ISLR Notes and Exercises

Nathaniel

March 2018

Contents

Chapter 4: Classification

Problems with OLS for Classification

Recall key assumptions of Ordinary Linear Regression:

1. Linearity in parameters $y = X\beta + u$
2. Exogeneity: $E(u|X) = 0$ or Predeterminedness: $E(u_t|X_t) = 0$
3. Homoscedasticity: $v(u_t) = \sigma^2$
4. Normality: $\varepsilon \sim Normal$
5. No Perfect Collinearity

For binary data, the regression model is

$$E(y_i) = p_i = \alpha + \beta x_i$$

with variance,

$$var(u) = (p_i)(1 - p_i) = (\alpha + \beta x_i)(1 - \alpha - \beta x_i)$$

There are at least three problems using OLS regression on categorical dependent variable.

Comparing with OLS assumption 4, one can see that homoscedasticity assumption is violated. The disturbance variance, $var(u)$, is at a maximum when $p_i = 0.5$ and gets small when p_i is near 1 or 0. There are two implications. First, the coefficient estimates are no longer efficient. Second, the standard error estimates are no longer consistent estimates of the true standard errors, meaning that the estimated standard errors could be biased (either upward or downward) to unknown degrees, so is the derived test statistics (one may have to resort to heteroscedasticity consistent covariance estimator).

The conditional distribution of $p(y \mid x)$ is a Bernoulli distribution rather than a Gaussian distribution.

Interpretability is another problem of using OLS. The predicted values (p_i) are not restricted to $[0, 1]$. Negative probabilities hinder interporetability.

Logistic Regression

The logit (log odd) model can be represented by:

$$\log(\frac{p_i}{1-p_i}) = X\beta$$

Odd ratios will then be:

$$\frac{p_i}{1-p_i} = e^{X\beta}$$

equivalently,

$$p(X) = \frac{e^{X\beta}}{1+e^{X\beta}}$$

Logistic regression does not make many of the key assumptions of linear regression and general linear models that are based on ordinary least squares algorithms such as linearity, normality, homoscedasticity, and measurement level. Logistic regression requires:

- Discrete outcome (dependent variable): the dependent variable should be dichotomous in nature;
- No outliers in the data, which can be assessed by converting the continuous predictors to standardized, or z-scores, and remove values below -3.29 or greater than 3.29;
- No high multicollinearity among the predictors.
- Linearity with the odds ratio and independent variable(s);
- Large sample size: This is because maximum likelihood (ML) estimates are less powerful than ordinary least squares. Whilst OLS needs 5 cases per independent variable in the analysis, ML needs at least 10 cases per independent variable, some statisticians recommend at least 30 cases for each parameter to be estimated.
- Independent error terms: Logistic regression requires each observation to be independent.

Marginal Effects

Marginal effects reflect the change in the probability of $y = 1$ given a 1 unit change in an independent variable x. The question is which data point we should pick to evaluate the cofficient.

Specifically, the marginal effects at the mean is given as:

$$\frac{\partial p}{\partial x_j} = F'(\bar{x}\beta)\beta_j$$

But such an "average" person may not exist, thereby hindering interpretability. Alternatively, we can use the average marginal effect,

$$\frac{\partial p}{\partial x_j} = \frac{1}{n}\sum F'(x\beta)\beta_j$$

Logit and Probit generally have similar marginal effects.

Case Control Sampling

Case control sampling is a chearper alternative than random experiment trial. Sampling more controls than cases reduces the variance of parameters estimated. But after the ratio of 5 to 1, the variance reduction flatens out, indicating a deminishing return from collecting more controls.

With case control sampling, one needs to adjust for the intercept through $\hat{\beta}_0^* = \hat{\beta}_0 + \log(\frac{\pi}{1-\pi}) - \log(\frac{\hat{\pi}}{1-\hat{\pi}})$, where π us the true prior probability and $\hat{\pi}$ is the estimated prior probability.

Stock Market Data ISLR Example

This eample refers to ISLR P154.

```
# The Stock Market Data
library(ISLR)
# summary(Smarket)
# pairs(Smarket,col=Smarket$Direction)
# cor(Smarket[,-9])
attach(Smarket)
```

```
# Logit Model
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
```

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

```
##  (Intercept)         Lag1         Lag2         Lag3         Lag4
## -0.126000257 -0.073073746 -0.042301344  0.011085108  0.009358938
```

```
##         Lag5       Volume
##  0.010313068  0.135440659
```

```
glm.probs=predict(glm.fit,type="response")
glm.probs[1:10] # Prob close to 0.5 indicates weak prediction
```

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

```
contrasts(Smarket$Direction)
```

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

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

# Generating Confusion Matrix
table(glm.pred,Direction)
```

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

```
(507+145)/1250
```

```
## [1] 0.5216
```

```
mean(glm.pred == Direction)
```

```
## [1] 0.5216
```

```
# Make training and test set
train = (Year < 2005)
Smarket.2005 = Smarket[!train,]
dim(Smarket.2005)
```

```
## [1] 252   9
```

```
Direction.2005 = Direction[!train]

# Refitting the data with sample from 2001-2004
glm.fit = glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,
              data=Smarket, family = binomial, subset = train)
glm.probs = predict(glm.fit, Smarket.2005, type = "response")
glm.pred=rep("Down", 252)
glm.pred[glm.probs > .5] = "Up"
table(glm.pred,Direction.2005)
```

```
##         Direction.2005
## glm.pred Down Up
##     Down   77 97
##     Up     34 44
```

```
mean(glm.pred == Direction.2005)
```

```
## [1] 0.4801587
```

```
mean(glm.pred != Direction.2005)
```

```
## [1] 0.5198413
```

Logistic regression correctly predicted the movement of the market 52.2% of the time. But the training error rate is often overly optimistic. It tends to underestimate the test error rate. If we trained and tested the model on the same set of 1250 observations, we get the training error rate of 0.48. Perhaps by removing the variables that appear not to be helpful in predicting Direction, we can obtain a more effective model.

```
# Fit smaller model
glm.fit = glm(Direction~Lag1+Lag2,data=Smarket,family=binomial,subset=train)
summary(glm.fit) # Nothing becomes significant anyway
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2, family = binomial, data = Smarket,
##     subset = train)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.345  -1.188   1.074   1.164   1.326
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.03222    0.06338   0.508    0.611
## Lag1        -0.05562    0.05171  -1.076    0.282
## Lag2        -0.04449    0.05166  -0.861    0.389
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1383.3  on 997  degrees of freedom
## Residual deviance: 1381.4  on 995  degrees of freedom
## AIC: 1387.4
##
## Number of Fisher Scoring iterations: 3
```

```
glm.probs = predict(glm.fit,newdata = Smarket[!train,],type = "response")
glm.pred=ifelse(glm.probs > 0.5,"Up","Down")
table(glm.pred,Direction.2005)
```

```
##         Direction.2005
## glm.pred Down  Up
##     Down   35  35
##     Up     76 106
```

```
mean(glm.pred == Direction.2005)
```

```
## [1] 0.5595238
```

```
# Logistic regression has a 58% accuracy rate predicting an increase in the market
106/(106+76)
```

```
## [1] 0.5824176
```

```
# To predict Direction on a day when Lag1 and Lag2 equal 1.2 and 1.1, use the predict() function:
predict(glm.fit,newdata = data.frame(Lag1 = c(1.2,1.5),
          Lag2 = c(1.1,-0.8)), type ="response")
```

