MATH167 Final Project Report
Customer Churn Prediction

Dan Zhai, Liyuan Sun, Shuai Li, Sangram Kapre
M.S Statistics, San Jose State University
May 2020

Introduction

In this project, our goal is to build a model for churn prediction, by analyzing the customer data. A model can then take new customer's data as input, and provide the prediction. This would be useful to identify the customers who are at a high risk of churning away from the company. A special attention can be given to those customers in order to retain them. We would try multiple models for classification and compare them based on various criterias.

Part 1: Data Description

The dataset is available at https://www.kaggle.com/blastchar/telco-customer-churn. The raw data contains 7043 rows and 21 columns. Each row represents a customer, and each column contains customer's attributes such as: *Gender, PhoneService, MonthlyCharges, Churn*, etc. "*Churn*" column is the target column. It is a binary variable thus making a problem a binary classification problem.
We grouped feature columns into 3 groups as follow:

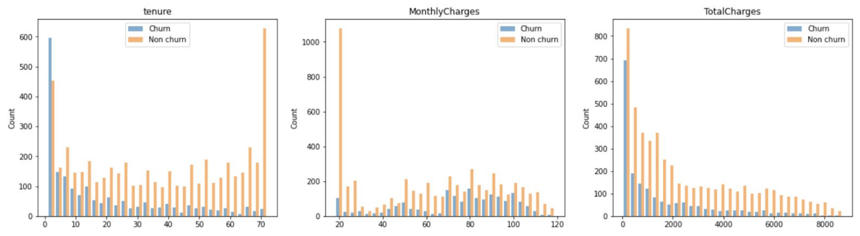
- Demographic information about customers: customerID / gender / SeniorCitizen / Partner / Dependents / tenure
- Services that each customer has signed up for: PhoneService / MultipleLines / InternetService / OnlineSecurity / OnlineBackup / DeviceProtection / TechSupport / StreamingTV / StreamingMovies
- Customer account information: Contract / PaperlessBilling / PaymentMethod / MonthlyCharges / TotalCharges

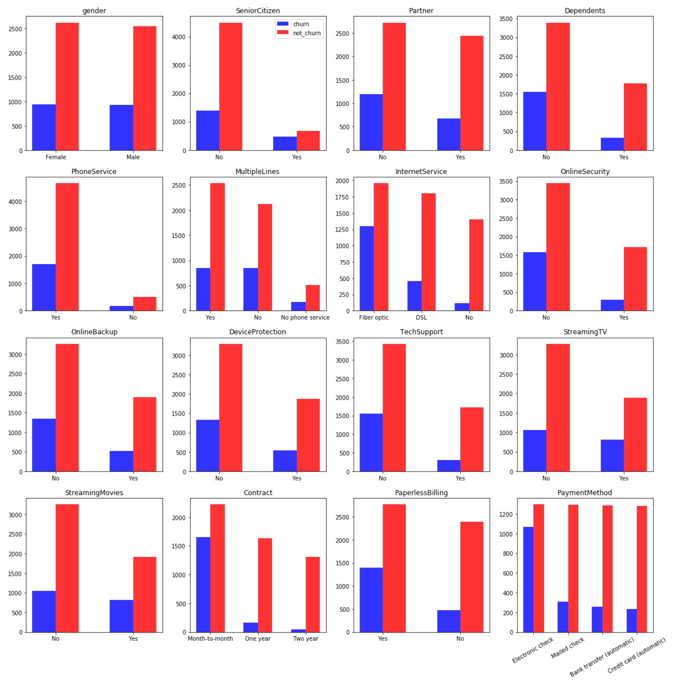
Part 2: Data Manipulation and Feature Preprocessing

Before feeding the labeled data to an ML algorithm, it is a good practice to inspect data through data summaries and visualizations. To make data more meaningful, we manipulate data and make feature preprocessing including:

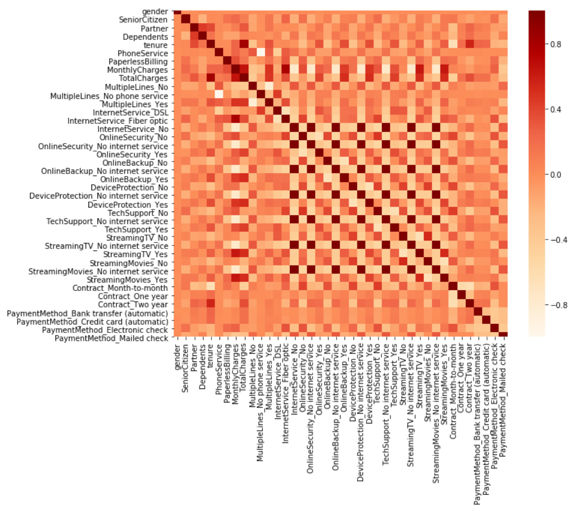
- Change data type to accommodate our analysis.
- Replacing or deleting missing values.
- Manipulate multilevel categorical variables, combining levels or making dummy variables.
- Change binary variables from (Yes, No) to (1, 0) for model training.

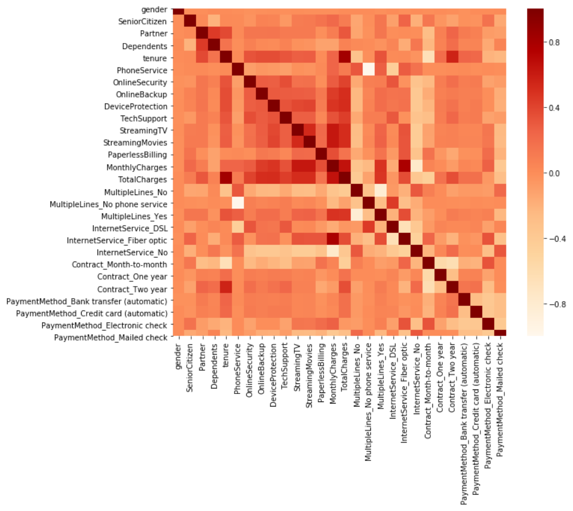
The plots below show us the distribution of numerical and categorical features based on different labels.





When analyzing the correlations of raw data, we identify a correlation problem. The following plots show the differences before and after the change we made.





Part 3: Model Training and Evaluation

In the raw customer dataset, the “Churn” column is our target. The training data consists of multi-dimensional (21 columns) observations with categorical labels: ‘Churn’ or ‘No_Churn’. The goal is to predict labels for new observations. First, to fit each model, we need to split the dataset into 2 parts:

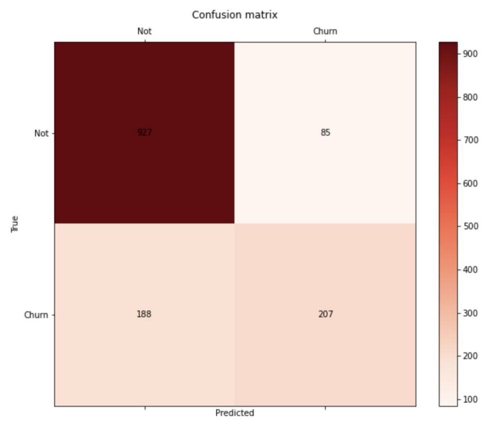
- Training data has 5625 observation with 28 features (80%)
- Testing data has 1407 observation with 28 features (20%)

Then we would fit 6 different classification models in the training dataset and make predictions based on each model. These 6 supervised learning methods are: Logistic Regression, K-Nearest Neighbor, Support Vector Machine, Random Forest, and Ensemble Model. For each model, we would show the best fitted cross validation model with hyperparameters. And the corresponding model evaluation results.

3.1 Logistic Regression

Logistic Regression is one of the most simple and commonly used Machine Learning algorithms for two-class classification. Logistic Regression predicts the probability of occurrence of a binary event utilizing a sigmoid function.

After applying the random grid search technology, We found our best parameters set is {‘penalty’=’L2’, ‘C’=29.7635}, where ‘penalty’ specifies the type of normalization used; ‘C’ is the regularization parameter of the error term, smaller values of this hyper-parameter indicates a stronger regularization.

