PCET's

PIMPRI CHINCHWAD COLLEGE OF ENGINEERING

TITLE- BANK CUSTOMERS EXITNG PREDICTIVE MODEL

VARUN GADDE - BECOB203

GUIDE: PROF. PRIYA SURANA

VEDANT UPGANLAWAR - BECOB270

1. DESCRIPTION

1.A PROBLEM STATEMENT

Consider a labeled dataset belonging to an application domain. Apply suitable data preprocessing steps such as handling of null values, data reduction, discretization. For prediction of class labels of given data instances, build classifier models using different techniques (minimum 3), analyze the confusion matrix and compare these models. Also apply cross validation while preparing the training and testing datasets.

1.B DATASET DESCRIPTION

A bank is investigating a very high rate of customer leaving the bank. Here is a 10,000 records dataset to investigate and predict which of the customers are more likely to leave the bank soon.

We will be guiding you with the implementation part simultaneosuly. Current project can be run on any Python,Jupyter notebook,Google Colab etc.

The project requires the following imports:

1. Numpy - for Linear Algebra
2. Pandas - for Data Preprocessing and CSV I/O
3. Matplotlib - Data Visualization
4. sklearn.modelselection - for Modelling
5. sklearn.linear_model - for Logistic Regression Classifier
6. sklearn.svm - for Support Vector Classifier
7. sklearn.naive_bayes - for Gaussian Naive Bayes Classifier
8. sklearn.neighbors - for KNeighbors Classifier
9. sklearn.ensemble - for Random Forest Classifier
10. sklearn.metrics - for Accuracy Score, Confusion Matrix and Classification Report

As and when the need aries the imports are executed in the project.

Title of the dataset that is to be anaysed throughout the project is 'Churn_Modelling.csv' which is manually uploaded to the execution environment.We loaded the dataset into a variable 'data'.Following is the description of all the attributes of the dataframe -

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   RowNumber        10000 non-null  int64
 1   CustomerId       10000 non-null  int64
 2   Surname          10000 non-null  object
 3   CreditScore      10000 non-null  int64
 4   Geography        10000 non-null  object
 5   Gender           10000 non-null  object
 6   Age              10000 non-null  int64
 7   Tenure           10000 non-null  int64
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64
 10  HasCrCard        10000 non-null  int64
 11  IsActiveMember   10000 non-null  int64
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64
dtypes: float64(2), int64(9), object(3)
```

1.C ATTRIBUTES ANALYSIS

On analysis of the attributes of dataset, attributes are categorized as follows:

1. **Categorical Variables**
   - Geography
   - Gender
   - HasCrCard
   - IsActiveMember
   - Exited
   - NumOfProducts
2. **Numerical Variables**
   - CreditScore
   - Age
   - Tenure
   - Balance
   - EstimatedSalary

From the following features,**'Exited'** is a categorical feature which determines whether the customer left the bank or not.

2. PREPROCESSING

2.A MISSING VALUES

We can check missing value also by isnull() method as below:

```
data.isnull().any()
```

```
RowNumber          False
CustomerId         False
Surname            False
CreditScore        False
Geography          False
Gender             False
Age                False
Tenure             False
Balance            False
NumOfProducts      False
HasCrCard          False
IsActiveMember     False
EstimatedSalary    False
Exited             False
dtype: bool
```

From above, it can be concluded that there are no missing value in any of the columns. There are 10000 items and each column has 10000 non-null items.

2.B DROPPING UNNECESSARY COLUMNS

```
data.drop(["RowNumber","CustomerId","Surname"], axis=1, inplace=True)
data.head()
```

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

Here we can clearly observe that columns**'Rownumber', 'CustomerId' and 'Surname'** does not provide any meaningful information. Hence we are dropping the above mentioned columns.

2.C DISCRETIZATION

Now we will convert the following attributes into categorical values.

1. Geography

Geography_France	Geography_Germany	Geography_Spain
1	0	0
0	0	1
1	0	0
1	0	0
0	0	1

2. Gender

Gender_Female	Gender_Male
1	0
1	0
1	0
1	0
1	0

3. Tenure

Tenure_0	Tenure_1	Tenure_2	Tenure_3	Tenure_4	Tenure_5	Tenure_6	Tenure_7	Tenure_8	Tenure_9	Tenure_10
0	0	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0
0	1	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0

4. NumOfProducts

NumOfProducts_1	NumOfProducts_2	NumOfProducts_3	NumOfProducts_4
1	0	0	0
1	0	0	0
0	0	1	0
0	1	0	0
1	0	0	0

3. CLASSIFICATION

3.A MODELLING

Modelling includes creation of train, test data by splitting the dataset. The steps followed below are:

1. Create X_train by dropping the dependent attribute **"Exited"** from the dataframe df.
2. Create y_train by considering the dependent attribute **"Exited"** from the dataframe df.
3. Normalizing necessary attributes (CreditScore, Age, Balance, EstimatedSalary)
4. Splitting the X_train and y_train as X_Train,y_Train,X_Test,y_test. The split ratio varies according to the discretion of the user. Here we have used the splitting ratio as 0.33.

Here we have executed some instructions for better understanding of the splitted data shape.

