AG947_Assignment_Code

```
library(readxl)
library(randomForest)
```

```
## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.
```

```
library(caret)
```

```
## Loading required package: ggplot2

##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:randomForest':
##
##     margin

## Loading required package: lattice
```

```
library(iml)


setwd("/home/ahmed/Desktop/Strath/AG947/Assignment")

training_data_orig <- read_excel("Corporate ESG data - 2012-2023 - filtered - E-S-G.xlsx",
                                 sheet = "ESG Sample - Training")
testing_data_orig <- read_excel("Corporate ESG data - 2012-2023 - filtered - E-S-G.xlsx",
                                sheet = "ESG Sample - Testing")

#specific columns for analysis
training_data <- subset(training_data_orig,
                        select = c('HighE75perc','TOTALASSETS','TOTALDEBTCOMMONEQUITY',
                                   'EARNINGSPERSHARE','RETURNONEQUITYTOTAL',
                                   'RETURNONASSETS','NETSALESORREVENUES'))
testing_data <- subset(testing_data_orig, select = c('HighE75perc','TOTALASSETS',
                                                     'TOTALDEBTCOMMONEQUITY','EARNINGSPERSHARE',
                                                     'RETURNONEQUITYTOTAL','RETURNONASSETS',
                                                     'NETSALESORREVENUES'))

# Encoding the target features from the training and testing data
training_data$HighE75perc <- factor(training_data$HighE75perc)
testing_data$HighE75perc <- factor(testing_data$HighE75perc)
```

```r
# Dealing with missing values
testing_data[,-1] <- lapply(testing_data[,-1],
                            function(x) ifelse(is.na(x), mean(x, na.rm = TRUE), x))

# Training the random forest model
rf_model_train <- randomForest(HighE75perc ~ TOTALASSETS + TOTALDEBTCOMMONEQUITY +
                                 EARNINGSPERSHARE + RETURNONEQUITYTOTAL +
                                 RETURNONASSETS + NETSALESORREVENUES,
                               data = training_data)

cm_train_e <- confusionMatrix(rf_model_train$predicted, training_data$HighE75perc)

predicted_test <- predict(rf_model_train, newdata = testing_data, type = "response")
cm_test_e <- confusionMatrix(predicted_test, testing_data$HighE75perc)

print(cm_train_e)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 8747 1099
##          1  536 1995
##
##                Accuracy : 0.8679
##                  95% CI : (0.8618, 0.8738)
##     No Information Rate : 0.75
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.625
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9423
##             Specificity : 0.6448
##          Pos Pred Value : 0.8884
##          Neg Pred Value : 0.7882
##              Prevalence : 0.7500
##          Detection Rate : 0.7067
##    Detection Prevalence : 0.7955
##       Balanced Accuracy : 0.7935
##
##        'Positive' Class : 0
##
```

```r
print(cm_test_e)
```

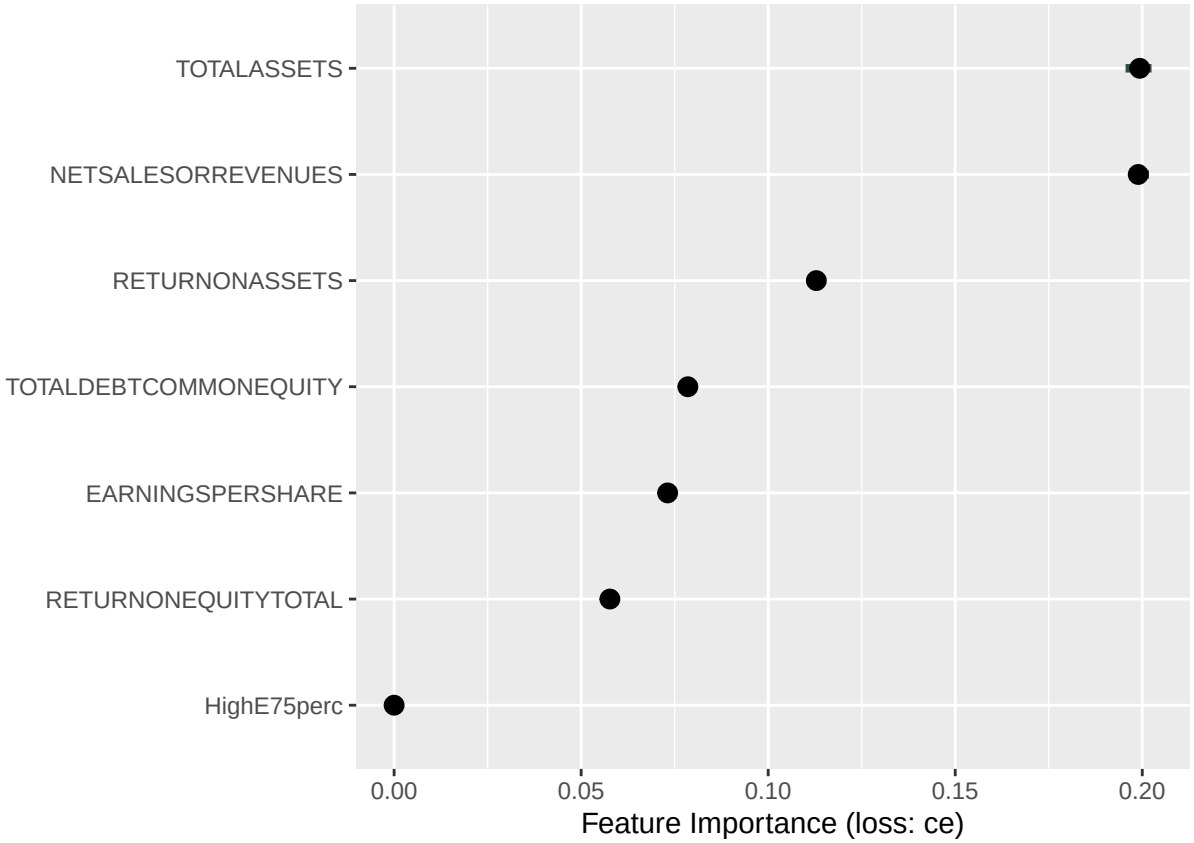
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 2037  451
```

```
##          1   87  436
##
##                Accuracy : 0.8213
##                  95% CI : (0.8072, 0.8349)
##     No Information Rate : 0.7054
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5117
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9590
##             Specificity : 0.4915
##          Pos Pred Value : 0.8187
##          Neg Pred Value : 0.8337
##              Prevalence : 0.7054
##          Detection Rate : 0.6765
##    Detection Prevalence : 0.8263
##       Balanced Accuracy : 0.7253
##
##        'Positive' Class : 0
##
```

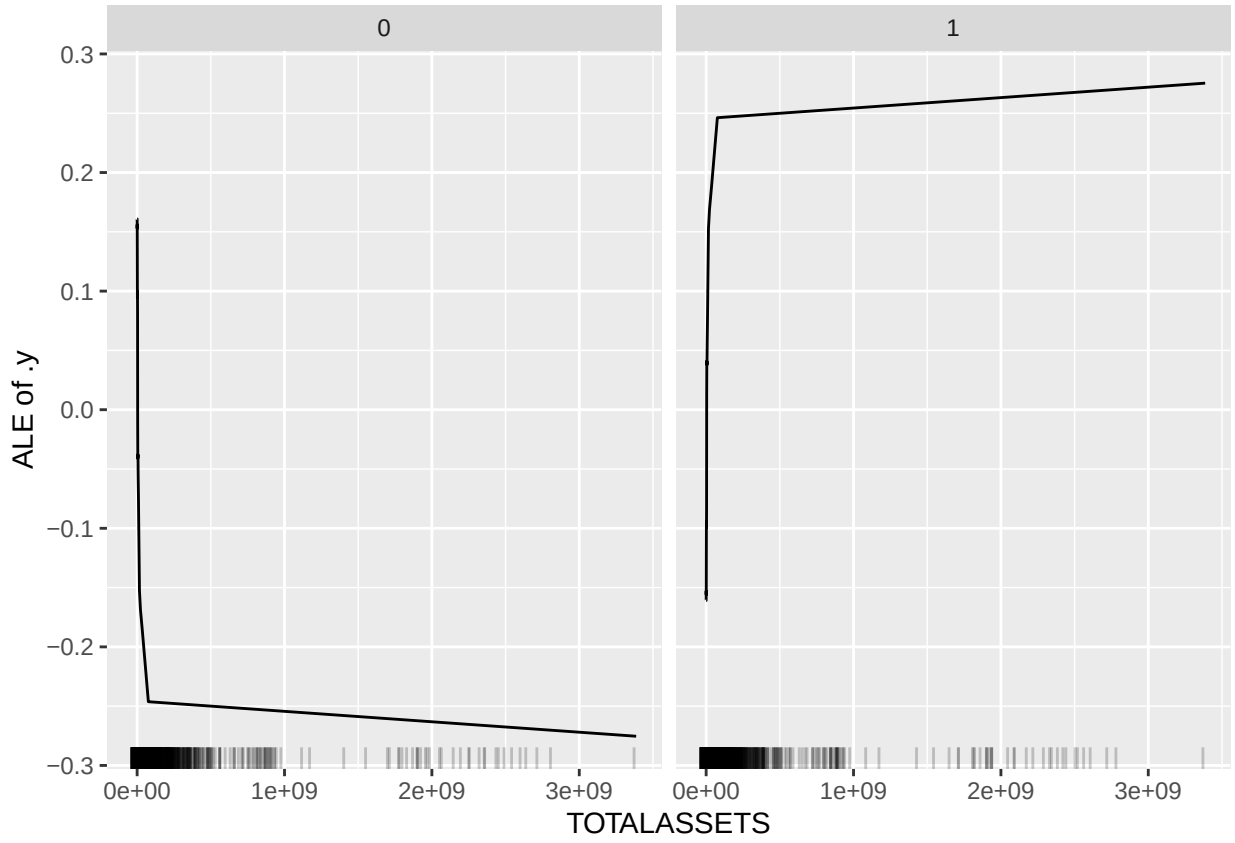
```
mod <- Predictor$new(rf_model_train, data = training_data, y = training_data$HighE75perc)
# creating predictor object

#1. Feature importance - model level
imp <- FeatureImp$new(mod, loss = "ce", compare = "difference")
```

