HW2 Intelligent Systems

Supervised Machine Learning

Fardin Abbasi 810199456

School of Electrical & Computer Engineering

College of Engineering

University of Tehran

Fall 2023

Questions

Q1: Decision Tree

A: Designing the classifier

Here is the pseudocode of ID3 algorithm:

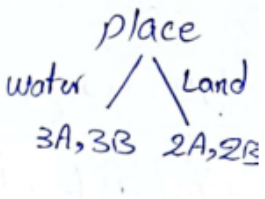
```
ID3(Examples, Attributes, TargetAttribute)

Create RootNode for the tree
if all members of Examples are in the same class C
        then RootNode = single-node tree with label = C
else if Attributes is empty
        then RootNode = single-node tree with label = most common value of Target_attribute in
                                                      Examples;
else
        A = element in Attributes that maximizes InformationGain(Examples, A)
A is decision attribute for RootNode
for each possible value v of A
        add a Branch below RootNode, testing for A = v
        Examples_v = subset of Examples with A = v
        if Examples_v is empty
                then below Branch add Leaf with label = most common value
                                                        of Target_attribute in Examples;
        else
                below Branch add Subtree ID3(Examples_v, Attributes - {A}, TargetAttribute);

return RootNode;
```

$H(y) = -0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$

Attribute 1: Place

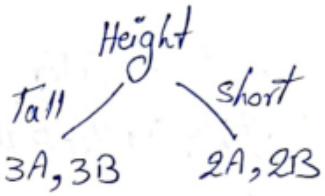


$$H(y|x) = \frac{|D_0|}{|D|} H(y|x=0) + \frac{|D_1|}{|D|} H(y|x=1)$$

$$= 0,6(-0,5 \log 0,5 - 0,5 \log 0,5) + 0,4(-0,5 \log 0,5 - 0,5 \log 0,5) = 1$$

$$I(y;x) = H(y) - H(y|x) = 0$$

Attribute 2: Height

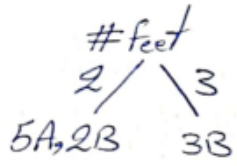


$$H(y|x) = \frac{|D_0|}{|D|} H(y|x=0) + \frac{|D_1|}{|D|} H(y|x=1)$$

$$= 0,6(-0,5 \log 0,5 - 0,5 \log 0,5) + 0,4(-0,5 \log 0,5 - 0,5 \log 0,5) = 1$$

$$I(y;x) = H(y) - H(y|x) = 0$$

Attribute 3: #Feet

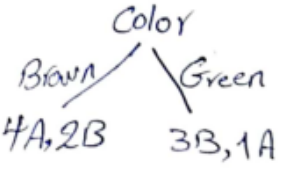


$$H(y|x) = \frac{|D_0|}{|D|} H(y|x=2) + \frac{|D_1|}{|D|} H(y|x=3)$$

$$= 0,7\left(-\frac{5}{7} \log \frac{5}{7} - \frac{2}{7} \log \frac{2}{7}\right) + 0,3(-\log 1) = 0,6$$

$$I(y;x) = H(y) - H(y|x) = 0,4$$

Attribute 4: Color



$$H(y|x) = \frac{|D_0|}{|D|} H(y|x=0) + \frac{|D_1|}{|D|} H(y|x=1)$$

$$= 0,6\left(-\frac{1}{3} \log \frac{1}{3} - \frac{2}{3} \log \frac{2}{3}\right) + 0,4\left(-\frac{3}{4} \log \frac{3}{4} - \frac{1}{4} \log \frac{1}{4}\right) = 0,875$$

$$I(y;x) = 0,125$$

Therefore the root node is #feet attribute.

$$H(y|x_1 = 2) = -\frac{5}{7}\log_2\frac{5}{7} - \frac{2}{7}\log_2\frac{2}{7} = 0.863$$

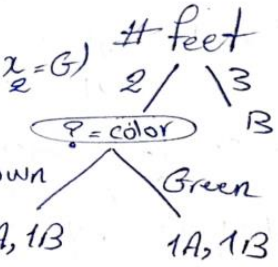
Attribute 1: Color

$$H(y|x_1=2, x_2) = \frac{|D_B|}{|D|} H(y|x_1=2, x_2=B) + \frac{|D_G|}{|D|} H(y|x_1=2, x_2=G)$$

$$= \frac{5}{7}\left(-\frac{4}{5}\log\frac{4}{5} - \frac{1}{5}\log\frac{1}{5}\right) + \frac{2}{7}(-0,5\log 0,5 - 0,5\log 0,5)$$

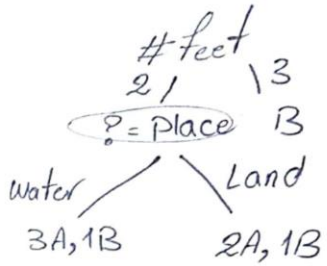
$$= 0,8$$

$$I(y; x_1=2, x_2) = H(y|x_1=2) - H(y|x_1=2, x_2) = 0,863 - 0,8 = 0,063$$



Attribute 2: Place

$$I(y; x_1=2, x_2) = H(y|x_1=2) - H(y|x_1=2, x_2)$$



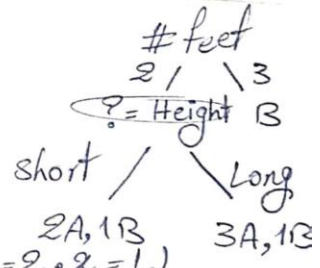
$$H(y|x_1=2, x_2) = \frac{|D_W|}{|D|} H(y|x_1=2, x_2=W) + \frac{|D_L|}{|D|} H(y|x_1=2, x_2=L)$$

