Final Project

Proposal

Introduction to Data Mining

Chitra Karki

**Prediction of carbon emission rate by the vegetations in Jornada**

Desert.

Introduction:

I took the data set from a friend of mine who is working in Environmental Science Department (UTEP). Their research is on studying the interaction of different vegetation habitat of Jornada desert in New Mexico state with the environment. They have experimental setup on the desert and the robot equipped with multiple sensors records different data in a certain radius. The data, I am interested in for this project is about the carbon dioxide exchange of the vegetations with the environment. In the data set there are 72 feature variables and a response variable. The response variable is the carbon dioxide exchange rate. Total number of observations are 116.

For the process of exchanging carbon dioxide, multiple factors may come in to play in nature, so as in their data collection. The key motive is to find the most important features which can describe the carbon dioxide exchange process.

Data cleaning:

Unfortunately ,the data set contains missing values. The sources of the missing values are circumstances due to which data were not able to be collected. One of the main reasons is the failure of the devices. The six rows contained more than 50 % missing values. They were removed and the rest of the missing values were imputed with package “mice” in R using method “CART”. Figure 1 shows the imputed values in 5 iterations. The missing values were then completed with command complete in R. The cleaned data set contains 116 observations and 70 features including response variable.

Correlation:

The correlation among the variables was calculated and plotted, Figure 2. The feature variables are correlated with each other. Moreover, the features also have good correlation with response variable. Due to presence of collinearity, the simple linear regressions were not performed in the process of model building.

Chart, histogram

Description automatically generated

A picture containing text

Description automatically generatedFig 1: Imputation of missing values using mice package with method CART. The features GPP.response and GCC were imputated in 5 iterations. Red represent imputated iterations and blue is the distribution of non-missing values of respective features.

Fig 2: Correlation plot

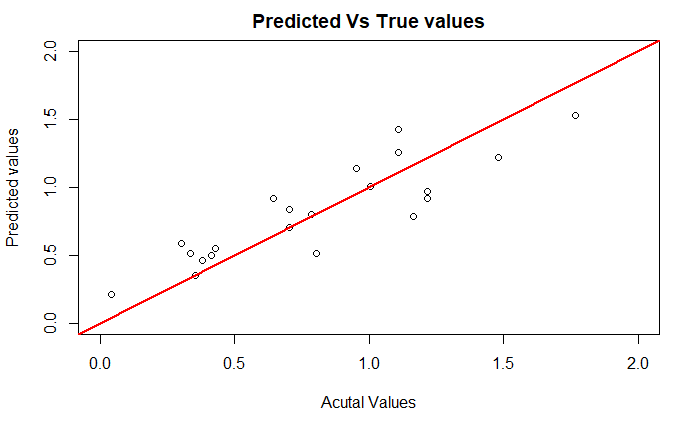
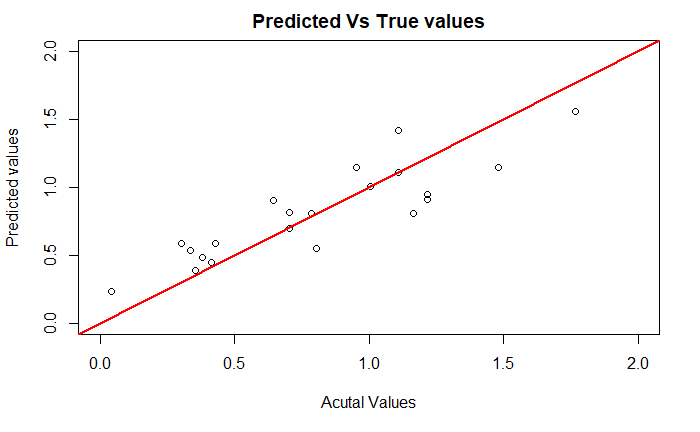
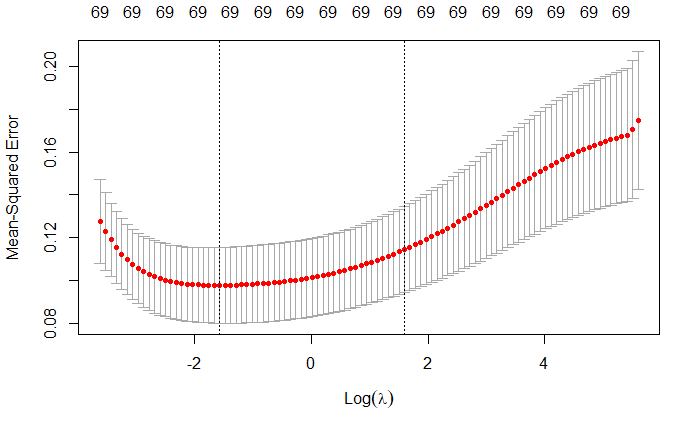
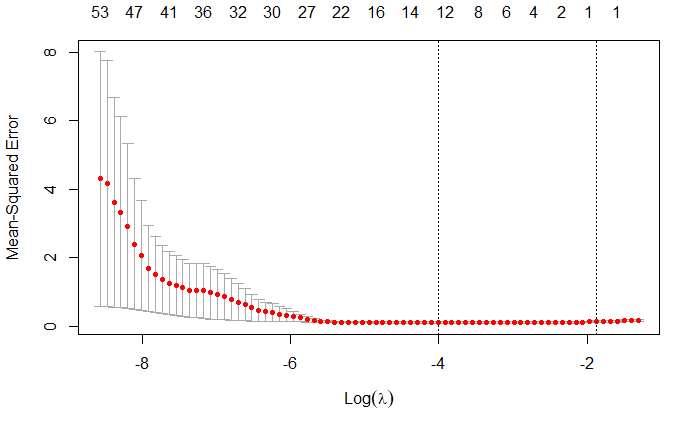
Model building :

