#Practical Machine Learning Project : Prediction Assignment Writeup

##I. Overview This document is the final report of the Peer Assessment project from Coursera's course Practical Machine Learning, as part of the Specialization in Data Science. It was built up in RStudio, using its knitr functions, meant to be published in html format. This analysis meant to be the basis for the course quiz and a prediction assignment writeup. The main goal of the project is to predict the manner in which 6 participants performed some exercise as described below. This is the "classe" variable in the training set. The machine learning algorithm described here is applied to the 20 test cases available in the test data and the predictions are submitted in appropriate format to the Course Project Prediction Quiz for automated grading.

##II. 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, your 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.

More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

##Machine Learning - Step by Step process (7 Steps) a) Data Collection (Loading) b) Data Preparation (Cleaning) c) Choosing a model d) Training the Model e) Evaluate the Model f) Parameter Tuning g) Make Predictions

##III. Data Loading and Exploratory Analysis

#a) Data Collection (Loading) The training data for this project are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

```
rm(list=ls())                 # free up memory for download of the data sets
knitr::opts_chunk$set(echo = TRUE)
```

##Environment Preparation We first upload the R libraries that are necessary for the complete analysis.Setting seed, loading libraries and dataset.

```
set.seed(1967)
library(knitr)
library(lattice)
library(ggplot2)
#install.packages("caret", dependencies = TRUE)
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.0.2
```

```
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 4.0.2
```

```
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 4.0.2
```

```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.0.2
```

```
library(RColorBrewer)
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 4.0.2
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.0.2
```

```
library(data.table)
library(corrplot)
library(plotly)
```

```
## Warning: package 'plotly' was built under R version 4.0.2
```

```
library(RGtk2)
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 4.0.2
```

Set working directory

```
setwd("~/0JohnsHopkins/8-PracticalMachineLearning/Course Project I")
```

Load the training and test datasets The next step is loading the data set. The training dataset is then partinioned in 2 to create a Training set (70% of the data) for the modeling process and a Test set (with the remaining 30%) for the validations. The testing data set is not changed and will only be used for the quiz results generation.

```
#command to read trainig and test set
data_train <- read.csv("pml-training.csv")
data_quiz <- read.csv("pml-testing.csv")
dim(data_train)
```

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

```
dim(data_quiz)
```

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

Both created datasets have 160 variables. Those variables have plenty of NA, that can be removed with the cleaning procedures below.

```
in_train  <- createDataPartition(data_train$classe, p=0.70, list=FALSE)
train_set <- data_train[ in_train, ]
test_set  <- data_train[-in_train, ]
dim(train_set)
```

```
## [1] 13737   160
```

```
dim(test_set)
```

```
## [1] 5885  160
```

#b) Data Preparation (Cleaning) The Near Zero variance (NZV) variables are also removed and the ID variables as well.

```
nzv_var <- nearZeroVar(train_set)
train_set <- train_set[ , -nzv_var]
test_set  <- test_set [ , -nzv_var]
dim(train_set)
```

```
## [1] 13737   104
```

```
dim(test_set)
```

```
## [1] 5885  104
```

```
#Remove variables that are mostly NA. A threshlod of 95 % is selected
na_var <- sapply(train_set, function(x) mean(is.na(x))) > 0.95
train_set <- train_set[ , na_var == FALSE]
test_set  <- test_set [ , na_var == FALSE]
dim(train_set)
```

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

```
dim(test_set)
```

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

```
#Since columns 1 to 5 are identification variables only, they will be removed as well
train_set <- train_set[ , -(1:5)]
test_set  <- test_set [ , -(1:5)]
dim(train_set)
```

