به نام خدا

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده برق و کامپیوتر

درس سیستم‌های هوشمند

تمرین شماره سوم

نام و نام خانوادگی:

اشکان جعفری فشارکی

شماره دانشجویی:

۸۱۰۱۹۷۴۸۳

آذر ۱۴۰۰

فهرست سوالات

Question#1

A)

In this Part, we calculate Information Gain for each feature and then we do it again in every branch. Brief formula is shown in Figure1.

$$I.G(M, A_i) = E(M) - E(M, A_i)$$

$$E(M, A_i) = \sum_j P(A_i = j)\, E(M|j)$$

Figure1. Entropy and Information Gain Formula

The procedure and explanation of calculation is shown below.

First, we label every feature for simplicity:

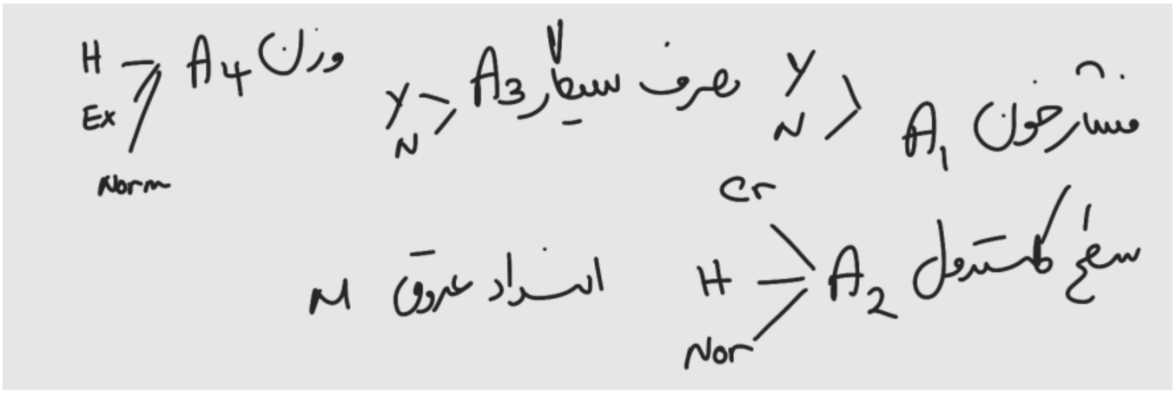
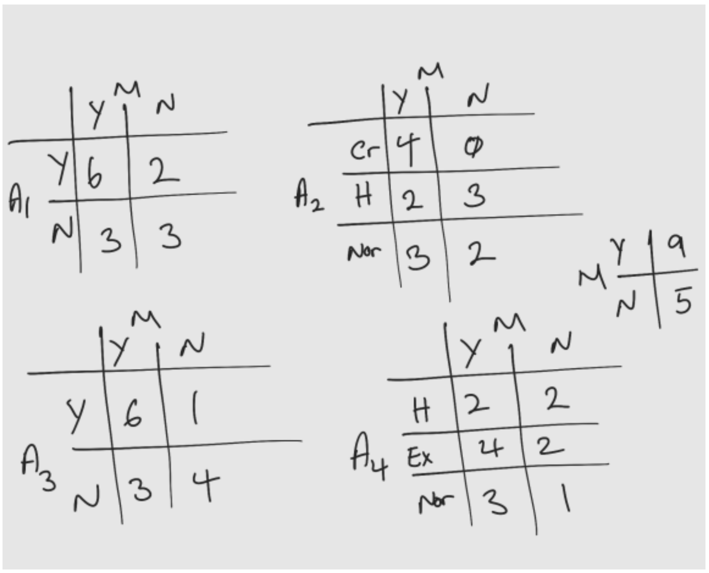


Figure2. labeling each feature

Second as we look for the best feature in order to choose it for the root, we calculate every feature's Information Gain. The process is as below:



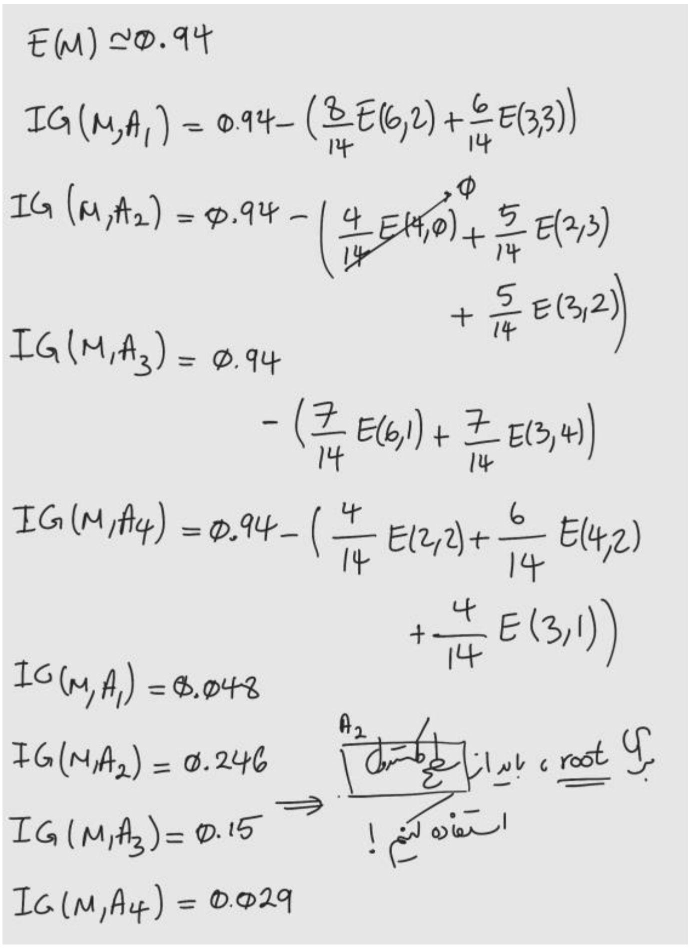
A_1	M: Y	M: N
Y	6	2
N	3	3

A_2	M: Y	M: N
Cr	4	$\emptyset$
H	2	3
Nor	3	2

M	
Y	9
N	5

A_3	M: Y	M: N
Y	6	1
N	3	4

A_4	M: Y	M: N
H	2	2
Ex	4	2
Nor	3	1



$$E(M) \simeq 0.94$$

$$IG(M,A_1) = 0.94 - \left(\frac{8}{14}E(6,2) + \frac{6}{14}E(3,3)\right)$$

$$IG(M,A_2) = 0.94 - \left(\cancel{\frac{4}{14}E(4,0)}^{\,0} + \frac{5}{14}E(2,3) + \frac{5}{14}E(3,2)\right)$$

$$IG(M,A_3) = 0.94 - \left(\frac{7}{14}E(6,1) + \frac{7}{14}E(3,4)\right)$$

$$IG(M,A_4) = 0.94 - \left(\frac{4}{14}E(2,2) + \frac{6}{14}E(4,2) + \frac{4}{14}E(3,1)\right)$$

$$IG(M,A_1) = 0.048$$

$$IG(M,A_2) = 0.246$$

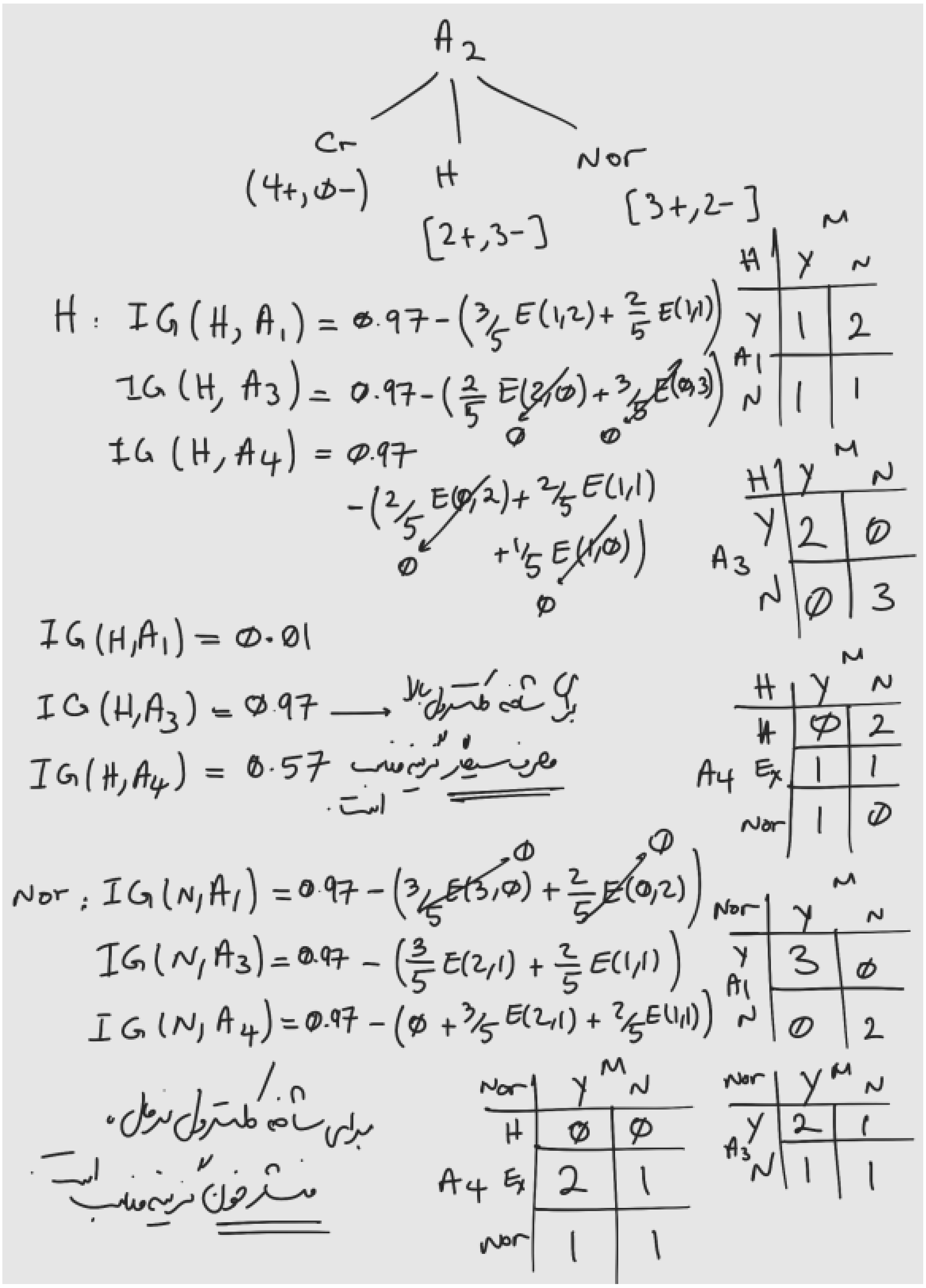
$$IG(M,A_3) = 0.15 \Rightarrow$$

$$IG(M,A_4) = 0.029$$

برای root، باید از $\overset{A_2}{\boxed{\text{سطح کلسترول}}}$ استفاده کنم!

According to calculated IGs (Information Gains), Colestrol Level is the best choise for the root.

In the following steps we do same procedure to find the best feature for the branch classifiers:



$$\begin{tikzcd}
 & A_2 \arrow[dl, dash] \arrow[d, dash] \arrow[dr, dash] & \\
Cr\ (4+,0-) & H\ [2+,3-] & Nor\ [3+,2-]
\end{tikzcd}$$

$H: IG(H,A_1) = 0.97 - \left(\frac{3}{5}E(1,2) + \frac{2}{5}E(1,1)\right)$

$IG(H,A_3) = 0.97 - \left(\frac{2}{5}\cancel{E(2,0)}^{0} + \frac{3}{5}\cancel{E(0,3)}^{0}\right)$

$IG(H,A_4) = 0.97 - \left(\frac{2}{5}\cancel{E(0,2)}^{0} + \frac{2}{5}E(1,1) + \frac{1}{5}\cancel{E(1,0)}^{0}\right)$

H		M: Y	N
A_1	Y	1	2
	N	1	1

H		M: Y	N
A_3	Y	2	0
	N	0	3

$IG(H,A_1) = 0.01$

$IG(H,A_3) = 0.97 \longrightarrow$ برای شاخه کلسترول بالا

$IG(H,A_4) = 0.57$ مصرف سیگار بهترین انتخاب است.

H		M: Y	N
A_4	H	0	2
	Ex	1	1
	Nor	1	0

$Nor: IG(N,A_1) = 0.97 - \left(\frac{3}{5}\cancel{E(3,0)}^{0} + \frac{2}{5}\cancel{E(0,2)}^{0}\right)$

$IG(N,A_3) = 0.97 - \left(\frac{3}{5}E(2,1) + \frac{2}{5}E(1,1)\right)$

$IG(N,A_4) = 0.97 - \left(0 + \frac{3}{5}E(2,1) + \frac{2}{5}E(1,1)\right)$

Nor		M: Y	N
A_1	Y	3	0
	N	0	2

برای شاخه کلسترول نرمال، مصرف چربی بهترین انتخاب است.

Nor		M: Y	N
A_4	H	0	0
	Ex	2	1
	Nor	1	1

Nor		M: Y	N
A_3	Y	2	1
	N	1	1

At the end, the Decision Tree was designed properly and is shown in Figure3.

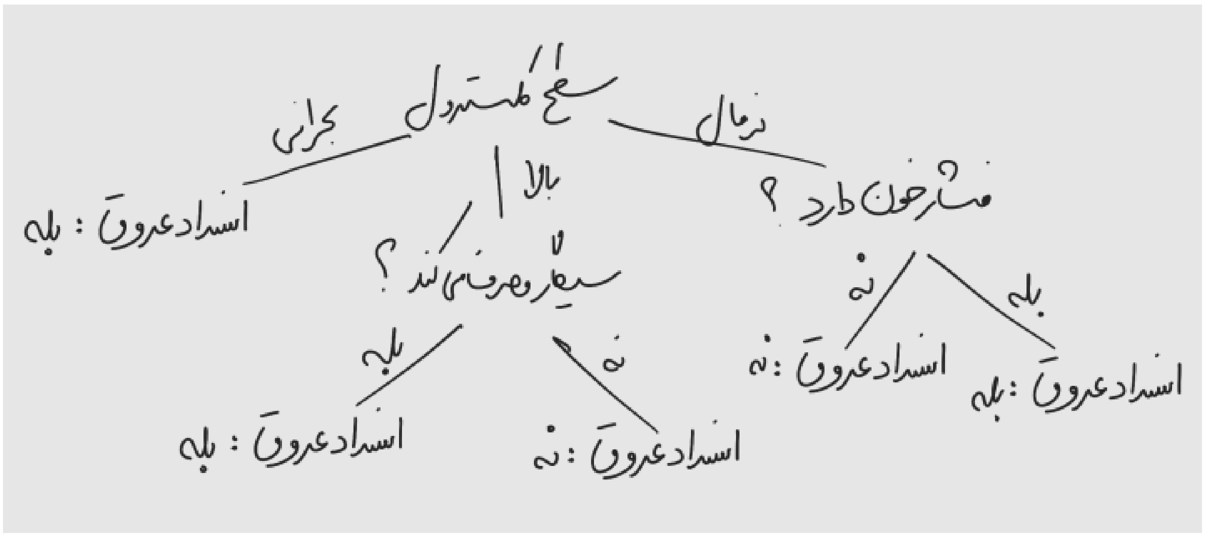
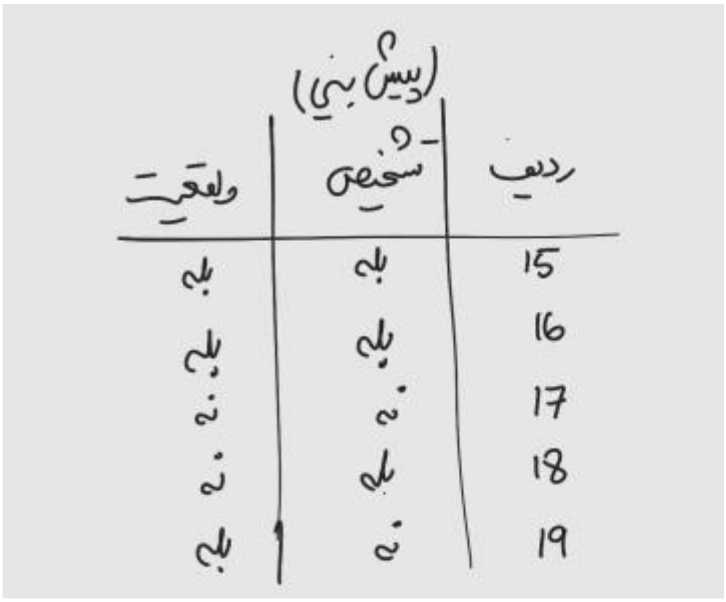


Figure3. Final Decision Tree

B)

First of all, we predict test data manually as is in Figure4.

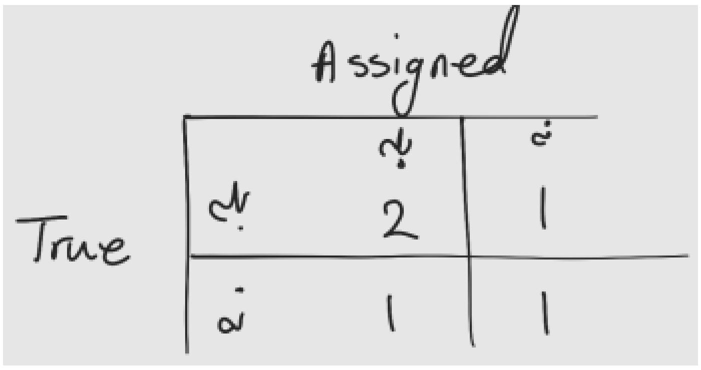


ردیف	تشخیص (پیش بینی)	واقعیت
15	بله	بله
16	بله	بله
17	نه	نه
18	بله	نه
19	نه	بله

Figure3. Prediction Table

Due to Figure 4 and predicted labels, confusion matrix is shown in Figure5.

This Decision-Tree accuracy is about: 0.6



		Assigned	
		بله	نه
True	بله	2	1
	نه	1	1

Figure5. Confusion Matrix

C)

Decision trees are prone to overfitting, especially when a tree is particularly deep. This is due to the amount of specificity we look at leading to smaller sample of events that meet the previous assumptions. This small sample could lead to unsound conclusions.

There are 2 methods to avoid this problem but as you know there is no free-lunch:

1) post-prune

   In this case we grow our tree deep and complete. At the end we start to prune the tree all over the levels which do not classify data significantly.

2) pre-prune

   In this case we start to grow but stop when there is no significant classifying.

The side-effect of these methods is high bias and bigger error!

Question 2

A)

In this part we implement ID3 Algorithm and design a decision tree in which the target feature is Recidivism.

We implement several functions which are:

def Ent(data, clsf):

Calculating Entropy

def Information_Gain (featName, data, clsf):

Calculating Information Gain of the given feature using Entropy

def Max_Info_Feature(data, clsf_list):

Finding which feature has the most information to help us for spliting

def Raw_Tree(featName, data, clsf, depth):

Initializing the tree and splitting data based on the given feature. This function is used in Recurv_ID3.

def Recurv_ID3(root, prev_feat_val, data, clsf, depth):

Recursively implement the ID3 Algorithm till our desire depth is satisfied.

def Generate_Tree(data):

Just for simplicity and calling the Recurv_ID3 with the proper input class

def Database_Spliting(data):

splitting data in 80-20 order

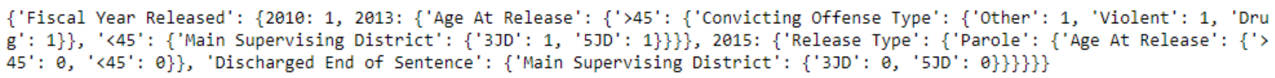
def Predict(features, tree)

def Confusion_Matrix(prediction, targets)

def Test_ConfusionMatrix(test, tree):

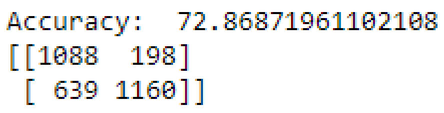
Calculating the Accuracy and calling Confusion_Matrix

Final Decision-Tree is as below:



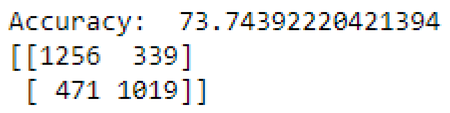
```
{'Fiscal Year Released': {2010: 1, 2013: {'Age At Release': {'>45': {'Convicting Offense Type': {'Other': 1, 'Violent': 1, 'Dru
g': 1}}, '<45': {'Main Supervising District': {'3JD': 1, '5JD': 1}}}}, 2015: {'Release Type': {'Parole': {'Age At Release': {'>
45': 0, '<45': 0}}, 'Discharged End of Sentence': {'Main Supervising District': {'3JD': 0, '5JD': 0}}}}}}
```

The accuracy and confusion matrix for DEPTH=3,6,10 is as Figures 6,7,8.



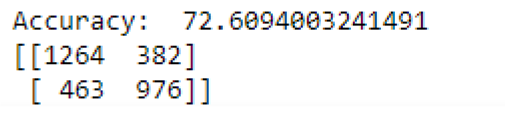
```
Accuracy:  72.86871961102108
[[1088  198]
 [ 639 1160]]
```

Figure6. Confusion Matrix and Accuracy for DEPTH=3



```
Accuracy:  73.74392220421394
[[1256  339]
 [ 471 1019]]
```

Figure7. Confusion Matrix and Accuracy for DEPTH=6



```
Accuracy:  72.6094003241491
[[1264  382]
 [ 463  976]]
```

Figure8. Confusion Matrix and Accuracy for DEPTH=10

As you see in Figures 6 to 8, It could be easily interpreted that increasing the depth of a tree will not always increases our accuracy because of reasons like overfitting, meaningless details and “True-Positives”. It is obvious that when we increase DEPTH, it means we are classifying in more details and this will not always help us!

Another reason would be realized by Confusion Matrix. As you can see in Figures 6 to 8, increasing DEPTH would not certainly helps us because it could make mistakes like choosing TN instead of TPs and this exactly the issue of increasing DEPTH.

B)

For this part and implementing Random-Forest, I used below functions:

def RandomF_Prep(data):

Splitting train data to 4 equal data inorder to run Random-Forest

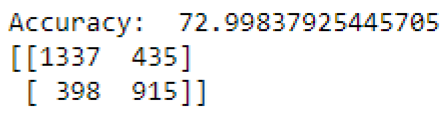
def Random_Forest(P1,P2,P3,P4):

Choosing the best label for each row by using Majority-Voting after applying ID3 algorithm on all 4 trees.

def RandomForest_Test_ConfusionMatrix(prediction,test):

Calculating the Accuracy and calling Confusion_Matrix

The result of Random-Forest is as Figure 9.



```
Accuracy:  72.99837925445705
[[1337  435]
 [ 398  915]]
```

Figure9. Confusion Matrix and Accuracy for 4-tree Random-Forest

As in Figure 9 we could see that in contrary to our expectation, there is no significant increase in accuracy!

