I. Perform the following tasks and record results by applying Decision Tree Algorithm (Gini Index impurity). Use the Tennis dataset from the link below for each task:
https://github.com/25-Madhuri/Dataset/blob/main/tennis.csv

<u>Sample Code:</u>

```
# Importing necessary libraries
import pandas as pd    # For data manipulation and analysis
import numpy as np     # For numerical operations
import matplotlib.pyplot as plt   # For plotting graphs


# Importing necessary modules from sklearn
from sklearn.preprocessing import LabelEncoder  # For converting categorical data to numerical data
from sklearn.model_selection import train_test_split  # For splitting the data into training and test sets
from sklearn import tree  # For creating decision tree model
from sklearn.tree import DecisionTreeClassifier, plot_tree  # Decision tree classifier for model and plot_tree for visualization.


# Loading the dataset from a CSV file
df = pd.read_csv('/content/tennis.csv')  # Read the CSV file and store it in the dataframe 'df'


# Displaying the dataframe to check the data
df  # This will show the first few rows of the dataset


# Initializing the LabelEncoder
le = LabelEncoder()


# Converting categorical columns to numerical values
df['outlook_n'] = le.fit_transform(df['outlook'])  # Converting 'outlook' column to numeric values
```

```
df['temp_n'] = le.fit_transform(df['temp'])  # Converting 'temp' column to numeric values

df['humidity_n'] = le.fit_transform(df['humidity'])  # Converting 'humidity' column to numeric values

df['windy_n'] = le.fit_transform(df['windy'])  # Converting 'windy' column to numeric values

df['play_n'] = le.fit_transform(df['play'])  # Converting 'play' column (target) to numeric values (0 or 1)


# Displaying the dataframe after label encoding

df  # This will show the dataframe with new numeric columns


# Dropping the original categorical columns after encoding

df = df.drop(['outlook', 'temp', 'humidity', 'windy', 'play'], axis='columns')  # Dropping the non-numeric columns


# Displaying the dataframe after dropping the original columns

df  # This will show the dataframe with only the numeric columns


# Splitting the data into independent (features) and dependent (target) variables

independent_variable = df.drop('play_n', axis='columns')  # Dropping the target column 'play_n' to use the remaining as features

dependent_variable = df['play_n']  # The target variable is 'play_n', which tells if tennis will be played (0 or 1)


# Initializing the Decision Tree Classifier model

model = tree.DecisionTreeClassifier()  # Creating the decision tree model


# Fitting the model on the data

model.fit(independent_variable, dependent_variable)  # Training the decision tree on the data


# Evaluating the model performance on the training data
```

```
model_score = model.score(independent_variable, dependent_variable)  # Calculate the accuracy of the model on the training set


# Displaying the model accuracy
model_score  # This will print the accuracy score of the model on the training data


# Making a prediction using the trained model
prediction = model.predict([[1, 2, 0, 1]])  # Predicting the target (play or not) for a given set of input features


# Displaying the predicted result
prediction  # This will print the prediction for the input features [outlook=1, temp=2, humidity=0, windy=1]


# Visualizing the decision tree
plt.figure(figsize=(120, 80))  # Adjusting the figure size for better readability


# Plotting the decision tree
plot_tree(model, filled=True, feature_names=independent_variable.columns, class_names=le.classes_)  # Plotting the tree with feature and class names
plt.show()  # Displaying the tree visualization.
```

Conclusion:

The Decision Tree model has been successfully trained to predict whether tennis will be played based on weather conditions like outlook, temperature, humidity, and wind.The model's accuracy can be evaluated, and it is able to make predictions for new instances based on the learned patterns.Visualizing the decision tree helps in understanding the decision-making process, where each node represents a decision based on a specific feature, and the leaf nodes show the final prediction.