Telecom Customer Churn Rate Predictor
Progress Report

Group Members:

Alba Mustafaj	21500009
Alp Ege Baştürk	21501267
Berat Biçer	21503050
Bora Ecer	21501757
H. Buğra Aydın	21501555

1 Introduction and Background Information

This project's objective is predicting user's churn rate, which occurs when a customer stops using the service. This information is important for a business which depends on continuous customer subscription. Information can be related with indicators like tenure and payment method. These indicators may seem straightforward, however there can be other features which can give additional information. Previously mentioned features like tenure, payment method, dependents etc. will be use to predict if a user is planning to stop using the service. Thus, several machine learning methods like SVM, kNN and logistic regression, decision tree classification, random forest classification and deep learning were applied to perform the prediction of user churn rate. The results will be later used to analyze the success rates of each algorithm to decide which algorithm provides better prediction. During the analysis process, the optimal parameters for each algorithm will firstly be detected and then algorithms will be compared to each other accordingly. Analysis results will give a better insight on which algorithms perform better at their optimal state.

2 Work Done So Far

Initially code for dataset preprocessing was written in Python. The dataset was preprocessed in order to create numerical fields from non-numeric ones. We have used pandas library to read the dataset, then convert non-numerical fields to numeric ones in the dataframe object and wrote to a processed data file. In addition we have removed unrelated field Customer-ID, which looks like an hashed data. This file was used by the training algorithms. The processed dataset was divided into train and test sets since models should be generalizable to data which they have not seen so far. This was done by using sklearn library. After data preparation, we have used the processed data with different algorithms.

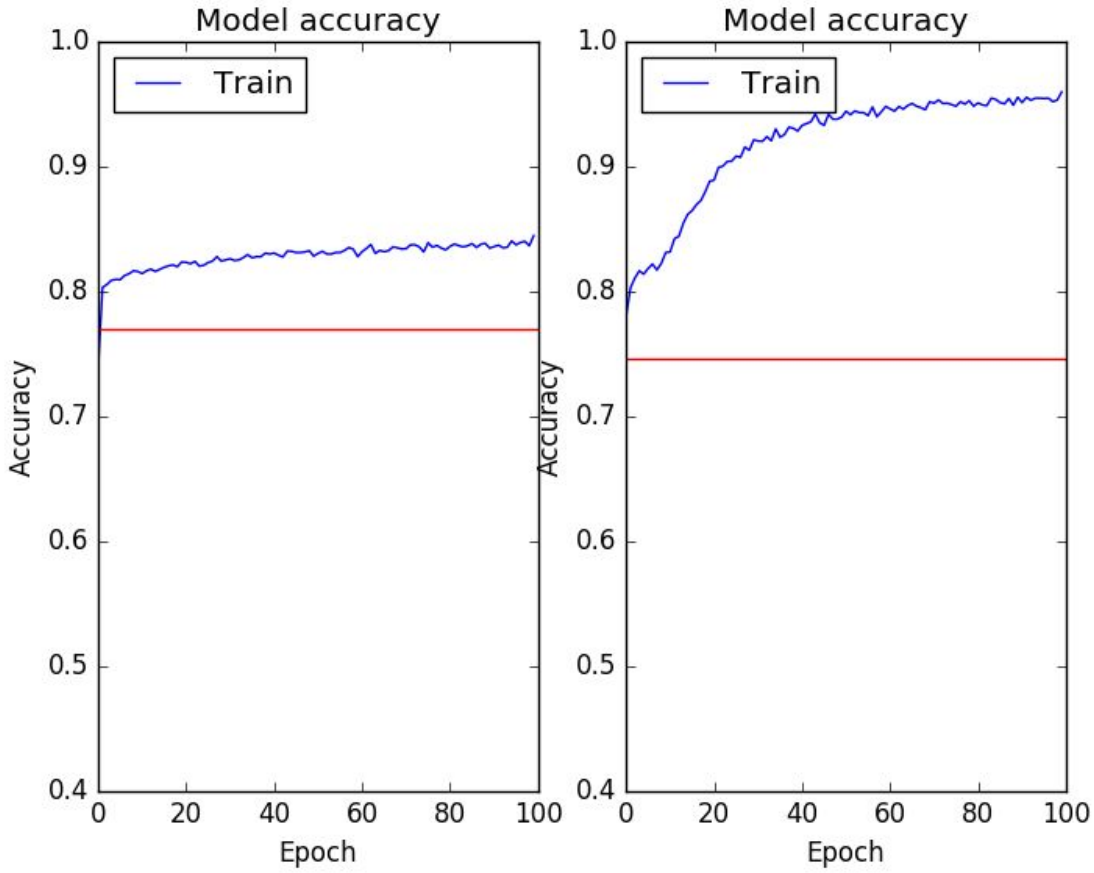
We have implemented prototypes for knn, logistic regression, random forest, SVM, decision tree and a simple neural network in Python using keras and sklearn libraries. We have given initial parameters which we think appropriate to test whether our code works. Also we have chosen train/test data split ratio as 60%. These Python programs use the processed training data to train the model, then get results on the test data. Following are the results obtained from initial parameters and default parameters of the library if not specified;

3NN	Accuracy	74.44996%		
	precision	recall	f1-score	support
0	0.81	0.83	0.82	2059
1	0.51	0.48	0.5	759
Logistic Regression	Accuracy	80.37615%		
	precision	recall	f1-score	support
0	0.85	0.89	0.87	2072
1	0.65	0.56	0.6	746
Random Forest	Accuracy	77.99858%		
	precision	recall	f1-score	support
0	0.81	0.91	0.86	2070
1	0.62	0.43	0.51	748
SVM	Accuracy	78.92122%		
	precision	recall	f1-score	support
0	0.83	0.89	0.86	2060
1	0.63	0.51	0.57	758
Decision Tree	Accuracy	72.78211%		
	precision	recall	f1-score	support
0	0.81	0.83	0.82	2049
1	0.5	0.47	0.48	769
Neural Network(Simple)	Accuracy	78.45990%		
	precision	recall	f1-score	support
0	0.83	0.87	0.85	2030
1	0.63	0.55	0.59	788
Neural Network(Advanced)	Accuracy	74.69837%		
	precision	recall	f1-score	support
0	0.82	0.84	0.83	2084
1	0.51	0.49	0.5	734

For knn, Initial choice of k=3 was determined using the information we have seen in the class. Thus it may not be optimal. Estimator number was chosen as 10 for random forest classifier. This value was also not analyzed completely so it may not be optimal. Used SVC with linear kernel for SVM. Different kernels like polynomial have not

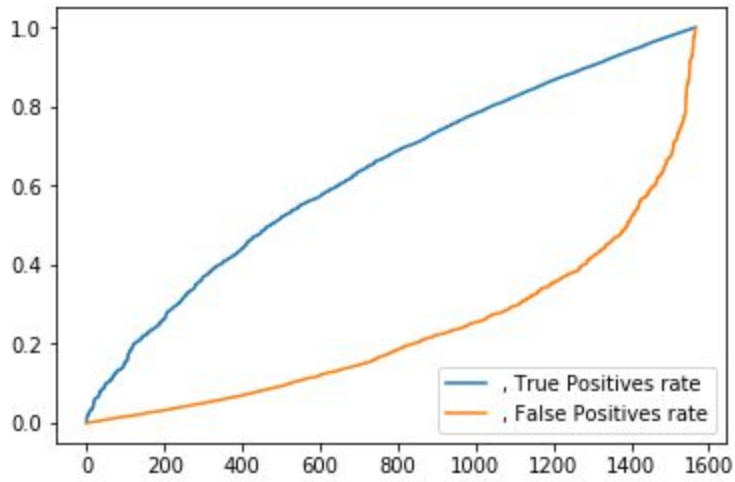
been tested thus these choices may not be optimal for the problem. Neural network (Simple) was constructed using 20 node input layer, single hidden layer with 5 nodes and an output layer with a single node. Relu was used for activation function of the first two layers and sigmoid was use for the last layer. Neural Network (Advanced) was constructed with input layer of 200 nodes, and four hidden layers of 150, 100, 50 and 25 neurons respectively. Output layer is the same as the simple one. This Neural network has lower accuracy on the test data compared to the simple one however, it had ~95% accuracy on the training data. Thus it can be said that complex network overfitted the training data. Using simpler networks is considered according to these results. Both NNs overfit the training data which can be understood from the data on the graphs.

2.1 Initial Graphs



Simple NN accuracy on train data

Complex NN accuracy on train data



True and false positive rates for Logistic regression

3 Remaining Work

We have implemented the infrastructure for different types of algorithms with initial parameters. In the remaining part, we will analyze the results, tune the parameters and find out the optimal parameters for each algorithm. We will generate graphs by tuning parameters of the models we have used so far. Also we may apply feature selection on the features. In addition, considering adding different models is still an option since it is easy to add new models. Furthermore, we may consider using a smaller portion of data for training since most of the models are relatively successful in terms of accuracy. Also, outlier detection and replacement can be a good solution to increase accuracy. It is possible to handle that by replacing the outlier values above 99.9% and under 0.01% weight with the weight 99.9% and 0.01%.

4 Division of Work

The work was divided for different models so far so that each member could take one of the algorithms to study. In addition, all members tried to learn the algorithms. Data preprocessing codes were written by Alp Ege Baştürk and Buğra Aydın. Alp Ege and Buğra has written the code skeletons to use keras and sklearn libraries. Also Buğra has merged the codes and created a Jupyter notebook file which is expected to reduce code repetition among the prototypes since most use similar parts. The remaining analysis work will be divided based on the algorithms such that each member will take one of the algorithms and tune its parameters. Also data and graphs for that algorithm will be prepared by the member responsible for that algorithm. Jupyter notebook file will

be maintained by Buğra and Alp Ege. Distribution of analysis work is as follows; Alba will analyze logistic regression, Alp Ege will analyze neural network, Buğra will analyze random forest, Bora will analyze decision tree and Berat will analyze knn. Work for SVM will be redistributed according to progress of members.