TOPIC: - LOGISTIC REGRESSION ON CREDIT CARD APPROVAL

GROUP – 12

Group Members:

Chinmay Bhat
Ram Warutkar
Sakshi Jain

INTRODUCTION:

Many times, bank faces application of a credit card issuance, going through each application sometimes becomes a lengthy process. By machine learning technique like classification technique we can build a model which will predict whether to accept an application or to reject an application.

We have tried to build a model which will predict for acceptance or rejection of an application. For this purpose, we have taken data from the UCI website.

The link for the dataset is:

https://archive.ics.uci.edu/ml/datasets/credit+approval

A snapshot of data

Gender	Age	Debt	Married	Bank_Customer	Education_Level	Ethnicity	Years_Employed	Prior_Default	Employed	Credit_Score	Driver_License	Citizen	Z
b	30.83	0.000	u	g	w	v	1.250	t	t	1	f	g	
a	58.67	4.460	u	g	q	h	3.040	t	t	6	f	g	
a	24.50	0.500	u	g	q	h	1.500	t	f	0	f	g	
b	27.83	1.540	u	g	w	v	3.750	t	t	5	t	g	
b	20.17	5.625	u	g	w	v	1.710	t	f	0	f	s	
b	32.08	4.000	u	g	m	v	2.500	t	f	0	t	g	
b	33.17	1.040	u	g	r	h	6.500	t	f	0	t	g	
a	22.92	11.585	u	g	cc	v	0.040	t	f	0	f	g	
b	54.42	0.500	y	p	k	h	3.960	t	f	0	f	g	
b	42.50	4.915	y	p	w	v	3.165	t	f	0	t	g	

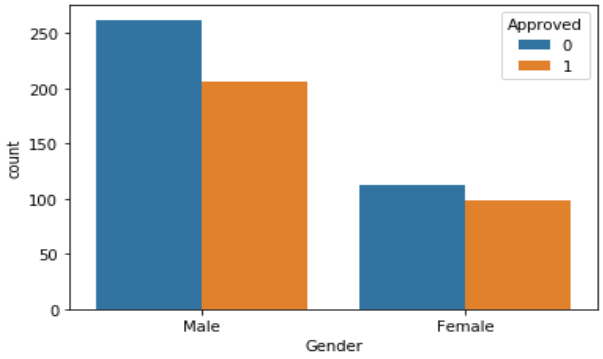
The dataset contains 9 categorical features and 7 quantitative features. There are 5 per cent missing values in the original dataset.

After reading the data, we went through the study of what are the major factors that affects the decision of approval or rejection of an applicant application.

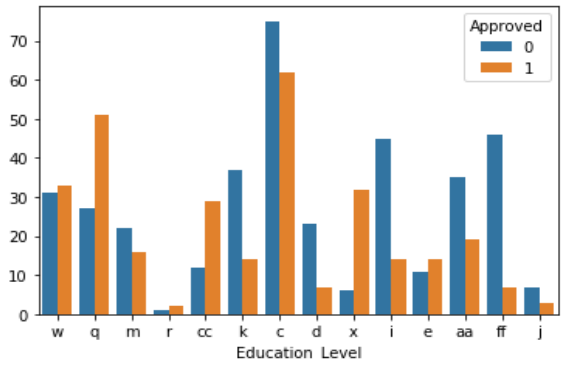
We came to know that the Income, Debt, Credit Score and whether the person has made any default in the past or not are some of the important features which act as a deciding factor.

But In our data, there are other factors as well like education, married or unmarried, education level, ethnicity, citizenship of an applicant.

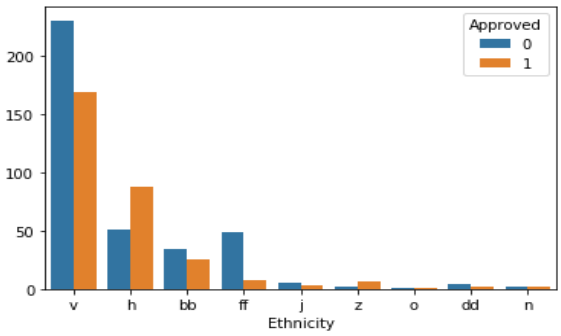
To know whether these factors really impact the decision, we performed Exploratory data analysis and through that we were able to eliminate some feature.