$$= \frac{4}{7}\left(-\frac{3}{4}\log\frac{3}{4} - \frac{1}{4}\log\frac{1}{4}\right) + \frac{3}{7}\left(-\frac{2}{3}\log\frac{2}{3} - \frac{1}{3}\log\frac{1}{3}\right) = 0,857$$

$$I(y; x_1=2, x_2) = 0,863 - 0,857 = 0,006$$

Attribute 3: Height

$$I(y; x_1=2, x_2) = H(y|x_1=2) - H(y|x_1=2, x_2)$$



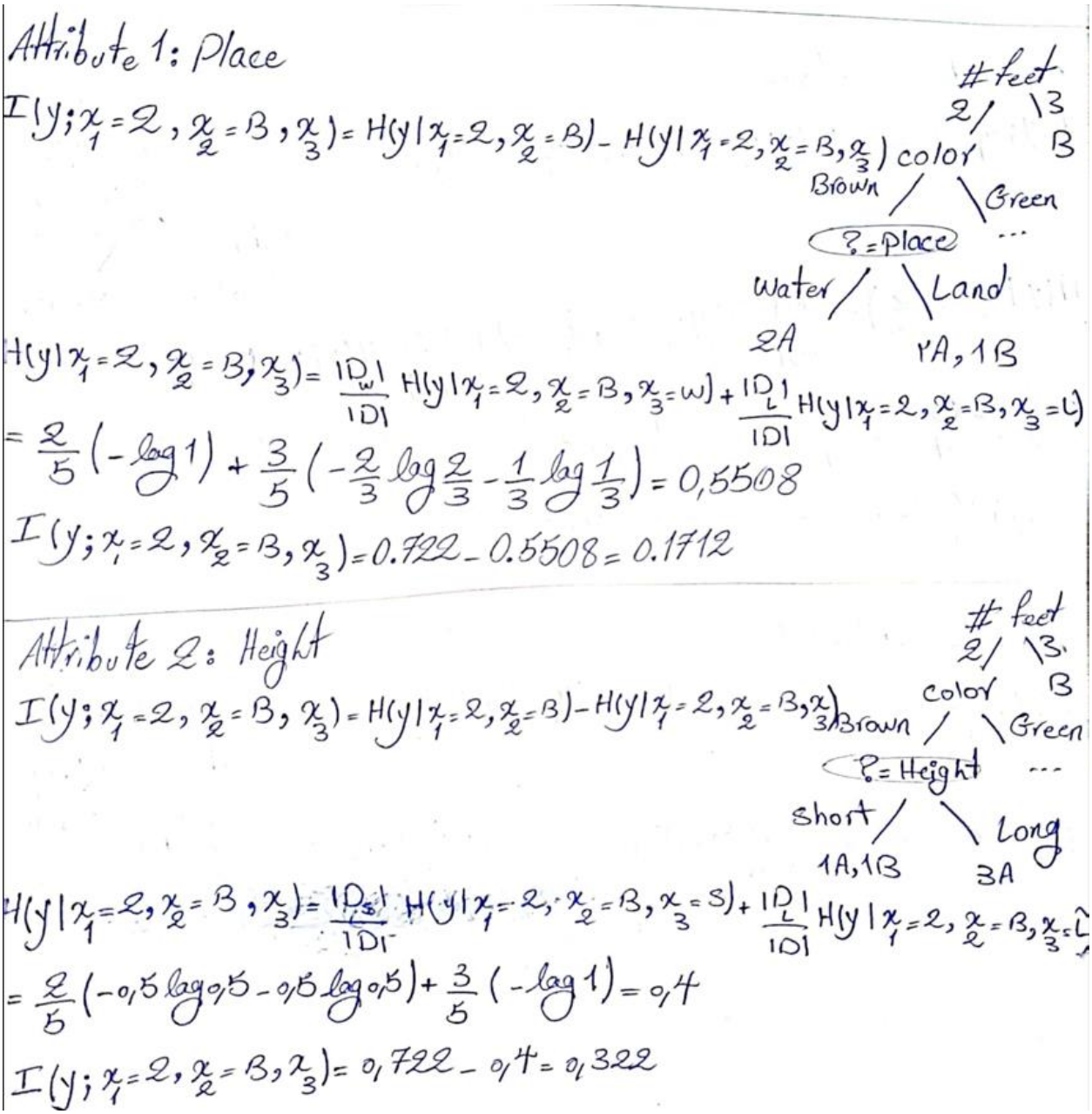
$$H(y|x_1=2, x_2) = \frac{|D_S|}{|D|} H(y|x_1=2, x_2=S) + \frac{|D_L|}{|D|} H(y|x_1=2, x_2=L)$$

$$= \frac{3}{7}\left(-\frac{2}{3}\log\frac{2}{3} - \frac{1}{3}\log\frac{1}{3}\right) + \frac{4}{7}\left(-\frac{3}{4}\log\frac{3}{4} - \frac{1}{4}\log\frac{1}{4}\right) = 0,857$$

$$I(y; x_1=2, x_2) = 0,863 - 0,857 = 0,006$$

Therefore the next node is color. Now we find the next node when color is brown.

$$H(y|x_1 = 2, x_2 = B) = -\frac{4}{5}\log_2\frac{4}{5} - \frac{1}{5}\log_2\frac{1}{5} = 0.722$$

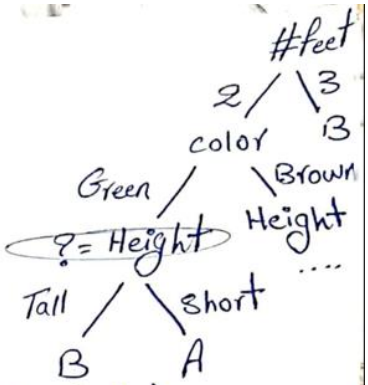


So the next node is height.

