IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Omar Mousteau
2022-09-07

Outline

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

Executive Summary

- Methodology
- Insights drawn from EDA
- Launch Sites Proximity Analysis
- Dashboard
- Predictive analysis (Classification)

Introduction

- On February 2018 : First Launch of a Falcon 9 Rocket by SpaceX.

Huge revolution in aero spatial industry for two reasons :

- ➢ The price of the rocket : 62M $ ‘only’.
- ➢ The first stage of the rocket can actually be re-used.

If you determine if the first stage will land, you determine the price of the rocket.

- SPACE Y Project :

1. Determine the price on a launch based on SpaceX Information
2. Try to predict if SpaceX will land successfully, using Machine Learning Models and SpaceX public Data sets.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - 2 methods used to get SpaceX data : Data Collection API & Web Scraping
  - Dealing with missing values & Classify successful and unsuccessful landings
- Perform data wrangling
  - Analyzing our data set (missing values, number or successful landings..)
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Build, tune, evaluate classification models that predict the outcome of a Launch.

Data Collection

We used two different methods to collect SpaceX Rocket Launches Data :

- First, we used SpaceX API with the following URL : https://api.spacexdata.com/v4/launches/past
- Then, we used Web Scraping with Requests and BeautifulSoup, analysing html code of SpaceX pages, based on this URL :

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

This data will be used for future analysis and prediction.

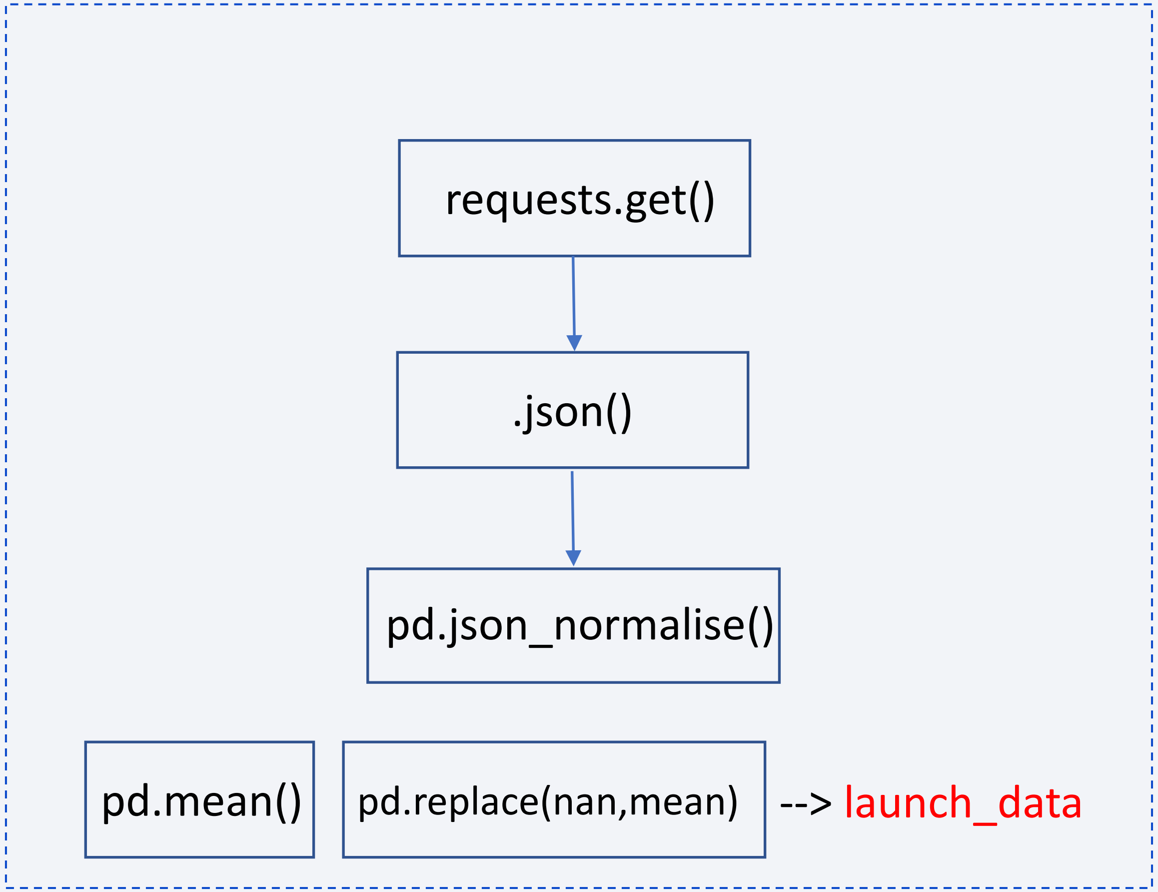
Data Collection – SpaceX API

Get Requests with SpaceX REST API :

- Extraction of the JSON file with a GET request and .json() method
- Transform it into Pandas DataFrame with 'json_normalize' function
- Then with some functions, we extract the Falcon9 data into a new Data Frame (launch_data)
- We give the missing values the mean value for Mayload Mass.

- URL of the notebook :

https://github.com/OmarMousteau/IBM-Data-Science-Capstone-Project/blob/8c59a85b7a466389c613a992ca434ebbd244ac0f/Data_Collection_API.ipynb



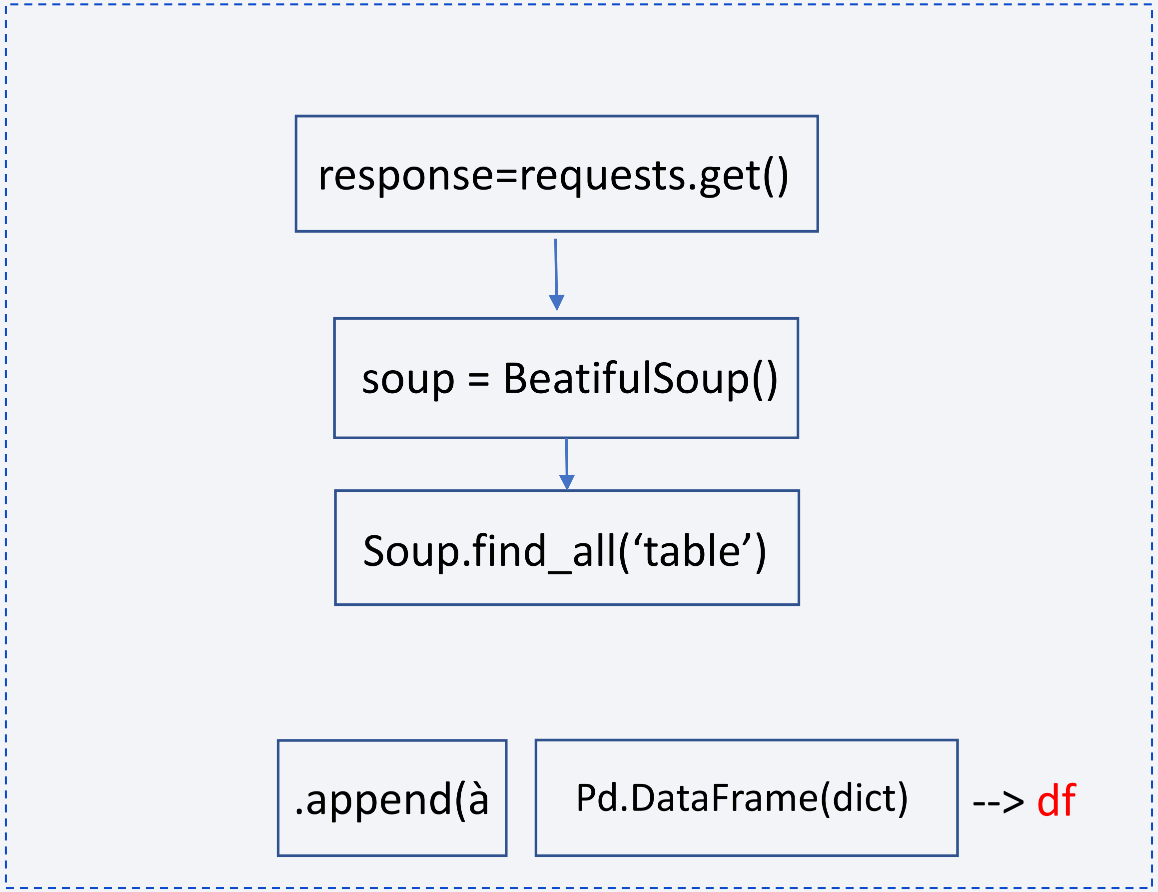
Data Collection - Scraping

Web-Scraping Falcon9 records using BeautifulSoup:

- Extract html table from Wikipedia.
- Parse the table and convert it to a data frame, creating a dictionary and implementing its values.

