1.
Software tool used is LibSVM because of the following reasons:-
It has got a very rich and versatile set of APIs. The versatility lies in the fact that using a common interface one can implement SVM classifiers of various kernel types like linear, rbf, sigmoid etc. These APIs are also designed to accept various configuration parameters for the SVM Classifier. The return type of these API calls are also very informative and quite extensive. The return type object can be used for predicting any input feature vector, finding out hyper-plane coefficients while using in linear Kernel mode as well as number of Support Vectors used from either classes. Thus it has got everything I need.

Using the linear model, the cardinality of the Support Vector Set is 6625 (of which 3312 are from Class 1 and 3313 are from Class 3) and**the hyper-planer parameters are the followings [[-0.00111633 -0.02052913 -0.00064846 -0.00010827]]**

The Confusion Matrix, when the linear kernel SVM is applied on the training set is the following

$$\begin{matrix} 3515 & 1485 \\ 1432 & 3568 \end{matrix}$$

The first row of the Confusion Matrix represents true Class 1 and the second row represents true Class 2, whereas the 1st column represents computed Class 1 and 2nd column represents computed Class 2.
The accuracy of classification is 70.83 % for the linear kernel SVM

While using RBF Kernel, I am obtaining perfect classification and the Confusion Matrix looks like the following, with same meaning of the axes as the previous.

$$\begin{matrix} 5000 & 0 \\ 0 & 5000 \end{matrix}$$

As for the RBF kernel, the classification accuracy is 100 %. Here all the 10,000 vectors are used for coming up with the classifier.

2. Algorithm used is described below
Step a. First c number of classes are chosen which in our case will be 3, 4 and 5 in turn
Step b. Then from with the training data set, H, randomly c number of vectors are chosen. These c vectors will act as dummy means for each of the c classes respectively.
Step c. Each of the 15000 vectors in H are then classified depending upon the "nearness" (used Euclidean and Manhattan distances as the measure of nearness) of these vectors to the pre-chosen means. Thus is a vector is closest to the ith mean then that vector will be classified to class i.
Step d. Means are recomputed based on the current classification and steps c and d are repeated.
Step e.STOP the iterations once the means converges to a value
Using Euclidean distances here are the centroids

For c = 3
```
[[-70.72548809  16.00143305 -27.45824181 -58.57647383]
 [ 40.38019866  28.26561527 -24.62521793 -52.27950297]
 [ -0.3482364   51.50287921 -24.7268482   76.55301184]]
```

For c = 4
```
[[ 50.50599064 -12.45517988 -26.55975048 -49.02820159]
 [  9.15619035  79.23216045 -23.00196553 -45.03443127]
 [ -0.55100145  46.36279    -24.99159065  88.21848002]
 [-79.33735013   6.92000772 -27.67275869 -58.62591479]]
```

For c = 5

```
[[ -1.78971096 -20.49661376 -68.83360168 -36.55086293]
 [ -0.56064174  48.34152968 -23.35109797  92.53411098]
 [-85.76600047  11.30813085  -6.32436828 -63.96855326]
 [ 80.5202782   11.45732905  -2.25810182 -57.0397841 ]
 [ -0.5297544   80.58158822 -24.56290212 -40.73734688]]
```

The cardinality of each group for Euclidean distances are,
For c = 3   [4078, 6516, 4406]
For c = 4[3868, 4174, 3697, 3261]
For c = 5   [2694, 3426, 2556, 2500, 3824]

As is evident from the cardinality of the groups for c = 3, both class 1 and 3 has lost substantial member feature vectors whereas class 2 has gained 1516 feature vectors. In the training set H, means of class 1 and 3 were quiet close, but in the new clustering they are comparatively far apart. The original means of class 1 and 3 were only 15.637 units distances apart using Euclidean Distance measure, whereas after clustering they became 112.67 units distances apart. Not only that, average distances between 3 means of all the three classes has also widened a lot after clustering.

