DATA SCIENCE CAPSTONE PRESENTATION

By: Kieran Chian

INTRODUCTION

- Project background and context

SpaceX is the most successful company of the commercial space age, making space travel affordable. The company 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. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch. Based on public information and machine learning models, we are going to predict if SpaceX will reuse the first stage.

- Questions to be answered

How do variables such as payload mass, launch site, number of flights, and orbits affect the success of the first stage landing?

Does the rate of successful landings increase over the years?

What is the best algorithm that can be used for binary classification in this case?

EXECUTIVE SUMMARY

GITHUB CODE

- Data Analysis Methodology
- Data Collection and Wrangling
- Exploratory Data Analysis with SQL
- Exploratory Data Analysis with Pandas and Seaborn
- EDA and interactive visual analytics with Folium
- Building Dashboard with Plotly Dash dashboard
- Predictive Analysis (Classification)
- Result and Conclusion

Data Collection and Wrangling

Web scrap Falcon 9 launch records with `BeautifulSoup`:

* Extract a Falcon 9 launch records HTML table from Wikipedia
* Parse the table and convert it into a Pandas data frame

Perform exploratory Data Analysis and determine
Training Labels

* Exploratory Data Analysis
* Determine Training Labels

Data Collection and Wrangling

Web scrap Falcon 9 launch records with `BeautifulSoup`:

* Extract a Falcon 9 launch records HTML table from Wikipedia
* Parse the table and convert it into a Pandas data frame

Perform exploratory Data Analysis and determine Training Labels

* Exploratory Data Analysis
* Determine Training Labels

▪ EDA and interactive visual analytics

Exploratory Data Analysis with SQL

Task 1

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

```
[6]: %sql select distinct(Launch_Site) from SPACEXTBL
```

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

[6]:

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

Task 2

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

```
[7]: %sql select * from SPACEXTBL where Launch_Site like 'CCA%' limit 5
```

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

[7]:

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

Task 3

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

[8]:
```
%sql select sum(PAYLOAD_MASS__KG_) from SPACEXTBL where Customer = 'NASA (CRS)'
```

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

[8]:

sum(PAYLOAD_MASS__KG_)
45596

Task 4

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

[9]:
```
%sql select avg(PAYLOAD_MASS__KG_) as average from SPACEXTBL where Booster_Version = 'F9 v1.1'
```

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

[9]:

average
2928.4

Task 5

List the date when the first succesful landing outcome in ground pad was acheived.

Hint:Use min function

[50]:
```
%sql select MIN(Date) from SPACEXTBL where `Landing _Outcome` = 'Success (ground pad)'
```

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

Task 5

List the date when the first succesful landing outcome in ground pad was acheived.

Hint:Use min function

[50]:
```
%sql select MIN(Date) from SPACEXTBL where `Landing _Outcome` = 'Success (ground pad)'
```

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

[50]:

MIN(Date)
01-05-2017

Task 6

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000 ¶

[44]:
```
%sql select Booster_Version,PAYLOAD_MASS__KG_ from SPACEXTBL where `Landing _Outcome` = 'Success (drone ship)' and PAYLOAD_MASS__KG_ <=6000 and PAYLOAD_MASS__KG_ >=4000
```

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

[44]:

Booster_Version	PAYLOAD_MASS__KG_
F9 FT B1022	4696
F9 FT B1026	4600
F9 FT B1021.2	5300
F9 FT B1031.2	5200

Task 7

List the total number of successful and failure mission outcomes

[12]:
```
%sql select Mission_Outcome,count(Mission_Outcome) from SPACEXTBL group by Mission_Outcome
```

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

[12]:

Mission_Outcome	count(Mission_Outcome)
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Task 8

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

[13]:
```
%sql select Booster_Version from SPACEXTBL where (PAYLOAD_MASS__KG_=(select MAX(PAYLOAD_MASS__KG_) from SPACEXTBL))
```

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

[13]:

Booster_Version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5

Task 9

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

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

[14]:
```
%sql select substr(Date,4,2), `Landing _Outcome`, Booster_Version, Launch_Site from SPACEXTBL where substr(Date,7,4)='2015'
```

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

[14]:

substr(Date,4,2)	Landing _Outcome	Booster_Version	Launch_Site
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
02	Controlled (ocean)	F9 v1.1 B1013	CCAFS LC-40
03	No attempt	F9 v1.1 B1014	CCAFS LC-40
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40
04	No attempt	F9 v1.1 B1016	CCAFS LC-40
06	Precluded (drone ship)	F9 v1.1 B1018	CCAFS LC-40
12	Success (ground pad)	F9 FT B1019	CCAFS LC-40

Task 10

Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

[35]:
```
%sql select `Landing _Outcome`, count(`Landing _Outcome`) from (select * from SPACEXTBL where `Landing _Outcome` like '%Success%') group by `Landing _Outcome`
```

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

[35]:

Landing _Outcome	count(`Landing _Outcome`)
Success	38
Success (drone ship)	14
Success (ground pad)	9

