Practical Machine Learning - Project

1. Overview

This report describes how data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants is used to predict whether they perform barbell lifts correctly or incorrectly.

1.1 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://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

1.2 Data sources

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

2. Loading libraries and data, Cleaning data

2.1 Loading Libraries

```
library(lattice)
library(ggplot2)
library(caret)
library(kernlab)
library(rattle)
library(corrplot)
set.seed(13)
```

2.2 Download data

```
trainUrl <-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

if (!file.exists("./data")) {
  dir.create("./data")
}
if (!file.exists("./data/pml-training.csv")) {
  download.file(trainUrl, destfile = "./data/pml-training.csv", method = "curl")
}
if (!file.exists("./data/pml-testing.csv")) {
  download.file(testUrl, destfile = "./data/pml-testing.csv", method = "curl")
}
```

2.3 Read data

```
train_dta <- read.csv("./data/pml-training.csv")
test_dta <- read.csv("./data/pml-testing.csv")
```

The training set has 19622 **observations of** 160 **variables, while the testing set has** 20 **observations of** 160 **variables.**

2.4 Cleaning data

We want to keep only the relevant variables of the training set. We exlude irrelevant variables in three steps:

a) Remove N/A variables:

```
train_dta <- train_dta[,colMeans(is.na(train_dta)) < .9]
```

b) Remove metadata:

```
train_dta <- train_dta[,-c(1:7)]
```

c) Remove variables with near zero variance:

```
nzv <- nearZeroVar(train_dta)
train_dta <- train_dta[,-nzv]
```

After the cleaning the training set has 19622 **observations of** 53 **variables.**

2.5 Spilt data into training and test dataset

```
inTrain <- createDataPartition(y = train_dta$classe, p = 0.7, list = FALSE)
train <- train_dta[inTrain,]
test <- train_dta[-inTrain,]
```

3. Modeling and testing models

The following models will be tested:

- Decision Tree
- Random Forest
- Gradient Boosted Trees
- Support Vector Machine

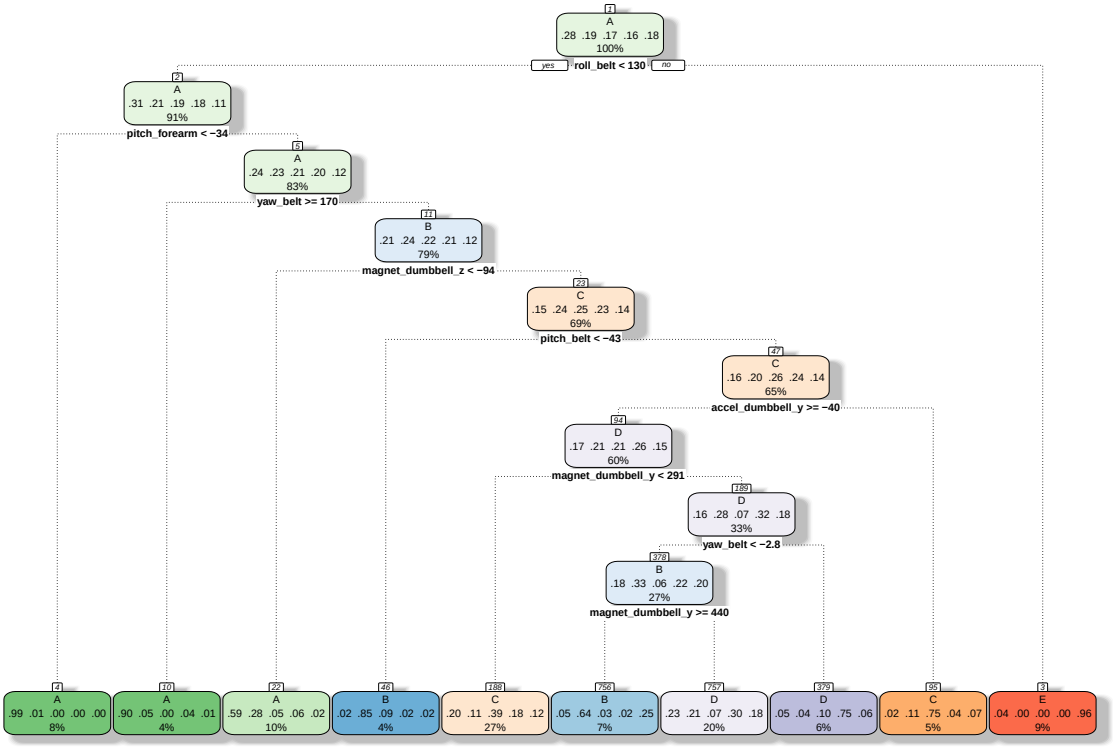
We also want to conduct 3-fold cross validation. Therefore we set up a control for training:

```
crl <- trainControl(method = "cv", number = 3, verboseIter = FALSE)
```

