Winning Space Race with Data Science

Enrico Ras

01 May 2023

Outline

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

Executive Summary

- **Summary of methodologies**
  - Data collection
  - Data Wrangling
  - EDA with SQL
  - EDA Data Visualisation
  - Interactive Dashboard
  - Interactive Map with Folium
  - Predictive Analysis
- **Summary of all results**
  - Exploratory Data Analysis
  - Predictive Data Analysis

Introduction

SpaceX has gained worldwide attention for a series of historic milestones. It is the only private company ever to return a spacecraft from low-earth orbit, which it first accomplished in December 2010. SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars whereas other providers cost upward of 165 million dollars each, much of the savings is because Space X can reuse the first stage.

Objective:
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 SpaceX for a rocket launch

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
- Perform data wrangling
- 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

I collected the SpaceX data by making use of the REST API. I received the data in a JSON format file, which I then converted to a dataframe. I made use of the json_normalize function to normalize the structured json data into a flat table.

Data Collection – SpaceX API

- I used the following code to collect the data and convert it to a data frame. The link to the full code can be access through the link below.
- data_collection_GitHub

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

In [7]:
```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

In [8]:
```
response = requests.get(spacex_url)
```

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

In [12]:
```
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

Using the dataframe `data` print the first 5 rows

In [13]:
```
# Get the head of the dataframe
data.head(5)
```

Data Collection - Scraping

I also performed webscraping after I collected the data. The purpose of the webscraping was to collect Falcon 9 historical launch records from a Wikipedia page. I extracted a Falcon 9 launch records HTML table from Wikipedia, then I parsed the table and converted it into a Pandas data frame.

Data Collection - Scraping

- I used the following code to perform webscraping, the rest of the code is available on my GitHub account. See link down below:
- [Webscraping_GitHub]()

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

In [7]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
# 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 [8]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.text, 'html')
```

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

In [9]:
```
# Use soup.title attribute
soup.title
```

Data Wrangling

After I collected the data, I did some data wrangling on the SpaceX dataset. I identified and calculated the missing values in each attribute, identified which columns are numerical and categorical, calculated the number of launches on each site, determined the number and occurrence of each orbit in the column Orbit, determined the number and occurrence of mission outcome of the orbits, I also created a landing outcome label from Outcome column and lastly I determined the success rate.

Data Wrangling – SpaceX API

- I used the following code to perform data wrangling techniques on the dataset, the rest of the code can be accessed on my GitHub account.
- [Data_Wrangling_GitHub]()

Identify and calculate the percentage of the missing values in each attribute

In [4]:
```
df.isnull().sum()/df.count()*100
```

Identify which columns are numerical and categorical:

In [5]:
```
df.dtypes
```

EDA with Data Visualization

- After collecting the data, I performed Exploratory Data Analysis to identify which variable(s) would affect the launch outcome the most.
- I compared different variables against each other by making use of data visualization to visualize the relationship between the two variables
- I compared:

Flightnumber vs PayLoadMass, Flightnumber vs LaunchSite, PayLoadMass vs LaunchSite, Orbit vs Class, FlightNumber vs Orbit Type and PayLoadMass vs Orbit type.

EDA with Data Visualization

- I made use of Scatterplots to visualize the relationships between the numeric variables
- I used one bar chart to represent the categorical data, the success rate of each orbit
- I also visualized the launch success yearly trend by making use of a line plot

EDA with Data Visualization

Click the link below to view the code of my plots and EDA

EDA with Data Visualisation – GitHub

EDA with SQL

- We must connect to the Db2 instance to perform EDA with SQL. We must perform the following queries
- Display names of the unique launch sites
- 5 records of launch sites starting with CCA
- Display the total pay load mass and average pay load mass
- Display date of first successful launch
- List boosters which have success with a pay load mass between 4000-6000

GitHub code EDA_SQL

Build an Interactive Map with Folium

The launch success rate may depend on many factors such as payload mass, orbit type, and so on. It may also depend on the location and proximities of a launch site, i.e., the initial position of rocket trajectories. Finding an optimal location for building a launch site certainly involves many factors and hopefully we could discover some of the factors by analysing the existing launch site locations

objectives

- Mark all launch sites on a map
- Mark the success/failed launches for each site on the map
- Calculate the distances between a launch site to its proximities

GitHub code [Folium]

Build a Dashboard with Plotly Dash

- With interactive visual analytics, users could find visual patterns faster and more effectively.
- Instead of presenting your findings in static graphs, interactive data visualization, or dashboarding, can always tell a more appealing story
- This dashboard application contains input components such as a dropdown list and a range
- slider to interact with a pie chart and a scatter point chart.

GitHub code [dashboard]

Predictive Analysis (Classification)

- Space X 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 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.
- Perform exploratory Data Analysis and determine Training Labels
- create a column for the class
- Standardize the data
- Split into training data and test data
- -Find best Hyperparameter for SVM, Classification Trees and Logistic Regression
- Find the method performs best using test data

- GitHub code predictive analysis

Results

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

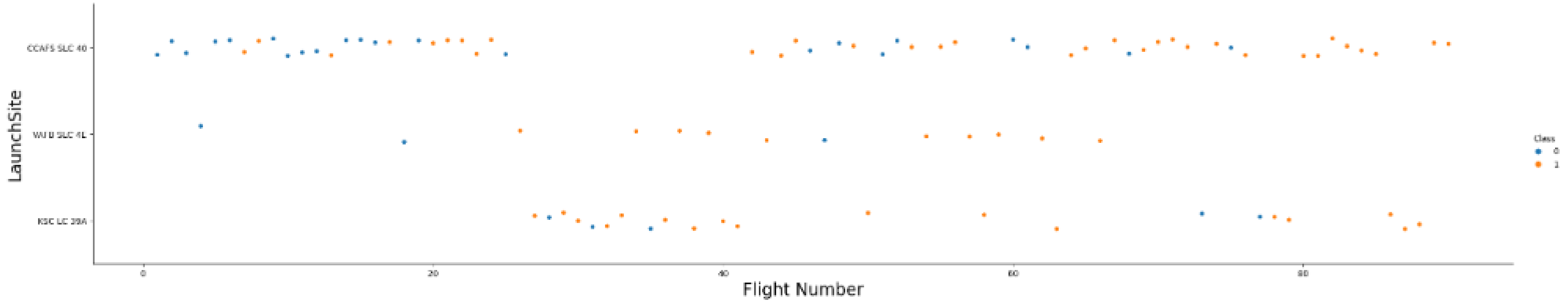
Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

In [5]:

```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class valu
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("LaunchSite",fontsize=20)
plt.show()
```



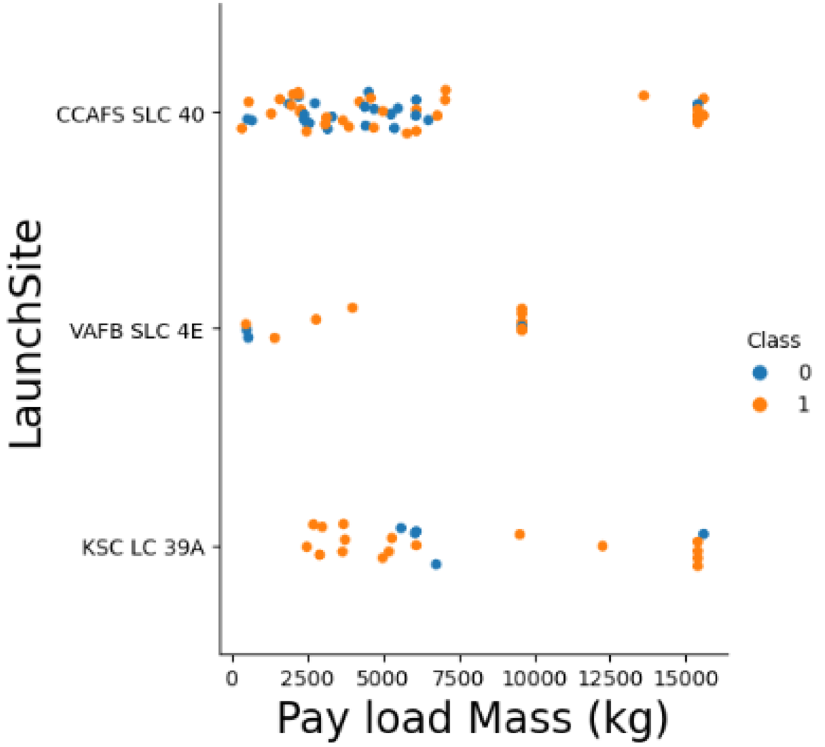
- If you observe the Flight Number vs Launch Site scatter plot, you will find for the KSC LC 39A there are no rockets launched for Flight Number below 20
- Also observe that for the VAFB SLC 4E there are no rockets launched for Flight Number above 80

Payload vs. Launch Site

- We observe the Payload vs. Launch Site scatter plot and notice that for the VAFB-SLC launch site there are no rockets launched for heavy payload mass(greater than 10000)

In [7]:

```
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be the class
sns.catplot(x="PayloadMass", y="LaunchSite", hue="Class", data=df)
plt.ylabel("LaunchSite",fontsize=20)
plt.xlabel("Pay load Mass (kg)",fontsize=20)
plt.show()
```

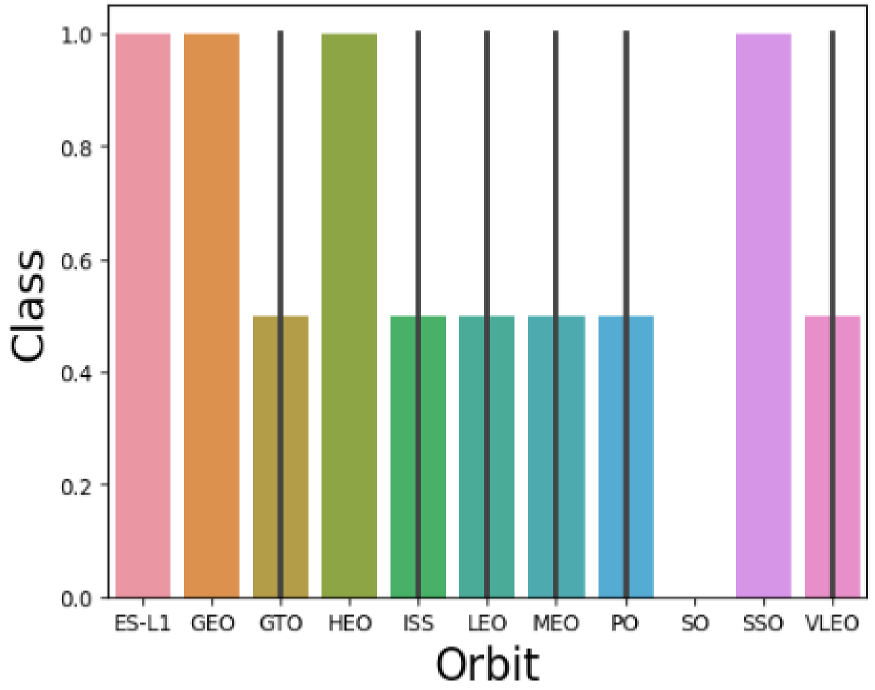


Success Rate vs. Orbit Type

We observe that ES-L1, GEO, HEO and SSO have the highest success rate

In [8]:
```
# HINT use groupby method on Orbit column and get the mean of Class column
t = df.groupby(['Orbit', 'Class'])['Class'].agg(['mean']).reset_index()
sns.barplot(y="Class", x="Orbit", data=t)

plt.xlabel("Orbit",fontsize=20)
plt.ylabel("Class",fontsize=20)
plt.show()
```

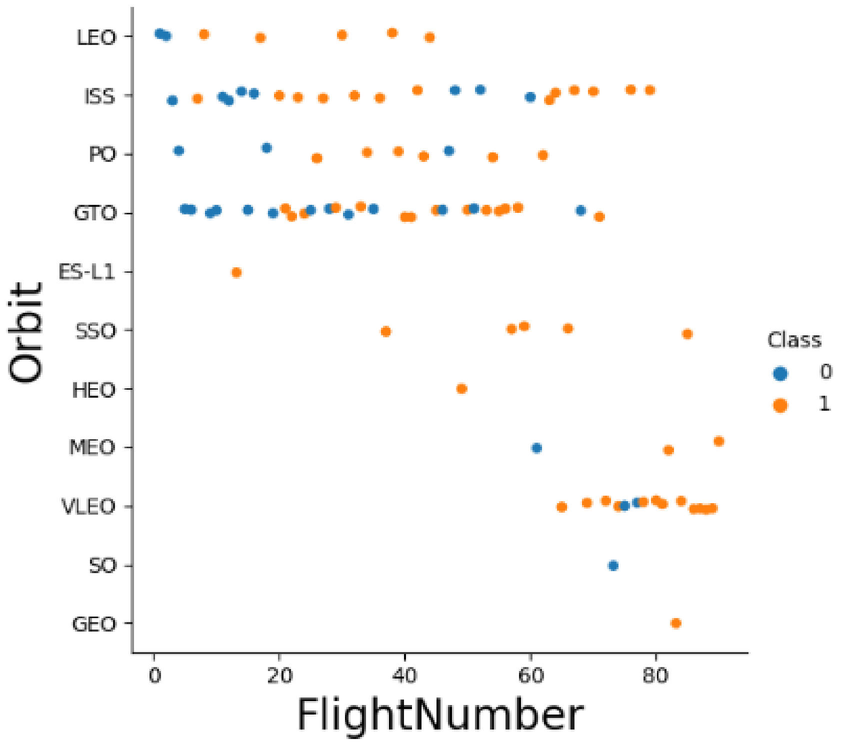


Flight Number vs. Orbit Type

- We observe that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit

In [9]:

```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value
sns.catplot(x="FlightNumber", y="Orbit", hue="Class", data=df)
plt.ylabel("Orbit",fontsize=20)
plt.xlabel("FlightNumber",fontsize=20)
plt.show()
```

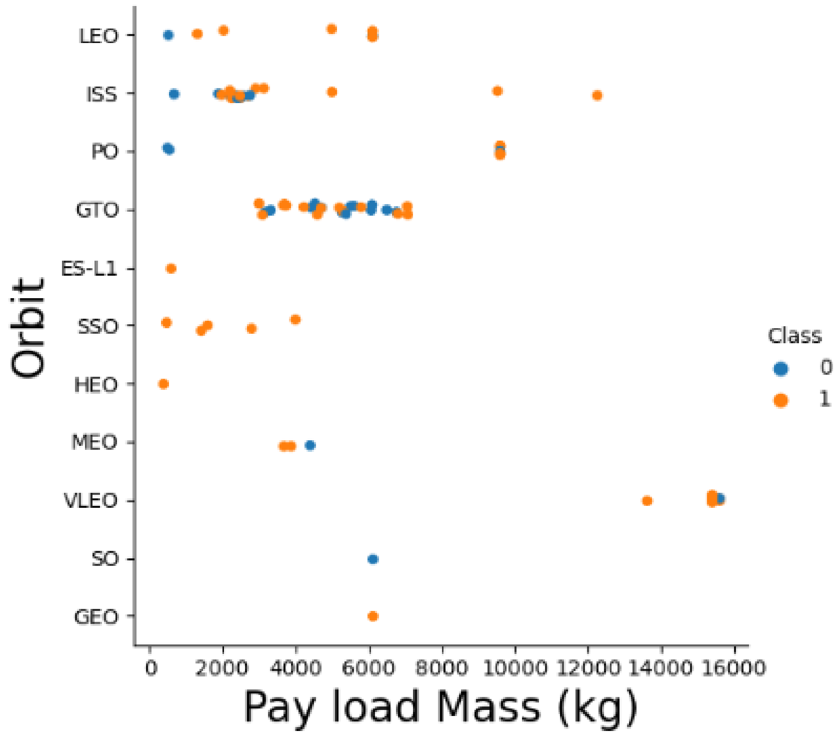


Payload vs. Orbit Type

- With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS.
- However, for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.

In [13]:

```
# Plot a scatter point chart with x axis to be Payload and y axis to be the Orbit, and hue to be the class value
sns.catplot(x="PayloadMass", y="Orbit", hue="Class", data=df)
plt.ylabel("Orbit",fontsize=20)
plt.xlabel("Pay load Mass (kg)",fontsize=20)
plt.show()
```

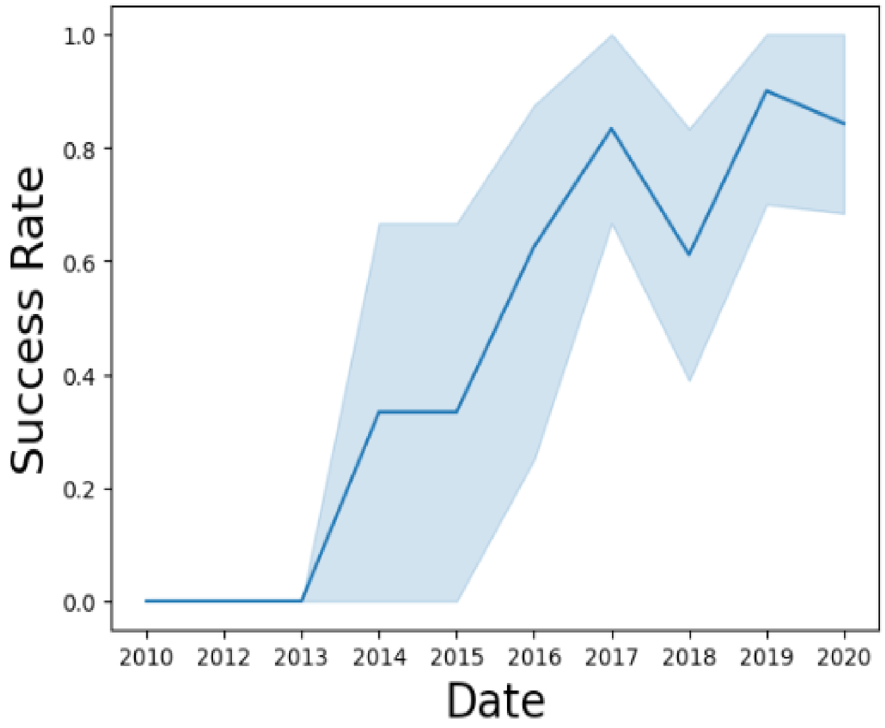


Launch Success Yearly Trend

We observe that after 2013, the yearly launch success trend increases each year

In [16]:

```
sns.lineplot(data=df1, x="Date", y="Class")
plt.xlabel("Date",fontsize=20)
plt.ylabel("Success Rate",fontsize=20)
plt.show()
```



All Launch Site Names

```
In [6]: %sql select Unique(LAUNCH_SITE) from SPACEXTBL;

 * ibm_db_sa://ktf76410:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

Out[6]:

launch_site
CCAFS LC-40
CCAFS SLC-40
CCAFSSLC-40
KSC LC-39A
VAFB SLC-4E

This is the list of the unique launch sites in the data set

Launch Site Names Begin with 'CCA'

```
In [7]: %sql SELECT LAUNCH_SITE from SPACEXTBL where (LAUNCH_SITE) LIKE 'CCA%' LIMIT 5;

 * ibm_db_sa://ktf76410:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

Out[7]:

launch_site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

List of launch sites that begin with ‘CCA’

Total Payload Mass

```
In [9]: %sql select sum(PAYLOAD_MASS__KG_) as payloadmass from SPACEXTBL;

 * ibm_db_sa://ktf76410:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.

Out[9]:
```

payloadmass
619967

The total payload mass is 619967 kg.

Average Payload Mass by F9 v1.1

```
In [10]: %sql select avg(PAYLOAD_MASS__KG_) as payloadmass from SPACEXTBL;

 * ibm_db_sa://ktf76410:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.

Out[10]:
```

payloadmass
6138

- The average payload mass carried by booster version F9 v1.1 is 6138kg

First Successful Ground Landing Date

Hint:Use min function

```
In [11]: %sql select min(DATE) from SPACEXTBL;
 * ibm_db_sa://ktf76410:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
Out[11]:
```

1
2010-06-04

The first successful landing took place on 2010-06-04

Successful Drone Ship Landing with Payload between 4000 and 6000

```
In [12]: %sql select BOOSTER_VERSION from SPACEXTBL where LANDING__OUTCOME='Success (drone ship)' and PAYLOAD_MASS__KG_ BETWEEN 4000 and 6

 * ibm_db_sa://ktf76410:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.

Out[12]:
```

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

- List of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

Total Number of Successful and Failure Mission Outcomes

In [16]:
```
%sql select count(MISSION_OUTCOME) as missionoutcomes from SPACEXTBL GROUP BY MISSION_OUTCOME;
```

```
 * ibm_db_sa://ktf76410:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

Out[16]:

missionoutcomes
1
99
1

- Total number of successful and failure mission outcomes

Boosters Carried Maximum Payload

```
In [19]: %sql select BOOSTER_VERSION as boosterversion from SPACEXTBL where PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_) from SPACEXTE
 * ibm_db_sa://ktf76410:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
Out[19]:
```

boosterversion
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

- List of the boosters which have carried the maximum payload mass

2015 Launch Records

```
In [43]: %sql SELECT MONTH(DATE),MISSION_OUTCOME,BOOSTER_VERSION,LAUNCH_SITE FROM SPACEXTBL where EXTRACT(YEAR FROM DATE)='2015';
```

```
 * ibm_db_sa://ktf76410:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

Out[43]:

1	mission_outcome	booster_version	launch_site
1	Success	F9 v1.1 B1012	CCAFS LC-40
2	Success	F9 v1.1 B1013	CCAFS LC-40
3	Success	F9 v1.1 B1014	CCAFS LC-40
4	Success	F9 v1.1 B1015	CCAFS LC-40
4	Success	F9 v1.1 B1016	CCAFS LC-40
6	Failure (in flight)	F9 v1.1 B1018	CCAFS LC-40
12	Success	F9 FT B1019	CCAFS LC-40

- List of failed landing outcomes in drone ship, their booster versions, and launch site for 2015

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

- Rank of the landing outcomes (such as Failure (drone ship) or Success

In [44]:
```
%sql SELECT LANDING__OUTCOME FROM SPACEXTBL WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' ORDER BY DATE DESC;
```

```
 * ibm_db_sa://ktf76410:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

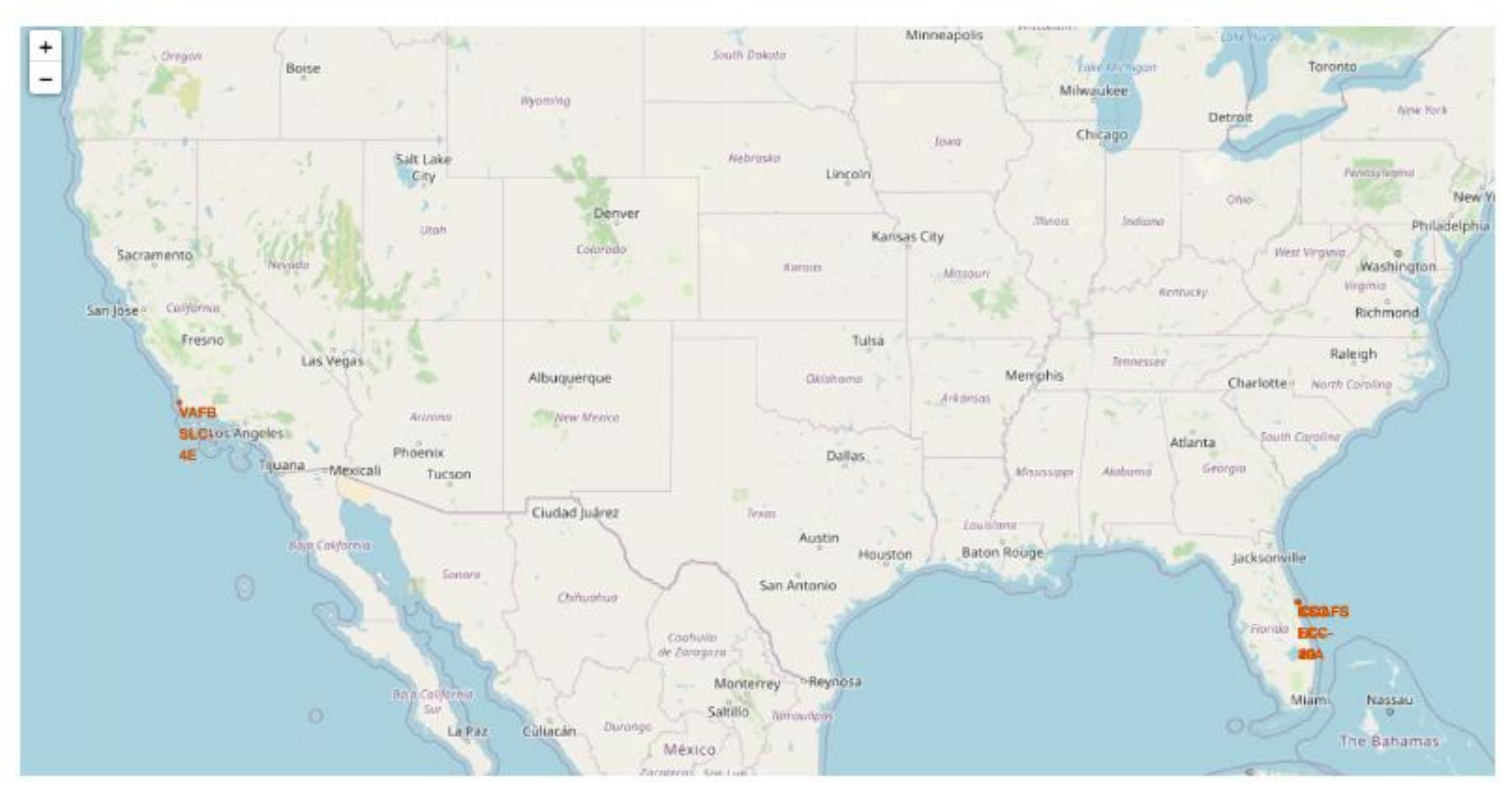
Out[44]:

landing__outcome
No attempt
Success (ground pad)
Success (drone ship)
Success (drone ship)
Success (ground pad)
Failure (drone ship)
Success (drone ship)
Success (drone ship)
Success (drone ship)
Failure (drone ship)
Failure (drone ship)
Success (ground pad)
Precluded (drone ship)
Failure (drone ship)
No attempt
Controlled (ocean)
Failure (drone ship)
Uncontrolled (ocean)
No attempt
No attempt
Controlled (ocean)
Controlled (ocean)
No attempt
No attempt
Uncontrolled (ocean)
No attempt
No attempt
No attempt
Failure (parachute)
Failure (parachute)

Section 3

Launch Sites Proximities Analysis

Marked launched sites



Success/Failed launches for each site

Distance between launch site to its proximity

Section 4

Build a Dashboard with Plotly Dash

Launch success in pie chart

PayLoad range slider

Section 5

Predictive Analysis (Classification)

Classification Accuracy

After fitting KNN Classifier, Logistic regression, Decision tree and Support Vector Machine models to identify which one performs the best

We conclude that the KNN model performed the best

```
In [29]: parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                       'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
                       'p': [1,2]}

         KNN = KNeighborsClassifier()

In [30]: # Instantiate the GridSearchCV object: knn_cv
         knn_cv = GridSearchCV(KNN, parameters, cv=10)

         # Fit it to the data
         knn_cv.fit(X_train, Y_train)

Out[30]: ▸ GridSearchCV
         ▸ estimator: KNeighborsClassifier
             ▸ KNeighborsClassifier

In [31]: print("tuned hpyerparameters :(best parameters) ",knn_cv.best_params_)
         print("accuracy :",knn_cv.best_score_)

         tuned hpyerparameters :(best parameters)  {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
         accuracy : 0.8482142857142858
```

Confusion Matrix

- After analysing all the models, the KNN was the best model with an accuracy of 84% and a score of 83%

In [29]:
```
parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
              'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
              'p': [1,2]}

KNN = KNeighborsClassifier()
```

In [30]:
```
# Instantiate the GridSearchCV object: knn_cv
knn_cv = GridSearchCV(KNN, parameters, cv=10)

# Fit it to the data
knn_cv.fit(X_train, Y_train)
```

Out[30]:
```
▸ GridSearchCV
▸ estimator: KNeighborsClassifier
  ▸ KNeighborsClassifier
```

In [31]:
```
print("tuned hpyerparameters :(best parameters) ",knn_cv.best_params_)
print("accuracy :",knn_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
accuracy : 0.8482142857142858
```

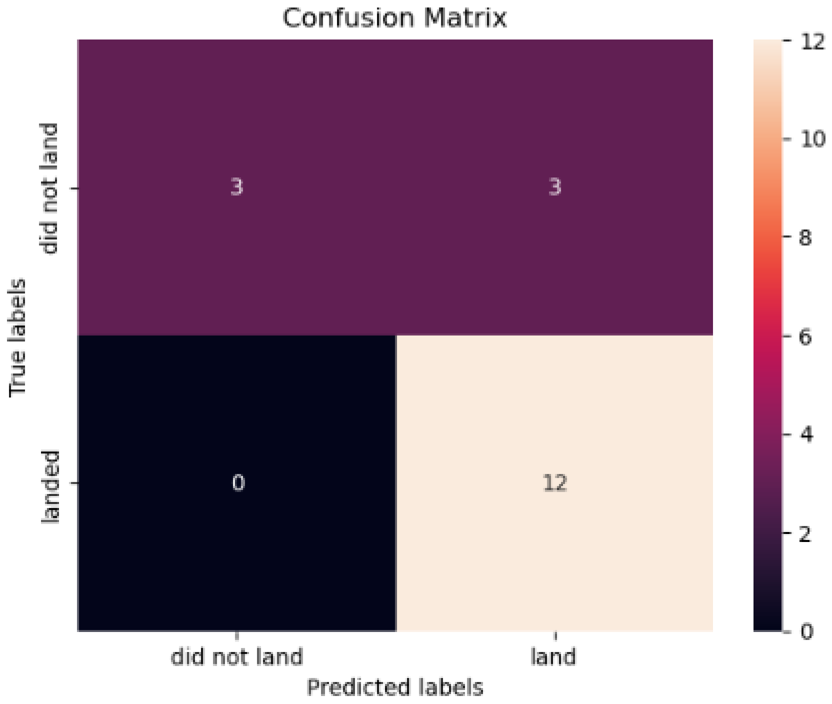
Confusion Matrix

In [32]:
```
knn_cv.score(X_test, Y_test)
```

Out[32]:
```
0.8333333333333334
```

We can plot the confusion matrix

In [33]:
```
yhat = knn_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



Conclusions

- Using Existing Data and Analysing the data ,SpaceX and other rocket companies can be able to see the best way to reduce the cost of launches,and evolve before there tradition costly launches lead to their absoluteness and losing their client

Appendix

- EDA_SQL Code notebook
- Machine Learning Code notebook
- Dashboard Code notebook

Thank you!