

SCHOOL OF COMPUTER SCIENCE



MACHINE LEARNING LAB

BCSE3093

Submitted to –

Pratyush Deka

Submitted by:

Aditi Rai

(18SCSE1010534)

Introduction

Naive Bayes algorithms are a set of supervised machine learning algorithms based on the Bayes probability theorem, which we'll discuss in this article. Naive Bayes algorithms assume that there's no correlation between features in a dataset used to train the model.

In spite of this oversimplified assumption, naive Bayes classifiers work very well in many complex real-world problems. A big advantage of naive Bayes classifiers is that they only require a relatively small number of training data samples to perform classification efficiently, compared to other algorithms like logistic regression, decision tree, and svm. Before we dive into the implementation, let's first cover some key terms related to naive Bayes.

Key Terms

Bayes Theorem

The Bayes theorem describes the probability of a feature, based on prior knowledge of situations related to that feature. For example, if the probability of someone having diabetes is related to his or her age, then by using the Bayes theorem, the age can be used to more accurately predict the probability of diabetes.

Naive

The word **naive** implies that every pair of features in the dataset is independent of each other. All naive Bayes classifiers work on the assumption that the value of a particular feature is independent from the value of any other feature for a given the class.

For example, a fruit may be classified as an orange if it's round, about 8 cm in diameter, and is orange in color. With a naive Bayes classifier, each of these three features (shape, size, and color) contributes independently to the probability that this fruit is an orange. Also, it's assumed that there is no possible correlation between the shape, size, and color attributes.

Problem: Write a program to implement the naïve Bayesian classifier for a sample training data set stored as a .CSV file.

Case study is:

The problem statement is to classify patients as diabetic or non diabetic. The dataset can be downloaded from Kaggle website that is 'PIMA INDIAN DIABETES DATABASE'. The datasets had several different medical predictor features and a target that is 'Outcome'. Predictor variables include the number of pregnancies the patient has had, their BMI, insulin level, age, and so on.

Now, Summarize data

the naive bayes model is comprised of a summary of the data in the training dataset.

This summary is then used when making predictions.

#involves the mean and the standard deviation for each attribute, by class value

And

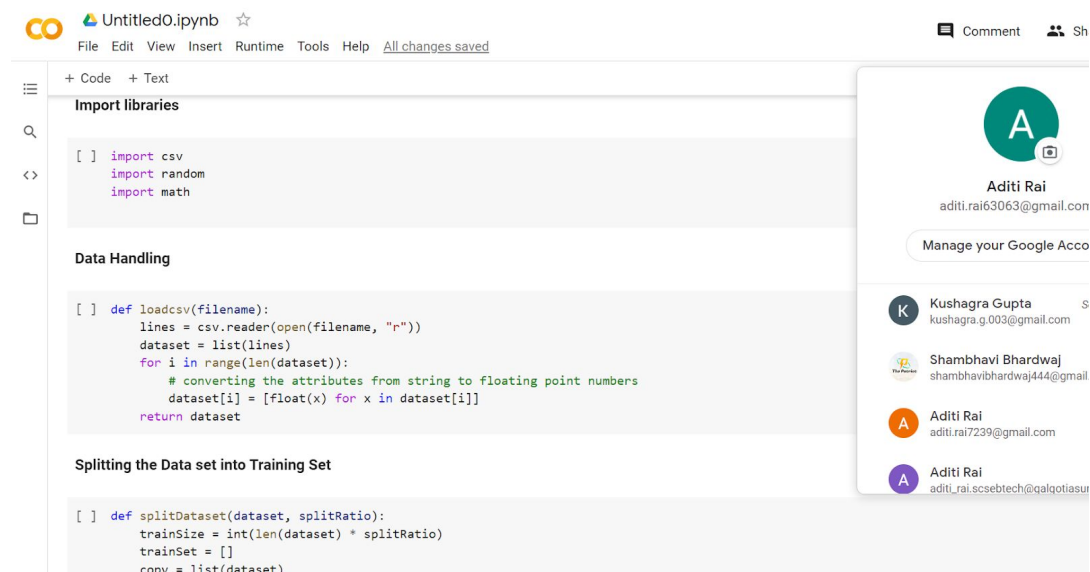
Separate Data By Class

Function to categorize the dataset in terms of classes

The function assumes that the last attribute (-1) is the class value.

The function returns a map of class values to lists of data instances.

def separateByClass(dataset):



The screenshot shows a Jupyter Notebook titled 'Untitled0.ipynb' with a menu bar (File, Edit, View, Insert, Runtime, Tools, Help) and a status bar ('All changes saved'). The notebook is divided into three sections: 'Import libraries', 'Data Handling', and 'Splitting the Data set into Training Set'. The 'Import libraries' section contains code to import csv, random, and math. The 'Data Handling' section contains a function 'loadcsv(filename)' that reads a CSV file, converts string attributes to floats, and returns the dataset. The 'Splitting the Data set into Training Set' section contains a function 'splitDataset(dataset, splitRatio)' that splits the dataset into a training set and a copy. On the right side of the notebook, there is a sidebar with a user profile for 'Aditi Rai' (aditi.ra163063@gmail.com) and a list of other users: 'Kushagra Gupta' (kushagra.g.003@gmail.com), 'Shambhavi Bhardwaj' (shambhavi.bhardwaj444@gmail.com), and another 'Aditi Rai' (aditi.ra17239@gmail.com).

```
[ ] import csv
import random
import math

def loadcsv(filename):
    lines = csv.reader(open(filename, "r"))
    dataset = list(lines)
    for i in range(len(dataset)):
        # converting the attributes from string to floating point numbers
        dataset[i] = [float(x) for x in dataset[i]]
    return dataset

def splitDataset(dataset, splitRatio):
    trainSize = int(len(dataset) * splitRatio)
    trainSet = []
    copy = list(dataset)
```

