Telecom Customer Retention

Zheng Qi Hongyu Su

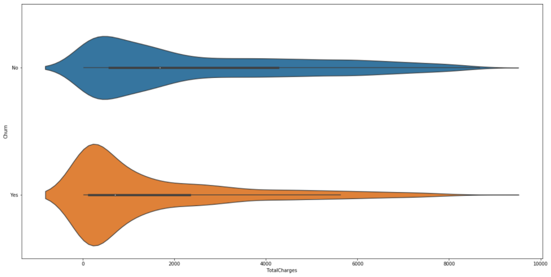
ABSTRACT

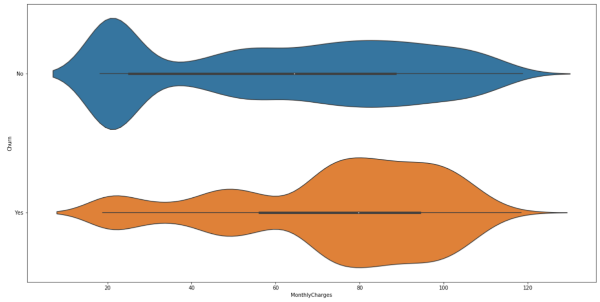
Customer Churn is one of the most important metrics for evaluating business growth potential. In this project, we used dataset from Kaggle to predict the retention of customers based on their behaviors and personal information, such as payment method, demographic information and related services. We used 4 machine learning methods: Random Forest, Naive Bayes, K-Nearest Neighbors (KNN) and Support Vector Machine (SVM). After a series of tuning and remodeling, we achieved the highest accuracy of 77% on the test set produced by KNN, SVM and Naive Bayes. Data and our implementation is available at https://github.com/issacqzh/COMP562_Project

I. Introduction

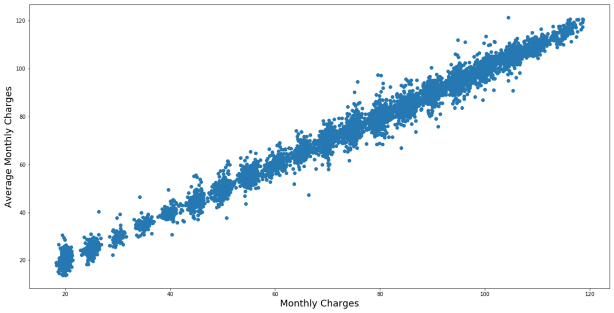
In the telecommunication field, it's important for companies to analyze customers' behaviors to better retain their customers. With the analysis of customer behavior data, companies can better develop their specific customer retention programs to improve their revenue. In the project we used dataset from Kaggle with 20 variables about customer behaviors and 7043 data to make predictions about customers' retention. Here, we used 4 machine learning methods: Random Forest, Naive Bayes, K-Nearest Neighbors and Support Vector Machine.

II. Data Representation

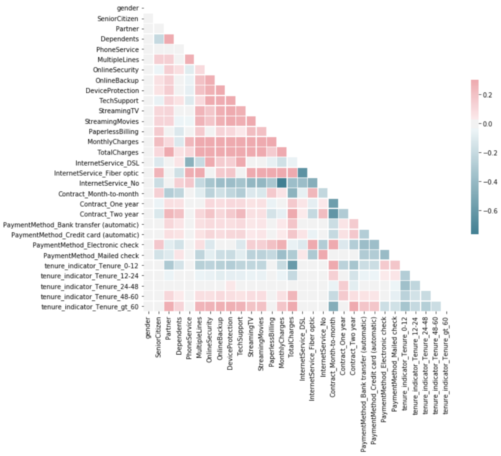




Based on these two violin graphs, 'Total Charges' and 'Monthly Charges' have different distributions for 'Churn' and 'Not Churn' groups, which means these two features will probably have significant effects on the churn prediction.



Since monthly charges and tenure should have correlations toward each other, we divided total charges by tenure (months) to obtain average monthly charges. Then we drew a scatter plot to show the correlation between Average Monthly Charges and Monthly Charges. Based on the graph, these two features nearly fit on a straight line.



Moreover, according to the correlation heatmap, we didn’t see many feature pairs which have extremely high correlations, so we didn’t drop any feature.

III. Procedure

A. Data Preprocessing

For data preprocessing, we first converted column ‘senior citizen' [1,0] to [yes, no] so that all categorical answers would be categorized as yes or no . Since feature “tenure” had a large spread and we didn’t think that slight difference in tenure would affect the churn too much, we converted “tenure” also into a categorical feature. For missing values, there were not too many missing values and only the column “total charges” had missing values so we dropped the rows with missing data. We then replaced all values ending “no internet service” with “no” to make sure that they fall in the same category with “no”. We then used one hot encoding for multiple value columns and standardized all features having value inputs. We then encoded every label. We also plotted the correlation table of all features so we could have better feature selection.

