Stefanos Hadjiefstathiou
28 March 2023

Outline

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

Executive Summary

- **Summary of methodologies**
  - Data Collection through API
  - Data Collection with Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis with Data Visualization
  - Interactive Visual Analytics with Folium
  - Machine Learning Prediction
- **Summary of all results**
  - Exploratory Data Analysis result
  - Interactive analytics in screenshots
  - Predictive Analytics result

Introduction

- **Project background and context**
  - **Background**

    Falcon 9 rocket of Space X cost 62 million dollars while at the same time the competitor's price is around 165 million dollars each. The Space X rockets are mainly divided in two parts the first and the second stage. If the second stage lands successfully it can be used again which is the key point of Space X low price.

  - **Context**

    An alternate company wants to bit against Space X for a rocket launch. In order to be able to do this the alternate company has to be able to predict if the first stage will be landed successfully. The aim of this project is to create a machine learning pipeline to predict if the first stage will land.

- **Problems you want to find answers**
  - What factors determine if the rocket will land successfully?
  - The interaction amongst various features that determine the success rate of a successful landing.
  - What operating conditions needs to be in place to ensure a successful landing program.

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using SpaceX API and web scraping from Wikipedia.
- Perform data wrangling
  - One-hot encoding was applied to categorical features
- 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

- **The data was collected using various methods**
  - Data collection was done using SpaceX API and requests library.
  - Response content was decoded as a Json using .json() function call and turn it into a pandas dataframe using .json_normalize().
  - Next, the data was cleaned , checked for missing values and fill in missing values where necessary.
  - Later, web scraping was performed from Wikipedia for Falcon 9 launch records using BeautifulSoup library.
  - The goal was to extract the launch records as HTML table, parse the table and convert it to a pandas dataframe for further analysis.

Data Collection – SpaceX API

- Using the get request to the SpaceX API the data was collected. Afterwards, basic data wrangling and filtering was performed.
- The full procedure is described in the following notebook: https://github.com/stefcy96/final_assignment/blob/35aab46f3870b3480c76c61bf11cbd0ede92db9a/Data%20Collection%20API%20Lab1.ipynb

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/da
response = requests.get(static_json_url)
```

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

```
response.status_code
```

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

