IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Vineetha Koduri
7 July 2022

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

- SpaceX is a private spaceflight company which has gained worldwide attention for a series of historic milestones by offering a rocket launches specifically Falcon 9 as low as 62 million dollars.
- It is the only company ever to make much of the savings because it can reuse the first stage.

Goal of this project:

To create the machine learning pipeline to predict the landing outcome of the first stage in the future

➢ Problems you want to find answers

- Identify the factors that influence the landing outcome
- Relationship between each variables i.e how it is affecting the outcome
- 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 scrapping 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

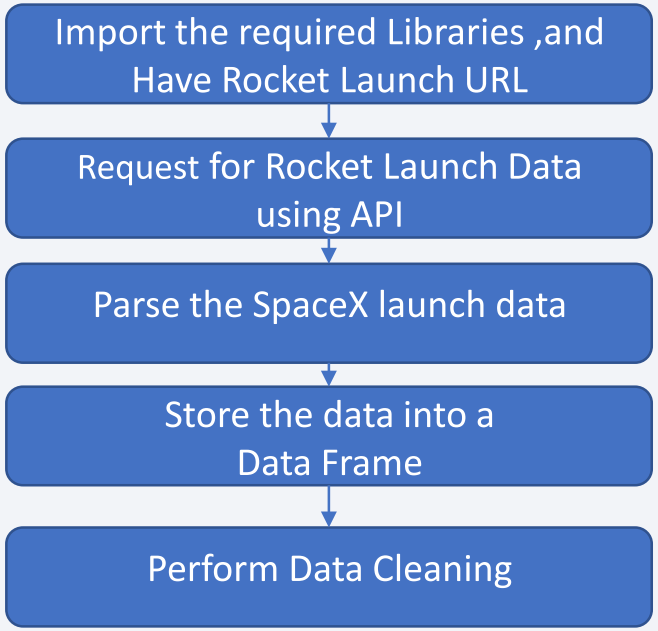
- Data collection can be done using REST APIs, web scrapping or web crawling etc.
- For REST API's, its started by using the get request, then decode the response content as json format and then convert into Data frames. Then we can easily clean the data.
- For Web Scrapping, we use Beautiful Soup to extract the launch record in HTML tables, then parse these tables into Data Frames for further analysis.

- SpaceX API
- Scraping Wikipedia

The process of gathering accurate data from a variety of relevant sources to find answers to research problems

Data Collection – SpaceX API

- Flow Chart



GitHub URL:

- https://github.com/vineetha727/Data-Science-Coursera/blob/main/course-10/jupyter-labs-spacex-data-collection-api%20(2).ipynb

```python
# Requests allows us to make HTTP requests which we will use to get data from an API
import requests
# Pandas is a software library written for the Python programming language for data manipulation and analysis.
import pandas as pd
# NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays
import numpy as np
# Datetime is a library that allows us to represent dates
import datetime
```

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

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

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

```python
# 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 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 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

- Flow Chart

Import the required Libraries

↓

Request the Falcon9 Launch Wiki page from url

↓

Create a Soup object using Beautiful Soup from HTML response

↓

Extract Data from HTML header

↓

Store the data into a Data Frame

```
import sys

import requests
from bs4 import BeautifulSoup
import re
import unicodedata
import pandas as pd
```

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

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

```
]:  '<!DOCTYPE html>\n<html class="client-nojs" lang="en" dir="ltr">\n<head>\n<meta charset="UTF-8"/>\n<title>List of Falcon 9 and Falcon Heavy launch
```

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(data, 'html5lib')
```

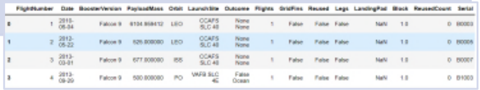
```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if 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 dictonary
        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)
```

- GitHub URL
  - https://github.com/vineetha727/Data-Science-Coursera/blob/main/course-10/jupyter-labs-webscraping%20(1).ipynb

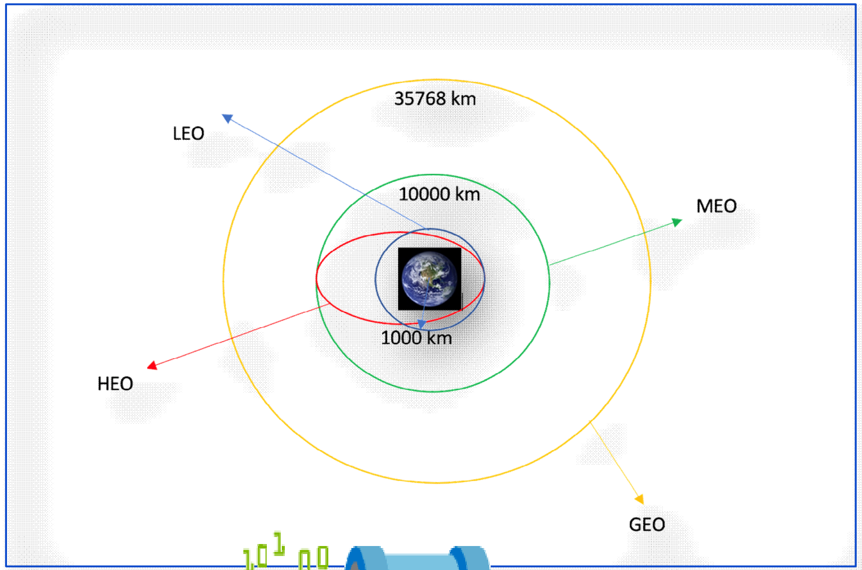
DATA Wrangling

Definition	DATA Wrangling is the process of cleaning and unifying messy and complex data sets for easy access and can perform Exploratory Data Analysis (EDA) and Determine Training Labels further.
Process	We will start calculating 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.
GitHub URL	https://github.com/vineetha727/Data-Science-Coursera/blob/main/course-10/labs-jupyter-spacex-Data%20wrangling%20(1).ipynb



	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003

- Exploratory Data Analysis (EDA)
  - Determine Training Labels



matplotlib and seaborn

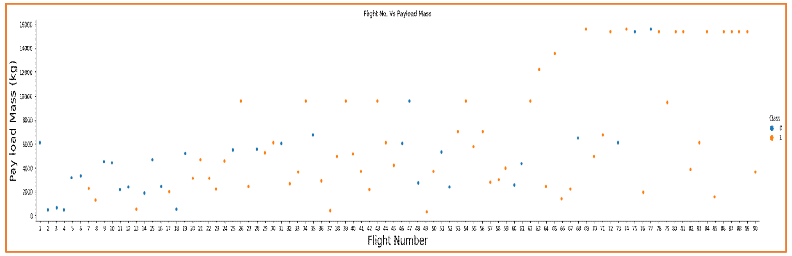
EDA with Data Visualization

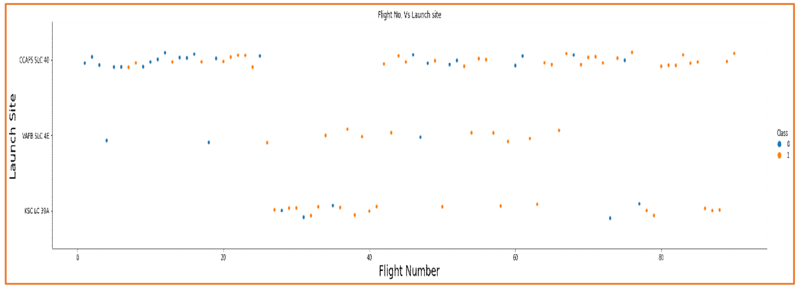
- We first started by using scatter graph to find the relationship between the attributes using Matplotlib and Seaborn. The graphs are [lotted for following attributes such as
  - Payload and Flight Number.
  - Flight Number and Launch Site.
  - Payload and Launch Site.
  - Flight Number and Orbit Type.
  - Payload and Orbit Type.

