Coursework-2.R

Susana

2020-04-01

library(boot)  
library(gbm)

library(ggplot2)  
library(mgcv)

library(e1071)

library(randomForest)

#  
#Dataset Preparation  
math = read.csv("C:\\Users\\Susana\\Documents\\Universidad\\Machine Learning\\Coursework\\student-mat.csv", sep=";", header=TRUE)  
port = read.csv("C:\\Users\\Susana\\Documents\\Universidad\\Machine Learning\\Coursework\\student-por.csv", sep=";", header=TRUE)  
#Searching for nulls  
sum(is.na(math),is.na(port))

## [1] 0

#Combining both datasets  
math = math[,c(1:30,33)]  
port = port[,c(1:30,33)]  
category = rep("math", nrow(math))  
math\_new = data.frame(category, math)  
math\_new[1:5,1:10]

## category school sex age address famsize Pstatus Medu Fedu Mjob  
## 1 math GP F 18 U GT3 A 4 4 at\_home  
## 2 math GP F 17 U GT3 T 1 1 at\_home  
## 3 math GP F 15 U LE3 T 1 1 at\_home  
## 4 math GP F 15 U GT3 T 4 2 health  
## 5 math GP F 16 U GT3 T 3 3 other

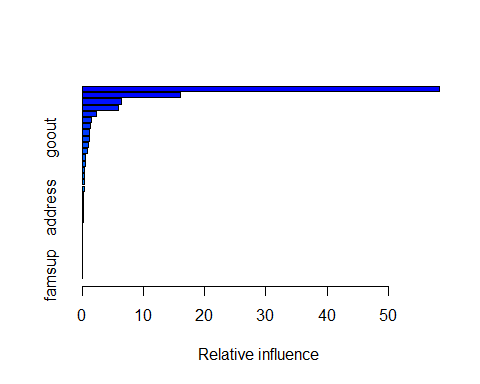
category = rep("port", nrow(port))  
port\_new = data.frame(category, port)  
port\_new[1:5,1:10]

## category school sex age address famsize Pstatus Medu Fedu Mjob  
## 1 port GP F 18 U GT3 A 4 4 at\_home  
## 2 port GP F 17 U GT3 T 1 1 at\_home  
## 3 port GP F 15 U LE3 T 1 1 at\_home  
## 4 port GP F 15 U GT3 T 4 2 health  
## 5 port GP F 16 U GT3 T 3 3 other

mp = rbind(math\_new, port\_new)  
summary(mp)

## category school sex age address famsize Pstatus  
## math:395 GP:772 F:591 Min. :15.00 R:285 GT3:738 A:121   
## port:649 MS:272 M:453 1st Qu.:16.00 U:759 LE3:306 T:923   
## Median :17.00   
## Mean :16.73   
## 3rd Qu.:18.00   
## Max. :22.00   
## Medu Fedu Mjob Fjob reason   
## Min. :0.000 Min. :0.000 at\_home :194 at\_home : 62 course :430   
## 1st Qu.:2.000 1st Qu.:1.000 health : 82 health : 41 home :258   
## Median :3.000 Median :2.000 other :399 other :584 other :108   
## Mean :2.603 Mean :2.388 services:239 services:292 reputation:248   
## 3rd Qu.:4.000 3rd Qu.:3.000 teacher :130 teacher : 65   
## Max. :4.000 Max. :4.000   
## guardian traveltime studytime failures schoolsup  
## father:243 Min. :1.000 Min. :1.00 Min. :0.0000 no :925   
## mother:728 1st Qu.:1.000 1st Qu.:1.00 1st Qu.:0.0000 yes:119   
## other : 73 Median :1.000 Median :2.00 Median :0.0000   
## Mean :1.523 Mean :1.97 Mean :0.2644   
## 3rd Qu.:2.000 3rd Qu.:2.00 3rd Qu.:0.0000   
## Max. :4.000 Max. :4.00 Max. :3.0000   
## famsup paid activities nursery higher internet romantic   
## no :404 no :824 no :528 no :209 no : 89 no :217 no :673   
## yes:640 yes:220 yes:516 yes:835 yes:955 yes:827 yes:371   
##   
##   
##   
##   
## famrel freetime goout Dalc   
## Min. :1.000 Min. :1.000 Min. :1.000 Min. :1.000   
## 1st Qu.:4.000 1st Qu.:3.000 1st Qu.:2.000 1st Qu.:1.000   
## Median :4.000 Median :3.000 Median :3.000 Median :1.000   
## Mean :3.936 Mean :3.201 Mean :3.156 Mean :1.494   
## 3rd Qu.:5.000 3rd Qu.:4.000 3rd Qu.:4.000 3rd Qu.:2.000   
## Max. :5.000 Max. :5.000 Max. :5.000 Max. :5.000   
## Walc health absences G3   
## Min. :1.000 Min. :1.000 Min. : 0.000 Min. : 0.00   
## 1st Qu.:1.000 1st Qu.:3.000 1st Qu.: 0.000 1st Qu.:10.00   
## Median :2.000 Median :4.000 Median : 2.000 Median :11.00   
## Mean :2.284 Mean :3.543 Mean : 4.435 Mean :11.34   
## 3rd Qu.:3.000 3rd Qu.:5.000 3rd Qu.: 6.000 3rd Qu.:14.00   
## Max. :5.000 Max. :5.000 Max. :75.000 Max. :20.00

