Final Report: Minnesconsin Insurance Company Modeling Problem

Thao Nguyen

December 21, 2017

Abstract

This case study aimed to investigate influential factors that would affect an insurance policy-holder to cancel their policy before the end of term. A prediction model was used to predict a list of people who potentially would cancel their policies at Minnesconsin Insurance Company in the next year, based on 4 years of property insurance policy records. Based on previous records, some of the factors that would affect policy cancellation include credit scores, medium through which the policy was purchased, the number of years with the insurance company, length at one's residence, age, marital status, and zip code.

1. Introduction

Understanding how long a policy-holder would stay with an insurance company has always been an important question, as the upfront underwriting expenses associated with insuring an auto are expensive. The main purpose of this study was to create a retention model based on historical policy data for Minnesconsin Insurance Company. In order to understand what factors help renewal retention and what are most influential in a policy cancellation, a prediction model was used to identify policies that would be most likely to cancel before the end of their term.

2. Data Analysis

2.1. Data Description

The modeling data set was given by Travelers Analytics, based on 4 years of property insurance policies at Minnesconsin Insurance Company from 2013 to 2016. There were roughly 7500 observations in the training data set, each one representing a unique policy. A variable *ID* that keeps track of the policy ID numbers was not used in modelling but only used to output the list of prediction. There were 17 variables that were used in modeling, 6 of which were continuous and 11 were categorical.

The continuous variables include *tenure* (number of years with Minnesconsin Insurance Company), *n.adults* (number of adults in the property), *n.children* (number of children in the property), *premium* (price of the policy), *len.at.res* (length at residence), *ni.age* (age of policy-holder).
The categorical variables include 3 binary variables *claim.ind* (occurrence of claim), *ni.marital.status* (marital status of policy-holder), and *cancel* (cancellation indicator), which was the response variable. The rest of the categorical variables are *ni.gender* (gender of the policy-holder), *sales.channel* (medium through which policy was purchased), *coverage.type* (type of coverage), *dwelling.type* (type of dwelling), *credit* (financial credit level of policy-holder), *house.color* (color of house), *year* (year of the policy), and *zip.code* (zip code of the property).
In the test data set, the variable *cancel* was missing, as the prediction model would be used to predict the cancel indicator for each policy in the test data.

2.2. Data Pre-processing

Looking at the response variable *cancel* in the training data, which was expected to be binary, we found several observations to have the value -1, which we decided to delete from the training set with the consensus that we could not impute those values as any pattern found by these observations would not be reliable.
For zip code data, we tried clustering them by the first n digits, with n ranging from 1 to 5. However, we found no difference between the different clustering methods. Therefore, in the final model we just chose to cluster zip codes by their first digits.
We simply chose to delete the missing values in the training set, as there didn't seem to be much difference with or without the missing values. In the test data, we imputed the missing values by the median for numeric variables, and the mode for categorical variables. In the test data, there was also a 'Landlord' type that did not appear in the training data, so we simply treated it as 'House' like in the training data.
Several other data pre-processing methods were considered, like Box-Cox transformation of numerical data, centering and scaling data, or Markov blanket for feature selection. However, these methods did not improve our result. Therefore, we did not implement these data pre-processing methods in our final model.

2.3. Modeling Methods

2.3.1. Model Selection

For this classification problem, several methods were considered, namely Logistic Regression, Random Forest, SVM, Neural Network, and Boosted Trees. We evaluated our models by two criteria: *C-statistic*, which is the area under the ROC curve that measures how well the model is at rank ordering those most likely to cancel their policy, and *misclassification error*, which measures overall prediction accuracy. In order to penalize false positive errors (misclassifying a cancellation as a non-cancellation), we assigned a loss of weight 5 to this type of error, and a loss of weight 1 to the other type of error (misclassifying a non-cancellation as a cancellation). Under this asymmetric loss function, we decided to pick a cut-off point of 1/6 for the class label prediction to minimize the loss function. For each method, we divided

our training data into subsets and performed repeated 10-fold cross-validation to average for the *C-stastistic* and *misclassification error* under the asymmetric loss.
The final model was chosen as the one that outperformed in both criteria. Table 1 (Appendix A.1) illustrates the results of the models, from which Logistic Regression was chosen as the final predictive model.

