WeightLiftingDataset

ValdrichFernandes

5/18/2020

Summary

Downloading and opening the datasets.

The data is downloaded to a folder named Data which is in the working directory of the project.

```
if(!file.exists('./Data')) dir.create('./Data')
download.file('https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv',
              './Data/pml-training.csv')
download.file('https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv',
              './Data/pml-testing.csv')
training = read.csv('./Data/pml-training.csv', na.strings=c("", "NA"))
testing = read.csv('./Data/pml-testing.csv', na.strings=c("", "NA"))
```

Loading the required libraries

```
library(dplyr)
library(caret)
library(caretEnsemble)
library(e1071)
library(dplyr)
library(tidyr)
library(ROCR)
library(doParallel)
cluster <- makeCluster(detectCores() - 1)
registerDoParallel(cluster)
```

Data exploration

Cleaning the dataset

The training data consists of 19622 rows and 160 columns. However, of these columns many have upto 0.1019264% missing observations. The columns without any missing observations are selected for the future analysis. Furthermore, the columns user_name and classe are categorical variables and are converted into factors to reflect this.

```
completeCols = which(colSums(is.na(training))==0&names(training)!='X')
training     = training[,completeCols]
timeCols     = grep('time|window', names(training), value = T)

usernames    = unique(training$user_name)
classeUnique = unique(training$classe)
training     = select(training, -all_of(timeCols))%>%
    mutate(user_name = factor(user_name, levels = usernames),
           classe    = factor(classe, levels = classeUnique))
```

The resulting complete training set consists of 19622 rows and 54 columns.

Splitting the training set

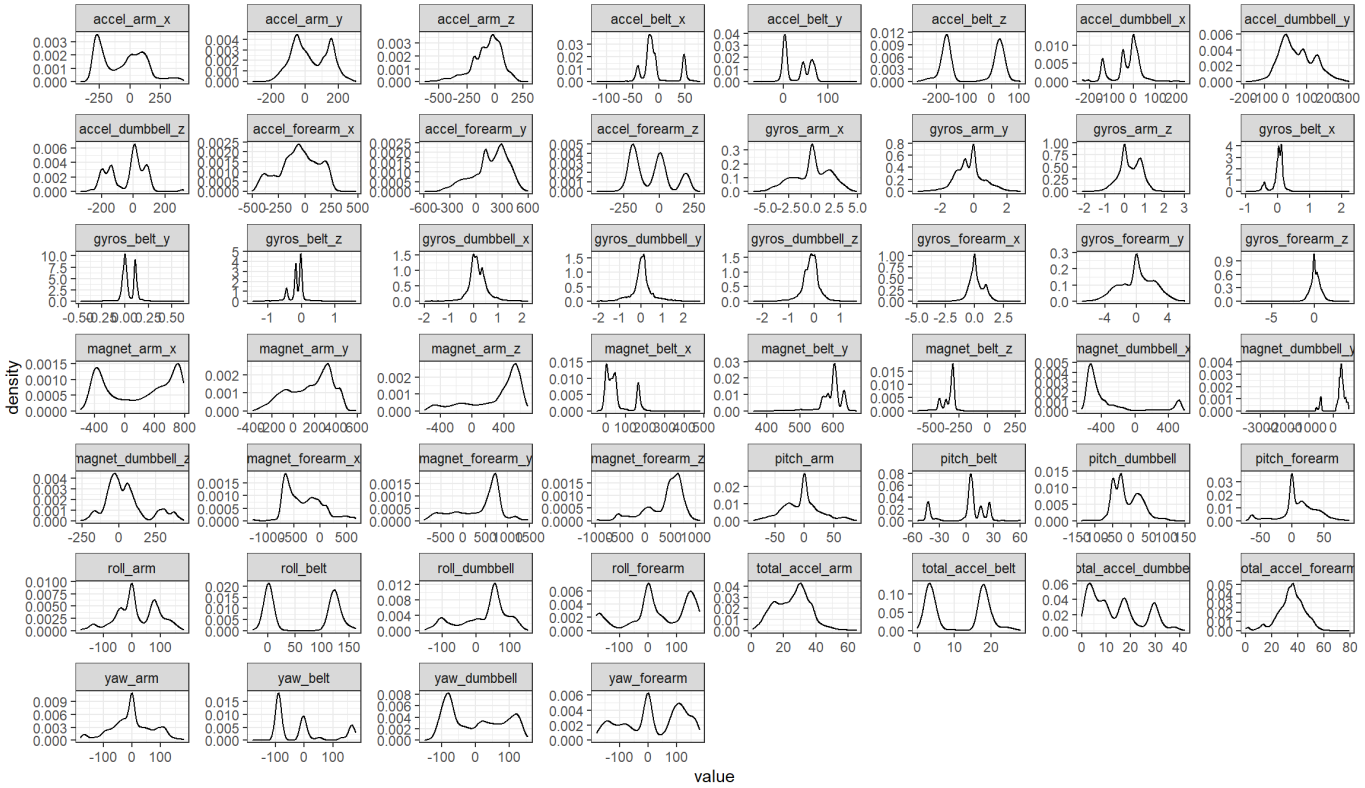
The training data can be further split into a training set and validation set to help evaluate the performance of the different models. They are split in a ration of 60-20-20 for training-testing-validation

```
set.seed(123)
testInd = createDataPartition(training$classe, p = 0.2, list = F)
test = training[testInd,]
train = training[-testInd,]
validationInd = createDataPartition(train$classe, p = 0.25, list = F)
valid = train[validationInd,]
trainE = train[-validationInd,]
rm(train)
```

Density plots

To summarise the data, its range and distribution, a density plot of every column is used. This can help inform any required preprocessing for the model.

```
gathered = gather(trainE, -classe, -user_name,key = 'var', value = 'value')
ggplot(gathered, aes(value))+
    geom_density()+
    facet_wrap(~var, scales = 'free')+
    theme_bw()
```



The variables have different ranges. For most classification models, the data would need to be centered at 0 and scaled to have a stadard deviation of 1. This is to reduce the influence of one variable over the rest hence creating better predictions.

Model preparation

Individual models

The training data is scaled and centered as part of the preprocessing. Furthermore, a Pricipal Component Analysis was used to reduce the number of variables and reduce the variance between models. Three types of models are chosen to classify the data:

1. Random forest (rf)
2. K-Nearest Neighbours (knn)
3. C5.0 (cart)

   All the models have a high accuracy (>90%) with cart being the most accurate as can be seen in the sumary table below.

```
trnCntrl = trainControl(method = "cv", number = 5, allowParallel = T, classProbs = T)
# Random forest
mod.rf = train(classe~., trainE, method = 'rf', trControl = trnCntrl)
# C5.0
mod.cart <- train(classe ~ ., trainE, method="C5.0", trControl = trnCntrl)
# K nearest neighbours
mod.knn = train(classe~., trainE, method = 'knn', trControl = trnCntrl, preProcess = c("center","scale"))
singleRes = list(rf = mod.rf, cart = mod.cart, knn = mod.knn)
summary(resamples(singleRes))$statistics$Accuracy
```

```
#           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
# rf   0.9834254 0.9872557 0.9893798 0.9888684 0.9914966 0.9927844    0
# cart 0.9876858 0.9885350 0.9914966 0.9911640 0.9940501 0.9940527    0
# knn  0.9383765 0.9401020 0.9447514 0.9438347 0.9468989 0.9490446    0
```

Ensemble model

The accuracy of the final prediction can potentially be improved by grouping the three models and predicting based on the estimated probablities of each model. This was done by creating a dataframe containing the probablities in rows and training a random forest model to it. The validation dataset was used for this purpose

```
createProbDf = function(objList, newdata){
    rf = predict(objList$rf, newdata, type = 'prob')
    names(rf) = paste('rf.', names(rf))
    knn = predict(objList$knn, newdata, type = 'prob')
    names(knn) = paste('knn.', names(knn))
    cart = predict(objList$cart, newdata, type = 'prob')
    names(cart) = paste('cart.', names(cart))

    data.frame(rf, knn, cart)
}

ProbDf = createProbDf(singleRes, valid)
ensem = train(ProbDf, valid$classe, method = 'rf', trControl = trnCntrl)
stopCluster(cluster)
registerDoSEQ()
```

Testing the model

The test set is used to evaluate the accuracy of the model in a new dataset. For this, a confusion Matrix is used.

```
confMat = confusionMatrix(predict(ensem, createProbDf(singleRes, test)), test$classe)
confMat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1115    0    0    0    0
##          B    1  758    3    0    2
##          C    0    2  679    3    2
##          D    0    0    3  640    1
##          E    0    0    0    1  717
##
## Overall Statistics
##
##                Accuracy : 0.9954
##                  95% CI : (0.9928, 0.9973)
##     No Information Rate : 0.2842
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9942
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9991   0.9974   0.9912   0.9938   0.9931
## Specificity            1.0000   0.9981   0.9978   0.9988   0.9997
## Pos Pred Value         1.0000   0.9921   0.9898   0.9938   0.9986
## Neg Pred Value         0.9996   0.9994   0.9981   0.9988   0.9984
## Prevalence             0.2842   0.1935   0.1744   0.1640   0.1839
## Detection Rate         0.2839   0.1930   0.1729   0.1630   0.1826
## Detection Prevalence   0.2839   0.1946   0.1747   0.1640   0.1828
## Balanced Accuracy      0.9996   0.9977   0.9945   0.9963   0.9964
```

Conclusion

With an accuracy of 0.9954 on a test dataset, which is higher than the highest accuracy of individual models and hence the ensembled model proves to be effective in generating accurate predictions.