ML REPORT ASSIGNMENT 2

1. (t-SNE, PCA & SVD. Use Dataset A and perform the following operations. You can use sklearn library for these operations.

(a) Read about PCA and write a note on how it works.

PCA (Principal component analysis) is an unsupervised statistical technique which allows us to summarize the data with a high number of features by eliminating and extracting features from existing ones thus performing dimensionality reduction.
PCA seeks to maximize variance and preserves large pairwise distances. I.e. different things end up far apart same as in cluster formation or nearest neighbours, thus this can lead to poor visualization when structures are non-linear.
PCA converts a set of observations of possibly correlated variables (entities each of which takes on various numerical values) into a set of values of linearly uncorrelated variables.

Steps of PCA:

1) First, it performs mean normalization, for every feature, that is find mean of all the samples and then subtract it from the value of that specific feature for each sample, thus ensuring that mean of new samples for that particular feature is 0.
2) Generate a covariance matrix of all combinations of features which is basically K X K matrix.
3) Find Eigenvectors and Eigenvalues which intuition is basically the following: Find a line whose sum of the perpendicular distance to that line is minimum (same as finding line for linear regression), then the sum of the square of the distances of the projections on the line is called as eigenvalue for the vector formed by the combination of the features we formed. Now representing the projection of the point on that line and scaling such that distance from 0,0 is 1 that combination or ratio of x to y (plotted feature) is called as eigenvector.
4) Sort and choose the top eigenvectors (whose variation show up to a threshold percentage of variation or let's say we take m).
5) Now, use this matrix of m eigenvectors to transform the original data.
   I.e. Transformed data = feature matrix * top_m_eigenvector_matrix
   Thus dimension is reduced from k to m

(b) Read about SVD and write a note on how it works.

SVD is singular value decomposition which is basically a method to reduce dimensions by factorizing data into 3 components that are U, sigma and V. It is popular for dimensionality reduction of sparse data

Steps for SVD are as follows:

1) A matrix (of n samples and k features) is broken down as 3 matrices, U sigma and V, where U is known as left singular matrix, sigma is a diagonal matrix whose diagonal elements are the eigenvalues of the matrix while non-diagonal elements are all zero, and V is a right singular matrix.
2) Now for the dimensionality reduction, we will choose m Rows of Sigma which have the largest eigenvalue, thus we will have to choose the columns of V same as a number of rows in order to maintain matrix multiplicativity.
3) For converting the original matrix to reduced matrix we will multiply it with sigma*VT which is updated by choosing the m rows with max reign value and similarly m rows from V or m columns from Vt prime which when multiplied by original matrix give us the transformation matrix with dimensionality reduced from nXk to nXm.

(c) Read about t-SNE and write a note on how it works.

T- SNE (t-Distributed Stochastic Neighbor Embedding) is an unsupervised and non-linear technique mainly used for data visualization of high dimensional data and dimensionality reduction.
It can also help us visualize data which cant be done by PCA as it is non-linear.

It works on the concept of similarity i.e finding the euclidean distance from one point to all the other points and then plotting them on a curve such that the similar points come closer to the centre while non- similar point stays at the end of the curve.

Steps for TSNE:

1) Randomly chose a point between all points and then find a similarity score between all the point to the point chosen using the gaussian distribution curve.
2) Do this again but replacing the gaussian with Cauchy distribution.
3) Similar points attract each other while dissimilar points repel each other, using this idea try change the coordinates and repeat this process until you see that nothing better can achieve. This idea is done by Kullback-Liebler divergence (KL) which is being used to compare these probabilities and gradient descent is used to minimize KL cost function.

(d) Read about stratified sampling and perform an 80:20 stratified train-test split on the dataset. Comment on the class frequency of training and testing samples.

Stratified sampling is basically a sampling technique where data is divided into some k parts and then the random element is chosen from each part equally to maintain balance in the data which can be an imbalance if random and picking are done. This ensures the division of data into balanced training and test samples.

The ratio for the train to test for each class is coming out to be approximately 4 for each class (i.e from 0 - 9), which is a good sign as it shows us that each class is divided almost in the ratio 80:20 thus dataset for train and test is balanced

Regarding the count, we can see that the frequencies of each class are almost the same, thus training of the model will be better because data is balanced without any biases towards a particular class.

```
ratio of train to test for classes
4.0
3.98989898989899
3.9746835443037973
3.988235294117647
4.0120481927710845
3.975
4.011363636363637
4.011627906976744
4.0
4.038461538461538
train count 320 395 314 339 333 318 353 345 328 315
test count 80 99 79 85 83 80 88 86 82 78
```

(e) Use PCA on the dataset provided. Train a Logistic Regression model on the training set and report the test accuracy. Further, use t-SNE to analyze the training data.

```
from sklearn.linear_model import LogisticRegression
lr= LogisticRegression(max_iter=10000)
lr.fit(train_X,train_Y)
```

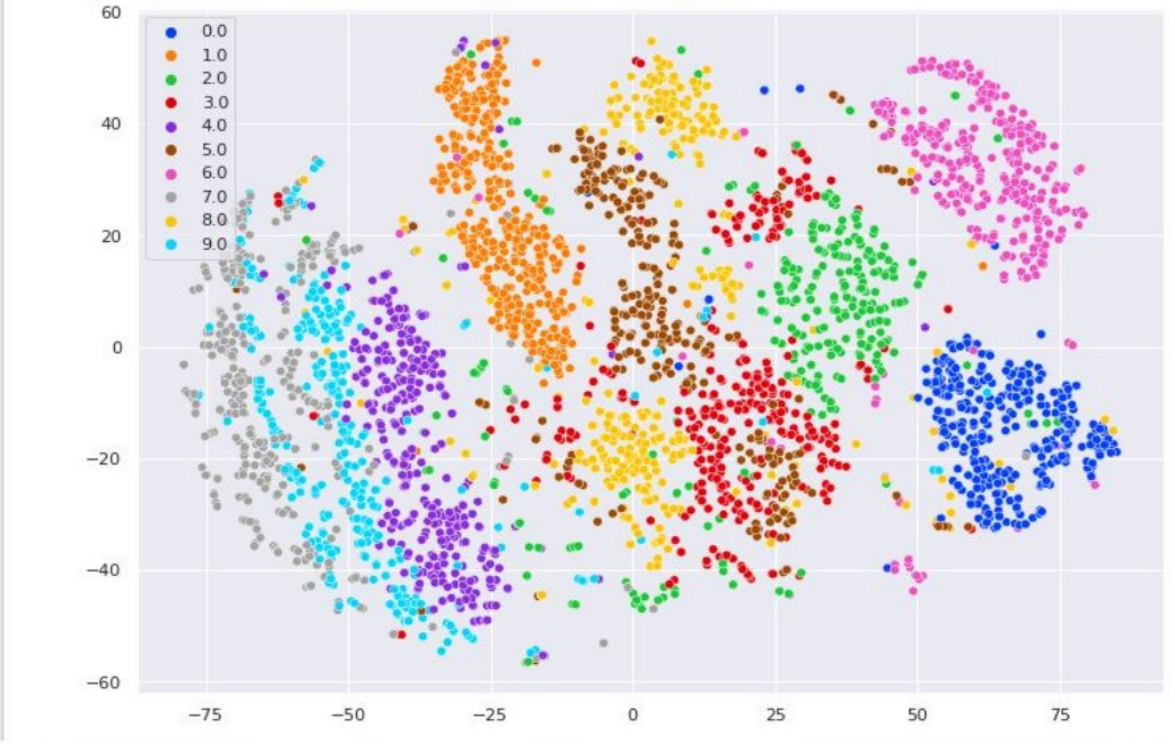
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=10000,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
lr.score(test_X,test_Y)*100
```

```
87.26190476190476
```

ACCURACY: 87.2619%

```
/usr/local/lib/python3.6/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the f
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f55b1fe3cf8>
```



(f) Use SVD on the dataset provided. Train a Logistic Regression model on the training set and report the test accuracy. Further, use t-SNE to analyze the training data.

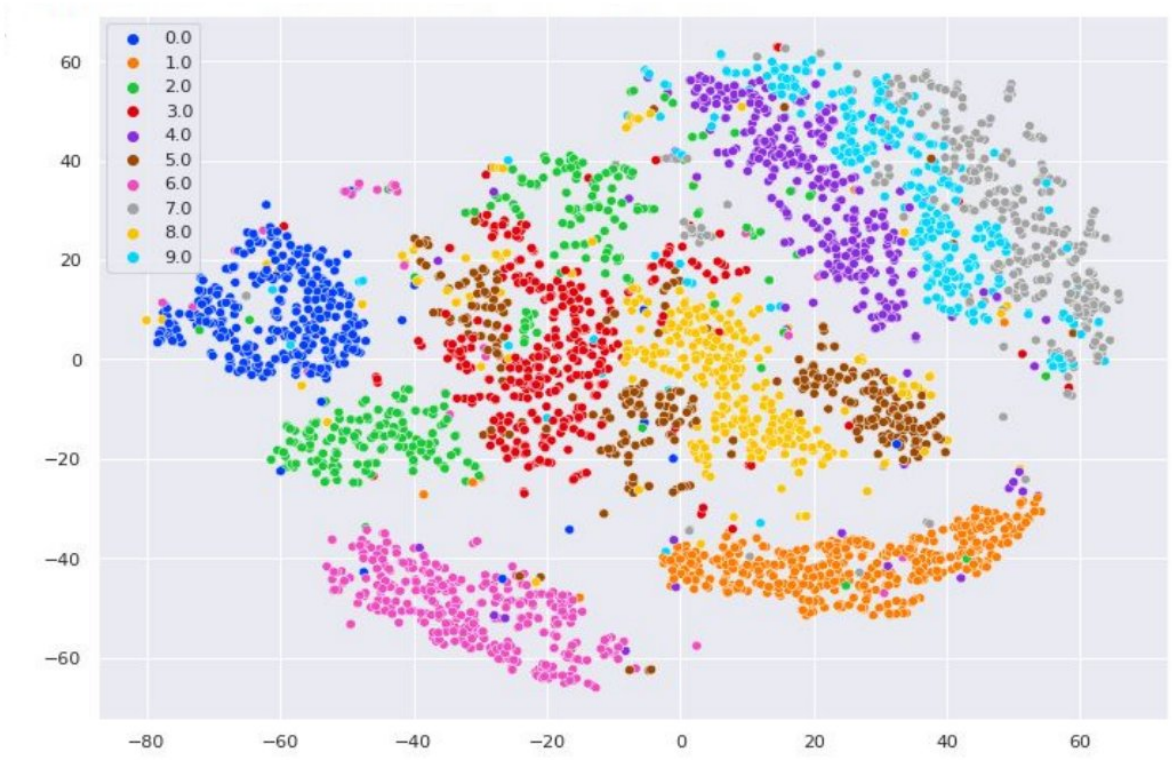
```
from sklearn.linear_model import LogisticRegression
lr= LogisticRegression(max_iter=10000)
lr.fit(train_X,train_Y)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=10000,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
lr.score(test_X,test_Y)*100
```

