Actuarial Analytics Project 1

Tsung-Wei Chen

3/27/2020

Introduction

Aims

The goal of project is to explore dataset provided by The Insurance Company (TIC) Benchmark, which contains insureds' information. The data consists of 86 variables and includes product usage data and socio-demographic data derived from zip area codes. I need to predict the potential customers whether they are potentially interested in a caravan insurance policy or not.

QUESTION: Can you predict who would be interested in buying a caravan insurance policy and give an explanation why?

Data

There are three datasets in KDD archive at Irvine.

- TICDATA2000.txt: Dataset to train and validate prediction models and build a description (5822 customer records). Each record consists of 86 attributes, containing sociodemographic data (attribute 1-43) and product ownership (attributes 44- 86).The sociodemographic data is derived from zip codes. All customers living in areas with the same zip code have the same sociodemographic attributes. Attribute 86, "CARA- VAN:Number of mobile home policies", is the target variable.

- TICEVAL2000.txt: Dataset for predictions (4000 customer records). It has the same format as TICDATA2000.txt, only the target is missing. Participants are supposed to return the list of predicted targets only. All datasets are in tab delimited format.

- TICTGTS2000.txt: Targets for the evaluation set.

```
ticdata2000 <- read_table2("http://kdd.ics.uci.edu/databases/tic/ticdata2000.txt",
col_names = F)
ticeval2000 <- read_table2("http://kdd.ics.uci.edu/databases/tic/ticeval2000.txt",
col_names = F)
tictgts2000 <- read_table2("http://kdd.ics.uci.edu/databases/tic/tictgts2000.txt",
col_names = F)
colnames <- c("MOSTYPE", "MAANTHUI", "MGEMOMV", "MGEMLEEF", "MOSHOOFD", "MGODRK",
"MGODPR", "MGODOV", "MGODGE", "MRELGE", "MRELSA", "MRELOV", "MFALLEEN", "MFGEKIND",
"MFWEKIND", "MOPLHOOG", "MOPLMIDD", "MOPLLAAG", "MBERHOOG", "MBERZELF", "MBERBOER",
"MBERMIDD", "MBERARBG", "MBERARBO", "MSKA", "MSKB1", "MSKB2", "MSKC", "MSKD", "MHHUUR",
"MHKOOP", "MAUT1", "MAUT2", "MAUT0", "MZFONDS", "MZPART", "MINKM30", "MINK3045",
"MINK4575", "MINK7512", "MINK123M", "MINKGEM", "MKOOPKLA", "PWAPART", "PWABEDR",
"PWALAND", "PPERSAUT", "PBESAUT", "PMOTSCO", "PVRAAUT", "PAANHANG", "PTRACTOR", "PWERKT",
"PBROM", "PLEVEN", "PPERSONG", "PGEZONG", "PWAOREG", "PBRAND", "PZEILPL", "PPLEZIER",
"PFIETS", "PINBOED", "PBYSTAND", "AWAPART", "AWABEDR", "AWALAND", "APERSAUT", "ABESAUT",
"AMOTSCO", "AVRAAUT", "AAANHANG", "ATRACTOR", "AWERKT", "ABROM", "ALEVEN", "APERSONG",
"AGEZONG", "AWAOREG", "ABRAND", "AZEILPL", "APLEZIER", "AFIETS", "AINBOED", "ABYSTAND",
"CARAVAN")
```

```r
colnames(ticdata2000) <- colnames
colnames(ticeval2000) <- colnames[1:85]
colnames(tictgts2000) <- colnames[86]

# Check NA
ticdata2000 <- ticdata2000[complete.cases(ticdata2000),]
ticeval2000 <- ticeval2000[complete.cases(ticeval2000),]
```

Check the numbers and proportions of target variables, `CARAVAN`. This is an imblanaced data that around 6% of insureds have bought CARAVAN insurance policy.

```r
setDT(ticdata2000)[, .N, .(CARAVAN)][, Prop := round(N/sum(N),4)][]

##    CARAVAN    N   Prop
## 1:       0 5474 0.9402
## 2:       1  348 0.0598
```

Modeling

Logistic Regression

First, find correlations to exclude from the model. This function searches through a correlation matrix and returns a vector of integers corresponding to columns to remove to reduce pair-wise correlations.

```r
highcor <- findCorrelation(cor(ticdata2000), cutoff = .75, names = F)
ticdata2000_logit <- subset(ticdata2000, select = c(-highcor))
```