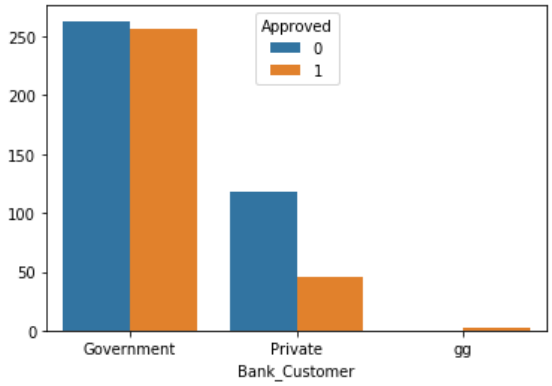
- This is the plot of gender and count of approval and un approval rate among each gender. As we can see that, there's not much difference between a male getting approval or a female getting approval.



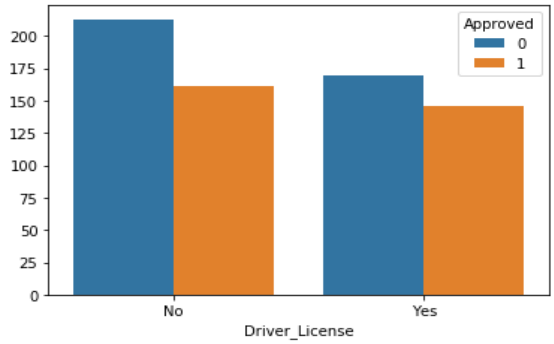
- By looking this graph, it shows the education level has pretty balanced class except the education x where the success rate is pretty high. So, we can infer a person with education x is more likely to get approval



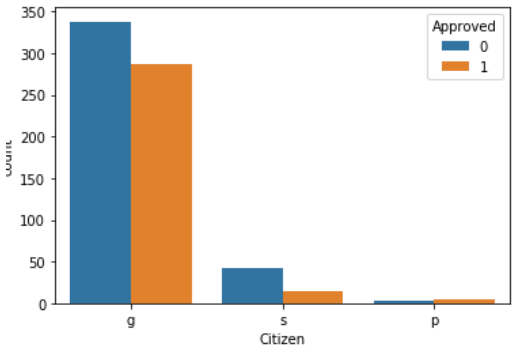
- The difference between approval and un approval is not noticeable in any group, so we can conclude that ethnicity of an applicant does not play much role.



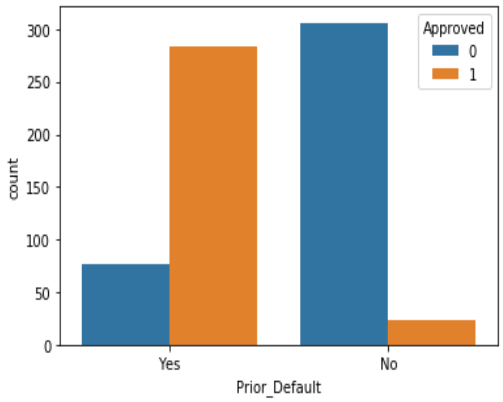
- So, for a government employee the rate is quite similar, for a private customer the rate for un approval is quite high and for the other it does not matter because they are very less in number.
- So, we are not sure at this stage that we can drop this factor or not.



- It seems that having driving license or not having does not affect the decision. So, we can drop this feature.

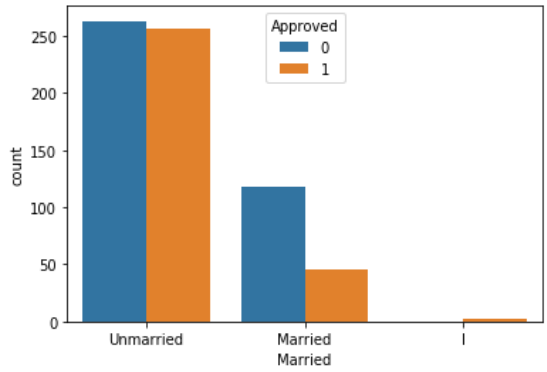


- The citizen graph also does not show any noticeable pattern. The rate is quite same for each group of citizens.

