Lab_10_Naive_Bayes

October 18, 2021

0.1 *LAB 10 : Naive Bayes Classifier*

1. Binary Classification using Naive Bayes Classifier

2. Sentiment Analysis using Naive Bayes

```
[4]: import numpy as np
     import matplotlib.pyplot as plt
```

0.2 Binary Classification using Naive Bayes Classifier

Useful References : 1. https://machinelearningmastery.com/naive-bayes-classifier-scratch-python/

2. https://www.analyticsvidhya.com/blog/2021/01/a-guide-to-the-naive-bayes-algorithm/

3. https://towardsdatascience.com/implementing-naive-bayes-algorithm-from-scratch-python-c6880cfc9c41

Note : The goal of this experiment is to perform and understand Naive Bayes classification by applying it on the below dataset, you can either fill in the below functions to get the result or you can create a class of your own using the above references to perform classification

1. Generation of 2D training data

```
[57]: mean1=np.array([0,0])
      mean2=np.array([4,5])
      var=np.array([[1,0.1],[0.1,1]])
      np.random.seed(0)
      data1=np.random.multivariate_normal(mean1,var,50)
      data2=np.random.multivariate_normal(mean2,var,50)
      data=np.concatenate((data1,data2))
      label=np.concatenate((np.zeros(data1.shape[0]),np.ones(data2.shape[0])))

      plt.figure()
      plt.scatter(data[:,0],data[:,1],c=label)
      plt.title('Data visualization')
```

```
[57]: Text(0.5, 1.0, 'Data visualization')
```

Data visualization
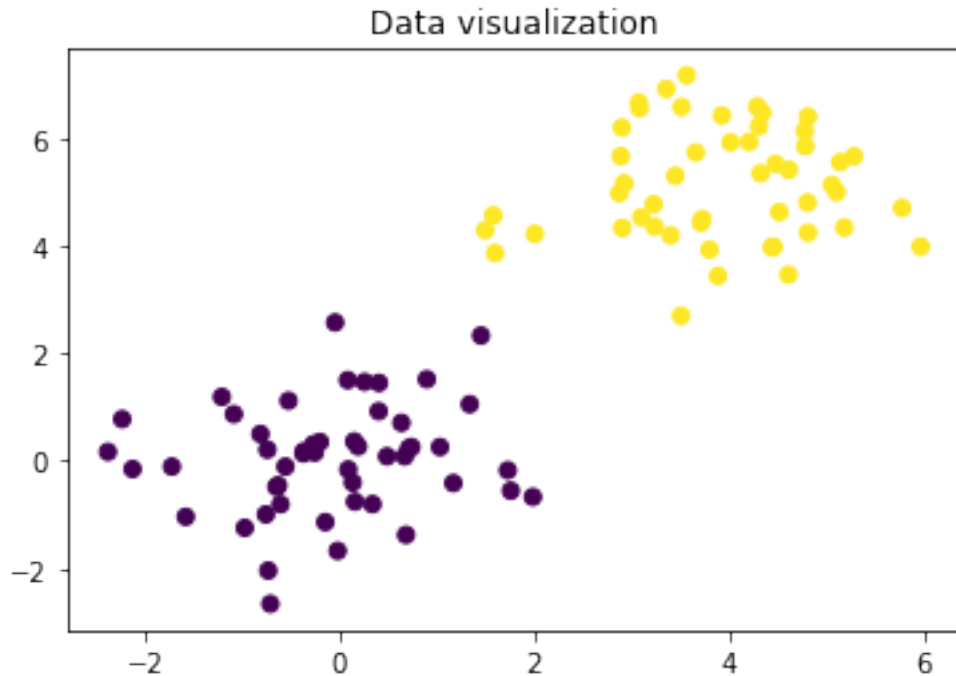
2. Split the Dataset by Class Values (Create a Dictionary)

```
[58]: def class_dictionary(data,label):
    class_dict = {}

    ## Write your code here

    return class_dict
```

```
This is the Class Dictionary :
{0: [array([-1.57668985, -1.03982262]), array([-2.22908837,  0.77738534]),
array([-0.72944014, -2.04059601]), array([-0.60307093, -0.80613793]),
array([-0.1988887,  0.351987 ]), array([-1.0823819 ,  0.86873075]),
array([-0.64602276, -0.4827786 ]), array([-0.55301333, -0.10534224]),
array([-0.97041435, -1.24566304]), array([ 0.34076762, -0.80512206]),
array([1.45488724, 2.3318086 ]), array([-0.14322361, -1.13894247]),
array([-0.70767693, -2.65891323]), array([ 0.09163132, -0.15950217]),
array([-2.12241532, -0.15106366]), array([-0.36859122,  0.13876704]), array([
1.9871582 , -0.67035913]), array([0.15313648, 0.36290063]), array([-1.71898891,
-0.10582707]), array([0.49004011, 0.08445841]), array([ 1.73020468, -0.1749493
]), array([-0.04321608,  2.57402376]), array([0.67183735, 0.084099  ]),
array([0.40754152, 1.45065429]), array([1.3396092 , 1.05418816]),
array([0.40455369, 0.92363786]), array([ 1.17081537, -0.41316892]),
array([-0.26643325,  0.30823425]), array([-0.25223481,  0.15357382]),
```