+ Code + Text


Splitting the Data set into Training Set

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

Separate Data By Class

```
[ ] 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
```

Calculate mean


 Untitled0.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

+ Code + Text

```
[ ] def summarizeByClass(dataset):
    separated = separateByClass(dataset)
    summaries = {}
    for classValue, instances in separated.items():
        summaries[classValue] = summarize(instances)
    return summaries
```

Make Prediction

```
[ ] 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
```

Untitled0.ipynb

File Edit View Insert Runtime Tools Help [All changes saved](#)

+ Code + Text

Prediction : look for the largest probability and return the associated class

```
[ ] 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
```

Accuracy

```
[ ] def getAccuracy(testSet, predictions):
    correct = 0
    for i in range(len(testSet)):
```

Untitled0.ipynb

File Edit View Insert Runtime Tools Help [All changes saved](#)

+ Code + Text

Accuracy

```
[ ] 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
```

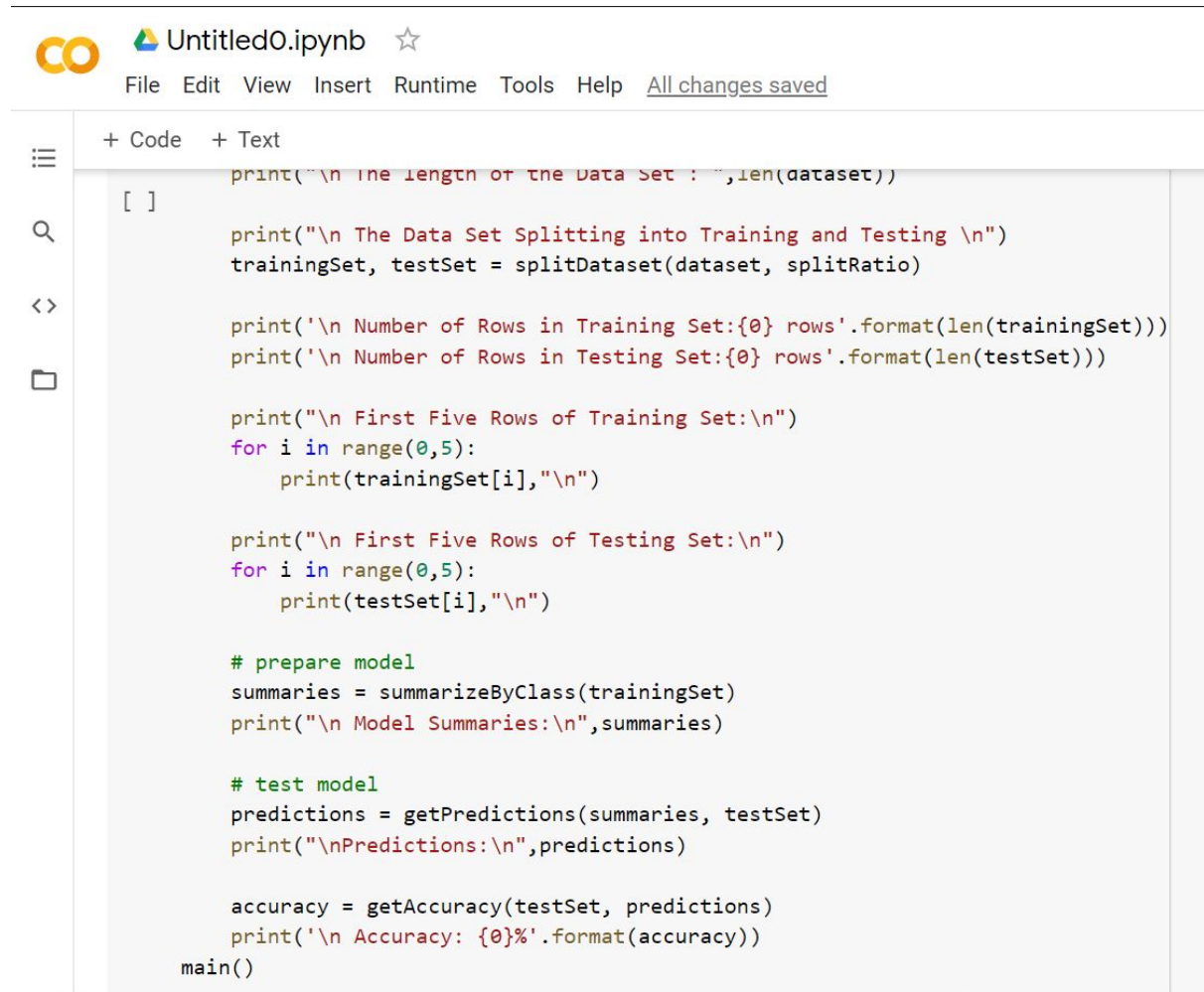
Main function

```
[ ] def main():
    filename = 'C:\\Users\\Aditi \\Desktop\\Data\\pima-indians-diabetes.csv'
    splitRatio = 0.67
    dataset = loadcsv(filename)

    #print("\n The Data Set :\n",dataset)
    print("\n The length of the Data Set : ",len(dataset))

    print("\n The Data Set Splitting into Training and Testing \n")
    trainingSet, testSet = splitDataset(dataset, splitRatio)

    print('\n Number of Rows in Training Set:{0} rows'.format(len(trainingSet)))
    print('\n Number of Rows in Testing Set:{0} rows'.format(len(testSet)))
```



