Minh Tuan Vu

3rd May,2020

Capstone Project

Machine learning Customer Churn

Definition

Project Overview

Customer churn in simple terms means the customer chooses to end your services and goes to another provider.

Companies whose revenue generation solely depends on subscription are largely affected by customer churn. Churn rate is defined as the number of individuals which move out of a particular group over a period of time. Telecom companies today face extremely stiff competition today, as each company comes with a new scheme every month to attract customers. Losing customers in such a business is very costly, as it is extremely difficult to attract NEW customers. Bad service reviews on the internet, word of mouth reviews are factors that discourage potential customers from joining a new network. It becomes extremely essential for such companies, to maintain its current clientele. Finding out what could be possible factors that cause a customer to churn would help companies offer better services customised to a particular user's needs. At times it could happen that a particular user has a higher need for data and the current provider turns out to be too expensive and the user switches to another company. Predicting which users could possible switch can help telecom companies devise a new plan for those specific customers so as to prevent them from leaving the network. In fact this is not new, the moment you put in a request to discontinue services, and you get a call from them with some new plan to entice you. However, from a personal experience, once a customer is fed up with your services, and has already made the decision to switch, it is extremely difficult to convince them to stay. It is better to offer such plans BEFORE they make the decision.

Problem Statement

This project aims to predict if and when a customer could probably churn based on the company's data from the previous month, so as to offer those customers better services. This is a supervised learning problem

At the very basic level, the tasks involved in this are

- Load the dataset from IBM'S Telecom customer churn dataset.
- This dataset contains a lot of categorical variables. Pre-process this categorical data.
- Train a Logistic Regression model on this cleaned data, make predictions and note the accuracy. This would be used as a benchmark model for the project
- Train a XGBoost Model.Tune it's hyperparameters.
- Train a neural network. Tune it's hyperparameters.

Metrics

Confusion Matrix: A confusion Matrix number of correctly classified positives and negatives- true positives and true negatives, and incorrectly classified positives and negatives i.e false positives and false negatives. Although not a measure on it's own, its values are used to calculate precision, recall accuracy.

Accuracy is one of the most common metrics for binary classifiers.

Accuracy=True positives+true negatives/TP+FP+TN+FN.

Accuracy is a good measure to use here as , the dataset is not imbalanced. In case of imbalanced datasets, it is more likely to predict in favour of the target variable which is present in large proportion in the training dataset. While it incorrectly make predictions, the accuracy metric will still be high in value though that does not mean the model performs well.

ANALYSIS

Data Exploration

Dataset

IBM Watson Telecom customer churn Dataset

https://www.ibm.com/communities/analytics/watson-analytics-blog/guide-to-sample-datasets/

The name of the Data set is WA_Fn UseC_ Telco Customer Churn.csv.

This data set contains 7043 rows and 21 columns. Currently the dataset does not seem to have an imbalanced dataset.

Columns:

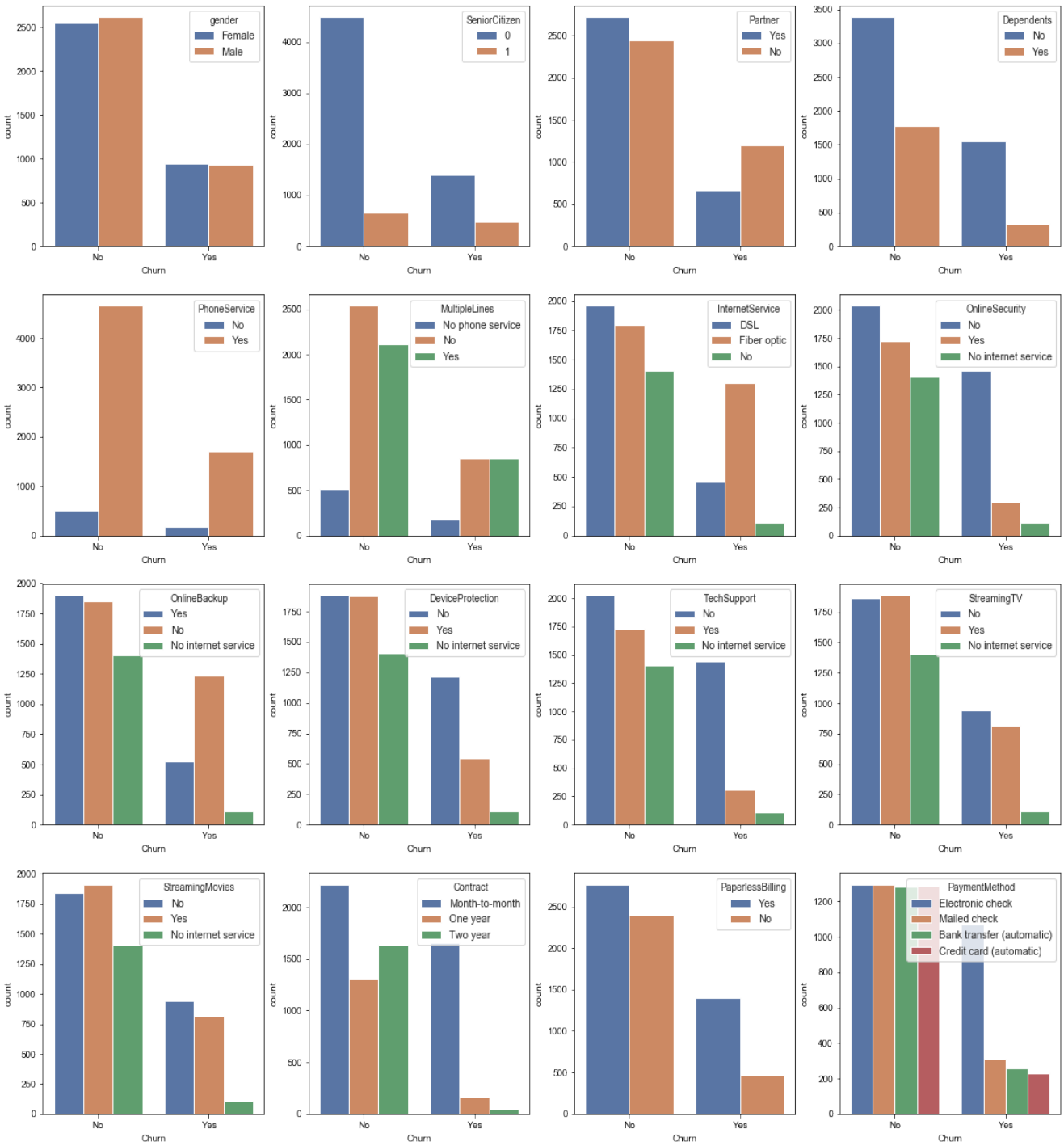
- Churn: the column we're trying to predict. This has a value of 0 or 1. This represents which customer left churned within the last month.
- Service Information: The types of services provided to customers-phone, multiple lines, internet, online security, online backup, device protection, tech support, and streaming TV and movie. They contain Yes or No as values.
- Demographic information
- Gender: Male or Female
- Senior Citizen
- Partners: Yes or No Values.
- Dependents: Whether they have anyone dependent on them financially. Yes or No values.
- Customer account information.
- Customer id: Unique id of customer of type char
- Tenure: how long they've been a customer of this company.
- Payment method: Contains values like Electronic check, mailed check, bank transfer
- Paperless Billing: Values of Yes or No. Indicator of whether customer has opted for paperless billing.
- Monthly charges: Charge per month. Numeric values.
- Total charges: Contains numeric values. Indicates charges over the contract period.

- Contract: Indicates whether the customer has a month to month contract, or a yearly contract.
- Inputs- All columns except churn column would be the input.
- Output- the column Churn is the column we're trying to predict.

Data Visualisation

- We have used bar plots, present in the notebook file, which indicate which category among the people who churned are more like to churn.
- For example, according to the graph, people with a two year contact were less likely to churn, and the highest number of people who churned were people with a month to month contract.
- Now this makes sense as the data taken is only from the last month, if you were to allow month to month contract as well as a two year one, people with a full two year contract would have to wait till the end of the contract to finish, and then they would switch.
- A month to month contract makes it very easy to switch as well. Also, according to the graphs, people with no device protection, no online security, and no tech support were far more likely to churn than people who were availing those facilities.

