Practical Machine Learning Coursera Course Project

Predicting the manner in which people do excercise

Diego Alonso Lazo Paz

```
train<-read.csv("pml-training.csv")
test<-read.csv("pml-testing.csv")
val_df<-test
```

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.
- The objective of this project is to predict the manner in which people do excercise using different machine learning models.

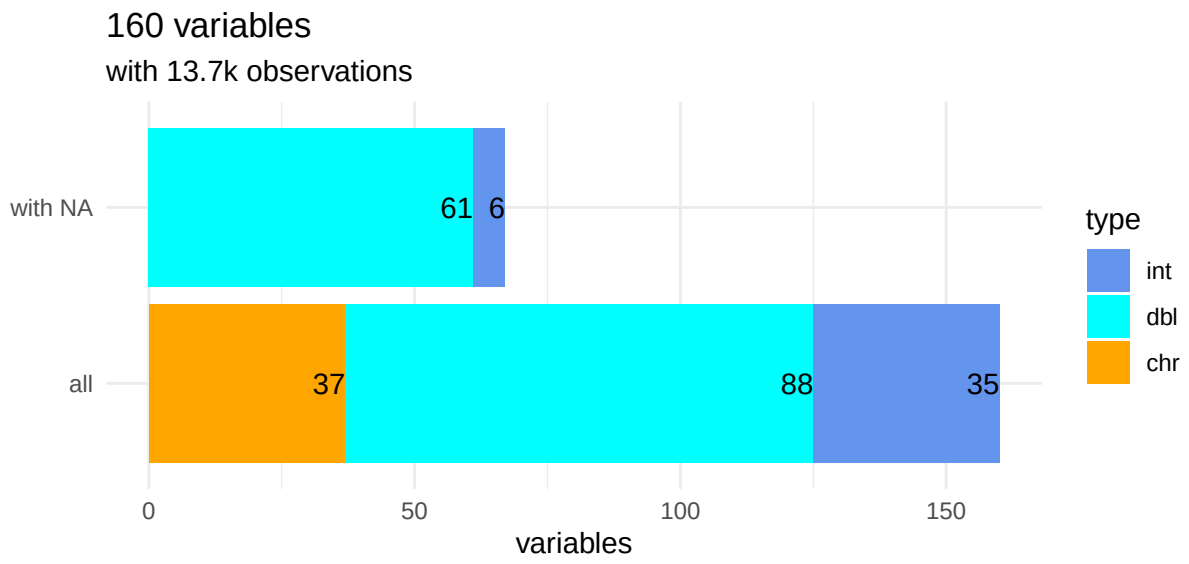
Partition of the traning data

```
indx  <- createDataPartition(train$classe, p=0.7, list=FALSE)
Traindf0 <- train[indx, ]
Testdf0  <- train[-indx, ]
```

Exploratory data Analysis

- In this part we look to understand the data in order to obtain insights and have a first look into the behaviour of the variables

```
Traindf0 %>% explore_tbl()
```

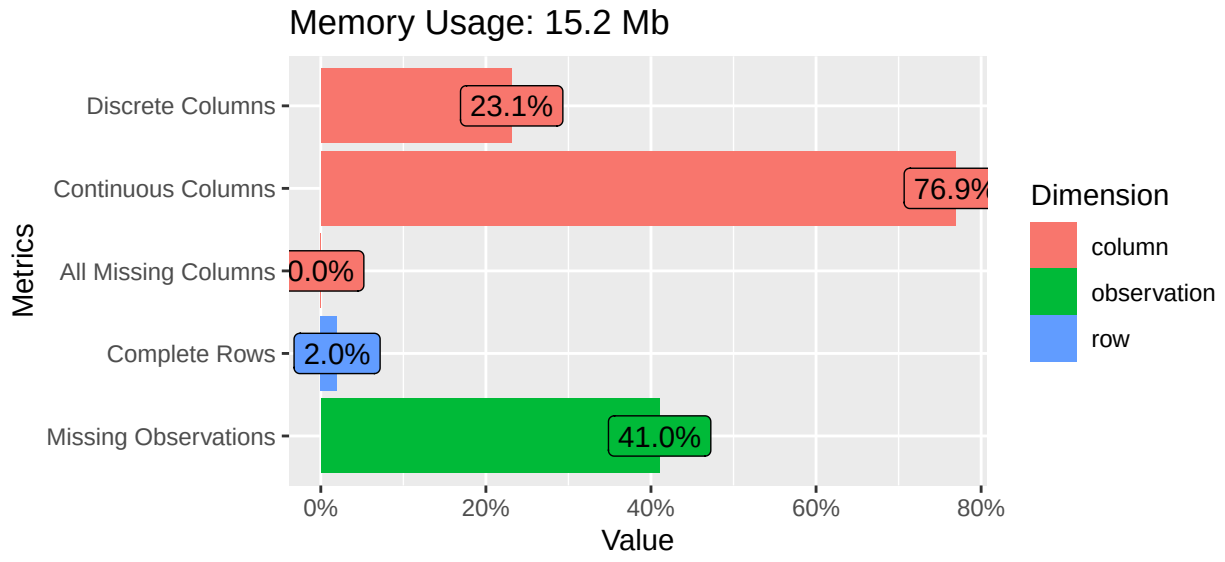



```
introduce(Traindf0)
```

```
##    rows columns discrete_columns continuous_columns all_missing_columns
## 1 13737     160               37                123                   0
##   total_missing_values complete_rows total_observations memory_usage
## 1               902222           271            2197920     15975560
```

- We 19.6k observations and 160 variables, which 123 are double or integer, the rest are characters.
- 61 double variables have NA values.

```
plot_intro(Traindf0)
```

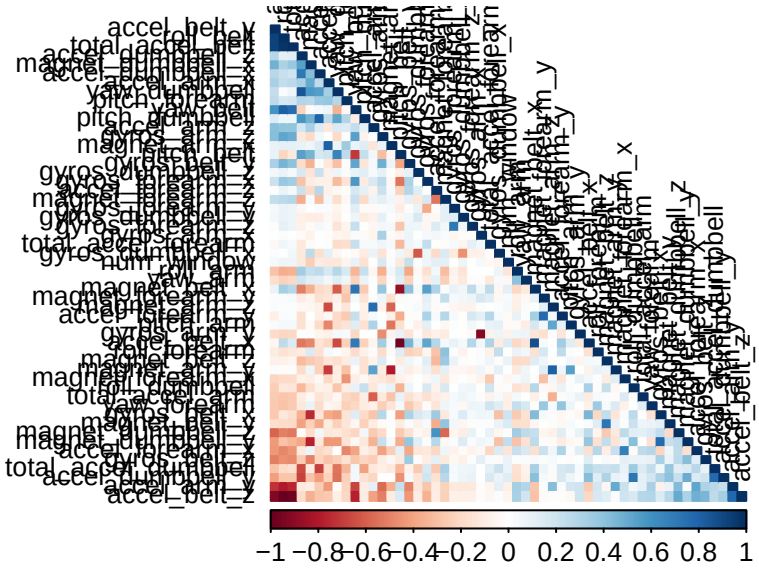


* So we are going to remove the variables with too many missing values and also the the variables with near zero variance

```
#Remove variable with many NAS
Traindf1<- Testdf0[, colSums(is.na(Testdf0)) == 0]
Testdf1 <- Testdf0[, colSums(is.na(Testdf0)) == 0]
# remove variables with Nearly Zero Variance
near_zero_var <- nearZeroVar(Traindf1)
Traindf2 <- Traindf1[, -near_zero_var]
Testdf2  <- Testdf1[, -near_zero_var]
# remove identification only variables (columns 1 to 5)
Traindf3 <- Traindf2[, -(1:5)]
Testdf3  <- Testdf2[, -(1:5)]
```

Correlation Analysis

```
corMatrix <- cor(Traindf3[, -54])
corrplot(corMatrix, order = "FPC", method = "color", type = "lower",
         tl.cex = 0.8, tl.col = rgb(0, 0, 0))
```



* The plot is not much of a help , so with use a function to find the variables with the highest correlation

```
highlyCorrelated = findCorrelation(corMatrix, cutoff=0.80)
names(Traindf3)[highlyCorrelated]
```

```
##  [1] "accel_belt_z"     "roll_belt"        "accel_belt_y"     "accel_dumbbell_z"
##  [5] "accel_belt_x"     "pitch_belt"       "accel_dumbbell_x" "accel_arm_x"
##  [9] "magnet_arm_y"     "gyros_arm_x"
```

