CUSTOMER CHURN PREDICTION

PROJECT REPORT

PROFESSOR: Xiao, Ting

(Ting.Xiao@unt.edu)

Project Members

NAME & ID	MAIL ID	ROLE
HARSHAVARDHAN BHUPATHI **(11514028)**	HarshaVardhanBhupathi@my.unt.edu	Data Modeling and Analysis/Evaluation
VAMSHI KUMAR KONDURU **(11516045)**	VamshiKumarKonduru@my.unt.edu	Data Modeling and Analysis/Evaluation
SANJAY REDDY MANDA **(11524794)**	SanjayReddyManda@my.unt.edu	Data Collection
DILIP KUMAR PITTALA **(11515093)**	DilipKumarPittala@my.unt.edu	Data Preparation

The Team meeting/Discussion will take place in Willis Library and Discovery Park Library on Tuesdays (12:30 pm to 5 pm) and Saturdays (6:00 pm to 10:00 pm) in-person or else through Zoom meeting Through the following Link:

https://us04web.zoom.us/j/7867491946?pwd=ZXZ2Q3Z3QzJiOHhOcnZYenJMKzlqQT09

TABLE OF CONTENTS

PROJECT ABSTRACT:

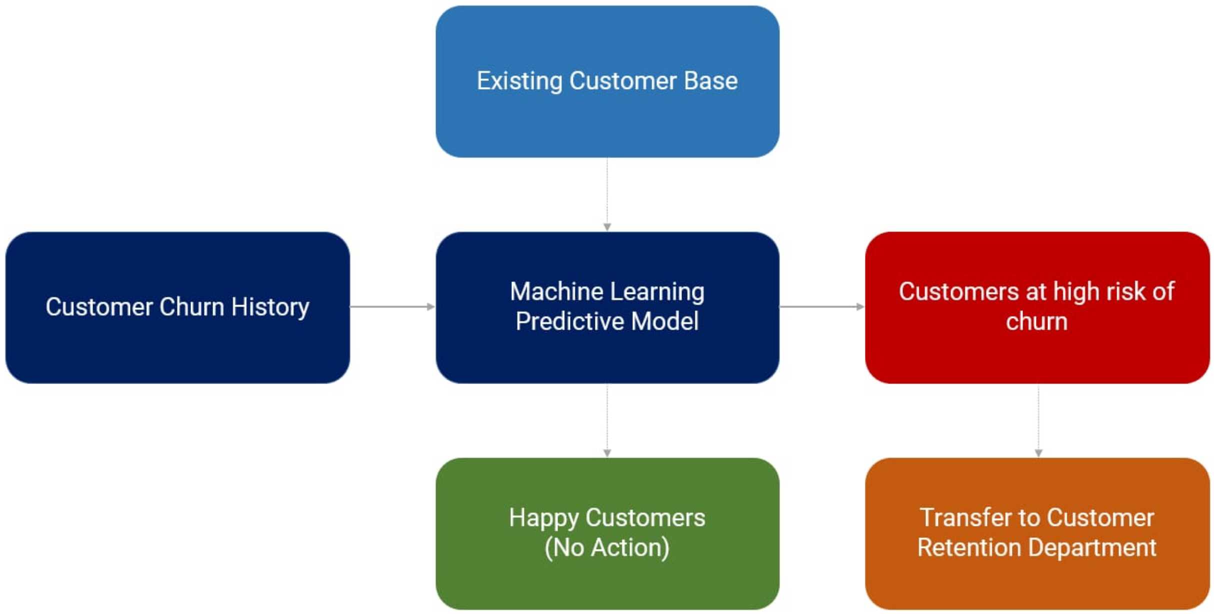
Customer Churn is a financial term used to indicate the loss of customers or clients, which happens when customers cease showing interest in our business or leave the business to join the Competitors. The rate at which companies lose their subscribers at specific periods such as monthly, quarterly, or annually is called the churn rate. So, any company/business needs to identify/predict these potential customers who are willing to cease doing business in the future. Due to the increase in technology, and demand the Banks are having high competition and have started showing interest in their customers' needs and demands. So, we will be performing the Binary classification task using different Classification Machine learning Algorithms for the customer churn prediction. The prediction involves the classification of churn and non-churn customers. This prediction would help retain the customers by proactive engagement/interaction with those sets of customers. This project results in displaying the best model, which classifies the churn prediction of customers well.

DATA SPECIFICATION

Predictor Variables	Type	Variable Details
RowNumber	Int64	The row number in the dataset.
CustomerId	Int64	The customer ID is different for each customer.
Surname	Object	The last name of the customer.
CreditScore	Int64	The credit score of the customer.
Geography	Object	The location of the customer.
Gender	Object	The gender of the customer.
Age	Int64	The Age of the customer.
Tenure	Int64	The number of years the customer has been a member of the Bank.
Balance	Float64	The balance the customer possesses in his account.
NumOfProducts	Int64	The number of products the customer has bought through the bank.
HasCrCard	Int64	If the customer has a credit card or not.
IsActiveMember	Int64	If the customer is actively using the account for any purchases or deposits.
EstimatedSalary	Float64	The estimated Salary of the customer.
Exited	Int64	If the customer has left the bank.

The data set comprised 10,000 customers details from three different countries that are Germany, France, and Spain. The tenure is of years from 0 to 10 years. We performed a binary classification task using Supervised learning Classification ML Algorithms to predict the customers who are willingly churned out of the bank.

WORKFLOW:



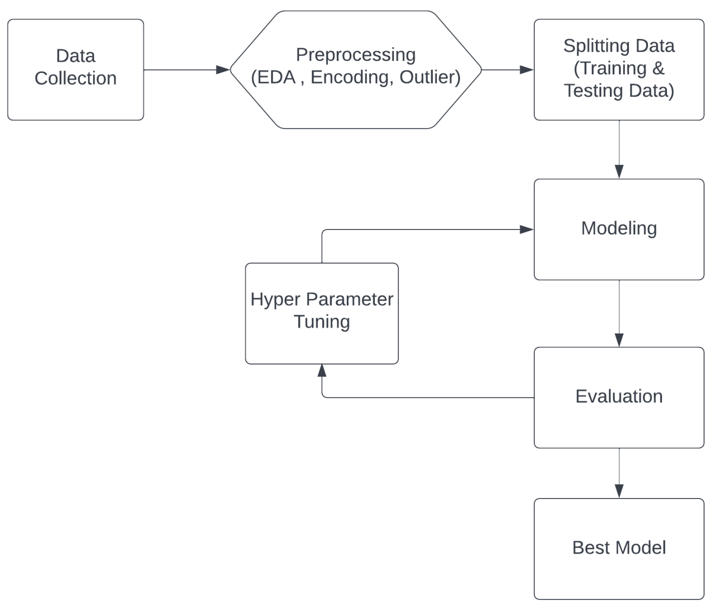
Based on the customer churn history, we built a prediction model that predicts whether the customer is willing to be churned or not. If customer is likely to be churned, the data of such customers will be sent to customer retention department which involves proactive engagement to retain customers.