B. KNN Classifier

We first tried the KNN classifier. We implemented it with Kneighborsclassifier from scikit-learn with number of neighbors=50. All points in each neighborhood were weighted equally and we used KD tree search to compute the nearest neighbors. For the leaf size passed to that, we used 30.

C. Random Forest Classifier

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting.[1]. It randomly samples training data points when building trees and uses random subsets of features when splitting nodes. in the project, we used the random forest classifier from scikit learn. For the criterion to measure the quality of each split, we used Gini Impurity. The Gini impurity of a node is the probability that a randomly chosen sample in a node would be incorrectly labeled if it was labeled by the distribution of samples in the node.[2].

$$I_G(n) = 1 - \sum_{i=1}^{J} (p_i)^2$$

At each node, the tree searches through the value to split on that results in greatest deduction in Gini impurity. In the models, we set number of trees in the forest as 100 and maximum depth of the tree to be 10. We then used the RandomForestClassifier(bootstrap= “true”, criterion=”Gini”, max_depth=10) to predict.

D. Bernoulli Naïve Bayes

Bayesian classifier is a collection of classification algorithms based on Bayes’ theorem that share a principle: every pair of features is independent of each other. The Bayes’ Theorem finds the probability of an event occurring given on probability that another event has already occurred. Its mathematical formula is

$$p(B|A) = \frac{p(A,B)}{p(A)} = \frac{p(A|B)p(B)}{p(A)} = \frac{p(A|B)p(B)}{\int_b p(A|B=b)p(B=b)db}$$

The extension of naive Bayes is Bernoulli Naive Bayes. We then implemented Bernoulli Naive Bayes with scikit learn BernoulliNB().

E. Support Vector Machine

The fourth method that could be used is support vector machine. The SVM modeling algorithm finds an optimal hyperplane with the maximal margin to separate 2 classes.

$$minimize\ \|w\|^2\ \ if\ y_i(w^T x_i + b) >= 1$$

We can solve it through gradient descent algorithm. To minimize the cost function, we can use the iterative updates.

$$w_{t+1} = w_t - \eta(\lambda w_t - y_i x_i) \; if \; y_i f(x_i) < 1$$
$$else \; w_{t+1} = w_t - \eta\lambda w_t$$

Another approach is deducted through Lagrange

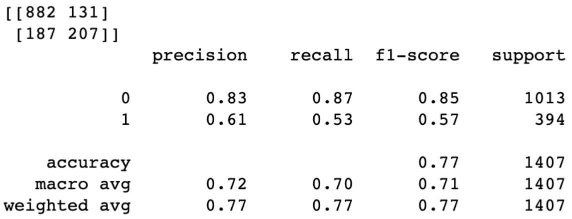
$$maximize \sum a_i - \frac{1}{2}\sum a_i a_j y_i y_j K(x_i, x_j)$$
$$we \; can \; get \; w = \sum a_j x_j y_j \; and \; \sum a_j y_j = 0$$

K is the kernel function that measure the similarity between the 2 samples. In our project, we used RBF kernel.

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right)$$

IV. Result

A. KNN Classifier



```
[[882 131]
 [187 207]]
              precision    recall  f1-score   support

           0       0.83      0.87      0.85      1013
           1       0.61      0.53      0.57       394

    accuracy                           0.77      1407
   macro avg       0.72      0.70      0.71      1407
weighted avg       0.77      0.77      0.77      1407
```

Table 1: Accuracy Table for KNN model

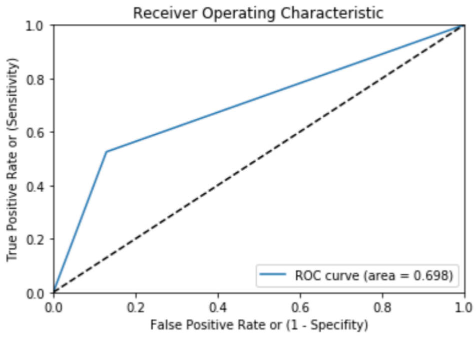
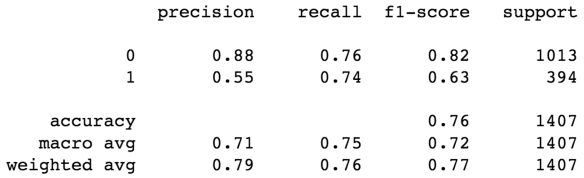


Table 2: ROC Curve for KNN model

With the KNN Classifier model, we achieved 77% accuracy and 0.7 AUC.

B. Random Forest Classifier



```
              precision    recall  f1-score   support

           0       0.88      0.76      0.82      1013
           1       0.55      0.74      0.63       394

    accuracy                           0.76      1407
   macro avg       0.71      0.75      0.72      1407
weighted avg       0.79      0.76      0.77      1407
```

Table 3: Accuracy Table for random forest classifier

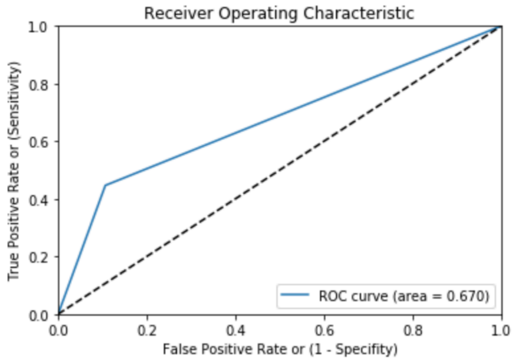
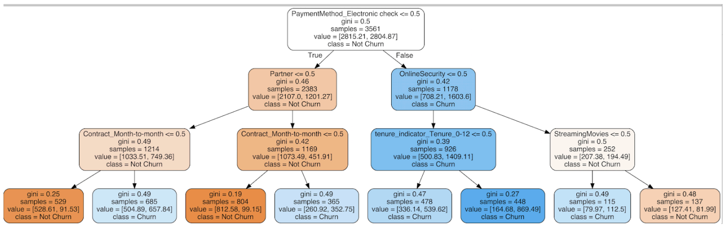


Table 4: ROC Curve for random forest classifier

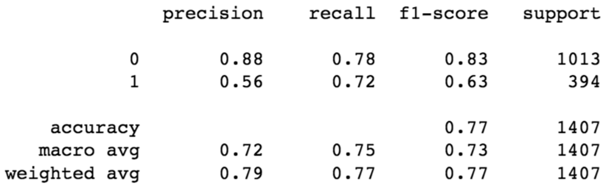


PaymentMethod_Electronic check <= 0.5 gini = 0.5 samples = 3561 value = [2815.21, 2804.87] class = Not Churn
True
False
Partner <= 0.5 gini = 0.46 samples = 2383 value = [2107.0, 1201.27] class = Not Churn
OnlineSecurity <= 0.5 gini = 0.42 samples = 1178 value = [708.21, 1603.6] class = Churn
Contract_Month-to-month <= 0.5 gini = 0.49 samples = 1214 value = [1033.51, 749.36] class = Not Churn
Contract_Month-to-month <= 0.5 gini = 0.42 samples = 1169 value = [1073.49, 451.91] class = Not Churn
tenure_indicator_Tenure_0-12 <= 0.5 gini = 0.39 samples = 926 value = [500.83, 1409.11] class = Churn
StreamingMovies <= 0.5 gini = 0.5 samples = 252 value = [207.38, 194.49] class = Not Churn
gini = 0.25 samples = 529 value = [528.61, 91.53] class = Not Churn
gini = 0.49 samples = 685 value = [504.89, 657.84] class = Churn
gini = 0.19 samples = 804 value = [812.58, 99.15] class = Not Churn
gini = 0.49 samples = 365 value = [260.92, 352.75] class = Churn
gini = 0.47 samples = 478 value = [336.14, 539.62] class = Churn
gini = 0.27 samples = 448 value = [164.68, 869.49] class = Churn
gini = 0.49 samples = 115 value = [79.97, 112.5] class = Churn
gini = 0.48 samples = 137 value = [127.41, 81.99] class = Not Churn

Table 4: One Decision Tree in Random Forest Graph

With the random forest tree model, we achieved 76% accuracy and AUC is 0.75. The disadvantage of the model is that it requires much more time to train because of its complexity of multiple decision trees.

C. Bernoulli Naïve Bayes



```
              precision    recall  f1-score   support

           0       0.88      0.78      0.83      1013
           1       0.56      0.72      0.63       394

    accuracy                           0.77      1407
   macro avg       0.72      0.75      0.73      1407
weighted avg       0.79      0.77      0.77      1407
```

