Practice Machine Learning Course Project

Ujfloss Obianumba

10/17/2019

Introduction

To begin, I loaded the two datasets, and split the larger of the two into a 60%/40% training/testing split. The smaller of the two will be held out entirely and used as a validation set for the very end. Ideally this dataset is larger but for the purposes of this project, that is what will be used.

The question we are trying to address is “Given data on physical activity, can we predict what excerise is the user attempting?”

It appears we are predicting one of five classes, labeled A-E. Our error rate will be defined as “1 - Accuracy,” the % of the predictions which are incorrectly predicted by the model.

In order to navigate through the 160 variables, I first removed the observation number and timestamps from the dataset, along with variables which were missing 95% of their observations or more. It also appears that many of the variables are “near-zero variance,” so these were also removed. This resulted in 54 useful possible predictors in the training dataset.

```
#preprocess

training <- training[ ,colSums(is.na(training)) < nrow(training)*0.95]#remove cols with many NAs

training <- training[,-c(1:5)]#do not predict on name or observations number

testing <- testing[,-c(1:5)]#do not predict on name or observations number

validation <- validation[,-c(1:5,160)]#do not predict on name or observations number

nzv <- nearZeroVar(training, saveMetrics = TRUE)#remove low variance predictors

training <- training[, which(nzv$nzv==FALSE)]
```

Before attempting any further preprocessing technqiues, such as Principal Component Analysis, or creating additional features, I decided to run a simple random forest model on the data to get an initial sense of the predictive power of the data. Random forests are popular models and typically great at multi-class classification.

To implement cross validation, I used the trControl() feature of the train() function, and used k-fold cross validation with three folds. My understanding is that k-fold cross validation is typically a safe bet, and while a larger number of folds could be preferable, using a larger number of folds will be computationally expensive when running locally on a machine with memory limitations.

```
#model (random forest)
modFit_rf <- train(classe~.,
                   method = "rf",
                   trControl = trainControl(method = "cv", number = 3),
                   metric = "Accuracy",
                   data = training)

print(modFit_rf)
```

```
## Random Forest
##
## 11776 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 7851, 7850, 7851
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9908287  0.9883978
##   27    0.9957542  0.9946292
##   53    0.9938860  0.9922662
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

Our in-sample error rate for this simple random forest model appears to be 0.4% (depending on the seed). I then applied the model to the testing dataset, to determine out-of-sample error. We should expect this error rate to be larger than the in-sample error rate, since the model is trained on the training data and theoretically is overfit to “noise” from the training data.

```
#test
pred_rf <- predict(modFit_rf, testing)
confusionMatrix(pred_rf, testing$classe)
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    1    0    0    0
##          B    0 1513    7    0    0
##          C    0    3 1361    1    0
##          D    0    1    0 1285    7
##          E    0    0    0    0 1435
##
## Overall Statistics
##
##                Accuracy : 0.9975
##                  95% CI : (0.9961, 0.9984)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9968
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9967   0.9949   0.9992   0.9951
## Specificity            0.9998   0.9989   0.9994   0.9988   1.0000
## Pos Pred Value         0.9996   0.9954   0.9971   0.9938   1.0000
## Neg Pred Value         1.0000   0.9992   0.9989   0.9998   0.9989
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1928   0.1735   0.1638   0.1829
## Detection Prevalence   0.2846   0.1937   0.1740   0.1648   0.1829
## Balanced Accuracy      0.9999   0.9978   0.9971   0.9990   0.9976
```

After testing the model on the test set, we see an out-of-sample error rate of 0.2%. While this is unexpected, it confirms that this model is not overfit to the training data and is a perfectly fine model to move forward with. Thus, we should expect an out-of-sample error rate of under 1%, although I am unable to be more precise than that.

After trying out a few other techniques of preprocessing and experimenting with different models, I did not see a reason to deviate from the original random forest model. I then used the model to predict the excerises from the 20 observations in the validation set:

```
#apply model to validation set

val_predictions <- rbind(validation$problem_id, as.character(predict(modFit_r
f, validation)))

val_predictions
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]

## [1,] "B"  "A"  "B"  "A"  "A"  "E"  "D"  "B"  "A"  "A"   "B"   "C"   "B"

##      [,14] [,15] [,16] [,17] [,18] [,19] [,20]

## [1,] "A"   "E"   "E"   "A"   "B"   "B"   "B"
```

As you can see, this model accurately predicted 20 out of 20 excersises from the validation set.