PROJECT DESIGN

TOOLS AND FRAMEWORKS:

Tools: Python Jupyter Notebook, Visual Studio Code.
Frameworks: React JS, MD Bootstrap, Netlify Hosting Site
Libraries:

- Pandas – data analysis and associated manipulation of tabular data in Data frames,
- NumPy – Mathematical Operations for arrays,
- Scikit Learn – used for predictive data analysis,
- Matplotlib – Used for creating static, animated, and interactive visualizations in Python,
- Seaborn – Python data visualization library based on matplotlib



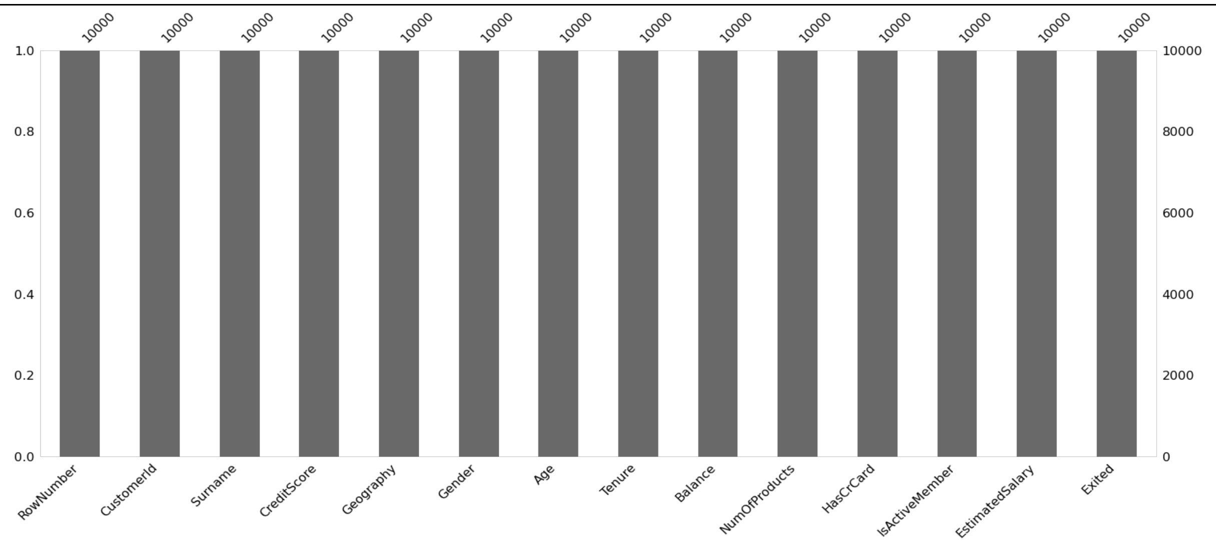
DATA COLLECTION:

The data set is gathered to perform analysis and modeling for the prediction of the best model for Bank customer churn sourced from Kaggle.

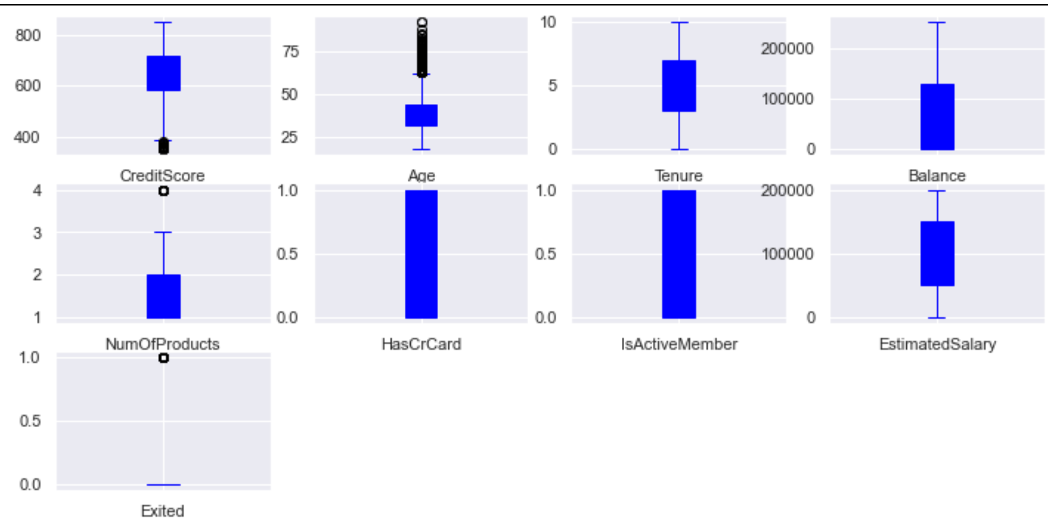
DATA PREPROCESSING:

EXPLORATORY DATA ANALYSIS:

In EDA, the data gathered is analyzed through visuals such as bar graphs, Line plots, histograms, box plots, heat maps, Density plots, and some statistical decisions were made.



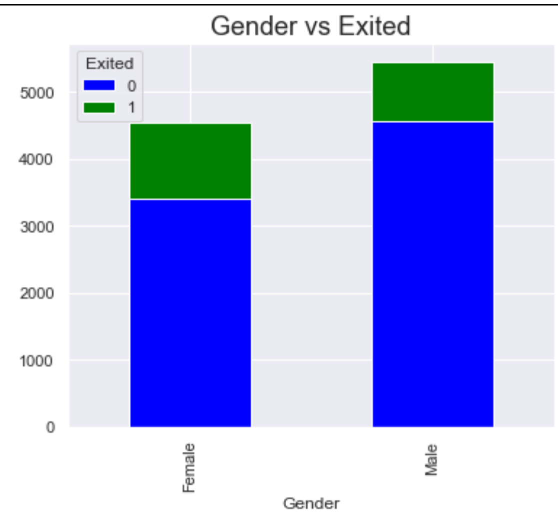
A.



