MACHINE LEARNING

6.PROGRAM

import numpy as np

import matplotlib.pyplot as plt

def estimate\_coef(x, y):

n = np.size(x)

m\_x = np.mean(x)

m\_y = np.mean(y)

SS\_xy = np.sum(y\*x) - n\*m\_y\*m\_x

SS\_xx = np.sum(x\*x) - n\*m\_x\*m\_x

# calculating regression coefficients

b\_1 = SS\_xy / SS\_xx

b\_0 = m\_y - b\_1\*m\_x

return (b\_0, b\_1)

# plotting the actual points as scatter plot

def plot\_regression\_line(x, y, b):

plt.scatter(x, y, color = "m",

marker = "o", s = 30)

y\_pred = b[0] + b[1]\*x

plt.plot(x, y\_pred, color = "g")

plt.xlabel('x')

plt.ylabel('y')

plt.show()

def main():

# observations / data

x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

y = np.array([1, 3, 2, 5, 7, 8, 8, 9, 10, 12])

# estimating coefficients

b = estimate\_coef(x, y)

print("Estimated coefficients:\nb\_0 = {} \

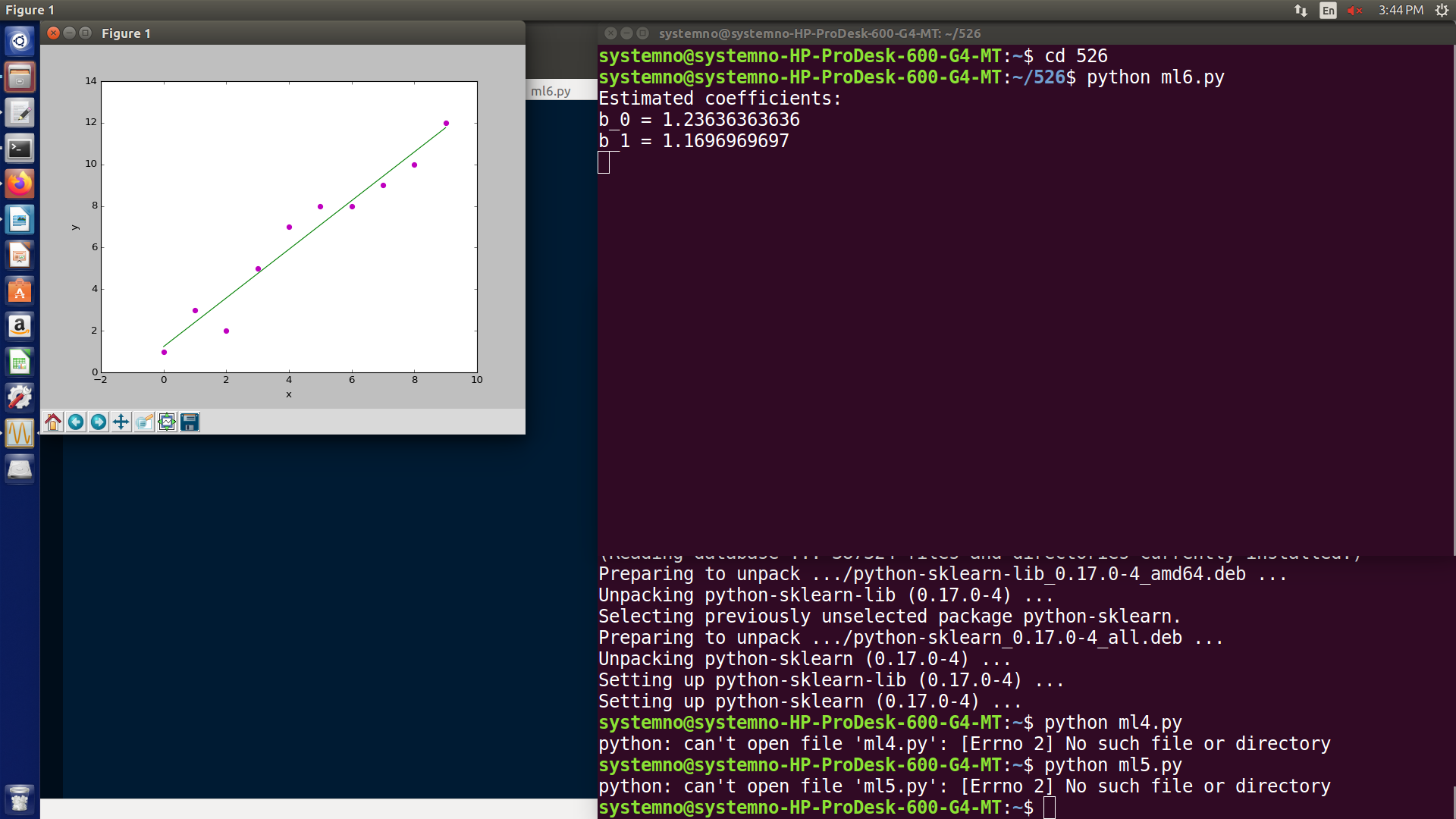
\nb\_1 = {}".format(b[0], b[1]))

# plotting regression line

plot\_regression\_line(x, y, b)

if \_\_name\_\_ == "\_\_main\_\_":

main()



3.PROGRAM

from math import sqrt

def eaclidean\_distance(row1,row2):

distance=0

for i in range(len(row1)-1):

distance += (row1[i]-row2[i])\*\*2

return sqrt(distance)

def get\_neighbor(train,test\_row,num\_neighbors):

distances=list()

for train\_row in train:

dist=eaclidean\_distance(test\_row,train\_row)

distances.append((dist,train\_row))

distances.sort()

neighbors=list()

for i in range(num\_neighbors):

neighbors.append(distances[i][1])

return neighbors

dataset=[[1,2,0], [3,4,0] ,[3,3,0],[7,2,1],[5,2,2],[7,3,1]]

neighbors=get\_neighbor(dataset,[5,4,2],2)

for neighbor in neighbors:

print(neighbor)

OUTPUT:

python ml3.py

[3, 4, 0]

[5, 2, 2]

4.PROGRAM

from pandas import DataFrame

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

Data={'x':[1.713,0.180,0.353,0.940,1.486,1.266,1.540,0.5459,0.773],'y':[1.586,1.786,1.240,1.566,1.759,10.106,0.419,1.799,0.816]}

df=DataFrame(Data,columns=['x','y'])

kmeans=KMeans(n\_clusters=3).fit(df)

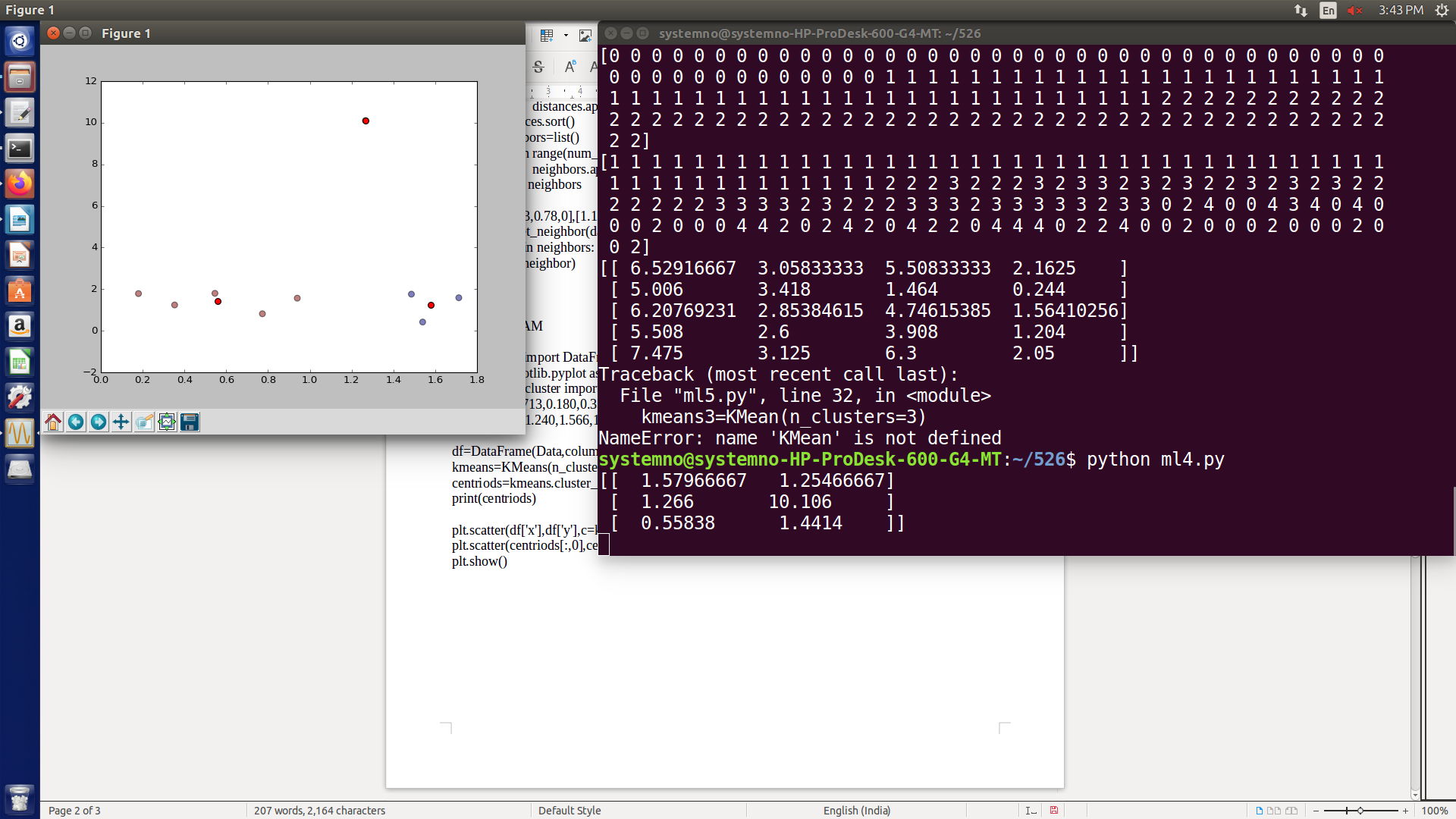
centriods=kmeans.cluster\_centers\_

print(centriods)

plt.scatter(df['x'],df['y'],c=kmeans.labels\_.astype(float),s=50,alpha=0.5)

plt.scatter(centriods[:,0],centriods[:,1],c='red',s=50)

plt.show()



5.PROGRAM

import numpy as np

import pandas as pd

import matplotlib as plt

from sklearn.cluster import KMeans

from sklearn import datasets

df=datasets.load\_iris()

x=df.data

y=df.target

print(x)

print(y)

kmean=KMeans(n\_clusters=5)

y\_kmean=kmean.fit\_predict(x)

print(y\_kmean)

print(kmean.cluster\_centers\_)

Error=[]

for i in range(1,11):

kmeans=KMeans(n\_clusters=i).fit(x)

kmeans.fit(x)

Error.append(kmeans.inertia\_)

import matplotlib.pyplot as plt

plt.plot(range(1,11),Error)

plt.title('Elbow method')

plt.xlabel('no of clusters')

plt.ylabel('Error')

plt.show()

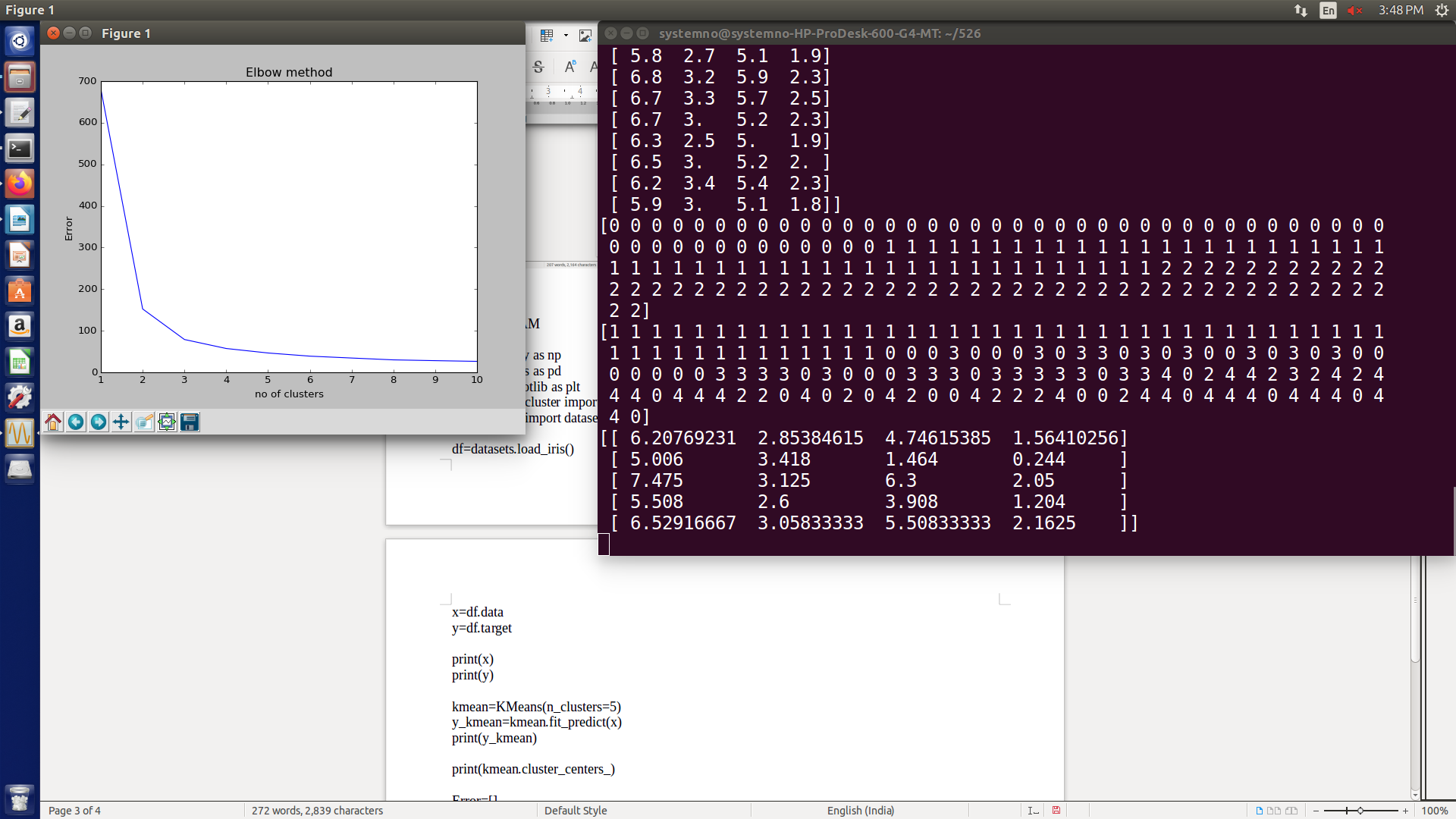
kmeans3=KMean(n\_clusters=3)

y\_kmeans3=kmeans3.fit\_predict(x)

print(y\_kmeans3)

print(kmeans3.cluster\_centers\_)

OUTPUT



7.PROGRAM

print("NAIVE BAYES ENGLISH TEST CLASSIFICATION")

import numpy as np, pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.datasets import fetch\_20newsgroups

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.naive\_bayes import MultinomialNB

from sklearn.pipeline import make\_pipeline

from sklearn.metrics import confusion\_matrix, accuracy\_score

sns.set()

data = fetch\_20newsgroups()

text\_categories = data.target\_names

train\_data = fetch\_20newsgroups(subset="train", categories=text\_categories)

test\_data = fetch\_20newsgroups(subset="test", categories=text\_categories)

print("We have {} unique classes".format(len(text\_categories)))

print("We have {} training samples".format(len(train\_data.data)))

print("We have {} test samples".format(len(test\_data.data)))

model = make\_pipeline(TfidfVectorizer(), MultinomialNB())

model.fit(train\_data.data, train\_data.target)

predicted\_categories = model.predict(test\_data.data)

print(np.array(test\_data.target\_names)[predicted\_categories])

mat = confusion\_matrix(test\_data.target, predicted\_categories)

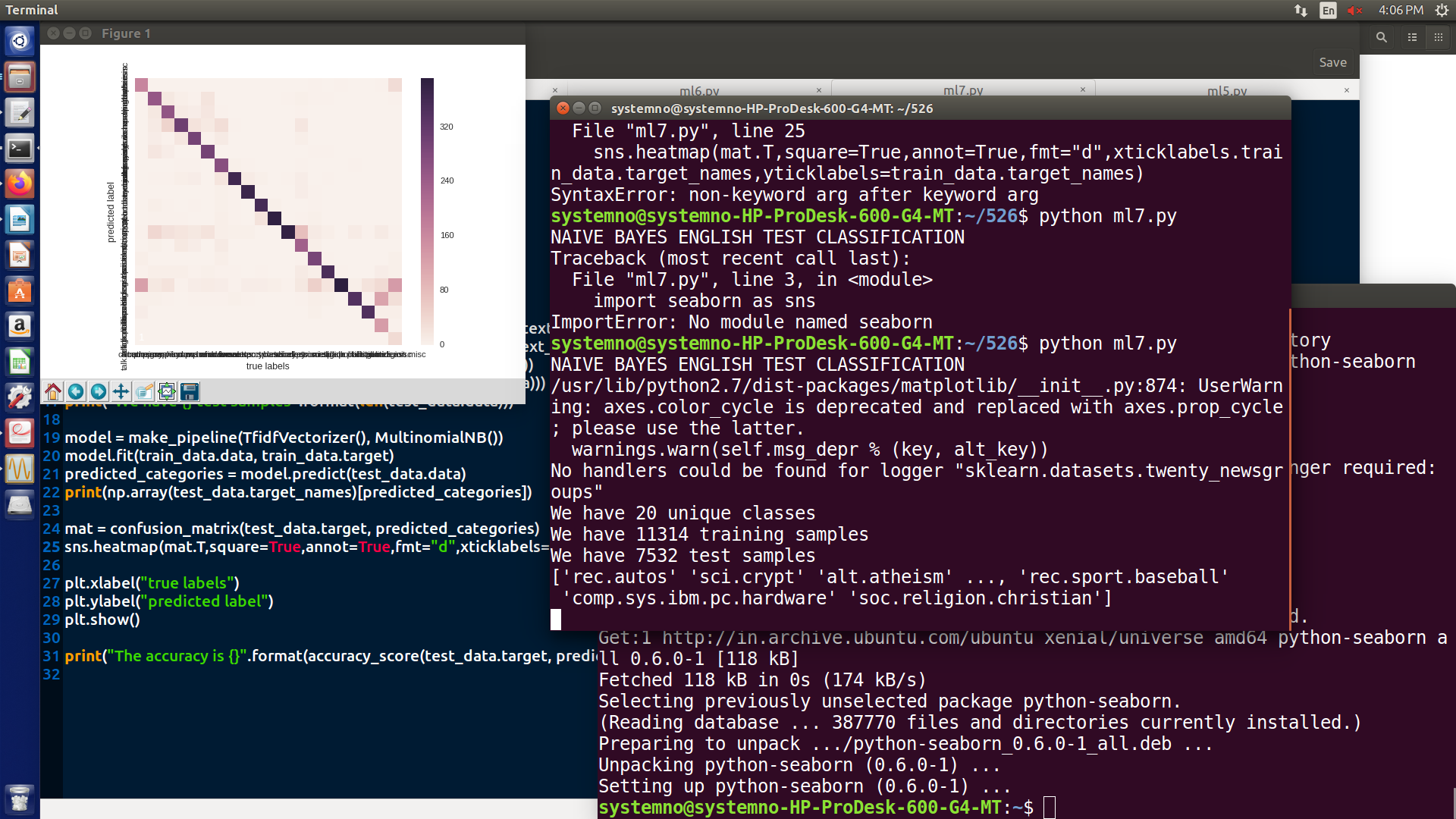
sns.heatmap(mat.T,square=True,annot=True,fmt="d",xticklabels=train\_data.target\_names,yticklabels=train\_data.target\_names)

plt.xlabel("true labels")

plt.ylabel("predicted label")

plt.show()

print("The accuracy is {}".format(accuracy\_score(test\_data.target, predicted\_categories)))

****