Assignment 3

Xiying Zhao

2023-04-21

```
setwd("C:/Users/zhaox/OneDrive/Desktop/UIUC/BADM 575/Assignment/hw3")
d = read.csv("Supplier_Disruption_Risk.csv")
```

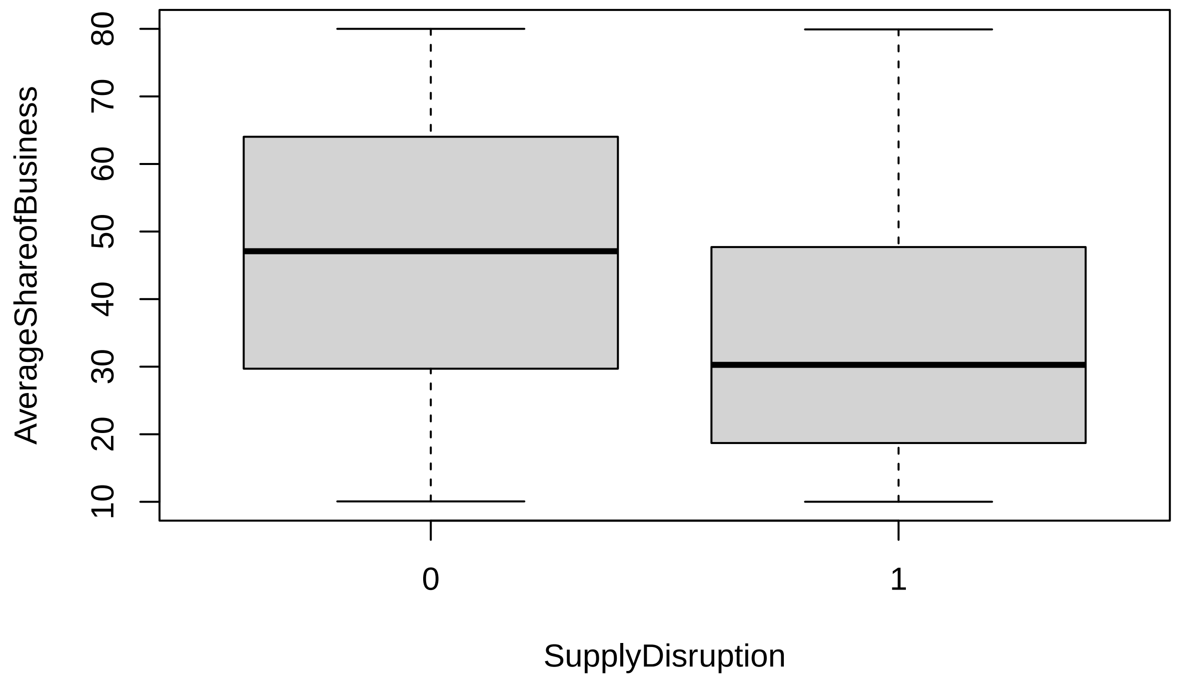
```
colnames(d)
```

```
##  [1] "SupplierID"                "NumberOfCustomers"
##  [3] "AverageShareofBusiness"    "AverageSizeofSuppliers"
##  [5] "AverageSizeofCustomers"    "CAGRBusiness"
##  [7] "TechnologyInvestment"      "CountryGDP"
##  [9] "CountryGDPGrowth"          "IndexOfPoliticalTurmoil"
## [11] "IndexOfSocialTurmoil"      "NumberofProducts"
## [13] "NumberofNewProducts"       "ProfitabilityLast5Years"
## [15] "SupplyDisruption"
```

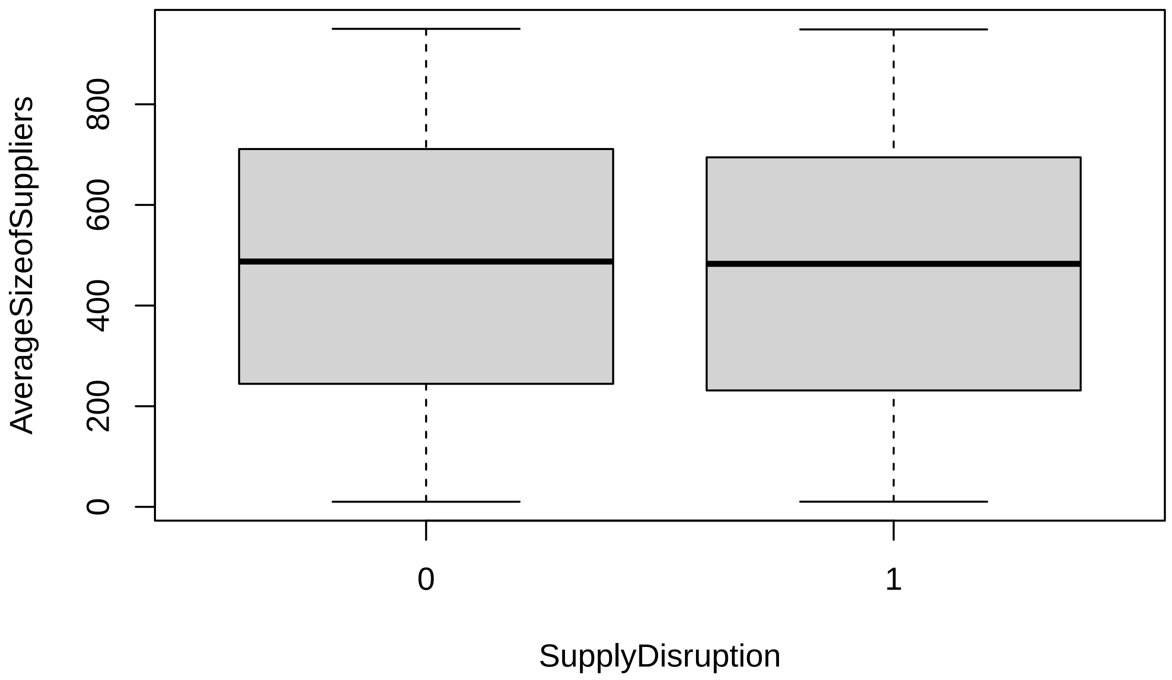
Question 1. Graphical Exploration of the Data

1. Create box plots for all continuous variables against supply chain disruption. One side-by-side box plot for class 0 and class 1 is required.

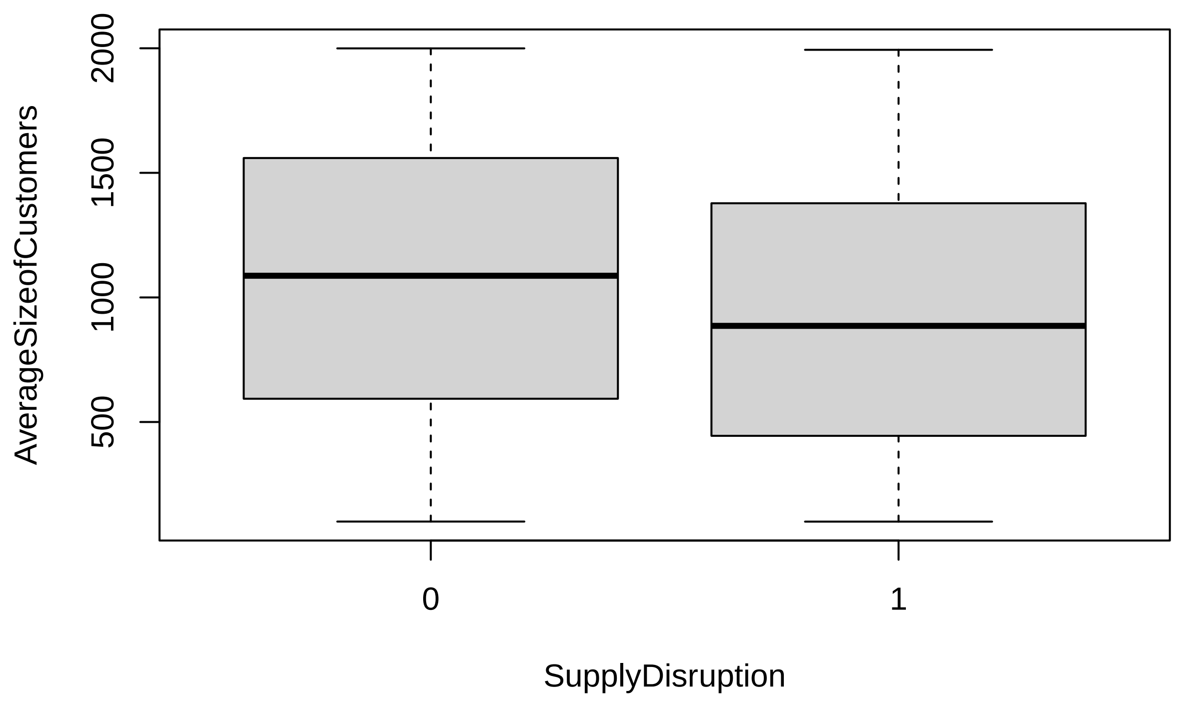
```
boxplot(AverageShareofBusiness~SupplyDisruption, data = d)
```



