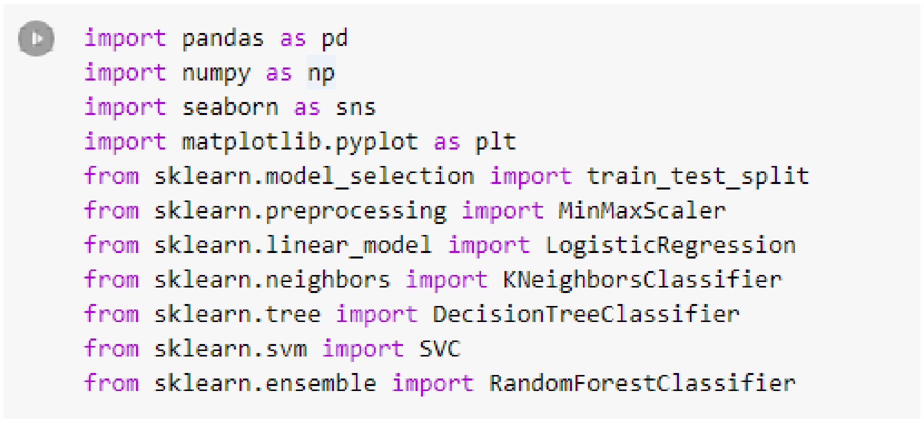
# Minor Project

## Problem Statement:

The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements such as Pregnancies, Glucose level, Blood Pressure, Skin Thickness, BMI, Insulin level, Diabetes Pedigree, and Age. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

## Python Packages/Modules used:

Python Packages used for this project are **Pandas, Numpy, Seaborn, Matplotlib, Scikit Learn.**



```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
```

Google Colab is used to write the code.

## Solving Method:

As it is a classification problem, we have to use different classifiers such as Logistic Regression, KNN, and Decision Tree, etc. to get a prediction.

The whole dataset splits among training set, test set, and cross-validation set. The training set is used to fit the classification model, the cross-validation set is used for hyperparameter tuning, and the test set to get the final accuracy of the model.

After exploratory data analysis, five classification models namely Logistic Regression, K Nearest Neighbors, Decision Tree Classifier, Random Forest Classifier, and Support Vector Classifier are used.

Logistic regression is a statistical model that uses the logistic or sigmoid function to model conditional probability. It is used to model binary classification, predicts Y = 1 or 0 for a given X. Logistic regression is easier to implement, interpret, and very efficient to train.

*k*-nearest neighbors algorithm (*k*-NN) is a non-parametric classification method. This algorithm classifies data by calculating the euclidean distance of the data point from one class. Required computational time is very less for k-NN and it is a very simple algorithm to interpret. Sometimes, KNN gives the highest accuracy compared to other supervised algorithms.

Decision Tree is a supervised classification algorithm. The objective of a Decision Tree is to construct a training model that can be used to predict the class or value of the target variable by learning basic decision rules inferred from previous data. We start at the root of the tree when using Decision Trees to forecast a class label for a record. Decision Tree doesn't require any scaling or normalization of data. Missing values of data don’t affect the decision tree building to a considerable extent.