```
## Warning: package 'generics' was built under R version 4.3.2

## Warning: package 'pkgconfig' was built under R version 4.3.2

## Warning: package 'withr' was built under R version 4.3.2

## Warning: package 'evaluate' was built under R version 4.3.2

## Warning: package 'rstudioapi' was built under R version 4.3.2

## Warning: package 'R6' was built under R version 4.3.2
```

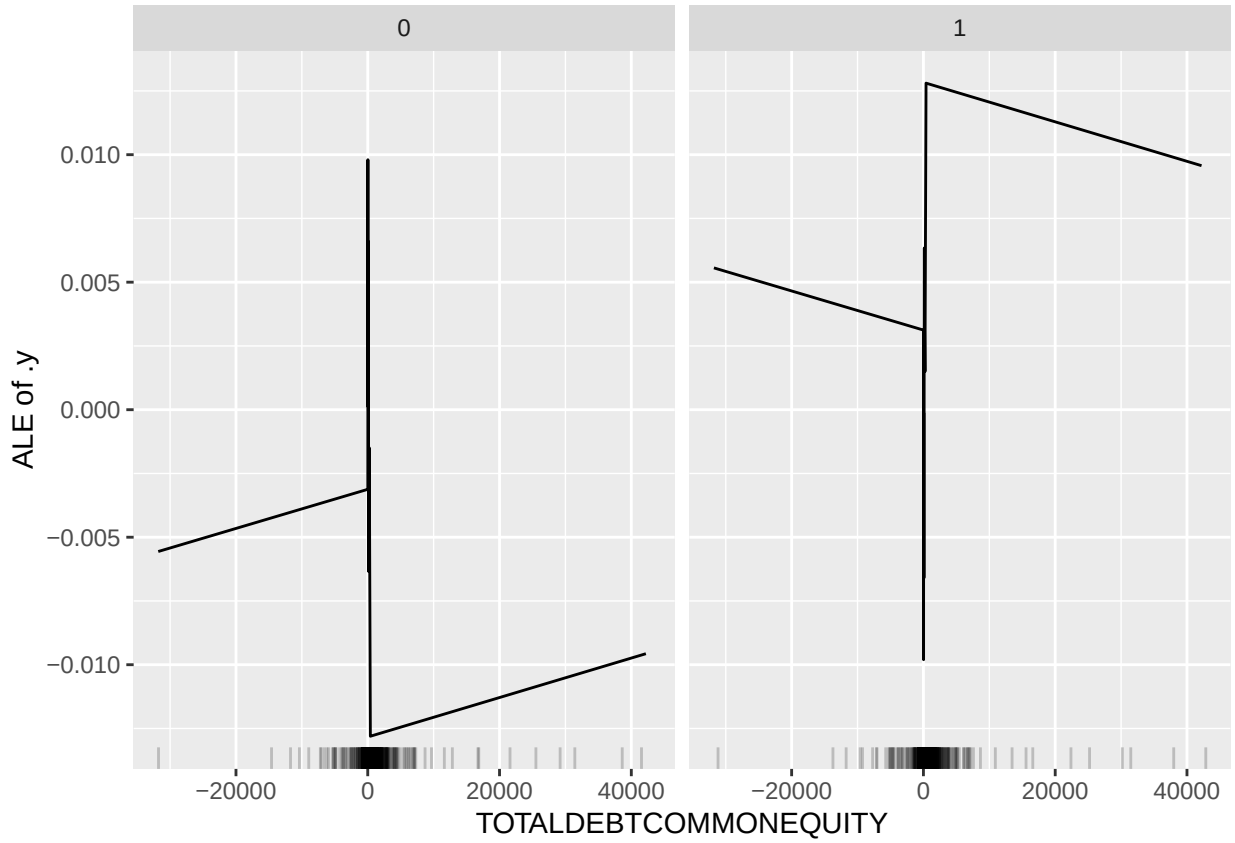
```
plot(imp)
```



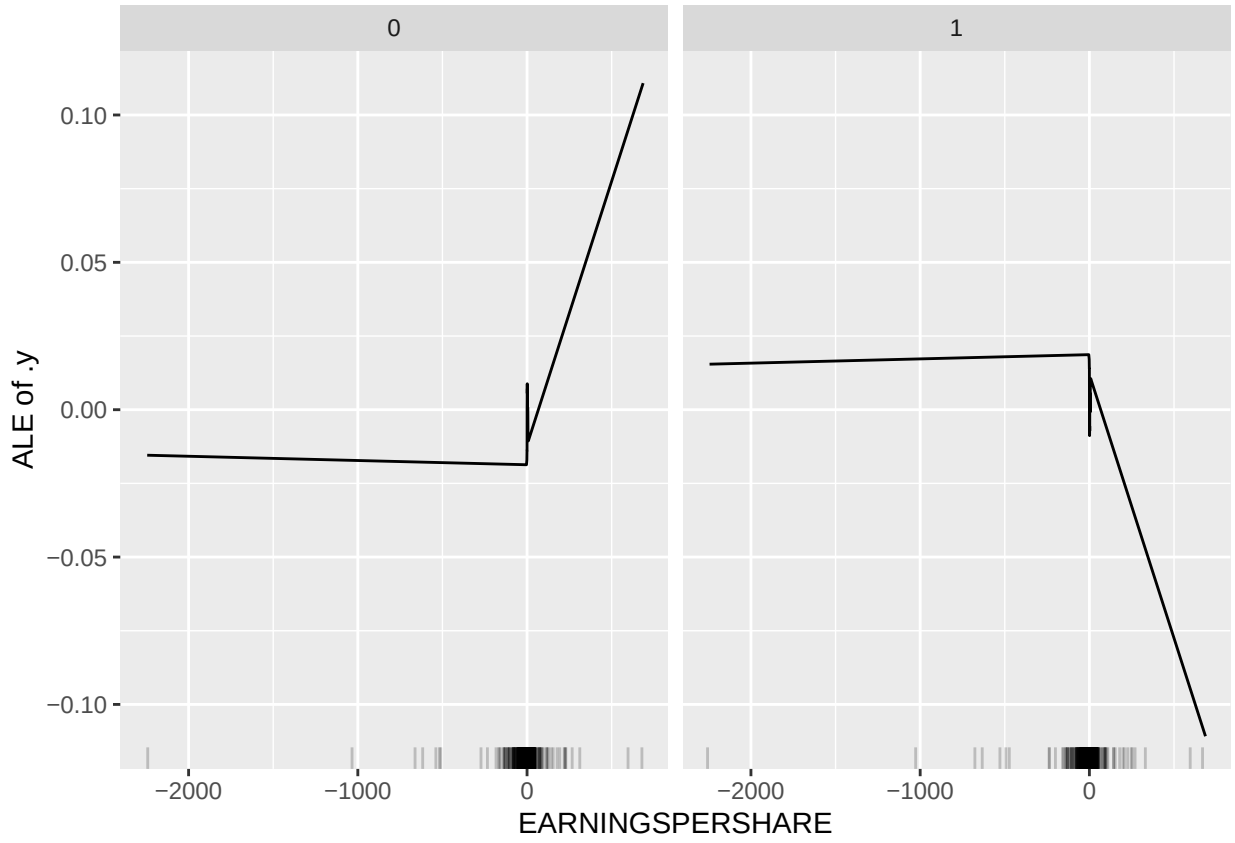

```
#2. Feature effects - model level
eff_1 <- FeatureEffect$new(mod, feature = "TOTALASSETS", method = 'ale')
plot(eff_1)
```



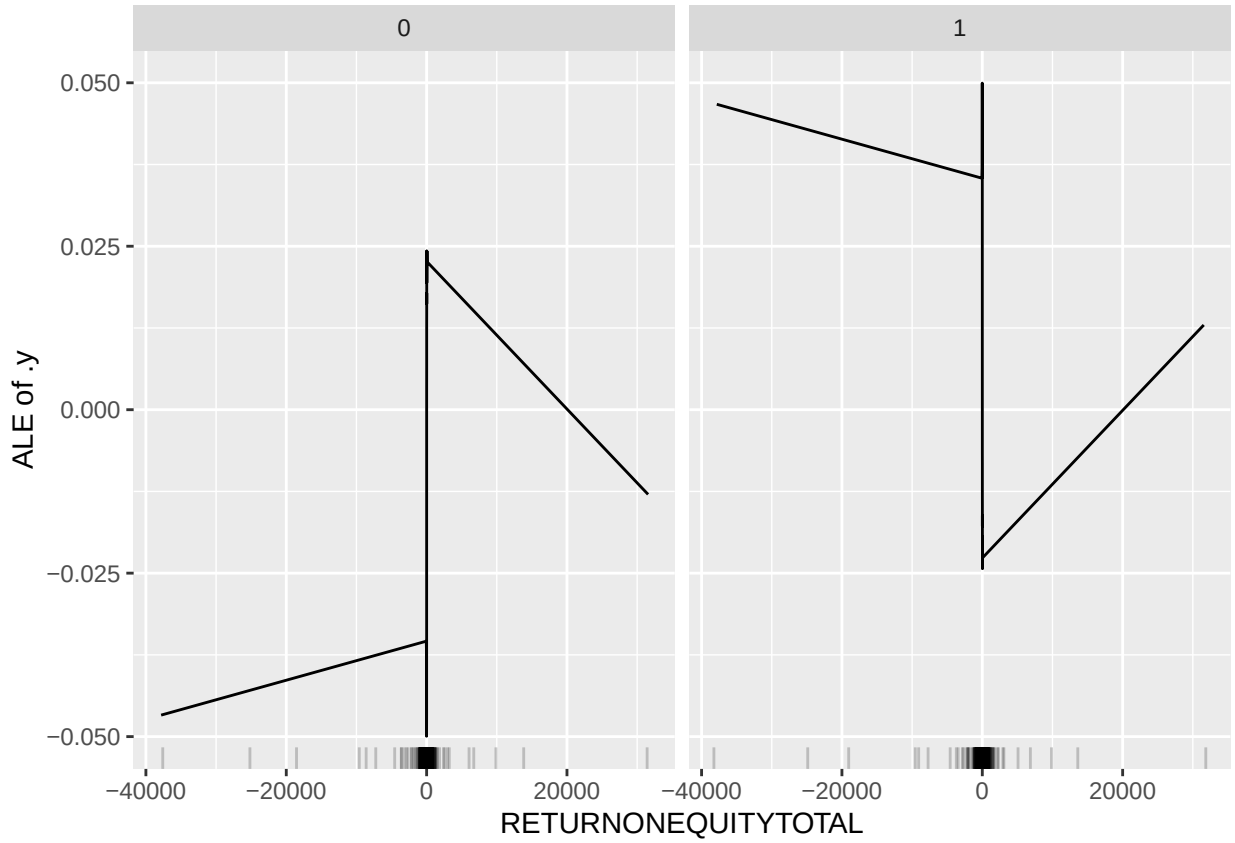

```
eff_2 <- FeatureEffect$new(mod, feature = "TOTALDEBTCOMMONEQUITY", method = 'ale')
plot(eff_2)
```



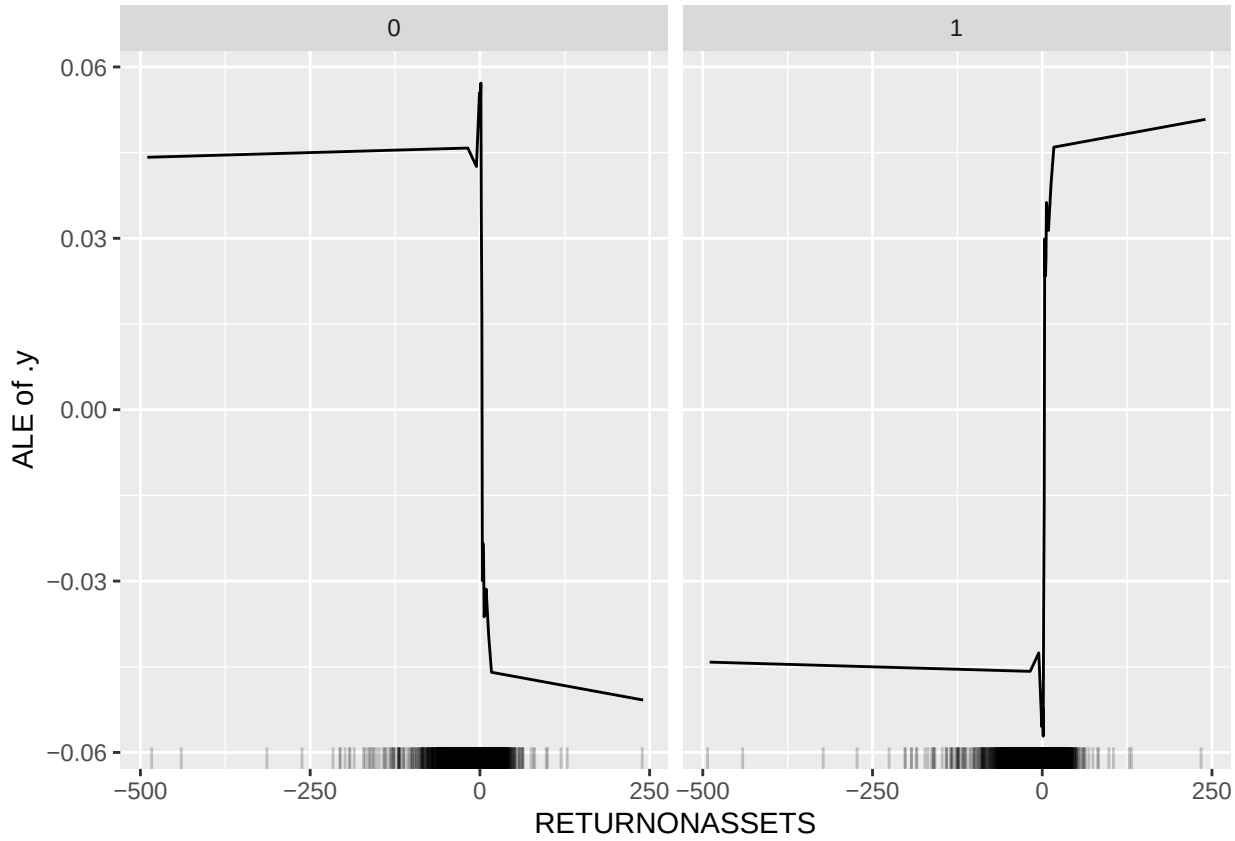

