SML: Assignment 4

Shreya Sharma
2015096

1) OUTPUT OF THE CODE :

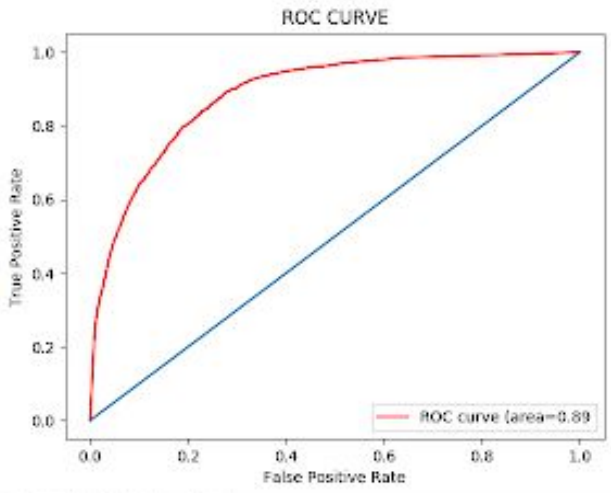
   a) Running Code submitted. OUTPUT :

```
length of class 0 : 22654
length of class 1 : 7508
length of training data = 15081
length of testing data = 15081
training neural network ...

training accuracy: 0.843578
testing accuracy: 0.839268

CONFUSION MATRIX
          | predicted:1 | predicted:0 |
|actual:1 |tn =  0.699  |fp =   0.052 |
|actual:0 |fn =  0.108  |tp =   0.141 |

EER VALUE : 0.1935
```

   b) CONFUSION MATRIX:

	PREDICTED : 1	PREDICTED : 0
ACTUAL : 1	TN = 0.699	FP = 0.052
ACTUAL : 0	FN = 0.108	TP = 0.141

c) ROC CURVE:



d) EER VALUE on the testing data : 0.1935.

EER value is the value where FRR (False rejection rate) and FPR (False positive rate) intersect. FRR = 1-TPR. Lower the EER, better is the classifier.

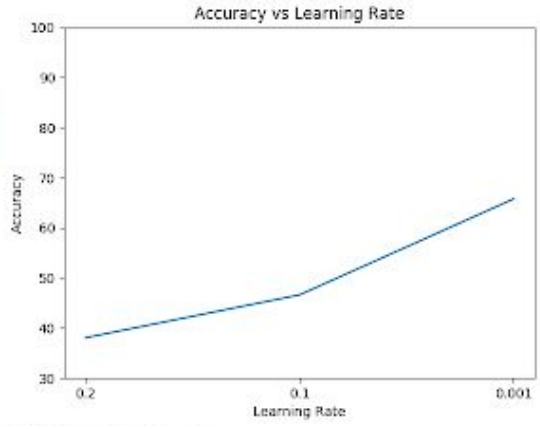
---

2) Neural Network with 3 hidden layers: H1: 256, H2: 128, H3: 64, output: 47, Activation function: linear, epochs: 100.

a) Varying learning rate 0.2, 0.1, 0.001:

OUTPUT : [38.090425531914896, 46.638297872340424, 65.76595744680851]

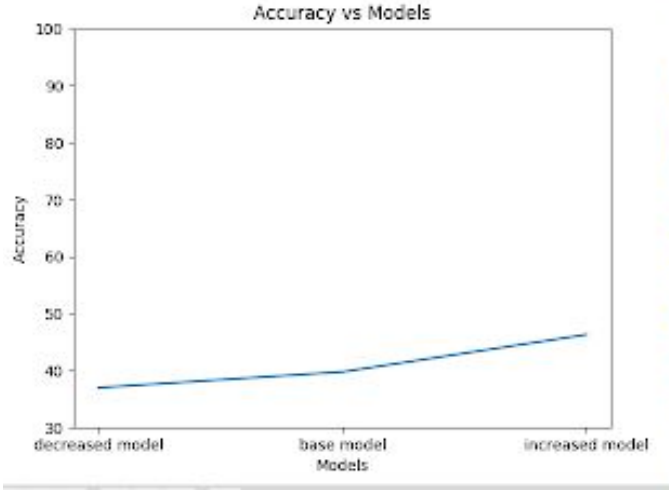
PLOT :



