1. **Implement the logistic regression algorithm using Gradient Descent in analogy with neural networks for p attributes. The objective function, which your program should minimize, can be either the training MSE or negative log likelihood (NLL).**

   The logistic regression algorithm was implemented using gradient descent in analogy with neural networks for p attributes.

   The objective function used is **NLL negative log-likelihood**. And in all the other tasks NLL was used instead if MSE.

2. **Use the Auto data set. Create a new variable high that takes values:**

   high: = (1 if mpg ≥ 23, 0 otherwise (i.e., if mpg ≤ 22).

   Apply your program to the Auto data set and new variable to predict high given horsepower, weight, year, and origin. (In other words, high is the label and horsepower, weight, year, and origin are the attributes.) Since origin is a qualitative variable, you will have to create appropriate dummy variables and normalize the attributes.

   A new attribute high was create with values 1 if mpg>=23, 0 otherwise. Since origin is qualitative variable, 3 new dummy variables origin.1, origin.2 and origin.3 was created and appended to the dataset. The new dataset was normalized with mean=0 and variance=1.

3. **Split the data set randomly into two equal parts, which will serve as the training set and the test set. Use your birthday (in the format MMDD) as the seed for the pseudorandom number generator.**

A random seed was set using the birthday as mention in the question and the dataset was randomly divided into two equal parts for each Training and Testing.

4. **Train your algorithm on the training set using independent random numbers in the range [−0.7, 0.7] as the initial weights. Find the MSE on the test set. Try different values of the learning rate η and of the number of training steps (so that your stopping rule is to stop after a given number of steps). Give a small table of test and training MSEs.**

   Training using the Training Set: train.X and train.Y

Output:

```
     horsepower   weight     year    origin.1    origin.2    origin.3
210    1.042487  -0.2844008  0.6044447  1.260322   -0.9798536  1.036273
[1] -0.1523494 # final bias
[1] 2.685469 #negative log likelihood
```

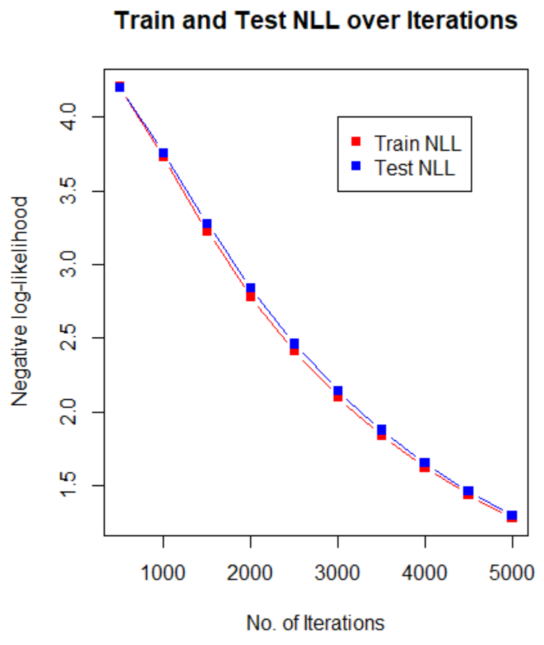
   Finding Negative Log Likelihood on Test Set: test.X and test.Y

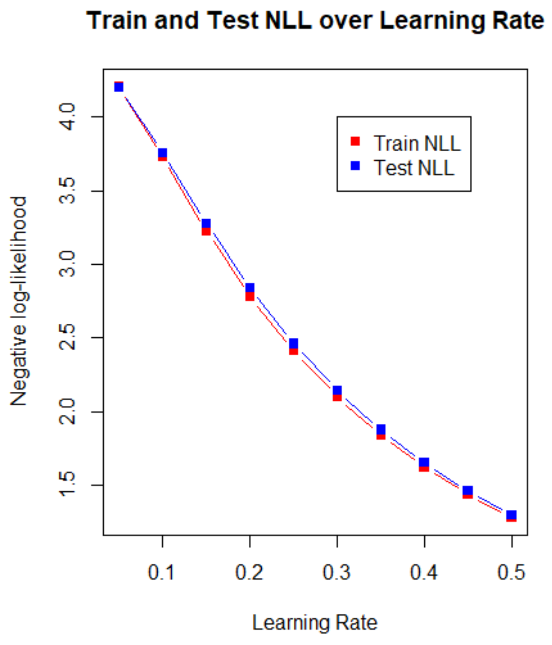
Output:

```
    horsepower    weight     year     origin.1     origin.2     origin.3
397 -0.04836452  -0.2567816  1.243885  1.227959  -0.4149348  0.00150012
[1] -0.005037129 # final bias
[1] 2.740296 #negative log likelihood
```

NLL: Negative log likelihood

Sl.no	Training NLL	Test NLL	eta(learning rate)	No. of iterations
1	4.206186	4.197755	0.05	500
2	3.728034	3.757745	0.10	1000
3	3.223296	3.275329	0.15	1500
4	2.781503	2.837132	0.20	2000
5	2.411032	2.46134	0.25	2500
6	2.101758	2.144296	0.30	3000
7	1.842491	1.877297	0.35	3500
8	1.624127	1.651955	0.40	4000
9	1.439557	1.46128	0.45	4500
10	1.283125	1.299568	0.50	5000





5. **Try different stopping rules, such as: stop when the value of the objective function (the training MSE) does not change by more than 1% of its initial value over the last 10 training steps.**

   A program to stop function when the value of NLL does not change by more than 1% of its initial value over the last 10 training steps.

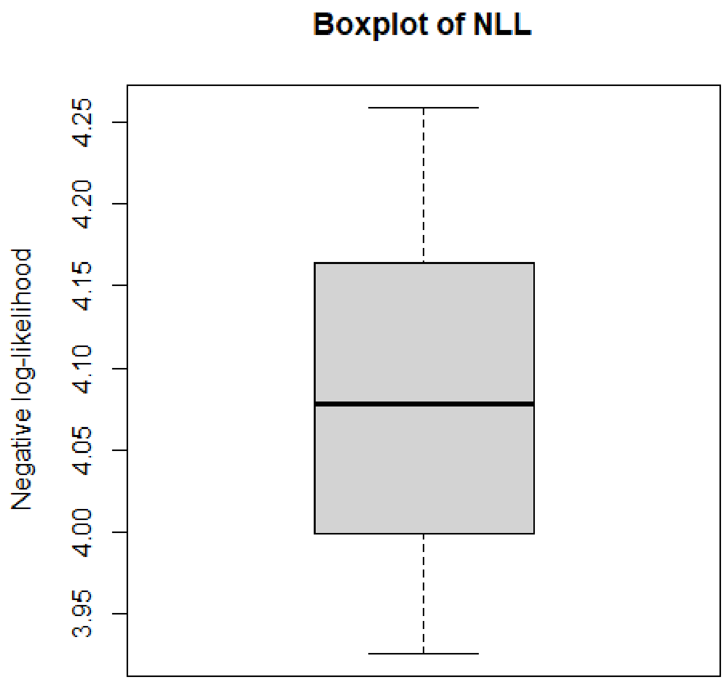
Output:

```
[1] 4.254223
```

6. Run your logistic regression program for a fixed value of η and for a fixed stopping rule (producing reasonable results in your experiments so far) 100 times, for different values of the initial weights (produced as above, as independent random numbers in [−0.7, 0.7]). In each of the 100 cases compute the test MSE and show it in your report as a boxplot.

Output:

```
[1] 4.258967 4.250318 4.246203 4.242410 4.238649 4.234901 4.231164 4.227438 4.223722
4.220017 4.216323 4.212639 4.208966 4.205303 4.201651 4.198010 4.194379

 [18] 4.190758 4.187148 4.183548 4.179959 4.176379 4.172810 4.169252 4.165703 4.162164
4.158636 4.155117 4.151609 4.148111 4.144622 4.141143 4.137675 4.134216

 [35] 4.130767 4.127327 4.123897 4.120477 4.117067 4.113666 4.110275 4.106893 4.103521
4.100158 4.096805 4.093461 4.090126 4.086801 4.083485 4.080178 4.076881

 [52] 4.073592 4.070313 4.067043 4.063782 4.060529 4.057286 4.054052 4.050827 4.047610
4.044402 4.041204 4.038013 4.034832 4.031659 4.028495 4.025340 4.022193

 [69] 4.019055 4.015925 4.012804 4.009691 4.006587 4.003491 4.000403 3.997323 3.994252
3.991189 3.988135 3.985088 3.982050 3.979019 3.975997 3.972983 3.969976

 [86] 3.966978 3.963988 3.961005 3.958031 3.955064 3.952105 3.949153 3.946210 3.943274
3.940346 3.937425 3.934512 3.931607 3.928709 3.925819

Mean Negative log-likelihood: 4.082411
```



7. **Redo the experiments in items 4–5 modifying the training procedure as follows. Instead of training logistic regression once using Gradient Descent, train it 4 times using Gradient Descent with different values of the initial weights and then choose the prediction rule with the best training MSE.**

Random weights were assigned for 4 runs of the Gradient Descent function GD.R

Output:

```
[1] "The Best NLL "

[1] 2.783341

[1] "is achieved by the weights: "

[1]  0.2541547  0.1020931  1.2837565 -1.1050295 -1.3046548  0.4495285
```

General Observations:

The negative log-likelihood decreases with increasing learning rate.

The negative log-likelihood decreases with increasing number of iterations.

The initial weights act as a vital parameter in finding negative log-likelihood.