The reason is not so simple. It needs to observe dataset.

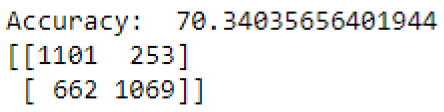
If we see the data carefully, we could easily see a feature which is significantly correlated to target label. And It is by far more correlated than the other features!

It means that Random-Forest could not helps us when there is a very correlated feature in our data to the target label because this feature absolutely will be the root of our Decision

Tree and after classifying by this feature, there would be no more significant classifying by the other features.

It is interesting that in such situation, not only Random-Forest could not help but also there is a probability that it fails to reach the Decision-Tree accuracy. The reason is that maybe Random-Forest in one its trees choose the root incorrectly.

At the end I implemented Random-Forest based on *sklearn* library and the result is shown in Figure 10, which is same as non-library one.



```
Accuracy:  70.34035656401944
[[1101  253]
 [ 662 1069]]
```

Figure10. Confusion Matrix and Accuracy for Library-base Random-Forest

Question 3

A)

In this part we implement KNN Algorithm in which the target feature (CLASS) is the first column. Functions and Class that I used are as below:

class KNN_Classifier():

all the 5 below functions are the instances of this CLASS which is my KNN classifier

def __init__(self, K) :

Nothing important. Just to use CLASS.

def fit(self, X_train, Y_train) :

Try to fit the generated model to train data

def predict(self, X_test) :

Predicting the test data by the represented model

def find_neighbors(self, x) :

Finding near neighbors by using the Euclidean function. It is used in predict function and it is actually the model we want to apply on test data

def euclidean(self, x, x_train) :

Calculating distance by Euclidean Formula

….

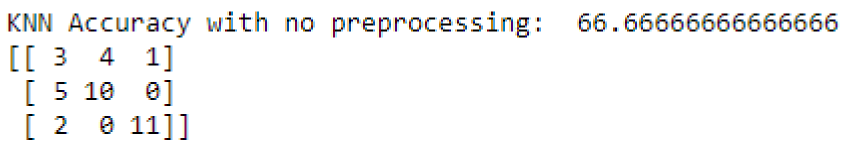
def Confusion_Matrix(prediction, targets):

def Database_Spliting(data):

splitting data in 80-20 order

def Accuracy(Y_pred,Y_test):

The result for 5-NN is as Figure11.

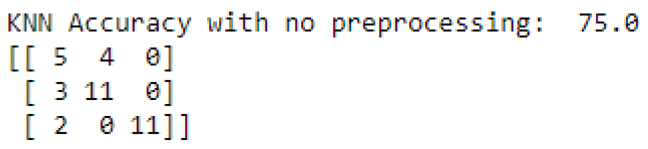


```
KNN Accuracy with no preprocessing:  66.66666666666666
[[ 3  4  1]
 [ 5 10  0]
 [ 2  0 11]]
```

Figure11. Confusion Matrix and Accuracy for No-Preprocessing 5-NN

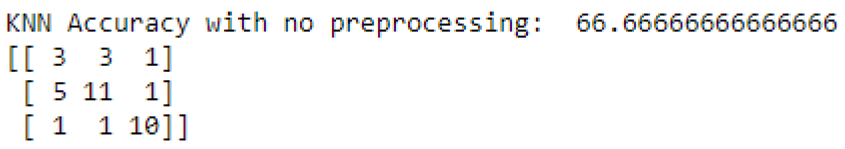
The result of Figure 11 varies between 65 to 85. And the mean accuracy I got was about 74.

In Figures 12 and 13, I try 7 and 9 neighbors to run the algorithm.



```
KNN Accuracy with no preprocessing:  75.0
[[ 5  4  0]
 [ 3 11  0]
 [ 2  0 11]]
```

Figure12. Confusion Matrix and Accuracy for No-Preprocessing 7-NN



```
KNN Accuracy with no preprocessing:  66.66666666666666
[[ 3  3  1]
 [ 5 11  1]
 [ 1  1 10]]
```

Figure13. Confusion Matrix and Accuracy for No-Preprocessing 9-NN

According to above figures, by increasing K, there is no certainty that accuracy increases. And it is because it would be meaningless to increase neighbors when there is no need for more than a specific number of neighbors.

But if we use 3-NN which is number of our classes, our accuracy will be more than 5-NN and it means changing K could help to get better accuracy but not always increasing! In some situations, decreasing K would help us.

B)

As in this part we use library functions so there is no need to explain my code.

The algorithms I used in this section was LMNN and NCA.

