Outline

- Executive Summary

- Introduction

- Methodology

- Results

- Conclusion

- Appendix

Executive Summary

- Data is collected by requesting rocket launch data from SpaceX API and turning it into a Pandas dataframe.

- Web scraping is performed using BeautifulSoup to collect Falcon 9 historical launch records from a Wikipedia page titled List of Falcon 9 and Falcon Heavy launches.

- Performing some Exploratory Data Analysis (EDA) to find some patterns in the data and determining what would be the label for training supervised models. Also, missing values are replaced by the appropriate values.

- Data visualization was carried to find out whether there was any relation between variables and the outcome of the launch.

- EDA with SQL was also carried out to find out the launch site names, total payload mass, number of successful launches etc.

- An Interactive Map with Folium is created and markers, popups are added to indicate launch sites, distance to coasts, highways etc

- A dashboard application is also created which contains input components such as a dropdown list and a range slider to interact with a pie chart and a scatter point chart to observe the mission outcomes

- Finally, Predictive Analysis (Classification) is carried out using different models like decision tree, svm, knn etc.

Introduction

- SpaceX advertises Falcon 9 rocket launches on its website, with a cost of 62 million dollars, while other providers cost upward of 165 million dollars each. Much of the savings is because SpaceX can reuse the first stage.

- In this capstone, we will predict if the Falcon 9 first stage will land successfully. If we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.

- In other words, in this classification, we have to determine the best model which can predict if the first stage will land successfully or not, given a set of inputs such as payload, the orbit to which the payload is intended, etc.

Section 1

Methodology

Methodology

Executive Summary

- **Data collection methodology:**

  - Rocket launch data is requested from SpaceX API and it is converted into a Pandas dataframe.

  - The data from these requests like booster name, payload, orbit, launch site, its coordinates, outcome of the landing, number of flights etc. will be stored in lists and will be used to create a new dataframe.

  - Finally, the Falcon 1 launches are removed keeping only the Falcon 9 launches.

- **Performing data wrangling**

  - Exploratory Data Analysis (EDA) is performed to find some patterns in the data and labels are determined for training supervised models.

- A variable landing_class is created where the element is zero if the corresponding row in Outcome is in the set bad_outcome, otherwise, it is one

- Also, the np.nan values in PayloadMass is replaced with its mean.

- The LandingPad column will retain None values to represent when landing pads were not used.

- **Exploratory data analysis (EDA)** using visualization and SQL is also performed

- **Interactive visual analytics** using Folium and Plotly Dash is also done

- Finally, **predictive analysis** is carried out using different classification models. The steps include

- Standardizing the variables

- Splitting the data into training and testing set

- Beginning with logistic regression, creating and fitting the object and finding the best parameters. Finally, accuracy of the model is calculated. These steps are repeated for other models to find the best model.

Data Collection

- Rocket launch data is requested from SpaceX API and is converted into a Pandas dataframe.

- Helper functions are defined to extract information.

- The data from these requests is stored in lists and will be used to create a new dataframe.

Data Collection – SpaceX API

- Finally , the dataframe is filtered to only keep the Falcon 9 launches.

- https://github.com/Ryzen17/My_rep/ blob/main/jupyter_labs_spacex_data _collection_api.ipynb

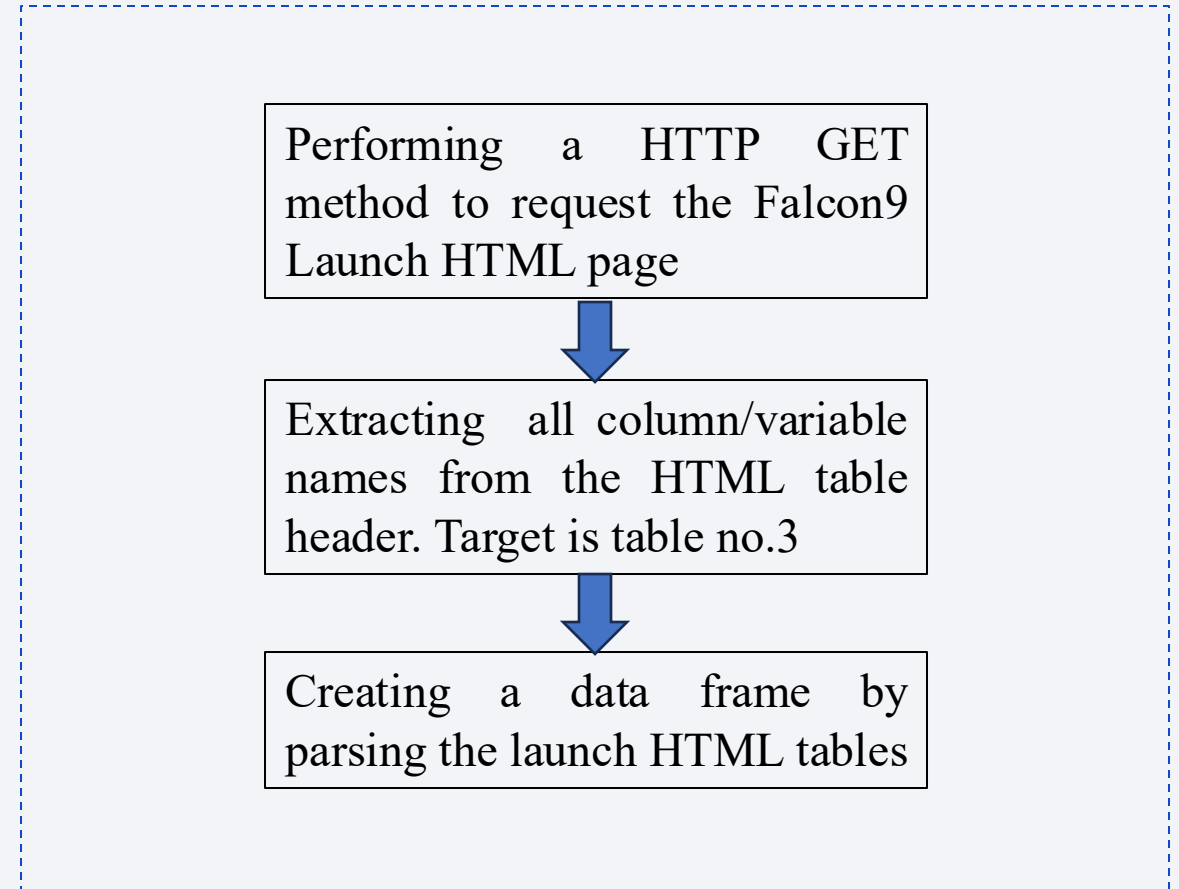requesting rocket launch data from SpaceX API

Decoding the response content as a Json using .json() and turn it into a Pandas dataframe using .json_normalize()

defining a series of helper functions to use the API to extract information using identification numbers in the launch data.

Data Collection - Scraping

- Web scraping is performed using BeautifulSoup to collect Falcon 9 historical launch records from a Wikipedia page titled List of Falcon 9 and Falcon Heavy launches

- Using helper functions, the required HTML table is scraped

- https://github.com/Ryzen17/My_rep/blob/main/jupyter-labs-webscraping.ipynb

Performing a HTTP GET method to request the Falcon9 Launch HTML page

⬇

Extracting all column/variable names from the HTML table header. Target is table no.3

⬇

Creating a data frame by parsing the launch HTML tables

Data Wrangling

Calculating the number of launches on each site using value_counts()

↓

Calculating the number and occurrence of each orbit using the method value_counts()

↓

Calculating the number and occurrences of mission outcome of the orbits and assigning it to a variable landing_outcomes.

↓

Creating a variable landing_class where the element is zero if the corresponding row in Outcome is in the set bad_outcome; otherwise, it's one

→

Replacing np.nan values in PayloadMass with its mean.

- Performing some Exploratory Data Analysis (EDA) to find some patterns in the data and determining what would be the label for training supervised models.

- https://github.com/Ryzen17/My_rep/blob/main/labs_jupyter_spacex_Data_wrangling.ipynb

13

EDA with Data Visualization

- A scatter plot between FlightNumber and PayloadMass is made and it is overlaid with the outcome of the launch. This is done to check whether there is any relation between flight number, payload mass and the outcome of the launch.

- Similarly, scatter plots between FlightNumber and LaunchSite & PayloadMass annd LaunchSite are made to reveal any relation between flight number, payload mass and launch sites with the outcome of the launches.

- Also, a bar chart depicting the success rate of each orbit is also plotted.

- Another scatter plot between FlightNumber and Orbit type is plotted to reveal their relationship with outcome.

- Further, a plot between the Payload Mass and Orbit is made to show how Payload Mass and Orbit type is related to the outcome of the launch.

- Finally, a line chart is plotted between Year and the average success rate to understand the launch success trend.

- https://github.com/Ryzen17/My_rep/blob/main/edadataviz.ipynb

EDA with SQL

- Queries for the following requests were created:
  - To display the names of the unique launch sites in the space mission
  - To display 5 records where launch sites begin with the string 'CCA'
  - To display the total payload mass carried by boosters launched by NASA (CRS)
  - To display average payload mass carried by booster version F9 v1.1
  - To list the date when the first successful landing outcome in ground pad was achieved

- To list the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000.

- To list the total number of successful and failure mission outcomes

- To list all the booster_versions that have carried the maximum payload mass

- To 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.

- To 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

- https://github.com/Ryzen17/My_rep/blob/main/eda_with_sql.ipynb

Build an Interactive Map with Folium

- A blue circle popup at NASA Johnson Space Center's coordinate with a label showing its name is created

- A red circle marker is then added for each launch site in the data frame launch_sites.

- The launch success rate may also depend on the location and proximities of a launch site, in addition to payload mass, orbit, etc. Hopefully we could discover some of the factors by analyzing the existing launch site locations.

- Next, markers are created for all launch records- green marker for a successful launch and a red marker for an unsuccessful launch. Marker clusters are used to simplify the map.

- Finally, Distance and line markers are also added to indicate the distance between the launch site and the nearest coastline, highway and railroad.