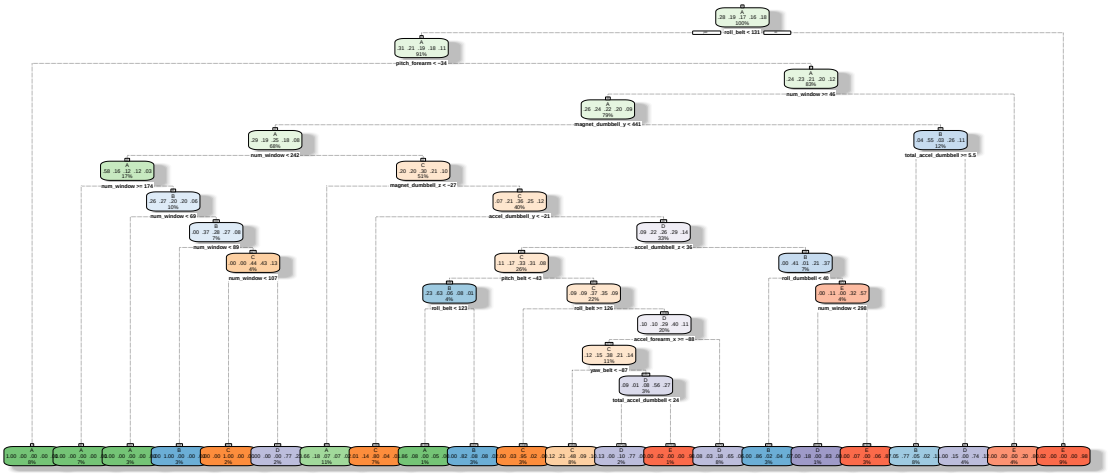
Predictive Modelling

- In order to obtain the best prediction, we are going to use differente models to get the best prediction, and if it is possible, combine predictions to get the best prediction possible
- For this part we are going to use the next models: simple classification tree (rpart), K-Nearest Neighbords (knn), random forest and a Generalized Boosted Regression Models(gbm)

Classification tree

```
set.seed(123)
class_tree <- rpart(classe ~ ., data=Traindf3, method="class")
fancyRpartPlot(class_tree)
```

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



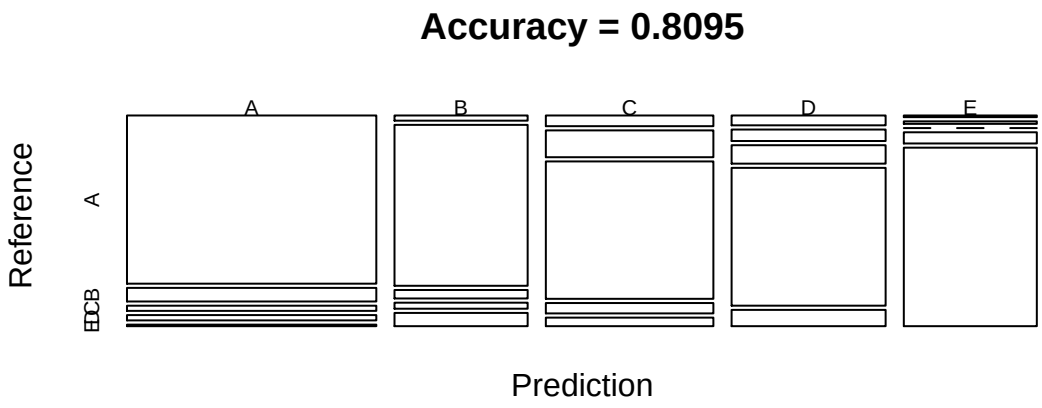
Rattle 2020−Jul.−03 13:38:00 USUARIO

```
predict_clstree <- predict(class_tree, Testdf3, type = "class")
conf_mtree <- confusionMatrix(predict_clstree, Testdf3$classe %>% as.factor())
conf_mtree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1521  123   47   48   14
##          B   25  777   41   29   64
##          C   65  163  835   63   52
##          D   55   64  103  770   91
##          E    8   12    0   54  861
##
## Overall Statistics
##
##                Accuracy : 0.8095
##                  95% CI : (0.7992, 0.8195)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.759
##
##  Mcnemar's Test P-Value : < 2.2e-16
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9086   0.6822   0.8138   0.7988   0.7957
## Specificity            0.9449   0.9665   0.9294   0.9364   0.9846
## Pos Pred Value         0.8677   0.8301   0.7088   0.7110   0.9209
## Neg Pred Value         0.9630   0.9269   0.9594   0.9596   0.9554
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2585   0.1320   0.1419   0.1308   0.1463
## Detection Prevalence   0.2979   0.1590   0.2002   0.1840   0.1589
## Balanced Accuracy      0.9268   0.8243   0.8716   0.8676   0.8902
```

```
plot(conf_mtree$table, col = conf_mtree$byClass, main = paste("Accuracy =", round(conf_mtree$overall['A
```



