Individual Report of ID5059 Project Two

170010240

Contents

Introduction

This project focuses on a Kaggle competition named "Porto Seguro's Sage Driver Prediction". This competition aims to predict whether a driver will do an insurance claim next year based on given 58 variables. Although it is a classification problem, comparing to submit binary predictions to Kaggle, probabilities of a person will claim an insurance next year will gain higher score. To do modelling, R and some packages in R are applied. My work is concentrated on neural network models and the best public score on Kaggle is about 0.2612 (the third best model in Group Kraken). In this report, firstly, data and data processing will be introduced. Secondly, works on neural network models will be discussed. Finally, a conclusion about neural network will be given.

Data and Data processing

Both training dataset (train.csv) and testing dataset (test.csv) are very complex. There is one response variable named "target" with binary 0/1 values[1] and 57 potential explanatory variables (31 categorical variables and 26 numeric variables).

Two main problems appear among the data. Firstly, both datasets contain missing values. When a value is -1, it means it is a missing value. To explore datasets, function aggr() in VIM package is applied. The function can display situations about missing values by graphs. Results of missing values are attached in Appendix B. In general, train.csv contains 846,458 missing values (2.49% of total training data) while test.csv has 1,270,295 (2.50 % of total testing data) missing values. The second problem is meanings of variables are not given explicitly. This means some variables may be not related to the response variable but we cannot detect them directly. Also, we cannot think about potential correlations among explanatory variables.

To deal with these problems, our team assign the data exploration work to individuals. The distribution of each column, missing data situations, and correlation matrix are used. Eventually, 25 variables are dropped. However, worries about dropping so many variables just because of missing values do exist. Thus, I consider to try to keep dataset integrated as much as possible when modelling.

[1] Target: 0 means the customer did not claim the insurance in 2016; 1 means the customer did.

Neural Network

Introduction to Neural Network

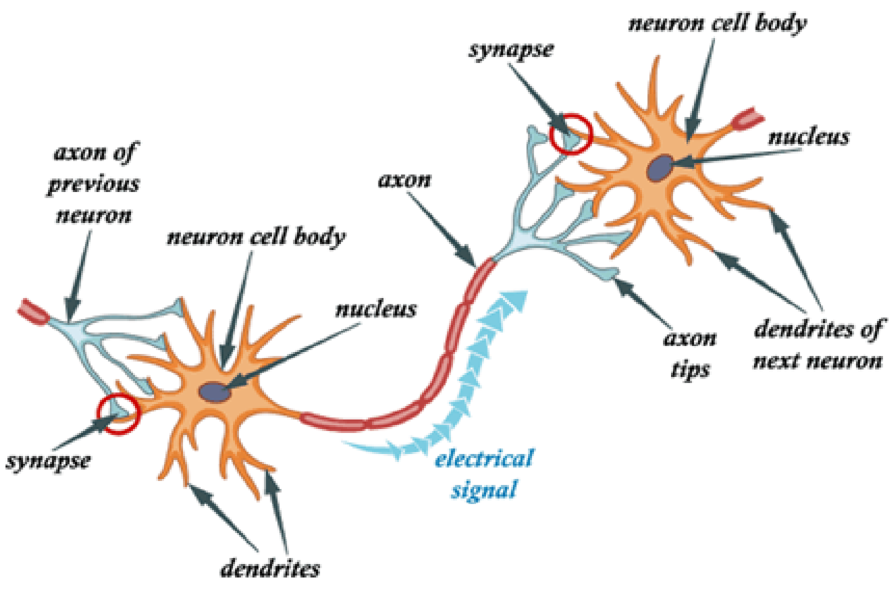

Figure 1 Biological Neural Network [1]

Neural network is a powerful model for prediction. It works as human's neural net (as shown in Figure 1). A stimulation works as inputs to neural networks. Then nerve cells deal with the stimulation layer by layer, and finally give response to this stimulation. Figure 2 describes the rough idea of a neural network model. One model consists of three parts: inputs (X1, X2, and X3 with weights W1, W2, W3), neural networks ($\sum f$) and outputs (y). Data work as inputs of the model. Each variable gives a stimulation to the network which works as a black box. One network contains at least two layers which are responsible to control inputs and outputs. An optional layer is a hidden layer. A hidden layer is training algorithms works without any manual inputs and outputs. The more a neural network topology complicated, the lower variance the model can have. However, overfitting problems may occur [2].