#Dividing in two datasets for training and testing  
#combined dataset  
set.seed(0)  
sampling = sample(nrow(mp), (nrow(mp)/3)\*2)  
mp.train = mp[sampling,]  
mp.test = mp[-sampling, 1:31]  
mp.test.G3 = mp[-sampling, 32]  
#math  
set.seed(0)  
sampling.math = sample(nrow(math), (nrow(math)/3)\*2)  
math.train = math[sampling.math,]  
math.test = math[-sampling.math, 1:30]  
math.test.G3 = math[-sampling.math, 31]  
#portuguese  
set.seed(0)  
sampling.port = sample(nrow(port), (nrow(port)/3)\*2)  
port.train = port[sampling.port,]  
port.test = port[-sampling.port, 1:30]  
port.test.G3 = port[-sampling.port, 31]  
#  
#Tree-Based with Boosting  
#combined  
boost = gbm(G3~., data=mp.train, distribution="gaussian", shrinkage=0.00000001, n.trees=5000, interaction.depth=4)  
summary(boost)



## var rel.inf  
## failures failures 58.385841538  
## absences absences 16.003173036  
## category category 6.433207015  
## Mjob Mjob 5.916216285  
## Medu Medu 2.247337156  
## studytime studytime 1.582325021  
## higher higher 1.334378080  
## schoolsup schoolsup 1.225245329  
## goout goout 1.213804204  
## reason reason 1.053339058  
## Dalc Dalc 0.821906958  
## traveltime traveltime 0.552159253  
## Fedu Fedu 0.490565299  
## internet internet 0.403243231  
## Fjob Fjob 0.401701109  
## school school 0.369882305  
## health health 0.303138377  
## romantic romantic 0.244525443  
## Walc Walc 0.234826487  
## address address 0.165859199  
## freetime freetime 0.165224409  
## guardian guardian 0.143532659  
## sex sex 0.075633149  
## famrel famrel 0.072466716  
## age age 0.034617655  
## paid paid 0.033958812  
## Pstatus Pstatus 0.025470662  
## nursery nursery 0.024005540  
## famsize famsize 0.023695478  
## activities activities 0.016891893  
## famsup famsup 0.001828642

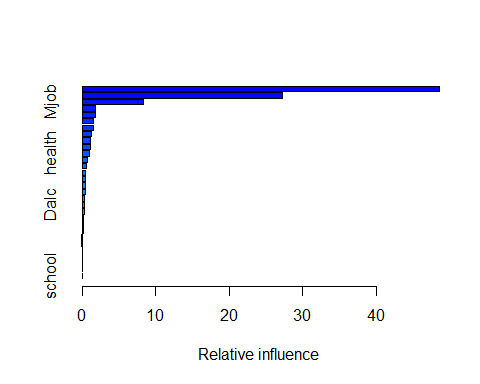
boost.pred = predict(boost, newdata=mp.test, n.trees=5000)  
mean.tree = mean((boost.pred - mp.test.G3)^2)  
mean.tree

## [1] 15.87832

tree.acc = mean(round(boost.pred) == mp.test.G3)\*100  
tree.acc

## [1] 12.93103

#math  
boost.math = gbm(G3~., data=math.train, distribution="gaussian", shrinkage=0.00000001, n.trees=5000, interaction.depth=4)  
summary(boost.math)



## var rel.inf  
## failures failures 48.579292813  
## absences absences 27.191356284  
## Mjob Mjob 8.361376961  
## reason reason 1.785967194  
## schoolsup schoolsup 1.766088944  
## studytime studytime 1.567971559  
## Fjob Fjob 1.564275821  
## Medu Medu 1.191565973  
## sex sex 1.174931387  
## goout goout 1.158274169  
## health health 0.933041892  
## freetime freetime 0.766573640  
## age age 0.523241727  
## romantic romantic 0.495677477  
## Fedu Fedu 0.461326818  
## paid paid 0.407893565  
## Walc Walc 0.368187627  
## famsup famsup 0.320552704  
## Dalc Dalc 0.271599943  
## traveltime traveltime 0.234146054  
## guardian guardian 0.221780056  
## famrel famrel 0.213970179  
## activities activities 0.149015176  
## Pstatus Pstatus 0.089018217  
## famsize famsize 0.080600732  
## address address 0.055681118  
## internet internet 0.030609438  
## nursery nursery 0.027598557  
## higher higher 0.004985005  
## school school 0.003398971

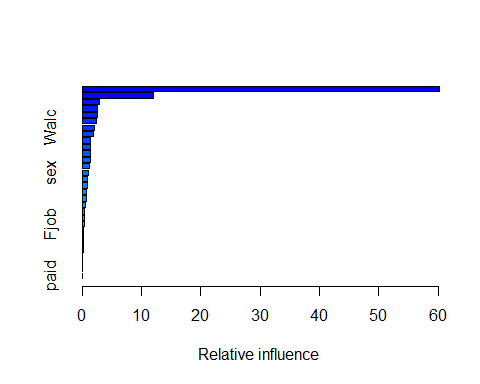
boost.pred.m = predict(boost.math, newdata=math.test, n.trees=5000)  
mean.tree.m = mean((boost.pred.m - math.test.G3)^2)  
mean.tree.m

## [1] 20.81799

tree.acc.m = mean(round(boost.pred.m) == math.test.G3)\*100  
tree.acc.m

## [1] 12.87879

#portuguese  
boost.port = gbm(G3~., data=port.train, distribution="gaussian", shrinkage=0.00000001, n.trees=5000, interaction.depth=4)  
summary(boost.port)



## var rel.inf  
## failures failures 60.251370472  
## school school 11.938365852  
## higher higher 2.871042003  
## reason reason 2.627093026  
## absences absences 2.599857142  
## studytime studytime 2.335508439  
## Walc Walc 2.054707910  
## activities activities 1.879242447  
## Dalc Dalc 1.415020908  
## Medu Medu 1.381587392  
## Fedu Fedu 1.369660878  
## Mjob Mjob 1.302796299  
## health health 1.146949823  
## sex sex 1.110019711  
## romantic romantic 0.889274621  
## address address 0.858414684  
## age age 0.756730617  
## goout goout 0.649788139  
## freetime freetime 0.541972611  
## famrel famrel 0.451504015  
## schoolsup schoolsup 0.444220139  
## Fjob Fjob 0.313547837  
## guardian guardian 0.212274186  
## famsup famsup 0.188640916  
## traveltime traveltime 0.164740742  
## internet internet 0.156219846  
## famsize famsize 0.059864530  
## nursery nursery 0.022781820  
## Pstatus Pstatus 0.006802993  
## paid paid 0.000000000

boost.pred.p = predict(boost.port, newdata=port.test, n.trees=5000)  
mean.tree.p = mean((boost.pred.p - port.test.G3)^2)  
mean.tree.p

## [1] 9.006527

tree.acc.p = mean(round(boost.pred.p) == port.test.G3)\*100  
tree.acc.p

## [1] 11.98157

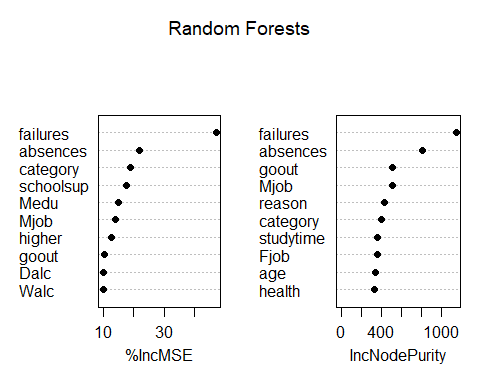
#  
#Random Forests  
#combined  
set.seed(5)  
forests = randomForest(G3~., data=mp.train, importance=TRUE, ntree=1000)  
forests.pred = predict(forests, newdata=mp.test)  
forests

