Bank Customer Churn Prediction with Explainable AI

Apoorva Banubakode, Sai Chaitanya Dasari, Sathya Sri Pasham, Renu Dige
Department of Software Engineering,
San Jose State University, United States

***Abstract*—Customer churn is a major problem of customers leaving your products/subscription and moving to another service. Due to direct effect on profit margins, businesses now are looking to identify customers who are at the risk of churning and retaining them by personalized promotional offers. In order to retain them, they need to identify the customers as well as the reason of churning so that they can provide the customers with personalized offers and products. The aim of our project is to solve this problem for banking domain, by identifying which customers are at risk of churning and what are the reasons for churning with the help of data mining and machine learning algorithms. The project focuses on 2 deliverables - Predict customers likely to churn using supervised learning classification algorithms and customer segmentation of customers using unsupervised learning to validate the similarities in the 'likely to churn' customer subset to come up with different segments. The reasons for a particular customer churn can vary from internal factors as well as external factors but we will try to understand the reasons of churning depending on internal factors using explainable AI, which breaks into the blackbox of machine learning algorithms and gives a clear explanation of the predictions.**

Keywords-churn prediction; Explainable AI, churning, segmentation, unsupervised learning

I. Introduction

The term Customer Churn refers to the case when a customer or subscriber of a particular company terminates his or her relationship with the services provided by that company. The cost of customer churn in total includes the revenue, which is lost, and the costs associated with marketing to replace the old customers with new ones. Apart from that, the cost of initially acquiring a customer might not have been fully covered by that customer's spending to date. Hence, minimizing customer churn is a main concern for most companies. Therefore, the capacity of predicting the customers who are on the verge of exiting the company's services while there is still time left to take necessary actions to prevent it from happening appears to be a huge source of potential revenue for the companies.

Banking Industries which are mostly customer centric are as well not exempt from the repercussions of customer churning. They track interactions of the customer with the company to detect behaviors such as dormancy of account status reduce transactions etc in order to detect early signs of churn and prevent it. The top factors. Which lead customers to look for other alternatives seem to be bad quality of service. Exorbitant. Banking fees in today's world each and every poor customer service. Seemingly unfair prices and lack of access to the. Services at all times and places trigger customer sentiments which lead to risk of churn and major revenue loss in turn.

Therefore, attaining the ability to predict churn which in turn enables the banks to prevent it by understating the customer's needs. Choices and sentiments is the motive behind this project

Data Visualization

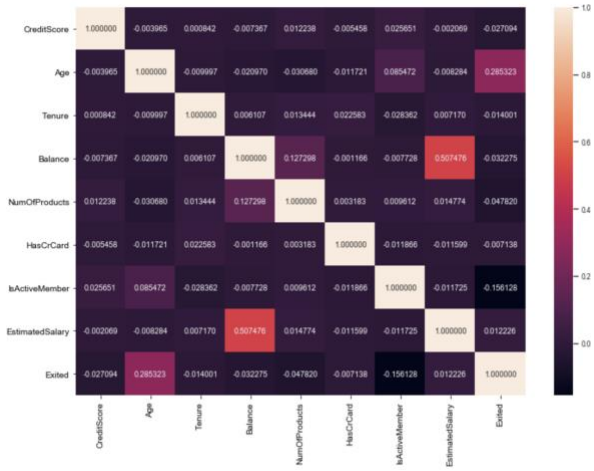
The very first step is to visualize data, create histograms, box plots and interquartile plots to understand the distributions visually, and to check if there is skewness in the data. A lot of times in real life, such data is usually imbalanced in nature, meaning a lot more labels for one class than another which is a problem as model doesn't have enough data for one class (churn class in our case). Hence, we will explore ways to cater to imbalanced datasets, by measures like undersampling, oversampling.

The dataset we used has 10,000 instances and there are 13 featu res like {'CreditScore', 'Geography', 'Gender', 'Age', 'Tenure', 'B al

ance', 'NumOfProducts', 'HasCrCard', 'IsActiveMember', 'Estim atedSalary','Exited'}.

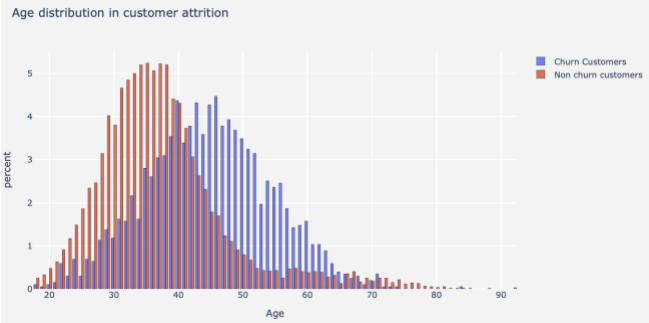
As the bank data source was Kaggle, due to absence of open so urce confidential data, there were no missing values, but in real life scenario, missing values are very important to be notes and made sense of, as to why and where are values missing, what ca n be done to impute those smartly, or should they be dropped? S uch decision often help in building good and robust model. We descried the features and see the max and min, standard deviatio n, maximum value available etc. Next step is to visualize the dis tribution of features with respect to churn and not churn (target) as well as with other features using a heatmap and note any imp ortant considerations, dependence or similarity which can be inf erred. Below are some of the visualizations and interpretations:

Heatmap is used to identify correlation amongst features and tar get values.

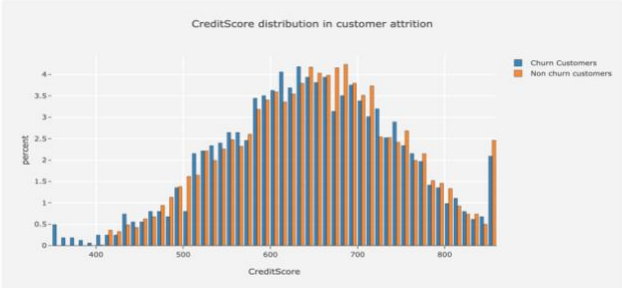


The plot suggests that the Age, Balance, and Estimated Salary positively correlates with target variable 'Exited' (churn) meani ng as value increases, probability of exited being 1 increases w hereas creditscore, Tenure, NumofProducts, HasCredit Card, i

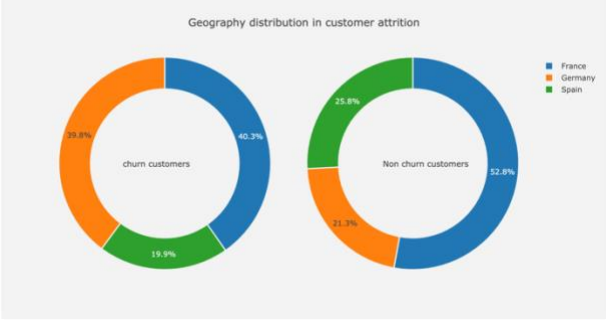
sActive member negatively correlates meaning the values of t
hese increase chances of
exiting goes down, ie exiting =0 increases.



