AllState Insurance Claim Prediction

FrannieK

16 October 2016

Abstract

developing automated methods of predicting the cost, and hence severity, of claims. In this recruitment challenge, Kagglers are invited to show off their creativity and flex their technical chops by creating an algorithm which accurately predicts claims severity. Aspiring competitors will demonstrate insight into better ways to predict claims severity for the chance to be part of Allstate's efforts to ensure a worry-free customer experience."_

Random Forest Algorithm & Operation in R

Whilst abundent literatures explain the Random forest algorithm, it is still nonethelss prudent to outline here but also point out the technical limitations of performing in R which subseuqently led to an intermediate data reduction step (or dimension reduction - although not strictly, as some varaibles are entirely lost in the process).

Random forest for regression is an ensemble of decision trees whereby the final prediction is an **average** of indvidual nominal prediction of each tree which was first suggested by Breiman [1].

Data and Variables

The training data consists of 188318 observations with 116 categorical predictor variables (CVs) and 14 continuous predictor variables (CnVs). The dependent variable is the loss amount per claim which is continuous. The categorical values have been anonymised but from tabling of each variable, it is found that 75% of the CVs feature "A" the most and 85.7% of CnVs have mode values that are <0.5. It can be inferred that many variables are associated with each other (i.e. multicollinearity).

As a form of an explanatory analysis, I've plotted distribution of loss for all CV values. For binary CVs, "B" features tend to have higher median and 75% percentile loss amounts. I suspect these are yes/no questions. It also has broader interquantile range. This may suggest 1. "B" clients have higher claims and 2. they have bigger variance in the "majority" of the subset. It is yet clear which and how variables are associated to each other.

Modelling

Random Forest for regression will be used to predict the claim amount of the test set. Upon intial trial with all predictor variables as input, the operation took too much time. It is believed the number of variables as well as the large sample size (188318) are leading to long time calculation. Therefore, I reduced the number of intput variables by examining the association between 1. the dependent variable (DV) and and predictor variables and 2. between the predictor variables (PVs).

Between DV and PVs, pearson correlation waas adopted in order to remove "cont 13" and ANOVA to remove cat{2,4, 16, 17, 19, 51, 52, 55, 56, 65, 68, 69, 70, 74, 88, 97, 98, 104}.[2] There are 110 significant PVs to loss (p-value <0.05)

[1] Breiman, L. 2001. Random Forests, *Machine Learning*, Vol.45 Issue1, pp.5-32.

[2] "Correlation and Multicollinearity to reduce variables.spv"

Next, multiple comparisons of pair-wise correlation analysis were conducted in order to identify the multi-collinearity problems among the predictor variables.[3] For detecting affect size within CnVs, between CVs and CnVs and within CVs, pearson correlation, Eta^2 and Cramer's V are used respectively. Based on ρ >0.7 in SPSS, CnV cont{6,7,9,10,11,12,13} are removed. Based on η^2 >0.14, CV cat{75, 86, 88, 94, 95, 97, 98, 99, 104, 105, 106, 107, 109, 110, 112, 113, 115, 116} are removed. Finally, based on Cramer's V > 0.2 (which is more loose than the literature-recommended threshold of 0.5), CV cat{ 5 6 9 10 11 12 13 16 17 18 19 23 24 25 26 27 28 29 30 31 36 37 38 39 40 41 43 44 45 49 50 52 53 54 56 57 58 59 60 61 65 66 67 72 73 74 76 77 78 79 80 81 82 83 85 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116} are removed.

As a result, **31** input variables were ultimately selected at this stage. An RF was constructed without pruning to derive a regression method.

```
contcv = seq(1:14)
cv = seq(1:116)

# pearson correlation for test of significant correlation
# between y (loss continuous) and continuous x variable

tr.cont = train[, grepl("cont", names(train))]


corr = function(x) {
    with(train, cor.test(loss, x))

}

corr.t = sapply(tr.cont, corr, simplify = FALSE)
print(corr.t)
sapply(corr.t, "[[", "p.value") > 0.05

# SPSS produces the same result i.e. take out cont13.

rm1 = 13  #cont

# Use SPSS output

rm2 = c(2, 4, 16, 17, 19, 51, 52, 55, 56, 65, 68, 69, 70, 74,
    88, 97, 98, 104)  #cat

# Using SPSS Correlation table indicates the following
# continuous variables are significantly correlated based on
# Pearson's correlation >0.7.

# cont6 - cont1 cont9 - cont1 - cont6 cont10 - cont1 - cont6
# - cont9 cont11 - cont6 - cont7 - cont10 cont12 - cont6 -
# cont7- cont10 - cont11 cont13 - cont6 - cont10 (already
# redundant)

rm3 = c(6, 7, 9, 10, 11, 12, 13)

# between CV x and ContV x the multicollinearity is measured
```

