A PRACTICAL REPORT

"Applied Artificial intelligence Practical"

FOR

UNIVERSITY OF MUMBAI

SUBMITTED BY

NAME: Rupesh Patil

Roll NO: 17

M.Sc. (INFORMATION TECHNOLOGY)

PART-2 SEM-III

2020 – 2021

CONDUCTED AT

VALIA C.L. COLLEGE OF COMMERCE

&

VALIA L.C. COLLEGE OF ARTS

ANDHERI (W), MUMBAI - 400053

Cosmopolitans

VALIA C. L. COLLEGE OF COMMERCE

&

VALIA L.C. COLLEGE OF ARTS

(Affiliated to University of Mumbai)

D.N. Nagar, Andheri (W), Mumbai-400053.

CERTIFICATE

This is to certify that **Mr. rupesh patil**, Roll No.: **17**, studying in Master of Science (Information Technology) SEM: III has satisfactorily completed the practicals in the **Course: Applied Artificial Intelligence"**, as prescribed by the University of Mumbai, during the academic year 2020-2021.

_____ _____

Course In charge **Head, Dept. Of Information Technology**

_____

DATE & COLLEGE SEAL **Examiner**

INDEX

Practical No.	Topic	Signature
1	Design an expert system using AIML	
2	Bot using AIML	
3	Bayes Theorem	
4	Conditional probability and joint probability	
6	Fuzzy based Application	
7	Supervised learning model	
8	Clustering Algorithm (K-means)	

Practical 1

Aim: Design an Expert system using AIML.

Description: An expert system is a computer program that uses artificial intelligence (AI) technologies to simulate the judgment and behavior of a human or an organization that has expert knowledge and experience in a particular field.

Typically, an expert system incorporates a knowledge base containing accumulated experience and an inference or rules engine -- a set of rules for applying the knowledge base to each particular situation that is described to the program. The system's capabilities can be enhanced with additions to the knowledge base or to the set of rules. Current systems may include machine learning capabilities that allow them to improve their performance based on experience, just ashumans do.

Code:

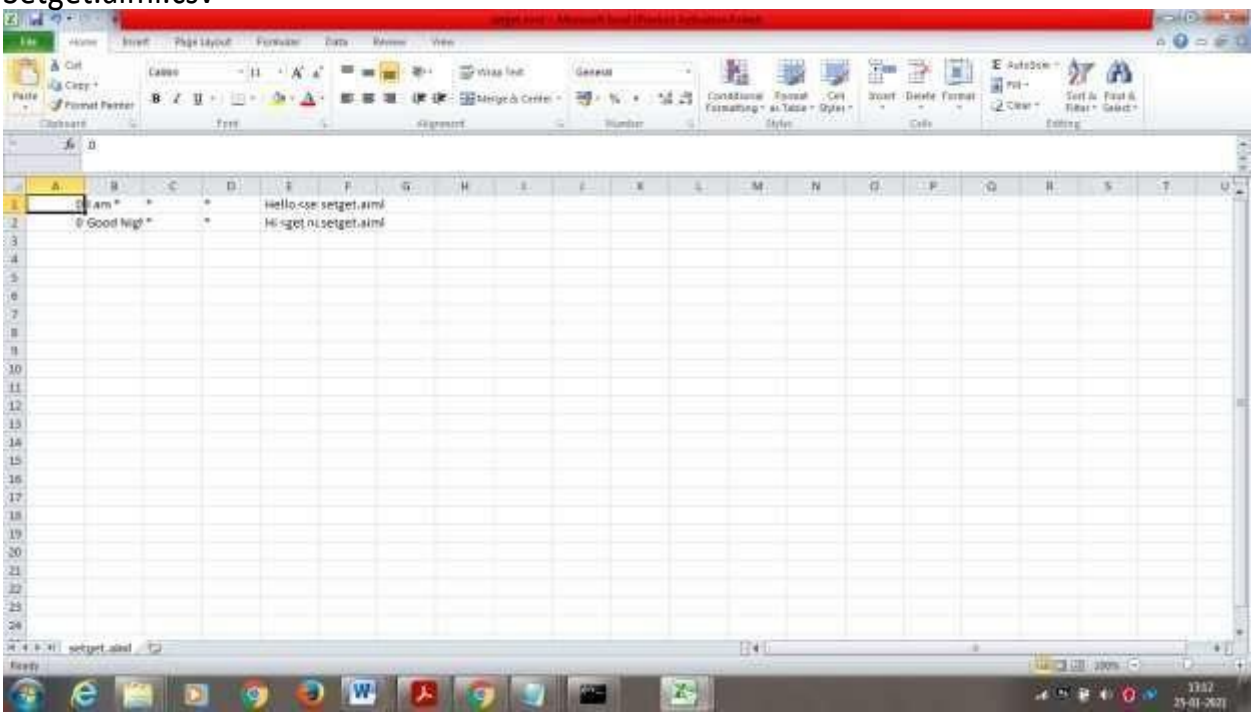
setget.aiml

```
<?xml version = "1.0" encoding = "UTF-8"?>
<aiml version = "1.0.1" encoding = "UTF-8"?>
  <category>
    <pattern>I am *</pattern>
    <template>
      Hello <set name = "username"> Welcome to our clinic <star/>! </set>
    </template>
  </category>
  <category>
    <pattern>Good Night</pattern>
    <template>
      Hi <get name = "username"/> Thanks for the conversation! take the medicine
properly
    </template>
  </category>
</aiml>
```

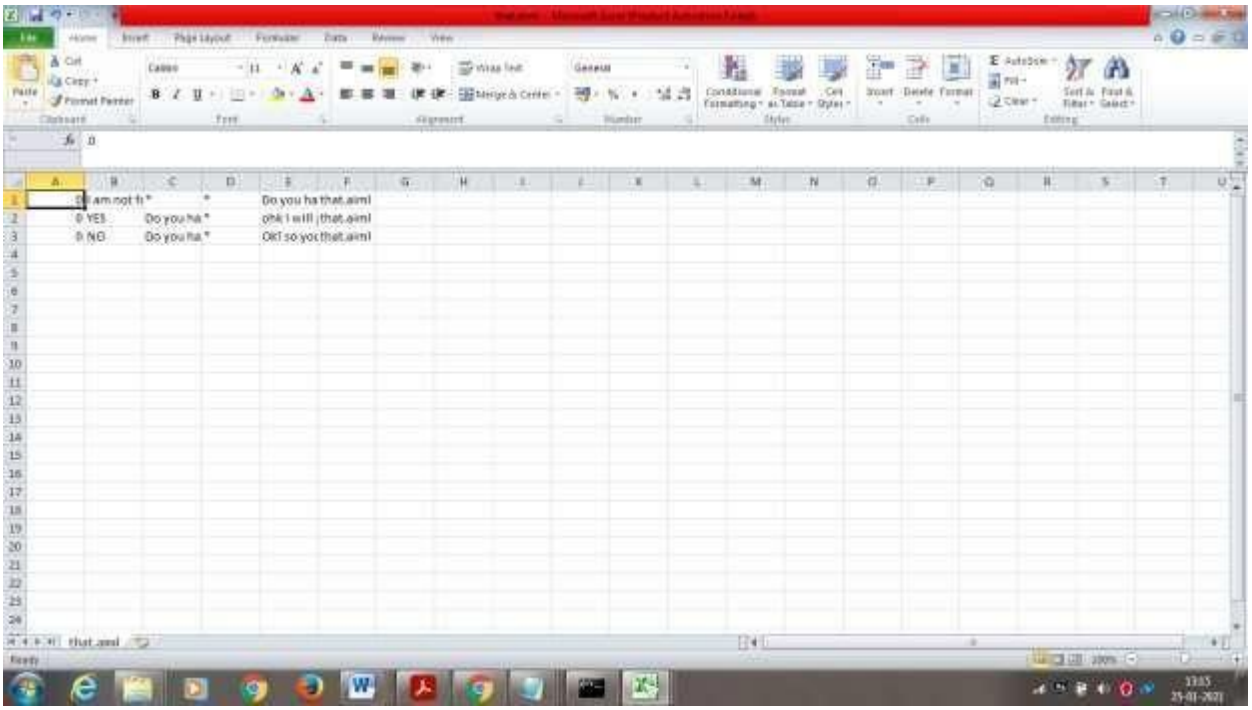
that.aiml

```
<?xml version = "1.0" encoding = "UTF-8"?>
<aiml version = "1.0.1" encoding = "UTF-8"?>
  <category>
    <pattern>I am not feeling well</pattern>
    <template>Do you have cough or cold</template>
  </category>
  <category>
    <pattern>YES</pattern>
    <that>Do you have cough or cold</that>
    <template>ohk I will prescribe you some medicene and syrup too.</template>
  </category>
  <category>
    <pattern>NO</pattern>
    <that>Do you have cough or cold</that>
    <template>Ok! so you have viral infection dont worry you will be fine.</template>
  </category>
</aiml>
```

Setget.aiml.csv



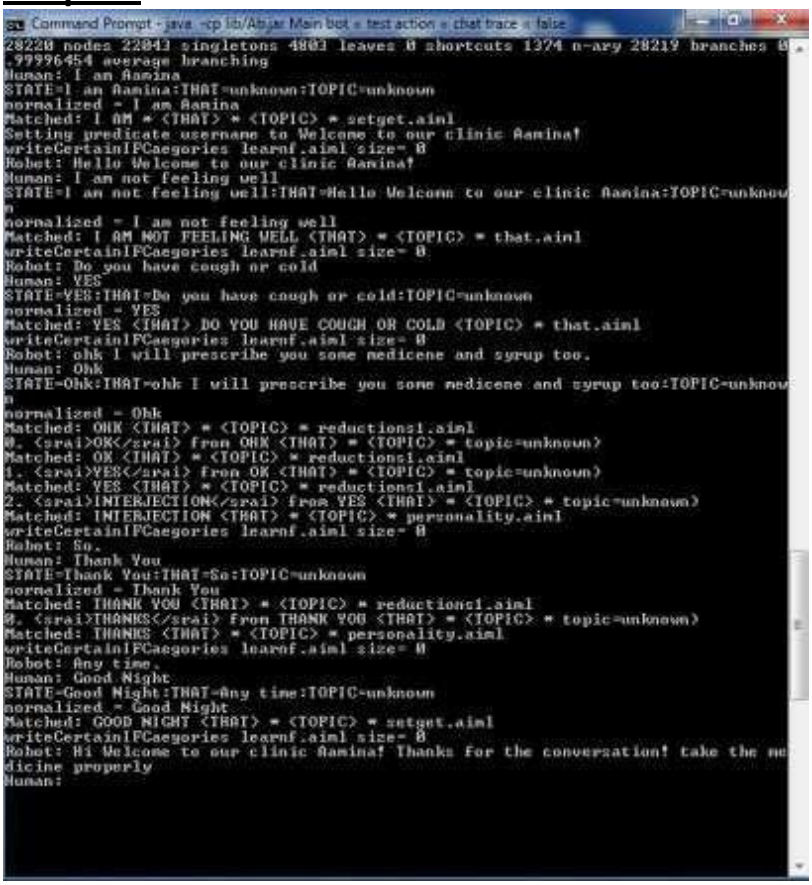
That.aiml.csv



<u>Execution command:</u>

java -cp lib/Ab.jar Main bot = test action = chat trace = false

Output:-


Command Prompt - java -cp lib/Ab.jar Main bot = test action = chat trace = false
28220 nodes 22843 singletons 4803 leaves 0 shortcuts 1374 n-ary 28219 branches 0
.99996454 average branching
Human: I am Aamina
STATE=I am Aamina:THAT=unknown:TOPIC=unknown
normalized = I am Aamina
Matched: I AM * <THAT> * <TOPIC> * setget.aiml
Setting predicate username to Welcome to our clinic Aamina!
writeCertainIPCategories learnf.aiml size= 0
Robot: Hello Welcome to our clinic Aamina!
Human: I am not feeling well
STATE=I am not feeling well:THAT=Hello Welcome to our clinic Aamina:TOPIC=unknow
n
normalized = I am not feeling well
Matched: I AM NOT FEELING WELL <THAT> * <TOPIC> * that.aiml
writeCertainIPCategories learnf.aiml size= 0
Robot: Do you have cough or cold
Human: YES
STATE=YES:THAT=Do you have cough or cold:TOPIC=unknown
normalized = YES
Matched: YES <THAT> DO YOU HAVE COUGH OR COLD <TOPIC> * that.aiml
writeCertainIPCategories learnf.aiml size= 0
Robot: ohk I will prescribe you some medicene and syrup too.
Human: Ohk
STATE=Ohk:THAT=ohk I will prescribe you some medicene and syrup too:TOPIC=unknow
n
normalized = Ohk
Matched: OHK <THAT> * <TOPIC> * reductions1.aiml
0. <srai>OK</srai> from OHK <THAT> * <TOPIC> * topic=unknown)
Matched: OK <THAT> * <TOPIC> * reductions1.aiml
1. <srai>YES</srai> from OK <THAT> * <TOPIC> * topic=unknown)
Matched: YES <THAT> * <TOPIC> * reductions1.aiml
2. <srai>INTERJECTION</srai> from YES <THAT> * <TOPIC> * topic=unknown)
Matched: INTERJECTION <THAT> * <TOPIC> * personality.aiml
writeCertainIPCategories learnf.aiml size= 0
Robot: So,
Human: Thank You
STATE=Thank You:THAT=So:TOPIC=unknown
normalized = Thank You
Matched: THANK YOU <THAT> * <TOPIC> * reductions1.aiml
0. <srai>THANKS</srai> from THANK YOU <THAT> * <TOPIC> * topic=unknown)
Matched: THANKS <THAT> * <TOPIC> * personality.aiml
writeCertainIPCategories learnf.aiml size= 0
Robot: Any time,
Human: Good Night
STATE=Good Night:THAT=Any time:TOPIC=unknown
normalized = Good Night
Matched: GOOD NIGHT <THAT> * <TOPIC> * setget.aiml
writeCertainIPCategories learnf.aiml size= 0
Robot: Hi Welcome to our clinic Aamina! Thanks for the conversation! take the me
dicine properly
Human:

Practical 2: Bot using AIML

Aim: Design a bot using AIML

Description: AIML stands for Artificial Intelligence Modelling Language. AIML is an XML based markup language meant to create artificial intelligent applications. AIML makes it possible to create human interfaces while keeping the implementation simple to program, easy to understand and highly maintainable. This tutorial will teach you the basics of AIML. All the basic components of AIML with suitable examples have been discussed in this tutorial.

AIML Tags/Description

S.No.	AIML Tag / Description
1	<aiml> Defines the beginning and end of a AIML document.
2	<category> Defines the unit of knowledge in Alicebot's knowledge base.
3	<pattern> Defines the pattern to match what a user may input to an Alicebot.
4	<template> Defines the response of an Alicebot to user's input.

S.No.	AIML Tag / Description
1	<star> Used to match wild card * character(s) in the <pattern> Tag.
2	<srai> Multipurpose tag, used to call/match the other categories.

3	<random> Used <random> to get random responses.
4	 Usedto represent multiple responses.
5	<set> Used to set value in an AIML variable.
6	<get> Used to get value stored in an AIML variable.
7	<that> Used in AIML to respond based on the context.
8	<topic> Used in AIML to store a context so that later conversation can be done based on that context.
9	<think> Used in AIML to store a variable without notifying the user.
10	<condition> Similar to switch statements in programming language. It helps ALICE to respond to matching input.

Code:

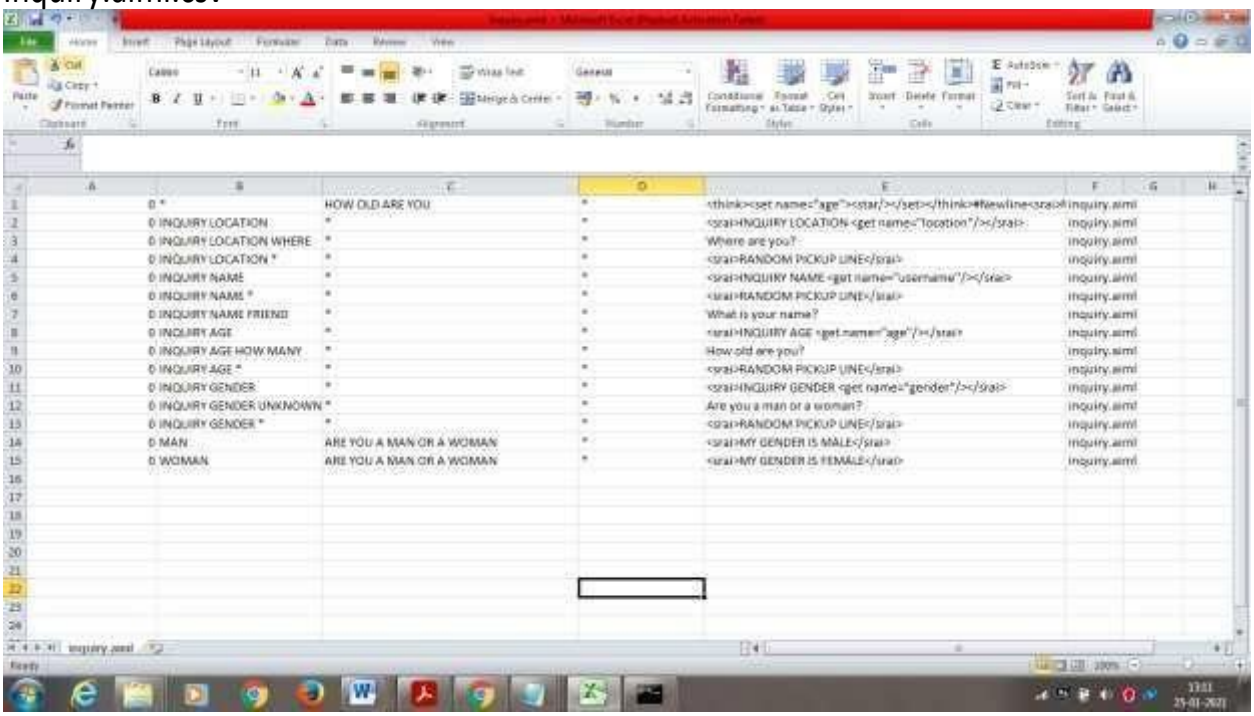
Inquiry.aiml

```
<?xml version="1.0" encoding="UTF-8"?>
<aiml>
<category><pattern>*</pattern><that>HOW OLD ARE YOU</that>
```