The data set was splitted into training and testing sets in the ratio 4:1. The performance of the models will be tested on the testing set after the model is build. The mean square error (MSE) will be used to check the quality of the model.

Model 1 : Lasso and Ridge.

Since there was collinearity in the data set, the regularization methos like Lasso and Ridge regression were performed. 10-fold cross-validation (CV) was used the obtain the value of lambda for which CV error is minimum. The minimum lambda was found to be 0.01815451. Family type “gaussian” was used. The CV plot is shown in Figure 3a and the plot of predicted vs actual values of the test set Figure 3b. The MSE for Lasso was 0.04404099. The regression coefficients of the features "chl1a" "curvature" "gitelson4" "pri1" "rffr1" "rfred" "sr09" "sr12" "sr17" "wbi" "GCC" were not shrunked to zero.

Similar methods were incorporated for Ridge regression. The lambda value was 0.2087328 for the CV. The MSE was 0.04432362. The CV plot and the of predicted versus actual values are shown in figure 3(b) and 3(d).



**b**

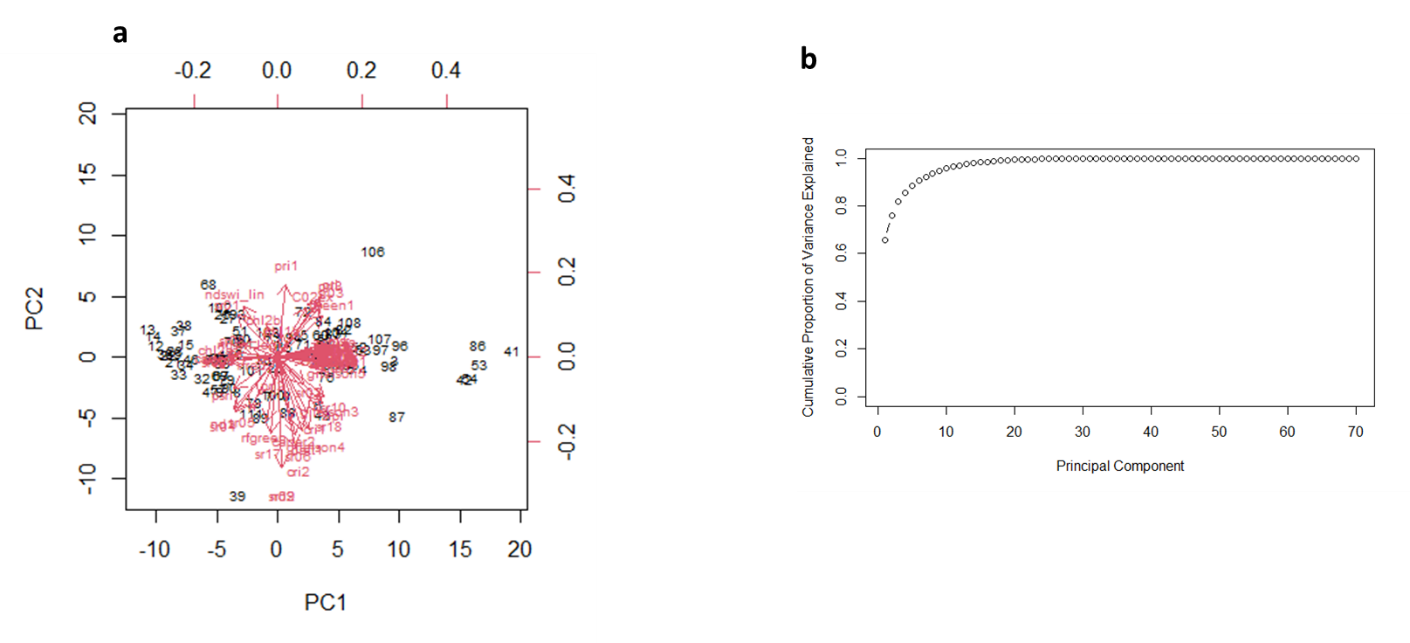
**a**

**c**

**d**

Fig 3: Comparison Lasso vs Ridge. (a) and (b) are tunning of lambda for lasso and Ridge regression using 10-fold CV. (c) and (d) are predicted vs actual values of lasso and Ridge, respectively.

