

**Credit Card Fraud Detection**

Data Science Workflow and Applications

Prof. Patrick Erdelt

Prepared by: Yaqoob David (s914527)

\*Note: each section of this report is related to one attached RapidMiner file.

Introduction to the Data:

The data contains credit card transactions over a period of time, and the aim is to determine the fraudulent transactions so that the customers are not charged for items they did not purchase.

The data has 227,846 transaction examples, including 394 fraudulent ones.

There are 28 numerical attributes (v1 to v28), the 'time' attribute refers to the time between the current transaction and the first one, the 'amount' refers to the transaction amount, the 'class' is the label attribute which takes '1' in case of fraud transection and '0' otherwise.

The task is a classification task, which should determine the future fraudulent transactions depending on the given historical data. A detailed look at the data is imbalanced (the fraud is 0.173% of the total).

Task 1: Basic Workflow (file: Final Exam import 1)

In this initial stage of workflow:

* We import the data from the .csv file to RapidMiner, by storing them in a local repository.
* Change the type of class attribute from integer to binomial, and then substitute the values: 0 to ‘correct’, 1 to ‘fraud’.
* Split the data into 30% test and 70% train using stratified sampling method, ensuring that the class distribution in the train and test set is the same as in the whole data set. We keep the test dataset to the end.

In the screenshot attached below, the process is provided for your reference.

Diagram

Description automatically generated

Figure 1 Preparation of Dataset

Task 1: Modeling (file: Final Exam basic model import)

\*Note: in this basic modelling, it took about one hour to train the data using KNN (k=5) for one-fold; therefore, my CPU did not help to do cross-validation on KNN in this stage.

Now let us take an overview of the behaviour of the data and apply three classification algorithms without any refinement.

The training and test data are obtained by executing the file Exam-1-import.

The model is trained by KNN (k=5), decision tree, and Naïve Bayes (cross-valid-fold=10).

The first observation is that KNN is much slower than others.

The accuracy is more than 99% for KNN and decision tree and about 97% for Naïve Bayes (validation and test data), which is suitable for all. Still, by looking closely, we see the recall of positive class (fraud) is only 5.08% in KNN increased to 67.67% in the decision tree (validation) and 76.27% decision tree (test), and 82.25% Naïve Bayes (validation) and 84.75% in naïve Bayes.

The recall of fraud or specificity (TNR) represents how often the fraud is predicted if the example is actual fraud. The complement of the error-type-1 (FPR)=1-TNR, which means the higher TNR is, the lower the probability of missing a fraudulent transaction. As missing the fraud is worse than checking a correct example ad fraud, the focus will be on TNR to be higher during the modelling.

This vast difference between the total accuracy and the recall of the fraud is resulted primarily because of the imbalance in the data. This basic modelling gives the first impression that the naïve Bayes could be the best algorithm for modelling.

Diagram

Description automatically generated

Figure 2 Modeling 1

In the screenshot attached above and below, the process is provided for your reference.

Diagram, schematic

Description automatically generated

Figure 3 Modeling 2

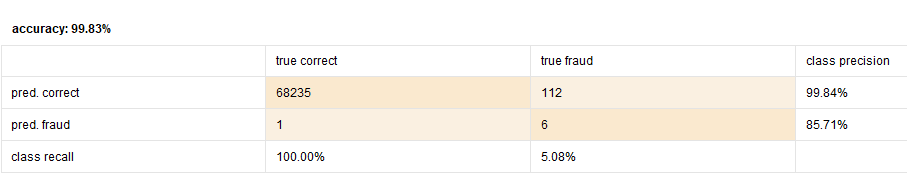


Figure 4 accuracy KNN (test)

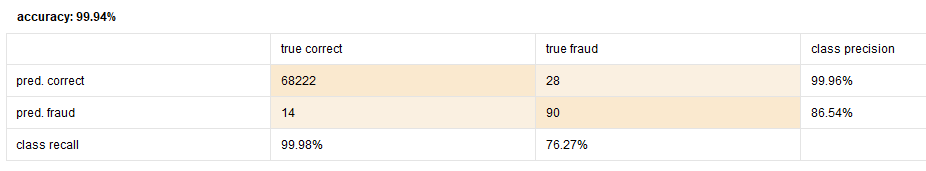


Figure 5 accuracy decision tree (test)

