**CLASSIFICATION: LOGISTIC REGRESSION.**

## #1. Chap 4, exercise 1

Either take the first expression and solve for $(\beta_0 + \beta_1 x)$, or take the second expression and solve for $p(x)$.

Expression $p(x) = \dfrac{e^{\beta_0+\beta_1 x}}{1+e^{\beta_0+\beta_1 x}}$ is equivalent to $\left(1+e^{\beta_0+\beta_1 x}\right) p(x) = e^{\beta_0+\beta_1 x}$, then to

$$p(x) + p(x)e^{\beta_0+\beta_1 x} = e^{\beta_0+\beta_1 x},$$

$$p(x) = e^{\beta_0+\beta_1 x} - p(x)e^{\beta_0+\beta_1 x} = (1-p(x))e^{\beta_0+\beta_1 x},$$

$$e^{\beta_0+\beta_1 x} = \frac{p(x)}{1-p(x)}, \text{ and finally,}$$

$$\beta_0 + \beta_1 x = \ln\left(\frac{p(x)}{1-p(x)}\right).$$

## #2. Chap 4, exercise 9.

(a) Solving the equation p/(1-p) = 0.37 for p, we get p = 0.37/(1+0.37) = **0.27**.
On the average, 27% of people default on their credit card payments.

(b) The odds are 0.16/(1-0.16) = 4/21 or **0.19**.

## #3. Chap. 4, exercise 6.

(a) The estimated probability is exp( -6 + 0.05*40 + 1*3.5 ) / (1 + exp( -6 + 0.05*40 + 1*3.5 )) = **0.378**.

(b) Here we are essentially solving the equation

P = exp( -6 + 0.05x + 1*3.5 ) / (1 + exp( -6 + 0.05x + 1*3.5 )) = 0.5 in terms of x.

P = 0.5 implies logit = ln(0.5/0.5) = 0. According to the logistic model, this logit also equals
Logit(P) = -6 + 0.05x + 1*3.5 = -2.5 + 0.05x.

From -2.5 + 0.05x = 0, we get x = **50**. The student has to study for **50 hours** instead of 40, to have a 50% predicted chance of getting an A.

## #4. Titanic data

```
# Read the Titanic data (use the suitable path)
> T = read.csv("C:\\Users\\baron\\627StatisticalMachineLearning\\data\\Titanic.csv")
> T1 = na.omit(T)       # Removing rows with missing data
> names(T1)
[1] "last"     "first"    "gender"   "age"      "class"    "fare"     "embarked"
[8] "survived"

# Convert the response variable to 0 and 1 for logistic regression
> T1$survival = 1*(T1$survived=="yes");
# At the same time, convert variable class to a factor, because it shouldn't be numeric
> T1$class.cat = as.factor(T1$class);

# Split the data at random into training and testing subsamples
> n = dim(T1)[1]        # Sample size
> Z = sample(n,n/2)

# Use training data to fit the logistic regression models, with and without variable fare
>  lreg.full  =  glm(  survival  ~  gender+age+fare+embarked+class.cat,  family=binomial,
data=T1[Z,] )
>  lreg.reduced  =  glm(  survival  ~  gender+age+embarked+class.cat,  family=binomial,
data=T1[Z,] )

> # Prediction probabilities
> prob.full = predict( lreg.full, newdata=T1[-Z,], type="response" )
> prob.reduced = predict( lreg.reduced, newdata=T1[-Z,], type="response" )

> # Classification of the testing data
> Yhat.full = 1*(prob.full > 0.5)
> Yhat.reduced = 1*(prob.reduced > 0.5)

> # Prediction error rates
> mean( Yhat.full != T1$survival[-Z] )
[1] 0.1848739
> mean( Yhat.reduced != T1$survival[-Z] )
[1] 0.1792717
```

**# Reduced model, without variable fare, shows a lower prediction error rate. Based on this cross-validation, we conclude that removing variable "fare" led to a lower error rate, and therefore, a better model.** *Your results may be slightly different due to randomization.*

## #5. Chap. 4, exercise 13 (Weekly data project).

**(a) Get Weekly data and fit the logistic model predicting Direction of the market.**

```
> install.packages("ISLR2")
> library(ISLR)
> attach(Weekly)
> model = glm(Direction ~ Volume + Lag1 + Lag2 + Lag3 + Lag4 + Lag5, family="binomial")
> summary(model)
```

```
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.26686    0.08593   3.106   0.0019 **
Volume      -0.02274    0.03690  -0.616   0.5377
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
```

**Only Lag2 is *marginally significant*, which is the percentage return of the market 2 weeks ago. Other variables do not appear significant. Apparently, it is hard to predict the market behavior based on the previous 5 weeks without stochastic processes commonly used in financial modeling – financial time series, geometric Brownian motion, etc. Indeed, we can see from these data that the market trends on 5 consecutive weeks are very weakly correlated:**

```
> cor(Weekly[,2:6])
             Lag1        Lag2        Lag3        Lag4         Lag5
Lag1  1.000000000 -0.07485305  0.05863568 -0.07127388 -0.008183096
Lag2 -0.074853051  1.00000000 -0.07572091  0.05838153 -0.072499482
Lag3  0.058635682 -0.07572091  1.00000000 -0.07539587  0.060657175
Lag4 -0.071273876  0.05838153 -0.07539587  1.00000000 -0.075675027
Lag5 -0.008183096 -0.07249948  0.06065717 -0.07567503  1.000000000
```

