IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Cleiton Otavio da Exaltação Rocha
25/12/2021

Outline

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

Executive Summary

- Summary of methodologies
- Summary of all results

Introduction

- Project background and context
- Problems you want to find answers

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Describe how data was collected
- Perform data wrangling
  - Describe how data was processed
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

- Data were collected through two processes: 1) Webscrapping, where in this case information from a website is collected, stored and later analyzed and 2) through an API, which in short is a program that bridges the gap between the data request and its source.

1 – Webscrapping

2 – API

Data Collection – SpaceX API

- A request of type 'GET' is made to get the information back in a JSON format. After that, the JSON is transformed into a dataframe in Pandas format, ready for analysis
- https://github.com/CleitonOERocha/IBM_DS_Final_Project/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

1 – Link:
spacex_url="https://api.spacexdata.com/v4/launches/past"

2 – GET Requistion:
response = requests.get(spacex_url)

3 – json_normalize:
data = pd.json_normalize(response.json())

Data Collection - Scraping

- Present your web scraping process using key phrases and flowcharts

- https://github.com/CleitonOERocha/IBM_DS_Final_Project/blob/main/jupyter-labs-webscraping.ipynb

1 – Link:
static_url="https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"

2 – GET Requisition:
response = requests.get(static_url)

3 – Transform in list:
soup = BeautifulSoup(response.text, "html.parser")

4 – Transform in df:
html_tables = soup.find_all('table')

Data Wrangling

- The first step is to load the data using the Pandas 'read_csv' function. After that, some information is extracted from the dataset, namely: 1) the percentage of missing values in each column was calculated. 2) the number of launches at each site was calculated 3) the number and occurrence of the mission result by orbit type was calculated and 4) a landing result label was created in the 'Result' column
- https://github.com/CleitonOERocha/IBM_DS_Final_Project/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

1 – Link:
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_1.csv")

2 – NA Count:
df.isnull().sum()/df.count()*100

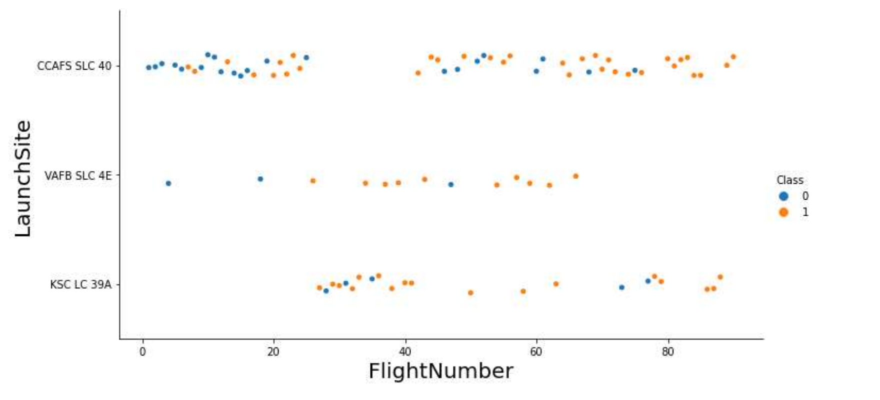
3 – Orbit count:
df["Orbit"].value_counts()

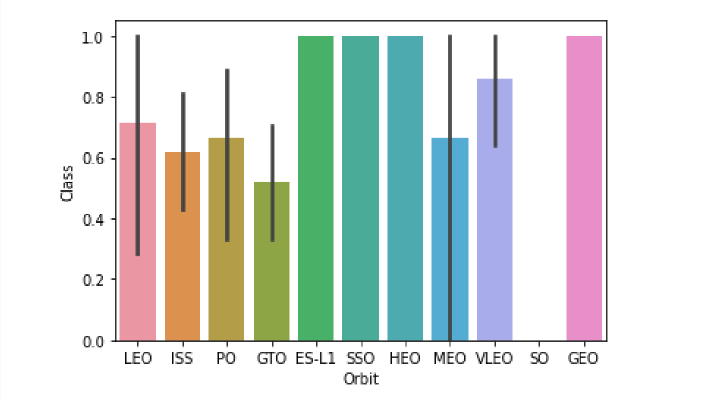
4 – Outcome count:
landing_outcomes = df["Outcome"].value_counts()

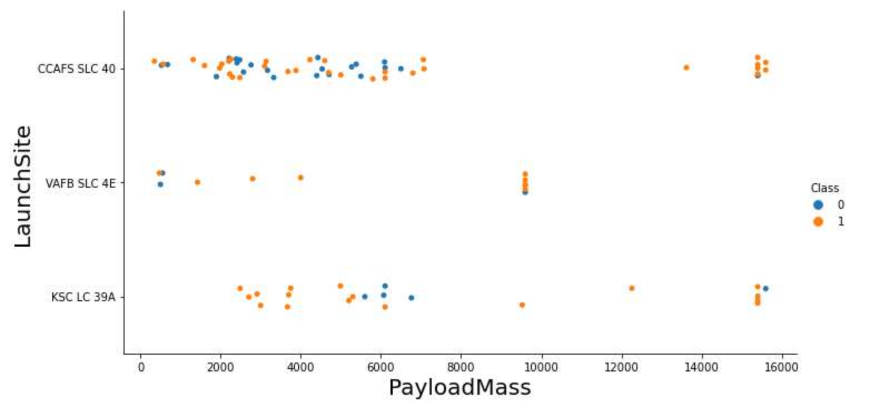
5 – Outcome Condition:
landing_class = np.where(df["Outcome"] == bad_outcomes, 0, 1)

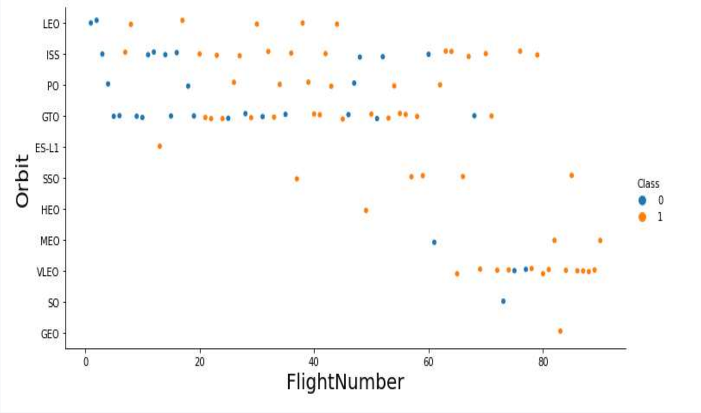
EDA with Data Visualization

- The generated graphics serve to understand more quickly some relevant information about the rockets, such as mission success, mass loss, etc.
- https://github.com/CleitonOERocha/IBM_DS_Final_Project/blob/main/jupyter-labs-eda-dataviz.ipynb









EDA with SQL

- The queries performed were to select variables, group, summarize, filter, among others; Queries are performed using SQL queries through the pandas sql package.
- https://github.com/CleitonOERocha/IBM_DS_Final_Project/blob/main/jupyter-labs-eda-sql-coursera.ipynb

