Winning Space Race with Data Science

Executive Summary

- To predict if the first stage of the SpaceX Falcon 9 rocket will land successfully or not.
- As the above question tells us that we will be using the classification method on previous launches to predict if this launch will be successful or not.
- We will be using many analytical and visualizing options to convey a strong message about our findings.

Introduction

- 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 first stage
- Thus we have to predict if the first stage of the SpaceX Falcon 9 rocket will land successfully

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected through SpaceX REST API and wikipedia through Beautiful Soup package
- Perform data wrangling
  - For analysis we converted categorical variables to numericals through one hot encoding and replaced all null values with mean of that column.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform predictive analysis using classification models
  - We can build 4 types of classification models for comparison and use the constructor object and further use that in GridSearchCV so that we can get best parameters for modelling by measuring there accuracy.

Data Collection

- Our Dataset was collected with the help of SpaceX Rest API
- We will use this URL to target a specific endpoint of the API to get past launch data. We will perform a get request using the requests library to obtain the launch data, which we will use to get the data from the API. This result can be viewed by calling the .json() method.

We will be using the Python BeautifulSoup package to web scrape some HTML tables that contain valuable Falcon 9 launch records. Then we need to parse the data from those tables and convert them into a Pandas data frame for further visualization and analysis.

Now let's start requesting rocket launch data from SpaceX API with the following URL:

```
In [6]: spacex_url="https://api.spacexdata.com/v4/launches/past"

In [7]: response = requests.get(spacex_url)
```

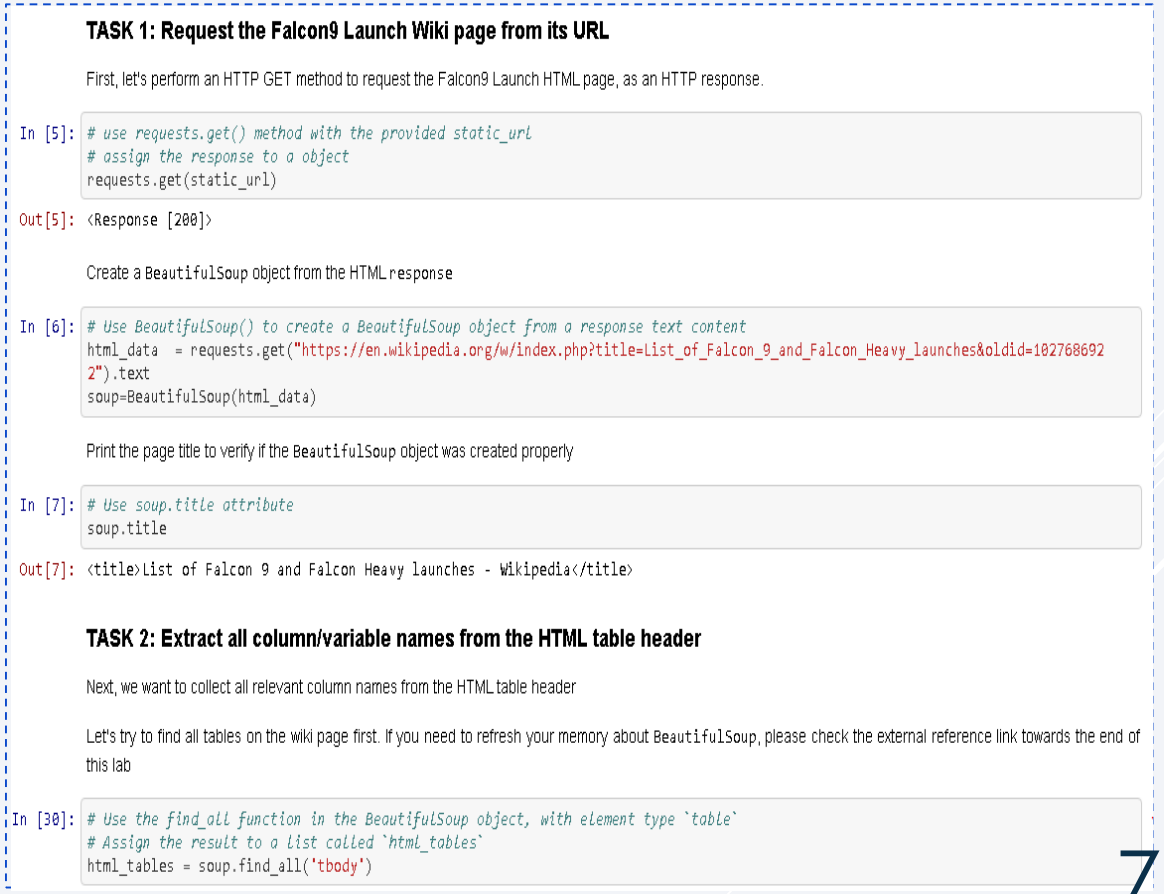
Check the content of the response

```
In [8]: print(response.content)
```


```
b'[{"fairings":{"reused":false,"recovery_attempt":false,"recovered":false,"ships":[]},"links":{"patch":{"small":"https://im
ages2.imgbox.com/3c/0e/T8iJcSN3_o.png","large":"https://images2.imgbox.com/40/e3/GypSkayF_o.png"},"reddit":{"campaign":nul
l,"launch":null,"media":null,"recovery":null},"flickr":{"small":[],"original":[]},"presskit":null,"webcast":"https://www.yo
utube.com/watch?v=0a_00nJ_Y88","youtube_id":"0a_00nJ_Y88","article":"https://www.space.com/2196-spacex-inaugural-falcon-1-r
ocket-lost-launch.html","wikipedia":"https://en.wikipedia.org/wiki/DemoSat"},"static_fire_date_utc":"2006-03-17T00:00:00.00
0Z","static_fire_date_unix":1142553600,"net":false,"window":0,"rocket":"5e9d0d95eda69955f709d1eb","success":false,"failure
s":[{"time":33,"altitude":null,"reason":"merlin engine failure"}],"details":"Engine failure at 33 seconds and loss of vehic
le","crew":[],"ships":[],"capsules":[],"payloads":["5eb0e4b5b6c3bb0006eeb1e1"],"launchpad":"5e9e4502f5090995de566f86","flig
ht_number":1,"name":"FalconSat","date_utc":"2006-03-24T22:30:00.000Z","date_unix":1143239400,"date_local":"2006-03-25T10:3
0:00+12:00","date_precision":"hour","upcoming":false,"cores":[{"core":"5e9e289df35918033d3b2623","flight":1,"gridfins":fals
e,"legs":false,"reused":false,"landing_attempt":false,"landing_success":null,"landing_type":null,"landpad":null}],"auto_upd
ate":true,"tbd":false,"launch_library_id":null,"id":"5eb87cd9ffd86e000604b32a"},{"fairings":{"reused":false,"recovery_attem
pt":false,"recovered":false,"ships":[]},"links":{"patch":{"small":"https://images2.imgbox.com/4f/e3/I0lkuJ2e_o.png","larg
e":"https://images2.imgbox.com/be/e7/iNqsqVYM_o.png"},"reddit":{"campaign":null,"launch":null,"media":null,"recovery":nul
l},"flickr":{"small":[],"original":[]},"presskit":null,"webcast":"https://www.youtube.com/watch?v=Lk4zQ2wP-Nc","youtube_i
d":"Lk4zQ2wP-Nc","article":"https://www.space.com/3590-spacex-falcon-1-rocket-fails-reach-orbit.html","wikipedia":"https://
en.wikipedia.org/wiki/DemoSat"},"static_fire_date_utc":null,"static_fire_date_unix":null,"net":false,"window":0,"rocket":"5
e9d0d95eda69955f709d1eb","success":false,"failures":[{"time":301,"altitude":289,"reason":"harmonic oscillation leading to p
remature engine shutdown"}],"details":"Successful first stage burn and transition to second stage, maximum altitude 289 km,
Premature engine shutdown at T+7 min 30 s, Failed to reach orbit, Failed to recover first stage","crew":[],"ships":[],"caps
ules":[],"payloads":["5eb0e4b6b6c3bb0006eeb1e2"],"launchpad":"5e9e4502f5090995de566f86","flight_number":2,"name":"DemoSa
t","date_utc":"2007-03-21T01:10:00.000Z","date_unix":1174439400,"date_local":"2007-03-21T13:10:00+12:00","date_precisio
n":"hour","upcoming":false,"cores":[{"core":"5e9e289ef35918416a3b2624","flight":1,"gridfins":false,"legs":false,"reused":fa
lse,"landing_attempt":false,"landing_success":null,"landing_type":null,"landpad":null}],"auto_update":true,"tbd":false,"lau
nch_library_id":null,"id":"5eb87cd9ffd86e000604b32b"},{"fairings":{"reused":false,"recovery_attempt":false,"recovered":fals
```

Data Collection - Scraping

- We want to transform this raw data into a clean dataset which provides meaningful data on the situation we are trying to address: Wrangling Data using an API, Sampling Data, and Dealing with Nulls.



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 [5]: # use requests.get() method with the provided static_url
        # assign the response to a object
        requests.get(static_url)
Out[5]: <Response [200]>
```

Create a `BeautifulSoup` object from the HTML `response`

```
In [6]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
        html_data  = requests.get("https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922").text
        soup=BeautifulSoup(html_data)
```

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

```
In [7]: # Use soup.title attribute
        soup.title
Out[7]: <title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

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 [30]: # 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('tbody')
```

Data Wrangling

- The column “LaunchSite” contains the different launch sites, including:
  - Vandenberg AFB Space Launch
  - Kennedy Space Center
  - CCAFS SLC 40
- The column orbits are the different orbits of the payload.

For Example:

- LEO: Low Earth orbit (LEO)is an Earth-centered orbit with an altitude of 2,000 km
- GTO A geosynchronous orbit is a high Earth orbit that allows satellites to match Earth's rotation. It is located at 22,236 miles (35,786 kilometers) above Earth's equator.
- The column Outcome indicates if the first stage successfully landed.

EDA with Data Visualization

- Three visualizations were mainly used and those were:

1. Scatter Point Chart: Mainly to compare two elements like launch site vs payload mass so as to get knowledge that which site can bear more load.
2. Bar Chart: for sucess rate of each orbit
3. Line Chart: to get the average launch success trend

EDA with SQL

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

- Display the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string 'CCA
- Display the total payload mass carried by boosters launched by NASA (CRS)
- Display average payload mass carried by booster version F9 v1.1
- List the date when the first successful landing outcome in ground pad was acheived.
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- ETC.

Build an Interactive Map with Folium

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

- Firstly addition the launch sites on the map
- Next addition of marker cluster to mark successful or failed landings
- Thirdly visulaizing how much far or near are launch sites to proximities.

Predictive Analysis (Classification)

- We will train the model and perform Grid Search, allowing us to find the hyperparameters that allow a given algorithm to perform best.
- Using the best hyperparameter values, we will determine the model with the best accuracy using the training data.
- We will test Logistic Regression, Support Vector machines, Decision Tree Classifier, and K-nearest neighbors. Finally, we will output the confusion matrix.

Results

1. Exploratory data analysis results:

After exploring data we found the success rate of landing but it was purely based on previous landings, no model was constructed

We can use the following line of code to determine the success rate:

```
In [13]: df["Class"].mean()
Out[13]: 0.6666666666666666
```

Results

2. Predictive analysis results:

Analysis of our classification Models are

i. KNN

Create a k nearest neighbors object then create a `GridSearchCV` object `knn_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

In [25]:
```
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()
```

In [26]:
```
gs3 = GridSearchCV(KNN,parameters,cv=10)
knn_cv = gs3.fit(X_train,Y_train)
```

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

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

Results

2. Predictive analysis results:

Analysis of our classification Models are

ii. Decision Tree

TASK 8

Create a decision tree classifier object then create a `GridSearchCV` object `tree_cv` with `cv = 10`. Fit the object to find the best parameters from the dictionary `parameters`.

In [20]:
```
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 [21]:
```
gs2 = GridSearchCV(tree,parameters,cv=10)
tree_cv = gs2.fit(X_train,Y_train)
```

In [22]:
```
print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
print("accuracy :",tree_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 10, 'max_features': 'auto', 'min_samples_leaf': 1, 'min
_samples_split': 10, 'splitter': 'random'}
accuracy : 0.8875000000000002
```

Results

2. Predictive analysis results:

Analysis of our classification Models are

iii. Logistic Regression

TASK 4

Create a logistic regression object then create a GridSearchCV object `logreg_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

In [10]:
```
parameters ={'C':[0.01,0.1,1],
             'penalty':['l2'],
             'solver':['lbfgs']}
```

In [11]:
```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
lr=LogisticRegression()
gs = GridSearchCV(lr,parameters,cv=10)
logreg_cv = gs.fit(X_train,Y_train)
```

We output the `GridSearchCV` object for logistic regression. We display the best parameters using the data attribute `best_params\_` and the accuracy on the validation data using the data attribute `best_score\_`.

In [12]:
```
print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
print("accuracy :",logreg_cv.best_score_)
```

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

Results

2. Predictive analysis results:

Analysis of our classification Models are

iv. Support Vector Machine

TASK 6

Create a support vector machine object then create a GridSearchCV object svm_cv with cv - 10. Fit the object to find the best parameters from the dictionary parameters.

In [15]:
```
parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
              'C': np.logspace(-3, 3, 5),
              'gamma':np.logspace(-3, 3, 5)}
svm = SVC()
```

In [16]:
```
gs1 = GridSearchCV(svm,parameters,cv=10)
svm_cv = gs1.fit(X_train,Y_train)
```

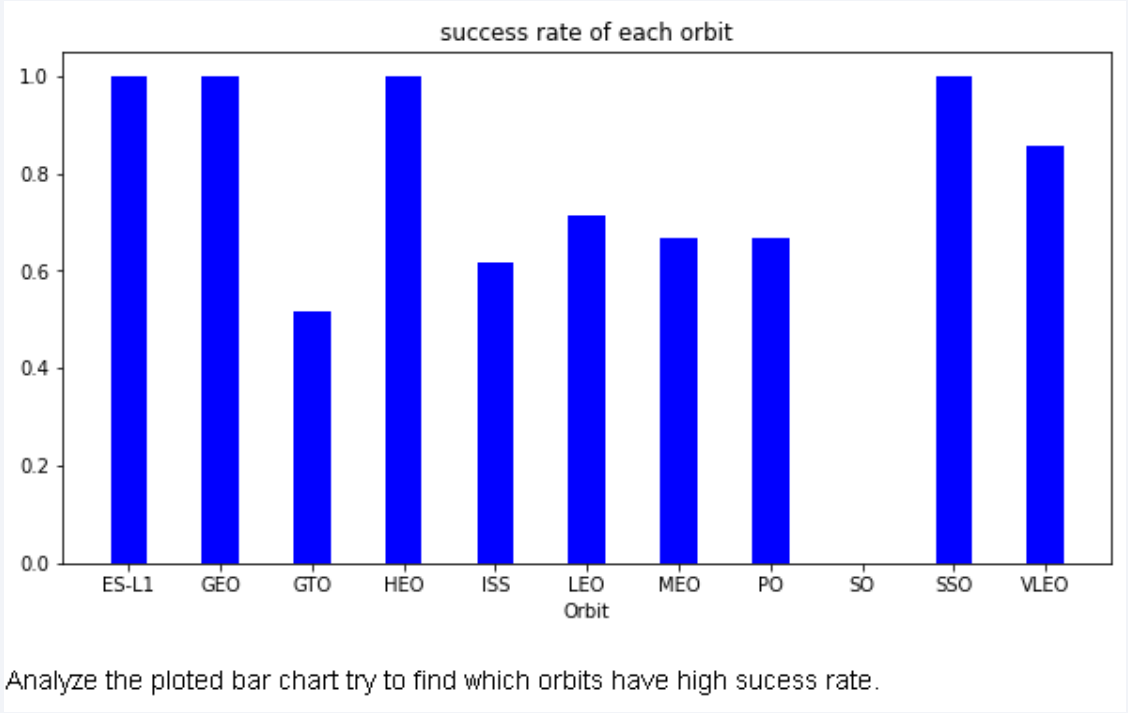
In [17]:
```
print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
print("accuracy :",svm_cv.best_score_)
```

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

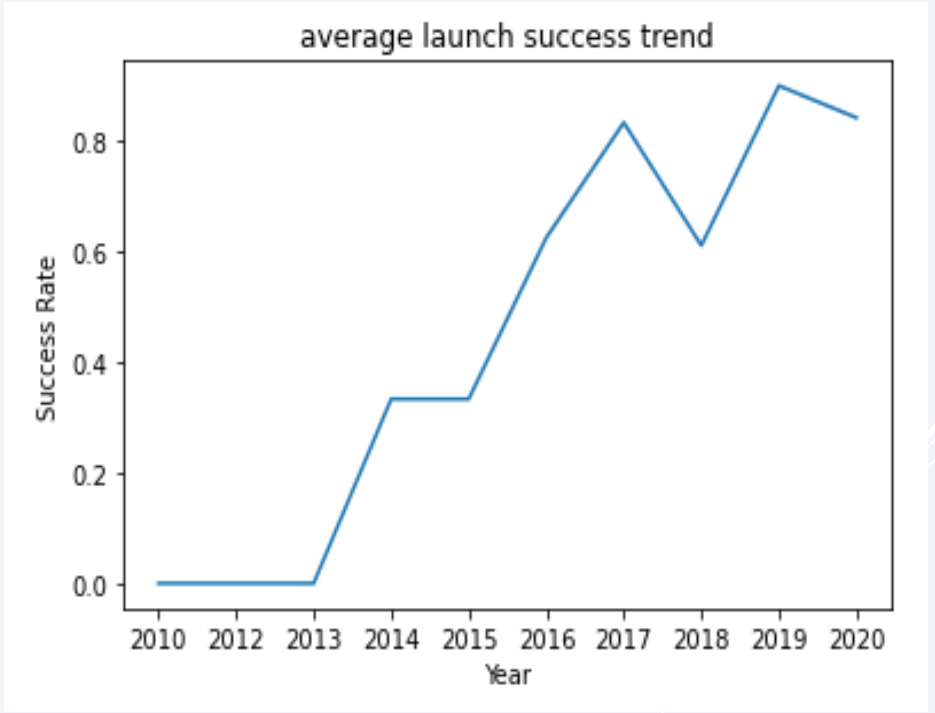
Results

3. Interactive analytics demo in screenshots

Through Visual Analytics we got to know success rate of each orbit and average launch success trend.

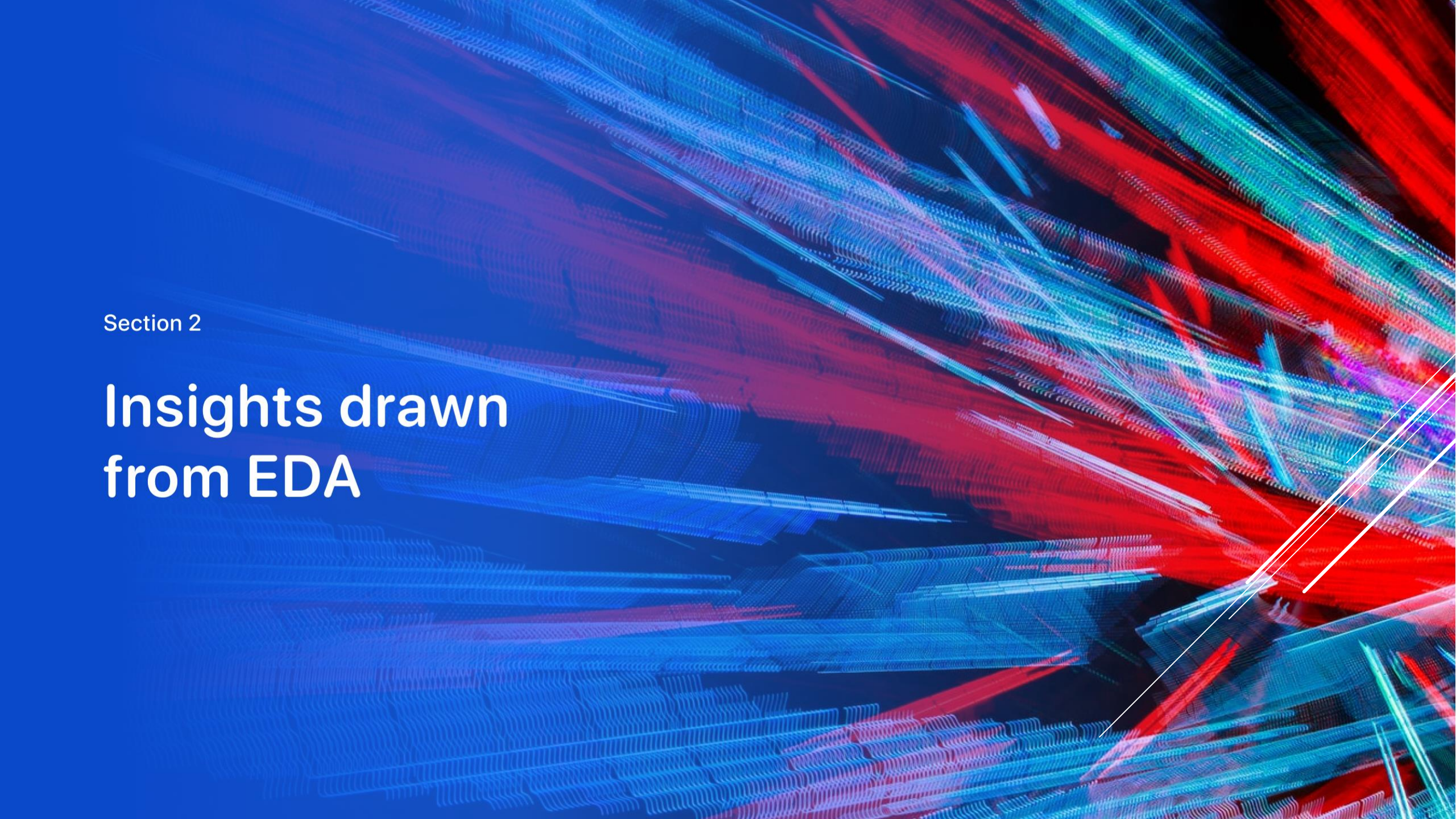


Analyze the ploted bar chart try to find which orbits have high sucess rate.

