Lab7

Dr. Purna Gamage

3/9/2021

```
library(ISLR)
library(gam)
```

```
## Loading required package: splines
```

```
## Loading required package: foreach
```

```
## Loaded gam 1.16.1
```

```
library(tidyverse)
```

```
## -- Attaching packages ----- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.4
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## -- Conflicts -------- tidyverse_conflicts() --
## x purrr::accumulate() masks foreach::accumulate()
## x dplyr::filter()     masks stats::filter()
## x dplyr::lag()        masks stats::lag()
## x purrr::when()       masks foreach::when()
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(dslabs)
```

#Example

```
data(heights)
head(heights)
```

```
##      sex height
## 1   Male     75
## 2   Male     70
## 3   Male     68
## 4   Male     74
## 5   Male     61
```

```
## 6 Female     65
```

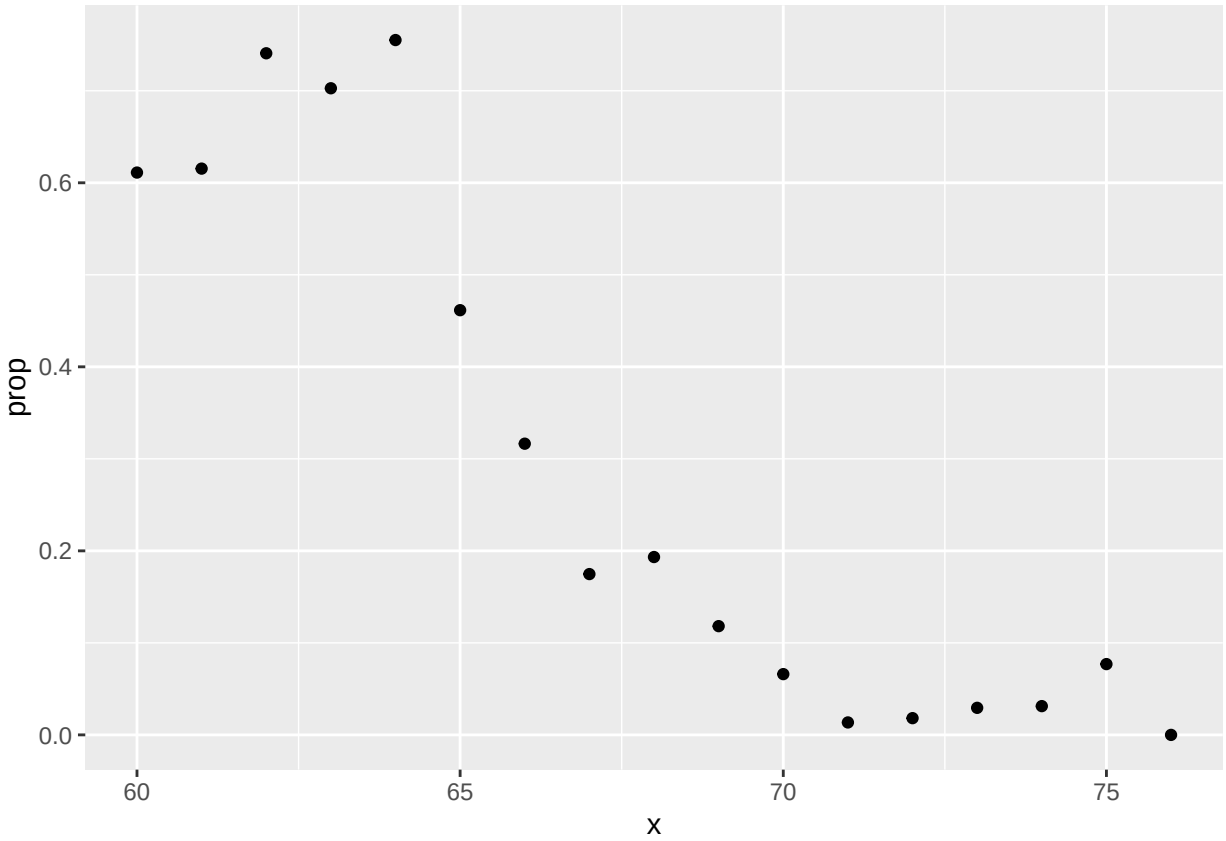
```
set.seed(2008)
y<- heights$sex
test_index <- createDataPartition(y, times = 1, p = 0.5, list = FALSE)
test_set <- heights[test_index, ]
train_set <- heights[-test_index, ]
```

let's provide a prediction for a student that is 66 inches tall. What is the conditional probability of being female if you are 66 inches tall? In our dataset, we can estimate this by rounding to the nearest inch and computing:

```
train_set %>%
  filter(round(height)==66) %>%
  summarize(y_hat = mean(sex=="Female"))
```

```
##     y_hat
## 1 0.34375
```

```
heights %>%
  mutate(x = round(height)) %>%
  group_by(x) %>%
  filter(n() >= 10) %>%
  summarize(prop = mean(sex == "Female")) %>%
  ggplot(aes(x, prop)) +
  geom_point()
```



the results from the plot above look close to linear.

Fit with least squares.

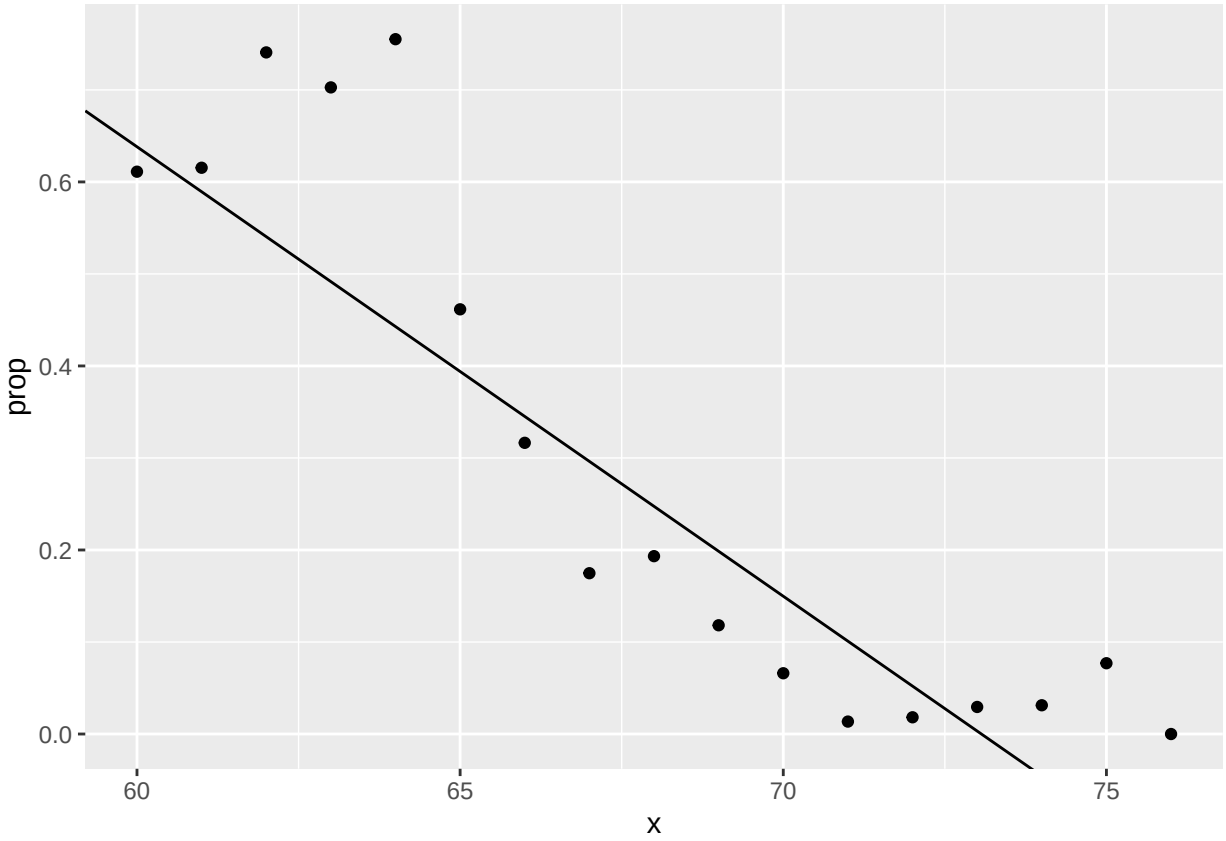
```r
lm_fit <- mutate(train_set, y = as.numeric(sex == "Female")) %>%
  lm(y ~ height, data = .)
```

```r
p_hat <- predict(lm_fit, test_set)
y_hat <- ifelse(p_hat > 0.5, "Female", "Male") %>% factor()
confusionMatrix(y_hat, test_set$sex)[["Accuracy"]]
```

```
## NULL
```

Generalized Linear Models

```r
heights %>%
  mutate(x = round(height)) %>%
  group_by(x) %>%
  filter(n() >= 10) %>%
  summarize(prop = mean(sex == "Female")) %>%
  ggplot(aes(x, prop)) +
  geom_point() +
  geom_abline(intercept = lm_fit$coef[1], slope = lm_fit$coef[2])
```




```r
range(p_hat)
```

```
## [1] -0.4694498  1.1265985
```

use Logistic regression

```r
glm_fit <- train_set %>%
  mutate(y = as.numeric(sex == "Female")) %>%
```

```
  glm(y ~ height, data=., family = "binomial")
```

```
p_hat_logit <- predict(glm_fit, newdata = test_set, type = "response")
```

```
y_hat_logit <- ifelse(p_hat_logit > 0.5, "Female", "Male") %>% factor
confusionMatrix(y_hat_logit, test_set$sex)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Female Male
##     Female     46   21
##     Male       73  385
##
##                Accuracy : 0.821
##                  95% CI : (0.7854, 0.8528)
##     No Information Rate : 0.7733
##     P-Value [Acc > NIR] : 0.004489
##
##                   Kappa : 0.396
##
##  Mcnemar's Test P-Value : 1.439e-07
##
##             Sensitivity : 0.38655
##             Specificity : 0.94828
##          Pos Pred Value : 0.68657
##          Neg Pred Value : 0.84061
##              Prevalence : 0.22667
##          Detection Rate : 0.08762
##    Detection Prevalence : 0.12762
##       Balanced Accuracy : 0.66742
##
##        'Positive' Class : Female
##
```

```
confusionMatrix(y_hat_logit, test_set$sex)[["Accuracy"]]
```

```
## NULL
```

Class note Example

```
head(Default)
```

```
##   default student   balance    income
## 1      No      No  729.5265 44361.625
## 2      No     Yes  817.1804 12106.135
## 3      No      No 1073.5492 31767.139
## 4      No      No  529.2506 35704.494
## 5      No      No  785.6559 38463.496
## 6      No     Yes  919.5885  7491.559
```

```
glm(default~balance, family = "binomial", data = Default)
```

```
##
## Call:  glm(formula = default ~ balance, family = "binomial", data = Default)
```

```
##
## Coefficients:
## (Intercept)      balance
##  -10.651331     0.005499
##
## Degrees of Freedom: 9999 Total (i.e. Null);  9998 Residual
## Null Deviance:       2921
## Residual Deviance: 1596  AIC: 1600
```

```
glm(default~student, family = "binomial", data = Default)
```

```
##
## Call:  glm(formula = default ~ student, family = "binomial", data = Default)
##
## Coefficients:
## (Intercept)   studentYes
##     -3.5041       0.4049
##
## Degrees of Freedom: 9999 Total (i.e. Null);  9998 Residual
## Null Deviance:       2921
## Residual Deviance: 2909  AIC: 2913
```

```
glm(default~balance + income, family = "binomial", data = Default)
```

```
##
## Call:  glm(formula = default ~ balance + income, family = "binomial",
##     data = Default)
##
## Coefficients:
## (Intercept)      balance       income
##  -1.154e+01    5.647e-03    2.081e-05
##
## Degrees of Freedom: 9999 Total (i.e. Null);  9997 Residual
## Null Deviance:       2921
## Residual Deviance: 1579  AIC: 1585
```

```
glm(default~balance + student + income, family = "binomial", data = Default)
```

```
##
## Call:  glm(formula = default ~ balance + student + income, family = "binomial",
##     data = Default)
##
## Coefficients:
## (Intercept)      balance   studentYes       income
##  -1.087e+01    5.737e-03   -6.468e-01    3.033e-06
##
## Degrees of Freedom: 9999 Total (i.e. Null);  9996 Residual
## Null Deviance:       2921
## Residual Deviance: 1572  AIC: 1580
```

Example

https://daviddalpiaz.github.io/r4sl/logistic-regression.html

```
library(ISLR)
library(tibble)
```

```
as_tibble(Default)
```

```
## # A tibble: 10,000 x 4
##    default student balance income
##    <fct>   <fct>     <dbl>  <dbl>
##  1 No      No         730. 44362.
##  2 No      Yes        817. 12106.
##  3 No      No        1074. 31767.
##  4 No      No         529. 35704.
##  5 No      No         786. 38463.
##  6 No      Yes        920.  7492.
##  7 No      No         826. 24905.
##  8 No      Yes        809. 17600.
##  9 No      No        1161. 37469.
## 10 No      No           0  29275.
## # ... with 9,990 more rows
```

```
set.seed(42)
default_idx = sample(nrow(Default), 5000)
default_trn = Default[default_idx, ]
default_tst = Default[-default_idx, ]
```

Before moving on to logistic regression, why not plain, old, linear regression?

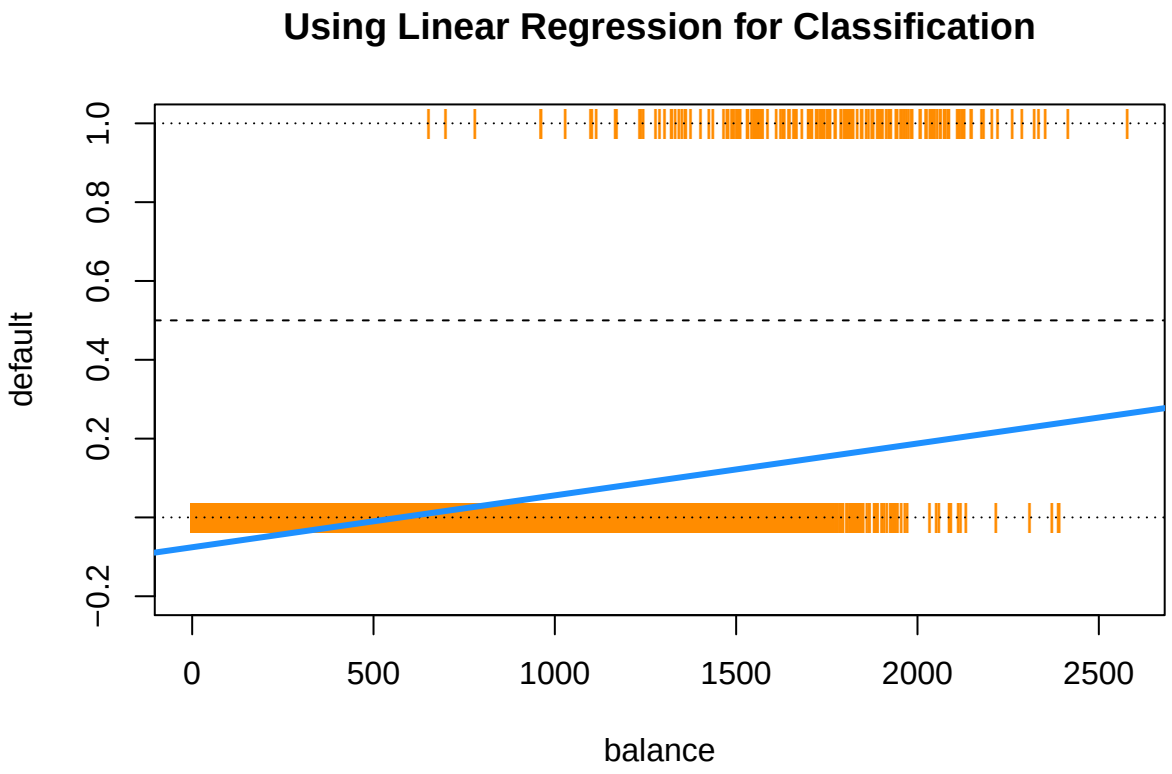
```
default_trn_lm = default_trn
default_tst_lm = default_tst
```

Since linear regression expects a numeric response variable, we coerce the response to be numeric. (Notice that we also shift the results, as we require 0 and 1, not 1 and 2.) Notice we have also copied the dataset so that we can return the original data with factors later.

```
default_trn_lm$default = as.numeric(default_trn_lm$default) - 1
default_tst_lm$default = as.numeric(default_tst_lm$default) - 1
```

```
model_lm = lm(default ~ balance, data = default_trn_lm)
```

```
plot(default ~ balance, data = default_trn_lm,
     col = "darkorange", pch = "|", ylim = c(-0.2, 1),
     main = "Using Linear Regression for Classification")
abline(h = 0, lty = 3)
abline(h = 1, lty = 3)
abline(h = 0.5, lty = 2)
abline(model_lm, lwd = 3, col = "dodgerblue")
```



predicted probabilities are below 0.5.

```
all(predict(model_lm) < 0.5)
```

```
## [1] TRUE
```

predicted probabilities less than 0.

```
any(predict(model_lm) < 0)
```

```
## [1] TRUE
```

