FITE7410 Financial Fraud Analytics

First Semester, 2022-2023

Assignment 2

Lee Chung Ho 3036030037

Introduction

2 supervised learning models, XGBoost and KNN, are built and compared on dataset Auto Insurance Claims Data.

Basic EDA

Referring to figure 1 in appendix, the dataset contain 1000 data entries and 40 variables, 19 in character, 2 in Date, 1 in logical and 18 in numeric type. There is a fraud label named "fraud_reported".

Figure 2 shows that 75.3% of claims are non-fraudulent and 24.7% of claims are fraudulent.

Only column "_c39" has missing values. Unknown values in the dataset are denoted as "?". For example, figure 3 shows "police_report_available" has 343 of "?" entries.

Data pre-processing

Columns "_c39", "policy_number" and "insured_zip" are removed.

Incident location is simplified as location type, such as, "Lane", "St", "Ridge", etc.

Categorical features are target-encoded based on fraud mean. With target encoding, categorical values are replaced with the aggregation of the target variable.

Date variables are turned into number of days.

The dataset is then split into training and testing set with 7:3 ratio. And SMOTE is performed on the training set. Figure 4 shows that fraud class in training set is balanced after SMOTE, and testing set remains imbalanced.

eXtreme Gradient Boosting (XGBoost)

The train set resampled with SMOTE is then put into an XGBoost model. Basic parameters are set and 5-fold validation is used for training. Number of trees is first set to from 1 to 200 with early stop. Figure 5 shows the best iteration is at 20th round since the validation accuracy has not improved since 20th round. Figure 6 is an graphical representation of training and validation error against number of iteration.

Using number of trees=20, an XGBoost model is trained and tested. Figure 7 shows the training and testing error over 20 rounds. Figure 8 shows the training and testing result by confusion matrix. The training AUC is 0.997 and testing AUC is 0.854, indicating a slight over-fit.

Figure 9 shows the feature importance in XGBoost model. The top 3 important features are those encoded based on "incident severity", "insured hobbies", and "auto model".

Parameter tuning is then introduced. Parameters, including booster, max_depth, min_child_weight, subsample and colsample_bytree, are tuned based on random search. 5-fold validation is used and search metric is based on training loss. Tuned parameters are shown in figure 10.

Figure 11 shows the test result with best tuned parameters. There is only a very slight increase in AUC.

K-Nearest Neighbor (KNN)

The train set resampled with SMOTE is then put into an KNN model. Initially, a model with k=1 is trained and tested. The test result is shown in figure 12 and the AUC is low as 0.706.

Then, by grid search from k=1 to 20, with a 5-fold cross validation and AUC as validation metric, figure 13 shows that the best number of neighbors is 3 for the dataset. Figure 14 shows KNN train and test result with the best tuned k. The test AUC increases to 0.750.

Model comparison

The test results of XGBoost and KNN are put together in figure 15 for comparison. The two models has similar Recall, 0.866(XGB) and 0.881(KNN), which means they both detected more than 86% of fraud cases correctly.

However, KNN predicted more non-fraud cases to be fraud, resulting in a precision 0.401(KNN) lower than 0.644(XGB), thus a lower F1 score 0.551(KNN) than 0.739(KNN).

ROC curve of both models are plotted in figure 16. The area under curve of KNN is 0.750, also lower than that of XGB 0.864. This means the overall balanced accuracy of XGB is higher.

The performance indicators above indicate that the XGBoost model is better than the KNN model.

Model suggestion

From the dataset, we can obtain the mean of total_claim_amount of all fraudulent claims as $60302.

The average salary of a Insurance Fraud Investigator is $50564 per year.[1] There is only 30 days for insurer to investigate a case.[2] Assume 1 investigator can handle a case in 1 month. Cost of investigation per case is $4214. Cost of investigation could also be time cost, investigation equipment fee, investigation team management fee, etc. But for simple estimation, only salary of the investigator is measured.

Using the above numbers, a cost matrix can be constructed as figure 17. The estimated cost of XGBoost model is $921949 and that of KNN is $1101826.