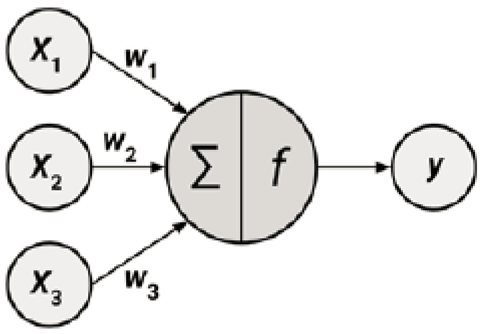

Figure 2 A Neural Network Progress [2]

Model Fitting and Assessment

In general, the neural network model for this project does not work ideally. The Kaggle public score is only 0.2612. The main reason can be neural network is not suitable for this complex dataset.

To do model fitting, package nnet and neuralnet are applied with R in RStudio are used. nnet() in package nnet is the most basic neural network function in R [3]. It contains basic parameters such as the number of single hidden layers, weights for each example, the number of iterations. neuralnet() in package neuralnet is more complicated than nnet(). It can establish neural network with multi-layers. However, neuralnet() do not assist categorical variables directly [4]. To evaluate performances of the two models, firstly, samples which contain 32 explanatory variables and do not have missing values are used. confusionMatrix() in package caret shows that there is no obvious difference between two models at the sample complexity (as shown in Table 1). I decided to use both models at next step.

Model	Accuracy	95% CI	Sensitivity	Specificity	Negative Prediction	Positive Prediction
nnet()	0.88	(0.87, 0.89)	0.88	0.97	0.05	0.99
neuralnet()	0.83	(0.82, 0.84)	0.83	0.93	0.02	0.94

Table 1 Results of confusion matrix for nnet and neuralnet with the same complexity

In terms of model assessment, a function found on Kaggle, to calculate Gini coefficient is applied (as shown in Appendix C). This Gini result is roughly consistent with Kaggle's score. In addition, variances are referred as a criterion.

nnet( )

While prediction results of samples are ideal, at beginning, the modelling for all train.csv is not good. All predictions are zeros no matter when 57 variables or 32 variables are applied. Interestingly, even normalisation is applied to numeric variables to make values between 0 and 1, the outcome still did not be improved. After reading discussion on Kaggle, I decided to drop all "calc_" variables, which seems not related to target [5]. Next, according to one competitor, Snow Dog's code [6], which has 0.279 public score on Kaggle, I modified my code (Appendix A). Figure 3 shows the algorithm of the code.

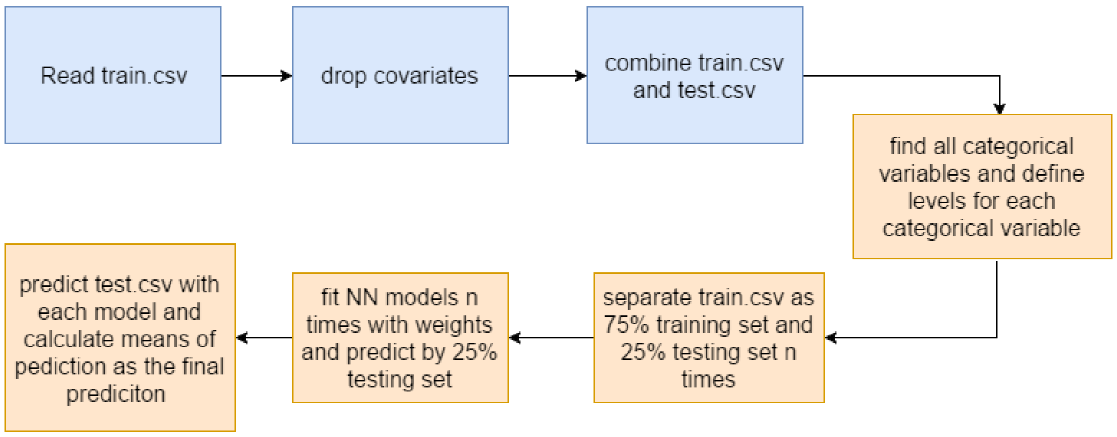
Read train.csv
drop covariates
combine train.csv and test.csv
find all categorical variables and define levels for each categorical variable
separate train.csv as 75% training set and 25% testing set n times
fit NN models n times with weights and predict by 25% testing set
predict test.csv with each model and calculate means of pediction as the final prediciton

