Maryam Bokhari

12/02/2019

Assignment #5

QUESTION 1

```
library(caret)
> animal_data <-scat
> str(scat)
'data.frame':	110 obs. of  19 variables:
 $ Species  : Factor w/ 3 levels "bobcat","coyote",..: 2 2 1 2 2 2 1 1 1 1 ..
.
 $ Month    : Factor w/ 9 levels "April","August",..: 4 4 4 4 4 4 4 4 4 4 ...
 $ Year     : int  2012 2012 2012 2012 2012 2012 2012 2012 2012 2012 ...
 $ Site     : Factor w/ 2 levels "ANNU","YOLA": 2 2 2 2 2 2 1 1 1 1 ...
 $ Location : Factor w/ 3 levels "edge","middle",..: 1 1 2 2 1 1 3 3 3 2 ...
 $ Age      : int  5 3 3 5 5 5 1 3 5 5 ...
 $ Number   : int  2 2 2 2 4 3 5 7 2 1 ...
 $ Length   : num  9.5 14 9 8.5 8 9 6 5.5 11 20.5 ...
 $ Diameter : num  25.7 25.4 18.8 18.1 20.7 21.2 15.7 21.9 17.5 18 ...
 $ Taper    : num  41.9 37.1 16.5 24.7 20.1 28.5 8.2 19.3 29.1 21.4 ...
 $ TI       : num  1.63 1.46 0.88 1.36 0.97 1.34 0.52 0.88 1.66 1.19 ...
 $ Mass     : num  15.9 17.6 8.4 7.4 25.4 ...
 $ d13C     : num  -26.9 -29.6 -28.7 -20.1 -23.2 ...
 $ d15N     : num  6.94 9.87 8.52 5.79 7.01 8.28 4.2 3.89 7.34 6.06 ...
 $ CN       : num  8.5 11.3 8.1 11.5 10.6 9 5.4 5.6 5.8 7.7 ...
 $ ropey    : int  0 0 1 1 0 1 1 0 0 1 ...
 $ segmented: int  0 0 1 0 1 0 1 1 1 1 ...
 $ flat     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ scrape   : int  0 0 1 0 0 0 1 0 0 0 ...
>
> head(animal_data[, 1:10])
  Species   Month Year Site Location Age Number Length Diameter Taper
1  coyote January 2012 YOLA     edge   5      2    9.5     25.7  41.9
2  coyote January 2012 YOLA     edge   3      2   14.0     25.4  37.1
3  bobcat January 2012 YOLA   middle   3      2    9.0     18.8  16.5
4  coyote January 2012 YOLA   middle   5      2    8.5     18.1  24.7
6  coyote January 2012 YOLA     edge   5      4    8.0     20.7  20.1
7  coyote January 2012 YOLA     edge   5      3    9.0     21.2  28.5
>
> sum(is.na(animal_data$Species))
[1] 0
> #Q1.  Takes Species as the target value and convert into numerical
> animal_data$Species <-ifelse(animal_data$Species =="bobcat", 1, ifelse(anim
al_data$Species =="coyote", 2, 3))
> target<-animal_data$Species
> str(target)
 num [1:110] 2 2 1 2 2 2 1 1 1 1 ...
```