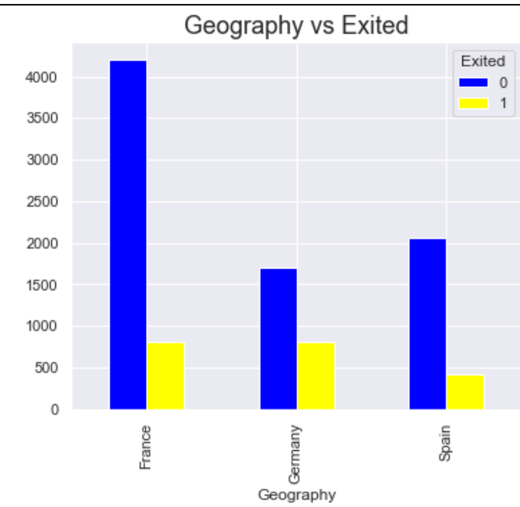
B.

A. The bar graph is used to check for any missing/null values. It is built by importing the missingno library. The graph shows nothing about missing data since the data contains no null values.

C. The above box plot shows the distribution of numeric data and outliers can be observed for each feature if exists. You could See Features like CreditScore, Age and NumOfproducts have Outliers in them which can be treated by trimming.



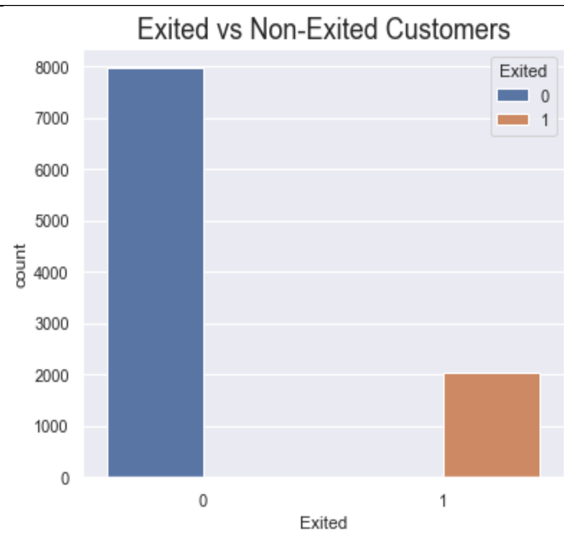
D.



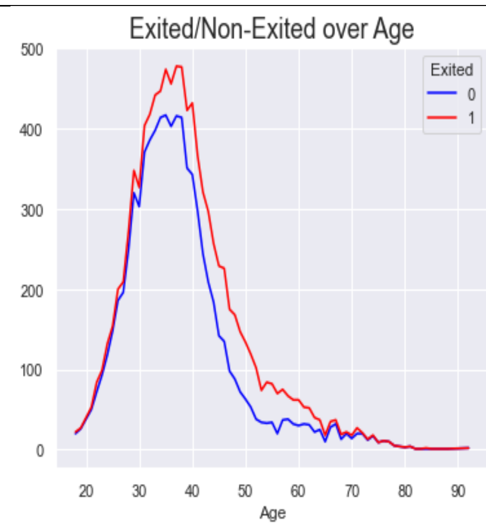
E.

C. Even though male populous, Surprisingly Female customers opted out of the bank.

D. People with accounts in France and Germany have churned the most.



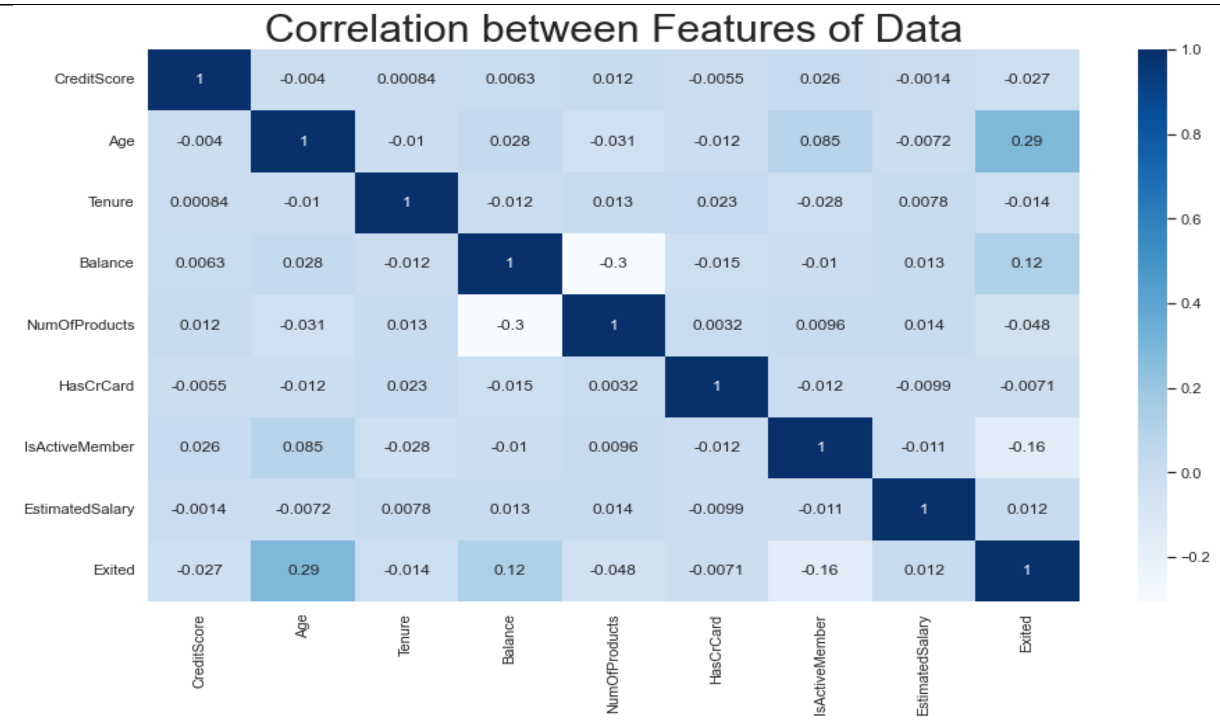
F.