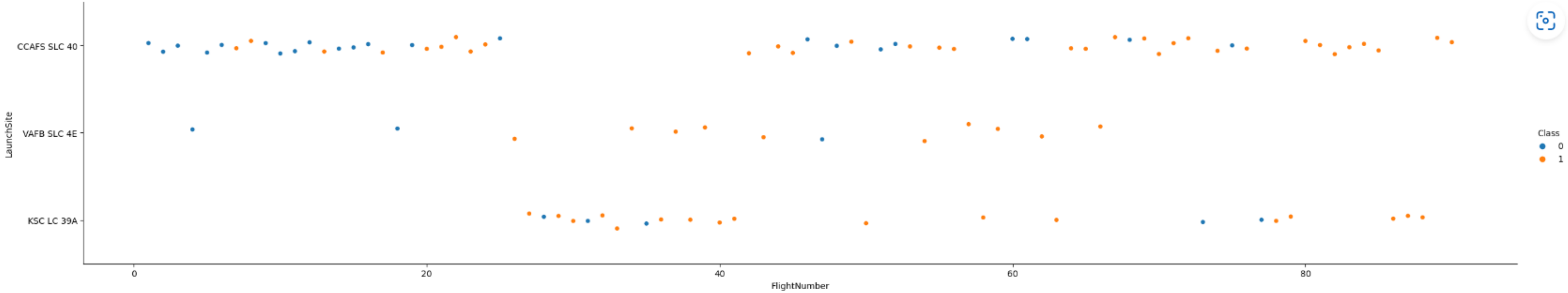
Exploratory Data Analysis with Pandas and Seaborn

TASK 1: Visualize the relationship between Flight Number and Launch Site

Use the function `catplot` to plot `FlightNumber` vs `LaunchSite`, set the parameter `x` parameter to `FlightNumber`,set the `y` to `Launch Site` and set the parameter `hue` to `'class'`

[4]:
```
# 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(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
```

[4]: <seaborn.axisgrid.FacetGrid at 0x25bd21d2310>



Now try to explain the patterns you found in the Flight Number vs. Launch Site scatter point plots.

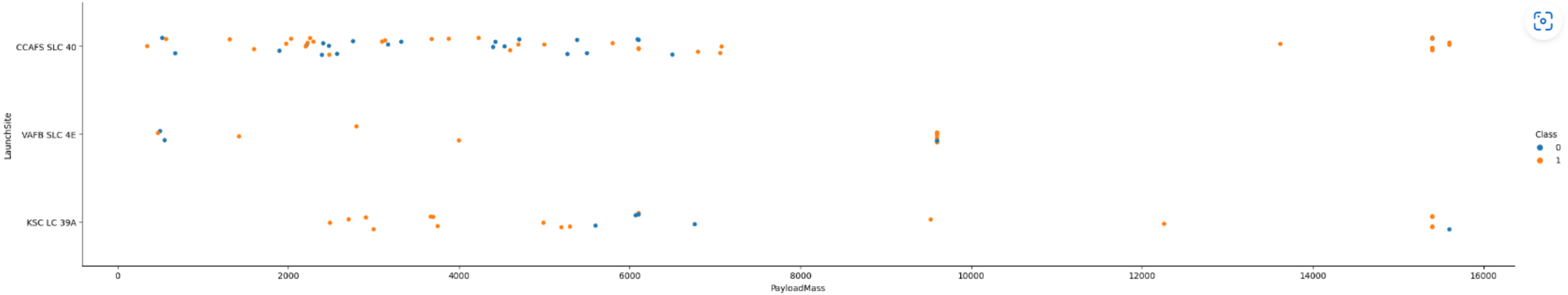
TASK 2: Visualize the relationship between Payload and Launch Site

We also want to observe if there is any relationship between launch sites and their payload mass.

[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 value

sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 5)
```

[7]: <seaborn.axisgrid.FacetGrid at 0x25bd2e1ecd0>



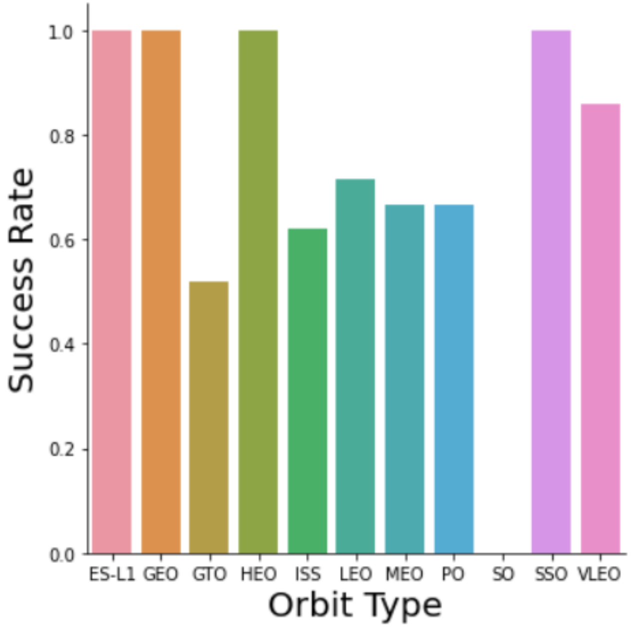
Now if you observe Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

TASK 3: Visualize the relationship between success rate of each orbit type

Next, we want to visually check if there are any relationship between success rate and orbit type.

Let's create a `bar chart` for the sucess rate of each orbit

[6]:
```
# HINT use groupby method on Orbit column and get the mean of Class column
sns.catplot(x= 'Orbit', y = 'Class', data = df.groupby('Orbit')['Class'].mean().reset_index(), kind = 'bar')
plt.xlabel('Orbit Type',fontsize=20)
plt.ylabel('Success Rate',fontsize=20)
plt.show()
```

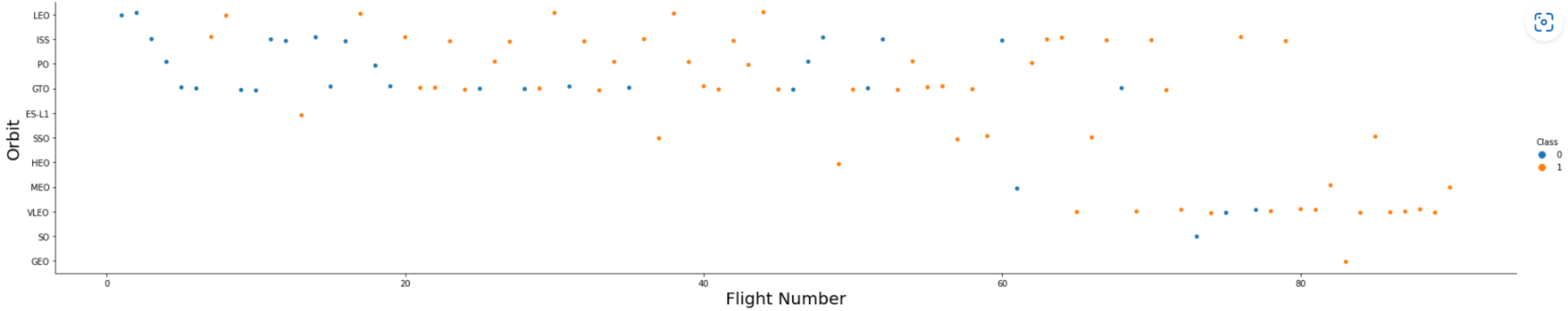


TASK 4: Visualize the relationship between FlightNumber and Orbit type

For each orbit, we want to see if there is any relationship between FlightNumber and Orbit type.

[7]:

```
# 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, aspect = 5)
plt.xlabel('Flight Number', fontsize = 20)
plt.ylabel('Orbit', fontsize = 20)
plt.show()
```

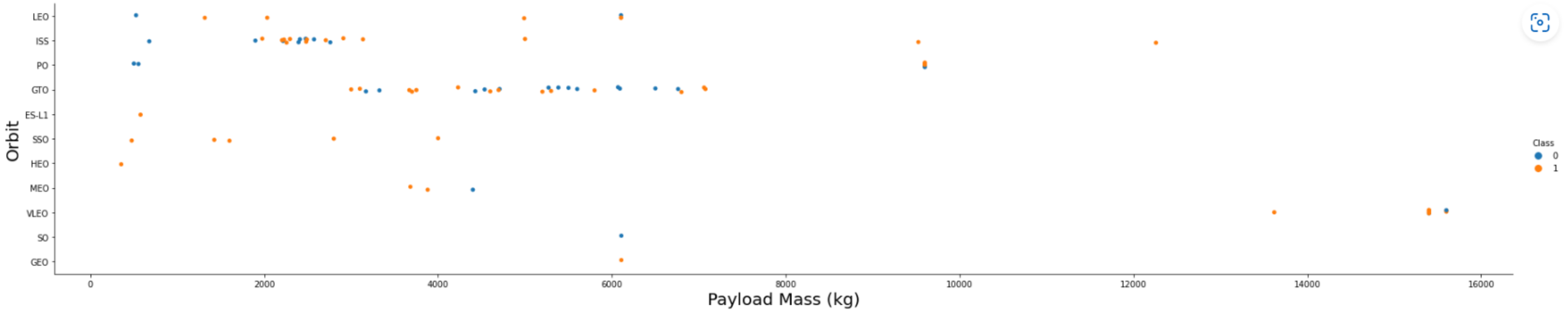


You should see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

TASK 5: Visualize the relationship between Payload and Orbit type

Similarly, we can plot the Payload vs. Orbit scatter point charts to reveal the relationship between Payload and Orbit type

[8]:
```
# 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, aspect = 5)
plt.xlabel('Payload Mass (kg)', fontsize = 20)
plt.ylabel('Orbit', fontsize = 20)
plt.show()
```



You should observe that Heavy payloads have a negative influence on GTO orbits and positive on GTO and Polar LEO (ISS) orbits.

TASK 6: Visualize the launch success yearly trend

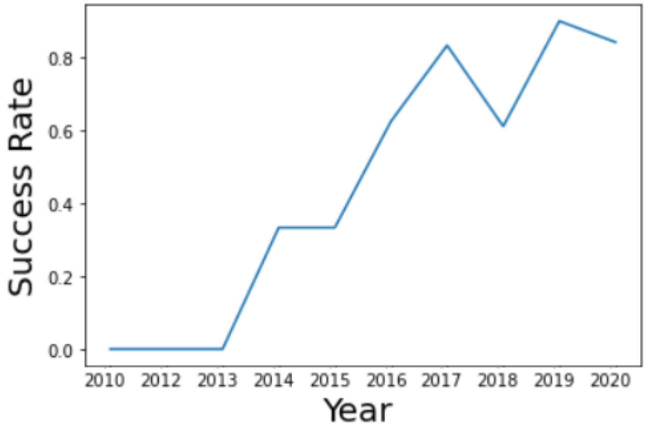
You can plot a line chart with x axis to be `Year` and y axis to be average success rate, to get the average launch success trend.

The function will help you get the year from the date:

[9]:
```
# A function to Extract years from the date
year=[]
def Extract_year(date):
    for i in df["Date"]:
        year.append(i.split("-")[0])
    return year
```

[10]:
```
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
years = df.groupby(Extract_year(df['Date'])).mean()['Class']