For c=4 and 5 the clusters became more spread out and more evenly distributed. One notable feature in the cardinality set of both these cases is that cardinality of the 1st cluster and the last cluster got significantly reduced and that reduction is mostly because of the outliers that were once part of class 1 and 3. It is my belief that these outliers got blended among each other and some became part of the middle classes.

Using Manhattan Distances here are the centroids
For c = 3

```
 [[-64.9030   13.4748  -27.1649  -56.6268],
   [29.9565   26.0166  -24.9164  -55.6467],
   [-0.2039   52.3997  -24.2703   52.7203]]
```

For c = 4

```
 [[ -70.7808   10.8533  -27.3242  -65.1164],
    [1.6935   75.9870  -24.9774  -34.1064],
   [-0.2687   42.9095  -23.8489   75.7907],
   [37.2492  -12.3072  -25.5716  -48.9505]]
```

For c = 5

```
  [[ -0.2233   44.4557  -23.9216   97.2484],
   [41.7830   38.9103  -22.5992  -84.9836],
    [2.6228  -23.5797  -28.4661  -26.0504],
  [-79.7010   18.1016  -27.7976  -65.0250],
   [ 0.1778   73.8014  -24.2998   -2.9643]]
```

The cardinality of each group for Manhattan distances are,

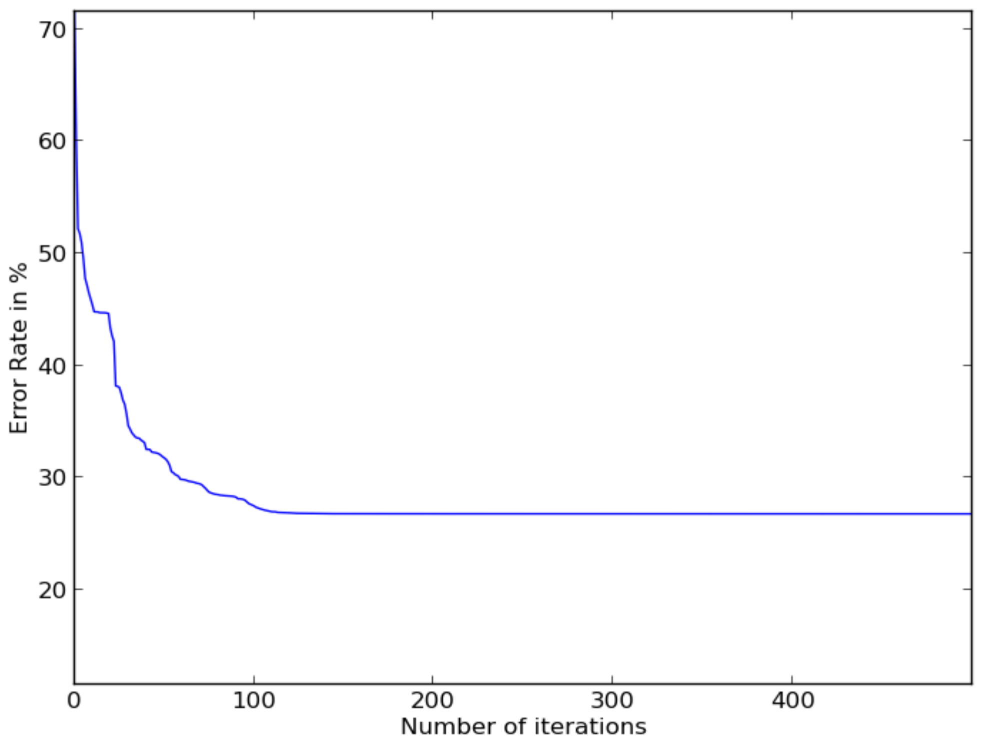
For c = 3	[3811, 5890, 5299]
For c = 4	[3123, 4267, 3835, 3775]
For c = 5	[2717, 2969, 3139, 2716, 3459]

In case of Manhattan distances, for c = 3, class 1 lost 1189 feature vectors where as both class 2 and 3 gained some feature vectors. Here also centroids of the classes are more comfortably spaced than what they were as per the original distributions. The average distances between the centroids are greater than that of the means of the original classes.

