STATS 3ST3

Winter 2021

Machine Learning in Severity Prediction

Anyi Fu	Boyang Wei	Xiwen Cui	Zhiming Zhang
400136866	400104356	400144427	400135152

April 2021

Contents

1 Introduction

Machine learning brings major changes to the way risks, claims, and customer experiences are handled by insurance companies. These companies are able to work with large amounts of data and identify important insights of the data thanks to machine learning [2]. In this report, we will be adopting machine learning methods on two datasets provided by The Co-operators General Insurance Company to predict auto loss severity.

1.1 Background

The Co-operators is a leading Canadian multi-line insurance and financial services co-operative with $47.3 billion in assets under administration [1]. Its companies provide services in four main areas: property and casualty (PC) insurance, life insurance, institutional asset management and brokerage operations [1]. The datasets that will be used in later parts of this report are taken from the final round of the Second McMaster & Co-operators Problem Solving e-Workshop, organized by the Department of Mathematics & Statistics at McMaster University and The Co-operators General Insurance Company. It consists 16995 profiles and 80 characteristics of the insureds. The main coverage used in these datasets are the auto Accident Benefits (AB) and the Third Party Liability (TPL).

1.2 Supervised Learning and Classification

Supervised learning, also know as supervised machine learning, is a subcategory of machine learning and artificial intelligence [4]. It is defined by its use of labeled datasets to train algorithms to classify data or predict outcomes [4]. There are two labeled dataset that is used in the process of supervised learning: a training set and a testing set. The training set, used to train a model, contains massive amount of inputs and its corresponding correct outputs. The testing set, used to test the trained model, contains input and output pairs that are independent to the training set. Ultimately, Our goal is to use this training set to teach our algorithms to yield an output with an acceptable correctness given a new input from the testing set [3].

Classification is a supervised learning concept, used to draw conclusions on how an entity should be labelled [4]. In this report, we will predict loss severity by labeling whether a profile is a big loss or not. We will achieve this by using two most common classification methods: K-Nearest Neighbors and Random Forest.

2 Algorithms and Model Fitting

2.1 k-Nearest Neighbors

The K-Nearest Neighbors (k-NN) algorithm is a simple implementation of supervised learning that can be used to solve both classification and regression problems. The k-NN algorithm takes on the moral that similar things are close to each other, as the famous saying says, ”Birds of a feather flock together”.

