

Assignment 2

- 1)
- a) Principal component analysis(PCA) refers to an unsupervised statistical procedure that converts a set of correlated features into a set of uncorrelated features. It is a dimensionality reduction algorithm; it includes projecting data on a lower-dimensional subspace, also known as principal subspace, by maximizing the variance of the projected data. This is done using the concepts of linear algebra such as eigenvectors, SVD, matrix decomposition.
- Steps are as follows:
1. Standardisation of data
 2. Generation of correlation matrix for reduced dimensions
 3. Calculation of principal components and variance
 4. Arrange principal components into decreasing order and select the desired number.
- b) Square vector decomposition(SVD), as the name suggests it decomposes any matrix into a multiplication of square vectors consisting of a diagonal matrix and an orthogonal matrix comprised an orthogonal matrix consisting of orthonormal eigenvectors and its inverse. I.e., and matrix $A(m \times n)$ can be represented as follows:

$$A = USV^T$$

Where,

U is an $m \times m$ matrix.

S is an $m \times n$ rectangular diagonal matrix of Singular Values arranged in decreasing order.

V is an $n \times n$ matrix.

By doing this, we can write matrix A as a linear combination of low-rank matrices, and we can then only choose those columns which have larger singular values, and the rest all can be ignored. By combining these convert U, S, and V, we can obtain a new value of Matrix A representing the same information but with fewer features.

- c) t-distributed stochastic neighbor embedding (t-SNE) is also a dimensionality reduction algorithm. It does so by assigning probabilities based on the similarity of features, i.e., high similarity gets high probability, and low similarity gets low probability. It helps in visualizing large dimensional datasets into two dimensions. It preserves the small pairwise similarities in order to preserve the structure of a non linear data. It helps in visualizing a dataset with large number of features by plotting into 2 or 3 dimensional space
- d) Stratified sampling refers to dividing data into groups(strata) with similar features and then randomly picking data from those groups. This method ensures that we are not training our model more for some labels and less for others.
- Stratified sampling of Dataset A gave the following Results.

Training Data											
Label	0	1	2	3	4	5	6	7	8	9	Total
Frequency	320	395	314	339	333	318	353	345	328	315	3360
Percentage	9.52	11.75	9.34	10.08	9.91	9.46	10.50	10.26	9.76	9.37	100

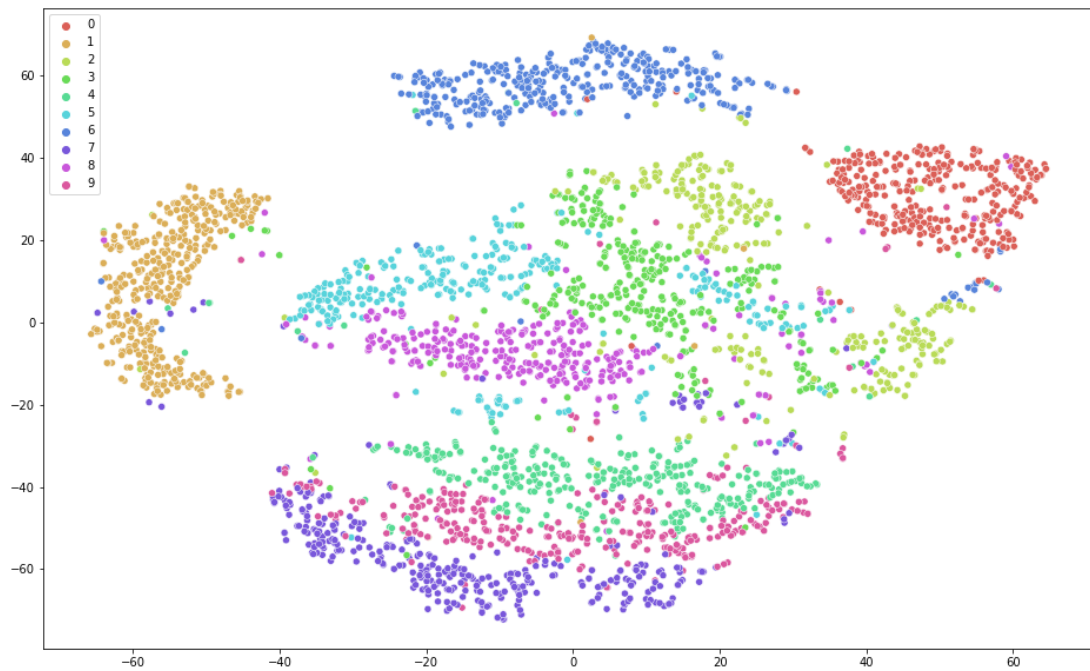
Testing Data											
Label	0	1	2	3	4	5	6	7	8	9	Total
Frequency	80	99	79	85	83	80	88	86	82	78	840
Percentage	9.52	11.78	9.40	10.11	9.88	9.52	10.47	10.23	9.76	9.28	100

It is evident from the percentage of frequencies that.

- All labels have almost the same frequency in both datasets.
- Both training and testing data have the same percentage of frequency for every label.