The cost of XGBoost is lower than KNN by 16.3%. The XGBoost model is more suitable for this case as it would save the insurance company more money considered cost of fraud detection and investigation.

Reference

[1] ZipRecuiter, Insurance Fraud Investigator Salary, https://www.ziprecruiter.com/Salaries/Insurance-Fraud-Investigator-Salary

[2] Coalition Against Insurance Fraud, https://insurancefraud.org/current_legislation/claim-investigation-time-limit/

Appendix

```
> skim(hw2)
── Data Summary ────────────────────────
                           Values
Name                       hw2
Number of rows             1000
Number of columns          40
_______________________
Column type frequency:
  character                19
  Date                     2
  logical                  1
  numeric                  18
________________________
Group variables            None
```

Figure 1 Broad overview of the dataset

```
> table(hw2$fraud_reported)

  N   Y 
753 247 
> round(prop.table(table(hw2$

    N     Y 
0.753 0.247 
```

Figure 2 Fraud ratio

```
> colSums(is.na(hw2))
         months_as_customer                         age               policy_number           policy_bind_date
                          0                           0                           0                          0
               policy_state                  policy_csl           policy_deductable      policy_annual_premium
                          0                           0                           0                          0
             umbrella_limit                 insured_zip                 insured_sex    insured_education_level
                          0                           0                           0                          0
         insured_occupation             insured_hobbies        insured_relationship              capital-gains
                          0                           0                           0                          0
               capital-loss               incident_date               incident_type             collision_type
                          0                           0                           0                          0
          incident_severity       authorities_contacted              incident_state              incident_city
                          0                           0                           0                          0
          incident_location    incident_hour_of_the_day number_of_vehicles_involved            property_damage
                          0                           0                           0                          0
            bodily_injuries                   witnesses     police_report_available         total_claim_amount
                          0                           0                           0                          0
               injury_claim              property_claim               vehicle_claim                  auto_make
                          0                           0                           0                          0
                auto_model                   auto_year              fraud_reported                       _c39
                          0                           0                           0                       1000
> a <- table(hw2$police_report_available)
> as.numeric(a[names(a)=="?"])
[1] 343
```

Figure 3 Missing and unknown values

(a)
```
> table(train$fraud_reported)

  0   1
522 180
```
(b)
```
> table(train_SMOTE$fraud_reported)

  0   1
720 540
```
(c)
```
> table(test_original$fraud_reported)

  0   1
231  67
```

Figure 4 Fraud cases in (a) original training set, (b) training set resampled by SMOTE, and (c) testing set

```
> params <- list(booster = "gbtree", objective = "binary:logistic", eta=0.3, gamma=0, max_depth=6,
 min_child_weight=1, subsample=1, colsample_bytree=1)
> xgbcv <- xgb.cv( params = params, data = dtrain, nrounds = 200, nfold = 5, showsd = T, stratified
 = T, print.every.n = 10, early.stop.round = 20, maximize = F)
[1]	train-logloss:0.495311+0.002350 test-logloss:0.520010+0.007067
Multiple eval metrics are present. Will use test_logloss for early stopping.
Will train until test_logloss hasn't improved in 20 rounds.

[11]	train-logloss:0.083437+0.005646 test-logloss:0.210523+0.017024
[21]	train-logloss:0.033870+0.002733 test-logloss:0.193862+0.023901
[31]	train-logloss:0.019886+0.001391 test-logloss:0.197251+0.030830
Stopping. Best iteration:
[20]	train-logloss:0.036167+0.002837 test-logloss:0.192601+0.023138
```