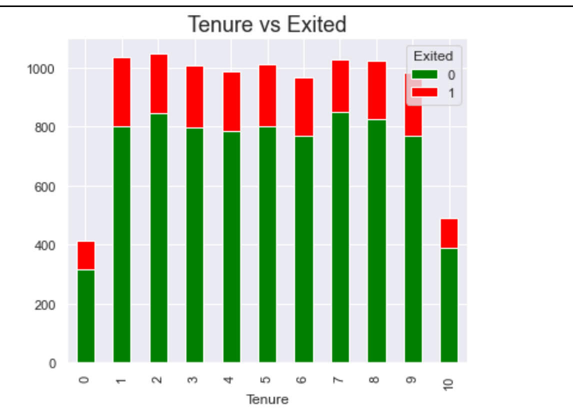
G.

E. The customers who left the bank are more in percentage. This may Causean an imbalance in data.

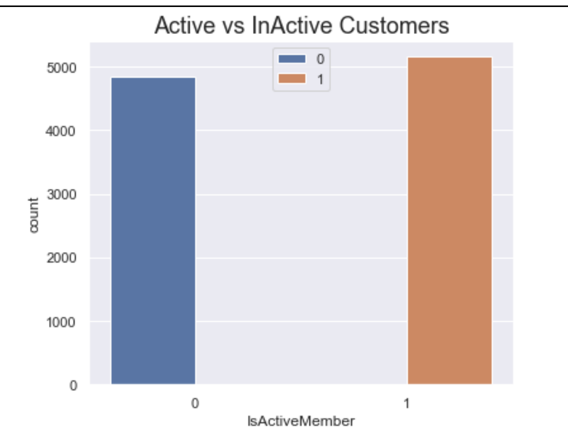
F. The Age group between 30-and 45 has sa een rise in the trend below.



H.



I.



J.

G. The below HeatMap defines How the features are correlated either positive or negative.

H. Customers that have tenure of 1-5 years have observed an increase in churn rate.

I. You could observe more Active Customers in our data plotted through Countplot.

K. 

L. Distribution of Data using Histogram plot.

ENCODING OF THE DATA:

The categorical data are transformed into numerical data using get_dummies from pandas Library which is known as one-hot encoding.

OUTLIER DETECTION & TREATMENT:

Outliers are the data points that differ significantly from the rest of the data due to the variability in the measurement. These are detected by the Inter Quartile Range (IQR) proximity rule. The data points which fall below the lower limit or above the upper limit of the distribution are considered outliers.
The Outliers are treated using the trimming technique. Trimming excludes the outliers values from the analysis.

SPLIT–TRAINING DATA/TEST DATA:

The data is split into two parts training data and testing data. 80% goes to training and the rest come under testing data. Train data is used for training the model and the testing set is used for validation of the trained model.

MODELING:

PIPELINE:

An ML pipeline automates the machine learning workflow by allowing datato be transformed an d correlated into a model, which can then be examined to produce results.
Scaling also has been imported into Pipelines.

Machine Learning Classification Algorithm techniques to be used in this Customer Churn prediction

- Logistic Regression
- Naive Bayes
- K-Nearest Neighbors
- Decision Tree
- Random Forest
- Support Vector Machines

The evaluation metrics were used to evaluate the performance of the model built.
It includes Accuracy, Precision, Recall, F1, Confusion matrix for this classification problem.

LOGISTIC REGRESSION:

Logistic regression is a classification model rather than a regression model. Logistic regression is a simple and more efficient method for binary and linear classification problems. It is a classification model, which is very easy to realize and achieves very good performance with linearly separable classes. It is a process of modeling the probability of a discrete outcome given an input variable. It is a useful analysis method for classification problems. The logistic regression model parameters are roughly the weights for the features. Each weighted feature vector is mapped to a value between 0 and 1 via the S-shaped logistic function. Logistic regression does not require a linear relationship between inputs and outputs due to its nonlinear log transformation.

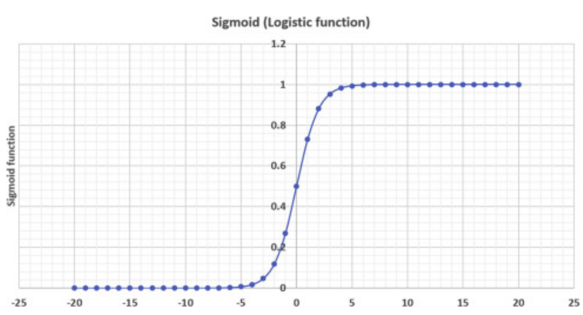


Fig: Logistic Function Curve

$$\text{Logistic function} = \frac{1}{1+e^{-x}}$$

Fig: Logistic Function Formulae

RANDOM FOREST ALGORITHM:

For categorization tasks, random is used. A random forest is a supervised machine learning algorithm that is constructed from decision tree algorithms. This algorithm is applied in various industries such as banking and e-commerce to predict behavior and outcomes. A random forest is a machine learning technique that's used to solve regression and classification problems. It utilizes ensemble learning, which is a technique that combines many classifiers to provide solutions to complex problems.

The 'forest' generated by the random forest algorithm is trained through bagging or bootstrap aggregating. Bagging is an ensemble meta-algorithm that improves the accuracy of machine learning algorithms.

Bagging– It creates a different training subset from sample training data with replacement & the final output is based on majority voting. For example, Random Forest.

SVM:

Support Vector Machine is a supervised studying approach that can be carried out to type and regression issues. It's frequently used to solve categorization challenges. Each information item is plotted as a factor in the n-dimensional area in this method. In which n is the number of features you've got, and the fee of each function is a coordinate value. Then we classify the information with the aid of finding the hyperplane that separates the two training.

DECISION TREE:

Both classification and regression troubles may be solved by the usage of decision trees. Rather than predicting a quantitative solution, it is used to expect a qualitative reaction. We forecast that every observation will fall into the maximum commonplace class. It's a supervised studying algorithm that has a predetermined goal variable. While its miles maximum is commonly used for classification jobs, it may also handle numeric records. To create a prediction, this method divides a statistics pattern into extra homogeneous sets primarily based on the most considerable differentiator in input variables. A thing of a tree is formed with each cut up. As a result, a tree is formed, including choice nodes and leaf nodes (decisions or classifications).

