Winning Space Race with Data Science

BAMIDELE OPEYEMI
27th January, 2023

Outline

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

Executive Summary

- The concept of this project is to make a prediction if the first stage of Falcon-9 will land successfully at every rocket launch by SpaceX.

- SpaceX had advertised Falcon 9 rocket launches on its website with a cost of 62 million dollars whereas other providers cost upward of 165 million dollars each, this is been so for SpaceX because much of the savings is gotten from the reuse the Falcon 9 first stage. Therefore if we can determine if the first stage will land successfully, we can determine the cost of each launch.

- Results :

- Our prediction is that Falcon 9 first stage will land successfully for every launch.

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 because SpaceX can reuse the Falcon 9 first stage. Therefore if we can determine if the first stage will land, we can determine the cost of a launch.

- Problems

- I am making a prediction to know if the Falcon 9 first stage will land successfully at every rocket launch, this information can be used if an alternate company wants to bid against SpaceX for a rocket launch.

Section 1

Methodology

Methodology

- Data collection methodology: **requests.get**
- Requests allows us to make HTTP requests which was used to get data from SpaceX API
- Perform data wrangling: All missing data were replaced with the mean of sum-total of data per column(Data Normalization), the categorical data were converted to binary numbers through one-hot encoding. This makes the data accessible and easier to analyze.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - We standardize the data and use the function train_test_split to split the data X and Y into training and test data, then used Logistic Regression, Support Vector Machine, Decision Tree and K-Nearest Neighbour for predictive analysis.

Data Collection

- Describe how data sets were collected.
- You need to present your data collection process use key phrases and flowcharts
- Data were collected from the SpaceX API using request.get
- Requests allows us to make HTTP requests which I used to get data from SpaceX API
- **import** requests
- **import** pandas **as** pd
- *Takes the dataset and uses the rocket column to call the API and append the data to the list*
- **def** getBoosterVersion(data):
- **for** x **in** data['rocket']:
- **if** x:
- response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
- BoosterVersion.append(response['name'])
-
-

Data Collection – SpaceX API

Taking the dataset and using the cores column to call the API and append the data to the lists

```
def getCoreData(data):
    for core in data['cores']:
            if core['core'] != None:
                response = requests.get("https://api.spacexdata.com/v4/cores/"+core['core']).json()
                Block.append(response['block'])
                ReusedCount.append(response['reuse_count'])
                Serial.append(response['serial'])
            else:
                Block.append(None)
                ReusedCount.append(None)
                Serial.append(None)
            Outcome.append(str(core['landing_success'])+' '+str(core['landing_type']))
            Flights.append(core['flight'])
            GridFins.append(core['gridfins'])
```

Taking the dataset and using the launchpad column to call the API and append the data to the list

```
def getLaunchSite(data):
    for x in data['launchpad']:
        if x:
            response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
            Longitude.append(response['longitude'])
            Latitude.append(response['latitude'])
            LaunchSite.append(response['name'])
```

Taking the dataset and using the payloads column to call the API and append the data to the lists

```
def getPayloadData(data):
    for load in data['payloads']:
        if load:
            response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
            PayloadMass.append(response['mass_kg'])
            Orbit.append(response['orbit'])
```

Data Collection - Scraping

using requests.get() method with the provided static_url and assigning the response to a object

```
'df=requests.get(static_url).text'
```

Using BeautifulSoup() to create a BeautifulSoup object from a response text content

```
'soup=BeautifulSoup(df, 'html.parser')'
```

Using soup.title attribute

```
'soup.title'
```

I used the find_all function in the BeautifulSoup object, with element type `table` and assign the result to a list called `html_tables`

```
'html_tables=soup.find_all('table')'
```

- https://github.com/DATASCIENTISTBAMIDELE/Pthon/blob/main/jupyter-labs-webscraping%20(1).ipynb

Starting from the third table was our target:

```
first_launch_table = html_tables[3]
print(first_launch_table)
```

I needed to iterate through the <th> elements and apply the provided extract_column_from_header() to extract column name one by one

```
column_names = []
labels = first_launch_table.find_all('th')
for label in labels:
    name = extract_column_from_header(label)
   # header = str(label.text).strip()
    #header = str(header).split("($)Footnote", 1)[0]
    if name != None:
      if len(name) > 0:
          column_names.append(name)
print(column_names)  and the dataframes.
```

Data Wrangling

All missing data were replaced with the mean of sum-total of data per column(Data Normalization), the categorical data were converted to binary numbers through one-hot encoding. This makes the data accessible and easier to analyze.

Hierarchical Data, Handling categorical data, reshaping and transforming structures, one-hot encoding using the data.get_dummiess() method.

Below is the data wrangling process/flowchart

Identify and calculate the percentage of the missing values in each attribute

df.isnull().sum()/df.shape[0]*100

We use the method value_counts() on the column LaunchSite to determine the number of launches on each site

df['LaunchSite'].value_counts()

```
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
```

landing_class = 0 if bad_outcome while landing_class = 1 otherwise

```
landing_class = []
for outcome in df['Outcome']:
    if outcome in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

EDA with Data Visualization

First, plotting the FlightNumber vs. PayloadMass using a catplot and a scatter plot, it was observed that the higher the flight number, the first stage is more likely to land successfully. Also the more massive the PayloadMas, the less likely the first stage will return.

We see that different launch sites have different success rates. CCAFS LC-40, has a success rate of 60 %, while KSC LC-39A and VAFB SLC 4E has a success rate of 77%.

Secondly, a Visualization scatter plot and bar chart was done to show the relationship between success rate of each orbit type, and it was observed that several orbit, ES-L1, GEO, HEO, SSO and VLEO had high success rates.

Also, I made a line plot of the success rate against the extracted year and we observed that the success rate kept increasing from 2013 till 2020

https://github.com/DATASCIENTISTBAMIDELE/Pthon/blob/main/IBM-DS0321EN-SkillsNetwork_labs_module_2_jupyter-labs-eda-dataviz.ipynb.jupyterlite%20(1).ipynb

EDA with SQL

First, plotting the FlightNumber vs. PayloadMass using a catplot and a scatter plot, it was observed that the higher the flight number, the first stage is more likely to land successfully. Also the more massive the PayloadMas, the less likely the first stage will return.

We see that different launch sites have different success rates. CCAFS LC-40, has a success rate of 60 %, while KSC LC-39A and VAFB SLC 4E has a success rate of 77%.

Secondly, a Visualization scatter plot and bar chart was done to show the relationship between success rate of each orbit type, and it was observed that several orbit, ES-L1, GEO, HEO, SSO and VLEO had high success rates.

Also, I made a line plot of the success rate against the extracted year and we observed that the success rate kept increasing from 2013 till 2020

https://github.com/DATASCIENTISTBAMIDELE/Pthon/blob/main/IBM-DS0321EN-SkillsNetwork_labs_module_2_jupyter-labs-eda-dataviz.ipynb.jupyterlite%20(1).ipynb

Build an Interactive Map with Folium

All the launch_sites were located on the Folium map: CCAFS LC-40, CCAFS SLC-40, KSC LC-39A, VAFB SLC-4E using their latitude and longitude coordinates

Combining all the maps together:

```
latitude = 28.562302
longitude = -80.577356
All_map = folium.Map(location=[latitude, longitude], zoom_start=4)
for lat, lng, name  in zip(launch_sites_df.Lat, launch_sites_df.Long, launch_sites_df.Launch):
    All_map.add_child(folium.Circle([lat,lng],radius=1000, color='#d35400',
    fill=True).add_child(folium.Popup(name))
                      )
    All_map.add_child(folium.map.Marker([lat,lng],
                       icon=DivIcon(icon_size=(20,20),
                       icon_anchor=(0,0),
                       html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % name,))
                      )
All_map
```

- https://github.com/DATASCIENTISTBAMIDELE/Pthon/blob/main/IBM-DS0321EN-SkillsNetwork_labs_module_3_lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

All the launch_sites were located on the Folium map: CCAFS LC-40, CCAFS SLC-40, KSC LC-39A, VAFB SLC-4E using their latitude and longitude coordinates

Combining all the maps together:

```
latitude = 28.562302
longitude = -80.577356
All_map = folium.Map(location=[latitude, longitude], zoom_start=4)
for lat, lng, name  in zip(launch_sites_df.Lat, launch_sites_df.Long, launch_sites_df.Launch):
    All_map.add_child(folium.Circle([lat,lng],radius=1000, color='#d35400',
    fill=True).add_child(folium.Popup(name))
                      )
    All_map.add_child(folium.map.Marker([lat,lng],
                       icon=DivIcon(icon_size=(20,20),
                       icon_anchor=(0,0),
                       html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % name,))
                      )
All_map
```

- https://github.com/DATASCIENTISTBAMIDELE/Pthon/blob/main/IBM-DS0321EN-SkillsNetwork_labs_module_3_lab_jupyter_launch_site_location.jupyterlite.ipynb

Predictive Analysis (Classification)

At the initial stage, I standardize X-data using: X=preprocessing.StandardScaler().fit(X).transform(X)

I split the data into training and testing data using the function train_test_split. The training data is divided into validation data, a second set used for training data; then the models are trained and hyperparameters are selected using the function GridSearchCV.

We output the GridSearchCV object for Logistic Regression, Decision Tree, K-Nearest Neighbour, and Support Vector Machine. We display the best parameters using the data attribute best_params_ and the accuracy on the validation data using the data attribute best_score_

Calculated the accuracy on the test data using the method score and plotted the confusion matrix

Result shows all Predictive analysis are best as the result shows the same.

Flowchart

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

print ('Train set:', X_train.shape,  Y_train.shape)

print ('Test set:', X_test.shape,  Y_test.shape)

parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}

lr=LogisticRegression()

logreg_cv = GridSearchCV(lr, parameters, cv=10)

logreg_cv.fit(X_train, Y_train)
```

https://github.com/DATASCIENTISTBAMIDELE/Pthon/blob/main/IBM-DS0321EN-SkillsNetwork_labs_module_4_SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Results

