kamileyagci / dsc-phase-3-project Public

forked from learn-co-curriculum/dsc-phase-3-project

View license

0 stars 58 forks

Star Watch

Code Pull requests Actions Projects Wiki Security Insights

main

This branch is 49 commits ahead of learn-co-curriculum:main. Contribute Fetch upstream

kamileyagci modified ... 31 seconds ago 62

View code

README.md

SyriaTel Customer Churn Study

Author: Kamile Yagci

Blog URL: https://kamileyagci.github.io/

Overview

In this study, I analyzed the 'SyriaTel Customer Churn' data. The SyriaTel is a telecommunication company. The purpose of the study is to predict whether a customer will ("soon") stop doing business with SyriaTel.

learn-co-c.../dsc-phase-3-project Public

58 Updated on Oct 14, 2021

The telecommincation company, SyriaTel, hired me to analyze the Customer Churn data. The company wants to understand the customer's decision to discontinue their business with SyriaTel. The results of the analysis will be used make business decisions for improving the company finances.

This study will

- Search for the predictable pattern for customer decision on stop or continue doing business with SyriaTel
- Choose a model which will best identify the customers who will stop doing business with SyriaTel

Method

I followed the following steps in this project:

1. Data
    - Load
    - Scrub/Explore
2. Model
    - Pre-Process
    - Evaulation Metrics
    - Logictic Regression
    - K-Nearest Neighbor
    - Decision Trees
    - Random Forest
    - XGBoost
3. Interpret
4. Future work

Data

Load

I used SyriaTel Customer Churn data for this study. The data file is downloaded from Kaggle.

The file name is 'bigml_59c28831336c6604c800002a.csv'.

Tha raw data has 3333 entries and 21 columns.

Scrub/Explore

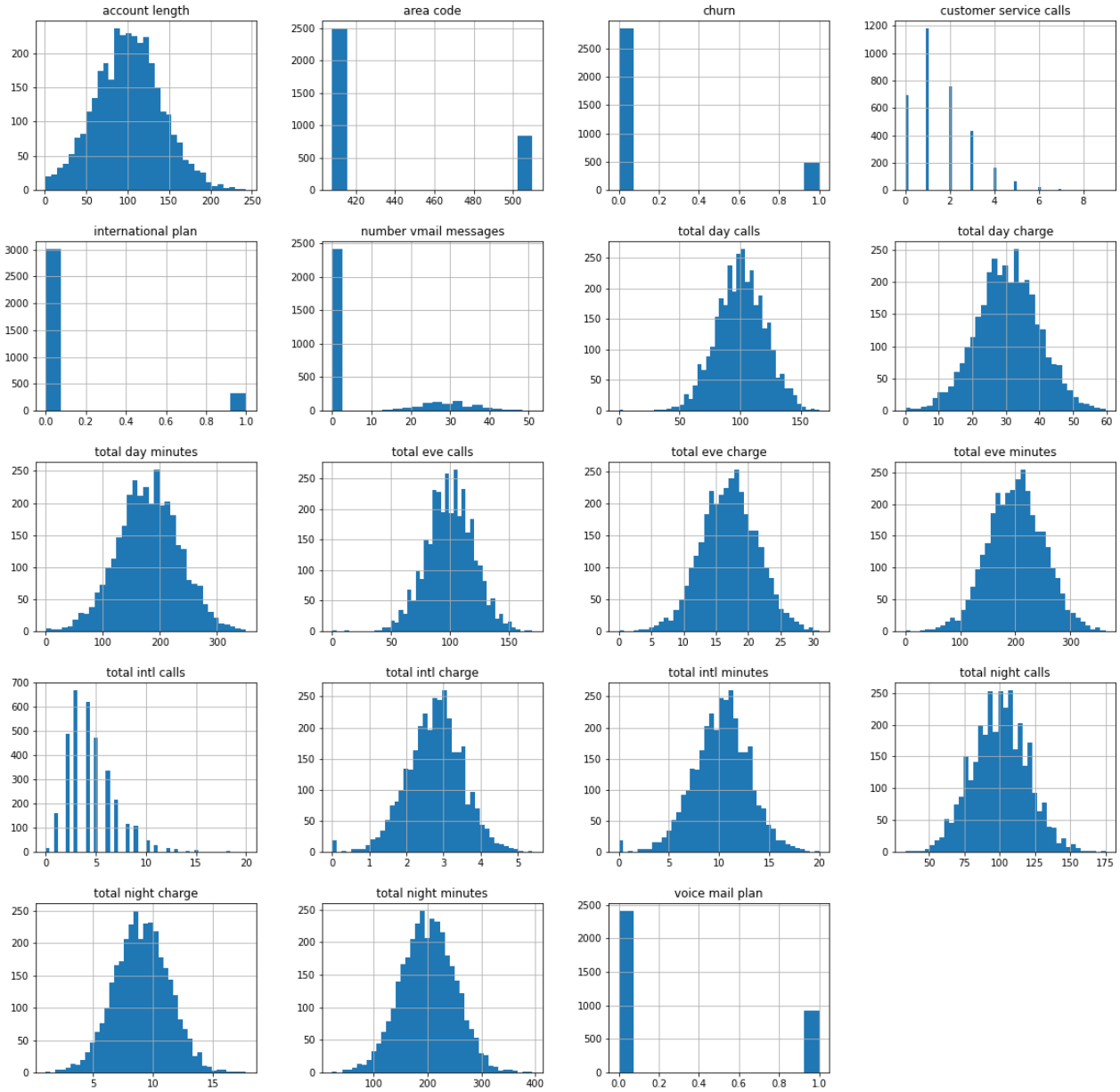
The column/variable names are:

- state
- account length
- area code
- phone number
- international plan
- voice mail plan
- number vmail messages
- total day minutes
- total day calls
- total day charge
- total eve minutes
- total eve calls
- total eve charge
- total night minutes
- total night calls
- total night charge
- total intl minutes
- total intl calls
- total intl charge
- customer service calls
- churn

The data doesn't have any missing values.

I removed the column 'phone number' from dataset. Most digits in the phone number are random, and it will not have much use in modeling. This variable will also be a problem in dummy variable creation, because each phone number value is unique.

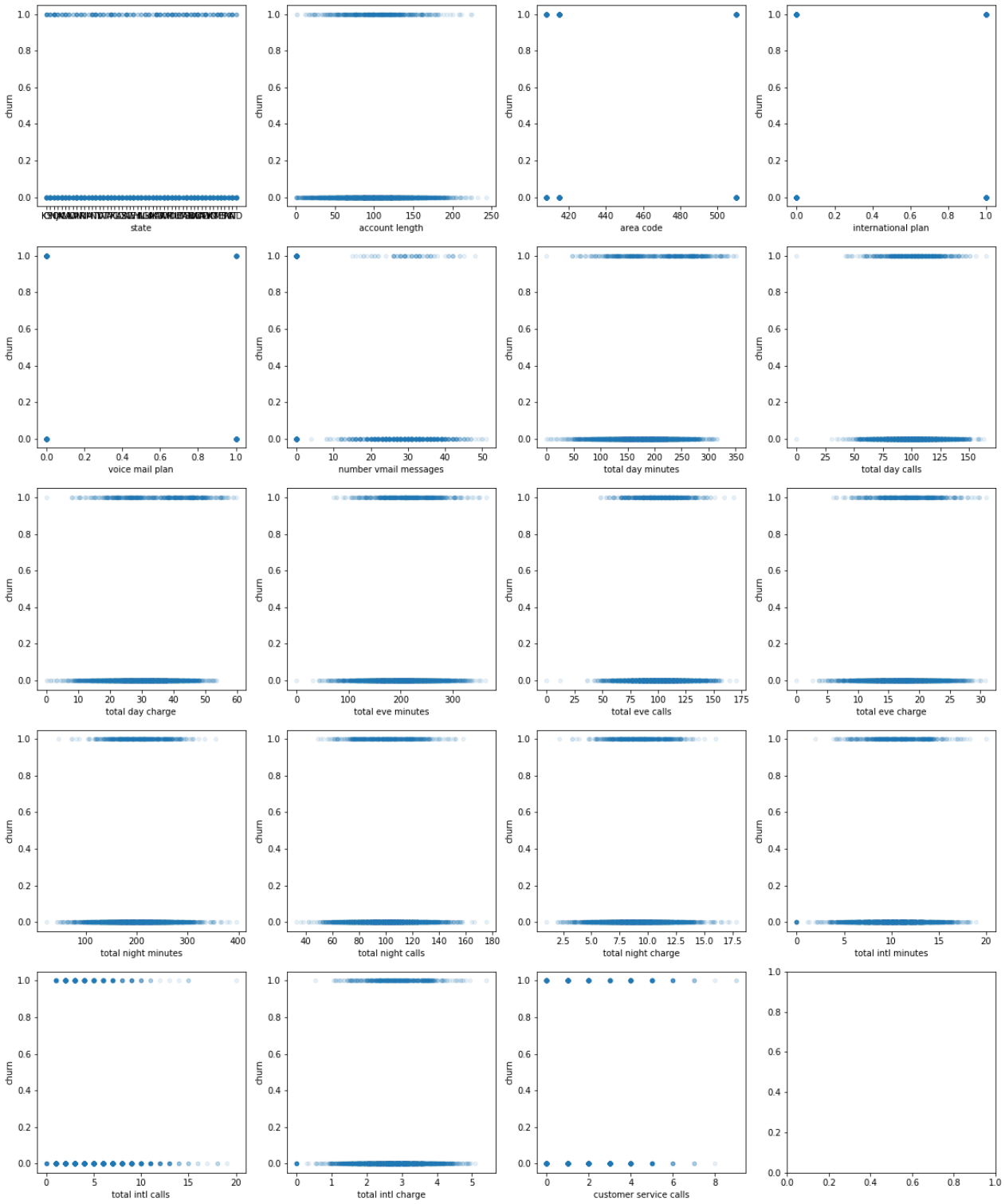
The distribution of variables are shown below. Click on the plot to see them closer.



The target variable for this study is 'churn'. The rest of the variables in the dataset will be predictors.

'churn': activity of customers leaving the company and discarding the services offered

The scatter graphs for 'churn' vs predictors are shown below. Click on the plot to see them closer.



It is hard to recognize any patterns or correlation for 'churn' in these plots.

We will now look at the models to derive patterns and predictions.

Model

In this study, we are trying to predict customer's decision on stopping the business with the company. The prediction will be True (1) or False (0). Therefore we will use binary classification model.

Pre-process

