



IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Vasilios Philippou
10/4/2021



Outline

EXECUTIVE SUMMARY

INTRODUCTION

METHODOLOGY

RESULTS

CONCLUSION

Executive Summary

- Summary Of Methodologies

- Data Collection
- Data Wrangling
- EDA with Data Visualization
- EDA with SQL
- Interactive Map with Folium
- Predictive Analysis (Classification)

- Summary Of All Results

- Exploratory Data Analysis Results
- Interactive Analytics Demo in Screenshots
- Predictive Analytics Results

INTRODUCTION

- Project Background and Context

We predicted 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, when other providers cost upward of 165 millions dollars each, much of the saving is because SpaceX can reuse the first stage. Therefore, if we can determine if the first stage will land, we can determine the cost of the launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch

- Common problems that needed solving

What influences if the rocket will land successfully

The effect each relationship with certain rocket variables will impact in determining the success rate of a successful landing

What conditions does SpaceX have to achieve to get the best results and ensure the best rocket success landing rate.

Methodology

- Data Collection Methodology:

- SpaceX Rest API

- Web Scraping

- Performed Data Wrangling (Transforming Data for ML)
 - One Hot Encoding data fields for Machine Learning and dropping irrelevant columns
 - Performed exploratory Data Analysis (EDA) using visualization and SQL
 - Plotting: Scatter Graphs, Bar Graphs to show relationships between variables and show patterns of data
 - Performed interactive visual analytics using Folium and Plotly Dash
 - Performed predictive analysis using classification models
 - How to build, tune and evaluate classification models

Data Collection

- The following Data Set was collected by:
 - We worked with SpaceX launch data that is gathered from the SpaceX Rest API.
 - This API will give us data about launches, including information about the rocket used, payload delivered, launch specifications, landing specifications, and landing outcome
 - Our goal is to use this data to predict whether SpaceX will attempt to land a rocket or not
 - The SpaceX REST API endpoints, or URL, starts with `api.spacexdata.com/v4/`.
 - Another popular data source for the obtaining Falcon 9 Launch data is web scraping Wikipedia using BeautifulSoup

Data Collection- SpaceX API

<https://github.com/VasiliosPhilippou/IBM-Data-Science/blob/main/Caps%20IBM%20Project.ipynb>

1) Getting Response from API

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:

```
In [70]: static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

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

```
In [71]: response.status_code
```

```
Out[71]: 200
```

2) Apply custom function to clean data/ Assign list to dictionary then dataframe

```
In [77]: # Call getBoosterVersion  
getBoosterVersion(data)
```

the list has now been update

```
In [78]: BoosterVersion[0:5]
```

```
Out[78]: ['Falcon 1', 'Falcon 1', 'Falcon 1', 'Falcon 1', 'Falcon 9']
```

we can apply the rest of the functions here:

```
In [79]: # Call getLaunchSite  
getLaunchSite(data)
```

```
In [80]: # Call getPayloadData  
getPayloadData(data)
```

```
In [81]: # Call getCoreData  
getCoreData(data)
```

Finally lets construct our dataset using the data we have obtained. We we combine the columns into a dictionary.

```
In [82]: 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}
```


3) Filter Data Frame and export to flat file (.csv)

Data Collection- SpaceX API

<https://github.com/VasiliosPhilippou/IBM-Data-Science/blob/main/Caps%20IBM%20Project.ipynb>

4) Dealing with missing values

Task 2: Filter the dataframe to only include Falcon 9 launches

Finally we will remove the Falcon 1 launches keeping only the Falcon 9 launches. Filter the data dataframe using the BoosterVersion column to only keep the Falcon 9 launches. Save the filtered data to a new dataframe called data_falcon9.

```
In [93]: # Hint data['BoosterVersion']!='Falcon 1'
data_falcon9=df.loc[df['BoosterVersion']!='Falcon 1']
data_falcon9['BoosterVersion'].value_counts()
```

```
Out[93]: Falcon 9    90
Name: BoosterVersion, dtype: int64
```

Now that we have removed some values we should reset the FlightNumber column

```
In [86]: data_falcon9.loc[:, 'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

Task 3: Dealing with Missing Values

Calculate below the mean for the PayloadMass using the .mean(). Then use the mean and the .replace() function to replace np.nan values in the data with the mean you calculated.

```
In [88]: # Calculate the mean value of PayloadMass column
data_falcon9['PayloadMass'].mean()

# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'].replace(np.nan, data_falcon9['PayloadMass'].mean())
```

```
Out[88]: 4      6123.547647
5      525.000000
6      677.000000
7      500.000000
8      3170.000000
...
89     15600.000000
90     15600.000000
91     15600.000000
92     15600.000000
93     3681.000000
Name: PayloadMass, Length: 90, dtype: float64
```

You should see the number of missing values of the PayloadMass change to zero.

```
In [89]: data_falcon9.isnull().sum()
```

```
Out[89]: FlightNumber    0
Date                  0
BoosterVersion         0
PayloadMass           5
Orbit                  0
LaunchSite             0
Outcome                0
Flights                0
GridFins               0
Reused                 0
Legs                   0
LandingPad            26
Block                 0
```


Data Collection- Web Scrapping

<https://github.com/VasiliosPhilippou/IBM-Data-Science/blob/main/CAP%20IBM%20-%20WEB%20SCRABLING.ipynb>

1) Request the Falcon9 Launch Wiki page from its URL

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.

```
In [48]: # use requests.get() method with the provided static_url
response=requests.get(static_url)

# assign the response to a object
```

Create a BeautifulSoup object from the HTML response

```
In [61]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup=BeautifulSoup(response.text,'html.parser')
```

Print the page title to verify if the BeautifulSoup object was created properly

```
In [62]: # Use soup.title attribute
soup.find_all('title')
```

```
Out[62]: [<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>]
```

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

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

```
In [16]: # 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')
```

Starting from the third table is our target table contains the actual launch records.

```
In [17]: # Let's print the third table and check its content  
first_launch_table = html_tables[2]  
print(first_launch_table)
```

Data Collection- Web Scrapping

<https://github.com/VasiliosPhilippou/IBM-Data-Science/blob/main/CAP%20IBM%20-%20WEB%20SCRABLING.ipynb>

3) Create a data frame parsing the launch HTML tables

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

```
In [41]: launch_dict= dict.fromkeys(column_names)

# Remove an irrelevant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

6. Appending data to keys (refer) to notebook block 12

