This notebook contains codes executed for Assignment-4 (**Predictive Data Mining**) for the course **CSIT558_01SP25** by:

- Chaudhari, Amishaben Natavarbhai (CWID:50129394)
- Javed, Syed Mehrose (CWID: 50128848)
- Subhani, Muhammad Kamal (CWID: 50136417)
- Thangavel Sathiyamoorthy, Punithan (CWID: 50135877)
- Venkata Appala Manoj Muvvala (CWID: 50122981)


```
# Importing the required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score
import joblib

import warnings
warnings.filterwarnings("ignore")
```

DATA LOADING AND EDA

```
# Loading the data
data = pd.read_csv("drug1000.csv")
data
```


```
{"summary":"{\n  \"name\": \"data\",\n  \"rows\": 1000,\n  \"fields\":
[\n    {\n      \"column\": \"Age\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 16,\n        \"min\": 15,\n
\"max\": 79,\n        \"num_unique_values\": 65,\n        \"samples\":
[\n          62,\n          73,\n          29\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Sex\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 2,\n
\"samples\": [\n          \"M\",\n          \"F\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"BP\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 3,\n
\"samples\": [\n          \"LOW\",\n          \"NORMAL\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Cholesterol\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 2,\n        \"samples\": [\n
\"NORMAL\",\n          \"HIGH\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Na_to_K\",\n      \"properties\":
```

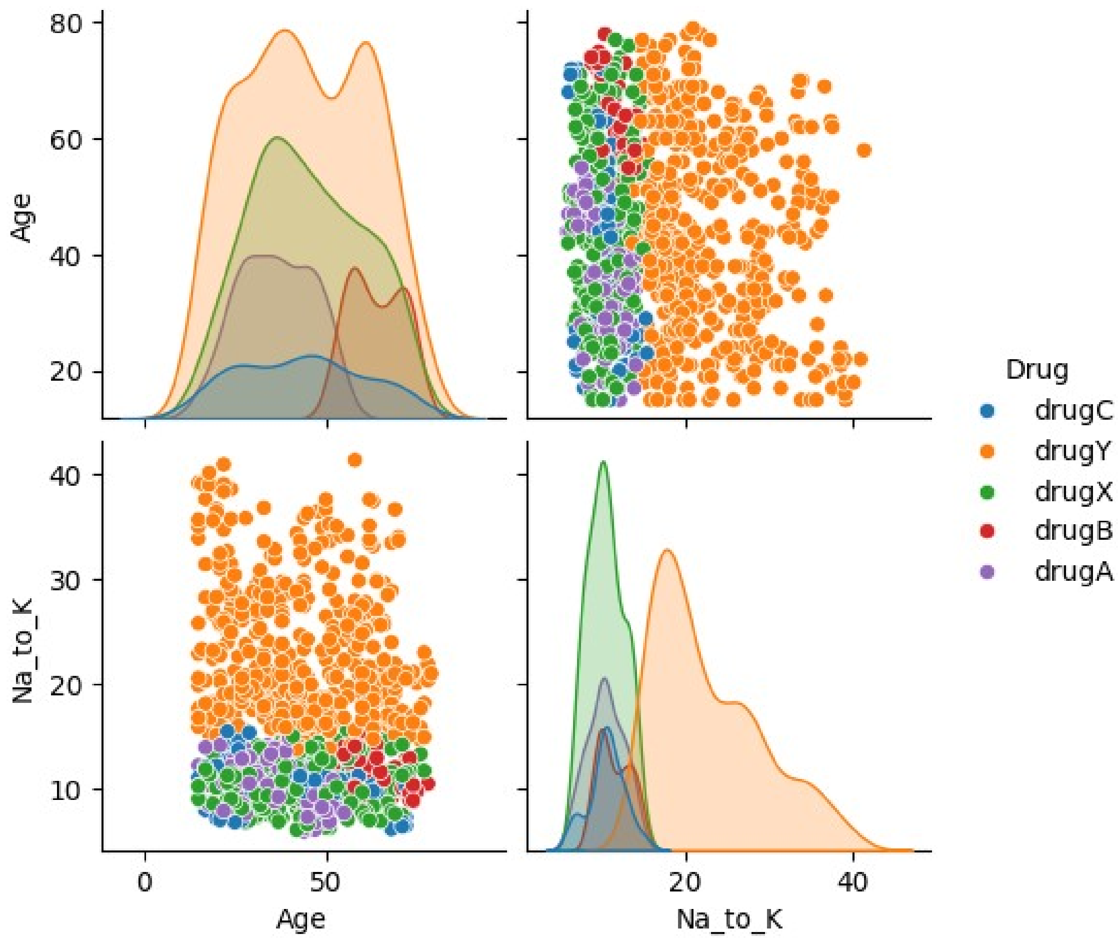
```
{\n        \"dtype\": \"number\",\n        \"std\":
7.652179968263982,\n        \"min\": 5.921618772385234,\n
\"max\": 41.33547607149792,\n        \"num_unique_values\": 1000,\n
\"samples\": [\n          10.38522916646769,\n
29.611525616870747\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Drug\",\n      \"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 5,\n        \"samples\": [\n
\"drugY\",\n          \"drugA\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"data"}
```


```
# Exploring the data
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Age          1000 non-null   int64
 1   Sex          1000 non-null   object
 2   BP           1000 non-null   object
 3   Cholesterol  1000 non-null   object
 4   Na_to_K      1000 non-null   float64
 5   Drug         1000 non-null   object
dtypes: float64(1), int64(1), object(4)
memory usage: 47.0+ KB
```

```
# Plotting a pair-plot of the attributes in the dataset
sns.pairplot(data=data, hue='Drug')
```

```
<seaborn.axisgrid.PairGrid at 0x79b6221add90>
```



DATA CLEANING

```
# Checking the data for null values
data.isnull().sum()

Age            0
Sex            0
BP             0
Cholesterol    0
Na_to_K        0
Drug           0
dtype: int64

# Checking for duplicates
duplicates = data.duplicated().sum()
print(f"Number of duplicate rows: {duplicates}")

Number of duplicate rows: 0

# checking the data by printing the initial 5 rows
data.head()
```

```
{"summary":"{\n  \"name\": \"data\",\n  \"rows\": 1000,\n  \"fields\":
[\n    {\n      \"column\": \"Age\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 16,\n        \"min\": 15,\n
\"max\": 79,\n        \"num_unique_values\": 65,\n        \"samples\":
[\n          62,\n          73,\n          29\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Sex\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 2,\n
\"samples\": [\n          \"M\",\n          \"F\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"BP\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 3,\n
\"samples\": [\n          \"LOW\",\n          \"NORMAL\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Cholesterol\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 2,\n        \"samples\": [\n
\"NORMAL\",\n          \"HIGH\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Na_to_K\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\":
7.652179968263982,\n        \"min\": 5.921618772385234,\n
\"max\": 41.33547607149792,\n        \"num_unique_values\": 1000,\n
\"samples\": [\n          10.38522916646769,\n
29.611525616870747\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Drug\",\n      \"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 5,\n        \"samples\": [\n
\"drugY\",\n          \"drugA\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"data"}
```


MODELING PREPARATION

```
#Printing the unique values of the categrical variables
print(data['Sex'].unique())
print(data['BP'].unique())
print(data['Cholesterol'].unique())
print(data['Drug'].unique())

['F' 'M']
['LOW' 'NORMAL' 'HIGH']
['HIGH' 'NORMAL']
['drugC' 'drugY' 'drugX' 'drugB' 'drugA']

# Mapping the categorical variables to numerical values
data['Sex'] = data['Sex'].replace({'F':1, 'M':2})
data['BP'] = data['BP'].replace({'LOW':1, 'NORMAL':2, 'HIGH':3})
data['Cholesterol'] = data['Cholesterol'].replace({'LOW':1,
'NORMAL':2, 'HIGH':3})
```

```
data['Drug'] = data['Drug'].replace({'drugA':1, 'drugB':2, 'drugC':3,
'drugX':4, 'drugY':5})
data.head()
```


{"summary":"{\n \"name\": \"data\",\n \"rows\": 1000,\n \"fields\": [\n   {\n   \"column\": \"Age\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 16,\n   \"min\": 15,\n   \"max\": 79,\n   \"num_unique_values\": 65,\n   \"samples\": [\n   62,\n   73,\n   29\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Sex\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 1,\n   \"max\": 2,\n   \"num_unique_values\": 2,\n   \"samples\": [\n   2,\n   1\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"BP\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 1,\n   \"max\": 3,\n   \"num_unique_values\": 3,\n   \"samples\": [\n   1,\n   2\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Cholesterol\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 2,\n   \"max\": 3,\n   \"num_unique_values\": 2,\n   \"samples\": [\n   2,\n   3\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Na_to_K\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 7.652179968263982,\n   \"min\": 5.921618772385234,\n   \"max\": 41.33547607149792,\n   \"num_unique_values\": 1000,\n   \"samples\": [\n   10.38522916646769,\n   29.611525616870747\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Drug\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 1,\n   \"min\": 1,\n   \"max\": 5,\n   \"num_unique_values\": 5,\n   \"samples\": [\n   5,\n   1\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"data"}


```
# Separating the target variable from the rest of the variables
features = ["Age", "Sex", "BP", "Cholesterol", "Na_to_K"]

x = data[features]
y = data.Drug

# Checking the variables other than the target variable
x.head()
```


