



IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Ahmad Maneshi>
<19th February 2024>



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 python
 - 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**

Space X claims that 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 Space X can reuse the first stage. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against space X for a rocket launch. This goal of the project is to create a machine learning pipeline to predict if the first stage will land successfully.

- **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 need to be in place to ensure a successful landing program



Section 1

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 Python
- Perform interactive visual analytics using Folium
- Perform predictive analysis using classification models

Data Collection

- **Describe how data sets were collected.**
 - Data collection was done using get request to the SpaceX API.
 - Next, we decoded the response content as a Json using `.json()` function call and turn it into a pandas dataframe using `.json_normalize()`.
 - Then we cleaned the data, checked for missing values and fill in missing values where necessary.
 - In addition, we performed web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.
 - The objective was to extract the launch records as HTML table, parse the table and convert it to a pandas dataframe for future analysis.

Data Collection – SpaceX API

- We used the get request to the SpaceX API to collect data, clean the requested data and did some basic data wrangling and formatting.
- The link to the notebook is : [https://github.com/imaneshi/IBM_Capstone-project_SpaceX-Falcon9/blob/main/jupyter_labs_spacex_data_collection_api%20\(1\).ipynb](https://github.com/imaneshi/IBM_Capstone-project_SpaceX-Falcon9/blob/main/jupyter_labs_spacex_data_collection_api%20(1).ipynb)

```
In [5]: # Takes the dataset and uses the cores column to call the API and append the data to the lists
def getCoreData(data):
    for core in data['cores']:
        if core['core'] != None:
            response = requests.get("https://api.spacexdata.com/v4/cores/"+core['core']).json()
            Block.append(response['block'])
            ReusedCount.append(response['reuse_count'])
            Serial.append(response['serial'])
        else:
            Block.append(None)
            ReusedCount.append(None)
            Serial.append(None)
    Outcome.append(str(core['landing_success'])+' '+str(core['landing_type']))
    Flights.append(core['flight'])
    GridFins.append(core['gridfins'])
    Reused.append(core['reused'])
    Legs.append(core['legs'])
    LandingPad.append(core['landpad'])
```

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)
```


Data Collection - Scraping

- We applied web scrapping to webscrap Falcon 9 launch records with BeautifulSoup
- We parsed the table and converted it into a pandas dataframe.
- The link to the notebook is :https://github.com/imaneshi/IBM_Capstone-project_SpaceX-Falcon9/blob/main/jupyter_labs_webscrapping.ipynb

```
In [6]: # use requests.get() method with the provided static_url
# assign the response to a object
html_data = requests.get(static_url)
html_data.status_code
```

```
Out[6]: 200
```

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

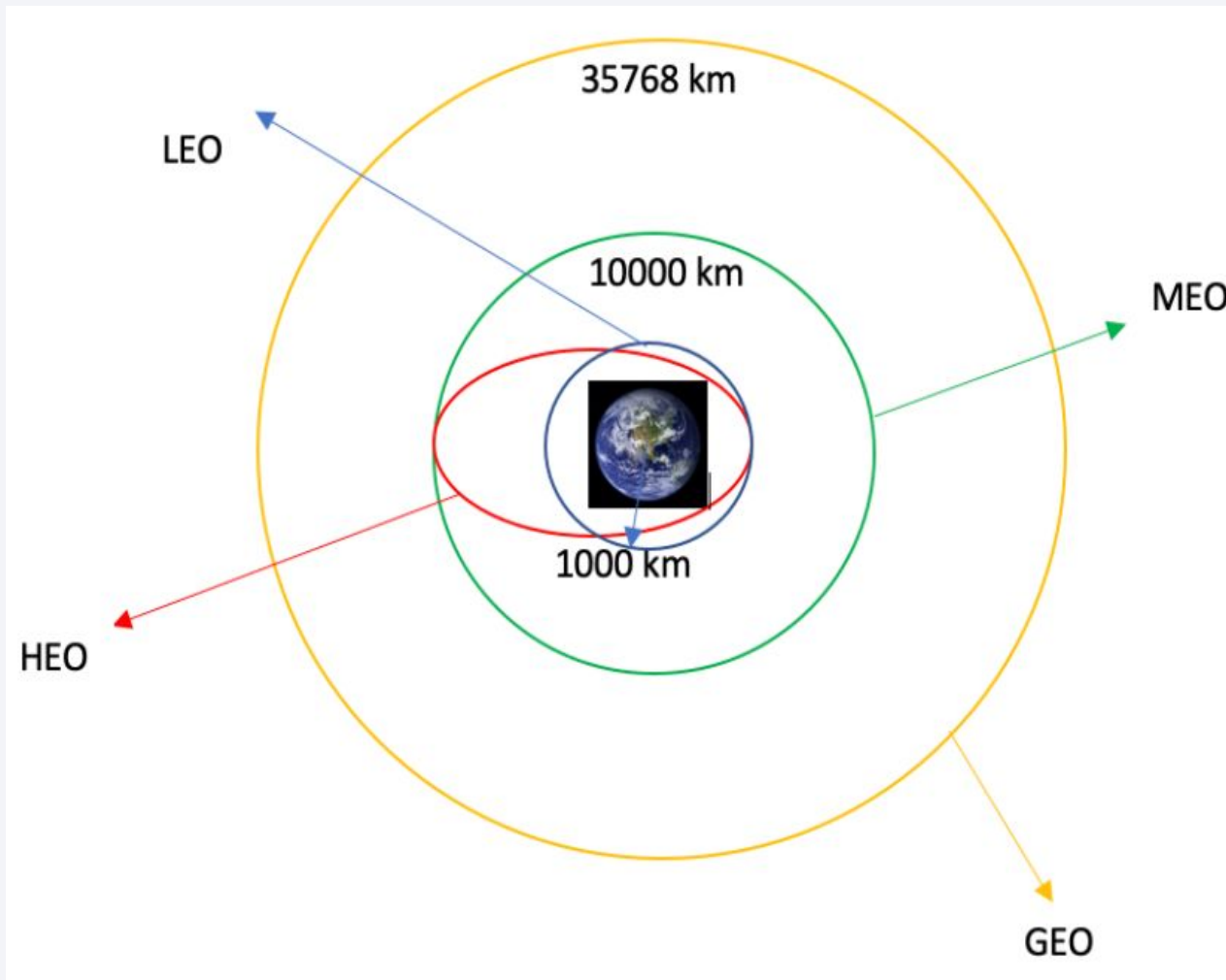
```
In [7]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(html_data.text, 'html.parser')
```

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

