Cancer Detection - Predictive Analytics In-class

```
# load the required libraries
library("readxl") # used to read excel files
library("dplyr") # used for data munging
library("FNN") # used for knn regression (knn.reg function)
library("caret") # used for various predictive models
library("class") # for using confusion matrix function
library("rpart.plot") # used to plot decision tree
library("rpart")  # used for Regression tree
library("glmnet") # used for Lasso and Ridge regression
library('NeuralNetTools') # used to plot Neural Networks
library("PRROC") # top plot ROC curve
library("ROCR") # top plot lift curve
library("e1071")
```

1. Classification

1.1 Data loading and transformation

Data Description: https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.names (https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.names)

```
# Load the Breast Cancer data set

cancer_data = read.csv("wdbc.data", header = FALSE)
cancer_data$V2 <- as.factor(cancer_data$V2)

# create Y and X data frames
cancer_y = cancer_data %>% pull("V2")
# exclude V1 since its a row number
cancer_x = cancer_data %>% select(-c("V1", "V2"))
```

Create a function that normalises columns since scale for each column might be different.

```
# function to normalize data (0 to 1)
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}
```

```
# Normalize x variables since they are at different scale
cancer_x_normalized <- as.data.frame(lapply(cancer_x, normalize))
```

Create Training and Testing data sets

```
# 75% of the data is used for training and rest for testing
smp_size <- floor(0.75 * nrow(cancer_x_normalized))

# randomly select row numbers for training data set
#seq_len will take one argument and create a sequence from 1 to that argument
train_ind <- sample(seq_len(nrow(cancer_x_normalized)), size = smp_size)

# creating test and training sets for x
cancer_x_train <- cancer_x_normalized[train_ind, ]
cancer_x_test <- cancer_x_normalized[-train_ind, ]

# creating test and training sets for y
cancer_y_train <- cancer_y[train_ind]
cancer_y_test <- cancer_y[-train_ind]

# Create an empty data frame to store results from different models
clf_results <- data.frame(matrix(ncol = 5, nrow = 0))
names(clf_results) <- c("Model", "Accuracy", "Precision", "Recall", "F1")
```

Cross validation

It is a technique to use same training data but some portion of it for training and rest for validation of model. This technique reduces chances of overfitting

Hyperparamter tuning

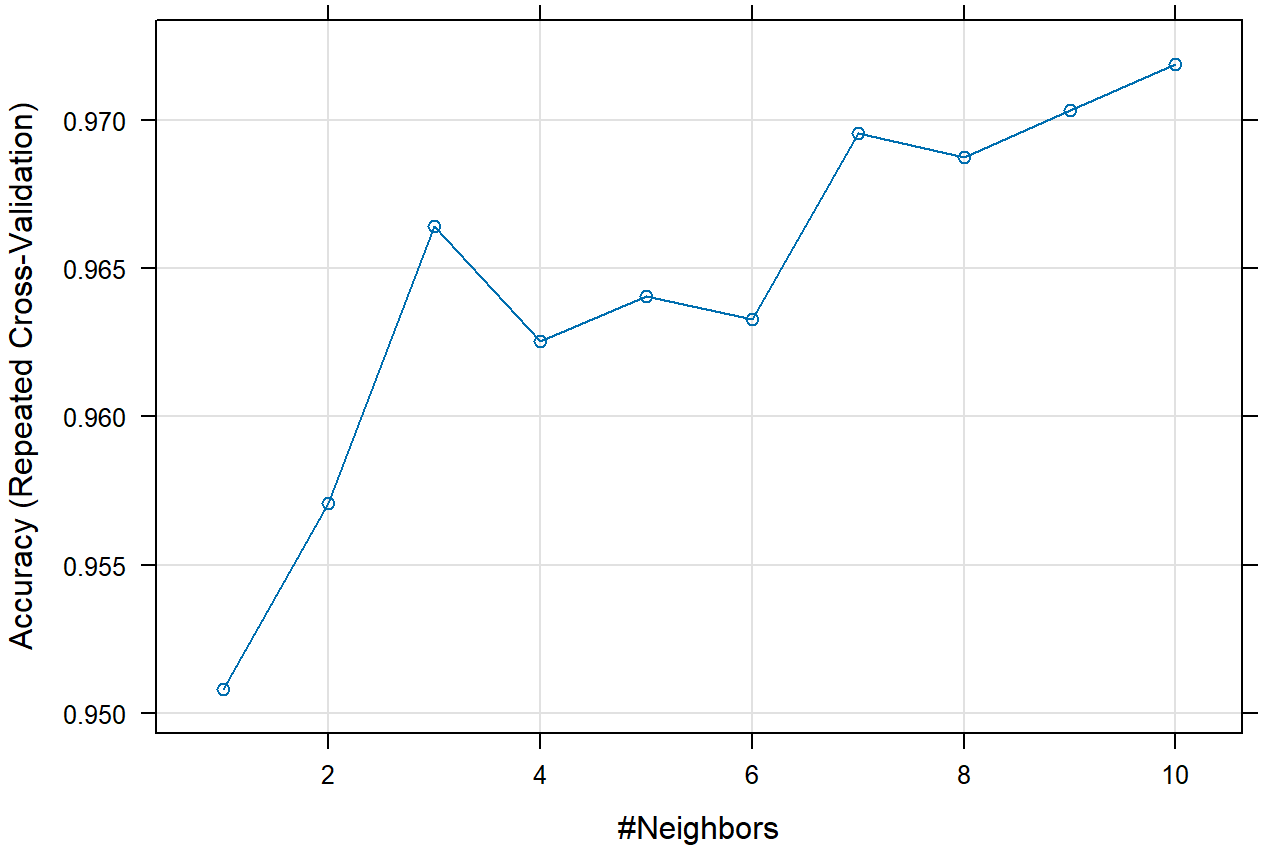
We provide a list of hyperparameters to train the model. This helps in identifying best set of hyperparameters for a given model like Decision tree. **train** function in caret library automatically stores the information of the best model and its hyperparameters.