The visualisation below shows us the count of people churning from each category for categorical variables.



The mean values for numerical columns are given here.

	SeniorCitizen	tenure	MonthlyCharges
count	7043.000000	7043.000000	7043.000000
mean	0.162147	32.371149	64.761692

	SeniorCitizen	tenure	MonthlyCharges
std	0.368612	24.559481	30.090047
min	0.000000	0.000000	18.250000
25%	0.000000	9.000000	35.500000
50%	0.000000	29.000000	70.350000
75%	0.000000	55.000000	89.850000
max	1.000000	72.000000	118.750000

As most of the data was categorical, bar plots were used to plot relation between the target variable and a particular column

Algorithms and Techniques

1. Logistic Regression

- Logistic Regression is primarily used for classification problems. The most common examples would be classifying whether an email is spam or not spam, and in this case, classifying whether a customer may churn or no. Logistic Regression makes use of a sigmoid function to map the data into two categories, of 0 and 1 in terms of a binary classification. Logistic Regression makes use of Maximum Likelihood Regression to estimation of regression coefficients. The coefficients obtained are that set of coefficients for which the probability of getting the data we have observed is maximum

- This is used as the benchmark model.
- Logistic Regression was chosen here as it is one of the most common regression algorithms for binary classification in Supervised learning.
- Logistic Regression has shown an accuracy of around 78.60sss percent, which was sent as the benchmark metric to beat. Please look at Data Pre-processing section to understand how the data was encoded for this particular task.

2. XGBOOST
    - XGBOOST stands for Xtreme Gradient Boosting. The idea of boosting means, you use multiple weak learners to build a stronger model. Each weak learner is a decision tree

with a single split. You use multiple learners to build a stronger and robust model, as each learner qualifies and corrects a mistake made by a previous learner.
For the basics of Boosting, please refer

- Gradient Boosting is a method where new models are created to rectify the residual errors of the previous learner to ultimately make a strong learner. It is called gradient boosting because it uses gradient descent to minimize the loss function.

- XGBoost algorithm is an implementation of gradient boosted decision trees for performance and speed.
- This model is well suited for structured and tabular data and this was used to beat the performance of Logistic Regression.

3. Artificial Neural Network
    - Neural networks are built from neurons, which are trained on certain input data, and using activation functions, predict which class a particular data point falls. Each input connected to the layer has certain weights attached to it, which signify how important or how much of weight that particular feature has in predicting the target variable. The activation function, sums up these weights and functions, and based on a certain criteria, that neuron gets activated and allows the input to pass from that node. Cross entropy is then used to determine, how far the prediction is from the actual values and based on this, the weights for each node or input feature are adjusted in order to minimize the error. The neurons thus learn which features actually contribute and how much for making a prediction.
    - An artificial Neural network was built to understand patterns between the data, and was used this particular problem is a good fit for the neural network.
    - The training data isn't much, and can will not take that long to train. Neural networks update the weights for each node according the accuracy obtained during forward propagation, and learn which features provide more accuracy.

Benchmark model

The benchmark model whose performance we're looking to beat is Logistic Regression model. For performance of this model please refer to the Implementation

Methodology

Data Pre-processing

- First, the Total charges column was converted to a numeric value, and then the rows which had missing values for that column were dropped. The number of missing columns here were 11, which is a small number, and were hence dropped from the data frame.
- All the categorical variables were encoded and converted to numerical values. Pandas getdummies method was used to encode the data.
- Standard scaling was used to scale to balance the difference in values of the various columns
- The data was split into training and testing sets

Implementation

A Jupiter Notebook and the editor Spyder have been used for the implementation.

Logistic Regression

The following steps were performed for Logistic Regression.

- The data was loaded into a dataframe using pandas
- Pandas methods info() and describe were used to display statistics and general information about the dataset.
- Data Pre-processing was performed as mentioned above using libraries from sklearn.
- Data was split into training and testing as mentioned above.
- The module Logistic Regression was imported from sklearn.linear_models
- The training data was fit to the classifier and then a prediction was made on the testing set.
- Performance: This model gave an accuracy of 78.60 on unseen data.