Table 5: Accuracy Table for Naïve Bayes

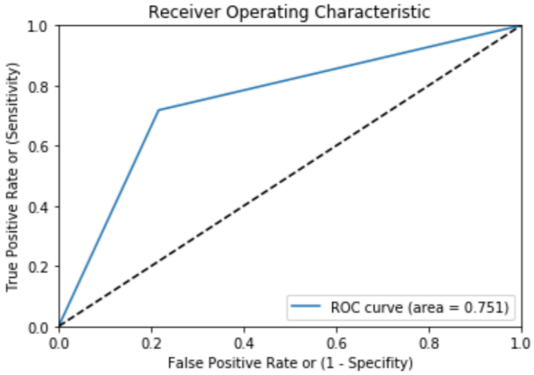
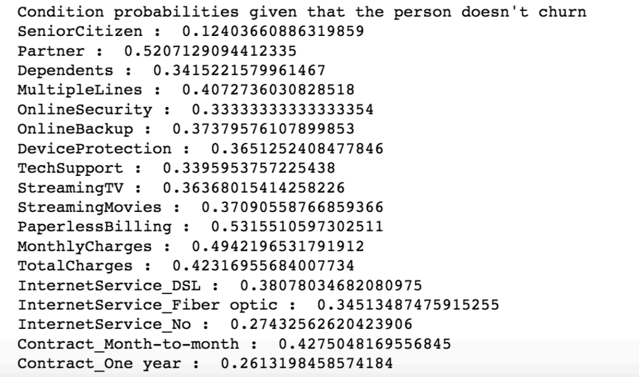


Table 6: ROC curve for Naïve Bayes model

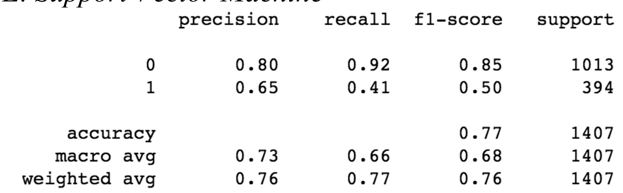


```
Condition probabilities given that the person doesn't churn
SeniorCitizen :  0.12403660886319859
Partner :  0.5207129094412335
Dependents :  0.3415221579961467
MultipleLines :  0.4072736030828518
OnlineSecurity :  0.33333333333333354
OnlineBackup :  0.37379576107899853
DeviceProtection :  0.3651252408477846
TechSupport :  0.3395953757225438
StreamingTV :  0.36368015414258226
StreamingMovies :  0.37090558766859366
PaperlessBilling :  0.5315510597302511
MonthlyCharges :  0.4942196531791912
TotalCharges :  0.42316955684007734
InternetService_DSL :  0.38078034682080975
InternetService_Fiber optic :  0.34513487475915255
InternetService_No :  0.27432562620423906
Contract_Month-to-month :  0.42750481695556845
Contract_One year :  0.2613198458574184
```

Table 7: conditional probability table

For Bernoulli Naïve Bayes model, it achieved 77% accuracy and AUC is 0.75. We also included part of the conditional table of different factors given that the person didn't churn.

E. Support Vector Machine



```
              precision    recall  f1-score   support

           0       0.80      0.92      0.85      1013
           1       0.65      0.41      0.50       394

    accuracy                           0.77      1407
   macro avg       0.73      0.66      0.68      1407
weighted avg       0.76      0.77      0.76      1407
```

Table 8: Accuracy table for SVM

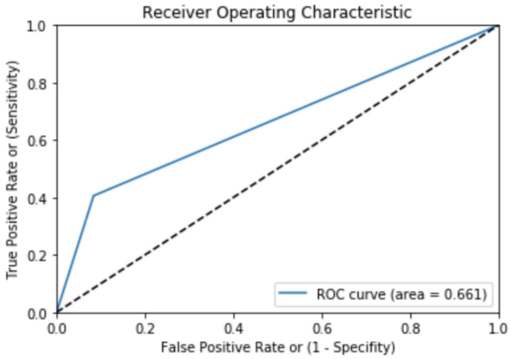


Table 9: ROC curve for SVM

In SVM model, we achieved accuracy of 77% with RBF Kernel and the AUC is 0.66.

V. Discussion and Conclusion

In conclusion, we developed four machine learning models, including Random Forest, Naive Bayes, K-Nearest Neighbors (KNN) and Support Vector Machine (SVM), and each of these four models generate decent performances. The highest test performance of our model achieved an accuracy of 78%. Moreover, based on our model, we found that features Total Charges, Contract, tenure, Internet Service, Payment Method play a crucial role in the prediction. That means, they are important features. Since we only have about 7000 samples, the model may not converge completely. With more samples, I believe the performance will increase.

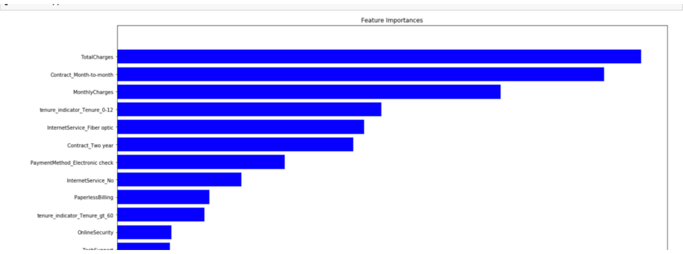


Table 10: Feature Importance generated by Random Forest Tree

Reference

[1]]https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[2] https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76