Figure 3 The Algorithm of Neural Network Model Fitting

The code aims to solve two problems. Firstly, for some categorical variables, levels in train.csv and test.csv are not consistent. At beginning, I dealt with this kind of categorical variables by treating them as numerical data, which was not reasonable. Snow Dog gave me the hint that combing both datasets and use function factor() can solve this problem. factor() can identify all levels in the dataset, say the dataset as AllData. When train.csv is split from AllData, even if train.csv does not contains values at some levels, the definition can still exist. Figure 4 describes the idea of this progress.

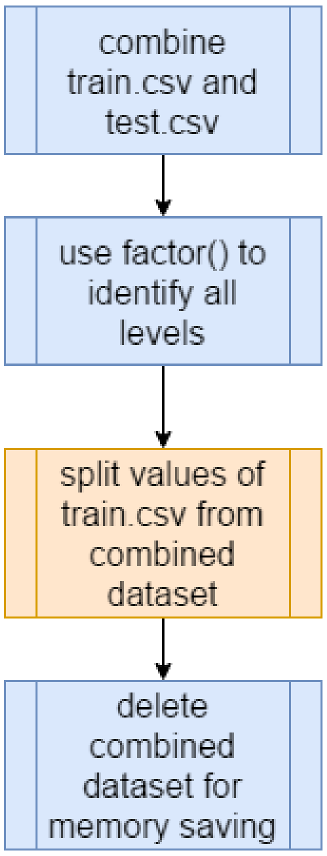
combine train.csv and test.csv
use factor() to identify all levels
split values of train.csv from combined dataset
delete combined dataset for memory saving

Figure 4 The Progress of Dealing with Different Levels

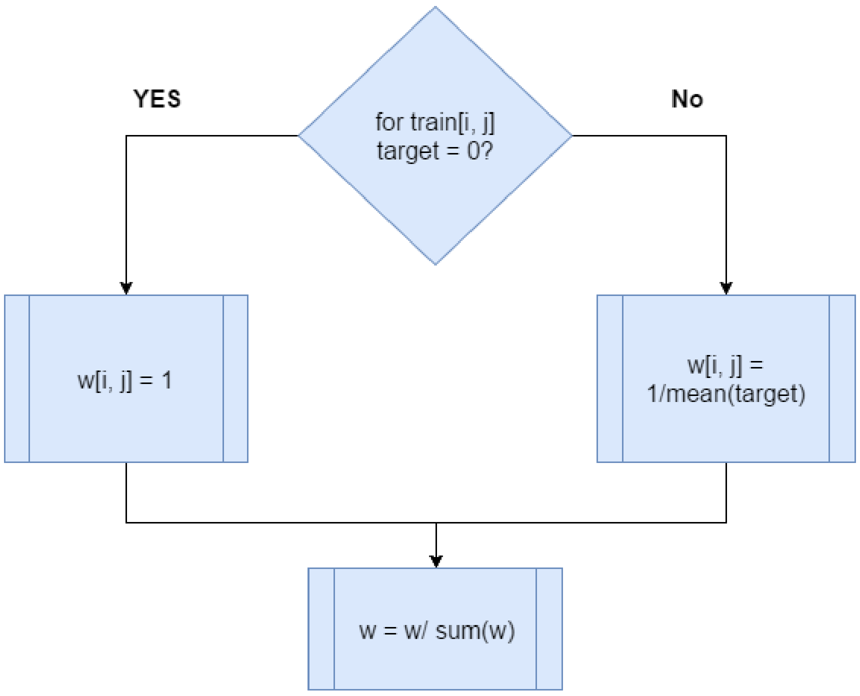

Figure 5 The Flowchart of Defining Weights[2]