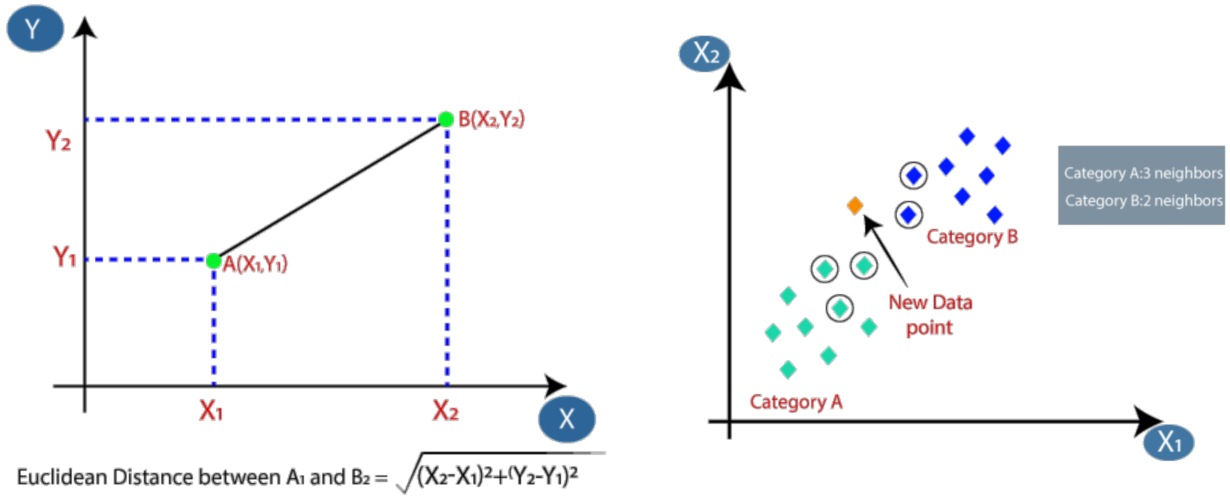


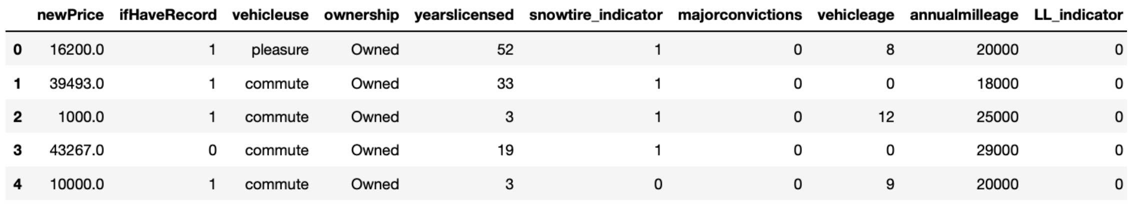
Figure 1: simple k-NN classification [7]

The most popular way to calculate the similarity or closeness between data points is by using Euclidean distance as shown above. Aside from being able to quantify the closeness, we also need some procedures to finally implement k-NN: [5]

1. Initialize k number of nearest neighbors by running through each possible value for k and choose the optimal one with maximum accuracy or minimum error.
2. For each input value:
   i Calculate the Euclidean distance between the current data and its query data point.
   ii Add the result and the index of the current data to an ordered list.
3. Sort the ordered list consisting all distances and indices in ascending order by distances, and choose the first K data point from this sorted list.
4. The input values will be assigned to the category that is most common among the k nearest neighbors.

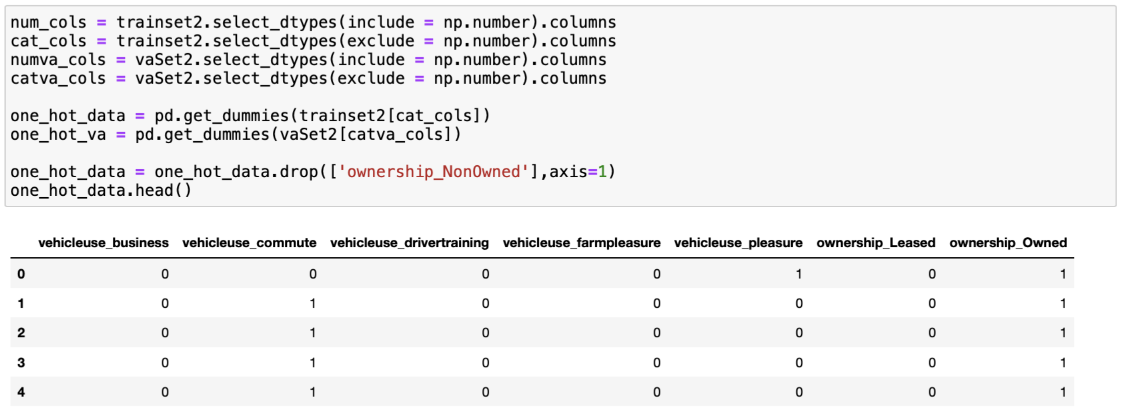
2.1.1 Application in Python

Owing to the fact that our dataset contains too many variables, we choose to drop variables that are relatively unimportant to predicting a loss severity and columns that contain too many null values. Among the remaining variables, some of them are categorical variables that take on the form of strings, as we can see in Figure 2.



	newPrice	ifHaveRecord	vehicleuse	ownership	yearslicensed	snowtire_indicator	majorconvictions	vehicleage	annualmilleage	LL_indicator
0	16200.0	1	pleasure	Owned	52	1	0	8	20000	0
1	39493.0	1	commute	Owned	33	1	0	0	18000	0
2	1000.0	1	commute	Owned	3	1	0	12	25000	0
3	43267.0	0	commute	Owned	19	1	0	0	29000	0
4	10000.0	1	commute	Owned	3	0	0	9	20000	0