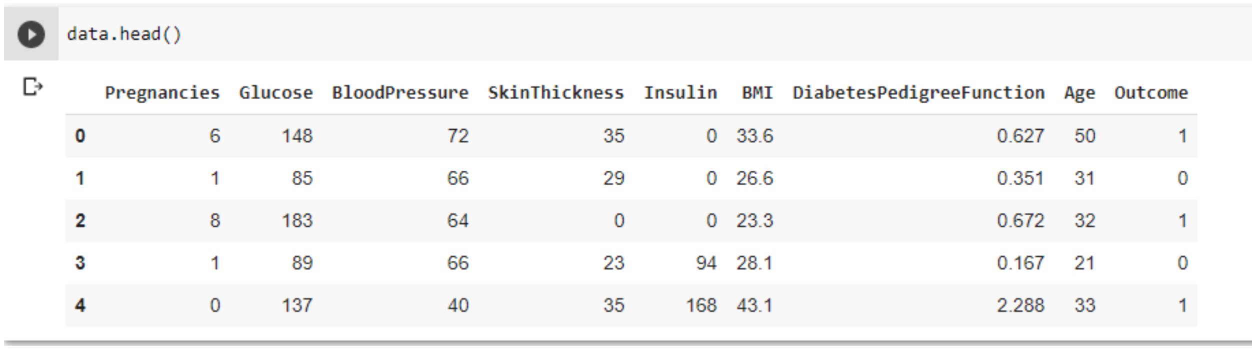
Random Forest is an ensemble method of classification. It operates by constructing multiple decision trees at the training time. The output of the random forest is the class selected by most trees. It reduces the overfitting of the data and improves accuracy. Missing values do not affect the construction of random forest.

Support Vector classifier is a non-probabilistic supervised learning algorithm. The most ideal decision boundary is taken in SVC to classify the data. On either side of the decision boundary, there is a margin. If any data point falls within the margin then penalty or losses is taken to optimize.

Hyperparameter tuning is done using a cross-validation dataset for all the models. Accuracy, Confusion Matrix, Classification Report are generated and ROC Curve is plotted on the test dataset for all classification models and finally, the Accuracy of different models is compared using a bar plot.

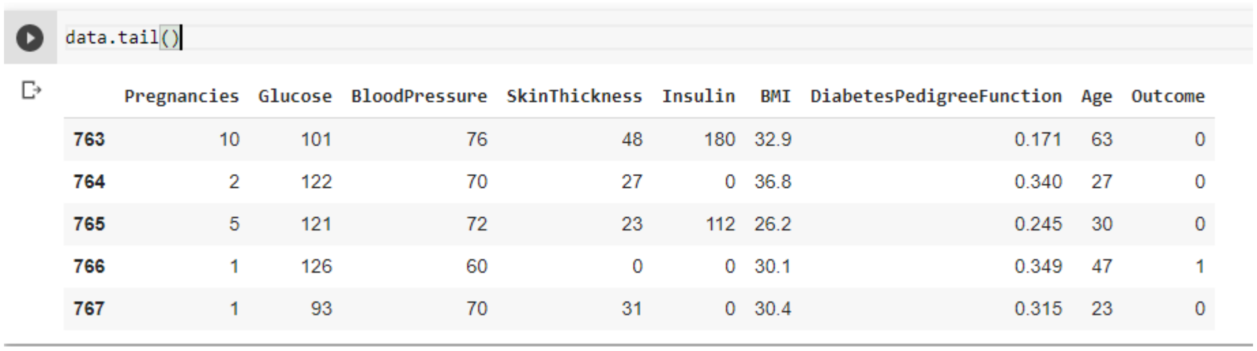
## Important parts of the Code:

### Dataset:



```
data.head()
```

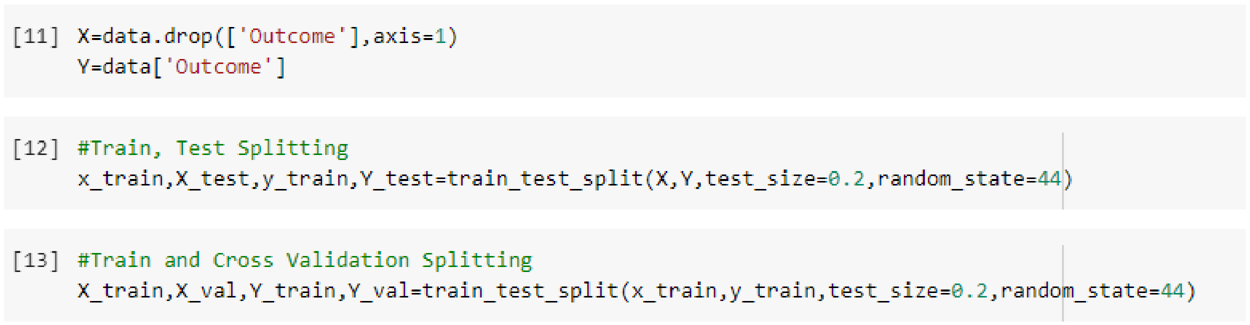
|   | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 6 | 148 | 72 | 35 | 0 | 33.6 | 0.627 | 50 | 1 |
| **1** | 1 | 85 | 66 | 29 | 0 | 26.6 | 0.351 | 31 | 0 |
| **2** | 8 | 183 | 64 | 0 | 0 | 23.3 | 0.672 | 32 | 1 |
| **3** | 1 | 89 | 66 | 23 | 94 | 28.1 | 0.167 | 21 | 0 |
| **4** | 0 | 137 | 40 | 35 | 168 | 43.1 | 2.288 | 33 | 1 |



```
data.tail()
```

|  | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
|---|---|---|---|---|---|---|---|---|---|
| **763** | 10 | 101 | 76 | 48 | 180 | 32.9 | 0.171 | 63 | 0 |
| **764** | 2 | 122 | 70 | 27 | 0 | 36.8 | 0.340 | 27 | 0 |
| **765** | 5 | 121 | 72 | 23 | 112 | 26.2 | 0.245 | 30 | 0 |
| **766** | 1 | 126 | 60 | 0 | 0 | 30.1 | 0.349 | 47 | 1 |
| **767** | 1 | 93 | 70 | 31 | 0 | 30.4 | 0.315 | 23 | 0 |

## Train, Test & Cross-Validation Splitting:



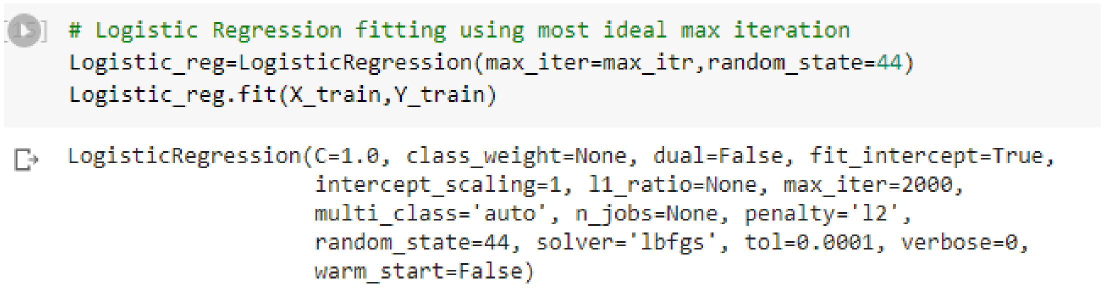
```
[11] X=data.drop(['Outcome'],axis=1)
     Y=data['Outcome']
```

```
[12] #Train, Test Splitting
     x_train,X_test,y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=44)
```

```
[13] #Train and Cross Validation Splitting
     X_train,X_val,Y_train,Y_val=train_test_split(x_train,y_train,test_size=0.2,random_state=44)
```

## Classification Models:

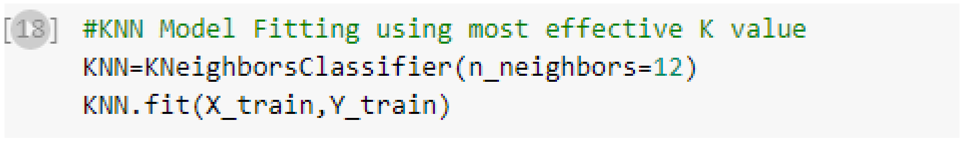
### Logistic Regression:



```
# Logistic Regression fitting using most ideal max iteration
Logistic_reg=LogisticRegression(max_iter=max_itr,random_state=44)
Logistic_reg.fit(X_train,Y_train)
```