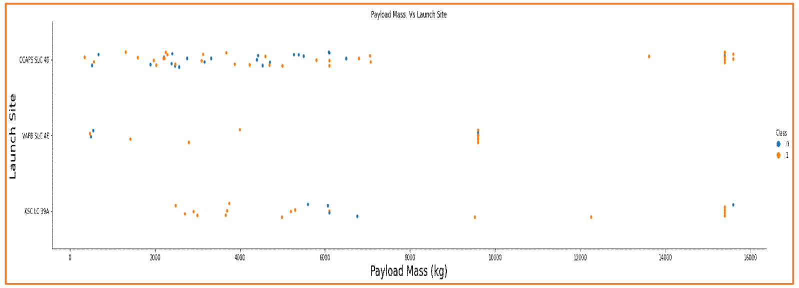
- Scatter plots show dependency of attributes on each other. Once a pattern is determined from the graphs. It's very easy to visualize which factors affecting the most to the success of the landing outcomes.

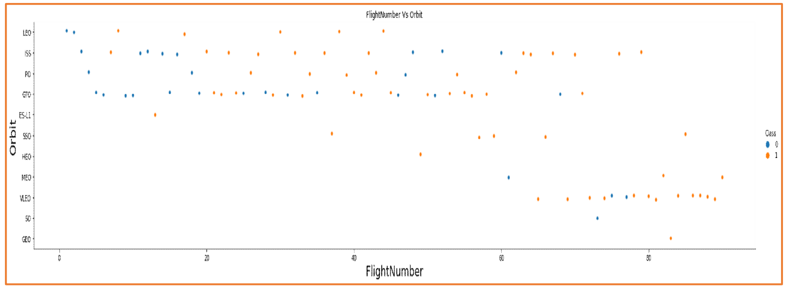
- **Github URL:**
- **https://github.com/vineetha727/Data-Science-Coursera/blob/main/course-10/jupyter-labs-eda-dataviz%20(1).ipynb**

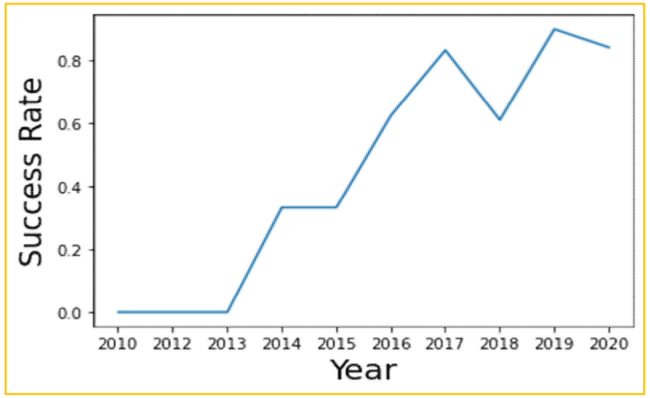
Graphs

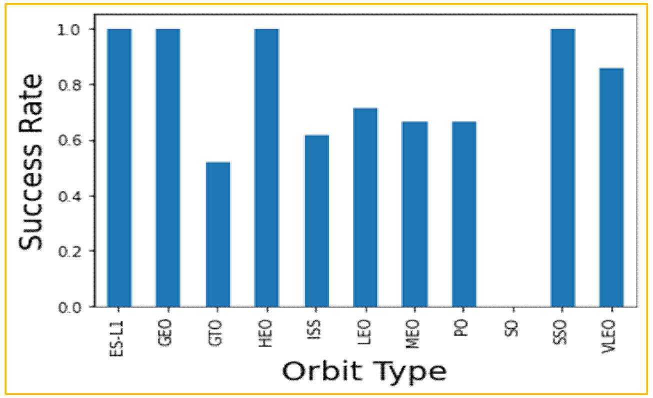


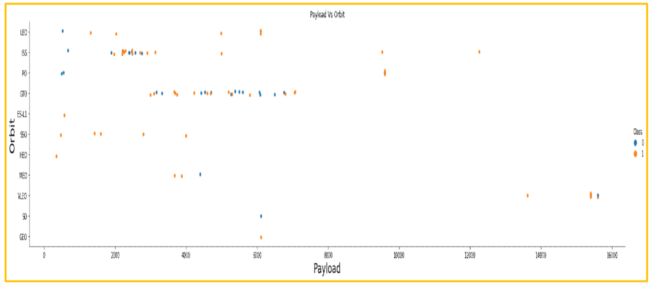












EDA with SQL

- By using SQL, we have performed the following queries to get better understating on the dataset
  - Display the names of the unique launch sites in the space mission
  - Display 5 records where launch sites begin with the string 'CCA'
  - Display the total payload mass carried by boosters launched by NASA (CRS)
  - Display average payload mass carried by booster version F9 v1.1
  - List the date when the first successful landing outcome in ground pad was achieved.
  - List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
  - List the total number of successful and failure mission outcomes
  - List the names of the booster_versions which have carried the maximum payload mass. Use a subquery
  - List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
  - Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

Github URL:

https://github.com/vineetha727/Data-Science-Coursera/blob/main/course-10/jupyter-labs-eda-sql-coursera%20(2).ipynb

Build an Interactive Map with Folium

We visualize the launch data into an interactive visual analytics using Folium. We took the latitude and longitude coordinates at each launch site and add a circle marker around each launch site with a label of the name of the launch site.

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

Use the Haversine's formula to calculated the distance of the launch sites to various landmark to find answer to the questions of:

1. How close the launch sites with railways, highways and coastlines?
2. How close the launch sites with nearby cities?

GitHub URL: https://github.com/vineetha727/Data-Science-Coursera/blob/main/course-10/lab_jupyter_launch_site_location%20(1)%20(2).ipynb

Build a Dashboard with Plotly Dash

- We built an interactive dashboard using Plotly Dash which allows the user to choose the data what they need.
- We have plotted Pie Charts showing the Total Launches by a Certain Sites and Scatter Plots to show the relationship with Outcome and Payload Mass for different booster version.

