PML Week 4 assignment

Karan Sandam

23/08/2020

Summary

The aim of this project is to predict the 'classe' variable in the dataset and test the model on provided test set. The preprocessing stage involves removing columns with NA and having near zero variance between them. The provided training set is split into two parts viz. train1 and train2 for traning and testing respectively. For final prediction provided test dataset is used.
Two models have been fitted 1.SVM 2.Random Forest(ranger). Out of these random forest gives better accuracy on test data of training set and hence will be used for predictions in final test set.

Setting up the Environment

```r
library(caret)
packageVersion("caret")
```

```
## [1] '6.0.86'
```

```r
library(e1071)
packageVersion("e1071")
```

```
## [1] '1.7.3'
```

```r
library(dplyr)
packageVersion("dplyr")
```

```
## [1] '1.0.1'
```

```r
## For taking advantage of multicore CPU
library(doParallel)
packageVersion("doParallel")
```

```
## [1] '1.0.15'
```

Setting up multicore environment

By default R uses single core for a rsession. In order to take advantage of remaining cores we will use the doParallel package and allocate 50%(no. of cores/2) of cpu to a rsession.

```r
## Only physical cores
cores = detectCores(logical = FALSE)
cr <- makePSOCKcluster(as.integer(cores/2))
registerDoParallel(cr)
```

Now we can utilize 50% of our cpu to train the models.

Loading the data

```r
training <- read.csv("pml-training.csv")
dim(training)
```

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

```r
testing <- read.csv("pml-testing.csv")
dim(testing)
```

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

Data Preprocessing

We will preprocess the training data and then extract only those columns from testing data which are left after preprocessing.

```r
col_names <- names(training)
na_cols <- ""           ##columns which are empty
for(n in col_names){
        if(sum(is.na(training[,n]))>10){
                na_cols<-append(na_cols,n)
        }
}
head(na_cols)
```

```
## [1] ""                     "max_roll_belt"       "max_picth_belt"
## [4] "min_roll_belt"       "min_pitch_belt"      "amplitude_roll_belt"
```

```r
## removing 1st element since it was dummy
na_cols <- na_cols[2:length(na_cols)]
head(na_cols)
```

```
## [1] "max_roll_belt"        "max_picth_belt"       "min_roll_belt"
## [4] "min_pitch_belt"       "amplitude_roll_belt"  "amplitude_pitch_belt"
```

Selecting only those columns from training set which are not there in na_cols(columns in na_cols have NA's hence need to be excluded)

```r
training <- select(training,!all_of(na_cols))
dim(training)
```

```
## [1] 19622    93
```

Now lets take a look at our training set

```r
names(training)
```

```
##  [1] "X"                       "user_name"
##  [3] "raw_timestamp_part_1"    "raw_timestamp_part_2"
##  [5] "cvtd_timestamp"          "new_window"
##  [7] "num_window"              "roll_belt"
##  [9] "pitch_belt"              "yaw_belt"
## [11] "total_accel_belt"        "kurtosis_roll_belt"
## [13] "kurtosis_picth_belt"     "kurtosis_yaw_belt"
## [15] "skewness_roll_belt"      "skewness_roll_belt.1"
## [17] "skewness_yaw_belt"       "max_yaw_belt"
## [19] "min_yaw_belt"            "amplitude_yaw_belt"
```

```
## [21] "gyros_belt_x"           "gyros_belt_y"
## [23] "gyros_belt_z"           "accel_belt_x"
## [25] "accel_belt_y"           "accel_belt_z"
## [27] "magnet_belt_x"          "magnet_belt_y"
## [29] "magnet_belt_z"          "roll_arm"
## [31] "pitch_arm"              "yaw_arm"
## [33] "total_accel_arm"        "gyros_arm_x"
## [35] "gyros_arm_y"            "gyros_arm_z"
## [37] "accel_arm_x"            "accel_arm_y"
## [39] "accel_arm_z"            "magnet_arm_x"
## [41] "magnet_arm_y"           "magnet_arm_z"
## [43] "kurtosis_roll_arm"      "kurtosis_picth_arm"
## [45] "kurtosis_yaw_arm"       "skewness_roll_arm"
## [47] "skewness_pitch_arm"     "skewness_yaw_arm"
## [49] "roll_dumbbell"          "pitch_dumbbell"
## [51] "yaw_dumbbell"           "kurtosis_roll_dumbbell"
## [53] "kurtosis_picth_dumbbell" "kurtosis_yaw_dumbbell"
## [55] "skewness_roll_dumbbell"  "skewness_pitch_dumbbell"
## [57] "skewness_yaw_dumbbell"   "max_yaw_dumbbell"
## [59] "min_yaw_dumbbell"       "amplitude_yaw_dumbbell"
## [61] "total_accel_dumbbell"    "gyros_dumbbell_x"
## [63] "gyros_dumbbell_y"       "gyros_dumbbell_z"
## [65] "accel_dumbbell_x"       "accel_dumbbell_y"
## [67] "accel_dumbbell_z"       "magnet_dumbbell_x"
## [69] "magnet_dumbbell_y"      "magnet_dumbbell_z"
## [71] "roll_forearm"           "pitch_forearm"
## [73] "yaw_forearm"            "kurtosis_roll_forearm"
## [75] "kurtosis_picth_forearm"  "kurtosis_yaw_forearm"
## [77] "skewness_roll_forearm"   "skewness_pitch_forearm"
## [79] "skewness_yaw_forearm"    "max_yaw_forearm"
## [81] "min_yaw_forearm"        "amplitude_yaw_forearm"
## [83] "total_accel_forearm"     "gyros_forearm_x"
## [85] "gyros_forearm_y"        "gyros_forearm_z"
## [87] "accel_forearm_x"        "accel_forearm_y"
## [89] "accel_forearm_z"        "magnet_forearm_x"
## [91] "magnet_forearm_y"       "magnet_forearm_z"
## [93] "classe"
```

```r
head(training[1:10,1:10])
```

```
##   X user_name raw_timestamp_part_1 raw_timestamp_part_2   cvtd_timestamp
## 1 1  carlitos           1323084231               788290 05/12/2011 11:23
## 2 2  carlitos           1323084231               808298 05/12/2011 11:23
## 3 3  carlitos           1323084231               820366 05/12/2011 11:23
## 4 4  carlitos           1323084232               120339 05/12/2011 11:23
## 5 5  carlitos           1323084232               196328 05/12/2011 11:23
## 6 6  carlitos           1323084232               304277 05/12/2011 11:23
##   new_window num_window roll_belt pitch_belt yaw_belt
## 1         no         11      1.41       8.07    -94.4
## 2         no         11      1.41       8.07    -94.4
## 3         no         11      1.42       8.07    -94.4
## 4         no         12      1.48       8.05    -94.4
## 5         no         12      1.48       8.07    -94.4
## 6         no         12      1.45       8.06    -94.4
```

It can be seen that initial 7 columns are not needed for building the model. Hence dropping them

```
training <- training[,8:93]
dim(training)
```

```
## [1] 19622    86
```

So major part of pre processing is done. Now lets remove columns with near zero variance.

```
nzv <- nearZeroVar(training)
training <- training[,-nzv]
dim(training)
```

```
## [1] 19622    53
```

Now we are left with 52 variables(excl. output) to feed into our model.

Keeping only those columns in test set which are present in training

```
col_names <- names(training)
##removing the classe because its the output in test set which is to be predicted
col_names <- col_names[1:length(col_names)-1]
testing <- select(testing,all_of(col_names))
dim(testing)
```

```
## [1] 20 52
```

The dimensions of our training and testing set are now matching.(excluding output variable)

Partitioning the dataset

```
set.seed(45)
intrain <- createDataPartition(training$classe, p=0.7, list=FALSE)
train1 <- training[intrain,]
train2 <- training[-intrain,]
```

Building the SVM

```
set.seed(46)
library(e1071)
model_svm <- svm(classe ~.,data = train1, type="C")
```

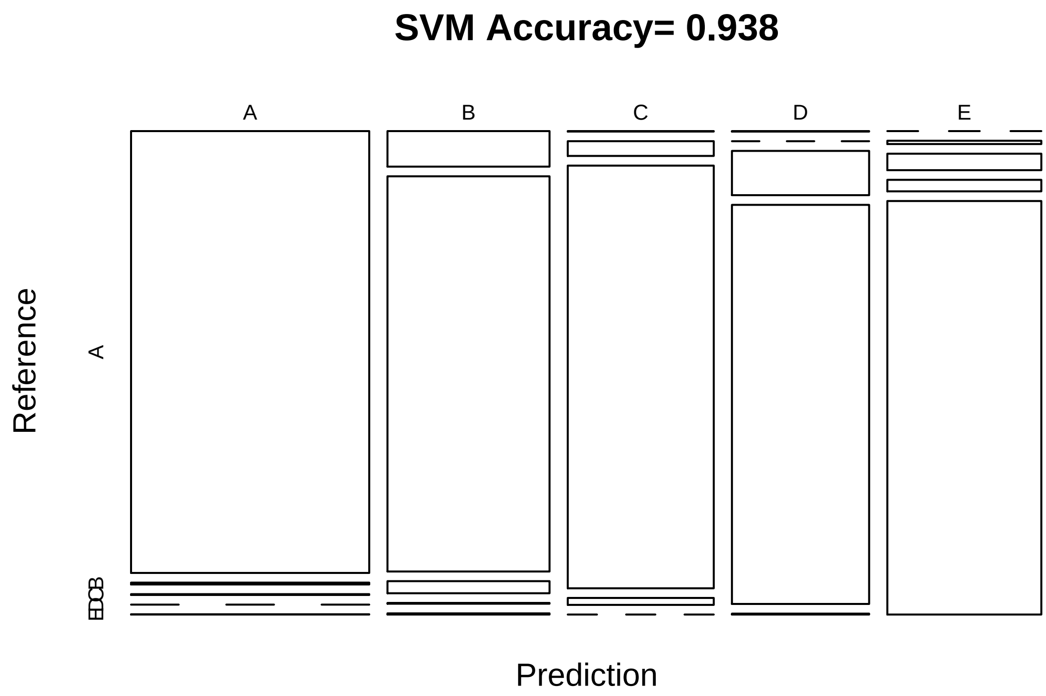
Prediction by SVM

```
library(e1071)          ##predict function from this package
pred_svm <- predict(model_svm, train2)
conf_svm<-confusionMatrix(train2$classe, pred_svm)
conf_svm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1663    7    3    0    1
##          B   91 1012   31    2    3
```

```
##          C    1   34  975   16    0
##          D    1    0   96  865    2
##          E    0    8   40   28 1006
##
## Overall Statistics
##
##                Accuracy : 0.9381
##                  95% CI : (0.9317, 0.9442)
##     No Information Rate : 0.2984
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9216
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9470   0.9538   0.8515   0.9495   0.9941
## Specificity            0.9973   0.9737   0.9892   0.9801   0.9844
## Pos Pred Value         0.9934   0.8885   0.9503   0.8973   0.9298
## Neg Pred Value         0.9779   0.9897   0.9650   0.9907   0.9988
## Prevalence             0.2984   0.1803   0.1946   0.1548   0.1720
## Detection Rate         0.2826   0.1720   0.1657   0.1470   0.1709
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9722   0.9637   0.9204   0.9648   0.9892
```

```r
plot(conf_svm$table, col=conf_svm$byClass,
     main = paste ("SVM Accuracy=", round (conf_svm$overall['Accuracy'],3)))
```



Model Accuracy:0.938

Building a RandomForest(ranger)

Ranger is a fast implementation of random forests (Breiman 2001) or recursive partitioning, particularly suited for high dimensional data.

```
fitControl <- trainControl(
        method = "oob", ##for 'oob(out of bag) score for random forest'
        number = 3,)
```

Lets fit the model now with train function

```
set.seed(47)
model_rf <- train(as.factor(classe) ~ ., data = train1,
                  method = "ranger",
                  trControl = fitControl,
                  verbose = TRUE)
model_rf
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
```

```
## Resampling results across tuning parameters:
##
##   mtry  splitrule   Accuracy   Kappa
##    2    gini        0.9922108  0.9901465
##    2    extratrees  0.9913373  0.9890416
##   27    gini        0.9921380  0.9900549
##   27    extratrees  0.9948315  0.9934623
##   52    gini        0.9862415  0.9825948
##   52    extratrees  0.9951227  0.9938306
##
## Tuning parameter 'min.node.size' was held constant at a value of 1
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were mtry = 52, splitrule = extratrees
##  and min.node.size = 1.
```

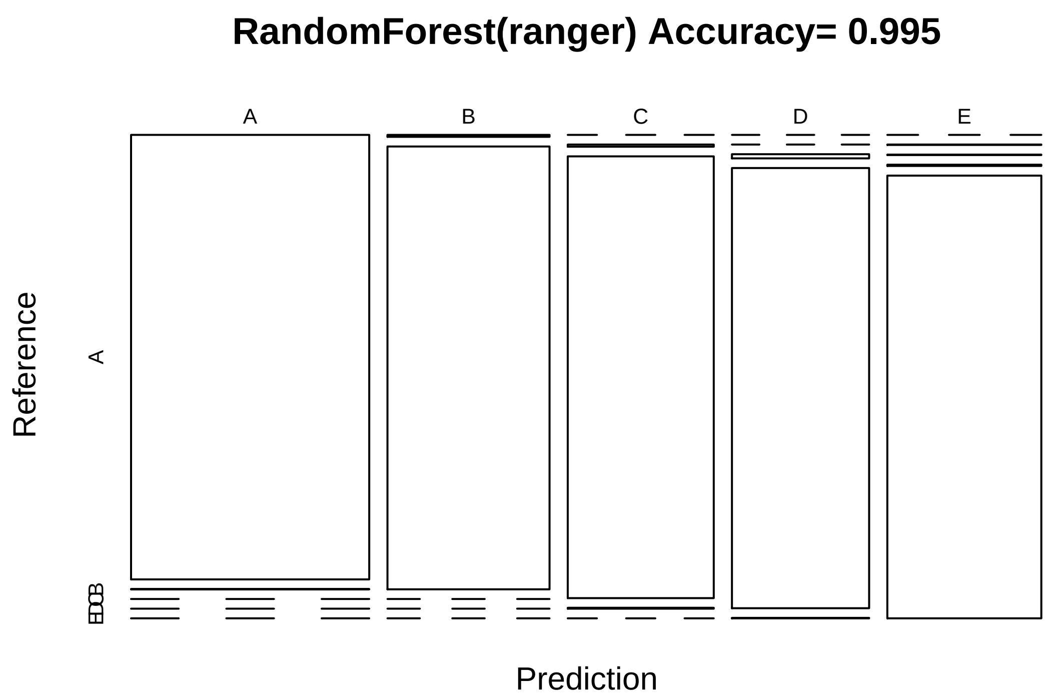
Prediction by RandomForest(ranger)

```r
library(caret)          ##predict function from this package
pred_rf <- predict(model_rf,train2)
conf_rf<-confusionMatrix(train2$classe, pred_rf)
conf_rf
```

