Data Science Capstone Project

Jonathan I Pollyn

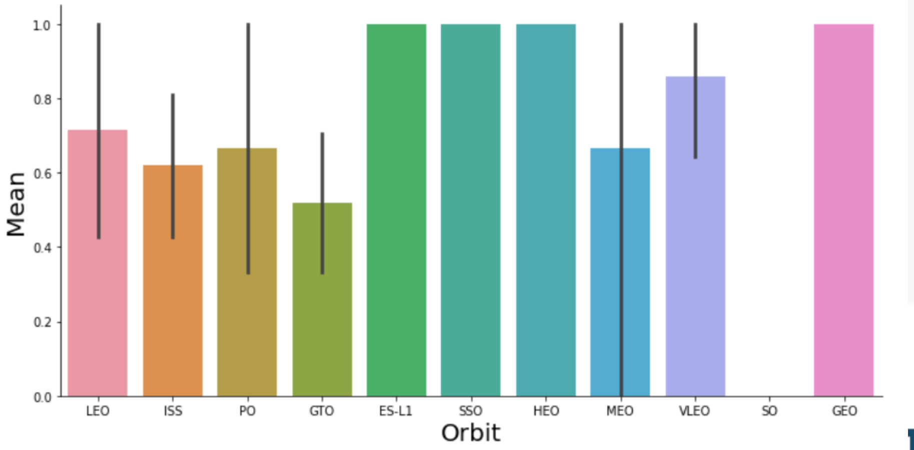
01/03/2022

OUTLINE

- Executive Summary
- Introduction
- Methodology
- Results
  - Visualization – Charts
  - Dashboard
- Discussion
  - Findings & Implications
- Conclusion
- Appendix

EXECUTIVE SUMMARY

- **Methodologies in Brief**
  - Data Collected through API, SQL, and Web scraping
  - Wrangling and Analysis
  - Interactive Maps for Classification using Folium
  - Predictive Analysis
- **Summary of the Final Results**
  - Data analysis with interactive visualization
  - The most effective model for Predictive Analysis



INTRODUCTION

- **Project History**
  - We forecast if the Falcon 9 first stage will land successfully in this capstone in this project. On its website, SpaceX offers Falcon 9 rocket flights at 62 million dollars; other suppliers charge upwards of 165 million dollars per launch; most of the savings come from the fact that SpaceX can reuse the first stage. As a result, if we can determine if the first stage will land, we can figure out how much a launch will cost. If another firm wishes to compete with SpaceX for a rocket launch, this information can be used.
- **Questions to be answered**
  - What are the criteria that will ensure a successful landing of the rocket?
  - What are the consequences of each rocket variable relationship on output?
  - What are the conditions that will help SpaceX achieve the greatest possible result?

METHODOLOGY

- **Data Collection Methodology**
  - SpaceX Data Via Rest API
  - Web Scrapping
- **Data Wrangling**
  - Machining One Hot Encoding
  - Dropping unnecessary column
- **Exploratory Data Analysis (EDA) – using SQL**
  - Scattered plot
  - Bar graph

Methodology-Data Collection

- The project starts with data collection which a process that give us the ability to gather all the necessary data required for this project
- The object set was to
  - Request to the SpaceX API
  - Clean the requested data
- Getting the data from the API
  - Used the request property
  - Data was put into a DataFrame
  - Data is then filtered to weed out unnecessary columns
  - See figure 1.1

METHODOLOGY - Data Wrangling

- Data wrangling technique is applied to the data in other to perform data cleans and unifying missing values
- Percentage of missing values in each attributes
- Calculate the number of launches on each site
- Calculate the number and occurrence of each orbit
- Calculate the number and occurrence of mission outcome per orbit

METHODOLOGY - Exploratory Data Analysis (EDA) – using SQL

- Exploratory Data Analysis was performed whiled extracting data from the database
- Table was created and the data loaded
- The data is then queried using SQL language
- Names of unique launch sites in the space mission was queried
- Query result to display the total payload mass carried by boosters
- Query result for first mission outcome
- Query result for failed landing outcomes in drone ship

