Department of Computer Science & Engineering

CSE 574 – MACHINE LEARNING

Classification and Regression

Programming Assignment – 3

Presented By:

Pratik **Agarwal** [50290570]
Sourav **Bihani** [50290969]
Tannu **Priya** [50290768]

Group - 23

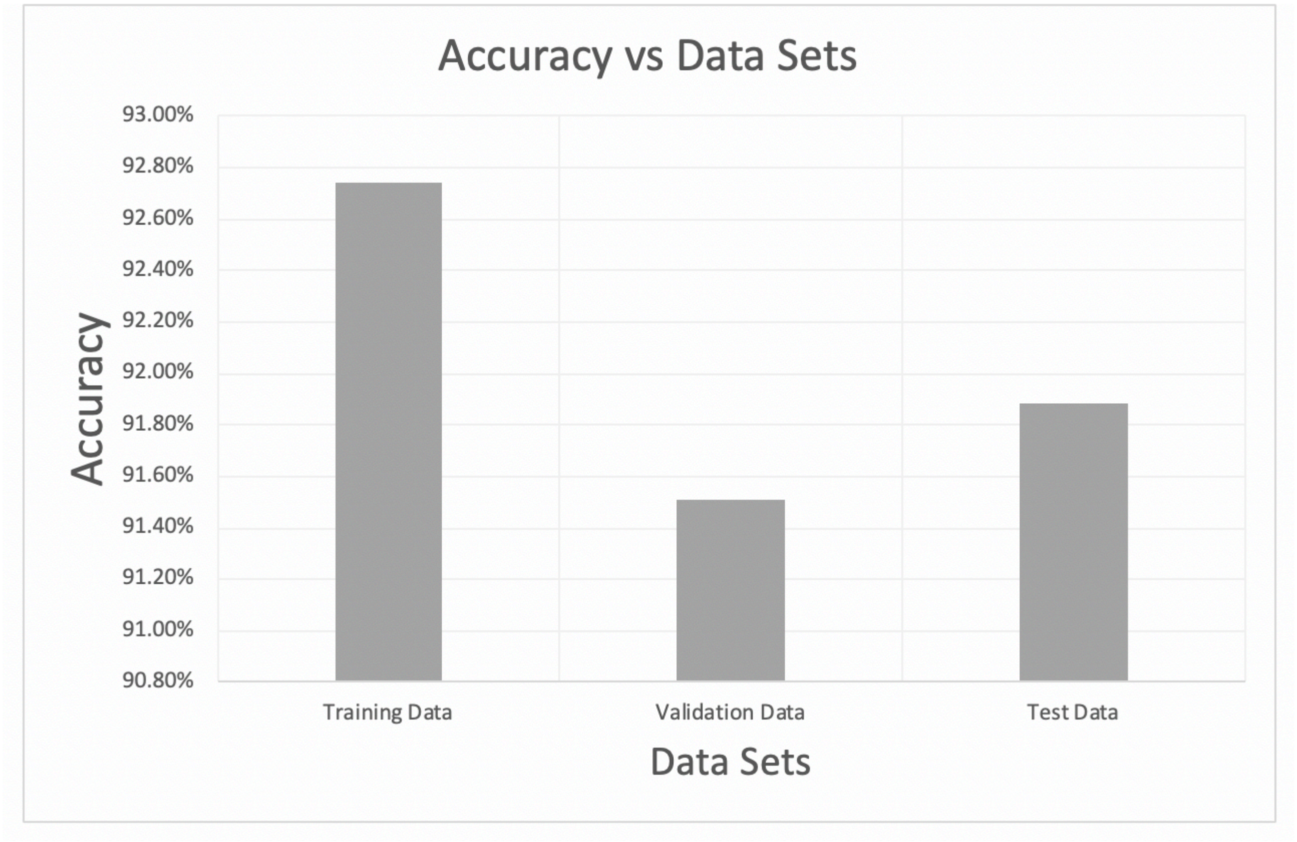
Table of Contents:

Problem 1: Implementation of Logistic Regression

The following table shows various accuracies obtained in Logistic Regression against varied data set:

Data	Accuracy
Training Data	92.746 %
Validation Data	91.51 %
Test Data	91.88 %

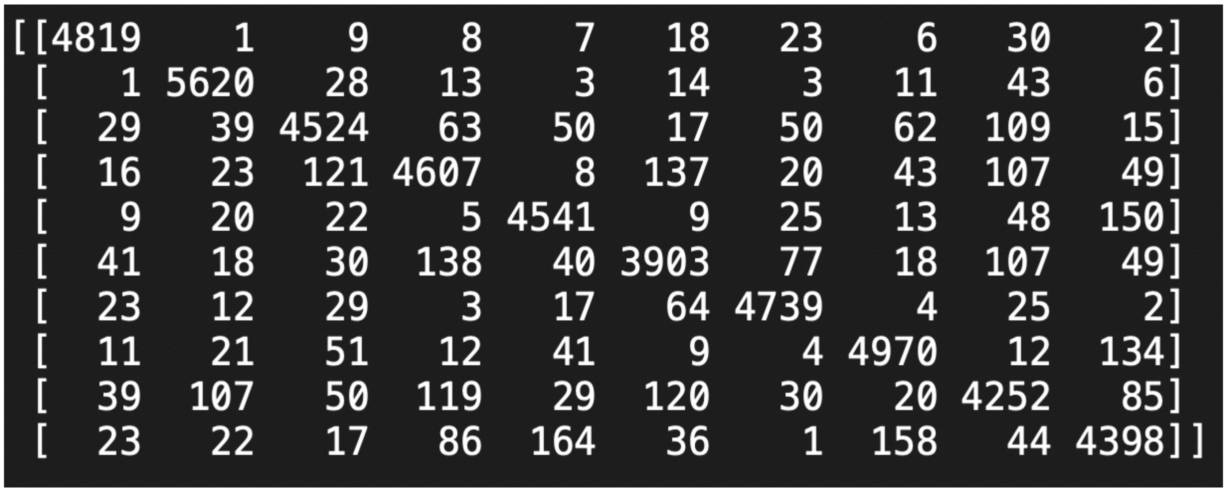
The total time taken by our code for Logistic Regression is 808.7755 seconds.



