Bangladesh University of Business & Technology (BUBT)

Assignment 01

”Data Mining Lab”
CSE-476

Submitted by

Habibullah
ID: 18192103080
Sec: 03
Intake: 41

Submitted to

Khan Md. Hasib

Assistant Professor

Department of Computer Science & Engineering

1. Apply data preprocessing steps (such as: Viewing your data, Handling duplicates, Column cleanup, DataFrame slicing, selecting, extracting) in the following dataset https://www.kaggle.com/datasets/selinraja/irish-data

1. Import Library

```
import pandas as pd
```

2. Upload the dataset & Viewing the data

```
[2] from google.colab import drive
    drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
[3] iris = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/Iris_Data.csv")
    iris
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

3. View the top 10 rows of the dataset.

```
iris.head(10)
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa

4. Showing the description of the whole dataset.

```
iris.describe()
```

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

5. Showing the info of the dataset.

```
iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

6. Dropping the duplicate data

```
display(iris.drop_duplicates())
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

7. Column cleanup

```
[25] for x in iris.index:
       if iris.loc[x, "sepal_length"] > 5:
         iris.loc[x, "sepal_length"] = 5
```

```
iris.head(10)
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.0	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.0	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa

8. Showing the unique data of a specific column.

```
print("Species")
print(iris['species'].unique())
```

```
Species
['Iris-setosa' 'Iris-versicolor' 'Iris-virginica']
```

9. Showing the data frame slicing.

```
iris1=iris.iloc[0:7]
iris1
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.0	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.0	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa

10. Showing the data frame slicing.

```
[14] iris2=iris.loc[:,'sepal_length':'petal_width']
     iris2
```

	sepal_length	sepal_width	petal_length	petal_width
0	5.0	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	5.0	3.0	5.2	2.3
146	5.0	2.5	5.0	1.9
147	5.0	3.0	5.2	2.0
148	5.0	3.4	5.4	2.3
149	5.0	3.0	5.1	1.8

```
copy=iris[['sepal_length','sepal_width','petal_length']]
copy
```

	sepal_length	sepal_width	petal_length
0	5.0	3.5	1.4
1	4.9	3.0	1.4
2	4.7	3.2	1.3
3	4.6	3.1	1.5
4	5.0	3.6	1.4
...	...	...	...
145	5.0	3.0	5.2
146	5.0	2.5	5.0
147	5.0	3.0	5.2
148	5.0	3.4	5.4
149	5.0	3.0	5.1

11. Showing the data frame extracting.

```
first = iris.iloc[3]
first
```

```
sepal_length            4.6
sepal_width             3.1
petal_length            1.5
petal_width             0.2
species         Iris-setosa
Name: 3, dtype: object
```

```
row2 = iris.iloc [[3, 5, 7]]
row2
```

	sepal_length	sepal_width	petal_length	petal_width	species
3	4.6	3.1	1.5	0.2	Iris-setosa
5	5.0	3.9	1.7	0.4	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa

Bangladesh University of Business & Technology (BUBT)

Assignment 02

"Data Mining Lab"
CSE-476

Submitted by

Habibullah
ID: 18192103080
Sec: 03
Intake: 41

Submitted to

Khan Md. Hasib

Assistant Professor

Department of Computer Science & Engineering

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
```

Bangladesh University of Business & Technology (BUBT)

Assignment 03

”Data Mining Lab”
CSE-476

Submitted by

Habibullah
ID: 18192103080
Sec: 03
Intake: 41

Submitted to

Khan Md. Hasib

Assistant Professor

Department of Computer Science & Engineering

01.Apply calculating mathematical statistics techniques (such as: mean -average value, median - middle value, median - middle value, median -middle value) in the following dataset https://www.kaggle.com/datasets/muthuj7/weather-dataset

1.Import library

```
import pandas as pd
import statistics
```

```
[ ] from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

2. Upload the dataset & Viewing the data

```
weather=pd.read_csv("/content/drive/MyDrive/Colab Notebooks/weatherHistory.csv")
weather
```

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
0	2006-04-01 00:00:00.000 +0200	Partly Cloudy	rain	9.472222	7.388889	0.89	14.1197	251.0	15.8263	0.0	1015.13	Partly cloudy throughout the day.
1	2006-04-01 01:00:00.000 +0200	Partly Cloudy	rain	9.355556	7.227778	0.86	14.2646	259.0	15.8263	0.0	1015.63	Partly cloudy throughout the day.
2	2006-04-01 02:00:00.000 +0200	Mostly Cloudy	rain	9.377778	9.377778	0.89	3.9284	204.0	14.9569	0.0	1015.94	Partly cloudy throughout the day.
3	2006-04-01 03:00:00.000 +0200	Partly Cloudy	rain	8.288889	5.944444	0.83	14.1036	269.0	15.8263	0.0	1016.41	Partly cloudy throughout the day.
4	2006-04-01 04:00:00.000 +0200	Mostly Cloudy	rain	8.755556	6.977778	0.83	11.0446	259.0	15.8263	0.0	1016.51	Partly cloudy throughout the day.
...	...	...	...	...	...	...	...	...	...	...	...	...
96448	2016-09-09 19:00:00.000 +0200	Partly Cloudy	rain	26.016667	26.016667	0.43	10.9963	31.0	16.1000	0.0	1014.36	Partly cloudy starting in the morning.
96449	2016-09-09 20:00:00.000 +0200	Partly Cloudy	rain	24.583333	24.583333	0.48	10.0947	20.0	15.5526	0.0	1015.16	Partly cloudy starting in the morning.
96450	2016-09-09 21:00:00.000 +0200	Partly Cloudy	rain	22.038889	22.038889	0.56	8.9838	30.0	16.1000	0.0	1015.66	Partly cloudy starting in the morning.
96451	2016-09-09 22:00:00.000 +0200	Partly Cloudy	rain	21.522222	21.522222	0.60	10.5294	20.0	16.1000	0.0	1015.95	Partly cloudy starting in the morning.
96452	2016-09-09 23:00:00.000 +0200	Partly Cloudy	rain	20.438889	20.438889	0.61	5.8765	39.0	15.5204	0.0	1016.16	Partly cloudy starting in the morning.

3. View the top 10 rows of the dataset.

```
weather.head(5)
```

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	Loud Cover	Pressure (millibars)	Daily Summary
0	2006-04-01 00:00:00.000 +0200	Partly Cloudy	rain	9.472222	7.388889	0.89	14.1197	251.0	15.8263	0.0	1015.13	Partly cloudy throughout the day.
1	2006-04-01 01:00:00.000 +0200	Partly Cloudy	rain	9.355556	7.227778	0.86	14.2646	259.0	15.8263	0.0	1015.63	Partly cloudy throughout the day.
2	2006-04-01 02:00:00.000 +0200	Mostly Cloudy	rain	9.377778	9.377778	0.89	3.9284	204.0	14.9569	0.0	1015.94	Partly cloudy throughout the day.
3	2006-04-01 03:00:00.000 +0200	Partly Cloudy	rain	8.288889	5.944444	0.83	14.1036	269.0	15.8263	0.0	1016.41	Partly cloudy throughout the day.
4	2006-04-01 04:00:00.000 +0200	Mostly Cloudy	rain	8.755556	6.977778	0.83	11.0446	259.0	15.8263	0.0	1016.51	Partly cloudy throughout the day.

4. Showing the mean value of the Humidity Column

```
[14] import statistics
```

```
mean=statistics.mean(weather["Humidity"])

print("Mean of Humidity is:", mean)
```

```
Mean of Humidity is: 0.7312608593566565
```

5. Showing the median value

```
median=statistics.median(weather["Humidity"])

print("Median of Humidity is:", median)
```

```
Median of Humidity is: 0.78
```

6. Showing the mode value

```
mode=statistics.mode(weather["Humidity"])

print("Mode of Humidity is:", mode)
```

```
Mode of Humidity is: 0.93
```

7. Showing the Standard deviation value

```
stdev=statistics.stdev(weather["Humidity"])

print("Standard deviation of Humidity is:", stdev)
```

```
Standard deviation of Humidity is: 0.19565322439944888
```