IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Ravipati Navya Sri
12th August 2023

Outline

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

Executive Summary

- **Summary of methodologies**

- ➢ Data Collection via API, SQL and Web Scraping
- ➢ Data Wrangling and Analysis
- ➢ Interactive Maps with Folium
- ➢ Predictive Analysis for each classification model

- **Summary of all results**

- ➢ Data Analysis along with Interactive Visualisations
- ➢ Best model for Predictive Analysis

Introduction

- **Project background and context**

Here we will predict if the Falcon 9 first stage will land successfully. SpaceX 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 SpaceX can reuse the first stage. Therefore, if we can determine if the first stage will land successfully. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.

- **Problems you want to find answers**

➢ With what factors, will rocket lands successfully?

➢ Effect of relationship of rocket variables on outcome

➢ Conditions which will aid SpaceX to achieve best results

Section 1

Methodology

Methodology

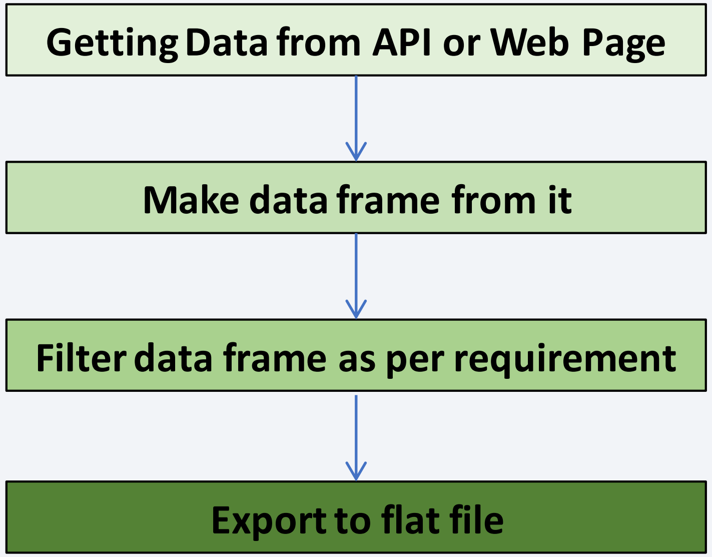
Executive Summary

- Data collection methodology:
  - Via SpaceX Rest API
  - Webscraping from Wikipedia
- Perform data wrangling
  - One hot encoding data fields for ML and dropping irrelevant columns(Transforming data for ML)
- Perform exploratory data analysis (EDA) using visualization and SQL
  - Scatter and bar graphs to show patterns between data
- Perform interactive visual analytics using Folium and Plotly Dash
  - Using Folium and Plotly Dash Visualisations
- Perform predictive analysis using classification models
  - Build and Evaluate classification models

Data Collection

- Data Collection is the process of gathering and measuring information related to targeted variables in an established system, which then enables one to answer relevant questions and evaluate outcomes.

STEPS INVOLVED:



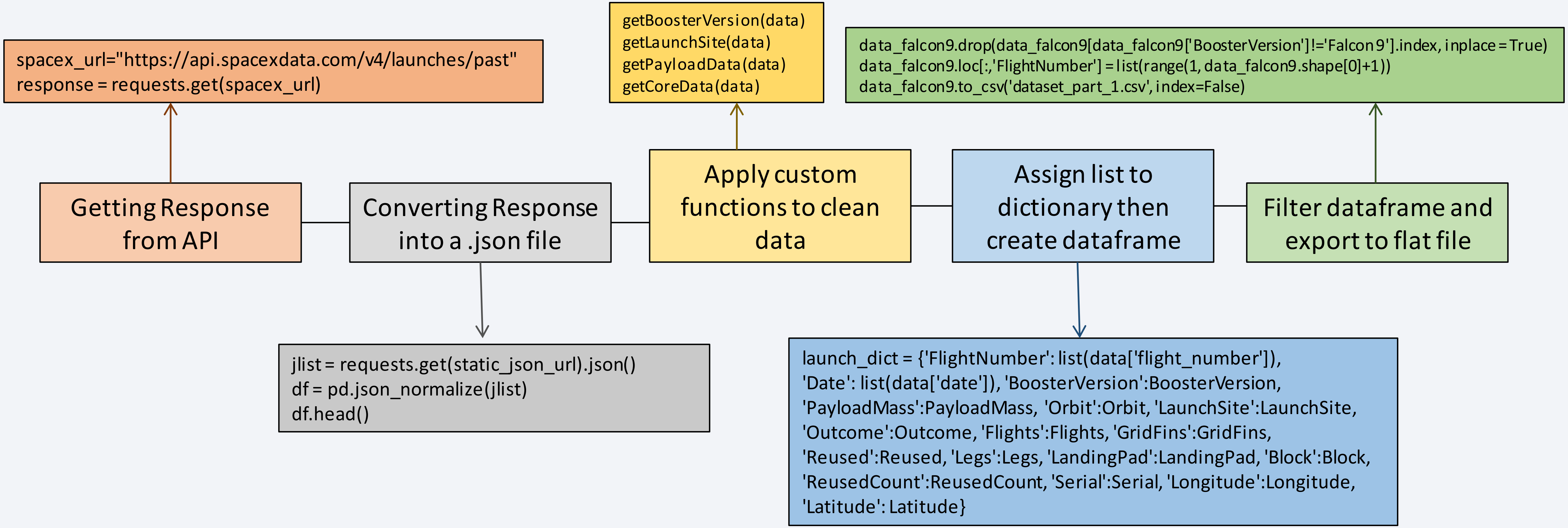
Getting Data from API or Web Page

Make data frame from it

Filter data frame as per requirement

Export to flat file

Data Collection – SpaceX API



Getting Response from API

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

Converting Response into a .json file

```
jlist = requests.get(static_json_url).json()
df = pd.json_normalize(jlist)
df.head()
```

Apply custom functions to clean data

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

Assign list to dictionary then create dataframe

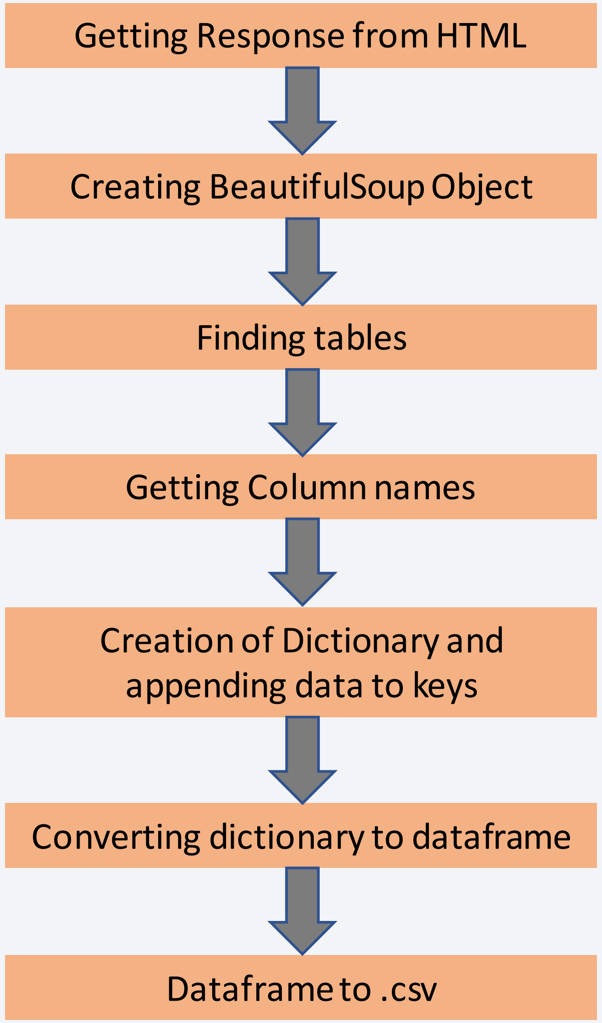
```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']), 'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass, 'Orbit':Orbit, 'LaunchSite':LaunchSite,
'Outcome':Outcome, 'Flights':Flights, 'GridFins':GridFins,
'Reused':Reused, 'Legs':Legs, 'LandingPad':LandingPad,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

Filter dataframe and export to flat file