ANALYSIS : From the graph we can tell that as the learning rate is decreased, the accuracy of neural network increases. Learning rate determines how

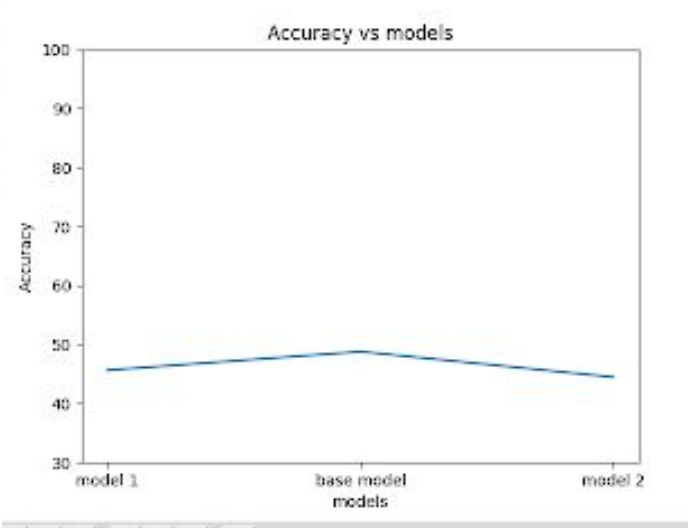
fast/slow our model learns and updates the weights of the neural network. If the learning rate is large, it will learn more quickly but there is a chance that it may not converge (as it may overshoot the optimal point). On the other hand a smaller learning rate will be more likely to converge to the optimal point but it may take longer. From this graph we can say that as the learning rate is decreased, it converges to the optimal point hence the accuracy is increased so much.

b) Changing Number of hidden nodes:
Model 1 : 64,32,16 - **37.021%**
Model 2 (Base Model) : 256,128,64 - **39.80%**
Model 3 : 1024,512,256 - **46.33%**
OUTPUT : [37.02127659574468, 39.797872340425535, 46.33510638297872]



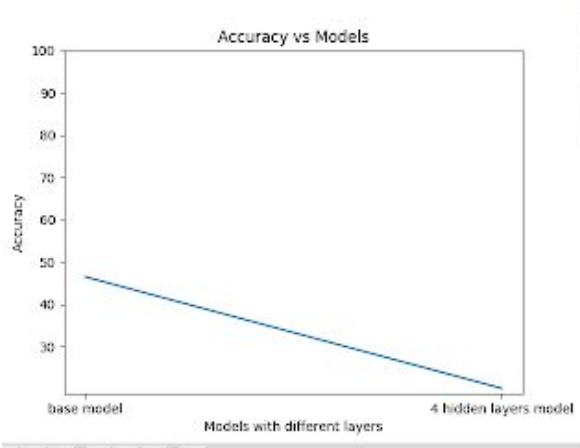
ANALYSIS - We can see from the graph, if the structure is kept similar (3 layers with decreasing number of nodes at each layer), increasing the number of nodes at corresponding layers, increases the accuracy.

Also,
For
Model 1 : (64,128,256) - **45.66%**
Model 2 (Base Model) : 256,128,64 - **48.81%**
Model 3 : (128,256,64) - **44.49%**
OUTPUT : [45.66489361702128, 48.81382978723404, 44.48936170212766]



