IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Nikhil Kuber Mankapure
10-03-2023

Outline

EXECUTIVE SUMMARY

Summary of methodologies

1. Data Collection using APIs, SQL and web scrapping.
2. Data Wrangling and analysis
3. Folium Maps(Interactive)
4. Interactive Dashboards for inferences
5. Predictive Analysis using different machine learning algorithms

Summary of all results

1. Interactive dashboards for inferences
2. Best model for predicting the desired results

Introduction

- Project background and context

We need to predict whether the falcon 9 first stage will land successfully or not. SPACEX has listed a cost cap of 62 million dollars for falcon 9 rocket launches in comparison to 165 million dollars each from other competitors. The savings are due to the recovery of first stage of the rocket. If we can determine is the first stage will land successfully then we can have a strong establishment on their success rate.

- Problems you want to find answers

Factors that results in successful landing.

The relationship of various features on successful landing

The conditions that result in the best outcome for the SPACEX launch.


METHODOLOGY

Methodology

Executive Summary

- Data collection methodology:

  SPACEX API

  Web scrapping through WIKIPEDIA

- Perform data wrangling
- Using the data to process and transform the data to introduce dummies for the machine learning models
- Perform exploratory data analysis (EDA) using visualization and SQL
- Scatter and bar charts to visualize the basic analysis.
- Perform interactive visual analytics using Folium and Plotly Dash

  Using plotly and folium tools for visualization.

- Perform predictive analysis using classification models

  How to build, tune, evaluate classification models

Data Collection

Through API

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

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

Through WEB

```
[25]: # Show the head of the dataframe
      df.head()
```

[25]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Lat
0	1	2006-03-24	Falcon 1	20.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin1A	167.743129	9.04
1						Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin2A	167.743129	9.04
2	4	09-28	Falcon 1	165.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin2C	167.743129	9.04
3	5	2009-	Falcon 1	200.0	LEO	Kwajalein	None	1	False	False	False	None	NaN	0	Merlin3C	167.743129	9.04
									lse	False		None	1.0	0	B0003	-80.577366	28.56

DATAFRAM

FILTER DATA

```
35]: # Calculate the mean value o        s column
     mean = df['PayloadMass'].mean()
     print(mean)
     # Replace the np.nan values with      mean value
     df['PayloadMass'].fillna(mean, in      ce=True)
     df['PayloadMass'].

5919.16534090909
```

Export to CSV

```
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_1.csv")
df.head(10)
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitu
0	1	2010-06-04	Falcon 9	6104.959412					False	False	False	NaN	1.0	0	B0003	-80.577366	28.561
1	2	2012-05-22	Falcon 9	525.000000	LEO		40 None	1	False	False	False	NaN	1.0	0	B0005	-80.577366	28.561
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007	-80.577366	28.561
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003	-120.610829	34.632
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004	-80.577366	28.561

Filtered DataFrame

Data Collection – SpaceX API

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

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

Getting Response from API

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

Response to JSON

```
def getBoosterVersion(data)
def getLaunchSite(data)
def getPayloadData(data)
def getCoreData(data)
```

Applying function to filter

```
22]: 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}
```

Then, we need to create a Pandas data frame from the dictionary launch_dict

Dictionary to dataframe

```
[25]: # Show the head of the dataframe
df.head()
```

[25]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Lat
0	1	2006-03-24	Falcon 1	20.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin1A	167.743129	9.04
1	2	2007-03-21	Falcon 1	NaN	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin2A	167.743129	9.04
2	4	2008-09-28	Falcon 1	165.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin2C	167.743129	9.04
3	5	2009-07-13	Falcon 1	200.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin3C	167.743129	9.04
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0003	-80.577366	28.56

https://github.com/NikhilMankapure/Capstone/blob/f0b4b0589abdc8da277df1ee028099f4bcd5e043/jupyter-labs-spacex-data-collection-api.ipynb

WEB SCRAPING

1. Getting Response from HTML

```
# use requests.get() method with the provided static_url
response = requests.get(static_url).text
# assign the response to a object
```

2. Creating Beautiful Soup Object

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
from html.parser import HTMLParser
soup = BeautifulSoup(response,parser='HTMLParser')
```

3. Finding Tables

```
# 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')
html_tables
```

4. Creating dictionary

```
launch_dict= dict.fromkeys(column_names)

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

# Let's initial the launch_dict with each value to be an empty list
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. Converting Dictionary to dataframe

```
df=pd.DataFrame(launch_dict)
```

```
(242, 11)
```

6. Dataframe to CSV

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

https://github.com/NikhilMankapure/Capstone/blob/f0b4b0589abdc8da277df1ee028099f4bcd5e043/jupyter-labs-webscraping.ipynb

Data Wrangling

Describe how data were processed

The data was first imported from csv type to Data Frame.

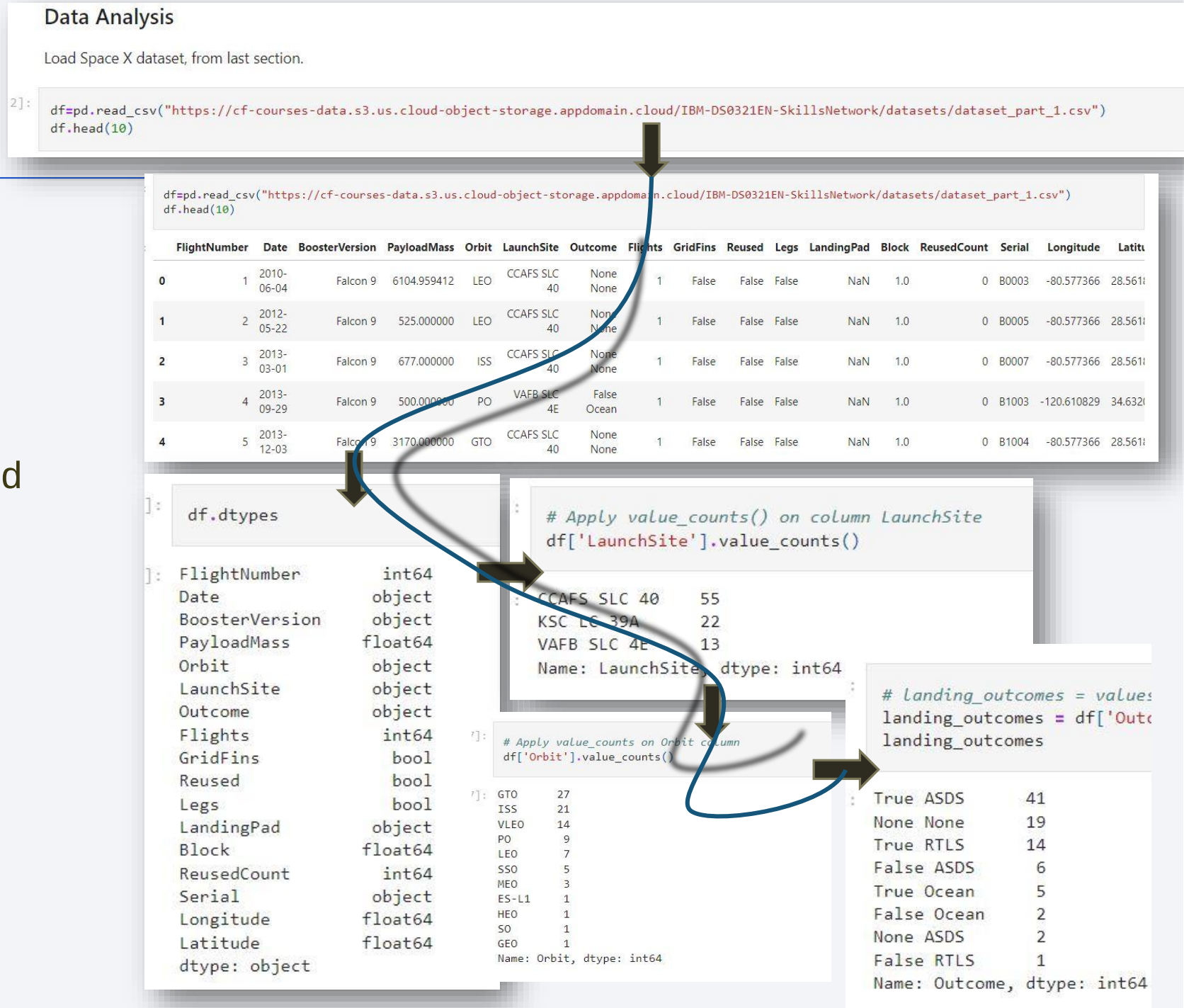
Checking for null values.

Eliminating Null values using required methods like mean, Midian, etc.

Checking Datatypes

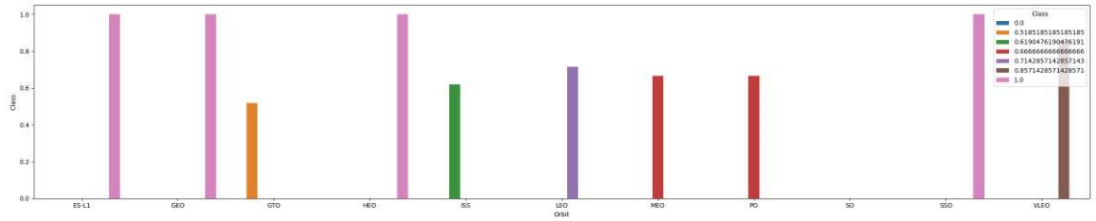
Sorting Data as per features (Orbit, Outcomes, launch sites, etc.)

[Capstone/labs-jupyter-spacex-Data wrangling.ipynb at main · NikhilMankapure/Capstone (github.com)]



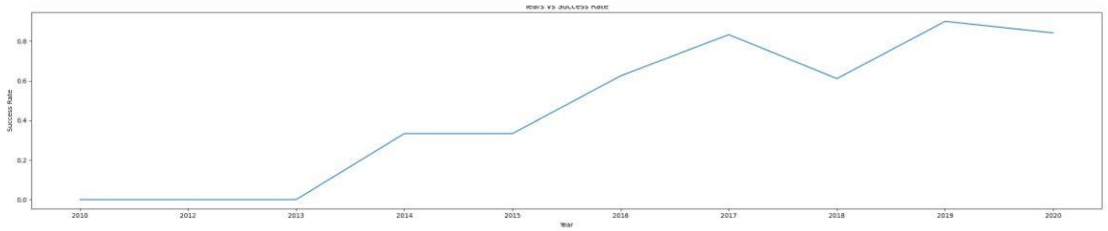
EDA with Data Visualization

```
# HINT use groupby method on Orbit column and get the mean of Class column
orbit_success= df.groupby('Orbit').mean()
orbit_success.reset_index(inplace=True)
sns.barplot(data=orbit_success, x='Orbit', y='Class', hue='Class')
plt.show()
```



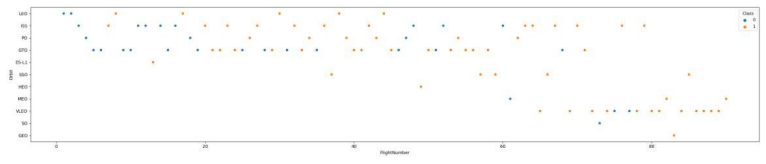
Bar Charts give a clear visualization of the data under consideration, the success rate based on the orbit is clearly depicted through the bar graph.

