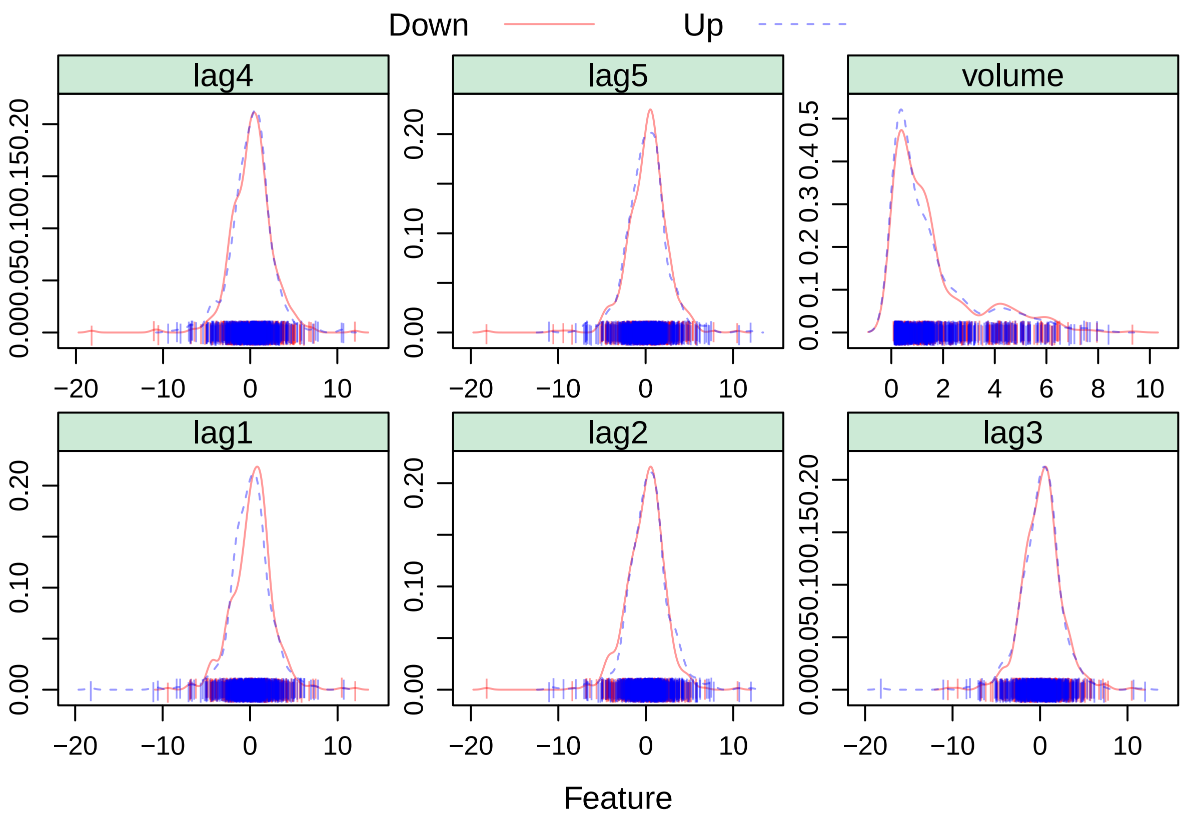
Homework 3

Ngoc Duong

4/14/2020

Import data from ISLR package

a) Produce some graphical summaries of the data



We can see the densities of the predictors do not differ much by the outcome, which indicate these predictors might not be useful for the classification task.

b) Perform logistic regression using full data

Fist, we divide the data up into a training and test set

```r
set.seed(7)
rowTrain <- createDataPartition(y = weekly$direction,
                                p = 3/4,
                                list = FALSE)
```

Next, we run a logistic regression on the training set using the five Lag variables and Volume as predictors, with Direction as the binary response.

```
glm.fit <- glm(direction~lag1+lag2+lag3+lag4+lag5+volume,
               data = weekly,
               subset = rowTrain,
               family = binomial)

broom::tidy(glm.fit) %>% knitr::kable()
```

term	estimate	std.error	statistic	p.value
(Intercept)	0.2176372	0.0989546	2.1993634	0.0278521
lag1	-0.0663648	0.0326086	-2.0351929	0.0418315
lag2	0.0539180	0.0315011	1.7116215	0.0869664
lag3	0.0123502	0.0319795	0.3861921	0.6993544
lag4	-0.0069759	0.0310504	-0.2246651	0.8222398
lag5	-0.0401298	0.0310541	-1.2922559	0.1962685
volume	0.0055021	0.0430427	0.1278291	0.8982842

```
#check the levels of the outcome variable
contrasts(weekly$direction) #1 represents upward trend, 0 downward trend
```

```
##      Up
## Down  0
## Up    1
```

