Machine Learning (ECE5XX)

# Assignment 1

Aniket Pradhan 2017133

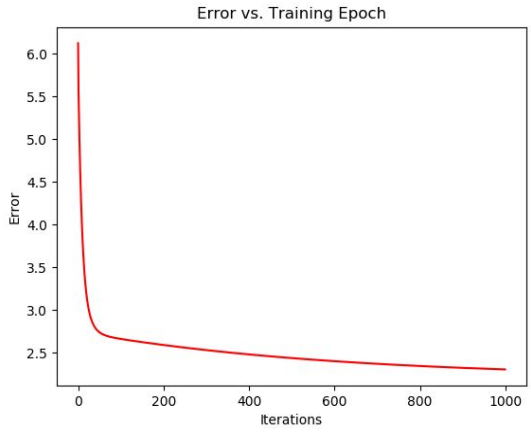
## Question 1

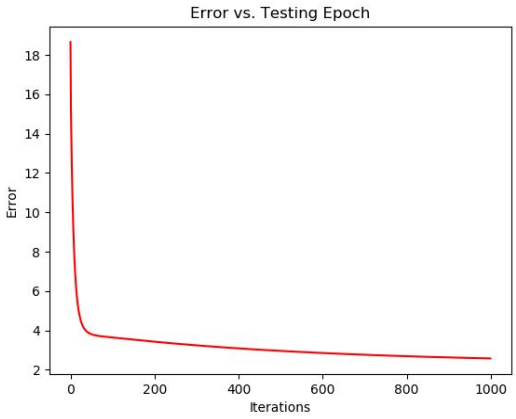
### Part 1:

I have shuffled the data, and encoded the feature ‘sex’ as per the following mapping:

```
mapping = {'M': 1, 'F': 2, 'I': 3}
```

I had set the learning rate as: 0.1 and number of iterations as: 1000. I have implemented k-fold validation as well. In this case, I have used k = 5. The average RMSE vs iterations for the training and validation (accidentally mentioned as testing in the below graph) set has been plotted as follows:




Training and Validation errors are given as follows:

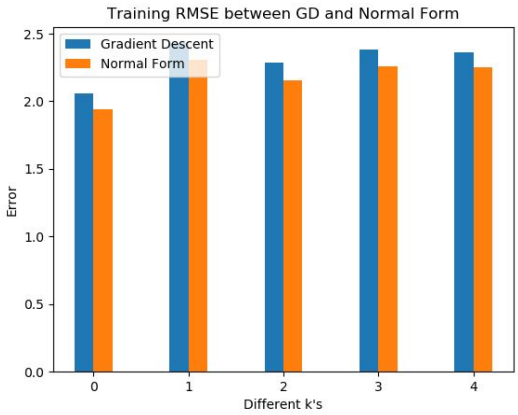
```
Training error for fold number: =  0 :  2.0585180660678764
Validation error for fold number: =  0 :  3.1605473220242053

Training error for fold number: =  1 :  2.4297372193525737
Validation error for fold number: =  1 :  1.30811009401006
```

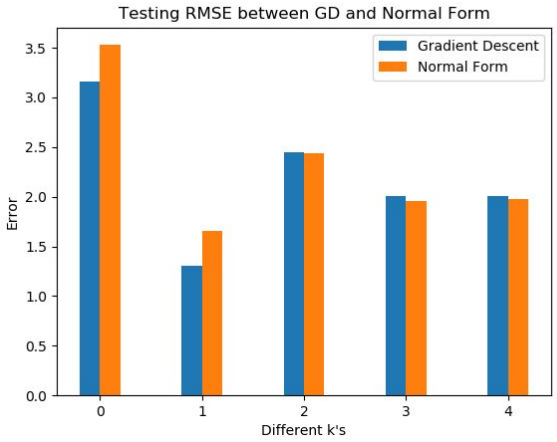
```
Training error for fold number: =  2 :  2.2884699039383687
Validation error for fold number: =  2 :  2.450173831224953

Training error for fold number: =  3 :  2.384901645838306
Validation error for fold number: =  3 :  2.006047205462333

Training error for fold number: =  4 :  2.3648094460672584
Validation error for fold number: =  4 :  2.008517726912091

Average train error:  2.305287256252877
Average validation error:  2.186679235926728
```

Final RMSE values for using the Normal equations are given as:

```
Training error for fold number: =  0 :  1.9429775953591537
Validation error for fold number: =  0 :  3.5283513819355945

Training error for fold number: =  1 :  2.309082074895573
Validation error for fold number: =  1 :  1.654603780659608

Training error for fold number: =  2 :  2.1546109517250116
Validation error for fold number: =  2 :  2.4379772820413557

Training error for fold number: =  3 :  2.2579507835280186
Validation error for fold number: =  3 :  1.9551937922522502

Training error for fold number: =  4 :  2.2483955945684877
Validation error for fold number: =  4 :  1.9728616570660953

Average train error:  2.1826034000152488
Average validation error:  2.309797578790981
```

This graph summarizes the difference between the gradient descent RMSE and Normal Equation RMSE values.



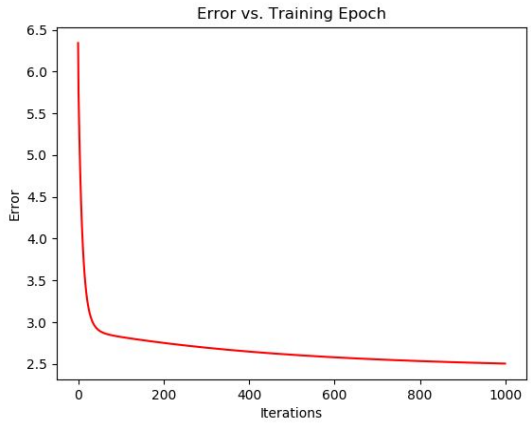

RMSE calculated using the normal equations is less than the RMSE from gradient descent. This is because in part 'b' we are using the analytical/normal solution of the linear regression problem, whereas in part 'a' we are computing everything manually using the gradient descent. The gradient descent algorithm depends upon the step-size and the number of epochs as well. If we use a bigger step-size, we might not end up at the minimum value of the required function, whereas if we use a smaller step-size, computation could take an indefinite amount of time. The number of iterations/epochs determine how long do we want to run the gradient descent algorithm. A smaller number, would lead to termination of the program before we reach the minima, whereas a bigger number would lead to more computation.

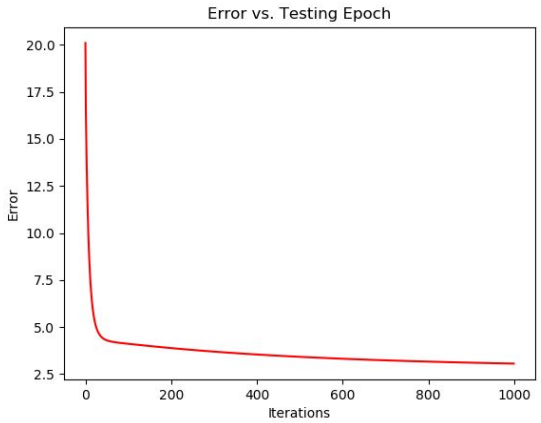
Hence, we the RMSE depends upon these two parameters, and it varies as we change them. On the other hand, the closed/normal form produces a perfect solution to the linear regression problem, but can be computationally  expensive based on the size of the data, as it requires inverting matrices and their multiplication as well.

## Part 2:

Ridge Regularization:

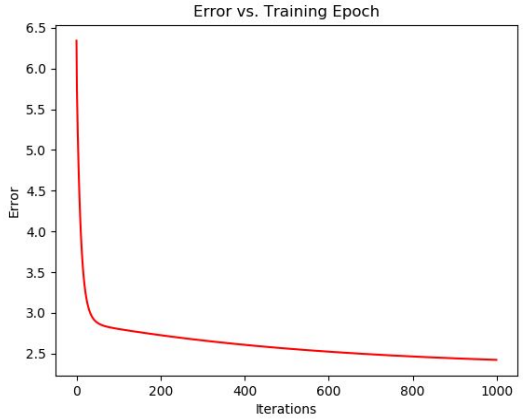
```
Finding the best hyperparameter for ridge regularization
Best hyperparameter for Ridge regularization:  0.7996554525892349
Training error for fold number: =  0 :  2.3335014402244303
Validation error for fold number: =  0 :  3.2623872224755344

Training error for fold number: =  1 :  2.5625566585540165
Validation error for fold number: =  1 :  2.4597996297202043

Training error for fold number: =  2 :  2.491450579601893
Validation error for fold number: =  2 :  2.818416208193973

Training error for fold number: =  3 :  2.5462710732527265
Validation error for fold number: =  3 :  2.4915289671128904

Training error for fold number: =  4 :  2.590348715060528
Validation error for fold number: =  4 :  2.100913620858779

Average train error:  2.504825693338719
Average validation error:  2.626609129672276
Test error: =  1.4973924574219633
```

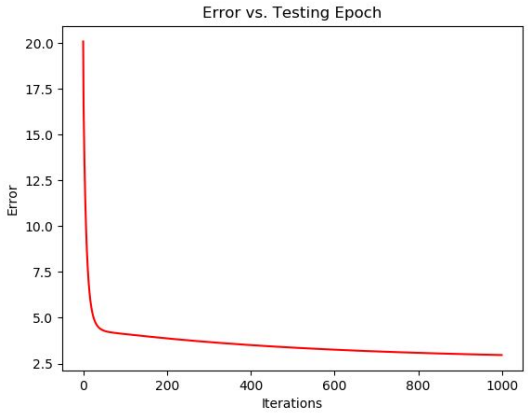


Lasso Regularization:

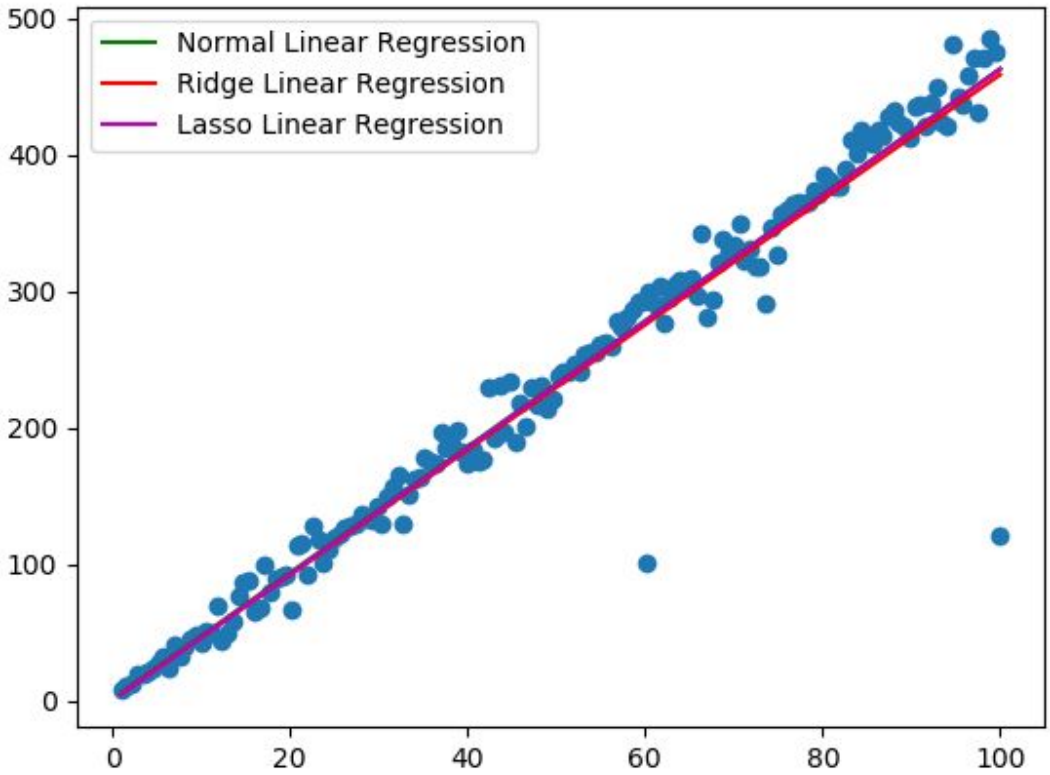
```
Finding the best hyperparameter for lasso regularization
Best hyperparameter for lasso regularization:  0.10280447320933092
Training error for fold number: =  0 :  2.2596076258834055
Validation error for fold number: =  0 :  3.0358884425269816

Training error for fold number: =  1 :  2.4775506106012104
Validation error for fold number: =  1 :  2.238702509644137

Training error for fold number: =  2 :  2.416044270291229
Validation error for fold number: =  2 :  2.593654946399076

Training error for fold number: =  3 :  2.4602355905740274
Validation error for fold number: =  3 :  2.271737881688197

Training error for fold number: =  4 :  2.5036126233401395
Validation error for fold number: =  4 :  1.8793048027042418

Average train error:  2.4234101441380025
Average test error:  2.4038577165925266
Test error: =  1.3272342068225447
```



## Part 3:

```
32.86766090878762 # error without regularization
33.096157721047646 # error for ridge
32.86766090878762 # error for lasso
```




All the lines are plotted together. These lines are overlapping with each other, which can be explained by the small change in the slope of the regressor or the small change in the `\theta`.
The small change can be observed in the RMSE values, but it is almost the same fit. Regularization has little effect on the model because of the good quality of data, as it lacks from noise, and hence already has a lower variance.

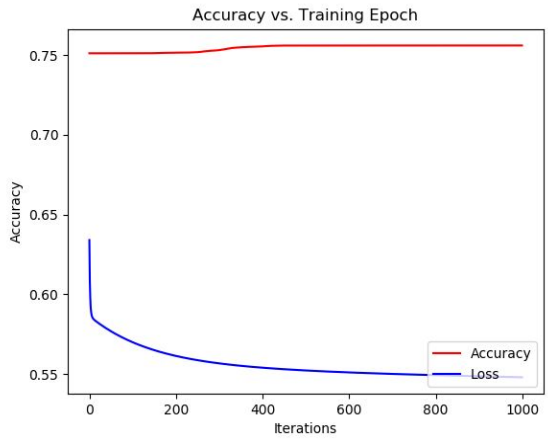
# Question 2

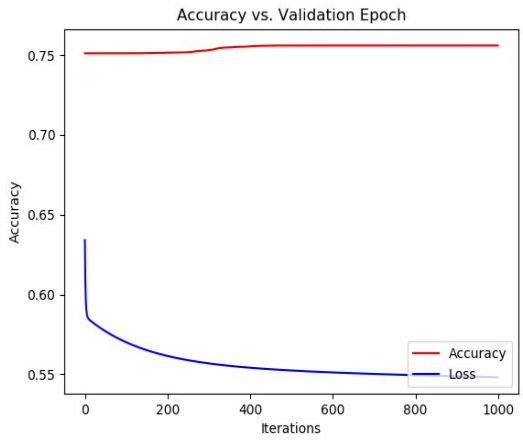
## Part 1

The complete data is used by implementing k-fold cross validation, with k = 5. Then the best model is used (with lowest validation error) on the test-set.

Without Regularization:

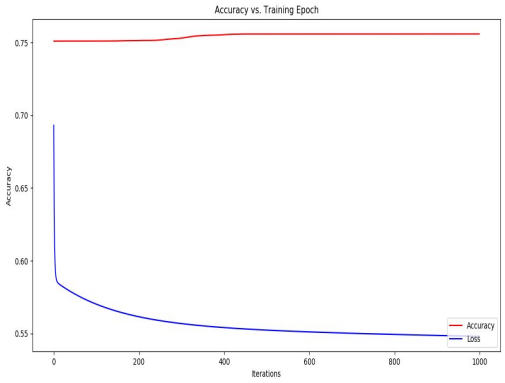
```
Training accuracy for fold #0: 0.7562683907331427
Validation accuracy for fold #0: 0.7548483341621084

Training accuracy for fold #1: 0.7545277466948486
Validation accuracy for fold #1: 0.7618100447538538

Training accuracy for fold #2: 0.7552009946125156
Validation accuracy for fold #2: 0.7591180371352785

Training accuracy for fold #3: 0.7565685868213842
Validation accuracy for fold #3: 0.7536472148541115

Training accuracy for fold #4: 0.7573559883961873
Validation accuracy for fold #4: 0.7504973474801061

Test accuracy: 0.7596945551128818
Average train accuracy:  0.7559843414516157
Average validation accuracy:  0.7559841956770916
```

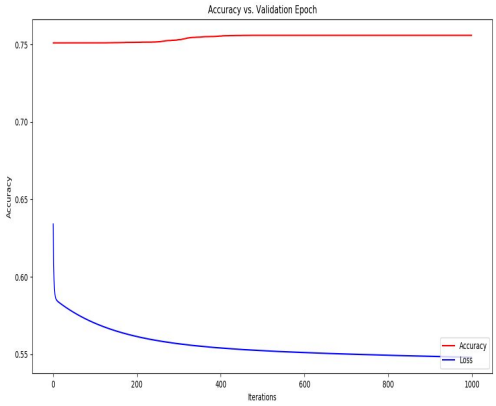


Ridge Regularization:

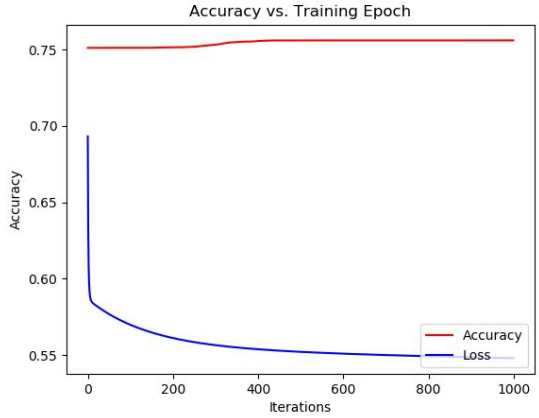
```
Ridge Logistic Regression
Using a alpha_ridge value of 1.2
Training accuracy for fold #0: 0.7562683907331427
Validation accuracy for fold #0: 0.7548483341621084

Training accuracy for fold #1: 0.7545277466948486
Validation accuracy for fold #1: 0.7618100447538538

Training accuracy for fold #2: 0.7552009946125156
Validation accuracy for fold #2: 0.7591180371352785

Training accuracy for fold #3: 0.7565685868213842
Validation accuracy for fold #3: 0.7536472148541115

Training accuracy for fold #4: 0.7573559883961873
Validation accuracy for fold #4: 0.7504973474801061

Test accuracy: 0.7596945551128818
Average train accuracy:  0.7559843414516157
Average validation accuracy:  0.7559841956770916
```

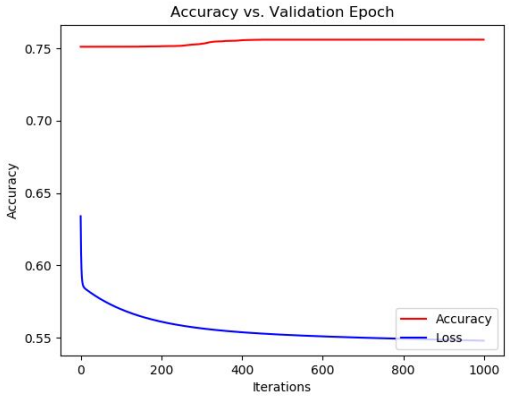


Lasso Regularization:

```
Lasso Logistic Regression
Using a alpha_lasso value of 1.2
Training accuracy for fold #0: 0.7562683907331427
Validation accuracy for fold #0: 0.7548483341621084

Training accuracy for fold #1: 0.7545277466948486
Validation accuracy for fold #1: 0.7618100447538538

Training accuracy for fold #2: 0.7552009946125156
Validation accuracy for fold #2: 0.7591180371352785

Training accuracy for fold #3: 0.7565685868213842
Validation accuracy for fold #3: 0.7536472148541115

Training accuracy for fold #4: 0.7573559883961873
Validation accuracy for fold #4: 0.7504973474801061

Test accuracy: 0.7596945551128818
Average train accuracy:  0.7559843414516157
Average validation accuracy:  0.7559841956770916
```




Small parameter coefficients can be approximated to zeros, which have little to no effect on the final cost. However, a non-zero parameter coefficient would add up to the final cost. Therefore, L1 regularization gives us a very sparse and varied solution.

## Part 2

```
Training accuracy for class 0 for L1 regularization:  0.9790646631774439
Training accuracy for class 0 for L2 regularization:  0.9790646631774439
testing accuracy for class 0 for L1 regularization:  0.9806122448979592
testing accuracy for class 0 for L2 regularization:  0.9795918367346939

Training accuracy for class 1 for L1 regularization:  0.9770097893800059
Training accuracy for class 1 for L2 regularization:  0.9774547611984574
testing accuracy for class 1 for L1 regularization:  0.9797356828193833
testing accuracy for class 1 for L2 regularization:  0.9806167400881057

Training accuracy for class 2 for L1 regularization:  0.9103726082578046
Training accuracy for class 2 for L2 regularization:  0.9110439744880833
testing accuracy for class 2 for L1 regularization:  0.8895348837209303
testing accuracy for class 2 for L2 regularization:  0.8895348837209303

Training accuracy for class 3 for L1 regularization:  0.897895938672321
Training accuracy for class 3 for L2 regularization:  0.8974066220844886
testing accuracy for class 3 for L1 regularization:  0.9089108910891089
testing accuracy for class 3 for L2 regularization:  0.906930693069307

Training accuracy for class 4 for L1 regularization:  0.9400890106128038
Training accuracy for class 4 for L2 regularization:  0.9404313591235878
testing accuracy for class 4 for L1 regularization:  0.9338085539714868
testing accuracy for class 4 for L2 regularization:  0.9327902240325866

Training accuracy for class 5 for L1 regularization:  0.8876591034864416
Training accuracy for class 5 for L2 regularization:  0.8874746356760745
testing accuracy for class 5 for L1 regularization:  0.8677130044843049
testing accuracy for class 5 for L2 regularization:  0.8654708520179372

Training accuracy for class 6 for L1 regularization:  0.9645150388644812
Training accuracy for class 6 for L2 regularization:  0.964346062859074
testing accuracy for class 6 for L1 regularization:  0.9467640918580376
testing accuracy for class 6 for L2 regularization:  0.9457202505219207

Training accuracy for class 7 for L1 regularization:  0.9420590582601756
Training accuracy for class 7 for L2 regularization:  0.9434956105347166
testing accuracy for class 7 for L1 regularization:  0.9241245136186771
testing accuracy for class 7 for L2 regularization:  0.9241245136186771
```

```
Training accuracy for class 8 for L1 regularization:  0.8788241326269014
Training accuracy for class 8 for L2 regularization:  0.8793368654930781
testing accuracy for class 8 for L1 regularization:  0.8716632443531828
testing accuracy for class 8 for L2 regularization:  0.8696098562628337

Training accuracy for class 9 for L1 regularization:  0.893427466801143
Training accuracy for class 9 for L2 regularization:  0.8940998487140696
testing accuracy for class 9 for L1 regularization:  0.889990089197225
testing accuracy for class 9 for L2 regularization:  0.8889990089197225
```

It looks like a good fit (neither an overfit nor much of an underfit), because the training accuracy is similar to the testing accuracy.

## Part 3