In terms of the second problem, it is how to generate relative reasonable predictions. I planned to use results of one single model. Snow Dog's code offers another reasonable solution. It involves the idea of cross-validation and weights. As shown in Figure 3, train.csv is separated as 75% training data and 25% testing data n times. The n with default value 5, can be defined by users. Now we suppose n is five. By a function supplied by Stack Overflow, get_folds() (Appendix D), five different train datasets and relative five test datasets will be generated. Then, weights of each data point are defined as Figure 5. Initial weights of datasets can help neural network to find the local optimum which is close to the global optimum. After preparations, the next step is model fitting five times. At each time, 25% testing data driven from trian.csv are used to predict and assess the model performance by Gini coefficients generated by normalizedGini() (Appendix C). Next, predictions of 25% testing set' predictions and those of test.csv are added together. Then, the mean of five-times predictions is taken as the final predictions. Without dealing with any missing values, this solution can help the model get 0.279 (by Snow Dog's model) score effectively. This idea is a combination of classical statistics and data mining. For our model, the best public score is 0.2612 by nnet(). The model has seven layers (five hidden layers with single point) and it converged at iteration 3,220. 37 explanatory variables are included in the model, which referred to variable selection of light GBM, the best model of term Kraken. The possible problem of this model is overfitting. Practice shows that models can behave worse when too many layers are designed (usually more than 12 layers). To solve this problem, advanced optimisation algorithms should be applied. However, because of time limit, it is not achieved.

[2] train[i, j] is the data point at row I and column j. w is the weight matrix with the same dimension as train.

neuralnet ( )

In terms of neuralnet(), maybe because this function cannot deal with categorial data directly, the performance of this model is not ideal. For a model contains five layers with three hidden layers (each hidden layer has three knots) and 32 variables, its public score is only 0.11 on Kaggle. Thus, although the package supplies an easy way to plot neural network models (an example is shown in Appendix E), this function is not applied at late term.

Pros and Cons of Neural Network in this Project

Based on one weeks' modelling, following advantages and disadvantages of neural network for this project are given:

Advantage

Firstly, because of neural network involves a black box progress, the modelling is relative easier comparing to statistical models such as GLM. Secondly, neural network requires simple variables but can process with very complex internal linear or non-linear relationships. For simple and clear inputs and outputs, it works well. If we can understand the relationship of response variables and explanatory variables well, weights and layers of neural network models can give ideal generalised predictions. Finally, for a neural network model, different from models such as Naïve Bayesian, there is no more assumptions such as "data should be normally distributed", "independent variables" or "constant variance". That is, no statistical restrictions to use the model [7].

Disadvantage

Comparing to our team's best and second-best model, light GBM and XGBoost, neural network has five main disadvantages. Firstly, light GBM and XGBoost do not need to consider missing values particularly. They have internal algorithms to deal with missing values automatically. For a neural network model (specifically, for nnet() and neuralnet()), if missing values in test.csv are not dealt with properly, prediction() in both caret and nnet packages cannot work and NA (Not Aviailable Values) will be given directly. Secondly, both functions in R for neural network consume a lot of time. For a model with 57 explanatory variables and three layers (one hidden layer), if the maximum number of iterations is assigned as 2,000, at least 40 minutes are required to generate a model. For light GBM, the best model only takes about 15 minutes. Another problem of neural network is that it required relatively high configuration of executive machines. RStudio crashed down several times when a laptop with Intel(R) Core(TM) i5-6200U CPU (8.00 GB RAM) is used, which works well with other models in this project, such as light GBM, GLM and XGBoost. In addition, neural network has higher requirements of inputs. For XGBoost, even if all train.csv data are used without preliminary treatment, according to my teammate, the model can also gain 0.23 public (as stated by my teammate who works on XGBoost) score on Kaggle while neural network cannot produce a model under this situation. Fifthly, neural network's critical feature – black box, which is mentioned as an advantage in last paragraph, can be a disadvantage when do model interpretation. Different from classical statistics, it may be difficult to explain the relationship between response variables and explanatory variables. For instance, it is difficult to explain missing values's role in this model. All missing values in datasets are marked as -1. This implies the model treat them as a new category or a new numeric value. This is different from treating them as NA.