Before modeling, I divided the dataset into target data series (y) and predictor dataframe (X).

- y: DataSeries of 'churn'
- X: DataFrame of all predictors

I also created dummy variables from categorical variables. The X DataFrame has 73 variables together with dummies.

Then, I seperated the data into train and test splits. I allocated 25% of the data for testing. I also assigned a random state for repeatability.

The shape of the splits:

- X_train shape = (2499, 73)
- y_train shape = (2499,)
- X_test shape = (834, 73)
- y_test shape = (834,)

shape = (number of rows/entries, number of columns/variables)

The next step is standardization. The data values have different ranges, so I did normalize/scale each variable in train and test data (X) before modeling. I used Scikit-Learn StandardScaler.

Evaluation Metrics

In the next steps, I will use several classifiers to model the data. I will check their performance using the evaluation metrics:

precision:

- Number of True Positives / Number of Predicted Positives
- How precise our predictions are?

recall:

- Nuber of True Positives / Number of Actual Total Positives

- What percentage of the classes we're interested in were actually captured by the model?

accuracy:

- (Number of True Positives + Number of True Negatives) / (Number of Total Observations)
- Out of all the predictions our model made, what percentage were correct?

f1-score:

- 2 * (Precision * Recall) / (Precision + Recall)
- Harmonic Mean of Precision and Recall.