3.1 Decision Tree

Model:

```
tree_mod <- train(classe~., data = train, method = "rpart", trControl = crl, tuneLength = 5)
fancyRpartPlot(tree_mod$finalModel)
```



Rattle 2021−Jun−16 18:06:01 Damster

Prediction:

```
tree_pred <- predict(tree_mod, test)
cmtrees <- confusionMatrix(tree_pred, factor(test$classe))
cmtrees
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 979 160  25  48  11
##          B  30 499  47  15 126
##          C 322 215 851 286 218
##          D 315 265 103 615 222
##          E  28   0   0   0 505
##
## Overall Statistics
##
##                Accuracy : 0.5861
##                  95% CI : (0.5734, 0.5987)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4841
##
##  Mcnemar's Test P-Value : < 2.2e-16
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.5848  0.43810   0.8294   0.6380  0.46673
## Specificity            0.9421  0.95407   0.7858   0.8161  0.99417
## Pos Pred Value         0.8005  0.69596   0.4498   0.4046  0.94747
## Neg Pred Value         0.8509  0.87616   0.9562   0.9200  0.89219
## Prevalence             0.2845  0.19354   0.1743   0.1638  0.18386
## Detection Rate         0.1664  0.08479   0.1446   0.1045  0.08581
## Detection Prevalence   0.2078  0.12184   0.3215   0.2583  0.09057
## Balanced Accuracy      0.7634  0.69609   0.8076   0.7270  0.73045
```

3.2 Random Forest

```
rf_mod <- train(classe~., data=train, method="rf", trControl = crl, tuneLength = 5)
rf_pred <- predict(rf_mod, test)
cmrf <- confusionMatrix(rf_pred, factor(test$classe))
cmrf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1671    6    0    0    0
##          B    2 1128    6    0    0
##          C    0    5 1016   13    0
##          D    0    0    4  951    5
##          E    1    0    0    0 1077
##
## Overall Statistics
##
##                Accuracy : 0.9929
##                  95% CI : (0.9904, 0.9949)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.991
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9903   0.9903   0.9865   0.9954
## Specificity            0.9986   0.9983   0.9963   0.9982   0.9998
## Pos Pred Value         0.9964   0.9930   0.9826   0.9906   0.9991
## Neg Pred Value         0.9993   0.9977   0.9979   0.9974   0.9990
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2839   0.1917   0.1726   0.1616   0.1830
```

```
## Detection Prevalence   0.2850   0.1930   0.1757   0.1631   0.1832
## Balanced Accuracy      0.9984   0.9943   0.9933   0.9923   0.9976
```