URL of the notebook :

https://github.com/OmarMousteau/IBM-Data-Science-Capstone-Project/blob/c3a5dee2d4b91d50be8f6b87d594f8019acff9d9/Data-Collection_Scrapping.ipynb



Data Wrangling

- ➢ Identifying the proportion of missing values in each attribute, also which columns are numerical and categorical.
- ➢ We calculate the number of launches on each site, using value_counts() method on Launch Site column.
- ➢ Also, the number and occurrence of each orbit.
- ➢ The number and occurrence of mission outcome per orbit type, defining a set of bad outcomes and good outcomes
- ➢ Finally, we created a landing outcome label from Outcome column (1 for successful landing and 2 for an unsuccessful one).

- URL of the notebook :

  https://github.com/OmarMousteau/IBM-Data-Science-Capstone-Project/blob/33f0a9bfc886aa7cc6b254e34af291dd0f5325ad/Data-Wrangling.ipynb

Data Wrangling

.value_counts()

df.isnull().sum()

df.dtypes

Df['Class']=landing_class

In [8]:
```
# landing_outcomes = values on Outcome column

landing_outcomes= df['Outcome'].value_counts()
landing_outcomes
```

Out[8]:
```
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
False Ocean     2
None ASDS       2
False RTLS      1
Name: Outcome, dtype: int64
```

```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
```

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise

landing_class=[]

l = df['Outcome'].shape[0]

for i in range (l):
    if df['Outcome'].loc[i] in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)