Figure 5 XGBoost training progress

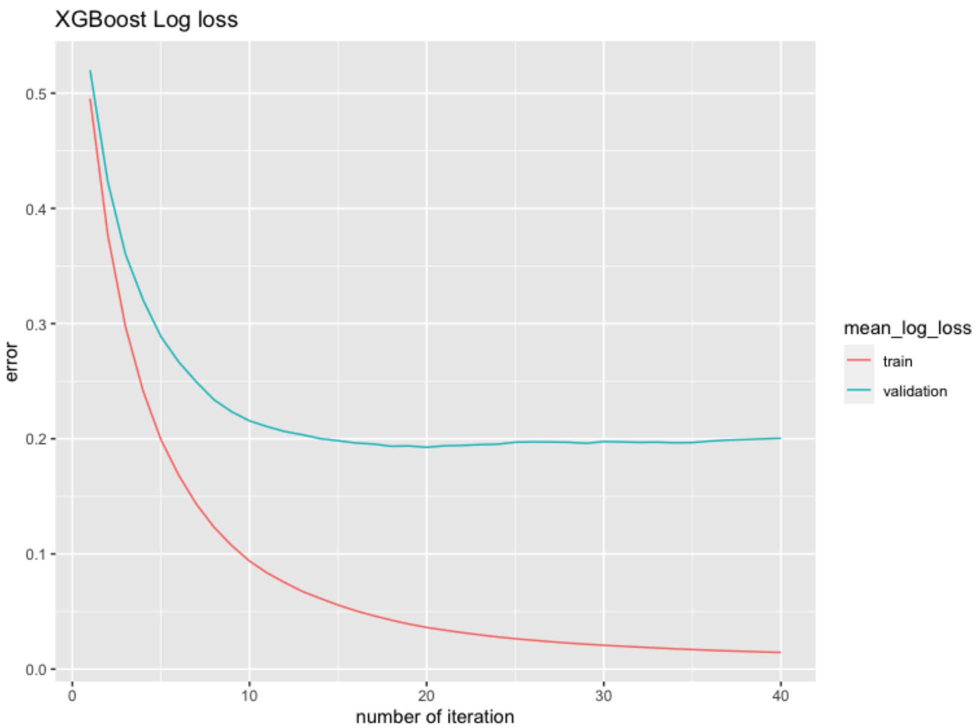

Figure 6 XGBoost Log loss during training

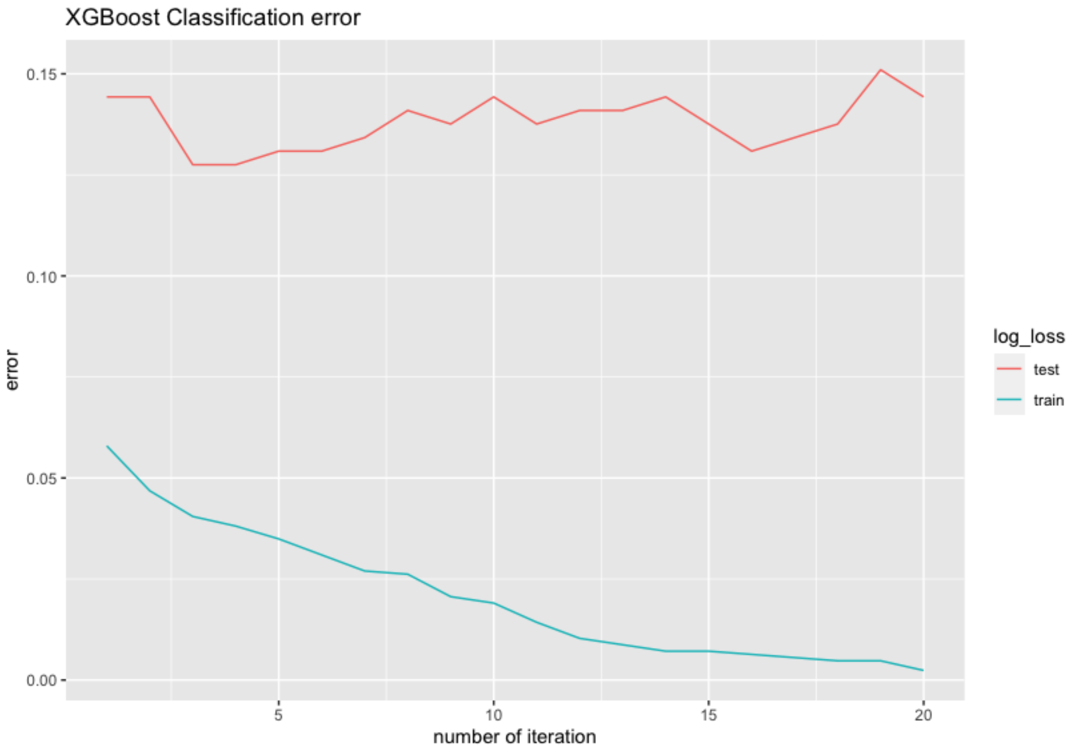

Figure 7 XGBoost classification error

(a)

```
> confusionMatrix (xgbpred_train, labels,
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 720   3
         1   0 537

               Accuracy : 0.9976
                 95% CI : (0.9931, 0.9995)
    No Information Rate : 0.5714
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9951

 Mcnemar's Test P-Value : 0.2482

            Sensitivity : 0.9944
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9959
              Precision : 1.0000
                 Recall : 0.9944
                     F1 : 0.9972
             Prevalence : 0.4286
         Detection Rate : 0.4262
   Detection Prevalence : 0.4262
      Balanced Accuracy : 0.9972

       'Positive' Class : 1
```

(b)

```
> confusionMatrix (xgbpred, ts_label, mode
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 198  10
         1  33  57

               Accuracy : 0.8557
                 95% CI : (0.8106, 0.8936)
    No Information Rate : 0.7752
    P-Value [Acc > NIR] : 0.0003233

                  Kappa : 0.631

 Mcnemar's Test P-Value : 0.0007937

            Sensitivity : 0.8507
            Specificity : 0.8571
         Pos Pred Value : 0.6333
         Neg Pred Value : 0.9519
              Precision : 0.6333
                 Recall : 0.8507
                     F1 : 0.7261
             Prevalence : 0.2248
         Detection Rate : 0.1913
   Detection Prevalence : 0.3020
      Balanced Accuracy : 0.8539

       'Positive' Class : 1
```

