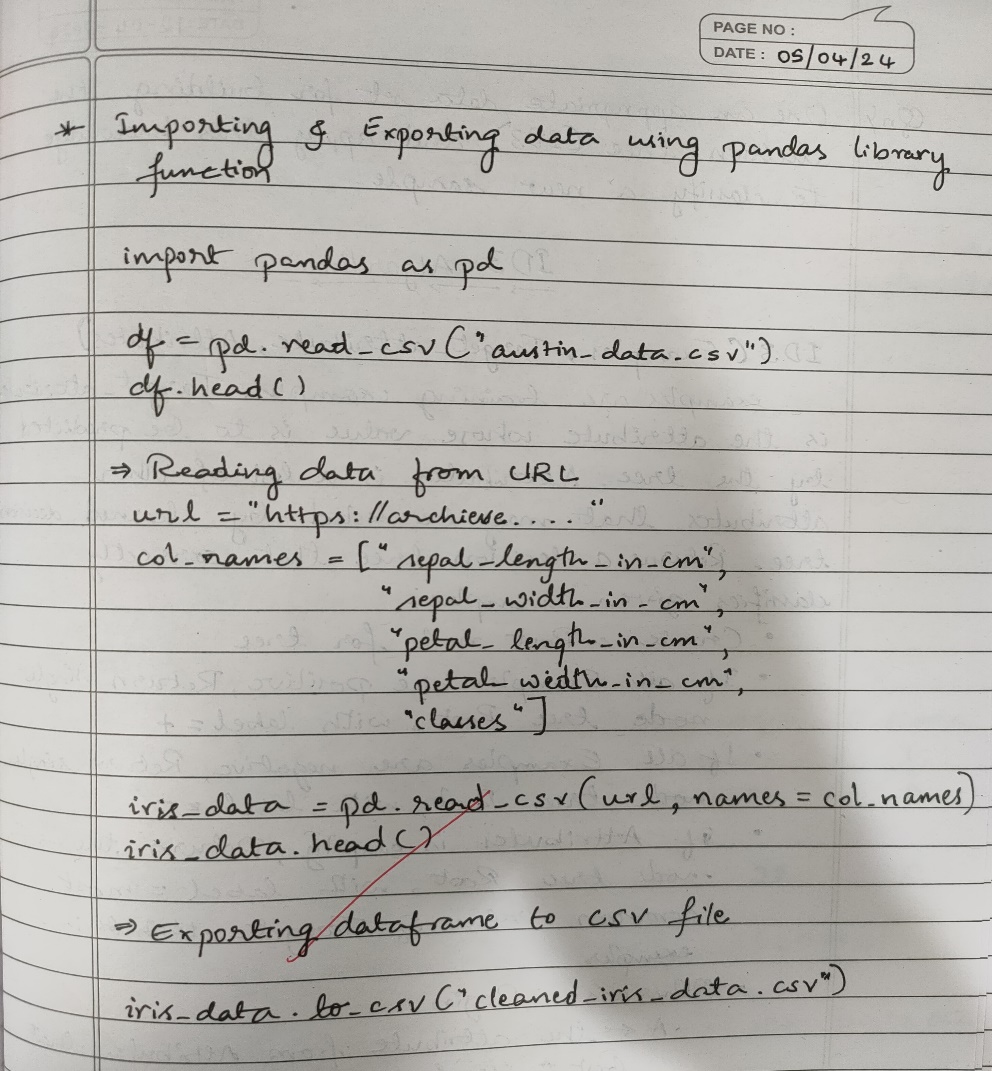
**Lab1**

**Date: 05/04/2024**

**Write a python program to import and export data using Pandas library functions**



**CODE:**

import pandas as pd

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

df.head()

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"

col\_names = ["sepal\_length\_in\_cm",

"sepal\_width\_in\_cm",

"petal\_length\_in\_cm",

"petal\_width\_in\_cm",

"class"]