A Blender Model with Neural Network and light GBM

After the best model light GBM achieved the public score 0.28947 on Kaggle, we are seeking for a way to improve the score to 0.29. Regarding to the discussion by the team ranked the second on private leader board, a blender model is a choice [8] .

There are two ways of doing the blender model. The first one is simple, light GBM is applied to find currently the most appropriate explanatory variables. Then light GBM and neural network will be used N times to generate N*2 predictions. Finally, a mean of the N*2 predictions is considered as the results. The second way is using light GBM to generate predictions (target) n times (n should be at least 20). The target is probabilities of one person will claim the insurance next year. Then, taking the n sets of predictions as n explanatory variables, the binary target (0/1) as the response variable, neural networking is applied to do the model fitting. The result is good. For example, only for one explanatory variable included in the model, the public score on Kaggle can be 0.18, which is higher than the more complex neuralnet model mentioned in Chapter neuralnet ( ). However, the public score still does not reach 0.29. The main reason can be that explanatory variables are highly correlated. Regarding to Sherif, collinearity can be an issue for neural network model [9]. There are two main ways to solve this problem. One is use convolutional neural network instead of nnet(). Packages for convolutional neural network mainly are developed with Java, C++, Python, and MatLab. Thus, it is not advised to use this way during a short period. Another way to solve this problem is to use some algorithms issued on github. This also requires time to finish. Although this part is not implemented, the blender model gives people a new way to improve prediction outcomes based on current best models.

Conclusion

In general, neural net can work for this competition. However, roughly, its performance is not as good as XGBoost and lightGBM. Neural network has higher requirements to inputs, missing values, time, and configuration. For current neural network model, the best one is with 0.2621 public Kaggle score. There still have space to improve the model. However, because time limit, the improvement of the model is not implemented. In terms of a blender model with neural network and light GBM, it is possible to use it to do better predictions comparing to single model, while problems such as collinearity need to be considered.

References

[1] Jahnavi M. Introduction to Neural Networks, Advantages and Applications. [Internet]. 2017 [cited 2018 Apr 26]; [15 screens]. Available from: https://towardsdatascience.com/introduction-to-neural-networks-advantages-and-applications-96851bd1a207.

[2] Brett L. Machine Learning with R. 2nd ed. Birmingham UK: Packt Publishing Ltd; 2015.

[3] Brian R, William V. Package 'nnet'. [Internet]. 2016 [cited 2018 Apr 26]; [11 screens]. Available from: https://cran.r-project.org/web/packages/nnet/nnet.pdf.

[4] Stefan F, Frauke G, Marc S, Sebastian M. [Internet]. 2016 [cited 2018 Apr 26]; [13 screens]. Available from: https://cran.r-project.org/web/packages/neuralnet/neuralnet.pdf.

[5] Why removing ps_calc features improve the results? [Internet]. 2017 [cited 2018 Apr 26]; [3 screens]. Available from: https://www.kaggle.com/c/porto-seguro-safe-driver-prediction/discussion/41970#latest-236554.

[6] Snow Dog. Old school nnet. [Internet]. 2017 [cited 2018 Apr 26]; [9 screens]. Available from: https://www.kaggle.com/snowdog/old-school-nnet/code.

[7] Subana S, Sandhy S. Artificial Neural Network Modelling. 1st ed. Swaziland: Springer International Publishing Swaziland; 2016.

[8] 2Nd Place solution NN model. [Internet]. 2016 [cited 2018 Apr 26]. Available from: https://www.kaggle.com/xiaozhouwang/2nd-place-solution-nn-model/comments#latest-313223.