2.3.2. Logistic Regression

First we selected only main effects variables. We used L_0 penalization for variable selection in order to reduce over-fitting and improve the prediction performance. The L_0 penalty was proven to be very useful in variable selection, as it directly penalizes the number of non-zero coefficients (Liu and Wu, 2007). We tuned the regularization parameter using cross-validation with 20% as the training data, repeated 50 times. After tuning for the optimal regularization parameter, we counted the appearances of the predictors, repeated 100 times using cross-validation. We selected the top 10 most frequently appeared variables as our main effects predictors.
Then we added interaction terms within continuous variables and categorical variables and chose the final model that minimized the difference between the *misclassification error* and the *C-statistic* (in other words, the model that had the highest *C-statistic* and lowest *misclassification error*), using cross-validation with 20% as the training data, repeated 100 times. The best model was used to make class predictions on the test data.

3. Results

3.1. Results Summary

The results of the best Logistic Regression model are presented in Table 2 (Appendix A.2).
Out of 16 predictors, only variables *claim.ind*, *sales.channel*, *credit*, *zip.code*, *tenure*, *ni.marital.status*, *tenure*, *len.at.res* were found to be significant (p-value < 0.001) as main effects terms.
The predictors *n.adults* and *n.children* were included in the model, but were not significant as main effects. However, there was significant association between these two variables (p-value < 0.001). There was also significant interaction between *len.at.res* and *ni.age* (p-value < 0.01).
Among the significant predictors, *credit* seemed to have a very strong effect on policy cancellation. People with low credit scores seemed most likely to cancel their policies. And compared to people with high credit scores, those with medium scores also seemed more likely to cancel their policies.
Another predictor that seemed to have a strong effect on policy cancellation was *sales.channel*. Compared to people who used a broker to purchase a policy, people who purchased their policies online and via phone also seemed more likely to cancel.
Compared to people who were not married, people who were married seemed less likely to cancel their policy. People who filed a claim seemed more likely to cancel compared to people who did not.
Some of the less influential predictors, although still proven to be significant, were *tenure*, *len.at.res*, *ni.age*. A unit change in the number of years with the company increases the log odds of cancellation by 0.03, implying that the longer the person stays with the company,

the more likely he or she is to cancel the policy. A unit change in the length at residence decreases the log odds of cancellation by 0.09, implying that the longer the person lives at the property, the less likely he or she is to cancel the policy. A unit change in age of the policy-holder decreases the log odds of cancellation by 0.05, implying that the older the person is, the less likely he or she is to cancel.
Adults who had children, or people who were older who had lived at their residence for a longer period of time also seemed more likely to cancel.
People with zip codes starting with 2 also seemed to be more likely to cancel their policies.

3.2. Evaluation

One interesting finding while doing model selection for this problem was that complex non-parametric models did not seem to perform as well as a simple Logistic Regression model. Non-parametric methods are useful as they do not make assumptions about the distribution of the population, especially when observations are subject to outliers. However, for this study, the assumptions for logistic regression still hold, as there were not a lot of abnormalities in the data and we already removed all missing values and potential outliers. The continuous predictors all hold some information about the response, and the factor predictors can be well described by linear functions. Furthermore, non-parametric methods are particularly useful when dealing with large sample sizes, but the sample size of this study was not very large. Therefore, for this case study, a simple parametric model like logistic regression outperformed other complex non-parametric methods.

4. Discussion and conclusions

This case study had raised insights on the influential factors in insurance policy cancellation. It made sense that a person with a lower credit score comes with a higher risk and would be more likely to cancel a policy before the effective term, compared to a person with a higher credit score. There had been numerous studies showing that there is a correlation between credit scores and potential insurance losses, in which high credit scores are associated with lower insurance losses (Hartwig, 2003).
It also made sense that a person who purchases a policy through a broker would be more committed, as he or she would have to go through a more well thought-out process to purchase the policy, compared to people who might have purchased the policy impulsively via phone or online.
Older people or people who had lived longer at their residence might be less likely to cancel their policies because they might be reluctant to change their current lifestyles, while people who had stayed a long time with the insurance company might be more likely to cancel if they have not had enough benefits from their policies, or want to try other insurance companies.
A surprising finding from this case study was the correlation between zip codes and the likelihood of cancellation. People whose zip codes started with number 2, which included the states WV, VA, NC, and SC, were shown to be far more likely to cancel their policies compared to other states. However, there remain several concerns about the use of zip codes

and the validity of the data, as zip codes are considered personal information and banned from being disclosed for some states in the US.

References

(1) Hartwig, R. 2003. The Use of Credit Information In Personal Lines Insurance Underwriting. Insurance Issues Series, Volume 1, Number 2.
(2) Liu, Y. and Wu, Y. 2007. Variable Selection via A Combination of the L_0 and L_1 Penalties. Journal of Computational and Graphical Statistics, Volume 16, Number 4, Pages 782-798.
(3) Travelers Analytics. 2017. Travelers Case Competition Instruction.

Appendix A

A.1

Model	C-statistic	Misclassification error
Logistic Regression	0.72	0.62
Random Forest	0.71	0.63
SVM	0.67	0.23
Neural Network	0.72	1.06

Table 1: C-statistic and misclassification error scores of implemented methods

A.2

Variable	Log Odds Estimate	Standard Error	
claim.ind 1	0.35	0.07	***
sales.channel Online	0.61	0.10	***
sales.channel Phone	0.83	0.06	***
credit low	1.51	0.07	***
credit medium	0.63	0.07	***
zip.code 2	1.31	0.15	***
zip.code 5	0.39	0.10	***
zip.code 8	0.41	0.09	***
zip.code 9	0.51	0.12	***
n.adults	-0.02	0.03	
n.children	0.01	0.02	
tenure	0.04	0.006	***
len.at.res	-0.09	0.02	***
ni.age	-0.05	0.009	***
ni.marital.status	-0.48	0.07	***
n.adults : n.children	0.05	0.009	***
len.at.res : ni.age	0.002	0.00005	**

Note: * = $p < 0.05$; ** = $p < 0.01$; *** = $p < 0.001$

Table 2: Logistic regression of factors that might affect insurance policy cancellation

