Predicting the outcome of a sale during Amazon.ca customer sessions: SVM and random forest modelling

Cimmaron Yeoman

2023-04-13

Introduction: Amazon purchases e-commerce data set

This is an e-commerce data set examining different variables related to Amazon.ca browsing sessions and purchases. The response variable, **Revenue**, indicates whether or a purchase and sale were made. Some of the predictor variables include: **Month**, **SpecialDay** (whether or not the purchase was made near a holiday), and **BounceRates** (visitors who visit a page and then leave, with no purchase or other action performed). Other variables describe visitor characteristics, geography, Google Analytics, etc.

The data set contains **12,330** observations of online shopping sessions. From these sessions, only **1,908** involved a purchase and sale, while **10,422** sessions did not. This is a sale rate of about **15.5%**. The response variable **Revenue** indicates whether or not an online shopping session resulted in a sale or not. Using non-linear Support Vector Machine (SVM) and random forest modelling, this report will attempt to predict the outcome of sale, based on the combinations of the predictor variables for each Amazon.ca browsing session.

Importing and cleaning data

The data set was imported into RStudio and named **shoppers**. The response variable **Revenue** was originally stored as TRUE/FAlSE but this was changed to a numeric 0/1 input, and stored as a factor. The predictor variable **Month** was included in the data set with abbreviated character names, which were changed to a numeric value between 1 and 12, and stored as a factor (January = 1, April = 4, etc.). The **VisitorType** variable was organized into "Returning_Visitor", "New_Visitor", or "Other". This was changed into numeric equivalents of 1, 2, or 3, and stored as a factor. Finally, the **Weekend** TRUE/FALSE variable was changed into a 0/1 format, and stored as a factor. This was done to make the data more uniform for data analysis. Generally, I prefer that data is organized in this way as well.

```
shoppers <- read_csv("online_shoppers_intention.csv")
shoppers$Revenue <- as.numeric(shoppers$Revenue)
shoppers$Revenue <- as.factor(shoppers$Revenue)
shoppers$Month <- as.integer(factor(shoppers$Month,
                  levels = c("Jan", "Feb", "Mar", "Apr",
                  "May", "June", "Jul", "Aug",
                  "Sep", "Oct", "Nov", "Dec")))
shoppers$Month <- as.factor(shoppers$Month)
VTlevels <- factor(c("Returning_Visitor", "New_Visitor", "Other"))
shoppers$VisitorType <- as.integer(factor(shoppers$VisitorType,
                                          levels = VTlevels))
shoppers$VisitorType <- as.factor(shoppers$VisitorType)
shoppers$Weekend <- as.numeric(shoppers$Weekend)
shoppers$Weekend <- as.factor(shoppers$Weekend)
```

Splitting into train and test sets

The **shoppers** data set was organized into train and test subsets. The train set had **9864** observations and **18** variables. The test set had **2466** observations and **18** variables. This was an 80/20 percent split, as the data set was quite large.

```
set.seed(88)
index <- 1:nrow(shoppers)
N <- trunc(length(index)/5)
testind <- sample(index, N)
test_st <- shoppers[testind,]
train_st <- shoppers[-testind,]
```

Specifying x and y for models

Before creating a support vector machine model, the test/train sets were grouped and the x/y values were specified.

```
int_1 <- data.frame (
  x = train_st[,-18], #dropping the 18th variable
  y = train_st$Revenue #specifying response
)
int_2 <- data.frame (
  x = test_st[,-18],
  y = test_st$Revenue
)
```

SVM models

I created several radial models with the **int_1** training set, using different cost and gamma values. A linear model would not run, and a polynomial model seemed to take a long time to compute. A radial model was the most practical.

```
library(e1071)
library(caret)
set.seed(88)
smod1 <- svm(y~., data = int_1, kernel = 'radial', cost = 10, gamma = 10)
smod2 <- svm(y~., data = int_1, kernel = 'radial', cost = 1, gamma = 1)
smod3 <- svm(y~., data = int_1, kernel = 'radial', cost = 1e5, gamma = 0.1)
smod4 <- svm(y~., data = int_1, kernel = 'radial', cost = 1e4, gamma = 0.1)
smod5 <- svm(y~., data = int_1, kernel = 'radial', cost = 1e5, gamma = 1)
```

