PML project

Smita

9/16/2020

Summary

In this project, the goal is 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. Using the training data set, we will first build a prediction model, and use the 20 test cases to test the efficiency of this prediction model. The data for this project come from this source: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har

Download file.

Find out file characteristics, and add NA for all places that don't have a value.

```
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
A1<-read.csv("pmltrain.csv", header=T, na.strings=c("NA","#DIV/0!",""))
A2<-read.csv("pmltest.csv", header=T, na.strings=c("NA","#DIV/0!",""))
dim(A1)
```

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

```
dim(A2)
```

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

Remove NA values from each table (training and test sets). Transform training data to a cleaner version by removing NAs; removing variables with zero variance to prune the number of variables.

```r
A1C <- A1[, colSums(is.na(A1))==0]
NZV <- nearZeroVar(A1C)
A2C <- A2[, colSums(is.na(A2)) ==0]
Qdata<- A2C[, -c(1:5)]
dim(A1C)
```

```
## [1] 19622    60
```

```r
dim(A2C)
```

```
## [1] 20 60
```

Partitioning the training data set (70%train and 30% test).This splitting will also help compute the out-of-sample errors.

```r
Part7030  <- createDataPartition(A1C$classe, p=0.7, list=FALSE)
TrainSet <- A1C[Part7030, ]
TestSet  <- A1C[-Part7030, ]
```

```r
TrainSet <- TrainSet[, -NZV]
TestSet  <- TestSet[, -NZV]
dim(TrainSet)
```

```
## [1] 13737    59
```

```r
dim(TestSet)
```

```
## [1] 5885   59
```

Further prune by removing ID variables from the dataset.

```r
TrainSet <- TrainSet[, -c(1:5)]
TestSet <- TestSet[, -c(1:5)]
dim(TrainSet)
```

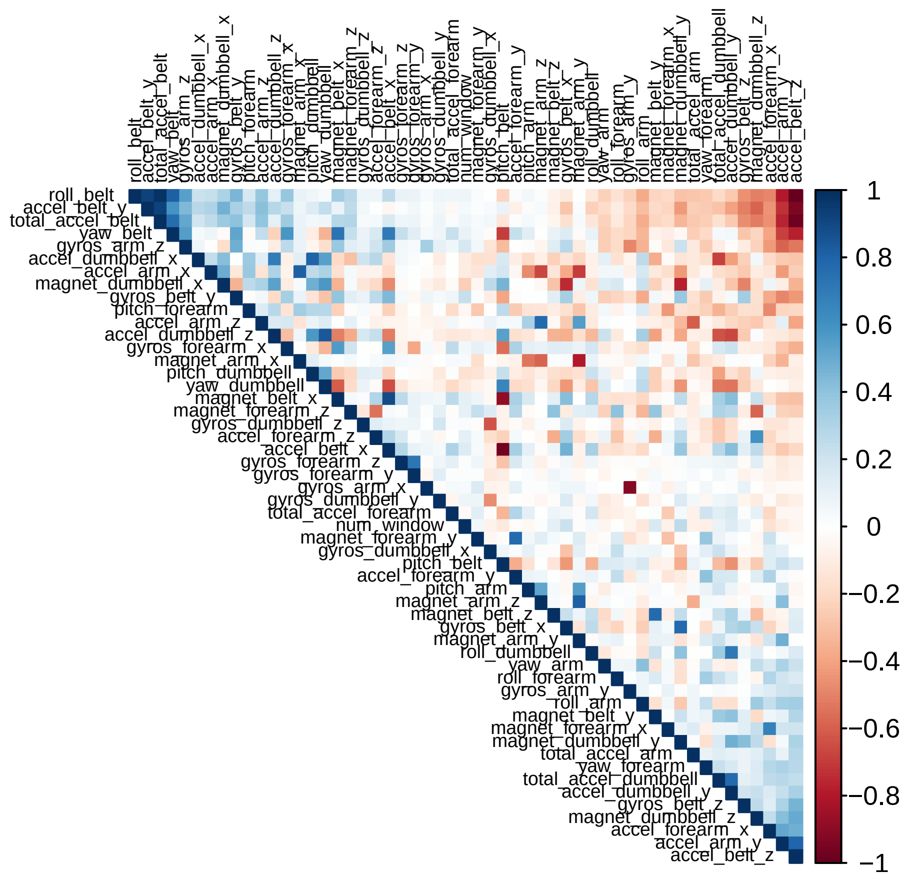
```
## [1] 13737    54
```

```r
dim(TestSet)
```

```
## [1] 5885   54
```

A general correlation between variables.

```
cor_vars <- cor(TrainSet[, -54])
corrplot(cor_vars, order = "FPC", method = "color", type = "upper", tl.cex = 0.6, tl.col = rgb(0, 0, 0))
```



Magnet_forearm_Y and accel_forearm_Y; these two variables are highly correlated.Including both these predictors may not be very useful. A weighted combo (PCA analysis)of predictors should capture most information, reduce number of predictors and reduced noise due to averaging.

