ISEN 613 - Assignment #4: Cross Validation

SANDEEP DASARI **UIN - 829002252**

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.

```
> library(ISLR)
```

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

```
> set.seed(1)
> fit.glm = glm(default ~ income+balance, family = "binomial", data = Default)
> 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) 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.

```
> train =  sample(dim(Default)[1], dim(Default)[1]/2)
```

ii. Fit a logistic regression model **using only the training data set.**

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

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5830  -0.1428  -0.0573  -0.0213   3.3395

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.194e+01  6.178e-01 -19.333  < 2e-16 ***
income       3.262e-05  7.024e-06   4.644 3.41e-06 ***
balance      5.689e-03  3.158e-04  18.014  < 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: 1523.8  on 4999  degrees of freedom
Residual deviance:  803.3  on 4997  degrees of freedom
AIC: 809.3
Number of Fisher Scoring iterations: 8
```

iii. Obtain a prediction of default status for each individual in the validation set using a threshold of 0.5.

```
> glm_prob = predict(fit.glm, newdata = Default[-train, ],type = "response")
> glm_pred = rep("No",5000)
> glm_pred[glm_prob > 0.5] = "Yes"
> glm_pred
  [1] "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
      "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
 [37] "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
      "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
 [73] "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "Yes" "No"  "No"  "No"  "No"  "No"  "No"
      "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
[109] "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
      "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
[145] "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
      "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "Yes" "No"
[181] "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
      "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
[217] "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
      "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
[253] "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
      "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
[289] "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
      "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
[325] "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
      "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "Yes"
[361] "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
      "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
[397] "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
      "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
```

```
[433] "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
    "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
[469] "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
    "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
[505] "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
    "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
[541] "No"  "Yes" "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
    "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
[577] "Yes" "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
    "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
[613] "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "Yes" "No"  "No"  "No"  "No"
    "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
[649] "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
    "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
[685] "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
    "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
[721] "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "Yes" "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
    "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
[757] "No"  "No"  "No"  "No"  "No"  "No"  "Yes" "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
    "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
[793] "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "Yes" "No"  "No"
    "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "Yes" "No"  "No"  "No"
[829] "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
    "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
[865] "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
    "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
[901] "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
    "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
[937] "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
    "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
[973] "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
    "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
 [ reached getOption("max.print") -- omitted 4000 entries ]
```

iv. Compute the validation set error, which is the fraction of the observations in the validation set that are misclassified.

```
> mean(glm_pred != Default[-train, ]$default)
[1] 0.0254
```

(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.

```
> 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] 0.0274

> 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] 0.0244

> 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] 0.0244
```

⇒ *The test error rate estimate changes with the observations included in training set and validation set.*

(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?

```
> 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] 0.0278
```

⇒ Yes, when we add the variable Student which is qualitative, the validation set test error rate decreases.

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).

```
> set.seed(4)
> x=rnorm(200)
> y=x-2*x^2+rnorm(200)
```

```
> p = 2
> n = 200
> simdata = data.frame(x,y)
> simdata
              x           y
1    0.2167548629  1.33751967
2   -0.5424925723 -2.67888922
3    0.8911446451 -0.99937894
4    0.5959805772  0.92480260
5    1.6356180011 -4.48271622
6    0.6892754419  1.26374675
7   -1.2812466301 -6.98651979
8   -0.2131445193  0.25232263
9    1.8965398719 -4.19165319
10   1.7768632137 -4.37118573
11   0.5666044982 -0.30093911
12   0.0157194540 -0.21318708
13   0.3830573385 -0.16359774
14  -0.0451371159  2.01906150
15   0.0343519074  1.61518166
16   0.1690267742 -0.93070407
17   1.1650268390 -1.55793563
18  -0.0442039973 -1.38219988
19  -0.1003684426  0.02601994
20  -0.2834445689 -1.23194964
21   1.5408149809 -3.49640423
22   0.1651690197  0.77744671
23   1.3076223603 -2.24876133
24   1.2882568779 -1.80686538
25   0.5928969406  1.00705004
26  -0.2829436843 -1.88155234
27   1.2558840256 -1.26423962
28   0.9098391512 -1.24497021
29  -0.9280281051 -1.52422800
30   1.2401808380 -1.93610087
```

