**Name-RUDRAKSH YADAV**

**Admission No-18SCSE1010434**

**Subject/Sec-ML /ELECTIVE 2**

## QUESTION-

1. Write a program to demonstrate the working of Naive Bayes classifier. Compute the accuracy of the classifier, considering few test data sets.

## AIM-

To demonstrate the working of Naive Bayes classifier. Compute the accuracy of the classifier, considering few test data sets.

## ALGORITHM-

**Step 1:** Convert the data set into a frequency table.

**Step 2:** Create Likelihood table by finding the probabilities like Overcast probability = 0.29 and probability of playing is 0.64.

**Step 3:** Now, use Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.

## CONCEPT-

Players will play if weather is sunny. Is this statement is correct? We can solve it using above discussed method of posterior probability. P(Yes | Sunny) = P( Sunny | Yes) * P(Yes) / P (Sunny) Here we have P (Sunny |Yes) = 3/9 = 0.33, P(Sunny) = 5/14 = 0.36, P( Yes)= 9/14 = 0.64 Now, P (Yes | Sunny) = 0.33 * 0.64 / 0.36 = 0.60, which has higher probability. Naive Bayes uses a similar method to predict the probability of different class based on

various attributes. This algorithm is mostly used in text classification and with problems having multiple classes.

## SOURCE CODE-

```
import csv
import random
import math
def loadCsv(filename):
lines = csv.reader(open(filename, "r"))
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)
while len(trainSet) < trainSize:
index = random.randrange(len(copy))
trainSet.append(copy.pop(index))
return [trainSet, copy]

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 sum(numbers)/float(len(numbers))
def stdev(numbers):
avg = mean(numbers)
variance = 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 = {}
for classValue, instances in separated.items():
summaries[classValue] = summarize(instances)
return summaries
def calculateProbability(x, mean, stdev):
exponent = math.exp(-(math.pow(x-
mean,2)/(2*math.pow(stdev,2))))
return (1 / (math.sqrt(2*math.pi) * stdev)) * exponent 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] *= calculateProbability(x, mean, stdev)
return probabilities
def predict(summaries, inputVector):
probabilities = calculateClassProbabilities(summaries, inputVector)
bestLabel, bestProb = None, -1
for classValue, probability in probabilities.items():
if bestLabel is None or probability > bestProb: bestProb =
probability
bestLabel = classValue
return bestLabel
def getPredictions(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
return (correct/float(len(testSet))) * 100.0

def main():
filename = 'data.csv'
splitRatio = 0.67
```

```
dataset = loadCsv(filename)
trainingSet, testSet = splitDataset(dataset, splitRatio) print('Split {0}
rows into train={1} and test={2} rows'.format(len(dataset),
len(trainingSet), len(testSet)))
# prepare model
summaries = summarizeByClass(trainingSet)
# test model
predictions = getPredictions(summaries, testSet) accuracy =
getAccuracy(testSet, predictions) print('Accuracy:
{0}%'.format(accuracy))
main()
```

## OUTPUT-

```
Split 306 rows into train=205 and test=101 rows Accuracy:
72.27722772277228%
```

## RESULT-

The Program Successfully run…..