ML Assignment 2

Anshul Mendiratta
2018219

Question 1

(a) Models built on datasets with high dimensionality(having a large number of features) tend to overfit. Further, the lower the dimensions the faster we are able to fit the training data. Hence, in datasets like these we want to reduce the number of features. PCA helps us in doing so.
PCA is a linear unsupervised learning algorithm which helps us construct a set of uncorrelated features from a set of correlated features. We define a set of axis using a linear combination of the features and rank them according to the amount of variance.
We want the highest ranked axis to have maximum variability and hence the ability to capture the most information. All other subsequent axis should capture the remaining variability while not being correlated with the other axis. We will extract the most important axis from the ones obtained and treat them as our new set of features.

(b) As seen from above, datasets with large number of features aren't desirable. SVD is a way of reducing these features using matrix decomposition.
Suppose we have a matrix A, which represents the set of features and their values for each datapoint. Using SVD, we can decompose A into,

$$A = U * \Sigma * V^T$$

Where A is our m x n original matrix, U is an mxm matrix of Left singular vectors, Σ is an mxn diagonal matrix and V is an n x n matrix of right singular vectors. Using these matrices we can represent the original matrix as a linear combination of low rank matrices. We can then pick the first few singular vectors and truncate the matrices accordingly. Hence, we have reduced our features.

(c) t-SNE (t-Distributed Stochastic Neighbor Embedding) is a non-linear algorithm which helps in reducing the dimension of a dataset. Contrary to PCA, which places emphasis on placing dissimilar data points far apart, t-SNE also focuses on representing similar ones closer. Furthermore, unlike PCA, t-SNE is based on probability distributions to find relationships between features. There are mainly 3 steps -

1. We measure similarities between points in the high dimensional space.
2. Then we try to measure similarities between points in the low dimensional space.
3. Finally, we want to map both of these structures and optimize the similarity measures using a cost function.

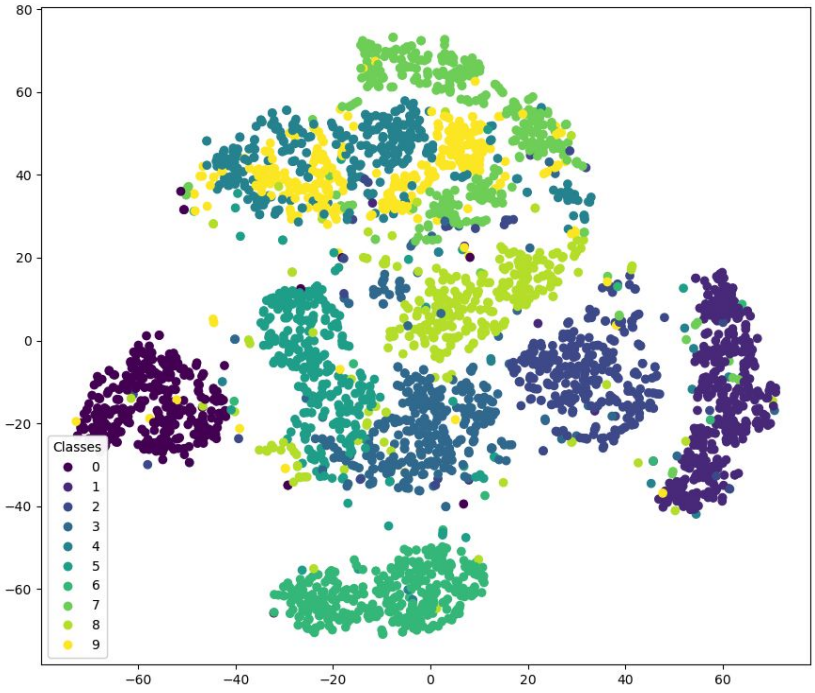
(d) Stratified sampling is a way of sampling the dataset such that the training and test data after splitting have approximately the same percentages of each target class. As expected, the class frequency of both test and train are similar. (Code in python file)
Class frequencies (Class vs frequency percentage) of train and test are -

{0: 9.523809523809524, 1: 11.755952380952381, 2: 9.345238095238095, 3: 10.089285714285715, 4: 9.910714285714285, 5: 9.464285714285714, 6: 10.505952380952381, 7: 10.267857142857142, 8: 9.761904761904763, 9: 9.375}

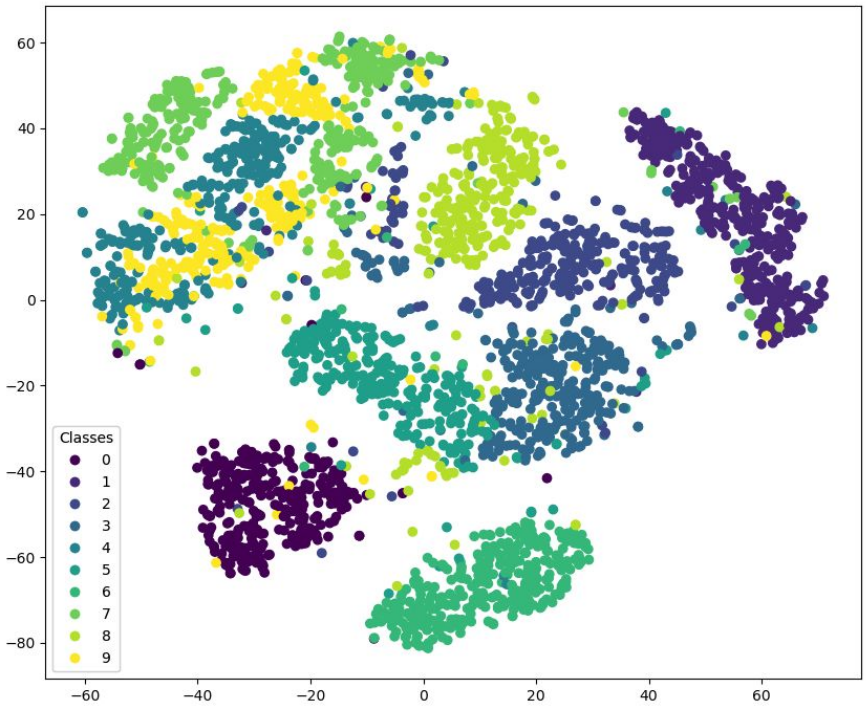
{0: 9.523809523809524, 1: 11.785714285714285, 2: 9.404761904761905, 3: 10.119047619047619, 4: 9.880952380952381, 5: 9.523809523809524, 6: 10.476190476190476, 7: 10.238095238095237, 8: 9.761904761904763, 9: 9.285714285714286}

(e) Using PCA and training Logistic Regression Model we get accuracy of 0.883. We then apply t-SNE on the training data obtained after applying PCA and plot a scatter plot, where x and y axis are the features obtained by t-SNE and each point is colored according to the class it belongs to.

We get the following plot -



(f) Using SCD and training Logistic Regression Model we get accuracy of 0.878. We then apply t-SNE and make a graph following same steps as above. We get the following plot -



(g) We get very similar accuracies for SVD and PCA, even though PCA performs negligibly better. We do see from the graphs obtained that the features extracted from SVD and PCA are different as the clusters of classes are not at the same place in both the graphs. Hence even though different features are being extracted in SVD and PCA, they perform well in both cases.

Question 2

We apply bootstrapping in the following way -

1) First we create a holdout from X. In this case we kept 20% of X as holdout. The rest is used for creating samples and we call this testX.
2) Next we create B samples each of size B.
3) We train our model on each sample S_b and compute values for each x in holdout. Hence we have B predicted values for each x in holdout.
4) Now we calculate bootstrap estimate, bias and variance for every point x in testX
5) We then average these values over testX and report them
6) Similarly to find MSE, we find MSE at each x in testX and calculate MSE - bias^2 - variance at every point x and report its mean.

(a) The bias and variance of the model came out to be - 0.169 and - 0.53 respectively.

(b) MSE - bias^2 - variance came out to be = -0.23
We know MSE = bias^2 + variance + irreducible uncertainty
Hence we can see that our bootstrap model satisfies the equation as -0.23 is a comparatively smaller value and can be attributed to irreducible uncertainty

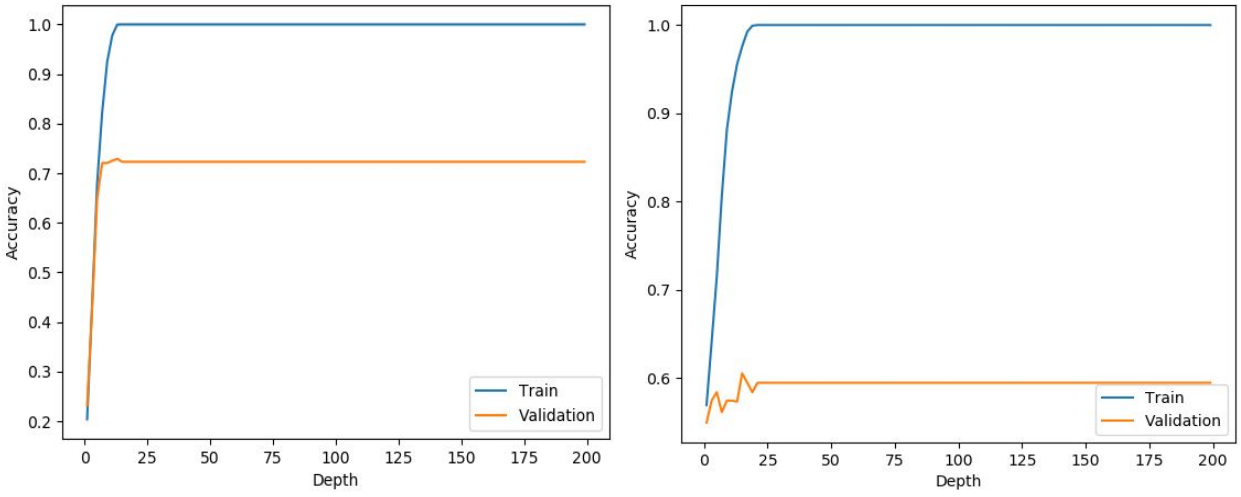
Question 3

(a)

1. Dataset A
    a. Decision Tree - Applying grid search and K fold we found the average accuracy (over all the folds) for depths from 1 to 25. We got the best accuracy at depth 16.
    b. Gaussian Naive Bayes - Using K fold we found the best accuracy of 0.619 was achieved at fold 4. Further average K fold accuracy was 0.59

2. Dataset B
    c. Decision Tree - Applying grid search and K fold we found the average accuracy (over all the folds) for depths from 1 to 25. We got the best accuracy at depth 20.
    d. Gaussian Naive Bayes - Using K fold we found the best accuracy of 0.595 was achieved at fold 2. Further average K fold accuracy was 0.569

(b) Plots for training and validation accuracy vs tree depth for dataset A and B are (Left is A and right is B) -



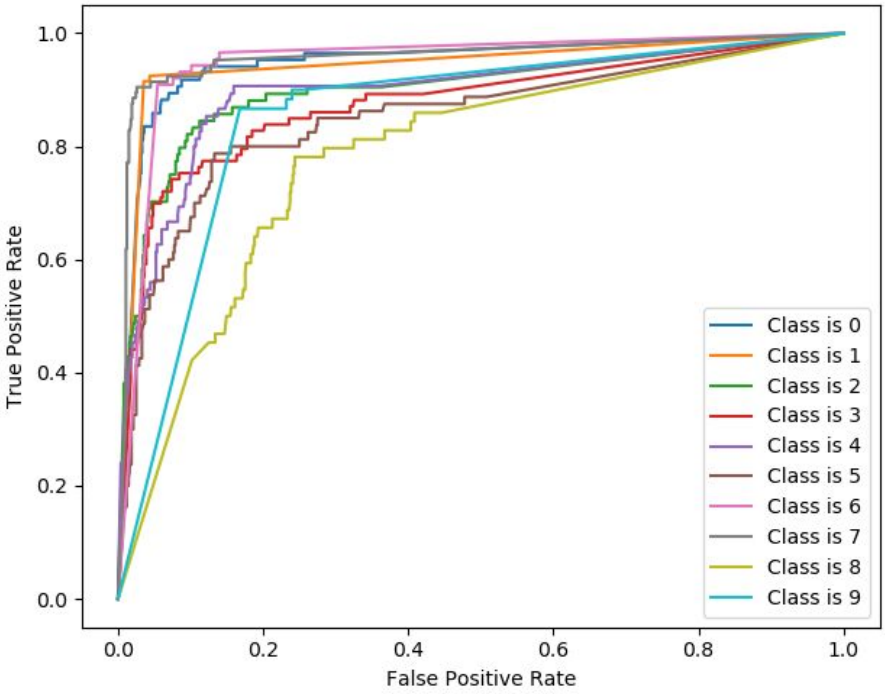
(c) Implemented in code and saved in weights folder