Subsetting variables with high positive correlations.

```
highcorr<- findCorrelation(cor_vars, cutoff = 0.75)
highcorr
```

```
##  [1] 11  2 23 10  5 37  9  3 38 36 39 35 22 24 26 14 49 19
```

```
names(TrainSet[highcorr])
```

```
##  [1] "accel_belt_z"      "roll_belt"          "accel_arm_y"
##  [4] "accel_belt_y"      "total_accel_belt"   "accel_dumbbell_z"
##  [7] "accel_belt_x"      "pitch_belt"         "magnet_dumbbell_x"
## [10] "accel_dumbbell_y"  "magnet_dumbbell_y"  "accel_dumbbell_x"
## [13] "accel_arm_x"       "accel_arm_z"        "magnet_arm_y"
## [16] "magnet_belt_z"     "accel_forearm_y"    "gyros_arm_x"
```

Model building

For this project, we will use the following algorithms: Decision trees, Random Forest and genrealised boosted regression model (gbm).

Predicting with Decision tree.

```
library(rpart)
library(rattle)
```

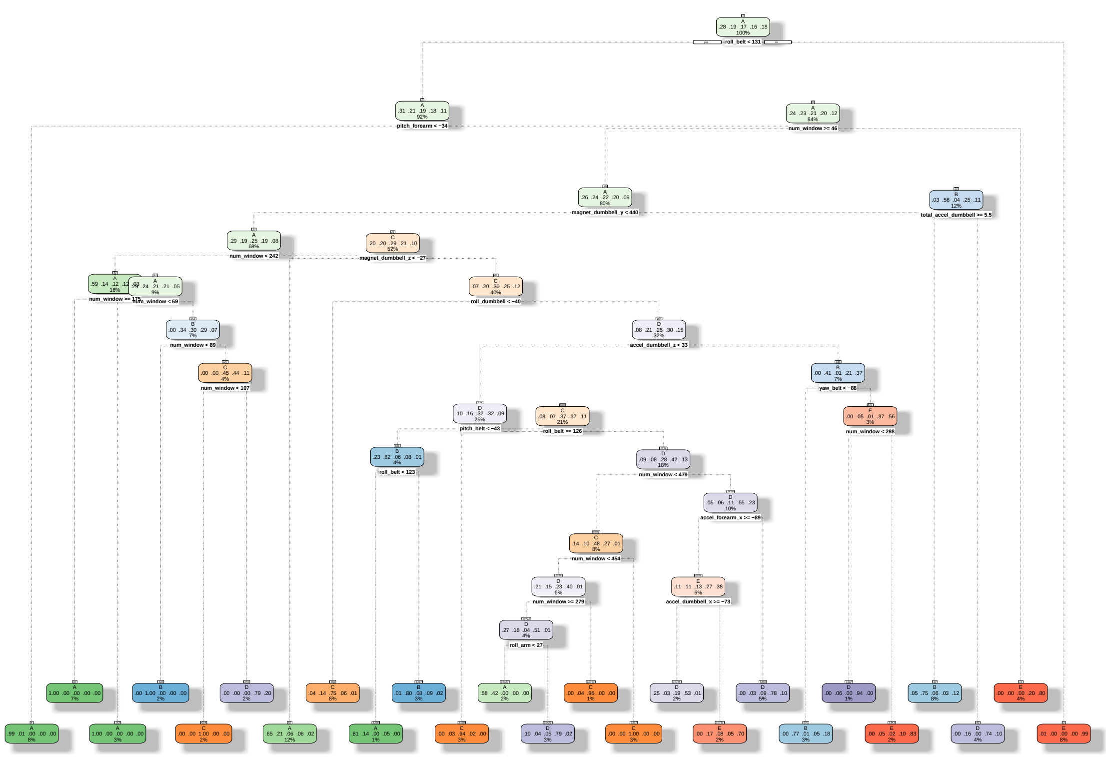
```
## Loading required package: tibble

## Loading required package: bitops

## Rattle: A free graphical interface for data science with R.
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
set.seed(2345)
modFit1<-rpart(classe ~., data=TrainSet, method="class")
fancyRpartPlot(modFit1)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



Rattle 2020−Sep−18 22:40:57 smitajha

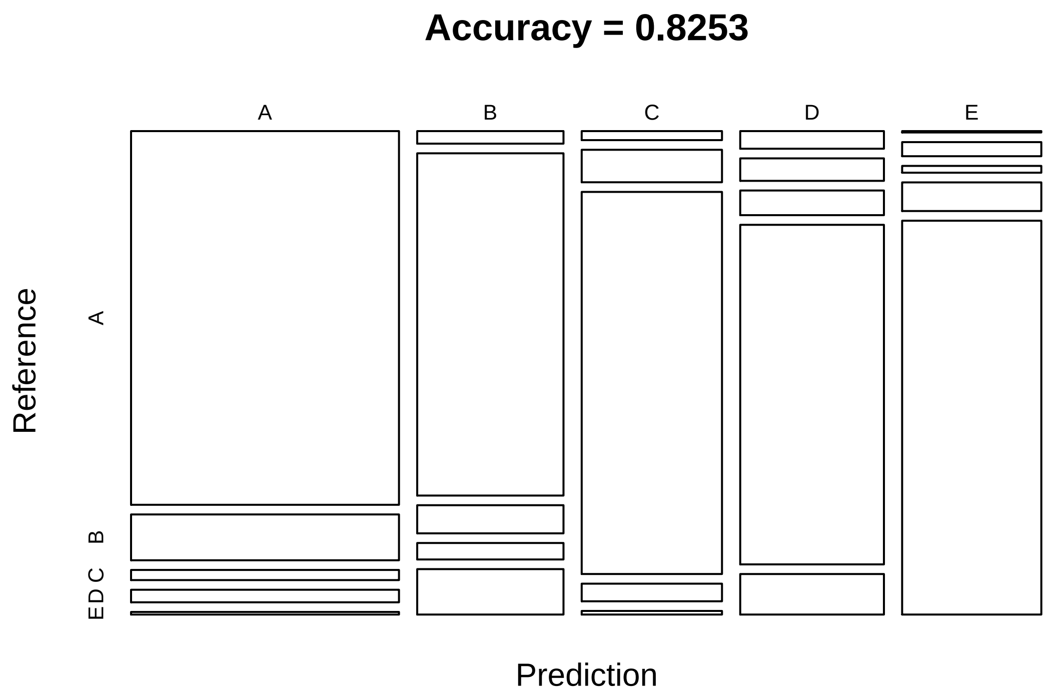
prediction on Test dataset