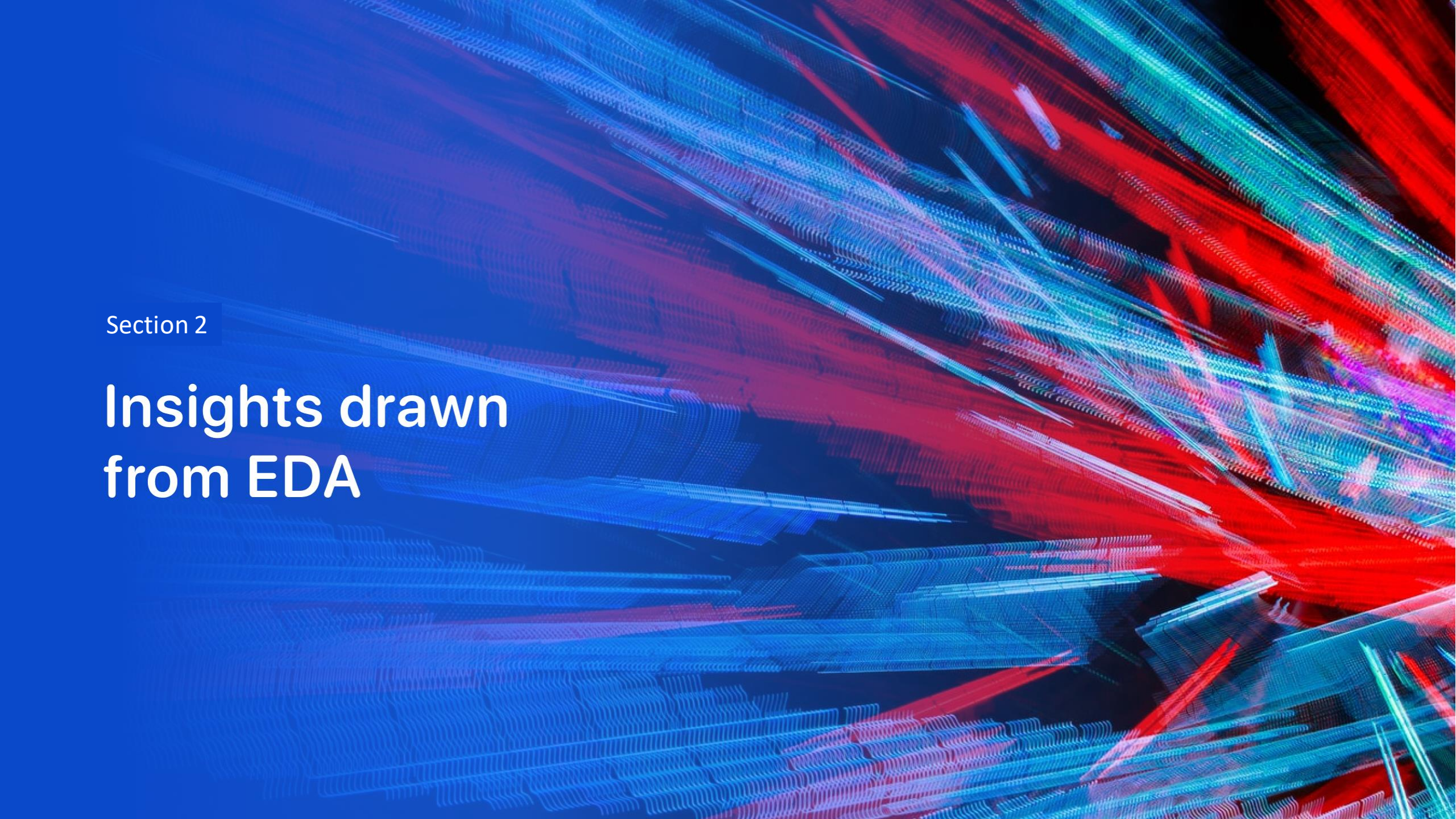
- Exploratory data analysis results

The

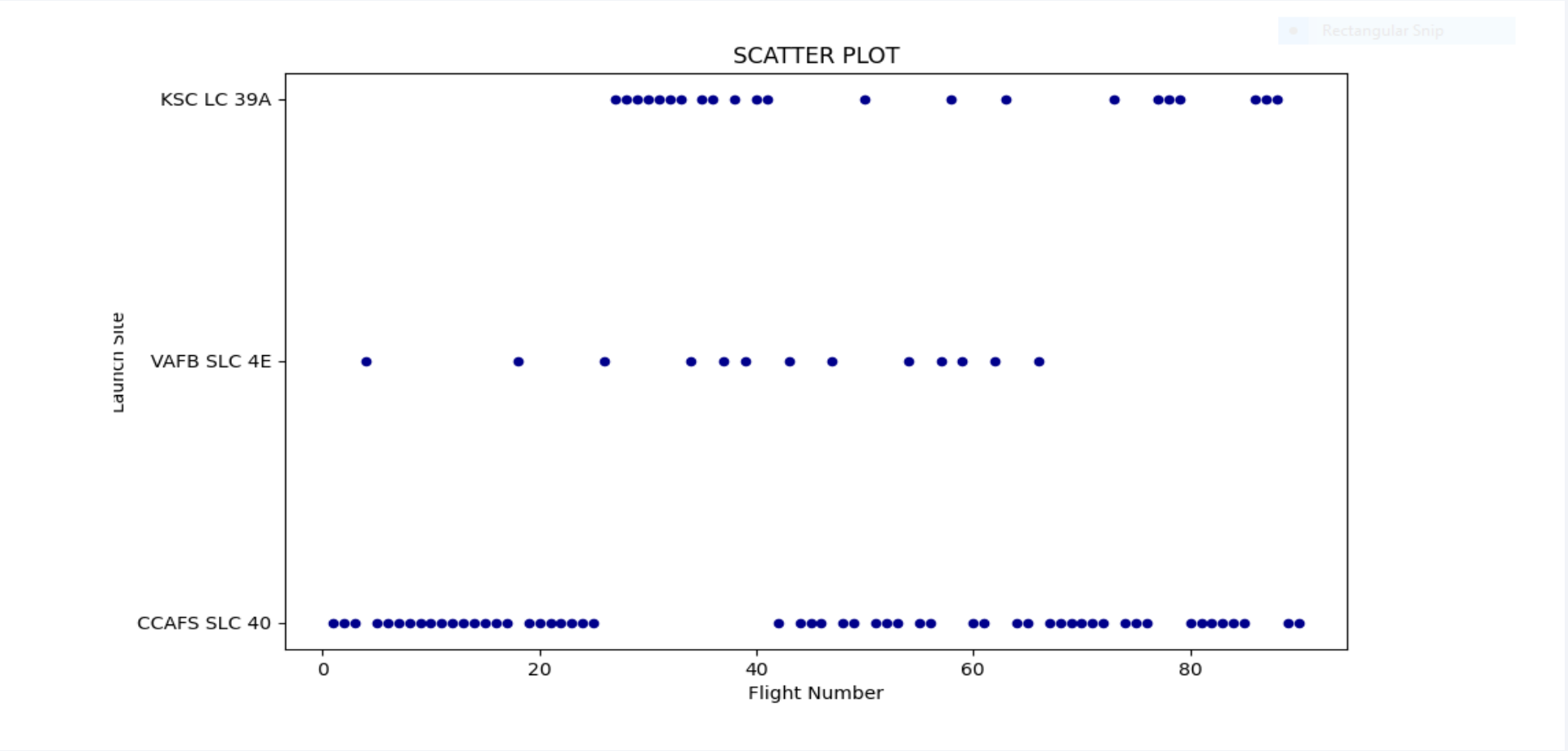
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

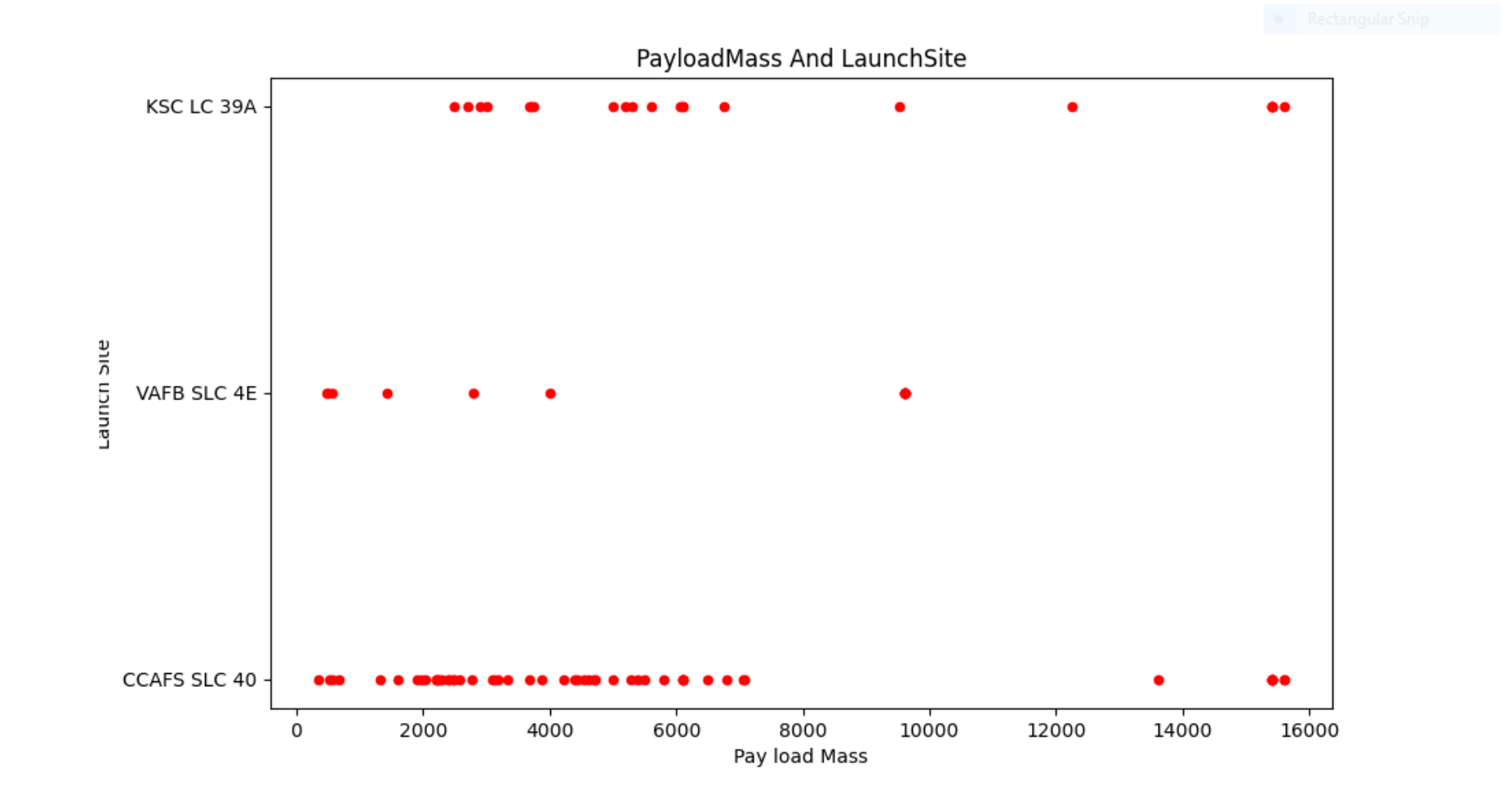


Flight Number vs. Launch Site



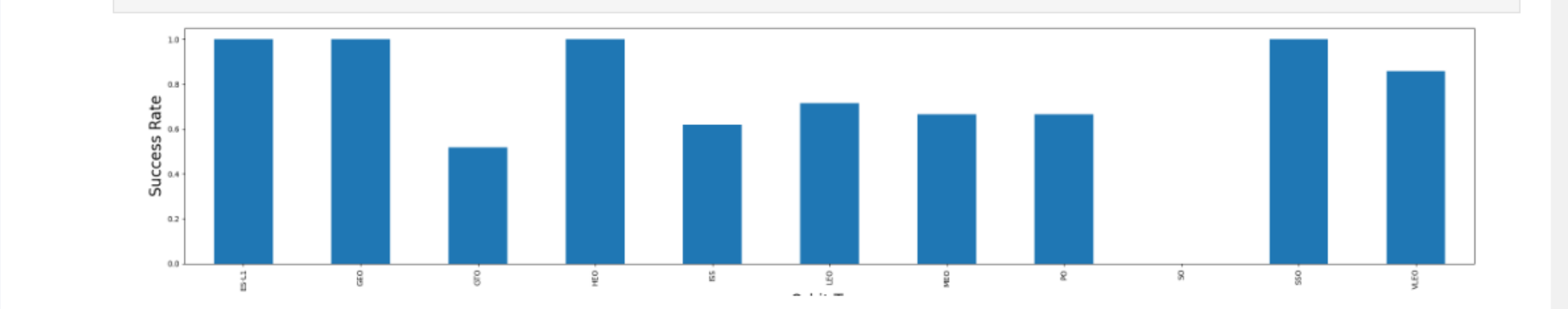
It was observed that more flight number was launched on site CCAFS SLC 40, which most likely had more first stage successful landing likewise KSC LC 39A

Payload vs. Launch Site



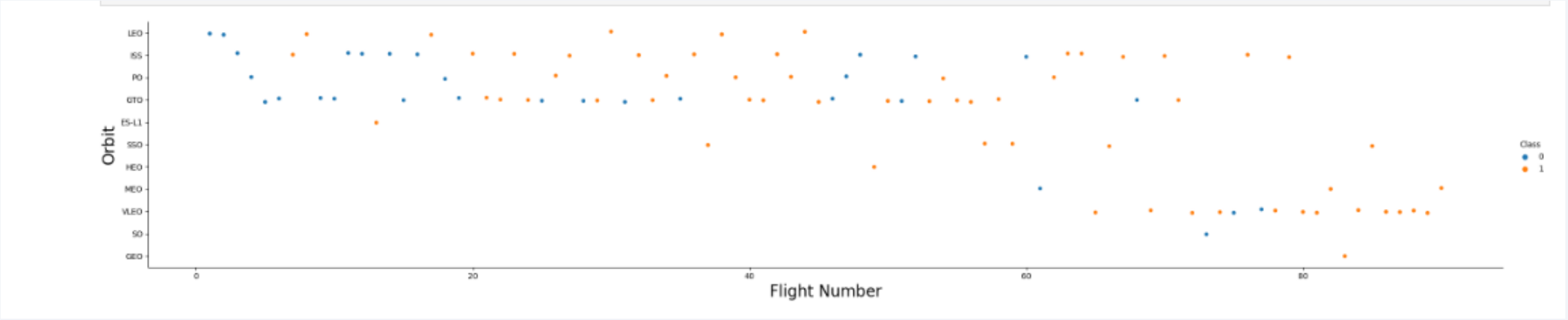
It was observed that the more massive the payload, the less likely the first stage will return. But nevertheless the higher the payload mass the lesser flight was launched.

Success Rate vs. Orbit Type



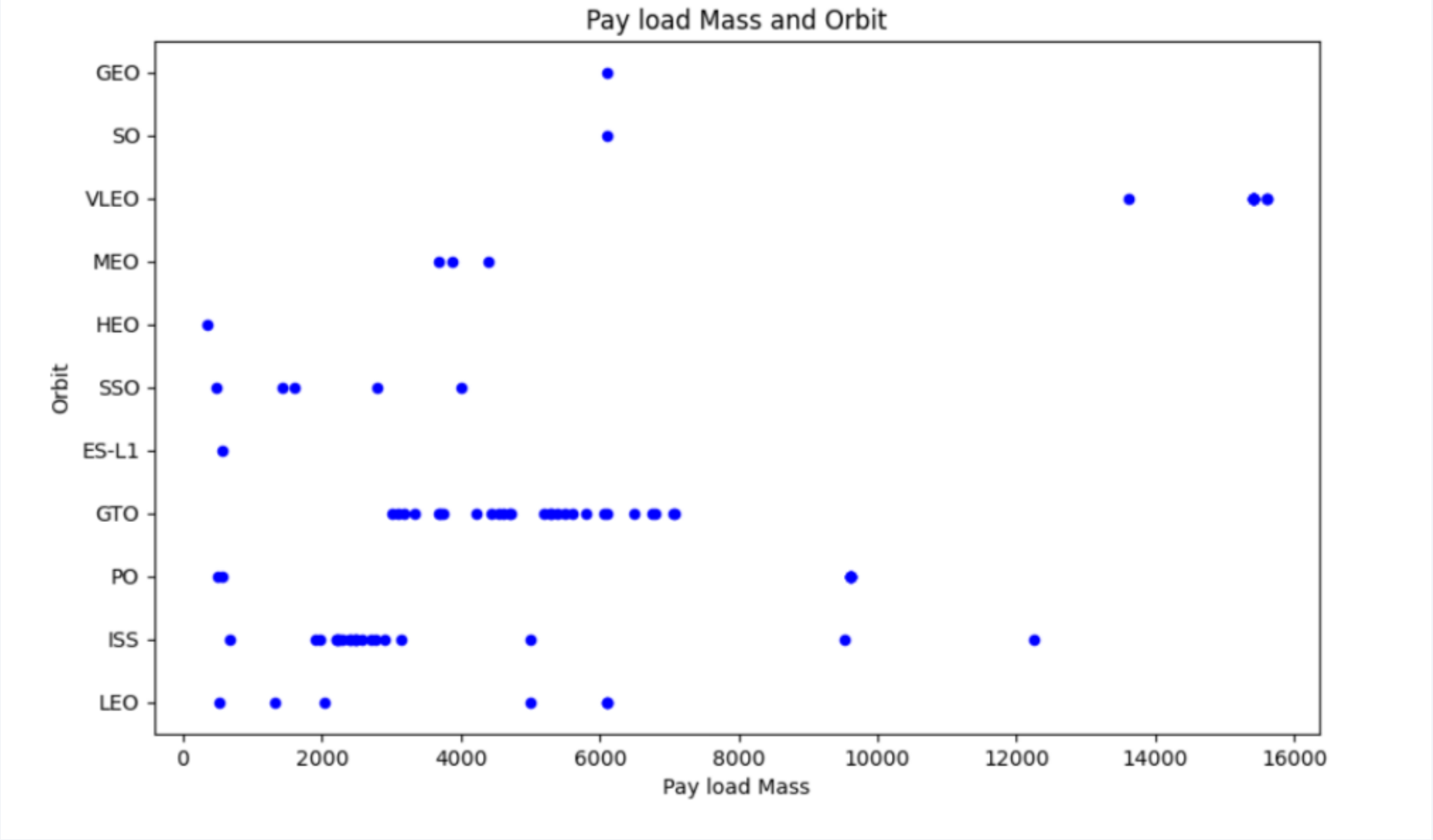
The bar chart shows that the orbit ES-L1, GEO, HEO, SSO had the highest success rates likewise VLEO, which was equally high.

