IBM DATA SCIENCE CAPSTONE

Nada Mohamed

28 Sept 2021


AGENDA
01
EXECUTIVE SUMMARY
02
INTRODUCTION
03
METHODOLOGY
04
RESULTS
05
CONCLUSION
06
APPENDIX

EXECUTIVE SUMMARY

SUMMARY OF METHODOLOGIES

- Data Collection.
- Data Wrangling.
- Exploratory Data Analysis with Visualization. Exploratory Data Analysis with SQL.
- Visual Analytics with Folium.
- Dashboard with Dash.
- Predictive Analysis working

SUMMARY OF RESULT

- EDA Findings.
- Prediction through Support vector machine, Classification Tree & Logistics Regression.
- For SVM sigmoid kernel provides better result on validation dataset.
- Hyper parameters for decision tree classifier through validation data.

INTRODUCTION

- **Project background and context**
  - The spaceX advertises the launch of Falcon 9 rocket through their website, it has a cost of 62 million USD, whereas the other providers cost are 165 million USD each, the main reason of the savings of spaceX is because that it can be reused the first stage.
- **Problems you want to find answers**
  - We can determine the cost of the launch if we are determining that spaceX first stage will land.
  - The extracted information can be used for other companies who are interested to bid spaceX for launch of the rocket. We will be predicting that either spaceX first stage would be successfully be landed or not.

METHODOLOGY

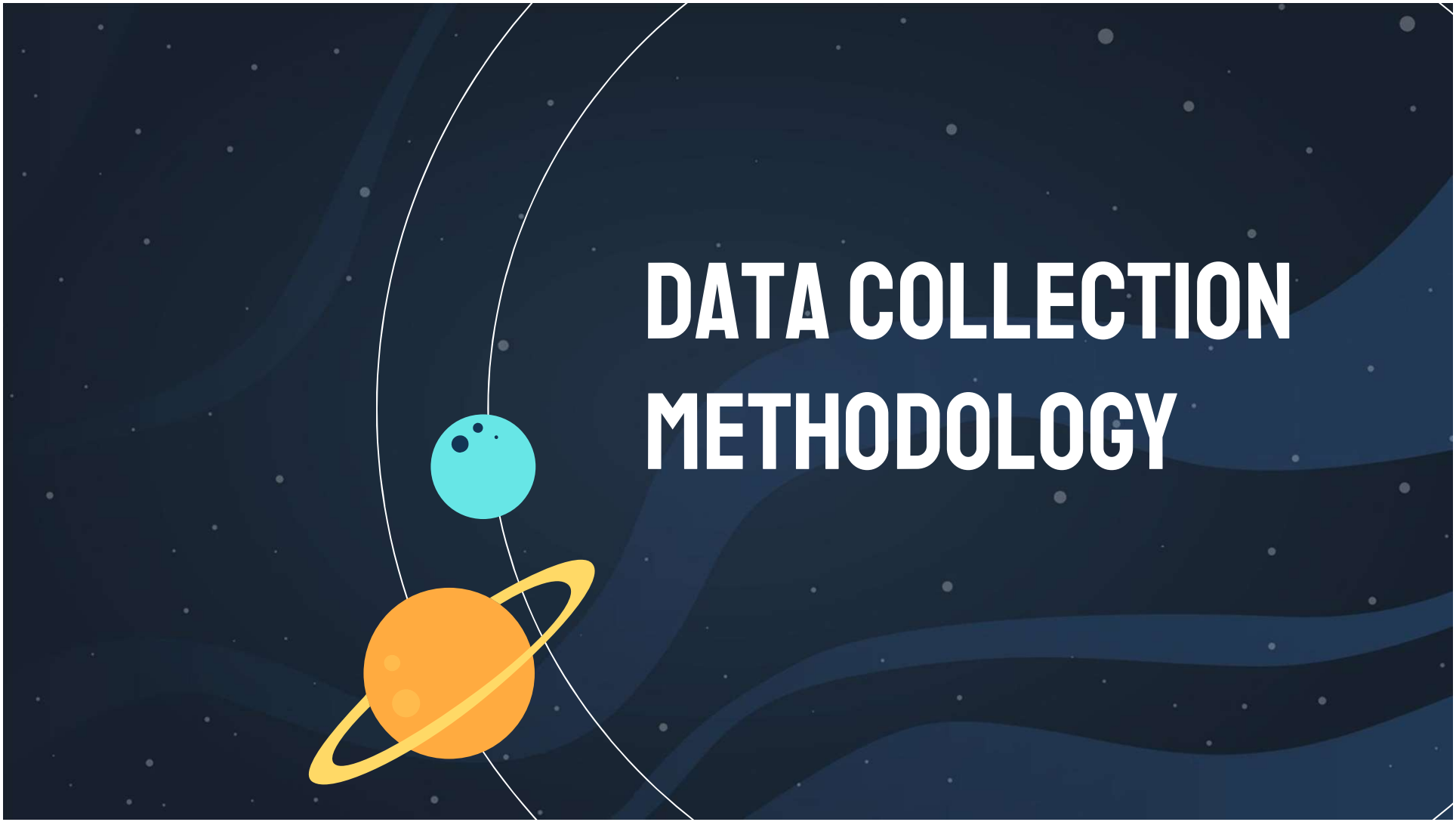
- **Data Collection:**
  - SpaceX Rest API.
  - Web Scrapping Wikipedia.

