

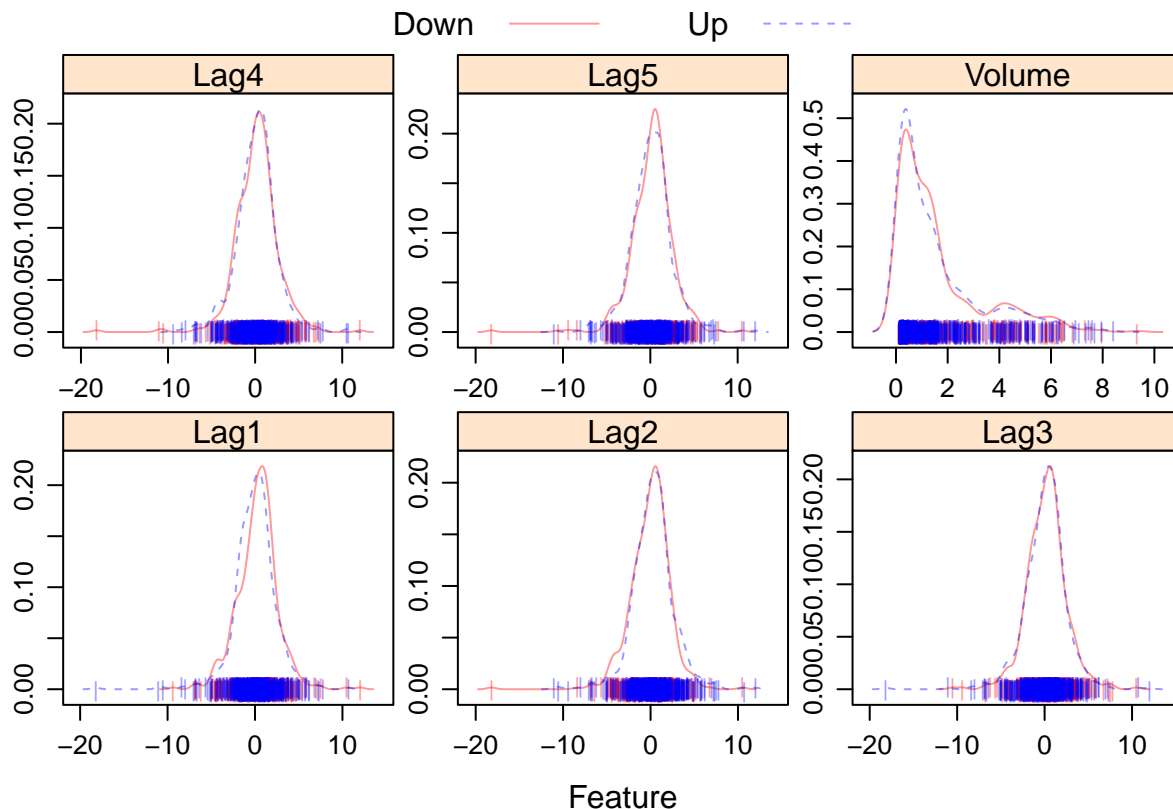
DS2_HW3

Siyan Chen

4/6/2019

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

```
transparentTheme(trans = .4)
featurePlot(x = df[, 2:7],
            y = df$Direction,
            scales = list(x=list(relation = "free"),
                          y=list(relation = "free")),
            plot = "density", pch = "|",
            auto.key = list(columns = 2))
```



(b) Use the full data set to 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?

```
glm_fit = glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
              data = df,
              family = binomial)
contrasts(df$Direction)
```

```
##      Up
```

```
## Down 0
## Up 1
summary(glm_fit)

##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##      Volume, family = binomial, data = df)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -1.6949  -1.2565   0.9913   1.0849   1.4579
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.26686    0.08593   3.106  0.0019 **
## Lag1        -0.04127    0.02641  -1.563  0.1181
## Lag2         0.05844    0.02686   2.175  0.0296 *
## Lag3        -0.01606    0.02666  -0.602  0.5469
## Lag4        -0.02779    0.02646  -1.050  0.2937
## Lag5        -0.01447    0.02638  -0.549  0.5833
## Volume      -0.02274    0.03690  -0.616  0.5377
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##      Null deviance: 1496.2  on 1088  degrees of freedom
## Residual deviance: 1486.4  on 1082  degrees of freedom
## AIC: 1500.4
##
## Number of Fisher Scoring iterations: 4
```

Yes, predictors Lag2 appear to be statistically significant.

