Winning ‘Space Race’ with Data Science

BY:

Nirab Bhattarai

10 October 2023

Outline of the presentation

- Executive Summary: What is the summary of the research?
- Introduction: Why to do the research?
- Methodology: What and how to do the research.
- Results: What were found in the research?
- Conclusion: What can we conclude from the research?

Executive Summary

- Summary of methodologies
  - We collected data from Space X API as well as from Wikipedia through Web Scraping
  - We did data cleaning, data wrangling to prepare data for analysis
  - We did exploratory data analysis in SQL, in Python with data visualization
  - We created an interactive visual analysis with Folium and created a dashboard in Dash
  - We used machine learning predictive analysis for forecasting the landing.
- Summary of all results
  - Results of EDA and data visualization using Pandas and Seaborn
  - Results of EDA using SQL
  - Results of launch sites locations analysis with Folium
  - Results of Plotly Dash dashboard
  - Results of Machine Learning analysis results

Introduction

- **Project background and context**

As a data scientist for Space Y, company owned by Allon Mask and trying to replicate the success of Space X. Space X provides its successful Falcon 9 rockets which costs just 62 million dollars as oppose to other providers whose rocket costs upward of 165 million dollars each. It is due to the savings by Space X which reuses the first stage of the launch.

Hence, by identifying if the first stage will land we can estimate the cost of a launch. This information will be useful if we want to bid against space X for a rocket launch. So, as a data scientist our goal is to create a machine learning mechanism to predict the factors that will determine that the first stage will land successfully.

- **Problems we want to find answers**
  - Factors that determine if first stage of the rocket will land successfully?
  - The relation between features such as launch site, payload etc. which play a vital part in a successful landing.
  - What other conditions are needs to be in place for a successful landing.

Methodology

- We collected data from Space X API as well as from Wikipedia through Web Scraping
- We did data cleaning, data wrangling to prepare data for analysis
- We did exploratory data analysis in SQL, in Python with data visualization
- We created an interactive visual analysis with Folium and created a dashboard in Plotly Dash
- We used machine learning predictive analysis for forecasting the landing.

Data Collection

- The raw rocket launch data were collected from SpaceX API.
- The response contents were decoded as Json using .json() function call and were fed into a pandas dataframe using .json_normalize() function.
- The data were cleaned and missing values treated as required; i.e. were removed when not necessary and were filled with appropriate values where necessary.
- Data were also collected from Wikipedia through web scrapping using BeautifulSoup.

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/BNirab/Assignment/blob/ca5d1969cbd46509f00c3c55a5c23286262a57a6/jupyter-labs-spacex-data-collection-api.ipynb.

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

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

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

Check the content of the response

```
[11]: #print(response.content)
```

You should see the response contains massive information about SpaceX launches. Next, let's try to discover some more relevant information for this project.

Task 1: Request and parse the SpaceX launch data using the GET request

To make the requested JSON results more consistent, we will use the following static response object for this project:

```
[9]: static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

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

```
[10]: response.status_code
```

```
[10]: 200
```

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

```
[14]: # Use json_normalize meethod to convert the json result into a dataframe
data=pd.json_normalize(response.json())
```

Data Collection - Scraping

- We used web scrapping from Wikipedia to webscrap Falcon 9 launch records with BeautifulSoup and parsed the table and fed it into a pandas dataframe.
- The link to the notebook is : https://github.com/BNirab/Assignment/blob/ca5d1969cbd46509f00c3c55a5c23286262a57a6/jupyter-labs-webscraping.ipynb

[87]:
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

Next, request the HTML page from the above URL and get a `response` object

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.

[88]:
```
# use requests.get() method with the provided static_url
response = requests.get(static_url)
# assign the response to a object
```

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

[89]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.text, "html.parser")
```