Source: Flatiron Data Science Curriculum, Evaluation Metrics

Since my business problem is focusing on identfying the customers who stop doing business, I am interested mainly on the 'recall' metrics. However, when optimizing my model, I should also pay attention to the 'precision'. I want my predictions to be true, to be precise. The recall and precision are inversely proportional. Therefore, I choose to use the f1-score, Harmonic Mean of Precision and Recall, as the main metric for evaluating the performance of the model.

Logistic Regression

I started modeling with Logistic Regression classifier (LogisticRegression). I instantiated the model with default parameters and fit on training data. Then I checked the evaluation metrics both for training and testing data.

	f1-score	recall	precision	accuracy
Train	0.37	.27	0.64	0.85
Test	0.32	.22	0.56	0.86

- The metrics look similar for both training and testing data, just training is a bit better; so slight overfitting.
- The precision - recall - f1 scores are low (for churn=1), so the model prediction performance is not good.
- The high accuracy score is high, but misleading. It is caused by the imbalanced dataset.

Class imbalance effects the performance of the classification model. I have looked at the class distributions for the whole data: train + test:

	Value Counts in whole dataset	Normalized
churn = 0	2850	0.855
churn = 1	483	0.145

According to the dataset, 85.5% of the customers do continue with SyriaTel and 14.5% of customers stop business. If we make a prediction saying all customers will continue business, then we will have about 85.5% accuracy. This explains the high accuracy score of the model, despite the other low metric values.

I used SMOTE to create a synthetic training sample to take care of imbalance. After the resampling, the value counts in each class, in training data sample, became equal.

	Original training data, Value counts	Synthetic training data, Value counts
churn = 0	2141	2141
churn = 1	358	2141

I have then reapplied the Logistic Regression, using the resampled training data. The results:

	f1-score	recall	precision	accuracy
Train	0.49	.75	0.36	0.78
Test	0.52	.78	0.39	0.79

After resampling, the Logistic Regression Model performance (f1-score and recall) is clearly improved.

I initially used the default paremeters for the Logistic Regression model. I then applied parameter tuning with GridSearchCV. It determined the best parameter combination for the given parameter grid. I used the f1-score for tuning.

The results of parameter tuning:

- f1-score for test data: 0.5166240409207161

- Best Parameter Combination: {'C': 0.001, 'solver': 'liblinear'}

It looks like the parameter tuning, with the given parameter grid, didn't improve the performance of Logistic Regression much. The f1-score didn't change.

K-Nearest Neighbors

My next classifier is K-Nearest Neighbors (KNeighborsClassifier). I used the resampled training data for fitting the model with default parameters.

	f1-score	recall	precision	accuracy
Train	0.64	.99	0.47	0.84
Test	0.39	.62	0.29	0.71

- The performance in training data is better than test data. This is a sign of overfitting.
- The fitting on resampled training data has a better performance than unsampled data. The f1-score for test increased from 0.15 to 0.39. (The full results for unsampled data is not shown here).

Then, I used GridSearchCV for parameter tuning. The results of parameter tuning:

- f1-score for test data: 0.27751196172248804
- Best Parameter Combination: {'n_neighbors': 1, 'p': 4}

Parameter tuning, with the given parameter ranges, didn't improve the KNN model performance. Actually, f1-score decreased. Why?

Decision Tress

I firstly used DecisionTreeClassifier with default parameters, then applied GridSearchCV to find the optimum parameteres.

	f1-score	recall	precision	accuracy
Train	1.00	1.00	1.00	1.00
Test	0.75	.75	0.75	0.93

The results of parameter tuning:

- f1-score for test data: 0.8088888888888889

- Best Parameter Combination: {'criterion': 'gini', 'max_depth': 6, 'min_samples_split': 6}

The parameter tuning significantly improved the Decision Trees performance.

Random Forests

Next, I used ensemble method Random Forests (RandomForestClassifier), which uses DecisionTreeClassifier.

	f1-score	recall	precision	accuracy
Train	1.00	1.00	1.00	1.00
Test	0.77	.63	0.98	0.94

The results of parameter tuning:

- f1-score on test data: 0.7326732673267325
- Best Parameter Combination: {'criterion': 'gini', 'max_depth': None, 'max_features': 8, 'min_samples_split': 3, 'n_estimators': 100}

The paremeter tuning didn't improve the performance of Random Forest model.

