Machine Learning : Writeup Project

Sevtap Ozisik

April 22, 2015

Data preparation

Load both datasets.

```
raw_training <- read.csv('pml-training.csv')
raw_testing <- read.csv('pml-testing.csv')
```

Partition training data provided into two sets. One for training and one for cross validation.

```
library(caret)
```

```
## Loading required package: lattice
## Loading required package: ggplot2
```

```
set.seed(1234)
trainingIndex <- createDataPartition(raw_training$classe, list=FALSE, p=.9)
training = raw_training[trainingIndex,]
testing = raw_training[-trainingIndex,]
```

Remove indicators with near zero variance.

```
library(caret)
nzv <- nearZeroVar(training)

training <- training[-nzv]
testing <- testing[-nzv]
raw_testing <- raw_testing[-nzv]
```

Filter columns to only include numeric features and outcome. Integer and other non-numeric features can be trained to reliably predict values in the training file provided, but when used to predict values in the testing set provided, they lead to misclassifications.

```
num_features_idx = which(lapply(training,class) %in% c('numeric')  )
```

We then would like to impute missing values as many exist in our training data.

```
preModel <- preProcess(training[,num_features_idx], method=c('knnImpute'))

ptraining <- cbind(training$classe, predict(preModel, training[,num_features_idx]))
ptesting <- cbind(testing$classe, predict(preModel, testing[,num_features_idx]))
prtesting <- predict(preModel, raw_testing[,num_features_idx])

#Fix Label on classe
names(ptraining)[1] <- 'classe'
names(ptesting)[1] <- 'classe'
```

Model

We can build a random forest model using the numerical variables provided. As we will see later this provides good enough accuracy to predict the twenty test cases. Using [caret][caret], we can obtain the optimal mtry parameter of 32. This is a computationally expensive process, so only the optimized parameter is shown below.

```
library(randomForest)
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

```
rf_model  <- randomForest(classe ~ ., ptraining, ntree=500, mtry=32)
```

Cross Validation

We are able to measure the accuracy using our training set and our cross validation set. With the training set we can detect if our model has bias due to ridgity of our mode. With the cross validation set, we are able to determine if we have variance due to overfitting.

In-sample accuracy

```
training_pred <- predict(rf_model, ptraining)
print(confusionMatrix(training_pred, ptraining$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 5022    0    0    0    0
##          B    0 3418    0    0    0
##          C    0    0 3080    0    0
##          D    0    0    0 2895    0
##          E    0    0    0    0 3247
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9998, 1)
##     No Information Rate : 0.2843
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

The in sample accuracy is 100% which indicates, the model does not suffer from bias.

Out-of-sample accuracy

```
testing_pred <- predict(rf_model, ptesting)
```

Confusion Matrix:

```
print(confusionMatrix(testing_pred, ptesting$classe))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 556   2   0   0   0
##          B   0 373   0   0   2
##          C   1   3 339   1   0
##          D   0   0   3 319   0
##          E   1   1   0   1 358
##
## Overall Statistics
##
##                Accuracy : 0.9923
##                  95% CI : (0.9874, 0.9957)
##     No Information Rate : 0.2847
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9903
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9964   0.9842   0.9912   0.9938   0.9944
## Specificity            0.9986   0.9987   0.9969   0.9982   0.9981
## Pos Pred Value         0.9964   0.9947   0.9855   0.9907   0.9917
## Neg Pred Value         0.9986   0.9962   0.9981   0.9988   0.9987
## Prevalence             0.2847   0.1934   0.1745   0.1638   0.1837
## Detection Rate         0.2837   0.1903   0.1730   0.1628   0.1827
## Detection Prevalence   0.2847   0.1913   0.1755   0.1643   0.1842
## Balanced Accuracy      0.9975   0.9915   0.9941   0.9960   0.9963
```

The cross validation accuracy is greater than 99%, which should be sufficient for predicting the twenty test observations. Based on the lower bound of the confidence interval we would expect to achieve a 98.7% classification accuracy on new data provided.

One caveat exists that the new data must be collected and preprocessed in a manner consistent with the training data.

Test Set Prediction Results

Applying this model to the test data provided yields 100% classification accuracy on the twenty test observations.

```
answers <- predict(rf_model, prtesting)
answers
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  B  A  B  A  A  E  D  B  A  A  B  C  B  A  E  E  A  B  B  B
## Levels: A B C D E
```

Conclusion

We are able to provide very good prediction of weight lifting style as measured with accelerometers.