```r
library(e1071)
predictTree <- predict(modFit1, newdata = TestSet, type = "class")
confTree <- confusionMatrix(predictTree, as.factor(TestSet$classe))
confTree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1582  194   43   53   11
##          B   29  791   65   38  105
##          C   20   72  847   39    8
##          D   40   51   56  771   92
##          E    3   31   15   63  866
##
## Overall Statistics
##
##                Accuracy : 0.8253
##                  95% CI : (0.8154, 0.8349)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7781
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9450   0.6945   0.8255   0.7998   0.8004
## Specificity            0.9285   0.9501   0.9714   0.9514   0.9767
## Pos Pred Value         0.8401   0.7695   0.8590   0.7634   0.8855
## Neg Pred Value         0.9770   0.9284   0.9635   0.9604   0.9560
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2688   0.1344   0.1439   0.1310   0.1472
## Detection Prevalence   0.3200   0.1747   0.1675   0.1716   0.1662
## Balanced Accuracy      0.9368   0.8223   0.8985   0.8756   0.8885
```

Plot the matrix results.

```r
plot(confTree$table, col=confTree$byClass, main=paste("Accuracy =", round(confTree$overall['Accuracy'],
```



Using ML algorithm, Random Forest for prediction.

```
library(randomForest)
```

```
## randomForest 4.6-14

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

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:rattle':
##
##     importance

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

```
modFit <- train(classe ~., method="rf", data=TrainSet, trControl=trainControl(method='cv'), number=5, al
modFit$finalModel
```

```
##
## Call:
```

```
##  randomForest(x = x, y = y, mtry = param$mtry, number = 5, allowParallel = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.21%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3906    0    0    0    0 0.000000000
## B    7 2645    5    1    0 0.004890895
## C    0    5 2390    1    0 0.002504174
## D    0    0    5 2247    0 0.002220249
## E    0    1    0    4 2520 0.001980198
```

##Apply the above model fit to the test data set.