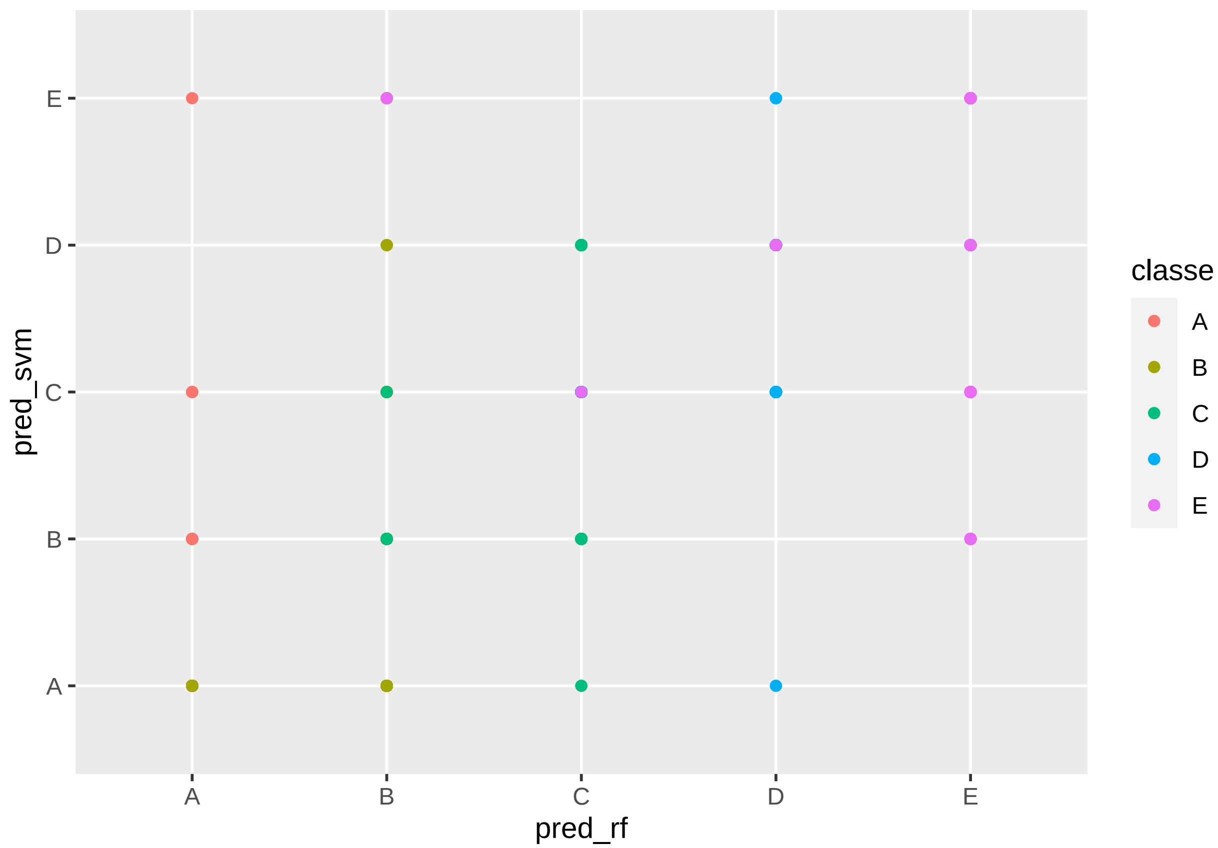
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    1    0    0    0
##          B    5 1134    0    0    0
##          C    0    5 1019    2    0
##          D    0    0    9  954    1
##          E    0    1    1    3 1077
##
## Overall Statistics
##
##                Accuracy : 0.9952
##                  95% CI : (0.9931, 0.9968)
##     No Information Rate : 0.2851
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.994
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9970   0.9939   0.9903   0.9948   0.9991
## Specificity            0.9998   0.9989   0.9986   0.9980   0.9990
## Pos Pred Value         0.9994   0.9956   0.9932   0.9896   0.9954
## Neg Pred Value         0.9988   0.9985   0.9979   0.9990   0.9998
## Prevalence             0.2851   0.1939   0.1749   0.1630   0.1832
## Detection Rate         0.2843   0.1927   0.1732   0.1621   0.1830
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
```

```
## Balanced Accuracy      0.9984   0.9964   0.9944   0.9964   0.9990
```

```
plot(conf_rf$table, col=conf_rf$byClass,
     main = paste ("RandomForest(ranger) Accuracy=", round (conf_rf$overall['Accuracy'],3)))
```




```
library(ggplot2)
qplot(pred_rf,pred_svm,colour=classe, data=train2)
```



The differences in predictions can be clearly seen.

Model Accuracy:0.995

Out of both the models RandomForest(ranger) has greater out of sample accuracy. Hence will be used for final predictions.

Final Prediction

Test data is already preprocessed, so we can directly predict it.

```
library(caret)
fin_pred <- predict(model_rf, testing)
fin_pred
```

```
##  [1] 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
```

```
library(doParallel)
```

```
## Loading required package: foreach

## Loading required package: iterators

## Loading required package: parallel
```

```
##stopping the multi-core environment
stopImplicitCluster()
```

References

Caret Package
Caret Tutorial