```
data_falcon9.drop(data_falcon9[data_falcon9['BoosterVersion']!='Falcon 9'].index, inplace=True)
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Github URL

Data Collection - Scraping



Getting Response from HTML

Creating BeautifulSoup Object

Finding tables

Getting Column names

Creation of Dictionary and appending data to keys

Converting dictionary to dataframe

Dataframe to .csv

```
static_url = https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922
data = requests.get(static_url).text
```

```
soup = BeautifulSoup(data, ‘html5lib’)
```

```
html_tables = soup.find_all("table")
first_launch_table = html_tables[2]
```

```
ths = first_launch_table.find_all('th')
for th in ths:
    name = extract_column_from_header(th)
    if name is not None and len(name) > 0:
        column_names.append(name)
```

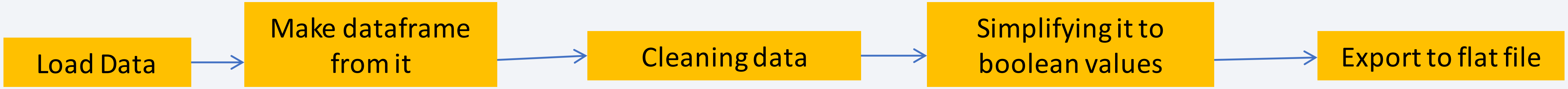
```
launch_dict = dict.fromkeys(column_names)
```

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

Github URL

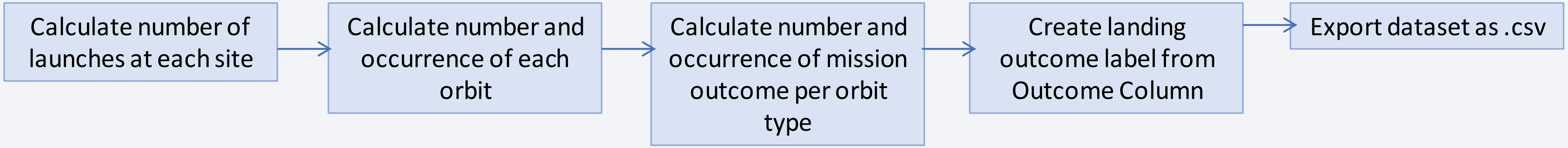
Data Wrangling

- Data wrangling is the process of cleaning and unifying messy and complex data sets for easy access and analysis.
- Here, those outcomes are marked as training label 1, if the booster successfully landed whereas for unsuccessful landings, marked as 0 (df['Class'] = df['Outcome'].apply(lambda landing_class: 0 if landing_class in bad_outcomes else 1))



Load Data → Make dataframe from it → Cleaning data → Simplifying it to boolean values → Export to flat file

STEPS FOLLOWED:



Calculate number of launches at each site → Calculate number and occurrence of each orbit → Calculate number and occurrence of mission outcome per orbit type → Create landing outcome label from Outcome Column → Export dataset as .csv

Github URL

EDA with Data Visualization

- Exploratory data analysis is an approach of analyzing data sets to summarize their characteristics, using statistical graphics and other data visualization methods.

STEPS FOLLOWED:

(1.) Load data (2.) Make dataframe from it (3.) Create Visualizations (4.) Collect Insights

Scatter Graphs Drawn:
Payload and Flight Number
Flight Number and Launch Site
Payload and Launch Site
Flight Number and Orbit Type
Payload and Orbit Type

It shows dependency of attributes on each other. Once the pattern is determined from graphs it's very easy to predict which factors will lead to maximum probability of success in both outcome and landing

Bar Graph Drawn:
Success Rate vs. Orbit Type

Bar graphs are easiest to interpret a relationship between attributes. Through this bar graph, orbits that have highest probability of success can be determined

Line Graph Drawn:
Launch Success yearly trend

Line graphs can show trends clearly and can aid in predictions for the future.

Github URL

EDA with SQL

SQL is most useful for data scientists as most of the real-world data is stored in databases. Besides being the standard language for Relational Database operations, it is also an incredibly powerful tool for analyzing data and drawing useful insights from it. We use IBM's Db2 for Cloud, which is a fully managed SQL Database provided as a service.

SQL Queries performed:

❑ Displaying the names of the unique launch sites in the space mission

❑ Display 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 F9v1.1

❑ Listing the date when the successful landing outcome in drone ship was achieved

❑ Listing the names of the boosters which have success in ground pad 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 year 2015

❑ Ranking the count of landing outcomes (such as Failure (droneship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

Github URL

Build an Interactive Map with Folium

Folium makes it easy to visualize data that's been manipulated in Python on an interactive leaflet map. We use the latitude and longitude coordinates for each launch site and added a Circle Marker around each launch site with a label of the name of the launch site. It is also easy to visualize the number of success and failure for each launch site with Green and Red markers on the map.

Map Objects	Code	Result
Map Marker	folium.Marker()	Map object to make a mark on map
Icon Marker	folium.Icon()	Create an icon on map
Circle Marker	folium.Circle()	Create a circle where Marker is being placed
PolyLine	folium.PolyLine()	Create a line between points
Marker Cluster Object	MarkerCluster()	This is a good way to simplify a map containing many markers having the same coordinate
AntPath	folium.plugins.AntPath()	Create an animated line between points

Github URL

Build a Dashboard with Plotly Dash

Pie Chart here shows the total success for all sites or by certain launch site

- Percentage of success in relation to launch site

Scatter Graph here shows the correlation between Payload and Success for all sites or by certain launch site

- It shows relation between Success rate and Booster Version Category

Map Objects	Code	Result

Dash and its components	import dash
	import dash_html_components as html
	import dash_core_componemts as dcc
	from dash.dependencies import input, Output
		Plotly stewards Python's leading data viz and UI libraries. With Dash Open Source, Dash apps run on local laptop or server. The Dash Core Component library contains a set of higher-level components like sliders, graphs, dropdowns, tables, and more.
		Dash provides all available HTML tags as user-friendly Python classes

Pandas	import pandas as pd	Fetching values from csv and creating a Dataframe
Plotly	import plotly.express as px	Plots the graphs with interactive plotly library
Dropdown	dcc.Dropdown()	Creates a dropdown for launch sites
Rangeslider	dcc.RangeSlider()	Creates a rangeslider for Payload Mass range selection
Pie Chart	px.pie()	Creates pie chart for success percentage display
Scatter Chart	px.scatter()	Creates Scatter chart for correlation display

Predictive Analysis (Classification)

BUILDING MODEL:

- Load our feature engineered data into dataframe
- Transform it into NumPy arrays
- Standardize and transform data
- Split data into training and test data sets
- Check how many test samples has been created
- List down machine learning algorithms we want to use
- Set our parameters and algorithms to GridSearchCV
- Fit our datasets into the GridSearchCV objects and train our model

```
y = data[‘Class’].to_numpy()

transform = preprocessing.StandardScaler()

X = transform.fit(X).transform(X)

X_train, X_test, Y_train, Y_test = train_test_split(X,y,test_size=0.2,random_state=2)

Y_test.shape
```

EVALUATING MODEL:

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

```
yhat = algorithm.predict(X_test)
plot_confusion_matrix(Y_test, yhat)
```

