Name : Aniket Narbariya

UID : 2019130043

Subject : AIML

Experiment : 4

Aim : For a given problem statement classify using Naïve Bayes Algorithm.

Theory:

Naïve Bayes

o Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.

o It is mainly used in text classification that includes a high-dimensional training dataset.

o Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.

o It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.

o Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.

The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:

o Naïve: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of colour, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple.
Hence each feature individually contributes to identify that it is an apple without depending on each other. o Bayes: It is called Bayes because it depends on the principle of Bayes' Theorem.

Program :

#ln[1]

```
import pandas as pd
import numpy as np
```

#ln[2]

```
df = pd.read_csv('./exp4.csv')
```

#ln[3]

```
df.head(20)
```

Output:

	Name	Gender	Height(in m)	Output
0	Kristina	F	1.60	Short
1	Jim	M	2.00	Tall
2	Maggie	F	1.90	Medium
3	Martha	F	1.88	Medium
4	Stephanie	F	1.70	Short
5	Bob	M	1.85	Medium
6	Kathy	F	1.60	Short
7	Dave	M	1.70	Short
8	Worth	M	2.20	Tall
9	Steven	M	2.10	Tall
10	Debbie	F	1.80	Medium
11	Todd	M	1.95	Medium
12	Kim	F	1.90	Medium
13	Amy	F	1.80	Medium
14	Wynette	F	1.75	Medium
15	Walter	M	1.65	Short
16	Corey	M	1.92	Medium
17	Natasha	F	1.50	Short

#ln[4]

```
def dispDiv(num, den, p):
 return (str(num)+"/"+str(den)+" = "+str(p))

total = df.shape[0]

total_short = df['Output'].value_counts()['Short']

total_medium = df['Output'].value_counts()['Medium']

total_tall = df['Output'].value_counts()['Tall']


# Prior Probability
p_short = total_short/total
```

```
p_medium = total_medium/total
p_tall = total_tall/total
```

#ln[5]

```
ef conditional_prob(attribute, att_value, output):
 cp = df.apply(lambda x: True if (x[attribute]==att_value and x['Output']==output) else False,
axis=1)
 cp = len(cp[cp == True])
# print(cp)
 return cp

total_m = df['Gender'].value_counts()['M']
total_f = df['Gender'].value_counts()['F']
m_short = conditional_prob("Gender","M", "Short")
m_medium = conditional_prob("Gender","M","Medium")
m_tall = conditional_prob("Gender","M","Tall")
f_short = conditional_prob("Gender","F","Short")
f_medium = conditional_prob("Gender","F","Medium")
f_tall = conditional_prob("Gender","F","Tall")
p_m_short = m_short/total_short
p_m_medium = m_medium/total_medium
p_m_tall = m_tall/total_tall
p_f_short = f_short/total_short
p_f_medium = f_medium/total_medium
p_f_tall = f_tall/total_tall
print("P(M|Short) = ",dispDiv(m_short, total_short, p_m_short))
print("P(M|Medium) = ",dispDiv(m_medium, total_medium, p_m_medium))
print("P(M|Tall) = ",dispDiv(m_tall, total_tall, p_m_tall))
print("P(F|Short) = ",dispDiv(f_short, total_short, p_f_short))
print("P(F|Medium) = ",dispDiv(f_medium, total_medium, p_f_medium))
print("P(F|Tall) = ",dispDiv(f_tall, total_tall, p_f_tall))
```

#ln[6]

```
#returns mean and std deviation
def get_mean(output, column):
 mean_arr = df.loc[df['Output']==output]
 mean_arr = mean_arr[column].to_numpy()
 print("Mean (",output,"): ",np.mean(mean_arr),",\n StdDeviation (",output,"):
",np.std(mean_arr, ddof=1), "\tArray- ",mean_arr)
 return (np.mean(mean_arr),np.std((mean_arr), ddof=1))

s_mean_height, s_sd_height = get_mean("Short", "Height(in m)")
m_mean_height, m_sd_height = get_mean("Medium", "Height(in m)")
```

```
t_mean_height, t_sd_height = get_mean("Tall", "Height(in m)")
```

Output:

```
Mean ( Short ):  1.6328571428571428 ,
 StdDeviation ( Short ):  0.07227592630889983   Array-  [1.6  1.7  1.6  1.7  1.65 1.5  1.68]
Mean ( Medium ):  1.8577777777777778 ,
 StdDeviation ( Medium ):  0.0672268134336623   Array-  [1.9  1.88 1.8  1.95 1.9  1.8  1.75 1.92 1.82]
Mean ( Tall ):  2.1 ,
 StdDeviation ( Tall ):  0.10000000000000009    Array-  [2.  2.2 2.1]
```

#ln[7]

```
#Creating a Function.
def normal_dist(x , mean , sd):
# prob_density = (np.exp(-0.5*((x-mean)/sd)**2))/(sd*np.sqrt(2*np.pi))
prob_density = 1/(sd * np.sqrt(2 * np.pi)) * np.exp( - (x - mean)**2 / (2 * sd**2) )
# print(prob_density)
 return prob_density
```

#ln[8]

```
def bayes(name, gender, h):
    if gender=="M":
        p_x_short = p_short*p_m_short*normal_dist(h,s_mean_height,s_sd_height)
        p_x_medium = p_medium*p_m_medium*normal_dist(h,m_mean_height,m_sd_height)
        p_x_tall = p_tall*p_m_tall*normal_dist(h,t_mean_height,t_sd_height)
        output = max(p_x_short, p_x_medium, p_x_tall)
        if output == p_x_short:
            print(name," is short")
        elif output == p_x_medium:
            print(name," is medium")
        elif output == p_x_tall:
            print(name," is tall")
    elif gender=="F":
        p_x_short = p_short*p_f_short*normal_dist(h,s_mean_height,s_sd_height)
        p_x_medium = p_medium*p_f_medium*normal_dist(h,m_mean_height,m_sd_height)
        p_x_tall = p_tall*p_f_tall*normal_dist(h,t_mean_height,t_sd_height)
        output = max(p_x_short, p_x_medium, p_x_tall)
        if output == p_x_short:
            print(name," is short")
        elif output == p_x_medium:
            print(name," is medium")
        elif output == p_x_tall:
```

```
        print(name," is tall")
```

#ln[8]

```
Param = bayes("Param","M",1.97)
```

Output:

```
Param  is tall
```

#ln[9]

```
Divya = bayes("Divya","F",1.72)
```

Output:

```
 Divya  is short
```

Conclusion:

In the above experiment of AIML Lab, I learnt about Naïve Bayes Algorithm. This algorithm is used for supervised learning models. This algorithm is easy to use as we make assumption that the attributes are independent and then classify the input.