1. Demonstrate the following data preprocessing tasks using python libraries.
2. Loading the dataset

Program:

from sklearn import datasets

from sklearn.datasets import fetch\_california\_housing

df=fetch\_california\_housing()

x=df.data

print(x)

Output:  


b) Identifying the dependent and independent variables.

Programm:

from sklearn.datasets import load\_iris

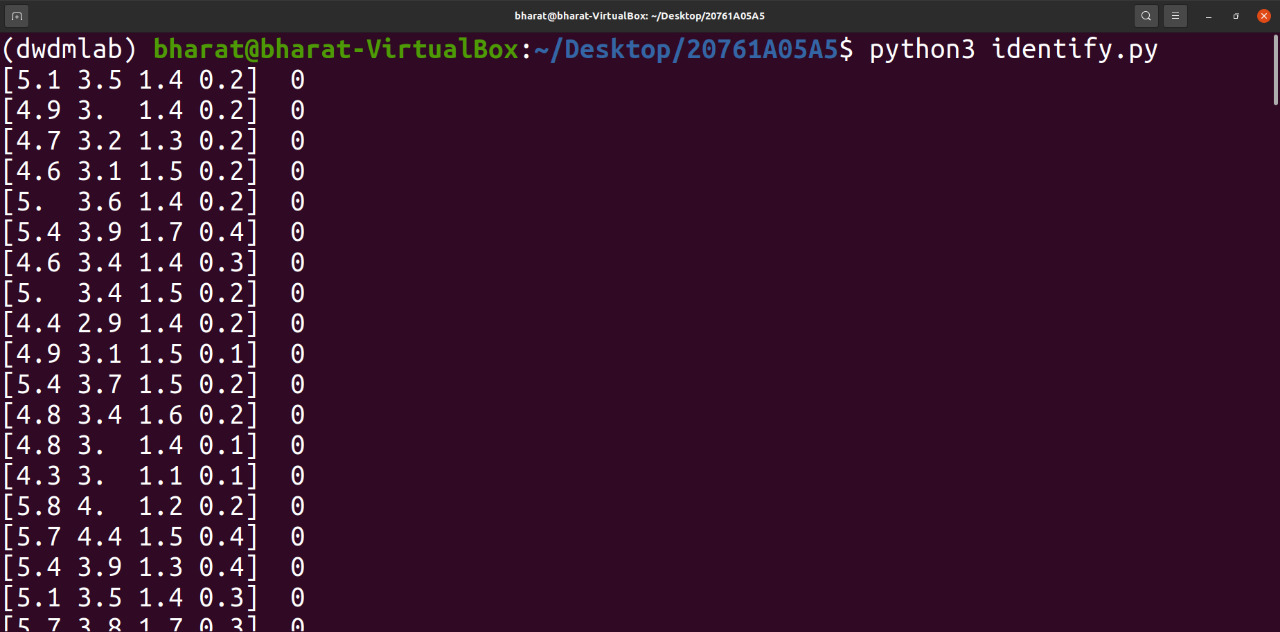
i=load\_iris()

X,Y=i.data,i.target

for i in range(0,len(X)):

print(X[i],"",Y[i])

Output:



c) Dealing with missing data

Programm:

import numpy as np

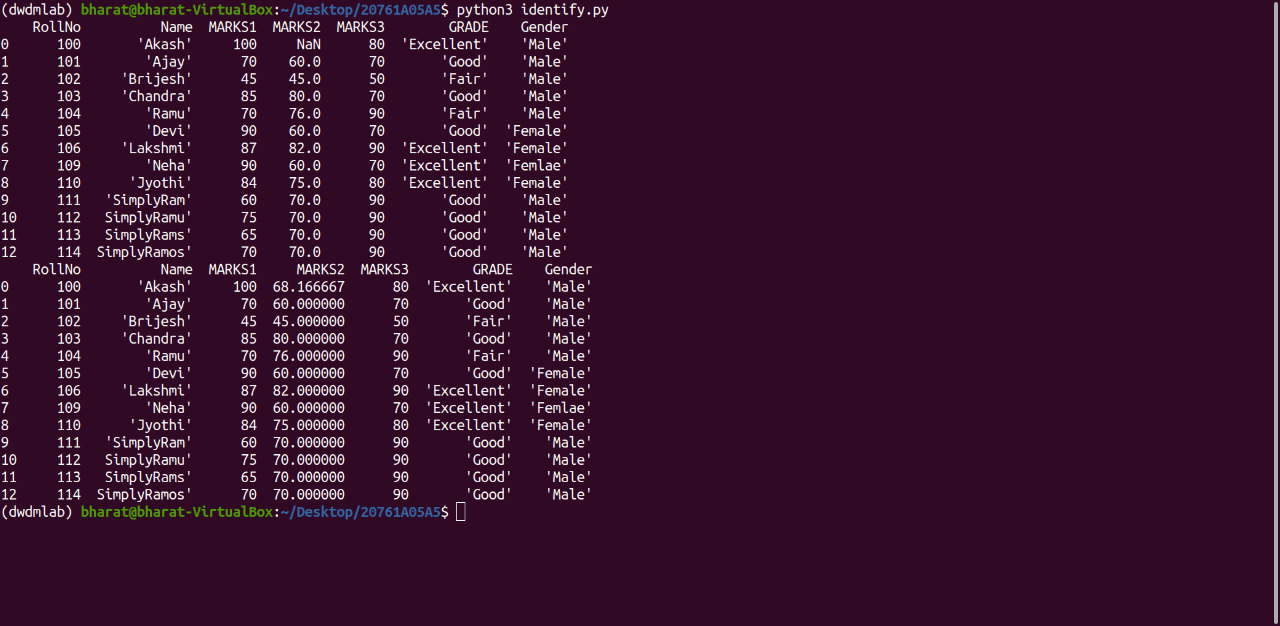
import pandas as pd

df=pd.read\_csv('stu.csv')

print(df)

df['MARKS2']=df['MARKS2'].fillna(df['MARKS2'].mean())

print(df)



2. Demonstrate the following data preprocessing tasks using python libraries.

a) Dealing with categorical data

Programm: using LabelEncoder

import pandas as pd

import numpy as np

from sklearn.preprocessing import LabelEncoder

df=pd.read\_csv('stu.csv')

print(df['Name'])

a=LabelEncoder()

df['Name']=a.fit\_transform(df['Name'])

print(df['Name'])

Output:



Programm: Using LabelBinarizer

import pandas as pd

import numpy as np

from sklearn.preprocessing import LabelBinarizer

df=pd.read\_csv('stu.csv')

print(df['Name'])

a=LabelBinarizer()

df=a.fit\_transform(df['Name'])

print(df)

Output:



Programm: Using asType,cat.Codes

import numpy as np

import pandas as pd

df=pd.read\_csv('stu.csv')

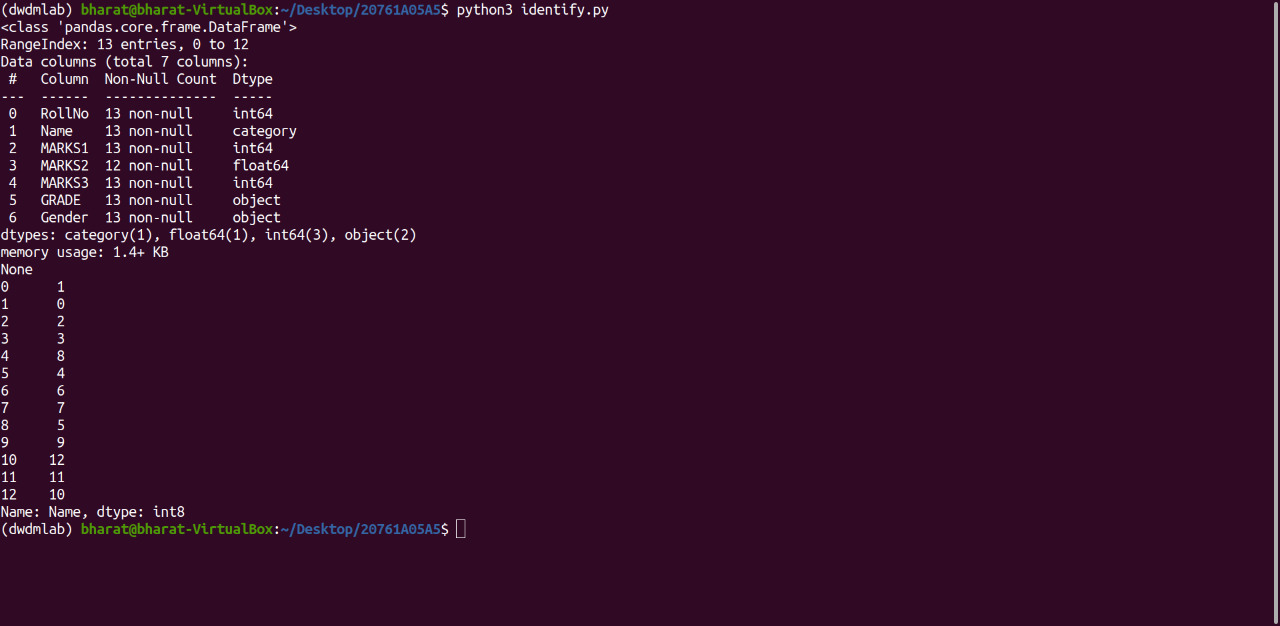
df['Name']=df['Name'].astype('category')

print(df.info())

df['Name']=df['Name'].cat.codes

print(df['Name'])

Output:



1. Scaling the features

import pandas as pd

from sklearn.preprocessing import MinMaxScaler,StandardScaler

df = pd.read\_csv('iris.csv')

x = df.iloc[:, 1:3].values

min\_max = MinMaxScaler(feature\_range =(0, 1))

min\_max1=StandardScaler()

x\_after\_min\_max = min\_max.fit\_transform(x)

x\_after\_min\_max1=min\_max1.fit\_transform(x)

print("Min Max Scaler output is\n", x\_after\_min\_max)

print("Standard Scaler output is\n",x\_after\_min\_max1)

Output:





c) Splitting dataset into Training and Testing Sets

Programm:

from sklearn.datasets import load\_iris

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

i=load\_iris()

X,Y=i.data,i.target

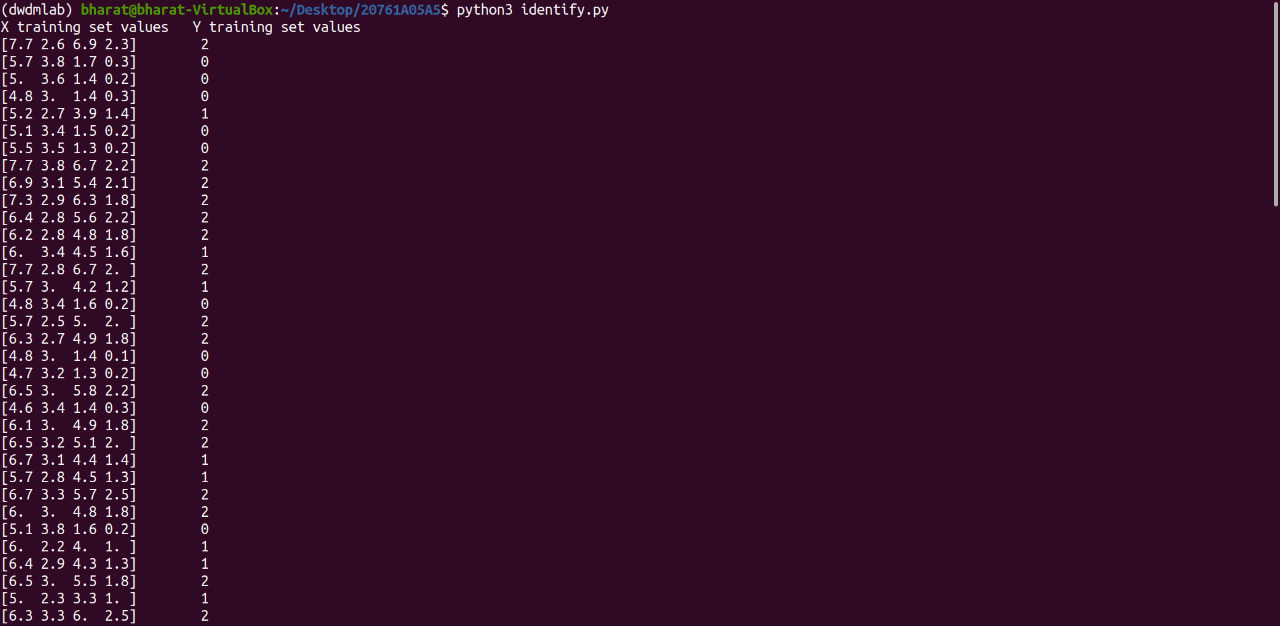
X\_train,X\_test,Y\_train,Y\_test=train\_test\_split(X,Y,test\_size=0.3,random\_state=1)

print("X training set values\tY training set values")

for i in range(0,len(X\_train)):

print(X\_train[i],'\t',Y\_train[i])

Output:



3. Demonstrate the following Similarity and Dissimilarity Measures using python

a) Pearson’s Correlation

Programm:

from scipy.stats import pearsonr

X=[-2,-1,0,1,2]

Y=[4,1,3,2,0]

corr=pearsonr(X,Y)

print("pearson correlation:",corr)

b)Cosine Similarity

Programm:

from sklearn.feature\_extraction.text import CountVectorizer

import pandas as pd

from sklearn.metrics.pairwise import cosine\_similarity

A="deep learning can be hard"

B="deep learning can be soft"

documents=[A,B]

ob=CountVectorizer()

X=ob.fit\_transform(documents)

Y=X.todense()

df=pd.DataFrame(Y,columns=ob.get\_feature\_names\_out(),index=['A','B'])

print(df)

print("similarity matrix:\n",cosine\_similarity(df,df))

1. Jaccard Similarity

Programm:

import numpy as np

from scipy.spatial.distance import jaccard

a=np.array([1,0,1,0,0,1])

b=np.array([0,1,0,1,0,1])

print(" jaccard distance:",jaccard(a,b))

1. Manhattan Distance

Programm:

import numpy as np

import pandas as pd

from sklearn.metrics.pairwise import manhattan\_distances

X=np.ones((1,2))

Y=np.full((2,2),2)

manhattan\_distances(X,Y,sum\_over\_features=False)

e) Euclidean Distance

Programm:

from sklearn.metrics.pairwise import euclidean\_distances

X=[[0,1],[1,1]]

print(euclidean\_distances(X,X))

print("get distance from origin")

print(euclidean\_distances(X,[[0,0]]))

Output:

