Assignment 3

Yihan Feng

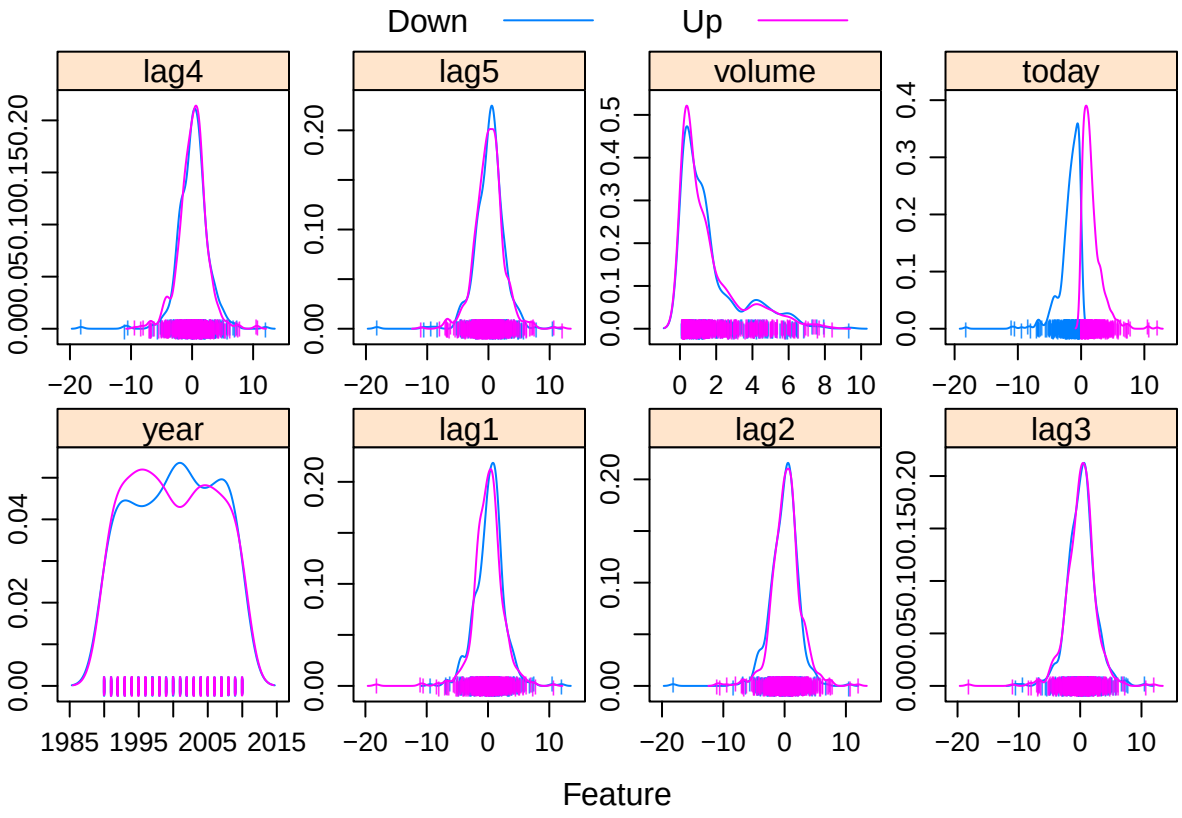
2021/3/22

This questions will be answered using the Weekly data set, which is part of the ISLR package. This data is similar in nature to the Smarket data on the textbook (ISL, Chapter4.6) except that it contains 1,089 weekly returns for 21 years, from the beginning of 1990 to the end of 2010. A description of the data can be found by typing? Weekly in the Console. (Note that the column Today is not a predictor.)

