

Predictive Analytics in-class exercise on Corporate Bankruptcy Detection

- 1. Classification
 - 1.1 Data loading and transformation
 - 1.3 Decision Tree Classification
 - 1.4 Logistic regression
 - 1.5 XGBoost classification
 - 1.6 Neural Network classification
 - 1.8 ROC and Lift curves for all models
 - Visualizing results

```
# 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("tidyverse")
library("skimr")
library(lime)
```

1. Classification

1.1 Data loading and transformation

```
# Load the Corporate Ratios Dataset to predict bankruptcy

corpRatios <- read_excel("CL-bankruptcy.xls",
  sheet = "data", col_types = c("skip",
    "text", "text", "numeric", "numeric",
    "numeric", "numeric", "numeric",
    "numeric", "numeric", "numeric",
    "numeric", "numeric", "numeric",
    "numeric", "numeric", "numeric",
    "numeric", "numeric", "numeric",
    "numeric"))

skim(corpRatios)
```

Data summary

























Name	corpRatios
Number of rows	132
Number of columns	26
Column type frequency:	
character	2
numeric	24
Group variables	
None	

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
---------------	-----------	---------------	-----	-----	-------	----------	------------

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
D	0	1	1	1	0	2	0
YR	0	1	2	2	0	12	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
R1	0	1	0.29	0.52	0.00	0.07	0.15	0.27	4.92	
R2	0	1	0.06	0.12	0.00	0.02	0.03	0.06	1.00	
R3	0	1	0.07	0.08	0.00	0.03	0.05	0.09	0.64	
R4	0	1	0.16	0.21	0.00	0.04	0.08	0.17	1.38	
R5	0	1	0.04	0.17	-1.40	0.01	0.03	0.08	0.46	
R6	0	1	0.05	0.11	-0.60	0.01	0.06	0.10	0.49	
R7	0	1	0.12	0.22	-0.61	0.02	0.09	0.21	1.16	
R8	0	1	8.08	15.38	0.00	2.80	3.84	6.26	113.54	
R9	0	1	1.96	1.18	0.27	1.33	1.59	2.22	9.48	
R10	0	1	0.50	0.88	0.10	0.27	0.40	0.51	10.18	
R11	0	1	0.61	0.21	0.06	0.52	0.65	0.77	0.98	
R12	0	1	0.61	0.23	0.14	0.42	0.62	0.80	1.00	
R13	0	1	0.00	0.20	-2.10	0.00	0.02	0.05	0.34	
R14	0	1	0.01	0.12	-0.63	-0.01	0.03	0.07	0.47	
R15	0	1	0.09	0.32	-0.47	-0.01	0.04	0.14	2.99	
R16	0	1	0.06	0.13	-0.59	0.01	0.05	0.11	0.57	
R17	0	1	0.07	0.12	-0.43	0.02	0.08	0.14	0.50	
R18	0	1	0.21	0.55	-0.45	0.03	0.12	0.29	5.81	
R19	0	1	15.54	25.73	0.76	4.36	6.86	11.69	170.96	
R20	0	1	1.77	1.09	0.06	1.20	1.51	2.11	6.75	
R21	0	1	1.90	1.35	0.67	1.37	1.53	2.16	14.68	
R22	0	1	0.05	0.18	-1.49	0.01	0.05	0.09	0.53	
R23	0	1	0.06	0.13	-0.45	0.02	0.07	0.12	0.50	
R24	0	1	0.19	0.55	-0.44	0.03	0.11	0.24	5.81	

```
# create Y and X data frames
corpRatios_y = corpRatios %>% pull("D") %>% as.factor()
# exclude X1 since its a row number
corpRatios_x = corpRatios %>% select(-c("D"))
corpRatios_x$YR <- as.factor(corpRatios_x$YR)
```

Create Training and Testing data sets

```

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

# randomly select row numbers for training data set
train_ind <- sample(seq_len(nrow(corpRatios_x)), size = smp_size)

# creating test and training sets for x
corpRatios_x_train <- corpRatios_x[train_ind, ]
corpRatios_x_test <- corpRatios_x[-train_ind, ]

# creating test and training sets for y
corpRatios_y_train <- corpRatios_y[train_ind]
corpRatios_y_test <- corpRatios_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")

# Create an empty data frame to store TP, TN, FP and FN values
cost_benefit_df <- data.frame(matrix(ncol = 5, nrow = 0))
names(cost_benefit_df) <- c("Model", "TP", "FN", "FP", "TN")

```

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.