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

[90]:
```
# Use soup.title attribute
soup.title
```

[90]:
```
<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

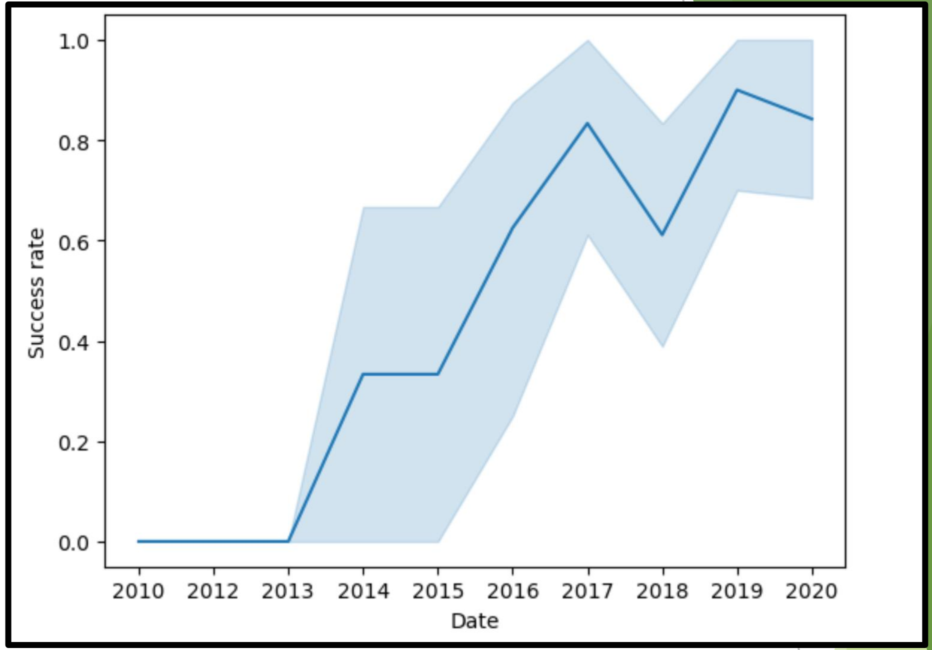
[91]:
```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables=soup.find_all('table')
```

Data Wrangling

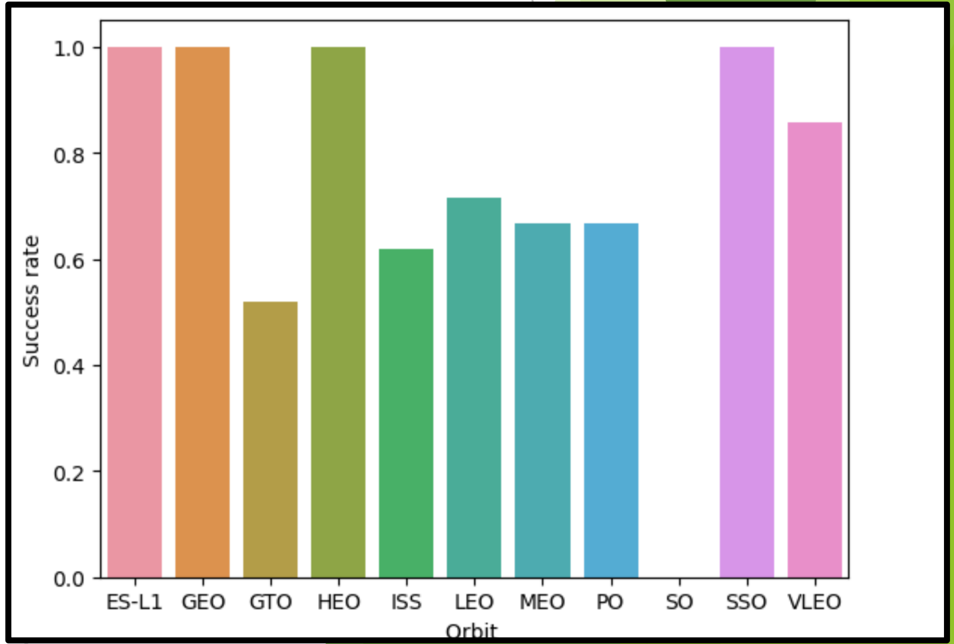
- We performed data wrangling and exploratory data analysis to determined the training labels.
- We calculated the number of null values in the data.
- We calculated the number of launches at each site .
- We calculate the number and occurrences of mission outcome per orbit type
- We created landing outcome label from outcome column and exported the results to csv.
- The link to the notebook is https://github.com/BNirab/Assignment/blob/ca5d1969cbd46509f00c3c55a5c23286262a57a6/IBM-DS0321EN-SkillsNetwork_labs_module_1_L3_labs-jupyter-spacex-data_wrangling_jupyterlite.jupyterlite.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 and
  - The launch success yearly trend .



- The link to the notebook is https://github.com/BNirab/Assignment/blob/5fa2e7f3970c06a7ec9908a360d47d83804fc6f8/IBM-DS0321EN-SkillsNetwork_labs_module_2_jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb



EDA with SQL

- We loaded the SpaceX dataset into a SQL database.
- We applied EDA with SQL to get insight from the data by writing queries 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/BNirab/Assignment/blob/5fa2e7f3970c06a7ec9908a360d47d83804fc6f8/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- We marked all four launch sites and added map objects such as markers, circles and lines to mark each site on the folium map.
- We assigned the launch outcomes failure for class 0 and success for class 1.
- We used colored marker clusters to insert all the success and failure to the sites to find which launch sites have 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/BNirab/Assignment/blob/5fa2e7f3970c06a7ec9908a360d47d83804fc6f8/IBM-DS0321EN-SkillsNetwork_labs_module_3_lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