31	0.1534641796	-1.00348035
32	1.0519325790	-0.50326746
33	-0.7542112128	-1.93517633
34	-1.4821891197	-5.24756907
35	0.8611318725	-1.55879354
36	-0.4045198308	-1.08917582
37	-0.2274054173	-0.37590478
38	0.9340961709	-1.15902900
39	-0.4658958798	-1.25898773
40	-0.6375434986	-1.05485168
41	1.3437086262	-3.25134128
42	0.1815353846	0.09292463
43	1.2925123364	-2.93321415
44	-1.6880485759	-6.90202413
45	-0.8209935776	-3.00058511
46	-0.8621461441	-1.17558727
47	0.0988436891	0.14031900
48	-0.3756551442	-0.50113510
49	0.7239041553	0.14473546
50	-1.7973820186	-7.71338546
51	-0.6637431416	-1.14925089
52	-0.6237264887	-2.32315930
53	-0.0796324318	-0.76888769
54	0.4356247628	-1.88780841
55	1.9709009697	-5.67801714
56	-0.5967586725	-0.04292323
57	-0.5525072116	-2.26102245
58	0.6959666337	0.16513398
59	-0.1556639646	0.41226654
60	1.3488981952	-2.27228484
61	-1.0685230705	-4.59119521
62	1.0644507468	-1.70588097
63	-1.3127217645	-3.44241695
64	2.0636947023	-5.65954675
65	0.1313830107	0.58189420

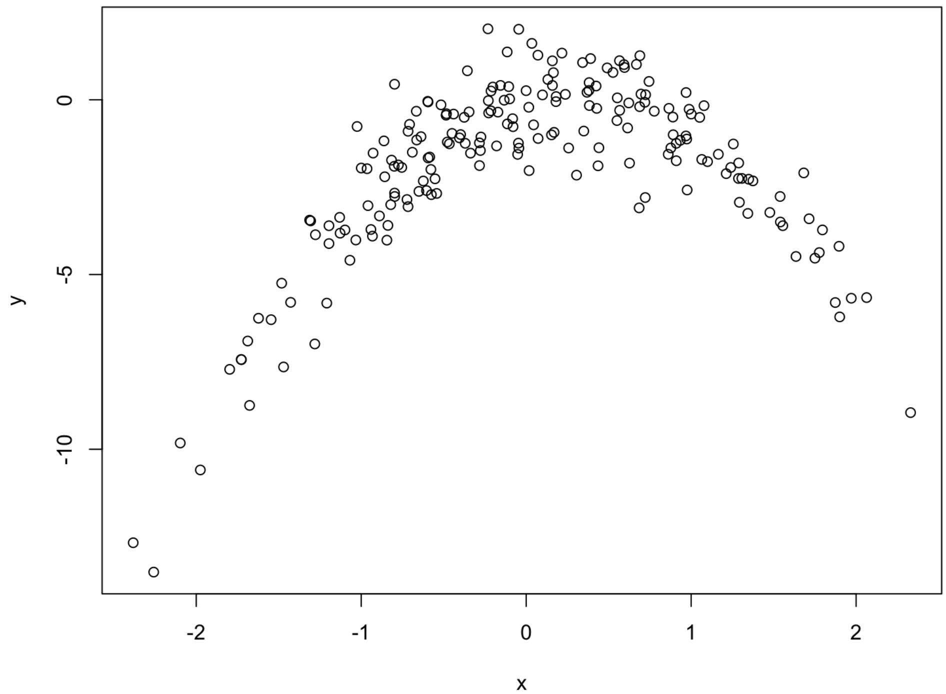
66	-0.2316884489	2.03198429
67	-0.3973555230	-0.99484339
68	0.8894320823	-0.49375130
69	0.5261690395	0.78269654
70	-0.1712732430	-0.35220088
71	0.1586768974	1.11603962
72	-0.4856650662	-0.43982404
73	-0.9589060750	-3.02787919
74	0.1805172921	-0.05228947
75	0.7217342828	-2.79610810
76	-0.3695404781	-1.24381503
77	0.2375383125	0.15619802
78	-0.6659221124	-0.32670204
79	-0.7968075098	-2.76582673
80	-0.0516969313	-1.56119087
81	1.2869283333	-2.25062174
82	-0.2141496627	-0.31480651
83	-0.5747454643	-2.71377182
84	-1.4707270443	-7.64455205
85	-1.0327384328	-4.01186658
86	-1.3065248552	-3.46479151
87	-0.8382524073	-3.59408856
88	-1.1306536810	-3.36359404
89	0.3687481753	0.21559276
90	-0.2018030203	0.36636424
91	-1.2776599028	-3.86087185
92	-0.7980124807	0.44585533
93	0.1590824229	0.41099298
94	0.6147976331	-0.80496528
95	0.6879479624	-0.19564098
96	-0.0470510111	-1.23914143
97	2.3303216783	-8.95097997
98	-0.5775659910	-1.99235139
99	0.9684791343	-1.03375179
100	-0.2775356275	-1.45025477