* With this simple model we get an accuracy of 0.82

K-Nearest Neighbords

```
set.seed(12345)
ctrlKNN <- trainControl(method="cv", number=5, classProbs= TRUE, summaryFunction = multiClassSummary)
m_knn <- train(classe~.,
               data=Traindf3,
               trControl = ctrlKNN,
               method="knnn" )
m_knn$finalModel
```

```
##
## Call:
## kknn::train.kknn(formula = .outcome ~ ., data = dat, kmax = param$kmax,     distance = param$distanc
##
## Type of response variable: nominal
## Minimal misclassification: 0.03568394
```

```
## Best kernel: optimal
## Best k: 1
```

```
predict_knn <- predict(m_knn, newdata=Testdf3)
cmkn <- confusionMatrix(predict_knn, Testdf3$classe %>% as.factor())
cmkn
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    0 1139    0    0    0
##          C    0    0 1026    0    0
##          D    0    0    0  964    0
##          E    0    0    0    0 1082
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9994, 1)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

Random Forest

```
set.seed(12345)
ctrlRF <- trainControl(method="cv", number=5, verboseIter=FALSE)
rf_model <- train(classe ~ .,
                  data=Traindf3,
                  trControl = ctrlRF,
                  method="rf")
rf_model$finalModel
```

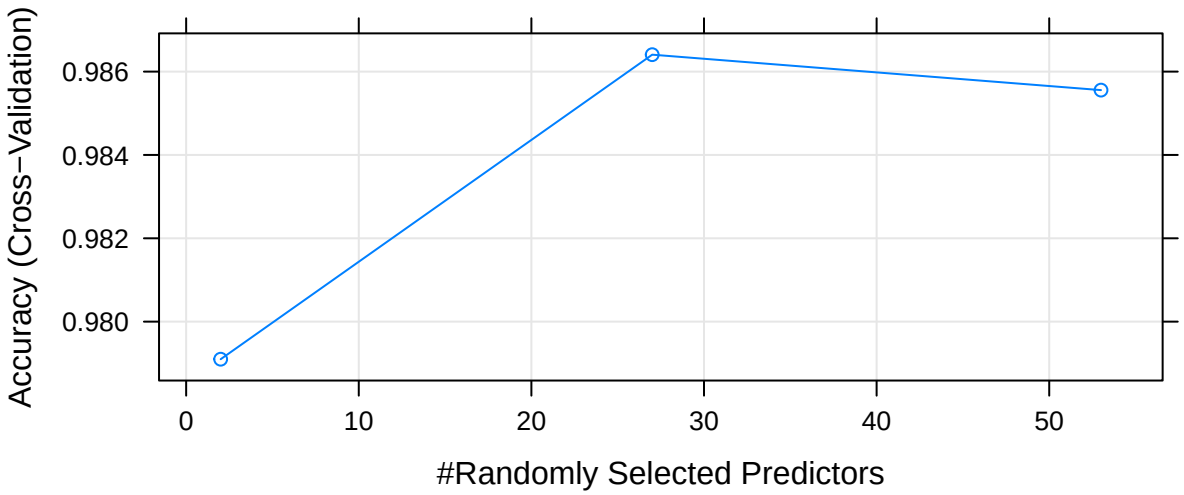
```
##
```

```
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 27
##
##         OOB estimate of  error rate: 0.99%
## Confusion matrix:
##      A    B    C   D    E  class.error
## A 1673    1    0   0    0 0.0005973716
## B   12 1121    6   0    0 0.0158033363
## C    0   14 1010   2    0 0.0155945419
## D    0    0   13 950    1 0.0145228216
## E    0    4    0   5 1073 0.0083179298
```