GitHubURL:

https://github.com/vineetha727/Data-Science-Coursera/blob/main/course-10/spacex_dash_app.py

Predictive Analysis (Classification)

Building the Model:

* Load the Data Set using Pandas
* Transform the Data
* Split the Data into Training and Testing Datasets
* Choose the ML Algorithms (classification)
* Set the parameters like GridSearchCV .
* Fit the dataset to train the Model
* Test the model with new values

Evaluation the Model:

* Check the Accuracy of the defined model
* Plot the Confusion Matrix
* Test the model with New data

Improving the Model

* Use Feature Engineering techniques and Algorithm Tuning

Finding the Best Model

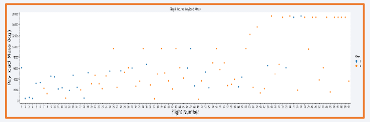
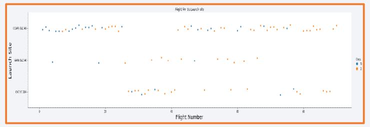
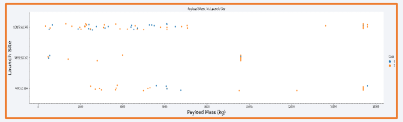
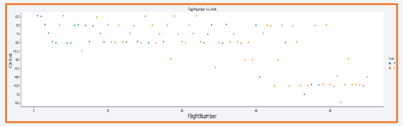
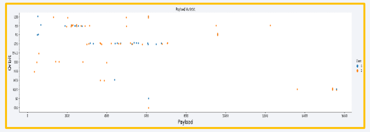
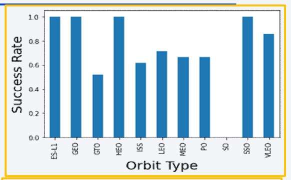
* The model with good accuracy score and precision from confusion matrix will be the best performing model.

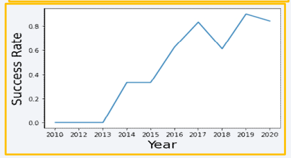
Github URL:

https://github.com/vineetha727/Data-Science-Coursera/blob/main/course-10/SpaceX_Machine%20Learning%20Prediction_Part_5%20(2).ipynb

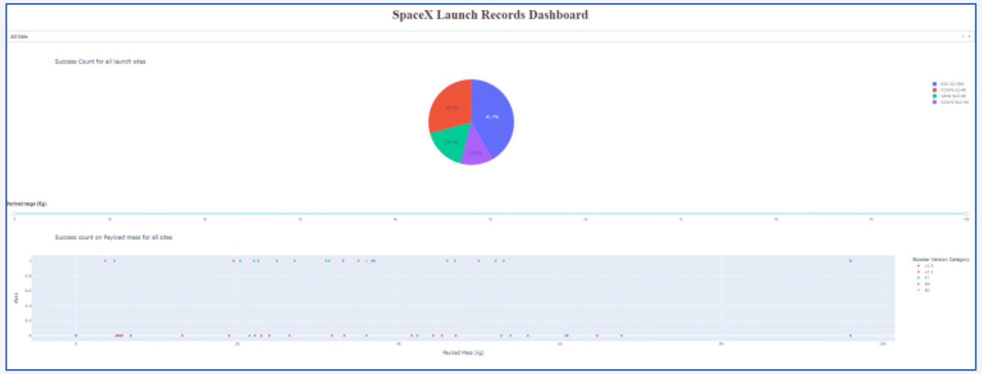
Results

- Exploratory data analysis results



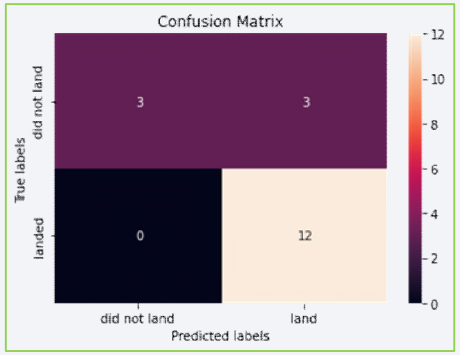
- Interactive analytics demo in screenshots



- Predictive analysis results

```
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 0.8892857142857145
Best Params is : {'criterion': 'entropy', 'max_depth': 4, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 10, 'splitter': 'rando
m'}
```

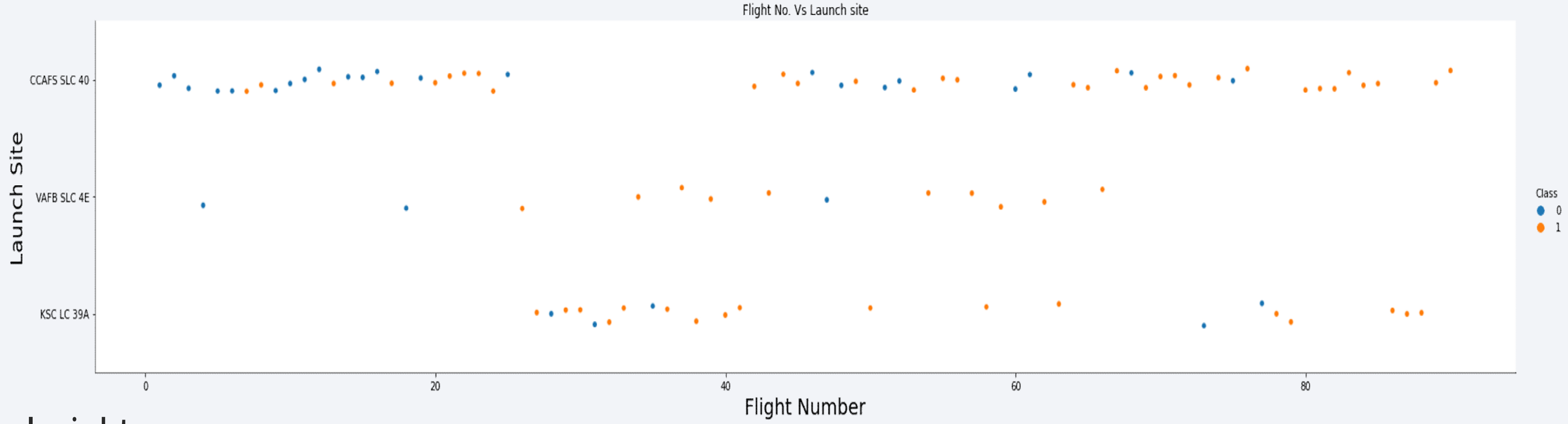


Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

- Scatter plot of Flight Number vs. Launch Site



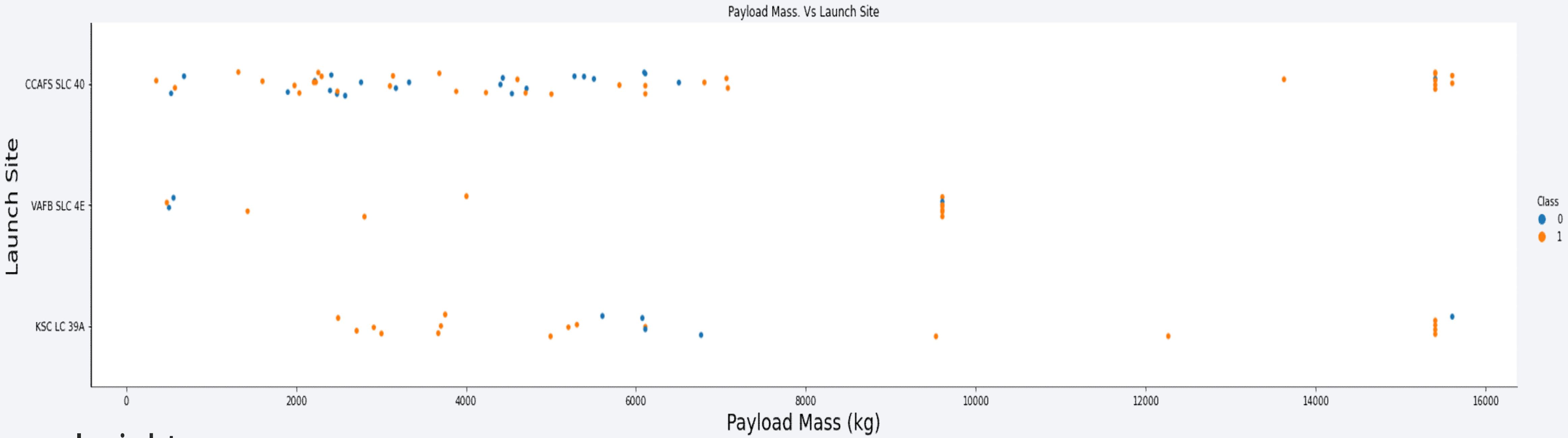
- Insights:

The above plot will show that the if amount of launch site is larger then the success rate will be larger.

Ex: CCAFF SLC40

Payload vs. Launch Site

- Show a scatter plot of Payload vs. Launch Site

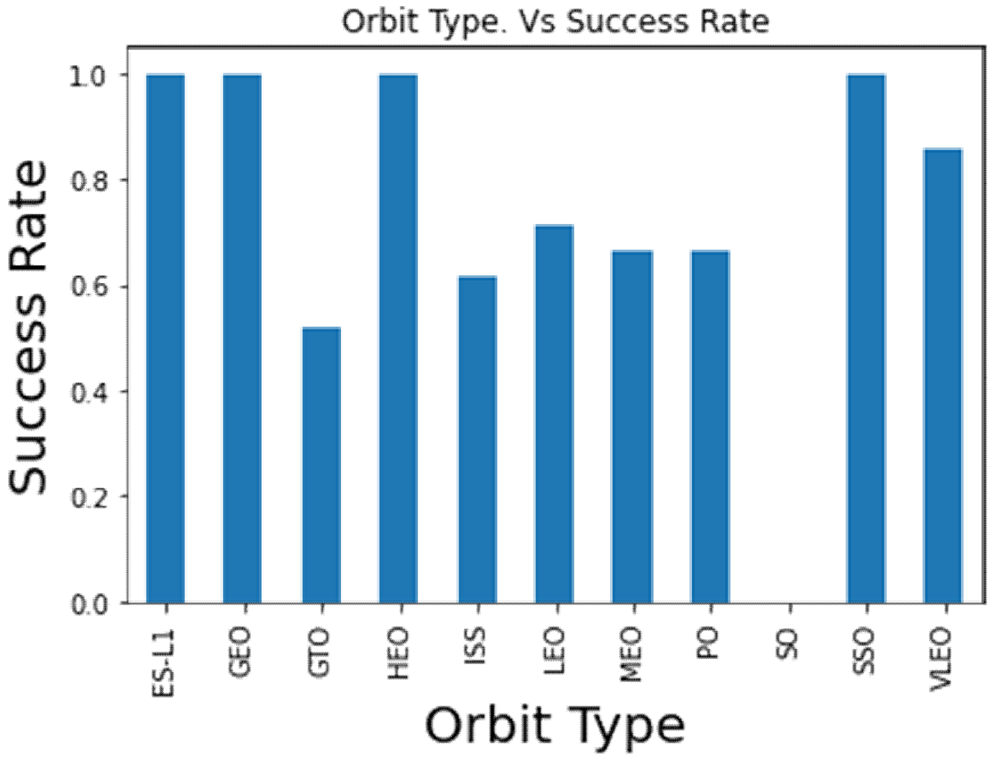


Insights:

- As the Payload mass is greater than 7000Kg, the probability of success is increased.

VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

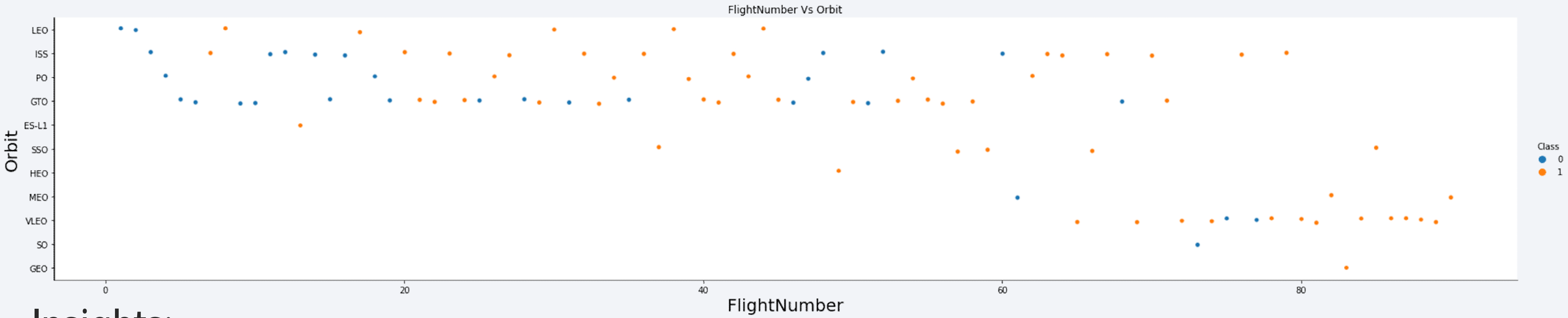
Success Rate vs. Orbit Type



- It shows that few orbits are having 100% success rates like ES-L1, GEO, SSO,HEO . But, SO is having 0% success rate.
- Some orbits like ES-L1, GEO, SSO,HEO has only 1 occurrence which mean this data need more dataset to see beyond the pattern or trend before .So that we draw some conclusion.