1.3 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)

dtree_fit <- train(corpRatios_x_train,
                  corpRatios_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"))

```

[illegible]

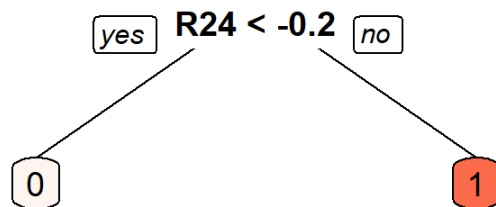
```
# check the accuracy for different models
dtree_fit
```

```
## CART
##
## 99 samples
## 25 predictors
## 2 classes: '0', '1'
##
## Pre-processing: centered (24), scaled (24), ignore (1)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 89, 88, 89, 89, 90, 90, ...
## Resampling results across tuning parameters:
##
##    maxdepth  Accuracy  Kappa
##    2         0.81367   0.6259088
##    3         0.81367   0.6259088
##    4         0.81367   0.6259088
##    5         0.81367   0.6259088
##    6         0.81367   0.6259088
##    7         0.81367   0.6259088
##    8         0.81367   0.6259088
##    9         0.81367   0.6259088
##    10        0.81367   0.6259088
##
## 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= 99
##
## node), split, n, loss, yval, (yprob)
##      * denotes terminal node
##
## 1) root 99 47 1 (0.4747475 0.5252525)
##    2) R24< -0.2001602 45 6 0 (0.8666667 0.1333333) *
##    3) R24>=-0.2001602 54 8 1 (0.1481481 0.8518519) *
```

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



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

```
# Print Confusion matrix, Accuaracy, Sensitivity etc
confusionMatrix(dtree_predict, corpRatios_y_test , positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0   1
##           0 14   2
##           1   5 12
##
##           Accuracy : 0.7879
##           95% CI : (0.6109, 0.9102)
##           No Information Rate : 0.5758
##           P-Value [Acc > NIR] : 0.009148
##
##           Kappa : 0.5777
##
## Mcnemar's Test P-Value : 0.449692
##
##           Sensitivity : 0.8571
##           Specificity : 0.7368
##           Pos Pred Value : 0.7059
##           Neg Pred Value : 0.8750
##           Prevalence : 0.4242
##           Detection Rate : 0.3636
##           Detection Prevalence : 0.5152
##           Balanced Accuracy : 0.7970
##
##           'Positive' Class : 1
##
```

```
# Add results into clf_results dataframe
x2 <- confusionMatrix(dtree_predict, corpRatios_y_test , positive = "1")["overall"]
y2 <- confusionMatrix(dtree_predict, corpRatios_y_test , positive = "1")["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("Accuracy is ", round(x2[["Accuracy"]],3), "and F1 is ", round (y2[["F1"]],3) )
```

```
## Accuracy is 0.788 and F1 is 0.774
```

```
# Add results into cost_benefit_df dataframe for cost benefit analysis
a2 <- confusionMatrix(dtree_predict, corpRatios_y_test )

cost_benefit_df[nrow(cost_benefit_df) + 1,] <- list(Model = "Decision Tree",
                                                    TP = a2[["table"]][4],
                                                    FN = a2[["table"]][3],
                                                    FP = a2[["table"]][2],
                                                    TN = a2[["table"]][1])
```

1.4 Logistic regression

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

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

# one of the factor levels is not in the tes set
#Let find out which row this is
#rowMissingFactor <- which(corpRatios_x_test$YR == "80", arr.ind=TRUE)
#corpRatios_x_test <- corpRatios_x_test %>% filter(YR != "80")

glm_predict_prob <- predict(glm_fit, newdata = corpRatios_x_test, type="prob")
```

convert probability outcome into categorical outcome

```
y_pred_num <- ifelse(glm_predict_prob[,2] > 0.5, 1, 0)
```

```
# Print Confusion matrix, Accuarcy, Sensitivity etc
confusionMatrix(as.factor(y_pred_num), as.factor(corpRatios_y_test), positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0   1
##           0 13   2
##           1   6 12
##
##           Accuracy : 0.7576
##           95% CI : (0.5774, 0.8891)
##           No Information Rate : 0.5758
##           P-Value [Acc > NIR] : 0.02392
##
##           Kappa : 0.5217
##
##  Mcnemar's Test P-Value : 0.28884
##
##           Sensitivity : 0.8571
##           Specificity : 0.6842
##           Pos Pred Value : 0.6667
##           Neg Pred Value : 0.8667
##           Prevalence : 0.4242
##           Detection Rate : 0.3636
##           Detection Prevalence : 0.5455
##           Balanced Accuracy : 0.7707
##
##           'Positive' Class : 1
##
```



```
# Add results into clf_results dataframe
x3 <- confusionMatrix(as.factor(y_pred_num), as.factor(corpRatios_y_test), positive = "1")["overall"]
y3 <- confusionMatrix(as.factor(y_pred_num), as.factor(corpRatios_y_test), positive = "1")["byClass"]

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("Accuracy is ", round(x3[["Accuracy"]],3), "and F1 is ", round (y3[["F1"]],3) )
```

```
## Accuracy is  0.758 and F1 is  0.75
```

```
# Add results into cost_benefit_df dataframe for cost benefit analysis
a3 <- confusionMatrix(as.factor(y_pred_num), as.factor(corpRatios_y_test))

#be careful about accurately pickign up the TP, FN, FP and TN
cost_benefit_df[nrow(cost_benefit_df) + 1,] <- list(Model = "Logistic Regression",
      TP = a3[["table"]][4],
      FN = a3[["table"]][3],
      FP = a3[["table"]][2],
      TN = a3[["table"]][1])
```