KNN:

The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems. The KNN algorithm assumes that similar things exist in proximity. In other words, similar things are near to each other. We run the KNN algorithm several times with different values of K and choose the K that reduces the number of errors we encounter while maintaining the algorithm's ability to accurately make predictions when it's given data it hasn't seen before.

COMPLIMENT NAÏVE BAYES:

A Naive Bayes classifier is a probabilistic machine learning model that's used for classification tasks. The crux of the classifier is based on the Bayes theorem.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Using the Bayes theorem, we can find the probability of **A** happening, given that **B** has occurred. Here, **B** is the evidence and **A** is the hypothesis. The assumption made here is that the predictors/features are independent. That is the presence of one feature does not affect the other. Hence it is called naive.

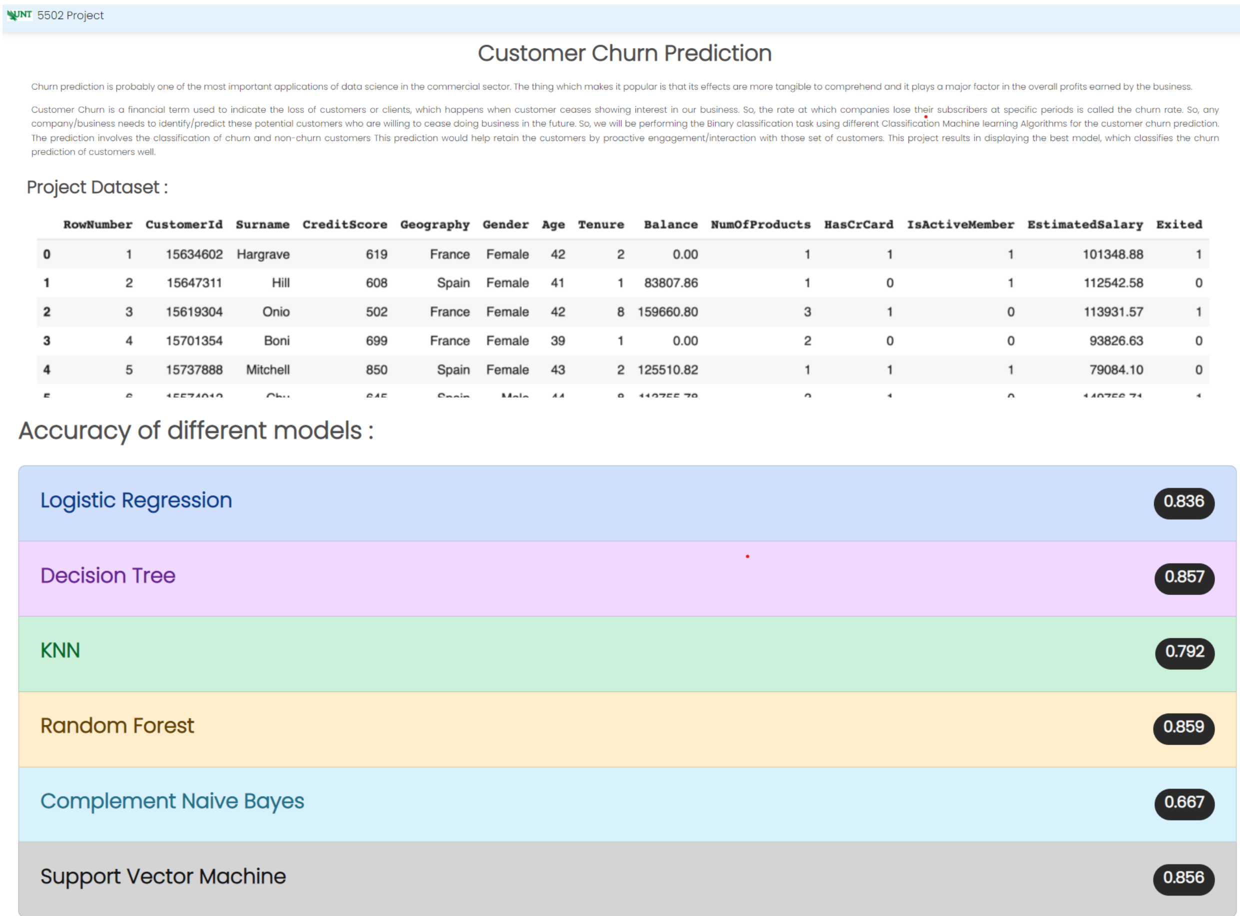
Complement Naive Bayes is somewhat a modification of the standard Multinomial Naive Bayes algorithm. Multinomial Naive Bayes is not able to do very well with unstable data. Imbalanced data sets are instances where the number of instances belonging to a particular class is greater than the number of instances belonging to different classes. This implies the spread of the examples is not even. This kind of data can be difficult to analyze as models can easily overfit this data to benefit a class with a larger instance.

EVALUATION/TUNING:

The evaluation Phase helps us to know Which Algorithm suits best for given data for solving the problem that has been addressed. It's simply finding the best fit. Evaluation is measured using some metrics like accuracy, f1_score, precision-recall, and RoC Curve. Since Our Data is imbalanced as we got to know through EDA, we cannot go with the Accuracy because it might work poorly for the minority class .so we need to look upon F1_score. F1_score is simply the harmonic mean of Precision and recall.
Hyperparameter Tuning involves choosing a set of optimal hyperparameters for the learning algorithm. Hyperparameters Search i.e., Grid search picks out a grid of hyperparameter values and evaluates all of them, and finds the best parameter to the model.

USER INTERFACE:

We have developed web application for our project called customer churn prediction. The web app is developed by using React JS frontend web framework and deployed in Netlify app. Basically, the web app is a simple walkthrough of the project, which explains about what customer churn and a snapshot of dataset is we have used in the project. Data Visualizations of our project which we can draw some insights by seeing those visualizations. Finally, the accuracy values and f1-scores of respective algorithms can be seen in application. The best model can be declared by comparing the accuracy and f1-scores of the algorithms.



UNT 5502 Project

Customer Churn Prediction

Churn prediction is probably one of the most important applications of data science in the commercial sector. The thing which makes it popular is that its effects are more tangible to comprehend and it plays a major factor in the overall profits earned by the business.

Customer Churn is a financial term used to indicate the loss of customers or clients, which happens when customer ceases showing interest in our business. So, the rate at which companies lose their subscribers at specific periods is called the churn rate. So, any company/business needs to identify/predict these potential customers who are willing to cease doing business in the future. So, we will be performing the Binary classification task using different Classification Machine learning Algorithms for the customer churn prediction. The prediction involves the classification of churn and non-churn customers This prediction would help retain the customers by proactive engagement/interaction with those set of customers. This project results in displaying the best model, which classifies the churn prediction of customers well.

Project Dataset :

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

Accuracy of different models :

Logistic Regression	0.836
Decision Tree	0.857
KNN	0.792
Random Forest	0.859
Complement Naive Bayes	0.667
Support Vector Machine	0.856

PROJECT MILESTONES:

We have found some trends in how the features like Age, CreditScore, and, others have impacted the customer churn count.

After performing Hyperparameter tuning, the decision tree model is optimized and has shown better F1 Score than the other models. It led to giving better classification when dealing with the minority class since our data is imbalanced.

PROJECT RESULTS:

We have Applied 6 different Supervised machine learning Classification Algorithms namely KNN classification, Decision Tree, Random Forest, Complement Naive Bayes, Support Vector Machine (SVM), and Logistic Regression. Our concrete goal is to find the best fit model based on the data considered for the prediction of customer churn in the banking institution and its Achieved. Before Tuning the Hyperparameters of the model, Random Forest has performed best for the data since its F1-score is 0.51. After Tuning, we can consider the Decision Tree Model since the F1-score is 0.54 which is higher and better than the other models. So, we can conclude Decision Tree offers best for the customer churn prediction on this data.

DECISION TREE:

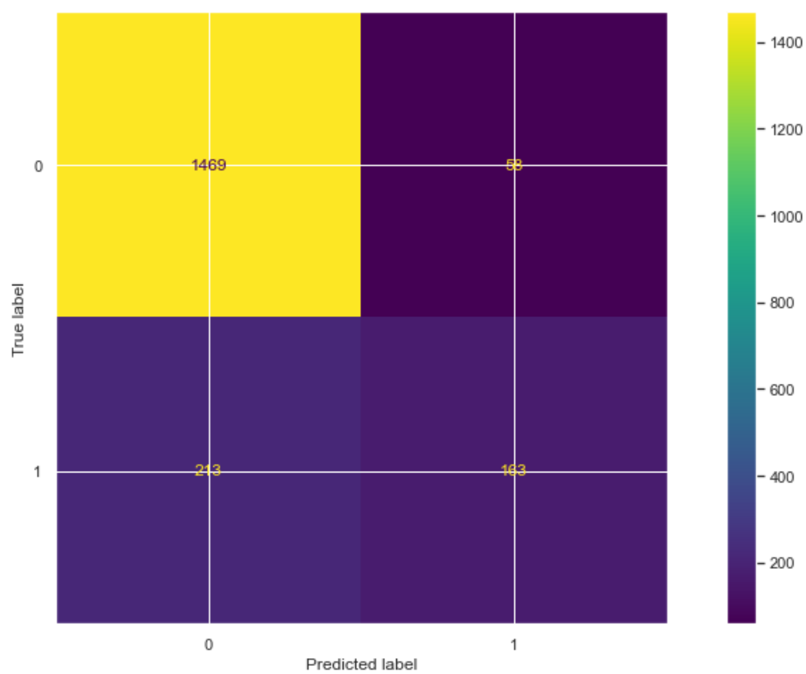


Fig: Confusion Matrix

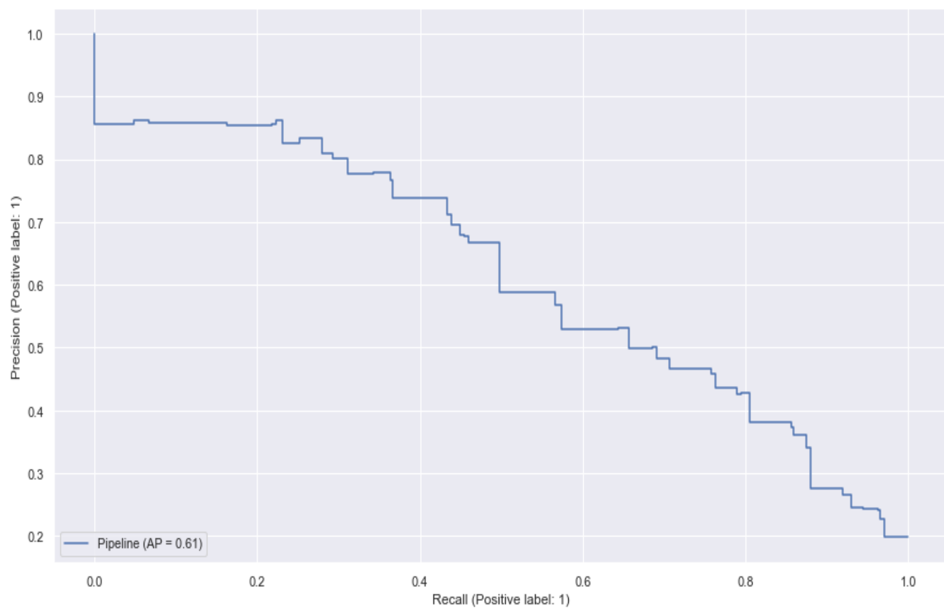


Fig: Precision Recall Curve

Decision Tree :
F1 Score : 0.5460636515912898

	precision	recall	f1-score	support
0	0.87	0.96	0.92	1527
1	0.74	0.43	0.55	376
accuracy			0.86	1903
macro avg	0.81	0.70	0.73	1903
weighted avg	0.85	0.86	0.84	1903

Fig: Accuracy and F1-Scores of Decision Tree Model

For Future scope we can try resampling the data which deals with data imbalance and use tons of data to find more accurate model with much improve and better features by implementing various hyperparameters. This model can be used in various fields such as customer churn for clothing companies, electric appliances, and many other industries to find various reasons to help the companies find reasons to improve their customers satisfaction and needs.

REPOSITORY LINK:

Link: https://github.com/Vam564/5502_Project_2022

USER INTERFACE LINK:

Link: https://customer-churn-prediction.netlify.app/

REFERENCES:

1. https://www.sciencedirect.com/topics/computer-science/logistic-regression#:~:text=Logistic%20regression%20is%20a%20process,%2Fno%2C%20and%20so%20on.

