Predicting Customer Churn Using Machine Learning

Shivam Soni
sonishivama@gmail.com

ABSTRACT

Customer churn is a recurrent problem in the various sectors The following project aims to take data of bank customers and use it to train classification models that will be able to predict future churn behavior of customers. The concepts of Data Preprocessing, and Classification are applied for the same.

1. Overview

Churn Rate" is a business term describing the rate at which customers leave or cease paying for a product or service. It's a critical figure in many businesses, as it's often the case that acquiring new customers is a lot more costly than retaining existing ones (in some cases, 5 to 20 times more expensive).

Understanding what keeps customers engaged, therefore, is incredibly valuable, as it is a logical foundation from which to develop retention strategies and roll out operational practices aimed to keep customers from walking out the door. Consequently, there's growing interest among companies to develop better churn-detection techniques, leading many to look to data mining and machine learning for new and creative approaches.

Predicting churn is particularly important for businesses w/ subscription models such as cell phone, cable, or merchant credit card processing plans. But modeling churn has wide reaching applications in many domains.

1.1 The Problem

To avoid customer churn, the project aims to accurately predict whether a customer is going to churn soon or not provided some information about him. The following sections will provide more insight into how the same is achieved.

1.2 The Data

The customer data provided consists of 10000 records of 14 attributes in a csv file. Of the attributes,.The attributes belong to following major logical categories:

:Attributes related to usage (tenure, estimated_salary, credit score, etc.)
:Personal Attributes (surname, age,gender geography, etc.)

3. DATA AND PREPROCESSING

IMPORTING DATASET-
The csv file is imported in the Python console with the simple command:

```
dataset = pd.read_csv('Churn_Modelling.csv')
```

With the file read, we can manipulate the attributes as needed

ENCODING CATEGORICAL DATA-
Encoding is done to convert data containg text to apropriate numbers.

This can be done using the scikit learn librabry and its class LabelEncoder:

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder_X_1 = LabelEncoder()
X[:, 1] = labelencoder_X_1.fit_transform(X[:, 1])
labelencoder_X_2 = LabelEncoder()
X[:, 2] = labelencoder_X_2.fit_transform(X[:, 2])
  #Not ordinal, therefore we need to creat dummy variable for countries (as more than 2) --> includes also a new measure
onehotencoder = OneHotEncoder(categorical_features = [1])
X = onehotencoder.fit_transform(X).toarray()
X=X[:, 1:]#to avoid dummy variable trap, we need to drop one of countries
```

SPLITTING TO TEST SET AND TRAINING SET
The data is divided into training and test datasets of 8000 and 2000 records each according to values of the attribute. All models will be trained on the Training dataset and validated on the Test set.

4. Classifier

We make use of XGBoost (eXtreme Gradient Boosting) which is an advanced implementation of gradient boosting algorithm.

The implementation of the model supports the features of the scikit-learn and R implementations, with new additions like regularization. Three main forms of gradient boosting are supported:

:Gradient Boosting algorithm also called gradient boosting machine including the learning rate.
:Stochastic Gradient Boosting with sub-sampling at the row, column and column per split levels.
:Regularized Gradient Boosting with both L1 and L2 regularization.

XGBoost dominates structured or tabular datasets on classification and regression predictive modeling problems.

It can be implemented using the following code(fiited to training set):

```
from xgboost import XGBClassifier
classifier = XGBClassifier()
classifier.fit(X_train, y_train)

# Predicting the Test set results
y_pred = classifier.predict(X_test)

# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

CONFUSION MATRIX

	0	1
0	1521	74
1	197	208

Classifier accuracies :
Received accuracy of 86%

4.1 Conclusion

In conclusion, I tried several machine learning algorithms to get get best accuracy for test set and result were best from xgboost classifier . Future work might attempt to develope a better feature set or a better algorithm.