Practical ML project

MJ Kamal

February 11, 2018

```
library(caret)
```

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

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

```
# Read the training data - will read the testing later below
training <- read.csv("project_training.csv")
```

Feature engineering

```
# Build filtered list of features - excluding obvious ones & ones with large number of missing v
alues (from eye balling)
features <- c("roll_belt", "pitch_belt", "yaw_belt", "total_accel_belt", "gyros_belt_x", "gyros_
belt_y", "gyros_belt_z","accel_belt_x","accel_belt_y","accel_belt_z","magnet_belt_x","magnet_bel
t_y","magnet_belt_z","roll_arm", "pitch_arm","yaw_arm","total_accel_arm","gyros_arm_x","gyros_ar
m_y","gyros_arm_z","accel_arm_x",
"accel_arm_y","accel_arm_z","magnet_arm_x","magnet_arm_y","magnet_arm_z","roll_dumbbell","pitch_
dumbbell","yaw_dumbbell","gyros_dumbbell_x","gyros_dumbbell_y","gyros_dumbbell_z","accel_dumbbel
l_x","accel_dumbbell_y",
 "accel_dumbbell_z","magnet_dumbbell_x","magnet_dumbbell_y","magnet_dumbbell_z","roll_forearm",
"pitch_forearm","yaw_forearm","total_accel_forearm","gyros_forearm_x","gyros_forearm_y","gyros_f
orearm_z","accel_forearm_x","accel_forearm_y","accel_forearm_z", "magnet_forearm_x","magnet_fore
arm_y","magnet_forearm_z","classe")

# Extract out the filtered features from training data
training = subset(training,select = features)

# Remove leftover observations with missing values in any of the features
training = training[complete.cases(training), ]

# In addition remove low variance features
nzv_cols <- nearZeroVar(training)
if(length(nzv_cols) > 0) training <- training[, -nzv_cols]

# Explicitly Convert the target column to factors to be safe
training$classe <- as.factor(training$classe)

# Now take out 10% of the training data as test data to check quality of model
inTrain = createDataPartition(training$classe, p = 9/10)[[1]]
training_subset = training[ inTrain,]
testing = training[-inTrain,]
```

Build and train the model

```
# Setting up cross-validation with 10-fold
# Each cross-validation is repeated 3 times (randomly partioned each time)
# The model tuning parameters grid is to be set randomly
fitControl <- trainControl( method = "repeatedcv",  number = 10, repeats = 3, search = "random")

# Train with Random Forest of 50 trees max using the above cross-validation spec
rndfit <-train(classe ~ ., data = training_subset,method='rf',ntree=50, trControl=fitControl)
```

```
## Loading required package: randomForest
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

Get stats on the fitted model

```
print(rndfit)
```

```
## Random Forest
##
## 17662 samples
##    51 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 15896, 15894, 15895, 15895, 15895, 15896, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    7    0.9954138  0.9941987
##   32    0.9926019  0.9906414
##   40    0.9918090  0.9896385
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 7.
```

```
# Predict on the testing portion extracted out above
pred <- predict(rndfit, newdata = testing)

# Get the confusion matrix for above prediction
confusionMatrix(pred,testing$classe)$table
```

```
##           Reference
## Prediction   A   B   C   D   E
##          A 558   2   0   0   0
##          B   0 376   2   0   0
##          C   0   1 339   1   1
##          D   0   0   1 320   1
##          E   0   0   0   0 358
```

Predict on the given test data

```
# Read the test data
testing <- read.csv("project_test.csv")
pred <- predict(rndfit, newdata = testing)
```