3.3 Gradient Boosted Trees

```
gbm_mod <- train(classe~., data=train, method="gbm", trControl = crl, tuneLength = 5, verbose = F)
gbm_pred <- predict(gbm_mod, test)
cmgbm <- confusionMatrix(gbm_pred, factor(test$classe))
cmgbm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1667    8    0    0    0
##          B    6 1122   10    0    1
##          C    1    9 1009   15    2
##          D    0    0    7  945    7
##          E    0    0    0    4 1072
##
## 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            0.9958   0.9851   0.9834   0.9803   0.9908
## Specificity            0.9981   0.9964   0.9944   0.9972   0.9992
## Pos Pred Value         0.9952   0.9851   0.9739   0.9854   0.9963
## Neg Pred Value         0.9983   0.9964   0.9965   0.9961   0.9979
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2833   0.1907   0.1715   0.1606   0.1822
## Detection Prevalence   0.2846   0.1935   0.1760   0.1630   0.1828
## Balanced Accuracy      0.9970   0.9907   0.9889   0.9887   0.9950
```

3.4 Support Vector Machine

```
svm_mod <- train(classe~., data=train, method="svmLinear", trControl = crl, tuneLength = 5, verbose = F)
svm_pred <- predict(svm_mod, test)
cmsvm <- confusionMatrix(svm_pred, factor(test$classe))
cmsvm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1516  136   80   61   47
##          B   36  824   95   38  158
##          C   52   72  796  118   70
##          D   59   22   32  710   64
##          E   11   85   23   37  743
##
## Overall Statistics
##
##                Accuracy : 0.7798
##                  95% CI : (0.769, 0.7903)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7204
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9056   0.7234   0.7758   0.7365   0.6867
## Specificity            0.9231   0.9311   0.9358   0.9640   0.9675
## Pos Pred Value         0.8239   0.7159   0.7184   0.8005   0.8265
## Neg Pred Value         0.9609   0.9335   0.9519   0.9492   0.9320
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2576   0.1400   0.1353   0.1206   0.1263
## Detection Prevalence   0.3127   0.1956   0.1883   0.1507   0.1528
## Balanced Accuracy      0.9143   0.8273   0.8558   0.8503   0.8271
```

3.5 Model comparison

```
mods <- c("Tree", "RF", "GBM", "SVM")
acc <- round(c(cmtrees$overall[1], cmrf$overall[1], cmgbm$overall[1], cmsvm$overall[1]),3)
oos_err <- 1 - acc
data.frame(accuracy = acc, oos_error = oos_err, row.names = mods)
```

Accuracy and Out of Sample Error

```
##      accuracy oos_error
## Tree    0.586     0.414
## RF      0.993     0.007
## GBM     0.988     0.012
## SVM     0.780     0.220
```

Conclusion: The analysis shows that the best model is the Random Forest model. It has an accuracy of 0.9928632 **and an out of sample error rate of** 0.0071368**.**

4. Predictions on Test Set (with Random Forest model)

Here we predict the classe (5 levels) outcome for 20 cases:

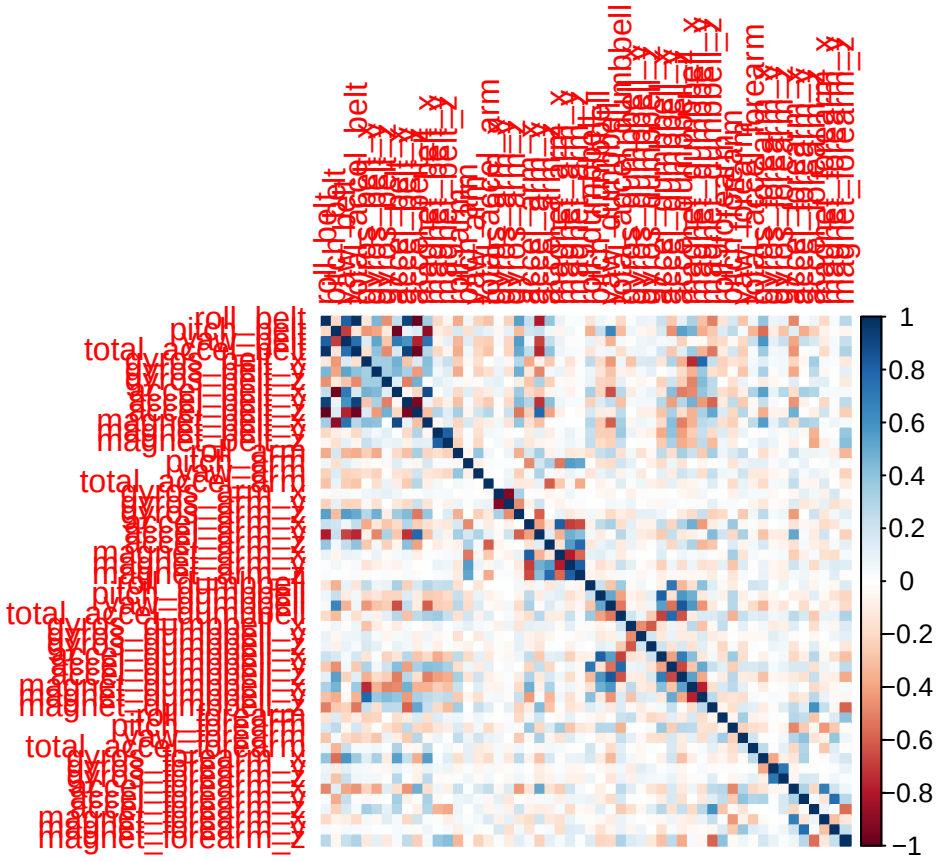
```
pred <- predict(rf_mod, test_dta)
print(pred)
```

```
##  [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
```

5. Appendix

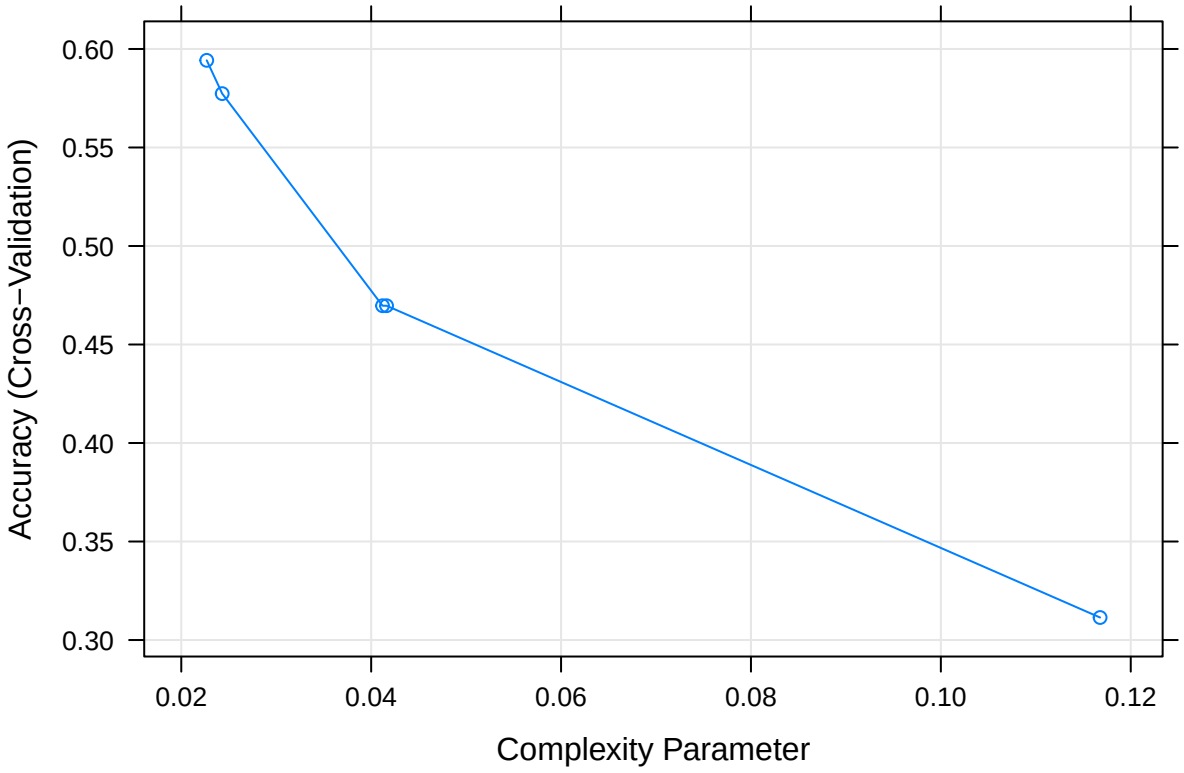
5.1 Training set variables' correlation matrix

```
corrPlot <- cor(train[, -length(names(train))])
corrplot(corrPlot, method="color")
```