METHODOLOGY - Exploratory Data Analysis (EDA) – using visualization

- Visualize the relationship between flight number and launch site
- Visualize the relationship between payload and launch site
- Visualize the relationship between success rate of each orbit type
- Visualize the relationship between flight number and orbit type
- Visualize the launch success yearly trend

RESULTS – Sample data collected from API

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs
0	1	2006-03-24	Falcon 1	20.0	LEO	Kwajalein Atoll	None None	1	False	False	False
1	2	2007-03-21	Falcon 1	NaN	LEO	Kwajalein Atoll	None None	1	False	False	False
2	4	2008-09-28	Falcon 1	165.0	LEO	Kwajalein Atoll	None None	1	False	False	False
3	5	2009-07-13	Falcon 1	200.0	LEO	Kwajalein Atoll	None None	1	False	False	False
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False

RESULTS – Wrangling

Data wrangling shows that there are 55 launches CCAFS SLC 40, 22 launches from KSC LC 39A and 13 from VAFB SLC 4E

```
# Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()
```

```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

RESULTS – Wrangling

The total number and occurrence of each orbit shows that GTO have the highest of 27 orbits

```
# Apply value_counts on Orbit column
df['Orbit'].value_counts()
```

```
]: GTO      27
   ISS      21
   VLEO     14
   PO        9
   LEO       7
   SSO       5
   MEO       3
   SO        1
   GEO       1
   HEO       1
   ES-L1     1
   Name: Orbit, dtype: int64
```

RESULTS – Wrangling

- Calculating the number and occurrence of mission outcome per orbit reveals that ASDS has 41 true and 6 false and RTLS recorded 14 true with a false

```
# landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()
landing_outcomes
```

```
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
False Ocean     2
None ASDS       2
False RTLS      1
Name: Outcome, dtype: int64
```

RESULTS – EDA using SQL

- Querying for unique launch sites shows that there are only 4 unique launch site, and they are CCAFS LC-40, CCAFS SLC-40, KSC LC-39A and VAFB SLC-4E

```
%sql SELECT DISTINCT launch_site FROM SPACEXXTBL
```

```
 * ibm_db_sa://cyy86290:***@fbd88901-ebdb-4a4f-a32e-9
pdomain.cloud:32731/BLUDB
Done.
```

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

RESULTS – using SQL

- The total payload mass carried by boosters was queried and it revealed that boosters launched by NASA (CRS) is 99980 and booster version F9 v1.1 is 2534

```
%sql SELECT avg(PAYLOAD_MASS__KG_) \
FROM SPACEXXTBL \
WHERE BOOSTER_VERSION LIKE 'F9 v1.1%'
```

```
 * ibm_db_sa://cyy86290:***@fbd88901-el
pdomain.cloud:32731/BLUDB
Done.
```

1
2534

```
%sql SELECT sum(PAYLOAD_MASS__KG_) \
FROM SPACEXXTBL \
WHERE CUSTOMER LIKE 'NASA%'
```

```
 * ibm_db_sa://cyy86290:***@fbd88901-ebdb
pdomain.cloud:32731/BLUDB
Done.
```

1
99980

RESULTS – using SQL

- The first successful landing outcome in ground pad was recorded 2015-12-22

```
%sql SELECT min(DATE) \
FROM SPACEXXTBL \
WHERE LANDING__OUTCOME LIKE 'Success (ground pad)'
```

```
 * ibm_db_sa://cyy86290:***@fbd88901-ebdb-4a4f-a32e
pdomain.cloud:32731/BLUDB
Done.
```

1
2015-12-22

RESULTS – using SQL

- The total number of successful and failure mission outcome shows that there are 100 successful outcome and only 1 failure.

```
%sql SELECT COUNT(MISSION_OUTCOME) AS SUCCESSFUL, \
(SELECT COUNT(MISSION_OUTCOME) AS FAILURE FROM SPACEXXTBL WHERE MISSION_OUTCOME LIKE 'Failure%') \
FROM SPACEXXTBL WHERE MISSION_OUTCOME LIKE 'Success%'
```

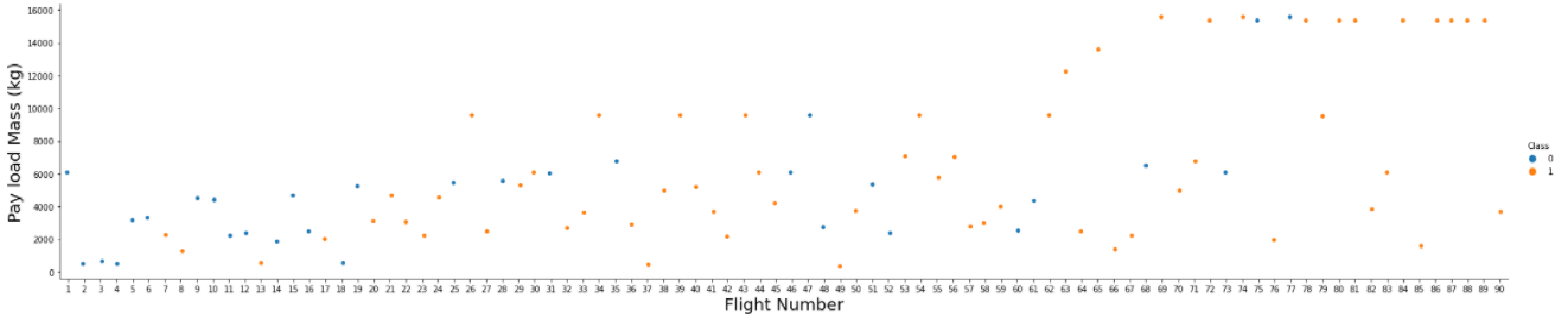
```
 * ibm_db_sa://cyy86290:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomai
Done.
```

successful	failure
100	1

RESULTS – Visualization

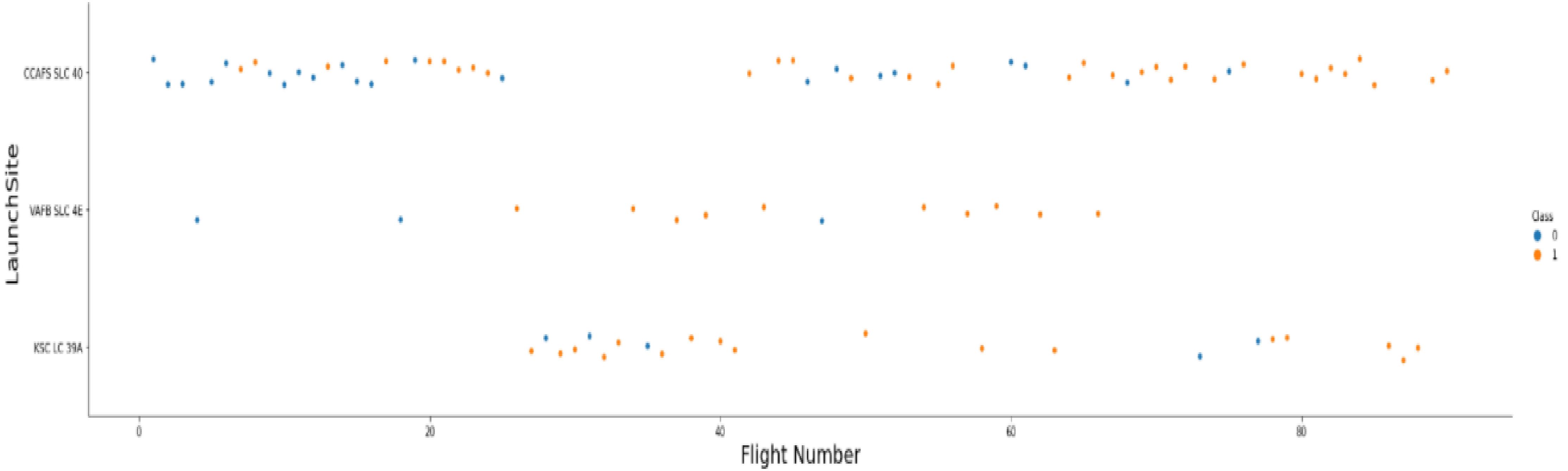
- Varying launch sites have different success percentages, as we can see. CCAFS LC-40 has a 60 percent success rate, whereas KSC LC-39A and VAFB SLC 4E have a 77 percent success rate.

```
sns.catplot(y="PayloadMass", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Pay load Mass (kg)",fontsize=20)
plt.show()
```



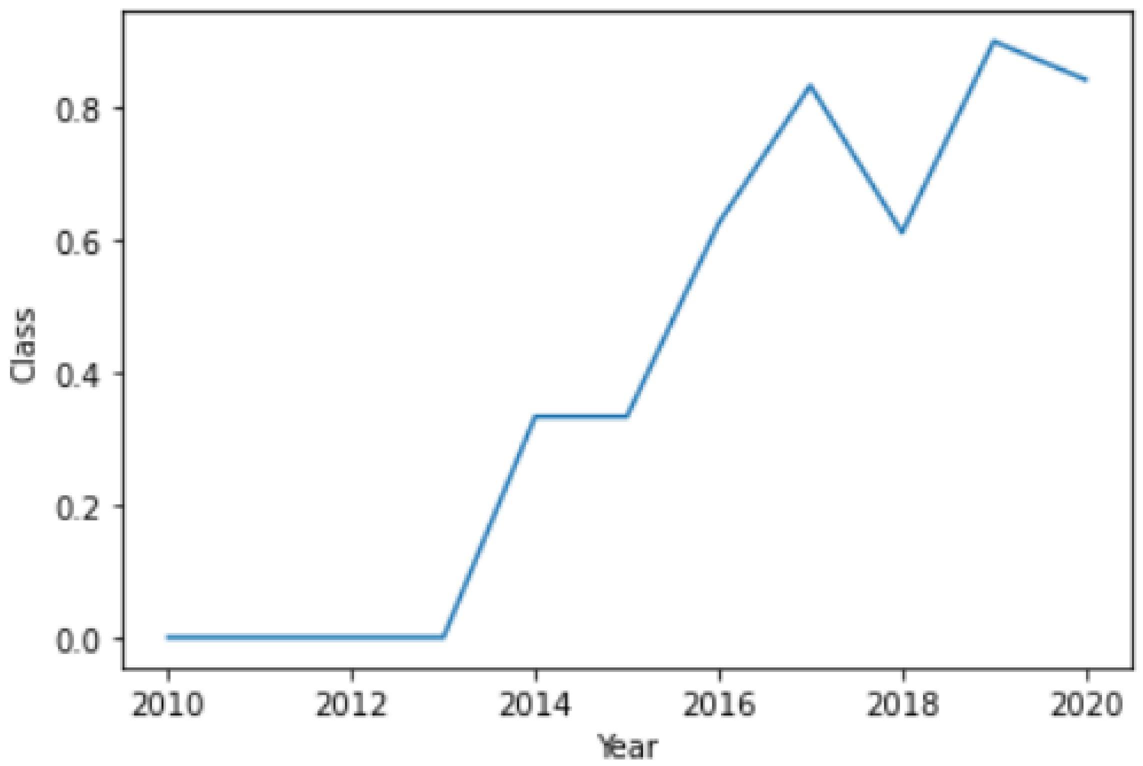
RESULTS – Visualization

