Al-Ahliyya Amman University
Faculty of information technology
Subject: Machine Learning and Deep Learning

Name of the Dataset:	Drug Classification
Link to the Dataset:	https://www.kaggle.com/datasets/prathamtripathi/drug-classification?resource=download
Number of Attributes:	6
Number of Samples:	200
List of Attributes and Types:	Age int64 Sex object BP object Cholesterol object Na_to_K float64 Drug object

Student name: Osama Mohammed Abuzraiq

Student ID: 202010646

➢ Importing Libraries and reading the file:

In [1]:
```
#Import python libraries:
import numpy as np
import pandas as pd
import sklearn as sk
import matplotlib.pyplot as plt
import sklearn.model_selection as skmodel
import sklearn.neighbors as skng
import sklearn.tree as skdt
import sklearn.metrics as skmet
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn import svm

from sklearn.linear_model import LinearRegression
```

In [2]:
```
#Read the data:
df = pd.read_csv(r"C:\Users\osama\Desktop\Third year - First semester\Machine Learning and Deep Learning\Project\drug200.csv")
df.head(10)
```

Out[2]:

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	F	HIGH	HIGH	25.355	DrugY
1	47	M	LOW	HIGH	13.093	drugC
2	47	M	LOW	HIGH	10.114	drugC
3	28	F	NORMAL	HIGH	7.798	drugX
4	61	F	LOW	HIGH	18.043	DrugY
5	22	F	NORMAL	HIGH	8.607	drugX
6	49	F	NORMAL	HIGH	16.275	DrugY
7	41	M	LOW	HIGH	11.037	drugC
8	60	M	NORMAL	HIGH	15.171	DrugY
9	43	M	LOW	NORMAL	19.368	DrugY

➢ Tuning data and separating features to training and testing

In [4]:
```
#Categorical to Numeric data conversion:
df['Sex']=pd.factorize(df['Sex'])[0]
df['BP']=pd.factorize(df['BP'])[0]
df['Cholesterol']=pd.factorize(df['Cholesterol'])[0]
df['Drug']=pd.factorize(df['Drug'])[0]
df.head(10)
```

Out[4]:

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	0	0	0	25.355	0
1	47	1	1	0	13.093	1
2	47	1	1	0	10.114	1
3	28	0	2	0	7.798	2
4	61	0	1	0	18.043	0
5	22	0	2	0	8.607	2
6	49	0	2	0	16.275	0
7	41	1	1	0	11.037	1
8	60	1	2	0	15.171	0
9	43	1	1	1	19.368	0

In [5]:
```
#Separate features from class-labels:
X= df.iloc[:, 0:5] #features from 0-4 (Age, Sex, BP, Cholesterol, NA_to_K)
Y= df.iloc[:, 5] #last feature (Drug)
```

In [6]:
```
#Separate Training from Testing
X_train, X_test, Y_train, Y_test = skmodel.train_test_split(X, Y,
test_size=0.25)
```

➢ Nearest Neighborhood Classification (KNN =4)

In [7]:
```
#Create model for KNN classifier with K=4:
KNN = skng.KNeighborsClassifier(n_neighbors=4)
```

In [8]:
```
#Fit data into the KNN model:
KNN.fit(X_train,Y_train)
```

Out[8]:
```
KNeighborsClassifier(n_neighbors=4)
```

In [9]:
```
#Predict Y data with KNN classifier:
Y_predict = KNN.predict(X_test)
```

- **Results and confusion matrix:**

In [10]:
```
#Print results of KNN:
report = skmet.classification_report(Y_test, Y_predict)
print(report)
```

```
              precision    recall  f1-score   support

           0       0.96      0.88      0.92        26
           1       0.00      0.00      0.00         6
           2       0.44      0.73      0.55        11
           3       0.50      0.75      0.60         4
           4       1.00      0.33      0.50         3

    accuracy                           0.70        50
   macro avg       0.58      0.54      0.51        50
weighted avg       0.70      0.70      0.68        50
```

In [11]:
```
#Print confusion matrix of KNN:
matrix = skmet.confusion_matrix(Y_test, Y_predict)
print(matrix)
```

```
[[23  0  2  1  0]
 [ 0  0  5  1  0]
 [ 1  1  8  1  0]
 [ 0  0  1  3  0]
 [ 0  0  2  0  1]]
```

➢ SVM Classification:

In [23]:
```
#Create model for SVM classifier:
clf = svm.SVC(kernel='linear') # Linear Kernel
```

In [24]:
```
#Fit data into the SVM model:
clf.fit(X_train, Y_train)
```

Out[24]:
```
SVC(kernel='linear')
```

In [25]:
```
Y_predict = clf.predict(X_test)
```

- **Results and confusion matrix:**

In [26]:
```
#Print results of SVM:
report = skmet.classification_report(Y_test, Y_predict)
print(report)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        26
           1       1.00      1.00      1.00         6
           2       1.00      1.00      1.00        11
           3       1.00      1.00      1.00         4
           4       1.00      1.00      1.00         3

    accuracy                           1.00        50
   macro avg       1.00      1.00      1.00        50
weighted avg       1.00      1.00      1.00        50
```

In [27]:
```
#Print confusion matrix of SVM:
matrix = skmet.confusion_matrix(Y_test, Y_predict)
print(matrix)
```

```
[[26  0  0  0  0]
 [ 0  6  0  0  0]
 [ 0  0 11  0  0]
 [ 0  0  0  4  0]
 [ 0  0  0  0  3]]
```

➢ Random Forest Classification:

- Hyper Parameter Tuning for random forest

In [12]:
```
#Hyperparameter tuning using grid search for Random forest:
RFC = RandomForestClassifier(random_state=42, n_jobs =-1, oob_score=True)
```

In [13]:
```
parameters = {
    'max_depth': [2,3,5,10,20],
    'min_samples_leaf': [5,10,20,50,100,200],
    'n_estimators': [10,25,30,50,100,200]
}
```

In [14]:
```
#Make Grid Search model:
grid = GridSearchCV(estimator=RFC,
                    param_grid=parameters,
                    cv = 4,
                    n_jobs=-1,verbose=1, scoring ="accuracy")
```

In [15]:
```
#Fit data into the model:
grid.fit(X_train,Y_train)
```

```
Fitting 4 folds for each of 180 candidates, totalling 720 fits
```

Out[15]:
```
GridSearchCV(cv=4,
             estimator=RandomForestClassifier(n_jobs=-1, oob_score=True,
                                              random_state=42),
             n_jobs=-1,
             param_grid={'max_depth': [2, 3, 5, 10, 20],
                         'min_samples_leaf': [5, 10, 20, 50, 100, 200],
                         'n_estimators': [10, 25, 30, 50, 100, 200]},
             scoring='accuracy', verbose=1)
```

In [16]:
```
#Check the Grid best score:
print(grid.best_score_)
```

```
0.9270981507823612
```

- Using best hyperparameters for the model

In [17]:
```
#output the best estimated hyperparameters for the random forest model:
rf_best = grid.best_estimator_
print(rf_best)
```

```
RandomForestClassifier(max_depth=5, min_samples_leaf=5, n_estimators=200,
                       n_jobs=-1, oob_score=True, random_state=42)
```

In [18]:
```
#Use hyperparameters from grid search in random forest model:
RFC2 = RandomForestClassifier(max_depth=5, min_samples_leaf=5, n_estimators=10,
                       n_jobs=-1, oob_score=True, random_state=42)
```

In [19]:
```
#Fit data into the Random Forest model:
RFC2.fit(X_train, Y_train)
```

Out[19]:
```
RandomForestClassifier(max_depth=5, min_samples_leaf=5, n_estimators=10,
                       n_jobs=-1, oob_score=True, random_state=42)
```

In [20]:
```
#Predict Y data with Random Forest classifier:
Y_predict = RFC2.predict(X_test)
```

- Results and confusion matrix

In [21]:
```
#Print results of Random Forest:
report = skmet.classification_report(Y_test, Y_predict)
print(report)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        26
           1       1.00      0.33      0.50         6
           2       0.73      1.00      0.85        11
           3       1.00      1.00      1.00         4
           4       1.00      1.00      1.00         3

    accuracy                           0.92        50
   macro avg       0.95      0.87      0.87        50
weighted avg       0.94      0.92      0.91        50
```

In [22]:
```
#Print confusion matrix of Random Forest:
matrix = skmet.confusion_matrix(Y_test, Y_predict)
print(matrix)
```

```
[[26  0  0  0  0]
 [ 0  2  4  0  0]
 [ 0  0 11  0  0]
 [ 0  0  0  4  0]
 [ 0  0  0  0  3]]
```

- Conclusion:

SVM Classification had the best results out of the three classifications used based on the results.