```
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
average_by_year = df.groupby(by="Date").mean()
average_by_year.reset_index(inplace=True)
plt.plot(average_by_year['Date'],average_by_year['Class'])
plt.xlabel('Year')
plt.ylabel('Success Rate')
plt.title('Years vs Success Rate')
plt.show()
```

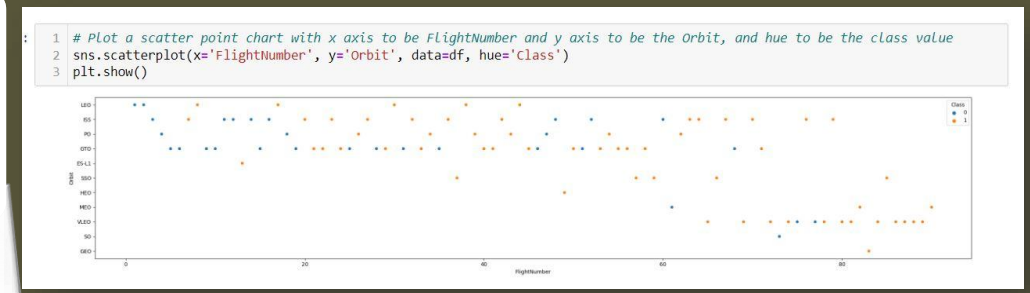


Line Graph: Easy interpretation and future prediction are Two most important features of the line graph. It simple to Understand and take insight from the data.

```
# 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.scatterplot(x='FlightNumber', y='Orbit', data=df, hue='Class')
plt.show()
```



```
# 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.scatterplot(x='FlightNumber', y='Orbit', data=df, hue='Class')
plt.show()
```

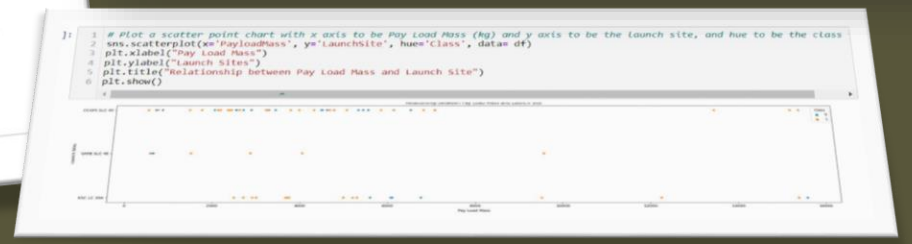


Scatter Graph:

Correlation can be easily Interpreted using scatter chart. It gives the type of correlation Between the variables and Their intensity.

```
sns.catplot(x='FlightNumber', y='LaunchSite', hue='Class', data= df, aspect= 5)
plt.xlabel("Flight Number")
plt.ylabel("Launch Sites")
plt.title("Relationship between Flight Number and Launch Site")
plt.show()
```

```
# 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.scatterplot(x='PayloadMass', y='LaunchSite', hue='Class', data= df)
plt.xlabel("Pay Load Mass")
plt.ylabel("Launch Sites")
plt.title("Relationship between Pay Load Mass and Launch Site")
plt.show()
```



https://github.com/NikhilMankapure/Capstone/blob/f0b4b0589abdc8da277df1ee028099f4bcd5e043/EDA%20Analysis.ipynb

EDA with SQL

SQL is a language that is widely used to retrieve data from the database on cloud or server for drawing insights From the stored data. This language is popular for its robust application and its ease of adaptation. As in the assignment We have used to DB2 database from IBM cloud for computing the outputs required for the project.

The following insights were drawn from the database using the DB2 data base management tool:

- Displaying unique names of the unique launch sites in the space mission.
- Displaying 5 records where launch sites begin with the string 'CCA‘.
- Displaying the total payload mass carried by boosters launched by NASA (CRS)
- Displaying average payload mass carried by booster version F9 v1.1
- Listed the date when the first successful landing outcome in ground pad was achieved.
- Listed the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- Listed the total number of successful and failure mission outcomes
- Listed the names of the booster versions which have carried the maximum payload mass. Used a subquery
- 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
- Rank the count of successful landing outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

https://github.com/NikhilMankapure/Capstone/blob/f0b4b0589abdc8da277df1ee028099f4bcd5e043/jupyter-labs-eda-sql-coursera.ipynb

Built an Interactive Map with Folium

1. Map marker
2. Marker cluster
3. Line marker using polyline
4. Circle Marker
5. Icon Marker

1. Easy visualization using folium maps on a leaflet.
2. Distances from various locations like ocean, city, railway, highway, etc. can be easily visualized.
3. Marker cluster clearly depicts the number of success and failure outcomes with reference to the location.
4. Line markers are used to mark the nearest distance from important locations.

https://github.com/NikhilMankapure/Capstone/blob/f0b4b0589abdc8da277df1ee028099f4bcd5e043/Folium%20map.ipynb

Build a Dashboard with Plotly Dash

Pie chart: Showing total success of all sites or by certain selected sites.
Percentage of success in relation to launch sites.
Scatter Graph showing correlation between Payload and Success for all sites or by certain launch sites.
It shows relationship between success rate and the booster version.

Map Objects	Code
Dash and its components	Import dash Import dash_html_components as html Import dash_core_components as dcc From dash.dependencies import input, output
Pandas	Import pandas as pd
Plotly	Import plotly.express as px
Dropdown	dcc.Dropdown(
Rangeslider	dcc.Rangeslider(
Pie Chart	px.Pie(
Scatter Chart	px.Scatter(

https://github.com/NikhilMankapure/Capstone/blob/f0b4b0589abdc8da277df1ee028099f4bcd5e043/dashaboard.docx

Predictive Analysis (Classification)

Load the dataframe

Load the data

```
from js import fetch
import io

URL1 = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_pa
resp1 = await fetch(URL1)
text1 = io.BytesIO((await resp1.arrayBuffer()).to_py())
data = pd.read_csv(text1)
```

Load data

```
data.head()
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	B
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	

Y Data

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

```
array([0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1,
       1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1], dtype=int64)
```

	FlightNumber	PayloadMass	Flights	Block	ReusedCount	Orbit_ES-L1	Orbit_GEO	Orbit_GTO	Orbit_HEO	Orbit_ISS	...	Ser
0	1.0	6104.959412	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	
1	2.0	525.000000	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	
2	3.0	677.000000	1.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	...	
3	4.0	500.000000	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	
4	5.0	3170.000000	1.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	...	
...	...	...	...	...	...	...	...	...	...	...	...	...
85	86.0	15400.000000	2.0	5.0	2.0	0.0	0.0	0.0	0.0	0.0	...	
86	87.0	15400.000000	3.0	5.0	2.0	0.0	0.0	0.0	0.0	0.0	...	
87	88.0	15400.000000	6.0	5.0	5.0	0.0	0.0	0.0	0.0	0.0	...	
88	89.0	15400.000000	3.0	5.0	2.0	0.0	0.0	0.0	0.0	0.0	...	

X Data

```
URL2 = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_pa
resp2 = await fetch(URL2)
text2 = io.BytesIO((await resp2.arrayBuffer()).to_py())
X = pd.read_csv(text2)
```

```
X.head(100)
```

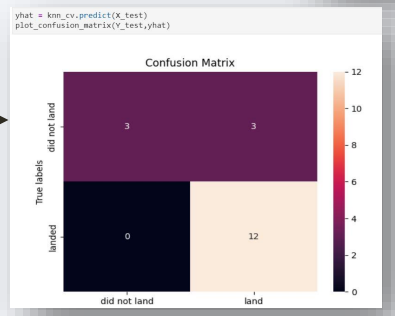
```
Y = data['Class'].to_numpy()
Y
```

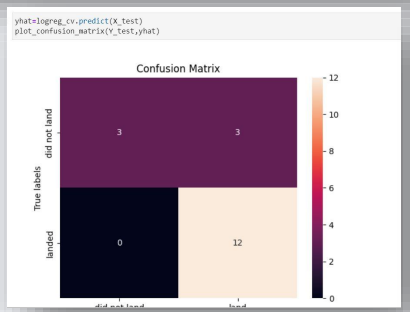
```
array([0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1,
       1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1], dtype=int64)
```

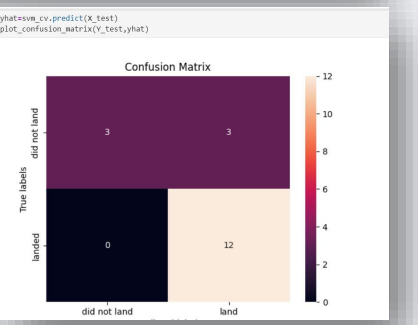
Confusion matrix

```
def plot_confusion_matrix(y,y_predict):
    "this function plots the confusion matrix"
    from sklearn.metrics import confusion_matrix

    cm = confusion_matrix(y, y_predict)
    ax= plt.subplot()
    sns.heatmap(cm, annot=True, ax = ax); #annot=True to annotate cells
    ax.set_xlabel('Predicted labels')
    ax.set_ylabel('True labels')
    ax.set_title('Confusion Matrix');
    ax.xaxis.set_ticklabels(['did not land', 'land']); ax.yaxis.set_ticklabels(['did not land', 'landed'])
    plt.show()
```







Models Considered for predictions:

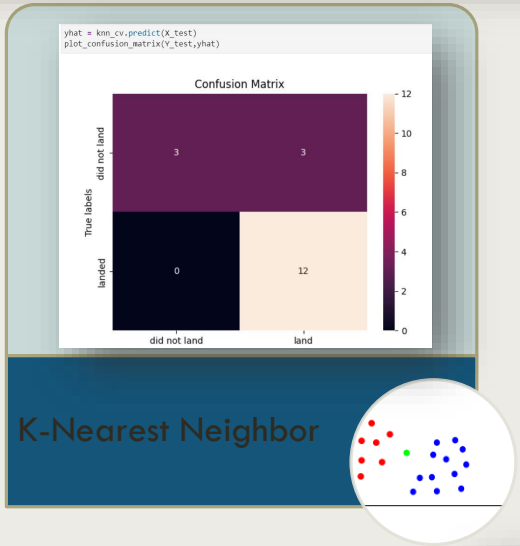
```
X_train, X_test, Y_train, Y_test
```

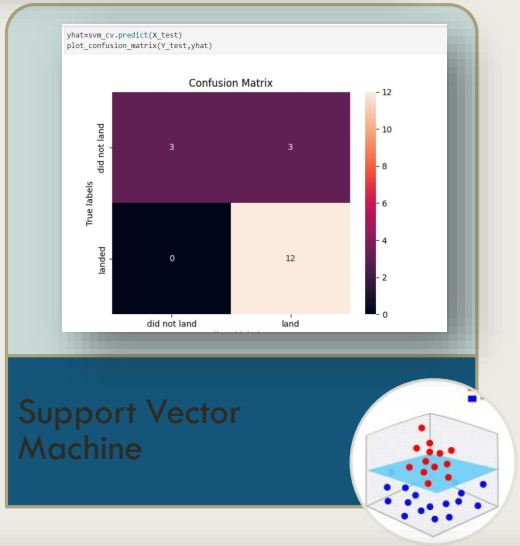
```
[10]: X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
print("The shape of the train set is", X_train.shape)
print("The shape of the train set is", X_test.shape)
print("The shape of the train set is", Y_train.shape)
print("The shape of the train set is", Y_test.shape)
```