```
predict_RF <- predict(rf_model, newdata=Testdf3)
cmrf <- confusionMatrix(predict_RF, Testdf3$classe %>% as.factor())
cmrf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    0 1139    0    0    0
##          C    0    0 1026    0    0
##          D    0    0    0  964    0
##          E    0    0    0    0 1082
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9994, 1)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

```r
plot(rf_model)
```



Generalized Boosted Regression Models(gbm)

```r
set.seed(12345)
ctrlGBM <- trainControl(method = "repeatedcv", number = 5)
gbm_model <- train(classe ~ .,
                   data=Traindf3,
                   trControl = ctrlGBM,
                   method="gbm",
                   verbose=FALSE)
gbm_model$finalModel
```

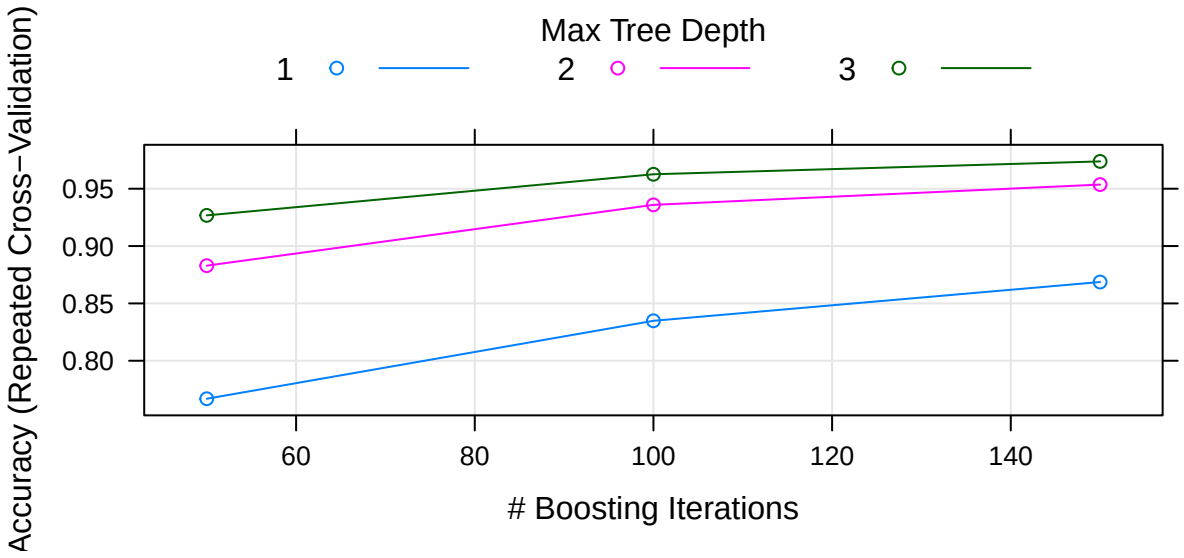
```
## A gradient boosted model with multinomial loss function.
## 150 iterations were performed.
## There were 53 predictors of which 53 had non-zero influence.
```

```r
predict_gbm <- predict(gbm_model, newdata=Testdf3)
cmgbm <- confusionMatrix(predict_gbm, Testdf3$classe %>% as.factor())
cmgbm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    8    0    0    0
##          B    1 1125    5    2    1
##          C    0    6 1020    7    2
##          D    0    0    1  955    4
##          E    0    0    0    0 1075
##
```

```
## Overall Statistics
##
##                Accuracy : 0.9937
##                  95% CI : (0.9913, 0.9956)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.992
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9877   0.9942   0.9907   0.9935
## Specificity            0.9981   0.9981   0.9969   0.9990   1.0000
## Pos Pred Value         0.9952   0.9921   0.9855   0.9948   1.0000
## Neg Pred Value         0.9998   0.9971   0.9988   0.9982   0.9985
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1912   0.1733   0.1623   0.1827
## Detection Prevalence   0.2856   0.1927   0.1759   0.1631   0.1827
## Balanced Accuracy      0.9988   0.9929   0.9955   0.9948   0.9968
```

```
plot(gbm_model)
```



Apply the best model to validation data

The accuracy of the 4 classification models are: Decision Tree : 0.8054 KNN: 0.999 Random Forest : 0.999 GBM : 0.9932 * We choose the GBM because de RF and KNN are overfitting the data

```
Results <- predict(gbm_model, newdata=val_df)
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
```