Final Project

Bert

2022-04-14

library(readr)  
library(corrplot)

## corrplot 0.92 loaded

library(ggplot2)  
library(leaps)  
library(glmnet)

## Loading required package: Matrix

## Loaded glmnet 4.1-8

library(tree)  
library(randomForest)

## randomForest 4.7-1.1

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

##   
## Attaching package: 'randomForest'

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

library(gbm)

## Loaded gbm 2.1.8.1

library(reshape2)

#import data  
med = read.csv("C:/Users/bert0/Documents/Bert-school/Columbia University/Spring 2022/PM/insurance.csv", header=TRUE)  
df = read.csv("C:/Users/bert0/Documents/Bert-school/Columbia University/Spring 2022/PM/insurance.csv", header=TRUE)

#check missing data  
which(is.na(med))

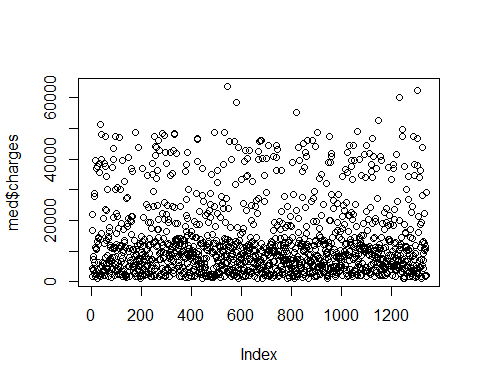
## integer(0)

#change data structure  
med$sex <- as.factor(med$sex)  
med$smoker <- as.factor(med$smoker)  
med$region <- as.factor(med$region)  
  
levels(med$sex) <- c("female", "male")  
levels(med$smoker) <- c("No", "Yes")  
levels(med$region) <- c("northeast", "northwest", "southwest","southeast")

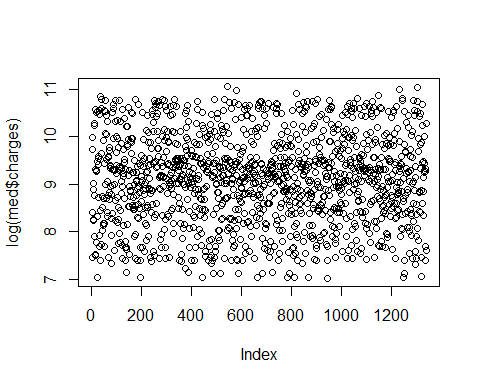
#data visualization  
summary(med)

## age sex bmi children smoker   
## Min. :18.00 female:662 Min. :15.96 Min. :0.000 No :1064   
## 1st Qu.:27.00 male :676 1st Qu.:26.30 1st Qu.:0.000 Yes: 274   
## Median :39.00 Median :30.40 Median :1.000   
## Mean :39.21 Mean :30.66 Mean :1.095   
## 3rd Qu.:51.00 3rd Qu.:34.69 3rd Qu.:2.000   
## Max. :64.00 Max. :53.13 Max. :5.000   
## region charges   
## northeast:324 Min. : 1122   
## northwest:325 1st Qu.: 4740   
## southwest:364 Median : 9382   
## southeast:325 Mean :13270   
## 3rd Qu.:16640   
## Max. :63770

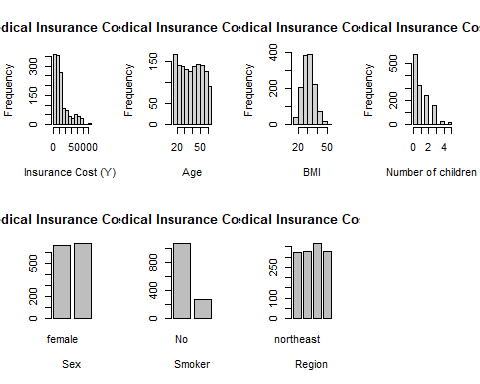
plot(med$charges)



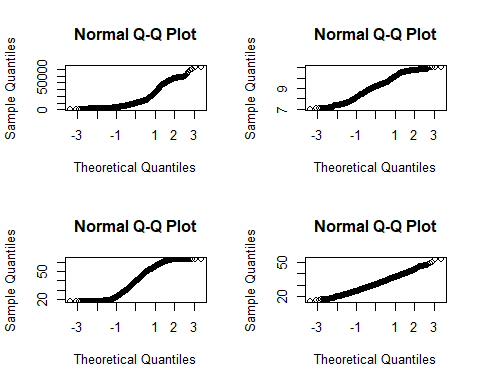
plot(log(med$charges))



par(mfrow = c(2,4))  
#4 numerical data  
hist(med$charges,xlab = "Insurance Cost (Y) ", main="Medical Insurance Cost Data")  
hist(med$age,xlab = "Age", main="Medical Insurance Cost Data")  
hist(med$bmi,xlab = "BMI", main="Medical Insurance Cost Data")  
hist(med$children,xlab = "Number of children", main="Medical Insurance Cost Data")  
#3 categorical data  
barplot(table(med$sex),xlab="Sex",  
main = "Medical Insurance Cost Data")  
barplot(table(med$smoker),xlab="Smoker",  
main = "Medical Insurance Cost Data")  
barplot(table(med$region),xlab="Region",  
main = "Medical Insurance Cost Data")



par(mfrow = c(2,2))  
qqnorm(med$charges)  
qqnorm(log(med$charges))  
qqnorm(med$age)  
qqnorm(med$bmi)



#Charts :1.charges, 2.ln(charge) 3.age 4.bmi

#Change charges into ln()  
med$charges<-log(med$charges)

#Validation Set  
set.seed(1)  
train=sample(c(TRUE,FALSE), nrow(med),rep=TRUE)  
#sum(train==TRUE);sum(train==FALSE)  
test=(!train)

#full regression model   
y=med$charges  
res=lm(y~age+sex+bmi+children+smoker+region,data=med)  
summary(res)