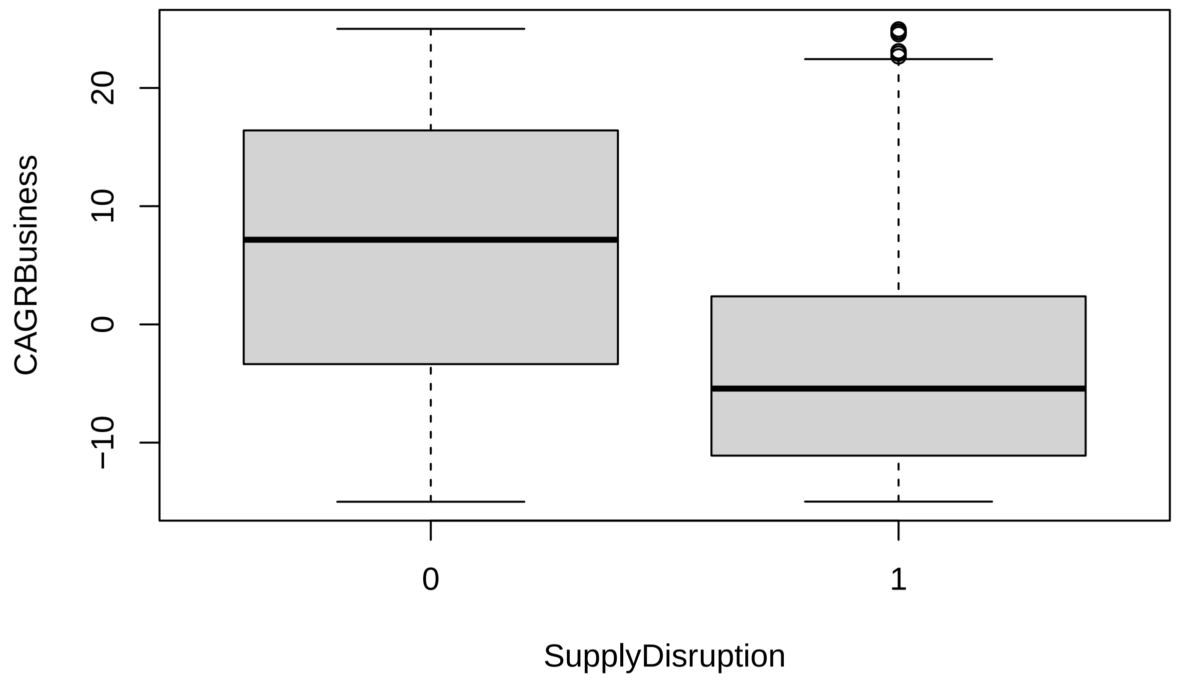
```
boxplot(AverageSizeofSuppliers~SupplyDisruption, data = d)
```



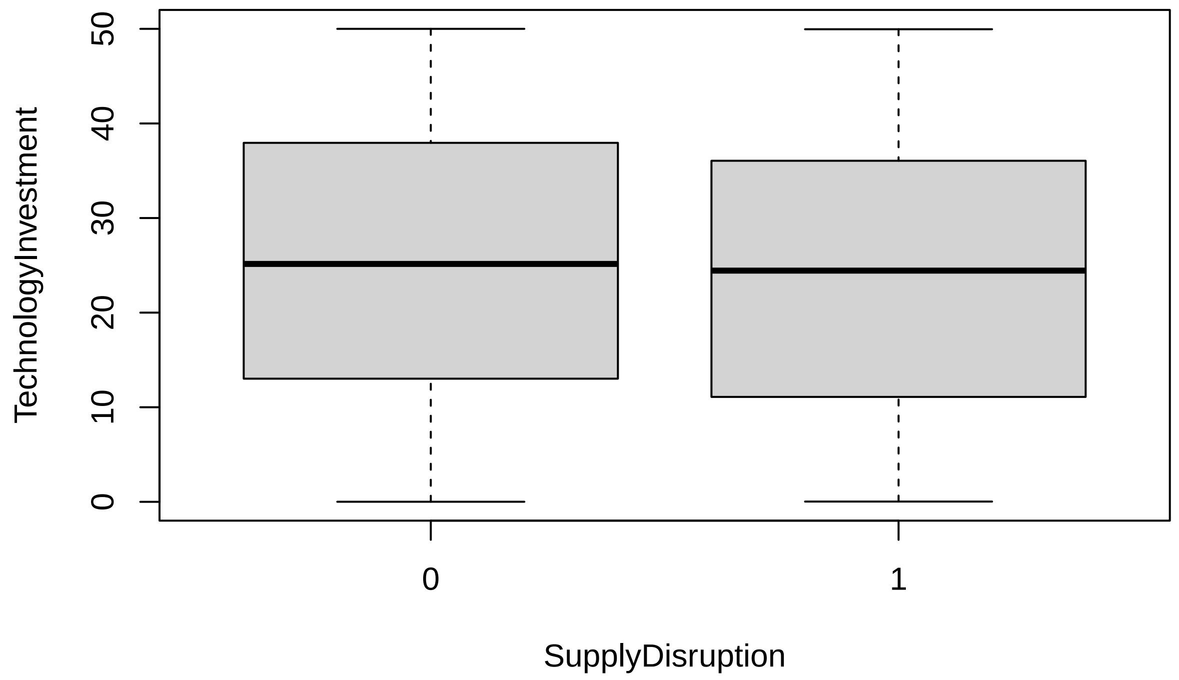
```
boxplot(AverageSizeofCustomers~SupplyDisruption, data = d)
```



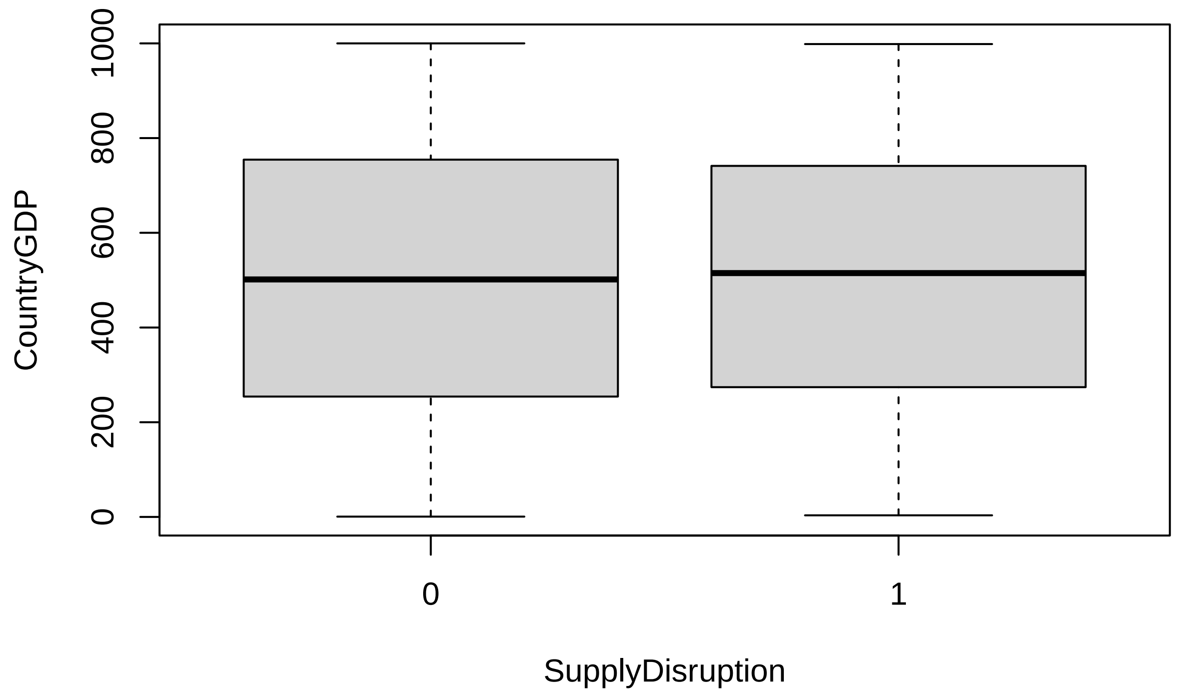
```r
boxplot(CAGRBusiness~SupplyDisruption, data = d)
```



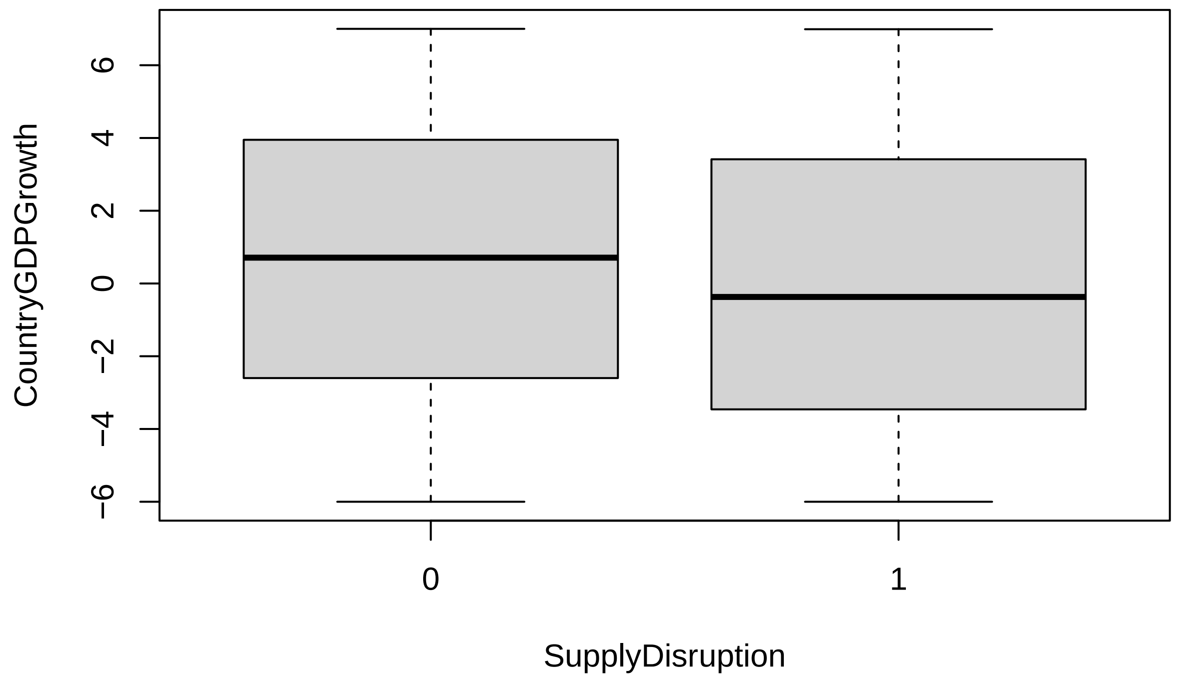
```
boxplot(TechnologyInvestment~SupplyDisruption, data = d)
```



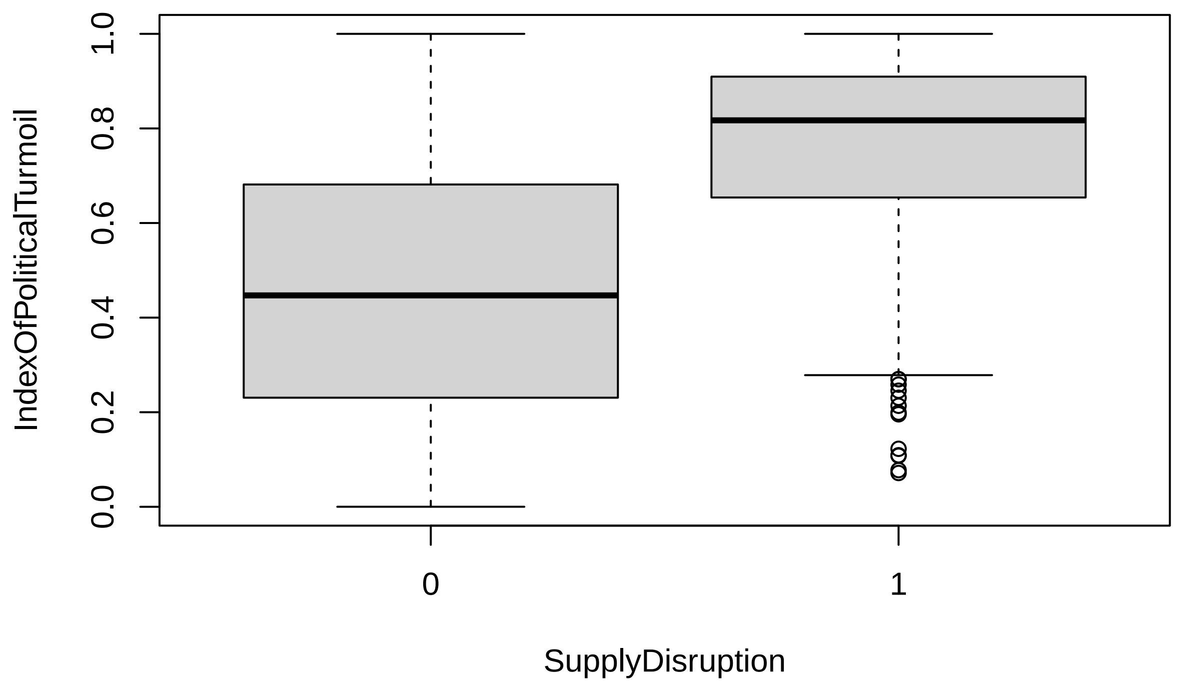
```
boxplot(CountryGDP~SupplyDisruption, data = d)
```



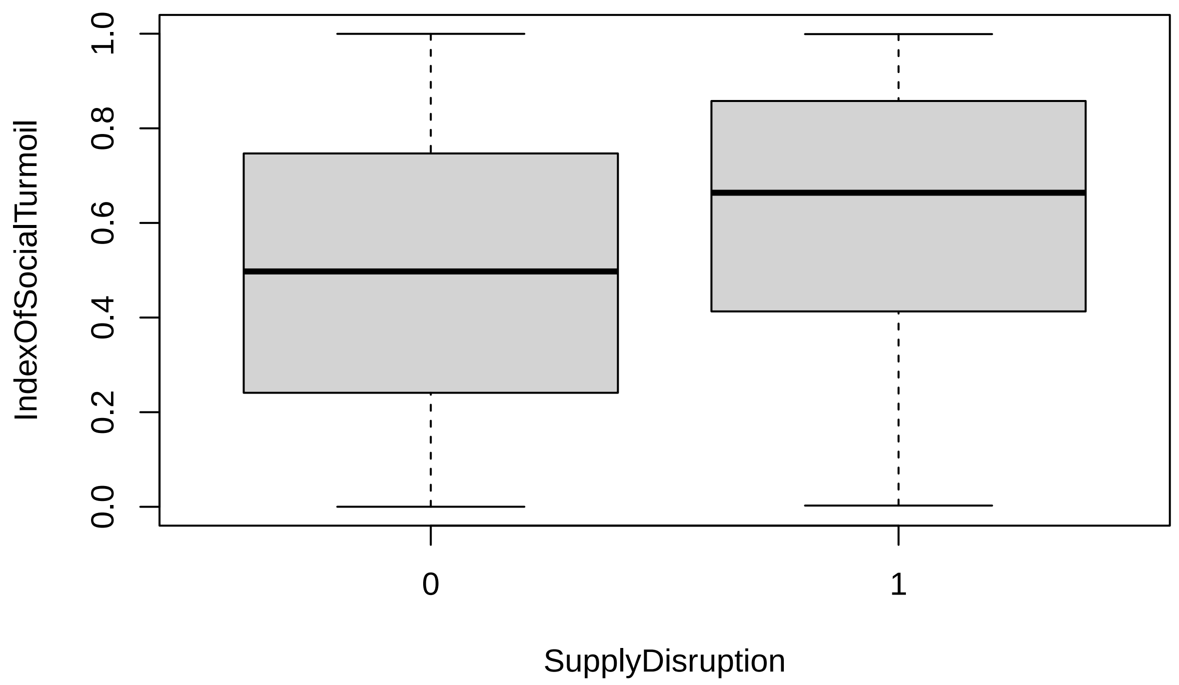
```
boxplot(CountryGDPGrowth~SupplyDisruption, data = d)
```



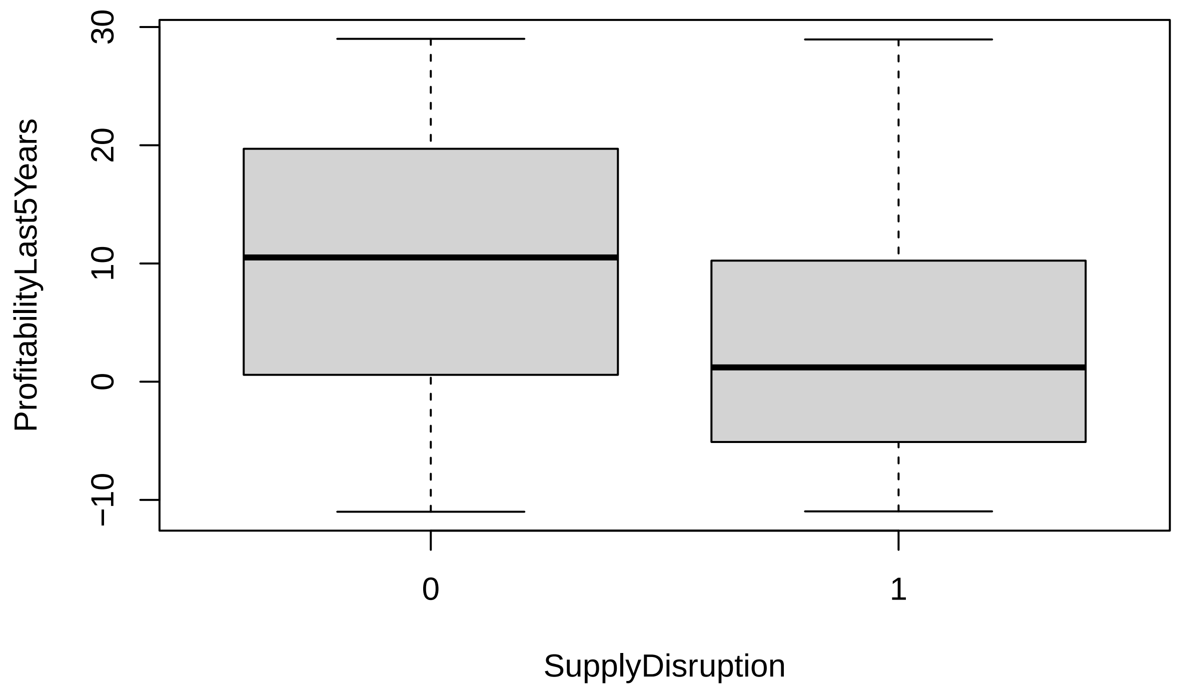
```
boxplot(IndexOfPoliticalTurmoil~SupplyDisruption, data = d)
```



```
boxplot(IndexOfSocialTurmoil~SupplyDisruption, data = d)
```



```
boxplot(ProfitabilityLast5Years~SupplyDisruption, data = d)
```