##   
## Call:  
## randomForest(formula = G3 ~ ., data = mp.train, importance = TRUE, ntree = 1000)   
## Type of random forest: regression  
## Number of trees: 1000  
## No. of variables tried at each split: 10  
##   
## Mean of squared residuals: 10.08755  
## % Var explained: 30.18

importance(forests)

## %IncMSE IncNodePurity  
## category 19.0410993 403.99768  
## school 7.0544944 156.34103  
## sex 8.1988661 134.41941  
## age 7.5015457 336.87710  
## address 4.0155022 127.06946  
## famsize 0.3856728 122.01057  
## Pstatus 3.8636274 92.02906  
## Medu 14.9757233 329.25212  
## Fedu 7.6648269 311.06737  
## Mjob 13.8650433 507.80021  
## Fjob 6.9424549 359.28866  
## reason 9.7041888 434.21961  
## guardian 3.8351109 204.98021  
## traveltime 7.6122694 231.86187  
## studytime 9.0754389 365.85321  
## failures 46.9767123 1149.44905  
## schoolsup 17.5254167 158.26323  
## famsup 1.5506564 116.09520  
## paid 2.9059404 113.60972  
## activities 3.2720361 125.21670  
## nursery 3.5810929 122.55904  
## higher 12.5904422 183.60264  
## internet 5.6546457 128.44224  
## romantic 2.9903156 140.14535  
## famrel 1.9794525 258.44704  
## freetime 7.3733824 322.16374  
## goout 10.5331087 510.54813  
## Dalc 10.1257142 269.46746  
## Walc 10.0360761 307.61420  
## health 6.9556044 334.78013  
## absences 21.7635267 812.88287

varImpPlot(forests, n.var=10, main="Random Forests", pch=16)



mean.randomf = mean((forests.pred-mp.test.G3)^2)  
mean.randomf

## [1] 10.54378

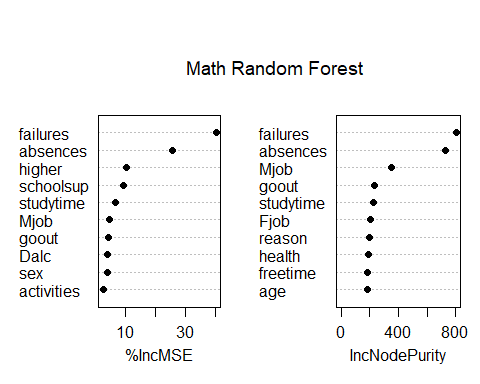
f.acc = mean(round(forests.pred) == mp.test.G3)\*100  
f.acc

## [1] 15.51724

#math  
set.seed(5)  
forests.m = randomForest(G3~., data=math.train, importance=TRUE, ntree=1000)  
forests.pred.m = predict(forests.m, newdata=math.test)  
importance(forests.m)

## %IncMSE IncNodePurity  
## school -1.28025903 28.11627  
## sex 4.05722016 84.23683  
## age 0.43410388 179.55881  
## address -0.39977214 48.79901  
## famsize 1.09739079 54.15479  
## Pstatus 0.29714647 37.65706  
## Medu 1.55179991 176.53165  
## Fedu 0.31474228 136.70442  
## Mjob 4.85798440 347.99556  
## Fjob -2.53562291 204.53775  
## reason -1.00981112 200.31743  
## guardian 1.49609033 113.52147  
## traveltime 1.54428399 132.86368  
## studytime 6.64100467 225.21631  
## failures 40.11308404 804.53159  
## schoolsup 9.31796313 96.98781  
## famsup 0.57347073 60.71335  
## paid -0.03656218 92.65242  
## activities 2.75640201 72.13015  
## nursery 0.79811125 54.98943  
## higher 10.47108230 109.88797  
## internet -0.66980628 38.15788  
## romantic -0.80651323 76.13240  
## famrel -2.23339811 147.96838  
## freetime 1.27184641 180.40319  
## goout 4.33503744 228.98241  
## Dalc 4.09917566 85.95645  
## Walc 0.94694594 165.45269  
## health 2.14192057 190.50503  
## absences 25.79820796 727.56856