FINDING BEST PERFORMING CLASSIFICATION MODEL:

The model with best accuracy score is best performing model

```
algorithms={‘KNN’: knn_cv.best_score_, ‘Decision Tree’:
tree_cv.best_score_, ‘Logistic Regression’: logreg_cv.best_score_}
best_algorithm = max(algorithms, key = lambda x: algorithms[x])
```

BEST MODEL

Github URL

Results

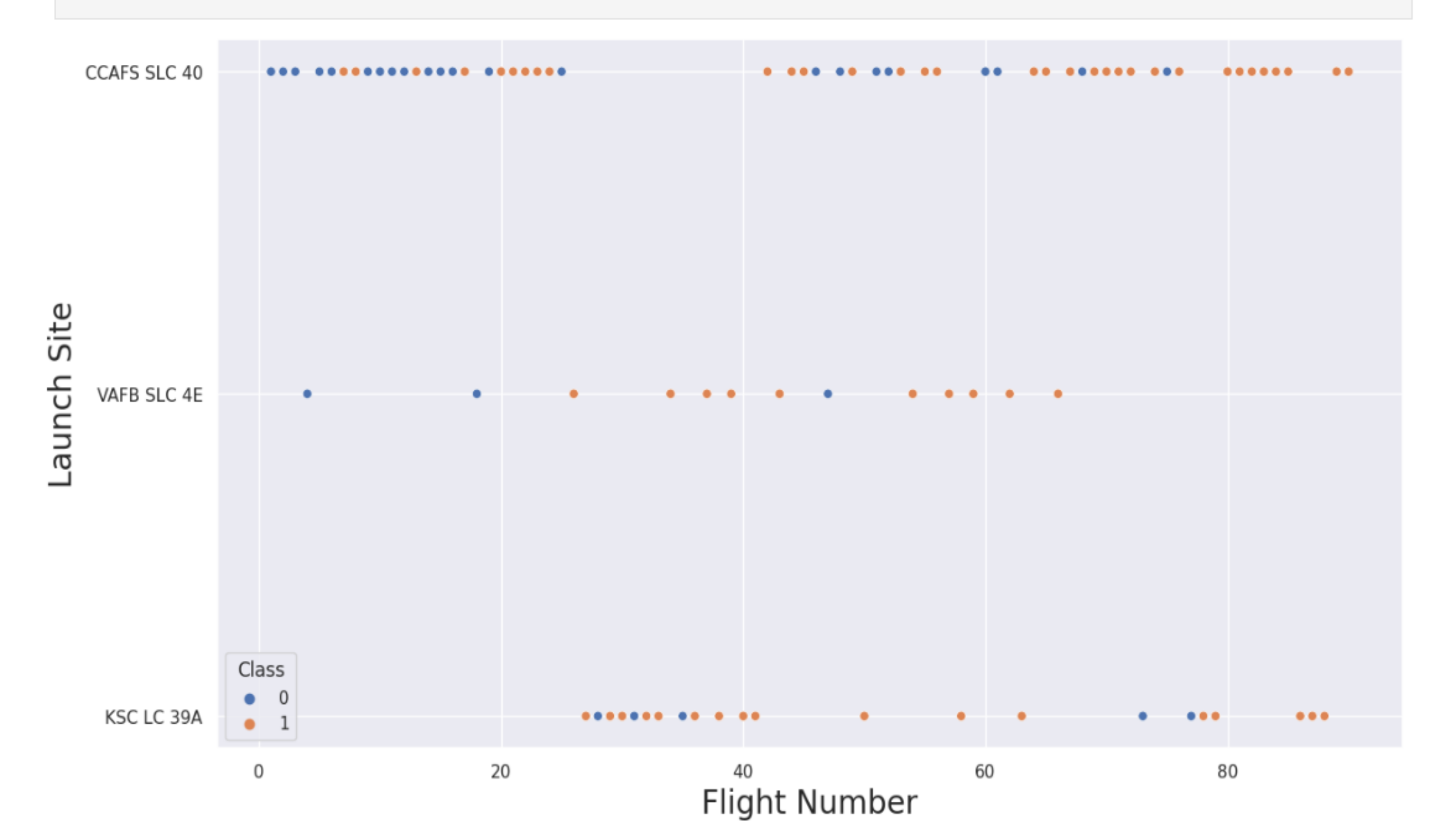
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

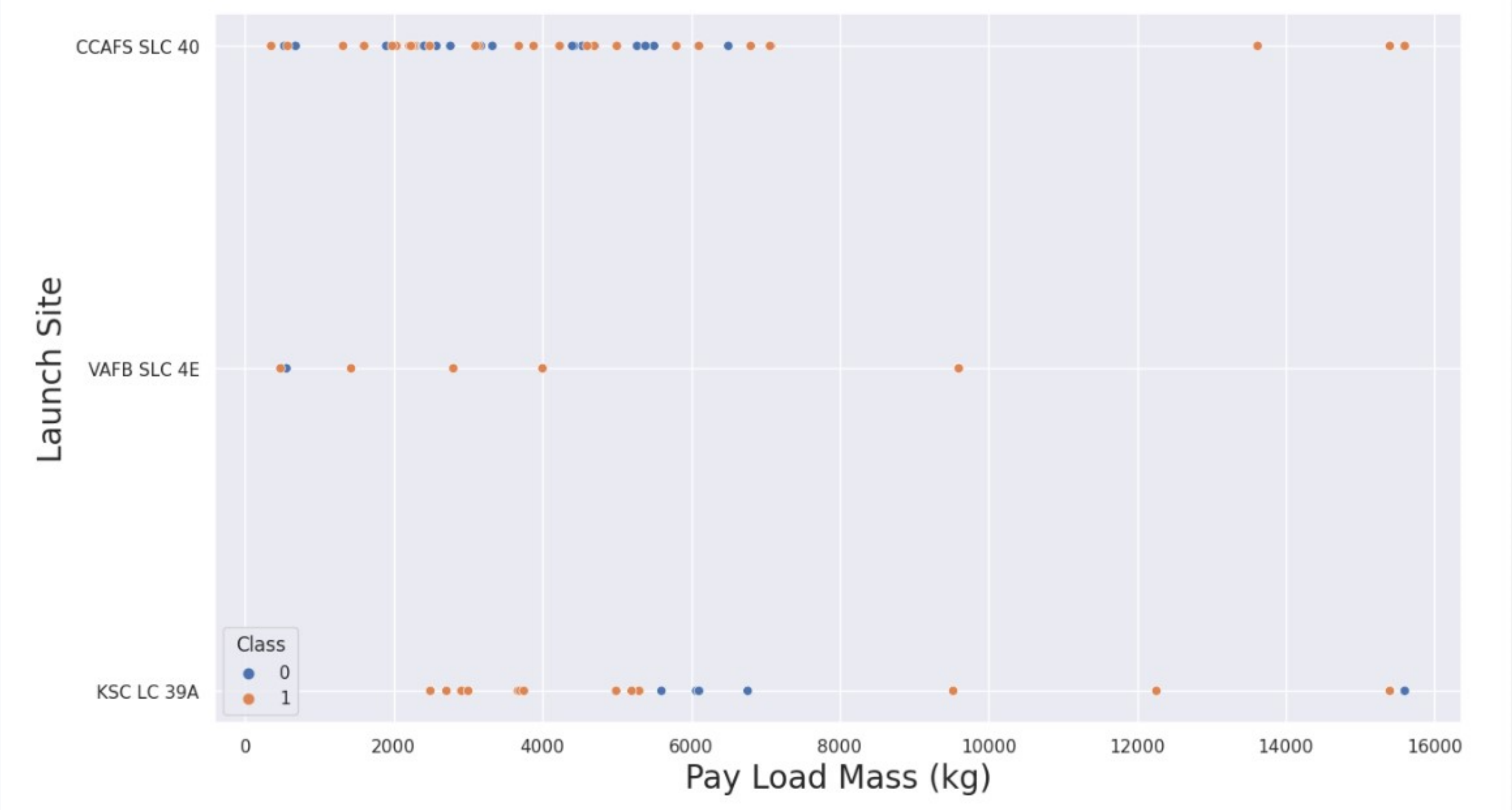
Flight Number vs. Launch Site

With higher flight numbers (greater than 30) the success rate for the Rocket is increasing