```
In [8]: # Use soup.title attribute
soup.title
```

```
Out[8]: <title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

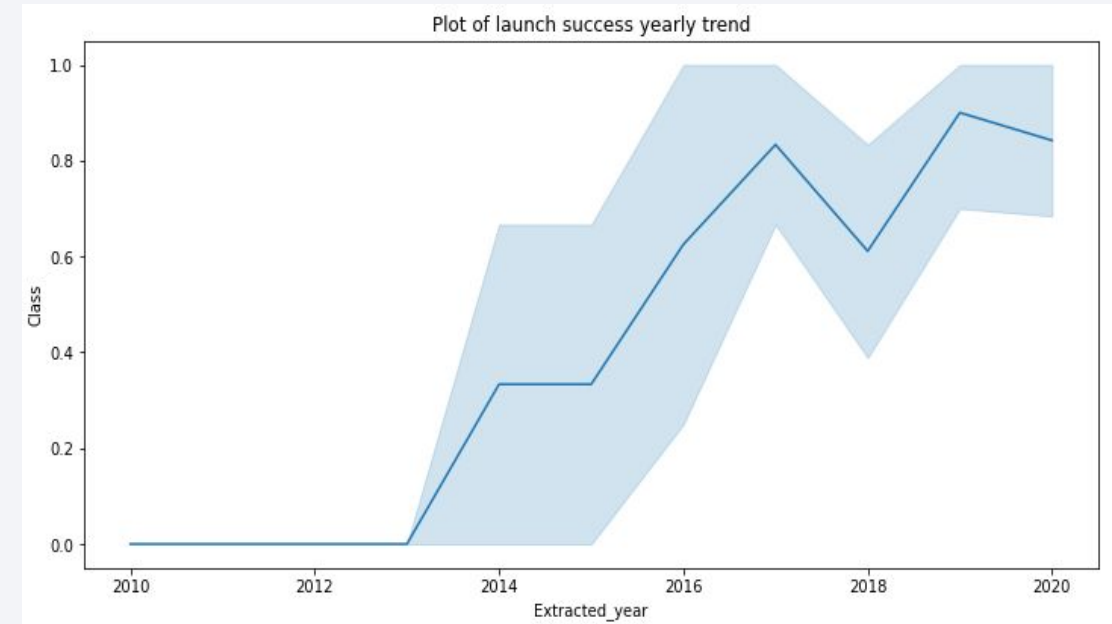
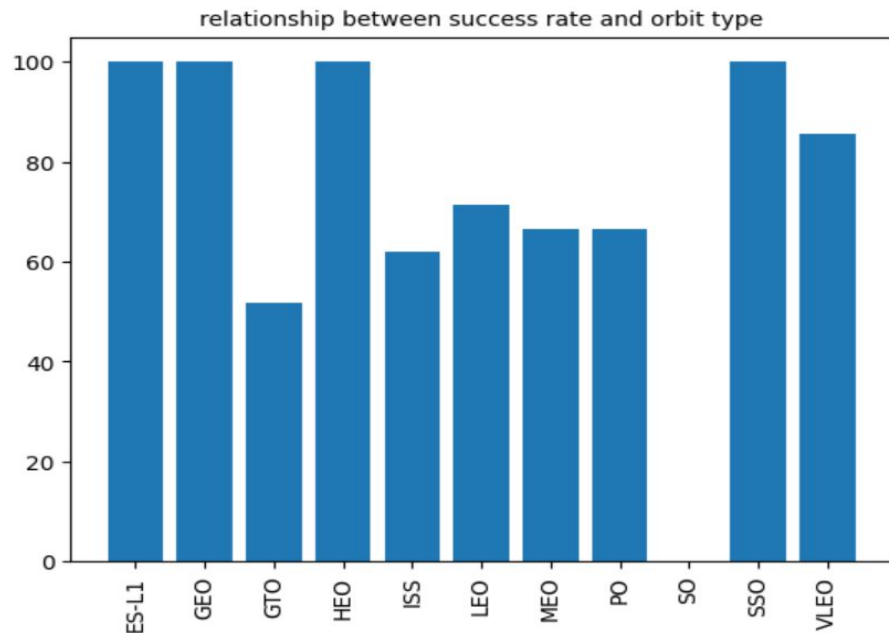
Data Wrangling



- We performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits
- We created landing outcome label from outcome column and exported the results to csv.
- The link to the notebook is [:https://github.com/imaneshi/IBM_Capstone-project_SpaceX-Falcon9/blob/main/labs_jupyter_spacex_Data_wrangling%20\(1\).ipynb](https://github.com/imaneshi/IBM_Capstone-project_SpaceX-Falcon9/blob/main/labs_jupyter_spacex_Data_wrangling%20(1).ipynb)

EDA with Data Visualization

- We explored the data by visualizing the relationship between 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 link to the notebook is:
[https://github.com/imaneshi/IBM_Capstone-project_SpaceX-Falcon9/blob/main/jupyter_labs_eda_dataviz_\(1\).ipynb](https://github.com/imaneshi/IBM_Capstone-project_SpaceX-Falcon9/blob/main/jupyter_labs_eda_dataviz_(1).ipynb)

EDA with SQL

- We loaded the SpaceX dataset into a Python database without leaving the jupyter notebook.
- We applied EDA with python instead of SQL to get insight from the data. We wrote Commands to find out for instance:
 - The names of unique launch sites in the space mission.
 - The total payload mass carried by boosters launched by NASA (CRS)
 - The average payload mass carried by booster version F9 v1.1
 - The total number of successful and failure mission outcomes
 - The failed landing outcomes in drone ship, their booster version and launch site names
- The link to the notebook is
: [https://github.com/imaneshi/IBM_Capstone-project_SpaceX-Falcon9/blob/main/jupyter_labs_eda_sql_coursera_sqlite_\(1\).ipynb](https://github.com/imaneshi/IBM_Capstone-project_SpaceX-Falcon9/blob/main/jupyter_labs_eda_sql_coursera_sqlite_(1).ipynb)

Build an Interactive Map with Folium

- We marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- We assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.
- Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate.
- We calculated the distances between a launch site to its proximities. We answered some question for instance:
 - Are launch sites near railways, highways and coastlines.
 - Do launch sites keep certain distance away from cities.
- The link to the notebook is:
[https://github.com/imaneshi/IBM_Capstone-project_SpaceX-Falcon9/blob/main/lab_jupyter_launch_site_location_\(1\).ipynb](https://github.com/imaneshi/IBM_Capstone-project_SpaceX-Falcon9/blob/main/lab_jupyter_launch_site_location_(1).ipynb)

Predictive Analysis (Classification)