1.2 KNN Classification

```
# Cross validation
cross_validation <- trainControl(## 10-fold CV
                                 method = "repeatedcv",
                                 number = 10,
                                 ## repeated three times
                                 repeats = 3)
# Hyperparamter tuning
# k = number of nrearest neighbours
Param_Grid <-  expand.grid( k = 1:10)

# fit the model to training data
knn_clf_fit <- train(cancer_x_train,
                     cancer_y_train,
                     method = "knn",
                     tuneGrid = Param_Grid,
                     trControl = cross_validation )

# check the accuracy for different models
knn_clf_fit
```

```
## k-Nearest Neighbors
##
## 426 samples
##  30 predictor
##   2 classes: 'B', 'M'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 383, 383, 384, 383, 384, 383, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.9507903  0.8931570
##    2  0.9570657  0.9064020
##    3  0.9664066  0.9260325
##    4  0.9625306  0.9176577
##    5  0.9640626  0.9209346
##    6  0.9632874  0.9192139
##    7  0.9695451  0.9326250
##    8  0.9687330  0.9309484
##    9  0.9703388  0.9344875
##   10  0.9718883  0.9379386
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 10.
```

```
# Plot accuracies for different k values
plot(knn_clf_fit)
```




```
# print the best model
print(knn_clf_fit$finalModel)
```

```
## 10-nearest neighbor model
## Training set outcome distribution:
##
##   B   M
## 273 153
```

```
# Predict on test data
knnPredict <- predict(knn_clf_fit, newdata = cancer_x_test)
```

```
# Print Confusion matrix, Accuracy, Sensitivity etc
confusionMatrix(knnPredict, cancer_y_test, positive = "M", mode = "prec_recall")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 83  6
##          M  1 53
##
##                Accuracy : 0.951
##                  95% CI : (0.9017, 0.9801)
##     No Information Rate : 0.5874
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8977
##
##  Mcnemar's Test P-Value : 0.1306
##
##               Precision : 0.9815
##                  Recall : 0.8983
##                      F1 : 0.9381
##              Prevalence : 0.4126
##          Detection Rate : 0.3706
##    Detection Prevalence : 0.3776
##       Balanced Accuracy : 0.9432
##
##        'Positive' Class : M
##
```

```
# Add results into clf_results dataframe
x1 <- confusionMatrix(knnPredict, cancer_y_test,positive = "M")[["overall"]]
y1 <- confusionMatrix(knnPredict, cancer_y_test, positive = "M")[["byClass"]]

clf_results[nrow(clf_results) + 1,] <-  list(Model = "KNN",
                                             Accuracy = round(x1[["Accuracy"]],3),
                                            Precision = round(y1[["Precision"]],3),
                                            Recall = round(y1[["Recall"]],3),
                                            F1 = round (y1[["F1"]],3))
# Print Accuracy and F1 score

cat("Accuarcy is ", round(x1[["Accuracy"]],3), "and F1 is ", round (y1[["F1"]],3)  )
```

```
## Accuarcy is  0.951 and F1 is  0.938
```