```
# 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)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("LaunchSite",fontsize=20)
plt.show()
```



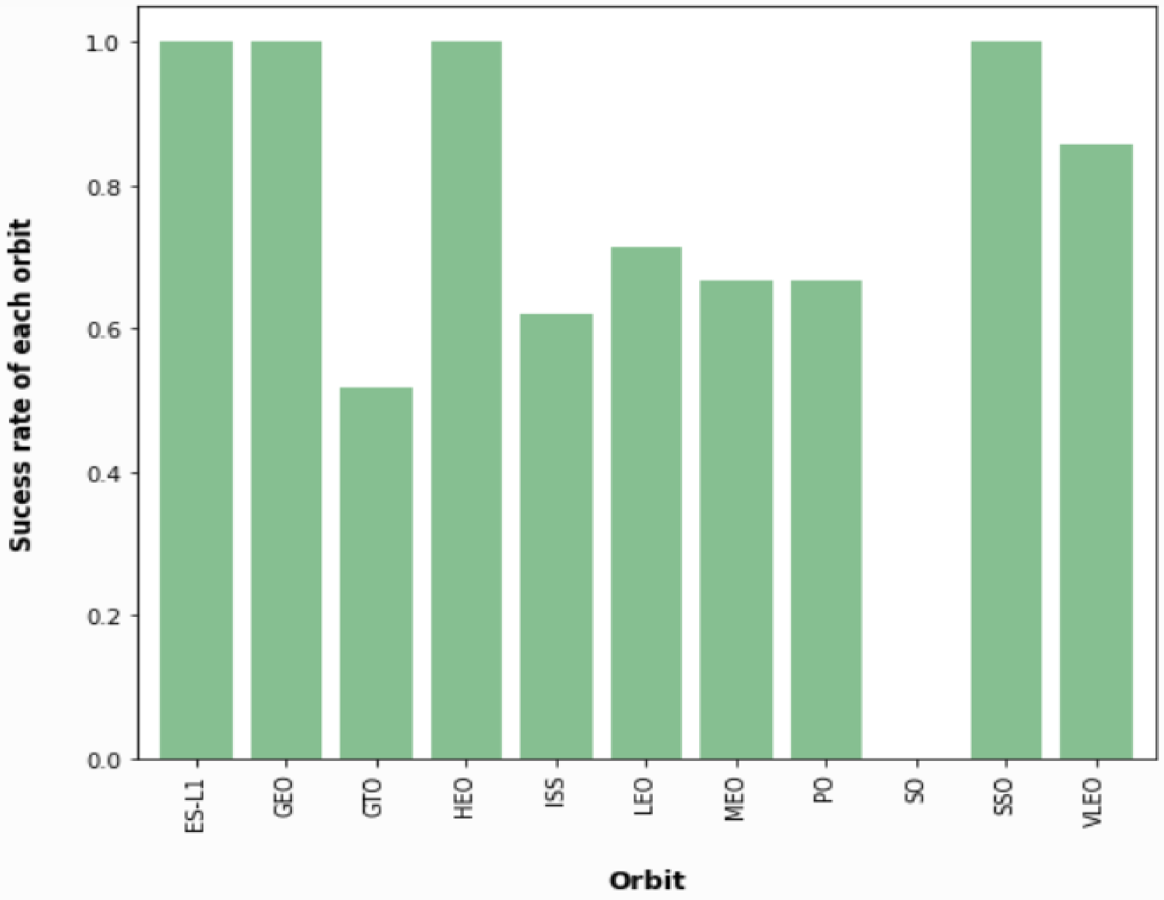
RESULTS – Visualization

- Visualization revealed the launch success yearly trend, which shows that from 2013 launch have been successful. However, there was a drop in successful launches in 2018, but it improved the following year.



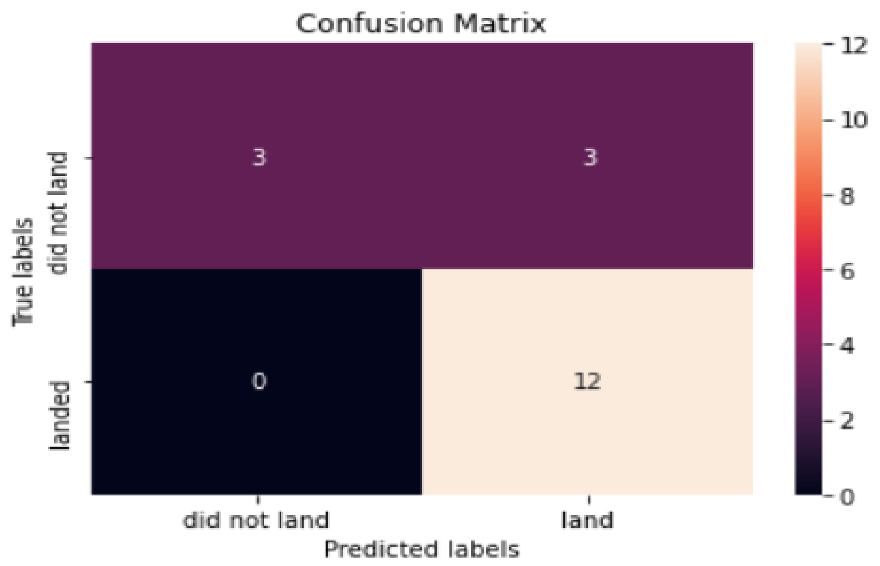
RESULTS – Visualization

- The relationship for the success rate of each Orbit was visualized, which shows that the GTO, ISS, LEO, MEO, PO, and VLEO have a very poor success rate.



RESULTS – Predictive

- Regression analysis shows that there is an accuracy of 87% with test data producing an accuracy of 83%




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