One notable nature of the feature vectors, irrespective of the distance measure adopted is that, the classes are very much intermingled with one another. When the number of classes are increased from 3 to 4 to 5 the cardinality of every classes has decreased every time which shows that there is not a single group of vectors that is completely separated from the herd.

3. Since it is quite evident that samples in class 1 and 3 are not linearly separable in the training set H, we need to have atleast 1 hidden layer in the ANN architecture. Since the vectors are 4 dimensional, the number of nodes used in the hidden layer is 9(2*d + 1).
I have used tanh function as my activation function. So, my target set consists of +1s for class 1 and -1 for class 3. In the output layer, I only have one node. If what is coming out from that node is +ve then I am classifying it as class 1 and if –ve then I am classifying it as class 3.
On the input layer side I have 5 nodes, 4 for each component of feature vector and the 1(the 1st one) for the bias.
The algorithm used is Feed Forward Back Propagation. Initially 2 setsof weights are chosen randomly, one for the edges joining the input nodes and the hidden layer nodes and the other joining the hidden layer nodes and the output node. At every node all the incoming input values are summed up and the resultant value is applied to the tanh function.
Each node in the hidden layer is getting input from all the 5 nodes in the input layer. All these inputs are added, after multiplying with the corresponding associated weights, and tanh function is applied on the sum. The result from each hidden layer node is then fed to the input of the output node, after multiplying with the associated weights. Here again all the inputs are added, and then tanh function is applied on the sum. Whatever result is obtained thus, is the net result of classification for the given feature vector.
Before beginning with the process, as a pre-processing step, first 1 is prepended to each feature vector (in the training set corresponding to class 1 and 3) as the 5th component. Then the vectors corresponding to class 3 are negated.
Using the initial random weights and feeding the feature vector components into the neural network each vector is classified. Since the weights are chosen randomly obviously there will be some vectors that will be some errors in the classification. The error is computed subtracting the computed classification and the target and is given by the following
E = T – O, where T is the target output and O is the computed one. Using the gradient descent approach by taking derivative of E with respect to the weights, and adjusting the weights by subtracting a value proportional to the derivative. The equation is given by the following
$W_i(k+1) = W_i(k) - \alpha * \partial E/\partial W_i$, where α is the learning rate. This is an iterative algorithm where 1st set of weights are chosen randomly and the subsequent ones are computed by using the above formula. This iteration will be continued until the error rate converges. This is done on both the hidden layer nodes and output node

The following graph illustrates the variation of error rate with number of iterations.



The learning rate chosen is 0.2 ..

The final set of weights connecting input layer nodes and hidden layer nodes are the followings. Each row consists of 4 values, corresponding to each dimension of the input vector and since there are 9 nodes in the hidden layer, there are 9 of them.

```
[ 1.66753733  8.02837013 -1.21193995  3.16894126]
[  1.66172278 -15.36159276 -19.46759257  -0.28540657]
[ -8.09896074e-05  -5.34060052e-04   4.80301629e-03   2.65224465e-05]
[-0.79191669 -2.10568957 -3.06144882  6.62438086]
[ 0.7848287  -1.95979801 -1.50777661  6.48192134]
[ 1.66391124 -9.29904858  4.75129982  4.5124326 ]
[-12.97454285  28.40920937  -0.3446582   -8.90738587]
[-11.94801362  23.3160322   -0.65584983  -6.58895638]
[ -1.51350779e-05   2.46443215e-04   3.73020160e-03  -1.51654116e-05]
```