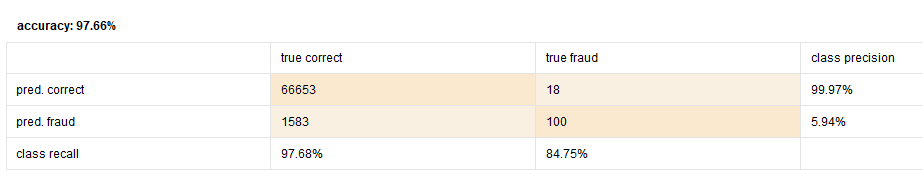


Figure 6 accuracy naive Bayes (test)

We can assume that the decision tree's performance is much better than other algorithms and good precision and sufficient recall numbers by looking at the results.

Task 2: Data Preparation (file: Final Exam Data preparation 1)

In this stage, the data is prepared for modelling and refinement:

* The unprocessed training and test data are obtained by executing the file Exam-1-import (section 1).
* By analyzing the data, there are no missing values
* Any duplicated examples in the training set are checked and removed.
* The training set is normalized (transformed to get variables with mean=0 and standard deviation std=1), a dimension reduction method (PCA) is applied (to keep 95% of the total variance, 27 attributes have remained in the output of the PCA, which gives the impression that the original data (numerical ones) is almost uncorrelated and might be the output of another PCA operation).
* The normalized and PCA models are grouped to be applied on the test set later to keep the exact dimensions.
* Finally, the preprocessed training and test data are stored in the local repository to save preprocessing time each time we run the model.

Diagram

Description automatically generated with medium confidence

Figure 7 Normalization & PCA

Task 2: Hyperparameter Optimization**: (file: Final Exam Hyper Parameter Optimization)**

The stored preprocessed training and test data set from the previous section avoid imbalance classes mentioned before. The classes (correct, fraud) are to be balanced using sample operation.

Diagram

Description automatically generated

Figure 8 Hyperparameter Optimization 1

Diagram, schematic

Description automatically generated

Figure 9 Hyper Parameter Optimization 2

* Decision Tree: the number of folds in cross-validation and the max depth of the tree are optimized using optimize parameter operation. The best result is by using cross-validation of 20 folds and tree with max-depth=4.

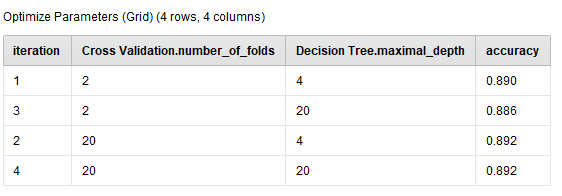


Figure 10 optimized parameters (decision tree)

Looking at the performance of the model on the test data, we obtain an accuracy of 96.15% with 88.98% recall of the fraud class (TNR), which is much better than the one in basic modelling, even though that the precision of the fraud is much lower (3.86%), but as mentioned above the actual fraud is more (to some extend).

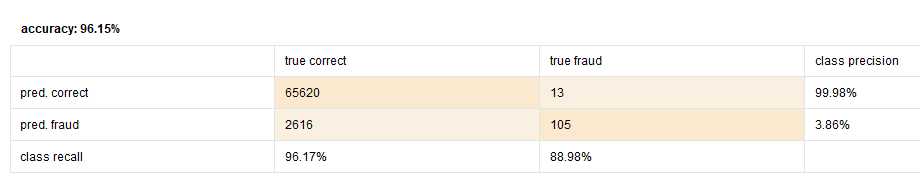


Figure 11 accuracy (decision tree)

Chart, waterfall chart

Description automatically generated

Figure 12 the Decision Tree (with optimized parameters)

* Naive Bayes: with optimizing the number of folds in the cross-validation, the accuracy of the test data is 94.01%, with 88.98% recall of the fraud.

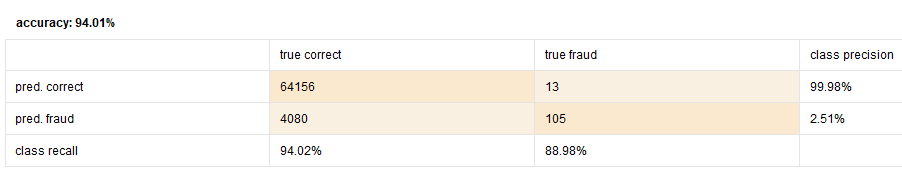


Figure 13 accuracy (Naive Bayes)

* KNN: the number of folds is optimized between 2 and 12-fold, and k of the KNN model is optimized between 2 and 10, getting the best k=10 and cross-Val-fold=12.

Table

Description automatically generated with medium confidence

Figure 14 optimize k and n-fold of KNN.

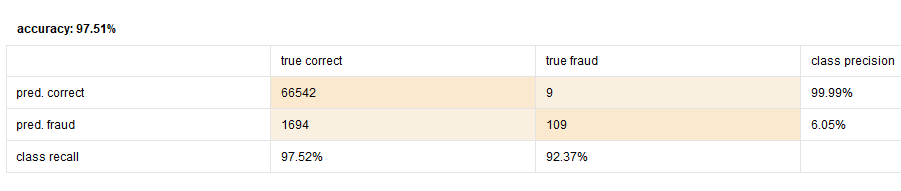


Figure 15 Accuracy-KNN (optimized K, cross-val-fold)

Task 2: Ensemble (Voting) (file: Final Exam Ensemble)

In this section, the voting ensemble technique is used to help choose the best model of the three.

In voting, to classify an example, each classifier has one vote, and the example is classified as the class having the highest votes.

Using this technique with optimizing the hyperparameters (cross-valid-folds, k in KNN, max-depth, cross-val-fold in decision tree, and cross-val-fold in Naïve Bayes), an accuracy of 96.32% is obtained. The recall of the fraud is 88.98% which is less than the recall of fraud in optimized KNN, which means that voting does not give a better result.

Graphical user interface, diagram

Description automatically generated

Figure 16 Ensemble (voting) 1

Diagram

Description automatically generated

Figure 17 Ensemble (Voting) 2



Figure 18 Accuracy (using vote)

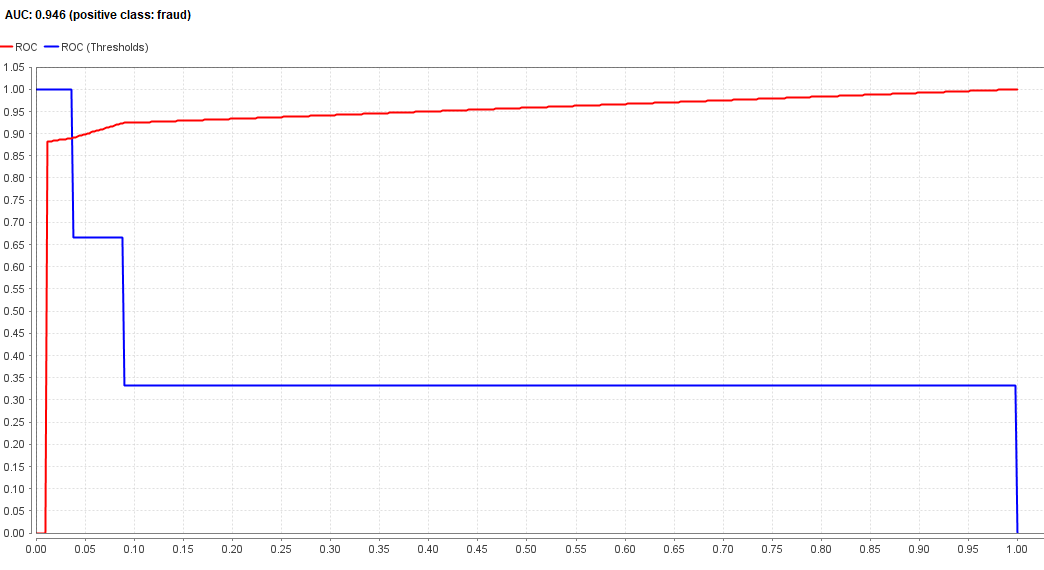


Figure 19 AUC- (using vote)

Task 2: Final Model: (file: Exam-1-model.rmp)

Before discussing the result, another ensemble technique is used, which is Bagging.

Bagging depends on bootstrapping technique where the training set for every classifier comes from a common training set by sampling and replacement. Then all the results are aggregated for Bagging.

The bagging technique is applied to the three models. The following hyperparameters are fixed to their best values found so far, and the parameter optimization is applied to the sample ratio of the bagging operator.

Diagram, schematic

Description automatically generated

Figure 20 Final Model 1

Graphical user interface, diagram, application

Description automatically generated

Figure 21 Final Model 2

The fixed parameters are:

* KNN: k = 10, cross-valid-fold = 12
* Decision Tree: max depth = 4.

By applying this final design on the test data, the result is:

* Naïve Bayes: Accuracy= 93.94% with recall of fraud=88.98%

AUC =0.956, and ROC moved up closer to the perfection

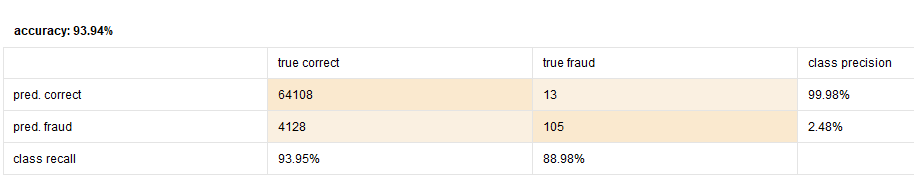


Figure 22 Accuracy-Naive Bayes-final model

* Decision Tree: Accuracy =94.94% with the recall of fraud 90.68%, AUC=0.976, and ROC gets closer to the top left corner, which means the imbalance of cost gets reduced by using Bagging.

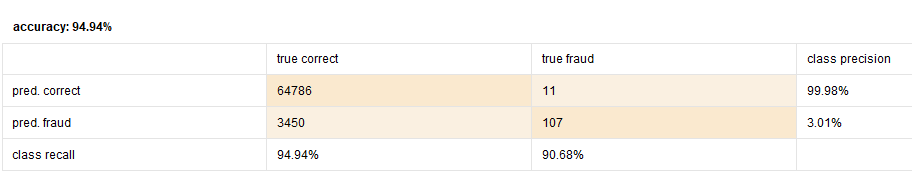


Figure 23 Accuracy-Decision Tree- final model

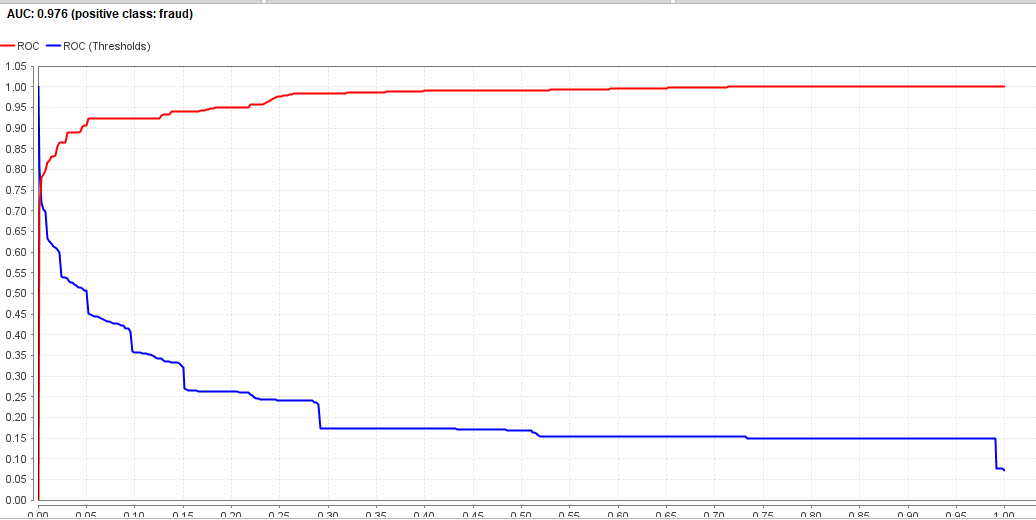


Figure 24 AUC- Decision Tree- Final model

* KNN: Accuracy is 98.17% with recall of fraud = 92.37%,

The error-type-1 (FPR) = 1 - 92.37% = 7.63%

The error-type-2 (FNR) = 1 - 98.18% = 1.82%

This means the two types of errors get much smaller, and the cost of imbalance is reasonably reduced.