- We built an interactive dashboard using plotly's Dash
- We plotted pie charts showing the total launches by a certain sites
- We plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.
- The link to the notebook is https://github.com/BNirab/Assignment/blob/5fa2e7f3970c06a7ec9908a360d47d83804fc6f8/spacex_dash_app.py

Predictive Analysis

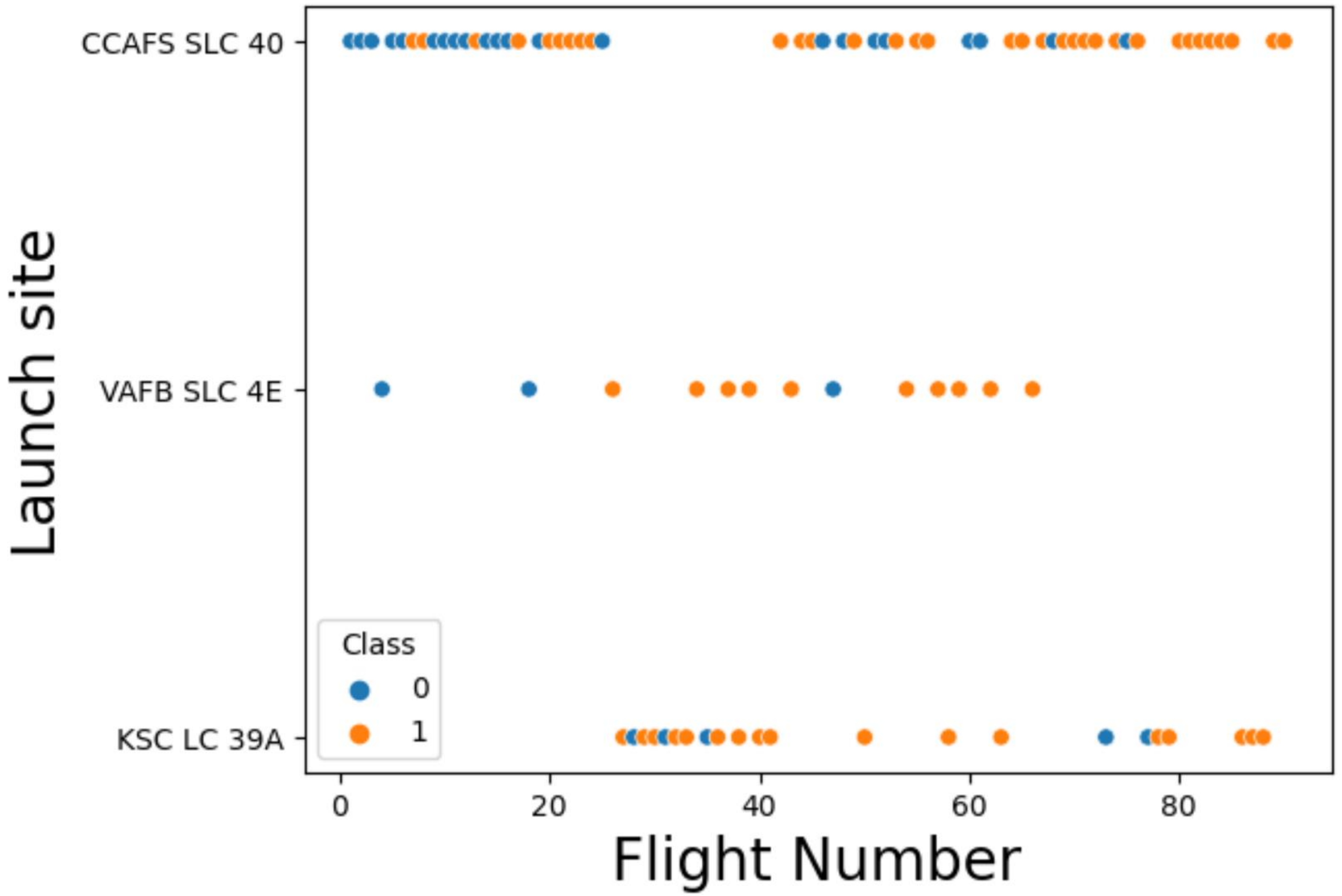
- We loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- We built different machine learning models and tune different hyperparameters.
- We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- We found the best performing classification model.
- The link to the notebook is : https://github.com/BNirab/Assignment/blob/5fa2e7f3970c06a7ec9908a360d47d83804fc6f8/IBM-DS0321EN-SkillsNetwork_labs_module_4_SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Results

- The results of the analysis are presented in following sections
  - EDA and data visualization using Pandas and Seaborn
  - EDA using SQL
  - Launch Sites Locations Analysis with Folium
  - Plotly Dash results
  - Machine Learning analysis results

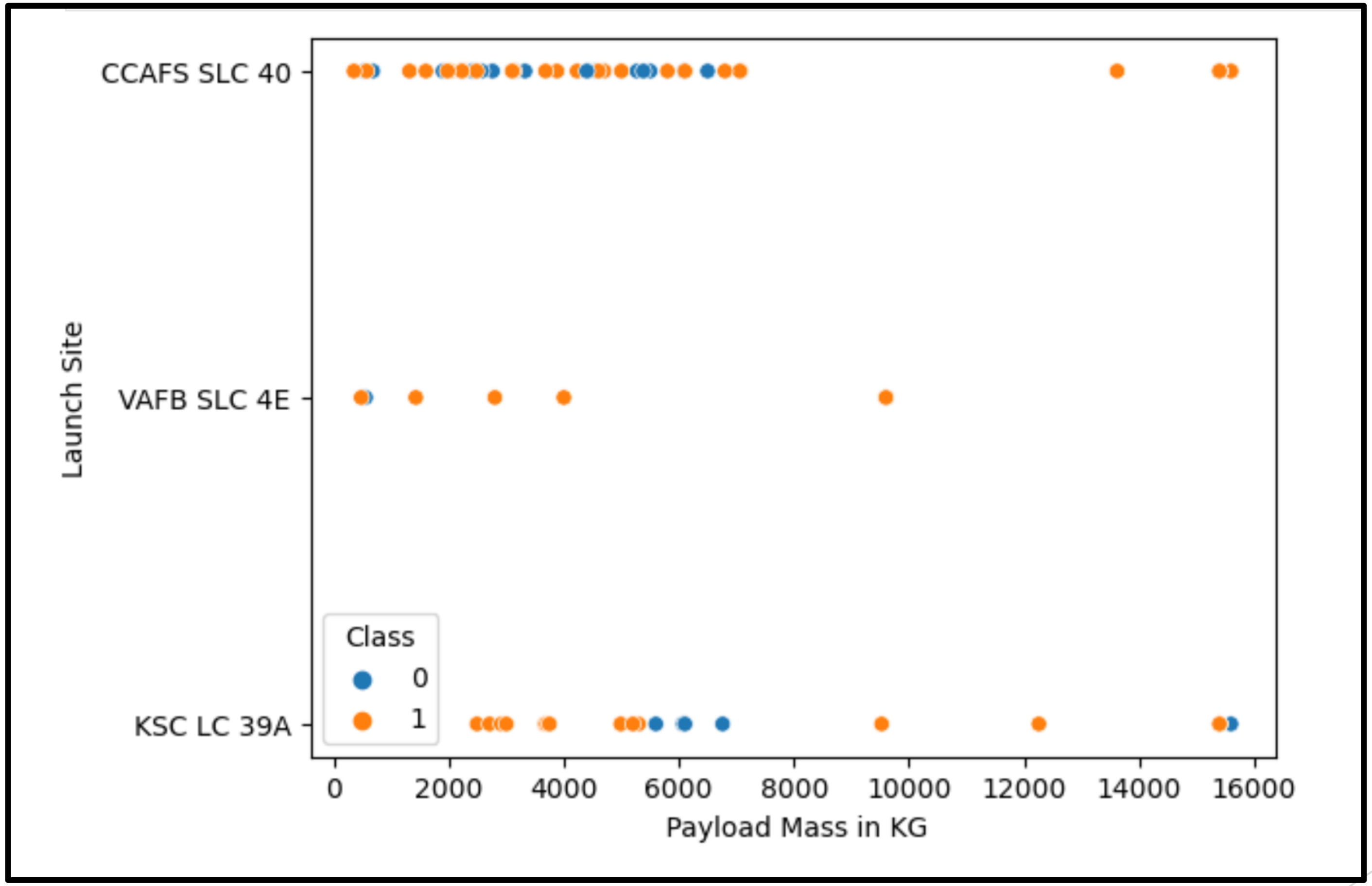
EDA and data visualization using Pandas and Seaborn

- We found that the success rate at a launch increases with number of flight from that launch site.



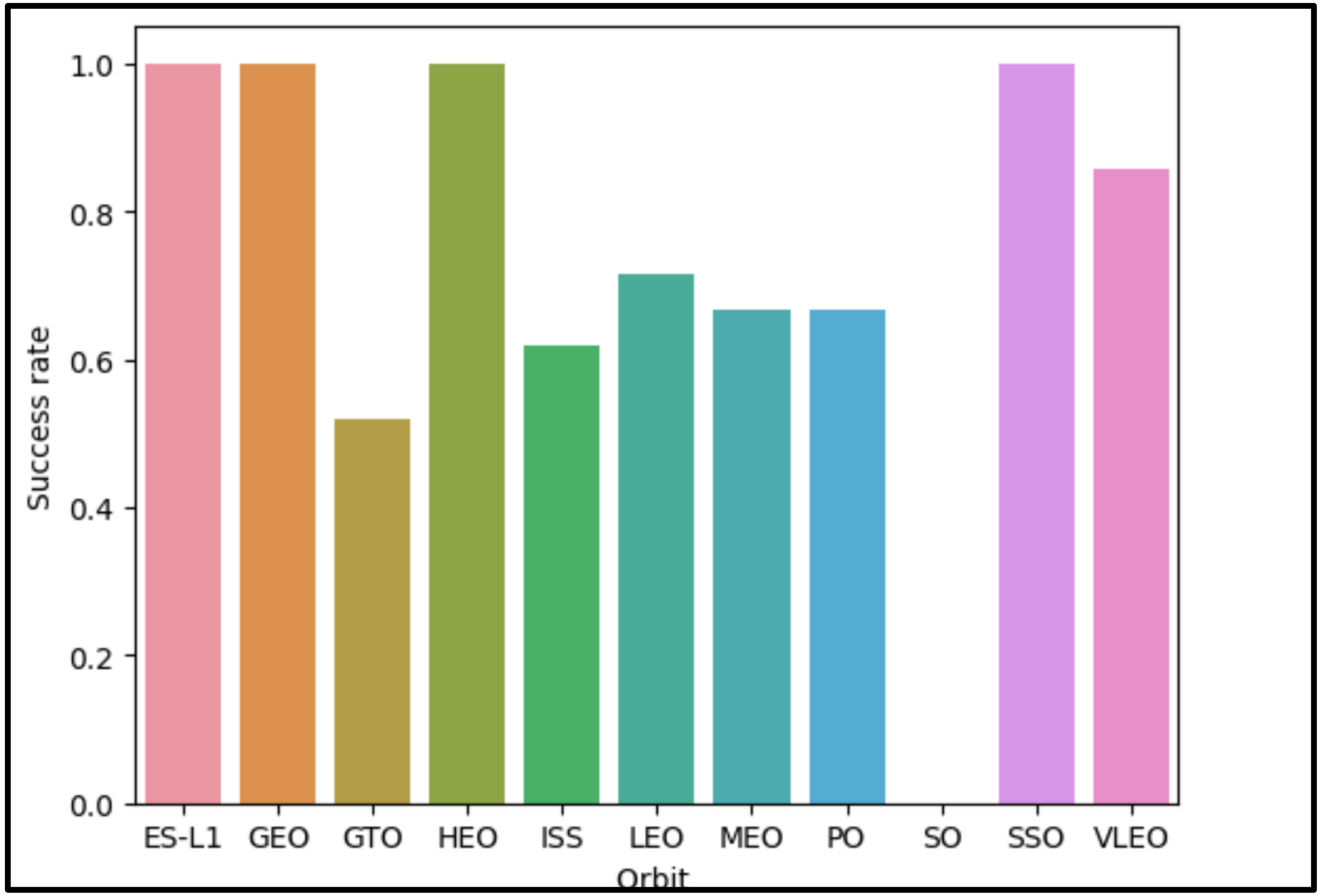
EDA and data visualization using Pandas and Seaborn

- The success rate for higher payload mass is higher than success rate for lower payload mass



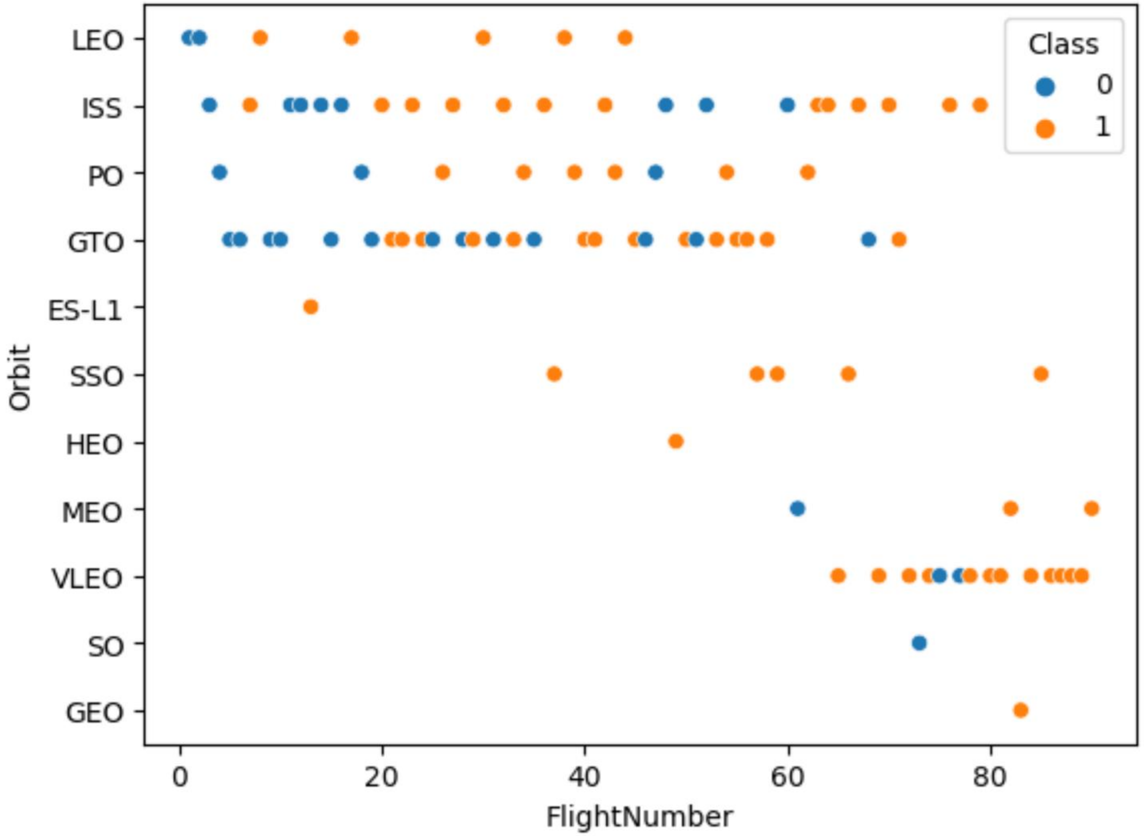
EDA and data visualization using Pandas and Seaborn

- The orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate and SO had the least.



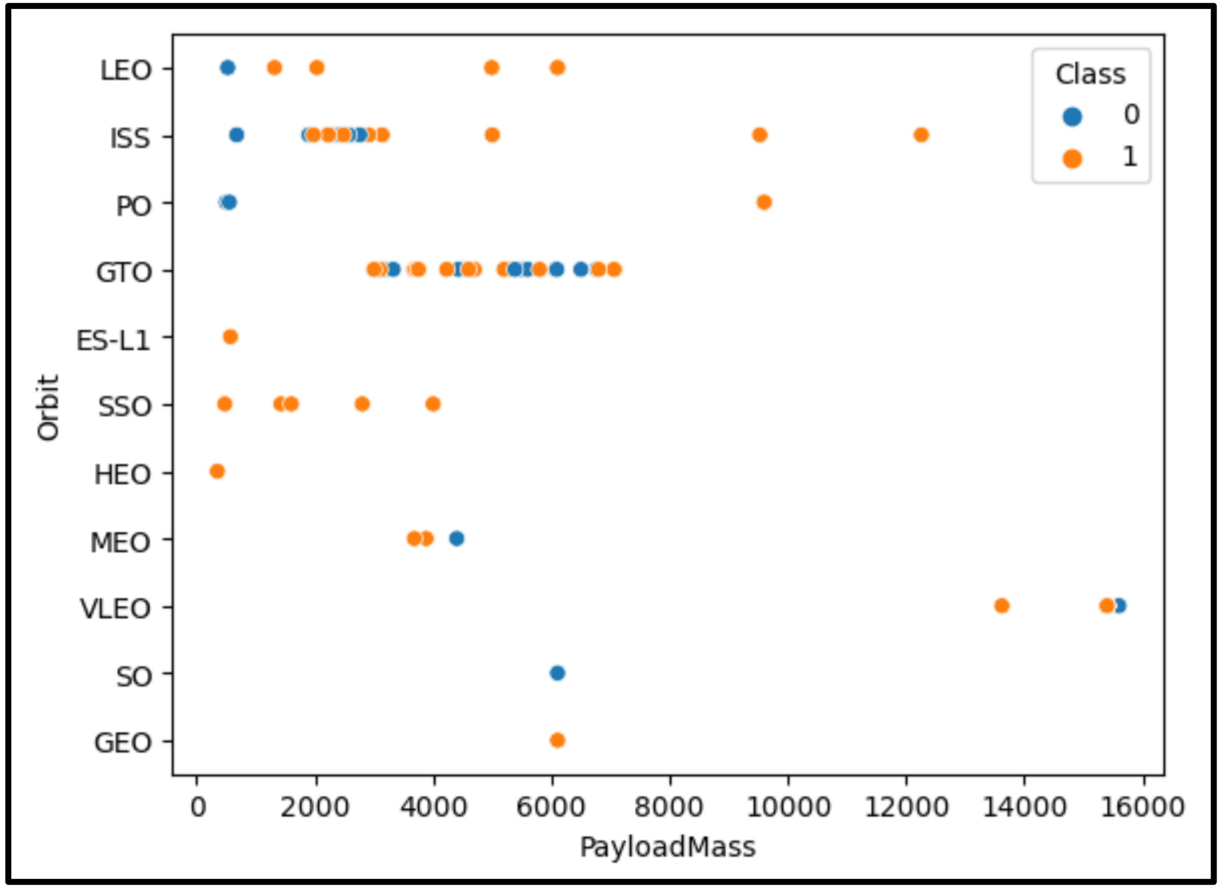
EDA and data visualization using Pandas and Seaborn

- We can see from the plot that for LEO orbit success is related to the number of flights but for others, there are no distinct relationship.



