ML_Project

Nima Taghidoost

3/21/2020

Summary

In this project, first of all we split the data into train and test set; Then we standardized the variables and removerd the columns which were mostly NAs and also the columns which mostly had the same value in each case; Then we found the high correlated variables and removed them to have a clean data frame with enough columns. We did the PCA process and got help from 3 algorithms: Random Forrest,SVM and Boosting. Then we combined these models to have a better prediction model.

Libraries

```
library(caret)
library(data.table)
library(dplyr)
library(e1071)
library(corrplot)
library(Hmisc)
```

Read Data

```
set.seed(33833)
pml <-read.csv("C:/Users/Nima/Desktop/ML_Project/pml-training.csv")
validation <-read.csv("C:/Users/Nima/Desktop/ML_Project/pml-testing.csv")
```

Train & Test sets

```
inTrain <- createDataPartition(pml$classe,p=0.7,list=FALSE)

training <- pml[inTrain,]

testing <- pml[-inTrain,]
```

Preprocess

Remove columns which are mostly NAs We removed the columns that have more than 90 percent NA values.

```
NAPercent <- function(Col) {

                              Percent <- mean(is.na(Col)) >0.9
}

naVars <- data.frame(sapply(training, NAPercent) )
naVars <- setDT(naVars, keep.rownames = "Var")[]
naVars <- as.list(naVars[sapply.training..NAPercent.==1,1])

training <- select(training,-naVars[["Var"]])
testing <- select(testing,-naVars[["Var"]])
validation <- select(validation,-naVars[["Var"]])
```

scale,center and impute nulls We used the KNN method to replace the nulls.

```
preObj <- preProcess(training,method= c("scale","center","knnImpute"))

training <- predict(preObj,training)
testing <- predict(preObj,testing)
validation <- predict(preObj,validation)
```

Finding near zero variables We removerd the features that had mostly the same data.

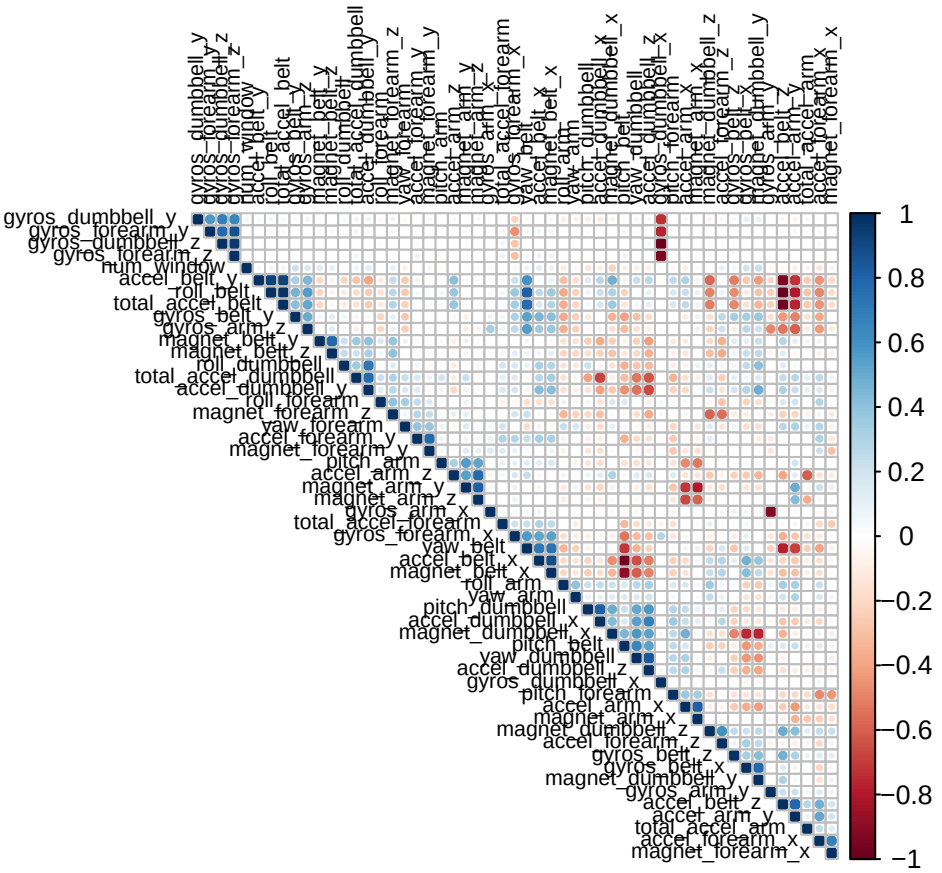
```
NZV <- nearZeroVar(training,saveMetrics = TRUE)
NZV <- setDT(NZV, keep.rownames = "Var")[]
NearZeros <- NZV[NZV$nzv=="TRUE",Var]


training <- select(training,-all_of(NearZeros))
testing <- select(testing,-all_of(NearZeros))
validation <- select(validation,-all_of(NearZeros))
```

Remove the first columns Colmns 1 to 5 do not have any impact on the result so we ommit them.

```
training <- training[,-c(1:5)]
testing <- testing[,-c(1:5)]
validation <- validation[,-c(1:5)]
```

Remove High Correlated Features

```
CorTraining <- as.matrix(select(training,-classe))
M <- rcorr(CorTraining)
corrplot(M$r,method="circle",type = "upper",order="hclust",p.mat=M$P,
         sig.level = 0.01,tl.cex=0.7,tl.col = "black",insig = "blank")
```




```
SigCorIndex <- findCorrelation(M$r,cutoff = 0.85)

SigCorNames <- names(training[,c(SigCorIndex)])

SigCorNames
```

```
## [1] "accel_belt_z"     "roll_belt"        "accel_belt_y"     "accel_belt_x"
## [5] "pitch_belt"       "gyros_forearm_y"  "gyros_dumbbell_x" "gyros_dumbbell_z"
## [9] "gyros_arm_x"
```

```
training <- select(training,-all_of(SigCorNames))
testing <- select(testing,-all_of(SigCorNames))
validation <- select(validation,-all_of(SigCorNames))
```

PCA

Here we use the PCA method with 95 percent treshold.

```
PCApreProc <- preProcess(training,method="pca",thresh=0.95)
trainPC <- predict(PCApreProc,training)
testPC <- predict(PCApreProc,testing)
validationPC <- predict(PCApreProc,validation)
```

Random Forrest

We used "Random Forrest" with 50 trees.

```
Model_rf_all <- train(classe ~ ., data=training, method="rf",ntree=50)
predrf <-  predict(Model_rf_all,testing)
confusionMatrix(predrf , testing$classe)$overall[1]
```

```
## Accuracy
## 0.997791
```

Important Variables

We can get the important variables from "Random Forrest".

```
imp <- varImp(Model_rf_all)
imp <- data.frame( imp[["importance"]] )
imp <- data.frame(overall = imp$Overall,
                  names   = rownames(imp))
imp <- imp[order(imp$overall,decreasing = T),]
top_n(imp,10,imp$overall)
```

```
##      overall             names
## 1  100.00000        num_window
## 2   40.52754          yaw_belt
## 3   37.50533     pitch_forearm
## 4   31.90527 magnet_dumbbell_z
## 5   26.05472 magnet_dumbbell_y
## 6   18.94861     magnet_belt_y
## 7   17.09374      roll_forearm
## 8   11.97802  accel_dumbbell_y
## 9   10.87031     roll_dumbbell
## 10  10.70962 magnet_dumbbell_x
```

SVM

We used the SVM method to predict.

```
Model_svm_pca <- svm(classe~.,verbose=FALSE,data=trainPC)

predsvmPCA <- predict(Model_svm_pca,testPC)
confusionMatrix(predsvmPCA,testing$classe)$overall[1]
```

```
##  Accuracy
## 0.9338997
```

Boosting

Here we used Boosting method and cross validation with 5 folds.

```
trControl <- trainControl(method="cv", number=5)
Model_gbm <- train(classe~.,method="gbm",data=trainPC,verbose=FALSE,trControl=trControl)


predgbmPCA <- predict(Model_gbm,testPC)
confusionMatrix(predgbmPCA,testPC$classe)$overall[1]
```

```
##  Accuracy
## 0.8242991
```

Combining the models

We combined the methods with random forrest method.

```
CombModelDF <- data.frame(predgbmPCA,predsvmPCA,predrf,classe =testing$classe)

Model_comb <- train(classe~. , method="rf",data=CombModelDF,ntree=50)

predComb <- predict(Model_comb,CombModelDF)
confusionMatrix(predComb  , CombModelDF$classe)$overall[1]
```

```
## Accuracy
## 0.997791
```

Test on Validation Set

Here we used the method to predict 20 cases.

```
predrf_val <-  predict(Model_rf_all,validation)
predsvmPCA_val <- predict(Model_svm_pca,validationPC)
predgbmPCA_val <- predict(Model_gbm,validationPC)

CombModelDF_val <- data.frame(predgbmPCA=predgbmPCA_val,predsvmPCA=predsvmPCA_val,predrf=predrf_val)

predComb_val <- data.frame( Predict =predict(Model_comb,CombModelDF_val) )

predComb_val
```

```
##    Predict
## 1        B
## 2        A
## 3        B
## 4        A
## 5        A
## 6        E
## 7        D
## 8        B
## 9        A
## 10       A
## 11       B
```

```
## 12       C
## 13       B
## 14       A
## 15       E
## 16       E
## 17       A
## 18       B
## 19       B
## 20       B
```