1. Consider the multiple regression model \( Y_i = \mathbf{X}_i^T \beta + \varepsilon_i \), where \( \varepsilon_i \sim N(0, \sigma^2) \). Show that the maximum likelihood estimator is equivalent to the least-squares estimator, which finds \( \hat{\beta} \) to minimize
\[
\sum_{i=1}^{n} (Y_i - \mathbf{X}_i^T \hat{\beta})^2
\]
and
\[
\hat{\sigma} = \sqrt{\frac{\text{RSS}}{n}},
\]
where \( \text{RSS} = \sum_{i=1}^{n} (Y_i - \mathbf{X}_i^T \hat{\beta})^2 \) and \( \hat{\beta} \) is the least-squares solution.

2. Let us consider the 129 macroeconomic time series again as described in the lecture notes. Let \( Y_t = \log(\text{PCE}_t) \) be the personal consumption expenditure. Let us take
\[
X_{t,1} = \log(\text{PCE}_{t-1}), \quad X_{t,2} = \text{Unrate}_{t-1}, \quad X_{t,3} = \Delta \log(\text{IndPro}_t), \quad X_{t,4} = \Delta \log(\text{M2Real}_t),
\]
\[
X_{t,5} = \Delta \log(\text{CPI}_t), \quad X_{t,6} = \Delta \log(\text{SPY}_t), \quad X_{t,7} = \text{HouSta}_t, \quad X_{t,8} = \text{FedFund}_t
\]
Let us again take the last 10 years data as testing set and remaining as training set. Conduct a similar analysis as those in the lecture notes. Answer in particular the following questions.

(a) What are \( \hat{\sigma}^2 \), adjusted \( R^2 \) and insignificant variables?

(b) Now perform the stepwise deletion, eliminating one least significant variable at a time (by looking at the small \( |t| \)-statistic) until all variables are statistically significant. Let us call this model as model \( \hat{\mathcal{M}} \). (The function \texttt{step} can do the job automatically, but we ask you to do manually to gain the insights)

(c) Using model \( \hat{\mathcal{M}} \), what are root mean-square prediction error and mean absolute deviation prediction error for the test sample?

(d) Compute the standardized residuals. Present the time series plot of the residuals, fitted values versus the standardized residuals, and QQ plot for the standardized residuals.

3. Download the Boston housing data from the class website. Let \( Y \) be the median housing values and \( X \) be the remaining variables. The following R-code help you gets started.

```r
boston = scan("boston.housing.txt",skip=19) # skip 19 lines
boston = matrix(boston, ncol=14, byrow=T) # put in a matrix form
dimnames(boston) = list(NULL, c("crim", "zn", "indus", "chas", "nox","rm", "age", "dist", "rad", "tax", "ptratio", "b", "lstat", "medv"))
## name the column but not the rows
boston = data.frame(boston)
attach(boston)
x = boston[,1:13]
y= boston[,14]
```
(a) Create additional variables $X_{13} = rm^2$, $X_{14} = rm * dis$. Further, instead of using the variable "zn" directly, we define two indicators $X_{15} = I(zn == 0)$, $X_{16} = I(zn > 30)$. These variables are created in order to illustrate the concept of model building.

(b) Fit the multiple regression model using all the data with 16 variables. Furnish necessary diagnostic plots.

(c) Use the five fold cross-validation to estimate the prediction error (rMSE and MADE).

4. Fit the motorcycle data using quadratic splines with the basis functions:

$$B_1(x) = x, B_2(x) = x^2, \quad B_{2+i}(x) = \begin{cases} (x - t_i)^2 & \text{if } x \geq t_i \\ 0 & \text{otherwise} \end{cases}$$

where the knots $\{t_i\} = \text{seq}(5, 40, \text{by}=5)$.

5. Upright Human Detection in Photos

Go to the instructor’s class website, download the image data pictures.zip and its associated preliminary codes human.r. In this problem, we are going to create a human detector that tells us whether there is a upright human in a given photo. We treat this as a classification problem with two classes: having humans or not in a photo. You are provided with two datasets POS and NEG that have photos with and without upright humans respectively.

(a) Load Pictures and Extract Features (The code has already written for you)

The tutorial below that explains the data loading and feature extraction in human.r. If you do not want to read, you can just execute the code to get the extracted features. Remember to install the package png by using install.packages("png") and to change the working directory to yours.

i. Install the package png by using install.packages("png"), and use the function readPNG to load photos. The function readPNG() will return the grayscale matrix of the picture.

ii. Use the function grad to obtain the gradient field of the central $128 \times 64$ part of the grayscale matrix.

iii. Use the function hog (Histograms of Oriented Gradient) to extract a feature vector from the gradient field obtained in the previous step. Your feature vector should have 96 components. Please see the appendix for parameter configuration of this function.
Figure 1: Illustration of feature extraction for a positive example from POS.

(b) Logistic Regression (You are responsible for writing the code).

In this question, we will apply logistic regression to the data and use the cross validation to test the classification accuracy of the fitted model. Please finish the following steps:

i. Set the random seed to be 245, i.e., type in `set.seed(245)` in your code. Randomly divide your whole data into five parts. Let the first four parts be the training data and the rest be your testing data.

ii. Feed the training data to the function `glm` and store the fitted model as `fitted1` in R.

iii. Use the function `predict` to apply your fitted model `fitted1` to do classification on the testing data and report the misclassification rate (testing error).

iv. Let us now select the features using the stepwise selection, by issuing the R-function `step(fitted1)`. Let us call the selected model (the last model in the output) `fitted2`. Report the model `fitted2` and the misclassification rate (testing errors) of `fitted2`.

Appendix

A. Histogram of Oriented Gradient

Here we give a brief introduction of what `hog(xgrad, ygrad, hn, wn, an)` does. First of all, it uniformly partitions the whole picture into `hn*wn` small parts with `hn` partitions on the height and `wn` partitions on the width. For each small part, it counts the gradient direction whose angle falls in the intervals \([0, 2\pi/an), [2\pi/an, 4\pi/an), ..., [2(an-1)\pi/an, 2\pi)\) respectively. So `hog` can get \(an\) frequencies for each small picture. Applying the same procedure to all the small parts, `hog` will have `hn*wn*an` frequencies that constitute the final feature vector for the given gradient field.
B. Useful Functions

(a) \texttt{crop.r}(X, h, w) randomly crops a sub-picture that has height $h$ and width $w$ from $X$. The output is therefore a sub-matrix of $X$ with $h$ rows and $w$ columns.

(b) \texttt{crop.c}(X, h, w) crops a sub-picture that has height $h$ and width $w$ at the center of $X$. The output is therefore a sub-matrix of $X$ with $h$ rows and $w$ columns. This function helps the \texttt{hog(...)} function. For the cropping in your assignment, use \texttt{crop.r()}. 

(c) \texttt{grad}(X, h, w, \texttt{pic}) yields the gradient field at the center part of the given grayscale matrix $X$. The center region it examines has height $h$ and width $w$. It returns a list of two matrices \texttt{xgrad} and \texttt{ygrad}. The parameter \texttt{pic} is a boolean variable. If it is \texttt{TRUE}, the generated gradient filed will be plotted. Otherwise the plot will be omitted.

(d) \texttt{hog(xgrad, ygrad, hn, wn, an)} returns a feature vector in the length of $hn*wn*an$ from the given gradient field. \((xgrad[i,j], ygrad[i,j])\) gives the grayscale gradient at the position \((i,j)\). $hn$ and $wn$ are the partition number on height and width respectively. $an$ is the partition number on the angles (or the interval \([0,2\pi]\) equivalently).