QUESTION 2

```
>
> #Q2.  Remove the Month, Year, Site, Location features
>
> animal_data$Month<-NULL
> animal_data$Year<-NULL
> animal_data$Site<-NULL
> animal_data$Location<-NULL
```

```
>str(animal_data)
'data.frame':	110 obs. of  15 variables:
 $ Species  : num  2 2 1 2 2 2 1 1 1 1 ...
 $ Age      : int  5 3 3 5 5 5 1 3 5 5 ...
 $ Number   : int  2 2 2 2 4 3 5 7 2 1 ...
 $ Length   : num  9.5 14 9 8.5 8 9 6 5.5 11 20.5 ...
 $ Diameter : num  25.7 25.4 18.8 18.1 20.7 21.2 15.7 21.9 17.5 18 ...
 $ Taper    : num  41.9 37.1 16.5 24.7 20.1 28.5 8.2 19.3 29.1 21.4 ...
 $ TI       : num  1.63 1.46 0.88 1.36 0.97 1.34 0.52 0.88 1.66 1.19 ...
 $ Mass     : num  15.9 17.6 8.4 7.4 25.4 ...
 $ d13C     : num  -26.9 -29.6 -28.7 -20.1 -23.2 ...
 $ d15N     : num  6.94 9.87 8.52 5.79 7.01 8.28 4.2 3.89 7.34 6.06 ...
 $ CN       : num  8.5 11.3 8.1 11.5 10.6 9 5.4 5.6 5.8 7.7 ...
 $ ropey    : int  0 0 1 1 0 1 1 0 0 1 ...
 $ segmented: int  0 0 1 0 1 0 1 1 1 1 ...
 $ flat     : int  0 0 0 0 0 0 0 0 0 0 ...
 $ scrape   : int  0 0 1 0 0 0 1 0 0 0 ...
```

QUESTION 3

```
> #Q3. Check if any values are null. If there are, impute missing values usin
g KNN.
> sum(is.na(animal_data))
[1] 47
> library('RANN')
> preProcValues <- preProcess(animal_data, method = c("knnImpute","center","s
cale"))
> train_processed <- predict(preProcValues, animal_data)
> sum(is.na(train_processed))
[1] 0
> str(train_processed)
'data.frame':	110 obs. of  15 variables:
 $ Species  : num  0.356 0.356 -0.868 0.356 0.356 ...
 $ Age      : num  1.207 -0.252 -0.252 1.207 1.207 ...
 $ Number   : num  -0.433 -0.433 -0.433 -0.433 0.968 ...
 $ Length   : num  0.0587 1.3679 -0.0867 -0.2322 -0.3777 ...
 $ Diameter : num  1.8396 1.7623 0.0622 -0.1181 0.5516 ...
 $ Taper    : num  0.961 0.642 -0.726 -0.182 -0.487 ...
 $ TI       : num  0.0283 -0.1406 -0.7171 -0.24 -0.6277 ...
 $ Mass     : num  0.388 0.583 -0.458 -0.571 1.469 ...
 $ d13C     : num  0.00468 -1.26856 -0.85947 3.12113 1.66403 ...
 $ d15N     : num  -0.165 0.807 0.359 -0.546 -0.141 ...
 $ CN       : num  0.0276 0.7922 -0.0816 0.8468 0.6011 ...
 $ ropey    : num  -1.131 -1.131 0.876 0.876 -1.131 ...
 $ segmented: num  -1.131 -1.131 0.876 -1.131 0.876 ...
 $ flat     : num  -0.239 -0.239 -0.239 -0.239 -0.239 ...
 $ scrape   : num  -0.217 -0.217 4.562 -0.217 -0.217 ...
```

