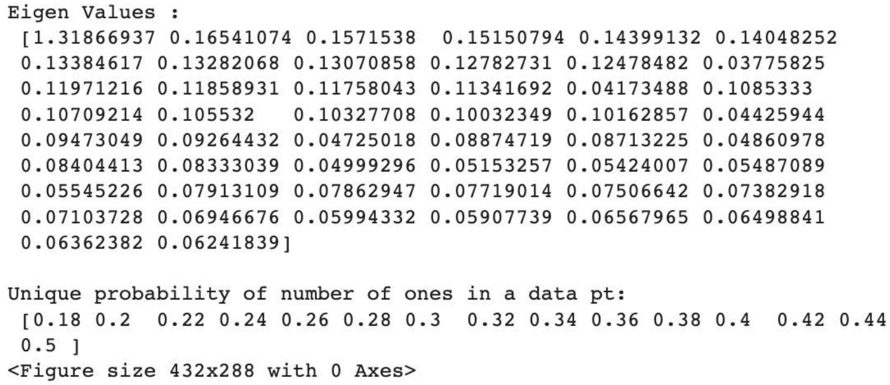
PRML ASSIGNMENT-2

-By Pranjul Dubey
Roll No. ED21S011

Question 1.

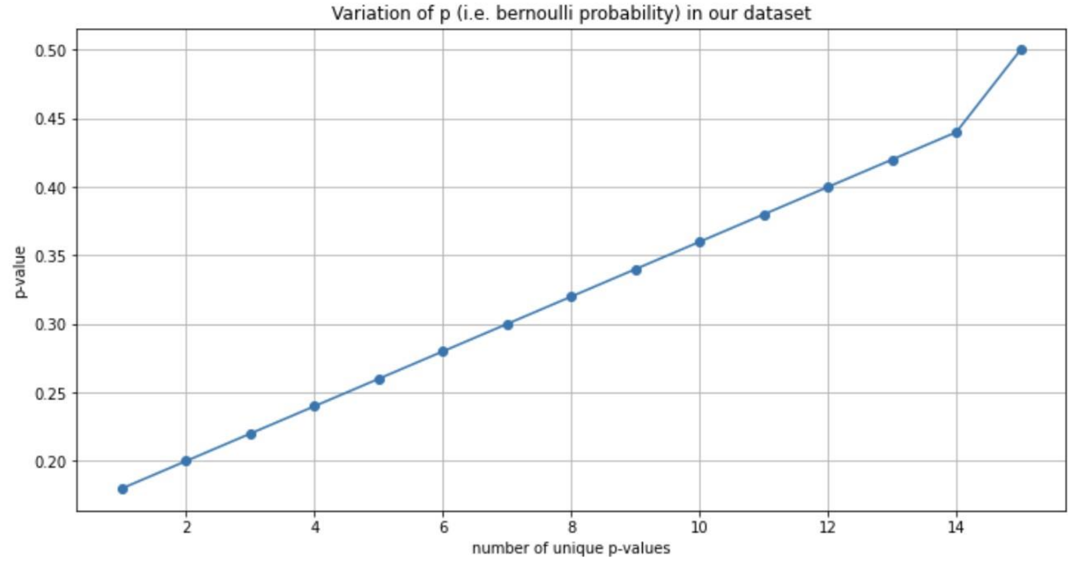
PART 1(i):