- We loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- We built different machine learning models and tune different hyperparameters using GridSearchCV.
- We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- We found the best performing classification model.
- The link to the notebook is
: https://github.com/imaneshi/IBM_Capstone-project_SpaceX-Falcon9/blob/main/SpaceX_Machine_Learning_Prediction_Part_5_jupyterlite.ipynb

Results

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

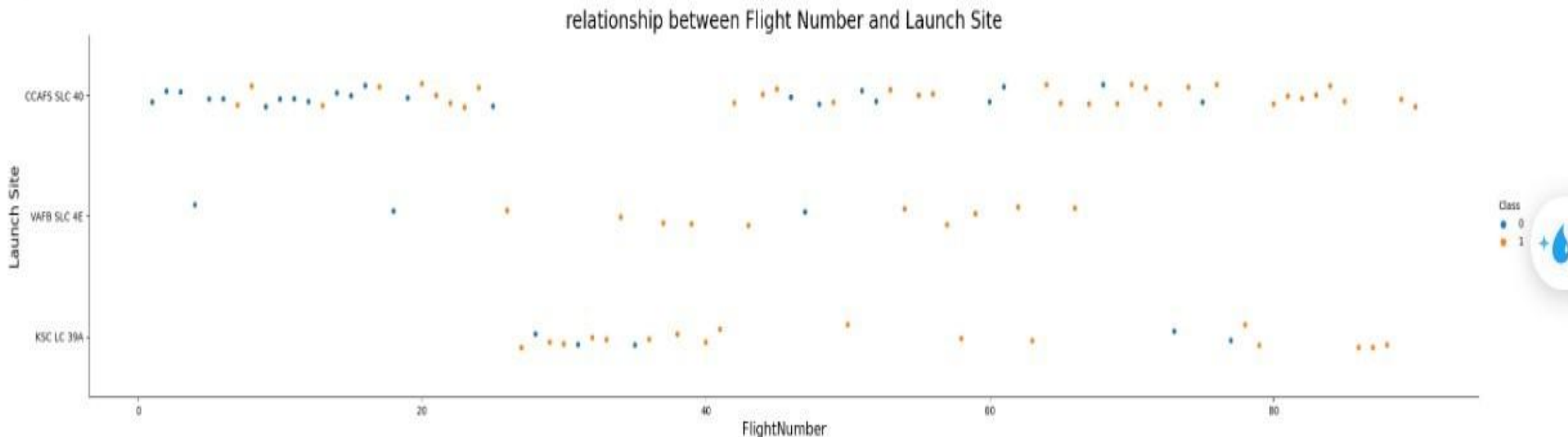
The background of the slide is an abstract composition. It features a solid blue area on the left side, which transitions into a dynamic pattern of diagonal streaks in shades of blue, red, and teal on the right. Overlaid on these streaks is a faint, semi-transparent grid pattern, giving the impression of a digital or data-driven environment.

Section 2

Insights drawn from EDA

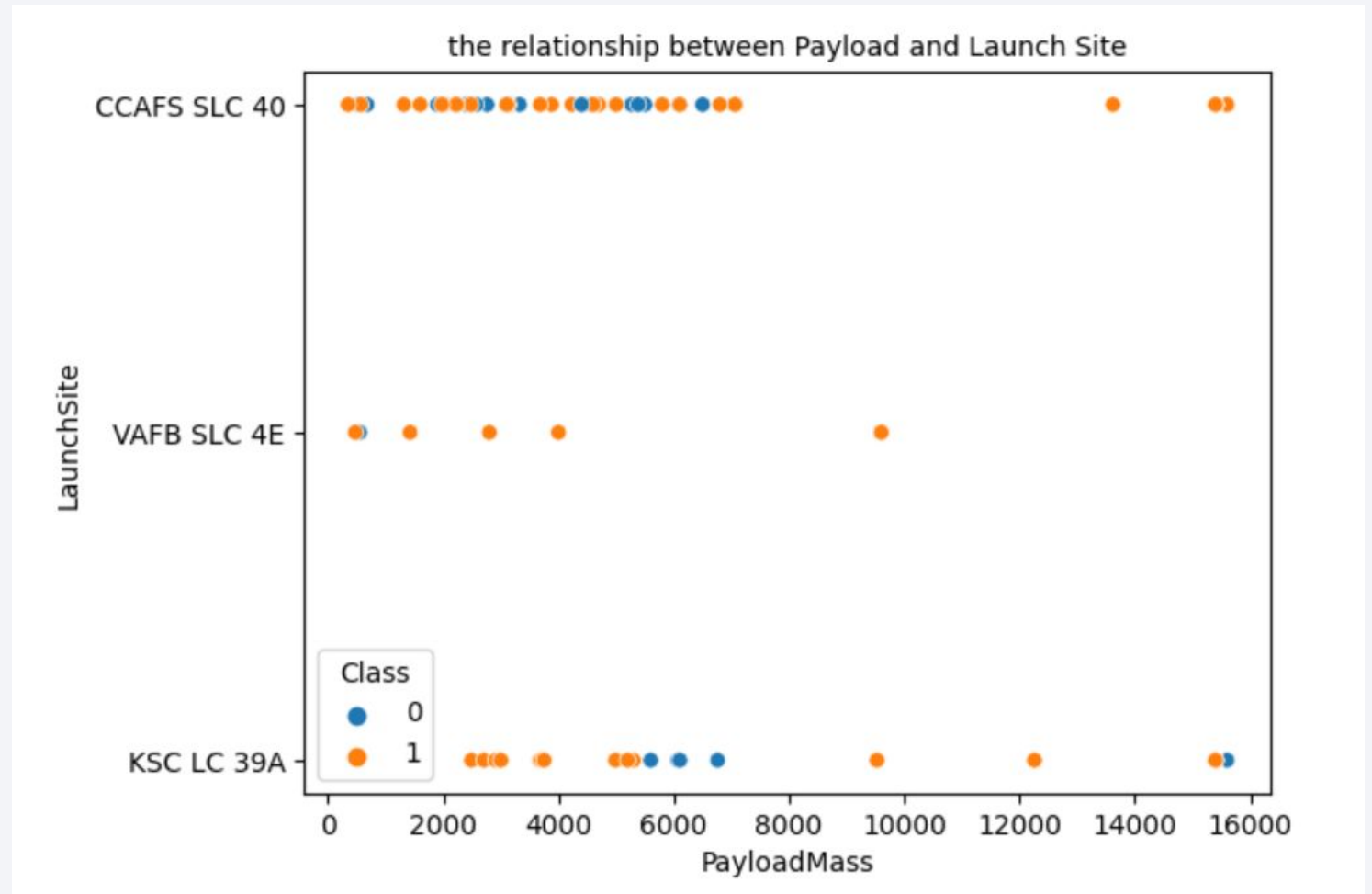
Flight Number vs. Launch Site

- Considering the plot, we found that the larger the flight amount at a launch site, the greater the success rate at a launch site.