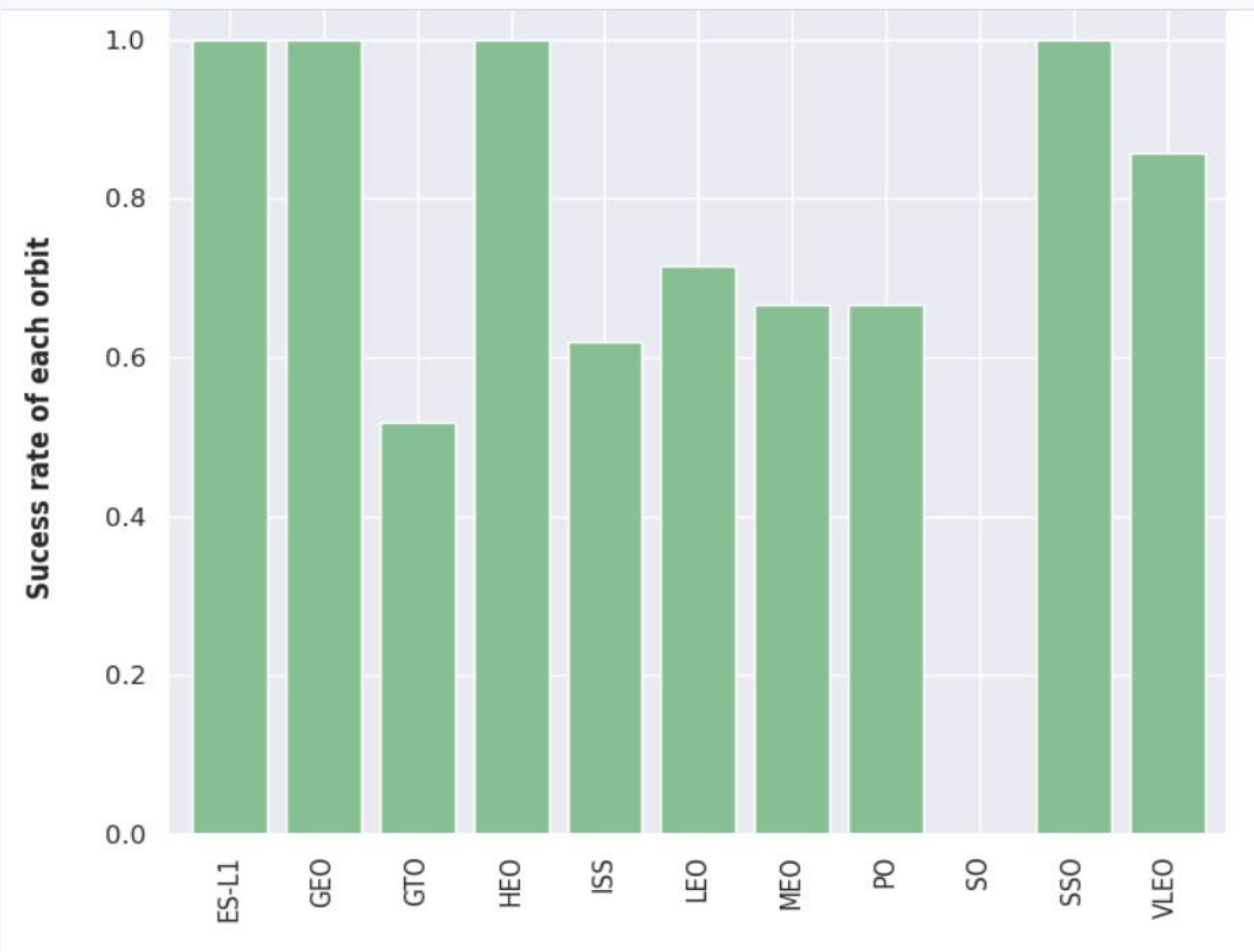
Payload vs. Launch Site

The greater the payload mass (greater than 7000 kg), higher the success rate for the Rocket. But there’s no clear pattern to take a decision, if the launch site is dependent on Payload mass for a success launch.