- https://github.com/Ryzen17/My_rep/blob/main/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- A Plotly Dash application is built for users to perform interactive visual analytics on SpaceX launch data in real-time.

- This dashboard application contains input components such as a dropdown list and a range slider to interact with a pie chart and a scatter point chart.

- Using the dropdown, it is possible to see which launch site has the largest success count. It is also possible to select one specific site and check its detailed success rate (class=0 vs. class=1).

- The general idea of this callback function is to get the selected launch site from site-dropdown and render a pie chart visualizing launch success counts.

- Next, to check the correlation between payload and mission outcome, a scatter plot between payload and launch outcome is created. The range slider allows us to visually observe how payload mass may be correlated with mission outcomes for all sites.

- It is also color-labeled with the Booster versions so that we may observe mission outcomes with different boosters.

- https://github.com/Ryzen17/My_rep/blob/main/spacex-dash-app.py

Predictive Analysis (Classification)

- The data is loaded into a dataframe from the csv file given in the link in the notebook.

- Next, a NumPy array is created from the column Class in data, by applying the method to_numpy() and it is assigned to the variable Y.

```
Y=data['Class'].to_numpy()
```
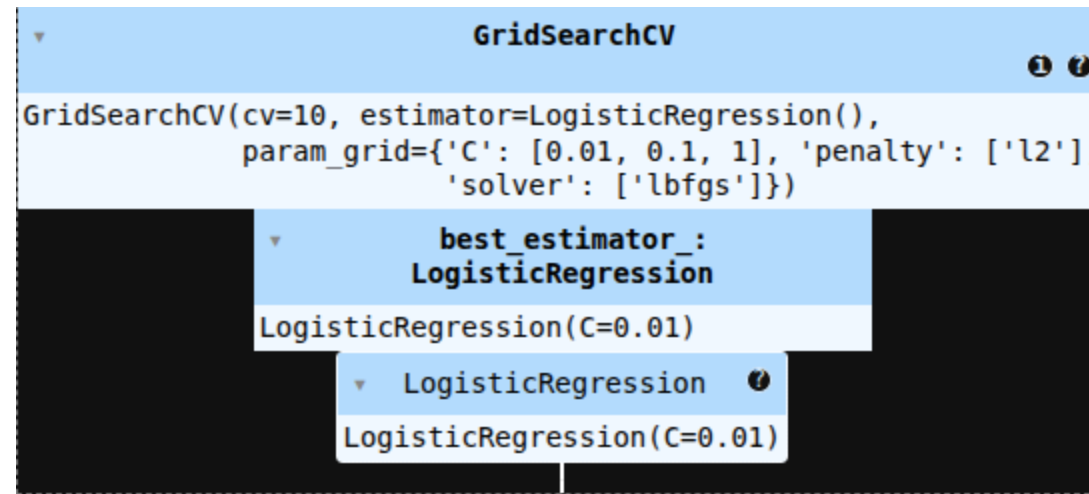
- The data in X is standardized and then reassigned to the variable X

```
transform = preprocessing.StandardScaler()
X=transform.fit_transform(X)
```

- Then, the data is split into training and testing data using the function train_test_split. The training data is divided into validation data, a second set used for training data; then the models are trained and hyperparameters are selected using the function GridSearchCV.

X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=2)

- Next, a logistic regression object is created and using this , GridSearchCV object logreg_cv with cv = 10 is also created. This is fitted with training data to find the best parameters from the dictionary 'parameters'.

- Also, the best parameters is displayed using the data attribute best_params_ .


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

- The accuracy on the validation data is estimated using the data attribute best_score_. Finally, the accuracy on the test data is calculated using the method score . The value obtained is accuracy_logreg- 0.8333333333333334
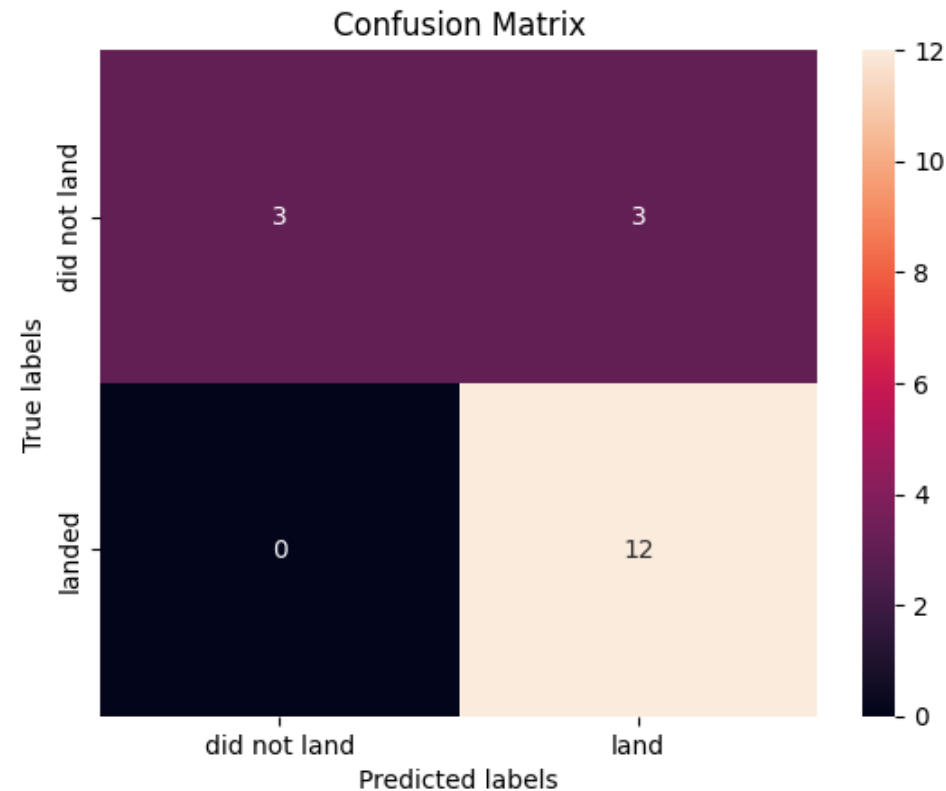
- Finally, a Confusion Matrix is plotted.
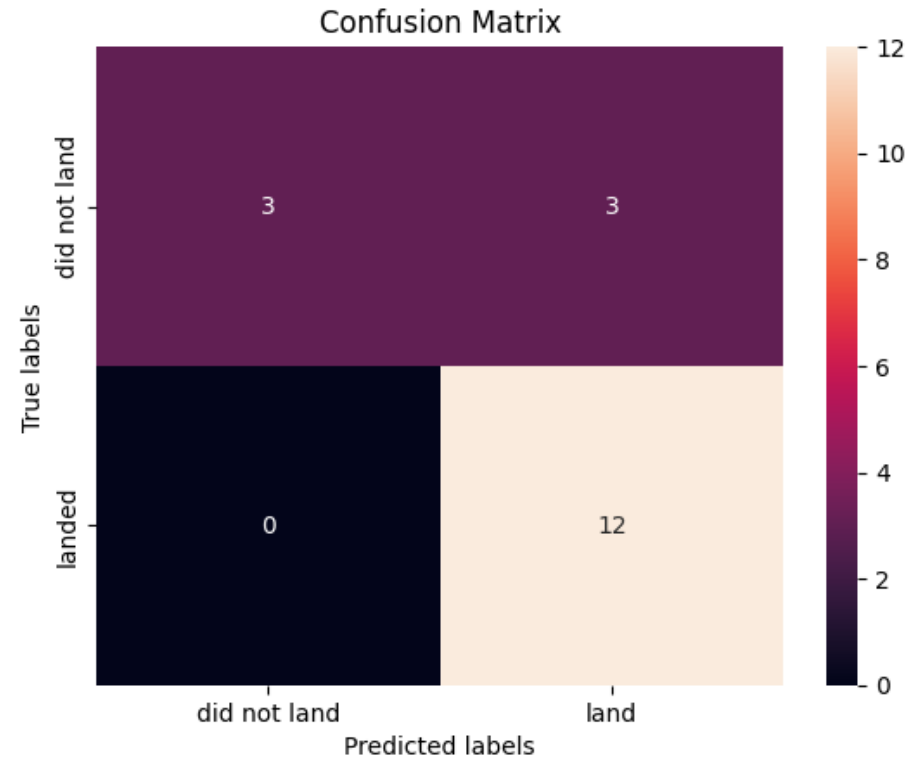
- We can see that logistic regression can distinguish between the different classes, however, the problem is false positives.
    - o True Postive - 12 (True label is landed, Predicted label is also landed)
    - o False Postive - 3 (True label is not landed, Predicted label is landed)

- Similarly, a support vector machine object is created and using this a GridSearchCV object svm_cv with cv = 10 is also created. This is fitted with the training data to find the best parameters from the dictionary 'parameters'.



```
                                    GridSearchCV                              ❶ ❷

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')})
                              best_estimator_: SVC
    SVC(C=np.float64(1.0), gamma=np.float64(0.03162277660168379), kernel='sigmoid')
                                    SVC                                        ❷
    SVC(C=np.float64(1.0), gamma=np.float64(0.03162277660168379), kernel='sigmoid')
```

- The accuracy on the validation data is estimated using the data attribute best_score_. Finally, the accuracy on the test data is calculated using the method score . The value obtained is accuracy_svm:- 0.8333333333333334

- Finally, a Confusion Matrix is plotted.

- We can see that, here again, the SVM model can distinguish between the different classes, however, the problem is false positives.
    - o True Postive - 12 (True label is landed, Predicted label is also landed)
    - o False Postive - 3 (True label is not landed, Predicted label is landed)

- Next, a decision tree classifier object is created and using this a GridSearchCV object tree_cv with cv = 10 is created. This is fitted with the training data to find the best parameters from the dictionary 'parameters'.

```
                              GridSearchCV                          ⓘ ⓘ

GridSearchCV(cv=10, estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [2, 4, 6, 8, 10, 12, 14, 16, 18],
                         'max_features': ['sqrt', None],
                         'min_samples_leaf': [1, 2, 4],
                         'min_samples_split': [2, 5, 10],
                         'splitter': ['best', 'random']})

                 best_estimator_: DecisionTreeClassifier

DecisionTreeClassifier(criterion='entropy', max_depth=6, max_features='sqrt',
                       min_samples_split=10)

                          DecisionTreeClassifier                         ⓘ

DecisionTreeClassifier(criterion='entropy', max_depth=6, max_features='sqrt',
                       min_samples_split=10)
```
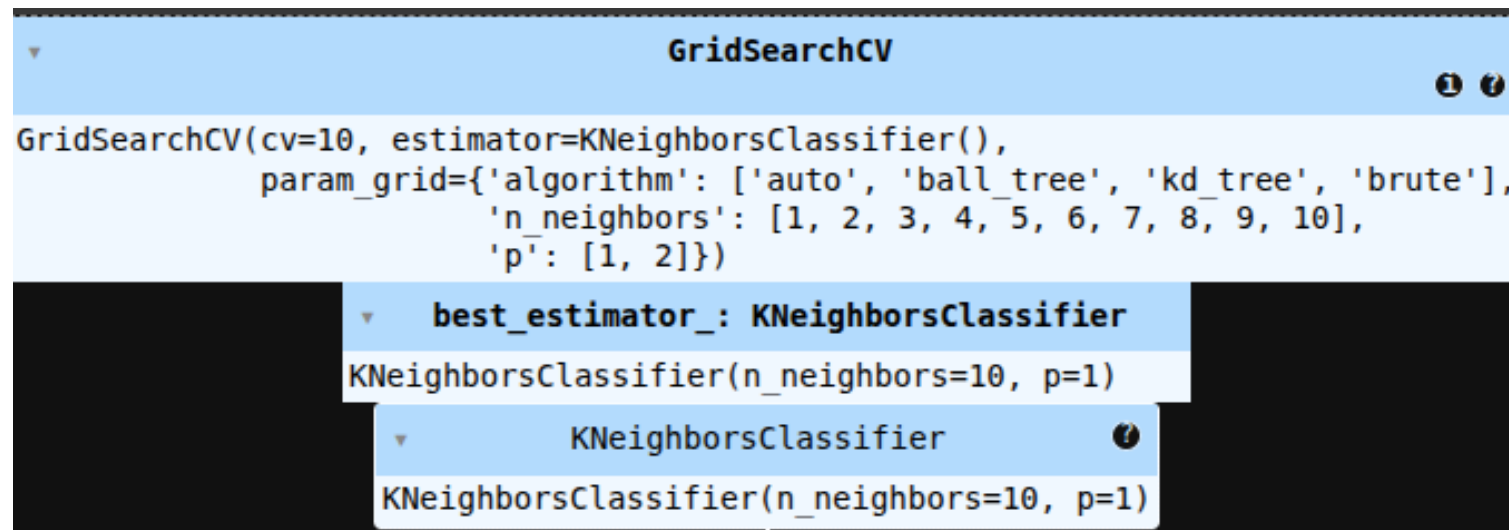
- The accuracy on the validation data is estimated using the data attribute best_score_. Finally, the accuracy on the test data is calculated using the method score . The value obtained is accuracy_tree: 0.94

- Finally, a Confusion Matrix is plotted.
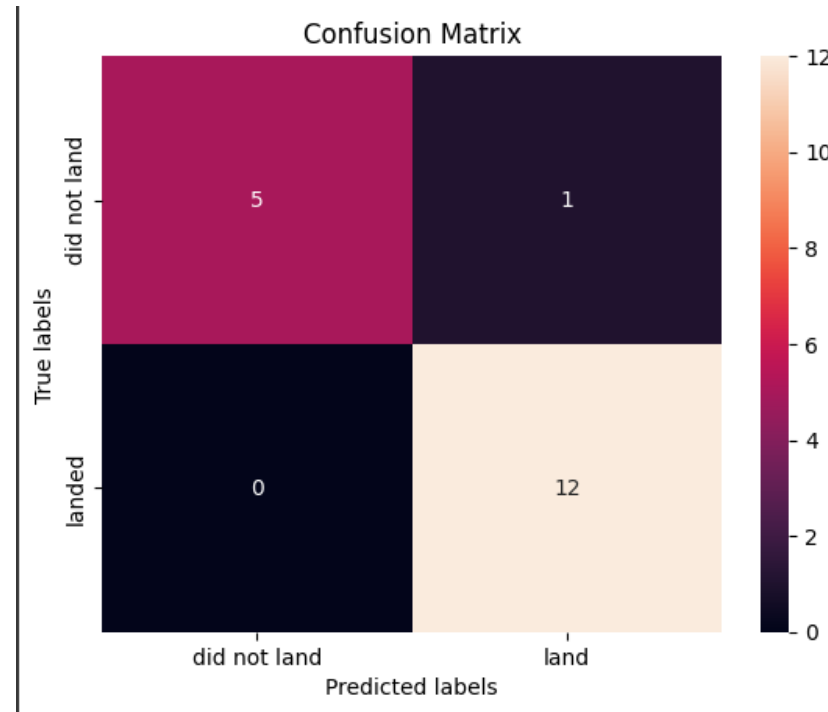
- But in case of Decision Tree, the number of false positives has reduced to 1 from the earlier 3 for other models.
  - True Postive - 12 (True label is landed, Predicted label is also landed)
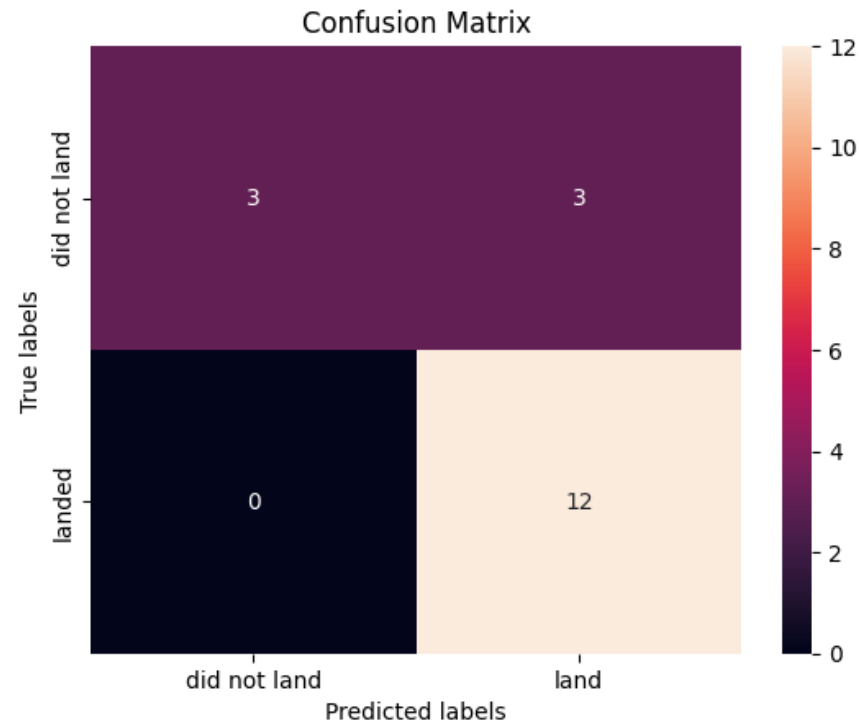 - False Postive - 1 (True label is not landed, Predicted label is landed)

- Finally, a k nearest neighbors object is created and using this a GridSearchCV object knn_cv with cv = 10 is created. This is fitted with the training data to find the best parameters from the dictionary 'parameters'.

```
GridSearchCV
                                              ❶ ❷

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]})

           best_estimator_: KNeighborsClassifier

KNeighborsClassifier(n_neighbors=10, p=1)

                  KNeighborsClassifier            ❷

KNeighborsClassifier(n_neighbors=10, p=1)
```

- The accuracy on the validation data is estimated using the data attribute best_score_. Finally, the accuracy on the test data is calculated using the method score . The value obtained is accuracy_knn:- 0.8333333333333334

- Finally, a Confusion Matrix is plotted.

Confusion Matrix

- We can see that, here again, the knn model can distinguish between the different classes, however, the problem is false positives.

- True Postive - 12 (True label is landed, Predicted label is also landed)

- False Postive - 3 (True label is not landed, Predicted label is landed)

```
Create a NumPy array from the
column Class and then assign it
to variable Y.
```

$\downarrow$

```
Standardize the rest of the
variables and assign it to
variable X
```

$\downarrow$

```
Split the data into training and
testing set using
rain_test_split()
```

$\downarrow$

https://github.com/Ryzen17/My_rep/blob/main/SpaceX_Machine_Learning_Prediction.ipynb

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

$\rightarrow$

```
Display the best parameters
using best_params_ and the
accuracy on the validation data
using best_score_. Calculate
the accuracy on the test data
using the method score
```

$\rightarrow$

```
Repeat the steps for Support
Vector Machine (SVM),
Decision Tree Classifier and K-
Nearest Neighbours (KNN) to
find the best model.
```

Results

- Exploratory Data analysis was carried out with visualization and sql queries

- Visualization included plots on launch site, payload mass, orbit etc., to understand how these parameters affected the outcome of the launch

- Using SQL queries, distinct launch sites, average payload mass, number of successful etc., were identified

- Launch outcomes were added on folium maps to identify which launch site had the maximum successful launches. Also, proximities to nearest infrastructure like highway, railroad etc were identified.

- Plotly dash apps were also created to identify the launch site, booster version, payload range that had the maximum number of successful launches.

- For classification, different models like Logistic Regression, SVM, Decision Tree and KNN were experimented upon to find the best model which gave the highest accuracy. Also, confusion matrices were plotted in each case.

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

- From the plot, it is clear that, as the flight number increases, so does the success rate.