Model 2: Principal Component Analysis and Regression

The first and second principal components explained 66 and 10 percent variation in the data. The bi-plot and cumulative contributions of the principal components are shown in figure 4. From the plot it is observed that 6 components explained 90% variation in the data. For the principal component regression, 2 components produced the minimum CV error, figure 5. The MSE was calculated to be 0.07712113.

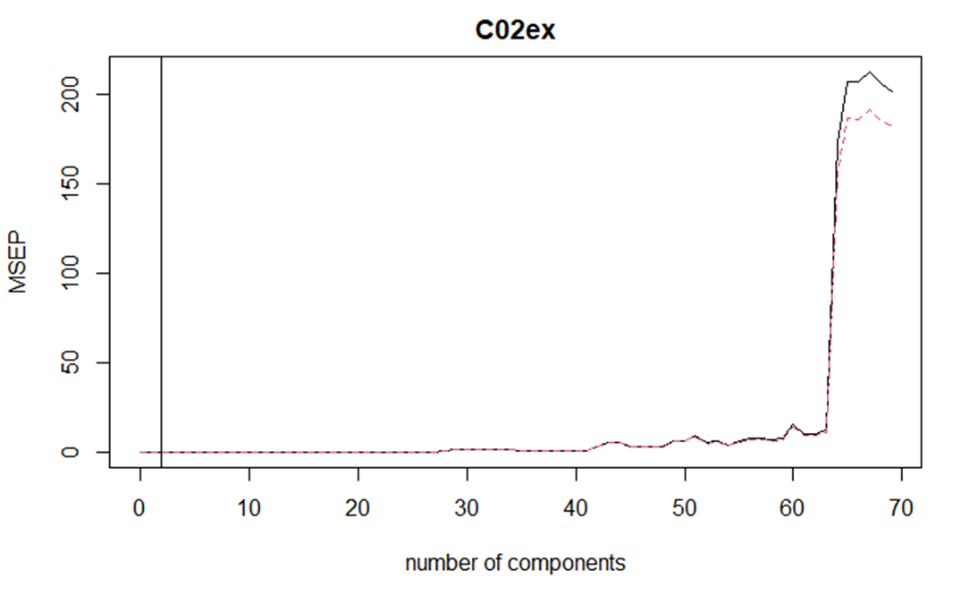
Fig 4: Principal component analysis. (a) pc1 and pc2 explain 66 and 10 percent variance. (b) plot of cumulative portion of each principal component. First six components explained 90 % variance.

Fig 5: Cross validation for PCR . Here MSE is used for error calculation. Red and black line indicate the cv errors and the adjusted cv errors, respectively.

Model 3: Regression Tree

Form CV its observed the tree of size 6 (figure 6 b) produced minimum CV error. This information was used to purne the tree. The purned model was then used for prediction over test set and to calculate MSE. MSE for the Regression tree was 0.0715114. The values on the leaves are the scaled. For the analysis purpose they can be unscaled. The whole tree, CV process and the purned tree are shown in figure 6.