```
##         1         2
## 0.4791462 0.4960939
```

Now the results appear to be a little better: 56% of the daily movements have been correctly predicted. It is worth noting that in this case, a much simpler strategy of predicting that the market will increase every day will also be correct 56% of the time! Hence, in terms of overall error rate, the logistic regression method is no better than the naive approach. However, the confusion matrix shows that on days when logistic regression predicts an increase in the market, it has a 58% accuracy rate. This suggests a possible trading strategy of buying on days when the model predicts an increasing market, and avoiding trades on days when a decrease is predicted. Of course one would need to investigate more carefully whether this small improvement was real or just due to random chance.

Linear Discrimant Analysis

Reasons for Linear Discrimant Analysis (LDA):

- When the classes are well-separated, the parameter estimates for the logistic regression model are surprisingly unstable (perfect classification problem). Also if n is small and the distribution of the predictors X is approximately normal in each of the classes, the linear discriminant model is more stable than the logistic regression model.
- Logistic regression uses the conditional likelihood based on $Pr(Y|X)$) known as discriminant learning while LDA uses the full likelihood based on $Pr(X, Y)$ known as generative learning.
- LDA is popular when we have more than two response classes.
- LDA assumes that the observations are drawn from a Gaussian distribution with a common covariance matrix in each class, and so can provide some improvements over logistic regression when this assumption approximately holds. Conversely, logistic regression can outperform LDA if these Gaussian assumptions are not met.

If model is correctly specified, Bayes classifier, which classifies an observation to the class for which $p_k(X)$ is largest, has the lowest possible error rate out of all classifiers. LDA attempts to come up with a estimator that approximates the Bayes classifier to minimize the total error rate out of all classifiers (on condition that the model is correctly specified).

An **inverse probability problem** has solution in the form of the probability that a certain process with a certain probability parameter of interest is being used to generate the observed data. In other owrds, *given the data we observed, which underlying data generateing process is more likely?* For instance, are the observations more likely generated from a fair coin heads or a bent coin heads given that for now we oberved n coins that are heads? The **Bayes Theorm** which specifies posterior probability as a function of marginal prior probability for class k, π_k provides a neat way to the problem. It states:

$$Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_l^K \pi_l f_l(x)}$$

where $f_k \equiv Pr(X = x|Y = k)$ denote the (with-in class) conditional density (likelihood) function of X for an observation that comes from the kth class. The distribution of f_k depends on the assumption one makes. In ISLS normality is assumed.

$$\text{posterior (observed)} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence (marginal)}}$$

where:

- posterior probability = probability after new data is observed
- prior probability = probability before any data is observed or before new data is observed

- likelihood = standard forward (conditional) probability of data given parameters
- marginal probability = probability of the data

In practice, parameters $\mu_1, ..., \mu_k, \pi_1, ..., \pi_K$, and σ^2 are estimated using

$$\hat{\pi}_k = \frac{n_k}{n}$$

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i$$

$$\hat{\sigma}^2 = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (x_i - \hat{\mu}_k)^2 = \sum_{k=1}^{K} \frac{n_k - 1}{n-K} \hat{\sigma}_k^2$$

where $\hat{\sigma}_k^2 = \frac{1}{n_k - 1} \sum_{i:y_i=k} (x_i - \hat{\mu}_k)^2$

With p predictors, a choice of assumption is the multivariate normal distribution ($N(\mu_\mathbf{k}, \mathbf{\Sigma})$) with $Var(X_1) = Var(X_2) = ... = Var(X_k)$ (K number of classes) and $Cov(X_1, X_k) = 0$ (the off-diagonal elements of the variance covariance matrix, Σ, common to all K classes is zero). A total of $K \times p$ linear cofficients are to be estimated.

Substituting the pdf of the assumed distribution into the Bayes Theorm, taking log and rearranging the terms, we get the discriminant score:

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log(\pi_k)$$

which is linear in x and can by expressed in the form $\delta_k(x) = c_{k_0} + c_{k_1} + ... + c_{k_p}$. Or one can combine c as a vector, $\mathbf{c}$, which are the coefficients of linear discriminants. The classifier assigns an observation $X = x$ to the class for which $\delta_k(x)$ is largest (by ranking $\delta_k(x)$).

Training error rates will usually be lower than test error rates, which are the real quantity of interest. Beaware of overfitting.

Importantly, the threhold for the posterior probability $p_k(x)$ can be varied according to domain knowledge such that error rate (*null rate*) of a given class could be decreased/increased. For example, we can change $Pr(default = Yes|X = x) > 0.5$ to $Pr(default = Yes|X = x) > 0.2$. The choice of the threshold may be backed by domain knowledge. The change in threhold can be seen in ROC curve (Power as a function of Type I Error), AUC, confusion matrix, sensitivity, specificity and other related concepts.

Naive Bayes

Back to the Bayes Theorm,

$$Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_l^K \pi_l f_l(x)}$$

If we impose conditional independence (a wrong but useful assumption) on the conditional density function, that is, letting

$$f_k(x) = \prod_{j=1}^{p} = f_{jk}(x_j)$$

the covariance matrix of for each class, $\mathbf{\Sigma_k}$ is diagonal. The discriminant score becomes:

$$\delta_k(x) = -\frac{1}{2} \sum_{j=1}^{p} \left[\frac{(x - \mu_{jk})^2}{\sigma_{kj}^2} \right] + \log \pi_k$$

Quadratic Discriminant Analysis

In contrast to LDA, Quadratic Discriminant Analysis (QDA) assumes that each class has its own covariance matrix. That is, it assumes that an observation from the kth class is of the form $X \sim N(\mu_k, \Sigma_k)$, where Σ_k is a covariance matrix for the kth class.

The discriminant score of QDA, $\delta_k(x)$, is a quadratic function of x with a total of $Kp(p+1)/2$ parameters to be estimated (recall that of LDA is $K \times p$):

$$\delta_k(x) = -\frac{1}{2}(x-\mu_k)^T\Sigma_k(x-\mu_k) + \log\pi_k - \frac{1}{2}\log|\Sigma_k|$$

Bias-variance trade-off is crucial in chosing between LDA and QDA. "Roughly speaking, LDA tends to be a better bet than QDA if there are relatively few training observations and so reducing variance is crucial. In contrast, QDA is recommended if the training set is very large, so that the variance of the classifier is not a major concern, or if the assumption of Σ_k is clearly untenable."