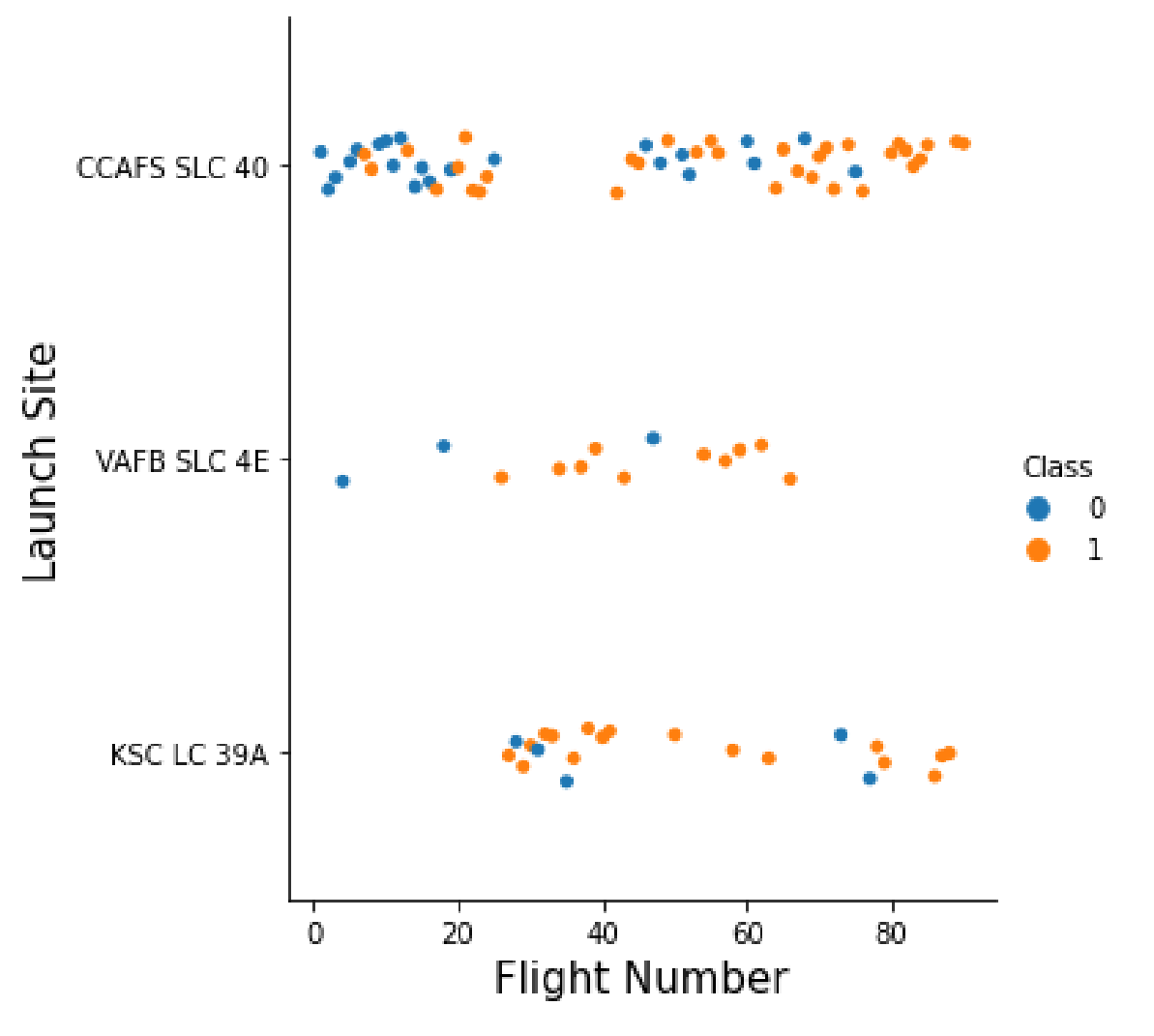


Section 2

Insights drawn from EDA

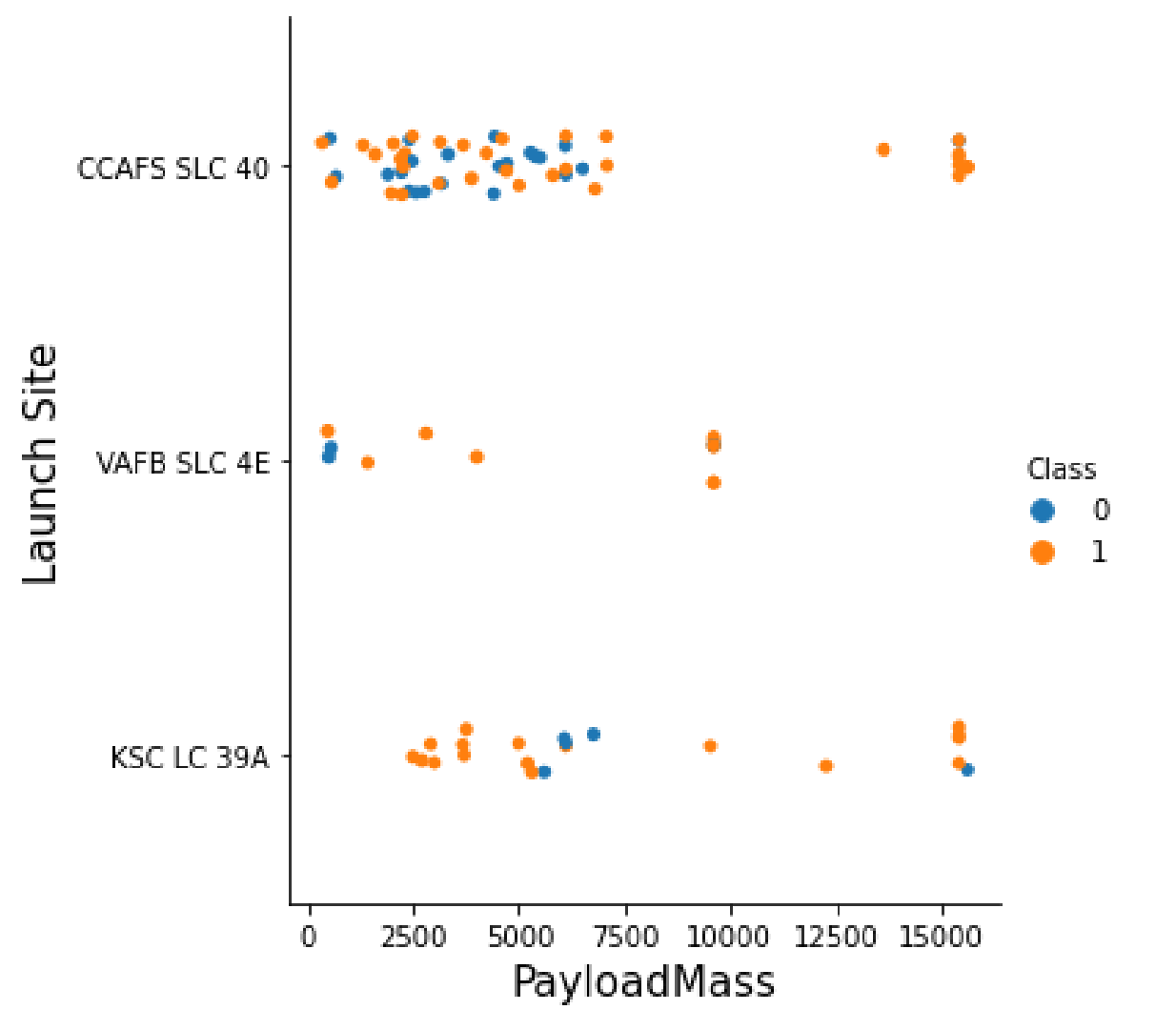


Flight Number vs. Launch Site



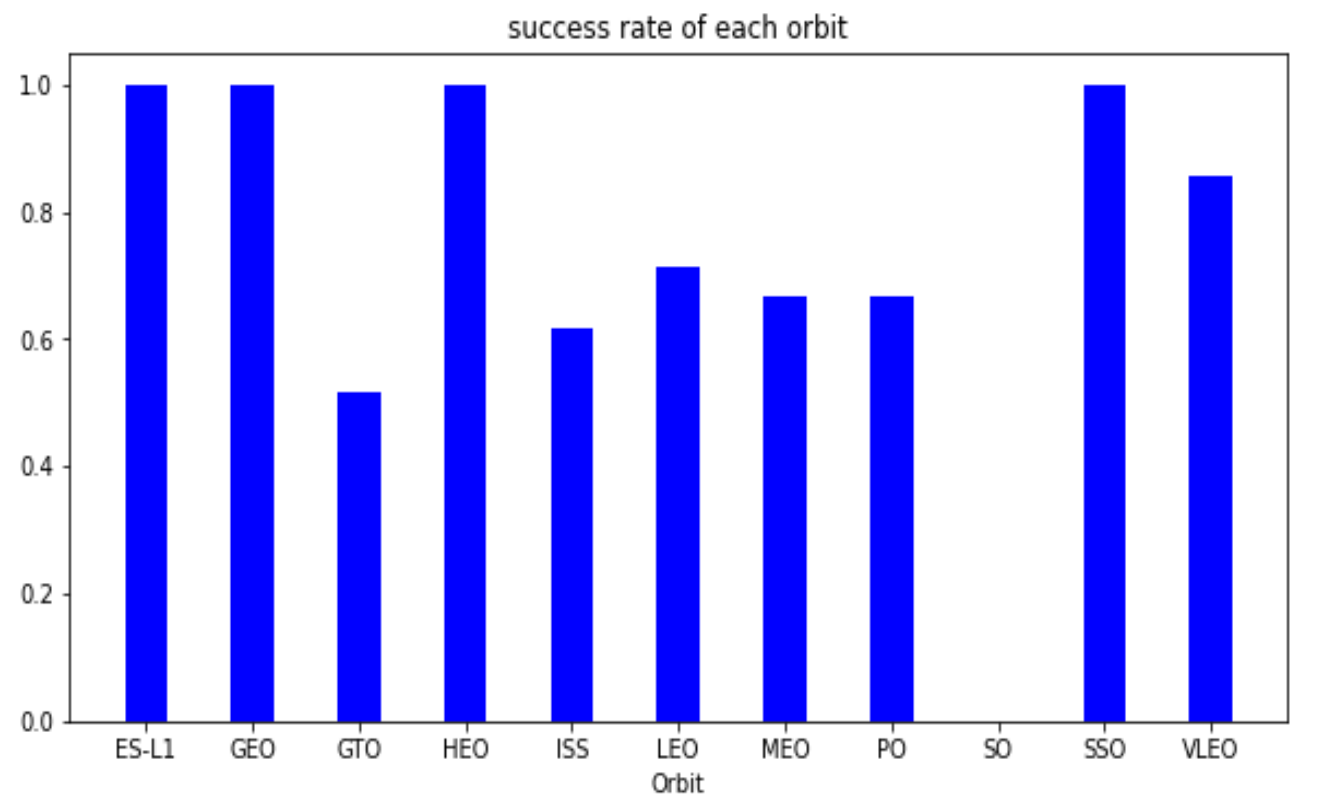
In this plot its evident that a