Practical Machine Learning - Project 1

Nick Allen

Wednesday, June 18, 2014

Load the project data into the workspace. The 'ProjectTemplate' package is used to structure the project.

```
set.seed (825)
library (ProjectTemplate)
load.project( )
```

The training data has been loaded into the environment.

```
dim (pml.training)
```

```
## [1] 19622   160
```

```
dim (pml.testing)
```

```
## [1]  20 160
```

The first 5 fields are not useful predictors. This includes a row identifier, subject name, etc. These predictors would not produce a generalizable model.

```
exclude <- 1:5
pml.training <- pml.training[, -exclude, with = FALSE]
dim (pml.training)
```

```
## [1] 19622   155
```

Many of the predictors have a large number of missing values and so very little variation. These will not be useful for training and are excluded.

```
nzv <- nearZeroVar (pml.training)
pml.training <- pml.training[, -nzv, with = FALSE]
dim (pml.training)
```

```
## [1] 19622   119
```

Many of the fields have large numbers of missing values. Exclude these as they will not be useful.

```
# if a column has more than 5% NAs, exclude it
max.nas <- nrow (pml.training) * 0.05
nas.by.column <- colSums (is.na (pml.training))
too.many.nas <- names (nas.by.column [ nas.by.column > max.nas])

# exclude the columns
keep <- setdiff (names (pml.training), too.many.nas)
pml.training <- pml.training[, keep, with = FALSE]
```

The original training data set is split 80/20 into a training set and a validation data set to use for cross-validation

```
train.index <- createDataPartition (pml.training$classe, p = 0.80, list = FALSE)[, 1]
train    <- pml.training [ train.index]
validate <- pml.training [-train.index]
```

Train a tree-based, Gradiant Boosting Machine (GBM) model to predict the outcome ‘classe’ based on the other 159 predictors. The GBM model performs feature selection automatically. This is important since there are a large number of potential predictors.

```
control <- trainControl (method = "cv", number = 3, repeats = 1)
profile <- train (classe ~ ., data = train, method = "gbm", trControl = control)
```