QUESTION 4

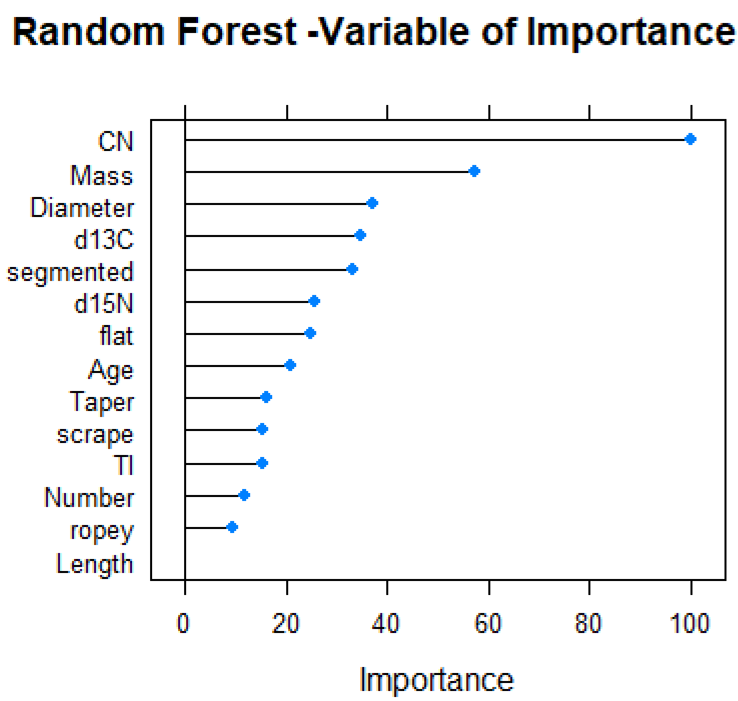
```
>#Q4. Converting every categorical variable to numerical.
> dmy <- dummyVars(" ~ .", data = train_processed ,fullRank = T)
> train_transformed <- data.frame(predict(dmy, newdata = train_processed))
> str(train_transformed)
'data.frame':	110 obs. of  15 variables:
 $ Species  : num  0.356 0.356 -0.868 0.356 0.356 ...
 $ Age      : num  1.207 -0.252 -0.252 1.207 1.207 ...
 $ Number   : num  -0.433 -0.433 -0.433 -0.433 0.968 ...
 $ Length   : num  0.0587 1.3679 -0.0867 -0.2322 -0.3777 ...
 $ Diameter : num  1.8396 1.7623 0.0622 -0.1181 0.5516 ...
 $ Taper    : num  0.961 0.642 -0.726 -0.182 -0.487 ...
 $ TI       : num  0.0283 -0.1406 -0.7171 -0.24 -0.6277 ...
 $ Mass     : num  0.388 0.583 -0.458 -0.571 1.469 ...
```

```
 $ d13C     : num  0.00468 -1.26856 -0.85947 3.12113 1.66403 ...
 $ d15N     : num  -0.165 0.807 0.359 -0.546 -0.141 ...
 $ CN       : num  0.0276 0.7922 -0.0816 0.8468 0.6011 ...
 $ ropey    : num  -1.131 -1.131 0.876 0.876 -1.131 ...
 $ segmented: num  -1.131 -1.131 0.876 -1.131 0.876 ...
 $ flat     : num  -0.239 -0.239 -0.239 -0.239 -0.239 ...
 $ scrape   : num  -0.217 -0.217 4.562 -0.217 -0.217 ...
> train_transformed$Species<-as.factor(train_transformed$Species)
```

QUESTION 5

```
> #Q5:With a seed of 100, 75% training, 25% testing . Build the following mod
els: randomforest, neural
#net, naive bayes and GBM.
> set.seed(100)
> index <- createDataPartition(train_transformed$Species , p=0.75, list=FALSE
)
> trainSet <- train_transformed[ index,]
> testSet <- train_transformed[-index,]
> control <- rfeControl(functions = rfFuncs,
+                       method = "repeatedcv",
+                       repeats = 3,
+                       verbose = FALSE)
> outcomeName<-'Species'
> predictors<-names(trainSet)[!names(trainSet) %in% outcomeName]
>
>
> Loan_Pred_Profile <- rfe(trainSet[,predictors], trainSet[,outcomeName],
+                          rfeControl = control)
There were 50 or more warnings (use warnings() to see the first 50)
> Loan_Pred_Profile

Recursive feature selection

Outer resampling method: Cross-Validated (10 fold, repeated 3 times)

Resampling performance over subset size:

 Variables   RMSE Rsquared    MAE RMSESD RsquaredSD   MAESD Selected
         4 0.6525   0.4420 0.5358 0.1276     0.2420 0.09025
         8 0.6473   0.4671 0.5220 0.1379     0.2663 0.10089        *
        14 0.6536   0.4539 0.5298 0.1345     0.2662 0.09616

The top 5 variables (out of 8):
   CN, Mass, segmented, d13C, Diameter

> predictors<-c("CN", "Mass", "segmented", "d13C", "Diameter")
> model_rf<-train(trainSet[,predictors],trainSet[,outcomeName],method='rf', i
mportance=T)
There were 50 or more warnings (use warnings() to see the first 50)
> print(model_rf)
Random Forest

83 samples
14 predictors

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 83, 83, 83, 83, 83, 83, ...
Resampling results across tuning parameters:
```

