DATE:

PAGE NO.

Lab Program-5

Write a program to implement the naïve Bayesian classifier for a sample training data set stored as .csv file. Compute the accuracy of the classifier considering few test data sets.

→ It is a classification technique based on Bayes Theorem with an assumption of independence among predictors. In simple terms, a Naïve Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of other feature. For example- a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as "Naïve".

Shubhransh Gupta : 1BG17CS094

DATE: PAGE NO.

```
print("Naive Bayes Classifier for Concept learning problem")
import csv
import random
import math
import operator.
def safe_div(x,y):
    if y==0:
        return 0
    return x/y

def loadCsv(filename):
    lines = csv.reader(open(filname))
    dataset = list(lines)
    for i in range(len(dataset)):
        dataset[i] = [float(x) for x in dataset[i]]
    return dataset

def splitDataset(dataset, splitRatio):
    trainSize = int(len(dataset) * splitRatio)
    trainSet = []
    copy = list(dataset)
    i=0
    while len(trainSet) < trainSize:
        trainSet.append(copy.pop(i))
    return [trainSet, copy]
```

Teacher's Remarks

45

Teacher's Signature

DATE:

PAGE NO.

```
def separateByClass(dataset):
    separated = {}
    for i in range(len(dataset)):
        vector = dataset[i]
        if (vector[-1] not in separated):
            separated[vector[-1]] = []
        separated[vector[-1]].append(vector)
    return separated

def mean(numbers):
    return safe_div(sum(numbers), float(len(numbers)))

def stdev(numbers):
    avg = mean(numbers)
    variance = safe_div(sum([pow(x-avg, 2)
                for x in numbers]), float(len(numbers)-1))
    return math.sqrt(variance)

def summarize(dataset):
    summaries = [(mean(attribute), stdev(attribute)) for
                attribute in zip(*dataset)]
    del summaries[-1]
    return summaries.

def summarizeByClass(dataset):
    separated = separateByClass(dataset)
    summaries = {}
```

Teacher's Remarks

47

Teacher's Signature

DATE:

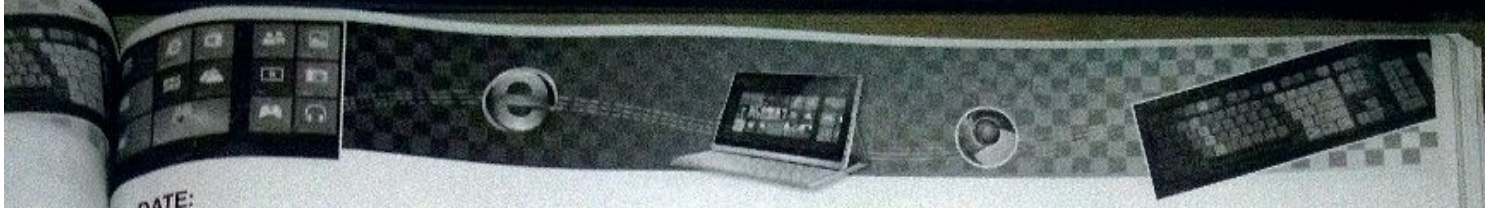
PAGE NO.

```
for classValue, instances in separated.items():
        summaries[classValue] = summarize(instances)
return summaries.

def calculateProbabilities(x, mean, stdev):
    exponent = math.exp(-safe_div(math.pow(x-mean,2),
                    (2*math.pow(stdev,2))))
    final = safe_div(1, (math.sqrt(2*math.pi)*stdev))
                    * exponent
    return final

def calculateClassProbabilities(summaries, inputVector):
    probabilities = {}
    for classValue, classSummaries in summaries.items():
        probabilities[classValue] = 1
        for i in range(len(classSummaries)):
            mean, stdev = classSummaries[i]
            x = inputVector[i]
            probabilities[classValue] *= calculateProbabilities
                    (x, mean, stdev)
    return probabilities.

def predict(summaries, inputVector):
    probabilities = calculateClassProbabilities(summaries,
                    inputVector)
    bestLabel, bestProb = None, -1
```

