Spring 2019

Predictive Analytics

Mahbubul Hasan (mhasan1)

Elizabeth James (enjames)

Omer Khan (oakhan)

Jennifer Tichenor (jtchnor2)

MIS 7621 - 001

Character strings within the data were factored in order for predictive analytics to be calculated. Of 300 observations, ninety percent of the data was used to create a training set; the other ten percent was set aside as a test set. Analysis was completed via ten-fold cross validation, decision tree modeling, logistic regression, SVM, and Naïve Bayes classification.

For all outputs:

* Precision indicates the fraction of predicted non-subscribers that are actually non-subscribers.
* Recall indicates the fraction of actual non-subscribers that were predicted correctly.
* Overall Accuracy indicates the overall percentage of correct predictions.

# Decision Tree

The decision predicted that there would be no subscribers through both the training and test data.

|  |  |  |  |
| --- | --- | --- | --- |
| Training Set | | Predicted | |
|  |  | subNo | subYes |
| Actual | subNo | 234 | 0 |
| subYes | 36 | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Set | | Predicted | |
|  |  | subNo | subYes |
| Actual | subNo | 26 | 0 |
| subYes | 4 | 0 |

|  |  |  |
| --- | --- | --- |
|  | Train | Test |
| Precision | 1 | 1 |
| Recall | 0.8667 | 0.8667 |
| OA\* | 0.8667 | 0.8667 |

\*Overall Accuracy

# Logistic Regression

All data was used to predict whether an individual would be a subscriber. In five iterations, the model determined that the model that best fit the training data had the following coefficients; the stand:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Coefficients: | Estimate(q) | Std.Error | zValue | Pr(>|z|) |
| (Intercept) | -2.21E+00 | 1.51E+00 | -1.458 | 0.1449 |
| age | 3.94E-02 | 3.49E-02 | 1.129 | 0.2589 |
| genderMale | -4.21E-03 | 3.76E-01 | -0.011 | 0.9911 |
| income | -1.30E-05 | 1.28E-05 | -1.014 | 0.3106 |
| kids | -8.06E-03 | 1.57E-01 | -0.052 | 0.9589 |
| ownHomeownYes | 2.47E-01 | 4.08E-01 | 0.605 | 0.5452 |
| SegmentSuburbMix | -1.41E+00 | 5.61E-01 | -2.514 | 0.0119 |
| SegmentTravelers | -1.51E+00 | 1.00E+00 | -1.509 | 0.1314 |
| SegmentUrbanHip | 1.44E-01 | 7.76E-01 | 0.186 | 0.8526 |

This indicates that for each variable, the odds of an individual subscribing to cable increase by exp(q) where ***q*** is equal to the estimate of the coefficient. Assuming a predictive probability threshold of 0.24, the following predictions were made and compared to the actual output.

|  |  |  |  |
| --- | --- | --- | --- |
| Logistic Results | | Actual | |
| Predicted |  | subNo | subYes |
| subNo | 23 | 4 |
| subYes | 3 | 0 |

|  |  |
| --- | --- |
| Logistic Output | |
| Precision | 0.85185 |
| Recall | 0.88462 |
| OA\* | 0.76667 |

\*Overall Accuracy

# Cross Validation

K-fold validation is used to partition data into equal parts so that each part is used as a test set for the remaining data to aid in machine learning and, hopefully, production of a better model. Using 10 as the k-value, the initial run offers the ROC curve, sensitivity, and specificity measures.

|  |  |  |  |
| --- | --- | --- | --- |
| Cross Validation | ROC | Sensitivity | Specificity |
| Fold01 | 0.5 | 1 | 0 |
| Fold02 | 0.5 | 1 | 0 |
| Fold03 | 0.5 | 1 | 0 |
| Fold04 | 0.5 | 1 | 0 |
| Fold05 | 0.5 | 1 | 0 |
| Fold06 | 0.5 | 1 | 0 |
| Fold07 | 0.5 | 1 | 0 |
| Fold08 | 0.5 | 1 | 0 |
| Fold09 | 0.5 | 1 | 0 |
| Fold10 | 0.5 | 1 | 0 |

* Sensitivity is the measure of correctly identified non-subscribers.
* Specificity is the measure of correctly identified subscribers.