Flight Number vs. Orbit Type



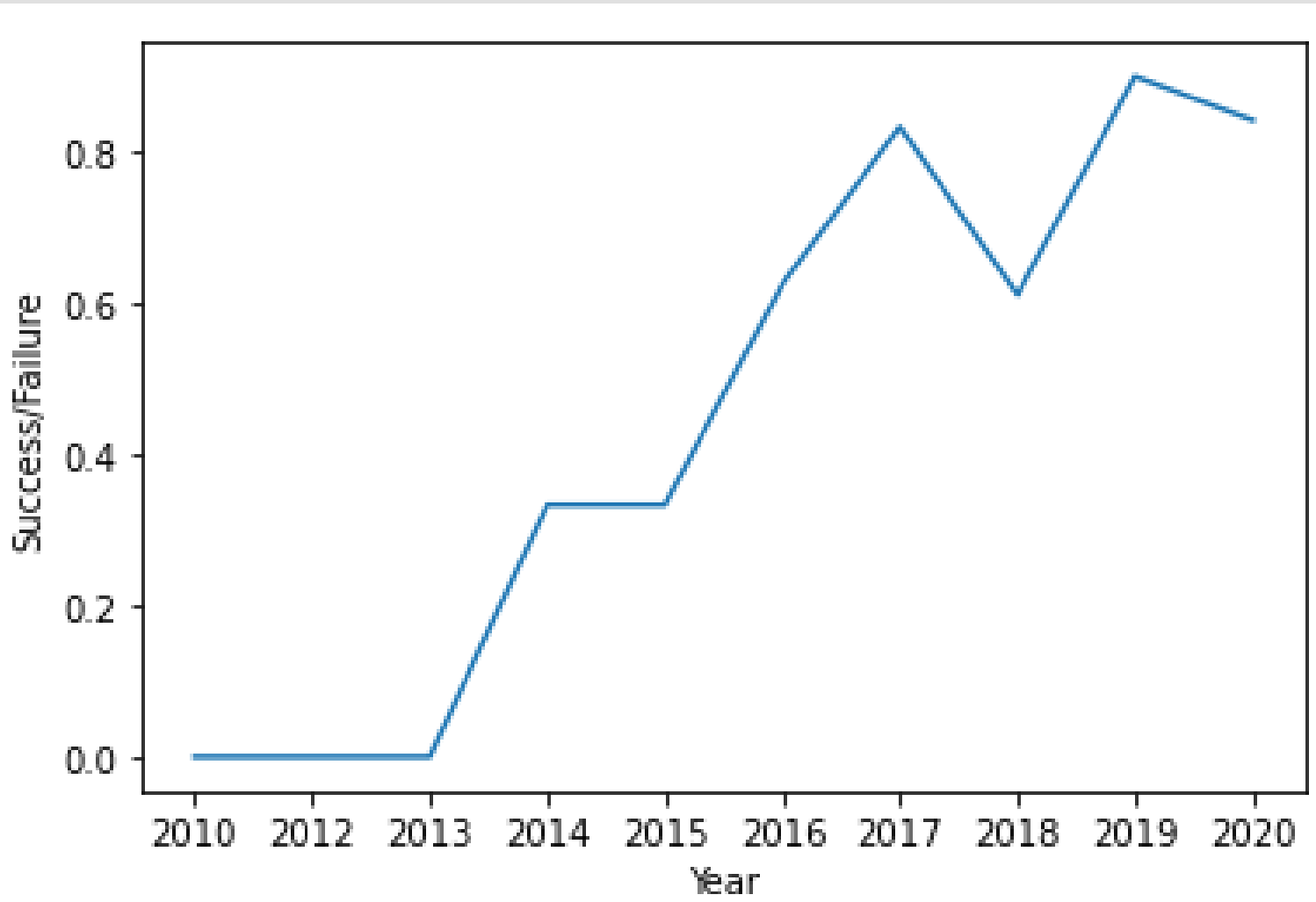
It was observed that more evenly flight number was launched on orbit GTO, PO, ISS, and LEO whereas VLEO had the highest Flight Number .

Payload vs. Orbit Type



My observation was that orbit VLEO had the highest mass being launched with ISS. The higher the mass the more likely the success of first stage landing.

Launch Success Yearly Trend



It was observed that the success rate since 2013 kept increasing till 2020

All Launch Site Names

Names of the unique launch sites:

%sql select Unique(LAUNCH_SITE) from SPACEXTBL;

CCAFS SLC 40', 'VAFB SLC 4E', 'KSC LC 39A,

launch_site
CCAFS LC-40
CCAFS SLC-40
CCAFSSLC-40
KSC LC-39A
VAFB SLC-4E

Launch Site Names Begin with 'CCA'

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

%sql SELECT LAUNCH_SITE from SPACEXTBL where (LAUNCH_SITE) LIKE 'CCA%' LIMIT 5;

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:

%sql select sum(PAYLOAD_MASS__KG_) as payloadmass from SPACEXTBL;

619967

Average Payload Mass by F9 v1.1

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

%sql select avg(PAYLOAD_MASS__KG_) as payloadmass from SPACEXTBL;

6138

First Successful Ground Landing Date

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

%sql select min(DATE) from SPACEXTBL;

1
2010-06-04

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 SPACEXTBL where LANDING__OUTCOME='Success (drone ship)' and PAYLOAD_MASS__KG_ BETWEEN;

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:

%sql select count(MISSION_OUTCOME) as missionoutcomes from SPACEXTBL GROUP BY MISSION_OUTCOME;

missionoutcomes
1
99
1

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass:
- `%sql select BOOSTER_VERSION as boosterversion from SPACEXTBL where PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_) from SP;`

Boosterversion

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

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

%sql SELECT MONTH(DATE),MISSION_OUTCOME,BOOSTER_VERSION,LAUNCH_SITE FROM SPACEXTBL where EXTRACT(YEAR FROM DATE)='2015';

1	mission_outcome	booster_version	launch_site
1	Success	F9 v1.1 B1012	CCAFS LC-40
2	Success	F9 v1.1 B1013	CCAFS LC-40
3	Success	F9 v1.1 B1014	CCAFS LC-40
4	Success	F9 v1.1 B1015	CCAFS LC-40
4	Success	F9 v1.1 B1016	CCAFS LC-40
6	Failure (in flight)	F9 v1.1 B1018	CCAFS LC-40
12	Success	F9 FT B1019	CCAFS LC-40

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

%sql SELECT LANDING__OUTCOME FROM SPACEXTBL WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' ORDER BY DATE DESC;

Landing__outcome
No attempt
Success (ground pad)
Success (drone ship)
Success (drone ship)
Success (ground pad)
Failure (drone ship)
Success (drone ship)
Success (drone ship)
Success (drone ship)
Failure (drone ship)
Failure (drone ship)
Success (ground pad)