101	0.6848019360	-3.09268851
102	-0.1151135095	1.36985331
103	-0.3564751798	0.83173028
104	-0.1057716076	0.37722241
105	0.0448827901	-0.71643939
106	-1.7261732320	-7.43628320
107	1.5557870203	-3.60201208
108	0.7764126917	-0.32629930
109	-1.0985075088	-3.72355360
110	-1.7280197536	-7.42958121
111	0.4276382246	-0.24511093
112	0.7445646452	0.52568034
113	0.8652207970	-0.24394978
114	0.3053288101	-2.15551072
115	-0.1140227912	-0.68997687
116	0.4236522402	0.39506982
117	-0.7977096868	-2.66853581
118	-0.6041972494	-2.59635014
119	1.7150105938	-3.40421367
120	-0.7159482778	-0.90014751
121	-0.1332356122	-0.01051404
122	-0.9997650626	-1.95232731
123	1.8737601171	-5.80104647
124	-0.3373884320	-1.52577030
125	0.9732702887	-1.12238823
126	0.9878279309	-0.27019526
127	-0.9412566085	-3.71121148
128	0.3491855939	-0.89431238
129	-0.5944186817	-1.66440068
130	-2.3822428313	-12.67473814
131	1.0780189737	-0.16727267
132	0.6682451050	1.01313897
133	-0.9646256667	-1.97151861
134	-1.9752373319	-10.59190125
135	-0.5847739007	-1.63653000

136	0.9692770362	0.20796480
137	0.5522923259	0.05596528
138	-0.0821555007	-0.54019582
139	-1.6767137584	-8.74517167
140	1.2126074270	-2.11267032
141	1.0004998710	-0.40837191
142	0.7193289908	-0.07048793
143	-0.8443641520	-4.01606610
144	0.6219853903	-0.08932241
145	-0.7226137804	-2.85298784
146	-0.4494786251	-0.96106433
147	-1.1955060501	-3.60278091
148	0.3904723630	1.17990462
149	-0.5163766426	-0.14370718
150	0.9098689779	-1.74336520
151	0.8769846530	-1.37705875
152	-0.8161958099	-1.72594614
153	1.5392932699	-2.76662090
154	1.3745257156	-2.31958453
155	-0.4832487112	-0.39821124
156	0.5503499503	-0.59318566
157	-0.8573656630	-2.20177086
158	-0.7069613662	-0.70266279
159	-2.0970775334	-9.82070865
160	1.0994367548	-1.76843817
161	0.3420340890	1.06650105
162	0.4908294804	0.91332704
163	-0.9319990260	-3.90057416
164	-1.4278919839	-5.79609941
165	0.9757650946	-2.58057887
166	-1.5463411878	-6.29127356
167	0.0177034792	-2.02560374
168	-0.7747174012	-1.86382201
169	-0.2293422872	-0.02333439
170	-0.2743821044	-1.06385761

```
171  1.7960637815  -3.72286011
172 -0.4781128994  -1.20797494
173 -0.5947628530  -0.06040863
174 -2.2579382170 -13.51519417
175  1.6826072118  -2.09161048
176  0.0722906844  -1.10625306
177 -0.4400240932  -0.40826205
178  0.6265733926  -1.81109669
179 -0.7997960594  -1.90223804
180 -1.1279860222  -3.81661442
181 -1.0250160534  -0.76362512
182  0.0710717295   1.27793900
183  0.3817111616   0.49359057
184 -1.6225883175  -6.25054679
185  1.9005426699  -6.21267209
186 -0.7161791664  -3.05721269
187  0.3804596689   0.26909412
188  0.4408428474  -1.37164919
189  0.2573258583  -1.37879719
190 -0.1794485371  -1.31925064
191 -0.6901276793  -1.50088908
192 -0.0004228025   0.26177169
193  0.5655808964   1.11973408
194 -1.2087470098  -5.81777880
195 -0.3461711560  -0.34579272
196 -0.6501970444  -2.62023640
197 -0.8895916708  -3.32253969
198  1.4770298873  -3.22494027
199 -1.1954751385  -4.11350783
200  1.7504948348  -4.53185060
> dim(simdata)
[1] 200   2
```

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



⇒ *From the plot obtained above, the data looks like a parabolic distribution. (atleast approximately)*

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

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

Compute the LOOCV errors that result from fitting these models.

```
> library(boot)
> glm.fit = glm(y~x,data=simdata)
> cv.err = cv.glm(simdata,glm.fit)
> cv.error = rep(0,4)
> for (i in 1:4){
 glm.fit = glm(y~poly(x,i),data=simdata)
  cv.error[i] = cv.glm(simdata,glm.fit)$delta[1]
 }
> cv.error
[1] 6.0292645 0.9487129 0.9597797 0.9742622
```

⇒ LOOCV can be computed directly for generalized linear models using glm() and cv.glm().

⇒ Since, linear reg belongs to generalized linear models, we can use the glm() rather than lm() to fit the model.

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

```
> set.seed(2)
> x1=rnorm(200)
> y1=x1-2*x1^2+rnorm(200)
> simdata1 = data.frame(x1,y1)
             x1           y1
1   -0.896914547  -2.207842334
2    0.184849185  -0.903041455
3    1.587845331  -0.583762837
4   -1.130375674  -3.467163993
5   -0.080251757  -1.059686775
6    0.132420284   0.481188203
7    0.707954729  -0.416341139
8   -0.239698024  -0.705155882
9    1.984473937  -5.292336053
10  -0.138787012   0.053963396
11   0.417650751   1.095032648
12   0.981752777  -1.470373105
13  -0.392695356   1.095091602
14  -1.039668977  -4.635688546
15   1.782228960  -4.431765003
16  -2.311069085 -12.541685985
17   0.878604581   0.549486686
18   0.035806718  -1.291238649
19   1.012828692  -2.178443003
20   0.432265155   1.734207397
21   2.090819205  -6.254084484
22  -1.199925820  -3.364050630
23   1.589638200  -4.257456151
24   1.954651642  -3.762822796
25   0.004937777   0.075329000
26  -2.451706388 -14.004154776
```

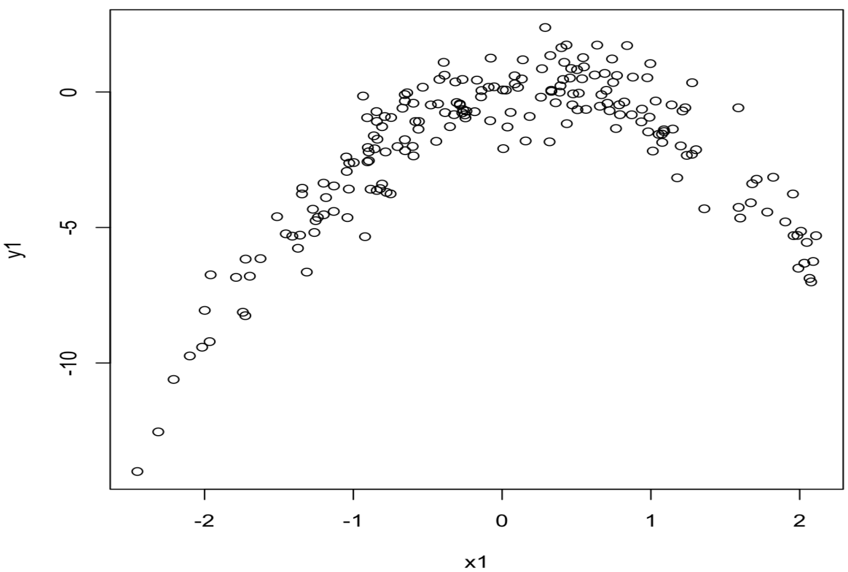
27	0.477237303	-0.068125677
28	-0.596558169	-0.414169834
29	0.792203270	-0.836678651
30	0.289636710	2.380109766
31	0.738938604	1.219979864
32	0.318960401	-1.838832805
33	1.076164354	-1.859417513
34	-0.284157720	-0.475596669
35	-0.776675274	-3.704782658
36	-0.595660499	-2.358740157
37	-1.725979779	-8.254654659
38	-0.902584480	-2.049187571
39	-0.559061915	-1.088238472
40	-0.246512567	-0.955889250
41	-0.383586228	-0.759643054
42	-1.959103175	-6.746855423
43	-0.841705060	-1.085814524
44	1.903547467	-4.793795233
45	0.622493930	0.626601054
46	1.990920436	-6.501739177
47	-0.305483725	-0.389170051
48	-0.090844235	0.174171928
49	-0.184161452	-0.726423506
50	-1.198767765	-4.530860283
51	-0.838287148	-1.742194659
52	2.066301356	-6.878030606
53	-0.562247053	-1.371876142
54	1.275715512	-2.298729192
55	-1.047572627	-2.398939052
56	-1.965878241	-9.211576113
57	-0.322971094	-0.835261896
58	0.935862527	-1.100910908
59	1.139229803	-0.476022607
60	1.671618767	-4.088319308
61	-1.788242207	-6.840849445

