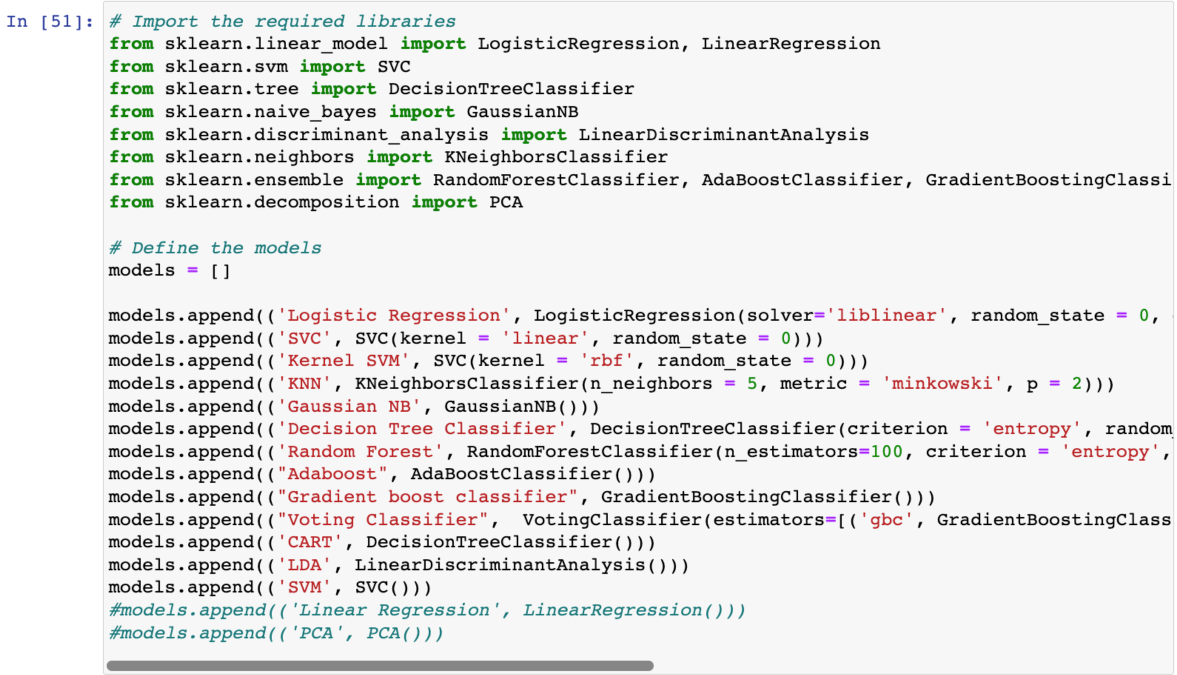
MODEL EXPLORATION AND SELECTION:

In the Screenshot below, the lines of code define a list of machine learning models that can be used for classification tasks. Each model is instantiated with various hyperparameters that have been chosen based on the problem domain and the dataset being used.



```
In [51]: # Import the required libraries
         from sklearn.linear_model import LogisticRegression, LinearRegression
         from sklearn.svm import SVC
         from sklearn.tree import DecisionTreeClassifier
         from sklearn.naive_bayes import GaussianNB
         from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
         from sklearn.neighbors import KNeighborsClassifier
         from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassi
         from sklearn.decomposition import PCA

         # Define the models
         models = []

         models.append(('Logistic Regression', LogisticRegression(solver='liblinear', random_state = 0,
         models.append(('SVC', SVC(kernel = 'linear', random_state = 0)))
         models.append(('Kernel SVM', SVC(kernel = 'rbf', random_state = 0)))
         models.append(('KNN', KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)))
         models.append(('Gaussian NB', GaussianNB()))
         models.append(('Decision Tree Classifier', DecisionTreeClassifier(criterion = 'entropy', random
         models.append(('Random Forest', RandomForestClassifier(n_estimators=100, criterion = 'entropy',
         models.append(("Adaboost", AdaBoostClassifier()))
         models.append(("Gradient boost classifier", GradientBoostingClassifier()))
         models.append(("Voting Classifier",  VotingClassifier(estimators=[('gbc', GradientBoostingClass
         models.append(('CART', DecisionTreeClassifier()))
         models.append(('LDA', LinearDiscriminantAnalysis()))
         models.append(('SVM', SVC()))
         #models.append(('Linear Regression', LinearRegression()))
         #models.append(('PCA', PCA()))
```