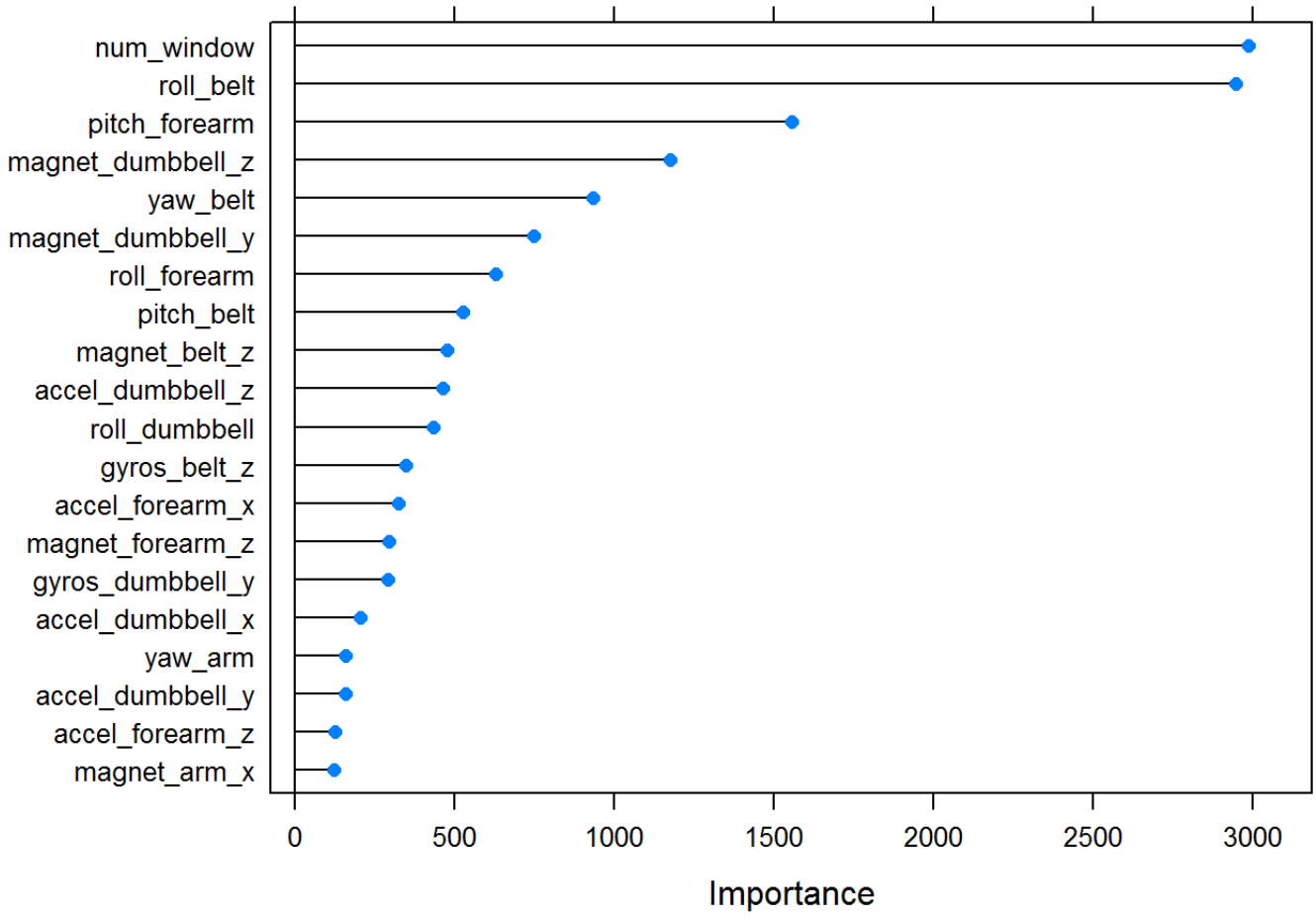
```
## Loading required package: gbm
## Loading required package: survival
## Loading required package: splines
##
## Attaching package: 'survival'
##
## The following object is masked from 'package:caret':
##
##     cluster
##
## Loading required package: parallel
## Loaded gbm 2.1
```

```
profile
```

```
## Stochastic Gradient Boosting
##
## 15699 samples
##    53 predictors
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
##
## Summary of sample sizes: 10466, 10465, 10467
##
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy  Kappa  Accuracy SD  Kappa SD
##   1                  50       0.8       0.7    0.01         0.02
##   1                  100      0.8       0.8    0.008        0.01
##   1                  200      0.9       0.8    0.006        0.007
##   2                  50       0.9       0.9    0.005        0.006
##   2                  100      0.9       0.9    0.006        0.007
##   2                  200      1         1      0.004        0.006
##   3                  50       0.9       0.9    0.007        0.009
##   3                  100      1         1      0.003        0.004
##   3                  200      1         1      0.004        0.005
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
## Accuracy was used to select the optimal model using  the largest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3 and shrinkage = 0.1.
```

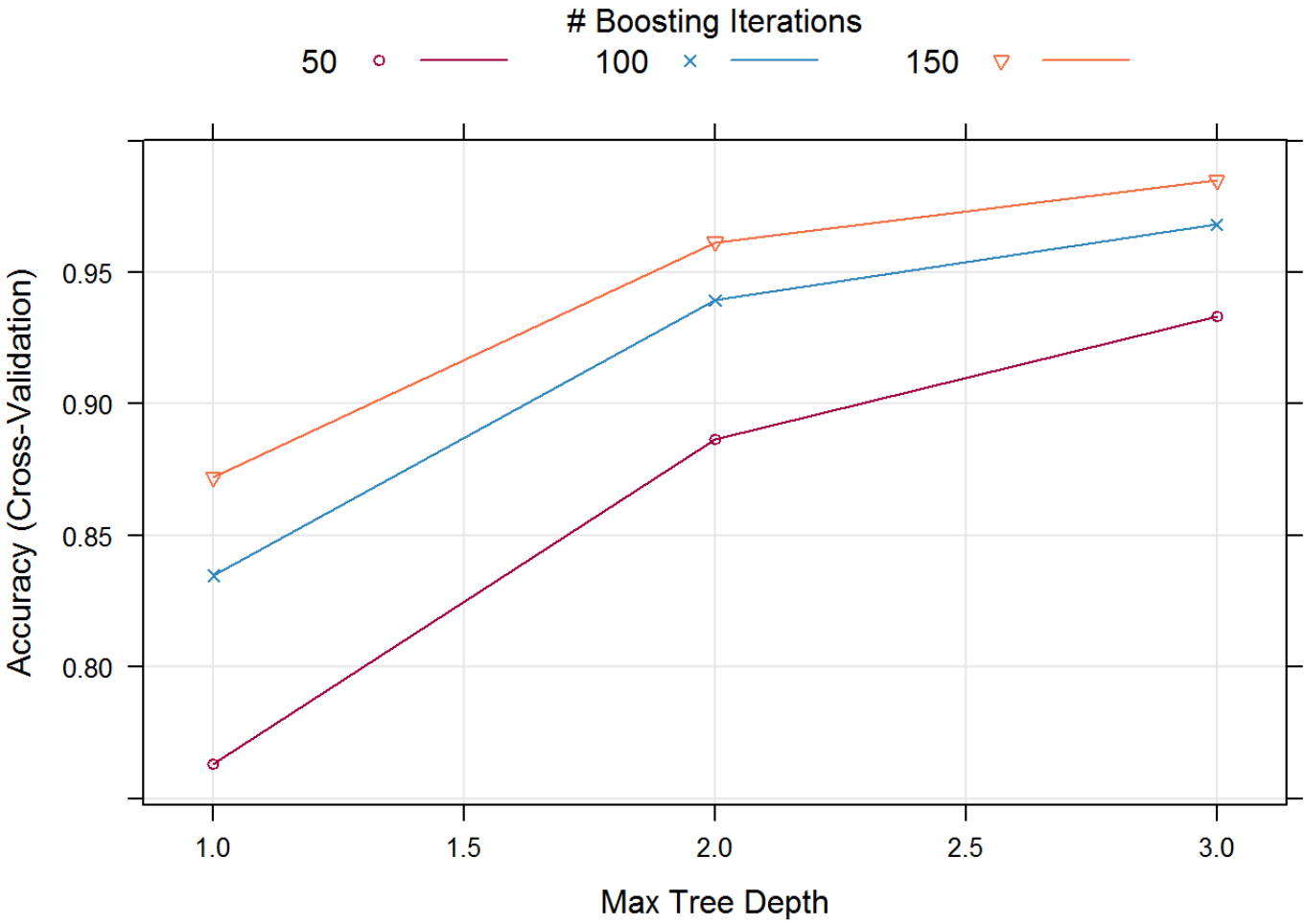
The model found the following predictors to be the most important for predicting the outcome.

```
plot (varImp (profile, scale = FALSE), top = 20)
```



Details which model parameters were most effective and ultimately selected for the final mode.

```
trellis.par.set (caretTheme())
plot (profile, type = c("g", "o"))
```



Using the trained model, predict the 'classe' for the validation set. Assuming that the provided training and test sets have similar characteristics, this will provide a fair measure of the model's accuracy.

```
validate [, classe.hat := predict (profile, newdata = .SD)]
```

A confusion matrix identifies how good the predictions were. The model is achieving a >98% level of accuracy. The results are similar across all reference classes.

```
confusionMatrix(validate$classe.hat, validate$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1116    1    0    0    0
##          B    0  752    5    4    2
##          C    0    6  677   12    0
##          D    0    0    2  626    7
##          E    0    0    0    1  712
##
## Overall Statistics
##
##                Accuracy : 0.99
##                  95% CI : (0.986, 0.993)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.987
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             1.000    0.991    0.990    0.974    0.988
## Specificity             1.000    0.997    0.994    0.997    1.000
## Pos Pred Value          0.999    0.986    0.974    0.986    0.999
## Neg Pred Value          1.000    0.998    0.998    0.995    0.997
## Prevalence              0.284    0.193    0.174    0.164    0.184
## Detection Rate          0.284    0.192    0.173    0.160    0.181
## Detection Prevalence    0.285    0.194    0.177    0.162    0.182
## Balanced Accuracy       1.000    0.994    0.992    0.985    0.994
```

Using the trained model, make predictions on the test data that needs to be submitted as part of this project.

```
pml.testing [, classe.hat := predict (profile, newdata = .SD)]
answers <- pml.testing [["classe.hat"]]
```

Output the results so they can be submitted.

```
#
# write the solutions for the test set to separate files for submission
#
pml_write_files = function (x, directory = "solutions") {
  dir.create (directory)

  n = length(x)
  for(i in 1:n){
    filename = paste0 ("problem_id_",i,".txt")
    filename = file.path (directory, filename)
    write.table (x[i], file=filename, quote=FALSE, row.names=FALSE, col.names=FALSE)
  }
}

pml_write_files (answers)
```

```
## Warning: 'solutions' already exists
```