(d)

1. For dataset A
    a. Gaussian Naive Bayes

```
Confusion Matrix -
[63, 0, 7, 1, 4, 6, 2, 1, 1, 1]
[0, 97, 1, 6, 0, 6, 3, 3, 6, 1]
[2, 0, 31, 1, 1, 0, 1, 0, 1, 0]
[0, 2, 6, 40, 0, 4, 0, 0, 1, 0]
[0, 1, 1, 1, 18, 1, 0, 2, 0, 0]
[1, 2, 1, 0, 0, 13, 1, 0, 2, 0]
[4, 2, 18, 10, 6, 5, 80, 0, 2, 0]
[0, 0, 0, 1, 1, 1, 0, 39, 2, 3]
[9, 0, 17, 20, 10, 36, 1, 1, 29, 3]
[6, 2, 2, 13, 35, 8, 0, 59, 20, 52]
Accuracy 0.55
Macro precission 0.6467557459590406
Precission for every class {0: 0.7325581395348837, 1: 0.7886178861788617, 2: 0.
8378378378378378, 3: 0.7547169811320755, 4: 0.75, 5: 0.65, 6: 0.629921259842519
7, 7: 0.8297872340425532, 8: 0.23015873015873015, 9: 0.2639593908629442}
Macro Recall 0.5458237103326363
Recall for every class {0: 0.7411764705882353, 1: 0.9150943396226415, 2: 0.3690
4761904761907, 3: 0.43010752688172044, 4: 0.24, 5: 0.1625, 6: 0.909090909090909
1, 7: 0.37142857142857144, 8: 0.453125, 9: 0.8666666666666667}
Macro F1 score 0.5235258300489847
F1 score for every class {0: 0.736842105263158, 1: 0.8471615720524018, 2: 0.512
3966942148761, 3: 0.5479452054794521, 4: 0.36363636363636365, 5: 0.26, 6: 0.744
1860465116279, 7: 0.5131578947368421, 8: 0.30526315789473685, 9: 0.404669260700
38917}
Micro Precission 0.55
Micro Recall 0.55
Micro F1Score 0.55
```