Figure 2: First five rows of training set



```
num_cols = trainset2.select_dtypes(include = np.number).columns
cat_cols = trainset2.select_dtypes(exclude = np.number).columns
numva_cols = vaSet2.select_dtypes(include = np.number).columns
catva_cols = vaSet2.select_dtypes(exclude = np.number).columns

one_hot_data = pd.get_dummies(trainset2[cat_cols])
one_hot_va = pd.get_dummies(vaSet2[catva_cols])

one_hot_data = one_hot_data.drop(['ownership_NonOwned'],axis=1)
one_hot_data.head()
```

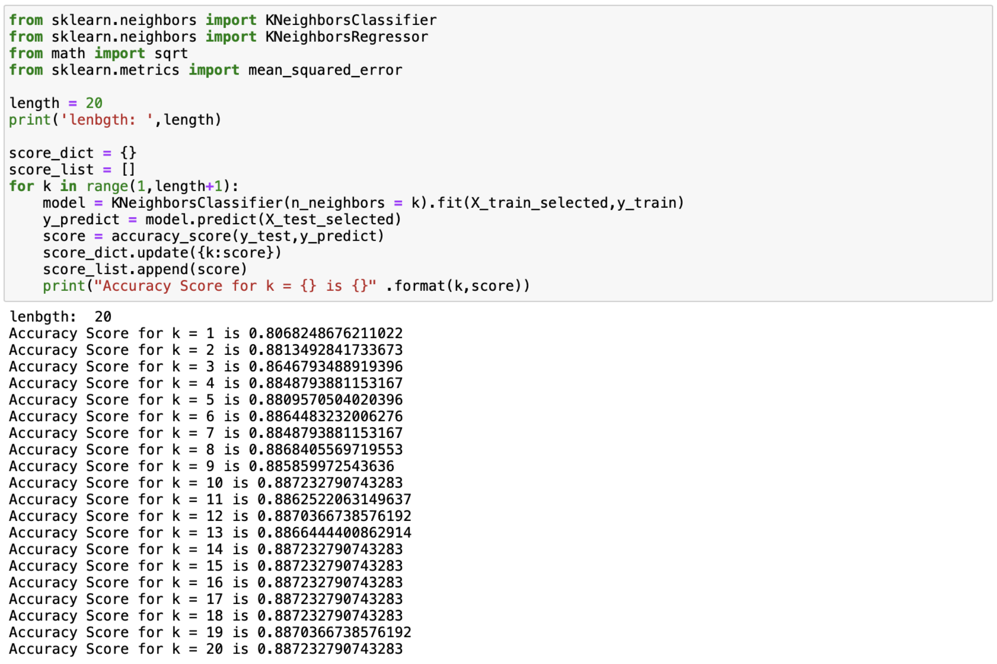
	vehicleuse_business	vehicleuse_commute	vehicleuse_drivertraining	vehicleuse_farmpleasure	vehicleuse_pleasure	ownership_Leased	ownership_Owned
0	0	0	0	0	1	0	1
1	0	1	0	0	0	0	1
2	0	1	0	0	0	0	1
3	0	1	0	0	0	0	1
4	0	1	0	0	0	0	1

Figure 3: New columns generated after one-hot encoding

In order to analyze these categorical variables mathematically, we first need to convert these categorical variable into numerical variable with binary values using one-hot encoding. As an illustration, because we know that the variable 'vehicleuse' takes on five categorical values: "pleasure", "business", "commute", "drivertraining", and "farmpleasure", we can create five new columns namely 'vehicleuse_pleasure', 'vehicleuse_business', 'vehicleuse_commute', 'vehicleuse_drivertraining', and 'vehicleuse_farmpleasure' to hold an indicator of 1 or 0 representing a yes or no to that subcategory. For example, the initial vehicle use for the first insured at row 0 is pleasure, so as illustrated in Figure 3, the new corresponding value for 'vehicleuse_pleasure' is 1, and all the other subcategories are 0.

In addition to modifying the dataset, we decide to split 70% of the dataset to use it as the train set and use the remaining 30% as the test set. Particularly, we split this dataset into X_train, X_test, y_train, y_test using train_test_split() function. The reason for this decision is that we can use X_train and y_train to train the model and compare the predictions to y_test to check the model's performance. we

After perfecting our datasets, we turn to implementing the k-NN algorithm. To make things easy, we simply import *KNeighborsClassifier* from *sklearn.neighbors package* instead of implement k-NN algorithm from scratch. To find parameter k, we write a function, shown in Figure 4, that computes each accuracy score for k from 1 to 20, and choose the one that is the most appropriate.



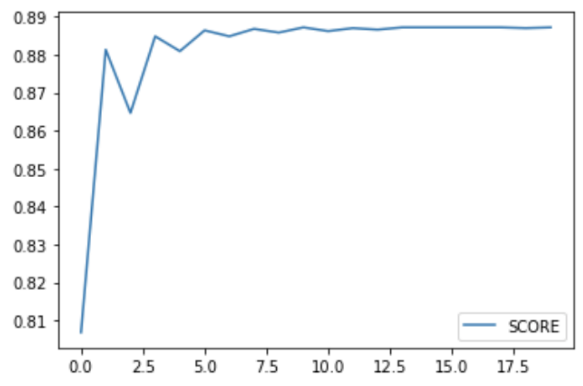
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neighbors import KNeighborsRegressor
from math import sqrt
from sklearn.metrics import mean_squared_error

length = 20
print('lenbgth: ',length)

score_dict = {}
score_list = []
for k in range(1,length+1):
    model = KNeighborsClassifier(n_neighbors = k).fit(X_train_selected,y_train)
    y_predict = model.predict(X_test_selected)
    score = accuracy_score(y_test,y_predict)
    score_dict.update({k:score})
    score_list.append(score)
    print("Accuracy Score for k = {} is {}" .format(k,score))
```

```
lenbgth:  20
Accuracy Score for k = 1 is 0.8068248676211022
Accuracy Score for k = 2 is 0.8813492841733673
Accuracy Score for k = 3 is 0.8646793488919396
Accuracy Score for k = 4 is 0.8848793881153167
Accuracy Score for k = 5 is 0.8809570504020396
Accuracy Score for k = 6 is 0.8864483232006276
Accuracy Score for k = 7 is 0.8848793881153167
Accuracy Score for k = 8 is 0.8868405569719553
Accuracy Score for k = 9 is 0.885859972543636
Accuracy Score for k = 10 is 0.887232790743283
Accuracy Score for k = 11 is 0.8862522063149637
Accuracy Score for k = 12 is 0.8870366738576192
Accuracy Score for k = 13 is 0.8866444400862914
Accuracy Score for k = 14 is 0.887232790743283
Accuracy Score for k = 15 is 0.887232790743283
Accuracy Score for k = 16 is 0.887232790743283
Accuracy Score for k = 17 is 0.887232790743283
Accuracy Score for k = 18 is 0.887232790743283
Accuracy Score for k = 19 is 0.8870366738576192
Accuracy Score for k = 20 is 0.887232790743283
```

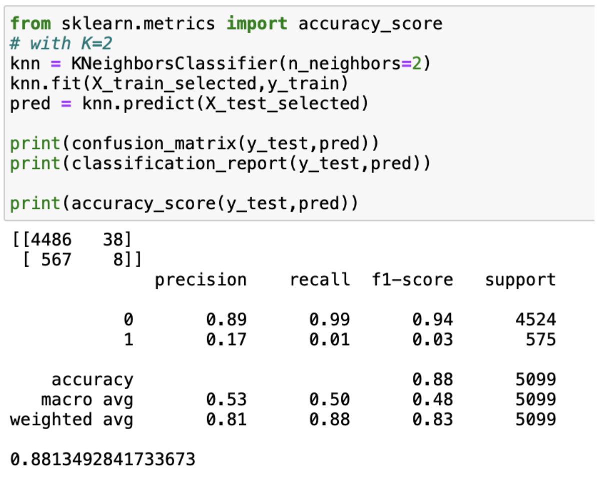
Figure 4: Find parameter k through accuracy score

2.1.2 Model Fitting and Validation

Following the computation on all accuracy scores for k from 1 to 20, we plot these scores for a better analysis.
As illustrated in the Figure 5a above, the accuracy score dips at k = 3, then raises to about 0.885 at k = 4, and then succeeding to a constant trend. The classification boundaries seem to be smoother as k becomes larger, so to prevent over-fitting, k = 2 is being considered. From there on, the next step is to fit the model and compute confusion matrix and classification report. From the



(a) Accuracy scores for k from 1 to 20



```
from sklearn.metrics import accuracy_score
# with K=2
knn = KNeighborsClassifier(n_neighbors=2)
knn.fit(X_train_selected,y_train)
pred = knn.predict(X_test_selected)

print(confusion_matrix(y_test,pred))
print(classification_report(y_test,pred))

print(accuracy_score(y_test,pred))
```

```
[[4486   38]
 [ 567    8]]
              precision    recall  f1-score   support

           0       0.89      0.99      0.94      4524
           1       0.17      0.01      0.03       575

    accuracy                           0.88      5099
   macro avg       0.53      0.50      0.48      5099
weighted avg       0.81      0.88      0.83      5099

0.8813492841733673
```

(b) Confusion matrix and classification report

Figure 5

confusion matrix and classification report shown in Figure 5b, we see that 4486 out of 4524 values are predicted correctly in the class 0 (LL indicator for a small loss), however only 8 out of 575 values are predicted correctly in the class 1(LL indicator for a big loss). On the other hand, the overall f1-score is 0.88, which is not bad.

Although the f1-score for class 0 is high, this score alone is not enough. For an insurance company, it is expected to precisely identify if an insured has potential high-severity. In this case, the accuracy in predicting class 1 insureds needs to be improved. One possible adaption is to add the dropped variables as they might contribute to the model fitting.

2.2 Random Forest

Random forest is also a widely used supervised learning algorithm due to its simple and diverse nature. This algorithm can be used for both classification and regression tasks. Therefore, it can deal with qualitative, quantitative, or a mixed type of data set.

2.2.1 Decision Tree

To understand this algorithm better, we first go over how decision trees work using an example. [12]

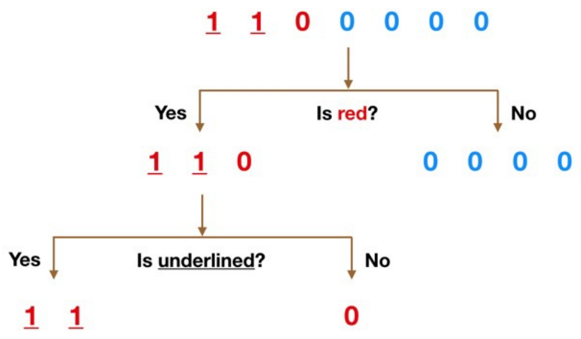


Figure 6: Decision Tree

Imagine that our dataset contains those number at the top of the decision tree illustrated in Figure 6. Notably, there are some characteristics to these numbers. Now let's say that our goal is to find out which observations exhibits the colour red and is also being underlined. We can start by checking if a given observation is red. This allows us to split all observations into two groups based on their color: red and blue. We can then check if a given observation is underlined. This allows us to further divide the two groups into two more groups each based on their underlying property: underlined and not underlined. As a result, we find that there are two observations in this decision tree that matches our goal. Ultimately, we are trying to capture all the patterns in a dataset, and this process is called the model fitting. Furthermore, we can feed new datasets to this fitted model and make predictions based on the same characteristics our training dataset exhibits. Obviously, our data won't be as clean and simple as our exmample here in real life, but the logic to a decision tree remains the same.

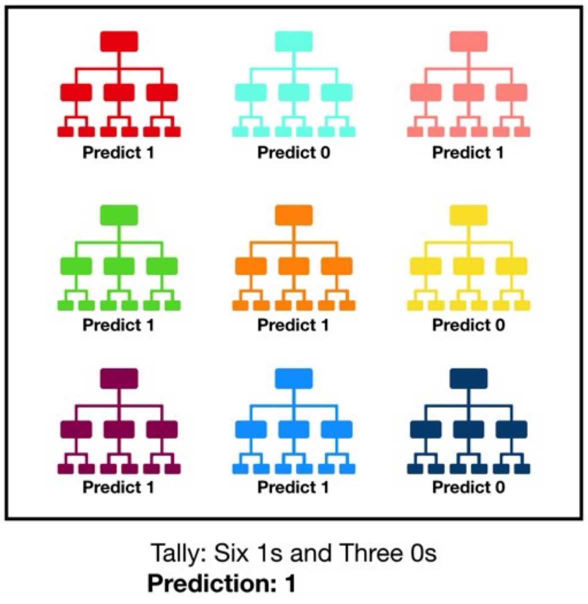


Figure 7: Random Forest

2.2.2 Random Forest

A random forest is just a collection of decision trees. Each individual tree generate a class prediction and the class with the majority votes becomes our prediction model. In the case of Figure 7, Predict 1 is our prediction model because it has majority votes. Although the voting strategy can give us a more accurate and stable prediction, we still have to keep in mind that the correlation between each model has to be low, meaning our forest has to consist of only randomly created decision trees, to maintain precision.

2.2.3 Application in R - Exploratory Data Analysis

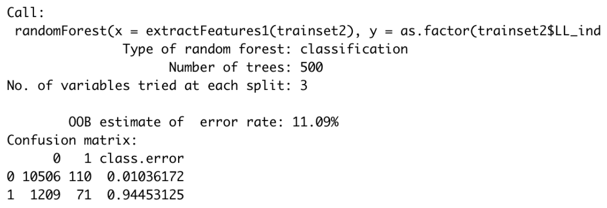
To start our analysis, we first load required packages into R, as shown in codes below. We then load our datasets into R. However, we notice that "Costnew" variable have null values that we'll need to take car of. Additionally, some of the variables we selected are categorical variables, so we decide to turn them into factor variables. This way, these variables can be implemented correctly in Random Forest modelling. Lastly, like the dataset used for k-NN, we split 70% of the dataset as the train set and use the rest 30% as the test set. Detailed coding on these modification is shown belown.

```
library(magrittr)
library(tidyverse)
library(dplyr)
library(ggplot2)
library(randomForest)

trainset = read_csv('~/Desktop/workshop day package/
McMaster Workshop Train set.csv')

# build a extract variables function
extractFeatures1 <- function(data) {
  features <- c("purchaseprice",
                "costnew",
                "vehicleuse",
                "ownership",
                "yearslicensed",
                "snowtire_indicator",
                "majorconvictions",
                "vehicleage",
                "annualmilleage")
  fea <- data[,features]
  fea$costnew[is.na(fea$costnew)] <- median(fea$costnew, na.rm=TRUE)
  fea$vehicleuse <- as.factor(fea$vehicleuse)
  fea$ownership <- as.factor(fea$ownership)
  fea$snowtire_indicator<-as.factor(fea$snowtire_indicator)
  fea$majorconvictions<-as.factor(fea$majorconvictions)
  return(fea)
}

index <- sample(nrow(trainset), 0.5∧nrow(trainset), replace = FALSE)
trainset2 <- trainset[index,]
testset2 <- trainset[-index,]
```

2.2.4 Model Estimation and Evaluation

Following the modification of datasets, we proceed to fit our model using Random forest algorithm. The code below gives us a summary of our result, shown in Figure 8. The out-of-bag error score gives us an unbiased prediction error of the model and we can see that our score is 11.09%, which is not bad.

```
rf <- randomForest(extractFeatures1(trainset2),
as.factor(trainset2$LL_indicator),
ntree=500, mtyr=5,importance=TRUE)
```