Flight Number vs. Orbit Type

- Scatter point of Flight number vs. Orbit type

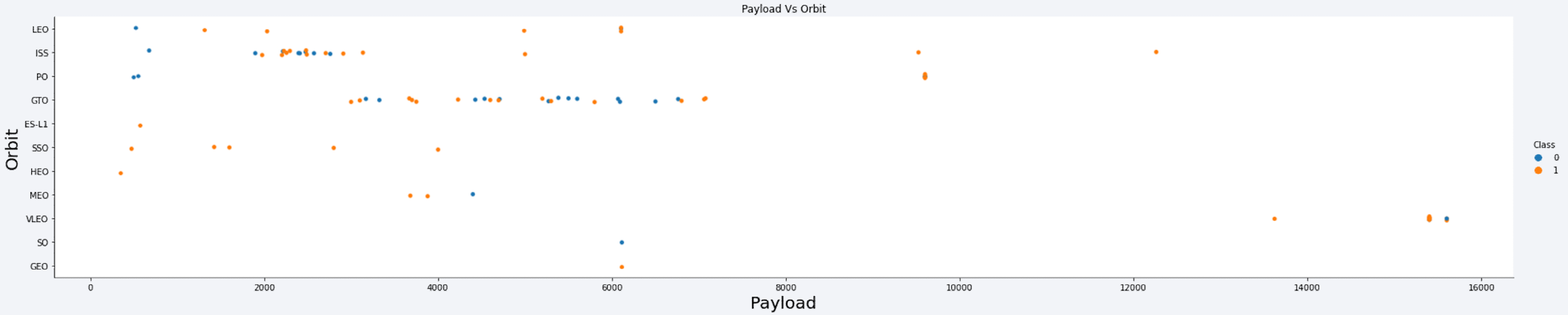


Insights:

- If we look at LEO orbit, Larger the flight number, the success rate is High. This is applicable for all orbits except GTO orbit, as it has no relationship with both attributes.

Payload vs. Orbit Type

- Scatter point of payload vs. orbit type

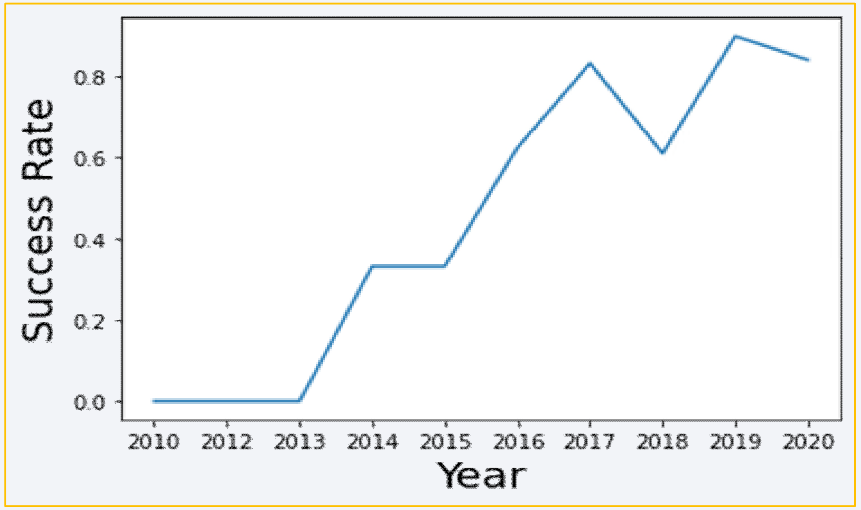


- Insights:

- With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.
- GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.

Launch Success Yearly Trend

- Show a line chart of yearly average success rate



- Insights:

- The success rate since 2013 kept increasing till 2020. Only in 2018 it is some what decreased slightly.

All Launch Site Names

- Find the names of the unique launch sites

We have to use DISTINCT keyword in the SQL query to see the unique launch sites in the SpaceX data.

Display the names of the unique launch sites in the space mission

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

```
 * ibm_db_sa://mtm08062:***@2d46b6b4-cbf6-40eb-bbce-6251e6ba0300.bs2io90l08kqb1o
d8lcg.databases.appdomain.cloud:32328/bludb
Done.
```

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

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`

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

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

```
 * ibm_db_sa://mtm08062:***@2d46b6b4-cbf6-40eb-bbce-6251e6ba0300.bs2io90l08kqb1o
d8lcg.databases.appdomain.cloud:32328/bludb
Done.
```

launch_site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

Total Payload Mass

- Calculate the total payload carried by boosters from NASA

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

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS "Total Payload Mass by NASA (CRS)
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3
sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

Total Payload Mass by NASA (CRS)
45596

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1
- The average Payload is 2928

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

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

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.clogj3
sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

Average Payload Mass by Booster Version F9 v1.1
2928

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad

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

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.clogj3
sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

First Succesful Landing Outcome in Ground Pad
2015-12-22

- The first successful landing outcome data is on 22-12-2015.

Successful Drone Ship Landing with Payload between 4000 and 6000

- List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

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

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.clogj3sd0tgtu0lqde00.datab
ases.appdomain.cloud:32731/bludb
Done.
```

booster_version
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes

List the total number of successful and failure mission outcomes

```
%sql SELECT COUNT(MISSION_OUTCOME) AS "Successful Mission" FROM SPACEX WHERE MISSION_OUTCOME LIKE 'Success%';
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.clogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

Successful Mission
100

```
%sql SELECT COUNT(MISSION_OUTCOME) AS "Failure Mission" FROM SPACEX WHERE MISSION_OUTCOME LIKE 'Failure%';
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.clogj3sd0tgtu0lqde00.databases.appdomain.clou
d:32731/bludb
Done.
```

Failure Mission
1

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass

```
%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);
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

Booster Versions which carried the Maximum Payload Mass
F9 B5 B1048.4
F9 B5 B1048.5
F9 B5 B1049.4
F9 B5 B1049.5
F9 B5 B1049.7
F9 B5 B1051.3
F9 B5 B1051.4
F9 B5 B1051.6
F9 B5 B1056.4
F9 B5 B1058.3
F9 B5 B1060.2
F9 B5 B1060.3

- We can use WHERE clause and MAX() function to determine the maximum carried payload mass.

2015 Launch Records

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

```
%sql SELECT BOOSTER_VERSION, LAUNCH_SITE FROM SPACEX WHERE DATE LIKE '2015-%' AND \
LANDING__OUTCOME = 'Failure (drone ship)';
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.
databases.appdomain.cloud:32731/bludb
Done.
```

booster_version	launch_site
F9 v1.1 B1012	CCAFS LC-40
F9 v1.1 B1015	CCAFS LC-40

- We can use the combinations of WHERE, LIKE, AND and BETWEEN keywords to filter the failed outcomes.

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

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