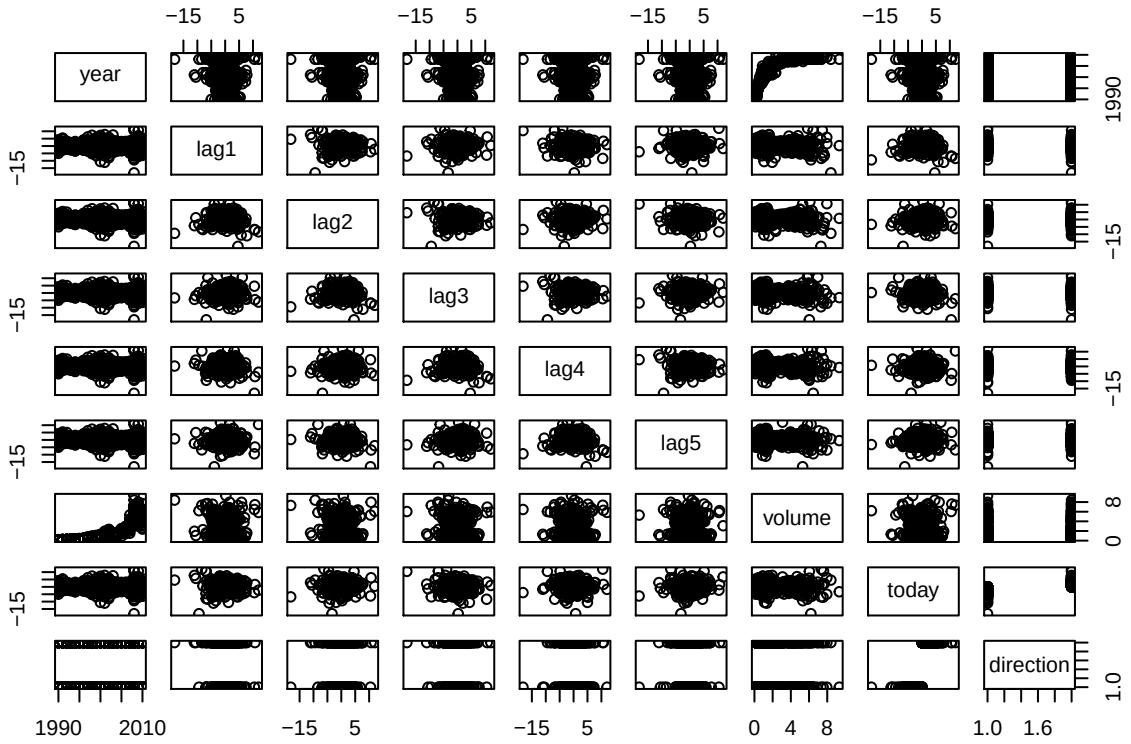
```
week.df = Weekly %>%
  janitor::clean_names()
```

```
featurePlot(x = week.df[, 1:8],
            y = week.df$direction,
            scales = list(x = list(relation = "free"),
                          y = list(relation = "free")),
            plot = "density", pch = "|",
            auto.key = list(columns = 2))
```

(a) Produce some graphical summaries of the Weekly data.




```
pairs(week.df)
```




```
x.train = week.df$year <= 2008
x.test = week.df[!x.train,]

glm.fit = glm(direction ~ lag1 + lag2 + lag3 + lag4 + lag5 + volume,
              data = week.df,
              subset = x.train,
              family = binomial(link = "logit"))
summary(glm.fit)
```