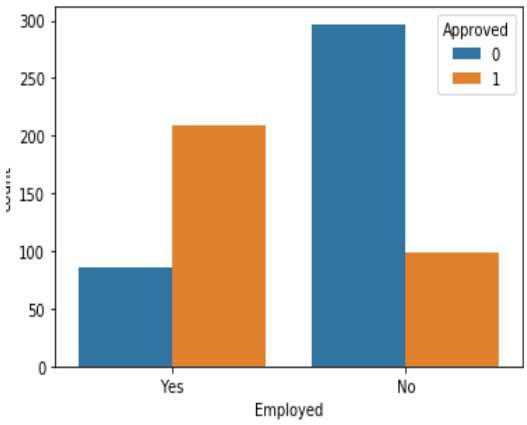


• This is the graph of, whether the person has made default in the past or not.

• As we can see that person with no default history are more likely to get credit card then person with a default history



• The applicant is married or not married does not have a much impact on the decision.



• The applicant is employed or not employed, has a major impact on the decision of approval.

After removing all the unimportant features, the snapshot of the data is,

	Age	Debt	Bank_Customer	Education_Level	Years_Employed	Prior_Default	Employed	Credit_Score	Income	Approved
0	31.0	0.00	Government	w	1.25	Yes	Yes	1	0	1
1	59.0	4.46	Government	q	3.04	Yes	Yes	6	560	1

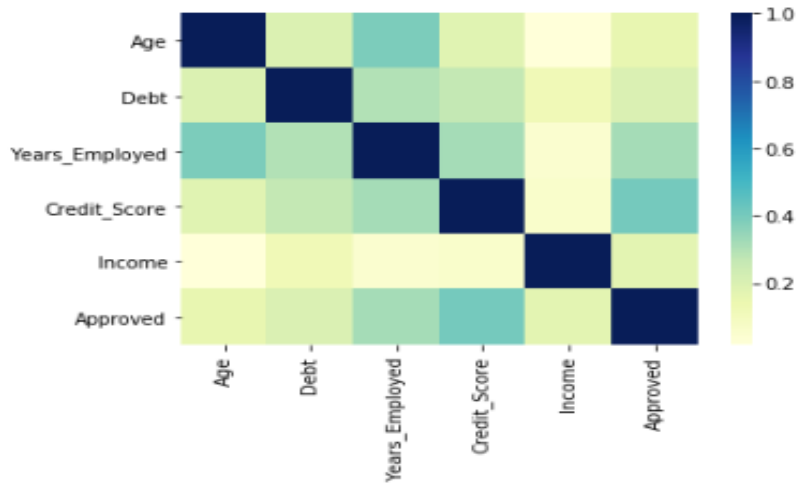
The data set also contain some missing value which has been filled with the mean, when it is a quantitative feature and with the mode when it is qualitative qualitative feature.

```
# Filling Missing Values
import statistics as st
df["Gender"].replace(to_replace=np.NaN,value=st.mode('Gender'),inplace = True)
df["Age"].replace(to_replace=np.NaN,value=np.mean(df['Age']),inplace = True)
df["Married"].replace(to_replace=np.NaN,value=st.mode(df['Married']),inplace = True)
df["Education_Level"].replace(to_replace=np.NaN,value=st.mode(df['Education_Level']),inplace = True)
df["Bank_Customer"].replace(to_replace=np.NaN,value=st.mode(df['Bank_Customer']),inplace = True)
```

The Correlation Matrix for quantitative features is:

	Age	Debt	Years_Employed	Credit_Score	Income	Approved
Age	1.000000	0.199504	0.391851	0.185704	0.018554	0.160007
Debt	0.199504	1.000000	0.298902	0.271207	0.123121	0.206294
Years_Employed	0.391851	0.298902	1.000000	0.322330	0.051345	0.322475
Credit_Score	0.185704	0.271207	0.322330	1.000000	0.063692	0.406410
Income	0.018554	0.123121	0.051345	0.063692	1.000000	0.175657
Approved	0.160007	0.206294	0.322475	0.406410	0.175657	1.000000

The heatmap for the same:



As we can see that there are some categorical features which needs to be encoded.

```
# Converting categorical into quantitative
from sklearn.preprocessing import OneHotEncoder
Prior_Default=pd.get_dummies(df.Prior_Default,drop_first=True)
Bank_Customer=pd.get_dummies(df.Bank_Customer,drop_first=True)
Employed=pd.get_dummies(df.Employed,drop_first=True)
df1=pd.concat([df,Prior_Default,Bank_Customer,Employed],axis=1)
```

After encoding the data we have split the data into training and testing part.