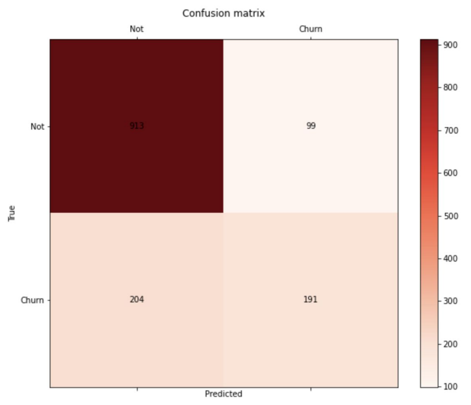


We then applied our training data into the Logistic Regression classifier with the best parameters, and got the confusion matrix with accuracy = 0.81.

3.2 K-Nearest Neighbor

The *k*-nearest neighbors algorithm (*k*-NN) is a non-parametric method used for classification. The input consists of the *k* closest training examples in the feature space. KNN works by finding the distances between a query and all inputs in the data. Next, it selects a specified number of inputs, say K, closest to the query. And then it votes for the most frequent label (in the case of classification).

After applying the random grid search technology, We found our best parameters set is {'weights': 'uniform', 'n_neighbors': 20, 'metric':’manhattan’}, where ‘weights’ specifies where each neighbor within the boundary carries the same weight ; ‘n_neighbors’ specifies the number of nearest neighbors k in the k-NN algorithm; ‘metric’ specifies the distance function/similarity metric for k-NN. Normally this defaults to the Euclidean distance.



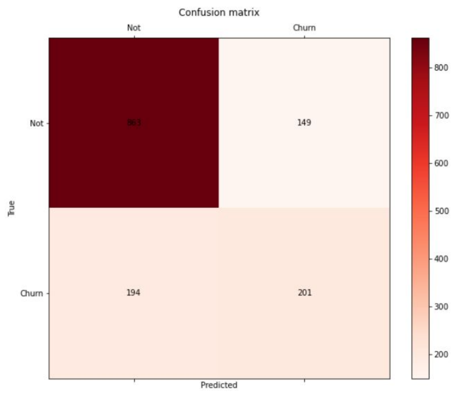
We then applied our training data into the K-Nearest Neighbor classifier with the best parameters, and got the confusion matrix with accuracy = 0.78.

3.3 Support Vector Machine

Support Vector Machines (SVM) is another supervised learning model used for this classification problem. The algorithm is to maximize the distances of nearest points from the separating boundary.

SVM is efficient for high dimensional data and highly nonlinear boundaries. The computer uses a kernel function that takes low-dimensional input vectors in the input space and returns the dot product of the vectors in a higher dimensional feature space. In this way, every point is mapped into a higher dimensional space. So that a nonlinear space is transformed into a linear space and we can find the optimal hyper-plan to separate our data.

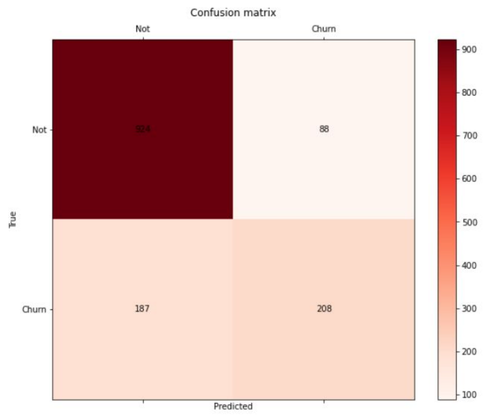
After applying the random grid search technology, We found our best parameters set is {'kernel': 'poly', 'gamma': 0.03, 'C': 60}, where ‘kernel’ specifies the kernel function used for finding our optimal hyper-plan; ‘gamma’ is the kernel coefficient. The higher the value of gamma, will better fit the training data set, and cause an overfitting problem; ‘C’ is the regularization parameter of the error term.



We then applied our training data into the SVM classifier with the best parameters, and got the confusion matrix with accuracy = 0.76.

