ML ASSIGNMENT-2 REPORT

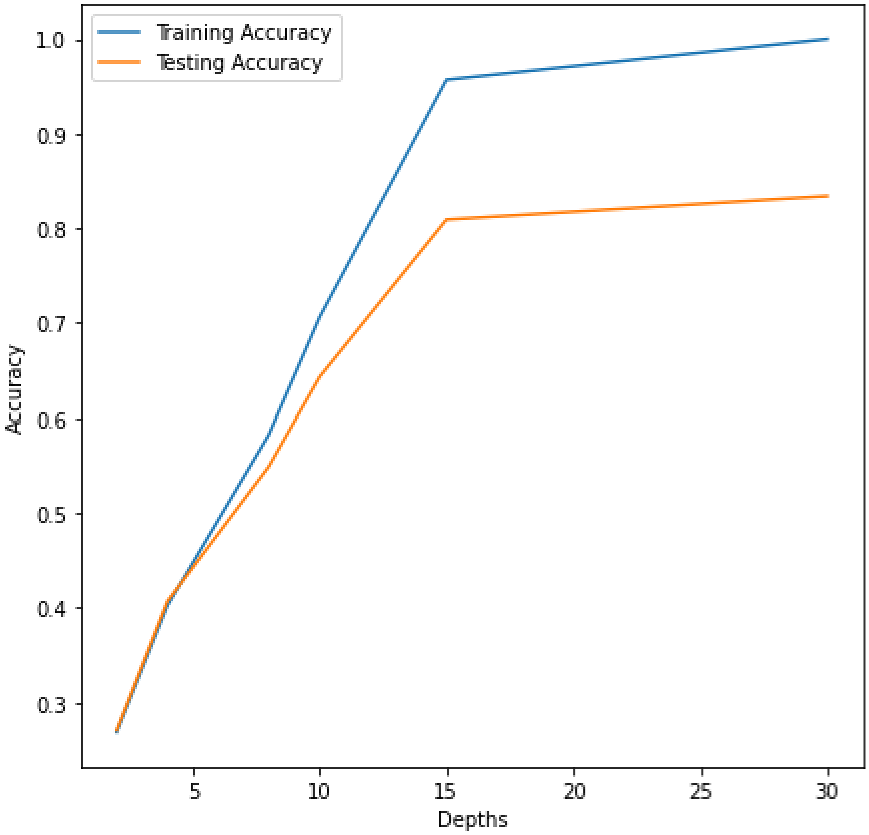
UDIT BHATI
2019281

Solution-1)
The dataset has 12 columns and 43824 rows. After removing the 'No' column from the dataset. I changed NA values to the median from pm2.5 which has originally 2067 NA values. Then I labelled the 'cbwd' column data. after that, I split the data into train, valid, test in a 70:15:15 ratio.

Part-A:
Using Gini Index:
Model Accuracy: 0.8226346212351688
Using Entropy:
Model Accuracy: 0.8303924551262549

Here, can we see Entropy gives better accuracy than the Gini index. We will use Entropy in further parts. Entropy generally performs better than the Gini index for stable data but the complexity of Entropy is higher than the Gini index which results in more processing time.

Part-B:



We can see from the graph that at depth=30, the training accuracy is around 100% which means our model has overfitted the data at this point. At depth=15, both accuracies are at 80%

and the difference between them is minimum, which means the model has not overfitted the data. So we take the best value of depth to be 15.

Part-C:

Accuracy:
training_accuracy : 0.3488068848611292
testing_accuracy : 0.33860663218740494
validation_accuracy : 0.351992698509279
Here, the accuracies are very poor as compared to 1a,1b where we were getting around 80% accuracy but here it is only around 30%. This is because the stumps are very weak classifiers at the max depth of 3.

Part-D:

(Max Depth, Total Stumps)	Training Accuracy	Testing Accuracy	Validation Accuracy
4,100	0.41	0.40	0.41
8,100	0.65	0.61	0.62
10, 100	0.82	0.76	0.76
15, 100	0.99	0.90	0.90
20, 100	0.99	0.91	0.91
4, 150	0.41	0.40	0.41
8, 150	0.66	0.62	0.63
10,150	0.82	0.76	0.76
15, 150	0.99	0.90	0.91
20, 150	0.99	0.91	0.92
4, 200	0.41	0.41	0.41
8, 200	0.66	0.62	0.63
10, 200	0.82	0.75	0.76
15, 200	0.99	0.90	0.91
20, 200	0.99	0.91	0.92

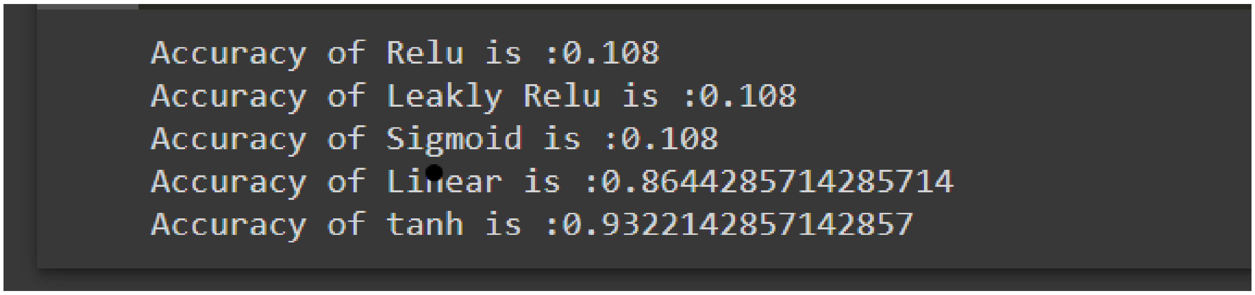
Here, we can see that as the max depth increases the training accuracy increases with other accuracies, at a max depth of 20 we have 99% of training accuracy which is implying the overfitting of the model, also increasing the total stumps slightly increases the accuracy.

Part-E:
Using AdaBoost:
Model Accuracy: 0.8323699421965318

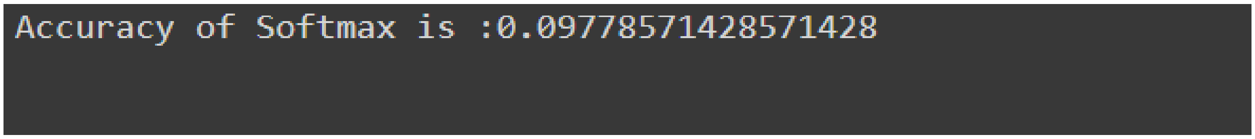
Estimator	Training Accuracy	Testing Accuracy	Validation Accuracy
4	0.87	0.74	0.75
8	0.86	0.74	0.74
10	0.86	0.72	0.73
15	0.87	0.74	0.74
20	0.86	0.73	0.73

Adaboost is a boosting technique that is used to boost the overall performance of the decision tree. AdaBoost uses a sequel ensembling while RF uses parallel ensembling
From the above table of 1d, we can see that RF tends to overfit the model at high max depths while it also goods. Overall we can see RF gives better performance.

Solution-2)
train.csv, test.csv is first combined with each other which gives a total data of 70,000 rows and 785 features.
Since the pixel values are between 0-255, scaling is done after dividing them by 255, in order to get more accurate results.
Train,test,valid is split into 70:20:10 ratio.
Part-1:
Architecture of neural network=[input,256,64,32,output], with learning rate=0.08, epochs=150, normal weights initialization is used.
Result:

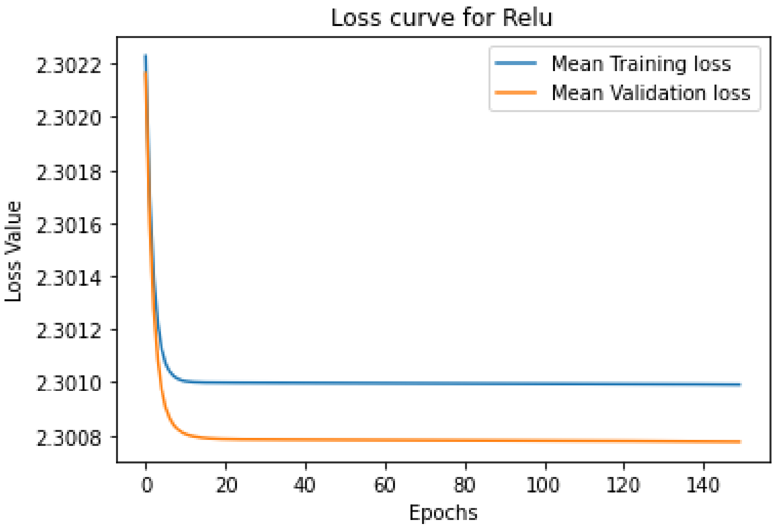


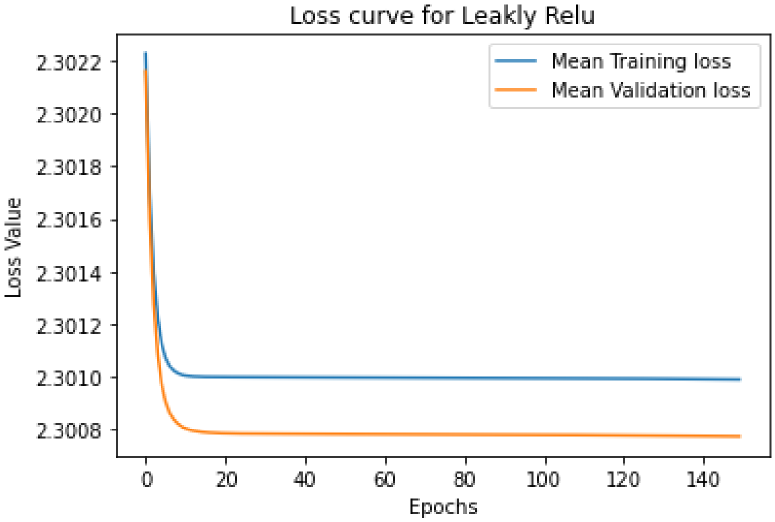
```
Accuracy of Relu is :0.108
Accuracy of Leakly Relu is :0.108
Accuracy of Sigmoid is :0.108
Accuracy of Linear is :0.8644285714285714
Accuracy of tanh is :0.9322142857142857
```

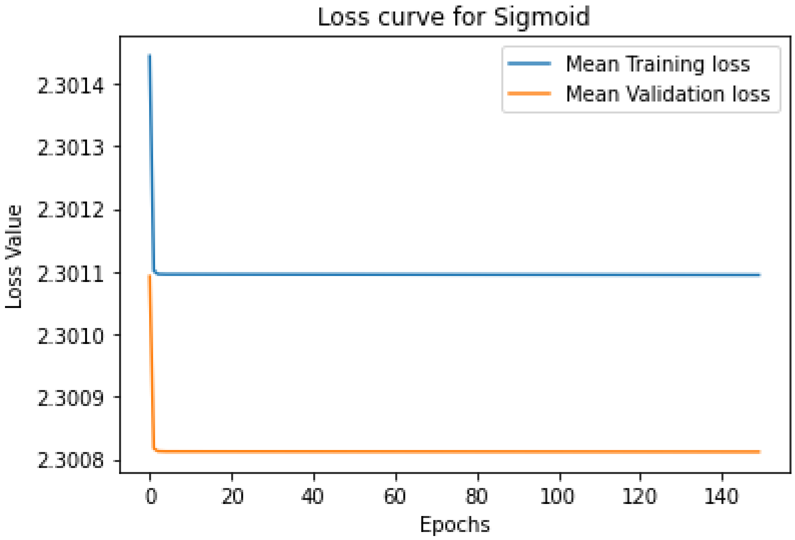


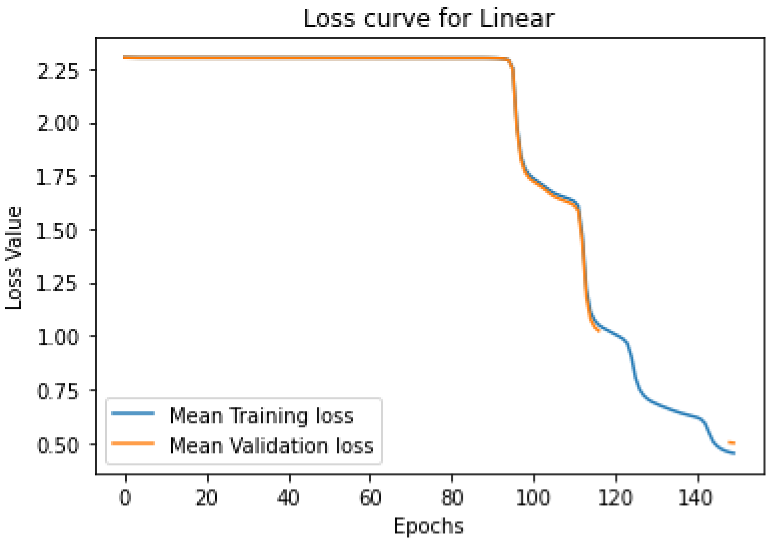
```
Accuracy of Softmax is :0.09778571428571428
```

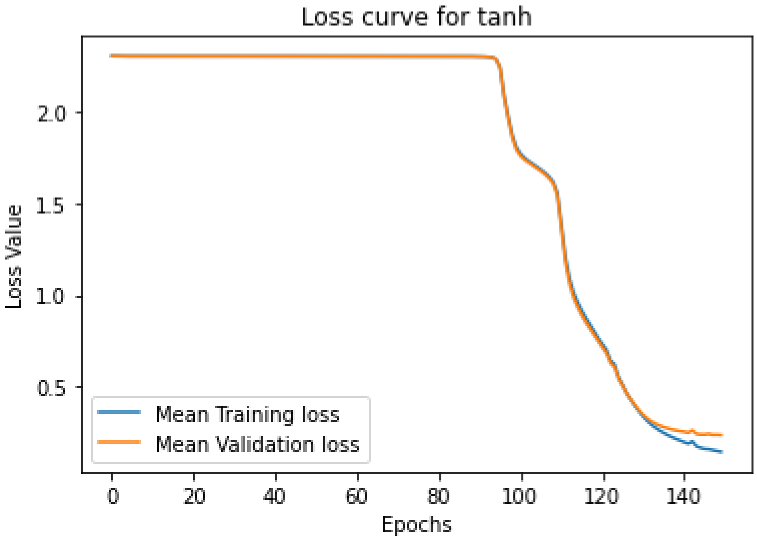
Part-2:

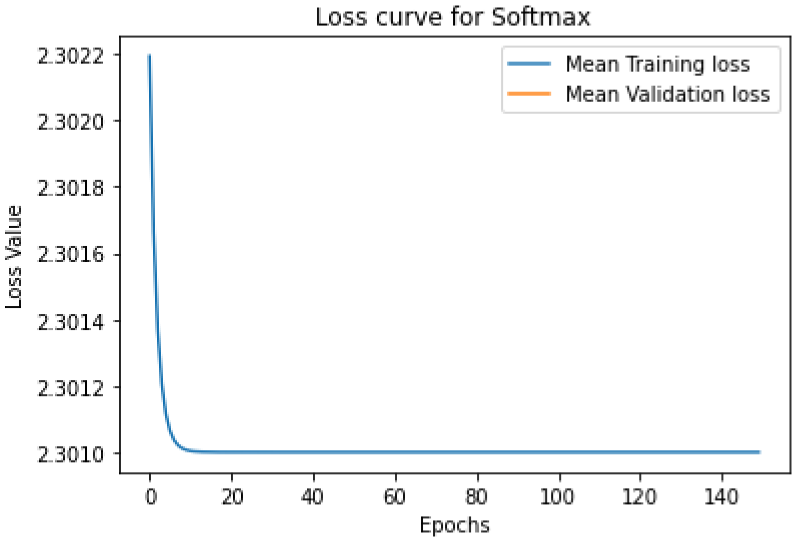

Loss curve for Relu
Mean Training loss
Mean Validation loss
Loss Value
2.3022
2.3020
2.3018
2.3016
2.3014
2.3012
2.3010
2.3008
0
20
40
60
80
100
120
140
Epochs


Loss curve for Leakly Relu
Mean Training loss
Mean Validation loss
Loss Value
2.3022
2.3020
2.3018
2.3016
2.3014
2.3012
2.3010
2.3008
0
20
40
60
80
100
120
140
Epochs


Loss curve for Sigmoid
Mean Training loss
Mean Validation loss
Loss Value
2.3014
2.3013
2.3012
2.3011
2.3010
2.3009
2.3008
0
20
40
60
80
100
120
140
Epochs


Loss curve for Linear
Loss Value
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0
20
40
60
80
100
120
140
Epochs
Mean Training loss
Mean Validation loss