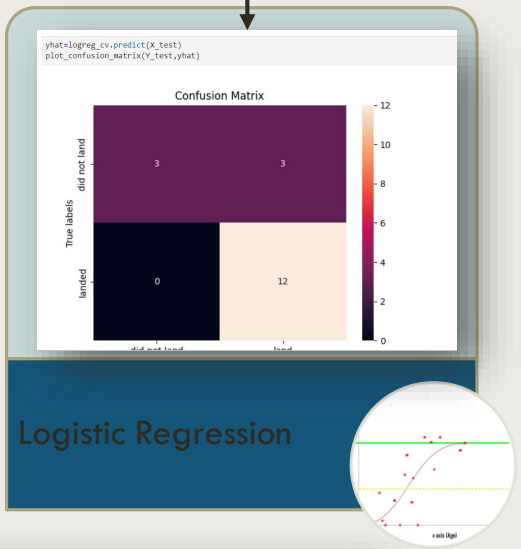
```
[12]: parameters ={'C':[0.01,0.1,1],
                  'penalty':['l2'],
                  'solver':['lbfgs']}
```

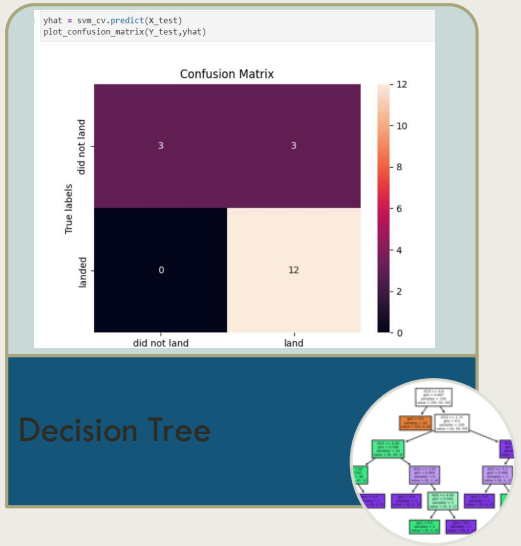
```
[13]: parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
lr=LogisticRegression()
grid_search = GridSearchCV(lr, parameters, cv=10)
logreg_cv = grid_search.fit(X_train, Y_train)
```

Confusion Matrix Function









Accuracy:

```
print('Accuracy for Logistics Regression method:', logreg_cv.score(X_test, Y_test))
print( 'Accuracy for Support Vector Machine method:', svm_cv.score(X_test, Y_test))
print('Accuracy for Decision tree method:', tree_cv.score(X_test, Y_test))
print('Accuracy for K nearsdt neighbors method:', knn_cv.score(X_test, Y_test))
```

```
Accuracy for Logistics Regression method: 0.8333333333333334
Accuracy for Support Vector Machine method: 0.8333333333333334
Accuracy for Decision tree method: 0.8888888888888888
Accuracy for K nearsdt neighbors method: 0.8333333333333334
```

https://github.com/NikhilMankapure/Capstone/blob/f0b4b0589abdc8da277df1ee028099f4bcd5e043/Machine%20Learning%20prediction.ipynb

Section 2

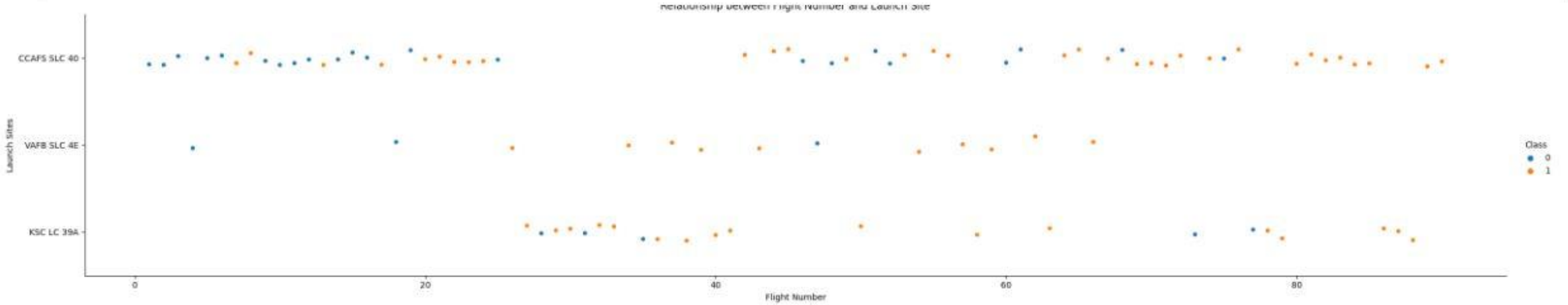
Insights drawn from EDA

Flight Number vs. Launch Site

Scatter plot of Flight Number vs. Launch Site

Flights with higher number is having a better success rate when compared to that with lower flight number

```
# 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 value
sns.catplot(x='FlightNumber', y='LaunchSite', hue='Class', data= df, aspect= 5)
plt.xlabel("Flight Number")
plt.ylabel("Launch Sites")
plt.title("Relationship between Flight Number and Launch Site")
plt.show()
```

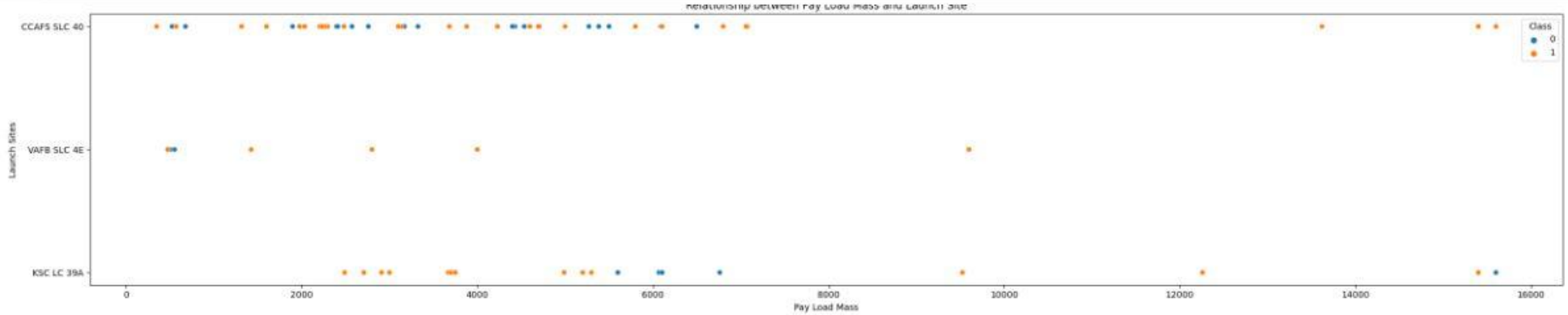


Payload vs. Launch Site

Scatter plot of Payload vs. Launch Site:

13]:

```
# 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.scatterplot(x='PayloadMass', y='LaunchSite', hue='Class', data= df)
plt.xlabel("Pay Load Mass")
plt.ylabel("Launch Sites")
plt.title("Relationship between Pay Load Mass and Launch Site")
plt.show()
```

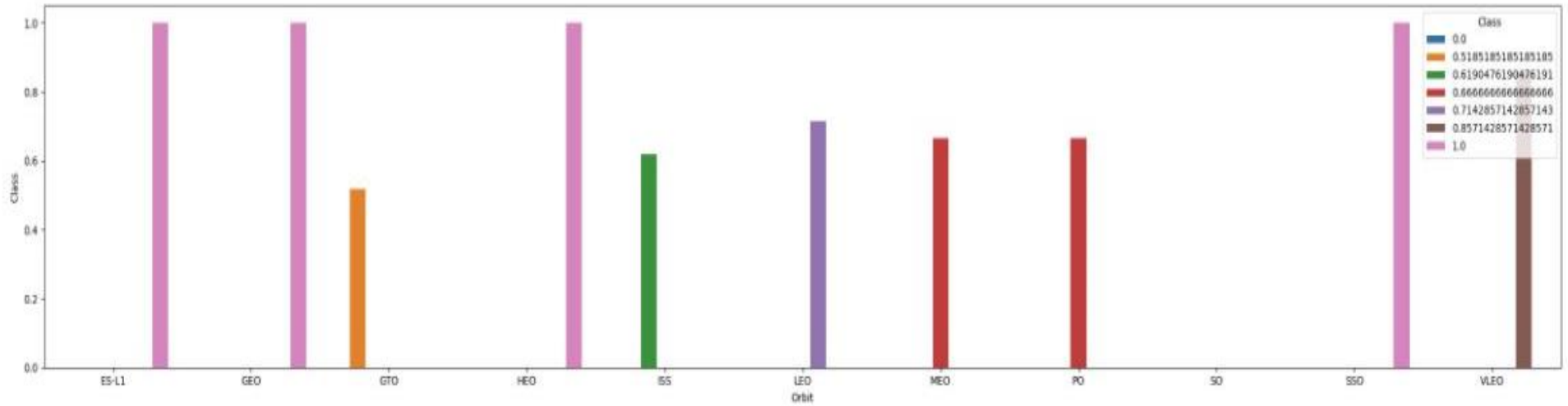


The Success rate improves when the payload increases (i.e. more than 7000kgs but the data is too less to have a clear answer to the inference drawn.

Success Rate vs. Orbit Type

Bar chart for the success rate of each orbit type

```
orbit_success= df.groupby('Orbit').mean()
orbit_success.reset_index(inplace=True)
sns.barplot(data=orbit_success, x='Orbit', y='Class', hue='Class')
plt.show()
```

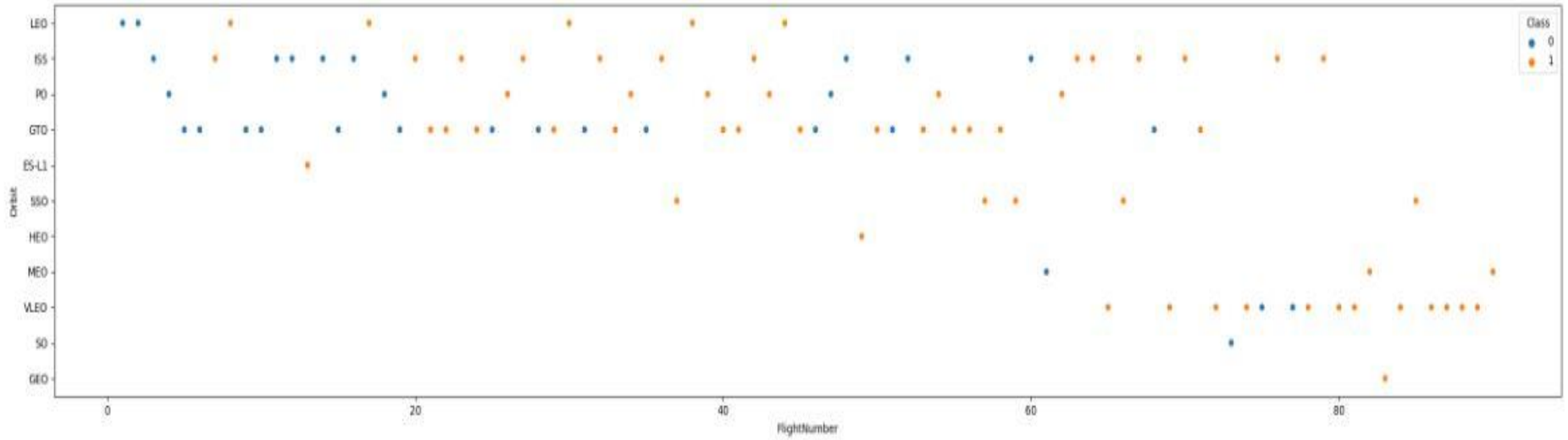


ES-L1, GEO, HEO, SSO have the highest success Rate, which means that the launches happening in these Orbits have a higher success rate than those in other orbits.

Flight Number vs. Orbit Type

Show a scatter point of Flight number vs. Orbit type

```
sns.scatterplot(x='FlightNumber', y='Orbit', data=df, hue='Class')
plt.show()
```

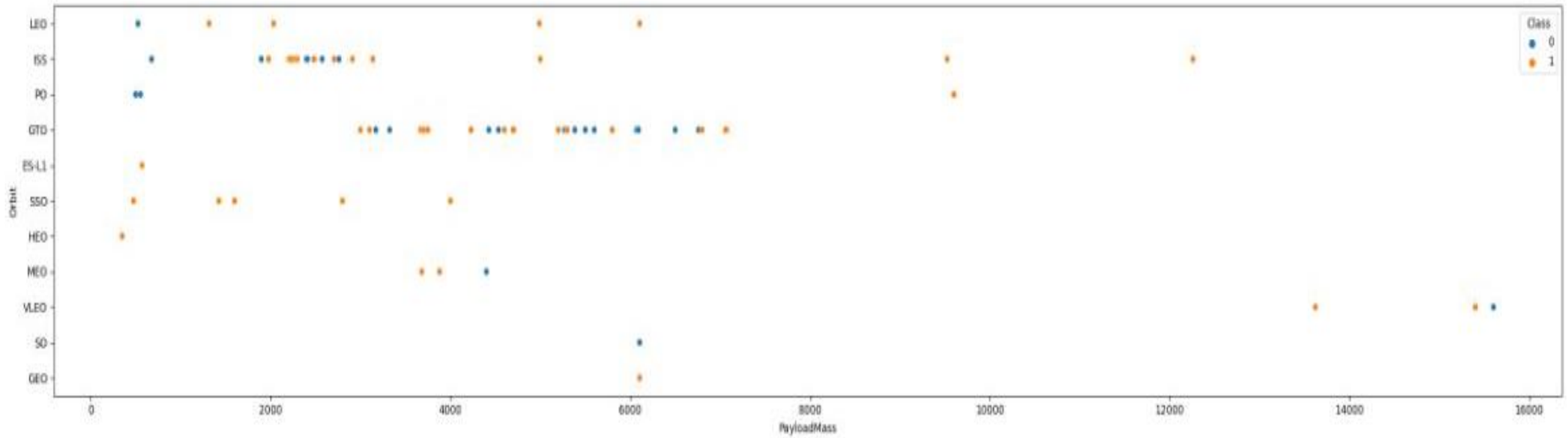


The success increases with the increase in flight number in case of LEO orbit.

Payload vs. Orbit Type

Show a scatter point of payload vs. orbit type

```
sns.scatterplot(x='PayloadMass', y='Orbit', data=df, hue='Class')
plt.show()
```

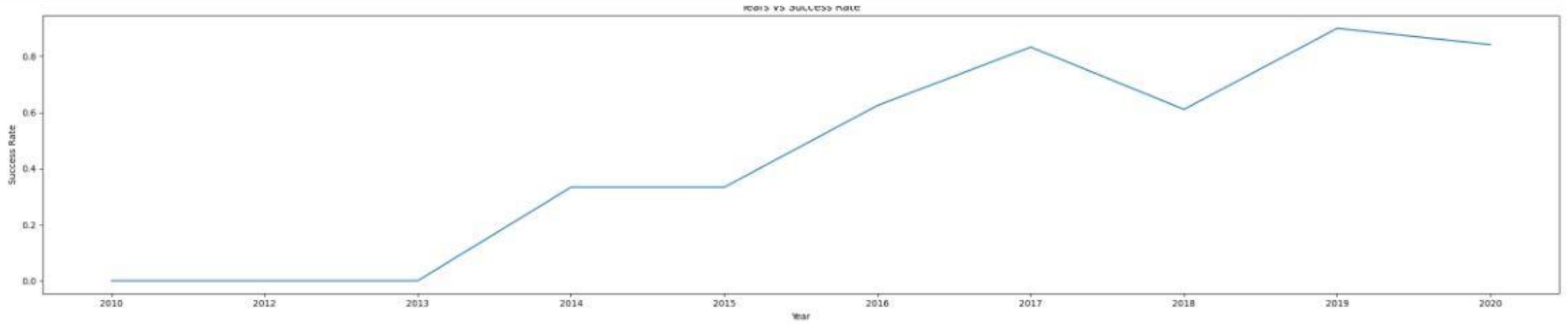


We observe that any launch in SSO orbit be it any payload has resulted in a success, so competing with SPACEX for the launches in this orbit would be highly inappropriate.

Launch Success Yearly Trend

Line chart of yearly average success rate

```
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
average_by_year = df.groupby(by="Date").mean()
average_by_year.reset_index(inplace=True)
plt.plot(average_by_year['Date'],average_by_year['Class'])
plt.xlabel('Year')
plt.ylabel('Success Rate')
plt.title('Years vs Success Rate')
plt.show()
```



An increasing trend can be observed in the success rate since 2013 with some dip after year 2017 and 2019.

All Launch Site Names

Find the names of the unique launch sites

```
%%sql
select unique(launch_site)
from spacex
limit 5;
```

```
 * ibm_db_sa://ntx17291:***
Done.
```

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

The ‘Unique’ command in SQL is used to choose the unique Entries in the column and list them in a tabular form.

Launch Site Names Begin with 'CCA'

Find 5 records where launch sites begin with `CCA`

```
%%sql
select * from spacex
where launch_site like 'CCA%'
limit 5;
```

```
 * ibm_db_sa://ntx17291:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb
Done.
```

DATE	time_utc	booster_version	launch_site	payload	payload_mass_kg_	orbit	customer	mission_outcome	landing_outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	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	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Using the 'Like' command in SQL
We can find the sites which start with a Particular alphabet as in our case CCA.

Total Payload Mass

Calculate the total payload carried by boosters from NASA

```
%%sql
select sum(payload_mass_kg_) as total_payload_mass
from spacex
where customer = 'NASA (CRS)';
```

```
 * ibm_db_sa://ntx17291:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c
Done.
```

total_payload_mass
45596

We use 'Where' clause to filter out the needed data.

Average Payload Mass by F9 v1.1

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

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

```
%%sql
select avg(payload_mass_kg_) as avg_payload_mass
from spacex
where booster_version = 'F9 v1.1';
```

```
 * ibm_db_sa://ntx17291:***@ea286ace-86c7-4d5b-8580-3fbfa46
Done.
```

avg_payload_mass
2928

The 'AVG' function and the 'where' clause Together to get average payload mass.

First Successful Ground Landing Date

Find the dates of the first successful landing outcome on ground pad

```
%%sql
select min(DATE)
from spacex
where landing_outcome = 'Success (ground pad)';
```

```
 * ibm_db_sa://ntx17291:***@ea286ace-86c7-4d5b-8
Done.
```

1
2015-12-22

'MIN' function is used to choose the Minimum date using the 'where' clause For the outcome as Success (ground pad).

Successful Drone Ship Landing with Payload between 4000 and 6000

List the names of boosters which have successfully landed on drone ship and had 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

```
%%sql
select booster_version
from spacex
where landing_outcome = 'Success (drone ship)' and payload_mass_kg_ between 4000 and 6000;
```

```
 * ibm_db_sa://ntx17291:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases
Done.
```

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

'Where' clause combined with the 'Between' clause is used to filter Out the required result.

Total Number of Successful and Failure Mission Outcomes

Calculate the total number of successful and failure mission outcomes

List the total number of successful and failure mission outcomes ¶

```
%%sql
select unique(mission_outcome), count(*) as total_number
from spacex
group by mission_outcome;
```

```
 * ibm_db_sa://ntx17291:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08
Done.
```

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

We use 'Unique' function to count
The different types of outcomes
As success and failure and then
Group-by them the mission outcome
To get the total numbers.

Boosters Carried Maximum Payload

List the names of the booster which have carried the maximum payload mass

```
%%sql
select booster_version,payload_mass_kg_
from spacex
where payload_mass_kg_ = (select max(payload_mass_kg_) from spacex)
limit 5;
```

```
 * ibm_db_sa://ntx17291:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2i
Done.
```

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

We have used a 'sub-query' in the 'where' Clause to get the expected results.

2015 Launch Records

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

```
%%sql
select booster_version, launch_site, landing_outcome
from spacex
where date like '2015%' and landing_outcome = 'Failure (drone ship)';
```

```
 * ibm_db_sa://ntx17291:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2i
Done.
```

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

'Where' clause as the outline we Have used 'and' to add another condition To filter out the data with failure as the Landing outcome in the drone ship.

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

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

```
%%sql
select landing_outcome, count(*) as total
from spacex
where DATE between '2010-06-04' and '2017-03-20'
group by landing_outcome
order by total desc;
```

```
 * ibm_db_sa://ntx17291:***@ea286ace-86c7-4d5b-8580-3fbfa
Done.
```