```
In [12]: extracted_row = 0
#Extract each table
for table_number,table in enumerate(
    # get table row
    for rows in table.find_all("tr")
    #check to see if first table
```

7. Converting dictionary to dataframe

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

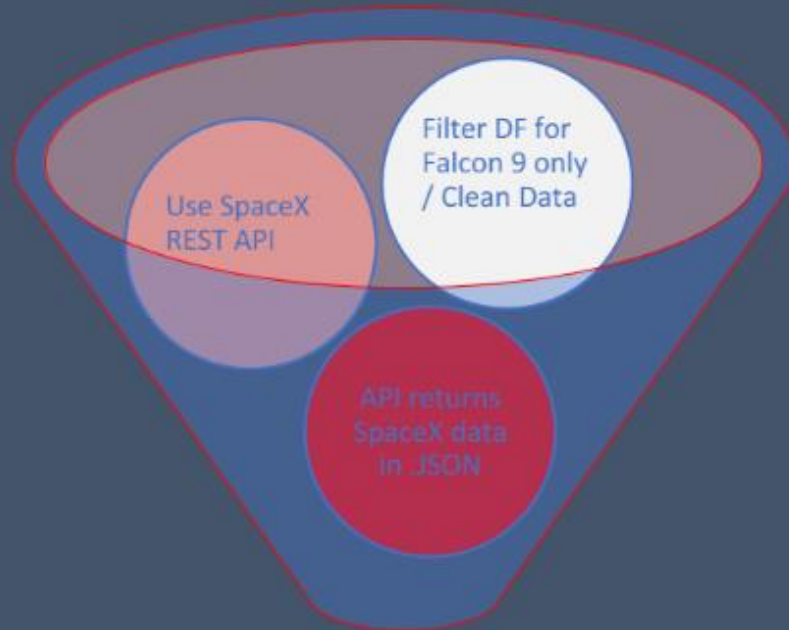
8. Dataframe to .CSV

```
df.to_csv('spacex_web_scraped.csv', index=False)
```

Data Collection- Web Scrapping

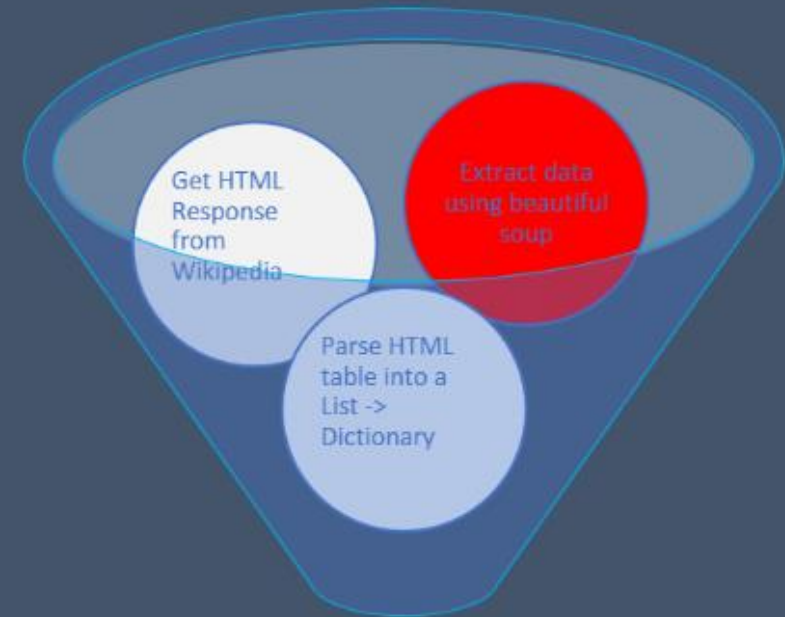
<https://github.com/VasiliosPhilippou/IBM-Data-Science/blob/main/CAP%20IBM%20-%20WEB%20SCRABLING.ipynb>

Data collection – SpaceX API



Normalize data into flat data
file such as .csv

Data collection – Web Scrapping



Normalize data into flat data
file such as .csv

Data Wrangling

- In the data set, there are several different cases where the booster did not land successfully. Sometimes a landing was attempted but failed due to an accident; for example, True Ocean means the mission outcome was successfully landed to specific region of the ocean while False Ocean means the mission outcome was unsuccessfully landed to a specific region of the ocean. True RTLS means the mission outcome was successfully landed to a ground pad, False RTLS means the mission outcome was unsuccessfully landed to a ground pad. True ASDS means the mission outcome was successfully landed on a drone ship and False ASDS means the mission outcome was unsuccessfully landed on a drone ship.
- We mainly convert those outcomes into Training Labels with 1 means the booster successfully landed and 0 means it was unsuccessful.

Data Wrangling

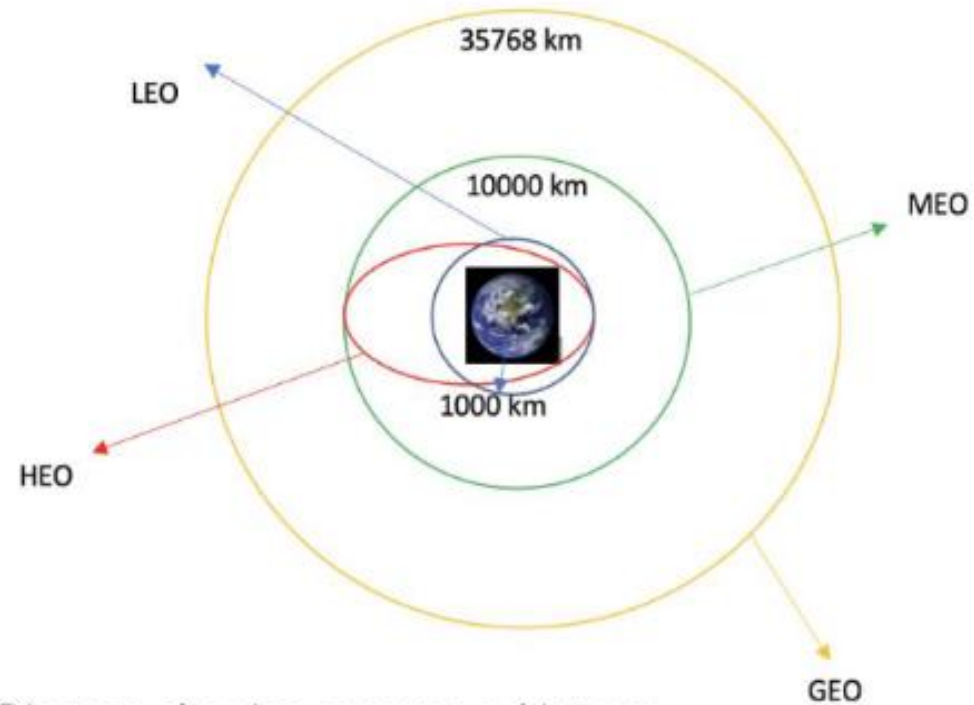
<https://github.com/VasiliosPhilipou/IBM-Data-Science/blob/main/CAP%20IBM%20-%20DATA%20WRANGLING%20EDA.ipynb>

- Phases :
 - Perform Exploratory Data Analysis EDA on dataset
 - Calculate number of launches at each site
 - Calculate number and occurrence of each orbit
 - Calculate the number/occurrence of mission outcome per orbit type
 - Export data set as .CSV
 - Create a landing outcome label from Outcome column
 - Work out success rate for every landing dataset

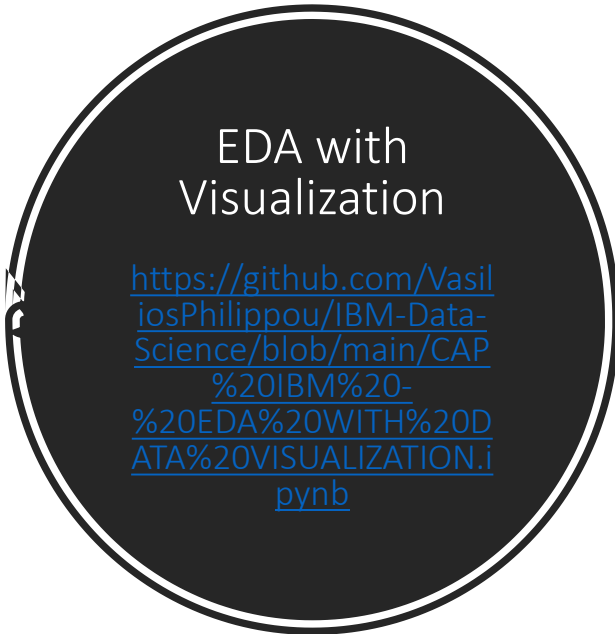
Data Wrangling

<https://github.com/VasiliosPhilippou/IBM-Data-Science/blob/main/CAP%20IBM%20-%20DATA%20WRANGLING%20EDA.ipynb>

Each launch aims to an dedicated orbit, and here are some common orbit types:



*Diagram showing common orbit types
SpaceX uses*



EDA with Visualization

<https://github.com/VasiliosPhilippou/IBM-Data-Science/blob/main/CAP%20IBM%20-%20EDA%20WITH%20DATA%20VISUALIZATION.ipynb>

Scatter Graphs :

- Flight Number VS. Payload Mass
- Flight Number VS. Launch Site
- Payload VS. Launch Site
- Orbit VS. Flight Number
- Payload VS. Orbit Type
- Orbit VS. Payload MASS

Scatter plots show how much one variable is affected by another. The relationship between two variables is called correlation. Scatterplots usually consist of a large body of data, but they don't explain causation between the two variables

EDA with Visualization

<https://github.com/VasiliosPhilippou/IBM-Data-Science/blob/main/CAP%20IBM%20-%20EDA%20WITH%20DATA%20VISUALIZATION.ipynb>

- Bar Graphs :

Mean VS. Orbit

A bar diagram makes it easy to compare sets of data between different groups at a glance. The graph represents categories on one axis and a discrete value in the other. The goal is to show the relationship between the two axes. Bar charts can also show big changes in data over time

- Line Graph :

Success Rate VS. Year

Line graphs are useful in that they show data variables and trends very clearly and can help to make predictions about the results of data not yet recorded

EDA with SQL

<https://github.com/VasiliosPhilippou/IBM-Data-Science/blob/main/CAP%20IBM%20-%20EDA%20WITH%20SQL.ipynb>

Performed SQL queries to gather information about the dataset.

For example of some questions we were asked about the data we needed information about. Which we are using SQL queries to get the answers in the dataset :

- Displaying the names of the unique launch sites in the space mission
- Displaying 5 records where launch sites begin with the string 'KSC'
- 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 where 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 records which will display the month names, successful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017
- Ranking the count of successful landing_outcomes between the date 2010-06-04 and 2017-03-20 in descending order.

Building an interactive Map with Folium

[https://github.com/VasiliosPhilippou/IBM-Data-Science/blob/main/CAP%20IBM%20-%20DASHBOARDS%20\(1\).ipynb](https://github.com/VasiliosPhilippou/IBM-Data-Science/blob/main/CAP%20IBM%20-%20DASHBOARDS%20(1).ipynb)

- **To visualize the Launch Data into an inter active map.** We took the Latitude and Longitude Coordinates at each launch site and added a Circle Marker around each launch site with a label of the name of the launch site.
- **We assigned the dataframe launch_outcomes(failures, successes) to classes 0 and 1** with Green and Red markers on the map in a MarkerCluster()
- **Using Haversine's formula we calculated the distance** from the Launch Site to various landmarks to find various trends about what is around the Launch Site to measure patterns. Lines are drawn on the map to measure distance to landmarks
- **Example of some trends in which the Launch Site is situated in.**
 - 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.

Predictive Analysis with Classification

<https://github.com/VasiliosPhilippou/IBM-Data-Science/blob/main/CAP%20IBM%20-%20CLASSIFICATION.ipynb>

• Building Model

- Loading our dataset into NumPy and Pandas
- Transform Data
- Split our data into training and test data
- Check how many test samples we have
- Decide which type of machine learning algorithms we want to use
- Set our parameters and algorithms to GridSearchCV
- Fit our datasets into the GridSearch objects and train our data

Evaluating Model

- Check accuracy for each model
- Get tuned hyperparameters for each type of algorithm
- Plot Confusion Matrix

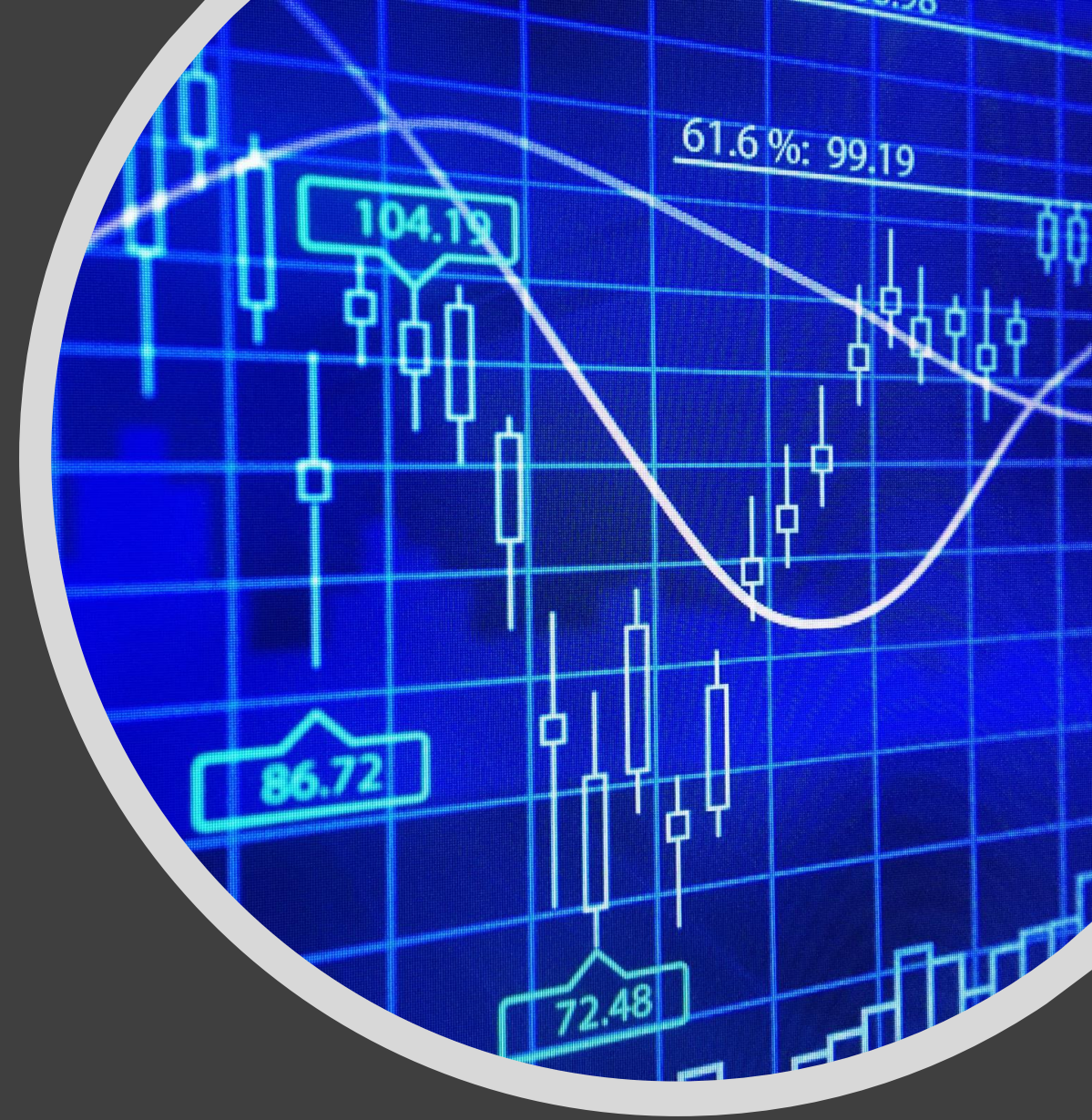
Improving Model

- Feature Engineering
- Algorithm Tuning

Find the best predicting Classification Model

Results

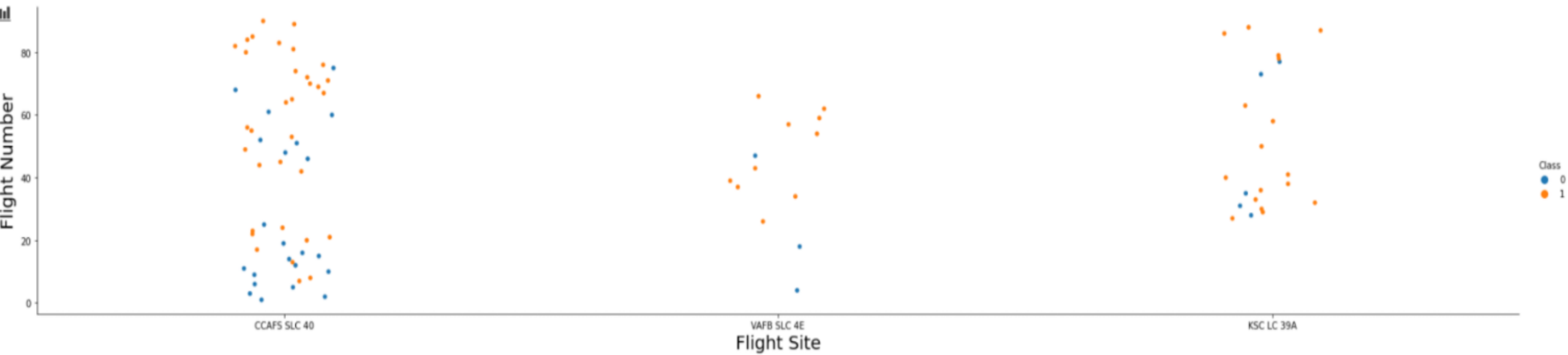
- Exploratory Data Analysis
- Interactive Analysis Demo in Screenshots
- Predictive Analysis Results





EDA WITH VISUALIZATION

Flight Number vs. Flight Site



The more amount of flights at a launch site the greater the success rate at a launch site.

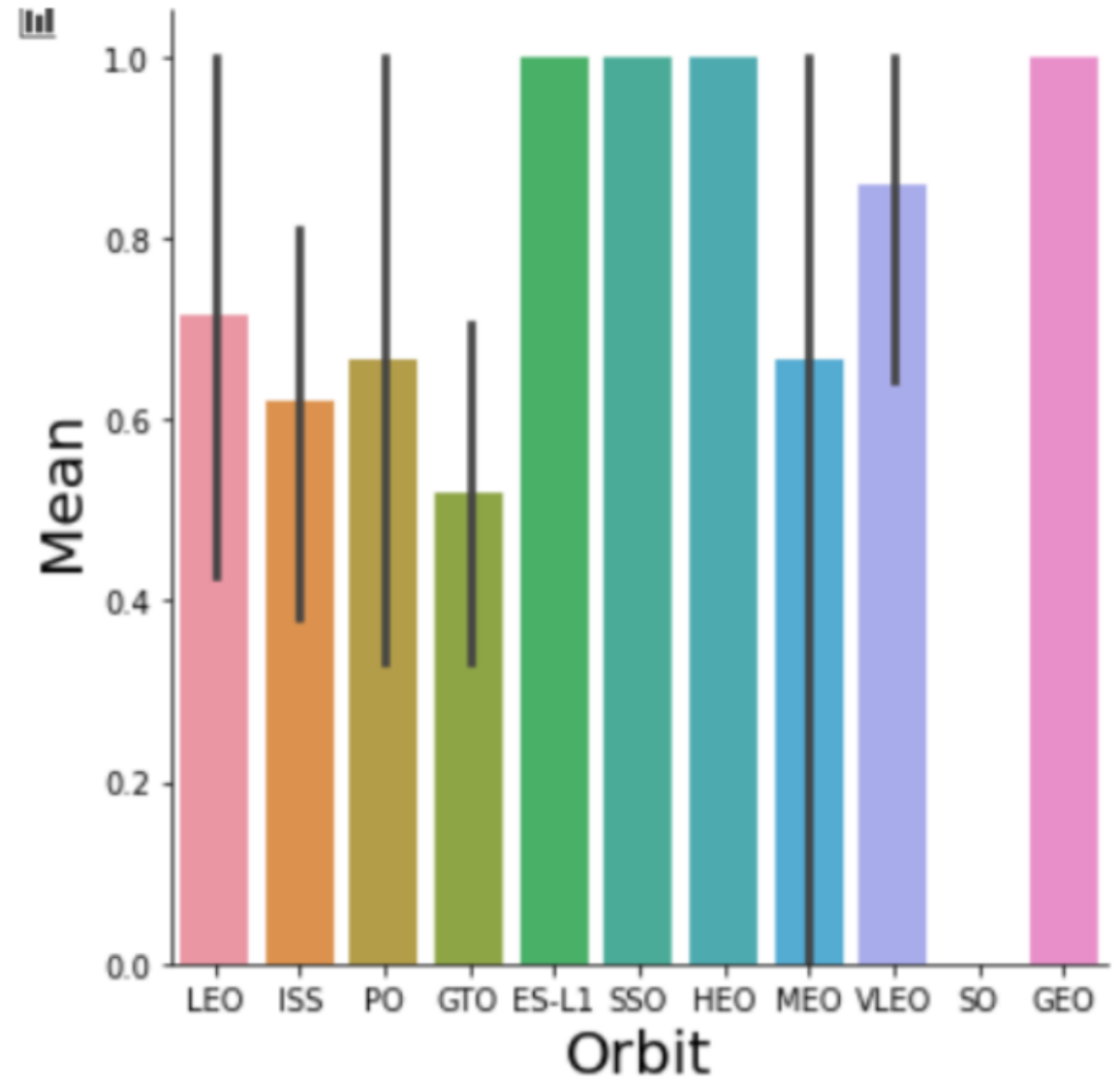
Payload Mass vs. Launch Site



The greater the payload mass for Launch Site CCAFS SLC 40 the higher the success rate for the Rocket. There is not quite a clear pattern to be found using this visualization to make a decision if the Launch Site is dependant on Pay Load Mass for a success launch.

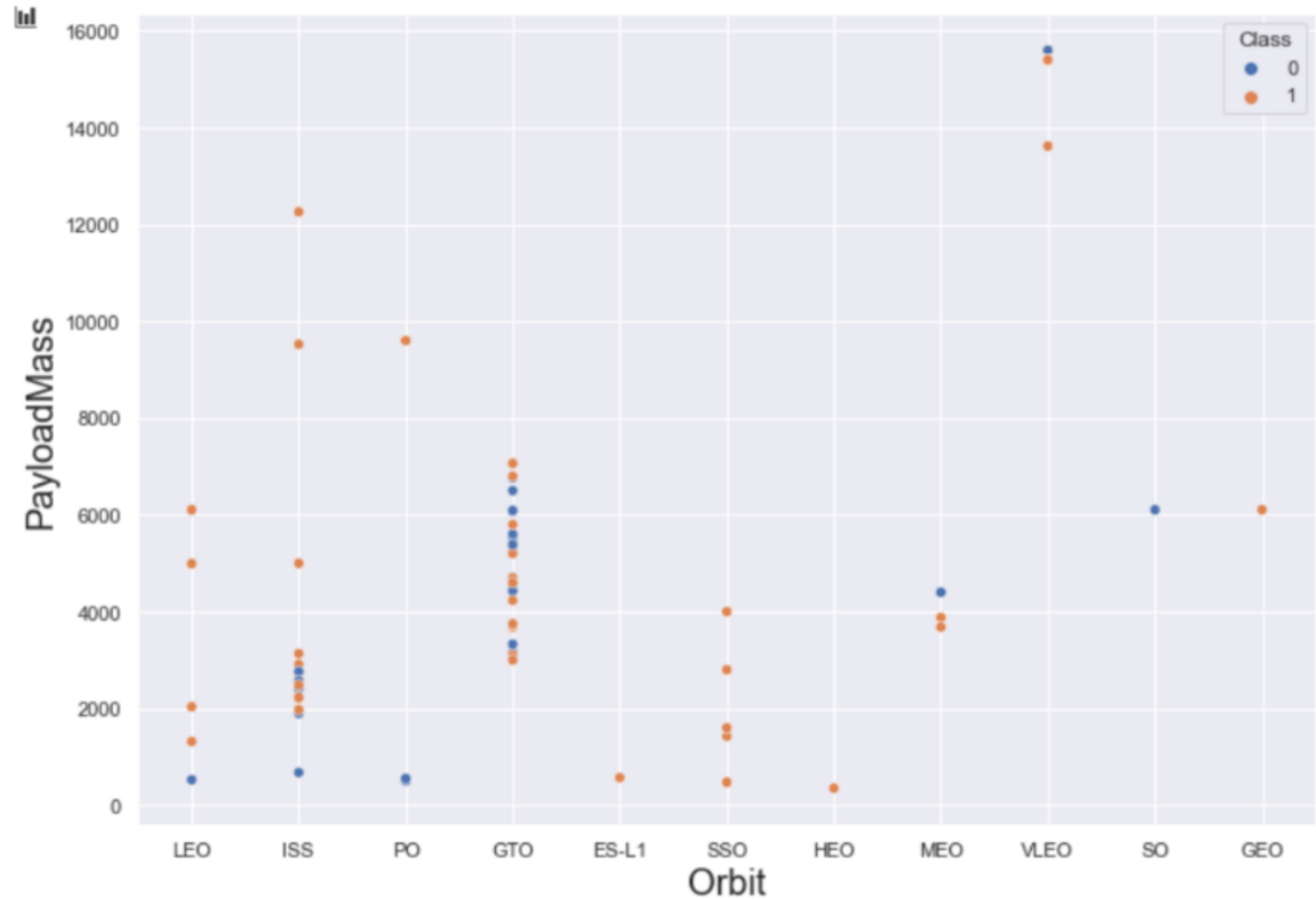
Success rate vs. Orbit type

Orbit GEO,HEO,SSO,ES-L1 has the best Success Rate



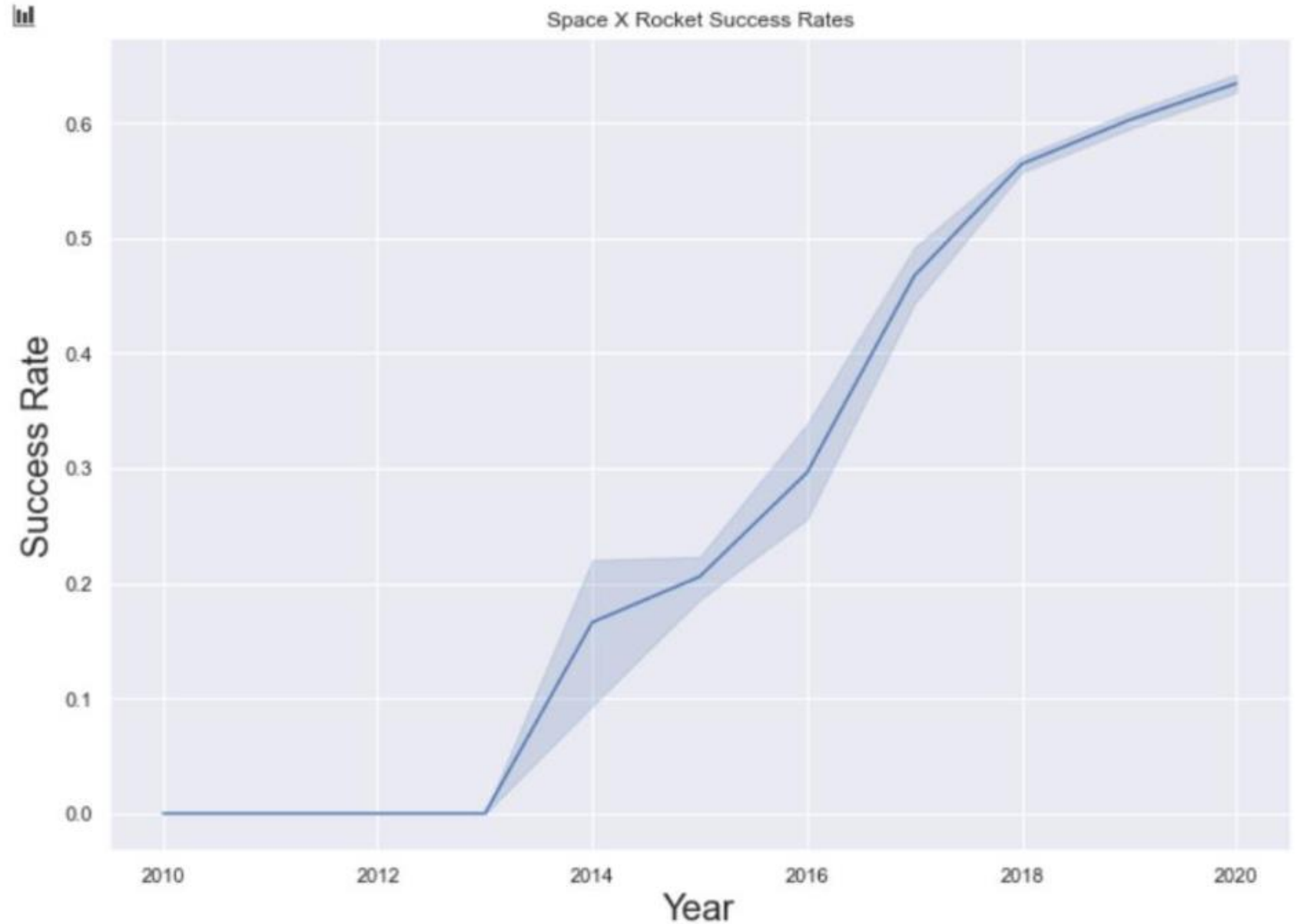
Payload vs. Orbit type

You should observe that Heavy payloads have a negative influence on GTO orbits and positive on GTO and Polar LEO (ISS) orbits.



Launch success yearly trend

you can observe that the success rate since 2013 kept increasing till 2020





EDA WITH SQL

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

Task 1

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

In [19]: %sql SELECT DISTINCT LAUNCH_SITE FROM XXQ88963.SPACEXTBL

* ibm_db_sa://xxq88963:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/BLUDB
Done.

Out[19]:

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

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

Task 2

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