Category wise error:

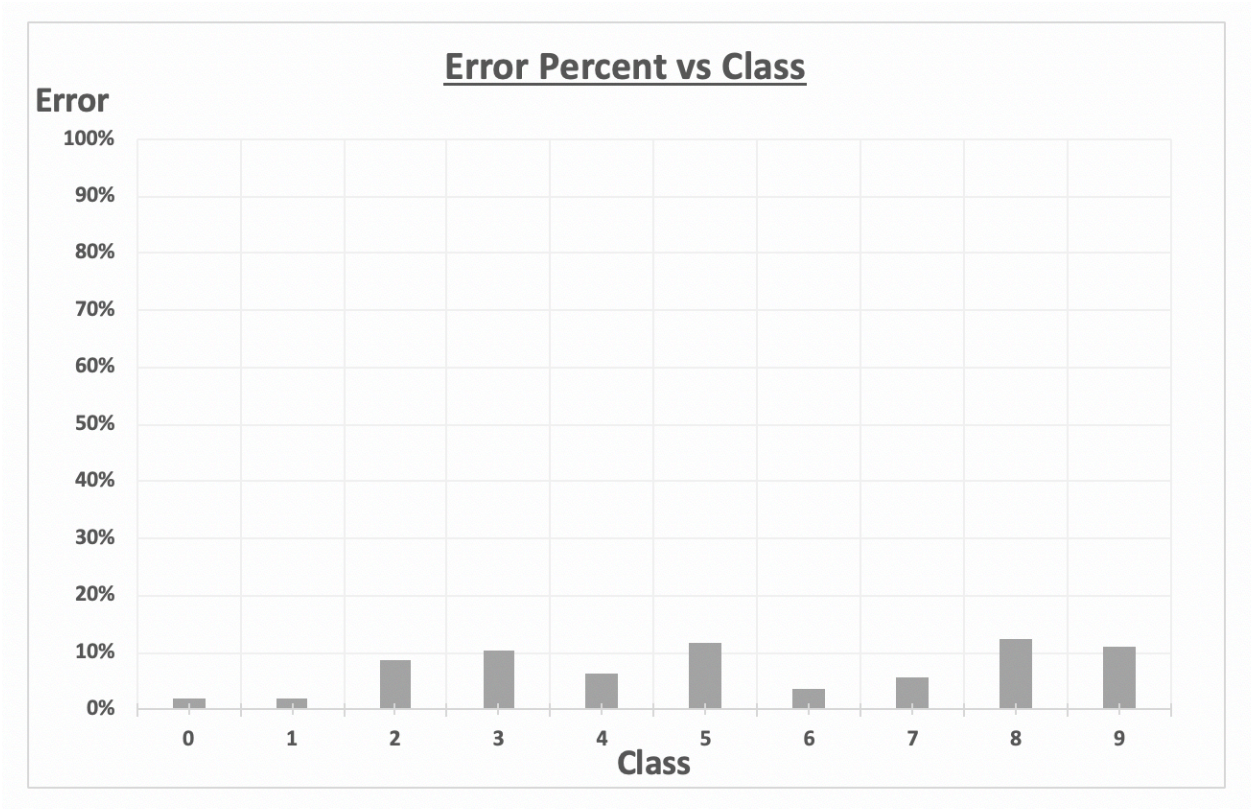
1. Training Data:

   Confusion Matrix:



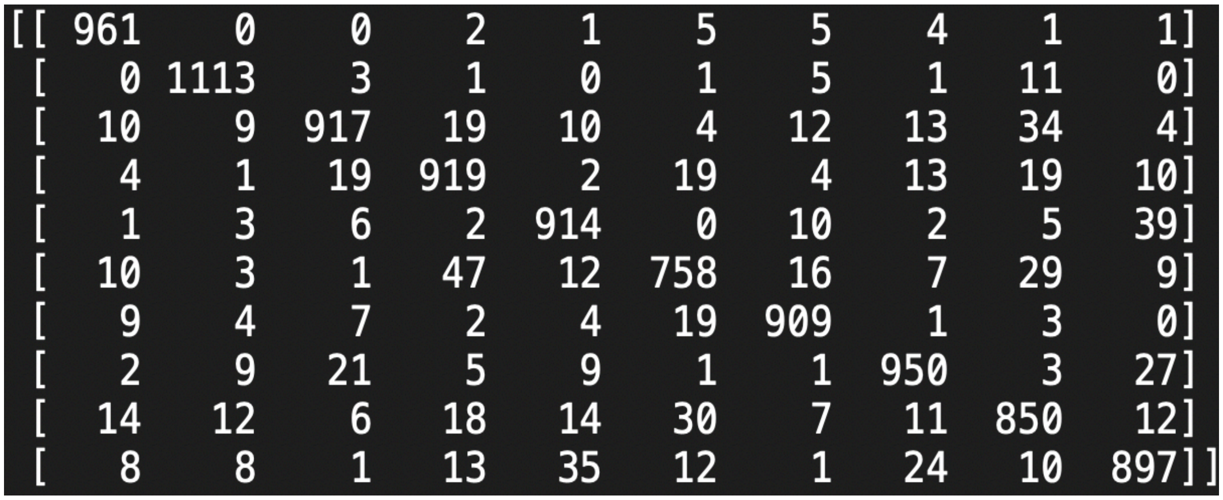
```
[[4819    1    9    8    7   18   23    6   30    2]
 [   1 5620   28   13    3   14    3   11   43    6]
 [  29   39 4524   63   50   17   50   62  109   15]
 [  16   23  121 4607    8  137   20   43  107   49]
 [   9   20   22    5 4541    9   25   13   48  150]
 [  41   18   30  138   40 3903   77   18  107   49]
 [  23   12   29    3   17   64 4739    4   25    2]
 [  11   21   51   12   41    9    4 4970   12  134]
 [  39  107   50  119   29  120   30   20 4252   85]
 [  23   22   17   86  164   36    1  158   44 4398]]
```

Class	Error %
0	2.11%
1	2.12%
2	8.75%
3	10.21%
4	6.22%
5	11.72%
6	3.64%
7	5.60%
8	12.35%
9	11.13%