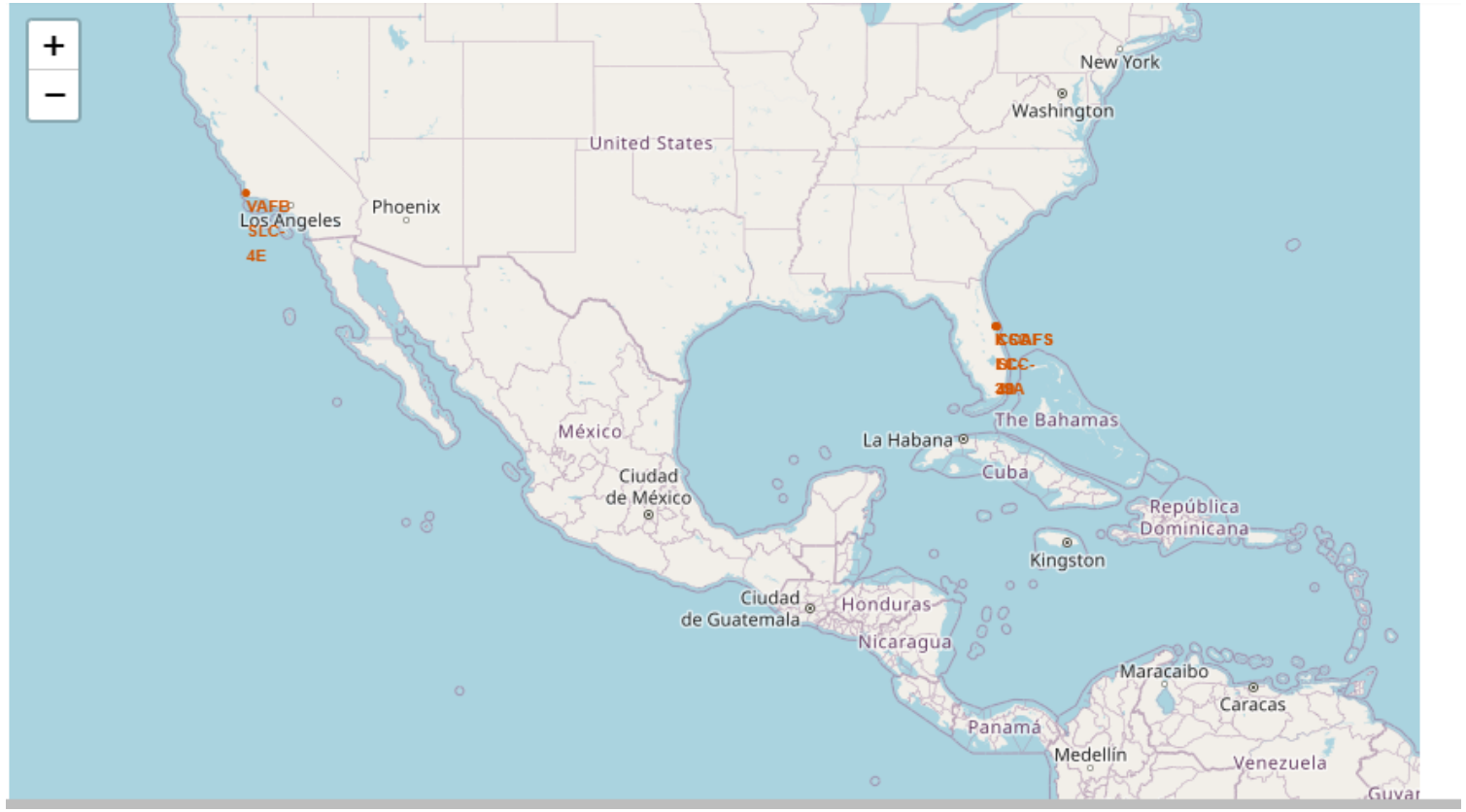
Precluded (drone ship)
No attempt
Failure (drone ship)
No attempt
Controlled (ocean)
Failure (drone ship)
Uncontrolled (ocean)
No attempt
No attempt
Controlled (ocean)
Controlled (ocean)
No attempt
No attempt

Uncontrolled (ocean)
No attempt
No attempt
No attempt
Failure (parachute)