```
#creating X_train and y_train
X_train = df.drop("Exited", axis = 1)
y_train = df["Exited"]

#Features (CreditScore, Age, Balance, EstimatedSalary) to be normalized:
for each in ["CreditScore", "Age", "Balance", "EstimatedSalary"]:
    normalize_feature(each, X_train)

#X_train and Y_train data shape summary
print("Length of X_train: ",len(X_train))
print("Shape of X_train: ", X_train.shape)
print("Length of Y_tain: ", len(y_train))
print("Shape of Y_train: ", y_train.shape)
```

```
Length of X_train:  10000
Shape of X_train:  (10000, 26)
Length of Y_tain:  10000
Shape of Y_train:  (10000,)
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X_train,
    y_train,
    test_size = 0.33,
    random_state = 42
)

print("Length of X_train: ",len(X_train))
print("Length of X_test: ",len(X_test))
print("Length of y_train: ",len(y_train))
print("Length of y_test: ",len(y_test))
```

```
Length of X_train:  6700
Length of X_test:  3300
Length of y_train:  6700
Length of y_test:  3300
```

3.B LOGISTIC REGRESSION ☐

3.C SVM ☐

3.D NAIVE BAYES ☐

3.E KNN ☐

3.F RANDOM FOREST ☑

CLASSIFY

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

rf = RandomForestClassifier(n_estimators = 24, random_state = 42)
rf.fit(X_train, y_train)
y_pred=rf.predict(X_test)

print("RF accuracy with test data :", rf.score(X_test, y_test)*100)
cm=confusion_matrix(y_test,y_pred)
print()
print(cm)
print()
print(classification_report(y_test, y_pred))
```

```
RF accuracy with test data : 86.51515151515152

[[2554  103]
 [ 342  301]]

              precision    recall  f1-score   support

           0       0.88      0.96      0.92      2657
           1       0.75      0.47      0.57       643

    accuracy                           0.87      3300
   macro avg       0.81      0.71      0.75      3300
weighted avg       0.86      0.87      0.85      3300
```

4. METRICS

4.A K-FOLD CROSS VALIDATION

In this case of cross validation, we will divide the complete dataset into k parts.Each section will act as the test set and rest of the k-1 sections will be used for training.Hence we get k different accuracy scores.For this dataset, we have **k=10.**Finally from these k scores, we obtain a mean accuracy which helps us compare the classifiers with better precision.Below is implementation of k-fold cross validation on Random Forest Classifier -

```
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator =rf, X = X_train, y = y_train, cv = 10)
print("accuracies :",accuracies)
print("mean accuracy :", accuracies.mean())
```

```
accuracies : [0.84029851 0.85970149 0.85074627 0.86268657 0.86567164 0.85970149
 0.84179104 0.85223881 0.84477612 0.85373134]
mean accuracy : 0.8531343283582089
```

Further classifier's summary is given in the below table (Following table is arranged in descending order of their respective mean accuracies) -

Classifier	Accuracy (Test Data)	K-Fold Cross Validation Accuracy	Confusion Matrix	Classification Report
Random Forest (RF)	86.51	85.31	[[2554 103] [342 301]]	precision recall f1-score support 0 0.88 0.96 0.92 2657 1 0.75 0.47 0.57 643
Logistic Regression (LR)	85.0	83.61	[[2552 105] [399 244]]	precision recall f1-score support 0 0.86 0.96 0.91 2657 1 0.70 0.38 0.49 643
Support Vector (SVM)	84.66	83.46	[[2602 55] [451 192]]	precision recall f1-score support 0 0.85 0.98 0.91 2657 1 0.78 0.30 0.43 643
Naive Bayes (NB)	82.78	81.56	[[2638 19] [549 94]]	precision recall f1-score support 0 0.83 0.99 0.90 2657 1 0.83 0.15 0.25 643
K Nearest Neighbour (KNN)	82.0	80.20	[[2559 98] [502 141]]	precision recall f1-score support 0 0.84 0.96 0.90 2657 1 0.59 0.22 0.32 643