Weekly Return

Winter Nguyen

2025-12-09

Summary:

This project analyzes the Weekly stock market dataset (1990–2010) to predict whether the market will go Up or Down using several classification methods. After exploring the data, Lag2 appears to be the only meaningful predictor. Logistic regression using all variables performs poorly, but simplifying the model to use only Lag2 increases test accuracy to 62.5%. LDA and QDA models using Lag2 achieve similar performance, while Naive Bayes and KNN perform worse due to data imbalance and overfitting. Across all experiments, including models with interactions, Volume, Lag1, and polynomial transformations, the best results consistently come from simple models driven by Lag2, confirming that it provides the strongest predictive signal.

A data frame with 1089 observations on the following 9 variables:

Year: The year that the observation was recorded

Lag1: Percentage return for previous week

Lag2: Percentage return for 2 weeks previous

Lag3: Percentage return for 3 weeks previous

Lag4: Percentage return for 4 weeks previous

Lag5: Percentage return for 5 weeks previous

Volume: Volume of shares traded (average number of daily shares traded in billions)

Today: Percentage return for this week

Direction: A factor with levels Down and Up indicating whether the market had a positive or negative return on a given week.

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

```
library(ISLR2)
dim(Weekly)
## [1] 1089    9
summary(Weekly)
##       Year           Lag1                Lag2                Lag3          
##  Min.   :1990   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950  
##  1st Qu.:1995   1st Qu.: -1.1540   1st Qu.: -1.1540   1st Qu.: -1.1580  
##  Median :2000   Median :  0.2410   Median :  0.2410   Median :  0.2410  
##  Mean   :2000   Mean   :  0.1506   Mean   :  0.1511   Mean   :  0.1472  
##  3rd Qu.:2005   3rd Qu.:  1.4050   3rd Qu.:  1.4090   3rd Qu.:  1.4090  
##  Max.   :2010   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260  
##       Lag4                Lag5              Volume           Today          
```

```
##  Min.   :-18.1950   Min.   :-18.1950   Min.   :0.08747   Min.   :-18.1950
##  1st Qu.: -1.1580   1st Qu.: -1.1660   1st Qu.:0.33202   1st Qu.: -1.1540
##  Median :  0.2380   Median :  0.2340   Median :1.00268   Median :  0.2410
##  Mean   :  0.1458   Mean   :  0.1399   Mean   :1.57462   Mean   :  0.1499
##  3rd Qu.:  1.4090   3rd Qu.:  1.4050   3rd Qu.:2.05373   3rd Qu.:  1.4050
##  Max.   : 12.0260   Max.   : 12.0260   Max.   :9.32821   Max.   : 12.0260
##  Direction
##  Down:484
##  Up  :605
##
##
##
##
cor(Weekly[1:8]) # correlation between variables except Direction
##                Year         Lag1        Lag2        Lag3         Lag4
## Year    1.00000000 -0.032289274 -0.03339001 -0.03000649 -0.031127923
## Lag1   -0.03228927  1.000000000 -0.07485305  0.05863568 -0.071273876
## Lag2   -0.03339001 -0.074853051  1.00000000 -0.07572091  0.058381535
## Lag3   -0.03000649  0.058635682 -0.07572091  1.00000000 -0.075395865
## Lag4   -0.03112792 -0.071273876  0.05838153 -0.07539587  1.000000000
## Lag5   -0.03051910 -0.008183096 -0.07249948  0.06065717 -0.075675027
## Volume  0.84194162 -0.064951313 -0.08551314 -0.06928771 -0.061074617
## Today  -0.03245989 -0.075031842  0.05916672 -0.07124364 -0.007825873
##                Lag5      Volume        Today
## Year   -0.030519101  0.84194162 -0.032459894
## Lag1   -0.008183096 -0.06495131 -0.075031842
## Lag2   -0.072499482 -0.08551314  0.059166717
## Lag3    0.060657175 -0.06928771 -0.071243639
## Lag4   -0.075675027 -0.06107462 -0.007825873
## Lag5    1.000000000 -0.05851741  0.011012698
## Volume -0.058517414  1.00000000 -0.033077783
## Today   0.011012698 -0.03307778  1.000000000
```

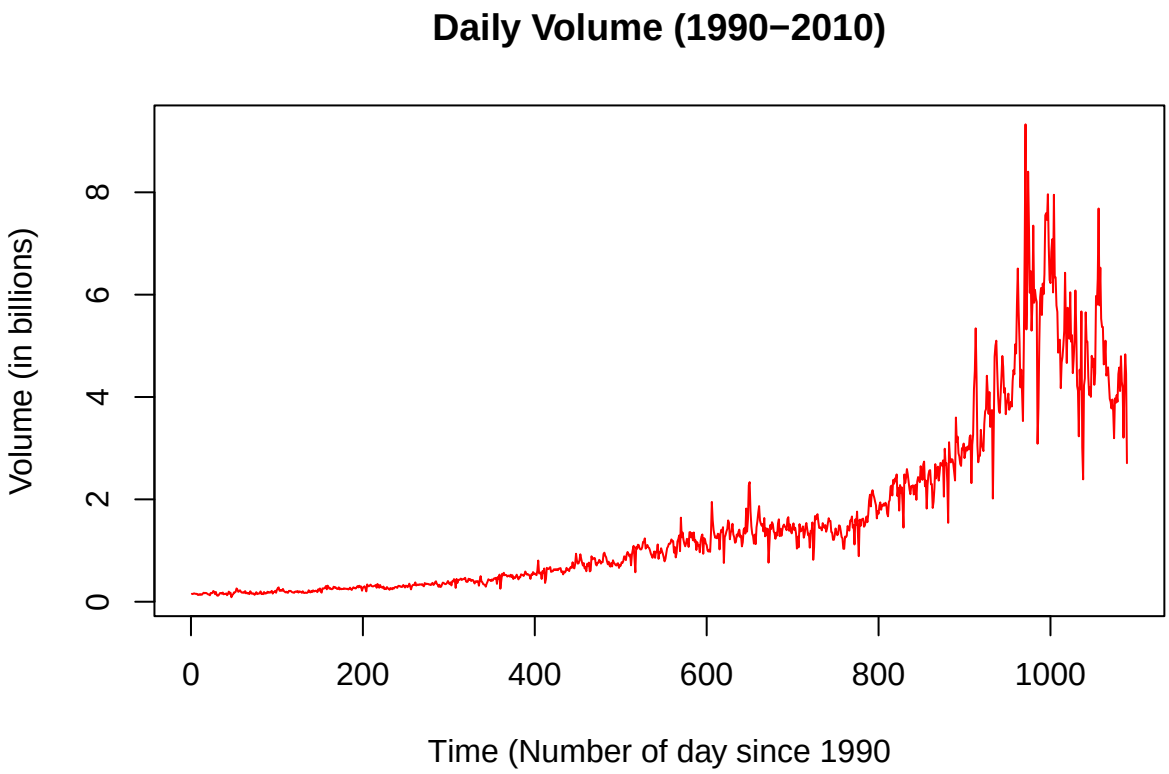
In general, the percentage returns for each of the five previous trading days have the approximately equal

- Mean: around 0.24%
- 1st quarter : -1.16%
- 3rd quarter : 1.4%
- Range from -18.195% to 12.026%

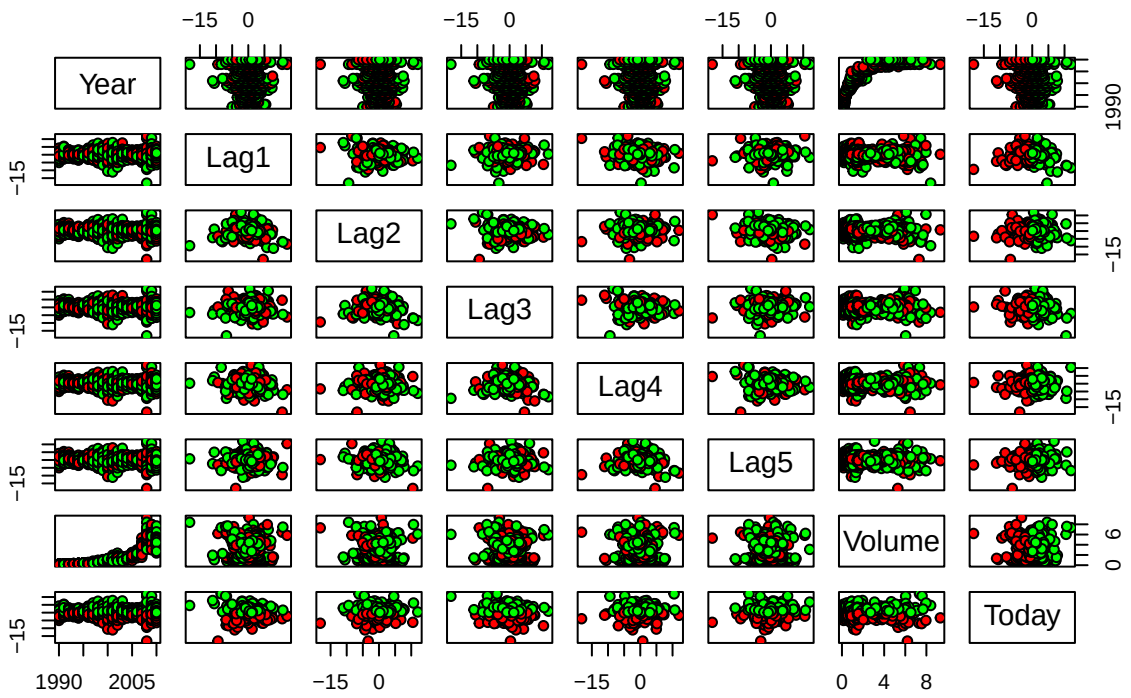
So we expect to see a similar shape for each of the variables.

```
attach(Weekly[1:8])
col_Direction=c("red","green")[unclass(Weekly$Direction)]

plot(Weekly$Volume, type="l", col="red",
     xlab="Time (Number of day since 1990",
     ylab="Volume (in billions)",
     main="Daily Volume (1990-2010)")
```




```
pairs(Weekly[1:8],
      bg=col_Direction,
      main="Pairwise Plot - Relationship Between Variables",
      pch=21)
```



From the scatterplots and correlation matrix, we observe that Volume increases with Year, while the remaining predictors show very low correlation with each other. Although low correlation does not guarantee independence, it suggests weak linear relationships among these variables.