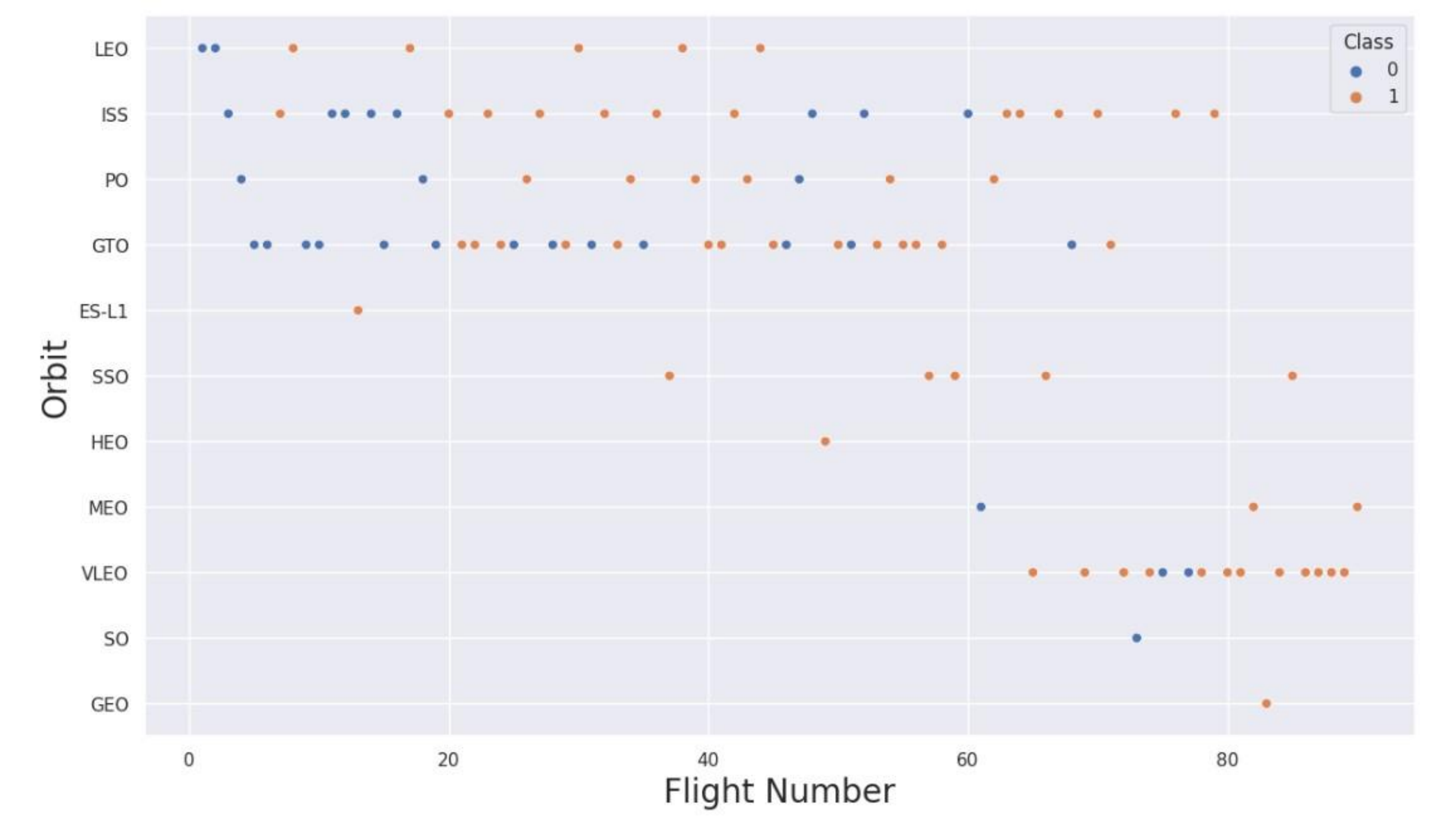
Success Rate vs. Orbit Type

ES-L1, GEO, HEO, SSO has highest Success rates



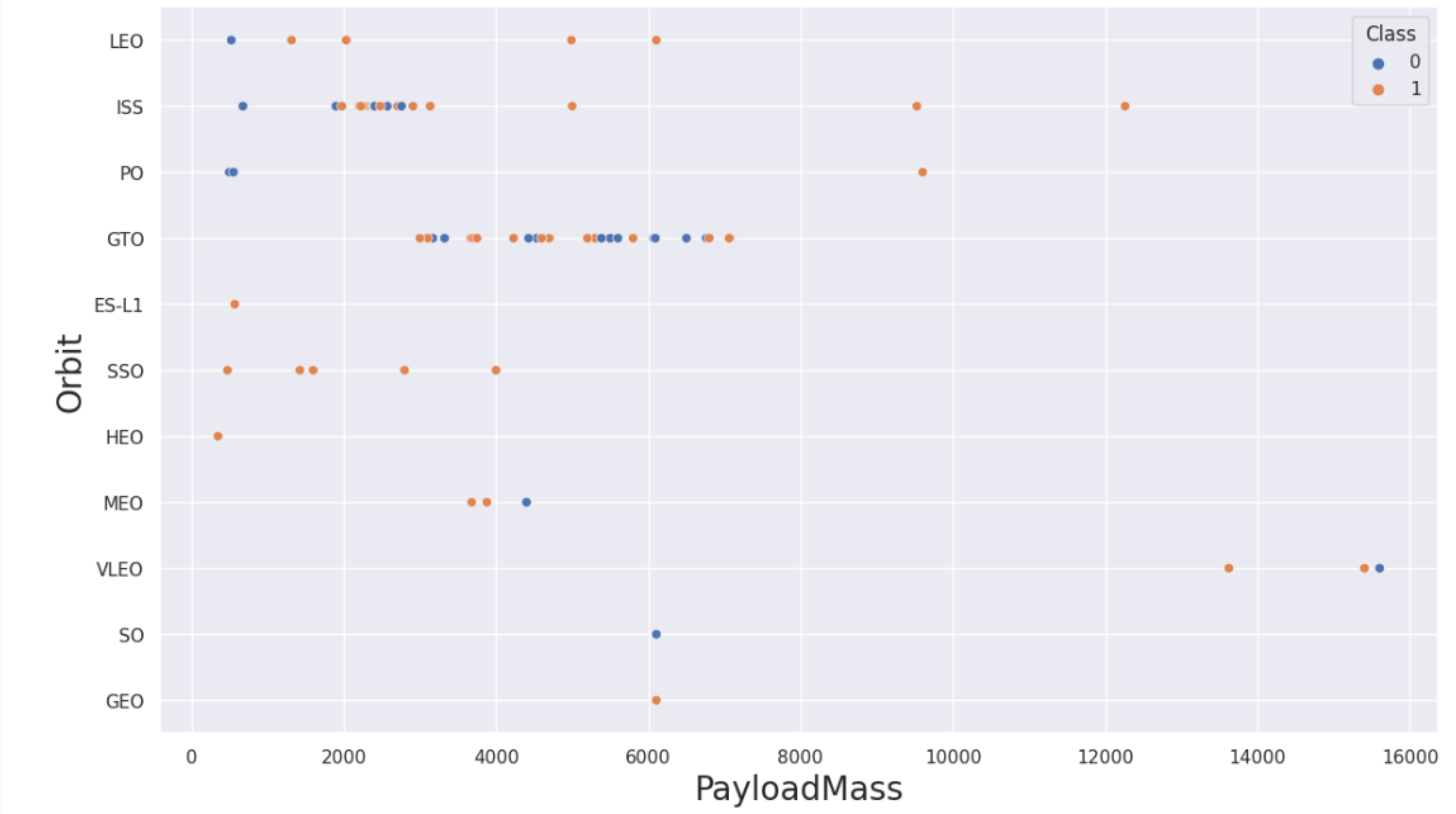
Flight Number vs. Orbit Type

- We can see that LEO orbit success increases with number of flights
- On the other hand, there seems to be no relationship between flight number and GTO orbit