```
eff_3 <- FeatureEffect$new(mod, feature = "EARNINGSPERSHARE", method = 'ale')
plot(eff_3)
```



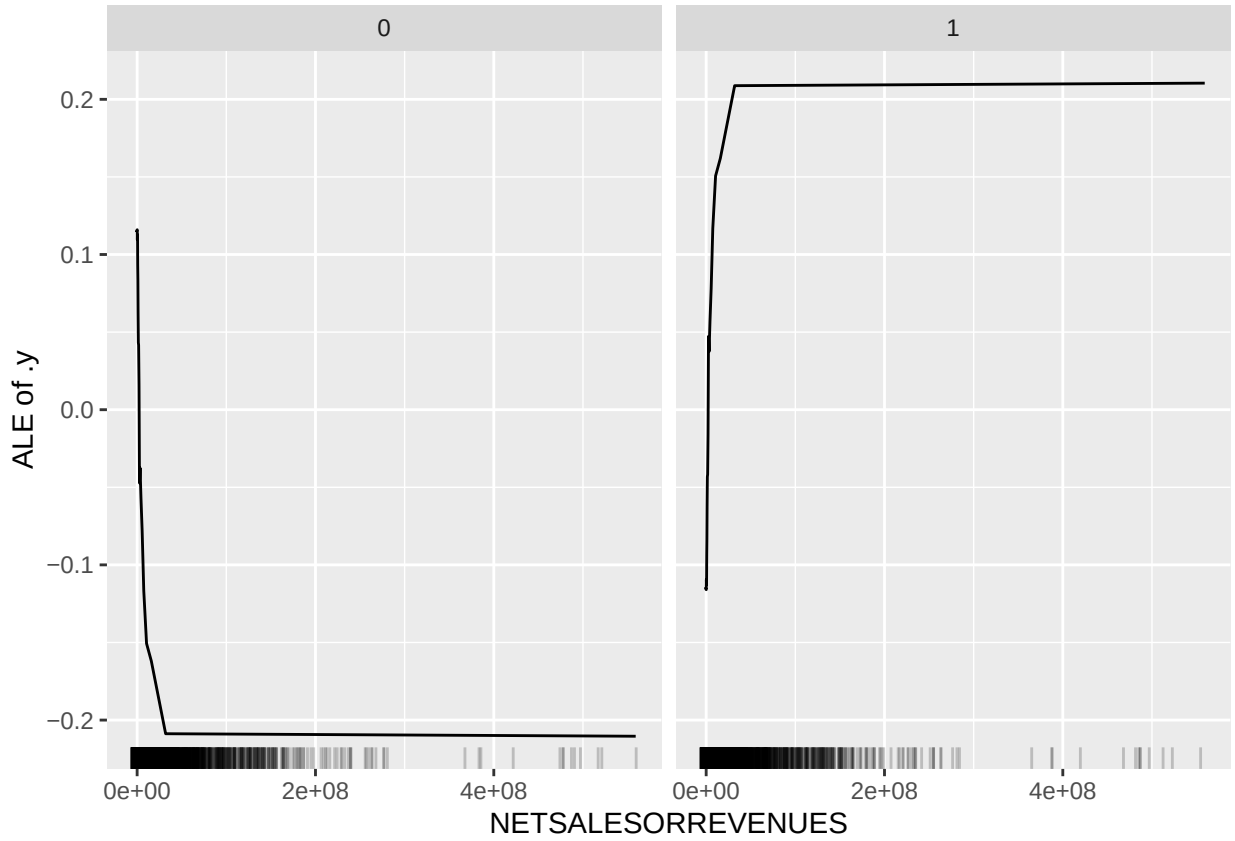

```
eff_4 <- FeatureEffect$new(mod, feature = "RETURNONEQUITYTOTAL", method = 'ale')
plot(eff_4)
```



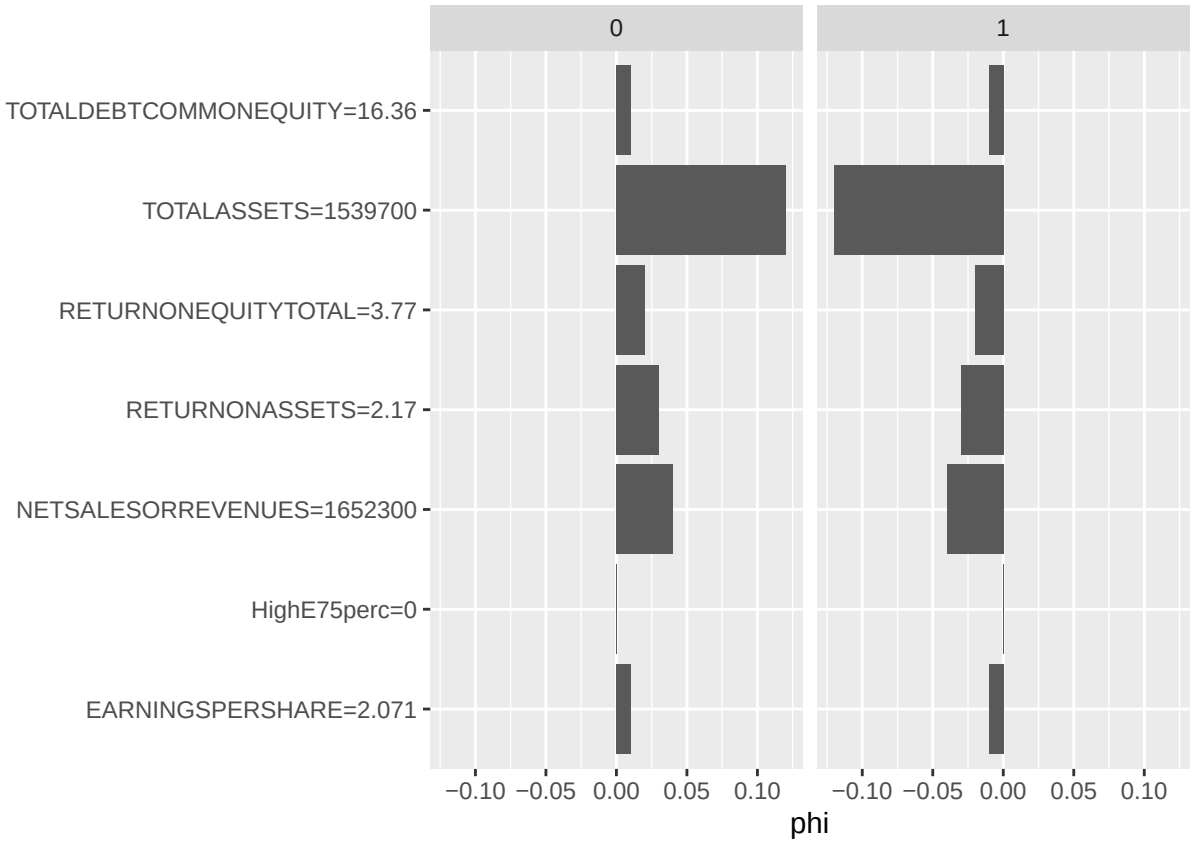

```
eff_5 <- FeatureEffect$new(mod, feature = "RETURNONASSETS", method = 'ale')
plot(eff_5)
```




```
eff_6 <- FeatureEffect$new(mod, feature = "NETSALESORREVENUES", method = 'ale')
plot(eff_6)
```

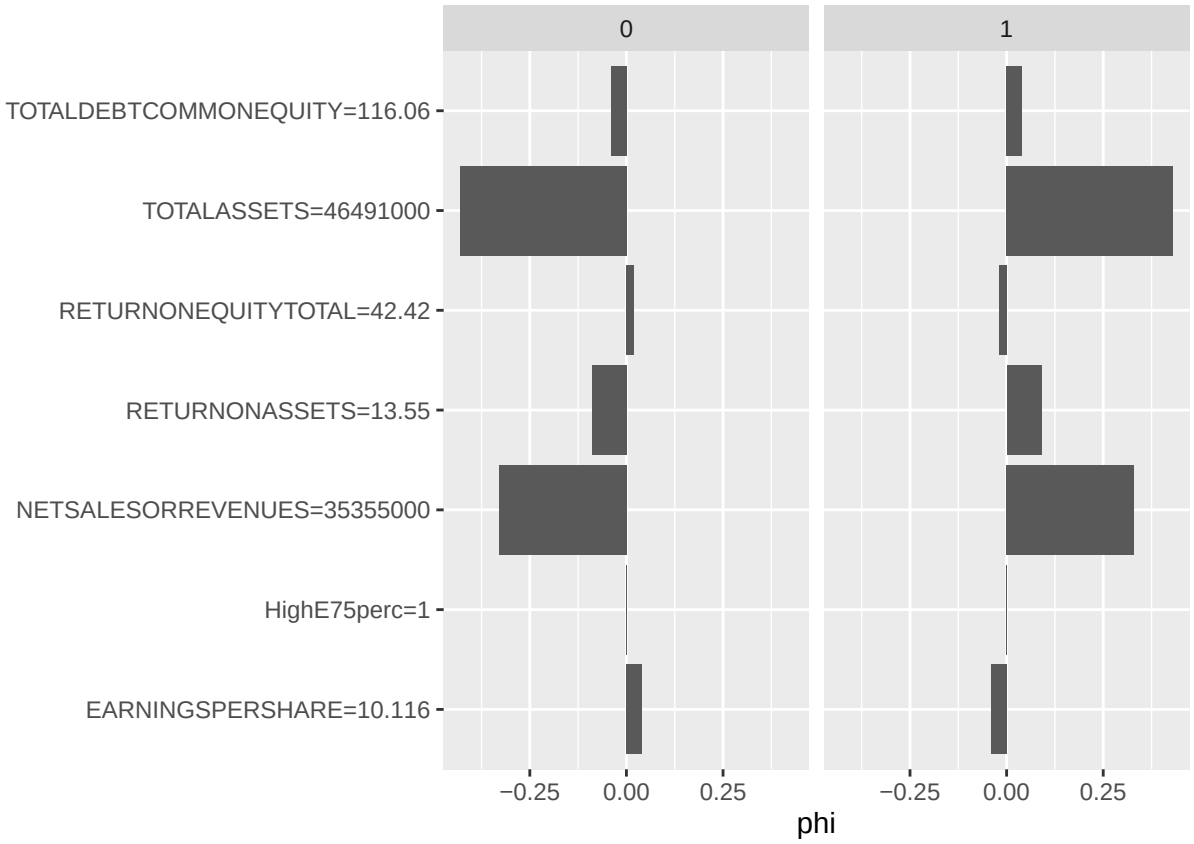



