IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Salvo Bazzano>
<26.04.24>

Executive Summary

- Summary of methodologies: We have performed a Data Analysis (data Cleaning, Data Wrangling and data Visualization) and Predictive Analysis using Machine Learning Model . The data analysis and the Predictive model show us which range of Launch parameters, as Payload range, Orbit and Site location permit to maximize the positive success of the launch

Outline

- Introduction
- Methodology
- Results
- Conclusion
- Appendix

Executive Summary

- Summary of methodologies
- Summary of all results

Introduction

- Project background and context

In the private business of the Space service, the company SpaceX can offer the best offer, at the least, in term of saving cost.
The main reason of this competitiveness is due to the reuse of the rocket used in the first stage. In particular, any mission consists in two stages:
First stage: The rocket launches (the most expensive between stage)
Second stage: The bring the payload in the orbit
The fact that the rockets can be landed successfuly and reuse again is a cost saving.

- Problems you want to find answers

Our target is to study the data from the company SpaceX and define the best predictive model in order to establish a new competitive Company named SpaceY

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology: The data has been collected using two different methods: Rest API and WEB Scraping
- Perform data wrangling: The data has been, normalized, filtered (selected the one useful for our analysis, and cleaned (i.e. replacing the null value with avg value);
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models: after the data has been standardized, we used GridSearch object to define the best hyperparameter to use for each models and to verify the accuracy of them

Data Collection

- Describe how data sets were collected.

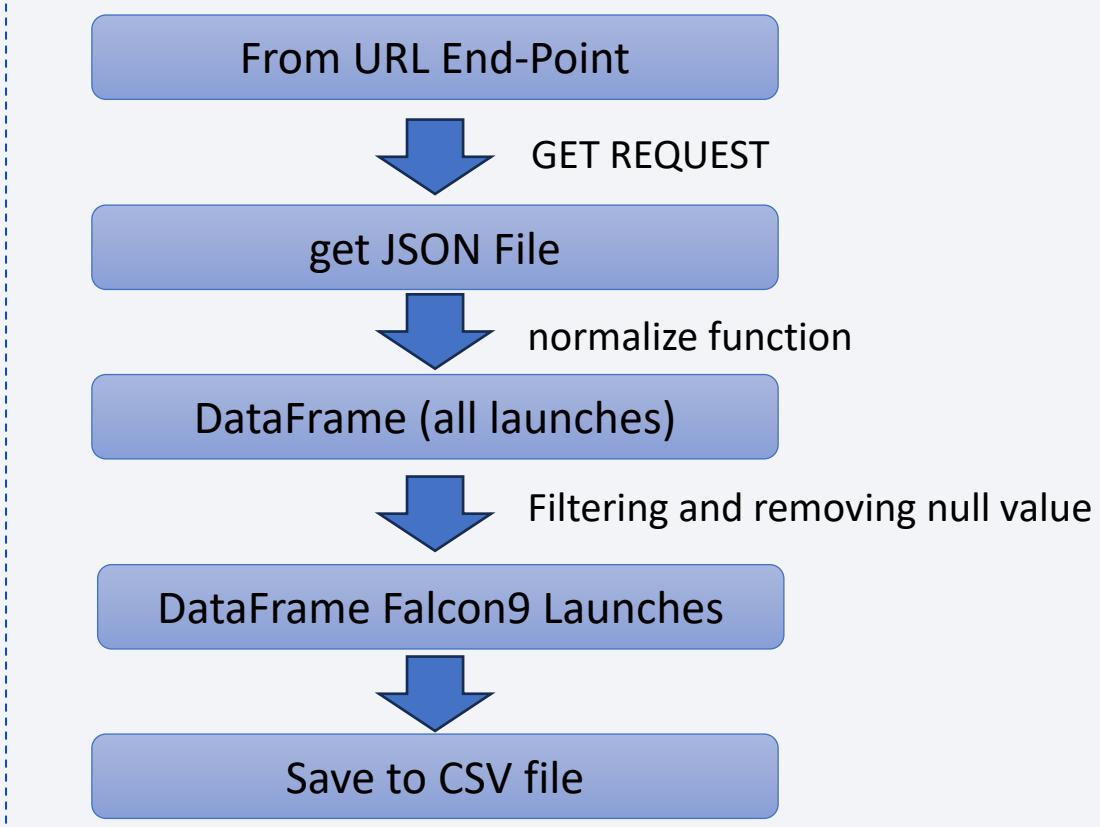
Using RESP API: we extracted the data from the spaceX Rest Papi in the form of JSON file and transform (normalize) this daa in order to get a DataFrame. This Dataframe has been cleaned filtering the data referred to the Launch Falcon9 and removed the null data then saved in CSV file

Using Web Scraping: we scraped Wiki pages (HTML table) containing data of Falcon9 launches then will parse the related table and convert in Dataframe and export it in CSV file

Data Collection – SpaceX API

- Present your data collection with SpaceX REST calls using key phrases and flowcharts

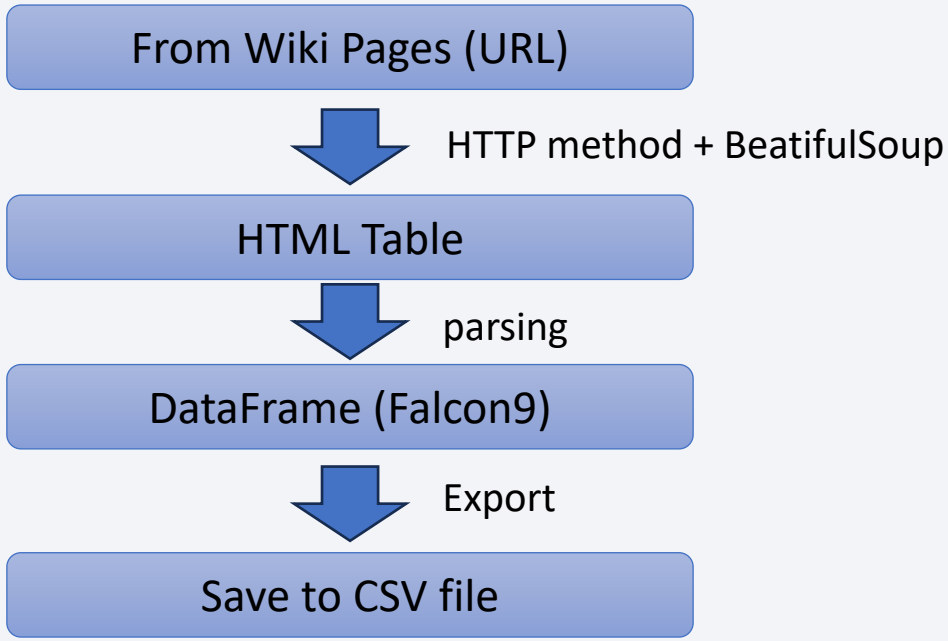
GITHub Link: lab_1_spacex-data-collection-api.ipynb



Data Collection - Scraping

- Present your web scraping process using key phrases and flowcharts

- GITHub Link
  lab_2_webscraping.ipynb

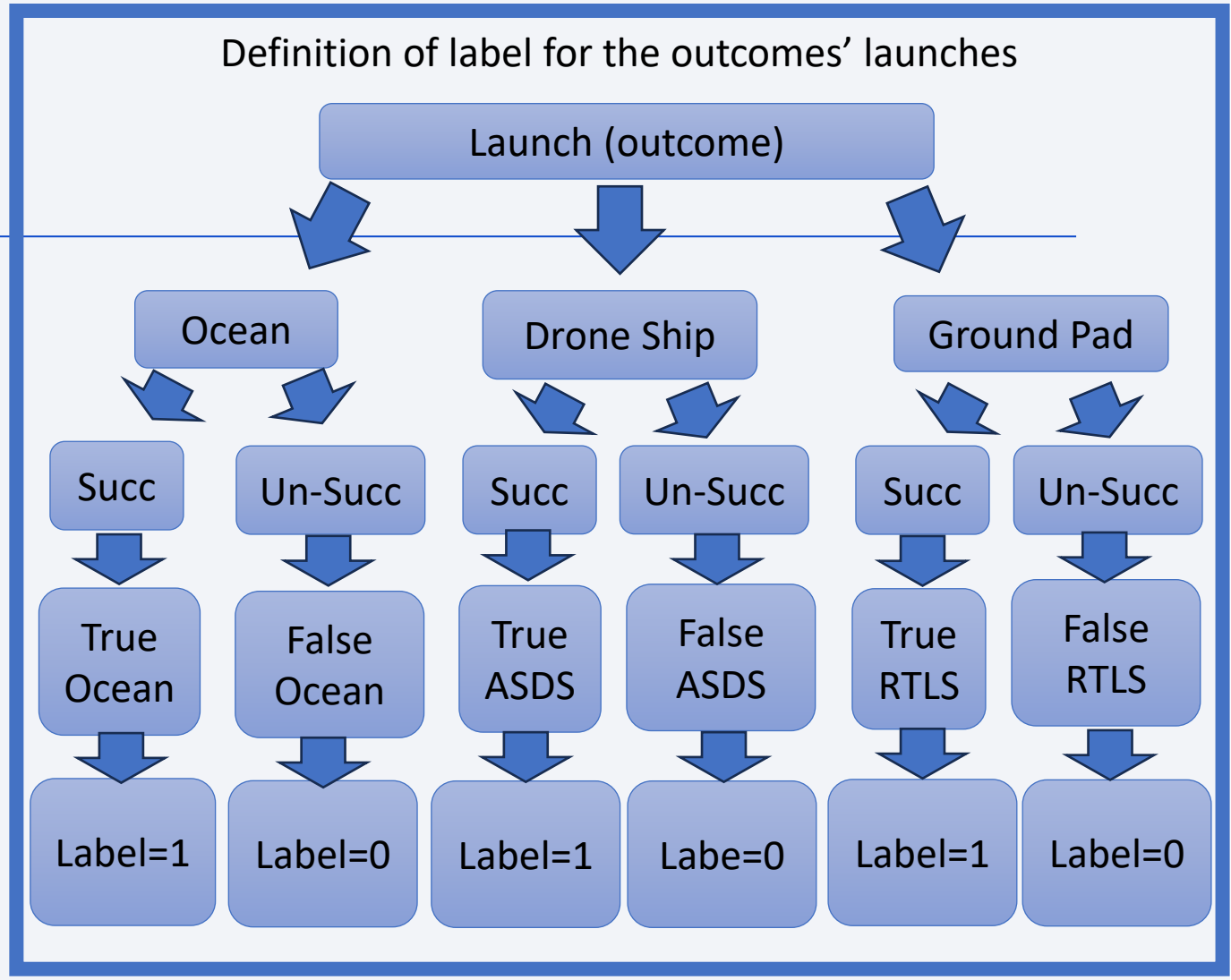


Data Wrangling

We have performed some Exploratory Data Analysis (EDA) to find some patterns in the data and determine what would be the label for training supervised models. In particular for the cases where the booster did not land successfully will assign a Training Labels the value 1, while for usuccessfully landed will assign value 0. For doing it we have to wrangle the data, determ all possible outcomes and assigns the related classes

Data Wrangling process

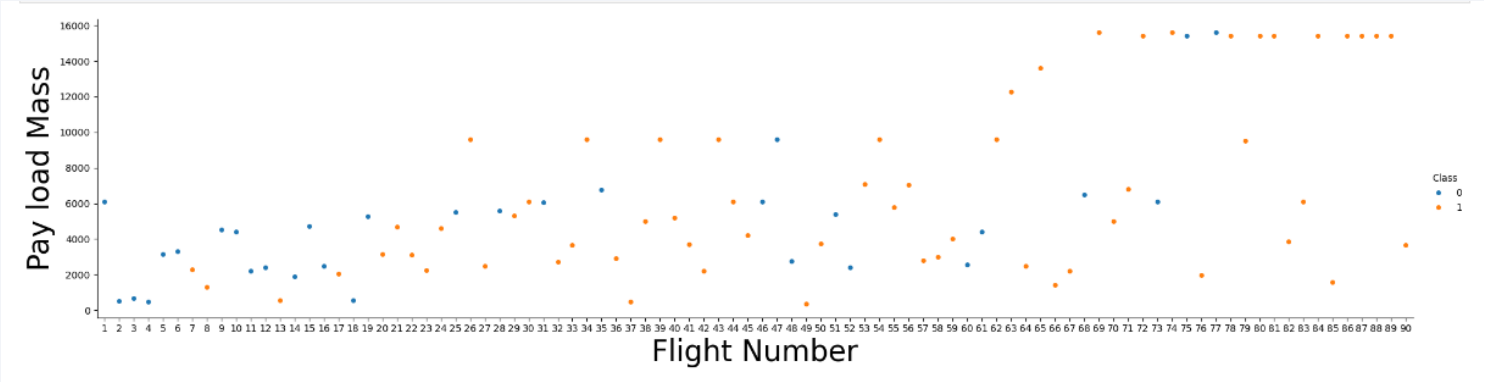
- Calculate n° launches on each Site
- Calculate n° occurance per Orbit
- Calculate n° occurance of mission of the Orbits
- Create and assign outcome label for each outcome



[GITHub Link
lab_3_data_wrangling.ipynb]

EDA with Data Visualization

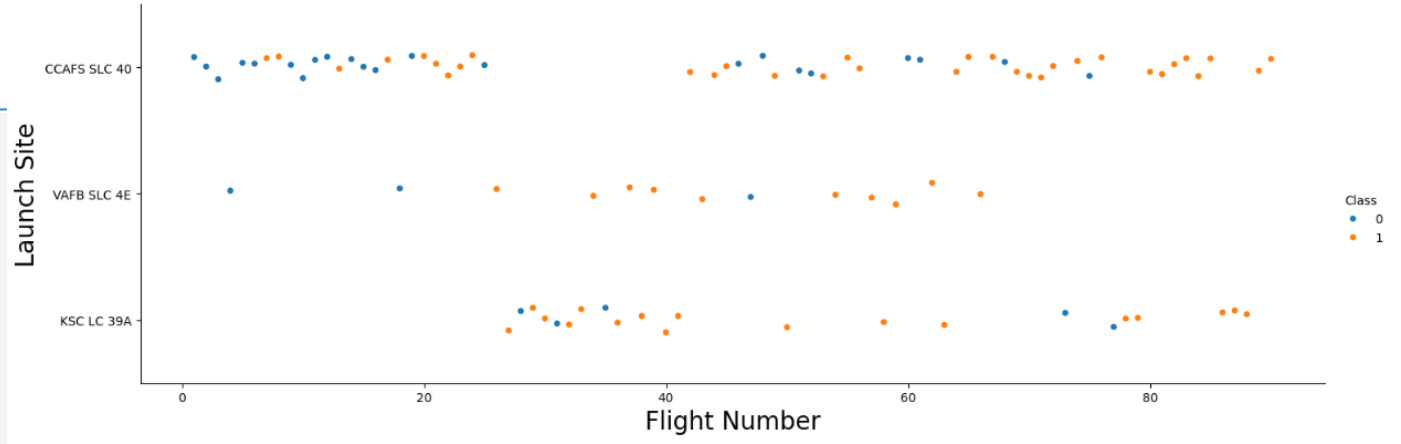
as the flight number increases, the first stage is more likely to land successfully. The payload mass is also important; it seems the more massive the payload, the less likely the first stage will return.