(c) Compute the confusion matrix and overall fraction of correct predictions. Briefly explain what the confusion matrix is telling you.

```
# Bayes classigier(cutoff 0.5)
pred.prob = predict(glm_fit, type = "response")
pred = rep("Down", length(pred.prob))
pred[pred.prob > 0.5] = "Up"

# confusionMatrix
confusionMatrix(data = as.factor(pred),
                 reference = as.factor(df$Direction),
                 positive = "Up")
```

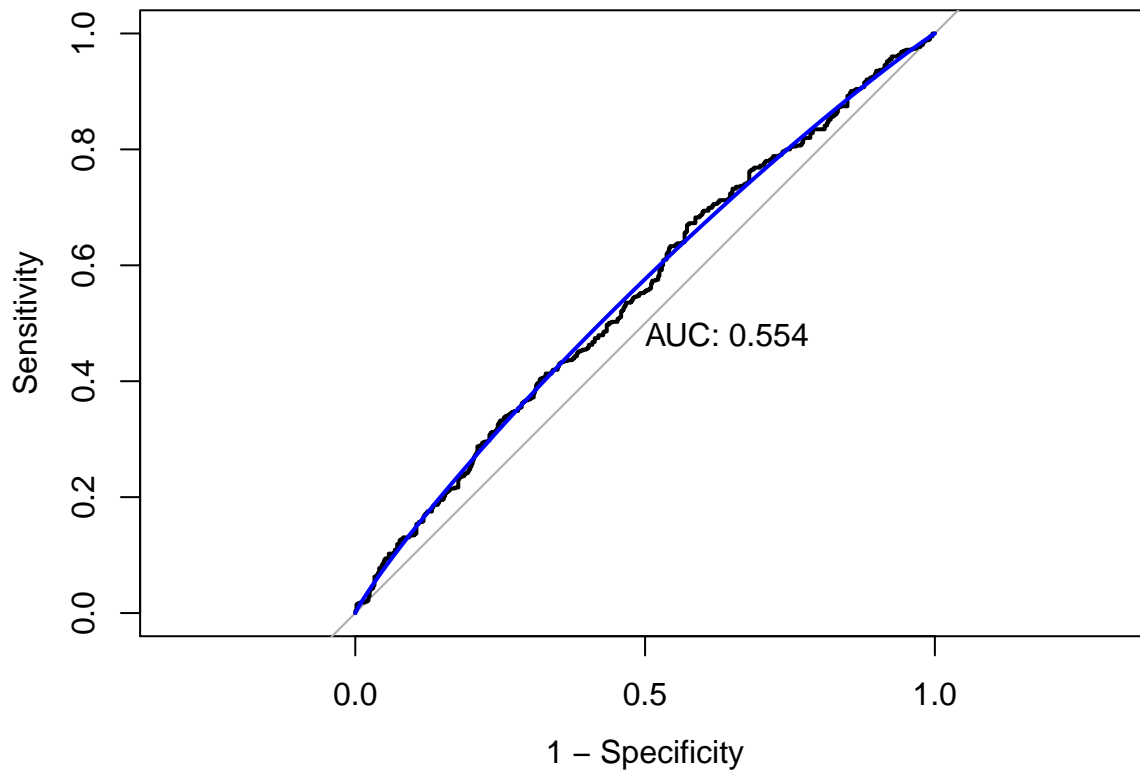
```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction Down  Up
##      Down    54  48
```

```
##      Up      430 557
##
##              Accuracy : 0.5611
##              95% CI : (0.531, 0.5908)
##      No Information Rate : 0.5556
##      P-Value [Acc > NIR] : 0.369
##
##              Kappa : 0.035
##      McNemar's Test P-Value : <2e-16
##
##              Sensitivity : 0.9207
##              Specificity : 0.1116
##              Pos Pred Value : 0.5643
##              Neg Pred Value : 0.5294
##              Prevalence : 0.5556
##              Detection Rate : 0.5115
##      Detection Prevalence : 0.9063
##      Balanced Accuracy : 0.5161
##
##      'Positive' Class : Up
##
```

From the confusion matrix, sensitivity is 0.9207 (the probability of true “Up” is 0.9207 when the direction is predicted to be Up) and the specificity is 0.1116 (the probability of not “Up” is 0.1322 when the direction is predicted to be down) Accuracy tells the overall probability of correct classifier is 0.5611. Kappa is 0.035, which measures the interrater agreement for categorical items.