```
# 3. Shapley values - local predictions
shapley <- Shapley$new(mod, x.interest = testing_data[8,])
plot(shapley)
```

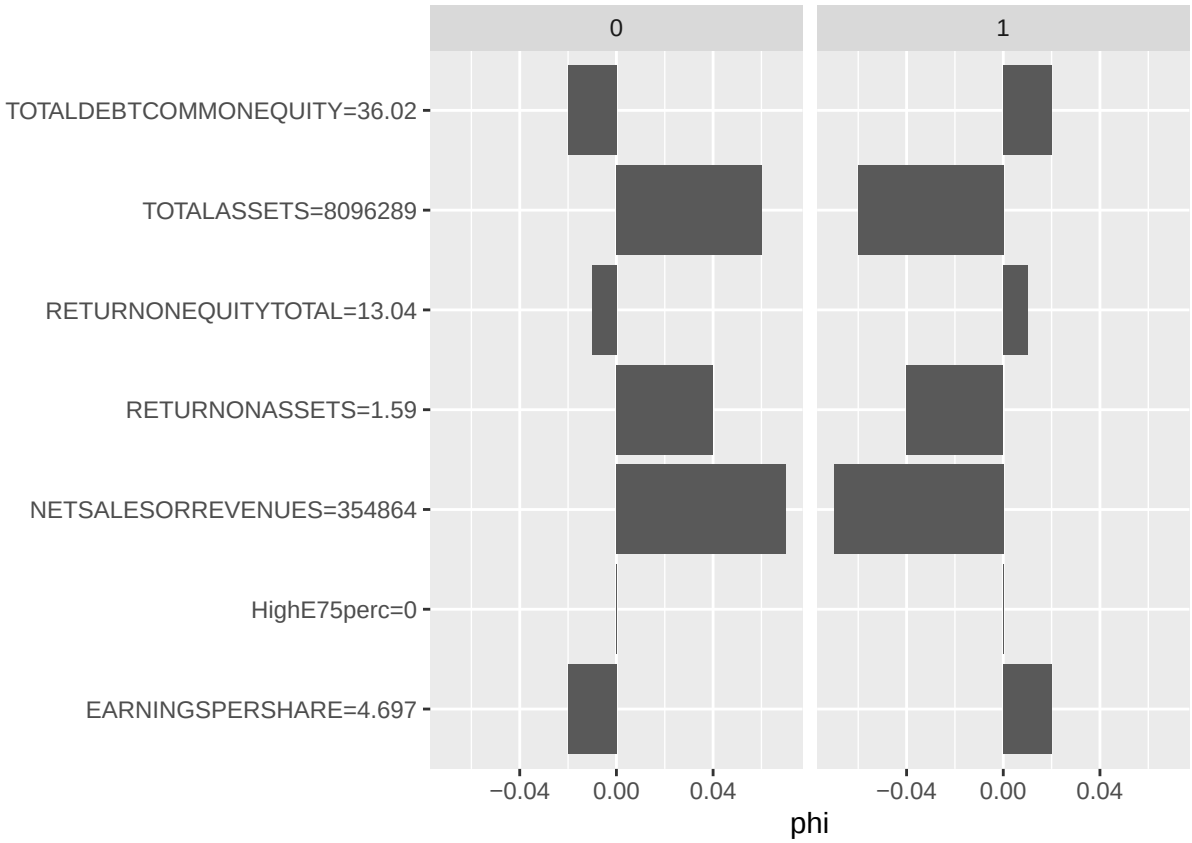



```r
# Analysis of feature importance based on four scenarios
results <- data.frame(Row = 1:nrow(testing_data),
                      Actual = testing_data$HighE75perc,
                      Predicted = predicted_test, row.names = NULL)


results$Outcome <- with(results, ifelse(Actual == Predicted & Actual == "1",
                                        "TP",
                                        ifelse(Actual == Predicted & Actual == "0",
                                               "TN",
                                               ifelse(Actual != Predicted & Actual == "1",
                                                      "FN", "FP"))))

# choosing one row in each scenario
TP_row <- subset(results, Outcome == "TP")$Row[1]
TN_row <- subset(results, Outcome == "TN")$Row[1]
FP_row <- subset(results, Outcome == "FP")$Row[1]
FN_row <- subset(results, Outcome == "FN")$Row[1]

predictor <- Predictor$new(rf_model_train,
                           data = testing_data, y = testing_data$HighE75perc)

# Function to plot Shapley values for a given instance index
plot_shapley_for_instance <- function(instance_index) {
  shapley <- Shapley$new(predictor, x.interest = testing_data[instance_index,])
  plot(shapley)
```

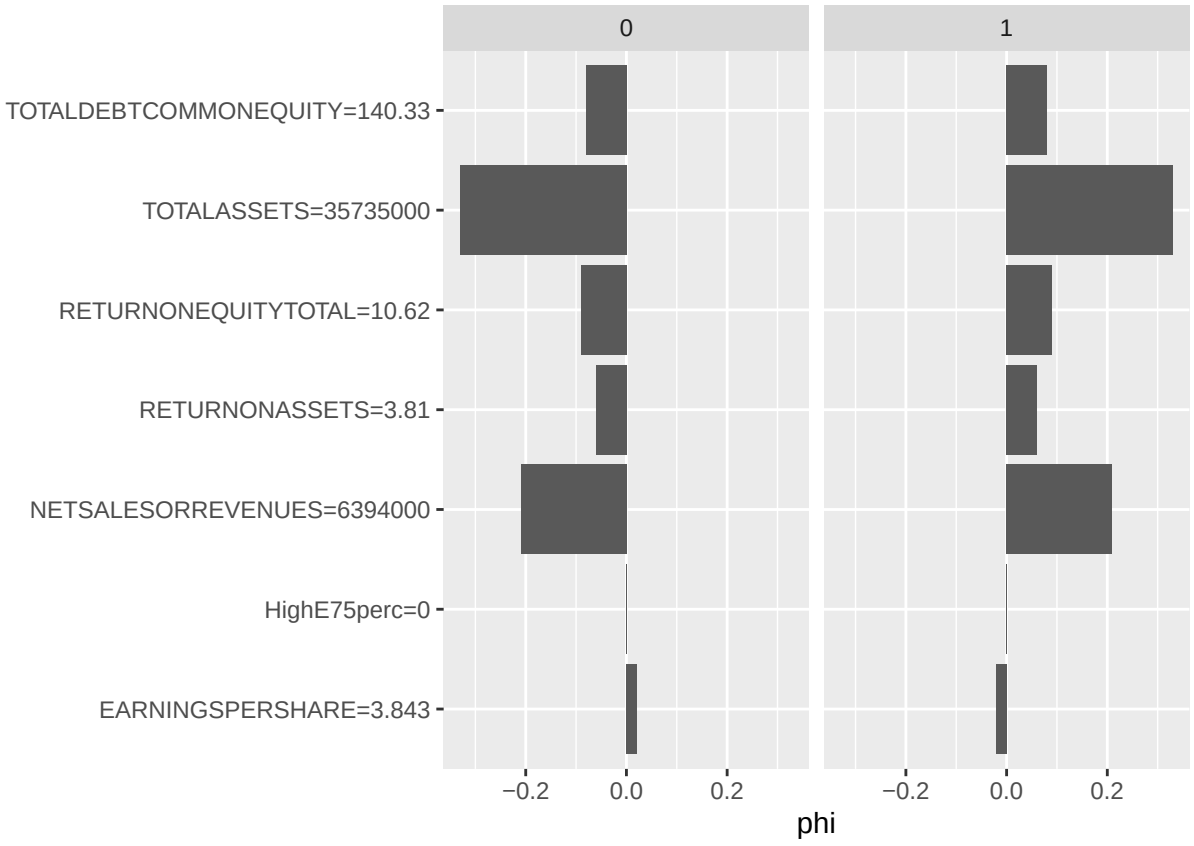
```
}

plot_shapley_for_instance(TP_row)
```



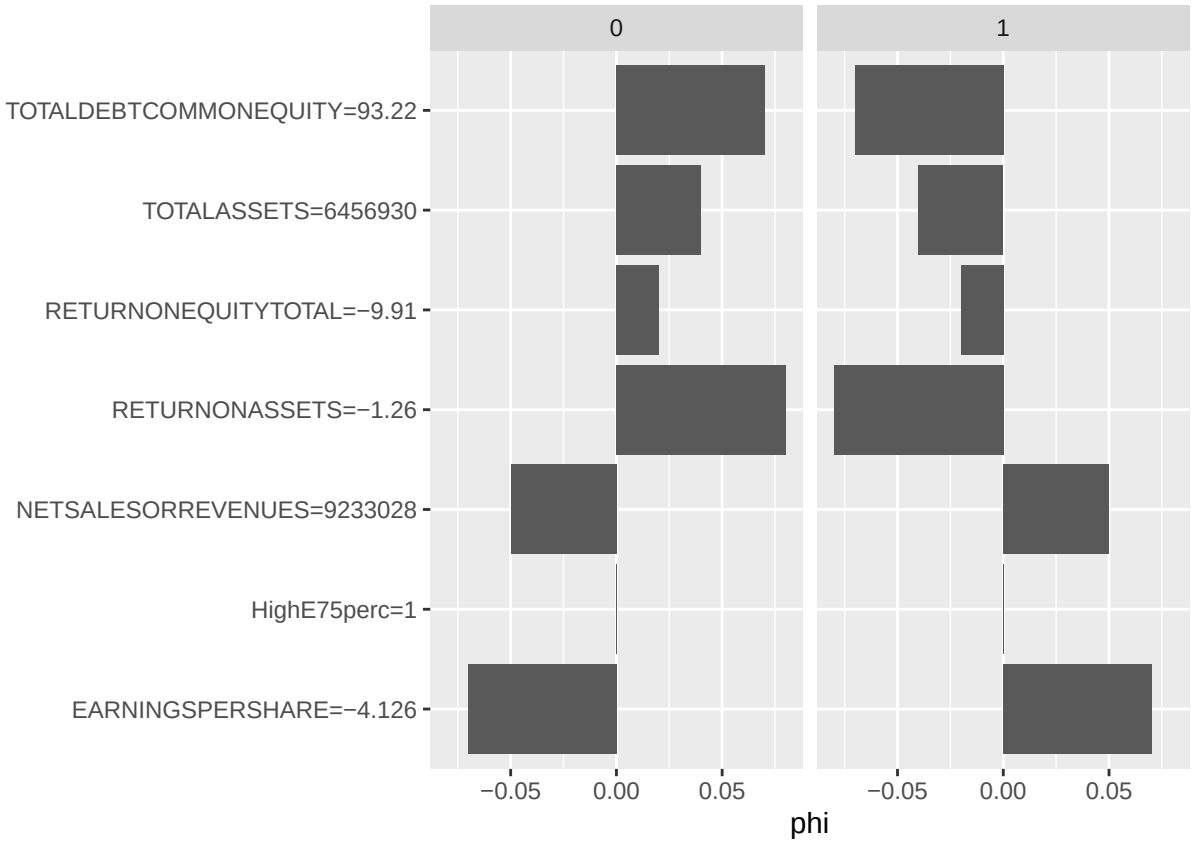