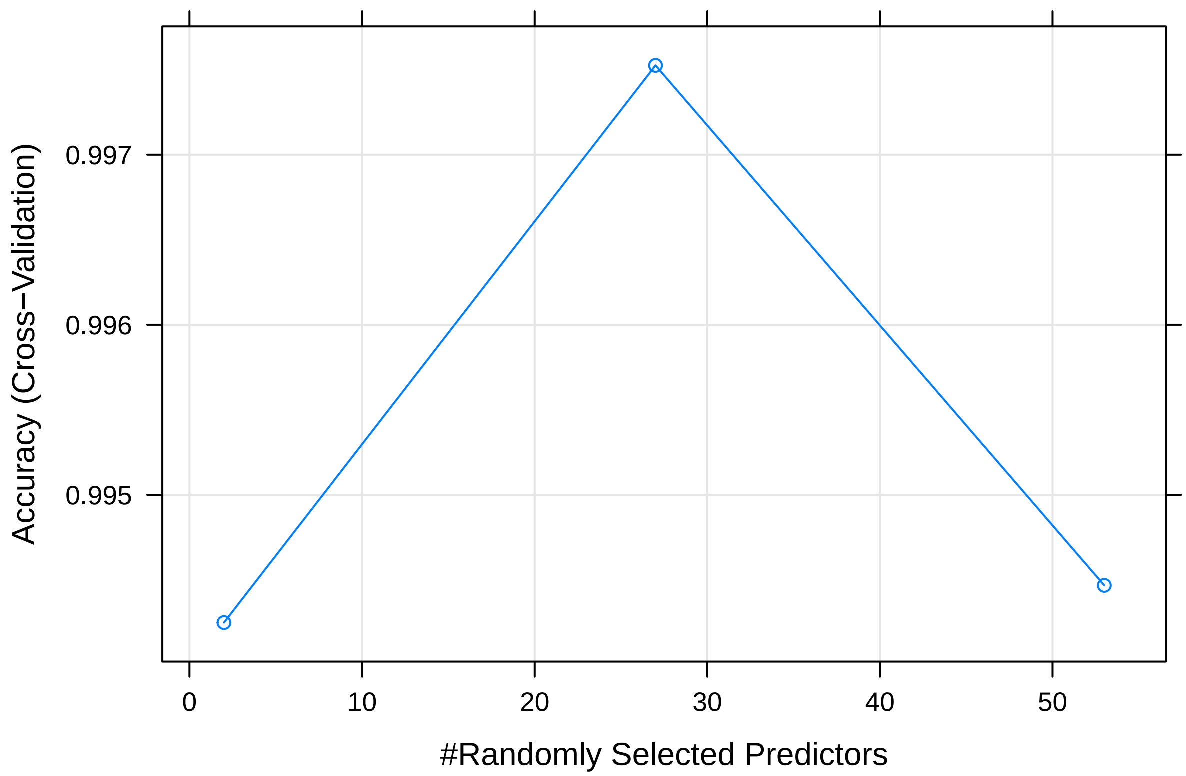
```r
library(e1071)
predictRF <- predict(modFit, newdata = TestSet)
confRF <- confusionMatrix(predictRF, as.factor(TestSet$classe))
confRF
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    6    0    0    0
##          B    1 1133    3    0    0
##          C    0    0 1023    2    0
##          D    0    0    0  962    0
##          E    1    0    0    0 1082
##
## Overall Statistics
##
##                Accuracy : 0.9978
##                  95% CI : (0.9962, 0.9988)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9972
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9988   0.9947   0.9971   0.9979   1.0000
## Specificity            0.9986   0.9992   0.9996   1.0000   0.9998
## Pos Pred Value         0.9964   0.9965   0.9980   1.0000   0.9991
## Neg Pred Value         0.9995   0.9987   0.9994   0.9996   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2841   0.1925   0.1738   0.1635   0.1839
## Detection Prevalence   0.2851   0.1932   0.1742   0.1635   0.1840
## Balanced Accuracy      0.9987   0.9969   0.9983   0.9990   0.9999
```

The accuracy rate using the random forest is very high at .9981. With high accuracy, out of sample error should be minimal but this can also be a case of over-fitting.