- Also, the launch site CCAFS SLC 40 has the maximum number of successful flights. VAFB SLC 4E has the least number of unsuccessful flights.
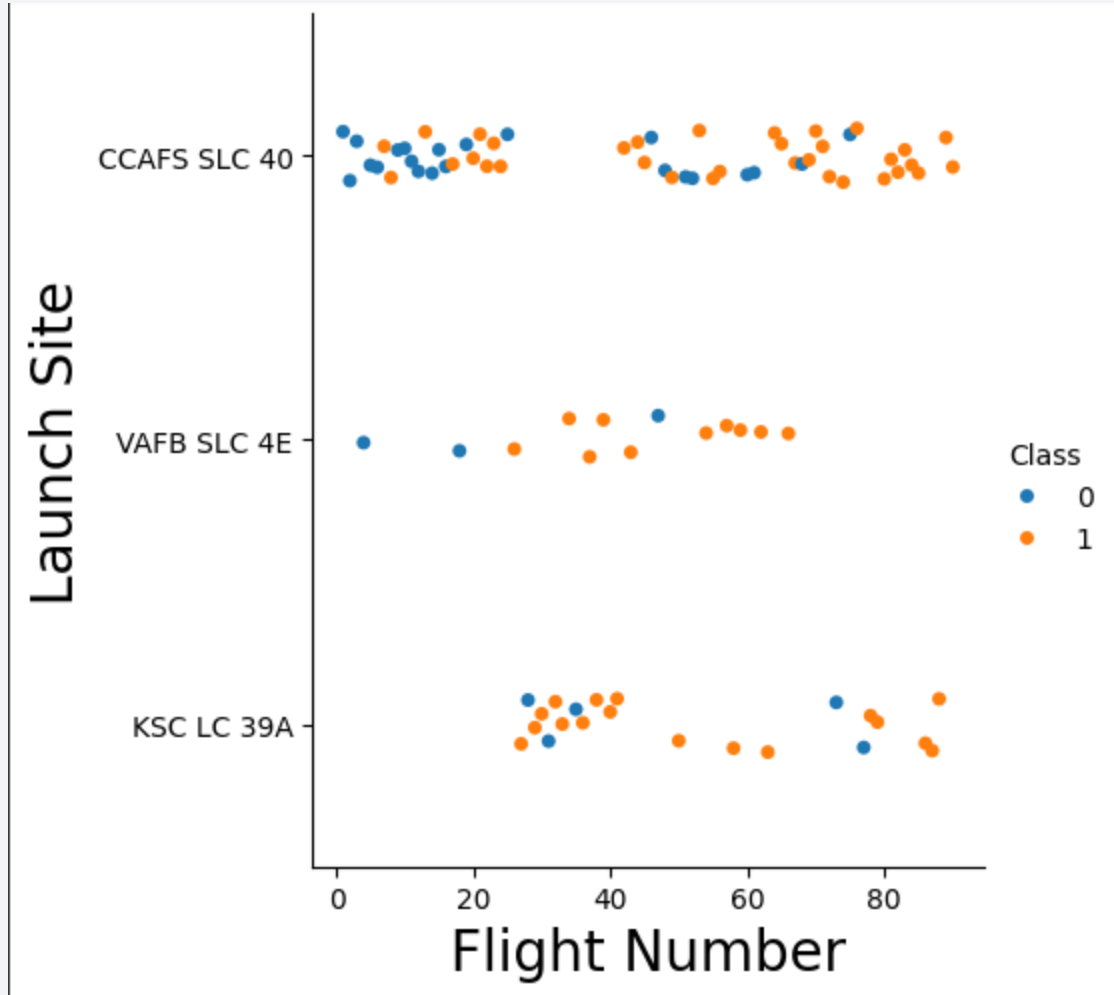
Payload vs. Launch Site
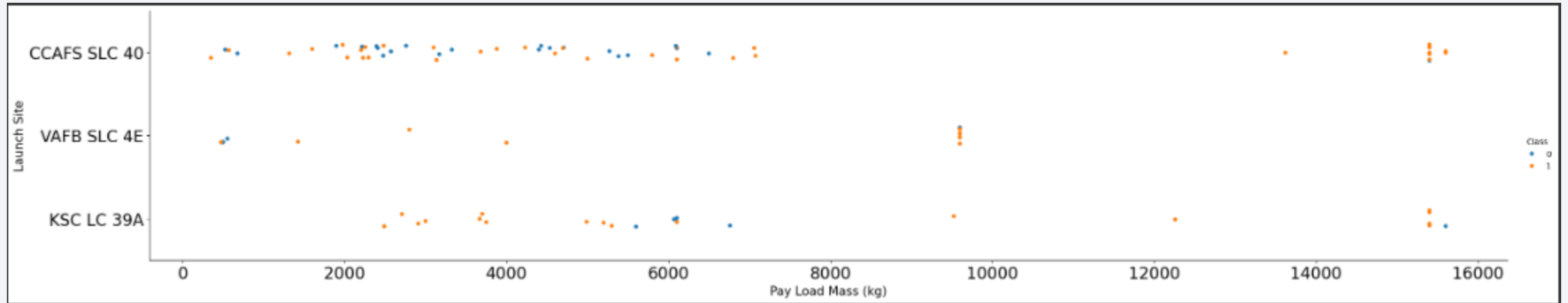
- From the above scatter plot, we can infer that in case of VAFB-SLC launchsite, there are no rockets launched for heavypayload mass(greater than 10000)

- KSC LC-39A has higher success rates along the payload mass range

Success Rate vs. Orbit Type

- From this bar chart, we can infer that orbits such as LEO, GTO, SO and ISS have the highest success rates
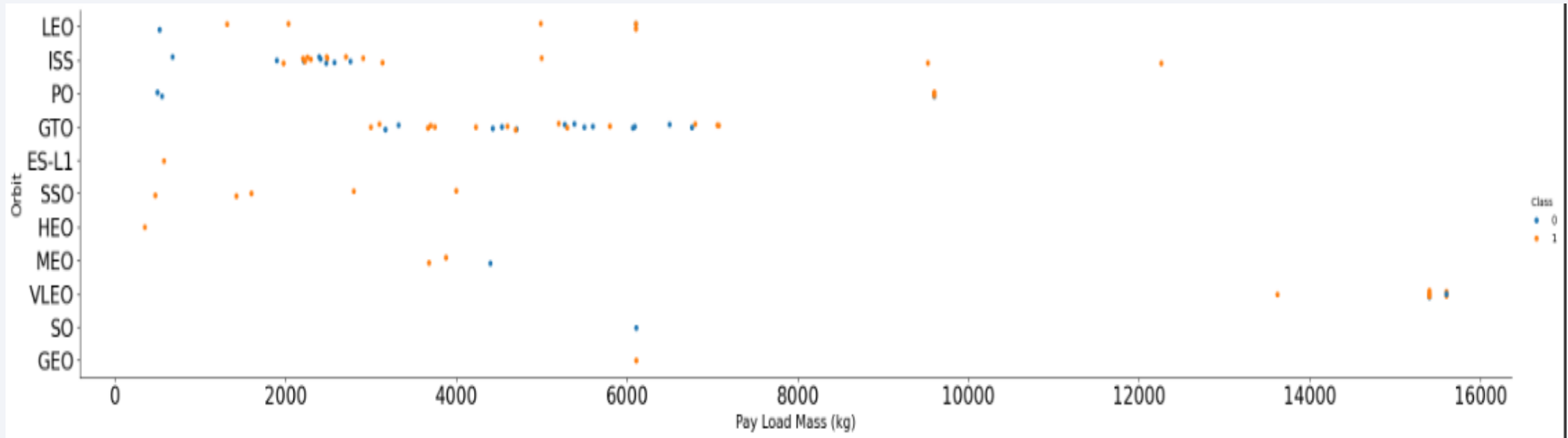
- PO has the lowest success rate
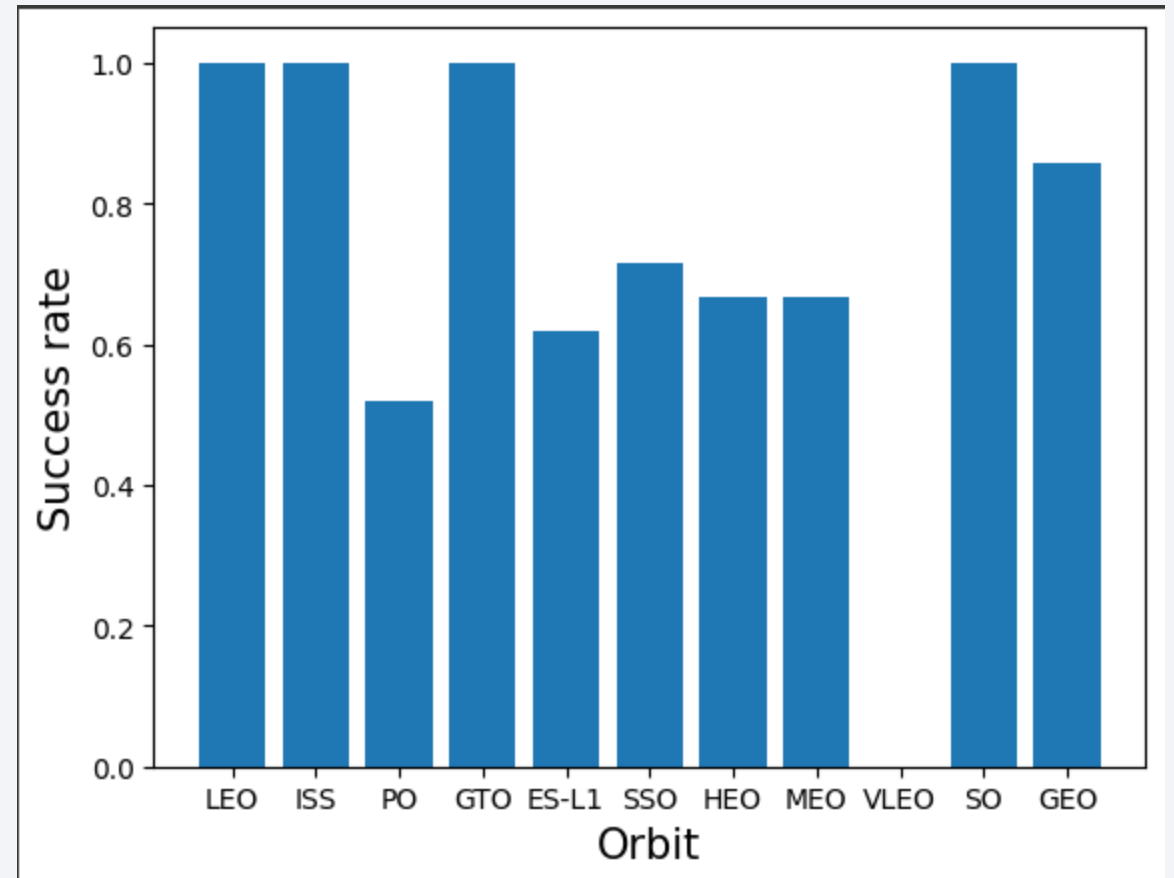
Flight Number vs. Orbit Type
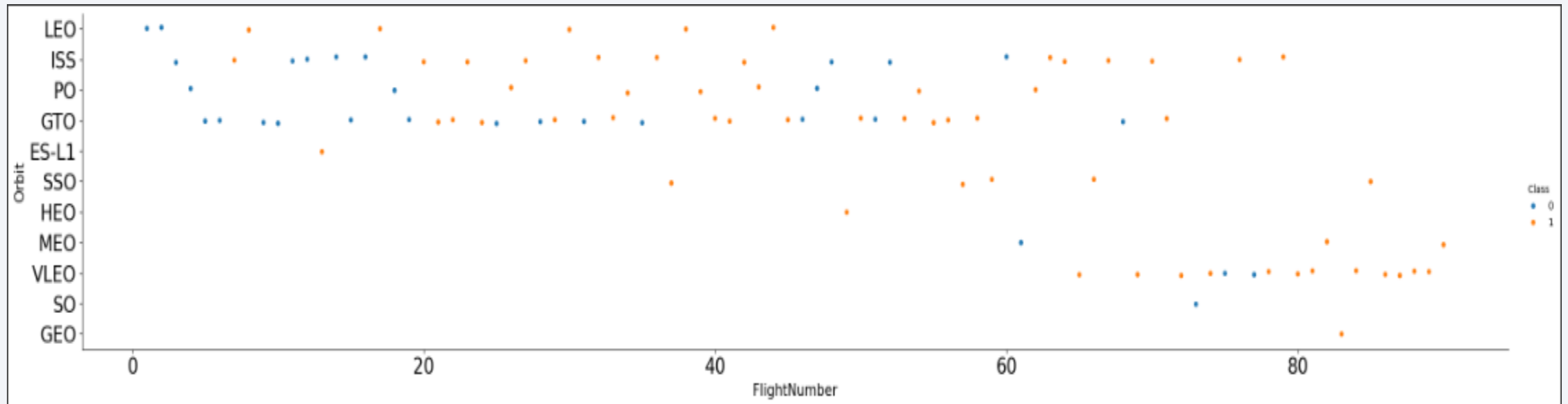
- In scatter plot between FlightNumber and Orbit type, it is observed that in the LEO orbit, success seems to be related to the number of flights. Conversely, in the GTO orbit, there appears to be no relationship between flight number and success

Payload vs. Orbit Type

- In the scatter plot between Payload Mass and Orbit, it is seen that with heavy payloads, the successful landing rates are more for Polar, LEO and ISS. However, for GTO, it is difficult to distinguish between successful and unsuccessful landings as both outcomes are present

Launch Success Yearly Trend

- In the line chart between Year and average success rate, it is clear that the success rate since 2013 kept increasing till 2020.

- However, the year 2018 was a particularly unsuccessful year.
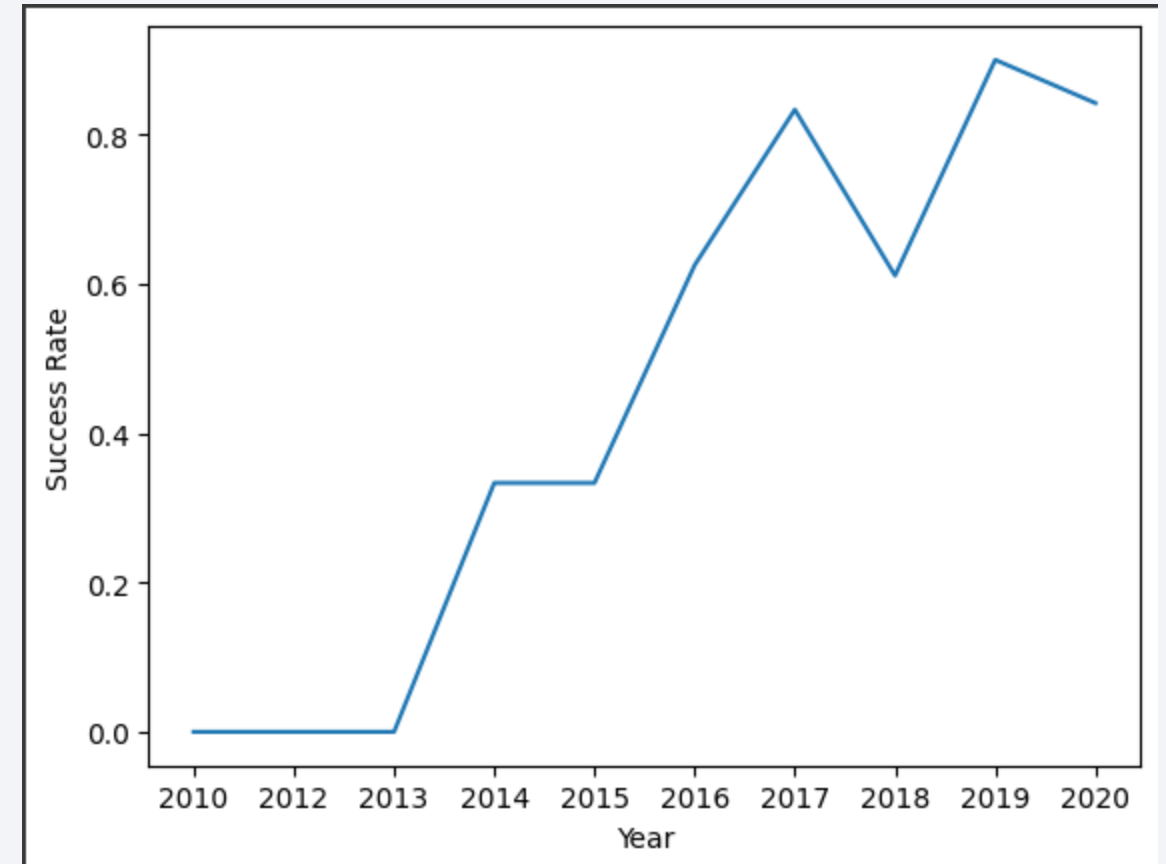
All Launch Site Names

- For the query to find the names of the unique launch sites, the keyword 'DISTINCT' is used

```
[ ] %sql Select DISTINCT Launch_Site from SPACEXTBL

    * sqlite:///my_data1.db
Done.
 Launch_Site
CCAFS LC-40
VAFB SLC-4E
KSC LC-39A
CCAFS SLC-40
```

Launch Site Names that Begin with 'CCA'


```
%sql Select * from SPACEXTBL where Launch_Site like 'CCA%' LIMIT 5
```

* sqlite:///my_data1.db
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

- To find 5 records where launch sites begin with 'CCA", we use the operator 'like'.

Total Payload Mass

```
%sql Select SUM(PAYLOAD_MASS__KG_) from SPACEXTBL where Customer='NASA (CRS)'

 * sqlite:///my_data1.db
Done.
SUM(PAYLOAD_MASS__KG_)
 45596
```

- For the query to calculate the total payload carried by boosters from NASA, the function SUM() is used along with the comparison operator '='.

Average Payload Mass by F9 v1.1

```
%sql Select AVG(PAYLOAD_MASS__KG_) from SPACEXTBL where Booster_Version='F9 v1.1'

 * sqlite:///my_data1.db
Done.
AVG(PAYLOAD_MASS__KG_)
2928.4
```

- To calculate the average payload mass carried by booster version F9 v1.1, we use the function AVG() and the comparison operator '='.

First Successful Ground Landing Date

```
%sql Select min(Date) from SPACEXTBL where Landing_Outcome='Success (ground pad)'

 * sqlite:///my_data1.db
Done.
 min(Date)
 2015-12-22
```

- For the query to find the date of the first successful landing outcome on ground pad, the function min() is used along with the comparison operator '='

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql Select Booster_Version from SPACEXTBL where Landing_Outcome='Success (drone ship)' and PAYLOAD_MASS__KG_ between 4000 and 6000

 * sqlite:///my_data1.db
Done.
Booster_Version
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2
```

- To list the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000, we use the operators 'between' and 'and'.

Total Number of Successful and Failure Mission Outcomes

```
%sql Select Mission_Outcome, COUNT(*) from SPACEXTBL GROUP BY Mission_Outcome

 * sqlite:///my_data1.db
Done.
      Mission_Outcome        COUNT(*)
Failure (in flight)          1
Success                      98
Success                      1
Success (payload status unclear) 1
```

- For the query to calculate the total number of successful and failure mission outcomes, we use the operator 'COUNT' and the clause 'GROUP BY'.

Boosters Carried Maximum Payload

```
%sql Select Booster_Version from SPACEXTBL where PAYLOAD_MASS__KG_=(Select MAX(PAYLOAD_MASS__KG_) from SPACEXTBL)

 * sqlite:///my_data1.db
Done.
Booster_Version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7
```

- To list the names of the booster which have carried the maximum payload mass, we use the function MAX() inside a subquery.

2015 Launch Records

```
%sql Select substr(Date,6,2), Landing_Outcome, Booster_Version, Launch_Site
from SPACEXTBL where Landing_Outcome='Failure (drone ship)'
and substr(Date,0,5)='2015'
```

* sqlite:///my_data1.db
Done.

substr(Date,6,2)	Landing_Outcome	Booster_Version	Launch_Site
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

- For the query to list the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015, we use the operators '=' and 'and'

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

