Machine Learning (CSE574)
Programming Assignment 3
Logistic Regression and Support Vector Machine
Project Group - 36

Shubham Gulati	***Mohammad Umair***	***Yichen Wang***
(sgulati3)	(m39)	(yichenwa)

1.1 Implementation of Logistic Regression

Training Accuracy (%)	Validation Accuracy (%)	Testing Accuracy (%)
92.718	91.5	92.04

Time taken: 1330 seconds

Confusion matrix for test data:

```
[[4770    1   15    7   12   40   34    8   34    2]
 [   1 5593   29   17    6   19    2   13   54    8]
 [  16   41 4509   79   56   25   59   55  102   16]
 [  16   19   97 4658    4  146   17   38   92   44]
 [   9   23   24    4 4538    7   46   16   27  148]
 [  43   16   42  122   32 3956   66   16   90   38]
 [  24   11   30    1   27   58 4743    2   20    2]
 [   9   18   50   19   38   11    4 4969   13  134]
 [  24   90   50  111   20  117   27   13 4340   59]
 [  20   22   11   58  121   31    2  128   40 4516]]
```

Errors for each of the classes:

Class	Error
0	2.17%
1	2.04%
2	8.89%
3	10.23%
4	6.09%
5	11.81%
6	3.68%
7	5.77%
8	12.45%
9	10.81%

Confusion matrix for test data:

```
[[ 959    0    2    3    0    6    6    3    1    0]
 [   0 1109    3    2    0    1    4    2   14    0]
 [   5    9  929   16    9    5   14   10   31    4]
 [   3    1   19  919    0   23    3   12   22    8]
 [   1    2    5    1  917    0   10    5    9   32]
 [  10    2    3   38   11  771   16    7   28    6]
 [  11    3    3    2    8   14  911    3    3    0]
 [   1    8   22    7    9    2    0  946    2   31]
 [   7    9    6   23    8   25   11    9  864   12]
 [  11    8    1    8   23    6    0   16    8  928]]
```

Errors for each of the classes:

Class	**Error**
0	1.94%
1	1.76%
2	11.24%
3	8.81%
4	6.42%
5	14.57%
6	5.22%
7	7.59%
8	13.04%
9	10.80%

Conclusion: Overall our model works quite well if not very good since our test and validation accuracy are around 92%. This also means that our model is not overfitting on the training data which is a good thing. The training accuracy is slightly more than test accuracy because the model is trained on training data and because of this the final hyperplane is more suited to divide training data into the classes correctly. The test data probably contains some extra data points which end up on the wrong side of the hyperplane.

1.2 Multi-class Logistic Regression

	Training Accuracy (%)	Validation Accuracy (%)	Testing Accuracy (%)
Multi-class LR	93.154	92.17	92.37
LR	92.718	91.5	92.04

Time taken: 80 seconds

Confusion matrix for training data:

```
[[4770    1   15    7   12   40   34    8   34    2]
 [   1 5593   29   17    6   19    2   13   54    8]
 [  16   41 4509   79   56   25   59   55  102   16]
 [  16   19   97 4658    4  146   17   38   92   44]
 [   9   23   24    4 4538    7   46   16   27  148]
 [  43   16   42  122   32 3956   66   16   90   38]
 [  24   11   30    1   27   58 4743    2   20    2]
 [   9   18   50   19   38   11    4 4969   13  134]
 [  24   90   50  111   20  117   27   13 4340   59]
 [  20   22   11   58  121   31    2  128   40 4516]]
```

Errors for each of the classes:

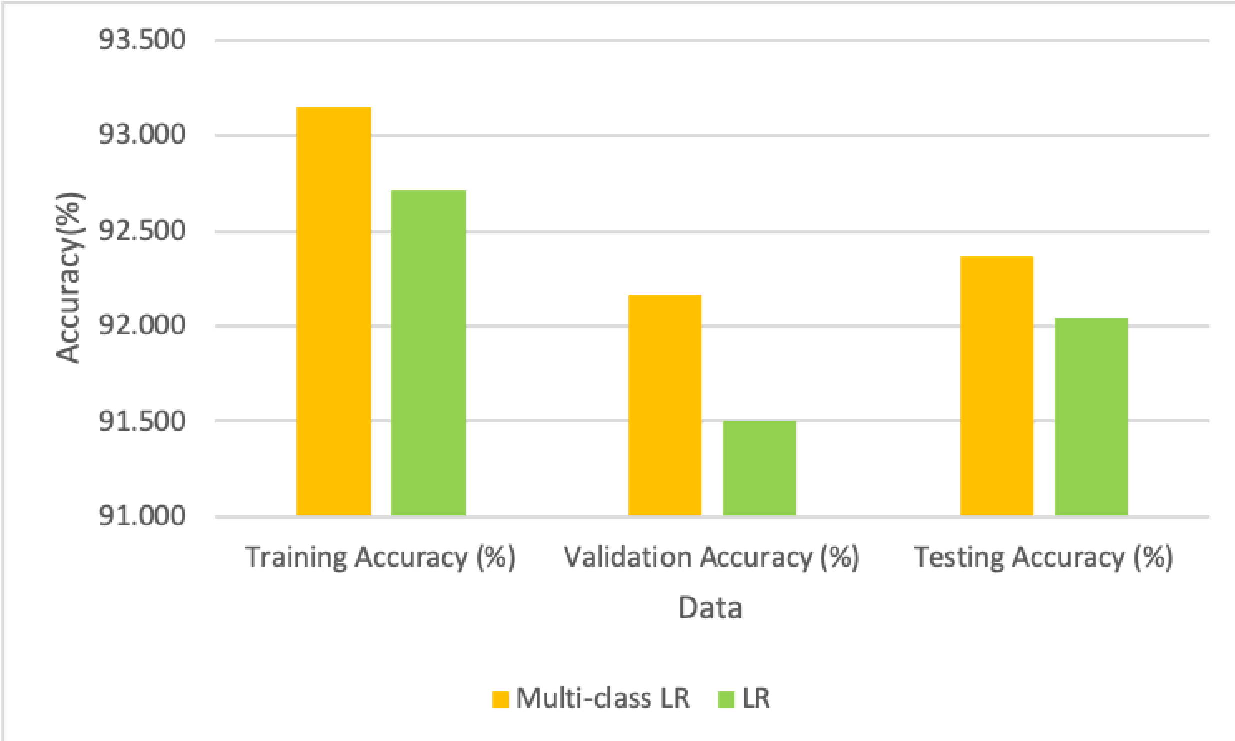
Class	Error
0	3.11%
1	2.59%
2	9.06%
3	9.22%
4	6.28%
5	10.52%
6	3.56%
7	5.62%
8	10.53%
9	8.75%

Confusion matrix for test data:

```
[[ 959    0    2    3    0    6    6    3    1    0]
 [   0 1109    3    2    0    1    4    2   14    0]
 [   5    9  929   16    9    5   14   10   31    4]
 [   3    1   19  919    0   23    3   12   22    8]
 [   1    2    5    1  917    0   10    5    9   32]
 [  10    2    3   38   11  771   16    7   28    6]
 [  11    3    3    2    8   14  911    3    3    0]
 [   1    8   22    7    9    2    0  946    2   31]
 [   7    9    6   23    8   25   11    9  864   12]
 [  11    8    1    8   23    6    0   16    8  928]]
```

Errors for each of the classes:

Class	**Error**
0	2.14%
1	2.29%
2	9.98%
3	9.01%
4	6.62%
5	13.57%
6	4.91%
7	7.98%
8	11.29%
9	8.03%



Conclusion: Our training set accuracy of multi-class logistic regression is better than testing set accuracy. The reason is actually same as it was in binary logistic regression. The model is being trained on the training data and that is why it gives better accuracy with training data and a little lesser on test data. Also, since the training accuracy is good, the model is quite good.

The accuracy in case of multi-class logistic regression is better than in case of binary logistic regression. This is because our dataset has multiple labels and not just two labels and multi-class logistic regression works better than binary logistic regression in case of more than two classes. Binary logistic regression divides the data in an unbalanced way and that is why it performs worse. The time taken is also more in case of binary logistic regression. Because of all these reasons, multi-class logistic regression is clearly a better choice for this problem.

1.3 Support Vector Machines

SVM with linear kernel and default values

kernel		
linear		
Training Accuracy (%)	Validation Accuracy (%)	Testing Accuracy (%)
92.746	91.55	91.679
Time (In secs)		
154.379517793655		

Time taken for SVM with linear kernel: 154 seconds, about 3 minutes.

SVM with radial basis function, gamma = 1

gamma=1		
kernel		
rbf		
Training Accuracy (%)	Validation Accuracy (%)	Testing Accuracy (%)
27.534	10.02	11.39
Time (In secs)		
676.615482330322		

Time taken for SVM with radial basis function, gamma = 1: 676 seconds, about 11 minutes.

SVM with radial basis function, gamma = default and training accuracy calculated on sampled data itself

gamma=1		
kernel		
rbf		
Training Accuracy (%)	Validation Accuracy (%)	Testing Accuracy (%)
100.0	14.27	15.8
Time (In secs)		
379.277814626693		

Observation: As we can see in the above table, that gamma=1 overtrains the model which ends up giving 100% training accuracy but very low test and validation accuracy. However, with same parameters, if we calculate accuracy on training data, it comes out to be really low as expected.

SVM with radial basis function, gamma = default

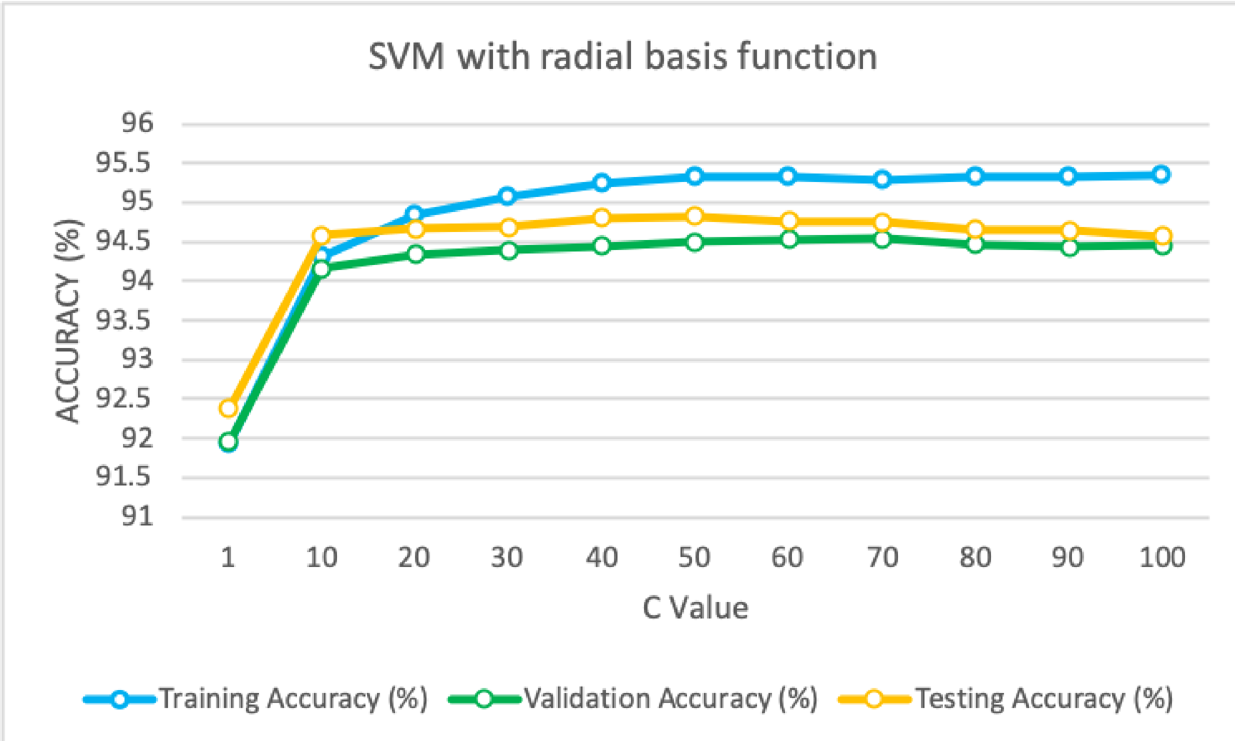
gamma=default		
kernel		
rbf		
Training Accuracy (%)	Validation Accuracy (%)	Testing Accuracy (%)
91.982	92.07	92.36
Time (In secs)		
309.483509778976		

Time taken for SVM with radial basis function, gamma = default: 309 seconds, about 5 minutes.

SVM with radial basis function, gamma set to default and different values of C

kernel			
rbf			
C values	Training Accuracy (%)	Validation Accuracy (%)	Testing Accuracy (%)
1	91.932	91.95	92.38
10	94.318	94.16	94.58
20	94.836	94.34	94.67
30	95.074	94.39	94.69
40	95.25	94.44	94.8
50	95.33	94.5	94.82
60	95.316	94.53	94.76
70	95.288	94.54	94.75
80	95.334	94.47	94.66
90	95.334	94.43	94.64
100	95.336	94.45	94.57
Time (In secs)			
2033.5802557468			

Time taken for SVM with radial basis function, different values of C: 2033 seconds, about 34 minutes.



As can be seen in the above tables and graphs, the best kernel is radial basis function, best value of gamma is default and best value of C is 50. There we use these values to fit our final model on the whole training data. The final results are as follows:

SVM trained on whole data with the optimal values

C = 50, gamma = default		
kernel		
rbf		
Training Accuracy (%)	Validation Accuracy (%)	Testing Accuracy (%)
99.002	97.31	97.19
Time (In secs)		
634.927506446838		

Conclusion: Radial basis gives better accuracy than linear kernel because radial basis takes the data into a higher dimension and in this higher dimension, we are able to fit our hyperplane much better. Something like this is not possible with linear kernel because the data is not linearly separable in the first place.

As we observed previously that gamma = 1 overfits the data the default value of gamma gives much better accuracy.

The value of C decides how much to penalize for mis-classifications. It is pretty evident from the graph that C=1 gives the worst accuracy on both training and test data.