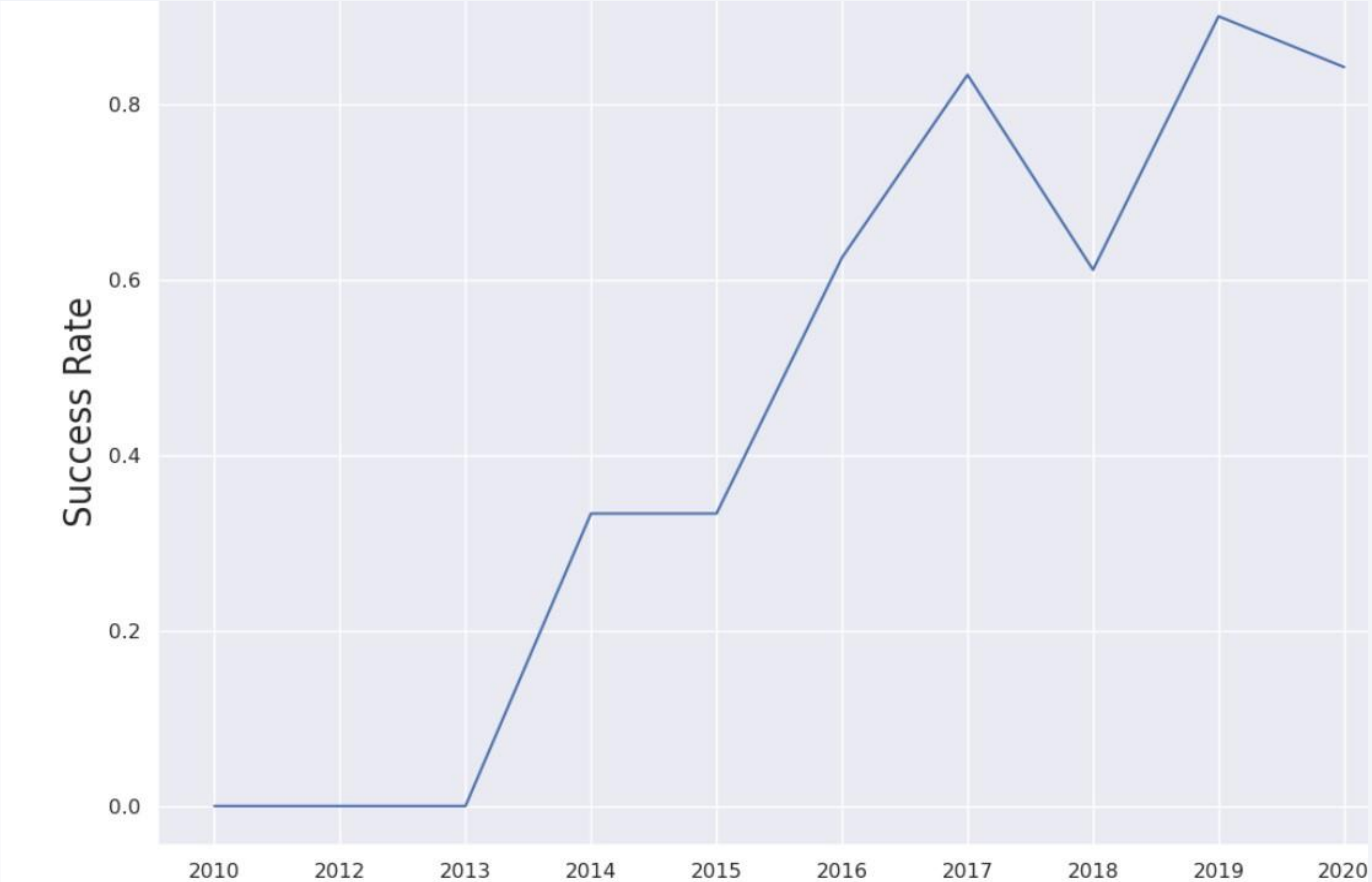
Payload vs. Orbit Type

- We can observe that heavy payloads have a negative influence on MEO, GTO, VLEO orbits
- Positive on LEO, ISS orbits



Launch Success Yearly Trend

We can observe that the success rate since 2013 kept increasing relatively though there is a slight dip after 2019



All Launch Site Names

SQL Query:

```
%sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEX;
```

DESCRIPTION:

Using the word DISTINCT in the query will give unique values for Launch_Site column from table SPACEX

Launch Site Names Begin with 'CCA'

SQL Query:

```
%sql SELECT * FROM SPACEX WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
```

DESCRIPTION:

Using keyword 'LIMIT 5' in the query will give 5 records from table SPACEX and with condition LIKE keyword with wild card - 'CCA%. The percentage in the end suggests that the Launch Site name must start with CCA.

Total Payload Mass

SQL Query:

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS "Total Payload Mass by NASA (CRS)" FROM SPACEX
WHERE CUSTOMER = 'NASA (CRS)';
```

DESCRIPTION:

Using the function SUM calculates the toatl in the column PAYLOAD_MASS_KG_ and WHERE clause filters data and gives Customer NASA(CRS) data

Average Payload Mass by F9 v1.1

SQL Query:

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS "Average Payload Mass by Booster Version F9 v1.1"
FROM SPACEX \ WHERE BOOSTER_VERSION = 'F9 v1.1';
```

DESCRIPTION:

Using the function AVG gives average of the column PAYLOAD_MASS_KG_

WHERE filters the dataset to only perform calculations on Booster version F9 v1.1

First Successful Ground Landing Date

SQL Query:

```
%sql SELECT MIN(DATE) AS "First Succesful Landing Outcome in Ground Pad" FROM SPACEX \
WHERE LANDING__OUTCOME = 'Success (ground pad)';
```

DESCRIPTION:

Using the function MIN gives out the minimum date in the Column ‘DATE’ and WHERE clause filters the data to perform calculations on Landing_Outcome with values “Success(ground pad)”

Successful Drone Ship Landing with Payload between 4000 and 6000

SQL Query:

```
%sql SELECT BOOSTER_VERSION FROM SPACEX WHERE LANDING__OUTCOME = 'Success (drone ship)' \
AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;
```

DESCRIPTION:

Selecting only Booster_Version,

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

AND clause specifies additional filter conditions

Payload_Mass_kg_ > 4000 AND Payload_Mass_kg _ < 6000

Total Number of Successful and Failure Mission Outcomes

SQL Query:

```
%sql SELECT COUNT(MISSION_OUTCOME) AS "Total Number of Successful and Failure Mission" FROM SPACEX
\ WHERE MISSION_OUTCOME LIKE 'Success%' OR MISSION_OUTCOME LIKE 'Failure%';
```

DESCRIPTION:

COUNT gives total number of successful and failure mission outcomes

LIKE gives all the mission outcomes which starts with either Success or Failure

Boosters Carried Maximum Payload

SQL Query:

```
%sql SELECT DISTINCT BOOSTER_VERSION AS "Booster Versions which carried the Maximum
Payload Mass" FROM SPACEX \ WHERE PAYLOAD_MASS__KG_=(SELECT
MAX(PAYLOAD_MASS__KG_) FROM SPACEX);
```

DESCRIPTION:

Using the function MAX gives the maximum payload mass in the column PAYLOAD_MASS_KG_ in sub query

WHERE clause filters Booster Version which had that maximum payload

2015 Launch Records

SQL Query:

```
%sql SELECT {fn MONTHNAME(DATE)} as "Month", BOOSTER_VERSION, LAUNCH_SITE FROM SPACEX
WHERE year(DATE) = '2015' AND \ LANDING__OUTCOME = 'Failure (drone ship)';
```

DESCRIPTION:

We need to 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.

Via year function we extract the year and where clause 'Failure (drone ship)' gives required values.

Also, using (fn MONTHNAME(DATE)) gives the Month name

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

SQL Query:

```
%sql SELECT LANDING_OUTCOME as “Landing Outcome”, COUNT(LANDING_OUTCOME) AS “Total Count” FROM SPACEX \
WHERE DATE BETWEEN ‘2010-06-04’ AND ‘2017-03-20’ \
GROUP BY LANDING_OUTCOME \
ORDER BY COUNT(LANDING_OUTCOME) DESC;
```

DESCRIPTION:

Selecting only LANDING_OUTCOME, WHERE clause filters data with DATE BETWEEN ‘2010-06-04’ AND ‘2017-03-20’

