IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Alex Huebner

14.10.2022

Outline

Executive Summary

The idea of this work was to analyze and to find insights in the data of the SPACEX Falcon 9

Summary of methodologies:

Following instruments and ideas were used to collect and analyze data. Machine learning techniques were used to predict the outcome

Summary of all results:

The presentation will show the results and outcomes in different explanation ways: Visualizations of the outcome, predictive models by using machine learning techniques and different advanced algorithms and interactive dashboards

Introduction

Project background and context

- *Privat sector recently has been invested a lot of money and contributed to the Space travel industry. However, the space business is still capital intensive, technological difficult but already exciting data give an opportunity for data scientist make an impact to the industry*

Problems you want to find answers

- *To find answers if the first stage of SpaceX Falcon will land successfully*
- *Correlations between launch sites and success rates*
- *To find the methods which are contributing to the landing outcomes*

Section 1

Methodology

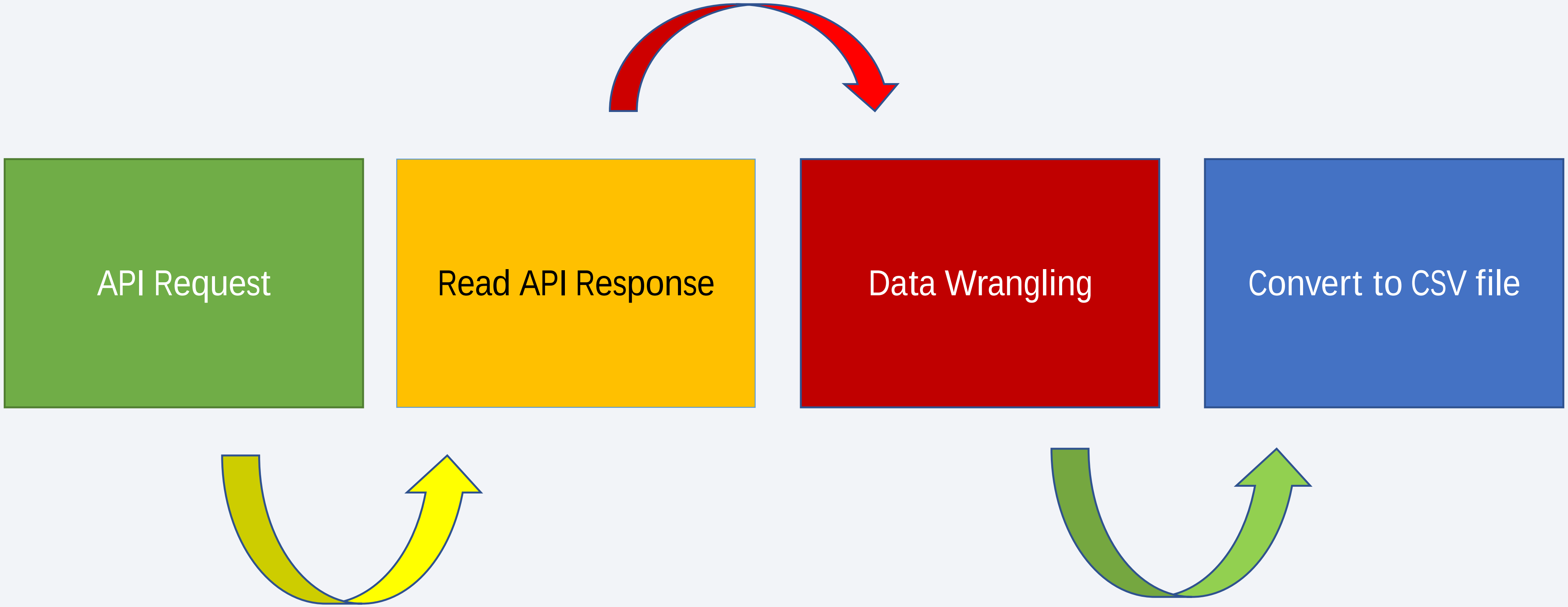
Methodology

Executive Summary

- Data collection methodology:
  - SpaceX API
  - Web scrap (Wikipedia)
- Perform data wrangling
  - Using the supervised models (machine learning techniques) the outcomes was determined as 0-unsuccessful, 1 - successful
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Find best Hyperparameter for SVM, Classification Trees and Logistic Regression

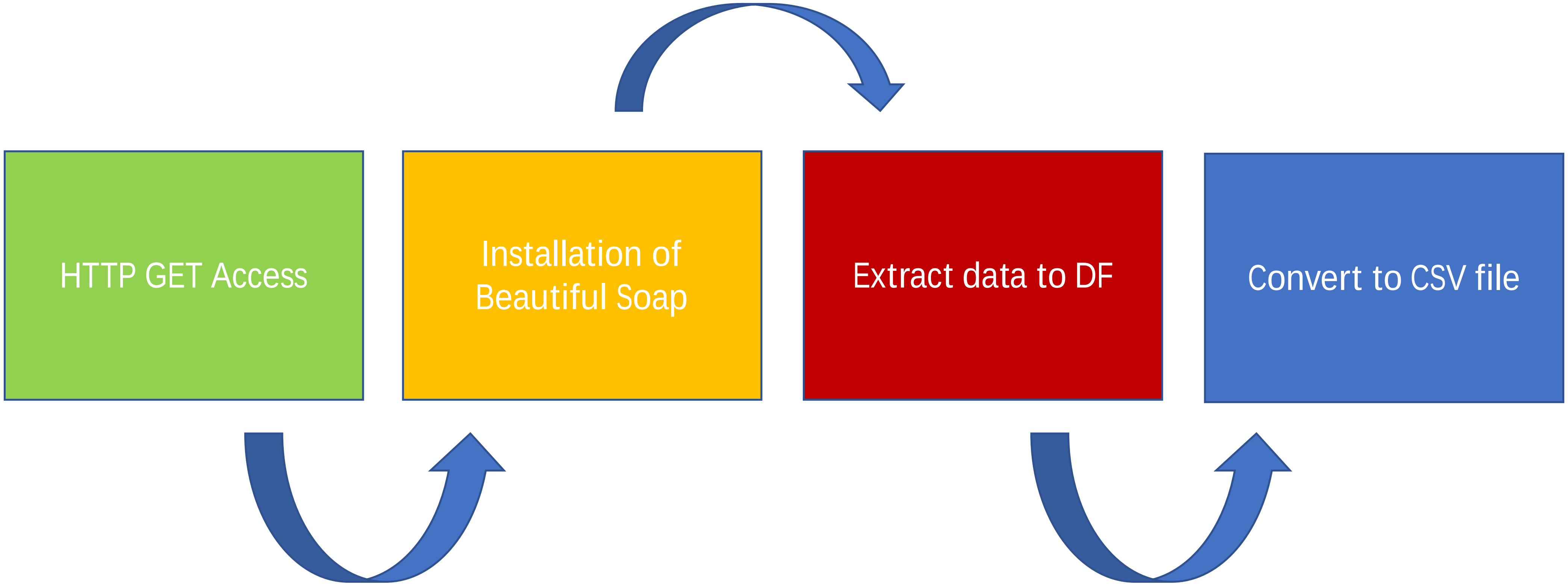
Data Collection

- Data sets for the analysis were collecting through different sources like Spacex API and Web scrapping. See below a SPACEX API approach and Web scraping from Wiki



Data Collection

Web scraping method



GitHub Link (code)

Data Collection – SpaceX API

1. Getting request from API

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

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

2. Converting request to a .json file

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

3. Work with a custom function to clean a data

- *getBoosterVersion*
- *getLaunchSite(data)*
- *getPlayloadData(data)*
- *getCoreData(data)*

4. Assign global variable lists to dictionary to get a relevant data

```
In [14]: #Global variables
         BoosterVersion = []
         PayloadMass = []
         Orbit = []
         LaunchSite = []
         Outcome = []
         Flights = []
         GridFins = []
         Reused = []
         Legs = []
         LandingPad = []
         Block = []
         ReusedCount = []
         Serial = []
         Longitude = []
         Latitude = []
```

```
In [21]: launch_dict = {'FlightNumber': list(data['flight_number']),
         'Date': list(data['date']),
         'BoosterVersion':BoosterVersion,
         'PayloadMass':PayloadMass,
         'Orbit':Orbit,
         'LaunchSite':LaunchSite,
         'Outcome':Outcome,
         'Flights':Flights,
         'GridFins':GridFins,
         'Reused':Reused,
         'Legs':Legs,
         'LandingPad':LandingPad,
         'Block':Block,
         'ReusedCount':ReusedCount,
         'Serial':Serial,
         'Longitude': Longitude,
         'Latitude': Latitude}
```

5. Filter a dataframe and convert to CSV file

```
In [22]: # Create a data from launch_dict
         df_launch = pd.DataFrame(launch_dict)
```

```
In [25]: # Hint data['BoosterVersion']!='Falcon 1'
         data_falcon9 = df_launch[df_launch['BoosterVersion']!= 'Falcon 1']
```

Now that we have removed some values we should reset the FlgihtNumber column

```
In [26]: data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
         data_falcon9
```

[GitHub Link (code)]

Data Collection - Scraping

1. Getting request from HTML

```
In [5]: # use requests.get() method with the provided static_url
        # assign the response to a object
        html_data = requests.get(static_url).text
```

2. Creating BeautifulSoup function

```
In [6]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
        soup = BeautifulSoup(html_data,"html.parser")
```

3. Looking for all tables

```
In [8]: # Use the find_all function in the BeautifulSoup obje
        # Assign the result to a list called `html_tables`
        html_tables = soup.find_all ('table')
```

4. Getting column names

```
In [ ]: launch_dict= dict.fromkeys(column_names)

        # Remove an irrelvant column
        del launch_dict['Date and time ( )']

        # Let's initial the launch_dict with each
        launch_dict['Flight No.'] = []
        launch_dict['Launch site'] = []
        launch_dict['Payload'] = []
        launch_dict['Payload mass'] = []
        launch_dict['Orbit'] = []
        launch_dict['Customer'] = []
        launch_dict['Launch outcome'] = []
        # Added some new columns
        launch_dict['Version Booster']=[]
        launch_dict['Booster landing']=[]
        launch_dict['Date']=[]
        launch_dict['Time']=[]
```

5. Creation of dictionary

```
In [ ]: extracted_row = 0
        #Extract each table
        for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheader
           # get table row
            for rows in table.find_all("tr"):
                #check to see if first table heading is as number corresponding to launch a
                if rows.th:
                    if rows.th.string:
                        flight_number=rows.th.string.strip()
                        flag=flight_number.isdigit()
                else:
                    flag=False
                #get table element
                row=rows.find_all('td')
```

6. Converting launch to dataframe

```
In [ ]: df=pd.DataFrame(launch_dict)
```

We can now export it to a **CSV** for the next section, but to mal

Following labs will be using a provided dataset to make each l

```
df.to_csv('spacex_web_scraped.csv', index=False)
```

GitHub Link (code)

Data Wrangling

Process of Data Wrangling

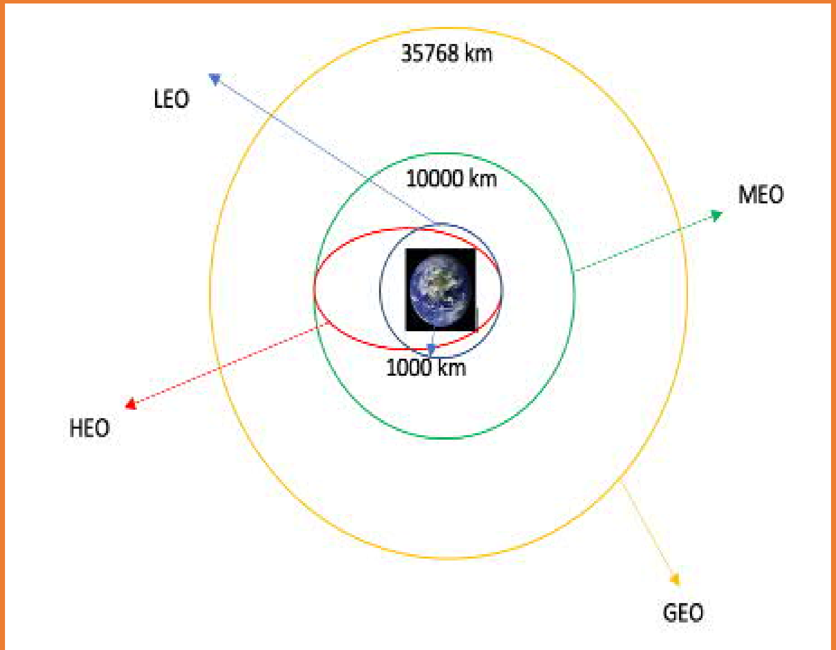
Exploratory Data Analysis(EDA) using to find some patterns in the data and determine the label for training supervised models

Do calculation of launches at each site

To calculation and occurence of mission outcome per orbit type

Do calculation and occurrence of each orbit

Create a landing outcome label from Outcome column



[GitHub Link (code)]

EDA with Data Visualization

For the Exploratory data analysis, following charts were used to gain more insights of the dataset and get a better outcome:

1. Scatter Graphs

the relationship between Flight Number and Launch Site

the relationship between Payload and Launch Site

the relationship between FlightNumber and Orbit type

the relationship between Payload and Orbit type

2. Bar Graph

the relationship between success rate of each orbit type

Success Rate of tach trbit type

3. Line Graph

the launch success yearly trend

[GitHub Link (code)]

EDA with SQL

By analyzing the SpaceX data set following task and questions were used to get better understanding the data

- Display the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string 'CCA'
- Display the total payload mass carried by boosters launched by NASA (CRS)
- Display average payload mass carried by booster version F9 v1.1
- List the date when the first successful landing outcome in ground pad was acheived.
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- List the total number of successful and failure mission outcomes
- List the names of the booster_versions which have carried the maximum payload mass. Use a subquery:
- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
- 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

GitHub Link (code)

Build an Interactive Map with Folium

Interactive Map with Folium is helpful if you need to analyse geospatial data to perform and understand the impact of location close to the rocket launch

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

- Dataframe launch_outcomes to classes 0 and 1 with Green and Red markers on the map in a MarkerCluster was added
- Calculated distances between a launch site to its proximities or to various landmarks to find several trends and patterns.

The Interactive Map with Folium helped the project to answer following questions:

- Are launch sites in close proximity to railways? YES
- 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

GitHub Link (code)

Build a Dashboard with Plotly Dash

PIE CHART – showing the total success for all sites/by certain launch site

- Percentage of success in relation to launch site

SCATTER GRAPH – showing a correlation between Payload and success for all sites/by certain launch site

- It shows the relationship between success rate and booster version category
- It is a good method to show a non-linear pattern
- The range of data flow, i.e. maximum and minimum value would be determined

Dashboard was created to solve following tasks:

Which site has the largest successful launches ?	Result: KSC LC-39A with 10
Which site has the highest launch success rate?	Result: KSC LC-39A with 76.9% success
Which payload range has the highest launch success rate?	Result: 2000-5000 KG

GitHub Link (code)

Predictive Analysis (Classification)

Create a NumPy array from the column CLASS in DATA

Standardize the data in

Train_Test_Split data into training and test data sets

Create a logistic regression

Calculate the accuracy on the test data

Create a support vector machine

Calculate the accuracy on the test data

Create a decision tree

Calculate the accuracy of decision tree algorithm

Create a k nearest neighbors

Find the algorithm performs best

```
In [5]: Y = data['Class'].to_numpy()
```

```
In [6]: # students get this
        transform = preprocessing.StandardScaler()
        X= preprocessing.StandardScaler().fit(X).transform(X)

In [7]: X[0:5]

Out[7]: array([[-1.71291154e+00, -1.94814463e-16, -6.53912840e-01,
```

```
In [8]: # Split data for training and testing data sets
        from sklearn.model_selection import train_test_split
        X_train, X_test, Y_train, Y_test = train_test_split( X, Y, test_size=0.2, random_state=2)
        print ('Train set:', X_train.shape,  Y_train.shape)
        print ('Test set:', X_test.shape,  Y_test.shape)

        Train set: (72, 83) (72,)
        Test set: (18, 83) (18,)
```

we can see we only have 18 test samples.

```
In [9]: Y_test.shape

Out[9]: (18,)
```

Out[32]:

	Algo Type	Accuracy Score	Test Data Accuracy Score
2	Decision Tree	0.876786	0.833333
3	KNN	0.848214	0.833333
1	SVM	0.848214	0.833333
0	Logistic Regression	0.846429	0.833333

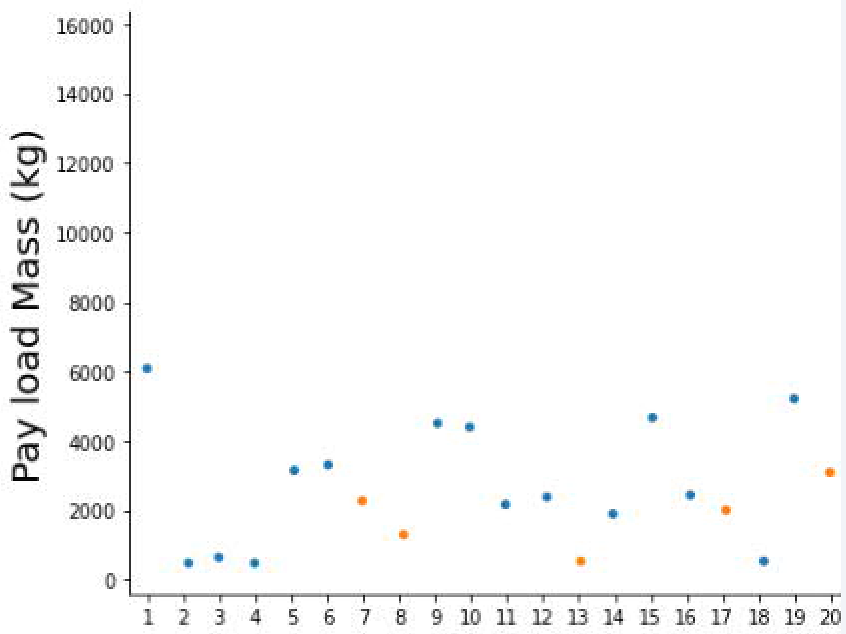
```
In [33]: i = Model_Performance_df['Accuracy Score'].idxmax()
         print('The best performing alogrithm is '+ Model_Performance_df['Algo Type'][i]
         + ' with score ' + str(Model_Performance_df['Accuracy Score'][i]))

         The best performing alogrithm is Decision Tree with score 0.8767857142857143
```


GitHub Link (code)

Results

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

```
sns.catplot(y="PayloadMass", x="FlightNumber", hue="C
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Pay load Mass (kg)",fontsize=20)
plt.show()
```



Out[32]:

	Algo Type	Accuracy Score	Test Data Accuracy Score
2	Decision Tree	0.876786	0.833333
3	KNN	0.848214	0.833333
1	SVM	0.848214	0.833333
0	Logistic Regression	0.846429	0.833333

In [33]:

```
i = Model_Performance_df['Accuracy Score'].idxmax()
print('The best performing alogrithm is '+ Model_Performance_df['Algo Type'][i]
+ ' with score ' + str(Model_Performance_df['Accuracy Score'][i]))
```

```
The best performing alogrithm is Decision Tree with score 0.8767857142857143
```

Section 2

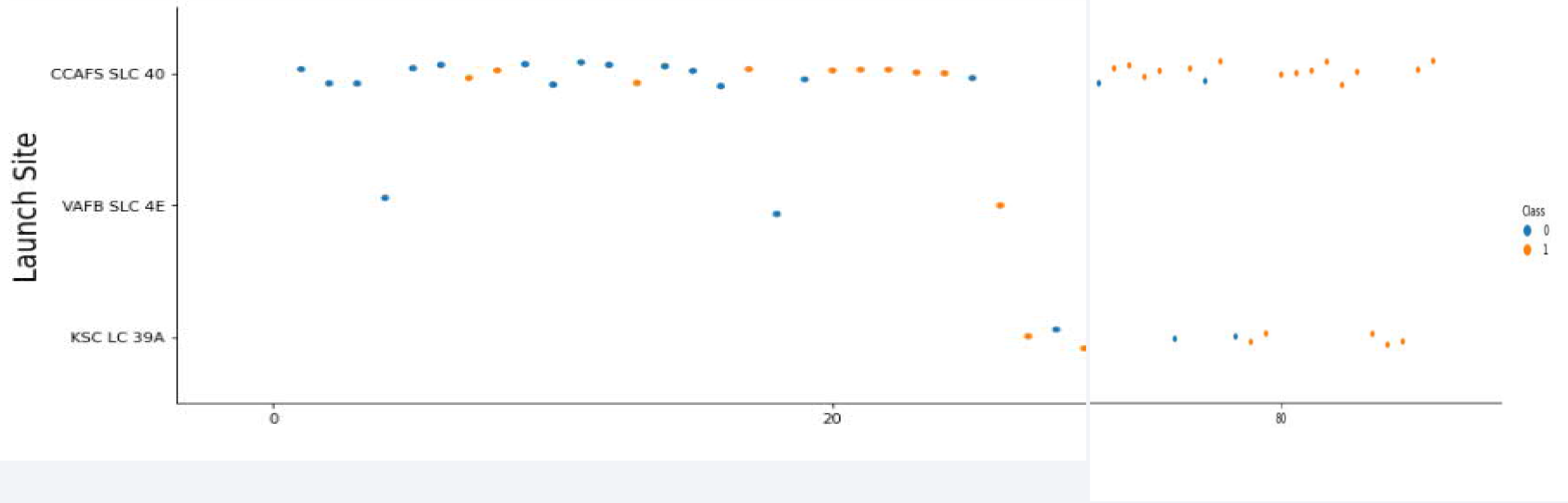
Insights drawn from EDA

Flight Number vs. Launch Site

Conclusion:

- *With higher flights numbers the success rate for the rocket is increasing*
- *For launch site KSC LC 39 A it takes at least 30 launches before a first successful launch*

```
In [4]: # Plot a scatter point chart with x axis to be Flight Number and y axis to be t
        sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
        plt.xlabel("Flight Number",fontsize=20)
        plt.ylabel("Launch Site",fontsize=20)
        plt.show()
```



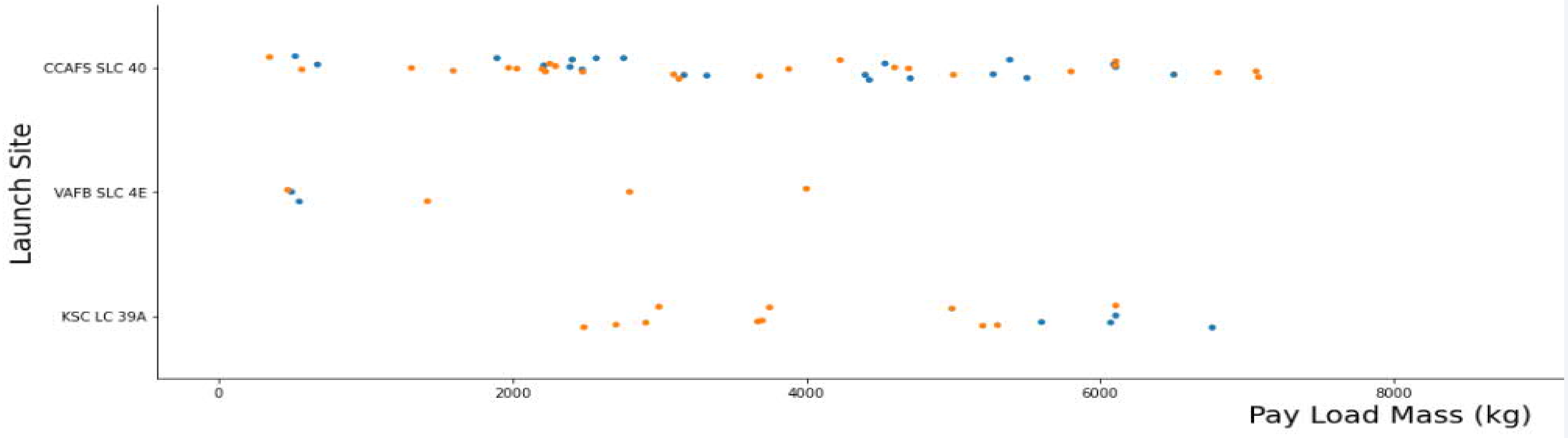
Payload vs. Launch Site

Observation if there is any relationship between launch sites and their payload mass:

- for launch site, there are no rockets launched for payload greater than 10 000KG
- no correlation between launch site and payload mass

In [5]:

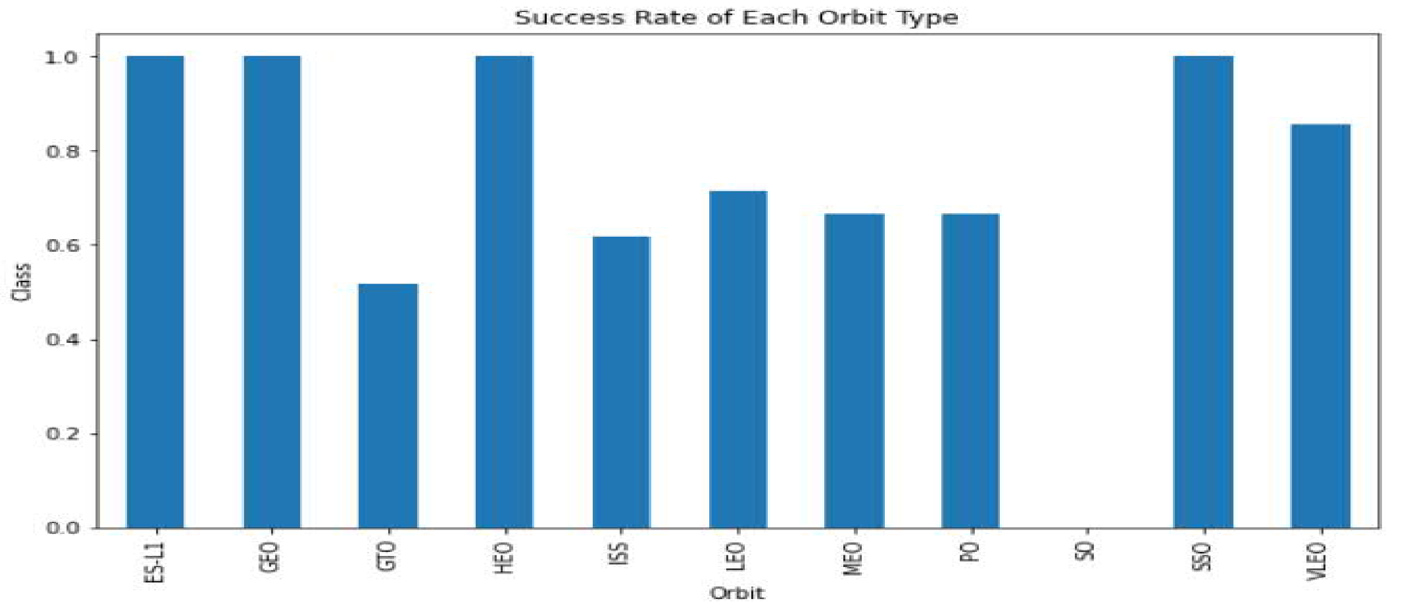
```
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the launch site, and hue to be the class val
sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("Pay Load Mass (kg)",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```



Success Rate vs. Orbit Type

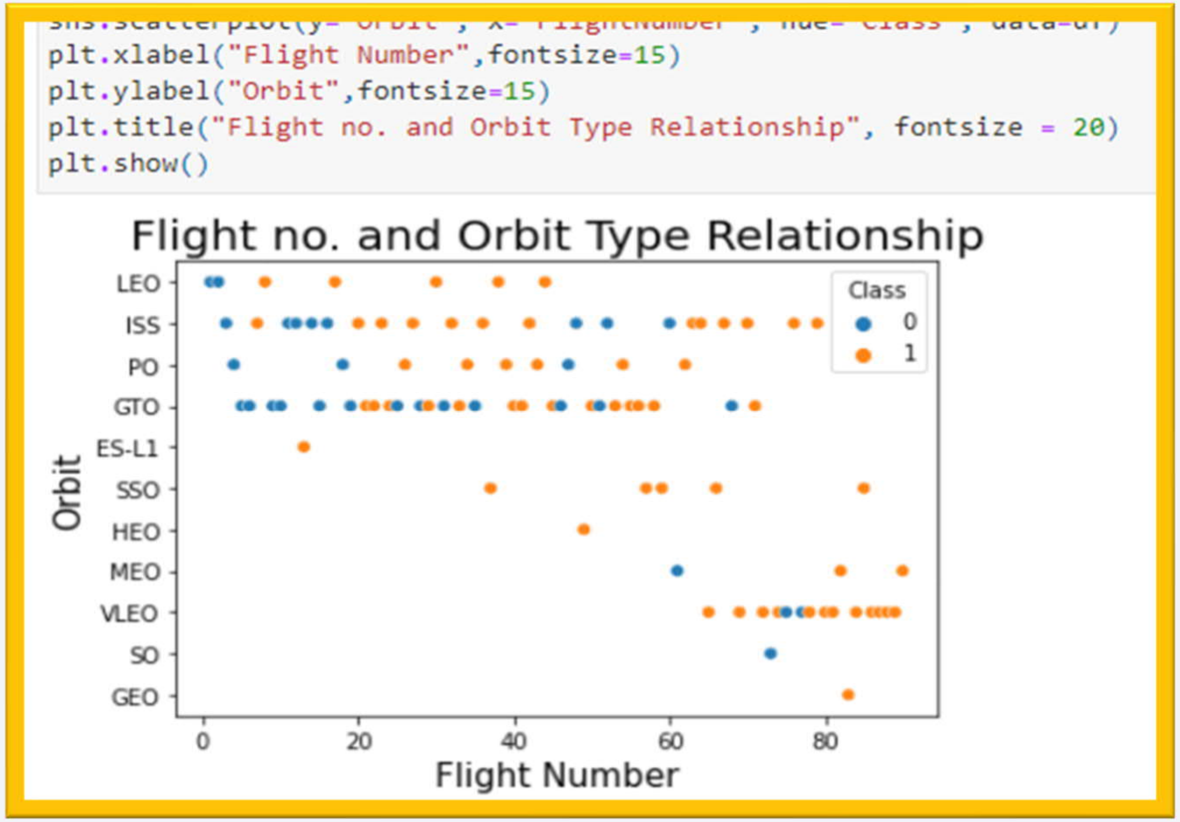
- ES-L1, GEO, HEO, SSO has highest success rate
- GTO orbit has the lowest success rate

```
In [6]: # HINT use groupby method on Orbit column and get the mean of Class column
df_bar = df.groupby(['Orbit'])['Class'].mean()
df_bar.plot(kind='bar', figsize=(10, 6))

plt.xlabel('Orbit') # add to x-label to the plot
plt.ylabel('Class') # add y-label to the plot
plt.title('Success Rate of Each Orbit Type') # add title to the plot

plt.show()
```



Flight Number vs. Orbit Type

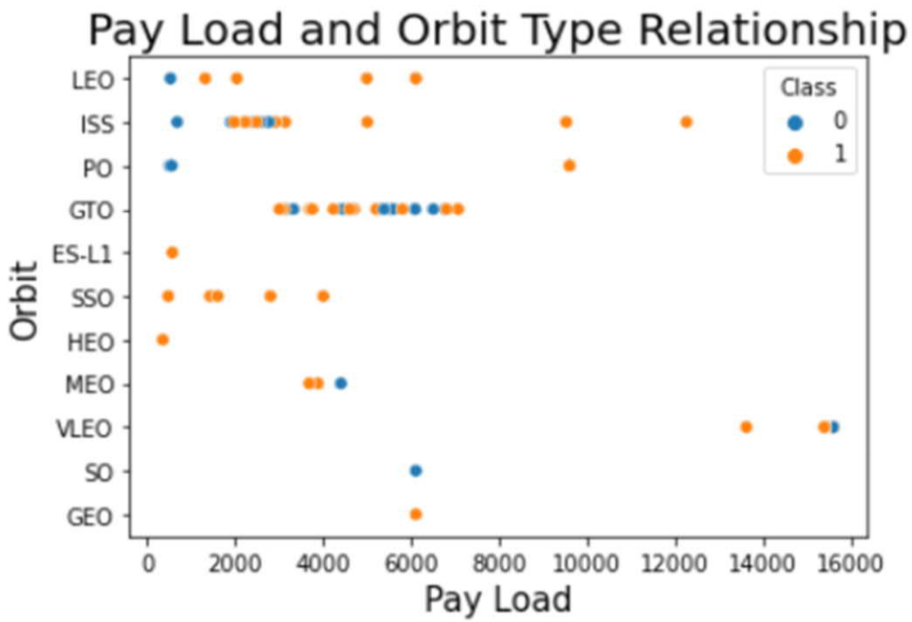
- No relationship between flight number and orbit for GTO
- For most orbits LEO, ISS, PO, VLEO successful landing rates appear to increase with flight number



Payload vs. Orbit Type

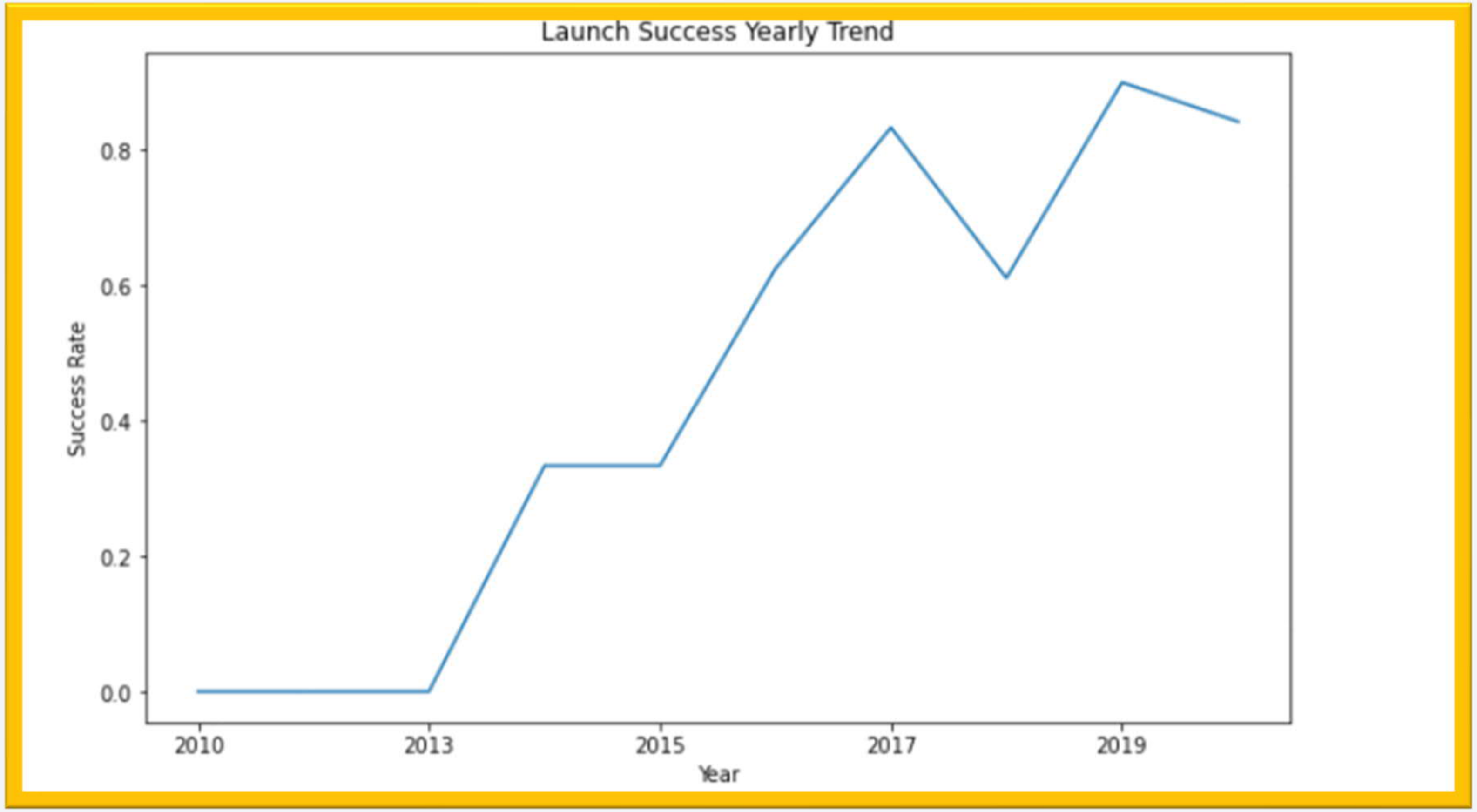
- ➢ GEO orbit no relationship between payload and orbit for successful landing
- ➢ Successful landing rates increasing with pay load features for following orbits LEO, ISS, PO, SSO

```
sns.scatterplot(y="Orbit", x="PayloadMass", hue="Class", data=df)
plt.xlabel("Pay Load",fontsize=15)
plt.ylabel("Orbit",fontsize=15)
plt.title("Pay Load and Orbit Type Relationship", fontsize = 20)
plt.show()
```



Launch Success Yearly Trend

- The success rate was since 2013 and 2020 about 80% kept significantly increasing
- But success rate decreases between 2017 and 2018, however 2019 and 2020 as well



All Launch Site Names

There are four unique launch sites

```
In [5]: %%sql

        select distinct Launch_Site from spacextbl
```

Unique Launch Sites
launch_site
CCAFS LC-40
CCAFS SLC-40
KSC LC-39A
VAFB SLC-4E

Launch Site Names Begin with 'CCA'

Query:

```
In [6]: %%sql

        select * from spacextbl where Launch_Site LIKE 'CCA%' limit 5;
```

Description:

- ➢ Select top limit five, returns only five records
- ➢ `LIKE` query and format `CCA%` returns where `Launch_Site` column beginn with CCA

launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

SQL QUERY

```
In [7]: %%sql

        select sum(PAYLOAD_MASS__KG_) from spacextbl where Customer = 'NASA (CRS)'
```

Outcome

45596

QUERY Explanation

Using the function `sum` adds column PAYLOAD_MASS_KG

The WHERE function filters the dataset `NASA (CRS)

Average Payload Mass by F9 v1.1

SQL QUERY

```
In [8]: %%sql

select avg(PAYLOAD_MASS__KG_) from spacextbl where Booster_Version LIKE 'F9 v1.1';
```

Outcome

2928

QUERY Explanation

`avg` function returns the average of payload in the column PAYLOAD_MASS_KG

First Successful Ground Landing Date

SQL QUERY

```
In [9]: %%sql

select min(Date) as min_date from spacextbl where Landing__Outcome = 'Success (ground pad)';
```

Outcome

Min_date
2015-12-22

QUERY Explanation

`min`(Data) functions works by selecting the minimum date in the column DATE

Successful Drone Ship Landing with Payload between 4000 and 6000

SQL QUERY

```
In [10]: %%sql

select Booster_Version from spacextbl where (PAYLOAD_MASS__KG_> 4000 and PAYLOAD_MASS__KG_ < 6000)
and (Landing__Outcome = 'Success (drone ship)');
```

Outcome

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

QUERY Explanation

Using only Booster_Version

Where function filters the dataset from Landing_Outcome = Success drone ship) and both conditions are true

Total Number of Successful and Failure Mission Outcomes

SQL QUERY

```
In [11]: %%sql

select Mission_Outcome, count(Mission_Outcome) as counts from spacextbl group by Mission_Outcome;
```

Outcome

mission_outcome	counts
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

QUERY Explanation

Group by function arranges data in a column group

Mission_Outcome are grouped in column `counts`

Boosters Carried Maximum Payload

SQL Query

```
In [12]: %%sql

select Booster_Version, PAYLOAD_MASS__KG_ from spacextbl where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from spacextbl);
```

Outcome

booster_version	payload_mass__kg_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

QUERY Explanation

By using the function `max` the query search the maximum payload mass

The function booster_version returns mass maximum with value of 15600

2015 Launch Records

SQL Query

```
In [13]: %%sql

select Landing_Outcome, Booster_Version, Launch_Site from spacextbl where Landing_Outcome = 'Failure (drone ship)' and year(Date) = '2015'
```

Outcome

landing__outcome	booster_version	launch_site
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

Query Explanation

The `year` function extracts a year from column `Date`

The function or query `landing outcome`, `booster version` and `launch site` showing where the launding outcome is failed during the period in 2015

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

Query

```
In [14]: %%sql

select Landing__Outcome, count(*) as LandingCounts from spacextbl where Date between '2010-06-04' and '2017-03-20'
group by Landing__Outcome
order by count(*) desc;
```

Outcome

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

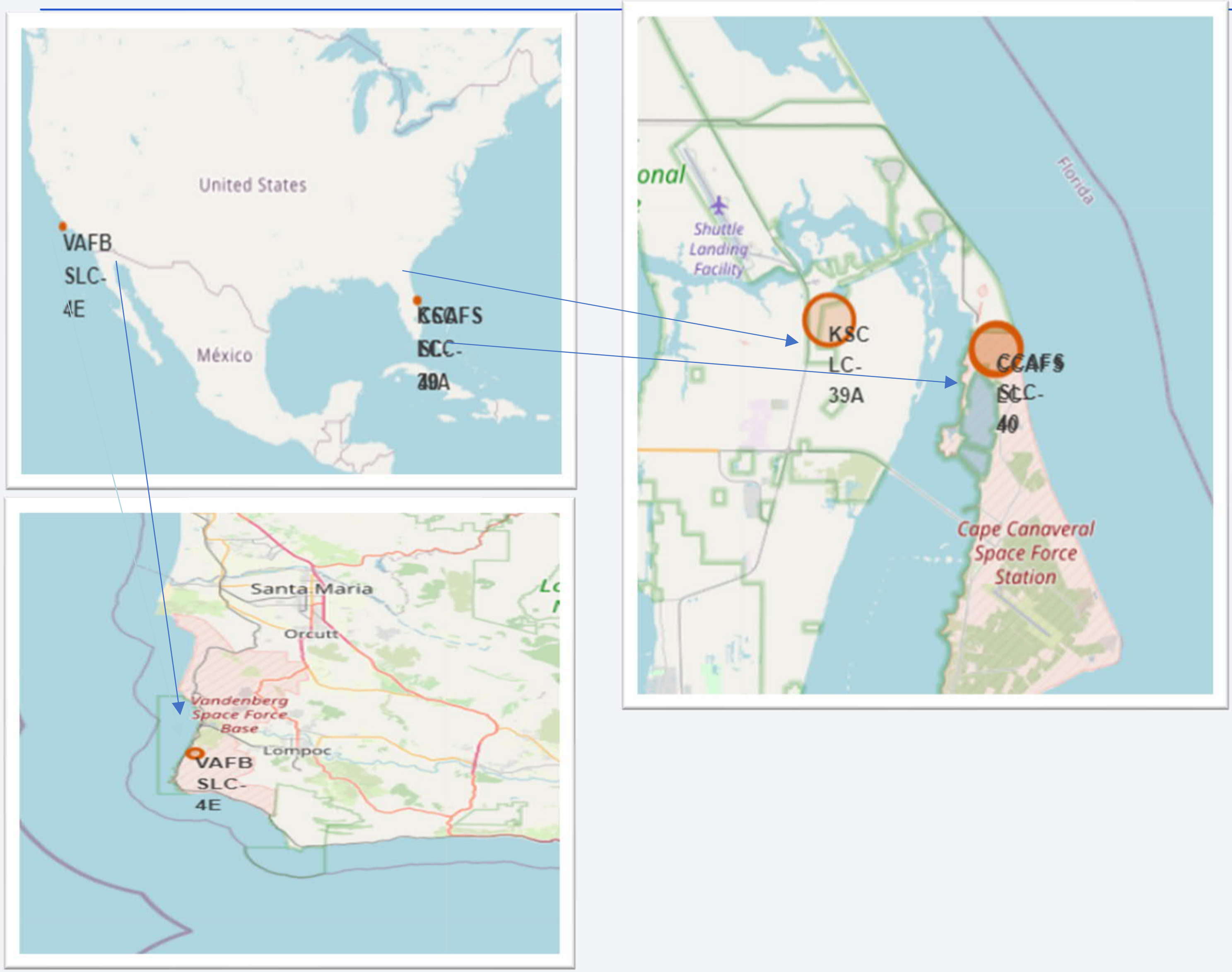
Query explanation

Count - function counts records in column
Group by – arranges the data into groups
AND - conditions
Order by – arranges in descending order

Section 3

Launch Sites Proximities Analysis

SITE_MAP



- Figures shows the Global map where Falcon 9 launch sites are located in USA. By the way all launch sites are close to the coast
- Another Figures shows zoomed the launch sites:

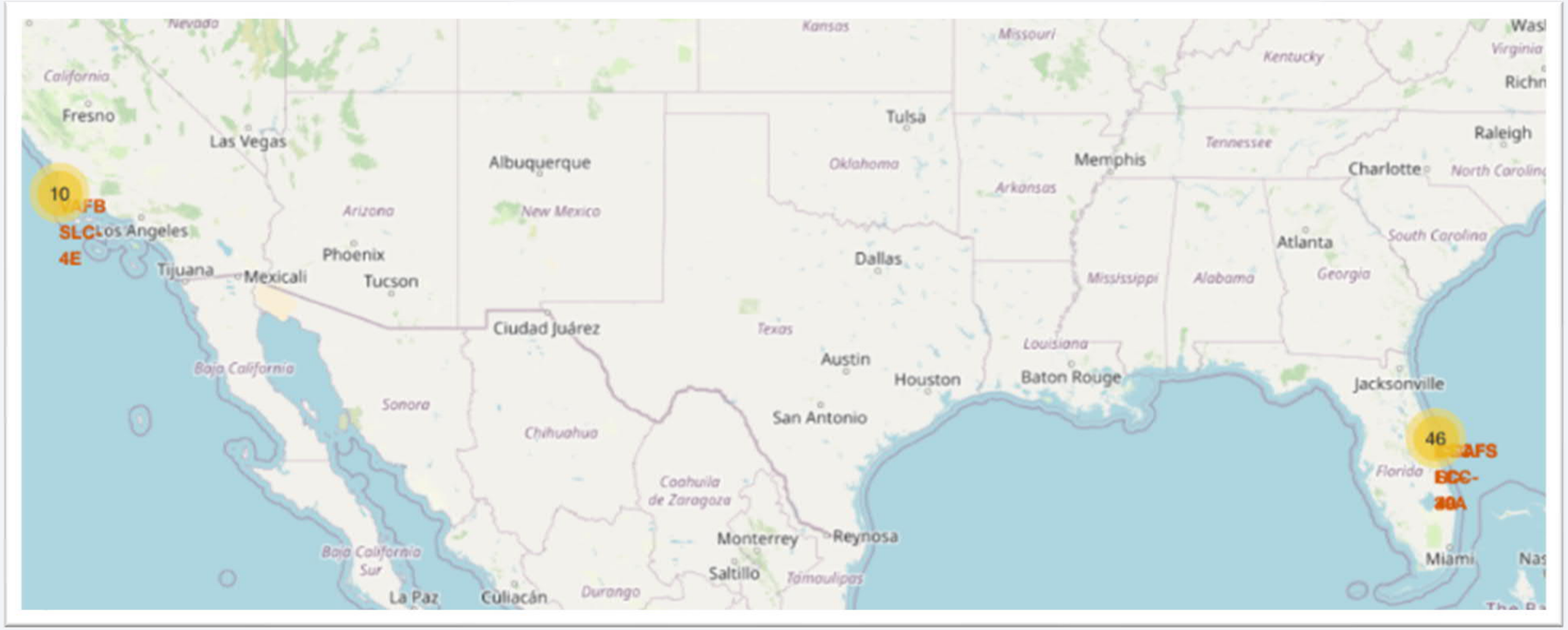
VAFB SLC-4E

CCAFS LC-40

KSC LC-39A

CCAFS SLC-40

All launch sites in USA



All launch sites are in USA, Florida and California

SpaceX Falcon9-Success and Failed launch sites labelled markers

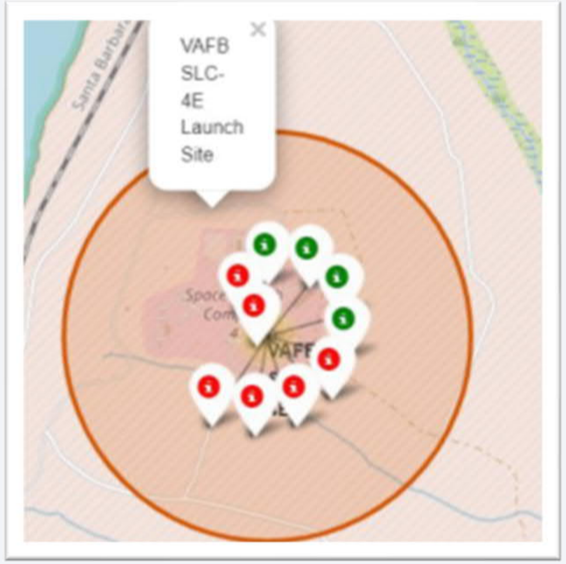


Figure 1

Shows successful (green marker) and failed (red markers) launches of the VAFB SLC-4E launch site CALIFORNIA

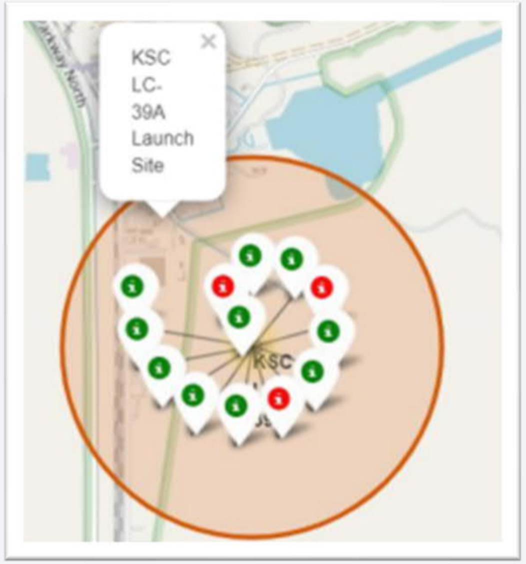


Figure 2

Shows successful (green marker) and failed (red markers) launches of the KSC-LC-39A launch site FLORIDA

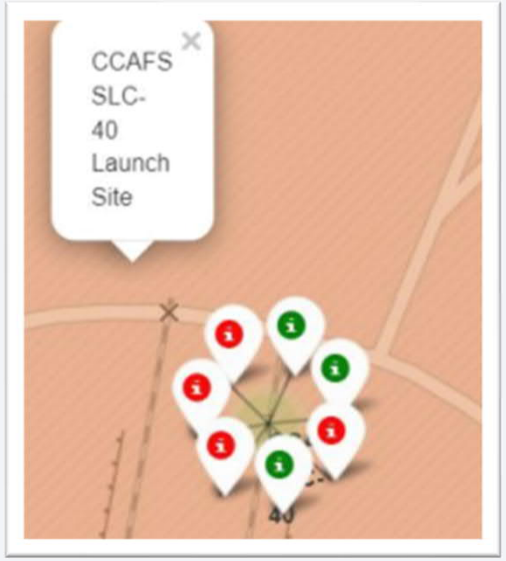


Figure 3

Shows successful (green marker) and failed (red markers) launches of the CCAFS-SLC-40 launch site FLORIDA

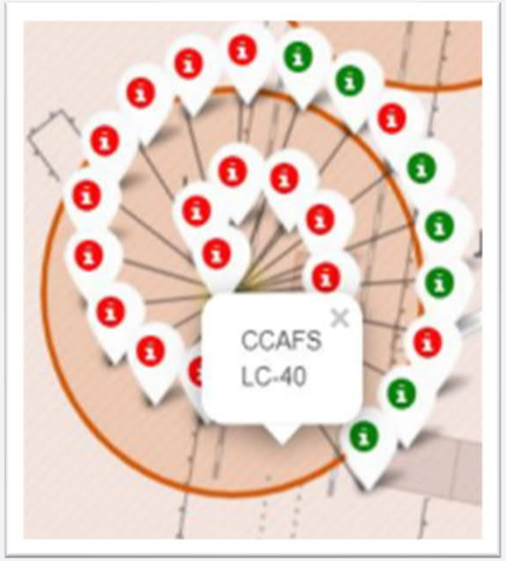


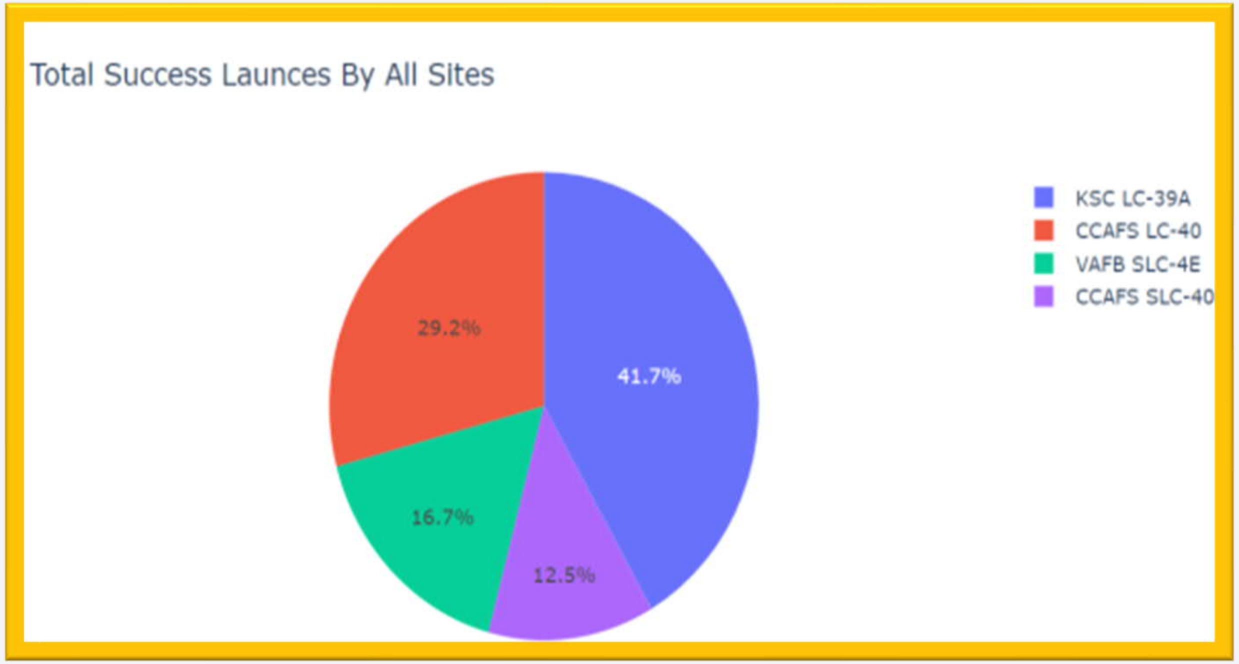
Figure 4

Shows successful (green marker) and failed (red markers) launches of the CCAFS LC-40 launch site FLORIDA

Section 4

Build a Dashboard with Plotly Dash

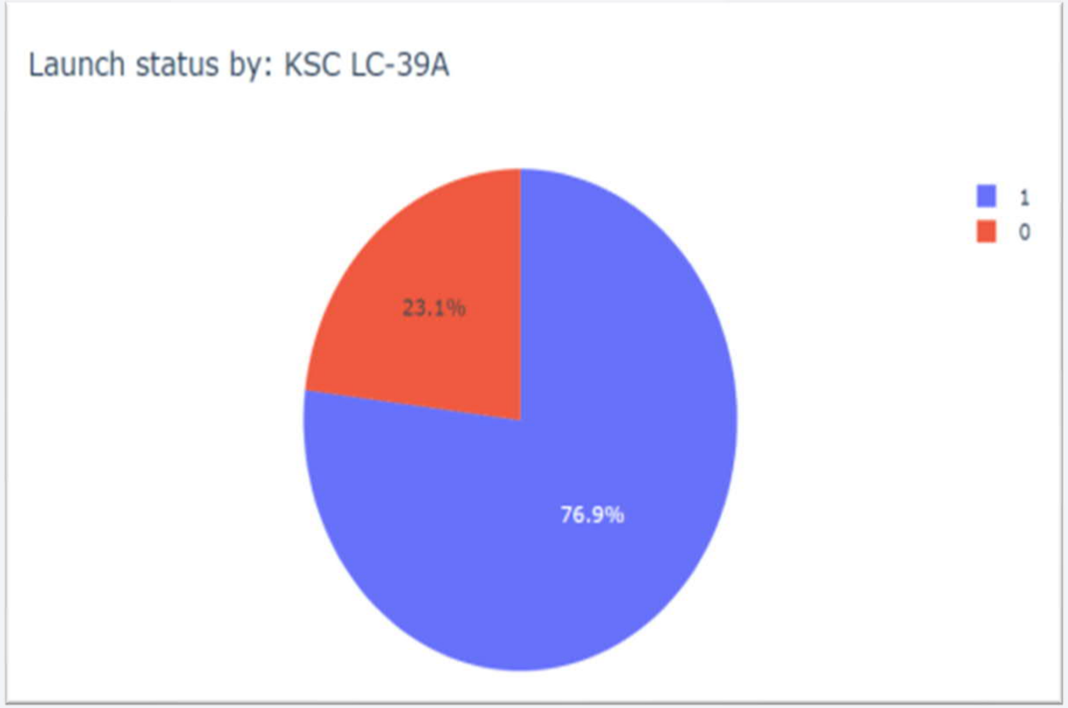
Dashboard showing the success of each launch site



Results:

KSC LC-39A has a highest succesful rate by all sites

Dashboard – Launch site with a highest launch success ratio



KSC LC-39A has a 76.9% success rate.
However the failure rate is by 23.1%

Payload versus Launch Outcome Scatter Plot for all sites

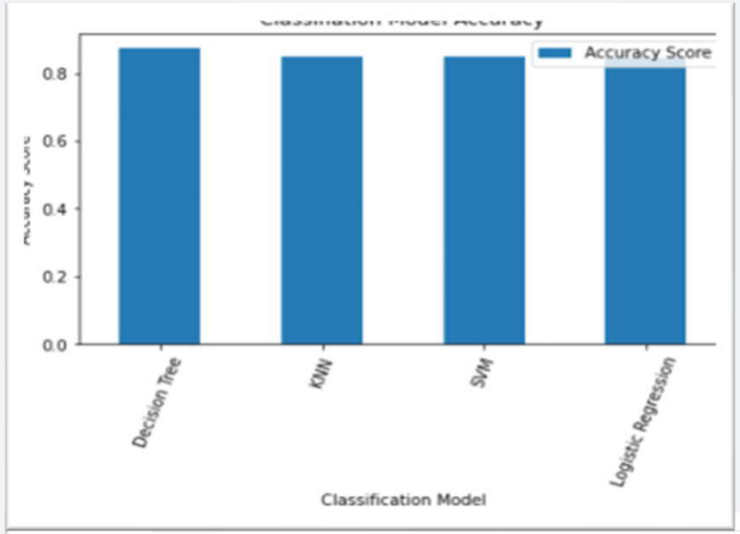
Results:

In the payload from 2000 between 5500 range are most successful launches

Section 5

Predictive Analysis (Classification)

Classification Accuracy



Out[32]:

	Algo Type	Accuracy Score	Test Data Accuracy Score
2	Decision Tree	0.876786	0.833333
3	KNN	0.848214	0.833333
1	SVM	0.848214	0.833333
0	Logistic Regression	0.846429	0.833333

In [33]:

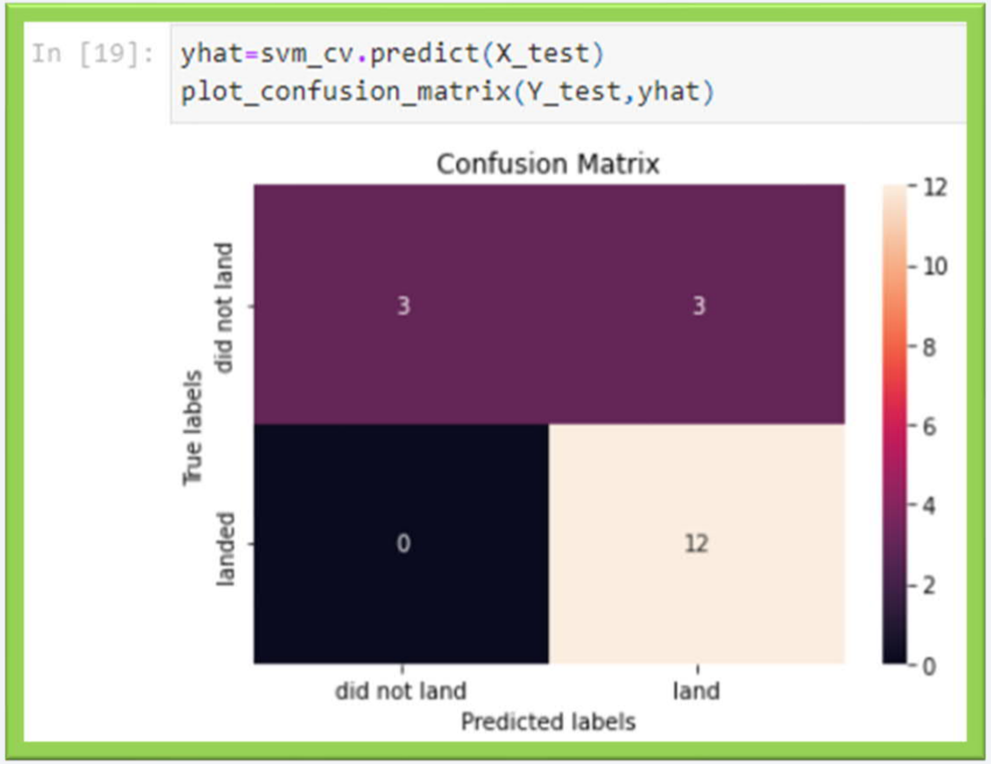
```
i = Model_Performance_df['Accuracy Score'].idxmax()
print('The best performing alogrithm is '+ Model_Performance_df['Algo Type'][i
+ ' with score ' + str(Model_Performance_df['Accuracy Score'][i]))
```

```
The best performing alogrithm is Decision Tree with score 0.8767857142857143
```

Results:

Decision Tree machine learning algorithm has a best fit to our model accuracy score with a value of 0.87

Confusion Matrix



Results:

- By analyzing the confusion matrix we can see that classifier has been made 18 predictions
- The confusion matrix has been build for models
- Three situations has been predicted NO for landing and TRUE positive
- Three situations has been predicted YES for landing, however they could not land successfully

Conclusions

- *The best performing algorithm Decision Tree with score 0.876 (87.5%)*
- *Lunch sites are located for a purpose close to the coast areas (avoiding public areas)*
- *Launch success rate increased during the 2013 to 2020 period by 80%*
- *Orbits ES-L1, GEO, HEO, SSO are being the highest success rates, however orbit GTO has the lowest success rate*

Appendix

- IBM Watson Studio Notebook
- Python code

Thank you!