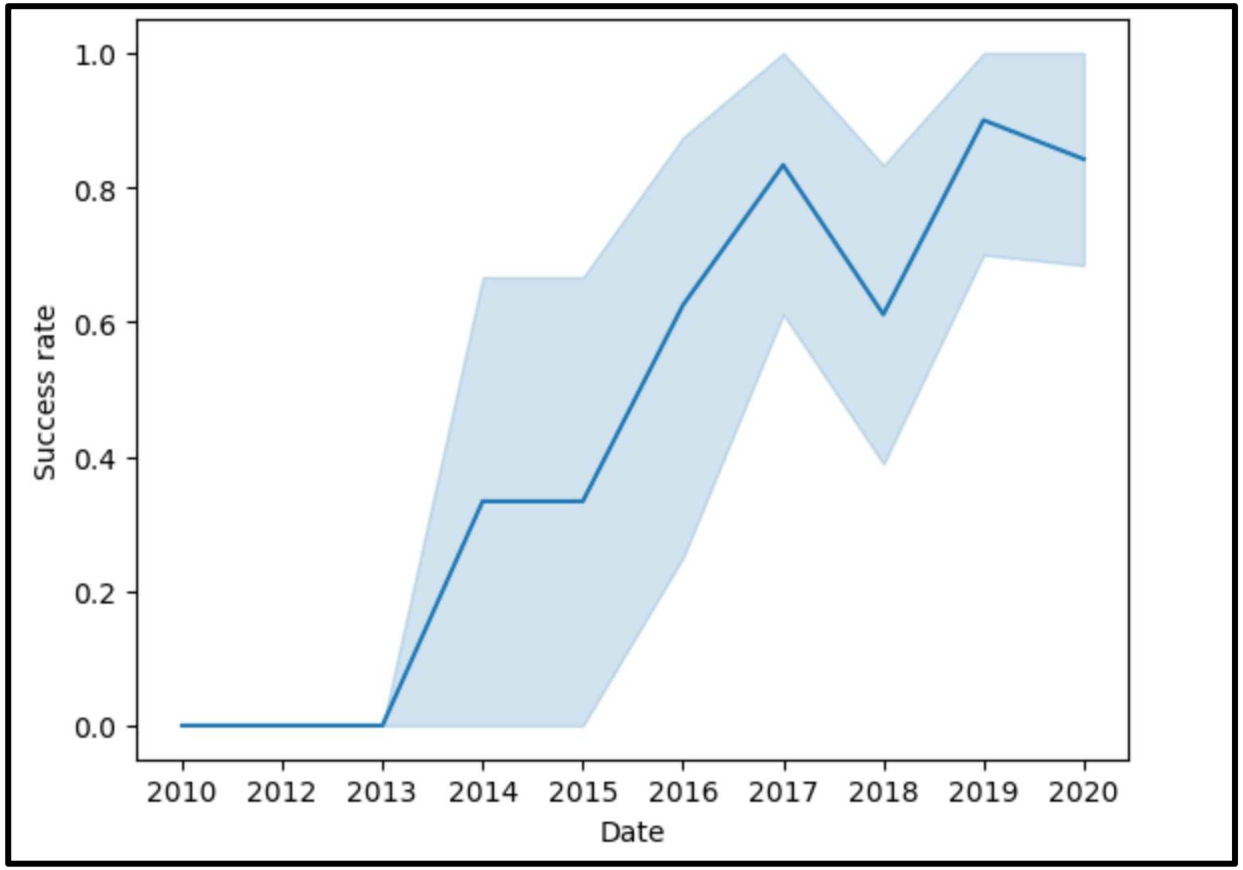
EDA and data visualization using Pandas and Seaborn

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



EDA and data visualization using Pandas and Seaborn

- The launch success yearly trend shows that the success rate started increasing from 2013 to 2017 took a plunge and again has started taking off..



EDA using SQL

- We found unique launch sites from the SpaceX data using SELECT DISTINCT statement .

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

In [21]:
```
%sql select distinct("Launch_Site") from SPACEXTABLE
```

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

Out[21]:

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

EDA using SQL

- We wrote query to display 5 records where launch sites begin with `CCA`

Task 2

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

In [23]:
```
%sql select * from SPACEXTABLE where "Launch_Site" like "CCA%" limit 5
```

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

Out[23]:

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

EDA using SQL

- We queried the total payload carried by boosters from NASA as 45596 which was total of 45,596.0 kg.

Task 3

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

In [25]:
```
%sql select sum("PAYLOAD_MASS__KG_") as Payload_mass from SPACEXTABLE where Customer="NASA (CRS)"
```

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

Out[25]:

Payload_mass
45596

EDA using SQL

- We found the average payload mass carried by F9 v1.1 booster version was 2534.6 kg.

Task 4

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

In [26]:
```
%sql select avg("PAYLOAD_MASS__KG_") as Average from SPACEXTABLE where "Booster_Version" like "%v1.1%"
```

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

Out[26]:

Average
2534.6666666666665

EDA using SQL

- We found that the first successful landing outcome on ground pad was on 22nd December 2015.

Hint:Use min function

```
In [31]: %sql select Date from SPACEXTABLE where "Landing_Outcome" like "%Success (ground pad)%" limit 1

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

Out[31]:

Date
2015-12-22

EDA using SQL

- We used the **WHERE** clause to filter for boosters which have successfully landed on drone ship and applied the **AND** condition to determine successful landing with payload mass greater than 4000 but less than 6000

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

In [32]:
```
%sql select "Booster_Version" from SPACEXTABLE where ("Landing_Outcome" like "%Success (drone ship)%") and ("PAYLOAD_MASS__
```

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

Out[32]:

Booster_Version
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

EDA using SQL

- We found that there were 100 successful mission outcomes and one failure .

Task 7

List the total number of successful and failure mission outcomes

In [40]:

```
X1=%sql select count("Mission_Outcome") as Successful from SPACEXTABLE where "Mission_Outcome" like "Success%"

X2=%sql select count("Mission_Outcome") as failure from SPACEXTABLE where "Mission_Outcome" not like "%Success%"
print(X1,X2)
```

```
 * sqlite:///my_data1.db
Done.
 * sqlite:///my_data1.db
Done.
+------------+
| Successful |
+------------+
|    100     |
+------------+ +---------+
| failure |
+---------+
|    1    |
+---------+
```

EDA using SQL

- We determined the booster that have carried the maximum payload using a subquery in the **WHERE** clause and the **MAX()** function.

Task 8

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

In [41]:
```
%sql select distinct("Booster_Version") from SPACEXTABLE where "PAYLOAD_MASS__KG_"= (select max("PAYLOAD_MASS__KG_") from SP
```

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

Out[41]:

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

EDA using SQL

- We used a combinations of the **WHERE** clause, **LIKE**, **AND**, and **BETWEEN** conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015.

Task 9

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.

In [66]:
```
%sql select substr(Date,6, 2) as Month, "Landing_Outcome", "Booster_Version", "Launch_Site" from SPACEXTABLE where (Date lik
```

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

Out[66]:

Month	Landing_Outcome	Booster_Version	Launch_Site
10	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

EDA using SQL

- We selected Landing outcomes and the **COUNT** of landing outcomes from the data and used the **WHERE** clause to filter for landing outcomes **BETWEEN** 2010-06-04 to 2010-03-20.
- We applied the **GROUP BY** clause to group the landing outcomes and the **ORDER BY** clause to order the grouped landing outcome in descending order.

Task 10

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.