Teacher's Remarks

49

Teacher's Signature



DATE:

PAGE NO.

```
    for classValue, probability in probabilities.items():
        if bestLabel is None or probability > bestProb:
            bestProb = probability
            bestLabel = classValue
    return bestLabel.

def getPrediction(summaries, testSet):
    predictions = []
    for i in range(len(testSet)):
        result = predict(summaries, testSet[i])
        predictions.append(result)
    return predictions.

def getAccuracy(testSet, predictions):
    correct = 0
    for i in range(len(testSet)):
        if testSet[i][-1] == predictions[i]:
            correct += 1
    accuracy = safe_div(correct, float(len(testSet)))
                                        * 100

    return accuracy

def main():
    filename = 'Concept Learning.csv'
    splitRatio = 0.80
    dataset = loadCsv(filename)
```

Teacher's Remarks

Teacher's Signature

DATE:

PAGE NO.

```
trainingSet, testSet = splitDataset(dataset, splitRatio)
print('Split {0} rows into'.format(len(dataset)))
print('Number of Training data' + (repr(len(trainingSet))))
print('Number of Test Data' + (repr(len(testSet))))
print('The values assumed for the concept learning attributes are')
print('OUTLOOK => Sunny = 1
                  Overcast = 2
                  Rain = 3
 TEMPERATURE => Hot = 1
                Mild = 2
                Cool = 3
 HUMIDITY => High = 1
             Normal = 2
 Wind => Weak 1, Strong = 2')
print('Target Concept Play tennis' => Yes = 10 No = 5')
print('The training set are:')
    for x in trainingSet:
        print(x)
    print('The Test dataSet are:')
    for x in testSet:
        print(x)
    print('\n')
summaries = summarizeByClass(trainingSet)
predictions = getPredictions(summaries, testSet)
actual = []
```

Teacher's Remarks

53

Teacher's Signature

DATE: PAGE NO.

```
for i in range(len(testSet)):
    vector = testSet[i]
    actual.append(vector[-1])
print('Actual value: {0}'.format(actual))
print('Predictions: {0}'.format(predictions))
accuracy = getAccuracy(testSet, predictions)
print('Accuracy: {0}'.format(accuracy))

main()
```

Teacher's Signature

DATE:

Output-

Naive Bayes Classifier for concept Learning problem
Split 16 rows into
Number of Training data 12
Number of Test data 4

The values assumed for the concept learning attributes are

OUTLOOK ⇒ Sunny=1 Overcast=2 Rain=3
TEMPERATURE ⇒ Hot=1 Mild=2 Cool=3
HUMIDITY ⇒ High=1 Normal=2
WIND ⇒ Weak=1 Strong=2
Target Concept: Play tennis ⇒ Yes=10 No=5

The Training set are:

[1.0, 1.0, 1.0, 1.0, 5.0]
[1.0, 1.0, 1.0, 2.0, 5.0]
[2.0, 1.0, 1.0, 2.0, 10.0]
[3.0, 2.0, 1.0, 1.0, 10.0]
[3.0, 3.0, 2.0, 1.0, 10.0]
[3.0, 3.0, 2.0, 2.0, 5.0]
[2.0, 3.0, 3.0, 2.0, 10.0]
[1.0, 2.0, 1.0, 1.0, 5.0]
[1.0, 3.0, 20, 1.0, 10.0]
[3.0, 2.0, 2.0, 2.0, 10.0]
[1.0, 2.0, 2.0, 2.0, 10.0]
[2.0, 2.0, 1.0, 2.0, 10.0]

Teacher's Remarks

46

Teacher's Signature

DATE:

The test data set are :

[2.0, 1.0, 2.0, 1.0, 10.0]
[3.0, 2.0, 1.0, 2.0, 5.0]
[10, 2.0, 1.0, 2.0, 10.0]
[1.0, 2.0, 1.0, 2.0, 5.0]

Actual values : [10.0, 5.0, 10.0, 5.0]
Predictions : [5.0, 10.0, 5.0, 5.0]
Accuracy 25.0 %.