```
In [41]: %sql SELECT * FROM XXQ88963.SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5
```

```
* ibm_db_sa://xxq88963:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/BLUDB
Done.
```

```
Out[41]:
```

DATE	time__utc_	booster_version	launch_site	payload_mass_kg	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	None	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	None	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	None	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40	None	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	None	677	LEO (ISS)	NASA (CRS)	Success	No attempt

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

Task 3

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

In [42]: %sql select SUM(PAYLOAD_MASS__KG_) from XXQ88963.SPACEXTBL where Customer = 'NASA (CRS)'

* ibm_db_sa://xxq88963:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/BLUDB
Done.

Out[42]:

1
45596

Display average payload mass carried by booster version F9 V1.1

Task 4

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

```
In [43]: %sql select AVG(PAYLOAD_MASS__KG_) from XXQ88963.SPACEXTBL where BOOSTER_VERSION = 'F9 v1.1'
```

```
* ibm_db_sa://xxq88963:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/BLUDB
Done.
```

```
Out[43]:
```

1
2928

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

Task 5

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

Hint: Use min function

```
In [45]: %sql select MIN(Date) from XXQ88963.SPACEXTBL where LANDING__OUTCOME = 'Success'
```

```
* ibm_db_sa://xxq88963:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/BLUDB  
Done.
```

```
Out[45]:
```

1
2018-07-22

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

Task 6

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

```
In [48]: %sql select BOOSTER_VERSION from XXQ88963.SPACEXTBL where (LANDING__OUTCOME = 'Success') AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000
* ibm_db_sa://xxq88963:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/BLUDB
Done.
```

```
Out[48]:
```

booster_version
F9 B5 B1046.2
F9 B5 B1047.2
F9 B5 B1046.3
F9 B5 B1048.3
F9 B5 B1051.2
F9 B5B1060.1
F9 B5 B1058.2
F9 B5B1062.1

List the total number of successful and failure mission outcomes

Task 7

List the total number of successful and failure mission outcomes

```
In [57]: %sql SELECT Count(MISSION_OUTCOME) from XXQ88963.SPACEXTBL where MISSION_OUTCOME = 'Success'
```

```
* ibm_db_sa://xxq88963:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/BLUDB  
Done.
```

Out[57]:

1
99

```
In [58]: %sql SELECT Count(MISSION_OUTCOME) from XXQ88963.SPACEXTBL where MISSION_OUTCOME = 'Failure'
```

```
* ibm_db_sa://xxq88963:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/BLUDB  
Done.
```

Out[58]:

1
0

List the names of the booster_version which have carried the maximum payloadmass. Use Subquery

Task 8

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

