IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Maede Miri
2023-September
https://github.com/maedemiri/coursera-Space-X

Outline

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

Executive Summary

-Summery of methodologies

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

-Summery of all results

- ✓EDA
- ✓Interactive analysis
- ✓Predictive analysis

Introduction

-Project background and context

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 due to the fact that SpaceX can reuse the first stage.

-Problems you want to find answers

In this project, we will predict if the Falcon 9 first stage will land successfully.

Section 1

Methodology

Methodology

-Executive Summary

✓ Data collection methodology:

- SpaceX Rest API
- Web Scraping from Wikipedia

✓ Perform data wrangling

- Using One Hot Encoding techniques for data fields for Machine Learning and data clening of null values and dispensable columns

✓ Perform exploratory data analysis (EDA) using visualization and SQL

✓ Perform interactive visual analytics using Folium and Plotly Dash

✓ Perform predictive analysis using classification models

- Linear Regression(LR),K Nearest Neighbor(KNN), Support Vector Machine(SVM) and Decision Tree(DT) models to determine the classifier data and evaluation.

Data Collection

-Describe how data sets were collected.

Collection SpaceX launch data from the SpaceX REST API, this API give us data about launches and information about rocket used, payload, launch speciffications and landing speciffications and landing outcome. API URL: api.spacex.com/v4/

Also, another way for obtaining Falcon 9 launch data is web scraping Wikipedia and using Beautiful Soup.

Data Collection - SpaceX API

1

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

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

2

Task 1: Request and parse the SpaceX launch data using the GET request

To make the requested JSON results more consistent, we will use the following static response object for this project:

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets
```

We should see that the request was successfull with the 200 status response code

```
response.status_code
```

```
200
```

3

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

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

Using the dataframe `data` print the first 5 rows

```
# Get the head of the dataframe
data.head()
```

Data Collection - Scraping

1 Using this URL

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

2

TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

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

```
<Response [200]>
```

3

TASK 2: Extract all column/variable names from the HTML table header

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please check the external reference link towards the end of this lab

