Learning outcomes

After solving these exercises, you should be able to understand the following:

1. Applying the Random Forest and Adaboost algorithms to solve classification problems.
2. Applying stacking techniques
3. Interpreting the results generated from each algorithm in R.
4. Comparison of the model performance in terms of precision, recall and accuracy

Random Forest: Hepatitis Dataset

The hepatitis dataset has 20 variables and 155 records. Use “target” as target variable. Goal is to determine whether the person will live or die.

1. Import the data into R

```
data = read.table('hepatitis.txt', header=F, dec='.',
        col.names=c('target','age','gender','steroid','antivirals',
            'fatigue','malaise','anorexia','liverBig',
            'liverFirm','spleen','spiders','ascites',
            'varices','bili','alk','sgot','albu','protime',
            'histology'),
        na.strings=c('?'), sep=',')
```

2. Study dataset

```
 # Understand the
    data str(data)
    summary(data)
    table(data$target)
    str(data$target) # 1: Die; 2: Live
  # Convert 1s and 2s into 1s and 0s
      data$target= ifelse(data$target==1, 1, 0 ) # 1: Die(+ve); 0: Live (-ve)
```

3. Using domain knowledge separate categorical and numeric attributes. Convert them into appropriate type.

```
 num_Attr = c("age", "bili", "alk", "sgot", "albu",
"protime") cat_Attr = setdiff(names(data), num_Attr)


# Seperate numerical and categorical variables and convert them into appropriate type
data = data.frame(sapply(data,as.character))
cat_Data = data.frame(sapply(data[,cat_Attr], as.factor))
num_Data = data.frame(sapply(data[,num_Attr], as.numeric))
```

data = cbind(num_Data, cat_Data)

4. Fill the missing values using knnImputation.

```
sum(is.na(data))
data = knnImputation(data = data, k = 5)
```

5. Split dataset into train and test

```
set.seed(123)
train_RowIDs = sample(1:nrow(data), nrow(data)*0.7)
train_Data = data[train_RowIDs,]
test_Data = data[-train_RowIDs,]
```

6. Build the classification model using randomForest

```
model = randomForest(target ~ ., data=train_Data,
            keep.forest=TRUE, ntree=50)
```

7. View results and understand important attributes

```
# Print and understand the model
print(model)

# Important attributes
model$importance
round(importance(model), 2)
```

8. View results and understand important attributes

```
# plot (directly prints the important attributes)
varImpPlot(model)
```

9. Predict on Train and Test datasets and calculate accuracy.

```
# Predict on Train data
pred_Train = predict(model, train_Data[,setdiff(names(train_Data),"target")],
        type="response", norm.votes=TRUE)
# Build confusion matrix and find accuracy
cm_Train = table("actual"= train_Data$target, "predicted" =
pred_Train); accu_Train= sum(diag(cm_Train))/sum(cm_Train)

# Predicton Test Data
pred_Test = predict(model, test_Data[,setdiff(names(test_Data),"target")],
      type="response", norm.votes=TRUE)

# Build confusion matrix and find accuracy
cm_Test = table("actual"= test_Data$target, "predicted" =
pred_Test); accu_Test= sum(diag(cm_Test))/sum(cm_Test)
```

10. Extract and store the important variables obtained from Random Forest model. rf_Imp_Attr = data.frame(model$importance)
rf_Imp_Attr = data.frame(row.names(rf_Imp_Attr),rf_Imp_Attr[,1])

```
colnames(rf_Imp_Attr) = c('Attributes', 'Importance')
rf_Imp_Attr = rf_Imp_Attr[order(rf_Imp_Attr$Importance, decreasing = TRUE),]
```

11. Select the important attributes and follow the aforementioned steps to build the model and calculate the accuracy.

```
top_Imp_Attr = as.character(rf_Imp_Attr$Attributes[1:9])
```

Adaboost: Universal Bank Dataset

The Universal Bank dataset has 14 variables and 5000 records. Use “Personal.Loan” as target variable.

1. Import the data into R

```
library(vegan)
library(dummies)
library(ada)
attr = c('id', 'age', 'exp', 'inc', 'zip',
  'family', 'ccavg', 'edu', 'mortgage',
  'loan', 'securities', 'cd', 'online', 'cc')
# Read the data using csv file
data = read.csv(file = "UniversalBank.csv", header = TRUE, col.names = attr)
```

2. Drop ID & ZIP Code, exp

```
drop_Attr = c("id", "zip", "exp")
attr = setdiff(attr, drop_Attr)
data = data[, attr]
```

3. Using domain knowledge separate categorical and numeric attributes. Convert them into appropriate types.

```
cat_Data <- data.frame(sapply(data[,cat_Attr], as.factor))
num_Data <- data.frame(sapply(data[,num_Attr], as.numeric))

data = cbind(num_Data, cat_Data)
```

4. Standardize numerical data using range method

```
cla_Data = decostand(data[,ind_Num_Attr], "range")
```

5. Convert all categorical attributes into numeric

```
# Using dummy function, convert education and family categorical attributes into numeric attributes
edu = dummy(data$edu)
family = dummy(data$family)
cla_Data = cbind(cla_Data, edu, family)
ind_Cat_Attr = setdiff(ind_Cat_Attr, c("edu",
"family")) rm(edu, family)
# Using as.numeric function, convert remaining categorical attributes into numeric attributes
```

```
cla_Data = cbind(cla_Data, sapply(data[,ind_Cat_Attr],
as.numeric)) rm(ind_Cat_Attr)
ind_Attr = names(cla_Data)
```

