Bangladesh University of Business and Technology

Department of CSE

Name	:	Al Ahad Sufian
ID	:	18192103056
Intake	:	41
Section	:	03
Course Code	:	CSE-476
Course Title	:	Data Mining

What are the differences between the following data sets?

1- Contact Lens data: http://archive.ics.uci.edu/ml/datasets/Lenses
2- 2- Iris data: http://archive.ics.uci.edu/ml/datasets/Iris From the following algorithms which one is expected to perform best on the Contact Lens data? Implement those on the Contact Lens data→ K-Nearest Neighbors Decision Tree Neural Networks

Ans :

The Contact Lens data set is a collection of data related to contact lens usage compiled from a survey of 24 participants. The data set consists of four attributes: age, spectacle prescription, astigmatism, and tear production rate. The data is presented in a comma-separated values (CSV) file format, containing a total of 24 records with no missing values.

The Iris data set is a collection of data related to three species of irises compiled from measurements of 50 samples from each species. The data set consists of five attributes: sepal length and width, petal length and width, and species. The data is presented in a comma-separated values (CSV) file format, containing a total of 150 records with no missing values.

The primary difference between these two data sets is the type of data being recorded and the number of attributes and records in each data set. The Contact Lens data set contains four continuous numerical attributes and 24 records, while the Iris data set contains five continuous numerical attributes and 150 records.

From the algorithms mentioned, K-Nearest Neighbors is expected to perform best on the Contact Lens data. K-Nearest Neighbors is a supervised learning algorithm which searches for the most similar data points in a dataset and uses them to classify new data points. K-Nearest Neighbors is particularly suited for smaller datasets with a limited number of attributes, such as the Contact Lens data set, since it is relatively easy to search for similar data points in a smaller dataset.

Decision Tree is another supervised learning algorithm which uses a tree structure to classify data points. This algorithm is suitable for datasets with a limited number of attributes, such as the Contact Lens data set, since it can easily determine which attribute is most relevant for classification.

Neural Networks are a type of machine learning algorithm which use a network of artificial neurons to classify data points. Neural Networks are better suited for larger datasets with a large number of attributes, such as the Iris data set, since they can learn complex patterns in the data and can be used to classify data points with a high degree of accuracy.

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

```
data = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/lenses.data',sep='\s+',header=None)
data.rename(columns={0:'index',1:'age',2:'spectacle prescription',3:'astigmatic', 4:'tear production rate',5:'lenses'}, inplace=True)
data.head(5)
```

	index	age	spectacle prescription	astigmatic	tear production rate	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

1. Splitting the data into train & test data

```
X = data.drop(["lenses","index"], 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
X_train
```

```
array([[1, 1, 2, 1],
       [3, 1, 2, 2],
       [1, 2, 2, 1],
       [1, 2, 2, 2],
       [3, 2, 1, 2],
       [1, 1, 1, 2],
       [3, 1, 1, 1],
       [1, 1, 1, 1],
       [2, 2, 2, 2],
       [3, 2, 2, 2],
       [3, 2, 2, 1],
       [2, 1, 1, 2],
       [2, 1, 1, 1],
       [2, 2, 1, 1],
       [2, 1, 2, 2],
       [1, 2, 1, 2]])
```

2. Applying the Decision Tree to the lens dataset

```
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)

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

print("Training Accurecy is %.2f TEST=%.2f" % (accuracy_train*100,accuracy_test*100))
```

```
Training Accurecy is 100.00 TEST=87.50
```

3. 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)

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

print("Training Accurecy is %.2f TEST=%.2f" % (accuracy_train*100,accuracy_test*100))
```

```
Training Accurecy is 100.00 TEST=87.50
```

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 and here we got an accuracy of 87%

```
[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
```