<template><think><set name="age"><star/></set></think>
<srai>MY AGE IS <star/></srai></template>
</category>
<category><pattern>INQUIRY LOCATION</pattern>
<template><srai>INQUIRY LOCATION <get name="location"/></srai></template>
</category>
<category><pattern>INQUIRY LOCATION WHERE</pattern>
<template>Where are you?</template>
</category>
<category><pattern>INQUIRY LOCATION *</pattern>
<template><srai>RANDOM PICKUP LINE</srai></template>
</category>
<category><pattern>INQUIRY NAME</pattern>
<template><srai>INQUIRY NAME <get name="userame"/></srai></template>
</category>
<category><pattern>INQUIRY NAME *</pattern>
<template><srai>RANDOM PICKUP LINE</srai></template>
</category>
<category><pattern>INQUIRY NAME FRIEND</pattern>
<template>What is your name?</template>
</category>
<category><pattern>INQUIRY AGE</pattern>
<template><srai>INQUIRY AGE <get name="age"/></srai></template>
</category>
<category><pattern>INQUIRY AGE HOW MANY</pattern>
<template>How old are you?</template>
</category>
<category><pattern>INQUIRY AGE *</pattern>
<template><srai>RANDOM PICKUP LINE</srai></template>
</category>
<category><pattern>INQUIRY GENDER</pattern>
<template><srai>INQUIRY GENDER <get name="gender"/></srai></template>
</category>
<category><pattern>INQUIRY GENDER UNKNOWN</pattern>
<template>Are you a man or a woman?</template>
</category>
<category><pattern>INQUIRY GENDER *</pattern>

<template><srai>RANDOM PICKUP LINE</srai></template>
</category>
<category><pattern>MAN</pattern><that>ARE YOU A MAN OR A WOMAN</that>
<template><srai>MY GENDER IS MALE</srai></template>
</category>
<category><pattern>WOMAN</pattern><that>ARE YOU A MAN OR A WOMAN</that>
<template><srai>MY GENDER IS FEMALE</srai></template>
</category>
</aiml>

Inquiry.aiml.csv



1	0 *	HOW OLD ARE YOU	*	<think><set name="age"><star/></set></think><srai>NEWLINE</srai>	inquiry.aiml
2	0 INQUIRY LOCATION	*	*	<srai>INQUIRY LOCATION <get name="location"/></srai>	inquiry.aiml
3	0 INQUIRY LOCATION WHERE	*	*	Where are you?	inquiry.aiml
4	0 INQUIRY LOCATION *	*	*	<srai>RANDOM PICKUP LINE</srai>	inquiry.aiml
5	0 INQUIRY NAME	*	*	<srai>INQUIRY NAME <get name="username"/></srai>	inquiry.aiml
6	0 INQUIRY NAME *	*	*	<srai>RANDOM PICKUP LINE</srai>	inquiry.aiml
7	0 INQUIRY NAME FRIEND	*	*	What is your name?	inquiry.aiml
8	0 INQUIRY AGE	*	*	<srai>INQUIRY AGE <get name="age"/></srai>	inquiry.aiml
9	0 INQUIRY AGE HOW MANY	*	*	How old are you?	inquiry.aiml
10	0 INQUIRY AGE *	*	*	<srai>RANDOM PICKUP LINE</srai>	inquiry.aiml
11	0 INQUIRY GENDER	*	*	<srai>INQUIRY GENDER <get name="gender"/></srai>	inquiry.aiml
12	0 INQUIRY GENDER UNKNOWN *	*	*	Are you a man or a woman?	inquiry.aiml
13	0 INQUIRY GENDER *	*	*	<srai>RANDOM PICKUP LINE</srai>	inquiry.aiml
14	0 MAN	ARE YOU A MAN OR A WOMAN	*	<srai>MY GENDER IS MALE</srai>	inquiry.aiml
15	0 WOMAN	ARE YOU A MAN OR A WOMAN	*	<srai>MY GENDER IS FEMALE</srai>	inquiry.aiml

Execution command:

java -cp lib/Ab.jar Main bot = test action = chat trace = false

Output:


Command Prompt - java -cp lib/Abcjai Main list - test action - chat trace= false
Usage: javac <options> <source files>
where possible options include:
@<filename>
-Akey[=value]
--add-modules <module>(,<module>)*
--boot-class-path <path>, -bootclasspath <path>
--class-path <path>, -classpath <path>, -cp <path>
-d <directory>
-deprecation
--enable-preview
-encoding <encoding>
-endorseddirs <dirs>
-extdirs <dirs>
-g
-g:{lines,vars,source}
-g:none
-h <directory>
--help, -help, -?
--help-extra, -X
-implicit:{none,class}
-J<flag>
--limit-modules <module>(,<module>)*
--module <module>(,<module>)*, -m <module>(,<module>)*
--module-path <path>, -p <path>
--module-source-path <module-source-path>
--module-version <version>
-nowarn
-parameters
-proc:{none,only}
-processor <class1>[,<class2>,<class3>...]
--processor-module-path <path>
--processor-path <path>, -processorpath <path>
-profile <profile>
--release <release>

Practical 3: Bayes theorem

Aim: Implement Bayes Theorem using Python

Description: Bayes Theorem is a method of calculating conditional probability. The traditional method of calculating conditional probability (the probability that one event occurs given the occurrence of a different event) is to use the conditional probability formula, calculating the joint probability of event one and event two occurring at the same time, and then dividing it by the probability of event two occurring. However, conditional probability can also be calculated in a slightly different fashion by using Bayes Theorem.

When calculating conditional probability with Bayes theorem, you use the following steps:

- Determine the probability of condition B being true, assuming that condition

A is true.

- Determine the probability of event A being true.
- Multiply the two probabilities together.
- Divide by the probability of event B occurring.

This means that the formula for Bayes Theorem could be expressed like this:

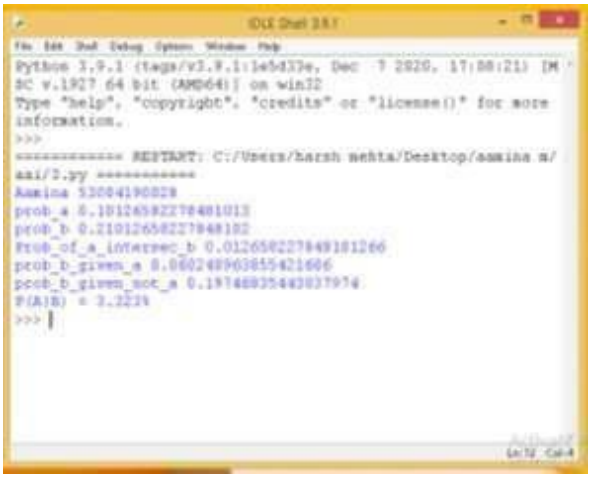
$P(A|B) = P(B|A) * P(A) / P(B)$

Code:

```
print("Aamina 53004190028")
def bayes_theorem(p_a,  p_b_given_a,  p_b_given_not_a):
   not_a = 1 -p_a
        # calculate P(B)
   p_b = p_b_given_a * p_a + p_b_given_not_a * not_a
```

```
        # calculate P(A|B)
    p_a_given_b = (p_b_given_a * p_a) / p_b
    return p_a_given_b
Prob_a = (35 + 5) / (35 + 5 + 277 + 78)
print('prob_a',Prob_a)
prob_b= (78 + 5) / (35 + 5 + 277 + 78)
print('prob_b',prob_b)
Prob_of_a_intersec_b = 5 / (35 + 5 + 277 + 78)
print('Prob_of_a_intersec_b',Prob_of_a_intersec_b)
prob_b_given_a=Prob_of_a_intersec_b/prob_b
print("prob_b_given_a",prob_b_given_a)
prob_b_given_not_a=prob_b-Prob_of_a_intersec_b
print("prob_b_given_not_a",prob_b_given_not_a)
result = bayes_theorem(Prob_a, prob_b_given_a, prob_b_given_not_a)
print('P(A|B) = %.3f%%' % (result * 100))
```

Output:



PRACTICAL 4

AIM: To implement conditional probability and joint probability using python

DESCRIPTION:

Conditional probability is the probability of one event occurring with some relationship to one or more other events. For example:Event A is that it is raining outside, and it has a 0.3 (30%) chance of raining today.Event B is that you will need to go outside, and that has a probability of 0.5 (50%).A conditional probability would look at these two events in relationship with one another, such as the probability that it is both raining and you will need to go outside.

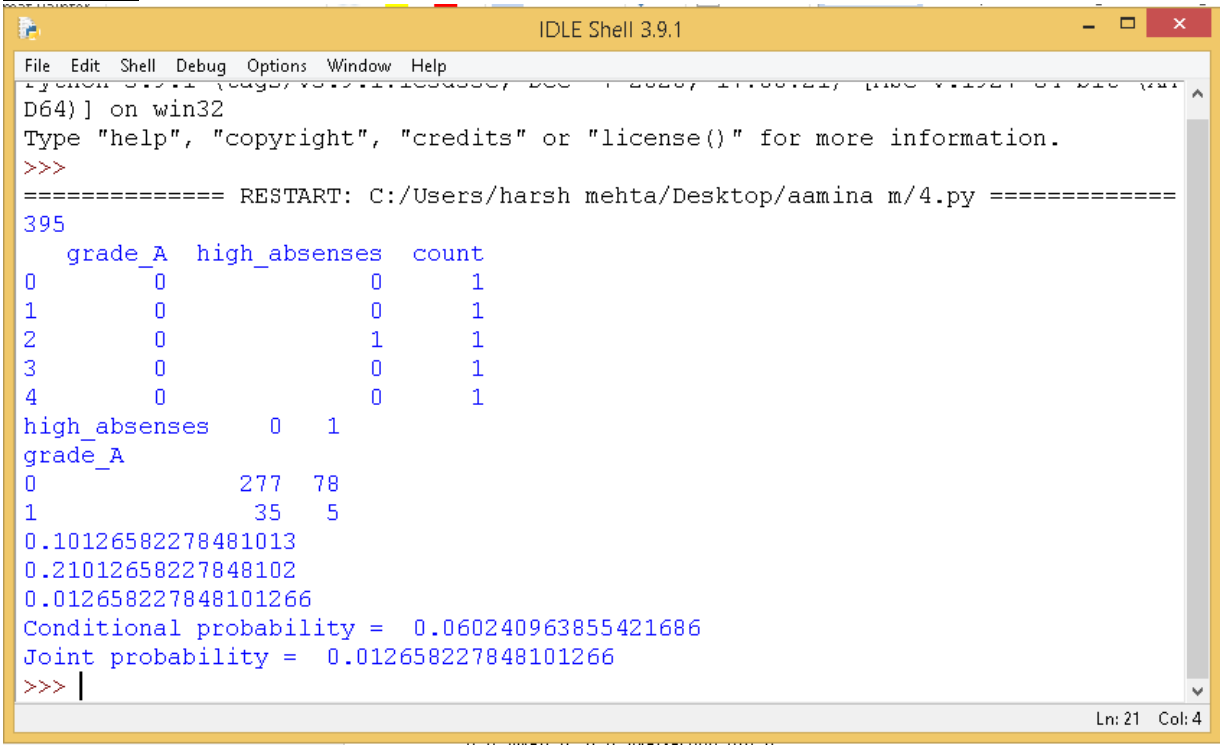
Joint probability is a statistical measure that calculates the likelihood of two events occurring together and at the same point in time. Joint probability is the probability of event Y occurring at the same time that event X occurs.

CODE:

```
import pandas as pd
import numpy as np
df = pd.read_csv('student-mat.csv')
df.head(3)
print(len(df))
df['grade_A'] = np.where(df['G3']*5 >= 80, 1, 0)
#Make another boolean column called high_absenses with a value of 1 if a
student missed 10 or more classes.
df['high_absenses'] = np.where(df['absences'] >= 10, 1, 0)
#Add one more column to make building a pivot table easier.
df['count'] = 1
#And drop all columns we don't care about.
df = df[['grade_A','high_absenses','count']]
print(df.head())
#Now we'll create a pivot table from this.
pt = pd.pivot_table(
   df,
   values='count',
   index=['grade_A'],
```

```
    columns=['high_absenses'],
    aggfunc=np.size,
    fill_value=0
)
print(pt)
#P(A) = (35 + 5) / (35 + 5 + 277 + 78) = 0.10126582278481013
#P(B) = (78 + 5) / (35 + 5 + 277 + 78) = 0.21012658227848102
#P(A ∩ B) = 5 / (35 + 5 + 277 + 78) = 0.012658227848101266
#And per the formula, P(A|B) = P(A ∩ B) / P(B), put it together.
#P(A|B) = 0.012658227848101266/ 0.21012658227848102= 0.06
p_A = (pt[0][1]+pt[1][1])/(pt[0][0]+pt[0][1]+pt[1][0]+pt[1][1])
print(p_A)
p_B = (pt[1][0]+pt[1][1])/(pt[0][0]+pt[0][1]+pt[1][0]+pt[1][1])
print(p_B)
p_A_intersection_B = (pt[1][1])/(pt[0][0]+pt[0][1]+pt[1][0]+pt[1][1])
print(p_A_intersection_B)
p_A_given_B = p_A_intersection_B/p_B
print('Conditional probability = ',p_A_given_B)
print('Joint probability = ',p_A_intersection_B)
```

OUTPUT:



Practical 6: Fuzzy based application

Aim: Design a Fuzzy based application using Python/R.

Discription: Let's create a fuzzy control system which models how you might choose to tip at a restaurant. When tipping, you consider the service and food quality, rated between 0 and 10. You use this to leave a tip of between 0 and 25%. We would formulate this problem as:

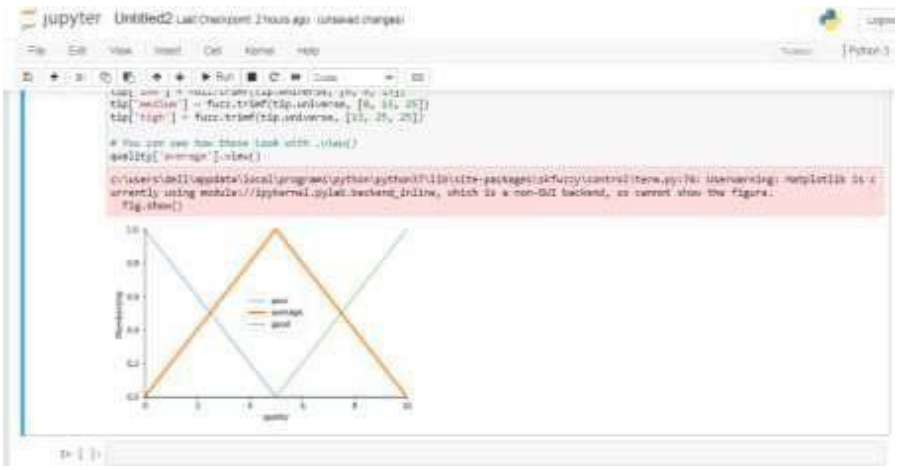
- **Antecedents (Inputs)**
  - **service**
    - Universe (i.e., crisp value range): How good was the service of the wait staff, on a scale of 0 to 10?
    - Fuzzy set (i.e., fuzzy value range): poor, acceptable, amazing
  - **food quality**
    - Universe: How tasty was the food, on a scale of 0 to 10?
    - Fuzzy set: bad, decent, great
- **Consequents (Outputs)**
  - **tip**
    - Universe: How much should we tip, on a scale of 0% to 25%
    - Fuzzy set: low, medium, high
- **Rules**
  - IF the *service* was good *or* the *food quality* was good, THEN the tip will be high.
  - IF the *service* was average, THEN the tip will be medium.
  - IF the *service* was poor *and* the *food quality* was poor THEN the tip will be low.
- **Usage**
  - **If I tell this controller that I rated:**
    - the service as 9.8, and
    - the quality as 6.5,
  - **it would recommend I leave:**
    - a 20.2% tip.

Code:

```
# -*- coding: utf-8 -*-
"""
Created on Thu Jan 21 15:02:50 2021
```

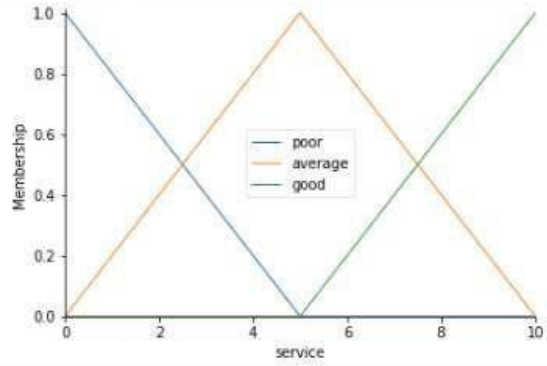
```
@author: aamina
"""
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
# New Antecedent/Consequent objects hold universe variables and membership
# functions
quality = ctrl.Antecedent(np.arange(0, 11, 1), 'quality')
service = ctrl.Antecedent(np.arange(0, 11, 1), 'service')
tip = ctrl.Consequent(np.arange(0, 26, 1), 'tip')
# Auto-membership function population is possible with .automf(3, 5, or 7)
quality.automf(3)
service.automf(3)
# Custom membership functions can be built interactively with a familiar,
# Pythonic API
tip['low'] = fuzz.trimf(tip.universe, [0, 0, 13])
tip['medium'] = fuzz.trimf(tip.universe, [0, 13, 25])
tip['high'] = fuzz.trimf(tip.universe, [13, 25, 25])
# You can see how these look with .view()
quality['average'].view()
```

Output:

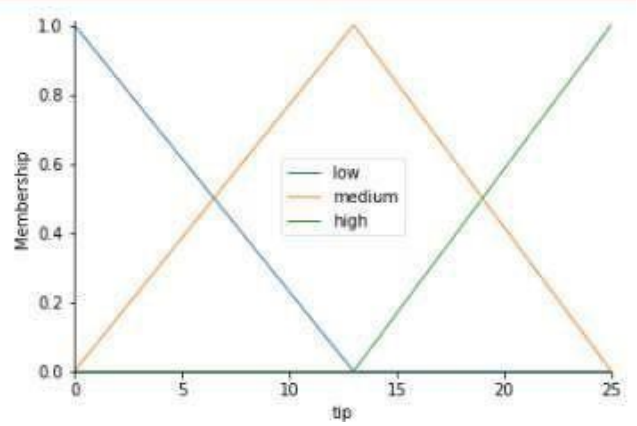


service.view()

```
fig.show()
```



tip.view()



Now, to make these triangles useful, we define the *fuzzy relationship* between input and output variables. For the purposes of our example, consider three simple rules:

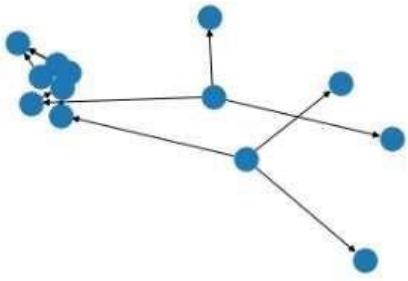
1. If the food is poor OR the service is poor, then the tip will be low
2. If the service is average, then the tip will be medium
3. If the food is good OR the service is good, then the tip will be high.

Most people would agree on these rules, but the rules are fuzzy. Mapping the imprecise rules into a defined, actionable tip is a challenge. This is the kind of task at which fuzzy logic excels.

```
rule1 = ctrl.Rule(quality['poor'] | service['poor'], tip['low'])
rule2 = ctrl.Rule(service['average'], tip['medium'])
rule3 = ctrl.Rule(service['good'] | quality['good'], tip['high'])
```

rule1.view()

```
Out[12]: (<Figure size 432x288 with 1 Axes>, <AxesSubplot:>)
```



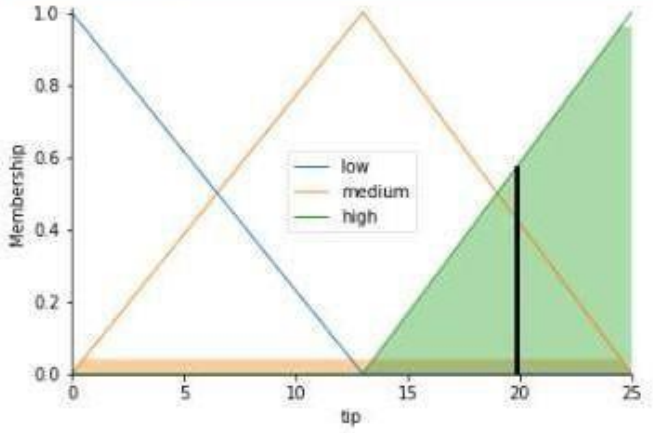
tipping_ctrl=ctrl.ControlSystem([rule1,rule2,rule3])

tipping=ctrl.ControlSystemSimulation(tipping_ctrl)

Suppose we rated the quality 6.5 out of 10 and the service 9.8 of 10.

```
# Pass inputs to the ControlSystem using Antecedent labels with Pythonic API
# Note: if you like passing many inputs all at once, use .inputs(dict_of_data)
tipping.input['quality'] = 6.5
tipping.input['service'] = 9.8

# Crunch the numbers
tipping.compute()
print (tipping.output['tip'])
tip.view(sim=tipping)
```

```
19.847607361963192

c:\users\dell\appdata\local\programs\python\python3
otlib is currently using module://ipykernel.pylab.ba
  fig.show()
```



The resulting suggested tip is **19.84%**

Practical 7

Supervised learning model

Aim: Write an application to simulate supervised and un-supervised learning model.

Description:

Supervised learning model

DecisionTreeClassifier:-

A decision tree is a flowchart-like tree structure where an internal node represents feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome. The topmost node in a decision tree is known as the root node. It learns to partition on the basis of the attribute value. It partitions the tree in recursively manner call recursive partitioning. This flowchart-like structure helps you in decision making. It's visualization like a flowchart diagram which easily mimics the human level thinking. That is why decision trees are easy to understand and interpret.

Import the pacakges. Let's first load the dataset using pandas' read CSV function. You can download the data here .Here, you need to divide given columns into two types of variables dependent(or target variable) and independent variable(or feature variables).

To understand model performance, dividing the dataset into a training set and a test set is a good strategy.

Let's split the dataset by using function train_test_split(). You need to pass 3 parameters features, target, and test_set size.

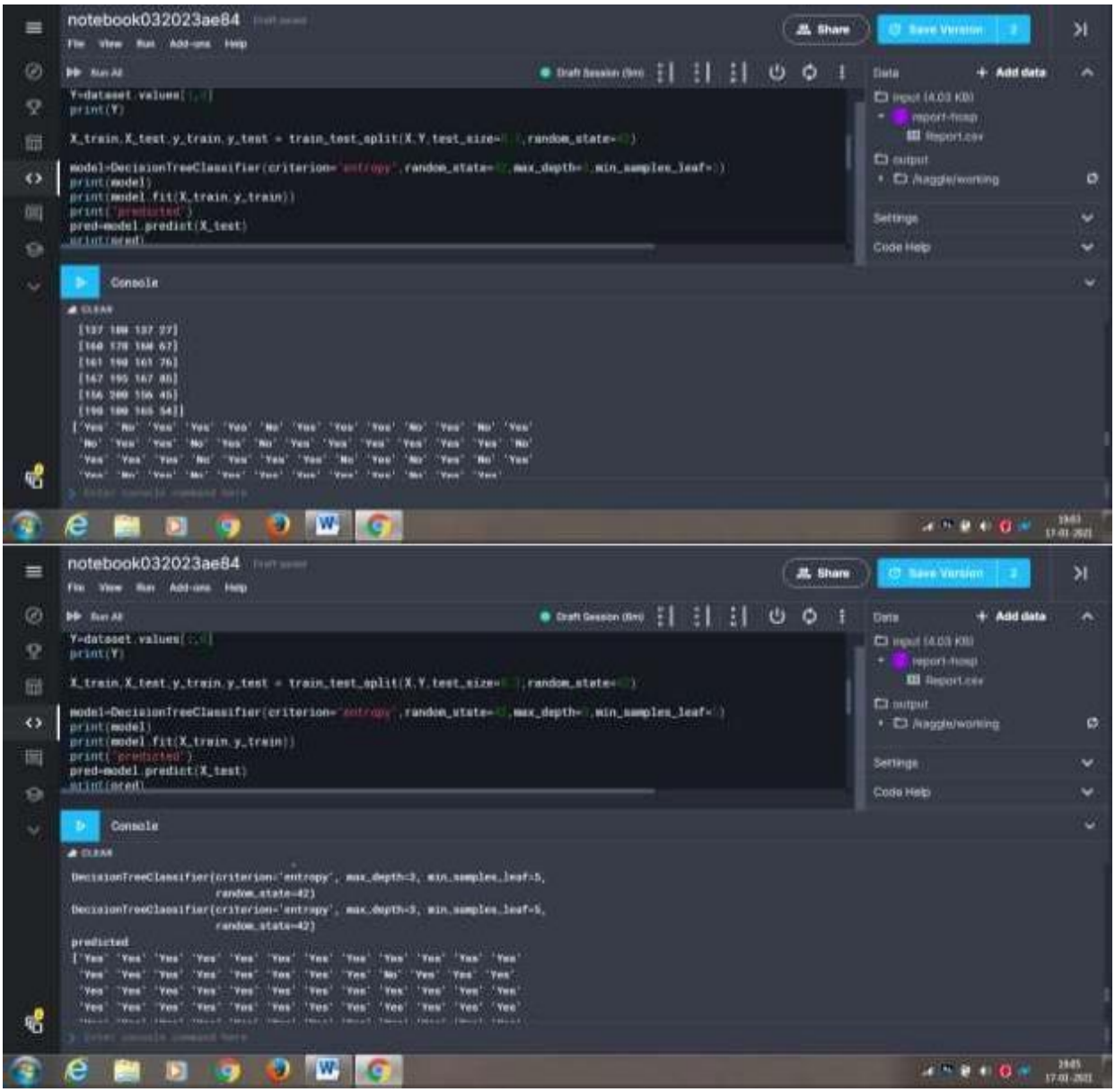
Let's estimate, how accurately the classifier or model can predict the type of cultivars.Accuracy can be computed by comparing actual test set values and predicted values.

Code:

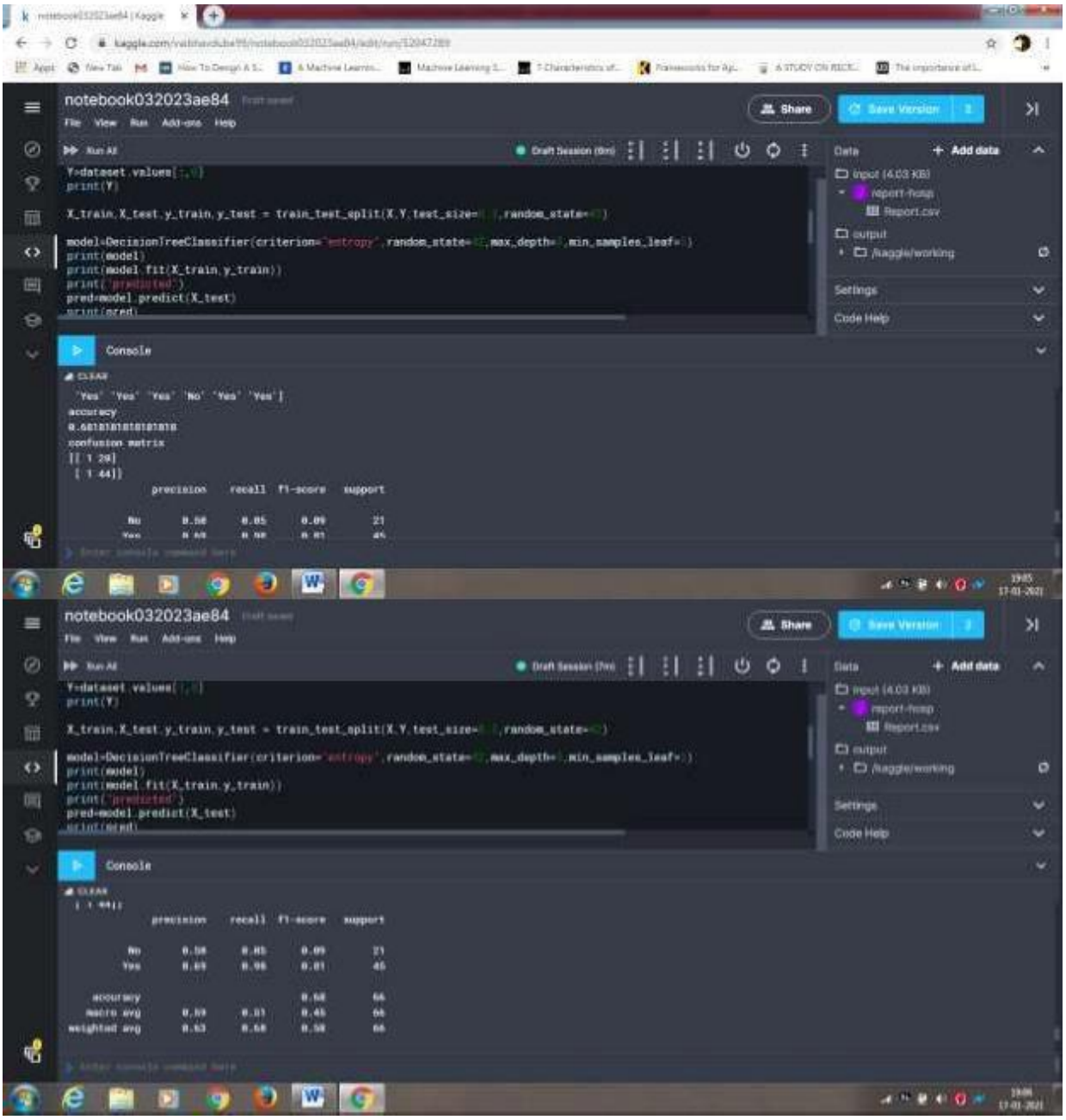
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from sklearn import metrics
from sklearn.model_selection importtrain_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from matplotlib import pyplot as plt
```

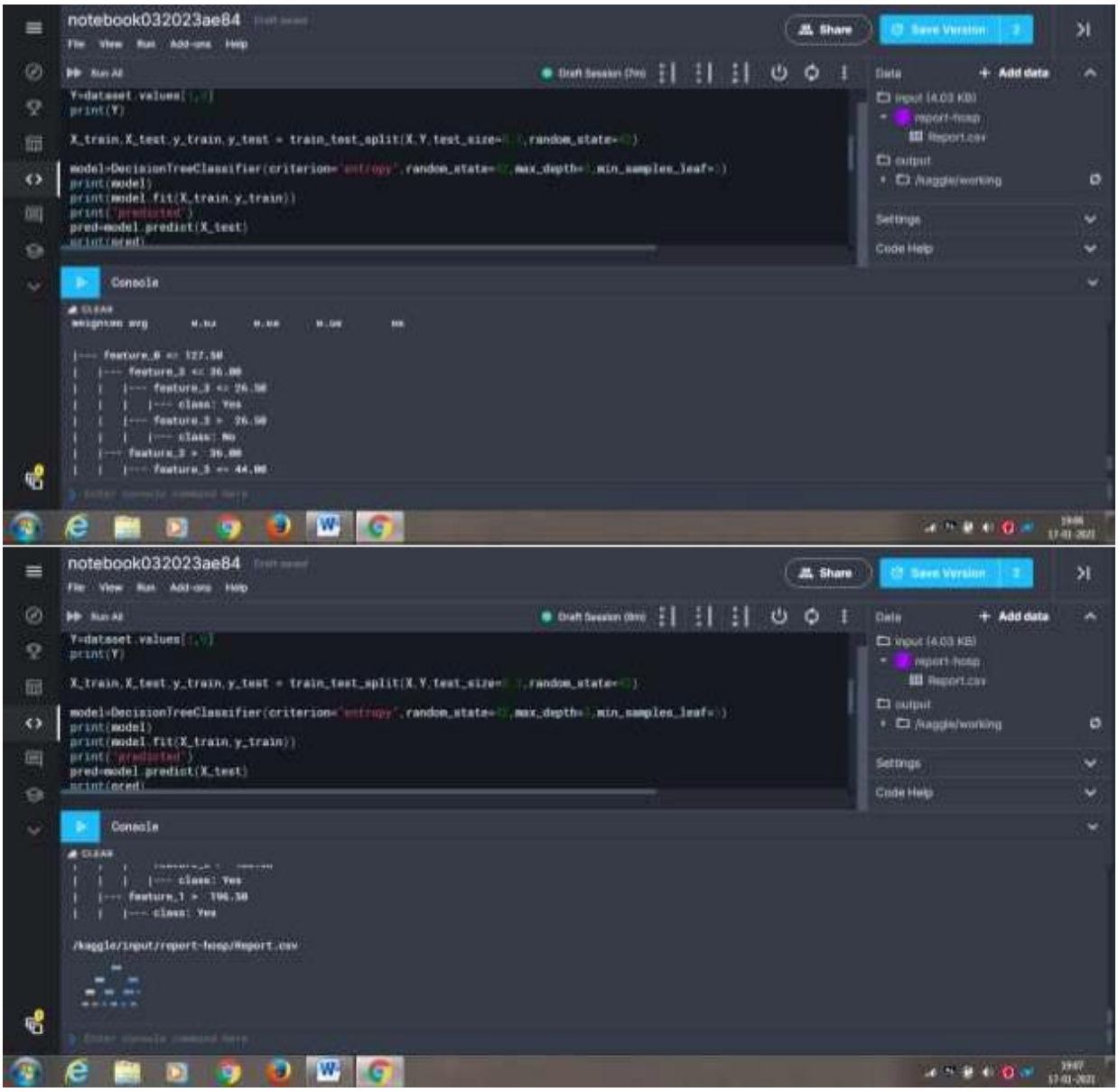
```
dataset=pd.read_csv("../input/report-hosp/Report.csv")
print(dataset)
size=len(dataset)
print(size)
print(dataset.head())
X=dataset.values[:,1:5]
print(X)
Y=dataset.values[:,0]
print(Y)
X_train,X_test,y_train,y_test =
train_test_split(X,Y,test_size=0.3,random_state=42)
model=DecisionTreeClassifier(criterion="entropy",random_state=42,max_depth=
3,min_samples_leaf=5)
print(model)
print(model.fit(X_train,y_train))
print("predicted")
pred=model.predict(X_test)
print(pred)
accuracy=metrics.accuracy_score(y_test,pred)
print("accuracy")
print(accuracy)
print("confusion matrix")
print(metrics.confusion_matrix(y_test,pred))
print(metrics.classification_report(y_test,pred))
text_format=tree.export_text(model)
print(text_format)
fig=plt.figure(figsize=(25,20))
tree.plot_tree(model, feature_names=X, class_names=Y,filled=True)
```

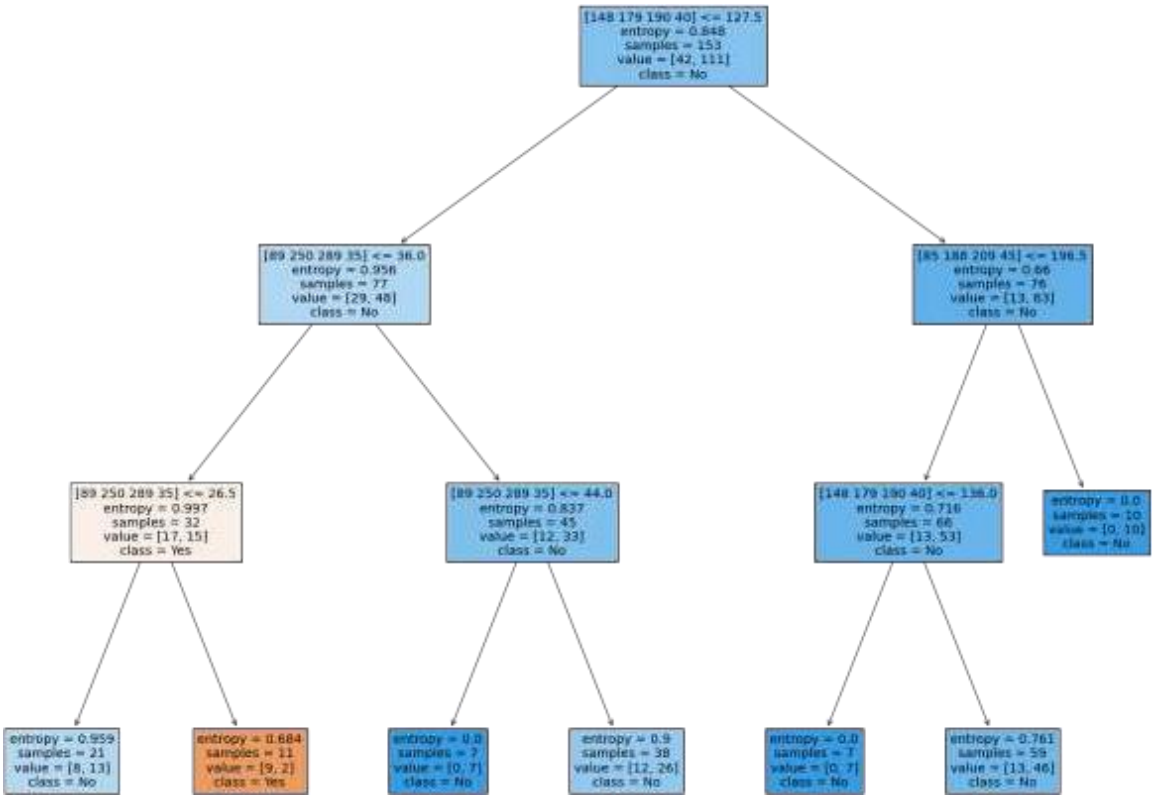
Output:


notebook032023ae84
Share
Save Version
Add data
Console
['Yes' 'No' 'Yes' 'Yes' 'Yes' 'No' 'Yes' 'Yes' 'Yes' 'No' 'Yes' 'No' 'Yes'
'No' 'Yes' 'Yes' 'No' 'Yes' 'No' 'Yes' 'Yes' 'Yes' 'Yes' 'Yes' 'Yes' 'No'
'Yes' 'Yes' 'Yes' 'No' 'Yes' 'Yes' 'Yes' 'No' 'Yes' 'No' 'Yes' 'No' 'Yes'
'Yes' 'No' 'Yes' 'No' 'Yes' 'Yes' 'Yes' 'Yes' 'Yes' 'No' 'Yes' 'Yes']

notebook032023ae84
Share
Save Version
Add data
Console
DecisionTreeClassifier(criterion='entropy', max_depth=3, min_samples_leaf=5,
random_state=42)
DecisionTreeClassifier(criterion='entropy', max_depth=3, min_samples_leaf=5,
random_state=42)
predicted
['Yes' 'Yes' 'Yes' 'Yes' 'Yes' 'Yes' 'Yes' 'Yes' 'Yes' 'Yes' 'Yes' 'Yes'
'Yes' 'Yes' 'Yes' 'Yes' 'Yes' 'Yes' 'Yes' 'Yes' 'No' 'Yes' 'Yes' 'Yes'
'Yes' 'Yes' 'Yes' 'Yes' 'Yes' 'Yes' 'Yes' 'Yes' 'Yes' 'Yes' 'Yes' 'Yes'
'Yes' 'Yes' 'Yes' 'Yes' 'Yes' 'Yes' 'Yes' 'Yes' 'Yes' 'Yes' 'Yes' 'Yes'


notebook032023ae84
Share
Save Version
Run All
Draft Session (0m)
Data
Add data
Console
'Yes' 'Yes' 'Yes' 'No' 'Yes' 'Yes']
accuracy
confusion matrix
[[1 20]
[1 44]]
precision recall f1-score support
No 0.50 0.05 0.09 21
Yes 0.69 0.98 0.81 45
notebook032023ae84
Share
Save Version
Run All
Draft Session (7m)
Data
Add data
Console
precision recall f1-score support
No 0.50 0.05 0.09 21
Yes 0.69 0.98 0.81 45
accuracy 0.68 66
macro avg 0.59 0.51 0.45 66
weighted avg 0.63 0.68 0.58 66


notebook032023ae84
Share
Save Version
Add data
Console
|--- feature_0 <= 127.50
|--- feature_3 <= 36.00
|--- feature_3 <= 26.50
|--- class: Yes
|--- feature_3 > 26.50
|--- class: No
|--- feature_3 > 36.00
|--- feature_3 <= 44.00
notebook032023ae84
Share
Save Version
Add data
Console
|--- class: Yes
|--- feature_1 > 196.50
|--- class: Yes
/kaggle/input/report-heap/Report.csv


[148 179 190 40] <= 127.5
entropy = 0.848
samples = 153
value = [42, 111]
class = No
[89 250 289 35] <= 36.0
entropy = 0.956
samples = 77
value = [29, 48]
class = No
[85 188 209 45] <= 196.5
entropy = 0.66
samples = 76
value = [13, 63]
class = No
[89 250 289 35] <= 26.5
entropy = 0.997
samples = 32
value = [17, 15]
class = Yes
[89 250 289 35] <= 44.0
entropy = 0.837
samples = 45
value = [12, 33]
class = No
[148 179 190 40] <= 136.0
entropy = 0.716
samples = 66
value = [13, 53]
class = No
entropy = 0.0
samples = 10
value = [0, 10]
class = No
entropy = 0.959
samples = 21
value = [8, 13]
class = No
entropy = 0.684
samples = 11
value = [9, 2]
class = Yes
entropy = 0.0
samples = 7
value = [0, 7]
class = No
entropy = 0.9
samples = 38
value = [12, 26]
class = No
entropy = 0.0
samples = 7
value = [0, 7]
class = No
entropy = 0.761
samples = 59
value = [13, 46]
class = No

PRACTICAL 8

AIM: To implement clustering Algorithm (K-means Algorithm)

DESCRIPTION: Kmeans algorithm is an iterative algorithm that tries to partition the dataset into *K*pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to **only one group**. It tries to make the intra-cluster data points as similar as possible while also keeping the clusters as different (far) as possible. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum. The less variation we have within clusters, the more homogeneous (similar) the data points are within the same cluster.

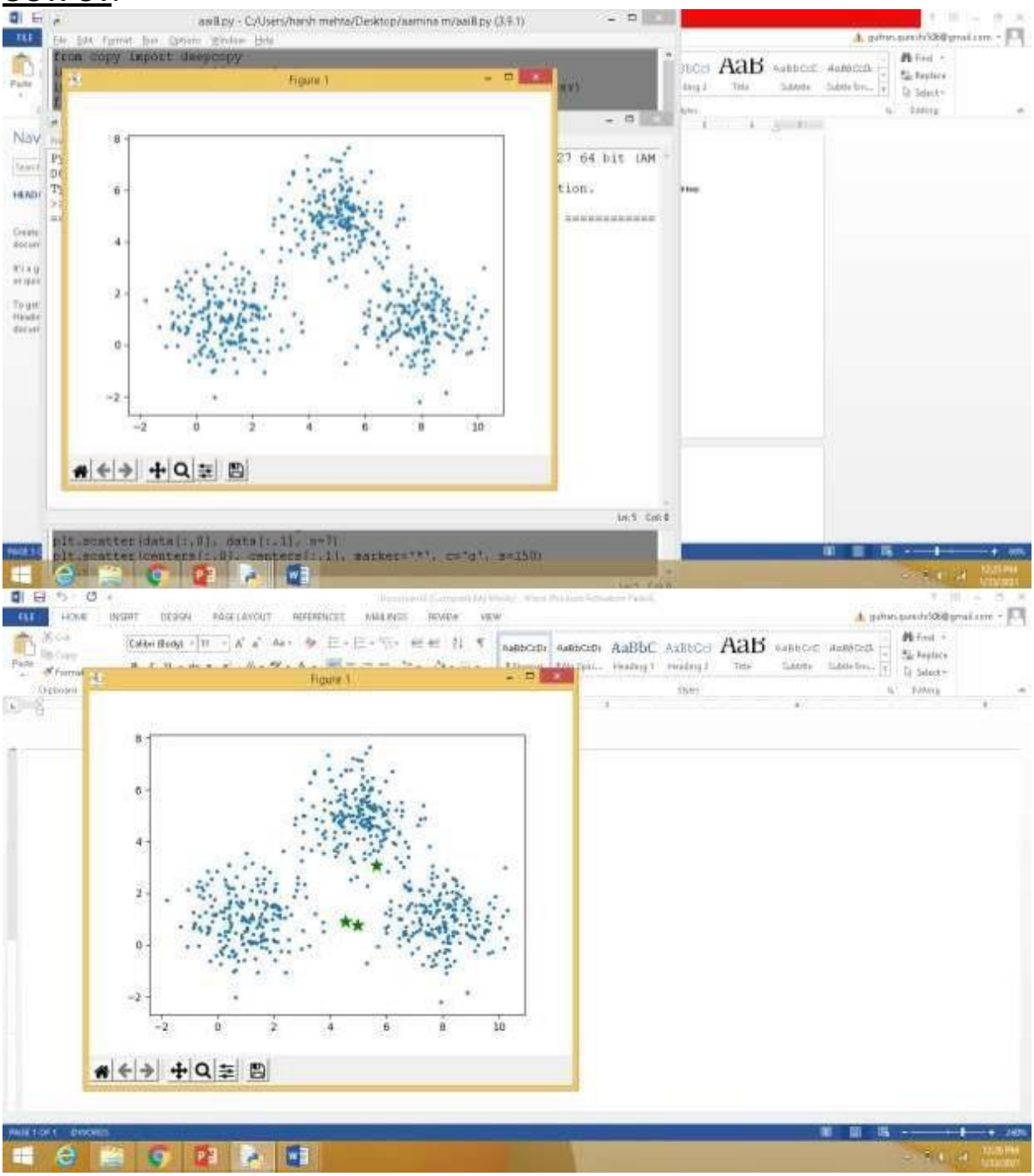
CODE:

```
from copy import deepcopy
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from matplotlib import pyplot as plt

# Set three centers, the model should predict similar results
center_1 = np.array([1,1])
center_2=np.array([5,5])
center_3 = np.array([8,1])

# Generate random data and center it to the three centers
data_1 = np.random.randn(200, 2) + center_1
data_2 = np.random.randn(200,2) + center_2
data_3 = np.random.randn(200,2) + center_3
data = np.concatenate((data_1, data_2, data_3), axis = 0)

plt.scatter(data[:,0],  data[:,1],  s=7)
plt.show()

#create K-mean algorithm
# Number of clusters
k = 3
```

```
# Number of training data
n = data.shape[0]
# Number of features in the data
c = data.shape[1]

mean = np.mean(data, axis = 0)
std = np.std(data, axis = 0)
centers = np.random.randn(k,c)*std + mean

# Plot the data and the centers generated as random
plt.scatter(data[:,0], data[:,1], s=7)
plt.scatter(centers[:,0], centers[:,1], marker='*', c='g', s=150)
plt.show()
centers_old = np.zeros(centers.shape) # to store old centers
centers_new = deepcopy(centers) # Store new centers

data.shape
clusters = np.zeros(n)
distances = np.zeros((n,k))

error = np.linalg.norm(centers_new - centers_old)

# When, after an update, the estimate of that center stays the same, exit loop
while error != 0:
    # Measure the distance to every center
    for i in range(k):
        distances[:,i] = np.linalg.norm(data - centers[i], axis=1)
    # Assign all training data to closest center
    clusters = np.argmin(distances, axis = 1)
    centers_old = deepcopy(centers_new)
    for i in range(k):
        centers_new[i] = np.mean(data[clusters == i], axis=0)
    error = np.linalg.norm(centers_new - centers_old)
centers_new

# Plot the data and the centers generated as random
plt.scatter(data[:,0], data[:,1], s=7)
```

```
plt.scatter(centers_new[:,0],centers_new[:,1],marker='*',c='g',s=150)
plt.show()
```

OUTPUT:



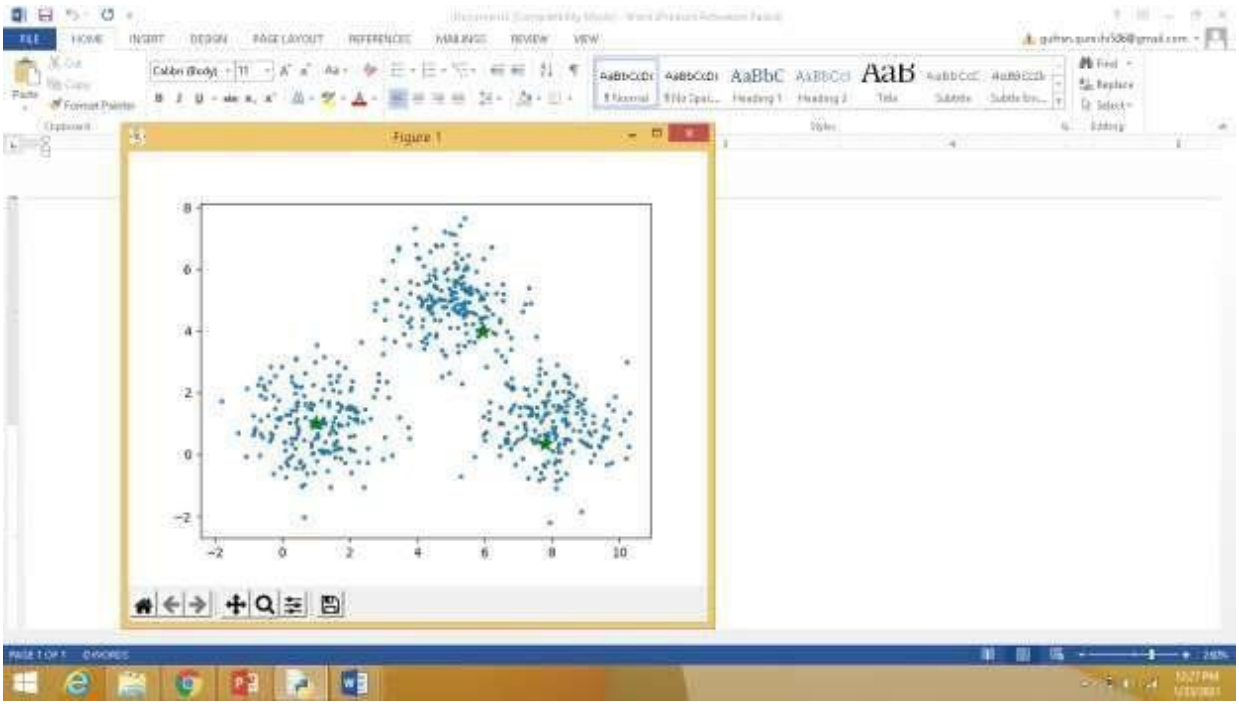


Figure 1
8
6
4
2
0
−2
−2
0
2
4
6
8
10