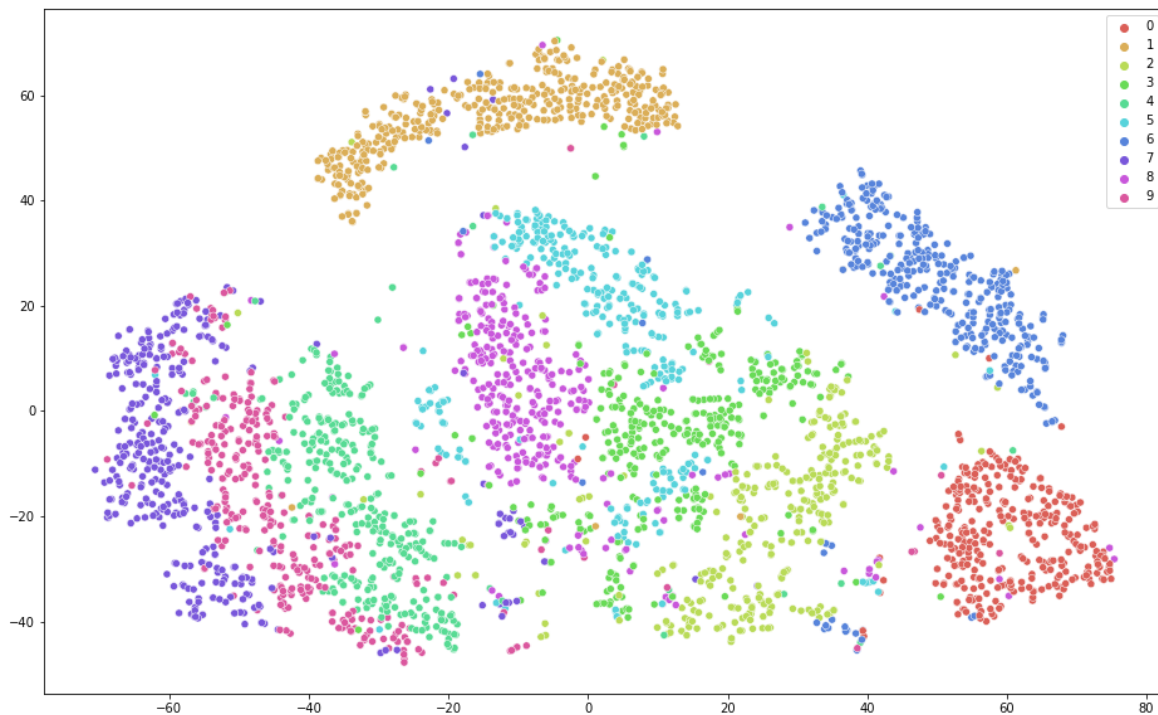
- e) Reported accuracy using SKlearn's logistic regression on dimensionally reduced dataset A by using SKlearn's PCA algorithm with 50 components: 87.7%

```
[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 3360 samples in 0.023s...
[t-SNE] Computed neighbors for 3360 samples in 1.222s...
[t-SNE] Computed conditional probabilities for sample 1000 / 3360
[t-SNE] Computed conditional probabilities for sample 2000 / 3360
[t-SNE] Computed conditional probabilities for sample 3000 / 3360
[t-SNE] Computed conditional probabilities for sample 3360 / 3360
[t-SNE] Mean sigma: 4.431989
[t-SNE] Computed conditional probabilities in 0.218s
[t-SNE] Iteration 50: error = 84.1780396, gradient norm = 0.0006523 (50 iterations in 1.884s)
[t-SNE] Iteration 100: error = 79.1078796, gradient norm = 0.0046838 (50 iterations in 1.695s)
[t-SNE] Iteration 150: error = 78.7954102, gradient norm = 0.0005034 (50 iterations in 1.307s)
[t-SNE] Iteration 200: error = 78.7879791, gradient norm = 0.0001236 (50 iterations in 1.296s)
[t-SNE] Iteration 250: error = 78.7872009, gradient norm = 0.0000741 (50 iterations in 1.311s)
[t-SNE] KL divergence after 250 iterations with early exaggeration: 78.787201
[t-SNE] Iteration 300: error = 1.8057337, gradient norm = 0.0010719 (50 iterations in 1.201s)
[t-SNE] Iteration 350: error = 1.5355535, gradient norm = 0.0004085 (50 iterations in 1.206s)
[t-SNE] Iteration 400: error = 1.4307141, gradient norm = 0.0002335 (50 iterations in 1.213s)
[t-SNE] Iteration 450: error = 1.3783107, gradient norm = 0.0001652 (50 iterations in 1.225s)
[t-SNE] Iteration 500: error = 1.3488255, gradient norm = 0.0001288 (50 iterations in 1.246s)
[t-SNE] Iteration 550: error = 1.3309402, gradient norm = 0.0001103 (50 iterations in 1.231s)
[t-SNE] Iteration 600: error = 1.3196228, gradient norm = 0.0000940 (50 iterations in 1.238s)
[t-SNE] Iteration 650: error = 1.3118364, gradient norm = 0.0000854 (50 iterations in 1.249s)
[t-SNE] Iteration 700: error = 1.3059659, gradient norm = 0.0000736 (50 iterations in 1.234s)
[t-SNE] Iteration 750: error = 1.3010006, gradient norm = 0.0000671 (50 iterations in 1.238s)
[t-SNE] Iteration 800: error = 1.2970519, gradient norm = 0.0000678 (50 iterations in 1.231s)
[t-SNE] Iteration 850: error = 1.2936760, gradient norm = 0.0000705 (50 iterations in 1.222s)
[t-SNE] Iteration 900: error = 1.2906569, gradient norm = 0.0000667 (50 iterations in 1.229s)
[t-SNE] Iteration 950: error = 1.2880832, gradient norm = 0.0000567 (50 iterations in 1.220s)
[t-SNE] Iteration 1000: error = 1.2858400, gradient norm = 0.0000551 (50 iterations in 1.239s)
[t-SNE] KL divergence after 1000 iterations: 1.285840
```



- f) Reported accuracy using SKlearn's logistic regression on dimensionally reduced dataset A by using SKlearn's SVD algorithm with 50 components: 87.97%

```
[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 3360 samples in 0.021s...
[t-SNE] Computed neighbors for 3360 samples in 1.217s...
[t-SNE] Computed conditional probabilities for sample 1000 / 3360
[t-SNE] Computed conditional probabilities for sample 2000 / 3360
[t-SNE] Computed conditional probabilities for sample 3000 / 3360
[t-SNE] Computed conditional probabilities for sample 3360 / 3360
[t-SNE] Mean sigma: 4.482303
[t-SNE] Computed conditional probabilities in 0.219s
[t-SNE] Iteration 50: error = 84.0945587, gradient norm = 0.0050289 (50 iterations in 1.703s)
[t-SNE] Iteration 100: error = 78.9681015, gradient norm = 0.0029738 (50 iterations in 1.590s)
[t-SNE] Iteration 150: error = 78.8233185, gradient norm = 0.0004196 (50 iterations in 1.428s)
[t-SNE] Iteration 200: error = 78.8177338, gradient norm = 0.0001039 (50 iterations in 1.480s)
[t-SNE] Iteration 250: error = 78.8168640, gradient norm = 0.0000554 (50 iterations in 1.390s)
[t-SNE] KL divergence after 250 iterations with early exaggeration: 78.816864
[t-SNE] Iteration 300: error = 1.8086151, gradient norm = 0.0010522 (50 iterations in 1.215s)
[t-SNE] Iteration 350: error = 1.5403881, gradient norm = 0.0004114 (50 iterations in 1.214s)
[t-SNE] Iteration 400: error = 1.4372334, gradient norm = 0.0002306 (50 iterations in 1.233s)
[t-SNE] Iteration 450: error = 1.3855076, gradient norm = 0.0001610 (50 iterations in 1.235s)
[t-SNE] Iteration 500: error = 1.3544227, gradient norm = 0.0001362 (50 iterations in 1.241s)
[t-SNE] Iteration 550: error = 1.3367876, gradient norm = 0.0001176 (50 iterations in 1.248s)
[t-SNE] Iteration 600: error = 1.3259468, gradient norm = 0.0001007 (50 iterations in 1.327s)
[t-SNE] Iteration 650: error = 1.3178842, gradient norm = 0.0000895 (50 iterations in 1.344s)
[t-SNE] Iteration 700: error = 1.3114910, gradient norm = 0.0000797 (50 iterations in 1.273s)
[t-SNE] Iteration 750: error = 1.3061848, gradient norm = 0.0000751 (50 iterations in 1.275s)
[t-SNE] Iteration 800: error = 1.3017068, gradient norm = 0.0000695 (50 iterations in 1.294s)
[t-SNE] Iteration 850: error = 1.2981895, gradient norm = 0.0000598 (50 iterations in 1.313s)
[t-SNE] Iteration 900: error = 1.2951746, gradient norm = 0.0000541 (50 iterations in 1.298s)
[t-SNE] Iteration 950: error = 1.2924190, gradient norm = 0.0000514 (50 iterations in 1.308s)
[t-SNE] Iteration 1000: error = 1.2900641, gradient norm = 0.0000523 (50 iterations in 1.293s)
[t-SNE] KL divergence after 1000 iterations: 1.290064
```



g) Accuracy Reported for SVD is 0.888095 whereas Accuracy Reported is 0.886904 . This shows us that the SVD is performing slightly better when compared with PCA but it doesn't have much significance.

PCA actually uses SVD for its calculation. PCA actually keeps the significant terms, and drops all the less significant terms. This is a method used in PCA for dimension reduction. Whereas in SVD we use algebraic expressions by which we receive diagonalized matrices of the singular values. This method is used for reducing the dimension of datasets.

2) In preprocessing, the height and weight columns were interchanged as values in both the columns were not matching to any real-world example.

a) Running SKlearn's linear regression on Dataset C with 100 bootstrap samples of size 9000 each and tested on testing data of size remaining 1000 data, resulted in the following data.