landing_class
```

EDA with Data Visualization

Using Matplotlib and Seaborn, we observed :

- ➢ Relationship between Flight Number and Launch Site
- ➢ Relationship between Payload and Launch Site
- ➢ Relationship between success rate of each orbit type
- ➢ Relationship between Flight number and Orbit type.
- ➢ Relationship between Payload and Orbit Type
- ➢ The launch success yearly trend

- URL of the notebook : https://github.com/OmarMousteau/IBM-Data-Science-Capstone-Project/blob/30c3d1af2587430ccb6b8cf7641e0db942ec470c/EDA-DataViz.ipynb

EDA with SQL

Here are some SQL requests we have made :

- 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 achieved
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- List the total number of successful and failure mission outcomes
- List the names of the booster_versions which have carried the maximum payload mass

…

URL of the notebook : https://github.com/OmarMousteau/IBM-Data-Science-Capstone-Project/blob/33fc2db6dd6b6440d14382e85e444e7bf3b773f3/EDA-SQL.ipynb

Build an Interactive Map with Folium

- TASK 1: Mark all launch sites on a map
- TASK 2: Mark the success/failed launches for each site on the map
- TASK 3: Calculate the distances between a launch site to its proximities

We created Circles on Launch sites, lines between two points on the Folium Map…

By using a colored marker, we can see which site has a good Success Rate.

These objects allow us to visualize spatially the launch maps, the successful and unsuccessful ones.

URL of the notebook : https://github.com/OmarMousteau/IBM-Data-Science-Capstone-Project/blob/020a239e415b1d935b775e2f7d7f74722ed08d85/SiteAnalysis_Folium.ipynb

Build a Dashboard with Plotly Dash

Interactive Dashboard using Plotly Dash, showing :

- A Pie Chart presenting the number of success launches by site.
- A scatter plot chart presenting the correlation between payload mass and the outcome, for different booster version.

- Add the GitHub URL of your completed Plotly Dash lab, as an external reference and peer-review purpose
- URL of the python code : https://github.com/OmarMousteau/IBM-Data-Science-Capstone-Project/blob/2775d66e8f661c164c62e0a715f8ff0fcda5c482/DashboardPlotly.py

Predictive Analysis (Classification)

- Our idea was to predict the Landing Outcome, using our data : Flight Number, Payload Mass, Orbit …
- Separate our data into a train set and a test set with the train_test_split() method.
- We use 4 different classification model : Logistic Regression, SVC, Decision Tree, K Nearest Neighbors.
- We use GridSearchCV to find the best hyperparameters for each model
- We plot the confusion matrix and show the score of each model.

- URL of the notebook : https://github.com/OmarMousteau/IBM-Data-Science-Capstone-Project/blob/41697c727fa82169a7d8a60234f049364874f489/Data_Prediction.ipynb

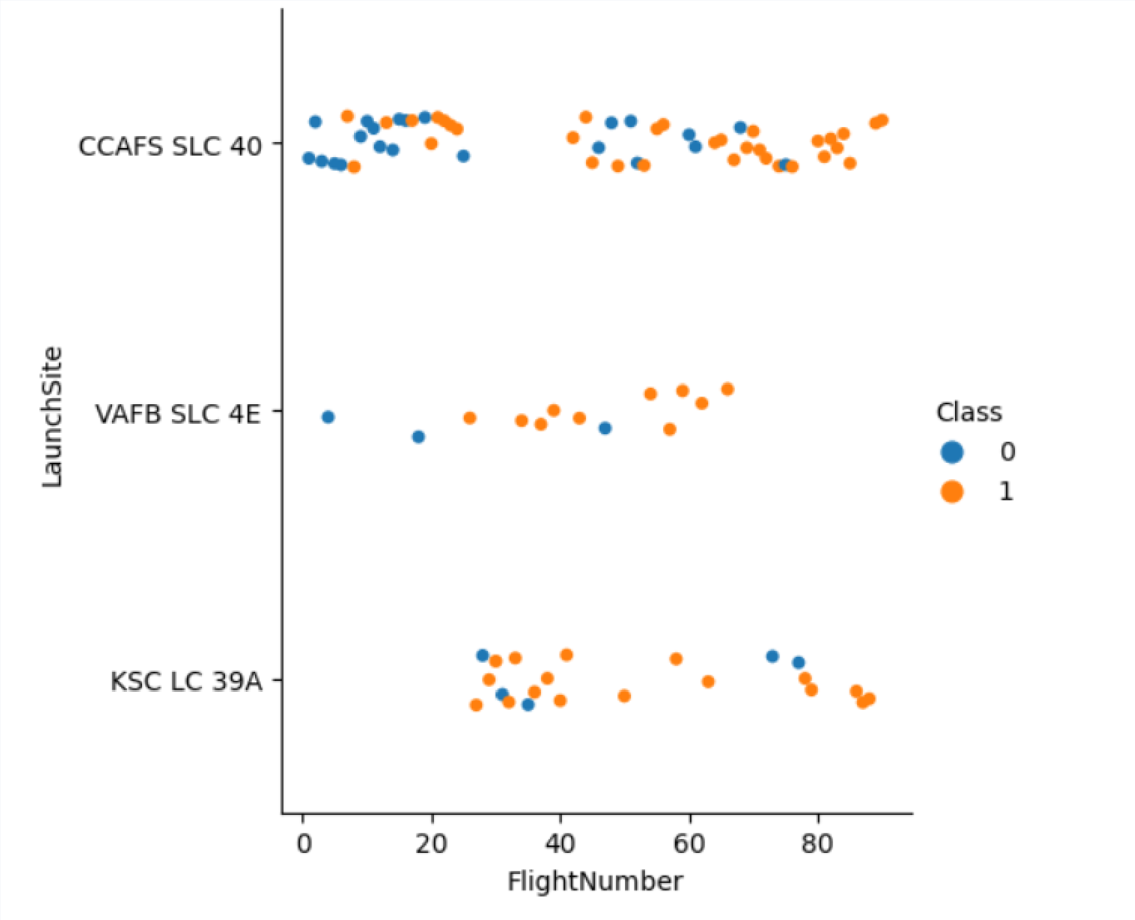
Results

- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

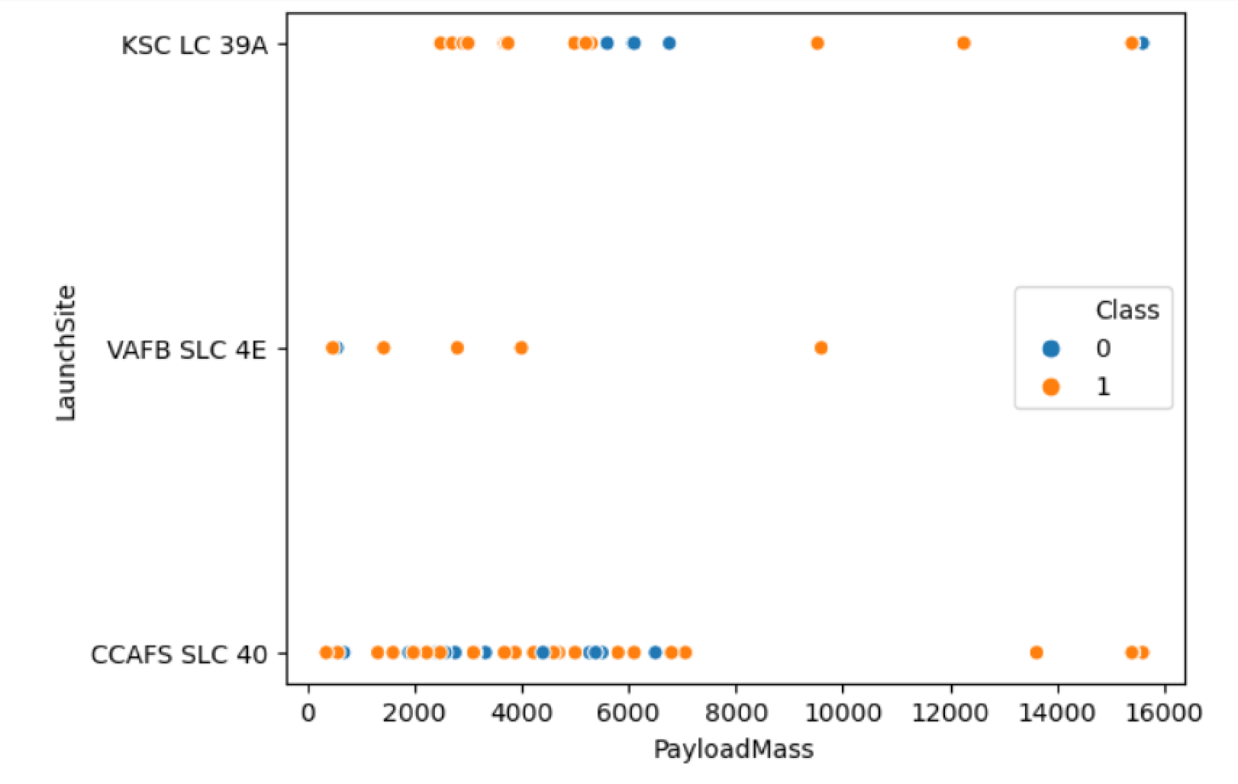
Insights drawn from EDA

Flight Number vs. Launch Site



- We see in Orange the successful landings and in Blue the unsuccessful.
- We see the Flight Number for each flight, with its Launch Site.

Payload vs. Launch Site

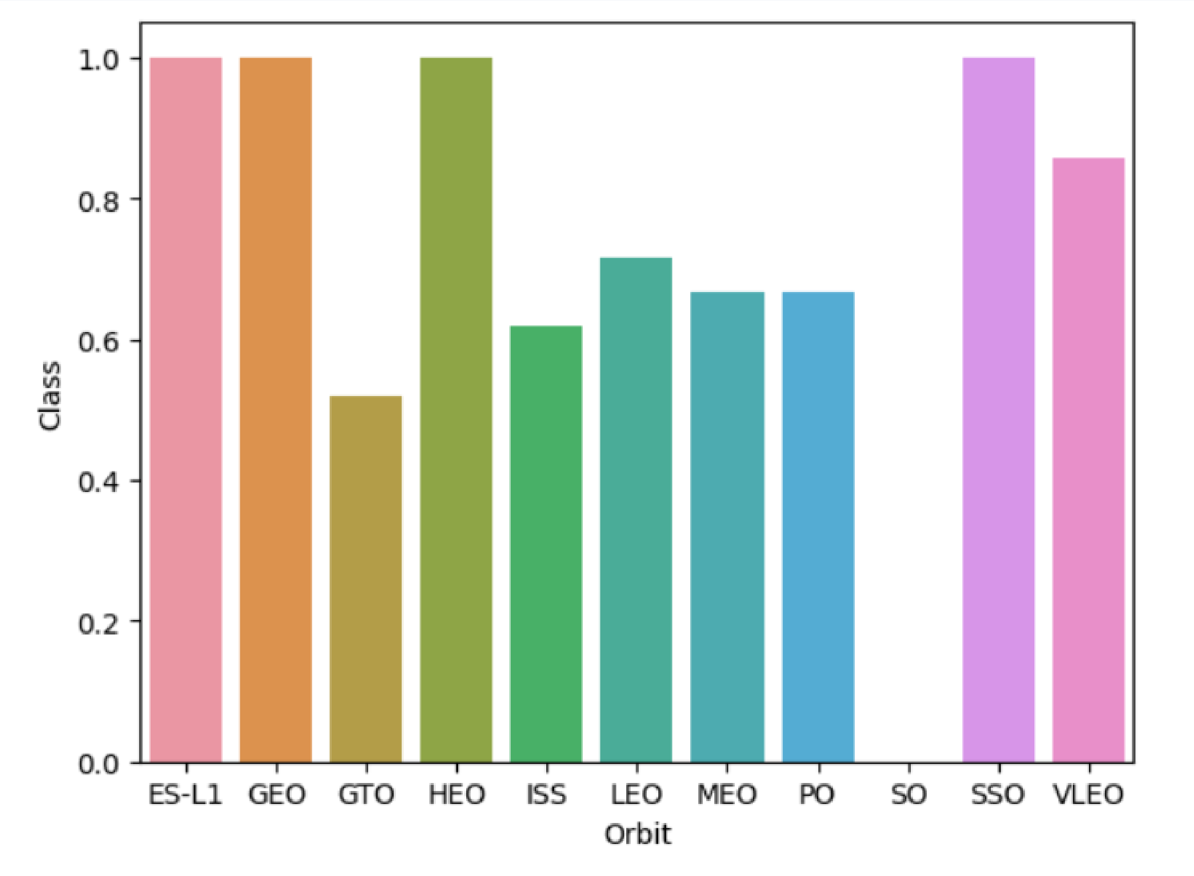


- We see in Orange the successful landings and in Blue the unsuccessful.
- We see the Payload Mass for each flight, with its Launch Site.

Success Rate vs. Orbit Type

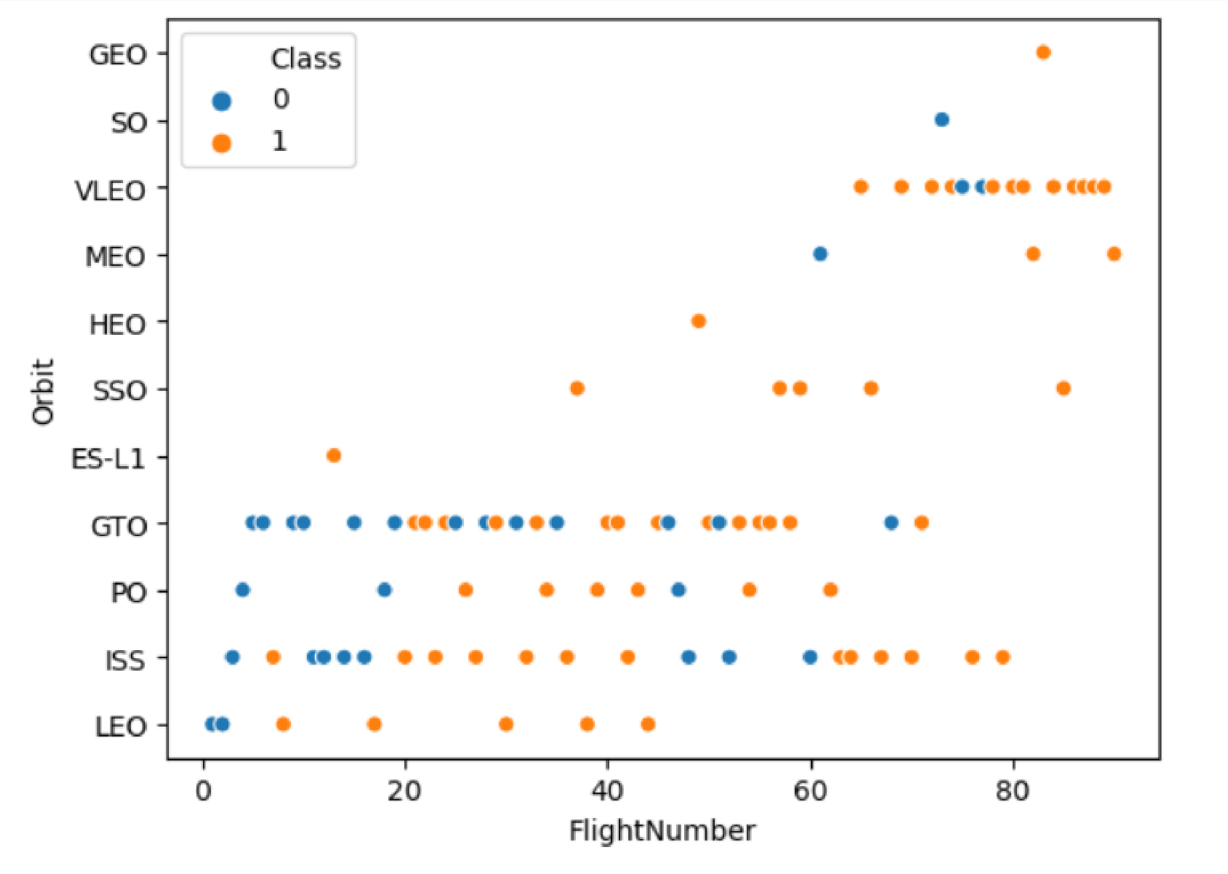
- We see the success rate depending on the orbit Type in this bar chart

- The best obits seem to be ES-L1 ; GEO ; HEO ; SSO



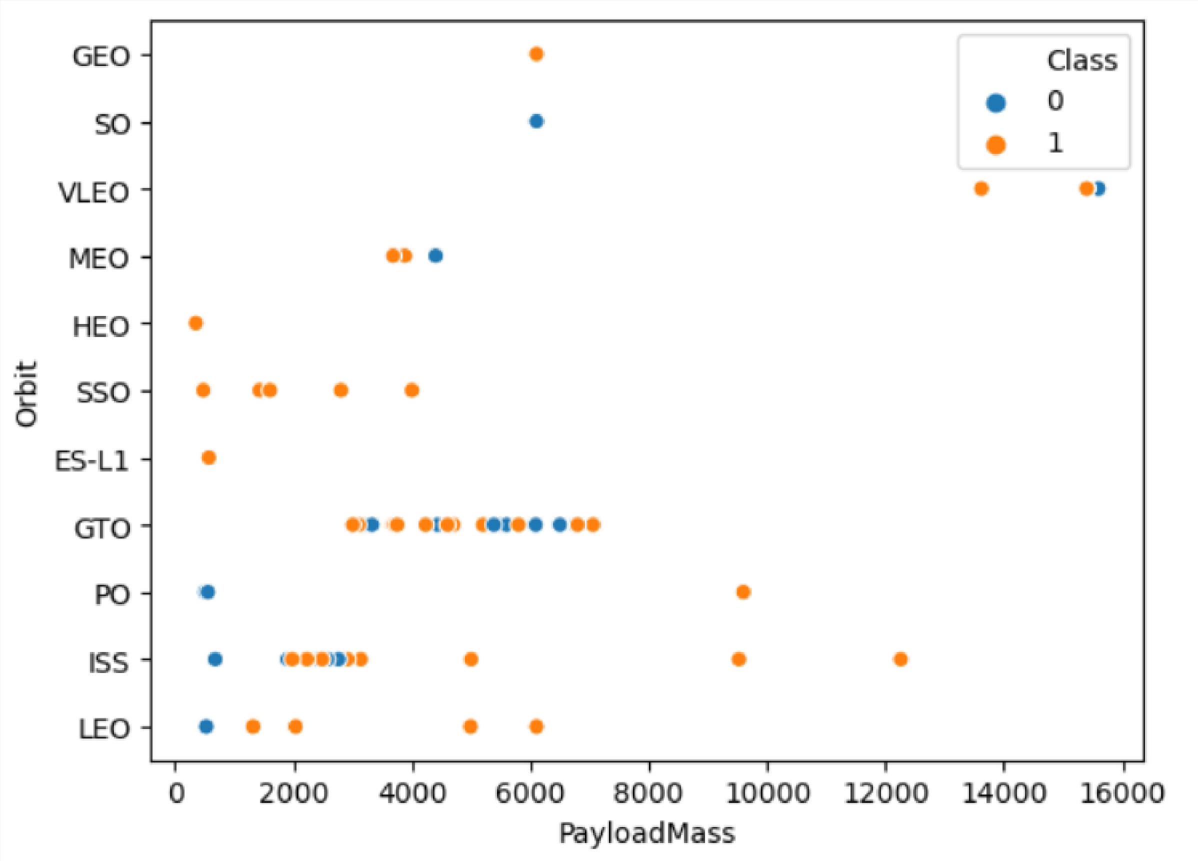
Flight Number vs. Orbit Type

- We see in Orange the successful landings and in Blue the unsuccessful.
- We see Flight Number for each flight, with its Orbit.



Payload vs. Orbit Type

- We see Payload Mass for each flight, with its Orbit.
- With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS.
- However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.



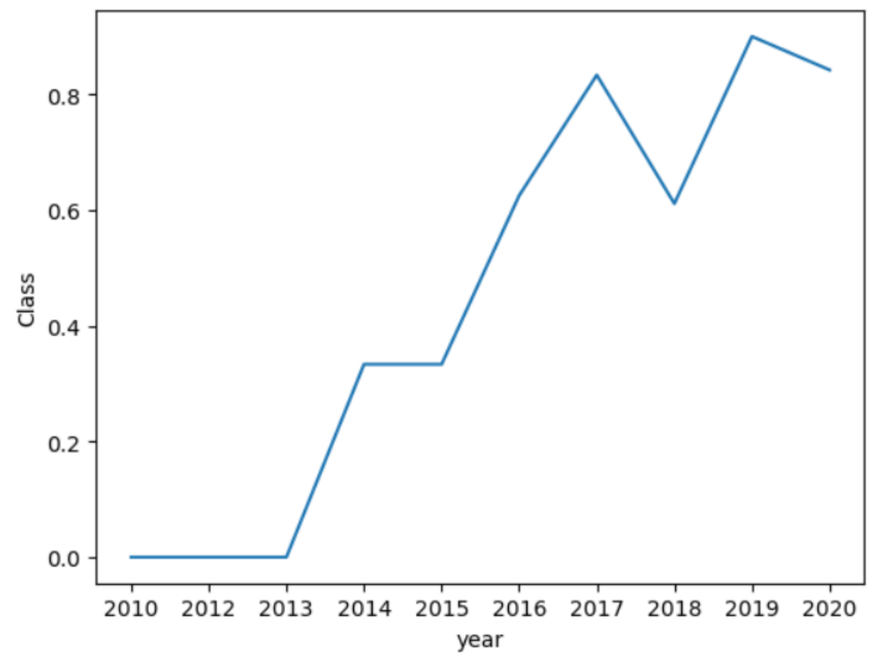
Launch Success Yearly Trend