```
88.69047619047619
```

ACCURACY: 88.6904%



(g) Compare the accuracy obtained while using PCA & SVD and write a note on the results obtained.

The Accuracies of both dimensionality reduction by PCA and SVD came out to be almost the same (87.26% and 88.6%).
And without PCA /SVD accuracy came out to be 87% thus we are successfully able to reduce the dimension of data without losing much information and thus our computation power decreased a lot,

2. (a) Use dataset C and train a Linear Regression model to predict weight based on the height of the person. Using bootstrapping, measure the bias & variance of the model and report them. Hint - Refer to Lecture 6 (Revision).
(b) Assuming noise in the data to be zero, report the value of MSE − Bias2 − Variance and give a comment on the value obtained.**

Ans: no of bootstrap samples -20
No .of samples in each bootstrap sample -100

```
bootstrap(np_array,20,100)
```

```
(2000, 2)
(2000, 20)
bias 9.838830668017383 variance 2.9034668047580583
AVG MSE 155.1347894533053
irreducible uncertainity 55.42873373462785
```

This is the uncertainty or the error demonstrated by the model. This includes the uncertainty involved in the data we are trying to model as well as whether the model is correctly modeling this data which is coming out to be 55.4%.

3. Use sklearn Decision Tree(DT) and Gaussian Naive Bayes(GNB) classifier train it on Dataset A & B independently. Split the data into 60 − 20 − 20 train-Val-test split.
(a) Find optimal depth as a parameter in-case of DT using Grid Search and use K-Fold cross-validation to validate it. Implement your own K-Fold cross-validation function from scratch and use for both GNB and DT. Make these functions in such a way so that these can be used in future assignments.

K fold

```
def k_fold(k,model,train,ans,ans_train,dp=2):

  '''
  k fold- runs a k fold for a specific model as given in the params

  input

  k - represnt k for k fold ( integer)
  model - any model from sklearn which has capabilities of set param, .fit and .predict
  train - train data ( X an Y both added to maintain order for shuffling)
  ans - validation error for each fold as 2 d array
  ans_train - training error for each fold

  optional:
  dp=2 - depth if its descision tree ( default set 2 for every other model where depth deosnt make sense)

  output - none ( ans,ans_train are modified in place)
  '''
```

SKLEARN decision tree:

DATASET A:

```
print("validation accuracies" ,*np.mean(ans,axis=1))
max_v = np.argmax(np.mean(ans,axis=1))
print("optimal depth = ", max_v+2)
```

```
validation accuracies 32.55952380952381 40.20833333333333 53.720238095238095 61.66666666666666 65.89285714285714 69.464285
optimal depth =  38
```

DATASET B

```
print("validation accuracies" ,*np.mean(ans,axis=1))
max_v = np.argmax(np.mean(ans,axis=1))
print("optimal depth = ", max_v+2)
```

```
validation accuracies 56.964285714285715 56.04166666666667 55.625 56.69642857142857 56.81547619047619 56.63690476190476 56.
optimal depth =  12
```

GNB:
DATASET A:

```
print("validation accuracies" ,*ans)
max_v = np.argmax(np.mean(ans,axis=1))
```

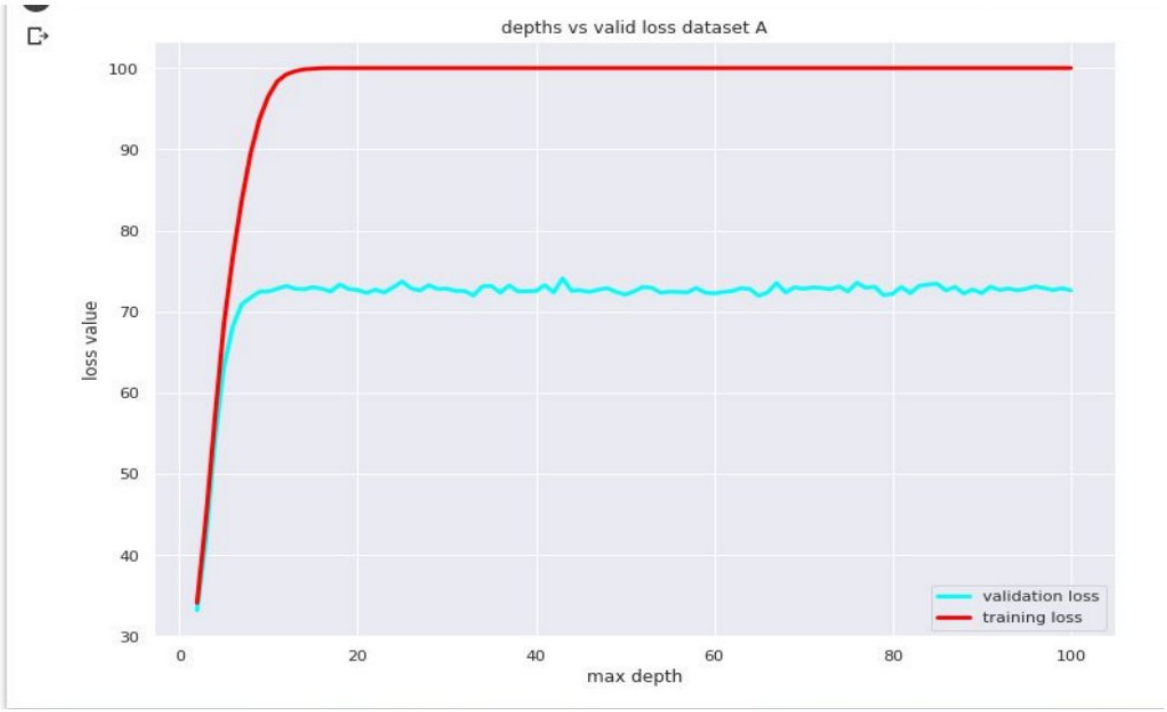
```
validation accuracies [57.38095238 57.38095238 58.69047619 57.02380952]
```

DATASET B:

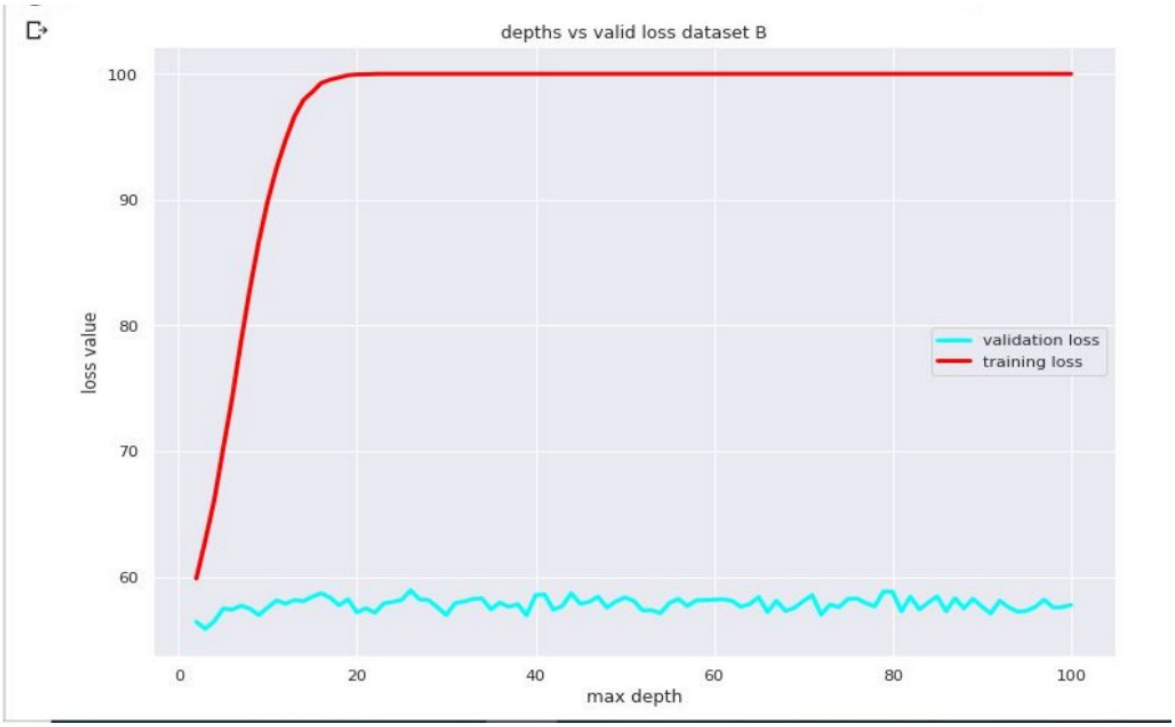
```
print("validation accuracies" ,*ans)
max_v = np.argmax(np.mean(ans,axis=1))
```

```
validation accuracies [56.42857143 60.11904762 59.88095238 55.83333333]
```

(b) For DT plot training and validation accuracy plot with respect to tree depth and write your analysis.
DATASET A



DATASET B



(c) Save the best model, load the saved model to predict the results on the test data.

DATASET A:

```
    clf = tree.DecisionTreeClassifier(max_depth=max_v+2)

    clf.fit(train[:,0:len(train[0])-1],train[:,len(train[0])-1:],)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=38, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

```
import pickle
f = "best_model.sav"
pickle.dump(clf,open(f,'wb'))

c = pickle.load(open(f,'rb'))
r= c.score(test_X,test_Y)
ypred = c.predict(test_X)
print(r)
```

```
0.7357142857142858
```

DATASET B

```
[169] max_v = np.argmax(np.mean(ans,axis=1))
      clf = tree.DecisionTreeClassifier(max_depth=max_v+2)

      clf.fit(train[:,0:len(train[0])-1],train[:,len(train[0])-1:],)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=12, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

```
import pickle
f = "best_model.sav"
pickle.dump(clf,open(f,'wb'))

c = pickle.load(open(f,'rb'))
r= c.score(test_X,test_Y)
ypred = c.predict(test_X)
print(r)
```

```
0.6
```

(d) Write a function evaluation metric to evaluate testing data. The function should calculate accuracy, precision, recall, F1-Score, plot ROC-curve and return the confusion

matrix. In the case of multi-class data, it should return Macro and Micro Average values. Read Macro and Micro average values in Multi-class data.

DATASET A:

```
CONFUSION MATRIX
[[72  0  2  3  1  5  1  1  0  2]
 [ 0 83  5  0  1  0  0  3  0  0]
 [ 2  1 57  4  7  0  7  4  3  0]
 [ 0  5  4 55  1  9  1  0 10  4]
 [ 1  2  3  1 62  1  0  1  3  9]
 [ 7  3  3  5  1 45  1  1  3  3]
 [ 3  0  1  1  2  4 70  2  4  1]
 [ 1  0  3  3  1  1  0 62  1  9]
 [ 1  0  3  5  6  2  4  1 54  3]
 [ 2  2  3  4  6  0  0  4  5 58]]
          SKLEARN EVALUATION METRIC
precision        recall          F1_score
0.809            0.8276          0.8182
0.8646           0.9022          0.883
0.6786           0.6706          0.6746
0.679            0.618           0.6471
0.7045           0.747           0.7251
0.6716           0.625           0.6475
0.8333           0.7955          0.814
0.7848           0.7654          0.775
0.6506           0.6835          0.6667
0.6517           0.6905          0.6705
0.7328           0.7325          0.7322
accuracy =  73.57142857142858
```

NOTE: (The 2nd last row is MACRO AVG of the data)

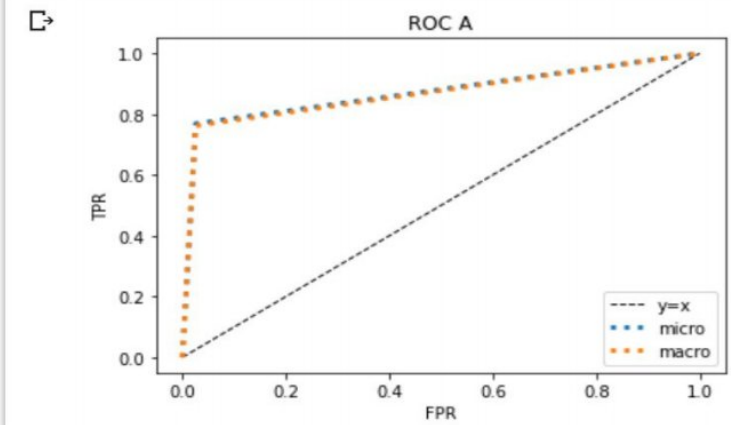
Comparing it with inbuilt function:

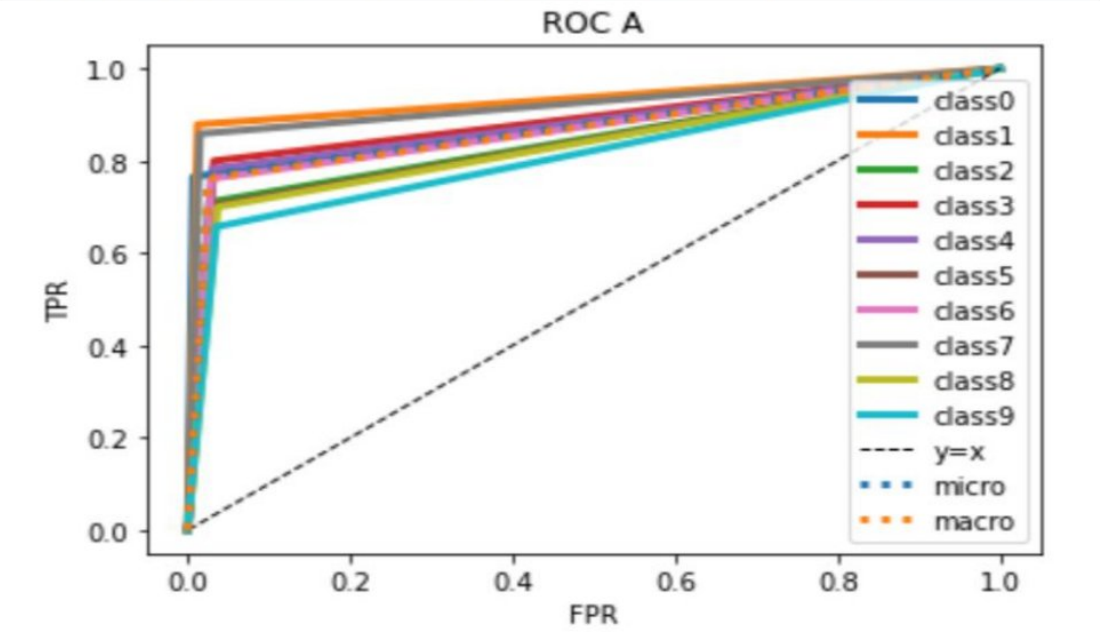
```
from sklearn.metrics import classification_report
print(classification_report(ypred,test_Y.T[0]))
```

```
              precision    recall  f1-score   support

           0       0.81      0.83      0.82        87
           1       0.86      0.90      0.88        92
           2       0.68      0.67      0.67        85
           3       0.68      0.62      0.65        89
           4       0.70      0.75      0.73        83
           5       0.67      0.62      0.65        72
           6       0.83      0.80      0.81        88
           7       0.78      0.77      0.77        81
           8       0.65      0.68      0.67        79
           9       0.65      0.69      0.67        84

    accuracy                           0.74       840
   macro avg       0.73      0.73      0.73       840
weighted avg       0.74      0.74      0.74       840
```

ROC curve:

```
PLOT_ROC(c,test_X,test_Y.T[0],10,40)
```





DATASET B:

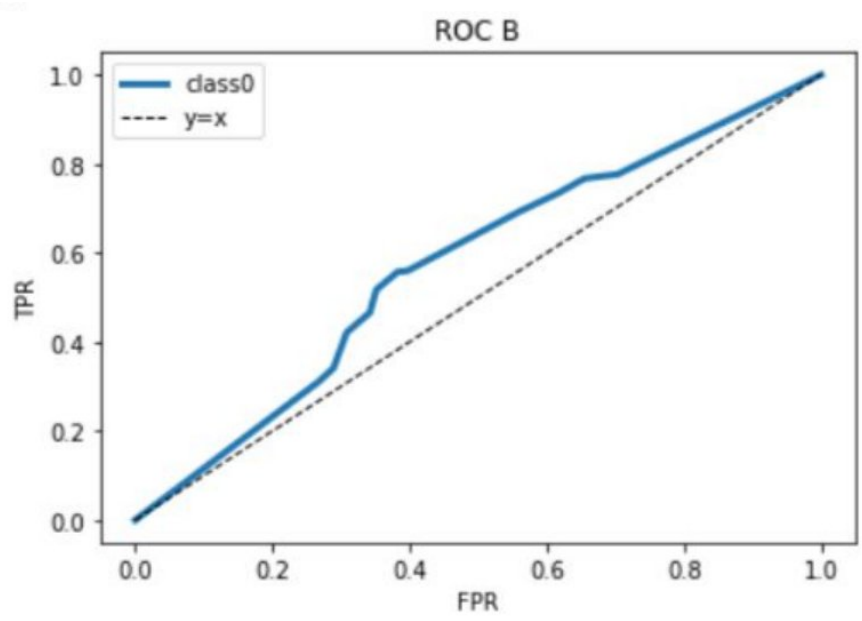
```
evaluation_report(ypred,test_Y.T[0],2)
```

```
CONFUSION MATRIX
[[261 165]
 [164 250]]
         SKLEARN EVALUATION METRIC
precision        recall          F1_score
0.6141          0.6127          0.6134
0.6024          0.6039          0.6031
0.6083          0.6083          0.6083
accuracy =  60.83333333333333
```

```
from sklearn.metrics import classification_report
print(classification_report(ypred,test_Y.T[0]))
```

```
              precision    recall  f1-score   support

           0       0.61      0.61      0.61       426
           1       0.60      0.60      0.60       414

    accuracy                           0.61       840
   macro avg       0.61      0.61      0.61       840
weighted avg       0.61      0.61      0.61       840
```

ROC curve:



4. Implement Gaussian Naive Bayes from scratch(use of Numpy and Scipy allowed) and compare results with sklearn's implementation on dataset A, B

DATASET A: (80-20 split)

```
m_X,var_X = mean_variance(train,train_Y)
p =predict(test,m_X,var_X,p_c)
```

```
(784, 10)
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
```

```
accuracy(p,test_Y)
```

```
47.73809523809524
```

GNB:

```
[230] from sklearn.naive_bayes import GaussianNB
      nb = GaussianNB()
      nb.fit(train, Y_l[:8*len(X)//10])
```

```
GaussianNB(priors=None, var_smoothing=1e-09)
```

```
pr =nb.predict(test)
from sklearn.metrics import accuracy_score
accuracy_score(pr,Y_l[8*len(X)//10:])*100
```

```
51.66666666666667
```

For dataset A, there seems to be a difference of 5% between validation error from GNB and self made code, i think that is because GNB from sklearn does further optimizations and manipulations to fir or predict the model perfectly when two probabilities are close or equal in predict proba.

DATASET B:

```
m_X,var_X = mean_variance(train,train_Y)
p =predict(test,m_X,var_X,p_c)
```

```
(2048, 2)
[[4299.77037645 4100.24749942]
 [4187.01282942 3857.70363452]
 [4179.13123198 4024.16993884]
 ...
 [2939.81519062 3244.27668254]
 [3311.10094518 3689.94690847]
 [3018.55283309 3392.54105244]]
```

```
accuracy(p,test_Y)
```

```
58.333333333333336
```

GNB:

```
[245] from sklearn.naive_bayes import GaussianNB
      nb = GaussianNB()
      nb.fit(train, Y_l[:8*len(X)//10])
```

```
GaussianNB(priors=None, var_smoothing=1e-09)
```

```
pr =nb.predict(test)
from sklearn.metrics import accuracy_score
accuracy_score(pr,Y_l[8*len(X)//10:])*100
```

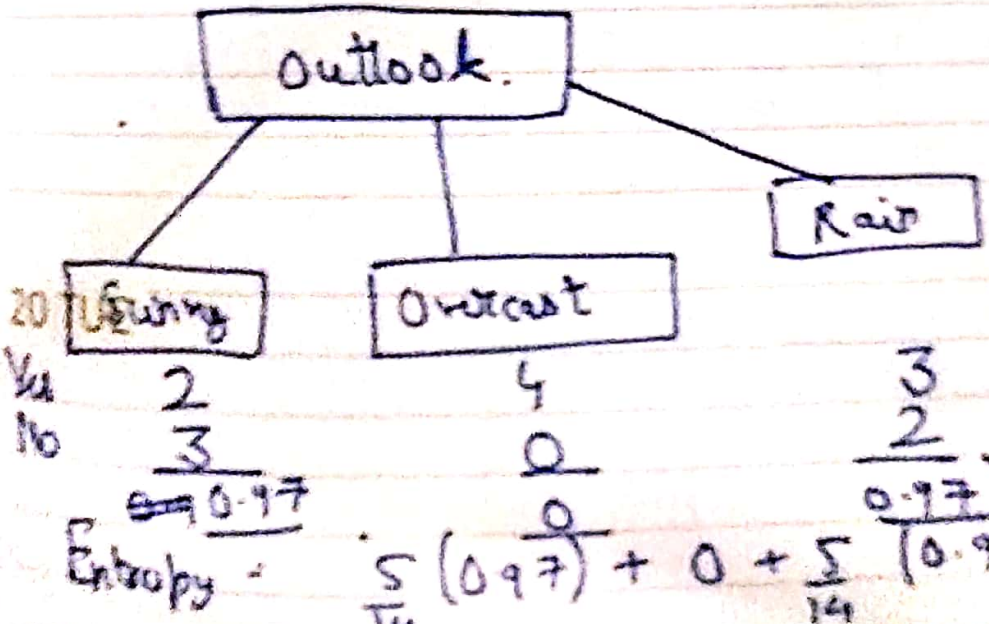
```
58.333333333333336
```

It turns out to be EXACT SAME :)

2012 March

Q5 a) Selecting best attribute to split do which has the least entropy i.e. $-\sum_{i=1}^{n} p_i \log p_i$ will be selected.

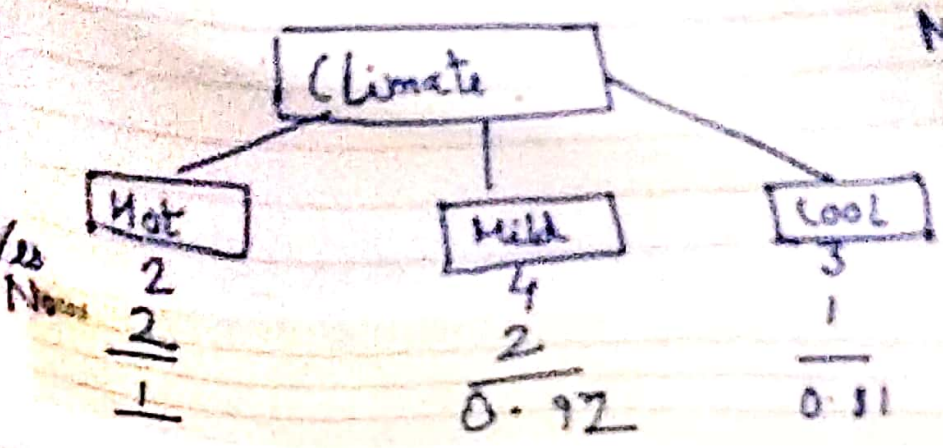
for outlook.



$-\left[\frac{3}{5}\log\frac{3}{5} + \frac{2}{5}\log\frac{2}{5}\right]$
$\Rightarrow (0.97)$

Entropy = $\frac{5}{14}(0.97) + 0 + \frac{5}{14}(0.97)$

$= 0.693$



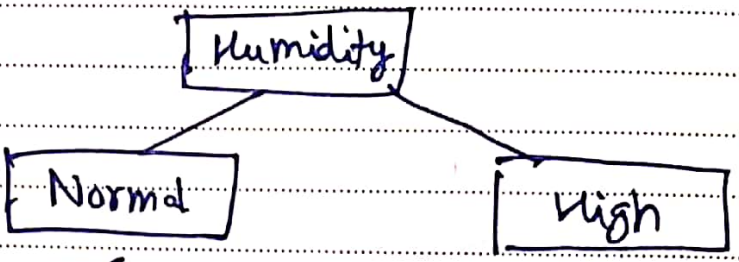
Net Entropy

$= \frac{4}{14}(1) + \frac{5}{14}(0.92) + \frac{4}{14}(0.81)$

$= 0.911$

for Humidity.

21 WED



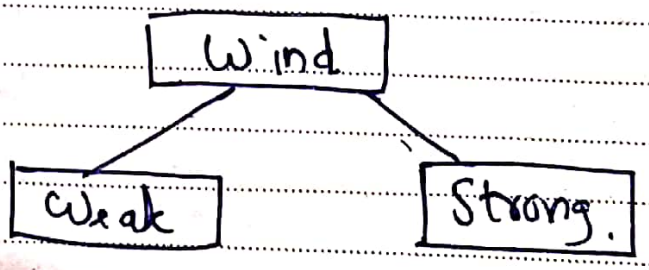
	Normal	High
Yes	6	3
No	1	4
	0.59	0.98.

$$\text{Net} = \frac{7}{14}(0.59) + \frac{7}{14}(0.98)$$

$$= 0.785$$

for Wind

22 THU



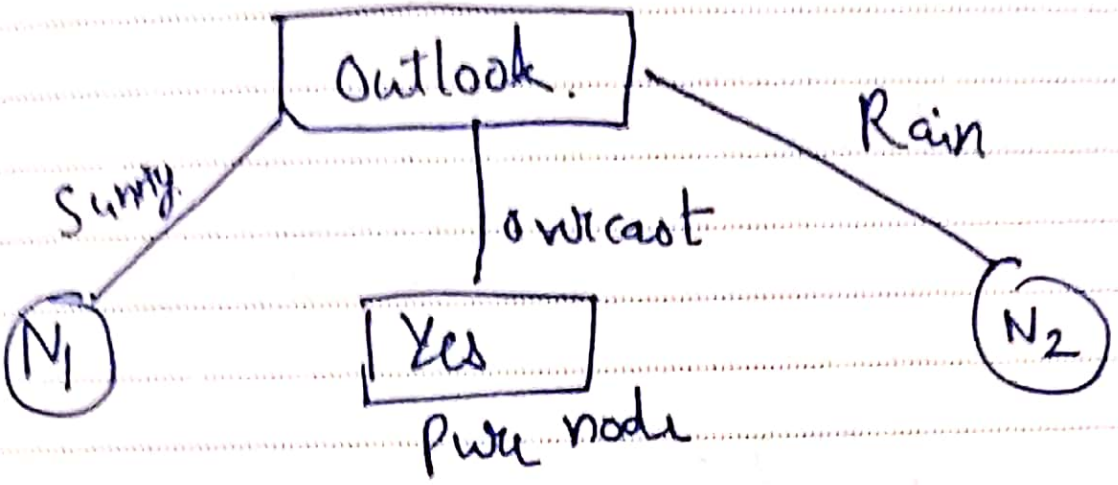
	Weak	Strong.
Yes	6	3
No	2	3
	0.81	1

$$\frac{8}{14}(0.81) + \frac{6}{14}(1) = 0.891$$

Thus Outlook gives best split

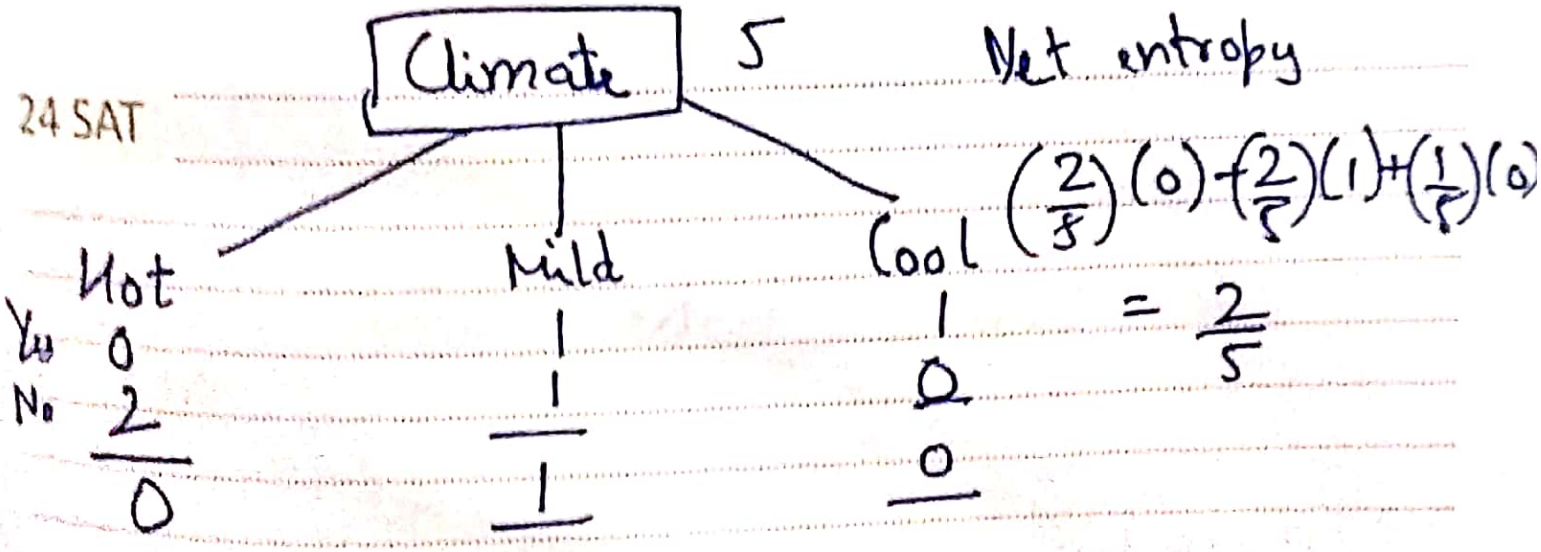
Notes

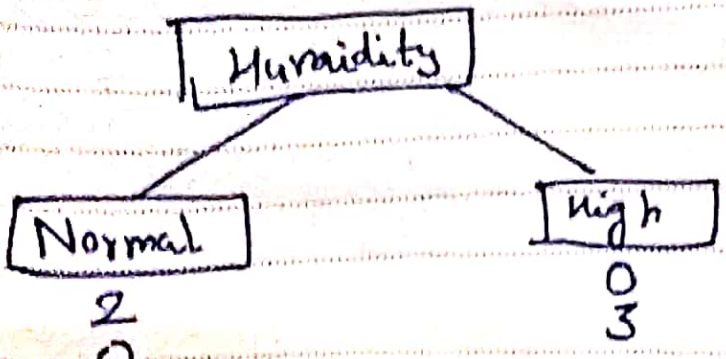
23 FR Choosing Outlook.



for N1 (outlook = Sunny)

24 SAT



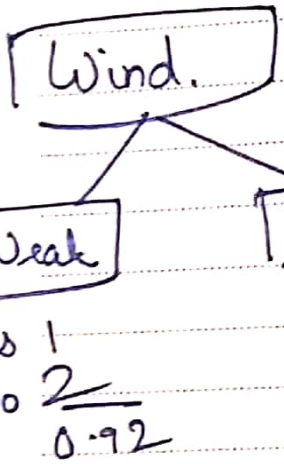


Pure node entropy = 0.

Notes

March 2012

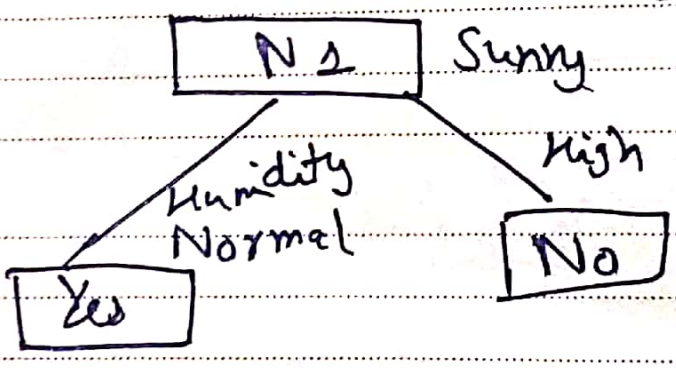
25 SUN



Entropy

$\left(\frac{3}{5}\right)(0.92) + \left(\frac{2}{5}\right)(1) = 0.952$

Thus choosing humidity to be best with entropy = 0.



26 MON

for N2 (Outlook = Rain)

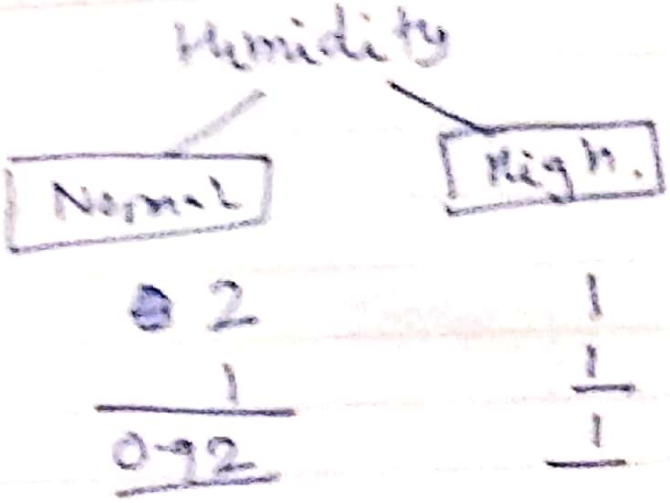
	Hot	Mild	Cool
Yes	0	2	1
No	0	1	1
	0	0.92	1

Entropy

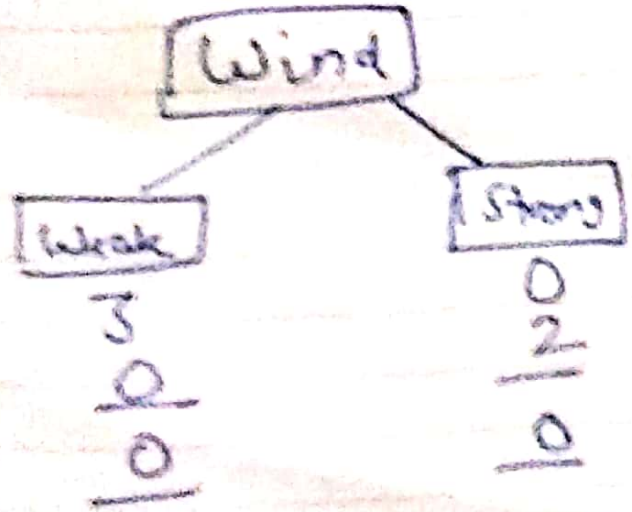
$\left(\frac{3}{5}\right)(0.92) + \frac{2}{5}(1)$

$= 0.952$

Notes



$$\left(\frac{5}{7}\right)(0.92) + \left(\frac{2}{7}\right)(1) = 0.952$$



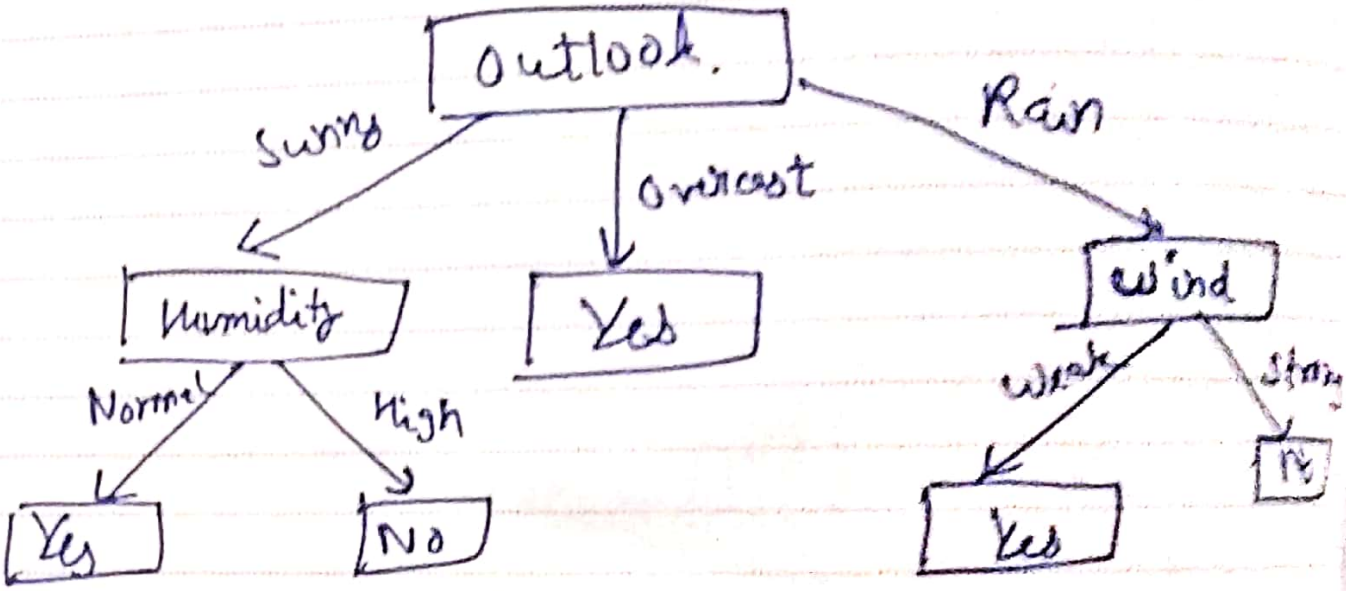
Entropy = 0

But in wind for N2 then

N2 =

March 2012

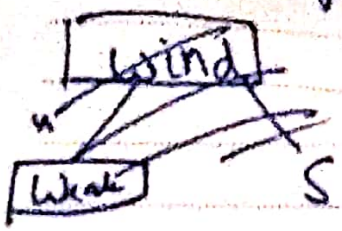
Final DT



5 b) Let new sample be ⇒

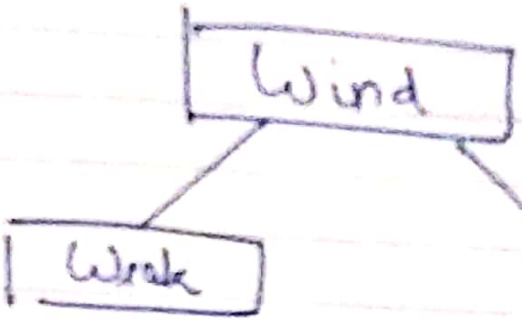
{ Day = D15, Outlook = Rain,
Wind = Weak,
Humidity = High
Climate = Hot
play match = No.

Then for node ~~is~~ N2.



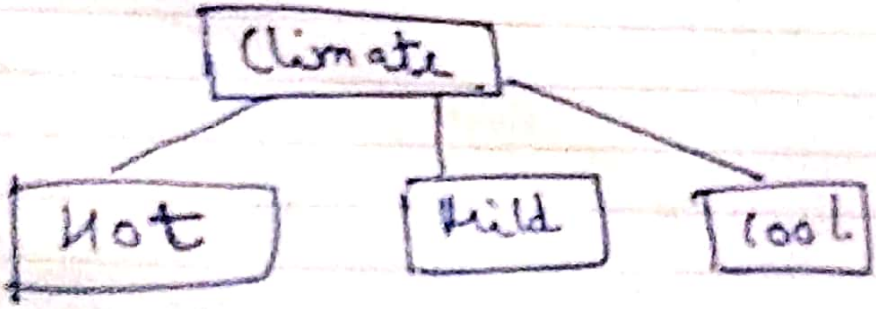
Notes

2012 March



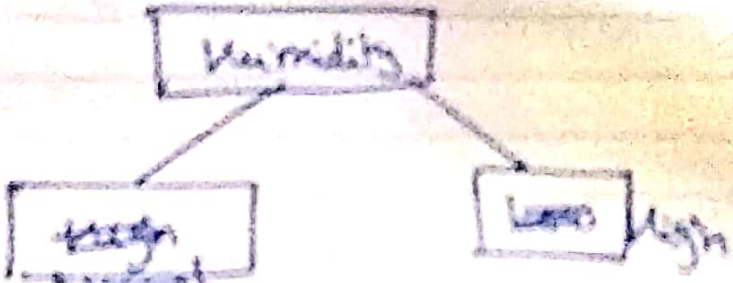
	Weak	Strong
Yes	3	0
No	1	2
	0.81	0

$\frac{4}{6}(0.81) + \frac{2}{6}) = 0.54$



	Hot	Mild	Cool
Yes	0	2	1
No	1	1	1
	0	0.92	1

$\frac{1}{6}(0) + (\frac{3}{6})(0.92) + (\frac{2}{6})(1) = 0.79$

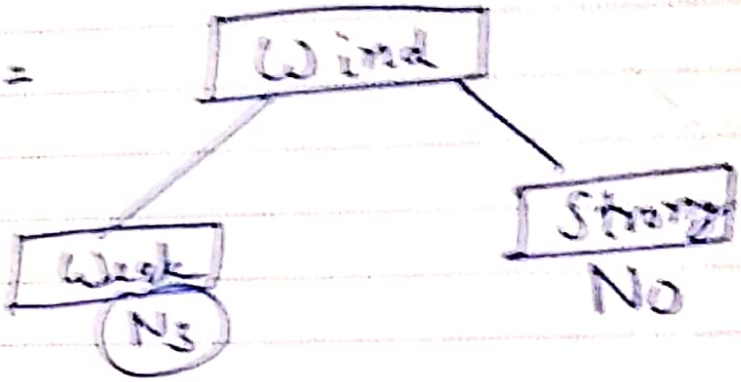


	High	Normal
Yes	2	1
No	1	2
	0.92	0.92

$(\frac{1}{2})(0.92) + \frac{1}{2}(0.92) = 0.92$

April 2012

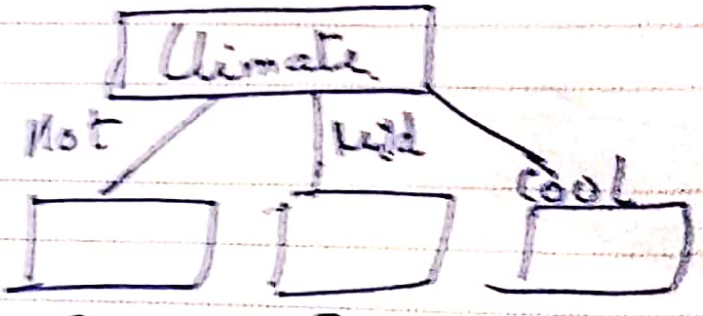
Thus wind is chosen ;

N2 = 

Pure node.

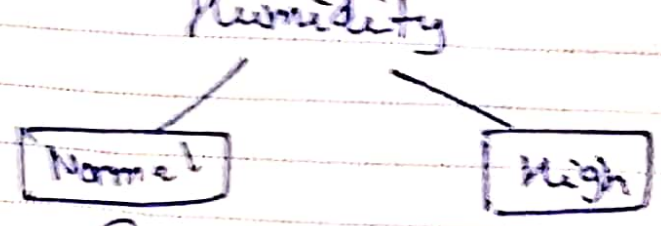
For wind = weak

N_3 = Climate



	Hot	Mild	Cool
Yes	0	2	1
No	1	0	0
	0	0	0

Entropy = 0.



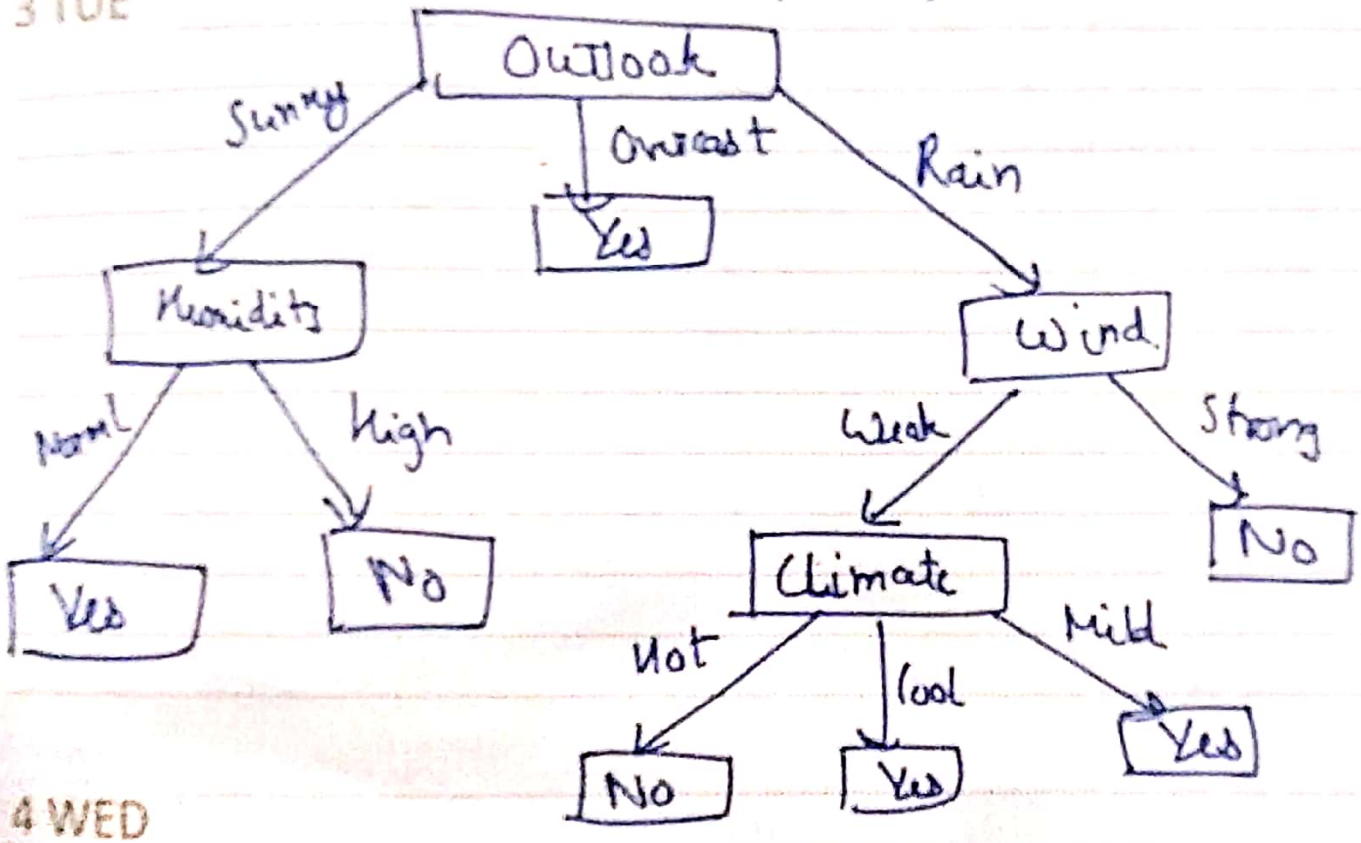
	Normal	High
Yes	2	1
No	0	1
	0	1

$(\frac{2}{4})(1) = 0.5$

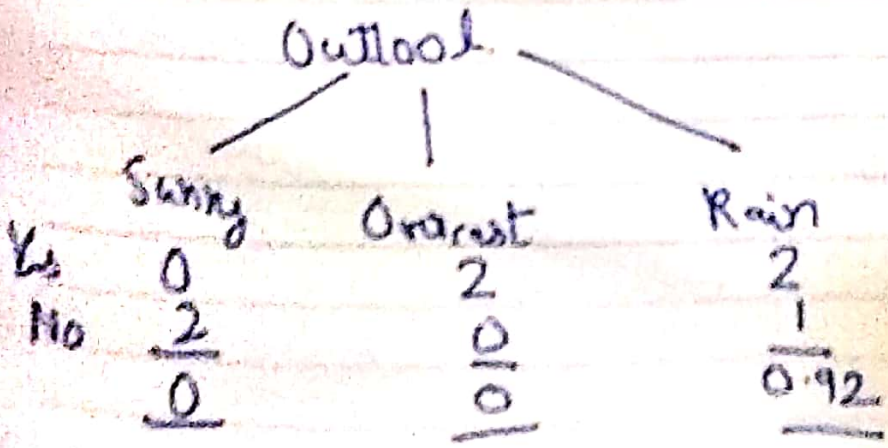
Thus Climate selected.

2012 April

Final DT for (b)



c) D1 - D7 training.



Entropy $= \left(\frac{3}{7}\right)(0.92) + \frac{2}{7}(0) + \frac{2}{7}(0)$

$= 0.394$

Scanned with CamScanner

5 TH

Humidity

	Normal	High
Yes	2	2
No	1	2
	1	1
	0.92	

$\frac{3}{7}(0.92) + \frac{4}{7}(1)$

$= 0.96$

Wind

	Weak	Strong
Yes	3	1
No	1	2
	0.81	0.92

$= \frac{4}{7}(0.81) + \frac{3}{7}(0.92) = 0.85$

Climate

	Hot	Mild	Cool
Yes	1	1	2
No	2	2	0
	0.92	0	0.92

$= \frac{3}{7}(0.92) + \frac{3}{7}(0.92)$

$= 0.78$

Thus Outlook is chosen.

Notes

2012 April

7 SAT

Outlook

Sunny → No — Pure node

Rain → N1

Outcast → Yes — Pure node

N1

Wind

Weak → Yes

Strong → No

Wind

	Weak	Strong
Yes	2	0
No	0	1
	0	0

8 SUN Entropy = 0

~~The~~ ~~choose~~

~~Yes~~

Humidity

	High	~~Low~~ Normal	$\left(\frac{2}{3}\middle\vert 1\right) = 0.66$
Yes	1	1	
No	0	1	
	0	1	

Climate

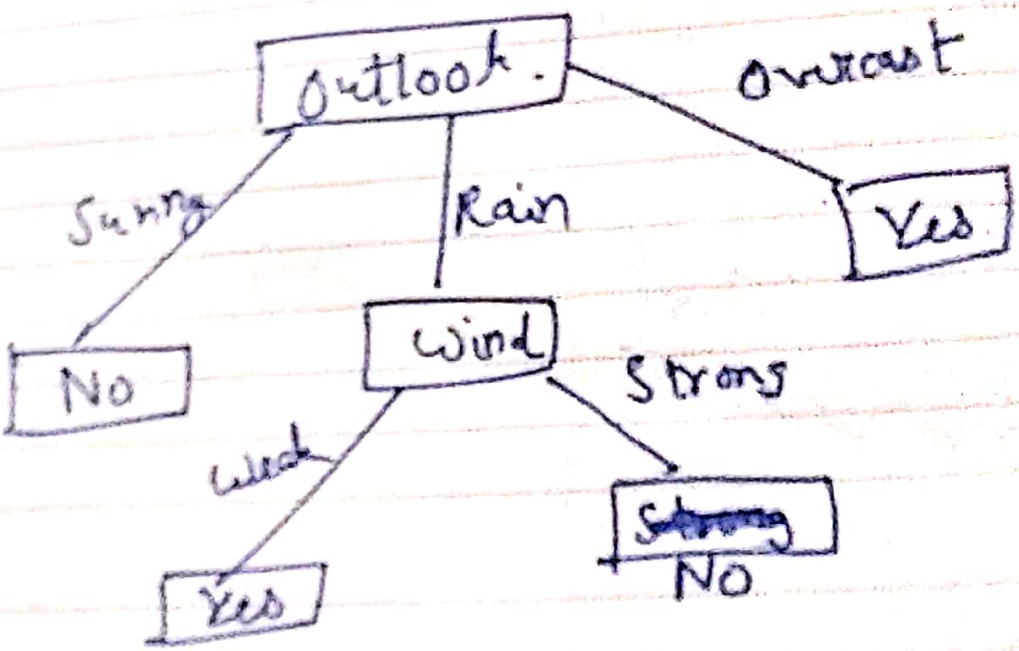
	Hot	Mild	Cool	$\frac{2}{3}(1) = 0.66$
Yes	0	1	1	
No	0	0	1	
	0	0	1	

Notes

April 2012

Thus wind is chosen to

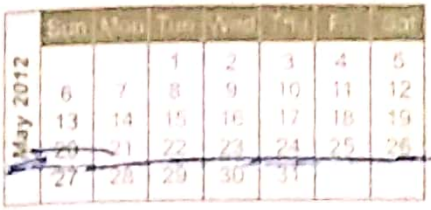
Final DT for (c)



testing for D8 - D14

	Ypred	Yactual	
D8	No	No	✓
D9	No	Yes	X
D10	Yes	Yes	✓
P11	No	Yes	X
P12	Yes	Yes	✓
D13	Yes	Yes	✓
D14	No	No	✓

$= \left(\frac{5}{7}\right) \times 100 = 71.43\%$



11 WED Its because of lack of training data, as decision tree tries to get 100% accuracy on training data i.e overfitting, its a case of overfitting due to lack of data.

The other data was also dependent on humidity and such as seen in (a) but This decision tree lacks it.

The lack of training data was unable to understand the true rules for the DT of the full table data.

12 THU

★ Note: This all is from lecture of DHG

(d) A model is said to be overfitting when we train the model with data a lot so when trained give a lot of error in test.

Now given ID3, there are several approaches to avoid overfitting in which one can control the depth of the DT i.e Reduced Error pruning.

Notes

April 2012

13 FRI

There are two types of pruning :

(i) Pre-pruning :- Stop growing the tree when it reaches the optimal depth (i.e overfitting increases after optimal depth).
Deciding optimal depth by : MDL, Statistical bounds, complexity of DT.

(ii) Post-pruning :- First grow the tree fully which results as perfect classification, then apply some rules to reduce the depth.

14 SAT

★ Pre-pruning :-

⟹ known as early stopping rule

⇒ Stopping condition for a node
- → Stop if all instance belong to same class
- → Stop if all attribute values are same.

⇒ Stop if number of instances is less than the threshold

Notes

15 SUN

⇒ Stop if class distribution is independent of available features

⇒ Stop if expanding doesn't improve impurity measures (Gini, information gain, entropy)

⇒ Stop if generalization error falls below Threshold.

★ Post pruning:

Grow entirely then reduce nodes and depth by :-

16 MON

⇒ Trim nodes of DT by bottom up approach

⇒ Generalization error ~~increase~~ ~~after~~ improve after trimming, then replace subtree by leaf node.

⇒ Replace subtree with most frequently used branch.

Notes

Q.6 ⇒ P(tough / tough) = 0.7

P(cough / tough) = 0.3

P(tough / cough) = 0.5

P(cough / cough) = 0.5

12 MON

w_1 = tough w_3 = cough w_2 = ?? w_4 = cough

Now $P(w_3 / w_1 w_2 w_4)$ = ?? [we have to find out]

Now by markov assumption, w_1 doesn't change w_2 Thus

$$P(w_3 / w_1, w_2 w_4) = P(w_3 / w_2, w_4)$$

Now now bayes Theorem suggests that

$$P(a / bc) = \frac{P(b) ac) \cdot P(a) c)}{P(b / c)}$$

March 2012

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

Now by markov assumption, $P(w_4/w_2 w_3) = P(w_3/w_3)$

(only n-1 ~~class~~ depends

Thus

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

14 WE

15 THU

Now as per question ω_2 = course
ω_4 = course

do $P(\omega_3 / \text{course}, \text{course}) = \frac{P(\text{course}/\omega_3) \cdot P(\omega_3/\text{course})}{P(\omega_4/\omega_2)}$

Let $P(\omega_4/\omega_2) = X$

↳ can't be calculated for now

$P(\text{tough}/\text{course}, \text{course}) = \frac{P(\text{course}/\text{tough}) \cdot P(\text{tough}/\text{course})}{X}$

16 FRI

$P(\text{tough}/\text{course}, \text{course}) = \frac{P(0.3)(0.5)}{X}$

$P(\text{course}/\text{course}, \text{course}) = \frac{(0.5)(0.5)}{X}$

Now as either tough or course will come then

$= P(\text{tough}/\text{course}, \text{course}) + P(\text{course}/\text{course}, \text{course}) = 1$

$= \frac{0.15}{X} + \frac{0.25}{X} = 1$

Notes $= \frac{0.40}{X} = 1 \qquad X = 0.4$

Probability That w_3 = tough

$$= \frac{0.15}{0.40} = \frac{15}{40} = 0.375$$

w_3 = course

$$= \frac{0.25}{0.40} = \frac{25}{40} = 0.625$$

Ans

Another method

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

$=$

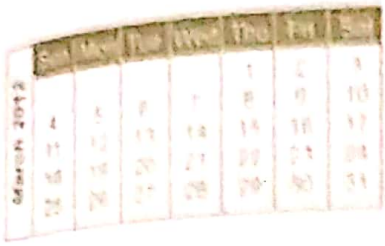
putting w_2 = course

w_4 = course

w_3 = tough

$$= \frac{(0.5)(0.3)}{(0.5)(0.8)} = 0.375$$

w_3 = course $= \frac{(0.5)(0.5)}{(0.5)(0.8)} = 0.625$



2012 February

29 WED

Q7 a) The major advantages of DT over logistic regression models are:→

1) It is easy to interpret DT, rule based models can easily be build using DT by traveling all the paths from root to leaf node for a tree.

The rules for one to classify as yes or no is basically following the internal nodes as a search tree where if you end at the leaf node, it will become sure that the ans is true / false or any other class.

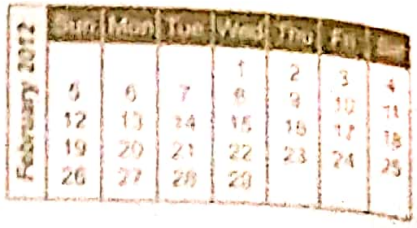
SQL queries can also be built very easily here

2) DT doesn't assume features to be independent i.e logistic regression assume all the features are independent while its not the case in DT.

b) Biggest weakness of DT:

1) DT chooses alternatives from various alternative, Thus DT is high likely

Notes



to overfit the data as seen in Q3 while training accuracy is still 1, validation accuracy is only 70%, which is not the case for logistic regression 1 THU

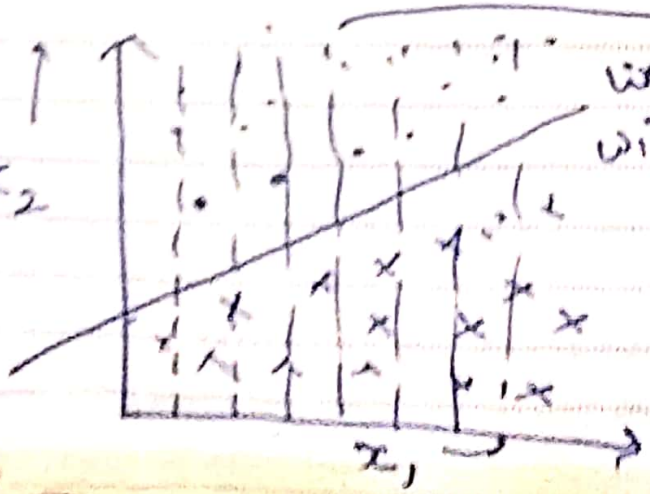
* In higher dimensional data, DT can draw only horizontal or vertical boundries while LR can chage the slope.

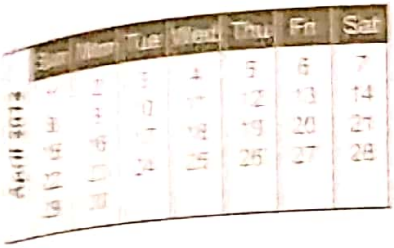
* DT doesn't always give the best line, while LR always give the best line.

c) → No marks.

2 FRI

d) Yes, the descision tree can correctly classify these vectors, we know as the data is linearly separable, there will be threshold to line $w_1 x_1 + w_2 x_2 + b > 0$





2012 March

3 SAT Upper bound on depth:

We only have to binary split at each point

Thus no. of node at i^{th} level is $= 2^i$

n points give → n leaf node in worst case for each to correctly classify.

for each leaf node, we only need one more split or one more level for x_2

4 SUN Thus depth is $O(\log n + 1) \approx O(\log n)$

e) Yes, same strategy is applicable as in (d) part.

What one can do is divide the 2D plane into $(n+1) \times (n+1)$ blocks of diff size where + 1 row or 1 column will contain 1 point in the worst case. Thus total nodes will be $O(n^2)$ = Thus depth = $\log(n^2)$

$= 2\log(n)$

Notes

March 2012

Bonus

5 MON

Q.8 $X = \langle x_1, x_2, x_3 \ldots x_n \rangle$ vector of boolean variables

assuming $P(Y=1) = \pi$ (i.e a constant)

To show: $P(Y=1/X) = \frac{1}{1+e^{(w_0 + \sum w_i x_i)}}$

Proof: let $\theta_{i1} = P(x_{i1} \mid Y=1)$

$P(x_i=0 \mid Y=1) = 1 - \theta_{i1}$

Similarly $\theta_{i0} = P(x_i=1 \mid Y=0)$

$P(x_i=0 \mid Y=0) = 1 - \theta_{i0}$

6 TUE

$$P(Y=1/X) = \frac{P(X \mid Y=1) \cdot P(Y=1)}{P(X)}$$

$$P(X \mid Y=1) \cdot P(Y=1) + P(X \mid Y=0) \cdot P(Y=0)$$

$$P(Y=1/X) = \frac{1}{1 + \frac{P(X \mid Y=0) \cdot P(Y=0)}{P(X \mid Y=1) \cdot P(Y=1)}}$$

$P(Y=1) = \pi$ $P(Y=0) = 1-\pi$

Notes

7 WED

$$= \frac{1}{1 + \frac{P(x|y=0)}{P(x|y=1)} \cdot \left(\frac{1-\pi}{\pi}\right)}$$

$$= \frac{1}{1 + \exp\left(\ln\left(\frac{P(x|y=0)}{P(x|y=1)} \cdot \left(\frac{1-\pi}{\pi}\right)\right)\right)}$$

$$= \frac{1}{1 + \exp\left(\ln\left(\frac{P(x|y=0)}{P(x|y=1)}\right) + \ln\left(\frac{1-\pi}{\pi}\right)\right)}$$

8 THU

$$\left[\because \ln(a \cdot b) = \ln a + \ln b\right]$$

$$P(x_i \mid y=0) = \theta_{i0}^{x_i} \cdot (1-\theta_{i0})^{1-x_i}$$

(Binomial distribution)

When $x_i = 1 \quad \theta_{i0} = 4$

$x_i = 0 \quad = 1 - \theta_{i0}$.

$$P(x_i \mid y=1) = \theta_{i1}^{x_i} (1-\theta_{i1})^{1-x_i}$$

$$\ln\left(\frac{P(x|y=0)}{P(x|y=1)}\right) = \sum_{i=1}^{n} \ln\left(\frac{P(x_i|y=0)}{P(x_i|y=1)}\right)$$

$\ln(a \cdot b) = \ln a + \ln b$

Notes

9 FRI

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

$$= \sum_{i=1}^{n} \ln\left(\frac{\theta_{i0}}{\theta_{i1}}^{x_i}\right) + \ln\left(\frac{1-\theta_0}{1-\theta_i}\right)^{1-x_i} \quad [\because \ln(a \cdot b) = \ln a + \ln b]$$

$$= \sum_{i=1}^{n} \ln\left(\frac{\theta_{i0}}{\theta_{i1}}\right)^{x_i} + \ln\left(\frac{1-\theta_{0i}}{1-\theta_{i1}}\right)^{1-x_i}$$

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

$[\because \ln(a^k) = k \ln a]$

10 SAT

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

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

$$P(Y=1/X) = \frac{1}{1 + \exp\left(\ln\left(\frac{1-\pi}{\pi}\right) + \sum_{i=1}^{n} \ln\left(\frac{1-\theta_{i0}}{1-\theta_{i1}}\right) + \sum_{i=1}^{n} x_i \ln\left(\frac{\theta_{i0}}{\theta_{i1}}\right) - \ln\left(\frac{1-\theta_{i0}}{1-\theta_i}\right)\right)}$$

w_0

w_0'

Notes

April 2012

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

2012 March

11 SUN $P(y=1/x) = \frac{1}{1+\exp(\omega_0 + \sum_{i=1}^{m} \omega_i x_i)}$