Now we find the next node when color is green.

$$H(y|x_1 = 2, x_2 = G) = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1$$

Attribute 1: Height



$$I(y; x_1=2, x_2=G, x_3) = H(y|x_1=2, x_2=G) - H(y|x_1=2, x_2=G, x_3)$$

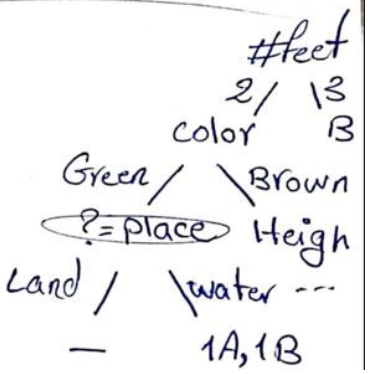
$$H(y|x_1=2, x_2=G, x_3) = \frac{|D_T|}{|D|} H(y|x_1=2, x_2=G, x_3=T) + \frac{|D_S|}{|D|} H(y|x_1=2, x_2=G, x_3=S)$$

$$= 0{,}5(-\log 1) + 0{,}5(-\log 1) = 0$$

$$I(y; x_1=2, x_2=G, x_3) = 1 - 0 = 1$$

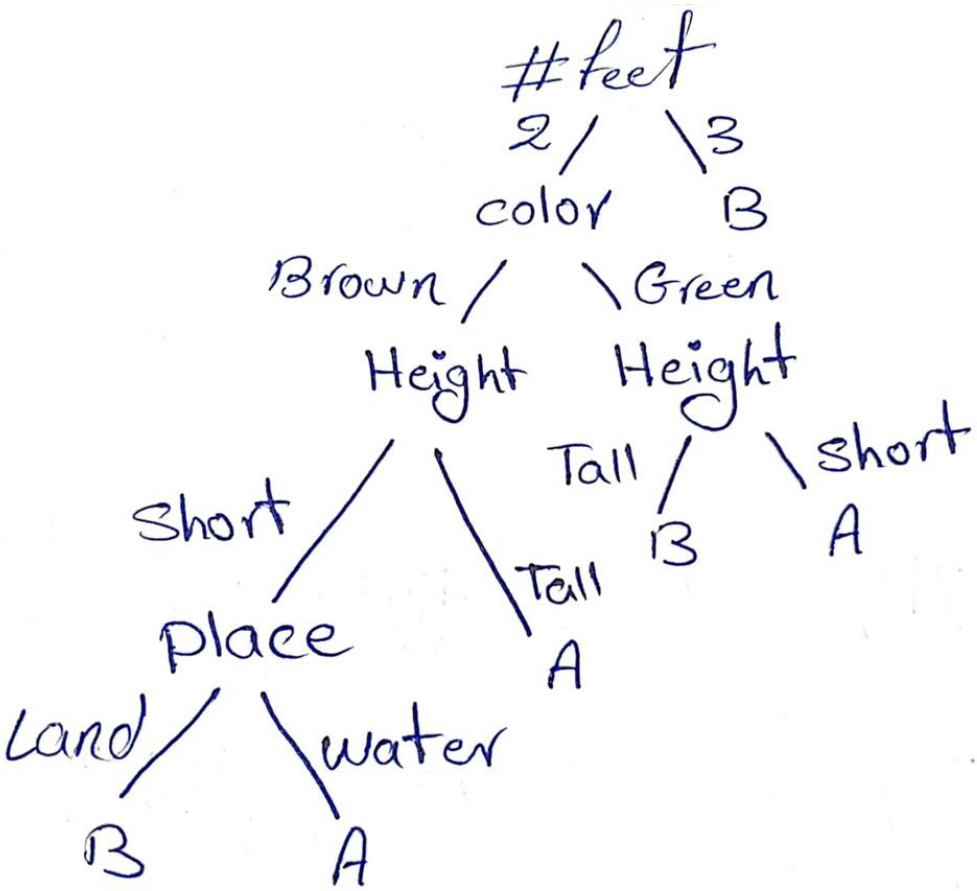
Attribute 2: place

$$I(y; x_1=2, x_2=G, x_3) = 1 - 1 = 0$$



Therefore the next node is height.

The final decision tree is depicted below.



B: Testing the classifier

#	Predicted species	True species
1	A	A
2	B	B
3	B	B
4	B	B
5	A	A

Confusion matrix		True species	
		A	B
Predicted species	A	2	0
	B	0	3

C: ID3 with greedy approach

Since ID3 is a greedy algorithm, the best attributes are chosen in each node. Therefore #feet and color are 2 best attributes but they're not able to classify all train data with zero error.

D: More robust!

Since decision trees fully grow to pure leaf, they may lose their generalization capabilities and are not robust to overfitting!