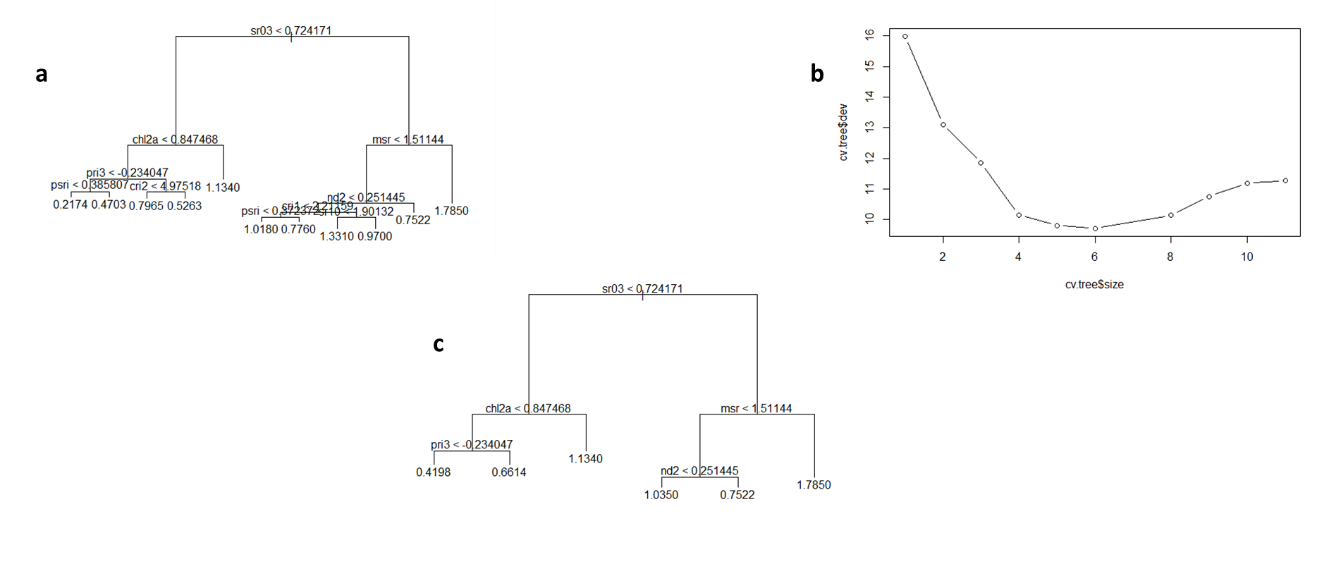


Fig 6: Regression Tree. (a) Full Tree. (b) cv for size of the tree. (c) purned tree

Model 4: Random Forest

1. Bagging: Form the CV analysis, 250 trees produced the minimum CV error, figure 7 a. This value was used for the bagging model. The plot for variable importance are shown in figure 7 b. The calculated MSE for the Bagging model was 0.05355638

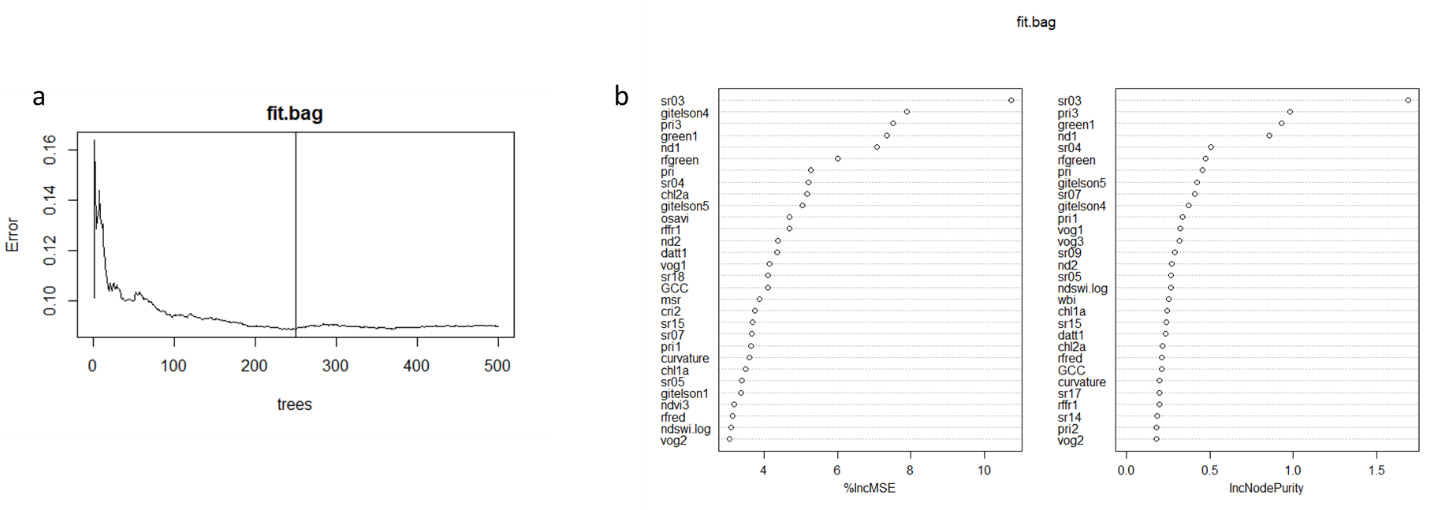


Fig 7: Bagging, (a) cv for number of trees . (b) variable importance

1. Random Forest: Form CV analysis, the optimum values of mtry and tress are 23 and 411, respectively. They produced minimum CV error as shown in figure 8. Moreover, figure contains the variable importance from Random Forest.

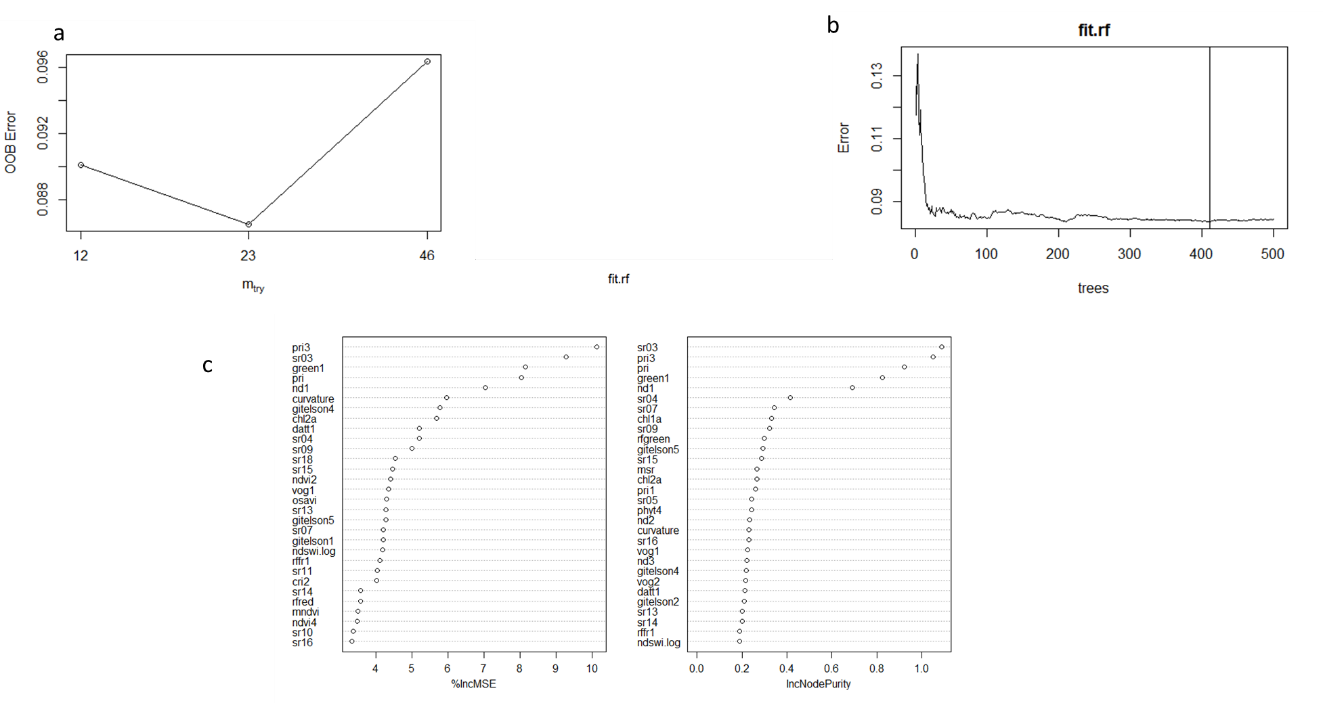


Fig 8: Random Forest. (a) tunning for mtry. (b) tunning for ntree. (c) variable importance.

Model 5: Boosting

The learning parameter was optimized with the cross validation. It obtained to be 0.80. And the MSE was calculated to be 0.07871491. the variable importance plot is shown in figure 9.