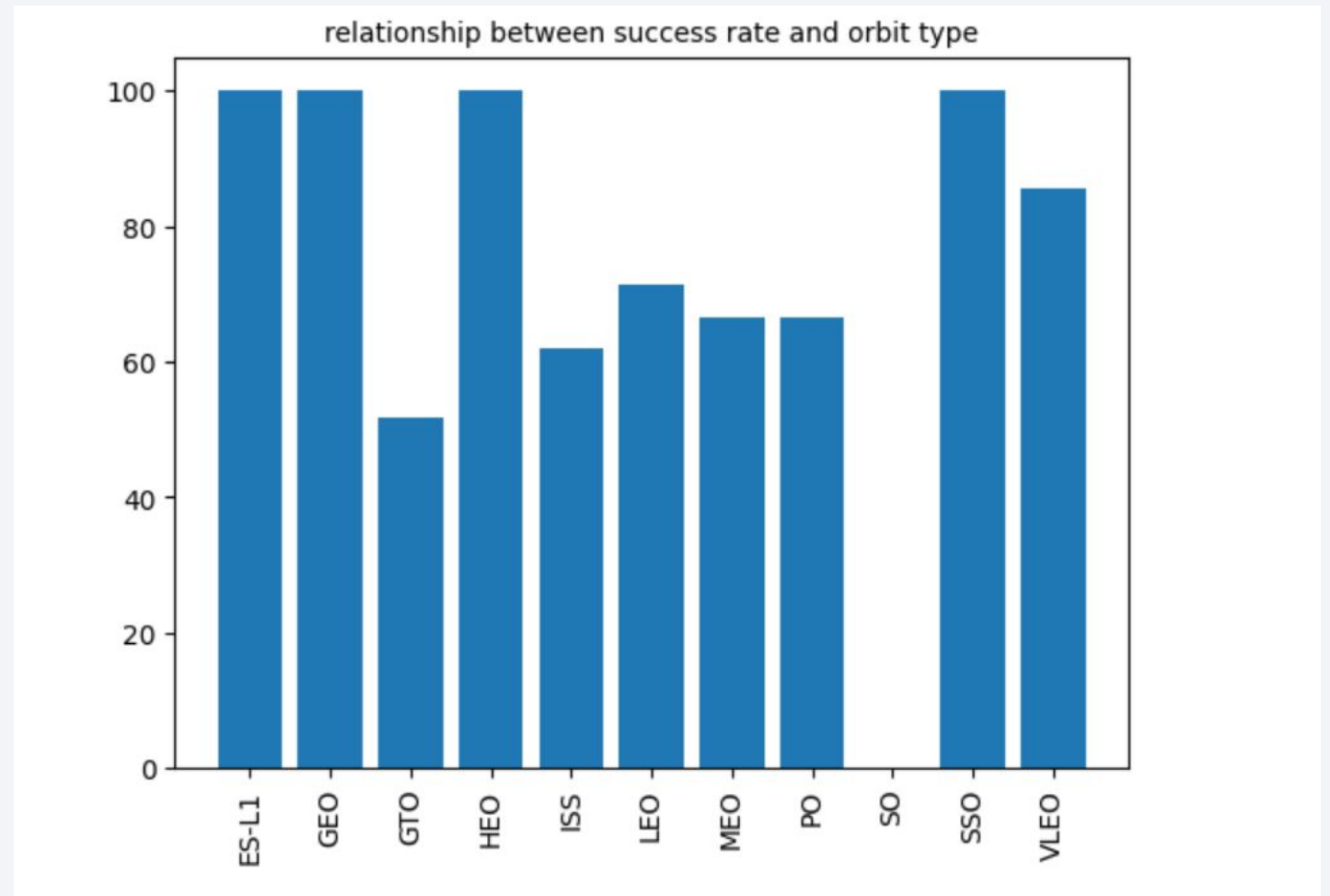
Payload vs. Launch Site

- we will find for the VAFB-SLC launchsite
- there are no rockets launched for heavy payload mass(greater than 10000)



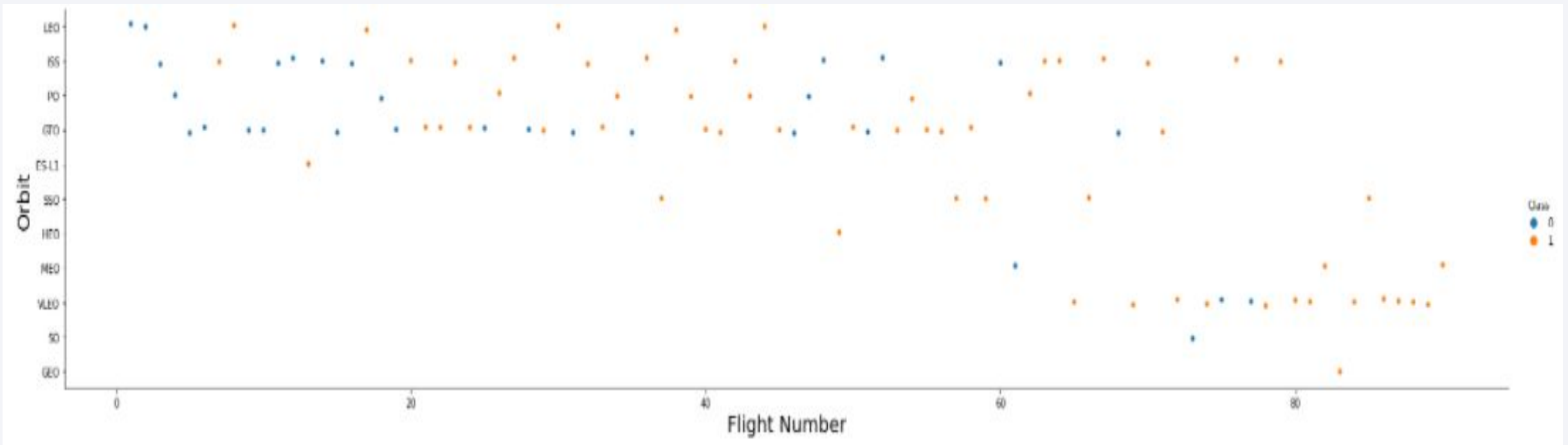
Success Rate vs. Orbit Type

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



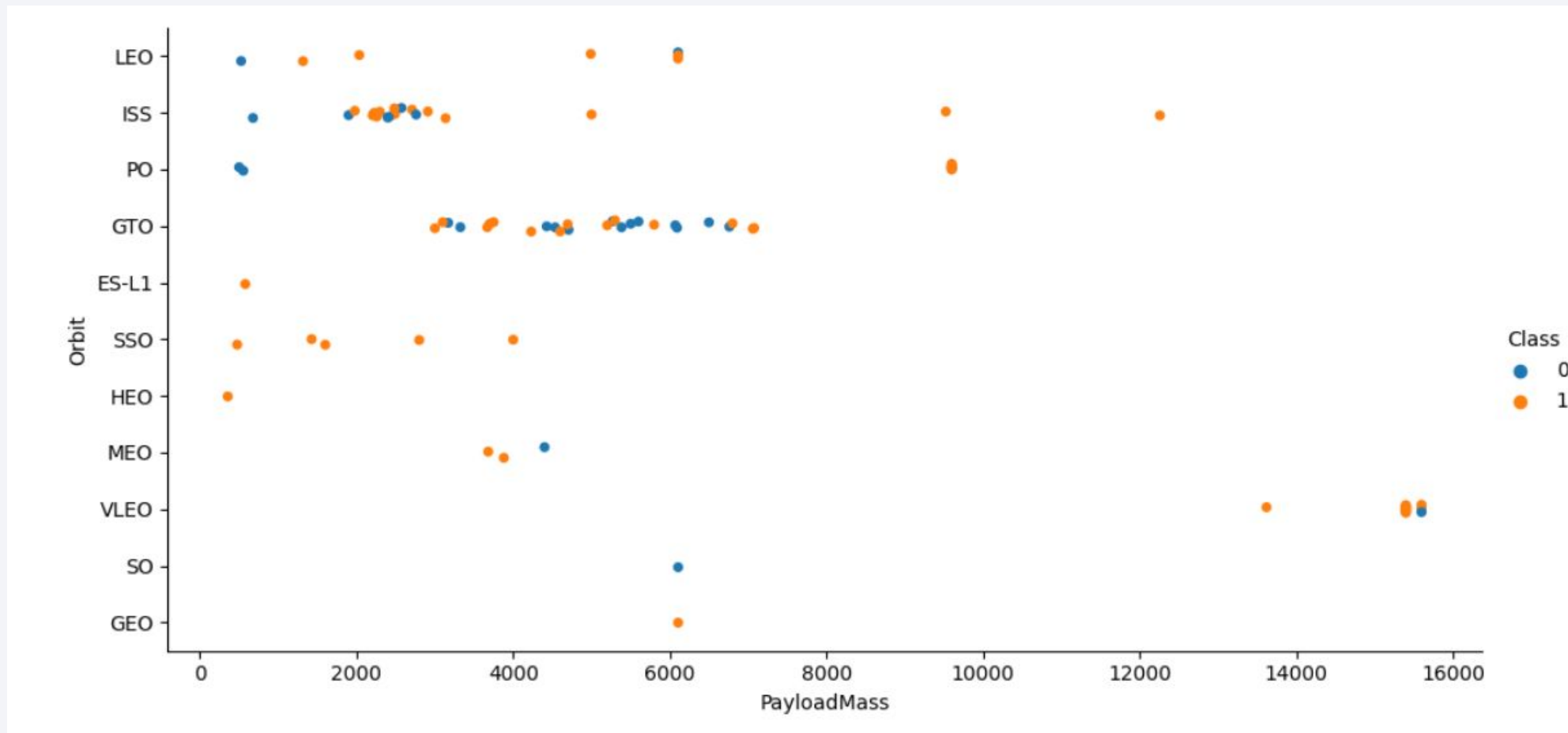
Flight Number vs. Orbit Type

- The plot below shows the Flight Number vs. Orbit type. We observe that in the LEO orbit, success is related to the number of flights whereas in the GTO orbit, there is no relationship between flight number and the orbit.