sns.lineplot(x = years.index, y = years)
plt.xlabel('Year', fontsize = 20)
plt.ylabel('Success Rate', fontsize = 20)
plt.show()
```



you can observe that the sucess rate since 2013 kept increasing till 2020

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.

[12]:

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

[12]:

	Orbit_ES-L1	Orbit_GEO	Orbit_GTO	Orbit_HEO	Orbit_ISS	Orbit_LEO	Orbit_MEO	Orbit_PO	Orbit_SO	Orbit_SSO	...	Serial_B1048	Serial_B1049	Serial_B1050	Serial_B1051	Serial_B1054	Serial_B1056	Se
0	0	0	0	0	0	1	0	0	0	0	...	0	0	0	0	0	0	
1	0	0	0	0	0	1	0	0	0	0	...	0	0	0	0	0	0	
2	0	0	0	0	1	0	0	0	0	0	...	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	1	0	0	...	0	0	0	0	0	0	
4	0	0	1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	

5 rows × 72 columns

TASK 8: Cast all numeric columns to `float64`

Now that our `features_one_hot` dataframe only contains numbers cast the entire dataframe to variable type `float64`

[13]:
```python
# HINT: use astype function
features_one_hot.astype('float64')
```

[13]:

	Orbit_ES-L1	Orbit_GEO	Orbit_GTO	Orbit_HEO	Orbit_ISS	Orbit_LEO	Orbit_MEO	Orbit_PO	Orbit_SO	Orbit_SSO	...	Serial_B1048	Serial_B1049	Serial_B1050	Serial_B1051	Serial_B1054	Serial_B1056	S
0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
85	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
86	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
87	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	1.0	0.0	0.0	
88	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
89	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	

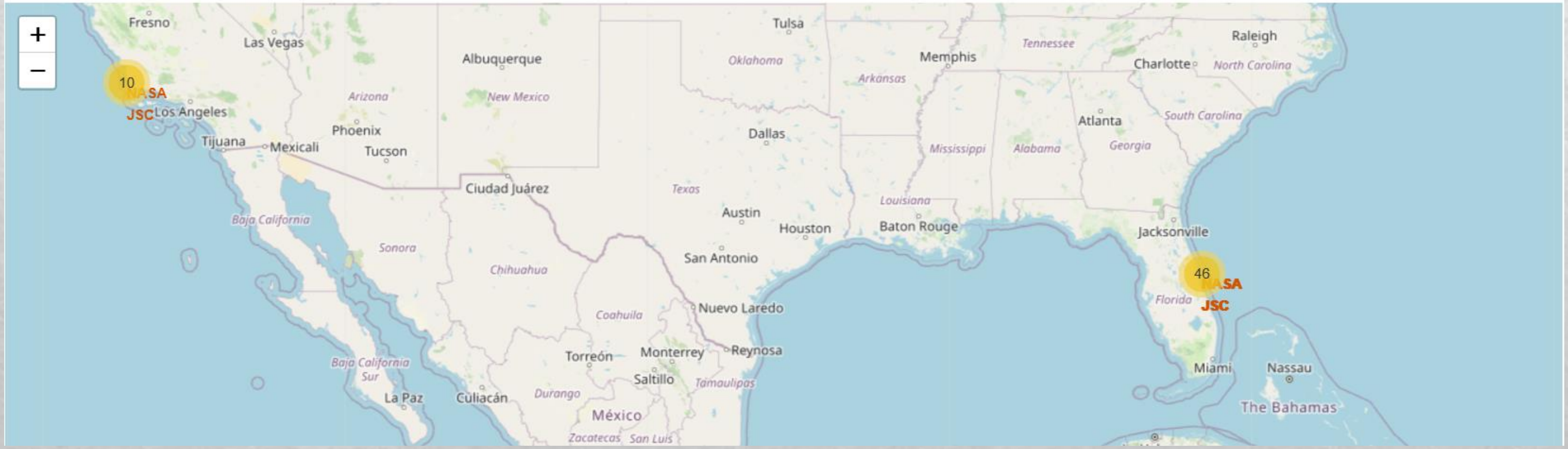
90 rows × 72 columns

▪ Predictive Analysis Methodology

Interactive Visual Analytics with Folium

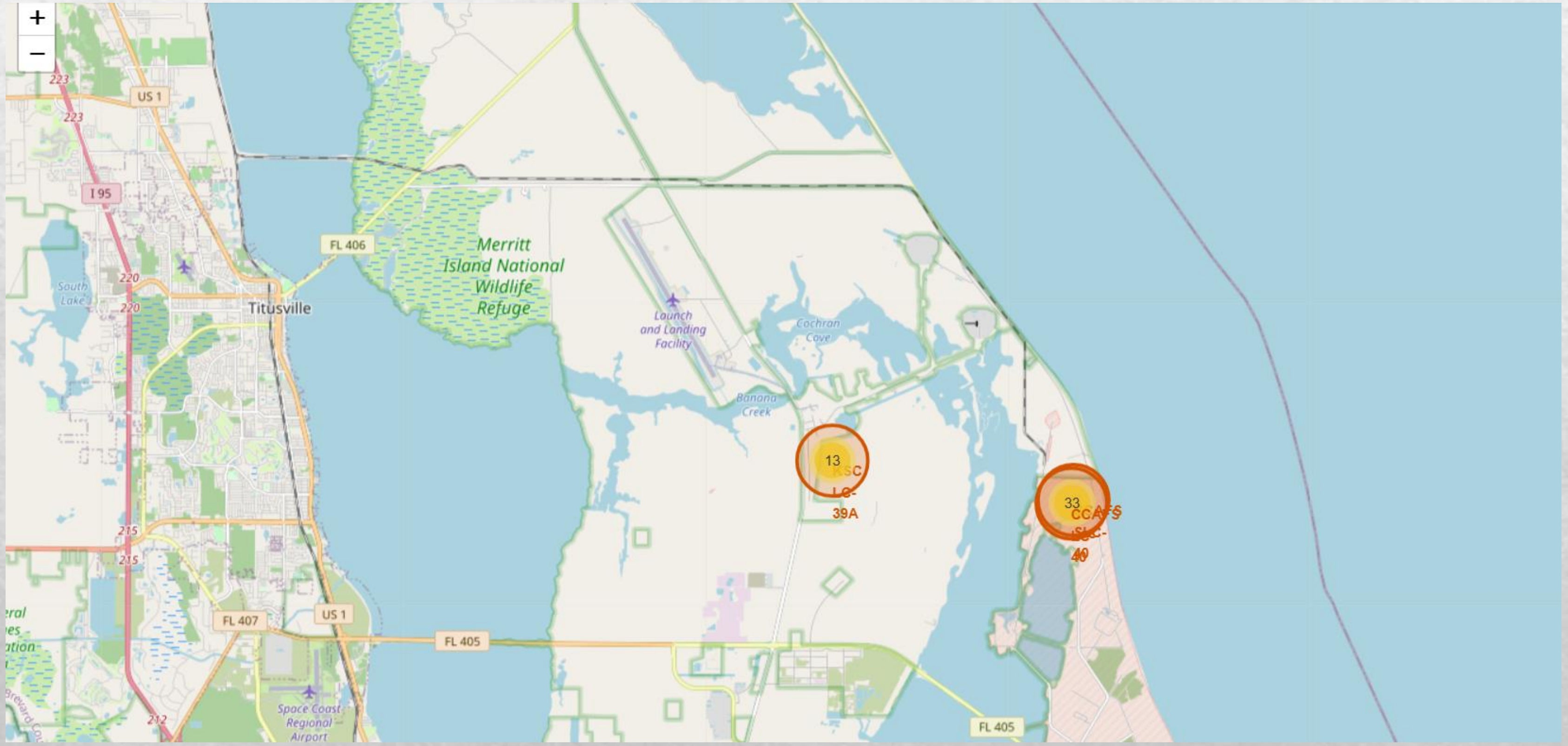
```python
# Add marker_cluster to current site_map
site_map.add_child(marker_cluster)