continious flight number is scattered at launch site CCAFS SLC 40 whereas it is not at VAFB SLC 4E

Payload vs. Launch Site



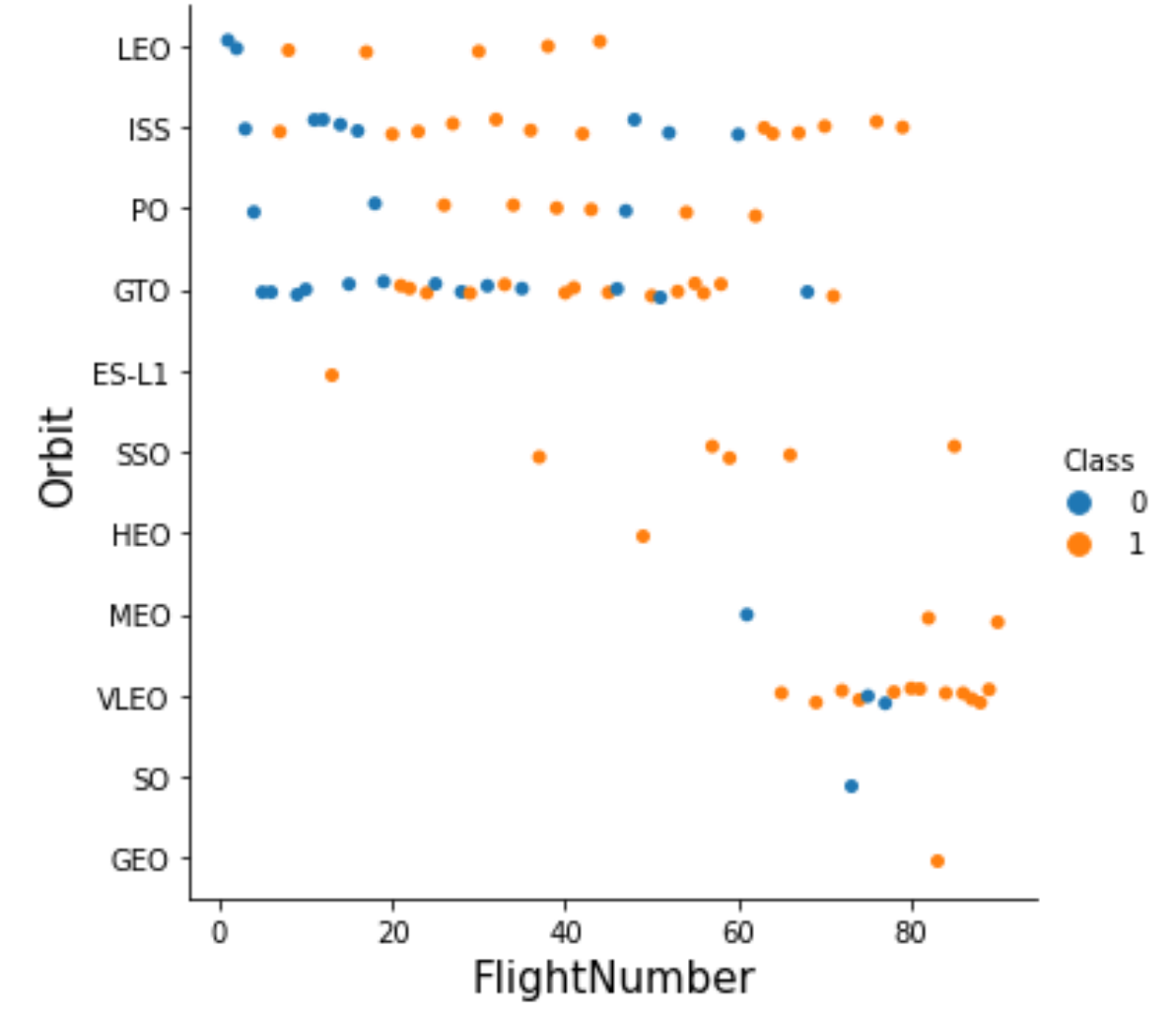
CCAFS SLC 40 Launch site majorly caters low payload mass launches