```
#import json
# Use json_normalize meethod to convert the json result into a dataframe
resp_dict = response.json()
data = pd.json_normalize(resp_dict)
```

Task 3: Dealing with Missing Values

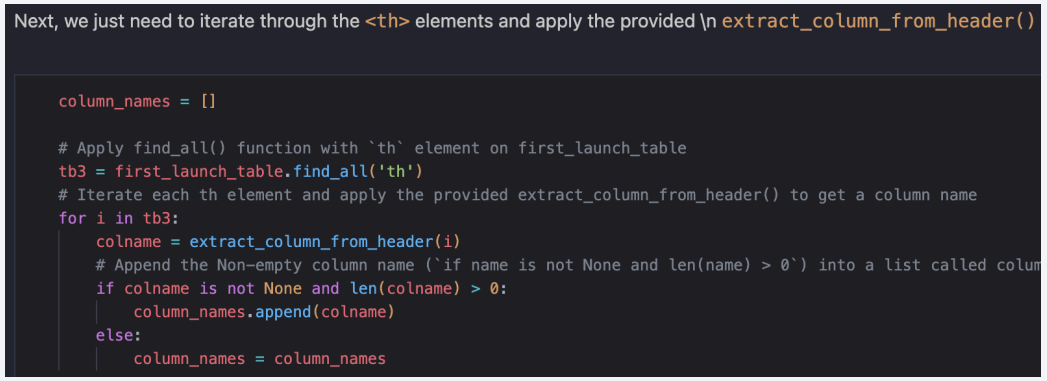
Calculate below the mean for the `PayloadMass` using the `.mean()`. Then use the mean and the `.replace()` function to replace `np.nan` values

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

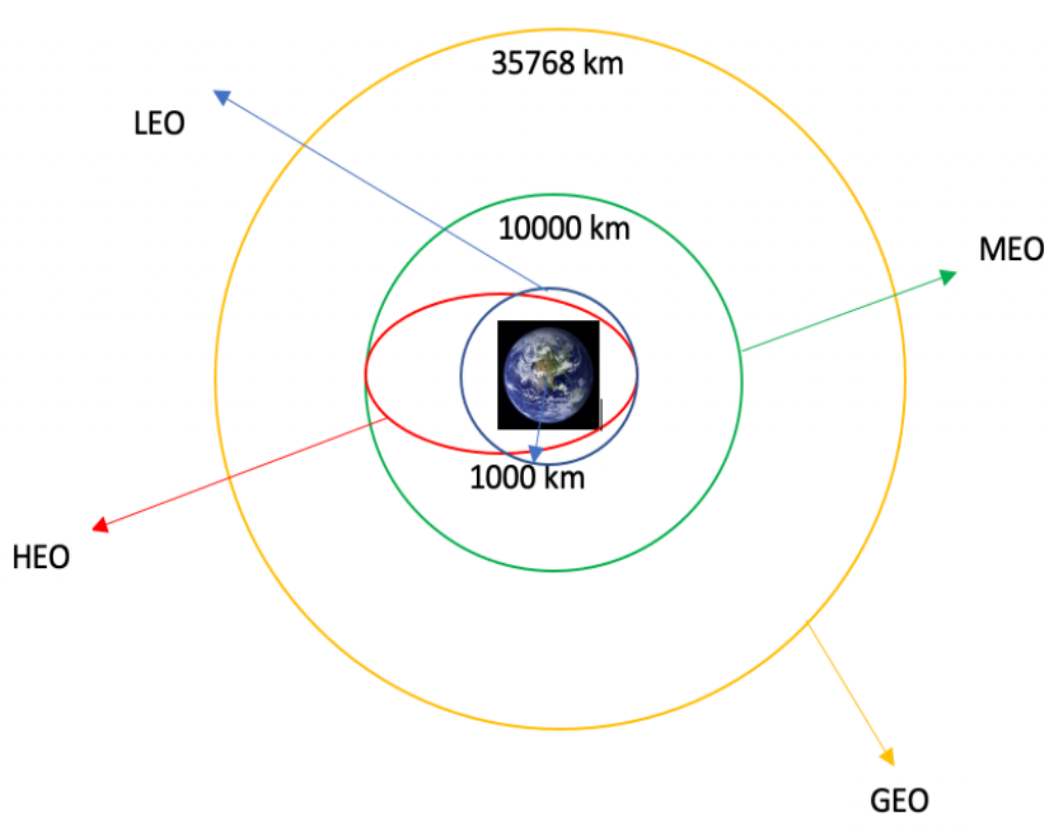
✓ 0.0s

Data Collection - Scraping

- Web scrapping was applied to Falcon 9 launch records with BeautifulSoup
- Then the content of tables was extracted and after filtering and cleaning the data was passed to a dataframe
- The full procedure is described in the following notebook:https://github.com/stefcy96/final_assignment/blob/bad7c2dcedb5d347dfec3993095afdfc2cd5d7c5/Web%20Scraping%20Lab-2.ipynb



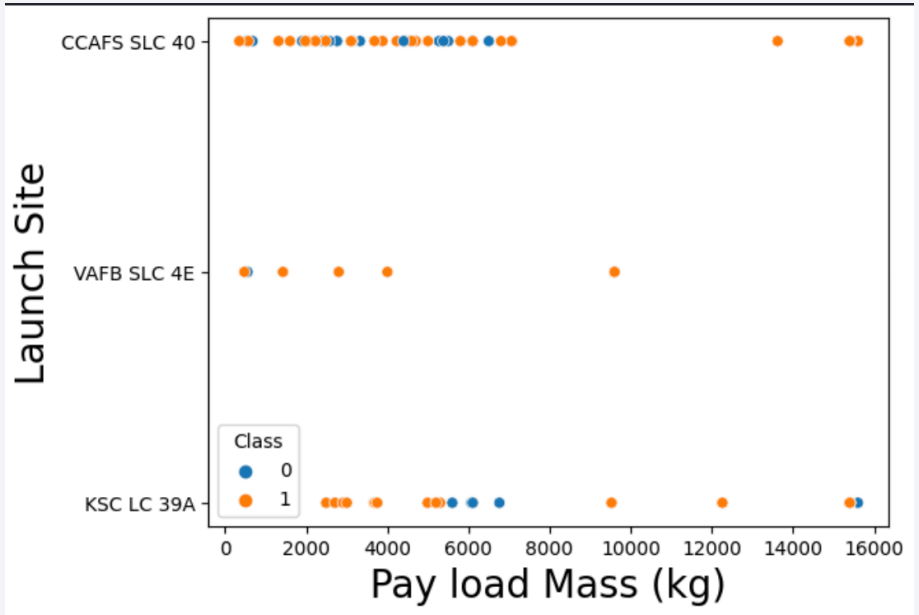
Data Wrangling

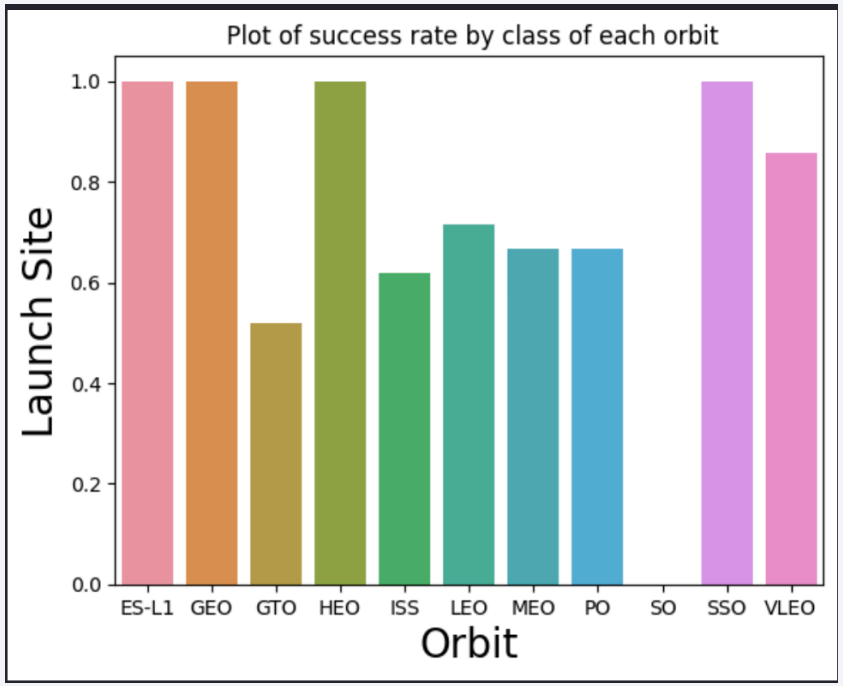


- Exploratory data analysis was performed and the training labels were determined.
- The number of lunches for each site was calculated and also the orbit occurence.
- The landing outcome was created and from outcome column.
- The data was then exported to csv.
- The full procedure is described in the following notebook: https://github.com/stefcy96/final_assignment/blob/bad7c2dcedb5d347dfec3993095afdfc2cd5d7c5/Data%20Wrangling%20Lab3.ipynb

EDA with Data Visualization

- Using data visualization the following relationships were examined: flight number and launch Site, payload and launch site, success rate of each orbit type, flight number and orbit type, the launch success yearly trend.





- The full procedure is described in the following notebook: https://github.com/stefcy96/final_assignment/blob/bad7c2dcedb5d347dfec3993095afdfc2cd5d7c5/jupyter-labs-eda-dataviz.ipynb

EDA with SQL

- In order to use SQL for analysing SpaceX dataset into jupyter notebook sqlite was imported using the following command : %sql sqlite:/// my_data1.db
- EDA was applied with SQL to get insights from the data. A variety of queries were performed to explore further the data.
- The full procedure is described in the following notebook:

Interactive Map with Folium

- All launch sites were marked and added to the map. Objects such as markers, circles, lines were used to present lunch sites in a useful way.
- Feature launch outcomes was created (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.
- Color-labeled marker clusters were used to identify which launch sites have relatively high success rate.
- Different distances were calculated from the lunch sites in order to answer to some questions regarding to the lunch site locations.
- The full procedure is described in the following notebook: https://github.com/stefcy96/final_assignment/blob/bad7c2dcedb5d347dfec3993095afdfc2cd5d7c5/Folium-Lab.ipynb

Dashboard with Plotly Dash

- An interactive dashboard with Plotly dash was created.
- Pie charts are showing the total launches for certain sites
- Scatter graph is showing the relationship for Outcome and Payload Mass (Kg) for different booster versions.
- The full procedure is described in the following notebook: https://github.com/stefcy96/final_assignment/blob/bad7c2dcedb5d347dfec3993095afdfc2cd5d7c5/Dashboard_ploty.ipynb

Predictive Analysis (Classification)

- The data was loaded using numpy and pandas, then transformed and finally split the data into training and testing.
- Different machine learning models were build and tune different hyperparameters using GridSearchCV.
- Accuracy was used as the metric for the model.
- The best performing classification model was found based on the accuracy.
- The full procedure is described in the following notebook: https://github.com/stefcy96/final_assignment/blob/bad7c2dcedb5d347dfec3993095afdfc2cd5d7c5/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

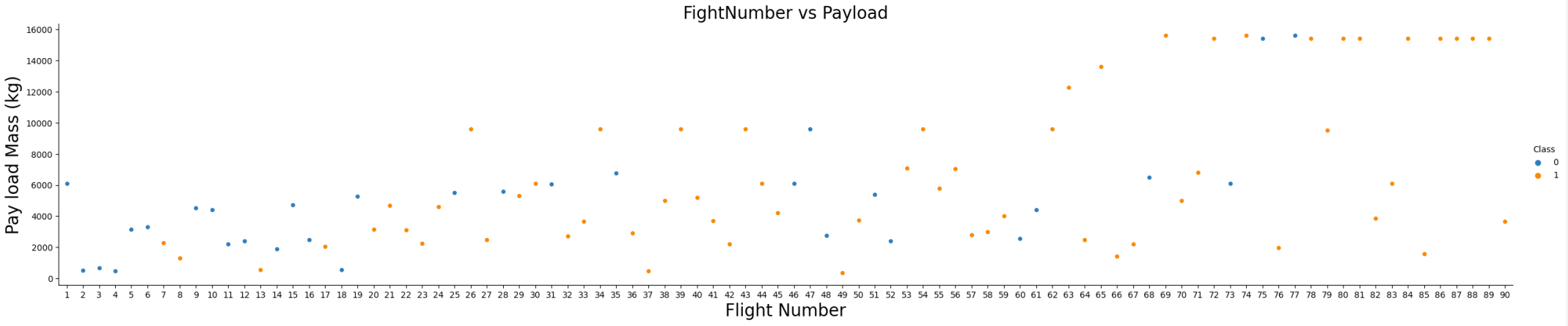
Results

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

EDA

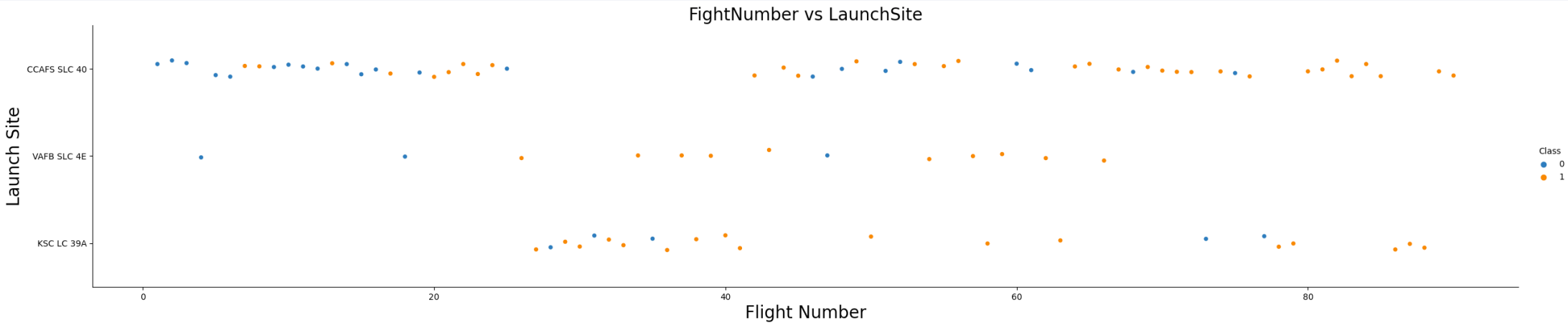
Flight Number vs. Payload

- It can be seen that as the number of flights increases the successful landing percentages are rising. Payload mass does not seem to affect the landing after the first 30 flights.



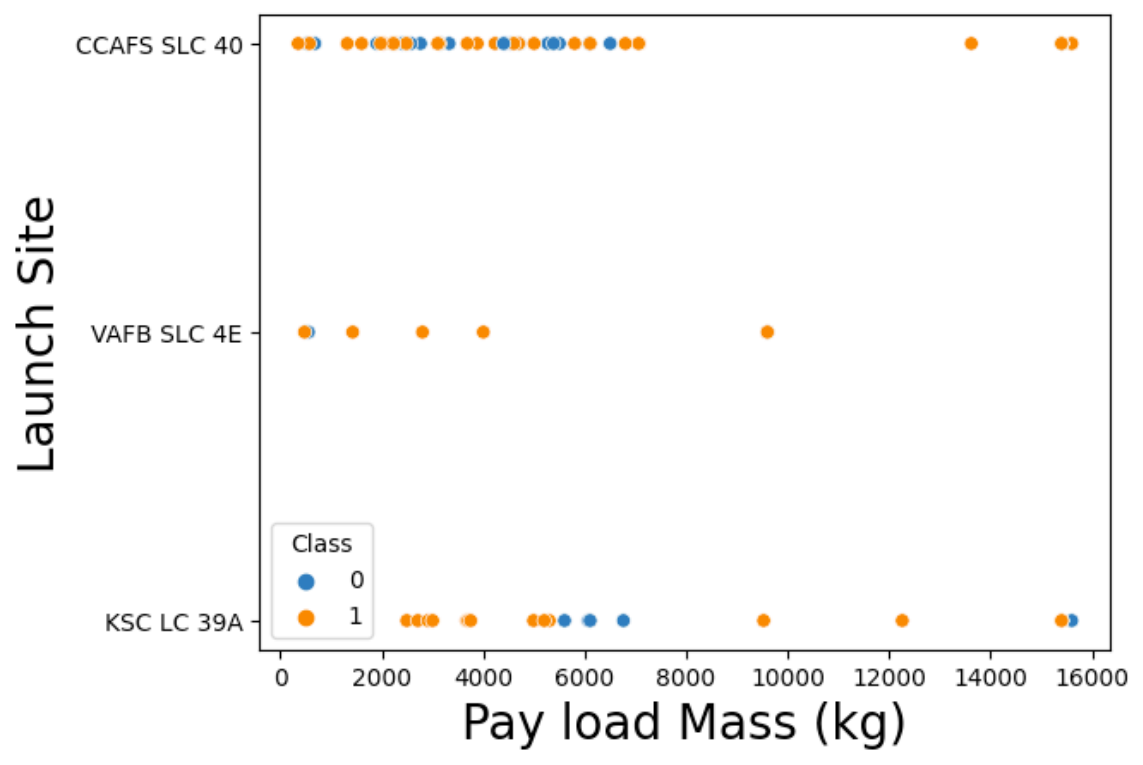
Flight Number vs. LaunchSite

- It can be seen that there is a clear difference in the success percentage in the different LaunchSites. The worst is the CCAFS SLC 40 is not very accurate result as this site has the most early flights.



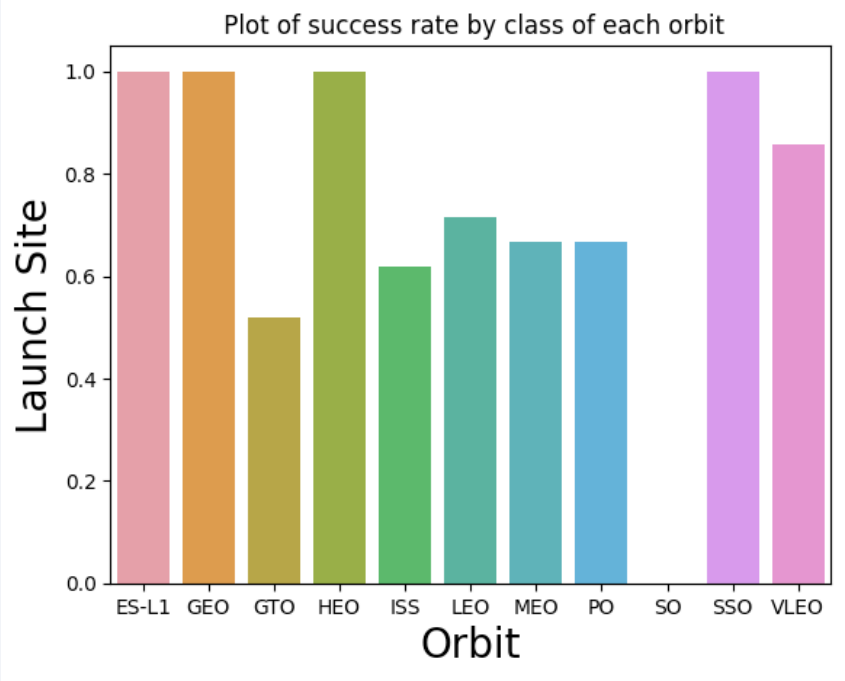
PayLoad vs. LaunchSite

- Most of the flights have PayLoad less than 6000kg. Is is also visible that VAFB SLC 4E has no flight over 1000kg.



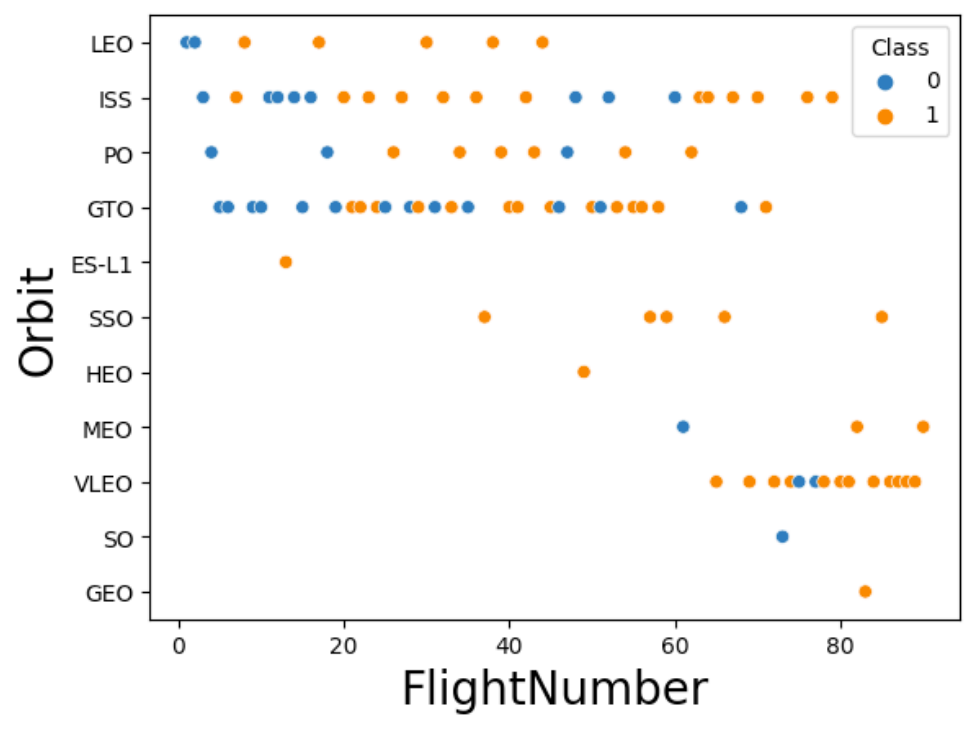
Success Rate vs. Orbit Type

- From the plot, we can see that ES-L1, GEO, HEO, SSO, VLEO had the highest success rate.



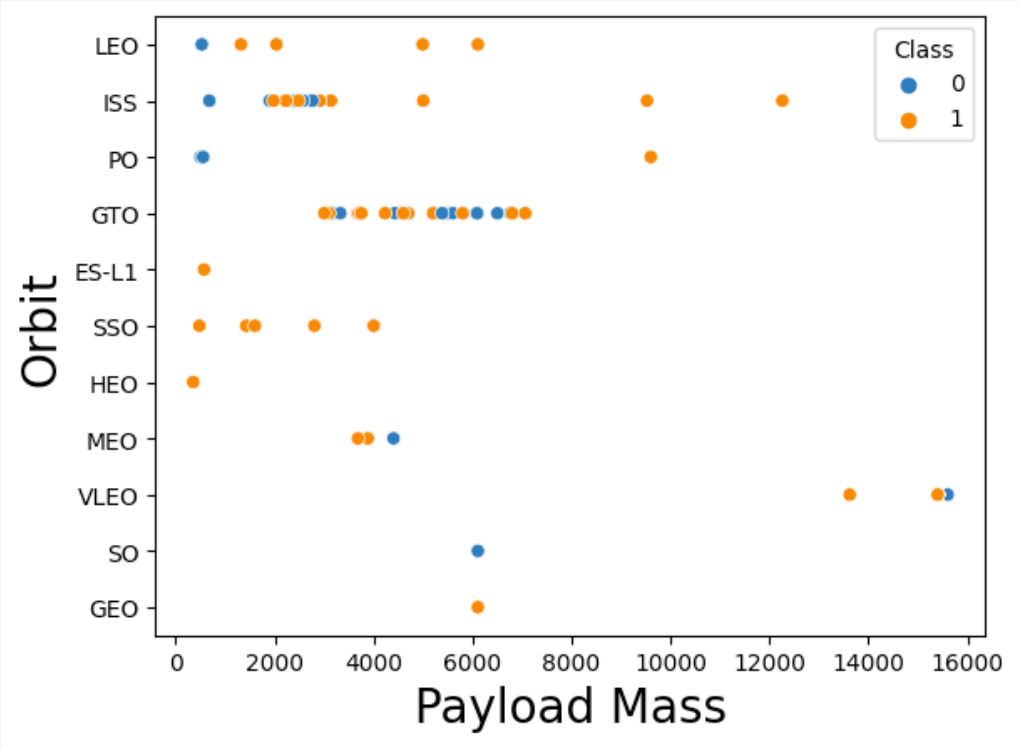
Flight Number vs. Orbit Type

- LEO and PO had successes in the early flights compared to ISS and GTO. Additionally VLEO started launches after flight 60.



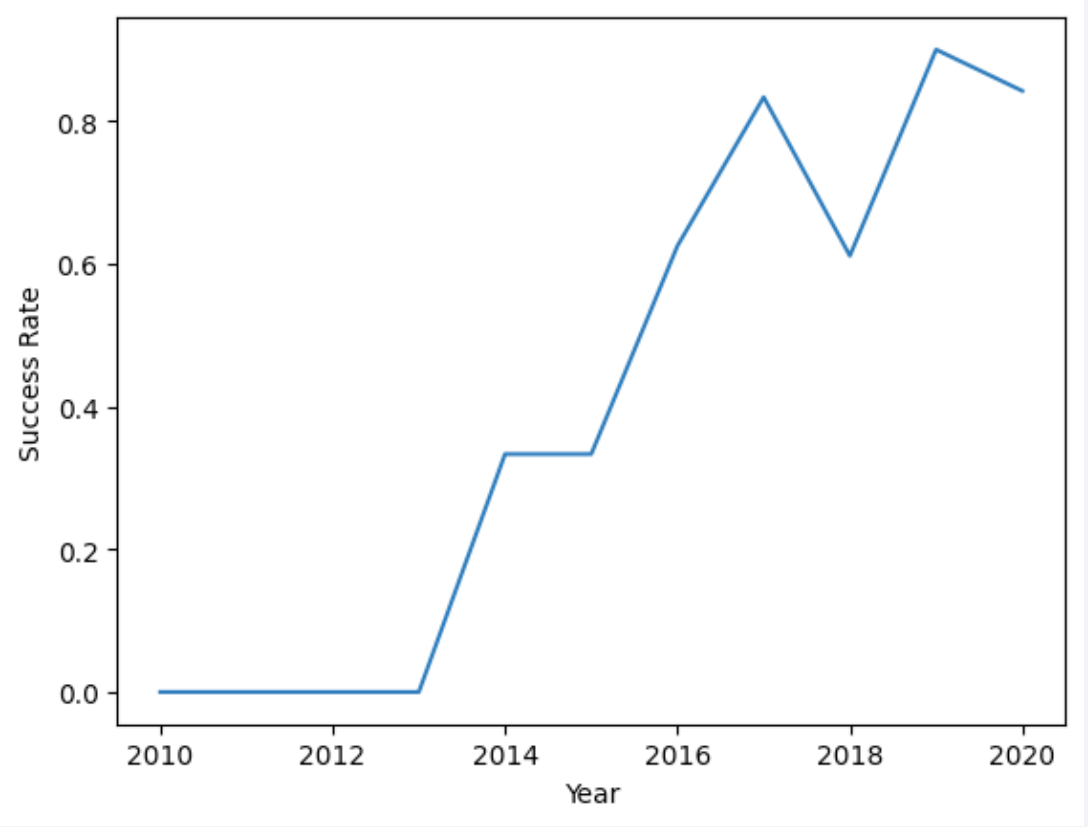
Payload vs. Orbit Type

- No clear relationship between Payload and Orbit Type.



Payload vs. Orbit Type

- The plot shows an overall uptrend from 2013 until 2020. There is a significant decrease in success rate at 2018 and it may worth further investigation.



EDA Using SQLite

- Create a connection with sqlite with a database name “my_data1.db”
- Passing pandas dataframe to a table called “SPACEXTBL” the database.

Unique Lunch Sites (1)

- Using the distinct we can find all the unique Lunch Sites from SPACEXTBL .

```
task1 = con.execute(" select distinct(Launch_Site) from SPACEXTBL")
task1.fetchall()
```

✓ 0.0s

```
[('CCAFS LC-40',), ('VAFB SLC-4E',), ('KSC LC-39A',), ('CCAFS SLC-40',)]
```

Launch Site Names Begin with ‘CCA' (2)

```
task2 = con.execute("select * from SPACEXTBL where Launch_Site like 'CCA%' limit 5 ")
j = task2.fetchall()
j

df2 = pd.DataFrame(j)
df2.head(40)
```

✓ 0.0s

	0	1	2	3	4	5	6	7	8	9
0	04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	08-12-2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2	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
3	08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

- Using like ‘CCA%’ we can filter the Launch Site column and limit the answer to 5.

Total Payload mass of NASA Customers (3)

- Using the the sum of payload where and NASA (CRS) is the customer .

```
task3 = con.execute("select SUM(PAYLOAD_MASS__KG_) from SPACEXTBL where Customer = 'NASA (CRS)' ")
x=task3.fetchone()
print(x)

#pandas cross checking check
t3 = df[df['Customer']== 'NASA (CRS)']
t32 =t3.PAYLOAD_MASS__KG_.sum()
print(t32)
```

✓ 0.0s

```
(45596,)
45596
```

Average Payload mass carried by Booster Version F9 v1.1(4)

- Using where we filter the Booster_version to be F9 v1.1

```
task4 = con.execute("""select avg(PAYLOAD_MASS__KG_)
                       from SPACEXTBL where Booster_Version like 'F9 v1.1'
                    """)
x = task4.fetchone()
print(x)

#pandas cross checking
t4 = df[df['Booster_Version'] == 'F9 v1.1']
t42 = t4.PAYLOAD_MASS__KG_.mean()
print(t42)
```

[25] ✓ 0.0s

```
(2928.4,)
2928.4
```

First Successful Landing Outcome in ground pad (5)

- First we filter the column Landing Outcome to be ‘Success (ground pad)’ and the we order the records by date and limit them to 1.

```
task5 = con.execute("""
                    select Date from SPACEXTBL
                    where "Landing _Outcome" = 'Success (ground pad)'
                    order by Date desc
                    limit 1
                    """)
x = task5.fetchall()
print(x)

# pandas solution
t5 = df[df['Landing _Outcome'] == 'Success (ground pad)'].sort_values(by=['Date'], ascending=False)
print(t5.iloc[0,0])
```

✓ 0.0s

```
[('22-12-2015',)]
22-12-2015
```

Name of Booster for successful lands on drone ships and have payload mass between 4000 and 6000 kg. (6)

- First we filter the column Landing Outcome to be ‘Success (drone ship)’ and the we also filter the PAYLOAD_MASS to be between 4000 and 6000.

```
task6 = con.execute("""
                    select Booster_Version from SPACEXTBL
                    where "Landing _Outcome" = 'Success (drone ship)'
                    and (PAYLOAD_MASS__KG_ > 4000 and PAYLOAD_MASS__KG_<6000)
                     """)
print(task6.fetchall(),"\n")


# pandas solution
t6 = df[df['Landing _Outcome'] == 'Success (drone ship)']
t62 = t6[ (t6['PAYLOAD_MASS__KG_'] >4000) & (t6['PAYLOAD_MASS__KG_']<6000)]
t63=t62['Booster_Version']
print(t63)
```

✓ 0.0s

```
[('F9 FT B1022',), ('F9 FT B1026',), ('F9 FT  B1021.2',), ('F9 FT  B1031.2',)]

23       F9 FT B1022
27       F9 FT B1026
31    F9 FT  B1021.2
42    F9 FT  B1031.2
```

All possibles outcomes and their quantities (7)

- First we select Mission_Outcome and we count the records and then we group and order the records based on Mission_Outcome.

```
t7 = con.execute("""
        SELECT Mission_Outcome, count(*) FROM SPACEXTBL
        group by Mission_Outcome
        order by Mission_Outcome;


        """)
t7.fetchall()
```

✓ 0.0s

```
[('Failure (in flight)', 1),
 ('Success', 98),
 ('Success ', 1),
 ('Success (payload status unclear)', 1)]
```

98+1+1 = 100 success

1 failure

Name of Boosters which carried the maximum load. (8)

- First we select Booster_Version and then we filtered using a subquery to find Booster Version which carried the maximum load.

```
t8 = con.execute("""
                 select Booster_Version from SPACEXTBL
                   where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTBL)
                   order by Booster_Version
                 """)
t8.fetchall()
```

✓ 0.0s

```
[('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',)]
```

Name of Boosters which carried the maximum load. (8)

- First we select Booster_Version and then we filtered using a subquery to find Booster Version which carried the maximum load.

```
t8 = con.execute("""
                 select Booster_Version from SPACEXTBL
                   where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTBL)
                   order by Booster_Version
                 """)
t8.fetchall()
```

✓ 0.0s

```
[('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',)]
```

Name of Boosters which carried the maximum load. (8)

- First we select Booster_Version and then we filtered using a subquery to find Booster Version which carried the maximum load.

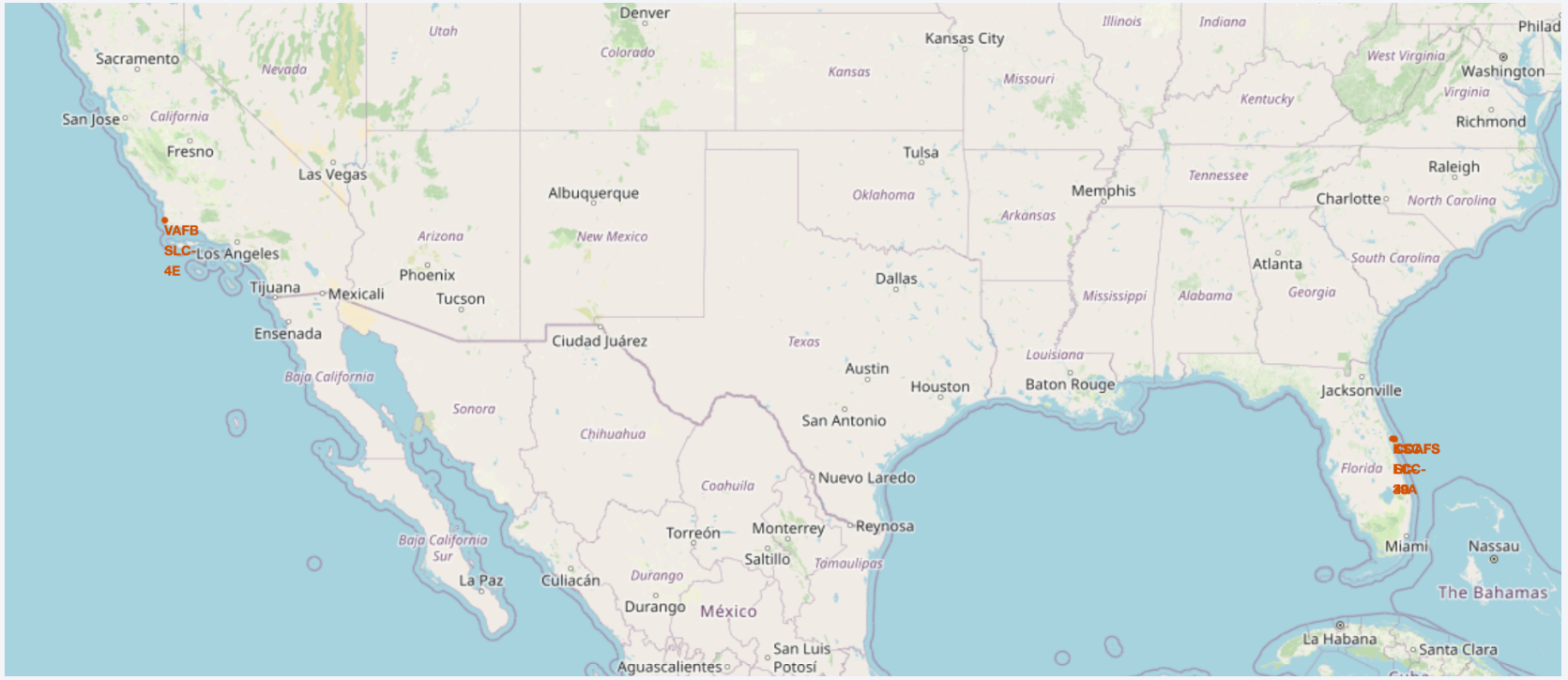
```
t8 = con.execute("""
                 select Booster_Version from SPACEXTBL
                   where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTBL)
                   order by Booster_Version
                 """)
t8.fetchall()
```

✓ 0.0s

```
[('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',)]
```

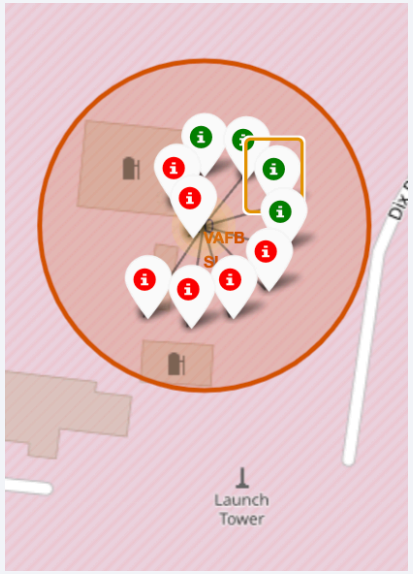
Launch Site Location Analysis with folium

All launch sites global map markers

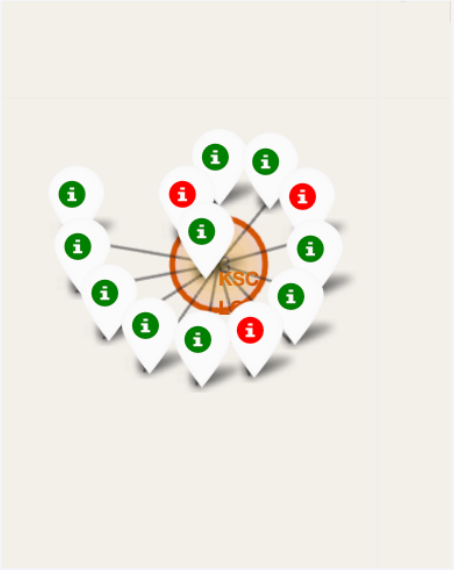


Details for each Launch Site

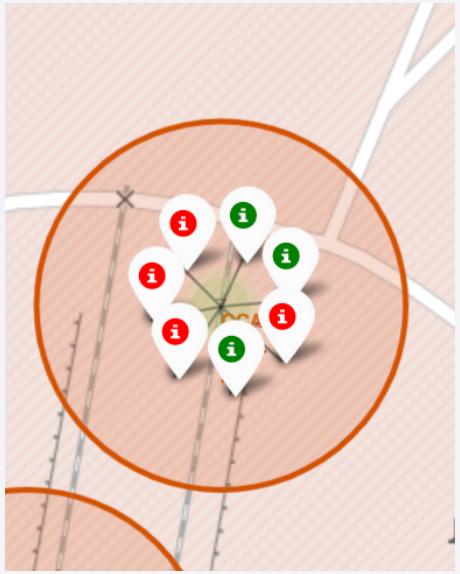
VAFB SLC-4E



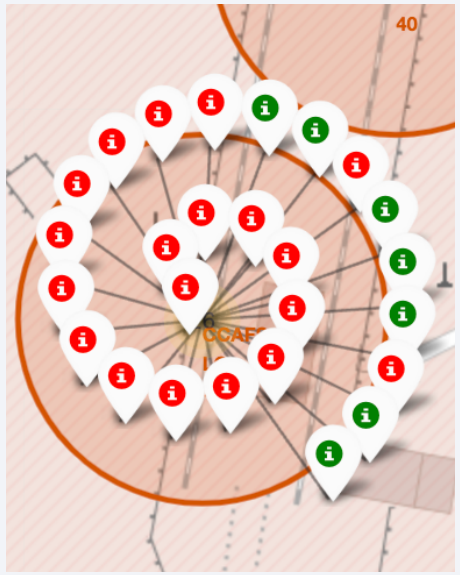
KSC LC-39A



CCAFS SLC-40



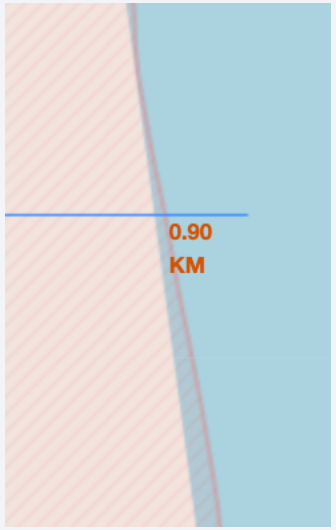
CCAFS LC-40



Green markers indicate that the landing was successful while re markers indicate failure.

Details for each Launch Site

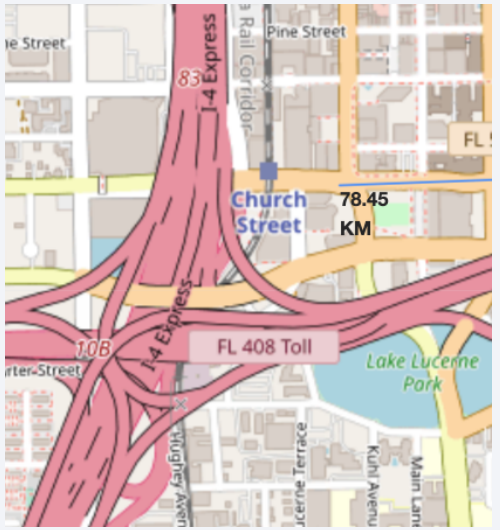
Costline



Highway

Railway

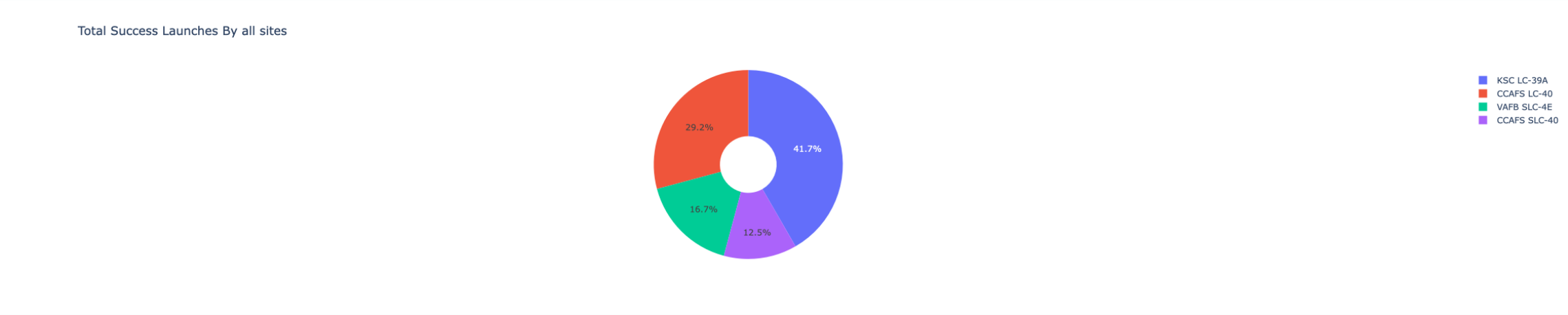
City



The screenshots are showing the closest distance in km from CCAFS SLC-40 to each category. Generally launch sites are close to coastline and away from highways, railways and cities.

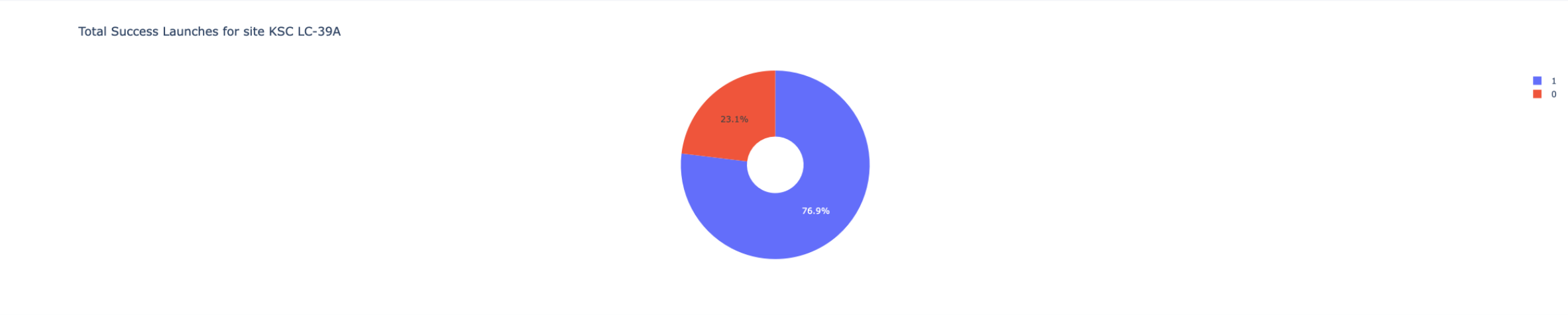
Dashboard using Plotly Dash

Pie chart showing the success percentage achieved by each launch site



KSC LC-39A launch site has the highest success rate of 41,7% of all the successful sites

Pie chart showing the success percentage achieved by each launch site



KSC LC-39A launch site has the highest success to failure ratio of 76.9%

Scatter plot of Payload vs Launch Outcome for all sites, with different payload selected in the range slider

- From this graph it can be seen that the lower the payload the higher the success but without further investigation is needed before concluding to any results.

Predictive Analysis (Classification)

Finding the best classification model to use based on accuracy

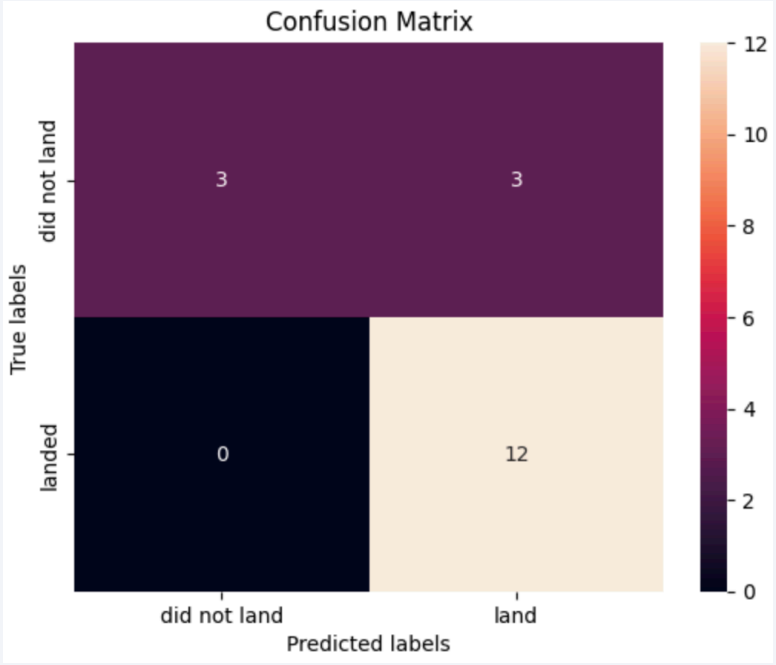
```
algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_}
bestalgorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
if bestalgorithm == 'Tree':
    print('Best Params 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_)
```

✓ 0.0s

```
Best Algorithm is Tree with a score of 0.875
Best Params is : {'criterion': 'gini', 'max_depth': 8, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'splitter': 'best'}
```

After running the piece of code in the screenshot
the best classification algorithm was found to be the Tree
with a score of 0.875.

Confusion Matrix of Decision Tree Algorithm



The confusion matrix is a way of visualising the accuracy of the prediction model.
For example here we have the predicted and the actual values when these values are the same it means that we have correct prediction. The other case is a false prediction like here we have 3 cases that did not land and the model predicted that differently.

Conclusions

- The larger the flight amount at a launch site, the greater the success rate at a launch site.
- Launch success is increasing from 2013 till 2020.
- Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
- KSC LC-39A had the most successful launches of any sites.
- The Decision tree classifier is the best machine learning algorithm for this task.
- Location of lunch sites is close to coastlines and way from highways, railways and cities.

Thank You!