```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables=soup.find_all('table')
```

4

TASK 3: Create a data frame by parsing the launch HTML tables

We will create an empty dictionary with keys from the extracted column names in the previous task. Later, this dictionary will be converted into a Pandas dataframe

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

Data Wrangling

Calculating the non values and the type of each column

Calculating the number and occurence of mission outcome per orbit type

Calculating the number of launches on each site

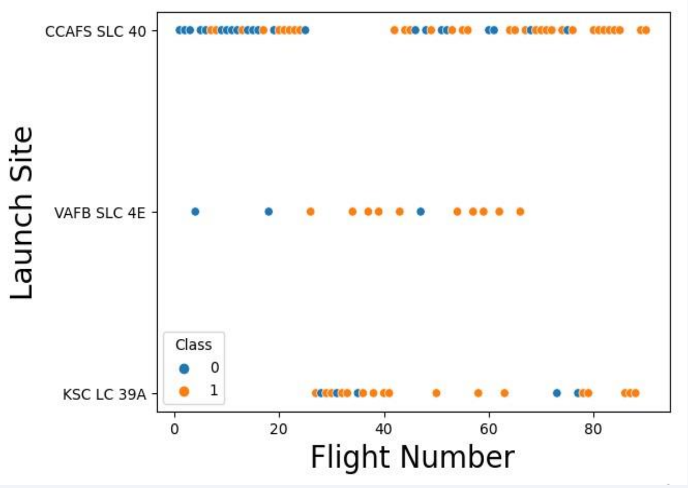
Generating a label for landing outcomes baced on the data in the outcome column

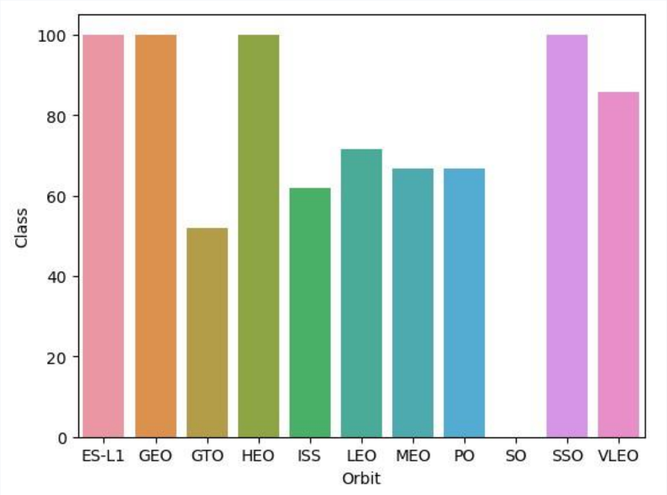
Calculating the number and occurrence of each orbit

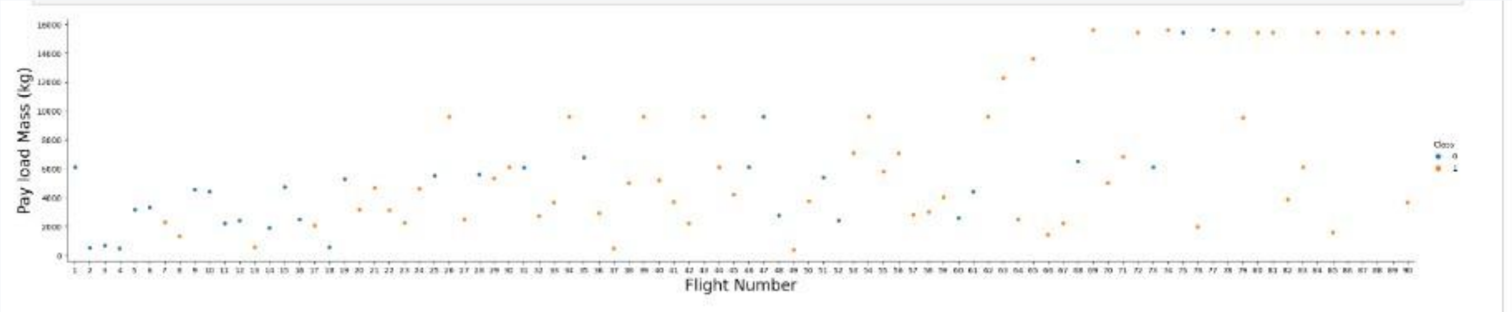
Make a new column(Class) for the labels of outcome column

EDA with Data Visualization

- To explore data, we used of catplot, scatterplot, and barplot to visualize the relationship between columns and finding the importance of each column.





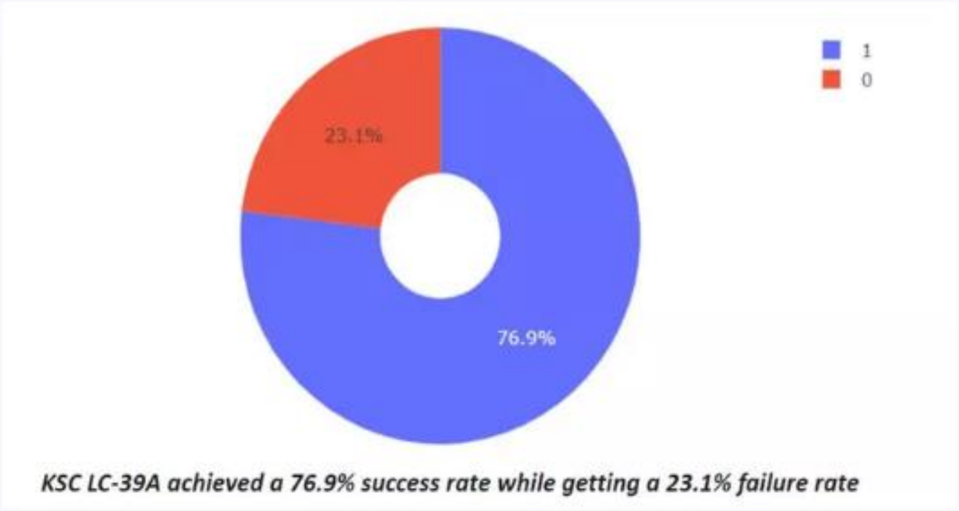


EDA with SQL

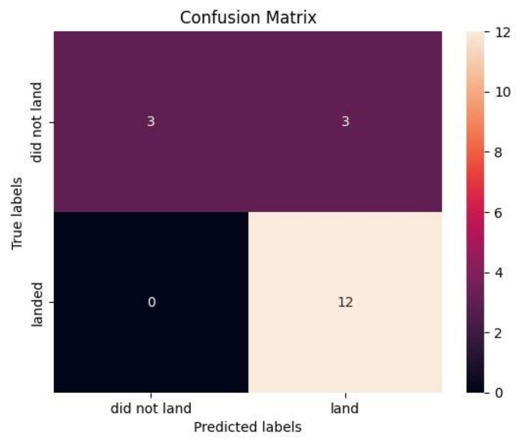
- ✓ Displaying the names of the unique launch sites in the space mission
- ✓ Displaying 5 records where launch sites begin with the string 'CCA‘
- ✓ Displaying the total payload mass carried by boosters launched by NASA (CRS)
- ✓ Displaying average payload mass carried by booster version F9 v1.1
- ✓ Listing the date when the first succesful 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 records which will display the month names, failure landing outcomes in drone ship, booster versions, launch site for the months in year 2015
- ✓ Ranking 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

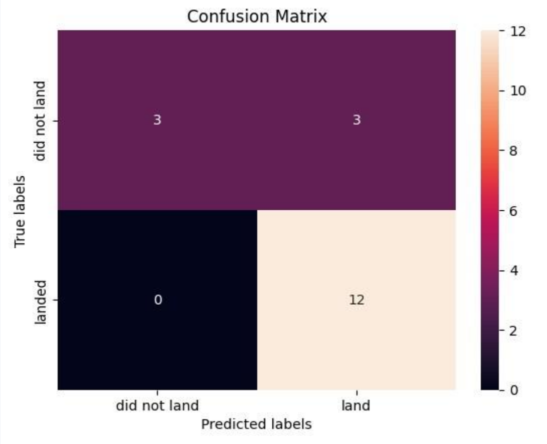
Build an Interactive Map with Folium

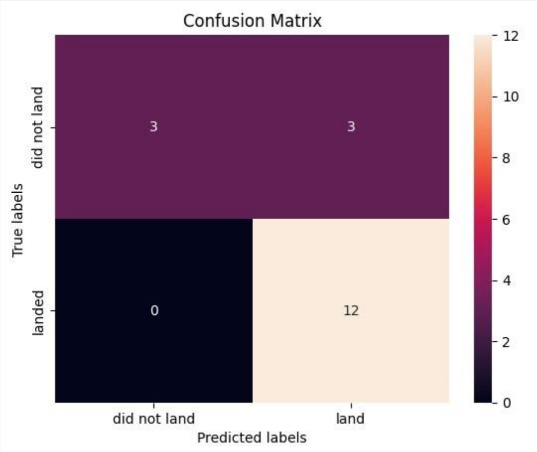
Build a Dashboard with Plotly Dash

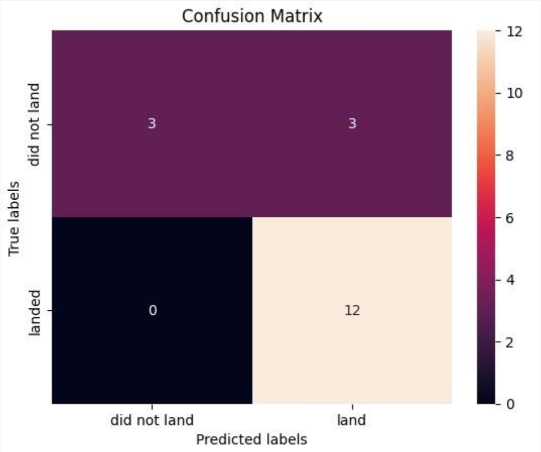


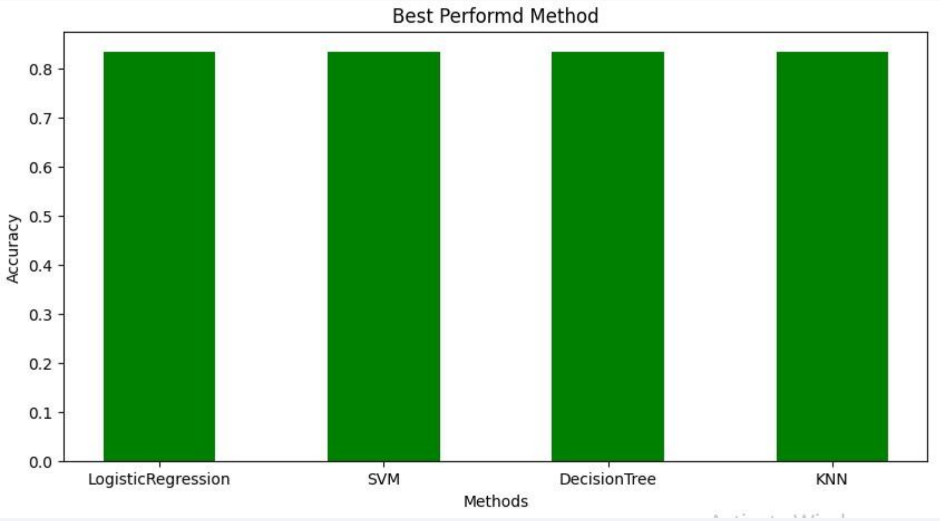
Predictive Analysis (Classification)











- ✓ All of the models, Logestig Regression, KNN, SVM and Decision Tree gave equal accuracy and it was 83.33%. So it doesn't matter which model we use, They’re similar.

Results

- ✓Logistic Regression, KNN ,SVM and Decision Tree models are the best in term of prediction accuracy for this dataset.
- ✓Light weight payload perform better than heavier payload.
- ✓The success rate of SpaceX launches is directly propotional to the number of years the launches are completed.
- ✓Between the launch sites, KSC LC 39A had the most successful launch.
- ✓Among all orbits,GEO, SSO, HEO and ES L1 have the best success rate.

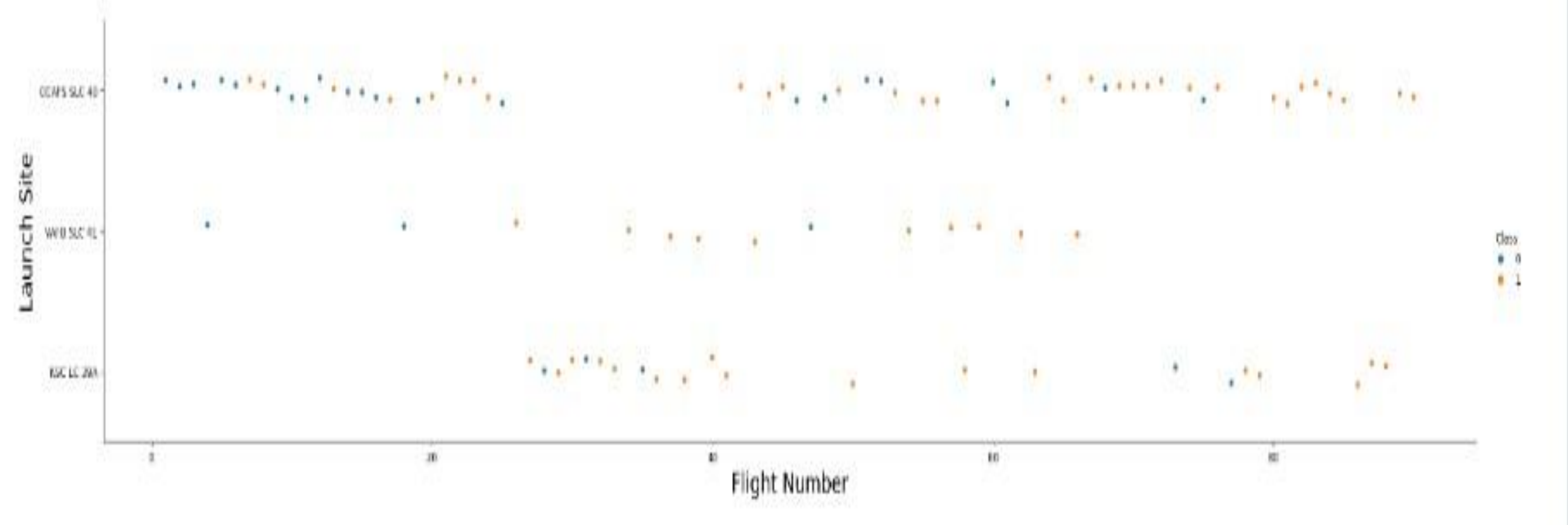
Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

- Show a scatter plot of Flight Number vs. Launch Site

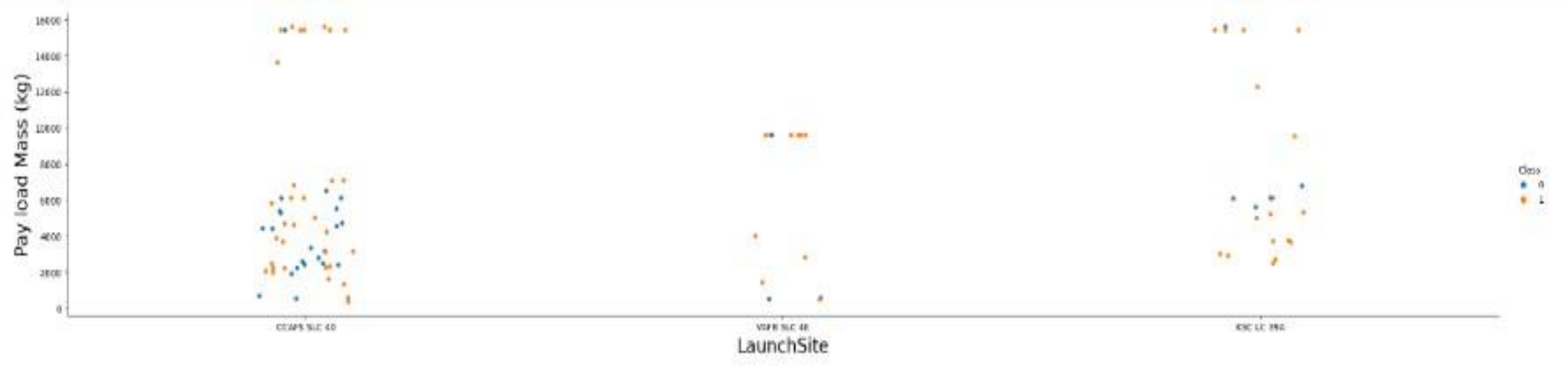
In this plot we have 3 lanch site in two class(0=Failed(Blue) , 1=Success(Orange))



Payload vs. Launch Site

- Show a scatter plot of Payload vs. Launch Site

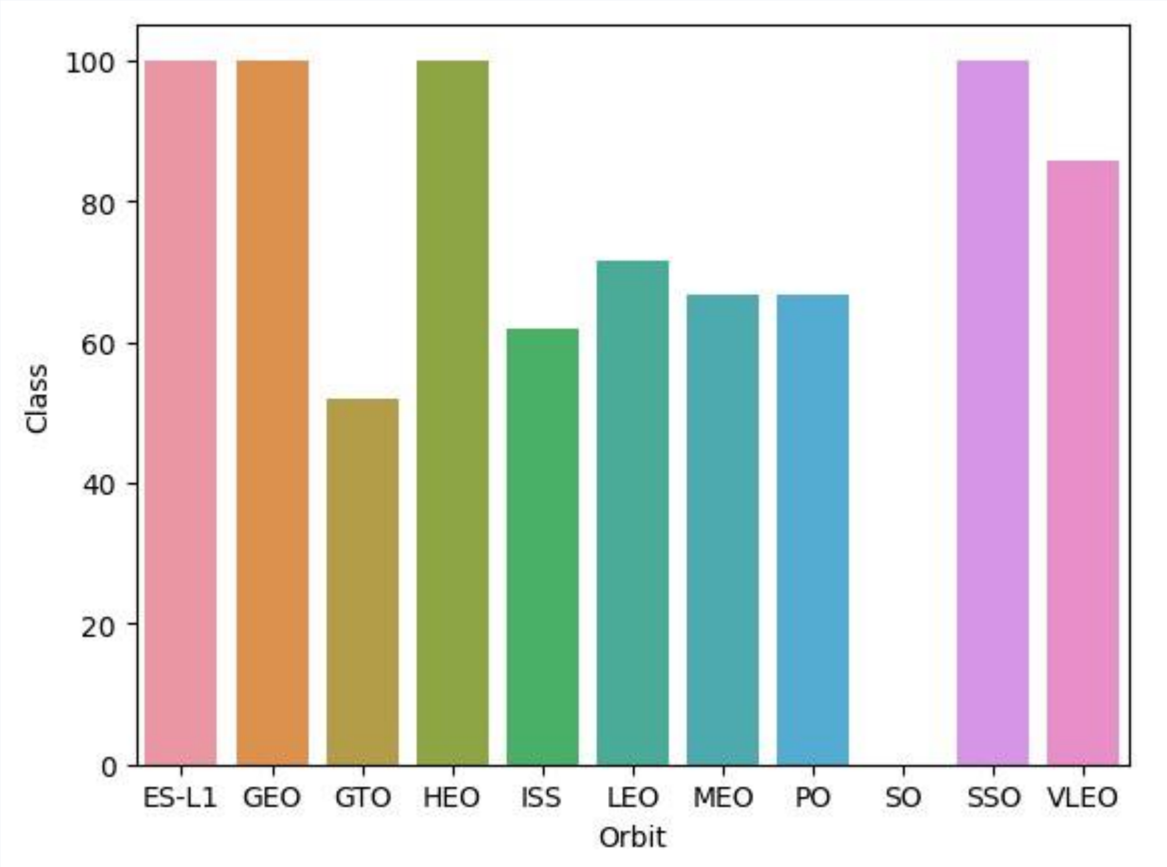
Inclass(0=Failed(Blue) , 1=Success(Orange)) this plot we have 3 launch site and its payload(continuous variable) in two



Success Rate vs. Orbit Type

- Show a bar chart for the success rate of each orbit type

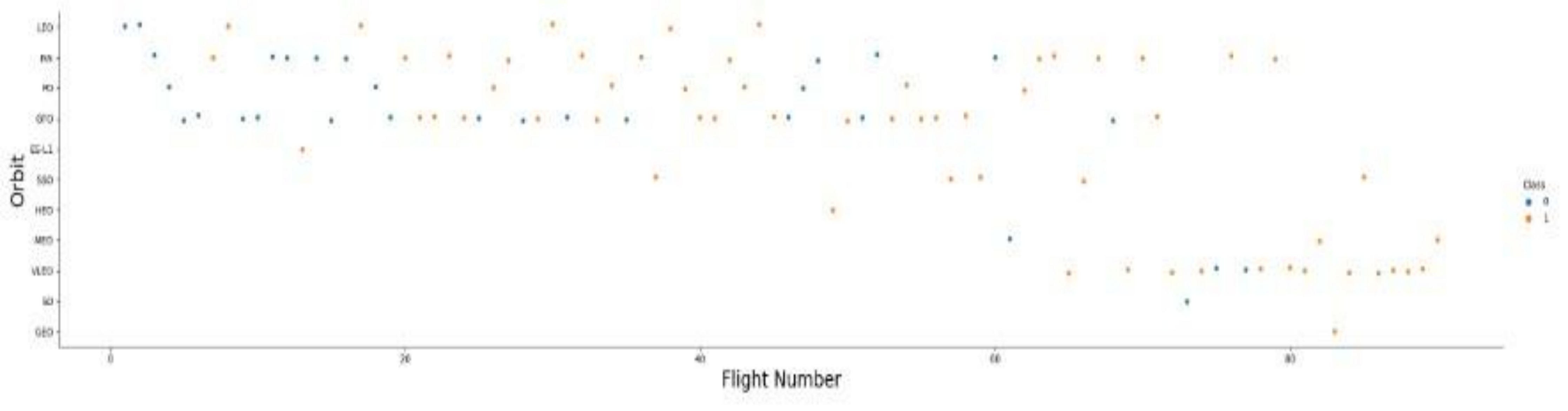
In this plot we show the success rate of each orbit



Flight Number vs. Orbit Type

- Show a scatter point of Flight number vs. Orbit type

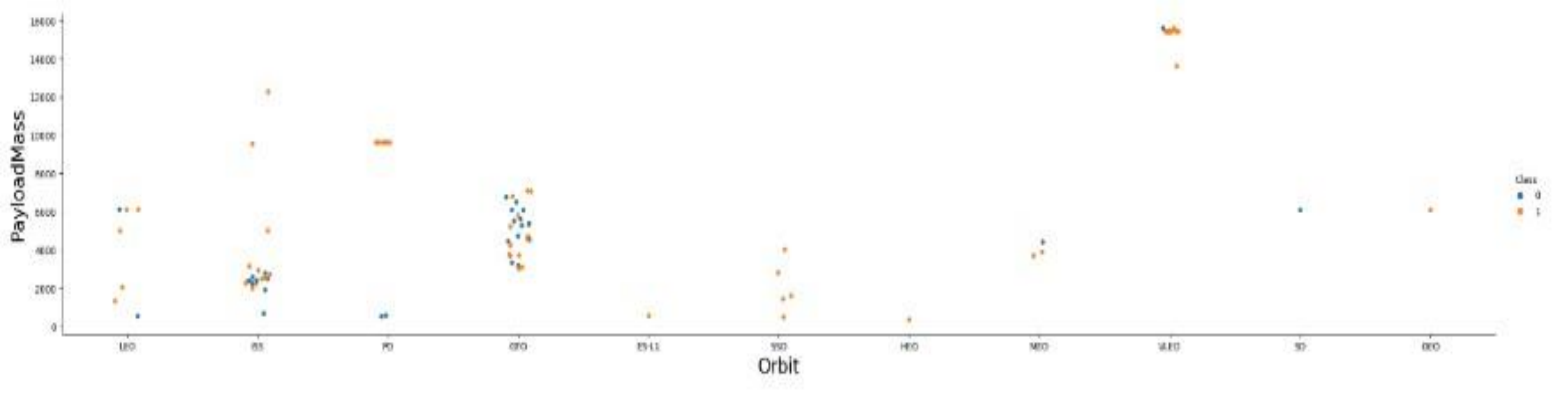
In this plot we show the number of flight for each orbit



Payload vs. Orbit Type

- Show a scatter point of payload vs. orbit type

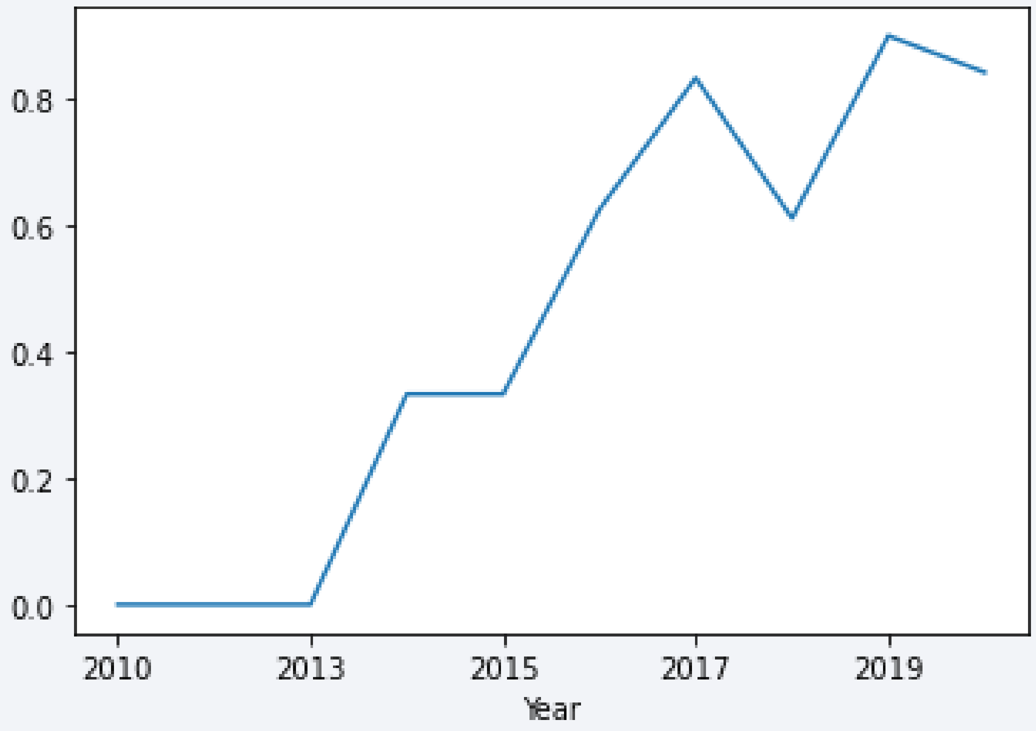
In this plot we show the payload of each orbit in two class(0=Failed(Blue) , 1=Success(Orange))



Launch Success Yearly Trend

- Show a line chart of yearly average success rate

In this plot we have the successs rate for each year(2010-2020)



All Launch Site Names

- We have three uniqe launch site

Out[5]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights
0	1	6/4/2010	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1
1	2	5/22/2012	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1
2	3	3/1/2013	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1
3	4	9/29/2013	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1
4	5	12/3/2013	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1

In [6]:

```
df['LaunchSite'].unique()
```

Out[6]:

```
array(['CCAFS SLC 40', 'VAFB SLC 4E', 'KSC LC 39A'], dtype=object)
```

Launch Site Names Begin with 'CCA'

- The following code there are 5 roes of Launch Site column that start with 'CCA'

```
In [9]: df[df['LaunchSite'].str.startswith('CCA')].head(5)
```

Out[9]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Se
0	1	6/4/2010	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1	0	B0
1	2	5/22/2012	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1	0	B0
2	3	3/1/2013	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1	0	B0
4	5	12/3/2013	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1	0	B1
5	6	1/6/2014	Falcon 9	3325.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1	0	B1

Total Payload Mass

- After using the code below, we understood the total payloadmass carried by boosters from NASA was zero

```
In [10]: total_payload_nasa = df[df['BoosterVersion'] == 'NASA']['PayloadMass'].sum()
         print("Total payload carried by NASA boosters:", total_payload_nasa)

         Total payload carried by NASA boosters: 0.0
```

Average Payload Mass by F9 v1.1

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

```
In [16]: total_payload_nasa = df[df['BoosterVersion'] == 'Falcon 9']['PayloadMass'].sum()
         print("Total payload carried by NASA boosters:", total_payload_nasa)

Total payload carried by NASA boosters: 549446.3470600001
```

First Successful Ground Landing Date

- The dates of the first successful landing outcome on the groung pad were shown here:

```
In [17]: success_df = df[df['Class'] == 1]
         success_landing_dates = success_df.groupby('LandingPad')['Date'].min()
         print("Dates of the first successful landing pad outcomes:")
         print(success_landing_dates)

Dates of the first successful landing pad outcomes:
LandingPad
5e9e3032383ecb267a34e7c7     1/8/2018
5e9e3032383ecb554034e7c9    10/8/2018
5e9e3032383ecb6bb234e7ca    1/29/2020
5e9e3033383ecbb9e534e7cc    1/11/2019
Name: Date, dtype: object
```

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

```
# Find unique values in the 'BoosterVersion' column
unique_boosters = df['BoosterVersion'].unique()

# Filter the DataFrame for boosters with successful landing outcomes, payload ma
filtered_boosters = df[(df['LandingPad'] == 'Success') & (df['PayloadMass'] > 40

# Count the occurrences of each booster
booster_counts = filtered_boosters['BoosterVersion'].value_counts()

# Print the number of unique boosters and the booster names along with their res
print("Number of unique boosters:", len(unique_boosters))
print("\nCount of boosters that successfully landed on drone ship with payload ma
print(booster_counts)
```

```
Number of unique boosters: 1

Count of boosters that successfully landed on drone ship with payload mass betw
een 4000 and 6000:
Series([], Name: BoosterVersion, dtype: int64)
```

Total Number of Successful and Failure Mission Outcomes

- Here is the total number of successful and failure mission outcomes

```
print(df["Class"].mean())
print(df["Class"].value_counts())
```

```
0.6666666666666666
1    60
0    30
Name: Class, dtype: int64
```

Boosters Carried Maximum Payload

```
In [38]: max_payloads = df.groupby('BoosterVersion')['PayloadMass'].max()

         # Find the booster(s) with the maximum payload mass
         max_payload_boosters = max_payloads[max_payloads == max_payloads.max()]

         # Print the names of boosters with the maximum payload mass
         print("Booster(s) with the maximum payload mass:")
         for booster in max_payload_boosters.index:
             print(booster)

         Booster(s) with the maximum payload mass:
         Falcon 9
```

2015 Launch Records

- Here is a List the failed landing outcomes in drone ship, their booster versions, and launch site names for in year 2015

```
In [32]: df['Date'] = pd.to_datetime(df['Date'], format='%m/%d/%Y')

         # Filter the DataFrame for dates with year 2015
         dates_2015 = df[df['Date'].dt.year == 2015]

         # Print the results
         print("Dates from the year 2015:")
         print(dates_2015)
```

```
Dates from the year 2015:
    FlightNumber       Date BoosterVersion  PayloadMass  Orbit    LaunchSite  \
11            12 2015-01-10       Falcon 9       2395.0    ISS  CCAFS SLC 40
12            13 2015-02-11       Falcon 9        570.0  ES-L1  CCAFS SLC 40
13            14 2015-04-14       Falcon 9       1898.0    ISS  CCAFS SLC 40
14            15 2015-04-27       Falcon 9       4707.0    GTO  CCAFS SLC 40
15            16 2015-06-28       Falcon 9       2477.0    ISS  CCAFS SLC 40
16            17 2015-12-22       Falcon 9       2034.0    LEO  CCAFS SLC 40

       Outcome  Flights  GridFins  Reused   Legs                LandingPad  \
11  False ASDS        1      True   False   True  5e9e3032383ecb761634e7cb
12  True Ocean        1      True   False   True                       NaN
13  False ASDS        1      True   False   True  5e9e3032383ecb761634e7cb
14   None None        1     False   False  False                       NaN
15   None ASDS        1      True   False   True  5e9e3032383ecb6bb234e7ca
16   True RTLS        1      True   False   True  5e9e3032383ecb267a34e7c7

    Block  ReusedCount Serial  Longitude   Latitude  Class
11      1            0  B1012 -80.577366  28.561857      0
12      1            0  B1013 -80.577366  28.561857      1
13      1            0  B1015 -80.577366  28.561857      0
14      1            0  B1016 -80.577366  28.561857      0
15      1            0  B1018 -80.577366  28.561857      0
16      1            0  B1019 -80.577366  28.561857      1
```

```
In [35]: df['Date'] = pd.to_datetime(df['Date'], format='%m/%d/%Y')

         # Filter the DataFrame for failed landing outcomes (Class 0) and the year 2015
         failed_landings_2015 = df[(df['Class'] == 0) & (df['Date'].dt.year == 2015)]

         # Print the results
         for index, row in failed_landings_2015.iterrows():
             print("Booster Version:", row['BoosterVersion'])
             print("Launch Site:", row['LaunchSite'])
             print("------------------------------")
```

```
Booster Version: Falcon 9
Launch Site: CCAFS SLC 40
------------------------------
Booster Version: Falcon 9
Launch Site: CCAFS SLC 40
------------------------------
Booster Version: Falcon 9
Launch Site: CCAFS SLC 40
------------------------------
Booster Version: Falcon 9
Launch Site: CCAFS SLC 40
------------------------------
```

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

- Here is the 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

In [37]:

```
# Convert the 'Date' column to datetime format
df['Date'] = pd.to_datetime(df['Date'], format='%m/%d/%Y')

# Filter the DataFrame for the specified date range
start_date = '2010-06-04'
end_date = '2017-03-20'
filtered_data = df[(df['Date'] >= start_date) & (df['Date'] <= end_date)]

# Count the landing outcomes based on 'LandingPad', 'Class', and 'Date'
landing_outcome_counts = filtered_data.groupby(['LandingPad', 'Class'])['Class']

# Sort the counts in descending order
sorted_counts = landing_outcome_counts.sort_values(ascending=False)

# Print the results
print("Ranking of landing outcomes between", start_date, "and", end_date, ":\n")
print(sorted_counts)
```

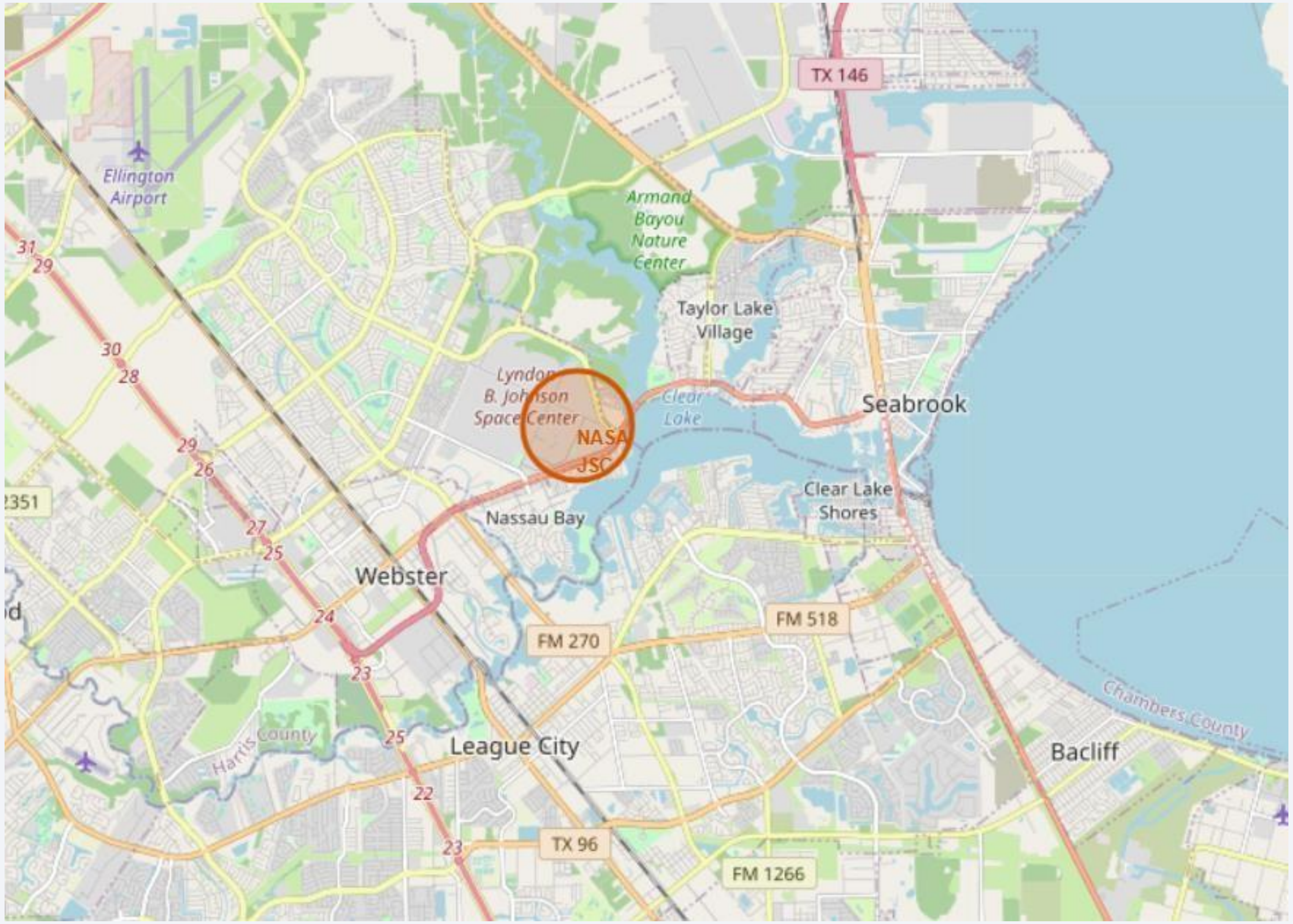
```
Ranking of landing outcomes between 2010-06-04 and 2017-03-20 :

