****

**CERTIFICATE**

**This is to certify that Mr. Borate Akash Rajendra student of M.Sc(C.S.) Semester III at Suryadatta College of Management Information Research & Technology (SCMIRT), Pune, has successfully completed the assigned practical journal in Machine Learning prescribed by the Savitribai Phule Pune University during the academic year 2022-2023.**

**Internal Examiner External Examiner**

**HOD Principal**

**Place: Pune**

**Date: /01/2023**

**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr.No.** | **Name** | **Page No.** | **Remark** | **Sign** |
| **1)** | **Write a python program to Prepare Scatter Plot (Use Forge Dataset / Iris Dataset)** | **3** |  |  |
| **2)** | **Write a python program to find all null values in a given data set and remove them.** | **4 - 6** |  |  |
| **3)** | **Write a python program the Categorical values in numeric format for a given dataset.** | **7** |  |  |
| **4)** | **Write a python program to implement simple Linear Regression for predicting house price.** | **8 - 9** |  |  |
| **5)** | **Write a python program to implement multiple Linear Regression for a given dataset.** | **10 - 12** |  |  |
| **6)** | **Write a python program to implement Polynomial Regression for given dataset.** | **13 - 15** |  |  |
| **7)** | **Write a python program to Implement Naïve Bayes.** | **16 - 17** |  |  |
| **8)** | **Write a python program to Implement Decision Tree whether or not to play tennis.** | **18 - 19** |  |  |
| **9)** | **Write a python program to implement linear SVM.** | **20 - 22** |  |  |
| **10)** | **Write a python program to implement k-nearest Neighbors ML algorithm to build prediction model (Use Forge Dataset)** | **23 - 24** |  |  |

1. **Write a python program to Prepare Scatter Plot (Use Forge Dataset / Iris Dataset)**

**Ans:**

import pandas as pd

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

print(df)

import matplotlib.pyplot as plt

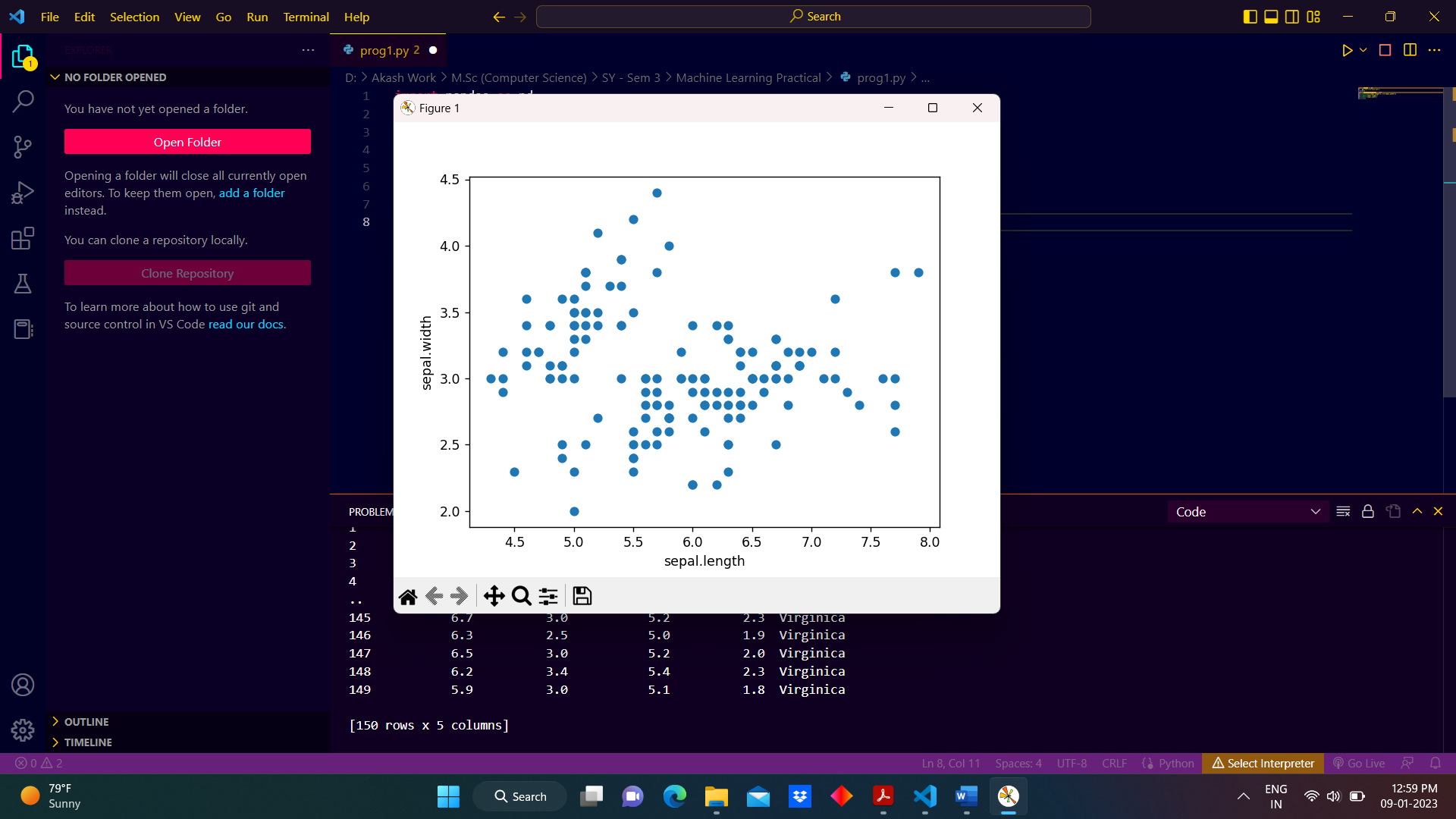
plt.scatter(df['sepal.length'], df['sepal.width'])

plt.xlabel('sepal.length')

plt.ylabel('sepal.width')

plt.show()

**Output:**



1. **Write a python program to find all null values in a given data set and remove them.**

**Ans:**

import pandas as pd

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

print(df)

df.isnull().sum()

df[df['sepal.length'].isnull()]

df[df['sepal.width'].isnull()]

df.fillna('default')

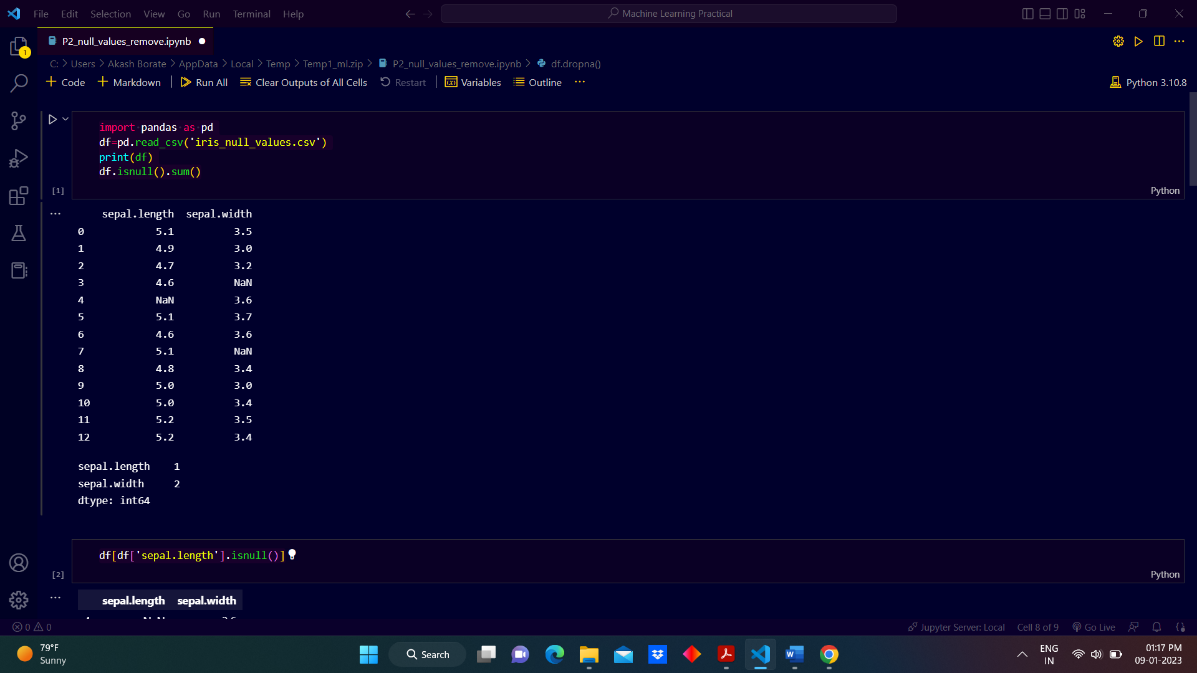
df.isnull()

df.notnull()