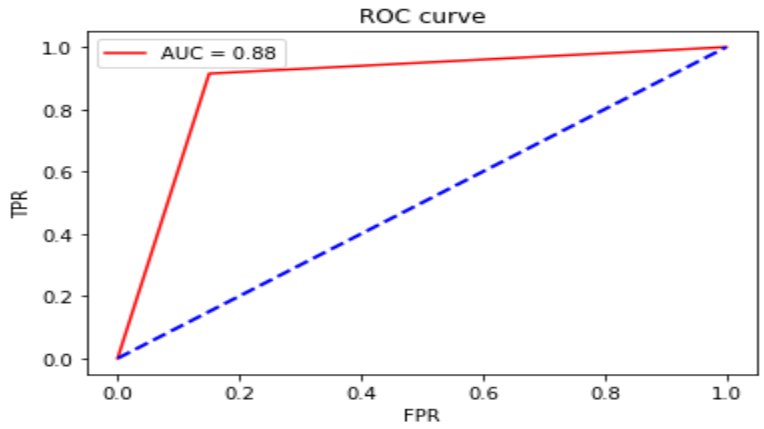
```
#Splitting the dataset into train and test part
from sklearn.model_selection import train_test_split
seed = 10
test_size = 0.3
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = test_size, random_state = seed)
```

On the training dataset we have applied various classification techniques.

LOGISTIC REGRESSION

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(random_state = 10, max_iter = 1000)
model.fit(X_train,y_train)
```

The Logistic regression has performed good on the dataset, giving AUC of 0.88 on testing dataset.



The confusion matrix for logistic regression is:

Actual /Predicted ->	0	1
0	96	17
1	8	86

The confusion matrix also shows that model has perfomed good on the dataset. The True positive rate and True Negative Rate are also satisfactory and moreover the False Negative rate and False Positive Rate are quite low. That means the model has been able to identify that what should be rejected and what should not be rejected.

K- NEAREST NEIGHBOUR

Through GridSearchCV we found the best value of K i.e. 6

```
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
estimator =KNeighborsClassifier ()
param_grid = { 'n_neighbors':[6,8,7,9,10,11,12]}
```

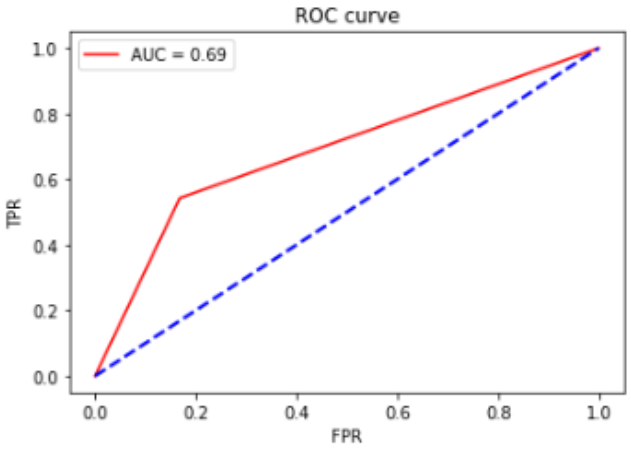
```
print(grid.best_params_)
```

```
{'n_neighbors': 7}
```

Fitting the model

```
# K - Nearest Neighbour
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=6)
knn.fit(X_train, y_train)
```

AUC for KNN is:



The KNN has performed poor in comparision with Logistic Regression, As we know KNN is a distance based algorithm and in the data the values are in different range, Maybe KNN can perform better after scaling the data.

KNN AFTER SCALING THE DATA

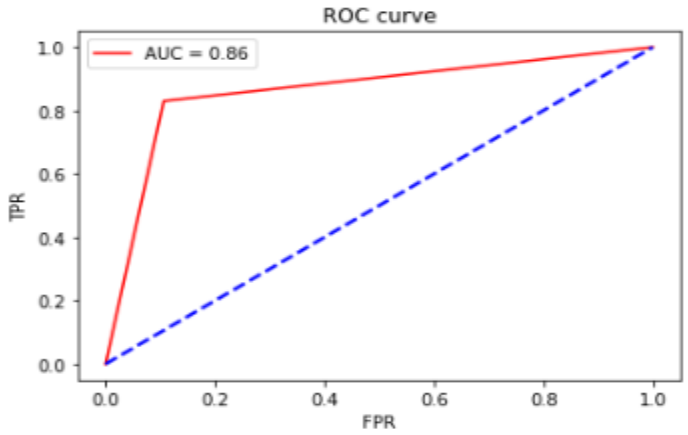
```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
X_train_s=sc.fit_transform(X_train)
X_test_s=sc.fit_transform(X_test)
```

FITTING THE MODEL

Through GridSerchCV we were able to find the best value of K that is 7 for scaled data

