Practical Machine Learning: Prediction Assignment Writeup

Serra Gengec

22 12 2021

Background

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, the goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways.

Uploading the Dataset

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

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

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

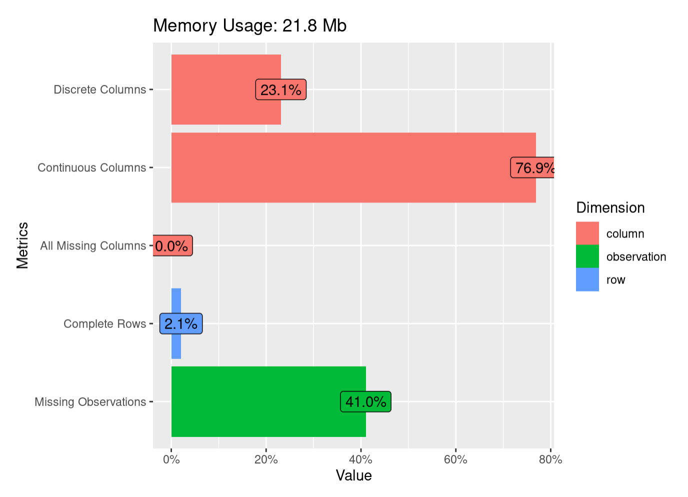
```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
## Loaded gbm 2.1.8
```

```
traindata <- read.csv("pml-training.csv", na.strings=c("NA"))
validation <- read.csv("pml-testing.csv", na.strings=c("NA","#DIV/0!"))
```

Expolatory Analysis

The DataExplorer library is used to summarize the classes in the data as well as the missing values. The plot below shows that nearly 94% of all variables are continuous and more than half of the observations are missing. The summary also shows that the data do not include complete rows, so it is necessary to remove the variables with no observations.



Preprocessing

In the pre-processing step first the features with near-zero variances and the irrelevant variables are filtered. In addition to this filtering, the rows with NA observations are also removed. The same variables will be removed from the validation set as well as the problem_id column in this set.

```
filterData <- traindata[, 7:160]
filterValidation <- validation[, 7:160]
nzvt <- nearZeroVar(filterData)
filterData <- filterData[, -nzvt]
filterValidation <- filterValidation[, -nzvt]

nzvtv <- nearZeroVar(filterValidation)
filterData <- filterData[, -nzvtv]
filterValidation <- filterValidation[, -nzvtv]

completedata<- na.omit(filterData)
dim(completedata)
```

```
## [1] 19622    54
```

The next step is the analysis of correlated variables and limiting the correlation between variables to maximum 75%.

```
descrCor <- cor(completedata[, 1:(dim(completedata)[2]-1)])
summary(descrCor[upper.tri(descrCor)])
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.99201 -0.10713  0.00214  0.00217  0.09192  0.98092
```

```
highlyCorDescr <- findCorrelation(descrCor, cutoff = .75)
filteredNewData <- completedata[,-highlyCorDescr]
descrCor2 <- cor(filteredNewData[, 1:(dim(filteredNewData)[2]-1)])
summary(descrCor2[upper.tri(descrCor2)])
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -0.606983 -0.099503  0.007402  0.004357  0.087829  0.736546
```

```
dim(filteredNewData)
```

```
## [1] 19622    33
```

```
newValidation <- filterValidation[,-highlyCorDescr]
newValidation <- newValidation[, 1:(dim(newValidation)[2]-1)]
```

In the end the classe variable in training data is distributed as below. While moving forward the classe variable will be a factor variable with 5 levels.

```
filteredNewData$classe <- as.factor(filteredNewData$classe)
summary(filteredNewData$classe)
```

```
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

```
percentage <- prop.table(table(completedata$classe)) * 100
cbind(freq=table(completedata$classe), percentage=percentage)
```

```
##   freq percentage
## A 5580   28.43747
## B 3797   19.35073
## C 3422   17.43961
## D 3216   16.38977
## E 3607   18.38243
```

In the last step of pre-processing, the final data set prepared will be parted to training and test sets.

```
set.seed(7)
inTrain = createDataPartition(filteredNewData$class, p = 3/4, list = FALSE)
training = filteredNewData[ inTrain,]
testing = filteredNewData[-inTrain,]
```

Building the Models

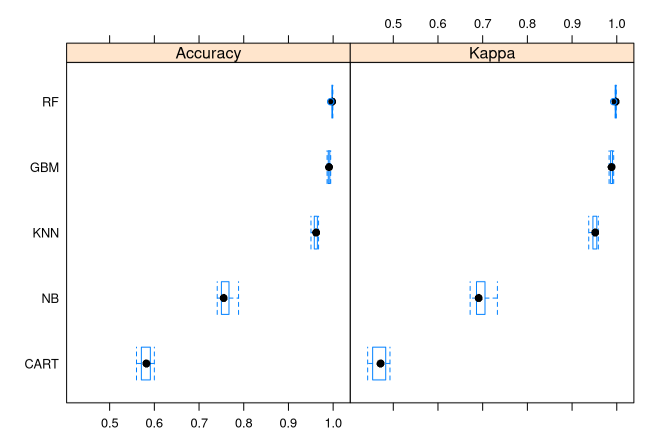
For the model building, 5 methods will be applied and compared, the best model will be selected to further analysis. The training will be done with cross validation for 10 folds.

Fitting the models and comparing with resamples()

```
control <- trainControl(method="cv", number=10, classProbs =  TRUE)

# train the K-Nearest Neighbors model
set.seed(7)
knnFit <- train(classe~., data=training, method="knn", trControl=control, preProc = c("center", "scale"))

# train the Naive Bayes model
set.seed(7)
nbFit <- train(classe~., data=training, method="naive_bayes", trControl=control)

# train the Classification Tree model
set.seed(7)
cartFit <- train(classe~., data=training, method="rpart", trControl=control)

# train the Stochastic Gradient Boosting model
set.seed(7)
gbmFit <- train(classe~., data=training, method="gbm", trControl=control, verbose = FALSE)

# train the Random Forest model
set.seed(7)
rfFit <- train(classe~., data=training, method="rf", trControl=control, importance = TRUE)

# collect resamples
results <- resamples(list(KNN=knnFit, NB=nbFit, CART=cartFit, GBM = gbmFit, RF = rfFit))
# summarize the distributions
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: KNN, NB, CART, GBM, RF
## Number of resamples: 10
##
## Accuracy
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## KNN  0.9504076 0.9585168 0.9619311 0.9607275 0.9643522 0.9674134    0
## NB   0.7406653 0.7509354 0.7551779 0.7596187 0.7656250 0.7884354    0
## CART 0.5600815 0.5712105 0.5822556 0.5809259 0.5908628 0.6000000    0
## GBM  0.9864130 0.9894669 0.9908288 0.9905558 0.9921814 0.9945652    0
## RF   0.9945652 0.9972826 0.9979620 0.9974187 0.9984704 0.9993197    0
##
## Kappa
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## KNN  0.9372770 0.9475136 0.9518177 0.9503141 0.9549229 0.9587779    0
## NB   0.6719501 0.6867722 0.6911201 0.6963835 0.7035052 0.7329229    0
## CART 0.4427110 0.4543595 0.4715214 0.4688971 0.4825846 0.4926204    0
## GBM  0.9828078 0.9866789 0.9884009 0.9880537 0.9901094 0.9931252    0
## RF   0.9931262 0.9965627 0.9974222 0.9967349 0.9980653 0.9991396    0
```

```
# boxplots of results
bwplot(results)
```



As it can be seen in the plots, GBM and Random Forest models performed better on the training set. These two models will be compared according to the test set prediction performances.

Predictions on the Test Set