The logistic regression result table shows that only the estimated effect of predictor lag1 is statistically significant at 5% significance level (p-value = 0.04).

c) Compute the confusion matrix

```
test.pred.prob <- predict(glm.fit, newdata = weekly[-rowTrain,],
                          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 = weekly$direction[-rowTrain],
                positive = "Up")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Down  Up
##       Down   15  27
##       Up    106 124
##
##                Accuracy : 0.511
##                  95% CI : (0.4499, 0.5719)
##     No Information Rate : 0.5551
##     P-Value [Acc > NIR] : 0.9361
##
##                   Kappa : -0.0586
```

```
##
##  Mcnemar's Test P-Value : 1.347e-11
##
##             Sensitivity : 0.8212
##             Specificity : 0.1240
##          Pos Pred Value : 0.5391
##          Neg Pred Value : 0.3571
##              Prevalence : 0.5551
##          Detection Rate : 0.4559
##    Detection Prevalence : 0.8456
##       Balanced Accuracy : 0.4726
##
##        'Positive' Class : Up
##
```

Briefly explain the confusion matrix:

Fraction of correct predictions is $(15 + 124)/272 = 0.511$ (also reported as "Accuracy")

The Kappa value of -0.0586 (less than 0) indicates the "agreement" between the predicted and observed outcomes are approximately random/worse than random.

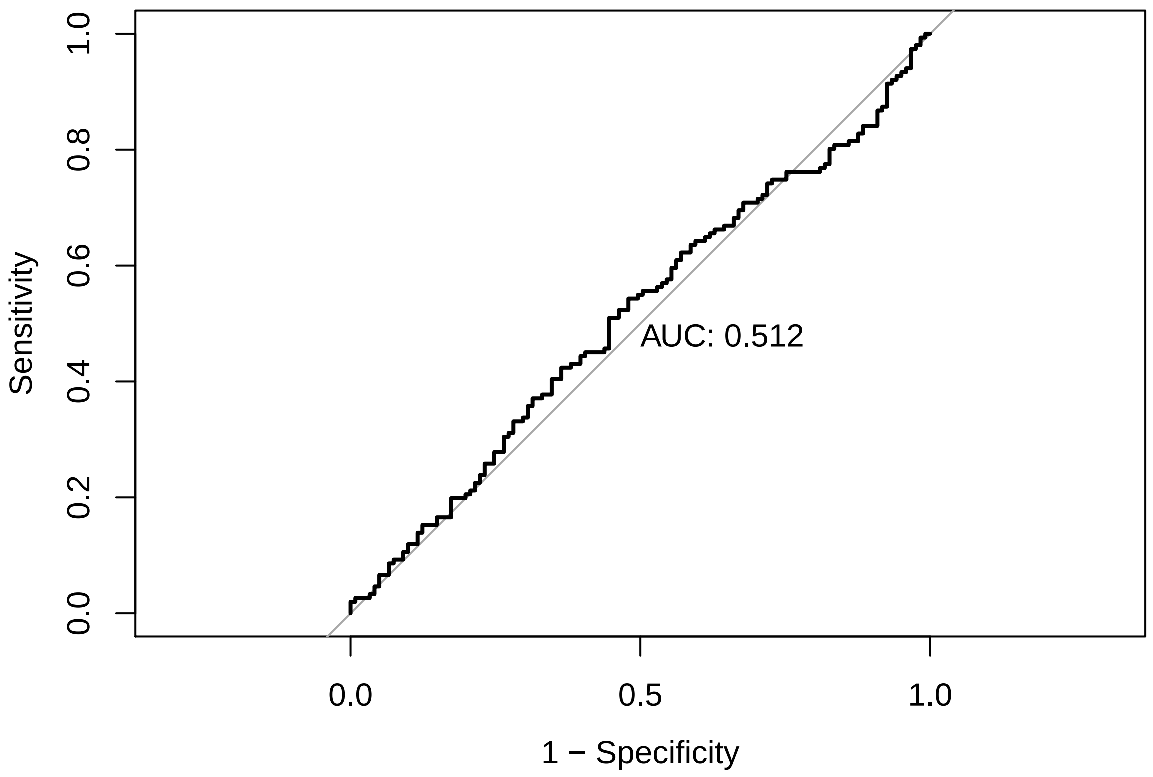
Sensitivity is 0.8212 means 82.12% of stocks with upward trend are correctly identified as "upward" by the model. Specificity is 0.124 means 12.4% of stocks with downward trend are correctly identified as "downward" by the model.

PPV is 0.5391, which means 53.91% of the stocks actually demonstrate upward trend given they are predicted (by the model) to have upward trend.