The following is the set of final weight of the connection joining the nodes in the hidden layer and the output node. There are 9 of them as there are 9 nodes in that hidden layer and one output node only

```
[  1.87350927e-02  -4.65526089e-02  -2.93987849e+01  -5.12716639e-02
```

```
  5.84374074e-02   5.17948311e-02  -2.27952496e-01   2.88257100e-01
  2.57263759e+01]
```

The confusion matrix obtained after applying the above algorithm on the training set for class 1 and 3 is

```
[[ 4240.   760.]
 [ 1190.  3810.]]
```

The top row represents true class 1 and the bottom row true class 3, whereas the first column represents computed class 1 classification and 2nd column computed class 3 classification.
The Confusion Matrix obtained after applying the algorithm on those vectors of test data that belongs to class 1 and 3 is given by the following

```
[[ 4225.   775. ]
 [ 1260.  3740.]]
```

4. The approach adopted over here is pretty much same as the previous. The only difference is the final neural net consists of three smaller neural nets and the architecture of these neural nets is exactly same as the one described in part three. The target for the first neural net consists of 5000 1s followed by 10000 -1s, that of second NN consists of 5000 -1s followed by 5000 1s followed by 5000 -1s and that of third consists of 10000 -1s followed by 5000 1s. So the neural nets are trained in a way so as to discriminate between class i and rest for i=1,2,3 . For training all the 15000 sample vectors from H are used.
This neural networkconsists of 3 nodes in the output node, one each from the smaller neural nets. A feature vector will be classified as class i, if the ith node in the o/p layer is positive and all other nodes are negative.

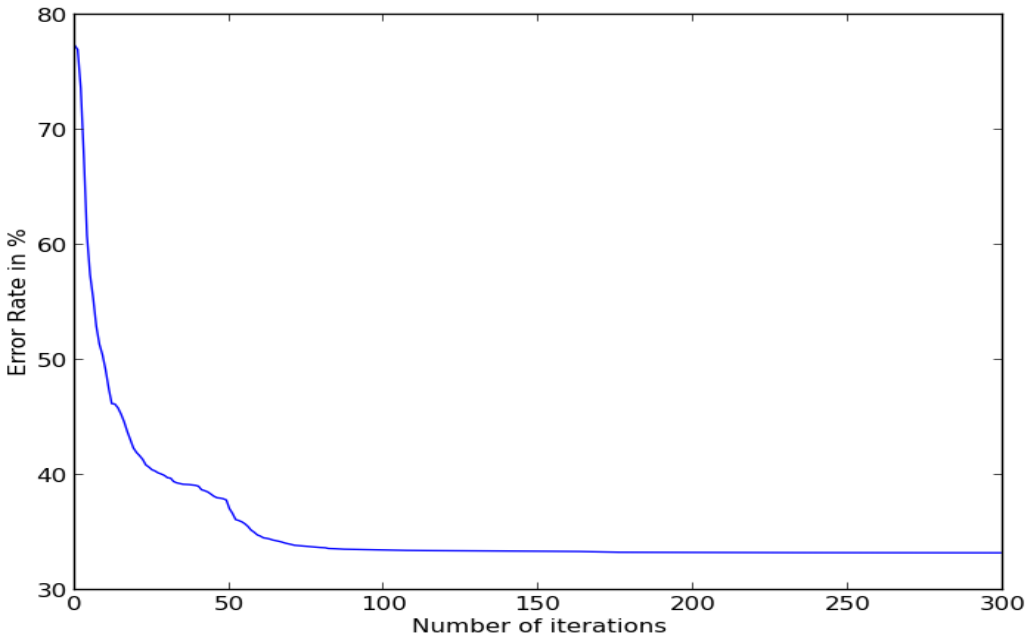
The connection weights for 1st neural net which lies between input layer and hidden layer are the followings

```
[-0.03953128  1.13293587 -3.63581509  0.74335857]
[-0.09484411 -0.00424504  0.00274509 -0.00396137]
[ 0.00284348 -0.00322799  0.02996712 -0.00149334]
[ 0.13113015 -0.00216339 -0.00097579 -0.00354836]
[-0.00148672  0.00174336  0.01017961  0.00176397]
[-0.00457802  0.00019776 -0.02112526  0.0006734 ]
[-0.2848224   0.19388398  0.49972138 -0.11556082]
[-0.04954371  0.15631313 -1.7534734   0.73254997]
[-0.00133291 -0.0007941  -0.0027326   0.00065304]
```