```
Call:
 randomForest(x = extractFeatures1(trainset2), y = as.factor(trainset2$LL_ind
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 11.09%
Confusion matrix:
      0   1 class.error
0 10506 110  0.01036172
1  1209  71  0.94453125
```

Figure 8: Summary of random forest

We can also evaluate the importance of each variables by how much it decreases the mean accuracy and gini index. From our variable importance plot (Figure 9), we can see that the 3 most important variables are "purchaseprice", "costnew", and "vehicleage". If we don't account one of these three variable, the mean accuracy and gini index will decrease at least 20% each. Additionally, variables "yearlicensed", "ownership", and "vehicleuse" are also relatively important.

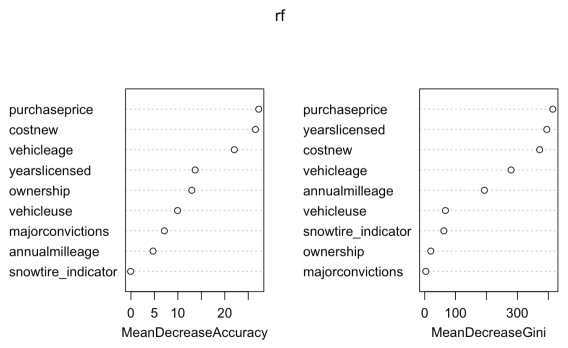
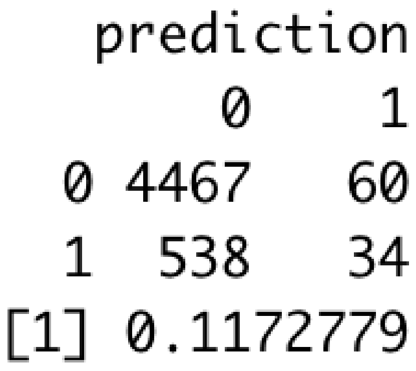


Figure 9: Variance Importance Plot

2.2.5 Validation

Finally, we can predict the target variable(LL indicator) for our test dataset and evaluate the model's performance. Using the codes shown below, we get our confusion matrix graph [9] shown in Figure 10. It is clear that only 60 class 0 predictions are wrong, and there are only 34 out of 538 class 1 predictions correct. Hence, the misclassification rate is bit high, but overall, the fitting performance is not bad since the error rate is below 12%.

```
prediction<-predict(rf,newdata=extractFeatures1(testset2))
tab <- table(testset2$LL_indicator, prediction)
tab
1- sum(diag(tab))/sum(tab)
```