NPV is 0.3571, which means 35.71% of the stocks actually demonstrate downwardward trend given they are predicted (by the model) to have downward trend.

d) Plot the test ROC curve

```r
roc.glm <- roc(weekly$direction[-rowTrain], test.pred.prob)
plot(roc.glm, legacy.axes = TRUE, print.auc = TRUE)
```



The AUC is 0.512, which is roughly the same as 0.5. This indicates the model has low discriminatory ability (only marginally better than a random classifier).

e) Logistic regression with training data from 1990 to 2008

For this model, we use Lag1 and Lag2 as the predictors

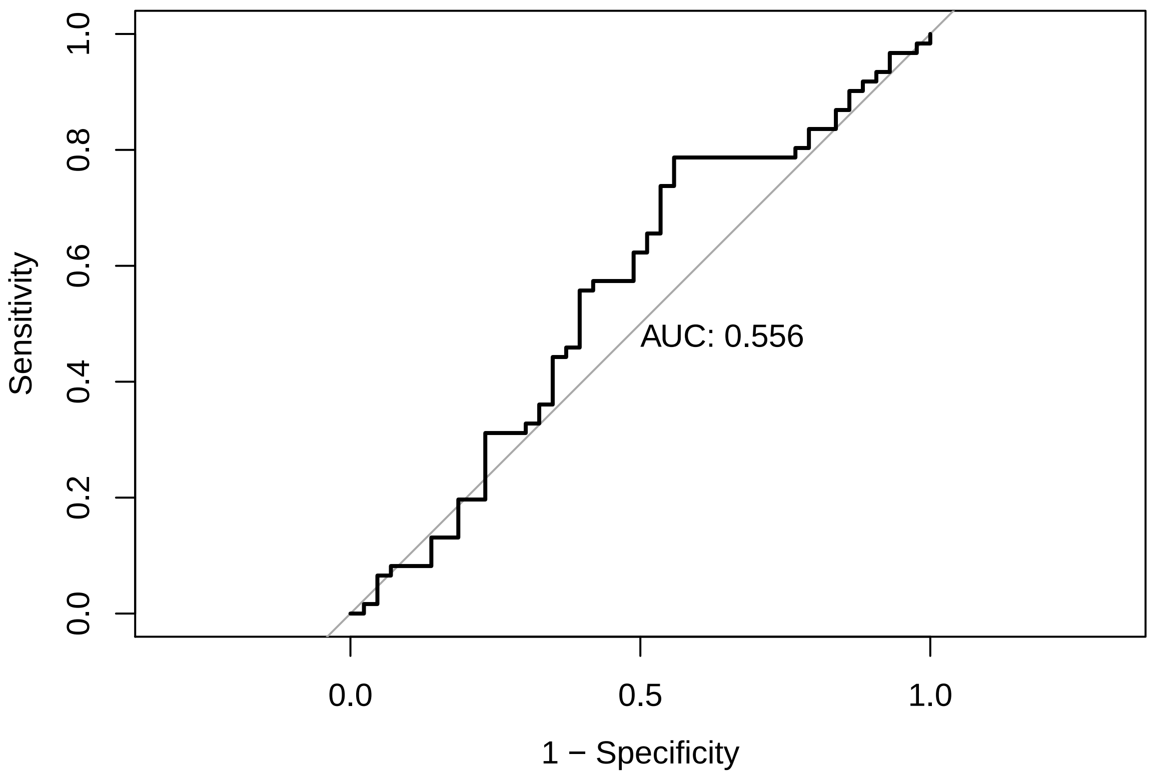
```
train_data = weekly %>% filter(year <= 2008)
test_data = weekly %>% filter(year > 2008)

glm.train <- glm(direction~lag1+lag2,
               data = train_data,
               family = binomial)
```

Plot the ROC curve using the held-out data (data from 2009-2010)

```
test.pred.prob.new <- predict(glm.train, newdata = test_data,
                              type = "response")
test.pred.new <- rep("Down", length(test.pred.prob.new))
test.pred.new[test.pred.prob.new>0.5] <- "Up"

roc.new <- roc(test_data$direction, test.pred.prob.new)
plot(roc.new, legacy.axes = TRUE, print.auc = TRUE)
```



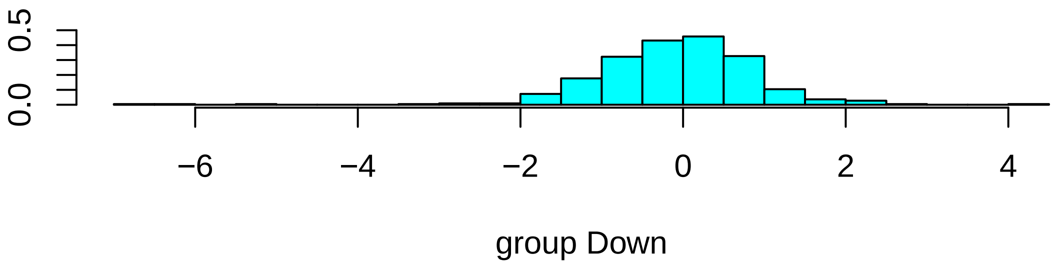
The AUC is 0.556, slightly better than the model above but still does not have good discriminatory ability.

