MAY 7, 2022

Database System Final Project Report

Computer Engineering Department

CUSTOMER CHURN PREDICTION

AMNA JAMSHAID, FALAK AMIN

Ma’am Darakhshan

2019 CE 09, 2019 CE 25

Project Title:

<u>Project overview</u>

Introduction of Project:

Churn Prediction is the process of forecasting which customers are most likely to cancel a subscription, or 'leave a firm,' based on how they use the service. From a business standpoint, this knowledge is crucial since obtaining new clients is generally more difficult and expensive than keeping old ones. Hence, the insights gained from Churn Prediction helps them to focus more on the customers that are at a high risk of leaving.

A classification machine learning problem for predicting customers churn from the company based on customers who left within the last month labeled by 'yes' or 'no'.

Project Description:

Many variables impact a customer's decision to churn. It may be that a new competitor has reached the industry and is providing better rates, or that the service they are receiving is not up to standard, and so on. As a result, there is no precise explanation as to why a client wants to churn, because there are several impacting variables. The goal of our project is to look for patterns in the data and observe what facts are produced during data analysis. Customer churn can be caused by a variety of circumstances, but avoiding it is typically simple. It is dependent on the company's ability to make clients feel special and give a personalized services.

Customer churn's most common reasons:

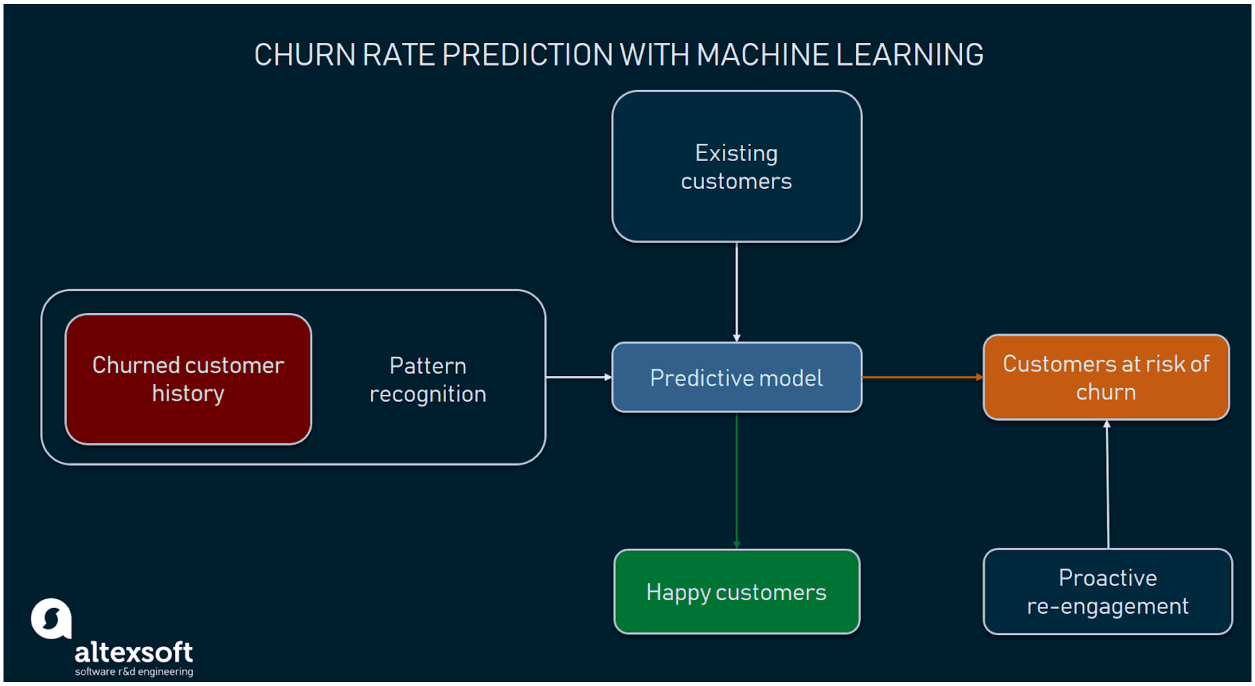
Churn can be caused by a variety of factors, including

- Customer service issues
- Failure to fulfil quality and standards set by the market
- Insufficient worth
- Customer-fit issues
- Other options have been discovered by customers.