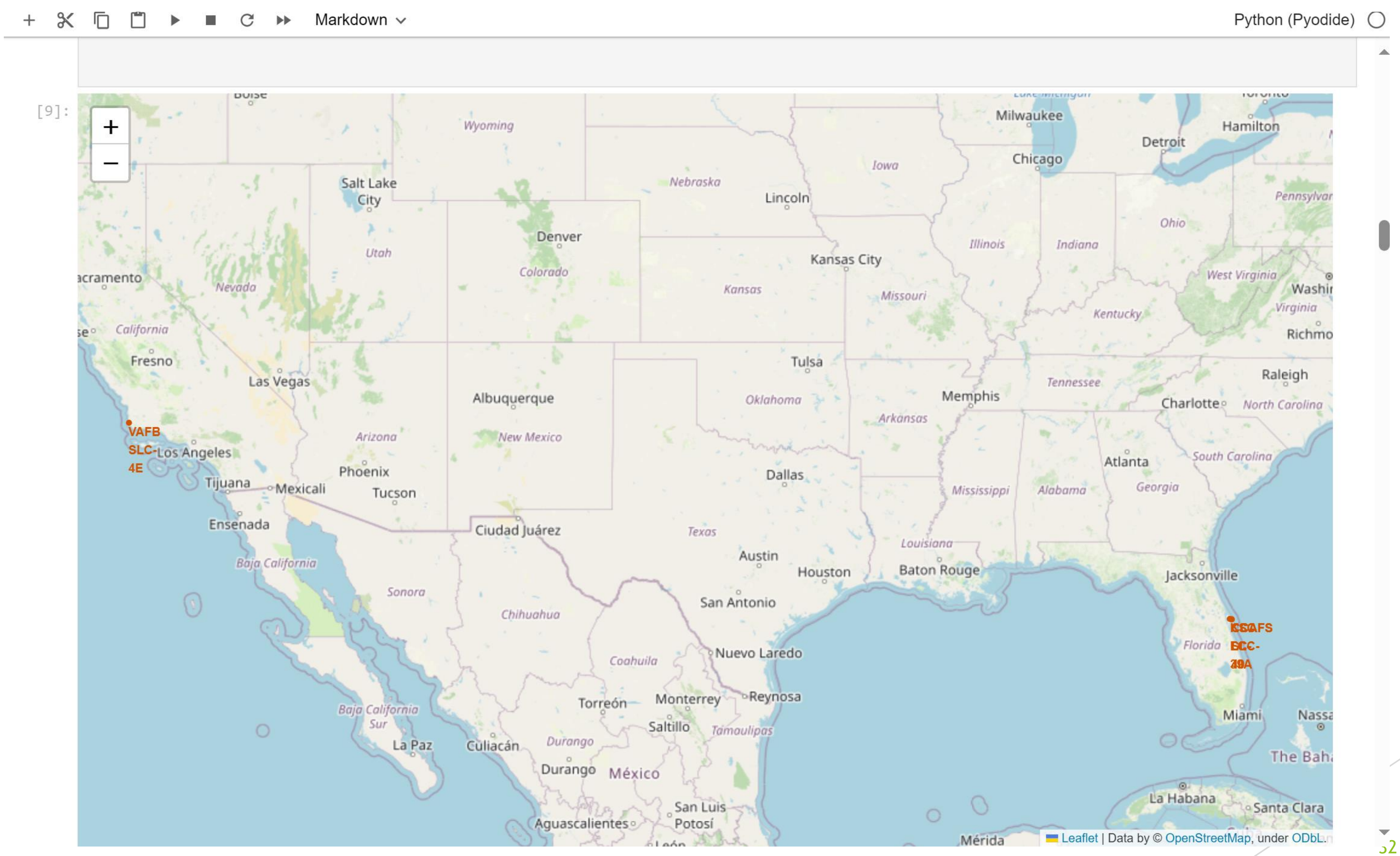
In [76]:
```
%sql select "Landing_Outcome", count(*) as Number from SPACEXTABLE where (Date>='2010-06-04' and Date<='2017-03-20') group b
```

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

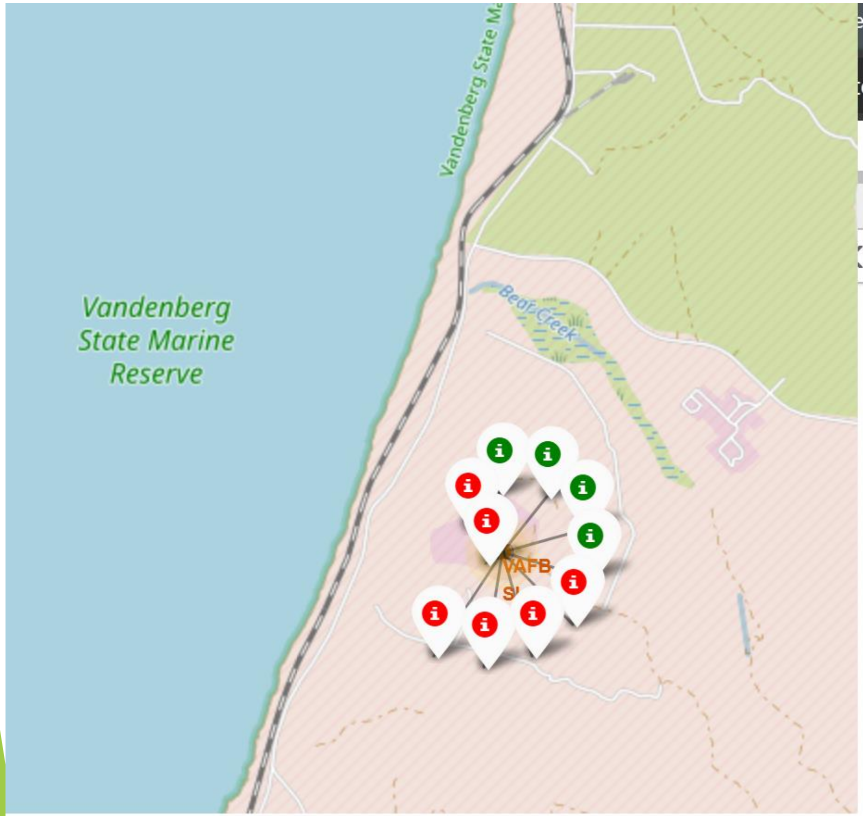
Out[76]:

Landing_Outcome	Number
No attempt	10
Success (ground pad)	5
Success (drone ship)	5
Failure (drone ship)	5
Controlled (ocean)	3
Uncontrolled (ocean)	2
Precluded (drone ship)	1
Failure (parachute)	1