Bias	1.216428160813979
Variance	0.0004516430705838671
MSE	2.3173975161759146

b) Value of $MSE - Bias^2 - Variance$

$$= 2.3173975161759146 - 1.4796974704212797 - 0.0004516430705838671$$

$$= 0.837248402684051$$

The above value represents the noise in the dataset, i.e., more the above value, the more is the noise in the dataset

3)

a)

- For Dataset A:

Test Accuracy Achieved: 74.5 %

Optimal depth: 9

```
{'max_depth': 2} Validation Accuracy: 0.32976190476190476 Training Accuracy: 0.35253968253968254
{'max_depth': 3} Validation Accuracy: 0.4269047619047619 Training Accuracy: 0.4583333333333333
{'max_depth': 4} Validation Accuracy: 0.5307142857142857 Training Accuracy: 0.5894444444444444
{'max_depth': 5} Validation Accuracy: 0.6173809523809524 Training Accuracy: 0.7013492063492064
{'max_depth': 6} Validation Accuracy: 0.678095238095238 Training Accuracy: 0.780952380952381
{'max_depth': 7} Validation Accuracy: 0.7083333333333333 Training Accuracy: 0.8492063492063492
{'max_depth': 8} Validation Accuracy: 0.7126190476190477 Training Accuracy: 0.9061904761904762
{'max_depth': 9} Validation Accuracy: 0.7178571428571429 Training Accuracy: 0.9438888888888888
{'max_depth': 10} Validation Accuracy: 0.7157142857142857 Training Accuracy: 0.9709523809523809
{'max_depth': 11} Validation Accuracy: 0.7147619047619047 Training Accuracy: 0.9857936507936508
{'max_depth': 12} Validation Accuracy: 0.7126190476190477 Training Accuracy: 0.9934920634920635
{'max_depth': 13} Validation Accuracy: 0.7119047619047619 Training Accuracy: 0.9973809523809525
{'max_depth': 14} Validation Accuracy: 0.7121428571428571 Training Accuracy: 0.9987301587301587
{'max_depth': 15} Validation Accuracy: 0.7116666666666667 Training Accuracy: 0.9993650793650793
{'max_depth': 16} Validation Accuracy: 0.7138095238095239 Training Accuracy: 0.9996825396825397
{'max_depth': 17} Validation Accuracy: 0.7152380952380952 Training Accuracy: 0.9997619047619046
{'max_depth': 18} Validation Accuracy: 0.7085714285714286 Training Accuracy: 1.0
{'max_depth': 19} Validation Accuracy: 0.7147619047619047 Training Accuracy: 0.999920634920635
```

- For Dataset B:

Test Accuracy Achieved: 55.83 %

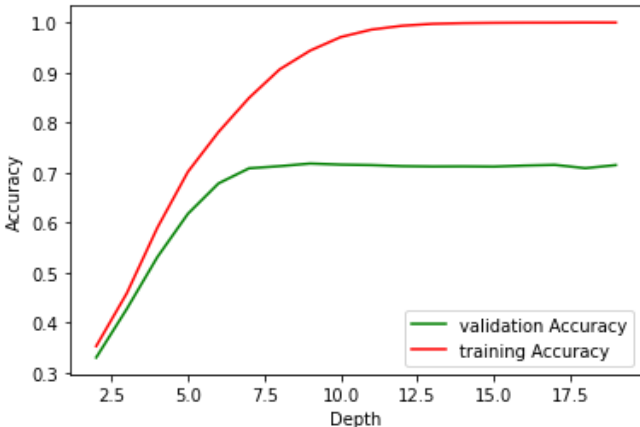
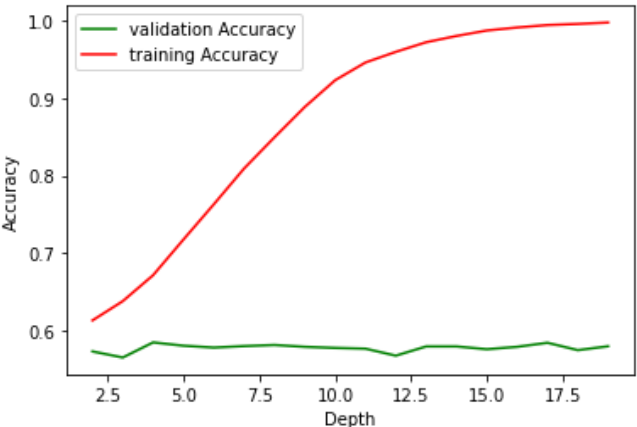
Optimal depth: 4

```

{'max_depth': 2} Validation Accuracy: 0.5728571428571428 Training Accuracy: 0.612936507936508
{'max_depth': 3} Validation Accuracy: 0.5650000000000001 Training Accuracy: 0.6378571428571428
{'max_depth': 4} Validation Accuracy: 0.5845238095238094 Training Accuracy: 0.6714285714285715
{'max_depth': 5} Validation Accuracy: 0.5802380952380952 Training Accuracy: 0.7175396825396826
{'max_depth': 6} Validation Accuracy: 0.5778571428571428 Training Accuracy: 0.7630158730158729
{'max_depth': 7} Validation Accuracy: 0.5797619047619048 Training Accuracy: 0.8093650793650793
{'max_depth': 8} Validation Accuracy: 0.5811904761904761 Training Accuracy: 0.8496031746031745
{'max_depth': 9} Validation Accuracy: 0.5788095238095238 Training Accuracy: 0.8888888888888889
{'max_depth': 10} Validation Accuracy: 0.5773809523809523 Training Accuracy: 0.9234126984126985
{'max_depth': 11} Validation Accuracy: 0.5764285714285715 Training Accuracy: 0.9463492063492064
{'max_depth': 12} Validation Accuracy: 0.5673809523809524 Training Accuracy: 0.9600793650793651
{'max_depth': 13} Validation Accuracy: 0.5792857142857143 Training Accuracy: 0.9725396825396825
{'max_depth': 14} Validation Accuracy: 0.5792857142857144 Training Accuracy: 0.9806349206349207
{'max_depth': 15} Validation Accuracy: 0.5757142857142858 Training Accuracy: 0.9876984126984126
{'max_depth': 16} Validation Accuracy: 0.5788095238095239 Training Accuracy: 0.9915873015873016
{'max_depth': 17} Validation Accuracy: 0.5840476190476191 Training Accuracy: 0.9947619047619047
{'max_depth': 18} Validation Accuracy: 0.5745238095238097 Training Accuracy: 0.9961904761904762
{'max_depth': 19} Validation Accuracy: 0.5795238095238096 Training Accuracy: 0.9980158730158731

```

b)

Dataset A		Dataset B	
Decision Tree			
			
Max Training Accuracy	Max Validation Accuracy	Max Training Accuracy	Max Validation Accuracy
99.99	72.07	99.80	58.38
Using Gaussian Naive Bayes of sklearn and my implementation of k fold with k=5			
Max Training Accuracy	Max Validation Accuracy	Max Training Accuracy	Max Validation Accuracy
60.79	56.19	58.25	60.11

c)

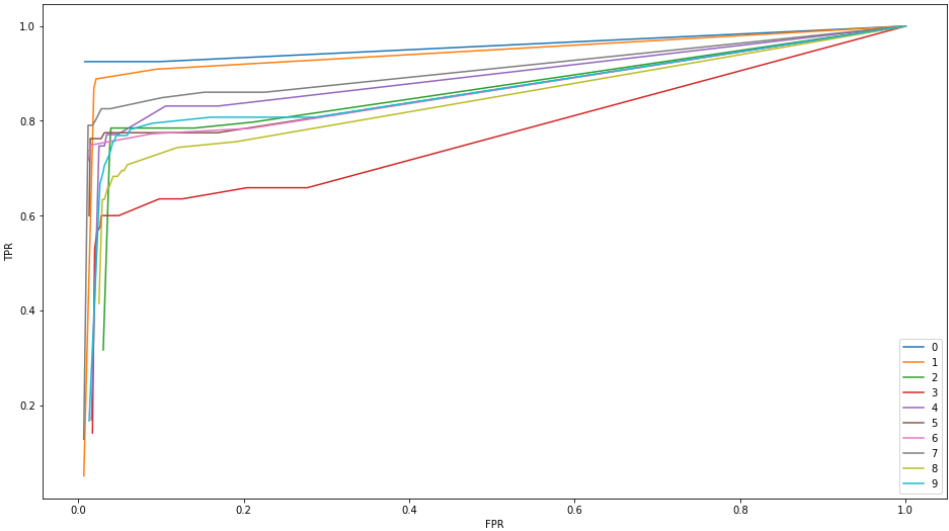
```
1  preprocessor = MyPreProcessor()
2  X, y = preprocessor.pre_process(0)
3  X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, test_size=0.20, random_state=40)
4  dt= DecisionTreeClassifier()
5  params = {'max_depth': list(range(2, 20))}
6  grid_search_cv = GridSearch(dt,params)
7
8  grid_search_cv.fit(X_train, y_train)
9
10 model=pickle.load(open('model','rb'))
11 model.score(X_test,y_test)
```

d)