Purpose of Project:

For various reasons, it is a useful tool for businesses:

- It assists in identifying potential threats.
- It helps companies to take precautionary measures.
- It aids in a better understanding of clients, making it easier to sustain productive client connections.
- It aids in the making of better business judgments.



<u>**Analysis of your dataset**</u>

The dataset used in this project is obtained from [Kaggle - Telco Customer Churn]

Out[2]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	Tec
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	
...	...	...	...	...	...	...	...	...	...	...	...	...	
7038	6840-RESVB	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes	...	Yes	
7039	2234-	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	No		Yes	

The data set includes information about:
- Customers who left within the last month – the column is called Churn
- Services that each customer has signed up for – phone, multiple lines, internet, online security, online backup, device protection, tech support, and streaming TV and movies.
- Customer account information – how long they’ve been a customer, contract, payment method, paperless billing, monthly charges, and total charges
- Demographic info about customers – gender, age range, and if they have partners and dependents

Methodology

At first 20% of the data were splitted for final testing; stratified by the 'Churn' (target) column.

Separate 20% of the data for testing

For model evaluation on totally unseen data

```
]: train_df, test_df = train_test_split(df, test_size=0.2, random_state=42, stratify=df['Churn'])
   train_df.reset_index(drop=True, inplace=True)
   test_df.reset_index(drop=True, inplace=True)
   train_df
```

]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	Tec
0	4950-BDEUX	Male	0	No	No	35	No	No phone service	DSL	No	...	Yes	
1	7993-NQLJE	Male	0	Yes	Yes	15	Yes	No	Fiber optic	Yes	...	No	
2	7321-ZNSLA	Male	0	Yes	Yes	13	No	No phone service	DSL	Yes	...	No	
3	4922-CVPDX	Female	0	Yes	No	26	Yes	No	DSL	No	...	Yes	
4	2903-YYTBW	Male	0	Yes	Yes	1	Yes	No	DSL	No	...	No	
...	...	...	...	...	...	...	...	...	...	...	...	...	
5629	6308-CQRBU	Female	0	Yes	No	71	Yes	Yes	Fiber optic	No	...	Yes	
5630	2842-	Male	0	No	No	2	Yes	No	DSL	No		No	

```
]: display(train_df.info())
```

Data cleaning:

- Convert 'TotalCharges' column which is of object type to float type using pd.to_numeric() with errors parameter set to 'coerce' to parse invalid data to NaN.
- Eight missing values were found in the 'TotalCharges' column and were imputed by the mean() value.
- Data has no duplicates.

Data Cleaning

1. **Converting 'TotalCharges' column to numeric**

```
[5]: train_df['TotalCharges'] = pd.to_numeric(train_df['TotalCharges'], errors='coerce')
```

8 Null values in 'TotalCharges' column

- Fill with Mean value

```
[7]: train_df['TotalCharges'].fillna((train_df['TotalCharges'].mean()), inplace=True)
```

3. **Check for duplicates**

```
[8]: train_df.duplicated().sum()
```

```
[8]: 0
```

No duplicates were found

<u>Exploring your questions, with appropriate visualizations</u>

Exploratory data analysis

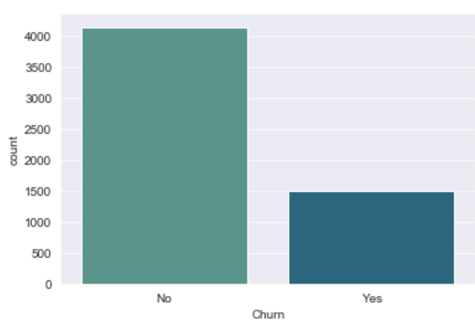
1. Count plot shows the distribution of the churn rate in the data which showed an imbalance in the data.