```
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
year_data=  Extract_year()

df['year']=pd.DataFrame(year_data)
df_year=df[['year','Class']]

df_year_mean=df_year.groupby('year').mean()
df_year_mean.reset_index(inplace=True)

sns.lineplot(data=df_year_mean, x='year', y='Class')
```



- In 2013, the success rate has started to increase, and since 2016, it has not been under 0.6

All Launch Site Names

- We show all the different values of the Launch Site column in the Table SPACEXTBL

```
In [10]: %sql SELECT DISTINCT(LAUNCH_SITE) FROM SPACEXTBL

 * sqlite:///my_data1.db
Done.
```

Out[10]:

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

Launch Site Names Begin with 'CCA'

[8]:
```
%sql SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' limit 5
```

```
 * sqlite:///my_data1.db
Done.
```

[8]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- We calculate the sum of all payload mass for NASA (CRS) Customer :

```
In [13]: %sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE CUSTOMER = 'NASA (CRS)'

 * sqlite:///my_data1.db
Done.
```

Out[13]:

SUM(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

- We calculate the average value of payload mass, for F9 v1.1 rockets.

```
In [23]: %sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE BOOSTER_VERSION= 'F9 v1.1'

 * sqlite:///my_data1.db
Done.
```

Out[23]:

AVG(PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

```
%sql select min(DATE) from spacextbl where landing__outcome = 'Success (ground pad)'
```

Issue with the Landing Outcome Column

```
 * sqlite:///my_data1.db
(sqlite3.OperationalError) no such column: landing__outcome
[SQL: select min(DATE) from spacextbl where landing__outcome = 'Success (ground pad)']
(Background on this error at: http://sqlalche.me/e/13/e3q8)
```

Successful Drone Ship Landing with Payload between 4000 and 6000

- We show the Booster Name for a Payload value between 4000 and 6000.

In [87]:
```
%%sql

select BOOSTER_VERSION from SPACEXTBL WHERE MISSION_OUTCOME='Success' AND PAYLOAD_MASS__KG_ Between 4000 and 6000
```

Out[87]:

Booster_Version
F9 v1.1
F9 v1.1 B1011
F9 v1.1 B1014
F9 v1.1 B1016
F9 FT B1020
F9 FT B1022
F9 FT B1026
F9 FT B1030
F9 FT B1021.2
F9 FT B1032.1
F9 B4 B1040.1
F9 FT B1031.2
F9 FT B1032.2
F9 B4 B1040.2
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

Total Number of Successful and Failure Mission Outcomes

- We count the number of raws with a successful mission.

The first output is this number, and the second one is the total number of missions minus the number of successful mission.

In [93]:
```
%sql SELECT COUNT(*) as Success,101-COUNT(*) as Fail FROM SPACEXTBL where MISSION_OUTCOME='Success'
```

```
 * sqlite:///my_data1.db
Done.
```

Out[93]:

Success	Fail
98	3

Boosters Carried Maximum Payload

- We use a sub-query to find the maximum Payload Mass.

```
In [96]: %%sql SELECT BOOSTER_VERSION from SPACEXTBL

         where PAYLOAD_MASS__KG_= (select max(PAYLOAD_MASS__KG_) from SPACEXTBL)

 * sqlite:///my_data1.db
Done.
```

Out[96]:

Booster_Version
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

```
%%sql SELECT landing__outcome, booster_version, launch_site FROM spacextbl
WHERE landing__outcome = 'Failure (drone ship)'  AND
DATE BETWEEN '2015-01-01' AND '2015-12-3'
```

Issue with the Landing Outcome Column

```
 * sqlite:///my_data1.db
(sqlite3.OperationalError) no such column: landing__outcome
[SQL: SELECT landing__outcome, booster_version, launch_site FROM spacextbl WHERE landing__outcome = 'Failure (drone ship)'  AND DATE BETWEEN '201
5-01-01' AND '2015-12-3']
(Background on this error at: http://sqlalche.me/e/13/e3q8)
```

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