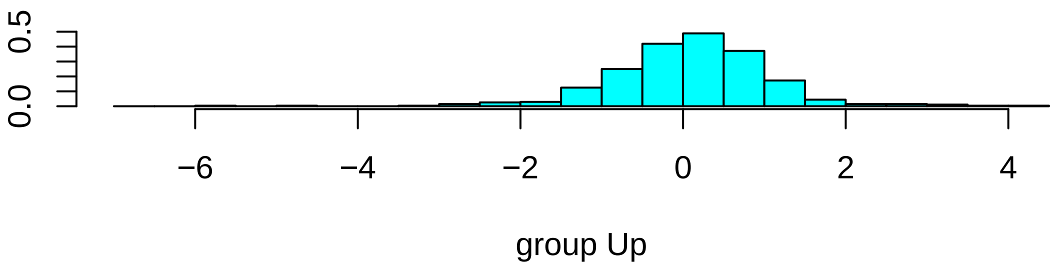
```
ctrl <- trainControl(method = "repeatedcv",
                     repeats = 5,
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE)
#Use caret for later comparison
glmnGrid <- expand.grid(.alpha = seq(0, 1, length = 6),
                        .lambda = exp(seq(-8, -2, length = 20)))
set.seed(7)
model.glmn <- train(x = train_data[,2:3],
                   y = train_data$direction,
                    method = "glmnet",
                    tuneGrid = glmnGrid,
                    metric = "ROC",
                    trControl = ctrl)
```

f) Use discriminant analysis – LDA and QDA

We can start with LDA first. We use the function "lda" in library "MASS" to perform LDA.

```
lda.fit <- lda(direction~lag1 + lag2, data = train_data)
plot(lda.fit)
```





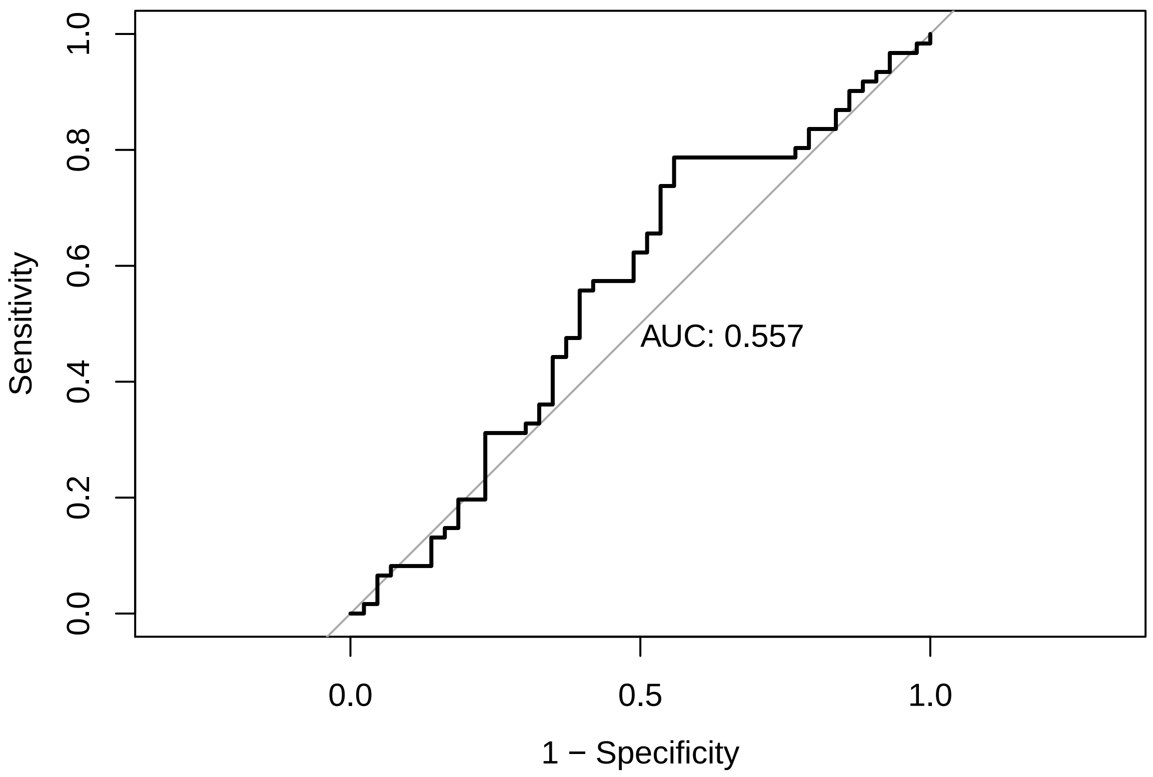
Evaluate the test set performance using ROC

```
lda.pred <- predict(lda.fit, newdata = test_data)
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_data$direction, lda.pred$posterior[,2],
               levels = c("Down", "Up"))

plot(roc.lda, legacy.axes = TRUE, print.auc = TRUE)
```