Success Rate vs. Orbit Type



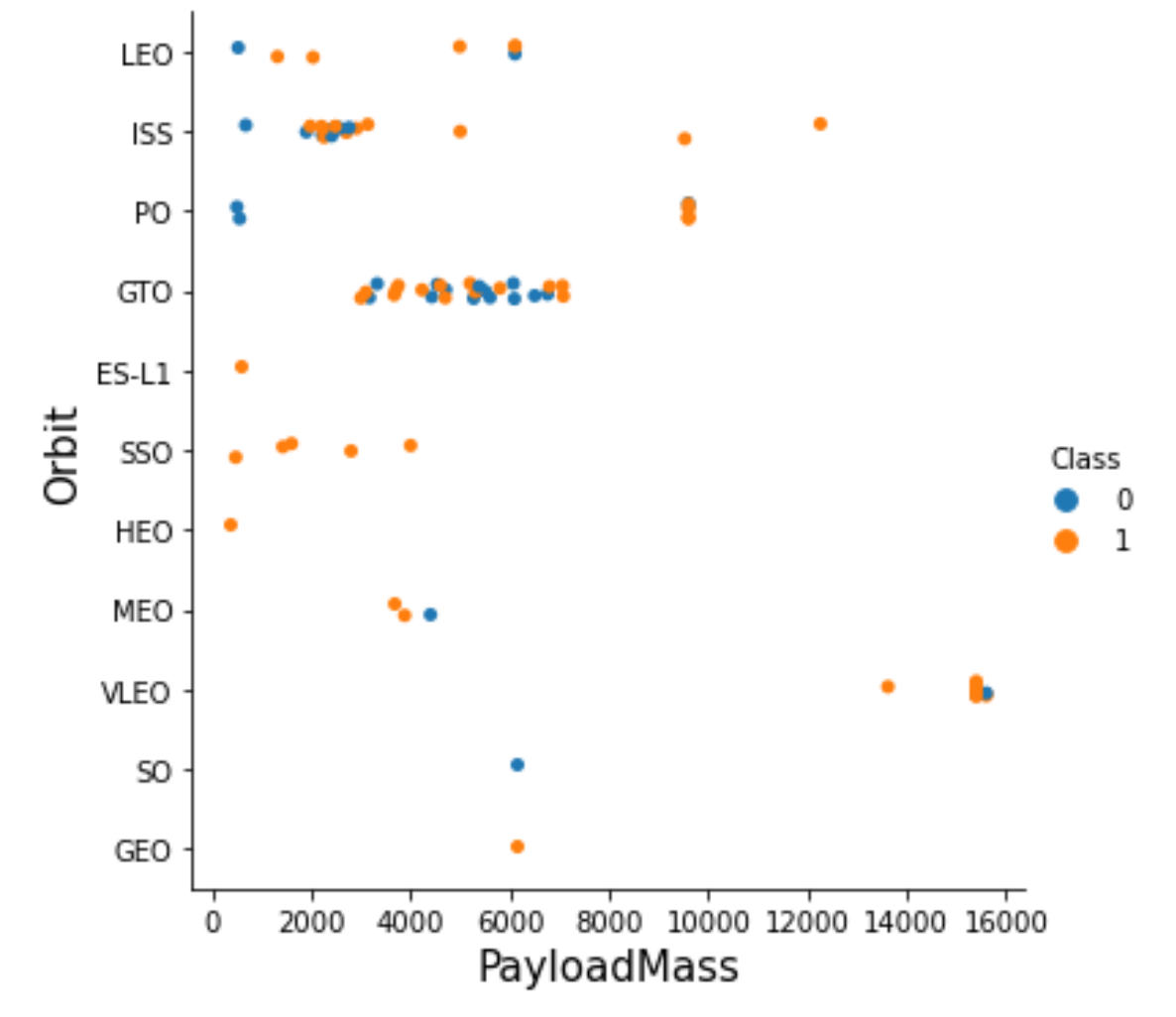
Success of ES-L1, GEO, HEO, SSO orbit is more as compared to other orbits.

Flight Number vs. Orbit Type



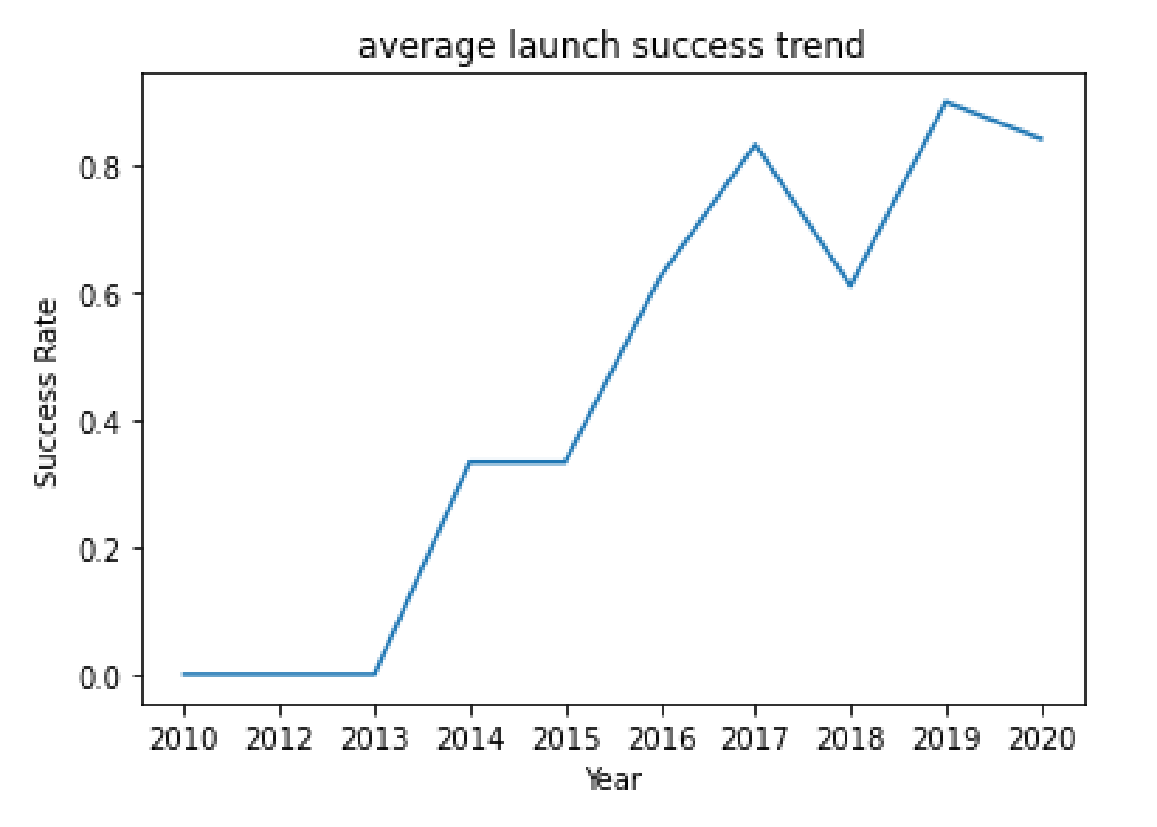
You should see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

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 sucess rate since 2013 kept increasing till 2020 but it also declined during 2017 to 2018.

All Launch Site Names

Find the names of the unique launch sites

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

In [4]:
```
%%sql

SELECT Distinct(LAUNCH_SITE)
FROM DLG69347.SPACEXTBL
```

```
 * ibm_db_sa://dlg69347:***@fbd88901-ebdb-4a4f-a32e-9822
Done.
```

Out[4]:

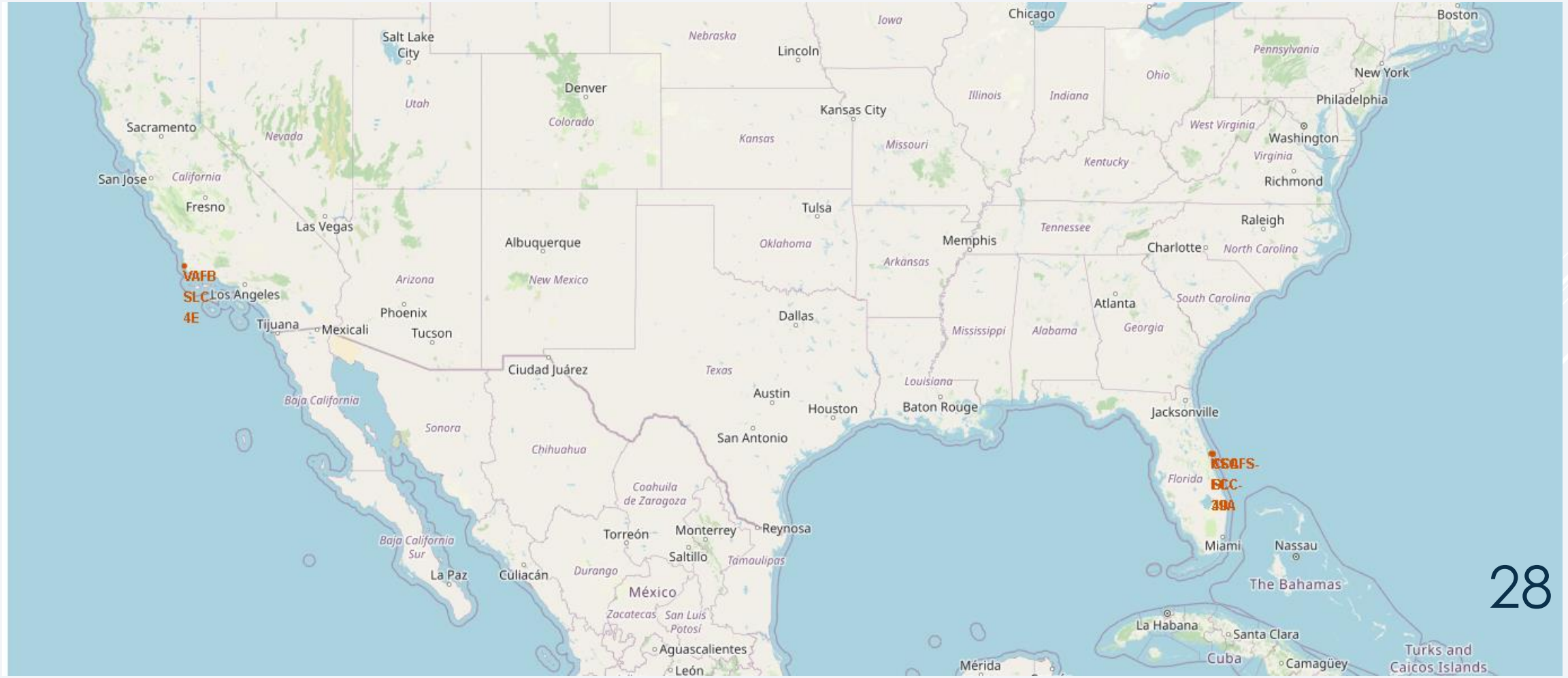
launch_site
CCAFS LC-40
CCAFS SLC-40
KSC LC-39A
VAFB SLC-4E