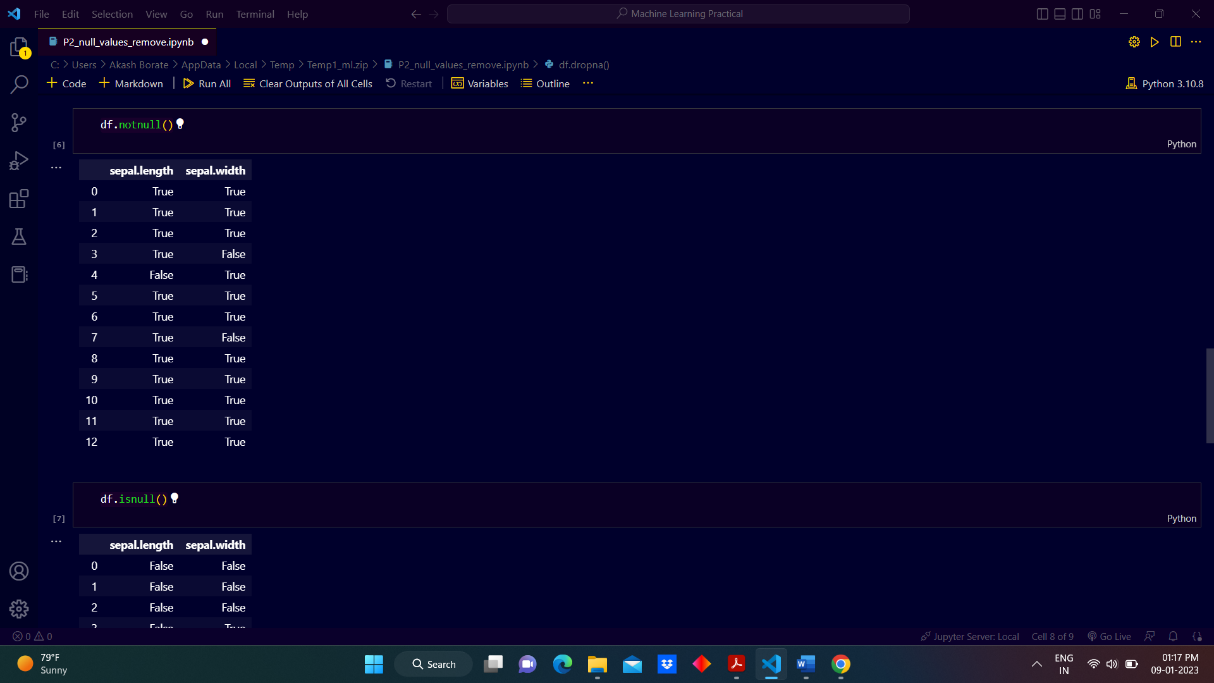
df.isnull()

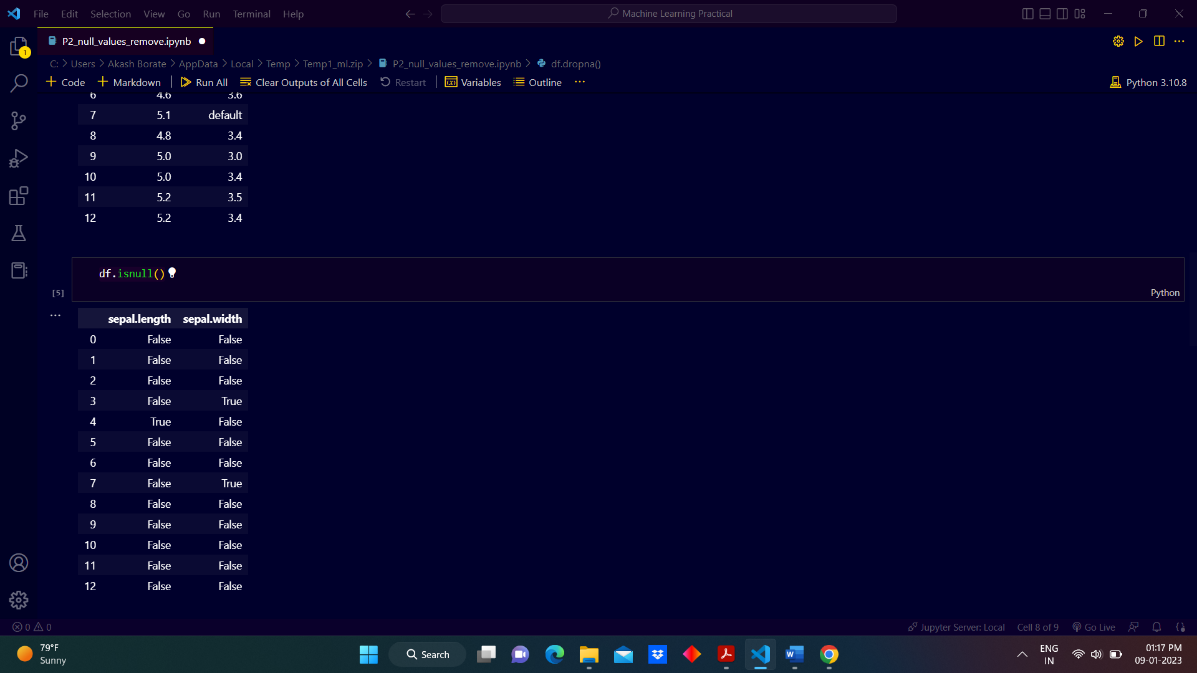
df.dropna()

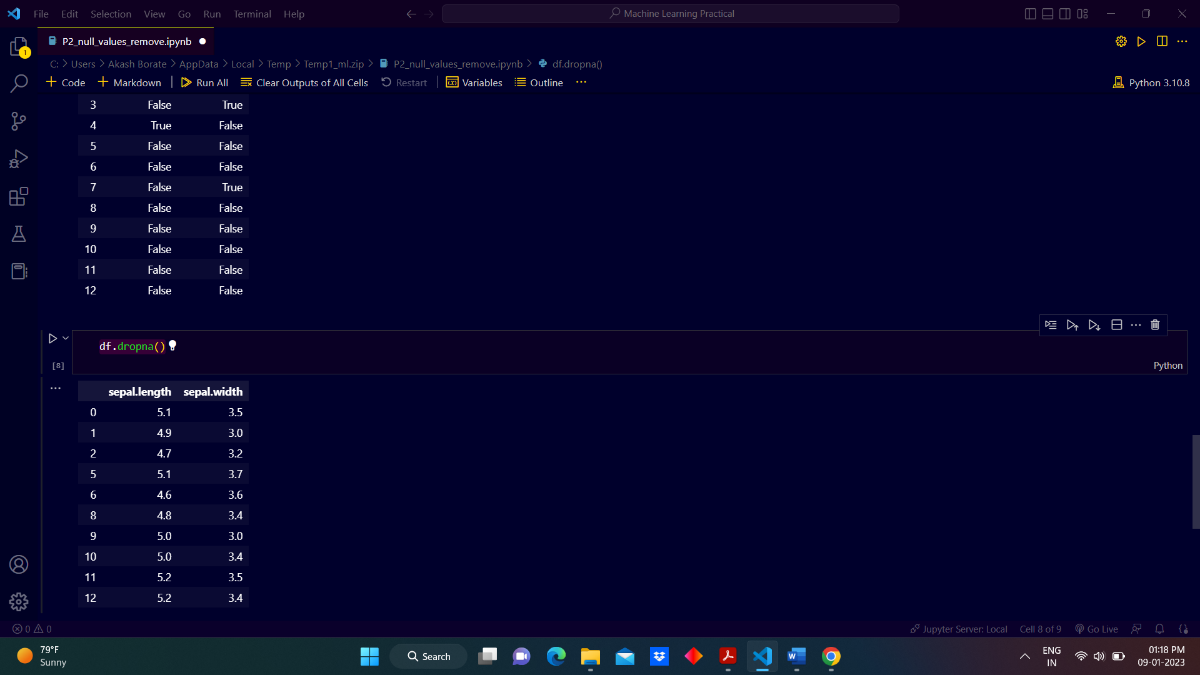
**Output:**











1. **Write a python program the Categorical values in numeric format for a given dataset.**

**Ans:**

import pandas as pd

df=pd.read\_csv('categorical\_data.csv')

print(df)

