CS464
Introduction to Machine Learning
Telco Customer Churn Rate Predictor
Final Report

Group 5

Group Members:

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

24.12.2018

Table Of Contents

1 Introduction

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 custog

mer subscription. 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 neural networks were applied to perform the prediction of user churn rate. The results were 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 were detected and then algorithms are compared to each other accordingly. Analysis results gave a better insight on which algorithms perform better at their optimal state.

2 Problem Description

The problem is users leaving a subscription service, which is a telecom service in our case. Knowing this might help future predictions about income and more importantly it might help making decisions to stop users from leaving. The question which is needed to be answered is predicting whether a user with given features will churn or not. Because all other actions depend on this. Thus, we have used different models to address this prediction problem. Also we have implemented a test code to gives price reductions. This is a basic loop which reduces monthly payments down to a certain point or until prediction turns from churn to not churn for given features.

3 Methods

We have applied SVM, kNN, logistic regression, decision tree, random forest classifiers on the data and also implemented a basic neural network using keras. Our dataset is Telco customer data [1].

3.1 Data Processing

Initial dataset included non numeric fields like gender, true/false fields and missing data. We have encoded these fields with numeric ones to be able to use them. Also user IDs were meaningless hashed field so we have dealt with such occurrences.

After converting data to useable format we have splitted the data to train and test data.

3.2 Model Description

We have applied aforementioned classifiers on the data. In this section, we explain the models, how they do the classification task and some important hyper para Cross validation was applied. In addition to cross validation, feature and parameter selections were applied in order to increase performance. Also, we have applied train/test ratio test to find out in which ratio the division of dataset into train and test datasets provided better accuracy. Also, we have used Confusion Matrix, Precision-Recall, ROC as our performance measures

3.2.1 SVM

Support Vector Machines discriminate data by finding separating hyperplanes. There are different options to create kernels such as linear, polynomial and rbf. Linear kernel tries to use linear relationships, which was not helpful considering the dataset. Rbf kernel can detect non-linear relationships, so this was a good choice for the dataset. C parameter gives regularization. For large C it acts like hard margin. Gamma determines influence of further data points on the calculation.

Feature selection using recursive feature elimination was used in order to get first *i* best features. Also hyperparameter selection was applied in order to choose kernel type, C and gamma parameters.

3.2.2 Neural Network

Neural networks try to reduce prediction error by making prediction and readjusting weights according to error. Two neural networks were used for experimenting with the dataset. One of them was a small network while the other one was a large network. Neural networks were not expected to be main focus of the project since other methods were expected to provide sufficient accuracy according to initial tests. However, results from the neural network experiments were used to support the limitations of dataset on the accuracy.

3.2.3 Decision Tree

Decision Tree method generates a tree-like model of decisions from the training data. The internal nodes in the decision tree represents a test on an attribute based on data whereas branching from a internal node represents an outcome of the test which can be binary yes/no condition or continuous value comparison (<, >, etc). The leaf nodes in the decision tree represents a class label or class distribution.The branching of the best split from an internal node is determined by the measuring impurity for class label distribution on that node. The common measures of the impurity are Information Gain using Entropy, Gini Index.Also, there are some other hyper parameters affecting the accuracy of the Decision Tree model. So the

overall hyper parameter to consider the impurity criterion, the maximum depth of the decision tree, minimum number of samples required to be at a leaf node, minimum number of samples required to split an internal node and finally splitter strategy.
The description of these hyper parameters can be given as following:

- Maximum depth of the tree hyper parameter denotes the maximum depth that the decision tree can have. Also, None parameter denotes that, during training phase the nodes will be expanded until all leaves contain the less samples than the minimum number of samples required to split an internal node.
- Minimum number of samples required to be at a leaf node denotes a split point at any depth will only be considered as a leaf if it has at least minimum number of samples required to be at a leaf node
- Minimum number of samples required to split an internal node limits the splitting so that an internal node can only be splitted if the required sample amount condition is met, this allows the model to have more educated decisions.
- As for the criterion, the information gain using entropy and gini index of a node split are given by the following formulas:
    - Entropy:
        - $Entropy = \sum -p_i log_2(p_i)$ in which p_i is the probability of class i
        - $Information\ Gain = Entropy_{Parent} - Average\ of\ Entropy_{children}$
        - the decision tree model favors splits with higher information gain.
    - Gini Index:
        - $Gini(n) = 1 - \sum_i p(i \mid n)^2$ in which $p(i \mid n)$ is the relative frequency of class i at node n.
        - Gini index is maximum (1- 1/ # of classes) when samples are equally distributed among all classes.
        - Gini index is minimum (0) when all samples belong to one class
        - Thus, the decision tree model favors splits with small gini index.
- As for the splitter strategy: best splitter strategy is used to choose the best split along the best feature among all possible features all possible splits and the random splitter strategy is used to choose a random feature splits it, and calculates gini index/information gain for that split, it repeats this random splits a number of times, and then selects the best one among them. The random splitter is less prone to overfitting than the best splitter.

In the section 4.3 experiments related to the hyper parameter values, as well as the feature selection, test/train ratio test results and performance measurement results are given.

3.2.4 KNN Classifier

k-nearest neighbors algorithm, KNN for short, is a nonparametric method used for classification and regression. In this project, we use KNN for binary classification which are churn and not-churn. Inputs to a KNN classifier are training examples in the feature space, and number of neighbors to be considered when predicting class label. An object is classified as class L iff majority of its neighboring samples are assigned to class L. One problem with KNN is that assignment of samples are dominated by the most frequent class if the data distribution is skewed. Most common approach in determining distance between any two samples is using Euclidean distance. If, as in our case, variables are discrete, Hamming distance can be used instead. One approach in improving test accuracy is to learn distance function from the data so that closely related samples are close to each other compared to another, irrelevant sample. Hyperparameters for this function are listed below:

- Number of closest neighbors to use when determining class assignment.
- Weights determining how assignment of a sample effects predicting a new sample. To clarify, a point with large weight will be more dominant compared to another one with small weight. Possible values are uniform where all points in the neighborhood contributes equally, distance where contribution is inversely proportional to the distance between two points, and a user defined weight function computed for each sample.
- Algorithm used to compute the nearest neighbors. Possible options are using a Ball Tree, using a KD Tree, a brute-force search, or an automatic selection based on the dataset.
- If a Ball Tree or a KD Tree is used, a leaf size defines the size of a leaf passed on to tree. Leaf size affects the speed of constructing the tree and queries on the tree, as well as memory requirements for storing the tree.
- Distance metric used for the trees. Default value is Minkowski distance with parameter p. When p is 1 it is equivalent to using Manhattan distance l1, and when p is 2, it is equivalent to Euclidean distance l2.

3.2.5 Logistic regression

Logistic regression is based on predicting coefficients, encapsulating probabilities between 0-1 and predicting the labels based on a certain threshold. Several train and test set sizes were used for this method in order to deduce the change in performance. The targeted parameters were: penalty which specifies the penalty norm, tolerance for stopping criteria, dual, fit intercept which specifies if a constant should be added to the decision function. Solver is also an important parameter which identifies the algorithm to be used in optimization. Considering the type of classification and size of our dataset this parameter should be tuned accordingly. Multi_class should also be set to ovr in order to be applicable for the binary classification problem. Furthermore, feature selection was performed using SelectKBest function which takes a score function parameter and the number of desired features. Cross validation was also performed several times with different number of folds and train/test sizes to observe the change in performance and obtain the best results. Confusion matrices and ROC curves were

generated and analysed in order to draw conclusions about the best parameters. Tables were also constructed to simplify the analysis process and compare results.

3.2.6 Random Forest Classifier

Random Forest Classification works in a divide and conquer manner. It creates many training samples from the training dataset, then creates many decision trees from those samples. Then it uses those decision trees while predicting the output. The outcome with the highest amount of votes from those decision trees is selected. Biggest advantage of this algorithm is that it eliminates the overfitting problem by creating many different decision trees and cancels out the biases. It has a downside, which is being slow due to amount of decision trees involved in the process. However, most of the modern libraries eliminate this problem by using parallelism.

In the scikit learn library, the algorithm is provided with several parameters. It is possible to give an exact number for the decision tree amount, customize the depth, minimum samples to form a leaf and maximum leaf nodes. Those parameters were used during the experiments.

4 Results

All experiments aimed to find optimal accuracy using that model.
The experiments we have followed were to answer the following questions:

- Which are the best hyper parameter values that results with the highest accuracy?
- Which are the best features that results with the highest accuracy once used with the best hyper parameters?
- Which train/test split is better in terms of accuracy?
- What is the relationship between the selected feature amount and the accuracy?
- What are the ROC, Precision-Recall results?

4.1 SVM

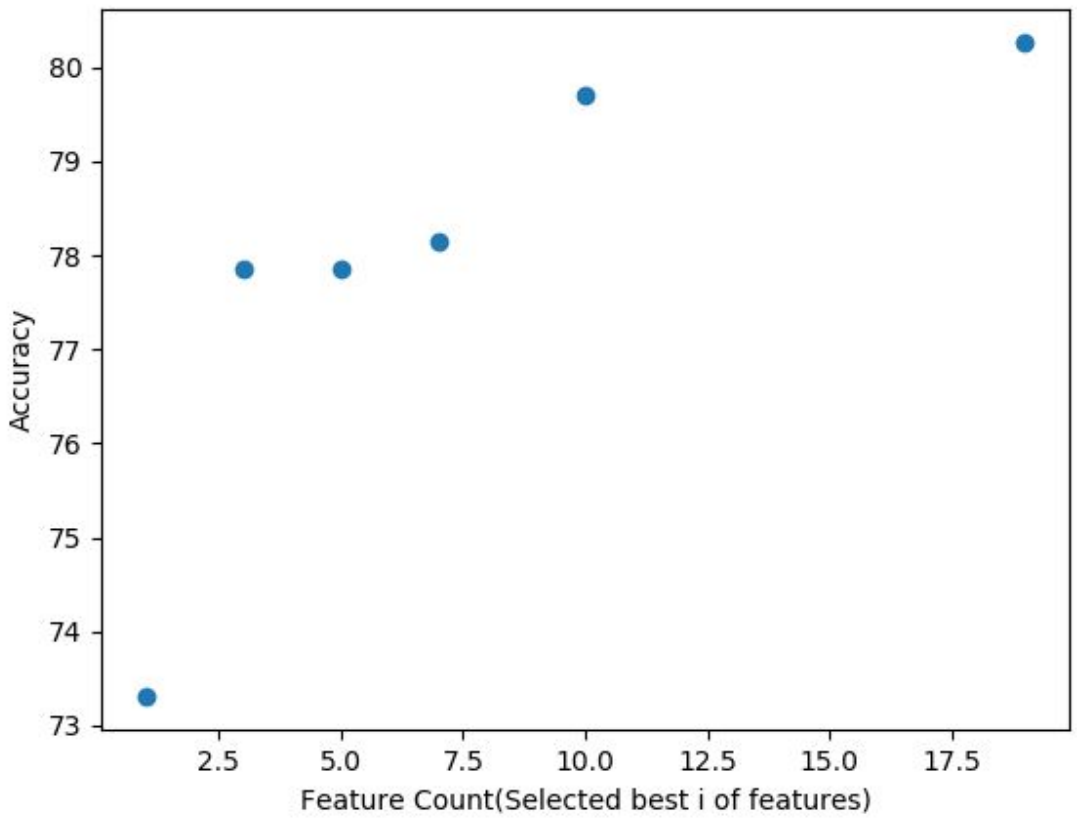
Following results were obtained using 40% of the dataset as test set. However choosing larger or smaller test sets didn't have significant effects, thus no graphs were drawn since results were inseparable from noise. In the case where test set was chosen as 5%, accuracy jumped to 83% however it dropped to 78% in the next run, thus results were unreliable for small test sets and there were no significant gains in terms of accuracy.