The image shows a Jupyter Notebook interface with a file named 'Untitled0.ipynb'. The notebook contains a Python script that performs the following steps: 1. Prints the length of the dataset (768). 2. Splits the dataset into training and testing sets. 3. Prints the number of rows in the training set (514) and testing set (254). 4. Prints the first five rows of the training set. 5. Prints the first five rows of the testing set. 6. Prepares a model by summarizing the training set. 7. Tests the model by getting predictions for the testing set. 8. Prints the accuracy of the model (0.463).

```
[ ]
print("\n The length of the Data Set : ",len(dataset))

print("\n The Data Set Splitting into Training and Testing \n")
trainingSet, testSet = splitDataset(dataset, splitRatio)

print('\n Number of Rows in Training Set:{0} rows'.format(len(trainingSet)))
print('\n Number of Rows in Testing Set:{0} rows'.format(len(testSet)))

print("\n First Five Rows of Training Set:\n")
for i in range(0,5):
    print(trainingSet[i],"\n")

print("\n First Five Rows of Testing Set:\n")
for i in range(0,5):
    print(testSet[i],"\n")

# prepare model
summaries = summarizeByClass(trainingSet)
print("\n Model Summaries:\n",summaries)

# test model
predictions = getPredictions(summaries, testSet)
print("\nPredictions:\n",predictions)

accuracy = getAccuracy(testSet, predictions)
print('\n Accuracy: {0}%'.format(accuracy))
main()
```

O/P:

The length of the Data Set : 768

The Data Set Splitting into Training and Testing

Number of Rows in Training Set:514 rows

Number of Rows in Testing Set:254 rows

Number of Rows in Training Set:514 rows

Number of Rows in Testing Set:254 rows

First Five Rows of Training Set:

[4.0, 116.0, 72.0, 12.0, 87.0, 22.1, 0.463, 37.0, 0.0]

ADITI RAI

First Five Rows of Training Set:

[4.0, 116.0, 72.0, 12.0, 87.0, 22.1, 0.463, 37.0, 0.0]

```
[0.0, 84.0, 64.0, 22.0, 66.0, 35.8, 0.545, 21.0, 0.0]
```

```
[0.0, 162.0, 76.0, 36.0, 0.0, 49.6, 0.364, 26.0, 1.0]
```

```
[10.0, 101.0, 86.0, 37.0, 0.0, 45.6, 1.136, 38.0, 1.0]
```

```
[5.0, 78.0, 48.0, 0.0, 0.0, 33.7, 0.654, 25.0, 0.0]
```

First Five Rows of Testing Set:

```
[1.0, 85.0, 66.0, 29.0, 0.0, 26.6, 0.351, 31.0, 0.0]
```

```
[8.0, 183.0, 64.0, 0.0, 0.0, 23.3, 0.672, 32.0, 1.0]
```

```
[4.0, 110.0, 92.0, 0.0, 0.0, 37.6, 0.191, 30.0, 0.0]
```

```
[10.0, 139.0, 80.0, 0.0, 0.0, 27.1, 1.441, 57.0, 0.0]
```

```
[7.0, 100.0, 0.0, 0.0, 0.0, 30.0, 0.484, 32.0, 1.0]
```

Model Summaries:

{0.0: [(3.3474320241691844, 3.045635385378286), (111.54380664652568, 26.040069054720693), (68.45921450151057, 18.15540652389224), (19.94561933534743, 14.709615608767137), (71.50151057401813, 101.04863439385403), (30.863141993957708, 7.207208162103949), (0.4341842900302116, 0.2960911906946818), (31.613293051359516, 12.100651311117689)], 1.0: [(4.469945355191257, 3.7369440851983082), (139.3879781420765, 33.733070931373234), (71.14754098360656, 20.694403393963842), (22.92896174863388, 18.151995092528765), (107.97814207650273, 146.92526156736633), (35.28633879781422, 7.783342260348583), (0.5569726775956286, 0.3942245334398509), (36.78688524590164, 11.174610282702282)]}

Prediction:

Predictions:

[illegible]

Accuracy:

Accuracy: 80.31496062992126%

In []: