```
library(caret)

library(ggplot2)

set1 <- read.csv("/Users/Ben/Desktop/Academic/QSAR PROJECT/R/set5/ILDesc.csv", stringsAsFactors = TRUE)

nzv <- nearZeroVar(set1)

set2 <- set1[, -nzv]


#Seprate by molecule class, training and testing

Indoles <- subset(set2, nO < 3)

Lactones <- subset(set2, nO > 2)


set.seed(859)

IndexI <- createDataPartition(Indoles$VAR001, p=.7, list = F)

IndexL <- createDataPartition(Lactones$VAR001, p=.7, list = F)

trainIndoles <- Indoles[IndexI, ]

testingIndoles <- Indoles[-IndexI, ]

trainlactones <- Lactones[IndexL, ]

testinglactones <- Lactones[-IndexL, ]

Indoles


#Use RFE to find selected descriptors

k <- ncol(trainIndoles)

k1 <- ncol(trainlactones)

DescI <- na.omit(trainIndoles[,-k])

BindingI <- na.omit(trainIndoles[,k])

DescL <- na.omit(trainlactones[,-k1])

BindingL <- na.omit(trainlactones[,k1])

numdesc <- 25

controlI <- rfeControl(functions=rfFuncs, method="cv", number = 10, repeats = 10)
```

```
controlL <- rfeControl(functions=rfFuncs, method="cv", number = 7, repeats = 10)

resultsI <- rfe(DescI, BindingI, sizes = numdesc, rfeControl=controlI)

resultsL <- rfe(DescL, BindingL, sizes = numdesc, rfeControl=controlL)

resultsI

resultsL


size = c(25,50,100,500,1000,2000)

resultsI <- rfe(DescI, BindingI, sizes = size, rfeControl=control)

resultsL <- rfe(DescL, BindingL, sizes = size, rfeControl=control)

ggplot(resultsI) + xlim(c(0, 2000)) + theme(text = element_text(size=20))

ggplot(resultsL) + xlim(c(0, 2000)) + theme(text = element_text(size=20))


colI <- c(names(resultsI$fit$forest$xlevels))

colL <- c(names(resultsL$fit$forest$xlevels))

trainI <-droplevels(trainIndoles[,c(colI,"VAR001")])

trainL <- droplevels(trainlactones[,c(colL,"VAR001")])


#brnns

controlI <- trainControl(method = "cv", number = 5, returnResamp = "final", search = "random")

controlL <- trainControl(method = "cv", number = 3, returnResamp = "final", search = "random")

metric <- "RMSE"

modelI <- train(VAR001~., data = trainI, method = "brnn", tuneLength = 1,trControl = controlI, verbose
=FALSE)

modelL <- train(VAR001~., data = trainL, method = "brnn", tuneLength = 1,trControl = controlL, verbose
=FALSE)


modelI

modelL
```

```
#predicted vs test set
predTestI <- predict(modelI, newdata = testingIndoles[,-k])
realTestI <- testingIndoles[,k]
RMSE<-caret::RMSE(predTestI,realTestI)
R2<-caret::R2(predTestI,realTestI)
print(R2)
print(RMSE)
ggplot(mapping = aes(x = predTestI,y = realTestI)) +                #pred vs test set
  geom_point() +
  theme(text = element_text(size=20)) +
  scale_x_continuous(name = "binding affinity predicted by caret") +
  scale_y_continuous(name = "binding affinity predicted by autodock") +
  geom_smooth(method = "lm", se= FALSE, color="black", aes(group=1)) +
  annotate("rect", xmin = -9.5, xmax = -10.4, ymin = -8.5, ymax = -8.85, fill="white", colour="red") +
  annotate("text", x=-10, y=-8.6, label = "R^2 == 0.875", parse=T) +
  annotate("text", x=-10, y=-8.75, label = "RMSE == 0.495", parse=T)

predTestL <- predict.train(modelL, newdata = testinglactones[,-k])
realTestL <- testinglactones[,k]
RMSE<-caret::RMSE(predTestL,realTestL)
R2<-caret::R2(predTestL,realTestL)
print(R2)
print(RMSE)
ggplot(mapping = aes(x = predTestL,y = realTestL)) +                #pred vs test set
  geom_point() +
  theme(text = element_text(size=20)) +
  scale_x_continuous(name = "binding affinity predicted by caret") +
  scale_y_continuous(name = "binding affinity predicted by autodock") +
  geom_smooth(method = "lm", se= FALSE, color="black", aes(group=1)) +
```

```
  annotate("rect", xmin = -9.33, xmax = -9.08, ymin = -8.73, ymax = -8.83, fill="white", colour="red") +
  annotate("text", x=-9.2, y=-8.75, label = "R^2 == 0.873", parse=T) +
  annotate("text", x=-9.2, y=-8.8, label = "RMSE == 0.189", parse=T)

#Predicted vs for all set
predTestI <- predict.train(modelI, newdata = Indoles[,-k])
realTestI <- Indoles[,k]
ggplot(mapping = aes(x = predTestI,y = realTestI)) +                #pred vs test set
  geom_point() +
  geom_smooth(method = "lm", se= TRUE, color="black", aes(group=1))
predTestL <- predict.train(modelL, newdata = Lactones[,-k])
realTestL <- Lactones[,k]