```
plot_shapley_for_instance(TN_row)
```




```
plot_shapley_for_instance(FP_row)
```




```
plot_shapley_for_instance(FN_row)
```




```
# S SCORE
setwd("/home/ahmed/Desktop/Strath/AG947/Assignment")

training_data_orig <- read_excel("Corporate ESG data - 2012-2023 - filtered - E-S-G.xlsx",
                                 sheet = "ESG Sample - Training")
testing_data_orig <- read_excel("Corporate ESG data - 2012-2023 - filtered - E-S-G.xlsx",
                                sheet = "ESG Sample - Testing")

training_data <- subset(training_data_orig, select = c('HighS75perc','TOTALASSETS',
                                                       'TOTALDEBTCOMMONEQUITY',
                                                       'EARNINGSPERSHARE','RETURNONEQUITYTOTAL',
                                                       'RETURNONASSETS','NETSALESORREVENUES'))
testing_data <- subset(testing_data_orig, select = c('HighS75perc','TOTALASSETS',
                                                     'TOTALDEBTCOMMONEQUITY','EARNINGSPERSHARE',
                                                     'RETURNONEQUITYTOTAL','RETURNONASSETS',
                                                     'NETSALESORREVENUES'))

training_data$HighS75perc <- factor(training_data$HighS75perc)
testing_data$HighS75perc <- factor(testing_data$HighS75perc)

# Filling missing values
testing_data[,-1] <- lapply(testing_data[,-1],
                            function(x) ifelse(is.na(x), mean(x, na.rm = TRUE), x))

#training random forest model
rf_model_train <- randomForest(HighS75perc ~ TOTALASSETS + TOTALDEBTCOMMONEQUITY +
```

```
                                  EARNINGSPERSHARE + RETURNONEQUITYTOTAL +
                                  RETURNONASSETS + NETSALESORREVENUES,
                                data = training_data)

# Creating confusion matrix
cm_train_s <- confusionMatrix(rf_model_train$predicted, training_data$HighS75perc)

predicted_test <- predict(rf_model_train, newdata = testing_data, type = "response")
cm_test_s <- confusionMatrix(predicted_test, testing_data$HighS75perc)

print(cm_train_s)
```

```
##      [,1]  [,2]
## [1,]    0     0
## [2,]    0 10205
```

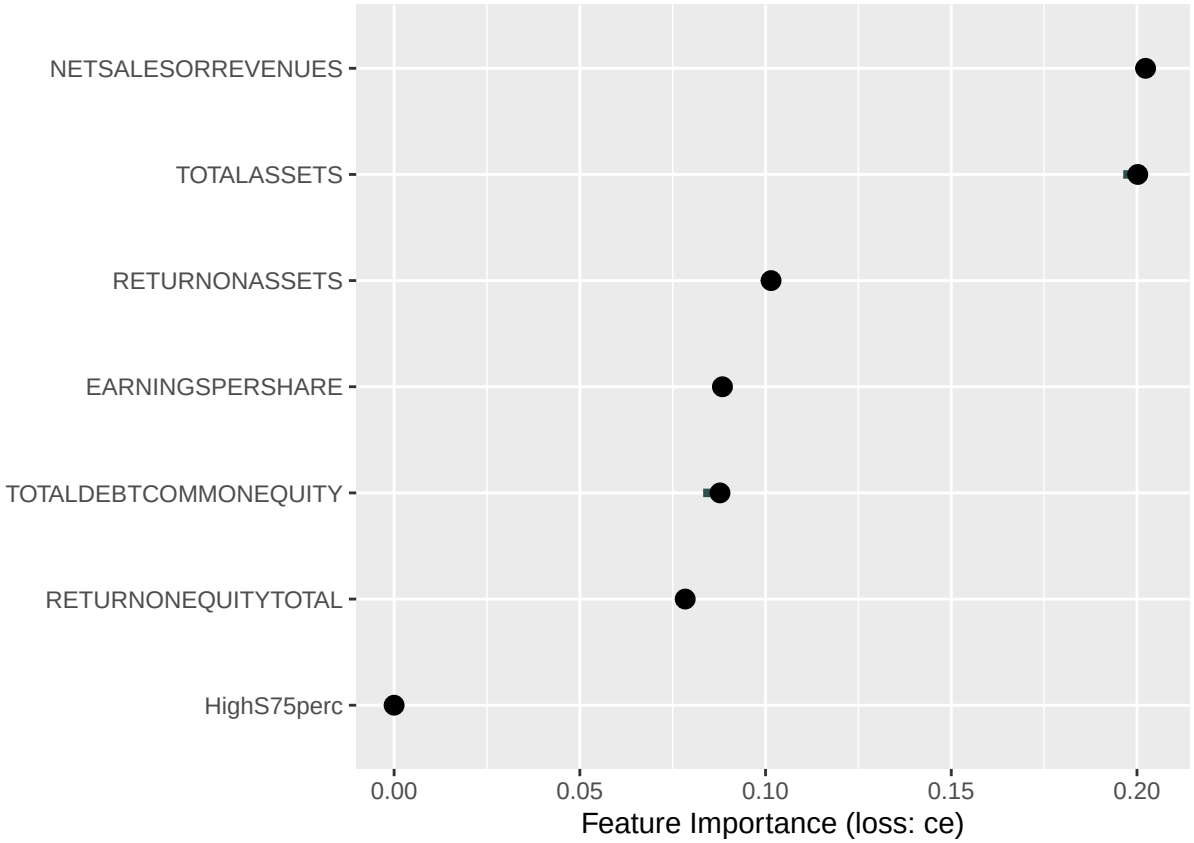
```
print(cm_test_s)
```

```
##      [,1] [,2]
## [1,]    0    0
## [2,]    0 2537
```

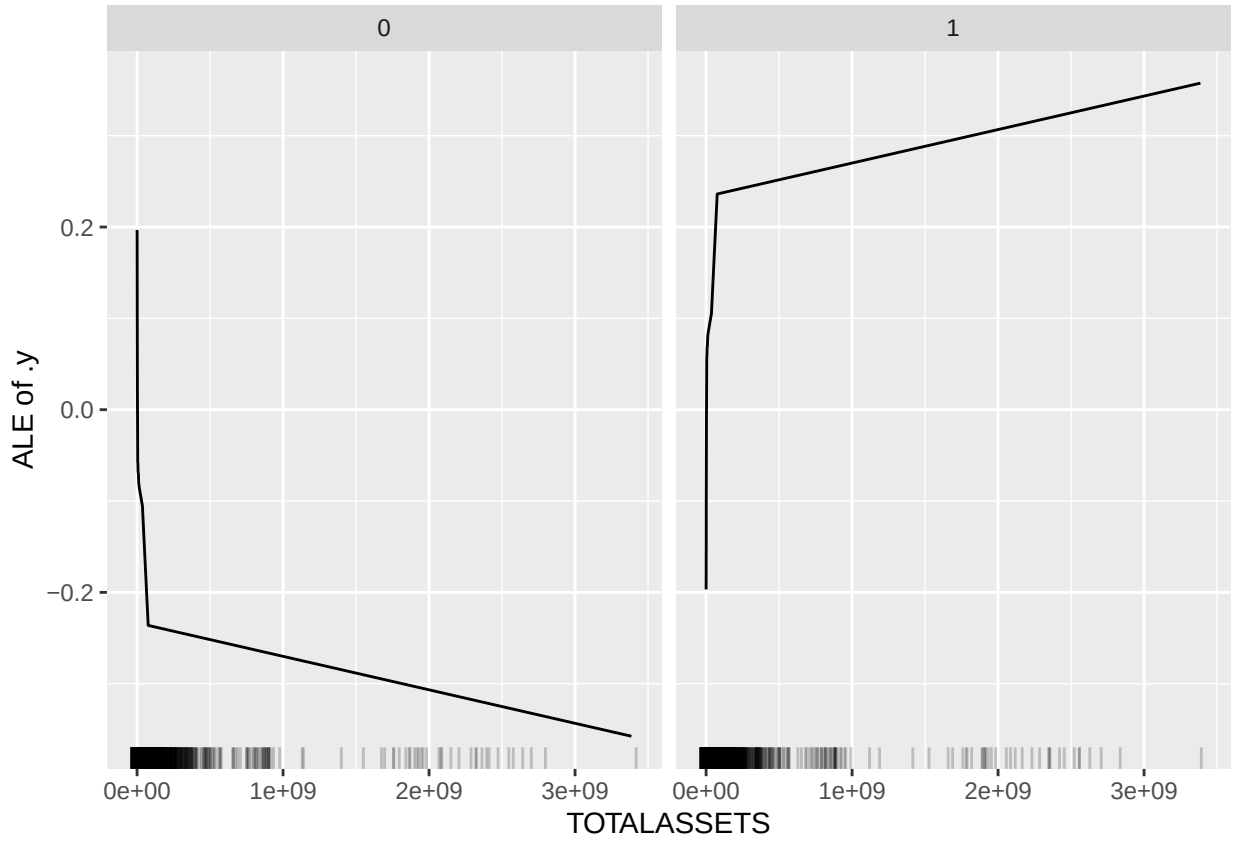
```
# EXPLAINABILITY VIA IML (INTERPRETABLE MACHINE LEARNING) PACKAGE
mod <- Predictor$new(rf_model_train,
                     data = training_data, y = training_data$HighS75perc)
#creating predictor object

#1. Feature importance - model level
imp <- FeatureImp$new(mod, loss = "ce", compare = "difference")
```