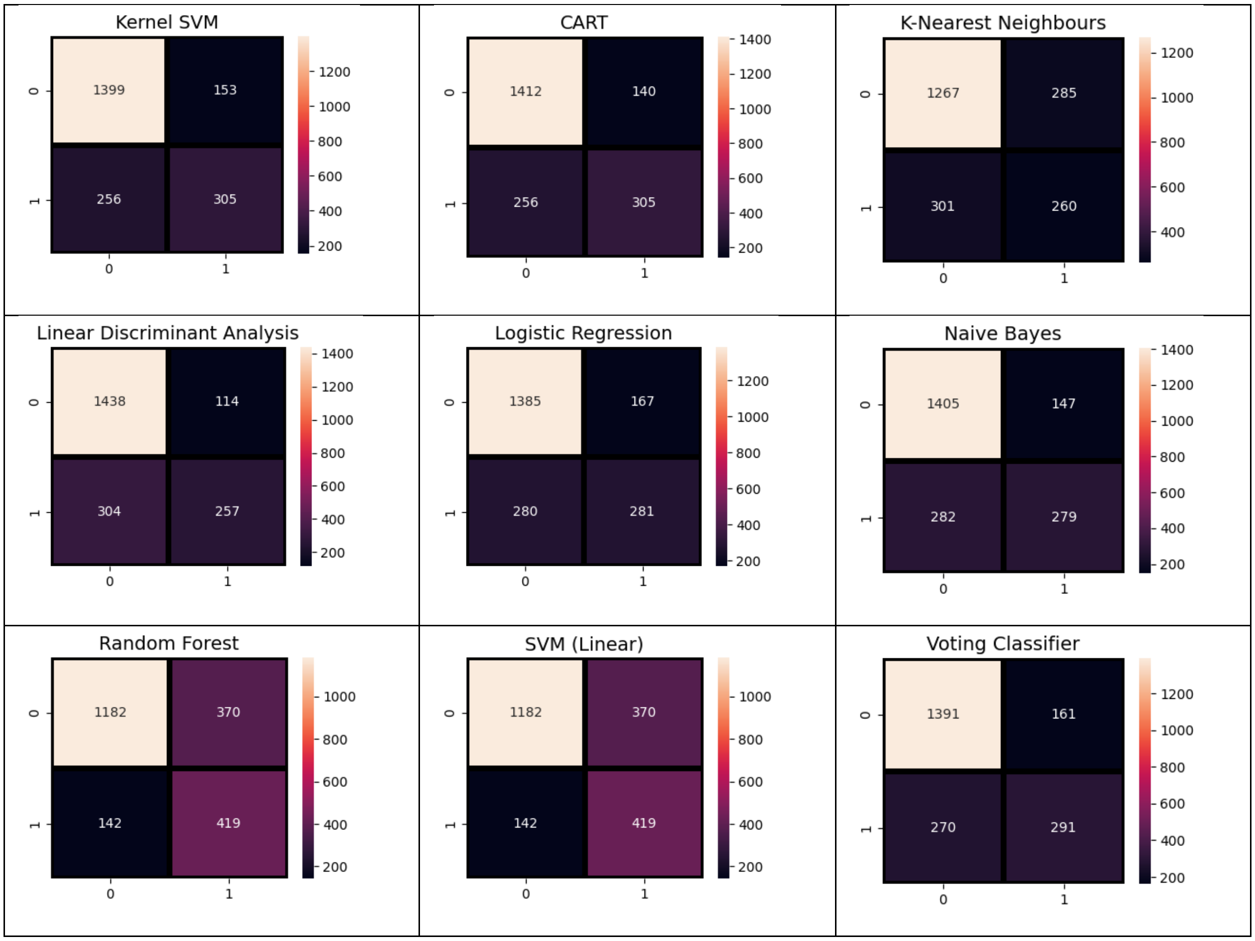
Furthermore, we perform k fold cross validation on the set of machine learning models using two different performance metrics, ROC AUC, and accuracy. The goal is to evacuate and compare the performance of these models on a given dataset.