1.5 XGBoost classification

```
XG_clf_fit <- train(corpRatios_x_train %>% select(-c("YR")),
      corpRatios_y_train,
      method = "xgbTree",
      preProc = c("center", "scale"))
```

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

```
## [01:18:08] WARNING: amalgamation/./src/c_api/c_api.cc:785: `ntree_limit` is deprecated, use `iteration_range` instead.
## [01:18:08] WARNING: amalgamation/./src/c_api/c_api.cc:785: `ntree_limit` is deprecated, use `iteration_range` instead.
## [01:18:08] WARNING: amalgamation/./src/c_api/c_api.cc:785: `ntree_limit` is deprecated, use `iteration_range` instead.
## [01:18:08] WARNING: amalgamation/./src/c_api/c_api.cc:785: `ntree_limit` is deprecated, use `iteration_range` instead.
## [01:18:08] WARNING: amalgamation/./src/c_api/c_api.cc:785: `ntree_limit` is deprecated, use `iteration_range` instead.
```

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

```
## ##### xgb.Booster
## raw: 44.2 Kb
## call:
##   xgboost::xgb.train(params = list(eta = param$eta, max_depth = param$max_depth,
##     gamma = param$gamma, colsample_bytree = param$colsample_bytree,
##     min_child_weight = param$min_child_weight, subsample = param$subsample),
##     data = x, nrounds = param$nrounds, objective = "binary:logistic")
## params (as set within xgb.train):
##   eta = "0.4", max_depth = "3", gamma = "0", colsample_bytree = "0.6", min_child_weight = "1", subsample = "0.5", objective = "binary:logistic", validate_parameters = "TRUE"
## xgb.attributes:
##   niter
## callbacks:
##   cb.print.evaluation(period = print_every_n)
## # of features: 24
## niter: 50
## nfeatures : 24
## xNames : R1 R2 R3 R4 R5 R6 R7 R8 R9 R10 R11 R12 R13 R14 R15 R16 R17 R18 R19 R20 R21 R22 R23 R24
## problemType : Classification
## tuneValue :
##   nrounds max_depth eta gamma colsample_bytree min_child_weight subsample
## 91      50      3 0.4      0      0.6      1      0.5
## obsLevels : 0 1
## param :
## list()
```

```
# Predict on test data
XG_clf_predict <- predict(XG_clf_fit,corpRatios_x_test)
```

```
# Print Confusion matrix, Accuracy, Sensitivity etc
confusionMatrix(XG_clf_predict, corpRatios_y_test, positive = "1" )
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##           0 12  1
##           1  7 13
##
##           Accuracy : 0.7576
##           95% CI : (0.5774, 0.8891)
##           No Information Rate : 0.5758
##           P-Value [Acc > NIR] : 0.02392
##
##           Kappa : 0.5302
##
## Mcnemar's Test P-Value : 0.07710
##
##           Sensitivity : 0.9286
##           Specificity : 0.6316
##           Pos Pred Value : 0.6500
##           Neg Pred Value : 0.9231
##           Prevalence : 0.4242
##           Detection Rate : 0.3939
##           Detection Prevalence : 0.6061
##           Balanced Accuracy : 0.7801
##
##           'Positive' Class : 1
##
```

```
# Add results into clf_results dataframe
x4 <- confusionMatrix(XG_clf_predict, corpRatios_y_test , positive = "1")["overall"]
y4 <- confusionMatrix(XG_clf_predict, corpRatios_y_test , positive = "1")["byClass"]

clf_results[nrow(clf_results) + 1,] <- list(Model = "XG Boost",
                                           Accuracy = round (x4[["Accuracy"]],3),
                                           Precision = round (y4[["Precision"]],3),
                                           Recall = round (y4[["Recall"]],3),
                                           F1 = round (y4[["F1"]],3))

# Print Accuracy and F1 score
cat("Accuracy is ", round(x4[["Accuracy"]],3), "and F1 is ", round (y4[["F1"]],3) )
```

```
## Accuracy is  0.758 and F1 is  0.765
```

```
# Add results into cost_benefit_df dataframe for cost benefit analysis
a4 <- confusionMatrix(XG_clf_predict, corpRatios_y_test )

cost_benefit_df[nrow(cost_benefit_df) + 1,] <- list(Model = "XG Boost",
                                                    TP = a4[["table"]][4],
                                                    FN = a4[["table"]][3],
                                                    FP = a4[["table"]][2],
                                                    TN = a4[["table"]][1])
```