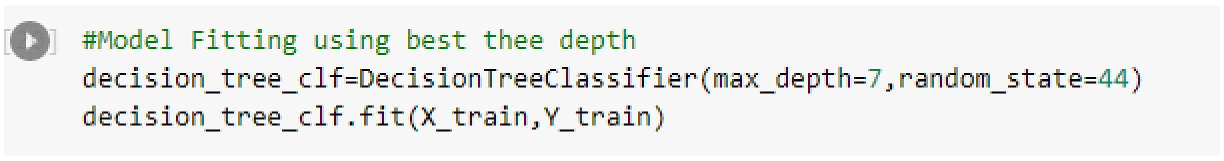
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=2000,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=44, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

### K-Nearest Neighbors:

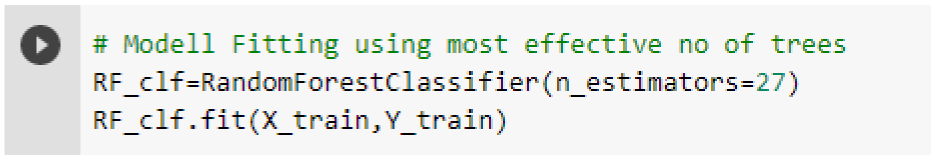


```
[18] #KNN Model Fitting using most effective K value
     KNN=KNeighborsClassifier(n_neighbors=12)
     KNN.fit(X_train,Y_train)
```

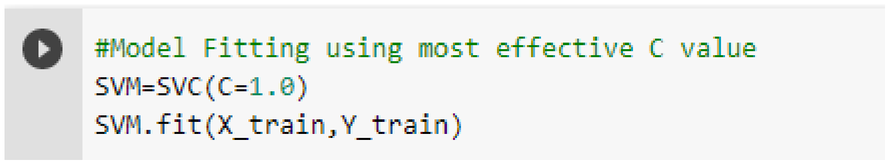
## Decision Tree:


```
#Model Fitting using best thee depth
decision_tree_clf=DecisionTreeClassifier(max_depth=7,random_state=44)
decision_tree_clf.fit(X_train,Y_train)
```

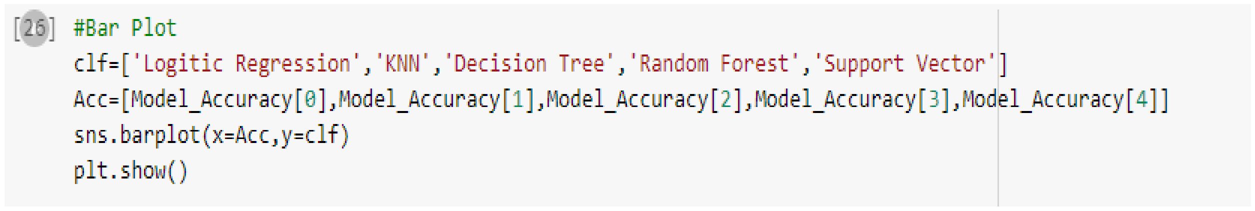
## Random Forest:


```
# Modell Fitting using most effective no of trees
RF_clf=RandomForestClassifier(n_estimators=27)
RF_clf.fit(X_train,Y_train)
```

## Support Vector Classifier:


```
#Model Fitting using most effective C value
SVM=SVC(C=1.0)
SVM.fit(X_train,Y_train)
```

## Bar Plot:

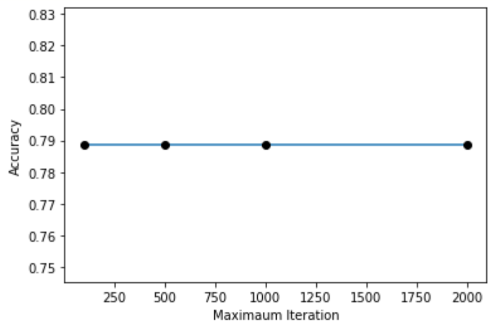

```
#Bar Plot
clf=['Logitic Regression','KNN','Decision Tree','Random Forest','Support Vector']
Acc=[Model_Accuracy[0],Model_Accuracy[1],Model_Accuracy[2],Model_Accuracy[3],Model_Accuracy[4]]
sns.barplot(x=Acc,y=clf)
plt.show()
```

## Result & Conclusion:

## Logistic Regression:

### Hyperparameter Tuning:

Cross-Validation Accuracy Vs Max Iteration

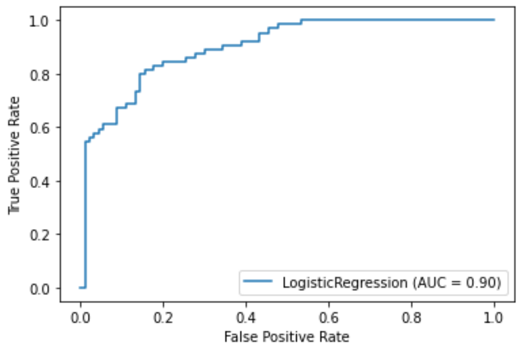


There is no change in Cross-Validation accuracy w.r.t maximum iterations. So, we can use 100 iterations to save computational time

### Result:

```
Accuracy on Test Data using Logistic Regression: 80.51948051948052 %

Confusion Matrix:
 [[88 28]
 [ 2 36]]

 Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.76      0.85       116
           1       0.56      0.95      0.71        38

    accuracy                           0.81       154
   macro avg       0.77      0.85      0.78       154
weighted avg       0.88      0.81      0.82       154
```

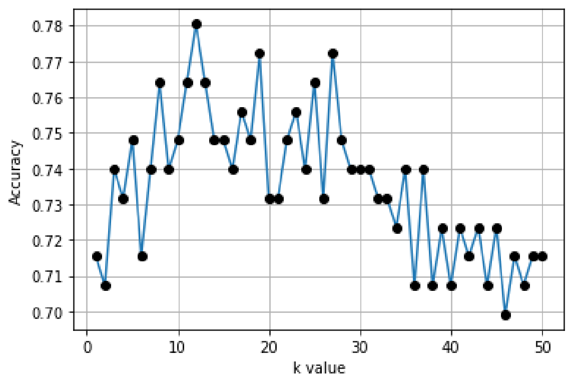
ROC Curve:



## K-Nearest Neighbors:

### Hyperparameter Tuning:

Cross-Validation Accuracy Vs K value

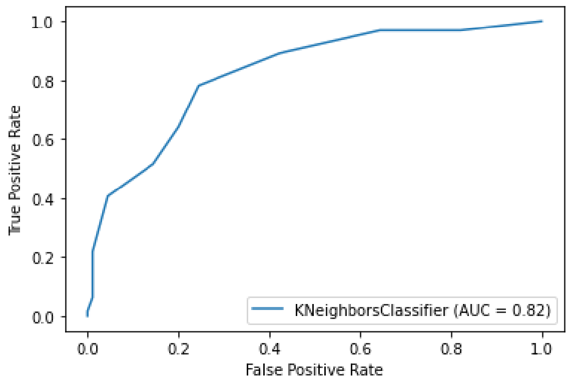


Highest Cross Validation Accuracy at K=12

### Result:

```
Accuracy on Test Data using K nearest Neighbors: 72.72727272727273 %

Confusion Matrix:
 [[86 38]
 [ 4 26]]

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.69      0.80       124
           1       0.41      0.87      0.55        30

    accuracy                           0.73       154
   macro avg       0.68      0.78      0.68       154
weighted avg       0.85      0.73      0.75       154
```

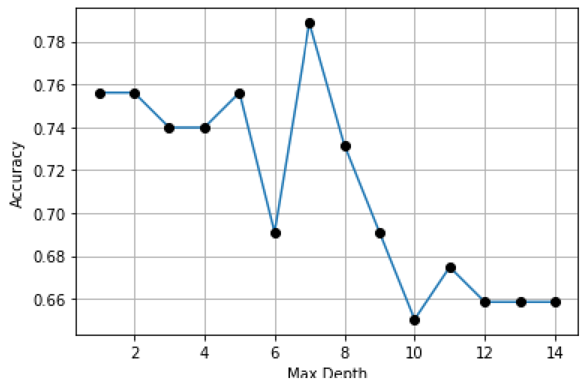
ROC Curve



## Decision Tree Classifier:

### Hyperparameter Tuning:

Cross-Validation Accuracy Vs Max Depth of the Tree

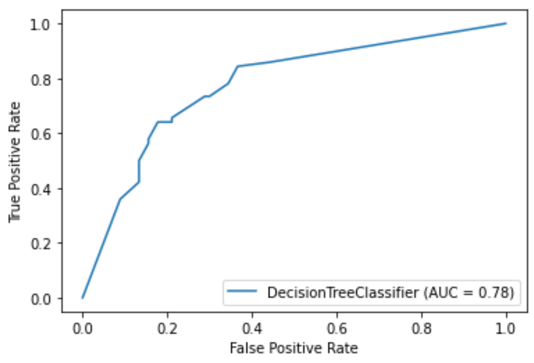


Highest cross-validation accuracy at depth=7

### Result:

```
Accuracy on Test Data using Decision Tree: 74.67532467532467 %

Confusion Matrix:
 [[74 23]
 [16 41]]

Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.76      0.79        97
           1       0.64      0.72      0.68        57

    accuracy                           0.75       154
   macro avg       0.73      0.74      0.73       154
weighted avg       0.76      0.75      0.75       154
```

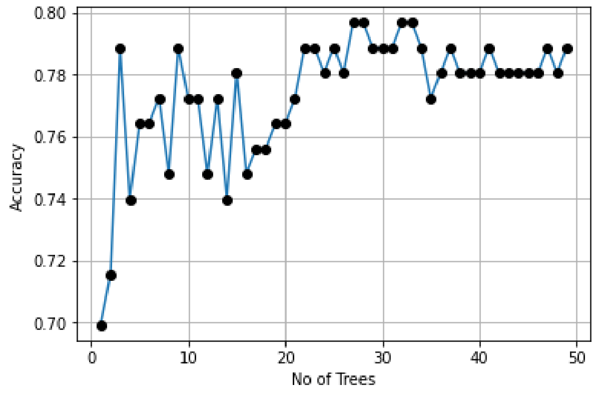
ROC Curve



## Random Forest Classifier:

### Hyperparameter Tuning:

Cross-Validation Accuracy Vs No of Trees

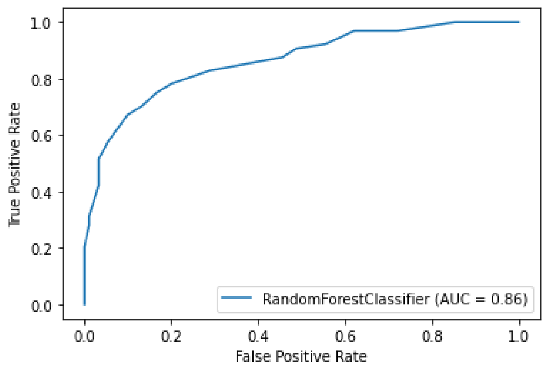


Highest cross-validation accuracy at trees of the Random Forest Classifier=27

### Result:

```
Accuracy on Test Data using Random Forest: 79.22077922077922 %

Confusion Matrix:
 [[85 27]
 [ 5 37]]

Classification Report:
               precision    recall  f1-score   support

           0       0.94      0.76      0.84       112
           1       0.58      0.88      0.70        42

    accuracy                           0.79       154
   macro avg       0.76      0.82      0.77       154
weighted avg       0.84      0.79      0.80       154
```