Second, model with `glm()` function.

```r
model_glm <- glm(CARAVAN ~ . , data = ticdata2000_logit, family = binomial(logit) )
summary_glm <- summary(model_glm)
summary_glm

##
## Call:
## glm(formula = CARAVAN ~ ., family = binomial(logit), data = ticdata2000_logit)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7343  -0.3714  -0.2657  -0.1887   3.2099
##
## Coefficients:
##               Estimate Std. Error z value             Pr(>|z|)
## (Intercept)  -7.366600   2.311631  -3.187              0.00144 **
## MAANTHUI     -0.172036   0.189217  -0.909              0.36325
## MGEMLEEF      0.236878   0.098376   2.408              0.01605 *
## MOSHOOFD      0.020673   0.029502   0.701              0.48348
## MGODRK       -0.105821   0.103505  -1.022              0.30660
## MGODPR       -0.023608   0.113784  -0.207              0.83564
## MGODOV       -0.003585   0.102202  -0.035              0.97202
## MGODGE       -0.053989   0.107221  -0.504              0.61460
## MRELSA       -0.074511   0.077976  -0.956              0.33929
## MRELOV       -0.043539   0.060902  -0.715              0.47466
## MFALLEEN     -0.027641   0.126198  -0.219              0.82663
## MFGEKIND     -0.069765   0.130692  -0.534              0.59347
## MFWEKIND     -0.020981   0.133685  -0.157              0.87529
## MOPLHOOG      0.010554   0.129285   0.082              0.93494
```

```
## MOPLMIDD     -0.079843   0.134724  -0.593             0.55342
## MOPLLAAG     -0.188800   0.135411  -1.394             0.16324
## MBERHOOG      0.098427   0.091456   1.076             0.28183
## MBERZELF      0.071079   0.097155   0.732             0.46441
## MBERBOER     -0.097590   0.107130  -0.911             0.36232
## MBERMIDD      0.157816   0.089803   1.757             0.07886 .
## MBERARBG      0.059221   0.089305   0.663             0.50724
## MBERARBO      0.106023   0.088912   1.192             0.23309
## MSKA          0.027646   0.101944   0.271             0.78624
## MSKB1        -0.004619   0.097943  -0.047             0.96239
## MSKB2         0.014029   0.088764   0.158             0.87442
## MSKC          0.110389   0.097575   1.131             0.25792
## MSKD         -0.021510   0.094420  -0.228             0.81979
## MHHUUR       -0.046118   0.026160  -1.763             0.07792 .
## MAUT1         0.199721   0.148267   1.347             0.17797
## MAUT2         0.175025   0.135473   1.292             0.19638
## MAUT0         0.116570   0.140747   0.828             0.40755
## MZFONDS       0.050856   0.042428   1.199             0.23067
## MINKM30       0.088167   0.096961   0.909             0.36319
## MINK3045      0.123799   0.094237   1.314             0.18895
## MINK4575      0.100522   0.094604   1.063             0.28799
## MINK7512      0.109628   0.100752   1.088             0.27655
## MINK123M     -0.177211   0.141999  -1.248             0.21204
## MINKGEM       0.092457   0.097046   0.953             0.34074
## MKOOPKLA      0.070226   0.044979   1.561             0.11845
## PWAPART       0.227595   0.073699   3.088             0.00201 **
## PWABEDR      -0.032476   0.194819  -0.167             0.86761
## PMOTSCO      -0.028212   0.063946  -0.441             0.65908
## PVRAAUT      -2.613628 129.191765  -0.020             0.98386
## PAANHANG      0.273752   0.259417   1.055             0.29131
## PWERKT       -4.943742 140.262024  -0.035             0.97188
## PLEVEN       -0.028635   0.056755  -0.505             0.61388
## PPERSONG     -0.197316   0.476371  -0.414             0.67872
## PGEZONG       0.202648   0.197595   1.026             0.30509
## PWAOREG       0.255897   0.107209   2.387             0.01699 *
## PPLEZIER      0.536239   0.116973   4.584          0.00000456 ***
## PBYSTAND      0.138275   0.092112   1.501             0.13332
## AWALAND      -0.627444   0.701798  -0.894             0.37129
## APERSAUT      0.792982   0.090933   8.721 < 0.0000000000000002 ***
## ABESAUT      -0.526409   0.790280  -0.666             0.50534
## ATRACTOR     -0.162629   0.416185  -0.391             0.69597
## ABROM        -0.550380   0.365350  -1.506             0.13195
## ABRAND        0.188770   0.133045   1.419             0.15594
## AZEILPL       0.876444   1.868149   0.469             0.63896
## AFIETS        0.428565   0.198090   2.163             0.03050 *
## AINBOED      -0.142423   0.588887  -0.242             0.80890
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2635.5  on 5821  degrees of freedom
## Residual deviance: 2322.8  on 5762  degrees of freedom
## AIC: 2442.8
```

```
##
## Number of Fisher Scoring iterations: 15
```

```r
print(paste0("The pseudo R square is: ", round( 1 - ( summary_glm$deviance /
summary_glm$null.deviance ), 2 )))
```

```
## [1] "The pseudo R square is: 0.12"
```

Third, a fast check on all the p-values of the variables and remove insignificant one, which are greater than `0.05` and model again.

```r
ticdata2000_logit <- ticdata2000_logit[,c("MGEMLEEF", "PWAPART", "PWAOREG", "PPLEZIER",
"APERSAUT", "AFIETS", "CARAVAN")]
model_glm_2 <- glm(CARAVAN ~ . , data = ticdata2000_logit, family = binomial(logit) )
summary_glm_2 <- summary(model_glm_2)
summary_glm_2
```

```
##
## Call:
## glm(formula = CARAVAN ~ ., family = binomial(logit), data = ticdata2000_logit)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8342  -0.3326  -0.3152  -0.2209   2.7436
##
## Coefficients:
##             Estimate Std. Error z value             Pr(>|z|)
## (Intercept) -3.81814    0.23655 -16.141 < 0.0000000000000002 ***
## MGEMLEEF     0.03901    0.06932   0.563              0.57364
## PWAPART      0.34585    0.05760   6.004        0.00000000192 ***
## PWAOREG      0.22081    0.09536   2.315              0.02059 *
## PPLEZIER     0.53366    0.10682   4.996        0.00000058538 ***
## APERSAUT     0.79493    0.08483   9.371 < 0.0000000000000002 ***
## AFIETS       0.54469    0.19423   2.804              0.00504 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2635.5  on 5821  degrees of freedom
## Residual deviance: 2463.5  on 5815  degrees of freedom
## AIC: 2477.5
##
## Number of Fisher Scoring iterations: 6
```

```r
print(paste0("The pseudo R square is: ", round( 1 - ( summary_glm_2$deviance /
summary_glm_2$null.deviance ), 2 )))
```

```
## [1] "The pseudo R square is: 0.07"
```

```r
ticdata2000_logit_final <- ticdata2000_logit[,c("PWAPART", "PWAOREG", "PPLEZIER",
"APERSAUT", "AFIETS", "CARAVAN")]
model_glm_3 <- glm(CARAVAN ~ . , data = ticdata2000_logit_final, family = binomial(logit)
)
summary_glm_3 <- summary(model_glm_3)
summary_glm_3
```

```
##
## Call:
## glm(formula = CARAVAN ~ ., family = binomial(logit), data = ticdata2000_logit_final)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8326  -0.3263  -0.3102  -0.2210   2.7294
##
## Coefficients:
##             Estimate Std. Error z value             Pr(>|z|)
## (Intercept) -3.70046    0.10879 -34.014 < 0.0000000000000002 ***
## PWAPART      0.34528    0.05759   5.995        0.00000000203 ***
## PWAOREG      0.22054    0.09539   2.312              0.02078 *
## PPLEZIER     0.53327    0.10677   4.994        0.00000059008 ***
## APERSAUT     0.79441    0.08487   9.360 < 0.0000000000000002 ***
## AFIETS       0.54875    0.19406   2.828              0.00469 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2635.5  on 5821  degrees of freedom
## Residual deviance: 2463.8  on 5816  degrees of freedom
## AIC: 2475.8
##
## Number of Fisher Scoring iterations: 6
```

```
print(paste0("The pseudo R square is: ", round( 1 - ( summary_glm_3$deviance /
summary_glm_3$null.deviance ), 3 )))
```

```
## [1] "The pseudo R square is: 0.065"
```

The nagelkerke() function of rcompanion package provides three types of Pseudo R-squared value (McFadden, Cox and Snell, and Cragg and Uhler) and Likelihood ratio test results. The McFadden Pseudo R-squared value is the commonly reported metric for binary logistic regression model fit.