In [52]:

```
acc_results =[]
auc_results =[]
names = []

result_col = ["Algorithm", "ROC AUC Mean", "ROC AUC STD", "Accuracy Mean", "Accuracy STD"]
model_results = pd.DataFrame(columns = result_col)

i=0
# K- fold cross validation

for name, model in models:
    names.append(name)
    kfold = model_selection.KFold(n_splits=10)

    cv_acc_results = model_selection.cross_val_score(model, X_train, y_train,
                    cv = kfold, scoring="accuracy")
    cv_auc_results = model_selection.cross_val_score(model, X_train, y_train,
                    cv = kfold, scoring="roc_auc")
    acc_results.append(cv_acc_results)
    auc_results.append(cv_auc_results)

    model_results.loc[i] = [name, 
                           round(cv_auc_results.mean()*100,2),
                           round(cv_auc_results.std()*100,2),
                           round(cv_acc_results.mean()*100,2),
                           round(cv_acc_results.std()*100,2)]
    i+=1

model_results.sort_values(by = ['ROC AUC Mean'], ascending=False)

print(model_results)
```

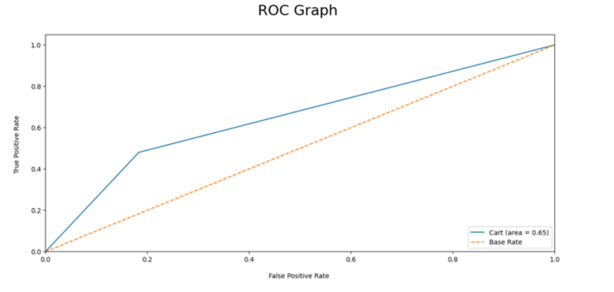
```
                    Algorithm  ROC AUC Mean  ROC AUC STD  Accuracy Mean  \
0         Logistic Regression         84.25         1.88          74.38   
1                         SVC         83.04         1.54          79.37   
2                  Kernel SVM         78.84         2.57          79.23   
3                         KNN         76.81         1.97          75.66   
4                 Gaussian NB         82.29         2.25          74.97   
5    Decision Tree Classifier         66.61         2.60          73.65   
6               Random Forest         82.32         2.40          78.56   
7                    Adaboost         84.16         1.94          79.80   
8   Gradient boost classifier         84.26         1.82          79.86   
9           Voting Classifier         84.55         1.88          80.02   
10                       CART         64.19         2.77          71.78   
11                        LDA         83.45         1.83          79.53   
12                        SVM         78.84         2.57          79.23   

    Accuracy STD  
0           1.70  
1           1.59  
2           1.41  
3           1.18  
4           2.04  
5           1.72  
6           1.67  
7           1.32  
8           1.52  
9           1.39  
10          2.19  
11          1.58  
12          1.41  
```

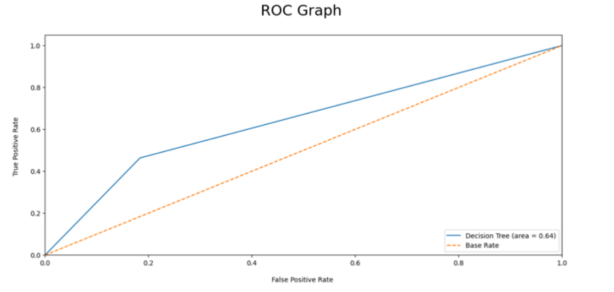
The models that were used in the problem statement were later compared and analyzed to determine the right one to solve the problem effectively. Several methods were implemented. For example, all the performance metrics were evaluated, i.e., accuracy, precision, recall and F1-score using a confusion matrix. ROC graphical representation was plotted and compared to determine performance levels of each model to the problem statement in question. K fold cross validation was also conducted to compare the multitude of models implemented in the study on the dataset.

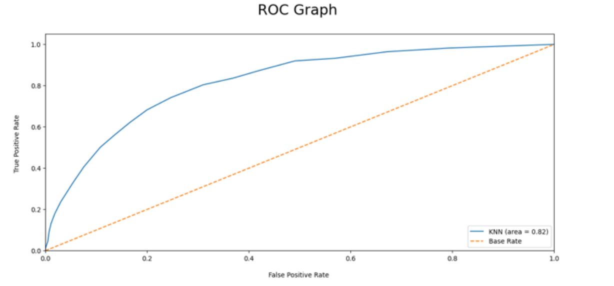


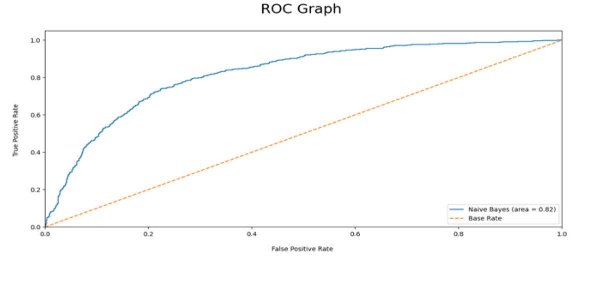
In the figure above we have plotted a confusion matrix in the form of a heatmap indicating the different actual and non-actual prediction values of the various machine learning algorithms or models that we have implemented. From this table, we can infer that the two models that stand out are the Logistic Regression model and the Linear Discriminant Analysis model that have both scored good true positive scores as depicted.