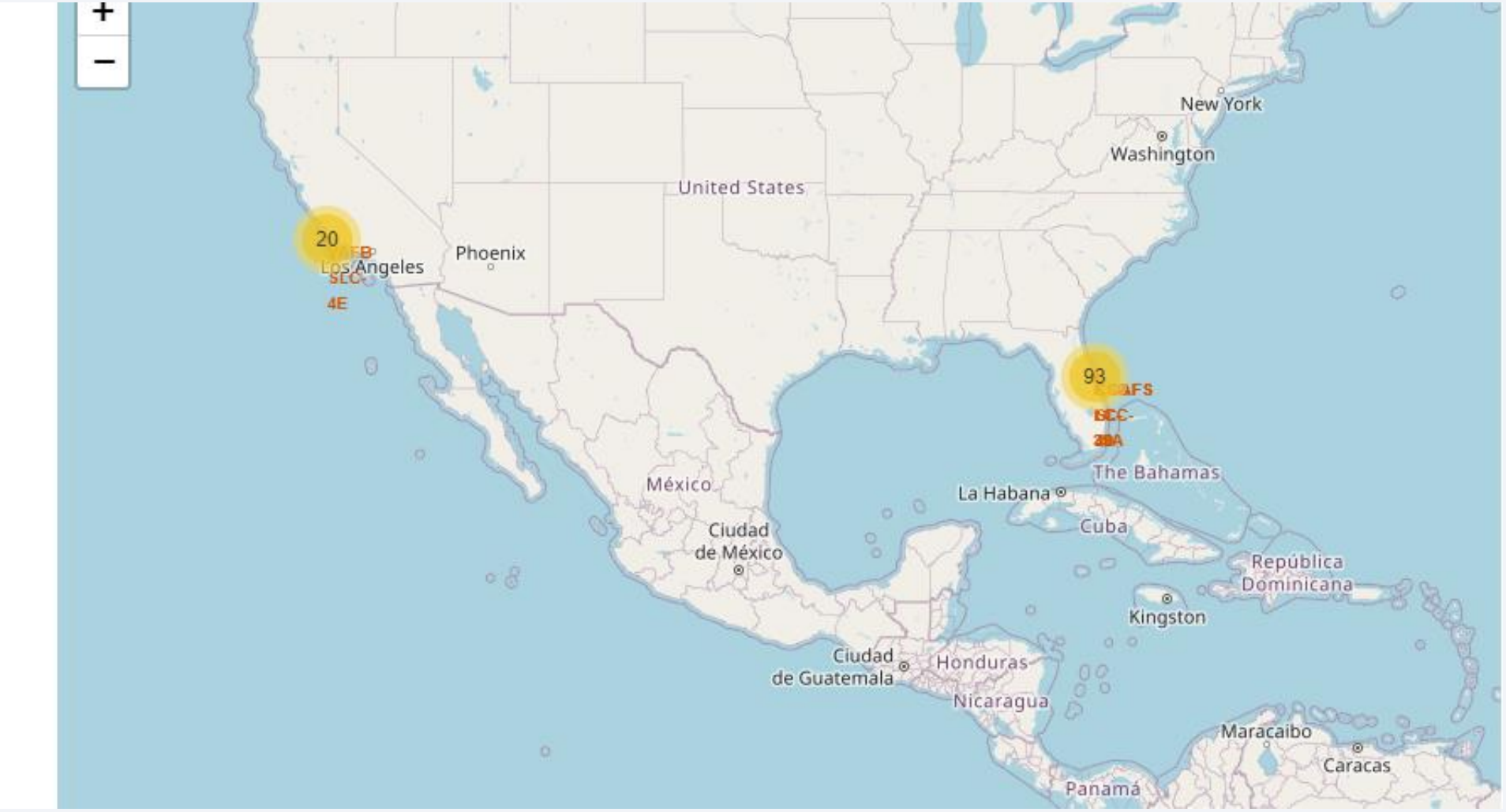

Section 3
Launch Sites
Proximities Analysis

Folium Map of all launch sites

- The marked elements are Launch Sites,

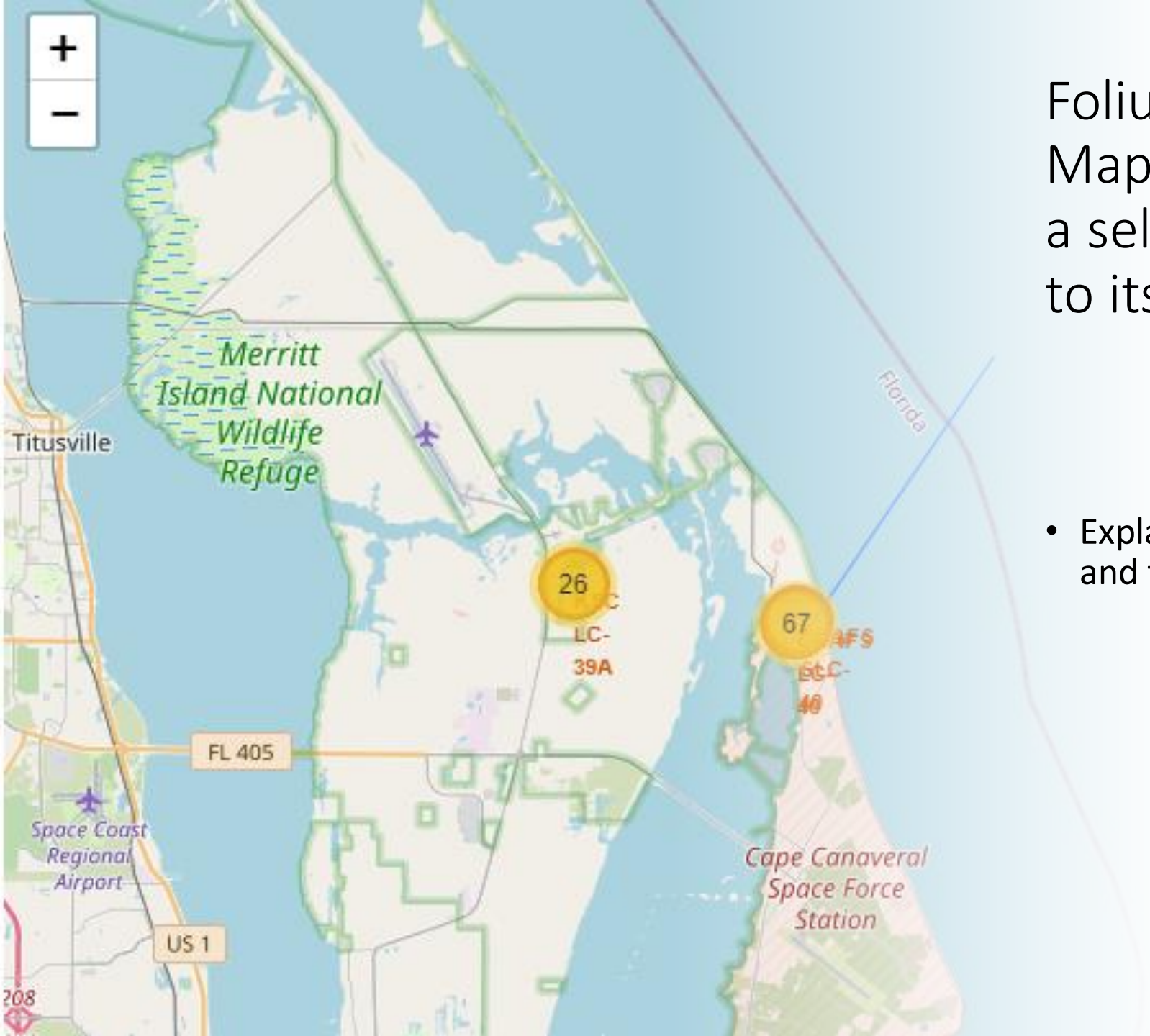


Folium Map color-labeled launch outcomes



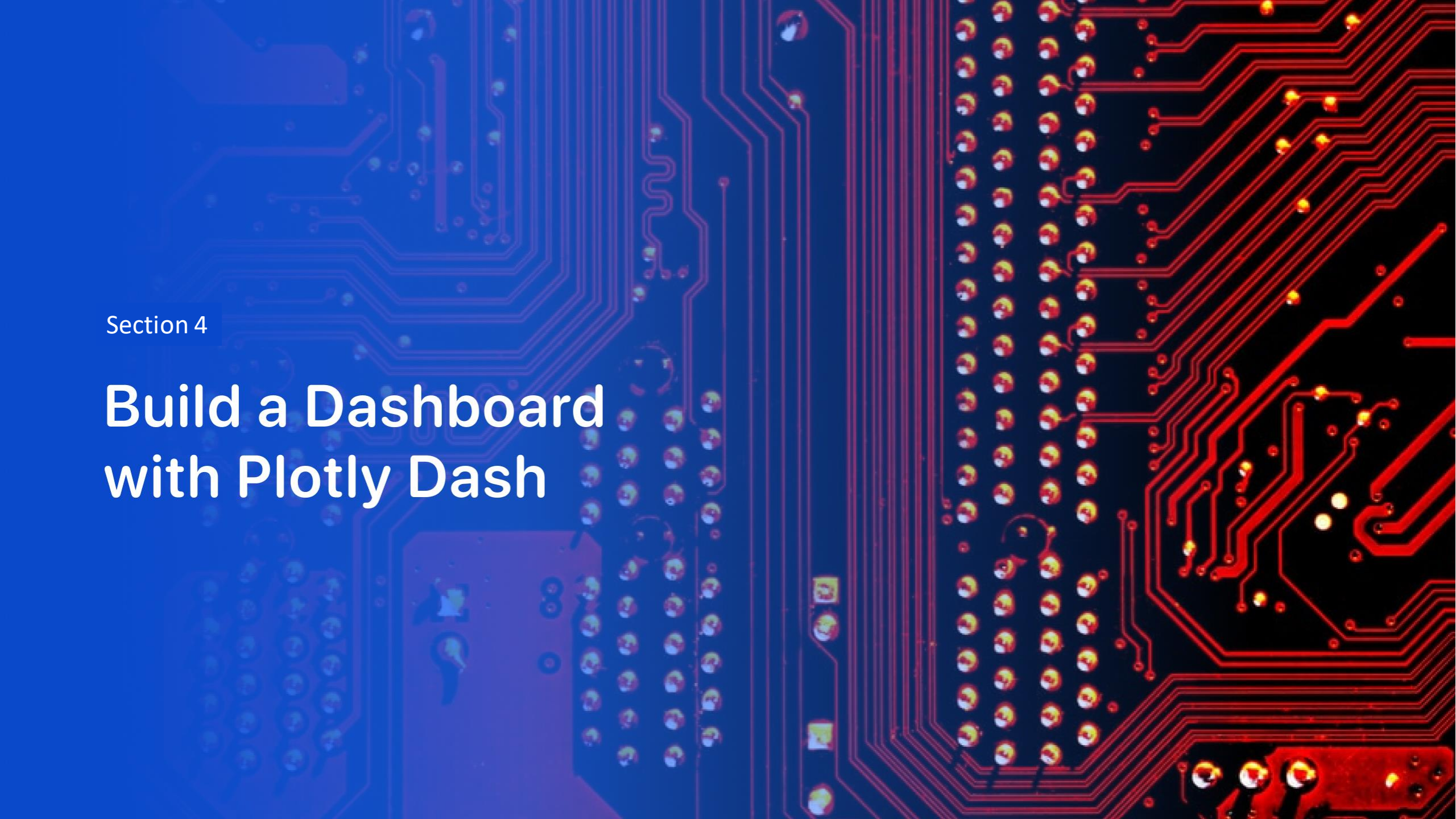
color-labeled launch outcomes on the map. When zoomed, the red means unsuccessful(class = 0), while the green means successful(class = 1)

