Bangladesh University of Business & Technology Dhaka,

Commerce College Road

Mirpur-2, Dhaka-1216

Course Code : CSE 476

Course Name : Data Mining Lab

Assignment 03

Submitted By	Submitted To
Sadeka Jafrin Id:18192103069 Intake:41 sec:03	Khan Md. Hasib Assistant Professor Dept of CSE, BUBT

CO3 What are the differences between the following data sets?
1- Contact Lens data: http://archive.ics.uci.edu/ml/datasets/Lenses
2- Iris data: http://archive.ics.uci.edu/ml/datasets/Iris

Ans: The main differences between the Lenses and Iris datasets are:

Purpose: The Lenses dataset is a diagnostic dataset used for classifying patients based on certain medical features, while the Iris dataset is a classification dataset used for identifying different species of iris flowers based on their petal and sepal dimensions.

Size and attributes: The Lenses dataset contains 24 instances and 5 attributes, while the Iris dataset contains 150 instances and 4 attributes.

Attribute types: The attributes in the Lenses dataset are a mixture of categorical and continuous variables, while the attributes in the Iris dataset are all continuous variables.

Class labels: The Lenses dataset has 3 class labels, while the Iris dataset has 3 class labels as well.

Complexity: The Iris dataset is a relatively simple dataset with well-separated classes, while the Lenses dataset is more complex and may require more sophisticated models to achieve good accuracy.

From the following algorithms which one is expected to perform best on
the Contact Lens data?
Ans: It is difficult to determine which model will perform the best without testing them on the dataset and evaluating their performance. Each of the algorithms you mentioned has its own strengths and weaknesses and is suitable for different types of problems.

For the Lenses dataset, which contains categorical features and a small number of instances, decision trees may perform well because they are good at handling categorical data and can produce interpretable models.

Neural networks can also be effective on this dataset as they can handle non-linear relationships between the input features and output labels. However, for small datasets like this, overfitting can be a problem if the model is too complex or the number of training examples is too small.

K-Nearest Neighbors (KNN) is also a simple algorithm that can perform well on this dataset. However, its performance may be impacted by the choice of k and the distance metric used.
In general, the best approach would be to try all three algorithms and evaluate their performance using appropriate metrics such as accuracy, precision, recall, and F1-score. Cross-validation can also be used to estimate the performance of each model and reduce the risk of overfitting.

Implement those on the Contact Lens data.
K-Nearest Neighbors
Decision Tree
Neural Networks

1. Import library

```
[3]  import pandas as pd
     import numpy as np
     from sklearn.model_selection import train_test_split
     from keras.models import Sequential
     from keras.layers import Dense, Conv2D, Flatten
     from tensorflow.keras import layers, models
     from sklearn.metrics import accuracy_score
```

2.Upload the dataset and assign column names and showing the top 5 values

```
[13] # load the dataset and assign column names
     data = pd.read_csv('/content/lenses.data',sep='\s+',header=None)
     data.rename(columns={0:'idd',1:'age',2:'perscription',3:'astigmatic', 4:'tears',5:'lenses'}, inplace=True)
     data.head(5)
```

	idd	age	perscription	astigmatic	tears	lenses
0	1	1	1	1	1	3
1	2	1	1	1	2	2
2	3	1	1	2	1	3
3	4	1	1	2	2	1
4	5	1	2	1	1	3

3. Showing the correlation of the dataset

```
[5]  data.corr()
```

	idd	age	perscription	astigmatic	tears	lenses
idd	1.000000	9.436285e-01	2.889260e-01	1.444630e-01	7.223151e-02	0.161804
age	0.943629	1.000000e+00	-7.931816e-17	-1.473052e-16	0.000000e+00	0.200745
perscription	0.288926	-7.931816e-17	1.000000e+00	-7.401487e-17	9.251859e-18	0.163908
astigmatic	0.144463	-1.473052e-16	-7.401487e-17	1.000000e+00	1.850372e-17	-0.163908
tears	0.072232	0.000000e+00	9.251859e-18	1.850372e-17	1.000000e+00	-0.710266
lenses	0.161804	2.007449e-01	1.639075e-01	-1.639075e-01	-7.102659e-01	1.000000

4.Spliting the data in train & test data

```
X = data.drop(["lenses","idd"], axis=1).values
Y = data["lenses"].values

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=1)
X_train.shape
```

```
(16, 4)
```

5. Applying Decision Tree

```
[7] from sklearn.tree import DecisionTreeClassifier

    tree = DecisionTreeClassifier(criterion="gini")
    tree.fit(X_train, Y_train)

    y_pred_train = tree.predict(X_train)
    y_pred = tree.predict(X_test)# this is the predictive model

    accuracy_train = accuracy_score(Y_train, y_pred_train)
    accuracy_test = accuracy_score(Y_test, y_pred)

    print("ACCURACY: TRAIN=%.4f TEST=%.4f" % (accuracy_train,accuracy_test))
```

```
ACCURACY: TRAIN=1.0000 TEST=0.8750
```

6. After the training , we got the training Accuracy = 100% &testing accuracy=87%

```
from sklearn.tree import DecisionTreeClassifier

tree = DecisionTreeClassifier(criterion="gini")
tree.fit(X_train, Y_train)

y_pred_train = tree.predict(X_train)
y_pred = tree.predict(X_test)# this is the predictive model

accuracy_train = accuracy_score(Y_train, y_pred_train)
accuracy_test = accuracy_score(Y_test, y_pred)

print("ACCURACY: TRAIN=%.4f TEST=%.4f" % (accuracy_train,accuracy_test))
```

```
ACCURACY: TRAIN=1.0000 TEST=0.8750
```

Applying K-Nearest Neighbors

1. Importing the KNeighborsClassifier library & also defining the k-neighbors model.

```
[8] from sklearn.neighbors import KNeighborsClassifier
    knn = KNeighborsClassifier(n_neighbors=3)
```

2. Fitting the dataset in the KNN Model.

```
[17] knn.fit(X_train, Y_train)
```

```
KNeighborsClassifier(n_neighbors=3)
```

3. Storing the predicted values

```
[21] prediction = []
     for i in range(8):
         p = knn.predict(X_test[i].reshape(1,-1))
         prediction.append(p[0])
```

```
(Y_test[:30] == prediction).sum()/len(prediction)
```

```
0.875
```