(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. Use the summary function to print the results.

```
glm.fits=glm(Weekly$Direction ~ Lag1 + Lag2 +Lag3+Lag4+Lag5+Volume, family = binomial, data=Weekly)
summary(glm.fits)
##
## Call:
## glm(formula = Weekly$Direction ~ Lag1 + Lag2 + Lag3 + Lag4 +
##     Lag5 + Volume, family = binomial, data = Weekly)
##
## 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
##
```

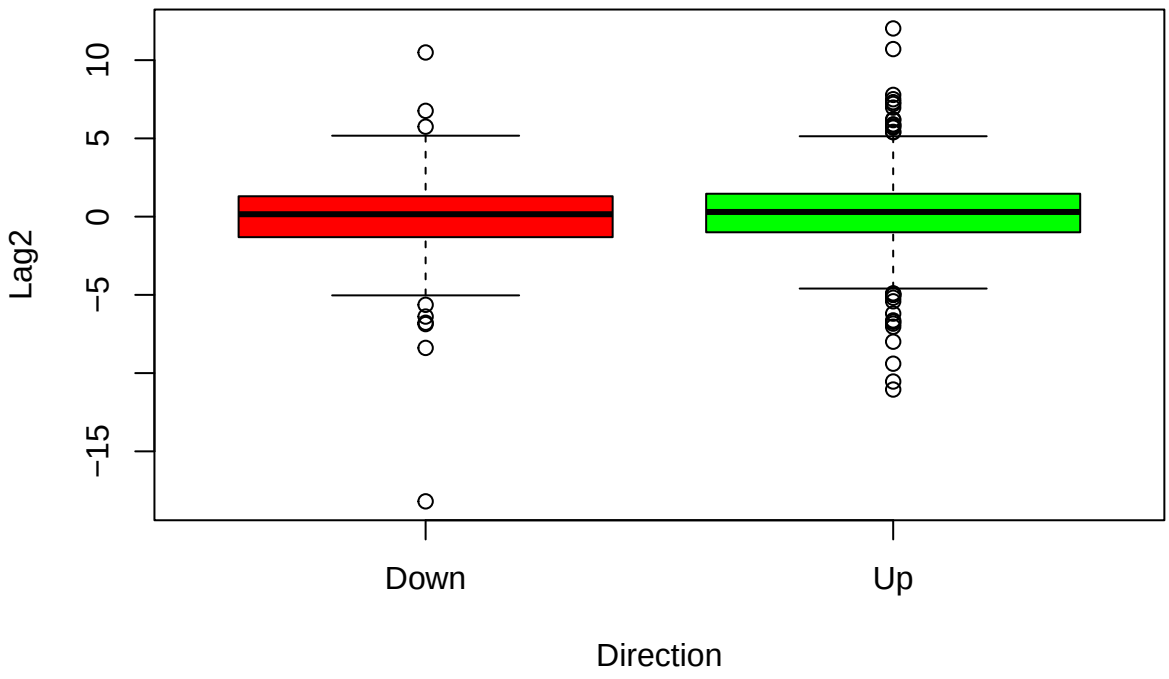
```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1496.2  on 1088  degrees of freedom
## Residual deviance: 1486.4  on 1082  degrees of freedom
## AIC: 1500.4
##
## Number of Fisher Scoring iterations: 4
```

There is statistically significant relationship between Lag2 and Direction as the p-value is 0.0296 which is less than 0.05. Since there is no correlation between predictor variables, we can say that Lag2 is not under effect of multicolinearity. Let us use VIF to confirm

```
library(car)
## Loading required package: carData
vif(glm.fits)
##     Lag1     Lag2     Lag3     Lag4     Lag5   Volume
## 1.017298 1.023770 1.019378 1.027388 1.014676 1.024789
```

VIF is approximately 1, confirming above statement.

```
plot(Weekly$Direction, Lag2,
     col=c("red","green"),
     xlab="Direction",
     ylab="Lag2")
```



(c) Compute the confusion matrix and overall fraction of correct predictions

```
# Confusion Matrix
glm.probs=predict(glm.fits,type="response")
contrasts(Weekly$Direction)
##      Up
## Down  0
## Up    1
glm.pred=rep("Down", 1089)
glm.pred[glm.probs>0.5]="Up"
table_glm=table(glm.pred, Weekly$Direction)
table_glm
##
## glm.pred Down  Up
##     Down   54  48
##     Up    430 557
correction_glm=mean(glm.pred==Weekly$Direction)
er_rate=1-correction_glm


#always predict up correction percentage
always_predict_up=sum(Weekly$Direction == "Up")/1089
```

So, the logistic regression model correctly predicted 56.11% of the time (with error training rate to be 43.89%, let us denote this model "Model A". By observation we see that.

-The model is biased toward predicting "Up" (605 "Up", while only 484 "Down"). So, it is very bad at predicting the "Down" days

-If we always predict "Up" then the percentage correct is 55.56, so the model is only -11.67% better.

However, the training error rate often underestimate the test error rate. Therefore, split data into training set and test set, then fit data on the training set and use that model to predict data in test set would give us a more realistic error rate.

```
train=(Weekly$Year < 2009 )
Weekly.test=Weekly[!train,]
Direction.test=Weekly$Direction[!train]
glm.fits.train=glm(Weekly$Direction ~ Lag1 + Lag2 +Lag3+Lag4+Lag5+
                     Volume, family = binomial, data=Weekly, subset = train)
glm.probs.train=predict(glm.fits.train, Weekly.test, type="response")
glm.pred.train=rep("Down", length(Direction.test))
glm.pred.train[glm.probs.train >0.5]="Up"
table(glm.pred.train,
      Direction.test)
##               Direction.test
## glm.pred.train Down Up
##           Down   31 44
##           Up     12 17
correction_glm_fix=mean(glm.pred.train==Direction.test)
correction_glm_fix
## [1] 0.4615385
```

So, the correction rate is 46.15 with the test error rate to be 53.85 %, increase in test error is expected and in fact this give a more reliable information than previous model. In general, this model is worse than just predict "Up" all the time. I will discuss more in the incoming question. Let us denote this model "Model B"

(d) Now fit the logistic regression model using a training data period from 1990 to 2008, with Lag2 as the only predictor. Compute the confusion matrix and the overall fraction of correct predictions for the held out data (that is, the data from 2009 and 2010).

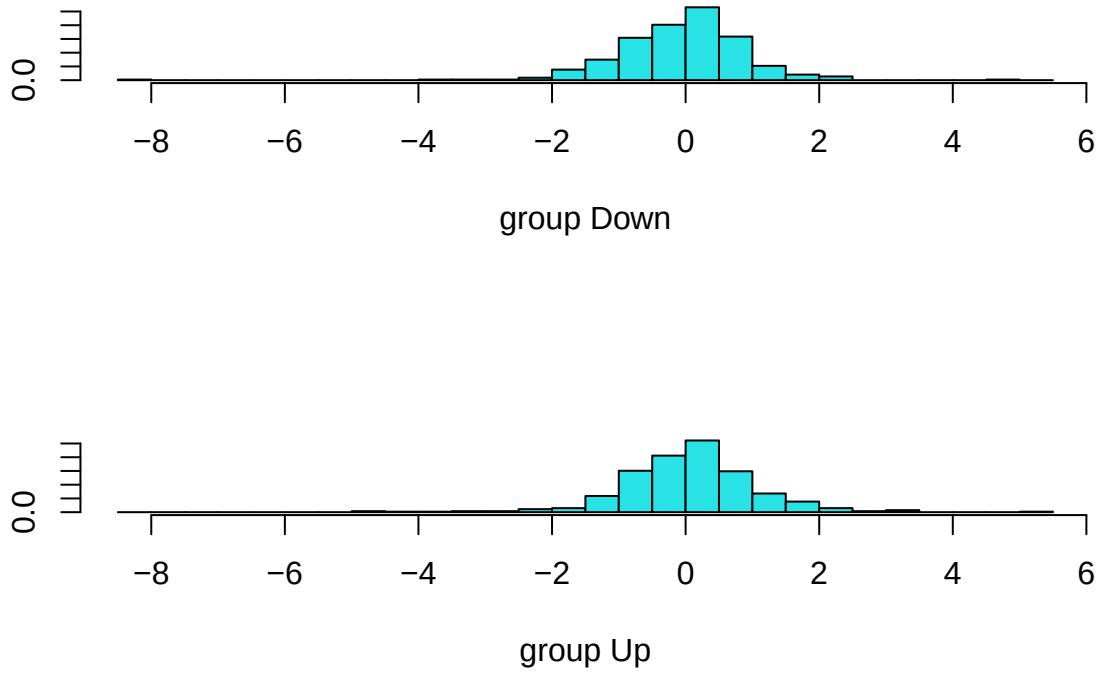
```
glm.fits.train2=glm(Weekly$Direction ~Lag2 , family = binomial, data=Weekly, subset = train)
glm.probs.train2=predict(glm.fits.train2, Weekly.test, type="response")
glm.pred.train2=rep("Down", length(Direction.test))
glm.pred.train2[glm.probs.train2 >0.5]="Up"
table(glm.pred.train2,
      Direction.test)
##                Direction.test
## glm.pred.train2 Down Up
##            Down    9  5
##            Up     34 56
correction_glm2=mean(glm.pred.train2==Direction.test)
correction_glm2
## [1] 0.625
correctionrate.data <- data.frame(
  ModelA = correction_glm,
  ModelB = correction_glm_fix,
  ModelC = correction_glm2,
  row.names="Correction Rate"
)
correctionrate.data
##                    ModelA    ModelB ModelC
## Correction Rate 0.5610652 0.4615385  0.625
```

The correction rate for this model (Lag2 as the only variable, denoted Model C) is 62.5% which is greater than the correction rate of splited training/test sets model using all variable (Model B) and the model using full data as training set (Model A). As mentioned above, it would be more realistic to compared training Model B and C because error rate in Model A is underestimated. Overall, Model C performed better with higher correction rate, indicate Lag2 should be prioritized when predict Direction.

(e) Repeat (d) using LDA.

```
library(MASS)
##
## Attaching package: 'MASS'
## The following object is masked from 'package:ISLR2':
##
##     Boston
lda.fit=lda(Direction~Lag2, data=Weekly, subset=train)
lda.fit
## Call:
## lda(Direction ~ Lag2, data = Weekly, subset = train)
##
## Prior probabilities of groups:
##      Down        Up
## 0.4477157 0.5522843
##
## Group means:
##              Lag2
## Down -0.03568254
## Up    0.26036581
```

```
##
## Coefficients of linear discriminants:
##             LD1
## Lag2 0.4414162
plot(lda.fit)
```



The out put indicates that prior propability for "Down" is 44.1% and for "Up" is 55.8% . In order words, 44.1% of trainining observations correspond to days during which the market **went down**.

```
lda.pred=predict(lda.fit, Weekly.test)
lda.class=lda.pred$class
table(lda.class, Direction.test)
##          Direction.test
## lda.class Down Up
##      Down    9  5
##      Up     34 56
correction_lda=mean(lda.class==Direction.test) # correction rate
correction_lda
## [1] 0.625
#number of days LDA predicted "Down"
down_days=sum(lda.pred$posterior[,1]>=0.5)

#number of days LDA predicted "Up"
up_days=sum(lda.pred$posterior[,1]<0.5)
```

Using the same split in d), LDA model gives the correction rate to be 58.7%, the model predicted correctly 58.7% of the time. The model predicted Down for 14 days, and Up for 90 days. Since Up days is larger than Down days, the market is on up trend overall. However, it doesn't nessacary mean the market has positive return.

Lag2 represents the market return from two weeks ago. The LDA group means show that two weeks prior to an Up week, the market return averages +0.26%, while two weeks prior to a Down week, the return averages -0.036%. Therefore, positive returns from two weeks earlier make an Up week more likely.

(f) Repeat (d) using QDA.

```
qda.fit=qda(Direction~Lag2, data=Weekly, subset=train)
qda.fit
## Call:
## qda(Direction ~ Lag2, data = Weekly, subset = train)
##
## Prior probabilities of groups:
##      Down        Up
## 0.4477157 0.5522843
##
## Group means:
##             Lag2
## Down -0.03568254
## Up    0.26036581
qda.class=predict(qda.fit, Weekly.test)$class # Up - Down
table(qda.class, Direction.test)
##          Direction.test
## qda.class Down Up
##      Down    0  0
##      Up     43 61
cat("The percentage of correct prediction")
## The percentage of correct prediction
correction_qda=mean(qda.class==Direction.test)
correction_qda
## [1] 0.5865385
```

Prior probabilities of QDA model are 44.8% for Down and 55.2% for "Up" (of training data)

The QDA model estimates separate normal distributions for Up and Down weeks. The group means indicate that two weeks prior to an Up week, the average return (Lag2) is +0.26%, whereas two weeks prior to a Down week, the average Lag2 return is -0.036%

so, if Lag2 is high (positive), QDA is more likely to predict "Up". If Lag2 is negative, QDA is more likely to predict "Down"

Notice the QDA predicts every week as "Up", the overall correction rate is 58.65% which is affected by the fact that "Up" weeks are more common in the test set.

(g) Repeat (d) using KNN with K = 1.

```
#Implement KNN wiht k = 1
library(class)
train.X=as.matrix(Weekly$Lag2[train]) #knn need matrices
test.X=as.matrix(Weekly$Lag2[!train])
train.Direction=Weekly$Direction[train]
set.seed(1)
knn.pred=knn(train.X, test.X, train.Direction,k=1)
```

```
table(knn.pred, Direction.test)
##         Direction.test
## knn.pred Down Up
##     Down   21 30
##     Up     22 31
correction_knn=mean(knn.pred==Direction.test)

cat("Percentage of correction:",correction_knn)
## Percentage of correction: 0.5
```

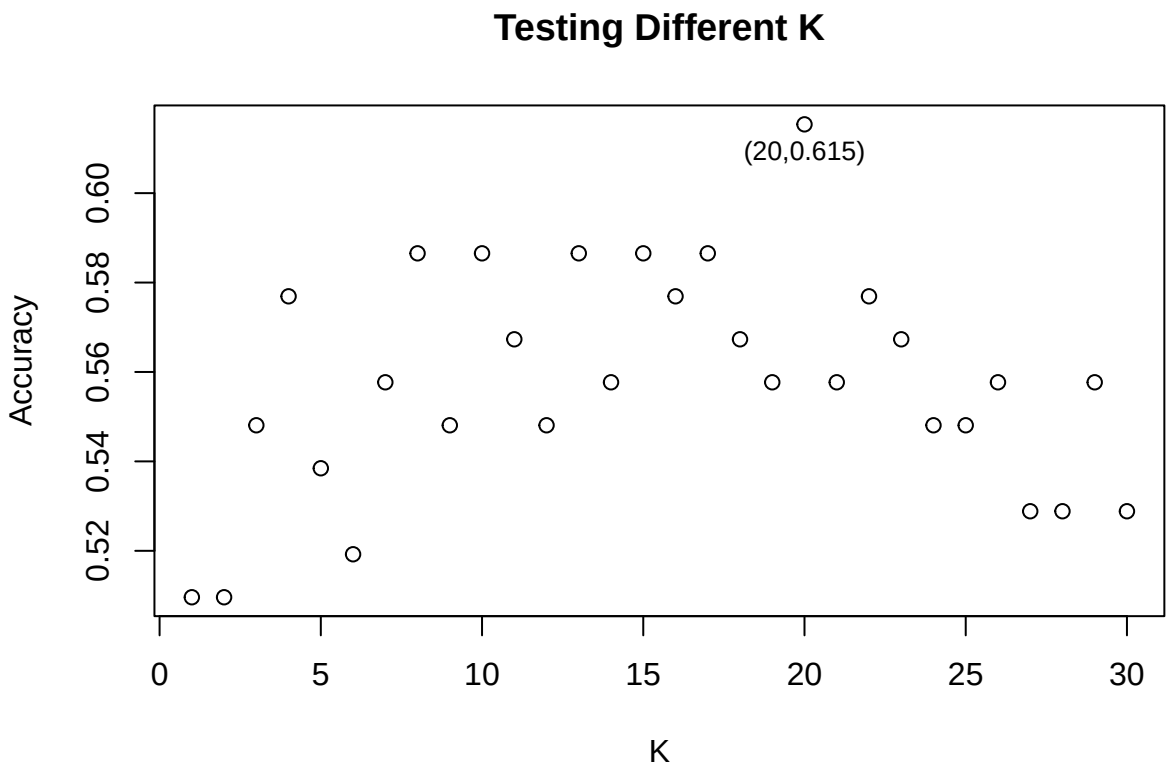
For K-Nearest-Neighbors, only 50% of the observations are correctly predicted which is not good compared to LDA, QDA, and logistic regression. However, this maybe the result of overfiting data because K=1 results in overly flexible fit to the data. Let us try different K.

```
set.seed(100)
accuracy_y=c()

for (i in 1:30) {
  knn.pred=knn(train.X, test.X, train.Direction,k=i)
  accuracy=mean(knn.pred == Direction.test)
  accuracy_y=c(accuracy_y, accuracy)}


plot(1:30, accuracy_y,
     main="Testing Different K",
     xlab="K",
     ylab="Accuracy")

# since we have the "Testing Different K" graph
# and we only care about the maximum point,
# I would only print the highest accuracy and
# its associated k
text(which.max(accuracy_y),max(accuracy_y),
     labels=paste0("(",which.max(accuracy_y),",",round(max(accuracy_y),3),")"),
     pos=1,
     cex=0.8)
```




```
cat("Percentage of correction for k=", which.max(accuracy_y), ":", max(accuracy_y))
## Percentage of correction for k= 20 : 0.6153846
```