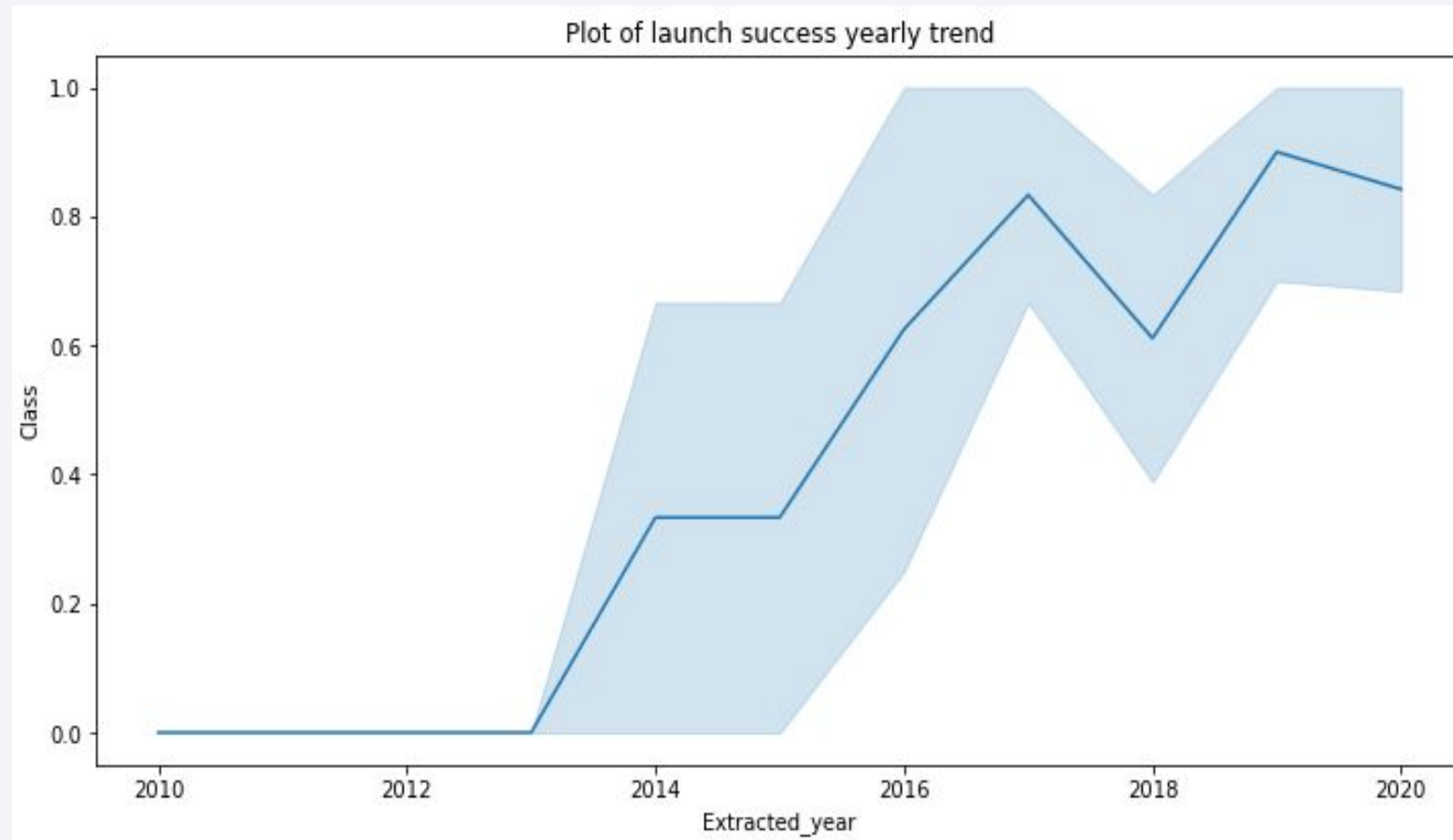
Payload vs. Orbit Type

- We can observe that with heavy payloads, the successful landing are more for PO, LEO and ISS orbits.



Launch Success Yearly Trend

Considering the plot, we can observe that success rate since 2013 kept on increasing till 2020.



All Launch Site Names

- the names of the unique launch sites :
{'CCAFS LC-40', 'CCAFS SLC-40', 'KSC LC-39A', 'VAFB SLC-4E'}

Task 1

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

```
In [10]: set(df['Launch_Site'])
```

```
Out[10]: {'CCAFS LC-40', 'CCAFS SLC-40', 'KSC LC-39A', 'VAFB SLC-4E'}
```

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with 'CCA' : CCAFS SLC-40

Task 2

Display 5 records where launch sites begin with the string 'CCA'

```
In [26]: df['Launch_Site'].value_counts()
```

```
Out[26]: CCAFS SLC-40    34  
         CCAFS LC-40    26  
         KSC LC-39A    25  
         VAFB SLC-4E    16  
         Name: Launch_Site, dtype: int64
```

```
In [29]: CCA1 = df[df['Launch_Site'].str.contains('CAFS SLC-40')]
```

```
In [ ]: CCA1.head()
```


Total Payload Mass

The total payload carried by boosters from NASA :

	PAYLOAD_MASS__KG_	Booster_Version
3	500	F9 v1.0 B0006
4	677	F9 v1.0 B0007
8	2296	F9 v1.1
12	2216	F9 v1.1 B1010
13	2395	F9 v1.1 B1012
16	1898	F9 v1.1 B1015
18	1952	F9 v1.1 B1018
22	3136	F9 FT B1021.1
26	2257	F9 FT B1025.1
29	2490	F9 FT B1031.1
34	2708	F9 FT B1035.1
38	3310	F9 B4 B1039.1

Task 3

Display the total payload mass carried by boosters launched by NASA (CRS)

In [104...

```
#by NASA (CRS)
df_CRS = df[df['Customer'].str.contains('CRS')]
```

In []:

```
df_Mass = df_CRS[['PAYLOAD_MASS__KG_', 'Booster_Version']]
df_Mass
```

Average Payload Mass by F9 v1.1

the average payload mass carried by booster version F9 v1.1: 2534.6666666666665

✓ Task 4

Display average payload mass carried by booster version F9 v1.1

✓
0s



```
df_F9 = df[df['Booster_Version'].str.contains('F9 v1.1')]  
df_F9['PAYLOAD_MASS__KG_'].mean()
```



```
2534.6666666666665
```

First Successful Ground Landing Date

The dates of the first successful landing outcome on ground pad : 2015-12-22


✓ Task 5

List the date when the first succesful landing outcome in ground pad was acheived.

Hint: Use min function

+ Code

+ Text

✓  `df['Landing_Outcome'].value_counts()`

✓ 0s `[19] df_Land = df[df['Landing_Outcome'] == 'Success (ground pad)']`

✓ 0s `[20] df_Land['Date'].min()`

`'2015-12-22'`

Successful Drone Ship Landing with Payload between 4000 and 6000

The names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000 :

{'F9 FT B1021.2', 'F9 FT B1031.2', 'F9 FT B1022', 'F9 FT B1026'}

✓ Task 6

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

```
✓ [22] df_boost = df[(df['PAYLOAD_MASS__KG_'] > 4000) & (df['PAYLOAD_MASS__KG_'] < 6000) & (df['Landing_Outcome'] == 'Success (drone ship)')]
```

```
✓ [23] set(df_boost['Booster_Version'])
```


```
{'F9 FT B1021.2', 'F9 FT B1031.2', 'F9 FT B1022', 'F9 FT B1026'}
```

Total Number of Successful and Failure Mission Outcomes

The total number of successful and failure mission outcomes is : 100

✓ Task 7

List the total number of successful and failure mission outcomes

✓  `df['Mission_Outcome'].value_counts()`

✓ 0s [25] `df_success = df[df['Mission_Outcome'].str.contains('Success')]`
`len(df_success['Mission_Outcome'])`

100

Boosters Carried Maximum Payload

The names of the booster which have carried the maximum payload mass :

{'F9 B5 B1048.4', 'F9 B5 B1060.2 ', 'F9 B5 B1051.3', 'F9 B5 B1056.4', 'F9 B5 B1051.6', 'F9 B5 B1051.4', 'F9 B5 B1058.3 ', 'F9 B5 B1049.4', 'F9 B5 B1048.5', 'F9 B5 B1060.3', 'F9 B5 B1049.5', 'F9 B5 B1049.7 '}

Task 8

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

✓
0s

```
[26] df_Load = df[df['PAYLOAD_MASS__KG_'] == df['PAYLOAD_MASS__KG_'].max()]
```

✓
0s

```
print(set(df_Load['Booster_Version']))
```

```
{'F9 B5 B1048.4', 'F9 B5 B1060.2 ', 'F9 B5 B1051.3', 'F9 B5 B1056.4', 'F9 B5 B1051.6', 'F9 B5 B1051.4
```

2015 Launch Records

The failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015 :

2015-01-10

2015-04-14

```
1 # ...

[30] df_date = df[['Date', 'Booster_Version', 'Launch_Site', 'Landing_Outcome']]

df_date['Date'] = pd.to_datetime(df_date['Date'] , format='%Y-%m-%d')

<ipython-input-31-5556fd9b828f>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stab
df_date['Date'] = pd.to_datetime(df_date['Date'] , format='%Y-%m-%d')

[32] df_2015 = df_date[df_date['Date'].dt.strftime('%Y') == '2015']

[33] df_month = df_2015[df_2015['Landing_Outcome'] == 'Failure (drone ship)']
df_month['Date']
# January
# April

13    2015-01-10
16    2015-04-14
Name: Date, dtype: datetime64[ns]
```

