Assignment-2
Group: 16
Rashi Dubey
Bhaumik Icchaporia

1. Use the read.table command to load this data into R. Make sure you set the 'header' option.

Ans:

>az_char<-read.table("C:/Users/dubey/OneDrive/Quarter3/Pattern/assign2/az-5000.txt",header = T)
>is.data.frame(az_char)

```
> is.data.frame(az_char) ##
[1] TRUE
```

Output:

	char	x1	y1	x2	y2	x3	y3	x4	y4	x5	y5	x6	y6	x7	y7	x8
1	n	0.1875	0.140625	0.09375	0.515625	0	0.8828125	0.1796875	0.5078125	0.4609375	0.140625	0.640625	0.109375	0.515625	0.5390625	0.3828125
2	h	0	0	0.046875	0.3125	0.0625	0.671875	0.15625	0.765625	0.328125	0.4609375	0.4609375	0.3359375	0.625	0.4609375	0.6015625
3	y	0.0078125	0.0859375	0.140625	0.1875	0.2578125	0.375	0.4140625	0.421875	0.5078125	0.1484375	0.484375	0.1953125	0.28125	0.5234375	0.125
4	h	0	0	0.0390625	0.25	0.0546875	0.5859375	0.046875	0.8359375	0.1015625	0.671875	0.2421875	0.625	0.3515625	0.828125	0.40625
5	y	0.03125	0	0.03125	0.2421875	0.1953125	0.25	0.3125	0.140625	0.40625	0.0390625	0.3046875	0.265625	0.203125	0.46875	0.0859375
6	q	0.453125	0.015625	0.1015625	0.0703125	0	0.3203125	0.2734375	0.3203125	0.3984375	0.1484375	0.40625	0.3671875	0.3984375	0.703125	0.328125
7	y	0.0234375	0.1640625	0.15625	0.3125	0.265625	0.40625	0.4375	0.53125	0.421875	0.1640625	0.3359375	0.3515625	0.203125	0.609375	0.1015625
8	t	0.0546875	0	0.203125	0.1328125	0.1953125	0.296875	0.15625	0.5703125	0.1484375	0.7421875	0.203125	0.9453125	0.3359375	0.9921875	0.0625
9	u	0	0.0234375	0.0078125	0.359375	0.0078125	0.65625	0.1171875	0.96875	0.40625	0.7265625	0.5625	0.421875	0.546875	0.3515625	0.53125
10	q	0.328125	0.1015625	0.0703125	0.0546875	0	0.3359375	0.21875	0.3359375	0.296875	0.1953125	0.296875	0.546875	0.296875	0.9375	0.484375
11	w	0.125	0.3359375	0	0.703125	0.1640625	0.9765625	0.4296875	0.671875	0.5703125	0.40625	0.4375	0.6953125	0.6328125	0.9765625	0.90625
12	c	0.6953125	0.2734375	0.9296875	0.109375	0.7890625	0.1875	0.546875	0.3671875	0.203125	0.5859375	0.0078125	0.765625	0.0546875	0.9921875	0.4609375
13	t	0.34375	0.078125	0.3359375	0.234375	0.296875	0.390625	0.2578125	0.6328125	0.2265625	0.796875	0.296875	0.9921875	0.4921875	0.9609375	0.1484375
14	o	0.4296875	0.046875	0.0859375	0.09375	0	0.4453125	0.109375	0.8359375	0.2890625	0.9921875	0.625	0.953125	0.859375	0.671875	0.875
15	m	0	0.609375	0.078125	0.8125	0.1953125	0.96875	0.3203125	0.7734375	0.4765625	0.8203125	0.6171875	0.84375	0.7109375	0.703125	0.9140625
16	b	0.046875	0.03125	0.09375	0.4453125	0.09375	0.8515625	0	0.828125	0.078125	0.4921875	0.328125	0.421875	0.546875	0.6953125	0.375
17	b	0.0625	0	0.046875	0.3515625	0.0078125	0.671875	0.015625	0.7109375	0.1484375	0.4609375	0.375	0.46875	0.3515625	0.734375	0.203125
18	j	0.3046875	0.421875	0.296875	0.5234375	0.3125	0.671875	0.3359375	0.7734375	0.3359375	0.875	0.2578125	0.953125	0.1484375	0.9765625	0.046875
19	m	0.0078125	0.1484375	0.046875	0.484375	0	0.90625	0.2578125	0.234375	0.5078125	0.3984375	0.5703125	0.7109375	0.609375	0.3203125	0.90625
20	l	0.03125	0.03125	0.0234375	0.140625	0.0234375	0.2421875	0.015625	0.3671875	0.015625	0.5546875	0.0078125	0.671875	0.0078125	0.7734375	0
21	v	0	0.3203125	0.0390625	0.1328125	0.140625	0.265625	0.140625	0.6015625	0.03125	0.890625	0.140625	0.921875	0.3515625	0.7265625	0.7265625
22	r	0	0.0859375	0.0390625	0.1953125	0.09375	0.3828125	0.1171875	0.546875	0.140625	0.75	0.1328125	0.9453125	0.140625	0.1953125	0.2109375
23	f	0.4453125	0.0703125	0.2734375	0.0703125	0.1875	0.328125	0.1796875	0.5703125	0.171875	0.8984375	0.1171875	0.9140625	0.0390625	0.7578125	0.1640625
24	n	0	0	0.03125	0.3203125	0.0859375	0.6875	0.2421875	0.5703125	0.34375	0.109375	0.5390625	0.0234375	0.7265625	0.3671875	0.859375
25	j	0.1875	0.2578125	0.1796875	0.40625	0.2265625	0.5625	0.234375	0.625	0.2421875	0.796875	0.1953125	0.96875	0.09375	0.9921875	0.0078125
26	a	0.4140625	0.7109375	0.359375	0.2421875	0.0234375	0.53125	0.046875	0.875	0.3828125	0.859375	0.5078125	0.6328125	0.3828125	0.609375	0.625
27	b	0.390625	0.109375	0.3515625	0.2734375	0.359375	0.671875	0.3984375	0.875	0.5	0.625	0.75	0.625	0.8828125	0.828125	0.4765625
28	u	0.0390625	0.0859375	0	0.3046875	0.015625	0.6640625	0.296875	0.8125	0.640625	0.6953125	0.7421875	0.4453125	0.71875	0.234375	0.765625
29	m	0	0.546875	0.203125	0.359375	0.234375	0.7734375	0.28125	0.7265625	0.46875	0.46875	0.546875	0.859375	0.71875	0.6640625	0.9453125
30	s	0.7109375	0.09375	0.5078125	0	0.2578125	0.1015625	0.3125	0.4140625	0.5	0.5390625	0.7734375	0.765625	0.671875	0.9921875	0.3359375
31	k	0.0859375	0.0859375	0.0625	0.328125	0.0390625	0.6640625	0.015625	0.96875	0.15625	0.28125	0.2578125	0.2265625	0.078125	0.40625	0.1796875
32	h	0.78125	0.4609375	0.2109375	0.0625	0.09375	0.359375	0.0078125	0.65625	0.0703125	0.8984375	0.4140625	0.6171875	0.5078125	0.7109375	0.4453125

1.b) Use the sample command to randomly select 80% of the data for training.

Ans:

>indices <- sample(1:nrow(az_char), size = (0.8*nrow(az_char)))##sampling 80% of the data
>training <- az_char[indices,] ##putting the sampled data in training vector
>test <- az_char[-indices,] ##putting the rest dataset in the test_data

Output:

	char	x1	y1	x2	y2	x3	y3	x4	y4	x5	y5	x6	y6	x7	y7	x8	y8	x9	y9
4671	u	0.0000000	0.3046875	0.0781250	0.6328125	0.1875000	0.8984375	0.5703125	0.9609375	0.9218750	0.6406250	0.9375000	0.2734375	0.8593750	0.1953125	0.9296875	0.4765625	0.9765625	0.9375000
4143	y	0.0078125	0.0000000	0.1562500	0.1640625	0.3046875	0.3906250	0.7812500	0.2734375	0.6250000	0.4687500	0.5000000	0.6093750	0.3671875	0.7500000	0.2734375	0.8515625	0.1093750	0.9921875
4501	j	0.3671875	0.0156250	0.3906250	0.4140625	0.4140625	0.5703125	0.4062500	0.7500000	0.3671875	0.9062500	0.2734375	0.9843750	0.1093750	0.9609375	0.0703125	0.9375000	0.1718750	0.9687500
1792	t	0.2343750	0.0000000	0.1796875	0.2031250	0.1562500	0.4218750	0.1328125	0.7343750	0.1875000	0.9921875	0.3671875	0.8984375	0.4531250	0.7578125	0.1015625	0.4453125	0.3750000	0.3593750
3037	q	0.3593750	0.0156250	0.1562500	0.0390625	0.0000000	0.1796875	0.0703125	0.3671875	0.3046875	0.1171875	0.3125000	0.3203125	0.2890625	0.5859375	0.2968750	0.8281250	0.3281250	0.9921875
1686	o	0.8046875	0.0625000	0.3906250	0.0625000	0.0468750	0.2890625	0.0000000	0.6171875	0.1406250	0.9375000	0.4531250	0.9921875	0.8046875	0.8906250	0.9843750	0.6484375	0.8984375	0.2500000
2155	b	0.0234375	0.0078125	0.0390625	0.3359375	0.0390625	0.6406250	0.0390625	0.9531250	0.0781250	0.7187500	0.2109375	0.5703125	0.4375000	0.7265625	0.2500000	0.9218750	0.0078125	0.8437500
3821	x	0.7265625	0.1250000	0.6796875	0.1796875	0.4609375	0.4609375	0.2500000	0.6718750	0.1250000	0.9218750	0.1953125	0.1953125	0.4531250	0.4296875	0.7187500	0.6171875	0.9531250	0.8046875
3309	y	0.0078125	0.0937500	0.1640625	0.2343750	0.2890625	0.3906250	0.4609375	0.0390625	0.3828125	0.2343750	0.2968750	0.4218750	0.2187500	0.6093750	0.1484375	0.7968750	0.0625000	0.9843750
1890	k	0.0000000	0.2343750	0.1875000	0.2812500	0.2109375	0.7187500	0.3906250	0.7500000	0.7031250	0.4453125	0.7890625	0.3125000	0.5468750	0.5000000	0.6718750	0.9140625	0.9843750	0.9921875
132	r	0.0000000	0.1406250	0.0468750	0.2968750	0.0781250	0.5468750	0.1015625	0.7109375	0.1015625	0.9375000	0.0390625	0.1250000	0.2187500	0.0000000	0.4296875	0.0078125	0.5546875	0.1250000
4506	b	0.1093750	0.0000000	0.1250000	0.2578125	0.1406250	0.5937500	0.1718750	0.7968750	0.2500000	0.5468750	0.4687500	0.5937500	0.4765625	0.8359375	0.2109375	0.9609375	0.0156250	0.9921875
942	k	0.0078125	0.0156250	0.0625000	0.5000000	0.1406250	0.8906250	0.1562500	0.7343750	0.3281250	0.4218750	0.4843750	0.1875000	0.2890625	0.4843750	0.4843750	0.7265625	0.6250000	0.9531250
4415	y	0.0312500	0.0781250	0.1796875	0.1953125	0.2812500	0.3203125	0.4453125	0.0468750	0.3906250	0.2578125	0.2968750	0.4765625	0.1875000	0.6718750	0.0937500	0.8281250	0.0000000	0.9921875
1775	t	0.6875000	0.0156250	0.5000000	0.1406250	0.4531250	0.3125000	0.3828125	0.5312500	0.2812500	0.9140625	0.5390625	0.9531250	0.0000000	0.3750000	0.2734375	0.3593750	0.5078125	0.3203125
2838	h	0.0078125	0.0000000	0.0000000	0.2187500	0.0000000	0.4296875	0.0468750	0.6250000	0.1484375	0.7265625	0.2421875	0.5312500	0.3750000	0.6328125	0.4218750	0.8046875	0.4453125	0.9687500
4995	m	0.0468750	0.2656250	0.0859375	0.6406250	0.1718750	0.7031250	0.2812500	0.1718750	0.5078125	0.5468750	0.5625000	0.8203125	0.7031250	0.2656250	0.9375000	0.4531250	0.9843750	0.9921875
2757	d	0.3515625	0.6875000	0.1640625	0.5156250	0.0000000	0.8281250	0.3359375	0.9218750	0.5078125	0.5625000	0.6015625	0.1718750	0.5468750	0.2734375	0.4765625	0.7500000	0.8984375	0.8281250
2001	u	0.0156250	0.0000000	0.0000000	0.4765625	0.2031250	0.8984375	0.5781250	0.8046875	0.5703125	0.3203125	0.5390625	0.1640625	0.5859375	0.6406250	0.7109375	0.9375000	0.8046875	0.6875000
4572	o	0.6640625	0.3593750	0.3750000	0.2343750	0.0703125	0.4062500	0.0312500	0.7187500	0.3750000	0.9531250	0.7968750	0.9296875	0.9843750	0.6875000	0.7890625	0.4062500	0.4140625	0.3046875
1929	l	0.0390625	0.0000000	0.0000000	0.1406250	0.0000000	0.3125000	0.0312500	0.4531250	0.0781250	0.5937500	0.1328125	0.7265625	0.2031250	0.8281250	0.2421875	0.8906250	0.3281250	0.9921875
4419	z	0.2343750	0.2187500	0.4140625	0.2109375	0.8984375	0.1796875	0.8125000	0.4296875	0.3984375	0.6328125	0.1171875	0.8125000	0.1562500	0.9921875	0.6171875	0.9062500	0.9375000	0.8671875

Showing 1 to 23 of 4,000 entries

```
training    4000 obs. of 19 variables
```

Shows the total of 4000 observations of 19 variables.

1.c) Use the table command to show the number of cases per class in the training data.
Ans:
>case_by_class <- table(training$char)
>print(case_by_class)
Output

```
  a   b   c   d   e   f   g   h   i   j   k   l   m   n   o   p   q   r   s   t   u   v   w   x   y   z 
140 136 159 155 160 135 147 146 154 156 133 178 153 131 176 165 167 169 164 153 178 165 152 152 156 120 
```

2. Linear Discriminant Analysis
2.a) Use the c() command to create a vector of prior probabilities equal to 1/26 for each class.
Ans:
>prior_vector <- c(rep(1/26, each=26))
>print(prior_vector)

Output: Each of the 26 values having a prior probability of **0.03846154**.

```
> print(prior_vector)
 [1] 0.03846154 0.03846154 0.03846154 0.03846154 0.03846154 0.03846154 0.03846154 0.03846154 0.03846154 0.03846154 0.03846154 0.03846154
[13] 0.03846154 0.03846154 0.03846154 0.03846154 0.03846154 0.03846154 0.03846154 0.03846154 0.03846154 0.03846154 0.03846154 0.03846154
[25] 0.03846154 0.03846154
> |
```

2.b) Use the lda command to run linear discriminant analysis on the training data with the equal priors above. You may need to load the “MASS” package. In R, the syntax "char ~." indicates the formula for our functional model – i.e., that we are trying to predict char (column one in the data) as a function of all the other variables.

Ans.

```
>require(MASS)
>lda_model <- lda(formula = char~., data = training, prior = prior_vector) ##using lda to on training dataset with each value of equal prior probability to predict character
>print(lda_model)
```

Output:

Name	Type	Value
lda_model	list [10] (S3: lda)	List of length 10
prior	double [26]	0.0385 0.0385 0.0385 0.0385 0.0385 0.0385 ...
counts	integer [26]	140 144 165 145 155 151 ...
means	double [26 x 18]	0.4106 0.0936 0.6551 0.3735 0.1537 0.2893 0.3120 0.1183 0.1744 0.4880 0.4342 0.3 ...
scaling	double [18 x 18]	3.1638 1.0381 -0.5445 0.8042 -0.6945 -1.6957 2.3666 0.1841 -0.7108 -1.0464 ...
lev	character [26]	'a' 'b' 'c' 'd' 'e' 'f' ...
svd	double [18]	24.4 23.3 16.9 14.6 14.4 13.3 ...
N	integer [1]	4000
call	language	lda(formula = char ~ ., data = training, prior = prior_vector)
terms	formula	char ~ x1 + y1 + x2 + y2 + x3 + y3 + x4 + y4 + x5 + y5 + x6 + y6 + x7 + y7 ...
xlevels	list [0]	List of length 0

2.c) c. Combine the functions table and predict to print a “confusion” matrix on the test data. This is a 26x26 matrix with diagonal elements equal to correct classifications and off-diagonal elements equal to mistakes. Which character had the best/worst performance?

Ans:

prediction

```
>test <- az_char[-indices] ## 20% data put in test vector
>predict_lda_test <- predict(object = lda_model, newdata = test)
>conf_matrix_test <- table(test$char, predict_lda_test$class)
>print(conf_matrix_test)
```

Output:

```
> require(MASS)
Loading required package: MASS
> lda_model <- lda(formula = char~.,
+                  data = training,
+                  prior = prior_vector)
> # prediction
> predict_lda_test <- predict(object = lda_model, newdata = test)
> conf_matrix_test <- table(test$char, predict_lda_test$class)
> print(conf_matrix_test)

     a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p  q  r  s  t  u  v  w  x  y  z
  a 21  0  0  4  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  4  1  7
  b  0 40  0  0  2  1  0  1  0  0  0  2  0  0  0  0  0  0  0  1  0  0  0  0  1  0
  c  1  0 33  1  3  0  0  1  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0
  d  0  0  0 25  0  0  0  1  0  0  0  0  0  0  0  0  0  0  1  1  0  0  0  4  0  0
  e  0  0  0  0 32  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  1
  f  0  0  1  0  0 37  0  0  0  0  0  2  0  0  0  5  0  2  0  4  0  0  0  1  0  0
  g  0  0  0  0  1  2 22  0  0  0  0  0  0  0  0  0  4  0  3  1  0  0  0  2 10  0
  h  3  0  0  0  3  0  0 23  0  0  2  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0
  i  0  0  0  0  0  1  0  0 28  6  0  1  0  0  0  0  0  0  0  2  0  0  0  0  0  0
  j  0  2  0  0  0  1  0  0  5 25  0  0  0  0  0  0  0  0  0  2  0  0  0  1  2  0
  k  1  0  1  0  0  0  0  4  0  0 26  2  1  1  0  0  0  0  0  0  4  0  0  0  0  0
  l  0  0  0  1  1  2  0  1  0  0  0 19  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  m  1  0  0  1  0  0  0  0  0  0  0  0 31  2  0  0  0  0  0  0  4  0  0  0  0  0
  n  0  0  0  0  0  0  0  1  0  0  1  0  0 26  0  0  1  0  0  0  4  0  0  0  0  0
  o  0  0  1  0  0  0  0  0  0  0  0  0  0  0 25  0  0  1  1  1  0  0  0  6  0  0
  p  0  0  0  0  0  2  0  0  0  0  0  0  0  0  0 38  0  0  0  0  0  2  0  1  0  0
  q  0  0  0  0  0  1  2  0  0  0  0  1  0  0  0  0 23  0  1  0  0  0  0  1  1  0
  r  0  0  0  0  0  0  0  0  0  0  0  0  0  2  0  0  3 36  0  1  1  3  0  0  1  1
  s  0  0  0  0  0  1  1  0  0  0  0  0  0  1  0  0  0  0 39  0  0  0  0  0  1  0
  t  0  0  0  0  2  2  0  0  1  1  0  1  0  0  0  0  0  0  0 27  1  0  0  2  0  1
  u  2  0  0  0  0  0  0  1  0  0  0  0  1  1  1  0  0  0  0  1 33  1  1  0  0  0
  v  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  3 29  2  0  0  0
  w  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2 44  0  0  0
  x  1  1  0  1  0  0  0  0  0  0  0  0  0  0  2  0  0  2  0  0  2  2  0 17  3  0
  y  0  0  0  0  0  2  0  0  0  0  0  1  0  0  0  0  1  1  1  1  0  0  0  0 28  0
  z  0  0  0  0  0  0  0  0  0  1  0  1  0  0  0  0  1  0  0  3  0  0  0  0  1 33
> 
```

```
> print(conf_matrix_test)

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p  q  r  s  t  u  v  w  x  y  z
 a 21  0  0  4  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  3  1  8
 b  0 40  0  0  2  1  0  1  0  0  0  2  0  0  0  0  0  0  0  1  0  0  0  0  1  0
 c  0  0 34  1  3  0  0  1  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0
 d  0  0  0 25  0  0  0  1  0  0  0  0  0  0  0  0  0  0  1  1  0  0  0  4  0  0
 e  0  0  0  0 33  1  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0
 f  0  0  0  1  0 37  0  0  0  0  0  2  0  0  0  6  0  1  0  4  0  0  0  1  0  0
 g  0  0  0  0  1  2 24  0  0  0  0  0  0  0  0  0  4  0  4  1  0  0  0  2  7  0
 h  3  0  0  0  3  0  0 23  0  0  2  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0
 i  0  0  0  0  0  1  0  0 31  5  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 j  0  2  0  0  0  1  0  0  5 25  0  0  0  0  0  0  0  0  0  2  0  0  0  1  2  0
 k  1  0  1  0  0  0  0  5  0  0 26  1  1  1  0  0  0  0  0  0  4  0  0  0  0  0
 l  0  0  0  1  1  2  0  1  0  0  0 19  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 m  1  0  0  0  0  0  0  0  0  0  0  0 32  2  0  0  0  0  0  0  4  0  0  0  0  0
 n  0  0  0  0  0  0  0  1  0  0  1  0  0 25  0  0  1  0  0  0  5  0  0  0  0  0
 o  0  0  1  0  0  0  0  0  0  0  0  0  0  0 25  0  0  1  1  1  0  0  0  6  0  0
 p  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0 39  0  0  0  0  0  1  0  1  1  0
 q  0  0  0  0  0  0  2  0  0  0  0  1  0  0  0  0 24  0  1  0  0  0  0  0  2  0
 r  0  0  0  0  0  0  0  0  0  0  1  0  0  1  0  0  3 36  0  1  1  3  0  0  1  1
 s  0  0  0  0  0  1  1  0  0  0  0  0  1  0  0  0  0  0 39  0  0  0  0  0  1  0
 t  0  0  0  0  2  2  0  0  1  1  0  1  0  0  0  0  0  1  0 26  1  0  0  2  0  1
 u  2  0  0  0  0  0  0  1  0  0  0  0  1  1  1  0  0  0  0  1 33  1  1  0  0  0
 v  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  3 29  2  0  0  0
 w  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2 44  0  0  0
 x  1  0  1  0  0  0  0  0  0  0  0  0  0  1  2  0  0  2  0  0  2  2  0 17  3  0
 y  0  0  0  0  0  1  0  0  0  0  0  1  0  0  0  0  1  1  0  2  0  0  0  0 29  0
 z  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  1  0  0  3  0  0  0  0  1 34
>
```

Character b has the best performance - 40/42 = **0.952388**.
Accuracy rate = 0.95238*100= 95.238%
Character x has the worst performance - 17/35 = **0.48**
Accuracy Rate = 0.48*100 = 48%

2.d) d. What was the total accuracy on the test and train sets?
Ans:

> accuracy_test <- sum(diag(conf_matrix_test))/sum(conf_matrix_test)
> print(accuracy_test)

Output: 0.76 or 76%

```
> accuracy_test <- sum(diag(conf_matrix_test))/sum(conf_matrix_test)
> print(accuracy_test)
[1] 0.76
> |
```

>test <- az_char[-indices,] ##the 20% data to be assigned to test_data

Accuracy for Training data

>training <- az_char[indices] ##80% sampled data
> predict_lda_training <- predict(object = lda_model, newdata = training)
> conf_matrix_training <- table(training$char, predict_lda_training$class)
> print(conf_matrix_training)

```
Console C:/Users/dubey/OneDrive/Quarter3/Pattern/assign2/assign2/
> test <- az_char[-indices,]
> training <- az_char[indices,]
> predict_lda_training <- predict(object = lda_model, newdata = training)
> conf_matrix_training <- table(training$char, predict_lda_training$class)
> print(conf_matrix_training)

      a   b   c   d   e   f   g   h   i   j   k   l   m   n   o   p   q   r   s   t   u   v   w   x   y   z
  a  85   0   2  15   2   2   1   1   0   0   1   2   0   0   0   0   1   0   0   1   1   0   1   7   2  16
  b   0 118   0   0   5   3   0   5   0   0   0   2   0   0   0   0   0   0   1   1   0   0   0   0   1   0
  c   1   0 144   0   6   0   0   3   0   0   0   2   1   0   0   0   0   0   0   1   0   0   1   0   0   0
  d  12   1   3 100   1   0   0   1   4   0   2   2   2   1   2   0   0   0   0   0   1   0   0  23   0   0
  e   0   0   3   0 151   0   0   1   0   0   0   0   0   0   0   2   0   0   0   1   0   0   0   1   1   0
  f   1   0   0   0   0  93   1   0   0   0   1   6   0   0   0  14   0   2   0  10   0   0   0   5   2   0
  g   1   0   0   0   0  12  91   0   0   0   0   0   0   0   0   0   5   0  21   5   1   0   0   2   8   1
  h   4   4   0   0   2   0   0 111   0   0   5   8   1   6   0   0   0   0   0   0   3   0   1   0   0   1
  i   0   0   0   0   0   3   0   0 114  14   0   8   0   0   0   2   0   1   0  12   0   0   0   0   0   0
  j   0   2   0   0   0   0   0   0  18 117   0   1   0   0   1   0   0   1   4   5   0   0   0   2   5   0
  k   0   1   0   0   1   1   0  19   0   0  86   4   6   2   0   0   0   0   0   3   5   1   1   1   2   0
  l   0   0   1   6  10   4   0  12   0   0   0 142   0   0   0   0   0   0   0   0   0   0   0   3   0   0
  m   1   0   0   1   0   0   0   0   0   0   1   0 136   5   0   0   1   0   0   0   6   0   2   0   0   0
  n   3   0   0   2   0   0   0   5   0   0   1   0   2  98   0   0   1   1   0   0  14   0   3   0   0   1
  o   1   0   1   4   0   4   4   0   0   0   0   0   0   0 152   1   0   2   2   1   0   2   2   0   0   0
  p   0   0   0   0   0   1   0   0   0   0   0   1   0   1   0 151   1   7   1   1   0   0   0   0   1   0
  q   1   0   0   0   4   6  16   0   0   0   0   3   0   0   0   0 125   0   1   1   0   0   0   2   8   0
  r   0   0   0   0   0   1   0   1   2   0   0   0   2   1   0   4   3 146   0   2   1   5   0   0   0   1
  s   0   0   0   0   0   0   6   0   0   0   0   2   1   2   0   0   0   1 147   2   0   0   0   0   3   0
  t   0   2   0   0   0   9   0   0   4   0   0   2   0   0   3   0   0   5   0 122   0   0   1   5   0   0
  u   8   0   0   1   0   0   0   9   0   0   3   0   2   4   0   1   0   0   0   0 123  24   3   0   0   0
  v   0   0   0   0   0   0   0   0   0   1   1   0   0   0   2   0   0   4   0   0   7 149   1   0   0   0
  w   0   0   0   0   0   0   0   0   0   0   1   0   1   0   0   0   0   1   0   0   0   4 145   0   0   0
  x   5   3   7   9   3   0   2   0   0   0   3   1   0   2   3   0   0   3   1   3   2  18   0  72  15   0
  y   0   0   0   0   0   3   2   0   0   0   0   4   0   0   0   0   0   0   3   5   0   1   0   0 138   0
  z   0   2   0   0   1   1   0   0   0   0   0   0   0   0   0   0   0   0   0   3   0   0   0   2   1 110
>
> |
```

> accuracy_training<- sum(diag(conf_matrix_training))/sum(conf_matrix_training)
> print(accuracy_training)

```
> accuracy_training<- sum(diag(conf_matrix_training))/sum(conf_matrix_training)
> print(accuracy_training)
[1] 0.7915
> |
```

The accuracy on training is 79.2% which is better than the test data.

3. Logistic Regression. The file “credit_data.txt” contains information about the financial characteristics of 885 firms, which applied for a bank loan. Use the sample command to randomly select 80% of the data for training. Use the table command to show the number of cases per class in the training and test data.

>credit_data<-read.table("C:/Users/dubey/OneDrive/Quarter3/Pattern/assign2/credit_data.txt",header = T) ## file read
>sampling<- sample(1:nrow(credit_data), size = (0.8*nrow(credit_data))) ##80% data sampled used for training
>train_data<-credit_data[sampling,]

>test_data<-credit_data[-sampling,]# testing data

Train_data with 708 observations of 15 variables

Untitled1* test_data

Filter

	Id	Fail	Leverage	CumulProfit	Liquid	OverDueDebt	WorkCap	OperProfit	ShortDebt	GuarDebt	StateLag	FiscalLag	InFinan	Links
2	2	0	1	0.08815	0.02957	0	0.03307	0.13316	0.01061	0.04186	136	35	0.09217	0.
10	10	0	1	0.02211	0.02029	0	0.09951	0.05124	0.02020	0.00000	206	160	0.04400	0.
11	11	0	0	0.05116	0.00870	0	0.54226	0.07555	0.34457	0.00000	184	177	0.08328	0.
14	14	1	0	0.01083	0.00294	1	0.01507	0.10429	0.15281	0.83896	366	311	0.20141	0.
17	17	0	1	0.15301	0.30415	0	0.19971	0.14883	0.00000	0.00000	172	151	0.16591	0.
19	19	0	1	0.09558	0.25440	0	-0.13621	-0.01643	0.00000	0.00000	247	100	-0.03902	0.
23	23	0	1	0.31044	0.00982	0	0.41486	0.08323	0.06233	0.00000	176	109	0.00895	0.
27	27	1	0	-0.07613	0.00320	1	0.57717	0.06571	0.42575	0.00000	179	72	0.06632	0.
28	28	0	1	0.46627	0.05241	0	0.64956	0.11040	0.17110	0.00000	149	131	0.07357	0.
29	29	0	1	0.21735	0.11486	0	0.44993	0.12197	0.38506	0.65291	243	174	0.15048	0.
31	31	0	0	0.09442	0.01080	0	-0.00286	0.05810	0.00000	0.35419	333	148	-0.01063	0.
36	36	0	1	0.36821	0.30347	0	0.05772	0.22207	0.00000	0.17524	167	148	0.03095	0.
52	52	1	1	0.04537	0.02590	1	0.17760	-0.05643	0.12111	0.00000	289	116	0.23265	0.
55	55	0	1	0.31327	0.24581	0	0.34631	-0.00875	0.00000	0.00000	125	98	0.07935	0.
58	58	1	0	-0.12942	0.00043	0	0.30786	-0.01843	0.17119	0.33301	158	130	-0.04411	0.
70	70	0	0	0.07263	0.04743	0	0.61638	0.00277	0.59292	0.00000	163	132	0.03280	0.
80	80	0	1	0.51462	0.14408	0	0.43635	0.33254	0.00000	0.00000	175	151	0.36931	0.
93	93	1	0	-0.00296	0.04848	1	0.07505	-0.10625	0.07249	0.52523	186	156	0.13988	0.
105	105	0	0	0.14052	0.04983	0	0.10704	-0.00464	0.56701	0.00000	284	59	-0.49910	0.
109	109	0	1	0.18332	0.00067	0	0.35893	0.04088	0.00074	0.61804	199	143	0.06659	0.
112	112	0	0	0.21065	0.00239	0	0.11854	0.01996	0.18929	0.44041	170	136	0.05777	0.
116	116	0	1	0.20049	0.02141	0	0.35451	0.19577	0.17479	0.00000	232	128	0.21197	0.

Showing 1 to 23 of 177 entries

Test_data with 177 observations out of 15 variables.

Untitled1* test_data train_data

Filter

	Id	Fail	Leverage	CumulProfit	Liquid	OverDueDebt	WorkCap	OperProfit	ShortDebt	GuarDebt	StateLag	FiscalLag	InFinan	Links
2	2	0	1	0.08815	0.02957	0	0.03307	0.13316	0.01061	0.04186	136	35	0.09217	0.
10	10	0	1	0.02211	0.02029	0	0.09951	0.05124	0.02020	0.00000	206	160	0.04400	0.
11	11	0	0	0.05116	0.00870	0	0.54226	0.07555	0.34457	0.00000	184	177	0.08328	0.
14	14	1	0	0.01083	0.00294	1	0.01507	0.10429	0.15281	0.83896	366	311	0.20141	0.
17	17	0	1	0.15301	0.30415	0	0.19971	0.14883	0.00000	0.00000	172	151	0.16591	0.
19	19	0	1	0.09558	0.25440	0	-0.13621	-0.01643	0.00000	0.00000	247	100	-0.03902	0.
23	23	0	1	0.31044	0.00982	0	0.41486	0.08323	0.06233	0.00000	176	109	0.00895	0.
27	27	1	0	-0.07613	0.00320	1	0.57717	0.06571	0.42575	0.00000	179	72	0.06632	0.
28	28	0	1	0.46627	0.05241	0	0.64956	0.11040	0.17110	0.00000	149	131	0.07357	0.
29	29	0	1	0.21735	0.11486	0	0.44993	0.12197	0.38506	0.65291	243	174	0.15048	0.
31	31	0	0	0.09442	0.01080	0	-0.00286	0.05810	0.00000	0.35419	333	148	-0.01063	0.
36	36	0	1	0.36821	0.30347	0	0.05772	0.22207	0.00000	0.17524	167	148	0.03095	0.
52	52	1	1	0.04537	0.02590	1	0.17760	-0.05643	0.12111	0.00000	289	116	0.23265	0.
55	55	0	1	0.31327	0.24581	0	0.34631	-0.00875	0.00000	0.00000	125	98	0.07935	0.
58	58	1	0	-0.12942	0.00043	0	0.30786	-0.01843	0.17119	0.33301	158	130	-0.04411	0.
70	70	0	0	0.07263	0.04743	0	0.61638	0.00277	0.59292	0.00000	163	132	0.03280	0.
80	80	0	1	0.51462	0.14408	0	0.43635	0.33254	0.00000	0.00000	175	151	0.36931	0.
93	93	1	0	-0.00296	0.04848	1	0.07505	-0.10625	0.07249	0.52523	186	156	0.13988	0.
105	105	0	0	0.14052	0.04983	0	0.10704	-0.00464	0.56701	0.00000	284	59	-0.49910	0.
109	109	0	1	0.18332	0.00067	0	0.35893	0.04088	0.00074	0.61804	199	143	0.06659	0.
112	112	0	0	0.21065	0.00239	0	0.11854	0.01996	0.18929	0.44041	170	136	0.05777	0.
116	116	0	1	0.20049	0.02141	0	0.35451	0.19577	0.17479	0.00000	232	128	0.21197	0.

Showing 1 to 23 of 177 entries

#case by class(fail) for test_data

>case_of_test_data<- table(test_data$Fail)

>print(case_of_test_data)

```
> case_of_test_data<- table(test_data$Fail)
> print(case_of_test_data)

  0   1
137  40
>
```

>case_of_train_data<-table(train_data$Fail)
>print(case_of_train_data)

```
> case_of_train_data<-table(train_data$Fail)
> print(case_of_train_data)

  0   1
556 152
```

(a) Use the glm (with family=binomial) command to fit a logistic regression to predict which firms will go bankrupt. Report the table of coefficients from R with their p-values. What are the 4 most important predictor variables?
Ans:
>lr_training_data <- glm(formula = Fail ~.,family=binomial,data=train_data)
>model_lr_train <- predict(object = lr_training_data, newdata = train_data)
>print(model_lr_train)

```
> lr_training_data <- glm(formula = Fail ~.,family=binomial,data=train_data)
> model_lr_train <- predict(object = lr_training_data, newdata = train_data)
> print(model_lr_train)
          670           333            94           544           745            59           132           349 
 -2.922805891  -0.452860410  -8.539188687  -1.636920483  -2.369191867  -7.998400607  -1.312800569  -2.321437024 
          481           725           836           282           715           357           601           194 
 -0.473301223  -1.137626694  -0.254599338  -1.743480916  -0.691742244  -3.054726285  -3.624532485  -2.629521205 
          698           472            42           341            20           791           191           846 
 -0.399025910  -1.259692210  -2.328017678  -2.028641620  -3.527489922  -0.002648529  -1.782767419   1.699665164 
           92           169           540           721           142           652           139           879 
 -5.503250004  -1.978432102  -0.712317314  -0.216700124  -0.756353624   1.557707237  -4.883294022  -2.297018344 
          290           841           564           589           260           692           374            30 
 -3.460427708  -0.280154289  -5.475031094  -0.842821252  -3.114134871  -1.115859304  -2.357130066  -3.975102426 
          351           241           547           876           656           880           815           368 
 -2.624159551  -4.379615055  -0.233960369   1.499565643  -1.313999530  -0.978209012   0.724704158  -1.027504270 
          464            56           735           218            82           789           367           298 
 -3.478670510  -2.212055127  -0.386390178  -1.014074357  -1.907073292  -0.497409311  -6.675074376  -2.923802511 
           53           198           873            22           304            38           864           730 
 -6.321119070  -0.865667880   0.748184359  -5.197412622  -3.161234041  -4.602333327  -1.316738896  -1.406031701 
          208           505           635           272           369           373           840           613 
 -5.193817338  -5.519104274  -0.724311930  -3.827861436  -6.937445042  -1.564774197   1.727144330  -1.411240580 
           15           197           311           623           617           353           431           365 
 -2.137768356  -3.976589473  -3.928062302  -2.397510144  -6.767704764  -3.795595663 -14.125355747  -1.078139412 
          339           133           214           667           772           281           583           343 
 -5.846193101  -5.385935224  -3.201359598  -2.709547719   4.636070703  -2.962963366  -3.046965332  -0.888176555 
          600           669           548           334            13           278           496           155 
 -8.267387205  -2.276894451  -0.586249995  -1.972214680  -0.663559732  -2.963904561   0.044811377  -2.961376200 
            1           270           602           517           215           428           662           165 
```

>coef(summary(lr_training_data))

```
> coef(summary(lr_training_data))
                 Estimate    Std. Error     z value     Pr(>|z|)
(Intercept)  -4.557556878 0.9031007756 -5.0465651 4.498236e-07
Id            0.002148536 0.0004681292  4.5896219 4.440495e-06
Leverage     -0.493995294 0.2466330535 -2.0029566 4.518195e-02
CumulProfit  -1.692239179 0.6903486445 -2.4512820 1.423484e-02
Liquid      -16.579712590 3.5296816244 -4.6972261 2.637185e-06
OverDueDebt   1.502811194 0.3167887305  4.7438910 2.096517e-06
WorkCap      -1.038433574 0.5859744658 -1.7721482 7.636997e-02
OperProfit   -0.285993727 0.5465026472 -0.5233163 6.007542e-01
ShortDebt     2.080630290 0.6522781399  3.1897900 1.423762e-03
GuarDebt     -1.043623812 0.4813653221 -2.1680494 3.015493e-02
StateLag      0.008573054 0.0022082772  3.8822366 1.035001e-04
FiscalLag    -0.003708271 0.0031337794 -1.1833223 2.366815e-01
InFinan      -0.378553989 0.5130731751 -0.7378168 4.606258e-01
Links         6.036591473 2.6616091602  2.2680233 2.332779e-02
CapStruct     1.894084645 0.7488028346  2.5294838 1.142304e-02
>
```

>tail(sort(abs(coef(summary(model))[,1])), 4) ## intercept is included in the 4 important values, **taking the absolute value**

```
> tail(sort(abs(coef(summary(model))[,1])), 4)
  ShortDebt (Intercept)       Links      Liquid
   2.080630    4.557557    6.036591   16.579713
```

>tail(sort(abs(coef(summary(model))[,1])), 5) ##taking the absolute values of the estimate

(Note: Here we are taking the 5 most important values because intercept is one of them)

```
> tail(sort(abs(coef(summary(model))[,1])), 5)
  CapStruct   ShortDebt (Intercept)       Links      Liquid
   1.894085    2.080630    4.557557    6.036591   16.579713
```

b)Do their signs appear to be what you'd expect?

Ans: Yes, according to us these variables play an important role in predicting whether a company would earn profits or go bankrupt. Clearly, parameter Liquid plays the most important factor in predicting about the bankruptcy.

(c) Suppose that we predict a firm will go bankrupt if the predicted probability P(Y = 1 | X = x) of bankruptcy is 0.5 or greater. Find the confusion matrix for such predictions on the test data.

Ans:

```
>new_predict_test <- rep("0", nrow(test_data)
>new_predict_test[predict_model_test >= 0.5] <- "1"
>print(new_predict_test)
>cm <- table(test_data$Fail, new_predict_test)
>print(cm)
>accuracy <- sum(diag(cm))/sum(cm)
>print(accuracy)
```

```
> new_predict_test <- rep("0", nrow(test_data))
> new_predict_test[predict_model_test >= 0.5] <- "1"
> print(new_predict_test)
  [1] "0" "0" "0" "0" "0" "0" "0" "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0"
 [28] "0" "0" "0" "0" "0" "1" "0" "0" "0" "0" "0" "0" "0" "0" "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0"
 [55] "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" "0" "0" "0"
 [82] "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" "0" "1" "0" "0" "0" "0" "1"
[109] "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" "0" "0" "0" "0" "0" "0" "0" "1" "0" "0" "0"
[136] "0" "0" "1" "0" "0" "0" "0" "0" "0" "1" "0" "0" "0" "0" "0" "0" "0" "1" "1" "1" "1" "0" "1" "1" "0" "0" "1"
[163] "0" "0" "1" "1" "0" "1" "1" "1" "1" "0" "1" "1" "1" "1" "0"
> cm <- table(test_data$Fail, new_predict_test)
> accuracy <- sum(diag(cm))/sum(cm)
> print(cm)
   new_predict_test
      0   1
  0 129   8
  1  20  20
> print(accuracy)
[1] 0.8418079
```

Accuracy = 84.18

4. Regularized Logistic Regression. The R package glmnet fits penalized logistic regression models using the Lasso penalty. We want to compare the regularized vs. the unregularized fit to the credit data.

(a) Use the cv.glmnet (with family=binomial) command to fit a regularized logistic regression to the same training data used in 3a (you may need a cast from data.frame to matrix and map y from 0/1 to -1/1). Plot the cross-validation curve. Explain the plot.

Ans:

>library(glmnet)

>train_data_matrix <- data.matrix(subset(train_data, select = -c(Fail)))

>train_data_y <- train_data[2] ## extracting the ~Fail column

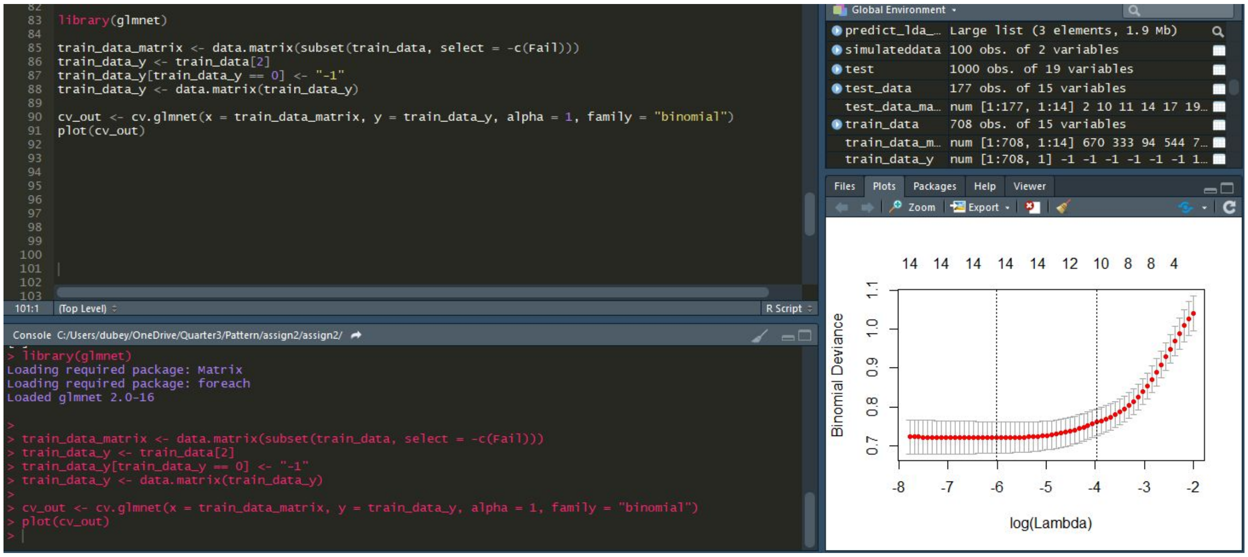
>train_data_y[train_data_y == 0] <- "-1" ##changing the values of Y_matrix, 0 to -1

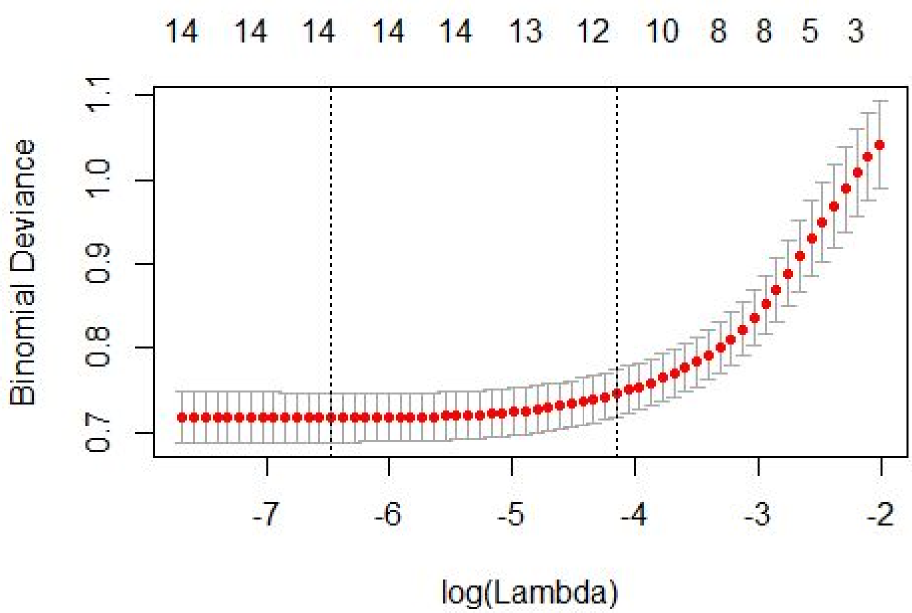
>train_data_y <- data.matrix(train_data_y)

>cv_out <- cv.glmnet(x = train_data_matrix, y = train_data_y, alpha = 1, family = "binomial")

```
> train_data_y <- data.matrix(train_data_y)
> #matrix_train_data <- model.matrix(Fail~.,train_data)
> train_data_matrix <- data.matrix(subset(train_data, select = -c(Fail)))
> train_data_y <- train_data[2]
> train_data_y[train_data_y == 0] <- "-1"
> train_data_y <- data.matrix(train_data_y)
> cv_out <- cv.glmnet(x = train_data_matrix, y = train_data_y, alpha = 1, family = "binomial")
>
```

>plot(cv_out)





Explanation - cv.glmnet() uses cross-validation to determine how each model used inside the function behaves in the dataset which can be visualised from the above plot. The above plot shows that the log(lambda) fits best to the values from -6.5 to -5.8. The lowest point in the curve indicates the optimal lambda: the log value of lambda that best minimised the error in cross-validation.

(b) The object returned by cv.glmnet() contains the value of the best lambda. Pass this value of lambda to the coef() function to retrieve the corresponding coefficient vector. Print the coefficients. Compare to your answer in 3a.

Ans:

>coef(cv_out)

```
> coef(cv_out)
15 x 1 sparse Matrix of class "dgCMatrix"
                       1
(Intercept) -3.990932086
Id           0.001712782
Leverage    -0.378537917
CumulProfit -0.888536430
Liquid      -4.747562235
OverDueDebt  1.001479324
WorkCap      .
OperProfit   .
ShortDebt    0.955522736
GuarDebt     .
StateLag     0.004184330
FiscalLag    .
InFinan     -0.045374038
Links        0.878463181
CapStruct    1.363400681
```

Below is the coefficient of 3(a). Here we get coefficient value in Workcap, OperProfit, GuarDebt, FiscalLag unlike above.

```
> coef(model)
  (Intercept)            Id      Leverage   CumulProfit        Liquid   OverDueDebt       WorkCap 
 -4.557556878   0.002148536  -0.493995294  -1.692239179 -16.579712590   1.502811194  -1.038433574 
   OperProfit     ShortDebt      GuarDebt      StateLag     FiscalLag       InFinan         Links 
 -0.285993727   2.080630290  -1.043623812   0.008573054  -0.003708271  -0.378553989   6.036591473 
    CapStruct 
  1.894084645 
> 
```

>cv_out[["glmnet.fit"]][["beta"]]## extracts the all the coefficients of the models predicted by Lasso regularization giving us beta values of 64 tested models.

```
> cv_out[["glmnet.fit"]][["beta"]]
14 x 64 sparse Matrix of class "dgCMatrix"
   [[ suppressing 64 column names 's0', 's1', 's2' ... ]]

Id          . .          0.0001779567  0.0003772601  0.0005547713  0.0007026127  0.0008342062
Leverage    . .          .             .             .            -0.0470797519 -0.1073121260
CumulProfit . -0.1476268 -0.2298587436 -0.3008665159 -0.3682696335 -0.4062472025 -0.4354373352
Liquid      . .          .             .             .             .             .
OverDueDebt . .          .             .             0.0720605673  0.1962213968  0.3031940682
WorkCap     . .          .             .             .             .             .
OperProfit  . .          .             .             .             .             .
ShortDebt   . .          .             .             .             .             .
GuarDebt    . .          .             .             .             .             .
StateLag    . .          .             .             .             .             .
FiscalLag   . .          .             .             .             .             .
InFinan     . .          .             .             .             .             .
Links       . .          .             .             .             .             .
CapStruct   .  0.1109991  0.2340852002  0.3464057248  0.4485375463  0.5516182243  0.6542732700

Id           0.0009401143  0.0010344917  0.001121520  0.001202126  0.001277042  0.001346843
Leverage    -0.1456711275 -0.1766877216 -0.204766055 -0.230248821 -0.253427518 -0.274536046
CumulProfit -0.4634928398 -0.4943855986 -0.527215128 -0.561521934 -0.596904425 -0.633092796
Liquid      -0.2104573204 -0.4876629112 -0.781412800 -1.093381795 -1.424816456 -1.776570507
OverDueDebt  0.3893332625  0.4656860466  0.535032577  0.598544748  0.657083526  0.711299493
WorkCap      .             .             .            .            .            .
OperProfit   .             .             .            .            .            .
ShortDebt    0.0969921627  0.2098277724  0.312082464  0.404985452  0.489519111  0.566509828
GuarDebt     .             .             .            .            .            .
StateLag     0.0004440165  0.0008905403  0.001301539  0.001682793  0.002038644  0.002372365
FiscalLag    .             .             .            .            .            .
InFinan      .             .             .            .            .            .
Links        .             .             .            .            .            .
CapStruct    0.7244925256  0.7853189308  0.844875351  0.903089356  0.959927465  1.015340934

Id           0.001412003  0.001472908  0.001529887  0.001583240  0.001632396  0.001674602
Leverage    -0.293791158 -0.311360219 -0.327392207 -0.342041735 -0.355689053 -0.368420113
CumulProfit -0.669823795 -0.706969295 -0.744424795 -0.782032121 -0.819603877 -0.855334683
Liquid      -2.148975809 -2.541955716 -2.954918332 -3.386659241 -3.834632043 -4.289641022
OverDueDebt  0.761692552  0.808663964  0.852541768  0.893597942  0.931929399  0.967604533
```


```
OverDueDebt  0.3893332625  0.4656860466  0.535032577  0.598544748  0.657083526  0.711299493
WorkCap      .             .             .            .            .            .
OperProfit   .             .             .            .            .            .
ShortDebt    0.0969921627  0.2098277724  0.312082464  0.404985452  0.489519111  0.566509828
GuarDebt     .             .             .            .            .            .
StateLag     0.0004440165  0.0008905403  0.001301539  0.001682793  0.002038644  0.002372365
FiscalLag    .             .             .            .            .            .
InFinan      .             .             .            .            .            .
Links        .             .             .            .            .            .
CapStruct    0.7244925256  0.7853189308  0.844875351  0.903089356  0.959927465  1.015340934

Id           0.001412003  0.001472908  0.001529887  0.001583240  0.001632396  0.001674602
Leverage    -0.293791158 -0.311360219 -0.327392207 -0.342041735 -0.355689053 -0.368420113
CumulProfit -0.669823795 -0.706969295 -0.744424795 -0.782032121 -0.819603877 -0.855334683
Liquid      -2.148975809 -2.541955716 -2.954918332 -3.386659241 -3.834632043 -4.289641022
OverDueDebt  0.761692552  0.808663964  0.852541768  0.893597942  0.931929399  0.967604533
```


```
WorkCap      .            .            .            .            .            .
OperProfit   .            .            .            .            .            .
ShortDebt    0.636655519  0.700588209  0.758877019  0.812029090  0.861094021  0.909182223
GuarDebt     .            .            .            .            .            .
StateLag     0.002686429  0.002982724  0.003262706  0.003527511  0.003773708  0.003987319
FiscalLag    .            .            .            .            .            .
InFinan      .            .            .            .            .           -0.011680254
Links        .            .            .            .            0.086615065  0.496950976
CapStruct    1.069343883  1.121904840  1.173012334  1.222711202  1.271160080  1.318351866

Id           0.001712782  0.001748641  0.001782339  0.001811533  0.001836528  0.001860163
Leverage    -0.378537917 -0.387876933 -0.396490352 -0.404358939 -0.411480118 -0.418073540
CumulProfit -0.888536430 -0.922708823 -0.957585605 -0.988729917 -1.023074731 -1.058072923
Liquid      -4.747562235 -5.215012902 -5.689055609 -6.187862166 -6.710262141 -7.227054070
OverDueDebt  1.001479324  1.033589353  1.064004079  1.092315910  1.118530254  1.143577888
WorkCap      .            .            .           -0.012447559 -0.083863274 -0.150928243
OperProfit   .            .            .            .            .            .
ShortDebt    0.955522736  0.997928854  1.036816106  1.081546238  1.152105805  1.218039999
GuarDebt     .            .            .           -0.058188653 -0.125109988 -0.187754039
StateLag     0.004184330  0.004371057  0.004547889  0.004716926  0.004884375  0.005043971
FiscalLag    .            .            .            .            .            .
InFinan     -0.045374038 -0.076508527 -0.105283152 -0.136071002 -0.149078720 -0.160928307
```

```
CumulProfit -0.888536430 -0.922708823 -0.957585605 -0.988729917 -1.023074731 -1.058072923
Liquid      -4.747562235 -5.215012902 -5.689055609 -6.187862166 -6.710262141 -7.227054070
OverDueDebt  1.001479324  1.033589353  1.064004079  1.092315910  1.118530254  1.143577888
WorkCap      .            .            .           -0.012447559 -0.083863274 -0.150928243
OperProfit   .            .            .            .            .            .
ShortDebt    0.955522736  0.997928854  1.036816106  1.081546238  1.152105805  1.218039999
GuarDebt     .            .            .           -0.058188653 -0.125109988 -0.187754039
StateLag     0.004184330  0.004371057  0.004547889  0.004716926  0.004884375  0.005043971
FiscalLag    .            .            .            .            .            .
InFinan     -0.045374038 -0.076508527 -0.105283152 -0.136071002 -0.149078720 -0.160928307
Links        0.878463181  1.228574903  1.551657794  1.870030635  2.172060125  2.455422781
CapStruct    1.363400681  1.407297522  1.450023823  1.489911603  1.504854656  1.520219928

Id           0.001882541  0.001903922  0.001923972  0.001942397  0.001959350   0.001975409
Leverage    -0.424198392 -0.430205074 -0.435481574 -0.439698034 -0.443025971  -0.446131060
CumulProfit -1.093308059 -1.127519069 -1.162362344 -1.194800176 -1.225538268  -1.255300506
Liquid      -7.735682762 -8.222051111 -8.707992207 -9.178238268 -9.631993612 -10.067318809
OverDueDebt  1.167459281  1.189984410  1.211560631  1.231866065  1.250974489   1.268991196
WorkCap     -0.213758299 -0.271779771 -0.326816406 -0.378061253 -0.425950939  -0.470324744
OperProfit   .            .            .           -0.009705785 -0.026083959  -0.041677595
ShortDebt    1.279613785  1.336337360  1.390081241  1.439969248  1.486466575   1.529684969
GuarDebt    -0.246375379 -0.301062922 -0.352313510 -0.400615254 -0.446018533  -0.488338262
StateLag     0.005195811  0.005338593  0.005475382  0.005602722  0.005721415   0.005833134
FiscalLag    .            .            .            .            .             .
InFinan     -0.171775378 -0.181835012 -0.190994713 -0.200522927 -0.210131357  -0.219137023
Links        2.721400533  2.970464486  3.204724873  3.423324956  3.627258544   3.818022952
CapStruct    1.535952614  1.551697139  1.567721268  1.584159215  1.600603869   1.616978124

Id            0.001990566   0.002002124  2.012242e-03  2.021813e-03  2.030854e-03   0.002039378
Leverage     -0.448982627  -0.452325007 -4.556394e-01 -4.587212e-01 -4.615827e-01  -0.464236367
CumulProfit  -1.284190171  -1.311620840 -1.337798e+00 -1.362790e+00 -1.386545e+00  -1.409034710
Liquid      -10.484284480 -10.893896408 -1.128789e+01 -1.166227e+01 -1.201672e+01 -12.351317050
OverDueDebt   1.285962595   1.301445700  1.315929e+00  1.329588e+00  1.342437e+00   1.354498495
WorkCap      -0.511645699  -0.552198449 -5.905323e-01 -6.261246e-01 -6.591264e-01  -0.689698637
OperProfit   -0.056512881  -0.070762718 -8.437064e-02 -9.731798e-02 -1.096246e-01  -0.121309599
ShortDebt     1.569982821   1.607826216  1.643341e+00  1.676559e+00  1.707576e+00   1.736500222
GuarDebt     -0.527757659  -0.566744530 -6.037632e-01 -6.382302e-01 -6.702741e-01  -0.700030994
StateLag      0.005938189   0.006112646  6.298826e-03  6.473076e-03  6.635693e-03   0.006787232
FiscalLag     .            -0.000223023 -4.955668e-04 -7.490701e-04 -9.844543e-04  -0.001202783
```

```
FiscalLag     .            -0.000223023 -4.955668e-04 -7.490701e-04 -9.844543e-04  -0.001202783
InFinan      -0.227515301  -0.238401116 -2.494241e-01 -2.596768e-01 -2.692025e-01  -0.278040422
Links         3.996330965   4.154230674  4.298930e+00  4.433724e+00  4.559170e+00   4.675800099
CapStruct     1.632964535   1.648591985  1.663886e+00  1.678736e+00  1.693057e+00   1.706795390

Id            0.002047402   0.002054939   0.002062007   0.002068623   0.002074805   0.002080571
Leverage     -0.466694057  -0.468967202  -0.471066847  -0.473003702  -0.474788148  -0.476430223
CumulProfit  -1.430249588  -1.450197216  -1.468898344  -1.486384378  -1.502695001  -1.517876116
Liquid      -12.666329073 -12.962156378 -13.239318167 -13.498427683 -13.740171381 -13.965290150
OverDueDebt   1.365797728   1.376362298   1.386221872   1.395407566   1.403951476   1.411886252
WorkCap      -0.717994684  -0.744160580  -0.768335347  -0.790651243  -0.811233962  -0.830202783
OperProfit   -0.132391303  -0.142887203  -0.152814493  -0.162190285  -0.171031805  -0.179356528
ShortDebt     1.763437898   1.788493330   1.811768625   1.833363672   1.853375943   1.871900245
GuarDebt     -0.727632681  -0.753206780  -0.776876606  -0.798761067  -0.818974529  -0.837626685
StateLag      0.006928249   0.007059293   0.007180910   0.007293634   0.007397990   0.007494488
FiscalLag    -0.001405080  -0.001592332  -0.001765484  -0.001925444  -0.002073081  -0.002209223
InFinan      -0.286229199  -0.293806783  -0.300810282  -0.307275791  -0.313238237  -0.318731254
Links         4.784121106   4.884616465   4.977750598   5.063970072   5.143704371   5.217366159
CapStruct     1.719913983   1.732387681   1.744202676   1.755354734   1.765847680   1.775692002

Id            0.002085940   0.002090932   0.002095566   0.002099891   0.002103867   0.002107542
Leverage     -0.477939607  -0.479325593  -0.480597064  -0.481818784  -0.482884480  -0.483857771
CumulProfit  -1.531978069  -1.545054144  -1.557159305  -1.568217718  -1.578549091  -1.588080758
Liquid      -14.174562650 -14.368790749 -14.548786994 -14.712582877 -14.866555722 -15.008820093
OverDueDebt   1.419244723   1.426059554   1.432362970   1.438143802   1.443518132   1.448474708
WorkCap      -0.847670706  -0.863744587  -0.878525272  -0.891950778  -0.904428064  -0.915885528
OperProfit   -0.187182271  -0.194527233  -0.201409992  -0.207822011  -0.213836978  -0.219448474
ShortDebt     1.889028470   1.904849359   1.919448286   1.932761620   1.945160756   1.956577803
GuarDebt     -0.854822442  -0.870661828  -0.885239946  -0.898588123  -0.910909058  -0.922227010
StateLag      0.007583622   0.007665871   0.007741694   0.007811074   0.007875341   0.007934457
FiscalLag    -0.002334661  -0.002450143  -0.002556380  -0.002653787  -0.002743502  -0.002825880
InFinan      -0.323787079  -0.328436481  -0.332708704  -0.336646839  -0.340245467  -0.343544950
Links         5.285351245   5.348038405   5.405789164   5.458691385   5.507583289   5.552529755
CapStruct     1.784903564   1.793502461   1.801512001   1.808822843   1.815731510   1.822137948

Id            0.002110937   0.002114069   0.002116956   0.002119614   0.002122077   0.002124325
Leverage     -0.484747770  -0.485561235  -0.486304401  -0.486983064  -0.487650726  -0.488215737
CumulProfit  -1.596860419  -1.604939150  -1.612366178  -1.619188471  -1.625340618  -1.631083684
Liquid      -15.140031090 -15.260921822 -15.372200500 -15.474543847 -15.565633704 -15.652002562
```

```
OperProfit   -0.224674981  -0.229535939  -0.234050714  -0.238238457  -0.242086147  -0.245675355
ShortDebt     1.967078000   1.976727810   1.985590118   1.993724079   2.001041684   2.007882640
GuarDebt     -0.932615205  -0.942144369  -0.950880789  -0.958886377  -0.966150934  -0.972863954
StateLag      0.007988780   0.008038663   0.008084442   0.008126428   0.008164395   0.008199659
FiscalLag    -0.002901469  -0.002970790  -0.003034331  -0.003092545  -0.003145538  -0.003194336
InFinan      -0.346568798  -0.349338601  -0.351874487  -0.354195196  -0.356330255  -0.358271127
Links         5.593812342   5.631704940   5.666465086   5.698334028   5.727244033   5.753991328
CapStruct     1.828065239   1.833541535   1.838594850   1.843252574   1.847371575   1.851317300

Id            0.002126389   0.002128283   0.002130021   0.002131623   0.002133084   0.002134421
Leverage     -0.488728014  -0.489195000  -0.489620804  -0.490050950  -0.490406646  -0.490726137
CumulProfit  -1.636356617  -1.641188508  -1.645613534  -1.649585406  -1.653285559  -1.656680335
Liquid      -15.731457992 -15.804350363 -15.871172934 -15.929176797 -15.985140595 -16.036688645
OverDueDebt   1.473605586   1.476123145   1.478429684   1.480488363   1.482420262   1.484192987
WorkCap      -0.973206484  -0.978889844  -0.984086589  -0.988690585  -0.993029141  -0.997005755
OperProfit   -0.248994905  -0.252060036  -0.254887293  -0.257458465  -0.259855484  -0.262064335
ShortDebt     2.014160193   2.019910704   2.025175637   2.029848363   2.034252535   2.038294014
GuarDebt     -0.979011759  -0.984636406  -0.989780423  -0.994403308  -0.998699668  -1.002632840
StateLag      0.008231986   0.008261582   0.008288664   0.008312819   0.008335460   0.008356212
FiscalLag    -0.003239010  -0.003279877  -0.003317249  -0.003351021  -0.003382233  -0.003410788
InFinan      -0.360044820  -0.361665715  -0.363146603  -0.364508383  -0.365743242  -0.366870270
Links         5.778497825   5.800927025   5.821446287   5.839872835   5.857018599   5.872721657
CapStruct     1.854954501   1.858296023   1.861363103   1.863956968   1.866530818   1.868909149

Id            0.002135645   0.002136765   0.002137790
Leverage     -0.491016767  -0.491281542  -0.491556040
CumulProfit  -1.659785577  -1.662624145  -1.665188846
Liquid      -16.083896275 -16.127084888 -16.163188927
OverDueDebt   1.485815083   1.487298418   1.488596307
WorkCap      -1.000639654  -1.003958560  -1.006856475
OperProfit   -0.264095326  -0.265961121  -0.267640291
ShortDebt     2.041990645   2.045369675   2.048317913
GuarDebt     -1.006226855  -1.009509578  -1.012413985
StateLag      0.008375185   0.008392521   0.008407626
FiscalLag    -0.003436877  -0.003460705  -0.003481985
InFinan      -0.367899026  -0.368837954  -0.369697780
Links         5.887074135   5.900186159   5.911781901
CapStruct     1.871089563   1.873085859   1.874634974
>
```

> which(cv_out$lambda == cv_out$lambda.min) ## to find out the minimum value of the lambda

The **minimum value** of the lambda, because the header row and column contains 0, we would extract the **43rd value from the Lasso model**. Because whenever we run the program the value of the Lambda minimum changes therefore we put this in a variable called **min_lambda index** and the value we need will be **min_lambda_index+1**

```
> which(cv_out$lambda == cv_out$lambda.min)
[1] 42
```

This is the **Best Value** of the lamda

```
> cv_out$lambda.min
[1] 0.002956224
```

0.002956

>min_lambda_index <- which(cv_out$lambda == cv_out$lambda.min)##extracting the coefficients of the 43rd row
>cv_out$glmnet.fit$beta[,min_lambda_index+1]
>cv_out$glmnet.fit$a0[min_lambda_index=1]

```
> min_lambda_index <- which(cv_out$lambda == cv_out$lambda.min)
> cv_out$glmnet.fit$beta[,min_lambda_index+1]
         Id     Leverage  CumulProfit        Liquid  OverDueDebt       WorkCap   OperProfit    ShortDebt     GuarDebt
0.002080571 -0.476430223 -1.517876116 -13.965290150  1.411886252 -0.830202783 -0.179356528  1.871900245 -0.837626685
   StateLag    FiscalLag      InFinan         Links    CapStruct
0.007494488 -0.002209223 -0.318731254   5.217366159  1.775692002
> cv_out$glmnet.fit$a0[min_lambda_index=1]
       s0
-1.296888
>
```

On extracting the Estimate values of the data we find that the

- **Liquid(-13.96~14)**
- **Links(5.21)**
- **ShortDebt(1.87)**
- **CapStruct(1.77)**

are the 4 most important predictor values. These values appear a to be approximately similar with the Estimate values in 3(a).

4.(c)Use the predict function with the same value of lambda to predict on the test data. Show the confusion matrix. Compare the accuracy with 3c.
Ans:

> new_cv_predict_class <- cv_out_predict_test >= cutoff #cutoff=0.5
> new_cv_predict_class <- as.numeric(new_cv_predict_class)
> cm_cv_test <- table(test_data$Fail, new_cv_predict_class)
> accuracy_cv <- sum(diag(cm_cv_test))/sum(cm_cv_test)
> print(accuracy_cv)

```
> new_cv_predict_class <- cv_out_predict_test >= cutoff
> new_cv_predict_class <- as.numeric(new_cv_predict_class)
> cm_cv_test <- table(test_data$Fail, new_cv_predict_class)
> accuracy_cv <-  sum(diag(cm_cv_test))/sum(cm_cv_test)
> print(accuracy_cv)
[1] 0.8474576
```

The accuracy of the cross validated data is approximately the same as is in 3(c) which is 84.74.

6. We will now perform cross-validation on a simulated data set.
(a) Generate a simulated data set as follows:

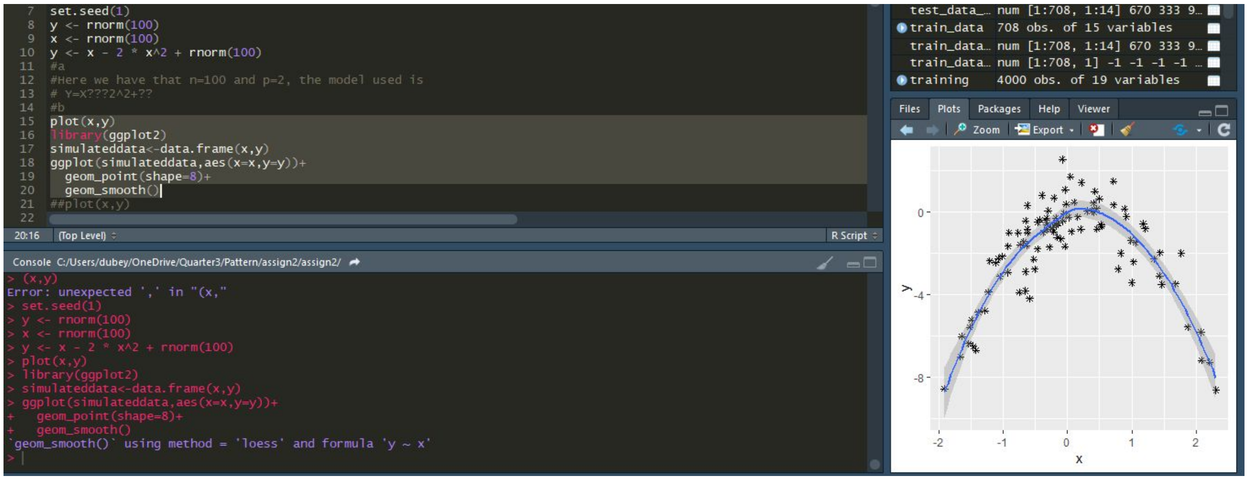
> set .seed (1)
> y=rnorm (100)
> x=rnorm (100)
> y=x-2* x^2+ rnorm (100)

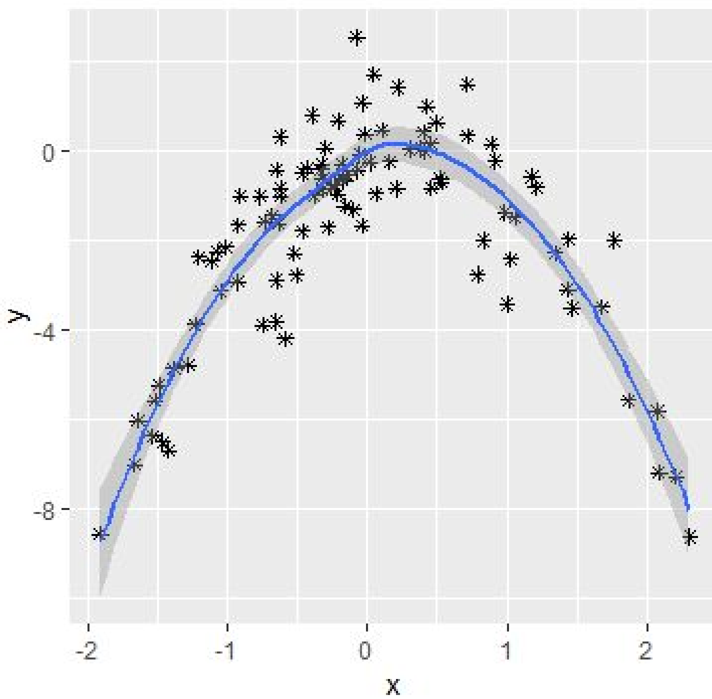
In this data set, what is n and what is p? Write out the model used to generate the data in equation form.

Ans: n= 100, p = 2. The model used here is Y = X - 2 * x^2.

(b)Create a scatterplot of X against Y . Comment on what you find.
Ans:

>plot(x,y)
>library(ggplot2)
>simulateddata<-data.frame(x,y)
ggplot(simulateddata,aes(x=x,y=y))+
geom_point(shape=8)+
geom_smooth()





Plot explanation - The plot here gives a parabolic curve or quadratic relationship between x and y.

(c) Set a random seed, and then compute the LOOCV errors that result from fitting the following four models using least squares:

i). Y = β0 + β1X +

Ans:

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

```
> 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. Y = β0 + β1X + β2X2 +
Ans:
> fit.glm.2 <- glm(y ~ poly(x, 2))
> cv.glm(Data, fit.glm.2)$delta[1]

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

iii. Y = β0 + β1X + β2X2 + β3X3 +
Ans:
> fit.glm.3 <- glm(y ~ poly(x, 3))
> cv.glm(Data, fit.glm.3)$delta[1]

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

iv. Y = β0 + β1X + β2X2 + β3X3 + β4X4 + . Note you may find it helpful to use the data.frame() function to create a single data set containing both X and Y .
Ans:

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

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

(d) Repeat (c) using another random seed, and report your results. Are your results the same as what you got in (c)? Why?
Ans:
> fit.glm.1 <- glm(y ~ x)
> cv.glm(Data, fit.glm.1)$delta[1]

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

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

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

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

The results above are identical to the results obtained in (c) **since LOOCV evaluates n folds of a single observation.**

(e) Which of the models in (c) had the smallest LOOCV error? Is this what you expected? Explain your answer.
Ans:
We may see that the LOOCV estimate for the test MSE is minimum for "fit.glm.2", this is not surprising since we saw clearly in (b) that the relation between "x" and "y" is quadratic.

(f) Comment on the statistical significance of the coefficient estimates that results from fitting each of the models in (c) using least squares. Do these results agree with the conclusions drawn based on the cross-validation results?
Ans:

>summary

```
>   summary(fit.glm.4)

Call:
glm(formula = y ~ poly(x, 4))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.8914  -0.5244   0.0749   0.5932   2.7796

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -1.8277     0.1041 -17.549   <2e-16 ***
poly(x, 4)1   2.3164     1.0415   2.224   0.0285 *
poly(x, 4)2 -21.0586     1.0415 -20.220   <2e-16 ***
poly(x, 4)3  -0.3048     1.0415  -0.293   0.7704
poly(x, 4)4  -0.4926     1.0415  -0.473   0.6373
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 1.084654)

    Null deviance: 552.21  on 99  degrees of freedom
Residual deviance: 103.04  on 95  degrees of freedom
AIC: 298.78

Number of Fisher Scoring iterations: 2
```

The p-values show that the linear and quadratic terms are statistically significant and that the cubic and 4th degree terms are not statistically significant. This agree strongly with our cross-validation results which were minimum for the quadratic model.