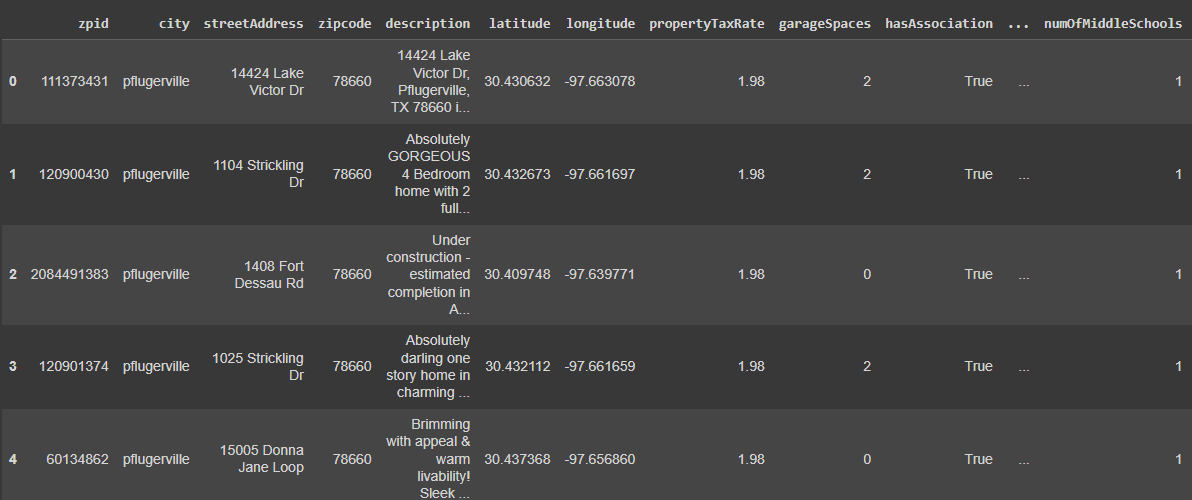
iris\_data = pd.read\_csv(url, names=col\_names)

iris\_data.head()

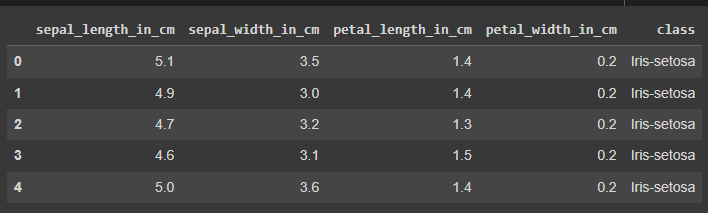
iris\_data.to\_csv("cleaned\_iris\_data.csv")

**OUTPUT :**

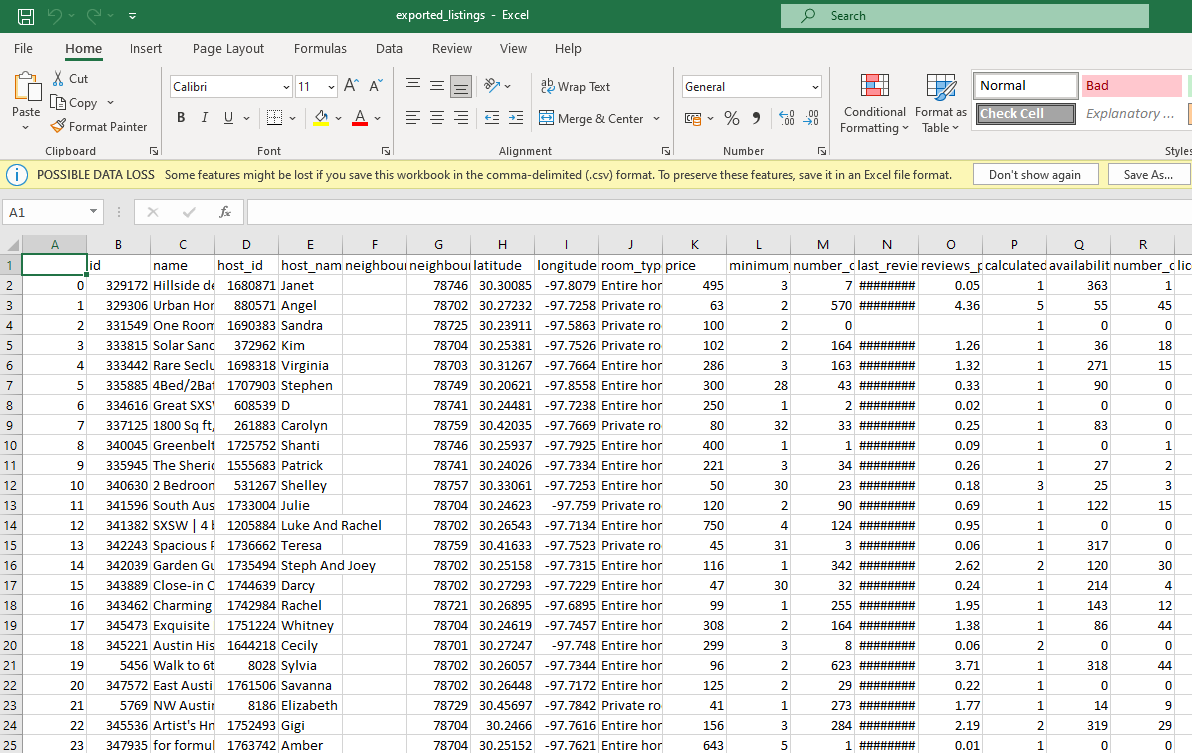
1. Dataset -



1. After reading dataset from URL –



1. CSV file after exporting –

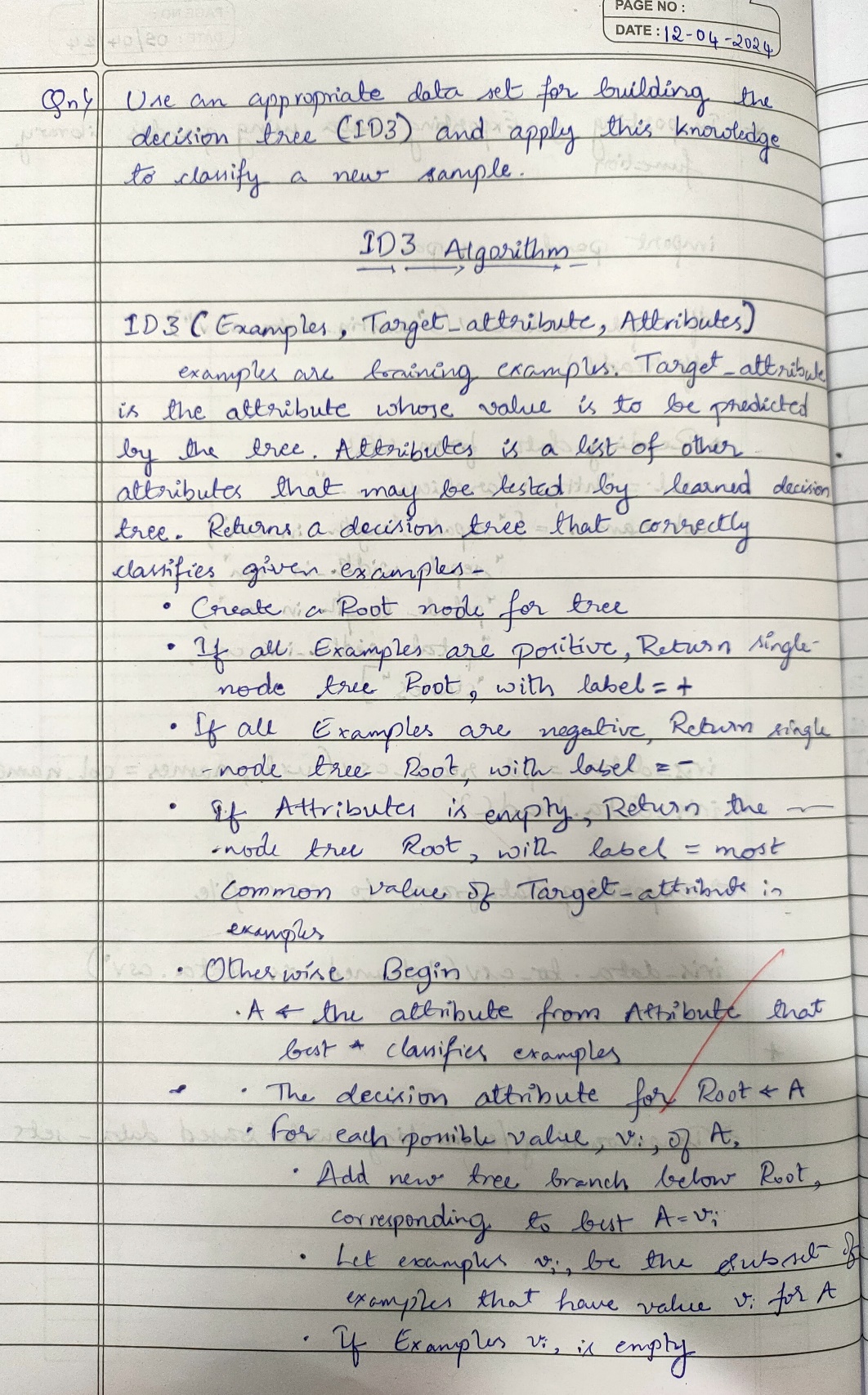


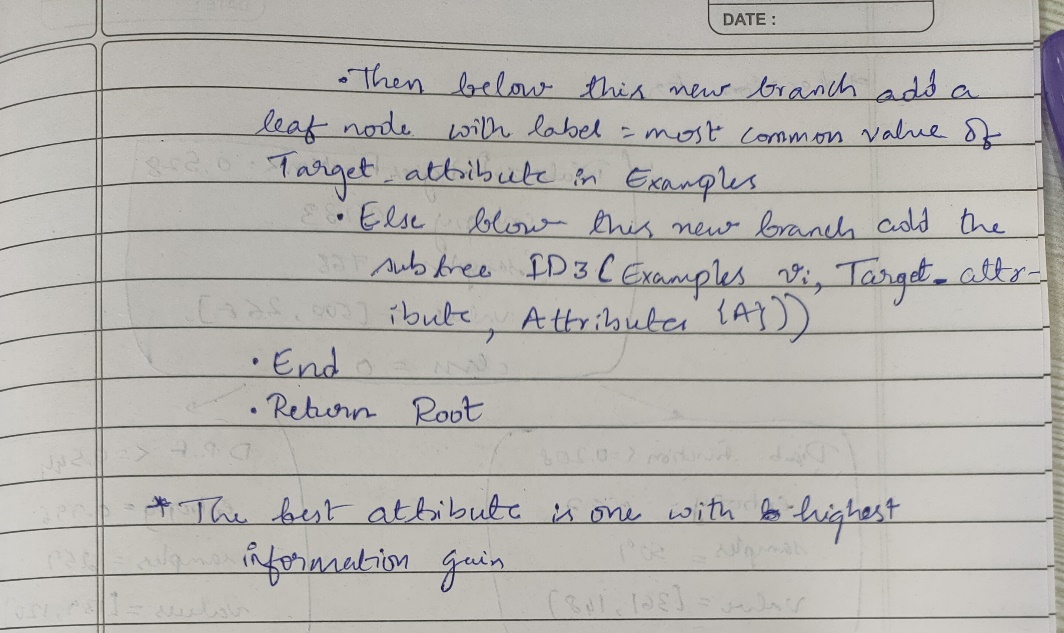
**Lab – 2**

**Date – 12-04-2024**

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

**Algorithm :**





**CODE :**

import pandas as pd

from sklearn.tree import DecisionTreeClassifier, plot\_tree

import matplotlib.pyplot as plt

import math

df = pd.read\_csv('/content/diabetes.csv')

df

def calculate\_entropy(data, target\_column):

total\_rows = len(data)

target\_values = data[target\_column].unique()

entropy = 0

for value in target\_values:

# Calculate the proportion of instances with the current value

value\_count = len(data[data[target\_column] == value])

proportion = value\_count / total\_rows

entropy -= proportion \* math.log2(proportion)

return entropy

entropy\_outcome = calculate\_entropy(df, 'Outcome')

print(f"Entropy of the dataset: {entropy\_outcome}")

def calculate\_entropy(data, target\_column): # for each categorical variable

total\_rows = len(data)

target\_values = data[target\_column].unique()

entropy = 0

for value in target\_values:

# Calculate the proportion of instances with the current value

value\_count = len(data[data[target\_column] == value])

proportion = value\_count / total\_rows

entropy -= proportion \* math.log2(proportion) if proportion != 0 else 0

return entropy

def calculate\_information\_gain(data, feature, target\_column):

# Calculate weighted average entropy for the feature

unique\_values = data[feature].unique()

weighted\_entropy = 0

for value in unique\_values:

subset = data[data[feature] == value]

proportion = len(subset) / len(data)

weighted\_entropy += proportion \* calculate\_entropy(subset, target\_column)

# Calculate information gain

information\_gain = entropy\_outcome - weighted\_entropy

return information\_gain

for column in df.columns[:-1]:

entropy = calculate\_entropy(df, column)

information\_gain = calculate\_information\_gain(df, column, 'Outcome')

print(f"{column} - Entropy: {entropy:.3f}, Information Gain: {information\_gain:.3f}")

# Feature selection for the first step in making decision tree

selected\_feature = 'DiabetesPedigreeFunction'

# Create a decision tree

clf = DecisionTreeClassifier(criterion='entropy', max\_depth=1)

X = df[[selected\_feature]]

y = df['Outcome']

clf.fit(X, y)

plt.figure(figsize=(8, 6))

plot\_tree(clf, feature\_names=[selected\_feature], class\_names=['0', '1'], filled=True, rounded=True)

plt.show()

def id3(data, target\_column, features):

if len(data[target\_column].unique()) == 1:

return data[target\_column].iloc[0]

if len(features) == 0:

return data[target\_column].mode().iloc[0]

best\_feature = max(features, key=lambda x: calculate\_information\_gain(data, x, target\_column))

tree = {best\_feature: {}}

features = [f for f in features if f != best\_feature]

for value in data[best\_feature].unique():

subset = data[data[best\_feature] == value]

tree[best\_feature][value] = id3(subset, target\_column, features)

return tree

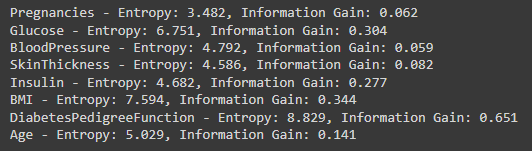
id3(df, 'Outcome', ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age'] )

**OUTPUT :**

1. Entropy of Dataset :



1. Entropy and Information Gain of each feature



1. Decision Tree :