for i, j in zip(spacex_df.Lat,spacex_df.Long):
    marker = folium.features.CircleMarker(
        [i,j],
        radius=5, # define how big you want the circle markers to be
        color='yellow',
        fill=True,
        fill_color='blue',
        fill_opacity=0.6
    )
    marker_cluster.add_child(marker)
site_map
```

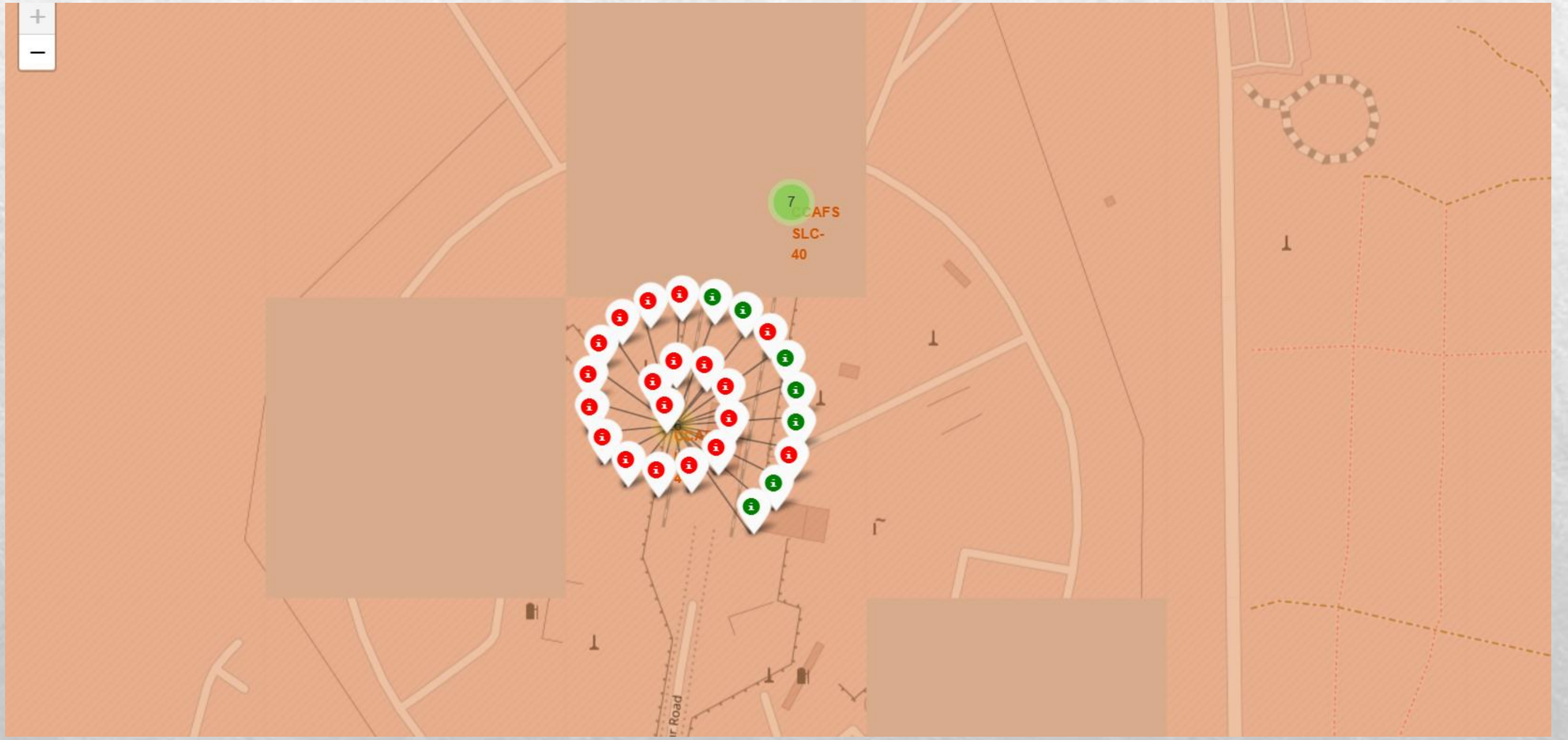


The launch success rate may depend on many factors such as payload mass, orbit type, and so on. It may also depend on the location and proximities of a launch site, i.e., the initial position of rocket trajectories.

- EDA with visualization results



- EDA with visualization results



From the colour-labeled markers we can easily identify which launch sites have relatively high success rates.

- Green Marker = Successful Launch
- Red Marker = Failed Launch

SpaceX Launch Records Dashboard

All Sites

Total Success Launches by Site

Payload range (Kg):

0 1000 2000 3000 4000 5000 6000 7000 8000 9000 10000

▪ Predictive Analysis (Classification)

TASK 1

Create a NumPy array from the column `Class` in `data`, by applying the method `to_numpy()` then assign it to the variable `Y`,make sure the output is a Pandas series (only one bracket df['name of column']).

[5]:
```
Y = pd.Series(data['Class'].to_numpy())
Y.head(10)
```

[5]:
```
0    0
1    0
2    0
3    0
4    0
5    0
6    1
7    1
8    0
9    0
dtype: int64
```

TASK 2

Standardize the data in `X` then reassign it to the variable `X` using the transform provided below.

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

[7]:
```
X=transform.fit_transform(X)
```

TASK 3

Use the function train_test_split to split the data X and Y into training and test data. Set the parameter test_size to 0.2 and random_state to 2. The training data and test data should be assigned to the following labels.

`X_train, X_test, Y_train, Y_test`

[8]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
```