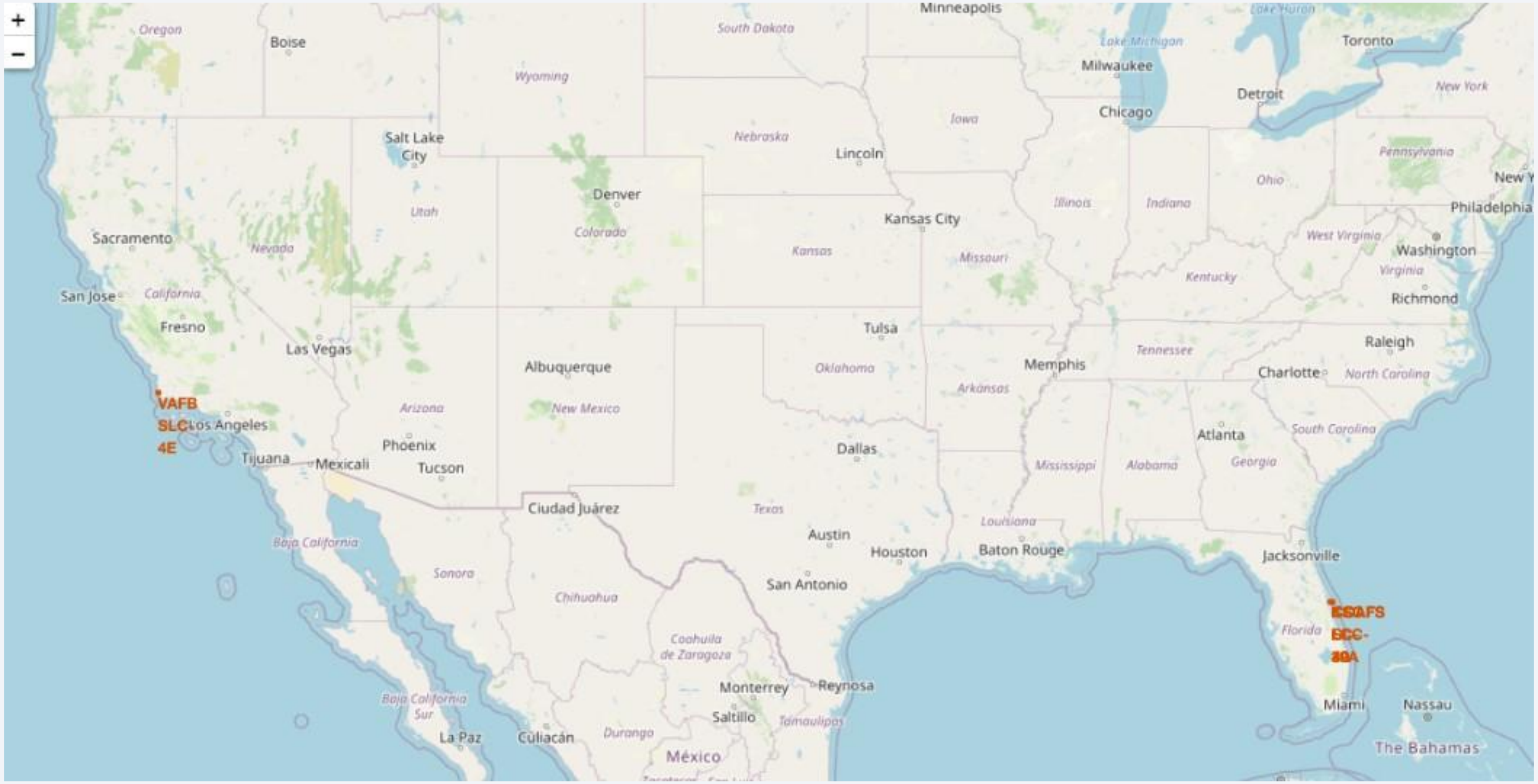
Grouping by LANDING_OUTCOME, Order by COUNT(LANDING_OUTCOME) in descending order

Section 3

Launch Sites Proximities Analysis

All Launch Sites on Folium Map

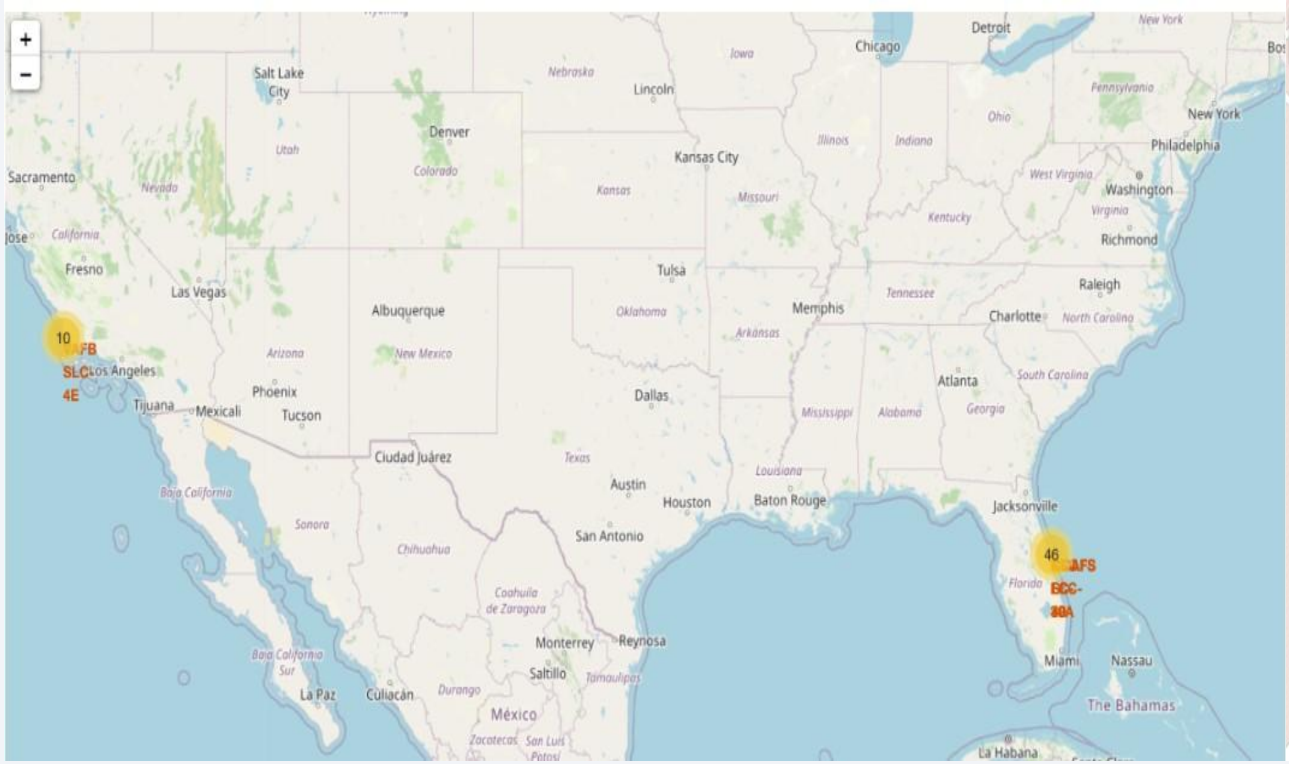
SpaceX launch sites are near to the United States of America coasts i.e., Florida and California Regions