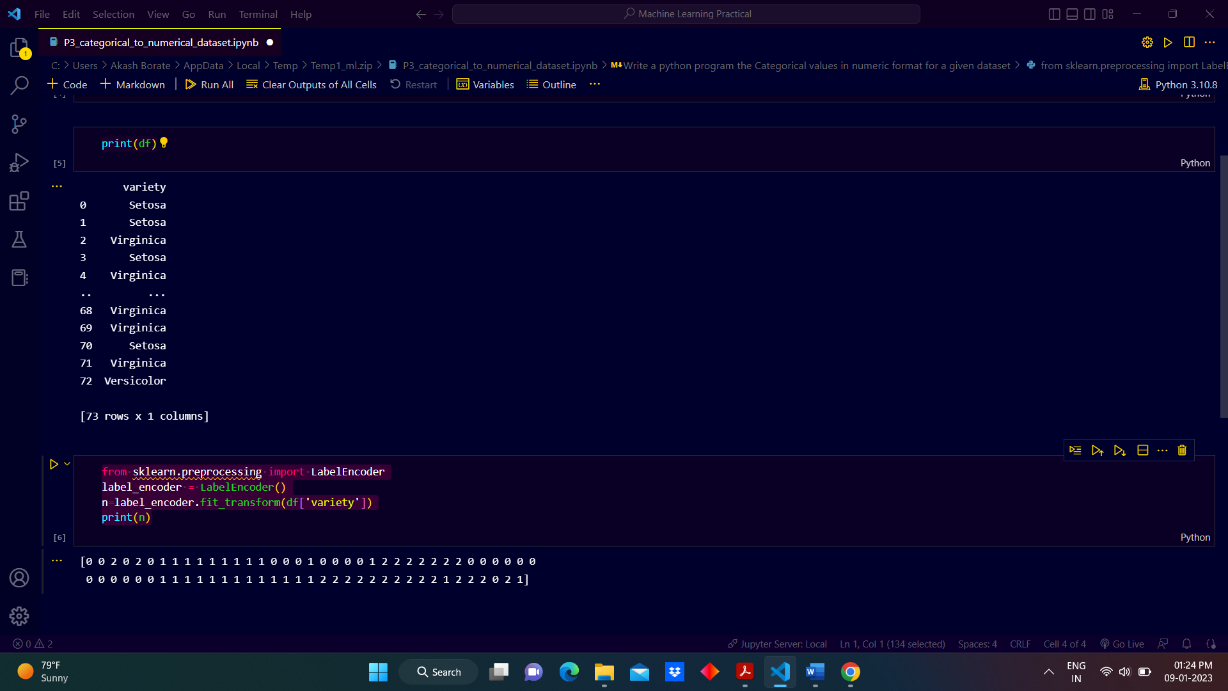
from sklearn.preprocessing import LabelEncoder

label\_encoder = LabelEncoder()

n=label\_encoder.fit\_transform(df['variety'])

print(n)

**output:**



1. **Write a python program to implement simple Linear Regression for predicting house price.**

**Ans:**

import pandas as pd

df=pd.read\_csv('housing\_data.csv')

print(df.head(10))

X = df['area (in cm)'].values

Y = df['price in lakhs'].values

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

n=len(X)

X = X.reshape((n, 1))

reg = LinearRegression()

reg = reg.fit(df[['area (in cm)']], Y)

Y\_pred = reg.predict(X)

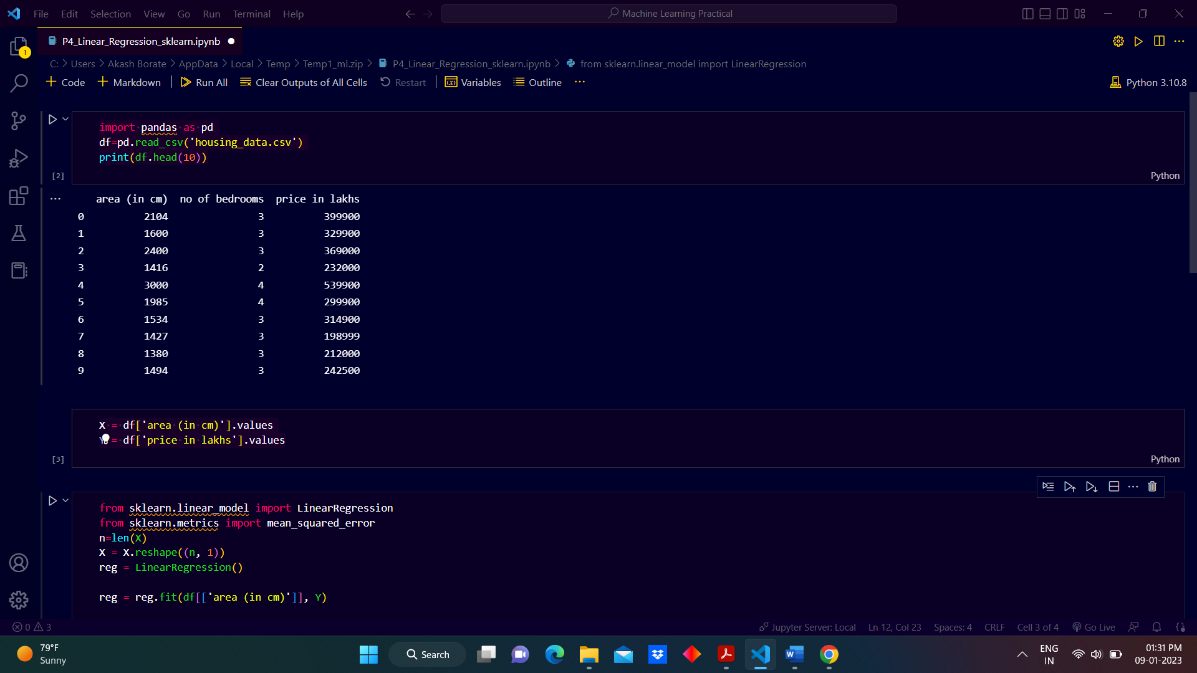
print(Y\_pred)

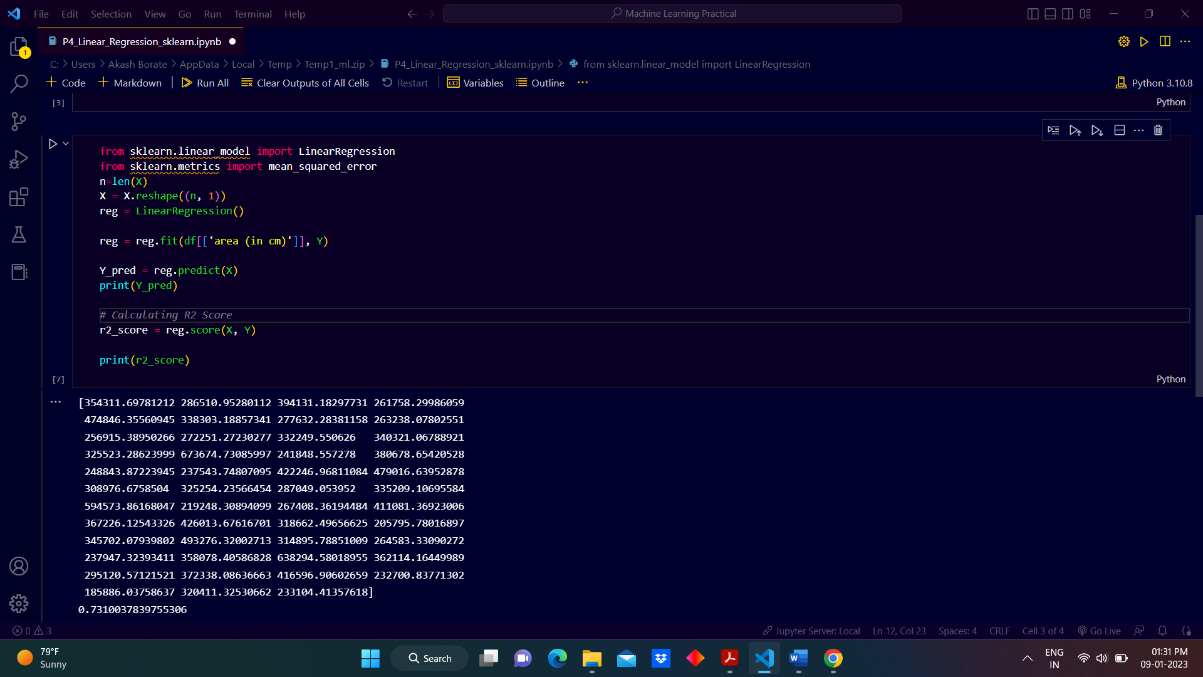
# Calculating R2 Score

r2\_score = reg.score(X, Y)

print(r2\_score)