```
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)
```

```
]: GridSearchCV(cv=10, estimator=LogisticRegression(),
             param_grid={'C': [0.01, 0.1, 1], 'penalty': ['l2'],
                         'solver': ['lbfgs']})
```

Ve output the `GridSearchCV` object for logistic regression. We display the best parameters using the data attribute
alidation data using the data attribute `best_score_`.

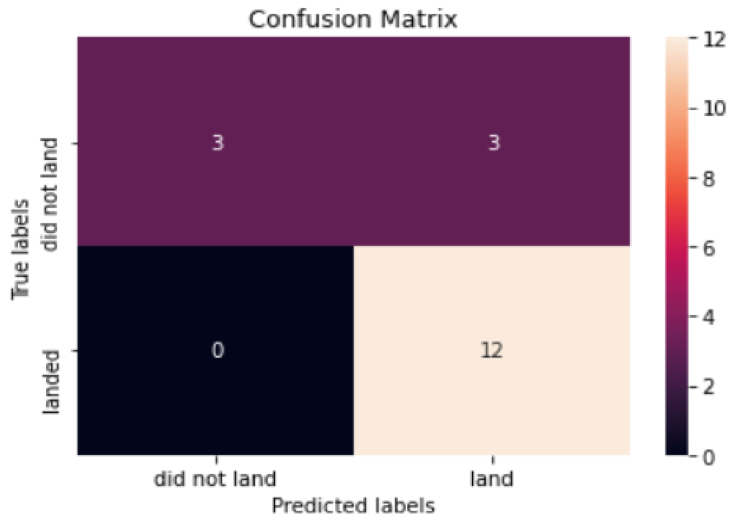
```
print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
print("accuracy :",logreg_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
accuracy : 0.8464285714285713
```

RESULTS – Predictive

- Support vector machine produced 85% accuracy on the training and 83% on the test set.

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




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

```
svm_cv = GridSearchCV(svm, parameters, cv = 10)
svm_cv.fit(X_train, Y_train)
```