Section 3

Launch Sites Proximities Analysis

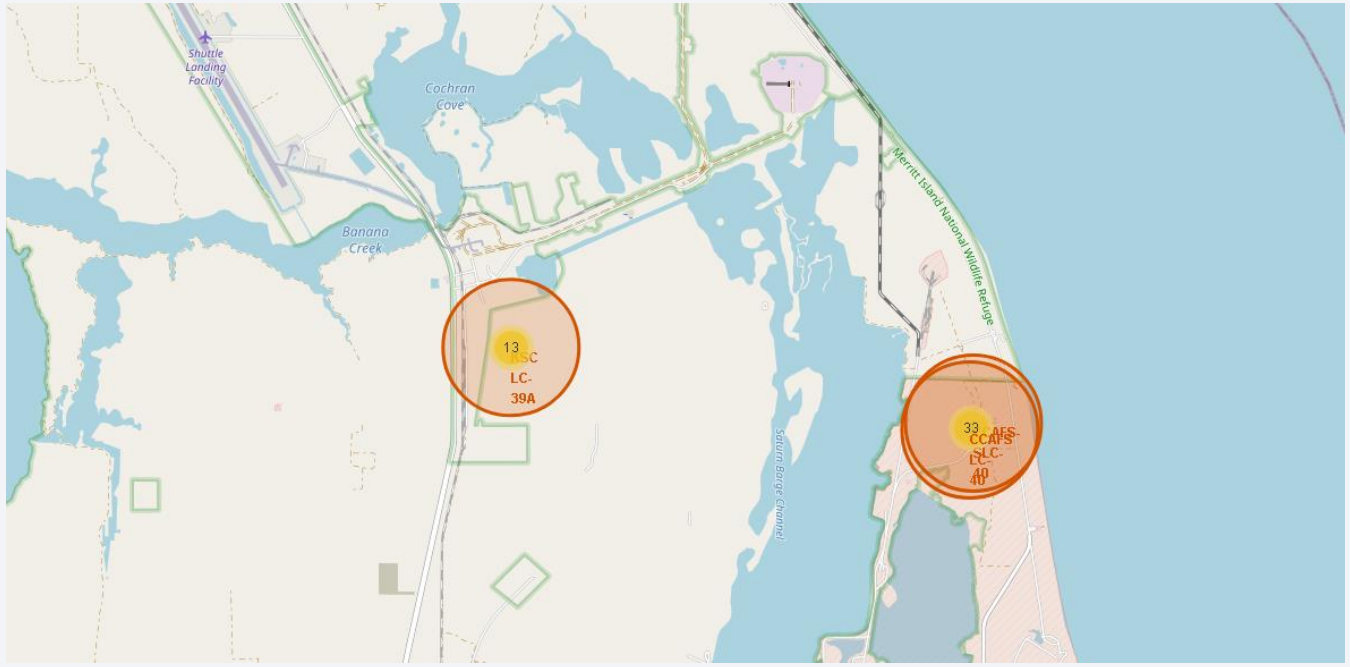
Marking All The Launch Site

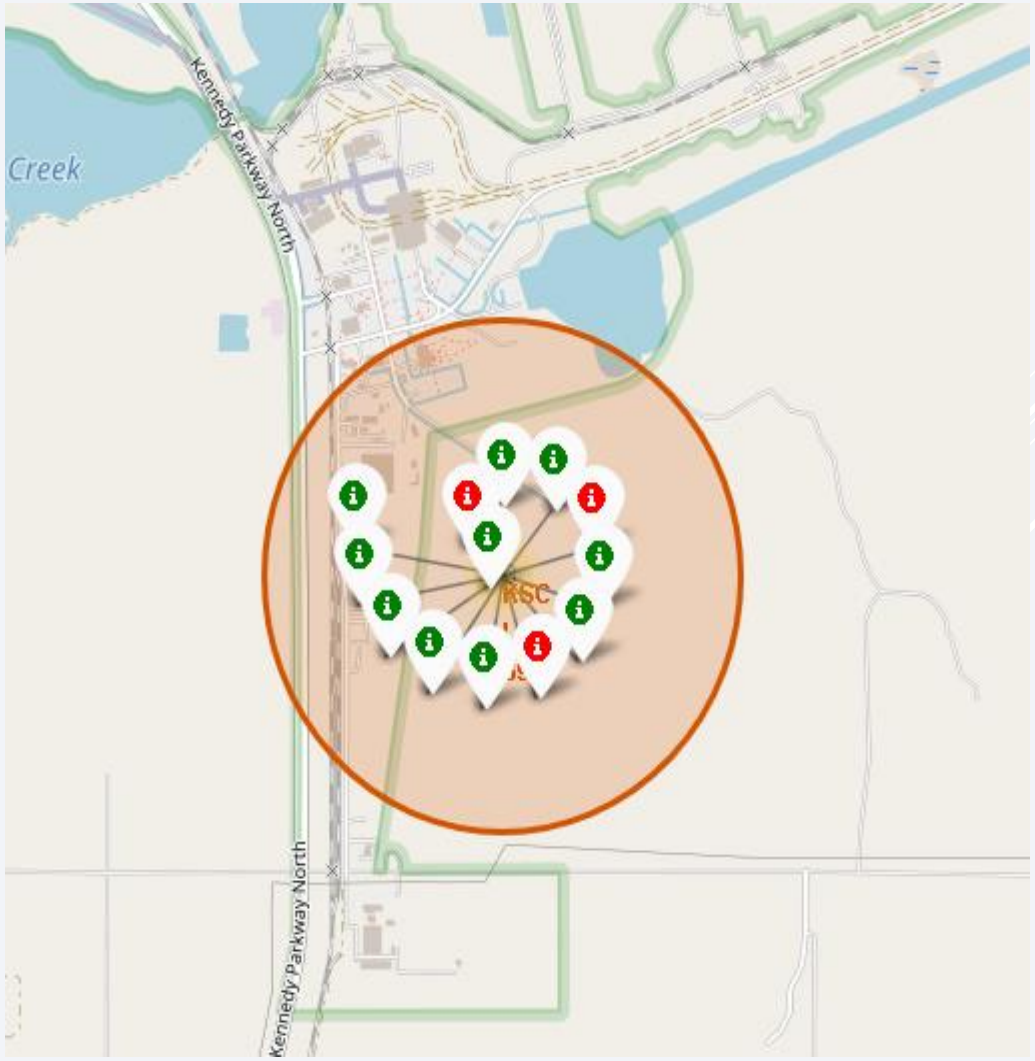
All the 4 launch sites are marked in this map



