Insurance Retention Modeling

CHUNLIN LI

Abstract. The goal of the retention modeling is to build a model with good predictive performance and strong explanatory power. In this project, we explored various approaches to this problem. The logistic regression demonstrates a superiority over other methods on the dataset. The final logit model is then constructed using the proposed procedure. With the help of this model, we are able to accurately identify a potential policy cancellation as well as understand what predictors are influential in causing a policy cancellation.

1. Introduction

Retention rate is a key metric of the performance of a company. To improve the competitiveness in the market, an insurance company is interested in (1) identifying the policies that will be canceled before the end of the term and (2) understanding the most influential predictors in causing policy cancellation.

The goal of this retention modeling project is to construct a model that has excellent predictive performance and strong explanatory power. For prediction, the model should accurately classifies whether a policy will be canceled or not during the effective term. For interpretation, the model should be able to give a good description of the important predictors in causing policy cancellation.

In this project, based on historical insurance data from 2013 to 2016, we explore several popular statistical and machine learning approaches to retention modeling, including random forest, support vector machine, neural network, boosted trees, and logistic regression. Interestingly, the logit model constantly outperforms the other nonparametric methods. In this report, we will focus on the logit model and provide a statistical justification for its excellent performance. A procedure is proposed for construction of the final model. This model is assessed with the test data of 2017. The result shows it is highly competitive in prediction. In addition, using this model, we are able to identify a set of important predictors in causing cancellation.

The rest of this report is organized as follows. Section 2 introduces our attempts and analysis of this dataset. It also proposed a construction procedure

This project is a joint work with the other members of Group 4.

for the final model. Section 3 summarizes of test results and interprets the model. Finally, Section 4 concludes the report.

2. Data analysis

The training data consist of 4 years of property insurance policies from 2013 to 2016. There are 7498 policies in the training dataset, among which 1786 were canceled during the effective term. The cancellation indicator takes value 1 if a policy was canceled, and value 0 otherwise. The test dataset has 2499 policies of 2017, for which the cancellation indicator is labeled as missing value (denoted by NA). Every policy in training and test data has 16 valid predictors with 6 continuous predictors and 10 categorical predictors (including 2 ordinal predictors).

2.1. **Data preprocessing.**

2.1.1. *Misspecified values.* The training data have some evidently misspecified values, including policies with (1) cancellation indicator being labeled as -1, or (2) the age of policyholder exceeds 120. The misspecified values in training data are removed. For test data, policies with the misspecified values (e.g. age of policyholder > 120) are kept.

2.1.2. *Missing values.* Several methods for dealing with missing values are considered, including simple imputation (by median or mode), adding an NA indicator, and deletion. We evaluate these approaches by the experiment. The result suggests that they are not significantly different, and deletion is preferred for simplicity.

For test data, the simple imputation for missing values is performed. Note that there are policies with dwelling.type (categorical) being "Landlord", which does not appear in the training data. This new label is treated as NA.

2.1.3. *Clustering zip code.* The predictor zip.code has many levels. Clustering zip code by first one, two, three, and four digits are considered. The experiment results suggest that there are no significant improvement when more than one digit are included. Hence, zip code is clustered using the first digit, which contains the information of state in the US.

2.1.4. *Regulation constraints.* The usage of gender, marital status, and property location for business may violate the state regulation. In this project, we build the model without regulation constraints. It is shown that the gender seems not an informative predictor while the marital status may be statistically important. The interpretation, however, will not be based on these controversial predictors.

2.2. **Choosing a method.** The prediction performance is assessed by AUC and the (misclassification) error, i.e., it is aimed to maximize AUC and minimize the error. Note that AUC requires probability estimation. Hence, we prefer classifiers that can accurately estimate the conditional probability $P(Y \mid X)$, where Y is the response and X is the predictor vector. For class prediction, the simple plug-in estimator $f(x) = I\{p(x) > c\}$ is adopted, where $p(x)$ is the estimate of $P(Y = 1 \mid X = x)$ and $I(\cdot)$ is the indicator function. The goal is to minimize the expected error $\mathrm{E}\, L(Y, f(X))$, where $L(y, f(x))$ is the asymmetric loss given by $L(1,0) = 5$, $L(0,1) = 1$, and $L(1,1) = L(0,0) = 0$. It can be shown that $f(x) = I\{P(Y = 1 \mid X = x) > 1/6\}$ is a Bayes decision rule. Hence, we fix the cutoff point $c = 1/6$.