Accuracy: 80.207%
Confusion Matrix:

Actual \ Predicted	0	1
0	1906	173
1	386	353

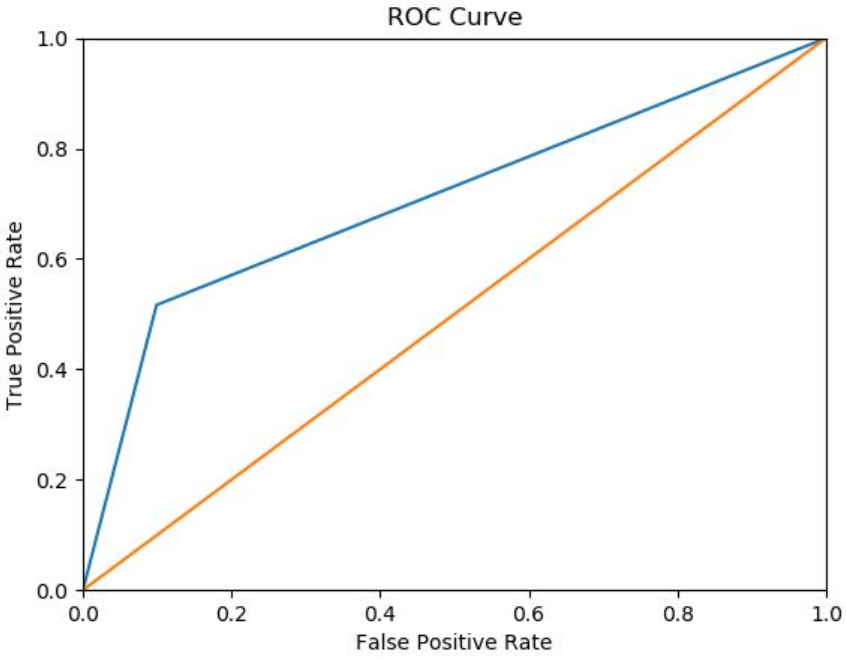
	precision	recall	f1-score	support
0	0.83	0.92	0.87	2079
1	0.67	0.48	0.56	739
micro_avg	0.8	0.8	0.8	2818
macro_avg	0.75	0.7	0.72	2818
weighted_avg	0.79	0.8	0.79	2818

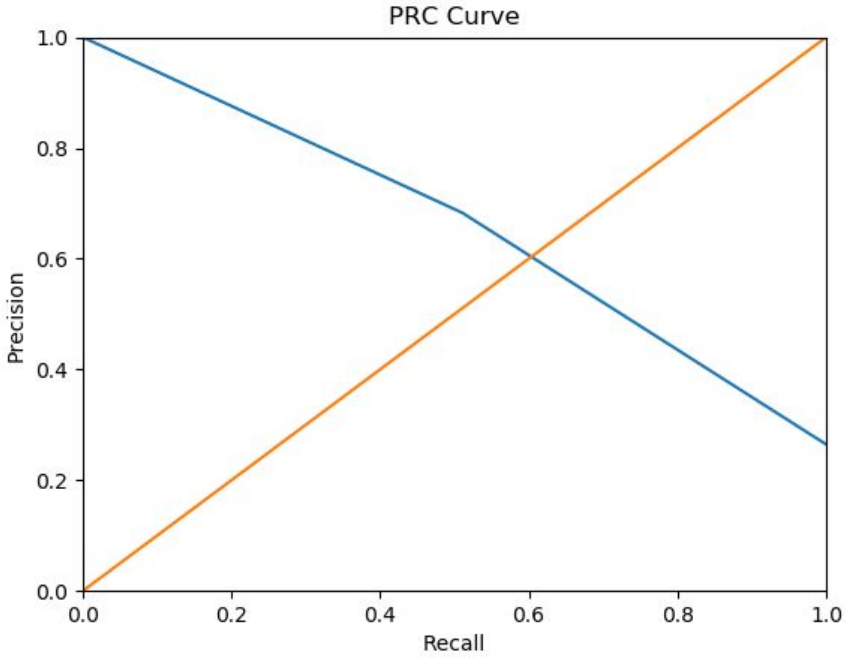
Following graph was obtained from feature selection. Most important 3 features give 78% accuracy as can be seen from the plot below. Accuracy, confusion matrix, PRC and ROC curves were calculated using all of the features since aim was finding best accuracy and 19 features were not computationally expensive to use.



Best 3 features: tenure, InternetService, Contract
Best 5 features: tenure, InternetService, OnlineSecurity, Contract, TotalCharges
Best 7 features: tenure, InternetService, OnlineSecurity, TechSupport, StreamingMovies, Contract, TotalCharges
Best 10 features: tenure, InternetService, OnlineSecurity, TechSupport, StreamingTV StreamingMovies, Contract, PaperlessBilling, MonthlyCharges, TotalCharges

Parameter selection was applied in order to find best parameters and following parameters were chosen as best for the classifier.
Best parameters: {'kernel': 'rbf', 'C': 100, 'gamma': 0.001}

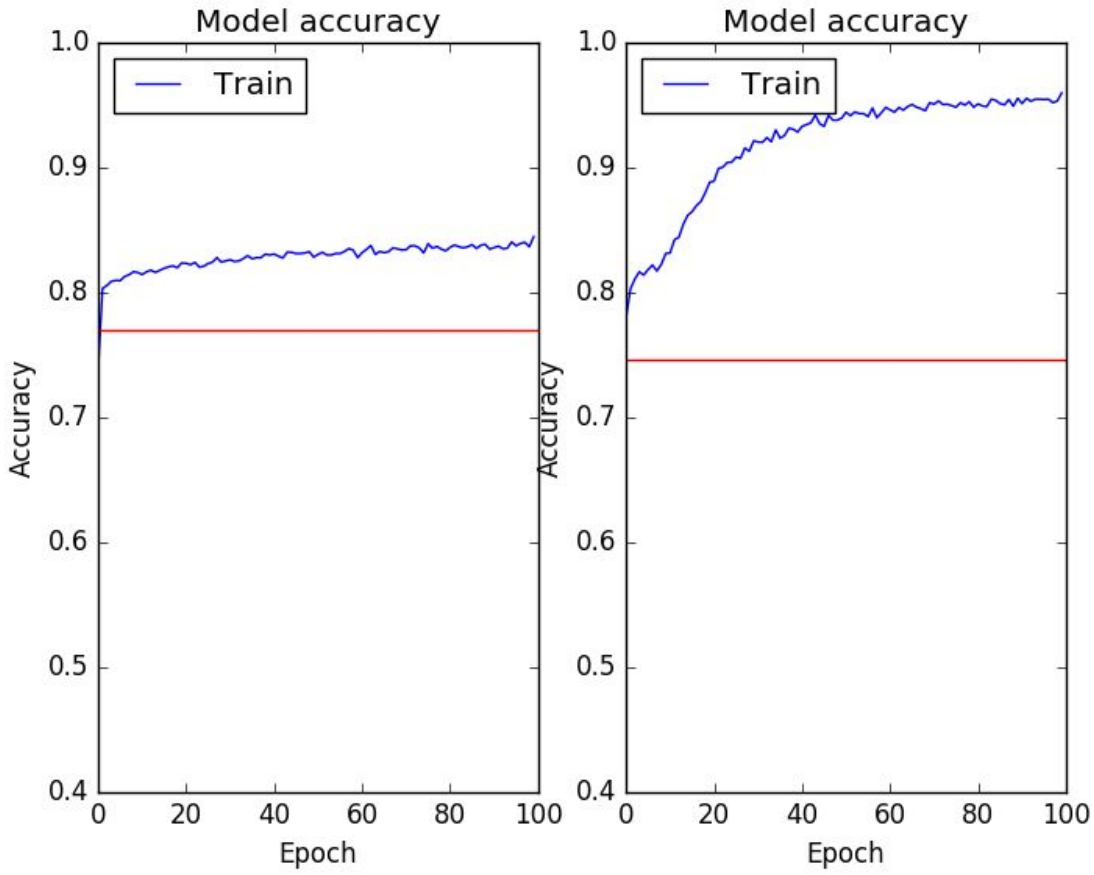

ROC Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0


PRC Curve
Precision
Recall
0.0
0.2
0.4
0.6
0.8
1.0

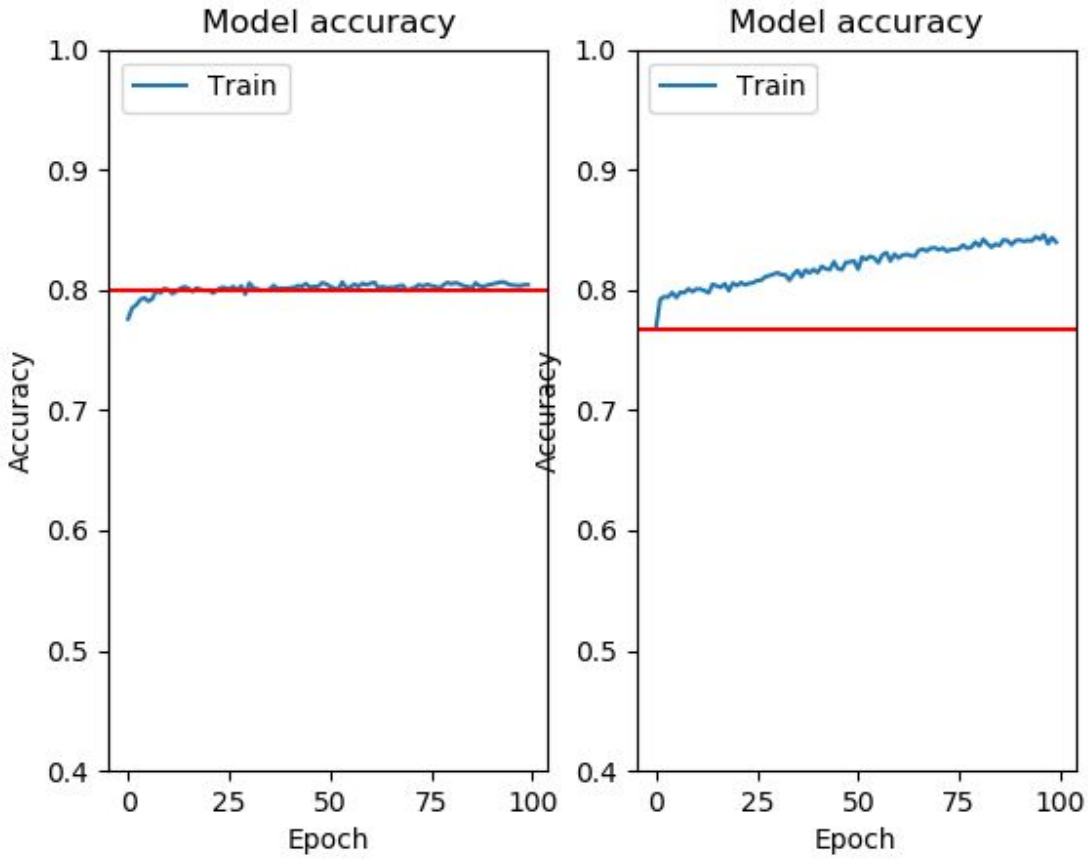
4.2 Neural Network

Following results were obtained from 2 neural networks using 40% of the dataset as the test set. First one is a simple one with three layers with 20, 5 and 1 nodes. Second one is a more complex one with 6 layers of 200, 150, 100, 50, 25, 1 nodes. Purpose of this experiment is finding a network which is fast and does not overfit the data. Blue line is the accuracy on the train set extracted from training process and the red line is the final accuracy obtained on the test set.

Using all 19 features;



Using best 10 features, mentioned in the SVM results;



4.3 Decision Tree

In order to address the first question,we have used the hyper parameters listed above in the section 3.2.3. In our experimentations, we have used 5-fold cross validation to all of the combinations of hyper parameter values given in the following:

- Maximum depth of the tree : [3, None]
- Minimum number of samples required to be at a leaf node : [1, 100]
- Minimum number of samples required to split an internal node: [2, 100]

- Criterion of impurity measurement : Gini Impurity and Entropy
- Splitter strategy: best and random

As for the result of the 5-fold cross validation the optimal hyper parameters we have obtained are the following:

- Maximum Depth: None
- Minimum number of samples required to be at a leaf node: 40⋯⋯
- Minimum number of samples required to split an internal node: 68
- Criterion of impurity measurement: Gini Index
- Splitter Strategy: Best
- Which resulted with accuracy rates of %79 on 5-fold cross validation and %79 on test set.

The following experiments is done in order to address the questions 2-5. We have used the best hyperparameters obtained from the hyper parameter experiment. In each train/test ratio value, feature selection experiment is conducted and we extracted the relationship between the selected feature amount and accuracy. Then, by using the best feature set with highest accuracy, we have calculated the precision, recall, F1-Score, and support values are obtained from the resulting feature set and only the values obtained from negative class labels were used.