XGBOOST

- Xgboost model was first imported.
- The XGBOOST classifier was fit onto the training set.
- A prediction was made on the test set.
- On the first prediction it had an accuracy of only over 78.89 percent, which is more than Logistic Regression, but only marginally
- Parameter tuning was then performed on it, after which the accuracy improved slightly to above 80 percent.

Artificial Neural Networks

- Installation of tensorflow and keras
- Importing classes like Sequential and Dense from keras modules
- Built a deep neural network using Sequential and Dense
- Set appropriate network parameters, loss function and estimator.
- Trained the network on training set
- Made a prediction on the testing data. Gave an accuracy of more than 79 percent, around 79.30

Refinement

XGBOOST

- For XGBoost , GridSearch CV was used to find out best values for n_Estimators,max_depth, Min_child_weight.

- Grid search cv was used to find out optimal parameters for XGBOOST
- After finidng out optimal parameters for them, the values were plugged into the model and the model was trained on the testing set.
- The grid search CV takes a bit of time to execute.

Max_Depth	Min_child_weight	Learningrate	Subsample	Colsamplebytree	Reg_alpha
Range(3,10,2)	Range(1,6,2)	0.1	0.8	0.8	
3	5,6,8,10,12	0.1	0.8	0.8	
3	12	0.1	Range(0.6,1)	Range(	
3	12	0.1	0.8	0.9	0.01

The accuracy was improved to about 80.02 percent on unseen data, which is more than the one obtained on Logistic Regression

Artificial Neural Network

- Dropout layers were added after the input layer and hidden layers as well, with a dropout rate of 0.2
- Grid search cv was conducted and the values for n_epoch and batch size was optimized.

Results

Model Evaluation and Validation

- Grid Search cv and k fold validation was used to test the model.
- The final parameters for XGBOOST were chosen because they gave the best accuracy, of around 80.5 percent. This was obtained by tuning each of the parameters and using grid search for each of the parameters.
- However, when a new model was created, its accuracy was only slightly better than that given by a logistic regression.
- For neural networks as well, accuracy is only more with a very tiny difference.

Model	Accuracy(in percent)
Logistic Regression	78.89
XGBOOST	80.02s
Artifical Neural Network	79.38

Conclusion

Reflection

The steps followed in this project can be summarized as follows

- Relevant dataset was found for the dataset.
- The data was loaded into dataframes in pandas and appropriate processing was performed.
- The data was split into training and testing sets

- Logistic Regression was chosen as a benchmark model whose performance XGBoost and am artificial neural network is supposed to beat
- The accuracy was found out both by using sklearn's function and generating the confusion matrix.
- XGBoost was imported and some further processing was done.
- Training data was fit to this classifier, and then tested on test data.
- GridSearch CV was used to find out the best parameters.
- Grid search CV here was a little slow
- Building a neural net for this problem once the pre-processing was done was fairly simple
- For the ANNS, grid search CV took a lot of time. Increasing the accuracy of the ANN was the most difficult part of the project.
- Dropout layers were added to try and tune the neural network.
- Before actually building an algorithm, lot of thinking had to go into the cleaning and pre-processing of data.
- I tried both one hot encoder and the getdummies method from pandas.
- I had to use predict_classes instead of predict as the final activation function in my neural network was sigmoid function.

Challenges faced

- The numpy installation had to be upgraded, which was giving a fatal error otherwise. The error is mentioned below
- `ModuleNotFoundError: No module named 'numpy.core._multiarray_umath'`

- Another error encountered because of the issue of the numpy version was.
  `An error ocurred while starting the kernel 2019 01:46:35.136525: F tensorflow/python/lib/core/bfloat16.cc:675] Check failed: PyBfloat16_Type.tp_base != nullptr`
- The Dense API had to be updated to the latest API.
- As the data had a lot of categorical variables, I tried both One Hot encoding and pandas.getdummies for this purpose.

Improvement

- Time for grid search cv of the neural network needs to be improved. Maybe better hardware or using a GPU should improve it.
- The models XGBOOST and ANN'S have not beaten the Logistic Regression model by a huge margin in terms of the accuracy score.