Assignment 6: Classification and prediction interval

Akshat Valse

Due Saturday, Aug. 12 at 8am

YOU MAY NOT USE CHATGPT FOR THIS ASSIGNMENT.

Please upload the `PDF` that you obtain by knitting the `Rmd` file that contains your `R` code and your text answering other questions. So this uploaded file will also show any output that `R` produces in addition to your code.

You should not touch the starter code as it will print out the necessary data frames and results for grading purposes.

```r
set.seed(123)
```

Part 1 Logistic Regression

So far, we have considered situations in which the response Y is continuous. However, many responses in practice fall under the binary (e.g. yes/no) framework. Some examples include:

- medical: presence or absence of a disease
- financial: bankrupt or solvent
- industrial: passes a quality control test or not

For binary outcomes, our task is **classification** as opposed to **regression**. This is because we are seeking to classify, based on input features, the class that the outcome belongs to.

One way we can model binary data is using **logistic regression**. Logistic regression actually belongs to a broader class of models called the **Generalized Linear Model** which you can read more about here. But for now, we will focus on logistic regression alone.

For this problem, you will analyze the some flu shot data. A local health clinic sent fliers to its clients to encourage everyone, but especially older persons at high risk of complications, to get a flu shot in time for protection against an expected flu epidemic.

In a pilot follow-up study, 50 clients were randomly selected and asked whether they actually received a flu shot.

Let the response Y = shot status $\in \{0, 1\}$. In addition, data were collected on their participants age and health awareness. Let X_1 = age and X_2 = health awareness.

Let us first read in the data:

```r
flu_data = read.table('http://stats191.stanford.edu/data/flu.table',
                      header=TRUE)
flu_data
```

```
##    Shot Age Health.Aware
## 1     0  38           40
## 2     1  52           60
## 3     0  41           36
## 4     1  46           59
## 5     1  41           70
## 6     0  43           49
## 7     1  57           59
## 8     0  34           50
## 9     0  31           48
## 10    1  49           59
## 11    1  53           49
## 12    0  54           48
## 13    0  63           40
## 14    1  39           59
## 15    0  38           45
## 16    0  28           58
## 17    0  42           61
## 18    1  53           69
## 19    0  36           71
## 20    0  45           41
## 21    0  47           47
## 22    1  49           70
## 23    0  53           32
## 24    0  42           42
## 25    1  49           50
## 26    0  42           32
## 27    1  46           68
## 28    0  48           33
## 29    1  54           71
## 30    0  46           47
## 31    1  63           43
## 32    0  44           27
## 33    1  56           57
## 34    1  64           70
## 35    0  46           44
## 36    0  35           44
## 37    1  52           61
## 38    1  46           43
## 39    0  40           64
## 40    1  57           54
## 41    1  56           57
## 42    0  40           64
## 43    0  64           34
## 44    1  46           56
## 45    0  34           54
## 46    0  38           45
## 47    1  47           56
## 48    0  56           46
## 49    0  45           35
## 50    0  33           45
```

A possible model for our data is the following:

$$\pi(X_1, X_2) = \frac{\exp(\beta_0 + \beta_1 X_1 + \beta_2 X_2)}{1 + \exp(\beta_0 + \beta_1 X_1 + \beta_2 X_2)}$$

This automatically enforces $0 \le \mathbb{E}[Y] = \pi(X_1, X_2) \le 1$. We need to make such a enforcement because in the case with binary outcomes Y, $\mathbb{E}[Y]$ is the probability that $Y = 1$ (i.e. received follow up shot). Now if you recall that a valid probability must be between 0 and 1. Thus, we need to have $0 \le \mathbb{E}[Y] \le 1$.

An astute reader will also notice among the wall of text that we have modeled

$$\mathbb{E}[Y] = \pi(X_1, X_2)$$

Let us rewrite the above line as the following:

$$\mathbb{E}[Y] = \pi(X_1, X_2)$$

so

$$\pi^{-1}(\mathbb{E}[Y]) = \beta_0 + \beta_1 X_1 + \beta_2 X_2$$

This modeling choice is not unique to the logistic regression. For all models that belong to the Generalized Linear Model class, you will notice this theme that we model the transformed mean of the outcome Y by some linear function of the predictors, meaning $\pi^{-1}(\mathbb{E}[Y]) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$ (if we have p features).

Exercise 1.1

Derive the inverse of the sigmoid function:

Let $\pi(x) = \frac{\exp(x)}{1+\exp(x)}$. Find $\pi^{-1}(x)$. Show your derivations. You can attach a photo of your derivations to the outputted pdf file by `RMarkdown`.