(b) Use the data from 1990 to 2008 as the training data and the held-out data as the test data. Perform a logistic regression with Direction as the response and the five Lag variables plus Volume as predictors. Do any of the predictors appear to be statistically significant? If so, which ones? Compute the confusion matrix and overall fraction of correct predictions using the test data. Briefly explain what the confusion matrix is telling you.

```
##
## Call:
## glm(formula = direction ~ lag1 + lag2 + lag3 + lag4 + lag5 +
##     volume, family = binomial(link = "logit"), data = week.df,
##     subset = x.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
```

```
## -1.7186  -1.2498   0.9823   1.0841   1.4911
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.33258    0.09421   3.530 0.000415 ***
## lag1        -0.06231    0.02935  -2.123 0.033762 *
## lag2         0.04468    0.02982   1.499 0.134002
## lag3        -0.01546    0.02948  -0.524 0.599933
## lag4        -0.03111    0.02924  -1.064 0.287241
## lag5        -0.03775    0.02924  -1.291 0.196774
## volume      -0.08972    0.05410  -1.658 0.097240 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1354.7  on 984  degrees of freedom
## Residual deviance: 1342.3  on 978  degrees of freedom
## AIC: 1356.3
##
## Number of Fisher Scoring iterations: 4
```

```
test.pred.prob = predict(glm.fit, newdata = week.df[-x.train,],
                         type = "response")
test.pred = rep("Down", length(test.pred.prob))
test.pred[test.pred.prob > 0.5] = "Up"

confusionMatrix(data = as.factor(test.pred),
                reference = week.df$direction[-x.train],
                positive = "Up")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Down  Up
##       Down  111 114
##       Up    372 491
##
##                Accuracy : 0.5533
##                  95% CI : (0.5232, 0.5831)
##     No Information Rate : 0.5561
##     P-Value [Acc > NIR] : 0.585
##
##                   Kappa : 0.0437
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.8116
##             Specificity : 0.2298
##          Pos Pred Value : 0.5689
##          Neg Pred Value : 0.4933
##              Prevalence : 0.5561
##          Detection Rate : 0.4513
##    Detection Prevalence : 0.7932
```

```
##       Balanced Accuracy : 0.5207
##
##        'Positive' Class : Up
##
```

statistically significant predictors:
lag1 is the statistically significant predictor, as its p value less that 0.05.

confusion matrix:
- accuracy: 0.5533: *The classifier is 55.33% correct.*
- Kappa: 0.0437: *The agreement between classification and truth values is about 4.37%.*
- Sensitivity: 0.8116: *The probability of correctly identified positive case is 81.16%.*
- Specificity: 0.2298: *The probability of correctly identified negative case is 22.98%.*

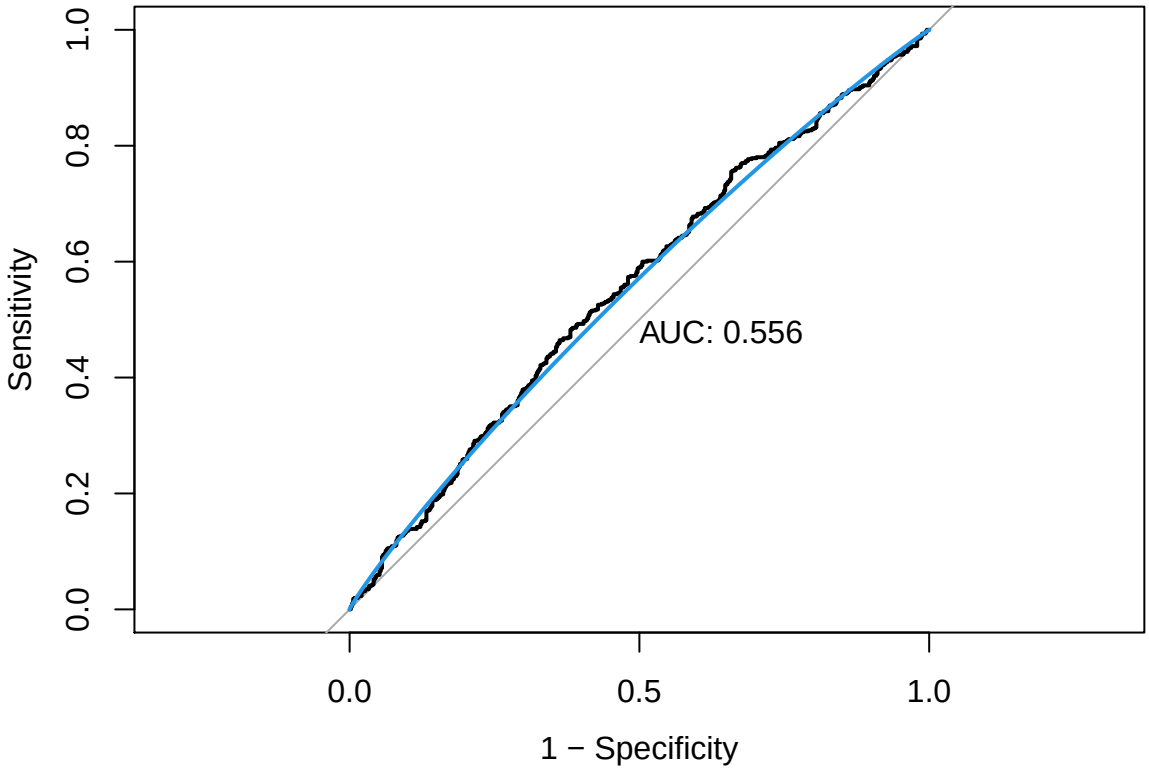
```
glm.fit1 = glm(direction ~ lag1 + lag2,
               data = week.df,
               subset = x.train,
               family = binomial(link = "logit"))

test.pred.prob = predict(glm.fit1, newdata = week.df[-x.train,],
                         type = "response")

roc.glm = roc(week.df$direction[-x.train], test.pred.prob)
```