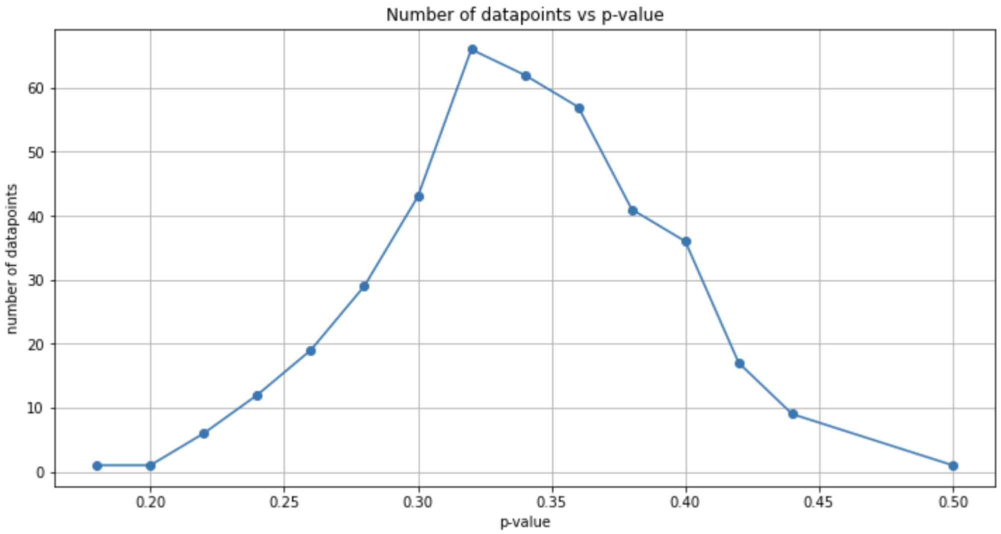
PRELIMINARY DATASET EXPLORATION:



```
Eigen Values :
 [1.31866937 0.16541074 0.1571538  0.15150794 0.14399132 0.14048252
 0.13384617 0.13282068 0.13070858 0.12782731 0.12478482 0.03775825
 0.11971216 0.11858931 0.11758043 0.11341692 0.04173488 0.1085333
 0.10709214 0.105532   0.10327708 0.10032349 0.10162857 0.04425944
 0.09473049 0.09264432 0.04725018 0.08874719 0.08713225 0.04860978
 0.08404413 0.08333039 0.04999296 0.05153257 0.05424007 0.05487089
 0.05545226 0.07913109 0.07862947 0.07719014 0.07506642 0.07382918
 0.07103728 0.06946676 0.05994332 0.05907739 0.06567965 0.06498841
 0.06362382 0.06241839]

Unique probability of number of ones in a data pt:
 [0.18 0.2  0.22 0.24 0.26 0.28 0.3  0.32 0.34 0.36 0.38 0.4  0.42 0.44
 0.5 ]
<Figure size 432x288 with 0 Axes>
```





The calculation of Eigenvalues of covariance matrix was to check if the given data lie in some lower dimension.
Finding: The eigenvalues are very close to each other hence the given dataset doesn't lie in lower dimension, so we cannot use PCA to lower the dimension and then use clusters to cluster data points.

The unique probability of finding ones in a particular datapoint in the whole dataset is a finite set and very small set.
Finding: The data points can be grouped using these unique probabilities.

I think the given data is generated from a Bernoulli mixture as the features are just binary.
So, the EM algorithm for this mixture is as follows:

$$L(p_1 \ldots p_k; x_1 \ldots x_n) = \prod_{i=1}^{n} P(x_i; p_1 \ldots p_k, \pi_i \ldots \pi_k)$$

$$\Rightarrow \pi \quad = \prod_{i=1}^{n} \Big[\sum_{k=1}^{4} {}_{k} \ \text{Bernoulli}(x_i ; p_k) \Big]$$

Taking log on both sides,

$$\text{LogL}(p_1 \ldots p_k; x_1 \ldots x_n) = \sum_{i=1}^{n} \log \sum_{k=1}^{4} \pi_k \ Bernoulli(x_i; p_k)$$

Now, the above equation is not analytically solvable due to the summation term inside logarithm.
So, Let's introduce new parameters, i.e. λ_k^i

$$\text{LogL}(p_1 \ldots p_k; x_1 \ldots x_n) = \sum_{i=1}^{n} log \sum_{k=1}^{4} \lambda_k^i \left(\frac{\pi_k \, Bernoulli(x_i; p_k)}{\lambda_k^i}\right)$$