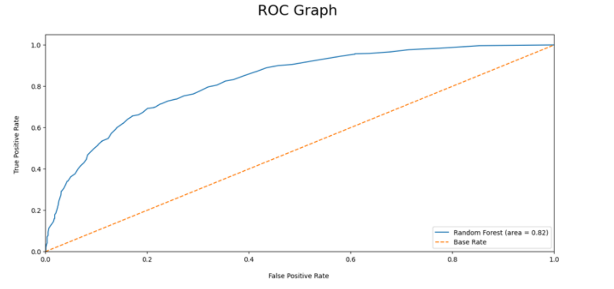
Along with determining the confusion matrix we also plotted ROC graphs to better compare and evaluate the performance levels of the different machine learning models. The performance metrics of each model was plotted in a tabular format for easy comparison and evaluation.

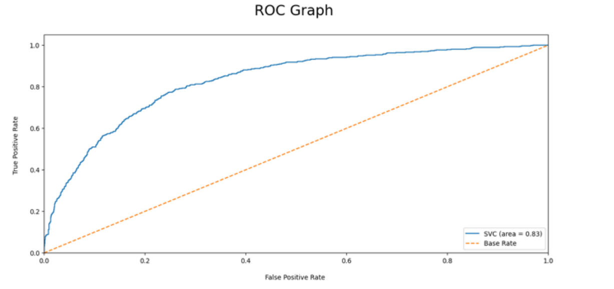


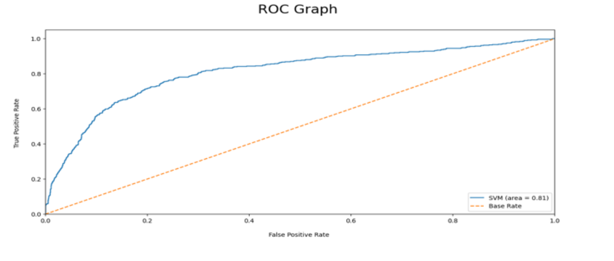


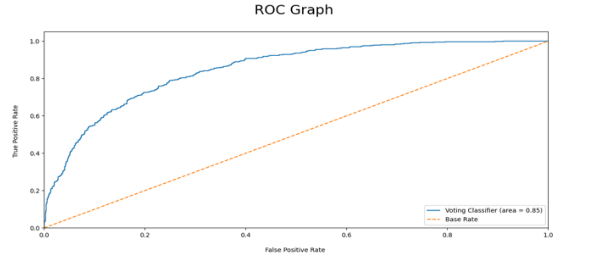


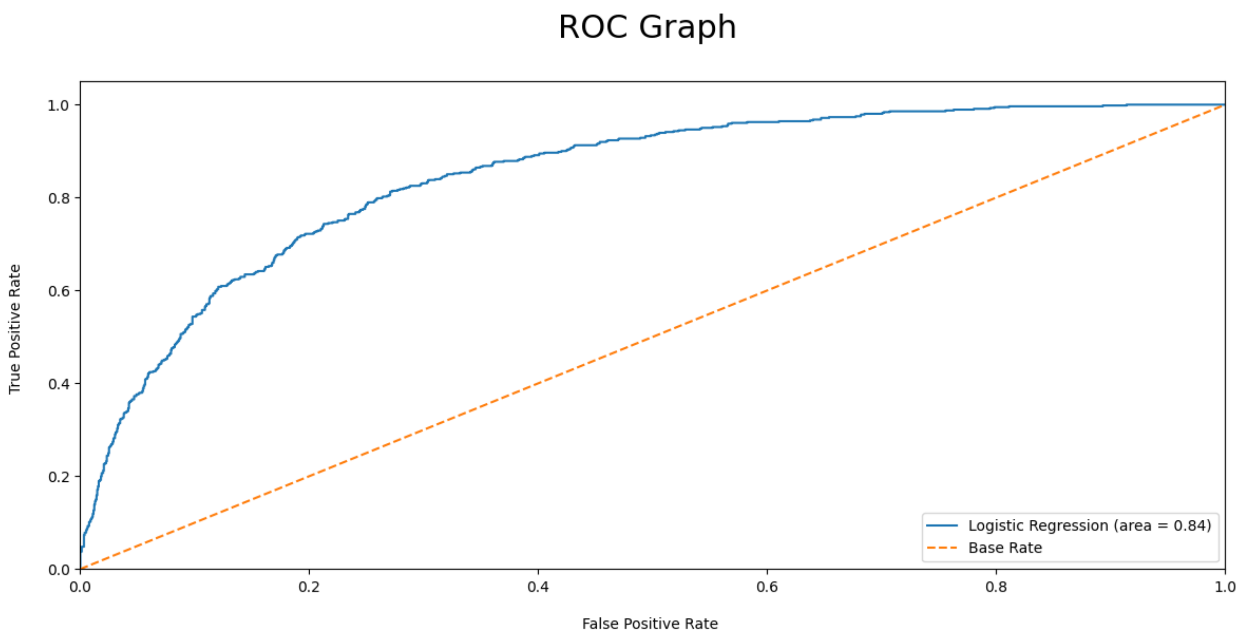






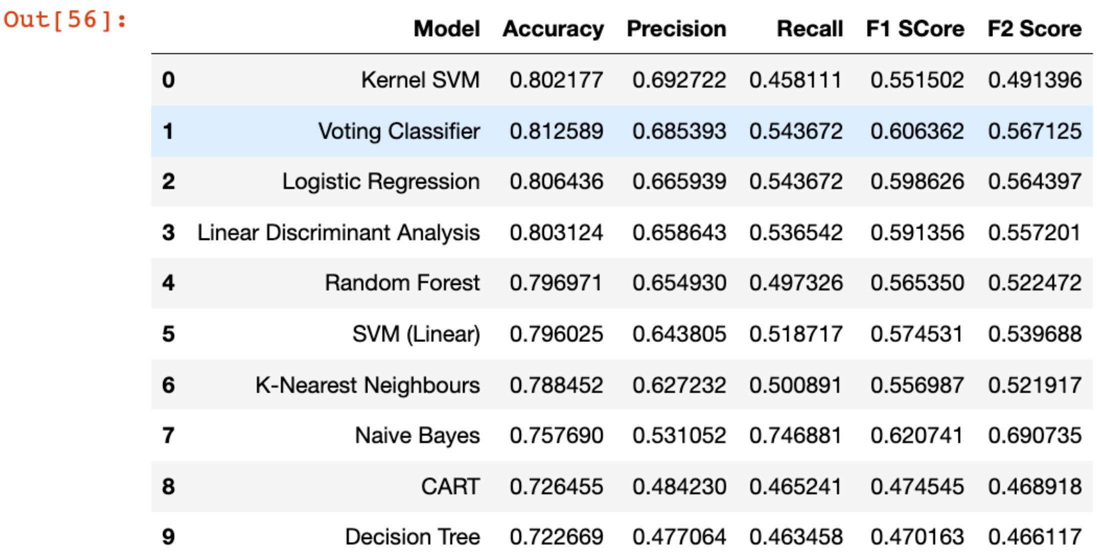






It’s clear from the ROC graphs that the ROC curves of the Logistic Regression model is superior when compared to the others purely because of the Area under the curve (AUC) score which is 0.84.

For further analysis we also have a tabular format of the performance metrics of the different models implemented below.



Out[56]:

	Model	Accuracy	Precision	Recall	F1 SCore	F2 Score
0	Kernel SVM	0.802177	0.692722	0.458111	0.551502	0.491396
1	Voting Classifier	0.812589	0.685393	0.543672	0.606362	0.567125
2	Logistic Regression	0.806436	0.665939	0.543672	0.598626	0.564397
3	Linear Discriminant Analysis	0.803124	0.658643	0.536542	0.591356	0.557201
4	Random Forest	0.796971	0.654930	0.497326	0.565350	0.522472
5	SVM (Linear)	0.796025	0.643805	0.518717	0.574531	0.539688
6	K-Nearest Neighbours	0.788452	0.627232	0.500891	0.556987	0.521917
7	Naive Bayes	0.757690	0.531052	0.746881	0.620741	0.690735
8	CART	0.726455	0.484230	0.465241	0.474545	0.468918
9	Decision Tree	0.722669	0.477064	0.463458	0.470163	0.466117

From this table, there are three models that outperform and stand out when compared to the other models. The three models are Voting Classifier, Logistic Regression and Linear Discriminant Analysis. Scoring accuracy values of 81%, 80.6% and 80.3% respectively.