SELECT

WHERE

GROUP BY

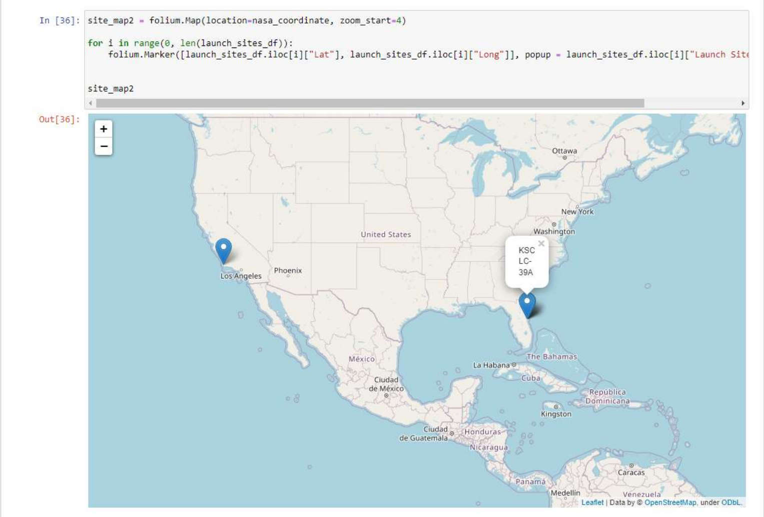
SUM

COUNT

AVERAGE

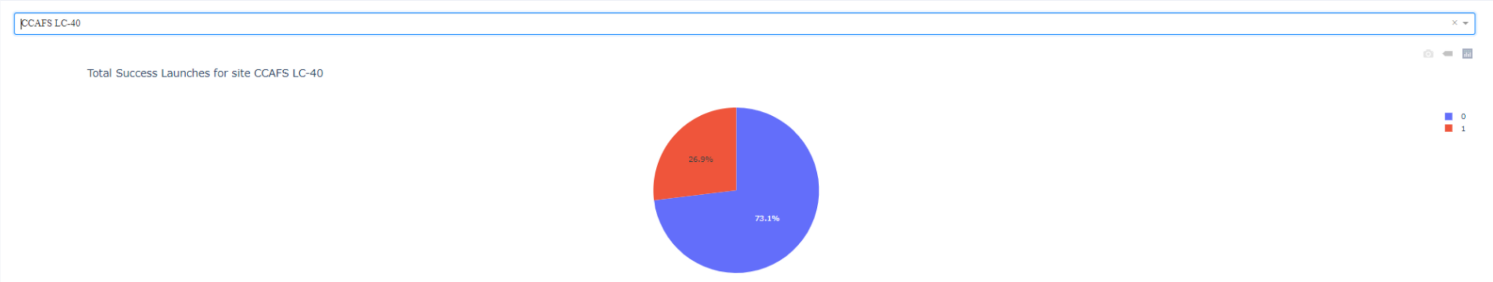
Build an Interactive Map with Folium

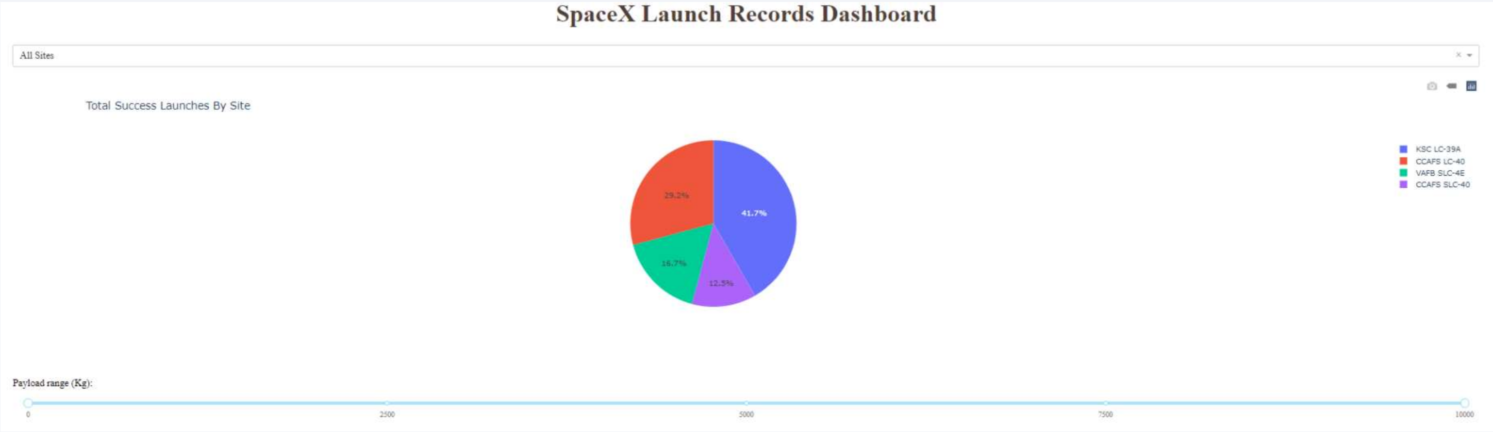
- The points on the map are for rocket launch locations, through the map you can see where these points are and where they are close to.
- https://github.com/CleitonOERocha/IBM_DS_Final_Project/blob/main/lab_jupyter_launch_site_location%20(1).ipynb



Build a Dashboard with Plotly Dash

- The purpose of the dashboard is to observe the behavior of the rockets over time, as well as to see their statistics to understand if the objectives were achieved over time.
- https://github.com/CleitonOERocha/IBM_DS_Final_Project





Predictive Analysis (Classification)

- The models were built from the sklearn library, each model is built and its results tested to see if the model has good accuracy. The best model is the one with the best accuracy.
- https://github.com/CleitonOERocha/IBM_DS_Final_Project/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

1 – Load dataframe:
data = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_2.csv")

2 – Split in train and test:
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state = 2)

3 – Model:
logreg_cv = GridSearchCV(lr, parameters, cv = 10)

4 – Fit model:
logreg_cv.fit(X_train, Y_train)

5 – Accuracy:
print("Tuned Logistic Regression Parameter: {}".format(logreg_cv.best_params_))
print("Tuned Logistic Regression Accuracy: {}".format(logreg_cv.best_score_))