```
from sklearn.neighbors import KNeighborsClassifier
knn_s = KNeighborsClassifier(n_neighbors=7)
knn_s.fit(X_train_s, y_train)
```

AUC for KNN is



Now, after scaling the data the AUC has incresed to 0.86

The Confusion Matrix for K-NEAREST NEIGHBOUR is:

Actual /Predicted ->	0	1
0	101	12
1	16	78

The True Negative Rate and False Negative Rate are quite higher then Logistic Regession whereas True Positive Rate and False Positive Rate are low as compare to Logistic Regression

SUPPORT VECTOR MACHINE

Through GridSearchCV we found the best value of C for the linear kernel that is 4

```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
estimator =SVC ()
param_grid = { 'kernel':['linear'],
               'C'     :[2,3,4] }
```

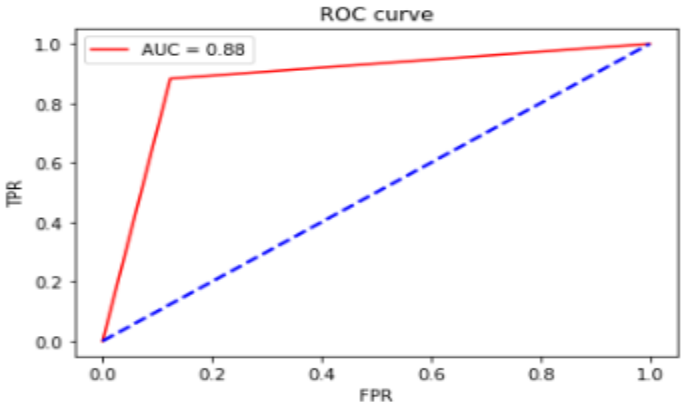
```
print(grid.best_params_)
```

```
{'C': 4, 'kernel': 'linear'}
```

```
from sklearn.svm import SVC
clf_2= SVC(kernel='linear',C=4)
clf_2.fit(X_train,y_train)

```

The AUC for Support Vector Classifier is:



The confusion matrix for Support Vector Classifier is:

Actual /Predicted ->	0	1
0	99	14
1	11	83

The True Negative Rate of Support Vector Classifier is low as compare to K- Nearest Neighbor and high as compare to Logistic Regression. The Support Vector Classifier has done a good job in identifying the applicants whom should be approved and whom should be rejected. As we all know approving an individual who should not be is more dangerous then rejecting an applicant who should be given approval. In the case of False positive the Support Vector has performed slightly poor then K- Nearest Neighbor but better then Logistic Regression.

SUPPORT VECTOR MACHINES

After this we perform Support Vector with kernel Radial, through GridSearchCV we found the best value of gamma

```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
estimator =SVC ()
param_grid = { 'kernel':['rbf'],
               'gamma'    :[0.1,0.01,0.001,0.0001] }
```

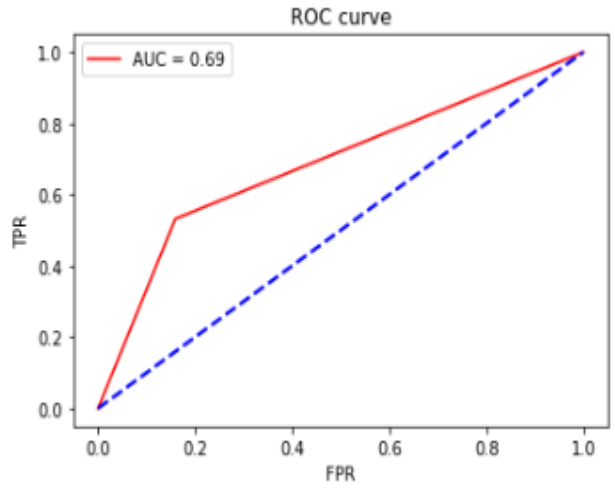
```
print(grid.best_params_)
```