```
# Linear Discriminant Analysis
require(MASS)
```

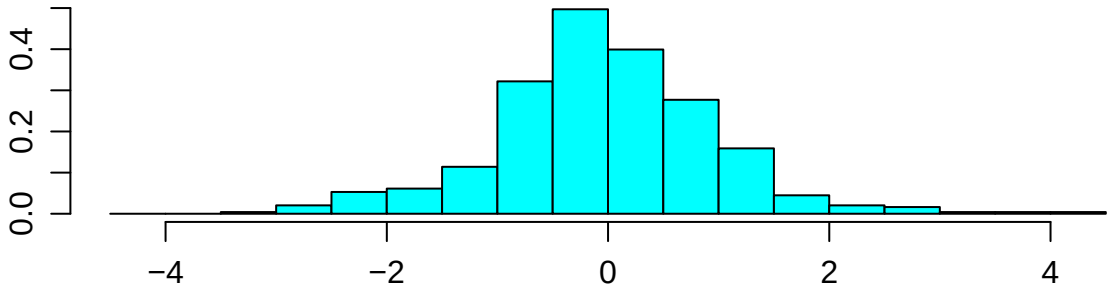
```
## Loading required package: MASS
```

```
data(Smarket)
# Instead of making training and test set. Use Year<2005
lda.fit = lda(Direction ~ Lag1 + Lag2, data = Smarket, subset = Year < 2005)
lda.fit
```

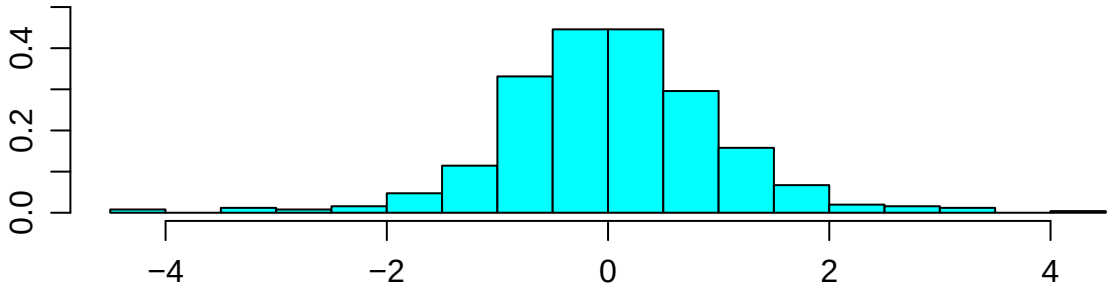
```
## Call:
## lda(Direction ~ Lag1 + Lag2, data = Smarket, subset = Year <
##     2005)
##
## Prior probabilities of groups:
##     Down       Up
## 0.491984 0.508016
##
## Group means:
##             Lag1        Lag2
## Down  0.04279022  0.03389409
## Up   -0.03954635 -0.03132544
##
## Coefficients of linear discriminants:
##             LD1
## Lag1 -0.6420190
## Lag2 -0.5135293
```

Prior probabilities of 0.5 may sugggest Random Walk. The coefficients of linear discriminants output provides the linear combination of `Lag1` and `Lag2` that are used to form the LDA decision rule. In other words, these are the multipliers of the elements of $X = x$ in (ISLR Equation 4.19). If $-0.642 * \text{Lag1} - 0.514 * \text{Lag2}$ is large, then the LDA classifier will predict a market increase, vice versa.

```
par(mar=c(3.5, 3.5, 2, 1), mgp=c(2.4, 0.8, 0)); plot(lda.fit)
```



group Down



group Up

```
lda.pred=predict(lda.fit, Smarket.2005)
names(lda.pred)
```

```
## [1] "class"     "posterior" "x"
```

```
lda.class=lda.pred$class
Direction.2005=Direction[!train]
table(lda.class,Direction.2005)
```

```
##          Direction.2005
## lda.class Down  Up
##      Down   35  35
##      Up     76 106
```

```
# Classification Rate:
mean(lda.class==Direction.2005)
```

```
## [1] 0.5595238
```

```
ls(lda.pred) # Same as names(lda.pred)
```

```
## [1] "class"     "posterior" "x"
```

```
# Notice that the posterior probability output by the model corresponds to
# the probability that the market will decrease:
sum(lda.pred$posterior[,1]>=.5) # Adding all predicted Ups
```

```
## [1] 70
```

```
sum(lda.class=="Down")
```

```
## [1] 70
```

```
sum(lda.pred$posterior[,1]<.5) # Adding all predicted Downs
```

```
## [1] 182
```

```
sum(lda.class=="Up")
```

```
## [1] 182
```

```
# = Row2 of the confusion matrix
lda.pred$posterior[1:20,1]
```

```
##       999      1000      1001      1002      1003      1004      1005
## 0.4901792 0.4792185 0.4668185 0.4740011 0.4927877 0.4938562 0.4951016
##      1006      1007      1008      1009      1010      1011      1012
## 0.4872861 0.4907013 0.4844026 0.4906963 0.5119988 0.4895152 0.4706761
##      1013      1014      1015      1016      1017      1018
## 0.4744593 0.4799583 0.4935775 0.5030894 0.4978806 0.4886331
```

```
lda.class[1:20]
```

```
##  [1] Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Down Up   Up
## [15] Up   Up   Up   Down Up   Up
## Levels: Down Up
```

```
sum(lda.pred$posterior[,1]>.9)
```

```
## [1] 0
```

```
max(lda.pred$posterior[,1])
```

```
## [1] 0.520235
```

```
# Change the posterior probability threshold to at least 90% and see the
# number of observations meeting the threshold. The result shows none yet.

# Quadratic Discriminant Analysis
qda.fit=qda(Direction~Lag1+Lag2,data=Smarket,subset=train)
qda.fit
```

```
## Call:
## qda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)
##
## Prior probabilities of groups:
##     Down       Up
## 0.491984 0.508016
##
## Group means:
##             Lag1        Lag2
## Down  0.04279022  0.03389409
## Up   -0.03954635 -0.03132544
```

```
qda.class=predict(qda.fit,Smarket.2005)$class
table(qda.class,Direction.2005)
```

```
##          Direction.2005
## qda.class Down  Up
##      Down   30  20
##      Up     81 121
```

```
mean(qda.class==Direction.2005)
```

```
## [1] 0.5992063
```

KNN Classifier

"KNN is a completely non-parametric approach: no assumptions are made about the shape of the decision boundary. Therefore, we can expect this approach to dominate LDA and logistic regression when the decision boundary is highly non-linear. On the other hand, KNN does not tell us which predictors are important; we don't get a table of coefficients" (ISLR P. 151).

QDA serves as a compromise between the non-parametric KNN method and the linear LDA and logistic regression approaches. Since QDA assumes a quadratic decision boundary, it can accurately model a wider range of problems than can the linear methods. Though not as flexible as KNN, QDA can perform better in the presence of a limited number of training observations because it does make some assumptions about the form of the decision boundary (parametric vs non-parametric).

```
train=Year<2005 # Indicater Variable
Xlag=cbind(Lag1,Lag2)
dim(Xlag)
```

```
## [1] 1250    2
```

```
dim(Xlag[train,])
```

```
## [1] 998   2
```

```
dim(Xlag[!train,])
```

```
## [1] 252   2
```

```
knn.pred=knn(Xlag[train,],Xlag[!train,],Direction[train],k=1)
summary(knn.pred)
```

```
## Down   Up
##  101  151
```

In R, `knn` function require 4 Inputs:

1. A matrix (dataframe) containing the predictors associated with the training data
2. A matrix (dataframe) containing the predictors associated with the data for which we wish to make predictions
3. A vector containing the class labels for the training observations,
4. A value for K, the number of nearest neighbors to be used by the classifier.

```
table(knn.pred,Direction[!train])
```

```
##
## knn.pred Down Up
##     Down   43 58
##     Up     68 83
```

```
mean(knn.pred==Direction[!train])
```

```
## [1] 0.5
```

```
# Alternatively
train.X=cbind(Lag1,Lag2)[train,]
test.X=cbind(Lag1,Lag2)[!train,]
```

```
train.Direction=Direction[train]
set.seed(1)
knn.pred=knn(train.X,test.X,train.Direction,k=1)
table(knn.pred,Direction.2005)
```

```
##         Direction.2005
## knn.pred Down Up
##     Down   43 58
##     Up     68 83
```

```
(83+43)/252
```

```
## [1] 0.5
```

```
knn.pred=knn(train.X,test.X,train.Direction,k=3)
table(knn.pred,Direction.2005)
```

```
##         Direction.2005
## knn.pred Down Up
##     Down   48 54
##     Up     63 87
```

```
mean(knn.pred==Direction.2005)
```

```
## [1] 0.5357143
```

```
# It appears that for this data, QDA provides the best results
# of the methods that we have examined so far.
```

Exercises

Question 4

Curse of dimensionality: When the number of features p is large, there tends to be a deterioration in the performance of KNN and other local approaches that perform prediction using only observations that are near the test observation for which a prediction must be made. This phenomenon is known as the curse of dimensionality and it ties into the fact that parametric approaches often perform poorly when p is large.

Suppose that we have a set of observations, each with measurements on $p = 1$ feature, X. We assume that X is uniformly (evenly) distributed on [0, 1]. Associated with each observation is a response value. Suppose that we wish to predict a test observation's response using only observations that are within 10% of the range of X closest to that test observation. For instance, in order to predict the response for a test observation with $X = 0.6$, we will use observations in the range $[0.55, 0.65]$.On average, what fraction of the available observations will we use to make the prediction?

ANS: $ 0.65-0.55/1=10%$

Now suppose that we have a set of observations, each with measurements on $p = 2$ features, X_1 and X_2. We assume that (X_1, X_2) are uniformly distributed on $[0, 1] \times [0, 1]$. We wish to predict a test observation's response using only observations that are within 10% of the range of X_1 and within 10% of the range of X_2 closest to that test observation. For instance, in order to predict the response for a test observation with $X_1 = 0.6$ and $X_2 = 0.35$, we will use observations in the range $[0.55, 0.65]$ for X_1 and in the range $[0.3, 0.4]$ for X_2. On average, what fraction of the available observations will we use to make the prediction?

ANS: For 2 features, $(0.65 - 0.55) * (0.4 - 0.3) = 1\%$

ANS: For 100 features, we will use 0.10^{100} of the available observations.

Consequencly, one drawback of KNN when p is large is that there are very few training observations "near" any given test observation. As the number of predictor, p, increases linearly, the percentage of observations that can be used to predict with KNN declines expoentially. This means that for a set sample size, more features leads to fewer neighbors.

A hypercube is a generalization of a cube to an arbitrary number of dimensions. When $p = 1$, a hypercube is simply a line segment, when $p = 2$ it is a square, and when $p = 100$ it is a 100-dimensional cube. Now suppose that we wish to make a prediction for a test observation by creating a p-dimensional hypercube centered around the test observation that contains, on average, 10% of the training observations. For $p = 1$, 2, and 100, what is the length of each side of the hypercube?

The length of each side of the hypercube increases as p rises. When

- $p = 1, \text{side length} = 0.10$
- $p = 2, \text{side length} = 0.10^{1/2} \approx 0.32$
- $p = 3, \text{side length} = 0.10^{1/3} \approx 0.46$
  ...
- $p = 100, \text{side length} = 0.10^{1/100} \approx 0.98$
- $p = N, \text{side length} = 0.10^{1/N}$

Thus when the number of features is high, to use on average 10% of the training observations would mean that length of the side will asymptotically approach 1. We would need to include almost the entire range of each individual feature.

Question 5

We now examine the differences between LDA and QDA.

(a) If the Bayes decision boundary is linear, do we expect LDA or QDA to perform better on the training set? On the test set?

If the Bayes decision boundary is linear, on the training set, QDA might outperform LDA as it implies a more flexible classifer, thereby entailing a better fit to observations in the training set. On the test set, however, QDA is expected to perform worse than LDA as QDA's estimators is more sensitive to a change in sample. In other words, QDA overfitting would yield a higher variance of estimators and thus higher expected test MSE. Recall that $E(y_0 - \hat{f}(x_0))^2 = Var(\hat{f}(x_0) + [Bias(\hat{f}(x_0))]^2 + Var(\varepsilon)$.

(b) If the Bayes decision boundary is non-linear, do we expect LDA or QDA to perform better on the training set? On the test set?

If the Bayes decision boundary is non-linear, on the training set, QDA is expected to outperform LDA as it exerts an extra parameter. Same in the test set given non-lnearity is the correct specified model.

(c) In general, as the sample size n increases, do we expect the test prediction accuracy of QDA relative to LDA to improve, decline, or be unchanged? Why?

As the sample size n increases, we expect the test prediction accuracy of QDA relative to LDA to improve. This is because variance of estimators, (which is captured in the overfitting problem) will decline as degrees of freedom increase, holding a given number of estimators.

(d) True or False: Even if the Bayes decision boundary for a given problem is linear, we will probably achieve a superior test error rate using QDA rather than LDA because QDA is flexible enough to model a linear decision boundary. Justify your answer

False: The reason is outlined in part(a).

Question 6

Suppose we collect data for a group of students in a statistics class with variables $X1$ = hours studied, $X2$ = undergrad GPA, and Y = receive an A. We fit a logistic regression and produce estimated coefficient, $\hat{\beta}_0 = -6$, $\hat{\beta}_1 = 0.05$, $\hat{\beta}_2 = 1$.

(a) Estimate the probability that a student who studies for 40 hours and has an undergrad GPA of 3.5 gets an A in the class.

$\Pr(\text{required}) = \frac{\exp(-6+0.05\times 40+3.5)}{1+\exp(-6+0.05\times 40+3.5)} = 0.3775$

(b) How many hours would the student in part (a) need to study to have a 50% chance of getting an A in the class?

$0.5 = \frac{\exp(-6+0.05\times x_1+3.5)}{1+\exp(-6+0.05\times x_1+3.5)} \Rightarrow x_1 = 50$ hours

Question 7

Suppose that we wish to predict whether a given stock will issue a dividend this year ("Yes" or "No") based on X, last year's percent profit.We examine a large number of companies and discover that the mean value of X for companies that issued a dividend was $X = 10$, while the mean for those that didn't was $X = 0$. In addition, the variance of X for these two sets of companies was $\hat{\sigma}^2 = 36$. Finally, 80% of companies issued dividends. Assuming that X follows a normal distribution, predict the probability that a company will issue a dividend this year given that its percentage profit was $X = 4$ last year.

Using the normal density function and Bayes Therom, $Pr(X = 4) = \frac{0.8\exp(-\frac{1}{2\times 36}(4-10)^2)}{0.8\exp(-\frac{1}{2\times 36}(4-10)^2)+(1-0.8)\exp(-\frac{1}{2\times 36}(4-0)^2)}$

Note that the $\frac{1}{2\pi\sigma^2}$ parts in the nominator and the denominator cancel out. The denominator comes from the infomation that the decision of paying dividend is bernoulli distributied. The required probibility is 0.75.

Question 8

Suppose that we take a data set, divide it into equally-sized training and test sets, and then try out two different classification procedures. First we use logistic regression and get an error rate of 20% on the training data and 30% on the test data. Next we use 1-nearest neighbors (i.e. K = 1) and get an average error rate (averaged over both test and training data sets) of 18%. Based on these results, which method should we prefer to use for classification of new observations? Why?

Logistic regression should be prefered. That average error rate for 1-nearest neighbors (1NN) is 18% does not imply that training error rate = test error rate \$ = 18%\$. Moe likely, the training error rate is going to be very low as 1NN is flexible enough to take almost all of the data points into account. This means that it is likely to overfit data in the test set and therefore the test error rate is going to be high (close to 36% in this case). Given that the test error rate for logistic regression is 30%, it seems that logistic regression may actually have a lower test error rate.

Question 9

This problem has to do with odds, $p(X)/[1 - p(X)]$. Recall on average nine out of every ten people with an odds of 9 will default, since $p(X) = 0.9$ implies an odds of $0.9/(1 - 0.9) = 9$ meaning that outcome of X occuring is 9 times more as likely as X not occuring.

(a) On average, what fraction of people with an odds of 0.37 of defaulting on their credit card payment will in fact default?

$\frac{p}{1-p} = 0.37 \Rightarrow p = 0.27$

(b) Suppose that an individual has a 16% chance of defaulting on her credit card payment. What are the odds that she will default?

$\frac{0.16}{1-0.16} = 0.19$

Question 10

This applied question works on the `Weekly` dataset which consist of weekly percentage returns for the S&P 500 stock index between 1990 and 2010. Except for `Year` and `Volume`, variables seem to be uncorrelated with one another.

```
library(ISLR)
dim(Weekly)
```

```
## [1] 1089    9
```

```
summary(Weekly)
```

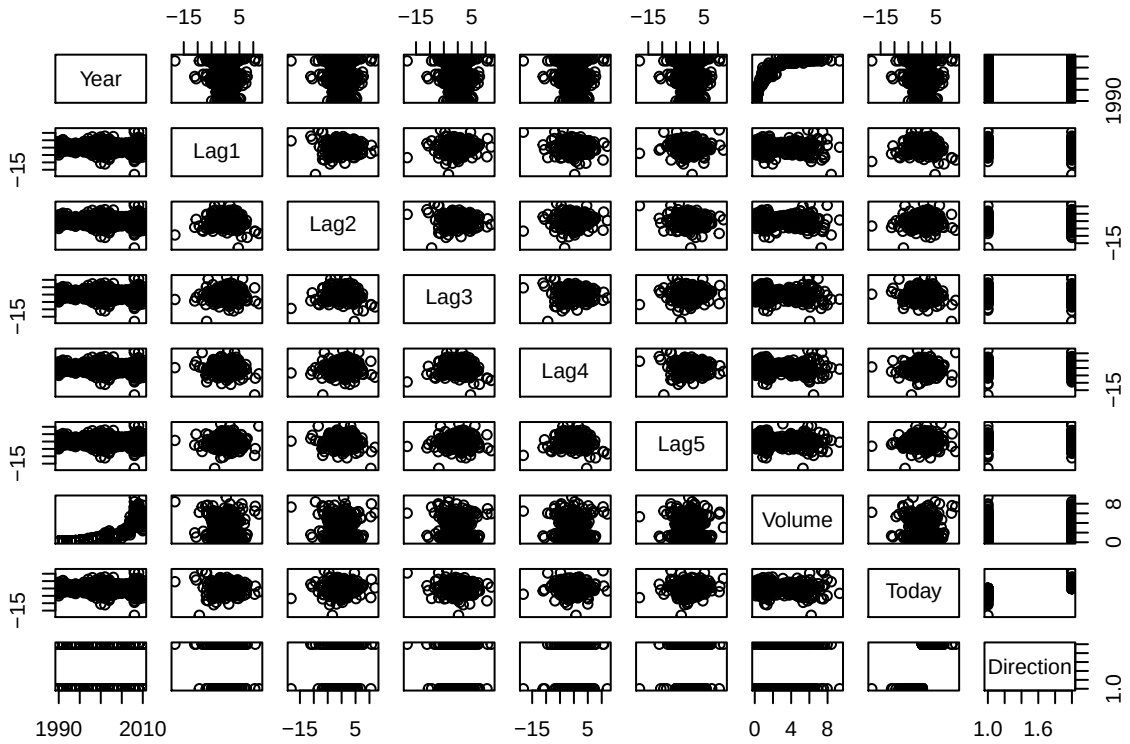
```
##       Year           Lag1               Lag2               Lag3         
##  Min.   :1990   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950  
##  1st Qu.:1995   1st Qu.: -1.1540   1st Qu.: -1.1540   1st Qu.: -1.1580  
##  Median :2000   Median :  0.2410   Median :  0.2410   Median :  0.2410  
##  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       
##  Min.   :-18.1950   Min.   :-18.1950   Min.   :0.08747  
##  1st Qu.: -1.1580   1st Qu.: -1.1660   1st Qu.:0.33202  
##  Median :  0.2380   Median :  0.2340   Median :1.00268  
##  Mean   :  0.1458   Mean   :  0.1399   Mean   :1.57462  
##  3rd Qu.:  1.4090   3rd Qu.:  1.4050   3rd Qu.:2.05373  
##  Max.   : 12.0260   Max.   : 12.0260   Max.   :9.32821  
##      Today          Direction 
##  Min.   :-18.1950   Down:484  
##  1st Qu.: -1.1540   Up  :605  
##  Median :  0.2410             
##  Mean   :  0.1499             
##  3rd Qu.:  1.4050             
##  Max.   : 12.0260
```

```
cor(Weekly[,2:8])
```

```
##                Lag1        Lag2        Lag3         Lag4         Lag5
## Lag1    1.000000000 -0.07485305  0.05863568 -0.071273876 -0.008183096
## Lag2   -0.074853051  1.00000000 -0.07572091  0.058381535 -0.072499482
## Lag3    0.058635682 -0.07572091  1.00000000 -0.075395865  0.060657175
## Lag4   -0.071273876  0.05838153 -0.07539587  1.000000000 -0.075675027
## Lag5   -0.008183096 -0.07249948  0.06065717 -0.075675027  1.000000000
## Volume -0.064951313 -0.08551314 -0.06928771 -0.061074617 -0.058517414
## Today  -0.075031842  0.05916672 -0.07124364 -0.007825873  0.011012698
##              Volume        Today
## Lag1   -0.06495131 -0.075031842
## Lag2   -0.08551314  0.059166717
## Lag3   -0.06928771 -0.071243639
## Lag4   -0.06107462 -0.007825873
## Lag5   -0.05851741  0.011012698
```

```
## Volume  1.00000000 -0.033077783
## Today  -0.03307778  1.000000000
```

```
pairs(Weekly)
```




```
# Logistic Regression (Full Model)
glm.fit=glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume,
data=Weekly,family=binomial(link = "logit"))
summary(glm.fit)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial(link = "logit"), data = Weekly)
##
## 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
```

```
glm.probs=predict(glm.fit,type="response")
glm.probs[2:8]
```

```
##         2         3         4         5         6         7         8
## 0.6010314 0.5875699 0.4816416 0.6169013 0.5684190 0.5786097 0.5151972
```

```
contrasts(Weekly$Direction) #See how R code the dummy response vaiable
```

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

```
# Odd Ratios
exp(glm.fit$coefficients)
```

```
## (Intercept)        Lag1        Lag2        Lag3        Lag4        Lag5
##   1.3058630   0.9595710   1.0601831   0.9840671   0.9725924   0.9856322
##      Volume
##   0.9775151
```

```
# Marginal Effect
(pdf <-  mean(dlogis(predict(glm.fit, type = "link"))))
```

```
## [1] 0.2447041
```

```
(marginal.effects <- pdf*coef(glm.fit))
```

```
##  (Intercept)         Lag1         Lag2         Lag3         Lag4
##  0.065302739 -0.010098677  0.014300915 -0.003930227 -0.006800377
##         Lag5       Volume
## -0.003541373 -0.005564945
```

The type="response" option tells R to output probabilities of the form $P(Y = 1|X)$, as opposed to other information such as the logit. From the result, only the coefficient of `Lag2` is statistically significant with a p-value of 0.0296.

One way to interpret the significant coefficient of `Lag2` is that holding other variables constant, an extra one percentage return for 2 weeks previous increases the probability of positive return of hte market on a given week by 1.43%.

```
# Confusion Matrix
glm.pred=rep("Down", length(glm.probs))
glm.pred[glm.probs >.5] = "Up"
(t<-table(glm.pred, Weekly$Direction))
```

```
##
## glm.pred Down  Up
##     Down   54  48
##     Up    430 557
```

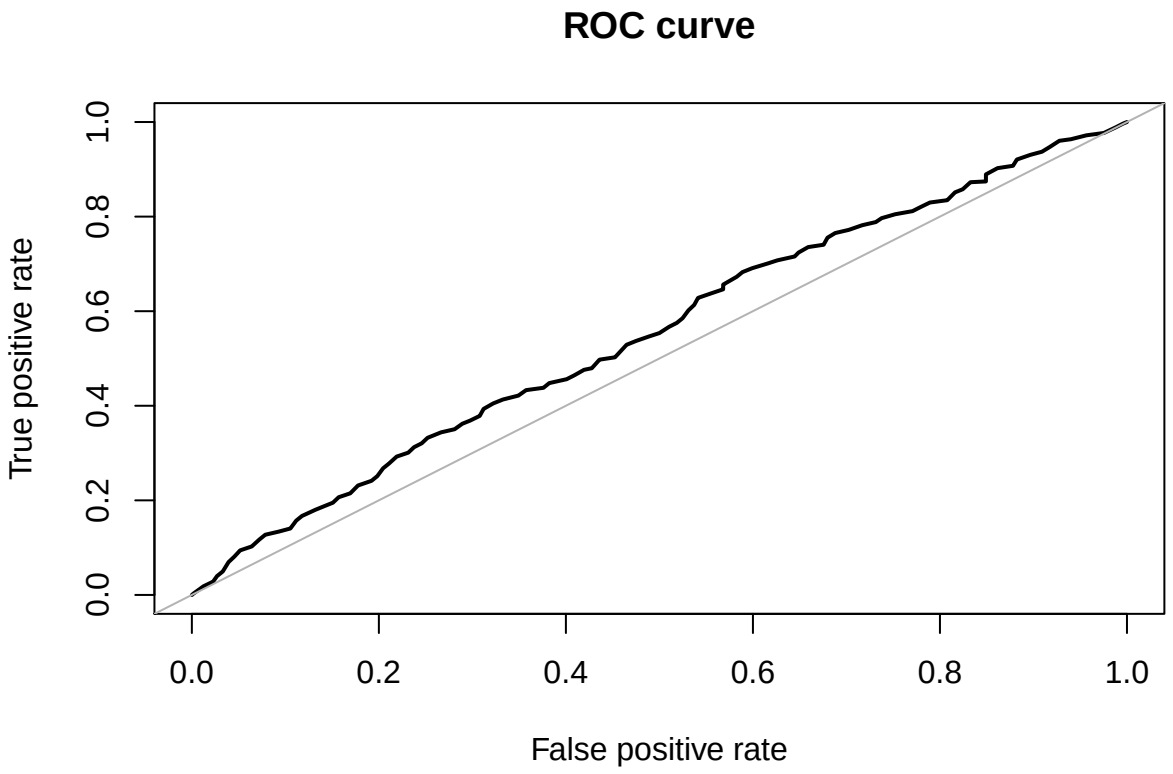
```
sum(diag(t))/sum(t) # True Positive Rate
```

```
## [1] 0.5610652
```

```
# ROC Curve
library(rpart);library(ROSE)
```

```
## Loaded ROSE 0.0-3
```

```
Dec_Tree<-rpart(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume,data=Weekly)
roc.curve(Weekly$Direction, glm.probs)
```




```
## Area under the curve (AUC): 0.554
```

```
plot(Dec_Tree) #text(Dec_Tree, pretty=0) #summary(Dec_Tree)
exp(coef(glm.fit)) # exponentiated coefficients
exp(confint(glm.fit)) # 95% CI for exponentiated coefficients
```

- False Positive Rate (FP/N) $= (48 + 430)/1089 = 43.89\%$ = Type I error = Training Error Rate.
- True Positive Rate (TP/N) $= (54 + 557)/1089 = 56.1\%$
- When `Direction` is `Up`, logit correctly predicts that $557/(557 + 48) = 92.06\%$ of the time.
- When `Direction` is `Down`, logit correctly predicts that $54/(54 + 430) = 11.15\%$ of the time.

It seems that logit model has a high accuracy in detecting market return rising but does a poor job in identifying market return falling. Overall, the model perform fairly as the Area under the curve (AUC) is merely 0.554, slightly higher than 0.5. But this is expected for any typical stock market data.

```
library(pROC)
auc(Weekly$Direction ~ glm.probs)
plot(roc(Weekly$Direction, glm.probs), main="ROC curve")
```

Moving on to applying logistic regression model on a training data period from 1990 to 2008 with `Lag2` as the only predictor, we can see noticeable improvement in accuracy which has risen to 62.5%. Yet, AUC, the percentage of the ROC plot that is underneath the curve, is 0.546, slightly less than that of the full model (0.554). These may reflect class imbalance (only 14 cases of `Down` but 90 for `Up`. Under this circumstance,

AUC is more reliable as it is immune to class imbalance.

Another advantage is that AUC does not require a classification threshold. Yet, AUC is not ideal as it uses different misclassification cost distributions for different classifiers. This means that using the AUC is equivalent to using different metrics to evaluate different classification rules. For further information, see https://www.r-bloggers.com/illustrated-guide-to-roc-and-auc/

```
# Applying Logit Model from train set 1990-2008 to test set 2009-2010
train.yrs <- Weekly$Year %in% (1990:2008)
  #this subsetting method is different from the book
train <- Weekly[train.yrs,]
test <- Weekly[!train.yrs,]
glm.fit2 <- glm(Direction~Lag2, data=train, family=binomial)
summary(glm.fit2)
```

```
##
## Call:
## glm(formula = Direction ~ Lag2, family = binomial, data = train)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.536  -1.264   1.021   1.091   1.368
##
## Coefficients:
##             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 *
## ---
## 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: 1350.5  on 983  degrees of freedom
## AIC: 1354.5
##
## Number of Fisher Scoring iterations: 4
```

```
glm.prob2 <- predict(glm.fit2, test, type="response")
summary(glm.prob2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.4488  0.5331  0.5573  0.5555  0.5826  0.6954
```

```
glm.pred2 <- ifelse(glm.prob2 > 0.5, "Up", "Down")
table(glm.pred2, test$Direction)
```

```
##
## glm.pred2 Down Up
##      Down    9  5
##      Up     34 56
```

```
mean(glm.pred2 == test$Direction)   # Accuracy=0.625
```

```
## [1] 0.625
```

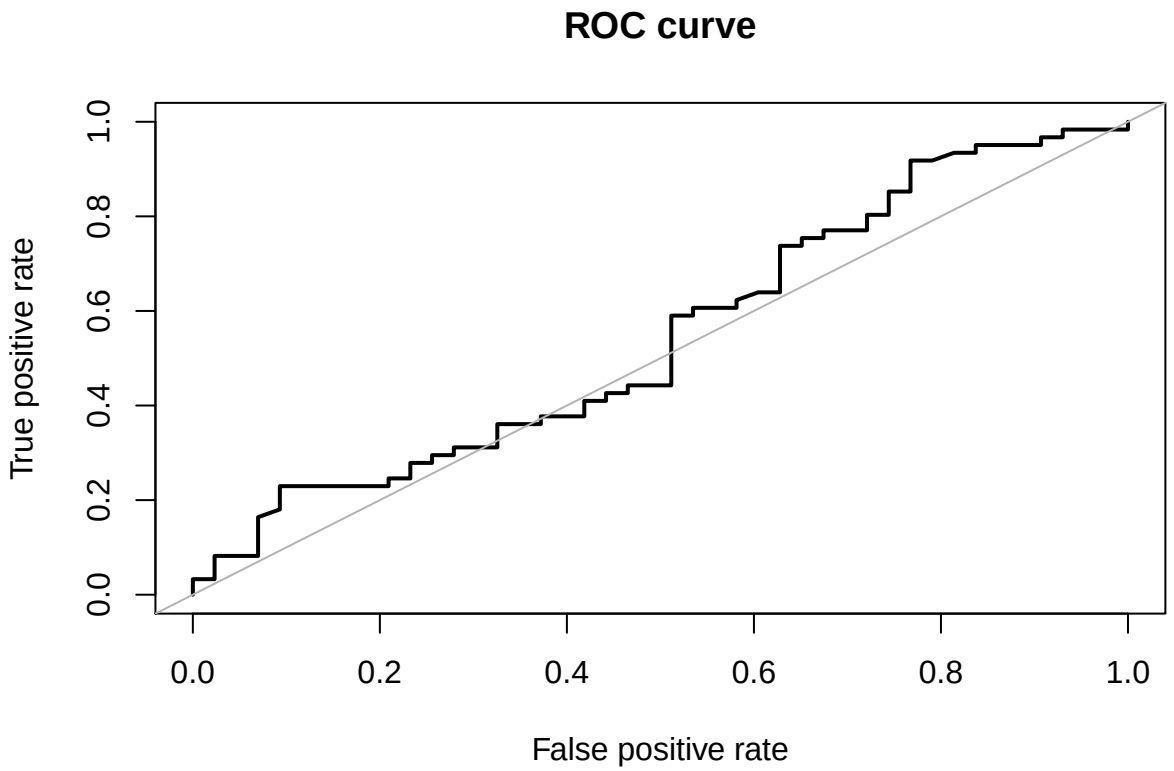
```
(pdf2 <-  mean(dlogis(predict(glm.fit2, type = "link"))))
```

```
## [1] 0.2462242
```

```
(marginal.effect2 <- pdf2*coef(glm.fit2)) # same magnitude as before
```

```
## (Intercept)        Lag2
##  0.05004689  0.01430446
```

```
# ROC from model on test set
library(rpart); library(ROSE)
Dec_Tree<-rpart(test$Direction~Lag2,data=test)
roc.curve(test$Direction, glm.prob2)
```




```
## Area under the curve (AUC): 0.546
```

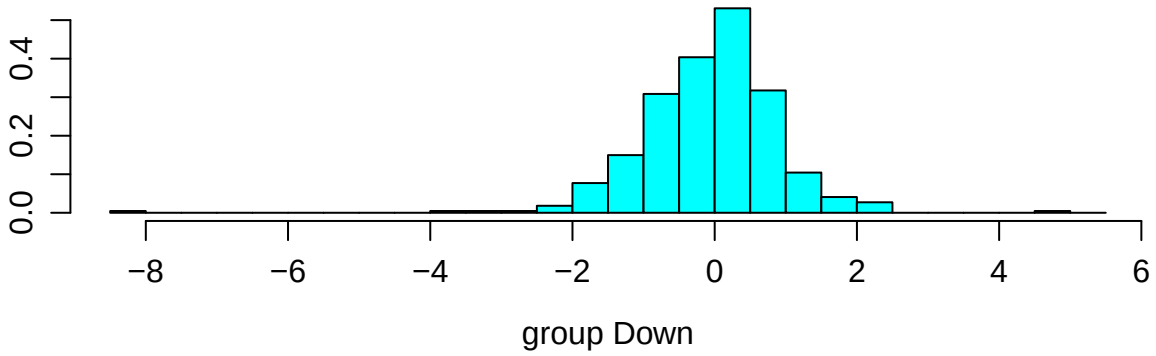
```
# Strangly pROC package gives AUC less than 0.5...
library(pROC)
(auc_2 <- auc(test$Direction ~ glm.prob2))
plot(roc(test$Direction, glm.prob2), main="ROC curve")
```

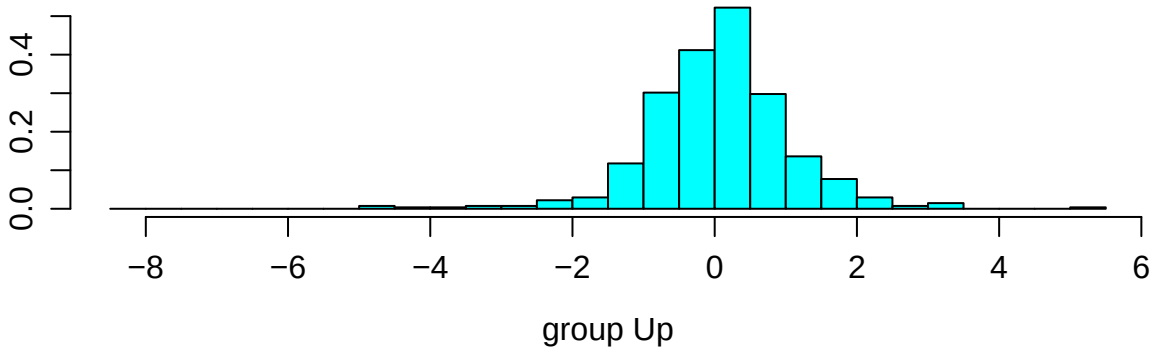
```
# Linear Discriminant Analysis
library(MASS)
(lda.fit = lda(train$Direction~Lag2, data=train))
```

```
## Call:
## lda(train$Direction ~ Lag2, data = 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
```

```
#pi hat for class 1&2
#Group mean for Lag2 Up = mean(test$Lag2[test$Direction=='Up'])
par(mar=c(3.5, 3.5, 2, 1), mgp=c(2.4, 0.8, 0)); plot(lda.fit)
```






```
#plot() produces plots of the linear discriminants: LD coefficient*Lag2.
#hist(0.3262636*test$Lag2[test$Direction=='Down'], breaks = 50)
#hist(0.3262636*test$Lag2[test$Direction=='Up'], breaks = 50)
lda.pred <- predict(lda.fit, test); names(lda.pred)
```

```
## [1] "class"     "posterior" "x"
```

```
table(lda.pred$class, test$Direction)
```

```
##
##        Down Up
##   Down    9  5
##   Up     34 56
```

```
mean(lda.pred$class == test$Direction)  # Accuracy = 0.625
```

```
## [1] 0.625
```

```
sum(lda.pred$posterior[,1]>=.5)
```

```
## [1] 14
```

```r
#count of obs over posterior prob threshold of 0.5
#Threshold can be adjusted to increase sensitivity or specificity
sum(lda.pred$posterior[,1]<.5) # sum of row 2 of the confusion matrix
```

```
## [1] 90
```

Applying LDA ($p = 1$) instead of Logit Model on the test set, one observes the same true positive rate of 0.625 and false positive rate of 0.375. Posterior probability of `Down` is 11.53% and that of `Up` is 88.46%. They are highter than the proportion of Direction = Down and Direction = Up (0.41 and 0.59 respectively), probably due to the positive coefficient of `Lag2` on `Direction` and the larger proportion of `Up` in `Lag2` (≈ 0.69) which indicates a high class imbalance.

"Accuracy is not a reliable metric for the real performance of a classifier, because it yields misleading results when the numbers of observations in different classes vary greatly." (Wiki).

```r
# Quadratic Discriminant Analysis
(qda.fit<-qda(Direction~Lag2,data=train))
```

```
## Call:
## qda(Direction ~ Lag2, data = train)
##
## Prior probabilities of groups:
##      Down        Up
## 0.4477157 0.5522843
##
## Group means:
##             Lag2
## Down -0.03568254
## Up    0.26036581
```

```r
qda.pred<-predict(qda.fit, test)
table(qda.pred$class, test$Direction)
```

```
##
##        Down Up
##   Down    0  0
##   Up     43 61
```

```r
mean(qda.pred$class == test$Direction) # Accuracy = 0.5865
```

```
## [1] 0.5865385
```

One noticeable result from QDA is that it scores an accuracy of 58.7% even though it only chooses `Up` the whole time. Not surprisingly, the prior probabilities of groups and group means are the same as in LDA (they are the inputs of the model!).

```r
# KNN with K = 1: 2 matrix, 1 vector, 1 scaler as inputs
library(class); set.seed(1);
train.Lag2 = as.matrix(train$Lag2); test.Lag2 = as.matrix(test$Lag2);
knn.pred = knn(train.Lag2, test.Lag2, train$Direction, k=1)
table(knn.pred, test$Direction)
```

```
##
## knn.pred Down Up
##     Down   21 30
##     Up     22 31
```

```r
mean(knn.pred == test$Direction)
```

```
## [1] 0.5
```

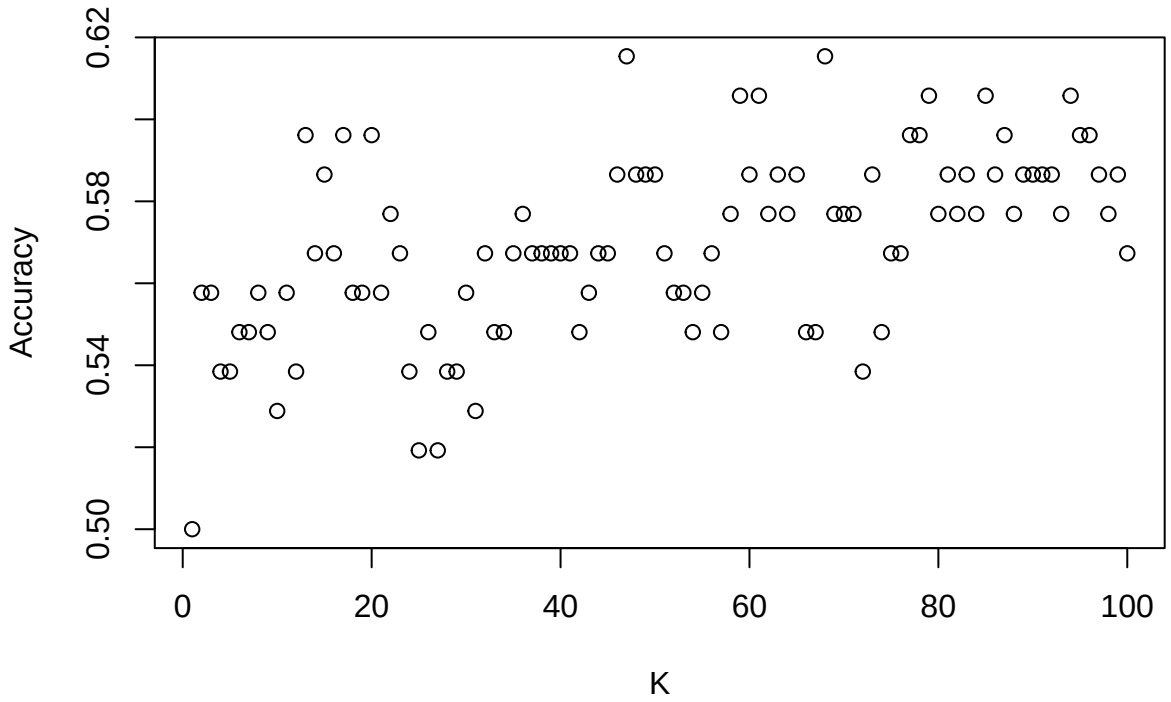
Next shows that the results using KNN with $K = 1$ on `Direction` and `Lag2` are not very good, since only 50% of the observations are correctly predicted. This is probably because $K = 1$ results in an overly flexible fit to the data. Increasing the number of nearest neighbor results in higher accuracy. $K = 4$ seems to be a reasonable choice at least locally. See below:

```
# Accuracy as a result of increase of K
set.seed(1); knn_acc <- c()
for (i in c(1:10)){
knn.pred = knn(train.Lag2, test.Lag2, train$Direction, k=i)
m <-mean(knn.pred == test$Direction)
knn_acc <- c(knn_acc, m)}
names(knn_acc)<-(1:length(knn_acc))
knn_acc; max(knn_acc);
```

```
##         1         2         3         4         5         6         7
## 0.5000000 0.5192308 0.5480769 0.5961538 0.5192308 0.5384615 0.5384615
##         8         9        10
## 0.5384615 0.5480769 0.5769231
```

```
## [1] 0.5961538
```

```
# More on accuracy as a result of increase of K
knn_acc <- c()
for (i in c(1:100)){
knn.pred = knn(train.Lag2, test.Lag2, train$Direction, k=i)
m <-mean(knn.pred == test$Direction)
knn_acc <- c(knn_acc, m)}
plot(knn_acc, xlab="K", ylab="Accuracy")
```




```
library(ggplot2); knn_acc<-data.frame(knn_acc);
ggplot(data = knn_acc) +
  geom_point(mapping = aes(x = c(1:length(knn_acc)), y = knn_acc)) +
```

```
  xlab("K") + ylab("Accuracy")
```

Based on the work presented so far, logistic regression and LDA ($p = 1$) provide similar and better test error rates and accuracy than QDA and KNN ($K = 1$).

Question 11