```
## Warning: package 'labeling' was built under R version 4.3.2
```

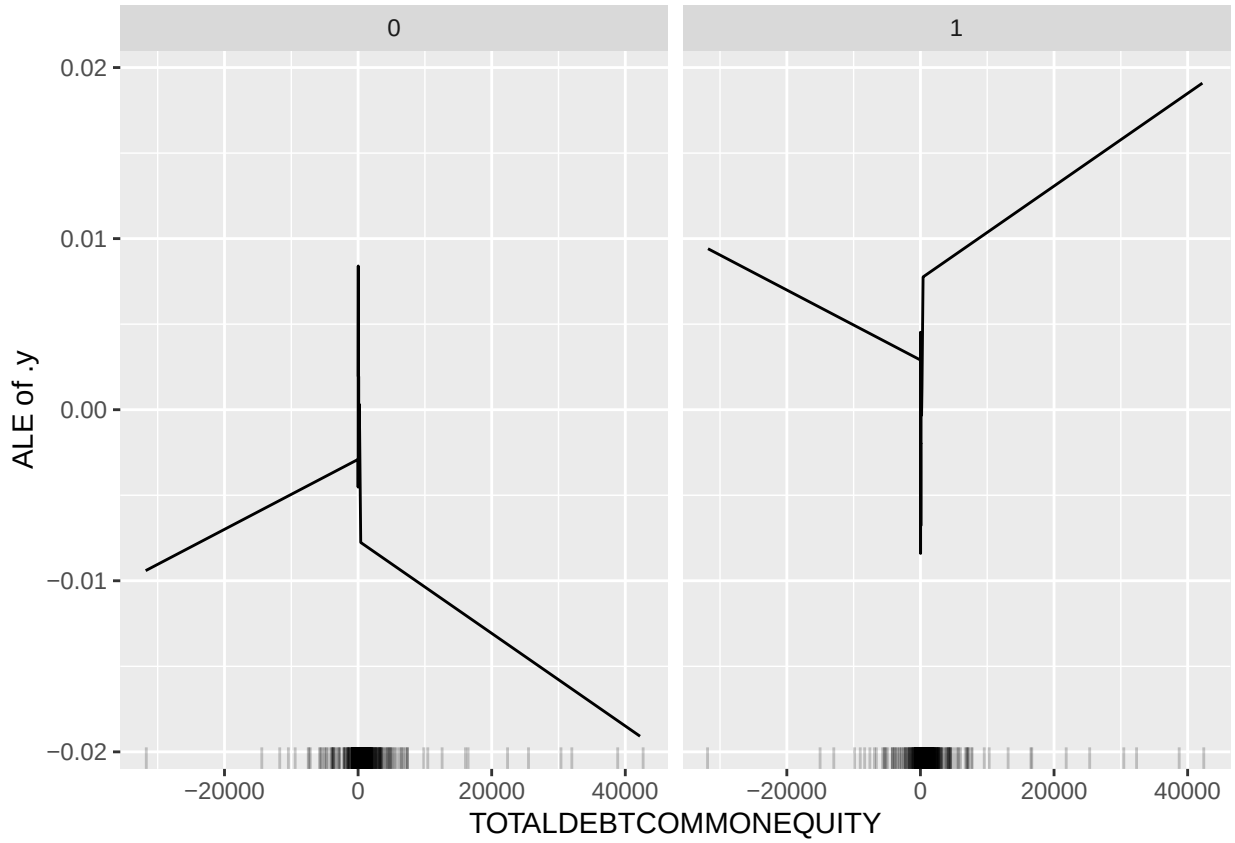
```
plot(imp)
```



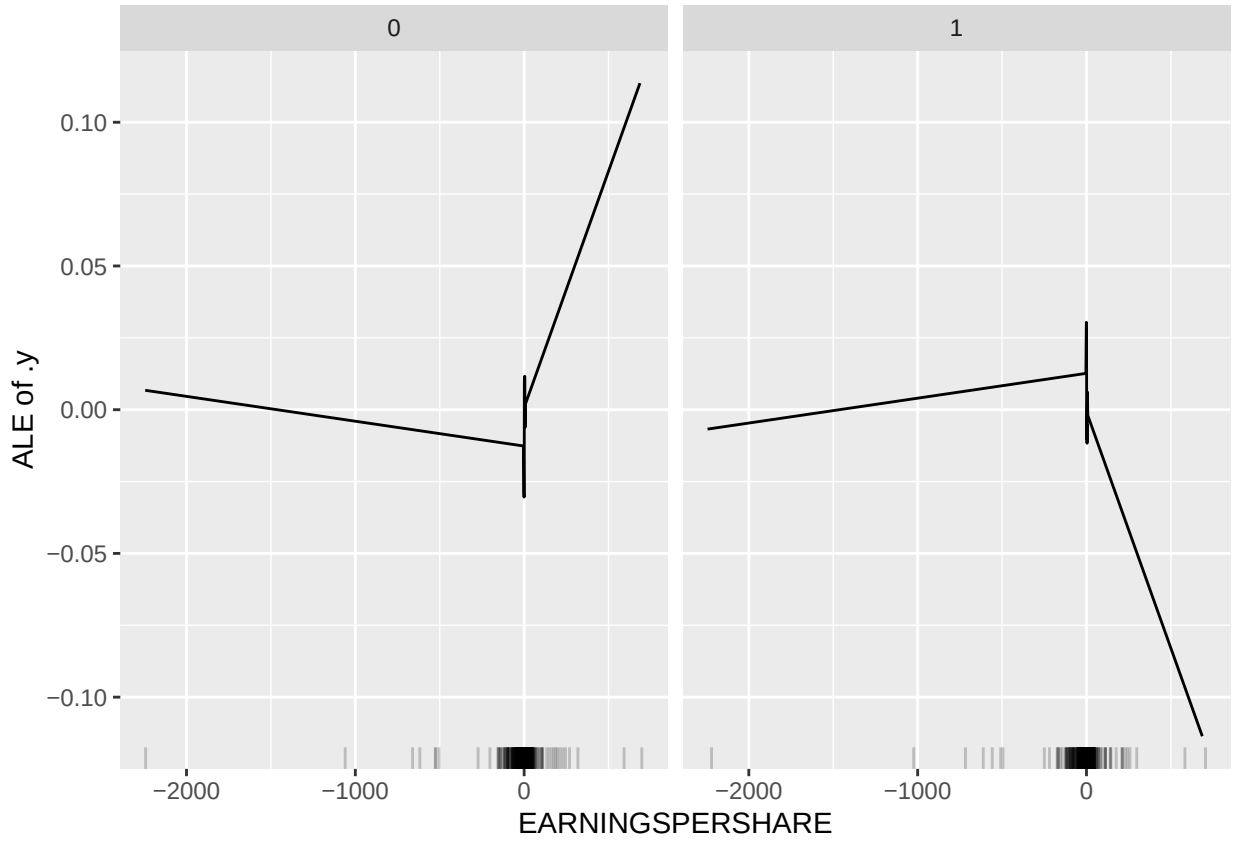

```
#2. Feature effects - model level
eff_1 <- FeatureEffect$new(mod, feature = "TOTALASSETS", method = 'ale')
plot(eff_1)
```



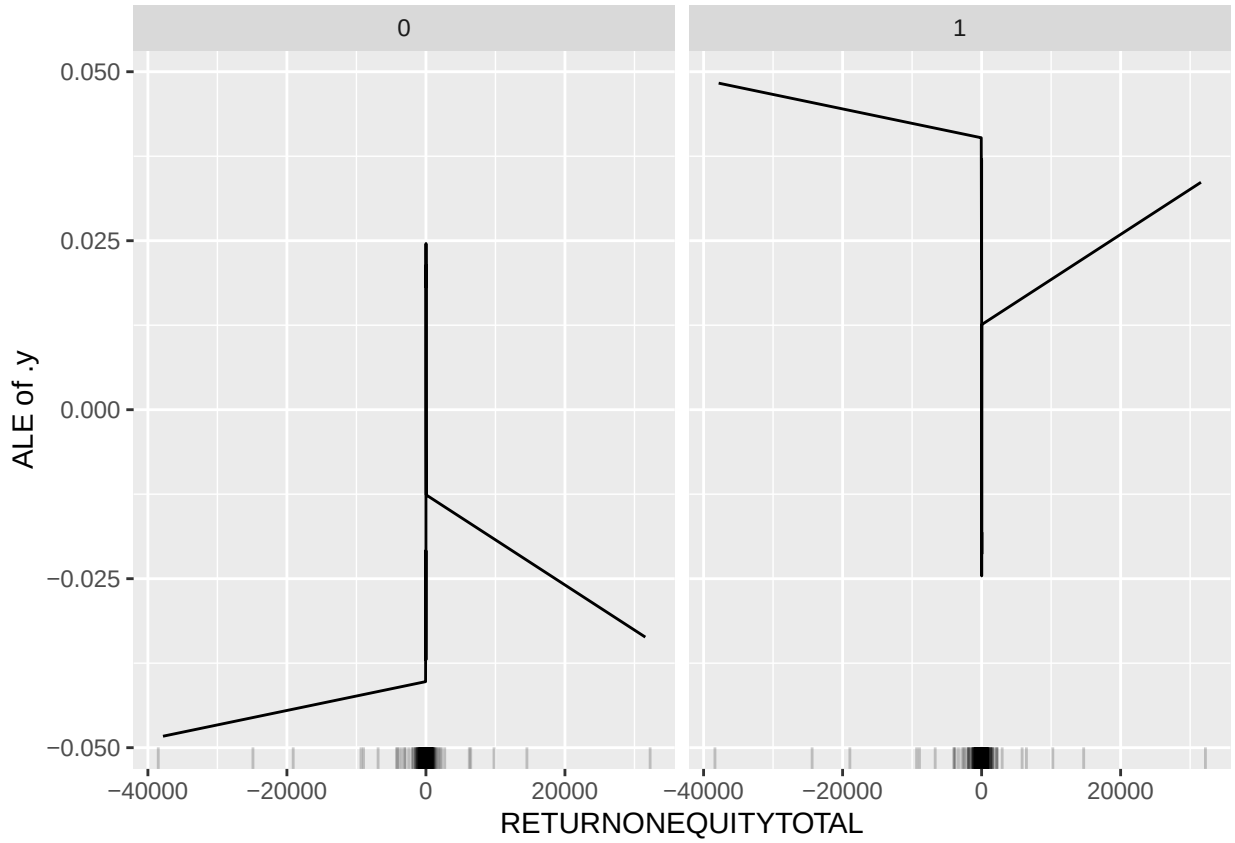

```
eff_2 <- FeatureEffect$new(mod, feature = "TOTALDEBTCOMMONEQUITY", method = 'ale')
plot(eff_2)
```



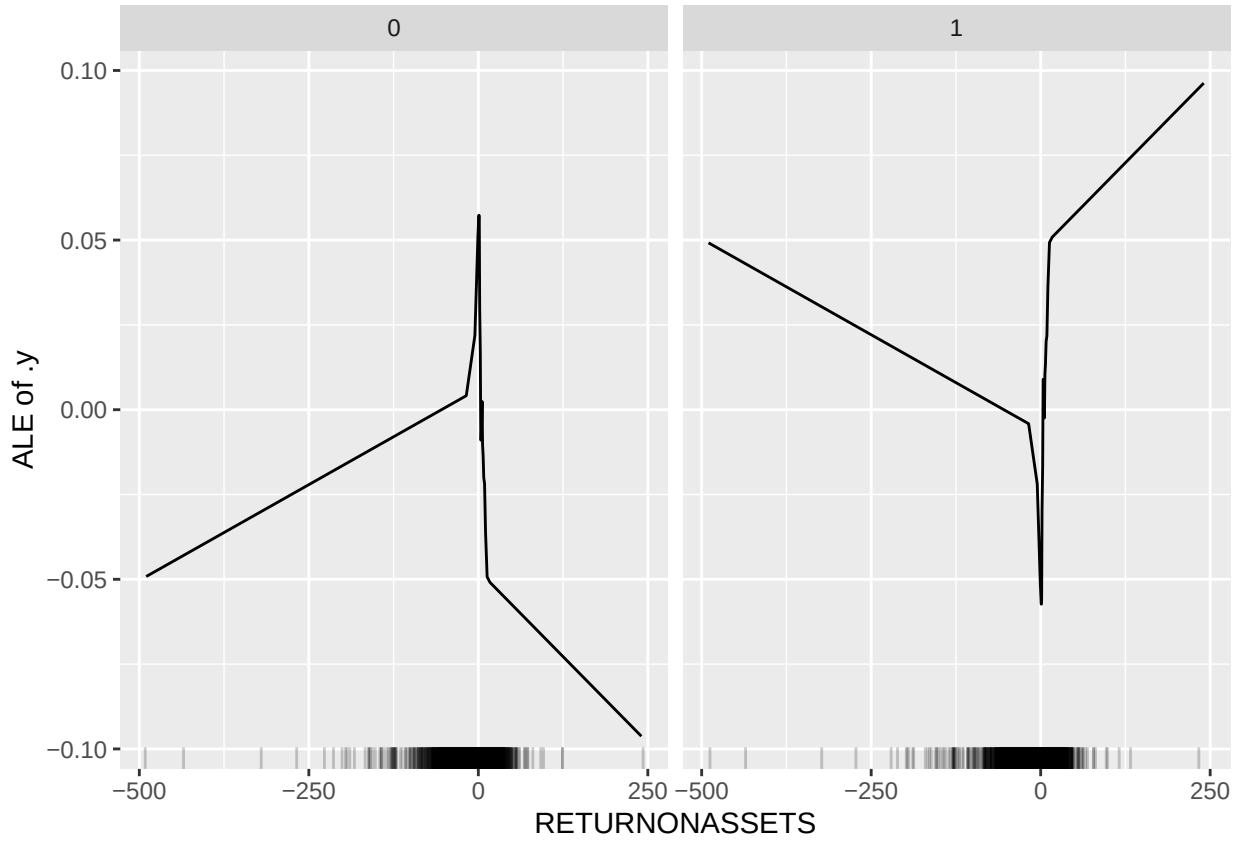