62	2.031242519	-6.319008256
63	-0.703144333	-2.013963719
64	0.158164763	-1.806601923
65	0.506234797	-0.653353936
66	-0.819995106	-3.564215899
67	-1.998846995	-8.058451344
68	-0.479292591	-0.477163075
69	0.084179904	0.597446138
70	-0.895486611	-2.535970423
71	-0.921275666	-5.338941397
72	0.330449503	0.064930239
73	-0.141660809	-0.177059907
74	0.434847762	-1.168641133
75	-0.053722626	0.191430348
76	-0.907110376	-0.947608906
77	1.303512232	-2.130295925
78	0.771789776	0.609888951
79	1.052525595	-1.563137581
80	-1.410038341	-5.319211370
81	0.995984590	1.047821839
82	-1.695764903	-6.797483312
83	-0.533372143	0.177007703
84	-1.372269451	-5.766715681
85	-2.207919779	-10.607407255
86	1.822122519	-3.145548452
87	-0.653393411	-0.334509752
88	-0.284681219	-0.440589076
89	-0.386949604	0.619102625
90	0.386694975	-0.005357864
91	1.600390852	-4.650860720
92	1.681154956	-3.384062630
93	-1.183606388	-3.896290671
94	-1.358457254	-5.282233898
95	-1.512670795	-4.601240782
96	-1.253104899	-4.747741938

97	1.959357077	-5.300154809
98	0.007645872	-2.089906587
99	-0.842615198	-3.633096121
100	-0.601160105	-2.008409906
101	1.074459406	-1.552586468
102	0.260597835	-0.190714883
103	-0.314271980	0.372516588
104	-0.749630095	-3.758941961
105	-0.862198330	-1.616790921
106	2.048040303	-5.550353199
107	0.939920078	-0.632126263
108	2.008687116	-5.139782751
109	-0.421373572	0.464971260
110	-0.350834423	-1.278174810
111	-1.027380598	-2.623071921
112	-0.250519127	-0.844249877
113	0.471859466	-0.475500567
114	1.358939821	-4.308486928
115	0.564168603	-0.641849716
116	0.455980090	0.515474958
117	1.230953663	-0.582322489
118	1.147136848	-1.367019479
119	0.106598041	0.172238471
120	-0.783316657	-2.211103708
121	1.241199827	-2.338537751
122	0.138858419	1.189962637
123	1.710631588	-3.220258494
124	-0.430640975	-0.436990715
125	-1.044229581	-2.929912262
126	0.537579525	0.489544452
127	-0.669585987	-0.597486973
128	0.638805611	1.731266130
129	-1.723989834	-6.165478286
130	-1.742430080	-8.126383194
131	0.689804173	0.686163264