[3]Liang, T. Lin, J. 2014. A two-stage segment and prediction model for morgage prepayment prediction and management, *International Journal of Forecasting*, Vol.30 pp.328 - 343

```r
# by Eta^2

eta2 = matrix(NA, nrow = max(contcv), ncol = max(cv))

# then loop it or sapply it ; looping is better i think

for (i in contcv) {
    for (j in cv) {
        f = as.formula(paste("cont", i, "~", "cat", j, sep = ""))
        eta = aov(f, data = train)
        eta2[i, j] = round(etasq(eta)[[1]][1], digits = 2)

    }
}

eta2 > 0.14
eta2.sig = which(eta2 > 0.14, arr.ind = TRUE)
eta2.sig = eta2.sig[eta2.sig[, 1] != eta2.sig[, 2], ]
rm4 = unique(eta2.sig[, 2])

# between correlation between CVs measured by Cramer's V or
# chi-squared test can work too.

cv = seq(1:116)

cramV = matrix(NaN, nrow = max(cv), ncol = max(cv))

# nested loop
for (i in cv) {
    for (j in i:max(cv)) {
        n = eval(parse(text = paste("train$cat", i, sep = "")))
        m = eval(parse(text = paste("train$cat", j, sep = "")))

        # cramV[[i]] = NaN if creating list, then assign value to
        # first element of a nesting vector or a list.
        cramV[i, j] = cramersV(n, m)

    }
}

# cramV result verified by sample comparison to SPSS output.
# now, filter by >0.5 which indicates strong correlation or
# effect size or multicollinearity but when I filtered by
# >0.5 it still left with too many CVs for RF i.e. takes too
# long to perform.so I loosened the criteria more to
# eliminate more.


criteria = cramV > 0.2
cramV.sig = which(criteria, arr.ind = TRUE)
cramV.sig = cramV.sig[cramV.sig[, 1] != cramV.sig[, 2], ]
rm5 = unique(cramV.sig[, 2])
```

The method for building a random forest follows these steps[4]:

Let D be a training dataset in a M-dimensional space X, and let Y be the class feature with total number of c distint classes.

1. Training data sampling: use the bagging method to generate K subsets of training data $\{D_1, D_2, \ldots, D_k\}$ by randomly sampling D with replacement. Due to this out-of-bagging method, there is no create a separate "train dataset" within the train dataset to test the accuracy of RF.

2. Feature subspace samping and tree regressor building: for each training dataset D_iv(1<=i<=K), usea decision tree alogrithm to grow an unpruned tree. At each node, randomly sample a subspace X_i of F features (F<<M), cpute all splits in subspace X_i, and select the best split as the splitting feature to generate a child node. Repeat this process until the stopping criteria is met, and a tree $h_i(D_i, X_i)$ built by trainig data D_i under subspace X_i is thus obtained.

3. Decision aggregation: ensemble the K trees $\{h_1(D_1, X_1), h_2(D_2, X_2), .., h_K(D_K, X_K)\}$ to for a random forest and use the average of these trees to make an ensemble regression decision

In order to further reduce the processing time, I've pre-determined the subspace to 10% of total sample size. The RF generation is then used to predict the claim amount in the test set.

```
final.cv = seq(1:116)
final.cv = final.cv[-c(rm2, rm4, rm5)]
final.contv = seq(1:14)
final.contv = final.contv[-c(rm1, rm3)]


trrfm = subset(train, select = c(paste("cat", final.cv, sep = ""),
    paste("cont", final.contv, sep = ""), "loss"))

samplesize = runif(500, 0, 1) * nrow(train)

rfm = randomForest(loss ~ ., data = trrfm, sampsize = nrow(train) *
    0.1, ntree = 500)
summary(rfm)
proc.time()

# claim prediction using RF
pred = predict(rfm, newdata = test)
head(pred)
```

Conclusion

```
sub = data.frame(id = test$id, loss = pred)

write.csv(sub, file = "FrannieK_Allstate.csv", row.names = FALSE)
# this leaves at 30% sample MAE of 1737.53900 which places me
# on 2421/2614 as at 27/11/2016 (top93%)
```

[4]Bharathidason, S., Venkataeswaran, C., 2014, Improving Classification Accuracy based on Random Forest Model with Uncorrelated High Performing Trees, *International Journal of Computer Applications*, Vol101. No.13

Random Forest Output Summary

Call: randomForest(formula = loss ~ ., data = trrfm, sampsize = nrow(train) * 0.1, ntree = 500) Type of random forest: regression Number of trees: 500 No. of variables tried at each split: 9

```
Mean of squared residuals: 6523387
          % Var explained: 22.65
```

	claim
Min	987.70
25th.Qu	2213.60
50th.Qu	2901.50
Mean	3072.80
75th.Qu	3502.30
Max	16139.80

Table 1: distribution of claim forecasts (USD)