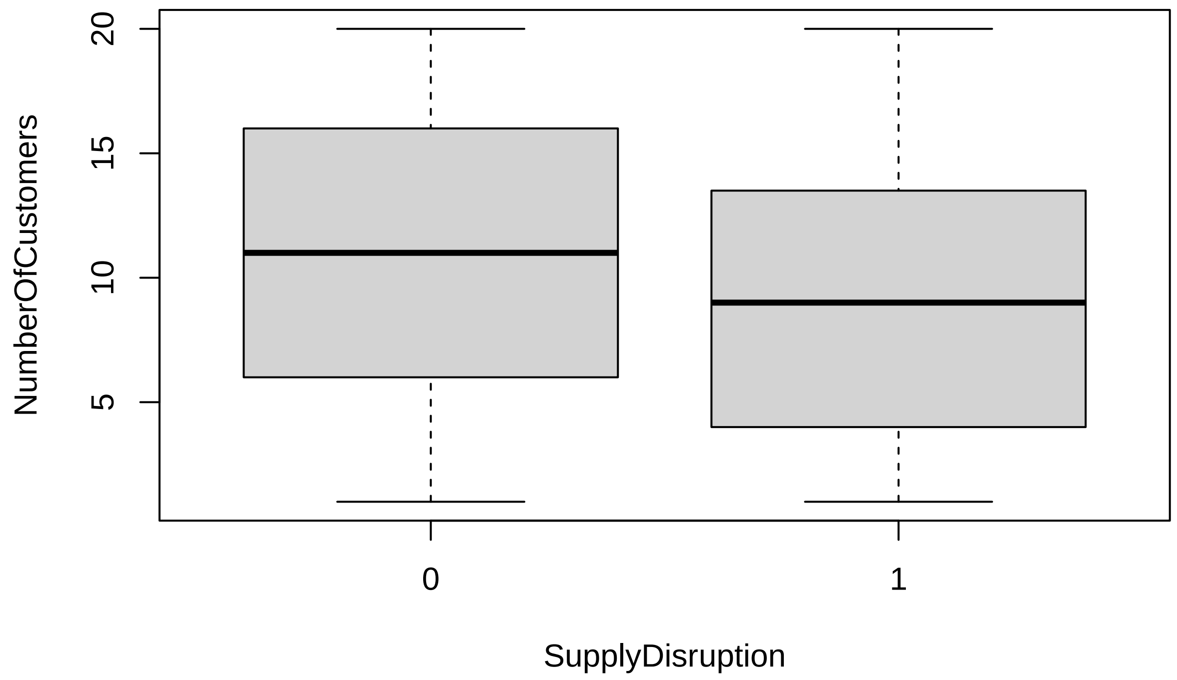
2. Comment on each of the continuous variables' ability to discriminate between the two classes.

Among the 10 boxplots, 8 demonstrate the capability to distinguish between the two classes, as evidenced by the difference in ranges and medians between class 0 and 1.

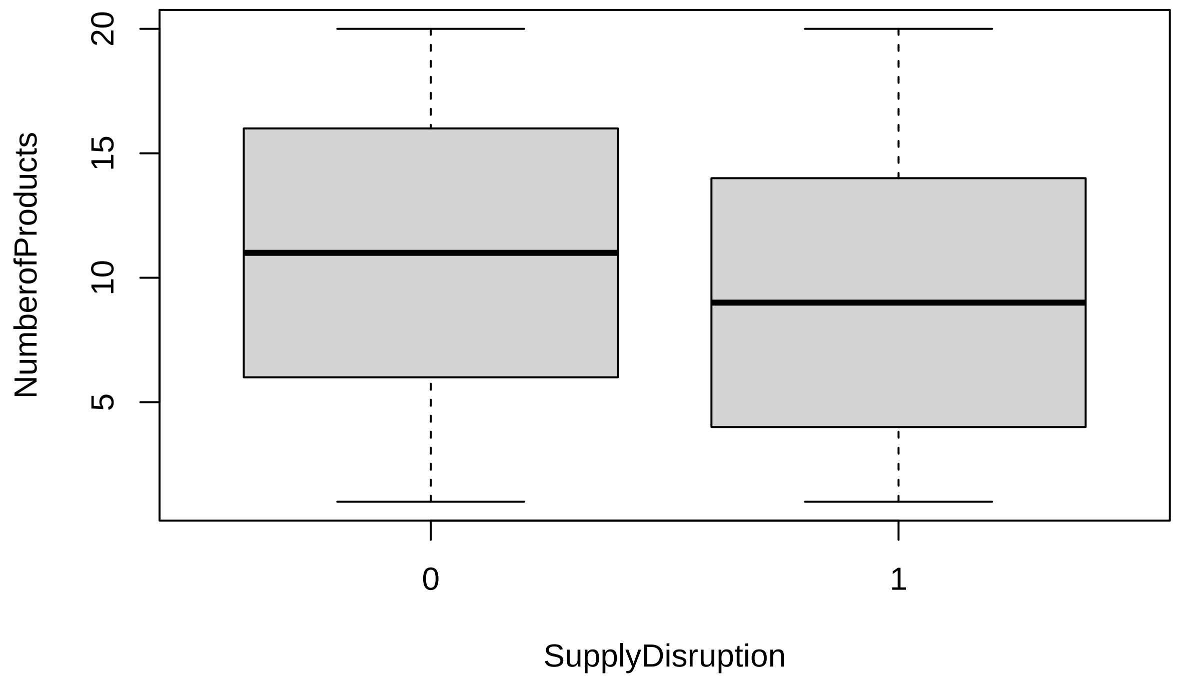
However, the discriminatory power is limited in the case of #2 (Average Size of Suppliers), #5 (Technology Investment), and #6 (CountryGDP) since the ranges and medians between the two classes are largely identical, particularly for #6.

3. Select at least eight variables that you feel are most discriminative. This need not be exact. For the rest of the questions, you will use these variables. Note that your selection may be different from others' selections. That is perfectly fine.

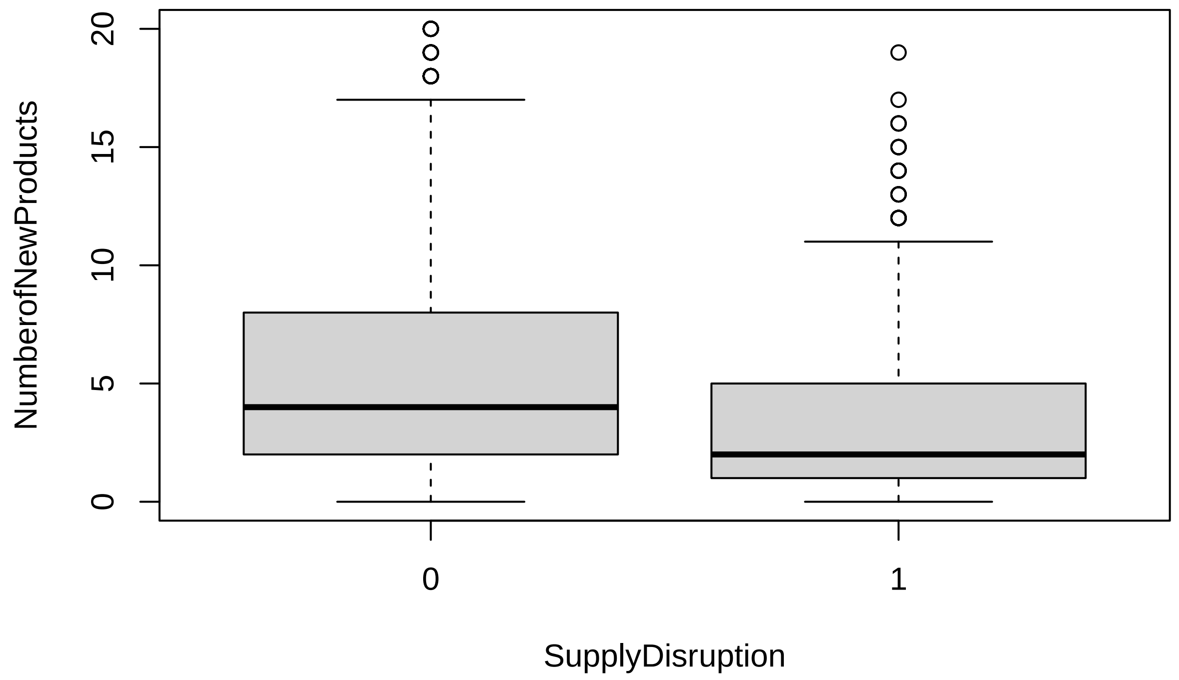
```r
# Discrete Variables
boxplot(NumberOfCustomers~SupplyDisruption, data = d)
```



```
boxplot(NumberofProducts~SupplyDisruption, data = d)
```



```
boxplot(NumberofNewProducts~SupplyDisruption, data = d)
```



Given that CountryGDPGrowth and CountryGDP are both GDP-related variables, we can exclude CountryGDP from consideration for SupplyDisruption as it had minimal impact. Additionally, the boxplot for continuous variables in last question suggests that the AverageSizeofSuppliers variable is not effective in discriminating between the two classes and therefore it will not be included.

As a result, we will select eight continuous variables to analyze: AverageShareofBusiness, AverageSizeofCustomers, CAGRBusiness, TechnologyInvestment, CountryGDPGrowth, IndexOfPoliticalTurmoil, IndexOfSocialTurmoil, and ProfitabilityLast5Years.

Question 2. Data Preparation.

1. Run the command set.seed(1234). Then randomly select 80% of the observations in training dataset (call 'train') and the rest of the observations in testing dataset (call 'test').

```
# set seed for reproducing the partition
set.seed(1234)
# row numbers of the training set
train.index <- sample(c(1:dim(d)[1]), dim(d)[1]*0.8)
# training set
train <- d[train.index,]
# test set
test <- d[-train.index,]
```

2. For both training and testing datasets, only retain the variables that you selected in the first question.

```
train <- train[-c(1,2,4,8,12,13)]
test <- test[-c(1,2,4,8,12,13)]
```

Question 3. Run classification models and predict.