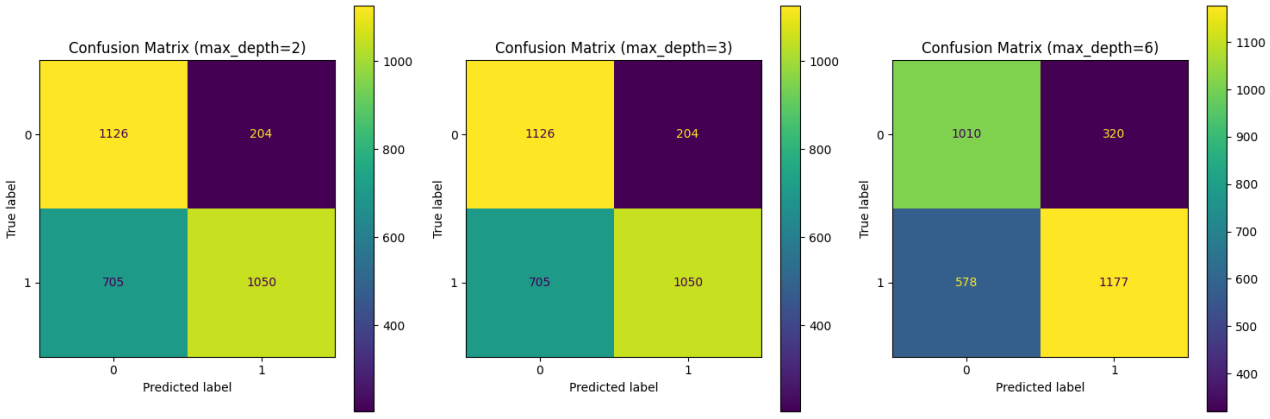
Here are some solutions to avoid overfitting:

$$\begin{cases} \boldsymbol{Pre-pruning} \begin{cases} max\_depth: Maximum\ depth\ of\ the\ tree \\ min\_samples\_leaf: Minimum\ number\ of\ samples\ required\ at\ a\ leaf\ node \\ min\_samples\_split: Minimum\ number\ of\ samples\ required\ to\ split\ an\ internal\ node \end{cases} \\ \boldsymbol{Post-pruning}\ or\ \boldsymbol{backward\ pruning}: \text{Minimum Description Length (MDL)} \\ \boldsymbol{Ensemble}: Random\ Forest \end{cases}$$

Q2: Decision tree Vs. Random forest

A: Implement classifier

Here are classification results of a decision tree classifier with different *max_depth* values:

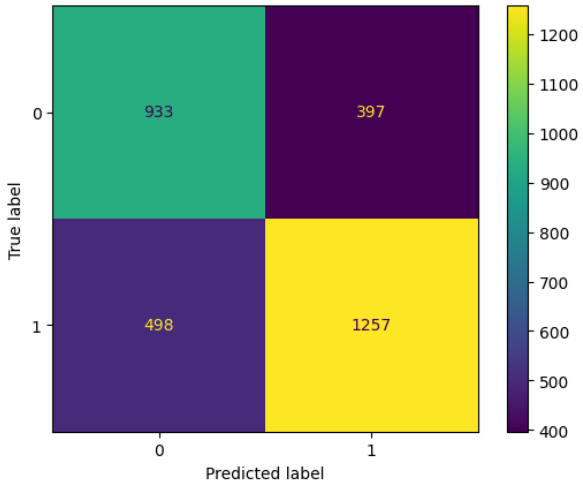


Positive label\max_depth	2	3	6
0	0.61	0.61	0.64
1	0.84	0.84	0.79

It can be inferred when tree grows more than optimal value (here 3), it overfits.

B: Random forest from scratch

Here are the classification results of a random forest with *num_trees = 3*, and each tree has a randomly assigned *max_length*.



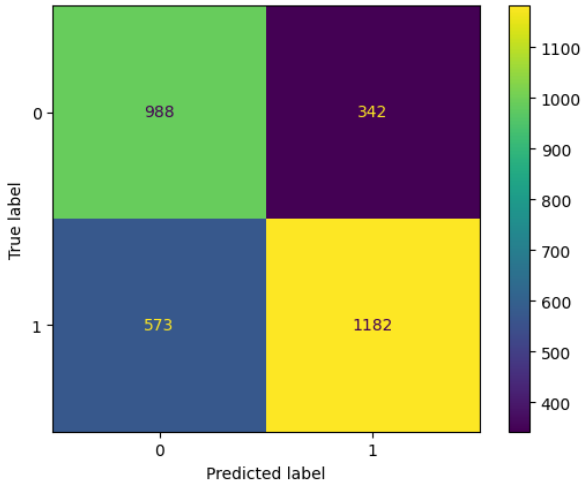

```
num_trees=3

   Precision
0       0.65
1       0.76
```

It slightly outperforms decision tree model since it votes the prediction of 3 trees and is more robust to overfitting.

C: Random forest with scikit-learn

The classification result of a built-in random forest model with *max_depth=3* is depicted below:



	precision	recall	f1-score	support
0	0.63	0.74	0.68	1330
1	0.78	0.67	0.72	1755
accuracy			0.70	3085
macro avg	0.70	0.71	0.70	3085
weighted avg	0.71	0.70	0.70	3085

Its result is nearly similar to the forementioned random forest which is implemented from scratch.

Q3: KNN classifier and metric-learning

A&B: KNN classifier

Here is the pseudocode of KNN classifier.

Algorithm The k-nearest neighbors classification algorithm

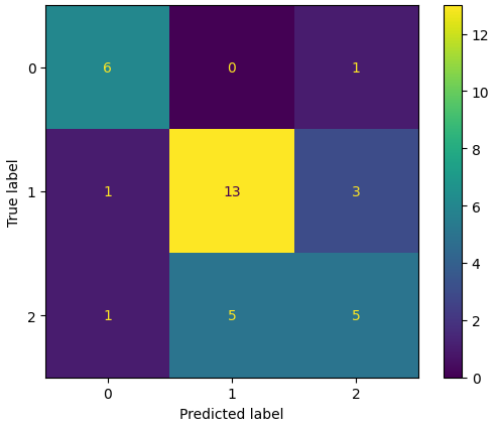
Input:

D: a set of training samples $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$
k: the number of nearest neighbors
$d(\mathbf{x}, \mathbf{y})$: a distance metric
$\mathbf{x}$: a test sample

1: **for each** training sample $(\mathbf{x}_i, y_i) \in D$ **do**
2: Compute $d(\mathbf{x}, \mathbf{x}_i)$, the distance between $\mathbf{x}$ and $\mathbf{x}_i$
3: Let $N \subseteq D$ be the set of training samples with the k smallest distances $d(\mathbf{x}, \mathbf{x}_i)$
4: **return** the majority label of the samples in N

The confusion matrix and precision of KNN classifier with different K is shown below:

K=1:



	Precision
0	0.75
1	0.72
2	0.56

K=[5,10,20]:

K=5

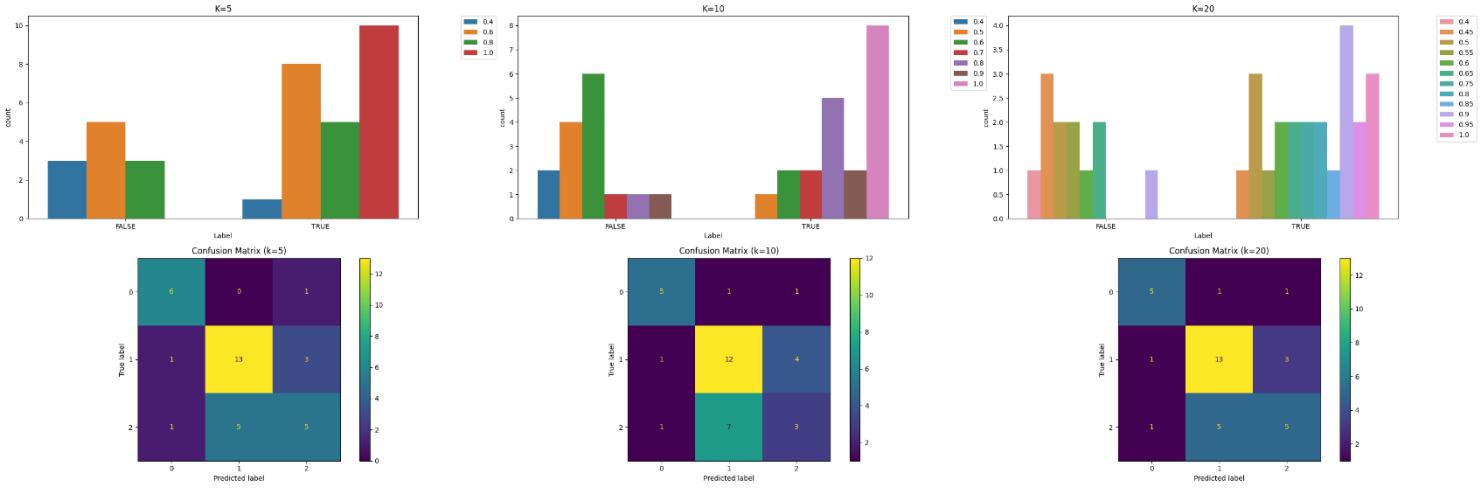
	Precision
0	0.75
1	0.72
2	0.56

K=10

	Precision
0	0.71
1	0.60
2	0.38

K=20

	Precision
0	0.71
1	0.68
2	0.56



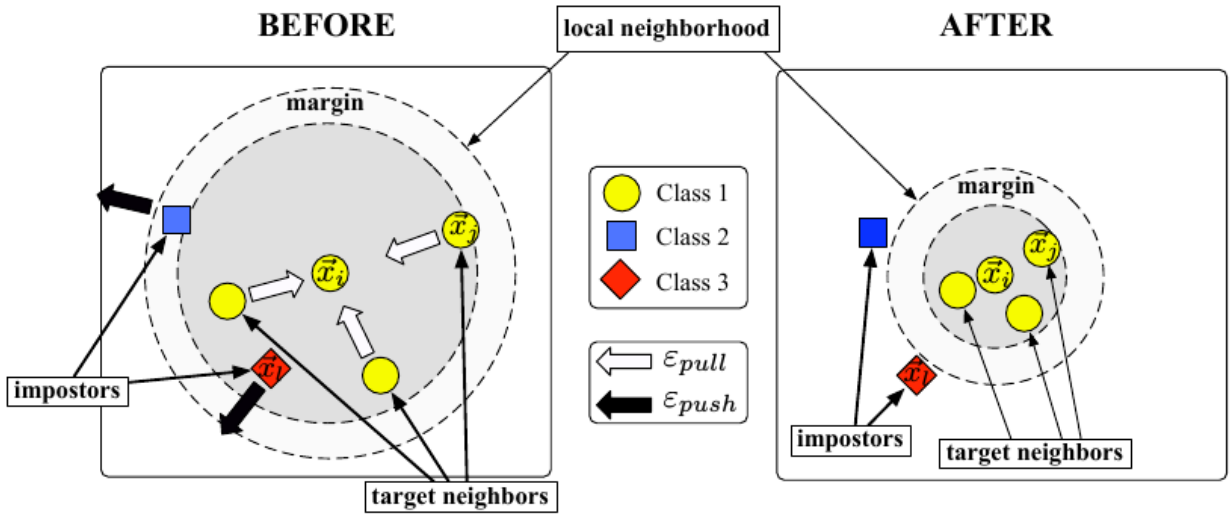
I've presented the counts of various probabilities of classification and misclassification. As you can see with k=20 the count of higher probabilities is higher therefore is expected this model performs better which is confirmed by its precision.