**(b) Predict the market direction.**

```
> Probability = predict(model, type ="response")
> Predicted.Direction = rep("Down",length(Probability))
> Predicted.Direction[ Probability > 0.5 ] = "Up"

# Here is the confusion matrix
> table( Direction, Predicted.Direction )

         Predicted.Direction
Direction Down  Up
     Down   54 430
     Up     48 557

> mean( Direction == Predicted.Direction )
[1] 0.5610652
> mean(Predicted.Direction == "Up")
[1] 0.9063361
> mean(Direction == "Up")
[1] 0.5555556
```

**We correctly predicted 54 weeks when market went down and 557 weeks when market went up, and overall, our correct classification rate is 56% and the error rate is 44%, which is only a little better than a coin toss. Our model classification rule is too *optimistic*, it predicts positive market 90.6% of the time whereas it only happens 55.6% of the time.**

**(c) Define training and testing data as they are defined in this exercise. Fit a logistic model with Lag2 as the only predictor.**

```
> Weekly_training = Weekly[ Year <= 2008, ]
> Weekly_testing = Weekly[ Year > 2008, ]
> model = glm(Direction ~ Lag2, family="binomial", data=Weekly_training)
> summary(model)

            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.20326    0.06428   3.162  0.00157 **
Lag2         0.05810    0.02870   2.024  0.04298 *
```

**Now, predict the *testing* data using the model that was calibrated based on the *training* data.**

```
> Probability = predict(model,data.frame(Weekly_testing),type ="response")
> Predicted_Direction = rep("Down",length(Probability))
> Predicted_Direction[ Probability > 0.5 ] = "Up"
> table( Weekly_testing$Direction, Predicted_Direction )

         Predicted_Direction
Direction Down Up
     Down    9 34
     Up      5 56

> mean( Weekly_testing$Direction == Predicted_Direction )
[1] 0.625
```

**Although this classification rule is also too optimistic, we got a 62.5% correct classification rate.**

**Why is it higher than before? A less flexible model, with only one independent variable that looked significant, has a better prediction power.**

**(d) Tuning, minimizing the prediction error rate.**

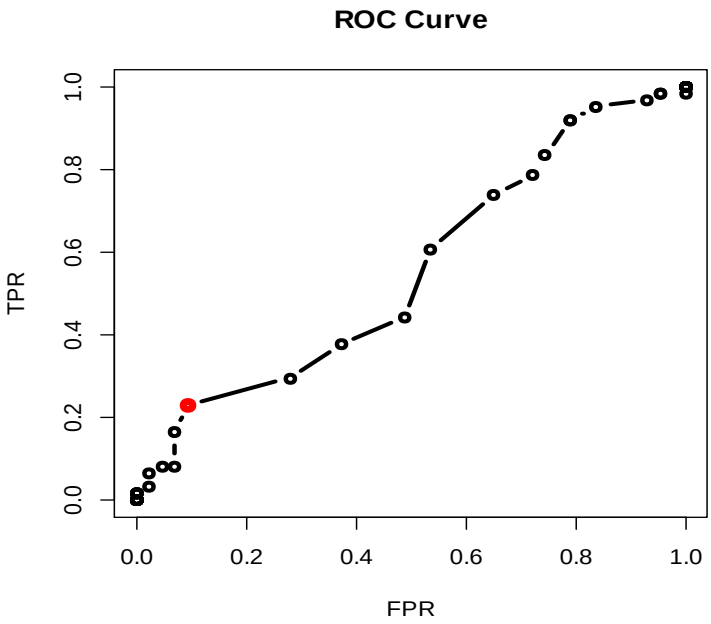
```
> error.rate = rep(0,100); TPR = rep(0,100); FPR = rep(0,100);
> threshold = seq(0.01,1,0.01);

> for (i in 1:100){
+ Predicted_Direction = rep("Down",length(Probability))
+ Predicted_Direction[ Probability > threshold[i] ] = "Up"
+ error.rate[i] = mean( Weekly_testing$Direction == Predicted_Direction )
+ TPR[i] = sum( Weekly_testing$Direction == "Up" & Predicted_Direction == "Up" ) / sum(
Weekly_testing$Direction == "Up" )
+ FPR[i] = sum( Weekly_testing$Direction == "Down" & Predicted_Direction == "Up" ) / sum(
Weekly_testing$Direction == "Down" )
+ }

> min(error.rate)
[1] 0.4134615
```

```
> threshold[which.min(error.rate)]
[1] 0.7
```

**The lowest prediction error rate of 0.413 is achieved by the threshold = 0.7 (instead of 0.5 that we used earlier). With this threshold, our prediction is correct only 58.7% of time.**

## (e) ROC curve

```
> plot(FPR,TPR,main="ROC Curve")
```




**# Searching for the elbow**

```
> x = max(which(TPR>0.2))
> points(FPR[x],TPR[x],lwd=5,col="red")
> threshold[x]
[1] 0.59
```

**# The red point is probably the closest to be an elbow. It corresponds to threshold = 0.59.**

## (f) Risk-based tuning

We are given a loss function L(k|j), where

$$L(up|up) = L(down|down) = 0 \text{ and } L(up|down) = 2L(down|1up).$$

Let p be the predicted probability of the up-trend. With this loss function, we should predict an up-trend if

$$\text{Risk}(up) < \text{Risk}(down)$$
$$(p)L(up|up) + (1-p)L(up|down) < (p)L(down|up) + (1-p)L(down|down)$$
$$(p)(0) + (1-p)(2) < (p)(1) + (1-p)(0)$$
$$2-2p < p$$
$$2 < 3p$$
$$p > 2/3$$

**The risk-based threshold is 2/3.**