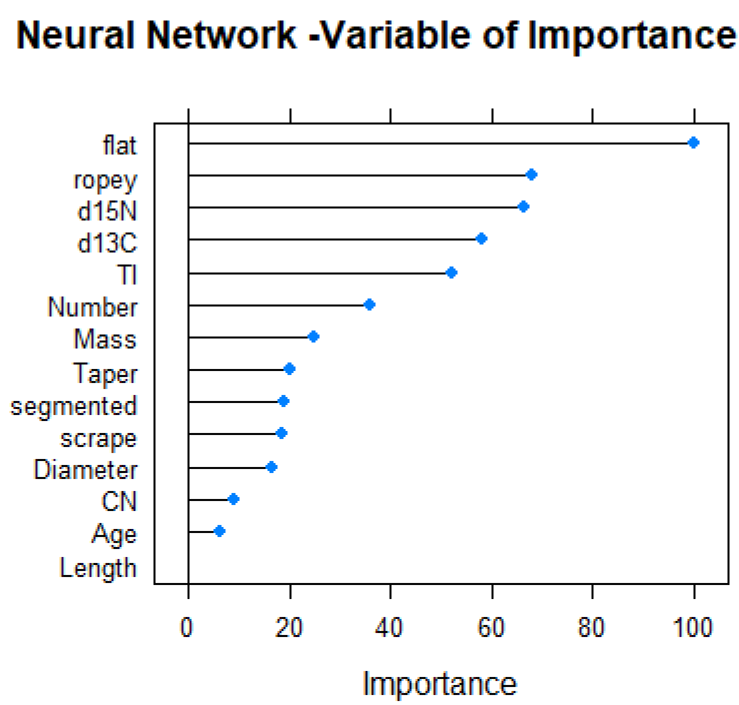
```
  mtry  RMSE       Rsquared   MAE
   2    0.6599280  0.3686107  0.5423345
   8    0.6597693  0.3662098  0.4854045
  14    0.6861925  0.3406830  0.4890397

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 8.
> #b)Ploting Variable of Importance for the predictions
> varImp(object=model_rf)
rf variable importance

          Overall
CN        100.000
Mass       57.487
Diameter   37.207
d13C       34.576
segmented  33.087
d15N       25.609
flat       24.796
Age        21.044
Taper      16.180
scrape     15.485
TI         15.331
Number     11.728
ropey       9.366
Length      0.000
> plot(varImp(object=model_rf), main= "Random Forest -Variable of Importance”
)
```




```
> model_rf2<-train(trainSet[,predictors],trainSet[,outcomeName],method='nnet'
)
> print(model_rf2)
Neural Network

83 samples
14 predictors
```

```
No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 83, 83, 83, 83, 83, 83, ...
Resampling results across tuning parameters:

  size  decay  RMSE      Rsquared    MAE
  1     0e+00  1.094372         NaN  0.7438447
  1     1e-04  1.094386  0.08934224  0.7438715
  1     1e-01  1.099033  0.17657025  0.7518263
  3     0e+00  1.094372         NaN  0.7438447
  3     1e-04  1.094392  0.06109625  0.7438911
  3     1e-01  1.098055  0.18614202  0.7505954
  5     0e+00  1.094372         NaN  0.7438447
  5     1e-04  1.094392  0.06514163  0.7438952
  5     1e-01  1.097430  0.19871418  0.7494915

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were size = 1 and decay = 0.

> #b)Ploting Variable of Importance for the predictions
> plot(varImp(object=model_rf2), main= "Neural Network -Variable of Importanc
e")
> confusionMatrix(predictions,testSet["Species"])
# Reference
# Prediction 0 1
#0 13 0
#1 27 1
table(predictions)
predictions
          1.0225           1.0443           1.0935 1.10273333333333 1.1334666
6666667           1.1634
               1                1                1                1
1                1
1.24006666666667 1.24416666666667           1.2731 1.32193333333333
1.3649 1.41943333333333
               1                1                1                1
1                1
          1.4666           1.5074 1.59856666666667           1.7113 1.8359666
6666667           1.8681
               1                1                1                1
1                1
          1.9892 2.08636666666667           2.1353 2.27113333333333 2.2964666
6666667 2.48036666666667
               1                1                1                1
1                1
          2.6934            2.699 2.76353333333333
               1                1                1
> testSet["Species"]
    Species
9         1
13        1
14        3
20        3
24        1
29        1
34        1
35        3
36        3
57        1
58        1
68        3
75        1
80        1
```

82	2
84	1
86	1
91	3
92	1
94	1
98	2
100	2
101	2
106	2
108	2
114	2
122	2

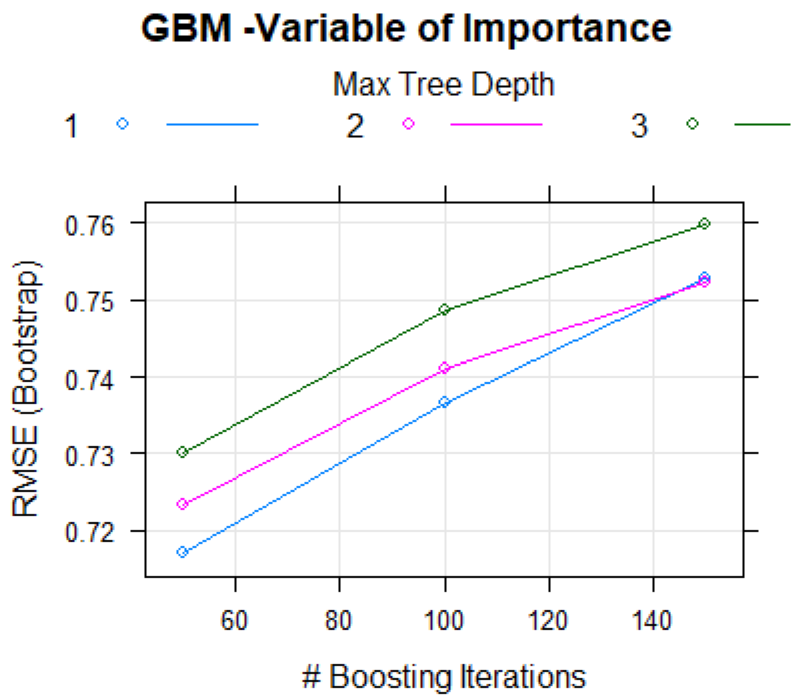



```
> model_rf3<-train(trainSet[,predictors],trainSet[,outcomeName],method='nb')
Error: wrong model type for regression
> print(model_rf3)
Error in print(model_rf3) : object 'model_rf3' not found

>
```

```
#b)Ploting Variable of Importance for the predictions
plot(model_rf3, main= "Neural Network -Variable of Importance")
predictions<-predict.train(object=model_rf3,testSet[,predictors],type="raw")
table(predictions)
predictions
testSet[,outcomeName]<-testSet[,outcomeName].asfactor
#c)confusion matrix
confusionMatrix(predictions,testSet[,outcomeName])

#Using GBM

….
Iter   TrainDeviance   ValidDeviance   StepSize   Improve
     1        0.4867             nan     0.1000    0.0200
```

```
     2        0.4647             nan     0.1000    0.0185
     3        0.4358             nan     0.1000   -0.0034
     4        0.4154             nan     0.1000    0.0123
     5        0.3955             nan     0.1000    0.0127
     6        0.3860             nan     0.1000    0.0041
     7        0.3697             nan     0.1000    0.0115
     8        0.3613             nan     0.1000    0.0043
     9        0.3495             nan     0.1000    0.0064
    10        0.3406             nan     0.1000   -0.0044
    20        0.2593             nan     0.1000    0.0009
    40        0.1727             nan     0.1000   -0.0026
    60        0.1240             nan     0.1000   -0.0010
    80        0.0964             nan     0.1000   -0.0002
   100        0.0792             nan     0.1000   -0.0028
   120        0.0636             nan     0.1000   -0.0011
   140        0.0507             nan     0.1000   -0.0010
   150        0.0453             nan     0.1000    0.0003

Iter   TrainDeviance   ValidDeviance   StepSize   Improve
     1        0.6333             nan     0.1000    0.0321
     2        0.6095             nan     0.1000    0.0062
     3        0.5974             nan     0.1000    0.0082
     4        0.5566             nan     0.1000    0.0323
     5        0.5353             nan     0.1000    0.0134
     6        0.5069             nan     0.1000    0.0150
     7        0.4893             nan     0.1000    0.0095
     8        0.4810             nan     0.1000    0.0023
     9        0.4702             nan     0.1000   -0.0024
    10        0.4558             nan     0.1000    0.0123
    20        0.3689             nan     0.1000   -0.0050
    40        0.3159             nan     0.1000   -0.0068
    50        0.3018             nan     0.1000   -0.0042

There were 12 warnings (use warnings() to see them)
> print(model_rf4)
Stochastic Gradient Boosting

83 samples
14 predictors

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 83, 83, 83, 83, 83, 83, ...
Resampling results across tuning parameters:

  interaction.depth  n.trees  RMSE       Rsquared   MAE