To compare the cross validation to other models, the data is translated to precision and recall values.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | AUC | Precision | Recall | F-score |
| Fold01 | 0 | 0.8666667 | 1 | 0.9285714 |
| Fold02 | 0 | 0.8666667 | 1 | 0.9285714 |
| Fold03 | 0 | 0.8666667 | 1 | 0.9285714 |
| Fold04 | 0 | 0.8666667 | 1 | 0.9285714 |
| Fold05 | 0 | 0.8666667 | 1 | 0.9285714 |
| Fold06 | 0 | 0.8666667 | 1 | 0.9285714 |
| Fold07 | 0 | 0.8666667 | 1 | 0.9285714 |
| Fold08 | 0 | 0.8666667 | 1 | 0.9285714 |
| Fold09 | 0 | 0.8666667 | 1 | 0.9285714 |
| Fold10 | 0 | 0.8666667 | 1 | 0.9285714 |

# SVM

Support Vector Machine models are used to divide data to one category or another by use of a line or gap and is a simple choice for binary (yes or no) predictions. New data is then mapped and assigned based on the relative position to the dividing gap. Three SVM models were used:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Actual | |
|  |  | subNo | subYes |
| Radial | subNo | 260 | 40 |
| subYes | 0 | 0 |
| Polynomial | subNo | 260 | 40 |
| subYes | 0 | 0 |
| Sigmoid | subNo | 251 | 39 |
| subYes | 9 | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Radial | Polynomial | Sigmoid |
| Precision | 0.8667 | 0.8667 | 0.8655 |
| Recall | 1 | 1 | 0.9654 |
| OA\* | 0.8667 | 0.8667 | 0.84 |

\*Overall Accuracy

# Naïve Bayes

Naive Bayes classifiers assume that the presence of a particular feature in a class is unrelated to the presence of any other feature. Using the training data to create the model, the model evaluated the test data as all refusing subscriptions.

|  |  |  |  |
| --- | --- | --- | --- |
| NB Test | | Actual | |
| Predicted |  | subNo | subYes |
| subNo | 27 | 3 |
| subYes | 0 | 0 |

|  |  |
| --- | --- |
| NB Output | |
| Precision | 0.9 |
| Recall | 1 |
| OA\* | 0.9 |

\*Overall accuracy

# Supervised Learning Conclusion

The only models to predict any yes subscribers were logistic regression and SVM (Sigmoid). Though both models presented with the lowest overall accuracy measures, 77% and 84% respectively, they are the most likely to provide relatively realistic predictions.

# Appendix

R-code for this portion of the report only.

#begin supervised learning

library(party)

CableTV5 <- subset(CableTVSubscriber2, select = c(1:5,7,6))

CableTV5$gender <- factor(CableTV5$gender, c("Female", "Male"))

CableTV5$ownHome <- factor(CableTV5$ownHome, c("ownNo", "ownYes"))

CableTV5$Segment <- factor(CableTV5$Segment, c("Moving up", "Suburb mix", "Travelers", "Urban hip"))

CableTV5$subscribe <- factor(CableTV5$subscribe, c("subNo", "subYes"))

head(CableTV5)

#train 90% test 10% no replacement

sampleSize <- sample(nrow(CableTV5), floor(.9\*nrow(CableTV5)), replace = F, prob = NULL)

trainCableTV <- CableTV5[sampleSize,]

testCableTV <- CableTV5[-sampleSize,]

#decision tree using all variable to predict subscribe

cableTree <- ctree(subscribe ~ ., data = trainCableTV)

plot(cableTree)