2. https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/

3. https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c

4. https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761

5. https://www.kdnuggets.com/2018/01/managing-machine-learning-workflows-scikit-learn-pipelines-part-3.html

6. https://scikit-learn.org/0.15/modules/pipeline.html

7. https://www.analyticsvidhya.com/blog/2021/06/5-techniques-to-handle-imbalanced-data-for-a-classification-problem/

8. https://app.netlify.com/sites/customer-churn-prediction/settings/domain

9. https://medium.com/dsights/export-charts-from-jupyter-notebook-to-flask-app-with-few-lines-of-code-ad4dca134ba

10. https://towardsdatascience.com/build-a-machine-learning-web-app-in-python-683480acbd37

11. https://www.jeremyjordan.me/preparing-data-for-a-machine-learning-model/

12. https://learnpython.com/blog/python-customer-churn-prediction/

13. https://seaborn.pydata.org/generated/seaborn.countplot.html

APPENDIX

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler,MinMaxScaler
import missingno as mn
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn import svm
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes  import GaussianNB,ComplementNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import
f1_score,confusion_matrix,plot_confusion_matrix,plot_roc_curve,ConfusionMatrixDisplay,plot_precision_recall_cu
rve,classification_report
import warnings
warnings.filterwarnings('ignore')

ban=pd.read_csv('Churn_Modelling.csv')

ban

ban.head(10)

ban. shape

ban.dtypes

ban.isnull().sum()

ban.isnull().sum().sum()

ban.isnull().sum().sum()

ban=ban.drop(['RowNumber','CustomerId','Surname'],axis=1)

ban['CreditScore'].unique()

ban['Age'].unique()
```

```
ban['Gender'].unique()

ban['Tenure'].unique()

ban['Balance'].nunique()

ban['NumOfProducts'].unique()

ban['CreditScore'].unique()
ban['CreditScore'].unique()

ban['EstimatedSalary'].unique()

ban['IsActiveMember'].unique()

ban.describe()

sns.set(rc = {'figure.figsize':(6,5)})
sns.countplot('IsActiveMember',hue='IsActiveMember',data=ban)
plt.legend(loc='upper center')
plt.title('Active vs InActive Customers',size=18)
plt.show()

sns.set(rc = {'figure.figsize':(6,5)})
sns.countplot('IsActiveMember',hue='IsActiveMember',data=ban)
plt.legend(loc='upper center')
plt.title('Active vs InActive Customers',size=18)
plt.show()

print(ban['Gender'].value_counts())
ban['Gender'].value_counts().plot(kind='bar', figsize=(6,5),color=['violet', 'orange'])
plt.xlabel("Count")
plt.ylabel("Gender")
plt.title("Count of Males and Females",size=18)
plt.show()

print(ban['Gender'].value_counts())
ban['Gender'].value_counts().plot(kind='bar', figsize=(6,5),color=['violet', 'orange'])
plt.xlabel("Count")
plt.ylabel("Gender")
plt.title("Count of Males and Females",size=18)
plt.show()

e=ban.groupby(by=['Age','Exited']).Exited.count().unstack()
e.plot(kind='line',stacked=True,color=['blue','red'],figsize=(6,5))
plt.title('Exited/Non-Exited over Age',size=18)
plt.show()

c=ban.groupby(by=['Tenure','Exited']).Exited.count().unstack()
```

```
c.plot(kind='hist',stacked=True,color=['green','red'],figsize=(6,5))
print(c)
plt.title('Tenure vs Exited',size=18)
plt.show()

c=ban.groupby(by=['Gender','Exited']).Exited.count().unstack()
c.plot(kind='bar',stacked=True,color=['blue','green'],figsize=(6,5))
print(c)
plt.title('Gender vs Exited',size=18)
plt.show()

c1=ban.groupby(by=['Geography','Exited']).Exited.count().unstack()
c1.plot(kind='bar',color=['blue','yellow'],figsize=(6,5))
print(c1)
plt.title('Geography vs Exited',size=18)
plt.show()

ban['Exited'].unique()

sns.set(rc = {'figure.figsize':(15,8)})
sns.heatmap(ban.corr(), cmap='Blues',annot=True)
plt.title('Correlation between Features of Data',size=30)
plt.show()

ban.dtypes

ban.plot(kind='box', subplots=True, layout=(4,4),sharex=False, sharey=False, color = 'blue', figsize=(12,8),
patch_artist=True )
plt.show()

for i ,j in enumerate(['CreditScore','Age','NumOfProducts']):
  print(j)

Q1=[]
Q3=[]
iqr=[]
upp=[]
low=[]

for i,j in enumerate(['CreditScore','Age','NumOfProducts']):
    Q1.append(ban[j].quantile(0.25))
    Q3.append(ban[j].quantile(0.75))
    iqr.append(np.array(Q3[i])-np.array(Q1[i]))
    upp.append(Q3[i] + 1.5 * iqr[i])
    low.append(Q1[i] - 1.5 * iqr[i])

ban[ban['CreditScore']>upp[0]]

ban[ban['CreditScore']<low[0]]
```

```
ban[ban['Age']>upp[1]]

ban[ban['Age']<low[1]]

ban[ban['NumOfProducts']<low[2]]

ban[ban['NumOfProducts']>upp[2]]

ban_no=ban[ban['CreditScore']>low[0]]
ban_no=ban_no[ban_no['Age']<upp[1]]
ban_no=ban_no[ban_no['NumOfProducts']<upp[2]]

ban_no

ban_cat=ban_no.select_dtypes(include='object')

ban_cat

ban_num=ban_no.select_dtypes(exclude='object')

ban_num

ban_cat=pd.get_dummies(ban_cat,prefix=None)

ban_new=pd.concat([ban_num,ban_cat],axis=1)

ban_new

x=ban_new.drop('Exited',axis=1)
y=ban_new['Exited']

from numpy.core.fromnumeric import size
axis=x.hist(figsize=(15,19))
colors = ["#e74c3c", "#2ecc71", "#3498db"]
plt.suptitle('Distrubution of Data using Histogram plot',size=20)
for i, ax in enumerate(axis.reshape(-1)):
    # Create a counter to ensure that if there are more than three bars containing a value.
    ## We don't try to access elements in colors that are out of range.
    w = 0
    ax.grid(False)
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)
    for rect in ax.patches:
        # If there's a value in the rect and we have defined a color
        if rect.get_height() > 0 and w < len(colors):
            # Set up the color
            rect.set_color(colors[w])
            # Increment the counter
```