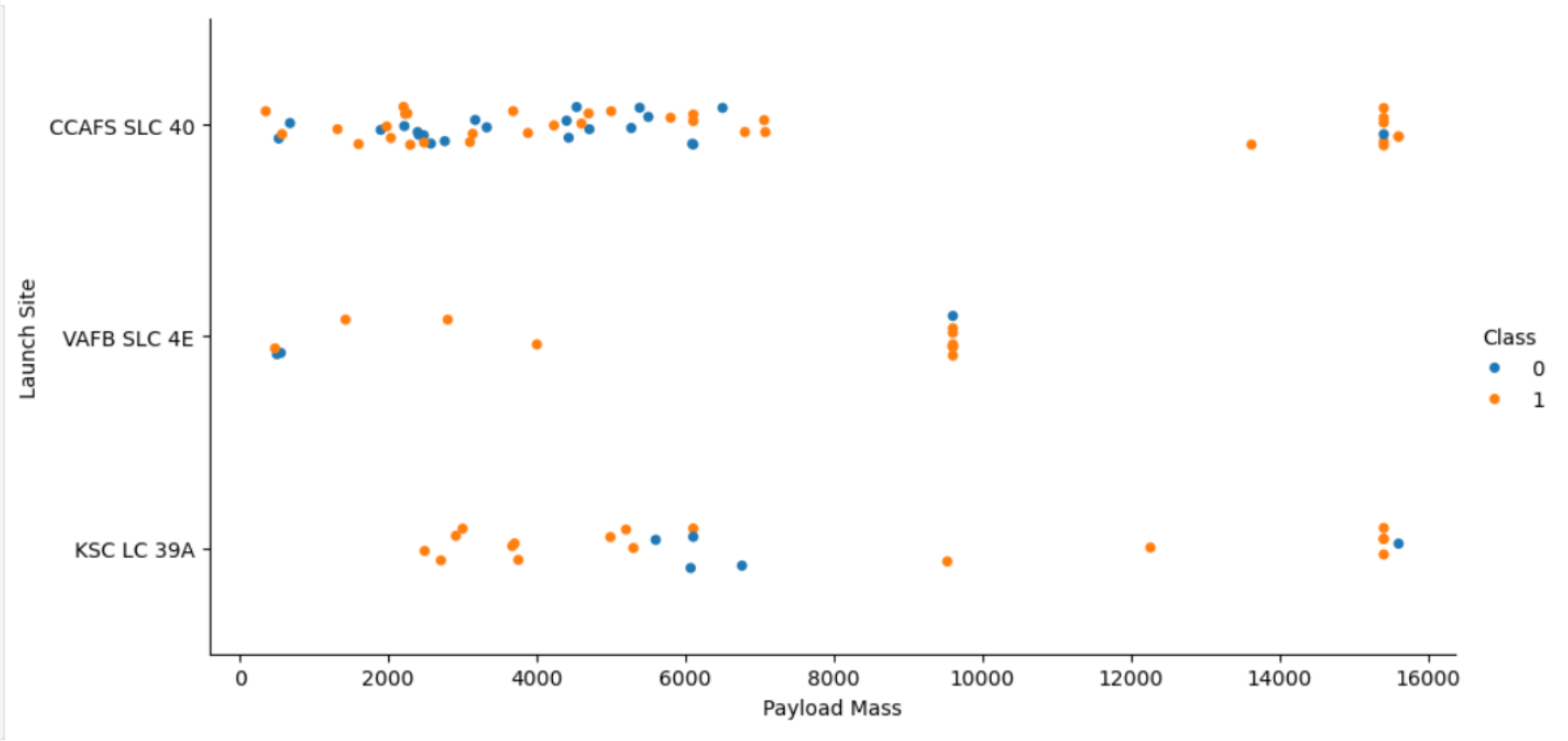
We see that different launch sites have different success rates. `CCAFS LC-40`, has a success rate of 60 %, while `KSC LC-39A` and `VAFB SLC 4E` has a success rate of 77%.

```
print(f"CCAFS success rate: {count_CCAFS['Class'].value_counts()[1]/count_CCAFS.shape[0]:.2%}")
print(f"KSC success rate: {count_KSC['Class'].value_counts()[1]/count_KSC.shape[0]:.2%}")
print(f"VAFB success rate: {count_VAFB['Class'].value_counts()[1]/count_VAFB.shape[0]:.2%}")
```

```
CCAFS success rate: 60.00%
KSC success rate: 77.27%
VAFB success rate: 76.92%
```