AUC= 0.983, and the ROC is almost perfectly good

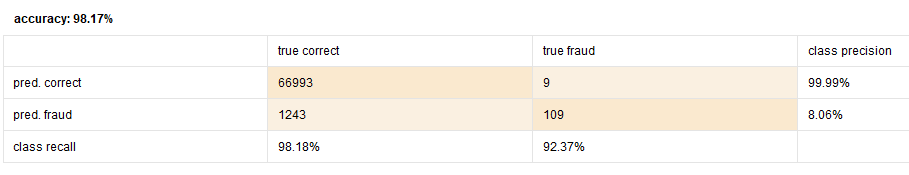


Figure 25 Accuracy-KNN-final model

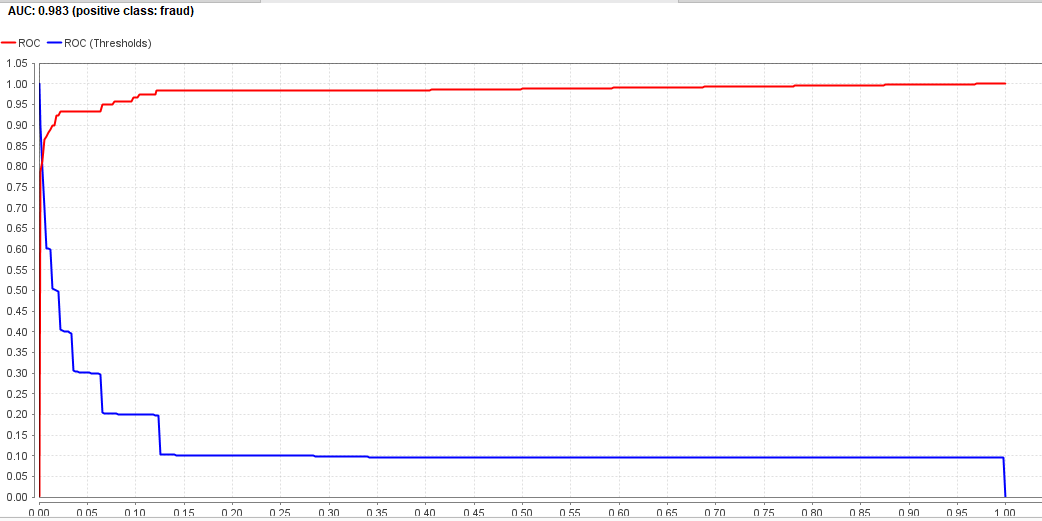


Figure 26 AUC- KNN- final model

The KNN model with these final configurations has the best performance and give the best prediction of the fraud even though it is not highly accurate in predicting the correct transaction as many correct transactions may be considered as a fraud.

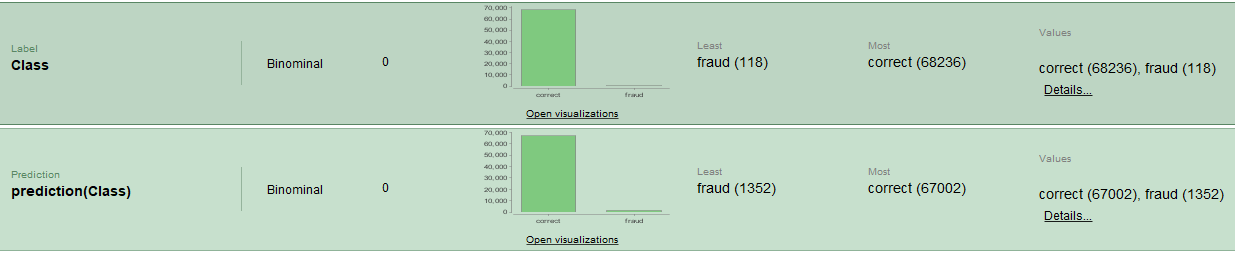


Figure 27 Statistics- KNN- test data

The predicted data (using KNN) show that there are 118 fraudulent transactions in the original test data, and 1352 transactions are predicted as fraud. This is resulted because of the imbalance in the cost between the two classes. Still, we accept that detecting the actual fraud is more important than determining a correct transaction as fraud.