Figure 8 XGBoost (a) training and (b) testing results

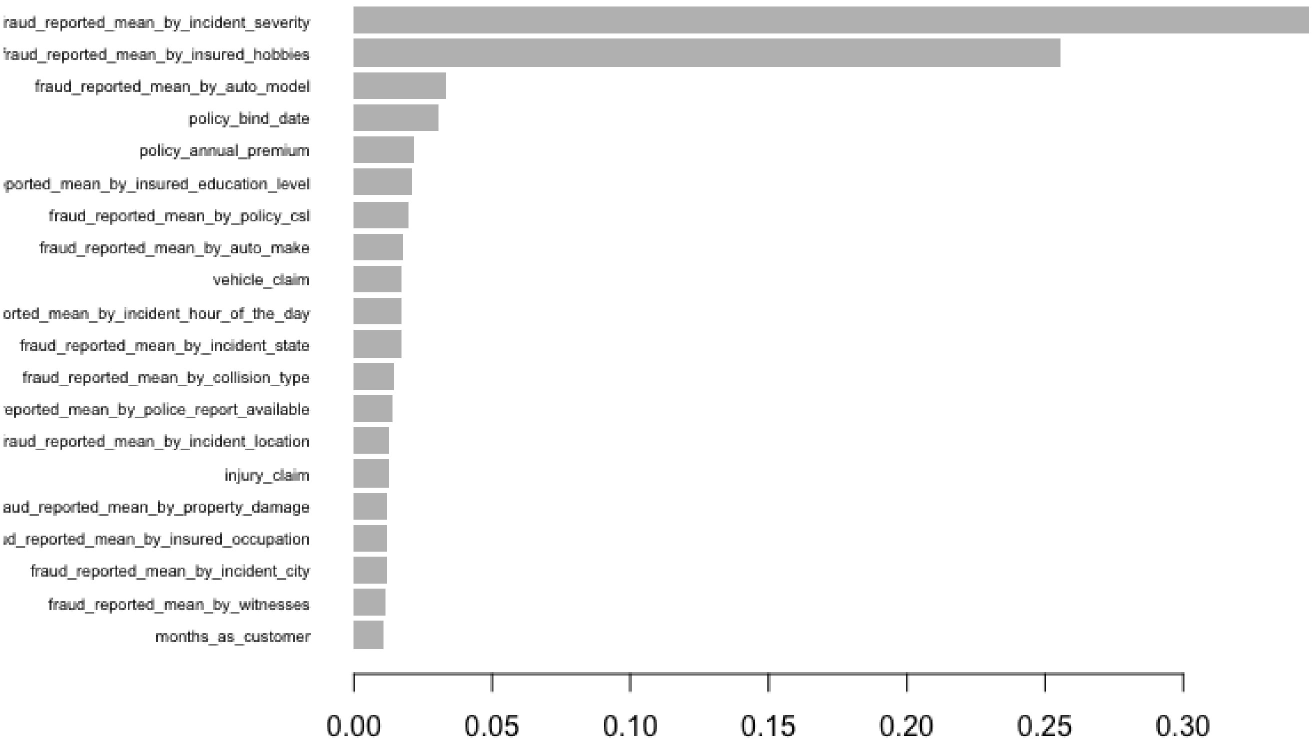

Figure 9 XGBoost model feature importance

```
> data.frame(mytune$x)
  booster max_depth min_child_weight subsample colsample_bytree
1  gbtree        29         3.785543 0.9873797        0.6272187
```

Figure 10 XGBoost tuned parameters

```
> confusionMatrix(xgpred$data$response,xgpred$data$truth,
="1")
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 199   9
         1  32  58

               Accuracy : 0.8624
                 95% CI : (0.818, 0.8994)
    No Information Rate : 0.7752
    P-Value [Acc > NIR] : 9.896e-05

                  Kappa : 0.6482

 Mcnemar's Test P-Value : 0.0005908

            Sensitivity : 0.8657
            Specificity : 0.8615
         Pos Pred Value : 0.6444
         Neg Pred Value : 0.9567
              Precision : 0.6444
                 Recall : 0.8657
                     F1 : 0.7389
             Prevalence : 0.2248
         Detection Rate : 0.1946
   Detection Prevalence : 0.3020
      Balanced Accuracy : 0.8636

       'Positive' Class : 1
```

Figure 11 XGBoost test result with best tuned parameters

```
> cm <- confusionMatrix(classifier_knn,y_test, mo
> cm
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 126   9
         1 105  58

               Accuracy : 0.6174
                 95% CI : (0.5596, 0.6729)
    No Information Rate : 0.7752
    P-Value [Acc > NIR] : 1

                  Kappa : 0.2725

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.8657
            Specificity : 0.5455
         Pos Pred Value : 0.3558
         Neg Pred Value : 0.9333
              Precision : 0.3558
                 Recall : 0.8657
                     F1 : 0.5043
             Prevalence : 0.2248
         Detection Rate : 0.1946
   Detection Prevalence : 0.5470
      Balanced Accuracy : 0.7056

       'Positive' Class : 1
```