varImpPlot(forests.m, n.var=10, main="", pch=16)  
mtext("Math Random Forest", cex=1.2, adj = -0.1)



mean.randomf.m = mean((forests.pred.m-math.test.G3)^2)  
mean.randomf.m

## [1] 16.05885

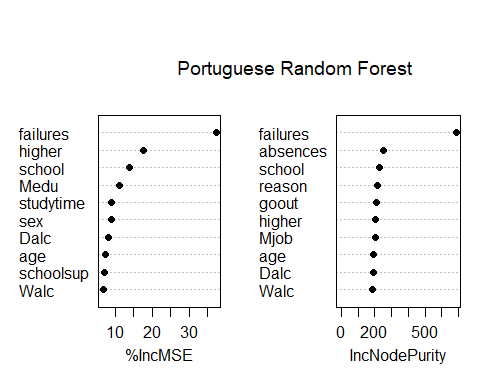
f.acc.m = mean(round(forests.pred.m) == math.test.G3)\*100  
f.acc.m

## [1] 8.333333

#portuguese  
set.seed(5)  
forests.p = randomForest(G3~., data=port.train, importance=TRUE, ntree=1000)  
forests.pred.p = predict(forests.p, newdata=port.test)  
importance(forests.p)

## %IncMSE IncNodePurity  
## school 13.91841292 229.68022  
## sex 8.89519920 79.25889  
## age 7.42432023 194.17787  
## address 3.49866413 64.94538  
## famsize -0.43475227 44.95215  
## Pstatus -0.42625081 41.01642  
## Medu 11.13619800 182.11241  
## Fedu 5.52803777 143.89658  
## Mjob 3.59316797 202.32970  
## Fjob 0.86699632 137.94017  
## reason 5.31186755 216.35557  
## guardian -0.09639722 64.35964  
## traveltime 4.97071754 91.59768  
## studytime 8.97893014 160.14365  
## failures 37.41444925 685.62245  
## schoolsup 6.92716647 78.10951  
## famsup -1.91987085 52.61984  
## paid 1.00734037 22.35544  
## activities 3.40864453 63.17356  
## nursery 1.42496460 39.36682  
## higher 17.56518182 202.95717  
## internet 0.79474395 54.02915  
## romantic 0.23212114 62.70335  
## famrel 1.72255628 136.78254  
## freetime 2.15565477 177.27290  
## goout 2.50510025 211.66464  
## Dalc 8.25354203 193.14789  
## Walc 6.71082987 184.55267  
## health 0.59065298 141.06974  
## absences 4.38236802 252.99056

varImpPlot(forests.p, n.var=10, pch=16, main="")  
mtext("Portuguese Random Forest", cex=1.2, adj = -1.25)



mean.randomf.p = mean((forests.pred.p-port.test.G3)^2)  
mean.randomf.p

## [1] 6.165701

f.acc.p = mean(round(forests.pred.p) == port.test.G3)\*100  
f.acc.p

## [1] 13.36406

#  
#SVM  
#Support Vector Machine  
#combined linear  
set.seed(6)  
tune.out = tune(svm, G3~., data=mp.train, kernel="linear", cost=1)  
best.linear = tune.out$best.model  
ypred.linear = predict(best.linear, mp.test)  
mean.svml = mean((ypred.linear-mp.test.G3)^2)  
mean.svml

## [1] 12.50918

svm.linear.acc = mean(round(ypred.linear) == mp.test.G3)\*100  
svm.linear.acc

## [1] 13.50575

#math linear  
set.seed(6)  
tune.out.m = tune(svm, G3~., data=math.train, kernel="linear", cost=1)  
best.linear.m = tune.out.m$best.model  
ypred.linear.m = predict(best.linear.m, math.test)  
mean.svml.m = mean((ypred.linear.m-math.test.G3)^2)  
mean.svml.m

## [1] 21.71095

svm.linear.acc.m = mean(round(ypred.linear.m) == math.test.G3)\*100  
svm.linear.acc.m

## [1] 9.090909

#portuguese linear  
set.seed(6)  
tune.out.p = tune(svm, G3~., data=port.train, kernel="linear", cost=1)  
best.linear.p = tune.out.p$best.model  
ypred.linear.p = predict(best.linear.p, port.test)  
mean.svml.p = mean((ypred.linear.p-port.test.G3)^2)  
mean.svml.p