(c) Now fit the logistic regression model using the training data period from 1990 to 2008, with Lag1 and Lag2 as the predictors. Plot the ROC curve using the test data and report the AUC.

```
## Setting levels: control = Down, case = Up
```

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

```
plot(roc.glm, legacy.axes = TRUE, print.auc = TRUE)
plot(smooth(roc.glm), col = 4, add = TRUE)
```



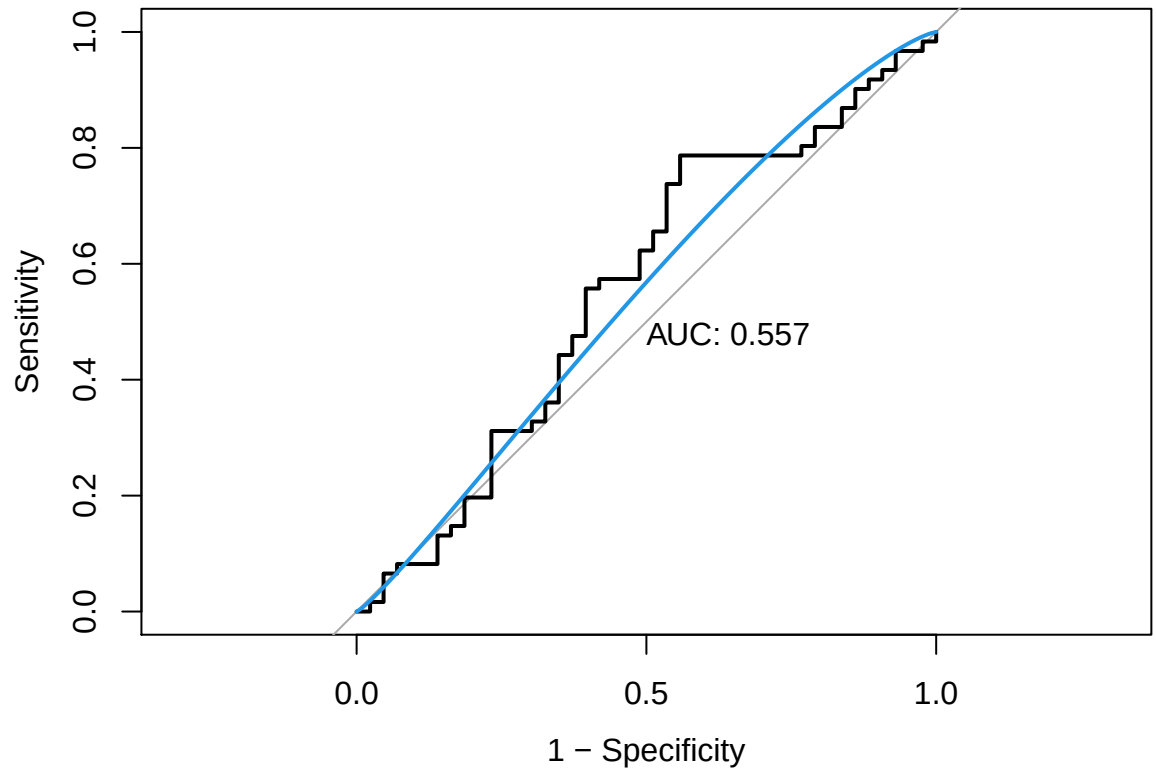
The AUC is 0.556.

```
# LDA
lda.fit = lda(direction ~ lag1 + lag2,
              data = week.df,
              subset = x.train)
lda.pred = predict(lda.fit,
                   newdata = x.test)

roc.lda = roc(x.test$direction, lda.pred$posterior[,2],
              levels = c("Down", "Up"))
```