Exploratory Data Analysis

Distribution of target column:

```
9]: print(train_df['Churn'].value_counts())
_ = sns.countplot(x='Churn', data=train_df, palette='crest')
```

```
No     4139
Yes    1495
Name: Churn, dtype: int64
```

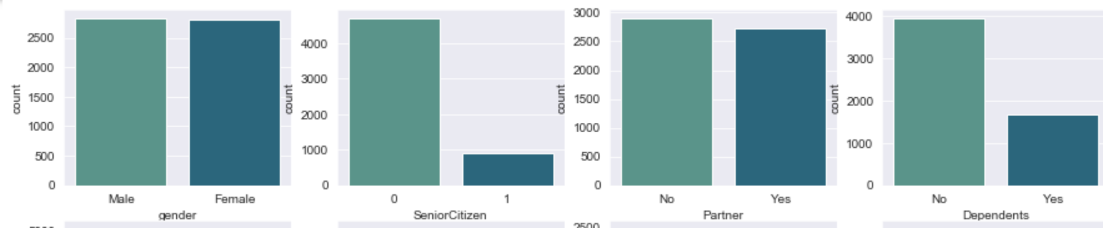


2. Categorical features count plot insights:

- Data is evenly distributed between the two genders; males and females, which might be useful in further analysis.
- No information added by 'No Internet Service' or 'No Phone Service' and 'No' categories.
- Replacing 'No Internet Service' and 'No Phone Service' entries with 'No'.

Categorical features count plot

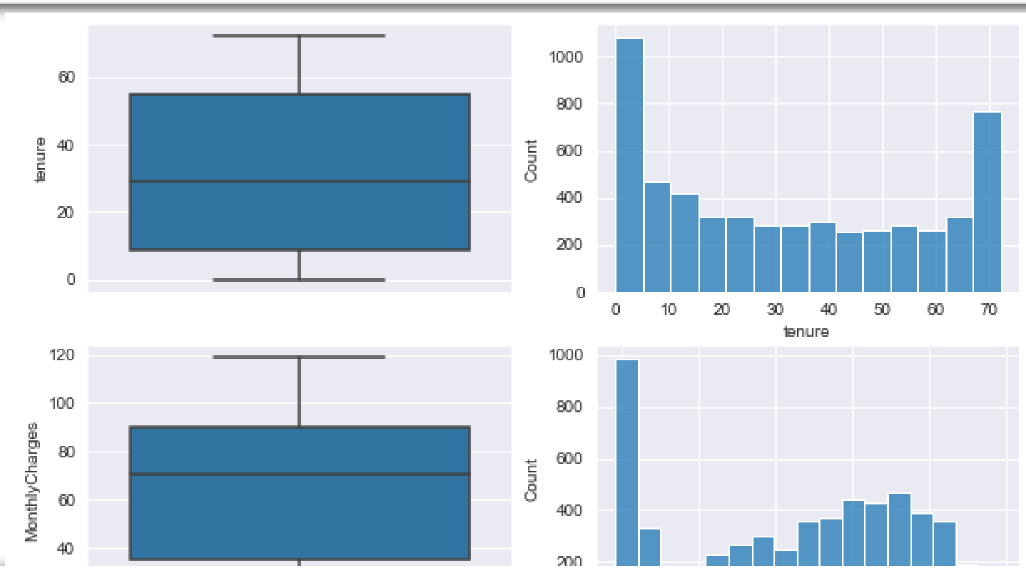
```
[10]: cat_cols = ['gender','SeniorCitizen','Partner','Dependents','PhoneService','MultipleLines','InternetService',
                 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies',
                 'Contract', 'PaperlessBilling']

plt.figure(figsize=(15, 15))
for n, variable in enumerate(cat_cols):
    ax = plt.subplot(5, 4, n + 1)
    g=sns.countplot(data=train_df, x=train_df[variable], ax=ax, palette='crest')
plt.show()
plt.figure(figsize=(11,3))
_ = sns.countplot(x= 'PaymentMethod', hue='Churn', data=train_df, palette='crest')
plt.show()
plt.tight_layout()
```



2. Histogram and box plot of continous features implies that:

- No outliers exists.
- 'TotalCharges' feature is right skewed.

```
_ = sns.histplot(x='TotalCharges', data=train_df)
```



4. Scatter plot of 'MonthlyCharges' vs. 'TotalCharges' shows a positive correlation between both and also it affects the Churn rate positively.

Scatter plot of Monthly Charges versus Total Charges

```
plt.figure(figsize=(10,4))
sns.scatterplot(data=train_df, x='MonthlyCharges', y='TotalCharges', hue='Churn')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1eb82f31f88>
```

Feature encoding

Several encoding techniques were tested on each categorical feature separately and One-Hot encoding all the categorical features gave the best results.