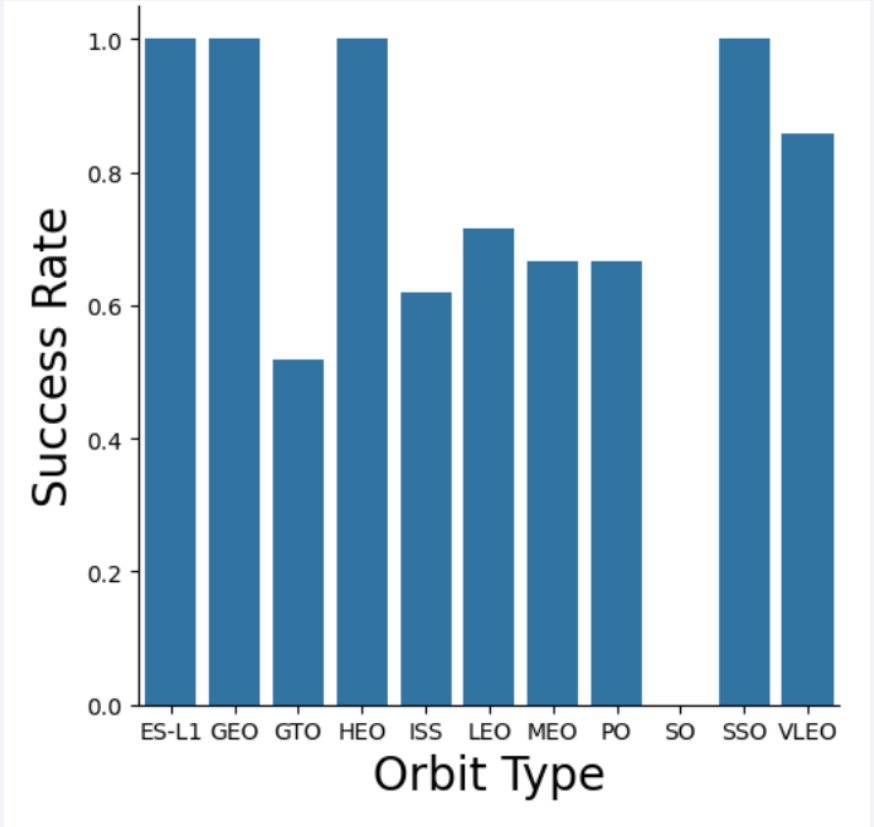


EDA with Data Visualization

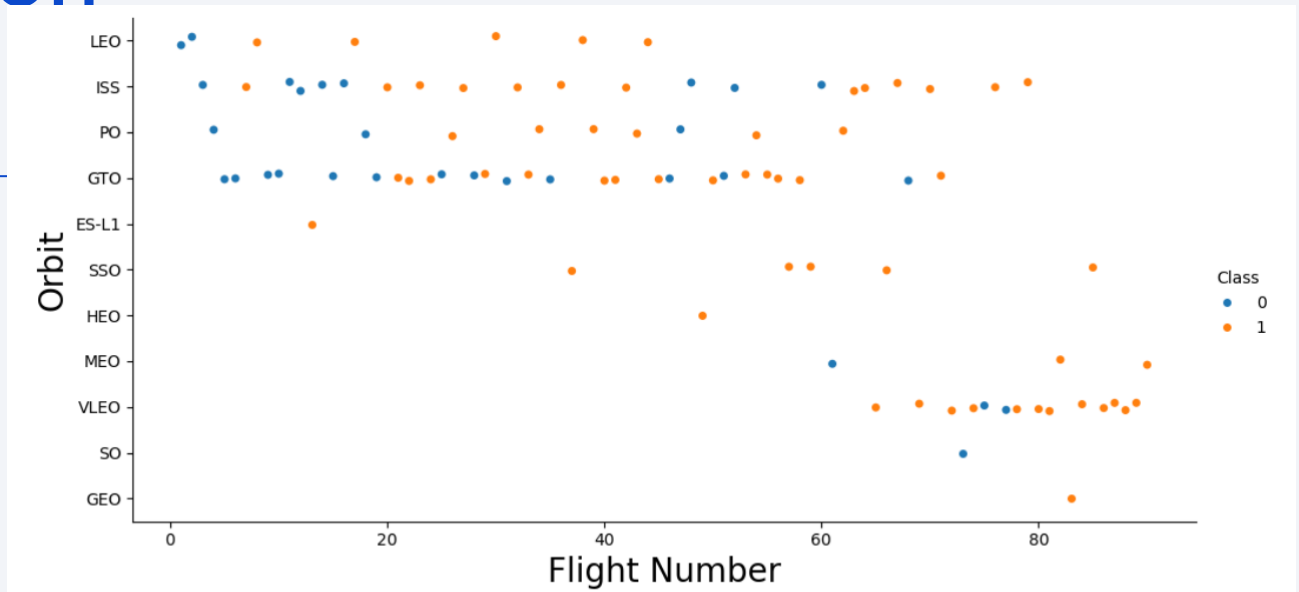


Now if you observe Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

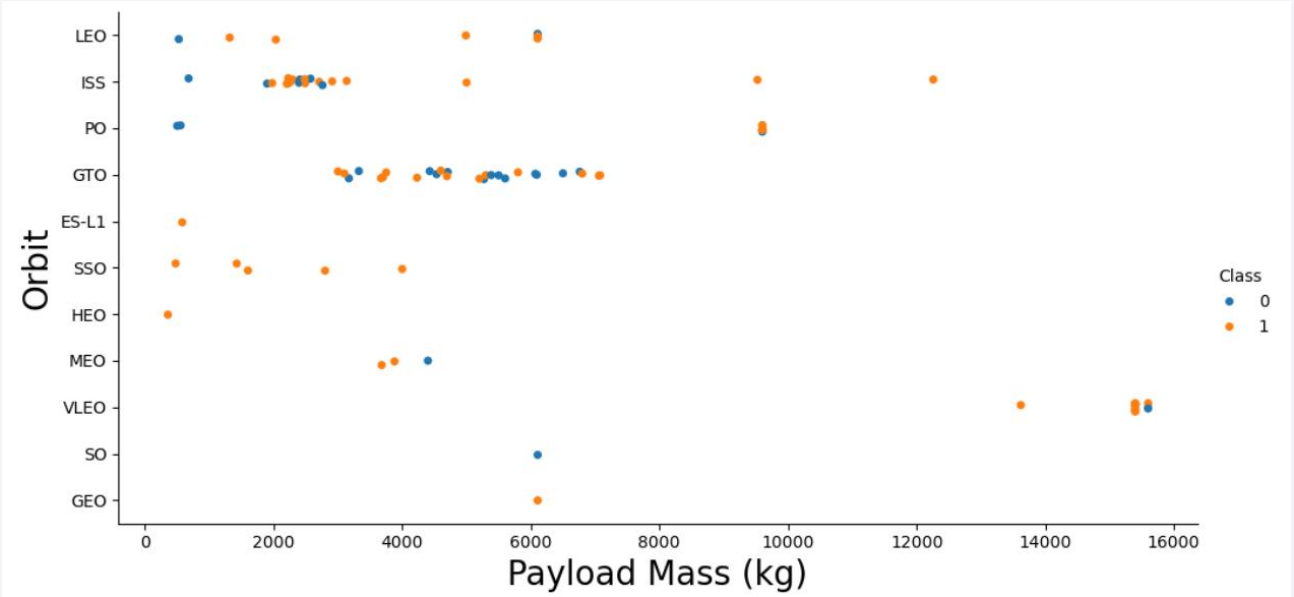
EDA with Data Visualization



The orbit Type influences the success Rate

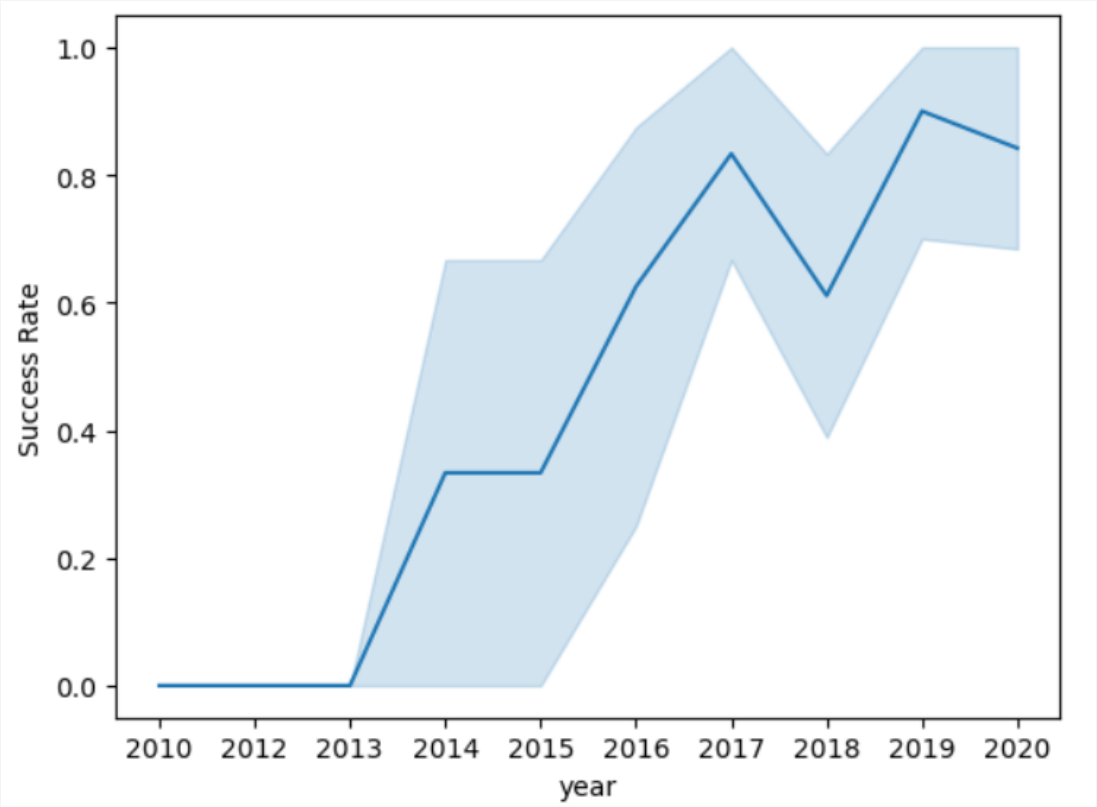


You should see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.