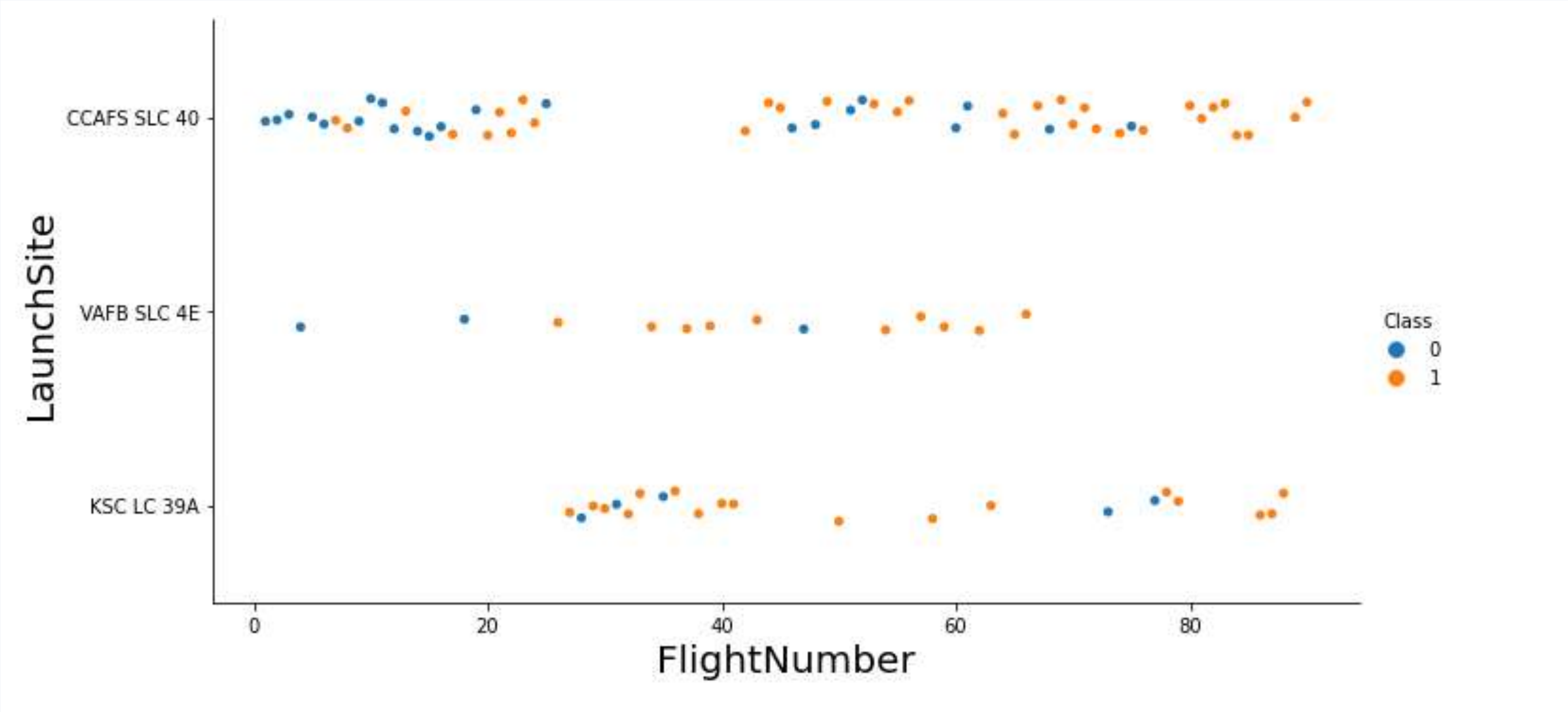
Results

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

Section 2

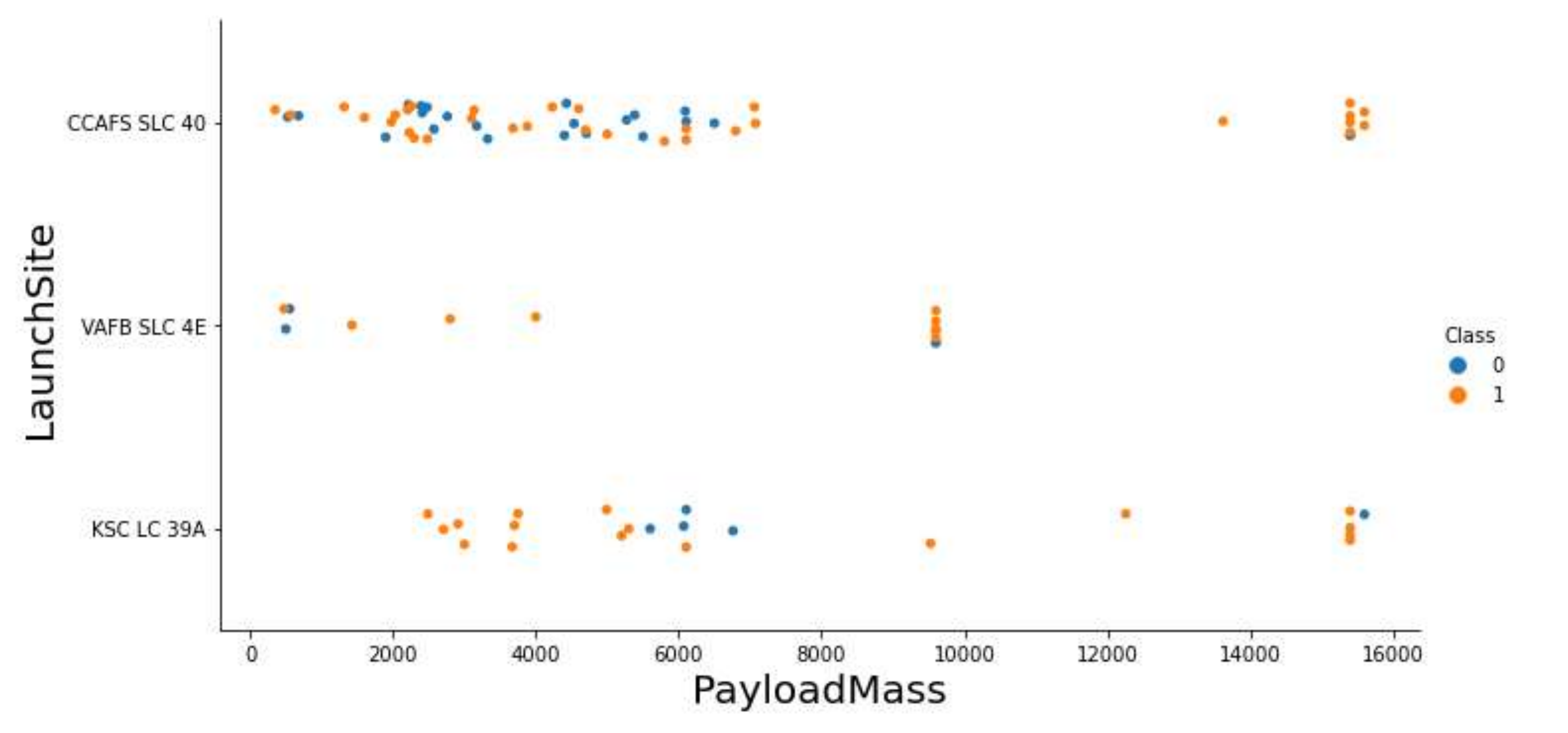
Insights drawn from EDA

Flight Number vs. Launch Site



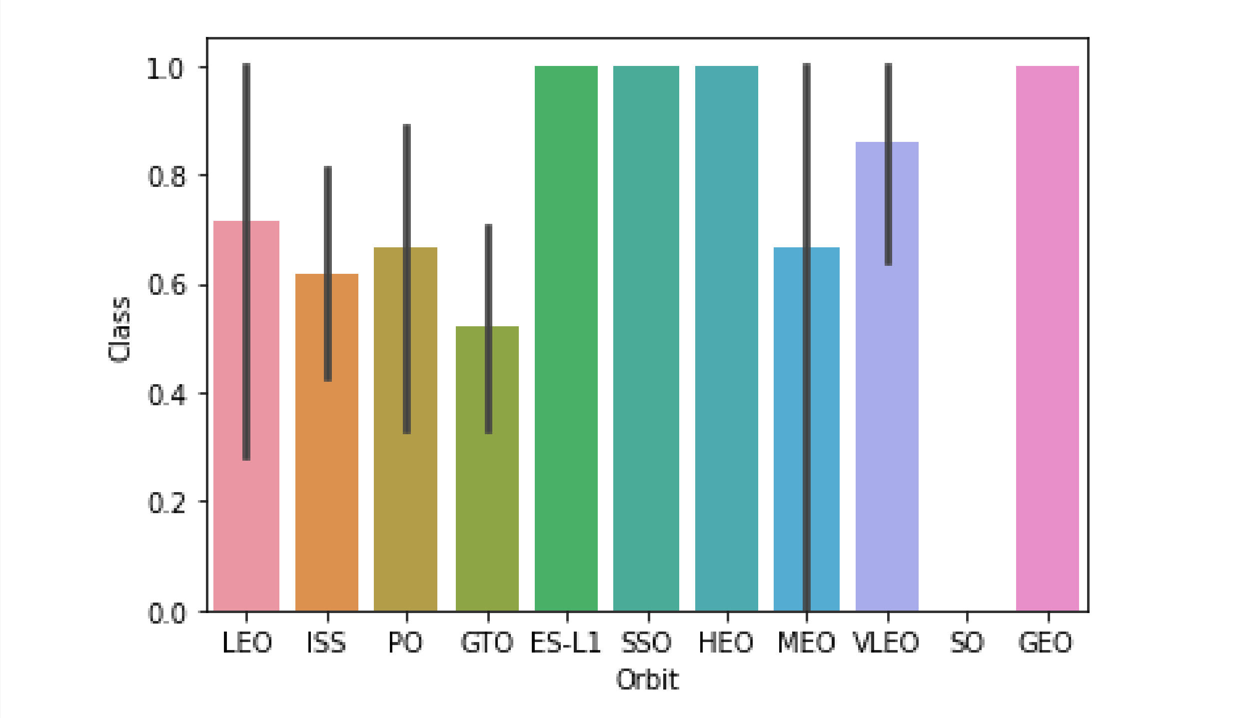