```
%%sql SELECT landing__outcome, COUNT(landing__outcome) FROM spacextbl
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY landing__outcome  ORDER BY count(landing__outcome) desc
```

Issue with the Landing Outcome Column

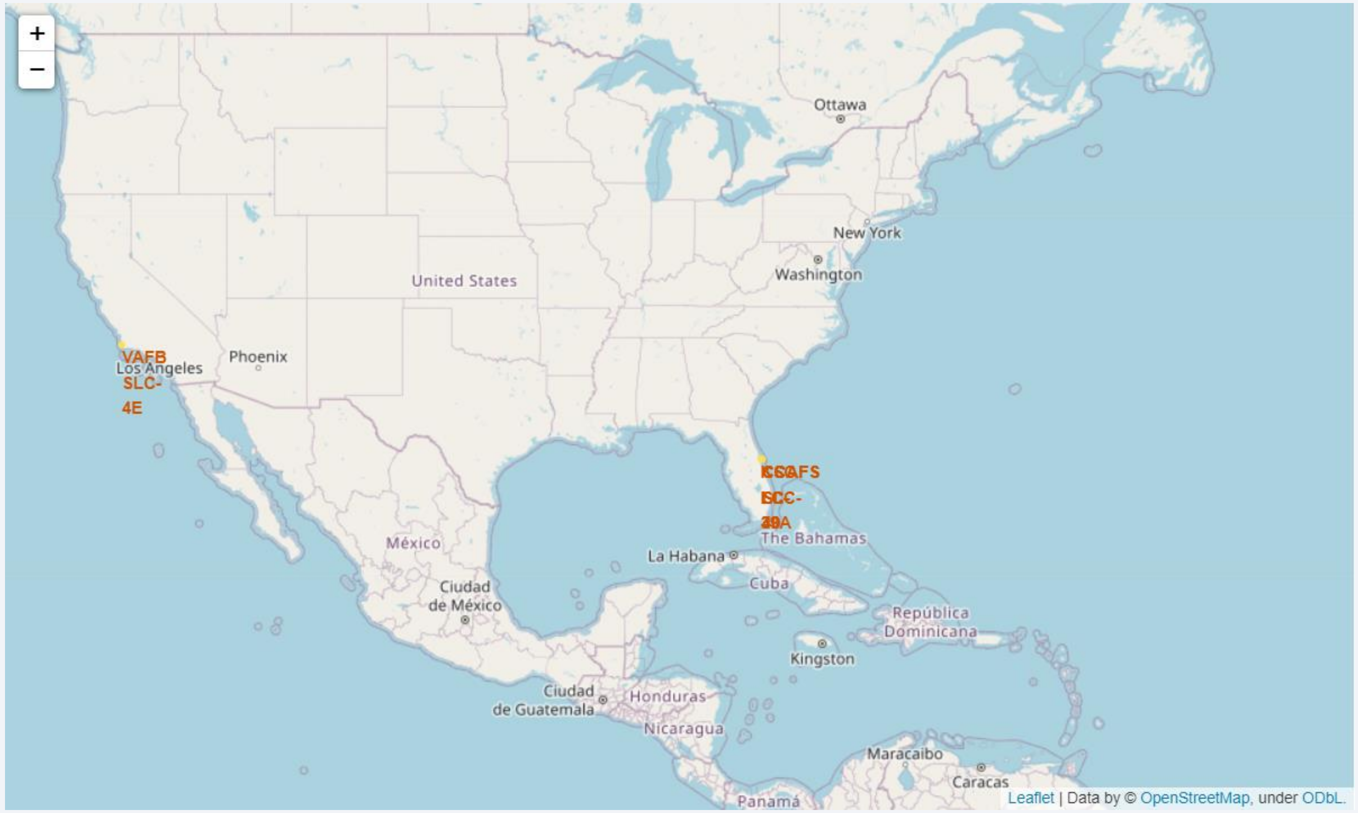
```
 * sqlite:///my_data1.db
(sqlite3.OperationalError) no such column: landing__outcome
[SQL: SELECT landing__outcome, COUNT(landing__outcome) FROM spacextbl
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY landing__outcome  ORDER BY count(landing__outcome) desc]
(Background on this error at: http://sqlalche.me/e/13/e3q8)
```

Section 3

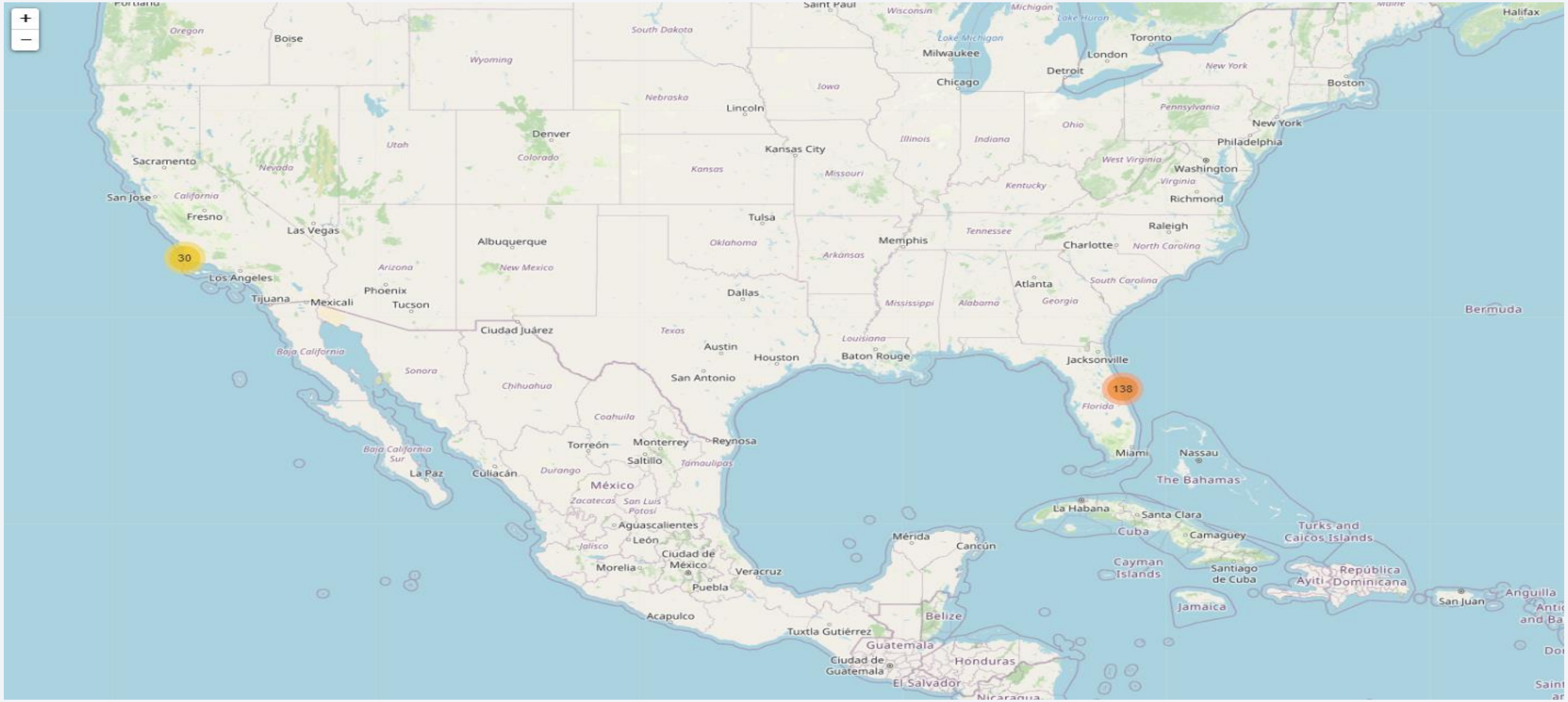
Launch Sites Proximities Analysis

Launch Sites Location

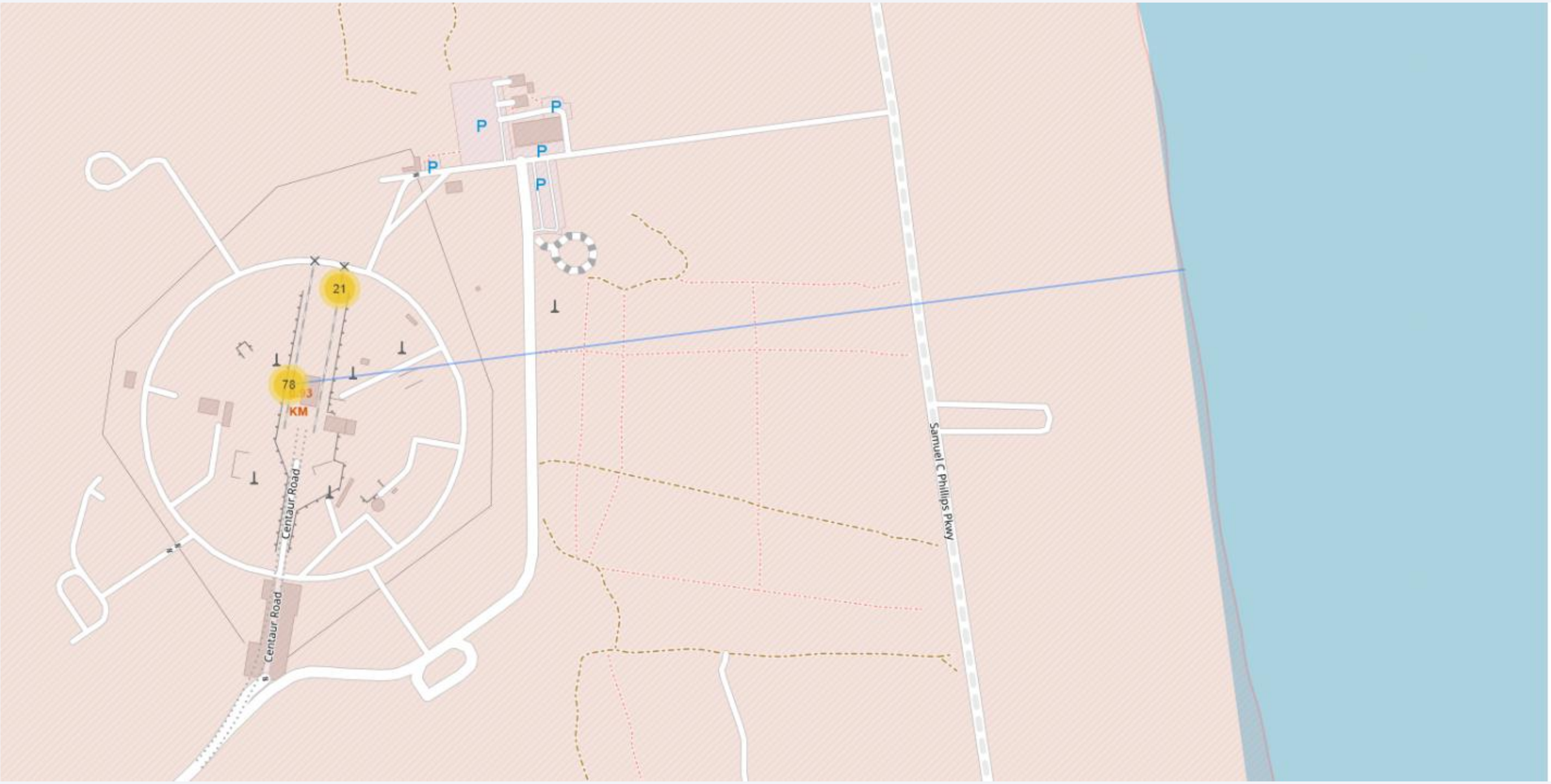