##   
## Call:  
## lm(formula = y ~ age + sex + bmi + children + smoker + region,   
## data = med)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -1.07186 -0.19835 -0.04917 0.06598 2.16636   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 7.0305581 0.0723960 97.112 < 2e-16 \*\*\*  
## age 0.0345816 0.0008721 39.655 < 2e-16 \*\*\*  
## sexmale -0.0754164 0.0244012 -3.091 0.002038 \*\*   
## bmi 0.0133748 0.0020960 6.381 2.42e-10 \*\*\*  
## children 0.1018568 0.0100995 10.085 < 2e-16 \*\*\*  
## smokerYes 1.5543228 0.0302795 51.333 < 2e-16 \*\*\*  
## regionnorthwest -0.0637876 0.0349057 -1.827 0.067860 .   
## regionsouthwest -0.1571967 0.0350828 -4.481 8.08e-06 \*\*\*  
## regionsoutheast -0.1289522 0.0350271 -3.681 0.000241 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.4443 on 1329 degrees of freedom  
## Multiple R-squared: 0.7679, Adjusted R-squared: 0.7666   
## F-statistic: 549.8 on 8 and 1329 DF, p-value: < 2.2e-16

We are going to select models, we use MSE and other values to compare models and following part have three methods to measure MSE: #1.using all data 2.validation set 3.Cross Validation

reg.best=regsubsets(charges~ . ,data=med,nvmax=8)  
summary(reg.best)

## Subset selection object  
## Call: regsubsets.formula(charges ~ ., data = med, nvmax = 8)  
## 8 Variables (and intercept)  
## Forced in Forced out  
## age FALSE FALSE  
## sexmale FALSE FALSE  
## bmi FALSE FALSE  
## children FALSE FALSE  
## smokerYes FALSE FALSE  
## regionnorthwest FALSE FALSE  
## regionsouthwest FALSE FALSE  
## regionsoutheast FALSE FALSE  
## 1 subsets of each size up to 8  
## Selection Algorithm: exhaustive  
## age sexmale bmi children smokerYes regionnorthwest regionsouthwest  
## 1 ( 1 ) " " " " " " " " "\*" " " " "   
## 2 ( 1 ) "\*" " " " " " " "\*" " " " "   
## 3 ( 1 ) "\*" " " " " "\*" "\*" " " " "   
## 4 ( 1 ) "\*" " " "\*" "\*" "\*" " " " "   
## 5 ( 1 ) "\*" " " "\*" "\*" "\*" " " "\*"   
## 6 ( 1 ) "\*" " " "\*" "\*" "\*" " " "\*"   
## 7 ( 1 ) "\*" "\*" "\*" "\*" "\*" " " "\*"   
## 8 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*"   
## regionsoutheast  
## 1 ( 1 ) " "   
## 2 ( 1 ) " "   
## 3 ( 1 ) " "   
## 4 ( 1 ) " "   
## 5 ( 1 ) " "   
## 6 ( 1 ) "\*"   
## 7 ( 1 ) "\*"   
## 8 ( 1 ) "\*"

#using whole data to train  
reg.Med.summary=summary(reg.best)  
reg.Med.summary$outmat

## age sexmale bmi children smokerYes regionnorthwest regionsouthwest  
## 1 ( 1 ) " " " " " " " " "\*" " " " "   
## 2 ( 1 ) "\*" " " " " " " "\*" " " " "   
## 3 ( 1 ) "\*" " " " " "\*" "\*" " " " "   
## 4 ( 1 ) "\*" " " "\*" "\*" "\*" " " " "   
## 5 ( 1 ) "\*" " " "\*" "\*" "\*" " " "\*"   
## 6 ( 1 ) "\*" " " "\*" "\*" "\*" " " "\*"   
## 7 ( 1 ) "\*" "\*" "\*" "\*" "\*" " " "\*"   
## 8 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*"   
## regionsoutheast  
## 1 ( 1 ) " "   
## 2 ( 1 ) " "   
## 3 ( 1 ) " "   
## 4 ( 1 ) " "   
## 5 ( 1 ) " "   
## 6 ( 1 ) "\*"   
## 7 ( 1 ) "\*"   
## 8 ( 1 ) "\*"

r2=reg.Med.summary$rsq  
adjr2=reg.Med.summary$adjr2   
cp=reg.Med.summary$cp  
bic=reg.Med.summary$bic  
table\_best=data.frame(model=c(1:8),r2,adjr2,cp,bic)  
table\_best

## model r2 adjr2 cp bic  
## 1 1 0.4428978 0.4424809 1856.61244 -768.341  
## 2 2 0.7395465 0.7391564 159.65828 -1778.456  
## 3 3 0.7572654 0.7567195 60.17950 -1865.527  
## 4 4 0.7621566 0.7614429 34.16713 -1885.564  
## 5 5 0.7639274 0.7630413 26.02496 -1888.365  
## 6 6 0.7657049 0.7646487 17.84507 -1891.278  
## 7 7 0.7673647 0.7661403 10.33949 -1893.591  
## 8 8 0.7679478 0.7665509 9.00000 -1889.750

#looking for coef  
coef(reg.best,3)

## (Intercept) age children smokerYes   
## 7.28772342 0.03528491 0.10163109 1.54427238

coef(reg.best,4)

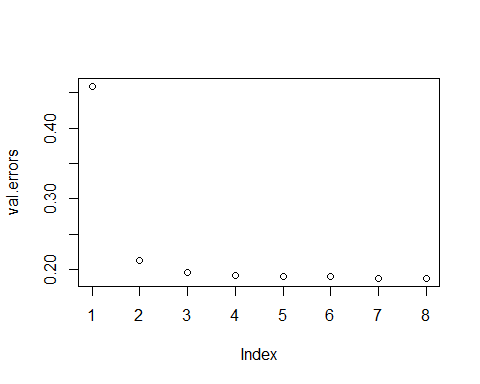
## (Intercept) age bmi children smokerYes   
## 6.98277656 0.03478256 0.01060965 0.10119760 1.54324382

prefer 3 or 4 model

#Validation Set  
regfit.full=regsubsets(charges~.,data=med[train,],nvmax=8)  
#summary(regfit.full)  
test.mat=model.matrix(charges~.,data=med[test,])  
val.errors=rep(NA,8)  
for (i in 1:8)  
{  
 coefi=coef(regfit.full,id=i)  
 pred=test.mat[,names(coefi)]%\*%coefi  
 val.errors[i]=mean((med$charges[test]-pred)^2)  
   