- the graph shows a relationship between locations and number of releases

Payload vs. Launch Site



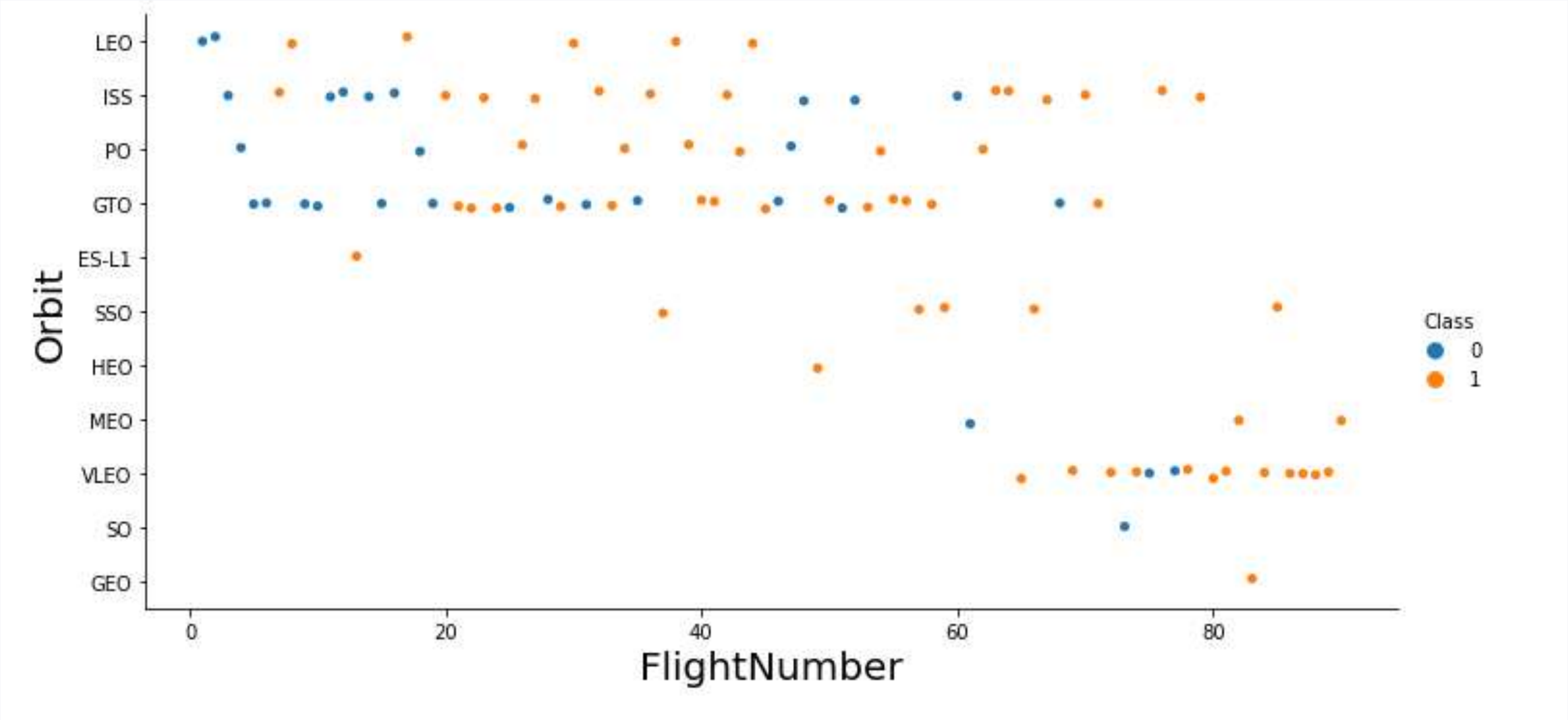
- the graphic shows a relationship between the locations and the payload mass of the postings