```
eff_3 <- FeatureEffect$new(mod, feature = "EARNINGSPERSHARE", method = 'ale')
plot(eff_3)
```



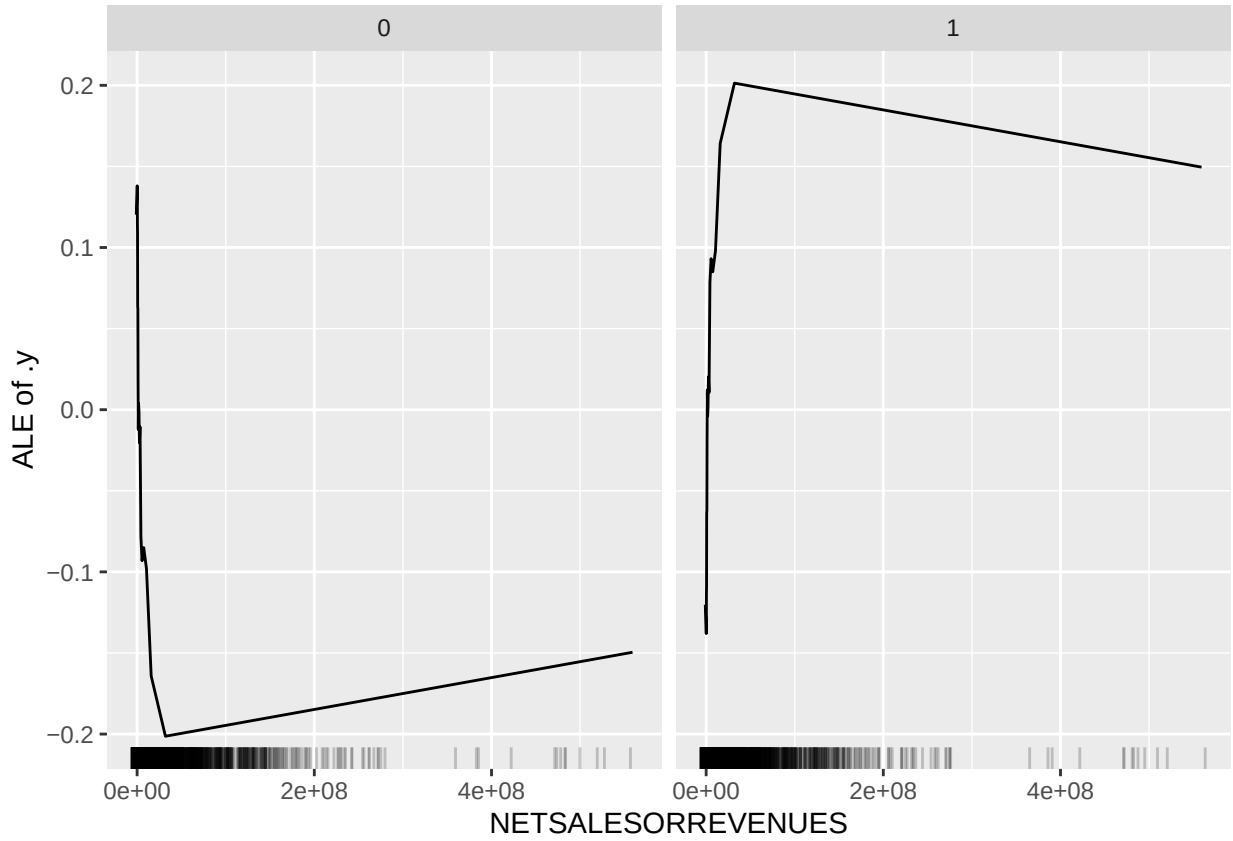

```
eff_4 <- FeatureEffect$new(mod, feature = "RETURNONEQUITYTOTAL", method = 'ale')
plot(eff_4)
```



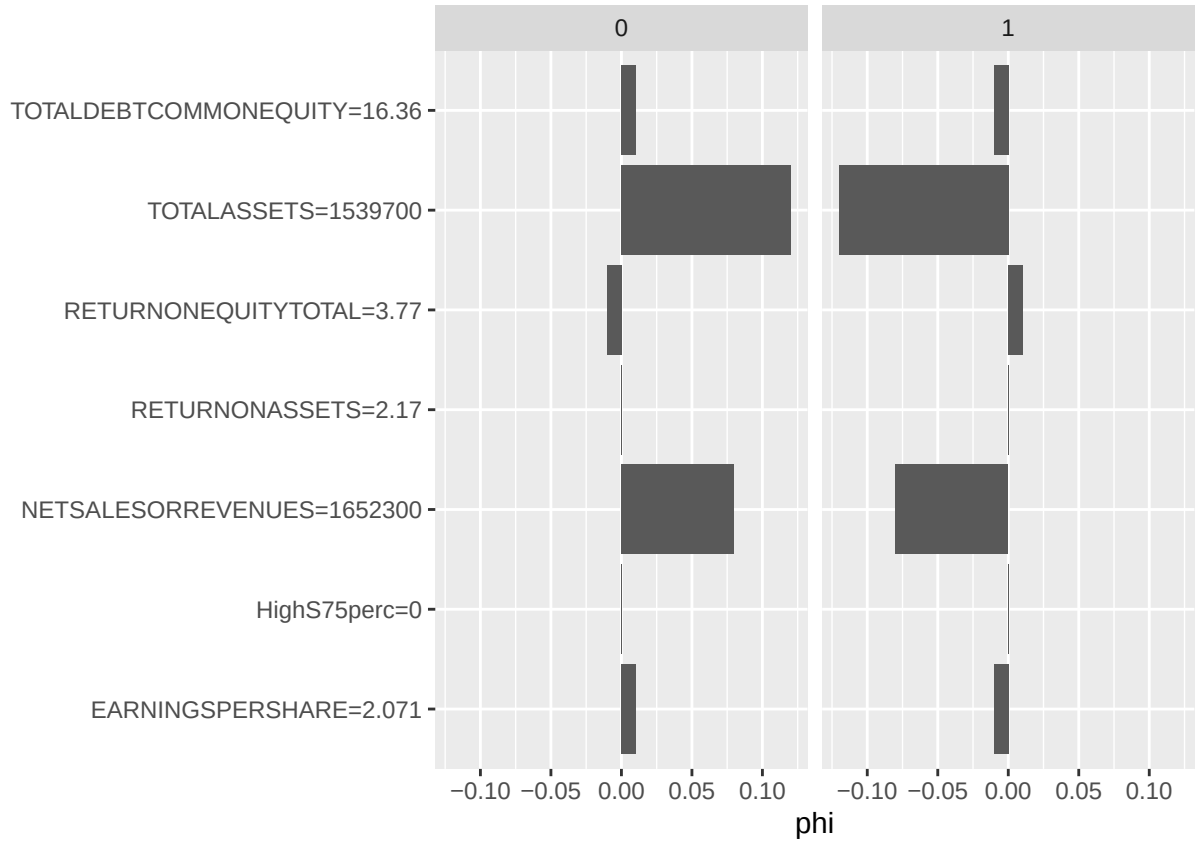

```
eff_5 <- FeatureEffect$new(mod, feature = "RETURNONASSETS", method = 'ale')
plot(eff_5)
```



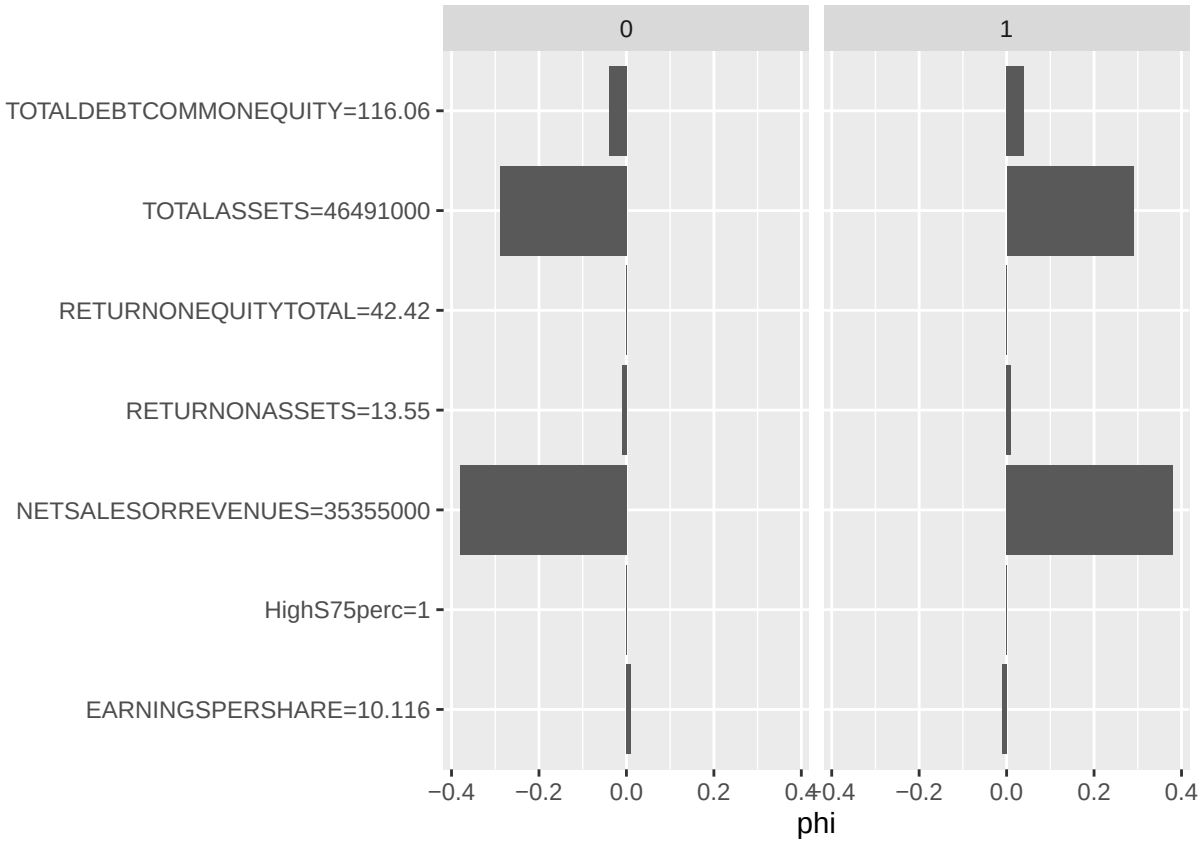