```
In [65]: %sql SELECT DISTINCT BOOSTER_VERSION FROM XXQ88963.SPACEXTBL WHERE PAYLOAD_MASS__KG_ = ( SELECT MAX(PAYLOAD_MASS__KG_) FROM XXQ88963.SPACEXTBL)

* ibm_db_sa://xxq88963:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/BLUDB
Done.
```

Out[65]:

booster_version
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

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

Task 9

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

In [80]: %sql SELECT LANDING__OUTCOME, BOOSTER_VERSION, LAUNCH_SITE FROM XXQ88963.SPACEXTBL WHERE LANDING__OUTCOME = 'Failure (drone ship)' AND DATE LIKE '2015%'

* ibm_db_sa://xxq88963:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/BLUDB
Done.

Out[80]:

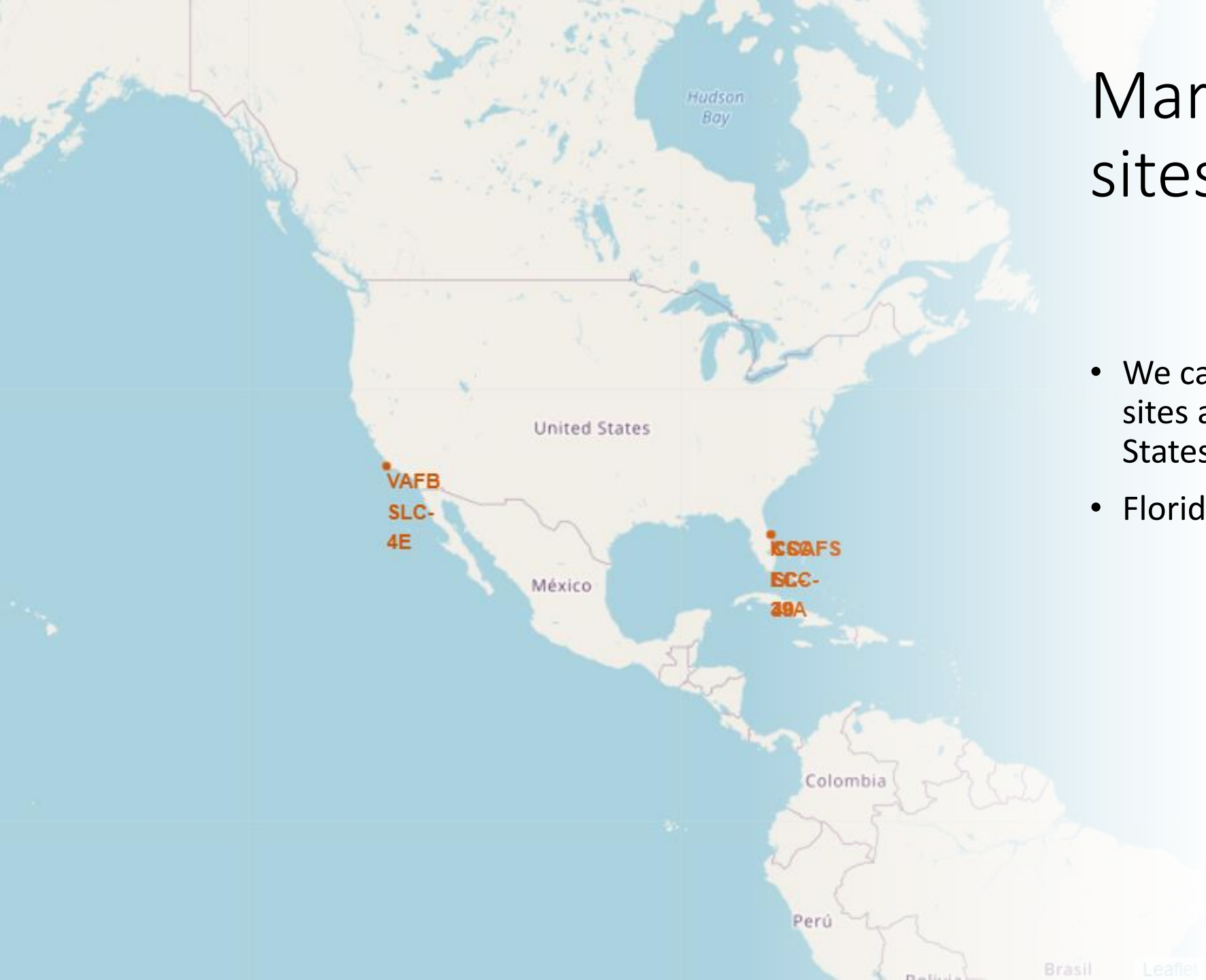
landing__outcome	booster_version	launch_site
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

The background is a blurred image of a map. A pen is visible in the upper right corner, pointing towards a specific location on the map. The map shows various geographical features and some numerical data, such as '2.5' and '2,47'.

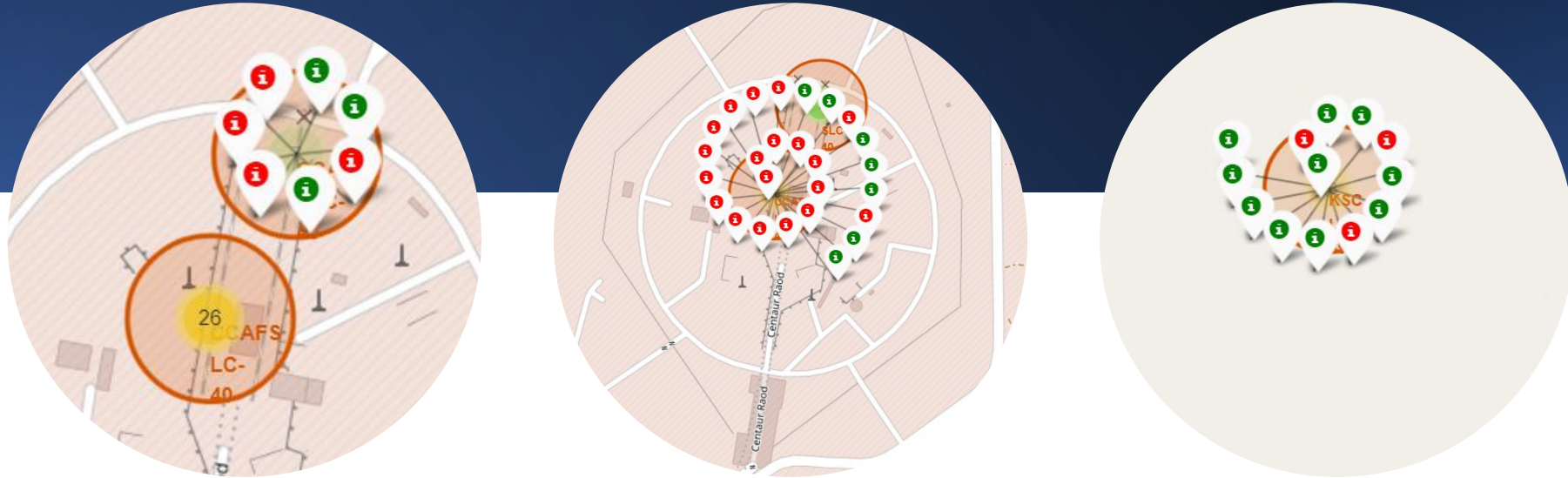
Interactive Maps with Folium

Mark all launch sites on the map

- We can see that SpaceX launch sites are both in the United States of America coasts.
- Florida and California

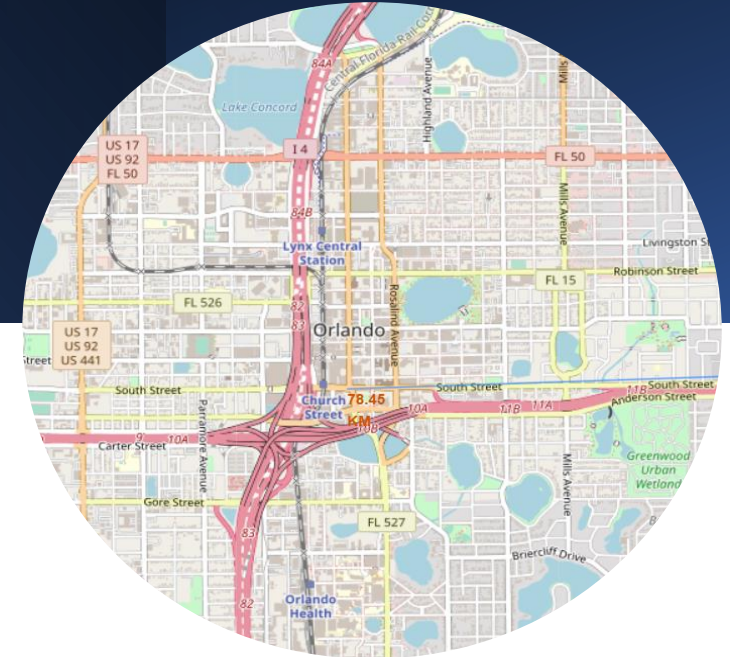
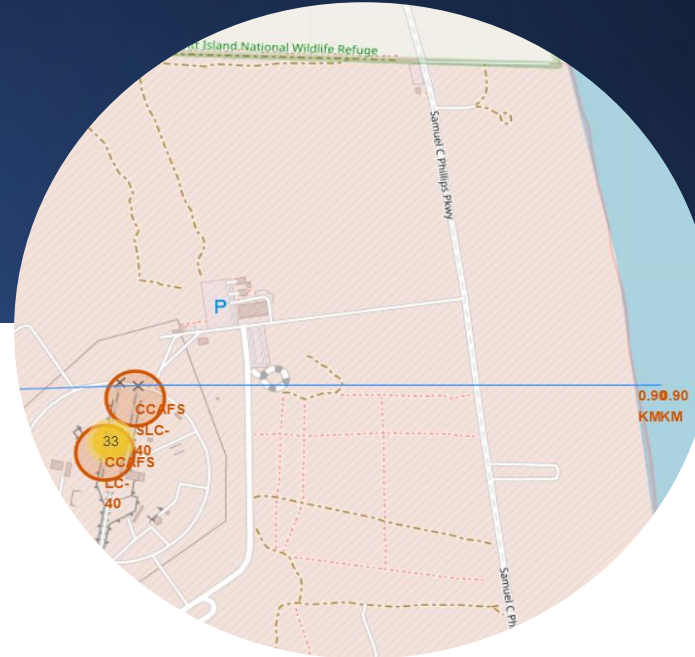


Mark the success/failed launches for each site on the map



- Green Markers show successful launches and Red Markers unsuccessful launches
- The 2 launching points on the left belong to Florida district
- The 1 on the right to California district

Calculate the distances between a launch site to its proximities



- 1st Distance to Railway Station
- 2nd Distance to Coast
- 3rd Distance to the closest Highway



Predictive Analytics Classification

TASK 12

Find the method performs best:

```
In [35]: 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 0.8875
Best Params is : {'criterion': 'gini', 'max_depth': 8, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 5, 'splitter': 'random'}
```

Classification
Accuracy using
test data

- Among the models that were tested, the tree classifier gives the best prediction accuracy (0,8875)
- This accuracy is a result of the selection of the best hyperparameters for the tree classifier, as you will see more analytically in the next page

Tree Classifier Execution

```
In [25]: parameters = {'criterion': ['gini', 'entropy'],
                        'splitter': ['best', 'random'],
                        'max_depth': [2*n for n in range(1,10)],
                        'max_features': ['auto', 'sqrt'],
                        'min_samples_leaf': [1, 2, 4],
                        'min_samples_split': [2, 5, 10]}
```

```
tree = DecisionTreeClassifier()
```

```
In [26]: gscv = GridSearchCV(tree,parameters,scoring='accuracy',cv=10)
tree_cv = gscv.fit(X_train,Y_train)
```

```
In [27]: print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
print("accuracy :",tree_cv.best_score_)

tuned hpyerparameters :(best parameters) {'criterion': 'gini', 'max_depth': 10,
'min_samples_split': 5, 'splitter': 'random'}
accuracy : 0.8875
```

- Examining the Confusion Matrix, we can see that the Tree can distinguish between different classes.
- We see that the major problem is false positives

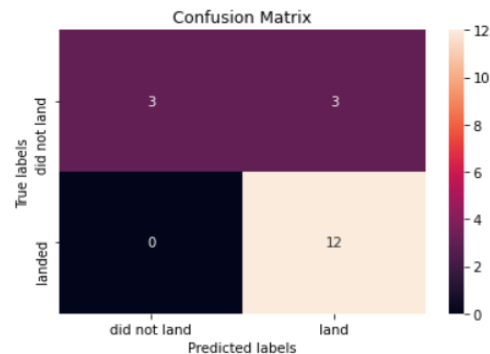
Calculate the accuracy of tree_cv on the test data using the method score:

```
In [28]: print("accuracy: ",tree_cv.score(X_test,Y_test))
```

accuracy: 0.8333333333333334

We can plot the confusion matrix

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



Conclusion

- The Machine Learning algorithm that fits best for the test data is the Tree Classifier
- Low weighted payloads perform better than higher payloads
- We can see that KSC LC-39A had the most successful launches
- We can predict if the first stage will land given the transformed and processed data we have, with accuracy of the prediction to be right 88.75 %
- We have to further examine if that percentage has statical significance

Thank you!