- Dataset A:
 - Decision Tree

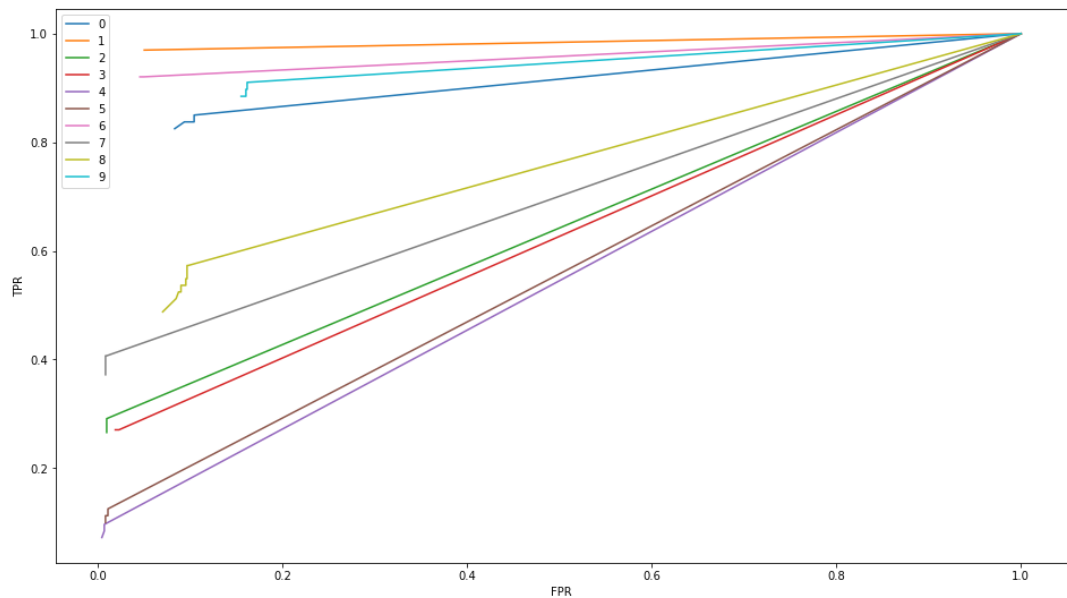
Confusion Matrix									
75	0	0	1	2	0	0	0	2	0
0	87	5	3	2	0	1	1	0	0
1	3	57	6	2	1	3	1	3	2
0	4	12	45	2	3	1	3	12	3
0	4	3	2	62	0	0	3	2	7
1	1	2	9	0	57	3	3	4	0
5	1	3	0	3	4	65	1	4	2
0	3	1	1	3	1	0	71	1	5
1	3	2	5	3	1	2	3	52	10
0	0	2	4	7	1	0	5	3	56

Precision	Recall	F1-score		
0.903614	0.9375	0.920245		
0.820755	0.878788	0.84878		
0.655172	0.721519	0.686747		
0.592105	0.529412	0.559006		
0.72093	0.746988	0.733728		
0.838235	0.7125	0.77027		
0.866667	0.738636	0.797546		
0.78022	0.825581	0.80226		
0.626506	0.634146	0.630303		
0.658824	0.717949	0.687117		
Accuracy/Micro-average:		0.746429		
Macro-Average:		0.746303	0.744302	0.7436



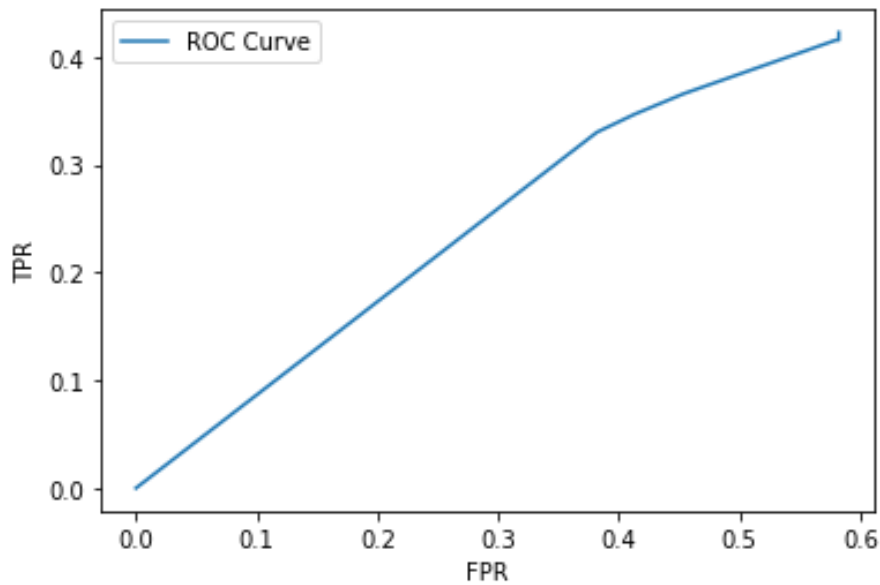
■ Gaussian Naive Bayes

Confusion Matrix										Precision	Recall	F1-score
67	0	3	1	0	2	6	0	0	1	0.471831	0.8375	0.603604
0	96	0	1	0	0	1	0	1	0	0.716418	0.969697	0.824034
13	1	23	11	1	4	13	0	12	1	0.766667	0.291139	0.422018
13	9	0	23	0	1	8	1	17	13	0.589744	0.270588	0.370968
10	4	1	1	7	0	7	1	14	38	0.583333	0.0843373	0.147368
26	4	1	2	2	10	3	0	26	6	0.555556	0.125	0.204082
1	4	2	0	0	0	81	0	0	0	0.680672	0.920455	0.782609
1	3	0	0	1	0	0	35	0	46	0.853659	0.406977	0.551181
6	12	0	0	1	1	0	1	44	17	0.385965	0.536585	0.44898
5	1	0	0	0	0	0	3	0	69	0.361257	0.884615	0.513011
										Accuracy/Micro-average:		
										Macro-Average:		
										0.541667	0.532689	0.486785



- Dataset B:
 - Decision Tree

Confusion Matrix		Precision	Recall	F1-score
244	175	0.584323	0.584323	0.584323
175	246			
		Accuracy: 0.583333		



■ Gaussian Naive Bayes

Confusion Matrix

```
---  ---
259  160
223  198
---  ---
```

Precision

Recall

F1-score

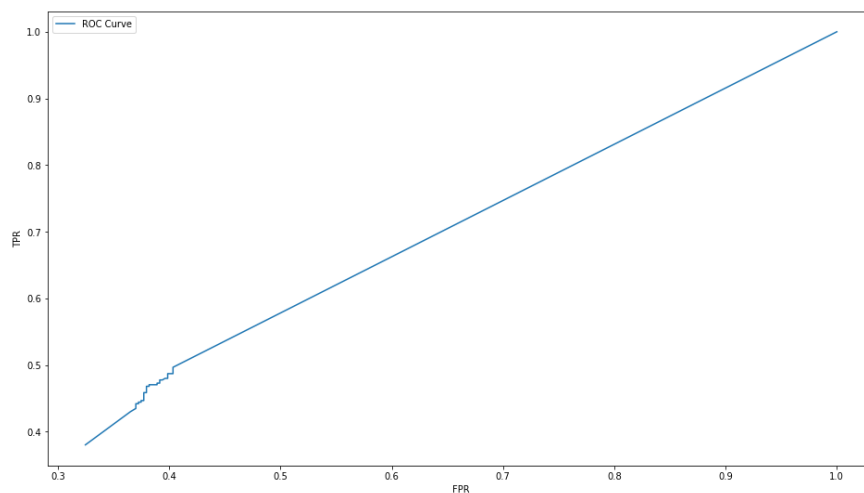
0.553073

0.470309

0.508344

Accuracy:

0.544048



4) The implementation of Gaussian Naive Bayes is attached to code files.