1.6 Neural Network classification

```

# Try different combinations of parameters like
# decay (prevents the weights from growing too large,)
# and size of Hidden layers
my.grid <- expand.grid(.decay = c(0.5, 0.1), .size = c(5, 7))

# stepmax is maximum steps for the training of the neural network
# threshold is set to 0.01, meaning that if the change in error during an iteration is
# less than 1%, then no further optimization will be carried out by the model
nn_clf_fit <- train(corpRatios_x_train,
                    corpRatios_y_train,
                    method = "nnet",
                    trace = F,
                    tuneGrid = my.grid,
                    linout = 0,
                    stepmax = 100,
                    threshold = 0.01 )

print(nn_clf_fit)

```

```

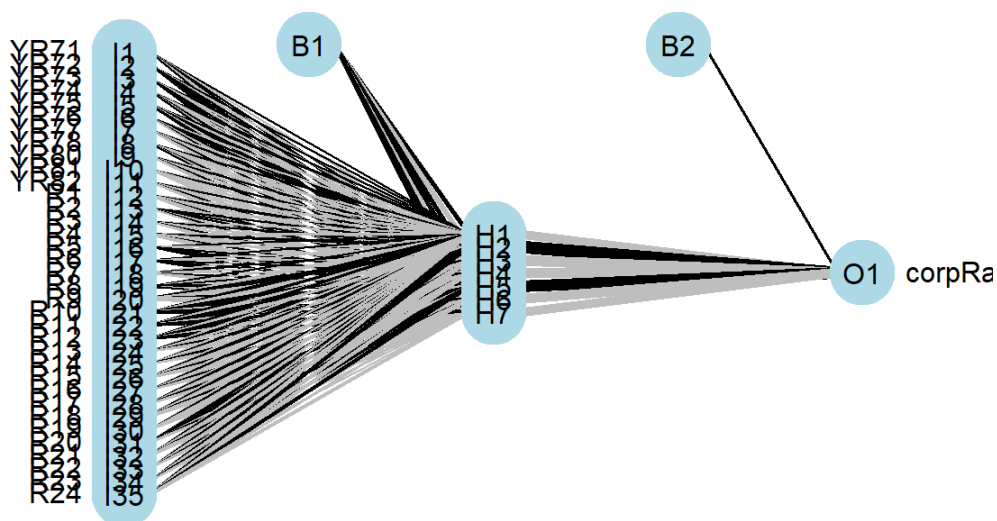
## Neural Network
##
## 99 samples
## 25 predictors
## 2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 99, 99, 99, 99, 99, ...
## Resampling results across tuning parameters:
##
##  decay  size  Accuracy  Kappa
##  0.1    5     0.6959966  0.3887285
##  0.1    7     0.6963369  0.3898208
##  0.5    5     0.7223306  0.4414132
##  0.5    7     0.7239338  0.4452731
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were size = 7 and decay = 0.5.

```

```

# Plot Neural Network
plotnet(nn_clf_fit$finalModel, y_names = "corpRatios Type")

```



```
# Predict on test data
nn_clf_predict <- predict(nn_clf_fit,corpRatios_x_test)
```

```
# Print Confusion matrix, Accuracy, Sensitivity etc
confusionMatrix(nn_clf_predict, corpRatios_y_test, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0   1
##           0 14   2
##           1   5 12
##
##           Accuracy : 0.7879
##           95% CI : (0.6109, 0.9102)
##    No Information Rate : 0.5758
##    P-Value [Acc > NIR] : 0.009148
##
##           Kappa : 0.5777
##
##  McNemar's Test P-Value : 0.449692
##
##           Sensitivity : 0.8571
##           Specificity : 0.7368
##           Pos Pred Value : 0.7059
##           Neg Pred Value : 0.8750
##           Prevalence : 0.4242
##           Detection Rate : 0.3636
##           Detection Prevalence : 0.5152
##           Balanced Accuracy : 0.7970
##
##           'Positive' Class : 1
##
```

```
# Add results into clf_results dataframe
x5 <- confusionMatrix(nn_clf_predict, corpRatios_y_test , positive = "1")["overall"]
y5 <- confusionMatrix(nn_clf_predict, corpRatios_y_test, positive = "1")["byClass"]

clf_results[nrow(clf_results) + 1,] <- list(Model = "Neural Network",
                                           Accuracy = round (x5[["Accuracy"]],3),
                                           Precision = round (y5[["Precision"]],3),
                                           Recall = round (y5[["Recall"]],3),
                                           F1 = round (y5[["F1"]],3))

# Print Accuracy and F1 score
cat("Accuracy is ", round(x5[["Accuracy"]],3), "and F1 is ", round (y5[["F1"]],3) )
```

```
## Accuracy is 0.788 and F1 is 0.774
```

```
# Add results into cost_benefit_df dataframe for cost benefit analysis
a5 <- confusionMatrix(nn_clf_predict, corpRatios_y_test)

cost_benefit_df[nrow(cost_benefit_df) + 1,] <- list(Model = "Neural Network",
                                                    TP = a5[["table"]][4],
                                                    FN = a5[["table"]][3],
                                                    FP = a5[["table"]][2],
                                                    TN = a5[["table"]][1])
```