```
%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 ;
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.c
loud:32731/bludb
Done.
```

Landing Outcome	Total Count
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

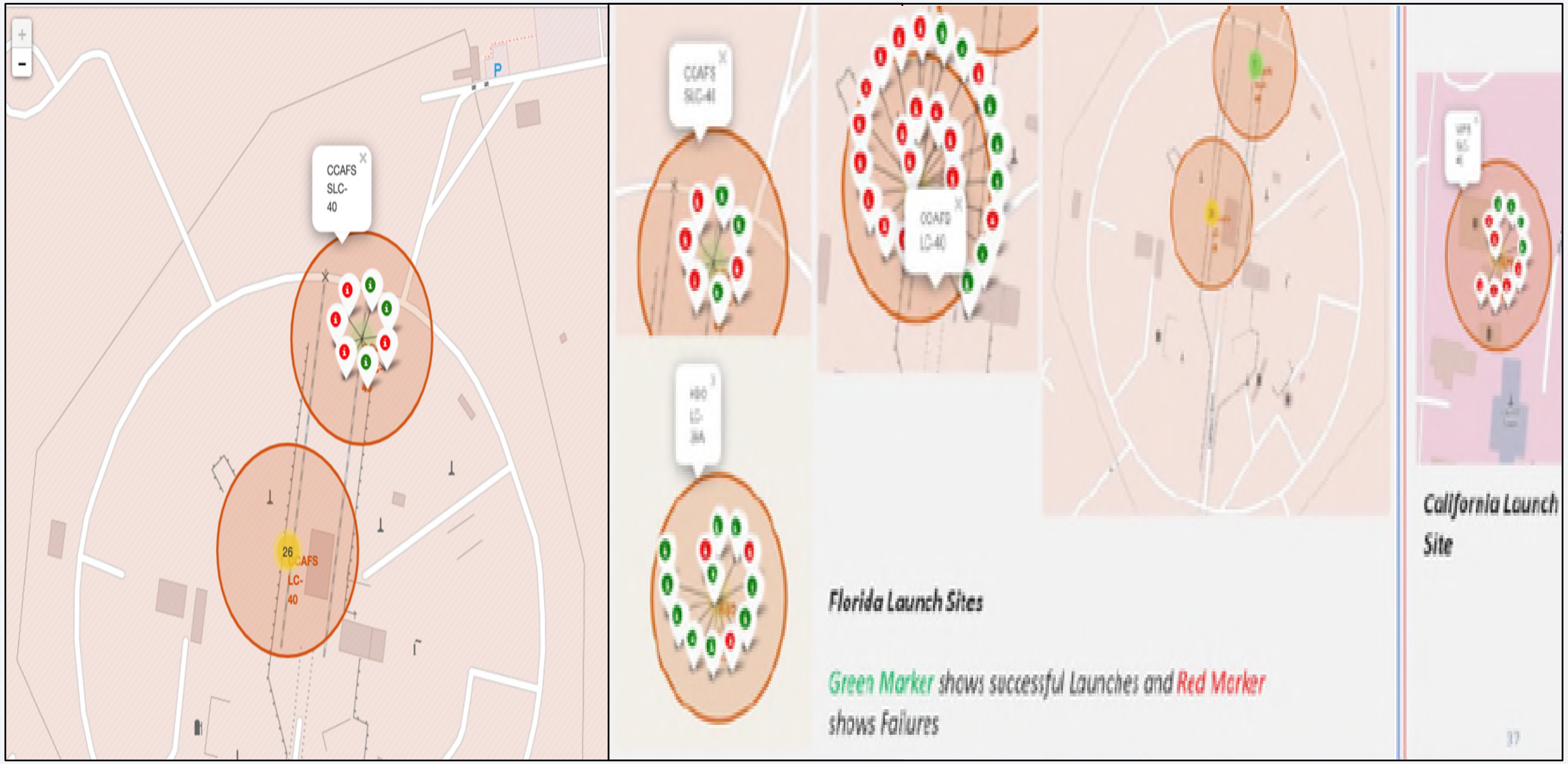
Section 3

Launch Sites Proximities Analysis

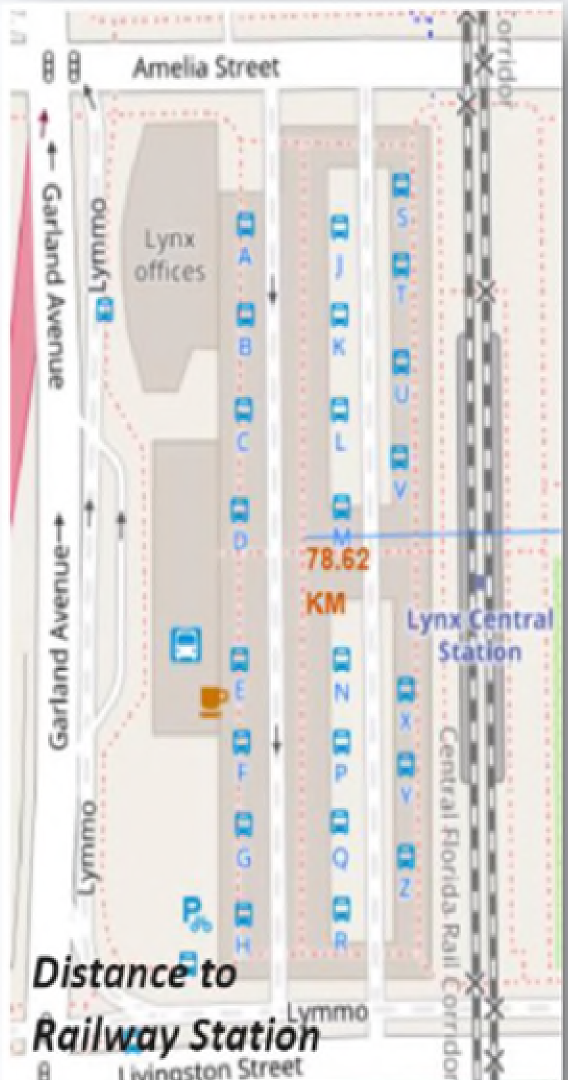
Location of all the Launch Sites

This map is about United States where all the SpaceX launch sites are located.

Launch Sites with Color Lables



Launch Sites Distance to Landmarks



•Are launch sites in close proximity to railways? No

•Are launch sites in close proximity to highways? No

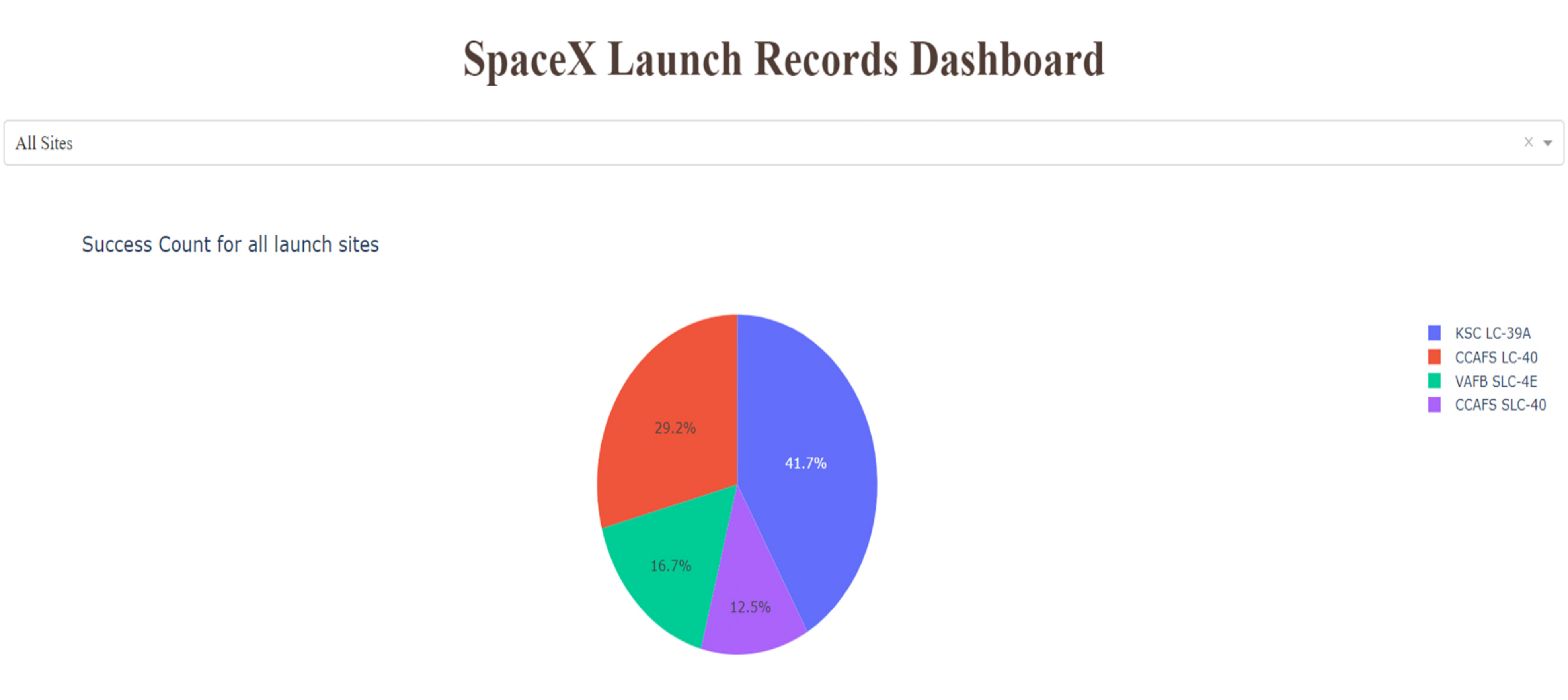
•Are launch sites in close proximity to coastline? Yes

•Do launch sites keep certain distance away from cities? Yes

Section 4

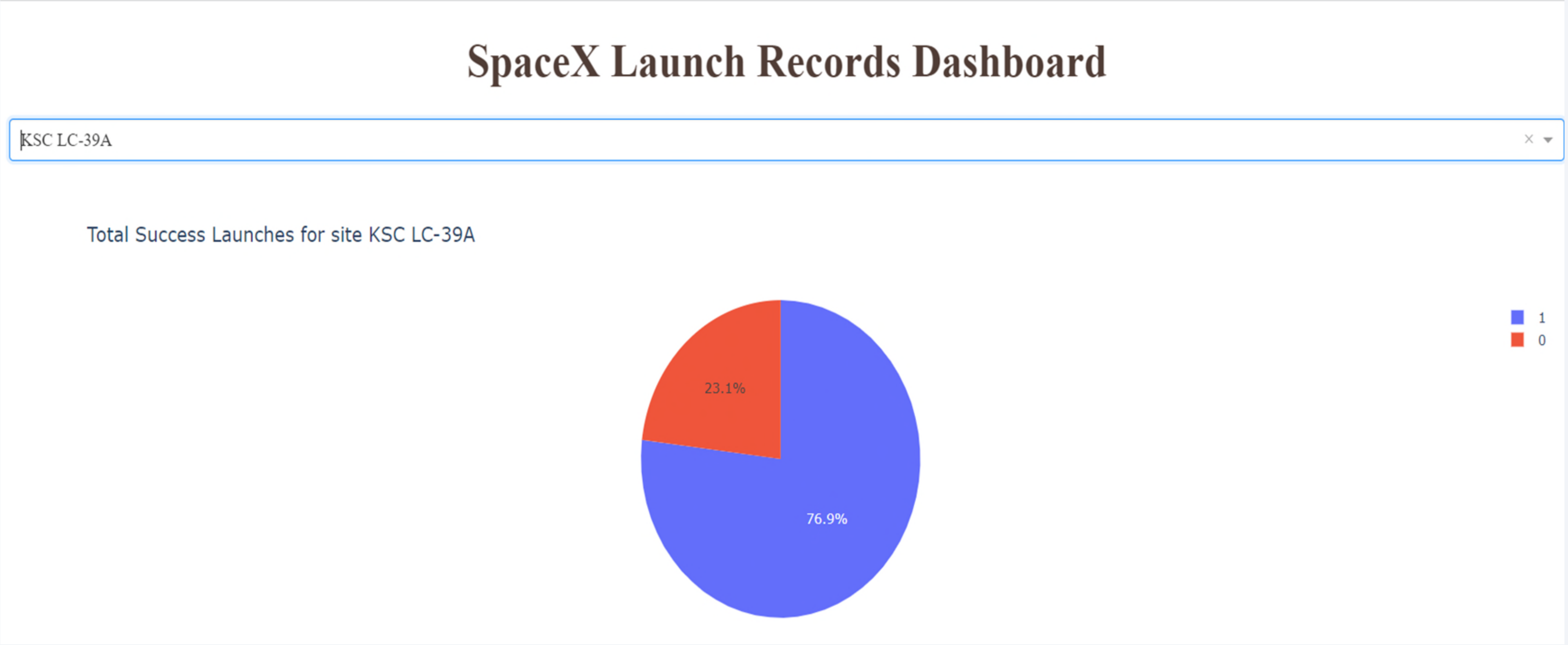