```
%sql Select Landing_Outcome,COUNT(*) from SPACEXTBL
where Date between '2010-06-04' and '2017-03-20'
GROUP BY Landing_Outcome order by count(*) desc
```

* sqlite:///my_data1.db
Done.

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

- To 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, we use the operator 'COUNT', the clause 'GROUP BY' and the keyword 'desc'

Section 3

Launch Sites Proximities Analysis

Marking launch sites

- All launch sites are in close proximity to the equator. This is because, at the equator, Earth has maximum rotational velocity and this gives an initial boost to the rockets, called the 'slingshot effect'. This corresponds to less fuel consumption to reach the 'escape velocity' and possibility of carrying heavier loads.

- Also, the launch sites are near the coasts, so that any unsuccessful launches do not cause any casualties and the debris may fall safely into the ocean. Also, it is ideal for transporting massive parts of the rockets through ships.

Adding launch outcomes
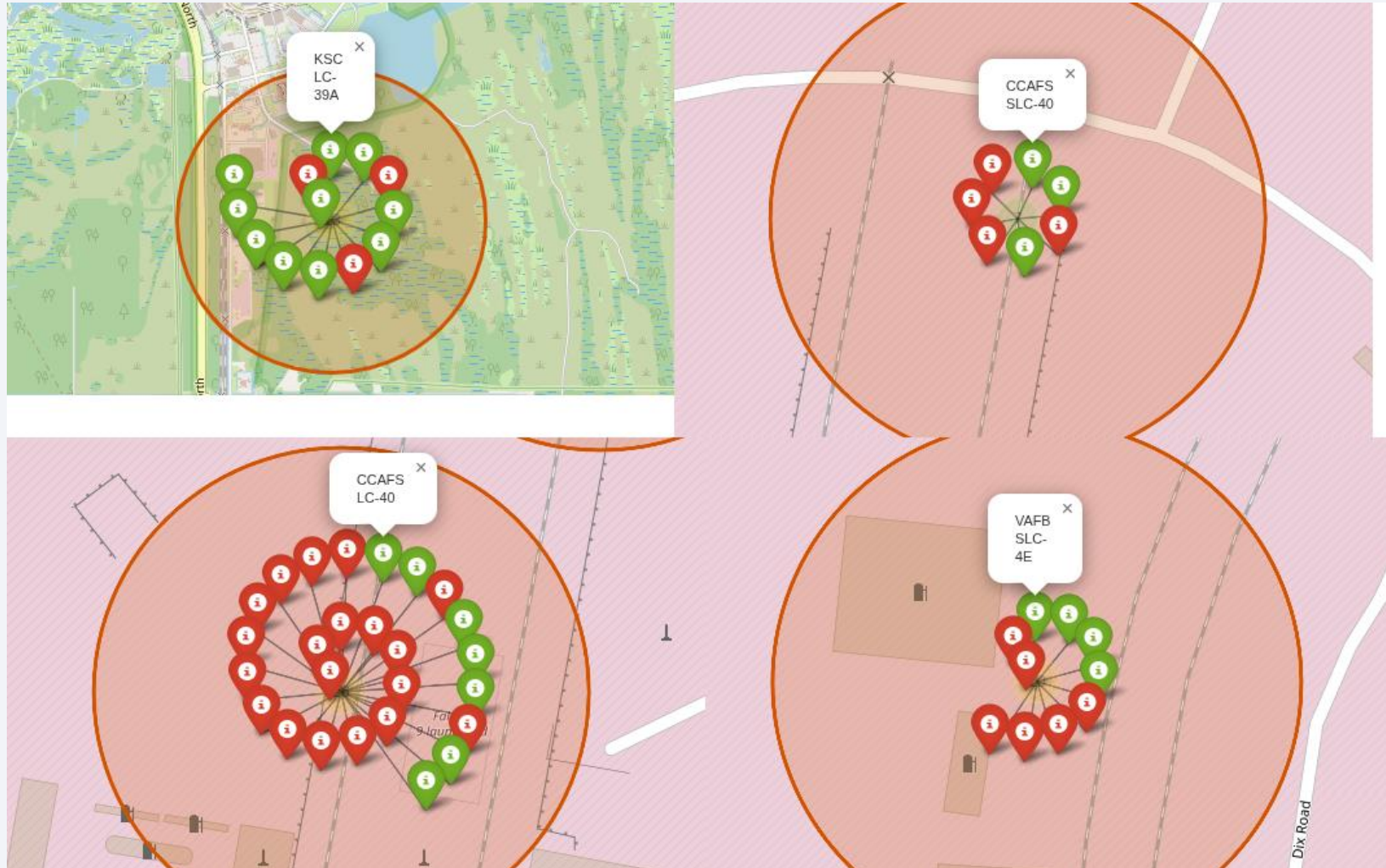
- Markers are created for all launch records. A green marker is used for a successful launch whereas a red one is used for an unsuccessful launch

- Since there are only four launch sites, marker clusters are used to simplify the map containing many markers having the same coordinates

- From the map, it is clear that KSC LC-39A has the maximum number of successful launches, followed by CCAFS LC-40 and VAFB SLC-4E. CCAFS SLC-40 has the least successful launches.

Proximity to the nearest coastline
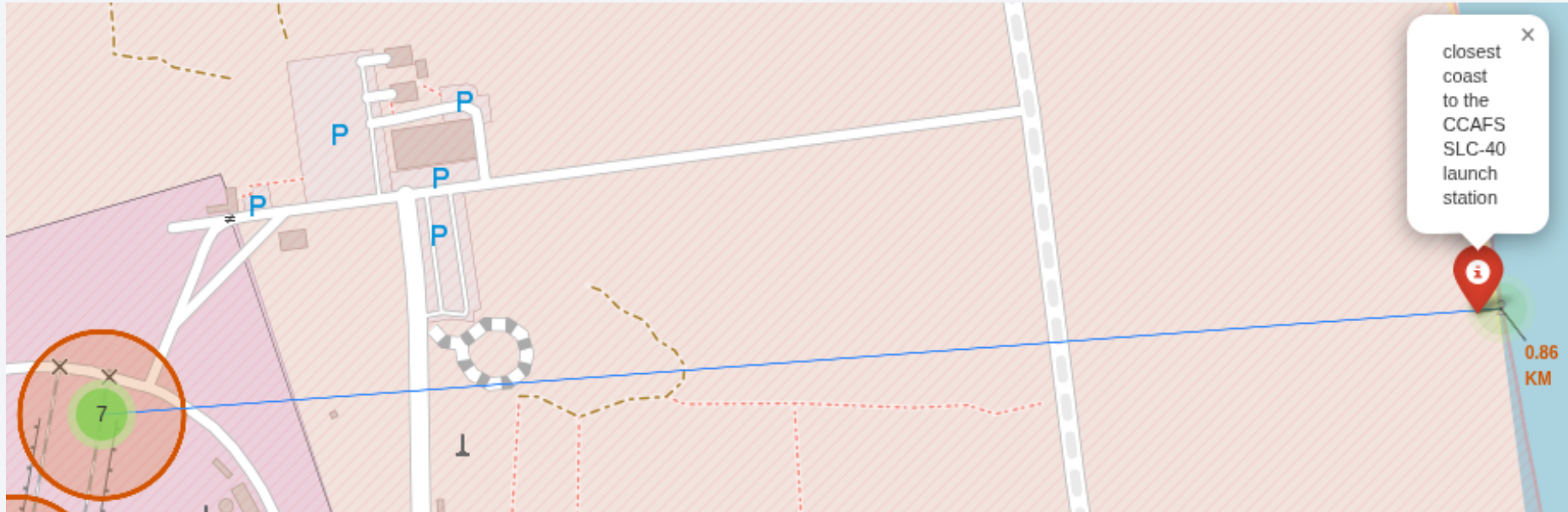
- the distance between the launch site CCAFS SLC-40 and the nearest coastline is calculated and marked. In this case, it is 0.86 km

- It is clear that the launch site is located close to coast so that any unsuccessful launch will not lead to any casualties and the debris can fall safely into the ocean. Also, it is easier to transport the bulky rocket and/or its parts through the ports

Proximity to the nearest highway
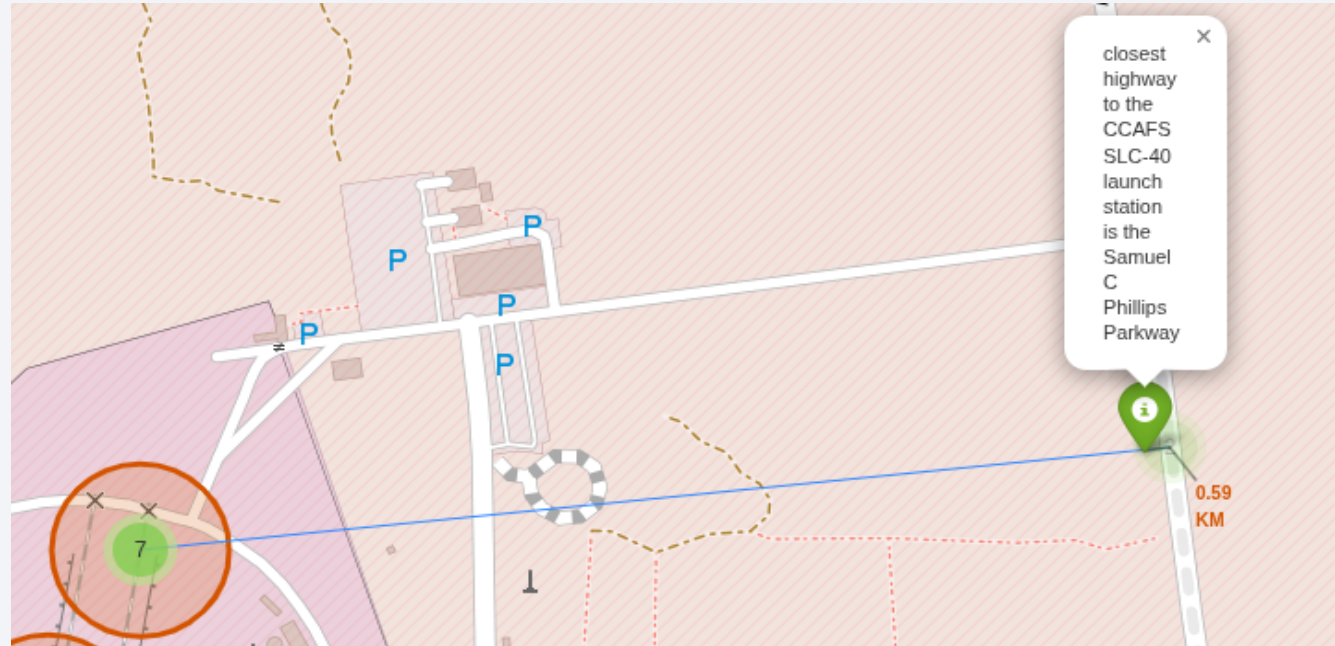
- the distance between the launch site CCAFS SLC-40 and the nearest highway is calculated and marked. In this case, the nearest highway is Samuel C Phillips Parkway and the distance is 0.59 km