{"summary":"{\n \"name\": \"x\",\n \"rows\": 1000,\n \"fields\": [\n   {\n   \"column\": \"Age\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 16,\n   \"min\": 15,\n

```
\"max\": 79,\n        \"num_unique_values\": 65,\n        \"samples\":
[\n          62,\n          73,\n          29\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Sex\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 1,\n
\"max\": 2,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          2,\n          1\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"BP\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 1,\n
\"max\": 3,\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          1,\n          2\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Cholesterol\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0,\n        \"min\": 2,\n
\"max\": 3,\n        \"num_unique_values\": 2,\n        \"samples\":
[\n          2,\n          3\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Na_to_K\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 7.652179968263982,\n        \"min\":
5.921618772385234,\n        \"max\": 41.33547607149792,\n
\"num_unique_values\": 1000,\n        \"samples\": [\n
10.38522916646769,\n          29.611525616870747\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe","variable_name":"x"}
```


```python
# Checking the traget variable
y.head()
```

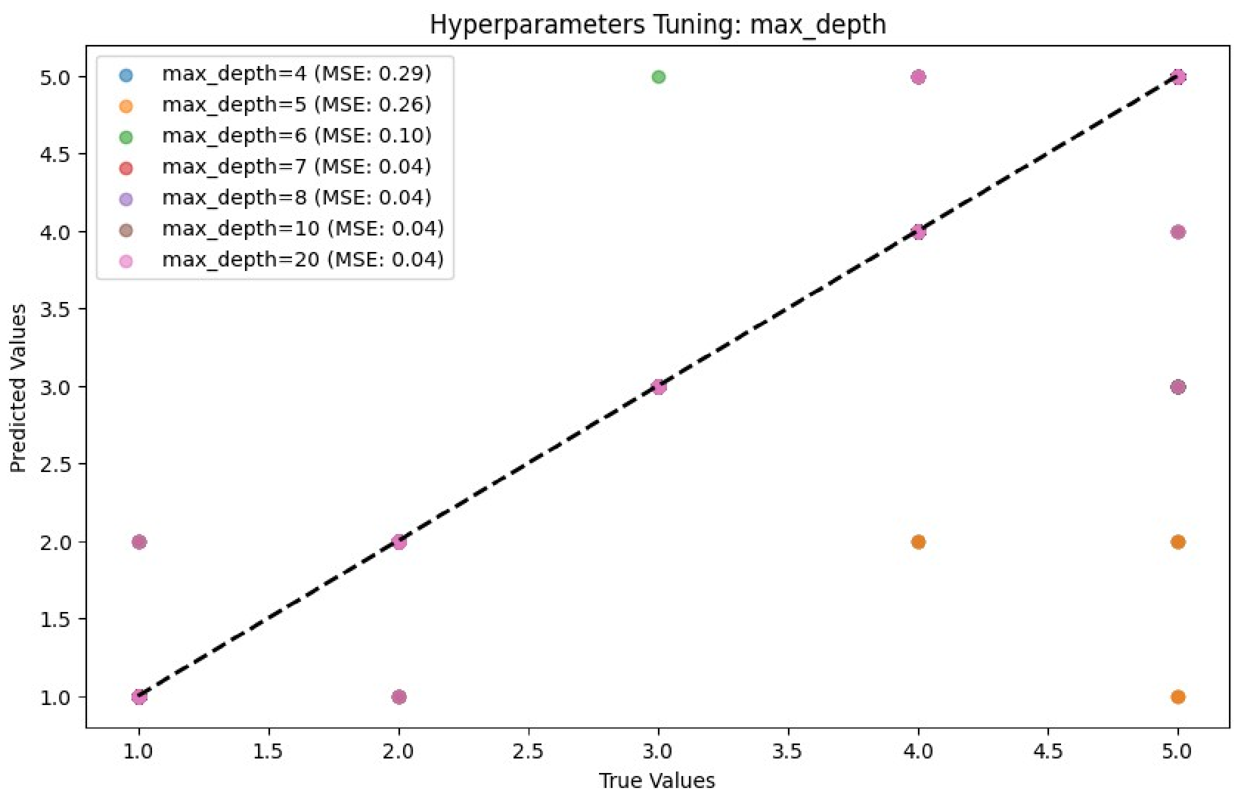
```
0    3
1    5
2    5
3    4
4    4
Name: Drug, dtype: int64
```

```python
# Splitting the data into Train and Test Sets
train_x, test_x, train_y, test_y = train_test_split(x, y, random_state
= 20)
```

TUNING THE HYPERPARAMETERS

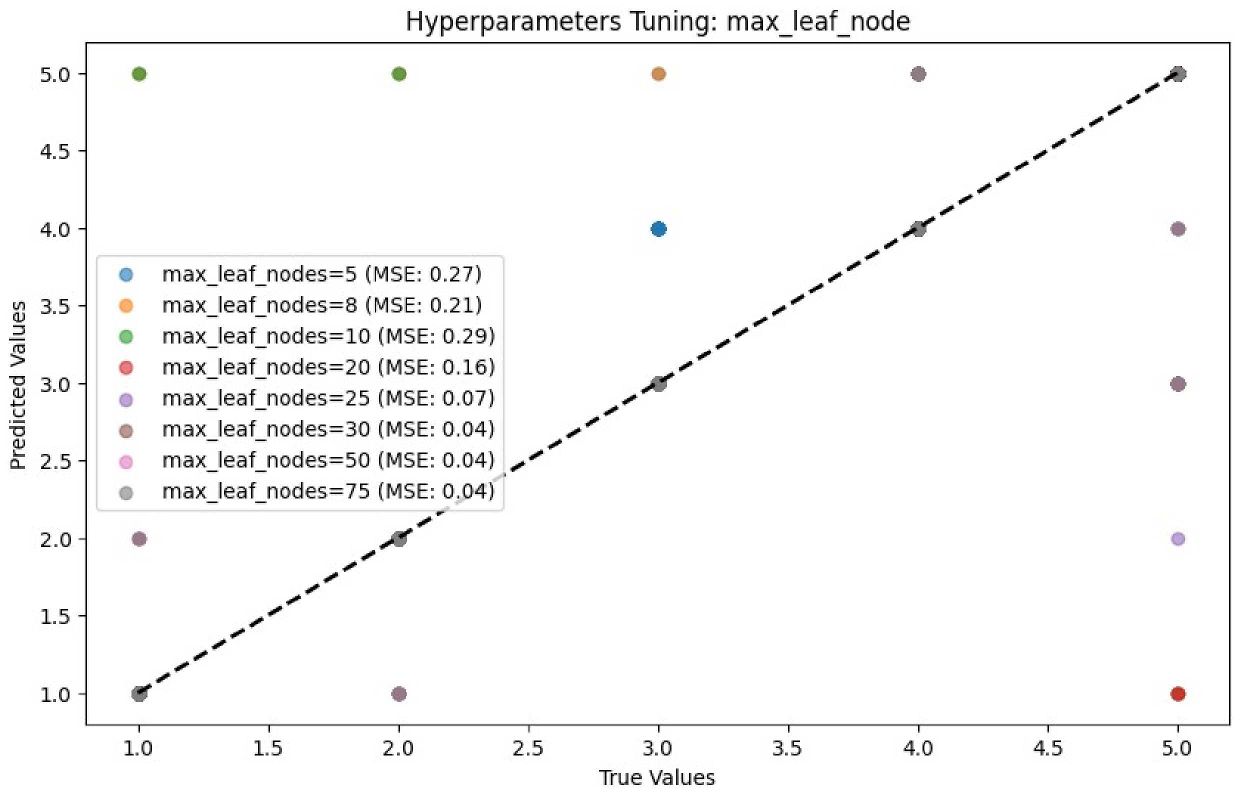
```python
# Tuning Hyperparameters: max_depth

# Training models with different max_depth values
depths = [4, 5, 6, 7, 8 , 10, 20]
models = {depth: DecisionTreeClassifier(max_depth=depth,
random_state=20).fit(train_x, train_y) for depth in depths}

# Making predictions
```

```python
predictions = {depth: models[depth].predict(test_x) for depth in
depths}

# Evaluating performance using mean_squared_error
mse_scores = {depth: mean_squared_error(test_y, predictions[depth])
for depth in depths}

# Visualizing True vs Predicted values
plt.figure(figsize=(10, 6))

for depth in depths:
    # Plotting the MSE's for different max_depth values
    plt.scatter(test_y, predictions[depth], alpha=0.6,
label=f"max_depth={depth} (MSE: {mse_scores[depth]:.2f})")

plt.plot([min(test_y), max(test_y)], [min(test_y), max(test_y)],
'k--', lw=2)  # Perfect prediction line
plt.xlabel("True Values")
plt.ylabel("Predicted Values")
plt.title("Hyperparameters Tuning: max_depth")
plt.legend()
plt.show()
```