1.3 Decision Tree Classification

```
# Cross validation
cross_validation <- trainControl(## 10-fold CV
                                 method = "repeatedcv",
                                 number = 10,
                                 ## repeated three times
                                 repeats = 3)
# Hyperparamter tuning
# maxdepth =  the maximum depth of the tree that will be created or
# the length of the longest path from the tree root to a leaf.

Param_Grid <-  expand.grid(maxdepth = 2:10)
modelLookup("rpart2") #to know which hyperparameter is associated with which model such as 'knn'
or 'rpart2'
```

	model	parameter	label	forReg	forClass	probModel
	<chr>	<chr>	<chr>	<lgl>	<lgl>	<lgl>
1	rpart2	maxdepth	Max Tree Depth	TRUE	TRUE	TRUE

1 row

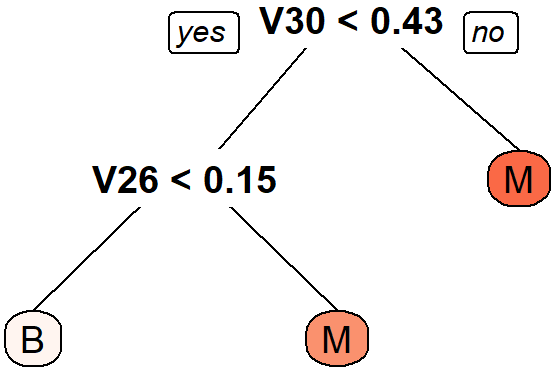
```
dtree_fit <- train(cancer_x_train,
                   cancer_y_train,
                   method = "rpart2",
                   # split - criteria to split nodes
                   parms = list(split = "gini"),
                  tuneGrid = Param_Grid,
                   trControl = cross_validation,
                  # preProc -  perform listed pre-processing to predictor dataframe
                   preProc = c("center", "scale"))

# check the accuracy for different models
dtree_fit
```

```
## CART
##
## 426 samples
##  30 predictor
##   2 classes: 'B', 'M'
##
## Pre-processing: centered (30), scaled (30)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 383, 384, 384, 384, 383, 383, ...
## Resampling results across tuning parameters:
##
##   maxdepth  Accuracy   Kappa
##    2        0.9403604  0.8705620
##    3        0.9388285  0.8671695
##    4        0.9388285  0.8671695
##    5        0.9388285  0.8671695
##    6        0.9388285  0.8671695
##    7        0.9388285  0.8671695
##    8        0.9388285  0.8671695
##    9        0.9388285  0.8671695
##   10        0.9388285  0.8671695
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was maxdepth = 2.
```

```
# print the final model
dtree_fit$finalModel
```

```
## n= 426
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 426 153 B (0.64084507 0.35915493)
##   2) V30< 0.4298199 289  23 B (0.92041522 0.07958478)
##     4) V26< 0.1483152 271   8 B (0.97047970 0.02952030) *
##     5) V26>=0.1483152 18   3 M (0.16666667 0.83333333) *
##   3) V30>=0.4298199 137   7 M (0.05109489 0.94890511) *
```

```
# Plot decision tree
prp(dtree_fit$finalModel, box.palette = "Reds", tweak = 1.2)
```




```
# Predict on test data
dtree_predict <- predict(dtree_fit, newdata = cancer_x_test)
```

```
# Print Confusion matrix, Accuarcy, Sensitivity etc
confusionMatrix(dtree_predict,  cancer_y_test, positive = "M", mode="prec_recall" )
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 78  5
##          M  6 54
##
##                Accuracy : 0.9231
##                  95% CI : (0.8665, 0.961)
##     No Information Rate : 0.5874
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8417
##
##  Mcnemar's Test P-Value : 1
##
##               Precision : 0.9000
##                  Recall : 0.9153
##                      F1 : 0.9076
##              Prevalence : 0.4126
##          Detection Rate : 0.3776
##    Detection Prevalence : 0.4196
##       Balanced Accuracy : 0.9219
##
##        'Positive' Class : M
##
```

```
# Add results into clf_results dataframe
x2 <- confusionMatrix(dtree_predict,  cancer_y_test, positive = "M" )[["overall"]]
y2 <- confusionMatrix(dtree_predict,  cancer_y_test, positive = "M" )[["byClass"]]

clf_results[nrow(clf_results) + 1,] <-  list(Model = "Decision Tree",
                                             Accuracy = round (x2[["Accuracy"]],3),
                                            Precision = round (y2[["Precision"]],3),
                                            Recall = round (y2[["Recall"]],3),
                                            F1 = round (y2[["F1"]],3))

# Print Accuracy and F1 score

cat("Accuarcy is ", round(x2[["Accuracy"]],3), "and F1 is ", round (y2[["F1"]],3)  )
```

```
## Accuarcy is  0.923 and F1 is  0.908
```