3.4 Neural Network

Neural networks take inspiration from the learning process in human brains. It is composed of several layers : one input and one output layer, and at least one hidden layer in between. The layers are made of nodes, also referred to as neurons, is just a place where computation happens. Each neuron is a function which produces an output, after receiving one or multiple inputs. Those outputs are then passed to the next layer of neurons, which use them as inputs of their own function, and produce further outputs. It continues until every layer of neurons have been considered, and the terminal neurons have received their input, and output the final result for the model [7].



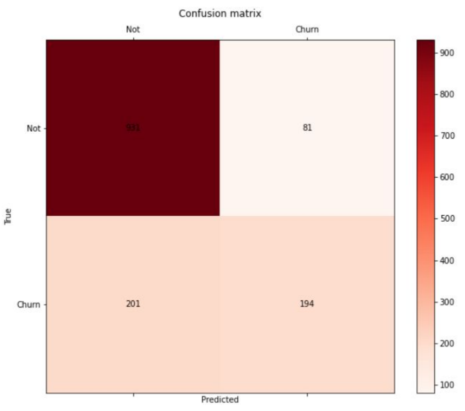
The best model gives us parameters set is {'max_iter': 80, 'activation': 'identity'', hidden_layer_sizes': (8,)}. ‘max_iter’ is the maximum number of iterations until convergence; ‘activation’ is the function we use in the hidden layer; ‘Hidden_layer_sizes’ is the number of neurons we want at each layer.

We then applied our training data into the neural networks classifier with the best parameters, and got the confusion matrix with accuracy = 0.80.

3.5 Random Forest

Random Forest is the Ensemble of multiple decision trees. It combines the ideas of bootstrapping and binary decision trees. Many trees are fitted on different subsets of the data. To predict the label for a new observation by taking the majority vote of these trees.

The best model parameters set is {'n_estimators': 400, 'max_features': 'auto', 'max_depth': 10, 'bootstrap': True}. 'n_estimators' is the number of trees in the forest; ‘max_depth’ is the maximum depth of the tree. If None, then nodes are expanded until all leaves are pure; ‘max_features’ is the number of features to consider when looking for the best split; ‘bootstrap’ controls whether bootstrap samples are used when building trees. If False, the whole dataset is used to build each tree.



We then applied our training data into the neural networks classifier with the best parameters, and got the confusion matrix with accuracy = 0.80.

3.6 Ensemble Model: Voting Classifier

Different classification models work based on different learning algorithms. Thus, each classifier tries to learn a specific set of patterns in the data. Sometimes, different models learn different patterns. So, one model might fail to predict on specific test data points where other models might work. The idea behind ensemble learning is to combine such individual models to improve the overall prediction performance on the test dataset. Ensemble model is a combination of multiple models (such as random forest, neural networks, KNN, etc.) that uses predictions from these individual models in order to predict better on a test dataset than each of the individual models.

There are multiple ways of building an ensemble model. Simplest way to build an ensemble classifier is to take a majority voting across predicted classes from individual models. This technique requires an odd number of models, and it completely ignores the predicted probabilities provided by the individual models. It usually does not work well in practice as the prediction probability information is lost during the aggregation process. Another way of combining predictions from multiple models is to use a simple average of prediction probabilities from each model to get the final prediction probability. This is better than majority voting as it uses probabilities from the individual models.

Even better approach to combine the probabilities from different models is to use a weighted average. Here, weights for each model can be decided based on their cross-validation performance in terms of AUC metric. Since each model is selected based on the best AUC score on cross-validation data, the ensemble model is expected to have the best overall AUC performance on the test data. We used this ensemble model as our final model for training on the train data and predicting on the test data. Following are the performance numbers for the ensemble model:

AUC on the test dataset: **84.3%**
Accuracy on the test dataset: **80%**

Part 4: Model Comparison

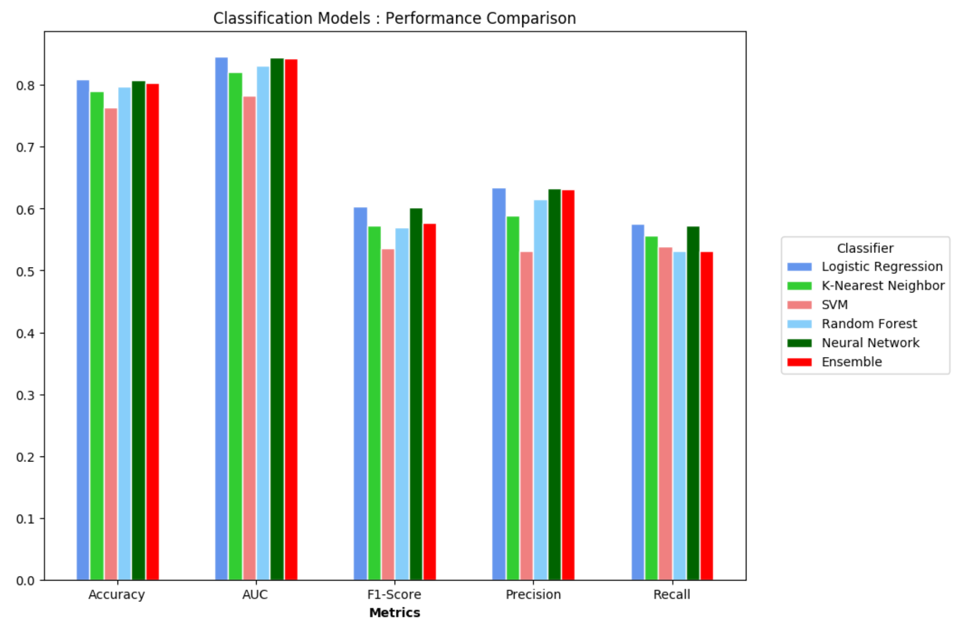
We used the following most commonly used metrics for classification to evaluate our models: **Accuracy, Area Under ROC Curve (AUC), Precision, Recall,** and **F-1 score**

The following table summarizes some of these metrics:

Accuracy	$\mathrm{ACC} = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}}$
Precision	$\mathrm{PRC} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$
Sensitivity	$\mathrm{SNS} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$
Specificity	$\mathrm{SPC} = \frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}}$
ROC	$ROC = \frac{\sqrt{SNS^2+SPC^2}}{\sqrt{2}}$
F_1 score	$F_1 = 2\frac{PRC \cdot SNS}{PRC+SNS}$

TP *= True Positive,* ***FP*** *= False Positive,* ***TN*** *= True Negative*
FN *= False Negative,* ***SNS*** *= Sensitivity (Recall),* ***SPC*** *= Specificity*

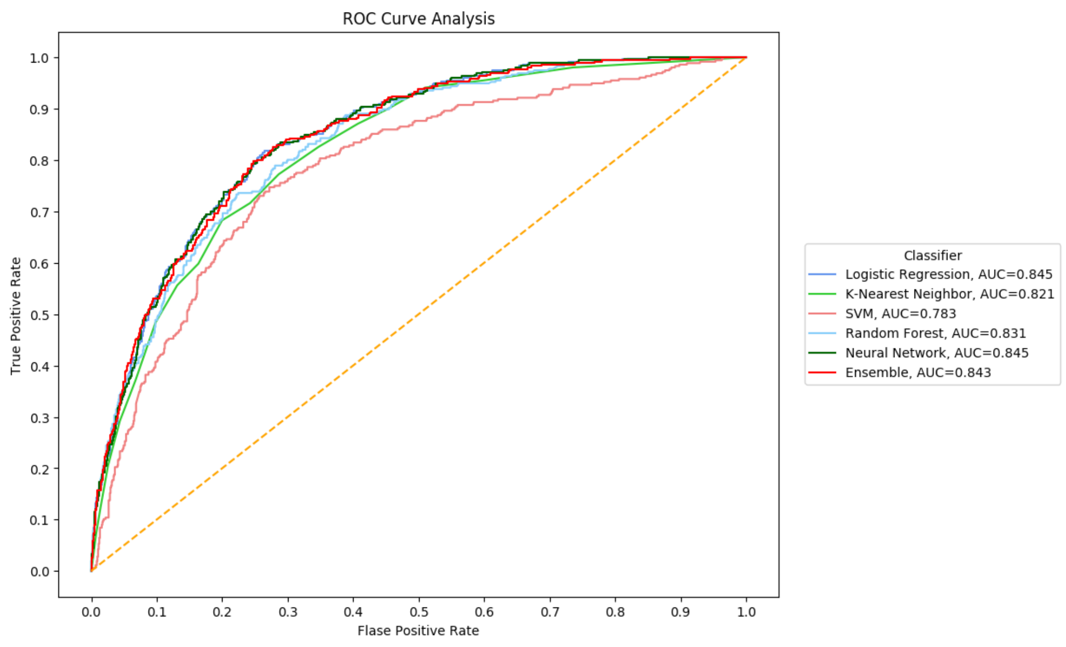
The following plot shows the comparison of all 6 models across the 5 metrics.