The AUC of this LDA model is 0.557, very similar discriminatory ability to the logistic regression model.

Using caret to find best LDA model (for comparison of tuned models later on)

```
set.seed(7)
model.lda <- train(x = train_data[,2:3],
                   y = train_data$direction,
                   method = "lda",
                   metric = "ROC",
                   trControl = ctrl)
```

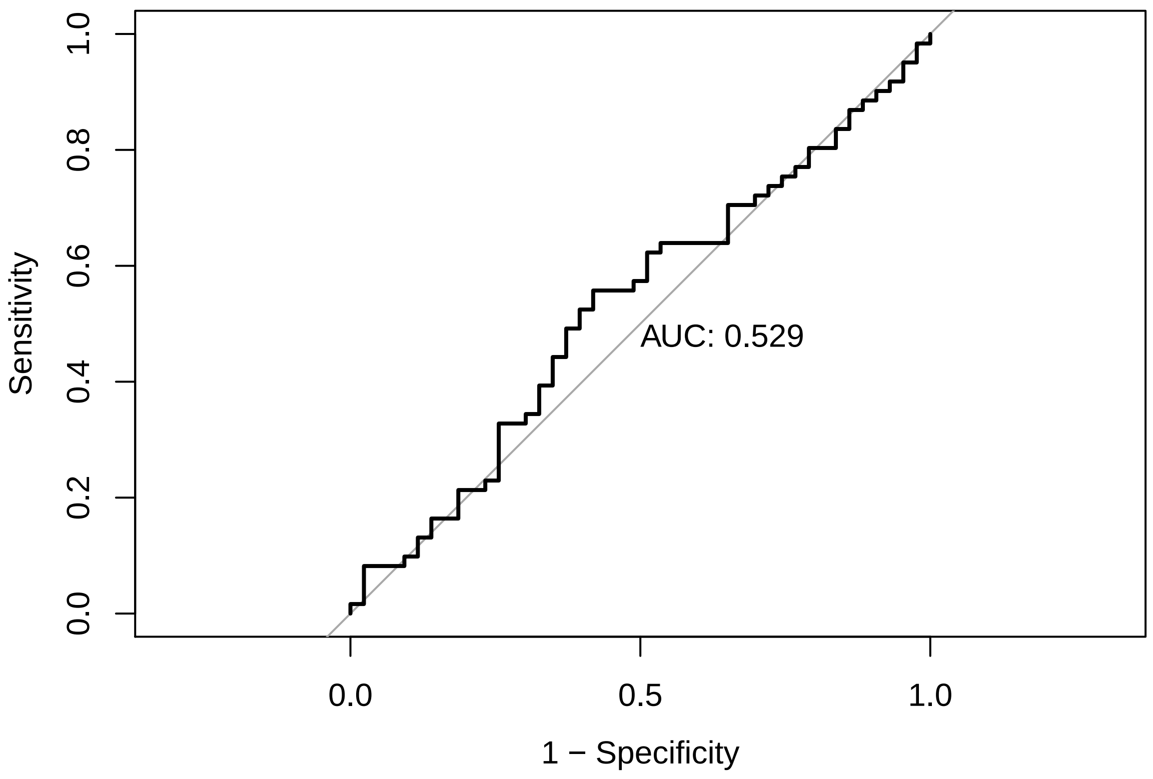
Next, we fit a QDA model using function "qda"

```
# use qda() in MASS
qda.fit <- qda(direction~lag1+lag2, data = train_data)

qda.pred <- predict(qda.fit, newdata = test_data)
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_data$direction, qda.pred$posterior[,2],
               levels = c("Down", "Up"))
plot(roc.qda, legacy.axes = TRUE, print.auc = TRUE)
```



The AUC of this QDA model is 0.529, slightly lower discriminatory ability to the two models above.