[9] Sherif H. Effects of Collineariry on Combing Neural Networks. Connection Science [Internet]. 2010 Jul [cited 2018 Apr 27]; 8:3-4, 315-336. Available from: https://www.tandfonline.com/doi/pdf/10.1080/095400996116794.

Appendix A

R code for neural network modelling:

```
library(VIM)
library(nnet)
library(caret)
lightgbm.train <- read.csv("D://train_probability.csv")
lightgbm.test <- read.csv("D://train2.csv")
train<-read.csv("D://pp2//train.csv")
test <- read.csv("D://pp2//test.csv")
target <- train$target
keep <- c(colnames(miaojie.test[,-1]),"id")
dropped <- setdiff(colnames(train),keep)
for( var in dropped) {
  train[[var]] <- NULL
  test[[var]] <-NULL
}
folds <- get_folds(nrow(train),5)
factors <- c(grep("cat",colnames(train)),grep("bin",colnames(train)))
all <- rbind(train,test)
for (fac in factors) {
  all[[fac]] <- as.factor(all[[fac]])
}
train <- all[1:357285,]
train<-cbind.data.frame(train,data.frame(target[1:595212]))
rm(all)
gc(reset = T)
# 25% as test, 75% as train for 5 times
ginis = list()
preds = 0
```

```
for(i in 1:length(folds)){
  idx = folds[[i]][['train']]
  train_tmp = train[idx, ]
  valid_tmp = train[-idx, ]
  pos_w = 1/mean(train_tmp$target)
  w = ifelse(train_tmp$target == 0, 1, pos_w)
  fit =
nnet(target~ps_ind_01+as.factor(ps_ind_02_cat)+ps_ind_03+as.factor(ps_ind_04_cat)+as.fac
tor(ps_ind_05_cat)+as.factor(ps_ind_06_bin)+as.factor(ps_ind_07_bin)+
as.factor(ps_ind_08_bin)+as.factor(ps_ind_09_bin)+as.factor(ps_ind_10_bin) , weights =
w,data = train_tmp, size = 5, MaxNWts = 10000,  decay = 0.00001, maxit = 500)
  pred = predict(fit, valid_tmp)[, 1]
  gini = normalizedGini(valid_tmp$target, pred)
  ginis[[i]] = gini
  cat("Fold ", i, "[gini]: ", gini, "\n", sep="")
  cat("Mean[pred]", mean(pred))
  cat("\n===================================\n\n")
  preds = preds + predict(fit, test)[, 1]
  rm(fit)
  gc(reset = T)
}
preds = preds/5
write.csv(data.frame(id=test$id, target = preds), 'D://sub_nnet_folds.csv', quote = F,
row.names = F)
rm(lightgbm.test)
rm(lightgbm.train)
```