Appendix B

R Codes of Main Methods

```
library(ROCR)
library(stringi)

############### Helper functions. ############################

# This functions removes observations with response -1.
remove.weird.response <- function(x) {
    # Inputs: x, dataframe of travelers insurance train or test
    # data.  Returns: a dataframe.
    return(subset(x, cancel != -1))
}

# This functions reorders columns so numerical, binary, and
# factor data are clustered together.
reorder <- function(x) {
    # Inputs: x, dataframe of travelers insurance train or test
    # data.  Returns: a dataframe.
    return(x[, c(18, 17, 15, 1, 3, 4, 7, 11, 14, 2, 6, 5, 8,
        9, 10, 12, 13, 16)])
}

# This function replaces zip code by its first digit,
# providing a region clustering of zip codes.
cluster.zip <- function(x) {
    x$zip.code <- factor(floor(x$zip.code/10000))
    return(x)
}

# This function converts factor data in the travelers train
# and test dataframes to labeled full rank dummy variables.
my.factor.to.dummy <- function(train, test = NULL) {
    # Inputs: train and test, travelers insurance train and test
    # dataframes.  Returns: a named list of two dataframes.
    dummy.obj <- dummyVars(~., train[, 10:18], fullRank = T)
    train <- cbind(train[, 1:9], predict(dummy.obj, train[, 10:18]))
    test <- cbind(test[, 1:9], predict(dummy.obj, test[, 10:18]))
    return(list(train = train, test = test))
}

# This function contains the full data preprocessing
```

```
# pipeline.
my.preprocess = function(train, test) {
    # Remove weird -1 responses.
    train <- remove.weird.response(train)
    # Reorder columns so numeric and factor are next to each
    # other.
    train <- reorder(train)
    test <- reorder(test)
    # Cluster by zip code.
    train <- cluster.zip(train)
    test <- cluster.zip(test)

    return(list(train = train, test = test))
}

# This function calculates misclassification error
misclassification = function(obs, pred) {
    temp = ifelse(obs == pred, 0, ifelse(obs == 1, 5, 1))
    mean(temp)
}

############## Preprocessing script ####################

col.classes = c("integer", "factor", "integer", "integer", "factor",
    "factor", "numeric", "factor", "factor", "factor", "numeric",
    "factor", "factor", "numeric", "integer", "integer", "integer",
    "integer")

train <- read.csv("train.csv", colClasses = col.classes)
test <- read.csv("test.csv", colClasses = col.classes)
# Performs preprocessing script on raw data
pp.result = my.preprocess(train, test)
# Training and test sets after preprocessing
data.train <- pp.result$train
data.test <- pp.result$test
data.train = data.train[, 2:18]
# Delete NA values in training data
data.train = na.omit(data.train)
# Delete observations with age > 100 in training data
data.train = data.train[data.train$ni.age < 100, ]
# Impute NAs by median for numeric variables or mode for
# factors
for (i in 1:16) {
    idx = which(is.na(data.test[, i]))
```

```
    if (class(data.test[, i]) == "numeric" | class(data.test[,
        i]) == "integer") {
        med <- median(data.test[, i], na.rm = T)
        data.test[idx, i] = med
    } else if (class(data.test[, i]) == "factor") {
        mod <- labels(which.max(summary(data.test[, i])))
        data.test[idx, i] = mod
    }
}
# Treat 'Landlord' variable in test data as 'House' like in
# training set
data.test$dwelling.type[data.test$dwelling.type == "Landlord"] = "House"


############ Logistic Regression Model Selection ############

##### Step 1: L0 penalization selection of main effects #####

# This function returns optimal penalization parameter
# 2:log(n) by CV
L0Selection.tune = function(cv.k = 100, cv.ratio = 0.2, lambda.step = 0.1,
    lambda.l = 1, lambda.u = 7.2) {
    n.data = nrow(data.train)

    lambda = seq(from = lambda.l, to = lambda.u, by = lambda.step)
    cv.err = numeric(length(lambda))
    cv.auc = numeric(length(lambda))

    for (j in 1:length(lambda)) {

        err = numeric(cv.k)
        cstat = numeric(cv.k)

        for (i in 1:cv.k) {

            train = sample(1:n.data)[1:round(n.data * cv.ratio)]
            logit = glm(cancel ~ factor(claim.ind) + sales.channel +
                credit + factor(zip.code) + n.adults + n.children +
                tenure + len.at.res + ni.age + ni.marital.status +
                premium + ni.gender + house.color + year + sales.channel +
           coverage.type, data = data.train[train, ], family = binomial(link = "logit"))

            temp = step(logit, direction = "both", trace = 0,
                k = lambda[j])
```

```
            logit = eval(temp$call)
            pred = predict(logit, data.train[-train, ], type = "response")

            input = prediction(pred, data.train[-train, ]$cancel)
            auc = performance(input, "auc")
            cstat[i] = unlist(auc@y.values)

            pred = ifelse(pred > 1/6, 1, 0)
            err[i] = with(data.train[-train, ], misclassification(cancel,
                pred))
        }
        cv.err[j] = mean(err)
        cv.auc[j] = mean(cstat)
    }
    idx = which.min(cv.err - cv.auc)
    return(lambda[idx])
}