(d) Plot the ROC curve using the predicted probability from logistic regression and report the AUC.

```
roc_glm = roc(df$Direction, pred.prob)
plot(roc_glm, legacy.axes = TRUE, print.auc = TRUE)
plot(smooth(roc_glm), col = 4, add = TRUE)
```



Based on the plot, the AUC is 0.554, which suggests that the capability of logistic models distinguishing between classes is not good.

(e) Now fit the logistic regression model using a training data period from 1990 to 2008, with Lag1 and Lag2 as the predictors. Plot the ROC curve using the held out data (that is, the data from 2009 and 2010) and report the AUC.

```
train_subset = df %>%
  filter(1990<=Year & Year<=2008)
test_subset = anti_join(df, train_subset)

## Joining, by = c("Year", "Lag1", "Lag2", "Lag3", "Lag4", "Lag5", "Volume", "Today", "Direction")
rowtrain = train_subset$Direction
rowtest = test_subset$Direction

glm_fit1 = glm(Direction~ Lag1 + Lag2,
  data = train_subset,
  family = binomial)
summary(glm_fit1)

##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2, family = binomial, data = train_subset)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -1.6149  -1.2565   0.9989   1.0875   1.5330
##
```

```

## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.21109    0.06456   3.269  0.00108 **
## Lag1        -0.05421    0.02886  -1.878  0.06034 .
## Lag2         0.05384    0.02905   1.854  0.06379 .
## ---
## 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: 1347.0  on 982  degrees of freedom
## AIC: 1353
##
## Number of Fisher Scoring iterations: 4

```

```

contrasts(train_subset$Direction)

##      Up
## Down  0
## Up    1

```

```

pred.test.value = predict(glm_fit1,
                          newdata = test_subset,
                          type = "response")

# Bayes Method Cutoff
pred.test = rep("Down", length(pred.test.value))
pred.test[pred.test.value>0.5] = "Up"

# Confusion Matrix
confusionMatrix(data = as.factor(pred.test),
                 reference = as.factor(rowtest),
                 positive = "Up")

```

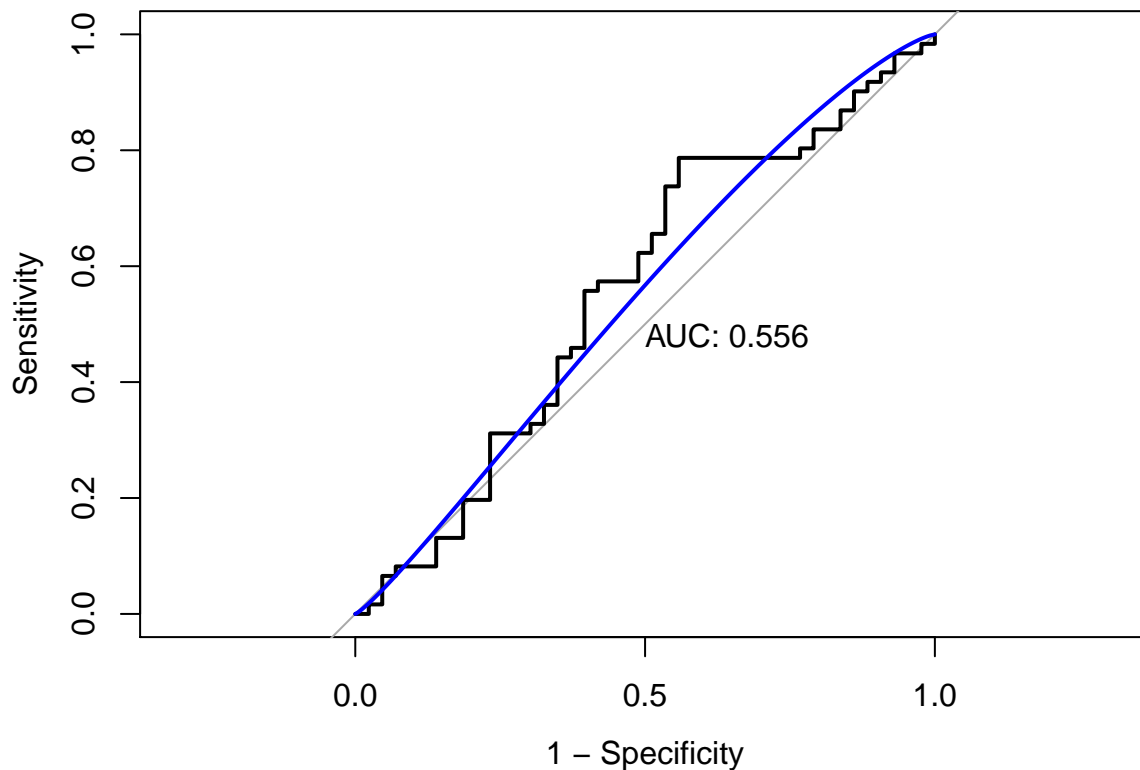
```

## Confusion Matrix and Statistics
##
##              Reference
## Prediction Down Up
##      Down    7  8
##      Up    36 53
##
##              Accuracy : 0.5769
##              95% CI : (0.4761, 0.6732)
##      No Information Rate : 0.5865
##      P-Value [Acc > NIR] : 0.6193
##
##              Kappa : 0.035
##  Mcnemar's Test P-Value : 4.693e-05
##
##              Sensitivity : 0.8689
##              Specificity : 0.1628
##      Pos Pred Value : 0.5955
##      Neg Pred Value : 0.4667
##              Prevalence : 0.5865
##      Detection Rate : 0.5096

```

```
## Detection Prevalence : 0.8558
## Balanced Accuracy : 0.5158
##
## 'Positive' Class : Up
##
```

```
# ROC
roc1 = roc(test_subset$Direction, pred.test.value)
plot(roc1, legacy.axes = TRUE, print.auc = TRUE)
plot(smooth(roc1), col = 4, add = TRUE)
```

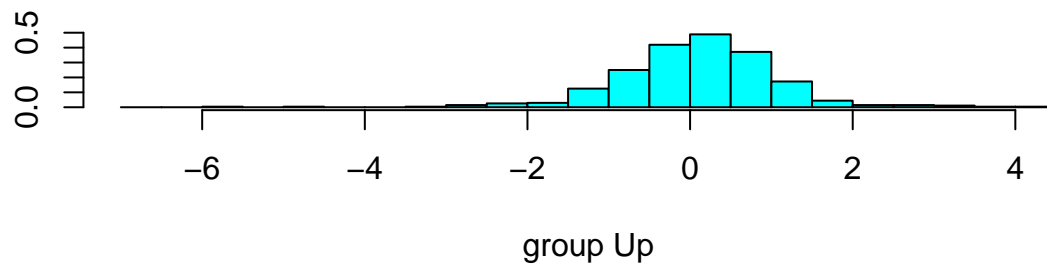
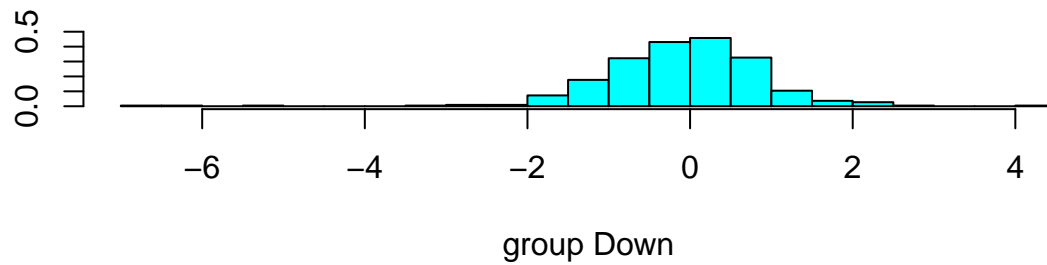


AUC is 0.556. The model does better job to classify compared to model in c.