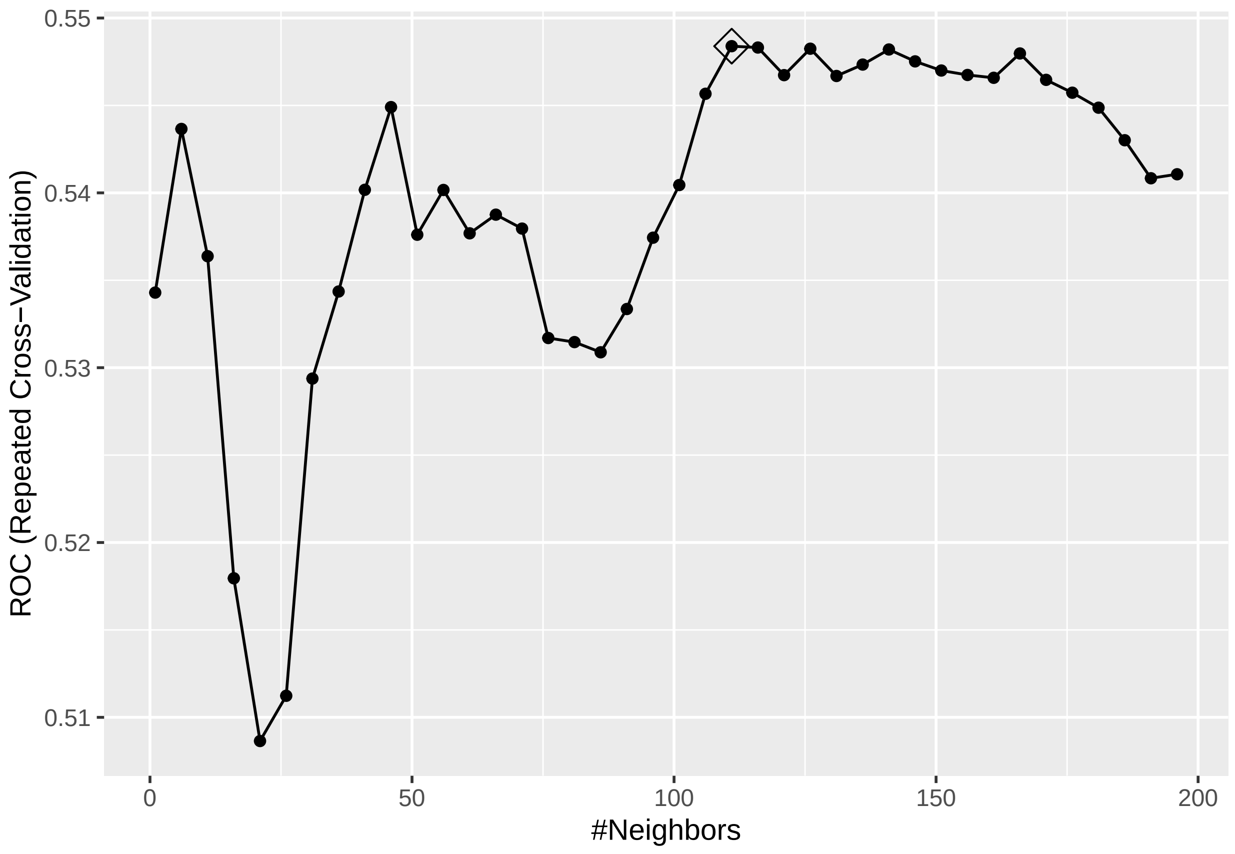
Using caret to find best QDA model (for comparisons later)

```
set.seed(7)
model.qda <- train(x = train_data[,2:3],
                   y = train_data$direction,
                   method = "qda",
                   metric = "ROC",
                   trControl = ctrl)
```

