CSE343- MACHINE LEARNING

REPORT

Assignment 2: Decision Tree, Random Forest, MLP

Submitted By
Dolly Sidar, 2019304

QUESTION 1: Decision Tree and Random Forest

Step 1 - Imported the needed libraries and loaded the dataset

Step 2 - Preprocessing

a. dropping 'No' column
b. converting the categorical value to an integer
c. replacing null values to mean of the column
d. shuffling of data to ensure iid
e. splitting the data into train, val, test: 70:15:15

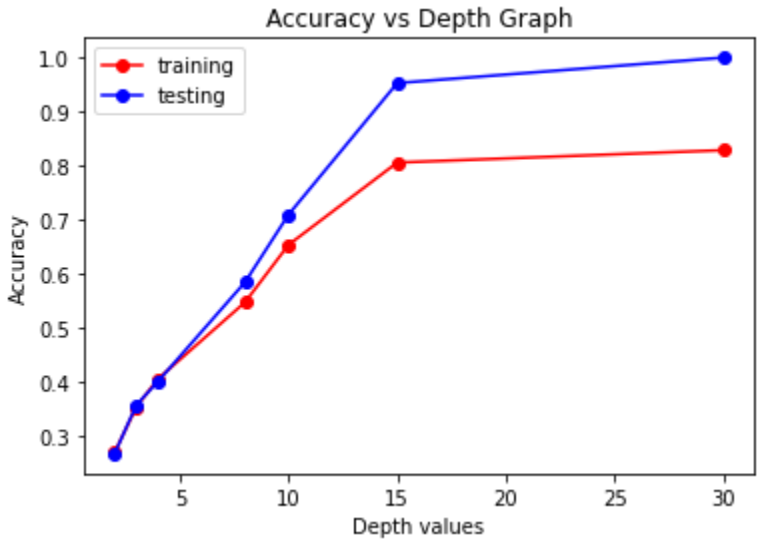
Step 3- Trained the data on the decision tree classifier
Step 4- Measured the performance of the model

Part A:

Accuracy using gini: 0.8206571341648919
Accuracy using entropy: 0.8290234256160632

Hence entropy gives better accuracy on the test data.

Part B:



The best value of depth comes out to be 30.

Part C:

Accuracy using Ensembling: 0.33830240340736234

The performance is very similar in both the methods; in part c the max depth has taken as 3 . Therefore the accuracy comes out to be less, whereas, in parts a and b by analyzing accuracy vs. depth graph we can see that at depth 3, accuracy is somehow the same.

Part D:

Accuracy at different depths

```
For depth  4
Accuracy on train set :  0.07833891086096745
Accuracy on val set :  0.08579251597201096
Accuracy on test set :  0.4093398235473076
For depth  8
Accuracy on train set :  0.08609674475205355
Accuracy on val set :  0.08107696988135077
Accuracy on test set :  0.6226041983571646
For depth  10
Accuracy on train set :  0.08655308792211744
Accuracy on val set :  0.0803163979312443
Accuracy on test set :  0.7578338910860968
For depth  15
Accuracy on train set :  0.08107696988135077
Accuracy on val set :  0.08107696988135077
Accuracy on test set :  0.9040158198965622
For depth  20
Accuracy on train set :  0.08275022817158503
Accuracy on val set :  0.0800121691512017
Accuracy on test set :  0.9132947976878613
For depth  30
Accuracy on train set :  0.08198965622147855
Accuracy on val set :  0.08062062671128689
Accuracy on test set :  0.9181624581685427
```

Best depth at 30 with 91.81 accuracy.

Accuracy at different number of tress

```
For the number of trees  50
Accuracy on train set:  0.08259811378156373
Accuracy on val set:  0.08138119866139337
```

```
Accuracy on test set:  0.909035594767265
For the number of trees  100
Accuracy on train set:  0.08351080012169151
Accuracy on val set:  0.0800121691512017
Accuracy on test set:  0.9152722847581382
For the number of trees  150
Accuracy on train set:  0.08335868573167021
Accuracy on val set:  0.08168542744143596
Accuracy on test set:  0.9172497718284149
```

Best performance at 150 number of tress.

Part E:

```
for estimator: 4
Accuracy on test set :  0.8275022817158503
for estimator : 8
Accuracy on test set :  0.8275022817158503
for estimator : 10
Accuracy on test set :  0.8275022817158503
for estimator : 15
Accuracy on test set :  0.8275022817158503
for estimator : 20
Accuracy on test set :  0.8275022817158503
```

We can see from the above result that the accuracy is greater for the ensembling method RF as compared to the AdaBoost result because AdaBoost tries to overcome bias and variance at the same time by training each individual model in a sequential way, each model learns from its previous error whereas ensembling method just focuses on reducing variance by averaging, in this, each model trained parallel by a random subset of the data. Therefore ensembling method performs better with less bias and less variance.

QUESTION 2: MLP

Part 1:

All parts are present in the code file.

Part 2:

1.

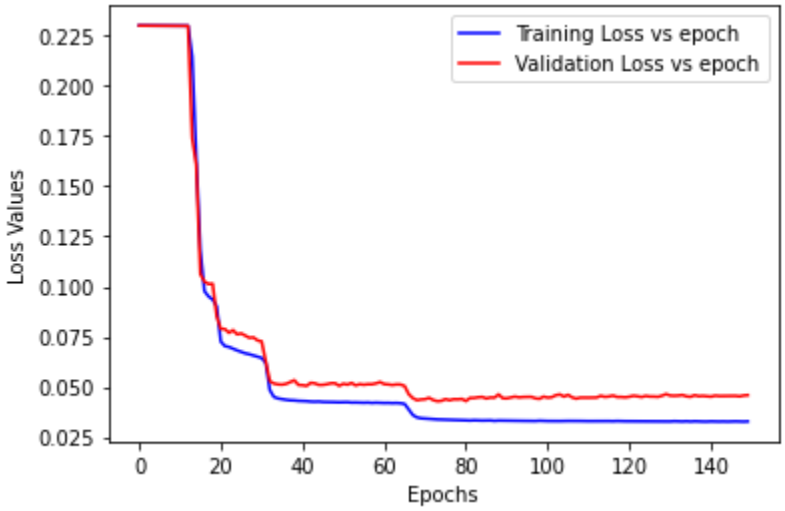
Test accuracies for all activations.
Batch size = 200

Activation Function	Test Accuracy
Linear	91.48
Sigmoid	95.97
Relu	96.99
Leaky Relu	96.92
Tanh	96.75

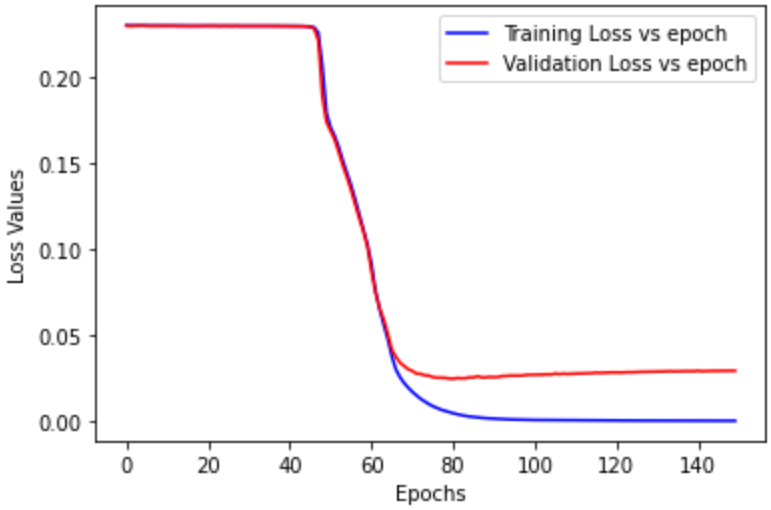
2. Training error vs Epochs and Validation loss vs Epochs

All the activation functions performed well with high accuracies comparison to each other. In every plot, we can see that the curve is converging towards minima with little fluctuation of gradient around the minima. The best performance can be seen in the leaky relu activation plot, the curve is very smoothly converging.

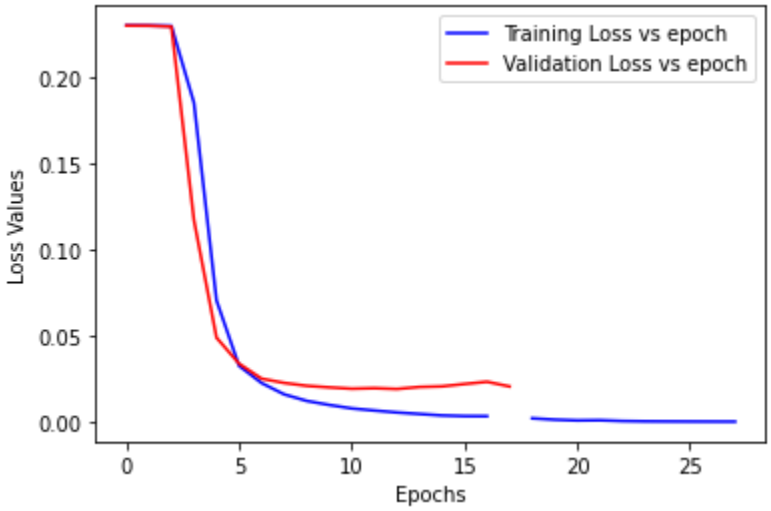
Linear activation function



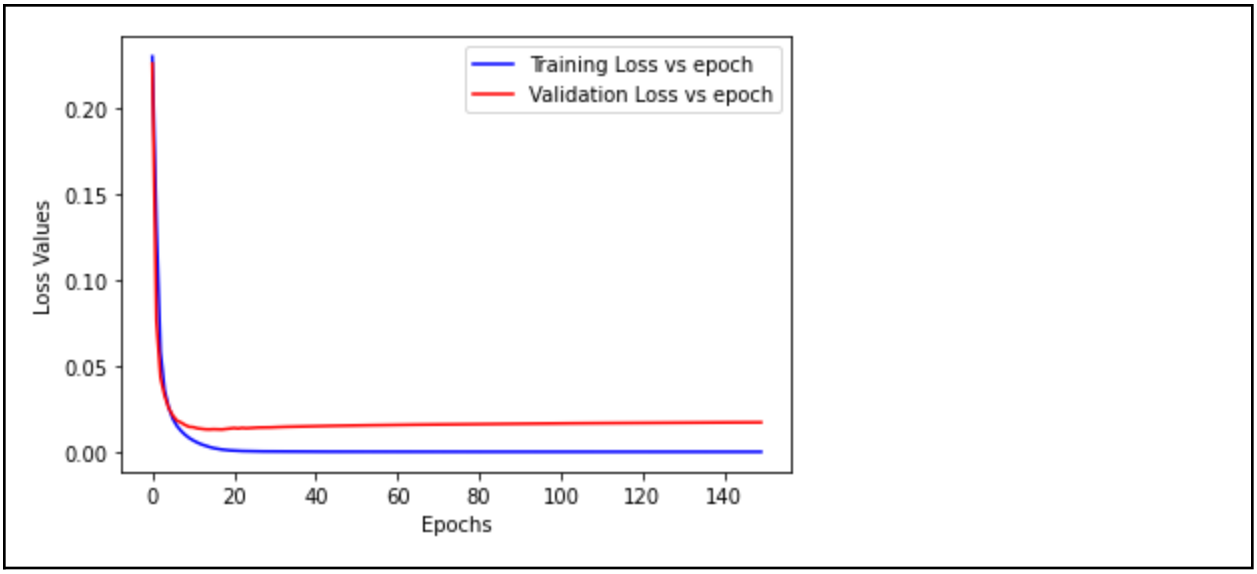
Sigmoid activation function = 50 batch size(for better accuracy)



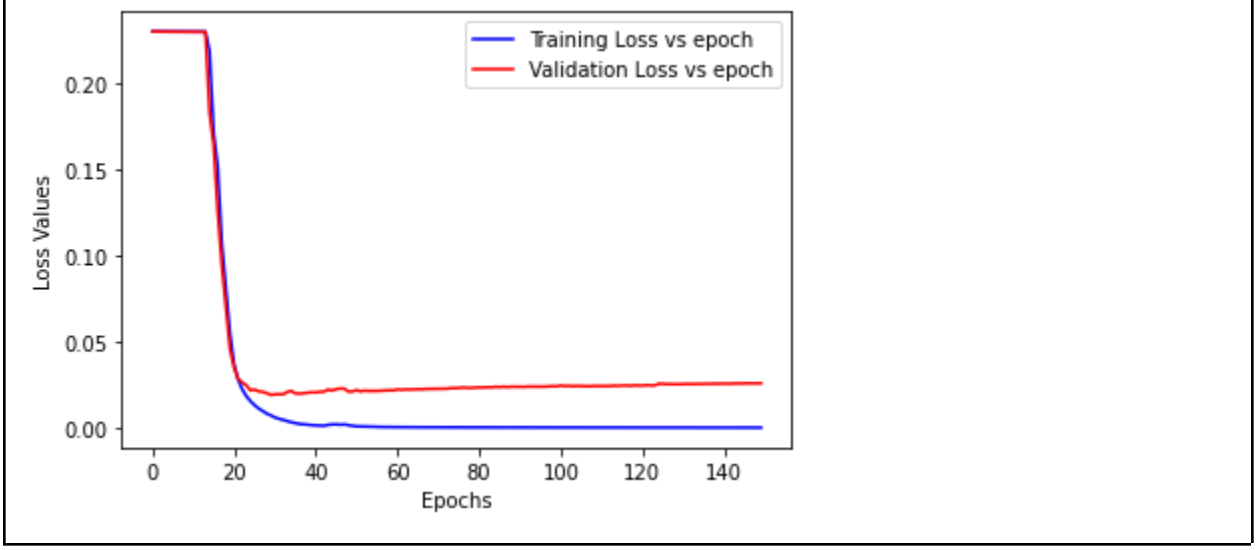
Relu activation function



Leaky Relu activation function



Tanh activation function



3.

The activation function for the output layer for every case should be Softmax. The output probabilities range between 0 - 1 and the sum of all output probabilities is 1 which is the main advantage of using the softmax function. The entropy loss function that we have used gives the output in the range 0-1. This activation function finds out the probability of more than 10 different events. So, if we use the softmax function for multiclassification, it returns the probability of each class with the probability of the target class being the highest.

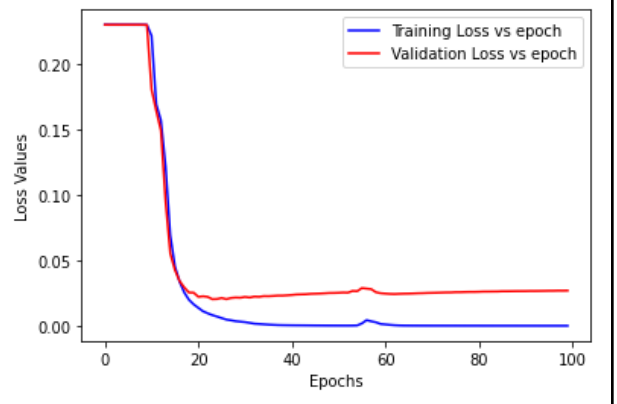
4. Running sklearn on the same dataset with the same parameters.

Activation Function	Test Accuracy (my implementation)	Test Accuracy(Sklearn)
Linear	91.48	0.83
Sigmoid	95.11	0.96
Relu	96.99	0.93
Tanh	96.75	0.93

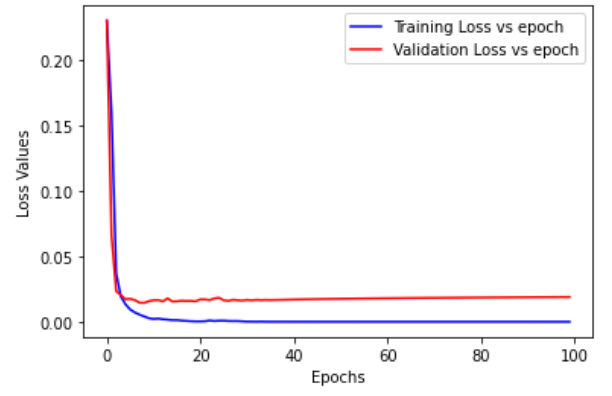
There is very little difference in the accuracies obtained, sklearn gave better results for every activation function. For relu, sigmoid and tanh we got similar results, but for linear, the scratch implementation gave better results as compared to sklearn. The small difference between the performance might be because of the effect of the seed of random_state attribute in sklearn implementation.

5.