1. Fit a logistic regression, random forest (with 500 trees), support vector machine (with 'radial basis' kernel) and neural network (with only two hidden layers, each hidden layer having the same number of nodes as the input data, set the other parameters the same as the class example). Use train sample to fit all models.

```


ml = glm(SupplyDisruption~ AverageShareofBusiness+ AverageSizeofCustomers+CAGRBusiness+TechnologyInvestm
summary(ml)
```

1.1 Logistic Regression

```
##
## Call:
## glm(formula = SupplyDisruption ~ AverageShareofBusiness + AverageSizeofCustomers +
##     CAGRBusiness + TechnologyInvestment + CountryGDPGrowth +
##     IndexOfPoliticalTurmoil + IndexOfSocialTurmoil + ProfitabilityLast5Years,
##     family = "binomial", data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.53446  -0.23051  -0.06404  -0.01211   2.74859
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)            -4.2008876  0.3503439 -11.991  < 2e-16 ***
## AverageShareofBusiness -0.0680690  0.0043893 -15.508  < 2e-16 ***
## AverageSizeofCustomers -0.0013679  0.0001406  -9.731  < 2e-16 ***
## CAGRBusiness           -0.1521800  0.0085531 -17.792  < 2e-16 ***
## TechnologyInvestment   -0.0266882  0.0049580  -5.383 7.33e-08 ***
## CountryGDPGrowth       -0.0783803  0.0186832  -4.195 2.73e-05 ***
## IndexOfPoliticalTurmoil  9.1251197  0.4528645  20.150  < 2e-16 ***
## IndexOfSocialTurmoil     3.4335436  0.2795091  12.284  < 2e-16 ***
## ProfitabilityLast5Years -0.1094087  0.0074351 -14.715  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3068.2  on 3999  degrees of freedom
## Residual deviance: 1327.2  on 3991  degrees of freedom
## AIC: 1345.2
##
## Number of Fisher Scoring iterations: 8
```

```
library(randomForest)
```

1.2 Random Forest (with 500 trees)

```
## Warning: package 'randomForest' was built under R version 4.2.3

## randomForest 4.7-1.1

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

```
set.seed(1234)
train$SupplyDisruption = as.factor(train$SupplyDisruption)
mr = randomForest(SupplyDisruption~ AverageShareofBusiness+ AverageSizeofCustomers+CAGRBusiness+Technolo
mr
```

```
##
## Call:
##  randomForest(formula = SupplyDisruption ~ AverageShareofBusiness +      AverageSizeofCustomers + CAG
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 8.6%
## Confusion matrix:
##      0   1 class.error
## 0 3437  49  0.01405622
## 1  295 219  0.57392996
```

```
library(e1071)
```

1.3 Support vector machine (with 'radial basis' kernel)

```
## Warning: package 'e1071' was built under R version 4.2.3
```

```
ms = svm(SupplyDisruption~ AverageShareofBusiness+ AverageSizeofCustomers+CAGRBusiness+TechnologyInvest
ms
```

```
##
## Call:
## svm(formula = SupplyDisruption ~ AverageShareofBusiness + AverageSizeofCustomers +
##     CAGRBusiness + TechnologyInvestment + CountryGDPGrowth + IndexOfPoliticalTurmoil +
##     IndexOfSocialTurmoil + ProfitabilityLast5Years, data = train,
##     type = "C-classification", probability = TRUE, kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  823
```

```
# Scale data for neuralnet model performance
max = apply(d , 2 , max)
min = apply(d, 2 , min)
scaled = as.data.frame(scale(d, center = min, scale = max - min))

# create train test set
set.seed(1234)
# row numbers of the training set
trainNN.index <- sample(c(1:dim(d)[1]), dim(d)[1]*0.8)
# training set
trainNN <- scaled[trainNN.index,]
# test set
testNN <- scaled[-trainNN.index,]
```

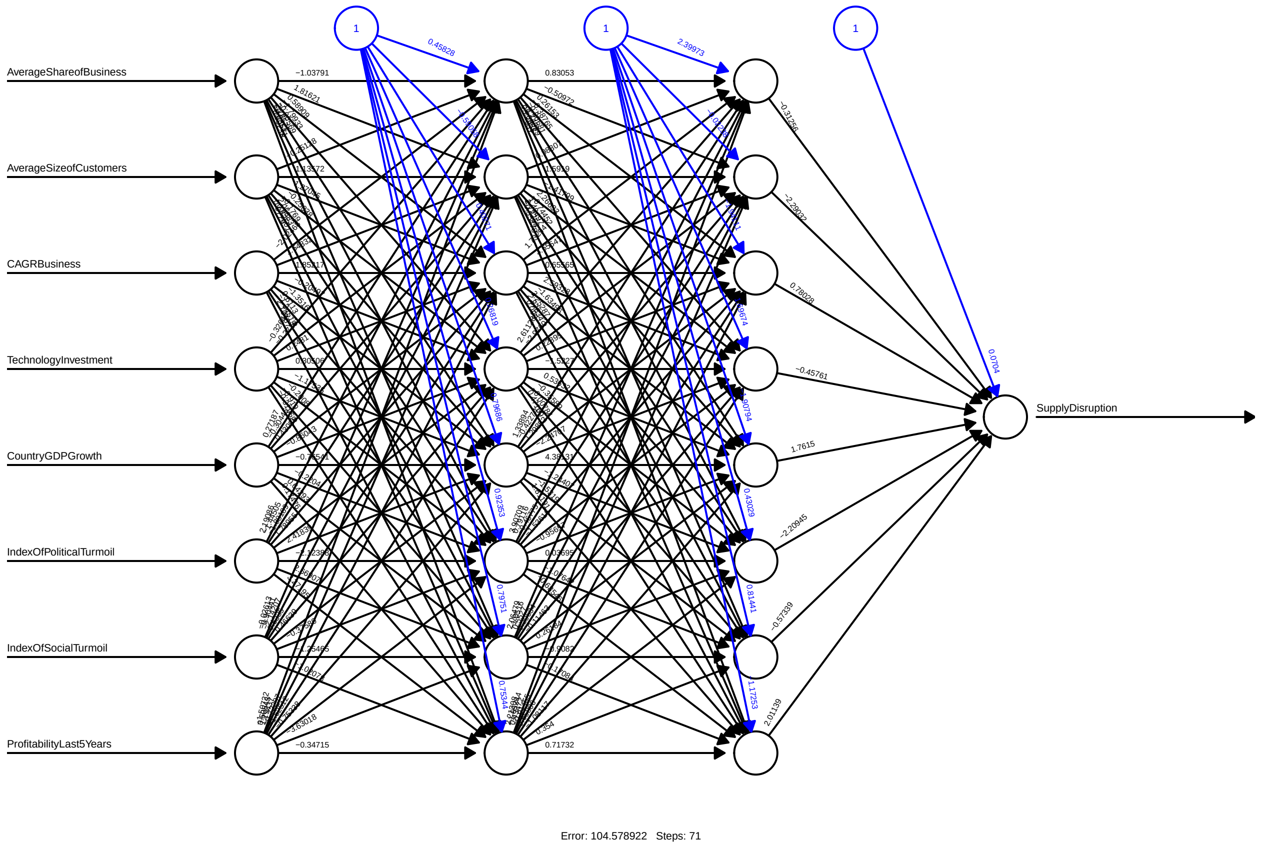
```
library(neuralnet)
```

1.4 neural network (with only two hidden layers, each hidden layer having the same number of nodes as the input data

```
## Warning: package 'neuralnet' was built under R version 4.2.3
```

```
mn = neuralnet(SupplyDisruption~AverageShareofBusiness+ AverageSizeofCustomers+ CAGRBusiness+ Technolog
```