XGBoost

Last, I used another ensemble method XGBoost (XGBClassifier).

	f1-score	recall	precision	accuracy
Train	1.00	1.00	1.00	1.00
Test	0.83	.74	0.94	0.95

The results of parameter tuning:

- f1-score on test data: 0.8288288288288288
- Best Parameter Combination: {'learning_rate': 0.1, 'max_depth': 10, 'min_child_weight': 1, 'n_estimators': 100, 'subsample': 0.7}

The parameter tuning didn't effect the XGBoost performance much.

Compare the models

I compared the classification models to choose the best one that identifies the customers who will study doing business with SyriaTel .

I looked at the evaluation metrics like precision, recall, accuracy and f1.

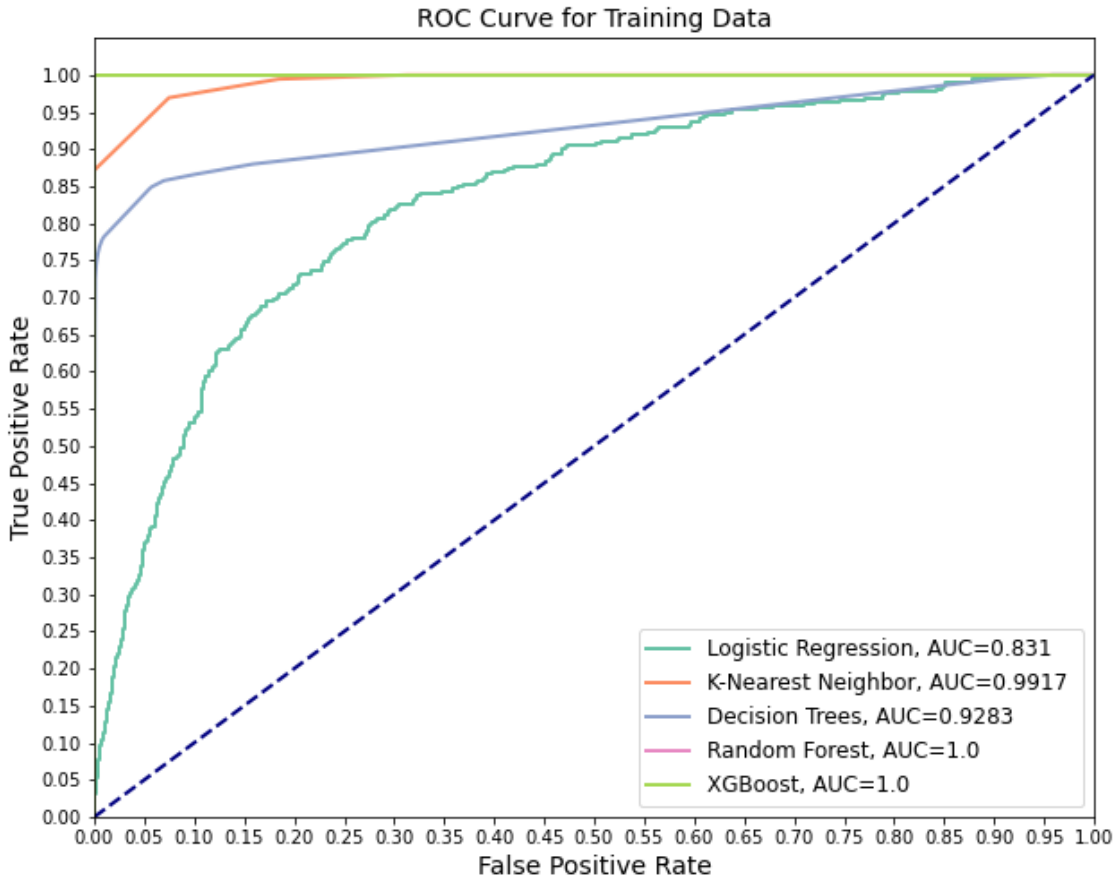
I also plotted ROC curves and calculated AUC for each model.