```
gbm_pred <- predict(gbmFit, newdata=testing)
# Check model performance
confusionMatrix(gbm_pred,testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1392    5    0    0    1
##          B    3  940   11    1    1
##          C    0    4  839    8    3
##          D    0    0    5  794    6
##          E    0    0    0    1  890
##
## Overall Statistics
##
##                Accuracy : 0.99
##                  95% CI : (0.9868, 0.9926)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9874
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9978   0.9905   0.9813   0.9876   0.9878
## Specificity            0.9983   0.9960   0.9963   0.9973   0.9998
## Pos Pred Value         0.9957   0.9833   0.9824   0.9863   0.9989
## Neg Pred Value         0.9991   0.9977   0.9960   0.9976   0.9973
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2838   0.1917   0.1711   0.1619   0.1815
## Detection Prevalence   0.2851   0.1949   0.1741   0.1642   0.1817
## Balanced Accuracy      0.9981   0.9932   0.9888   0.9924   0.9938
```

```
gbm_prob <- predict(gbmFit, newdata=testing, type = "prob")
gbm_auc <- multiclass.roc(testing$classe, gbm_prob)
print(gbm_auc$auc)
```

```
## Multi-class area under the curve: 0.9998
```

```
rf_pred <- predict(rfFit, newdata=testing)
# Check model performance
confusionMatrix(rf_pred,testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1394    1    0    0    0
##          B    1  944    4    0    0
##          C    0    3  850    2    0
##          D    0    0    1  802    0
##          E    0    1    0    0  901
##
## Overall Statistics
##
##                Accuracy : 0.9973
##                  95% CI : (0.9955, 0.9986)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9966
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9993   0.9947   0.9942   0.9975   1.0000
## Specificity            0.9997   0.9987   0.9988   0.9998   0.9998
## Pos Pred Value         0.9993   0.9947   0.9942   0.9988   0.9989
## Neg Pred Value         0.9997   0.9987   0.9988   0.9995   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2843   0.1925   0.1733   0.1635   0.1837
## Detection Prevalence   0.2845   0.1935   0.1743   0.1637   0.1839
## Balanced Accuracy      0.9995   0.9967   0.9965   0.9986   0.9999
```

```
rf_prob <- predict(rfFit, newdata=testing, type = "prob")
rf_auc <- multiclass.roc(testing$classe, rf_prob)
print(rf_auc$auc)
```

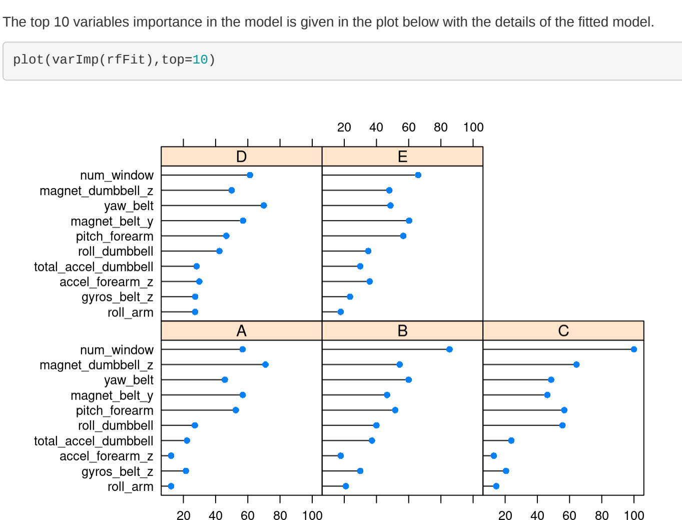
```
## Multi-class area under the curve: 1
```

When the two models are compared the Random Forest model perfroms better both in accuracy and AUC metrics. Random Forest model will be used moving forward.

Details of the RF model

The top 10 variables importance in the model is given in the plot below with the details of the fitted model.

```
plot(varImp(rfFit),top=10)
```




```
print(rfFit)
```

```
## Random Forest
##
## 14718 samples
##    32 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 13246, 13246, 13245, 13248, 13246, 13246, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9951086  0.9938128
##   17    0.9974187  0.9967349
##   32    0.9931385  0.9913214
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 17.
```

Predictions for the Validation Set

The predictions are made for the validation set with the selected GBM model.

```
finalPred <- predict(rfFit, newdata=newValidation)
print(finalPred)
```

```
##  [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
```