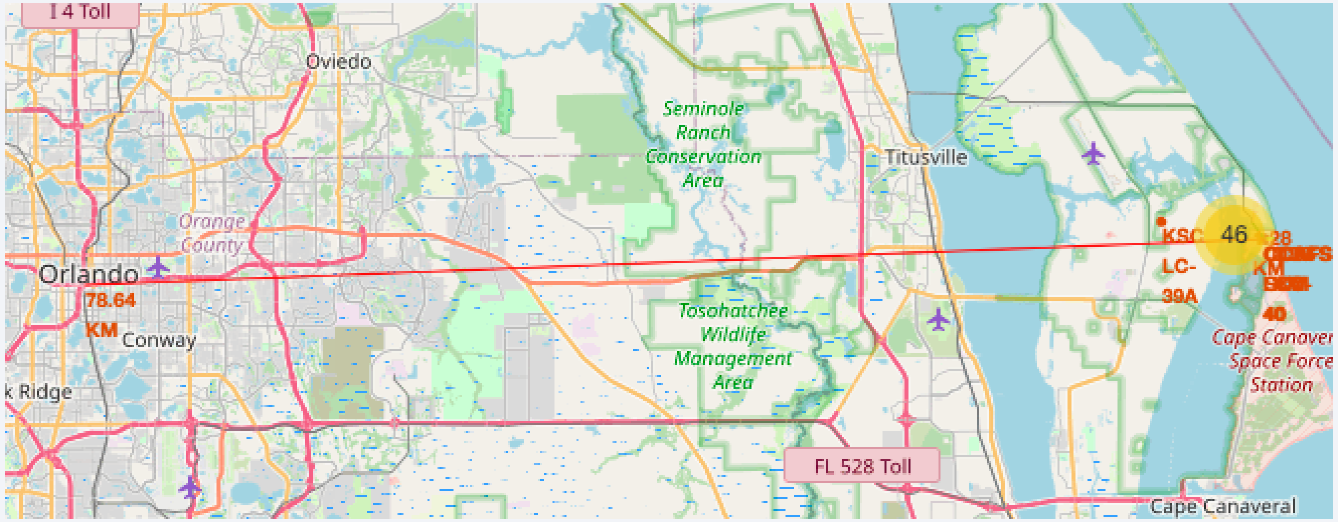


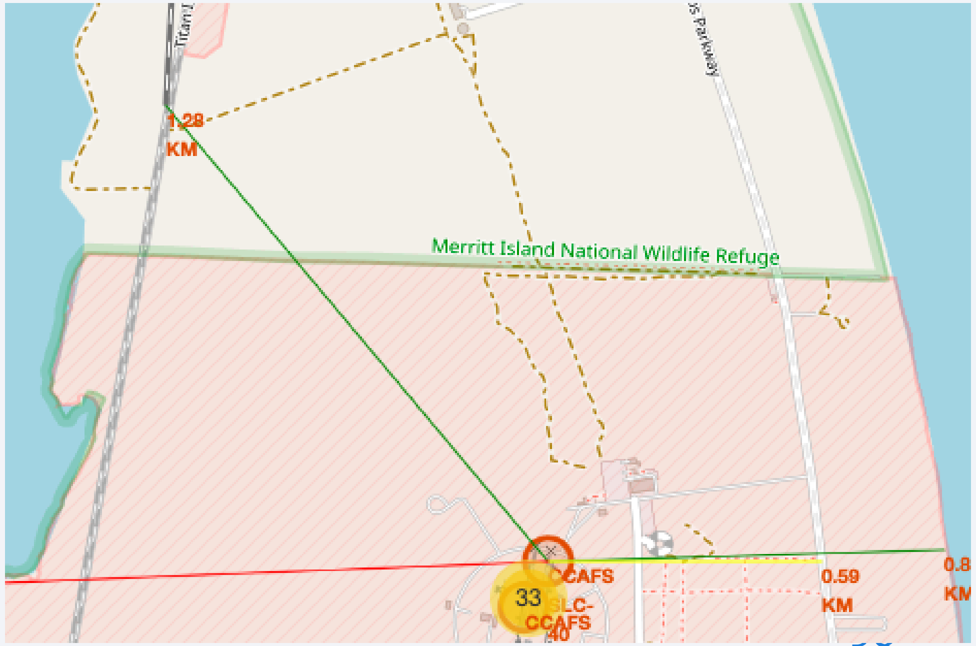
Florida Launch Sites

California Launch Sites

- *Green Markers* show successful launches
- *Red Markers* show Failures.

