CHURN Prediction

Prasan Ratnayake

Introduction

The assignment folder contains two Jupiter note books assignment1 and assignment2 which have the code.

The dataset “Telecom_user dataset for interviews” was provided for churn prediction. Dataset comprises data for the telecom company. The dataset had several columns like 'tenure','MonthlyCharges' and 'TotalCharges', which shows the information of the customer. The last column gives the churn of the customer.

The company wants to know the customers that churn out from their service. Therefore we have to use a machine learning approach to design a model to predict the churn of the customers.

We will use python libraries like tensorflow, keras, numpy, matplotlib, pandas and sklearn.

Tensorflow/keras – implementing the deep learning mode and training

Numpy – for handling arrays

Pandas – to hold data imported as a data frame

pandas_profiling – was used to analyse the data set

Matplotlib - plot graphs

Sklearn – import machine learning models for logistic regression, random forest, decision tree and support vector machine

Methodology

In order to proceed further the raw data had to undergo filtration so best features could be extracted as well as to use these data in a compatible format for machine learning.

1. **Data Preparation**

Here in the data preparation part, the following changes were made to convert data in a readable way. First change was the conversion of “No internet service” and “No phone service” into “No” since these answers mean that the customers won’t use those services anymore.

Then “Yes” and “No” values were changed to 1 and 0. This was done because the final dataset used for training would be numeric, therefore before proceeding to the model data were transformed. Similarly “Male” and “Female” values were set into 1 and 0. All these alteration were done in excel using the replace method. Then excel was saved as another csv file.

The two columns customer id and the first column were removed as they won’t be needed for the training part. Then the data was imported to the python Jupiternote book using the pandas

read_csv method. Using pandas profiling was used in assignment2.ipynb to analyze the variables. It provides a detailed description of all variables.

It was observed that there were 3 categorical variables named as 'tenure','MonthlyCharges' and 'TotalCharges'. These three columns were checked to validate if they were numeric. But the column TotatlCharges was string and it had some rows which had null values. Those were dropped and TotatlCharges was converted to float type afterwards. Figure below shows that how the TotalCharges was turned to numeric form using the pd.to_numeric method.

```
#converting TotalCharges to numeric after dropping null rows
data_set1.TotalCharges = pd.to_numeric(data_set1.TotalCharges)
data_set1.TotalCharges.dtypes
```

One hot encoding allows the representation of categorical data to be more expressive, therefore the categorical data columns as 'tenure','MonthlyCharges' and 'TotalCharges' were given dummy variables. Following figure shows how it was done.

```
#one hot encoding is applied to categorical data
data_set2 = pd.get_dummies(data=data_set1,columns = ['InternetService','PaymentMethod','Contract'],drop_first=True)
```

Then the values of these columns should be scaled before proceeding with the training. For that purpose the MinMaxScalar was used to convert these column values to numeric values between 0 and 1. Following shows the implementations of the MinMaxScalar.

```
#these columns were scaled so these values will be within 0 and 1 range
cols_to_scale = ['tenure','MonthlyCharges','TotalCharges']

from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
data_set2[cols_to_scale] = scaler.fit_transform(data_set2[cols_to_scale])
```

2. **Training and Test data**

The carefully prepared data was checked if all were numeric. After ensuring the all data was numeric we set 80% of the data for training and 20% for testing and validation. Since this was a small data set we have to use this ratio for more accurate results.

```
X = data_set2.drop('Churn',axis='columns')#input
y = data_set2['Churn']#target

#data set is made so that 20% is test data
#and 80% is the train data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,random_state=5)
```

3. Training Model

In this part we use a supervised learning model using keras and four other statistical machine learning models Logistic Regression, random Forest, Support Vector Machine and Decision Tree algorithm. We will evaluate the final accuracy in each model and see which model has the best prediction for the given data set.

Model using keras

The model was made to run for 150 epochs. The dense layers uses the relu function and the last layer uses sigmoid function as this is a classification problem.

The optimizer is adam since it is able to handle sparse gradients and noisy problems. Loss function is binary cross entropy since it is used for classification problems.

For improving the accuracy of the model a dropout layer was introduced at the beginning. As techniques used for prevent over fitting early stopping was used. Number of steps for being patient was given as 40.

```
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
import matplotlib.pyplot as plt
from keras.callbacks import EarlyStopping
import numpy

#model
model = Sequential()
model.add(Dropout(0.2, input_shape=(23,)))
model.add(Dense(23, input_shape=(23,), activation='relu'))
model.add(Dense(12, activation='relu'))
model.add(Dense(4, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

# early stopping
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=40)

model.compile(optimizer='adam',loss='binary_crossentropy', metrics=['accuracy'])
history = model.fit(X_train, y_train, epochs=150,batch_size=4, verbose=0,validat
```

Using the previously mentioned training data and test data we used sklearn library to import the training model and accuracy for the model was predicted. Following code were implemented for the model. In every model train and test data were split into 80% and 20%.

Here we present how the training was done for those models.

Logistic Regression model

```
#Fit the Logistic Regression Model
LogisticRegression = LogisticRegression(random_state=59)
LogisticRegression.fit(X_train,y_train)

#predict the value for new, unseen data
pred = LogisticRegression.predict(X_test)

#Find Accuracy using accuracy score method
LogisticRegression_accuracy = round(metrics.accuracy_score(y_test,pred)*100,2)
print(LogisticRegression_accuracy)
```

Logistic Regression model uses logistic function that measures the relationships between two dependent categorical and one or more independent variables. In this attempt it was found that Logistic Regression gave a higher accuracy for the dataset with 80.18 percentage.

Support vector Machine Model

```
#Fit the Support Vector Machine Model
smodel = SVC(kernel='linear',random_state=50,probability=True)
smodel.fit(X_train,y_train)

#Predict the value for new,unseen data
s_pred = smodel.predict(X_test)

#Find Accuracy using accuracy sscore method
s_accuracy = round(metrics.accuracy_score(y_test,s_pred)*100,2)
print(s_accuracy)
```

Support Vector Machine can be used for higher order dimensions in regression and classification problems.

Decision Tree Model

A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements.

```
#Fit the Decision Tree Classification Model
from sklearn.tree import DecisionTreeClassifier
dtmodel = DecisionTreeClassifier(criterion = "gini", random_state =50)
dtmodel.fit(X_train,y_train)

#Predict the value for new, unseen data
dt_pred = dtmodel.predict(X_test)

#Find Accuracy using accuraccy score method
dt_accuracy = round(metrics.accuracy_score(y_test,dt_pred)*100,2)
print(dt_accuracy)
```

Random Forest

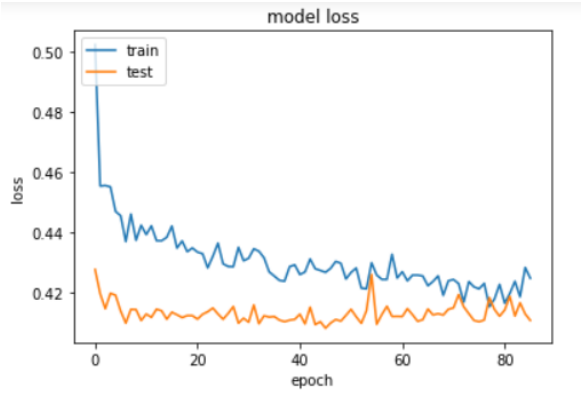
Random forest is another regression type solver which use a large number of decision trees and outputting an average of the result of individual trees.

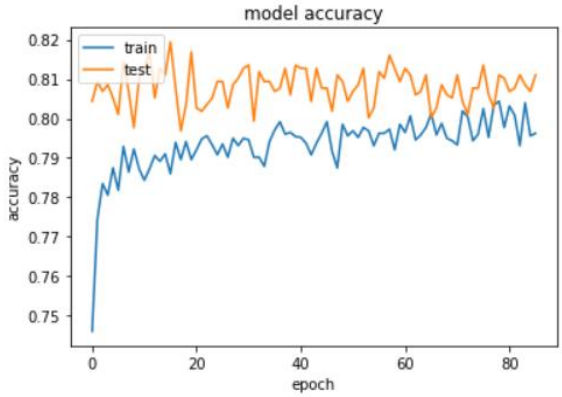
```
#Fit the Random Forest Classification Model
from sklearn.ensemble import RandomForestClassifier
rfmodel = RandomForestClassifier(n_estimators = 100, criterion = 'entropy',rando
rfmodel.fit(X_train,y_train)

#Predict the value for new, unseen data
rf_pred = rfmodel.predict(X_test)

#Find Accuracy using accuracy score method
rf_accuracy = round(metrics.accuracy_score(y_test,rf_pred)*100,2)
print(rf_accuracy)
```

4. Results

Summary of the results can be shown below. Which show that the model used in keras has the best accuracy so far. Also Logistic Regression stands at the second place.

Training model	Accuracy
Keras	81.10
Logistic Regression	80.18
Random Forest	79.52
Support Vector Machine	78.51
Decision Tree	72.83





Above figures show the variation for the accuracy and loss. As the first diagram suggests the training loss was above the test loss which showed no overfitting. As early stopping was called the training stopped after 86 epochs and returned an accuracy of 81.1%. Here training accuracy was bit lower than the test accuracy.

```
In [20]: model.evaluate(X_test, y_test)
         38/38 [==============================] - 0s 2ms/step - loss: 0.4106 - accuracy: 0.8110
Out[20]: [0.41055989265441895, 0.8110367655754089]
```

5. Discussion and Analysis

From the above results it is clear that the models obtained using logistic regression and using keras were best for churn predictions. The results could be improved if the model achieved a far greater accuracy than this. We could use a very large data set for a best fitted model since more data could increase the accuracy.

Since missing data was removed from the dataset it can be thought as a measure of preparing clean data. Also we have compared different statistical and machine learning models for the best accurate answer. It is always good to go through many other ways of getting the training done for best accuracy. Detection of outliers in this data set was quiet uncommon to have. Therefore we can neglect the presence of outliers for this dataset.

Also we treated the categorical data as dummy values in this process and we have scaled them properly. If it is not the case, it would decrease the overall accuracy. So different techniques such as data transformation and scaling were done for better data input.

A measure for improving the accuracy in the model dropout layers was introduced at the beginning of the dense layer and early stopping was introduced for stopping overfitting of data.

Apart from that we can try different architecture models till we reach a better accuracy. Also we can implement the model in a parallel environment which uses GPUs like Google Colab. It would give better and enhanced performance.