The connection weights for 1st neural net which lies between hidden layer and the output node are the followings

```
[ 0.12401386  1.78308997 -2.76684916  1.7228235   4.42045205  1.61122053
 -0.03728444 -0.17146642 -8.20053056]
```

Below graph depicts the variation of Error Rate with number of passes made while training the 1st NN



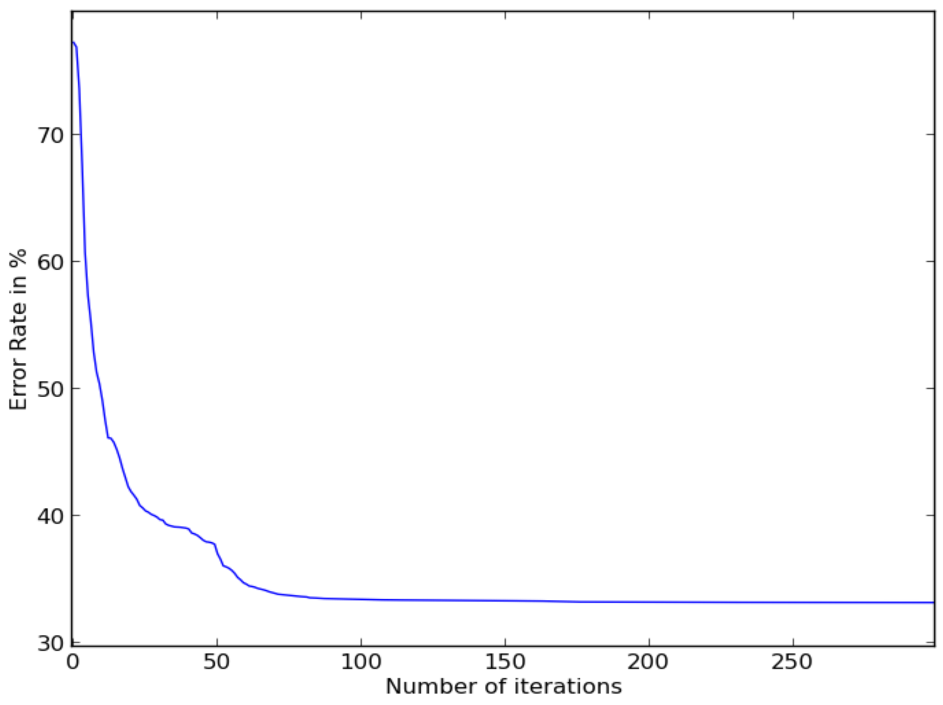
The connection weights for 2nd neural net which lies between input layer and hidden layer are the followings

```
[-2.1623276  -0.72376855  1.71911329  1.80364081]
[ 2.31322301 -0.3789228   7.34487575 -5.07393229]
[-11.70455276  -1.17759013   1.19984211   0.1664674 ]
[ -3.81793352   3.46258856  11.35382689  34.64925525]
[  0.5841481    9.19288888 -11.61085788  -7.65717785]
[ 3.14266465  0.06923941  0.19574188  0.14408517]
[-1.41784905 -1.73844842  2.77449413  5.21701551]
[ 3.89352961  1.00363615 -1.40127279  1.22771263]
[-2.42189287  0.05316754  0.14090691  0.07585365]
```

The connection weights for 2nd neural net which lies between hidden layer and the output node are the followings

```
[ 3.63248053 -3.83640096 -3.53448664  0.21754887 -0.13504524  1.1211335
 -0.2463538   0.49199997  2.2603195 ]
```