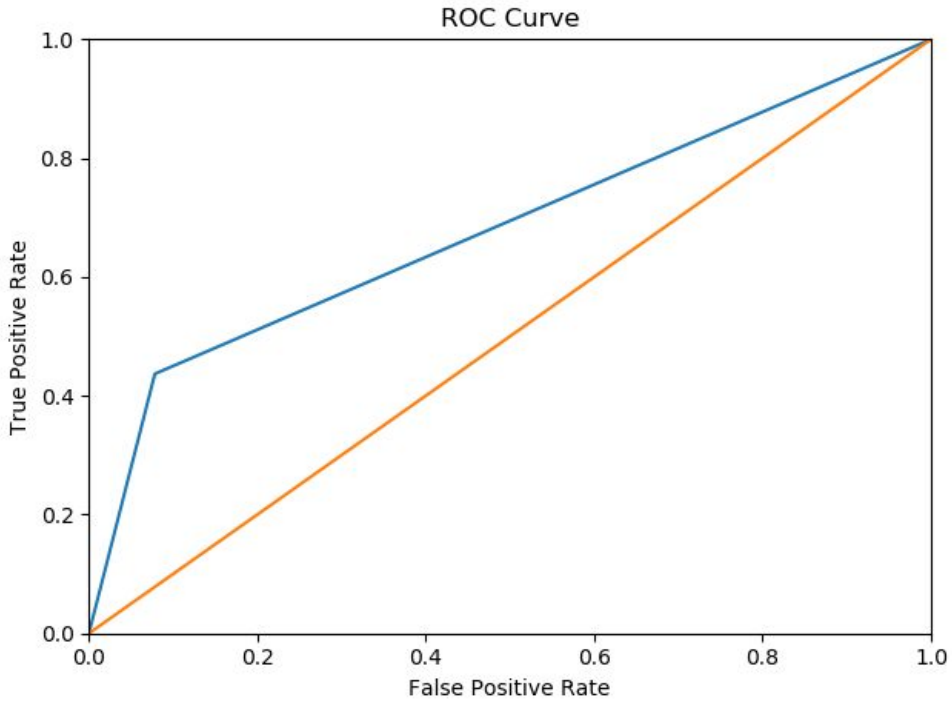
Train/Test Ratio	Accuracy	Precision	Recall	F1-Score	Support
2/8	79.1	0.81	0.93	0.87	4137
4/6	78.2	0.83	0.89	0.86	3083
6/4	79.2	0.84	0.88	0.86	2088
8/2	79.1	0.82	0.91	0.86	1028

As it can be seen, the accuracy score is maximum when the train/test ratio is 6/4. The feature set which gives this score is ['Contract', ‘StreamingMovies’, 'TechSupport', 'InternetService' , ‘MultipleLines’, ‘Dependents’]. The Confusion Matrix obtained with using optimal hyper parameters using the best features and best test/train ratio is the following:

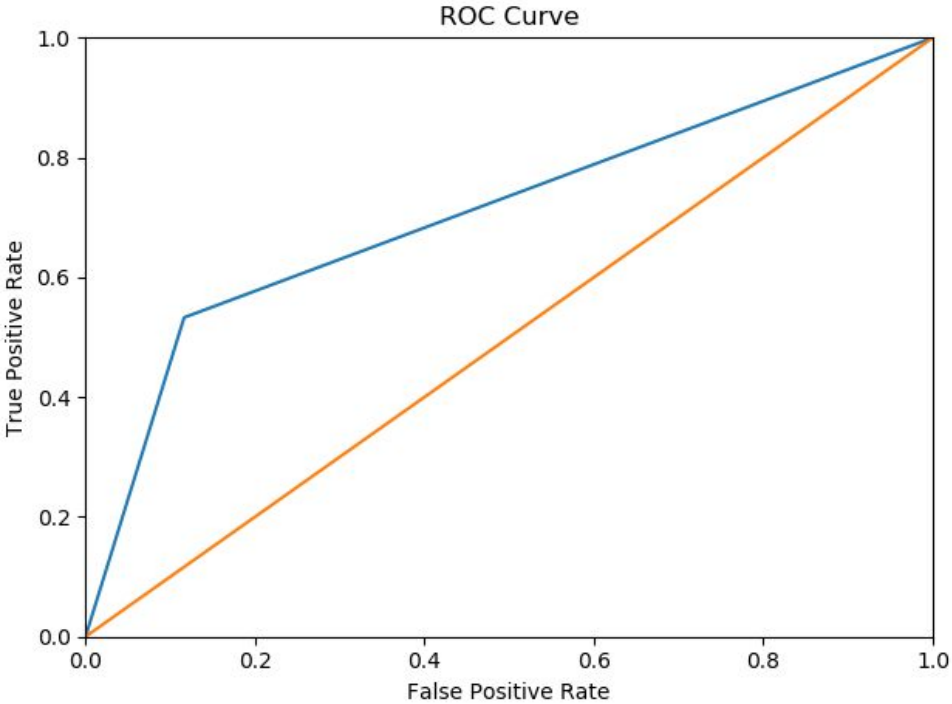
Actual \ Predicted	0	1
0	1903	165
1	419	331

Also, the minimum accuracy score is 4/6 which is obtained by the features ['PaymentMethod', 'StreamingMovies', 'DeviceProtection', 'InternetService', 'MultipleLines',

'Dependents']. It is important to note that this accuracy is the minimum among the best accuracies of the given ratios.The following graphs are obtained by these two ratio tests:

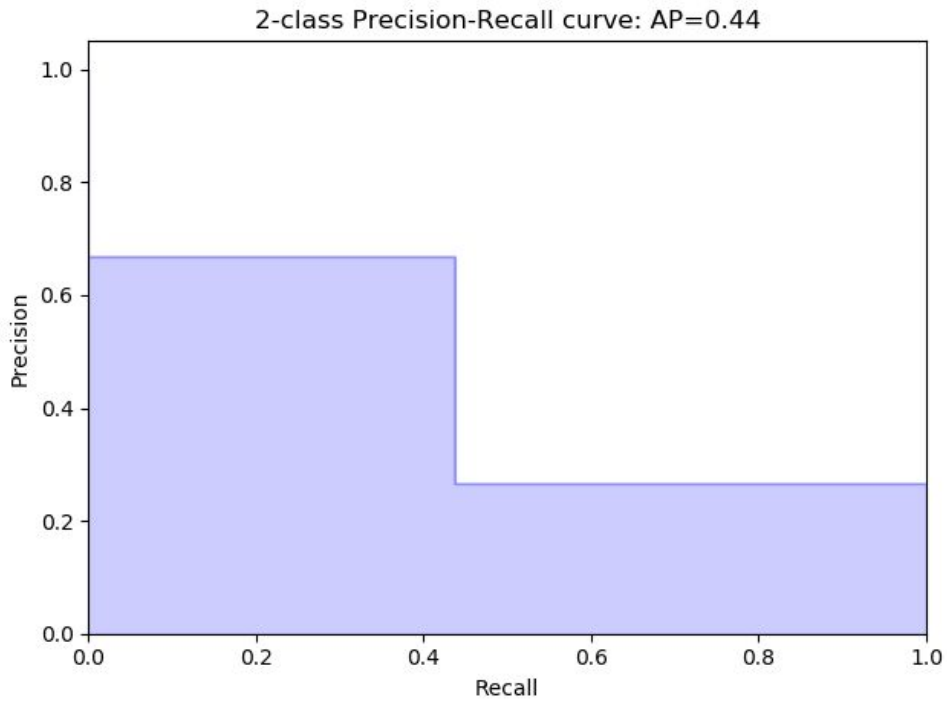


ROC curve for Ratio 4/6

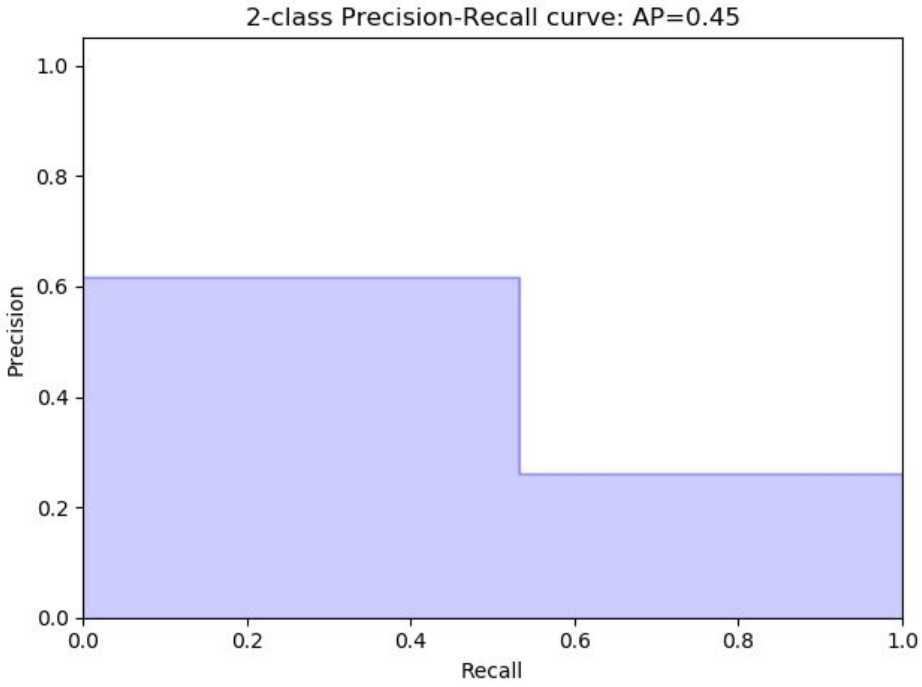


ROC Curve for ratio 6/4

As it can be seen, the ROC curves do not differ, which validates the ROC values given in the table. Therefore, it can be said that, the selection of the different features, and using different train/test ratios that are similar to each other for our dataset, does not alter the relationship between the true positive rate and the false positive rate.

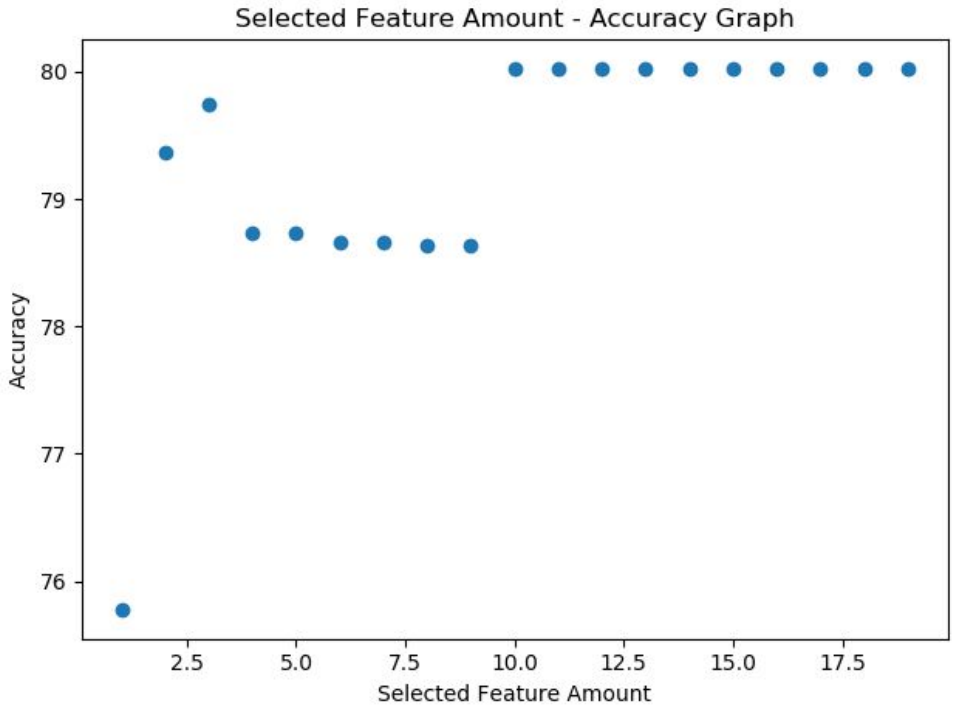


Precision-Recall Curve for ratio 4/6

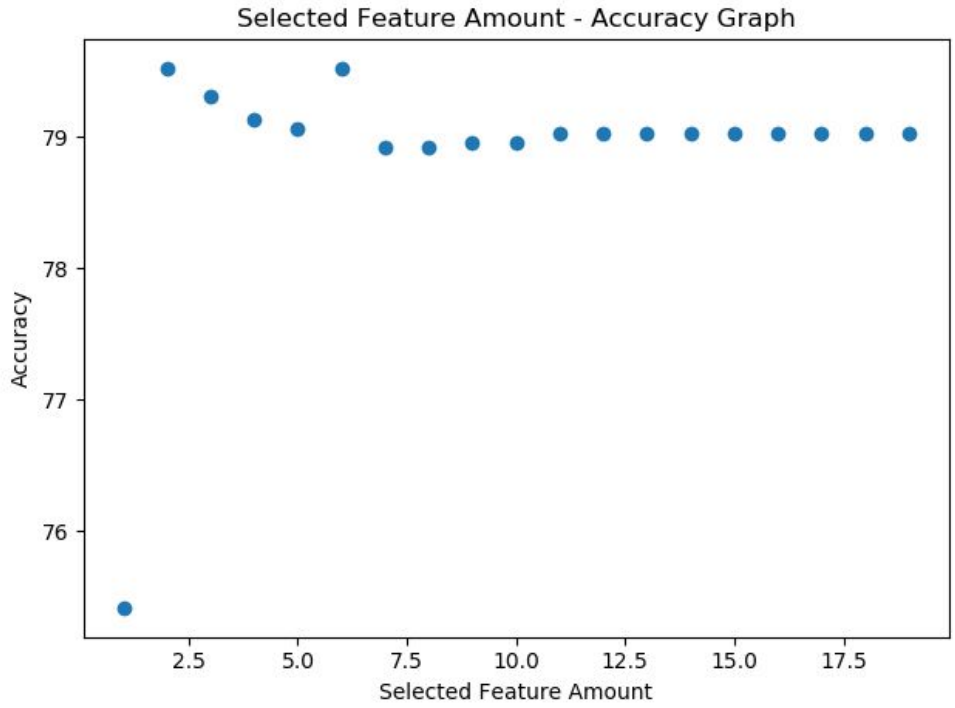


Precision-Recall Curve for ratio 6/4

The precision-recall curves do differ from each other. As it can be seen once we use the ratio 6/4 the recall values corresponding to precision value 0.6 are larger. However, the areas under the curves are somewhat similar so in terms of overall relationship between the precision and recall, the change is small.



Selected Feature Amount - Accuracy for ratio 4/6



Selected Feature Amount - Accuracy for ratio 6/4

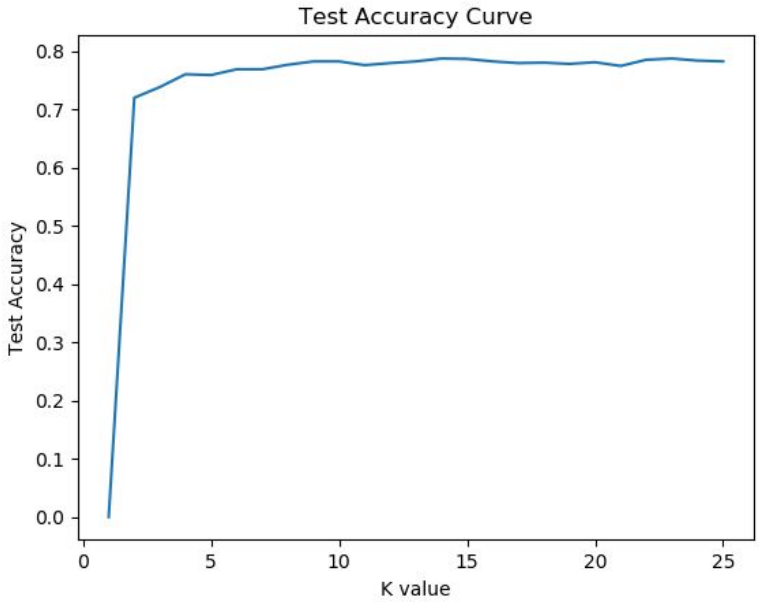
As it can be seen from the selected feature amount - accuracy graphs, feature selection does help increasing the accuracy. However, this increase in the accuracy is low. Therefore, the model can be applied without selecting features from the dataset.

4.4 KNN Classifier

We used the following hyperparameters for KNN classifier:

- Number of closest neighbors K = 4
- Uniform weights assigned to samples (default case)
- Automatic selection of algorithm for computing nearest neighbors (default case)
- Distance metric Minkowski with p = 2, equivalent to Euclidean L2 distance (distance)

To verify selection of hyperparameter K, we experimented with value of K in range [1, 25] and chose the value resulted in highest test accuracy. The graph showing test accuracy with respect to K is as follows:



After K = 4, we observed that test accuracy fluctuates in a small range, hardly improving. This is a sign which shows as we increase value of K, noise in class assignments increase as well. Hence, we chose K = 4 as optimum value.

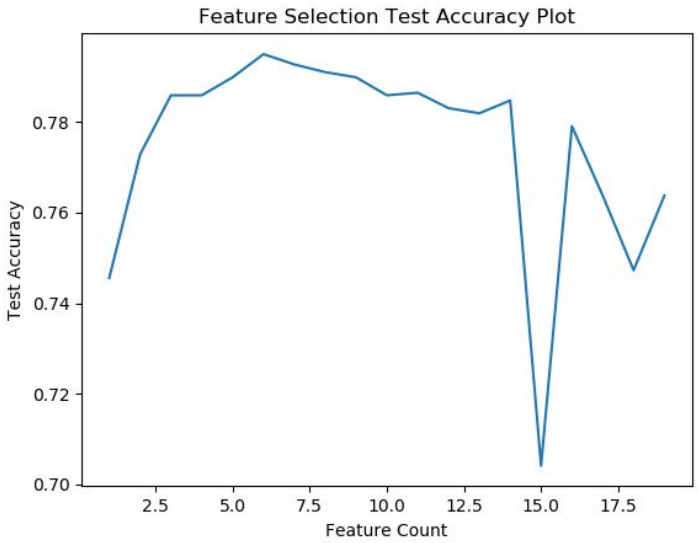
Next, we experimented on the train / test data ratio using hyperparameter K = 4, and obtained the following test accuracies:

Test Data / Entire Dataset	Test Accuracy
0.05	0.779

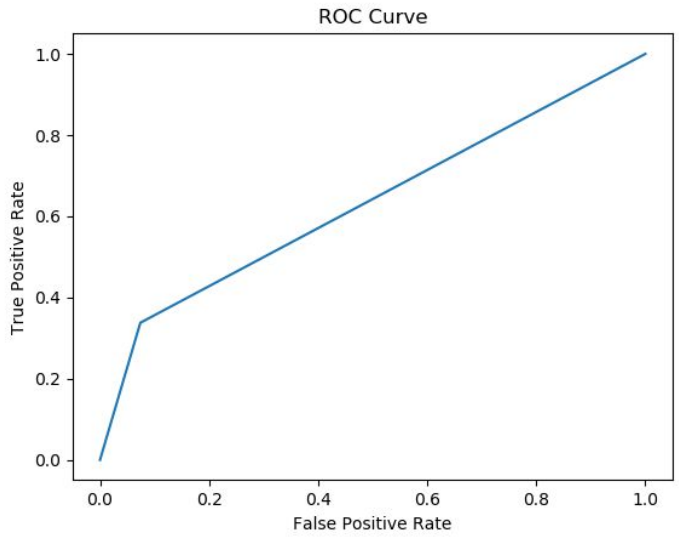
0.1	0.770
0.15	0.759
0.20	0.755
0.25	0.792
0.30	0.778
0.35	0.759
0.40	0.771
0.45	0.754
0.50	0.760

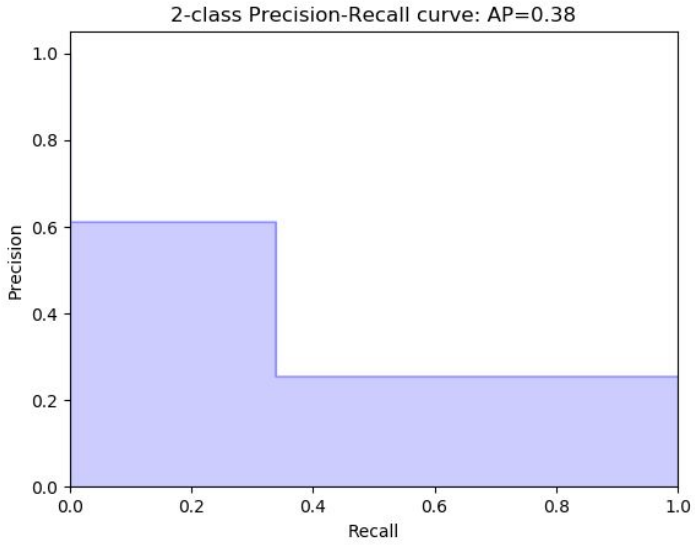
The highest accuracy is obtained at ratio 0.25 when combined with hyperparameter K = 4, as shown in the table above. Note that we randomly select train and test samples for each experiment, hence the results are dependable.

Next, we applied feature selection by selecting the n-best features using sklearn.SelectKBest function, which takes 2 parameters: a statistical measuring method such as Mutual Information of chi-squared test, and number of features desired, in this case, n. We experimented with every combination of features in the dataset and obtained the following test accuracies, shown in a graph:



From the graph, we observe that selecting first 6 best features yield in the best test accuracy, and these features are as follows: **['tenure', 'OnlineSecurity', 'TechSupport', 'Contract', 'MonthlyCharges', 'TotalCharges'].**
Using the parameters described above, we obtained the following ROC and Precision - Recall Curves:



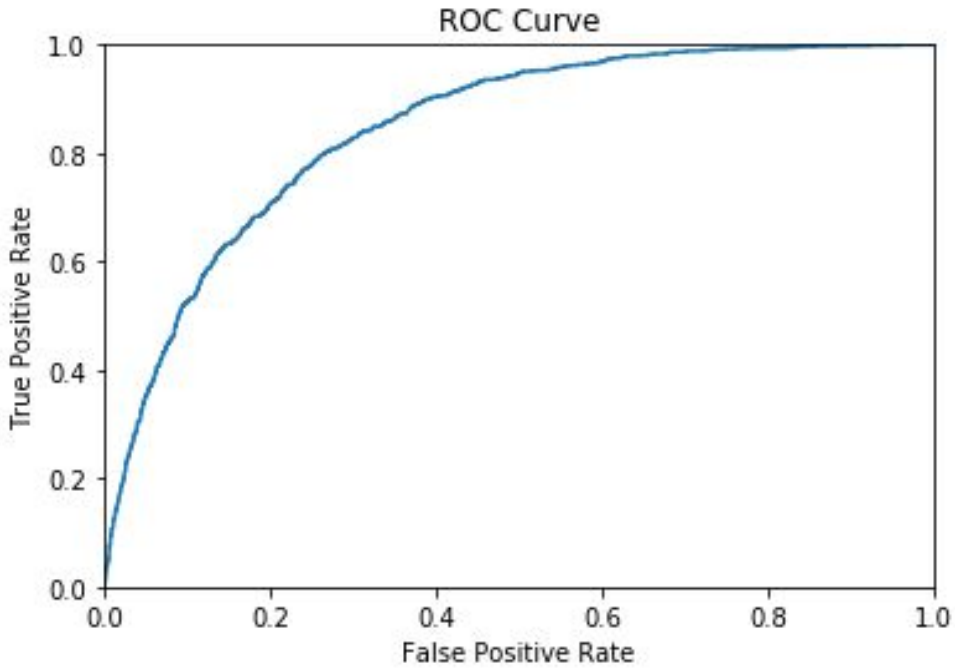


4.5 Logistic Regression

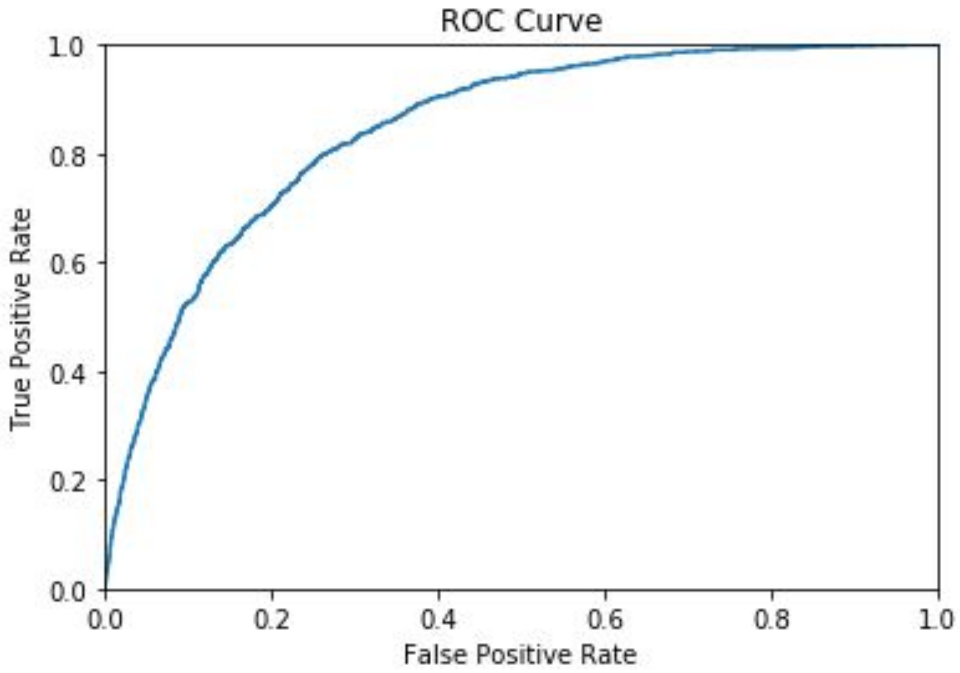
4.5.1 Adjusting Parameters

As mentioned in section 3.2.5, the parameters affecting the performance were penalty,fit_intercept, dual, tolerance, solver and multi_class. The later should be set to ‘ovr’

since it is being dealt with a binary classification. ‘Liblinear’ was chosen for solver as it is more suitable for the size of our dataset while ‘sag’ and ‘saga’ are used for large ones. Dual was chosen to be False, as the number of samples > number of features. Experiments were conducted to determine best values for fit_intercept, tolerance and penalty.
When experimenting with penalty and setting it to firstly ‘l1’ then ‘l2’ keeping the other parameters constant the following was obtained for ‘l1’:
Confusion matrix [[3178 434] [597 722]]
Accuracy:79.0914621780572%