The following table shows the accuracy obtained on the testing set using both implementations on both datasets.

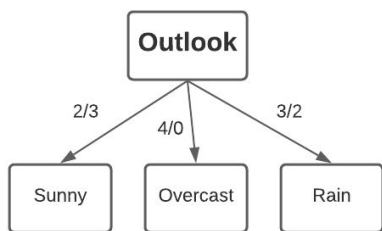
Dataset A	
Custom Implementation	72.38%
Sklearn's Implementation	55.24%
Dataset B	
Custom Implementation	54.05%
Sklearn's Implementation	54.05%

The custom naive bayes obtained a higher accuracy than the sklearn's implementation because the dataset has a lot of columns having very low(10^{-7}) mean and variance, and since these values were creating underflow in python thats why we have to ignore all those columns. Doing this increased the accuracy of our implementation greatly.

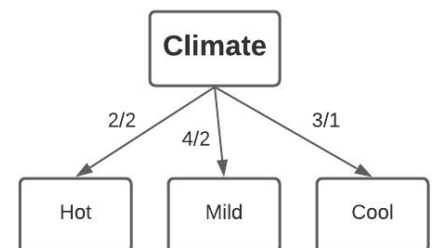
5)

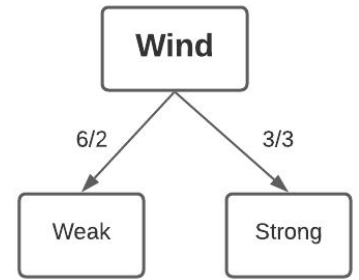
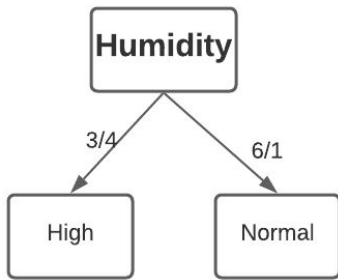
a) Given features: Outlook, Climate, Humidity, Wind

- Dataset Description:

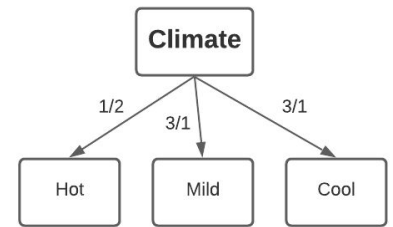
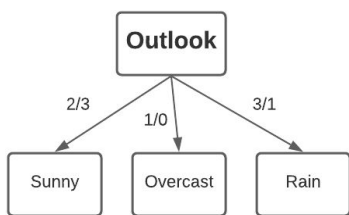


	Outlook	Climate	Humidity	Wind	Play
0	Sunny	Hot	High	Weak	No
1	Sunny	Hot	High	Strong	No
2	Overcast	Hot	High	Weak	Yes
3	Rain	Mild	High	Weak	Yes
4	Rain	Cool	Normal	Weak	Yes
5	Rain	Cool	Normal	Strong	No
6	Overcast	Cool	Normal	Strong	Yes
7	Sunny	Mild	High	Weak	No
8	Sunny	Cool	Normal	Weak	Yes
9	Rain	Mild	Normal	Weak	Yes
10	Sunny	Mild	Normal	Strong	Yes
11	Overcast	Mild	High	Strong	Yes
12	Overcast	Hot	Normal	Weak	Yes
13	Rain	Mild	High	Strong	No

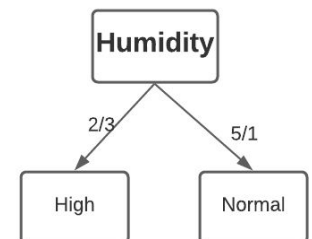
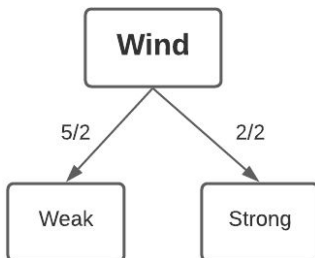




- Training Dataset:(D1-D11)



	Outlook	Climate	Humidity	Wind	Play
0	Sunny	Hot	High	Weak	No
1	Sunny	Hot	High	Strong	No
2	Overcast	Hot	High	Weak	Yes
3	Rain	Mild	High	Weak	Yes
4	Rain	Cool	Normal	Weak	Yes
5	Rain	Cool	Normal	Strong	No
6	Overcast	Cool	Normal	Strong	Yes
7	Sunny	Mild	High	Weak	No
8	Sunny	Cool	Normal	Weak	Yes
9	Rain	Mild	Normal	Weak	Yes
10	Sunny	Mild	Normal	Strong	Yes



- Testing Dataset:(D12-D14)

	Outlook	Climate	Humidity	Wind	Play
11	Overcast	Mild	High	Strong	Yes
12	Overcast	Hot	Normal	Weak	Yes
13	Rain	Mild	High	Strong	No

- **For Finding The root node:**

Gini Impurities of each feature:

→ Outlook:

$$\text{Sunny} = 1 - \left(\frac{2}{5}\right)^2 - \left(\frac{3}{5}\right)^2 = 0.48$$

$$\text{Overcast} = 1 - \left(\frac{2}{2}\right)^2 - \left(\frac{0}{2}\right)^2 = 0$$

$$\text{Rain} = 1 - \left(\frac{3}{4}\right)^2 - \left(\frac{1}{4}\right)^2 = 0.375$$

$$\text{Total} = \frac{5}{11}(\text{Sunny}) + \frac{2}{11}(\text{Overcast}) + \frac{4}{11}(\text{Rain}) \\ = 0.354$$

→ Climate:

$$\text{Hot} = 1 - \left(\frac{1}{3}\right)^2 - \left(\frac{2}{3}\right)^2 = 0.45$$

$$\text{Mild} = 1 - \left(\frac{3}{4}\right)^2 - \left(\frac{1}{4}\right)^2 = 0.375$$

$$\text{Cool} = 1 - \left(\frac{3}{4}\right)^2 - \left(\frac{1}{4}\right)^2 = 0.375$$

$$\text{Total} = \frac{3}{11}(\text{Hot}) + \frac{4}{11}(\text{Mild}) + \frac{4}{11}(\text{Cool}) \\ = 0.395$$

→ Wind:

$$\text{Weak} = 1 - \left(\frac{5}{7}\right)^2 - \left(\frac{2}{7}\right)^2 = 0.41$$

$$\text{Strong} = 1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2 = 0.5$$

$$\text{Total} = \frac{7}{11}(\text{Weak}) + \frac{4}{11}(\text{Strong}) \\ = 0.442$$

→ Humidity:

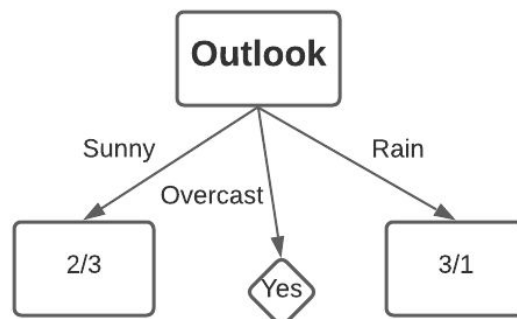
$$\text{Normal} = 1 - \left(\frac{5}{6}\right)^2 - \left(\frac{1}{6}\right)^2 = 0.28$$

$$\text{High} = 1 - \left(\frac{2}{5}\right)^2 - \left(\frac{3}{5}\right)^2 = 0.48$$

$$\text{Total} = \frac{6}{11}(\text{Normal}) + \frac{5}{11}(\text{High}) \\ = 0.371$$

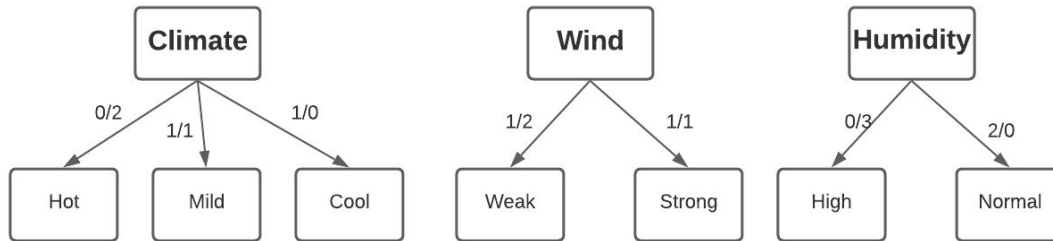
Since Outlook has the lowest GINI impurity value, that means it will be our root node of the Decision Tree.

Current Tree:



- **Finding the next node in Sunny Branch:**

	Outlook	Climate	Humidity	Wind	Play
0	Sunny	Hot	High	Weak	No
1	Sunny	Hot	High	Strong	No



Gini impurities of remaining features:

→ Climate:

$$\text{Hot} = 1 - \left(\frac{0}{2}\right)^2 - \left(\frac{2}{2}\right)^2 = 0$$

$$\text{Mild} = 1 - \left(\frac{1}{2}\right)^2 - \left(\frac{1}{2}\right)^2 = 0.5$$

$$\text{Cool} = 1 - \left(\frac{1}{1}\right)^2 - \left(\frac{0}{1}\right)^2 = 0$$

$$\text{Total} = \frac{2}{5}(\text{Hot}) + \frac{2}{5}(\text{Mild}) + \frac{1}{5}(\text{Cool}) = 0.2$$

→ Wind:

$$\text{Weak} = 1 - \left(\frac{1}{3}\right)^2 - \left(\frac{2}{3}\right)^2 = 0.45$$

$$\text{Strong} = 1 - \left(\frac{1}{2}\right)^2 - \left(\frac{1}{2}\right)^2 = 0.5$$

$$\text{Total} = \frac{3}{5}(\text{Weak}) + \frac{2}{5}(\text{Strong}) = 0.47^1$$

→ Humidity:

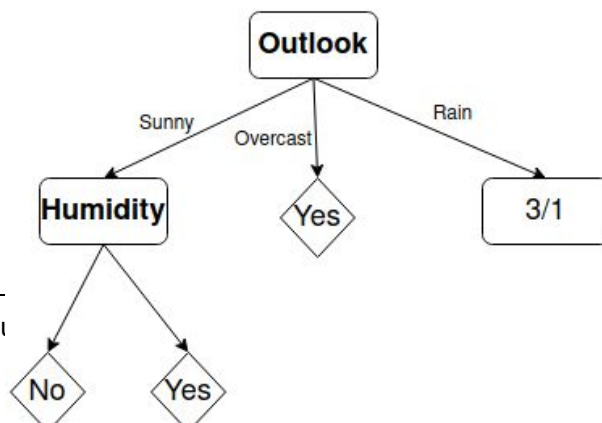
$$\text{Normal} = 1 - \left(\frac{0}{3}\right)^2 - \left(\frac{3}{3}\right)^2 = 0$$

$$\text{High} = 1 - \left(\frac{2}{2}\right)^2 - \left(\frac{0}{2}\right)^2 = 0$$

$$\text{Total} = \frac{3}{5}(\text{Normal}) + \frac{2}{5}(\text{High}) = 0$$

Since Humidity has the lowest GINI impurity then it will be our next node after

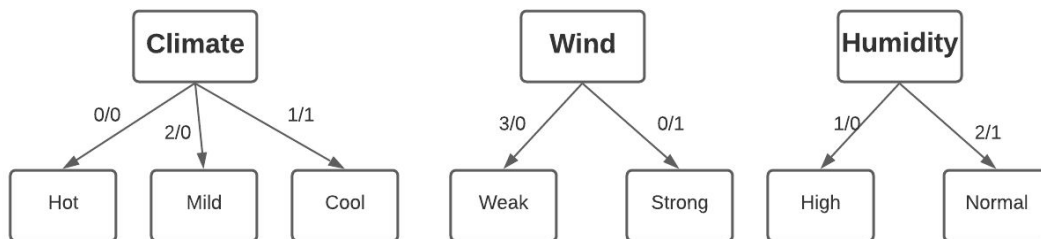
Current Tree:



¹ (a/b) pairs represent the frequency of 'Yes' and 'No' outcomes.

- Finding the next node in Rain Branch:

	Outlook	Climate	Humidity	Wind	Play
3	Rain	Mild	High	Weak	Yes
4	Rain	Cool	Normal	Weak	Yes
5	Rain	Cool	Normal	Strong	No
9	Rain	Mild	Normal	Weak	Yes



Gini impurities of remaining features:

→ Climate:

Hot => N. A

$$\text{Mild} = 1 - \left(\frac{2}{2}\right)^2 - \left(\frac{0}{2}\right)^2 = 0$$

$$\text{Cool} = 1 - \left(\frac{1}{2}\right)^2 - \left(\frac{1}{2}\right)^2 = 0.5^2$$

$$\begin{aligned} \text{Total} &= \frac{2}{4}(\text{Mild}) + \frac{2}{4}(\text{Cool}) \\ &= 0.25 \end{aligned}$$

→ Wind:

$$\text{Weak} = 1 - \left(\frac{3}{3}\right)^2 - \left(\frac{0}{3}\right)^2 = 0$$

$$\text{Strong} = 1 - \left(\frac{0}{1}\right)^2 - \left(\frac{1}{1}\right)^2 = 0$$

$$\begin{aligned} \text{Total} &= \frac{3}{4}(\text{Weak}) + \frac{1}{4}(\text{Strong}) \\ &= 0 \end{aligned}$$

→ Humidity:

$$\text{Normal} = 1 - \left(\frac{2}{3}\right)^2 - \left(\frac{1}{3}\right)^2 = 0.4$$

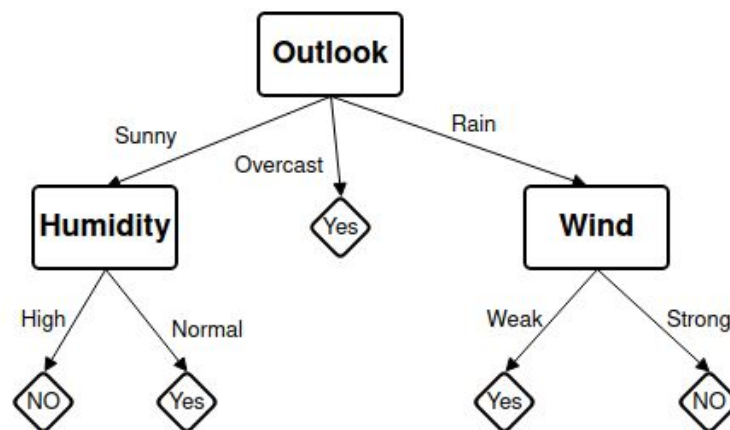
$$\text{High} = 1 - \left(\frac{1}{1}\right)^2 - \left(\frac{0}{1}\right)^2 = 0$$

$$\begin{aligned} \text{Total} &= \frac{3}{4}(\text{Normal}) + \frac{1}{4}(\text{High}) \\ &= 0.33 \end{aligned}$$

Since Wind has the lowest GINI impurity then it will be our next node after

² (a/b) pairs represent the frequency of that feature that gives output variable as (yes/no)

Final Tree:



- **Finding the accuracy**

Row #	Original value	Decision tree prediction
1	Yes	Yes
2	Yes	Yes
3	No	No

$$\text{Accuracy obtained} = \frac{\text{Number of correct Prediction}}{\text{Total number of samples}} = \frac{3}{3} = 100\%$$

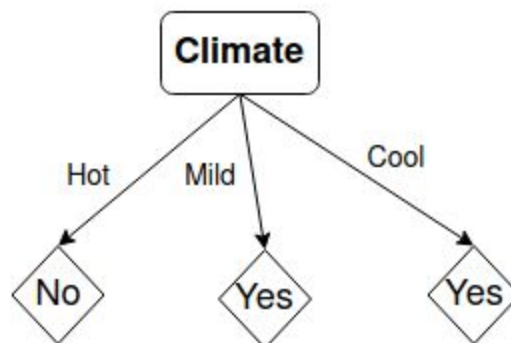
b)

Yes, the following set of training data will include climate data into the model's decision tree.