Hint. Divide the numerator and denominator by e^x$ and rewrite the sigmoid function before you compute the inverse. Let pi(x) = exp(x) / (1 + exp(x)) Find pi^-1(x)

Start with y = exp(x) / (1 + exp(x)) Solve for x: x = log(y / (1 - y)) Thus, pi^-1(x) = log(x / (1 - x))

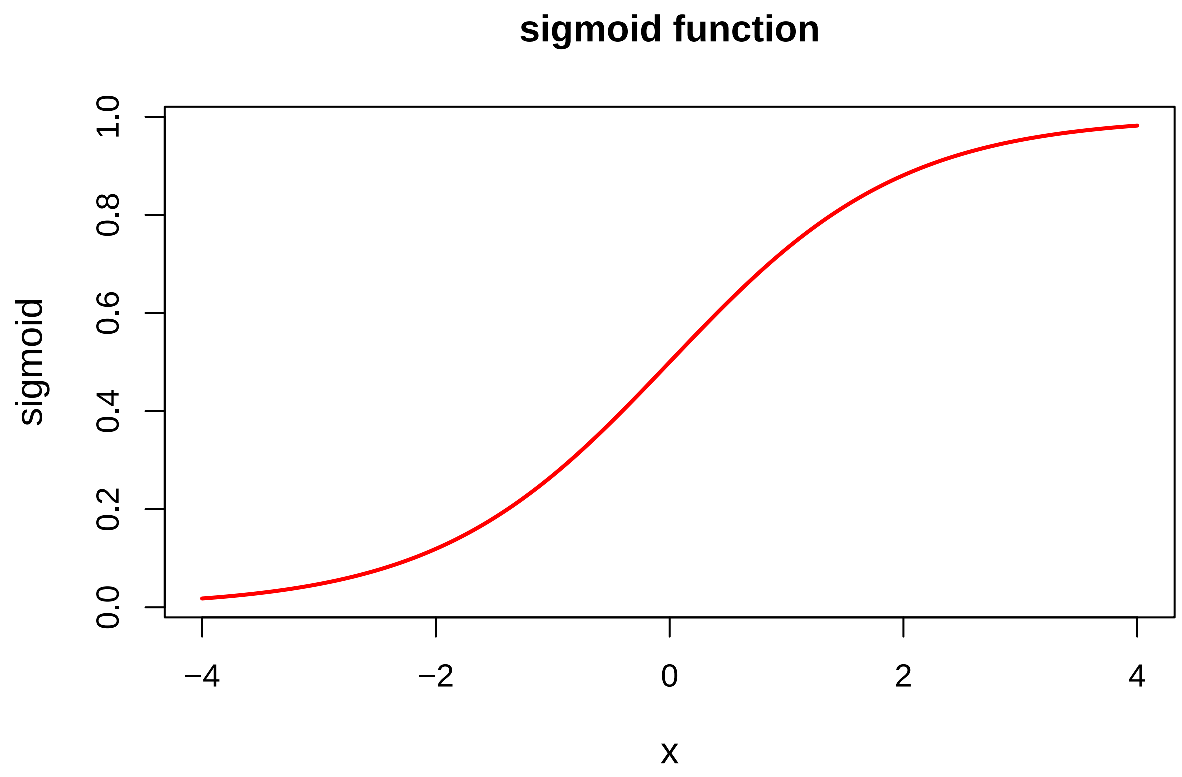
Exercise 1.2

We will first visualize the sigmoid function. Write a function that returns the output of the sigmoid function for a given input value.

```r
x = seq(-4, 4, length=200)

### YOUR CODE HERE
sigmoid = function(x) {
    return(exp(x) / (1 + exp(x)))
}
### END OF YOUR CODE

plot(x, sigmoid(x), lwd=2, type='l', col='red', cex.lab=1.2, ylab='sigmoid', main='sigmoid function')
```