ggplot(mapping = aes(x = predTestL,y = realTestL)) +                #pred vs test set
  geom_point() +
  geom_smooth(method = "lm", se= TRUE, color="black", aes(group=1))

predTest <- c(predTestI,predTestL)
realTest <- c(realTestI,realTestL)
RMSE<-caret::RMSE(predTest,realTest)
R2<-caret::R2(predTest,realTest)
print(R2)
print(RMSE)
ggplot() +
  theme(text = element_text(size=20)) +
  geom_point(mapping = aes(x = predTestI,y = realTestI, colour = 'blue')) +
  geom_point(mapping = aes(x = predTestL,y = realTestL, colour = 'red')) +
  scale_x_continuous(name = "binding affinity predicted by caret") +
  scale_y_continuous(name = "binding affinity predicted by autodock") +
  scale_color_manual(labels = c("Indoles", "Lactones"), values = c("blue", "red")) +
```

```
 geom_smooth(method = "lm", se= TRUE, color="black", aes(x = predTest,y = realTest)) +
 annotate("rect", xmin = -9.9, xmax = -11.1, ymin = -8.1, ymax = -8.7, fill="white", colour="red") +
 annotate("text", x=-10.5, y=-8.3, label = "R^2 == 0.961", parse=T) +
 annotate("text", x=-10.5, y=-8.5, label = "RMSE == 0.211", parse=T) +
 guides(fill=guide_legend(title="New Legend Title")) +
 theme(legend.title = element_blank())

#model type for accuracy
colI <- c(names(resultsI$fit$forest$xlevels))
trainI <-droplevels(trainIndoles[,c(colI,"VAR001")])
IGrid <- expand.grid(.size=c(numdescI), .decay=c(1))
bcontrol <- trainControl(method = "cv", number = 3, returnResamp = "final", search = "random")
metric <- "RMSE"
modelInn <- train(VAR001~. , data = trainI, method = "nnet",trControl = bcontrol,
                  tuneGrid = IGrid, maxit = 1000, trace = F, linout = TRUE, MaxNWts = 1200)
modelIbrnn <- train(VAR001~., data = trainI, method = "brnn", tuneLength = 2,trControl = bcontrol,
verbose =FALSE)
modelIrf <- train(VAR001~., data=trainI, method="rf", metric=metric, trControl=bcontrol)
modelIsvm <- train(VAR001~., data=trainI, method="svmRadial", metric=metric, trControl=bcontrol)
resultsI <- resamples(list(nn = modelInn,brnn = modelIbrnn, svm=modelIsvm, rf=modelIrf))
summary(resultsI)
bwplot(resultsI)

colL <- c(names(resultsL$fit$forest$xlevels))
trainL <- droplevels(trainlactones[,c(colL,"VAR001")])
LGrid <- expand.grid(.size=c(numdescL), .decay=c(0.1))
bcontrol <- trainControl(method = "cv", number = 3, returnResamp = "final", search = "random")
metric <- "RMSE"
modelLnn <- train(VAR001~. , data = trainL, method = "nnet",trControl = bcontrol,
```

```
            tuneGrid = lGrid, maxit = 1000, trace = F, linout = TRUE, MaxNWts = 1200)
modelLbrnn <- train(VAR001~., data = trainL, method = "brnn", tuneLength = 2,trControl = bcontrol,
verbose =FALSE)
modelLrf <- train(VAR001~., data=trainL, method="rf", metric=metric, trControl=bcontrol)
modelLsvm <- train(VAR001~., data=trainL, method="svmRadial", metric=metric, trControl=bcontrol)
resultsL <- resamples(list(nn = modelLnn,brnn = modelLbrnn, svm=modelLsvm, rf=modelLrf))
summary(resultsL)
bwplot(resultsL)

#number of descriptors for accuracy
descResults <- data.frame(matrix(nrow = 50, ncol = 2))
trials <- c(10:50)
for (i in trials){
  numdesc <- i
  descResults[i,1] <- i
  controlI <- rfeControl(functions=rfFuncs, method="cv", number = 10, repeats = 3)
  resultsI <- rfe(DescI, BindingI, sizes = numdescI, rfeControl=controlI)
  colI <- c(names(resultsI$fit$forest$xlevels))
  trainI <-droplevels(trainIndoles[,c(colI,"VAR001")])
  controlL <- trainControl(method = "cv", number = 3, returnResamp = "final", search = "random")
  metric <- "RMSE"
  modelI <- train(VAR001~., data = trainI, method = "brnn", tuneLength = 1,trControl = controlL, verbose
=FALSE)
  descResults[i,2] <- modelI$results$Rsquared
  print(i)
}
ggplot(mapping = aes( x = descResults[,1],y = descResults[,2])) +
  geom_point() +
  geom_smooth()
```

```
#descriptor matches

control <- trainControl(method="repeatedcv", number=10, repeats=3, search="grid")

tunegrid <- expand.grid(.mtry=c(sqrt(ncol(x))))

modellist <- list()

for (ntree in c(100, 150, 200, 250)) {

  set.seed(seed)

  fit <- train(Class~., data=dataset, method="rf", metric=metric, tuneGrid=tunegrid, trControl=control,
ntree=ntree)

  key <- toString(ntree)

  modellist[[key]] <- fit

}

Indy <- c(str(trainI))

Lacty <- c(str(trainL))

pmatch(Indy, Lacty)
```