- We generate a marker for each site location :



Success/failed launches for each site on the map



Distances between a launch site to its proximities

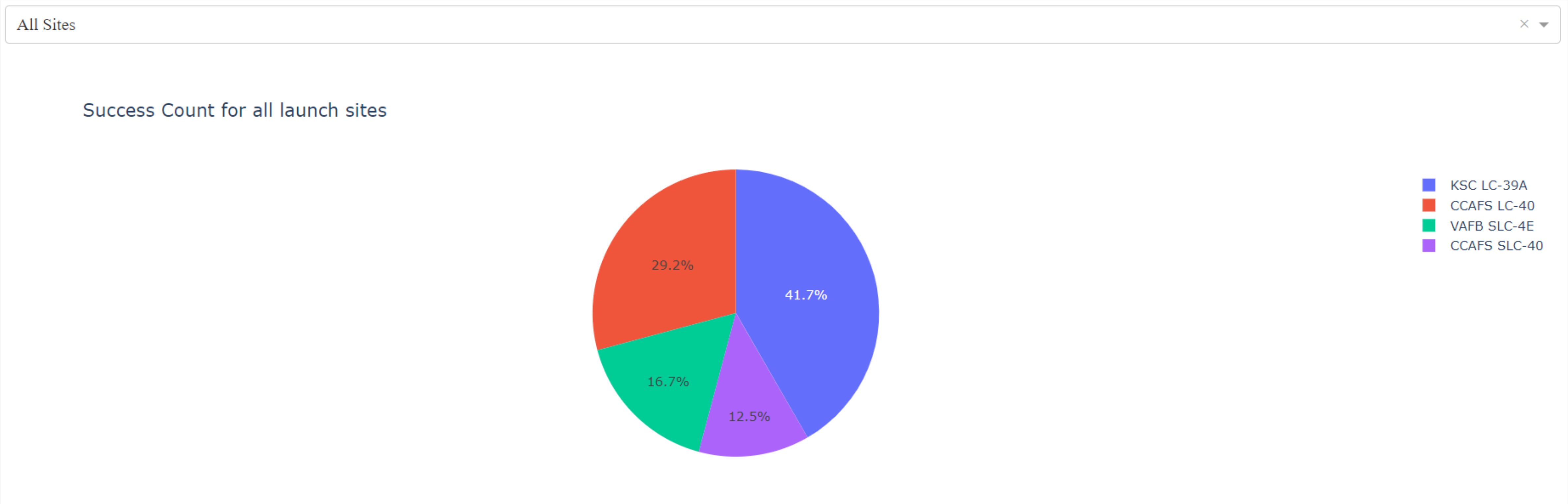


Section 4

Build a Dashboard with Plotly Dash

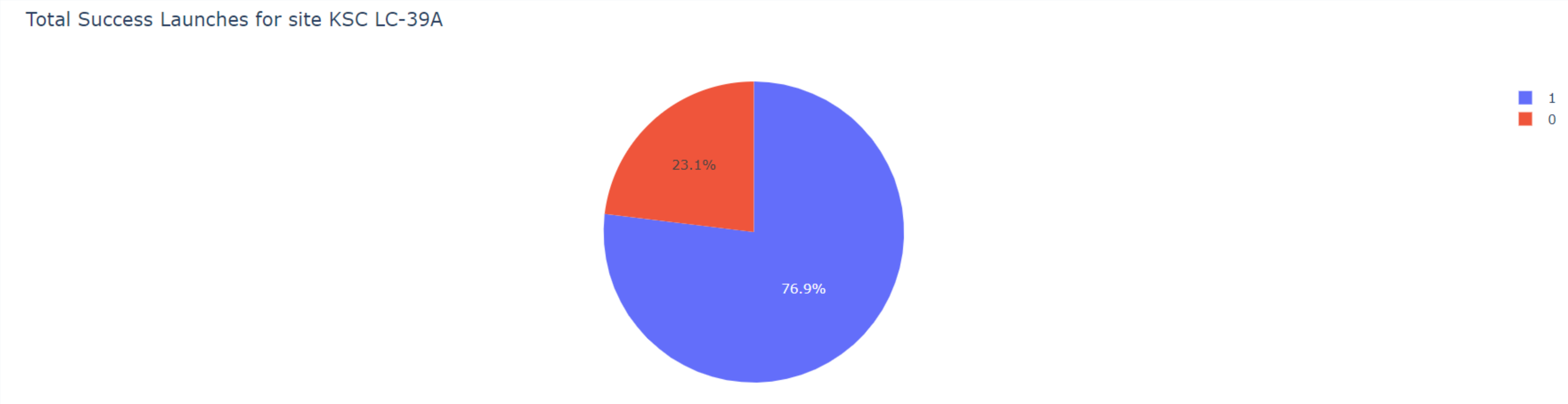
Success Count for all launch Sites

- We count successful landings for each sites :

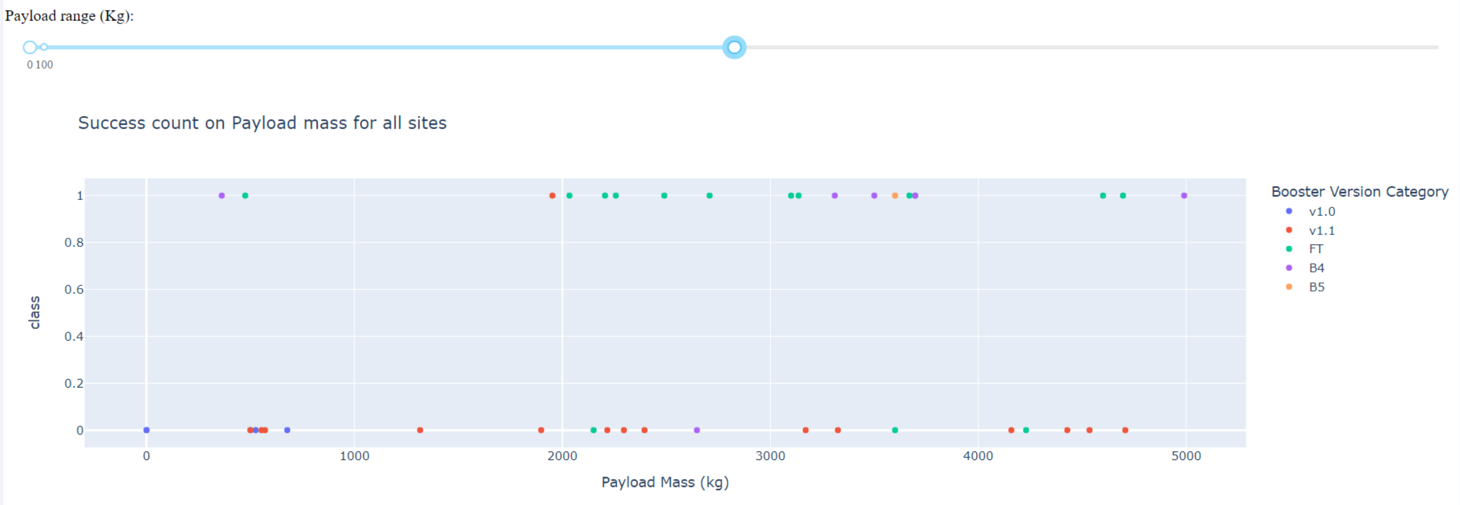


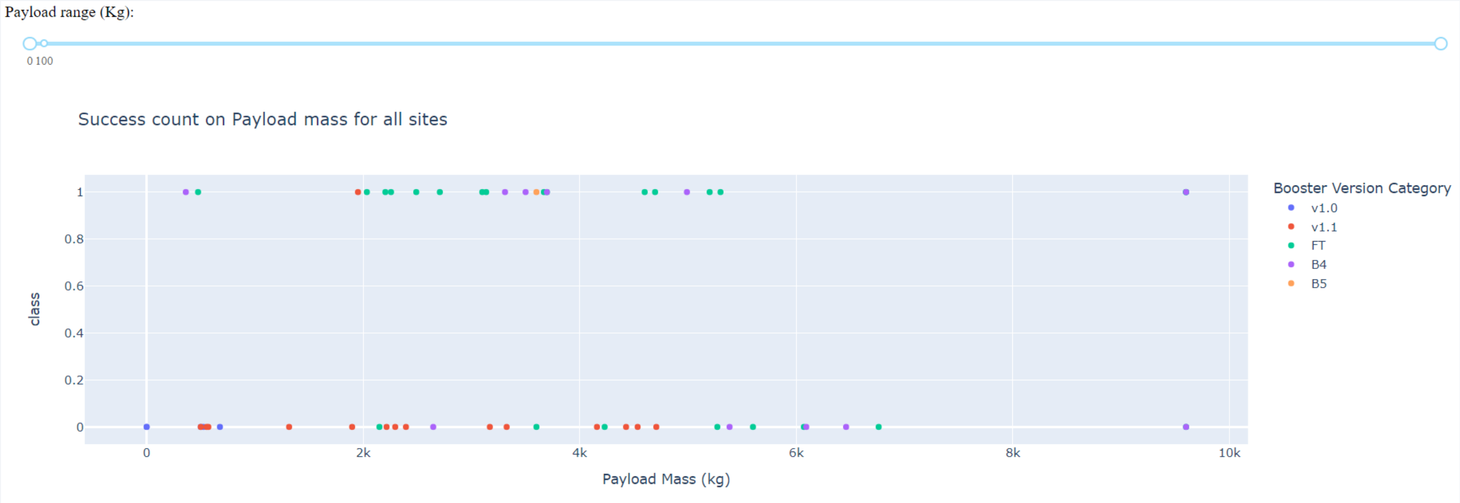
Success Count for the best Launch Site

- The site with the best success rate is KSC LC-39A :



Payload vs Launch Outcome for different payload range

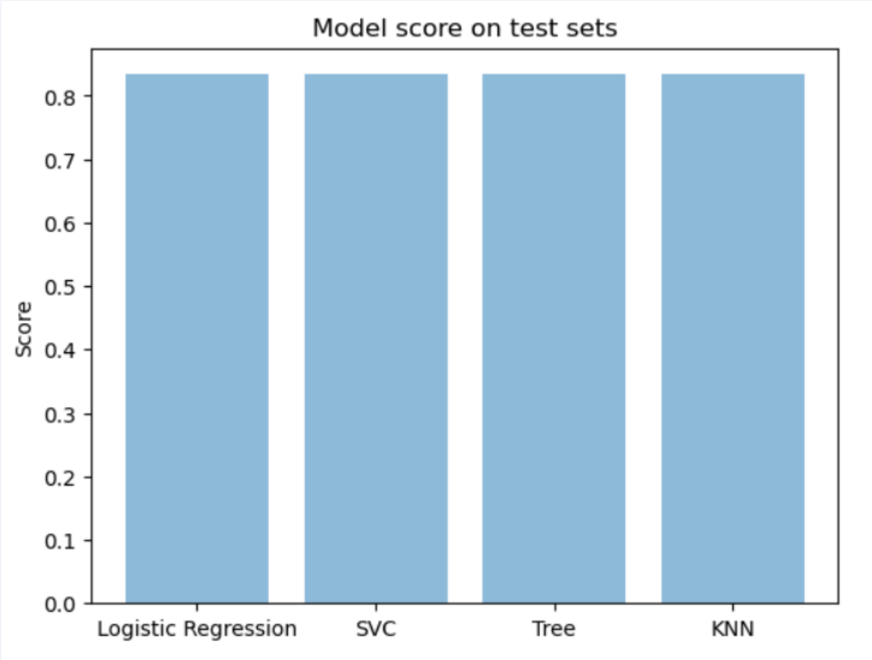




Section 5

Predictive Analysis (Classification)

Classification Accuracy