(d) Repeat (c) using LDA and QDA.

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

```
plot(roc.lda, legacy.axes = TRUE, print.auc = TRUE)
plot(smooth(roc.lda), col = 4, add = TRUE)
```

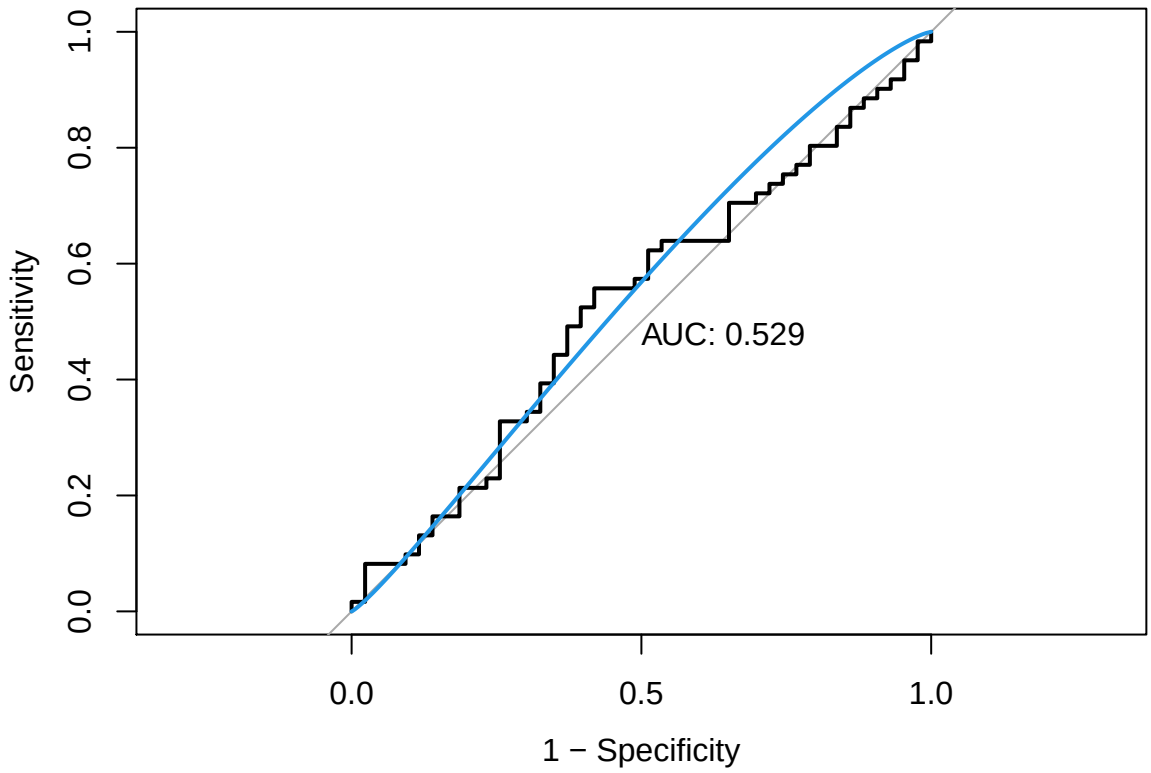



```
# QDA
qda.fit = qda(direction ~ lag1 + lag2,
              data = week.df,
              subset = x.train)
qda.pred = predict(qda.fit,
                   newdata = x.test)
```

```
roc.qda = roc(x.test$direction, qda.pred$posterior[,2],
              levels = c("Down", "Up"))
```

```
## Setting direction: controls > cases
```

```
plot(roc.qda, legacy.axes = TRUE, print.auc = TRUE)
plot(smooth(roc.lda), col = 4, add = TRUE)
```