(f) Repeat (e) using LDA and QDA.

```
#Discriminant analysis
## LDA
library(MASS)

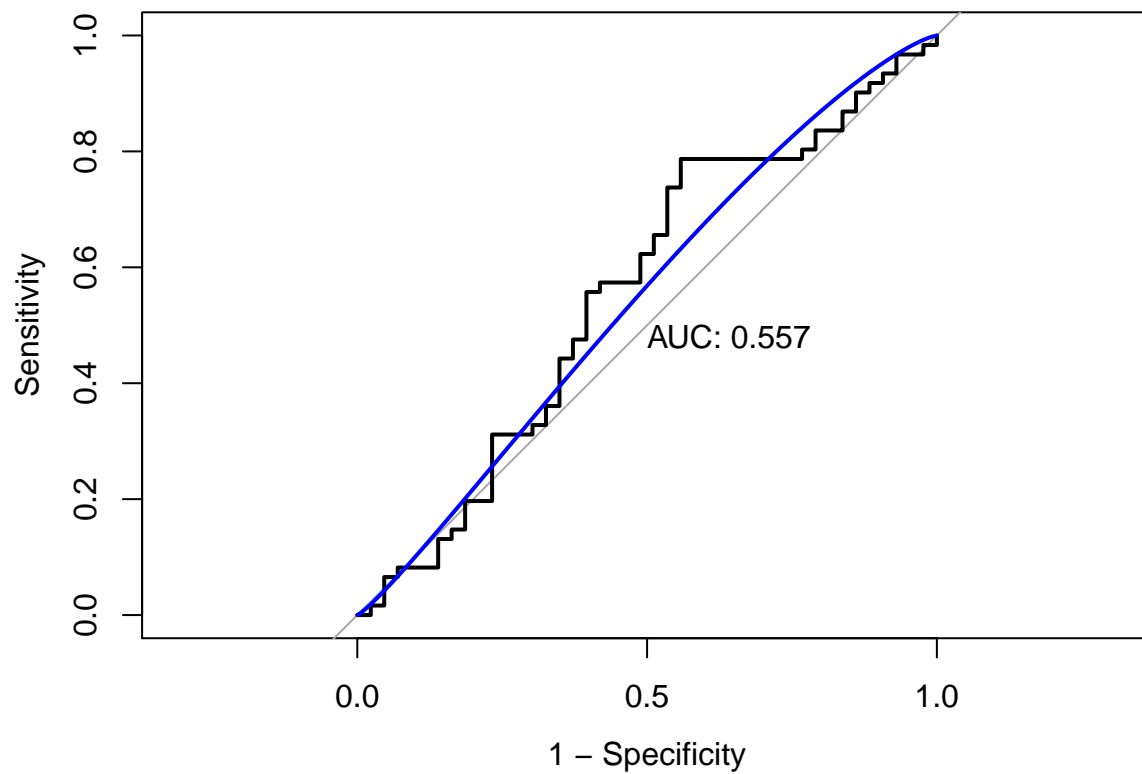
##
## Attaching package: 'MASS'
## The following object is masked from 'package:dplyr':
##
## select
lda.fit = lda(Direction ~ Lag1 + Lag2,
              data = train_subset)
plot(lda.fit)
```



```
# evaluate the test set performance using roc
lda.pred = predict(lda.fit, newdata = test_subset)
head(lda.pred$posterior)
```

```
##           Down           Up
## 1 0.5602039 0.4397961
## 2 0.3079163 0.6920837
## 3 0.4458032 0.5541968
## 4 0.4785107 0.5214893
## 5 0.4657943 0.5342057
## 6 0.5262907 0.4737093
```