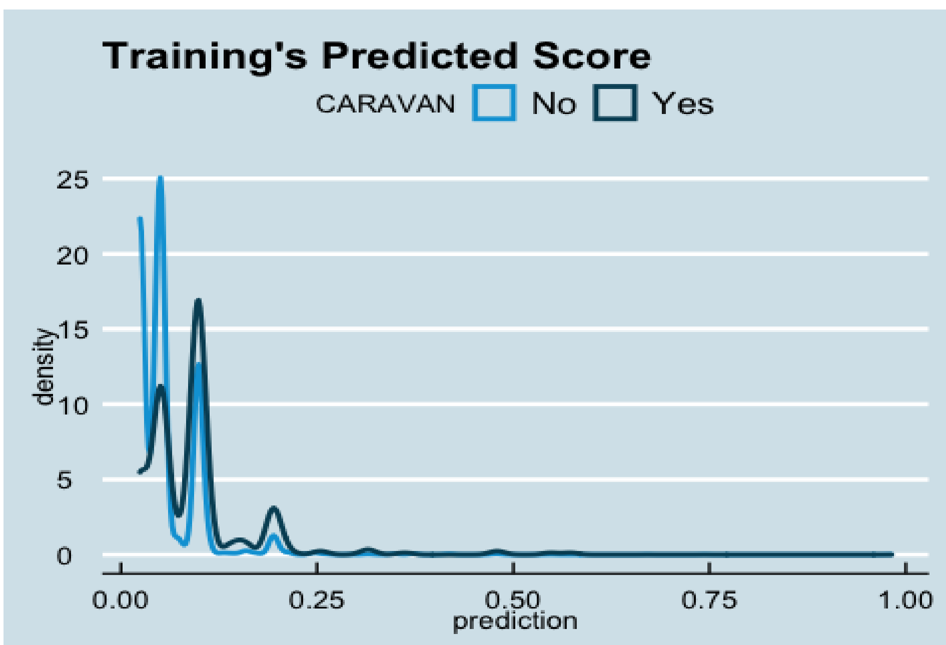
```
nagelkerke(model_glm_3)
```

```
## $Models
##
## Model: "glm, CARAVAN ~ ., binomial(logit), ticdata2000_logit_final"
## Null:  "glm, CARAVAN ~ 1, binomial(logit), ticdata2000_logit_final"
##
## $Pseudo.R.squared.for.model.vs.null
##                              Pseudo.R.squared
## McFadden                            0.0651570
## Cox and Snell (ML)                  0.0290649
## Nagelkerke (Cragg and Uhler)        0.0798306
##
## $Likelihood.ratio.test
##  Df.diff LogLik.diff  Chisq                                  p.value
##       -5     -85.862 171.72 0.00000000000000000000000000000000003128
##
## $Number.of.observations
##
## Model: 5822
```

```
## Null:  5822
##
## $Messages
## [1] "Note: For models fit with REML, these statistics are based on refitting with ML"
##
## $Warnings
## [1] "None"
```

Predicting whether customers are interested in insurance policy on both training and predicting set, and I'll perform an evaluation on the training set by plotting the probability (score). For a ideal double density plot, I want the distribution of scores to be separated, with the score of the "No" to be on the left and the score of the "Yes" to be on the right. However, both are skewed to the left.

```
ticeval2000_logit_final <- ticeval2000[,c("PWAPART", "PWAOREG", "PPLEZIER", "APERSAUT",
"AFIETS")]
# prediction
ticdata2000_logit_final$prediction <- predict( model_glm_3, newdata =
ticdata2000_logit_final, type = "response" )
ticeval2000_logit_final$prediction <- predict( model_glm_3, newdata =
ticeval2000_logit_final, type = "response" )

# distribution of the prediction score grouped by known outcome
ggplot( ticdata2000_logit_final, aes( prediction, color = as.factor(CARAVAN) ) ) +
  geom_density( size = 1 ) +
  ggtitle( "Training's Predicted Score" ) +
  scale_colour_economist( name = "CARAVAN", labels = c( "No", "Yes" ) ) +
  theme_economist()
```



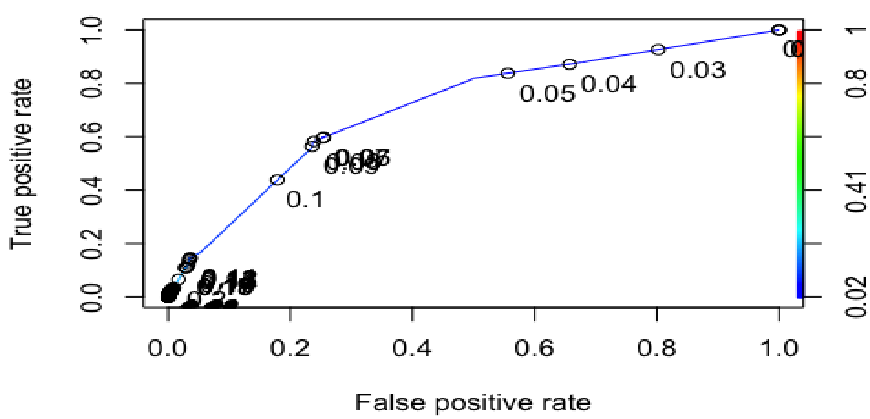
Accuracy is not the suitable indicator for the model on imbalanced dataset.

```
logit_test <- predict(model_glm_3, type = "response", newdata = ticdata2000_logit_final)
logit_roc_test <- roc(ticdata2000_logit_final$CARAVAN, logit_test, percent = T, positive
= '1')
auc(logit_roc_test)

## Area under the curve: 70.99%
```