- It is clear that the launch site is located in close proximity to the highway for ease of transportation of the rocket or its parts

Proximity to the nearest railroad

- the distance between the launch site CCAFS SLC-40 and the nearest railroad is calculated and marked. In this case, the nearest railroad is Florida East coast railway and the distance is 22.24 km.

- It is clear that the launch site is located close to the railroad for ease of transportation of parts of the rocket
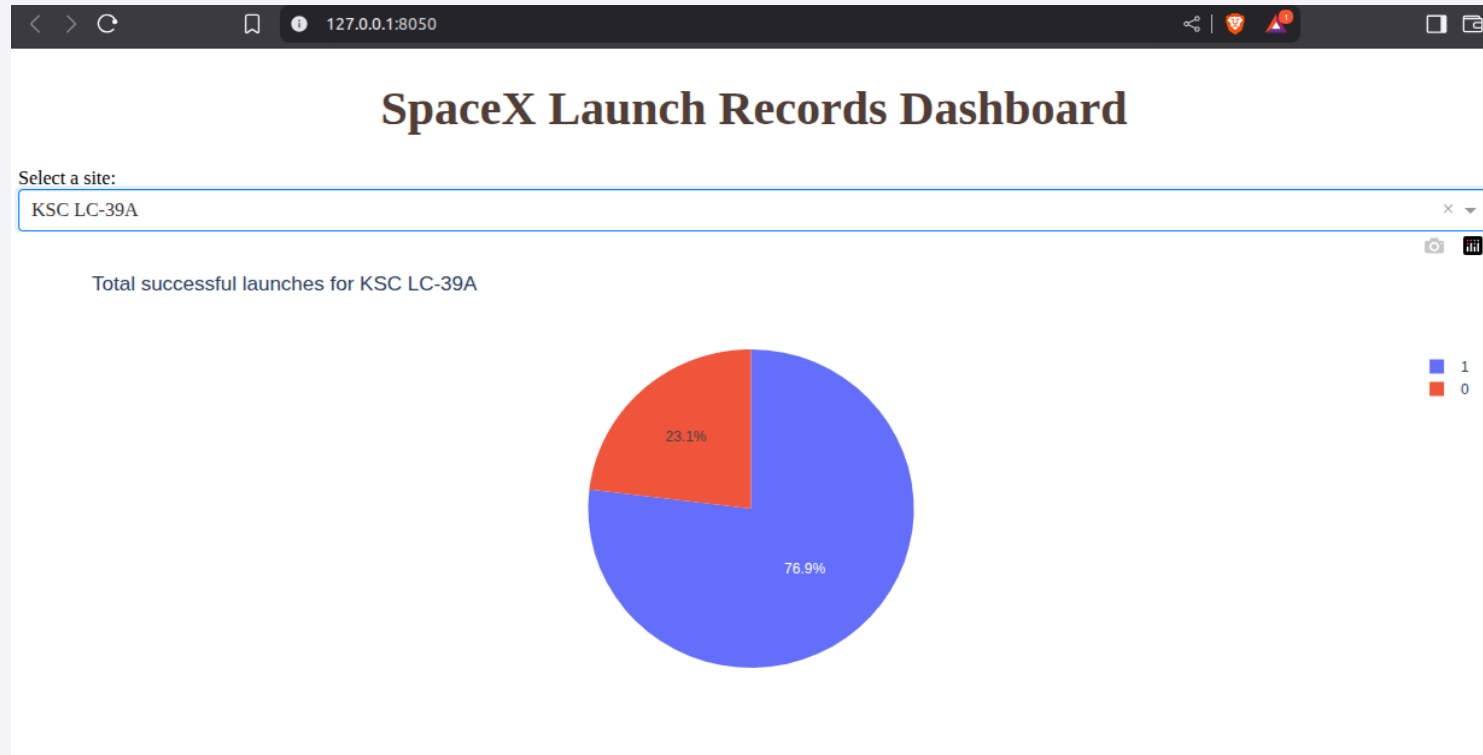
Build a Dashboard with Plotly Dash

Successful launches
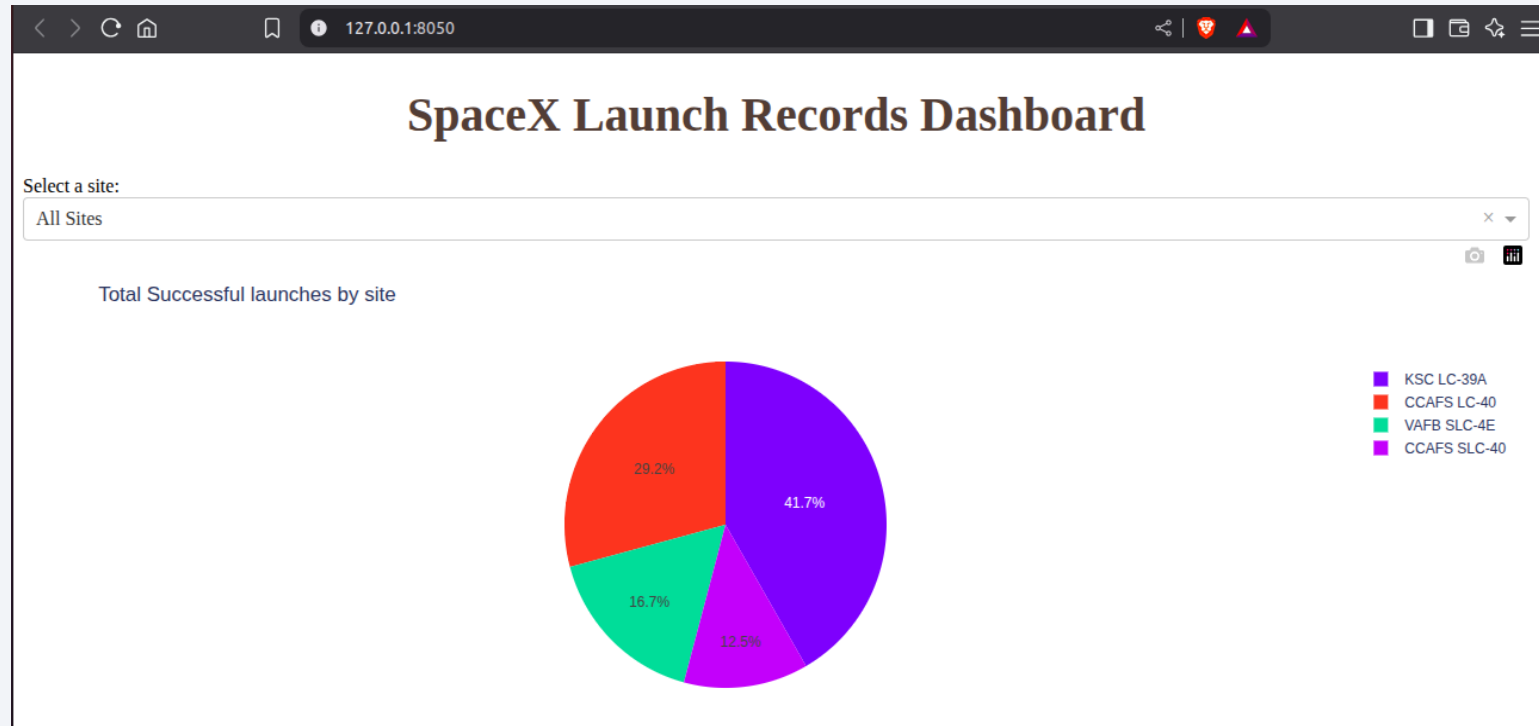
- From the pie-chart, it is clear that the launch site KSC LC-39A has the highest proportion of successful launches, followed by CCAFS LC-40 and VAFB SLC-4E. CCAFS SLC-40 has the least number of successful launches

Maximum successful launches

- The launch site KSC LC-39A has the highest proportion of successful launches.
- It has more than 75% success rate.

Relation between Payload range and success rate
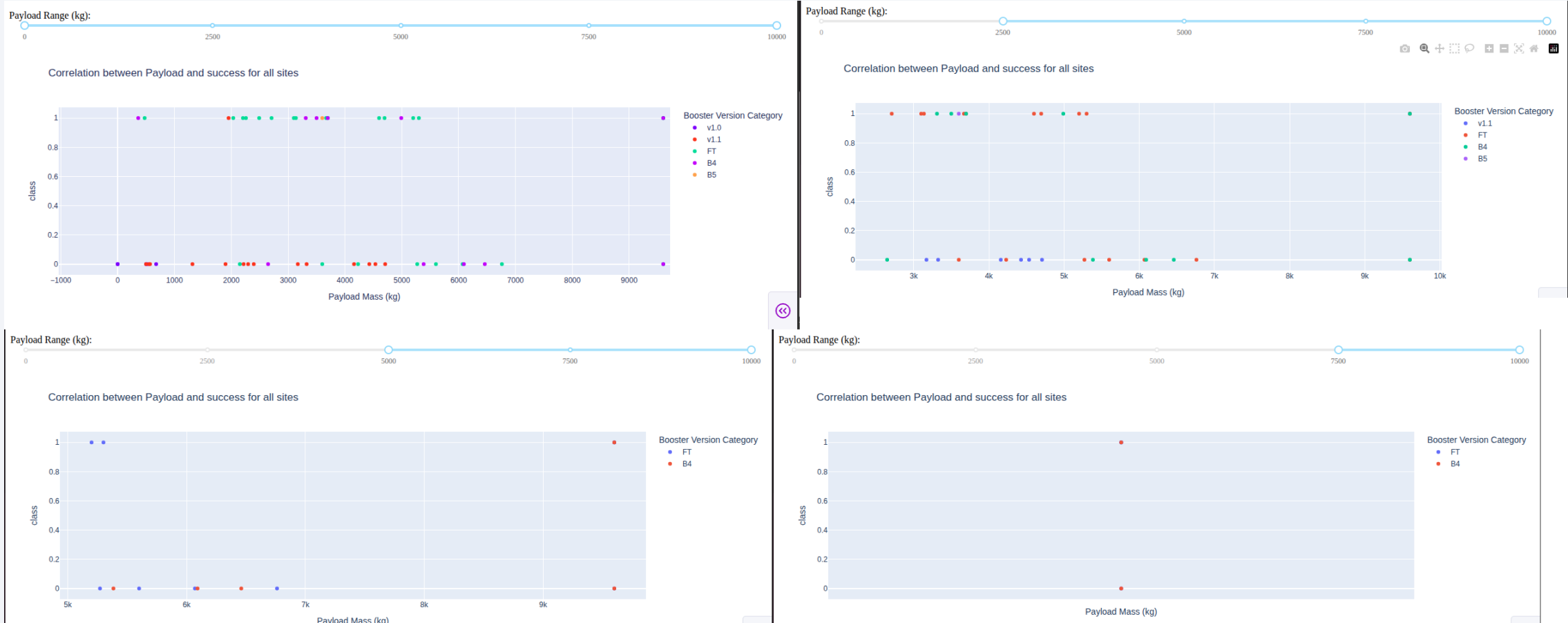
- From the scatter plot, we can infer that FT booster version of F9 has the highest success rate in the payload range<=2500 kg.
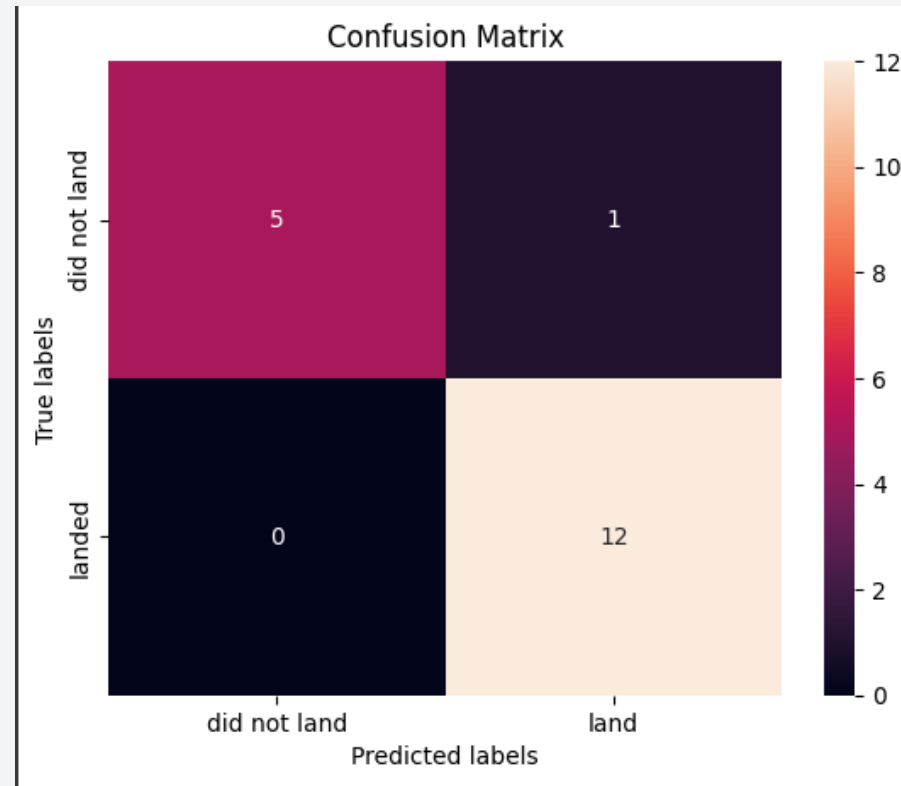
Section 5

Predictive Analysis (Classification)

Classification Accuracy

- A bar plot of accuracy for different models is plotted.

- It is clear that Decision Tree has the highest accuracy at around 94%.
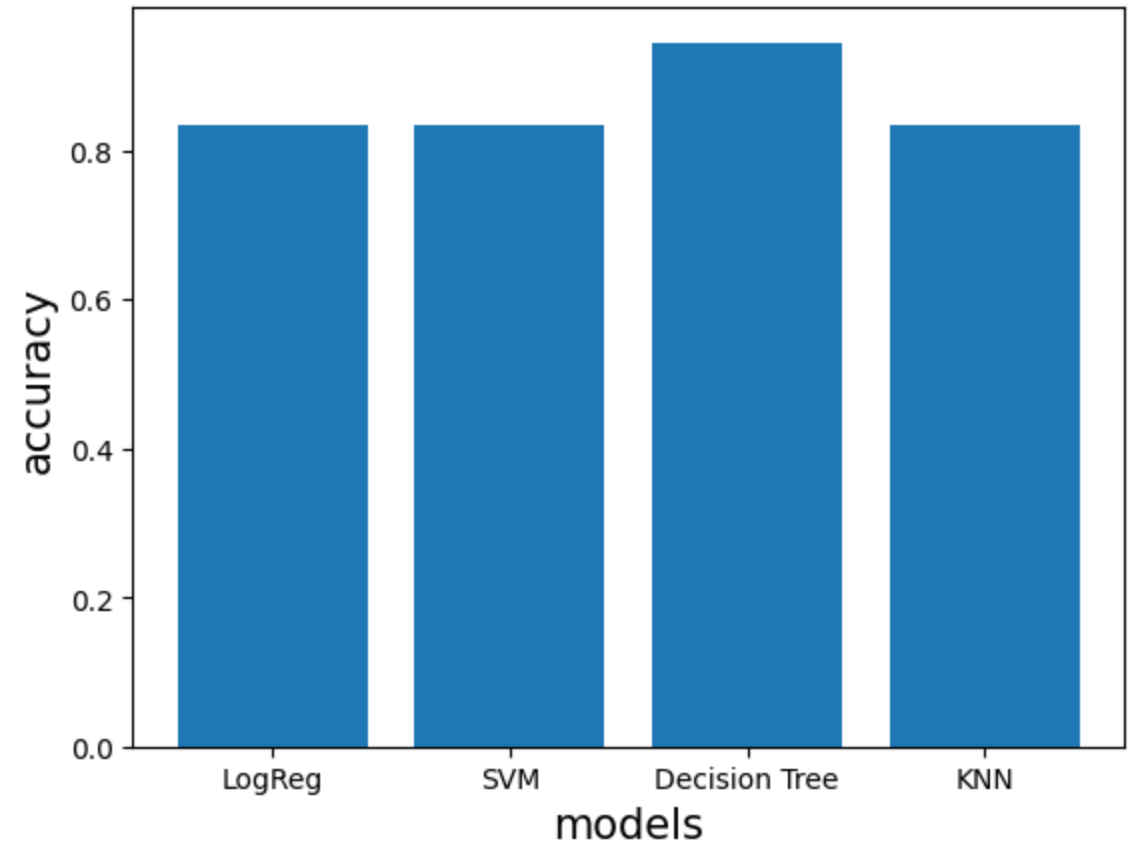
Confusion Matrix of the best model

- The best performing model is the Decision Tree Classifier with an accuracy score of 94%

- The Confusion matrix also shows fewer false positives (1) compared to the other models (3)

Conclusions

- In this capstone, we will predict if the Falcon 9 first stage will land successfully. If we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch

- From EDA using visualization, we can conclude that the launch site KSC LC-39A has highest success rates. Similarly, orbits such as LEO, GTO, SO and ISS have the highest success rates. Also, since 2013, there has been an increasing trend of successful launches

- During EDA with sql, we found out the the total payload carried by boosters from NASA as 45596 kg and the date of the first successful landing outcome on ground pad as 22/12/2015