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

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 : 3

```
✓ 0s df['Landing_Outcome'].value_counts()
```

```
➡ Success                38
   No attempt             21
   Success (drone ship)   14
   Success (ground pad)   9
   Controlled (ocean)     5
   Failure (drone ship)   5
   Failure                3
   Failure (parachute)    2
   Uncontrolled (ocean)   2
   Precluded (drone ship) 1
   No attempt             1
   Name: Landing_Outcome, dtype: int64
```

```
✓ 0s df_new = df[(df['Date'] > '2010-06-04') & (df['Date'] < '2017-03-20') & (df['Landing_Outcome'] == 'Success (ground pad)')]
```

```
✓ 0s [36] df_new['Landing_Outcome'].count()
```

A satellite view of Earth from space, showing the curvature of the planet and city lights at night. The image is a composite of a dark blue sky with stars and a view of the Earth's surface from space. The Earth's surface is mostly dark, with a dense network of yellow and orange lights representing city lights at night. The lights are concentrated in the lower right portion of the image, following the curve of the Earth. The upper portion of the image shows the dark blue sky with a few stars.

Section 3

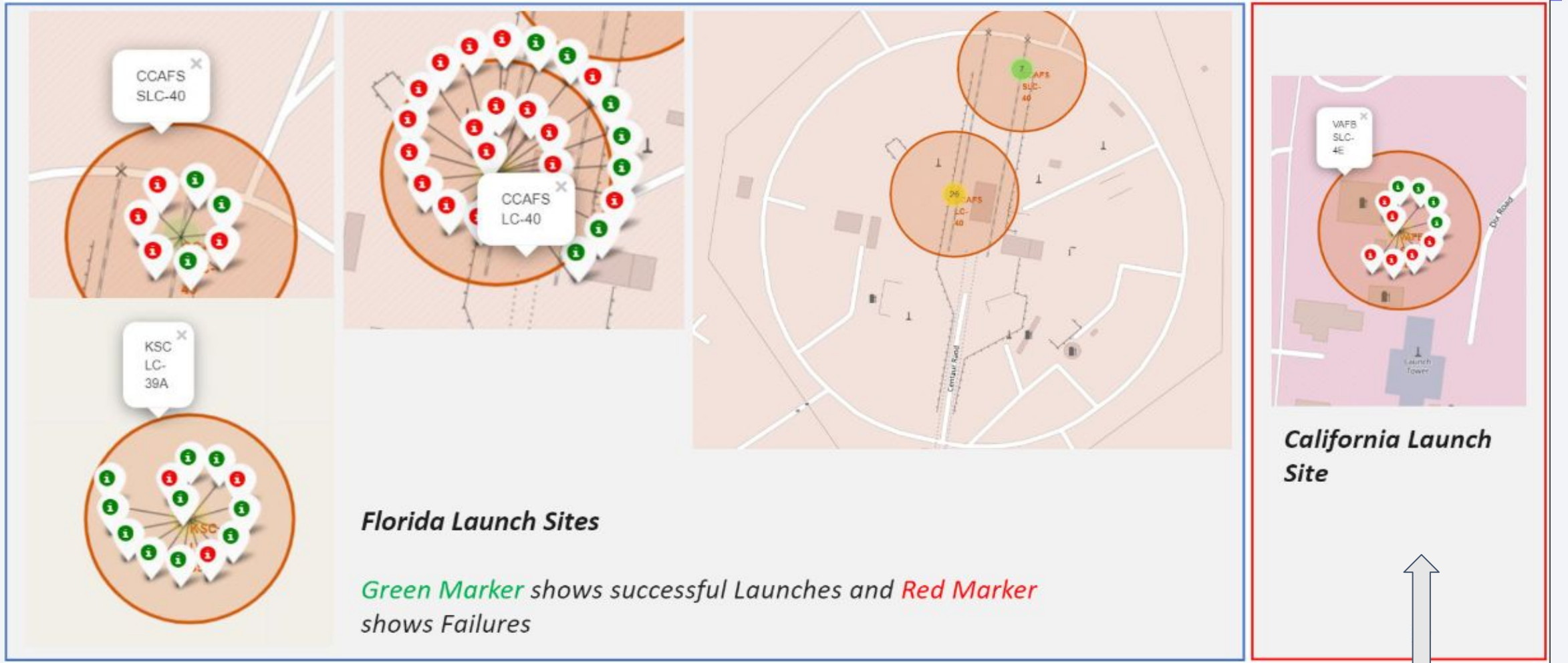
Launch Sites Proximities Analysis

<Folium Map Screenshot 1>

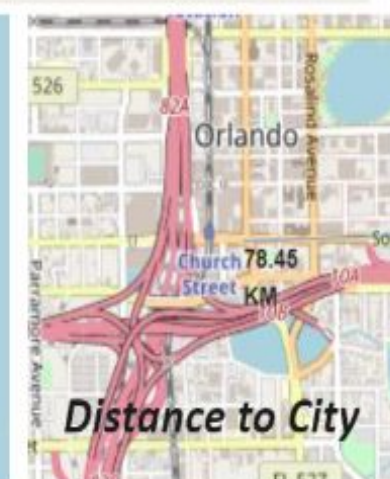
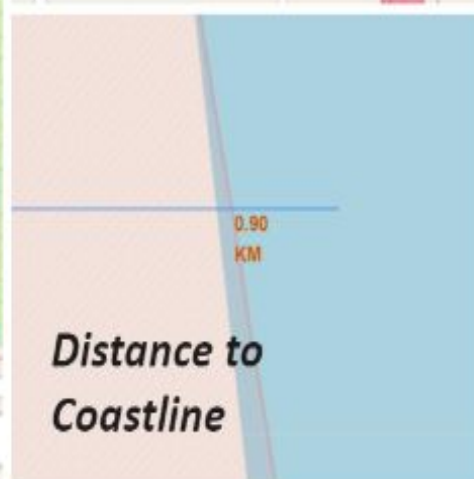
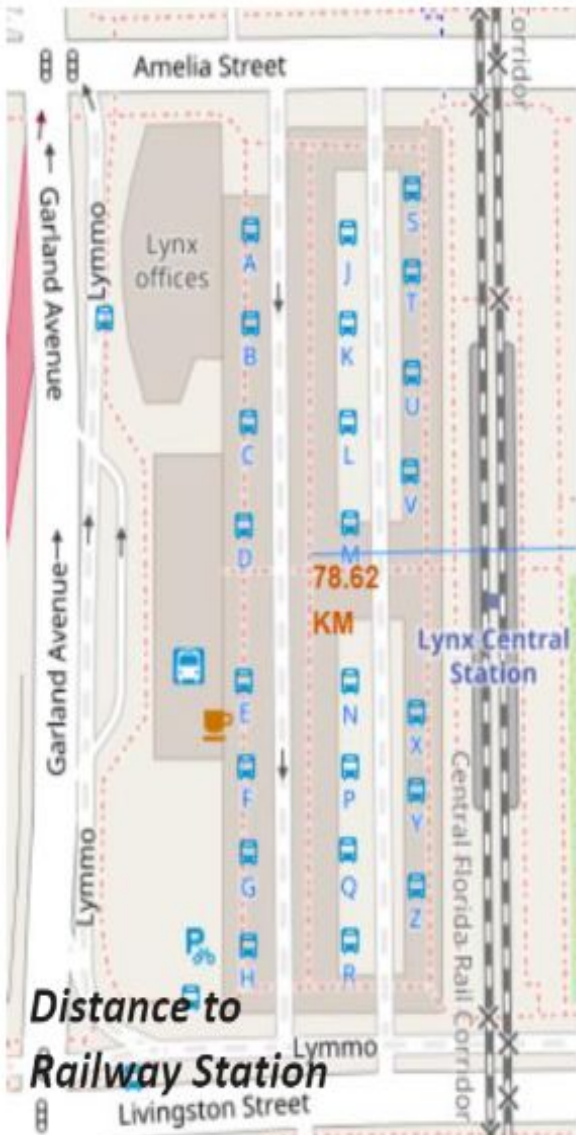