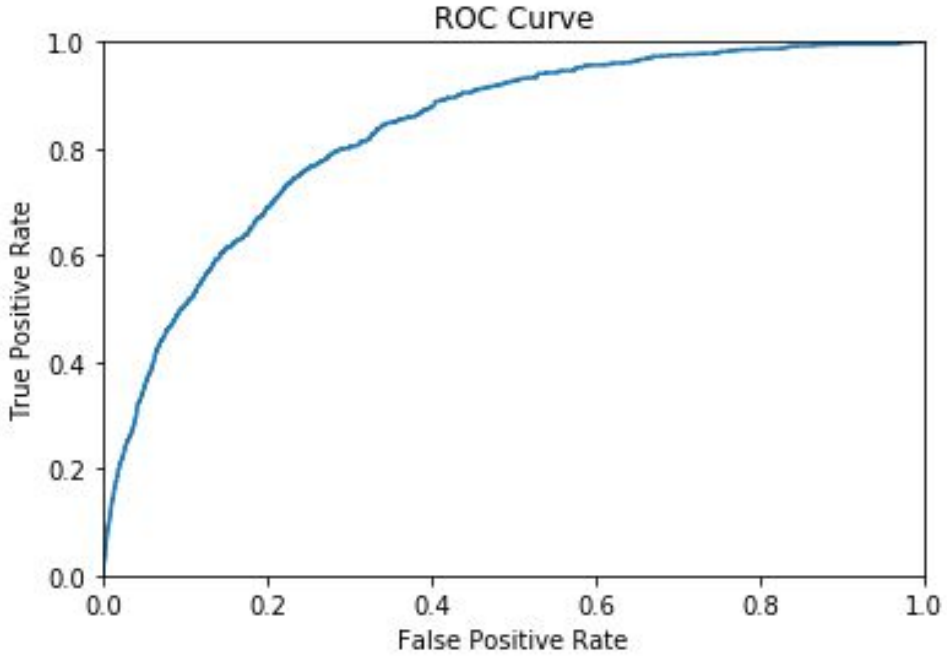


When experimenting for ‘l2’:
Confusion matrix: [[3179 433] [597 722]]
Accuracy: 79.11174204015413%



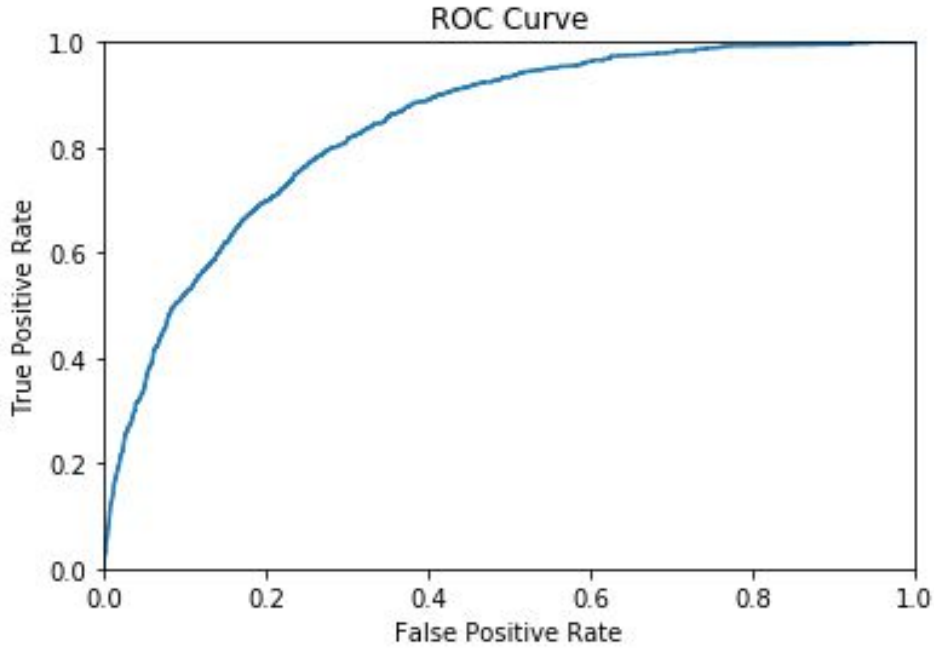
It can be observed that there is a slight accuracy of 1% and tpr over fpr improvement when setting penalty to ‘l2’.

When no constant was added and fit_intercept = False the following was obtained:
Confusion matrix: [[2364 1236] [207 1124]]
Accuracy:70.73615899411884%



However, the results were improved significantly when constant was added:
Confusion matrix: [[3218 382] [624 707]]

Accuracy:79.59845873048064%



The accuracy was increased with approximately 9.2% as well as tpr over fpr rate has a slight improvement considering the ROC curves.
Regarding tolerance, the default of 10 ^-4 value resulted better after trying several values:

Tol values	Accuracy	Precision	Recall
1e-2	0.795	0.79	0.80
1e-4	0.804	0.79	0.80
1e-8	0.797	0.79	0.80
1e-6	0.799	0.79	0.80

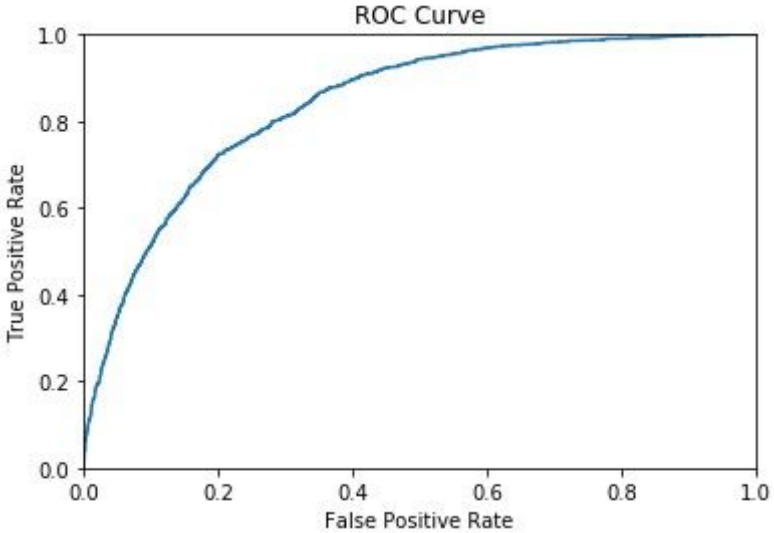
4.5.2 Adjusting test and train set sizes

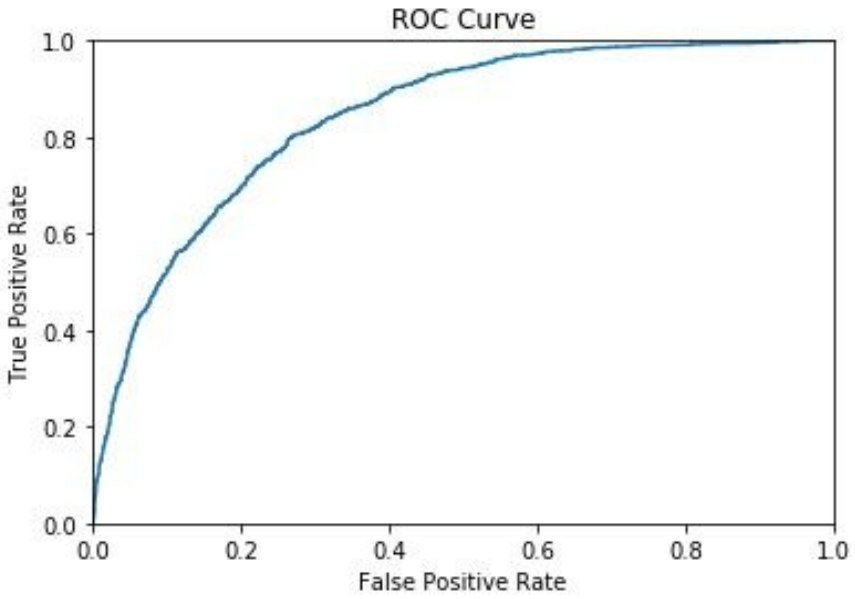
Train/Test ratio	Accuracy	Recall	F1-score	Support
2/8	0.792	0.80	0.79	5635
4/6	0.801	0.80	0.80	4226
6/4	0.795	0.80	0.79	2818
8/2	0.814	0.81	0.80	1409

Table - Measurements for different train and test set sizes

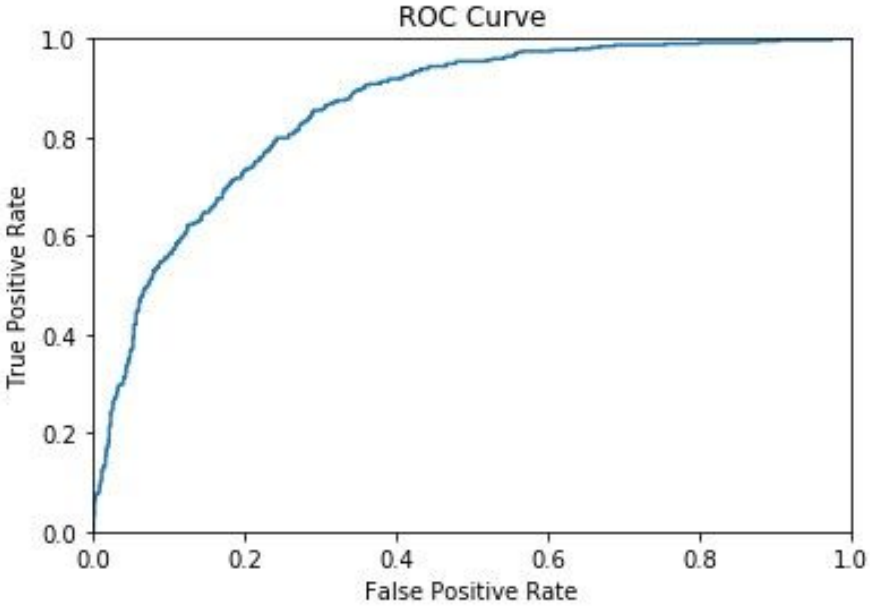
Accuracy matrix for ratio 2/8: [[3615 558] [615 847]]
Accuracy matrix for ratio 4/6: [[2776 340] [504 606]]
Accuracy matrix for ratio 4/6: [[1880 178] [349 411]]
Accuracy matrix for ratio 8/2: [[935 82] [185 207]]

For Ratio 2/8

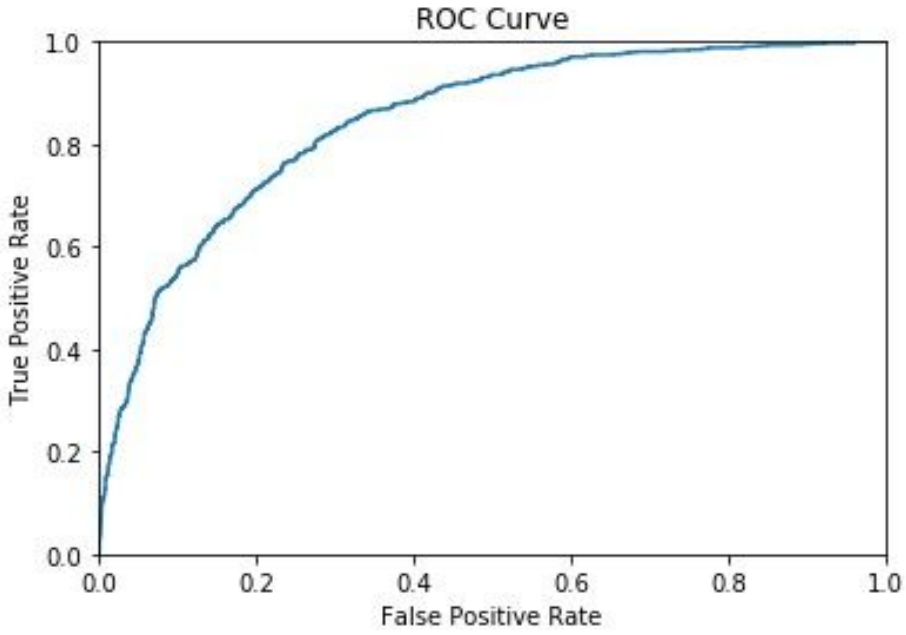




ROC for ratio 4/6



ROC for ratio 6/4



ROC for ratio 8/2

Initially the dataset was shuffled, otherwise it gave an accuracy of approximately 63% with logistic regression. After shuffling the dataset, 4 trials were conducted with different chunks of data for train and test sets. It is observed that although splitting of dataset was made in proportions significantly different from each other, the range of accuracy was between 79%-81%, with the lowest at 79% when train set contained 20% of the whole dataset. ROC curves for different cases do not differ substantially from each other and it easily distinguishable that tpr is higher than fpr. However there is a variation mainly on the part of the plot corresponding to x axis between 0 - 0.2, in which as the size of test set increases, so seems to do the tpr compared to fpr.

4.5.3 Feature selection

Since there were 19 features in total, it is worth considering that not all of them have equal importance in predicting churn. Thus, using SelectKBest with chi score function and choosing to distinguish 5 most important features gave the following results:

Train/Test ratio	Accuracy	Recall	F1-score	Support	Accuracy difference from using all features
4/6	0.786	0.79	0.78	4226	-0.015
6/4	0.787	0.79	0.78	2818	-0.008
8/2	0.795	0.79	0.79	1409	-0.06

Table for top 5 features

From the last column of the table it can be deduced that the top 5 features chosen yield results very similar to those recorded while using all 19 features. The top 5 features are respectively: tenure, online security, contract, monthly charge, total charge. Due to this miniscule difference in accuracy it is not necessary to increase the number of features aiming for better accuracy. Thus, the number of features was reduced further to 3 to observe how scores change:

Train/Test ratio	Accuracy	Recall	F1-score	Support	Accuracy difference from using all features
4/6	0.786	0.79	0.77	4226	-0.015
6/4	0.785	0.79	0.77	2818	-0.01
8/2	0.790	0.79	0.78	1409	-0.065

Table for top 3 features

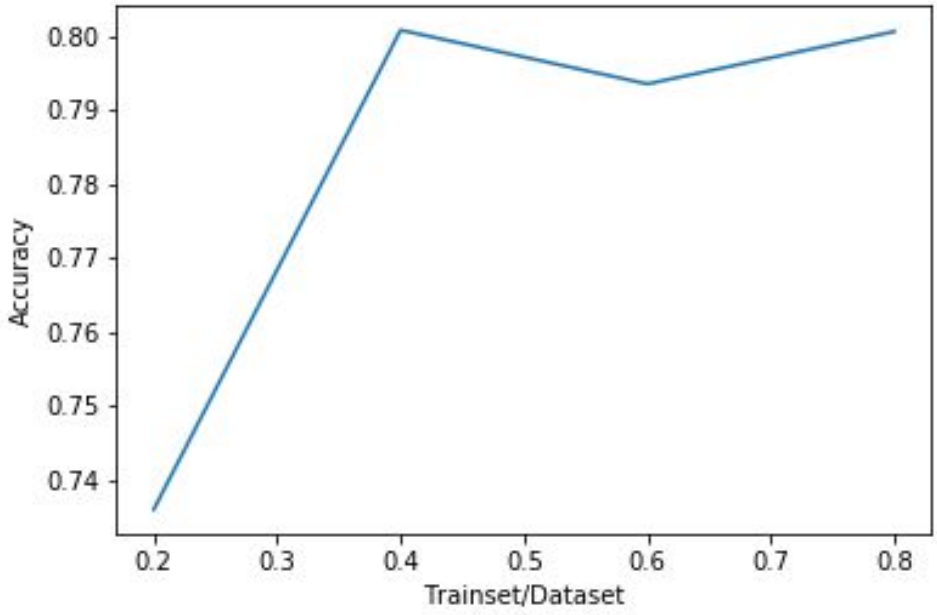
The 3 dominant features in this stage were tenure, monthly charge and total charge. From the obtained results it can be observed that the difference between using 5 and 3 top features is relatively small and seems that online security and contract features do not have fundamental contribution to the results as the accuracy does not vary significantly after discarding those features.

4.5.4 Cross Validation

Using LogisticRegressionCV with different number of fold values affected the model and accuracy. Initially cv = 4 was tried with different train/dataset size ratios and the following results were obtained:

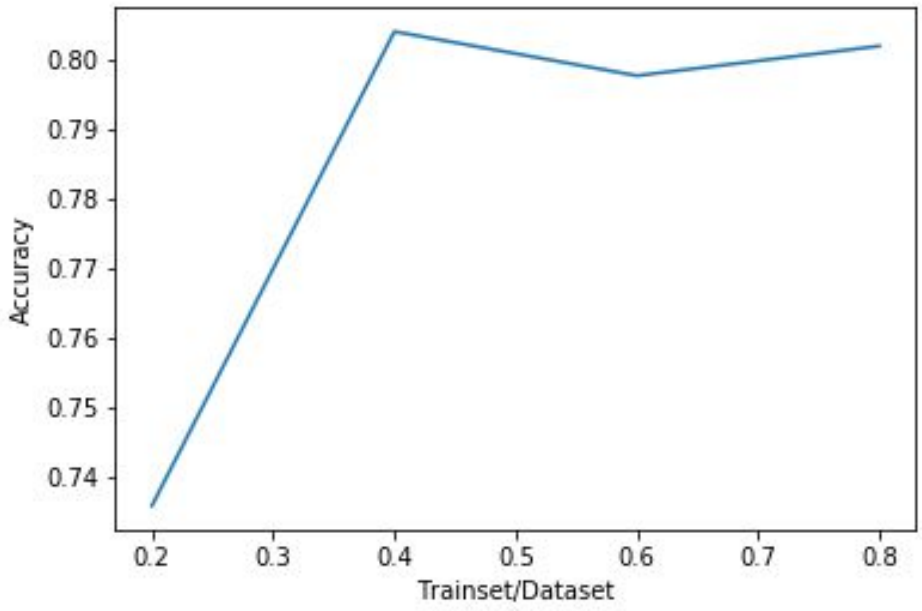
Table-Accuracy results for CV =4

Train/Test ratio	Accuracy	Recall	F1-score	Support	Accuracy difference w/o cross validation
2/8	0.736	0.34	0.62	5635	-0.056
4/6	0.801	0.80	0.80	4226	0
6/4	0.793	0.79	0.79	2818	-0.002
8/2	0.801	0.80	0.79	1409	0.054



Accuracy results for CV =4

While separating the train set into 10 folds, no significant difference in results were obtained. The following plot contains the accuracies obtained from the same process as that conducted formerly, but with parameter cv = 10:

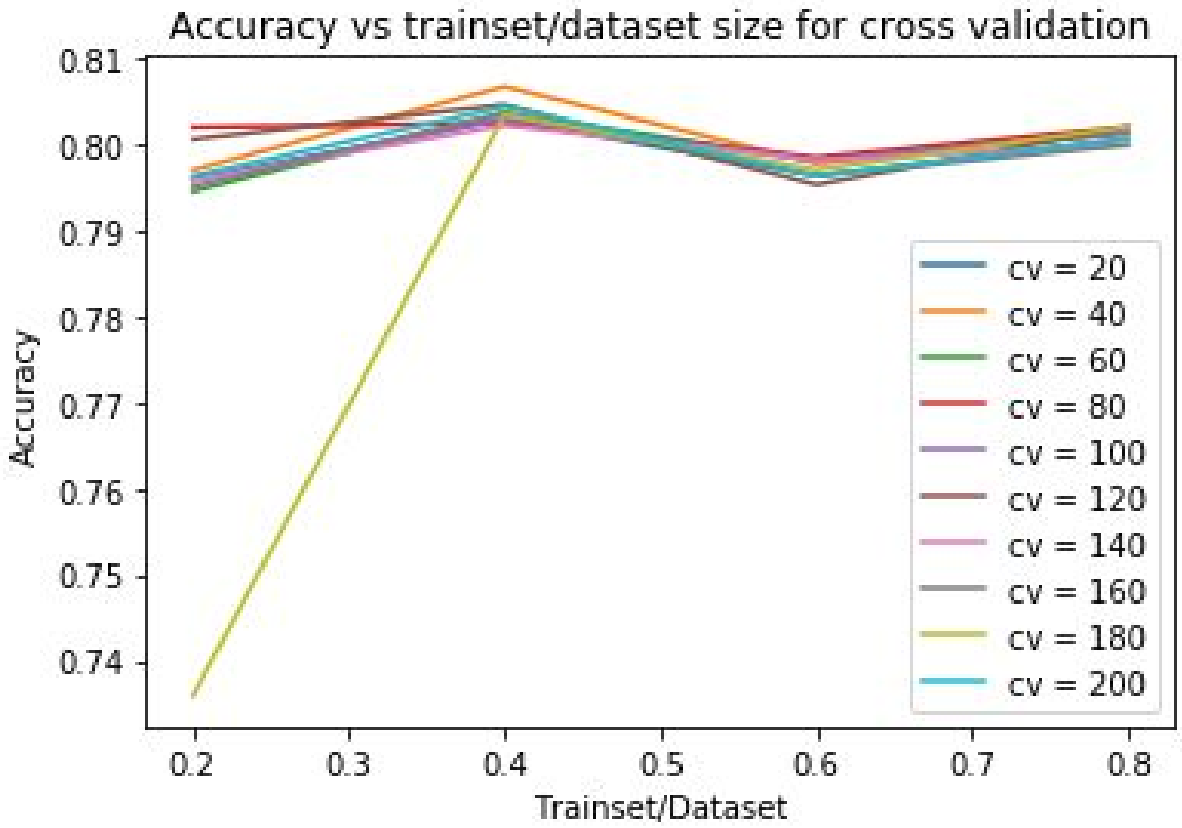


Accuracy results for CV =10

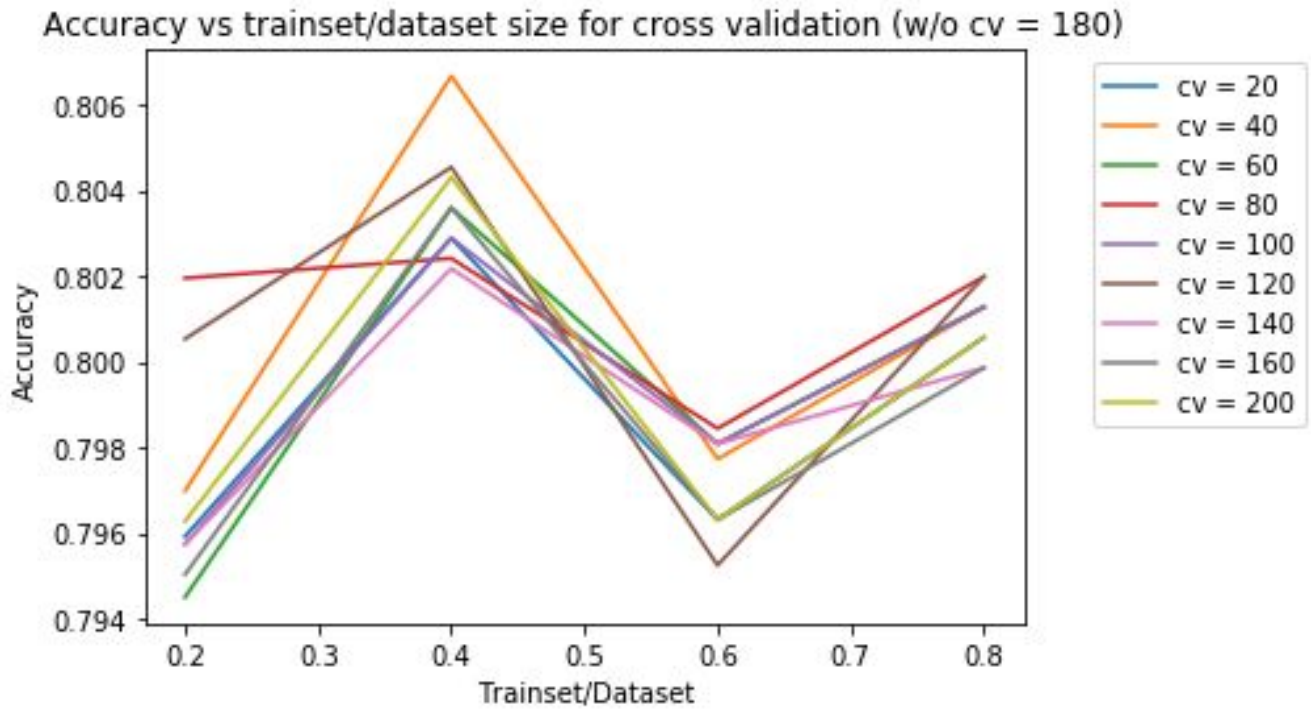
However, conducting several trials it was observed that, as the number of folds increased, the better was the accuracy obtained. This can be attributed to the fact that, if the number of folds is small then each of them will have a big chunk of data, thus leading to a smaller amount of data for training and a big one for validation. For instance in the case of having 4 folds: 5634/4 = 1408 rows would belong to the validation fold on each iteration and there would be less data for training. However, as the number of folds is increased, say 100, on which the accuracy is significantly higher (see table below), the number of rows in the validation fold is much lower 5634/100 = 56 and there are better chances to obtain a better performing model.

Ratio ---------- Nr Folds	Train/Test 2/8	Train/Test 4/6	Train/Test 6/4	Train/Test 8/2
20	0.795918	0.802887	0.796309	0.800568
40	0.796983	0.806673	0.797729	0.801278
60	0.794499	0.803597	0.798084	0.801278
80	0.801952	0.802414	0.798439	0.801987
100	0.795741	0.802887	0.798084	0.801278
120	0.800532	0.804543	0.795245	0.801987
140	0.795741	0.802177	0.798084	0.799858
160	0.795031	0.803597	0.796309	0.799858
180	0.735936	0.80336	0.797019	0.801987
200	0.796273	0.804307	0.796309	0.800568

Table- Accuracies for different number of folds and train and test set sizes



From the data and the plot it can be observed that the best accuracy is obtained when the train set contains about 40% of the whole dataset and the nr of folds = 40. It's observed that increasing the size of trainset beyond 40% does not yield better accuracy, as it did in the very first case with no cross validation. Considering the above discussion about the reasons why increasing the number of folds increases the accuracy and the recent results, we can deduce that the former reasoning does not always hold. For instance, the performance does increase as cv increases gradually for most of the cases, however when cv = 180 a strange result is observed where the accuracy is approximately 0.73 which is similar to the case above when cv = 4. Furthermore, the best performance does not correspond to the case with the largest number of folds. If we discard the penultimate case where cv = 180 and re-plot the graph to distinguish the accuracies relationship of those cases which start above 0.79, then we get the following:



The results become clearer in this graph and support the above claim that increasing the nr of folds improves the performance generally, with some exceptions. However, considering the accuracy presented in the y axis, it can be observed that the change between all cases occurs in the second decimal digit, thus meaning around 1%. This is not an essential change in performance as to completely dispute the argument made related to the number of folds. It should also be considered that there might be biased portion of data while training or validating which also affect the parameters and lead to over or under fitting.