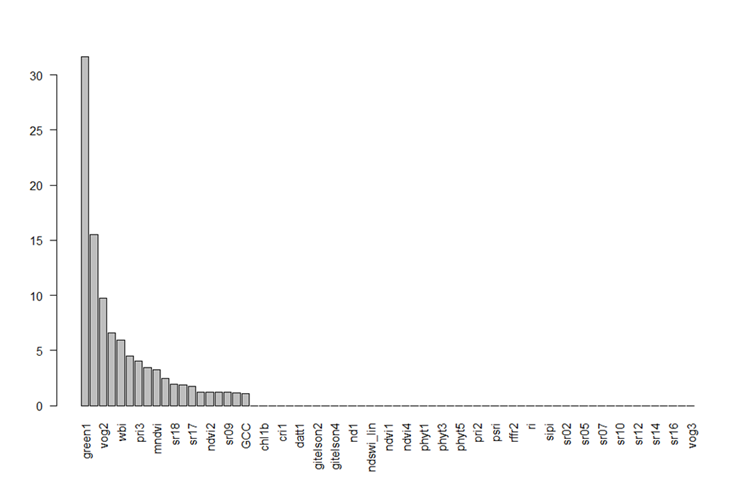
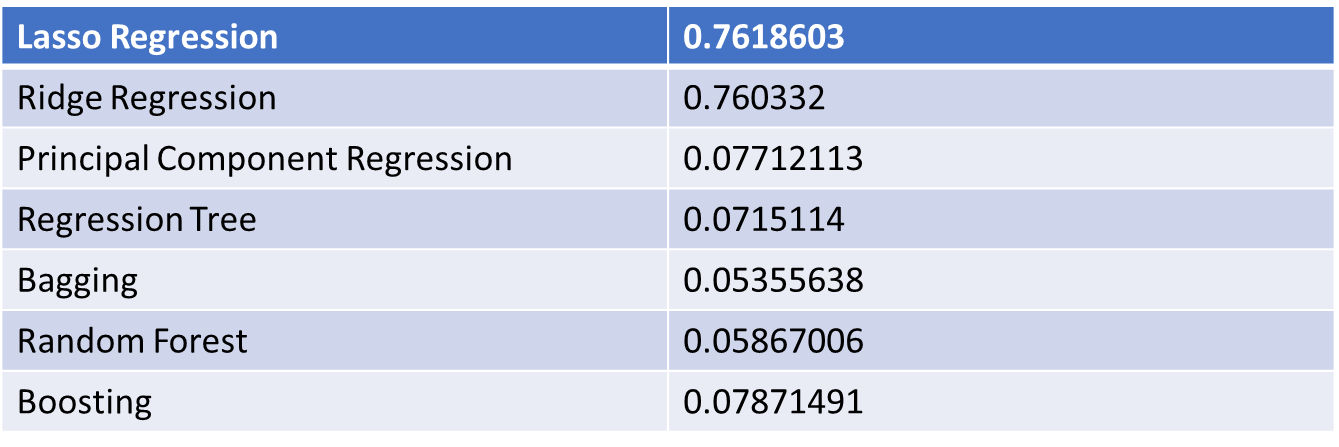


Fig 9: Variable importance form boosting

Conclusion:

The MSE calculated for different model is summarized in the table below.



* Various models were performed and their corresponding MSE are reported in table above.
* MSE are for test set.
* Different parameters of the models were tunned.
* Different models predicted different list of variable importance. We can determine the cutoff for the with further tests which is the scope of the future work.

.

The R- Codes are attached below.

---

title: "Final Project"

author: "Chitra Karki"

date: "11/23/2021"

output:

html\_document:

df\_print: paged

---

```{r setup, include=FALSE}

knitr::opts\_chunk$set(echo = TRUE)

```

``` {r}

df = read.csv(file = "resample\_gpp1.csv",header = T,

na.strings = c("inf","NA",""))

dim(df)

names(df)

na.per.col = apply(df,2,function(x) sum(is.na(x)/nrow(df)\*100)) # for col

table(na.per.col)

na.per.row = apply(df,1,function(x) sum(is.na(x)/ncol(df)\*100)) # for row

table(na.per.row)

which(na.per.row>50) # rows to be deleated. other missing values will be imputed

#using mice

df = df[-c(57,67,89,102,106),-1] # first column is the row name so remove for now.

# imputing the rest of the missing values

library(mice)

imputed\_df = mice(df,5,method = "cart",maxit = 5,seed = 500)

densityplot(imputed\_df)

df = complete(imputed\_df)

sum(is.na(df))

```

#correlation

``` {r}

df = data.frame(df[,c(1:68,70)],C02ex = df[,69])

names(df);dim(df)

str(df)

View(df)

cor = cor(df)

library("corrplot")

corrplot(cor,method="square",tl.pos = "n")

corrplot(cor.matrix)

corrplot(cor.matrix)

```

the correlation is seen among the features. also there is good correlation between the features and the response variable. some of the features has no correlation wit response.

# splliting data into test and validation set.

``` {r}

#dim(df) #after cleaning

set.seed(123)

train.index = sample(1:nrow(df),90) # 90 out of 111 for taining

# it will be easy for us to perform 10-fold cv

train = df[train.index,]

test = df[-train.index,]

```

## linear model (shirnkage method)

# lasso

``` {r}

# tunning lambda

library(glmnet)

#lambda = seq(0,80,0.01)

x = train[,-70]

y = train[,70]

# fitting lasso with 100 lamdas

fit.LR = glmnet(x=x,y=y,alpha = 1,family = "gaussian",nlambda =100)

# plot(fit,xvar = "lambda", label = T)

# plot(fit,xvar = "dev", label = T)

set.seed(123)

cv.LR = cv.glmnet(x = as.matrix(x),y = y,

nlambda = 100,nfolds = 10,alpha=1,family="gaussian")

plot(cv.LR)

# fitting with lamda min form cv

fit.minlam = glmnet(x=x,y=y,alpha = 1,family = "gaussian",lambda = cv.LR$lambda.min)

#plot(fit.minlam,label = T)

#coef(fit.minlam)

# prediction of LR with min lambda

pred.fit.minlam = predict(fit.minlam, as.matrix(test[,-70]),lambda = cv.LR$lambda.min)

# selected fearures with min lamda form cv

names(df)[which(fit.minlam$beta!=0)]

#coef(pred.fit.minlam)

# shrinkage of the features and cutoff of lambda

plot(fit.LR,xvar = "lambda", label = T)

abline(v=log(cv.LR$lambda.min))

# mean square error and plot of predicted vs actual

sse = sum((pred.fit.minlam - test[,70])^2)

mse.LR = sse/nrow(test); mse.LR

plot(y=pred.fit.minlam,x=test[,70],xlim = c(0,2),ylim = c(0,2),

xlab = "Acutal Values", ylab = "Predicted values",

main = "Predicted Vs True values")

abline(a=0,b=1,col="red",lwd=2)

# R-squared

sst = sum((test[,70]-mean(test[,70]))^2)

r.sq.LR = 1-sse/sst; r.sq.LR # 0.05103245

```

# Ridge

``` {r}

# fitting ridge with 100 lamdas

fit.RR = glmnet(x=x,y=y,alpha = 1,family = "gaussian",nlambda =100)

# plot(fit,xvar = "lambda", label = T)

# plot(fit,xvar = "dev", label = T)

# tunning lambda

set.seed(123)

cv.RR = cv.glmnet(x = as.matrix(x),y = y,

nlambda = 100,nfolds = 10,alpha=0,family="gaussian")

plot(cv.RR)

# fitting with lamda min form cv

fit.minlam = glmnet(x=x,y=y,alpha = 0,family = "gaussian",lambda = cv.RR$lambda.min)

#plot(fit.minlam,label = T)

#coef(fit.minlam)

# prediction of LR with min lambda

pred.fit.minlam = predict(fit.minlam, as.matrix(test[,-70]),lambda = cv.RR$lambda.min)

# selected fearures with min lamda form cv

names(df)[which(fit.minlam$beta!=0)]

# shrinkage of the features and cutoff of lambda

plot(fit.RR,xvar = "lambda", label = T)

abline(v=log(cv.RR$lambda.min))

# mean square error and plot of predicted vs actual

sse = sum((pred.fit.minlam - test[,70])^2)

mse.RR = sse/nrow(test); mse.RR # 0.05874862

plot(y=pred.fit.minlam,x=test[,70],xlim = c(0,2),ylim = c(0,2),

xlab = "Acutal Values", ylab = "Predicted values",

main = "Predicted Vs True values")

abline(a=0,b=1,col="red",lwd=2)

# R-squared

sst = sum((test[,70]-mean(test[,70]))^2)

r.sq.RR = 1-sse/sst; r.sq.RR # 0.6618741

```

# principal component analaysis

``` {r}

fit.pca = prcomp(train,scale. = T)

fit.pca

biplot(fit.pca,scale = 0,cex=0.65)

fit.pca$rotation

dim(fit.pca$x)

fit.pca$x

fit.pca$sdev

fit.pca.var = fit.pca$sdev ^2

propve = fit.pca.var/sum(fit.pca.var)

plot(propve, xlab = "principal component",

ylab = "Proportion of Variance Explained",

ylim = c(0, 1), type = "b",

main = "Scree Plot")

plot(cumsum(propve),

xlab = "Principal Component",

ylab = "Cumulative Proportion of Variance Explained",

ylim = c(0, 1), type = "b")

which(cumsum(propve) >= 0.9)[1]

```

# principal componenet Regression

``` {r}

library(pls)

set.seed(123)

fit.pcr = pcr(C02ex~.,data= train,scale=T,validation="CV")

summary(fit.pcr)

validationplot(fit.pcr,val.type = "MSEP")

abline(v=2)

pred.fit.pcr = predict(fit.pcr,test[,-70],ncomp = 2)

# mse

mse.fit.pcr = mean((pred.fit.pcr-test[,70])^2); mse.fit.pcr # 0.07632698

```

# partial least square regression

``` {r}

set.seed(123)

fit.pls = plsr(C02ex~.,data= train,scale=T,validation="CV")

summary(fit.pls)

validationplot(fit.pls,val.type="MSEP")

```

# regression tree

``` {r}

library(tree)

set.seed(123)

fit.tree = tree(C02ex~.,data= train)

summary(fit.tree)

plot(fit.tree)

text(fit.tree,pretty=0)

# cv to purn tree

set.seed(123)

cv.tree = cv.tree(fit.tree)

plot(cv.tree$size,cv.tree$dev,type="b") # the tree of size of 6 has minimum cv deviation.

# purning

prune.fit.tree = prune.tree(fit.tree,best = 6)

plot(prune.fit.tree); text(prune.fit.tree,pretty = 0)

# mse

pred.prune.fit.tree = predict(prune.fit.tree,newdata=test[,-70])

mse.predit.tree = mean((pred.prune.fit.tree - test[,70])^2); mse.predit.tree

```

# Bagging

``` {r}

library(randomForest)

set.seed(123)

fit.bag = randomForest(C02ex~.,data=train,mtry=69,importance=T)

which.min(fit.bag$mse)

plot(fit.bag); abline(v = 250)

varImpPlot(fit.bag)

importance(fit.bag)

pred.fit.bag = predict(fit.bag,newdata=test[,-70],ntree= 250)

mean((pred.fit.bag - test[,70])^2)

# fit.bag.opt = randomForest(C02ex~.,data=train,mtry=69,ntree=200)

#

# plot(fit.bag.opt)

# pred.fit.bag.opt = predict(fit.bag.opt,newdata=test[,-70])

# mean((pred.fit.bag.opt - test[,70])^2)

```

# Random Forest

``` {r}

set.seed(123)

tuneRF(train[,-70],train[,70]) ## oob error min for mtry 23

abline(v=23)

# set.seed(123)

# rf.cv = rfcv(train[,-70],train[,70])

# which.min(rf.cv$error.cv)

# plot(x=rf.cv$n.var,y=rf.cv$error.cv)

# abline(v = 34)

#which.min(rf.cv$error.cv)

set.seed(123)

fit.rf = randomForest(C02ex~.,data=train,importance=T,mtry=23 )

which.min(fit.rf$mse) # 411

plot(fit.rf); abline(v=411)

pred.fit.rf = predict(fit.rf,newdata=test[,-70],ntree =411)

#mse

mse.fit.rf = mean((pred.fit.rf - test[,70])^2);mse.fit.rf

importance(fit.rf)

varImpPlot(fit.rf)

```

# tunning parmeters for boosting

``` {r}

library("gbm")

hyper\_grid <- expand.grid(

shrinkage = c(.01, .1, .3),

interaction.depth = c(1, 3, 5),

n.minobsinnode = c(5, 10, 15),

bag.fraction = c(.65, .8, 1),

optimal\_trees = 0, # a place to dump results

min\_RMSE = 0 # a place to dump results

)

# total number of combinations

nrow(hyper\_grid)

# grid search

for(i in 1:nrow(hyper\_grid)) {

# reproducibility

set.seed(123)

# train model

gbm.tune <- gbm(

formula = C02ex ~ .,

distribution = "gaussian",

data = train,

n.trees = 5000,

interaction.depth = hyper\_grid$interaction.depth[i],

shrinkage = hyper\_grid$shrinkage[i],

n.minobsinnode = hyper\_grid$n.minobsinnode[i],

bag.fraction = hyper\_grid$bag.fraction[i],

train.fraction = .75,

n.cores = NULL, # will use all cores by default

verbose = FALSE

)

# add min training error and trees to grid

hyper\_grid$optimal\_trees[i] <- which.min(gbm.tune$valid.error)

hyper\_grid$min\_RMSE[i] <- sqrt(min(gbm.tune$valid.error))

}

library(dplyr)

hyper\_grid %>%

dplyr::arrange(min\_RMSE) %>%

head(10)

```

# Boosting

``` {r}

library(gbm)

set.seed(123)

#fit.boost = gbm(C02ex~.,data=train,distribution = "gaussian",cv.folds = 10)

fit.boost = gbm(C02ex~.,data = train,distribution = "gaussian",n.trees = 22,shrinkage = 0.30,interaction.depth =1 ,n.minobsinnode = 5, bag.fraction =0.80 )

tr = data.frame(summary(fit.boost))

barplot(tr$rel.inf,names.arg = tr$rel.inf,las=2)

pred.fit.boost=predict(fit.boost,newdata= test[,-70])

# mse

mean((pred.fit.boost - test[,70])^2)

```