K=4 is good choice since after that the percentage of correction is reduced or doesn't change much. The best K should be K=20 produce the most accurate result but only for this set of data (fixed using set.seed function). We can determine the best K using Cross Validation method, and that require the different splits of training set and the test set. I will implement this method in the incoming question.

(h) Repeat (d) using naive Bayes

As stated above, we assume independent predictors (since the correlations are relatively low), and has the normal distribution. So, we can use the function naiveBayes.

```
library(e1071)
nb.fit=naiveBayes(Direction~Lag2, data=Weekly, subset=train)
nb.fit
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##      Down        Up
## 0.4477157 0.5522843
```

```
##
## Conditional probabilities:
##       Lag2
## Y             [,1]     [,2]
##   Down -0.03568254 2.199504
##   Up    0.26036581 2.317485
```

Prior probabilities of Bayes model are 44.8% for "Down" and 55.2% for "Up" (of training data)

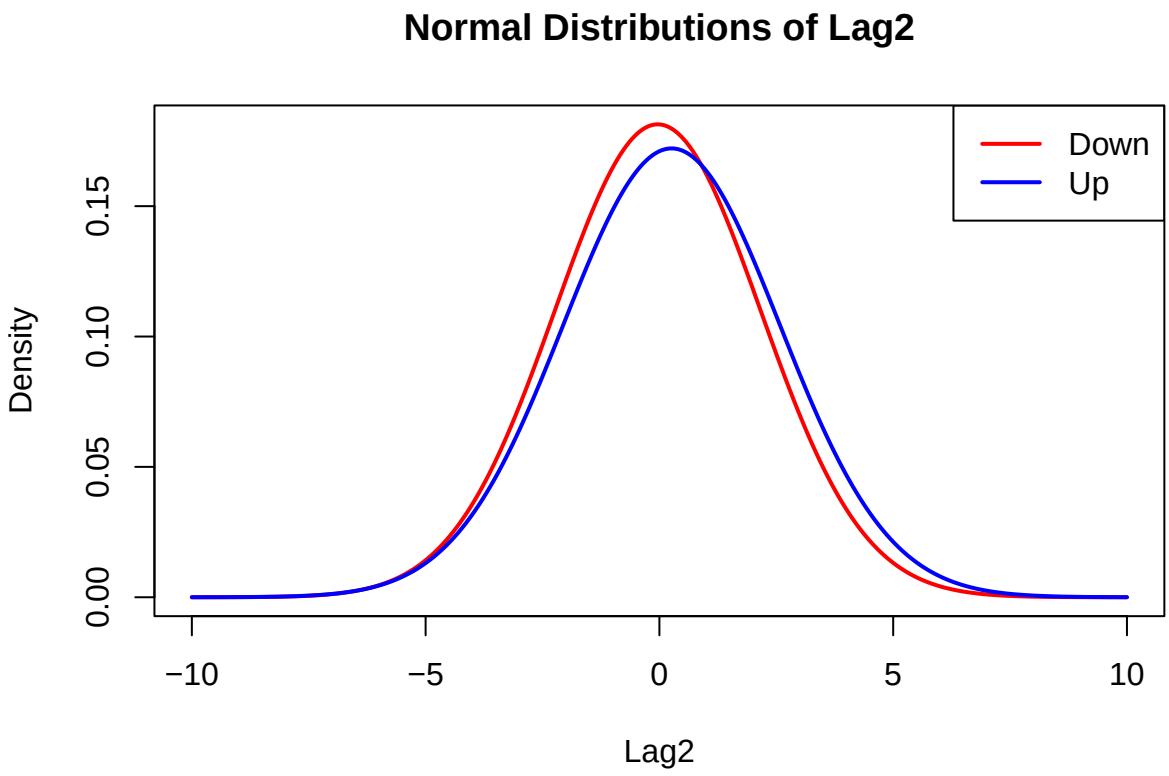
Naive Bayes assumes Lag2 is normally distributed within each class. Up weeks have a higher mean Lag2 (0.26) than Down weeks (–0.036), so larger Lag2 values indicate a higher probability of an Up week. Variances for Lag2 are similar between the classes (2.19 and 2.31).

```
#visualize the two
mu_down =-0.03568254
sd_down = 2.199504

mu_up = 0.26036581
sd_up = 2.317485

x=seq(-10,10,length=500)
plot(x, dnorm(x, mu_down, sd_down),
     type="l", lwd=2, col="red",
     xlab="Lag2", ylab="Density",
     main="Normal Distributions of Lag2")

lines(x, dnorm(x, mu_up, sd_up),
      lwd=2, col="blue")
legend("topright", legend=c("Down", "Up"),
       col=c("red","blue"), lwd=2)
```




```
nb.class=predict(nb.fit, Weekly.test)
table(nb.class, Direction.test)
##         Direction.test
## nb.class Down Up
##     Down    0  0
##     Up     43 61
correction_nb=mean(nb.class==Direction.test)
correction_nb
## [1] 0.5865385
```

So, the model predicted up for every observation, which makes sense since the probability of "Up" is higher than probability of "Down" in data set ($55.2\% > 44.8\%$). Also, the two Gaussian curves overlap almost perfectly, that is for almost every Lag2 value, the conditional probability of "Up" given Lag2 is greater than the conditional probability of "Down" given Lag2, so Naive Bayes always picks Up.

```
# made a percentage of correction table
results = data.frame(
  Method = c("Logistic Regression", "LDA", "QDA", "KNN, K=1", "Naive Bayes"),
  Accuracy = c(correction_glm, correction_lda, correction_qda, correction_knn, correction_nb)
)
results
##                Method  Accuracy
## 1 Logistic Regression 0.5610652
## 2                 LDA 0.6250000
## 3                 QDA 0.5865385
```

```
## 4            KNN, K=1 0.5000000
## 5        Naive Bayes 0.5865385
```

Based on the result, **LDA has the best performance**. Another observation is that LDA, QDA and Naive Bayes have identical means for "Up" and "Down". This makes sense since we use the same training data set and variable Lag2, so their means are supposed to be the same. Notice that QDA and Naive Bayes give very similar results in this case. Since we are using only one predictor (Lag2), both methods fit a normal distribution for Lag2 in each class, so their decision rules are almost the same. The Naive Bayes independence assumption doesn't really matter when there is only one predictor.

(j) Experiment with different combinations of predictors, including possible transformations and interactions, for each of the methods. Report the variables, method, and associated confusion matrix that appears to provide the best results on the held out data. Note that you should also experiment with values for K in the KNN classifier.

Since we are trying to predict the next week's return from the previous weeks' returns, so I think the two most recent weeks would give a better prediction. Let us try Lag1, Lag2, and Volume.

```
models = list(
  Direction ~ Lag1 + Lag2,
  Direction ~ Lag1 + Lag2 + Volume,
  Direction ~ Lag1 + Lag2 + Lag1:Lag2,
  Direction ~ poly(Lag2, 2)
)
#logistic regression
acc_lg = numeric(length(models))      # Store accuracies
conf_lg = vector("list", length(models))  # Store confusion matrices

for (i in 1:length(models)) {

  formula_i <- models[[i]]

  glm.fit <- glm(formula_i, data=Weekly, family=binomial, subset=train)

  glm.probs <- predict(glm.fit, Weekly.test, type="response")
  glm.pred  <- ifelse(glm.probs > 0.5, "Up", "Down")

  acc_lg[i]    <- mean(glm.pred == Weekly.test$Direction)
  conf_lg[[i]] <- table(glm.pred, Weekly.test$Direction)
}

data_lg=data.frame(
  Model = sapply(models, function(f) deparse(f)),
  Accuracy = acc_lg
)
best.index_lg = which.max(acc_lg)
best.model_lg = models[[best.index_lg]]
best.model_lg
## Direction ~ poly(Lag2, 2)
conf_lg[[best.index_lg]]
##
## glm.pred Down Up
##     Down    8  4
##     Up     35 57
```

```
acc_lg[best.index_lg]
## [1] 0.625
```

```
# LDA
conf_lda=vector("list", length(models))
acc_lda=numeric(length(models))
for (i in 1:length(models)) {

  formula_i <- models[[i]]
lda.fit=lda(formula_i, data=Weekly, subset=train)
lda.pred=predict(lda.fit, Weekly.test)
lda.class=lda.pred$class
conf_lda[[i]]=table(lda.class, Direction.test)
acc_lda[i]=mean(lda.class==Direction.test)
}
data_lda=data.frame(
  Model = sapply(models, function(f) deparse(f)),
  Accuracy = acc_lda)

best.index_lda = which.max(acc_lda)
best.model_lda = models[[best.index_lda]]
best.model_lda
## Direction ~ poly(Lag2, 2)
conf_lda[[best.index_lda]]
##          Direction.test
## lda.class Down Up
##      Down    7  4
##      Up     36 57
acc_lda[best.index_lda]
## [1] 0.6153846
```

```
#QDA
conf_qda=vector("list", length(models))
acc_qda=numeric(length(models))
for (i in 1:length(models)) {
  formula_i = models[[i]]
qda.fit=qda(formula_i, data=Weekly, subset=train)
qda.class=predict(qda.fit, Weekly.test)$class # Up - Down
conf_qda[[i]]=table(qda.class, Direction.test)
acc_qda[i]=mean(qda.class==Direction.test)
}
data_qda=data.frame(
  Model = sapply(models, function(f) deparse(f)),
  Accuracy = acc_qda)

best.index_qda = which.max(acc_qda)
best.model_qda = models[[best.index_qda]]
best.model_qda
## Direction ~ poly(Lag2, 2)
conf_qda[[best.index_qda]]
##          Direction.test
## qda.class Down Up
##      Down    7  3
```

```
##      Up     36 58
acc_qda[best.index_qda]
## [1] 0.625
```

```
#Naive Bayes
conf_nb=vector("list", length(models))
acc_nb=numeric(length(models))
for (i in 1:length(models)) {

  formula_i <- models[[i]]
nb.fit=naiveBayes(Direction~Lag2, data=Weekly, subset=train)
nb.class=predict(nb.fit, Weekly.test)
conf_nb[[i]]=table(nb.class, Direction.test)
acc_nb=mean(nb.class==Direction.test)
}
data_nb=data.frame(
  Model = sapply(models, function(f) deparse(f)),
  Accuracy = acc_nb)

best.index_nb = which.max(acc_nb)
best.model_nb = models[[best.index_nb]]
best.model_nb
## Direction ~ Lag1 + Lag2
conf_nb[[best.index_nb]]
##         Direction.test
## nb.class Down Up
##     Down    0  0
##     Up     43 61
acc_nb[best.index_nb]
## [1] 0.5865385
```

```
#KNN with cross validation to determine the best K
train_idx = Weekly$Year < 2009
Weekly.train =Weekly[train_idx, ]
Weekly.test  = Weekly[!train_idx, ]

Y_train = Weekly.train$Direction
K_values = 1:50
nfolds = 10
set.seed(300)


data.knn = list()   # Store results for each model

for (m in 1:length(models)) {

  # Extract predictor names except the response
  vars = all.vars(models[[m]])[-1]

  # Create standardized training matrix
  X_train = scale( Weekly.train[, vars, drop=FALSE] )

  # Create fold assignments
```

```
  fold_id = sample(rep(1:nfolds, length.out = nrow(X_train)))

  # Store CV accuracy for each K
  cv_acc = numeric(length(K_values))

  for (i in 1:length(K_values)) {

    k_val = K_values[i]
    fold_acc = numeric(nfolds)

    for (f in 1:nfolds) {

      val_bool = (fold_id == f)
      train_bool = !val_bool

      knn_pred = knn(
        train = X_train[train_bool, , drop=FALSE],
        test  = X_train[val_bool, , drop=FALSE],
        cl    = Y_train[train_bool],
        k     = k_val
      )


      fold_acc[f] = mean(knn_pred == Y_train[val_bool])
    }

    cv_acc[i] = mean(fold_acc)
  }

  # Save results for this model
  data.knn[[m]] = list(
    formula = models[[m]],
    cv_accuracy = cv_acc,
    best_k = K_values[which.max(cv_acc)],
    best_cv_acc = max(cv_acc)
  )
}

cv_vec = sapply(data.knn, function(x) x$best_cv_acc)
best_index_knn = which.max(cv_vec)
best_model_knn = data.knn[[best_index_knn]]

cat("Best KNN Model:\n")
## Best KNN Model:
cat("Formula:", deparse(best_model_knn$formula), "\n")
## Formula: Direction ~ Lag1 + Lag2 + Volume
cat("Best K:", best_model_knn$best_k, "\n")
## Best K: 24
cat("Best CV Accuracy:", best_model_knn$best_cv_acc, "\n")
## Best CV Accuracy: 0.5857349
#test new K on the same training set on previous problem
train = (Weekly$Year < 2009)
best_vars = c("Lag1", "Lag2", "Volume")
```

```
train.X = scale( Weekly.train[, best_vars] )
test.X = scale(
  Weekly.test[, best_vars],
  center = attr(train.X, "scaled:center"),
  scale  = attr(train.X, "scaled:scale"))
Direction.train = Weekly$Direction[train]
Direction.test  = Weekly$Direction[!train]
knn.pred=knn(train.X, test.X, Direction.train, k=best_model_knn$best_k)

#confusion table
conf_knn = table(knn.pred, Direction.test)
conf_knn
##         Direction.test
## knn.pred Down Up
##     Down   27 37
##     Up     16 24
mean(knn.pred==Direction.test)
## [1] 0.4903846
```

I tested:

- Direction ~ Lag1 + Lag2
- Direction ~ Lag1 + Lag2 + Volume
- Direction ~ Lag1 + Lag2 + Lag1:Lag2
- Direction ~ poly(Lag2, 2)

using logistic regression, LDA, QDA, and 10 folds cross validation KNN, and Naive Bayes.

The result indicates that:

- Logistic regression: Direction ~ poly(Lag2, 2)
- LDA: Direction ~ Lag2
- QDA: Direction ~ poly(Lag2, 2)

provides the best performance with correction rate is 62.5%, below is the detail of the model.

```
#logistic regression
best.model_lg # model configuration
## Direction ~ poly(Lag2, 2)
acc_lg[best.index_lg] # correction rate
## [1] 0.625
conf_lg[[best.index_lg]] # confusion matrix
##
## glm.pred Down Up
##     Down    8  4
##     Up     35 57
#
#QDA
best.model_qda # model configuration
```

```
## Direction ~ poly(Lag2, 2)
conf_qda[[best.index_qda]] # confusion matrix
##          Direction.test
## qda.class Down Up
##      Down    7  3
##      Up     36 58
acc_qda[best.index_qda] # correction rate
## [1] 0.625
#
#LDA
deparse(Direction ~ Lag2) # model configuration
## [1] "Direction ~ Lag2"
table(lda.class, Direction.test) # confusion matrix
##          Direction.test
## lda.class Down Up
##      Down    7  4
##      Up     36 57
correction_lda # correction rate
## [1] 0.625
```