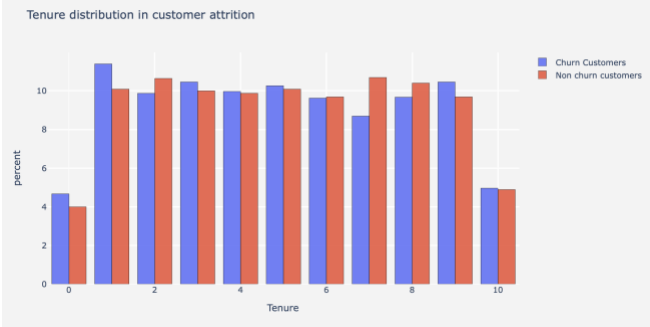
The distribution suggests that for age <40, middle or younger people are less probable of churning whereas middle aged people and senior members of society lie in the churning bucket. Hence, this could be an important demographic factor.



The above visualization is made based on Credit Score distribution. If we see the data, the percent of people churning when credit score is high is lesser than the percent of people not churning at similar credit score and vice versa for lower credit scores. Hence this also could be another deciding important factor for churning.



The above visualization depicts the percent of churning and non-churning people based on geographic location. France and Germany has maximum % of churners.

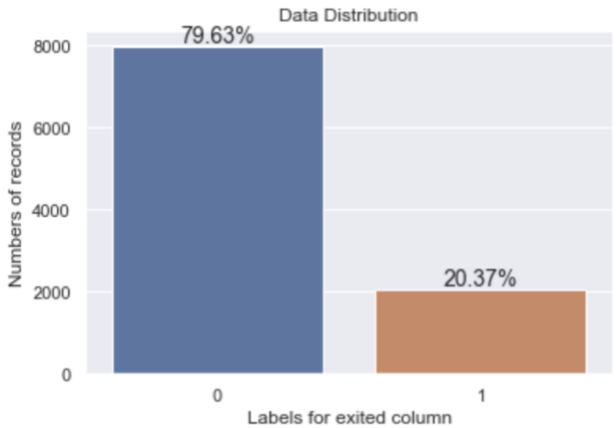


The above graph represents the Tenure distribution for the customers. There are no significant insights as the percent of people for every tenure bucket is comparable and not very distinct from each other.

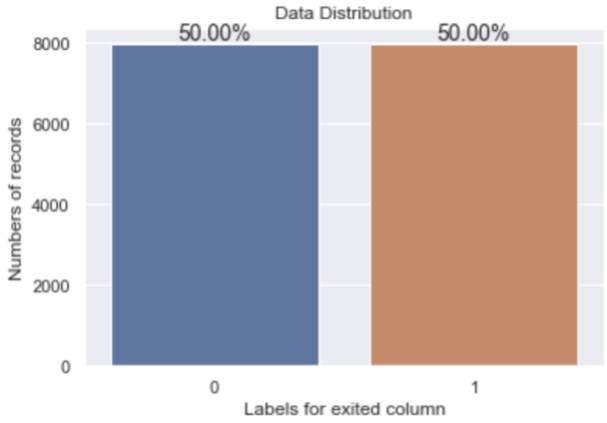
Data Preprocessing

Data preprocessing is pivotal in any information mining process as they legitimately sway achievement pace of the undertaking. This decreases intricacy of the information under investigation as information in real world is unclean. Information is said to be unclean on the off chance that it is missing values, outliers, contain commotion or anomalies and duplicate or wrong information. Nearness of any of these will corrupt nature of the outcomes.

Feature Selection broadly utilized in the pre-processing of ML which is alluded to choose P features from source include set of Q Features (P<Q). Feature Selection has been a prolific field of innovative work since 1980's and indicated extremely compelling in evacuating superfluous and repetitive. Our dataset comprises of 10,000 customers and 13 features which range from demographics to geography to their transactional details. We started by dropping 3 columns Index, CustomerId, Surname as they are not significant in the analysis. The data that we have is not balanced i.e there are unequal number of class labels. To make the class labels equal in proportion, up sampling of minority class was done using resample module. We then check the features for outliers. There are around 3000 rows where balance =0, while this makes sense for customers who have churned, this shouldn't be 0 for active customers whose estimated salary is comparatively high. We imputed these values by taking the average of balance to estimated salary values. Hence outliers were removed. Now, let us look at some graphs for balancing target variables,



Before sampling the data distribution is as show above:



After the up-sampling of data we get the minority class labels were increased.

Now the resampled data has 16000 customer rows. Next, to handle categorical features, one hot encoding instead of label encoding was done as label encoding misunderstands these labels as data values and eventually degrades model. Hence, one hot encoded all the categorical features. In this process we choose the categorical features like Geography, Gender and we added new columns as Geography_Spain, Geography_Germany and set it to binary values (0,1) As a result, we get all numerical values for categorical features. Next step is normalization of feature values. i.e. scaling the features and getting them to same scale between 0 and 1. This was done by Min Max Scalar. If feature values are not scaled some machine learning algorithms are sensitive to these values and they generate weight in such a way that features with high values are given more importance which might not be the case. After normalization our data is ready for modelling.

MODEL FOR CHURN PREDICTION

We have performed various supervised algorithms on our data set they are:
1)Logistic Regression being the base model 2) K Nearest Neighbors Classifier 3) Decision Trees 3) Random Forest 4) Support Vector Machine The paper then briefly describes the chosen algorithms and illustrates the results for churn.

1. Logistic Regression

Logistic regression is a classification algorithm often used as baseline model to set a benchmark. It suits well where our label is binary and categorical. LR uses predictive analysis to describe the tradeoff or relationship between a dependent binary variable and a set of independent variables. One drawback is that it doesn't handle collinearity and requires a large sample. It doesn't need the data to be linear in nature, it handles non linear relationships with the use of non linear log loss transformations. Here we get an accuracy of approximately 84%, recall of 76% and precision of 89%. We are more interested in recall than accuracy or precision as we need to correctly identify churners and stop then from churning by offering various marketing offers. We ran LR for different solvers along with regularization parameters to tune the model thus generated. The 'saga' solver with regularization parameter as elastic net with ratio = 0.9 performed the best.

```
solver liblinear : 0.6946421096693177
solver lbfgs : 0.6950606948514022
solver sag : 0.6950606948514022
solver newton-cg : 0.6950606948514022
solver saga : 0.6961071578066137
```

Confusion matrix as :

```
[[1695  655]
 [ 797 1631]]
```

And Classification report as :

	precision	recall	f1-score	support
0	0.68	0.72	0.70	2350
1	0.71	0.67	0.69	2428
accuracy			0.70	4778
macro avg	0.70	0.70	0.70	4778
weighted avg	0.70	0.70	0.70	4778

2. K Neighbors Classifier:

We have used the K Nearest Neighbors algorithm to see how good the model performs for this kind of data and the results are as follows:

```
neighbours  2 : score is 0.8359146086228547
neighbours  3 : score is 0.8359146086228547
neighbours  4 : score is 0.8359146086228547
neighbours  5 : score is 0.8359146086228547
[[1786  564]
 [ 220 2208]]
```

	precision	recall	f1-score	support
0	0.89	0.76	0.82	2350
1	0.80	0.91	0.85	2428
accuracy			0.84	4778
macro avg	0.84	0.83	0.83	4778
weighted avg	0.84	0.84	0.83	4778

Same results for 2,3,4,5 neighbours, accuracy is 84% and recall is 76%.

3. Decision Trees:

The basic structure of the decision tree consists of internal nodes and leaf nodes, where internal nodes checks certain conditions and splitting points and creates branches to reduce entropy. Leaf nodes that represent label values in our case: Exited status as 0 or 1. It supports categorical as well as continuous data.

```
[[1951  399]
 [ 507 1921]]
```

	precision	recall	f1-score	support
0	0.79	0.83	0.81	2350
1	0.83	0.79	0.81	2428
accuracy			0.81	4778
macro avg	0.81	0.81	0.81	4778
weighted avg	0.81	0.81	0.81	4778

Results suggest Recall is 83% and accuracy is 81%. Here we considered depth =9, minimum leaf=3, and min samples as 7 as optimal solution after checking for various values of params.

4. Support Vector Machine:

Support Vector Machine is a supervised learning technique and it is a discriminative classifier and is formally defined by the hyperplanes. It projects data into higher dimentions to separate them and form boundaries using support vectors and maximizing the distance between them. We used SVM for our bank dataset and the results are as below:

We tried fitting model for multiple type of kernels : linear, polynomial, radial, sigmoid and the results are as follows:

With linear kernel : we got 71% accuracy and recall 73%

```
[[1724  626]
 [ 785 1643]]
              precision    recall  f1-score   support

           0       0.69      0.73      0.71      2350
           1       0.72      0.68      0.70      2428

    accuracy                           0.70      4778
   macro avg       0.71      0.71      0.70      4778
weighted avg       0.71      0.70      0.70      4778
```

When the kernel is RBF and gamma is set to auto, we got 72% accuracy and recall 76%.

```
[[1788  562]
 [ 766 1662]]
              precision    recall  f1-score   support

           0       0.70      0.76      0.73      2350
           1       0.75      0.68      0.71      2428

    accuracy                           0.72      4778
   macro avg       0.72      0.72      0.72      4778
weighted avg       0.72      0.72      0.72      4778
```

When the kernel is sigmoid accuracy reduced to 65% and recall 71%.

```
[[1658  692]
 [ 970 1458]]
              precision    recall  f1-score   support

           0       0.63      0.71      0.67      2350
           1       0.68      0.60      0.64      2428

    accuracy                           0.65      4778
   macro avg       0.65      0.65      0.65      4778
weighted avg       0.65      0.65      0.65      4778
```

And with polynomial kernel, it reduced to very low values as we keep on increasing degrees from 2 to 8.

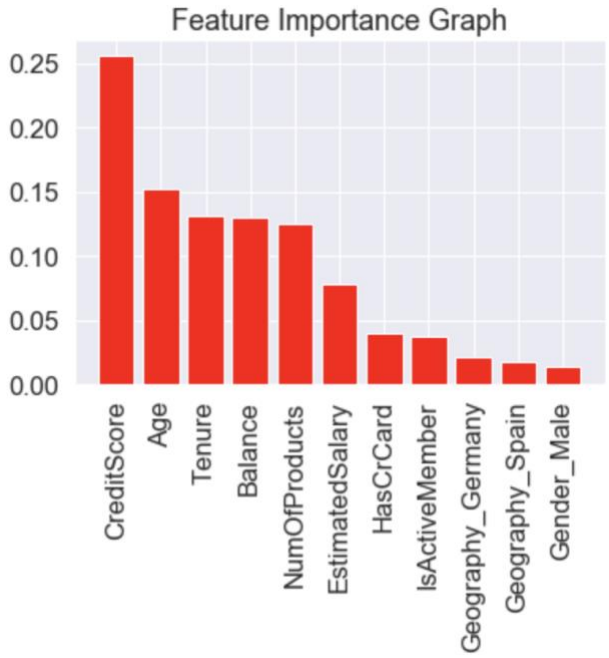
```
[[ 306 2044]
 [  94 2334]]
              precision    recall  f1-score   support

           0       0.77      0.13      0.22      2350
           1       0.53      0.96      0.69      2428

    accuracy                           0.55      4778
   macro avg       0.65      0.55      0.45      4778
weighted avg       0.65      0.55      0.46      4778
```

5. Random Forest:

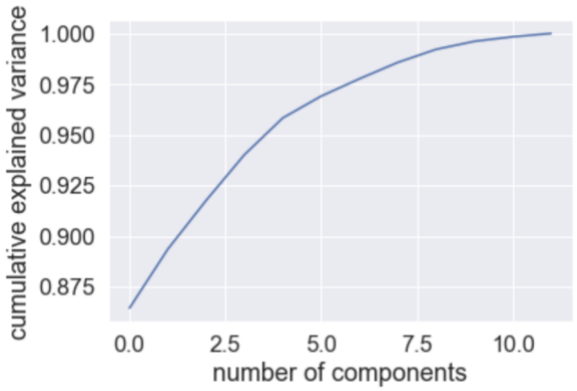
This is an Ensemble based model. Ensembles are a divide-and-conquer approach used to improve performance. In RF multiple decision tress are implemented together. The crux is that multiple weak learners together can be a strong learner and give better results. The weak learner in random forest are decision trees and multiple decision trees are formed using sampled data, then RF takes the decision which is supported by majority of the trees. It is used to reduce overfitting of decision trees by not depending on just 1 tree.
The performance increases as accuracy increased to 93% and recall increased to 91%. Also precision is increased to 96%. This tree was constructed by taking 100 decision trees.

```
[[2141  209]
 [  82 2346]]
              precision    recall  f1-score   support

           0       0.96      0.91      0.94      2350
           1       0.92      0.97      0.94      2428

    accuracy                           0.94      4778
   macro avg       0.94      0.94      0.94      4778
weighted avg       0.94      0.94      0.94      4778
```

The Feature Importance's are listed as below : we can see that credit score and age are top important feature as we guessed from data visualization.



We plot the ROC curve to see how the model perfoems compared to a random guess (red line), and the graph shows AUC score as 0.94 which validates that model is actually learning as compared to random guessing.



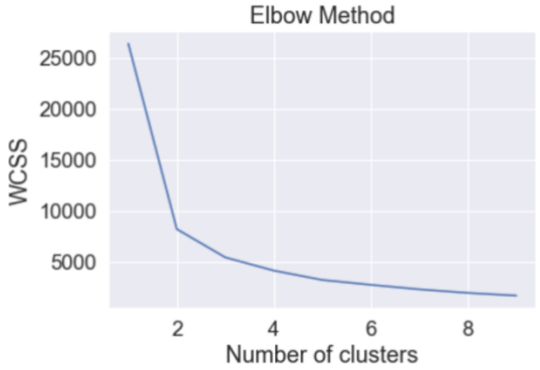
SEGMENTATION

Churn Customer Segmentation is a process of dividing a bank's customers, who are susceptible to churning, into separate clusters so as to find similarities amongst the people in each group and find appropriate ways to prevent them from churning.

Firstly, we separate all the rows whose exited value is 1 i.e we separate the people who are susceptible to churning as predicted by the model. Later we find out the optimal number of clusters that can be formed from the dataset using K means.

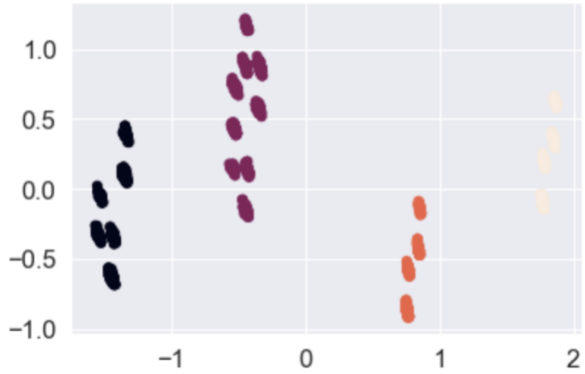
K Means algorithm is an iterative algorithm that
divides the dataset into K number of distinct subgroups or clusters, such that each data point belongs to just one cluster. The inter cluster distance is kept as low as possible while the clusters are kept far away from each other. A data point is assigned to a cluster to whose centroid it is the closest. We plot the below graph to determine the optimal number of clusters for our dataset.

We run the K means clustering on the dataset for various values of K ranging from 0 to 10 and choose the K value where the Within Cluster Sum of Squared Error begins to diminish. This is known as the elbow method. The elbow of the graph plotted above is 4. Elbow method is subjective in nature so we shall go ahead with 4 instead of 2.



We now perform Principal component analysis which reduces the dimensionality of the data and choose two features with the highest eigen values that covers nearly 90% variance in the data.

The scatterplot for the clusters hence formed, with these two principal components is shown below.



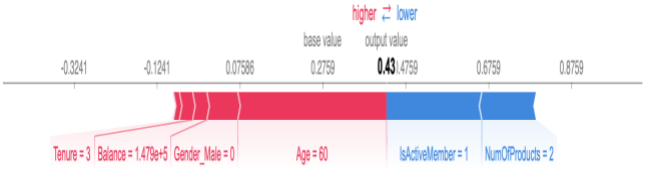
Cluster 1 : Black color
Cluster 2 : Purple color
Cluster 3 : Orange color
Cluster 4 : Crème color

EXPLAINABLE AI

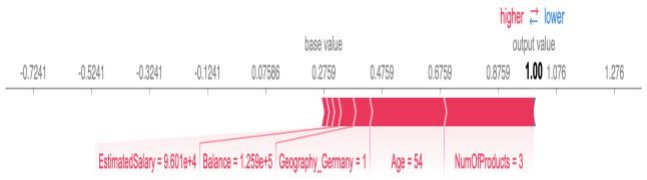
A lot of times, we build predictive models to solve very critical problems, and it's very crucial to understand why a model predicts one output as opposed to the other. Though we may understand all algorithms, some of them get extremely complicated by either the number of iterations or complexity of algorithm itself. Also visualizing and explaining this to stakeholders of bank to convince them to change a marketing policy etc. is difficult. But with the advent to explainable AI, we can now dig deeper into machine learning models and understand which features contribute and by how much to push a row to be predicted a certain label instead of other. Currently there are 2 state of art libraries that help us explain our ML models, SHAP and LIME. In comparison of both, SHAP is slower but eventually gives us an additive outcome of features which allow it to take a decision, whereas LIME is faster but takes only local decisions by generating linear models around

the data point of interest. We used SHAP to understand how our outputs are generated, what features are important, how is the overall effect of all features and unlike feature importance which tells us globally important features, SHAP helps us find for each test row, what values of features resulted into a particular decision. Lets us see some of the outputs and their explanability.

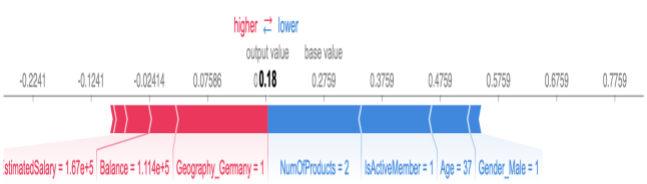
We can see the probability of churning for this customer is 0.43 ie eventually labelled as 0.
But now we can identify how this 0.43 is created using additive property of SHAP. Shap uses linear coefficients along with graph theory rules to generate and explainer which compared row wise feature values with background values.



For example, we saw that senior aged people are more likely to leave, hence age=60 contributes to the model in raising probability towards 1, but the member is an active member so less probable to leave hence this feature pushes the probability lower towards the left. Similarly for Gender and balance and tenure push customer to predict as churning but eventually probability rolls down to 0.43 meaning not churning.

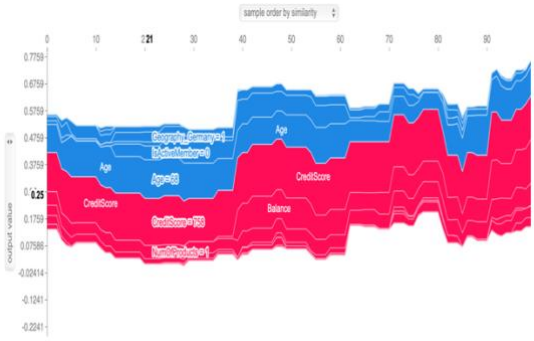


In the above example, we considered a test customer who was predicted to churn with a probability of 1. This was because, the values for features EstimatedSalary, Balance,Germany_geography,age,no of products=3 pushed the decision towards churning. We can see the extent of contribution also, age, products and geography=Germany being maximum.



Let us consider one more case, where probability is 0.18.
This is because, gender=male (unlike case 1 where gender =female and pushing probability higher), customer is young, products =2 and is an active member. So these outweigh the contributions of features increasing the probability and overall probability is lower in this case.

Now, lets see cumulative results for 100 such test cases, this gives us how probability is varying along with feature values.



RESULTS/CONCLUSION

In the conclusion, Random forest performed the best as compared to all other models with an accuracy of 93% and recall of 91%.

We used these results, probable churners to segment and understand customers similar to each other. In real life this can be scaled to a bank level, where big data tools can be used to analyze and understand the data. Moreover, once customer clusters are analyzed stakeholders and business can take decisions to provide personalized offers to retain customers and save revenue as well as customer worth. This real-life problem can also be extended to other domains wherein a similar approach can help companies know ahead of time about their customers and accordingly take actions.

LITERATURE SURVEY:

We came across many interesting articles for individual modules in our project. Of those, we particularly referred to some papers that stood out for our study and they have been briefly summarised as follows.

Churn Prediction

1) A Study On Customer Churn of Commercial Banks Based on learning from Label Proportions
https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8637415
Problem Studied – Prediction of customer churn for the bank is the main topic of research for this paper. With the available data from the customers, results are predicted with the better set of features from the available ones. However, the paper lacks sufficient amount of training data and labels for each segment of customers. Problem Solved – Customer churn for the bank in question is studied using logistic regression and support vector machine

2) Customer churn prediction in banking
https://ieeexplore.ieee.org/document/7130361

3) Predicting Credit Card Holder Churn in Banks of China Using Data Mining and MCDM
https://dl.acm.org/citation.cfm?id=1913933

4) Predicting Customer Churn: Extreme Gradient Boosting with Temporal Data
https://arxiv.org/abs/1802.03396

Machine Learning Algorithms

1) Applying the CG-logistic Regression Method to Predict the Customer Churn Problem
https://ieeexplore.ieee.org/document/8597855

Problem Studied – Prediction of customer churn for airlines. With the ever increasing population travelling by air, there is a need to retain old customers while attracting potential future customers. Churn conditional probability and likelihood methods are used in this paper to design the algorithm.

Problem Solved – A double calculated relapse model and bunch stratified examining strategic relapse is utilized for the airline's issue, the numerical models show the viability of the models in correlation with other customary techniques. CG-logistic regression model is used which in turn uses logistic regression algorithm as its foundation.

Explainable AI:

1) From Machine Learning to Explainable AI

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8490530

Problem Studied – Frameworks that explain the implementation of ML models is studied in this paper as most of the data in a model is unknown and abstract. The approach followed is by explaining the procedure and steps that contributed to the obtained results (what exactly helped in predicting things at first place).

Problem Solved - Challenges like uncertainty, interactive machine learning and its application are catered to. The paper aims to put the results in a language that helps humans out of the technical space to understand the working of the ML model and how the real-time data is proportional to the accuracy of the model.

2) Toward Human-Understandable, Explainable AI
https://ieeexplore.ieee.org/document/848125

Class Imbalance

1) Comparing Oversampling Techniques to Handle the Class Imbalance Problem: A Customer Churn Prediction Case Study
https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7707454

Problem Studied – Caters to the problem of imbalance in the classes of customer data leading to either undersampling or oversampling. As a result, customer churn prediction is more accurate once the imbalance in classes is minimized or in some cases removed altogether. Problem Solved - Rule generation algorithm is used on the existing public data sets to solve the problem under consideration.

2) Investigating Decision Tree in Churn Prediction with Class Imbalance
http://delivery.acm.org/10.1145/3230000/3224217/p11-Zhu.pdf?ip=130.65.254.6&id=3224217&acc=ACTIVE%20SERVICE&key=F26C2ADAC1542D74%2ED0BD0A8C52906328%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&__acm__=1569347599_a16f584b1bfc3f9b88db0aea6d0aa5f

REFERENCES

1. Xing E., Jordan M., Karp R, "Feature selection for high-dimensional genomic microarray data," Proceedings of the Eighteenth International Conference on Machine Learning, Massachusetts, USA: Morgan Kaufmann, 2001, 601–608.

2. Yang Yiming and Pederson J O, "A Comparative Study on Feature Selection in Text Categorization," Proceedings of the 14th International Conference on Machine Learning, Nashville: Morgan Kaufmann, 1997, 412-420.

3. Rui, Y., Huang, T. S. and Chang, S, "Imageretrieval: current techniques, promising directions and open issues," Journal of Visual Communication and Image Representation, 1999, 10: 39–62.

4. Ng, K. and Liu, H, "Customer retention via data mining," AI Review, 2000, (14): 569-590.

5. Lemmens, A. and Croux, C., "Bagging and boosting classification trees to predict churn," Journal of Marketing Research, 2006, 43(2): 276-286.

6. Qian, S. L., He, J., M. and Wang, C., L., "Telecom Customer Churn Prediction Model Based on Improved SVM," Journal of Management Sciences, 2007, 20(1):54-58.

7. Wang, R. and Chen, C. "The application of attribute selection based on data mining to churn predictive model," Computer Applications and Software, 2007, 24(11): 98-113.

8. Meng, X. L., Cai, S. Q. and Du, K. Q., "A Study on a Predictive Model of Customer defection in a Commercial Bank," Systems Engineering, 2004, 22(12): 67-71.

9. Blum, A. and Langley, P., "Selection of relevant features and examples in machine learning," Artificial Intelligence, 1997, 97(1-2): 245–271.

10. Dash, M. and Liu, H., "Feature selection for classifications," Intelligent Data Analysis: An International Journal, 1997, (1): 131– 156.

11. Kohavi, R. and John, G., "Wrappers for feature subset selection," Artificial Intelligence, 1997, 97(1-2): 273–324.

12. Guyon I., Weston J., Barnhill S. and Vapnik V., "Gene selection for Cancer Classification using Support Vector Machines," Machine Learning, 2002, 46(1/3): 389-422.

APPENDIX

```
# project preprocessing

import statsmodels.api as statsmodel
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.graphics.gofplots import qqplot_2samples
from statsmodels.graphics.gofplots import qqplot
get_ipython().run_line_magic('matplotlib', 'inline')
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
import plotly.graph_objs as po
import plotly.offline as py
from sklearn import tree
#from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import roc_auc_score
from sklearn.tree import export_graphviz
from sklearn.tree import plot_tree
import plotly.offline as py
py.init_notebook_mode(connected=True)
import plotly.graph_objs as go
import plotly.tools as tls
import plotly.figure_factory as ff


df=pd.read_csv('Churn_Modelling.csv')
print(df.head())

df.shape


df.drop(['RowNumber','CustomerId','Surname'],axis=1,inplace=True)
df.head()

testdf=df[df.Exited==0]
testdf.IsActiveMember.value_counts()
#around 3547 people are not active and not exited: active is subjective (how active what's the frequency etc)
#these are probably at higher risk of leaving.

testdf=df[df.Exited==0]
bal0rows=testdf[testdf.Balance==0]
print(bal0rows.shape)
# 3117 people ie 1/3rd people  not exited have a balance of 0

# cant go with this. need to impute these values.
bal0activerows=bal0rows[bal0rows.IsActiveMember==0]
print(bal0activerows.shape)
# 1424 people ie 15percent people who have not exited and are inactive have a balance of 0 :
# we can assume :: higher probability of leaving

testdf['Balance'].hist(bins=6)

testdf['EstimatedSalary'].hist(bins=6)




#at what balance did people exit bank;
testdf=df[df.Exited==1]
```

```
#testdf.Balance.hist()
testdf.Balance.mean()
# when exited. was the number of products 0?
testdf.NumOfProducts.hist()
# peope who exited had 1 product
# (affinity of leaving when only 1 product is more as probably: not keen in other products.)

#lets check what percentage of balance is estimated salary.
testdf=df[df.Balance!=0]
testdf['balance_percent_estimatedSalary']= (testdf['Balance']/testdf['EstimatedSalary'])
testdf.head()

testdf.boxplot(column=['EstimatedSalary'])
#outier

testdf.boxplot(column=['EstimatedSalary'])
#outier

#change estimated salary lower than 250000 to 250000.
df['EstimatedSalary']=np.where(df['EstimatedSalary']<25000,25000,df['EstimatedSalary'])

df.boxplot(column=['EstimatedSalary'])

df.boxplot(column=['Balance'])
testdf.boxplot(column=['Balance'])

#dont change balance as actual figures not estimated.
#only change where balance =0 by

testdf.boxplot(column=['balance_percent_estimatedSalary'])

# 6 % values are outliers so we cap it to a lower bound = 5%
testdfbal=testdf
testdfbal['balance_percent_estimatedSalary']=np.where(testdfbal['balance_percent_estimatedSalary']>5,5,testdfbal['balance_percent_estimatedSalary'])
testdfbal.boxplot(column=['balance_percent_estimatedSalary'])

testdfbal['balance_percent_estimatedSalary'].mean()

# for the balance which are 0 and are active and not exited , we will impute balance as 1.7 times* Estimated Salaries.
fil1=df['Balance']==0
fil3=df['IsActiveMember']==1
fil2= df['Exited']==0

#what is the avg of this balance_percent_estimatedsalary
avg_percent_f=testdf[testdf.Gender=='Female']
bins= [0,30,50,70,100,130,160,200,250,300,350,400,1000]
avg_percent_f = avg_percent_f.groupby(pd.cut(avg_percent_f['balance_percent_estimatedSalary'], bins=bins)).balance_percent_estimatedSalary.count()
avg_percent_f.plot(kind='bar')
plt.show()

#what is the avg of this balance_percent_estimatedsalary
avg_percent_f=testdf[testdf.Gender=='Male']
#avg_percent_f['balance_percent_estimatedSalary'].mean()
avg_percent_f.balance_percent_estimatedSalary.hist(bins=2)

#highly unbalanced dataset,
#lets describe data
df.describe()
#means no missing values, has 10000 values for every column,

# we are diving the  data into training set and test set the ration as 70/30 format
#train, test = train_test_split(df, test_size=0.3)

# to show there are no missing values
df.isnull().sum()

# visualize the dimension of training data
df.shape

# check the unique values of each attribute in the training data
df.nunique()

#No of Exited vs Active  get the percentage split figure
ExitedValues = df.Exited.value_counts()
labels = ["Loyal Customer","Churn Customers"]
colors = ['#66b3ff', '#ff9999']
fig1, f1 = plt.subplots()
f1.pie(ExitedValues,labels=labels, colors = colors, autopct='%1.1f%%',shadow=True, startangle=60)
f1.axis('equal')
plt.tight_layout()
plt.show()
```

```
#Implies dataset is imbalanced

#In the  bankdata we have biased distribution i.e is 80% custoemrs are loyal and 20% are not so we need to make the data set balanced

plt.hist(df['CreditScore'])
plt.xlabel('Credit Score Distribution')
plt.ylabel('Num of Customers')
plt.title('CreditScore')

# Nearly 55% of the customers creditscore is aroound 600-750.

plt.hist(df['Balance'])
plt.xlabel('Account Balance Distribution')
plt.ylabel('Num of Customers')
plt.title('Balance')

## for the balance which are 0
#we will impute balance as 1.7 times* Estimated Salaries.
df['Balance']=np.where(df['Balance']==0,(df['EstimatedSalary']*1.7),df['Balance'])

#and are active and not exited

# this plot is to show how Geography play a role at the customer churn
Geosplit = train.Geography.value_counts()
Geovalues  = df['Geography'].value_counts().values.tolist()
Geolabels  = df['Geography'].value_counts().keys().tolist()
colors = ['#66b3ff', '#ff9999' , '#ffcc99']
fig2, f2 = plt.subplots()
f2.pie(Geovalues,labels=Geolabels, colors = colors, autopct='%1.1f%%',shadow=True, startangle=90)
# Equal aspect ratio ensures that pie is drawn as a circle
f2.axis('equal')
plt.tight_layout()
plt.title('Percentage split based on Geography')
plt.show()

# this plot is to show how Gender play a role at the customer churn stuff
Gendervalues  = df['Gender'].value_counts().values.tolist()
GenderLabels = df['Gender'].value_counts().keys().tolist()
colors = ['#66b3ff', '#ff9999']
fig3, f3 = plt.subplots()
f3.pie(Gendervalues,labels=GenderLabels, colors = colors, autopct='%1.1f%%',shadow=True, startangle= 90)
# Equal aspect ratio ensures that pie is drawn as a circle
f3.axis('equal')
plt.title('Percentage split based on Gender')
plt.tight_layout()
plt.show()

sns.boxplot(data= df['Tenure'], orient="v")
#i  feel this is not needed

# this plot is to show how HasCrCard play a role at the customer churn
HasCardvalues  = train['HasCrCard'].value_counts().values.tolist()
HasCardlabels  = ["Having Card" , "No Card"]
colors = ['#99ff99','#ffcc99']
fig5, f5 = plt.subplots()
f5.pie(HasCardvalues ,labels=HasCardlabels, colors = colors,autopct='%1.1f%%',shadow=True, startangle=60)
f5.axis('equal')
plt.title('Percentage split based on Card Possession')
plt.tight_layout()
plt.show()

# lets have the plot for customers based on their age
sns.boxplot(df['Age'] , orient = "v")

#400 people are senior members ; most of them lie in middle aged group.

# let's see the correlation matrix of the data
df[df.columns].corr()


# generate the heatmap with the above data so that we can see what colums are corelated with the other one graphically
fig, ax = plt.subplots()
fig.set_size_inches(11.7, 8.27)
sns.set(font_scale = 0.75)
sns.heatmap(df.corr(), annot = True, fmt = ".6f")
plt.show()


# The above data says the Age, Balance, and Estimated Salary positively correlates with Exited meaning as value increases exited being 1 is more probable
# whereas  creditscore,Tenure, NumofProducts, Hs Credit Card, isACtive member negatively correlates meaning the values of these increase chances of exiting lessens
```

```
# balance attribute is negatively correlated with has credit card attribute.
#It means chances of getting credit card depends on balance as well, which reflects real life scenario.

#Separating churning  and noncharging customers
churn     = df[df["Exited"] == 1]
not_churn = df[df["Exited"] == 0]

#Let's Visulaize the data with respective to Exited column and numerical and categorical columns

target_column = ["Exited"]
cat_columns   = df.nunique()[df.nunique() < 5].keys().tolist()

cat_columns   = [x for x in cat_columns if x not in target_column]

num_columns   = [x for x in df.columns if x not in target_column + cat_columns ]

hist_visulalization(num_columns[1])
# as we deal with elder customers (in terms of age) their is a risk of losing them,

#plotting the pie plot for gender column
plot_visualization(cat_columns[1])

# Calling the function for plotting the histogram for tenure column
hist_visulalization(num_columns[2])

#   plotting the pie plot for gender column 70% of the people who leave,  do not
plot_visualization(cat_columns[3])

# Graphical representation of the target label percentage before upsampling
total_len = len(df['Exited'])
sns.set()
sns.countplot(df.Exited).set_title('Data Distribution')
ax = plt.gca()
for p in ax.patches:
   height = p.get_height()
   ax.text(p.get_x() + p.get_width()/2.,
         height + 2,
         '{:.2f}%'.format(100 * (height/total_len)),
         fontsize=14, ha='center', va='bottom')
sns.set(font_scale=1.5)
ax.set_xlabel("Labels for exited column")
ax.set_ylabel("Numbers of records")
plt.show()

df['Exited'].value_counts()

from sklearn.utils import resample

#upsampling minority class to match to majority class
df_majority = df[df.Exited==0]
df_minority = df[df.Exited==1]
df_minority_upsampled = resample(df_minority, replace=True,     # sample with replacement
                           n_samples=7963,    # to match majority class
                           random_state=123) # reproducible results
df_upsampled = pd.concat([df_majority, df_minority_upsampled])
df_upsampled.Exited.value_counts()

# Graphical representation of the target label percentage after balancing minority class
total_len = len(df_upsampled['Exited'])
sns.set()
sns.countplot(df_upsampled.Exited).set_title('Data Distribution')
ax = plt.gca()
for p in ax.patches:
   height = p.get_height()
   ax.text(p.get_x() + p.get_width()/2.,
         height + 2,
         '{:.2f}%'.format(100 * (height/total_len)),
         fontsize=14, ha='center', va='bottom')
sns.set(font_scale=1.5)
ax.set_xlabel("Labels for exited column")
ax.set_ylabel("Numbers of records")
plt.show()

# apply baseline model as logistic clasiification. As the problem is of classification,
# we will apply decision trees and random forest, and svm to the problem to check the accuracy later

df_new = df_upsampled

df_new.dtypes
```

```
df_new=pd.get_dummies(df_new,drop_first=True)

df_new.head()

df_new.columns

df_ex_ai=df_new

scale_down_column=pd.DataFrame(df_new[['CreditScore','Age','Tenure','Balance','NumOfProducts','EstimatedSalary']])

for col in scale_down_column.columns:
    scale_down_column[col]=scale_down_column[col].astype('float64')

scale_down_column.dtypes

type(scale_down_column)


from sklearn.preprocessing import MinMaxScaler
scalar=MinMaxScaler()

scalar.fit(scale_down_column)
scale_down_column1=scalar.transform(scale_down_column)

scaled_df = pd.DataFrame(scale_down_column1, columns=scale_down_column.columns)
scaled_df.head()

scaled_df.shape

df_new_half=df_new[[ 'HasCrCard',
       'IsActiveMember', 'Geography_Germany',
       'Geography_Spain', 'Gender_Male','Exited']]
df_new_half.shape


df_new_half=df_new_half.reset_index(drop=True)
df_new_half.head()

scaled_df=scaled_df.reset_index(drop=True)
scaled_df.head()

df_concat=pd.concat([scaled_df,df_new_half],axis=1)
df_concat.head()

df_concat.shape

df_new=df_concat

scalar=StandardScaler()
scaled_df = scalar.fit_transform(df_new)
scaled_df = pd.DataFrame(scaled_df, columns=df_new.columns)
scaled_df.head()

df_y=df_new.loc[:,'Exited']
df_x=df_new.drop('Exited',axis=1)

xtrain,xtest,ytrain,ytest=train_test_split(df_x,df_y,test_size=0.3, random_state=1)

ytest=ytest.reset_index(drop=True)

len(ytest)

#Logistic Regression used as base model.

for solver in ['liblinear','lbfgs','sag','newton-cg']:
    lr=LogisticRegression(penalty='l2',solver=solver).fit(xtrain, ytrain)
    ypredicted=pd.Series(lr.predict(xtest))
    print('solver',solver, ':',accuracy_score(ytest,ypredicted))


for solver in ['saga']:
    lr=LogisticRegression(penalty='elasticnet',solver=solver,l1_ratio=0.8).fit(xtrain, ytrain)
    ypredicted=pd.Series(lr.predict(xtest))
    print('solver',solver, ':',accuracy_score(ytest,ypredicted))

print(confusion_matrix(ytest, ypredicted))
print(classification_report(ytest, ypredicted))

#baseline predictions using Logistic Regression are around 70% accuracy score
#Recall is 0.72
#F1 is 0.70
```

```
# To get the weights of all the variables
weights = pd.Series(lr.coef_[0],index=xtrain.columns.values)
weights.sort_values(ascending = False)

for depth in [3,5,9,10]:
    for minsamples in [2,3,4]:
        for minleaf in [3,5,7]:
            dtree=DecisionTreeClassifier(max_depth=depth, min_samples_split=minsamples, min_samples_leaf=minleaf).fit(xtrain,ytrain)
            ypredicted=pd.Series(dtree.predict(xtest))
            print( 'with depth ',depth,'minsamples',minsamples, 'minleaf',minleaf, 'accuracy score is',accuracy_score(ytest,ypredicted))
print(confusion_matrix(ytest, ypredicted))
print(classification_report(ytest, ypredicted))


#with depth =9 min leaf =3 minsamples =2 we get optimal value of accuracy score. increasing these values too much can lead to overfitting so we just stop at points where we get optimal results

from sklearn.neighbors import KNeighborsClassifier
for neighbour in [2,3,4,5]:
    neigh = KNeighborsClassifier(n_neighbors=3,metric='minkowski',p=2)

    neigh.fit(xtrain,ytrain)

    ypredknn=pd.DataFrame(neigh.predict(xtest))
    print('neighbours ',neighbour, ': score is',accuracy_score(ytest,ypredknn))
print(confusion_matrix(ytest, ypredknn))
print(classification_report(ytest, ypredknn))

#same score so can take any, accuracy is 0.8359

#Random Forest model and generate the importance of the features
features_label = xtrain.columns
for estimator in [10,50,100]:
    forest = RandomForestClassifier (n_estimators = estimator, random_state = 23)
    forest.fit(xtrain, ytrain)
    importances = forest.feature_importances_
    indices = np.argsort(importances)[::-1]
    print("For estimator:",estimator)
    for i in range(xtrain.shape[1]):
        print("%2d) %-*s %f" % (i + 1, 30, features_label[i], importances[indices[i]]))

#best for n=100 trees, so lets go with it

# Visualization of the Feature importances of each one
plt.title('Feature Importance Graph')
plt.bar(range(xtrain.shape[1]), importances[indices], color = "red", align = "center")
plt.xticks(range(xtrain.shape[1]), features_label, rotation = 90)
plt.show()

ypredicted=pd.Series(forest.predict(xtest))
print(accuracy_score(ytest,ypredicted))

print(confusion_matrix(ytest, ypredicted))
print(classification_report(ytest, ypredicted))

ytest.value_counts()

# Random forest on unscaled data : for explainable Ai
# df_ex_ai data frame.
df_new_y=df_ex_ai.loc[:,'Exited']
df_new_x=df_ex_ai.drop('Exited',axis=1)

xtrain1,xtest1,ytrain1,ytest1=train_test_split(df_new_x,df_new_y,test_size=0.3, random_state=1)
features_label = xtrain1.columns
for estimator in [100]:
    forest1 = RandomForestClassifier (n_estimators = estimator, random_state = 23)
    forest1.fit(xtrain1, ytrain1)
    importances = forest1.feature_importances_
    indices = np.argsort(importances)[::-1]
    print("For estimator:",estimator)
    for i in range(xtrain1.shape[1]):
        print("%2d) %-*s %f" % (i + 1, 30, features_label[i], importances[indices[i]]))
plt.title('Feature Importance Graph')
plt.bar(range(xtrain1.shape[1]), importances[indices], color = "red", align = "center")
plt.xticks(range(xtrain1.shape[1]), features_label, rotation = 90)
plt.show()
ypredicted1=pd.Series(forest1.predict(xtest1))
print(accuracy_score(ytest1,ypredicted1))
```

```
from sklearn.svm import SVC

svmpoly=SVC(kernel='linear',verbose=False, max_iter=-1)
svmpoly.fit(xtrain,ytrain)
y_pred = svmpoly.predict(xtest)
print(confusion_matrix(ytest, y_pred))
print(classification_report(ytest, y_pred))


svmpoly=SVC(kernel='rbf', gamma='auto',verbose=False, max_iter=-1)

svmpoly.fit(xtrain,ytrain)
y_pred = svmpoly.predict(xtest)
print(confusion_matrix(ytest, y_pred))
print(classification_report(ytest, y_pred))

svmpoly=SVC(kernel='sigmoid',verbose=False, max_iter=-1)

svmpoly.fit(xtrain,ytrain)
y_pred = svmpoly.predict(xtest)

print(confusion_matrix(ytest, y_pred))
print(classification_report(ytest, y_pred))

for degree in [2,3,4,5,6,7]:
    svmpoly=SVC(kernel='poly', degree=degree,verbose=False)
    svmpoly.fit(xtrain,ytrain)
    y_pred = svmpoly.predict(xtest)
    print(confusion_matrix(ytest, y_pred))
    print(classification_report(ytest, y_pred))

#After testing all of these, we conclude that random forest performs the best, hence lets see the ROC/AUC graph :
from sklearn import metrics
import matplotlib.pyplot as plt
fpr, tpr, thresholds = metrics.roc_curve(ytest, ypredicted)
roc_auc = metrics.auc(fpr, tpr)
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
xtest1=pd.DataFrame(xtest)
y_pred1=pd.DataFrame(ypredicted)
xtest1=xtest1.reset_index(drop=True)
y_pred1=y_pred1.reset_index(drop=True)
testsvm=pd.concat([xtest1,y_pred1],axis=1)
testsvm.head()
testsvm.shape

testsvmonly_one = testsvm[testsvm.iloc[:,-1] == 1]
testsvmonly_one.shape

testsvmonly_one.head()

testsvmonly_one.columns

churn =testsvmonly_one
churn.drop(0,inplace=True,axis=1)
churn.head()

from sklearn.cluster import KMeans
wcss = []
for i in range(1, 10):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=0)
    kmeans.fit(churn)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 10), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()

fig = plt.figure(figsize=(8, 6))
#t = fig.suptitle('Wine Residual Sugar - Alcohol Content - Acidity - Total Sulfur Dioxide - Type - Quality', fontsize=14)
ax = fig.add_subplot(111, projection='3d')

clusterlist=[2,3,4,5,6,8,10]
```

```
for c in clusterlist:
    km= KMeans(n_clusters=c)
    clusters=km.fit_predict(churn)
    churn['clusters']=cluster
    #print(churn)

    from sklearn.decomposition import PCA
    pca = PCA(2,random_state=1)

    plot_columns = pca.fit_transform(churn)
    #print(plot_columns)
    plt.scatter(x=plot_columns[:,0], y=plot_columns[:,1],  c=churn["clusters"])
    plt.show()

    # Plot based on the two dimensions, and shade by cluster label
    #ax.scatter(xs=plot_columns[:,0], xy=plot_columns[:,1],ys=plot_columns[:,2],marker='o', c=churn["clusters"], s=30)
    #plt.show()

    #print(km.cluster_centers_)

print(pca.components_)
print(pca.explained_variance_)
pca.explained_variance_ratio_


pca = PCA().fit(churn)
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance');
clusterlist=[4]
for c in clusterlist:
    km= KMeans(n_clusters=c)
    clusters=km.fit_predict(churn)
    churn['clusters']=clusters

    #print(churn)
    from sklearn.decomposition import PCA
    pca = PCA(2,random_state=1)

    plot_columns = pca.fit_transform(churn)
    #print(plot_columns)
    plt.scatter(x=plot_columns[:,0], y=plot_columns[:,1],  c=churn["clusters"])
    plt.show()
# Explainable AI ?
import shap
shap.initjs()
X_train_summary = shap.kmeans(xtrain1, 10)

explainer = shap.KernelExplainer(model=forest1.predict_proba,data=X_train_summary,model_output="margin" )
shap_values = explainer.shap_values(xtest.iloc[:100])
shap.force_plot(explainer.expected_value[1], shap_values[1], xtest1.iloc[:100])
# very likely to churn, probability of churning is 1
xtest1.iloc[2]
shap_values = explainer.shap_values(xtest1.iloc[2])
shap.force_plot(explainer.expected_value[1], shap_values[1], xtest1.iloc[2])
# not likely to churn, probability of churning is 0.43
xtest1.iloc[4777]
shap_values = explainer.shap_values(xtest1.iloc[4777])
shap.force_plot(explainer.expected_value[1], shap_values[1], xtest1.iloc[4777])

# not likely to churn, probability of churning is 0.18
xtest1.iloc[2356]

shap_values = explainer.shap_values(xtest1.iloc[2356])
shap.force_plot(explainer.expected_value[1], shap_values[1], xtest1.iloc[2356]
```