# DecisionTree_RandomForest

January 20, 2025

[98]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix,␣
↪classification_report
```

[99]:
```
df = pd.read_csv("./glass.csv")
```

[100]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
scaler = StandardScaler()
```

[101]:
```
s =␣
↪"RI        Na        Mg        Al        Si        K        Ca        Ba        Fe        T
columns = s.split("\t")
```

[102]:
```
X = df[['RI', 'Na', 'Mg', 'Al', 'Si', 'K', 'Ca', 'Ba', 'Fe']].to_numpy()
y = df[["Type"]].to_numpy()
```

[103]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y,␣
↪random_state=22053747, test_size=0.2)
```

[104]:
```
y_test = y_test.ravel()
y_train = y_train.ravel()
```

[105]:
```
from sklearn.preprocessing import StandardScaler

scaler_X = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
```

[106]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier, plot_tree
```

[107]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
DT = DecisionTreeClassifier(criterion='entropy', random_state=42)
DT.fit(X_train, y_train)
```

[107]:
```
DecisionTreeClassifier(criterion='entropy', random_state=42)
```

[108]:
```
y_pred = dt.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.7674418604651163
Confusion Matrix:
[[10  0  0  0  1  0]
 [ 4  8  1  0  1  0]
 [ 1  0  2  0  0  0]
 [ 0  1  0  3  0  0]
 [ 0  0  0  0  2  1]
 [ 0  0  0  0  0  8]]
Classification Report:
              precision    recall  f1-score   support

           1       0.67      0.91      0.77        11
           2       0.89      0.57      0.70        14
           3       0.67      0.67      0.67         3
           5       1.00      0.75      0.86         4
           6       0.50      0.67      0.57         3
           7       0.89      1.00      0.94         8

    accuracy                           0.77        43
   macro avg       0.77      0.76      0.75        43
weighted avg       0.80      0.77      0.76        43
```

Random Forest

[109]:
```
classifier = RandomForestClassifier(max_leaf_nodes=20, n_estimators=500)
classifier.fit(X=X_train, y=y_train.ravel())
```

[109]:
```
RandomForestClassifier(max_leaf_nodes=20, n_estimators=500)
```

[110]:
```
y_pred_rf = rf.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred_rf))
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred_rf))
```

```
print("Classification Report:")
print(classification_report(y_test, y_pred_rf))
```

```
Accuracy: 0.8372093023255814
Confusion Matrix:
[[11  0  0  0  0  0]
 [ 4  9  0  0  0  1]
 [ 1  0  2  0  0  0]
 [ 0  1  0  3  0  0]
 [ 0  0  0  0  3  0]
 [ 0  0  0  0  0  8]]
Classification Report:
              precision    recall  f1-score   support

           1       0.69      1.00      0.81        11
           2       0.90      0.64      0.75        14
           3       1.00      0.67      0.80         3
           5       1.00      0.75      0.86         4
           6       1.00      1.00      1.00         3
           7       0.89      1.00      0.94         8

    accuracy                           0.84        43
   macro avg       0.91      0.84      0.86        43
weighted avg       0.87      0.84      0.83        43
```

[111]:
```
new_sample = np.array([[1.8, 13.45, 4.1, 1.9, 69.12, 0.42, 8.29, 0.0, 0.0]])
predicted_class_rf = classifier.predict(new_sample)
print("Predicted Class:", target_names[int(predicted_class_rf[0]) - 1])
```

```
Predicted Class: 2
```