**Output:**





1. **Write a python program to implement multiple Linear Regression for a given dataset.**

**Ans:**

import pandas as pd

df=pd.read\_csv('housing\_data.csv')

print(df.head(10))

import numpy as np

X = df[['area (in cm)','no of bedrooms']].values

Y = df['price in lakhs'].values

print(X,Y)

from sklearn import linear\_model

regr = linear\_model.LinearRegression()

regr.fit(X, Y)

predicted\_y = regr.predict([[1300, 4]])

print(predicted\_y)

print(regr.coef\_)

import pandas as pd

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

print(df)

from sklearn import linear\_model

X = df[['IQ','Study hours']].values

Y = df['Test score'].values

regr = linear\_model.LinearRegression()

regr.fit(X, Y)

predicted\_y = regr.predict([[110, 40]])

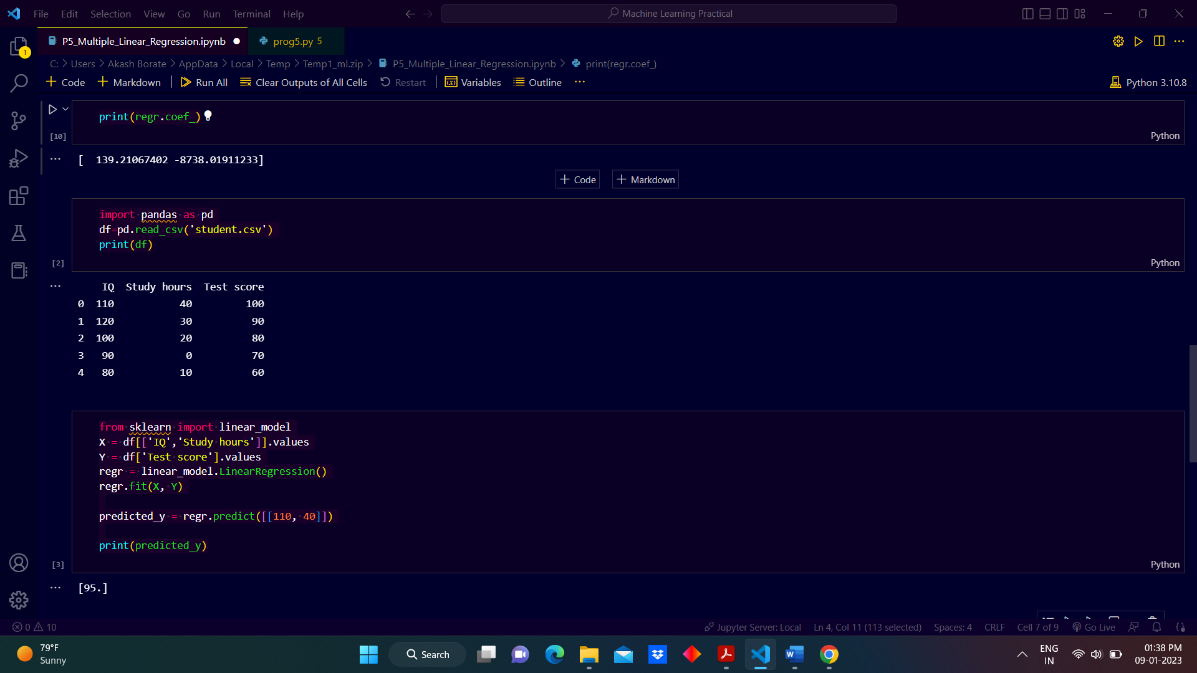
print(predicted\_y)

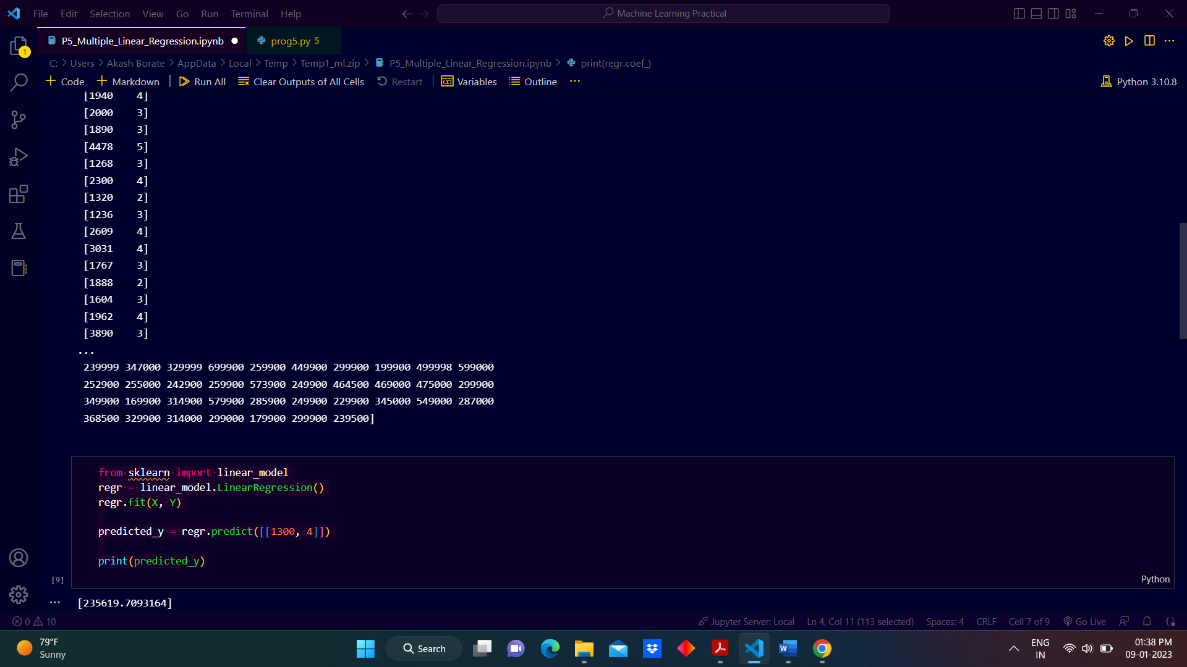
print(regr.coef\_)

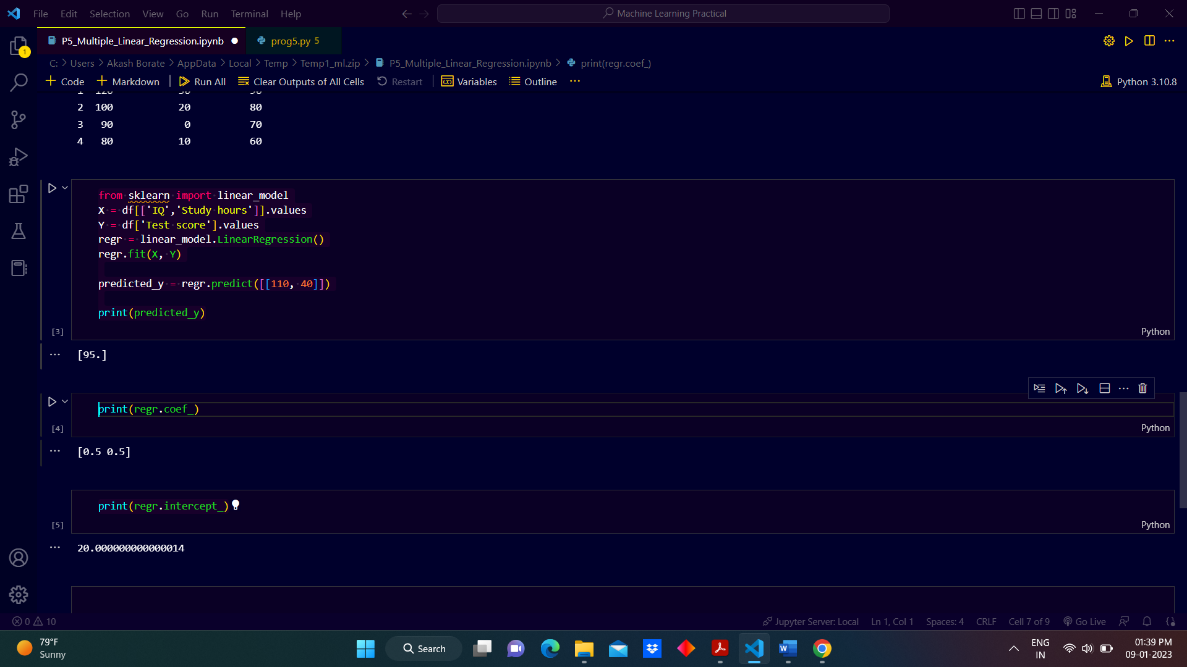
print(regr.intercept\_)

**Output:**









1. **Write a python program to implement Polynomial Regression for given dataset.**

**Ans:**

import pandas as pd

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

print(df)

X=df.iloc[:,1:2].values

y=df.iloc[:,2].values

print(X,y)

#fitting the polynomial regression model to the dataset

from sklearn.preprocessing import PolynomialFeatures

poly\_reg=PolynomialFeatures(degree=4)

X\_poly=poly\_reg.fit\_transform(X)

poly\_reg.fit(X\_poly,y)

lin\_reg2=LinearRegression()

lin\_reg2.fit(X\_poly,y)

#Visualising the pollynomial regression model results

import numpy as np

X\_grid=np.arange(min(X),max(X),0.1)

X\_grid=X\_grid.reshape((len(X\_grid),1))

plt.scatter(X,y,color='red')

plt.plot(X,lin\_reg2.predict(poly\_reg.fit\_transform(X)),color='blue')

plt.title('(Polynomial Regression)')

plt.xlabel('Position Level')

plt.ylabel('Salary')

plt.show()

from sklearn.linear\_model import LinearRegression

lin\_reg=LinearRegression()

lin\_reg.fit(X,y)

import matplotlib.pyplot as plt

plt.scatter(X,y,color='red')

plt.plot(X,lin\_reg.predict(X),color='blue')

plt.title('(Linear Regression)')

plt.xlabel('Position Level')

plt.ylabel('Salary')

plt.show()

import numpy as np

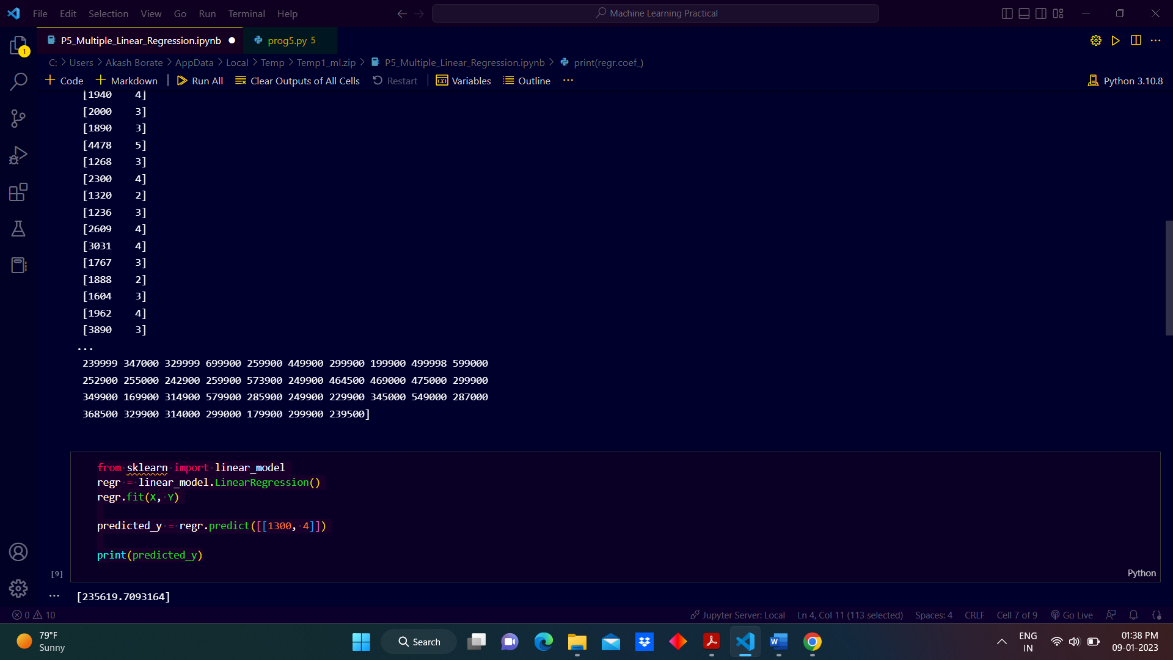
lin\_reg.predict(np.array([ [6.5] ]))

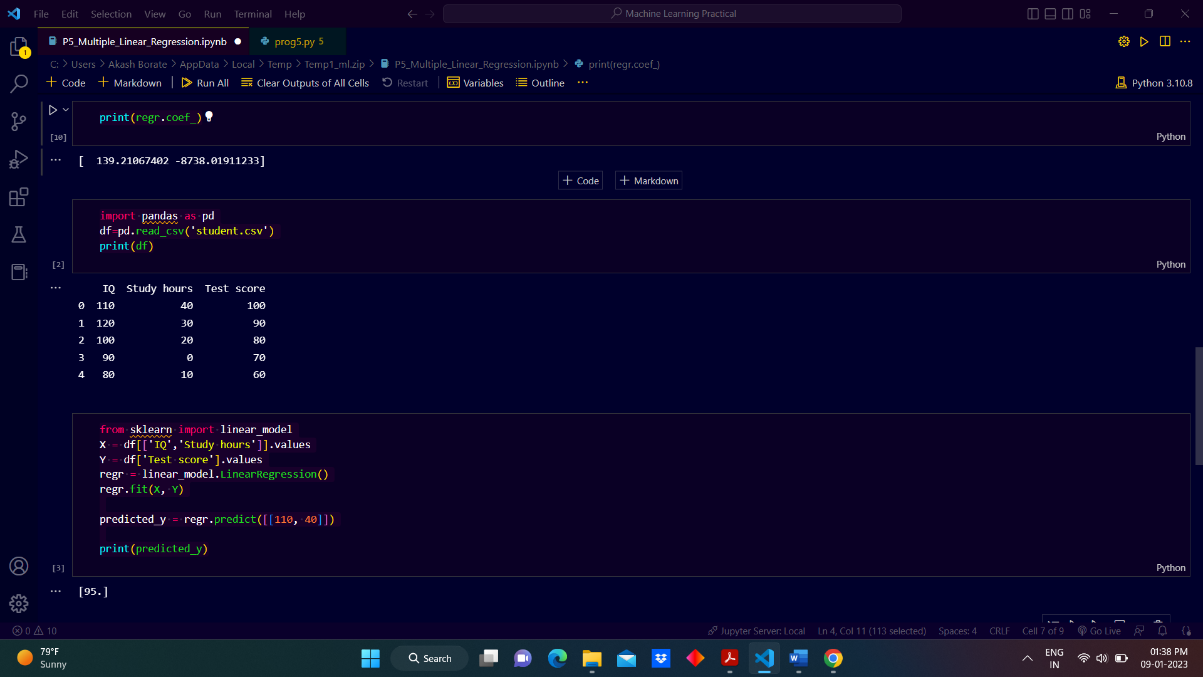
import numpy as np

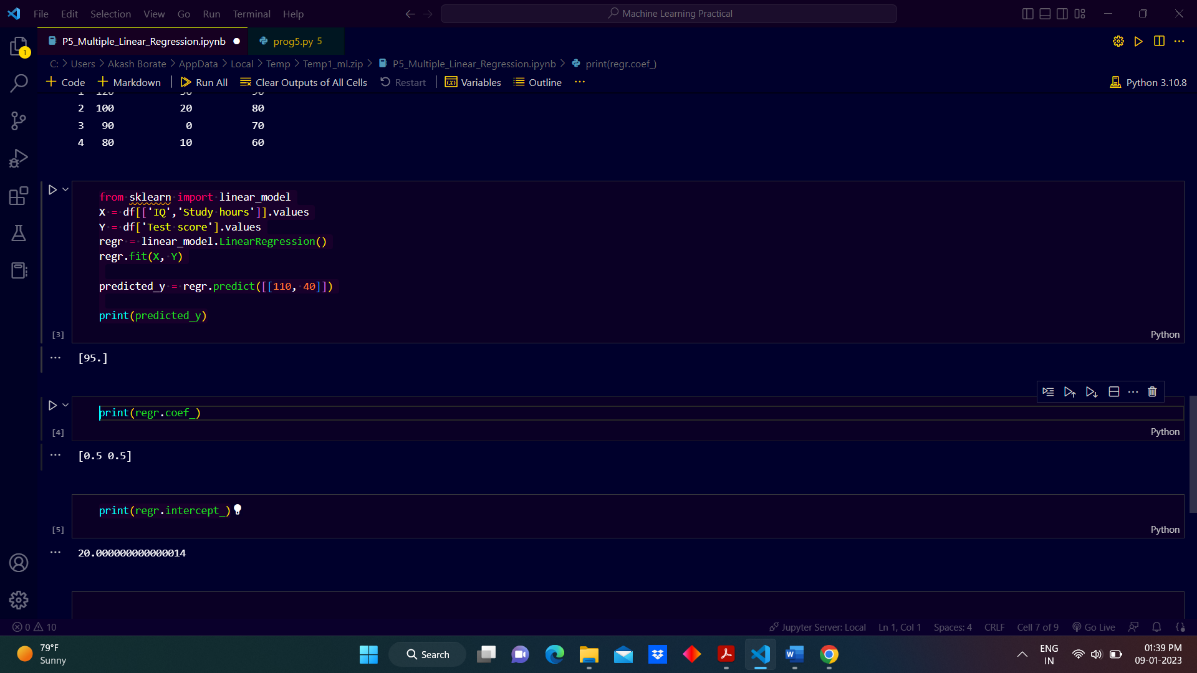
lin\_reg2.predict(poly\_reg.fit\_transform(np.array([ [6.5] ])))

**Output:**









1. **Write a python program to Implement Naïve Bayes.**

**Ans:**

import pandas as pd

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

print(df)

from sklearn import preprocessing

string\_to\_int= preprocessing.LabelEncoder() #transform categorical data into numerical form

df=df.apply(string\_to\_int.fit\_transform)

print(df)

from sklearn.naive\_bayes import GaussianNB

X = df[['outlook','temperature','humidity','windy']]

y= df['play']

#Create a Gaussian Classifier

model = GaussianNB()

# Train the model using the training sets

model.fit(X,y)

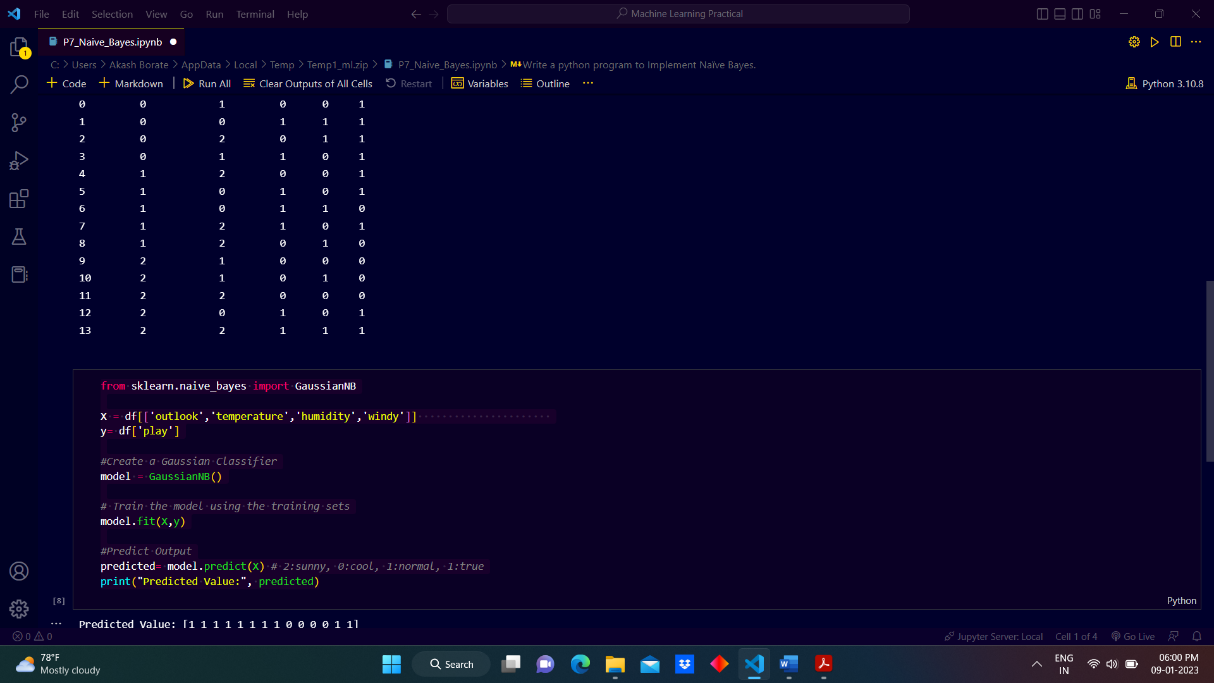
#Predict Output

predicted= model.predict(X) # 2:sunny, 0:cool, 1:normal, 1:true

print("Predicted Value:", predicted)

**Output:**





1. **Write a python program to Implement Decision Tree whether or not to play tennis.**

**Ans:**

**import** numpy as np

import pandas as pd

from sklearn import metrics

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

df

df.head()

from sklearn import preprocessing

string\_to\_int= preprocessing.LabelEncoder() #transform categorical data into numerical form

df=df.apply(string\_to\_int.fit\_transform)

df

X = df[['outlook','temperature','humidity','windy'] ]

y= df['play']

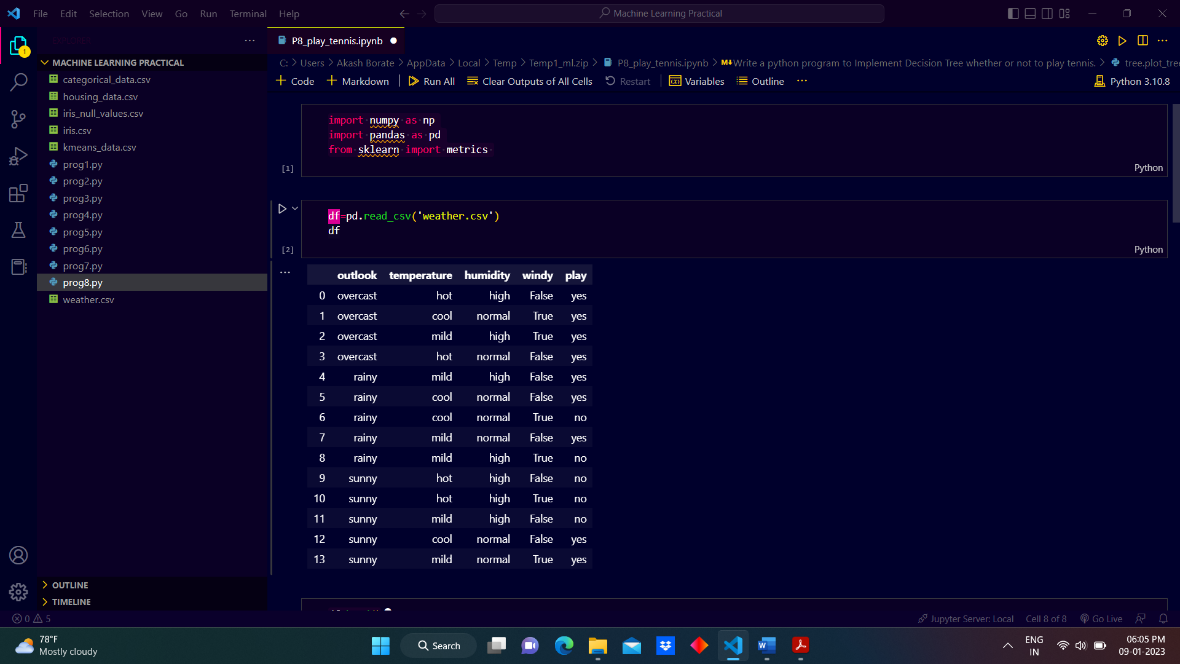
from sklearn import tree

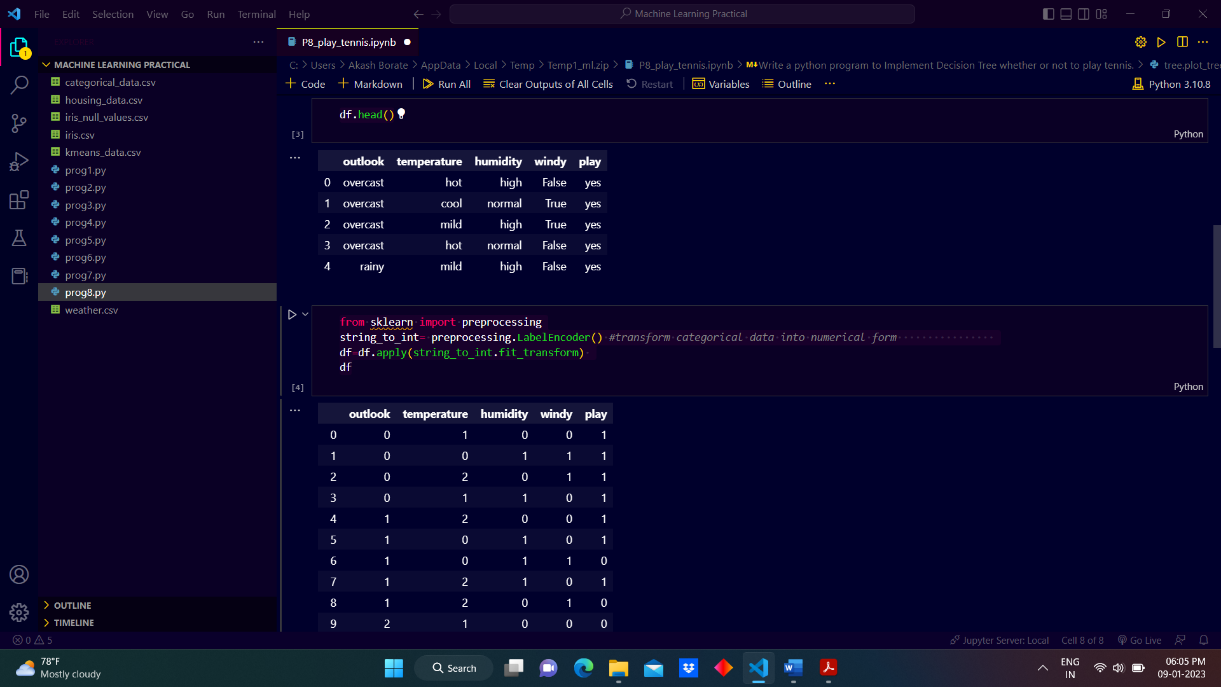
clf = tree.DecisionTreeClassifier(criterion = 'entropy')

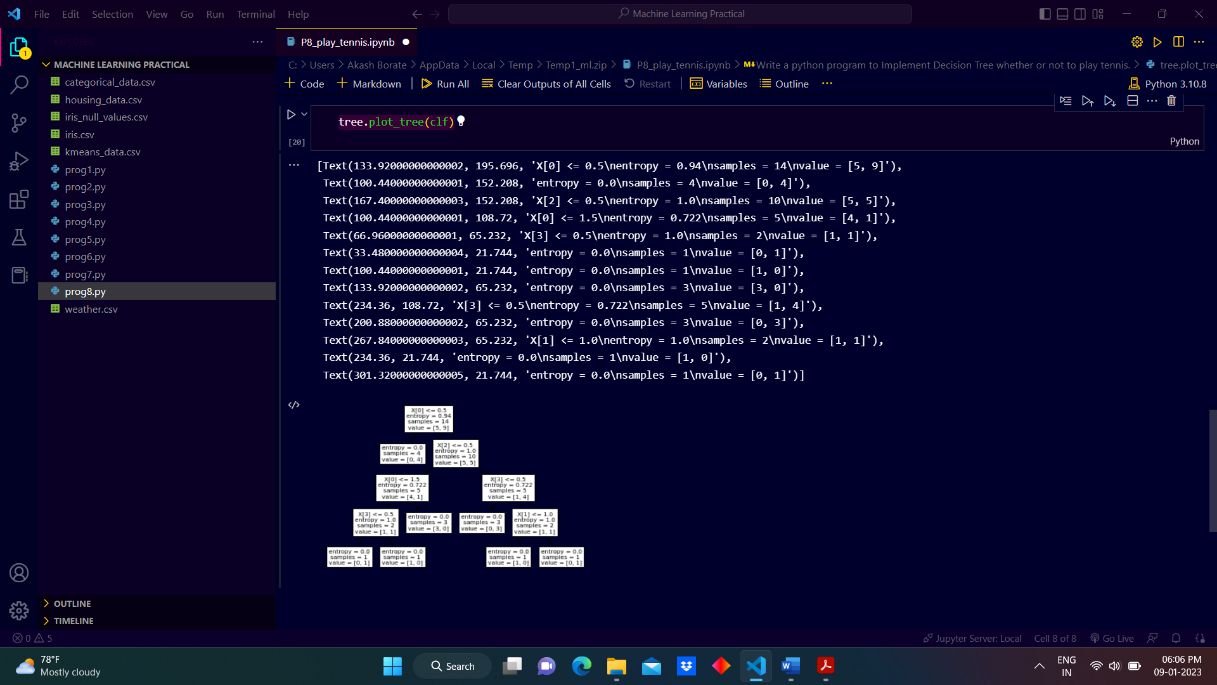
clf = clf.fit(X, y)

tree.plot\_tree(clf)

**Output:**







1. **Write a python program to implement linear SVM.**

**Ans:**

x = [1, 5, 1.5, 8, 1, 9]

y = [2, 8, 1.8, 8, 0.6, 11]

import matplotlib.pyplot as plt

from matplotlib import style

style.use("ggplot")

plt.scatter(x,y)

plt.show()

import numpy as np

from sklearn import svm

X = np.array([[1,2],

[5,8],

[1.5,1.8],

[8,8],

[1,0.6],

[9,11]])

y = [0,1,0,1,0,1]

clf = svm.SVC(kernel='linear', C = 1.0)

clf.fit(X,y)

print(clf.predict([[0.58,0.76]]))

w = clf.coef\_[0]

print(w)

a = -w[0] / w[1]

xx = np.linspace(0,12)

yy = a \* xx - clf.intercept\_[0] / w[1]

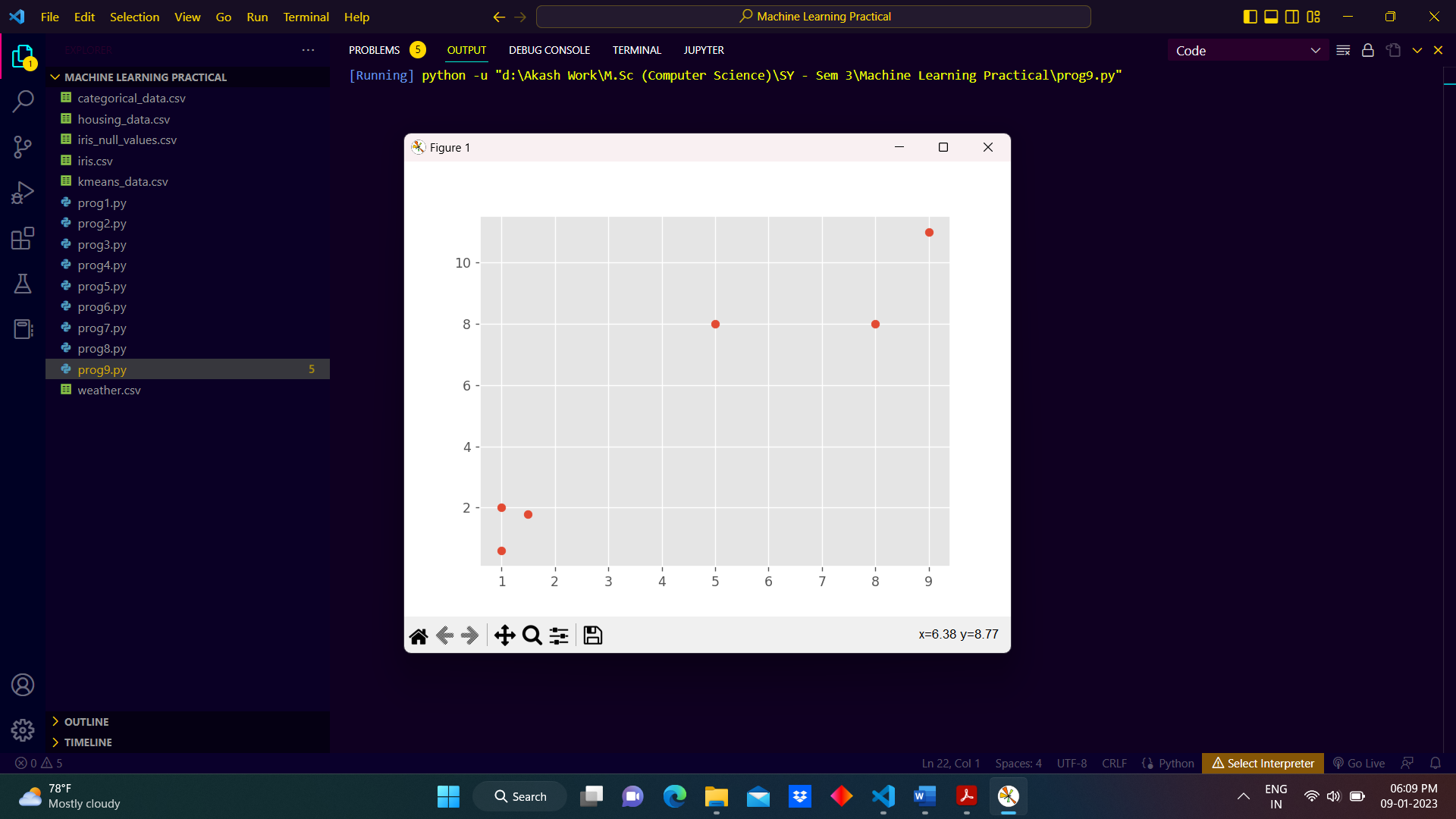
h0 = plt.plot(xx, yy, 'k-', label="non weighted div")