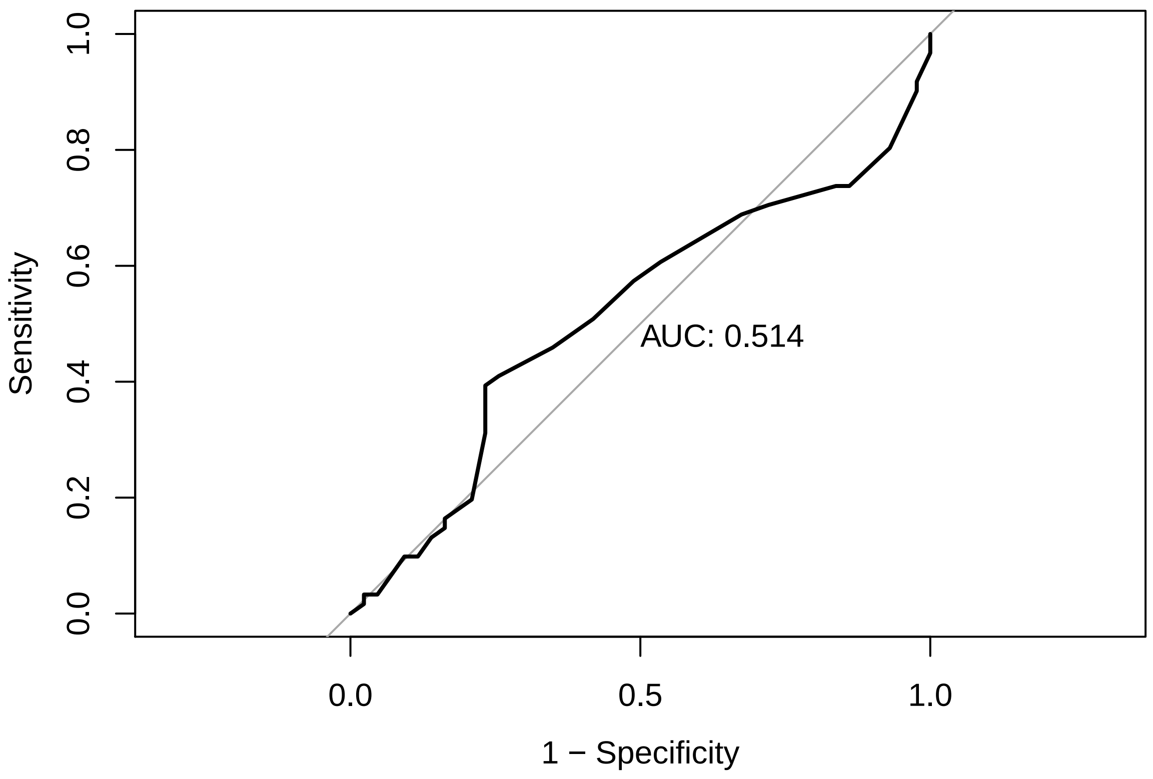
g) Use K-nearest-neighbors

```
set.seed(7)
model.knn <- train(x = train_data[,2:3],
                   y = train_data$direction,
                   method = "knn",
                   preProcess = c("center","scale"),
                   tuneGrid = data.frame(k = seq(1,200,by=5)),
                   trControl = ctrl)

ggplot(model.knn, highlight = TRUE)
```



```
knn.pred <- predict(model.knn, newdata = test_data, type = "prob")[,2]
roc.knn <- roc(test_data$direction, knn.pred, levels = c("Down", "Up"))
plot(roc.knn, legacy.axes = TRUE, print.auc = TRUE)
```



The AUC of this KNN model is 0.514, the lowest discriminatory ability of all four models looked at so far.

Model comparisons

```r
res <- resamples(list(GLMNET = model.glmn, LDA = model.lda,
                      QDA = model.qda, KNN = model.knn))
summary(res)
```

```
##
## Call:
## summary.resamples(object = res)
##
## Models: GLMNET, LDA, QDA, KNN
## Number of resamples: 50
##
## ROC
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## GLMNET 0.4208754 0.5068182 0.5502602 0.5509768 0.5825968 0.6673554    0
## LDA    0.4208754 0.5068182 0.5479798 0.5500203 0.5827020 0.6615702    0
## QDA    0.3737374 0.4930843 0.5267256 0.5255875 0.5529625 0.6462810    0
## KNN    0.4530724 0.5090479 0.5449380 0.5483898 0.5849403 0.6746633    0
##
## Sens
##              Min.    1st Qu.     Median       Mean    3rd Qu.      Max. NA's
## GLMNET 0.00000000 0.00000000 0.04545455 0.04307071 0.06818182 0.1818182    0
## LDA    0.00000000 0.06818182 0.09090909 0.09071717 0.11363636 0.2045455    0
## QDA    0.02272727 0.13636364 0.16843434 0.18731313 0.22727273 0.4090909    0
## KNN    0.15555556 0.25000000 0.27272727 0.27622222 0.31818182 0.3863636    0
```

```
##
## Spec
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## GLMNET 0.8545455 0.9132997 0.9629630 0.9519125 1.0000000 1.0000000    0
## LDA    0.8000000 0.8888889 0.9175084 0.9166397 0.9452020 1.0000000    0
## QDA    0.5818182 0.8037037 0.8441077 0.8332458 0.8848485 0.9629630    0
## KNN    0.5370370 0.6851852 0.7272727 0.7321077 0.7962963 0.8703704    0
```

Comments: Cross-validation shows regularized logistic regression seems to perform best with ROC as the criteria (highest median and mean AUC at 0.5502 and 0.551, respectively) compared to the LDA, QDA, and KNN models. LDA model also does comparatively well (with second-highest median and mean AUC).