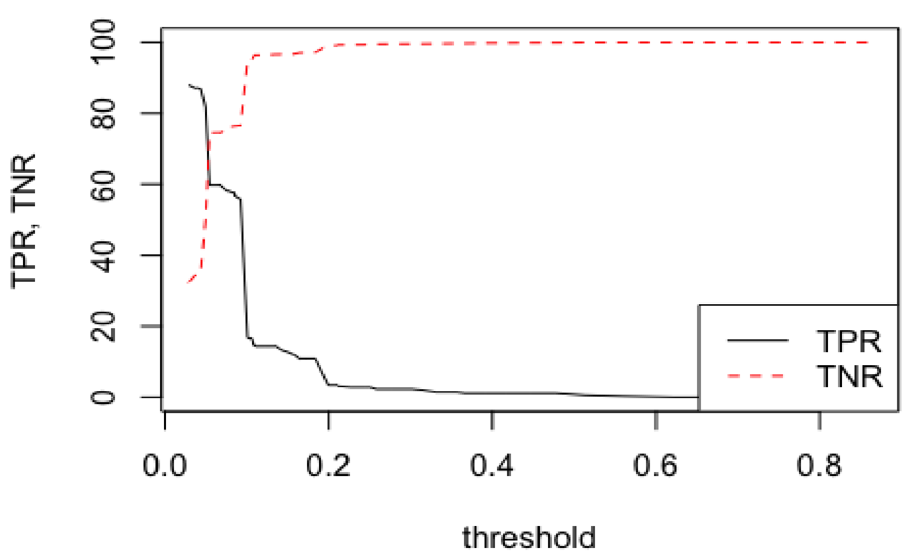
```
logit_bestthreshold <- coords(logit_roc_test, "best", "threshold", transpose = T)
logit_bestthreshold
```

```
##   threshold specificity sensitivity
##  0.07516712 76.10522470 58.33333333
```

```
plot(performance(ROCR::prediction(predictions = logit_test, labels =
ticdata2000_logit_final$CARAVAN), "tpr" , "fpr"),
     colorize = TRUE,
     print.cutoffs.at= seq(0,1,0.01),
     text.adj=c(-0.2,1.7))
```




```
matplot(data.frame(logit_roc_test$sensitivities, logit_roc_test$specificities), x =
logit_roc_test$thresholds, type='l', xlab = 'threshold', ylab='TPR, TNR')
legend('bottomright', legend=c('TPR', 'TNR'), lty=1:2, col=1:2)
```



I use 5%/6%/bestthreshold cutoff on training/tesing dataset to determine the final threshhold.

```
logit_cm_0.05_test <- confusionMatrix(data = factor(as.numeric(logit_test > 0.05)),
reference = factor(ticdata2000_logit_final$CARAVAN), positive = "1")
logit_cm_0.06_test <- confusionMatrix(data = factor(as.numeric(logit_test > 0.06)),
reference = factor(ticdata2000_logit_final$CARAVAN), positive = "1")
logit_cm_bestthreshold_test <- confusionMatrix(data = factor(as.numeric(logit_test >
logit_bestthreshold["threshold"])), reference = factor(ticdata2000_logit_final$CARAVAN),
positive = "1")
logit_cm_0.05_test$table

##           Reference
## Prediction    0    1
##          0 2728   63
##          1 2746  285

logit_cm_0.06_test$table

##           Reference
## Prediction    0    1
##          0 4081  140
##          1 1393  208

logit_cm_bestthreshold_test$table

##           Reference
## Prediction    0    1
##          0 4166  145
##          1 1308  203
```

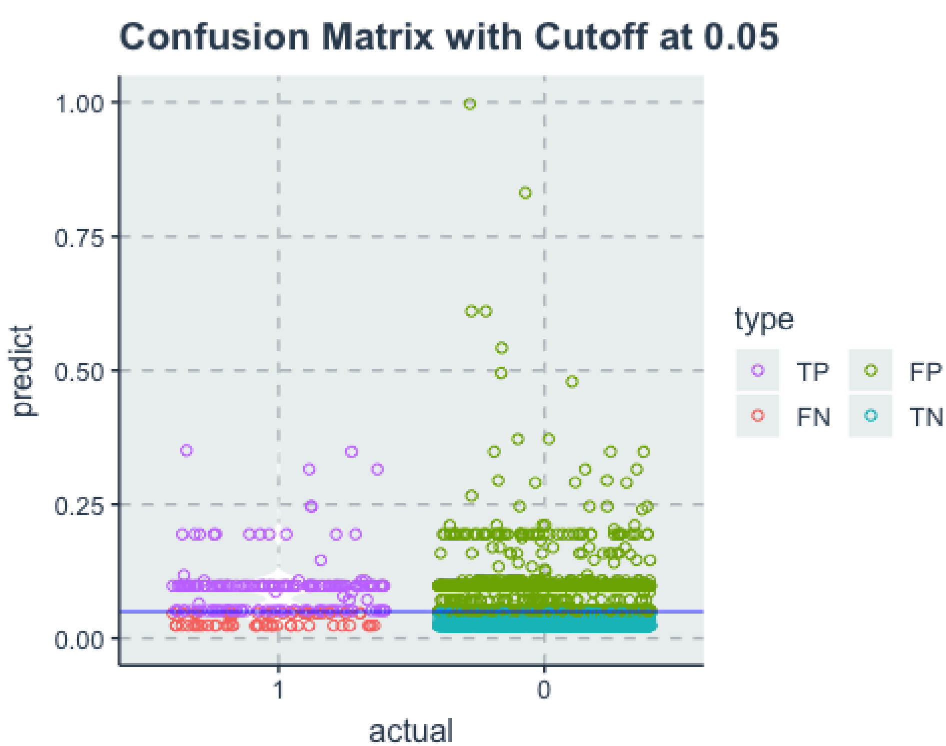
Next, predict on predicting dataset and compare with evaluating dataset.

```
logit_eval <- predict(model_glm_3, type = "response", newdata = ticeval2000_logit_final)
logit_roc_eval<- roc(tictgts2000$CARAVAN, logit_eval, percent = F, positive = '1')
auc(logit_roc_eval)

## Area under the curve: 0.6599

logit_cm_0.05_eval <- confusionMatrix(data = factor(as.numeric(logit_eval > 0.05)),
reference = factor(tictgts2000$CARAVAN), positive = "1")
logit_roc_0.05_eval <- roc(tictgts2000$CARAVAN, (as.numeric(logit_eval > 0.05)), positive
= 1)
logit_cm_0.06_eval <- confusionMatrix(data = factor(as.numeric(logit_eval > 0.06)),
reference = factor(tictgts2000$CARAVAN), positive = "1")
logit_roc_0.06_eval <- roc(tictgts2000$CARAVAN, (as.numeric(logit_eval > 0.06)), positive
= 1)
logit_cm_bestthreshold_eval <- confusionMatrix(data = factor(as.numeric(logit_eval >
logit_bestthreshold["threshold"])), reference = factor(tictgts2000$CARAVAN), positive =
"1")
logit_roc_bestthreshold_eval <- roc(tictgts2000$CARAVAN, (as.numeric(logit_eval >
logit_bestthreshold["threshold"])), positive = 1)
logit_cm_0.05_eval$table

##           Reference
## Prediction    0    1
##          0 1895   64
##          1 1867  174

auc(logit_roc_0.05_eval)
```

```
## Area under the curve: 0.6174

logit_cm_0.06_eval$table

##             Reference
## Prediction    0    1
##          0 2832  119
##          1  930  119

auc(logit_roc_0.06_eval)

## Area under the curve: 0.6264

logit_cm_bestthreshold_eval$table

##             Reference
## Prediction    0    1
##          0 2893  122
##          1  869  116

auc(logit_roc_bestthreshold_eval)

## Area under the curve: 0.6282
```