4.6 Random Forest Classifier

The random forest classification is not a consistent method that gives the same exact accuracy output in each experiment. The reason behind that is, the algorithm designs different decision trees in each experiment and they may not be exactly the same, resulting in a very small precision difference in different runs with the same parameters.

In the experiments following hyper parameter values were used.

- **Number of Decision Trees:** [1, 5, 10, 100]
- **Maximum Leaf Nodes:** [10, 100, 1000, None]
- **Max Depth:** [1, 10, 100, None]
- **Min Samples Required In Leaf:** [1, 2, 5, 10]

Following figure shows the combinations of different hyper parameters and the selection process:

```
Acc:  0.7923239232581283  ~ Tr#:  10  ~ MaxLN:  None  ~ MaxD:  10  ~ MinSR:  10
Acc:  0.799069070203948  ~ Tr#:  10  ~ MaxLN:  None  ~ MaxD:  100  ~ MinSR:  1
Acc:  0.7909080029729287  ~ Tr#:  10  ~ MaxLN:  None  ~ MaxD:  100  ~ MinSR:  2
Acc:  0.7933978307698122  ~ Tr#:  10  ~ MaxLN:  None  ~ MaxD:  100  ~ MinSR:  5
Acc:  0.7962290414824332  ~ Tr#:  10  ~ MaxLN:  None  ~ MaxD:  100  ~ MinSR:  10
Acc:  0.78913117418087  ~ Tr#:  10  ~ MaxLN:  None  ~ MaxD:  None  ~ MinSR:  1
Acc:  0.789131804038648  ~ Tr#:  10  ~ MaxLN:  None  ~ MaxD:  None  ~ MinSR:  2
Acc:  0.8011992492095285  ~ Tr#:  10  ~ MaxLN:  None  ~ MaxD:  None  ~ MinSR:  5
Acc:  0.7923232934003502  ~ Tr#:  10  ~ MaxLN:  None  ~ MaxD:  None  ~ MinSR:  10
Acc:  0.755401660305103  ~ Tr#:  100  ~ MaxLN:  10  ~ MaxD:  1  ~ MinSR:  1
Acc:  0.7575274303062368  ~ Tr#:  100  ~ MaxLN:  10  ~ MaxD:  1  ~ MinSR:  2
Acc:  0.7539807011576787  ~ Tr#:  100  ~ MaxLN:  10  ~ MaxD:  1  ~ MinSR:  5
Acc:  0.7461742438557374  ~ Tr#:  100  ~ MaxLN:  10  ~ MaxD:  1  ~ MinSR:  10
Acc:  0.7944534724059308  ~ Tr#:  100  ~ MaxLN:  10  ~ MaxD:  10  ~ MinSR:  1
Acc:  0.7909010745373696  ~ Tr#:  100  ~ MaxLN:  10  ~ MaxD:  10  ~ MinSR:  2
Acc:  0.792327072547019  ~ Tr#:  100  ~ MaxLN:  10  ~ MaxD:  10  ~ MinSR:  5
```

After the best hyperparameter selection with 5-fold cross validation, hyperparameters obtained are as follows with the accuracy 0.7955:

- **Number of Decision Trees:** 100
- **Maximum Leaf Nodes:** 100
- **Max Depth:** 10
- **Min Samples Required In Leaf:** 5

Maximum Train/Test split ratio obtained: 0.4

Train/Test Ratio	Accuracy
0.1	0.798
0.2	0.787
0.3	0.792
0.4	0.803
0.5	0.797
0.6	0.801
0.7	0.801
0.8	0.802
0.9	0.789

While deciding on the importance of the features, **mean decrease accuracy** method was used. This method's main idea is to permute the values of each feature and calculate the decrease in

the accuracy. For unimportant variables, permutation has a little effect and for the important variables, the effect is expected to be much higher.

The feature importance list after the model training with the best hyper parameters is given below:

```
[[180.55658198613608, 'tenure'],
 [120.371054657424, 'Contract'],
 [53.67898383371615, 'MonthlyCharges'],
 [37.41262509622634, 'OnlineSecurity'],
 [34.159353348728374, 'InternetService'],
 [24.39953810623464, 'TechSupport'],
 [17.892994611238716, 'SeniorCitizen'],
 [16.26635873748981, 'PaymentMethod'],
 [13.01308698999185, 'PaperlessBilling'],
 [13.013086989991773, 'StreamingMovies'],
 [11.386451116242792, 'MultipleLines'],
 [8.13317936874483, 'Partner'],
 [8.13317936874483, 'OnlineBackup'],
 [6.506543494995925, 'gender'],
 [4.879907621246943, 'PhoneService'],
 [4.8799076212468675, 'Dependents'],
 [3.2532717474979624, 'StreamingTV'],
 [1.6266358737489812, 'TotalCharges'],
 [1.6266358737489812, 'DeviceProtection']]
```

It is observable that the most important features are ‘tenure’ and ‘contract’. The least important ones are the ‘total charges’ and ‘device protection’.

Impact of features on the output if we don’t only use the magnitude but the value with its sign:

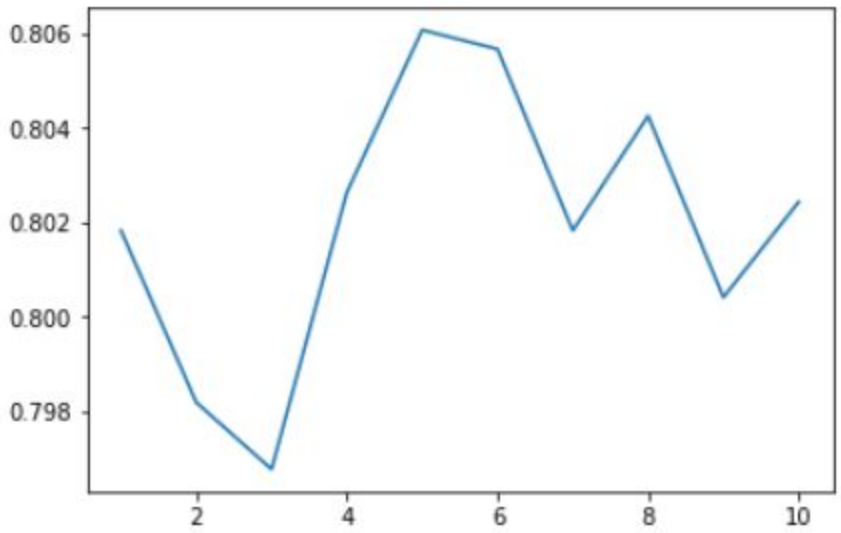
```
[[-14.566561809785105, 'tenure'],
 [-8.34395288133322, 'Contract'],
 [-3.959842045378472, 'OnlineSecurity'],
 [-3.959842045378472, 'TechSupport'],
 [-3.535573254802211, 'MonthlyCharges'],
 [-2.2627668830734144, 'OnlineBackup'],
 [-1.979921022689236, 'InternetService'],
 [-1.4142293019208791, 'SeniorCitizen'],
 [-0.989960511344618, 'Dependents'],
 [-0.8485375811525222, 'DeviceProtection'],
 [-0.8485375811525222, 'MultipleLines'],
 [-0.8485375811525222, 'StreamingMovies'],
 [-0.7071146509604396, 'StreamingTV'],
 [-0.5656917207683436, 'Partner'],
 [-0.4242687905762611, 'PhoneService'],
 [-0.0, 'PaperlessBilling'],
 [-0.0, 'PaymentMethod'],
 [-0.0, 'TotalCharges'],
 [0.1414229301920959, 'gender']]
```

It is observable that most of the features have a negative effect on the output, while ‘tenure’ and ‘contract’ still having the largest dramatic effect.

Accuracy change when we use most kth important features (k = {1..10}):

```
[0.8058823529411765,
 0.7943204868154158,
 0.8054766734279918,
 0.8052738336713997,
 0.8004056795131845,
 0.7977687626774849,
 0.8006085192697769,
 0.8038539553752535,
 0.7977687626774849,
 0.7977687626774849]
```

Accuracy change during the usage of most important n features is plotted below:

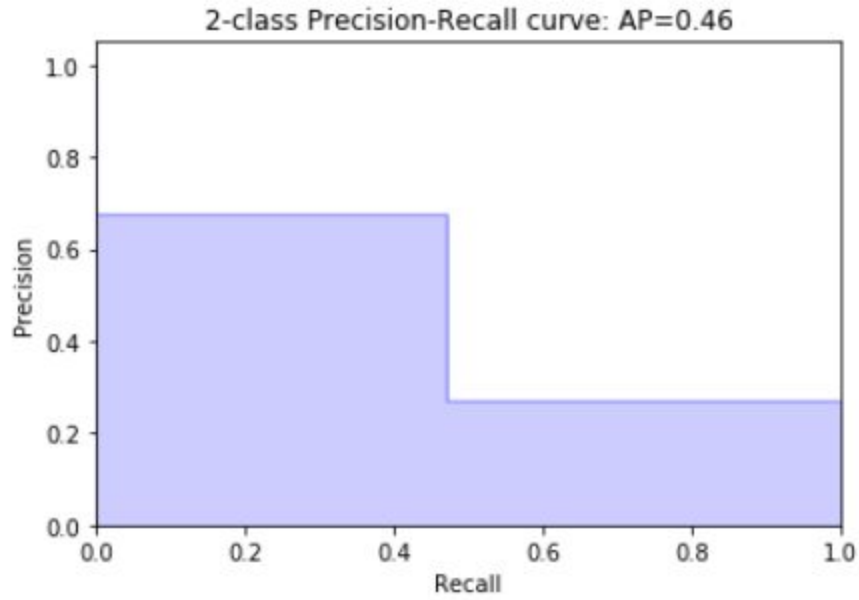


The highest accuracy is obtained with 5 of the most important features.

Confusion Matrix obtained from model with the 5 features and best hyper parameters:

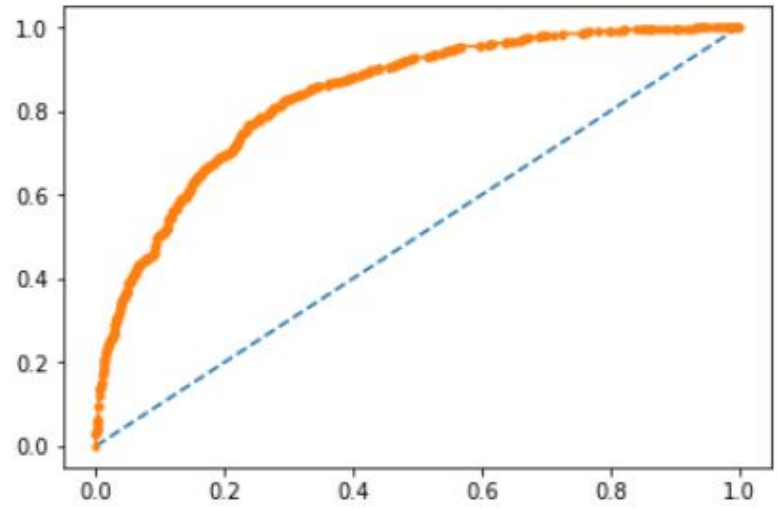
```
[[1418  136]
 [ 267  292]]
```

Precision-Recall Curve for test/train split ratio 0.3 and hyper parameters:

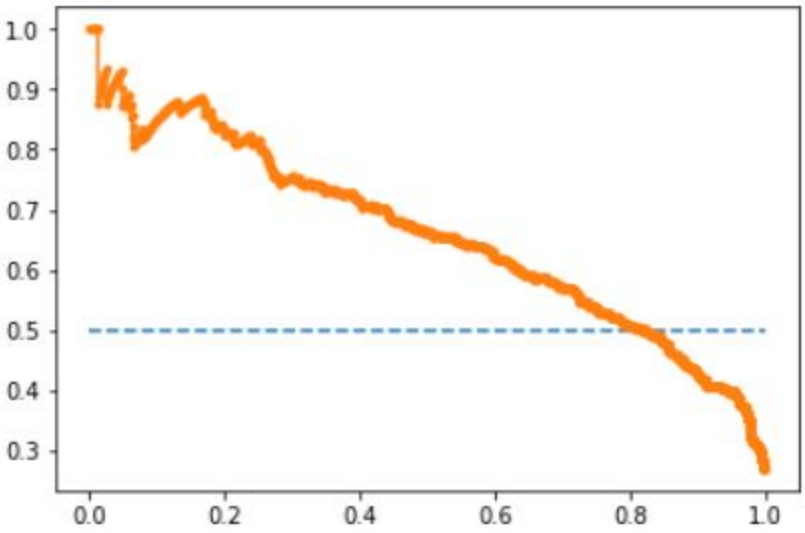


ROC curve and AUC:

AUC: 0.839



Precision-Recall Curve:



5 Discussion

Most of the algorithms give around 70% to 80% accuracy. Thus, the prediction problem can be solved by any of them. This makes the performance the most important metric. Feature selection helps reducing unnecessary features while keeping the accuracy in the mentioned range.