```
plot(mn,rep = "best", cex = 0.3, width=15, height=10, mar = c(8, 8, 8, 8))
```

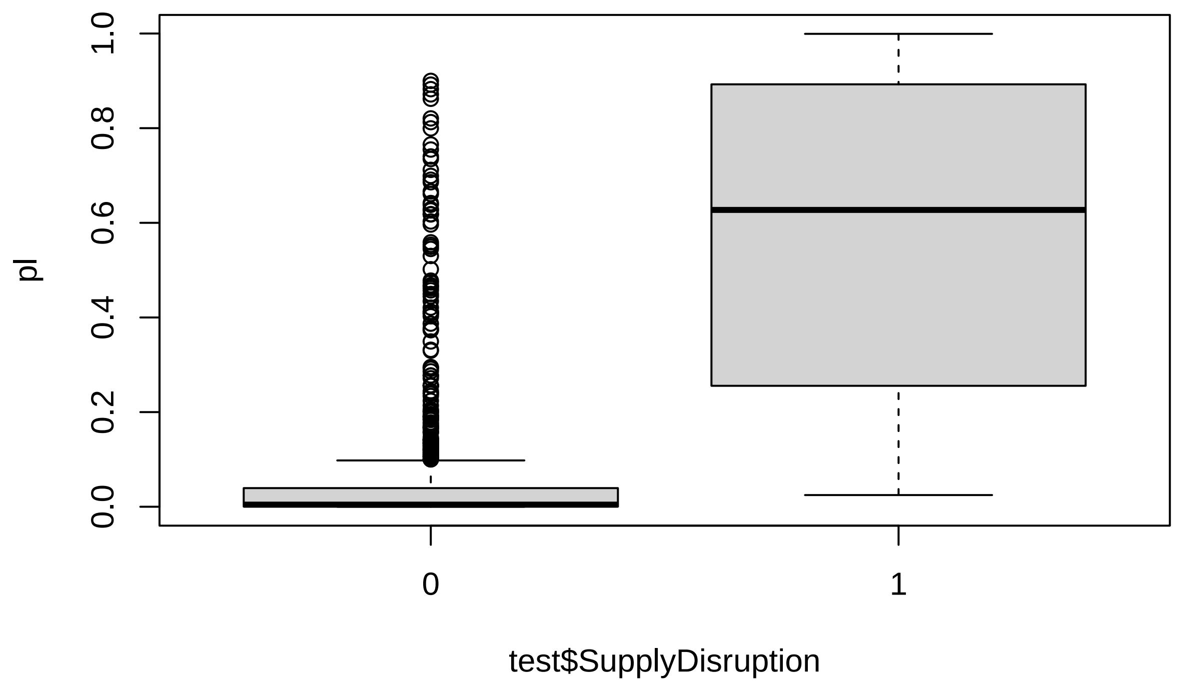


2. For each model predict the probability of supply chain disruption (class 1) in the test data.

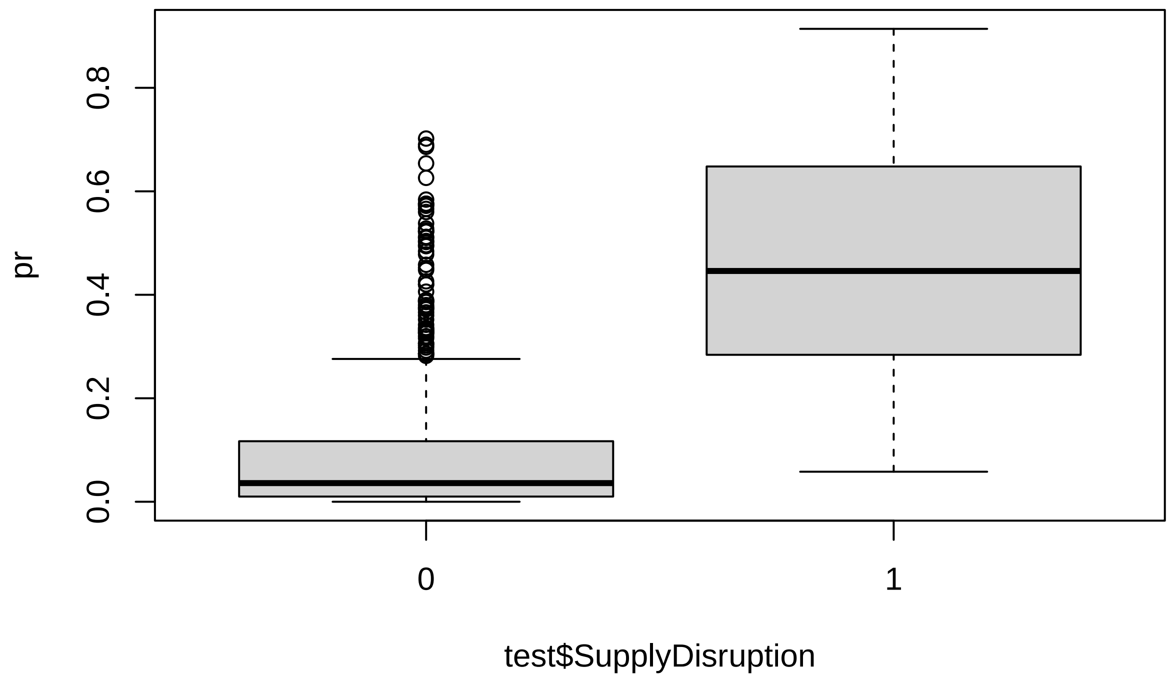
```
# Logistic Regression
pl=predict(ml, newdata = test, type = "response")
# Random Forest
test$SupplyDisruption = as.factor(test$SupplyDisruption)
pr=predict(mr, newdata = test, type = "prob")[, 2]
# Support vector machine
ps=predict(ms, newdata = test,probability = TRUE)
ps= attr(ps, "probabilities")[, 2]
# neural network
pn=predict(mn, newdata = testNN,type="prob")
```

3. For the predicted probabilities for all the four models, create a box plot (for each model) of the predicted probabilities against the test sample classes. Comment on the discriminative capacity of each model.

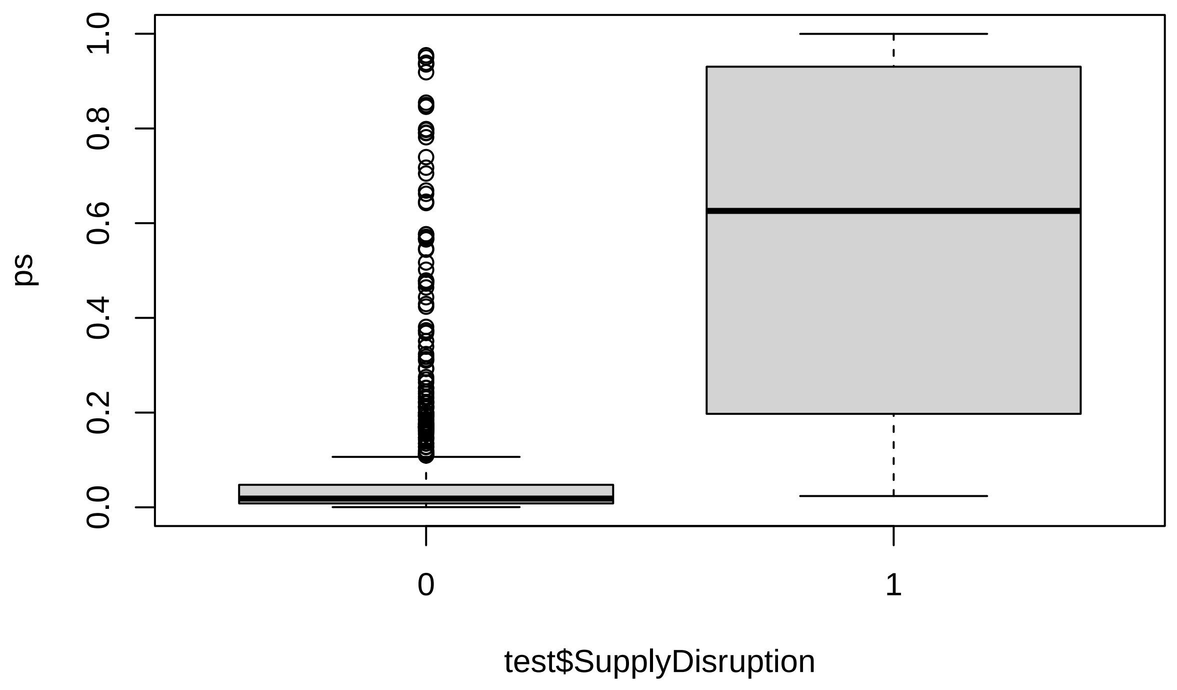
```
# Logistic Regression
boxplot(pl~test$SupplyDisruption)
```



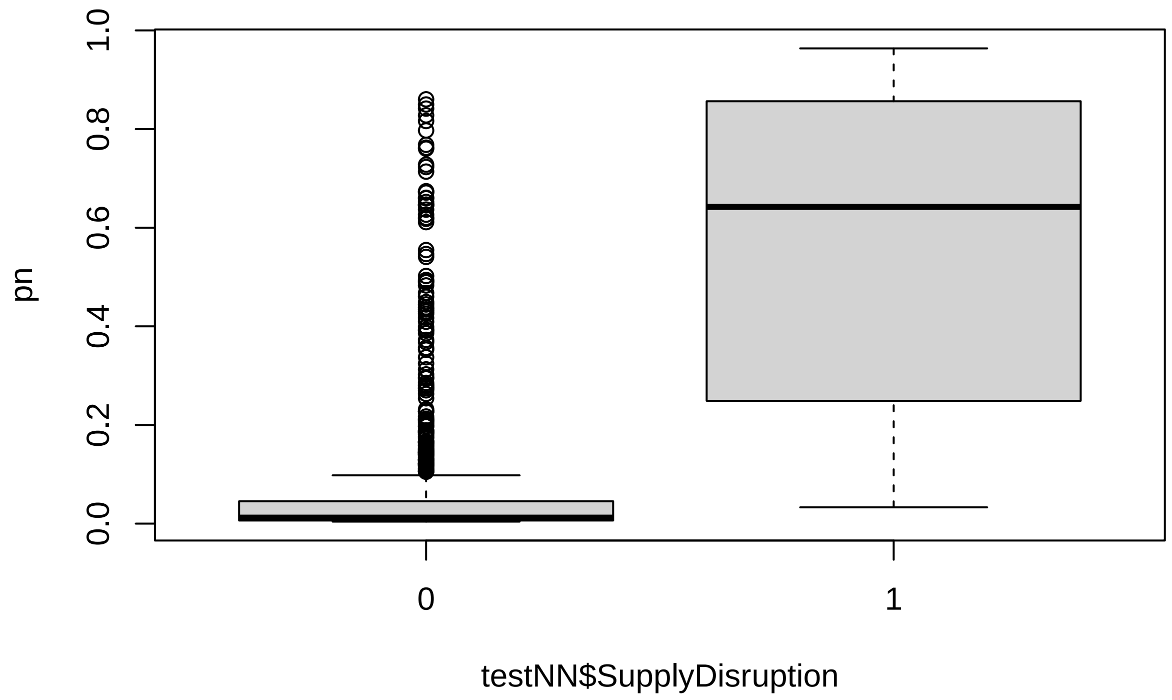
```
# Random Forest
boxplot(pr~test$SupplyDisruption)
```



```
# Support vector machine
boxplot(ps~test$SupplyDisruption)
```



```
# neural network
boxplot(pn~testNN$SupplyDisruption)
```



Observations:

The random forest model's boxplot demonstrates the most exceptional discriminatory capability since there is no overlap between the ranges of the two classes, and it has minimal outliers.

In contrast, the support vector machine model's boxplot has the poorest discriminatory capacity since the ranges overlap each other.

The linear regression and neural net model boxplots exhibit reasonably high discriminatory capacity as the range overlaps somewhat.

However, for the svm, linear regression, and neural net models, there are numerous outliers in their respective boxplots, suggesting that specific instances in the test sample may be misclassified.

Question 4. Performance comparison on the test sample.

1. Use a decision threshold equal to the median (ex. Median(p)) of the predicted probabilities. Create the contingency table / confusion matrix of each of the predictions for this threshold.

