Experiment 2.4

Student Name: YANA SRIVASTAVA **UID: 20BCS2279**
Branch: CSE **Section/Group:20BCS_WM_906/B**
Semester: 5th **Date of Performance: 04-11-22**
Subject Name: Machine Learning Lab **Subject Code: 21CSP-317**

1. Aim/Overview of the practical:

Decision Trees on digits dataset**.**

2. Source Code:

In [88]:
```
import pandas
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt
```

In [89]:
```
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_digits
```

In [90]:
```
digitsData = load_digits()
```

In [91]:
```
X = digitsData.data
y = digitsData.target
```

In [92]:
```
X
```

Out[92]:
```
array([[ 0.,  0.,  5., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ..., 10.,  0.,  0.],
       [ 0.,  0.,  0., ..., 16.,  9.,  0.],
       ...,
       [ 0.,  0.,  1., ...,  6.,  0.,  0.],
       [ 0.,  0.,  2., ..., 12.,  0.,  0.],
       [ 0.,  0., 10., ..., 12.,  1.,  0.]])
```

In [93]:
```
y
```

Out[93]:
```
array([0, 1, 2, ..., 8, 9, 8])
```

In [94]:
```
X_train, X_test, y_train, y_test = train_test_split(
X, y, test_size = 0.3, random_state=50)
```

In [95]:
```
dtree = DecisionTreeClassifier()
dtree.fit(X_train, y_train)
```

Out[95]:
```
DecisionTreeClassifier()
```

In [96]:
```
y_pred = dtree.predict(X_test)
```

In [97]:
```
y_pred
```

Out[97]:
```
array([2, 3, 4, 6, 2, 0, 4, 1, 3, 7, 4, 6, 1, 2, 6, 6, 3, 6, 9, 9, 0, 0,
       8, 2, 0, 2, 5, 4, 0, 7, 6, 5, 3, 0, 6, 2, 1, 6, 4, 3, 0, 2, 0, 5,
       4, 6, 8, 9, 7, 2, 9, 5, 7, 5, 5, 6, 4, 0, 8, 5, 8, 9, 0, 1, 9, 8,
       8, 2, 1, 5, 9, 9, 5, 5, 7, 0, 2, 8, 1, 3, 5, 6, 7, 1, 4, 8, 2, 5,
       6, 1, 8, 1, 4, 7, 4, 2, 8, 4, 9, 3, 9, 0, 4, 6, 6, 6, 2, 0, 8, 0,
       2, 5, 2, 0, 2, 4, 9, 0, 6, 9, 2, 5, 1, 3, 7, 6, 4, 3, 0, 9, 1, 9,
       8, 7, 7, 5, 0, 9, 2, 1, 1, 5, 9, 7, 9, 9, 0, 9, 0, 8, 7, 0, 9, 2,
       3, 3, 5, 9, 2, 2, 0, 0, 8, 3, 9, 2, 8, 2, 2, 2, 4, 6, 9, 2, 4, 3,
       7, 8, 2, 1, 7, 7, 8, 4, 1, 7, 2, 4, 6, 6, 6, 0, 8, 0, 6, 1, 5, 7,
       7, 0, 3, 9, 6, 8, 1, 9, 1, 3, 7, 6, 7, 4, 3, 2, 2, 4, 2, 0, 0, 8,
       6, 5, 8, 1, 0, 4, 2, 8, 1, 5, 7, 6, 8, 9, 1, 8, 1, 0, 2, 7, 0, 7,
       5, 4, 7, 0, 9, 3, 0, 6, 2, 2, 9, 2, 2, 8, 0, 9, 1, 1, 6, 5, 0, 1,
       4, 6, 1, 5, 7, 8, 6, 6, 5, 0, 1, 3, 6, 5, 3, 2, 2, 9, 4, 8, 1, 5,
       9, 8, 2, 3, 5, 8, 9, 7, 7, 7, 7, 7, 3, 5, 2, 4, 2, 8, 3, 3, 7, 1,
       1, 5, 5, 3, 4, 1, 4, 1, 0, 2, 2, 0, 2, 7, 9, 4, 9, 7, 3, 4, 4, 1,
       0, 1, 9, 6, 5, 3, 8, 9, 5, 4, 7, 8, 6, 3, 2, 4, 2, 1, 6, 5, 7, 6,
       0, 9, 2, 4, 4, 1, 4, 0, 0, 0, 5, 1, 5, 2, 6, 7, 0, 7, 3, 5, 0, 1,
       8, 0, 6, 5, 6, 2, 9, 5, 8, 4, 9, 9, 6, 0, 3, 6, 9, 8, 5, 3, 4, 5,
       7, 4, 0, 6, 5, 7, 0, 5, 0, 3, 6, 5, 6, 6, 7, 3, 1, 6, 9, 2, 5, 8,
       8, 8, 9, 1, 1, 4, 9, 1, 6, 1, 2, 9, 3, 5, 7, 7, 7, 4, 0, 1, 6, 8,
       0, 7, 8, 7, 4, 3, 9, 6, 7, 7, 5, 6, 0, 3, 4, 0, 9, 0, 5, 1, 5, 7,
       1, 4, 0, 8, 4, 4, 9, 4, 6, 5, 4, 8, 1, 4, 2, 4, 9, 9, 3, 2, 9, 5,
       2, 9, 3, 7, 9, 8, 4, 6, 8, 4, 8, 7, 2, 4, 0, 4, 8, 5, 4, 7, 0, 5,
       9, 4, 0, 4, 3, 0, 8, 7, 2, 6, 0, 3, 1, 9, 0, 3, 4, 7, 1, 4, 8, 1,
       2, 5, 8, 2, 7, 2, 1, 7, 5, 5, 6, 6])
```

In [98]:
```
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
cm_DT = confusion_matrix(y_test, y_pred)
print(f"Confusion Matrix for DT:\n{cm_DT}\n")
acc_DT = accuracy_score(y_test, y_pred)
print(f"Accuracy Score: {acc_DT}")
```

```
Confusion Matrix for DT:
[[61  0  0  0  0  0  0  0  1  1]
 [ 0 44  0  0  4  0  3  1  4  3]
 [ 0  1 51  1  0  0  0  0  2  0]
 [ 0  1  2 32  0  3  0  2  3  2]
 [ 0  2  0  0 47  1  2  2  0  1]
 [ 0  0  1  0  0 47  1  1  3  1]
 [ 1  0  0  2  1  0 48  0  0  1]
 [ 0  0  0  2  2  2  1 45  0  1]
 [ 0  2  6  1  1  1  0  1 34  1]
 [ 0  1  0  2  3  1  0  3  2 44]]

Accuracy Score: 0.8388888888888889
```

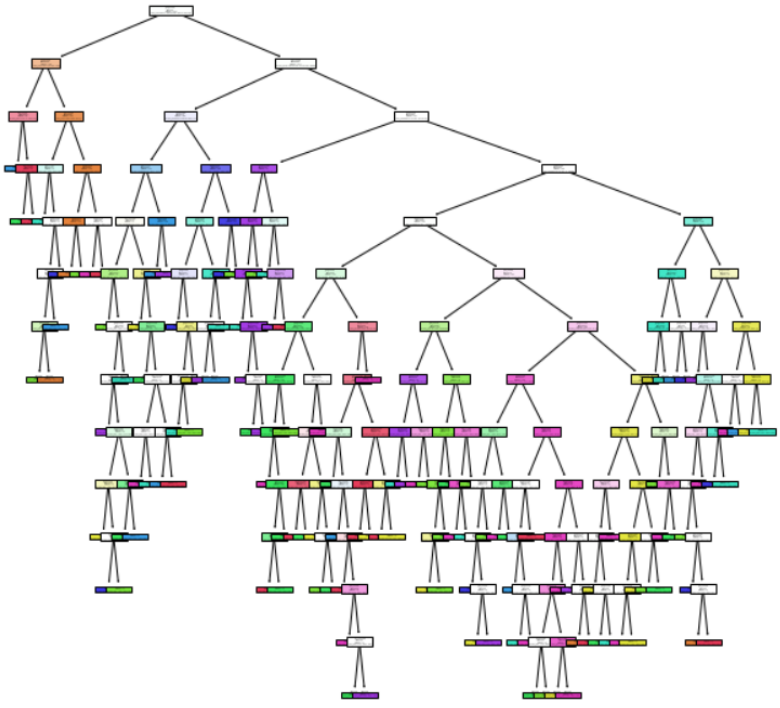
In [99]:
```
print(f"Classification Report for DT:\n{classification_report(y_test, y_pred)}\n")
```

```
Classification Report for DT:
              precision    recall  f1-score   support

           0       0.98      0.97      0.98        63
           1       0.86      0.75      0.80        59
           2       0.85      0.93      0.89        55
           3       0.80      0.71      0.75        45
           4       0.81      0.85      0.83        55
           5       0.85      0.87      0.86        54
           6       0.87      0.91      0.89        53
           7       0.82      0.85      0.83        53
           8       0.69      0.72      0.71        47
           9       0.80      0.79      0.79        56

    accuracy                           0.84       540
   macro avg       0.83      0.83      0.83       540
weighted avg       0.84      0.84      0.84       540
```

In [100]:
```
plt.figure(figsize=(10, 10)) # Resize figure
tree.plot_tree(dtree, filled=True)
plt.show()
```



In [101]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [102]:
```
classifier_rf = RandomForestClassifier(random_state=0, n_jobs=-1, max_depth=10,
n_estimators=100, oob_score=True)
```

In [103]:
```
classifier_rf.fit(X_train, y_train)
```

Out[103]:
```
RandomForestClassifier(max_depth=10, n_jobs=-1, oob_score=True, random_state=0)
```

In [104]:
```
y_pred_RF = classifier_rf.predict(X_test)
```

In [105]:
```
y_pred_RF
```

Out[105]:
```
array([8, 3, 7, 6, 2, 0, 4, 1, 9, 7, 4, 6, 1, 2, 6, 6, 3, 6, 9, 9, 0, 0,
       8, 2, 0, 2, 7, 4, 0, 7, 6, 5, 3, 0, 1, 2, 1, 6, 4, 3, 0, 2, 0, 5,
       4, 6, 8, 9, 7, 5, 9, 5, 7, 5, 2, 6, 4, 0, 1, 5, 0, 3, 0, 1, 1, 8,
       8, 8, 1, 4, 9, 9, 5, 5, 7, 0, 2, 1, 4, 3, 5, 6, 7, 1, 4, 8, 2, 5,
       4, 1, 8, 1, 9, 3, 4, 2, 1, 4, 9, 3, 9, 0, 4, 6, 6, 6, 2, 0, 3, 0,
       3, 5, 2, 0, 2, 4, 9, 0, 6, 9, 2, 5, 2, 3, 7, 6, 4, 3, 0, 9, 1, 3,
       8, 7, 7, 5, 0, 9, 8, 1, 1, 5, 9, 7, 9, 9, 6, 9, 0, 8, 7, 0, 9, 2,
       3, 8, 5, 9, 2, 8, 0, 0, 8, 3, 9, 2, 9, 2, 2, 2, 4, 6, 9, 2, 4, 3,
       7, 8, 2, 1, 7, 7, 2, 4, 1, 7, 2, 4, 6, 6, 6, 0, 9, 0, 6, 1, 7, 7,
       7, 0, 3, 9, 6, 8, 9, 9, 1, 3, 7, 1, 7, 4, 6, 2, 2, 4, 2, 0, 0, 1,
       6, 5, 8, 1, 0, 4, 2, 8, 1, 5, 4, 6, 8, 9, 1, 8, 1, 0, 2, 7, 0, 7,
       5, 8, 7, 0, 9, 3, 0, 6, 3, 2, 9, 2, 2, 1, 0, 9, 1, 8, 6, 5, 0, 1,
       4, 6, 1, 5, 5, 8, 6, 6, 5, 0, 1, 6, 6, 5, 9, 2, 2, 9, 4, 3, 1, 9,
       9, 8, 2, 3, 5, 8, 9, 7, 7, 7, 7, 7, 3, 5, 2, 1, 2, 2, 3, 2, 3, 1,
       1, 5, 5, 3, 4, 1, 4, 1, 0, 2, 2, 0, 2, 7, 1, 4, 0, 7, 3, 1, 1, 1,
       0, 1, 9, 6, 3, 3, 4, 9, 5, 4, 7, 8, 6, 7, 2, 6, 2, 1, 6, 5, 7, 6,
       0, 9, 2, 7, 4, 1, 4, 0, 0, 0, 5, 1, 5, 2, 6, 7, 0, 7, 3, 5, 0, 1,
       8, 0, 6, 5, 6, 8, 9, 5, 8, 4, 1, 7, 6, 0, 3, 6, 9, 8, 5, 3, 9, 5,
       7, 4, 0, 6, 5, 7, 0, 5, 0, 7, 7, 5, 6, 6, 4, 3, 1, 6, 9, 2, 5, 8,
       8, 3, 9, 4, 1, 4, 9, 1, 6, 8, 8, 9, 3, 5, 7, 7, 1, 4, 0, 1, 6, 8,
       0, 7, 8, 7, 4, 3, 9, 6, 7, 7, 5, 6, 0, 3, 4, 0, 9, 0, 5, 1, 5, 7,
       1, 4, 0, 9, 4, 4, 9, 4, 6, 5, 4, 8, 1, 4, 2, 4, 8, 3, 3, 2, 9, 5,
       2, 9, 3, 7, 0, 2, 4, 4, 8, 4, 8, 7, 2, 4, 0, 1, 8, 5, 4, 7, 0, 5,
       5, 4, 0, 9, 3, 0, 8, 7, 2, 6, 0, 3, 1, 9, 0, 3, 4, 7, 1, 4, 8, 1,
       2, 5, 8, 2, 1, 2, 1, 7, 5, 5, 6, 1])
```

In [106]:
```
# Confusion Matrix
cm_RF= confusion_matrix(y_test, y_pred_RF)
print(f"Confusion Matrix for RF:\n{cm_RF}\n")
```

```
Confusion Matrix for RF:
[[62  0  0  0  1  0  0  0  0  0]
 [ 0 59  0  0  0  0  0  0  0  0]
 [ 0  0 55  0  0  0  0  0  0  0]
 [ 0  0  0 42  0  2  0  0  1  0]
 [ 0  0  0  0 54  0  0  1  0  0]
 [ 1  0  0  0  0 49  1  0  0  3]
 [ 1  0  0  0  0  0 52  0  0  0]
 [ 0  0  0  0  0  0  0 52  0  1]
 [ 0  3  2  0  0  0  0  0 42  0]
 [ 0  0  0  1  0  1  0  3  1 50]]
```

In [21]:
```
# Accuracy Score
acc_RF = accuracy_score(y_test, y_pred_RF)
print(f"Accuracy Score: {acc_RF}")
```

```
Accuracy Score: 0.9555555555555556
```

In [22]:
```
# Classification Report
print(f"Classification Report for RF:\n{classification_report(y_test,y_pred_RF)}\n")
```

```
Classification Report for RF:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        14
           1       0.94      0.94      0.94        17
           2       0.93      0.93      0.93        14

    accuracy                           0.96        45
   macro avg       0.96      0.96      0.96        45
weighted avg       0.96      0.96      0.96        45
```

Learning outcomes (What I have learnt):

1. Learn about the Decision Trees.
2. Learn to perform the decision tree on digits dataset.
3. Learnt about the exploratory data analysis.
4. Learn to optimize the Model.