- **Data Wrangling:**
  - Hot encoding data fields and dropping irrelevant columns.

- **Exploratory Data Analysis (EDA) using visualization and SQL:**
  - Scatter/bar graphs.

- **Interactive Visual Analytics using Folium and Plotly.**

- **Predictive Analysis using classification models:**
  - Classification models.


DATA COLLECTION
METHODOLOGY

DATA COLLECTION VIA SPACEX API

FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial
1	2010-06-04	Falcon 9	6123.547647	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0003
2	2012-05-22	Falcon 9	525.000000	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0005
3	2013-03-01	Falcon 9	677.000000	ISS	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0007
4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	None	1.0	0	B1003
5	2013-12-03	Falcon 9	3170.000000	GTO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B1004

1. Get response from API
2. Convert to json file
3. Clean Data
4. Convert to dataframe
5. Filter & export as file

GitHub Link

DATA COLLECTION VIA WEB SCRAPING

Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date	Time
1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	F9 v1.0B0003.1	Failure	4 June 2010	18:45
2	CCAFS	Dragon	0	LEO	NASA	Success	F9 v1.0B0004.1	Failure	8 December 2010	15:43
3	CCAFS	Dragon	525 kg	LEO	NASA	Success	F9 v1.0B0005.1	No attempt	22 May 2012	07:44
4	CCAFS	SpaceX CRS-1	4,700 kg	LEO	NASA	Success	F9 v1.0B0006.1	No attempt	8 October 2012	00:35
5	CCAFS	SpaceX CRS-2	4,877 kg	LEO	NASA	Success	F9 v1.0B0007.1	No attempt	1 March 2013	15:10

1. Get response from HTML
2. Creat Soup Object
3. Find Tables
4. Get coloumn names
5. Creat Dict & append data to keys
6. Convert Dict to dataframe
7. Export as CSV