```
[36]: print('Logistic Regression score ->      test :', opt_lr.score(X_test,Y_test),';          train :', opt_lr.score(X_train,Y_train))
      print('Svc score ->      test :', opt_svm.score(X_test,Y_test),';          train :', opt_svm.score(X_train,Y_train))
      print('Tree score ->      test :', opt_tree.score(X_test,Y_test),';          train :', opt_tree.score(X_train,Y_train))
      print('Knn score ->      test :', opt_knn.score(X_test,Y_test),';          train :', opt_knn.score(X_train,Y_train))

      #The best model seem to be SVC!
```

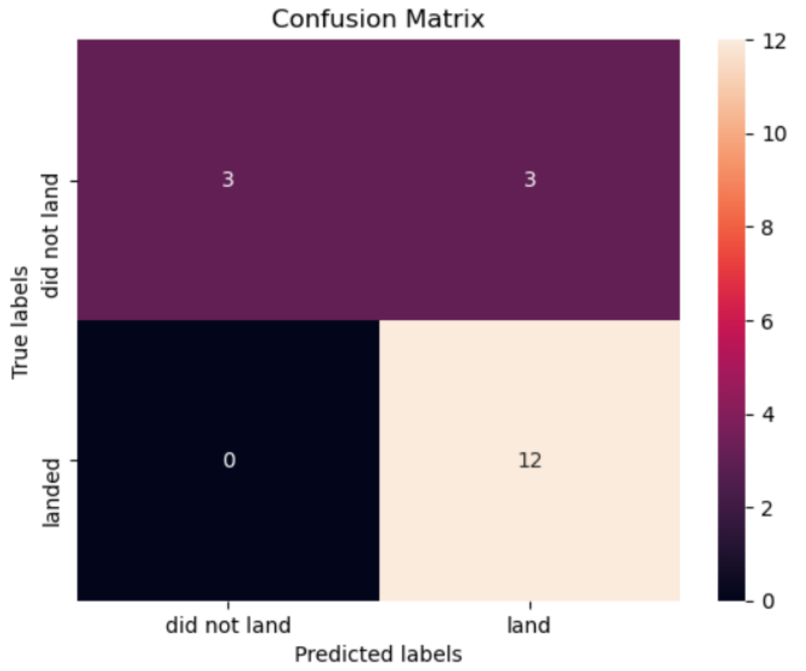
```
Logistic Regression score ->      test : 0.8333333333333334 ;          train : 0.875
Svc score ->      test : 0.8333333333333334 ;          train : 0.8888888888888888
Tree score ->      test : 0.8333333333333334 ;          train : 0.8611111111111112
Knn score ->      test : 0.8333333333333334 ;          train : 0.8472222222222222
```

Confusion Matrix

- This is the confusion matrix for the best model :

SVC with the best hyperparameters found with GridSearchCV()

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



Conclusions

- The larger the flight amount at a launch site, the most important the success rate at a launch site.
- Orbits ES-L1, GEO, HEO, SSO had the most success rate.
- With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS orbit.
- KSC LC-39A had the most successful launches of any sites.
- The SVC model is the best machine learning algorithm for this situation.

Thank you!