We rule out voting classifier from our final list because it was not included in the syllabus. We implemented both models. But as you can see from this image, we had obtained high accuracy levels for both Logistic Regression and Linear Discriminant Analysis with the difference in value between the two models to be almost negligible. On further investigation we notice difference of performance in the other metrics. But since the differences were too minimal, we had collectively decided to implement both models.

Since our focus was on classification, we decided to choose Linear discriminant analysis and Logistic regression for our problem since they are both binary classification models that can handle categorical target variables with two classes (“yes” or “no”).

Drawbacks and Limitations of the models:

Logistic Regression:

- This model can overfit the data if there are too many features. This makes the model more complex, and it might lead to poor performance.
- This model is sensitive to outliers that can affect the result and hinder performance.

Solutions:

- Regularization methods such as L1 and L2 can help reduce overfitting.
- Detecting and removing the outliers to improve performance.

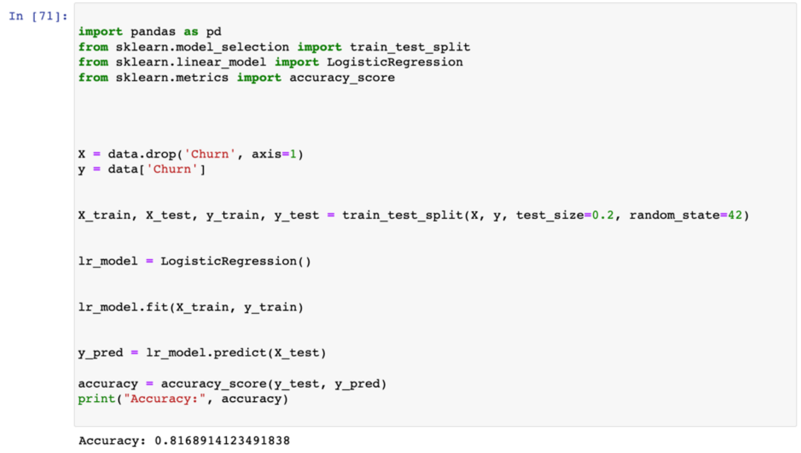
Linear Discriminant Analysis:

- This model assumes that the variance is the same for all classes which may not be the case for all datasets.
- This model does not perform well when the number of samples is smaller than the number of features.

Solutions:

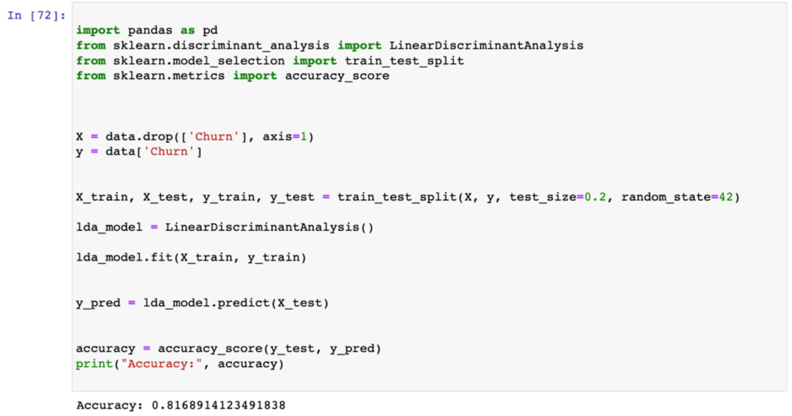
- Quadratic Discriminant Analysis is an effective way to tackle the variance problem.
- Dimensionality reduction techniques such as Principal Component Analysis (PCA) can help reduce the number of features and improve the model’s overall performance.

We implemented both these models and obtained accuracy levels of similar values that were almost negligible in difference.



```
In [71]:
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score



X = data.drop('Churn', axis=1)
y = data['Churn']


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


lr_model = LogisticRegression()


lr_model.fit(X_train, y_train)


y_pred = lr_model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

Accuracy: 0.8168914123491838
```

Logistic Regression



```
In [72]:
import pandas as pd
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score


X = data.drop(['Churn'], axis=1)
y = data['Churn']


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

lda_model = LinearDiscriminantAnalysis()

lda_model.fit(X_train, y_train)


y_pred = lda_model.predict(X_test)


accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

Accuracy: 0.8168914123491838
```

Linear Discriminant Analysis