Lab-2

(2) Use an appropriate data set for building the decision tree (ID3) and apply this knowledge to classify a new sample.

Step 1: Import necessary libraries.

import pandas as pd.
from sklearn.model_selection import train_test_split.
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn import metrics.
import matplotlib.pyplot as plt.

Step 2: Load the weather dataset from the csv file.

df = pd.read_csv("/content/tennis.csv")

Step 3: display the first few rows of the dataset to understand its structure.

print("First 5 rows of the dataset:")
print(df.head())

Step 4: pre processing

~~le_outlook~~ = LabelEncoder()
le_temp = LabelEncoder()
le_humidity = LabelEncoder()
le_windy = LabelEncoder()
le_play = LabelEncoder()

Encoding the relevant column (all lowercase at dataset) for your

```
df['outlook'] = le_outlook.fit_transform(df['outlook'])
df['temp'] = le_temp.fit_transform(df['temp'])
df['humidity'] = le_humidity.fit_transform(df['humidity'])
df['windy'] = le_windy.fit_transform(df['windy'])
df['play'] = le_play.fit_transform(df['play'])
```

#Step 5: Define features (X) and the target variable (y).

```
X = df[['outlook', 'temp', 'humidity', 'windy']]
y = df['play']
```

#Step 6: Split the dataset.

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.2, random_state=42)
```

Step 7: Create a Decision tree classifier.

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

Step 8: Train the classifier

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

step 9: predict the labels.

```
y_pred = clf.predict(X_test)
```

Step 10: Evaluate the model's accuracy.

```
accuracy = metrics.accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy*100:.2f}%")
```

Step 11: Classify a new weather sample.

```
new_sample = [[le_outlook.transform(['Sunny'])[0],
               le_temp.transform(['Cool'])[0],
               le_humidity.transform(['High'])[0],le_
               windy.transform(['False'])[0]]]
```

```
predicted_class = clf.predict(new_sample)

# output the predicted class.
print(f"predicted class for the new sample: {'yes' if
    le_play.inverse_transform([predicted_class[0]])
    [0] == 'yes' else 'no'}")

# Step 12: Visualize the decision tree.
plt.figure(figsize=(12,8))
plot_tree(clf, feature_names=['outlook', 'temp', 'humidity',
          'windy'], class_names=le_play.classes_,
          filled=True, rounded=True)
plt.title("Decision tree for weather data")
plt.show()
```

Output:

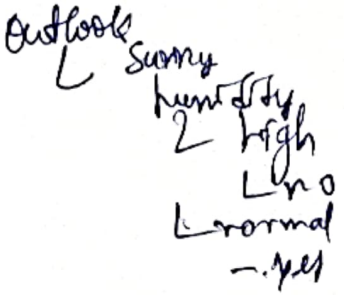
First 5 rows of the dataset:

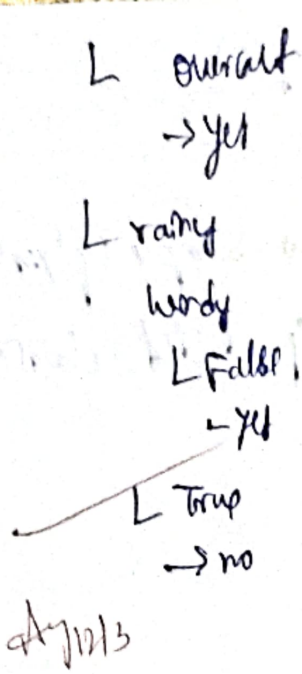
	Outlook	temp	humidity	windy	play
0	Sunny	hot	high	False	no
1	Sunny	hot	high	true	no
2	Overcast	hot	high	False	yes
3	rainy	mild	high	False	yes
4	rainy	cool	normal	False	yes

Accuracy: 100.00%

predicted class for the new Sample: no

Decision tree:





Ans)13