Build a Dashboard with Plotly Dash

Success Rate of All Launch Sites



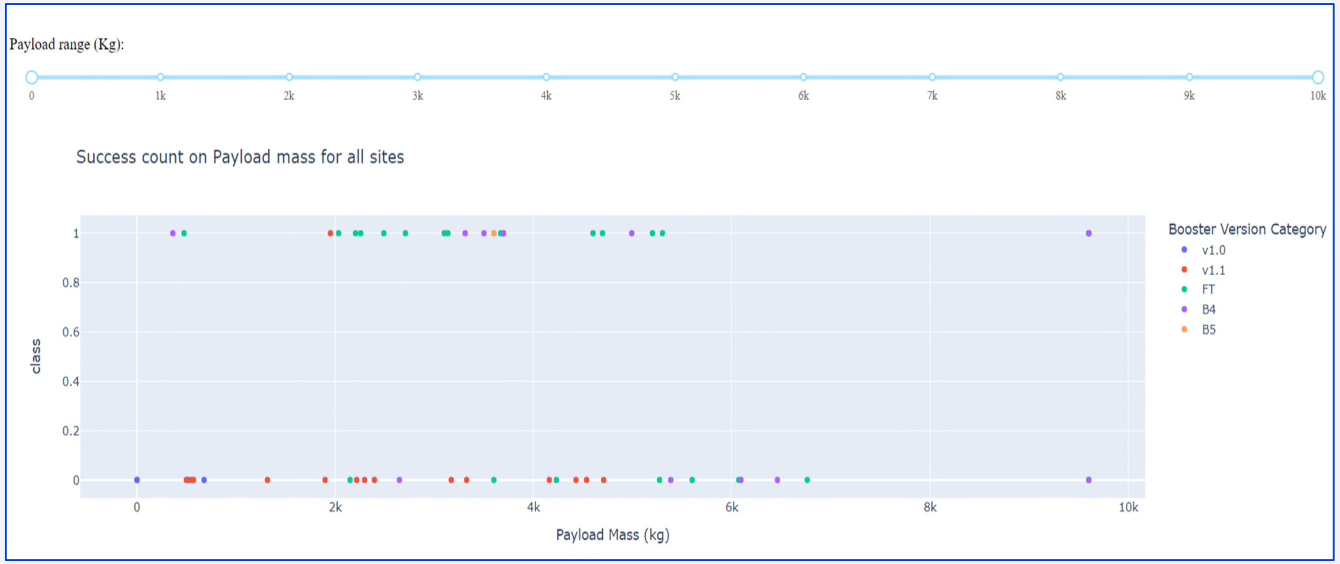
Success rate of KSC LC-39A is greater than the remaining ones.

Success ratio of KSC LC-39A



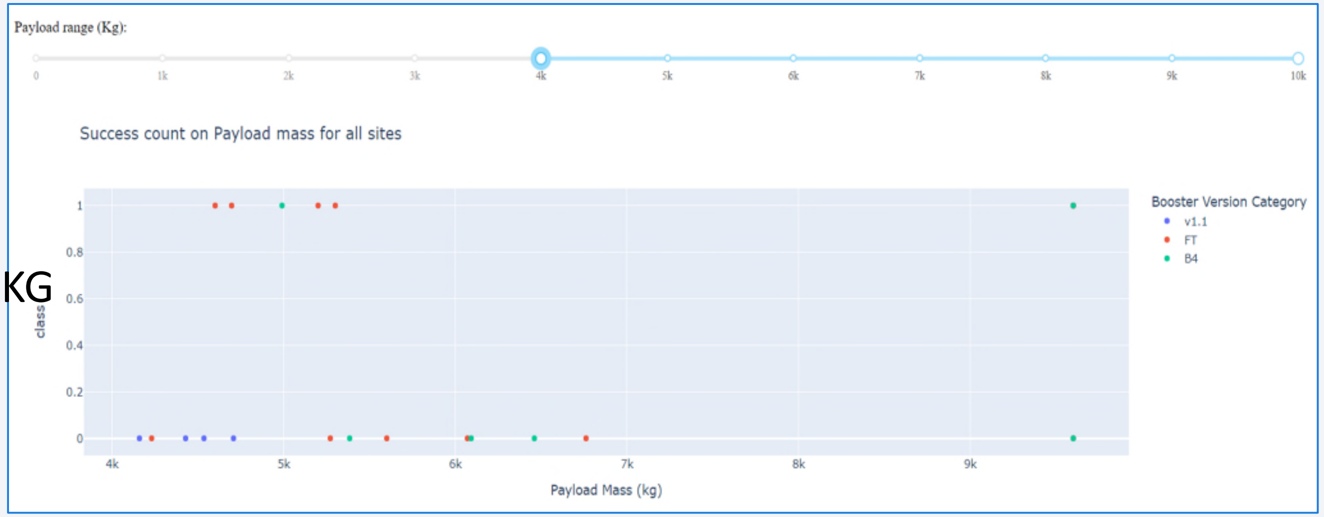
KSC LC-39A has 76.9% of success rate while 23.1% failure rate.

Payload Vs Launch Outcome



Pay load range : 0 to 10000 KG

Pay load range : 4000 to 10000 KG



Section 5

Predictive Analysis (Classification)

Classification Accuracy

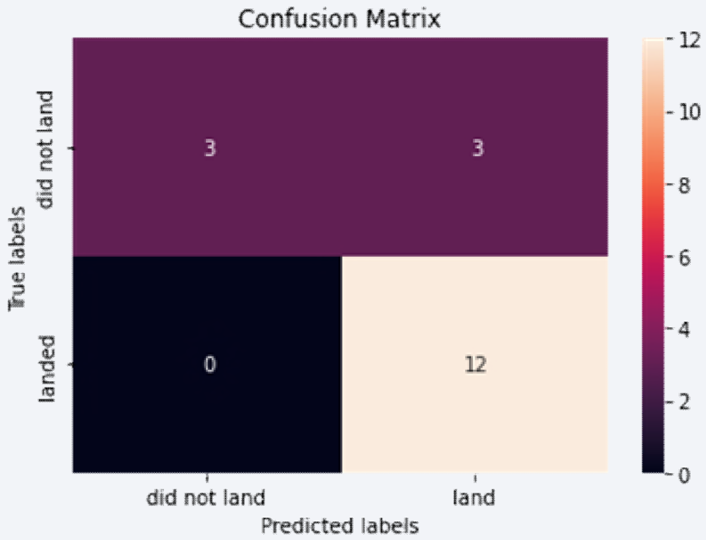
We can see that the best Algorithm is having the accuracy as 88.9% which Tree Algorithm

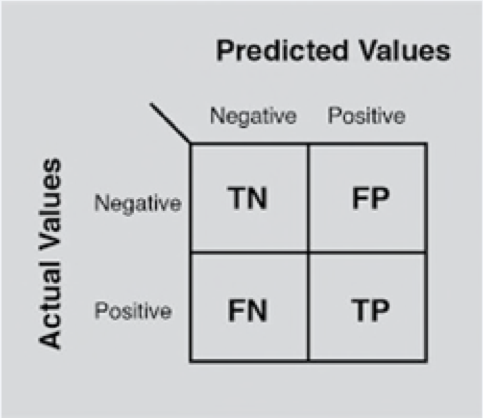
```
algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_}
bestalgorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
if bestalgorithm == 'Tree':
    print('Best Params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