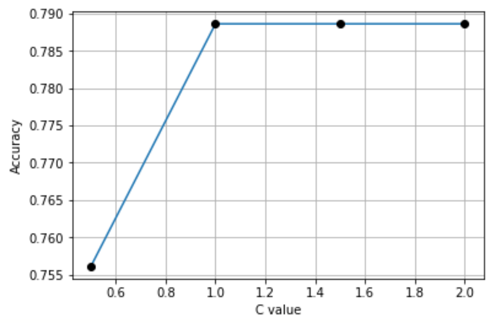
ROC Curve



## Support Vector Classifier:

### Hyperparameter Tuning:

Cross-Validation Accuracy Vs C value

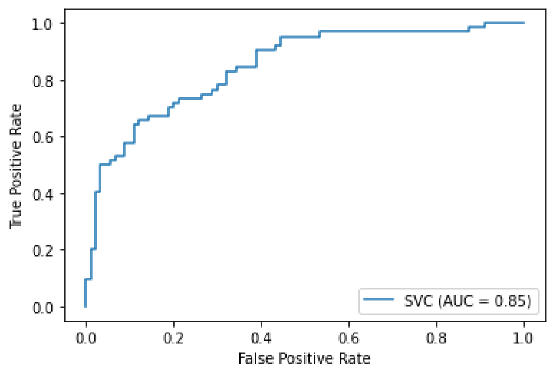


At C=1.0, cross-validation accuracy is the highest

### Result:

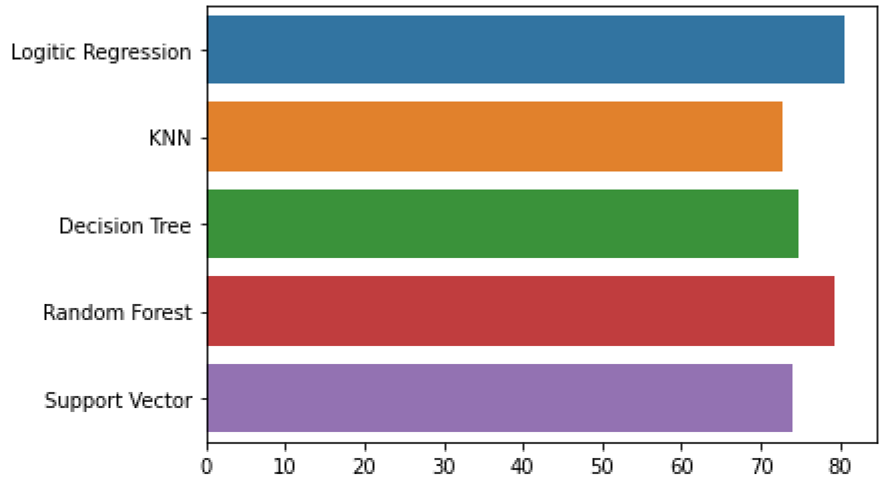
```
Accuracy on Test Data using Random Forest: 74.02597402597402 %

Confusion Matrix:
 [[87 37]
 [ 3 27]]

Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.70      0.81       124
           1       0.42      0.90      0.57        30

    accuracy                           0.74       154
   macro avg       0.69      0.80      0.69       154
weighted avg       0.86      0.74      0.77       154
```

ROC Curve



## Accuracy Comparison:

```
Accuracy using Logitic Regression: 80.51948051948052 %
Accuracy using K-Nearest Neighbor: 72.72727272727273 %
Accuracy using Decision Tree Classifier: 74.67532467532467 %
Accuracy using Random Forest Classifier: 79.22077922077922 %
Accuracy using Support Vector Classifier: 74.02597402597402 %
```



**Among these 5 classification models, Logistic Regression performed the best with an accuracy of 80.52% on the test data set and Random Forest is just behind Logistic Regression with an accuracy of 79.22%.**