Artificial Intelligence

Assignment 3

Classification using Decision Trees and Random Forests

Q.1. Implement the decision tree classifier using ID-3 algorithm for the Mushroom Dataset (agaricus-lepiota.data). Use 80/20 split to create Train/Test sets and report the performance of your model. Moreover, use appropriate measures to handle attributes with missing values.

Q.2. Implement a random forest classifier for the classification problem in Q.1. Use the square root of the total features (rounded up) to induce individual trees.

Loading Dataset

- The dataset provided does not contain the column header to identify attributes so the columns (attributes) are named using the information in the Dataset Description

```
import pandas as pd

attributes = [
    "class", "cap-shape", "cap-surface", "cap-color", "bruises",
"odor",
    "gill-attachment", "gill-spacing", "gill-size", "gill-color",
    "stalk-shape", "stalk-root", "stalk-surface-above-ring",
    "stalk-surface-below-ring", "stalk-color-above-ring",
    "stalk-color-below-ring", "veil-type", "veil-color",
    "ring-number", "ring-type", "spore-print-color", "population",
    "habitat"
]

df = pd.read_csv('agaricus-lepiota.data', names=attributes)
```

Handling Missing Values

- According to the Dataset Description, the attribute **stalk-root** contains missing values represented by **?**

```
df['stalk-root'].value_counts()
```

```
stalk-root
b    3776
?    2480
e    1120
```

```
c     556
r     192
Name: count, dtype: int64
```

- I am using the **mode value** (most frequent value) of the column and use it for imputing the missing values.

```
mode = df['stalk-root'].mode()[0]

df['stalk-root'].replace('?', mode)
df.iloc[:5,11]

0    e
1    c
2    c
3    e
4    e
Name: stalk-root, dtype: object
```

Encoding Data

- Since the dataset contains categorical data, I am **encoding** the values for training the model.

```
encode = df.drop('class', axis=1)
encode = pd.get_dummies(encode, drop_first=True)
```

- Using **0** for edible class and **1** for poisonous class

```
mapping = {'e':0, 'p':1}
target = df['class'].map(mapping)
```

Now data has been processed for training.

Decision Tree Classifier

- Using **80/20** split to create training and testing data.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(encode, target,
test_size=0.2, random_state=42)
```

- **Training** the model

```
from sklearn.tree import DecisionTreeClassifier

decision_tree = DecisionTreeClassifier(criterion='entropy',
random_state=42)
decision_tree.fit(X_train, y_train)

DecisionTreeClassifier(criterion='entropy', random_state=42)
```

- **Testing** the model and calculating **accuracy**

```
from sklearn.metrics import accuracy_score

y_pred = decision_tree.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)

print(f"The accuracy of the decision tree model is {accuracy * 100}
%")
```

```
The accuracy of the decision tree model is 100.0 %
```

Random Forest Classifier

- Intializing a random forest classifier that uses **square-root** of the total features for tree induction

```
from sklearn.ensemble import RandomForestClassifier

random_forest = RandomForestClassifier(n_estimators=70,
max_features='sqrt', random_state=42)
```

- **Training** the model

```
random_forest.fit(X_train, y_train)
```

```
RandomForestClassifier(n_estimators=70, random_state=42)
```

- **Testing** the model and calculating **accuracy**

```
y_pred = random_forest.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)

print(f"The accuracy of the random forest model is {accuracy * 100}
%")
```

```
The accuracy of the random forest model is 100.0 %
```