With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

EDA with Data Visualization



The success Rate of the launch increase as per year, according with the fact that increasing the number of fight increases the data available

GITHub link: lab_4_EDA_python_Vis.ipynb

EDA with SQL

- Using bullet point format, summarize the SQL queries you performed:

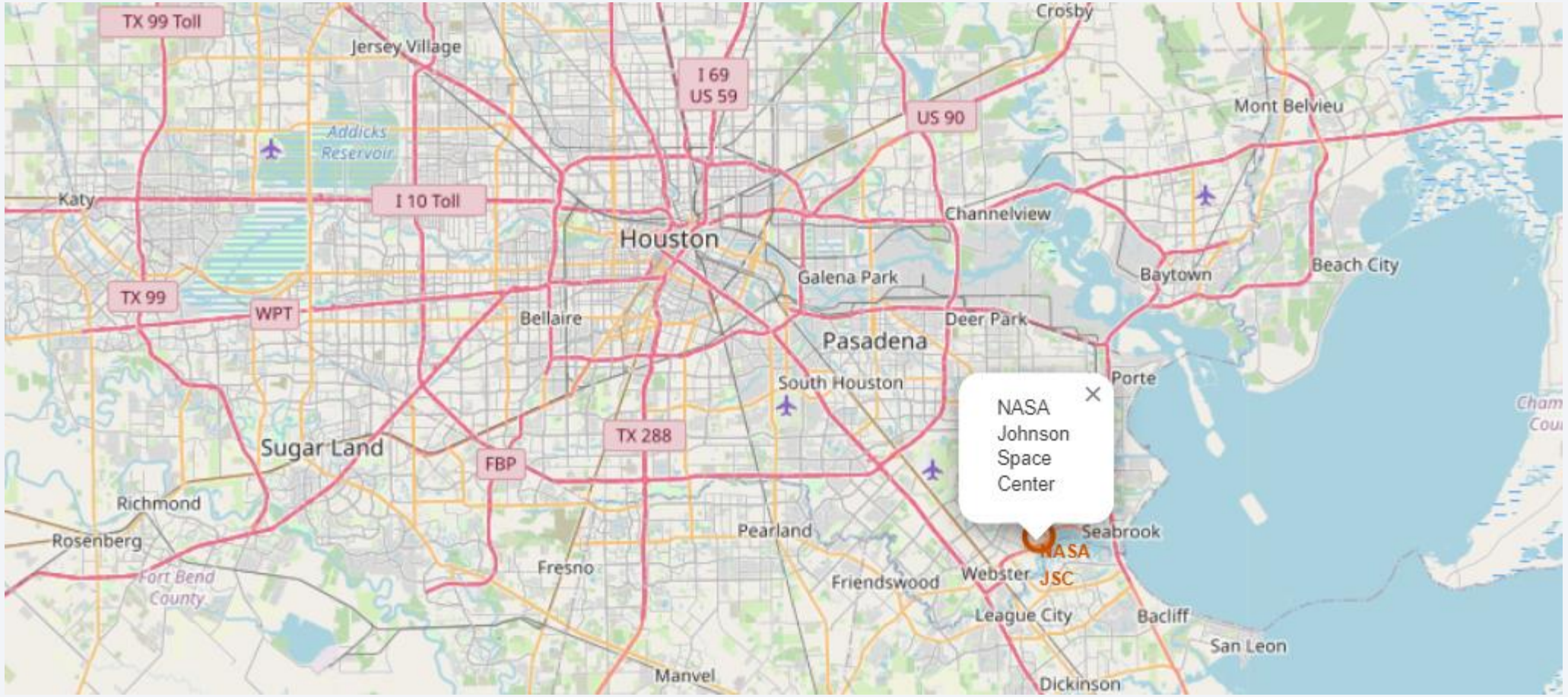
  - Task 1) %sql SELECT DISTINCT Launch_Site from SPACEXTABLE
  - Task 2) %sql select* from SPACEXTABLE WHERE Launch_Site LIKE 'CCA%' limit 5
  - Task 3) %sql select sum(payload_mass__kg_) from SPACEXTABLE where customer LIKE '%CRS%’
  - Task 4) %sql select avg(payload_mass__kg_) from SPACEXTABLE where booster_version='F9 v1.1’
  - Task 5) %sql select * from SPACEXTABLE where Landing_Outcome="Success (ground pad)“
  - Task 6) %sql select Booster_Version from SPACEXTABLE where Landing_Outcome="Success (ground pad)" and PAYLOAD_MASS__KG_>4000 and PAYLOAD_MASS__KG_<6000
  - Task 7) %sql SELECT Count(mission_outcome) from SPACEXTABLE where mission_outcome LIKE '%Success%’
  - Task 8) %sql SELECT Count(mission_outcome) from SPACEXTABLE where mission_outcome LIKE '%Failure%’
  - Task 9) %sql select booster_version, launch_site, Landing_Outcome,DATE from SPACEXTABLE where Landing_Outcome LIKE '%Failure (drone ship)%’
  - Task 10) %sql select Landing_Outcome, Date, count(Landing_Outcome) from SPACEXTABLE where DATE>'2010-06-04' AND DATE<='2017-03-20' Group by Landing_Outcome

GITHub link: lab_6_Folium.ipynb

Build an Interactive Map with Folium

- Summarize what map objects such as markers, circles, lines, etc. you created and added to a folium map

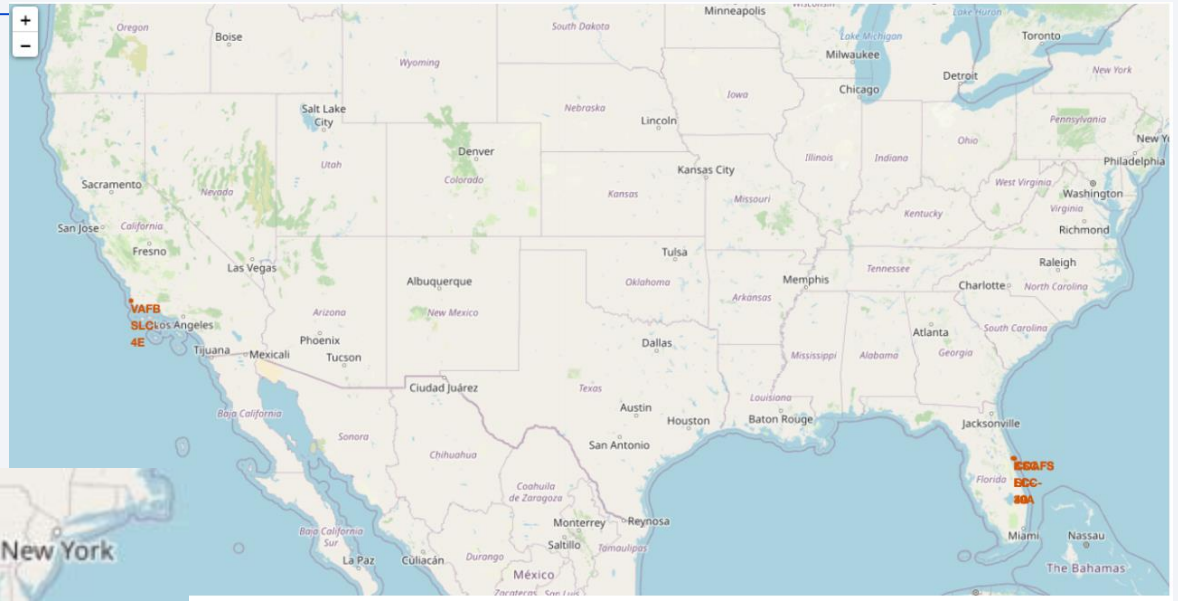
We first need to create a folium `Map` object, with an initial center location to be NASA Johnson Space Center at Houston, Texas.



Build an Interactive Map with Folium

- Summarize what map objects such as markers, circles, lines, etc. you created and added to a folium map

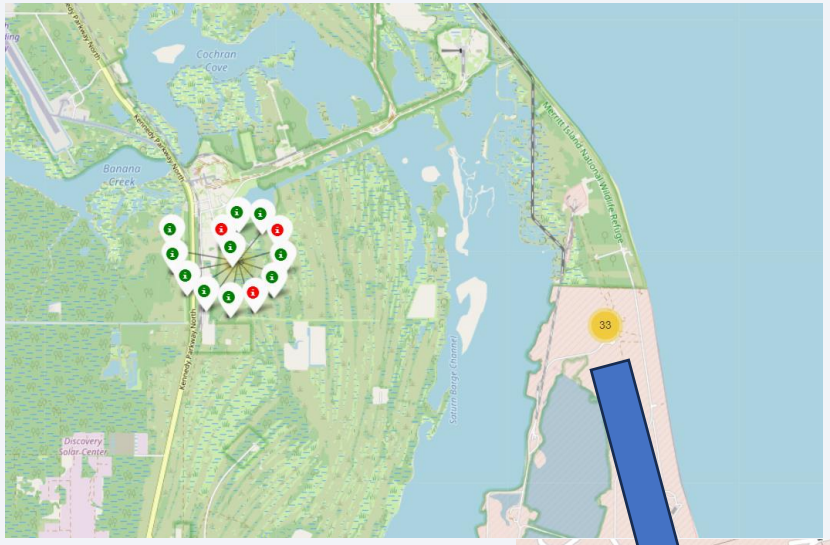
Now, let's add a circle for each launch site in data frame `launch_sites`

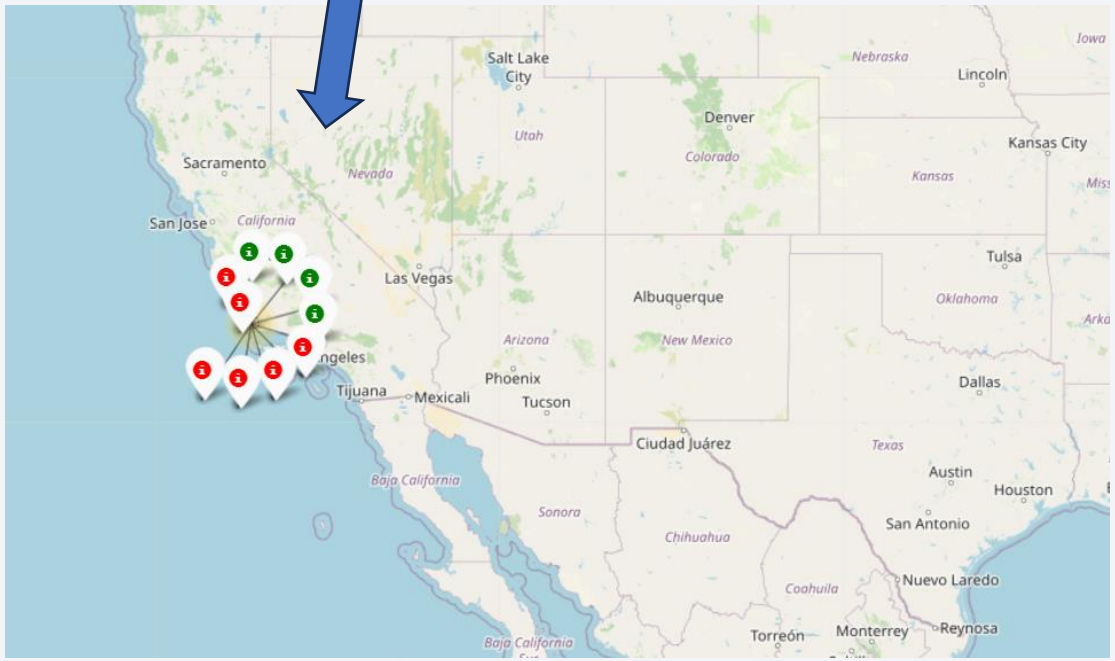


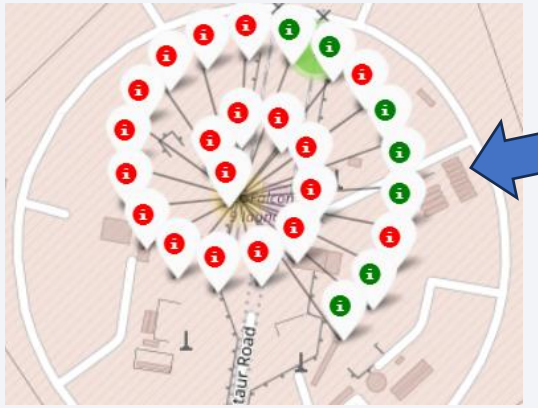
Build an Interactive Map with Folium

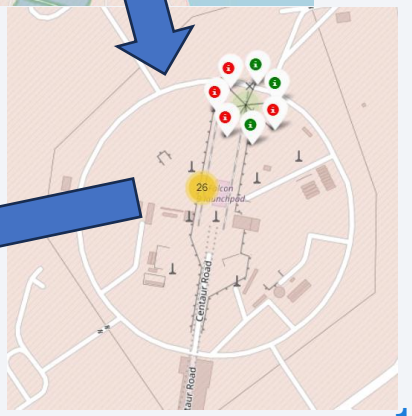
- Summarize what map objects such as markers, circles, lines, etc. you created and added to a folium map

Next, let's try to enhance the map by adding the launch outcomes for each site, and see which sites have high success rates.