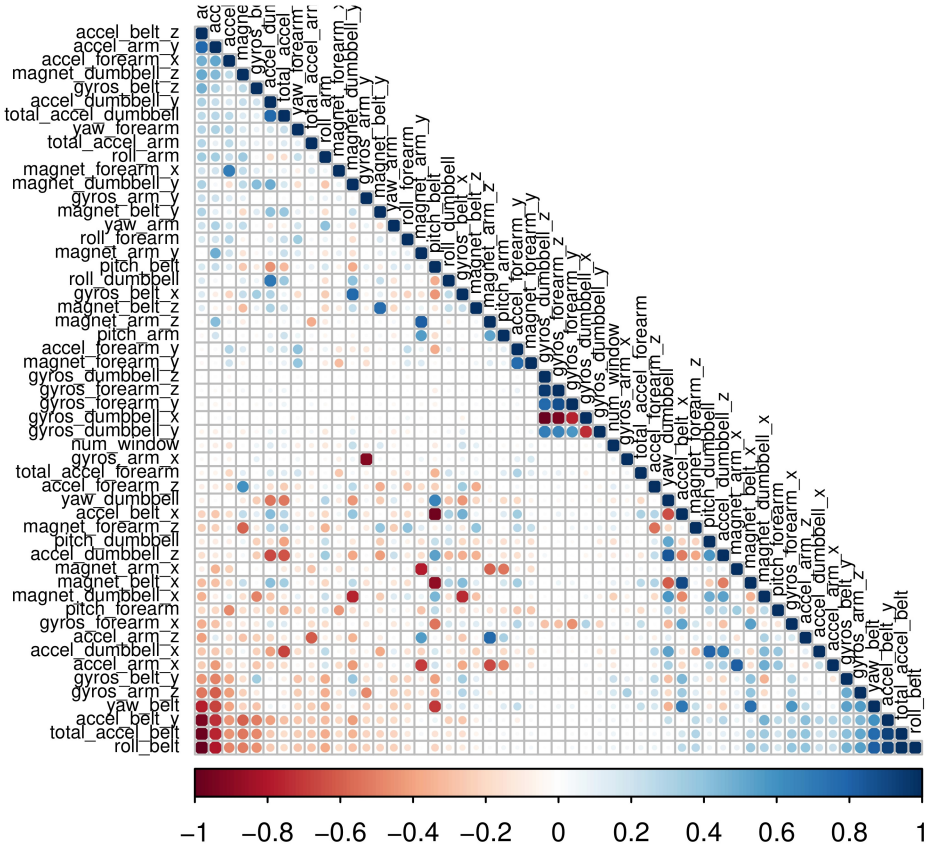
```
## [1] 13737    54
```

```
dim(test_set)
```

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

##Correlation Analysis A correlation among variables could be analyzed before proceeding to the modeling procedures.

```
corr_matrix <- cor(train_set[ , -54])
corrplot(corr_matrix, order = "FPC", method = "circle", type = "lower",
         tl.cex = 0.6, tl.col = rgb(0, 0, 0))
```



The number of variables for the analysis has been reduced from the original 160 down to 54

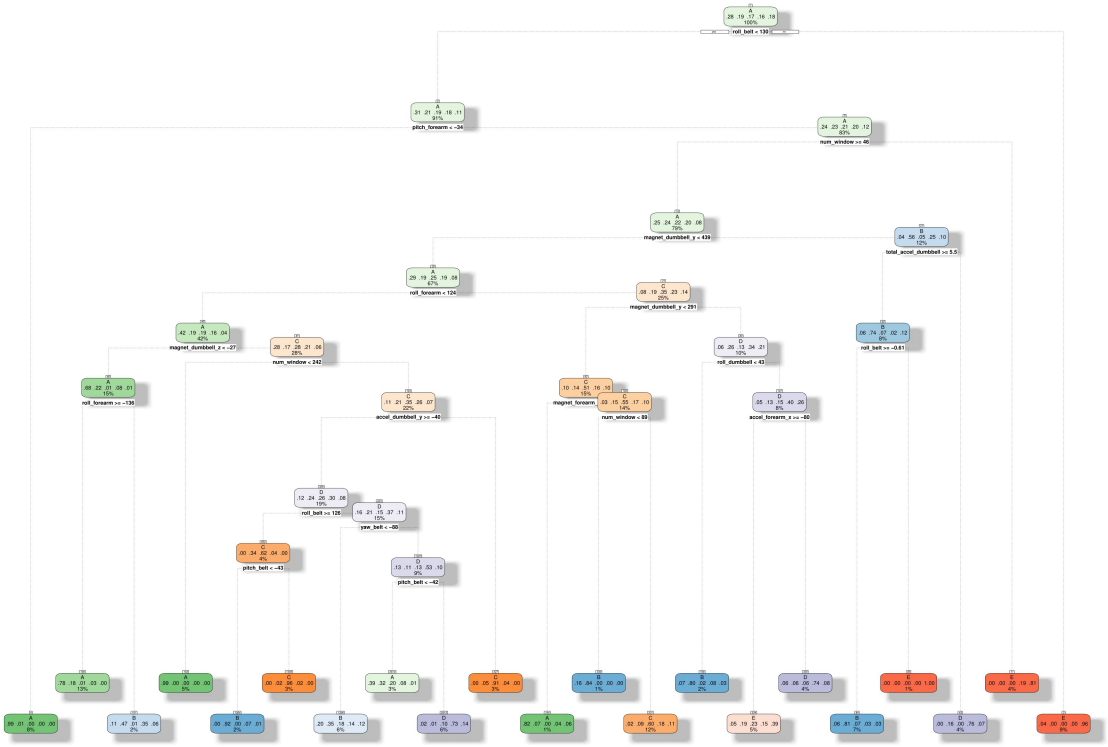
#c) Choosing a model Three methods will be applied to model the regressions (in the Train data set) and the best one (with higher accuracy when applied to the Test data set) will be used for the quiz predictions. The methods are: - Decision Tree, - Generalized Boosted Model (GBM) and - Random Forests (rf), as described below.