#run training data

CableTreeTrain <- predict(cableTree, trainCableTV)

summary(CableTreeTrain)

predictCableTreeTrain <- table(trainCableTV$subscribe, CableTreeTrain)

predictCableTreeTrain

precision(predictCableTreeTrain)

recall(predictCableTreeTrain)

#accuracy

sum(diag(predictCableTreeTrain))/sum(predictCableTreeTrain)

#run test data

CableTreeTest <- predict(cableTree, testCableTV)

summary(CableTreeTest)

predictCableTreeTest <- table(testCableTV$subscribe, CableTreeTest)

predictCableTreeTest

precision(predictCableTreeTest)

recall(predictCableTreeTest)

#accuracy

sum(diag(predictCableTreeTest))/sum(predictCableTreeTest)

#logistic regression using same train/test sets - rename as train2 and test 2 for reference

#save test2$subscribe to response.test; change test2$subscribe to null

head(CableTVSubscriber2)

train2 <- trainCableTV

head(train2)

test2 <- testCableTV

response.test <- test2$subscribe

test2$subscribe <- as.null(test2$subscribe)

library(stats)

model <- glm(subscribe ~ ., family = binomial, data = train2)

summary(model)

predicted.probability <- predict(model, newdata = data.frame(test2), type = "response")

predict.binary.output <- ifelse(predicted.probability >= .24,"subYes","subNo")

#.24 used because rounding at any % higher was causing all predictions to subNo

# resulting in error in precision and recall

logTableTest <- table(predict.binary.output, response.test)

logTableTest

precision(logTableTest)

recall(logTableTest)

#accuracy

sum(diag(logTableTest))/sum(logTableTest)

#10fold

library(pROC)

library(MLmetrics)

ctrl <- trainControl(method = "cv", number = 10, savePredictions = T, classProbs = T, summaryFunction = twoClassSummary)

mod <- train(subscribe ~ ., method = 'C5.0Tree', metric="ROC",data = CableTV5,trControl = ctrl)

names(mod)

mod$results

mod$resample

ctrl <- trainControl(method = "cv", number = 10, savePredictions = T, classProbs = T, summaryFunction = prSummary)

mod2 <- train(subscribe ~ ., method = 'C5.0Tree', metric="ROC",data = CableTV5,trControl = ctrl)

names(mod2)

mod2$results

mod2$resample

summary(mod2)

#svm

#read all tables as predicted rows; actual columns

library(e1071)

model <- svm(subscribe~., data=CableTV5)

print(model)

modelTable <- table(model$fitted, CableTV5$subscribe)

modelTable

model1 <- svm(subscribe~., data=CableTV5, kernel = "polynomial")

print(model1)

model1Table <- table(model1$fitted, CableTV5$subscribe)

model1Table

model2 <- svm(subscribe~., data=CableTV5, kernel = "sigmoid")

print(model2)

model2Table <- table(model2$fitted, CableTV5$subscribe)

model2Table

#for model

precision(modelTable)

recall(modelTable)

#accuracy

sum(diag(modelTable))/sum(modelTable)

#for model1

precision(model1Table)

recall(model1Table)

#accuracy

sum(diag(model1Table))/sum(model1Table)

#for model2

precision(model2Table)

recall(model2Table)

#accuracy

sum(diag(model2Table))/sum(model2Table)

#sigmoid is giving the most realistic output

#NaiveBayes

x <- subset(trainCableTV, select = -subscribe)

y <- trainCableTV$subscribe

model\_NB <- naiveBayes(x,y)

print(model\_NB)

NB\_prediction <- predict(model\_NB, testCableTV)

table(NB\_prediction)

NBoutcomes <- table(NB\_prediction, testCableTV$subscribe)

precision(NBoutcomes)

recall(NBoutcomes)

#accuracy

sum(diag(NBoutcomes))/sum(NBoutcomes)