Sipna College of Engineering & Technology, Amravati.
Department of Computer Science & Engineering
Session 2022-2023

Branch :- Computer Sci. & Engg. **Class :- Final Year**
Subject :-Artificial Intelligence and Machine Learning **Sem :- VIII**

Teacher Manual

PRACTICAL NO 5

AIM: To Write the program to calculate the accuracy, precision, and recall for your data set by using the naïve bayesian classifier model.

S/W REQUIRED: Python

DATA SET USED: Document.csv

A Naive Bayes Classifier

A Naive Bayes Classifier is a probabilistic classifier, and one of the most fundamental classification models. The reason we refer to the classifier as a "naive" one is because this classifier naively assumes that all features in the dataset are independent of each other, i.e. conditional independence**.**

From this standpoint, feature selection is not typically a strength of the Naive Bayes Classifier. Given the assumption that all features are independent of each other, this classifier is at risk of performing poorly when significant correlation exists between the features.

Bayes' Theorem is defined as follows:

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

This theorem works on the basis of conditional probability, i.e. the probability of an outcome occurring given the occurrence of a previous outcome. In this instance, the probabilities of a particular event occurring are updated as new information is gathered.

Confusion matrix

The confusion matrix is another metric that is often used to measure the performance of a classification algorithm. True to its name, the terminology related to the confusion matrix can be rather confusing, but the matrix itself is simple to understand.

The entries of the confusion matrix are the number of occurrences of each class for the dataset being analysed.

Accuracy

The first metric we are going to discuss is, perhaps, the simplest one, the accuracy. It answers the question:

"How often is the classifier correct?"

It can be obtained simply using the following formulae:

$$\text{Accuracy} = \frac{\#\text{correctly classified items}}{\#\text{all classified items}}$$

Precision: it answers the question:
"When it predicts the positive result, how often is it correct?"

This is obtained by using the following formulae:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Precision is usually used when the goal is to limit the number of false positives (FP). For example, this would be the metric to focus on if our goal with the spam filtering algorithm is to minimize the number of reals emails that are classified as spam.

Recall: it answers the question:
"When it is actually the positive result, how often does it predict correctly?"

This is obtained by using the following formulae:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Recall is usually used when the goal is to limit the number of false negatives (FN). In our example, that would correspond to minimizing the number of spam emails that are classified as real emails. Recall is also known as "sensitivity" and "true positive rate" (TPR).

Implementation:

```
import pandas as pd
msg = pd.read_csv('document.csv', names=['message', 'label'])
print("Total Instances of Dataset: ", msg.shape[0])
msg['labelnum'] = msg.label.map({'pos': 1, 'neg': 0})

X = msg.message
y = msg.labelnum
from sklearn.model_selection import train_test_split
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y)
from sklearn.feature_extraction.text import CountVectorizer

count_v = CountVectorizer()
Xtrain_dm = count_v.fit_transform(Xtrain)
Xtest_dm = count_v.transform(Xtest)

df = pd.DataFrame(Xtrain_dm.toarray(),columns=count_v.get_feature_names())
print(df[0:5])

from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB()
clf.fit(Xtrain_dm, ytrain)
pred = clf.predict(Xtest_dm)
```

```
for doc, p in zip(Xtrain, pred):
    p = 'pos' if p == 1 else 'neg'
    print("%s -> %s" % (doc, p))

from sklearn.metrics import accuracy_score, confusion_matrix, precision_score,
recall_score
print('Accuracy Metrics: \n')
print('Accuracy: ', accuracy_score(ytest, pred))
print('Recall: ', recall_score(ytest, pred))
print('Precision: ', precision_score(ytest, pred))
print('Confusion Matrix: \n', confusion_matrix(ytest, pred))
```

Output

```
Total Instances of Dataset:  18
   about  am  an  and  awesome  bad  beers  can  dance  deal  ...  these  \
0      1   0   0    0        0    0      1    0      0     0  ...      1
1      0   1   0    0        0    0      0    0      0     0  ...      0
2      0   0   0    0        0    0      0    0      0     0  ...      0
3      0   0   0    0        0    0      0    0      0     0  ...      0
4      0   0   0    0        0    0      0    0      0     0  ...      0

   this  tired  to  today  very  view  went  what  with
0     0      0   0      0     1     0     0     0     0
1     1      1   0      0     0     0     0     0     0
2     1      0   0      0     0     0     0     0     0
3     0      0   1      1     0     0     1     0     0
4     0      0   0      0     0     0     0     0     0

[5 rows x 45 columns]
I feel very good about these beers -> pos
I am tired of this stuff -> pos
I do not like this restaurant -> neg
I went to my enemy's house today -> neg
He is my sworn enemy -> neg
Accuracy Metrics: 

Accuracy:  0.6
Recall:  0.5
Precision:  1.0
Confusion Matrix: 
 [[1 0]
 [2 2]]
```

CONCLUSION: Thus we have implemented the concept of decision tree using ID3 Algorithm