Appendix B

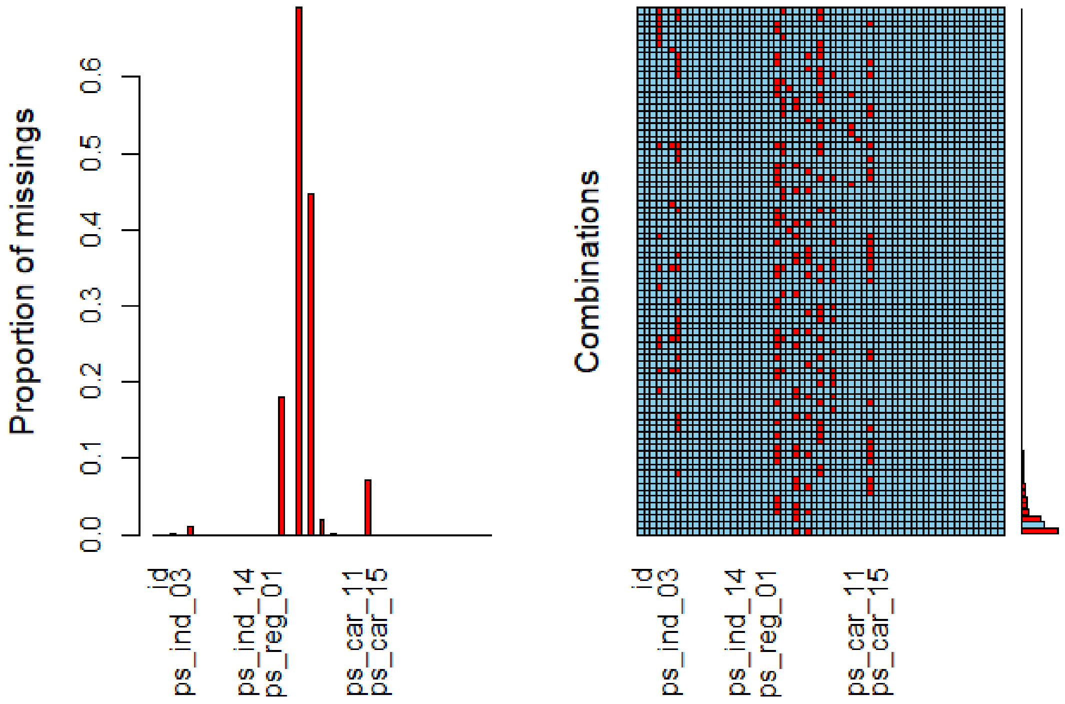

Figure 6 Missing values of train.csv

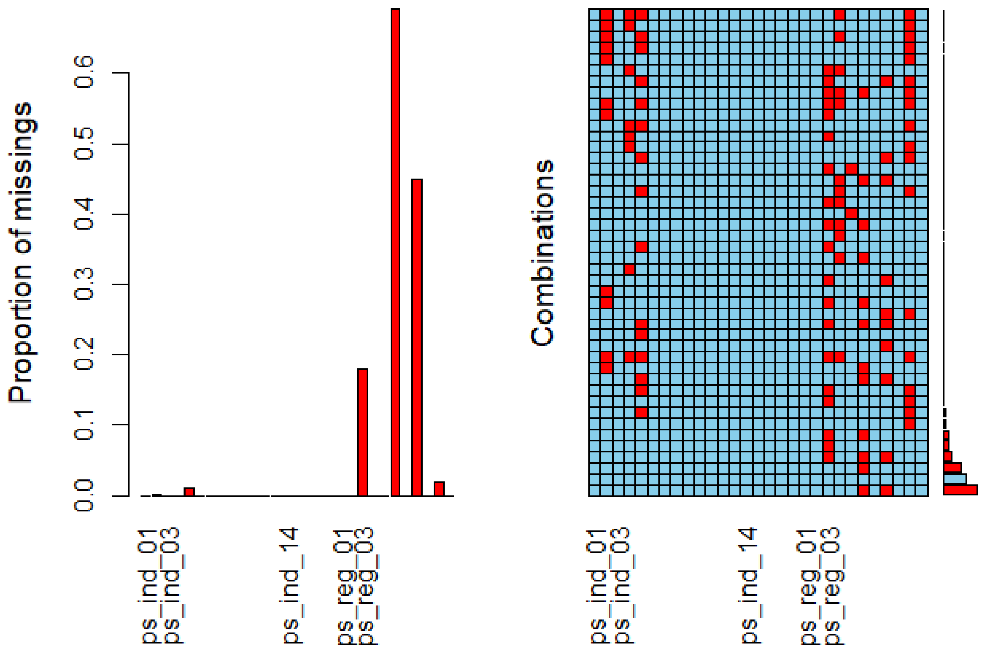

Figure 7 Missing values of test.csv

Appendix C

https://www.kaggle.com/c/ClaimPredictionChallenge/discussion/703

```
normalizedGini = function(aa, pp) {
  Gini <- function(a, p) {
    if (length(a) !=  length(p)) stop("Actual and Predicted need to be equal lengths!")
    temp.df <- data.frame(actual = a, pred = p, range=c(1:length(a)))
    temp.df <- temp.df[order(-temp.df$pred, temp.df$range),]
    population.delta <- 1 / length(a)
    total.losses <- sum(a)
    null.losses <- rep(population.delta, length(a)) # Hopefully is similar to accumulatedPopulationPercentageSum
    accum.losses <- temp.df$actual / total.losses # Hopefully is similar to accumulatedLossPercentageSum
    gini.sum <- cumsum(accum.losses - null.losses) # Not sure if this is having the same effect or not
    sum(gini.sum) / length(a)
  }
  Gini(aa,pp) / Gini(aa,aa)
}
```