Encoding categorical features

- One-Hot encoding all categorical features
- Encode by mapping target feature

```
# One-hot encoding
cat_cols = ['gender', 'InternetService','PaymentMethod','Partner', 'Dependents', 'PhoneService', 'PaperlessBilling', 'MultipleLi
            'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies']

train_df = pd.concat([train_df, pd.get_dummies(train_df[cat_cols])], axis='columns')
train_df = train_df.drop(columns=cat_cols)

# mapping
train_df['Churn'] = np.where(train_df['Churn'] == 'Yes', 1, 0)  # yes=1, No=0
```

Feature engineering

Binning 'tenure' feature into 6 ranges:

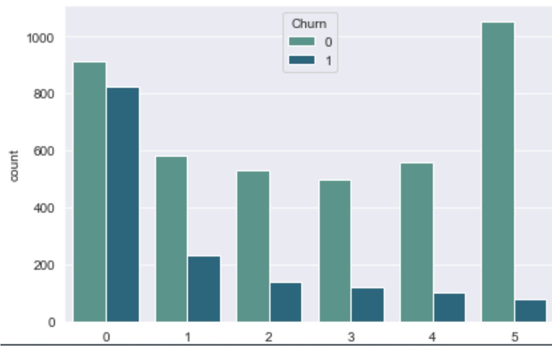
- 0-12 months --> '0-1 years'
- 12-24 months --> '1-2 years'
- 24-36 months --> '2-3 years'
- 36-48 months --> '3-4 years'
- 48-60 months --> '4-5 years'
- More than 60 months --> 'more than 5 years'

Feature engineering

1. **Binning 'tenure' feature into 6 ranges**

```
condition = [((train_df.tenure >= 0)&(train_df.tenure <= 12)), ((train_df.tenure > 12)&(train_df.tenure <= 24)),
             ((train_df.tenure > 24)&(train_df.tenure <= 36)),((train_df.tenure > 36)&(train_df.tenure <= 48)),
             ((train_df.tenure > 48)&(train_df.tenure <= 60)), (train_df.tenure > 60)]

#choice = ['0-1year','1-2years', '2-3years', '3-4years','4-5years','more than 5 years']
choice = [0,1, 2, 3, 4, 5]
train_df['tenure_range'] = np.select(condition, choice)
```

```
_ = sns.countplot(x= 'tenure_range', hue='Churn', data=train_df, palette='crest', order=choice)
plt.tight_layout()
```



Feature scaling:

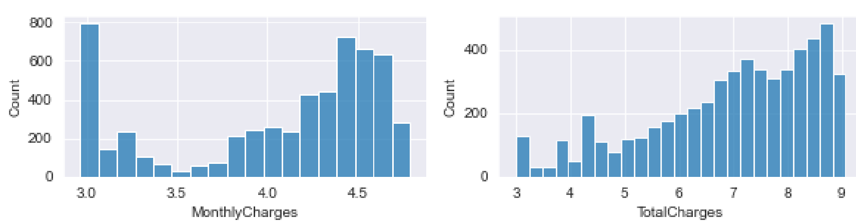
log transformation is very powerful in feature scaling specially with skewed data, hence, np.log1p() is applied on 'MonthlyCharges' and 'TotalCharges' features and with trials it proved giving the best results over MinMaxScaler() and StandaredScaler().

Feature Scaling

- Log transform

```
[17]: train_df['MonthlyCharges']=np.log1p(train_df['MonthlyCharges'])
      train_df['TotalCharges']=np.log1p(train_df['TotalCharges'])
```

```
[18]: plt.figure(figsize=(15,2))
      plt.subplot(1, 3, 2)
      _ = sns.histplot(x='MonthlyCharges', data=train_df)

      plt.subplot(1, 3, 3)
      _ = sns.histplot(x='TotalCharges', data=train_df)
```



Data imbalance:
Data imbalance affects machine learning models by tending only to predict the majority class and ignoring the minority class, hence, having major misclassification of the minority class in comparison with the majority class. Hence, we use techniques to balance class distribution in the data.

Even that our data here doesn't have severe class imbalance, but handling it shows results improvement.
Using **SMOTE** (Synthetic Minority Oversampling Technique) library in python that randomly increasing the minority class which is 'yes' in our case.