```
{'gamma': 0.001, 'kernel': 'rbf'}
```

Fitting the Model

```
from sklearn.svm import SVC
clf_3= SVC(kernel='rbf',gamma=0.001)
clf_3.fit(X_train,y_train)
```

The AUC for Support Vector with kernel Radial is:



The Confusion Matrix for Support Vector with kernel Radial is:

Actual /Predicted ->	0	1
0	95	18
1	44	50

Although the model has able to identify the applicants which should be rejected but considering other area the model has performed poor. This suggest that probably the decision boundary is linear and Radial trying to capture it non-linearly and hence performing poor.

DECISION TREE

Through Grid Search we found the best parameters:

```
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier
estimator = DecisionTreeClassifier()
param_grid = { 'criterion':['gini','entropy'],
               'random_state':[100,150,200,50],
                'max_depth'    :[1,2,3,4,5],
                  'min_samples_leaf':[1,2,3,4,5]}
```

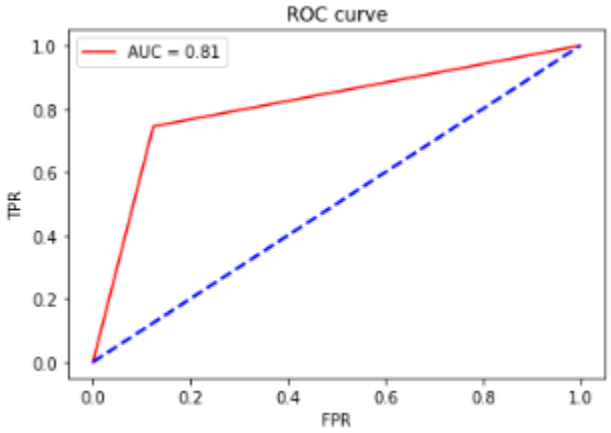
```
print(grid.best_params_)
```

```
{'criterion': 'entropy', 'max_depth': 4, 'min_samples_leaf': 4, 'random_state': 50}
```

Fitting the model

```
#fitting the model
from sklearn.tree import DecisionTreeClassifier
clf_tree=DecisionTreeClassifier(criterion = "entropy", random_state = 50,max_depth=4, min_samples_leaf=4)
clf_tree.fit(X_train,y_train)
```

The AUC for Classfication Tree is:



The Confusion Matrix for Classification Tree is:

Actual /Predicted ->	0	1
0	99	14
1	24	70

The Classification tree has performed slightly poor then other model (Logistic Regression, K-Nearest Neighbor, Support Vectors). The False Negative Rate is too much as compare to other models. If we compare Random Forest with other model, it has not performed good in case of identifying applicants

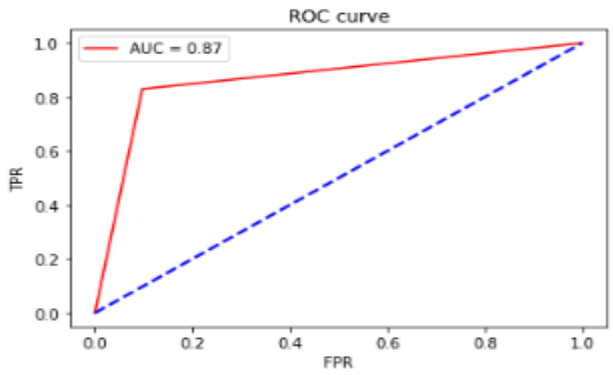
who should got approval or in other words the True Positive Rate is quite low as compare to other models.

RANDOM FOREST CLASSIFICATION

Fitting the model:

```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier()
rf.fit(X_train,y_train)
```

The AUC for Random Forest is:



The Confusion Matrix for Random Forest is:

Actual /Predicted ->	0	1
0	102	11
1	16	78

The model has performed good in terms of False Positive as compare to Decision Tree Classification. That means the Random Forest Classification has also done a good job in identifying the applicants who are not right for the credit card approval.

BOOSTING

Through grid search we found the best parameter.

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import AdaBoostClassifier
estimator = AdaBoostClassifier()
param_grid = { 'n_estimators':[100,200,300],
               'learning_rate':[0.001,0.01,0.1,1]}
```