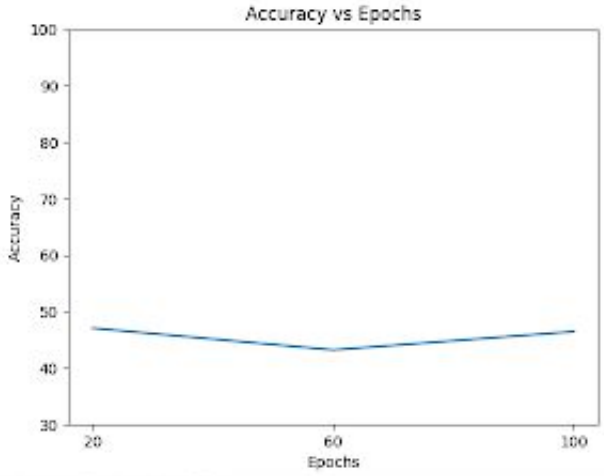
ANALYSIS - We can see That compared to the base model, when we reverse the order of number of nodes at each layer, ie when they are in increasing order, the accuracy decreases, this means it's better if the later layers have less number of nodes compared to earlier ones. Also, if the middle hidden layer has highest number of nodes, more than the mean of 1st and last, the accuracy decreases even more. Hence the middle hidden layer should not have nodes more than the average of the outer layers.

c) **3 Layers :** 46.48%
4 Layers : 20.10%



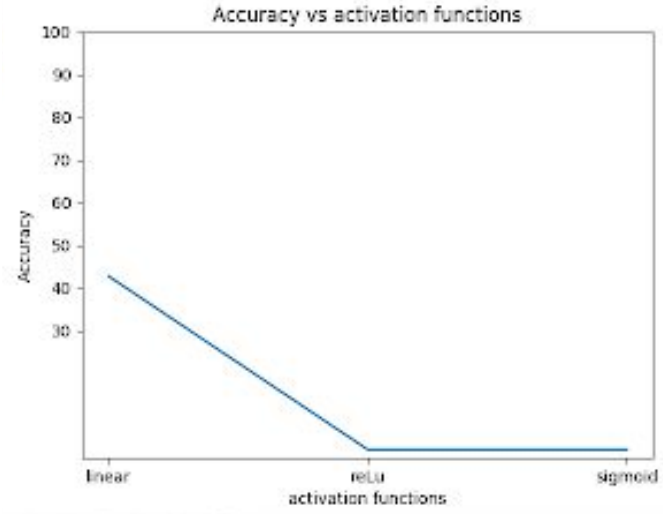
ANALYSIS - As we can see from the graph, adding one more layer of size 128 to the neural network, decreases the accuracy. This is due to the vanishing gradient problem. Each of the neural network weights are updated in proportion to the gradient of the error function with respect to the current weight in each iteration of training. There is a chance that gradient will be vanishingly small, and so prevent the weight from changing its value. In the worst case, this may completely stop the neural network from further training.

d) Epoch = 20 : 47.12%
Epoch = 60 : 43.24%
Epoch = 100 : 46.50%



ANALYSIS - From the graph one can see that epoch = 20 gives the highest accuracy and epoch = 100 gives 2nd highest and epoch = 60 has the lowest accuracy. At every iteration, the weights are updated and the neural network figures out weather to increase them or decrease them. If the number of iterations is too small, it may not be enough for training the model to get the optimal weight values. Too large iteration can lead to not much change in the weights in the later stages and we are just repeatedly doing redundant work.

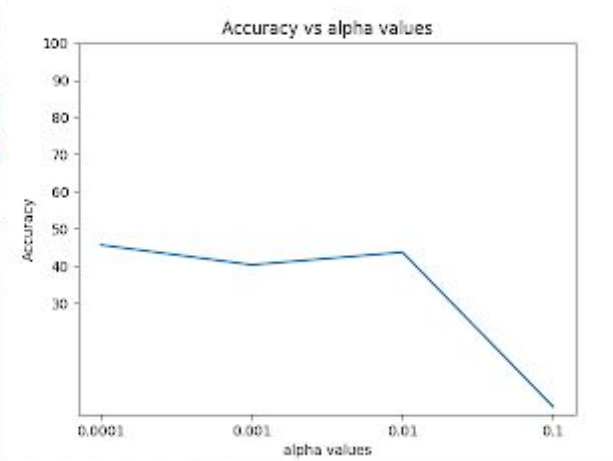
e) Activation Functions : Linear, ReLu, Sigmoid
Accuracies = [42.776595744680854, 2.127659574468085, 2.127659574468085]



ANALYSIS - From the graph we can see that the accuracies decreases sharply when the activation functions is changed from linear to Relu/Sigmoid. Sigmoid and ReLu give the same accuracy for this data. We can say that for this classification, linear activation function constructs an optimal neural network.

f) BONUS : varying alpha

Alpha is L2 penalty (regularization term) parameter**.** Increasing alpha may fix high variance (a sign of overfitting). Similarly, decreasing alpha may fix high bias (a sign of underfitting) potentially resulting in a more complicated decision boundary.



VALUES = [45.648936170212764, 40.31382978723404,43.691489361702125, 2.127659574468085]

From the graph we can see that increasing alpha drastically (from 0.0001 to 0.1) decreases the accuracy as the model is simpler compared to the dataset and is not able to fit it well. But we also see that there is no linear trend that's being followed as when alpha=0.01, the accuracy increased as compared to that at 0.001 but still less that 0.0001. Thus we can say going from 0.01 to 0.001 the model becomes inaccurate for the data.

---

3)

a) Code for boosting is submitted. I've used Sklearn's AdaBoostClassifier and hep_ml.nnet library for classification.

Iteration 1 :

Without applying boosting, the accuracy obtained = **0.281500**

Accuracy obtained : **0.2495**

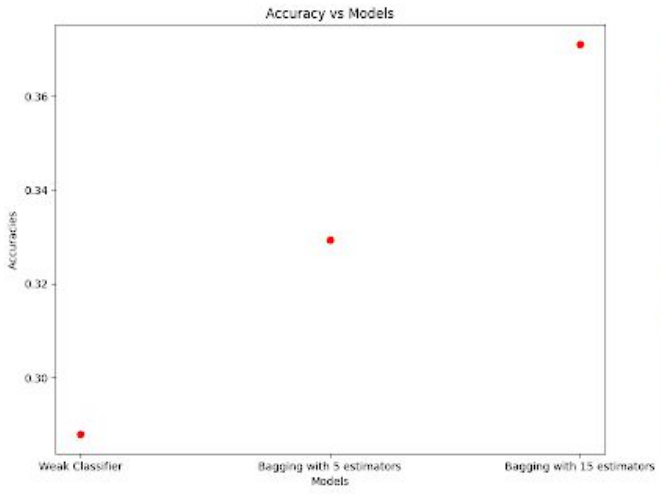
The accuracy decreased. It may be because the Neural Network is not a weak classifier and AdaBooster uses various weak classifiers to improve the performance. Hence we can say that AdaBooster increases the accuracy but not so much.

Iteration 2 :

Without applying boosting, the accuracy obtained = **0.2720**
*Interesting Observation: By keeping the learning_rate=0.1, the accuracy obtained = **0.28400***
We can see that the accuracy increased. From this observation we can say that the above model of AdaBooster was not fit for the dataset. But by changing the learning_rate we get an increment in accuracy.

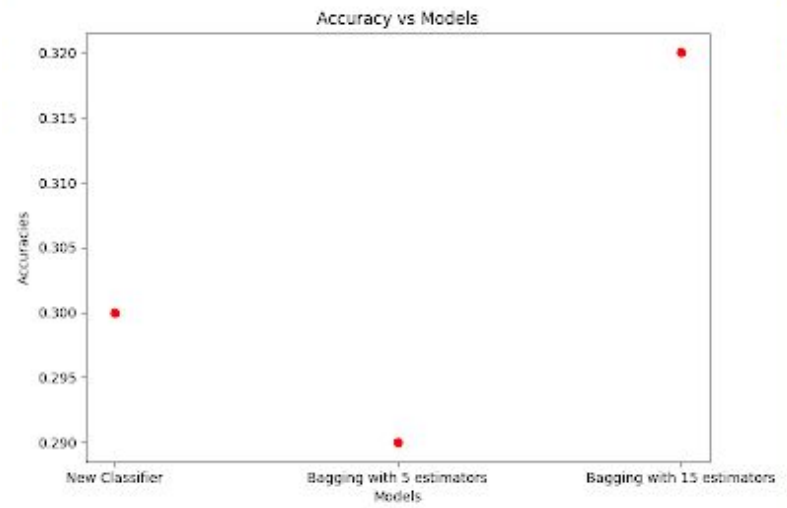
b) Without applying bagging, the accuracy obtained = **0.287900**
Bagging code is submitted. Used Sklearn's BaggingClassifier and MLPClassifier libraries.
ACCURACY OBTAINED **testing accuracy: 0.356000**
We can see that by applying bagging with default number of estimators = 10, the accuracy of the weak classifier increased. Thus we can conclude that the accuracy increases on applying Bagging.

c) **Bagging with 5 estimators accuracy : testing accuracy: 0.329300**
Bagging with 15 estimators accuracy : testing accuracy: 0.371100
We can see from the accuracies that increasing the number of estimators improves the performance of the classifier. Also, applying bagging to a weak classifier increases the accuracy.



<u>BOOSTING</u>
Iteration 1 : learning rate = 1
Boosting with 5 estimators accuracy : testing accuracy: testing accuracy: 0.2555
Boosting with 15 estimators accuracy : testing accuracy: 0.266
We can see from the accuracies that increasing the number of estimators improves the performance of the classifier slightly. But in general, applying

an unfit boosting to a weak classifier decreases the accuracy.



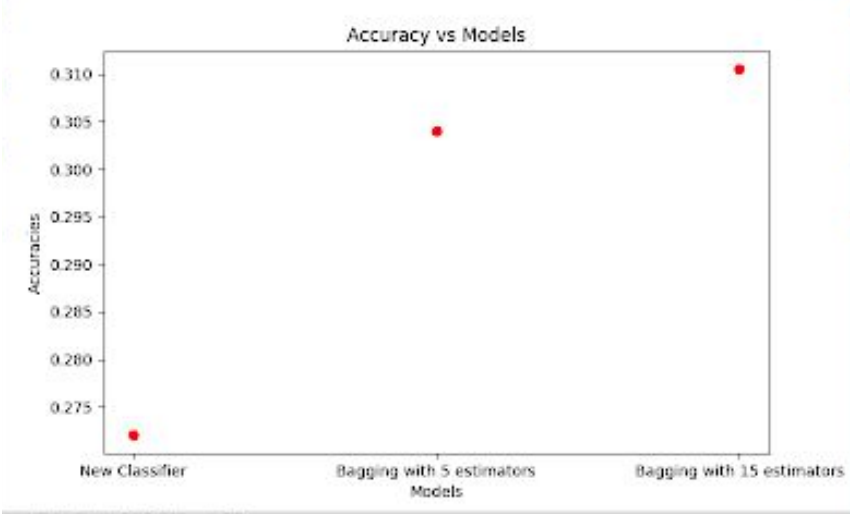
Iteration 2

Interesting Observation: By keeping the learning_rate=0.1 the accuracy increased

Boosting with 5 estimators accuracy : 0.304

Boosting with 15 estimators accuracy : 0.3105

We can see that increasing the number of estimators increases the performance and with this model of AdaBooster the accuracy increases.



With learning_rate = 0.1 we get the expected result. Hence the AdaBooster model is fit for the classification.

d) Analysis of a,b and c done in respective parts above.