C: Metric-learning

1. LMNN & LFDA Vs. KNN

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. This algorithm makes no assumptions about the distribution of the data.



Based on intuition and terminology above, we can define loss functions of LMNN:

$$\epsilon_{pull}(L) = \sum_{i \sim j} \left|\left|L(x_i - x_j)\right|\right|^2$$

$$\epsilon_{push}(L) = \sum_{i,j \sim i} \sum_{l} (1 - y_{il}) \left[1 + \left|\left|L(x_i - x_j)\right|\right|^2 - \left|\left|L(x_i - x_l)\right|\right|^2\right]_+$$

$$\epsilon(L) = (1 - \mu)\epsilon_{pull}(L) + \mu\epsilon_{push}(L)$$

where j $\sim$ i denotes j is target neighbors of i, y_{il} is indicator denoting whether x_i and x_l in same class, $[z]_+ = \max(z, 0)$ is hinge loss.

Local Fisher Discriminant Analysis (LFDA)

Local Fisher Discriminant Analysis (LFDA) is one of the best-performing supervised dimensionality reducing metric learning method among the others. It is a linear supervised dimensionality reduction method. It is particularly useful when dealing with multi-modality, where one or more classes consist of separate clusters in input space.

LFDA is inspired by both Fisher Discriminant Analysis (FDA) and Local Preserving Projection (LPP). [Read More]

The local within- and between-class scatter matrix is defined as:

$$S^{(w)} = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij}^{(w)}(x_i - x_j)(x_i - x_j)^T$$

$$S^{(b)} = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij}^{(b)}(x_i - x_j)(x_i - x_j)^T$$

Where $W_{ij}^{(w)} = \begin{cases} \frac{A_{ij}}{n_l}, & y_i = y_j = l \\ 0, & O.W \end{cases}$ & $W_{ij}^{(b)} = \begin{cases} A_{ij}(\frac{1}{n} - \frac{1}{n_l}), & y_i = y_j = l \\ \frac{1}{n}, & O.W \end{cases}$

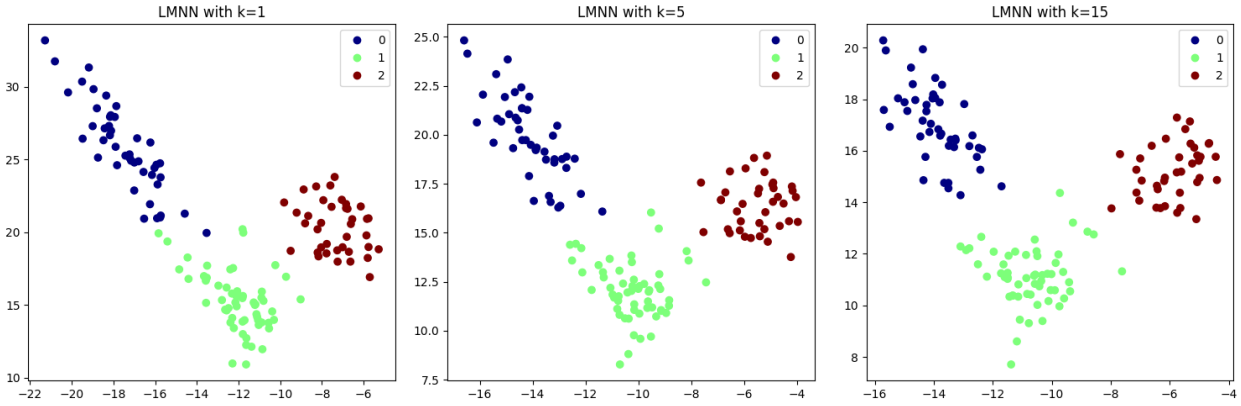
The LFDA transformation matrix L is then defined as the following:

$$L_{LFDA} = arg \max_{L \in R^{d \times n}} tr\left[\left(L^T S^{(w)} L\right)^{-1}\left(L^T S^{(b)} L\right)\right]$$

- In KNN, "K" represents the number of nearest neighbors that are considered when making a prediction for a new data point.
- LFDA does not have a direct "K" parameter like KNN does. Instead, LFDA is often used with a nearest neighbors graph, where the "K" parameter comes into play when defining the local neighborhood of each data point. The graph is used to capture the local relationships between data points.
- The "K" parameter in LMNN is related to the choice of the number of nearest neighbors considered during the optimization process.

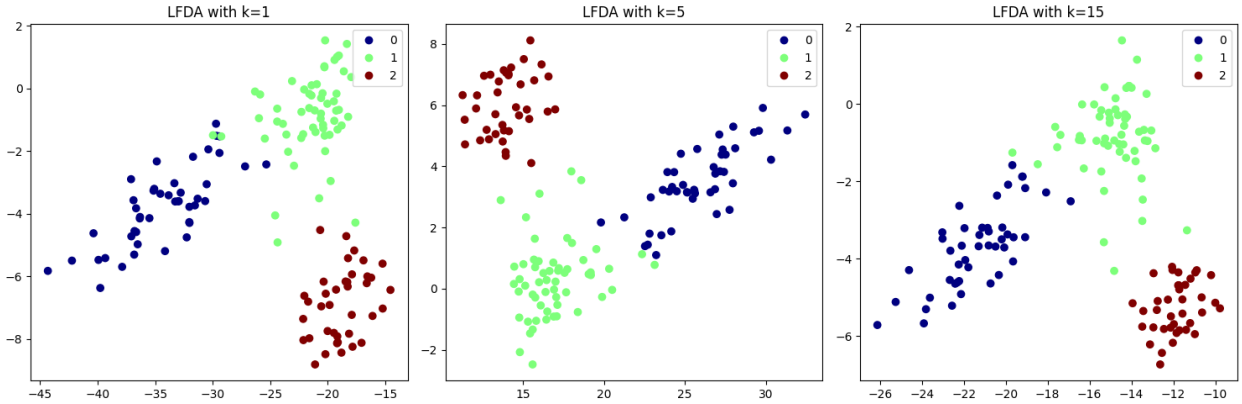
2. Dimension reduction

The dimension reduction of the **LMNN** method with *init='lda'* and different *n_neighbors* is depicted below.



K=15 is the best result since datapoints with the same class are closer, and datapoints with different classes are farther apart.

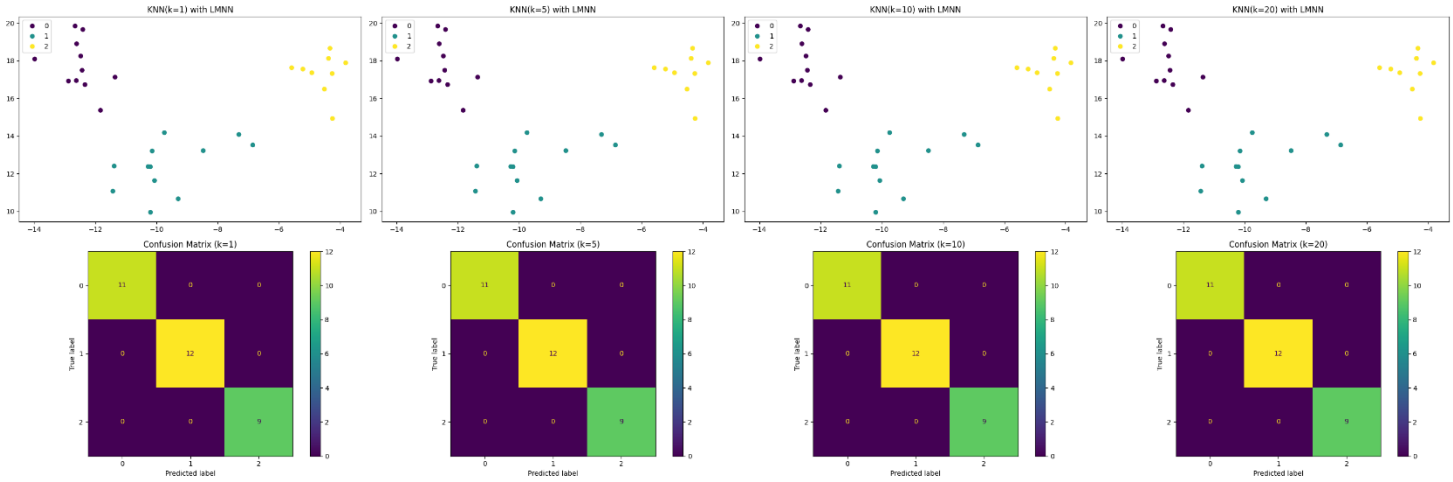
The dimension reduction of the **LFDA** method with different *k* is depicted below.



K=15 is the best result since datapoints with the same class are closer, and datapoints with different classes are farther apart.

3. Classification

KNN classification results of the LMNN method with different *k* are as follows:



k=1

	precision	recall	f1-score	support
0	1.00	1.00	1.00	11
1	1.00	1.00	1.00	12
2	1.00	1.00	1.00	9
accuracy			1.00	32
macro avg	1.00	1.00	1.00	32
weighted avg	1.00	1.00	1.00	32

k=5

	precision	recall	f1-score	support
0	1.00	1.00	1.00	11
1	1.00	1.00	1.00	12
2	1.00	1.00	1.00	9
accuracy			1.00	32
macro avg	1.00	1.00	1.00	32
weighted avg	1.00	1.00	1.00	32