}  
data.frame(model=c(1:8),val.errors)

## model val.errors  
## 1 1 0.4586110  
## 2 2 0.2126700  
## 3 3 0.1962058  
## 4 4 0.1916911  
## 5 5 0.1911752  
## 6 6 0.1905866  
## 7 7 0.1886521  
## 8 8 0.1881355

plot(val.errors)



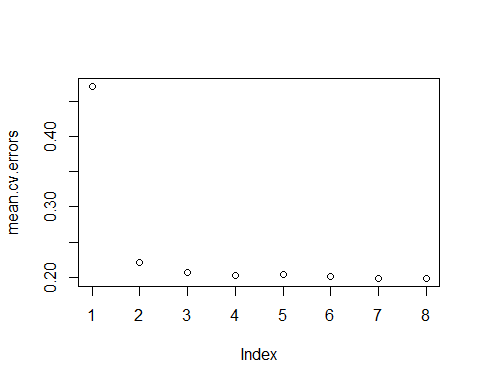
#minimum errors' model  
#coef(regfit.full, which(val.errors==min(val.errors)))

validation test show…

#prediction  
predict.regsubsets=function(object,newdata,id,...)  
{  
form=as.formula(object$call[[2]])  
mat=model.matrix(form,newdata)  
coefi=coef(object,id=id)  
xvars=names(coefi)  
mat[,xvars]%\*%coefi  
}  
#Cross validation  
k=10  
set.seed(1)  
folds=sample(1:k,nrow(med),replace=TRUE)  
#folds  
cv.errors=matrix(NA,k,8, dimnames=list(NULL, paste(1:8)))  
for(j in 1:k){  
 best.fit=regsubsets(charges~.,data=med[folds!=j,],nvmax=8)  
 for(i in 1:8){  
 pred=predict.regsubsets(best.fit,med[folds==j,],id=i)  
 cv.errors[j,i]=mean( (med$charges[folds==j]-pred)^2)  
 }  
}  
mean.cv.errors=apply(cv.errors,2,mean)  
#draw plot  
data.frame(model=c(1:8),mean.cv.errors)

## model mean.cv.errors  
## 1 1 0.4711180  
## 2 2 0.2209773  
## 3 3 0.2061846  
## 4 4 0.2021111  
## 5 5 0.2033770  
## 6 6 0.2013673  
## 7 7 0.1986568  
## 8 8 0.1982912

plot(mean.cv.errors)



#minimum errors' model#coef(regfit.full, which(mean.cv.errors==min(mean.cv.errors)))

CV shows ….

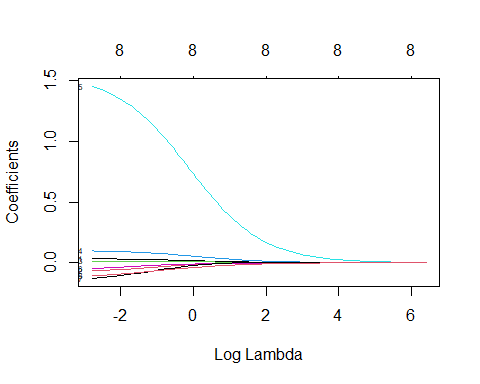
CONCLUSION could be: Although we have model 6 with lowest MSE, we prefer 3 factor model is the best, because it is not complicated and the accuracy is ratively high.

#3 factor model is the best, because it is not complicated and the accuracy is ratively high.  
res2=lm(y~age+children+smoker,data=med)  
summary(res2)

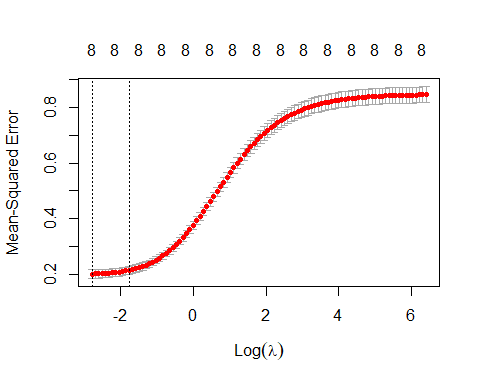
##   
## Call:  
## lm(formula = y ~ age + children + smoker, data = med)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.94939 -0.17632 -0.04368 0.04252 2.13501   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 7.2877234 0.0387040 188.294 <2e-16 \*\*\*  
## age 0.0352849 0.0008839 39.919 <2e-16 \*\*\*  
## children 0.1016311 0.0102990 9.868 <2e-16 \*\*\*  
## smokerYes 1.5442724 0.0307364 50.242 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.4535 on 1334 degrees of freedom  
## Multiple R-squared: 0.7573, Adjusted R-squared: 0.7567   
## F-statistic: 1387 on 3 and 1334 DF, p-value: < 2.2e-16

Also, we can try to do Ridge and Lasso to reduce Variance

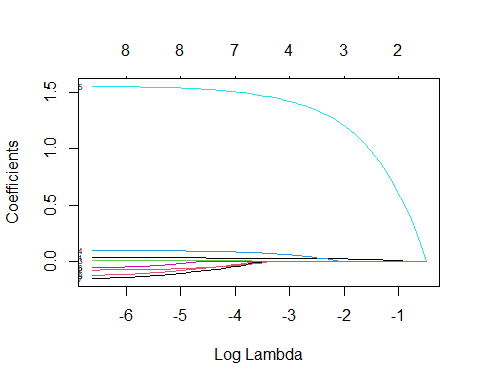
#Ridge & Lasso  
# make design matrix x and response y=charges  
x=model.matrix(charges~.,med)[,-1]  
y=med$charges  
fit.ridge = glmnet(x,y,alpha=0)  
plot(fit.ridge, xvar="lambda", label=TRUE)



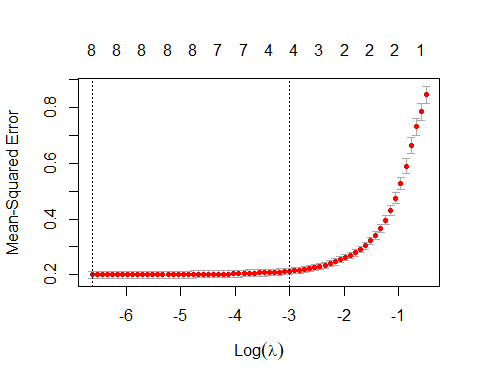
cv.ridge = cv.glmnet(x,y,alpha =0);plot(cv.ridge)



fit.lasso = glmnet(x,y,alpha=1)  
plot(fit.lasso, xvar="lambda", label=TRUE)



cv.lasso = cv.glmnet(x,y,alpha =1);plot(cv.lasso)



We can see which proportion of training set have best performance Both Ridge and Lasso. Table1:Ridge;Table2:Lasso

#decide training set   
set.seed(1)  
tp=rep(0,9)  
mse=rep(0,9)  
a=c(0.5,0.5625,0.625,0.6875,0.75,0.8125,0.875,0.9375,1)  
for (i in 1:9) {  
 if (i<1) {  
 train=sample(1:nrow(x), a[i]\*nrow(x))  
 test=(-train)  
 y.test = y[-train]  
 alpha0.fit =cv.glmnet(x[train,],y[train], alpha=0, type.measure="mse",family="gaussian")  
 alpha0.predicted = predict(alpha0.fit, s=alpha0.fit$lambda.1se,newx=x[-train,])  
 tp[i]=alpha0.fit$lambda.1se  
 mse[i]=mean((y.test-alpha0.predicted)^2)  
 }   
 else {  
 train=sample(1:nrow(x), a[i]\*nrow(x))  
 test=(train)  
 y.test = y[train]  
 alpha0.fit =cv.glmnet(x[train,],y[train], alpha=0, type.measure="mse",family="gaussian")  
 alpha0.predicted = predict(alpha0.fit, s=alpha0.fit$lambda.1se,newx=x[train,])  
 tp[i]=alpha0.fit$lambda.1se  
 mse[i]=mean((y.test-alpha0.predicted)^2)  
 }  
   
}  
  
table\_Ridge=data.frame(RatioOfTrainingSet = a, tp=tp,MSE=mse)  
table\_Ridge

## RatioOfTrainingSet tp MSE  
## 1 0.5000 0.1867904 0.2029582  
## 2 0.5625 0.2037921 0.2294636  
## 3 0.6250 0.1635435 0.1942499  
## 4 0.6875 0.1896704 0.2163271  
## 5 0.7500 0.1725343 0.2206320  
## 6 0.8125 0.1875701 0.2154752  
## 7 0.8750 0.1429752 0.2069787  
## 8 0.9375 0.1517741 0.2087969  
## 9 1.0000 0.1868110 0.2152947

#Lasso  
for (i in 1:9) {  
 if (i<1) {  
 train=sample(1:nrow(x), a[i]\*nrow(x))  
 test=(-train)  
 y.test = y[-train]  
 alpha1.fit =cv.glmnet(x[train,],y[train], alpha=1, type.measure="mse",family="gaussian")  
 alpha1.predicted = predict(alpha1.fit, s=alpha1.fit$lambda.1se,newx=x[-train,])  
 tp[i]=alpha1.fit$lambda.1se  
 mse[i]=mean((y.test-alpha1.predicted)^2)  
 }   
 else {  
 train=sample(1:nrow(x), a[i]\*nrow(x))  
 test=(train)  
 y.test = y[train]  
 alpha1.fit =cv.glmnet(x[train,],y[train], alpha=1, type.measure="mse",family="gaussian")  
 alpha1.predicted = predict(alpha1.fit, s=alpha1.fit$lambda.1se,newx=x[train,])  
 tp[i]=alpha1.fit$lambda.1se  
 mse[i]=mean((y.test-alpha1.predicted)^2)  
 }  
   
}  
  
table\_Lasso=data.frame(RatioOfTrainingSet = a, tp=tp,MSE=mse)  
table\_Lasso

## RatioOfTrainingSet tp MSE  
## 1 0.5000 0.05541572 0.2192705  
## 2 0.5625 0.07042788 0.2347370  
## 3 0.6250 0.04659242 0.2043661  
## 4 0.6875 0.06036091 0.2075990  
## 5 0.7500 0.06036840 0.2022688  
## 6 0.8125 0.05063202 0.2036441  
## 7 0.8750 0.04950568 0.2046174  
## 8 0.9375 0.05522830 0.2120856  
## 9 1.0000 0.06559293 0.2170027

conclude that ratio between 0.625-0.75 is better ratio Using Elastic\_Net Regression and using 0.6875 times of data as training data

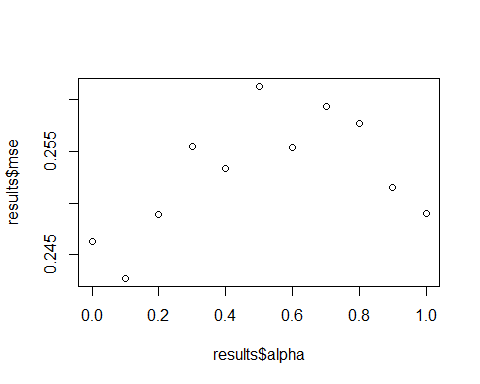
#Elastic\_Net Regression  
set.seed(1)  
train=sample(1:nrow(x), 0.6875\*nrow(x))  
test=(-train)  
y.test = y[-train]  
  
list.of.fits = list()  
for (i in 0:10) {  
 fit.name = paste0("alpha",i/10)  
 list.of.fits[[fit.name]] = cv.glmnet(x[train,], y[train],alpha=i/10,type.measure="mse", family="gaussian")  
}  
results= data.frame()  
  
for (i in 0:10) {  
 fit.name = paste0("alpha",i/10)  
 predicted =  
predict(list.of.fits[[fit.name]],s=list.of.fits[[fit.name]]$lambda.1se,newx=x[-train,])  
   
 mse = mean((y.test - predicted)^2)  
 temp = data.frame(alpha = i/10, mse=mse, fit.name = fit.name)  
 results = rbind(results,temp)  
}  
results

## alpha mse fit.name  
## 1 0.0 0.2463103 alpha0  
## 2 0.1 0.2426991 alpha0.1  
## 3 0.2 0.2488905 alpha0.2  
## 4 0.3 0.2554663 alpha0.3  
## 5 0.4 0.2533036 alpha0.4  
## 6 0.5 0.2613081 alpha0.5  
## 7 0.6 0.2553572 alpha0.6  
## 8 0.7 0.2593479 alpha0.7  
## 9 0.8 0.2576691 alpha0.8  
## 10 0.9 0.2514805 alpha0.9  
## 11 1.0 0.2489712 alpha1

results[match(min(results$mse),results$mse),]

## alpha mse fit.name  
## 2 0.1 0.2426991 alpha0.1

plot(results$alpha,results$mse)



alpha1=results[match(min(results$mse),results$mse),][1,1]  
alpha1

## [1] 0.1

list.of.fits = list()  
for (i in 0:10) {  
 fit.name = paste0("alpha",i/100 +alpha1)  
 list.of.fits[[fit.name]] = cv.glmnet(x[train,], y[train],alpha=i/100 +alpha1,type.measure="mse", family="gaussian")  
}  
results= data.frame()  
  
for (i in 0:10) {  
 fit.name = paste0("alpha",i/100 +alpha1)  
 predicted =  
predict(list.of.fits[[fit.name]],s=list.of.fits[[fit.name]]$lambda.1se,newx=x[-train,])  
   
 mse = mean((y.test - predicted)^2)  
 temp = data.frame(alpha = i/100+alpha1, mse=mse, fit.name = fit.name)  
 results = rbind(results,temp)  
}  
results[match(min(results$mse),results$mse),][1,1]

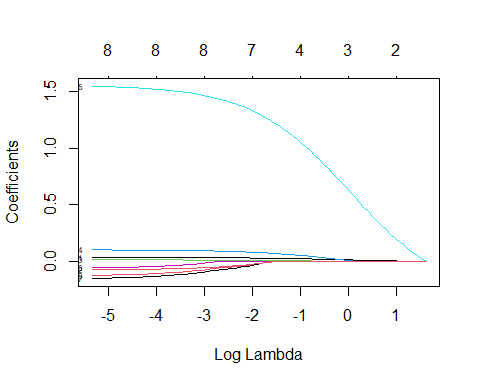
## [1] 0.11

results

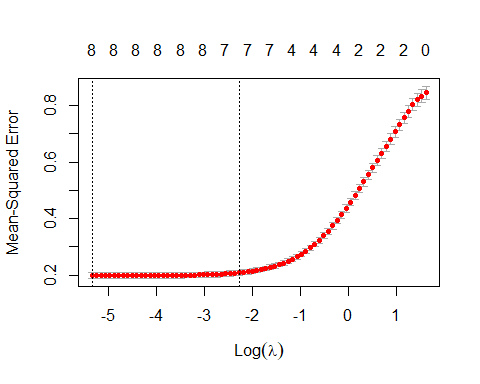
## alpha mse fit.name  
## 1 0.10 0.2511796 alpha0.1  
## 2 0.11 0.2430825 alpha0.11  
## 3 0.12 0.2454672 alpha0.12  
## 4 0.13 0.2462956 alpha0.13  
## 5 0.14 0.2525605 alpha0.14  
## 6 0.15 0.2511881 alpha0.15  
## 7 0.16 0.2525787 alpha0.16  
## 8 0.17 0.2490182 alpha0.17  
## 9 0.18 0.2505073 alpha0.18  
## 10 0.19 0.2541204 alpha0.19  
## 11 0.20 0.2584083 alpha0.2

get the result that alpha=0.12 in EN regression has lowest MSE

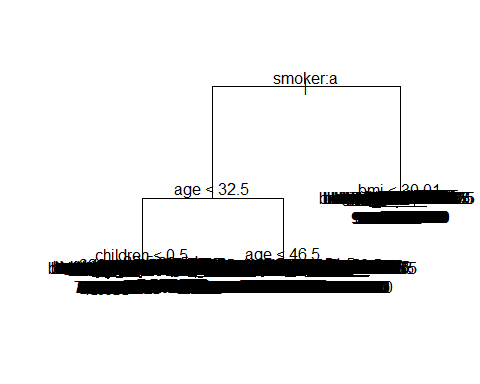
x=model.matrix(charges~.,med)[,-1]  
y=med$charges  
fit.model = glmnet(x,y,alpha=0.12)  
plot(fit.model, xvar="lambda", label=TRUE)



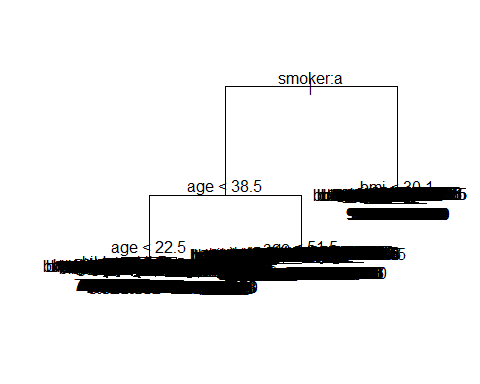
cv.model = cv.glmnet(x,y,alpha =0.12);plot(cv.model)



#Decision Tree- whole tree  
tree.M=tree(charges~. ,data=med,control= tree.control(nrow(med), mincut = 0,minsize = 2,mindev = 0))  
plot(tree.M)  
text(tree.M)



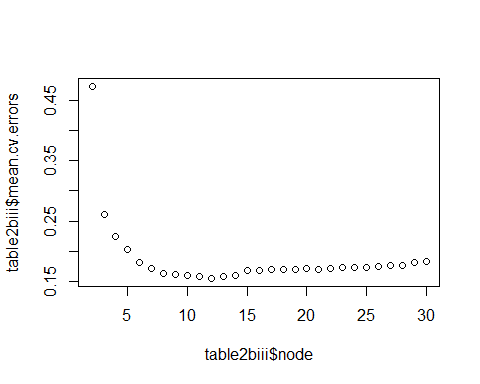
#using training set to build tree  
tree.M=tree(charges~. ,data=med[train,],control= tree.control(nrow(med[train,]), mincut = 1,minsize = 2,mindev = 0))  
plot(tree.M)  
text(tree.M)



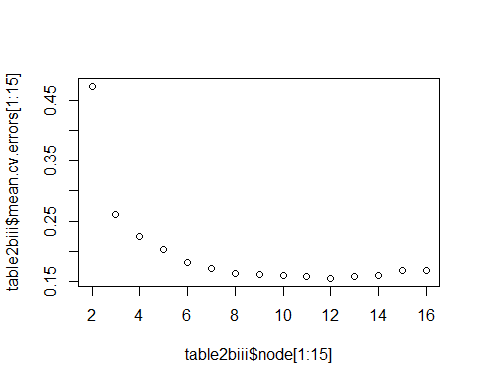
#MSE  
mean((med[test,"charges"]-predict(tree.M,med[test,]))^2)

## [1] 0.2845708

#Cross validation:Find the best pruned tree n=13  
k=10  
  
folds=sample(1:k,nrow(med),replace=TRUE)  
#folds  
cv.errors=matrix(NA,k,29, dimnames=list(NULL, paste(1:29)))  
for(j in 1:k){  
 set.seed(1)  
 tree.M2=tree(charges~. ,data=med[folds!=j,],control= tree.control(nrow(med[folds!=j,]), mincut = 1,minsize = 2,mindev = 0))  
 for(i in 1:29){  
 set.seed(1)  
 prune.M2=prune.tree(tree.M2,best=i+1)  
 pred=predict(prune.M2,med[folds==j,])  
 cv.errors[j,i]=mean((med[folds==j,"charges"]-pred)^2)  
 }  
}  
mean.cv.errors=apply(cv.errors,2,mean)  
#draw plot  
table2biii=data.frame(node=c(2:30),mean.cv.errors)  
plot(table2biii$node,table2biii$mean.cv.errors)



plot(table2biii$node[1:15],table2biii$mean.cv.errors[1:15])



n=which(table2biii$mean.cv.errors==min(table2biii$mean.cv.errors))+1  
n

## [1] 12

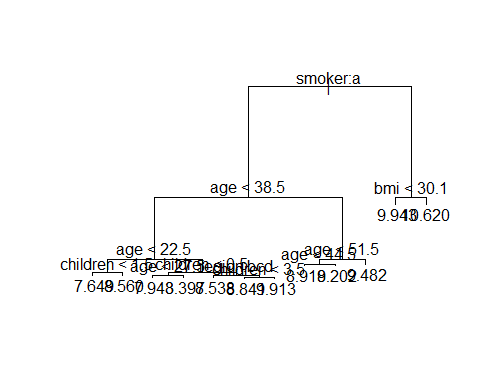
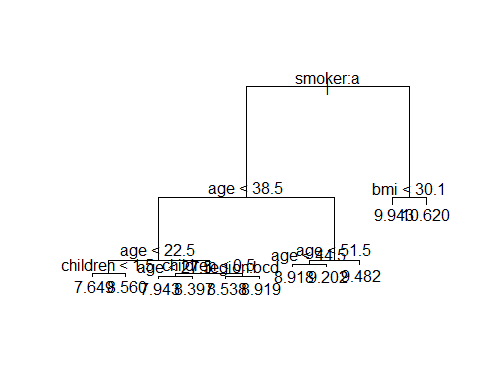
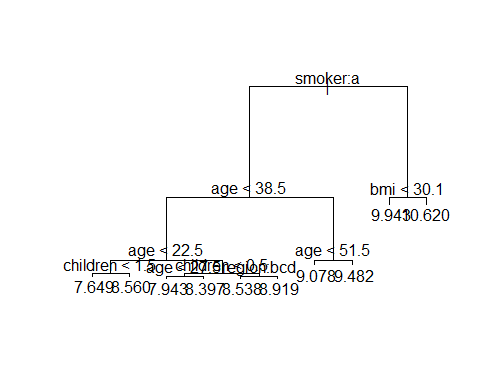
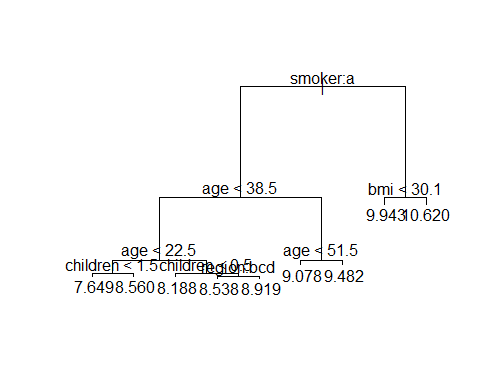
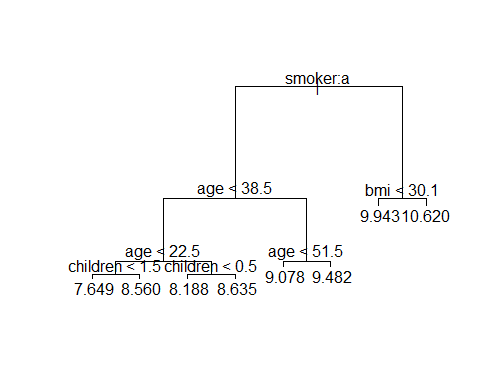
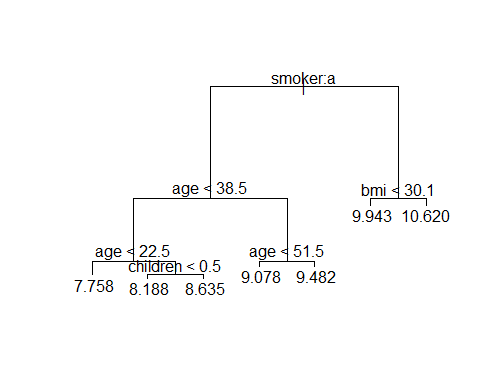
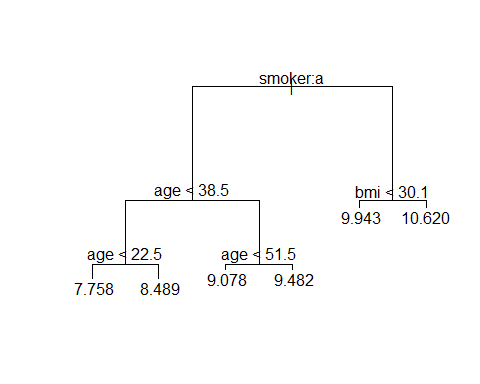
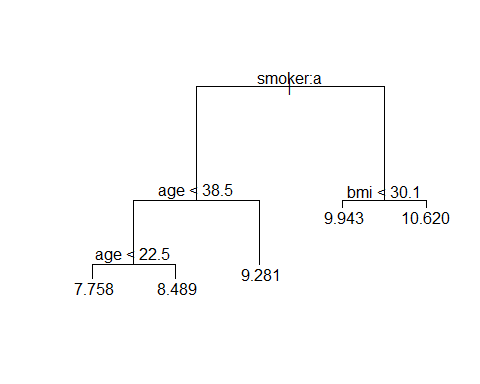
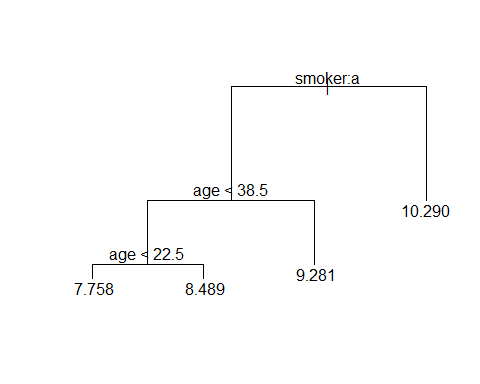
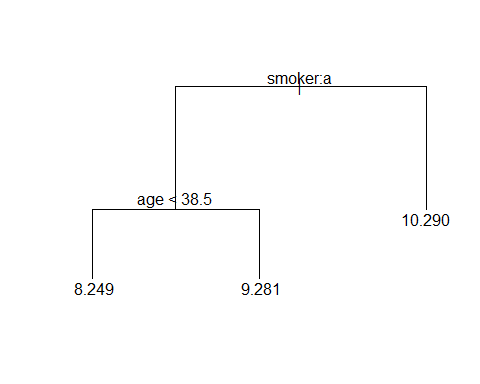
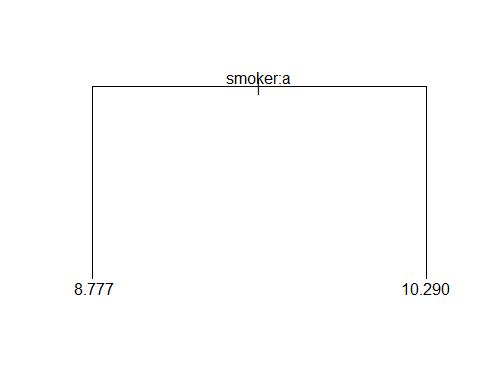
prune.M3=prune.tree(tree.M,best=n)  
mean((med[test,"charges"]-predict(prune.M3,med[test,]))^2)