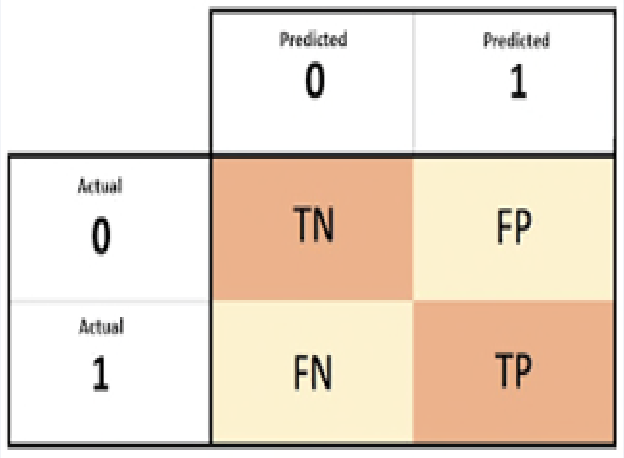
```
Best Algorithm is Tree with a score of 0.8892857142857145
Best Params is : {'criterion': 'entropy', 'max_depth': 4, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 10, 'splitter': 'rando
m'}
```

Confusion Matrix

- Below figure shows the confusion matrix of the best performing Decision Tree Classifier. Here the False Positives are 3 that is unsuccessful landing is marked as successful landing. This is the major problem here.
-







Conclusions

- We conclude that the Decision Tree Classifier is giving the best results for this dataset.
- We can observe that Starting from 2013 to 2020, the success rate in SapceX launches are increased directly, which says that is having perfect launches.
- KSCLC-39A is having the most successful launches with ~77%
- SSO orbit has most success rate.
- We can observe that Low Payload masses (below 4000kg) performed better than the Heavier ones.

Appendix

- All the code files
- https://github.com/vineetha727/Data-Science-Coursera/tree/main/course-10

Thank you!