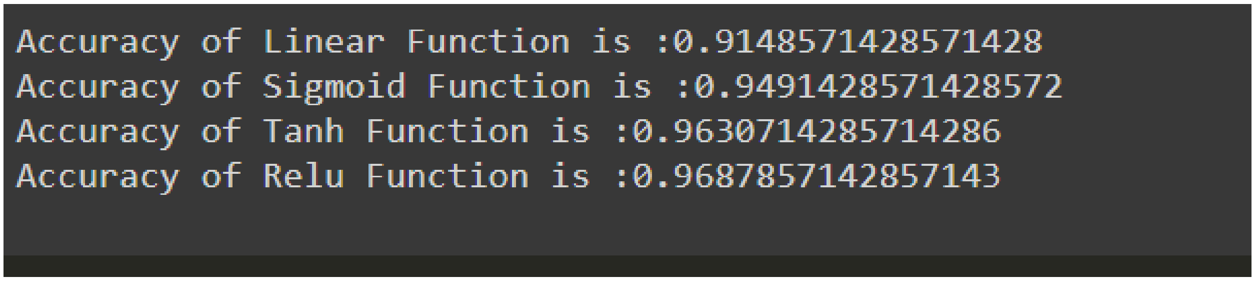

Loss curve for tanh
Loss Value
2.0
1.5
1.0
0.5
0
20
40
60
80
100
120
140
Epochs
Mean Training loss
Mean Validation loss


Loss curve for Softmax
Loss Value
2.3022
2.3020
2.3018
2.3016
2.3014
2.3012
2.3010
0
20
40
60
80
100
120
140
Epochs
Mean Training loss
Mean Validation loss

The loss plot of linear is highly unstable, we can also observe that its loss remains constant for 80 epochs and only after that reduces significantly, its accuracy is also pretty good.
The loss plot of tanh is also similar to that of linear, but its accuracy is better than any other curve.
Relu, leakyrelu and sigmoid have similar loss curves.

Part-3:
The softmax function will be used as the activation function of each output layer because here we have to predict results of 10 different labels, and since it is a multi-label problem we would be requiring the probabilities to predict the final class, and softmax function is the only activation function here which provides us with the probabilities.

Part-4:
Results from sklearn:

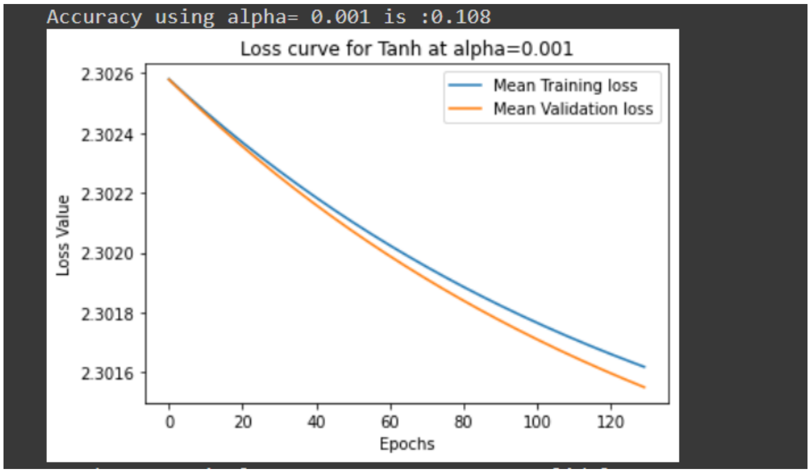


```
Accuracy of Linear Function is :0.9148571428571428
Accuracy of Sigmoid Function is :0.9491428571428572
Accuracy of Tanh Function is :0.9630714285714286
Accuracy of Relu Function is :0.9687857142857143
```

The difference in results between my implementation and sklearn is maybe because sklearn by default uses random weight initialization while we have used normal weight initialization, random produces more accurate results.

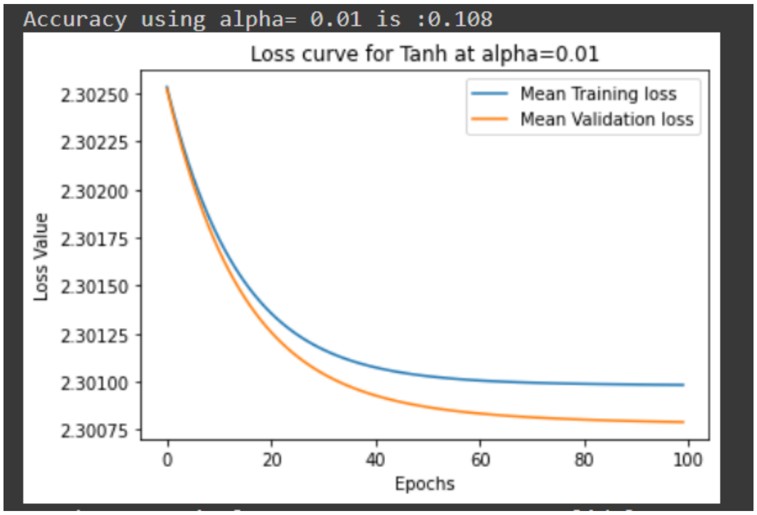
Part-5:
Since tanh activation function gave the highest accuracy i will be using it for this part, with epochs=100.

