PMLProj

Tania

10/19/2020

Introduction

Predicting the manner in which particpants performed the exercise with the help of the *Weight Lifting Exercise Dataset* from the website : http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har)

The data is collected from the arm,forearm,belt and dumbells of 6 participants.

Importing Libraries and Loading the Data

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.6.3
```

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

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

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.6.3
```

```
## randomForest 4.6-14
```

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

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

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

```
library(Metrics)
```

```
## Warning: package 'Metrics' was built under R version 3.6.3
```

```
##
## Attaching package: 'Metrics'
```

```
## The following objects are masked from 'package:caret':
##
##     precision, recall
```

```
trainData <- read.csv("pml-training.csv",header=TRUE,na.strings=c("NA",""))
testData <- read.csv("pml-testing.csv",header=TRUE,na.strings=c("NA",""))
#replacing all strings such as NA and empty strings to NA value of R
```

Dimensions of training and testing data:

```
dim(trainData)
```

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

```
dim(testData)
```

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

Preprocessing or Cleaning of Data

1. Eliminate variables that have a variance equal to or close to zero.
2. Drop missing values.
3. Drop unnecessary columns.

```
# Remove near zero covariates
NSV <- nearZeroVar(trainData,saveMetrics=TRUE)
trainData <- trainData[,!NSV$nzv]
testData <- testData[,!NSV$nzv]

# Drop missing values
train_filt_na <- trainData[,(colSums(is.na(trainData)) == 0)]
test_filt_na <- testData[,(colSums(is.na(testData)) == 0)]

# Drop unnecessary columns
rmCol_train <- c("user_name","raw_timestamp_part_1","raw_timestamp_part_2","cvtd_timestamp","num
_window")
rmCol_test <- c("user_name","raw_timestamp_part_1","raw_timestamp_part_2","cvtd_timestamp","num_
window","problem_id")
trainData_rmCol <- train_filt_na[,!(names(train_filt_na) %in% rmCol_train)]
testData_rmCol <- test_filt_na[,!(names(test_filt_na) %in% rmCol_test)]
```

The dimensions are:

```
dim(trainData_rmCol)
```

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

```
dim(testData_rmCol)
```

```
## [1] 20 53
```

Partitioning the Dataset

We create the training and validation dataset.

```
inTrain <- createDataPartition(y=trainData$classe, p=0.7, list=FALSE)
train_clean <- trainData_rmCol[inTrain,]
valid_clean <- trainData_rmCol[-inTrain,]
```

```
cor <- abs(sapply(colnames(train_clean[, -ncol(trainData)]), function(x) cor(as.numeric(train_cl
ean[, x]), as.numeric(train_clean$classe), method = "spearman")))
```

No predictors seem to be strongly correlated with the outcome. Linear regression may not be a good option.Therefore we select random forest model.

Random Forest Model

We attempt to fit a random forest model and test the model performance on the validation set.

```
set.seed(71)

# Fit randomforest model
model <- train(classe ~ ., method = "rf", data = train_clean, importance = TRUE, trControl = tra
inControl(method = "cv", number = 4))
model
```

```
## Random Forest
##
## 13737 samples
##    53 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (4 fold)
## Summary of sample sizes: 10302, 10303, 10304, 10302
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9962148  0.9952121
##   27    0.9998544  0.9998159
##   53    0.9997088  0.9996317
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

```
valid_pred <- predict(model, newdata=valid_clean)

# To check the performance of the model
confusionMatrix(valid_pred,valid_clean$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    1    0    0    0
##          B    0 1138    0    0    0
##          C    0    0 1026    1    0
##          D    0    0    0  963    0
##          E    0    0    0    0 1082
##
## Overall Statistics
##
##                Accuracy : 0.9997
##                  95% CI : (0.9988, 1)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9996
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9991   1.0000   0.9990   1.0000
## Specificity            0.9998   1.0000   0.9998   1.0000   1.0000
## Pos Pred Value         0.9994   1.0000   0.9990   1.0000   1.0000
## Neg Pred Value         1.0000   0.9998   1.0000   0.9998   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1934   0.1743   0.1636   0.1839
## Detection Prevalence   0.2846   0.1934   0.1745   0.1636   0.1839
## Balanced Accuracy      0.9999   0.9996   0.9999   0.9995   1.0000
```

Prediction

We now use this model to predict on the testing data.

```
test_pred <- predict(model, newdata=testData_rmCol)
write_files <- function(x) {
        n <- length(x)
        for (i in 1:n) {
                filename <- paste0("problem_id", i, ".txt")
                write.table(x[i], file=filename, quote=FALSE, row.names=FALSE,col.names=FALSE)
        }
}
write_files(test_pred)
```

Results

We used 52 variables to build the random forest model with 100 trees using 4 fold cross validation.

Tvjhdbfg sjkn vsgd sodgn atyddg kh asyfd jkvn sgdfv kjbn fc ehjb sdndfb iauh chvb ehb chjvb sjhd cjhb sjhdb vhb fjkgh dgv kjhvb dhv fkbj zdkhjfb dgljkn vjk dh dkjfb kjfbh zkdjf hoibas dfjkb kxjvh sfjkb fjgh wyg fgonikj