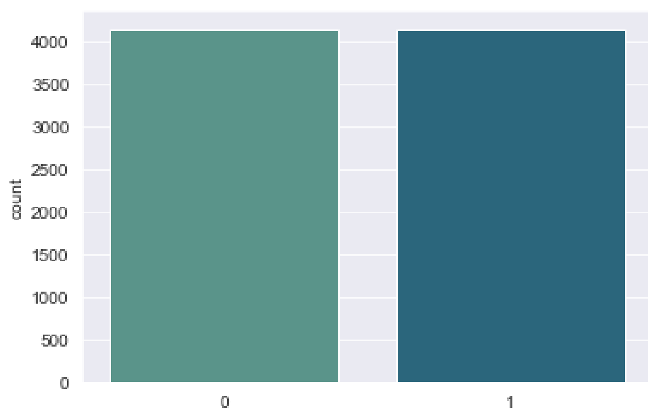
SMOTE synthetically creates new records of the minority class by randomly selecting one or more of the k-nearest neighbors for each example in the minority class. Here, k= 5 neighbors is used.

Handling data imbalance

In [19]:
```
X_train = train_df.drop(columns=['customerID','Churn'])
y_train = train_df['Churn']
```

In [20]:
```
sm = SMOTE(random_state = 0, sampling_strategy = 'minority', k_neighbors= 5)
X_train, y_train = sm.fit_resample(X_train, y_train.ravel())
```

In [21]:
```
_ = sns.countplot(x=y_train, palette='crest')
```



Predictions based on your explorations

To handle any expected missing values in the test set, a condition is added inside the function to map the mean value of its column in the train set.

```
In [22]: def test_prep (test_df):
             ### Data cleaning
             #Converting 'TotalCharges' column to numeric
             test_df['TotalCharges'] = pd.to_numeric(test_df['TotalCharges'], errors='coerce')

             #Replacing 'No internet service' and 'No phone service' with 'No'
             test_df.replace(['No internet service','No phone service'], 'No', inplace=True)

             # if there is null values in the continous features --> fill with the mean of columns in training set (mapping)
             for col in test_df.columns:
                 if test_df[col].isna().sum() > 0:
                     test_df[col] = test_df[col].fillna(train_df[col].map(np.mean))

             ### Categorical features encoding
             test_df = pd.concat([test_df, pd.get_dummies(test_df[cat_cols])], axis='columns')
             test_df = test_df.drop(columns=cat_cols)

             test_df['Churn'] = np.where(test_df['Churn'] == 'Yes', 1, 0)

             ### Feature engineering
             #Binning 'tenure' feature into 6 ranges
             condition = [((test_df.tenure >= 0)&(test_df.tenure <= 12)), ((test_df.tenure > 12)&(test_df.tenure <= 24)),
                          ((test_df.tenure > 24)&(test_df.tenure <= 36)),((test_df.tenure > 36)&(test_df.tenure <= 48)),
                          ((test_df.tenure > 48)&(test_df.tenure <= 60)), (test_df.tenure > 60)]
             #choice = ['0-1year','1-2years', '2-3years', '3-4years','4-5years','more than 5 years']
             choice = [0,1, 2, 3, 4, 5]
             test_df['tenure_range'] = np.select(condition, choice)

             ### Feature Scaling
```

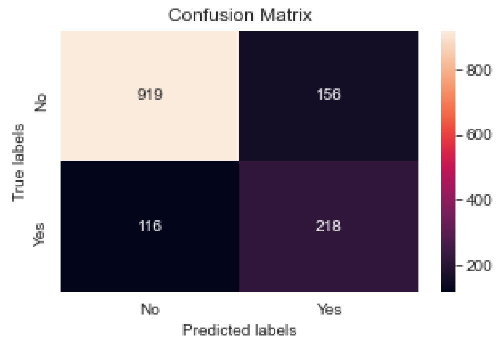
Models training:
Four different models were applied on the data and all results are reported with confusion matrix and classification report showing the precision, recall, and f1-score metrics.

1. **Logistic regression:**

The method of modelling the probability of a discrete result given an input variable is known as logistic regression. The most frequent logistic regression models have a binary result, which might be true or false, yes or no, and so forth. In statistical term, by estimating probabilities using a logistic regression equation, it is used in statistical software to comprehend the relationship between the dependent variable and one or more independent variables.