Launch Sites Distances from Landmarks



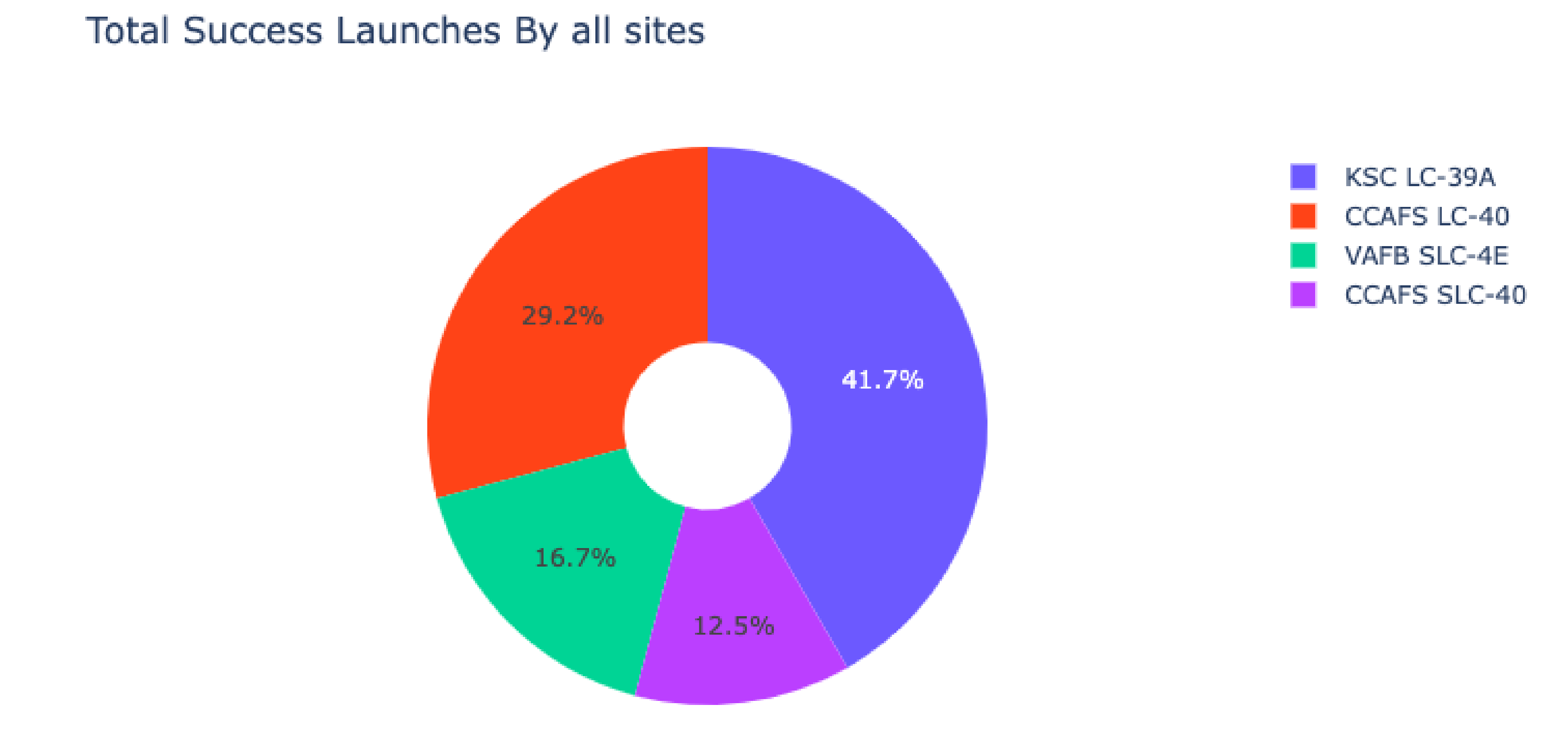


- All Launch Sites are in close proximity to the Coastline.
- All Launch Sites are away from the City and are also away from the Railways and Highways.