```
eff_6 <- FeatureEffect$new(mod, feature = "NETSALESORREVENUES", method = 'ale')
plot(eff_6)
```



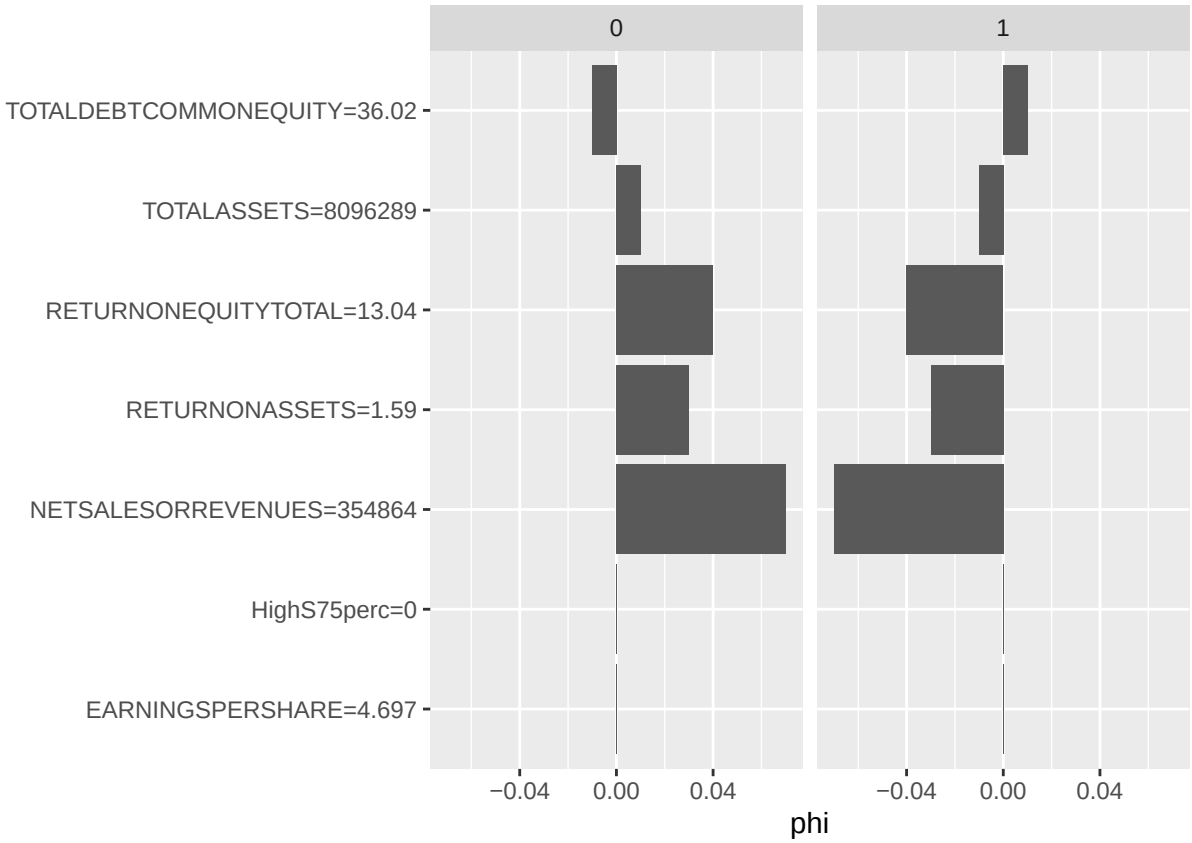

```
# 3. Shapley values - local predictions
shapley <- Shapley$new(mod, x.interest = testing_data[8,])
plot(shapley)
```



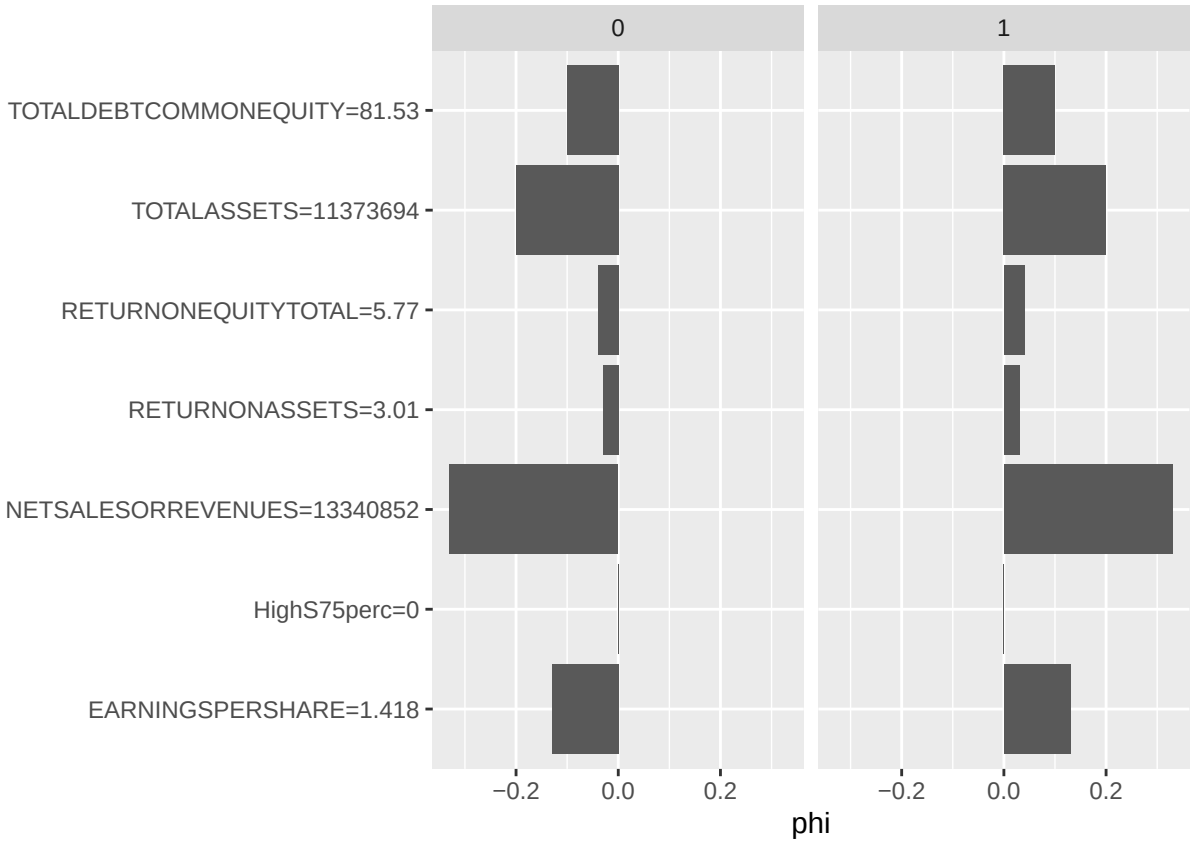

```
results <- data.frame(Row = 1:nrow(testing_data),
                      Actual = testing_data$HighS75perc,
                      Predicted = predicted_test, row.names = NULL)

results$Outcome <- with(results, ifelse(Actual == Predicted & Actual == "1",
                                        "TP",
                                        ifelse(Actual == Predicted & Actual == "0",
                                               "TN",
                                               ifelse(Actual != Predicted & Actual == "1",
                                                      "FN", "FP"))))

# one row from each scenario
TP_row <- subset(results, Outcome == "TP")$Row[1]
TN_row <- subset(results, Outcome == "TN")$Row[1]
FP_row <- subset(results, Outcome == "FP")$Row[1]
FN_row <- subset(results, Outcome == "FN")$Row[1]

predictor <- Predictor$new(rf_model_train,
                           data = testing_data, y = testing_data$HighS75perc)

plot_shapley_for_instance <- function(instance_index) {
  shapley <- Shapley$new(predictor, x.interest = testing_data[instance_index,])
  plot(shapley)
}

plot_shapley_for_instance(TP_row)
```



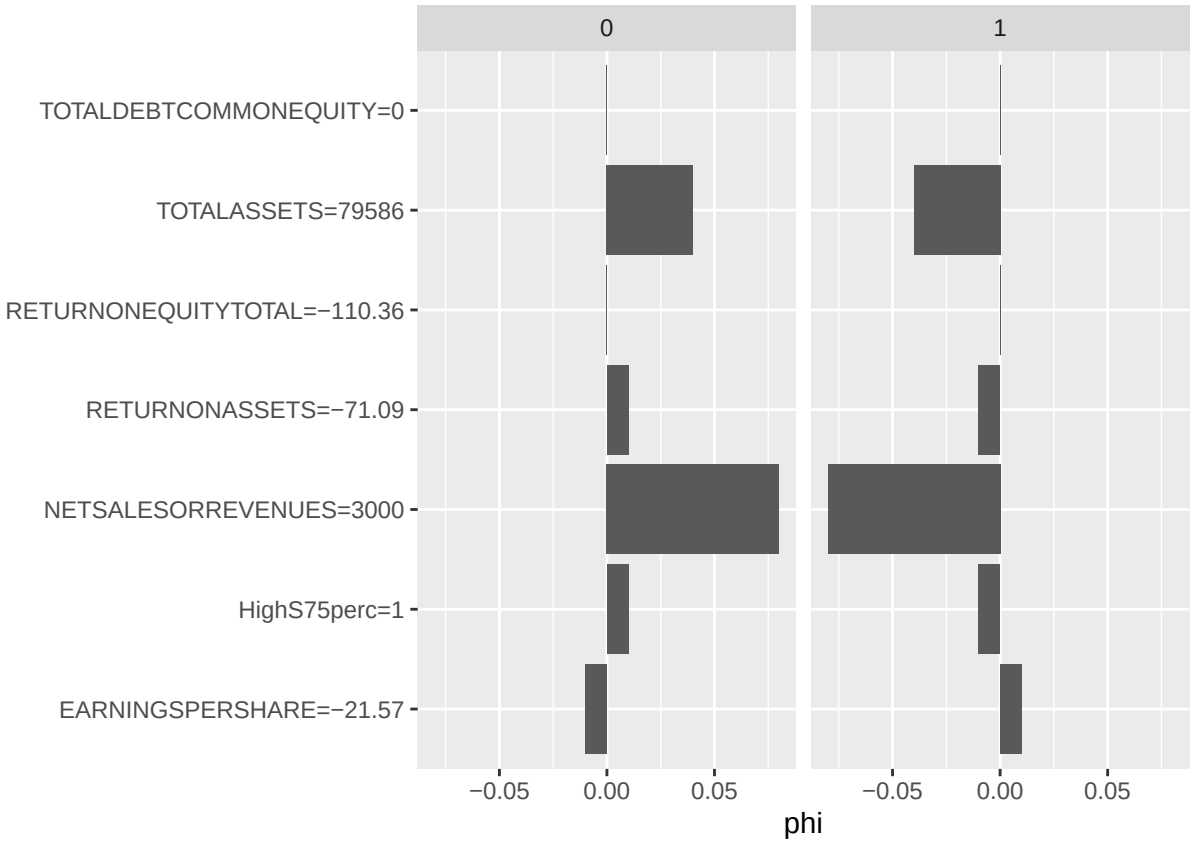

```
plot_shapley_for_instance(TN_row)
```




```
plot_shapley_for_instance(FP_row)
```




```
plot_shapley_for_instance(FN_row)
```


0
1
TOTALDEBTCOMMONEQUITY=0
TOTALASSETS=79586
RETURNONEQUITYTOTAL=−110.36
RETURNONASSETS=−71.09
NETSALESORREVENUES=3000
HighS75perc=1
EARNINGSPERSHARE=−21.57
−0.05
0.00
0.05
−0.05
0.00
0.05
phi