Best parameters after several trials: C=200 (very large c value trying to fit the data as possible without overfitting), max_iter=1000

```
Out[27]: [Text(0, 0.5, 'No'), Text(0, 1.5, 'Yes')]
```



- **Classification report**

```
In [28]: print(classification_report(y_test, log_pred, target_names=['No', 'Yes']))

              precision    recall  f1-score   support

          No       0.85      0.89      0.87      1035
         Yes       0.65      0.58      0.62       374

    accuracy                           0.81      1409
   macro avg       0.75      0.74      0.74      1409
weighted avg       0.80      0.81      0.80      1409
```

The major limitation of Logistic Regression is the assumption of linearity between the dependent variable and the independent variables. It not only provides a measure of how appropriate a predictor(coefficient size)is, but also its direction of association (positive or negative).

2. **Support vector classifier:**

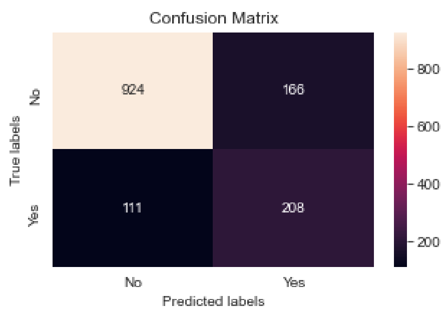
SVMs are supervised learning algorithms for classification, regression, and outlier identification. Support vector machines provide the following benefits:

- In high-dimensional environment it works well.
- When the number of dimensions exceeds the number of samples, the method remains effective.

➢ **How support vector classifier is better than logistic regression:**

While SVM aims to find the "best" margin (distance between the line and the support vectors) that separates the classes, logistic regression does not, and instead can have several decision boundaries with different weights that are near the optimal point.

Best prameters: kernel='linear', C=20



- **Classification report**

```
In [32]: print(classification_report(y_test, svm_pred, target_names=['No', 'Yes']))

              precision    recall  f1-score   support

          No       0.85      0.89      0.87      1035
         Yes       0.65      0.56      0.60       374

    accuracy                           0.80      1409
   macro avg       0.75      0.72      0.73      1409
weighted avg       0.80      0.80      0.80      1409
```

3. **XGBoost classifier:**

Extreme Gradient Boosting (XGBoost) is a distributed gradient-boosted decision tree (GBDT) machine learning toolkit that is scalable. It is the top machine learning package for regression, classification, and ranking tasks, and it includes parallel tree boosting.

- **Why does XGBoost perform better than SVM?**

As it meets a missing value on each node, XGBoost tries several things and learns which path to follow for missing values in the future. Random Forest is suitable for data sets with missing values. With incomplete data, SVM does not function effectively.
RandomizedSearchCV is used for hyperparameters tuning with StratifiedKFold of 5 splits.

- **Classification report**

```
In [68]: print(classification_report(y_test, xgb_pred, target_names=['No', 'Yes']))
```

```
              precision    recall  f1-score   support

          No       0.85      0.85      0.85      1035
         Yes       0.58      0.59      0.59       374

    accuracy                           0.78      1409
   macro avg       0.72      0.72      0.72      1409
weighted avg       0.78      0.78      0.78      1409
```

4. **Multi-layer Perceptron (MLP) classifier:**

Multi layer perceptron (MLP) is a supplement of feed forward neural network. It consists of three types of layers—the input layer, output layer and hidden layer.
The multilayer perceptron (MLP) is used for a variety of tasks, such as stock analysis, image identification, spam detection, and election voting predictions.

- **Classification report**

```
print(classification_report(mlp_pred,y_test, target_names=['No', 'Yes']))
```

```
              precision    recall  f1-score   support

          No       0.88      0.84      0.86      1074
         Yes       0.55      0.61      0.58       335

    accuracy                           0.79      1409
   macro avg       0.71      0.73      0.72      1409
weighted avg       0.80      0.79      0.79      1409
```

- **Why does MLP perform better than XGBoost?**
- Xgboost is an approach that focuses on interpretation, whereas neural nets-based deep learning focuses on accuracy.
- Xgboost is best for tabular data with few variables, but neural nets-based deep learning is best for pictures or data with many variables.

Conclusion:

So, we concluded that Multi-layer Perceptron MLP is best for large number of data.