Logistc Regression!!!

```
model_glm = glm(default ~ balance, data = default_trn, family = "binomial")
```

```
coef(model_glm)
```

```
##   (Intercept)       balance
## -10.493158288   0.005424994
```

understand is how the predict() function works with glm(). So, let's look at some predictions.

```
head(predict(model_glm))
```

```
##      2369      5273      9290      1252      8826       356
## -5.376670 -4.875653 -5.018746 -4.007664 -6.538414 -6.601582
```

By default, predict.glm() uses type = "link".

```
head(predict(model_glm, type = "link"))
```

```
##      2369      5273      9290      1252      8826       356
## -5.376670 -4.875653 -5.018746 -4.007664 -6.538414 -6.601582
```

these are returning the log odds, not predicted probabilities.

we need to use type = "response"

```
head(predict(model_glm, type = "response"))
```

```
##        2369        5273        9290        1252        8826         356
## 0.004601914 0.007572331 0.006569370 0.017851333 0.001444691 0.001356375
```

Note that these are probabilities, not classifications. To obtain classifications, we will need to compare to the correct cutoff value with an ifelse() statement.

```
model_glm_pred = ifelse(predict(model_glm, type = "link") > 0, "Yes", "No")
# model_glm_pred = ifelse(predict(model_glm, type = "response") > 0.5, "Yes", "No")
```

Once we have classifications, we can calculate metrics such as the trainging classification error rate.

```
calc_class_err = function(actual, predicted) {
  mean(actual != predicted)
}
```

```
calc_class_err(actual = default_trn$default, predicted = model_glm_pred)
```

```
## [1] 0.0284
```

As we saw previously, the table() and confusionMatrix() functions can be used to quickly obtain many more metrics.

```
train_tab = table(predicted = model_glm_pred, actual = default_trn$default)
library(caret)
train_con_mat = confusionMatrix(train_tab, positive = "Yes")
c(train_con_mat$overall["Accuracy"],
  train_con_mat$byClass["Sensitivity"],
  train_con_mat$byClass["Specificity"])
```

```
##    Accuracy Sensitivity Specificity
##   0.9716000   0.2941176   0.9954451
```

We could also write a custom function for the error for use with trained logist regression models.

```
get_logistic_error = function(mod, data, res = "y", pos = 1, neg = 0, cut = 0.5) {
  probs = predict(mod, newdata = data, type = "response")
  preds = ifelse(probs > cut, pos, neg)
  calc_class_err(actual = data[, res], predicted = preds)
}
```

This function will be useful later when calculating train and test errors for several models at the same time.

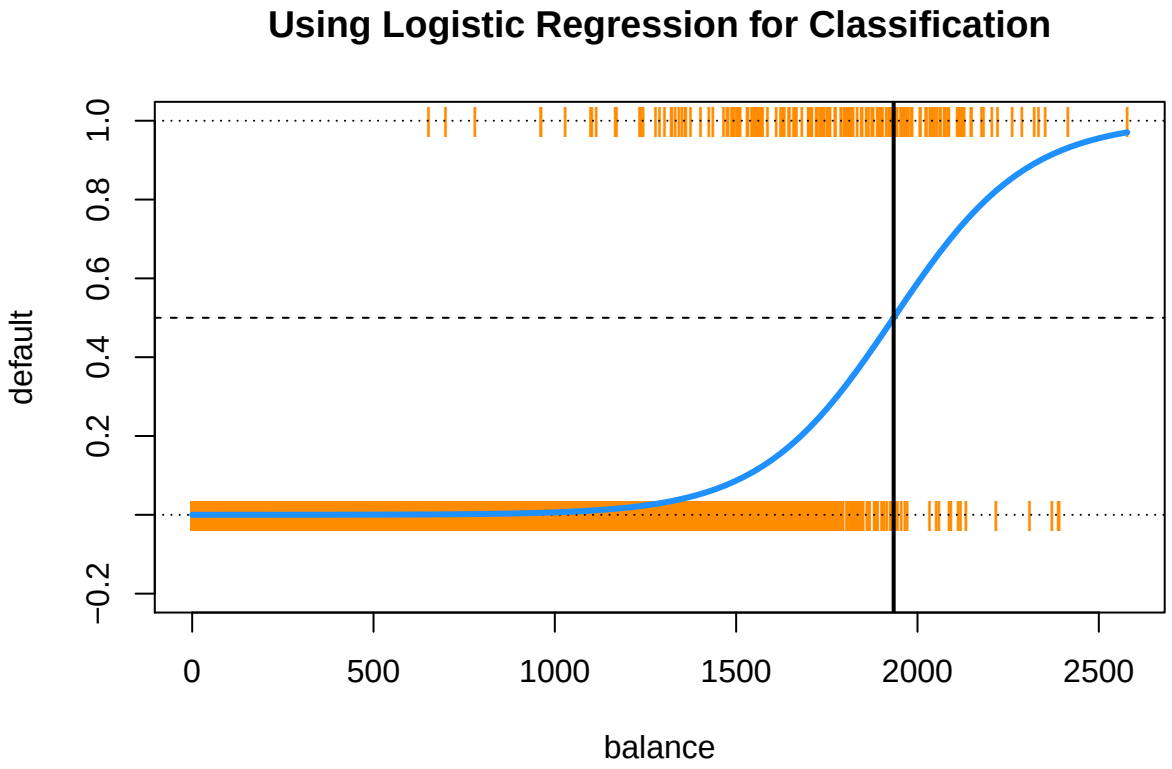
```
get_logistic_error(model_glm, data = default_trn,
                   res = "default", pos = "Yes", neg = "No", cut = 0.5)
```

```
## [1] 0.0284
```

To see how much better logistic regression is for this task, we create the same plot we used for linear regression.

```
plot(default ~ balance, data = default_trn_lm,
     col = "darkorange", pch = "|", ylim = c(-0.2, 1),
     main = "Using Logistic Regression for Classification")
abline(h = 0, lty = 3)
abline(h = 1, lty = 3)
abline(h = 0.5, lty = 2)
curve(predict(model_glm, data.frame(balance = x), type = "response"),
```

```
      add = TRUE, lwd = 3, col = "dodgerblue")
abline(v = -coef(model_glm)[1] / coef(model_glm)[2], lwd = 2)
```



solid vertical black line represents the decision boundary, the balance that obtains a predicted probability of 0.5. In this case balance = 1934.2247145.

Using the usual formula syntax, it is easy to add or remove complexity from logistic regressions.

```
model_1 = glm(default ~ 1, data = default_trn, family = "binomial")
model_2 = glm(default ~ ., data = default_trn, family = "binomial")
model_3 = glm(default ~ . ^ 2 + I(balance ^ 2),
              data = default_trn, family = "binomial")
```

Note that, using polynomial transformations of predictors will allow a linear model to have non-linear decision boundaries.

```
model_list = list(model_1, model_2, model_3)
train_errors = sapply(model_list, get_logistic_error, data = default_trn,
                      res = "default", pos = "Yes", neg = "No", cut = 0.5)
test_errors  = sapply(model_list, get_logistic_error, data = default_tst,
                      res = "default", pos = "Yes", neg = "No", cut = 0.5)
```

see the misclassification error rates for each model. The train decreases, and the test decreases, until it starts to increases. Everything we learned about the bias-variance tradeoff for regression also applies here.

```
diff(train_errors)
```

```
## [1] -0.0066  0.0000
```

```
diff(test_errors)
```

```
## [1] -0.0068  0.0006
```

Example: MNIST

Visualize the digits:

```
#library(keras)
#library(tensorflow)
# enable eager execution
# the argument device_policy is needed only when using a GPU
#tfe_enable_eager_execution(device_policy = "silent")
#tf$executing_eagerly()

#mnist <- dataset_mnist()
#train_images <- mnist$train$x
#dim(train_images)
#digit <- train_images[50, 1:28, 1:28]
#par(pty="s") # for keeping the aspect ratio 1:1
#image(t(digit), col = gray.colors(256), axes = FALSE)
```

```
mnist <- read_mnist()
length(mnist$train$labels) #60000
```

```
## [1] 60000
```

```
length(mnist$test$labels) #10000
```

```
## [1] 10000
```

is it a 2 or a 7?

```
#library(tidyverse)
#library(dslabs)
data("mnist_27")
mnist_27$train %>% ggplot(aes(x_1, x_2, color = y)) + geom_point()
```


```
library(dslabs)
data("mnist_27")
fit_glm <- glm(y ~ x_1 + x_2, data=mnist_27$train, family = "binomial")
p_hat_glm <- predict(fit_glm, mnist_27$test, type="response")
y_hat_glm <- factor(ifelse(p_hat_glm > 0.5, 7, 2))
confusionMatrix(y_hat_glm, mnist_27$test$y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  2  7
##          2 82 26
##          7 24 68
##
##                Accuracy : 0.75
##                  95% CI : (0.684, 0.8084)
##     No Information Rate : 0.53
##     P-Value [Acc > NIR] : 1.266e-10
##
##                   Kappa : 0.4976
##
##  Mcnemar's Test P-Value : 0.8875
##
##             Sensitivity : 0.7736
##             Specificity : 0.7234
##          Pos Pred Value : 0.7593
##          Neg Pred Value : 0.7391
```

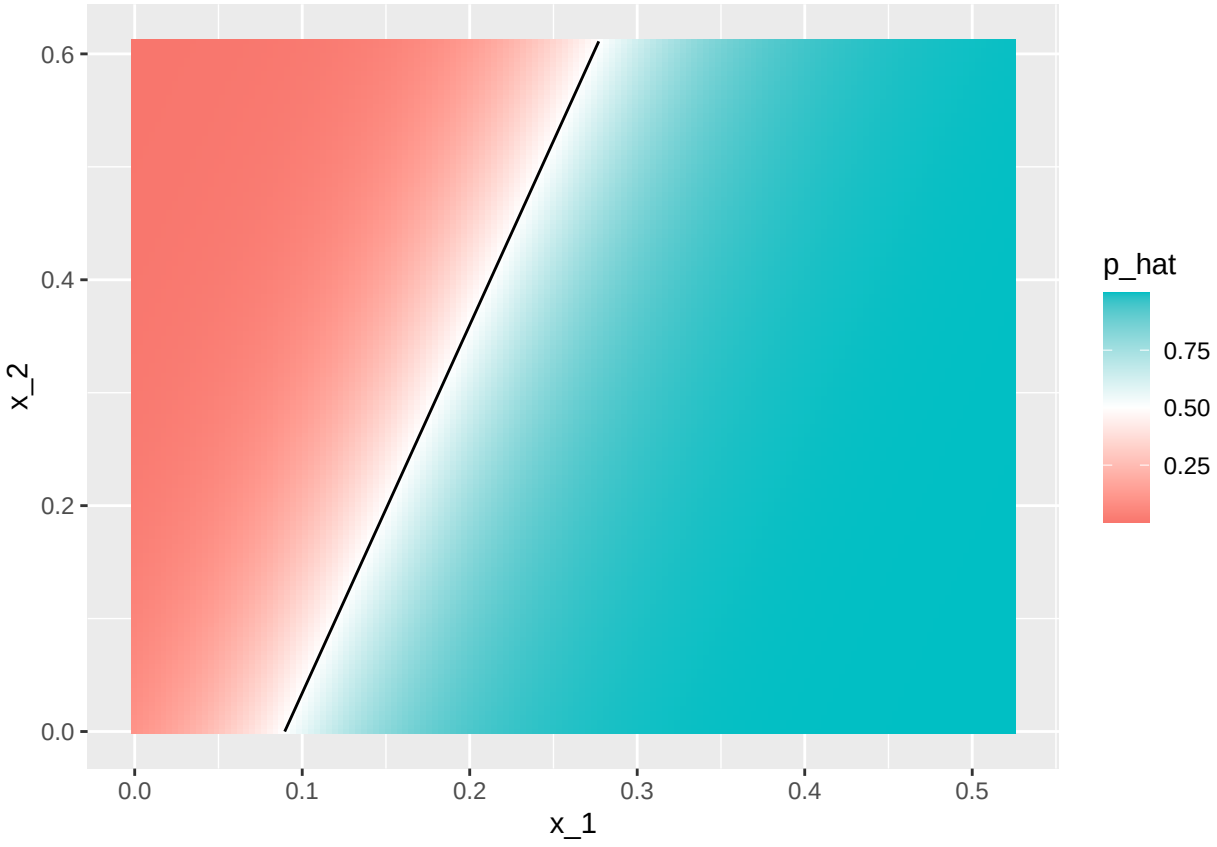
```
##              Prevalence : 0.5300
##          Detection Rate : 0.4100
##    Detection Prevalence : 0.5400
##       Balanced Accuracy : 0.7485
##
##        'Positive' Class : 2
##
```

```
confusionMatrix(y_hat_glm, mnist_27$test$y)$overall["Accuracy"]
```

```
## Accuracy
##     0.75
```

```
#> Accuracy
#>     0.75
```

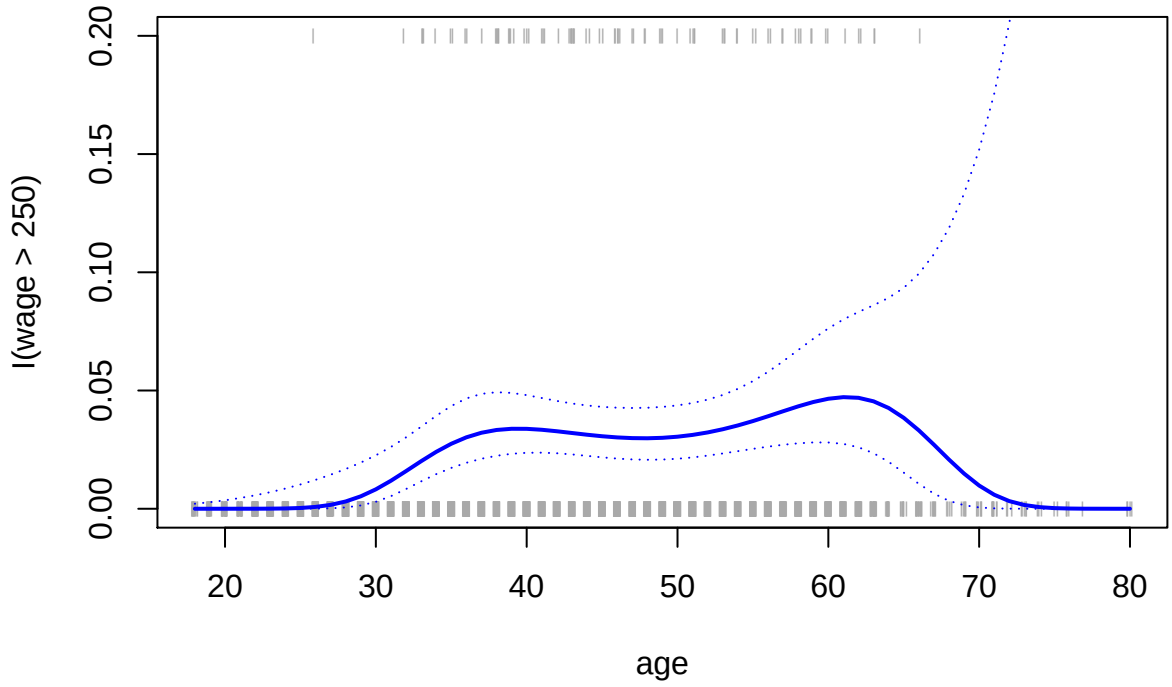
```
p_hat <- predict(fit_glm, newdata = mnist_27$true_p, type = "response")
mnist_27$true_p %>% mutate(p_hat = p_hat) %>%
  ggplot(aes(x_1, x_2,  z=p_hat, fill=p_hat)) +
  geom_raster() +
  scale_fill_gradientn(colors=c("#F8766D","white","#00BFC4")) +
  stat_contour(breaks=c(0.5), color="black")
```



Splines and gams

```
library(gam)
library(splines)
```

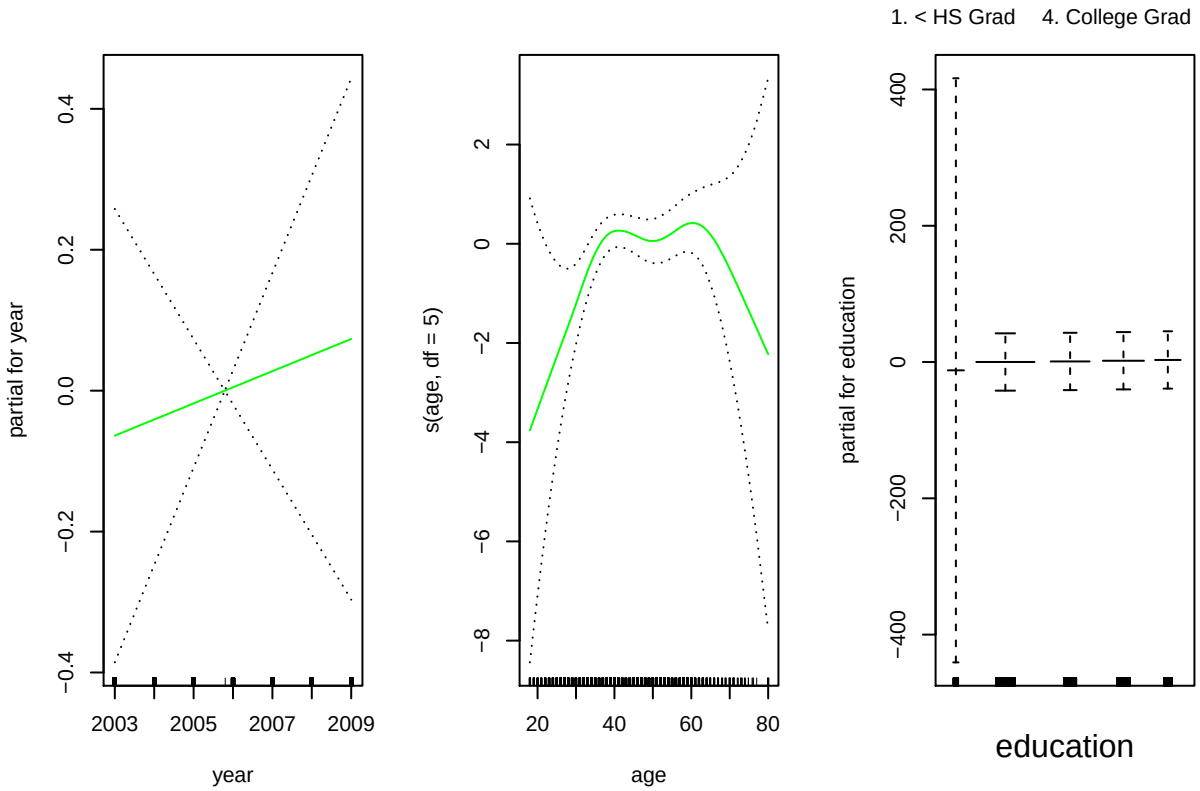
```
agelims=range(Wage$age)
age.grid=seq(from=agelims[1],to=agelims[2])

fit=glm(I(wage>250)~poly(age,4),data=Wage,family=binomial)
preds=predict(fit,newdata=list(age=age.grid),se=T)
pfit=exp(preds$fit)/(1+exp(preds$fit))
se.bands.logit = cbind(preds$fit+2*preds$se.fit, preds$fit-2*preds$se.fit)
se.bands = exp(se.bands.logit)/(1+exp(se.bands.logit))
preds=predict(fit,newdata=list(age=age.grid),type="response",se=T)


attach(Wage)
plot(age,I(wage>250),xlim=agelims,type="n",ylim=c(0,.2))
points(jitter(age), I((wage>250)/5),cex=.5,pch="|",col="darkgrey")
lines(age.grid,pfit,lwd=2, col="blue")
matlines(age.grid,se.bands,lwd=1,col="blue",lty=3)
```




```
gam.lr=gam(I(wage>250)~year+s(age,df=5)+education,family=binomial,data=Wage)
```

```
## Warning in model.matrix.default(mt, mf, contrasts): non-list contrasts argument
## ignored
```

```
par(mfrow=c(1,3))
plot(gam.lr,se=T,col="green")
```



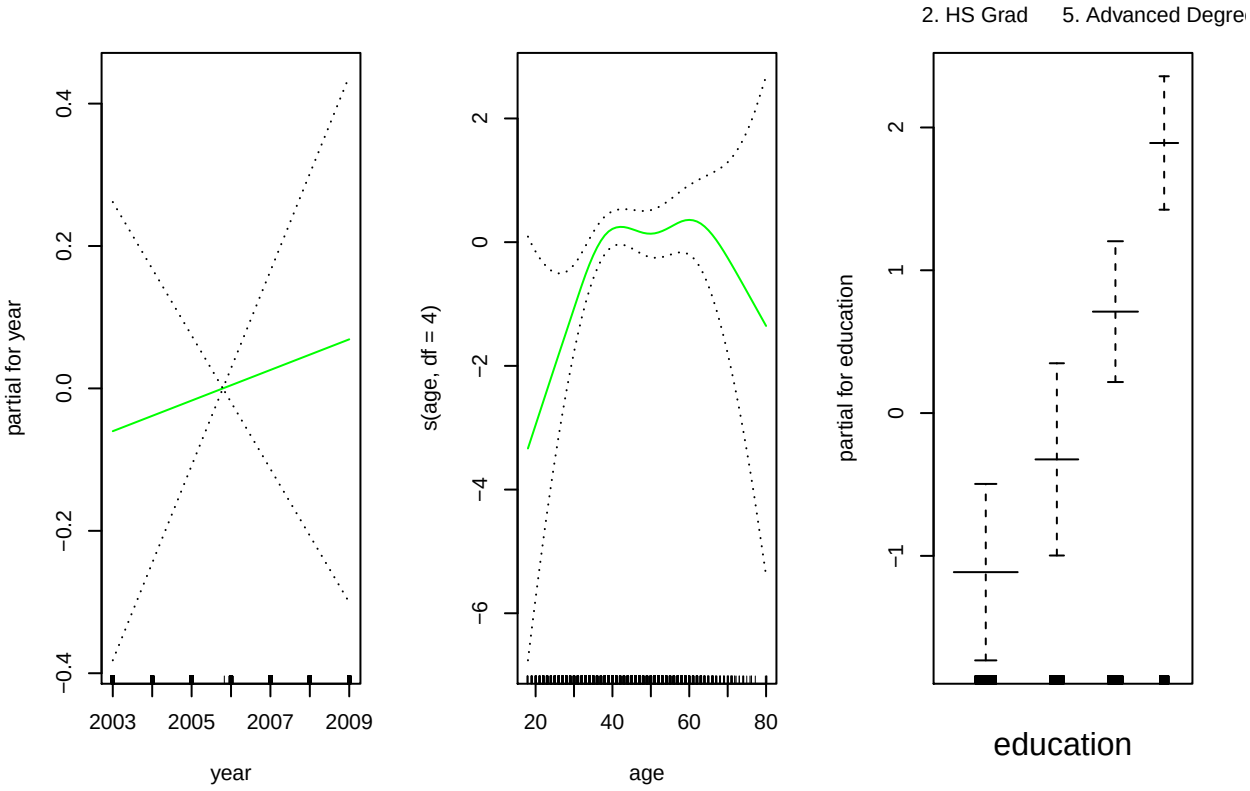

```
table(education,I(wage>250))
```

```
##
## education             FALSE TRUE
##   1. < HS Grad          268    0
##   2. HS Grad            966    5
##   3. Some College       643    7
##   4. College Grad       663   22
##   5. Advanced Degree    381   45
```

```
gam.lr.s=gam(I(wage>250)~year+s(age,df=4)+education,family=binomial,data=Wage,subset=(education!="1. < H
```

```
## Warning in model.matrix.default(mt, mf, contrasts): non-list contrasts argument
## ignored
```

```
plot(gam.lr.s,se=T,col="green")
```



Multinomial logistic Reression

MULTINOMIAL LOGISTIC REGRESSION | R DATA ANALYSIS EXAMPLES https://stats.idre.ucla.edu/r/dae/multinomial-logistic-regression/

From the text book

```
library(ISLR)
names(Smarket)
```

```
## [1] "Year"      "Lag1"      "Lag2"      "Lag3"      "Lag4"      "Lag5"
## [7] "Volume"    "Today"     "Direction"
```

```
dim(Smarket)
```

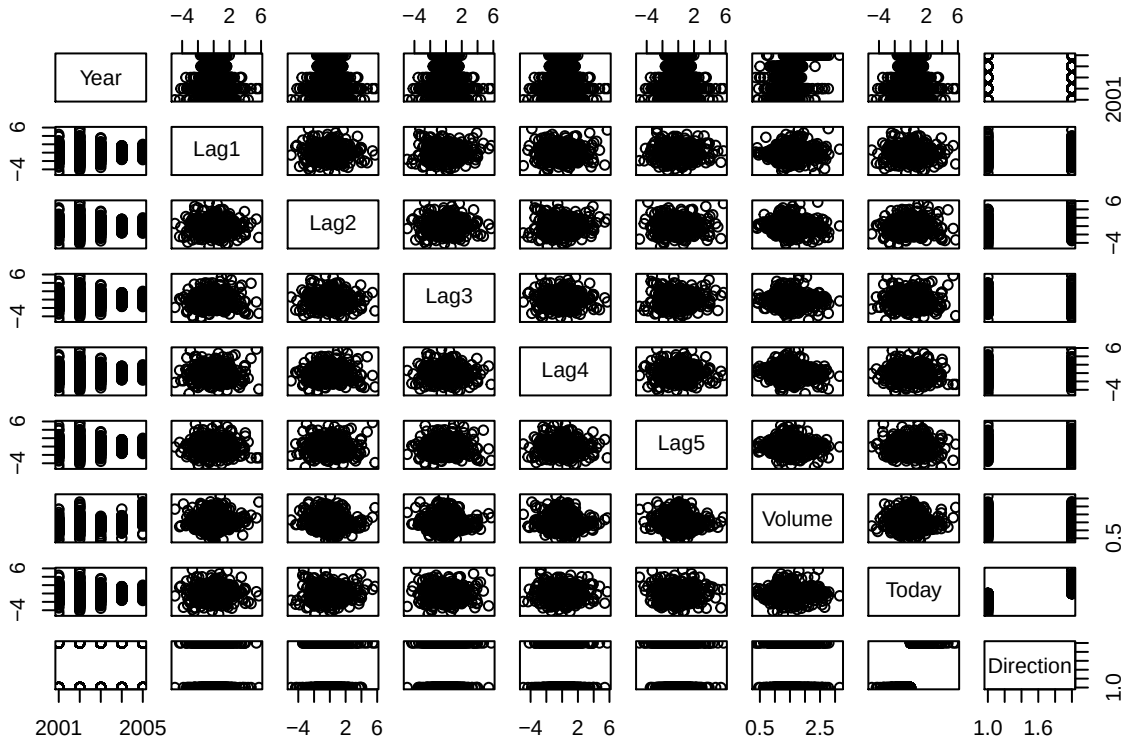
```
## [1] 1250    9
```

```
summary(Smarket)
```

```
##       Year           Lag1                Lag2                Lag3
##  Min.   :2001   Min.   :-4.922000   Min.   :-4.922000   Min.   :-4.922000
##  1st Qu.:2002   1st Qu.:-0.639500   1st Qu.:-0.639500   1st Qu.:-0.640000
##  Median :2003   Median : 0.039000   Median : 0.039000   Median : 0.038500
##  Mean   :2003   Mean   : 0.003834   Mean   : 0.003919   Mean   : 0.001716
##  3rd Qu.:2004   3rd Qu.: 0.596750   3rd Qu.: 0.596750   3rd Qu.: 0.596750
##  Max.   :2005   Max.   : 5.733000   Max.   : 5.733000   Max.   : 5.733000
##       Lag4                Lag5              Volume           Today
##  Min.   :-4.922000   Min.   :-4.92200   Min.   :0.3561   Min.   :-4.922000
##  1st Qu.:-0.640000   1st Qu.:-0.64000   1st Qu.:1.2574   1st Qu.:-0.639500
```

```
##  Median : 0.038500   Median : 0.03850   Median :1.4229   Median : 0.038500
##  Mean   : 0.001636   Mean   : 0.00561   Mean   :1.4783   Mean   : 0.003138
##  3rd Qu.: 0.596750   3rd Qu.: 0.59700   3rd Qu.:1.6417   3rd Qu.: 0.596750
##  Max.   : 5.733000   Max.   : 5.73300   Max.   :3.1525   Max.   : 5.733000
##  Direction
##  Down:602
##  Up  :648
##
##
##
##
```

```
pairs(Smarket)
```

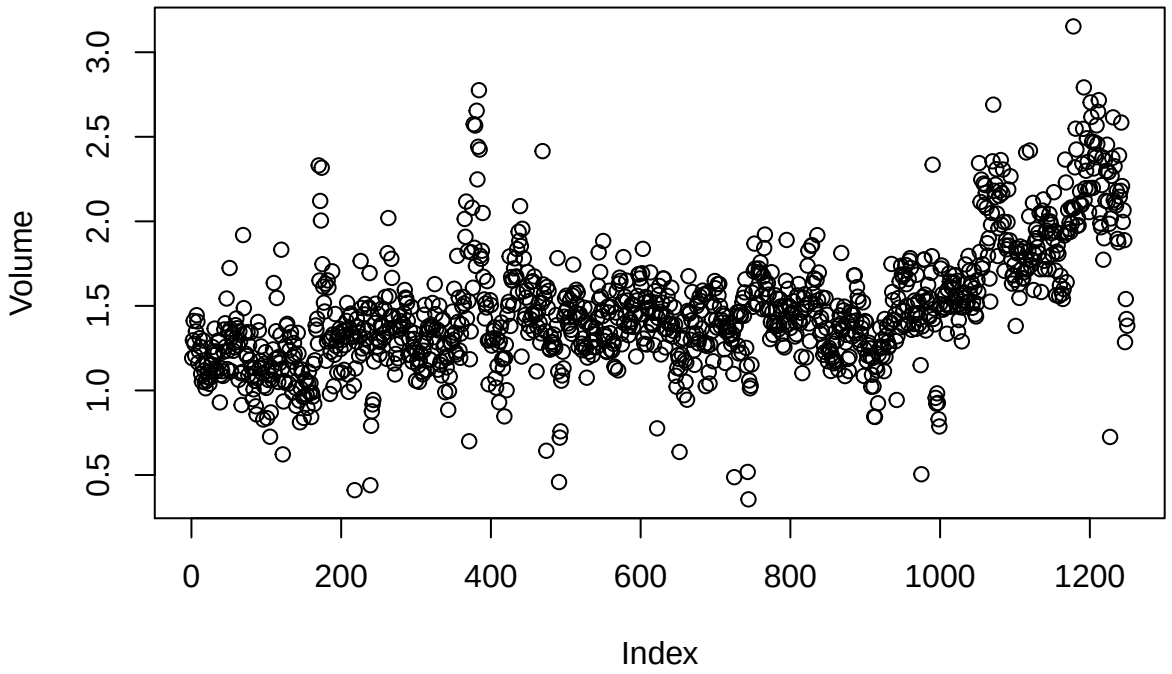



```
#cor(Smarket)
cor(Smarket[,-9])
```

```
##              Year         Lag1         Lag2         Lag3         Lag4
## Year   1.00000000  0.029699649  0.030596422  0.033194581  0.035688718
## Lag1   0.02969965  1.000000000 -0.026294328 -0.010803402 -0.002985911
## Lag2   0.03059642 -0.026294328  1.000000000 -0.025896670 -0.010853533
## Lag3   0.03319458 -0.010803402 -0.025896670  1.000000000 -0.024051036
## Lag4   0.03568872 -0.002985911 -0.010853533 -0.024051036  1.000000000
## Lag5   0.02978799 -0.005674606 -0.003557949 -0.018808338 -0.027083641
## Volume 0.53900647  0.040909908 -0.043383215 -0.041823686 -0.048414246
## Today  0.03009523 -0.026155045 -0.010250033 -0.002447647 -0.006899527
##                Lag5      Volume        Today
## Year    0.029787995  0.53900647  0.030095229
## Lag1   -0.005674606  0.04090991 -0.026155045
## Lag2   -0.003557949 -0.04338321 -0.010250033
## Lag3   -0.018808338 -0.04182369 -0.002447647
```

```
## Lag4   -0.027083641 -0.04841425 -0.006899527
## Lag5    1.000000000 -0.02200231 -0.034860083
## Volume -0.022002315  1.00000000  0.014591823
## Today  -0.034860083  0.01459182  1.000000000
```

```
attach(Smarket)
plot(Volume)
```