Appendix D

```
get_folds <- function(Nobs, K = 5){
  rs <- runif(Nobs)
  id <- seq(Nobs)[order(rs)]
  k <- as.integer(Nobs*seq(1,K-1)/K)
  k <- matrix(c(0,rep(k,each=2),Nobs),ncol=2,byrow=TRUE)
  k[,1] <- k[,1]+1
  l <- lapply(seq.int(K),function(x,k,d)
    list(train=d[!(seq(d) %in% seq(k[x,1],k[x,2]))],
         test=d[seq(k[x,1],k[x,2])]),k=k,d=id)
  return(l)
}
```

Appendix E

This is a model generated by neuralnet() with 32 explanatory variables.

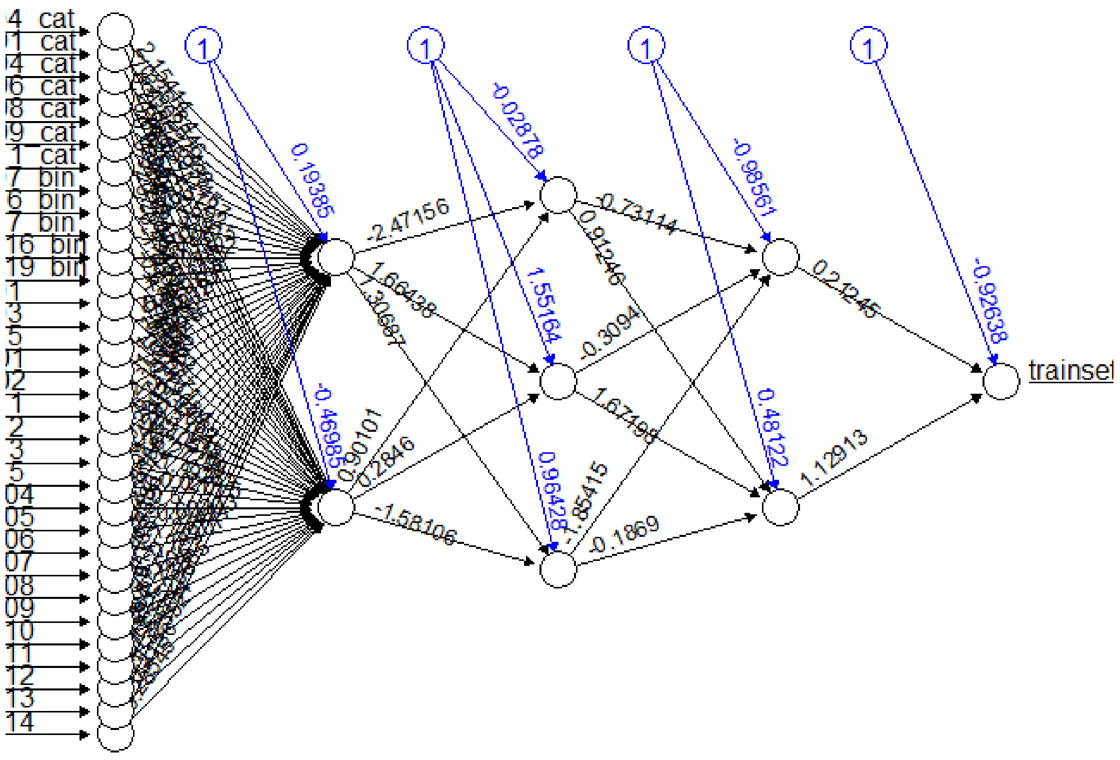

Figure 8 A Neural Network with Five Layers (three hidden layers)