[92]:



Folium
Map screenshot of a selected launch site to its proximities

- Explain the important elements and findings on the screenshot


Section 4
Build a Dashboard with Plotly Dash

<Dashboard Screenshot 1>

<Dashboard Screenshot 2>

<Dashboard Screenshot 3>

Section 5

Predictive Analysis (Classification)

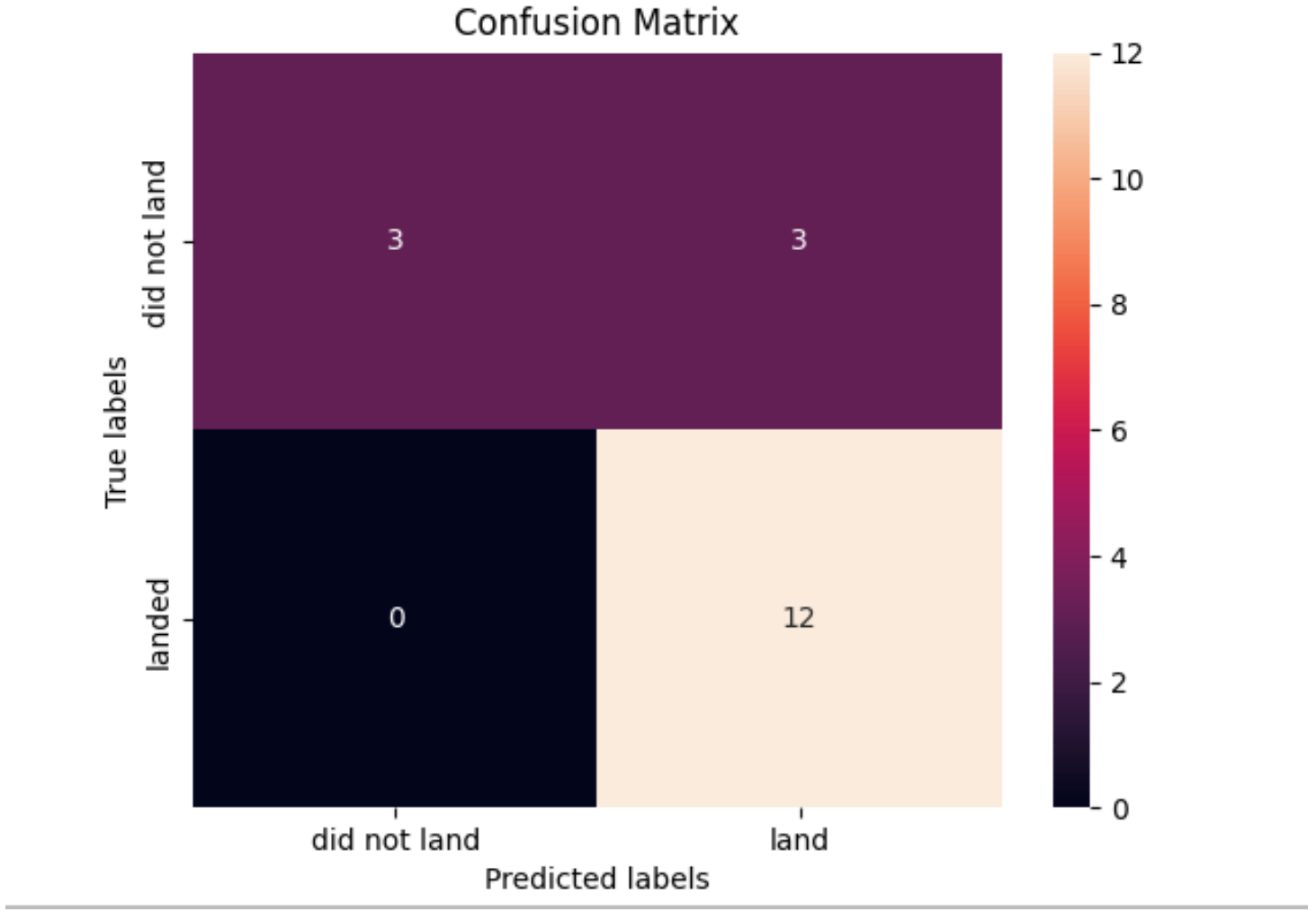
Classification Accuracy

- Visualize the built model accuracy for all built classification models, in a bar chart

- Find which model has the highest classification accuracy

Confusion Matrix

- The Confusion Matrix effectively shows that 12 landed successfully(positive_True but 3 did not land successfully Negative_false. Whereas none was positive_false and 3 were Negative_True



Conclusions

The Effective Prediction is that the next outcome of the Falcon 9 first stage will land successfully well

Appendix

- parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
- 'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
- 'p': [1,2]}
- KNN = KNeighborsClassifier()
- knn_cv = GridSearchCV(KNN, parameters, cv=10)
- knn_cv=knn_cv.fit(X_train,Y_train)
- knn_cv
- yhat = knn.predict(X_test)
- Yhat
- print("Train set Accuracy: ", metrics.accuracy_score(Y_train, knn.predict(X_train)))
- print("Test set Accuracy: ", metrics.accuracy_score(Y_test, yhat))
- print("tuned hpyerparameters :(best parameters) ",knn_cv.best_params_)
- print("accuracy :",knn_cv.best_score_)
- knn_cv.score(X_test,Y_test)
- print("Train set Accuracy: ", metrics.accuracy_score(Y_train, knn.predict(X_train)))
- print("Test set Accuracy: ", metrics.accuracy_score(Y_test, yhat))
- yhat = knn_cv.predict(X_test)
- plot_confusion_matrix(Y_test,yhat)

Thank you!