Rebecca Dias Rollno-19 PID-182027 TECMPNA

Experiment 3:

Implementation of Classification algorithm

Aim: To implement any one of the classifiers Naïve Bayes using any languages like Python

S/w Requirement: Python

Theory:

- **Explain Classification**
- Classification means arranging the mass of data into different classes or groups on the basis of their similarities and resemblances.
- All similar items of data are put in one class and all dissimilar items of data are put in different classes.
- Statistical data is classified according to its characteristics.
- For example, if we have collected data regarding the number of students admitted to a university in a year, the students can be classified on the basis of sex. In this case, all male students will be put in one class and all female students will be put in another class.
- The students can also be classified on the basis of age, marks, marital status, height, etc.
- The set of characteristics we choose for the classification of the data depends upon the objective of the study.
- For example, if we want to study the religions mix of the students, we classify the students on the basis of religion.

- **Explain Bayes theorem**

Bayes' theorem is used to determine the conditional probability of events. Essentially, the Bayes' theorem describes the probability of an event based on prior knowledge of the conditions that might be relevant to the event.

The Bayes' theorem is expressed in the following formula:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Where:

P(A|B) – the probability of event A occurring, given event B has occurred

P(B|A) – the probability of event B occurring, given event A has occurred

P(A) – the probability of event A

P(B) – the probability of event B

- **Explain Naïve Bayes Algorithm (pseudo code) with example**

Psuedo code:

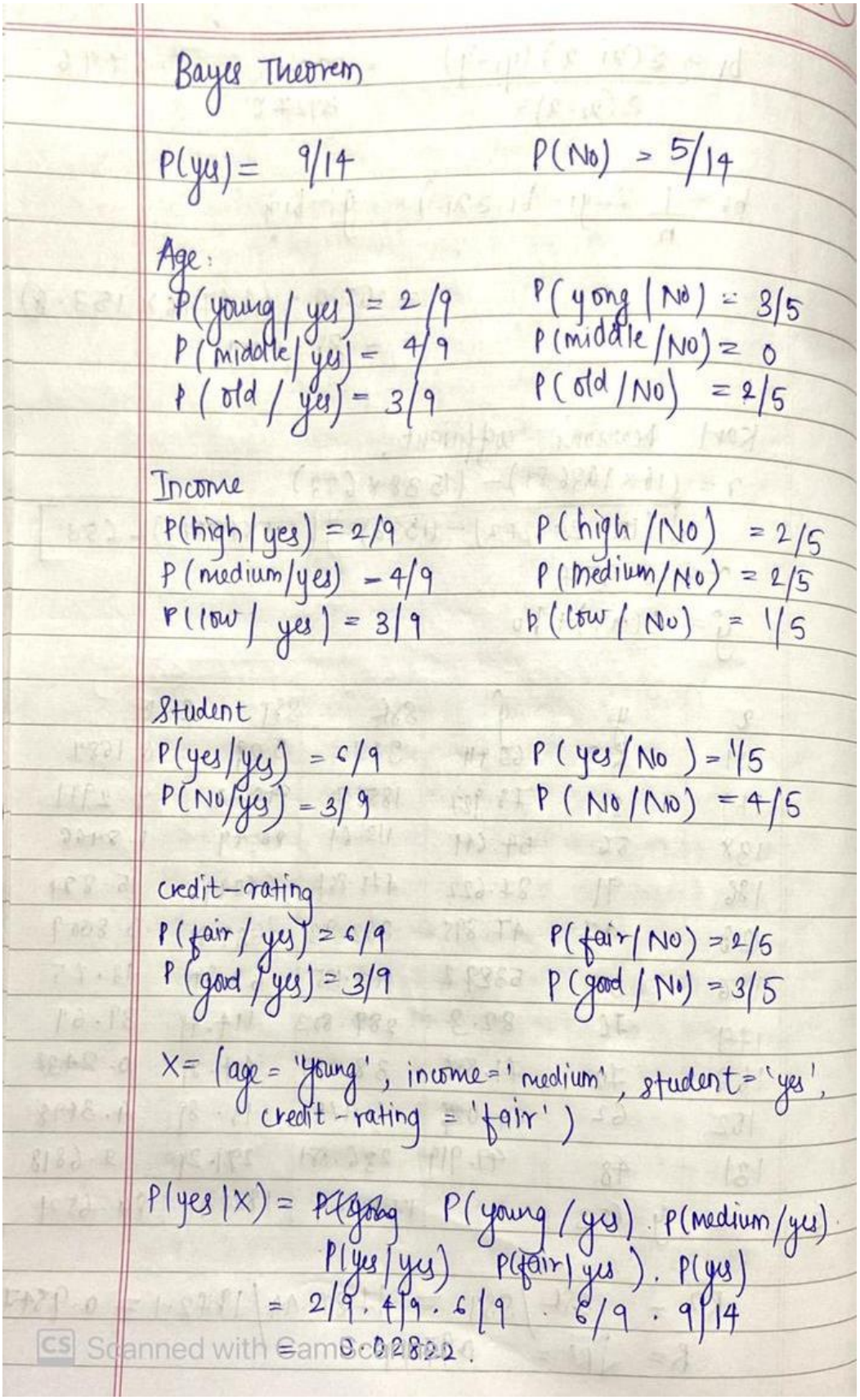
1. Read the training dataset T.
2. Calculate the Probabilities of all possible events
3. Read the new tuple to test the classifier
4. Calculate the likelihood for new tuple.
5. Get the greatest likelihood.

Example:

Apply the Naive Bayes classifier algorithm for buys computer classification classify the tuple X=(age=''young'', income=''medium'', student=''yes'' and credit-rating=''fair'')

Id	Age	Income	Student	Credit-rating	buys computer
1	young	high	no	fair	no
2	young	high	no	good	no
3	middle	high	no	fair	yes
4	old	medium	no	fair	yes
5	old	low	yes	fair	yes
6	old	low	yes	good	no
7	middle	low	yes	good	yes
8	young	medium	no	fair	no
9	young	low	yes	fair	yes
10	old	medium	yes	fair	yes
11	young	medium	yes	good	yes
12	middle	medium	no	good	yes
13	middle	high	yes	fair	yes
14	old	medium	no	good	no

Solution:



Bayes Theorem

P(yes) = 9/14 P(No) = 5/14

Age:

P(young / yes) = 2/9 P(young / No) = 3/5
P(middle / yes) = 4/9 P(middle / No) = 0
P(old / yes) = 3/9 P(old / No) = 2/5

Income

P(high / yes) = 2/9 P(high / No) = 2/5
P(medium / yes) = 4/9 P(medium / No) = 2/5
P(low / yes) = 3/9 P(low / No) = 1/5

Student

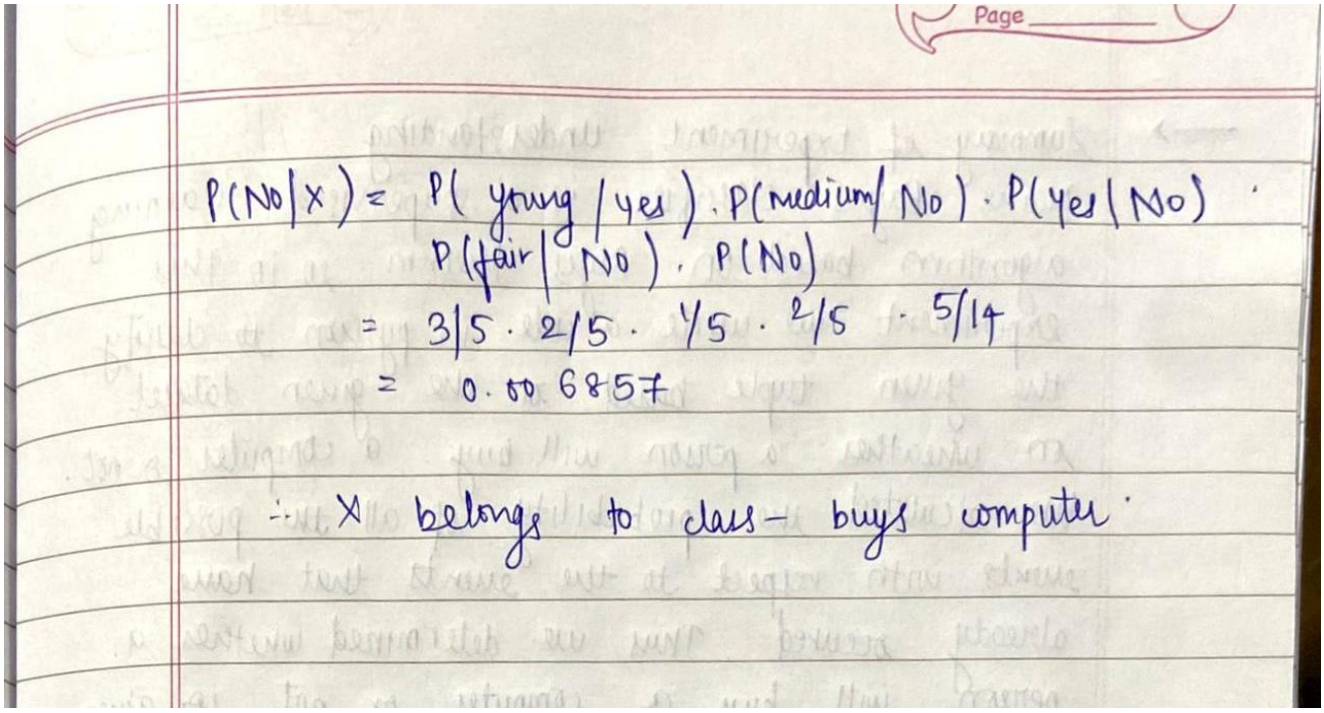
P(yes / yes) = 6/9 P(yes / No) = 1/5
P(No / yes) = 3/9 P(No / No) = 4/5

credit-rating

P(fair / yes) = 6/9 P(fair / No) = 2/5
P(good / yes) = 3/9 P(good / No) = 3/5

X = (age = 'young', income = 'medium', student = 'yes', credit-rating = 'fair')

P(yes | X) = ~~P(young~~ P(young / yes) . P(medium / yes) P(yes / yes) P(fair / yes) . P(yes)
= 2/9 . 4/9 . 6/9 . 6/9 . 9/14
= 0.02822.



Implementation:

- Training set is given in the lab manual and new tuple to test the classifier is also mentioned.
- Write code to classify the new tuple accordingly using the Bayes theorem.

Dataset:

Apply the Naive Bayes classifier algorithm for buys computer classification a classify the tuple X=(age=''young'', income=''medium'', student=''yes'' and credit-rating=''fair'')

Id	Age	Income	Student	Credit-rating	buys computer
1	young	high	no	fair	no
2	young	high	no	good	no
3	middle	high	no	fair	yes
4	old	medium	no	fair	yes
5	old	low	yes	fair	yes
6	old	low	yes	good	no
7	middle	low	yes	good	yes
8	young	medium	no	fair	no
9	young	low	yes	fair	yes
10	old	medium	yes	fair	yes
11	young	medium	yes	good	yes
12	middle	medium	no	good	yes
13	middle	high	yes	fair	yes
14	old	medium	no	good	no

Code:

```
def calc_count(x, y):
    count = 0
    for element in d:
        if(x in element and y in element):
            count += 1
    return count
d = [
    ["young", "high", "Sno", "fair", "no"],
    ["young", "high", "Sno", "good", "no"],
    ["middle", "high", "Sno", "fair", "yes"],
    ["old", "medium", "Sno", "fair", "yes"],
    ["old", "low", "Syes", "fair", "yes"],
    ["old", "low", "Syes", "good", "no"],
    ["middle", "low", "Syes", "good", "yes"],
    ["young", "medium", 'Sno', "fair", "no"],
    ["young", "low", "Syes", "fair", "yes"],
    ["old", "medium", "Syes", "fair", "yes"],
    ["young", "medium", 'Syes', "good", "yes"],
    ["middle", "medium", 'Sno', "good", "yes"],
    ["middle", "high", "Syes", "fair", "yes"],
    ["old", "medium", "Sno", "good", "no"]
]
y = 0
n = 0
for i in range(len(d)):
    if(d[i][4] == 'yes'):
        y += 1
    else:
        n += 1
py = y/len(d)
pn = n/len(d)
print("P(Yes)="+str(py)+"\nP(No)="+str(pn))
print("----------------------------------------------------")
p_ay = calc_count("young", "yes")
print("1.AGE:")
print("P(ageyoung|Yes)="+str(round(p_ay/y, 2)))
p_atfy = calc_count("middle", "yes")
print("P(age middle|Yes)="+str(round(p_atfy/y, 2)))
p_agfy = calc_count("old", "yes")
print("P(ageold|Yes)="+str(round(p_agfy/y, 2)))
p_an = calc_count("young", "no")
print("P(ageyoung|No)="+str(round(p_an/n, 2)))
p_atfn = calc_count("middle", 'no')
print("P(age middle|No)="+str(round(p_atfn/n, 2)))
p_agfn = calc_count("old", "no")
print("P(ageold|No)="+str(round(p_agfn/n, 2)))
print("----------------------------------------------------")
print("2.INCOME:")
p_ihy = calc_count("high", "yes")
print("P(income high|Yes)="+str(round(p_ihy/y, 2)))
p_imy = calc_count("medium", "yes")
print("P(income medium|Yes)="+str(round(p_imy/y, 2)))
p_ily = calc_count("low", "yes")
print("P(income low|Yes)="+str(round(p_ily/y, 2)))
p_ihn = calc_count("high", "no")
print("P(income high|no)="+str(round(p_ihn/n, 2)))
p_imn = calc_count("medium", "no")
print("P(income medium|no)="+str(round(p_imn/n, 2)))
```

```
p_iln = calc_count("low", 'no')
print("P(income low|no)="+str(round(p_iln/n, 2)))
print("----------------------------------------------------")
print("3.STUDENT:")
p_syy = calc_count("Syes", "yes")
print("P(student yes|yes)="+str(round(p_syy/y, 2)))
p_sny = calc_count("Sno", "yes")
print("P(student no|yes)="+str(round(p_sny/y, 2)))
p_syn = calc_count("Syes", "no")
print("P(student yes|no)="+str(round(p_syn/n, 2)))
p_snn = calc_count("Sno", 'no')
print("P(student no|no)="+str(round(p_snn/n, 2)))
print("----------------------------------------------------")
print("4.CREDIT RATING:")
p_crfy = calc_count("fair", "yes")
print("P(credit_rating fair|Yes)="+str(round(p_crfy/y, 2)))
p_crey = calc_count("good", "yes")
print("P(credit_rating good|Yes)="+str(round(p_crey/y, 2)))
p_crfn = calc_count("fair", "no")
print("P(credit_rating fair|no)="+str(round(p_crfn/n, 2)))
p_cren = calc_count("good", "no")
print("P(credit_rating good|no)="+str(round(p_cren/n, 2)))

data = "young,medium,Syes,fair"
newData = data.split(",")

print()
print("DATA SAMPLE")
print(newData)
p_newyes = (p_ay)*(p_imy)*(p_syy)*(p_crfy)*(py)
p_newno = (p_an)*(p_imn)*(p_syn)*(p_crfn)*(pn)
print('The student will ', end="")
if(p_newyes > p_newno):
    print('buy computer')
else:
    print('not buy computer')
```

Output:

```
C:\Users\toshiba\Desktop>python index.py
P(yes)= 9 / 14 = 0.6428571428571429
P(No)= 5 / 14 = 0.35714285714285715

Age:
P( middle /yes)= 4 / 9   P( middle /no)= 0 / 5
P( young /yes)= 2 / 9    P( young /no)= 3 / 5
P( old /yes)= 3 / 9      P( old /no)= 2 / 5
----------------------------------------------------------------
Income:
P( medium /yes)= 4 / 9   P( medium /no)= 2 / 5
P( low /yes)= 3 / 9      P( low /no)= 1 / 5
P( high /yes)= 2 / 9     P( high /no)= 2 / 5
----------------------------------------------------------------
Student:
P( yes /yes)= 6 / 9      P( yes /no)= 1 / 5
P( no /yes)= 3 / 9       P( no /no)= 4 / 5
----------------------------------------------------------------
Credit_rating:
P( fair /yes)= 6 / 9     P( fair /no)= 2 / 5
P( good /yes)= 3 / 9     P( good /no)= 3 / 5
----------------------------------------------------------------

Enter the values for tuple X = young medium yes fair

P(Yes|X) = 0.028218694885361547
P(No|X) = 0.006857142857142858

X is classified in class yes (buys computer) as P(Yes|X)>P(No|X)
```

Conclusion:

→ Summary of Experiment Understanding

Naive Bayes classifier is a supervised learning algorithm based on Bayes theorem so in this experiment we write a code in python to classify the given tuple based on the given dataset on whether a person will buy a computer or not. We calculated the probability of all the possible events with respect to the events that have already occured. Thus we determined whether a person will buy a computer or not for given tuple 'X' using naive bayes algorithm.

→ Importance of Algorithm

- This algorithm is the best suited when it comes to categorical input variables
- It performs better than other models and requires much less training data of its assumption of the independence of features hold true
- It also plays an important role when it comes to solving multi-class predictions problems as it works quickly and saves times.

→ Applications

1) Real Time Prediction
2) Text classification
3) Spam Filtering
4) Recommendation system.