From the plot, we can clearly see that the ensemble model performs the best in terms of AUC. This is expected as each individual model is trained to have the best AUC cross validation performance. Also, the weights of each model are proportional to cross-validation AUC performance of each classifier.

We can also see that the ensemble model has a high precision score but then recall and f1-score are degraded. This is because tuning the model on one metric beyond a certain point might have adverse effects on the performance in terms of other metrics. Thus we need to be careful about which metrics are of interest to us before tuning the classifiers and ensemble model. Finally we also observe that SVM had the worst performance, and KNN was also not as good as Neural Networks or Random Forest models.

The ROC plot below shows that, ensemble model did not just have the highest AUC, but also had the best performance across almost all pairs of true positive and false positive values. This is an indication that ensemble is indeed the model in terms of AUC. Again, we can see that SVM and KNN had lowest ROC curves confirming that these models are not as good as other individual models in terms of AUC performance.



Conclusions

For this data, logistic regression and neural networks seem to perform better. SVM's performance was the least, whereas KNN's performance was second to the last. The final ensemble model performed as well or even better than the individual models, in terms of AUC and accuracy on the test data. We also observed that tuning the models on a particular set of metrics might degrade the model performance on other sets of metrics. Thus it is important to have knowledge about which metrics are of the most interest to us before tuning the individual as well as the final ensemble model.

Future Work

The main goal of this project was to do comparisons of different classification models and to build an ensemble model which would perform better than individual models. In this project, we have showcased a simple pipeline of data cleaning, feature preprocessing, model building and model evaluation. For this reason, the other tasks such as feature selection, model tuning, advanced ensemble techniques are not yet explored.

The following points indicate the natural extension to our project:

- Feature Selection: For now, we have used all the features available to build the models. In the future, supervised and unsupervised feature selection techniques can be employed in order to use only a subset of features that would lead to building the best overall model.

- Model Tuning: In this project, the main focus was to improve the overall AUC. Next step would be to tune the models on different metrics in order to improve the performance on the metrics of interest, instead of just AUC.
- Advanced Ensemble Techniques: Instead of weighted average of probabilities, like we did in this project, we can explore advanced techniques of combining multiple models such as building new models on the predictions provided by the individual models to have the final prediction probability as a polynomial combination of individual probabilities.

References

[1] AWE Developer Guide (2020).
https://docs.aws.amazon.com/machine-learning/latest/dg/feature-processing.html

[2] AWS Developer Guide (2020).
https://docs.aws.amazon.com/machine-learning/latest/dg/feature-processing.html

[3] WIKIPEDIA. https://en.wikipedia.org/wiki/Confusion_matrix

[4] Tara Boyle. ‘Hyperparameter Tuning’. Feb 16, 2019.
https://towardsdatascience.com/hyperparameter-tuning-c5619e7e6624

[5] Kevin’s Blog, ‘A Complete Guide to K-Nearest-Neighbors with Applications in Python and R’ Jul 13, 2016. https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/

[6] O. Knocklein, “Classification Using Neural Networks”, *towards data science*, Jun 2019. Accessed on
https://towardsdatascience.com/classification-using-neural-networks-b8e98f3a904f

[7] Erik. G, ‘K-Neighbors Classifier with GridSearchCV Basics’. Oct 21, 2018.
https://medium.com/@erikgreenj/k-neighbors-classifier-with-gridsearchcv-basics-3c445ddeb657