The regularized logistic and LDA model also have high median specificity (at 0.963 and 0.918 respectively), which might be helpful if investors aim to correctly avoid stocks that having a downward trend (with high accuracy). This, however, might lead to higher chance of false negatives, which might prevent investors from investing in stocks that demonstrate upward trend.

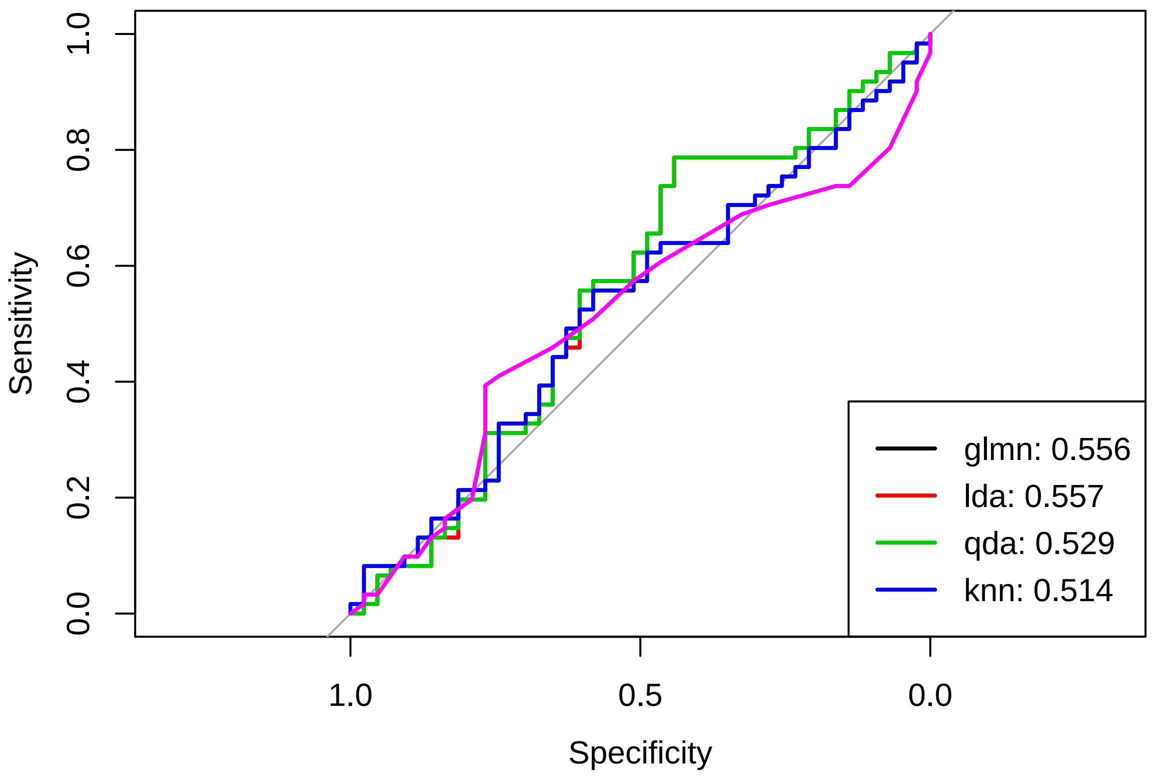
Next, we can pull all ROC curves together to look at the test set performance for these models

```r
glmn.pred <- predict(model.glmn, newdata = test_data, type = "prob")[,2]
lda.pred <- predict(model.lda, newdata = test_data, type = "prob")[,2]
qda.pred <- predict(model.qda, newdata = test_data, type = "prob")[,2]
knn.pred <- predict(model.knn, newdata = test_data, type = "prob")[,2]

roc.lda <- roc(test_data$direction, lda.pred)
roc.glmn <- roc(test_data$direction, glmn.pred)
roc.qda <- roc(test_data$direction, qda.pred)
roc.knn <- roc(test_data$direction, knn.pred)

auc <- c(roc.glmn$auc[1], roc.lda$auc[1],
         roc.qda$auc[1], roc.knn$auc[1])

plot(roc.glmn, col = 2)
plot(roc.lda, col = 3, add = TRUE)
plot(roc.qda, col = 4, add = TRUE)
plot(roc.knn, col = 6, add = TRUE)
modelNames <- c("glmn","lda","qda","knn")
legend("bottomright", legend = paste0(modelNames, ": ", round(auc,3)),
       col = 1:4, lwd = 2)
```



A comparison of ROC curves of these different models on the held-out set consolidates the finding that QDA and KNN do not perform as well as regularized logistic regression and LDA, although they all don't have excellent discriminatory ability.