Compare Accuracy for all Classification models

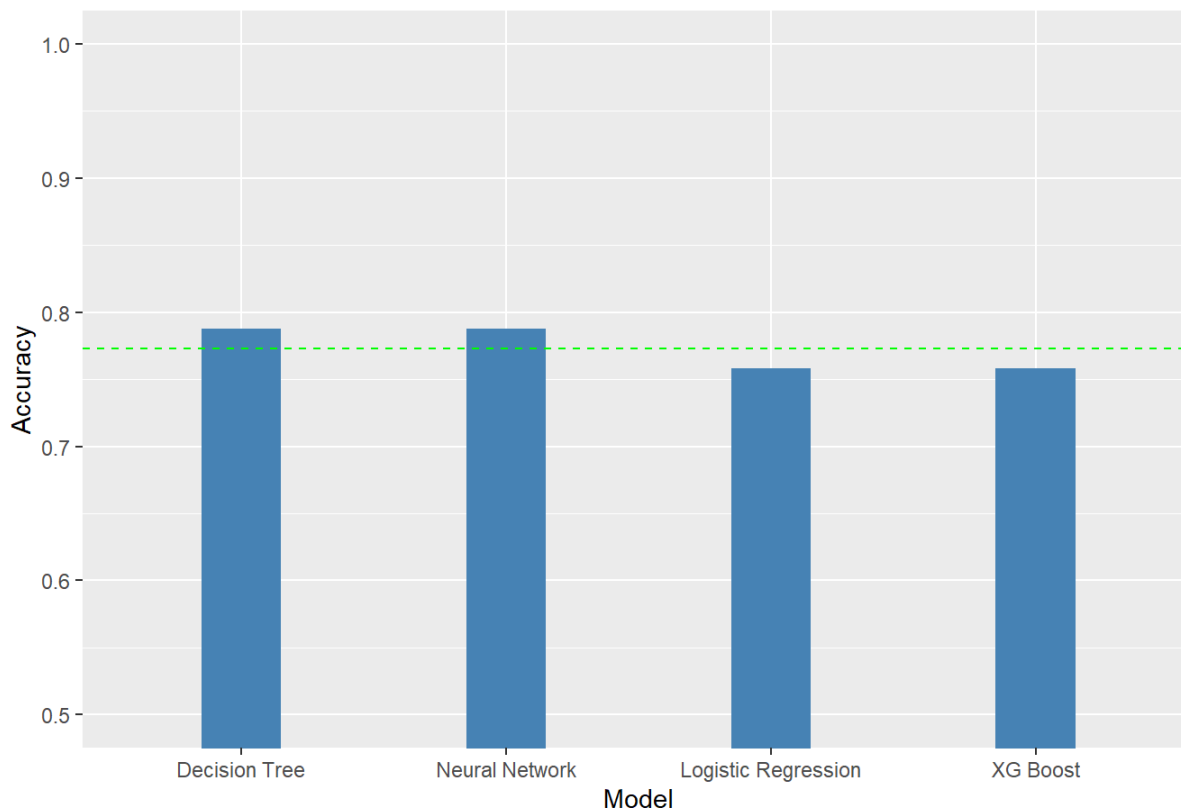
```
print(clf_results)
```

```
##           Model Accuracy Precision Recall    F1
## 1  Decision Tree    0.788     0.706  0.857 0.774
## 2 Logistic Regression 0.758     0.667  0.857 0.750
## 3      XG Boost     0.758     0.650  0.929 0.765
## 4   Neural Network    0.788     0.706  0.857 0.774
```

```
# 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(0.5, 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



1.8 ROC and Lift curves for all models

ROC curve - It is a performance measurement for classification problem at various thresholds settings. It tells how much a model is capable of distinguishing between classes.

Y axis - True Positive rate or Sensitivity = $(TP / TP + FN)$

X axis - False Positive rate or $(1 - \text{specificity}) = (FP / TN + FP)$

AUC - Area under ROC curve. Higher the AUC, better the model is at predicting 0s as 0s and 1s as 1s.

Lets Plot ROC curves for all the Models. The more “up and to the left” the ROC curve of a model is, the better the model. Also, higher the Area under curve, the better the model.


```
# Predict probabilities of each model to plot ROC curve
```

```
dtree_prob <- predict(dtree_fit, newdata = corpRatios_x_test, type = "prob", positive = "1")
XG_boost_prob <- predict(XG_clf_fit, newdata = corpRatios_x_test, type = "prob", positive = "1")
nn_clf_prob <- predict(nn_clf_fit, newdata = corpRatios_x_test, type = "prob", positive = "1")
```

```
# List of predictions
```

```
#pay attention to which column of the predictions you want to pick
```

```
#in this case its class with output = "1" ie bankruptcy
```

```
# head(dtree_prob) #good idea to see this
```

```
preds_list <- list( dtree_prob[,2],
                    glm_predict, XG_boost_prob[,2], nn_clf_prob[,2] )
```

```
# List of actual values (same for all)
```

```
m <- length(preds_list)
```

```
actuals_list <- rep(list(corpRatios_y_test), m)
```

```
# Plot the ROC curves
```

```
pred <- prediction(preds_list, actuals_list)
```

```
rocs <- performance(pred, "tpr", "fpr")
```

```
# calculate AUC for all models
```

```
AUC_models <- performance(pred, "auc")
```

```
auc_dt = round(AUC_models@y.values[[1]], 3)
```

```
auc_lr = round(AUC_models@y.values[[2]], 3)
```

```
auc_xg = round(AUC_models@y.values[[3]], 3)
```

```
auc_nn = round(AUC_models@y.values[[4]], 3)
```

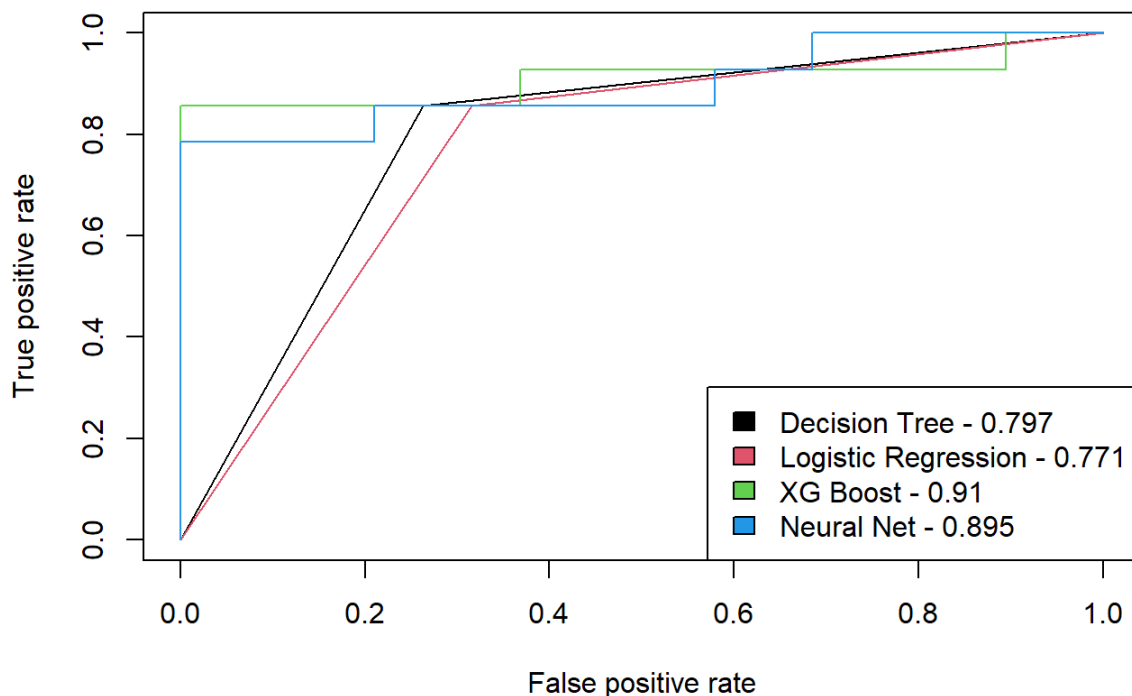
```
# Plot the ROC curves
```

```
plot(rocs, col = as.list(1:m), main = "ROC Curves of different models")
```

```
legend(x = "bottomright",
```

```
      legend = c( paste0("Decision Tree - ", auc_dt),
                  paste0("Logistic Regression - ", auc_lr),
                  paste0("XG Boost - ", auc_xg),
                  paste0("Neural Net - ", auc_nn)), fill = 1:m)
```

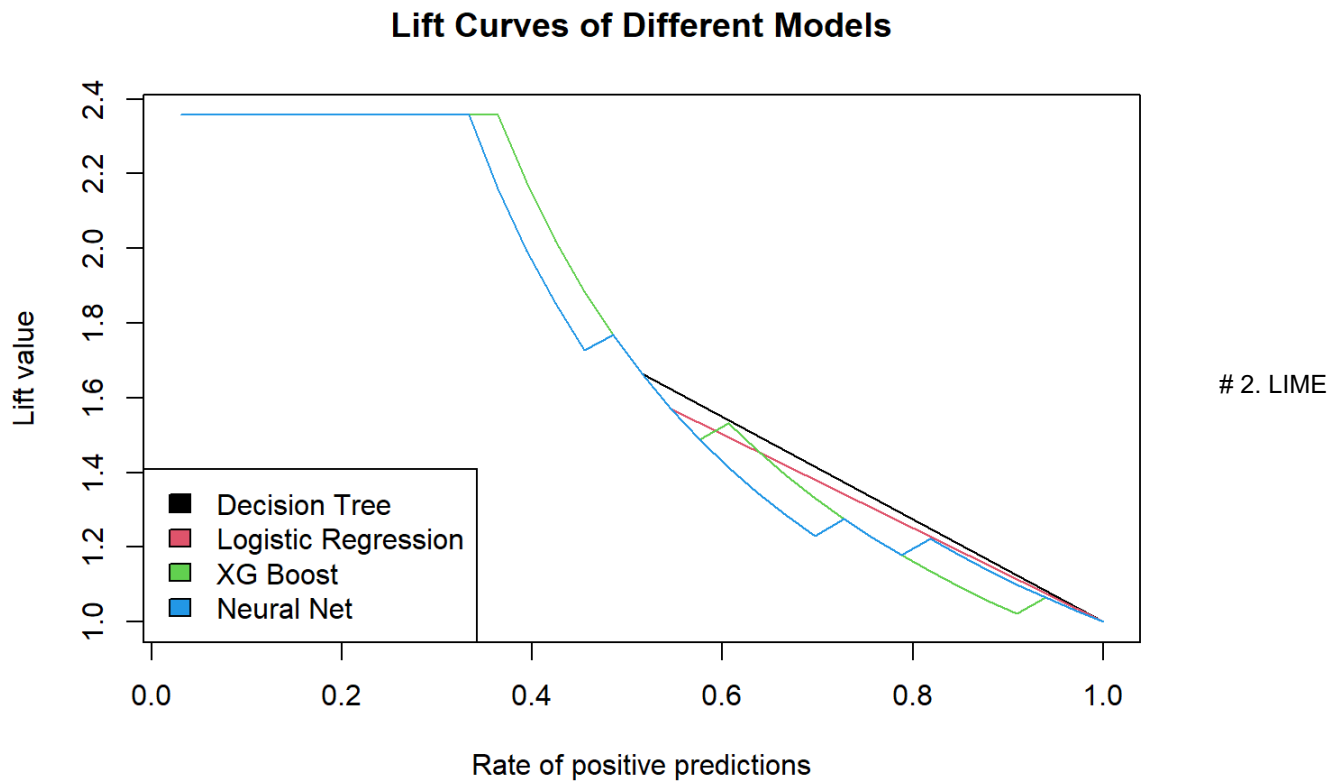
ROC Curves of different models



Lift curve - Lift is a measure of the effectiveness of a predictive model calculated as the ratio between the results obtained with and without the predictive model. The lift chart shows how much more likely we are to predict the correct outcome than a random guess.

```
lifts <- performance(pred, "lift", "rpp")

# Plot the Lift curves
plot(lifts, col = as.list(1:m), main = "Lift Curves of Different Models")
legend(x = "bottomleft",
      legend = c( "Decision Tree",
                  "Logistic Regression",
                  "XG Boost",
                  "Neural Net"), fill = 1:m)
```



Local Interpretable Model-agnostic Explanations (LIME) is a visualization technique that helps explain individual predictions. This method is model agnostic, so it can be applied to any supervised regression or classification model

First, create an “explainer” object using the lime function, which is a list that contains the ML model and the feature distributions for the training data. Note that we use the training data and Neural Net model we just trained to create the “explainer” object.