First, I am going to explain them a little bit.

Before that I should mention that all metric methods are going to transform the raw data to a space which finding neighbors is easier. The brief method is as the below transformer:

$$D(x,x') = \sqrt{(Lx - Lx')^T(Lx - Lx')}$$

Large Margin Nearest Neighbor (LMNN):

LMNN learns a “Mahalanobis” distance metric in the kNN classification setting. The learned metric attempts to keep close k-nearest neighbors from the same class, while keeping examples from different classes separated by a large margin. For this aim this algorithm tries to solve below optimization problem:

$$min_L \sum_{i,j} \eta_{ij} ||L(x_i - x_j)||^2 + c \sum_{i,j,l} \eta_{ij} (1 - y_{ij})[1 + |L(x_i - x_j)|^2 - ||L(x_i - x_l)||^2]_+$$

in which x_i, x_j and x_l are train, neighbor and all data.

Neighborhood Components Analysis (NCA)

NCA is a distance metric learning algorithm which aims to improve the accuracy of nearest neighbors’ classification compared to the standard Euclidean distance. The algorithm directly maximizes a stochastic variant of the leave-one-out k-nearest neighbors (KNN) score on the training set. It can also learn a low-dimensional linear transformation of data that can be used for data visualization and fast classification. The objective we want to maximize is the expected number of points correctly classified under this scheme:

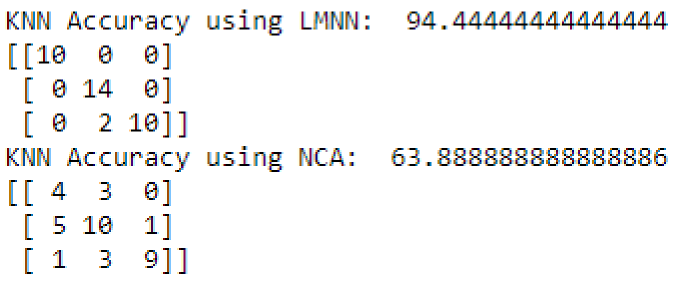
$$f(A) = \sum_i \sum_{j \in C_i} p_{ij} = \sum_i p_i$$

Where p_{ij} is defined using softmax over Euclidean distances in the transformed space:

$$p_{ij} = \frac{\exp(-||Ax_i - Ax_j||^2)}{\sum_{k \neq i} \exp(-||Ax_i - Ax_k||^2)}$$

Attention that the above Formula is the applied softmax on “Mahalanobis” distance.

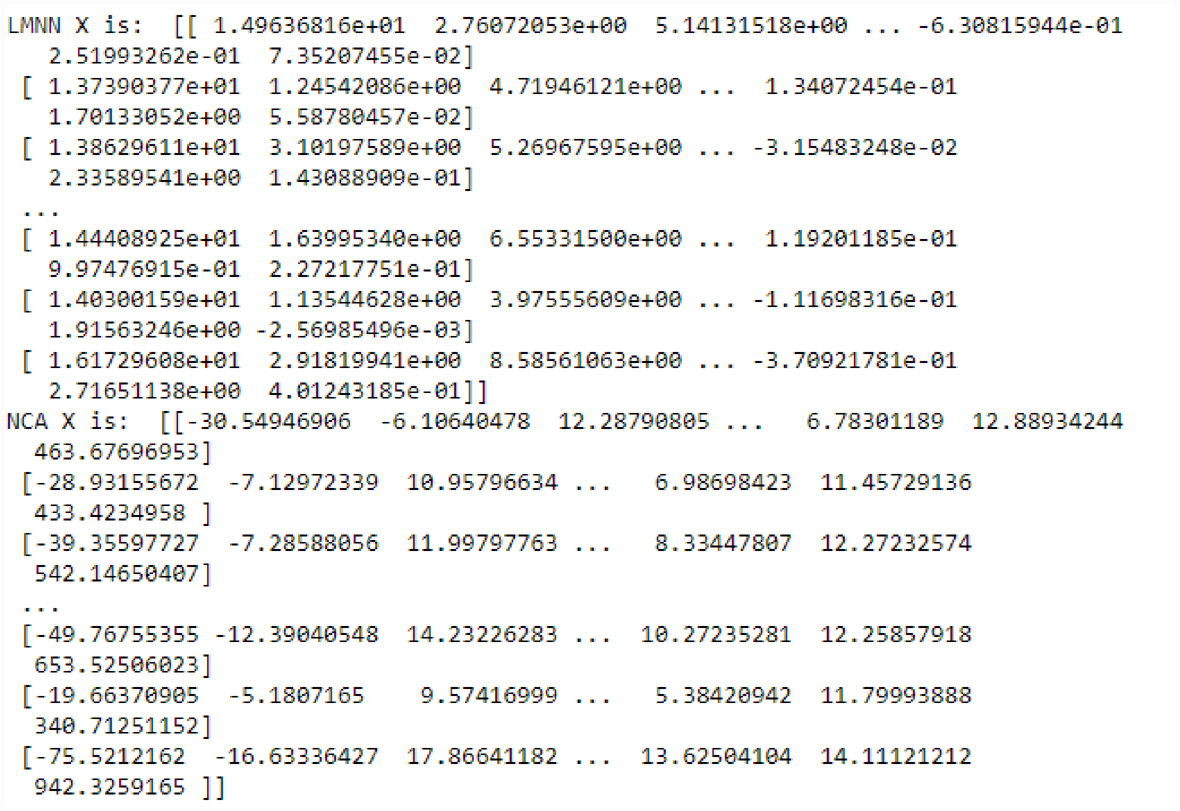
In figure14, accuracy and confusion matrix of KNN using LMNN and NCA preprocessing is shown.



```
KNN Accuracy using LMNN:  94.44444444444444
[[10  0  0]
 [ 0 14  0]
 [ 0  2 10]]
KNN Accuracy using NCA:  63.888888888888886
[[ 4  3  0]
 [ 5 10  1]
 [ 1  3  9]]
```