## [1] 6.647034

svm.linear.acc.p = mean(round(ypred.linear.p) == port.test.G3)\*100  
svm.linear.acc.p

## [1] 13.82488

#combined polynomial of 2nd degree  
set.seed(7)  
tune.out.p = tune(svm, G3~., data=mp.train, kernel="polynomial", cost=1, degree=2)  
ypred.p = predict(tune.out.p$best.model, mp.test)  
svm.p.acc = mean(round(ypred.p) == mp.test.G3)\*100  
svm.p.acc

## [1] 18.96552

mean.svmp = mean((ypred.p-mp.test.G3)^2)  
mean.svmp

## [1] 12.65869

#math polynomial of 2nd degree  
set.seed(6)  
tune.out.pm = tune(svm, G3~., data=math.train, kernel="polynomial", degree=2, cost=1)  
ypred.linear.pm = predict(tune.out.pm$best.model, math.test)  
mean.svml.pm = mean((ypred.linear.pm-math.test.G3)^2)  
mean.svml.pm

## [1] 18.46723

svm.acc.pm = mean(round(ypred.linear.pm) == math.test.G3)\*100  
svm.acc.pm

## [1] 12.12121

#portuguese polynomial of 2nd degree  
set.seed(6)  
tune.out.pp = tune(svm, G3~., data=port.train, kernel="linear", cost=1, degree=2)  
ypred.linear.pp = predict(tune.out.pp$best.model, port.test)  
mean.svml.pp = mean((ypred.linear.pp-port.test.G3)^2)  
mean.svml.pp

## [1] 6.647034

svm.acc.pp = mean(round(ypred.linear.pp) == port.test.G3)\*100  
svm.acc.pp

## [1] 13.82488

#combined polynomial of 3rd degree  
set.seed(8)  
tune.out.p3 = tune(svm, G3~., data=mp.train, kernel="polynomial", cost=1, degree=3)  
ypred.p3 = predict(tune.out.p3$best.model, mp.test)  
svm.p.acc3 = mean(round(ypred.p3) == mp.test.G3)\*100  
svm.p.acc3

## [1] 17.24138

mean.svmp3 = mean((ypred.p3-mp.test.G3)^2)  
mean.svmp3

## [1] 13.03258

#math polynomial of 3rd degree  
set.seed(6)  
tune.out.p3m = tune(svm, G3~., data=math.train, kernel="polynomial", degree=3, cost=1)  
ypred.linear.p3m = predict(tune.out.p3m$best.model, math.test)  
mean.svml.p3m = mean((ypred.linear.p3m-math.test.G3)^2)  
mean.svml.p3m

## [1] 19.18398

svm.acc.p3m = mean(round(ypred.linear.p3m) == math.test.G3)\*100  
svm.acc.p3m

## [1] 10.60606

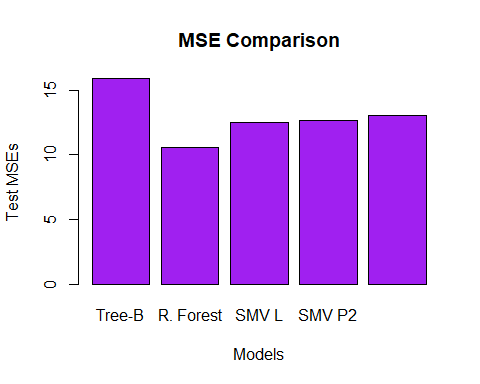
#portuguese polynomial of 3rd degree  
set.seed(6)  
tune.out.p3p = tune(svm, G3~., data=port.train, kernel="linear", cost=1, degree=3)  
ypred.linear.p3p = predict(tune.out.p3p$best.model, port.test)  
mean.svml.p3p = mean((ypred.linear.p3p-port.test.G3)^2)  
mean.svml.p3p

## [1] 6.647034

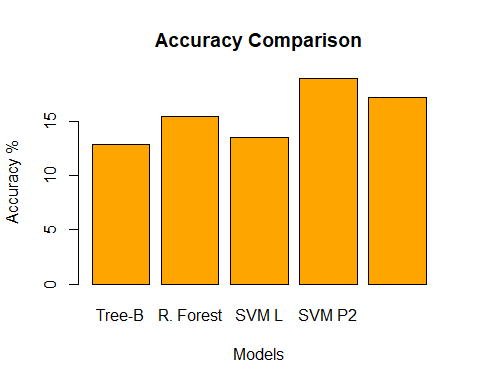
svm.acc.p3p = mean(round(ypred.linear.p3p) == port.test.G3)\*100  
svm.acc.p3p