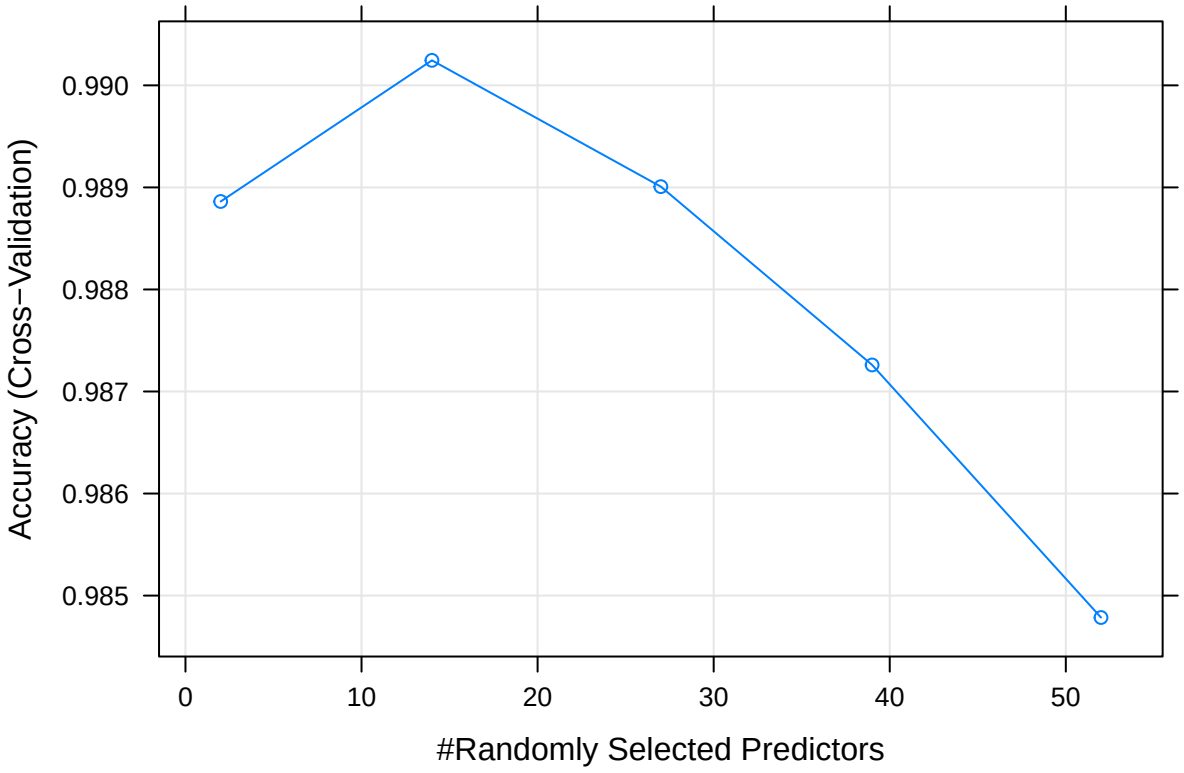


5.2 Plotting the models

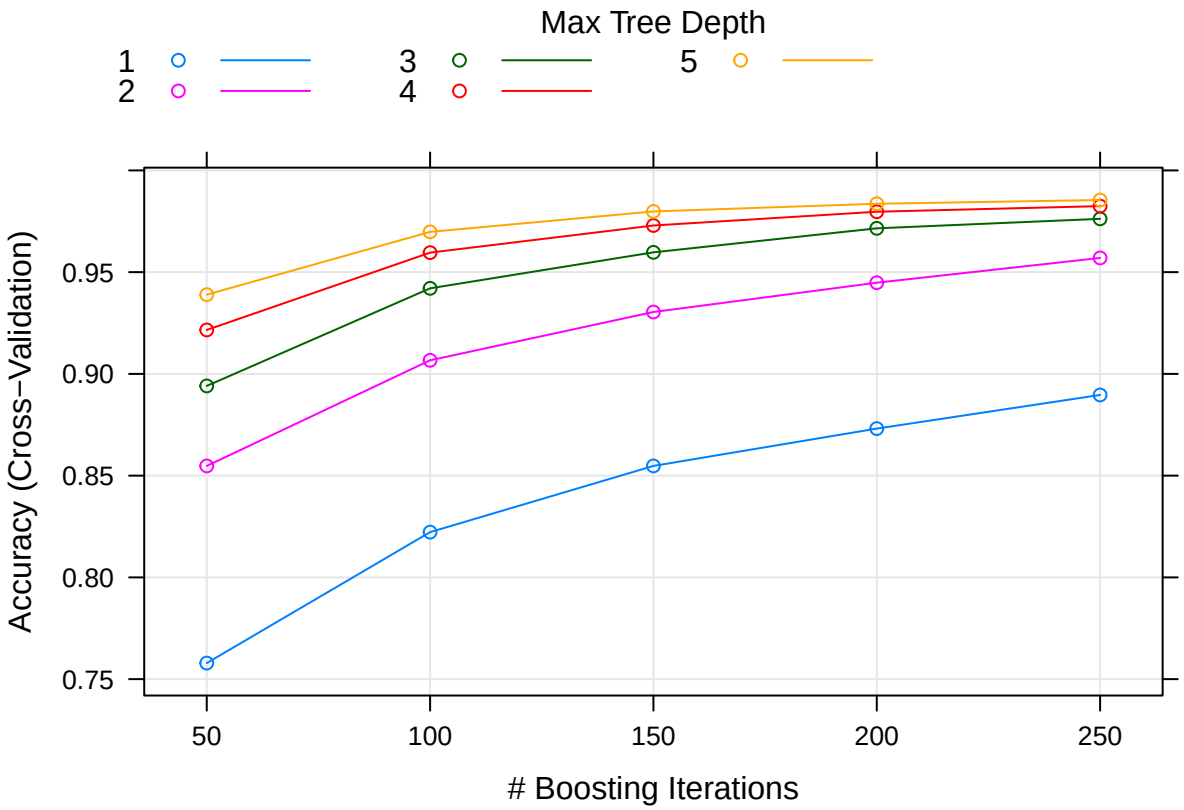
```
plot(tree_mod)
```


Accuracy (Cross−Validation)
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.02
0.04
0.06
0.08
0.10
0.12
Complexity Parameter

```
plot(rf_mod)
```


Accuracy (Cross−Validation)
0.990
0.989
0.988
0.987
0.986
0.985
0
10
20
30
40
50
#Randomly Selected Predictors

```
plot(gbm_mod)
```


Max Tree Depth
1
2
3
4
5
Accuracy (Cross−Validation)
0.75
0.80
0.85
0.90
0.95
50
100
150
200
250
Boosting Iterations