k=10

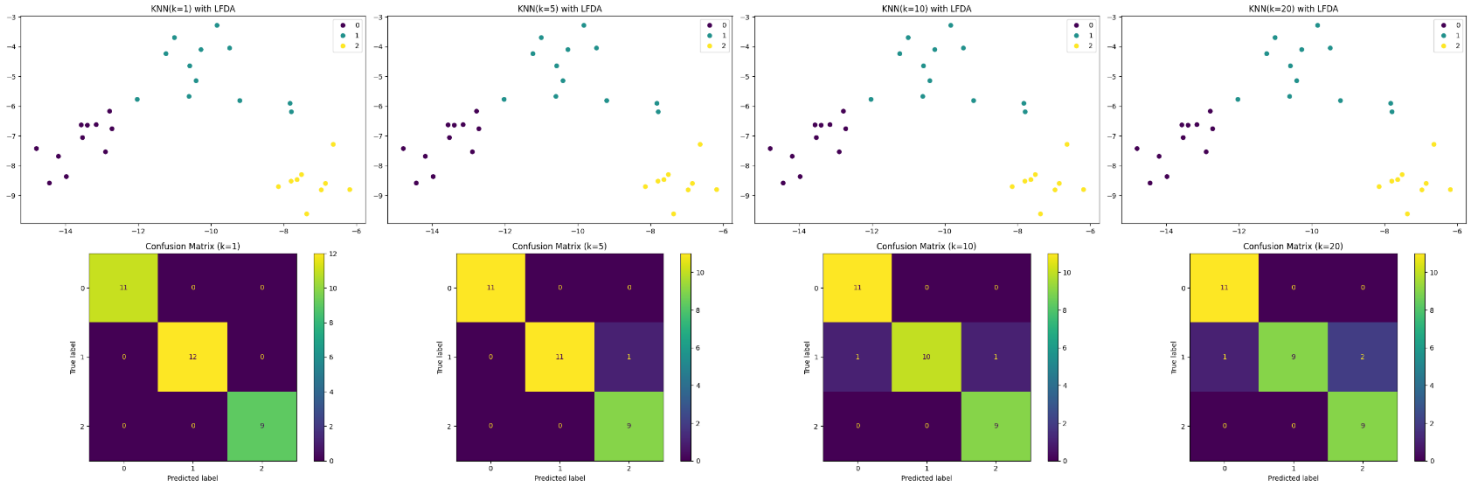
	precision	recall	f1-score	support
0	1.00	1.00	1.00	11
1	1.00	1.00	1.00	12
2	1.00	1.00	1.00	9
accuracy			1.00	32
macro avg	1.00	1.00	1.00	32
weighted avg	1.00	1.00	1.00	32

k=20

	precision	recall	f1-score	support
0	1.00	1.00	1.00	11
1	1.00	1.00	1.00	12
2	1.00	1.00	1.00	9
accuracy			1.00	32
macro avg	1.00	1.00	1.00	32
weighted avg	1.00	1.00	1.00	32

We can see after LMNN metric learning the results significantly improved.

KNN classification results of the LMNN method with different *k* are as follows:



k=1

	precision	recall	f1-score	support
0	1.00	1.00	1.00	11
1	1.00	1.00	1.00	12
2	1.00	1.00	1.00	9
accuracy			1.00	32
macro avg	1.00	1.00	1.00	32
weighted avg	1.00	1.00	1.00	32

k=5

	precision	recall	f1-score	support
0	1.00	1.00	1.00	11
1	1.00	0.92	0.96	12
2	0.90	1.00	0.95	9
accuracy			0.97	32
macro avg	0.97	0.97	0.97	32
weighted avg	0.97	0.97	0.97	32

k=10

	precision	recall	f1-score	support
0	0.92	1.00	0.96	11
1	1.00	0.83	0.91	12
2	0.90	1.00	0.95	9
accuracy			0.94	32
macro avg	0.94	0.94	0.94	32
weighted avg	0.94	0.94	0.94	32

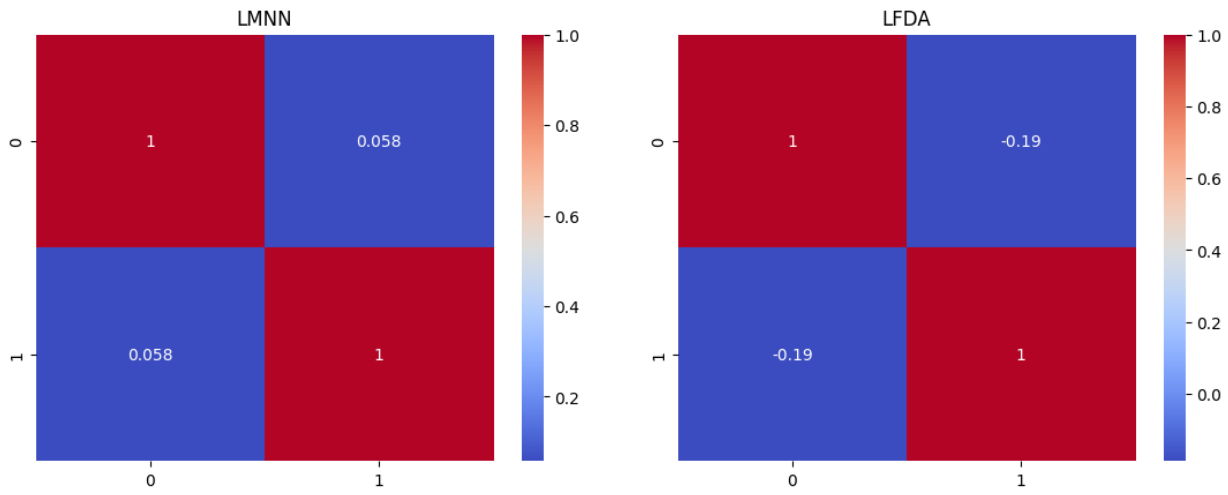
k=20

	precision	recall	f1-score	support
0	0.92	1.00	0.96	11
1	1.00	0.75	0.86	12
2	0.82	1.00	0.90	9
accuracy			0.91	32
macro avg	0.91	0.92	0.90	32
weighted avg	0.92	0.91	0.90	32

We can see after LFDA metric learning the results improved, however k=1 outperforms other values.

D: Correlation matrix

Correlation matrices of new feature spaces are depicted below:



From correlation matrix of LMNN method, can be inferred, the new features are highly uncorrelated which means new features are relatively better than ones from LFDA. This is the reason why LMNN had much better results during classification.