132	0.330963177	0.024217166
133	0.871067709	-0.840176510
134	-2.016245582	-9.417497348
135	1.212579104	-0.699712203
136	1.200494699	-1.989070683
137	1.032068326	-0.332520138
138	0.786410256	-0.471922659
139	2.110073514	-5.302219068
140	-1.453809847	-5.233257915
141	-0.583103848	-1.088147214
142	0.409723983	0.461217718
143	-0.806981635	-3.398314798
144	0.085550441	0.302419518
145	0.746243169	0.352632287
146	-0.653673061	-1.759681926
147	0.657105983	-0.525466447
148	0.549909235	0.928552326
149	-0.806729358	-1.280112918
150	-0.997379717	-2.604118451
151	0.975890638	0.526825322
152	-0.169423181	0.439628900
153	0.722191779	-0.690937408
154	-0.844418607	-0.722252935
155	1.277293685	0.343599389
156	-1.343110549	-3.551577872
157	0.765340669	-1.347397830
158	0.464202570	0.866535366
159	0.267993278	0.859239076
160	0.667522687	-0.100432217
161	0.398467284	1.636278509
162	-0.638071031	-0.029082227
163	-0.267712904	-0.769810465
164	0.359879565	-0.396890260
165	-1.312866094	-6.645088891
166	-0.883969610	-3.586400435

```
167  2.077094795  -7.008279795
168 -2.099225632  -9.739388568
169 -1.238505965  -4.626487321
170  0.990433090  -0.929809062
171  1.088661863  -1.458690619
172  0.839852254   1.713886253
173  0.056858639  -0.753217649
174  0.323878051   1.344717967
175 -0.904668668  -2.575964043
176 -0.652183848  -2.165312257
177 -0.262454638  -0.649819693
178 -0.934662841  -0.148738020
179  0.821161213  -0.369330966
180 -1.624259173  -6.148655723
181 -1.030403669  -3.581591123
182 -1.261929312  -5.185608814
183  0.392184626   0.228214426
184 -1.131438262  -4.406765221
185  0.544144484   1.268443711
186  1.176608935  -3.166475546
187  0.025228569   0.070116689
188  0.515133170  -0.044135014
189 -0.654109760  -0.096700115
190  0.503641991   0.825524488
191 -1.272119222  -4.325341777
192 -0.076771154   1.250998232
193 -1.345319376  -3.762758692
194 -0.266317560   0.463667903
195  1.087562995  -1.394450370
196  0.700567795   0.061805803
197 -0.442759515  -1.820302827
198 -0.788519966  -0.904335964
199 -0.856775710  -2.095075092
200 -0.746419015  -0.942131153
> plot(simdata1)
```




```
> glm.fit1 = glm(y1~x1,data=simdata1)
> cv.err1 = cv.glm(simdata1,glm.fit1)
>
> cv.error1 = rep(0,4)
> for (i in 1:4){
 glm.fit1 = glm(y1~poly(x1,i),data=simdata1)
 cv.error1[i] = cv.glm(simdata1,glm.fit1)$delta[1]
}
> cv.error1
[1] 7.1017678 0.9627777 0.9633221 0.9761597
```

⇒ *Yes, the results obtained are close to the previous because LOOCV works as whole dataset minus one observation. Again, the test error is smallest for second degree quadratic model.*

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

⇒ *Quadratic model (2ND degree) has the smallest LOOCV error. This is to be expected because the data in original model is in quadratic form.*

(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?

```
> library(boot)

> set.seed(5)

> cv.error.5 = rep(0,5)

> for (i in 1:5) {

 glm.fit = glm(y~poly(x,i),data=simdata)
```

```
 cv.error.5[i] = cv.glm(simdata,glm.fit,K=5)$delta[1]
 }
> cv.error.5
[1] 6.1170694 0.9544515 0.9697376 0.9607885 0.9789105
```

⇒ *CV error is again smallest for model with degree 2. These results are consistent with LOOCV.*

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

```
 > library(boot)
> set.seed(2)
> cv.error.10 = rep(0,10)
> for (i in 1:10) {
 glm.fit = glm(mpg~poly(horsepower,i),data=Auto)
 cv.error.10[i] = cv.glm(Auto,glm.fit,K=10)$delta[1]
 }
> cv.error.10
 [1] 24.22275 19.14140 19.31435 19.33767 19.13637 18.95081 18.82974 19.50147 19.16699
19.19373
```

⇒ *For 10-fold CV, model with degree 7 has the least CV error, unlike 5-fold CV where 2nd degree polynomial had least CV error.*