## [1] 13.82488

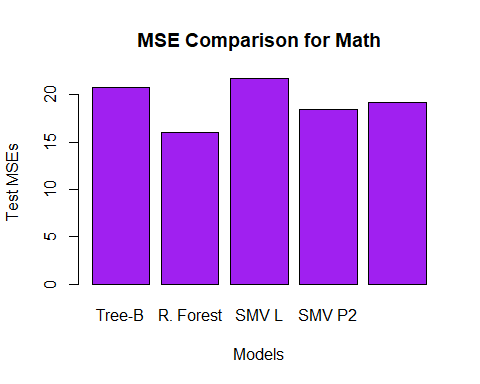
#  
#Models Comparison  
#combined  
barplot(c(mean.tree, mean.randomf, mean.svml, mean.svmp, mean.svmp3),  
 xlab="Models",  
 ylab="Test MSEs",  
 main="MSE Comparison",  
 names.arg=c("Tree-B","R. Forest", "SMV L", "SMV P2", "SMV P3"),  
 col="purple")



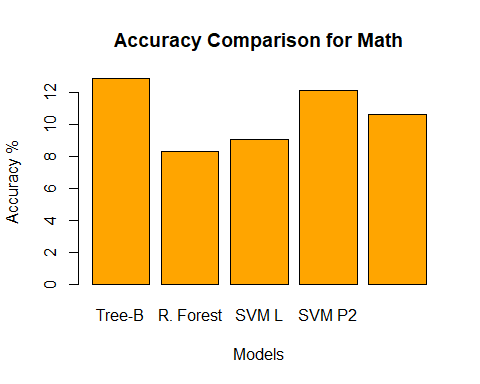
barplot(c(tree.acc, f.acc, svm.linear.acc, svm.p.acc, svm.p.acc3),  
 xlab="Models",  
 ylab="Accuracy %",  
 main="Accuracy Comparison",  
 names.arg=c("Tree-B","R. Forest", "SVM L", "SVM P2", "SVM P3"),  
 col="orange")



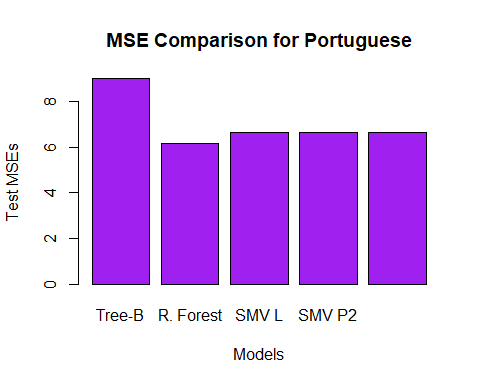
#math  
barplot(c(mean.tree.m, mean.randomf.m, mean.svml.m, mean.svml.pm, mean.svml.p3m),  
 xlab="Models",  
 ylab="Test MSEs",  
 main="MSE Comparison for Math",  
 names.arg=c("Tree-B","R. Forest", "SMV L", "SMV P2", "SMV P3"),  
 col="purple")



barplot(c(tree.acc.m, f.acc.m, svm.linear.acc.m, svm.acc.pm, svm.acc.p3m),  
 xlab="Models",  
 ylab="Accuracy %",  
 main="Accuracy Comparison for Math",  
 names.arg=c("Tree-B","R. Forest", "SVM L", "SVM P2", "SVM P3"),  
 col="orange")



#portuguese  
barplot(c(mean.tree.p, mean.randomf.p, mean.svml.p, mean.svml.pp, mean.svml.p3p),  
 xlab="Models",  
 ylab="Test MSEs",  
 main="MSE Comparison for Portuguese",  
 names.arg=c("Tree-B","R. Forest", "SMV L", "SMV P2", "SMV P3"),  
 col="purple")



barplot(c(tree.acc.p, f.acc.p, svm.linear.acc.p, svm.acc.pp, svm.acc.p3p),  
 xlab="Models",  
 ylab="Accuracy %",  
 main="Accuracy Comparison for Portuguese",  
 names.arg=c("Tree-B","R. Forest", "SVM L", "SVM P2", "SVM P3"),  
 col="orange")