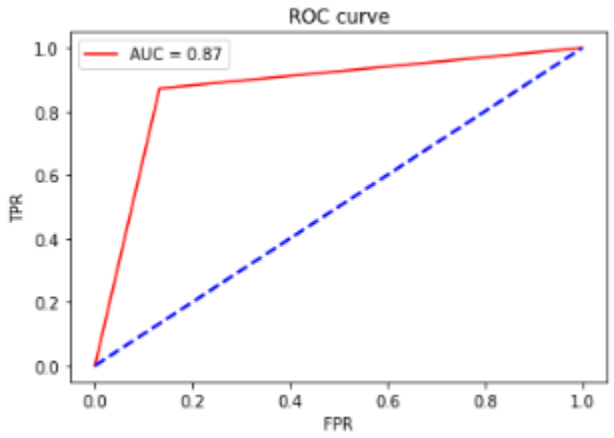
```
print(grid.best_params_)
```

```
{'learning_rate': 0.1, 'n_estimators': 100}
```

Fitting the model:

```
from sklearn.ensemble import AdaBoostClassifier
ad_boost=AdaBoostClassifier(n_estimators=100,learning_rate=0.1)
ad_boost.fit(X_train,y_train)
```

The AUC for AdaBoost is:



The Confusion Matrix for AdaBoost is:

Actual /Predicted ->	0	1
0	98	15
1	12	82

The AdaBoost model has performed good on the dataset. The False Positive Rate is quite low as compare to Decision Tree Classification. If we compare to other models other then Logistic Regression, the False Negative Rate is quite low in case of Adaboost Classification

COMPARISION OF ALL THE MODELS

AUC OF ALL THE MODELS:

ALGORITHM	AUC
LOGISTIC REGRESSION	0.88
KNN WITHOUT SCALING	0.69
KNN WITH SCALING	0.86
SUPPORT VECTOR WITH KERNEL LINEAR	0.88
SUPPORT VECTOR WITH KERNEL RADIAL	0.69
DECISION TREE	0.81
RANDOM FOREST	0.87
ADABOOST CLASSIFICATION	0.87

CONFUSION MATRIX OF ALL THE MODELS

LOGISTIC REGRESSION

Actual /Predicted ->	0	1
0	96	17
1	8	86

KNN WITHOUT SCALING

Actual /Predicted ->	0	1
0	101	12
1	16	78

SUPPORT VECTOR WITH KERNEL LINEAR

Actual /Predicted ->	0	1
0	99	14
1	11	83

SUPPORT VECTOR WITH KERNEL RADIAL

Actual /Predicted ->	0	1
0	95	18
1	44	50

DECISION TREE CLASSIFICATION

Actual /Predicted ->	0	1
0	99	14
1	24	70

RANDOM FOREST CLASSIFICATION

Actual /Predicted ->	0	1
0	102	11
1	16	78

ADABOOST CLASSIFICATION

Actual /Predicted ->	0	1
0	98	15
1	12	82

ALGORITHMS	TRUE POSITIVES	TRUE NEGATIVES	FALSE POSITIVES	FALSE NEGATIVES
LOGISTIC REGRESSION	86	96	17	8
KNN, WITH SCALING	78	101	12	16
SVC, LINEAR KERNEL	83	99	14	11
SVM, RADIAL KERNEL	50	95	18	44
DECISION TREE	70	99	14	24
RANDOM FOREST	78	102	11	16
ADABOOST CLASSIFICATION	82	98	15	12

SELECTION OF BEST MODEL:

• So, there are models which are giving high True Positives and models which giving True Negatives. To decide the best model among these algorithms is confusing because there are models which have same AUC.

• If a bank wants to identify applicants who should not be approved for credit card. In other words, the banks do not want to face loss then a bank should go with model where True Negative Rate is higher, Models like K-Nearest Neighbor with Scaling and Random Forest.

• And if a bank wants to identify potential applicants then a bank should go for model with high True Positives, models like Logistic Regression and Support Vector Classifier with kernel linear.

LESSONS LEARNT FROM THE PROJECT

• Through this project we were able to understand that, how EDA can help us to eliminate features which are not important for our model. This will also help us to interpret the model more correctly.

• Sometimes the two technique can perform same, but when selecting a technique, we will decide what we want, Identification of a correct application or rejecting the customer which can bring losses to the business. The TPR, TNR, FNR and FPR plays an important role in selecting the desired technique.

• And also, when decision boundary is linear, the model which are made for nonlinear boundary will not perform good.

• After scaling the data, the performance for the KNN model increases, which taught us that any algorithm which works on the distance, needs a scaled data because many times our data has different features and range of these features are not in the same scale.