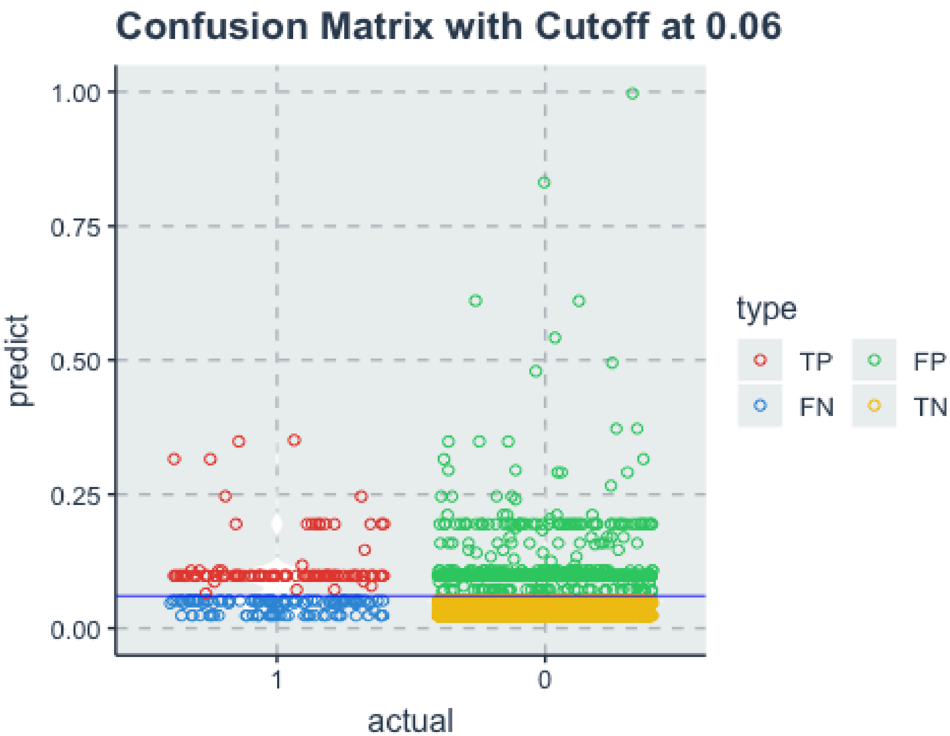
The plot below depicts the tradeoff when choosing a cutoff. If increasing the cutoff value, the number of true negative (TN) increases and the number of true positive (TP) decreases. If increasing the cutoff value, the number of false positive (FP) is lowered, while the number of false negative (FN) rises.

```
logit_cm_info <- ConfusionMatrixInfo(data = ticeval2000_logit_final, eval = tictgts2000,
predict = "prediction", actual = "CARAVAN", cutoff = 0.05)
ggthemr("flat")
logit_cm_info$plot
```




```
logit_cm_info <- ConfusionMatrixInfo(data = ticeval2000_logit_final, eval = tictgts2000,
predict = "prediction", actual = "CARAVAN", cutoff = 0.06)
```

```r
ggthemr("flat")
logit_cm_info$plot
```




```r
df <- data.table(threshold = c(0.05, 0.06, logit_bestthreshold["threshold"]), precision =
c(logit_cm_0.05_eval$byClass["Precision"], logit_cm_0.06_eval$byClass["Precision"],
logit_cm_bestthreshold_eval$byClass["Precision"]), recall =
c(logit_cm_0.05_eval$byClass["Recall"], logit_cm_0.06_eval$byClass["Recall"],
logit_cm_bestthreshold_eval$byClass["Recall"]), auc = c(auc(logit_roc_0.05_eval),
auc(logit_roc_0.06_eval), auc(logit_roc_bestthreshold_eval)), PredictedPurchasing =
c(logit_cm_0.05_eval$table[4], logit_cm_0.06_eval$table[4],
logit_cm_bestthreshold_eval$table[4]))
df[, `:=`(recall = round(recall,3), precision = round(precision,3), auc = round(auc,3))]
knitr::kable(df)
```

threshold	precision	recall	auc	PredictedPurchasing
0.0500000	0.085	0.731	0.617	174
0.0600000	0.113	0.500	0.626	119
0.0751671	0.118	0.487	0.628	116

Therefore, logistic model can correctly predict 174 customers in original 238 who are willing to buy the insurance policy with threshhold equals to 5%. The precision is around 8.5%, 174/(174+64) and the recall is around 73.1%.

To reduce FP, change threshold to 6%. The logistic model correctly predict 119 customers who are interested in CARAVAN policy with higher precision, 11.3%, but lower recall, 50%.

If the insurer wants more targeted clients those who are willing to buy CARAVAN without considering the costs, use threshold with 5%. With the prediction, insurer can target 174 customers and increase the profitability. However, if insurer needs to consider costs, threshhold with 6% may be better to consider.

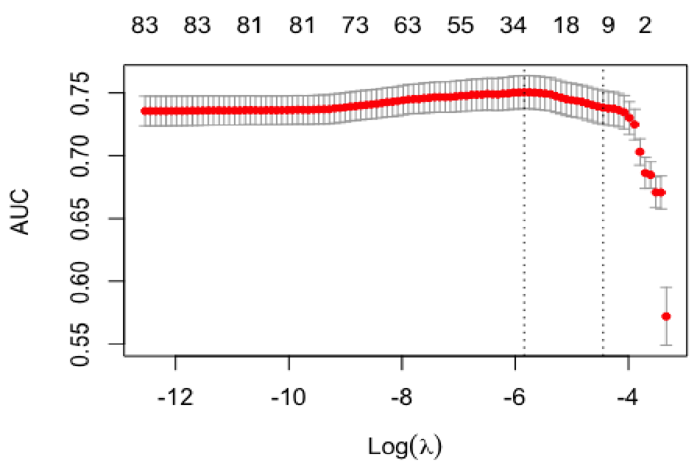
Penalized Logistic Regression

Penalized logistic regression imposes a penalty to the logistic model for having too many variables. This results in shrinking the coefficients of the less contributive variables toward zero, which is also known as regularization.