Below graph depicts the variation of Error Rate with number of passes made while training the 2nd NN



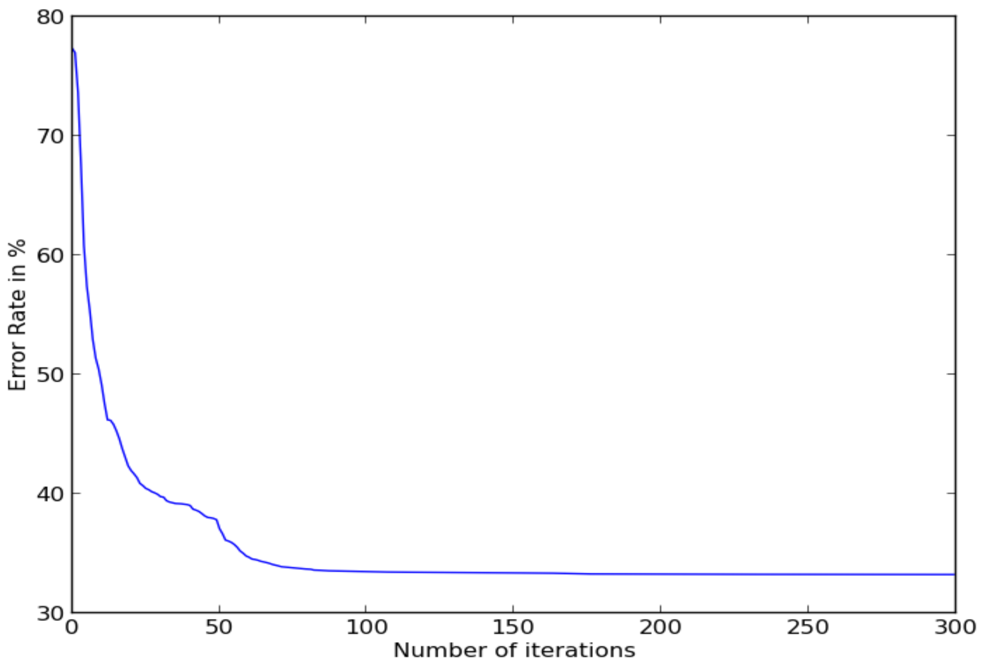
The connection weights for the 3rd Neural Net which lies between input layer and hidden layer are the followings

```
[ -1.44931324  10.55116966   4.24122265   2.06724187]
[-0.20487574 -0.25696062  0.27322951  1.7943912 ]
[ -8.04559955 -23.8832319   18.3277432    5.63552455]
[-0.0003006   0.00174483  0.01066396  0.00052508]
[-22.77293942 -58.60225093  16.98244165 -23.50935064]
[ 0.08848069  2.52344063 -0.9353845   0.67601323]
[-4.43836183 -1.33069967  1.74158357 -1.5704893 ]
[ -2.70996587 -12.20681973   1.22647137  -5.28090472]
[ -1.73755960e-05  -2.07967555e-03   1.04738399e-02  -1.65916633e-04]
```

The connection weights for 3nd neural net which lies between hidden layer and the output node are the followings

```
[-0.04148969 -0.29494279  0.04999261 -4.52874282 -0.11294745  0.02571685
  0.15321096  0.09637313  4.90255399]
```

Below graph depicts the variation of Error Rate with number of passes made while training the 3rd NN



Confusion Matrix on Test Data, where on the vertical axis I have plotted true classes 1,2, 3 from top to bottom in that order and on the horizontal axis, from left to right, the computed classes, 1,2,3 are plotted in that order.

```
[[ 4043.   310.   647.]
 [  441.  4269.   290.]
 [ 1329.   142.  3529.]]
```

Confusion Matrix on Training Set using the same layout is the following

```
[[ 4094.   281.   625.]
 [  381.  4357.   262.]
 [ 1286.   115.  3599.]]
```