5.1 SVM

Feature selection was not helpful for the accuracy, though it gave information that best 3 features provide sufficient accuracy. All 19 features were used in order to calculate curves and accuracy since it was not expensive. Hyperparameter selection gave rbf kernel as the best choice, which was expected from our dataset.

SVM is a good model considering it's simple enough to train and predict and also it provides sufficient accuracy to solve the problem. If necessary, first three features might be used in order to improve performance while sacrificing negligible accuracy.

5.2 Neural Network

Experiments on the neural networks show the overfitting issue. When using all 19 features, both neural networks overfit since their training accuracies (blue line) are much higher than the test set ones (red line). Also accuracy of the complex is close to 100%, however its accuracy on the test data is even lower than the simple one which also overfitted the data.

The second graph was obtained using feature selection to select best 10 features. The feature selection helped solving the overfitting issue on both models and increased their accuracy on the test set, though the complex network still overfits. Second graph shows that using the simple neural network with feature selection gives the best accuracy. Red line and blue line match closely for that model.

A simple neural network with first 10 features provides sufficient accuracy to solve the problem, however it is not necessary since most of the other methods are similar in terms of accuracy. Also neural network takes more time to train, thus more simpler methods are more preferable.

This method was used in order to support the limitations of the dataset. Because, firstly complex networks give close to 100% while failing at the test set due to overfitting. And secondly the simple one which can be considered as the most general classifier cannot surpass the 80% accuracy on both train and test sets, which is the limit that is reached by the other methods.

5.3 Decision Tree

For the Decision Tree model, feature selection slightly improved the accuracy. Selecting hyper parameters did have some effect, but the default parameters of Sklearn's Decision Tree Classifier was already finding the accuracy rate of %77. However, it is important to note that some of these default parameters were actually the same parameters with the best hyper parameters parameters that our experiments found out. Also, throughout our experiments we have observed that decision tree model is stable in terms of overall performance. It performs similarly in most of the cases we have provided. Since the splitting of the internal nodes eventually find out the similar accuracies regardless of how they were split. The change in the

split may affect the runtime performance of the method but in terms of accuracy, the difference is little.

5.4 KNN Classifier

Related to K, number of closest neighbors, we observed that test accuracy increase rapidly until K = 3. After that, a significant drop in rate of change of accuracy occurred and we observe fluctuations at values greater than 5. This is an indication that a significant amount of noise exists in assignment of samples at high K values.

Next, related to test/train set ratio, we observed that test accuracies we acquired changed from one experiment to another. This is reasonable considering we randomly assign data points to train and test sets for each execution. For example, initial experiment resulted in test accuracy of 0.792 for ratio 0.25 where same setup resulted in test accuracy of 0.76 in another execution.

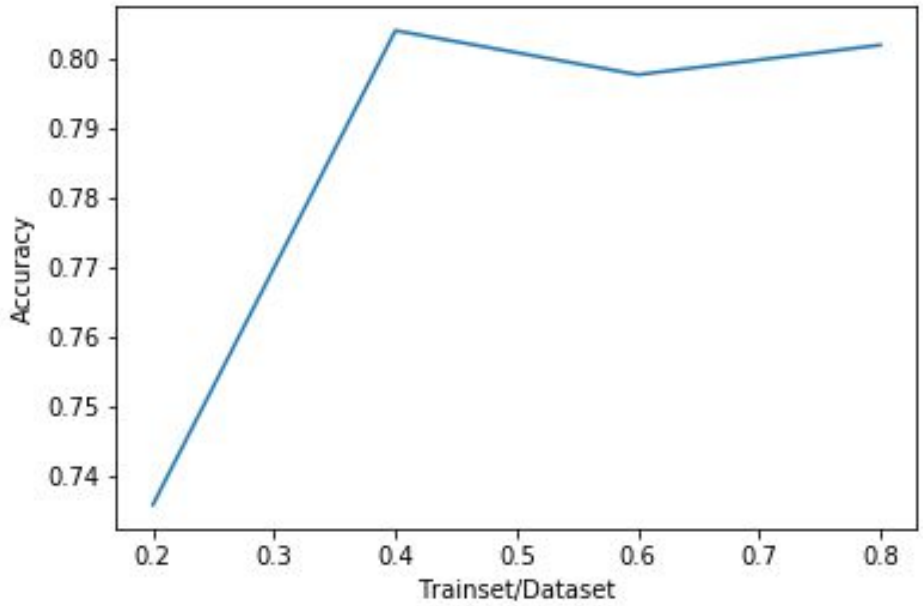
For feature selection, we observed increase in test accuracy until feature count 6. Any value higher than 6 caused loss of test accuracy, specifically between feature count [13, 15]. We suspect that choosing more than 6 best features cause overfitting, which suits the overall decrease in test accuracy as the feature count increases beyond 6.

5.5 Logistic Regression

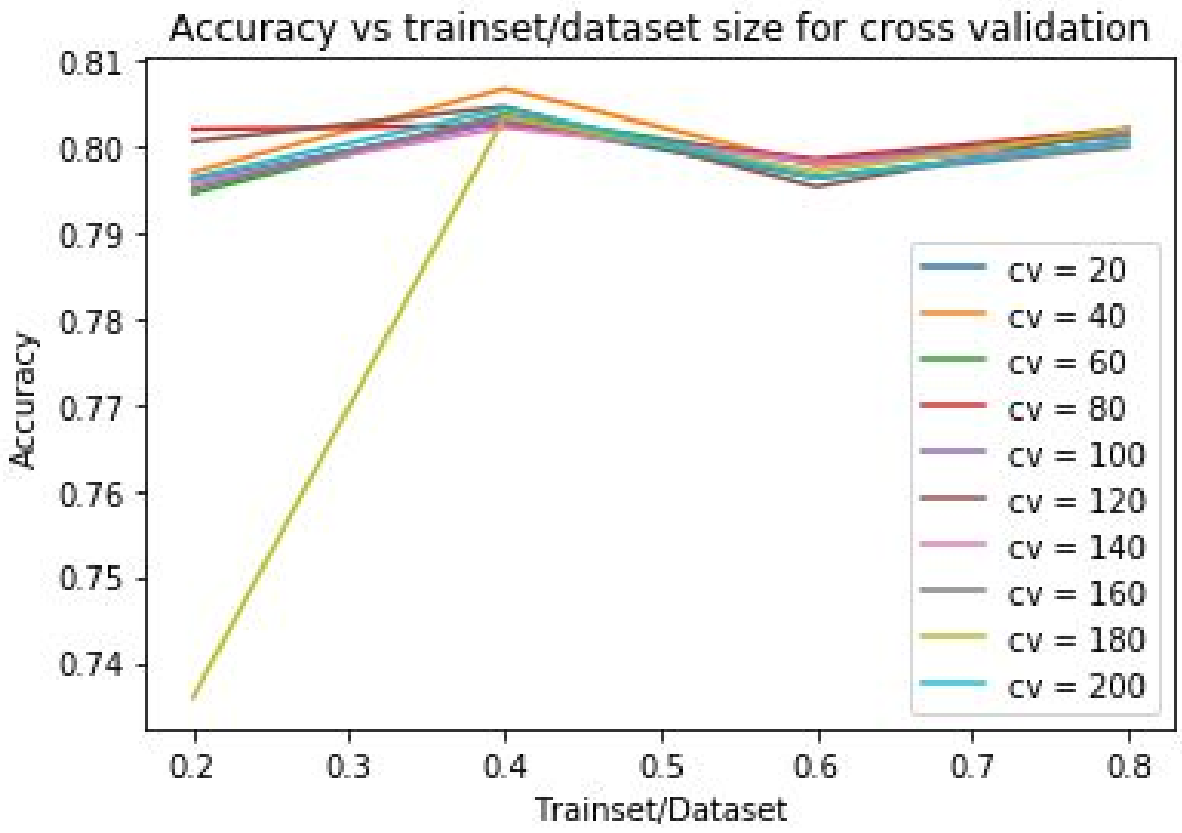
The following discussion is based on graphs, tables, numerical results and the analysis conducted in section 4.5. All in all, in terms of parameters, the following combination gives the best results:
Penalty norm ‘l2’ gave a slight accuracy improvement of about 0.1% and an increasing tpr compared to fpr. fit_intercept set to True means that a constant is used for the decision function. Using a constant yields a significant performance increase of approximately 7% as well as a tpr increase. multi_class should be set to ‘ovr’ since it was a binary classification problem .
‘Liblinear’ was chosen for solver as it is more suitable for the size of our dataset. Dual was chosen to be False, as the number of samples > number of features. Combining all optimal parameters the highest accuracy obtained was 80.4%
While determining the best train/test ratio it was understandable that increasing the train set size yields better results, however a logical balance had to be made to not use a high percentage of data as train data. Since the range of accuracy was between 79%-81%, with the lowest at 79% when train set contained 20% of the whole dataset, 4/6 split ratio was reasonable in this cas.
ROC curves for different cases do not differ substantially from each other and tpr is higher than fpr.
Using SelectKBest with chi score function and choosing to distinguish 5 most important features yields results very similar to those recorded while using all 19 features. The top 5 features were: tenure, online security, contract, monthly charge, total charge. Due to the miniscule difference in accuracy of about 0.01 it is not necessary to increase the number of features aiming for better accuracy. Reducing the number of features to 3 it was observed that the results were almost

identical with the previous using 5 features and 3 dominant features in this stage were tenure, monthly charge and total charge. Accuracy for the experiments with 5 and 3 was approximately 0.787.

Regarding cross validation, for 4 and 10 number of folds similar results were obtained and as can be seen from the graph, highest accuracy was obtained if train set was 40% of the whole dataset



Other number of folds from 20-200 were used and performance increases as cv increases gradually for most of the cases and reaches about 0.81, however when cv = 180 a strange result is observed where the accuracy is approximately 0.73. The following graph summarizes all cases for varying number of folds and trainset sizes and it can be deduced that the best performance was obtained when trainset was 40% of dataset and number of folds is 40. The accuracy is approximately 0.81.



5.6 Random Forest Classification

One of the most important observations was that the feature with the highest magnitude impacted the decisions of the model more than the other features combined. This makes sense since the ‘tenure’ feature means the user’s years spent on the operator. If a person spend many years at an operator, odds are much likely that the user is satisfactory with the provided service and has less probability of leaving it.

Feature selection didn’t really helped to increase the accuracy of the Random Forest Classification. However, it was a great approach to reduce the complexity and improve the performance speed.

The best train/test split ratio was 0.3. It is understandable, since smaller values leads to overfitting on the training set and makes the model less realistic when it encounters real values. Larger numbers were giving a smaller precision rate, probably because the model was not being fed with enough data to come up with logical outcomes.

It is really hard to make any comments on the hyper parameters because in each iteration, Random Forest Classifier creates different decision trees to use and this results in slightly different precision rates. So we concluded that random forest classification was an unstable approach. We observed different hyper parameter values being outputted in different iterations due to that. One of the reasonable outcomes from the experiments was that as the maximum number of leaf nodes increased, the model was overfitting.

6 Conclusion

Accuracy of most of the algorithms capped around 80%. Thus, this might be the limit which can be reached using this dataset. A better dataset might provide better accuracy. Thus answer to the prediction problem is one of the classifiers, preferably the easiest the compute ones. Because of this, neural network is not a good choice. We have learned about methods. Some of these methods were focused in the class and homeworks however we have learned about details of these methods. Also we have learned how to use them using libraries and use standard metrics on different models.

In the future, better datasets can be used in order to increase prediction accuracy. Also our code for an experimental commercial use is extremely primitive. This code can be improved so that a user can enter values using a GUI and get suggestions for price reduction and special offers on the GUI.

7 Appendix

7.1 Contributions

- Data preprocessing codes were written by **Alp Ege Baştürk** and **Buğra Aydın**.
- **Alp Ege** and **Buğra** have written the code skeletons to use keras and sklearn libraries.
- **Alp Ege** implemented the SVM and neural network and did experiments and analysis using these two methods.
- **Bora Ecer** has written the code related to the experiments of Decision Tree method which contained hyper parameter selection with cross validation, feature selection experiment that selected different features and selected the best out of them.
- **Alba Mustafaj** implemented different versions of Logistic Regression, experimenting with several sklearn functions for tuning parameters, performing feature selection, cross validation, etc, as well as visualized, discussed and drew conclusions based on the outcomes.
- **Buğra Aydın** completed the analysis of Random Forest classification, trying out different hyper parameters and feature combinations with k-fold cross validation and coming up with the best combination while visualizing and discussing the outcomes.
- **Berat Biçer** has implemented knn and done experiments using the model, including discussions about the results.

7.2 Code

Code for the project is stored on the following GitHub repository;
https://github.com/egebasturk/Telecom-User-Churn-Rate-Prediction.git

8. References

[1] Blast Char ”Telco Customer Churn”. *Kaggle.* [Online]. Available: https://www.kaggle.com/blastchar/telco-customer-churn. [Accessed: Oct. 2, 2018]