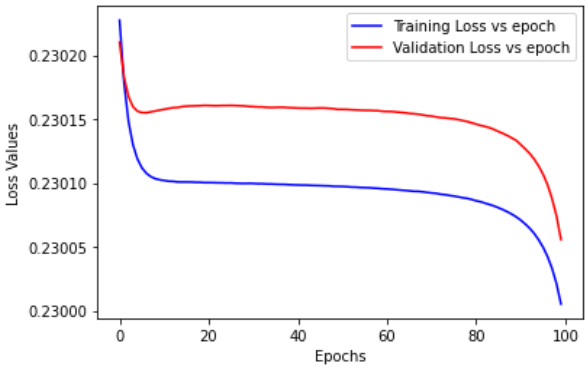
Learning rate	epochs/activation	Test Accuracy
0.001	100/tanh	11.8
0.01	100/tanh	11.10
0.1	100/tanh	97.34
1	100/tanh	95.77



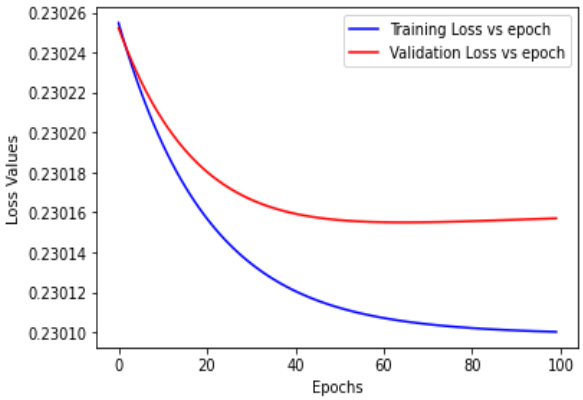
Lr = 1



Lr = 0.1



Lr = 0.01



Lr = 0.001

Observations:

For learning rate 1 :

The model trained well enough, we can see a smooth curve approaching towards zero as the number of epochs increases, also it becomes a little unstable at zero indicating little fluctuation while reaching minima. We are getting quite a high accuracy at learning rate 1, but 1 is a very high learning rate and also unstable. The accuracy is high because the model might be mugged up the data and resulting in high accuracy. Therefore the learning is not the best.

For learning rate 0.1 :

This learning rate performed best among all as it is not very large nor very small. The model shows a smooth loss vs epoch curve with high accuracy for both training and validation datasets. Therefore 0.1 is the best learning rate with a higher convergence comparison to others.

For learning rate 0.01 :

1. We can observe that the graph was smooth for only till 10 epochs, after that the graph fails to converge.
2. This happens because 0.01 is a low learning rate, the graph takes slow steps and hence could not converge with the minima.
3. Since the graph didn't converge very well, low accuracy was achieved.
4. Therefore the learning is not the best, showing a vast difference in convergence.

For learning rate 0.001 :

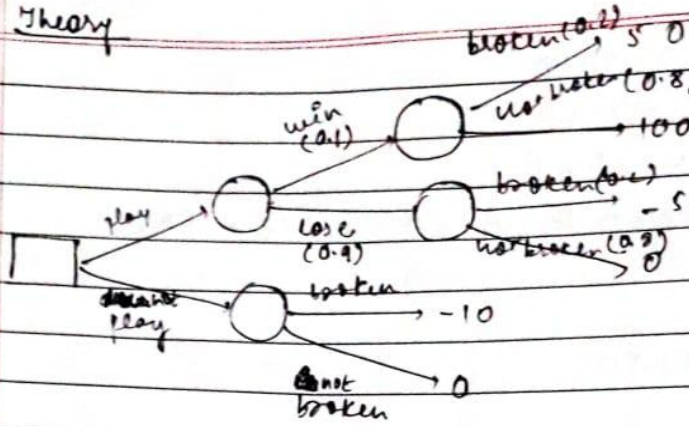
1. We can observe that the graph was smooth up until 10-15 epochs but didn't really converge at the end of 100 epochs.
2. Since 0.001 is a really low value, the graph moves in slow steps and hence is not able to converge.

3. Since the graph didn’t converge as well, we see very less accuracy was achieved.
4. Since the loss didn’t converge, definitely the learning is not the best.

QUESTION 3: Theory

Theory

1. 1)

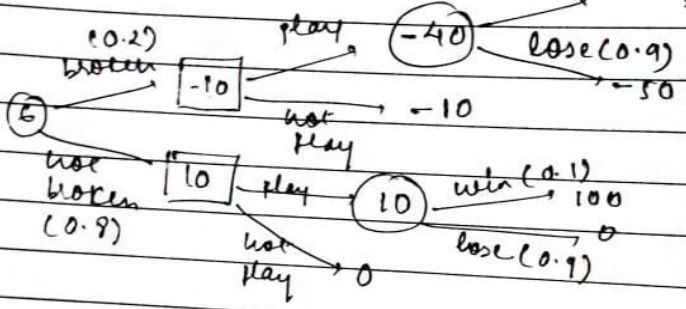


2) $U(play) = 0$ $U(\text{not play}) = -2$

So, player should play in the tournament.

We can collect more information about his leg by having more no. of tests.

3)

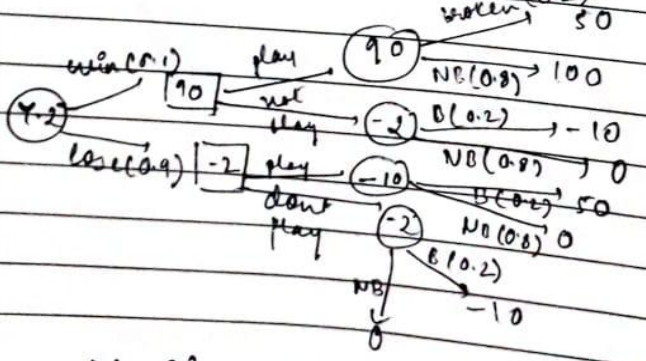


If given perfect info about the state of leg.

Expected value of the information = $E(V_{info}) - E(V_{noinfo})$

$= 6 - 0 = 6$

4)



If given perfect info about winning

Teacher's Signature .....

Peacock
Date : / /
Page No. :

Expected value of information = $E(V_{info}) - E(no\ info)$
$= 7.2 - 0$
$= 7.2$

5) Yes, is it possible. We can do it by replacing with branch after the broken branches and then apply conditional probabilities for the given state of the leg.

2) Let $f = \{0,1\}^d \rightarrow \{0,1\}$ boolean function of variable Therefore, we can express it in logical operator AND, OR, NOT.
We know neural network with 0 hidden layer can represent all AND, NOT. so we can use 1 hidden layer to combine all such functions and hence, we can represent a boolean function with 1 hidden layer.

3) $Y = P(Y=1|X) = \dfrac{\exp(\beta_1 x_1 + \beta_2 x_2)}{1 + \exp(\beta_1 x_1 + \beta_2 x_2)}$

$= \dfrac{1}{1 + \exp(-(\beta_1 x_1 + \beta_2 x_2))}$

Sigmoid function is given by

$\sigma(x) = \dfrac{1}{1+e^{-x}}$

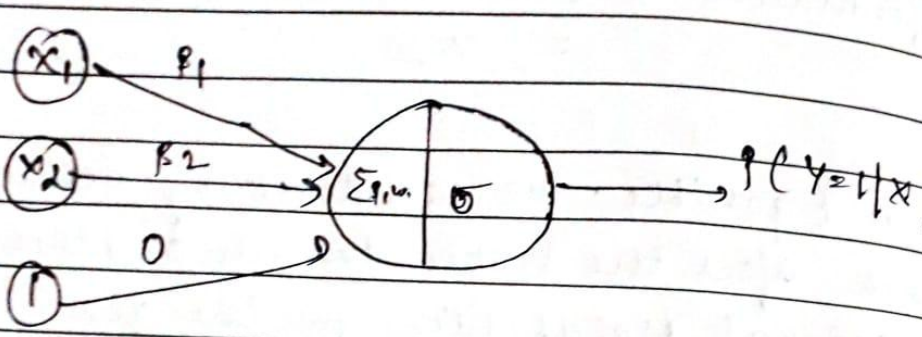
Let assume $X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

$W = [\beta_1, \beta_2]$ and $b = 0$

Therefore $z = wx + b$
$= \beta_1 x_1 + \beta_2 x_2$

Teacher's Signature .....

So, logistic linear regression classifier can learn as



final neural network.