```
        w += 1

x.plot(kind='density', subplots=True, layout=(5,5), sharex=False, legend=True, fontsize=1, figsize=(20,25))
plt.suptitle('Density plot distrubution of data',size=20)
plt.show()

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)


pipe_lreg = Pipeline([('scl', StandardScaler()),('clf', LogisticRegression(random_state=42))])

pipe_rfor=Pipeline([('scl',StandardScaler()),('clf',RandomForestClassifier(random_state=42))])

pipe_dt=Pipeline([('scl',StandardScaler()),('clf',DecisionTreeClassifier(random_state=42))])

pipe_cnb=Pipeline([('scl',MinMaxScaler()),('clf',ComplementNB())])

pipe_knn=Pipeline([('scl',StandardScaler()),('clf',KNeighborsClassifier())])

pipe_svm=Pipeline([('scl',StandardScaler()),('clf',svm.SVC(random_state=42))])

a=[pipe_lreg,pipe_rfor,pipe_dt,pipe_cnb,pipe_knn,pipe_svm]


dic={0:'Logistic Regression',1:'Random Forest',2:'Decision Tree',3:'Complement Naive Bayes'

for i,j in enumerate(a):
  j.fit(x_train,y_train)
  pred=j.predict(x_test)
  print(dic[i],':')
  plot_confusion_matrix( j, x_test, y_test)
  plt.show()

for i,j in enumerate(a):
  j.fit(x_train,y_train)
  pred=j.predict(x_test)
  print(dic[i],':')
  print(accuracy_score(y_test,pred))
  plt.show()

for i,j in enumerate(a):
  j.fit(x_train,y_train)
  pred=j.predict(x_test)
  print(dic[i],':')
  plot_precision_recall_curve(j,x_test,y_test)
  plt.show()

for i,j in enumerate(a):
  j.fit(x_train,y_train)
```

```
  pred=j.predict(x_test)
  print(dic[i],':')
  print(f1_score(y_test,pred))
  print(classification_report(y_test,pred))
  plt.show()

param_range = [1,2,4,7,9]
param_range_fl = [1.0, 0.5, 0.1,0.0001, 0.001]

params_grid_lr = [{'clf__penalty': ['l1', 'l2'],
                   'clf__C': param_range_fl,
                   'clf__solver': ['liblinear']}]

params_grid_rf = [{'clf__criterion': ['gini', 'entropy'],
                   'clf__min_samples_leaf': param_range,
                   'clf__max_depth': param_range,
                   'clf__min_samples_split': param_range[1:]}]

params_grid_svm = [{'clf__kernel': ['linear', 'rbf'],
                   'clf__C': param_range}]

params_grid_dt=[{'clf__criterion':['gini', 'entropy']
    ,'clf__max_depth' : param_range }]

params_grid_cnb ={'clf__alpha':[0.1,0.3,0.8,1]}
params_grid_knn=[{'clf__n_neighbors':param_range,'clf__weights':['uniform','distance']


jobs = -1

gs_lr = GridSearchCV(estimator=pipe_lreg,
                        param_grid=params_grid_lr,
                        scoring='f1',
                        cv=10)


gs_rf = GridSearchCV(estimator=pipe_rfor,
                        param_grid=params_grid_rf,
                        scoring='f1',
                        cv=10,
                        n_jobs=jobs)

gs_svm = GridSearchCV(estimator=pipe_svm,
                        param_grid=params_grid_svm,
                        scoring='f1',
                        cv=10,
                        n_jobs=jobs)

gs_knn = GridSearchCV(estimator=pipe_knn,
```

```
                                         param_grid=params_grid_knn,
                                         scoring='f1',
                                         cv=10,
                                         n_jobs=jobs)
gs_dt = GridSearchCV(estimator=pipe_dt,
                                         param_grid=params_grid_dt,
                                         scoring='f1',
                                         cv=10,
                                         n_jobs=jobs)
gs_cnb = GridSearchCV(estimator=pipe_cnb,
                                         param_grid=params_grid_cnb,
                                         scoring='f1',
                                         cv=10,
                                         n_jobs=jobs)


grids = [gs_lr,gs_rf, gs_dt,gs_cnb,gs_knn, gs_svm]
for i,gs in enumerate(grids):
   print('Estimator',dic[i])
   gs.fit(x_train,y_train)
   print('Best params: %s' % gs.best_params_)
   ypred=gs.predict(x_test)

pipe_lreg1 = Pipeline([('scl', StandardScaler()),('clf', LogisticRegression(C=1.0, penalty='l2',
solver='liblinear',random_state=42))])

pipe_rfor1=Pipeline([('scl',StandardScaler()),('clf',RandomForestClassifier(criterion= 'gini',max_depth=9,
min_samples_leaf= 1,min_samples_split= 2,random_state=42))])

pipe_dt1=Pipeline([('scl',StandardScaler()),('clf',DecisionTreeClassifier(criterion='gini', max_depth=
7,random_state=42))])

pipe_cnb1=Pipeline([('scl',MinMaxScaler()),('clf',ComplementNB(alpha= 0.1))])

pipe_knn1=Pipeline([('scl',StandardScaler()),('clf',KNeighborsClassifier(metric='euclidean',n_neighbors= 1,
weights='uniform'))])

pipe_svm1=Pipeline([('scl',StandardScaler()),('clf',svm.SVC(C= 9, kernel='rbf',random_state=42))])

b=[pipe_lreg1,pipe_rfor1,pipe_dt1,pipe_cnb1,pipe_knn1,pipe_svm1]

for i,j in enumerate(b):
  j.fit(x_train,y_train)
  pred1=j.predict(x_test)
  print(dic[i],':')
  print(accuracy_score(y_test,pred1))

for i,j in enumerate(b):
  j.fit(x_train,y_train)
```

```
 pred1=j.predict(x_test)
 print(dic[i],':')
 print(plot_confusion_matrix(j,x_test,y_test))
 plt.show()


for i,j in enumerate(b):
 j.fit(x_train,y_train)
 pred1=j.predict(x_test)
 print(dic[i],':')
 print(accuracy_score(y_test,pred1))
 print(f1_score(y_test,pred1))
 print(classification_report(y_test,pred1))
```