Consider the logit model, random forest, support vector machine, neural network, and boosted trees. Compare to the logit model, the other 4 methods are nonparametric and has greater approximation capacity. Nevertheless, the experiments show that simple main-effects logit model constantly outperforms the other methods. Figure 1 illustrates the experiment result of the logit model and well-tuned boosted trees. It is shown that boosted trees roughly require three times sample size to achieve the same AUC as the logit model. This may be because the logit model can well-approximate the true function in the given range of sample size. In this case, the estimation error dominates the approximation error and the logit model has a faster convergence rate in estimation error than the nonparametric methods.

2.3. **Logit model.** The simple main-effects logit model demonstrates its strength in prediction. In this section, we propose a procedure for the enhancement of the logistic regression. The procedure consists of two steps.

2.3.1. *Main-effects selection.* It is known that AIC is minimax optimal for estimating regression function in nonparametric scenario and BIC is consistent in variable selection in parametric scenario [2]. It is, however, difficult to decide which scenario is proper. In addition, most theoretical justifications of AIC and BIC are based on asymptotics, which are doubtful when the sample size is not very large. Hence, we adopt the method of L_0-penalization, given by

$$\min_{\beta} \quad -\log L(\beta) + \lambda \sum_{j=1}^{p} I(\beta_j \neq 0), \tag{2.1}$$

where $L(\beta)$ is the likelihood function, $\beta = (\beta_1, \ldots, \beta_p)$ are parameters, and $\lambda > 0$ is a tuning parameter. The formulation (2.1) include AIC ($\lambda = 1$) and BIC ($\lambda = \log(n)/2$) as its special case. To solve the nonconvex optimization problem (2.1), we utilize the forward and backward search, which can be readily implemented by using the "step" function in R.

The optimal value of λ is determined by minimizing (error $-$ AUC) on the validation set over a set of grid points $1 + 0.1k$; $k = 0, \ldots, 62$. The splitting ratio $n_{\text{train}}/n_{\text{valid}} = 2 : 8$ is used. Note that $\log(n_{\text{train}}) \approx 7.2$, so this approach can be understood as combining AIC and BIC. Then the optimal $\lambda^* \approx 6.8$ is used to select the main effects in the following manner. (1) Randomly sample 20% of training data and perform selection. (2) Repeat for 100 times. Count the appearances of each predictor, and select the top ten predictors that appear most frequently. This is motivated by the variable importance [1].

2.3.2. *Two-way interactions selection.* The stepwise forward selection for interaction terms is adopted. However, instead of AIC or BIC, the selection is based the cross validation (CV) estimates of AUC and error. Let M_0 be initialized with the main effects model discussed in Section 2.3.1. Let V_0 be initialized with the set of interactions of two continuous or two categorical predictors. The best two-way interaction model is searched via Algorithm 1.

input: M_0 and V_0;
while $V_0 \neq \emptyset$ **do**
  $R \leftarrow \emptyset$;
  for $v \in V_0$ **do**
    $M_1 \leftarrow M_0 \cup \{v\}$; caculate M_1.err and M_1.auc by CV for 100 times;
    if M_1.err $<$ M_0.err and M_0.auc $<$ M_1.auc **then**
      $R \leftarrow R \cup \{(M_1, v)\}$;
    else
      continue;
    end
  end
  if $R \neq \emptyset$ **then**
    $(M_0, v_0) \leftarrow \arg\min_{(M,v)\in R}(M.\text{err} - M.\text{auc})$; $V_0 \leftarrow V_0 \setminus \{v_0\}$;
  else
    break;
  end
end
output: M_0

Algorithm 1: Selection of two-way interactions

3. Results

This section applies the proposed model to the whole training data and perform prediction on test data. Based on the results, we are able to give a complete solution to the problem addressed in Section 1.

3.1. **Test result.** The proposed model yields the AUC 0.709 and the misclassification error 0.631 on the test data. Note that the bias of CV estimator is approximately 0.01, which is reasonably small. Our model outperforms the other models on the test dataset, demonstrating a strong ability in identifying a potential policy cancellation. This result also justifies the choice of logit model and the proposed construction procedure in Section 2.

3.2. **Interpretation.** Some key statistics of model fitting are summarized in Table 1. The test result shows that the model well captures the structure of the data, namely, the logistic regression assumptions are reasonably satisfied. Hence, the significance test shown Table 1 should be reliable. The significance test with the marginality principle shows that all the predictors are important. In particular, we make several observations. (1) The policies sold by insurance brokers have higher retention rate. This suggests that the company should pay more attention on the improvement of online and phone customer experience. (2) The policyholder who claimed tends to cancel before the term ends. This may be because the experience of claiming insurance is under the expectation. (3) The young people, big family, and low-credit policyholders are more likely to cancel the policy during the effective term. (4) Tenure seems to have negative effects on retention rate, which should be further explored.

4. Conclusion

In this project, we explored various approaches to the retention modeling problem. The logistic regression demonstrates a strong prediction ability on the dataset. The final model, built based on the proposed procedure, outperforms other models on test data. With the help of this model, we are able to accurately identify a potential policy cancellation as well as understand what predictors are influential in causing a policy cancellation.

Due to time limitation, various predictor transformations and partitions are not fully explored. Further analysis of this dataset can be conducted in these aspects.

References

1. Friedman, J., Hastie, T., & Tibshirani, R. (2001). *The Elements of Statistical Learning.* New York: Springer.
2. Yang, Y. (2005). Can the strengths of AIC and BIC be shared? A conflict between model indentification and regression estimation. *Biometrika*, 92(4), 937-950.

Appendix A. Figure 1 and Table 1

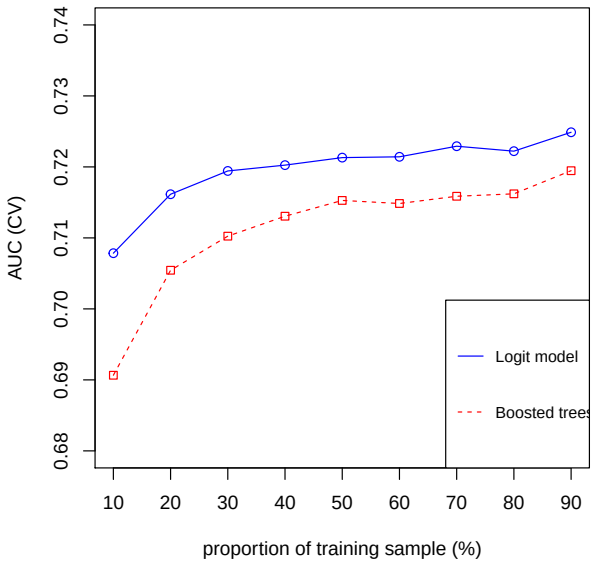

Figure 1. Simple logit model vs boosted trees

Table 1. Summary of final model fitting

	Estimate	Std. Error	Z value	$P(>\|Z\|)$
(Intercept)	−0.0964345	0.4095179	−0.235	0.813834
claim.ind = 1	0.3547357	0.0703431	5.043	< 0.001
sales.channel = Online	0.6104939	0.1031104	5.921	< 0.001
sales.channel = Phone	0.8308277	0.0628602	13.217	< 0.001
credit = low	1.5090241	0.0742303	20.329	< 0.001
credit = medium	0.6283415	0.0714722	8.791	< 0.001
zip.code = 2xxxx	1.3075537	0.1459212	8.961	< 0.001
zip.code = 5xxxx	0.3924349	0.1005980	3.901	< 0.001
zip.code = 8xxxx	0.4077199	0.0949045	4.296	< 0.001
zip.code = 9xxxx	0.5112482	0.1183128	4.321	< 0.001
n.adults	−0.0154627	0.0296857	−0.521	0.602450
n.children	0.0145507	0.0240757	0.604	0.545597
tenure	0.0378312	0.0063443	5.963	< 0.001
len.at.res	−0.0919500	0.0247059	−3.722	< 0.001
ni.age	−0.0538695	0.0090701	−5.939	< 0.001
ni.marital.status	−0.4807568	0.0712064	−6.752	< 0.001
n.adults:n.children	0.0528847	0.0093483	5.657	< 0.001
len.at.res:ni.age	0.0015699	0.0005479	2.865	0.004169

Appendix B. R codes

```r
library(ROCR)
library(stringi)

#####################################################
# preprocessing data
# remove missing values, remove ni.age >= 100

data.train = read.csv("pp_train.csv")
data.train = na.omit(data.train)
data.train = data.train[data.train$ni.age < 100,]
data.train = data.train[,1:17]

#####################################################
# this function calculates err

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

#####################################################
# 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)]
```

```
40        logit = glm(cancel ~ factor(claim.ind) + sales.channel +
    credit +
41        factor(zip.code) + n.adults + n.children + tenure +
42        len.at.res + ni.age + ni.marital.status + premium +
43        ni.gender + house.color + year + dwelling.type + coverage
    .type,
44        data = data.train[train,], family = binomial(link = "
    logit") )
45
46        temp = step(logit, direction = "both", trace = 0, k =
    lambda[j])
47        logit = eval(temp$call)
48        pred = predict(logit, data.train[-train,], type = "
    response")
49
50        input = prediction(pred, data.train[-train,]$cancel)
51        auc = performance(input, "auc")
52        cstat[i] = unlist(auc@y.values)
53
54        pred = ifelse(pred > 1/6,1,0)
55        err[i] = with(data.train[-train,], misclassification(
    cancel,pred))
56      }
57
58      cv.err[j] = mean(err)
59      cv.auc[j] = mean(cstat)
60    }
61
62    idx = which.min(cv.err - cv.auc)
63    return(lambda[idx])
64  }
65
66  # this function returns the counts of appearances of each
      predictors
67
68  L0Selection.count = function(cv.k = 500, cv.ratio = 0.2, lambda
      = 2){
69
70    c = numeric(16)
71    names(c) = c("factor(claim.ind)", "sales.channel", "credit",
      "factor(zip.code)",
72        "n.adults", "n.children", "tenure",  "len.at.res",
73        "ni.age","ni.marital.status", "premium", "ni.gender", "
    house.color",
```

```
74        "year", "dwelling.type", "coverage.type")
75
76    for (i in 1:cv.k) {
77
78      train = sample(1:n.data)[1:round(n.data*cv.ratio)]
79      logit = glm(cancel ~ factor(claim.ind) + sales.channel +
      credit +
80        factor(zip.code) + n.adults + n.children + tenure +
81        len.at.res + ni.age + ni.marital.status + premium +
82        ni.gender + house.color + year + dwelling.type + coverage
      .type,
83        data = data.train[train,], family = binomial(link = "
      logit") )
84        temp = step(logit, direction = "both", trace = 0, k =
      lambda)
85
86      str = paste(deparse(temp$call), collapse = '')
87      str = sub("~","@",str)
88      str = sub(",","@",str)
89      str = strsplit(str, split = "@")[[1]][2]
90      str = stri_replace_all_charclass(str, "\\p{WHITE_SPACE}", "
      ")
91      str = unlist(strsplit(str, split = "[+]"))
92
93      for(j in 1:16)
94        if(names(c)[j] %in% str)
95          c[j] = c[j] + 1
96    }
97    return(c)
98  }
99
100 ################################################
101 # step 2: greedy selection of two way interactions
102 # factor(claim.ind), sales.channel, credit, factor(zip.code),
      ni.marital.status
103 # n.adults, n.children, tenure, len.at.res, ni.age
104
105 V0 = c("factor(claim.ind):sales.channel", "factor(claim.ind):
      credit",
106       "factor(claim.ind):factor(zip.code)",
107       "factor(claim.ind):ni.marital.status",
108       "sales.channel:credit", "sales.channel:factor(zip.code)",
109       "sales.channel:ni.marital.status",
110       "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 = paste("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)
```

```
146         err[j] = with(data.train[-train,], misclassification(
    cancel,pred.c))
147       }
148 
149       M1.err = mean(err)
150       M1.auc = mean(cstat)
151 
152       if(M1.err < M0.err & M1.auc > M0.auc) {
153         R[i,1] = M1.err
154         R[i,2] = M1.auc
155       }
156 
157     }
158 
159     if( length(na.omit(R[,1])) > 0 ){
160       idx = which.min(R[,1] - R[,2])
161       M0.forluma = paste(M0.forluma, "+", V0[idx])
162       M0.err = R[idx,1]
163       M0.auc = R[idx,2]
164       V0 = V0[-idx]
165     }
166     else break
167   }
168   return(M0.forluma)
169 }
```