- ROC: Receiver Operating Characteristic curve illustrates the true positive rate against the false positive rate.
- AUC: Area Under Curve

I used the optimal/best parameter set selected by the GridSearchCV to instantiate my models.

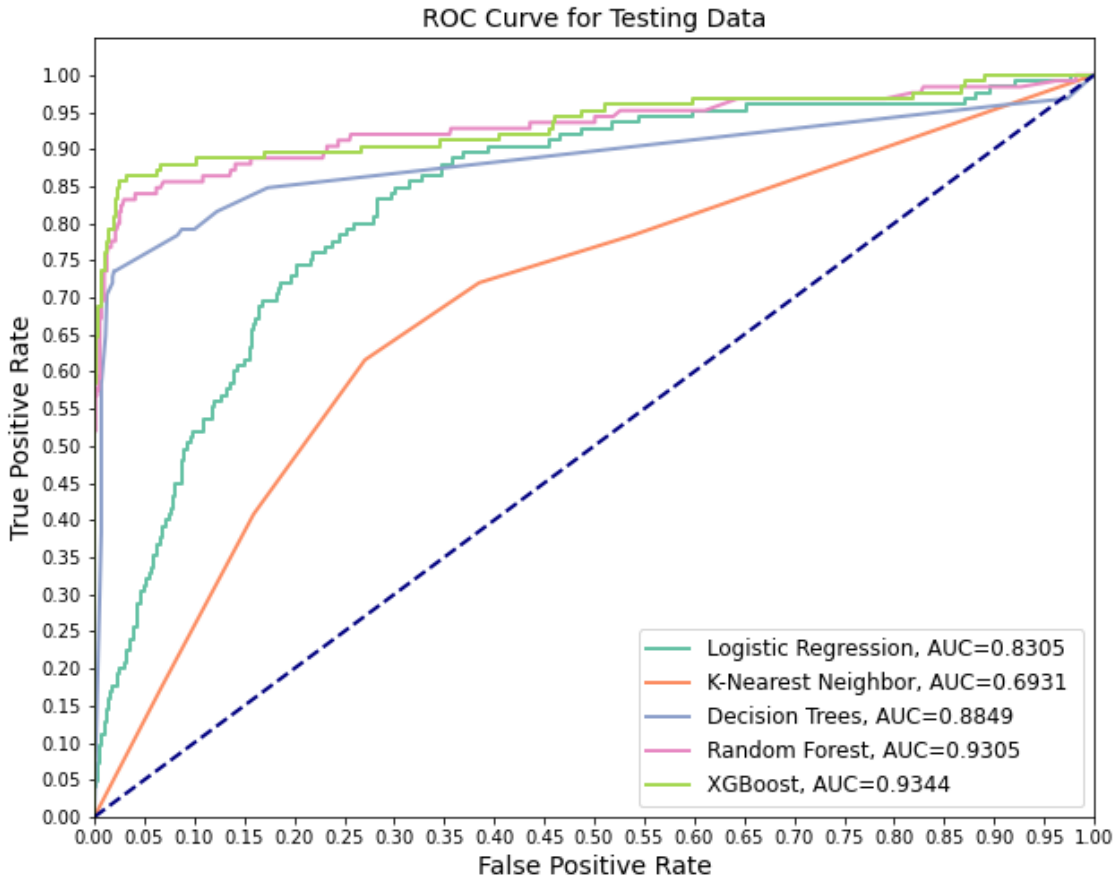
The evaluation metrics and ROC Curve for Training data:

model	precision	recall	accuracy	f1	auc
Logistic Regression	0.314378	0.818436	0.718287	0.454264	0.831022
K-Nearest Neighbor	0.474035	0.994413	0.841136	0.642020	0.991698
Decision Trees	0.971429	0.759777	0.962385	0.852665	0.928302
Random Forest	1.000000	0.986034	0.997999	0.992968	1.000000
XGBoost	1.000000	0.994413	0.999200	0.997199	1.000000



The evaluation metrics and ROC Curve for Test data:

model	precision	recall	accuracy	f1	auc
Logistic Regression	0.330189	0.840	0.720624	0.474041	0.830511
K-Nearest Neighbor	0.286245	0.616	0.712230	0.390863	0.693106
Decision Trees	0.873786	0.720	0.942446	0.789474	0.884897
Random Forest	0.961039	0.592	0.935252	0.732673	0.930482
XGBoost	0.948454	0.736	0.954436	0.828829	0.934409



Interpret

All of my models showed some pattern for customer decision on stop or continue doing business. They also did predictions to identify the customers who will discontinue service (churn customers).