Section 4

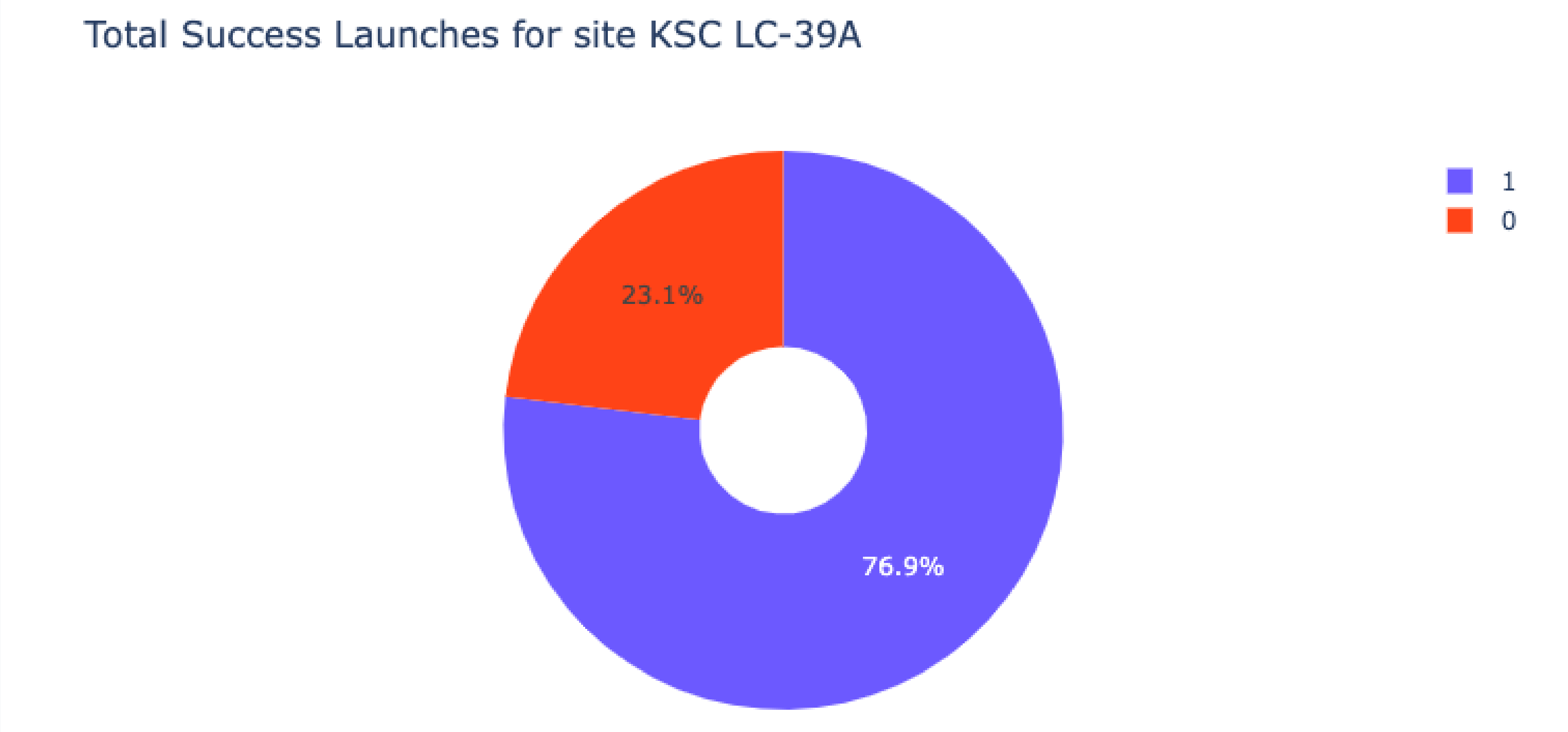
Build a Dashboard with Plotly Dash

<Dashboard Screenshot 1>



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

<Dashboard Screenshot 2>



- KSC LC-39A had a success rate of 76.9% and a failure rate of 23.1%.

<Dashboard Screenshot 3>

- We can see that success rate for lower payload is higher than for heavier payload.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

- We can observe that Decision Tree is the best performing algorithm with an accuracy of 88.75%

TASK 12

Find the method performs best:

```
[39]: algorithms = {'KNN':knn_cv.best_score_,'Decision 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 == 'Decision 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 Decision Tree with a score of 0.8875
Best Params is : {'criterion': 'entropy', 'max_depth': 2, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 1
0, 'splitter': 'random'}
```

Confusion Matrix

```
[29]: yhat = tree_cv.predict(X_test)
      plot_confusion_matrix(Y_test,yhat)
```

- This is the confusion matrix for Decision tree algorithm.
- The classifier can distinguish between different classes.
- Major problem is False positive, i.e, Unsuccessful landings classified as successful landings by the classifier.

Conclusions

We can conclude that:

- Success rates have increased from 2013 and there has been steady increase till 2020.
- Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
- However, deeper analysis show that some of this orbits has only 1 occurrence such as GEO, SO, HEO and ES-L1 which mean this data need more dataset to see pattern or trend before we draw any conclusion.
- KSC LC-39A have the most successful launches of any sites; 76.9%.
- The low weighted payloads (less than 4000kg) performed better than the heavy weighted payloads.
- The Decision tree classifier is the best performing machine learning algorithm for classifying landing outcomes.

Thank you!