Successful Launches At Sites

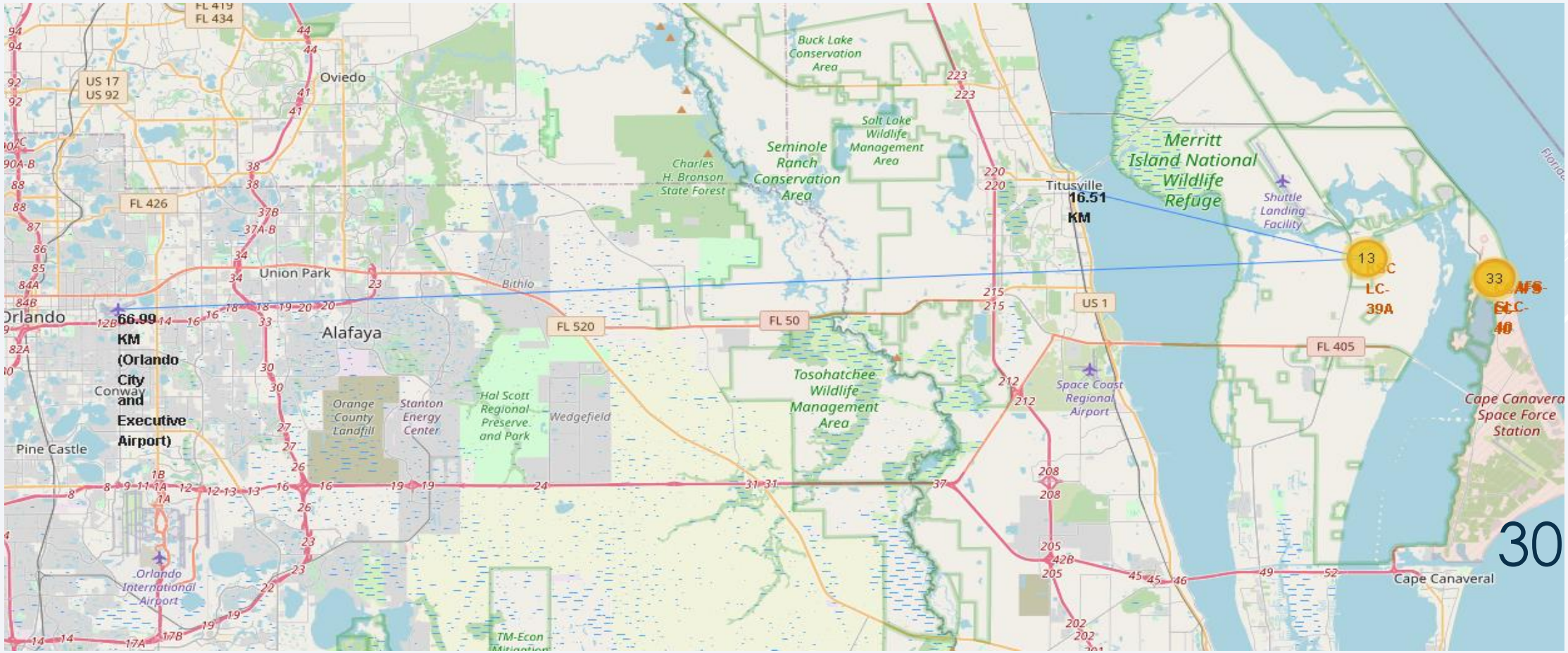
If a launch was successful (class=1), then we use a green marker and if a launch was failed, we use a red marker (class=0) and we make cluster of those points.





Distances Between Launch Site To Its Proximities

We explored two of the proximities which were railway station and orlando excutive airport. Both of the distance are also marked. Through each proximity we concluded that launch site are majorly away from public but nearby shore to get safe environment for landing.



Section 4

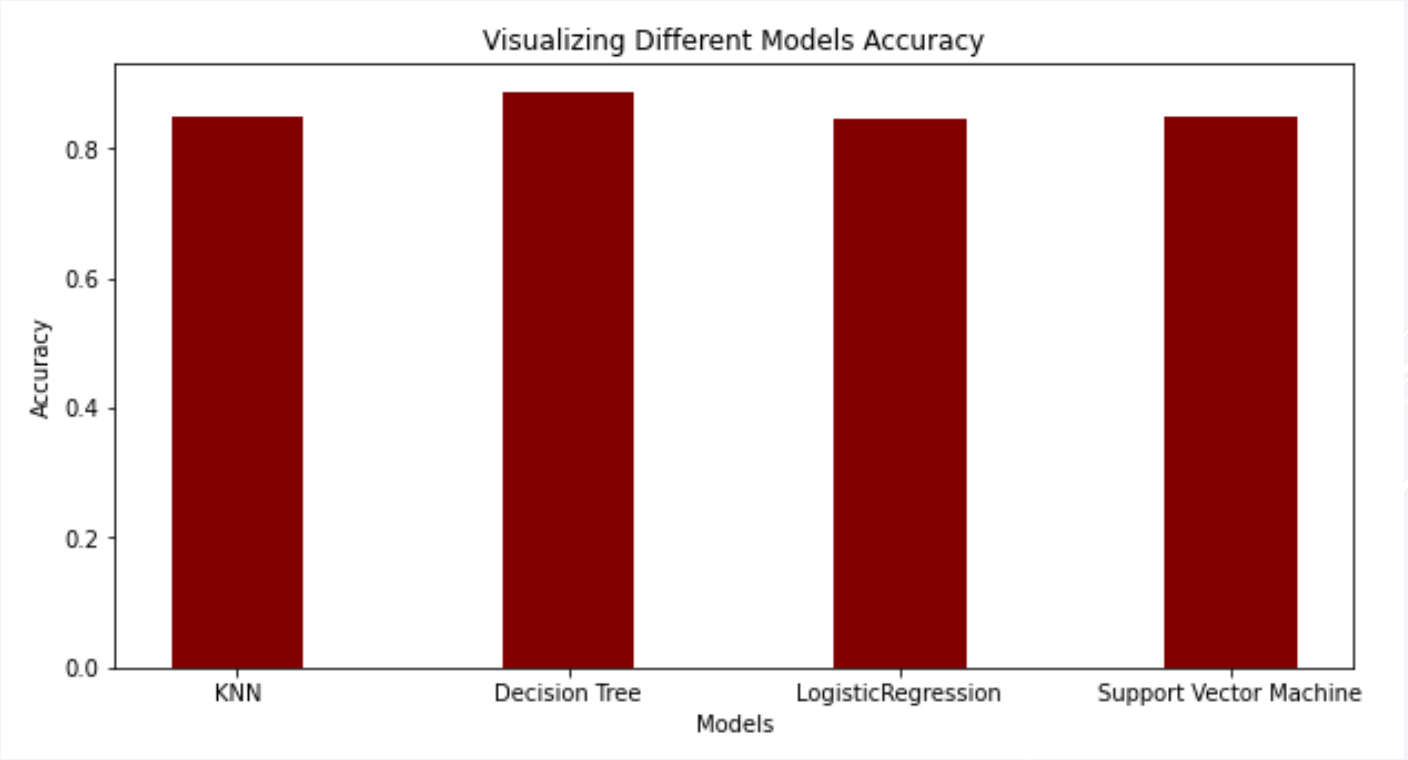
Predictive Analysis (Classification)

Classification Accuracy

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

Find which model has the highest classification accuracy

- Decision Tree have the highest accuracy



Confusion Matrix

In this Matrix, Y axis constitutes of True Outcomes and X axis constitutes of Predicted outcome and majorly the prediction and true outcomes are correct but the major problem is false positives. That means we predicted that it would land successfully but actually it didn't.

Conclusions

- . After training the model we tested it from a different data set and according to the accuracy we chose Decision tree as our model and the accuracy which we got was 0.944 that means its 94.4% chance that the first stage will be successful.
- We can compare our model result to our exploratory analysis result which was purely based on previous outcomes.
- According to exploratory analysis we were having 66.6% chance but our decision tree model analysed some of the elements which we couldn’t analyze in exploratory analysis

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

```
In [23]: print("accuracy: ", tree_cv.score(X_test,Y_test))
accuracy:  0.9444444444444444
```

Thank you!