The **smod1** model had **9690** support vectors, and the **smod2** model had over **7000** support vectors. The **smod3** and **smod4** models with the higher cost values and lower gamma both returned under **2200** support vectors. The **smod5** model with a gamma of 1, returned just over **6900** support vectors. I assumed that the models with more support vectors are over-fitting, especially if they return lots of errors or if they return no errors but another model with less support vectors is also error free.

```
summary(smod1)
summary(smod2)
summary(smod3)
```

```
summary(smod4)
summary(smod5)
```

Train error tables

The **smod3** model had the lowest number of support vectors (**2141**) and only made 1 error, an error rate of less than **0.1%**. The **smod4** model returned slightly more support vectors (**2283**) with an insignificant error rate increase. While **smod1** and **smod5** did not have any errors, they both produced over three times as many support vectors as **smod3** and **smod4**. The **smod2** model with the cost and gamma values of 1, made **211** errors or had an error rate of **2.1%**. This was not a high rate of error, but the model also had about three times more support vectors than **smod3** and **smod4**.

```
##
##        0    1
##   0 8323    0
##   1    0 1541
##
##        0    1
##   0 8309  197
##   1   14 1344
##
##        0    1
##   0 8323    1
##   1    0 1540
##
##        0    1
##   0 8323    5
##   1    0 1536
##
##        0    1
##   0 8323    0
##   1    0 1541
```

Test set prediction

My assumption that the models with more support vectors were over-fitting does not seem to be correct. The error rates for models one through five were **14.8%**, **13.5%**, **15.9%**, **15.3%**, and **14.6%**. While **smod2** and **smod4** did not have the lowest error rate overall, they both predicted the outcome of a sale more accurately. The **smod5** model had the lowest error rate but still only accurately predicted roughly a fifth of the true sales outcomes. I am still inclined to believe that **smod3** and **smod4** are better models for predicting the test values. **smod1** correctly predicted all of the no sale outcomes, but did not predict any sale outcome accurately. It may be best to choose a model with a gamma value between 0.1 and 1, and a higher cost value in the hundreds or thousands like **smod3** or **smod4**.

```
##
## pred.test1    0    1
##          0 2099  367
##          1    0    0
##
## pred.test2    0    1
##          0 2082  315
```

```
##          1   17   52
##
## pred.test3    0    1
##          0 1884  178
##          1  215  189
##
## pred.test4    0    1
##          0 1897  176
##          1  202  191
##
## pred.test5    0    1
##          0 2030  292
##          1   69   75
```

Trying random forest

The random forest model had more accurate results than the SVM models. The model had an error rate of about **9%**, or an accuracy of about **91%**. It correctly predicted more sales than the radial SVM models.

```
rf.shoppers <- randomForest(y~., data = int_1)
rf.shoppers_pred <- predict(rf.shoppers, newdata = int_2)
table(rf.shoppers_pred, int_2$y)
```

```
##
## rf.shoppers_pred    0    1
##                0 2026  147
##                1   73  220
```

Conclusion and summarized true values

The chunk and output below shows the sale and no-sale outcomes for the **shoppers** data set, the train set, and the test set. Both the random forest and SVM models predicted the no-sale outcomes quite accurately.

It is possible a polynomial or radial SVM model with different parameters could improve the test results. Attempting to use some of the 'tune' functions in the e1071 package would help, however I was unable to get an output while using these functions, and my PC was stuck trying to render them (no errors returned though). This was likely due to my own error. The radial SVM models were alright though, and decent enough make reasonable accurate predictions on the test set. Overall, the random forest model was the best model with a **91.1%** accuracy in predicting the response variable outcome of sale (**Revenue = 1**).

```
sum(shoppers$Revenue == 1) # total sales in data set
## [1] 1908
sum(shoppers$Revenue == 0) # total no-sale outcomes in data set
## [1] 10422
sum(int_1$y == 1) # total sales in train
## [1] 1541
sum(int_1$y == 0) # total no-sale outcomes in train
## [1] 8323
sum(int_2$y == 1) # total sales in test
## [1] 367
sum(int_2$y == 0) # total no-sale outcomes in test
## [1] 2099
```

This is a simple stacked barplot to summarize the actual no sale/sale outcome percentages above, for the main **shoppers** data set, the train set, and the test set.

Figure 1