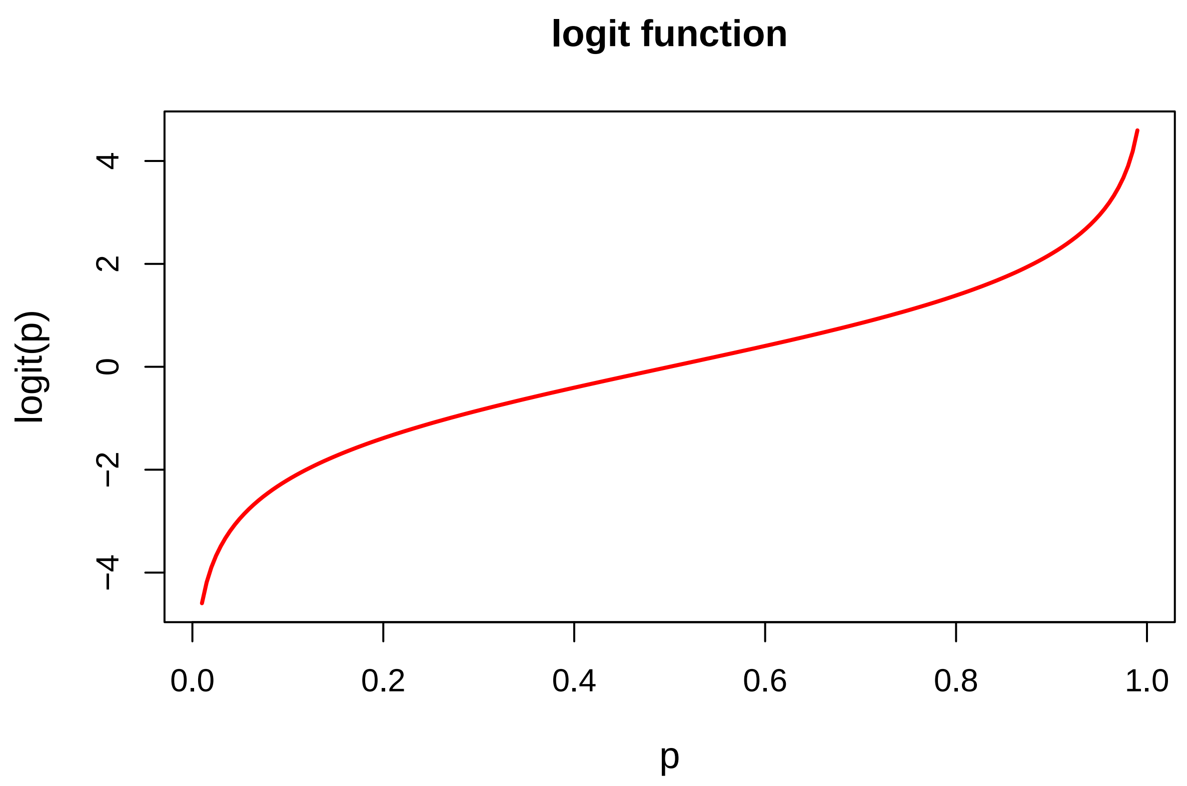
Exercise 1.3

We will now visualize the inverse sigmoid function (called logit function) that you computed in Exercise 1.0. Write a function that returns the output of the logit function for a given input value.

```
x = seq(-4, 4, length=200)

### YOUR CODE HERE
logit = function(p) {
    return(log(p / (1 - p)))
}
### END OF YOUR CODE

p = seq(0.01,0.99,length=200)
plot(p, logit(p), lwd=2, type='l', col='red', cex.lab=1.2, ylab='logit(p)', main='logit function')
```



Exercise 1.4

Before we perform logistic regression, we will first partition the dataset into a training set and a testing set. Using 70-30 partition, partition your data into a training set and test set below:

```
### YOUR CODE HERE
train_indices = sample(nrow(flu_data), 0.7 * nrow(flu_data))
train_data = flu_data[train_indices, ]
test_data = flu_data[-train_indices, ]
### END OF YOUR CODE
```

Exercise 1.5

We will now train a logistic regression model on the training set **alone**. You should not touch the test set at this point. To do so, we will utilize a very useful package called `glmnet`. You should first install `glmnet` if you haven't done so. The function that we will use to run the logistic regression is called `glm`.

```
require(glmnet)
```

```
## Loading required package: glmnet

## Loading required package: Matrix

## Loaded glmnet 4.1-7
```

```
?glm
```

```
## starting httpd help server ...
```

```
##  done
```

```
### YOUR CODE HERE
model = glm(Shot ~ Age + Health.Aware, data = train_data, family = "binomial")
### END OF YOUR CODE
```

Exercise 1.6

Now, use your model to predict the outcome for each of the individuals in the test set. Report the confusion matrix for the test data. To do so, you will use the function `confusionMatrix` from the `caret` library. You will have to install the `caret` packet if you haven't already done so.

```
require(caret)
```

```
## Loading required package: caret
```

```
## Loading required package: ggplot2
```

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

```
?confusionMatrix

### YOUR CODE HERE
train_data$Shot = factor(train_data$Shot, levels = c(0, 1))
test_data$Shot = factor(test_data$Shot, levels = c(0, 1))
predictions = predict(model, newdata = test_data, type = "response")

predicted_classes = ifelse(predictions > 0.5, 1, 0)

confusionMatrix(factor(predicted_classes, levels = c(0, 1)), test_data$Shot)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 0 1
##          0 7 3
##          1 1 4
##
##                Accuracy : 0.7333
##                  95% CI : (0.449, 0.9221)
##     No Information Rate : 0.5333
##     P-Value [Acc > NIR] : 0.09638
##
##                   Kappa : 0.4545
##
##  Mcnemar's Test P-Value : 0.61708
```

```
##
##             Sensitivity : 0.8750
##             Specificity : 0.5714
##          Pos Pred Value : 0.7000
##          Neg Pred Value : 0.8000
##              Prevalence : 0.5333
##          Detection Rate : 0.4667
##    Detection Prevalence : 0.6667
##       Balanced Accuracy : 0.7232
##
##        'Positive' Class : 0
##
```

```
### END OF YOUR CODE
```

Exercise 1.7

Report and state the confidence intervals for the coefficients using the `confint` function.

```
### YOUR CODE HERE
conf_intervals <- confint(model)
```

```
## Waiting for profiling to be done...
```

```
conf_intervals
```

```
##                    2.5 %      97.5 %
## (Intercept)  -59.9523218 -12.3332203
## Age            0.1017890   0.6133106
## Health.Aware   0.1141718   0.5609247
```

```
### END OF YOUR CODE
```