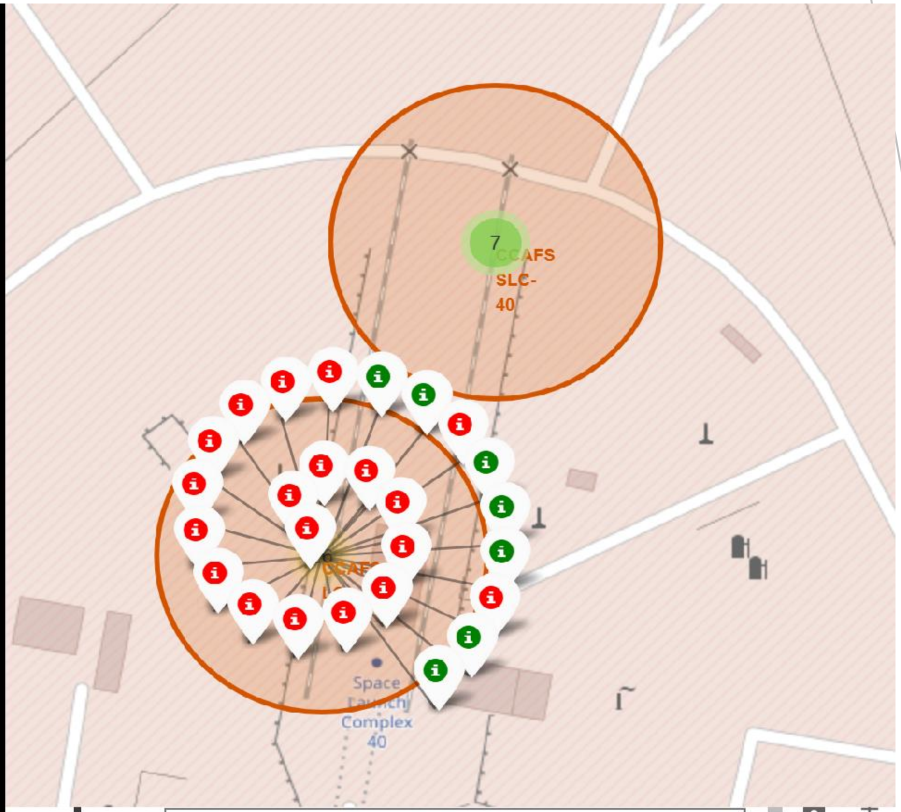
Launch Sites Locations Analysis with Folium



