**EX.N0. IMPLEMENT DECISION TREE BASED ID3 ALGORITHM USING THE**

**DATE: DIABETES DATA SET FOR BUILDING THE DECISION TREE AND APPLY THIS**

**KNOWLEDGE TO CLASSIFY A NEW SAMPLE**

**AIM:**

To implement decision tree based id3 algorithm using the diabetes data set for building the decision tree and apply this knowledge to classify a new sample.

**ALGORITHM:**

**STEP-1:** Prepare Data:Import libraries and load the dataset, Separate the feature columns (x) and target column (y), where y is the "Diabetes\_binary" column.

**STEP-2:** Train-Test Split: Split data into training and testing sets (80-20 split). Set a random state to ensure reproducibility of results.

**STEP-3:** Train Model:Initialize a DecisionTreeClassifier with the Gini index as the criterion and set the maximum depth of the tree to 3.Fit the classifier to the training data using .fit().

**STEP-4:** Evaluate Model:Predict with test data and calculate accuracy. Calculate the accuracy using accuracy\_score() by comparing the predicted and actual values.

**STEP-5**: Visualize Tree: Plot the decision tree using plot\_tree(), setting parameters like feature names, class names, and Gini impurity values and display it.

**CODE :**

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.tree import DecisionTreeClassifier,plot\_tree

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

x=df.drop("Diabetes\_binary",axis=1)

y=df["Diabetes\_binary"]

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

from sklearn.metrics import accuracy\_score

xtrain,xtest,ytrain,ytest=train\_test\_split(x,y,test\_size=0.2,random\_state=42)

clf=DecisionTreeClassifier(criterion='gini',max\_depth=3, random\_state=42)

clf.fit(xtrain,ytrain)

y\_pred=clf.predict(xtest)

acc=accuracy\_score(ytest,y\_pred)

print(“accuracy:”,acc)

plt.figure(figsize=(20, 12))

plot\_tree(

clf,

feature\_names=x.columns,

class\_names=['0','1'],

filled=True,

rounded=True,

impurity=True, # show gini value

label='all', # show class + impurity + samples

precision=2

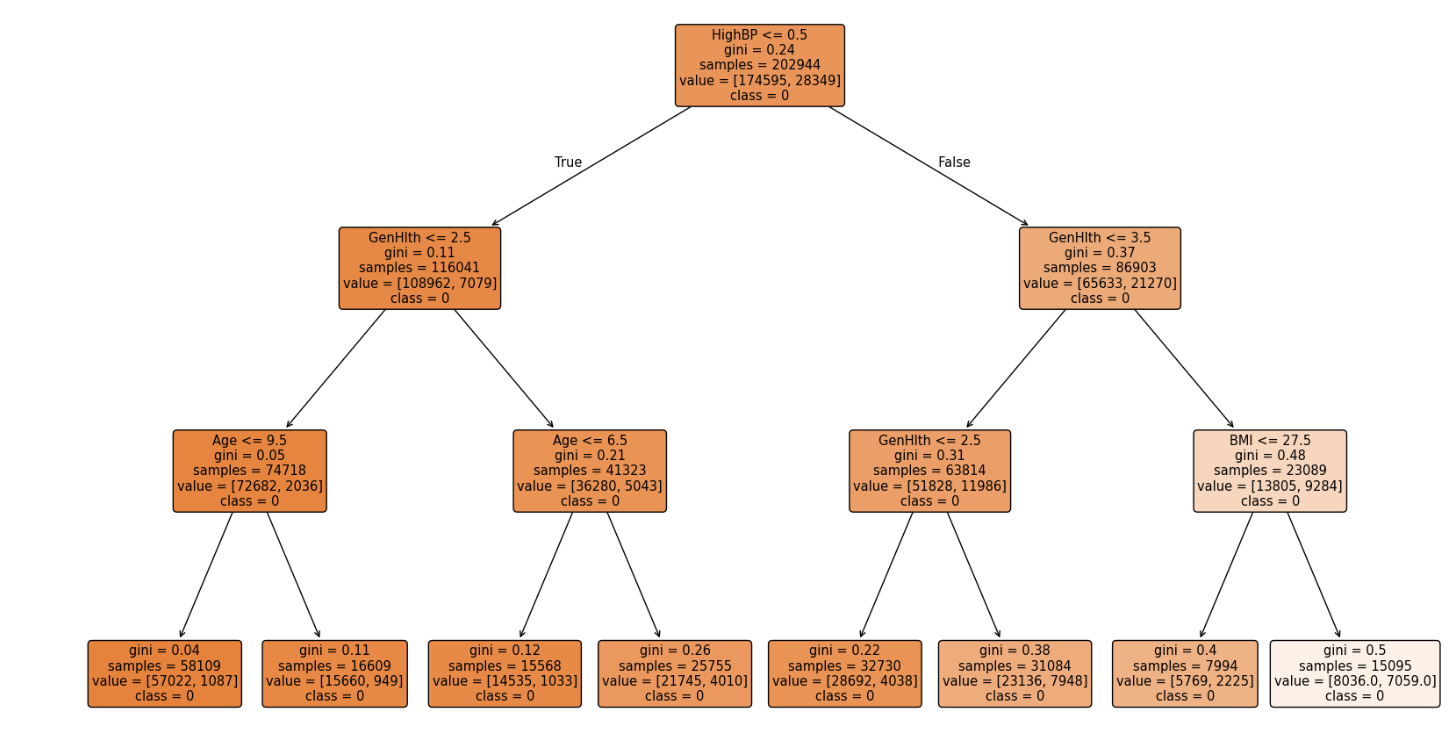
)

plt.title("Decision Tree (Gini)")

plt.show()

**OUTPUT:**

****

****

**RESULT:**

Thus the program to implement decision tree based id3 algorithm using the diabetes data set for building the decision tree and apply this knowledge to classify a new sample has been executed and output is verified successfully.

**EX.N0. IMPLEMENT XGBOOST REGRESSION TO PREDICT THE CAR PRICES**

**DATE: ANALYZE THE PERFORMANCE OF THE MODEL BY APPLYING VARIOUS**

**METRICS**

**AIM:**

To implement xgboost regression to predict the car prices analyze the performance of the model by applying various metrics.

**ALGORITHM:**

**STEP-1**: Load and preprocess data Read the CSV file, calculate car age, and apply one-hot encoding

to categorical columns.

**STEP-2:** Create binary labels, Define the target as 1 if Selling\_Price > 15, else 0.

**STEP-3:** Split the dataset, Use train\_test\_split to divide features and labels into training and test sets.

**STEP-4:** Train base models, Fit Logistic Regression and SVM models, then get their predicted

probabilities.

**STEP-5:** Stack and evaluate, Combine predictions, train an XGBoost classifier, predict on the test set,

and print accuracy and classification report.

**CODE :**

import pandas as pd

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

from sklearn.metrics import classification\_report

from sklearn.linear\_model import LogisticRegression

from sklearn.svm import SVC

from xgboost import XGBClassifier

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

df["age"]=2025-df["Year"]

x=df.drop(columns=['Selling\_Price','Car\_Name','Year'])

y=df['Selling\_Price']

x=pd.get\_dummies(x,columns=['Fuel\_Type','Seller\_Type' ,'Transmission'],drop\_first=True)

threshold = 15

y\_binary = (df['Selling\_Price'] > threshold).astype(int)

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

logreg\_model=LogisticRegression()

logreg\_model.fit(x\_train, y\_train)

svc\_model=SVC(probability=True)

svc\_model.fit(x\_train, y\_train)

x\_train\_logreg\_pred=pd.DataFrame(logreg\_model.predict\_proba(x\_train)[:, 1], columns=['logreg\_pred'])

x\_test\_logreg\_pred=pd.DataFrame(logreg\_model.predict\_proba(x\_test)[:, 1], columns=['logreg\_pred'])

x\_train\_combined=pd.concat([x\_train\_logreg\_pred, pd.DataFrame(svc\_model.predict\_proba(x\_train)[:, 1], columns=['svc\_pred'])], axis=1)

x\_test\_combined=pd.concat([x\_test\_logreg\_pred, pd.DataFrame(svc\_model.predict\_proba(x\_test)[:, 1], columns=['svc\_pred'])], axis=1)

xgb\_clf=XGBClassifier()

xgb\_clf.fit(x\_train\_combined, y\_train)

y\_pred=xgb\_clf.predict(x\_test\_combined)

accuracy=(y\_pred==y\_test).mean()

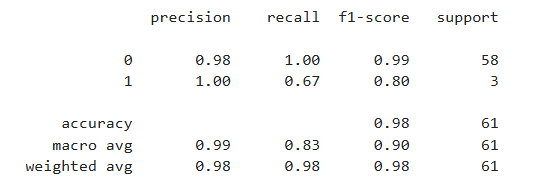
print("Accuracy=", accuracy)

print(classification\_report(y\_test,y\_pred))

**OUTPUT:**



Classification report:



**RESULT:**

Thus the program to implement xgboost regression to predict the car prices analyze the performance of the model by applying various metrics has been executed and output is verified successfully.