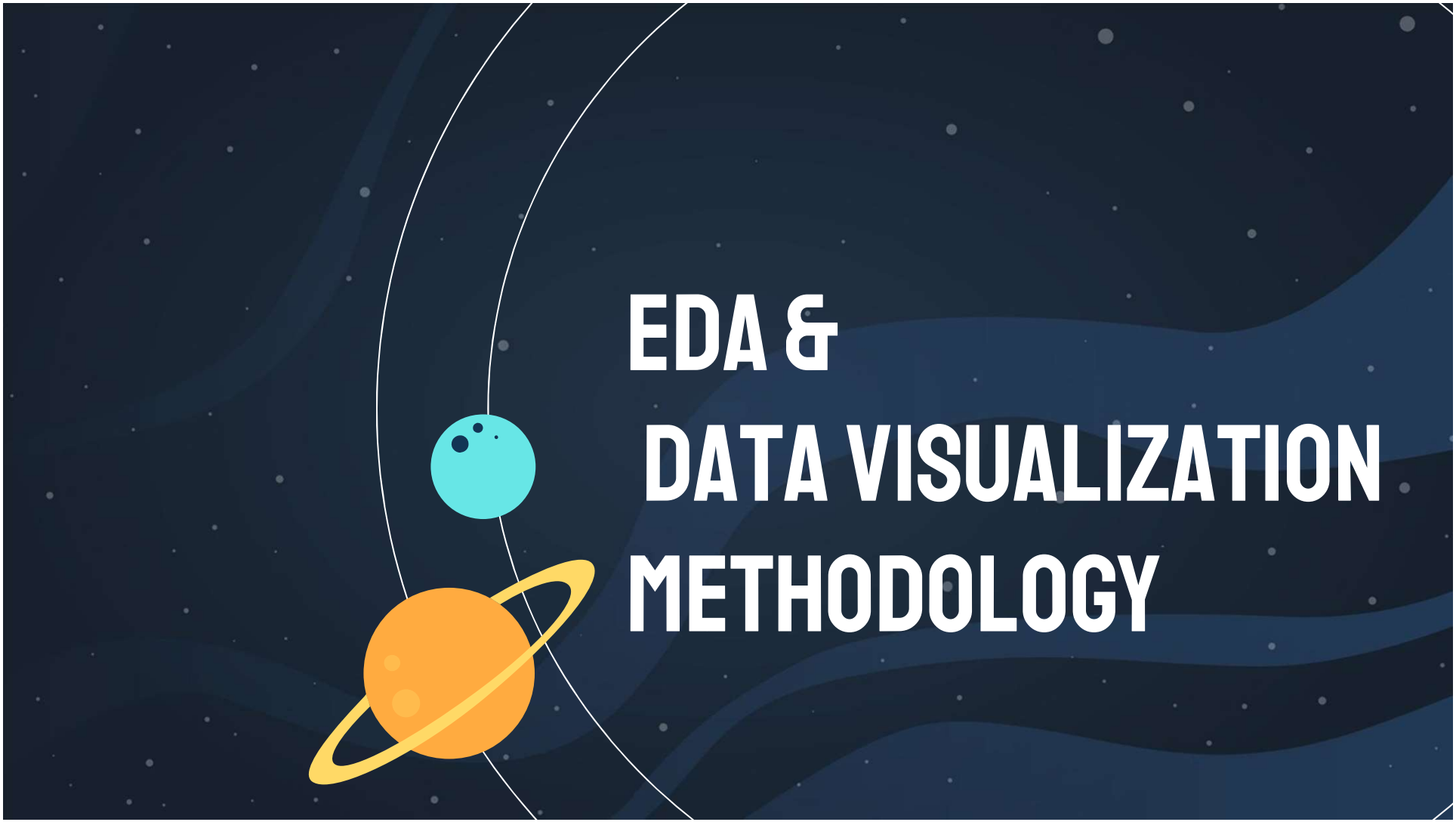
GitHub Link

DATA WRANGLING

FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial
1	2010-06-04	Falcon 9	6123.547647	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0003
2	2012-05-22	Falcon 9	525.000000	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0005
3	2013-03-01	Falcon 9	677.000000	ISS	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0007
4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	None	1.0	0	B1003
5	2013-12-03	Falcon 9	3170.000000	GTO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B1004

1. Calculate number of launches at each site
2. Calculate number and occurrence of each orbit
3. Calculate number and occurrence of mission outcome per orbit type
4. Create landing outcome label from Outcome column
5. Create dictionary and append data to keys
6. Export as .csv

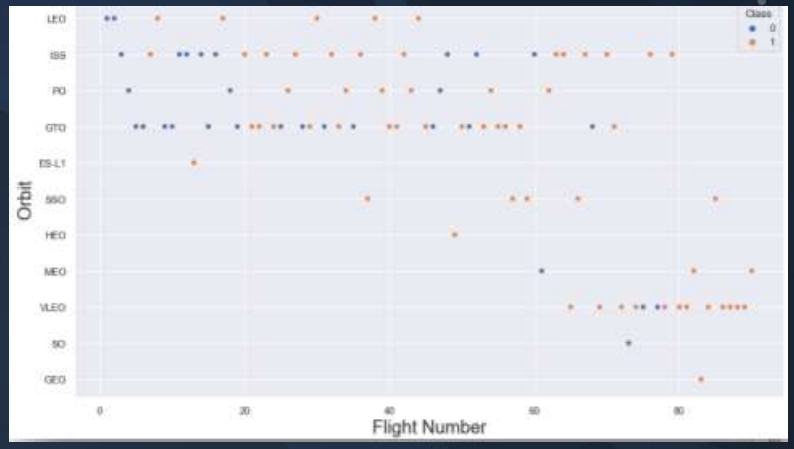
GitHub Link

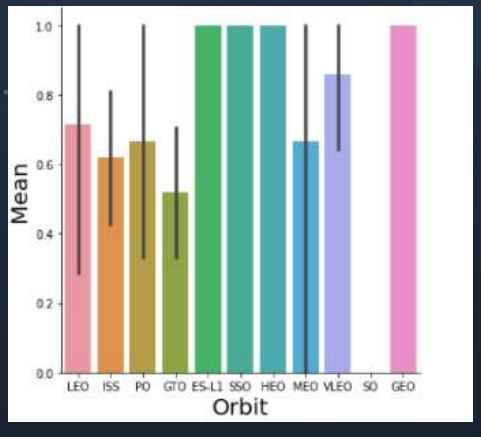

EDA &
DATA VISUALIZATION
METHODOLOGY

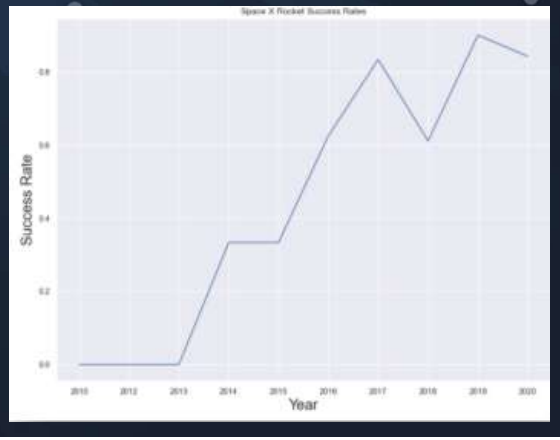
EDA WITH DATA VISULAZATION

- **Scatter graph:** to determine whether there is a noticeable dependency between the attributes
- **Bar graph:** to help identify any visual trends or relationships.
- **Line graph:** helps to track the direct relationship and pattern between the data points

GitHub Link

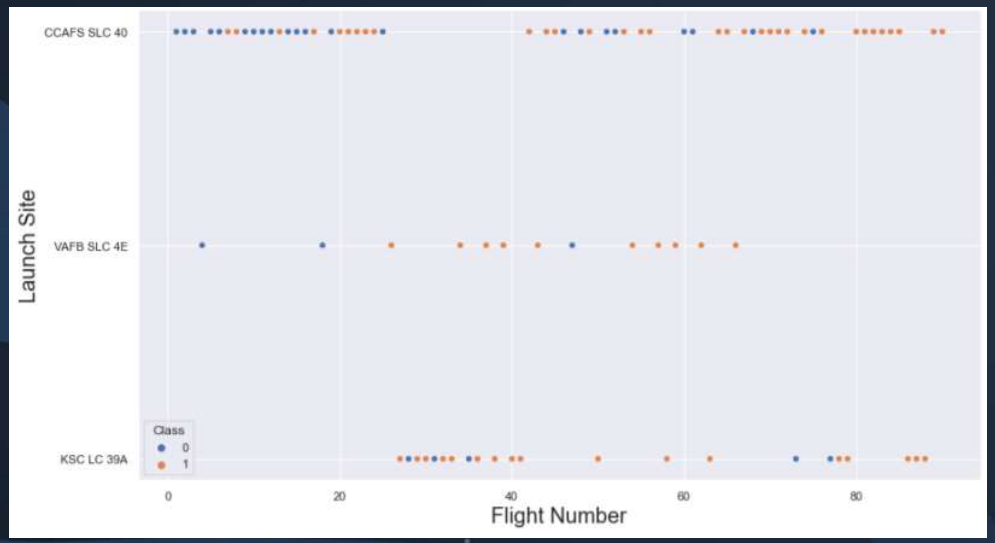






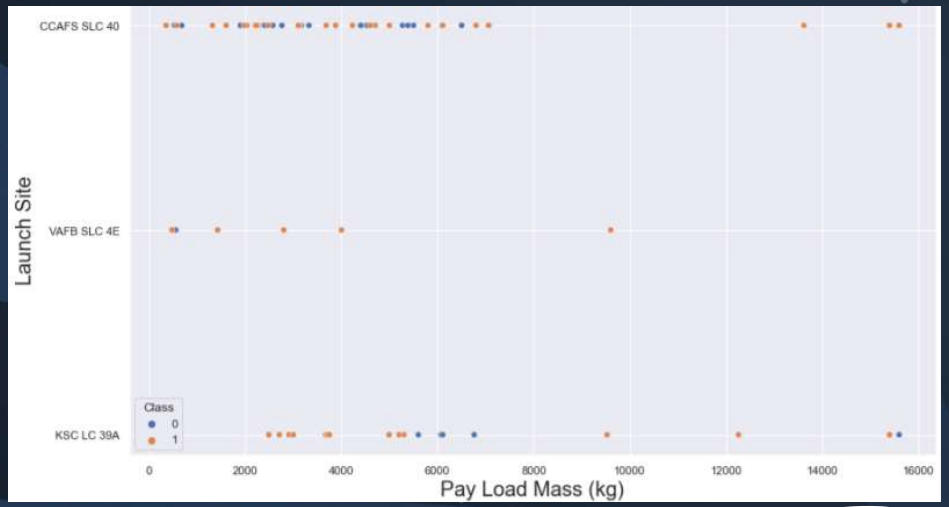
WHFLIGHT NUMBER VS. LAUNCH SITE

- With higher flight numbers (> 30), the success rate increases.



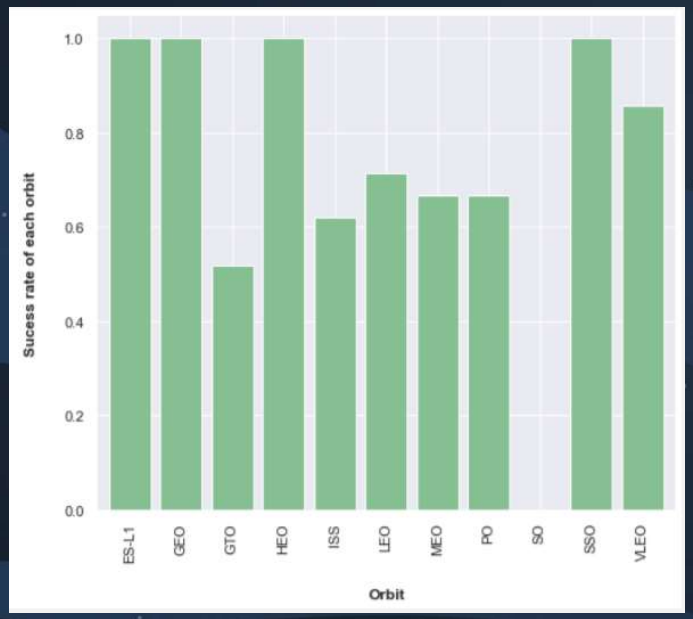
PAYLOAD VS LAUNCH SITE

- With greater payload mass (> 7000 KG), the higher the success rate for the rocket but payload mass and launch site are not directly correlated. .



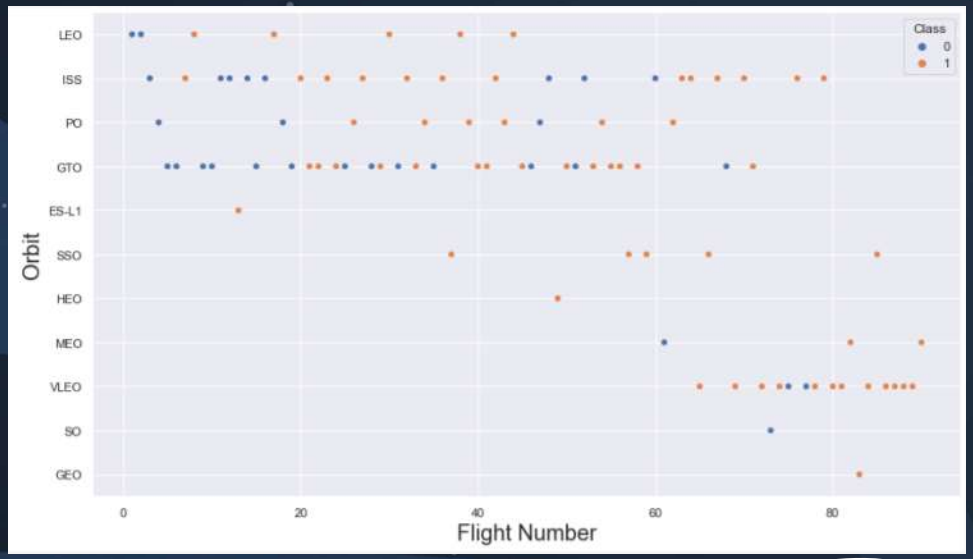
SUCCESS RATE VS. ORBIT TYPE

- The ES-L1, GEO, HEO and SSO orbits had the highest success rate. .



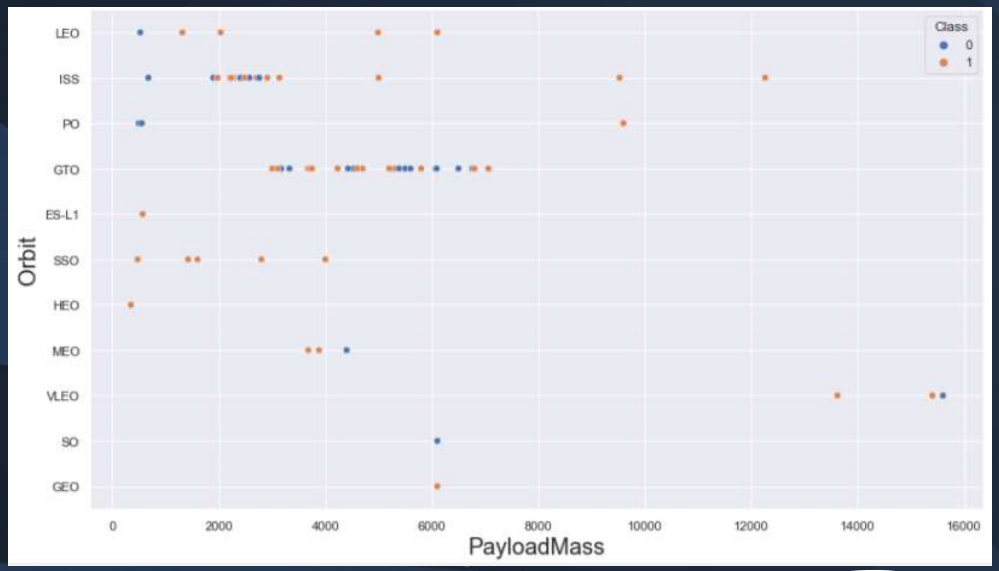
FLIGHT NUMBER VS ORBIT TYPE

- The LEO orbit had the highest success rate with a higher number of flights.



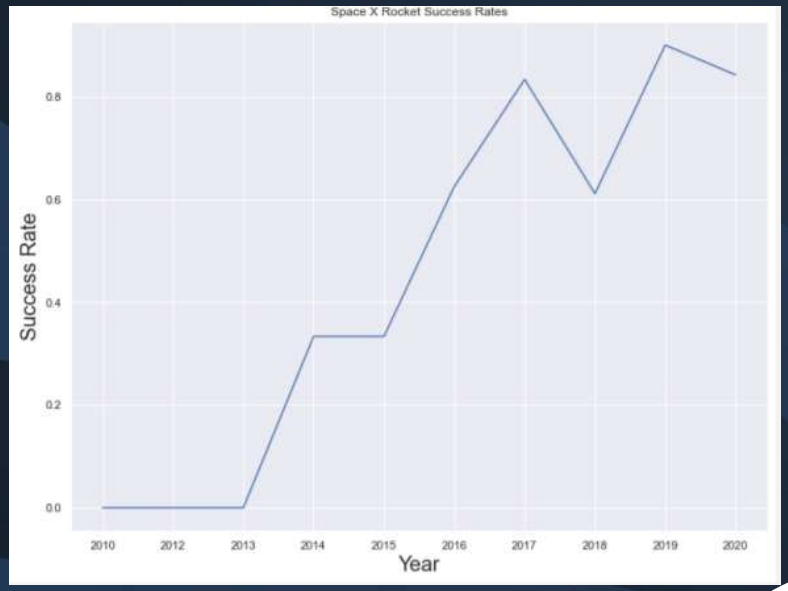
PAYLOAD VS. ORBIT TYPE

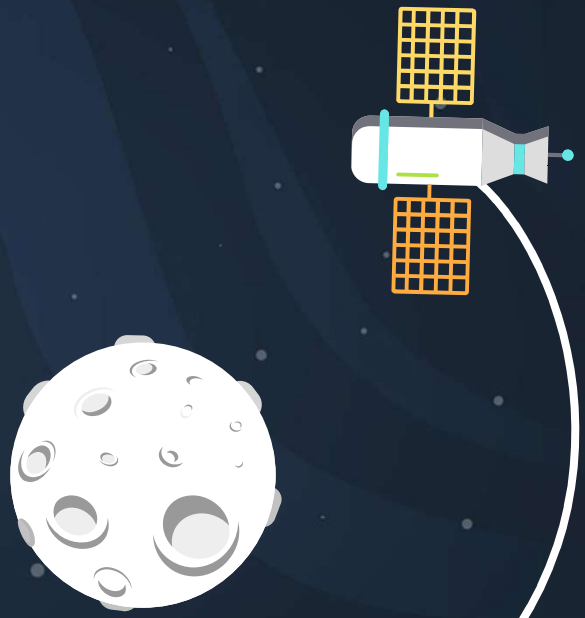
- Higher payloads negatively impact the orbits.



LAUNCH SUCCESS YEARLY TREND

- Success rate since 2013 has increased consistently.





EDA WITH SQL

Queries performed:

1. Display the names of the unique launch sites

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

3. Display total payload mass carried by boosters launched by NASA

4. Display average payload mass carried by booster version F9

5. List the date where the successful landing outcome in drone ship was achieved

6. List the names of the boosters which have success in ground pad and have a payload mass greater than 4000 and less than 6000

7. List the total number of successful and failed mission outcomes

8. List the names of the booster_versions which have carried the maximum payload mass

9. List the failed landing_outcomes in drone ship, booster versions and launch site names in 2015

10. Rank the count of landing outcomes between 2010 and 2017

```
%sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEX;
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

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

1. Using DISTINCT in the query, we pull all the unique values for the Launch_Site column from SPACEX table.

```
In [6]: %sql SELECT * FROM SPACEX WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

Out[6]:

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	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	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

2. Using keyword 'Limit 5' in the query, we get 5 records from the SPACEX table and search with wildcard 'CCA%'.

```
In [7]: %sql SELECT SUM(PAYLOAD_MASS__KG_) AS "Total Payload Mass by NASA (CRS)" FROM SPACEX WHERE CUSTOMER = 'NASA (CRS)';
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

Out[7]:

Total Payload Mass by NASA (CRS)
45596

3. Using the function SUM, we calculate the total in the PAYLOAD_MASS_KG_ column and WHERE clause to filter the data by the customer name.

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS "Average Payload Mass by Booster Version F9 v1.1" FROM SPACEX \
WHERE BOOSTER_VERSION = 'F9 v1.1';
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

Average Payload Mass by Booster Version F9 v1.1
2928

4. Using AVG, we determine the average of the column PAYLOAD_MASS_KG_ and the WHERE clause to filter the dataset.

```
%sql SELECT MIN(DATE) AS "First Succesful Landing Outcome in Ground Pad" FROM SPACEX \
WHERE LANDING__OUTCOME = 'Success (ground pad)';
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

First Succesful Landing Outcome in Ground Pad
2015-12-22

5. Using MIN, we determine the first date in the Date column and WHERE clause to filter the data for successful landings.

```
%sql SELECT BOOSTER_VERSION FROM SPACEX WHERE LANDING__OUTCOME = 'Success (drone ship)' \
AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

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

6. Selecting only Booster_Version, we use the WHERE clause to filter the dataset for successful landings and apply the payload parameters.