```
# Logistic Regression
cl = as.numeric(pl>=median(pl))
xl = xtabs(~test$SupplyDisruption+cl)
xl
```

```
##                       cl
## test$SupplyDisruption   0   1
##                        0 500 363
##                        1   0 137
```

```
# Random Forest
cr = as.numeric(pr>=median(pr))
xr = xtabs(~test$SupplyDisruption+cr)
xr
```

```
##                         cr
## test$SupplyDisruption   0   1
##                       0 499 364
##                       1   0 137
```

```
# Support vector machine
cs = as.numeric(ps>=median(ps))
xs = xtabs(~test$SupplyDisruption+cs)
xs
```

```
##                         cs
## test$SupplyDisruption   0   1
##                       0 500 363
##                       1   0 137
```

```
# neural network
cn = as.numeric(pn>=median(pn))
xn = xtabs(~testNN$SupplyDisruption+cn)
xn
```

```
##                           cn
## testNN$SupplyDisruption   0   1
##                         0 500 363
##                         1   0 137
```

2. Compute the sensitivity, specificity, and accuracy measures for each model. Comment on what you observe and compare the models.

```
# Logistic Regression
xt=xl
Sens = xt[2,2]/(xt[2,2]+xt[1,2])
Spec = xt[1,1]/(xt[1,1]+xt[2,1])
Acc = (xt[2,2]+xt[1,1])/(xt[2,2]+xt[1,1]+xt[2,1]+xt[1,2])
print(c(Sens, Spec, Acc))
```

```
## [1] 0.274 1.000 0.637
```

```
# Random Forest
xt=xr
Sens = xt[2,2]/(xt[2,2]+xt[1,2])
Spec = xt[1,1]/(xt[1,1]+xt[2,1])
Acc = (xt[2,2]+xt[1,1])/(xt[2,2]+xt[1,1]+xt[2,1]+xt[1,2])
print(c(Sens, Spec, Acc))
```

```
## [1] 0.2734531 1.0000000 0.6360000
```

```
# Support vector machine
xt=xs
Sens = xt[2,2]/(xt[2,2]+xt[1,2])
Spec = xt[1,1]/(xt[1,1]+xt[2,1])
Acc = (xt[2,2]+xt[1,1])/(xt[2,2]+xt[1,1]+xt[2,1]+xt[1,2])
print(c(Sens, Spec, Acc))
```

```
## [1] 0.274 1.000 0.637
```

```
# neural network
xt=xn
Sens = xt[2,2]/(xt[2,2]+xt[1,2])
Spec = xt[1,1]/(xt[1,1]+xt[2,1])
Acc = (xt[2,2]+xt[1,1])/(xt[2,2]+xt[1,1]+xt[2,1]+xt[1,2])
print(c(Sens, Spec, Acc))
```

```
## [1] 0.274 1.000 0.637
```

Observations: From the statistics presented, we can infer that the four models perform similarly well with very close sensitivity, specificity and accuracy.

However, the sensitivity values for all four results are relatively low, suggesting that the models struggle to accurately identify positive instances. To address this issue, we may need to adjust the classification threshold to increase sensitivity, which could lead to a reduction in specificity. This adjustment can help the model to better capture actual positive instances and reduce the number of false negatives.

3. Plot the ROC curves for each model's prediction on the test data. Compare the AUC-ROC of each model. Comment on the model comparisons and which model is best predictor.

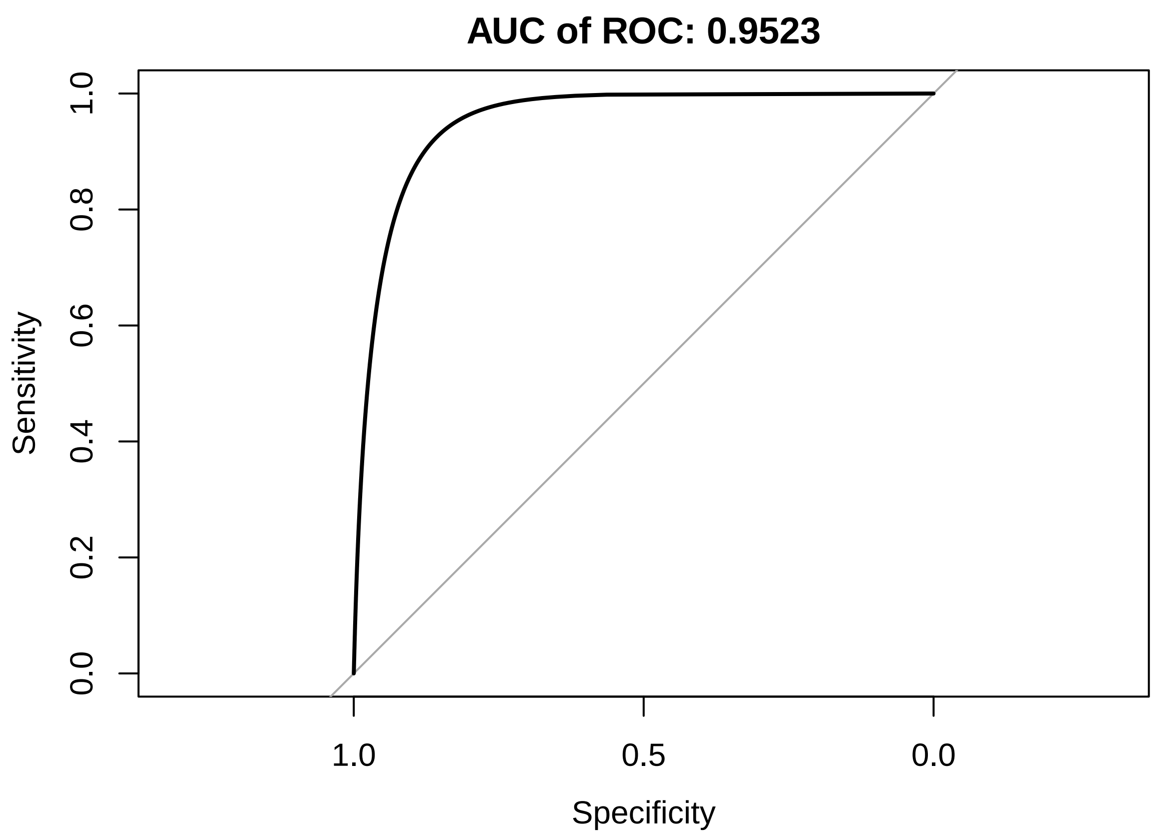
```
library(pROC)
```

```
## Warning: package 'pROC' was built under R version 4.2.3

## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
# Logistic Regression
rl = roc(test$SupplyDisruption, pl, smooth = TRUE)
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

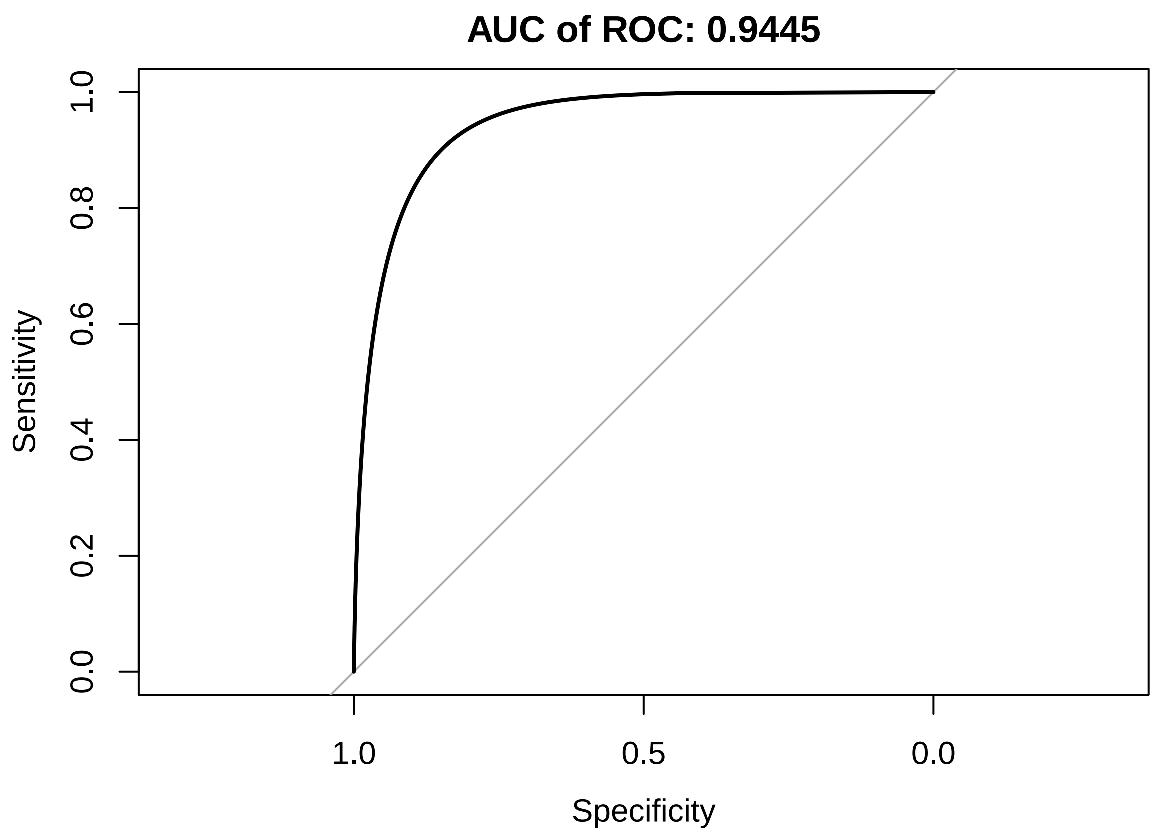
```
al = auc(rl)
plot(rl, main = paste("AUC of ROC:",round(al,4)))
```



```
# Random Forest
rr = roc(test$SupplyDisruption, pr,smooth = TRUE)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

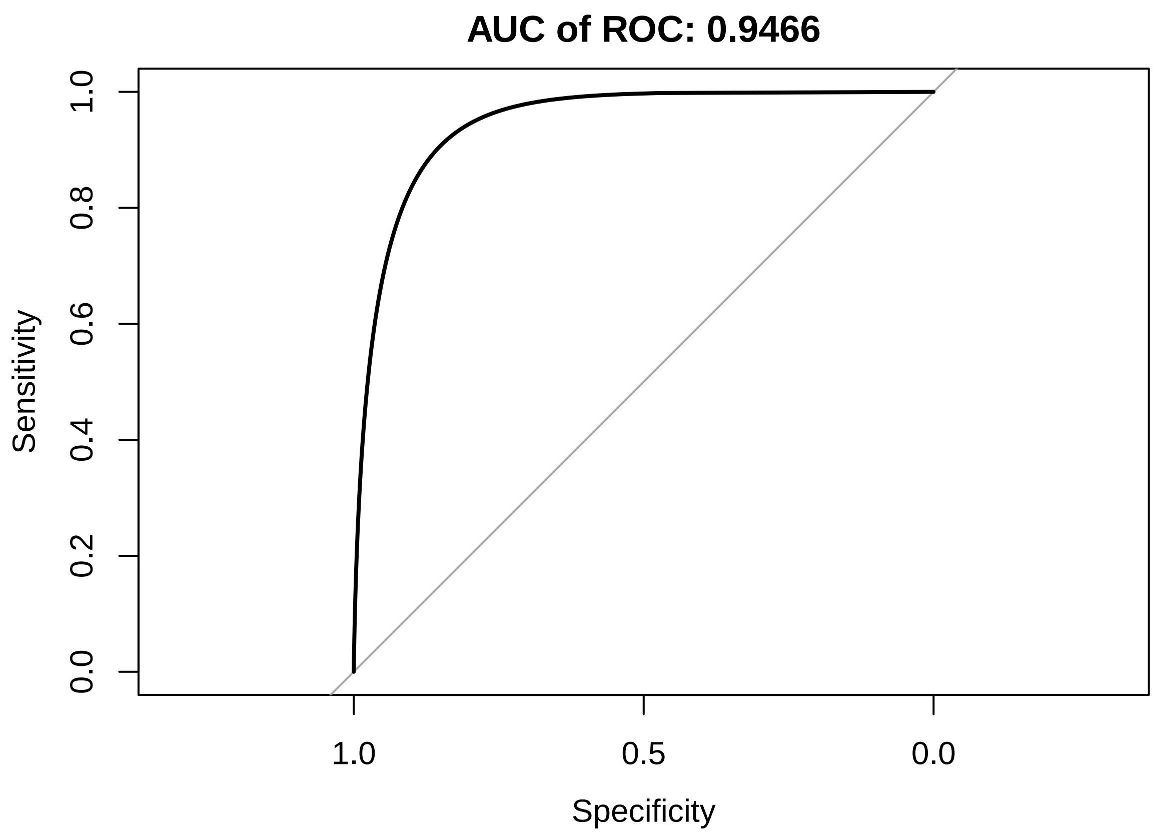
```
ar = auc(rr)
plot(rr, main = paste("AUC of ROC:",round(ar,4)))
```