e) **Without bagging : testing accuracy: 0.368800**

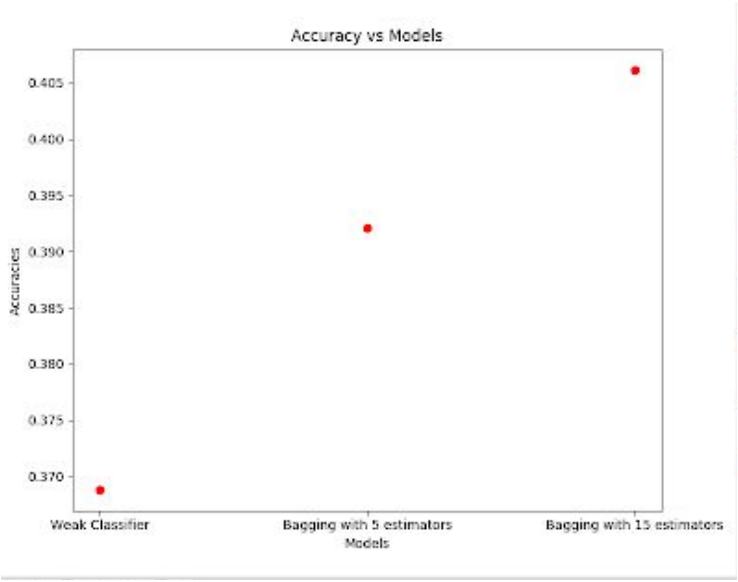
BAGGING :

By changing the classifier, the accuracy increased by about 7%.

i) With bagging : **testing accuracy: 0.404900** - Bagging increases the accuracy of the classifier. And compared to the classifier in part a,b and c, this classifier improves the accuracy even further.

ii) bagging with 5 estimators : testing accuracy: 0.392100

iii) **Bagging with 15 estimators : testing accuracy: 0.406100**



From the graph we can see that applying bagging to the classifier increases the accuracy. Moreover, increasing the number of estimators increases the accuracy as well.

Comparing both the Neural Networks : We can see that there is an increase in the accuracy on both the models if bagging is applied. Also, by increasing the estimators, the accuracy increases.

	Weak classifier	Bagging with 5 estimators	Bagging with 10 (default) estimators	Bagging with 15 estimators
2 hidden Layers	0.3688	0.3921	0.4049	0.4061
1 hidden Layers	0.2879	0.3293	0.356	0.3711

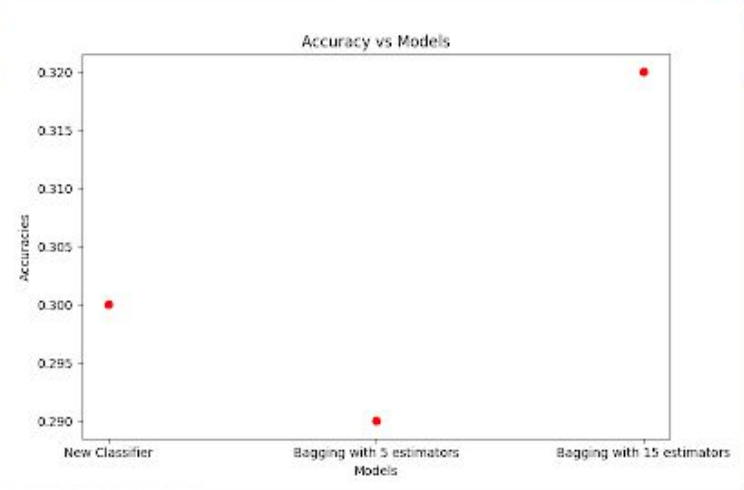
Without boosting : testing accuracy: 0.300000

BOOSTING :

*From above observation of part a and c, learning_rate was ket 0.1

By changing the classifier, the accuracy increased by about 7%.

1. With boosting : **testing accuracy: testing accuracy:0.330000** - boosting increases the accuracy of the classifier. And compared to the classifier in part a,b and c, this classifier improves the accuracy even further. Default value of estimators = 50
2. **boosting with 5 estimators : testing accuracy: 0.2900:** is less but still very close to the weak classifier's actual accuracy.
3. **boosting with 15 estimators : testing accuracy: 0.3200:** increased slightly than the weak classifier and 5 estimator classifier.



From the graph we can see that applying boosting to the classifier with less number of estimators deccreases the accuracy. But as the number of estimators is increased, there is an increase in the accuracy as well.

Comparing both the Neural Networks : *learning rate =0.1

	Weak classifier	Bagging with 5 estimators	Bagging with 50 (default) estimators	Bagging with 15 estimators
2 hidden Layers	0.300	0.290	0.33	0.32
1 hidden Layers	0.2720	0.304	0.2840	0.3105