LandingPad                Class
5e9e3032383ecb6bb234e7ca  1        4
5e9e3032383ecb267a34e7c7  1        3
5e9e3032383ecb6bb234e7ca  0        3
5e9e3032383ecb761634e7cb  0        2
5e9e3033383ecbb9e534e7cc  0        1
                          1        1
Name: Class, dtype: int64
```

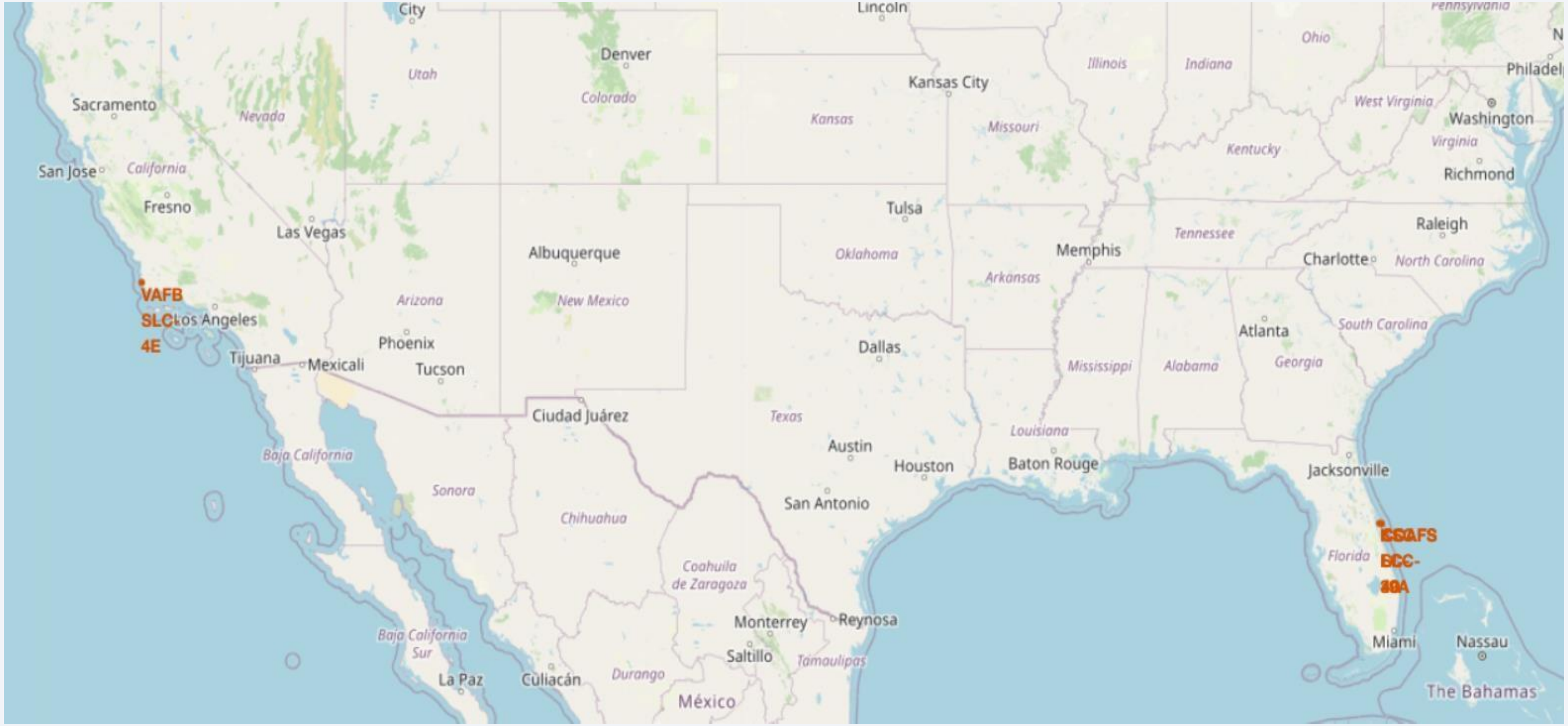
Section 3

Launch Sites Proximities Analysis

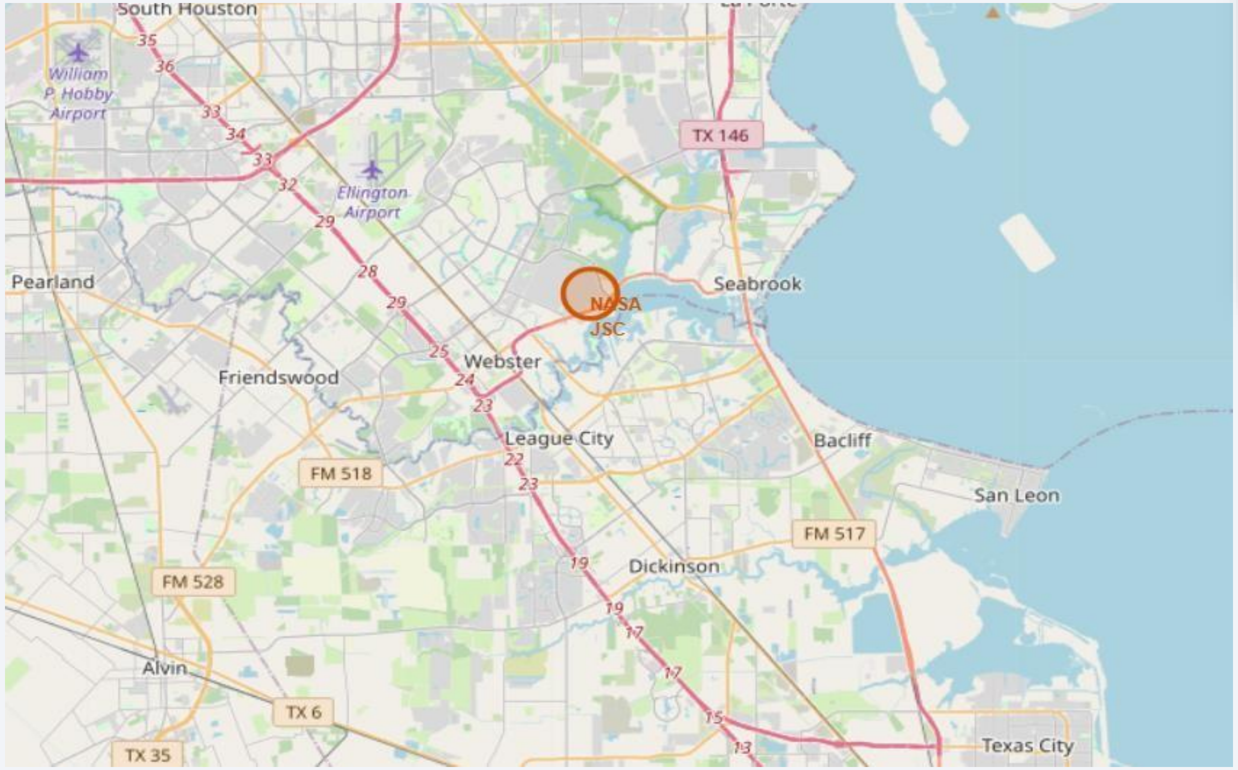
<Folium Map Screenshot 1>

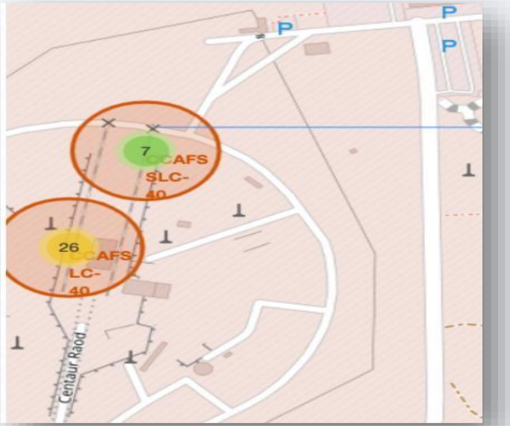


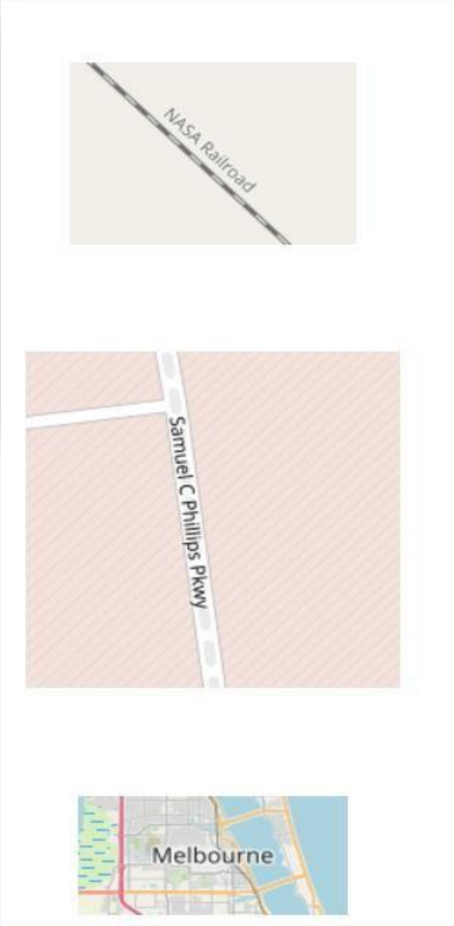
<Folium Map Screenshot 2>



<Folium Map Screenshot 3>





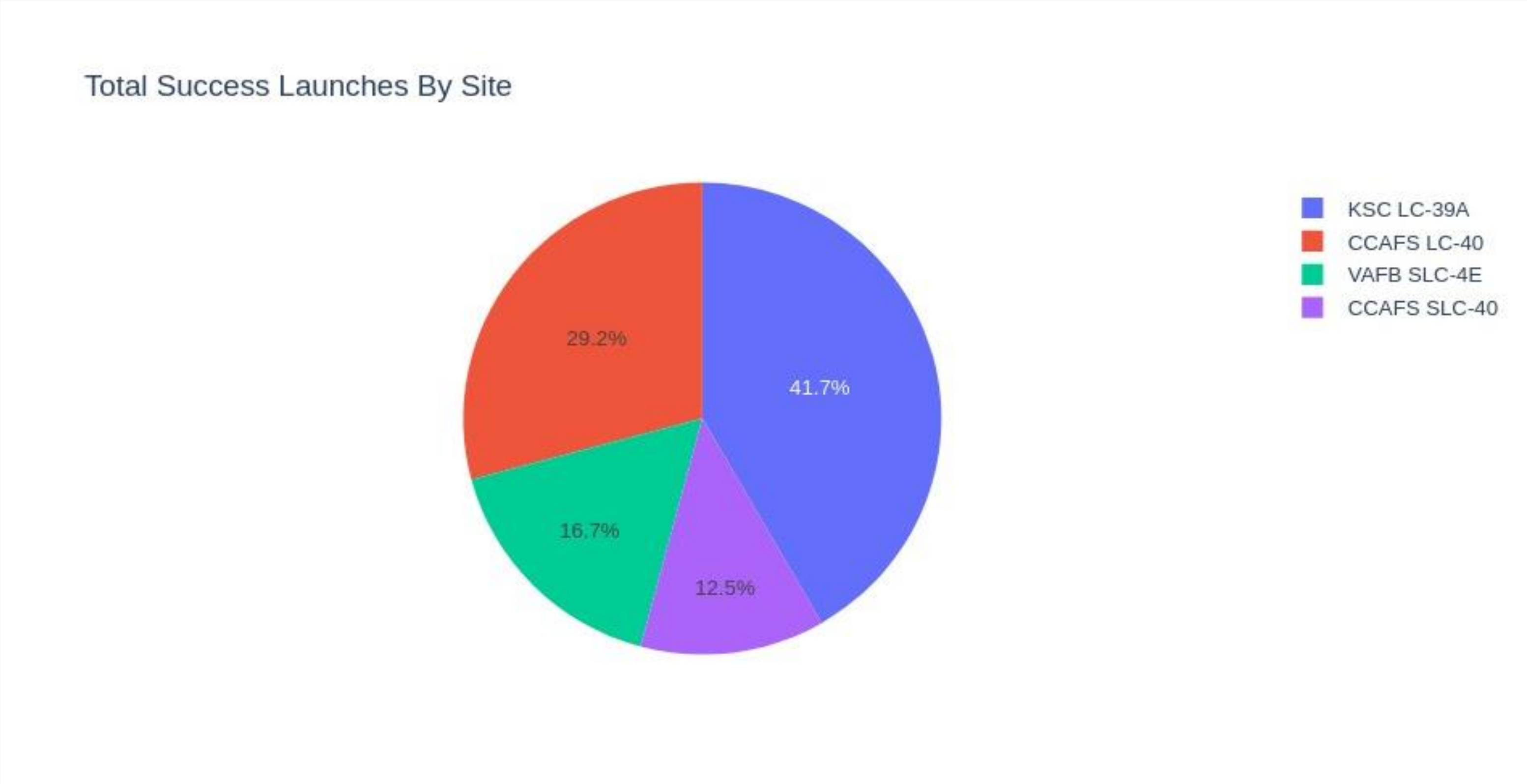


Section 4

Build a Dashboard with Plotly Dash