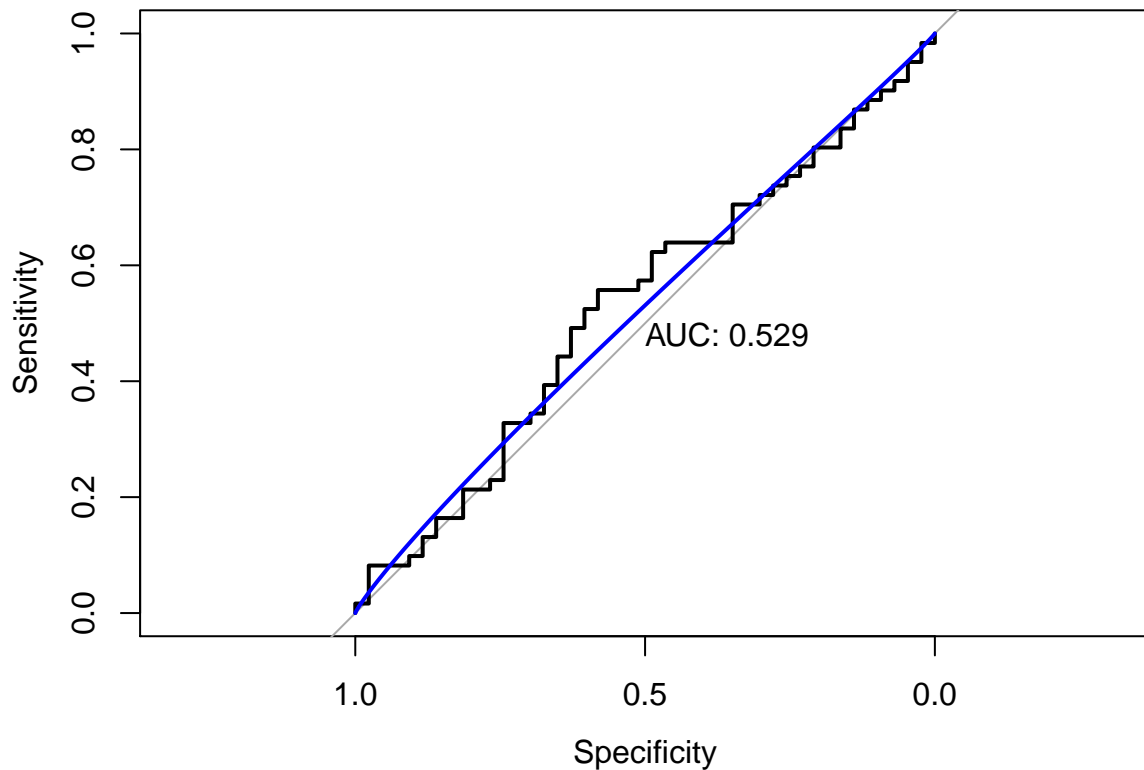
```
roc.lda = roc(test_subset$Direction, lda.pred$posterior[,2],
              levels = c("Down", "Up"))
plot(roc.lda, legacy.axes = TRUE, print.auc = TRUE)
plot(smooth(roc.lda), col = 4, add = TRUE)
```



```
## QDA
qda.fit = qda(Direction ~ Lag1 + Lag2,
              data = train_subset)
qda.pred = predict(qda.fit, newdata = test_subset)
head(qda.pred$posterior)
```

```
##      Down      Up
## 1 0.5436205 0.4563795
## 2 0.3528814 0.6471186
## 3 0.2227273 0.7772727
## 4 0.3483016 0.6516984
## 5 0.4598550 0.5401450
## 6 0.5119613 0.4880387
```

```
roc.qda = roc(test_subset$Direction, qda.pred$posterior[,2],
              levels = c("Down", "Up"))
plot(roc.qda, legacy.axis = TRUE, print.auc = TRUE)
plot(smooth(roc.qda), col = 4, add = TRUE)
```