we can see we only have 18 test samples.

[9]:
```
Y_test.shape
```

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

TASK 4

Create a logistic regression object then create a GridSearchCV object `logreg_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

[10]:
```
parameters ={'C':[0.01,0.1,1],
             'penalty':['l2'],
             'solver':['lbfgs']}
```

[11]:
```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
lr=LogisticRegression()
logreg_cv=GridSearchCV(lr, parameters, cv=10)
logreg_cv.fit(X_train, Y_train)
```

TASK 5

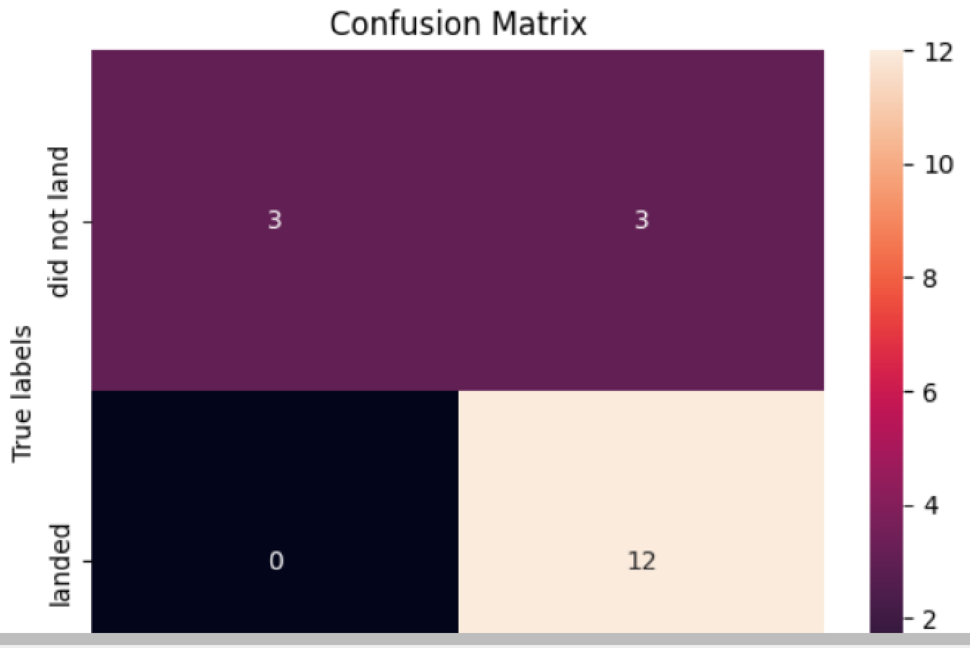
Calculate the accuracy on the test data using the method `score` :

[13]:
```
logreg_accuracy = logreg_cv.score(X_test, Y_test)
logreg_accuracy
```

[13]: 0.8333333333333334

Lets look at the confusion matrix:

[14]:
```
yhat=logreg_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



TASK 6

Create a support vector machine object then create a `GridSearchCV` object `svm_cv` with cv - 10. Fit the object to find the best parameters from the dictionary `parameters`.

[15]:
```
parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
              'C': np.logspace(-3, 3, 5),
              'gamma':np.logspace(-3, 3, 5)}
svm = SVC()
```

[16]:
```
svm_cv = GridSearchCV(svm, parameters, cv=10)
svm_cv.fit(X_train, Y_train)
```

[16]:
```
▸ GridSearchCV
▸ estimator: SVC
    ▸ SVC
```

[17]:
```
print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
print("accuracy :",svm_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
accuracy : 0.8482142857142856
```

TASK 7

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

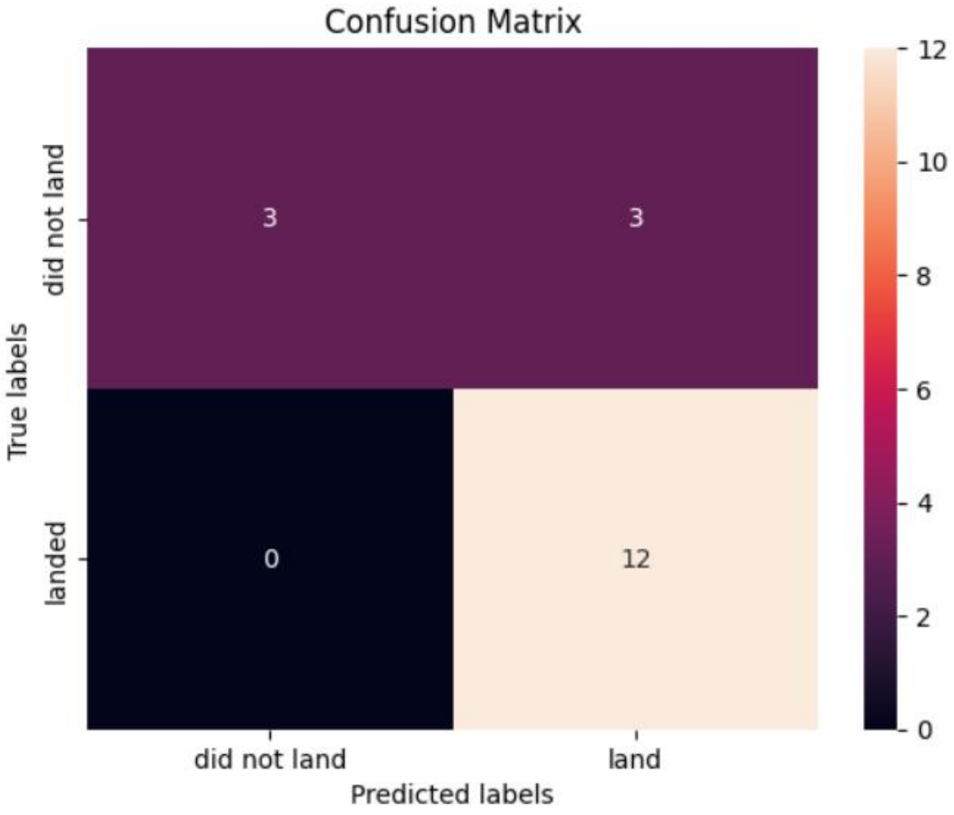
[18]:
```
svm_accuracy = svm_cv.score(X_test, Y_test)
svm_accuracy
```

[18]:
```
0.8333333333333334
```

[18]: 0.8333333333333334

We can plot the confusion matrix

```
[19]: yhat=svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



TASK 8

Create a decision tree classifier object then create a `GridSearchCV` object `tree_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

[20]:
```
parameters = {'criterion': ['gini', 'entropy'],
     'splitter': ['best', 'random'],
     'max_depth': [2*n for n in range(1,10)],
     'max_features': ['auto', 'sqrt'],
     'min_samples_leaf': [1, 2, 4],
     'min_samples_split': [2, 5, 10]}

tree = DecisionTreeClassifier()
```

[21]:
```
tree_cv = GridSearchCV(tree, parameters, cv=10)
tree_cv.fit(X_train, Y_train)
```

•••

[22]:
```
print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
print("accuracy :",tree_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 18, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 2, 'splitter': 'best'}
accuracy : 0.8892857142857145
```

TASK 9

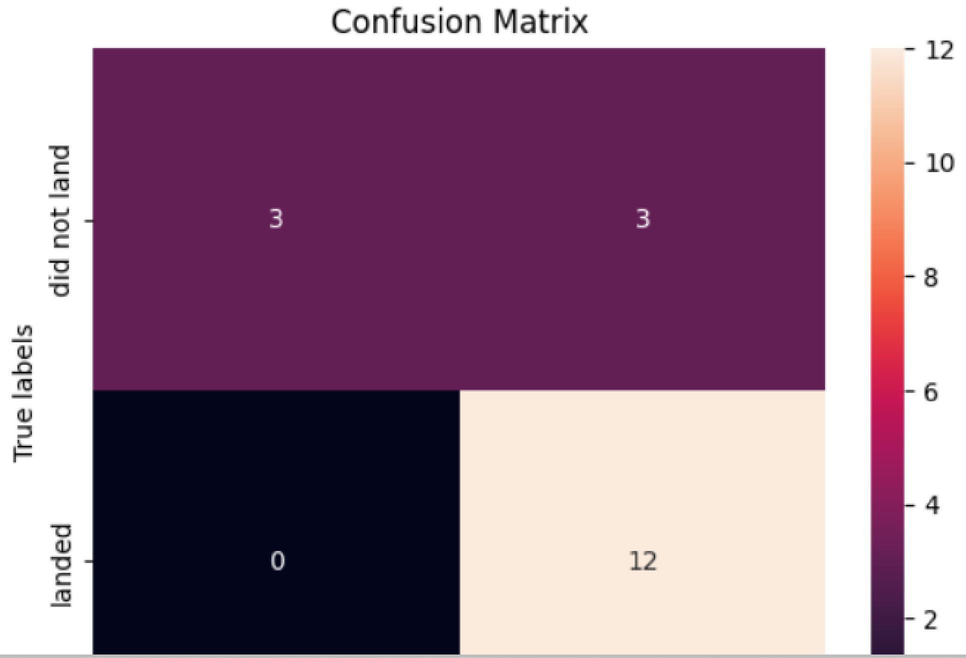
Calculate the accuracy of tree_cv on the test data using the method `score` :

```
[23]: tree_accuracy = tree_cv.score(X_test, Y_test)
      tree_accuracy
```

```
[23]: 0.8888888888888888
```

We can plot the confusion matrix

```
[24]: yhat = svm_cv.predict(X_test)
      plot_confusion_matrix(Y_test,yhat)
```



TASK 11

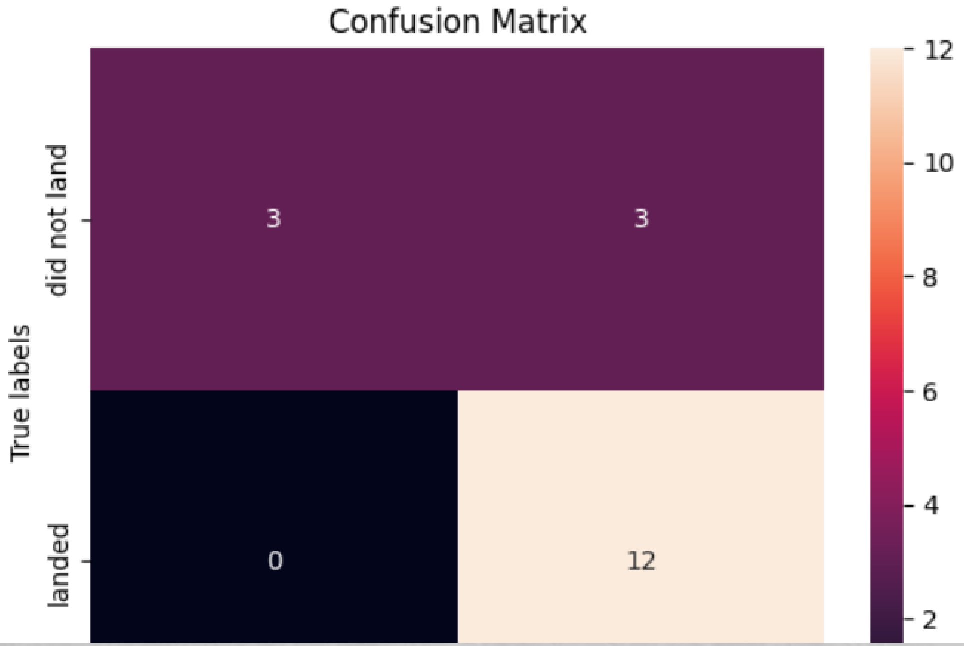
Calculate the accuracy of tree_cv on the test data using the method `score`:

[28]:
```
knn_accuracy = knn_cv.score(X_test, Y_test)
knn_accuracy
```

[28]:
```
0.8333333333333334
```

We can plot the confusion matrix

[29]:
```
yhat = knn_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



TASK 12

Find the method performs best:

[30]:

```
# Examining the scores from the whole Dataset
from sklearn.metrics import jaccard_score, f1_score
jaccard_scores = [
                  jaccard_score(Y, logreg_cv.predict(X), average='binary'),
                  jaccard_score(Y, svm_cv.predict(X), average='binary'),
                  jaccard_score(Y, tree_cv.predict(X), average='binary'),
                  jaccard_score(Y, knn_cv.predict(X), average='binary'),
                 ]

f1_scores = [
             f1_score(Y, logreg_cv.predict(X), average='binary'),
             f1_score(Y, svm_cv.predict(X), average='binary'),
             f1_score(Y, tree_cv.predict(X), average='binary'),
             f1_score(Y, knn_cv.predict(X), average='binary'),
            ]

accuracy = [logreg_cv.score(X, Y), svm_cv.score(X, Y), tree_cv.score(X, Y), knn_cv.score(X, Y)]

scores = pd.DataFrame(np.array([jaccard_scores, f1_scores, accuracy]),
                      index=['Jaccard_Score', 'F1_Score', 'Accuracy'],
                      columns=['LogReg', 'SVM', 'Tree', 'KNN'])
scores
```

[30]:

	LogReg	SVM	Tree	KNN
Jaccard_Score	0.833333	0.845070	0.967213	0.819444
F1_Score	0.909091	0.916031	0.983333	0.900763
Accuracy	0.866667	0.877778	0.977778	0.855556

Conclusion

- Decision Tree Model is the best algorithm for this dataset.
- Launches with a low payload mass show better results than launches with a larger payload mass.
- Most of launch sites are in proximity to the Equator line and all the sites are in very close proximity to the coast.
- The success rate of launches increases over the years.
- KSC LC-39A has the highest success rate of the launches from all the sites.
- Orbits ES-L1, GEO, HEO and SSO have 100% success rate