```
explainer_caret <- lime(corpRatios_x_train, nn_clf_fit, n_bins = 10)
class(explainer_caret)
```

```
## [1] "data_frame_explainer" "explainer"           "list"
```

```
summary(explainer_caret)
```

##	Length	Class	Mode
## model	21	train	list
## preprocess	1	-none-	function
## bin_continuous	1	-none-	logical
## n_bins	1	-none-	numeric
## quantile_bins	1	-none-	logical
## use_density	1	-none-	logical
## feature_type	25	-none-	character
## bin_cuts	25	-none-	list
## feature_distribution	25	-none-	list

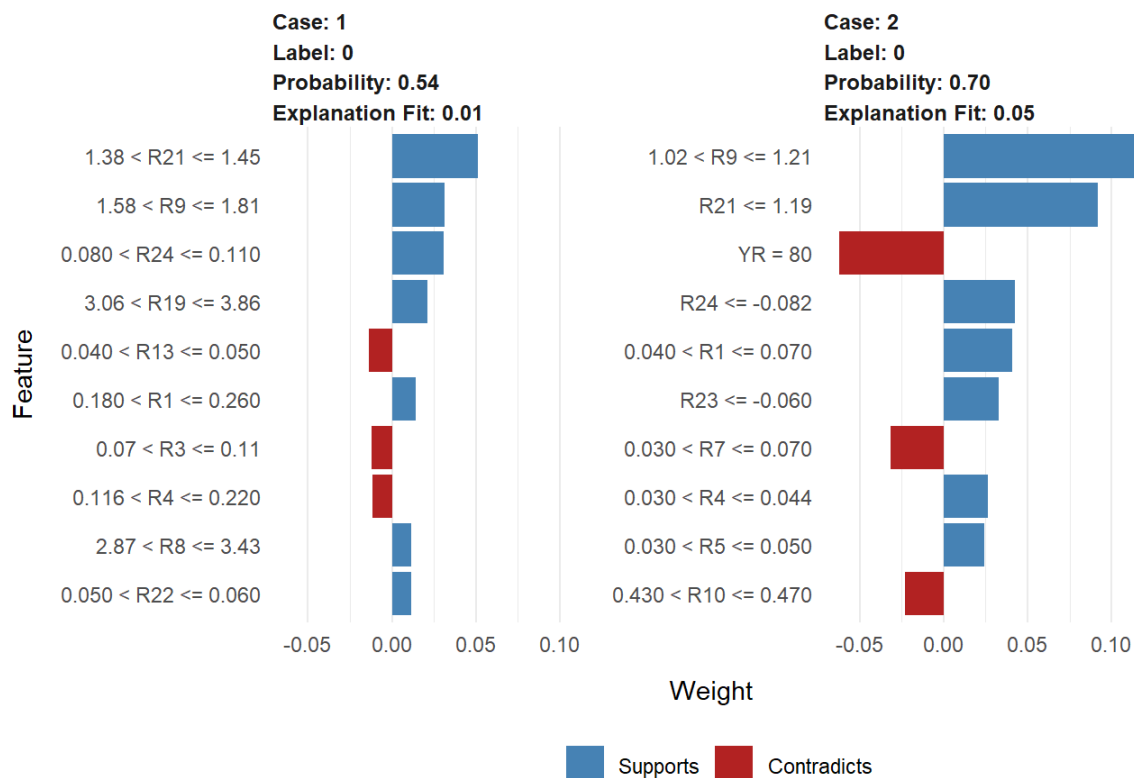
Once we create our lime object, we can now perform the LIME algorithm using the explain function. We create local explanations for the first two observations in the test dataset

```
explanation_caret <- explain(
  x = corpRatios_x_test[c(1,2),],
  explainer = explainer_caret ,
  n_features = 10,
  n_labels = 1)
```

Visualizing results

There are several plotting functions provided by lime for visualization but we are only concerned with two. The most important of which is `plot_features`. This will create a visualization containing an individual plot for each observation (case 1, 2, ..., n) in our test data. Since we specified `n_features = 10` it will plot the 10 most influential variables that best explain the linear model in that observations local region and whether the variable is causes an increase in the probability (supports) or a decrease in the probability (contradicts). It also provides us with the model fit for each model ("Explanation Fit: XX"), which allows us to see how well that model explains the local region.

```
plot_features(explanation_caret)
```



We can infer that case 1 has a

higher likelihood of bankruptcy out of the 2 observations and the variable R5 (CFF0/SALES) appears to influence this high probability.

The other plot we can create is a heatmap showing how the different variables selected across all the observations influence each case. This plot becomes useful if you are trying to find common features that influence all observations or if you are performing this analysis across many observations which makes plot_features difficult to discern.

```
plot_explanations(explanation_caret)
```