6. Append the target attribute

```
cla_Data = cbind(cla_Data, loan=data[,"loan"])
```

7. Split the data into train, test data sets

```
set.seed(123)

train_RowIDs = sample(1:nrow(cla_Data),
nrow(cla_Data)*0.6) train_Data = cla_Data[train_RowIDs,]
test_Data = cla_Data[-train_RowIDs,] 8.
```

Build the classification model using Adaboost:

```
model = ada(x = train_Data[,ind_Attr],
        y = train_Data$loan,
        iter=20, loss="logistic")
```

9. Predict the values using model on train and test data sets, and calculate accuracy.

```
# Predict on train data
pred_Train = predict(model, train_Data[,ind_Attr])

# Build confusion matrix and find accuracy
cm_Train = table(train_Data$loan, pred_Train)
accu_Train= sum(diag(cm_Train))/sum(cm_Train)
rm(pred_Train, cm_Train)

# Predict on test data
pred_Test = predict(model, test_Data[,ind_Attr])

# Build confusion matrix and find accuracy
cm_Test = table(test_Data$loan, pred_Test)
accu_Test= sum(diag(cm_Test))/sum(cm_Test)
rm(pred_Test, cm_Test)
```

Stacking: Universal Dataset

The Universal Bank dataset has 14 variables and 5000 records. Use “Personal.Loan” as target variable.

1. Import the data into R

```
library(vegan)
library(infotheo)
library(C50)
library(rpart)

# Set  working directory
setwd("C:/Users/jeevan/Desktop/Ensemble/")

attr = c('id', 'age', 'exp', 'inc', 'zip', 'family',
         'ccavg', 'edu', 'mortgage', 'loan',
         'securities', 'cd', 'online', 'cc')

# Read the data from csv file
data = read.csv(file = "UniversalBank.csv",
                header = TRUE, col.names = attr)
```

2. Drop ID & ZIP Code, exp

```
drop_Attr = c("id", "zip", "exp")
attr = setdiff(attr, drop_Attr)
data = data[, attr]
```

3. Convert the attributes to appropriate types.

```
cat_Attr = c("family", "edu", "securities",
             "cd", "online", "cc", "loan")
num_Attr = setdiff(attr, cat_Attr)

cat_Data = data.frame(sapply(data[,cat_Attr], as.factor))
num_Data = data.frame(sapply(data[,num_Attr], as.numeric))
```

4 . Using Equal Frequency Convert numeric attributes into categorical.

```
num_2_Cat_Data = data.frame(sapply(data[,num_Attr],
                                   function(x){discretize(x, disc = "equalfreq",
                                                          nbins = 4)}))
names(num_2_Cat_Data) = num_Attr

num_2_Cat_Data = data.frame(sapply(num_2_Cat_Data, as.factor))

data = cbind(num_2_Cat_Data, cat_Data)
rm(cat_Data, num_Data, num_2_Cat_Data, cat_Attr, num_Attr)
```

5. Split into Train and Test

```
set.seed(123)

train_RowIDs = sample(1:nrow(data), nrow(data)*0.7)
train_Data = data[train_RowIDs,]
test_Data = data[-train_RowIDs,]
```

6 . Build several classification models

```
#-----------------Ensemble:Stacking--------------------

# Build CART model on the training dataset
cart_Model = rpart(loan ~ ., train_Data, method = "class")
summary(cart_Model)

# Build C5.0 model on the training dataset
c50_Model = C5.0(loan ~ ., train_Data, rules = T)
summary(c50_Model)

# Build Logistic regression on the training dataset
glm_Model = glm(loan ~ ., train_Data, family = binomial)
summary(glm_Model)
```

7. Predicting on train dataset

```
# Using CART Model predict on train data
cart_Train = predict(cart_Model, train_Data, type = "vector")
table(cart_Train)

# if we choose type=vector, then replace 1 with 0 and 2 with 1
cart_Train = ifelse(cart_Train == 1, 0, 1)
table(cart_Train)

# Using C5.0 Model predicting with the train dataset
c50_Train = predict(c50_Model, train_Data, type = "class")
c50_Train = as.vector(c50_Train)
table(c50_Train)

# Using GLM Model predicting on train dataset
glm_Train = predict(glm_Model, train_Data, type = "response")
#it gives probabilities, so we #need to convert to 1's and 0's;
# if >0.5 show as 1 or else show as 0.
glm_Train = ifelse(glm_Train > 0.5, 1, 0)
table(glm_Train)
```

8.Combining the training predictions of all the models.

```
train_Pred_All_Models = data.frame(CART = cart_Train,
                                   C50 = c50_Train,
                                   GLM = glm_Train)
train_Pred_All_Models = data.frame(sapply(train_Pred_All_Models, as.factor))
```

9.View the predictions of each model

```
table(train_Pred_All_Models$CART) #CART
table(train_Pred_All_Models$C50)  #C5.0
table(train_Pred_All_Models$GLM)  #Logistic Regression
table(train_Data$loan) #Original Dataset DV
```

10. Add the original target variable to the dataset.

```
train_Pred_All_Models = cbind(train_Pred_All_Models, loan = train_Data$loan)
```

11.Ensemble the model with GLM as Meta Learner

```
ensemble_Model = glm(loan ~ ., train_Pred_All_Models, family.= binomial)
summary(ensemble_Model)
```

12.Check the ensembled model on train data

```
ensemble_Train = predict(ensemble_Model, train_Pred_All_Models,
                         type = "response")
ensemble_Train = ifelse(ensemble_Train > 0.5, 1, 0)
table(ensemble_Train)

cm_Ensemble = table(ensemble_Train, train_Pred_All_Models$loan)
sum(diag(cm_Ensemble))/sum(cm_Ensemble)
```

13.Follow the steps from 7 to 12 on the test data and check out the accuracy.

Exercise: Perform the above analysis in case of Regression problem as well.