Plot the RF model

```
plot(modFit)
```



##Prediction with Generalized Boosted Regression Models RF model already looks like a good prediction model, but we will go ahead and try one more ML algorithm.

```
library(gbm)
```

```
## Loaded gbm 2.1.8
```

```
set.seed(12345)
ctrlGBM <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
modGBM  <- train(classe ~ ., data=TrainSet, method = "gbm", trControl = ctrlGBM, verbose = FALSE)
modGBM$finalModel
```

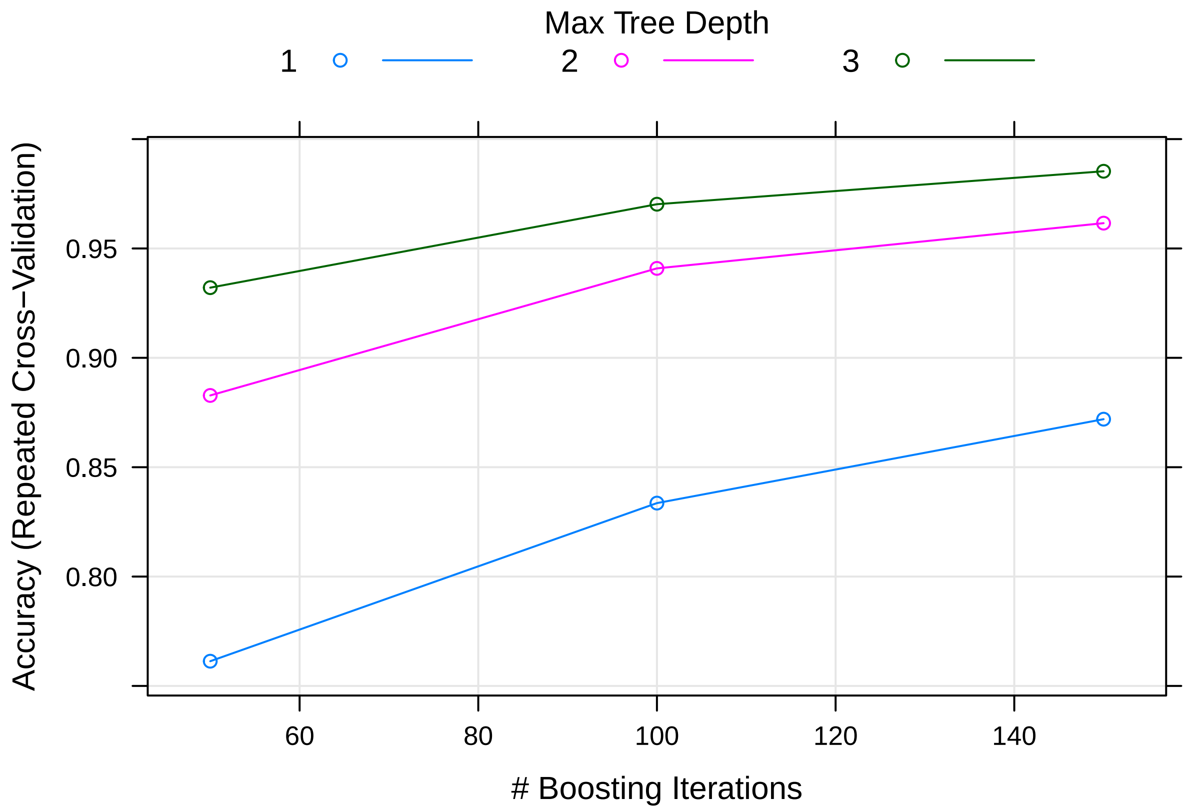
```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 53 predictors of which 53 had non-zero influence.
```

```r
predictGBM <- predict(modGBM, newdata=TestSet)
cmGBM <- confusionMatrix(predictGBM, as.factor (TestSet$classe))
cmGBM
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1661   14    0    0    0
##          B   11 1117    9    7    5
##          C    0    8 1010    9    2
##          D    2    0    5  948   16
##          E    0    0    2    0 1059
##
## Overall Statistics
##
##                Accuracy : 0.9847
##                  95% CI : (0.9812, 0.9877)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9807
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9922   0.9807   0.9844   0.9834   0.9787
## Specificity            0.9967   0.9933   0.9961   0.9953   0.9996
## Pos Pred Value         0.9916   0.9721   0.9815   0.9763   0.9981
## Neg Pred Value         0.9969   0.9954   0.9967   0.9967   0.9952
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2822   0.1898   0.1716   0.1611   0.1799
## Detection Prevalence   0.2846   0.1952   0.1749   0.1650   0.1803
## Balanced Accuracy      0.9945   0.9870   0.9902   0.9894   0.9892
```

Plot the gbm model.

```r
plot(modGBM)
```



The accuracy rate of RF model is the highest, compared to the other two models. So, we will use the RF model to test the required/provided cases.

Note that Test data (Qdata) is minimally processed (removing NAs.

```
Results <- predict(modFit, newdata=Qdata)
Results
```

```
##  [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
```

This predicted model matches with the course project prediction quiz.