Which model is best on identinfying churn customers?

I used the test data evaluation results to do model comparisons and choose the final model.

Here are my observations based on evaluation metrics and AUC:

- Overall performance: Decision Trees, Random Forest and XGBoost are top three.
- f1-score: Decision Trees, Random Forest and XGBoost are best
- recall: Decision Trees and XGBoost have better scores
- precision: Random Forest and XGBoost are best
- accuracy: Decision Trees, Random Forest and XGBoost are top three

- AUC: Random Forest and XGBoost have better value

The results showed that XGBoost classifier has the best performance in all aspects. It also has the best 'recall' and 'f1-score', which matters most for my study.

I choose the XGBoost model as my final model.

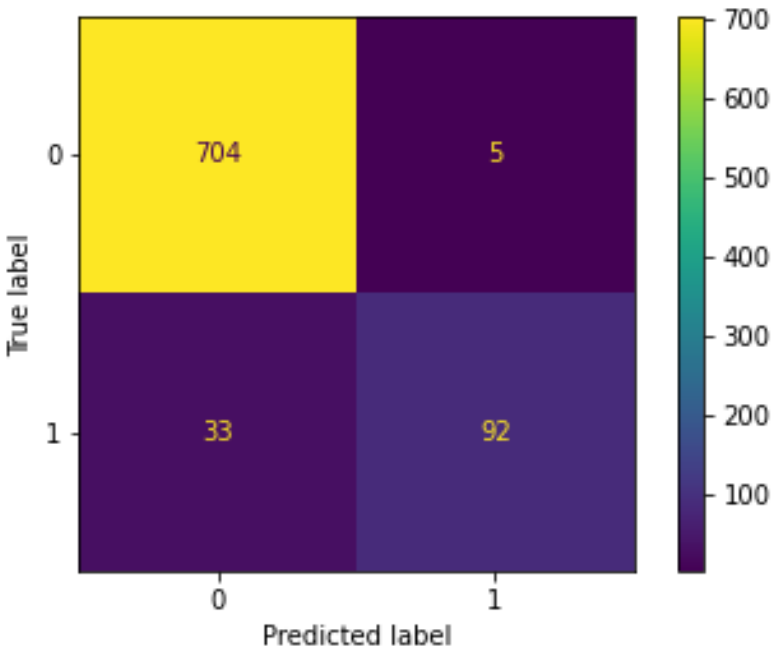
The evaluation metrics for final, XGBoost model:

	f1-score	recall	precision	accuracy
Test	0.83	.74	0.95	0.95

- The final model successfully indentifies the 74% of the true churn customers. (recall)
- Among the model predicted churn customers, 95% of them are true churn customers. (precision)
- The Harmonic Mean of Precision and Recall (f1-score) is 83%.

('churn': activity of customers leaving the company and discarding the services offered)

The confusion matrix for final, XGBoost model:



- Final model identification statistics on test data:
  - Number of true positives: 92
  - Number of true negatives: 704

    - Number of false positives: 5
    - Number of false negatives: 33
- The final model identifies 92 out of 125 churn customers correctly (74% recall).
- 92 out of 97 predicted churn customers are real churn (95% precision).

Future Work

- Improve the XGBT model performance with more detailed parameter tuning
    - Search each parameter seperately to undestand the effect on performance
    - Obtain a more sensitive range for each parameter to be used in grid search
    - Study the effect of other hyperparameters
- Study the parameter tuning with different scoring?
    - Try 'recall' metric for tuning. Will it decrease the precision significantly?
    - Maybe use multiparameter, recall and f1-score for tuning?

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

- **Jupyter Notebook** 100.0%