Data Science Project Space X

ambalmori

June 2023

OUTLINE

EXECUTIVE SUMMARY

- Methodologies
  - Data collection with API
  - Data collection with Web scraping
  - Data wrangling
  - Data Analyzing
  - Data Visualization
  - Launch Sites Analysis
  - Landing Prediction

- Results
  - Exploratory Data Analysis
  - Visual Analytics
  - Predictive Analysis

INTRODUCTION

- SpaceX advertises 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 **SpaceX can reuse the first stage**
- **Objective:** predict if the Falcon 9 first stage will land successfully
    - Determine if the first stage will land
    - Determine the cost of a launch

METHODOLOGY

- Data Collection
    - With API
    - With Web Scraping
- Data Wrangling
- Data Analyzing
- Data Visualization
- Launch Sites Analysis
- Landing Prediction

DATA COLLECTION

- With API

  We made a get request to the SpaceX API.

  1. We defined a series of helper functions that helped us use the API to extract information using identification numbers in the launch data.
  2. We requested rocket launch data from SpaceX API.
  3. To make the requested JSON results more consistent, we used a static response object, decoded the response content and turned it into a Pandas Data Frame.

  Complete notebook: final/1. jupyter-labs-spacex-data-collection-api.ipynb at main · ambalmori/final (github.com)

DATA COLLECTION

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:

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

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

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

Out[10]: 200

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

In [11]:
```
# Use json_normalize meethod to convert the json result into a dataframe
respjson = response.json()
data = pd.json_normalize(respjson)
```

Using the dataframe `data` print the first 5 rows

In [12]:
```
# Get the head of the dataframe
data.head()
```

Out[12]:

static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures	details	crew	ships	cap

DATA COLECTION

- With Web Scraping

  We collected Falcon 9 historical launch records from a Wikipedia page titled List of Falcon 9 and Falcon Heavy launches (https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches)

1. We performed an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.
2. We used BeautifulSoup and requested libraries and extract the Falcon 9 launch HTML table records.
3. We Parsed the table and converted it into a Pandas Data Frame.

Complete notebook: final/2. jupyter-labs-webscraping.ipynb at main · ambalmori/final · GitHub

DATA COLLECTION

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.

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

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

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

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

DATA WRANGLING

- After we obtained a Pandas Data Frame from the collected data:

  We converted these outcomes.

  1. Filtering the data using the BoosterVersion column to only keep the Falcon 9 launches.
  2. Replaced the missing data values in PayloadMass column using mean value of colum.

  Also, we performed some Exploratory Data Analysis to find patterns in the data.

  Complete notebook: final/3. labs-jupyter-spacex-data_wrangling_jupyterlite.jupyterlite.ipynb at main · ambalmori/final · GitHub

DATA WRANGLING

TASK 4: Create a landing outcome label from Outcome column

Using the `Outcome`, create a list where the element is zero if the corresponding row in `Outcome` is in the set `bad_outcome`; otherwise, it's one. Then assign it to the variable `landing_class`:

In [16]:

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
df['Class'] = df['Outcome'].apply(lambda x: 0 if x in bad_outcomes else 1)
df['Class'].value_counts()
```

Out[16]:

```
1    60
0    30
Name: Class, dtype: int64
```

This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

In [18]:

```
landing_class=df['Class']
df[['Class']].head(8)
```

Out[18]:

	Class
0	0
1	0
2	0
3	0

DATA VISUALIZATION EDA WITH VISUALIZATIONS

- We perform Exploratory Data Analysis and Feature Engineering using Pandas and Matplotlib:
  - Exploratory Data Analysis
  - Preparing Data Feature Engineering

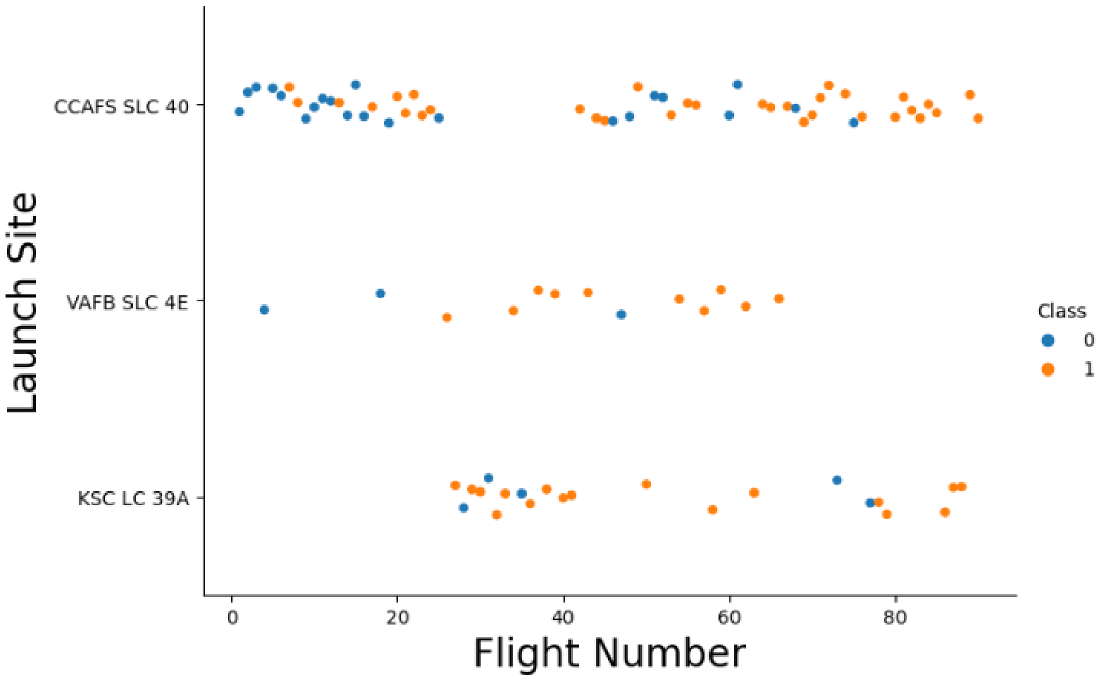
- We used scatter point chart to visualize the relationship between two variables:
  - Flight number x Launch site
  - Payload mass x Launch site
  - Flight number x Orbit type
  - Payload mass x Orbit type

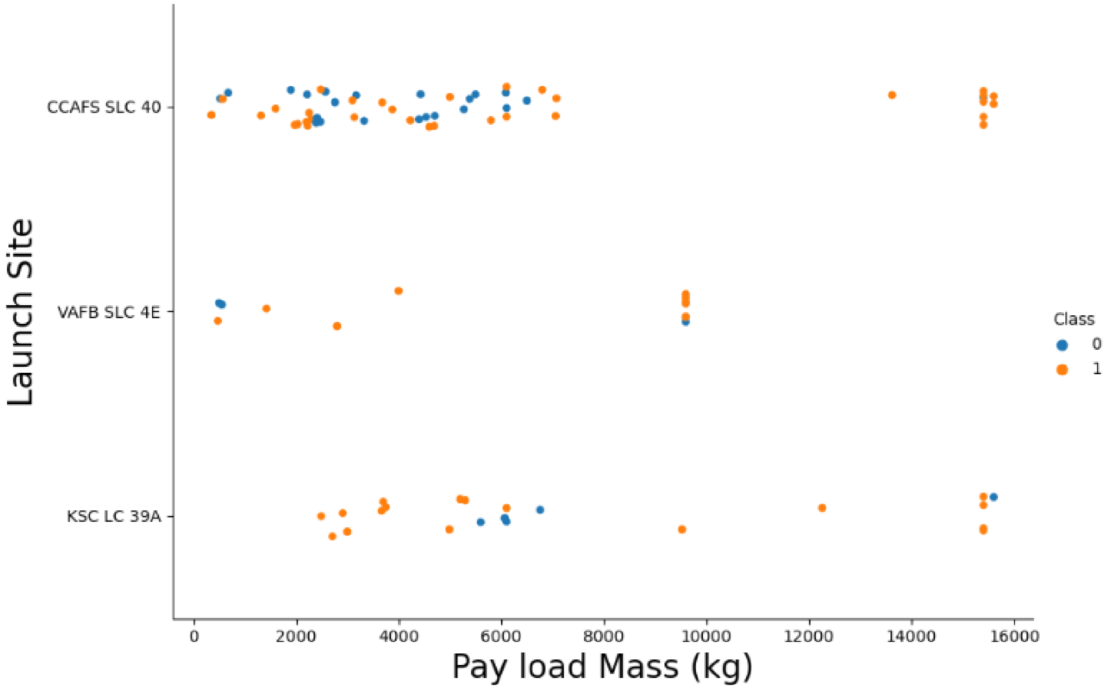
DATA VISUALIZATION EDA WITH VISUALIZATIONS

- We used bar chart to visualize the relationship between success rate of each orbit type.
- We used plot line chart for the launch success rate.

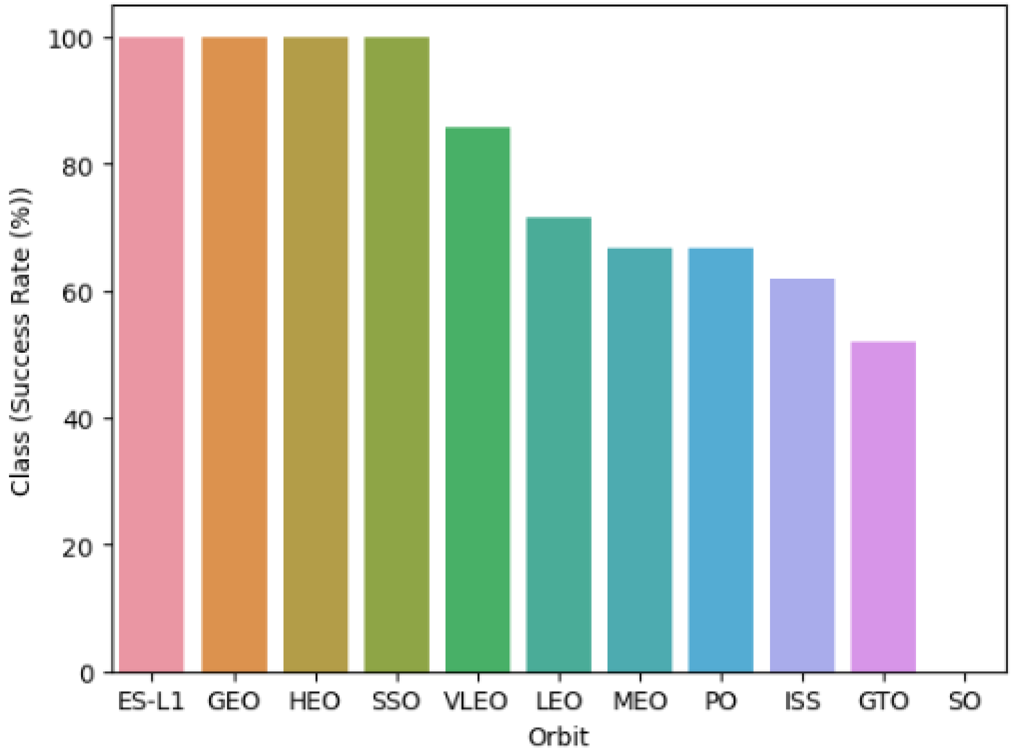
Complete notebook: final/5. jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb at main · ambalmori/final · GitHub

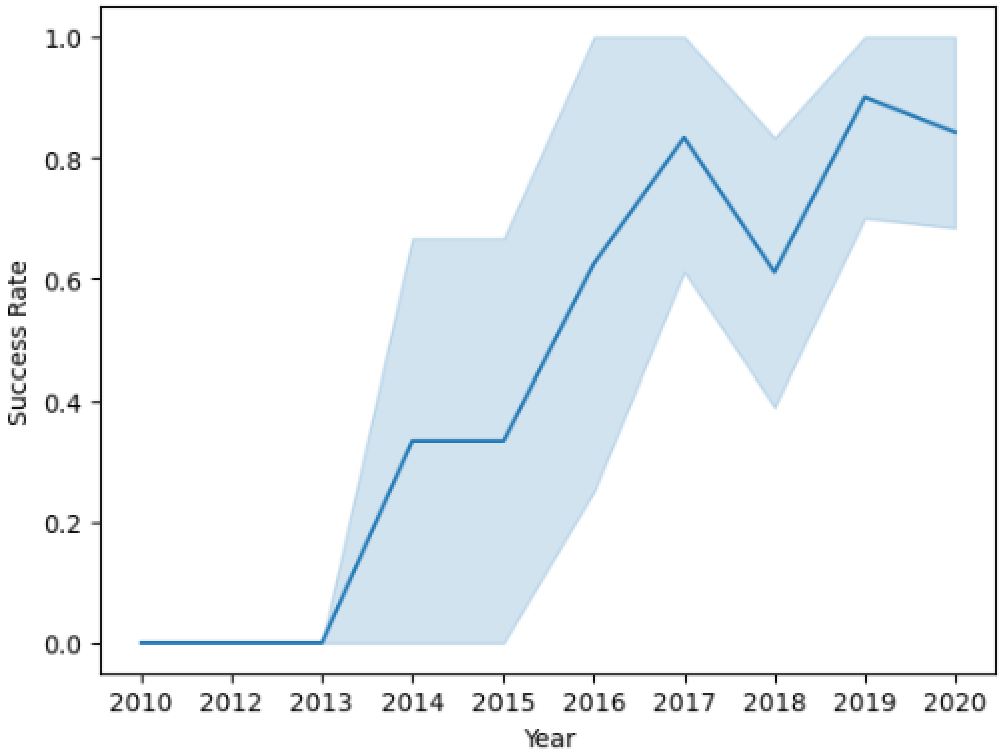
DATA VISUALIZATION EDA WITH VISUALIZATIONS



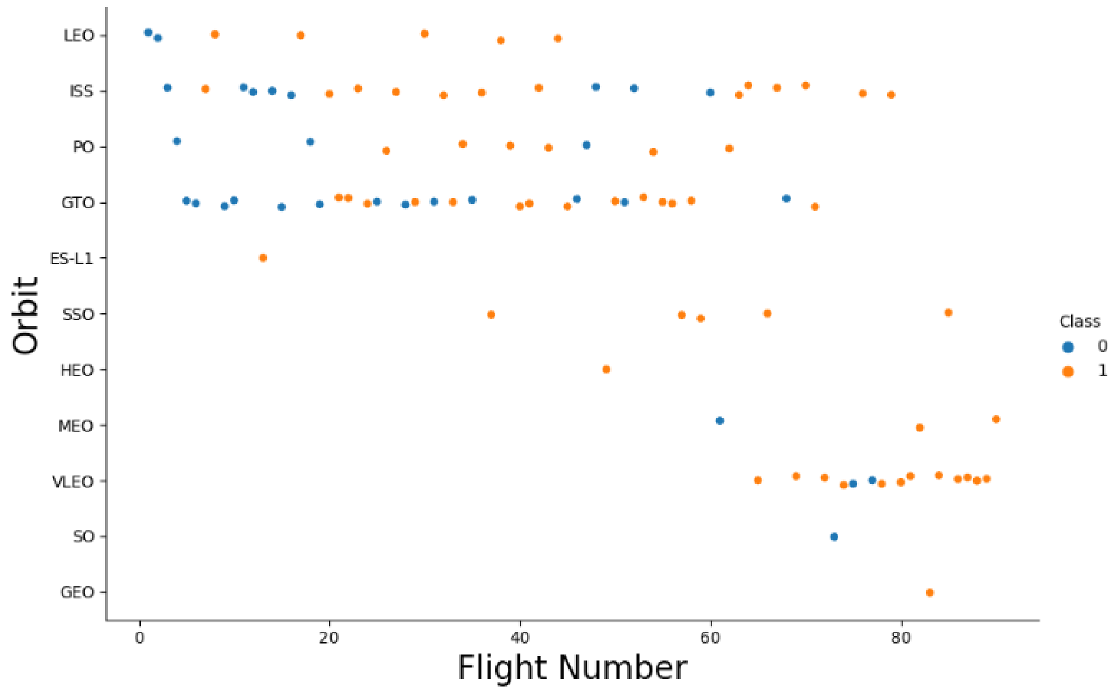


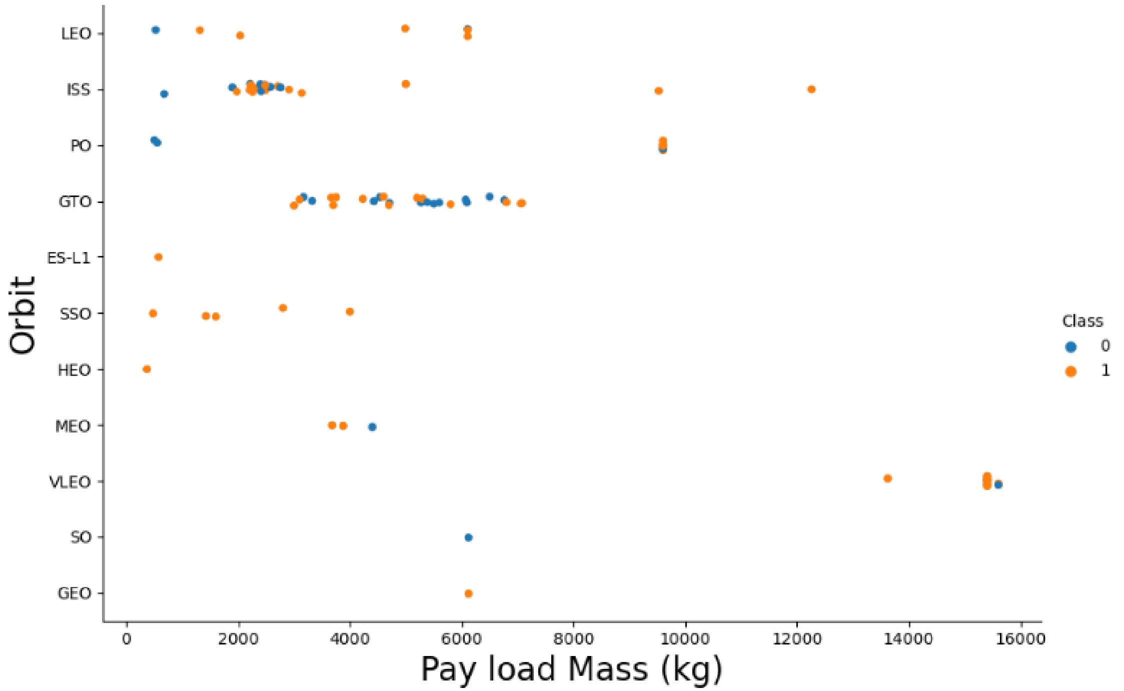
DATA VISUALIZATION EDA WITH VISUALIZATIONS





DATA VISUALIZATION EDA WITH VISUALIZATIONS





DATA VISUALIZATION EDA WITH SQL

- Display the names of the unique launch sites in the space misión.
- Display 5 récords where launch sites being with the string csa.
- Display the total payload mass carried bay booster launched by N.
- Display average payload mass carried by booster versión F9.
- List the date when the first successful landing outcome in ground pad was achieved.
- List the names of the booster which have success in drone ship and have payload mass greater than 4,000 but less than 6,000.
- List the total number of successful and failure missions outcome.
- List the names of the boosters versions which have carried the maximum payload mass.
- List the records which will display the months names, failure landing outcomes in drone ship, boosters versions, launch site for the month in 2015.

PREDICTIVE ANALISIS

- We obtained some preliminary insights about how each important variable would affect the success rate
- We select the features that will be used in success prediction.

In [23]:
```
### TASK  7: Create dummy variables to categorical columns
```

Use the function `get_dummies` and `features` dataframe to apply OneHotEncoder to the column `Orbits`, `LaunchSite`, `LandingPad`, and `Serial`. Assign the value to the variable `features_one_hot`, display the results using the method head. Your result dataframe must include all features including the encoded ones.

In [24]:
```
# HINT: Use get_dummies() function on the categorical columns
features_one_hot = pd.get_dummies(features, columns=['Orbit', 'LaunchSite', 'LandingPad', 'Serial'])
features_one_hot.head()
```

Out[24]:

	FlightNumber	PayloadMass	Flights	GridFins	Reused	Legs	Block	ReusedCount	Orbit_ES-L1	Orbit_GEO	...	Serial_B1048	Seri
0	1	6104.959412	1	False	False	False	1.0	0	0	0	...	0	
1	2	525.000000	1	False	False	False	1.0	0	0	0	...	0	
2	3	677.000000	1	False	False	False	1.0	0	0	0	...	0	
3	4	500.000000	1	False	False	False	1.0	0	0	0	...	0	
4	5	3170.000000	1	False	False	False	1.0	0	0	0	...	0	

5 rows × 80 columns

LAUNCH SITES ANALISIS INTERACTIVE MAP WITH FOLIUM

- We create folium maps to marked:
    - All launch sites on a map
    - Success/failed launches for each site on the map
    - Calculate the distances between a launch site to its proximities

- We create map objects:
    - Markers
    - Circles
    - Lines

- Complete notebook: final/6. lab_jupyter_launch_site_location.jupyterlite.ipynb at main · ambalmori/final · GitHub

LAUNCH SITES ANALISIS INTERACTIVE MAP WITH FOLIUM

In [5]:

```
## Task 1: Mark all launch sites on a map
```

First, let's try to add each site's location on a map using site's latitude and longitude coordinates

The following dataset with the name `spacex_launch_geo.csv` is an augmented dataset with latitude and longitude added for each site.

In [4]:

```
# Download and read the `spacex_launch_geo.csv`
from js import fetch
import io

URL = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/space
resp = await fetch(URL)
spacex_csv_file = io.BytesIO((await resp.arrayBuffer()).to_py())
spacex_df=pd.read_csv(spacex_csv_file)
```

Now, you can take a look at what are the coordinates for each site.

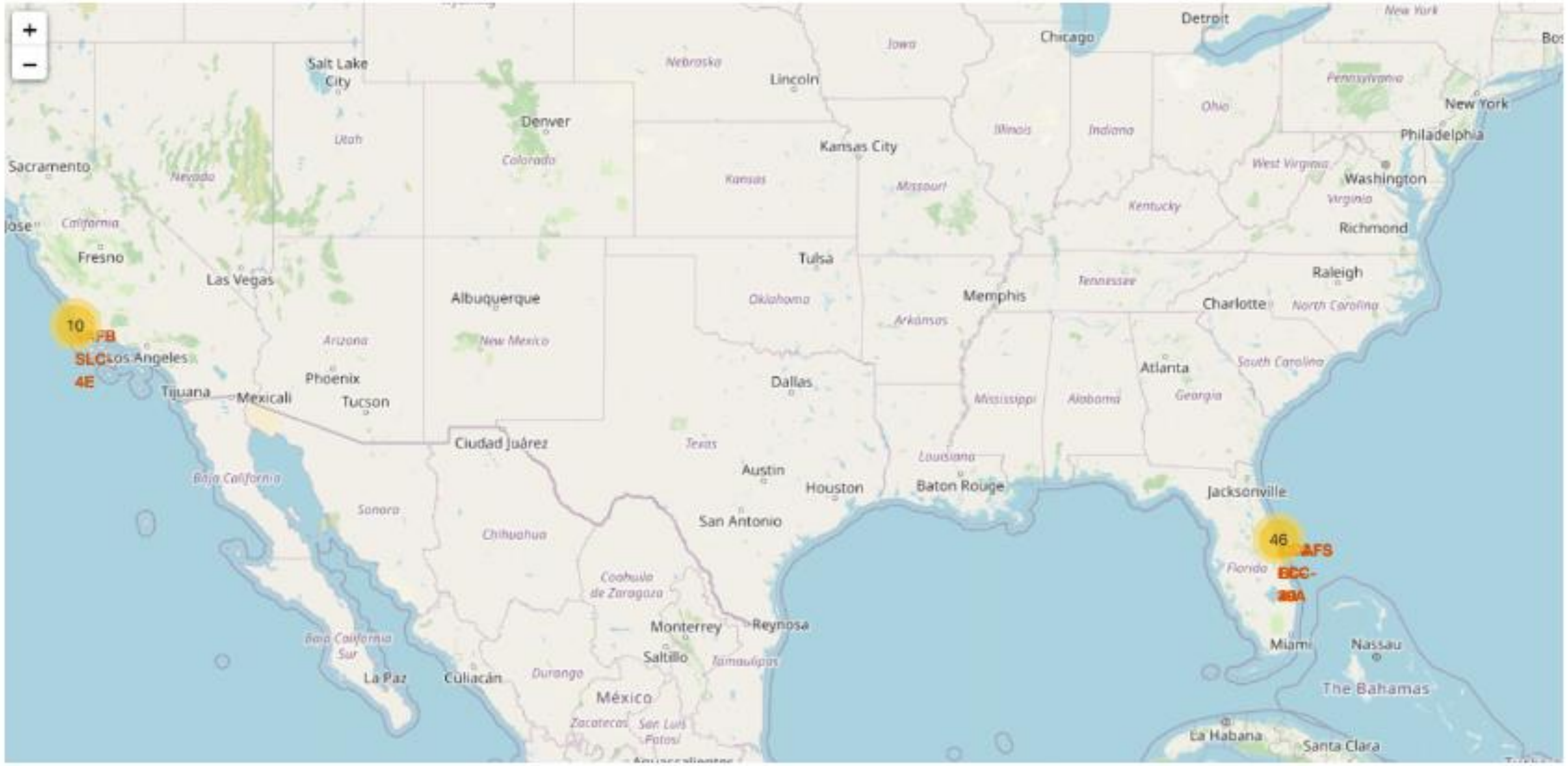
In [6]:

```
# Select relevant sub-columns: `Launch Site`, `Lat(Latitude)`, `Long(Longitude)`, `class`
spacex_df = spacex_df[['Launch Site', 'Lat', 'Long', 'class']]
launch_sites_df = spacex_df.groupby(['Launch Site'], as_index=False).first()
launch_sites_df = launch_sites_df[['Launch Site', 'Lat', 'Long']]
launch_sites_df
```

Out[6]:

	Launch Site	Lat	Long
0	CCAFS LC-40	28.562302	-80.577356
1	CCAFS SLC-40	28.563197	-80.576820

LAUNCH SITES ANALISIS INTERACTIVE MAP WITH FOLIUM



PREDICTIVE ANALISIS CLASSIFICATION

- We performed exploratory data analysis and determined training labels.
- We created a column of class to the dependent variable
- We standardized the data
- We split the information into training data and test data.
- We found the method perform best using test data between:
    - SVM
    - Classification trees
    - K nearest neighbors
    - Logistic Regression

PREDICTIVE ANALISIS CLASSIFICATION

- For each model:
  1. First we created an object for each of the algorithms then created a GridSearchCV object and assigned them a set of parameters.
  2. The GridSearchCV was created with cv = 10, then fit the training data into to find the best hyperparameter.
  3. After fitting the training set, we displayed the best parameters and the accuracy on the validation data.
  4. Finally we calculated on the test data and plot a confusion matrix using the test and predicted outcomes.

- Complete notebook: final/7. SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb at main · ambalmori/final · GitHub

PREDICTIVE ANALISIS CLASSIFICATION

TASK 9

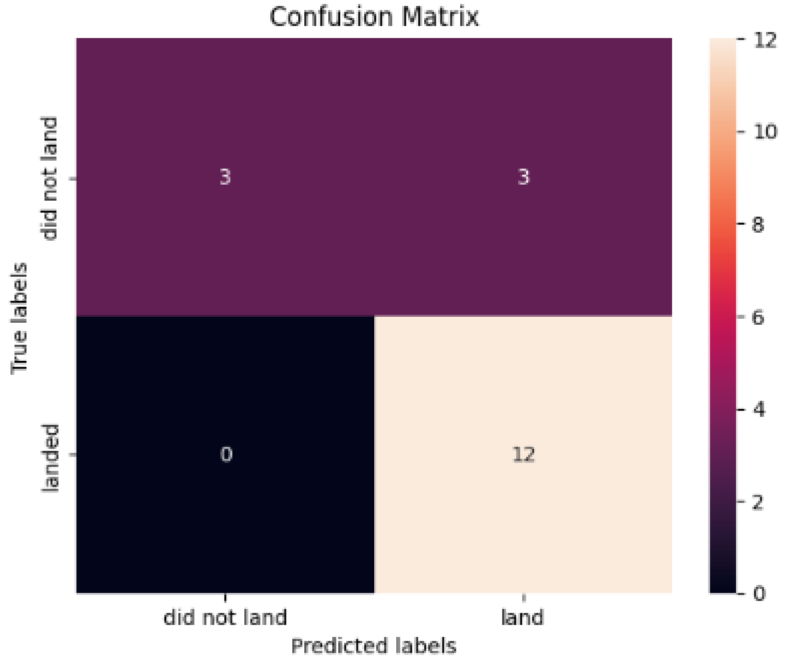
Calculate the accuracy of tree_cv on the test data using the method `score` :

In [30]:
```
print("Decision Tree accuracy on test set :",tree_cv.score(X_test, Y_test))
```

```
Decision Tree accuracy on test set : 0.8333333333333334
```

We can plot the confusion matrix

In [31]:
```
yhat = tree_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



CONCLUSION

- The use of data science tools, such as data collection, data wrangling, data analysis and data visualization, provide a competitive advantage with the increasing volume of data being used. With these methodologies it is possible to identify patterns in the data that without them it would be very difficult to observe, and it is possible to apply the machine learning model that provides greater precision for timely and adequate decision making.