The AUC of LDA is 0.557. The AUC of QDA is 0.529

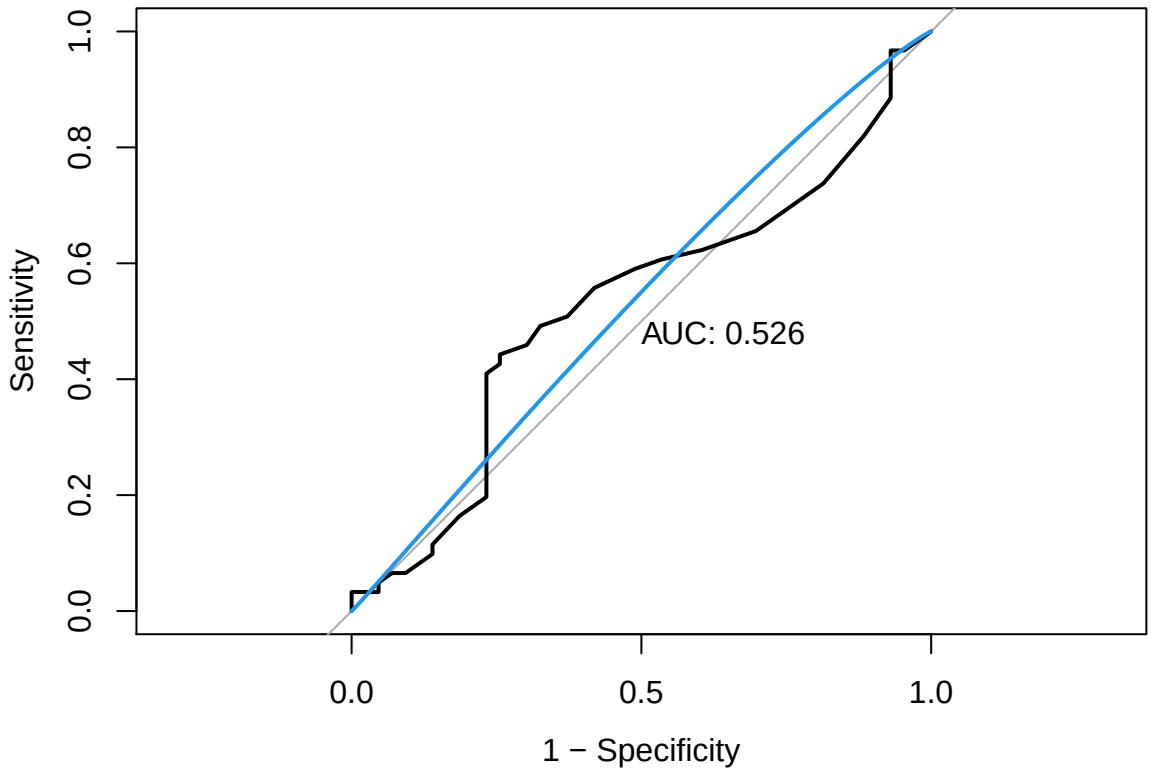
```
set.seed = (2)
control = trainControl(method = "repeatedcv",
                       repeats = 5,
                       summaryFunction = twoClassSummary,
                       classProbs = T)
knn.fit = train(x = week.df[x.train, 2:3],
                y = week.df$direction[x.train],
                method = "knn",
                metric = "ROC",
                preProcess = c("center", "scale"),
                tuneGrid = data.frame(k = seq(1, 200, by = 10)),
```

```
                trControl = control)
knn.pred = predict(knn.fit,
                   newdata = x.test,
                   type = "prob")
roc.knn = roc(x.test$direction, knn.pred$Up, levels = c("Down", "Up"))
```

(e) Repeat (c) using KNN. Briefly discuss your results in (c)-(e).