Success Rate vs. Orbit Type



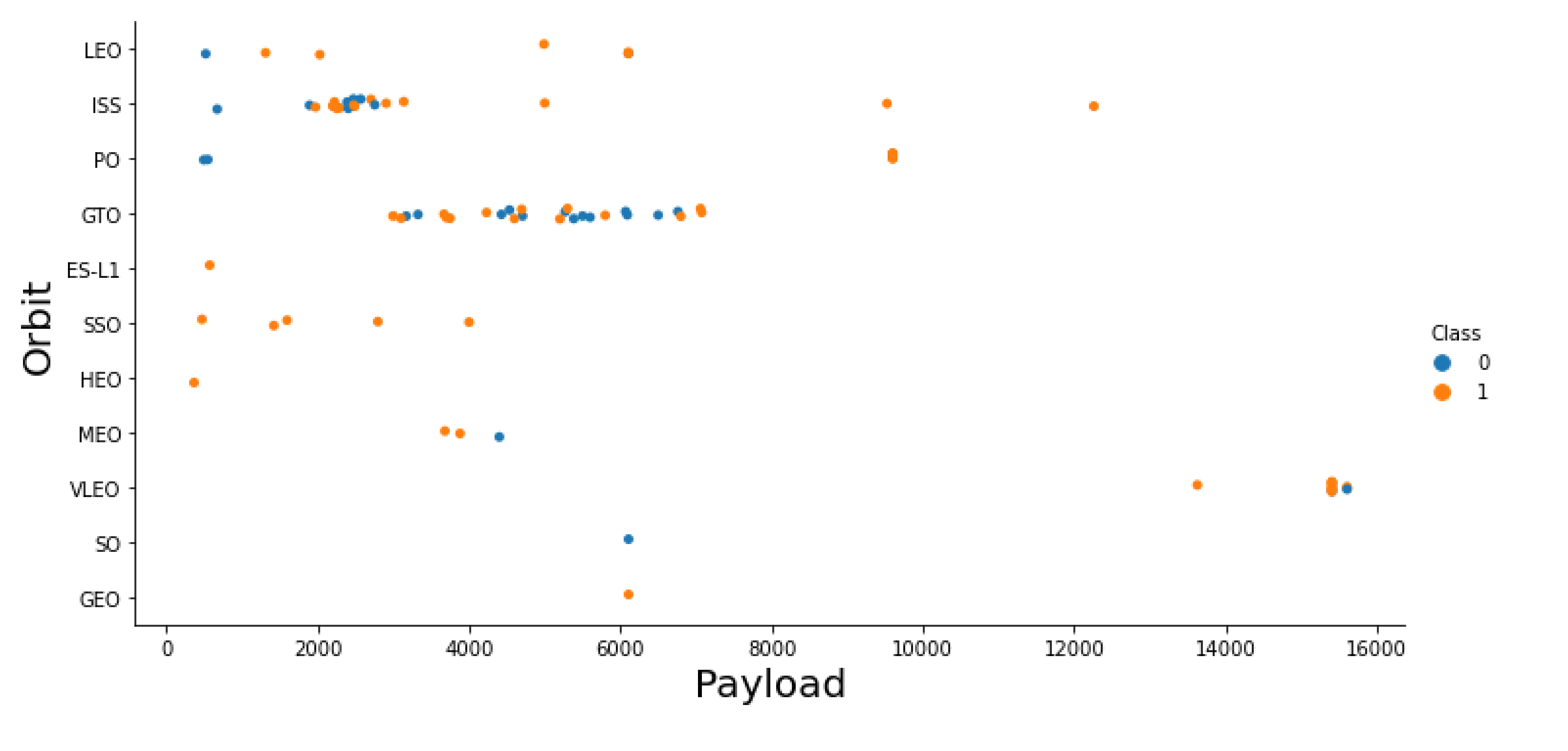
- Through the bar graph it is possible to notice how the SSO, GEO, ES-LI and HEO orbits stand out

Flight Number vs. Orbit Type



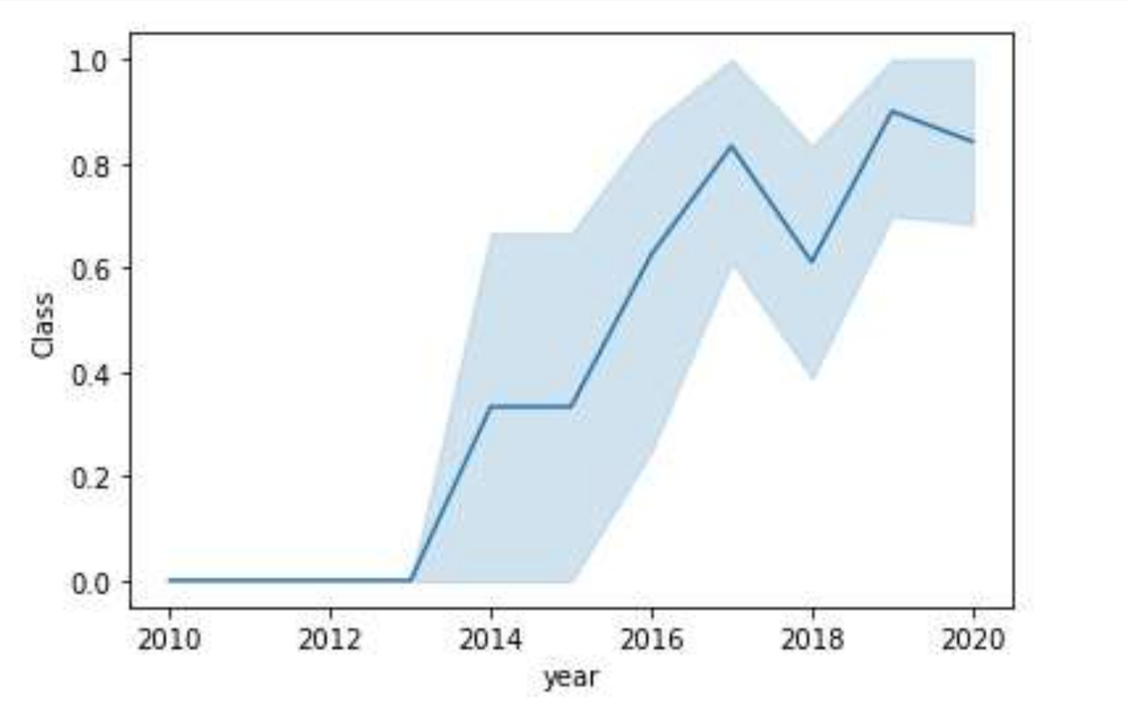
- the graphic shows a relationship between the Flight number and the Orbit type.

Payload vs. Orbit Type



- the graphic shows a relationship between the payload and the Orbit type.

Launch Success Yearly Trend



- the graphic shows a yearly average success rate.

All Launch Site Names

- Find the names of the unique launch sites

```
In [7]: print(sqldf("SELECT DISTINCT Launch_Site FROM df;", locals()))

     Launch_Site
0   CCAFS LC-40
1   VAFB SLC-4E
2    KSC LC-39A
3  CCAFS SLC-40
```

Launch Site Names Begin with 'CCA'

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

```
In [14]: print(sqldf("SELECT * FROM df WHERE Launch_Site LIKE '%CCA%';", locals()).head())

         Date Time (UTC) Booster_Version  Launch_Site  \
0  04-06-2010   18:45:00   F9 v1.0  B0003  CCAFS LC-40
1  08-12-2010   15:43:00   F9 v1.0  B0004  CCAFS LC-40
2  22-05-2012   07:44:00   F9 v1.0  B0005  CCAFS LC-40
3  08-10-2012   00:35:00   F9 v1.0  B0006  CCAFS LC-40
4  01-03-2013   15:10:00   F9 v1.0  B0007  CCAFS LC-40

                                          Payload  PAYLOAD_MASS__KG_  \
0            Dragon Spacecraft Qualification Unit                  0
1  Dragon demo flight C1, two CubeSats, barrel of...                  0
2                           Dragon demo flight C2                525
3                                    SpaceX CRS-1                500
4                                    SpaceX CRS-2                677

       Orbit         Customer Mission_Outcome     Landing _Outcome
0        LEO           SpaceX         Success  Failure (parachute)
1  LEO (ISS)  NASA (COTS) NRO         Success  Failure (parachute)
2  LEO (ISS)      NASA (COTS)         Success           No attempt
3  LEO (ISS)       NASA (CRS)         Success           No attempt
4  LEO (ISS)       NASA (CRS)         Success           No attempt
```

Total Payload Mass

- Calculate the total payload carried by boosters from NASA

```
In [16]: print(sqldf("SELECT Launch_Site, SUM(PAYLOAD_MASS__KG_) FROM df GROUP BY Launch_Site;", locals()).head())

     Launch_Site  SUM(PAYLOAD_MASS__KG_)
0    CCAFS LC-40                   67363
1   CCAFS SLC-40                  254037
2     KSC LC-39A                  208837
3    VAFB SLC-4E                   89730
```

Average Payload Mass by F9 v1.1

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

```
In [28]:
sqldf("SELECT Booster_Version, AVG(PAYLOAD_MASS__KG_) FROM df WHERE Booster_Version LIKE '%F9 v1.1%' GROUP BY Booster_Version;", l
```

