NAME: RUCHIR MAKARAND ADNAIK  
ROLL: 08  
SRN: 202200282  
DIV: B  
BATCH: B1

ASSIGNMENT 10

CODE:

import numpy as np

import pandas as pd

from sklearn.tree import DecisionTreeRegressor, plot\_tree

from sklearn.model\_selection import train\_test\_split, GridSearchCV

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score

import seaborn as sns

import matplotlib.pyplot as plt

import warnings

# Ignore warnings for cleaner output

warnings.filterwarnings("ignore")

# Load the dataset

df = pd.read\_csv("expenses.csv")

# Display dataset info

print(df.info())

# Check for missing values

print(df.isna().sum())

# Converting categorical columns into numeric

df['sex'].replace({"male": 0, "female": 1}, inplace=True)

sex\_value = {"female": 1, "male": 0}

print("Sex Value Encoding:", sex\_value)

df['smoker'].replace({"yes": 0, "no": 1}, inplace=True)

smoker\_value = {"no": 1, "yes": 0}

print("Smoker Value Encoding:", smoker\_value)

# Convert 'region' column into dummy variables (one-hot encoding)

df = pd.get\_dummies(df, columns=['region'])

print(df.head())

# Define feature matrix (X) and target variable (y)

X = df.drop("charges", axis=1)  # Dropping the target variable 'charges'

y = df["charges"]  # Target variable

# Split the data into training and testing sets (80% training, 20% testing)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Initialize and train the Decision Tree Regressor

dt\_reg = DecisionTreeRegressor()

dt\_reg.fit(X\_train, y\_train)

# Make predictions on the test set

y\_pred = dt\_reg.predict(X\_test)

# Evaluate the model

mae = mean\_absolute\_error(y\_test, y\_pred)

mse = mean\_squared\_error(y\_test, y\_pred)

rmse = np.sqrt(mse)

r2 = r2\_score(y\_test, y\_pred)

print(f"Mean Absolute Error (MAE): {mae}")

print(f"Mean Squared Error (MSE): {mse}")

print(f"Root Mean Squared Error (RMSE): {rmse}")

print(f"R² Score: {r2}")

# Plot the Decision Tree

plt.figure(figsize=(45, 40))

plot\_tree(dt\_reg, filled=True, feature\_names=X.columns)

plt.title("Decision Tree")

plt.show()

# Training error analysis

y\_pred\_train = dt\_reg.predict(X\_train)

mse\_train = mean\_squared\_error(y\_train, y\_pred\_train)

print("Mean Squared Error (Train) --->", mse\_train)

mae\_train = mean\_absolute\_error(y\_train, y\_pred\_train)

print("Mean Absolute Error (Train) --->", mae\_train)

rmse\_train = np.sqrt(mse\_train)

print("Root Mean Square Error (Train) --->", rmse\_train)

r2\_train = r2\_score(y\_train, y\_pred\_train)

print("R² Score (Train) --->", r2\_train)

# Display feature column names

column\_names = X.columns

print(column\_names)

# Create JSON-like data structure for model features and encodings

json\_data = {

    "sex": sex\_value,

    "smoker\_value": smoker\_value,

    "columns": list(column\_names)

}

print(json\_data)

# Define sample inputs for prediction

age = 21.0

sex = "female"

bmi = 33.7

children = 1.0

smoker = "no"

region = "northeast"

# Convert region to its corresponding column name in one-hot encoding

region = "region\_" + region

print(region)

# Find the index of the region column in the feature list

region\_index = list(column\_names).index(region)

print(region\_index)

# Prepare the input array for prediction

test\_array = np.zeros(len(column\_names))

test\_array[0] = age

test\_array[1] = json\_data['sex'][sex]

test\_array[2] = bmi

test\_array[3] = children

test\_array[4] = json\_data['smoker\_value'][smoker]

test\_array[region\_index] = 1

print(test\_array)

# Predict charges using the trained model

charges = round(dt\_reg.predict([test\_array])[0], 2)

print("Predicted Medical Insurance Charges is:", charges, "/- Rs. Only")

OUTPUT:

**DECISION TREE:**

A diagram of a company

Description automatically generated with medium confidence

A computer screen shot of a game

Description automatically generated