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

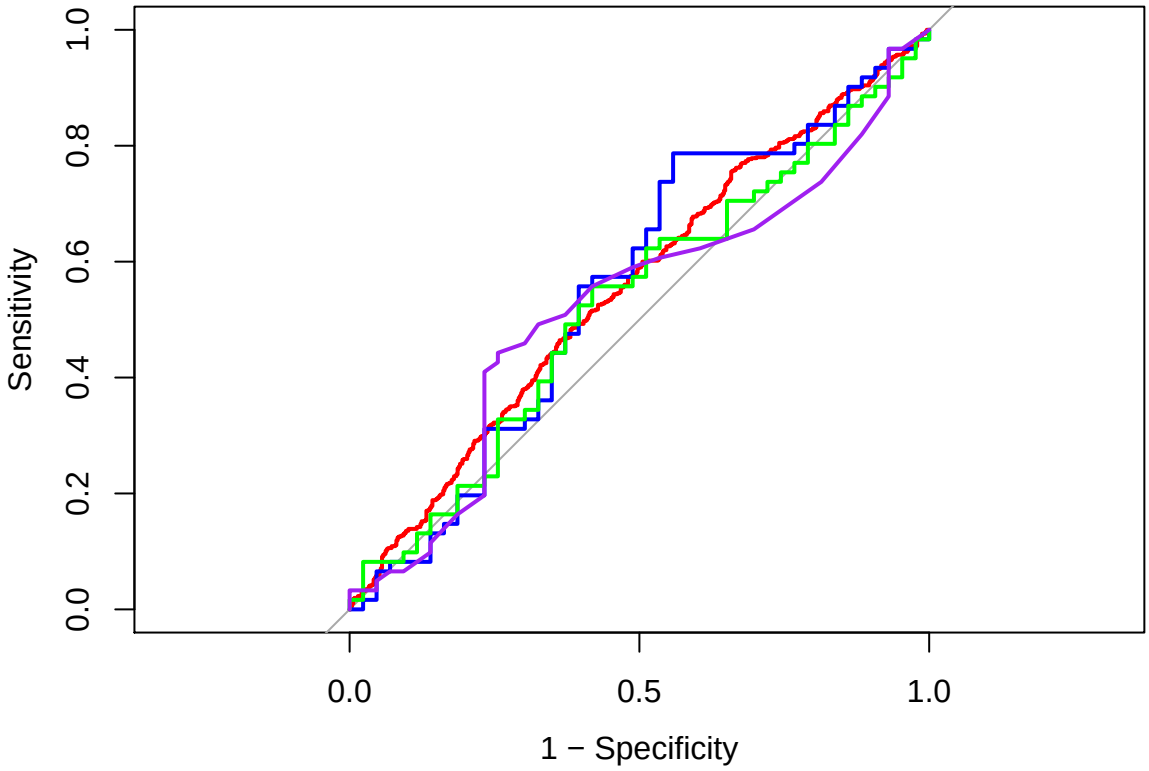
```
plot(roc.knn, legacy.axes = TRUE, print.auc = TRUE)
plot(smooth(roc.knn), col = 4, add = TRUE)
```



The AUC of KNN is 0.535.

Compare results in (c) - (e):

```
plot(roc.glm, col = "red", legacy.axes = TRUE)
plot(roc.lda, col = "blue", add = TRUE)
plot(roc.qda, col = "green", add = TRUE)
plot(roc.knn, col = "purple", add = TRUE)
```



Among the 4 methods, all of them have about 0.5 AUC, which means that they have about 50% chance that the model will be able to distinguish between the Up and Down classes.

I observed that LDA method has the largest AUC (0.557), and the QDA method has the smallest AUC (0.529). It indicates that LDA has the highest chance that the model will be able to distinguish between the Up and Down classes.