1.4 Logistic regression

```
glm_fit <- train(cancer_x_train,
                 cancer_y_train,
                 method = "glm",
                 family = "binomial",
                 preProc = c("center", "scale"))
```

```
# Predict on test data
glm_predict <- predict(glm_fit, newdata = cancer_x_test)
glm_predict_prob <- predict(glm_fit, newdata = cancer_x_test, type="prob")
```

convert probability outcome into categorical outcome

```
y_pred_num <- ifelse(glm_predict_prob[1] > 0.5, "B","M")
#column 1 is prob("B)
```

```
# Print Confusion matrix, Accuarcy, Sensitivity etc

confusionMatrix(as.factor(y_pred_num), as.factor(cancer_y_test), positive = "M", mode="prec_reca
ll")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 80  6
##          M  4 53
##
##                Accuracy : 0.9301
##                  95% CI : (0.8752, 0.966)
##     No Information Rate : 0.5874
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.855
##
##  Mcnemar's Test P-Value : 0.7518
##
##               Precision : 0.9298
##                  Recall : 0.8983
##                      F1 : 0.9138
##              Prevalence : 0.4126
##          Detection Rate : 0.3706
##    Detection Prevalence : 0.3986
##       Balanced Accuracy : 0.9253
##
##        'Positive' Class : M
##
```

```
# Add results into clf_results dataframe
x3 <- confusionMatrix(as.factor(y_pred_num), as.factor(cancer_y_test), positive = "M")[["overal
l"]]
y3 <- confusionMatrix(as.factor(y_pred_num), as.factor(cancer_y_test), positive = "M")[["byClas
s"]]

clf_results[nrow(clf_results) + 1,] <-  list(Model = "Logistic Regression",
                                             Accuracy = round (x3[["Accuracy"]],3),
                                            Precision = round (y3[["Precision"]],3),
                                            Recall = round (y3[["Recall"]],3),
                                            F1 = round (y3[["F1"]],3))

# Print Accuracy and F1 score
cat("Accuarcy is ", round(x3[["Accuracy"]],3), "and F1 is ", round (y3[["F1"]],3)  )
```