LDA is 0.557 and AUC for QDA is 0.529

AUC for

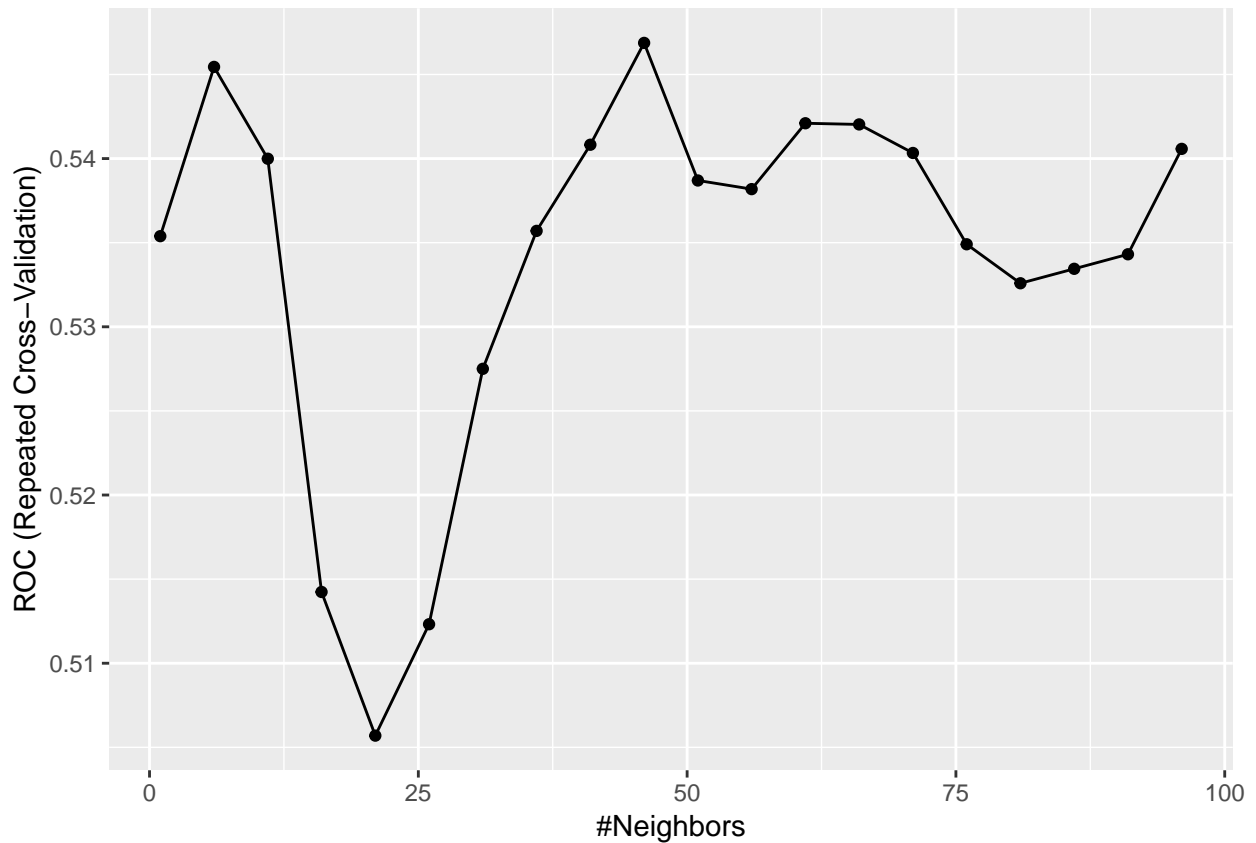
(g) Repeat (e) using KNN. Briefly discuss your results.

```
# neighbor for classification
# Using caret
ctrl <- trainControl(method = "repeatedcv", repeats = 5, summaryFunction = twoClassSummary, classProbs = TRUE)

model.knn = train(x = train_subset[, 2:3],
                  y = train_subset$Direction,
                  method = "knn",
                  preProcess = c("center", "scale"),
                  tuneGrid = data.frame(k = seq(1, 100, by = 5)),
                  trControl = ctrl)

## Warning in train.default(x = train_subset[, 2:3], y =
## train_subset$Direction, : The metric "Accuracy" was not in the result set.
## ROC will be used instead.

ggplot(model.knn)
```



```
# predict.train DO CENTRAL AND SCALE AUTOMATICALLY
# # how to central and scale here ?
# predic_knn = predict.train(model.knn, testX = test_subset[, 2:3],
#                             testY = test_subset$Direction)
```

According to the plot, the ROC is geatest when # neighbors is around 50 and the auc is not greater than 0.55. Compared the AUC, LDA model did better job to distinguish the two classes.