#d) Training the Model A Confusion Matrix is plotted at the end of each analysis to better visualize the accuracy of the models.

#Decision Tree Model

```
set.seed(1967)
fit_DT <- rpart(classe ~ ., data = train_set, method="class")
fancyRpartPlot(fit_DT)
```

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



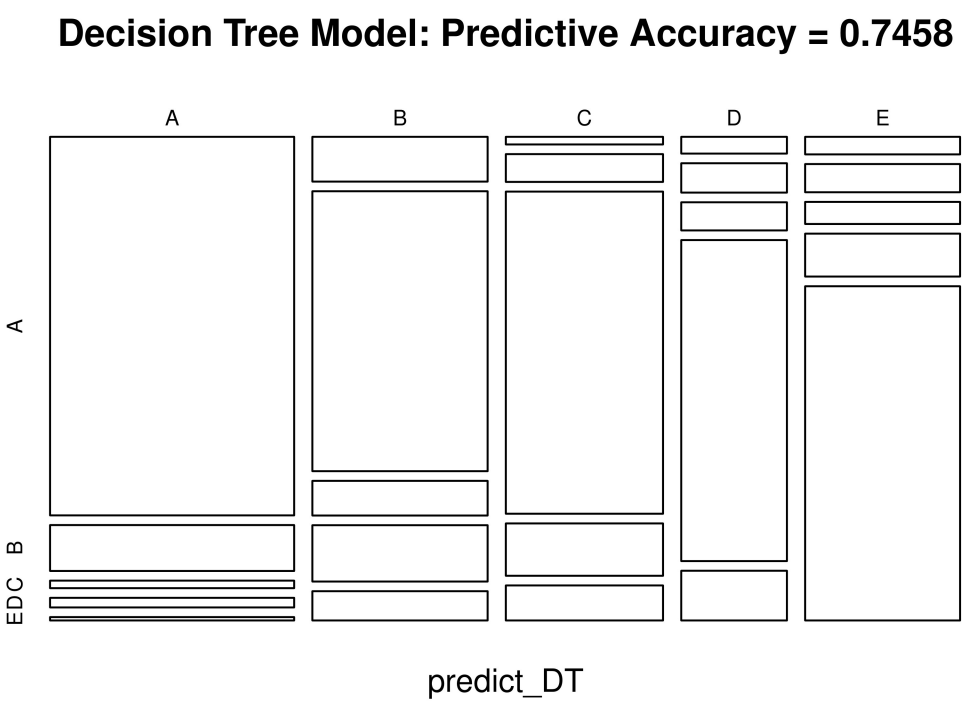
Rattle 2020–Jul–12 02:02:01 cesar

```
predict_DT <- predict(fit_DT, newdata = test_set, type="class")
conf_matrix_DT <- confusionMatrix(table(predict_DT, test_set$classe))
conf_matrix_DT
```

```
## Confusion Matrix and Statistics
##
##
## predict_DT    A    B    C    D    E
##          A 1460  177   29   37   13
##          B  124  775   96  156   81
##          C   19   69  800  130   87
##          D   28   49   47  536   83
##          E   43   69   54  105  818
##
## Overall Statistics
##
##                Accuracy : 0.7458
##                  95% CI : (0.7345, 0.7569)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6779
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
```

```
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8722   0.6804   0.7797  0.55602   0.7560
## Specificity            0.9392   0.9037   0.9372  0.95794   0.9436
## Pos Pred Value         0.8508   0.6291   0.7240  0.72140   0.7511
## Neg Pred Value         0.9487   0.9218   0.9527  0.91676   0.9450
## Prevalence             0.2845   0.1935   0.1743  0.16381   0.1839
## Detection Rate         0.2481   0.1317   0.1359  0.09108   0.1390
## Detection Prevalence   0.2916   0.2093   0.1878  0.12625   0.1850
## Balanced Accuracy      0.9057   0.7921   0.8585  0.75698   0.8498
```

```r
plot(conf_matrix_DT$table, col = conf_matrix_DT$byClass,
     main = paste("Decision Tree Model: Predictive Accuracy =",
                  round(conf_matrix_DT$overall['Accuracy'], 4)))
```