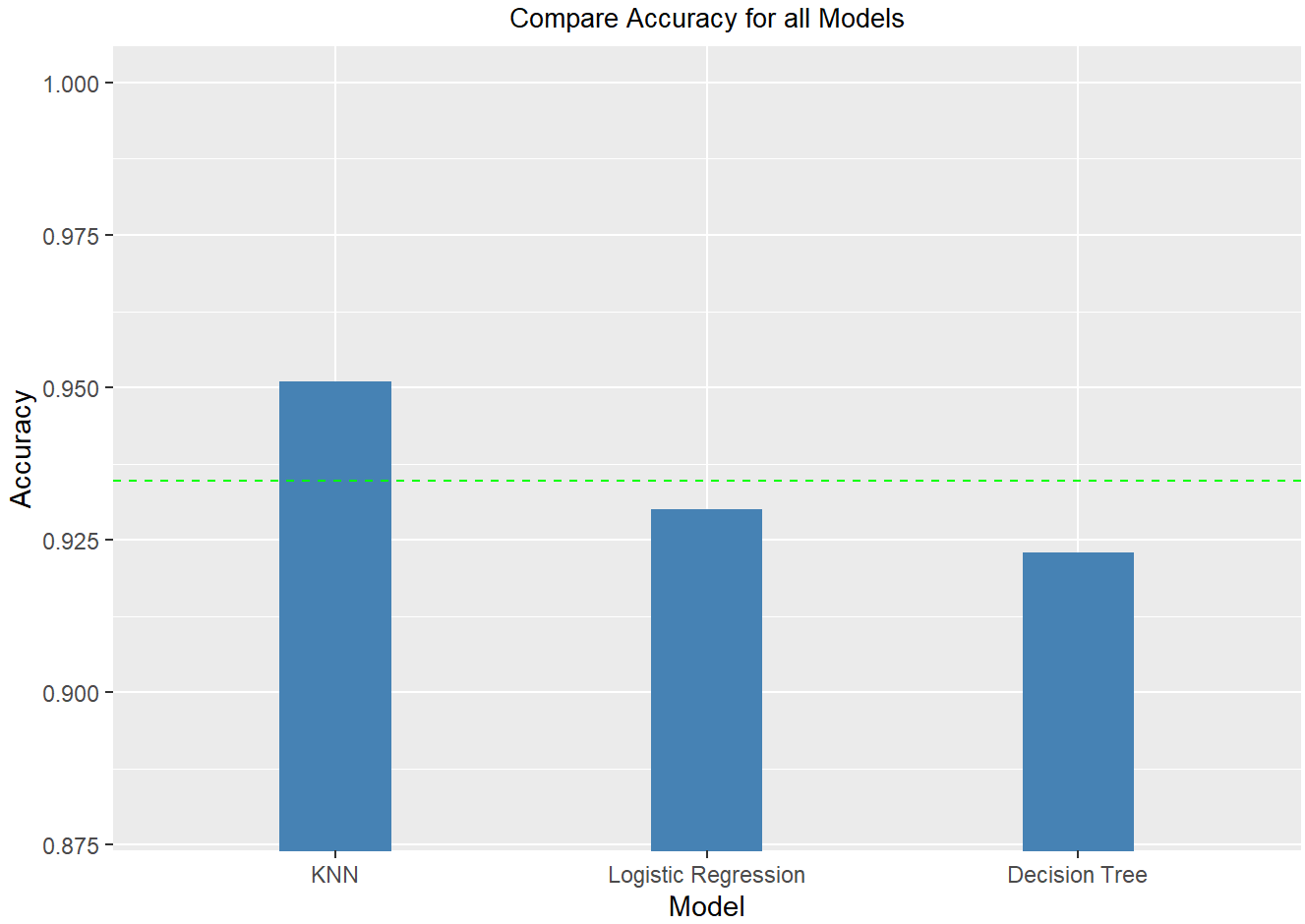
```
## Accuarcy is  0.93 and F1 is  0.914
```

Compare Accuracy for all Classification models

```
print(clf_results)
```

```
##                 Model Accuracy Precision Recall    F1
## 1                 KNN    0.951     0.981  0.898 0.938
## 2       Decision Tree    0.923     0.900  0.915 0.908
## 3 Logistic Regression    0.930     0.930  0.898 0.914
```

```
# Plot accuracy for all the Classification Models

ggplot(clf_results %>% arrange(desc(Accuracy)) %>%
       mutate(Model=factor(Model, levels=Model) ),
       aes(x = Model, y = Accuracy)) +
  geom_bar(stat = "identity" , width=0.3, fill="steelblue") +
  coord_cartesian(ylim = c(0.88, 1)) +
  geom_hline(aes(yintercept = mean(Accuracy)),
             colour = "green",linetype="dashed") +
  ggtitle("Compare Accuracy for all Models") +
  theme(plot.title = element_text(color="black", size=10, hjust = 0.5))
```


Compare Accuracy for all Models
1.000
0.975
0.950
0.925
0.900
0.875
Accuracy
KNN
Logistic Regression
Decision Tree
Model