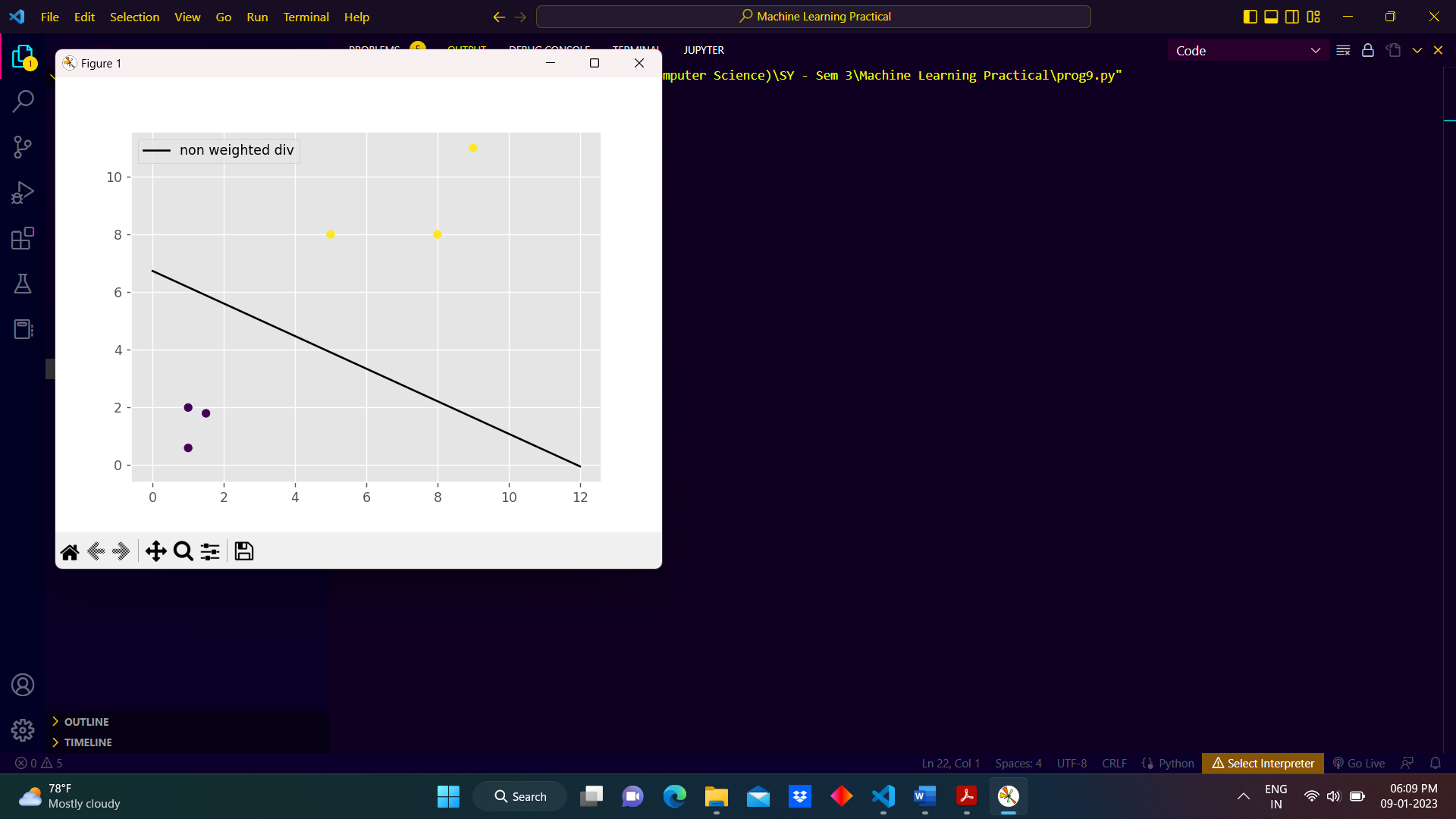
plt.scatter(X[:, 0], X[:, 1], c = y)

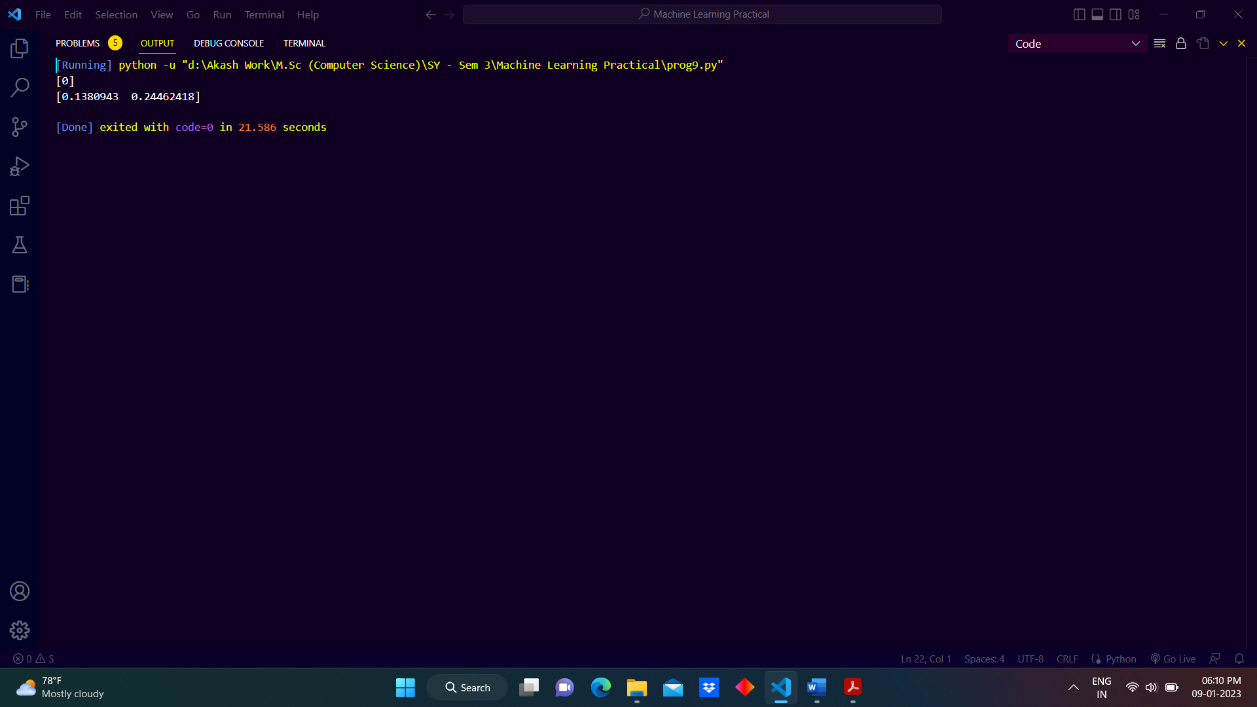
plt.legend()

plt.show()

**Output:**







**10) Write a python program to implement k-nearest Neighbors ML algorithm to build prediction model (Use Forge Dataset)**

**Ans:**

x = [4, 5, 10, 4, 3, 11, 14 , 8, 10, 12]

y = [21, 19, 24, 17, 16, 25, 24, 22, 21, 21]

classes = [0, 0, 1, 0, 0, 1, 1, 0, 1, 1]

data=list(zip(x,y))

data

import matplotlib.pyplot as plt

plt.scatter(x, y, c=classes)

plt.show()

from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n\_neighbors=3)

knn.fit(data, classes)

new\_x = 7

new\_y = 15

new\_point = [(new\_x, new\_y)]

prediction = knn.predict(new\_point)

new\_point

prediction

plt.scatter(x + [new\_x], y + [new\_y], c=classes + [prediction[0]])

plt.text(x=new\_x-1.7, y=new\_y-0.7, s=f"new point, class: {prediction[0]}")

plt.show()

**Output:**