```
GridSearchCV(cv=10, estimator=SVC(),
             param_grid={'C': array([1.00000000e-03, 3.16227766e-02, 1.00000000e+00, 3.16227766e+01,
       1.00000000e+03]),
                         'gamma': array([1.00000000e-03, 3.16227766e-02, 1.00000000e+00, 3.16227766e+01,
       1.00000000e+03]),
                         'kernel': ('linear', 'rbf', 'poly', 'rbf', 'sigmoid')})
```

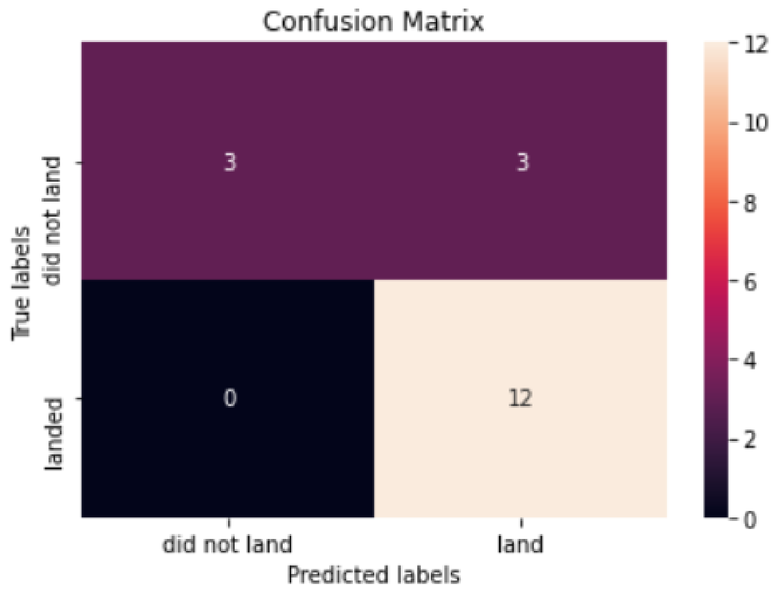
```
print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
print("accuracy :",svm_cv.best_score_)
```

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

RESULTS – Predictive

- Decision tree produced 89 percent accuracy on the training set and 83 percent on the test

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




```python
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()
```

```python
tree_cv = GridSearchCV(tree, parameters, cv=10)
tree_cv.fit(X_train,y_train)
```

```
GridSearchCV(cv=10, estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [2, 4, 6, 8, 10, 12, 14, 16, 18],
                         'max_features': ['auto', 'sqrt'],
                         'min_samples_leaf': [1, 2, 4],
                         'min_samples_split': [2, 5, 10],
                         'splitter': ['best', 'random']})
```

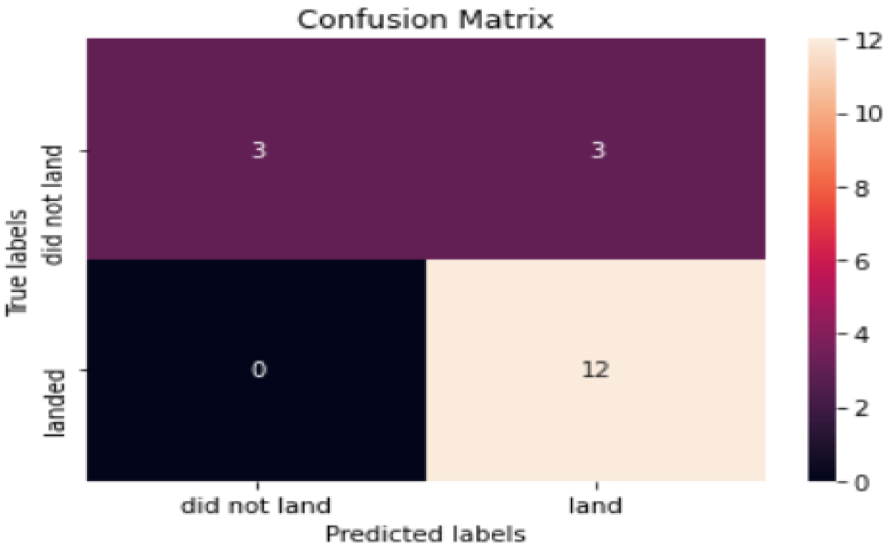
```python
print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
print("accuracy :",tree_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth
4, 'min_samples_split': 10, 'splitter': 'random'}
accuracy : 0.8910714285714285
```

RESULTS – Predictive

- The K Nearest neighbors produced 85 percent accuracy on training and 83 percent on test

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




```
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()
```

```
knn_cv = GridSearchCV(KNN, parameters, cv=10)
knn_cv.fit(X_train,y_train)
```

```
GridSearchCV(cv=10, estimator=KNeighborsClassifier(),
             param_grid={'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
                         'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                         'p': [1, 2]})
```

```
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
```

RESULTS – Predictive

- Overall, it shows that the decision tree came out to be the best performer

	index	Accuracy
0	KNN	0.848214
1	Decision Tree	0.891071
2	Logistic Regression	0.846429
3	SVM	0.848214