## [1] 0.1915374

branchs=c(1:20)  
MSE.tr=c(rep(NA,20))  
  
for(j in 2:20){  
 prune.test=prune.tree(tree.M,best=j)  
 MSE.tr[j]=mean((med[test,"charges"]-predict(prune.test,med[test,]))^2)  
}  
MSE.tr[1]=mean((med[test,"charges"]-mean(med[train,"charges"]))^2)  
  
  
  
table.tr=data.frame(branchs,MSE.tr)  
table.tr

## branchs MSE.tr  
## 1 1 0.8721997  
## 2 2 0.5020904  
## 3 3 0.2915389  
## 4 4 0.2442833  
## 5 5 0.2207118  
## 6 6 0.2029762  
## 7 7 0.2043064  
## 8 8 0.1908290  
## 9 9 0.1904854  
## 10 10 0.1907478  
## 11 11 0.1890266  
## 12 12 0.1915374  
## 13 13 0.1919413  
## 14 14 0.1914564  
## 15 15 0.1957887  
## 16 16 0.1957887  
## 17 17 0.1957887  
## 18 18 0.2308279  
## 19 19 0.2308279  
## 20 20 0.2308279

for(j in 2:12){  
 prune.test=prune.tree(tree.M,best=j)  
 plot(prune.test)  
 text(prune.test)  
}



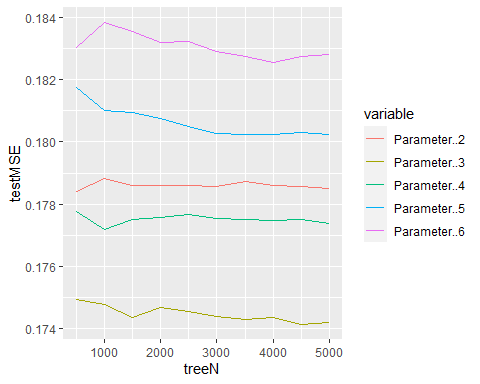
#test MSE  
MSE.lm=val.errors  
table.lm=data.frame(model=c(1:8),MSE.lm)  
table.compare=data.frame(table.lm,table.tr[5:12,])  
table.compare

## model MSE.lm branchs MSE.tr  
## 5 1 0.4586110 5 0.2207118  
## 6 2 0.2126700 6 0.2029762  
## 7 3 0.1962058 7 0.2043064  
## 8 4 0.1916911 8 0.1908290  
## 9 5 0.1911752 9 0.1904854  
## 10 6 0.1905866 10 0.1907478  
## 11 7 0.1886521 11 0.1890266  
## 12 8 0.1881355 12 0.1915374

p\_number=c(1:6)  
treeN=c(1:10)\*500  
error.mat=matrix(NA,length(treeN),length(p\_number))  
er.df=data.frame(error.mat)  
colnames(er.df) = paste0("Parameter#:",c(1:6))  
for(i in c(1:6)){  
   
 for(j in c(1:10)){  
 set.seed(1)  
 rf.med=randomForest(formula = charges ~ ., data = med[train,] , ntree = j\*500,mtry=i,proximity = TRUE)  
 yhat.rf.med=predict(rf.med, newdata = med[test,],n.trees =j\*500,mtry=i,proximity = TRUE)  
 er.df[j,i]=mean((yhat.rf.med$predicted-med[test,"charges"])^2)  
 }  
 print(i)  
}

