Learning outcomes

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

1. Applying the Random Forest 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: Cancer Dataset

The dataset represents the breast cancer data set. Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image.

Dataset Description:

Attribute Information:

1) ID number
2) Diagnosis (B/0 = benign ,M/1 = malignant)

Ten real-valued features are computed for each cell nucleus:

a) radius (mean of distances from center to points on the perimeter)
b) texture (standard deviation of gray-scale values)
c) perimeter
d) area
e) smoothness (local variation in radius lengths)
f) compactness (perimeter^2 / area - 1.0
g) concavity (severity of concave portions of the contour)
h) concave points (number of concave portions of the contour)
i) symmetry
j) fractal dimension ("coastline approximation" - 1)

 # R Code

1. Import the cancer_diagnosis.csv data into R
2. Study dataset
3. Convert all features into appropriate data types
5. Split dataset into train and test
6. Build the classification model using randomForest

```
library(randomForest)
model_rf  <- randomForest(target ~ ., data= train_data, ntree=50,mtry = 5)
```

8. View results and understand important attributes

```
print(model_rf)
model_rf $predicted
model_rf $importance
```

9. View results and understand important attributes

```
varImpPlot(model_rf)
```

10. Predict on Train and Test datasets
11. Calculate precision, recall and accuracy

Stacking Technique: Cancer Dataset

1. Use pre-processed data that is applied from step1 – step5 for random forest.

Building different Machine Learning algorithms

#(1) Build rpart model on the training dataset

```
library(rpart)
model_dt <- rpart(Cancer ~ . , train_data)


# Prediction on the train data
preds_train_dt <- predict(model_dt)
preds_train_tree <- ifelse(preds_train_dt[, 1] > preds_train_dt[, 2], 0, 1)


# Prediction on the test data
preds_dt <- predict(model_dt, test_data)
preds_tree <- ifelse(preds_dt[, 1] > preds_dt[, 2], 0, 1)
```

```
confusionMatrix(preds_tree, test_data$Cancer)
```

(2) Build KNN model on the training dataset

```
Library(caret)
# We'll build our KNN model, using the knn3() function from the caret package
model_knn <- knn3(Cancer ~ . , train_data, k = 5)

# Store the predictions on the train data
preds_train_k <- predict(model_knn, train_data)
preds_train_knn <- ifelse(preds_train_k[, 1] > preds_train_k[, 2], 0, 1)

# Prediction on the test data
preds_k <- predict(model_knn, test_data)
preds_knn <- ifelse(preds_k[, 1] > preds_k[, 2], 0, 1)
confusionMatrix(preds_knn, test_data$Cancer)
```

(3) Build bagging rpart model on the training dataset

```
library(ipred)
set.seed(1234)
model_tree_bag <- bagging(Cancer ~ . , data=train_data,nbagg = 10,control =
rpart.control(cp = 0.01, xval = 10))

# Prediction on the train data
preds_train_tree_bag <- predict(model_tree_bag)

# Prediction on the test data
preds_tree_bag <- predict(model_tree_bag, test_data)
confusionMatrix(preds_tree_bag, test_data$Cancer)
```

```
# (4) Preparing the train data for stacking model by combining training predictions of Random Forest, KNN, rpart & bagging models
    train_preds_df <- data.frame(rf = preds_train_rf, knn = preds_train_knn,
                                 tree = preds_train_tree, tree_bag = preds_train_tree_bag,
                                 Cancer = train_data$Cancer)
    # convert the target variable into a factor
    train_preds_df$Cancer <- as.factor(as.character(train_preds_df$Cancer))

# (5)  Check if there are any correlations in the data
    # Use the sapply() function to convert all the variables other than the target variable into a numeric type
    numeric_st_df <- sapply(train_preds_df[, !(names(train_preds_df) %in% "Cancer")], function(x) as.numeric(as.character(x)))
    cor(numeric_st_df)

# (6) The features are highly correlated, Apply PCA on the data
    # The outputs of the various models are extremely correlated let's use PCA to overcome the multicolinearity and identify the number of components to be consider
    pca_stack <- prcomp(numeric_st_df, scale = F)
    summary(pca_stack)
    # Transform the data into the principal components
    predicted_stack <- as.data.frame(predict(pca_stack, numeric_st_df))[1:2]

    # Prepare the data frame with PCA components and the target variable (Cancer)
    stacked_df <- data.frame(predicted_stack, Cancer = train_preds_df$Cancer)
```

(7) Build GLM Model with as Meta Learner

```
stacked_model <- glm(Cancer ~ . , data = stacked_df,family = "binomial")
```

Preparing the test data for stacking model

(8) Combining test predictions of Random Forest, KNN, rpart & bagging models

```
stack_df_test <- data.frame(rf = preds_rf, knn = preds_knn,
                tree = preds_tree, tree_bag = preds_tree_bag,
                Cancer = test_data$Cancer)
# Convert the target variable into a factor
stack_df_test$Cancer <- as.factor(stack_df_test$Cancer)
```

(9) Getting the principle components on the test data and preparing the final test data with the components

```
# Convert all other variables into numeric
numeric_st_df_test <- sapply(stack_df_test[, !(names(stack_df_test) %in%
"Cancer")],function(x) as.numeric(as.character(x)))
# Getting the principle components on test data
predicted_stack_test <- as.data.frame(predict(pca_stack, numeric_st_df_test))[1:2]
# Combine the target variable along with the PC dataset
stacked_df_test <- data.frame(predicted_stack_test, Cancer =
stack_df_test$Cancer)
```

(10) Check the "glm_ensemble model" on the test data

```
# * Now, apply the stacked model on the above dataframe
preds_st_test <-  predict(stacked_model, stacked_df_test,type = "response")
preds_st_test <- ifelse(preds_st_test > 0.5,"1","0")
```

(11) Evaluate the performance of all the individual model with the stacking model and identify the best model.