- From the launch sites proximity analysis using folium, we can conclude that all the 4 launch sites are located near the equator to take advantage of the 'slingshot effect' provided by the higher rotational velocity of the earth at the equator. Also, the launch sites are located near the coasts to avoid any casualties resulting from unsuccessful launches.

- Moreover, launch sites are in close proximity to highways and railroads for ease of transportation of the rocket components.

- From the plotly dash application, it is clear that the launch site KSC LC-39A has the highest proportion of successful launches with more that 75% success rate. Using the range slider, we can conclude that the FT booster version of F9 has the highest success rate in the payload range <=2500 kg.

- Finally in predictive analysis, different models were used for classification like logistic regression, svm, decision tree and knn, after standardizing the data.

- In the end, Decision Tree gave the highest accuracy score of 94% and fewer false positives in the confusion matrix, making it the best model for classification.

- Moreover, after running multiple iterations during training of the models, Logistic Regression, SVM and KNN gave the same accuracy on the test data and confusion matrices. This is because these models are deterministic and gives the same coeff and hyperparameters for the same training data, unless the train_test_split is different.

- However, when multiple iteration of training of Decision Tree was carried out, accuracy scores on the test data were different- 0.833, 0.77, 0.67, 0.94 etc. This is because during training there is a hyperparameter called 'splitter' whose values include 'random'. Due to this, the split at each node is randomly selected from the best options, leading to different trees and consequently different predictions, accuracy scores and confusion matrices.

Appendix

- For predictive analysis, the target values were taken from: https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_2.csv

- Similarly, the features for classfication were taken from: https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_3.csv

Thank you!