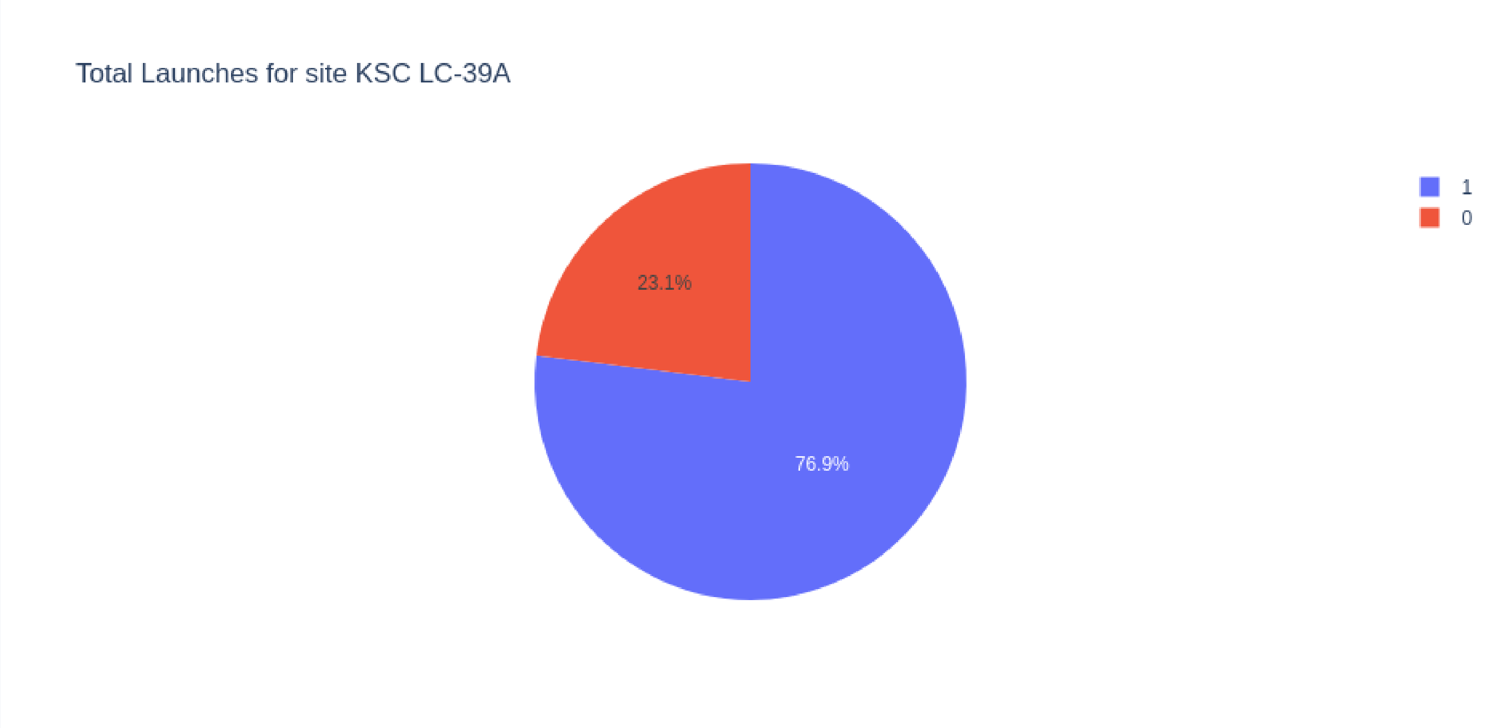
<Dashboard Screenshot 1>

- The place where launchs done successfully is so important and with this plot we can found the best site.

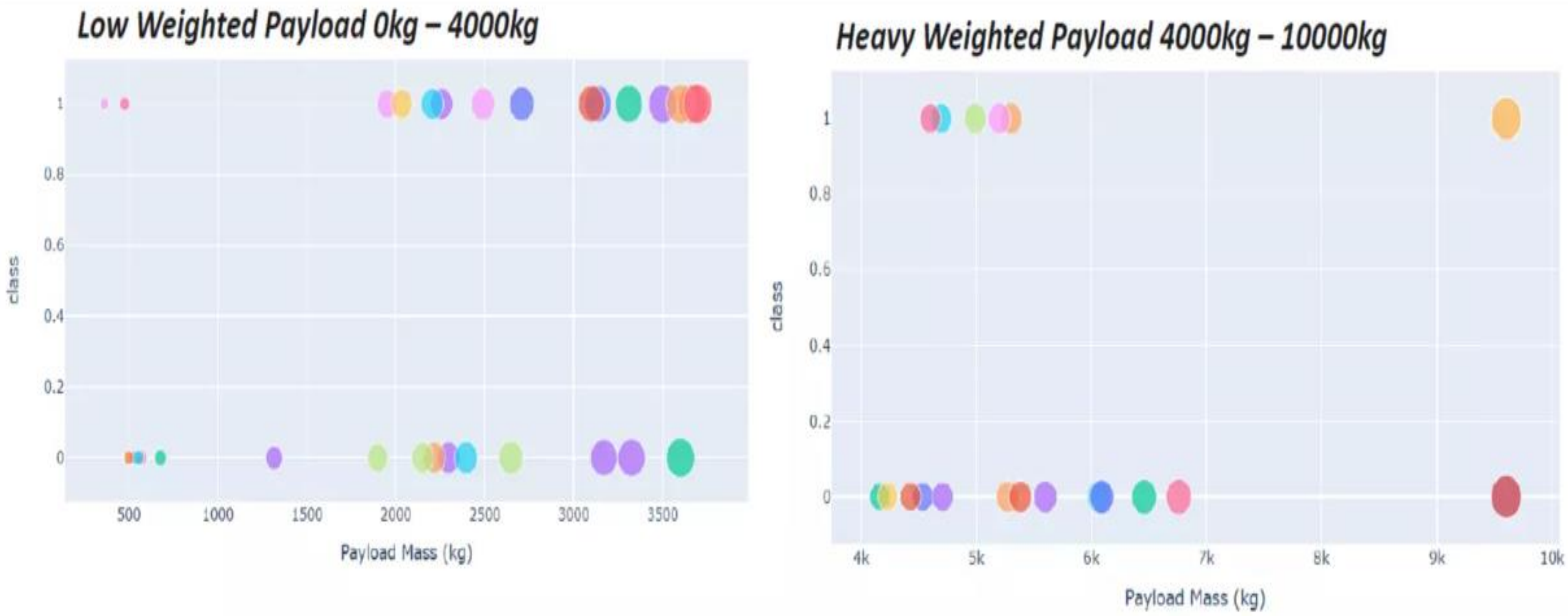


<Dashboard Screenshot 2>

- in this plot we can see the rate of success in KSC LC-39A site



<Dashboard Screenshot 3>



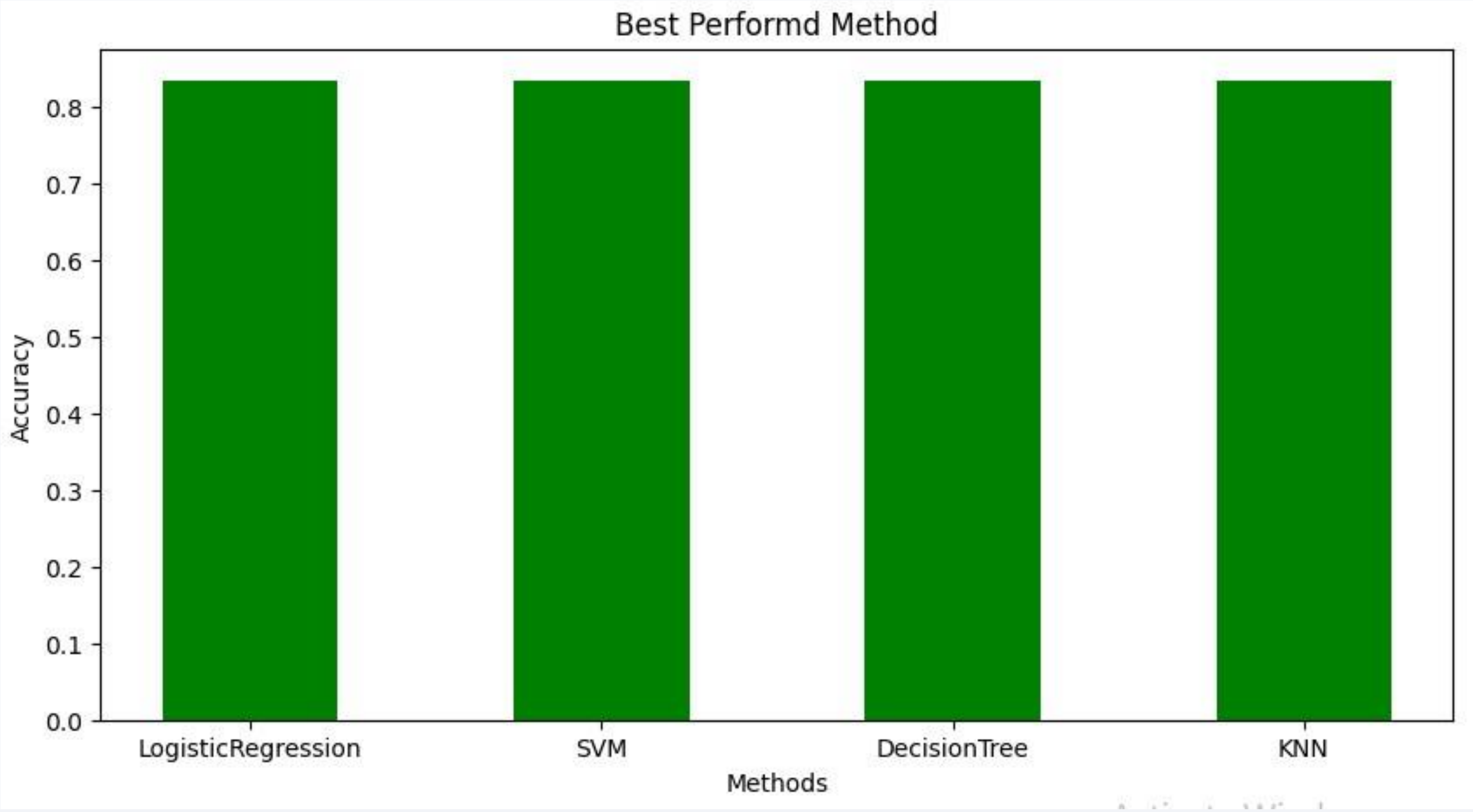
We can see the success rates for low weighted payloads is higher than the heavy weighted payloads

Section 5

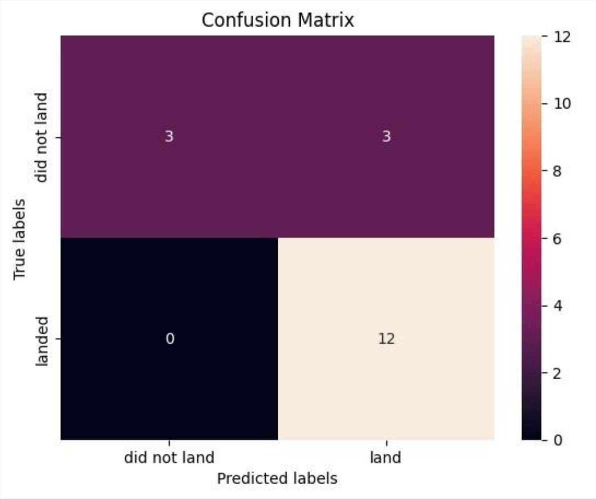
Predictive Analysis (Classification)

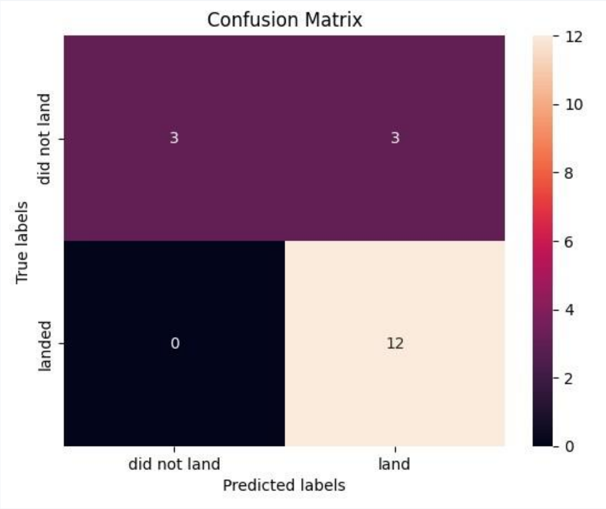
Classification Accuracy