#Generalized Boosted Model (GBM)

```r
set.seed(1967)
ctrl_GBM <- trainControl(method = "repeatedcv", number = 5, repeats = 2)
fit_GBM  <- train(classe ~ ., data = train_set, method = "gbm",
                  trControl = ctrl_GBM, verbose = FALSE)
fit_GBM$finalModel
```

```
## A gradient boosted model with multinomial loss function.
```

```
## 150 iterations were performed.
## There were 53 predictors of which 53 had non-zero influence.
```

```
predict_GBM <- predict(fit_GBM, newdata = test_set)
conf_matrix_GBM <- confusionMatrix(table(predict_GBM, test_set$classe))
conf_matrix_GBM
```

```
## Confusion Matrix and Statistics
##
##
## predict_GBM    A    B    C    D    E
##            A 1665   13    0    1    1
##            B    8 1120    8    2    4
##            C    0    6 1016   11    3
##            D    1    0    2  950   10
##            E    0    0    0    0 1064
##
## Overall Statistics
##
##                Accuracy : 0.9881
##                  95% CI : (0.985, 0.9907)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.985
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9946   0.9833   0.9903   0.9855   0.9834
## Specificity            0.9964   0.9954   0.9959   0.9974   1.0000
## Pos Pred Value         0.9911   0.9807   0.9807   0.9865   1.0000
## Neg Pred Value         0.9979   0.9960   0.9979   0.9972   0.9963
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2829   0.1903   0.1726   0.1614   0.1808
## Detection Prevalence   0.2855   0.1941   0.1760   0.1636   0.1808
## Balanced Accuracy      0.9955   0.9893   0.9931   0.9914   0.9917
```

#Random Forest

```
set.seed(1967)
ctrl_RF <- trainControl(method = "repeatedcv", number = 5, repeats = 2)
fit_RF  <- train(classe ~ ., data = train_set, method = "rf",
                 trControl = ctrl_RF, verbose = FALSE)
fit_RF$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, verbose = FALSE)
##                Type of random forest: classification
##                      Number of trees: 500
```

```
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.2%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3905    0    0    0    1 0.0002560164
## B    7 2648    3    0    0 0.0037622272
## C    0    4 2391    1    0 0.0020868114
## D    0    0    4 2246    2 0.0026642984
## E    0    0    0    6 2519 0.0023762376
```

```
predict_RF <- predict(fit_RF, newdata = test_set)
conf_matrix_RF <- confusionMatrix(table(predict_RF, test_set$classe))
conf_matrix_RF
```

```
## Confusion Matrix and Statistics
##
##
## predict_RF    A    B    C    D    E
##          A 1674    1    0    0    0
##          B    0 1137    1    0    0
##          C    0    1 1025    3    0
##          D    0    0    0  961    2
##          E    0    0    0    0 1080
##
## Overall Statistics
##
##                Accuracy : 0.9986
##                  95% CI : (0.9973, 0.9994)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9983
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9982   0.9990   0.9969   0.9982
## Specificity            0.9998   0.9998   0.9992   0.9996   1.0000
## Pos Pred Value         0.9994   0.9991   0.9961   0.9979   1.0000
## Neg Pred Value         1.0000   0.9996   0.9998   0.9994   0.9996
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1932   0.1742   0.1633   0.1835
## Detection Prevalence   0.2846   0.1934   0.1749   0.1636   0.1835
## Balanced Accuracy      0.9999   0.9990   0.9991   0.9982   0.9991
```

#e) Evaluate the Model (Fitting models)

#Applying the Best Predictive Model to the Test Data To summarize, the predictive accuracy of the three models evaluated is as follows:

Summary of the results: - Decision tree model - is the worst model running, has the low mean and the

highest standard deviation. - GBM model - has a decent mean accuracy but a little bit lower accuracy than RF, - Random Fores model - has the highest mean accuracy and lowest standard deviation

#f) Parameter Tuning Checking prediction accuracy on my own testing/validation set. I am expecting similar accuracy as the mean from the cross validation.

