PREDICTING THE TELCOM CUSTOMER CHURN

Sanjay Kumar Mahto

25-10-2020

CONTENTS

1. Problem Identification
2. Data wrangling
3. Exploratory data analysis
4. Preprocessing and training data development
5. Modelling
6. Using model and recommendations
7. Future work
8. Conclusion

Problem Identification

===========================================================================

The customer churn or customer attrition generally refers to a situation when customers stop using services of a company or simply leave that particular company. There can be multiple reasons of customer leave a company due to moving to other companies that offers better services, bad customer experience, or simply not needed the service any more. Suppose a particular customer was using air conditioner of a company but it requires repair due to faulty manufacturing etc. In market many ac’s of other companies have better ratings and after sales services at the similar price bases, so customer will churn. By being aware and monitoring churn rate, companies can determine customer retention success rates and identify strategies for improvement and providing better services. Here the machine learning model is being used to understand the precise customer behaviors and attributes which hints the likeliness of whether the customer will churn or not. Software requirements are Anaconda environment with Python and libraries as numpy, pandas, matplotlib and seaborn etc.

Data wrangling

===========================================================================

The dataset has been taken from [kaggle](https://www.kaggle.com/blastchar/telco-customer-churn/download) , has 7043 entries . All the columns has many features and a column called Churn that state if the customer has churned or not. The details are:

* **customerID:** Customer ID
* **gender:** Whether the customer is a male or a female
* **SeniorCitizen:** Whether the customer is a senior citizen or not (1, 0)
* **Partner:** Whether the customer has a partner or not (Yes, No)
* **Dependents:** Whether the customer has dependents or not (Yes, No)
* **Tenure:** Number of months the customer has stayed with the company
* **PhoneService:** Whether the customer has a phone service or not (Yes, No)
* **MultipleLines:** Whether the customer has multiple lines or not (Yes, No, No phone service)
* **InternetService:** Customer’s internet service provider (DSL, Fiber optic, No)
* **OnlineSecurity:** Whether the customer has online security or not (Yes, No, No internet service)
* **OnlineBackup:** Whether the customer has online backup or not (Yes, No, No internet service)
* **DeviceProtection:** Whether the customer has device protection or not (Yes, No, No internet service)
* **TechSupport:** Whether the customer has tech support or not (Yes, No, No internet service)
* **StreamingTV:** Whether the customer has streaming TV or not (Yes, No, No internet service)
* **StreamingMovies:** Whether the customer has streaming movies or not (Yes, No, No internet ser
* **Contract:** The contract term of the customer (Month-to-month, One year, Two year)
* **PaperlessBilling:** Whether the customer has paperless billing or not (Yes, No)
* **PaymentMethod:** The customer’s payment method (Electronic check, Mailed check, Bank transfer (automatic), Credit card (automatic))
* **MonthlyCharges:** The amount charged to the customer monthly
* **TotalCharges:** The total amount charged to the customer
* **Churn:** Whether the customer churned or not (Yes or No)

It has To do so, the following code snippet has been excecuted load the dataset and explore it with commands. The full notebook can be accessed at [github code telco notebook](https://github.com/sanapplegates/datascienceprojects/blob/master/Machine-Learning/Real%20State%20Price%20Prediction/Bangalore%20House%20price%20prediction%20.ipynb).

telco\_customer\_data.info() gives detailed information about every column.

object: Object format means variables are categorical. Categorical variables in our dataset are: customerID, gender, partner, dependents, phone service, multiple lines, internet service, online security, online backup, device protection, tech support, streaming tv, streaming movies, contract, paperless billing, payment method, total charges, and churn.

int64: It represents the integer variables. Senior citizen and tenure are of this format.

float64: It represents the variables which have some decimal values involved. They are also numerical variables. There is only one variable with this format in our dataset which is monthly charges.

Here is found that Total charges is an object variable, changed to float .

telco\_customer\_data['TotalCharges']=telco\_customer\_data['TotalCharges'].replace('\s+',np.nan,regex=True)

telco\_customer\_data['TotalCharges'] = pd.to\_numeric(telco\_customer\_data['TotalCharges'])

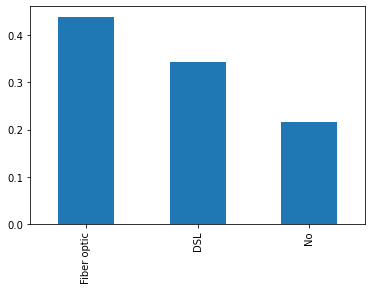
Exploratory data analysis

===========================================================================

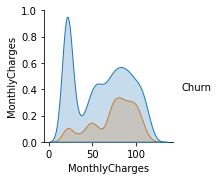
EDA refers to process of performing initial investigations on data set to discover patterns, spot anomalies, test hypothesis and check assumptions by summary statistics and graphical representations.

Here Python library -pandas\_profiling is used to generate the visuzalization and understanding of the distribution of each variable. It generates a report with all the information . It tells us the variables that contain NaN values, variables with many zeros, categorical variables with high cardinality, etc.

We can also use python pandas libraries to generate such visualisations and informations and find the underlying irelation between columns. For categorical features, we can use frequency table or bar plots which will calculate the number of each category in a particular variable. For numerical features, probability density plots can be used to look at the distribution of the variable. One such example: customer using following internet services : we find that most customer uses fiber optic for their internet services



Similary we can monthlycharges and churn rate of customers over month



More can be found in the [notebook](https://github.com/sanapplegates/datascienceprojects/blob/master/Machine-Learning/Real%20State%20Price%20Prediction/Bangalore%20House%20price%20prediction%20.ipynb)

Preprocessing and training data development

===========================================================================

Here , The dataset is to cleaned up by dropping irrelevant data, dealing with missing values,and converting data type to proper data type. In our dataset, we can see that customer ID is not needed for our model so we drop the variable. We do not need to treat missing values as there are none in this dataset. By using the Pandas function “get\_dummies()”, we can replace the gender column with “gender\_Female” and “gender\_Male”. We will use info() to show us which ones are categorical and numerical.

Splitting the dataset

First our model needs to be trained, second our model needs to be tested. Therefore, it is best to have two different datasets. As for now we only have one, it is very common to split the data accordingly. X is the data with the independent variables, Y is the data with the dependent variable. The test size variable determines in which ratio the data will be split. It is quite common to do this in a 70 Training / 30 Test ratio.

Using code snippet, using sklearn library

from sklearn.preprocessing import LabelEncoder # Churn to numeric value

label\_encoder = LabelEncoder()

telco\_customer\_data['Churn'] = label\_encoder.fit\_transform(telco\_customer\_data.Churn)

#also we can use pandas get\_dummies

telco\_df =pd.get\_dummies(telco\_customer\_data)

X = telco\_df.drop('Churn',axis=1)

X

y = telco\_df['Churn']

y

from sklearn.model\_selection import train\_test\_split

X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,y,test\_size=0.3,random\_state=20)

On splitting

X\_train.shape is (4930,45) which is 70% of dataset and X\_test shape is (2113,45) is 30% of dataset

Y\_train is 4930(70%) and y\_test is 2113(30%)

Modelling

===========================================================================We can use different models to predict the target variable. Using Scikit-learn library, different inbuilt functions can be used for modelling. The dataset has been already divided into training and test set in the previous section. Now let us use logistic regression and find accuracy score from sklearn module and fit the logistic regression model. Similarly, we can different algorithms apart from logistic regression for modeling.

from sklearn.linear\_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score

For logistic regression prediction

lm= LogisticRegession()

lm.fit(X\_train,y\_train)

y\_pred =lm.predict(X\_test)

lm\_accuracy = round(lm.score(X\_test,y\_test)\*100,2)

print(‘Test accuracy:’,lm\_accuracy)

Here we got accuracy of 81.45,

So for other alogos,accuracy score are

K-nearest neighbor ‘score is 78.04

Support Vector Classifier ‘s Score is 74.78

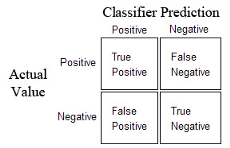
Decision Tree’s score is 75.72

Random forest’s score is 79.84

Out of these logistic regression model gives the very high accuracy score of 81.45 so this model will works better to predict the churn of customer

we can also check the confusion matrix for eg : taking for logistic regression

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known.



true positives (TP): These are cases in which we predicted yes (they have the disease), and they do have the disease.

true negatives (TN): We predicted no, and they don't have the disease.

false positives (FP): We predicted yes, but they don't actually have the disease. (Also known as a "Type I error.")

false negatives (FN): We predicted no, but they actually do have the disease. (Also known as a "Type II error.")

from sklearn.metrics import confusion\_matrix,recall\_score

cm\_lr = confusion\_matrix(y\_test,y\_pred)

print("confusion\_matrix:\n",cm\_lr)

out put is

confusion\_matrix:

[[1403 177]

[ 249 284]]

from sklearn.metrics import classification\_report

print(classification\_report(y\_test,y\_pred))

precision recall f1-score support

0 0.85 0.89 0.87 1580

1 0.62 0.53 0.57 533

accuracy 0.80 2113

macro avg 0.73 0.71 0.72 2113

weighted avg 0.79 0.80 0.79 2113

Using model and recommendations

===========================================================================

Conclusion

===========================================================================