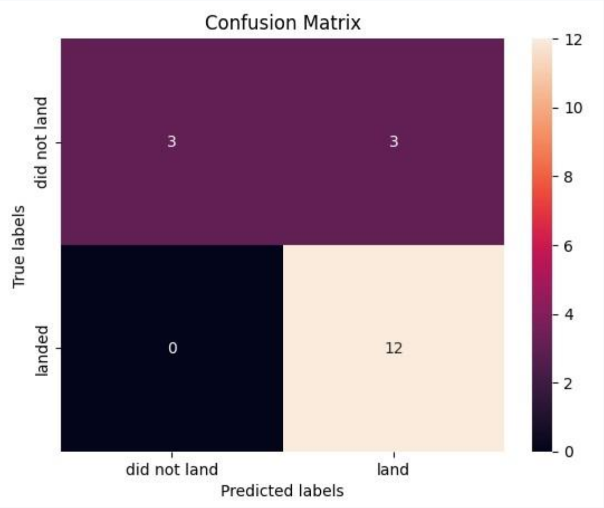
- After modeling and using confusion matrix, we see the accuracy rate of all models with GridSearch are equal.

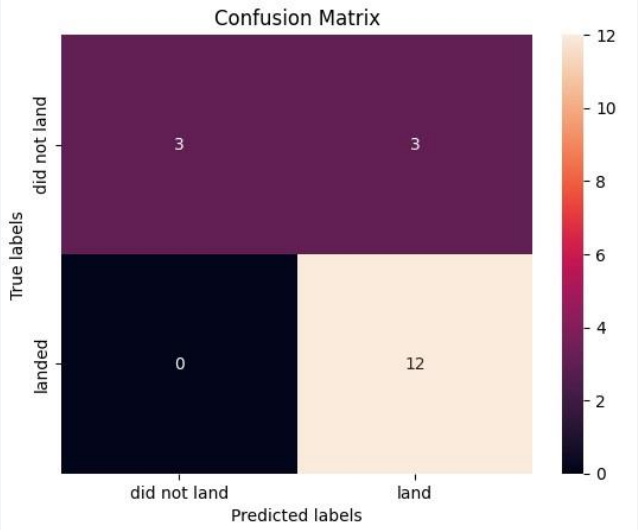


Confusion Matrix









Conclusions

- ✓Logistic Regression, KNN ,SVM and Decision Tree models are the best in term of prediction accuracy for this dataset.
- ✓Light weight payload perform better than heavier payload.
- ✓The success rate of SpaceX launches is directly propotional to the number of years the launches are completed.
- ✓Between the launch sites, KSC LC 39A had the most successful launch.
- ✓Among all orbits,GEO, SSO, HEO and ES L1 have the best success rate.

Thank you!