Lasso: alpha = 1

Least Absolute Shrinkage and Selection Operator (LASSO) creates a regression model that is penalized with the L1-norm which is the sum of the absolute coefficients. The coefficients of some less contributive variables are forced to be exactly zero. Only the most significant variables are kept in the final model.

```
set.seed(9080)
cv.lasso <- cv.glmnet(model.matrix(CARAVAN~., ticdata2000)[,-1], ticdata2000$CARAVAN,
alpha = 1, family = "binomial", nfolds = 20, type.measure = 'auc')
plot(cv.lasso)
```



The plot above displays the cross-validation area under curve based on the log of lambda. The left dashed vertical line indicates that the log of the optimal value of lambda is approximately -6, which is the one that maximizes the prediction auc. This lambda value will give the most accurate model.

```
cv.lasso$lambda.min

## [1] 0.00290187
```

Compute the final lasso model on training/testing dataset using lambda.min and use median/mean/bestthreshold to determine the threshold.

```
# Final model with lambda.min
lasso.model_min <- glmnet(model.matrix(CARAVAN~., ticdata2000)[,-1], ticdata2000$CARAVAN,
alpha = 1, family = "binomial", lambda = cv.lasso$lambda.min)

# Make prediction on test data
lasso_test <- predict(lasso.model_min, newx = as.matrix(ticdata2000[,1:85]))
lasso_roc_test <- roc(ticdata2000$CARAVAN, lasso_test, percent = T, positive = '1')
auc(lasso_roc_test)

## Area under the curve: 76.95%

lasso_bestthreshold <- coords(lasso_roc_test, "best", "threshold", transpose = T)
lasso_bestthreshold
```

```
##   threshold specificity sensitivity
##   -2.821137   64.139569   77.586207
```

```r
confusionMatrix(data = factor(as.numeric(lasso_test > median(lasso_test))), reference =
factor(ticdata2000$CARAVAN), positive = "1")$table
```

```
##           Reference
## Prediction    0    1
##          0 2853   58
##          1 2621  290
```

```r
confusionMatrix(data = factor(as.numeric(lasso_test > mean(lasso_test))), reference =
factor(ticdata2000$CARAVAN), positive = "1")$table
```

```
##           Reference
## Prediction    0    1
##          0 2916   58
##          1 2558  290
```

```r
confusionMatrix(data = factor(as.numeric(lasso_test > lasso_bestthreshold["threshold"])),
reference = factor(ticdata2000$CARAVAN), positive = "1")$table
```

```
##           Reference
## Prediction    0    1
##          0 3511   78
##          1 1963  270
```

As the result above, I choose to use average predicting probabilites of training/testing as threshold instead of `0.5` on imbalanced dataset.

Next, evaluting result displays below.

```r
lasso_eval <- predict(lasso.model_min, newx = as.matrix(ticeval2000))
lasso_cm_eval_mean <- confusionMatrix(data = factor(as.numeric(lasso_eval >
mean(lasso_test))), reference = factor(tictgts2000$CARAVAN), positive = "1")
lasso_roc_mean_eval <- roc(tictgts2000$CARAVAN, (as.numeric(lasso_eval >
mean(lasso_test))), positive = 1)
lasso_cm_eval_bestthreshold <- confusionMatrix(data = factor(as.numeric(lasso_eval >
lasso_bestthreshold["threshold"])), reference = factor(tictgts2000$CARAVAN), positive =
"1")
lasso_roc_bestthreshold_eval <- roc(tictgts2000$CARAVAN, (as.numeric(lasso_eval >
lasso_bestthreshold["threshold"])), positive = 1)
lasso_cm_eval_mean$table
```

```
##           Reference
## Prediction    0    1
##          0 2016   50
##          1 1746  188
```

```r
auc(lasso_roc_mean_eval)
```

```
## Area under the curve: 0.6629
```

```r
lasso_cm_eval_bestthreshold$table
```

```
##           Reference
## Prediction    0    1
```

```
##          0 2427   74
##          1 1335  164
```

```
auc(lasso_roc_bestthreshold_eval)
```

```
## Area under the curve: 0.6671
```

```
df_lasso <- data.table(threshold = c(round(mean(lasso_test),3),
round(lasso_bestthreshold["threshold"],3)), precision =
c(lasso_cm_eval_mean$byClass["Precision"],
lasso_cm_eval_bestthreshold$byClass["Precision"]), recall =
c(lasso_cm_eval_mean$byClass["Recall"], lasso_cm_eval_bestthreshold$byClass["Recall"]),
auc = c(auc(lasso_roc_mean_eval), auc(lasso_roc_bestthreshold_eval)), PredictedPurchasing
= c(lasso_cm_eval_mean$table[4], lasso_cm_eval_bestthreshold$table[4]))
df_lasso[, `:=`(recall = round(recall,3), precision = round(precision,3), auc =
round(auc,3))]
knitr::kable(df_lasso)
```

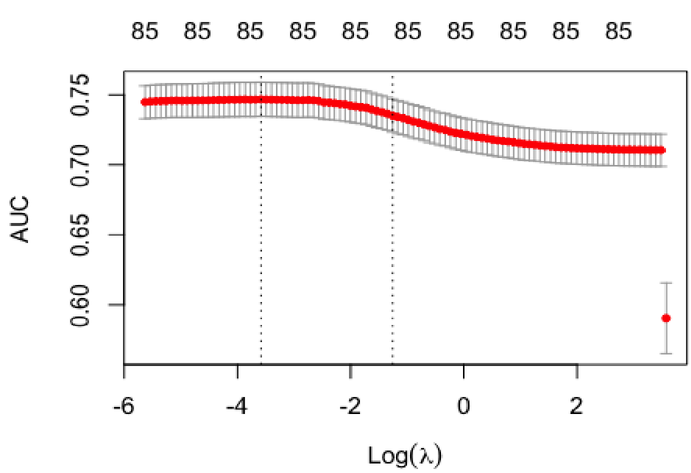
threshold	precision	recall	auc	PredictedPurchasing
-3.082	0.097	0.790	0.663	188
-2.821	0.109	0.689	0.667	164

Although the precision of lasso is not significantly better than logistic, the recall of lasso model is higher than logistic one. Moreover, the Lasso model correctly predict 188 in 238 customers. If insurer wants more targeted clients those who are willing to buy CARAVAN without considering the costs, lasso model is a good choice than logistic model on this imbalanced data.

Ridge: alpha = 0

Ridge Regression creates a linear regression model that is penalized with the L2-norm which is the sum of the squared coefficients. Variables with minor contribution have their coefficients close to zero. However, all the variables are incorporated in the model. This is useful when all variables need to be incorporated in the model according to domain knowledge.

```
set.seed(9080)
cv.ridge <- cv.glmnet(model.matrix(CARAVAN~., ticdata2000)[,-1], ticdata2000$CARAVAN,
alpha = 0, family = "binomial", nfolds = 20, type.measure = 'auc')
plot(cv.ridge)
```



The plot above displays the cross-validation area under curve based on the log of lambda. The left dashed vertical line indicates that the log of the optimal value of lambda is approximately -4, which is the one that maximizes the prediction auc. This lambda value will give the most accurate model.

```
cv.ridge$lambda.min

## [1] 0.02769975
```

Compute the final ridge model on training/testing dataset using lambda.min and use median/mean to determine the threshold.

```
# Final model with lambda.min
ridge.model_min <- glmnet(model.matrix(CARAVAN~., ticdata2000)[,-1], ticdata2000$CARAVAN,
alpha = 0, family = "binomial", lambda = cv.ridge$lambda.min)

# Make prediction on test data
ridge_test <- predict(ridge.model_min, newx = as.matrix(ticdata2000[,1:85]))
ridge_roc_test <- roc(ticdata2000$CARAVAN, ridge_test, percent = T, positive = '1')
auc(ridge_roc_test)

## Area under the curve: 77.52%

ridge_bestthreshold <- coords(ridge_roc_test, "best", "threshold", transpose = T)
ridge_bestthreshold

##   threshold specificity sensitivity
##   -2.509975   77.475338   65.229885

confusionMatrix(data = factor(as.numeric(ridge_test > median(ridge_test))), reference =
factor(ticdata2000$CARAVAN), positive = "1")$table

##           Reference
## Prediction    0    1
##          0 2857   54
##          1 2617  294

confusionMatrix(data = factor(as.numeric(ridge_test > median(ridge_test))), reference =
factor(ticdata2000$CARAVAN), positive = "1")$byClass["F1"]

##        F1
## 0.1804234

confusionMatrix(data = factor(as.numeric(ridge_test > mean(ridge_test))), reference =
factor(ticdata2000$CARAVAN), positive = "1")$table

##           Reference
## Prediction    0    1
##          0 2949   57
##          1 2525  291

confusionMatrix(data = factor(as.numeric(ridge_test > mean(ridge_test))), reference =
factor(ticdata2000$CARAVAN), positive = "1")$byClass["F1"]

##        F1
## 0.1839444

confusionMatrix(data = factor(as.numeric(ridge_test > ridge_bestthreshold)), reference =
factor(ticdata2000$CARAVAN), positive = "1")$table
```

```
##           Reference
## Prediction    0    1
##          0 5051  268
##          1  423   80
```

```r
confusionMatrix(data = factor(as.numeric(ridge_test > ridge_bestthreshold)), reference =
factor(ticdata2000$CARAVAN), positive = "1")$byClass["F1"]
```

```
##        F1
## 0.1880141
```

As the result above, I choose to use average predicting probabilites of training/testing as threshold instead of `0.5` on imbalanced dataset because of the higher F1 score. Although using best threshold will get higher F1 score, the TP is too low.

Next, evaluting result displays below.

```r
ridge_eval <- predict(ridge.model_min, newx = as.matrix(ticeval2000))
ridge_cm_eval_mean <- confusionMatrix(data = factor(as.numeric(ridge_eval >
mean(ridge_test))), reference = factor(tictgts2000$CARAVAN), positive = "1")
ridge_cm_eval_mean$table
```

```
##           Reference
## Prediction    0    1
##          0 2028   47
##          1 1734  191
```

```r
df_ridge <- data.table(threshold = c(round(mean(ridge_test),3)), precision =
c(ridge_cm_eval_mean$byClass["Precision"]), recall =
c(ridge_cm_eval_mean$byClass["Recall"]), auc = c(roc(factor(tictgts2000$CARAVAN),
as.numeric(ridge_eval))$auc), PredictedPurchasing = c(ridge_cm_eval_mean$table[4]))
df_ridge[, `:=`(recall = round(recall,3), precision = round(precision,3), auc =
round(auc,3))]
knitr::kable(df_ridge)
```

threshold	precision	recall	auc	PredictedPurchasing
-3.062	0.099	0.803	0.726	191

The precision and the recall of Ridge model is higher than Lasso one. Moreover, the ridge model correctly predict `191` in `238` customers. If insurer wants more targeted clients those who are willing to buy CARAVAN without considering the costs, ridge model is a good choice than lasso and logistic model.

Elastic Net: 0 < alpha < 1

Elastic Net produces a regression model that is penalized with both the L1-norm and L2-norm. The consequence of this is to effectively shrink coefficients (like in ridge regression) and to set some coefficients to zero (as in LASSO).

```r
# ELASTIC NET WITH 0 < ALPHA < 1
set.seed(9080)
registerDoParallel(cores = 4)
search <- foreach(i = seq(0.1, 0.9, 0.05), .combine = rbind) %dopar% {
  cv <- cv.glmnet(model.matrix(CARAVAN~., ticdata2000)[,-1], ticdata2000$CARAVAN, family
= "binomial", nfold = 10, type.measure = "auc", paralle = TRUE, alpha = i)
  data.frame(cvm = cv$cvm[cv$lambda == cv$lambda.min], lambda.min = cv$lambda.min, alpha
= i)
}
```

```
cv.elasticnet <- search[search$cvm == min(search$cvm), ]
elasticnet.model_min <- glmnet(model.matrix(CARAVAN~., ticdata2000)[,-1],
ticdata2000$CARAVAN, family = "binomial", lambda = cv.elasticnet$lambda.min, alpha =
cv.elasticnet$alpha)

# Make prediction on test data
elasticnet_test <- predict(elasticnet.model_min, newx = as.matrix(ticdata2000[,1:85]))
elasticnet_roc_test <- roc(ticdata2000$CARAVAN, elasticnet_test, percent = T, positive =
'1')
auc(elasticnet_roc_test)

## Area under the curve: 76.96%

elasticnet_bestthreshold <- coords(elasticnet_roc_test, "best", "threshold", transpose =
T)
elasticnet_bestthreshold

##   threshold specificity sensitivity
##   -2.809245   64.523201   77.298851

confusionMatrix(data = factor(as.numeric(elasticnet_test > median(elasticnet_test))),
reference = factor(ticdata2000$CARAVAN), positive = "1")$table

##           Reference
## Prediction    0    1
##          0 2854   57
##          1 2620  291

confusionMatrix(data = factor(as.numeric(elasticnet_test > median(elasticnet_test))),
reference = factor(ticdata2000$CARAVAN), positive = "1")$byClass["F1"]

##        F1
## 0.1785824

confusionMatrix(data = factor(as.numeric(elasticnet_test > mean(elasticnet_test))),
reference = factor(ticdata2000$CARAVAN), positive = "1")$table

##           Reference
## Prediction    0    1
##          0 2920   58
##          1 2554  290

confusionMatrix(data = factor(as.numeric(elasticnet_test > mean(elasticnet_test))),
reference = factor(ticdata2000$CARAVAN), positive = "1")$byClass["F1"]

##        F1
## 0.1817043

confusionMatrix(data = factor(as.numeric(elasticnet_test > elasticnet_bestthreshold)),
reference = factor(ticdata2000$CARAVAN), positive = "1")$table

##           Reference
## Prediction    0    1
##          0 4814  255
##          1  660   93

confusionMatrix(data = factor(as.numeric(elasticnet_test > elasticnet_bestthreshold)),
reference = factor(ticdata2000$CARAVAN), positive = "1")$byClass["F1"]
```

```
##        F1
## 0.1689373
```

As the result above, I choose to use average predicting probabilites of training/testing as threshold instead of `0.5` on imbalanced dataset because of the higher F1 score.

Next, evaluting result displays below.

```
elasticnet_eval <- predict(elasticnet.model_min, newx = as.matrix(ticeval2000))
elasticnet_cm_eval_mean <- confusionMatrix(data = factor(as.numeric(elasticnet_eval >
mean(elasticnet_test))), reference = factor(tictgts2000$CARAVAN), positive = "1")
elasticnet_cm_eval_mean$table

##           Reference
## Prediction    0    1
##          0 2012   50
##          1 1750  188

df_elasticnet <- data.table(threshold = c(round(mean(elasticnet_test),3)), precision =
c(elasticnet_cm_eval_mean$byClass["Precision"]), recall =
c(elasticnet_cm_eval_mean$byClass["Recall"]), auc = c(roc(factor(tictgts2000$CARAVAN),
as.numeric(elasticnet_eval))$auc), PredictedPurchasing =
c(elasticnet_cm_eval_mean$table[4]))
df_elasticnet[, `:=`(recall = round(recall,3), precision = round(precision,3), auc =
round(auc,3))]
knitr::kable(df_elasticnet)
```

threshold	precision	recall	auc	PredictedPurchasing
-3.077	0.097	0.79	0.724	188

Conclusion

Summary for GLM model: After comparison among logistic, lasso, ridge, and elastic net models, I will recommend insurance company to use *Ridge model* on these clients with marketing emails. Insurer can send the email to them with minor costs and successfully targets more than 190 CARAVAN prospects in 238's.

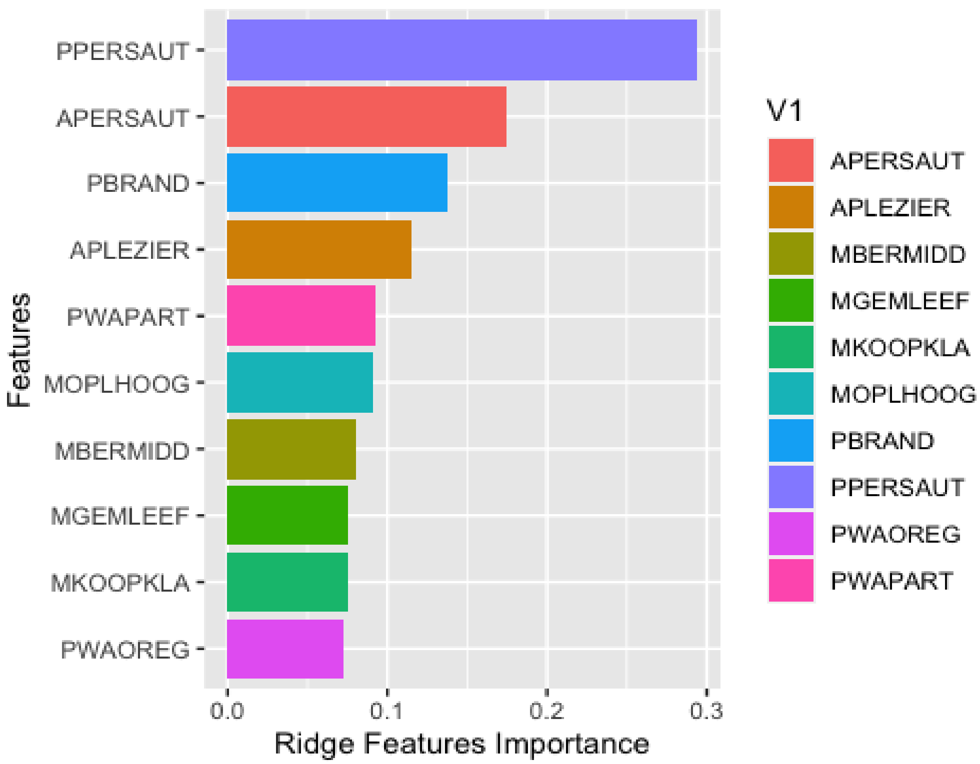
```
knitr::kable(rbindlist(sapply(list(Logistic = df, Lasso = df_lasso, Ridge = df_ridge,
"Elastic Net" = df_elasticnet), rbind, simplify = F), idcol = "Model"))
```

Model	threshold	precision	recall	auc	PredictedPurchasing
Logistic	0.0500000	0.085	0.731	0.617	174
Logistic	0.0600000	0.113	0.500	0.626	119
Logistic	0.0751671	0.118	0.487	0.628	116
Lasso	-3.0820000	0.097	0.790	0.663	188
Lasso	-2.8210000	0.109	0.689	0.667	164
Ridge	-3.0620000	0.099	0.803	0.726	191
Elastic Net	-3.0770000	0.097	0.790	0.724	188

Final Confusion Matrix and Performance of Ridge Model: successfully predicting `191` potential customers.

```
##           Reference
## Prediction    0    1
##          0 2028   47
##          1 1734  191

##          Sensitivity          Specificity       Pos Pred Value
##           0.80252101           0.53907496           0.09922078
##       Neg Pred Value            Precision               Recall
##           0.97734940           0.09922078           0.80252101
##                   F1           Prevalence       Detection Rate
##           0.17660656           0.05950000           0.04775000
## Detection Prevalence    Balanced Accuracy
##           0.48125000           0.67079798
```

Top 10 important features of Ridge model after standardization:



Reference

1. Penalized Logistic Regression Essentials in R: Ridge, Lasso and Elastic Net

2. Penalized Regression in R by Jason Brownlee on July 25, 2014 in R Machine Learning

3. Variable Selection with Elastic Net

4. Variable importance from GLMNET