Computational Assignment 5:
Logistic Regression and k Nearest Neighbors

In this assignment, we will explore how to use two powerful binary classification tools in the R environment: logistic regression and k Nearest Neighbors (kNN). We will apply both of these methods on a real dataset from the UCI Machine Learning repository. From the description: “The dataset contains cases from a study that was conducted between 1958 and 1970 at the University of Chicago’s Billings Hospital on the survival of patients who had undergone surgery for breast cancer.” Each patient has 4 associated variables: 1) age, 2) year of surgery (year - 1900), 3) number of positive axillary nodes detected, and 4) survival time: $1 = < 5$ years after surgery and $2 = \geq 5$ years after surgery. For more information about this dataset, please visit https://archive.ics.uci.edu/ml/datasets/Haberman’s+Survival. Our aim is to use variables (1) - (3) to classify long term and short term survivors.

1. Load and summarize the data: First load the data using the following commands:

```R
#Load the data

#Label the data with appropriate label names
names(survival.data) = c("Age", "Year", "Num.Axillary", "Survival")

Let's split the dataset (randomly) into a training and test set where the training data is 75% of our observed data. Do this using the following code:

```R
#find number of samples
num.samples = dim(survival.data)[1]

#Obtain a random sample of 75% of the data
rand.sample = sample(1:num.samples, floor(.75*num.samples), replace = FALSE)

#split into training and test data and labels
training.data = data.frame(Age = survival.data$Age, Year = survival.data$Year, Axillary = survival.data$Num.Axillary)
training.labels = survival.data$Survival
test.data = data.frame(Age = test.data$Age, Year = test.data$Year, Axillary = test.data$Num.Axillary)
test.labels= test.data$Survival

Note that the training.data and test.data data frames contain the three variables we want to use for prediction of survival. On the other hand, training.labels and test.labels contain the binary survival labels for each patient.
Using simple R exploratory tools mentioned in previous computational assignments, answer the following questions about the dataset. Be sure to provide the code used to answer these questions in your output.

**Questions**

(a) How many patients are there in the study? How many patients survived 5 or more years after surgery?

(b) What is the average and standard deviation of the age of all of the patients? What is the average and standard deviation of the age of those patients that survived 5 or more years after surgery? What is the average and standard deviation of the age of those patients who survived fewer than 5 years after surgery?

(c) Plot a histogram of the ages side by side with a histogram of the number of axillary nodes. For each histogram, set “n = 100” so that you show 100 bars.

(d) What is the earliest year of surgery in this dataset? What is the most recent year of surgery?

2. **Fitting a Logistic Regression:** Logistic regression falls under the framework of generalized linear models (glm). In particular, logistic regression is used on binary response data as a means of classification. In R, glmns are fit using the `glm()` function. In particular, when using this function we must specify what family, or distribution, the response data takes. For logistic regression, we must use `glm(object, family = binomial)`. Here, the object argument is the regression written $y \sim x$ where $y$ is the binary variable and $x$ is the data you want to use for regression. Fit a logistic regression on the training data using the following code:

```R
logistic.model = glm(as.factor(training.labels) ~ Age + Year + Axillary, data = training.data, family = binomial)
```

The `logistic.model` now contains the coefficients associated with the logistic regression performed on the training data. Get a summary of the fit using `summary(logistic.model)`. The output table of this command shows the estimated coefficients and p-values associated with the test of statistical significance of the coefficients for each variable. We will ask questions about this later in this problem.

Prediction in R requires the use of the `predict(object, data, type)` function. This function has the following arguments: `object` = a model that has been fit to previous data (can be a glm, lm, etc.); `data` = new data for which the labels will be predicted; `type` = type of prediction to be made. Predict the probability of each datum in the test set taking value 2 using the following command:

```R
logistic.prediction = predict(logistic.model, test.data, type = "response")
```

#Classify the test data labels by choosing 2 if the probability > 1/2 and 1 otherwise
```R
logistic.classification = rep(0,length(logistic.prediction)) #initialize a vector
logistic.classification[which(logistic.prediction > .5)] = 2
logistic.classification[which(logistic.prediction < .5)] = 1
```

Finally, we want to measure how well logistic regression classified the test data. To obtain a performance measure, calculate the proportion of misclassified labels with the following code:

```R
misclass.logistic = sum(abs(logistic.classification - test.labels))/dim(test.labels)[1]
```

**Questions:**

(a) Comment on the summary of the logistic regression that you fit above. Which variables have statistically significant coefficients? Which variables do not? Do you think that insignificant variables could affect the results of the classification? Why or why not?
(b) What is the misclassification proportion associated with the test set?

3. **Running kNN**: To run kNN in R, we must first install the package `class`. Do this by typing `require(class)`. (NOTE: if this doesn’t work, use `install.packages("class")` and `library(class)` instead. It depends on the version of R you have installed.) To run kNN we can simply use the function `knn(training, test, k, cl)` where `training` is the training data, `test` is the test data, `k` is the number of neighbors to use, and `cl` is the true labels of the training data. Of course, we can use any number `k` that we want. One way to choose an appropriate `k` is to minimize the proportion of misclassification on the training data. So let’s run kNN for `k = 1, \ldots, 100` and check the misclassification error of each run on the training data. Do this with the following code:

```r
misclass = rep(0,100)
for(i in 1:100){
  knn.class = knn(training.data, training.data, k = i, cl = as.factor(training.labels))
  misclass[i] = sum(abs(as.numeric(knn.class) - test.labels))/dim(test.labels)[1]
}
min.k = which.min(misclass)[1]
```

Using the “best” `k` according to the above training data, run kNN now on the test data and calculate the misclassification proportion. Then answer the following questions:

(a) Plot the misclassification proportion error on the training data across `k = 1, \ldots, 100`. Comment on distinguishing features of this plot. Is there a single "best" `k` to this curve? If not, which did we choose in the above code?

(b) Calculate the misclassification proportion error associated with the "best" `k` kNN run on the test data. Which classification tool, logistic regression or kNN performed better in this setting? Note that in real life applications you would not have access to the true labels of the data that you want to classify. In this situation, you would have to compare and contrast these two methods before a decision is made.