```
%sql SELECT sum(case when MISSION_OUTCOME LIKE '%Success%' then 1 else 0 end) AS "Successful Mission", \
    sum(case when MISSION_OUTCOME LIKE '%Failure%' then 1 else 0 end) AS "Failure Mission" \
FROM SPACEX;
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

Successful Mission	Failure Mission
100	1

7. Using the case when MISSION_OUTCOME LIKE '%Success%' then 1 else 0 end to return the Boolean value which we get the sum.

```
%sql SELECT DISTINCT BOOSTER_VERSION AS "Booster Versions which carried the Maximum Payload Mass" FROM SPACEX \
WHERE PAYLOAD_MASS__KG_ =(SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEX);
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

Booster Versions which carried the Maximum Payload Mass
F9 B5 B1048.4
F9 B5 B1048.5
F9 B5 B1049.4
F9 B5 B1049.5
F9 B5 B1049.7
F9 B5 B1051.3
F9 B5 B1051.4
F9 B5 B1051.6
F9 B5 B1056.4
F9 B5 B1058.3
F9 B5 B1060.2
F9 B5 B1060.3

8. Using the MAX function to determine the maximum payload in the column PAYLOAD_MASS_KG_ and filter for the Booster Version

```
%sql SELECT {fn MONTHNAME(DATE)} as "Month", BOOSTER_VERSION, LAUNCH_SITE FROM SPACEX WHERE year(DATE) = '2015' AND \
LANDING__OUTCOME = 'Failure (drone ship)';
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

Month	booster_version	launch_site
January	F9 v1.1 B1012	CCAFS LC-40
April	F9 v1.1 B1015	CCAFS LC-40

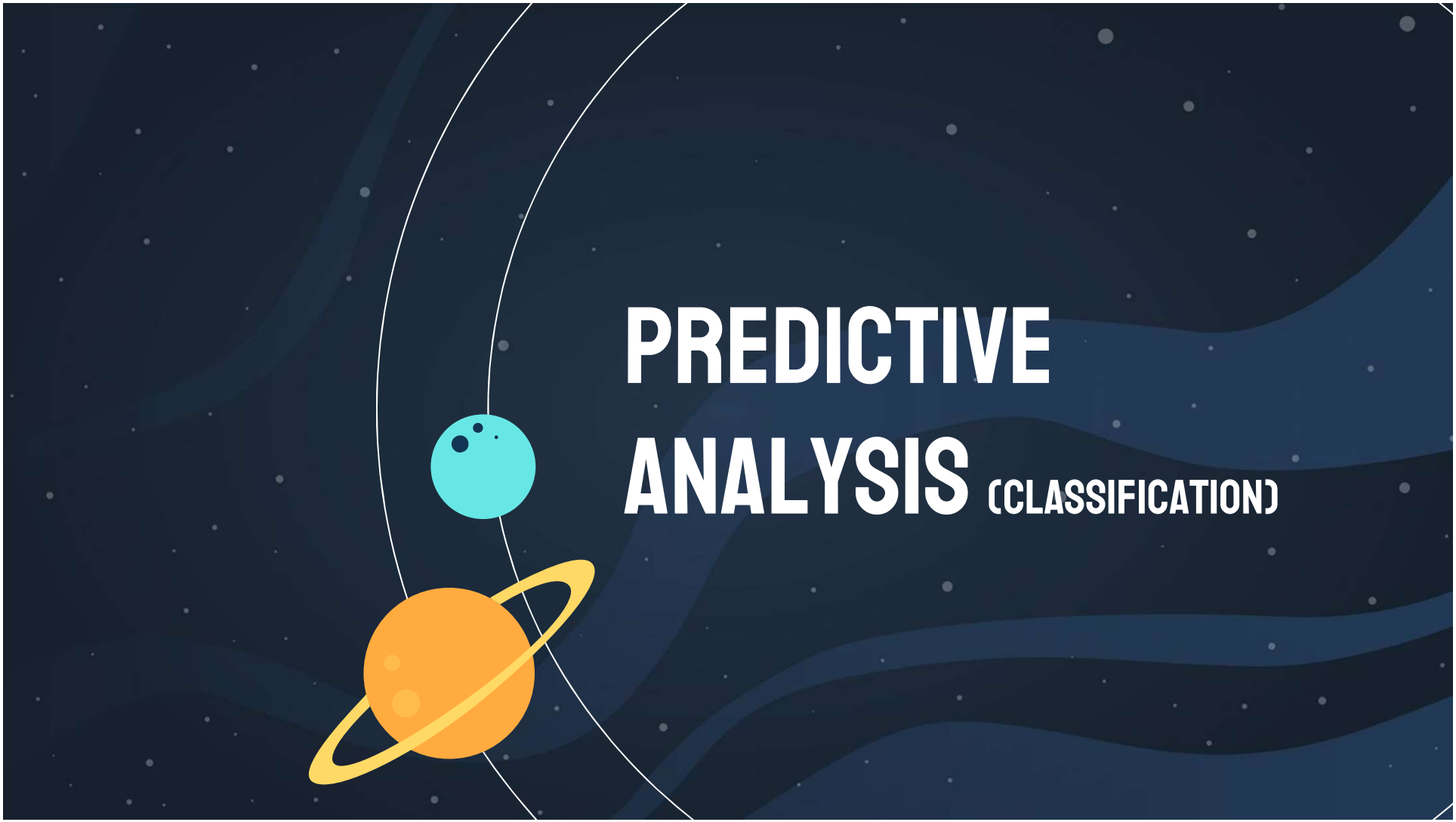
9. List the records to display the month names, failures in drone ship, booster versions, launch_site for

```
%sql SELECT LANDING__OUTCOME as "Landing Outcome", COUNT(LANDING__OUTCOME) AS "Total Count" FROM SPACEX \
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' \
GROUP BY  LANDING__OUTCOME \
ORDER BY COUNT(LANDING__OUTCOME) DESC ;
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

Landing Outcome	Total Count
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

10. Select only LANDING_OUTCOME, WHERE clause between DATE BETWEEN 2010-06-04 and 2017-03-20. Group by Count in descending order.


PREDICTIVE
ANALYSIS (CLASSIFICATION)

Built the model

- Load the engineered data into a dataframe
- Transform and standardize the data using Numpy
- Split the data into training and test data sets
- Check how many samples were created and set our parameters/algorithms
- Fit the datasets into the GridSearchCV objects to train the model

Evaluating the model

- Check the accuracy of the model and plot the Confusion Matrix
- Finding the best performing classification model
- Use the highest accuracy score

GitHub Link

KNN