Figure 12 KNN test result with k=1

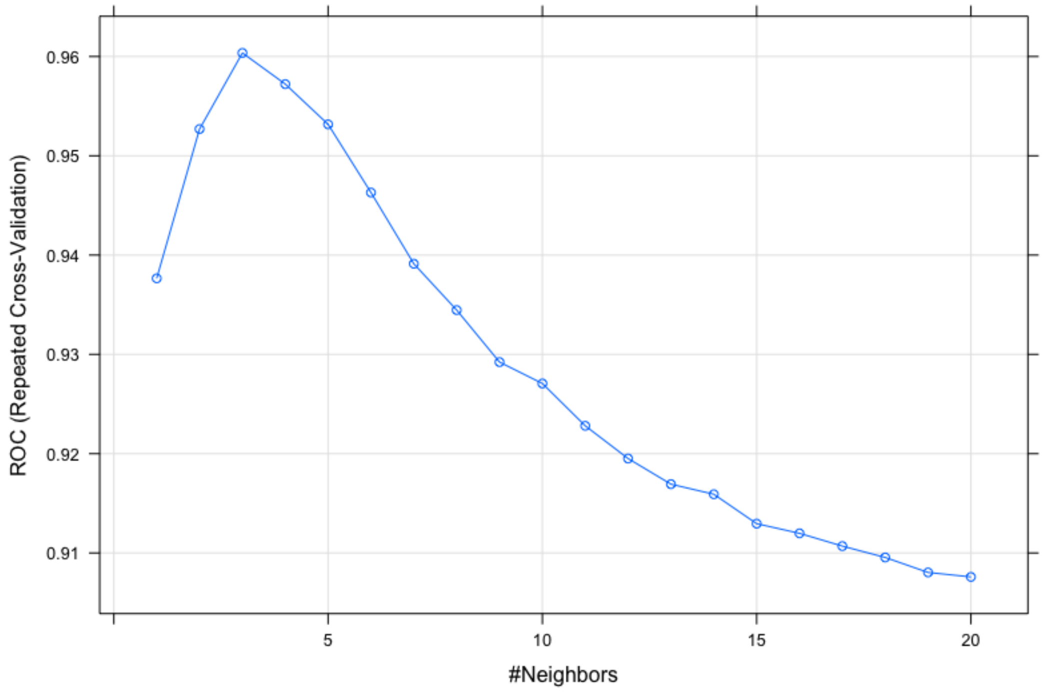

Figure 13 KNN ROC against number of neighbors

(a)

```
 confusionMatrix(as.factor(knnPredict_train),
e="1")
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 663   2
         1  57 538

               Accuracy : 0.9532
                 95% CI : (0.94, 0.9642)
    No Information Rate : 0.5714
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9056

 Mcnemar's Test P-Value : 2.062e-12

            Sensitivity : 0.9963
            Specificity : 0.9208
         Pos Pred Value : 0.9042
         Neg Pred Value : 0.9970
              Precision : 0.9042
                 Recall : 0.9963
                     F1 : 0.9480
             Prevalence : 0.4286
         Detection Rate : 0.4270
   Detection Prevalence : 0.4722
      Balanced Accuracy : 0.9586

       'Positive' Class : 1
```

(b)

```
> confusionMatrix(as.factor(knnPredict),as.
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 143   8
         1  88  59

               Accuracy : 0.6779
                 95% CI : (0.6215, 0.7306)
    No Information Rate : 0.7752
    P-Value [Acc > NIR] : 1

                  Kappa : 0.3509

 Mcnemar's Test P-Value : 7.45e-16

            Sensitivity : 0.8806
            Specificity : 0.6190
         Pos Pred Value : 0.4014
         Neg Pred Value : 0.9470
              Precision : 0.4014
                 Recall : 0.8806
                     F1 : 0.5514
             Prevalence : 0.2248
         Detection Rate : 0.1980
   Detection Prevalence : 0.4933
      Balanced Accuracy : 0.7498

       'Positive' Class : 1
```

Figure 14 KNN (a) train and (b) test result with best tuned k

(a)

```
confusionMatrix(xgpred$data$response,xgpred$data$t
)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 199   9
         1  32  58

               Accuracy : 0.8624
                 95% CI : (0.818, 0.8994)
    No Information Rate : 0.7752
    P-Value [Acc > NIR] : 9.896e-05

                  Kappa : 0.6482

 Mcnemar's Test P-Value : 0.0005908

            Sensitivity : 0.8657
            Specificity : 0.8615
         Pos Pred Value : 0.6444
         Neg Pred Value : 0.9567
              Precision : 0.6444
                 Recall : 0.8657
                     F1 : 0.7389
             Prevalence : 0.2248
         Detection Rate : 0.1946
   Detection Prevalence : 0.3020
      Balanced Accuracy : 0.8636

       'Positive' Class : 1
```

(b)

```
> confusionMatrix(as.factor(knnPredict),as.
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 143   8
         1  88  59

               Accuracy : 0.6779
                 95% CI : (0.6215, 0.7306)
    No Information Rate : 0.7752
    P-Value [Acc > NIR] : 1

                  Kappa : 0.3509

 Mcnemar's Test P-Value : 7.45e-16

            Sensitivity : 0.8806
            Specificity : 0.6190
         Pos Pred Value : 0.4014
         Neg Pred Value : 0.9470
              Precision : 0.4014
                 Recall : 0.8806
                     F1 : 0.5514
             Prevalence : 0.2248
         Detection Rate : 0.1980
   Detection Prevalence : 0.4933
      Balanced Accuracy : 0.7498

       'Positive' Class : 1
```

Figure 15 (a) XGBoost and (b) KNN test result with best tuned parameters

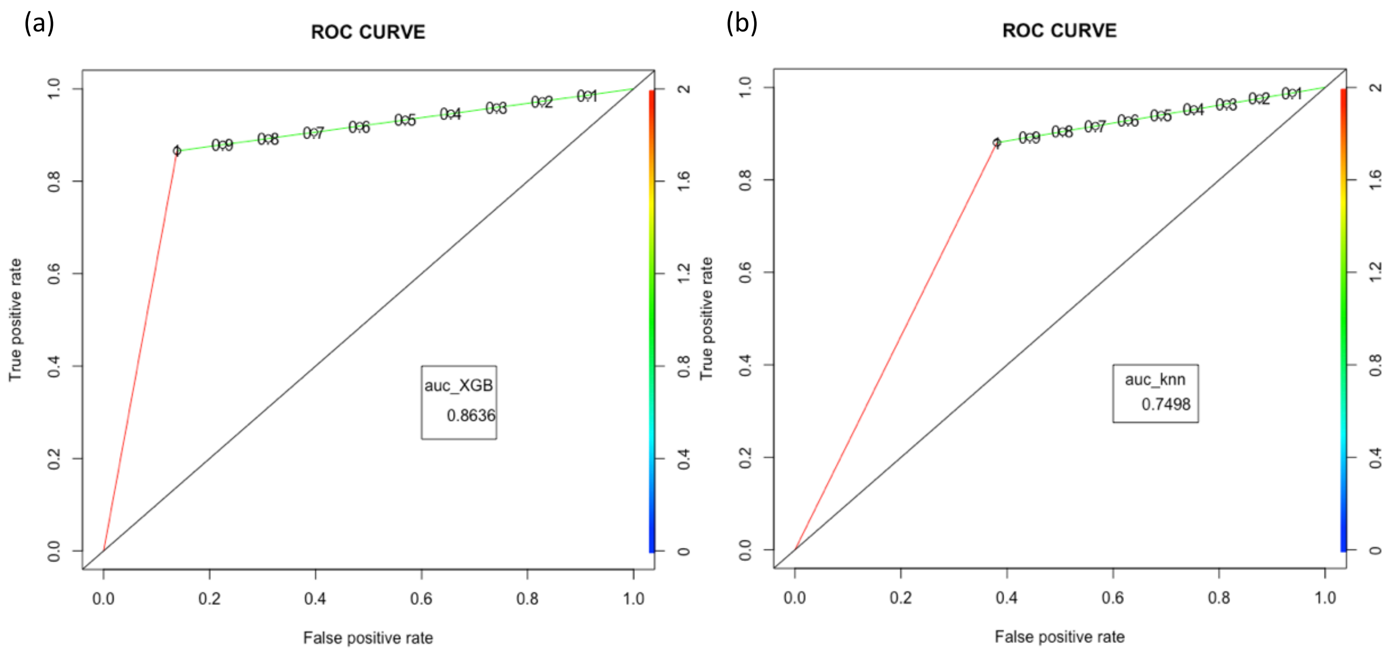

Figure 16 ROC curves of (a) XGBoost and (b) KNN test result

```
> amount <- mean(hw2_fraud$total_claim_amount)
> amount
[1] 60302.11
> investigation <- 50564/12
> investigation
[1] 4213.667
> cost_mat=matrix(c(0,investigation,amount,investigation),ncol=2)
> cost_mat
         [,1]      [,2]
[1,]    0.000 60302.105
[2,] 4213.667  4213.667
> cm_hw2_knn <- as.matrix(knn_eval[["table"]])
> cm_hw2_XGB <- as.matrix(XGB_eval[["table"]])
> XGB_cost <- sum(cost_mat*cm_hw2_XGB)
> XGB_cost
[1] 921948.9
> knn_cost <- sum(cost_mat*cm_hw2_knn)
> knn_cost
[1] 1101826
> (XGB_cost-knn_cost)/knn_cost
[1] -0.1632535
```

Figure 17 Cost estimation based on fraud claim amount and investigation cost