```r
# Support vector machine
rs = roc(test$SupplyDisruption, ps, smooth = TRUE)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
as = auc(rs)
plot(rs, main = paste("AUC of ROC:",round(as,4)))
```



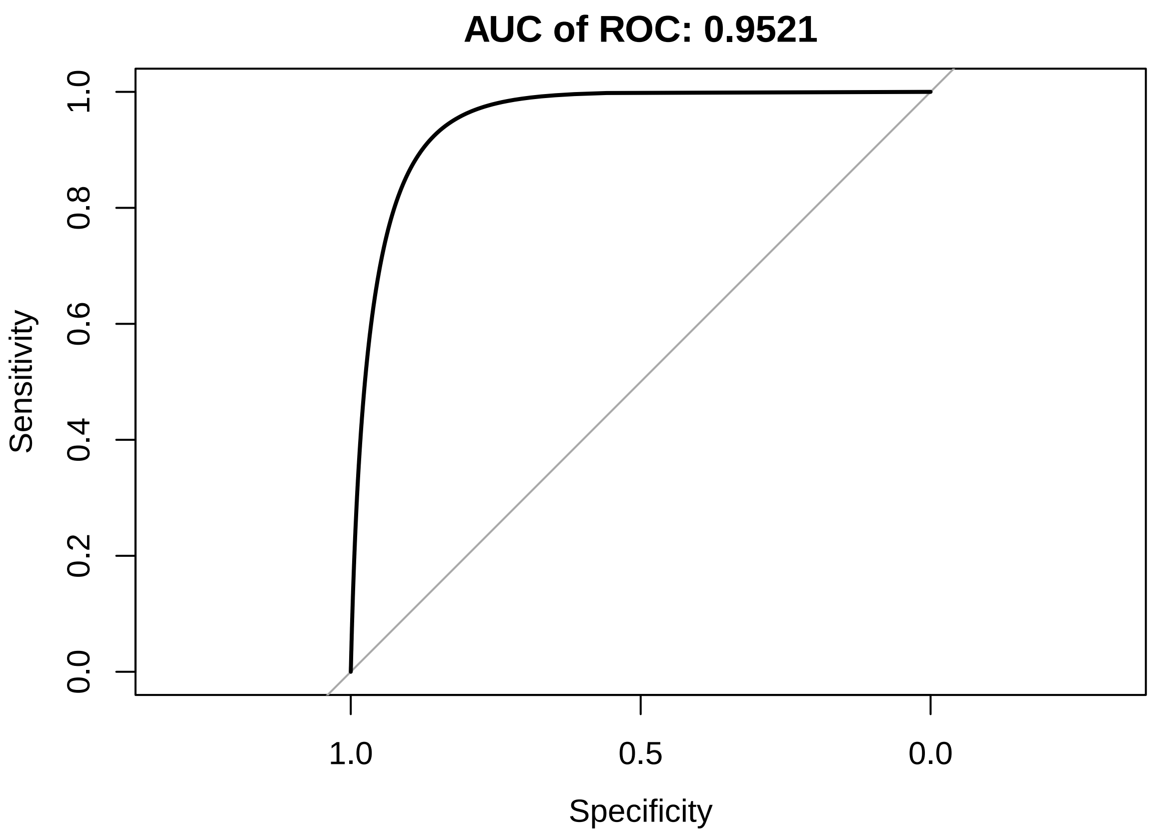
```r
# neural network
rn = roc(testNN$SupplyDisruption, pn, smooth = TRUE)
```

```
## Setting levels: control = 0, case = 1
```

```
## Warning in roc.default(testNN$SupplyDisruption, pn, smooth = TRUE): Deprecated
## use a matrix as predictor. Unexpected results may be produced, please pass a
## numeric vector.
```

```
## Setting direction: controls < cases
```

```r
an = auc(rn)
plot(rn, main = paste("AUC of ROC:",round(an,4)))
```



Observations: The AUC and ROC graphs demonstrate the trade-offs between sensitivity and specificity, and a better model would have an AUC-ROC score closer 1.0, indicating better separation between positive and negative cases.

Upon examining the graphs, we can conclude that the logistic model is the best performing model, with an AUC-ROC score of 0.9523, followed by neural network with score of 0.9521.

Question 5. Optimal Decision Threshold

1. Use only the prediction for the logistic model and random forest.

pl - Logistic pr - Random Forest

2. Now compute the optimal decision threshold using the test predictions for the two models and a cost of 0.9 and 1 for false positive and false negative errors respectively.

```
# Logistic
xt = xl
sens = xt[2,2]/(xt[2,2]+xt[1,2])
spec = xt[1,1]/(xt[1,1]+xt[2,1])
TCl = 0.9*(1-spec)+1*(1-sens)
TCl
```

```
## [1] 0.726
```

```
# Random Forest
xt = xr
sens = xt[2,2]/(xt[2,2]+xt[1,2])
spec = xt[1,1]/(xt[1,1]+xt[2,1])
TCr = 0.9*(1-spec)+1*(1-sens)
TCr
```

```
## [1] 0.7265469
```

```
# Using optimal threshold
# Logistic Regression
clo = as.numeric(pl>=TCl)
xlo = xtabs(~test$SupplyDisruption+clo)
xlo
```

```
##                        clo
## test$SupplyDisruption   0   1
##                       0 851  12
##                       1  77  60
```

```
# Random Forest
cro = as.numeric(pr>=TCr)
xro = xtabs(~test$SupplyDisruption+cro)
xro
```

```
##                        cro
## test$SupplyDisruption   0   1
##                       0 863   0
##                       1 112  25
```

```
# Logistic Regression
xt=xlo
Sens = xt[2,2]/(xt[2,2]+xt[1,2])
Spec = xt[1,1]/(xt[1,1]+xt[2,1])
Acc = (xt[2,2]+xt[1,1])/(xt[2,2]+xt[1,1]+xt[2,1]+xt[1,2])
print(c(Sens, Spec, Acc))
```

```
## [1] 0.8333333 0.9170259 0.9110000
```

```
# Random Forest
xt=xro
Sens = xt[2,2]/(xt[2,2]+xt[1,2])
Spec = xt[1,1]/(xt[1,1]+xt[2,1])
Acc = (xt[2,2]+xt[1,1])/(xt[2,2]+xt[1,1]+xt[2,1]+xt[1,2])
print(c(Sens, Spec, Acc))
```

```
## [1] 1.0000000 0.8851282 0.8880000
```

3. Comment on what you learned from this exercise.

Observations: In this stage, we raised the classification threshold by assigning a greater cost to false positives (type 1 error) to improve sensitivity. We obtained a threshold of 0.726 for logistic regression and 0.7265469 for random forest.

After tuning the models using the optimal threshold, we observed an improvement in overall sensitivity, specificity, and accuracy.

The logistic model has slightly lower sensitivity but higher specificity compared to the random forest model. Both models have relatively high accuracy, but the random forest model has a higher sensitivity score. The choice between the two models will depend on the specific needs of the application and the trade-off between the cost of false positives and false negatives.