	Booster_Version	AVG(PAYLOAD_MASS__KG_)
0	F9 v1.1	2928.4
1	F9 v1.1 B1003	500.0
2	F9 v1.1 B1010	2216.0
3	F9 v1.1 B1011	4428.0
4	F9 v1.1 B1012	2395.0
5	F9 v1.1 B1013	570.0
6	F9 v1.1 B1014	4159.0
7	F9 v1.1 B1015	1898.0
8	F9 v1.1 B1016	4707.0
9	F9 v1.1 B1017	553.0
10	F9 v1.1 B1018	1952.0

First Successful Ground Landing Date

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

```
In [76]: print(sqldf("SELECT Landing_Outcome, min(Date) FROM df WHERE Landing_Outcome == 'Success';", locals()))

  Landing_Outcome                   min(Date)
0         Success  2018-03-12 00:00:00.000000
```

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

In [53]:

```
("SELECT Booster_Version, PAYLOAD_MASS__KG_, Landing_Outcome FROM df WHERE Landing_Outcome == 'Success (drone ship)' AND PAYLOAD_
```

	Booster_Version	PAYLOAD_MASS__KG_	Landing_Outcome
0	F9 FT B1022	4696	Success (drone ship)
1	F9 FT B1026	4600	Success (drone ship)
2	F9 FT B1021.2	5300	Success (drone ship)
3	F9 FT B1031.2	5200	Success (drone ship)

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes

```
In [52]: print(sqldf("SELECT Landing_Outcome, COUNT(Landing_Outcome) FROM df GROUP BY Landing_Outcome", locals()))

             Landing_Outcome  COUNT(Landing_Outcome)
0        Controlled (ocean)                       5
1                   Failure                       3
2      Failure (drone ship)                       5
3       Failure (parachute)                       2
4                No attempt                      21
5               No attempt                        1
6    Precluded (drone ship)                       1
7                   Success                      38
8      Success (drone ship)                      14
9      Success (ground pad)                       9
10     Uncontrolled (ocean)                       2
```

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass

```
In [54]: print(sqldf("SELECT Booster_Version, max(PAYLOAD_MASS__KG_) FROM df;", locals()))

  Booster_Version  max(PAYLOAD_MASS__KG_)
0   F9 B5 B1048.4                   15600
```

2015 Launch Records

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

```
In [83]: f("SELECT Booster_Version, Launch_Site, Landing_Outcome, year FROM df WHERE Landing_Outcome == 'Failure (drone ship)' AND year ==
```

	Booster_Version	Launch_Site	Landing_Outcome	year
0	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)	2015
1	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)	2015

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

- Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

```
In [90]: print(sqldf("SELECT Landing_Outcome, COUNT(Landing_Outcome) FROM df WHERE (Date BETWEEN '2010-06-04'AND '2017-03-20') GROUP BY La
```

```
          Landing_Outcome  COUNT(Landing_Outcome)
0              No attempt                      10
1     Success (ground pad)                      5
2     Success (drone ship)                      5
3     Failure (drone ship)                      5
4       Controlled (ocean)                      3
5     Uncontrolled (ocean)                      2
6   Precluded (drone ship)                      1
7      Failure (parachute)                      1
```