[https://github.com/imaneshi/IBM_Capstone-project_SpaceX-Falcon9/blob/main/lab_jupyter_launch_site_location_\(1\).ipynb](https://github.com/imaneshi/IBM_Capstone-project_SpaceX-Falcon9/blob/main/lab_jupyter_launch_site_location_(1).ipynb)



<Folium Map Screenshot 2>



<Folium Map Screenshot 3>



- Are launch sites in close proximity to railways? **No**
- Are launch sites in close proximity to highways? **yes**
- Are launch sites in close proximity to coastline? **yes**
- Do launch sites keep certain distance away from cities? **yes**



Section 5

Predictive Analysis (Classification)

Classification Accuracy

- The decision tree classifier is the model with the highest classification accuracy

```
[▶] bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm, 'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)
```

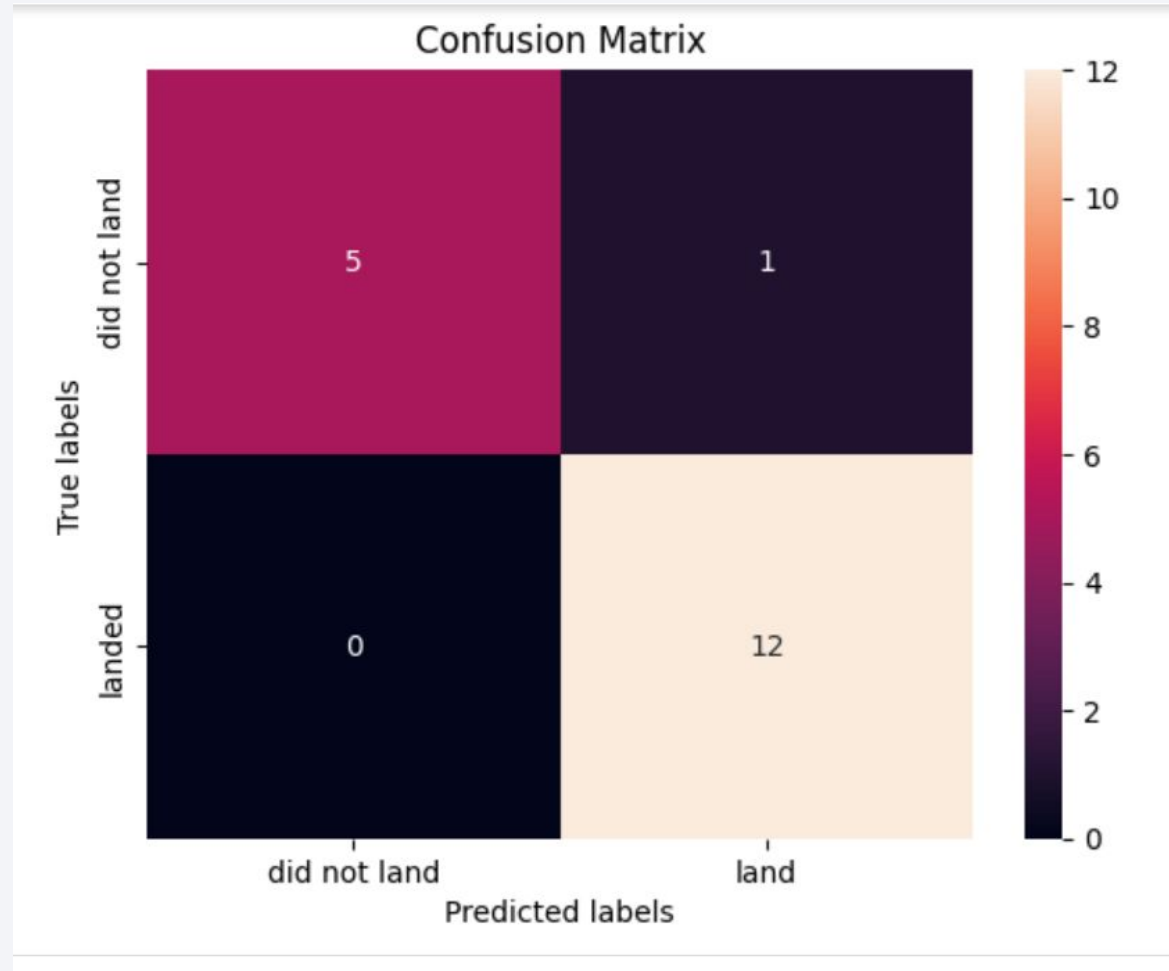
```
[->] Best model is DecisionTree with a score of 0.95
Best params is : {'criterion': 'gini', 'max_depth': 2, 'max_features': 'auto', 'min_
```

+ Code

+ Text

Confusion Matrix

- The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes. The major problem is the false positives .i.e., unsuccessful landing marked as successful landing by the classifier.



Conclusions

We can conclude that:

- The larger the flight amount at a launch site, the greater the success rate at a launch site.
- Launch success rate started to increase in 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.

Thank you!