Markers showing launch sites with color labels

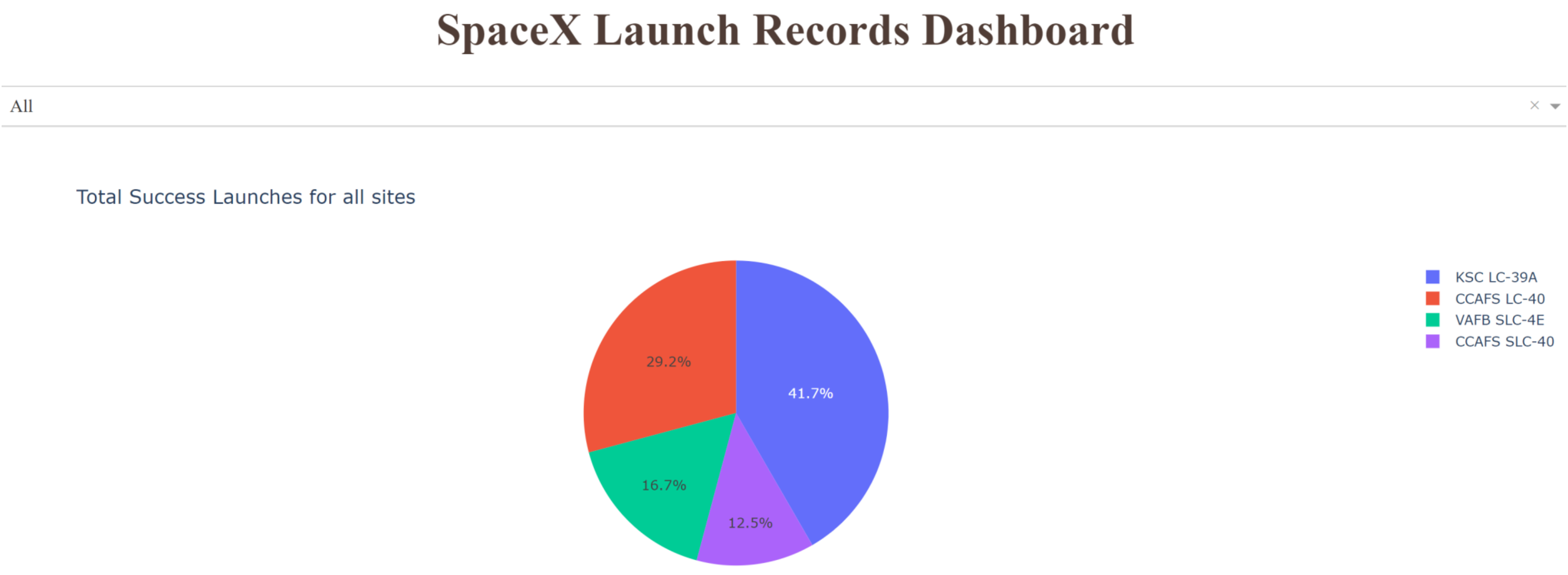


California Site

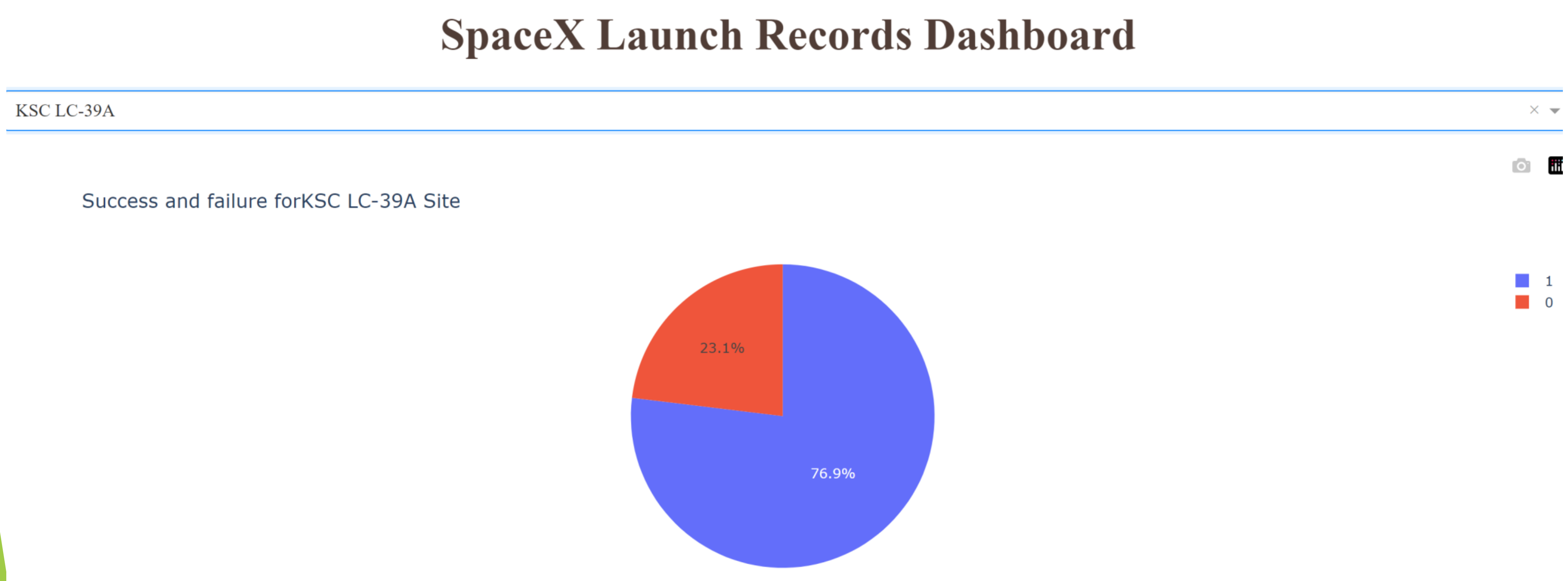


Florid Sites

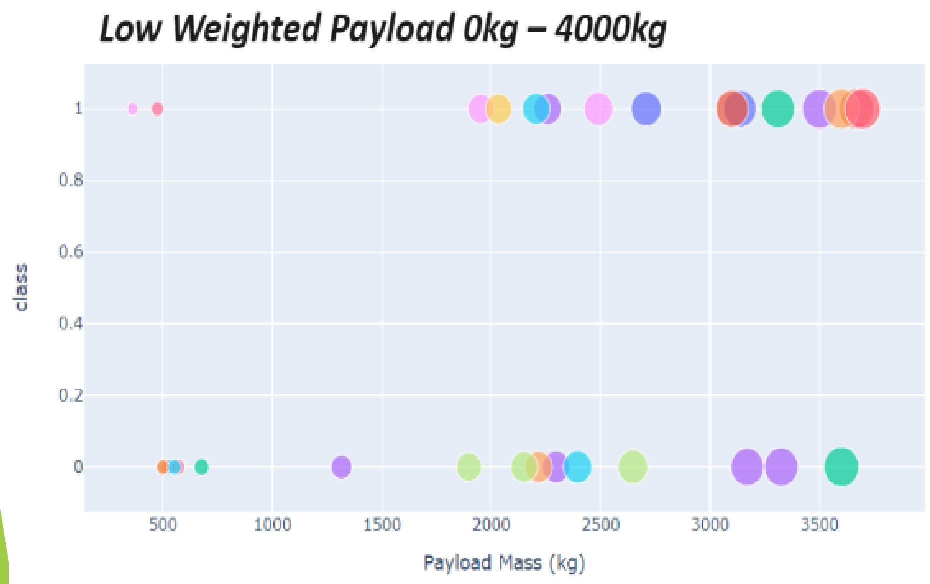
Plotly Dash results

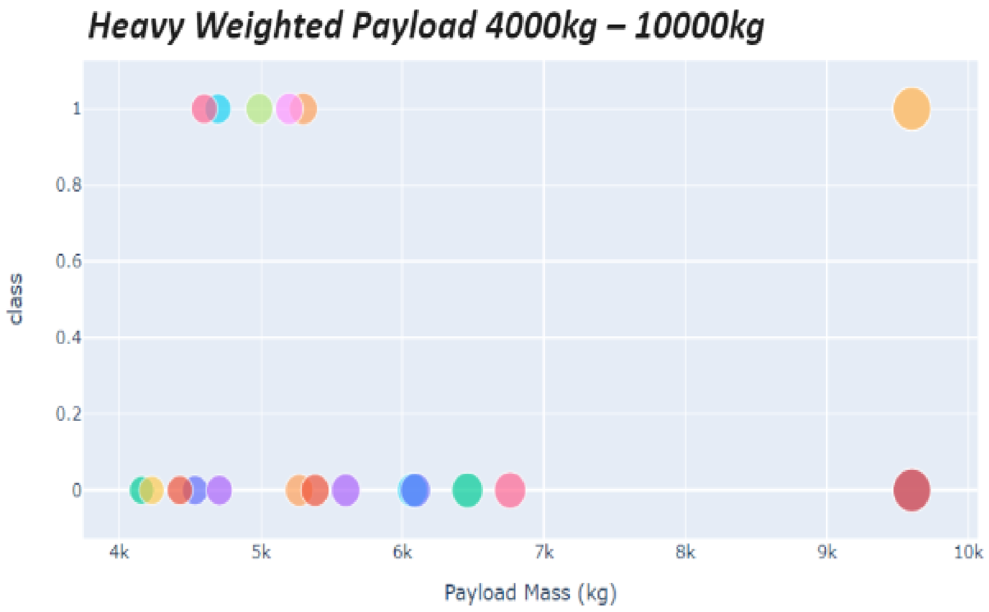


Plotly Dash results



Plotly Dash results





Machine Learning analysis results

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

TASK 12

Find the method performs best:

In [29]:

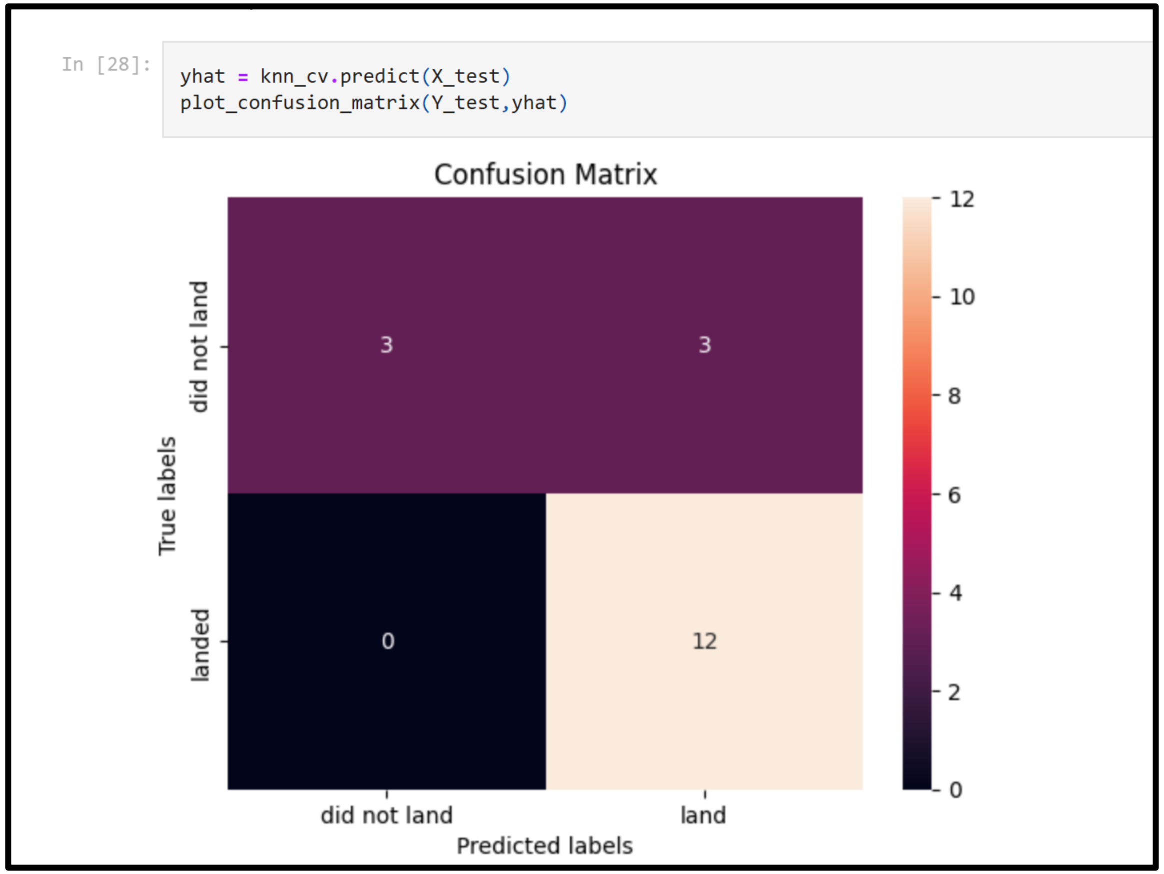
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

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 0.8732142857142856
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5,
'splitter': 'random'}
```

Machine Learning analysis results

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



Conclusions

From all the analysis 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 increased from 2013 to 2017 and dipped and again took off 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.