  1                   50      0.7169965  0.2826519  0.5603349
  1                  100      0.7364959  0.2628949  0.5705218
  1                  150      0.7528796  0.2449733  0.5827395
  2                   50      0.7233663  0.2775854  0.5565163
  2                  100      0.7410031  0.2590470  0.5711205
  2                  150      0.7521419  0.2480189  0.5797570
  3                   50      0.7300856  0.2685561  0.5614215
  3                  100      0.7484946  0.2524825  0.5730021
  3                  150      0.7596834  0.2410698  0.5834745

Tuning parameter 'shrinkage' was held constant at a value of 0.1
Tuning parameter 'n.minobsinnode' was
 held constant at a value of 10
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were n.trees = 50, interaction.depth = 1,
shrinkage = 0.1
 and n.minobsinnode = 10.
> #b)Ploting Variable of Importance for the predictions
```

```
> plot(model_rf4, main= "Neural Network -Variable of Importance")
```




```
> table(predictions)
predictions
0.961479934616743  1.02333347995371  1.02384115130274  1.11624873870248  1.13
003302711647  1.20863322767127 
                1                 1                 1                 1                 
1                 1 
 1.29786031511647  1.36495156253413  1.39444255863322  1.40068086263496  1.51
102916860124  1.55181238176809 
                1                 1                 1                 1                 
1                 1 
 1.55289533050491  1.62597273258745  1.65548093959087  1.68833362364427  1.94
567738843197  1.98404036080666 
                1                 1                 1                 1                 
1                 1 
  2.0380603483786  2.21593794988388   2.3700011726071  2.41763881015293  2.42
308108463865  2.53560728948833 
                1                 1                 1                 1                 
1                 1 
 2.57117619159293  2.60578296977321  2.74723277003618 
```

```
#Q7 Tune the GBM model using tune length = 20 and: a) print the model summary and b)

plot the

#models. (20 points)
```

```
fitControl <- trainControl(
  method = "repeatedcv",
  number = 20,
  repeats = 20)


modelLookup(model='gbm')


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))

model_gbm<-train(trainSet[,predictors],trainSet[,outcomeName],method='gbm',trControl=fitControl,tuneGrid=grid)


# a) summarizing the model
print(model_gbm)
# b) Plot the models
```

```
plot(model_gbm)
```

```
> table(predictions)
#8. Using GGplot and gridExtra to plot all variable of importance plots into
one single plot. (10

#points)

library("ggplot2")

library("maps")

library("magrittr")


test_data <- data.frame("RandomForest" =model_rf, "NNet" = model_rf2,
"Naive_B" = model_rf3, "GBM" =model_rf4)

ggplot(test_data)

#9. Which model performs the best? and why do you think this is the case? Can
we accurately

#predict species on this dataset? (10 points)
```