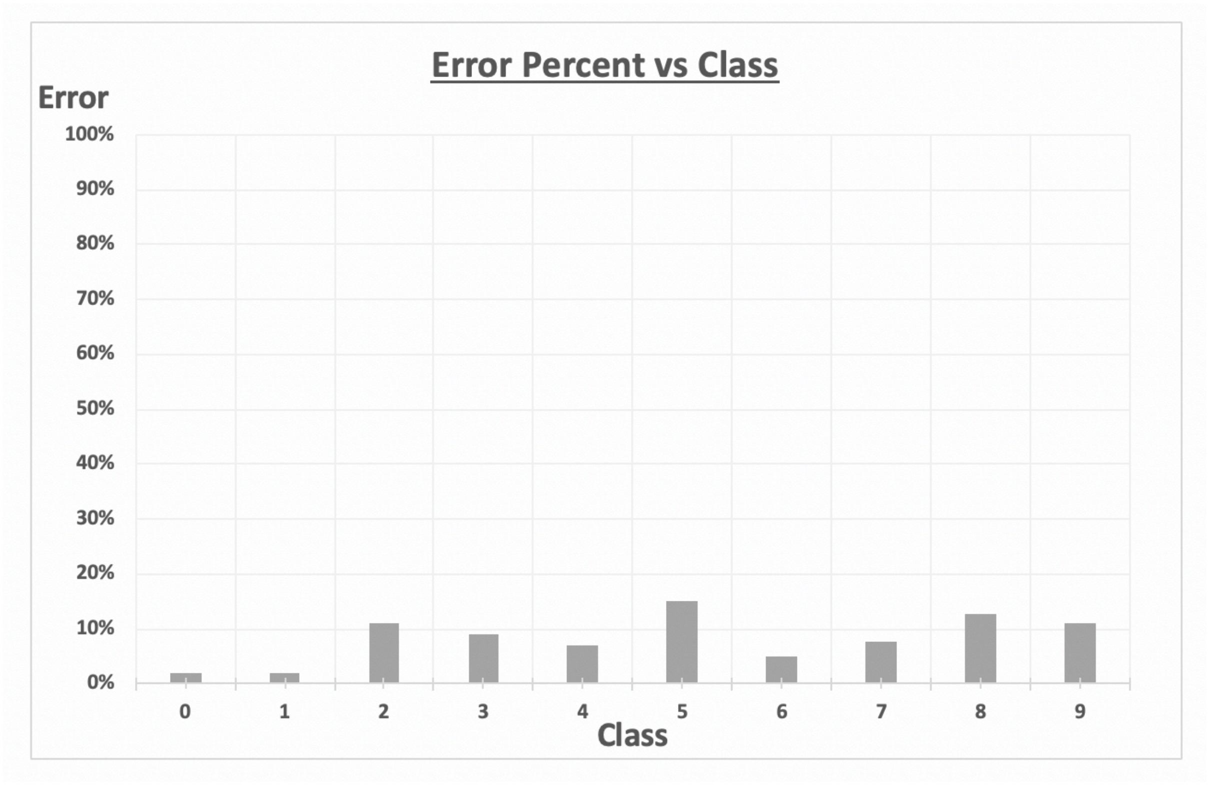
2. Test Data:

   Confusion Matrix:



```
[[ 961    0    0    2    1    5    5    4    1    1]
 [   0 1113    3    1    0    1    5    1   11    0]
 [  10    9  917   19   10    4   12   13   34    4]
 [   4    1   19  919    2   19    4   13   19   10]
 [   1    3    6    2  914    0   10    2    5   39]
 [  10    3    1   47   12  758   16    7   29    9]
 [   9    4    7    2    4   19  909    1    3    0]
 [   2    9   21    5    9    1    1  950    3   27]
 [  14   12    6   18   14   30    7   11  850   12]
 [   8    8    1   13   35   12    1   24   10  897]]
```

Class	Error
1	1.94%
2	1.94%
3	11.14%
4	9.01%
5	6.92%
6	15.02%
7	5.11%
8	7.59%
9	12.73%
10	11.10%



Our Implementation of Linear Regression incorporates building 10 binary classifiers (10 classes) to distinguish between different classes.

Conclusion:

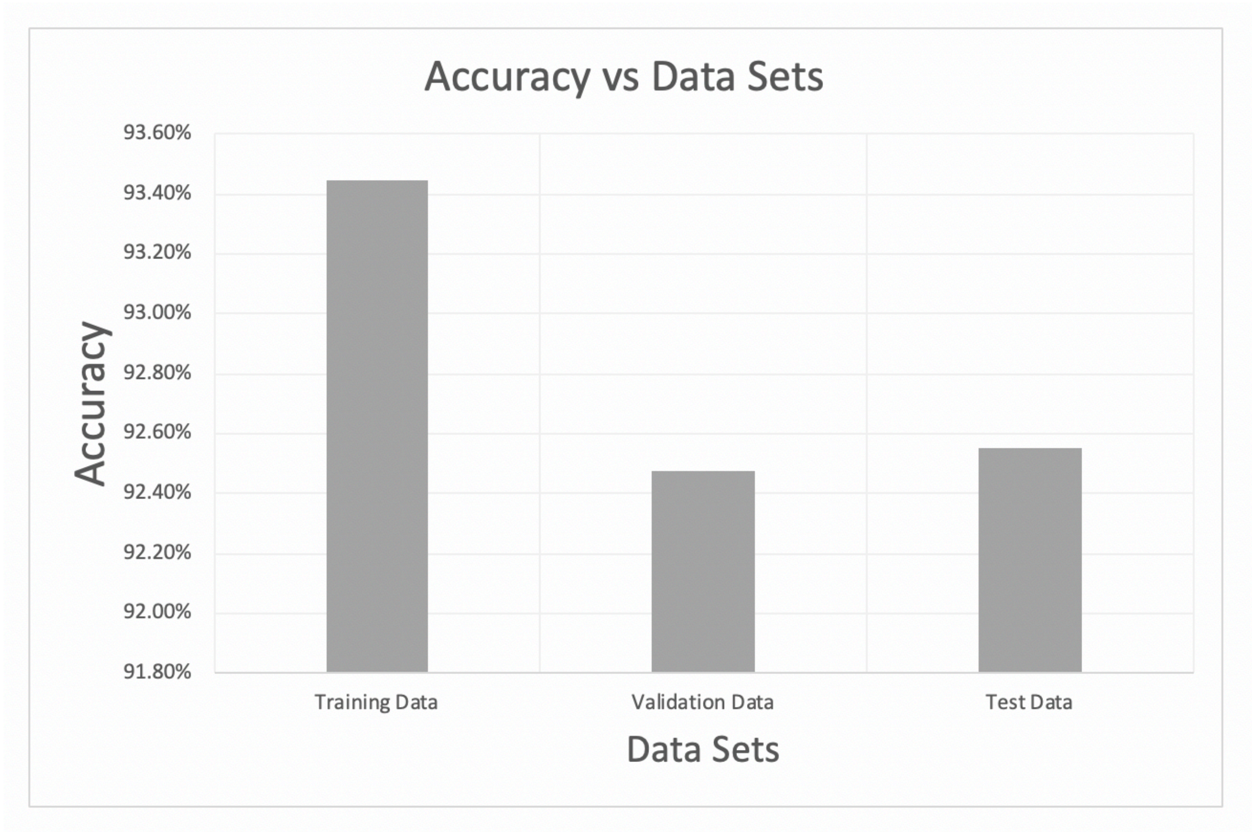
- The accuracy for the test data set is lower than the accuracy for training data set.
- There is a difference obtained between the training error and test error is due to the fact that Linear Regression progresses as a non-convex function.
- Training Accuracy is very slightly more than the Test Accuracy as the model was trained on the training data set and it would in turn favor the training accuracy rather the test accuracy. Hence, the hyperplane divides the training data set rather efficiently than test data set.