By using Jensen’s Inequality, we can rewrite above equation as:

$$\geq \sum_{i=1}^{n} \sum_{k=1}^{4} \lambda_k^i \, log\left(\frac{\pi_k \, Bernoulli(x_i; p_k)}{\lambda_k^i}\right)$$

$$\geq \sum_{i=1}^{n} \sum_{k=1}^{4} \lambda_k^i \, log\left(\frac{\pi_k \, [(p_k)^{x_i} . (1-p_k)^{1-x_i}]}{\lambda_k^i}\right)$$

Now, Differentiating w.r.t. p_k and equating to zero (dropping the terms which don’t depend on p_k), we get

$$\frac{\partial L}{\partial p_k} = \sum_{i=1}^{n} \left(\frac{\lambda_k^i x_i}{p_k} + \frac{\lambda_k^i . (1-x_i) . (-1)}{(1-p_k)}\right) =$$

$$\Rightarrow (1 - p_k) \sum_{i=1}^{n} \lambda_k^i x_i + p_k \sum_{i=1}^{n} \lambda_k^i (x_i - 1) = 0$$

$$\Rightarrow \widehat{p_k} \quad \frac{\sum_{i=1}^{n} \lambda_k^l x_i}{\sum_{i=1}^{n} {}_k^i}$$

Now, As discussed in the class,

$$\lambda_k^i = \frac{P(x_i | z_i = k) . P(z_i = k)}{P(x_i)}$$

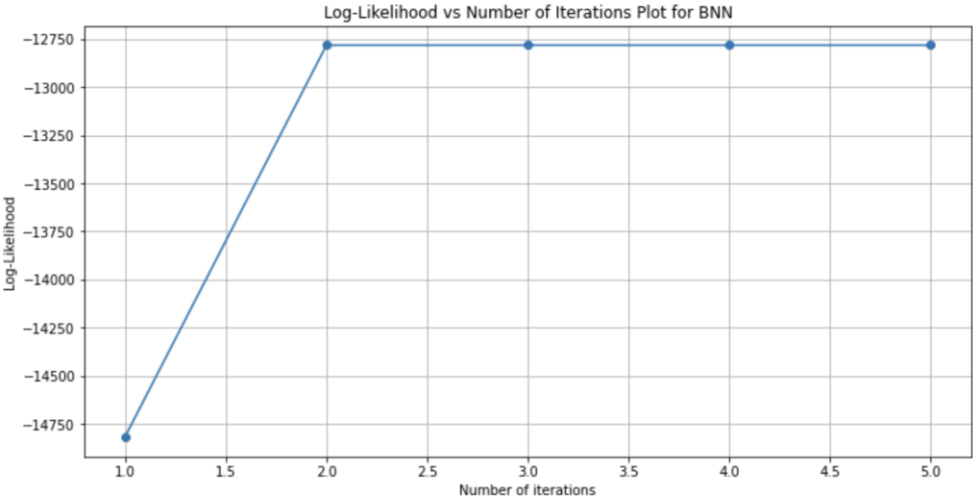
So, for our case,

$$\Rightarrow \widehat{\lambda_k^i} = \frac{\pi_k . (p_k)^{x_i} . (1-p_k)^{1-_i}}{\sum_{l=1}^{4} \pi_l . (p_l)^{x_i} . \quad {}_l^{1-_i}}$$

Our π_k’s will be same as discussed for GMMs,

$$\Rightarrow \widehat{\pi_k} \quad \frac{\sum_{i=1}^{n} \lambda_k^i}{n}$$

Upon averaging over 100 random initializations the plot of log-likelihood as a function of iterations looks like following:

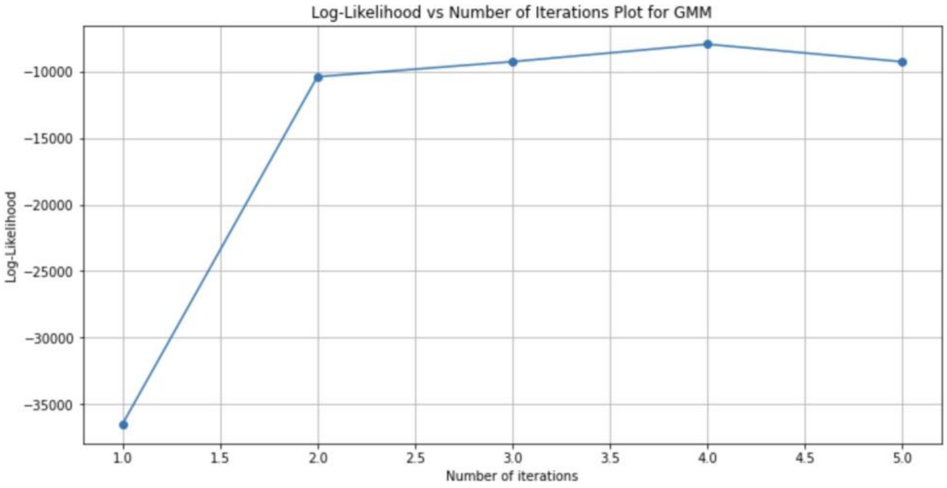


```
p-value of 4 Bernoulli mixtures:  [0.36979266 0.41270601 0.39350456 0.34871119]
```

Reference:P. Panagiotis,R. Magnus(2017),BayesBinMix: an R Package for Model Based Clustering of Multivariate Binary Data,arXiv

<u>*PART 1(ii):*</u>

Upon averaging over 100 random initializations the plot of log-likelihood as a function of iterations looks like following:

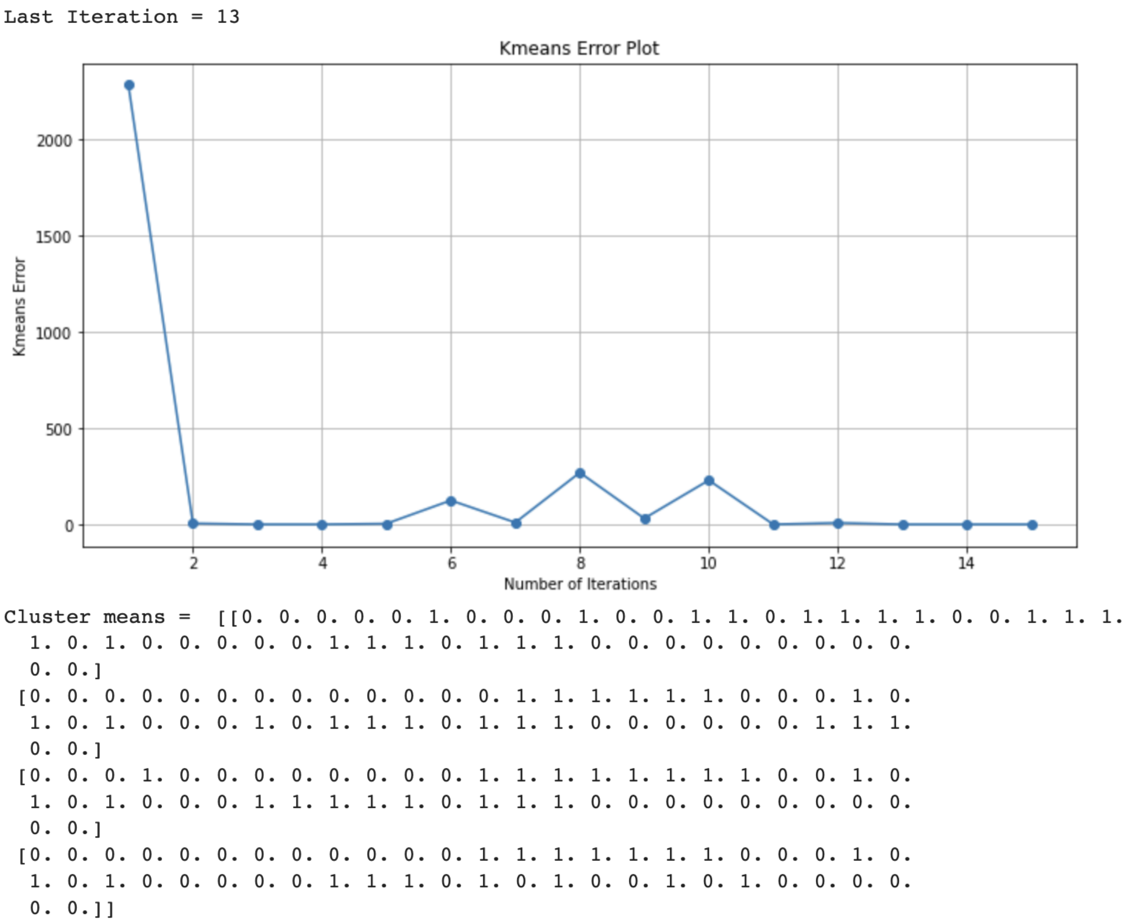


```
mu-value (i.e. mean) of 4 Gaussian mixtures:  [0.62818318 0.64918262 0.57925355 0.63883163]
sigma-sq-value (i.e. Variance) of 4 Gaussian mixtures:  [0.2096345  0.21349439 0.18560429 0.20813171]
```

Both Bernoulli mixture model and Gaussian mixture model try to maximize likelihoods. From the mean value of clusters obtained in GMM, it shows that it tries to cluster the datapoints about 0 and 1. Whereas, BMM cluster datapoints around clusters with different probabilities.

PART 1(iii):

K-means clustering objective plot (i.e. error plot) is shown below:



```
Last Iteration = 13
```

```
Cluster means =  [[0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 1. 1. 0. 1. 1. 1. 1. 0. 0. 1. 1. 1.
  1. 0. 1. 0. 0. 0. 0. 0. 1. 1. 1. 0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 1. 0. 0. 0. 1. 0.
  1. 0. 1. 0. 0. 0. 1. 0. 1. 1. 1. 0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 1. 1. 1.
  0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 1. 0.
  1. 0. 1. 0. 0. 0. 1. 1. 1. 1. 1. 0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 1. 1. 0. 0. 0. 1. 0.
  1. 0. 1. 0. 0. 0. 0. 0. 1. 1. 1. 0. 1. 0. 1. 0. 0. 1. 0. 1. 0. 0. 0. 0.
  0. 0.]]
```

PART 1(iv):

From the above observations for the given dataset, I would choose Bernoulli Mixture model as it clusters the datapoints w.r.t. different probabilities. We cannot use K-means clustering as it doesnot make much sense as the features are binary. I wouldn’t choose GMM because from the obtained gaussian means and variances it is clear that they are very close to each other and it wouldn’t give any fruitful information/result.

Question 2.

PART 2(i):

For calculating W_{ML} Analytically, we use the formula:

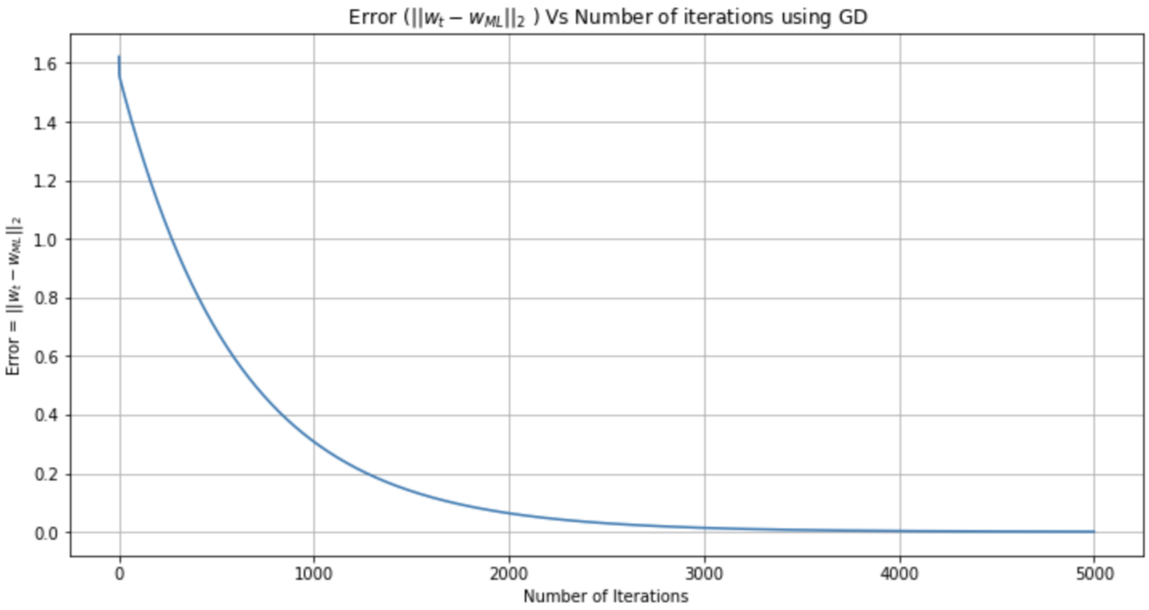
$W_{ML}=(XX^T)^{-1}Xy$

Where X dimensions= 100 x num_datapoints and y dimensions= num_datapoints

PART 2(ii):

Hyperparameters:

```
batch size = all datapoints at once
w is initialized with all zeros.
step_size=0.01
num_iterations=5000
```

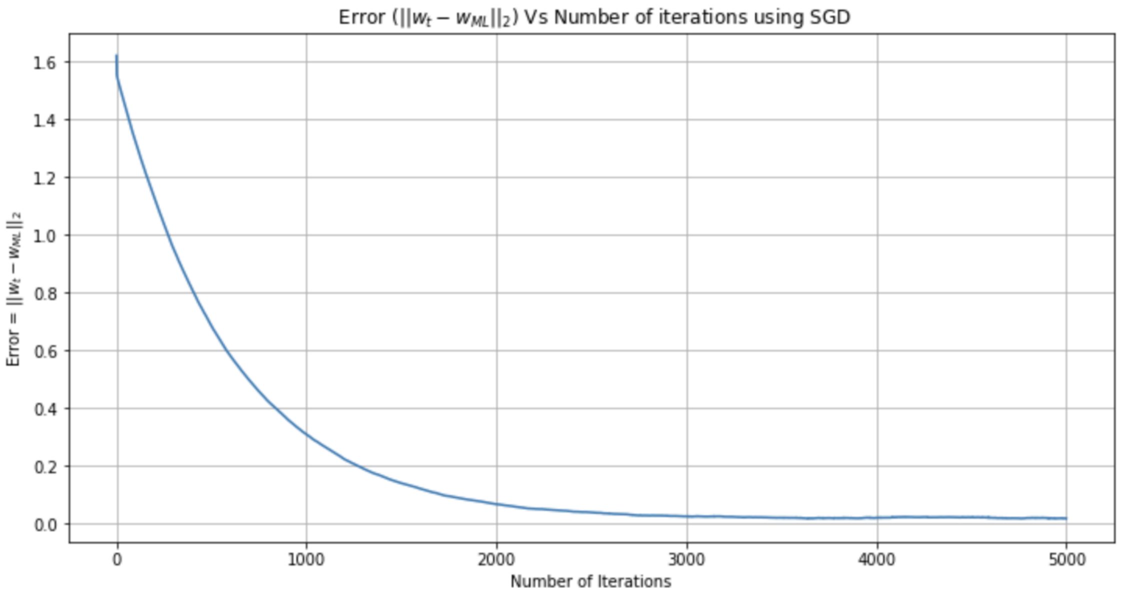


Gradient Descent converges to 0 error.

PART 2(iii):

Hyperparameters:

```
batch size = 100
w is initialized with all zeros.
step_size=0.01
num_iterations=5000
```



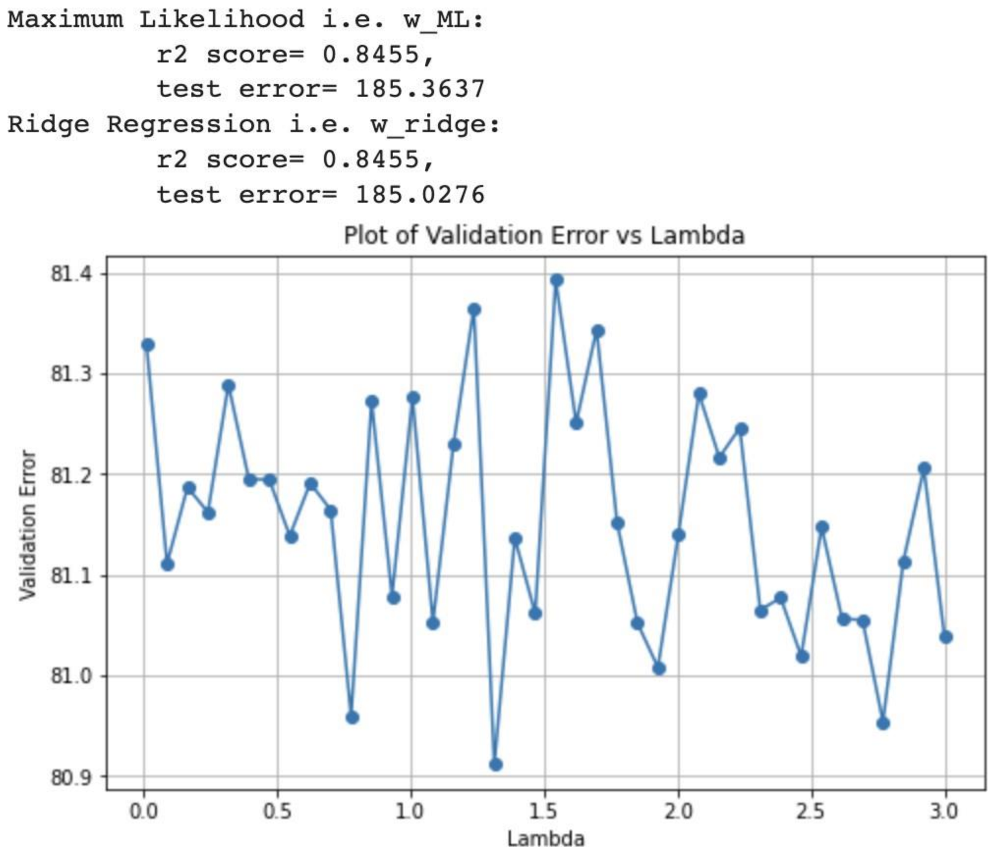
After using the same hyperparameters except batch size, we see that it takes some more number of iterations for SGD to converge (since we are dealing with small batches, the convergence depends on the quality of batch). However, the training time per iteration is reduced.

For the dataset that we have, the training time per iteration is not much pronounced but it becomes critical when we are dealing with a very large dataset which cannot be handled in a single computation because of large computation hardware requirements.

In the experiments conducted above, Gradient descent is found to be better than Stochastic Gradient Descent.

PART 2(iv):

For this experiment, I took a 40 lambdas in the range of (0.01,3) and used 5 fold cross validation to arrive at a lambda having minimum validation error.



```
Maximum Likelihood i.e. w_ML:
         r2 score= 0.8455,
         test error= 185.3637
Ridge Regression i.e. w_ridge:
         r2 score= 0.8455,
         test error= 185.0276
```

W_{ridge} performs better than W_{ML} on the test set. This is because of the reason that bayesian estimator which is a biased estimator has lesser error for some lambda (By Existence Theorem) as compared to the unbiased maximum likelihood estimator.

```
w_ML train error: 396.864
w_r train error: 396.872
```

For the given dataset and the performed experiment, I found that there is not much difference between the W_{ML} AND W_{Ridge} .

___________________________________xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx____________________________________