```
     prediction
        0    1
  0 4467   60
  1  538   34
[1] 0.1172779
```

Figure 10: Confusion Matrix

In general, the performance is lower than what we expected. The misclassification rate is very high for predicting class 1 and it is not a good sign in an insurance company's point of view. Class 1 represents a high loss in a profile, so not able to predict them well is a bad thing. There are a lot of improvements that can be made in the future. One way to improve this model is by using cross validation method. This technique chooses the optimal parameters for our model instead of using the default values of the parameters in random forest function like what we did here. As a result, the error rate could be lower.

3 Conclusion

3.1 Summarization on k-Nearest Neighbors

The k-Nearest Neighbor is a model that classifies data points based on their similarities. There are three key elements in k-NN:

- The Euclidean distance calculation.
- The selection of k.
- The decision of classification.

Advantages of k-NN is applicable for both classification and regression. Firstly, k-NN is easy to implement, since it does not require training period to build a model. Secondly, the only required calculation in k-NN is the euclidean distances between different data points on the basis of different features.

k-NN is called a lazy learner because it only memorizes the training dataset instead of learning a function from the training dataset. This means that no ”training” is required in k-NN algorithm. Although this disadvantage can be seen as a convenience, it is not always beneficial for dealing with large datasets. Another disadvantage of k-NN is that the model is too sensitive to numbers of k [10], which will affect prediction results. For example, a smaller k may lead to overfitting of the model. To explain further, because there are always some data points that belong to individual cases, the results may perform well on the training dataset but not on the new dataset. On the other hand, the classification boundary could be blurred and too smooth if k is too large. This is because if k is large enough to approach the number of data points in the dataset, the result becomes the frequency of occurrence of each data point and k-NN becomes meaningless. This is called underfitting and it causes model to perform poorly on the training data. Additionally, k-NN is not suitable for high dimensional datasets. Despite all of these disadvantages, k-NN algorithm is still used in various industries, such as forecasting stock market, credit rating in insurance, loan management in banking, and money laundering analyses [6].

3.2 Summarization on Random Forest

Random forest is a typical algorithm of ensemble machine learning called bootstrap aggregation or bagging. There are three key elements in the following:

- The weak generalization ability improves in decision tree.
- The correlation between each tree is proportional to error rate.
- The number of feature selection is proportional to correlation and classification ability.

One advantage of random forest algorithm is its diversity, it's applicable to both classification and regression problems. Another advantage is that random forest has the ability to deal with missing values in the dataset without affecting the accuracy of the fitting. The biggest advantage, however, is that random forest model does not over-fit as long as the data set is large enough, because it is a collection of decision trees and, therefore, it is insensitive to outliers. Compared with k-NN, random forest can better handle large dataset with higher dimensions. However, random forest algorithm also has some drawbacks. Random forest is effective for classification but not for regression. Furthermore, random forest algorithm suffers from the problem of interpretability of the decision trees, which is likely to lead to failure in determining the importance of each variable. As the number of decision trees in a random forest increases, the time required for training becomes larger, and since the algorithm builds numerous trees to combine their outputs, it also requires a lot of computational power. [11]Despite its disadvantages, random forest can also be applied to various industries, such as fraud detection, stock movement, credit rating, product recommendations in e-commerce, and disease detection. [8]

3.3 Challenge to Machine Learning

In modern society, machine learning algorithms have emerged in almost every industry, such as driverless cars, environmental forecasting, and financial management, all of which require algorithms to support their research. However, machine learning is based on big data, and model cannot be established without data. Therefore, while machine learning requires massive data to support its training process, the quality of data should also be good and unbiased. Meanwhile, in this process, a lot of personal privacy information and confidential documents will be exposed to the risk of leakage. For example, hackers can perform backward chaining on sensitive attributes of training data by interacting with the model. Therefore, the implementation of privacy protection in machine learning has become fundamental to the development of machine learning, and this issue has received increasing attention from the research community and industry.

In conclusion, this report first introduces the methods of machine learning, including supervised learning and unsupervised learning. In addition, the advantages and disadvantages of machine learning are compared, and two different machine learning algorithms, k-NN and random forest, are implemented using the same dataset from the 2nd McMaster/Co-operators e-workshop, demonstrating their different algorithmic concepts and computational methods. From the result of the models, the errors suggest that the models may need new and stronger correlation variables to achieve better results. Today, as technology flourishes, challenges of machine learning also emerge, and data security and privacy protection will become important factors and development challenges in the future.

References

[1] *Corporate overview*, https://www.cooperators.ca/en/About-Us/corporate-overview.aspx, 2021, [Online; accessed 13-April-2020].

[2] *Machine learning: What it is and why it matters*, https://www.sas.com/en_ca/insights/analytics/machine-learning.html, 2021, [Online; accessed 13-April-2020].

[3] Jason Brownlee, *Supervised and unsupervised machine learning algorithms*, =https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/, Aug 2020.

[4] IBM Cloud Education, *What is supervised learning?*, https://www.ibm.com/cloud/learn/supervised-learning, Aug 2020, [Online; accessed 13-April-2020].

[5] Onel Harrison, *Machine learning basics with the k-nearest neighbors algorithm*, https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761, September 10, 2018, [Online; accessed 11-April-2020].

[6] S.B. Imandoust and M. Bolandraftar, *Application of k-nearest neighbor (knn) approach for predicting economic events: Theoretical background*, https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.389.4234&rep=rep1&type=pdf, 2013, [Online; accessed 10-April-2020].

[7] JavaPoint, *K-Nearest Neighbor(KNN) Algorithm for Machine Learning*, https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning, [Online; accessed 11-April-2020].

[8] M. Meena, *Applications of Random Forest*, https://iq.opengenus.org/applications-of-random-forest/, [Online; accessed 10-April-2020].

[9] Sarang Narkhede, *Understanding Confusion Matrix*, https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62, 2018, [Online; accessed 10-April-2020].

[10] A. Soni, *Advantages and Disadvantages of KNN*, https://medium.com/@anuuz.soni/advantages-and-disadvantages-of-knn-ee06599b9336, July 2020, [Online; accessed 10-April-2020].

[11] Great Learning Team, *Random Forest Algorithm- An Overview*, https://www.mygreatlearning.com/blog/random-forest-algorithm/, February 2020, [Online; accessed 10-April-2020].

[12] Tony Yiu, *Understanding Random Fores*, https://towardsdatascience.com/understanding-random-forest-58381e0602d2, 2019, [Online; accessed 10-April-2020].