**Problem 1**

This question should be answered using the Default data set. In Chapter 4 on classification, we used logistic regression to predict the probability of default using income and balance. Now we will estimate the test error of this logistic regression model using the validation set approach. Do not forget to set a random seed before beginning your analysis.

(a) Fit a logistic regression model that predicts default using income and balance.

(b) Using the validation set approach, estimate the test error of this model. You need to perform the following steps:

i. Split the sample set into a training set and a validation set.
ii. Fit a logistic regression model using only the training data set.
iii. Obtain a prediction of default status for each individual in the validation set using a threshold of 0.5.
iv. Compute the validation set error, which is the fraction of the observations in the validation set that are misclassified.

(c) Repeat the process in (b) three times, using three different splits of the observations into a training set and a validation set. Comment on the results obtained.

(d) Consider another logistic regression model that predicts default using income, balance and student (qualitative). Estimate the test error for this model using the validation set approach. Does including the qualitative variable student lead to a reduction of test error rate?

Ans 1:

(a).

```
> library(ISLR)
Warning message:
package 'ISLR' was built under R version 3.4.3
> attach(Default)
> set.seed(1)
> fit.glm <- glm(default ~ income + balance, data = Default, family = "binomial")
> summary(fit.glm)


Call:
glm(formula = default ~ income + balance, family = "binomial",
    data = Default)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.4725  -0.1444  -0.0574  -0.0211   3.7245

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.154e+01  4.348e-01 -26.545  < 2e-16 ***
income       2.081e-05  4.985e-06   4.174 2.99e-05 ***
balance      5.647e-03  2.274e-04  24.836  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2920.6  on 9999  degrees of freedom
Residual deviance: 1579.0  on 9997  degrees of freedom
AIC: 1585

Number of Fisher Scoring iterations: 8
```

(b).

(i).
```
> train <- sample(dim(Default)[1], dim(Default)[1] / 2)
```

(ii).

```
> fit.glm <- glm(default ~ income + balance, data = Default, family = "binomial", subset = train)
> summary(fit.glm)

Call:
glm(formula = default ~ income + balance, family = "binomial",
    data = Default, subset = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3583  -0.1268  -0.0475  -0.0165   3.8116

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.208e+01  6.658e-01 -18.148   <2e-16 ***
income       1.858e-05  7.573e-06   2.454   0.0141 *
balance      6.053e-03  3.467e-04  17.457   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1457.0  on 4999  degrees of freedom
Residual deviance:  734.4  on 4997  degrees of freedom
AIC: 740.4

Number of Fisher Scoring iterations: 8
```

(iii).

```
> probs <- predict(fit.glm, newdata = Default[-train, ], type = "response")
> pred.glm <- rep("No", length(probs))
> pred.glm[probs > 0.5] <- "Yes"
>
```

(iv).

```
> mean(pred.glm != Default[-train, ]$default)
[1] 0.0286
```

Using the validation test approach, we get an error of 2.86% on this data set.

(c ).

```
> train <- sample(dim(Default)[1], dim(Default)[1] / 2)
> fit.glm <- glm(default ~ income + balance, data = Default, family = "binomial", subset = train)
> probs <- predict(fit.glm, newdata = Default[-train, ], type = "response")
> pred.glm <- rep("No", length(probs))
> pred.glm[probs > 0.5] <- "Yes"
> mean(pred.glm != Default[-train, ]$default)
[1] 0.0236
> train <- sample(dim(Default)[1], dim(Default)[1] / 2)
> fit.glm <- glm(default ~ income + balance, data = Default, family = "binomial", subset = train)
> probs <- predict(fit.glm, newdata = Default[-train, ], type = "response")
> pred.glm <- rep("No", length(probs))
> pred.glm[probs > 0.5] <- "Yes"
> mean(pred.glm != Default[-train, ]$default)
[1] 0.028
> train <- sample(dim(Default)[1], dim(Default)[1] / 2)
> fit.glm <- glm(default ~ income + balance, data = Default, family = "binomial", subset = train)
> probs <- predict(fit.glm, newdata = Default[-train, ], type = "response")
> pred.glm <- rep("No", length(probs))
> pred.glm[probs > 0.5] <- "Yes"
> mean(pred.glm != Default[-train, ]$default)
[1] 0.0268
>
```

Here, we can observe that the test error rates are variable in all the 3 cases which means that the test error rate depends on the choice of training data set and the validation data set.

(d).

```
> train <- sample(dim(Default)[1], dim(Default)[1] / 2)
> fit.glm <- glm(default ~ income + balance + student, data = Default, family = "binomial", subset = train)
> pred.glm <- rep("No", length(probs))
> probs <- predict(fit.glm, newdata = Default[-train, ], type = "response")
> pred.glm[probs > 0.5] <- "Yes"
> mean(pred.glm != Default[-train, ]$default)
[1] 0.0264
>
```

Adding a student dummy variable does not reduce the test error rate.

**Problem 2**

This question requires performing cross validation on a simulated data set.

(a) Generate a simulated data set as follows:

```
set.seed(1)
x=rnorm(200)
y=x-2*x^2+rnorm(200)
```

In this data set, what is $n$ and what is $p$? Write out the model used to generate the data in equation form (i.e., the true model of the data).

(b) Create a scatter plot of $Y$ vs $X$. Comment on what you find.

(c) Consider the following four models for the data set:

i. $Y = \beta_0 + \beta_1 X + \epsilon$
ii. $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \epsilon$
iii. $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \epsilon$
iv. $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \beta_4 X^4 + \epsilon$

Compute the LOOCV errors that result from fitting these models.

(d) Repeat (c) using another random seed, and report your results. Are your results the same as what you got in (c)? Why?

(e) Which of the models in (c) had the smallest LOOCV error? Is this what you expected? Explain your answer.

(f) Now we use 5-fold CV for the model selection. Compute the CV errors that result from fitting the four models. Which model has the smallest CV error? Are the results consistent with LOOCV?

(g) Repeat (f) using 10-fold CV. Are the results the same as 5-fold CV?

Ans 2:

(a).

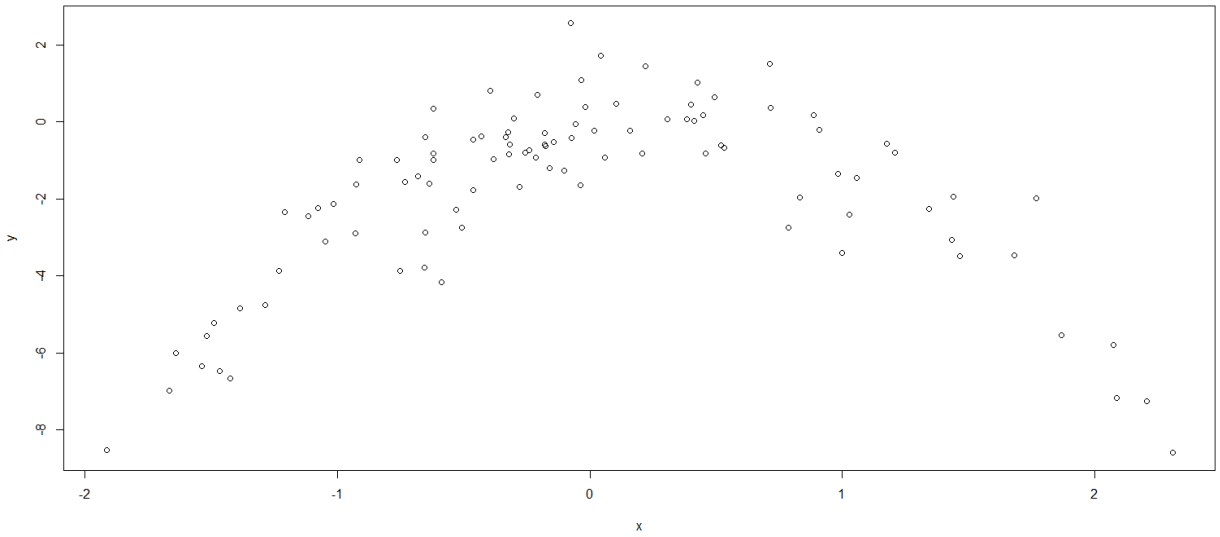
```
> set.seed(1)
> y <- rnorm(100)
> x <- rnorm(100)
> y <- x - 2 * x^2 + rnorm(100)
>
```

Here, the value of n is 100 and p is 2.

The model used here is $Y = \beta_0 + \beta_1 X + \epsilon$

(b).

```
> plot(x,y)
```




We can see that as the value of x increases, the value of y goes on to increase until the middle of the graph and the value of y decreases with more increase in x.

(c ).

(i).

```
> library(boot)
> set.seed(1)
> Data <- data.frame(x, y)
> fit.glm.1 <- glm(y ~ x)
> cv.glm(Data, fit.glm.1)$delta[1]
[1] 5.890979
>
```

(ii).

```
> fit.glm.2 <- glm(y ~ poly(x, 2))
> cv.glm(Data, fit.glm.2)$delta[1]
[1] 1.086596
```

(iii).

```
> fit.glm.3 <- glm(y ~ poly(x, 3))
> cv.glm(Data, fit.glm.3)$delta[1]
[1] 1.102585
```

(iv).

```
> fit.glm.4 <- glm(y ~ poly(x, 4))
> cv.glm(Data, fit.glm.4)$delta[1]
[1] 1.114772
```

(d).

```
> set.seed(10)
> fit.glm.1 <- glm(y ~ x)
> cv.glm(Data, fit.glm.1)$delta[1]
[1] 5.890979
> fit.glm.2 <- glm(y ~ poly(x, 2))
> cv.glm(Data, fit.glm.2)$delta[1]
[1] 1.086596
> fit.glm.3 <- glm(y ~ poly(x, 3))
> cv.glm(Data, fit.glm.3)$delta[1]
[1] 1.102585
> fit.glm.4 <- glm(y ~ poly(x, 4))
> cv.glm(Data, fit.glm.4)$delta[1]
[1] 1.114772
```

LOOCV stands for Leave One Out Cross Validation which evaluates n folds of a single observation. Therefore, even when we consider random seed, it does not affect LOOCV error.

(e ).

The relation between y and x in the 2nd case is quadratic as a result, we go on to observe the least test error for fit.glm.2 and is hence not surprising.

(f).

```
> library(boot)
> set.seed(1)
> cv.error.5 = rep(0,4)
> for (i in 1:4)
+ {
+ fit.glm = glm(y~poly(x,i),data=Data)
+ cv.error.5[i] = cv.glm(Data,fit.glm,K=5)$delta[1]
+ }
> cv.error.5
[1] 5.816178 1.099735 1.102200 1.089778
>
```

The model with polynomial of degree 4 has the smallest CV error for k=5. This is not consistent with LOOCV.

(g).

```
> library(boot)
> set.seed(1)
> cv.error.10 = rep(0,4)
> for (i in 1:4)
+ {
+   fit.glm = glm(y~poly(x,i),data=Data)
+   cv.error.10[i] = cv.glm(Data,fit.glm,K=10)$delta[1]
+ }
> cv.error.10
[1] 5.814726 1.087184 1.123487 1.099495
>
```

The model with polynomial of degree 3 has the smallest CV error for k=10. This is not consistent with the results obtained with k=5.