
# 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**

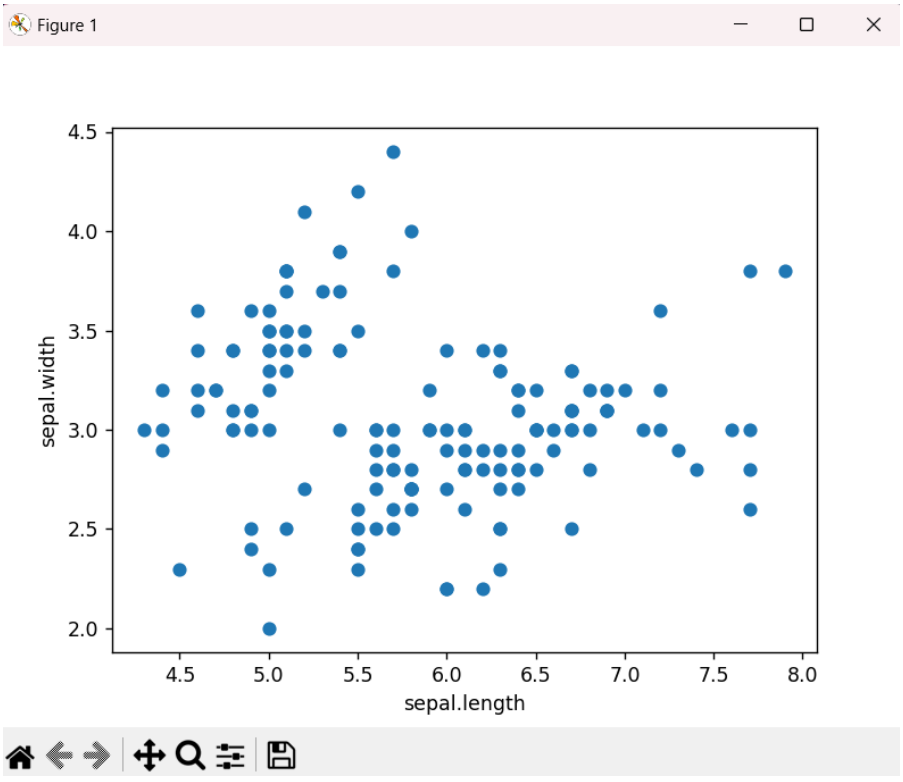
# INDEX

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:**

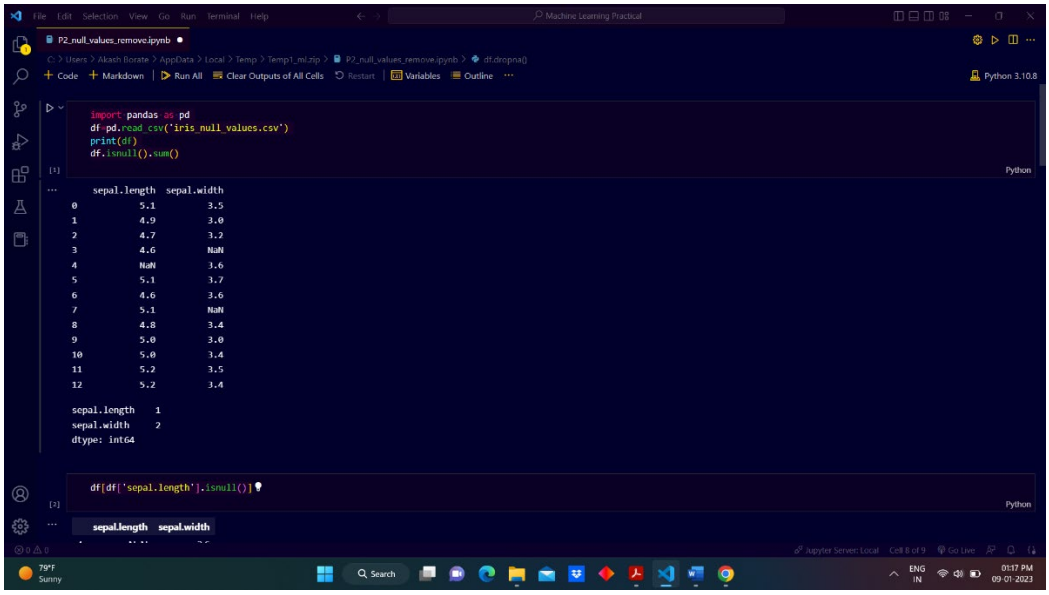


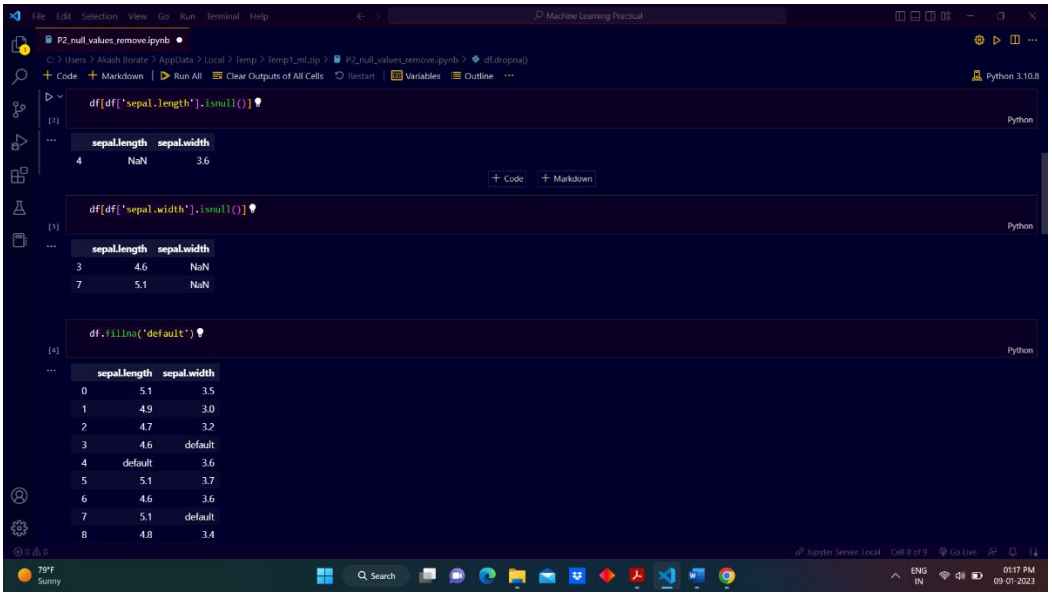
**2) Write a python program to find all null values in a given data set and remove them.**

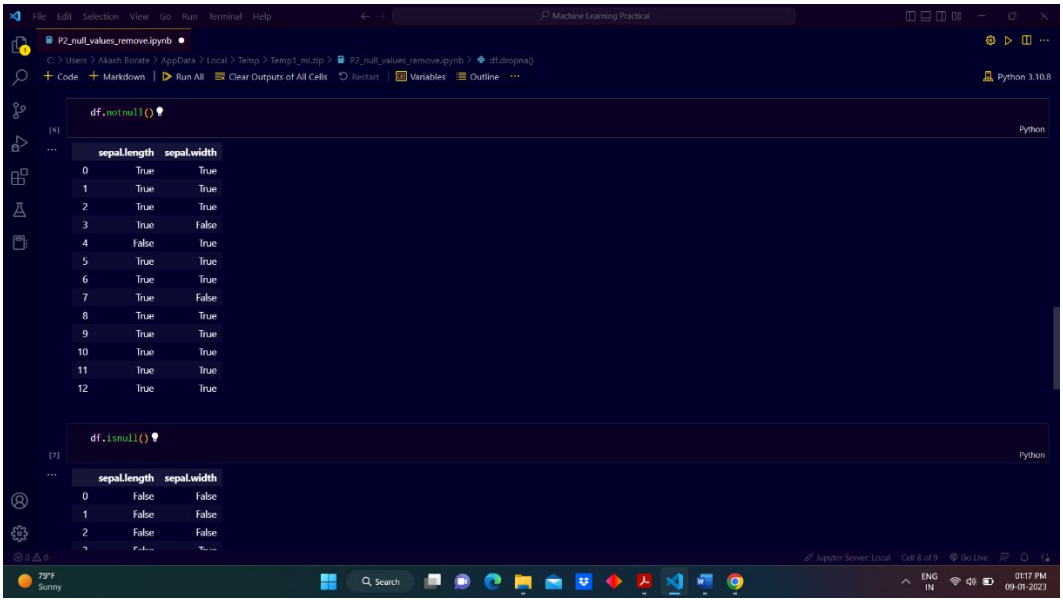
**Ans:**

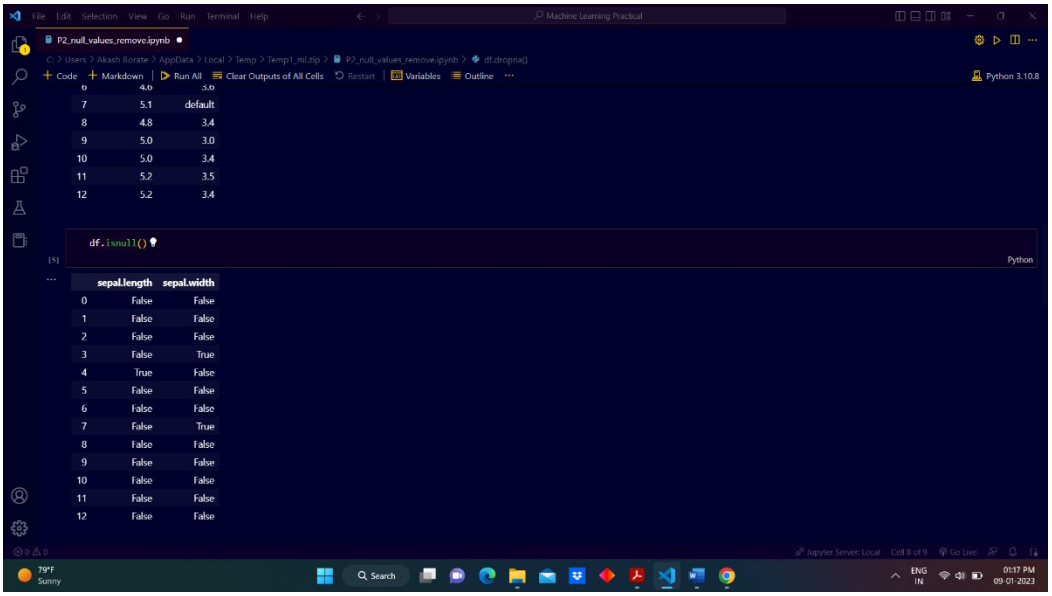
```
import pandas as pd
df=pd.read_csv('iris_null_values.csv')
print(df)
df.isnull().sum()
df[df['sepal.length'].isnull()]
df[df['sepal.width'].isnull()]
df.fillna('default')
df.isnull()
df.notnull()
df.isnull()
df.dropna()
```

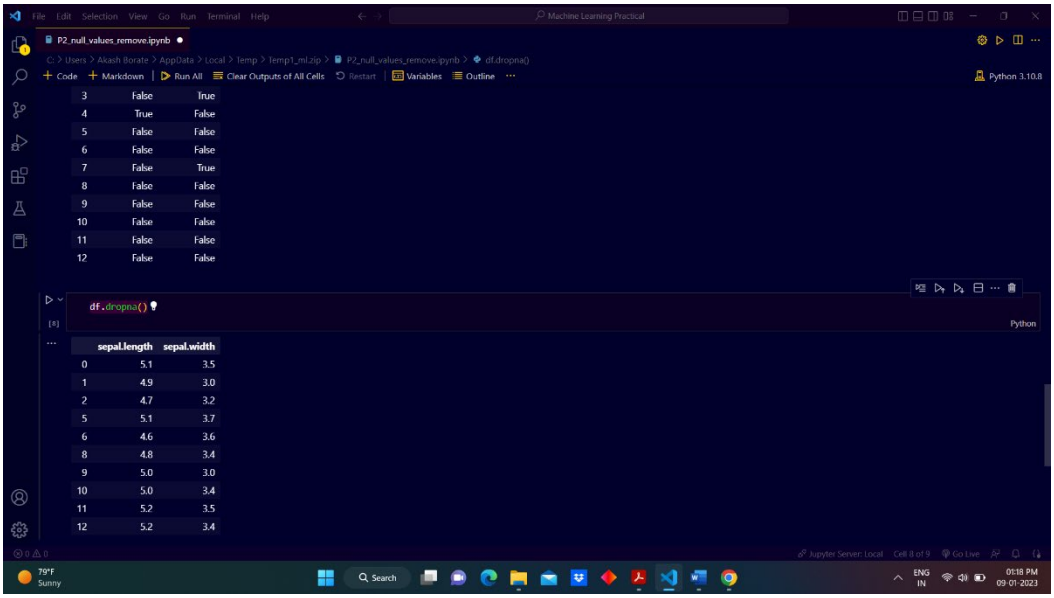
**Output:**




df[df['sepal.length'].isnull()]
sepal.length sepal.width
4 NaN 3.6
df[df['sepal.width'].isnull()]
sepal.length sepal.width
3 4.6 NaN
7 5.1 NaN
df.fillna('default')
sepal.length sepal.width
0 5.1 3.5
1 4.9 3.0
2 4.7 3.2
3 4.6 default
4 default 3.6
5 5.1 3.7
6 4.6 3.6
7 5.1 default
8 4.8 3.4


df.notnull()
sepal.length sepal.width
0 True True
1 True True
2 True True
3 True False
4 False True
5 True True
6 True True
7 True False
8 True True
9 True True
10 True True
11 True True
12 True True
df.isnull()
sepal.length sepal.width
0 False False
1 False False
2 False False


P2_null_values_remove.ipynb
Machine Learning Practical
+ Code + Markdown Run All Clear Outputs of All Cells Restart Variables Outline
Python 3.10.8
7 5.1 default
8 4.8 3.4
9 5.0 3.0
10 5.0 3.4
11 5.2 3.5
12 5.2 3.4
df.isnull()
Python
sepal.length sepal.width
0 False False
1 False False
2 False False
3 False True
4 True False
5 False False
6 False False
7 False True
8 False False
9 False False
10 False False
11 False False
12 False False
Jupyter Server: Local
Cell 8 of 9
Go Live
79°F
Sunny
Search
ENG
IN
01:17 PM
09-01-2023

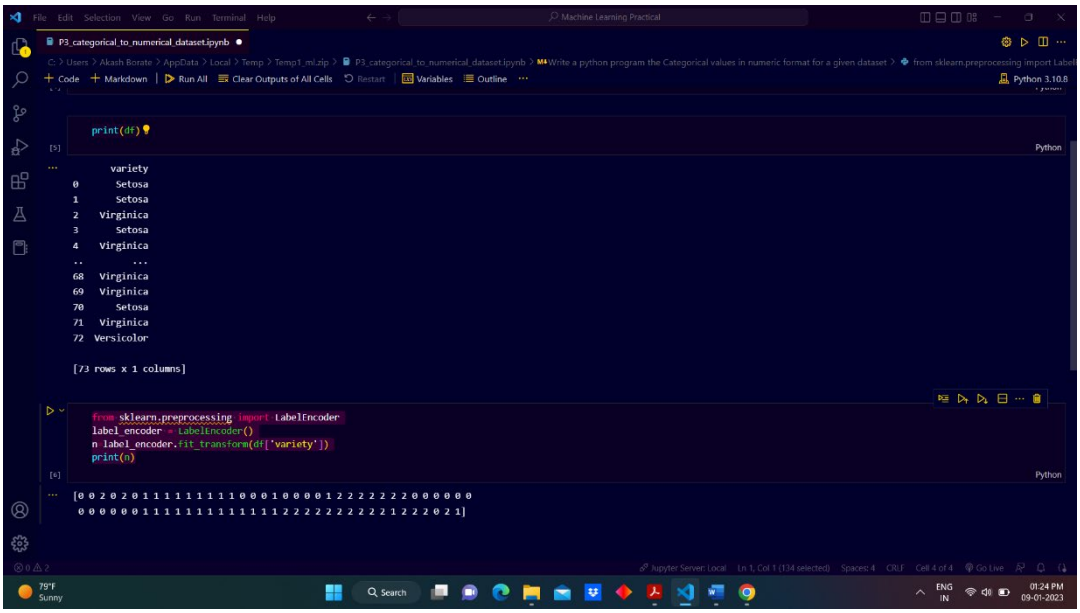

P2_null_values_remove.ipynb
Machine Learning Practical
+ Code + Markdown Run All Clear Outputs of All Cells Restart Variables Outline
Python 3.10.8
3 False True
4 True False
5 False False
6 False False
7 False True
8 False False
9 False False
10 False False
11 False False
12 False False
df.dropna()
Python
sepal.length sepal.width
0 5.1 3.5
1 4.9 3.0
2 4.7 3.2
5 5.1 3.7
6 4.6 3.6
8 4.8 3.4
9 5.0 3.0
10 5.0 3.4
11 5.2 3.5
12 5.2 3.4
Jupyter Server: Local
Cell 8 of 9
Go Live
79°F
Sunny
Search
ENG
IN
01:18 PM
09-01-2023

**3) 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:**

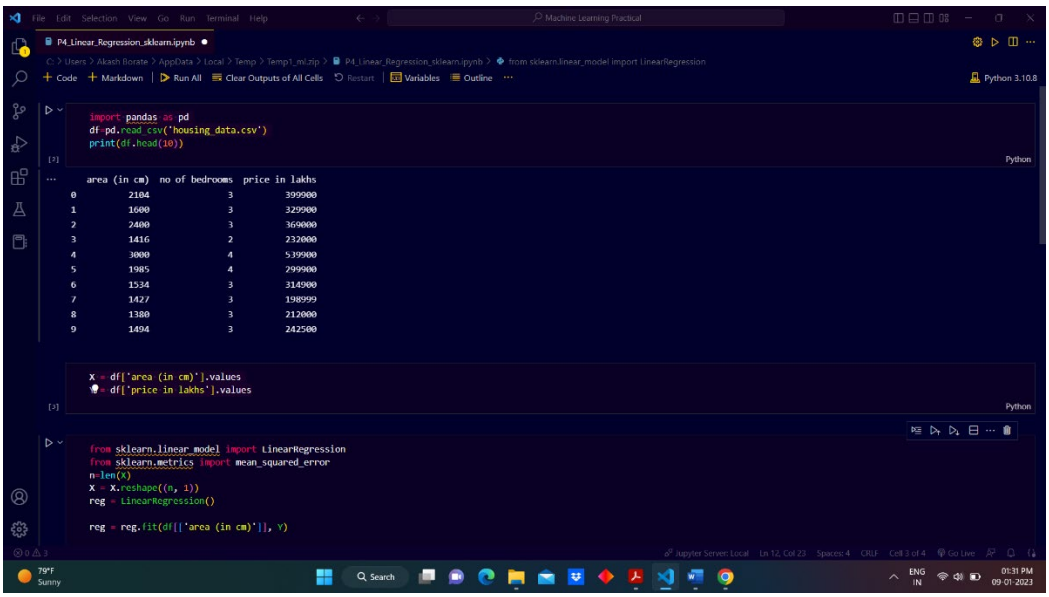


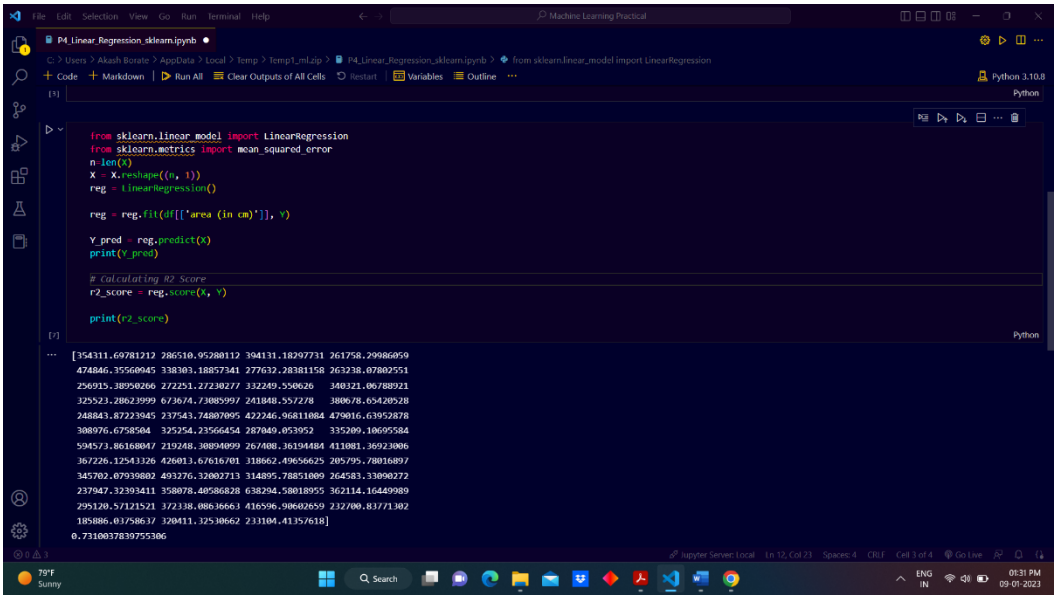
**4) Write a python program to implement simple Linear Regression for predicting house price.**

**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:**



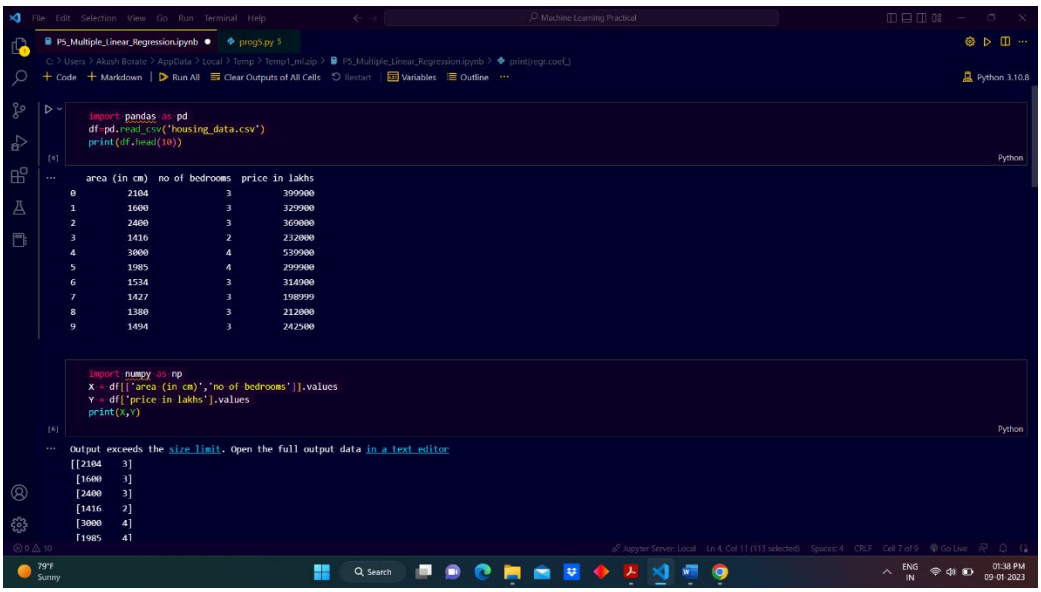

P4_Linear_Regression_sklearn.ipynb
Machine Learning Practical
Code
Markdown
Run All
Clear Outputs of All Cells
Restart
Variables
Outline
Python 3.10.8
Python
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
n=len(X)
X = X.reshape((n, 1))
reg = LinearRegression()
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)
[354311.69781212 286510.95280112 394131.18297731 261758.29986059
474846.35560045 338303.18857341 277632.28381158 263238.07802551
256915.38950266 272251.27230277 332249.550626 340321.06788921
325523.28623999 673674.73085997 241848.557278 380678.65420528
248843.87223945 237543.74807095 422246.96811084 479016.63952878
308976.6758504 325254.23566454 287049.053952 335209.10695584
594573.86168047 219248.30894099 267408.36194484 411081.36923006
367226.12543326 426013.67616701 318662.49656625 205795.78016897
345702.07939802 493276.32002713 314895.78851009 264583.33090272
237947.32393411 358078.40586828 638294.58018955 362114.16449989
295120.57121521 372338.08636663 416596.90602659 232700.83771302
185886.01758637 320411.32530662 233104.41357618]
0.7310037839755306
Search
79°F
Sunny
ENG
IN
01:31 PM
09-01-2023

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

**Ans:**

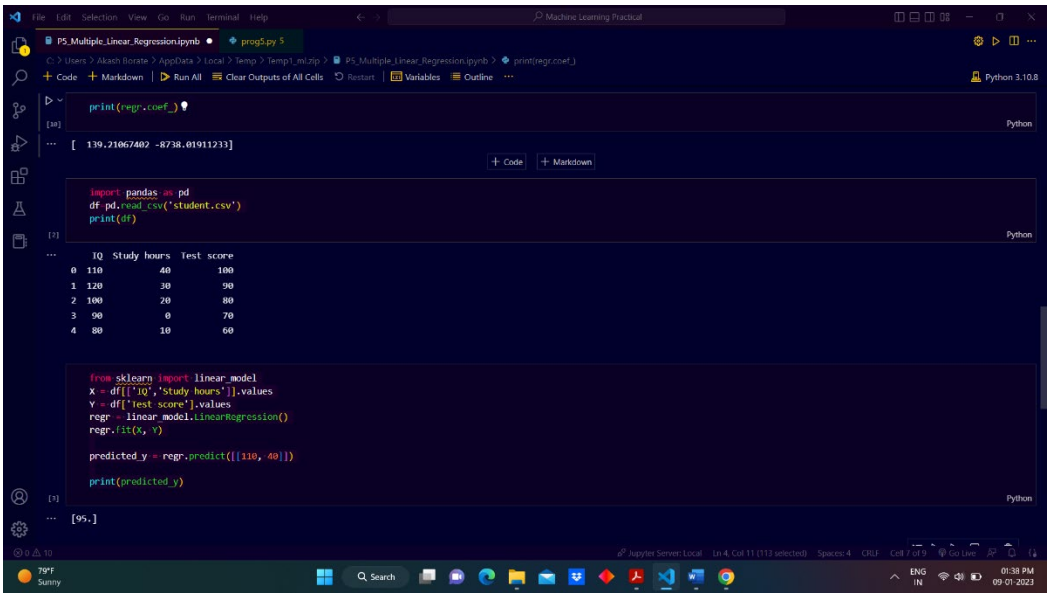
```
import pandas as pd
df=pd.read_csv('housing_data.csv')
print(df.head(10))
import numpy as np
X = df[['area (in cm)','no of bedrooms']].values
Y = df['price in lakhs'].values
print(X,Y)
from sklearn import linear_model
regr = linear_model.LinearRegression()
regr.fit(X, Y)
predicted_y = regr.predict([[1300, 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:



```
   area (in cm)  no of bedrooms  price in lakhs
0          2104               3         399900
1          1600               3         329900
2          2400               3         369000
3          1416               2         232000
4          3000               4         539900
5          1985               4         299900
6          1534               3         314900
7          1427               3         198999
8          1380               3         212000
9          1494               3         242500
```

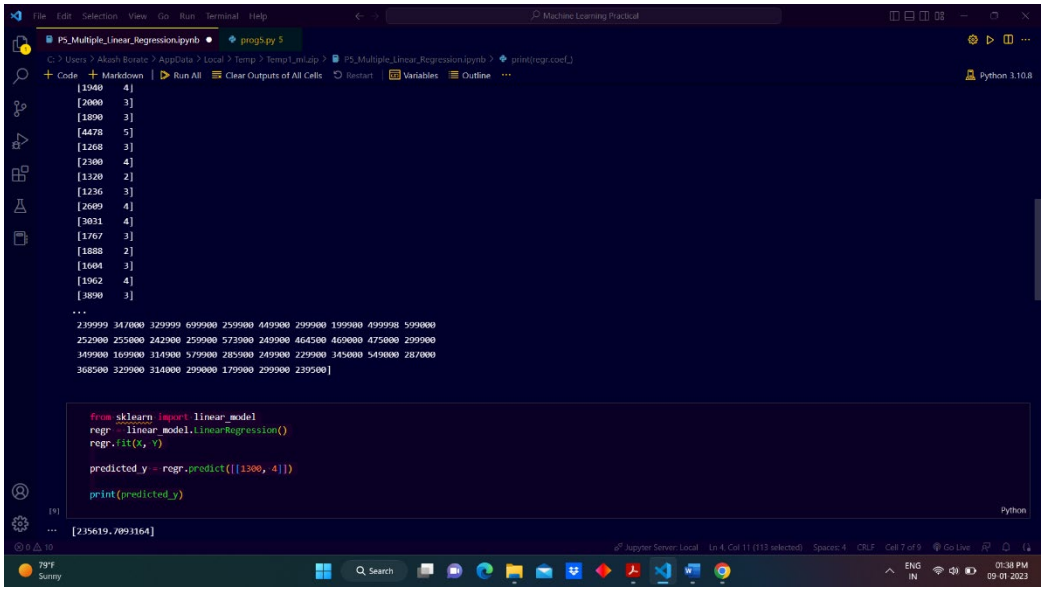
```
Output exceeds the size limit. Open the full output data in a text editor
[[2104    3]
 [1600    3]
 [2400    3]
 [1416    2]
 [3000    4]
 [1985    4]
```

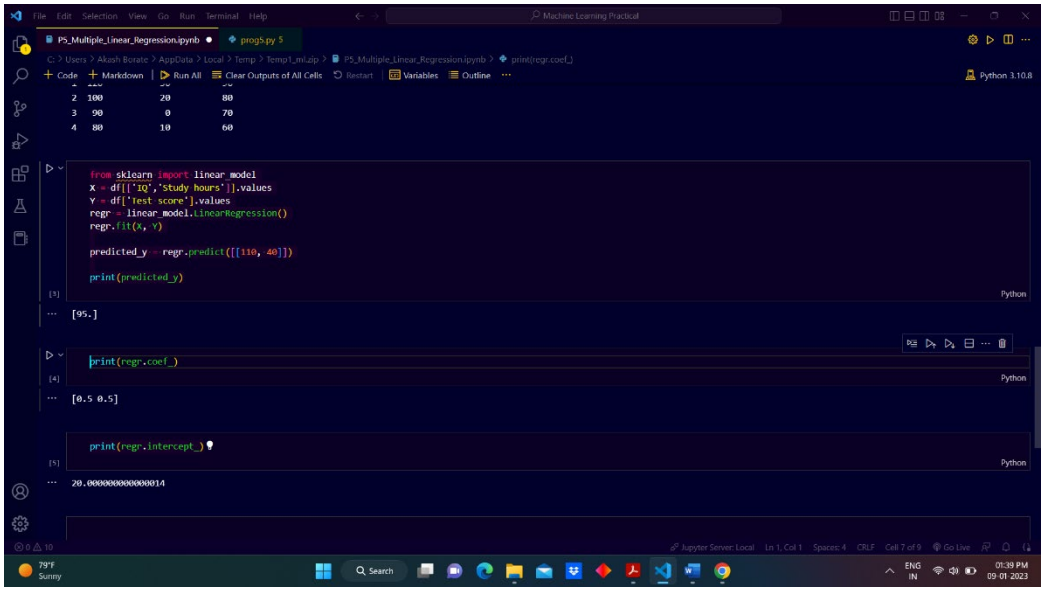


```
[ 139.21067402 -8738.01911233]
```

```
    IQ  Study hours  Test score
0  110           40         100
1  120           30          90
2  100           20          80
3   90            0          70
4   80           10          60
```

```
[95.]
```


P5_Multiple_Linear_Regression.ipynb
prog5.py 5
[1940 4]
[2000 3]
[1890 3]
[4478 5]
[1268 3]
[2300 4]
[1320 2]
[1236 3]
[2609 4]
[3031 4]
[1767 3]
[1888 2]
[1604 3]
[1962 4]
[3890 3]
239999 347000 329999 699900 259900 449900 299900 199900 499998 599000
252900 255000 242900 259900 573900 249900 464500 469000 475000 299900
349900 169900 314900 579900 285900 249900 229900 345000 549000 287000
368500 329900 314000 299000 179900 299900 239500]
from sklearn import linear_model
regr = linear_model.LinearRegression()
regr.fit(X, Y)
predicted_y = regr.predict([[1300, 4]])
print(predicted_y)
[235619.7093164]


P5_Multiple_Linear_Regression.ipynb
prog5.py 5
2 100 20 80
3 90 0 70
4 80 10 60
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)
[95.]
print(regr.coef_)
[0.5 0.5]
print(regr.intercept_)
20.000000000000014

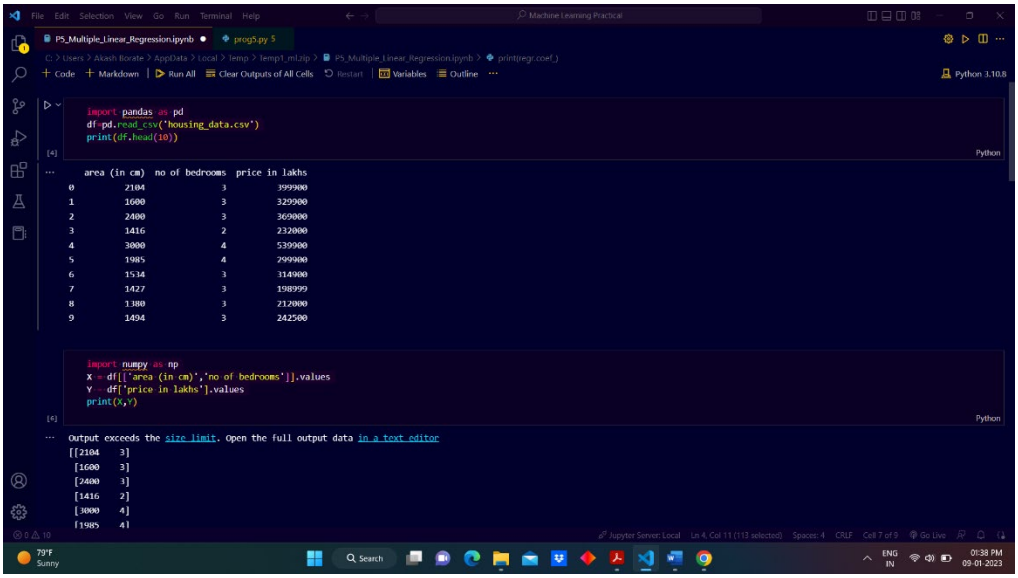
**6) Write a python program to implement Polynomial Regression for given dataset.**

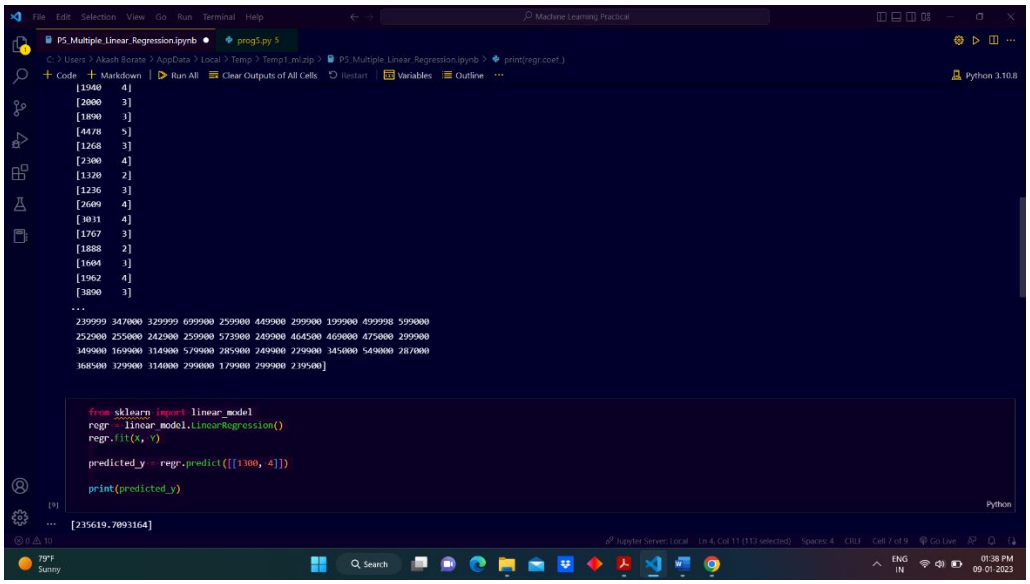
**Ans:**

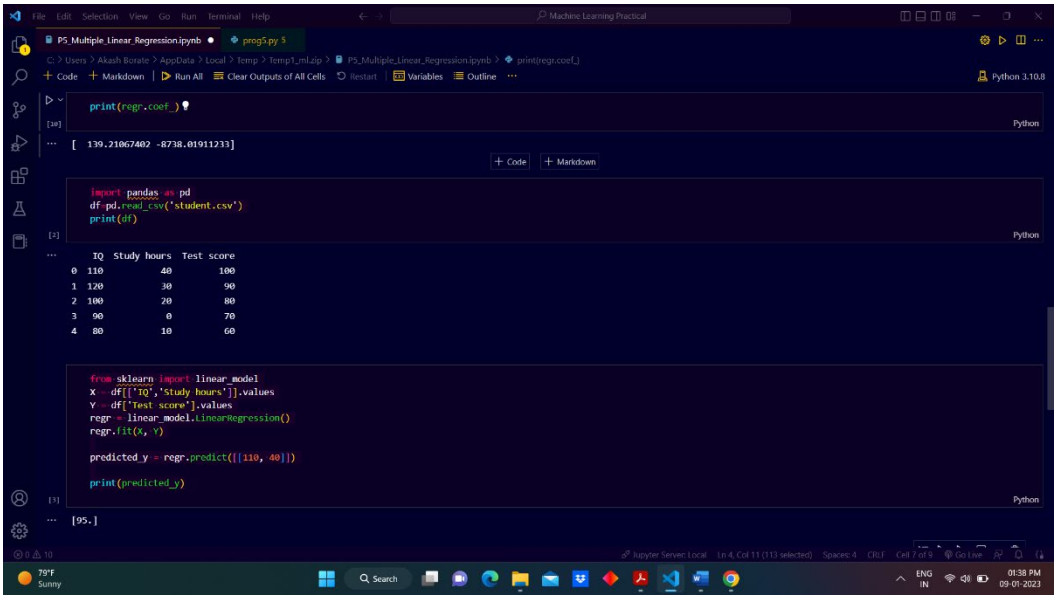
```
import pandas as pd
df=pd.read_csv('employees.csv')
print(df)
X=df.iloc[:,1:2].values
y=df.iloc[:,2].values
print(X,y)
#fitting the polynomial regression model to the dataset
from sklearn.preprocessing import PolynomialFeatures
poly_reg=PolynomialFeatures(degree=4)
X_poly=poly_reg.fit_transform(X)
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()
```

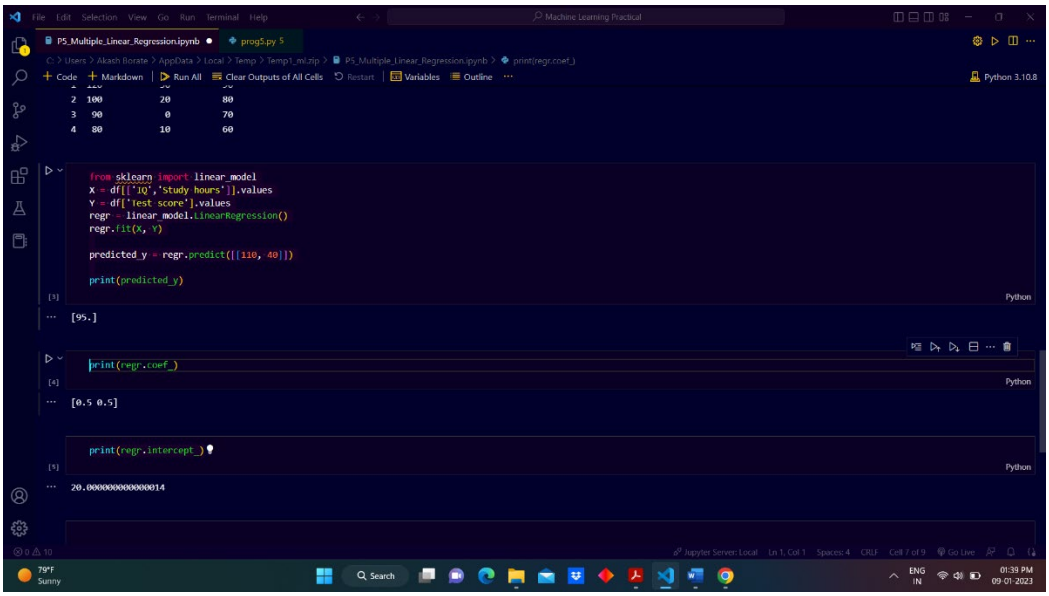
```
import numpy as np
lin_reg.predict(np.array([ [6.5] ]))
import numpy as np
lin_reg2.predict(poly_reg.fit_transform(np.array([ [6.5] ])))
```

**Output:**




print(regr.coef_)
[ 139.21067402 -8738.01911233]
import pandas as pd
df=pd.read_csv('student.csv')
print(df)
IQ Study hours Test score
0 110 40 100
1 120 30 90
2 100 20 80
3 90 0 70
4 80 10 60
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)
[95.]

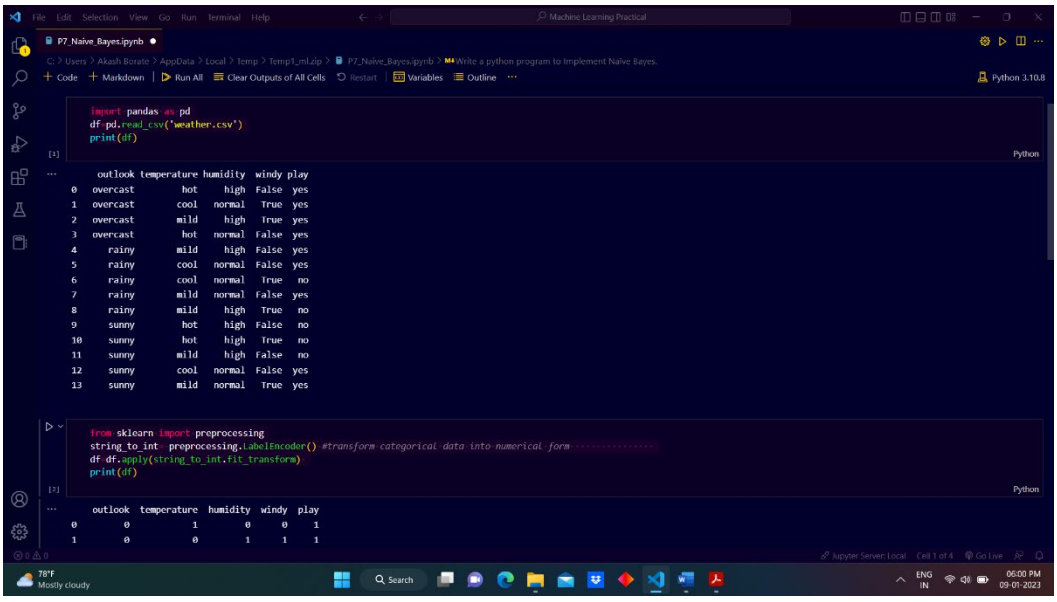

from sklearn import linear_model
X = df[['IQ','study hours']].values
Y = df['Test score'].values
regr = linear_model.LinearRegression()
regr.fit(X, Y)
predicted_y = regr.predict([[110, 40]])
print(predicted_y)
[95.]
print(regr.coef_)
[0.5 0.5]
print(regr.intercept_)
20.000000000000014

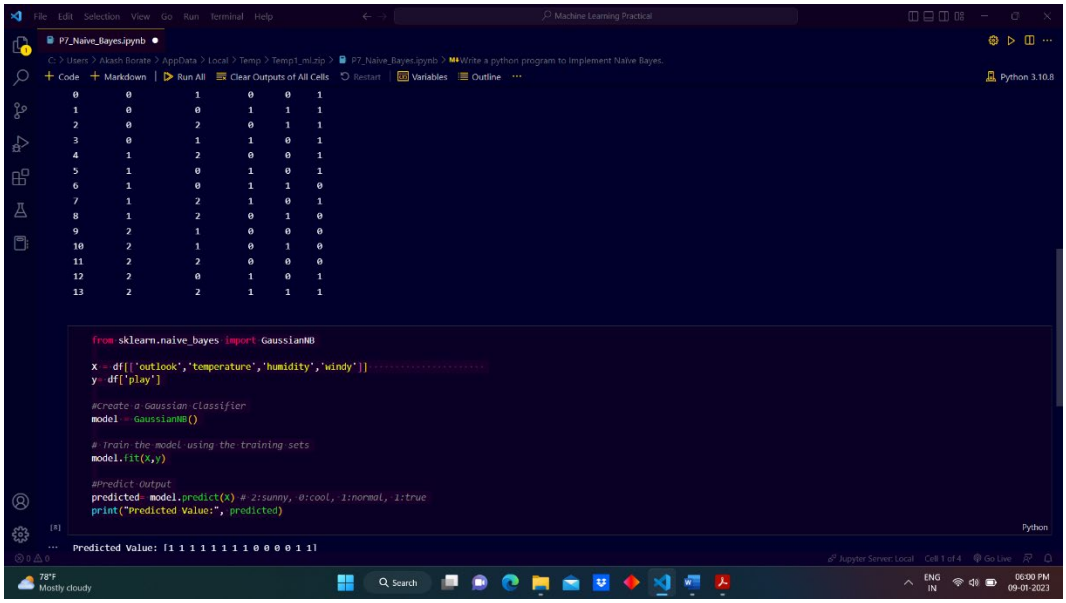
## 7) Write a python program to Implement Naïve Bayes.

**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:**



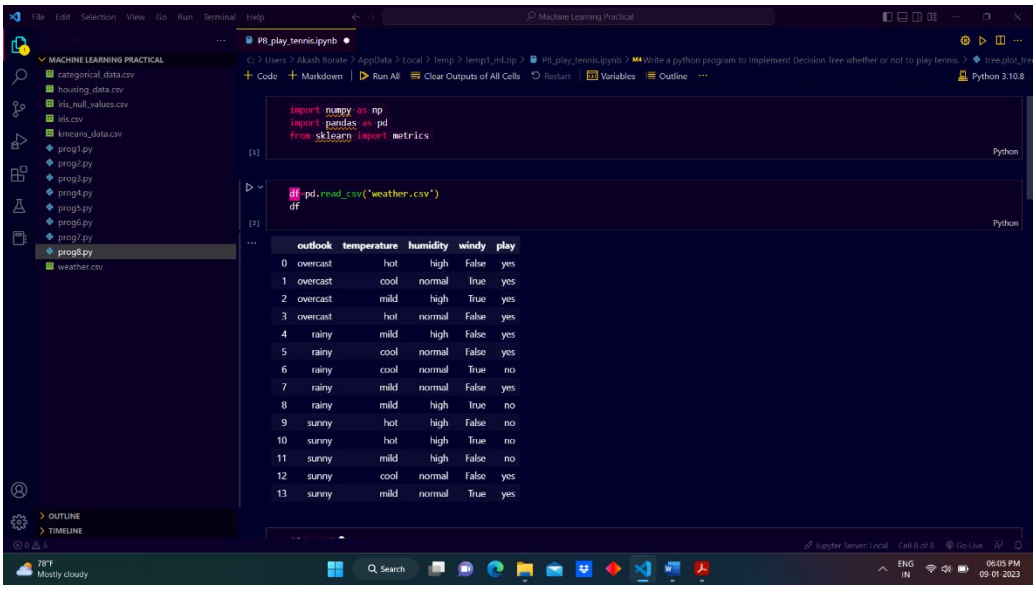

P7_Naive_Bayes.ipynb
Write a python program to Implement Naive Bayes.
Code
Markdown
Run All
Clear Outputs of All Cells
Restart
Variables
Outline
Python 3.10.8
0 0 1 0 0 1
1 0 0 1 1 1
2 0 2 0 1 1
3 0 1 1 0 1
4 1 2 0 0 1
5 1 0 1 0 1
6 1 0 1 1 0
7 1 2 1 0 1
8 1 2 0 1 0
9 2 1 0 0 0
10 2 1 0 1 0
11 2 2 0 0 0
12 2 0 1 0 1
13 2 2 1 1 1
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)
Python
Predicted Value: [1 1 1 1 1 1 1 1 0 0 0 0 1 1]
Jupyter Server: Local
Cell 1 of 4
Go Live
78°F
Mostly cloudy
Search
ENG
IN
06:00 PM
09-01-2023

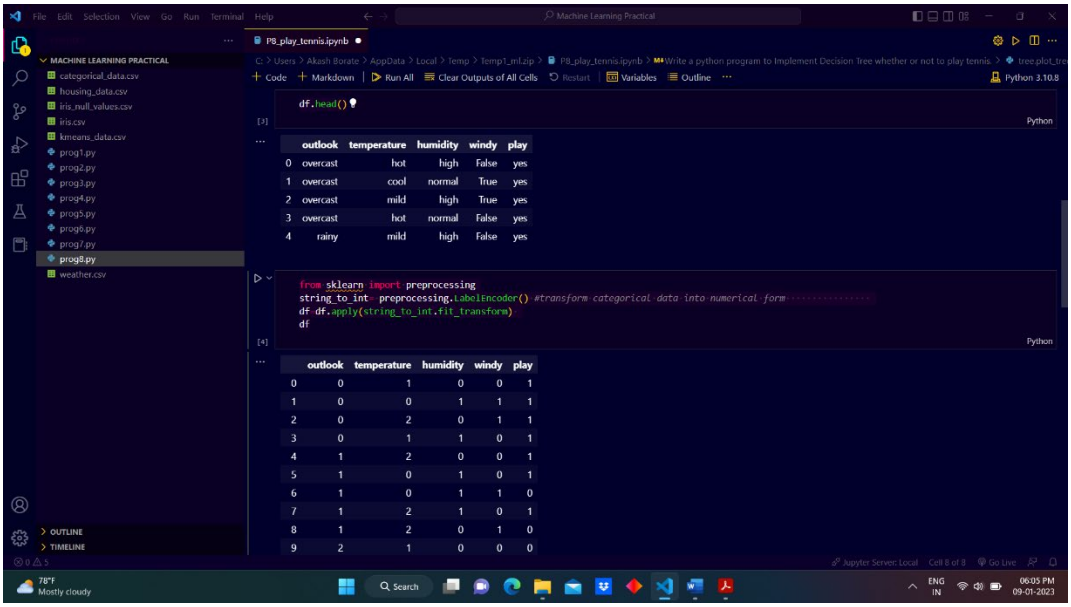
**8) Write a python program to Implement Decision Tree whether or not to play tennis.**

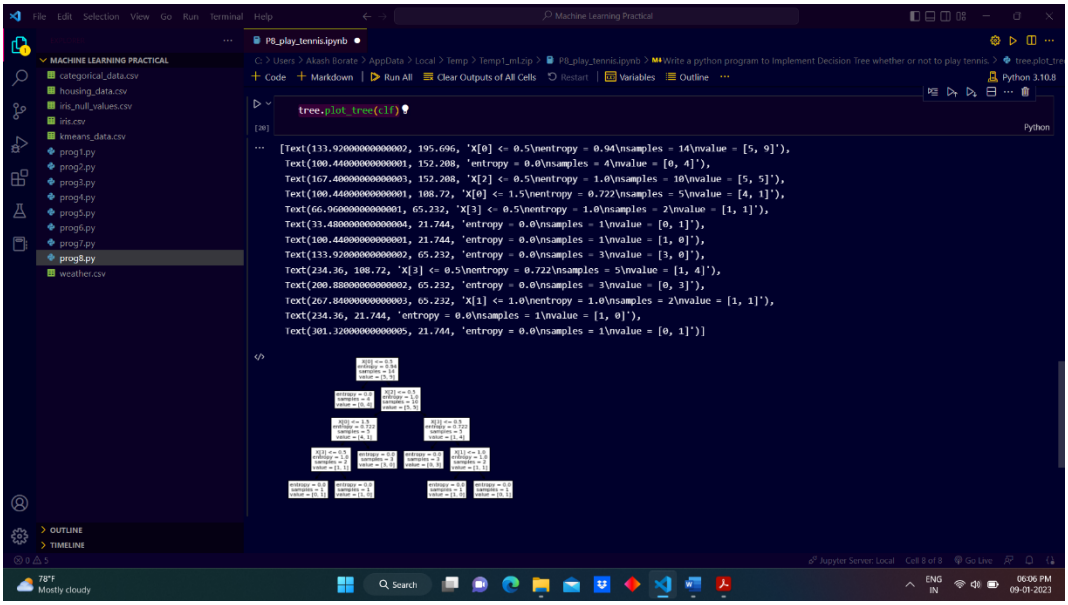
**Ans:**

```
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:**




df.head()
outlook temperature humidity windy play
0 overcast hot high False yes
1 overcast cool normal True yes
2 overcast mild high True yes
3 overcast hot normal False yes
4 rainy mild high False yes
from sklearn import preprocessing
string_to_int= preprocessing.LabelEncoder() #transform categorical data into numerical form
df_df.apply(string_to_int.fit_transform)
df
outlook temperature humidity windy play
0 0 1 0 0 1
1 0 0 1 1 1
2 0 2 0 1 1
3 0 1 1 0 1
4 1 2 0 0 1
5 1 0 1 0 1
6 1 0 1 1 0
7 1 2 1 0 1
8 1 2 0 1 0
9 2 1 0 0 0


tree.plot_tree(clf)
[Text(133.92000000000002, 195.696, 'X[0] <= 0.5\nentropy = 0.94\nsamples = 14\nvalue = [5, 9]'),
Text(100.44000000000001, 152.208, 'entropy = 0.0\nsamples = 4\nvalue = [0, 4]'),
Text(167.40000000000003, 152.208, 'X[2] <= 0.5\nentropy = 1.0\nsamples = 10\nvalue = [5, 5]'),
Text(100.44000000000001, 108.72, 'X[0] <= 1.5\nentropy = 0.722\nsamples = 5\nvalue = [4, 1]'),
Text(66.96000000000001, 65.232, 'X[3] <= 0.5\nentropy = 1.0\nsamples = 2\nvalue = [1, 1]'),
Text(33.48000000000004, 21.744, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1]'),
Text(100.44000000000001, 21.744, 'entropy = 0.0\nsamples = 1\nvalue = [1, 0]'),
Text(133.92000000000002, 65.232, 'entropy = 0.0\nsamples = 3\nvalue = [3, 0]'),
Text(234.36, 108.72, 'X[3] <= 0.5\nentropy = 0.722\nsamples = 5\nvalue = [1, 4]'),
Text(200.88000000000002, 65.232, 'entropy = 0.0\nsamples = 3\nvalue = [0, 3]'),
Text(267.8400000000003, 65.232, 'X[1] <= 1.0\nentropy = 1.0\nsamples = 2\nvalue = [1, 1]'),
Text(234.36, 21.744, 'entropy = 0.0\nsamples = 1\nvalue = [1, 0]'),
Text(301.3200000000005, 21.744, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1]')]

**9) 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:**

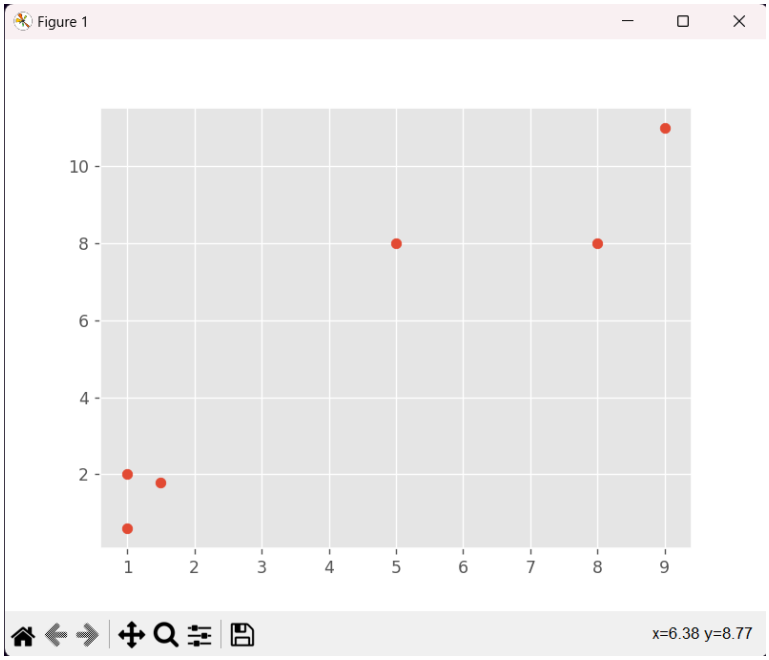


Figure 1
10
8
6
4
2
1
2
3
4
5
6
7
8
9
x=6.38 y=8.77

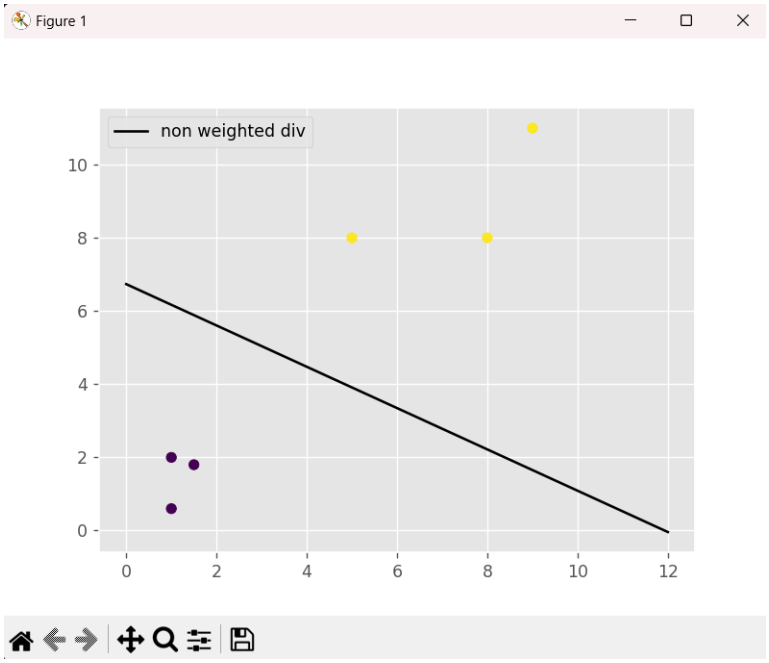
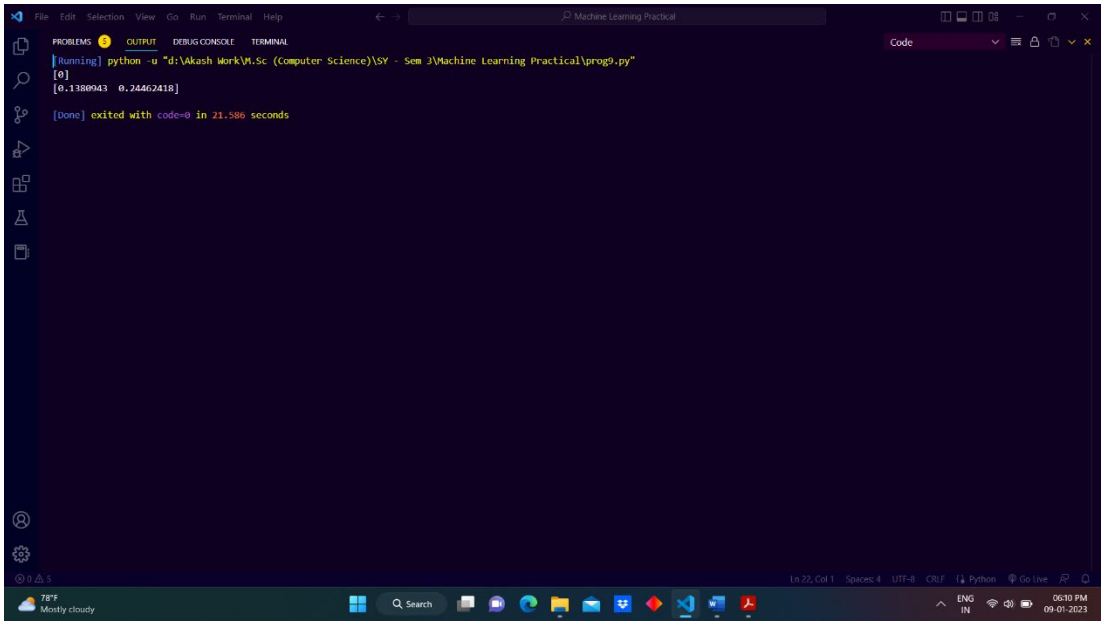


Figure 1
non weighted div
10
8
6
4
2
0
0
2
4
6
8
10
12

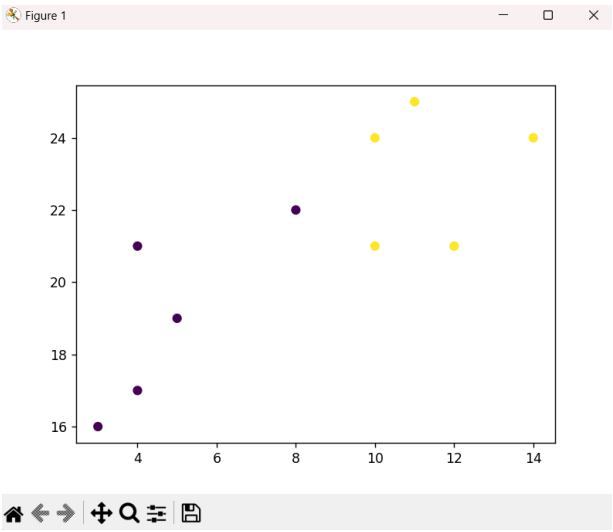

File Edit Selection View Go Run Terminal Help
Machine Learning Practical
PROBLEMS 5 OUTPUT DEBUG CONSOLE TERMINAL
Code
[Running] python -u "d:\Akash Work\M.Sc (Computer Science)\SY - Sem 3\Machine Learning Practical\prog9.py"
[0]
[0.1380943 0.24462418]
[Done] exited with code=0 in 21.586 seconds
Ln 22, Col 1 Spaces: 4 UTF-8 CRLF Python Go Live
78°F
Mostly cloudy
Search
ENG
IN
06:10 PM
09-01-2023

**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:**



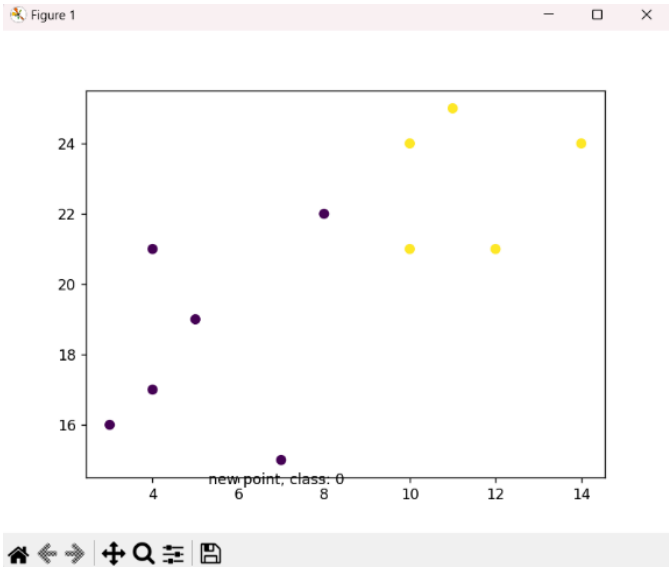
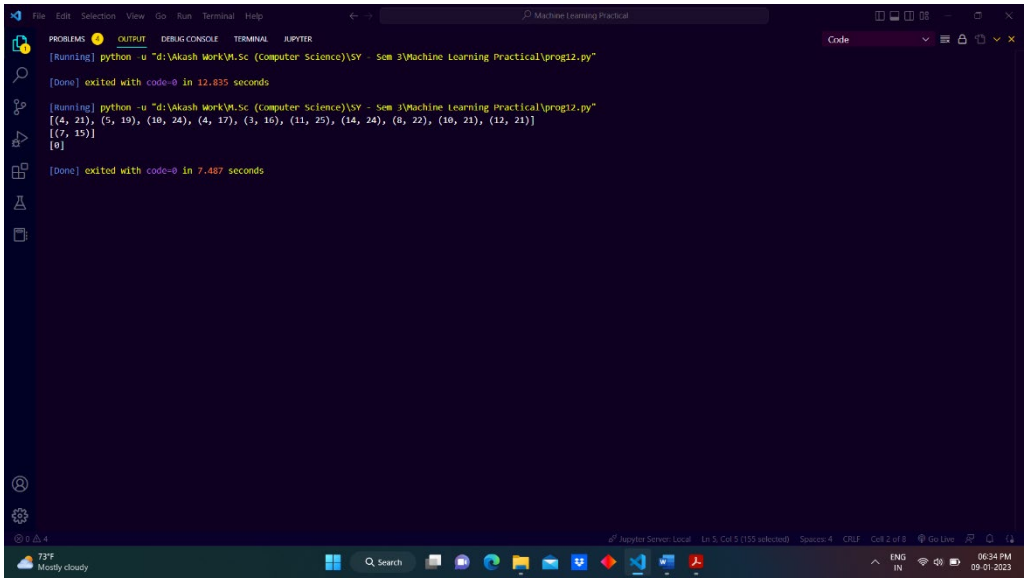


Figure 1
24
22
20
18
16
4
6
8
10
12
14
new point, class: 0


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
JUPYTER
Code
[Running] python -u "d:\Akash Work\M.Sc (Computer Science)\SY - Sem 3\Machine Learning Practical\prog12.py"
[Done] exited with code=0 in 12.835 seconds
[Running] python -u "d:\Akash Work\M.Sc (Computer Science)\SY - Sem 3\Machine Learning Practical\prog12.py"
[(4, 21), (5, 19), (10, 24), (4, 17), (3, 16), (11, 25), (14, 24), (8, 22), (10, 21), (12, 21)]
[(7, 15)]
[0]
[Done] exited with code=0 in 7.487 seconds