Section 4

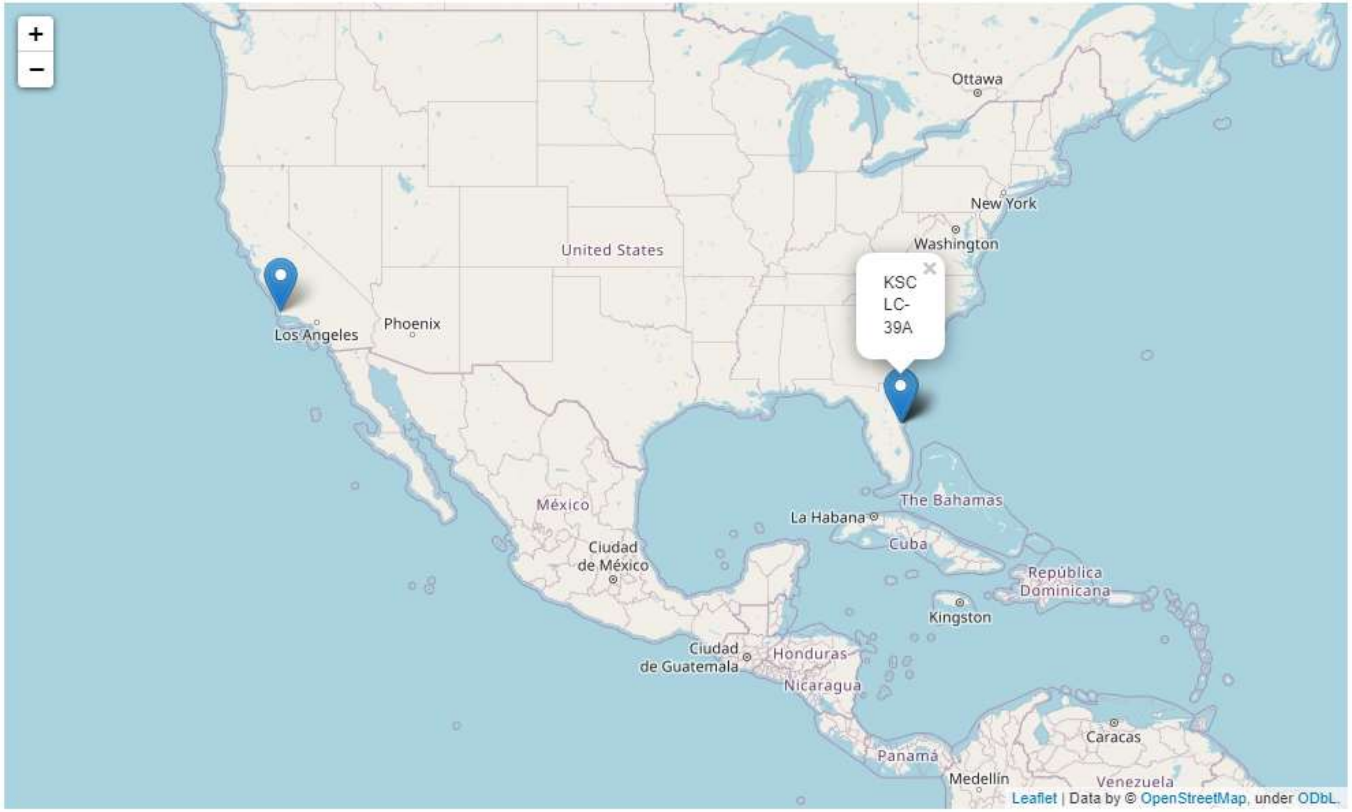
Launch Sites Proximities Analysis

Map with locations

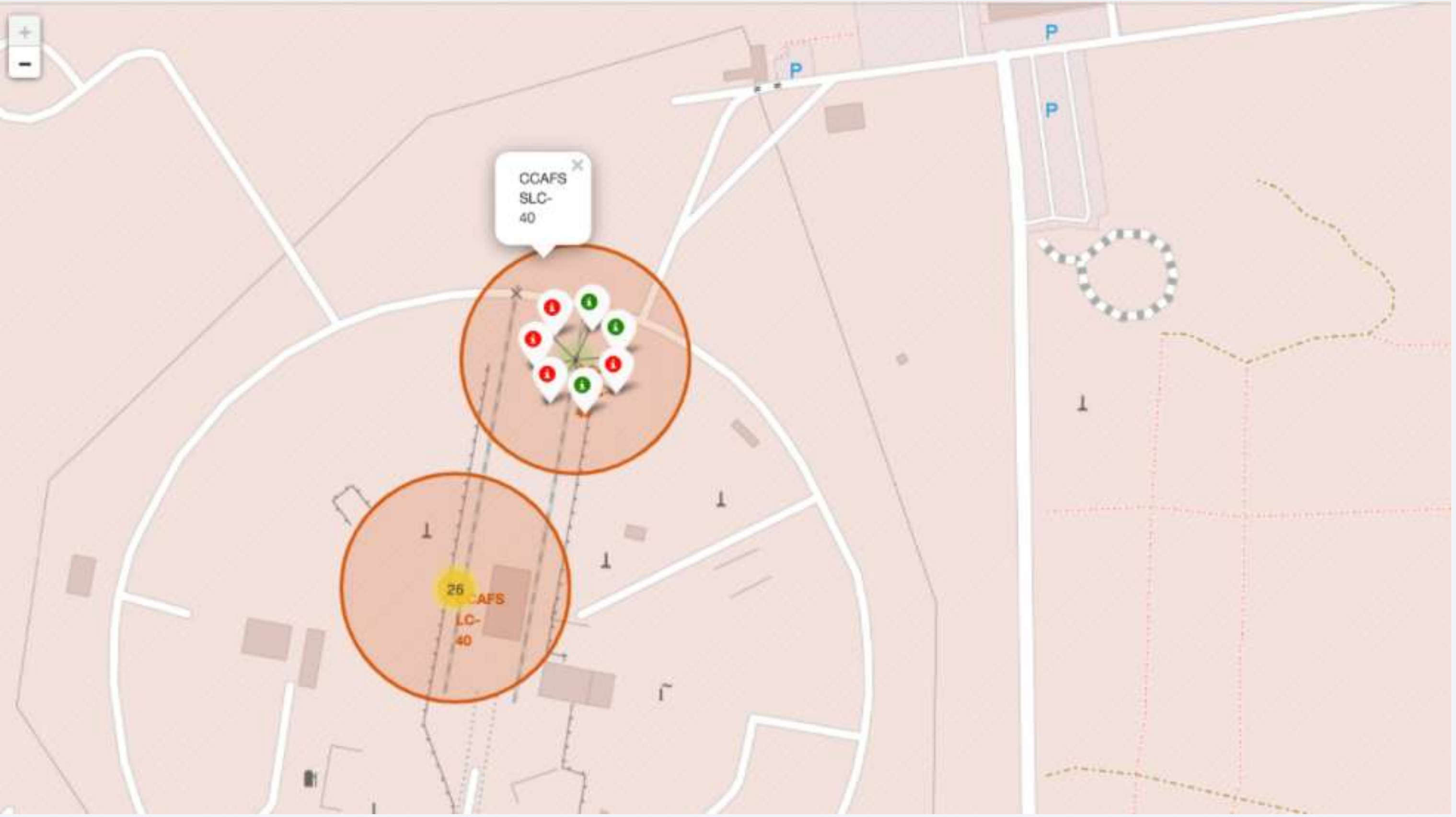
In [36]:

```
site_map2 = folium.Map(location=nasa_coordinate, zoom_start=4)

for i in range(0, len(launch_sites_df)):
    folium.Marker([launch_sites_df.iloc[i]["Lat"], launch_sites_df.iloc[i]["Long"]], popup = launch_sites_df.iloc[i]["Launch Site


site_map2
```

Out[36]:



Map with locations sucess

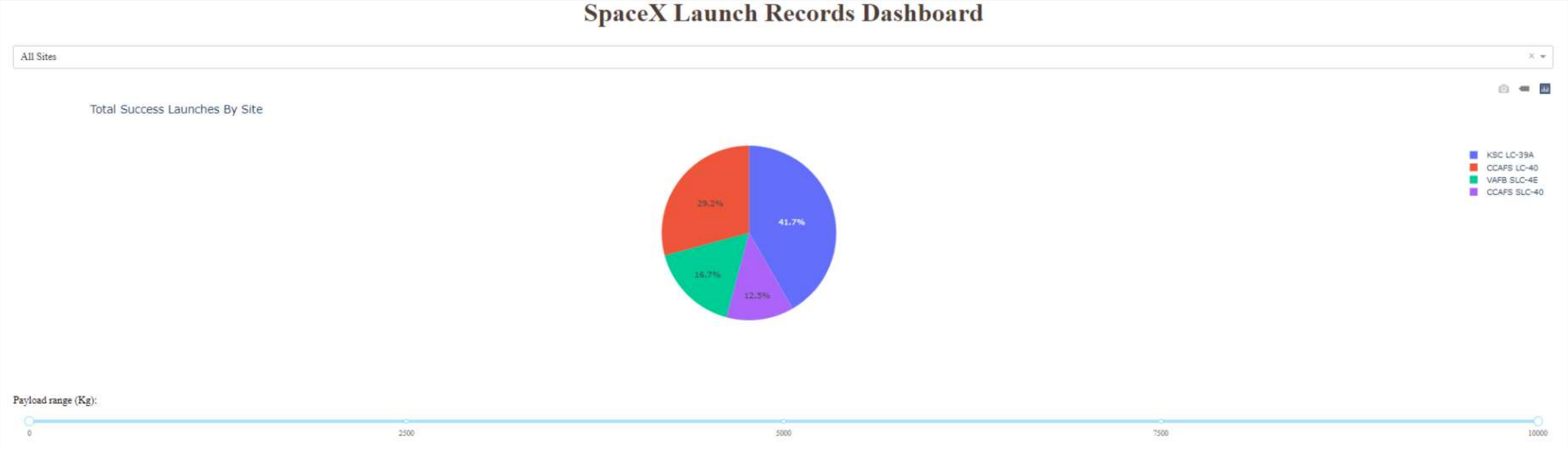


Map launch site to its proximities such as railway, highway, coastline

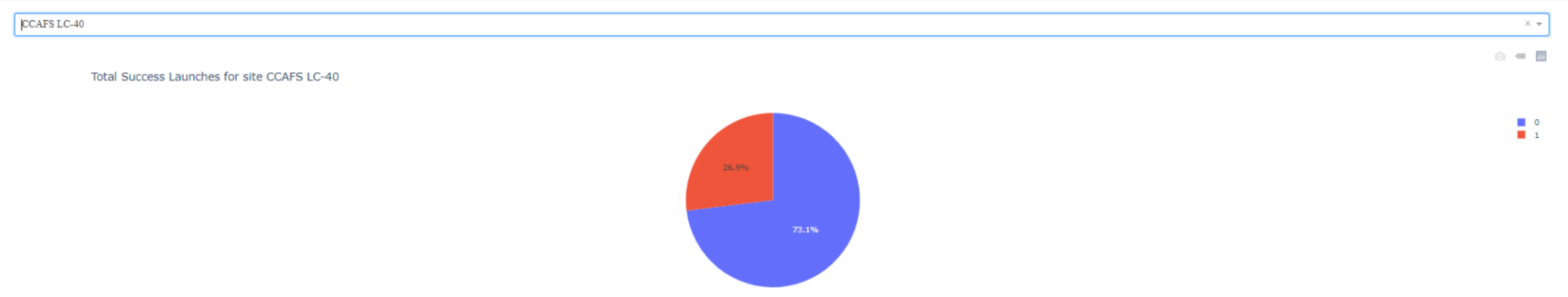
Section 5

Build a Dashboard with Plotly Dash

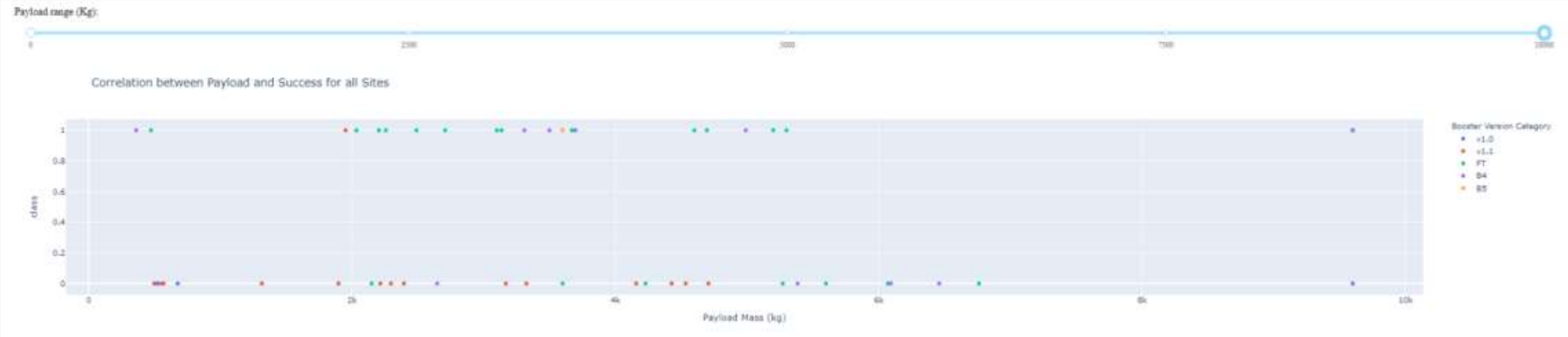
Space X launch records



Total success launches for sites



Correlation between payload and success for sites

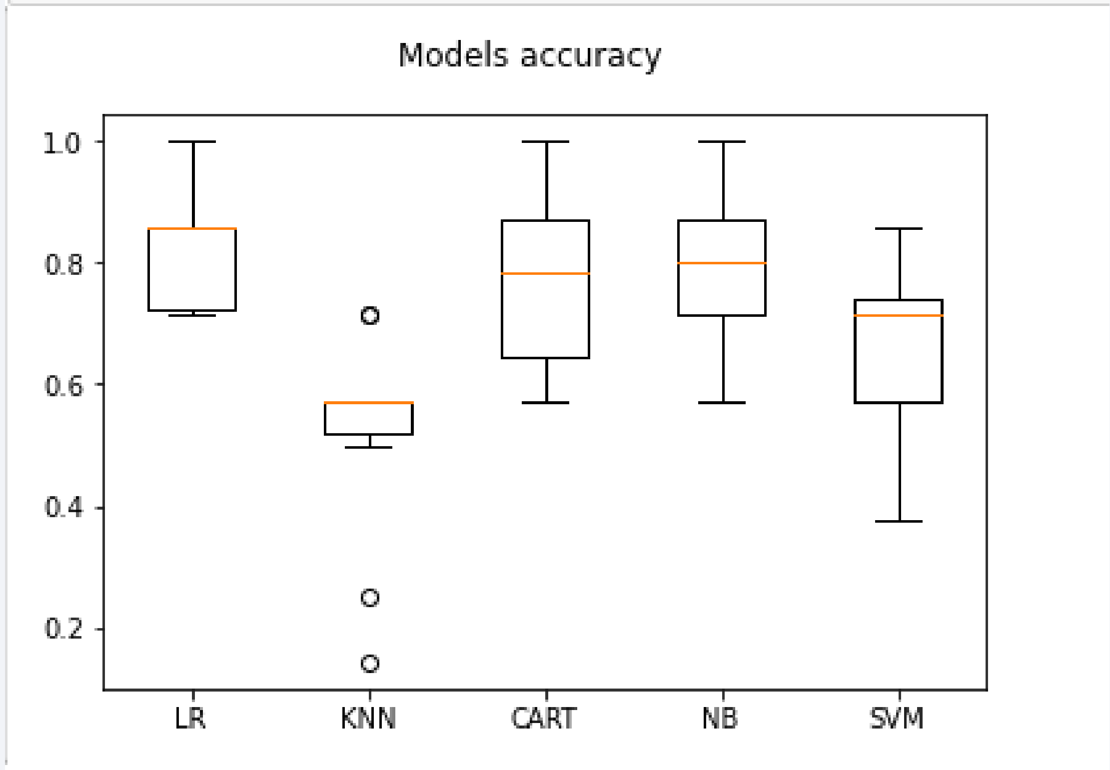


Section 6

Predictive Analysis (Classification)

Classification Accuracy

- Graph:



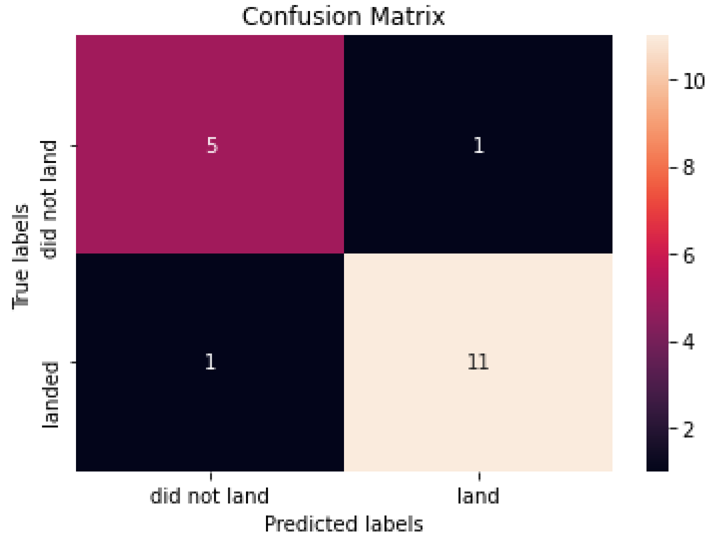
- Best accuracy:

```
In [26]: print("Best score is {}".format(tree_cv.best_score_))

Best score is 0.9444444444444443
```

Confusion Matrix

- The matrix shows how the model had a high assertiveness, since when comparing actual and predicted values, it proved to be robust in terms of success and minimized errors

```
In [27]: yhat = tree_cv.predict(X_test)
         plot_confusion_matrix(Y_test,yhat)
```



Conclusions

- Throughout the presentation, it is possible to observe several characteristics of the dataset, which represent the real behavior of Space X's rockets. This is also a demonstration of the company's own strategy. Furthermore, it is also possible to project the success of future missions through machine learning algorithms, which in this case, was the best one by Random Forest.

Appendix

- https://github.com/CleitonOERocha/IBM_DS_Final_Project

- https://towardsdatascience.com/using-spark-r-to-analyze-emergency-financial-assistance-data-in-brazil-92957e0e25a7

Thank you!