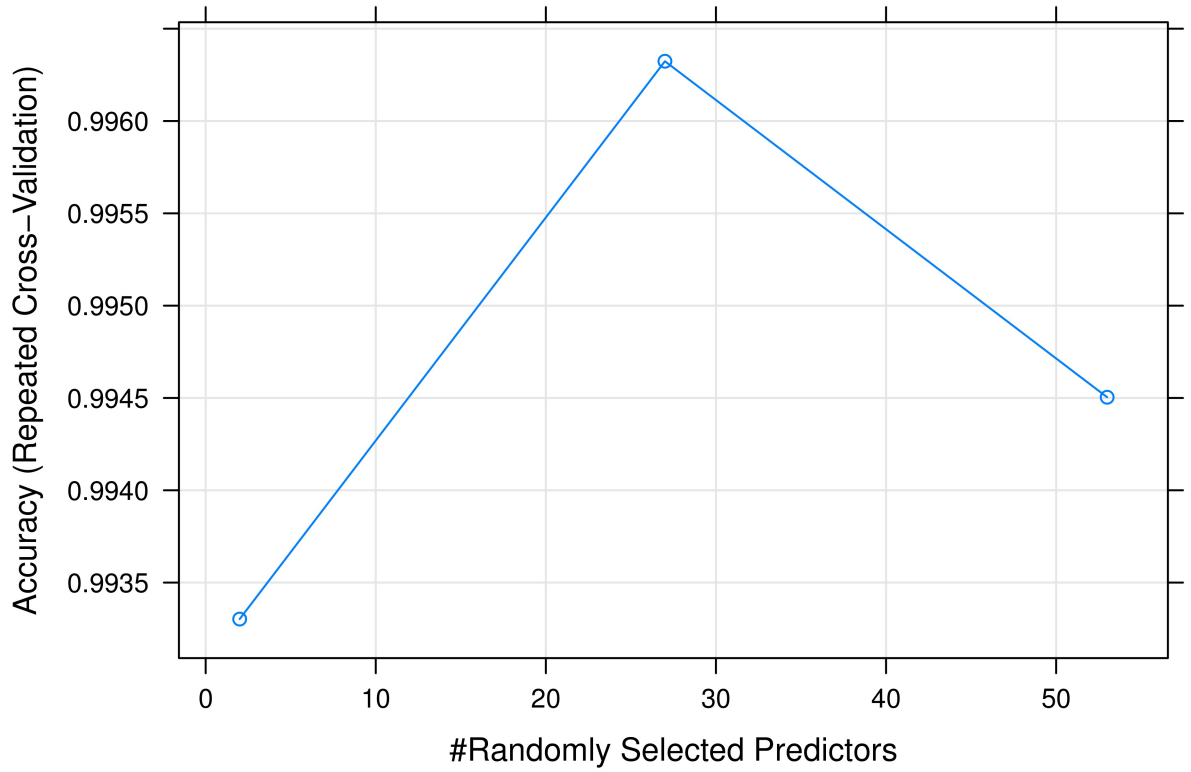
##The kappa statistic The kappa statistic (labeled Kappa in the previous output) adjusts accuracy by accounting for the possibility of a correct prediction by chance alone. Kappa values range to a maximum value of 1, which indicates perfect agreement between the model's predictions and the true values—a rare occurrence. Values less than one indicate imperfect agreement.

Depending on how your model is to be used, the interpretation of the kappa statistic might vary. One common interpretation is shown as follows: • Poor agreement = Less than 0.20 • Fair agreement = 0.20 to 0.40 • Moderate agreement = 0.40 to 0.60 • Good agreement = 0.60 to 0.80 • Very good agreement = 0.80 to 1.00

This three models preforms as expected, the deviation from the cross validation accuracy is low and I do not see a reason to change resampling method or adding repetitons.

Checking if there are anything to gain from increasing the number of boosting iterations.

```
plot(fit_RF)
```




```
print(fit_RF$bestTune)
```

```
##   mtry
## 2   27
```

The predictive accuracy of the Random Forest model is excellent at 99.8 %. Accuracy has plateaued, and further tuning would only yield decimal gain. - The best tuning parameters hads 150 trees (boosting iterations), - interaction depth 3 - shrinkage 0.1.

#g) Make Predictions Deciding to predict with this model.

Decision Tree Model: 74.58 % Generalized Boosted Model: 98.81 % Random Forest Model: 99.86 %

The Random Forest model is selected and applied to make predictions on the 20 data points from the original testing dataset (data_quiz).

```
cat("Predictions: ", paste(predict(fit_RF, data_quiz)))
```

```
## Predictions:  B A B A A E D B A A B C B A E E A B B B
```