```
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()
```

```
gscv = GridSearchCV(KNN,parameters,scoring='accuracy',cv=10)
knn_cv = gscv.fit(X_train,Y_train)
```

```
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
```

TASK 11

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

```
print("accuracy: ",knn_cv.score(X_test,Y_test))
```

```
accuracy:  0.8333333333333334
```

LOGISTIC REGRESSION

```
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()
```

```
gscv = GridSearchCV(lr,parameters,scoring='accuracy',cv=10)
logreg_cv = gscv.fit(X_train,Y_train)
```

We output the `GridSearchCV` object for logistic regression. We display the best parameters using the data attribute `best_params\_` and the accuracy on the validation 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
```

SVM

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

```
gscv = GridSearchCV(svm,parameters,scoring='accuracy',cv=10)
svm_cv = gscv.fit(X_train,Y_train)
```

```
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
```

DECISION TREE

```
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()
```

```
gscv = GridSearchCV(tree,parameters,scoring='accuracy',cv=10)
tree_cv = gscv.fit(X_train,Y_train)
```

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

```
tuned hpyerparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 18, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_s
amples_split': 2, 'splitter': 'best'}
accuracy : 0.8785714285714287
```

TASK 9

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

```
print("accuracy: ",tree_cv.score(X_test,Y_test))
```

```
accuracy:  0.9444444444444444
```

ACCURACY

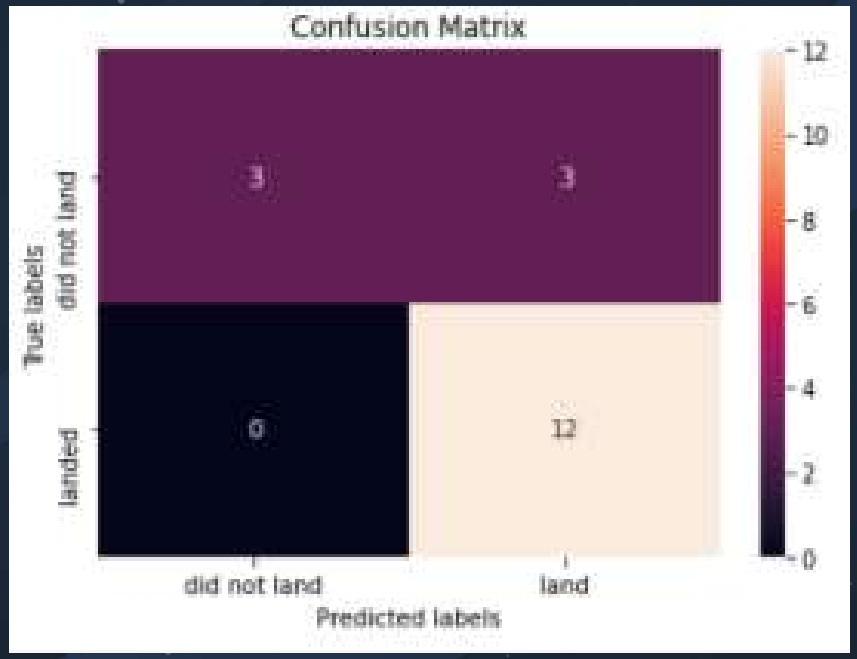
(Decision Tree has THE BEST accuracy)

```
algo_df.rename(columns = {'index': 'Algorithm'}, inplace = True)
algo_df.head()
```

	Algorithm	Accuracy
0	Logistic Regression	0.846429
1	SVM	0.848214
2	KNN	0.848214
3	Decision Tree	0.901786

CONFUSION MATRIX

(All models have the same confusion matrix)



CONCLUSION

- Number of launched success have increases over the years
- ELS1, GEO, HEO, SSO have good success rate
- KSCLC39 have best success rate.
- Launch outcome impacted by the payload mass.
- Accuracy is same on the test model
- Decision tree classifier has highest accuracy.

APPENDIX

Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project.

THANKS!

Do you have any questions?
nada9awad@gmail.com