	Outlook	Climate	Humidity	Wind	Play
0	Sunny	Hot	High	Weak	No
1	Sunny	Hot	High	Strong	No
3	Rain	Mild	High	Weak	Yes
4	Rain	Cool	Normal	Weak	Yes
9	Rain	Mild	Normal	Weak	Yes
10	Sunny	Mild	Normal	Strong	Yes
11	Overcast	Mild	High	Strong	Yes

In the first step itself, the Climate feature can efficiently differentiate whether the play will happen or not due to 0 GINI impurity.

The resulting Decision tree will be as follows.

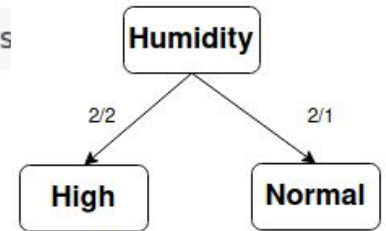
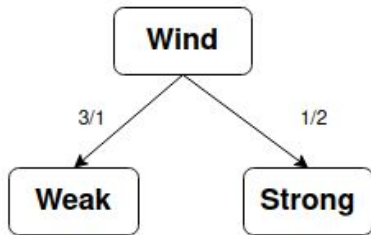


c)

- Training Dataset:(D1-D7)



	Outlook	Climate	Humidity	Wind	Play
0	Sunny	Hot	High	Weak	No
1	Sunny	Hot	High	Strong	No
2	Overcast	Hot	High	Weak	Yes
3	Rain	Mild	High	Weak	Yes
4	Rain	Cool	Normal	Weak	Yes
5	Rain	Cool	Normal	Strong	No
6	Overcast	Cool	Normal	Strong	Yes



- **Testing Dataset:(D8-D14)**

	Outlook	Climate	Humidity	Wind	Play
7	Sunny	Mild	High	Weak	No
8	Sunny	Cool	Normal	Weak	Yes
9	Rain	Mild	Normal	Weak	Yes
10	Sunny	Mild	Normal	Strong	Yes
11	Overcast	Mild	High	Strong	Yes
12	Overcast	Hot	Normal	Weak	Yes
13	Rain	Mild	High	Strong	No

- **For Finding The root node:**

Gini Impurities of each feature:

→ Outlook:

$$\text{Sunny} = 1 - \left(\frac{0}{2}\right)^2 - \left(\frac{2}{2}\right)^2 = 0$$

$$\text{Overcast} = 1 - \left(\frac{2}{2}\right)^2 - \left(\frac{0}{2}\right)^2 = 0$$

$$\text{Rain} = 1 - \left(\frac{2}{3}\right)^2 - \left(\frac{1}{3}\right)^2 = 0.45$$

$$\text{Total} = \frac{2}{7}(\text{Sunny}) + \frac{2}{7}(\text{Overcast}) + \frac{3}{7}(\text{Rain})$$

$$= 0.192$$

→ Climate:

$$\text{Hot} = 1 - \left(\frac{1}{3}\right)^2 - \left(\frac{2}{3}\right)^2 = 0.45$$

$$\text{Mild} = 1 - \left(\frac{1}{1}\right)^2 - \left(\frac{0}{1}\right)^2 = 0$$

$$\text{Cool} = 1 - \left(\frac{2}{3}\right)^2 - \left(\frac{1}{3}\right)^2 = 0.45$$

$$\text{Total} = \frac{3}{7}(\text{Hot}) + \frac{1}{7}(\text{Mild}) + \frac{3}{7}(\text{Cool})$$

$$= 0.385$$

→ Wind:

$$\text{Weak} = 1 - \left(\frac{3}{4}\right)^2 - \left(\frac{1}{4}\right)^2 = 0.375$$

$$\text{Strong} = 1 - \left(\frac{1}{3}\right)^2 - \left(\frac{2}{3}\right)^2 = 0.45$$

$$\text{Total} = \frac{4}{7}(\text{Weak}) + \frac{3}{7}(\text{Strong})$$

$$= 0.407$$

→ Humidity:

$$\text{Normal} = 1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2 = 0.5$$

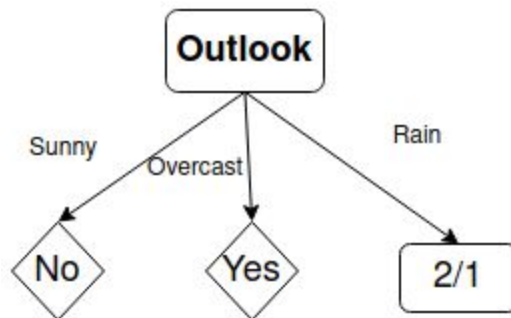
$$\text{High} = 1 - \left(\frac{2}{3}\right)^2 - \left(\frac{1}{3}\right)^2 = 0.45$$

$$\text{Total} = \frac{4}{7}(\text{Normal}) + \frac{3}{7}(\text{High})$$

$$= 0.478$$

Since Outlook has the lowest GINI impurity value, that means it will be our root node of the Decision Tree.

Current Tree:



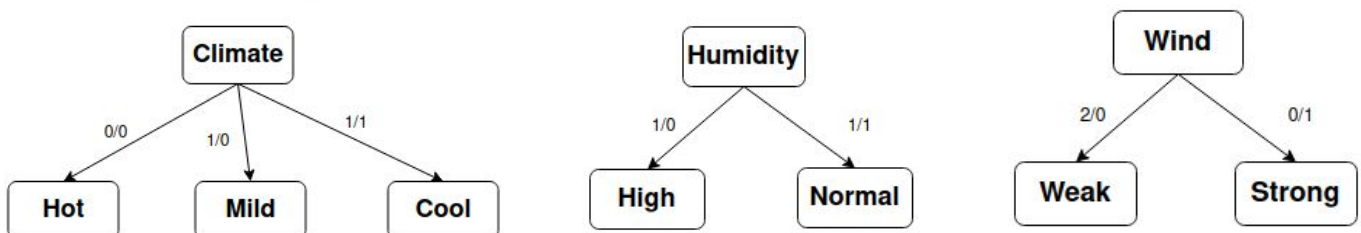
- Finding the next node in Rain Branch:

Outlook Climate Humidity Wind Play

3 Rain Mild High Weak Yes

4 Rain Cool Normal Weak Yes

5 Rain Cool Normal Strong No



Gini impurities of remaining features:

→ Climate:

Hot => N. A

$$\text{Mild} = 1 - \left(\frac{1}{1}\right)^2 - \left(\frac{0}{1}\right)^2 = 0$$

$$\text{Cool} = 1 - \left(\frac{1}{2}\right)^2 - \left(\frac{1}{2}\right)^2 = 0.5$$

$$\begin{aligned}\text{Total} &= \frac{1}{3}(\text{Mild}) + \frac{2}{3}(\text{Cool}) \\ &= 0.3\end{aligned}$$

→ Wind:

$$\text{Weak} = 1 - \left(\frac{2}{2}\right)^2 - \left(\frac{0}{2}\right)^2 = 0$$

$$\text{Strong} = 1 - \left(\frac{0}{1}\right)^2 - \left(\frac{1}{1}\right)^2 = 0$$

$$\begin{aligned}\text{Total} &= \frac{2}{3}(\text{Weak}) + \frac{1}{3}(\text{Strong}) \\ &= 0\end{aligned}$$

→ Humidity:

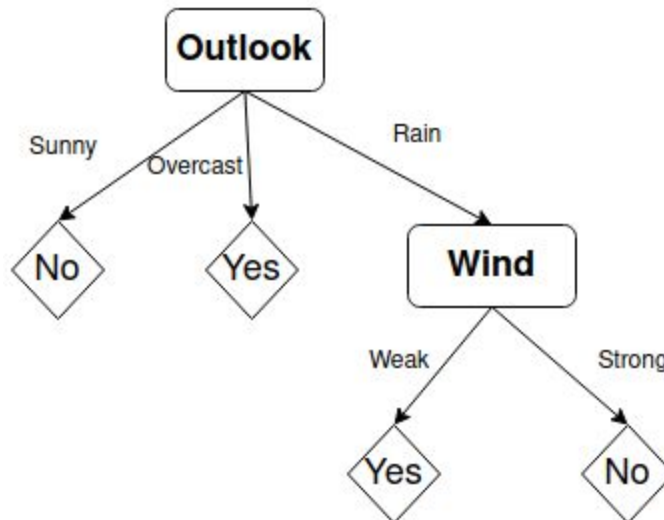
$$\text{Normal} = 1 - \left(\frac{1}{2}\right)^2 - \left(\frac{1}{2}\right)^2 = 0.5$$

$$\text{High} = 1 - \left(\frac{1}{1}\right)^2 - \left(\frac{0}{1}\right)^2 = 0$$

$$\begin{aligned}\text{Total} &= \frac{2}{3}(\text{Normal}) + \frac{1}{3}(\text{High}) \\ &= 0.34\end{aligned}$$

Since Wind has the lowest GINI impurity then it will be our next node after

Final Tree:



- Finding the accuracy

Row #	Original value	Decision tree prediction
1	No	No
2	Yes	No
3	Yes	Yes
4	Yes	No
5	Yes	Yes
6	Yes	Yes
7	No	No

$$\text{Accuracy obtained} = \frac{\text{Number of correct Prediction}}{\text{Total number of samples}} = \frac{5}{7} = 71.42\%$$

d) The following are the pruning strategies that can be applied with our current algorithm for avoiding overfitting:

- i) Set a limit on the max depth the tree can have, because more depth means more splits and hence more overfitting
- ii) Set a lower limit on the minimum number of samples required to be in a leaf node.
- iii) Cross-validation can help choose the best model.

6) Given,

$P(\text{tough} | \text{tough}) = 0.7$ \Rightarrow probability that TA said "Tough" given that the previous word was "Tough"

$P(\text{course} | \text{tough}) = 0.3$ \Rightarrow probability that TA said "Course" given that the previous word was "Tough"

$P(\text{tough} | \text{course}) = 0.5$ \Rightarrow probability that TA said "Tough" given that the previous word was "Course"

$P(\text{course} | \text{course}) = 0.5$ \Rightarrow probability that TA said "Course" given that the previous word was "Course"

To find,

$$\Rightarrow P(w_3 | w_1, w_2, w_4)$$

Using Markov Assumption we can ignore w_1

$$\Rightarrow P(w_3 | w_2, w_4)$$

Now,

Therefore,

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

$$\Rightarrow \frac{P(w_4 | w_2, w_3) * P(w_3 | w_2)}{P(w_4 | w_2)}$$

Again using markov assumption we can drop w_2 from first term in numerator and denominator can be rewritten as follows

$$\Rightarrow \frac{P(w_4 | w_3) * P(w_3 | w_2)}{\sum [P(w_4 | w_3) * P(w_3 | w_2)]}$$

$$\begin{aligned}
& \Rightarrow \frac{P(course | w3) * P(w3 | course)}{P(course | w3=course) * P(w3=course | course) + P(course | w3=tough) * P(w3=tough | course)} \\
& \Rightarrow \frac{P(course | w3) * P(w3 | course)}{0.5 \times 0.3 + 0.5 \times 0.5} \\
& \Rightarrow \frac{P(course | w3) * P(w3 | course)}{0.4}
\end{aligned}$$

Therefore for the posterior probability of $w3=tough$ is:

$$\Rightarrow \frac{0.3 * 0.5}{0.4} = 15/40 = 0.375$$

similarly for the posterior probability of $w3=course$ is:

$$\Rightarrow \frac{0.5 * 0.5}{0.4} = 25/40 = 0.625$$

7)

- a) Decision trees do not consider the independence of various features, whereas logistic regression treats features as an independent. Decision tree models are easy to prepare, easily visualizable, and it can handle both numerical and categorical data while logistic regression works only for numerical data.
- b) Decision trees are very unstable, i.e., a small change in the dataset can result in an entirely different tree being created while small changes don't affect the logistic regression model. Moreover, the Decision tree is more likely to overfit the data, whereas, in logistic regression, we associate only one parameter with each feature.
- c) Yes, we can split the dataset based on X_1 values, i.e., since the dataset is linearly separable, we can assume a cutoff value for each X_1 on X_2 . This kind of splitting can be done with a $\log(n)$ depth tree.
- d) Yes, similar to above first, we can split the dataset based on X_1 values, but since now data is not linearly separable, we cannot assume a cutoff on X_2 ; therefore, we need to split again on X_2 ; hence both the splitting will make the tree $2 * \log(n)$ height.

8)

$$\begin{aligned}
P(Y = 1|X) &= \frac{P(Y=1)P(X|Y=1)}{P(Y=1)P(X|Y=1) + P(Y=0)P(X|Y=0)} \\
&= \frac{1}{1 + \frac{P(Y=0)P(X|Y=0)}{P(Y=1)P(X|Y=1)}} \\
&= \frac{1}{1 + \exp(\ln \frac{P(Y=0)P(X|Y=0)}{P(Y=1)P(X|Y=1)})} \\
&= \frac{1}{1 + \exp(\ln \frac{P(Y=0)}{P(Y=1)} + \ln \frac{P(X|Y=0)}{P(X|Y=1)})} \\
&= \frac{1}{1 + \exp(\ln \frac{P(Y=0)}{P(Y=1)} + \sum_i \ln \frac{P(X_i|Y=0)}{P(X_i|Y=1)})}
\end{aligned}$$

$$= \frac{1}{1 + \exp(\ln \frac{1-\pi}{\pi} + \sum_i \ln \frac{P(X_i|Y=0)}{P(X_i|Y=1)})} \quad \text{----- (i)}$$

Now looking at the denominator, we can expand as follows using the gaussian distribution.

Let,

$$\theta_{i0} = P(X_i = 1|Y = 0)$$

$$\theta_{i1} = P(X_i = 1|Y = 1)$$

Then we can write as follows

$$\begin{aligned} \sum_i \ln \frac{P(X_i|Y=0)}{P(X_i|Y=1)} &= \sum_i \ln \frac{\theta_{i0}^{X_i} (1-\theta_{i0})^{(1-X_i)}}{\theta_{i1}^{X_i} (1-\theta_{i1})^{(1-X_i)}} \\ &= \sum_i (X_i \ln \frac{\theta_{i0}}{\theta_{i1}} + (1 - X_i) \ln \frac{1-\theta_{i0}}{1-\theta_{i1}}) \\ &= \sum_i (X_i \ln \frac{\theta_{i0}(1-\theta_{i1})}{\theta_{i1}(1-\theta_{i0})} + \ln \frac{1-\theta_{i0}}{1-\theta_{i1}}) \end{aligned}$$

Putting this value back in eqn (i).

$$P(Y = 1|X) = \frac{1}{1 + \exp(\ln \frac{1-\pi}{\pi} + \sum_i (X_i \ln \frac{\theta_{i0}(1-\theta_{i1})}{\theta_{i1}(1-\theta_{i0})} + \ln \frac{1-\theta_{i0}}{1-\theta_{i1}}))}$$

$$P(Y = 1|X) = \frac{1}{1 + \exp(w_0 + \sum_i^n (w_i X_i))}$$

Parameters:

$$w_0 = \exp(\ln \frac{1-\pi}{\pi} + \sum_i (\ln \frac{1-\theta_{i0}}{1-\theta_{i1}}))$$

$$w_1 = \ln \frac{\theta_{i0}(1-\theta_{i1})}{\theta_{i1}(1-\theta_{i0})}$$