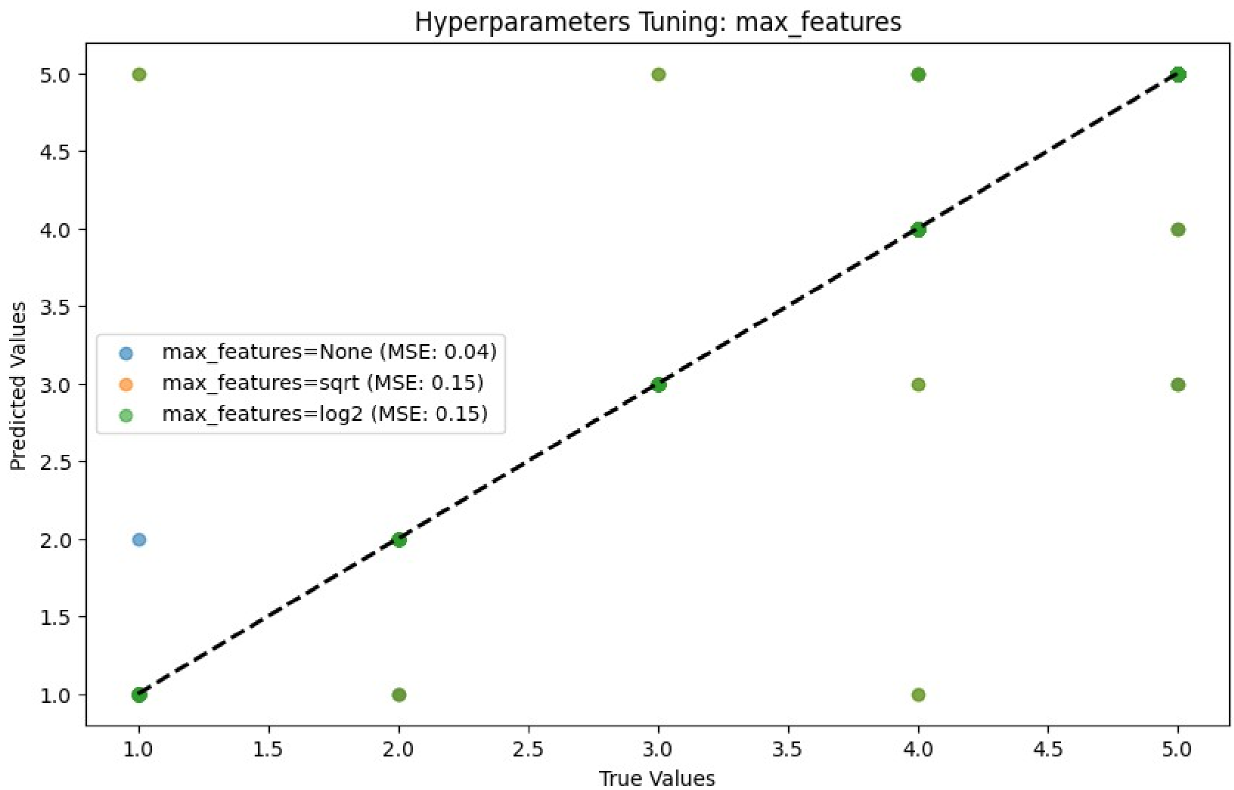

```python
# Tuning Hyperparameters: max_leaf_nodes
```

```
# Training models with different max_leaf_nodes values
leafnodes = [5, 8, 10, 20, 25, 30, 50, 75]
models = {leafnode: DecisionTreeClassifier(max_leaf_nodes=leafnode,
random_state=20).fit(train_x, train_y) for leafnode in leafnodes}

# Making predictions
predictions = {leafnode: models[leafnode].predict(test_x) for leafnode
in leafnodes}

# Evaluating performance using mean_squared_error
mse_scores = {leafnode: mean_squared_error(test_y,
predictions[leafnode]) for leafnode in leafnodes}

# Visualizing True vs Predicted values
plt.figure(figsize=(10, 6))

for leafnode in leafnodes:
    # Plotting the MSE's for different leaf nodes
    plt.scatter(test_y, predictions[leafnode], alpha=0.6,
label=f"max_leaf_nodes={leafnode} (MSE: {mse_scores[leafnode]:.2f})")

plt.plot([min(test_y), max(test_y)], [min(test_y), max(test_y)],
'k--', lw=2)  # Perfect prediction line
plt.xlabel("True Values")
plt.ylabel("Predicted Values")
plt.title("Hyperparameters Tuning: max_leaf_node")
plt.legend()
plt.show()
```




```
# Tuning Hyperparameters : max_features

# Training models with different max_leaf_nodes values
feature_settings = [None, 'sqrt', 'log2']
models = {feature: DecisionTreeClassifier(max_features=feature,
random_state=20).fit(train_x, train_y) for feature in
feature_settings}

# Making predictions
predictions = {feature: models[feature].predict(test_x) for feature in
feature_settings}

# Evaluating performance using mean_squared_error
mse_scores = {feature: mean_squared_error(test_y,
predictions[feature]) for feature in feature_settings}

# Visualizing True vs Predicted values
plt.figure(figsize=(10, 6))

for feature in feature_settings:
    plt.scatter(test_y, predictions[feature], alpha=0.6,
label=f"max_features={feature} (MSE: {mse_scores[feature]:.2f})") #
Plotting the MSE's for different leaf nodes

plt.plot([min(test_y), max(test_y)], [min(test_y), max(test_y)],
'k--', lw=2)  # Perfect prediction line
```

```python
plt.xlabel("True Values")
plt.ylabel("Predicted Values")
plt.title("Hyperparameters Tuning: max_features")
plt.legend()
plt.show()
```



CREATING THE FINAL CLASSIFIER MODEL BASED ON THE MOST OPTIMAL VALUES OF THE HYPERPARAMETERS

```python
# Traiing and fitting the Decision Tree Classifier model based on the
tuned hyperparameters
model = DecisionTreeClassifier(max_depth=7,  max_leaf_nodes = 30,
max_features=None, random_state = 20)
model.fit(train_x, train_y)
```

```
DecisionTreeClassifier(max_depth=7, max_leaf_nodes=30,
random_state=20)
```

```python
# Getting model parameters
model.get_params()
```

```
{'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': 7,
 'max_features': None,
```

```
 'max_leaf_nodes': 30,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'monotonic_cst': None,
 'random_state': 20,
 'splitter': 'best'}
```

```
# Verifying the model by checking the error
y_pred = model.predict(test_x)

mae = mean_absolute_error(test_y, y_pred)
mse = mean_squared_error(test_y, y_pred)
r2 = r2_score(test_y, y_pred)

print(f'MAE: {mae: .2f}')
print(f'MSE: {mse: .2f}')
print(f'R2: {r2: .2f}')
```

```
MAE:  0.03
MSE:  0.04
R2:  0.98
```

EXPERIMENTING WITH OTHER ACCEPTABLE VALUES OF THE HYPERPARAMETERS

WHEN max_depth=7, max_leaf_nodes = 25, max_features='log2'

```
# Traiing and fitting the model when
# max_depth=7,  max_leaf_nodes = 25, max_features='log2'
model = DecisionTreeClassifier(max_depth=7,
                               max_leaf_nodes = 25,
                               max_features='log2', random_state = 20)
model.fit(train_x, train_y)

# Verifying the model by checking the error
y_pred = model.predict(test_x)

mae = mean_absolute_error(test_y, y_pred)
mse = mean_squared_error(test_y, y_pred)
r2 = r2_score(test_y, y_pred)

print(f'MAE: {mae: .2f}')
print(f'MSE: {mse: .2f}')
print(f'R2: {r2: .2f}')
```

```
MAE:  0.12
MSE:  0.25
R2:  0.87
```

max_depth=10, max_leaf_nodes = 50, max_features='sqrt'

```
# Traiing and fitting the model when
# max_depth=10,  max_leaf_nodes = 50, max_features='sqrt'
model = DecisionTreeClassifier(max_depth=10,
                               max_leaf_nodes = 50,
                               max_features='sqrt', random_state = 20)
model.fit(train_x, train_y)

# Verifying the model by checking the error
y_pred = model.predict(test_x)

mae = mean_absolute_error(test_y, y_pred)
mse = mean_squared_error(test_y, y_pred)
r2 = r2_score(test_y, y_pred)

print(f'MAE: {mae: .2f}')
print(f'MSE: {mse: .2f}')
print(f'R2: {r2: .2f}')
```

```
MAE:  0.11
MSE:  0.28
R2:  0.86
```

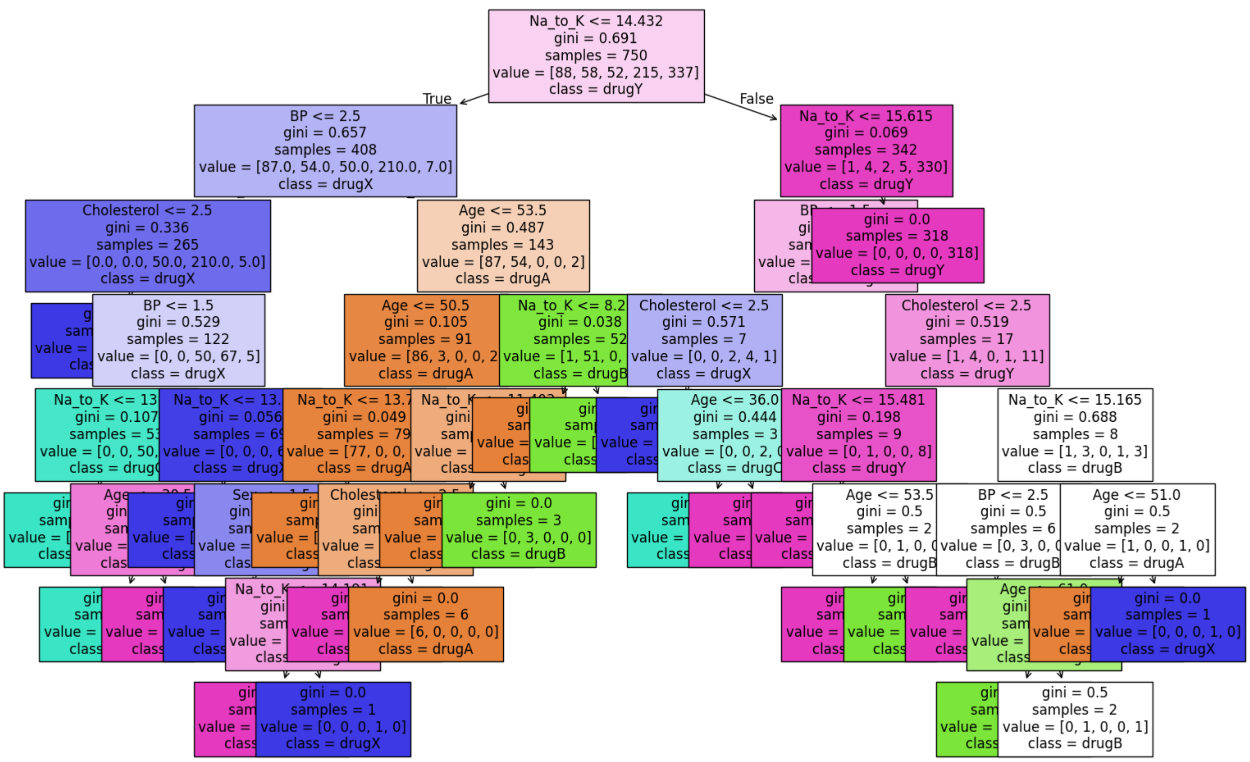
WHEN max_depth=8, max_leaf_nodes = 20, max_features='log2'

```
# Traiing and fitting the model when
# max_depth=8,  max_leaf_nodes = 20, max_features='log2'
model = DecisionTreeClassifier(max_depth=8,
                               max_leaf_nodes = 20,
                               max_features='log2', random_state = 20)
model.fit(train_x, train_y)

# Verifying the model by checking the error
y_pred = model.predict(test_x)

mae = mean_absolute_error(test_y, y_pred)
mse = mean_squared_error(test_y, y_pred)
r2 = r2_score(test_y, y_pred)

print(f'MAE: {mae: .2f}')
print(f'MSE: {mse: .2f}')
print(f'R2: {r2: .2f}')
```

```
MAE:  0.21
MSE:  0.47
R2:  0.76
```

PLOTTING THE DECISION TREE

```python
# Plotting the tree in diagramtic representation
from sklearn import tree

fig = plt.figure(figsize=(18, 12))
_ = tree.plot_tree(model,
                   feature_names=x.columns.tolist(),
                   class_names=['drugA', 'drugB', 'drugC', 'drugX',
'drugY'],
                   filled=True,
                   fontsize=12
                  )
```



CREATING THE CLASSIFIER TOOL

```python
# Storing the trained model as pickle file
joblib.dump(model, 'drug_classifier.pkl')
```

```
['drug_classifier.pkl']
```

```python
# Loading the trained model
model = joblib.load('drug_classifier.pkl')

# Defining the mappings for categorical variables
sex_map = {'M': 1, 'F': 2}
```

```python
bp_map = {'LOW': 1, 'NORMAL': 2, 'HIGH': 3}
cholesterol_map = {'LOW': 1, 'NORMAL': 2, 'HIGH': 3}

# Defining the function to predict the drug
def predict_drug(age, sex, bp, cholesterol, na_to_k):
    # Converting categorical inputs to numerical values
    sex = sex_map.get(sex.upper(), 0)
    bp = bp_map.get(bp.upper(), 0)
    cholesterol = cholesterol_map.get(cholesterol.upper(), 0)

    # Preparing input array
    features = np.array([[age, sex, bp, cholesterol, na_to_k]])

    # Making prediction
    prediction = model.predict(features)[0]

    # Mapping the Predicted numerical value to its corresponding Drug
Name
    drug_map = {1: 'drugA', 2: 'drugB', 3: 'drugC', 4: 'drugX', 5:
'drugY'}
    prediction = drug_map.get(prediction, 'Unknown')

    return prediction
```

Testing the Prediction with different sets of inputs

```python
# Input: 56 F LOW HIGH 11.567 drugC
age = 56
sex = "F"
bp = "LOW"
cholesterol = "HIGH"
na_to_k = 11.567

# Getting the Prediction
predicted_drug = predict_drug(age, sex, bp, cholesterol, na_to_k)

# Printing the Result
print(f"Predicted Drug: {predicted_drug}")
```

```
Predicted Drug: drugC
```

```python
# Input: 85 M HIGH LOW 46.5 drugY
age = 85
sex = "M"
bp = "HIGH"
cholesterol = "LOW"
na_to_k = 46.5

# Getting the Prediction
predicted_drug = predict_drug(age, sex, bp, cholesterol, na_to_k)
```

```python
# Printing the Result
print(f"Predicted Drug: {predicted_drug}")
```

```
Predicted Drug: drugY
```

```python
# Input: 75 M NORMAL NORMAL 12.33 drugX
age = 75
sex = "M"
bp = "NORMAL"
cholesterol = "NORMAL"
na_to_k = 12.33

# Getting the Prediction
predicted_drug = predict_drug(age, sex, bp, cholesterol, na_to_k)

# Printing the Result
print(f"Predicted Drug: {predicted_drug}")
```

```
Predicted Drug: drugX
```

```python
# Input:   19    F     HIGH  HIGH  13.313     drugA
age = 19
sex = "F"
bp = "HIGH"
cholesterol = "HIGH"
na_to_k = 13.313

# Getting the Prediction
predicted_drug = predict_drug(age, sex, bp, cholesterol, na_to_k)

# Printing the Result
print(f"Predicted Drug: {predicted_drug}")
```

```
Predicted Drug: drugA
```

```python
# Input:   60    F     HIGH  HIGH  13.303     drugB
age = 60
sex = "F"
bp = "HIGH"
cholesterol = "HIGH"
na_to_k = 13.303

# Getting the Prediction
predicted_drug = predict_drug(age, sex, bp, cholesterol, na_to_k)

# Printing the Result
print(f"Predicted Drug: {predicted_drug}")
```

```
Predicted Drug: drugB
```

INTERACTIVE INPUT SYSTEM

```
print("\n=== DRUG PREDICTION SYSTEM ===\n")
# Input Values
age = int(input("Enter Age: "))
sex = input("Enter Sex (M/F): ").strip().upper()
bp = input("Enter Blood Pressure (LOW/NORMAL/HIGH): ").strip().upper()
cholesterol = input("Enter Cholesterol Level (LOW/HIGH):
").strip().upper()
na_to_k = float(input("Enter Sodium to Potassium Ratio: "))

# Get Prediction
predicted_drug = predict_drug(age, sex, bp, cholesterol, na_to_k)

# Display Output
print("\n==============================")
print(f"Predicted Drug: {predicted_drug}")
print("==============================\n")
```

```
=== DRUG PREDICTION SYSTEM ===

Enter Age: 30
Enter Sex (M/F): M
Enter Blood Pressure (LOW/NORMAL/HIGH): Low
Enter Cholesterol Level (LOW/HIGH): high
Enter Sodium to Potassium Ratio: 23.545

==============================
Predicted Drug: drugY
==============================
```