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

A series of methods including

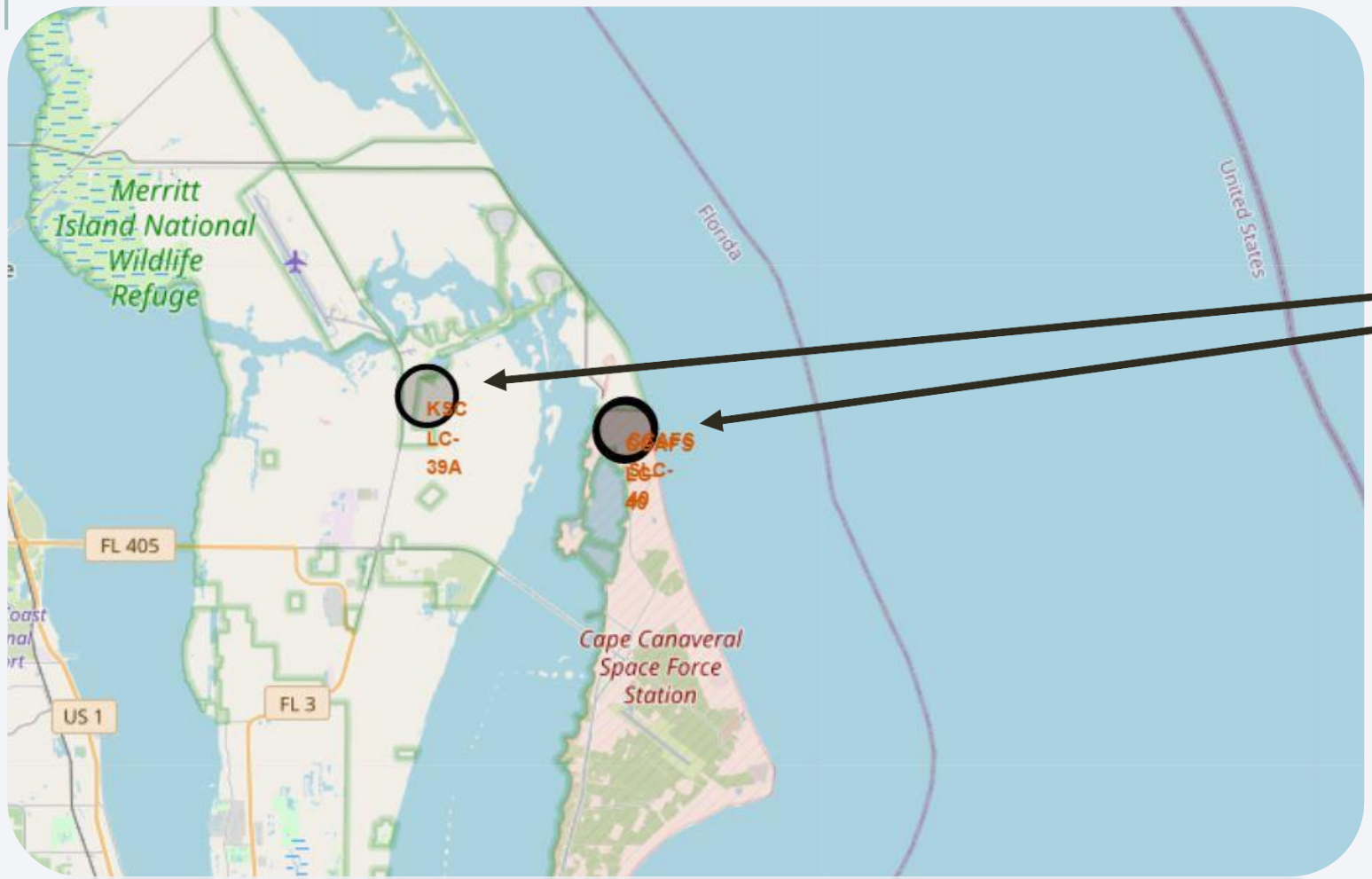
1) **Where clause**
2) **Group by to add up**
3) **Order by to rank them in descending order.**

Used together to filter out the expected results.

Section 3

Launch Sites Proximities Analysis

Folium map with circle marker at the launch sites

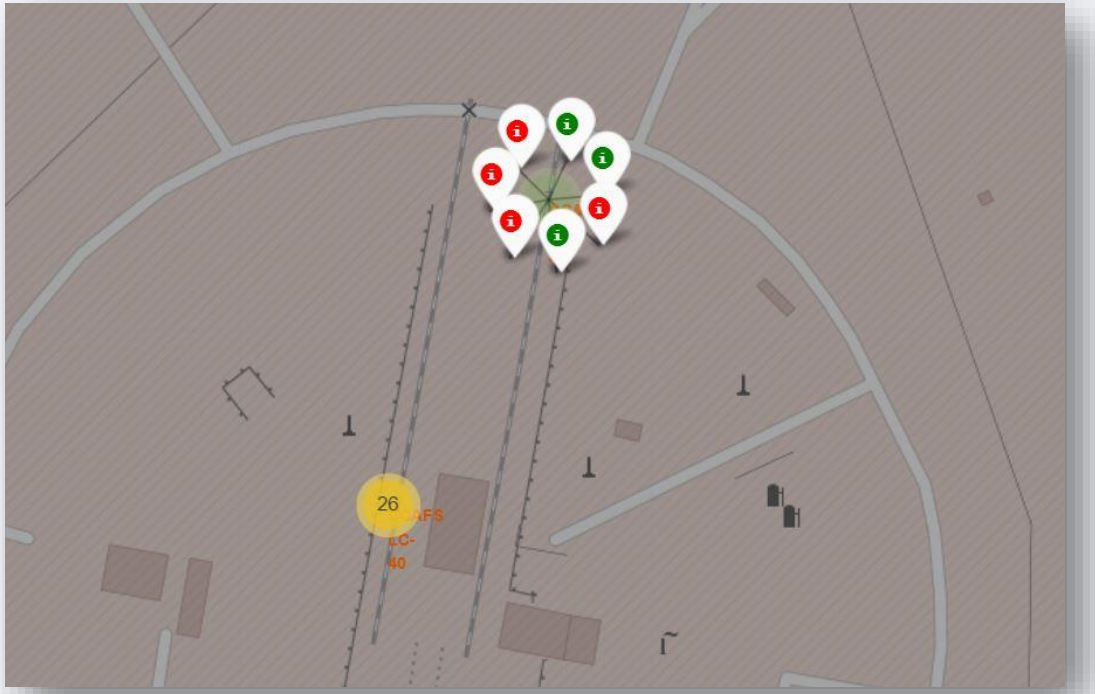


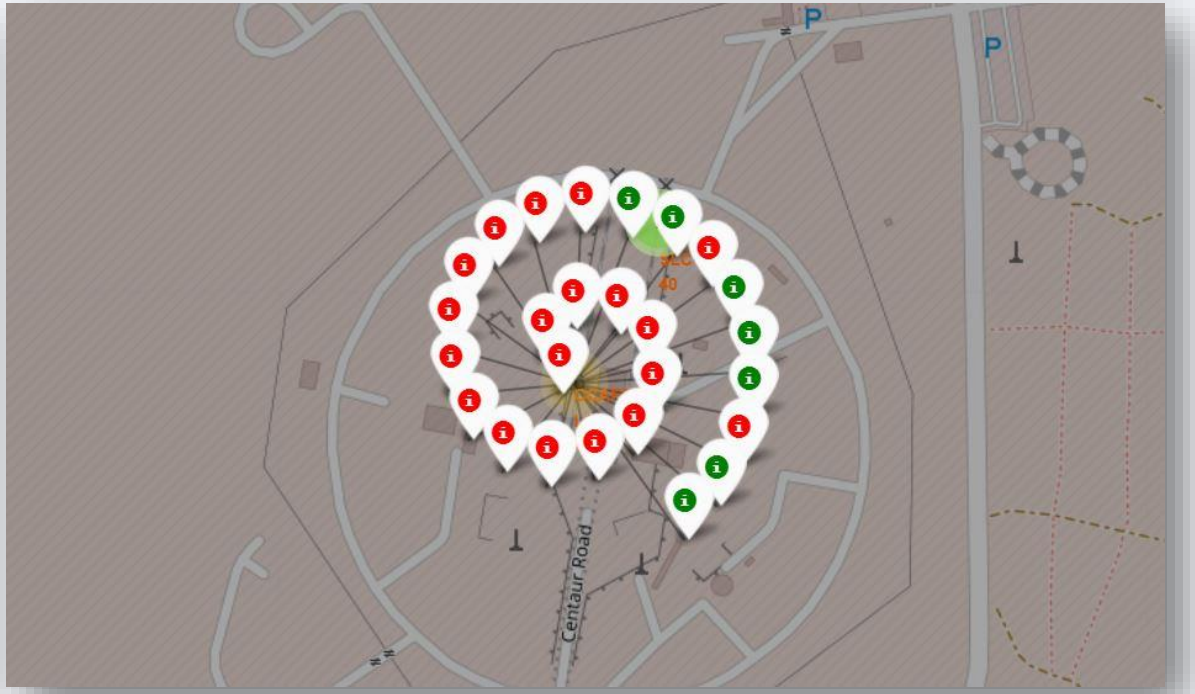
LAUNCH SITES

Launch Sites have been depicted by the circler marker on the global map.

Outcomes with different colors on map as per launch site

Launch site outcomes in different color on MAP.

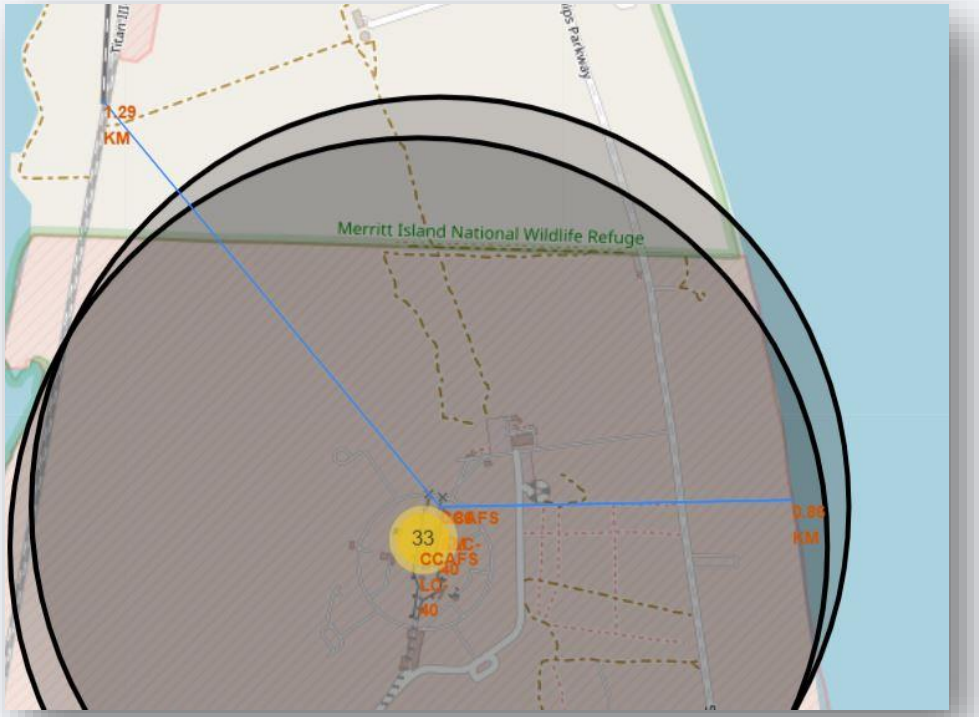




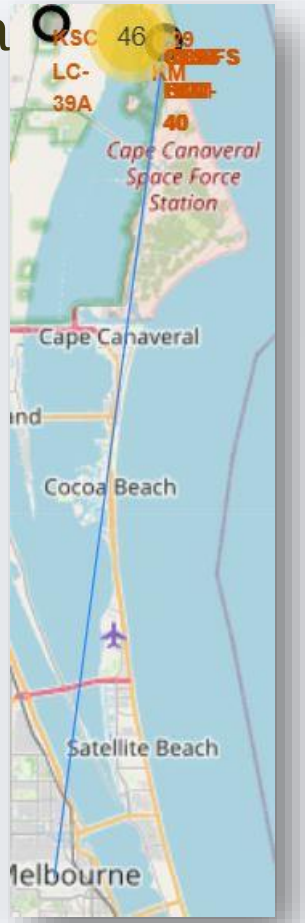
The different outcomes were shown for the launch sites where the green color pop up marker shows the success of the launch and the red one depicts the failure of the outcome.

Map with the proximities:

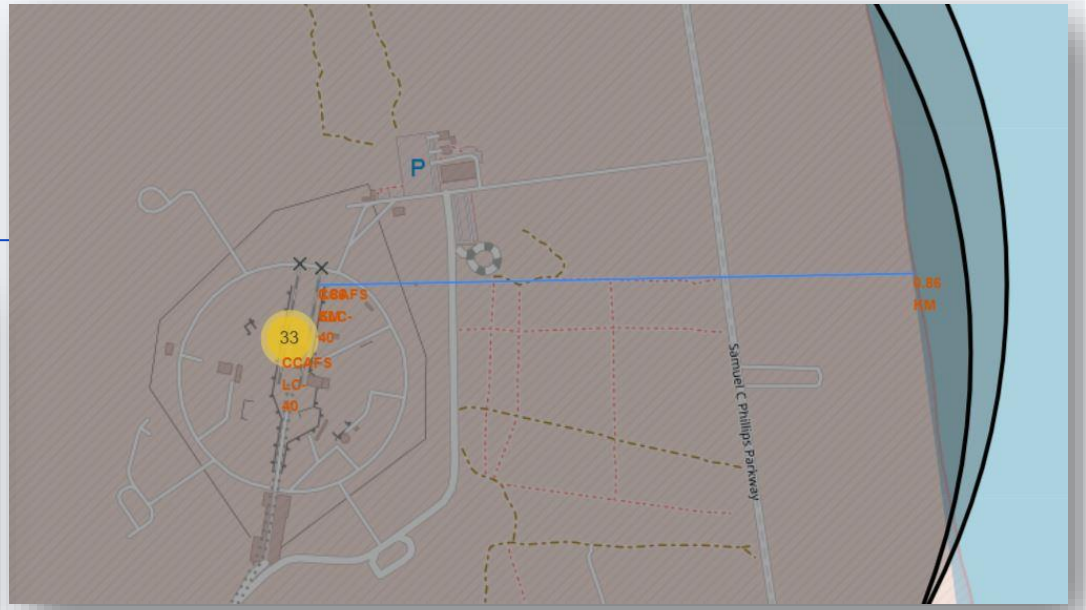
Map with proximities with the highwa railway, city and ocean.



Proximity with ocean and Highway



Proximity with
City ‘Melbourne’

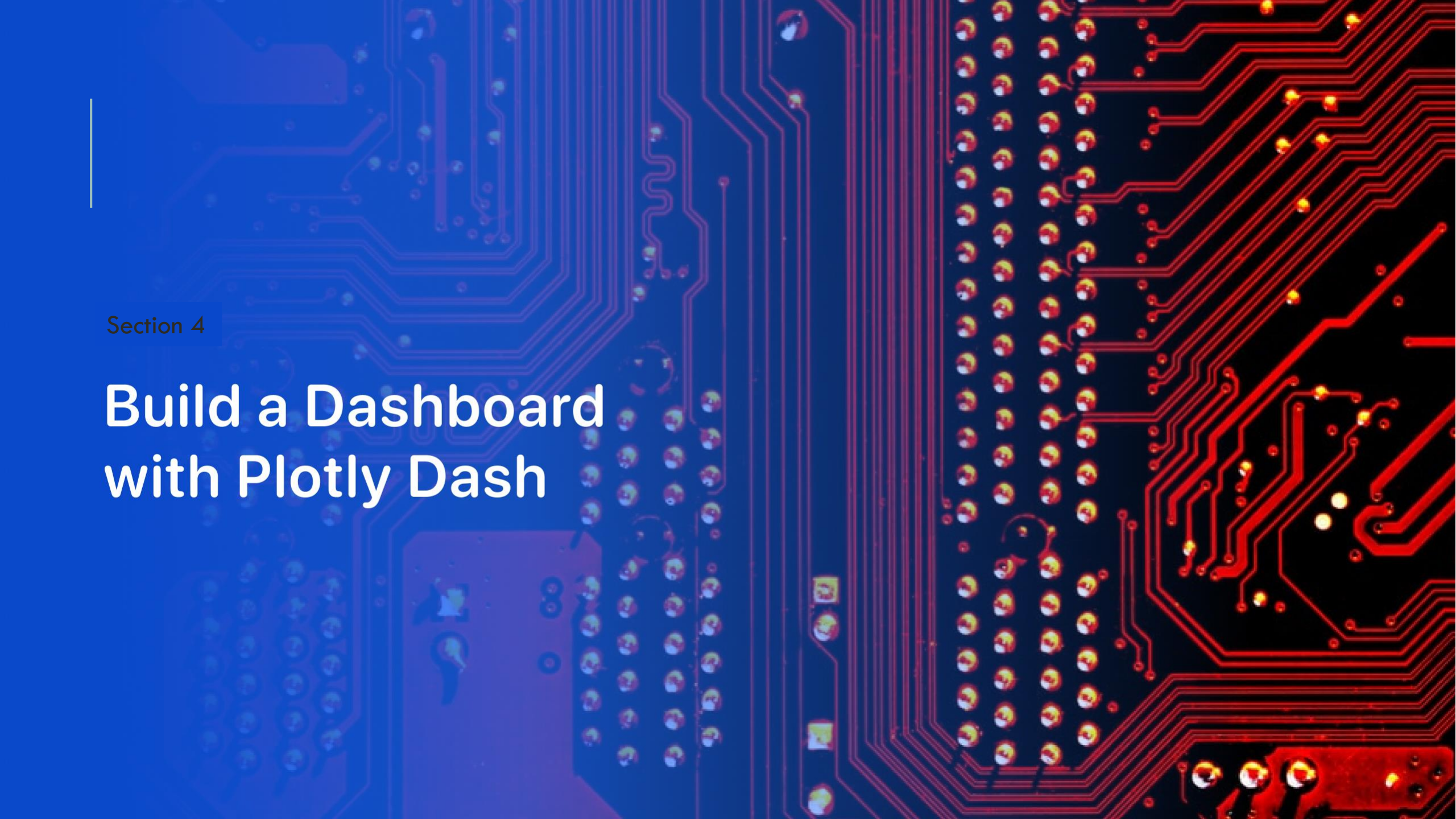


Proximity with the Ocean

Proximity with the Railway, Highway and Ocean

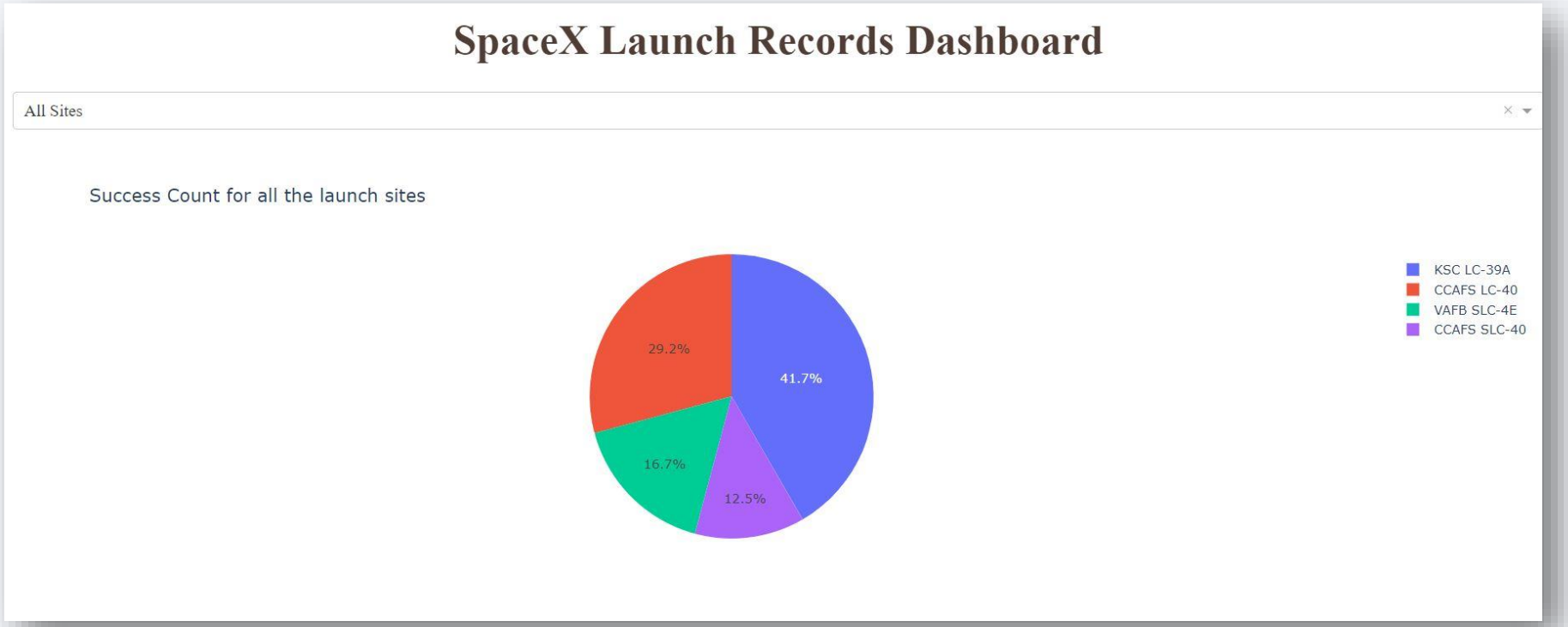
Section 4

Build a Dashboard with Plotly Dash



All sites success ratio

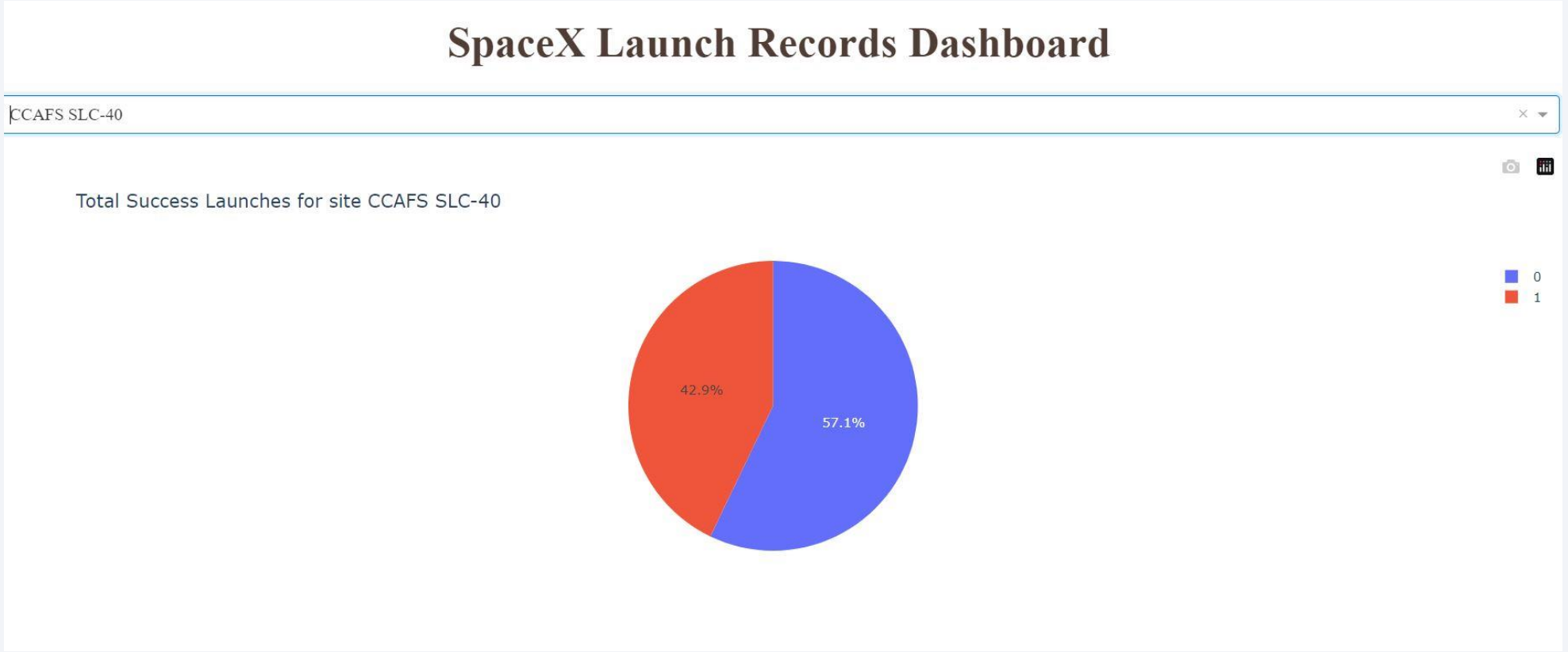
All Sites success ratio in comparison to each other



The site KSC LC-39A has the highest success share with 41.7 percent and CCAFS 5LC-40 with the lowest number of success.

Highest Success Rate

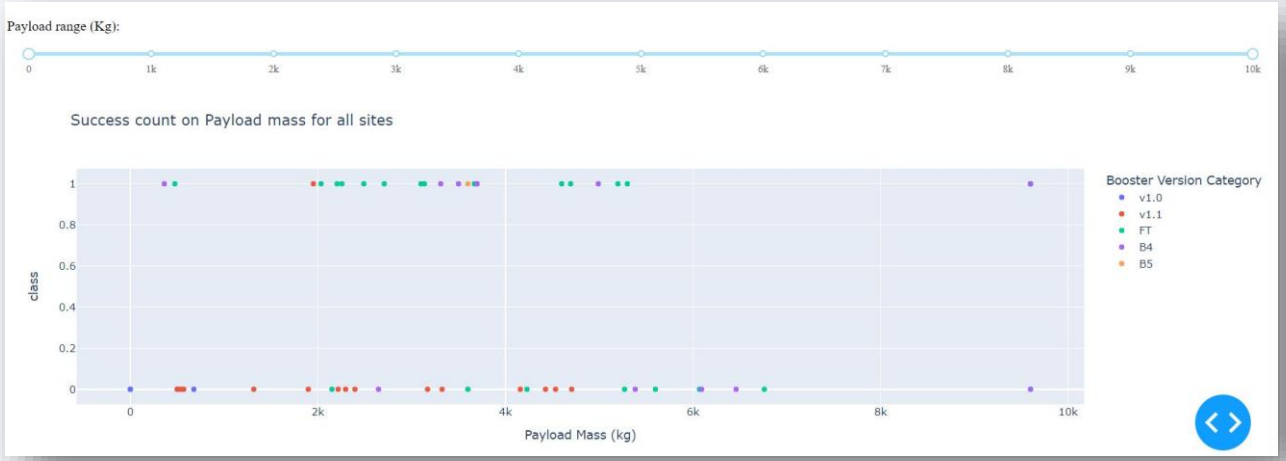
Highest success rate Pie Chart

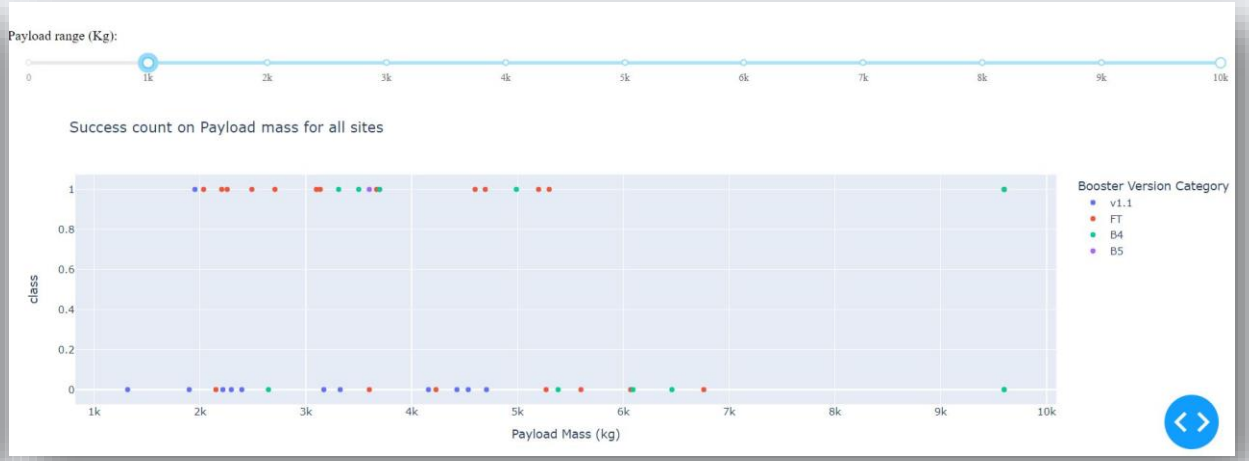


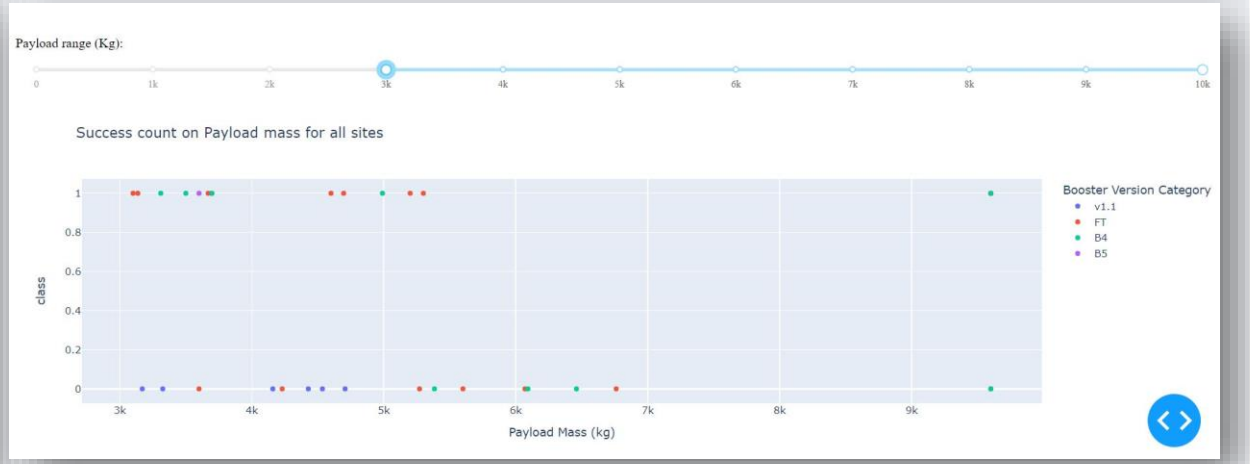
CCAFS SLC-40 has the highest success rate with a ratio of approx. 57:43 of success:failure outcomes.

Scatter Plot with different payloads

Different scatter plots with payload range modified.

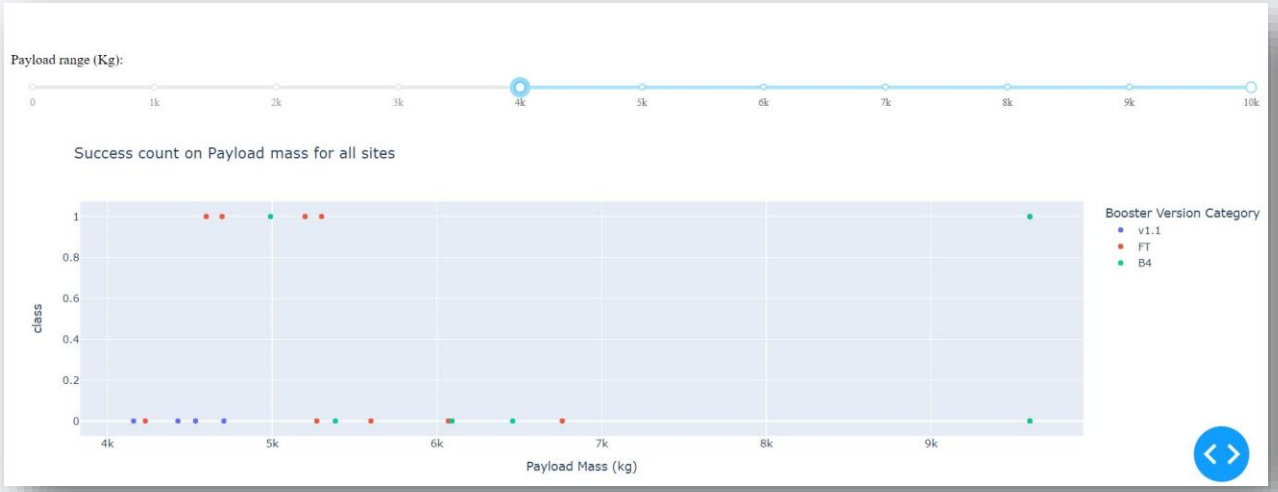


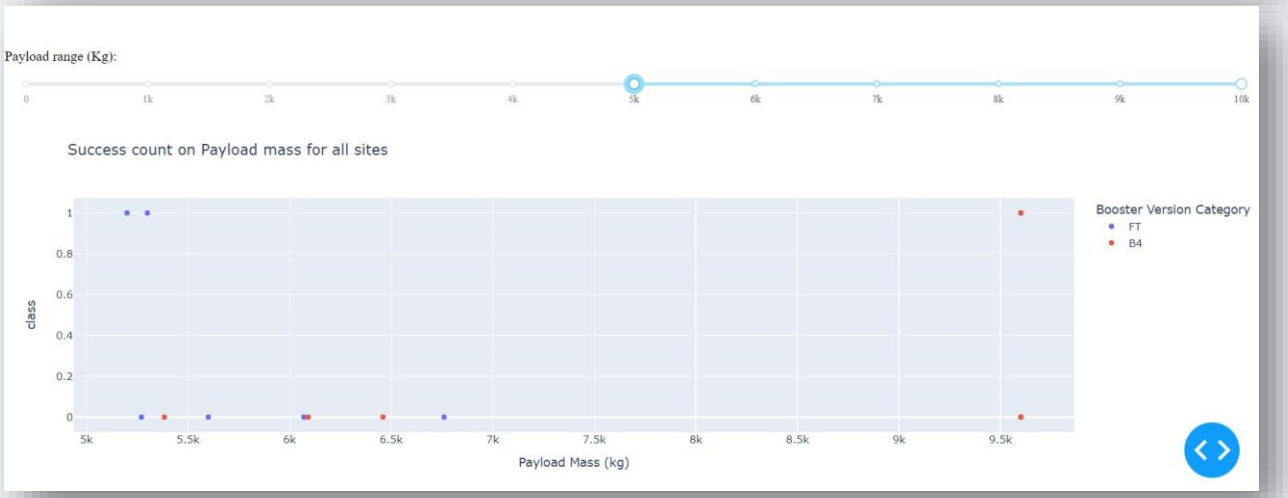




Scatter Plot with different payloads

Different scatter plots with payload range modified.





Payloads at the higher side have got a good success rate but that cannot be taken as the only deciding factor as the frequency of the data at higher payloads are limited.

Section 5

Predictive Analysis (Classification)

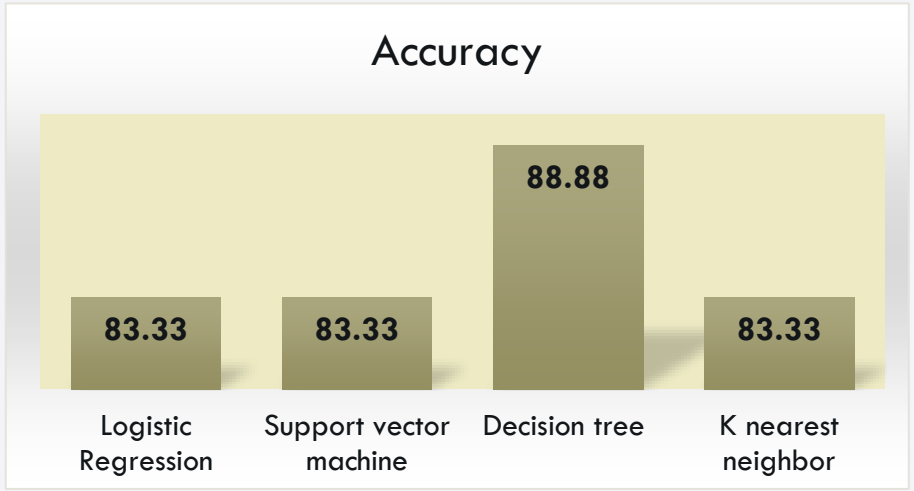
Classification Accurac

Confusion Matrix

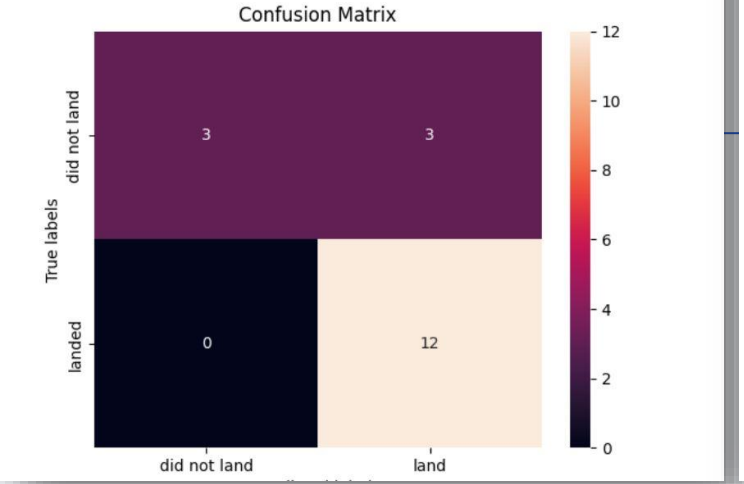
Accuracy Results

```
print('Accuracy for Logistics Regression method:', logreg_cv.score(X_test, Y_test))
print( 'Accuracy for Support Vector Machine method:', svm_cv.score(X_test, Y_test))
print('Accuracy for Decision tree method:', tree_cv.score(X_test, Y_test))
print('Accuracy for K nearsdt neighbors method:', knn_cv.score(X_test, Y_test))
```

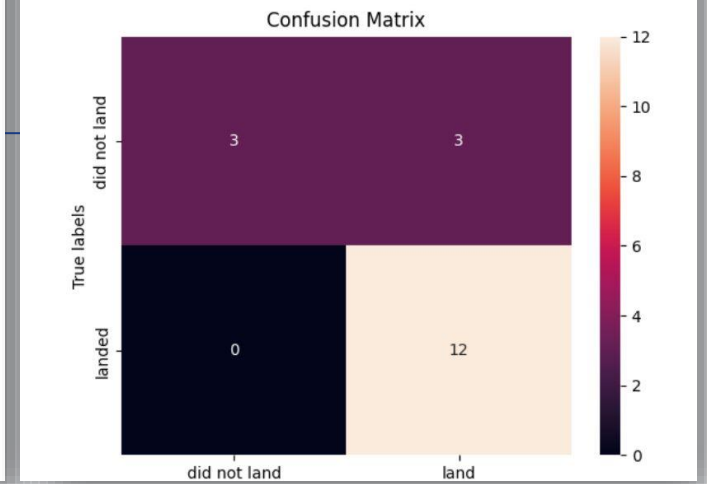
```
Accuracy for Logistics Regression method: 0.8333333333333334
Accuracy for Support Vector Machine method: 0.8333333333333334
Accuracy for Decision tree method: 0.8888888888888888
Accuracy for K nearsdt neighbors method: 0.8333333333333334
```



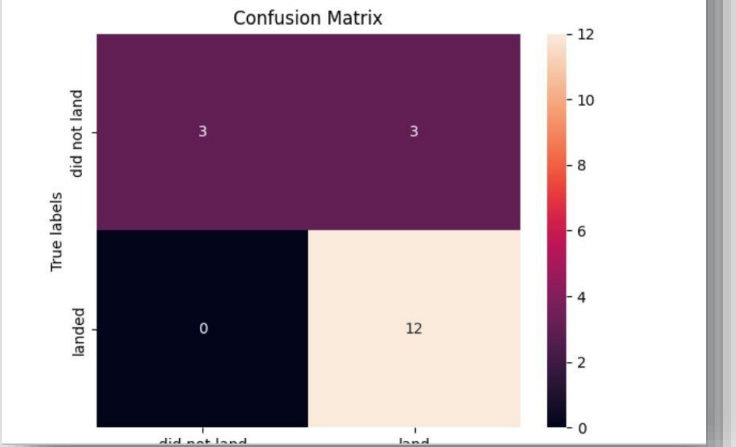

```
yhat=svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



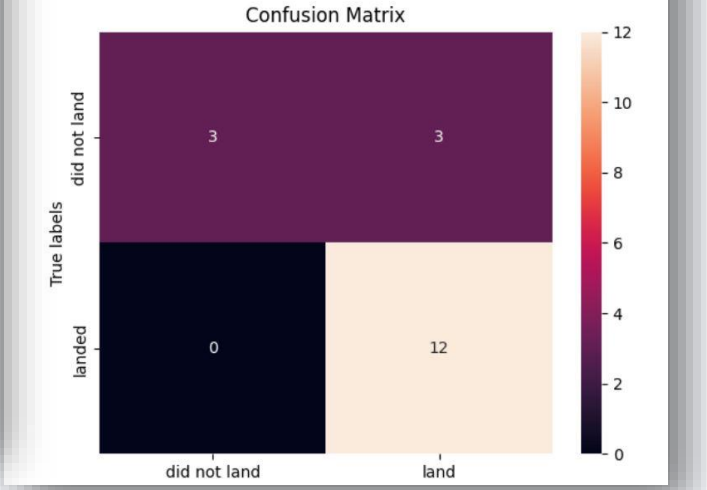

```
yhat = svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```




```
yhat=logreg_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



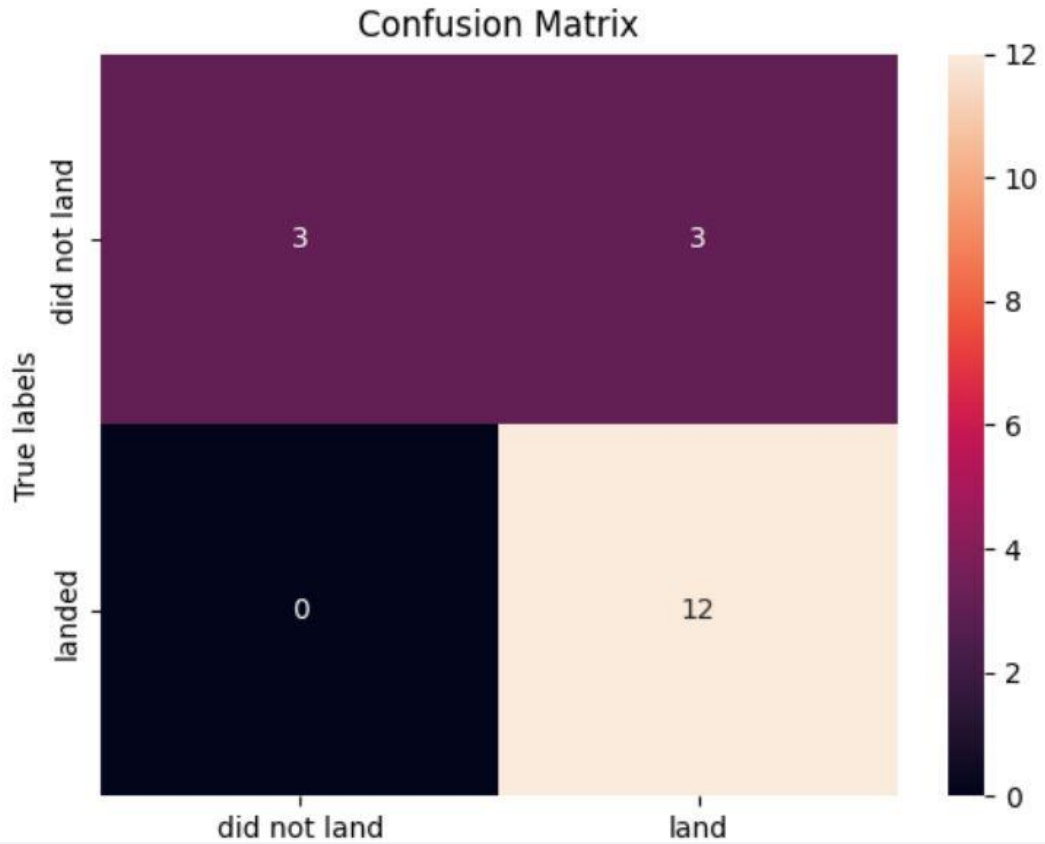

```
yhat = knn_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



Confusion Matrix

Confusion matrix of the best performing model with an explanation

```
yhat = svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



The decision tree model is the one with most accuracy result.
In this decision tree confusion matrix the total number of result in the test data where the outcome was landed is 12 and the predicted result using decision tree model is also 12 which is 100% accurate and the result showing did not land was total 6 out of which the model predicted 3 correct.

Conclusions

- Orbit ES-L1, GEO, HEO, SSO has highest success rate.
- By seeing the line graph of success rate with the passing year we can say that the success rate of SPACEX is increasing year by year and they might get the required results soon.
- KSC LC 39A had the most successful launches but increasing payloads have negative impact on its success
- Decision Tree algorithm is the best machine learning algorithm to predict the success of the launch and land.

Thank you!