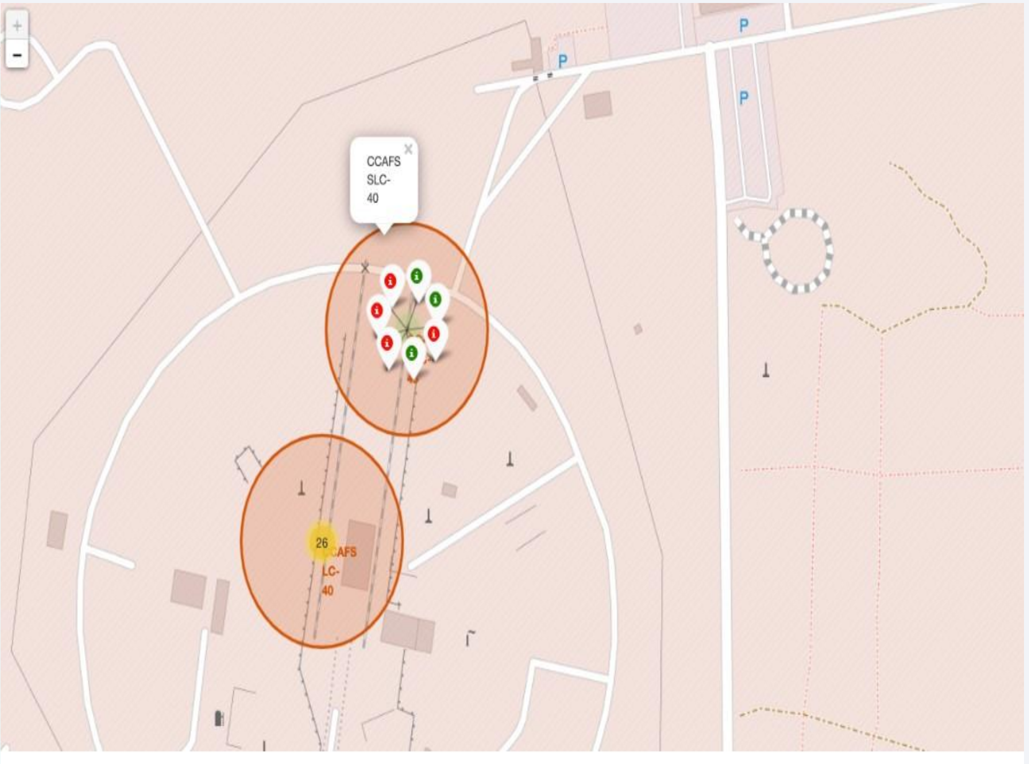


Color Labeled Launch Records

Green marker shows successful launches and Red marker shows failures

KSC LC-39A has the maximum probability of success

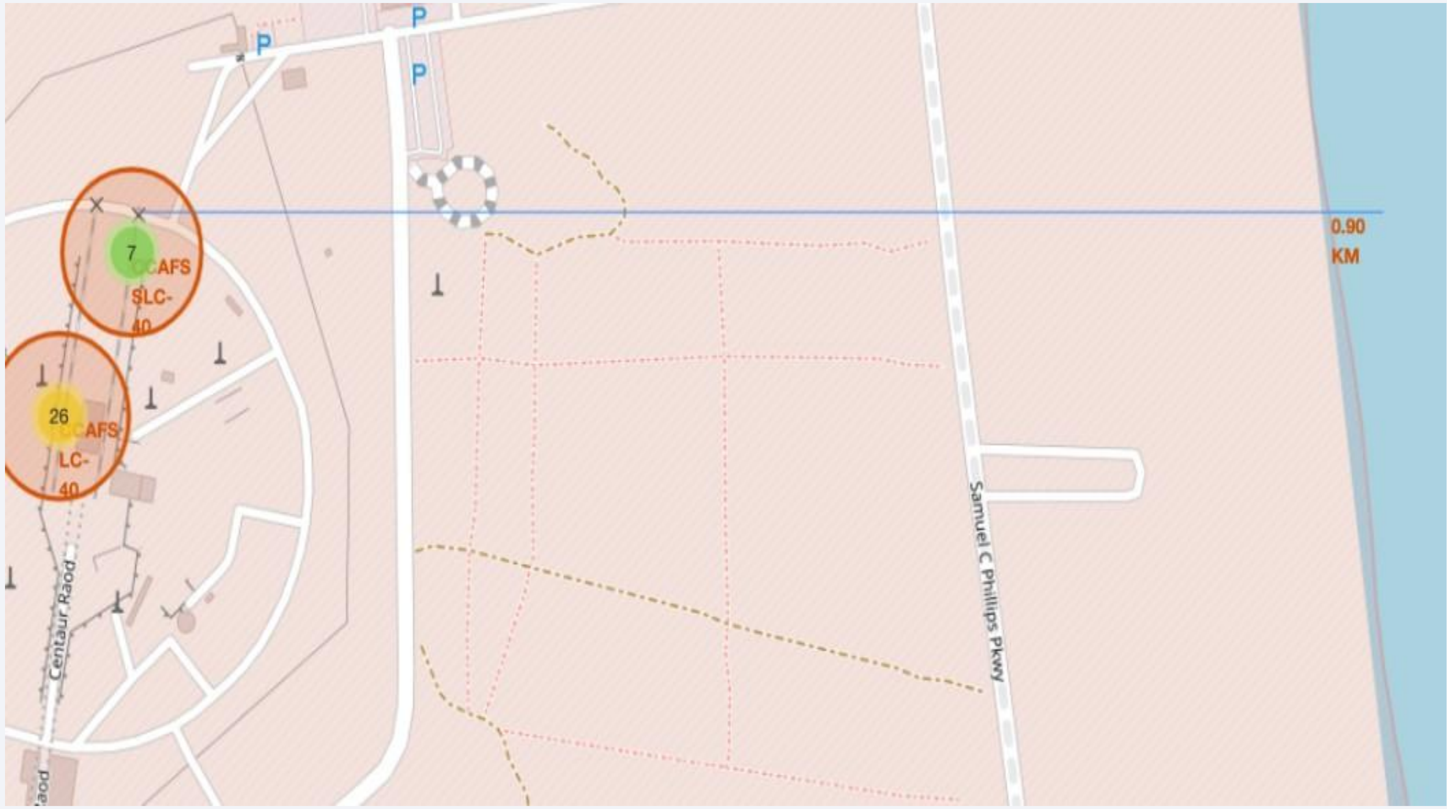




Launch Site Distances from Equator & Railways

Distance from Equator is greater than 3000 km for all sites

Distance for all launch sites from railway tracks are greater than 0.7 km. So they are not so far from railway tracks



Section 5

Predictive Analysis (Classification)

Classification Accuracy

- algorithms = {'KNN':knn_cv.best_score_,'Decision Tree':tree_cv.best_score_,'Logistic Regression':logreg_cv.best_score_,'SVM':svm_cv.best_score_} best_algorithm = max(algorithms, key= **lambda** x: algorithms[x])

  print('The method which performs best is \"',best_algorithm,'\" with a score of',algorithms[best_algorithm])

- The method which performs best is " Decision Tree " with a score of 0.9017857142857144

Confusion Matrix

Accuracy: (TP+TN)/Total

Misclassification Rate: (FP+FN)/Total

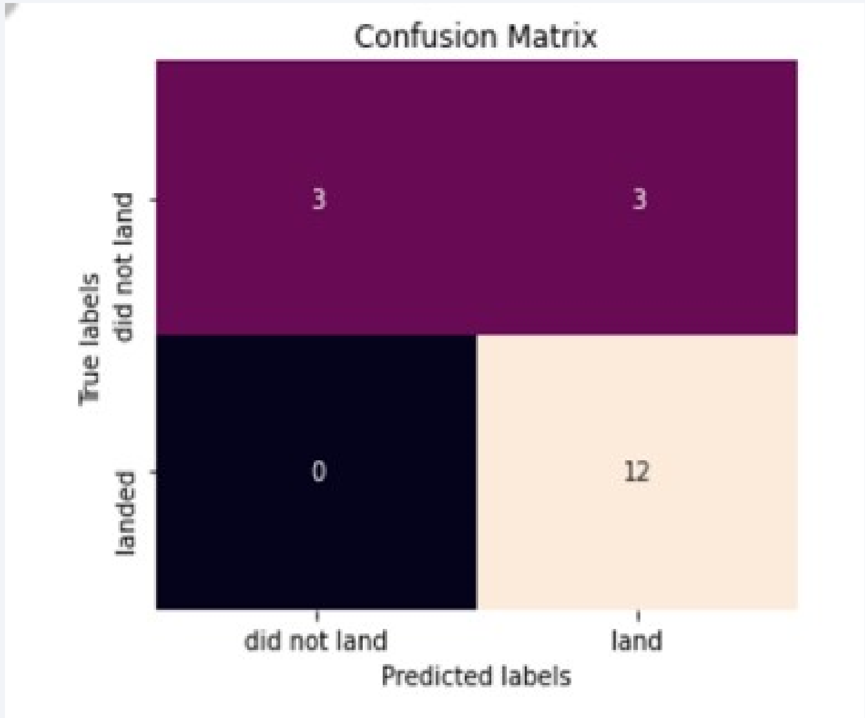
True Positive Rate: TP/Actual Yes

False Positive Rate: FP/Actual No

True Negative Rate: TN/Actual No

Precision: TP/Predicted Yes

Prevalence: Actual yes/Total



Confusion Matrix for best model – Decision Tree

Conclusions

- Orbits ES-L1, GEO, HEO, SSO has highest Success rates
- Success rates for SpaceX launches has been increasing relatively with time and it looks like soon they will reach the required target
- KSC LC-39A had the most successful launches but inceasing payload mass seems to have negative impact on success
- Decision Tree Classifier Algorithm is the best for Machine Learning Model for provided dataset

Thank you!