For alpha=0.001



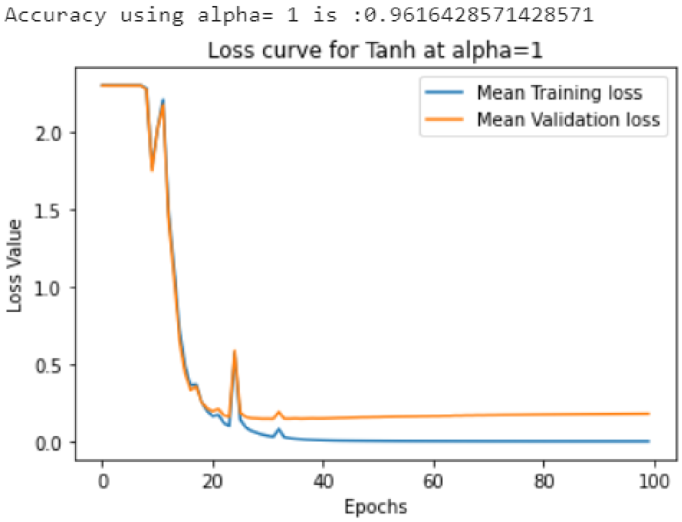
```
Accuracy using alpha= 0.001 is :0.108
```

For alpha=0.01



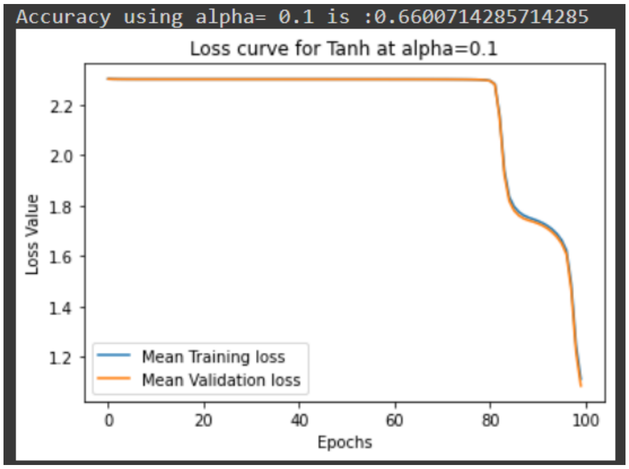
```
Accuracy using alpha= 0.01 is :0.108
```

For alpha=1



```
Accuracy using alpha= 1 is :0.9616428571428571
```

For alpha=0.1

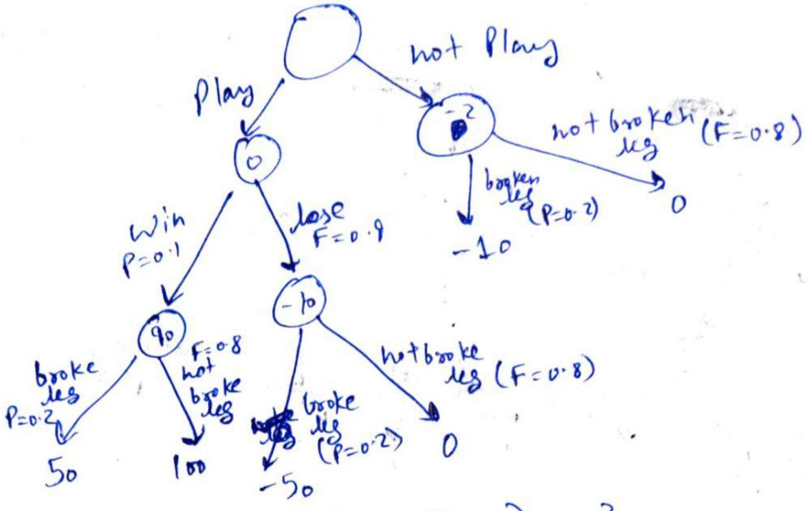


```
Accuracy using alpha= 0.1 is :0.6600714285714285
```

Alpha at 0.01 and 0.001 gives the best results, other loss plots are disrupted and not smooth implying the overshooting of local minima during convergence.

Solution-3)

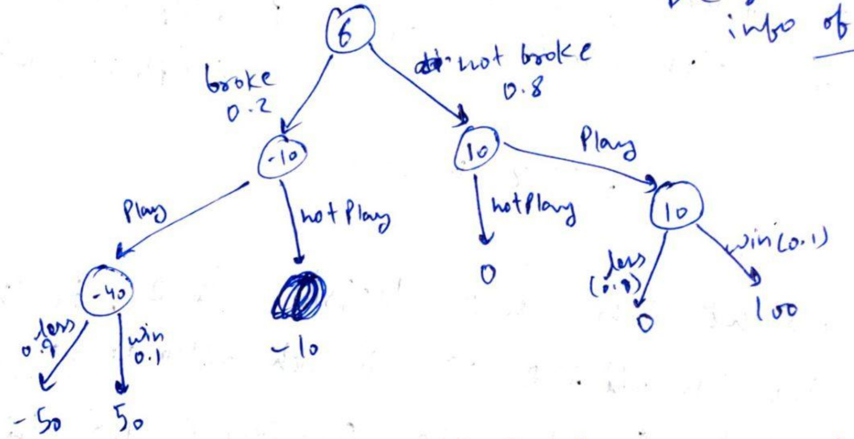
1)



D(Play) = 0 , D(not Play) = -2

DT given Perfect info of leg.

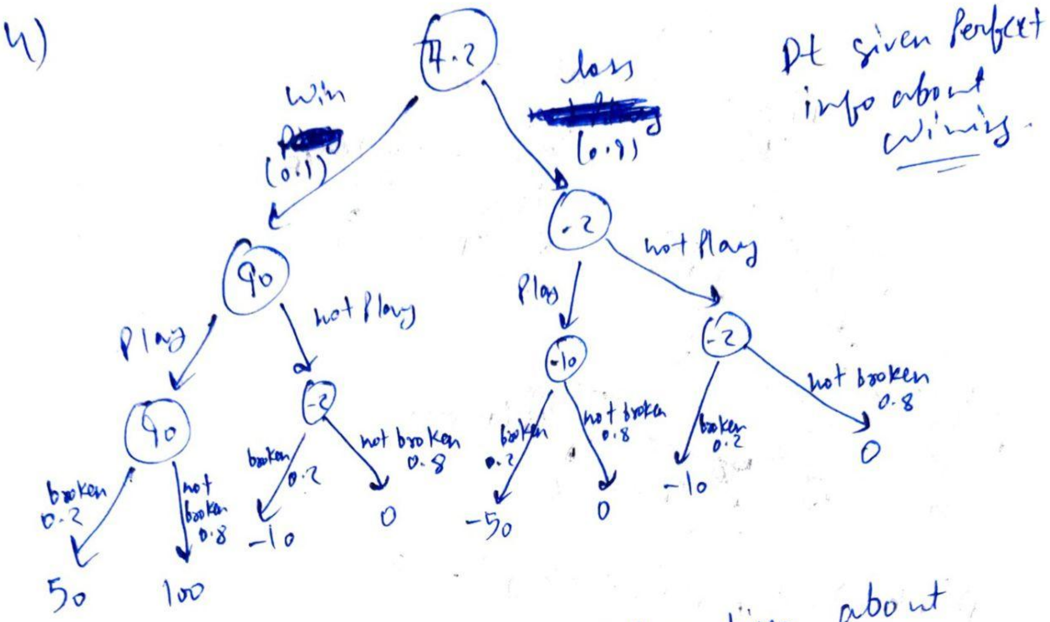
3)



⇒ E(D information) − E(D not information) = Expected value ~~of~~ of perfect information about the state of leg

⇒ 6 − 0 = 6

4)



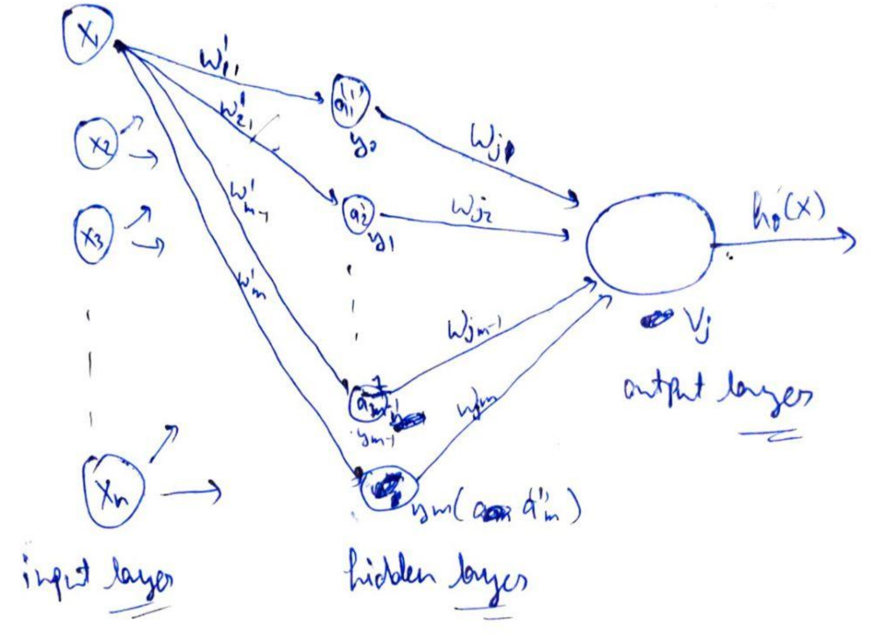
Expected value of perfect information about whether we'll win the race = E(D information) − E(D not information)

= 7.2 − 0 = 7.2

5) Yes we can use decision tree, in this Case we can add the Win branch after the leg branch and Compute the Conditional Probabilities accordingly.

(2.) ~~fo~~ $F : \{0,1\}^{d} \rightarrow \{0,1\}$

let input vector be $X = (x_1, x_2 . \ldots, x_n)$



let the hidden layer has m neurons.

dimension of $w'_{ij} = n \times m$.

dimension of $w_{ij} = m \times 1$

$y_0 = a_1^{(1)} = \sigma (w'_{11} x_1 + w'_{12} x_2 + \ldots + w'_{1n} x_n)$

where σ is the sigmoid activation function.

similarly.

$y_m = a_m^{(1)} = (w'_{m1} x_1 + w'_{m2} x_2 + \ldots + w'_{mn} x_n)$

$V_j = \sigma (W_{j1} y_0 + W_{j2} y_1 + \ldots + W_{jm} y_m)$

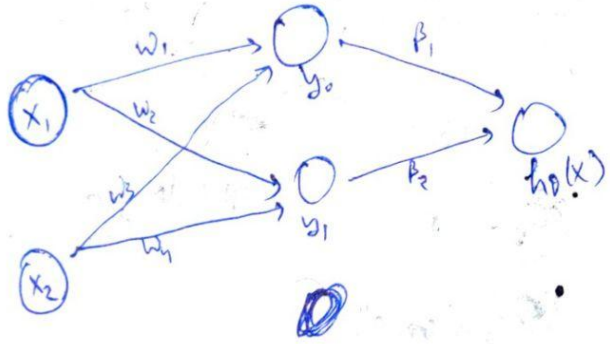
Since the range of Sigmoid is between 0 to 1.

$V_j \in [0,1]$

$$h_\theta(x) = \begin{cases} 1 & \text{if } V_j \geq 0.5 \\ 0 & \text{if } V_j < 0.5 \end{cases}$$

⇒ hence Boved.

(3.)



Sigmoid activation function, $F(x) = \frac{1}{1+e^{-x}}$

$y_0 = f(w_1 x_1 + w_3 x_2)$

$y_1 = F(w_3 x_1 + w_4 x_2)$

$h_\theta(x) = F(y_0 \beta_1 + y_1 \beta_2) = \frac{1}{1+e^{-(y_0\beta_1 + y_1\beta_2)}}$

Prob. of not of $h_\theta(x) = 1 - \frac{1}{1+e^{-(y_0\beta_1+y_1\beta_2)}}$

$$= \frac{e^{-(y_0\beta_1+y_1\beta_2)}}{1+e^{-(y_0\beta_1+y_1\beta_2)}}$$