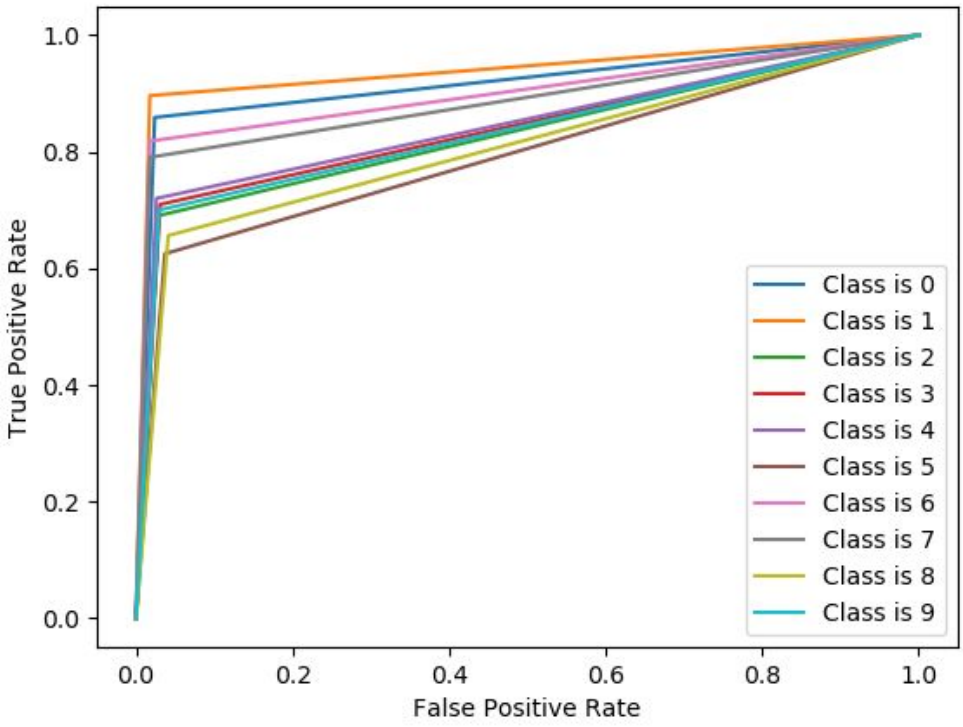
ROC Plot -



b. Decision Tree

```
Confusion Matrix -
[73, 0, 1, 2, 0, 7, 1, 3, 3, 1]
[0, 95, 3, 3, 1, 2, 1, 2, 0, 1]
[0, 2, 58, 4, 3, 2, 5, 2, 4, 0]
[3, 0, 5, 66, 1, 3, 0, 3, 6, 2]
[1, 3, 1, 1, 54, 1, 1, 2, 2, 8]
[3, 0, 5, 8, 2, 50, 4, 2, 3, 1]
[0, 0, 2, 0, 1, 9, 72, 0, 1, 0]
[0, 1, 5, 1, 1, 1, 0, 83, 0, 4]
[2, 5, 4, 6, 5, 3, 2, 4, 42, 1]
[3, 0, 0, 2, 7, 2, 2, 4, 3, 42]
Accuracy 0.7559523809523809
Macro precission 0.7444519406874827
Precission for every class {0: 0.8021978021978022, 1: 0.8796296296296297, 2: 0.
725, 3: 0.7415730337078652, 4: 0.7297297297297297, 5: 0.6410256410256411, 6: 0.
8470588235294118, 7: 0.8645833333333334, 8: 0.5675675675675675, 9: 0.6461538461
538462}
Macro Recall 0.7465111562995143
Recall for every class {0: 0.8588235294117647, 1: 0.8962264150943396, 2: 0.6904
761904761905, 3: 0.7096774193548387, 4: 0.72, 5: 0.625, 6: 0.8181818181818182,
7: 0.7904761904761904, 8: 0.65625, 9: 0.7}
Macro F1 score 0.744666756858741
F1 score for every class {0: 0.8295454545454546, 1: 0.8878504672897196, 2: 0.70
73170731707318, 3: 0.7252747252747254, 4: 0.7248322147651007, 5: 0.632911392405
0633, 6: 0.8323699421965318, 7: 0.8258706467661692, 8: 0.6086956521739131, 9: 0
.6719999999999999}
Micro Precission 0.7559523809523809
Micro Recall 0.7559523809523809
Micro F1Score 0.7559523809523809
```

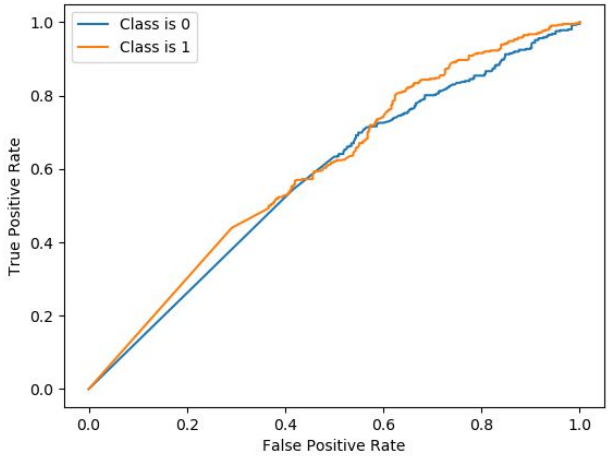
ROC Curve -



2. For Dataset B

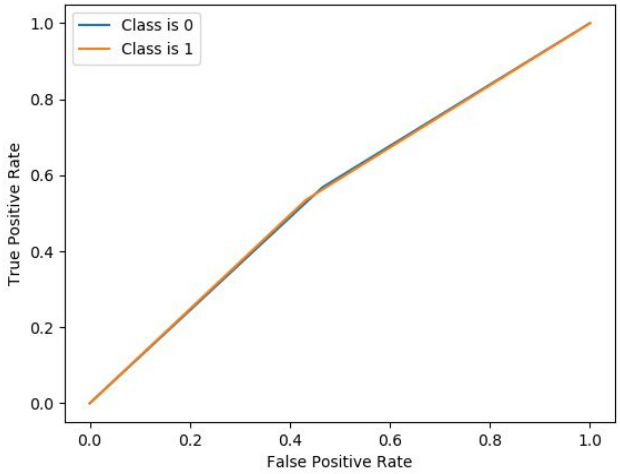
a. Gaussian

```
Confusion Matrix -
[261, 215]
[151, 213]
Accuracy 0.5642857142857143
Macro precission 0.5667420814479638
Precission for every class {0: 0.5483193277310925, 1: 0.5851648351648352
}
Macro Recall 0.5655793485164686
Recall for every class {0: 0.633495145631068, 1: 0.4976635514018692}
Macro F1 score 0.5628583128583129
F1 score for every class {0: 0.5878378378378378, 1: 0.537878787878788}
```



b. Decision Tree

```
Confusion Matrix -
[234, 199]
[178, 229]
Accuracy 0.5511904761904762
Macro precission 0.5515346335207767
Precission for every class {0: 0.5404157043879908, 1: 0.5626535626535627
}
Macro Recall 0.5515039470102532
Recall for every class {0: 0.5679611650485437, 1: 0.5350467289719626}
Macro F1 score 0.551174573929065
F1 score for every class {0: 0.5538461538461538, 1: 0.5485029940119761}
```



Question 4

1. Dataset 1
   Accuracy of my implementation - 0.53
   Accuracy of skLearn's implementation - 0.55

2. Dataset 2
   Accuracy of my implementation - 0.55
   Accuracy of skLearn's implementation - 0.56

Rest of the Questions have been attached below

Q.5

a) Initial Entropy,

$$H(Y) = -\sum_{i=1}^{n} P(Y=y_i) \log_2 P(Y=y_i)$$

$$= -\left(\frac{5}{14}\log_2\frac{5}{14} + \frac{9}{14}\log_2\frac{9}{14}\right)$$

$$= -(-0.9402) \approx 0.94$$

Now, we find information gain for each parameter,

$$IG(X) = H(Y) - H(Y|X)$$

$$\therefore IG(\text{Outlook}) = 0.94 - \left[\frac{5}{14}\left(\frac{3}{5}\log\frac{3}{5} + \frac{2}{5}\log\frac{2}{5}\right) + \frac{4}{14}\left(\frac{4}{4}\log\frac{4}{4}\right) + \frac{5}{14}\left(\frac{3}{5}\log\frac{3}{5} + \frac{2}{5}\log\frac{2}{5}\right)\right]$$

$$= 0.94 + \left[\frac{10}{14}\left(\frac{3}{5}\log\frac{3}{5} + \frac{2}{5}\log\frac{2}{5}\right)\right]$$

$$= 0.94 - 0.6935 \approx 0.24$$

$$IG(\text{Temperature}) = 0.94 + \left[\frac{4}{14}\left(\frac{2}{4}\log\frac{2}{4} + \frac{2}{4}\log\frac{2}{4}\right) + \frac{6}{14}\left(\frac{4}{6}\log\frac{4}{6} + \frac{2}{6}\log\frac{2}{6}\right) + \frac{4}{14}\left(\frac{3}{4}\log\frac{3}{4} + \frac{1}{4}\log\frac{1}{4}\right)\right]$$

$$= 0.94 + \left(-\frac{4}{14} - 0.39 - 0.23\right)$$

$$= 0.94 + (-0.90)$$

$$= 0.04$$

$$IG(\text{Humidity}) = 0.94 + \left[\frac{7}{14}\left(\frac{4}{7}\log\frac{4}{7} + \frac{3}{7}\log\frac{3}{7}\right) + \frac{7}{14}\left(\frac{6}{7}\log\frac{6}{7} + \frac{1}{7}\log\frac{1}{7}\right)\right]$$

$$= 0.94 + [-0.49 + (-0.29)]$$

$$= 0.94 - 0.78$$

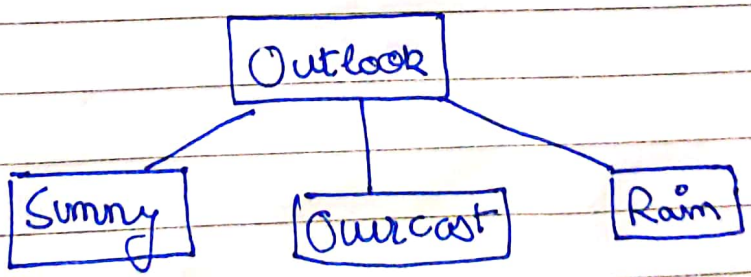
$$= 0.16$$

$$IG(\text{Wind}) = 0.94 + \left[\frac{8}{14}\left(\frac{2}{8}\log\frac{2}{8} + \frac{6}{8}\log\frac{6}{8}\right) + \frac{6}{14}\left(\frac{3}{6}\log\frac{3}{6} + \frac{3}{6}\log\frac{3}{6}\right)\right]$$

$$= 0.94 + (-0.463 + (-0.428))$$

$$= 0.05$$

We get max information gain at Outlook, so first split is at outlook



Now we find Entropy (Play Match | Sunny)

$$= -\left(\frac{3}{5}\log\frac{3}{5} + \frac{2}{5}\log\frac{2}{5}\right)$$

$$= 0.97$$

We then find corresponding IG.

$IG(\text{Temp}, \text{Sunny}) = 0.97 + \left[\frac{2}{5}\left(\frac{2}{2}\log\frac{2}{2}\right) + \frac{2}{5}\left[\frac{1}{2}\log\frac{1}{2} + \frac{1}{2}\log\frac{1}{2}\right] + \frac{1}{5}\left(\frac{1}{1}\log\frac{1}{1}\right)\right]$

$= 0.97 + (-0.4) = 0.57$

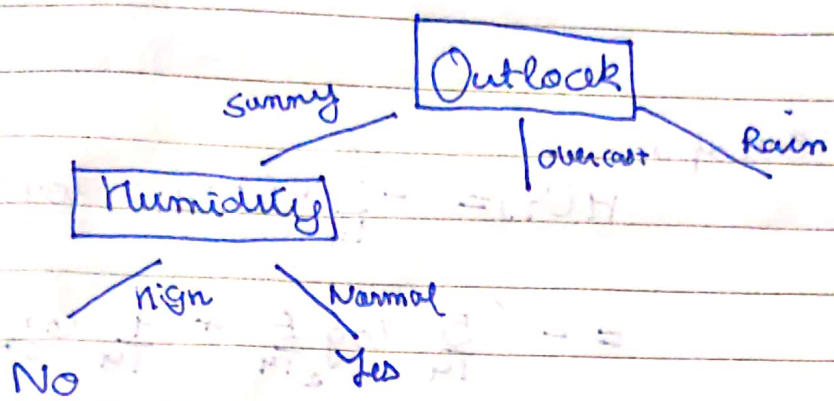
$IG(\text{Humidy}, \text{Sunny}) = 0.97 + \left[\frac{3}{5}\left(\frac{3}{3}\log\frac{3}{3}\right) + \frac{2}{5}\left(\frac{2}{2}\log\frac{2}{2}\right)\right]$

$= 0.97$

$IG(\text{Wind}, \text{Sunny}) = 0.97 + \left[\frac{3}{5}\left(\frac{3}{3}\log\frac{2}{3} + \frac{1}{3}\log\frac{1}{3}\right) + \frac{2}{5}\left[\frac{1}{2}\log\frac{1}{2} + \frac{1}{2}\log\frac{1}{2}\right]\right]$

$= 0.02$

We split at Humidity for sunny

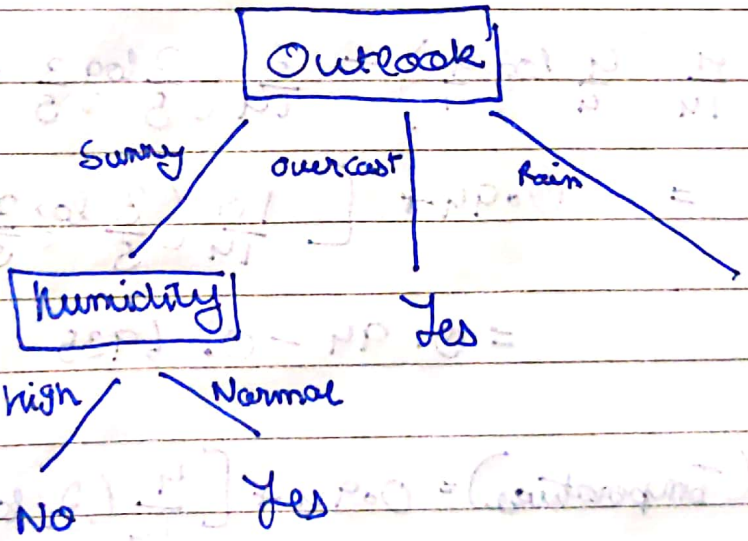


Now, for overcast,

Entropy, $H(\text{PlayMatch}, \text{Overcast})$

$= -\left(\frac{4}{4} \log \frac{4}{4}\right) = 0$

$\therefore$ No Splitting occurs



Now for rain

$H(\text{PlayMatch}, \text{Rain}) = -\left(\frac{3}{5} \log \frac{3}{5} + \frac{2}{5} \log \frac{2}{5}\right)$

$= +0.97$

Now, IG corresponding to Rain

$$IG(Temp, Rainy) = 0.97 + \left(\frac{3}{5}\left(\frac{2}{3}\log\frac{2}{3} + \frac{1}{3}\log\frac{1}{3}\right) + \frac{2}{5}\left(\log\frac{1}{2}\cdot\frac{1}{2} + \frac{1}{2}\log\frac{1}{2}\right)\right)$$

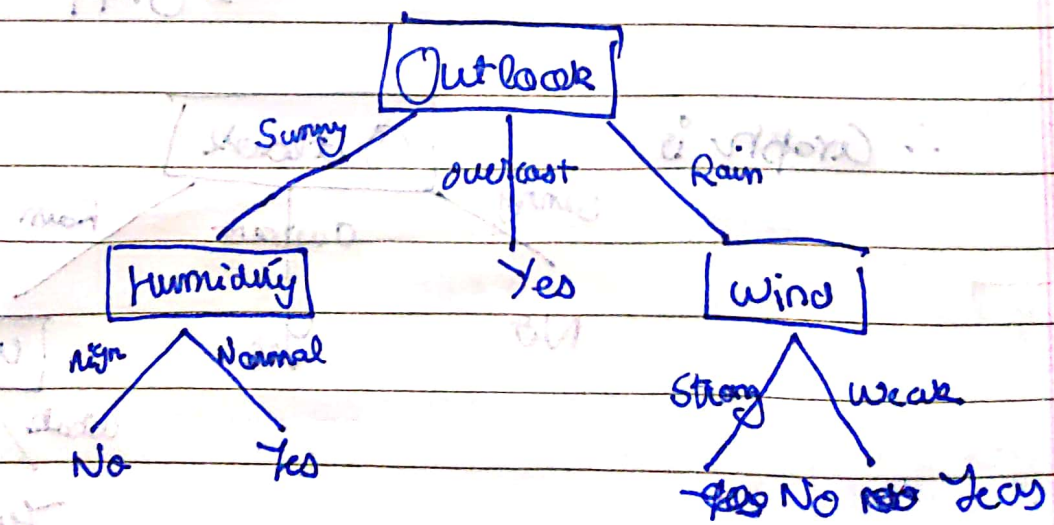
$$= 0.02$$

$$IG(Humidity, Rain) = 0.97 + \left[\frac{3}{5}\left(\frac{1}{3}\log\frac{1}{3} + \frac{2}{3}\log\frac{2}{3}\right) + \frac{2}{5}\left(\frac{1}{2}\log\frac{1}{2} + \frac{1}{2}\log\frac{1}{2}\right)\right]$$

$$= 0.02$$

$$IG(Wind, Rain) = 0.97 + \left[\frac{3}{5}\cdot\left(\frac{3}{3}\log\frac{3}{3}\right) + \frac{2}{5}\left(\frac{2}{2}\log\frac{2}{2}\right)\right] = 0.97$$

∴ Max IG for Wind, ∴ Finally we have

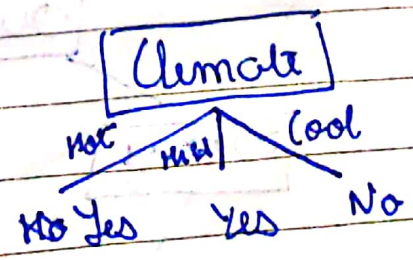


Day	Pred	Actual
D_1	No	No
D_2	No	No
D_3	Yes	Yes
D_4	Yes	Yes
D_5	Yes	Yes
D_6	No	No
D_7	Yes	Yes
D_8	No	No
D_9	Yes	Yes
D_{10}	Yes	Yes
D_{11}	Yes	Yes
D_{12}	Yes	Yes
D_{13}	Yes	Yes
	No	No

As seen from above all predicted values match hence accuracy is 100%

b) Yes, it is possible. Consider the dataset $\{D_3, D_{13}, D_4, D_{10}, D_{11}, D_{12}, D_6\}$

In such a dataset climate on its own is able to accurately predict the play match. Decision Tree would look like



c) Entropy, $H(Y) = -\left(\frac{4}{7}\log\frac{4}{7} + \frac{3}{7}\log\frac{3}{7}\right)$

$= 0.985$

$IG(\text{Outlook}) = 0.985 + \left(\frac{2}{7} \cdot \left(\frac{2}{2}\log\frac{2}{2}\right) + \frac{2}{7}\left(\frac{2}{2}\log\frac{2}{2}\right)\right.$

$\left. + \frac{3}{7}\left(\frac{2}{3}\log\frac{2}{3} + \frac{1}{3}\log\frac{1}{3}\right)\right)$

$= 0.985 - 0.39 = 0.595$

$IG(\text{Climate}) = 0.985 + \left[\frac{3}{7}\left(\frac{2}{3}\log\frac{2}{3} + \frac{1}{3}\log\frac{1}{3}\right)\right.$

$\left. + 0 + \frac{3}{7}\left(\frac{2}{3}\log\frac{2}{3} + \frac{1}{3}\log\frac{1}{3}\right)\right]$

$= 0.985 - 0.78 = 0.205$

$IG(\text{Humidity}) = 0.985 + \left[\frac{4}{7}\left(\frac{3}{4}\log\frac{3}{4} + \frac{1}{4}\log\frac{1}{4}\right) + \right.$

$\left. \frac{3}{7}\left(\frac{2}{3}\log\frac{2}{3} + \frac{1}{3}\log\frac{1}{3}\right)\right]$

$= 0.985 + [-0.39 - 0.57]$

$= 0.045$

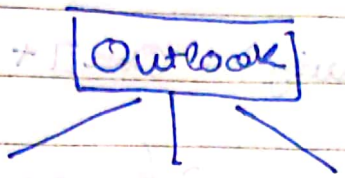
$IG(\text{Wind}) = 0.988 + \left[\frac{4}{7}\left(\frac{3}{4}\log\frac{3}{4} + \frac{1}{4}\log\frac{1}{4}\right) + \frac{3}{7}\right.$

$\left. \log\left(\frac{1}{3}\right) \cdot \frac{1}{3} + \frac{2}{3}\log\frac{2}{3}\right]$

$= 0.985 + [-0.39 + (-0.46)]$

$= 0.135$

∴ We split at Outlook



$H(Y|\text{Sunny}) = 0$

$H(Y|\text{Overcast}) = 0$

$H(Y|\text{Rain}) = -\left(\frac{2}{3}\log\frac{2}{3} + \frac{1}{3}\log\frac{1}{3}\right) = 0.91$

Finding IGs

$IG(\text{Climate}, \text{Rain}) = 0.91 + [0 + \frac{2}{3}(\log\frac{1}{2}\cdot\frac{1}{2} + \frac{1}{2}\log\frac{1}{2})]$

$= 0.91 - 0.66$

$= 0.25$

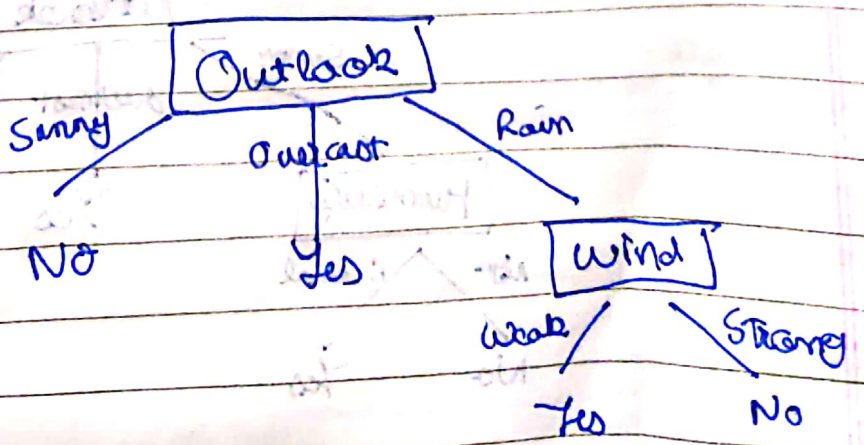
$IG(\text{Humidity}, \text{Rain}) = 0.91 + \left(\frac{2}{3}\left(\log\frac{1}{2} + \frac{1}{2}\log\frac{1}{2}\right)\right)$

$= 0.25$

$\therefore IG(\text{Wind}, \text{Rain}) = 0.91 + (0+0)$

$= 0.91$

∴ Graph is



Finding Accuracy

Day	Predicted	Actual
D_8	No	No
D_9	No	Yes
D_{10}	Yes	Yes
D_{11}	No	Yes
D_{12}	Yes	Yes
D_{13}	Yes	Yes
D_{14}	No	No

$\therefore$ Accuracy $= \frac{5}{7} \times 100 = 71.4\%$

There are a few reasons why this might be happening One could be that the data being trained on isn't diverse enough. This is evident as Sunny always predicts no, however training testing set varies and has data where sunny can take different values depending on different factors. The model is hence overfitting on the given training data leading to large errors on the diverse testing data

d) Overfitting usually occurs when a model is too complex. To make our tree simpler we want to reduce the number of leaves. We can use the following approach -

① Split data into training and testing
② Use training to build up a tree.
③ Now start from the bottom and try replacing head of a leaf node with a leaf node (This leaf node will be labelled with the majority class)
④ If error in testing decreases then we keep the leaf node else we revert back
⑤ Repeat until no longer possible.

The tree we get as a result will be much simpler and hence shouldn't overfit

Q6 In a first Order Markov Model, probability of next state only depends on the current state

$\therefore P(x_{n+1}=x_i \mid x_1=x_1, x_2=x_2 \ldots x_n=x_n) = P(x_{n+1}=x_{n+1} \mid x_n=x_n)$

Now, we need to find

$P(w_3 \mid w_1, w_2, w_4)$

From markov model's properties

$P(w_3 \mid w_1, w_2, w_4) = P(w_3 \mid w_2, w_4)$

Now using Bayes Rule for multiple variables,

$\left(P(R|H,S) = \frac{P(H|R,S)P(R|S)}{P(H|S)}\right)$, we have,

$$P(w_3 \mid w_2, w_4) = \frac{P(w_4 \mid w_3, w_2) \cdot P(w_3 \mid w_2)}{P(w_4 \mid w_2)}$$

Again from Markov models' properties,

$$= \frac{P(w_4 \mid w_3)\, P(w_3 \mid w_2)}{P(w_4 \mid w_2)}$$

Now $P(w_4 \mid w_2) = P(w_3 = \text{tough} \mid w_2) \cdot P(w_4, w_3 = \text{tough})$

$+ P(w_3 = \text{course} \mid w_2) \cdot P(w_4, w_3 = \text{course})$

$= [(0.5)\times(0.3)] + [(0.5)\times(0.5)] = 0.4$

$$\therefore P(w_3 = \text{tough} \mid w_2, w_4) = \frac{P(\text{cause} \mid \text{tough}) \cdot P(\text{tough} \mid \text{cause})}{0.4}$$

$$= \frac{0.3 \times 0.5}{0.4} = \frac{15}{40} = \frac{3}{8}$$

$$\text{and } P(w_3 = \text{cause} \mid w_2, w_4) = \frac{P(\text{cause} \mid \text{cause}) \cdot P(\text{cause} \mid \text{cause})}{0.4}$$

$$= \frac{0.25}{0.40} = \frac{25}{40} = \frac{5}{8}$$

Q-7

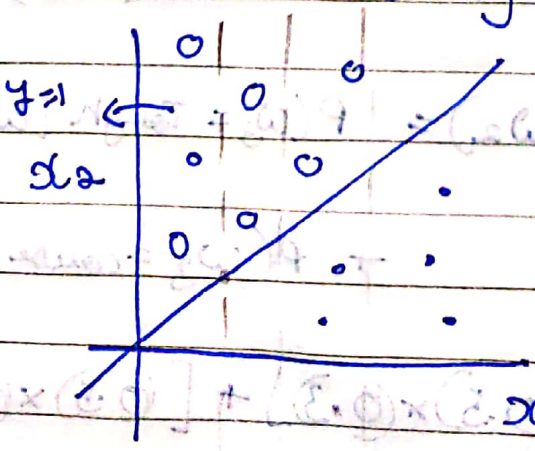
a) In the case of logistic regression we only have one decision boundary (line or plane dividing data into different classes). So for logistic regression to work properly, we'll need data to be linearly separable, whereas decision trees can make "zones" by using boundaries parallel to the axes and hence isn't bounded by this constraint.

b) Due to the above mentioned fact, decision tree also tend to overfit in case of large number of features. In these cases, logistic regression tends to perform better.

c) We have been given n 2 dimensional points which can be classified using a regression classifier into $y=1$ or $y=-1$, as

$$y = \begin{cases} 1, & w^T x + b > 0 \\ -1, & w^T x + b \leq 0 \end{cases}$$

This looks something like



Date: / /

Now, we can split based according to x_1. Now, as values are given as being linearly separable, there exists x_2 for every x_1 above which points belong to different class (x_1, x_2 lies on the original boundary). In worst case every point has unique x_1, hence n nodes to decide x_1 and one more to decide x_2. To represent n+1 nodes to check if point belongs to $y=1$ or $y=-1$ Hence we need log(n) height tree (2n nodes)

d) In case they aren't linearly separable, we can still divide using values of x_1. However, we don't have x_2 corresponding to x_1. Instead we'll have to check for x_2 over n points as well. Therefore we will have total of n^2 nodes, to represent this we need $\log(n^2)$ $= 2\log(n)$ height tree

Q-8 We are given x, vector set of n Boolean vectors, $x_1, x_2 \dots x_n$ and $y = y_k$ where $y_k = 0$ or 1 $\therefore$ Y binary as well

Now, $P(Y=1|X) = \dfrac{P(X|Y=1) \times P(Y=1)}{P(X)}$

$= \prod_{i=1}^{n} \dfrac{P(X_i|Y) \times P(Y)}{P(X_1, X_2 \dots X_n)}$

Now as Y is binary,

$P(X) = P(X \cap Y=0) + P(X \cap Y=1)$

Also, $P(X \cap Y=y_k) = P(X|Y=y_k) \cdot P(Y=y_k)$

$\therefore$ We have, $P(Y=1|X) = \prod_{i=1}^{n} \dfrac{P(X_i|Y) \times P(Y)}{P(Y=0)\,P(X|y) + P(Y=1) \cdot P(X|y)}$

$= \prod_{i=1}^{n} \dfrac{P(X_i|y) \times P(y)}{P(Y=0) \cdot \prod_{i=1}^{n} P(X_i|Y=0) + P(Y=1) \cdot \prod_{i=1}^{n} P(X_i|y=1)}$

Now, dividing numerator and divisor by $\prod_{i=1}^{n} P(X_i|y=0) \cdot P(Y=1)$

We have $\dfrac{1}{1 + \dfrac{P(Y=0) \cdot \prod_{i=1}^{n} P(X_i|Y=0)}{P(Y=1) \cdot \prod_{i=1}^{n} P(X_i|Y=1)}}$

$= \dfrac{1}{1 + e^{\left(\ln\left(\frac{P(Y=0) \prod_{i=1}^{n} P(X_i|y=0)}{P(Y=1) \prod_{i=1}^{n} P(X_i|y=1)}\right)\right.}}$

$$= \frac{1}{1+e^{\left[\ln\left(\frac{P(Y=0)}{P(Y=1)}\right) + \sum_{i=1}^{n} \ln\left(\frac{P(x_i|Y=0)}{P(x_i|Y=1)}\right)\right]}}$$

Now, $P(Y=1) = \Pi \quad \therefore P(Y=0) = 1-\Pi$

~~Also $P(X_i|Y=0$ $P(x_i=a)|Y=b$~~

Also, $P(X_i = J \mid Y=k) = \theta_{ik}^{J} \cdot (1-\theta_{ik})^{1-J}$

from given defn in hint

$\therefore$ We have

$$P(Y=1 \mid X) = \frac{1}{1+e^{\left[\ln\left(\frac{1-\Pi}{\Pi}\right) + \sum_{i=1}^{n} \ln\left(\frac{\theta_{i0}^{x_i}(1-\theta_{i0})^{1-x_i}}{\theta_{i1}^{x_i}(1-\theta_{i1})^{1-x_i}}\right)\right]}}$$

$$= \frac{1}{1+e^{\left(\ln\left(\frac{1-\Pi}{\Pi}\right) + \sum_{i=1}^{n} \ln\left(\frac{\theta_{i0}^{x_i}}{\theta_{i1}^{x_i}}\right) + \sum_{i=1}^{n} \ln\left(\frac{1-\theta_{i0}}{1-\theta_{i1}}\right)^{1-x_i}\right)}}$$

$$= \frac{1}{1+e^{\left[\ln\left(\frac{1-\Pi}{\Pi}\right) + \sum_{i=1}^{n} x_i \ln\left(\frac{\theta_{i0}}{\theta_{i1}}\right) + \sum_{i=1}^{n} (1-x_i) \ln\left(\frac{1-\theta_{i0}}{1-\theta_{i1}}\right)\right]}}$$

$$= \frac{1}{1+e^{\left[\ln\left(\frac{1-\Pi}{\Pi}\right) + \sum_{i=1}^{n} x_i \ln\left(\frac{\theta_{i0}(1-\theta_{i1})}{\theta_{i1}(1-\theta_{i0})}\right) + \sum_{i=1}^{n} \ln\left(\frac{1-\theta_{i0}}{1-\theta_{i1}}\right)\right]}}$$

Taking $w_0 = \ln\left(\frac{1-\Pi}{\Pi}\right) + \sum_{i=1}^{n} \ln\left(\frac{1-\theta_{i0}}{1-\theta_{i1}}\right)$

$$w_i = \ln\left(\frac{\theta_{i0}(1-\theta_{i1})}{\theta_{i1}(1-\theta_{i0})}\right)$$

$$\therefore P(Y=1|X) = \frac{1}{1+\exp(\omega_0 + \sum_{i=1}^{\hat{} } \omega_i x_i)}$$

Hence we have shown what was required