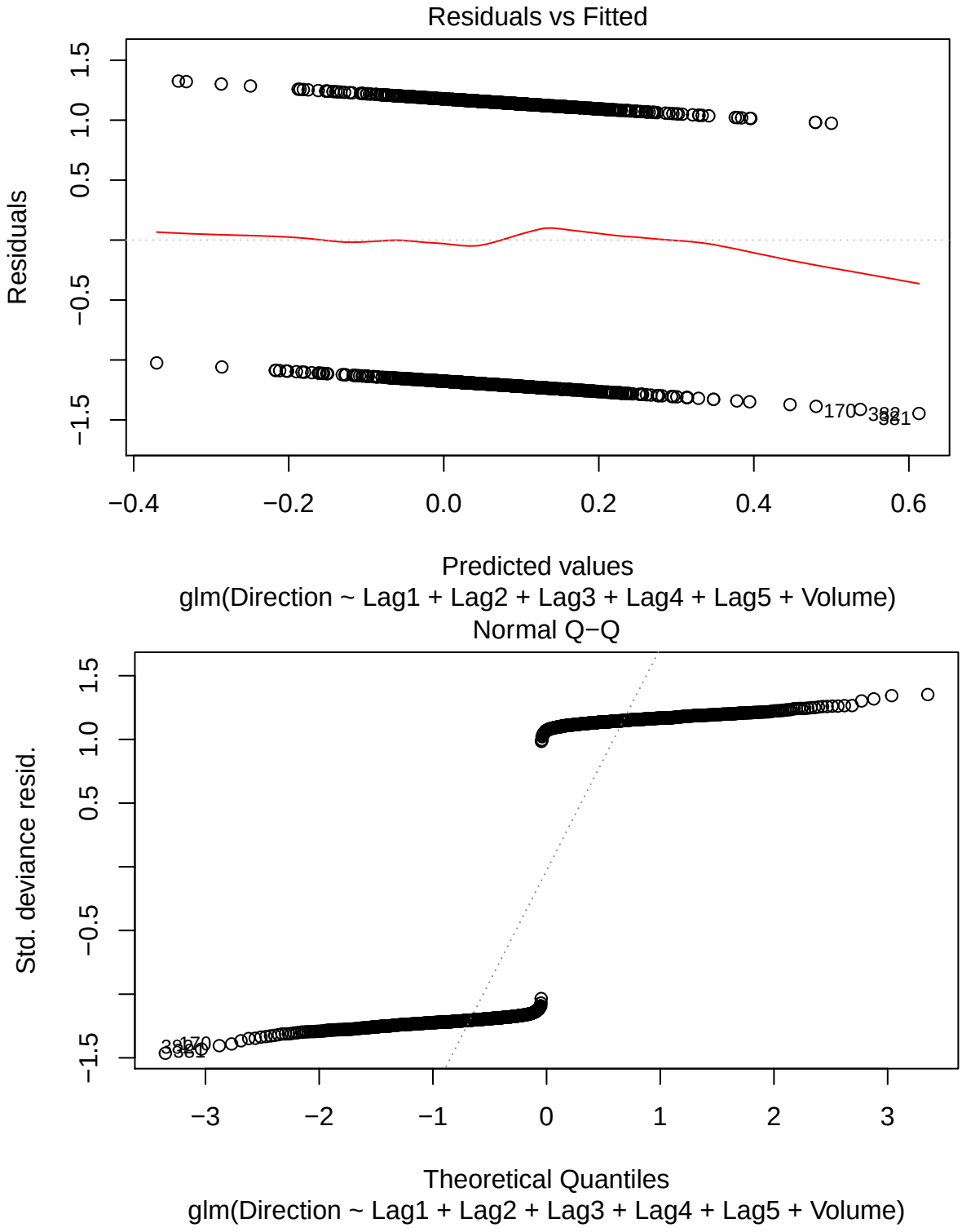


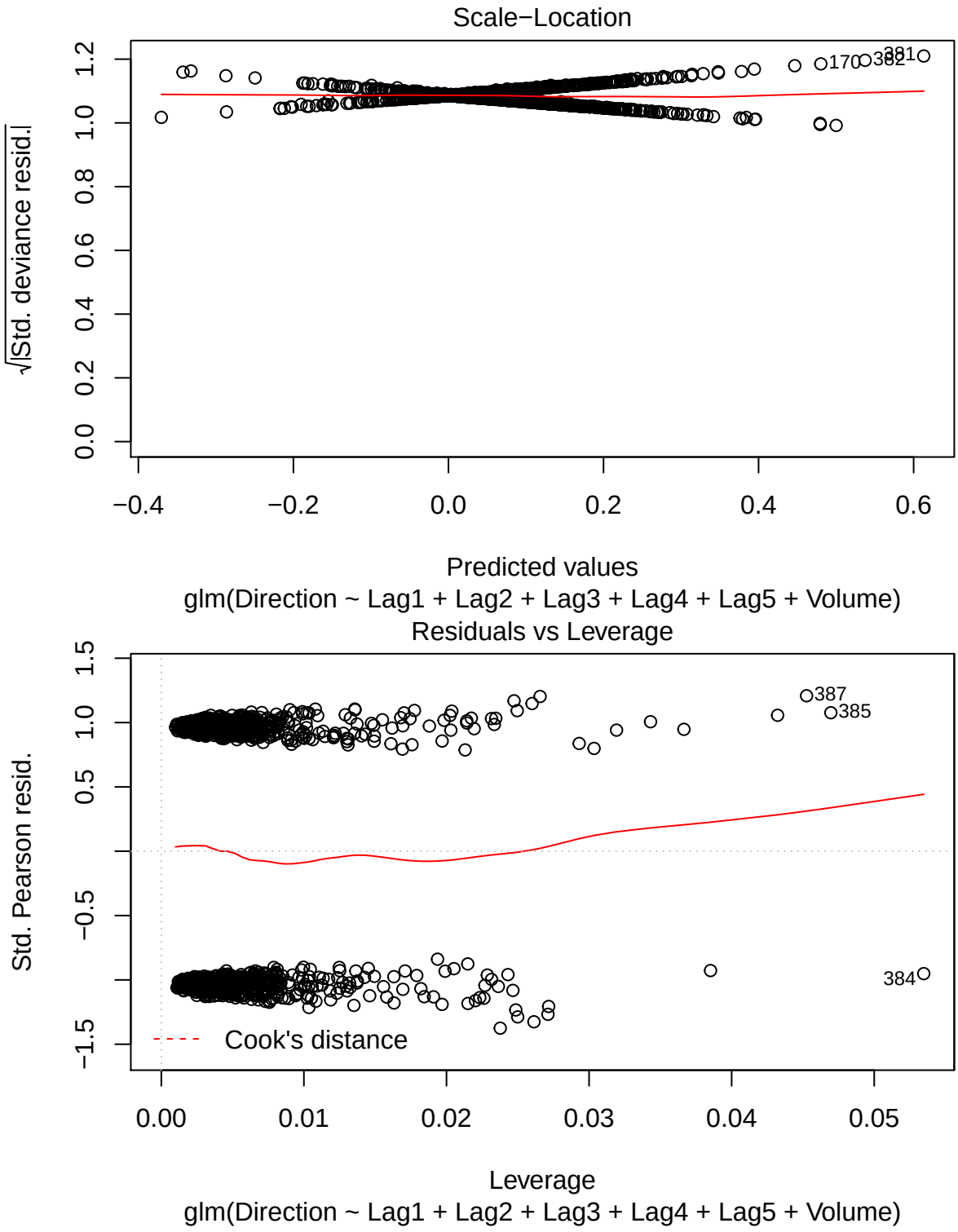

```
glm.fit=glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume,data=Smarket,family=binomial)
summary(glm.fit)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial, data = Smarket)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.446  -1.203   1.065   1.145   1.326
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.126000   0.240736  -0.523    0.601
## Lag1        -0.073074   0.050167  -1.457    0.145
## Lag2        -0.042301   0.050086  -0.845    0.398
## Lag3         0.011085   0.049939   0.222    0.824
## Lag4         0.009359   0.049974   0.187    0.851
## Lag5         0.010313   0.049511   0.208    0.835
## Volume       0.135441   0.158360   0.855    0.392
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1731.2  on 1249  degrees of freedom
## Residual deviance: 1727.6  on 1243  degrees of freedom
```

```
## AIC: 1741.6
##
## Number of Fisher Scoring iterations: 3
```

```
plot(glm.fit)
```






```
coef(glm.fit)
```

```
##  (Intercept)         Lag1         Lag2         Lag3         Lag4         Lag5
## -0.126000257 -0.073073746 -0.042301344  0.011085108  0.009358938  0.010313068
##       Volume
##  0.135440659
```

```
summary(glm.fit)$coef
```

```
##                 Estimate Std. Error    z value  Pr(>|z|)
## (Intercept) -0.126000257 0.24073574 -0.5233966 0.6006983
## Lag1        -0.073073746 0.05016739 -1.4565986 0.1452272
## Lag2        -0.042301344 0.05008605 -0.8445733 0.3983491
## Lag3         0.011085108 0.04993854  0.2219750 0.8243333
## Lag4         0.009358938 0.04997413  0.1872757 0.8514445
## Lag5         0.010313068 0.04951146  0.2082966 0.8349974
## Volume       0.135440659 0.15835970  0.8552723 0.3924004
```

```
glm.probs=predict(glm.fit,type="response")
glm.probs[1:10]
```

```
##         1         2         3         4         5         6         7         8
## 0.5070841 0.4814679 0.4811388 0.5152224 0.5107812 0.5069565 0.4926509 0.5092292
##         9        10
## 0.5176135 0.4888378
```

```
contrasts(Direction)
```

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

```
glm.pred=rep("Down",1250)
glm.pred[glm.probs>.5]="Up"
table(glm.pred,Direction)
```

```
##         Direction
## glm.pred Down  Up
##     Down  145 141
##     Up    457 507
```

KNN

```
knn_fit <- knn3(y ~ ., data = mnist_27$train)
knn_fit <- knn3(y ~ ., data = mnist_27$train, k = 5)
y_hat_knn <- predict(knn_fit, mnist_27$test, type = "class")
confusionMatrix(y_hat_knn, mnist_27$test$y)$overall["Accuracy"]
```

```
## Accuracy
##    0.815
```

```
#compare
y_hat_knn <- predict(knn_fit, mnist_27$train, type = "class")
confusionMatrix(y_hat_knn, mnist_27$train$y)$overall["Accuracy"]
```

```
## Accuracy
##   0.8825
```

```
#over-training
```

ROC curve

Example 1: https://rpubs.com/dvorakt/255527

Example 2

Let's go to our Default data example

```
model_glm = glm(default ~ balance, data = default_trn, family = "binomial")
```

We write a function which allows use to make predictions based on different probability cutoffs.

```
get_logistic_pred = function(mod, data, res = "y", pos = 1, neg = 0, cut = 0.5) {
  probs = predict(mod, newdata = data, type = "response")
  ifelse(probs > cut, pos, neg)
}
```

Let's use this to obtain predictions using a low, medium, and high cutoff. (0.1, 0.5, and 0.9)

```
test_pred_10 = get_logistic_pred(model_glm, data = default_tst, res = "default",
                                 pos = "Yes", neg = "No", cut = 0.1)
test_pred_50 = get_logistic_pred(model_glm, data = default_tst, res = "default",
                                 pos = "Yes", neg = "No", cut = 0.5)
test_pred_90 = get_logistic_pred(model_glm, data = default_tst, res = "default",
                                 pos = "Yes", neg = "No", cut = 0.9)
```

Now we evaluate accuracy, sensitivity, and specificity for these classifiers.

```
test_tab_10 = table(predicted = test_pred_10, actual = default_tst$default)
test_tab_50 = table(predicted = test_pred_50, actual = default_tst$default)
test_tab_90 = table(predicted = test_pred_90, actual = default_tst$default)

test_con_mat_10 = confusionMatrix(test_tab_10, positive = "Yes")
test_con_mat_50 = confusionMatrix(test_tab_50, positive = "Yes")
test_con_mat_90 = confusionMatrix(test_tab_90, positive = "Yes")
```

```
metrics = rbind(

  c(test_con_mat_10$overall["Accuracy"],
    test_con_mat_10$byClass["Sensitivity"],
    test_con_mat_10$byClass["Specificity"]),

  c(test_con_mat_50$overall["Accuracy"],
    test_con_mat_50$byClass["Sensitivity"],
    test_con_mat_50$byClass["Specificity"]),

  c(test_con_mat_90$overall["Accuracy"],
    test_con_mat_90$byClass["Sensitivity"],
    test_con_mat_90$byClass["Specificity"])

)

rownames(metrics) = c("c = 0.10", "c = 0.50", "c = 0.90")
metrics
```

```
##          Accuracy Sensitivity Specificity
## c = 0.10   0.9328  0.71779141   0.9400455
## c = 0.50   0.9730  0.31288344   0.9952450
## c = 0.90   0.9688  0.04294479   1.0000000
```

We see then sensitivity decreases as the cutoff is increased. Conversely, specificity increases as the cutoff increases. This is useful if we are more interested in a particular error, instead of giving them equal weight.

Note that usually the best accuracy will be seen near c=0.50.

Instead of manually checking cutoffs, we can create an ROC curve (receiver operating characteristic curve) which will sweep through all possible cutoffs, and plot the sensitivity and specificity.

```
library(pROC)
```

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

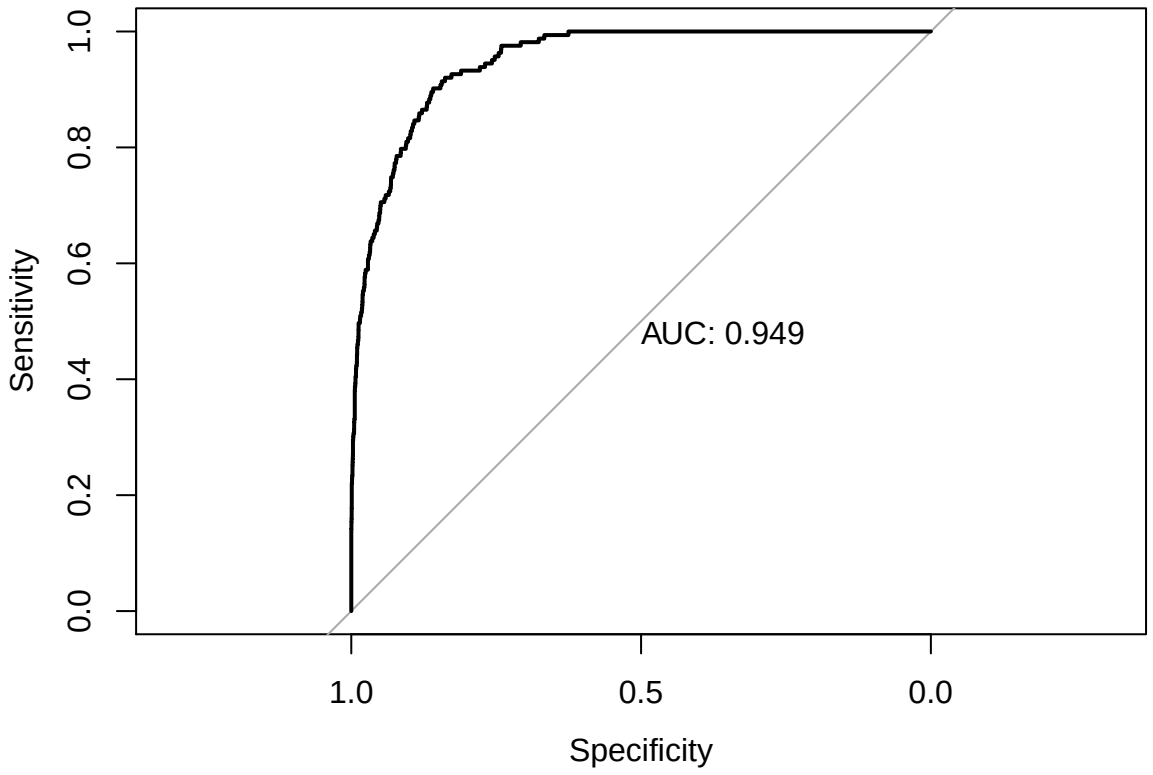
```
##
## Attaching package: 'pROC'
```

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

```
test_prob = predict(model_glm, newdata = default_tst, type = "response")
test_roc = roc(default_tst$default ~ test_prob, plot = TRUE, print.auc = TRUE)
```

```
## Setting levels: control = No, case = Yes
```

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




```
as.numeric(test_roc$auc)
```

```
## [1] 0.9492866
```

A good model will have a high AUC, that is as often as possible a high sensitivity and specificity.

Lab Assignment

Q1: Discuss this problem with your group members

```
require(ISLR)
require(MASS)
```

```
## Loading required package: MASS
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
require(pROC)
set.seed(3315)
```

MNIST Problems

In the following problems the goal is to make a classifier that can distinguish the digit 3 from the digit 5. You will have to use all training data and all test data for these two digits. Ignore the data for the other eight digits.

```
load("mnist_all.RData")
```

Extract training and test data for digits 3 and 5. Include an extra column z for training and test data that equals 1 if the digit is 5 and 0 if the digit is 3.

```
index.35.train <- train$y == 3 | train$y == 5
train.35 <- as.data.frame(train$x[index.35.train,])
train.35$z <- as.numeric(train$y[index.35.train] == 5)
index.35.test <- test$y == 3 | test$y == 5
test.35 <- as.data.frame(test$x[index.35.test,])
test.35$z <- as.numeric(test$y[index.35.test] == 5)
```

Xtra #25

Build a classifier that uses only one variable (pixel). This variable should have large variation. Give the summary of the model and write out the logistic regression equation that has been obtained. Determine the fraction of true positives on the test set if the fraction of false positives on the training set is kept to 0.1.

Solution.

Identify the 10 pixels with the largest variation and choose one of them as predictor.

```
train.var <- apply(train.35[,1:784],2,FUN = sd)
train.order = order(train.var, decreasing = T)
head(train.order, 10)
```

```
##  [1] 353 325 180 187 216 324 403 382 243 208
```

Let's choose variable **403**.

```
model.25 <- glm(z ~ V403, data = train.35, family = binomial)
summary(model.25)
```

```
##
## Call:
## glm(formula = z ~ V403, family = binomial, data = train.35)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.3337  -1.0306  -0.9499   1.1097   1.4234
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.5618542  0.0283669  -19.81   <2e-16 ***
## V403         0.0036162  0.0001719   21.04   <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: 15971  on 11551  degrees of freedom
## Residual deviance: 15518  on 11550  degrees of freedom
## AIC: 15522
##
## Number of Fisher Scoring iterations: 4
```

The model equation for the predicted probability $\hat{p}$ is

$$\hat{p} = \frac{e^{-.5618+0.003616 \cdot V324}}{1 + e^{-.5618+0.003616 \cdot V324}}$$

Q2: ISLR ch.4 #10abcd

This question should be answered using the Weekly data set, which is part of the ISLR package. This data is similar in nature to the Smarket data from this chapter's lab, except that it contains 1,089 weekly returns for 21 years, from the beginning of 1990 to the end of 2010.

(a) Produce some numerical and graphical summaries of the Weekly data. Do there appear to be any patterns?

(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. Do any of the predictors appear to be statistically significant? If so, which ones?

(c) Compute the confusion matrix and overall fraction of correct predictions. Explain what the confusion matrix is telling you about the types of mistakes made by logistic regression.

(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).