Figure14. Confusion Matrix and Accuracy for LMNN and NCA 5-NN

In figure15, transformed train data is shown both in LMNN and NCA algorithm:



```
LMNN X is:  [[ 1.49636816e+01  2.76072053e+00  5.14131518e+00 ... -6.30815944e-01
   2.51993262e-01  7.35207455e-02]
 [ 1.37390377e+01  1.24542086e+00  4.71946121e+00 ...  1.34072454e-01
   1.70133052e+00  5.58780457e-02]
 [ 1.38629611e+01  3.10197589e+00  5.26967595e+00 ... -3.15483248e-02
   2.33589541e+00  1.43088909e-01]
 ...
 [ 1.44408925e+01  1.63995340e+00  6.55331500e+00 ...  1.19201185e-01
   9.97476915e-01  2.27217751e-01]
 [ 1.40300159e+01  1.13544628e+00  3.97555609e+00 ... -1.11698316e-01
   1.91563246e+00 -2.56985496e-03]
 [ 1.61729608e+01  2.91819941e+00  8.58561063e+00 ... -3.70921781e-01
   2.71651138e+00  4.01243185e-01]]
NCA X is:  [[-30.54946906  -6.10640478  12.28790805 ...   6.78301189  12.88934244
  463.67696953]
 [-28.93155672  -7.12972339  10.95796634 ...   6.98698423  11.45729136
  433.4234958 ]
 [-39.35597727  -7.28588056  11.99797763 ...   8.33447807  12.27232574
  542.14650407]
 ...
 [-49.76755355 -12.39040548  14.23226283 ...  10.27235281  12.25857918
  653.52506023]
 [-19.66370905  -5.1807165    9.57416999 ...   5.38420942  11.79993888
  340.71251152]
 [-75.5212162  -16.63336427  17.86641182 ...  13.62504104  14.11121212
  942.3259165 ]]
```

Figure15. LMNN and NCA Transformed Train Data

<u>Last part:</u>

According to above explanation LMNN is by far the better method in accuracy but not in speed and calculations!

As it is shown in Figure16 LMNN and NCA are compared through different K.

It can be concluded that increasing K would decrease the accuracy very little but it is not a solid conclusion and, in some situations, increasing K would nor necessarily decreases the accuracy. [NEXT PAGE]

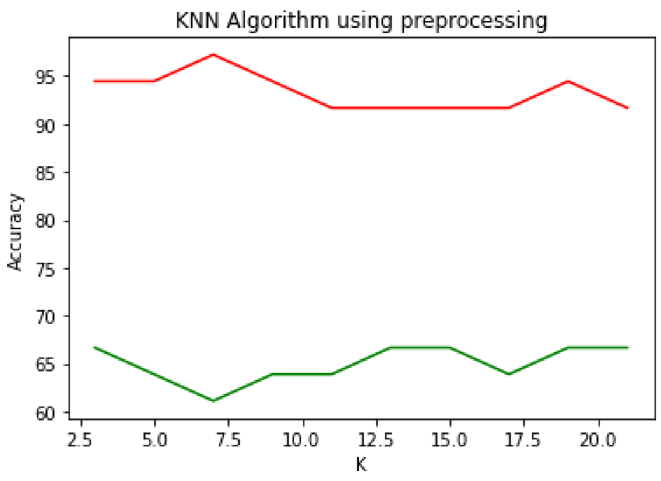


Figure16. Comparing LMNN and NCA in KNN over values of K

References: