VIT Vidyalankar Institute of Technology www.vit.edu.in

Department of Computer Engineering Exp. No.4

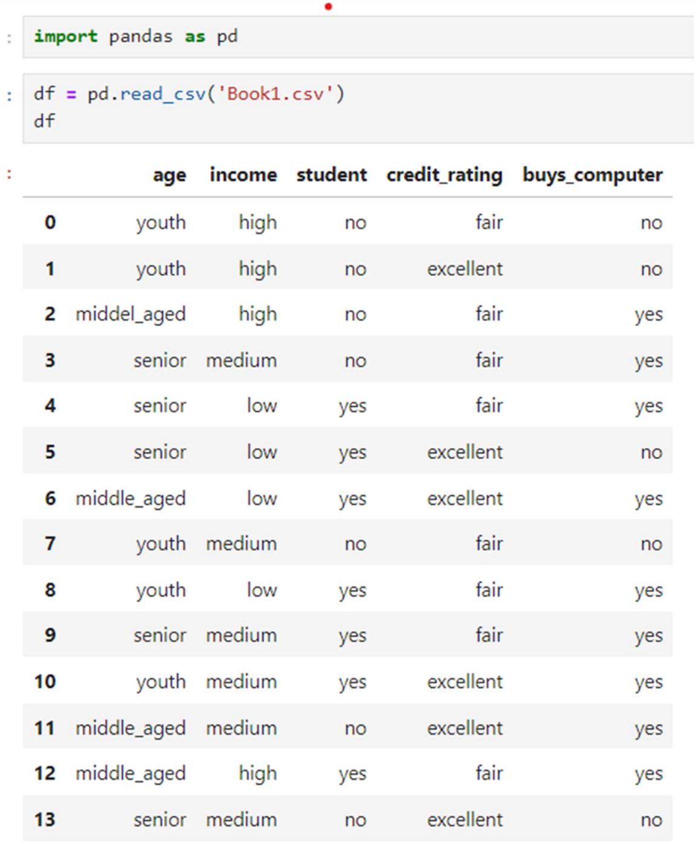
Semester	T.E. Semester VI – Computer Engineering
Subject	Data Warehousing and Mining
Subject Professor In-charge	Prof. Kavita Shirsat
Assisting Teachers	Prof. Kavita Shirsat
Laboratory	Lab 312 A

Student Name	Deep Salunkhe	
Roll Number	21102A0014	
Grade and Subject Teacher’s Signature		

Experiment Number	01	
Experiment Title	Naive Bayes classifier approach	
Resources / Apparatus Required	Hardware: Computer system	Software: Python
Description	Naive Bayes is a popular and simple machine learning classification algorithm that is based on Bayes' theorem. It's particularly useful for text classification and is known for its efficiency and effectiveness in various applications. Here's some key information about the Naive Bayes classifier approach: **1. Bayes' Theorem:** • The Naive Bayes classifier is built on Bayes' theorem, which is a fundamental probability theorem used to calculate conditional probabilities. • Bayes' theorem calculates the probability of an event based on prior knowledge of conditions that might be related to the event. **2. Independence Assumption:** • The "Naive" in Naive Bayes refers to the assumption that all features (attributes) used to predict the class label are independent of each other. In reality, this assumption is often not true, but it simplifies the calculations significantly and still produces good results in many cases. • Despite this simplification, Naive Bayes can perform surprisingly well, especially for text classification tasks. **3. Classification Task:** • Naive Bayes is primarily used for classification tasks, where the goal is to categorize data into predefined classes or labels. • It's commonly used for text classification problems, such as spam email detection, sentiment analysis, and document categorization. **4. Probability Calculation:**	

- The Naive Bayes classifier calculates the probability of an instance belonging to each possible class and assigns the instance to the class with the highest probability.
- It uses prior probabilities (based on training data) and conditional probabilities of each feature given the class to make these calculations.

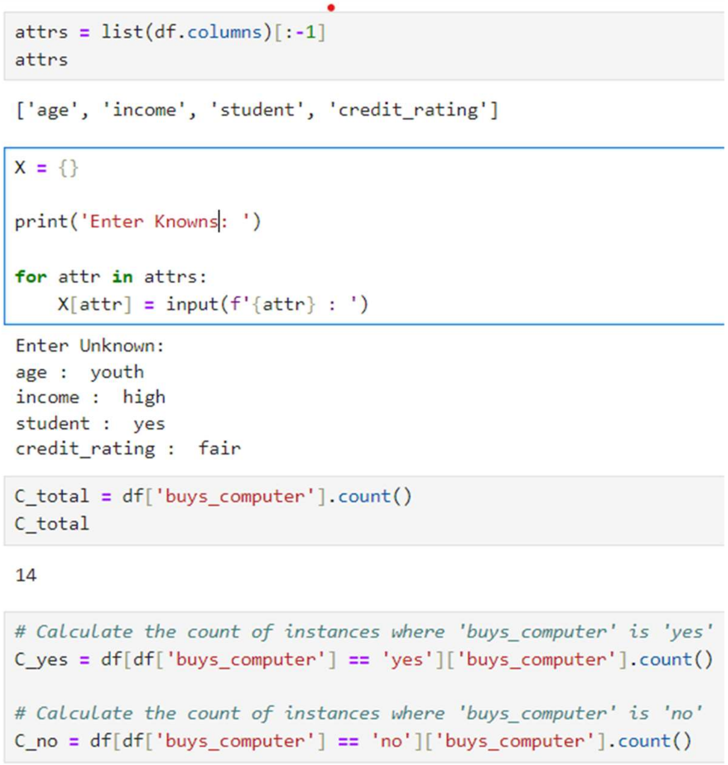
Program



```
import pandas as pd
```

```
df = pd.read_csv('Book1.csv')
df
```

	age	income	student	credit_rating	buys_computer
0	youth	high	no	fair	no
1	youth	high	no	excellent	no
2	middel_aged	high	no	fair	yes
3	senior	medium	no	fair	yes
4	senior	low	yes	fair	yes
5	senior	low	yes	excellent	no
6	middle_aged	low	yes	excellent	yes
7	youth	medium	no	fair	no
8	youth	low	yes	fair	yes
9	senior	medium	yes	fair	yes
10	youth	medium	yes	excellent	yes
11	middle_aged	medium	no	excellent	yes
12	middle_aged	high	yes	fair	yes
13	senior	medium	no	excellent	no



```
attrs = list(df.columns)[:-1]
attrs
```

```
['age', 'income', 'student', 'credit_rating']
```

```
X = {}

print('Enter Knowns: ')

for attr in attrs:
    X[attr] = input(f'{attr} : ')
```

```
Enter Unknown:
age :  youth
income :  high
student :  yes
credit_rating :  fair
```

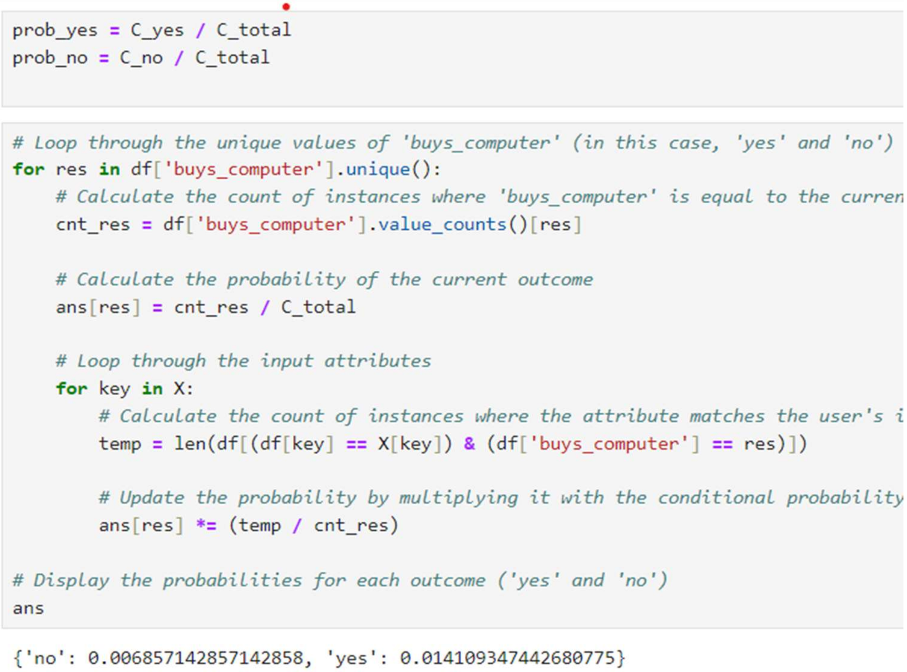
```
C_total = df['buys_computer'].count()
C_total
```

```
14
```

```
# Calculate the count of instances where 'buys_computer' is 'yes'
C_yes = df[df['buys_computer'] == 'yes']['buys_computer'].count()

# Calculate the count of instances where 'buys_computer' is 'no'
C_no = df[df['buys_computer'] == 'no']['buys_computer'].count()
```

Output



```
prob_yes = C_yes / C_total
prob_no = C_no / C_total
```

```
# Loop through the unique values of 'buys_computer' (in this case, 'yes' and 'no')
for res in df['buys_computer'].unique():
    # Calculate the count of instances where 'buys_computer' is equal to the curren
    cnt_res = df['buys_computer'].value_counts()[res]

    # Calculate the probability of the current outcome
    ans[res] = cnt_res / C_total

    # Loop through the input attributes
    for key in X:
        # Calculate the count of instances where the attribute matches the user's i
        temp = len(df[(df[key] == X[key]) & (df['buys_computer'] == res)])

        # Update the probability by multiplying it with the conditional probability
        ans[res] *= (temp / cnt_res)

# Display the probabilities for each outcome ('yes' and 'no')
ans
```

```
{'no': 0.006857142857142858, 'yes': 0.014109347442680775}
```

Conclusion:

Naive Bayes classifiers are a family of probabilistic classifiers that are particularly useful for text classification tasks and other situations where the independence assumption is a reasonable approximation. They are easy to implement, computationally efficient, and can deliver good results with proper data preprocessing and handling of feature independence.