Problem 2: Implementation of Multi-Class Logistic Regression

The following table shows various accuracies obtained in Multi-Class Logistic Regression against varied data set:

Data	Accuracy
Training Data	93.448 %
Validation Data	92.479 %
Test Data	92.55 %

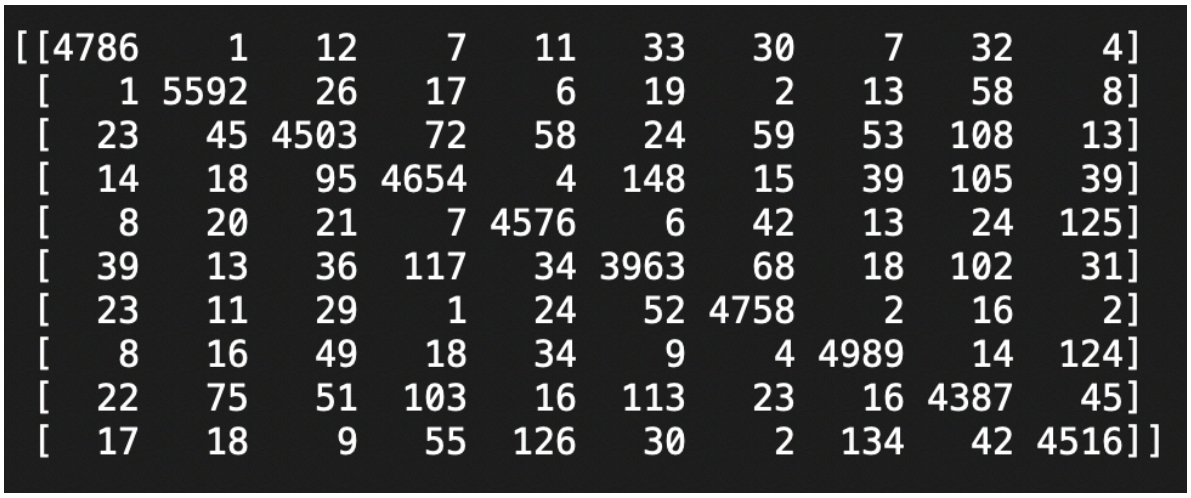
The total time taken by our code for Multi-Class Logistic Regression is 24.5829 seconds.



Category wise error:

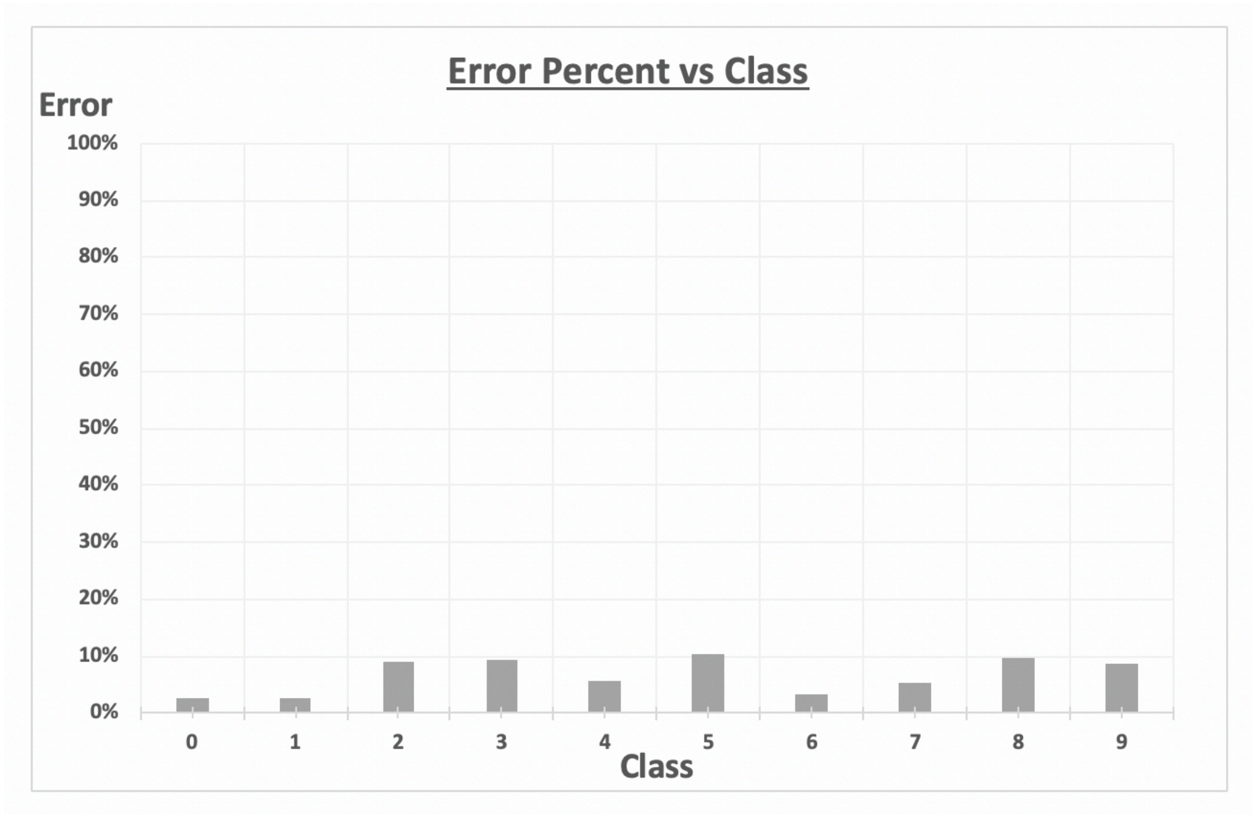
1. Training Data:

Confusion Matrix:



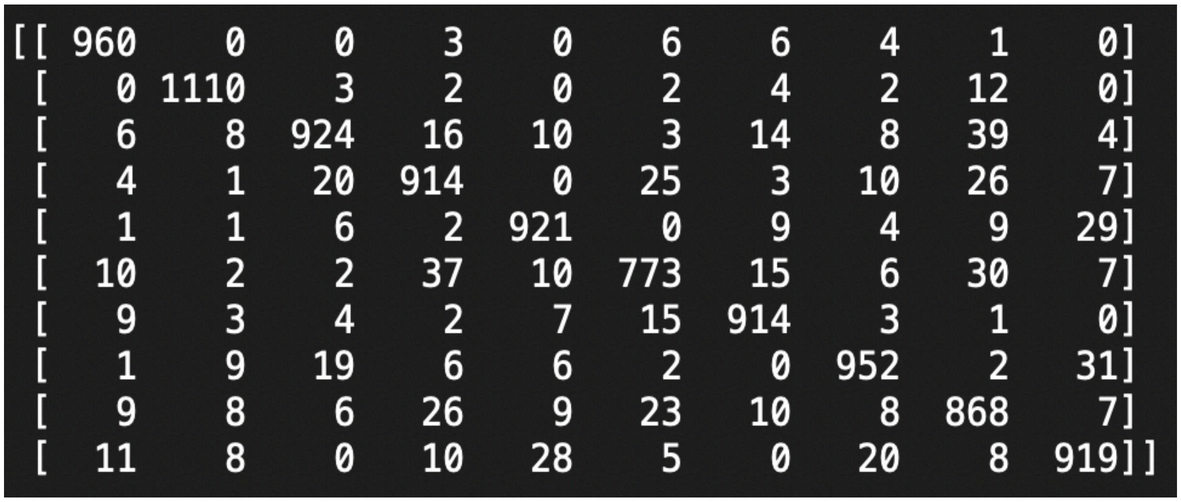
```
[[4786    1   12    7   11   33   30    7   32    4]
 [   1 5592   26   17    6   19    2   13   58    8]
 [  23   45 4503   72   58   24   59   53  108   13]
 [  14   18   95 4654    4  148   15   39  105   39]
 [   8   20   21    7 4576    6   42   13   24  125]
 [  39   13   36  117   34 3963   68   18  102   31]
 [  23   11   29    1   24   52 4758    2   16    2]
 [   8   16   49   18   34    9    4 4989   14  124]
 [  22   75   51  103   16  113   23   16 4387   45]
 [  17   18    9   55  126   30    2  134   42 4516]]
```

Class	Error %
0	2.78%
1	2.61%
2	9.18%
3	9.30%
4	5.49%
5	10.36%
6	3.25%
7	5.24%
8	9.57%
9	8.75%



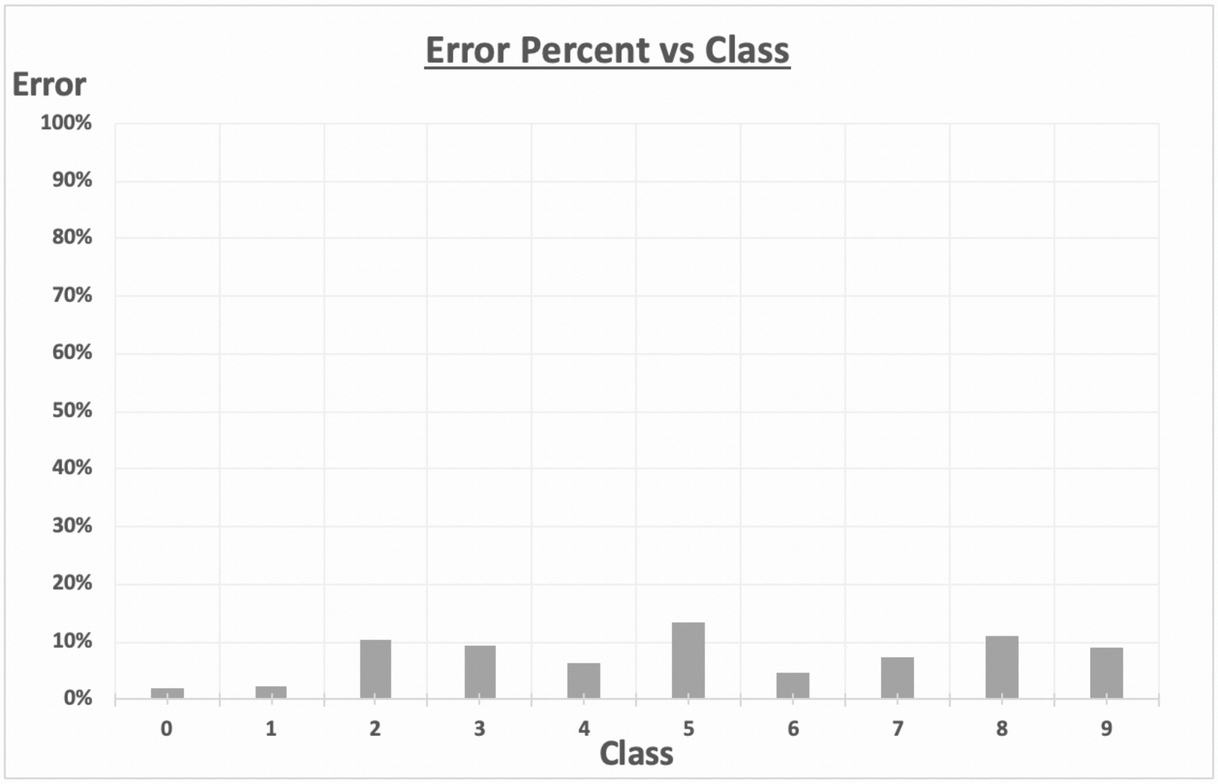
2. Test Data:

   Confusion Matrix:

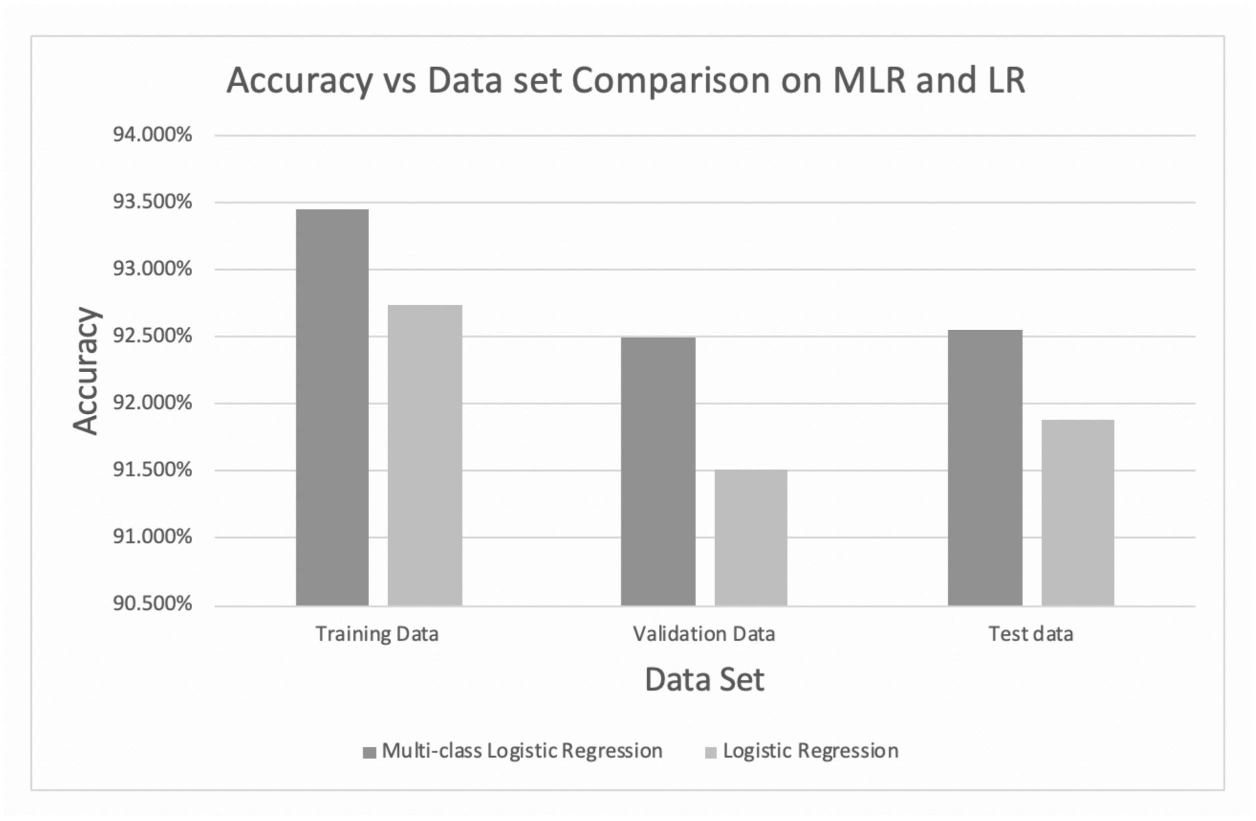


```
[[ 960    0    0    3    0    6    6    4    1    0]
 [    0 1110    3    2    0    2    4    2   12    0]
 [    6    8  924   16   10    3   14    8   39    4]
 [    4    1   20  914    0   25    3   10   26    7]
 [    1    1    6    2  921    0    9    4    9   29]
 [   10    2    2   37   10  773   15    6   30    7]
 [    9    3    4    2    7   15  914    3    1    0]
 [    1    9   19    6    6    2    0  952    2   31]
 [    9    8    6   26    9   23   10    8  868    7]
 [   11    8    0   10   28    5    0   20    8  919]]
```

Class	Error %
0	2.04%
1	2.20%
2	10.47%
3	9.50%
4	6.21%
5	13.34%
6	4.59%
7	7.39%
8	10.88%
9	8.92%



Conclusion:



Comparison:

Linear Regression	Multi-class Linear Regression
Time taken in Linear Regression is more than Multi-class Linear Regression as optimization is performed 10 times (one for each class – digits 0 to 9)	Multi-class Linear Regression takes lesser time than Linear Regression as the classifier can run at once for all the classes at once.
This performs worse than Multi-class Linear Regression as Linear Regression is primarily used for Binary Classification.	This performs better than Linear regression if we consider the one vs all Logistic Regression strategy.

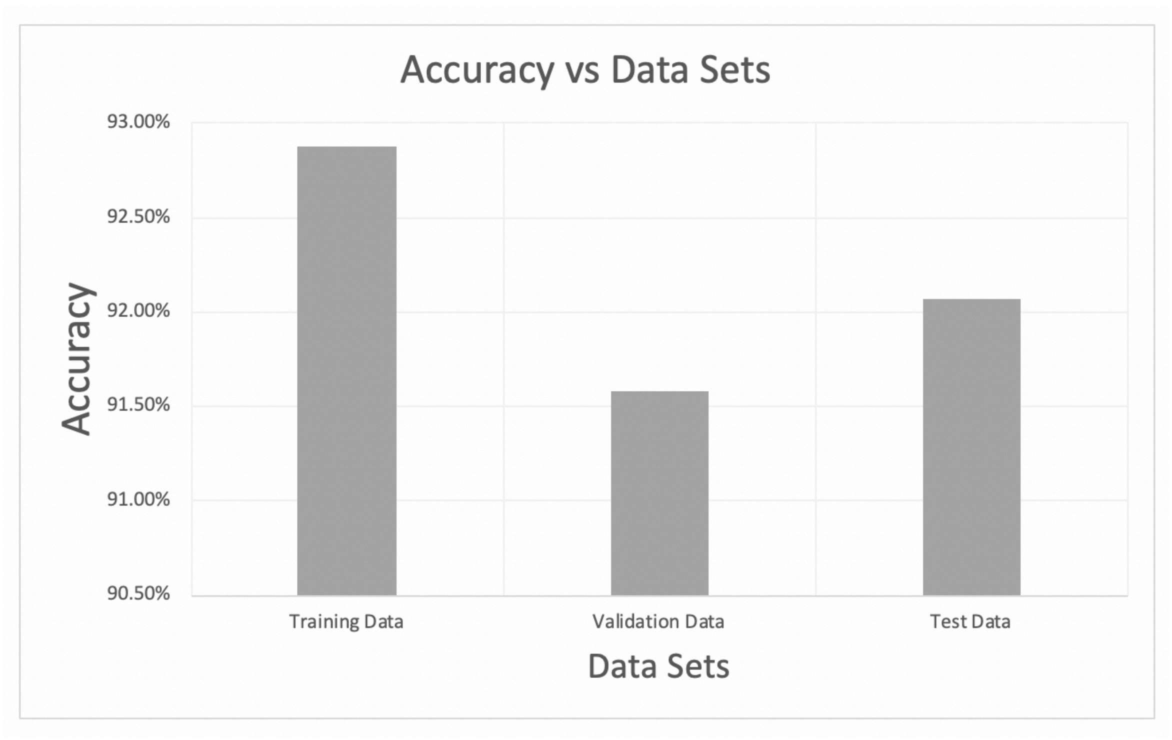
Problem 3: Implementation of Support Vector Machines

1. SVM using Linear Kernel:

The following table shows various accuracies obtained in SVM against varied data set:

Data	Accuracy
Training Data	92.874 %
Validation Data	91.58 %
Test Data	92.07 %

The total time taken by our code for SVM is 165.5910 seconds.



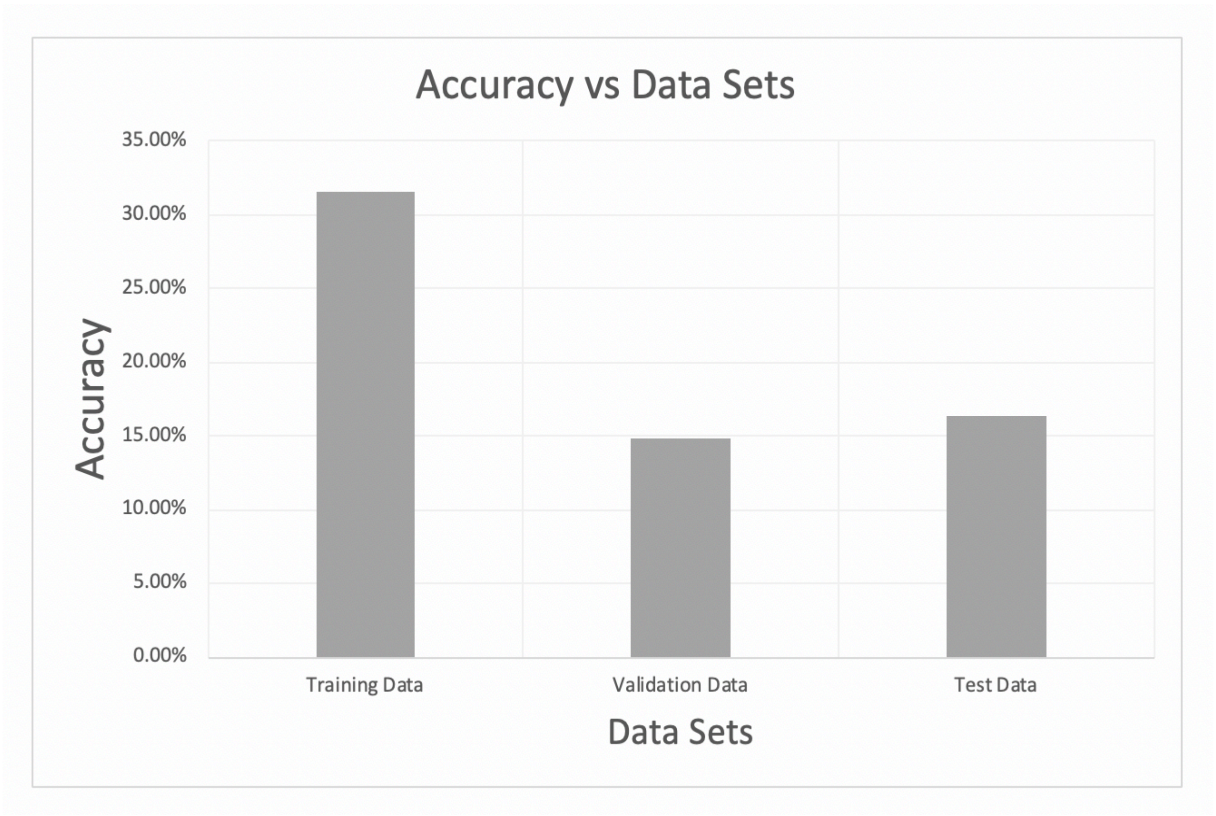
2. SVM with radial basis function for gamma = 1

The following table shows various accuracies obtained in SVM against varied data set:

Data	Accuracy
Training Data	31.5419 %
Validation Data	14.8799 %
Test Data	16.400 %

The total time taken by our code for SVM is 719.6619 seconds.

Note: We used the whole data set for the calculation of accuracy and did not sample the data. Hence, the accuracy we obtained is less.

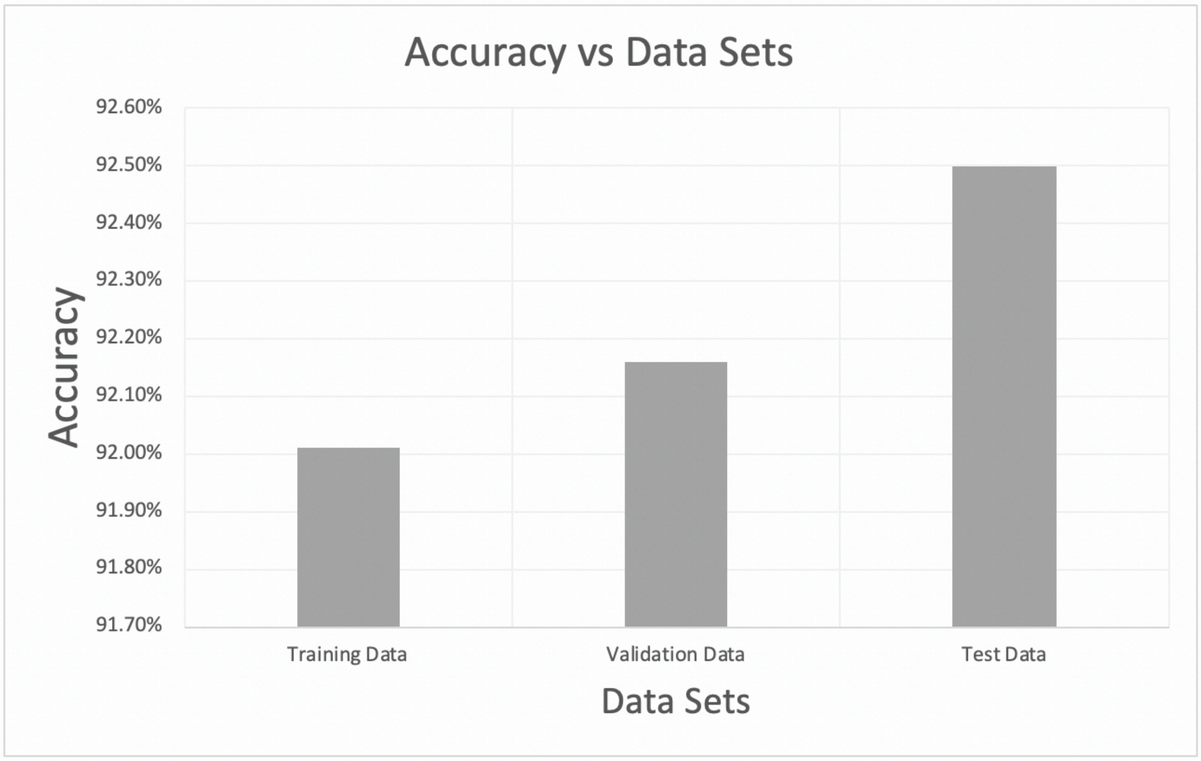


3. SVM with radial basis function for default gamma

The following table shows various accuracies obtained in SVM against varied data set:

Data	Accuracy
Training Data	92.01 %
Validation Data	92.16 %
Test Data	92.5 %

The total time taken by our code for SVM is 325.9381 seconds.

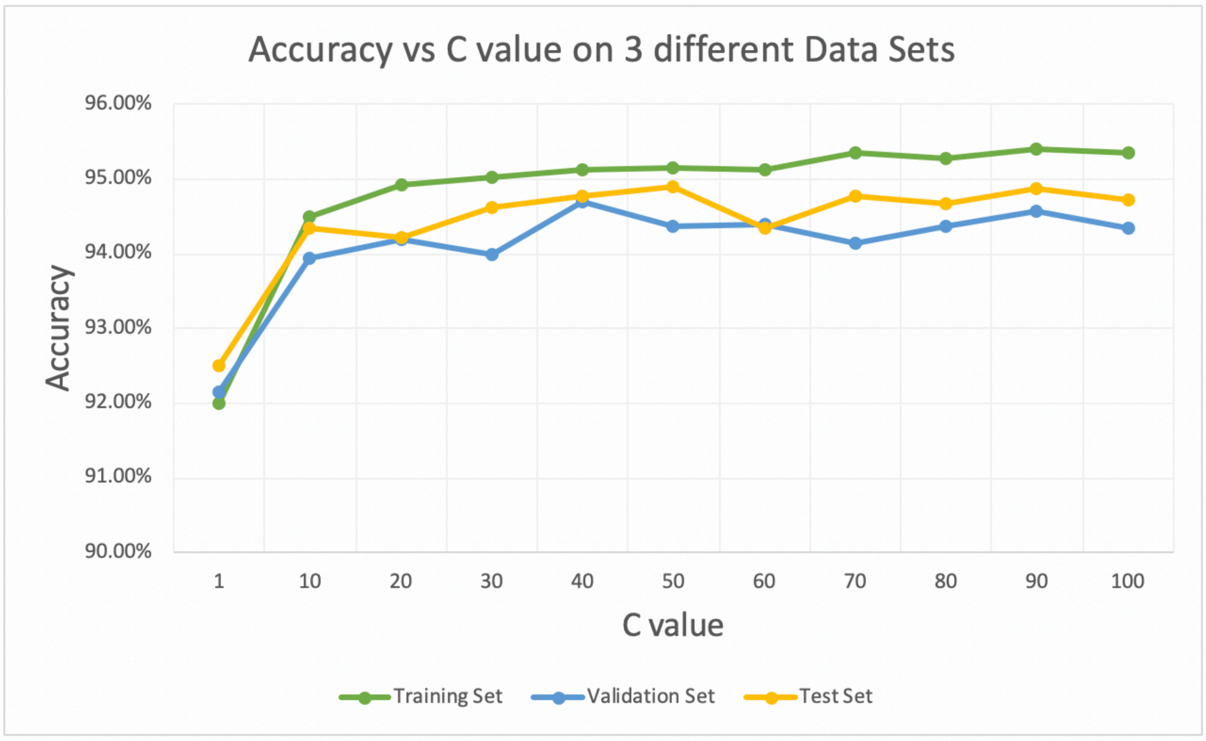


4. SVM with radial basis function for default gamma and varying value of C

The following table shows various accuracies obtained in SVM against varied values of C with gamma set to default:

C	Training Set	Validation Set	Test Set	Time (seconds)
0	92.01%	92.16%	92.50%	325.938
10	94.50%	93.93%	94.35%	205.322
20	94.92%	94.20%	94.21%	178.46
30	95.02%	93.99%	94.62%	179.701
40	95.13%	94.69%	94.78%	172.672
50	95.154%	94.36%	94.89%	171.063
60	95.13%	94.39%	94.35%	174.321
70	95.34%	94.13%	94.77%	417.445
80	95.27%	94.37%	94.67%	200.509
90	95.40%	94.58%	94.86%	185.77
100	95.34%	94.35%	94.73%	191.635

The total time taken by our code for SVM is 2402.836 seconds



From the above, we observed that the radial basis function kernel performs the best with default gamma. We can observe that with the increased value of C the accuracies increase, which substantiates the fact that with large value s of C, a smaller margin will be accepted. Hence, ***C behaves*** as a ***regularization parameter*** in the SVM. Also, as the train accuracies for all C values were almost similar but for C=50, the time taken was comparatively less. Hence, we opted all these parameters and trained with the whole training dataset and below are the accuracies reported:

Data	Accuracy
Training Data	99.002%
Validation Data	97.31%
Test Data	97.19%

The total time taken by our code with optimal parameters is 2372.103 seconds.

Conclusion:

SVM with radial basis function kernel performs well as compared to the linear kernel as it translates features to higher dimension and helps SVM to draw a better support vector (hyperplane) and thereby, performs better on the database with complex features and default gamma reduces chances of overfitting.