```
library(caret)
library(gbm)

data("scat")


preProcValues <- preProcess(scat, method = c("knnImpute","center","scale"))
library('RANN')
train_processed <- predict(preProcValues, scat)


unique(train_processed$Species)

#Question 1
train_processed$Species<-ifelse(scat$Species=='coyote',0,ifelse(scat$Species=='bobcat',1,2))

#Question 2

#https://www.listendata.com/2015/06/r-keep-drop-columns-from-data-frame.html
train_processed = subset(train_processed, select = -c(Month, Year, Site, Location) )


#Question 3
sum(is.na(train_processed))

#No null values

#Question 4


dmy <- dummyVars(" ~ .", data = train_processed,fullRank = T)
train_transformed <- data.frame(predict(dmy, newdata = train_processed))

#Question 5

train_transformed$Species<-as.factor(train_transformed$Species)

set.seed(100)
index <- createDataPartition(train_transformed$Species, p=0.75, list=FALSE)
trainSet <- train_transformed[ index,]
testSet <- train_transformed[-index,]

str(trainSet)
```

```
#Feature selection using rfe in caret
control <- rfeControl(functions = rfFuncs,
                      method = "repeatedcv",
                      repeats = 3,
                      verbose = FALSE)
outcomeName<-'Species'
predictors<-names(trainSet)[!names(trainSet) %in% outcomeName]
Species_Pred_Profile <- rfe(trainSet[,predictors], trainSet[,outcomeName],rfeControl = control)
Species_Pred_Profile

#Taking only the top 4 predictors
predictors <- c("d15N", "d13C", "Mass", "CN")

names(getModelInfo())

model_gbm<-train(trainSet[,predictors],trainSet[,outcomeName],method='gbm')
model_rf<-train(trainSet[,predictors],trainSet[,outcomeName],method='rf')
model_nnet<-train(trainSet[,predictors],trainSet[,outcomeName],method='nnet')
model_NB <- train(trainSet[,predictors],trainSet[,outcomeName],method='nb')

fitControl <- trainControl(
  method = "repeatedcv",
  number = 5,
  repeats = 5)

### Using tuneGrid ####
modelLookup(model='gbm')

#Creating grid
grid <-
expand.grid(n.trees=c(10,20,50,100,500,1000),shrinkage=c(0.01,0.05,0.1,0.5),n.minobsinnode
= c(3,5,10),interaction.depth=c(1,5,10))

# training the model
model_gbm<-train(trainSet[,predictors],trainSet[,outcomeName],method='gbm',trControl=fitCont
rol,tuneGrid=grid)

# summarizing the model
print(model_gbm)


# Visualizing the models
```

```
plot(model_gbm)

### Using tuneLength ###

#using tune length
model_gbm<-train(trainSet[,predictors],trainSet[,outcomeName],method='gbm',trControl=fitCont
rol,tuneLength=10)
print(model_gbm)

# visualize the models
plot(model_gbm)

############ Variable importance estimation using caret ##################
#Checking variable importance for GBM
#Variable Importance
varImp(object=model_gbm)


#Plotting Varianle importance for GBM
plot(varImp(object=model_gbm),main="GBM - Variable Importance")

#Checking variable importance for RF
varImp(object=model_rf)


#Plotting Varianle importance for Random Forest
plot(varImp(object=model_rf),main="RF - Variable Importance")

#Checking variable importance for NNET
varImp(object=model_nnet)
#nnet variable importance

#Plotting Variable importance for Neural Network
plot(varImp(object=model_nnet),main="NNET - Variable Importance")
```