## [1] 1  
## [1] 2  
## [1] 3  
## [1] 4  
## [1] 5  
## [1] 6

er.df2=data.frame(treeN,er.df[,2:6])  
data\_long <- melt(er.df2, id = "treeN")  
colnames(data\_long)[3]="testMSE"  
randomforest\_plot <- ggplot(data\_long,   
 aes(x = treeN,  
 y = testMSE,  
 color = variable)) + geom\_line()  
  
randomforest\_plot

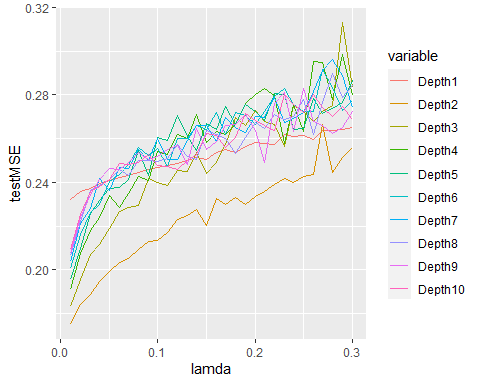


#using 3 parameter's rf has lowest MSE  
table.rf=er.df2[,c(1,3)]  
colnames(table.rf)[2]="MSE.rf\_p3"  
table.compare2=data.frame(table.compare,table.rf[3:10,])

d=c(1:10)  
lamda=c(1:30)/100  
error.mat.boost=matrix(NA,length(lamda),length(d))  
er.boost=data.frame(error.mat.boost)  
colnames(er.boost) = paste0("Depth",c(1:10))  
for(i in c(1:10)){  
 print(i)  
 for(j in c(1:30)){  
 set.seed(1)  
 boost.med=gbm(charges~.,data=med[train,],distribution = "gaussian",n.trees = 5000,interaction.depth = i,shrinkage = lamda[j])  
 yhat.boost=predict(boost.med,newdata = med[test,],n.trees = 5000,interaction.depth = i,shrinkage = lamda[j])  
 er.boost[j,i]=mean((yhat.boost-med[test,"charges"])^2)  
 }  
}

## [1] 1  
## [1] 2  
## [1] 3  
## [1] 4  
## [1] 5  
## [1] 6  
## [1] 7  
## [1] 8  
## [1] 9  
## [1] 10

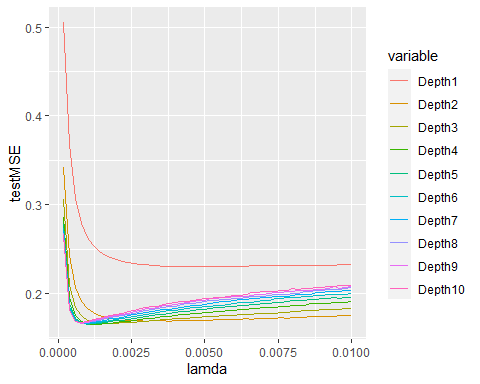
er.boost2=data.frame(lamda,er.boost)  
#colnames(er.df2)[1]="lamda"  
  
   
boost\_long <- melt(er.boost2, id = "lamda")  
colnames(boost\_long)[3]="testMSE"  
gfg\_plot <- ggplot(boost\_long,   
 aes(x = lamda,  
 y = testMSE,  
 color = variable)) + geom\_line()  
  
gfg\_plot



#lamda for 0.0002 ~0.01  
d.2=c(1:10)  
lamda.2=c(1:50)/5000  
error.mat.boost.2=matrix(NA,length(lamda.2),length(d.2))  
er.boost.2=data.frame(error.mat.boost.2)  
colnames(er.boost.2) = paste0("Depth",c(1:10))  
for(i in c(1:10)){  
 print(i)  
 for(j in c(1:50)){  
 set.seed(1)  
 boost.med.2=gbm(charges~.,data=med[train,],distribution = "gaussian",n.trees = 5000,interaction.depth = i,shrinkage = lamda.2[j])  
 yhat.boost.2=predict(boost.med.2,newdata = med[test,],n.trees = 5000,interaction.depth = i,shrinkage = lamda.2[j])  
 er.boost.2[j,i]=mean((yhat.boost.2-med[test,"charges"])^2)  
 }  
}

## [1] 1  
## [1] 2  
## [1] 3  
## [1] 4  
## [1] 5  
## [1] 6  
## [1] 7  
## [1] 8  
## [1] 9  
## [1] 10

er.boost2.2=data.frame(lamda.2,er.boost.2)  
#colnames(er.df2)[1]="lamda"  
  
   
boost\_long.2 <- melt(er.boost2.2, id = "lamda.2")  
colnames(boost\_long.2)[3]="testMSE"  
colnames(boost\_long.2)[1]="lamda"  
gfg\_plot.2 <- ggplot(boost\_long.2,   
 aes(x = lamda,  
 y = testMSE,  
 color = variable)) + geom\_line()  
  
gfg\_plot.2



er.boost2.2[which(er.boost2.2==min(er.boost2.2[,2:11]))%%50,1]

## [1] 0.0012

colnames(er.boost2.2)[which(er.boost2.2==min(er.boost2.2[,2:11]))%/%50+1]

## [1] "Depth4"

Depth=colnames(er.boost2.2)[2:11]  
MSE.bo\_l0.0012=t(er.boost2.2[6,2:11])[1:10]  
table.bo=data.frame(Depth,MSE.bo\_l0.0012)  
table.compare3=data.frame(table.compare2,table.bo[3:10,])

table.compare3

## model MSE.lm branchs MSE.tr treeN MSE.rf\_p3 Depth MSE.bo\_l0.0012  
## 5 1 0.4586110 5 0.2207118 1500 0.1743677 Depth3 0.1673605  
## 6 2 0.2126700 6 0.2029762 2000 0.1746719 Depth4 0.1652118  
## 7 3 0.1962058 7 0.2043064 2500 0.1745387 Depth5 0.1656875  
## 8 4 0.1916911 8 0.1908290 3000 0.1743732 Depth6 0.1667413  
## 9 5 0.1911752 9 0.1904854 3500 0.1742815 Depth7 0.1675478  
## 10 6 0.1905866 10 0.1907478 4000 0.1743549 Depth8 0.1684804  
## 11 7 0.1886521 11 0.1890266 4500 0.1741359 Depth9 0.1692687  
## 12 8 0.1881355 12 0.1915374 5000 0.1741971 Depth10 0.1702054