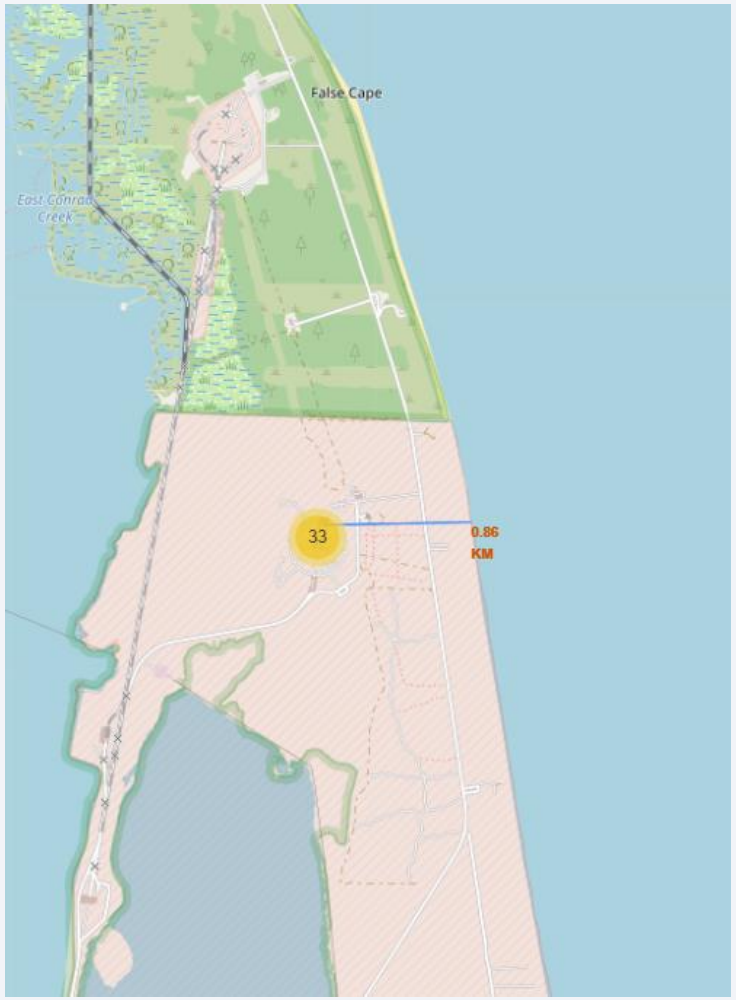




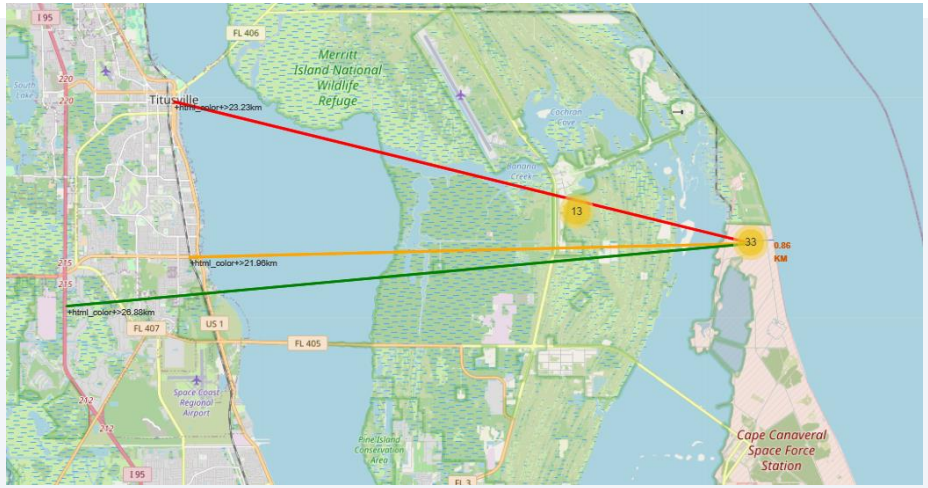
Build an Interactive Map with Folium

- Summarize what map objects such as markers, circles, lines, etc. you created and added to a folium map

Mark down a point on the closest coastline using MousePosition and calculate the distance between the coastline point and the launch site.



: Similarly, you can draw a line betwee a launch site to its closest city, railway, highway, etc. You need to use `MousePosition` to find the their coordinates on the map first



Build an Interactive Map with Folium

- Explain why you added those objects

After you plot distance lines to the proximities, you can answer the following questions easily:

- Are launch sites in close proximity to railways?
- Are launch sites in close proximity to highways?
- Are launch sites in close proximity to coastline?
- Do launch sites keep certain distance away from cities?

Also please try to explain your findings.

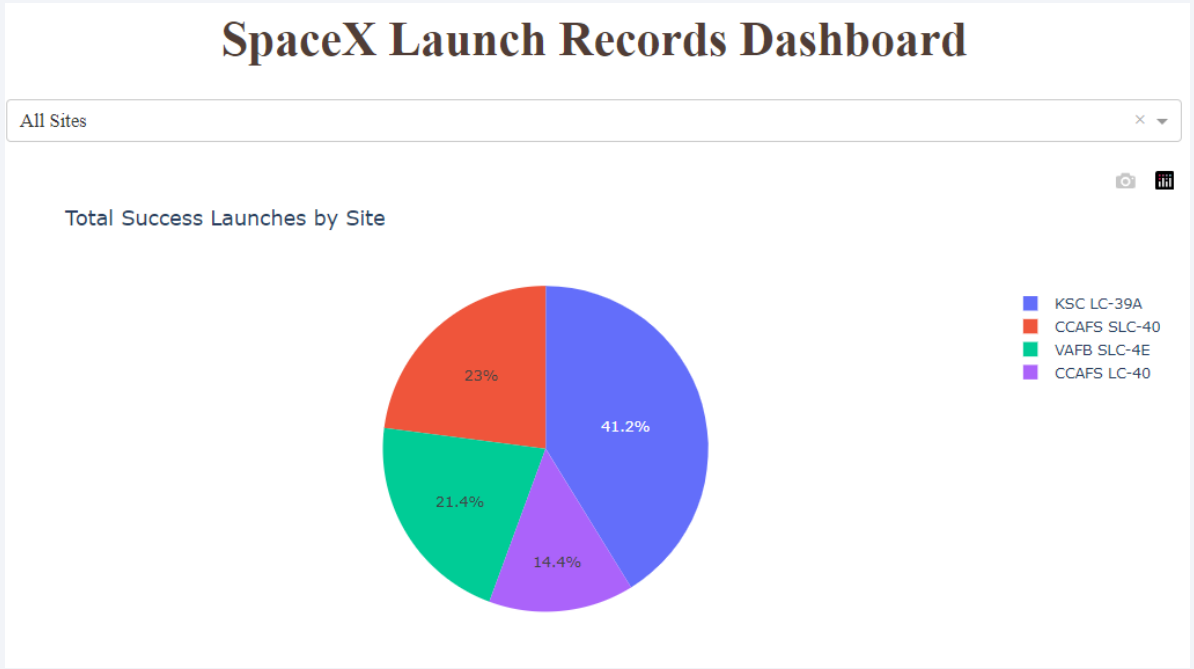
```
print("City Distance", city_distance)

print("Railway Distance", railway_distance)

print("Highway Distance", highway_distance)

print("Coastline Distance", distance_coastline)
```

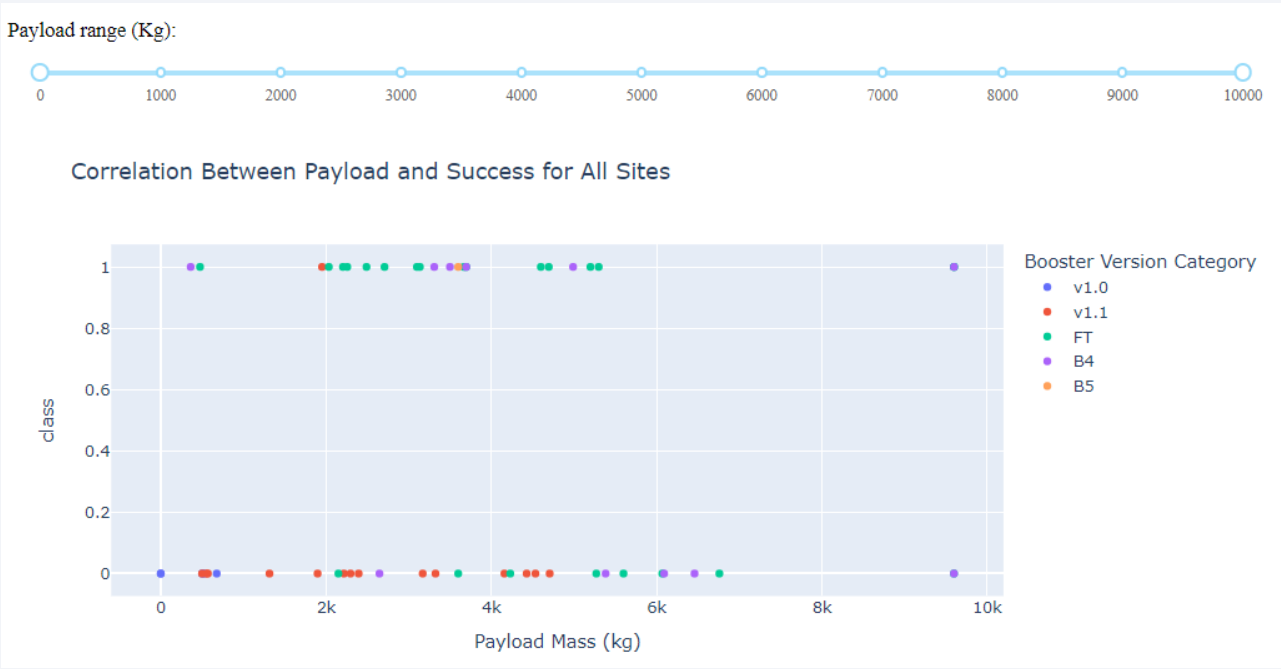
```
City Distance 23.234752126023245
Railway Distance 21.961465676043673
Highway Distance 26.88038569681492
Coastline Distance 0.8627671182499878
```

- 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? -> no
- Do launch sites keep certain distance away from cities? ->no

GitHub Link
lab_6_Folium.ipynb

Build a Dashboard with Plotly Dash





The Dashboard consists in tow main plots:
Pie-chart that show the total success Launches by Site, where we have added a drop down where is possible to select the single site launches
Scatter plot regarding the correlation between Payload and Success for all Site where we have had a range slider where we can filter by Payload range

[GitHub Link lab_7_Dash.py]

Predictive Analysis (Classification)

Load the Data

Preprocessing the data (standardize and trasmorm it)

transform = preprocessing.StandardScaler()X = transform.fit(X).transform(X)

Split the data in Train and Test set

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

For each model to be used (Log Regr, KNN, SVM, Dec Tree) define the hyperparameter and related value

I.e. for SVM model:
parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'), 'C': np.logspace(-3, 3, 5), 'gamma':np.logspace(-3, 3, 5)}

Tune the Hyperparameter for each ML model, fit on training data and find the related accuracy

i.e. for SVM model:
svm_cv = GridSearchCV(estimator=svm, cv=10, param_grid=parameters).fit(X_train, Y_train)
print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)print("accuracy :",svm_cv.best_score_)

Calculate the Confussion matrix and accuracy for each ML model for defining the best model

i.e. for SVM moder:
svm_cv_score = svm_cv.score(X_test, Y_test)
svm_yhat=svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,svm_yhat)

GitHub Link lab_8_ML.ipynb

Predictive Analysis (Classification)

Decision Tree Model

Best model

```
print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
print("accuracy :",tree_cv.best_score_)
print("best estimator :",tree_cv.best_estimator_)
```

```
tuned hpyerparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 16, 'max_features': 'sqrt', 'min_samples_lea
f': 1, 'min_samples_split': 2, 'splitter': 'random'}
accuracy : 0.8732142857142857
best estimator : DecisionTreeClassifier(criterion='entropy', max_depth=16, max_features='sqrt',
                       splitter='random')
```

```
tree_cv_score = svm_cv.score(X_test, Y_test)
print("score ",tree_cv_score)
```

```
score  0.8333333333333334
```

Logistic Regression

```
print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
print("accuracy :",logreg_cv.best_score_)
print("best estimator :",logreg_cv.best_estimator_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
accuracy : 0.8464285714285713
best estimator : LogisticRegression(C=0.01)
```

```
score_logreg = logreg_cv.score(X_test, Y_test)
print("accuracy_test: ",score_logreg)
```

```
accuracy_test:  0.8333333333333334
```

SVM Model

```
print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
print("accuracy :",svm_cv.best_score_)
print("best estimator :",svm_cv.best_estimator_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
accuracy : 0.8482142857142856
best estimator : SVC(gamma=0.03162277660168379, kernel='sigmoid')
```

```
score_svm = svm_cv.score(X_test, Y_test)
print("svm_accuracy_test: ",score_svm)
```

```
svm_accuracy_test:  0.8333333333333334
```

kNN Model

```
print("tuned hpyerparameters :(best parameters) ",knn_cv.best_params_)
print("accuracy :",knn_cv.best_score_)
print("best estimator :",knn_cv.best_estimator_)
```

```
tuned hpyerparameters :(best parameters)  {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
accuracy : 0.8482142857142858
best estimator : KNeighborsClassifier(n_neighbors=10, p=1)
```

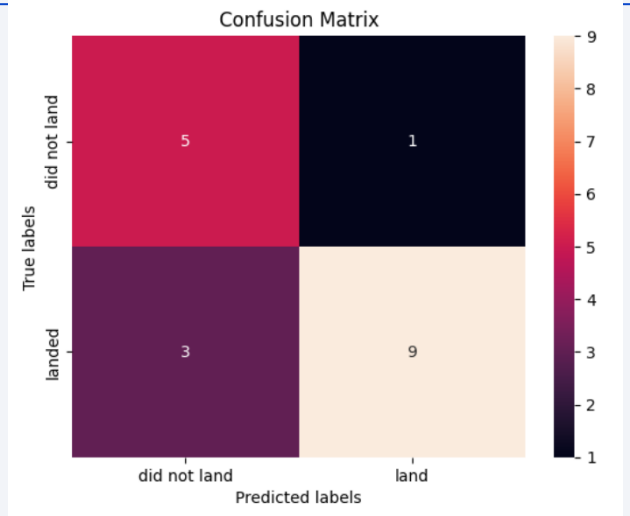
```
knn_cv_score = knn_cv.score(X_test, Y_test)
print("score :",knn_cv_score)
```

```
score : 0.8333333333333334
```

Predictive Analysis (Classification)

Decision Tree Model

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

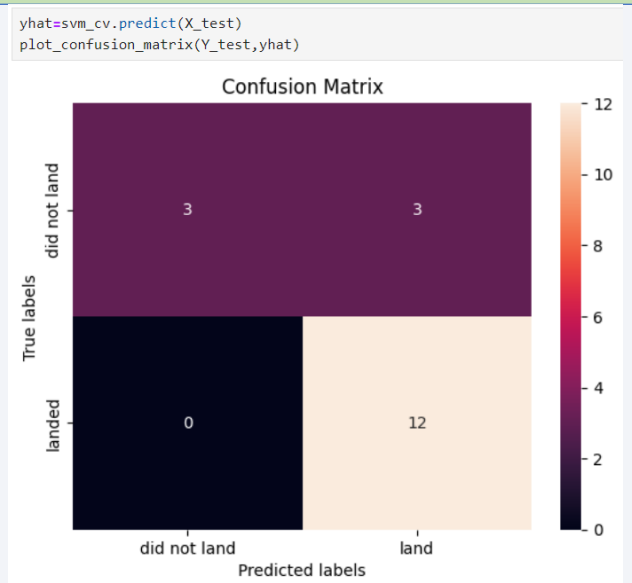


Logistic Regression

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

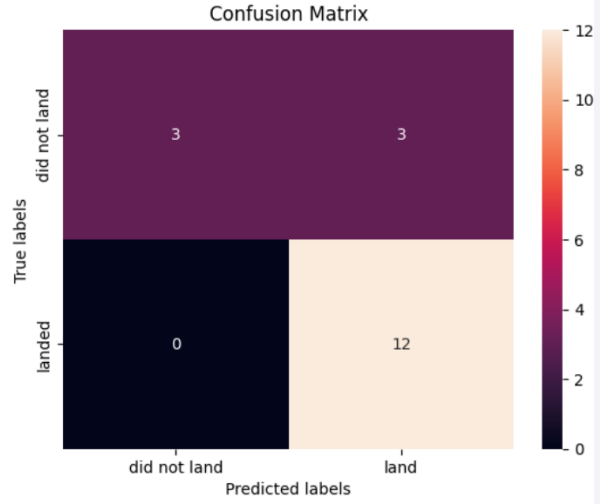
SVM Model

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



kNN Model

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



Results

Consididering that our target is to maximixe the success of the launch in phase 1 in order to compete with space X, we were able to :

- Exploratory data analysis results -> find the best parameter and value that impact the success of the Launch: number of launch, Payload of the launch and Site of Launch
- Interactive analytics demo in screenshots -> we have been able to identify the best Site for the launch and the best correlation between Payload and Success
- Predictive analysis results -> find that the Decision Tree is the best ML model to predict the outcome of the launch

Thank you!