# This function returns the counts of appearances of each
# predictors
L0Selection.count = function(cv.k = 500, cv.ratio = 0.2, lambda = 4) {
    c = numeric(16)
    names(c) = names(data.train)[2:17]
    for (i in 1:cv.k) {
        train = sample(1:n.data)[1:round(n.data * cv.ratio)]
        logit = glm(cancel ~ factor(claim.ind) + sales.channel +
            credit + factor(zip.code) + n.adults + n.children +
            tenure + len.at.res + ni.age + ni.marital.status +
            premium + ni.gender + house.color + year + sales.channel +
        coverage.type, data = data.train[train, ], family = binomial(link = "logit"))
        temp = step(logit, direction = "both", trace = 0, k = lambda)

        str = paste(deparse(temp$call), collapse = "")
        str = sub("~", "@", str)
        str = sub(",", "@", str)
        str = strsplit(str, split = "@")[[1]][2]
        str = stri_replace_all_charclass(str, "\\p{WHITE_SPACE}",
            "")
        str = unlist(strsplit(str, split = "[+]"))

        for (j in 1:16) if (names(c)[j] %in% str)
            c[j] = c[j] + 1
    }
    return(c)
```

```
}

##### Step 2: Greedy selection of two way interactions ######

# factor(claim.ind), sales.channel, credit, factor(zip.code),
# ni.marital.status, n.adults, n.children, tenure,
# len.at.res, ni.age

V0 = c("factor(claim.ind):sales.channel", "factor(claim.ind):credit",
   "factor(claim.ind):factor(zip.code)", "factor(claim.ind):ni.marital.status",
    "sales.channel:credit", "sales.channel:factor(zip.code)",
    "sales.channel:ni.marital.status", "credit:factor(zip.code)",
    "credit:ni.marital.status", "factor(zip.code):ni.marital.status",
    "n.adults:n.children", "n.adults:tenure", "n.adults:len.at.res",
    "n.adults:ni.age", "n.children:tenure", "n.children:len.at.res",
    "n.children:ni.age", "tenure:len.at.res", "tenure:ni.age",
    "len.at.res:ni.age")

M0.forluma = "n.adults + n.children +
tenure + sales.channel + credit +
factor(zip.code) + len.at.res + ni.age +
factor(claim.ind) + ni.marital.status"


# This function returns a formula of the best model with two
# way interactions
InteractSelection = function(V0, M0.forluma, M0.err = 0.75, M0.auc = 0.5,
    cv.k = 500, cv.ratio = 0.2) {

   express = c("glm(cancel ~", ",data = data.train[train,], family = binomial)")

    while (TRUE) {

        R = array(NA, dim = c(length(V0), 2))

        for (i in 1:length(V0)) {

            n.data = nrow(data.train)

            err = numeric(cv.k)
            cstat = numeric(cv.k)

            for (j in 1:cv.k) {
```

```
                train = sample(1:n.data)[1:round(n.data * cv.ratio)]
                M1 = eval(parse(paste(express[1], "+", M0.forluma,
                  V0[i], express[2])))
                pred.p = predict(M1, data.train[-train, ], type = "response")

                input = prediction(pred.p, data.train[-train,
                  ]$cancel)
                auc = performance(input, "auc")
                cstat[j] = unlist(auc@y.values)

                pred.c = ifelse(pred.p > 1/6, 1, 0)
                err[j] = with(data.train[-train, ], misclassification(cancel,
                  pred.c))
            }

            M1.err = mean(err)
            M1.auc = mean(cstat)

            if (M1.err < M0.err & M1.auc > M0.auc) {

                R[i, 1] = M1.err
                R[i, 2] = M1.auc
            }
        }

        if (length(na.omit(R[, 1])) > 0) {

            idx = which.min(R[, 1] - R[, 2])

            M0.forluma = paste(M0.forluma, "+", V0[idx])
            M0.err = R[idx, 1]
            M0.auc = R[idx, 2]

            V0 = V0[-idx]
        } else break
    }
    return(M0.forluma)
}

############ Logistic Regression Cross-validation ###########

err = numeric(100)
cstat = numeric(100)
n.data = nrow(data.train)
```

```r
for (i in 1:100) {

    train = sample(1:n.data)[1:round(n.data * 0.2)]

    # the best logit model M0
    logit0 = glm(cancel ~ factor(claim.ind) + sales.channel +
        credit + factor(zip.code) + n.adults * n.children + tenure +
        len.at.res * ni.age + ni.marital.status, data = data.train[train,
        ], family = binomial(link = "logit"))

    pred0.p = predict(logit0, data.train[-train, ], type = "response")

    input0 = prediction(pred0.p, data.train[-train, ]$cancel)
    auc0 = performance(input0, "auc")

    cstat[i] = unlist(auc0@y.values)  #- unlist(auc1@y.values)

    pred0.c = ifelse(pred0.p > 1/6, 1, 0)

    err[i] = with(data.train[-train, ], misclassification(cancel,
      pred0.c))  #- with(data.train[-train,], misclassification(cancel,pred1.c))

}

mean(err)
mean(cstat)

################ Prediction on Test data ####################

# Final logistic model
logit = glm(cancel ~ factor(claim.ind) + sales.channel + credit +
    factor(zip.code) + n.adults * n.children + tenure + len.at.res *
   ni.age + ni.marital.status, data = data.train, family = binomial(link = "logit"))

# Probability prediction in test data
pred.prob = predict(logit, data.test, type = "response")
# Classification with 1/6 cut-off threshold
pred.class = ifelse(pred.prob > 1/6, 1, 0)

output = list(id = data.test$id, prob = pred.prob, class = pred.class)
write.csv(output, file = "group4.csv")
```