2

```
array([0.71376003, 0.22709168]), array([0.73990561, 0.25751442]), array([
1.761071  , -0.55498015]), array([ 0.1379401 , -0.40110559]), array([0.89854368,
1.51943123]), array([0.63802444, 0.70771651]), array([-0.6272324 ,
-0.45418367]), array([-0.01665582, -1.67334851]), array([ 0.16100003,
-0.75776912]), array([1.03410461, 0.25749629]), array([0.19337668, 0.2687304 ]),
array([0.25980547, 1.46839103]), array([ 0.68519909, -1.3758881 ]),
array([-2.37551849,  0.16808376]), array([-0.75350908, -0.99490358]),
array([0.08674368, 1.50143912]), array([-0.52103706,  1.11904511]),
array([-0.80961024,  0.50068852]), array([-0.73827208,  0.2096953 ]),
array([-1.20578537,  1.19021132]), array([-0.36378319,  0.17554214])], 1:
[array([3.50752233, 2.6993138 ]), array([4.29192581, 6.59250797]),
array([3.56619087, 7.17383233]), array([4.80815598, 4.80534012]),
array([1.58074851, 4.56706754]), array([2.00718797, 4.22277444]),
array([3.35739154, 6.92001259]), array([3.6604525 , 5.73706073]),
array([3.40148307, 4.19351521]), array([2.92595151, 5.16322159]),
array([4.45833583, 3.98333488]), array([2.88905394, 5.66858736]),
array([4.61548321, 5.41472741]), array([3.08219469, 6.56324233]),
array([3.22807915, 4.77474657]), array([4.20924569, 5.9327244 ]),
array([4.4787459 , 5.52145107]), array([4.01779082, 5.92532132]),
array([3.71211986, 4.43265775]), array([4.43956151, 3.97306561]),
array([4.81119332, 6.40069914]), array([3.45039959, 5.30238367]),
array([1.59903643, 3.86618863]), array([3.92764978, 6.42628439]),
array([5.28554342, 5.66626269]), array([2.90126409, 6.19995457]),
array([5.10671678, 4.99793313]), array([4.51808911, 4.62793958]),
array([3.88890452, 3.44002436]), array([5.14468414, 5.55728691]),
array([3.0749813, 6.6637202]), array([3.23715953, 4.35462184]), array([4.3423914
, 6.47522321]), array([5.77804351, 4.7055156 ]), array([2.90632196,
4.33154509]), array([3.10326381, 4.53088183]), array([3.80127866, 3.92783617]),
array([4.30978752, 6.22424252]), array([5.05838861, 5.1332605 ]),
array([4.32610657, 5.34955732]), array([5.18487428, 4.34018354]),
array([5.96882436, 3.98581601]), array([4.61100889, 3.461623  ]),
array([4.78403921, 5.854026  ]), array([3.51338894, 6.58356026]),
array([4.77966518, 6.13795151]), array([4.81271386, 4.24555198]),
array([3.72700442, 4.49685086]), array([2.87520143, 4.98004819]),
array([1.49913108, 4.29227149])]}
```

3. Calculate Mean, Std deviation and count for each column in a dataset

```python
[60]: def get_variables(class_dict):
        var_dict = {}

        ## Write your code here

        return var_dict
```

```
This is the Variable Dictionary :
{0: [(-0.08143074021616499, 0.9927104621843587, 50), (0.05650317153356914,
```

1.0255393627439435, 50)], 1: [(3.891293527338697, 1.0208299436699904, 50),
(5.136474984683965, 1.0354673362145788, 50)]}

3. Calculate Class Probabilities

```python
[61]: def calculate_probability(x,mean,stdev):
        exponent = exp(-((x-mean)**2 / (2 * stdev**2 )))
        return (1 / (sqrt(2 * pi) * stdev)) * exponent

      def calculate_class_probabilities(summaries,row):
        probabilities = dict()

        ## Write your code here to calculate the class probabilities
        '''
        You can use the above function (calculate_probability) to calculate␣
        ↪probability of an individual data point belonging to a particular class
        based on mean and std deviation of that class

        '''
        return probabilities
```

4. Test the model using some samples

```python
[54]: ## Test Data Generation

      mean1=np.array([0,0])
      mean2=np.array([4,5])
      var=np.array([[1,0.1],[0.1,1]])
      np.random.seed(0)
      data1=np.random.multivariate_normal(mean1,var,10)
      data2=np.random.multivariate_normal(mean2,var,10)
      test_data=np.concatenate((data1,data2))
      y_test=np.concatenate((np.zeros(data1.shape[0]),np.ones(data2.shape[0])))
      print('Test Data Size : ',test_data.shape[0])
      plt.figure()
      plt.scatter(data[:,0],data[:,1],c=label)
      plt.title('Data visualization')
```

Test Data Size : 20

```
[54]: Text(0.5, 1.0, 'Data visualization')
```

4

Data visualization
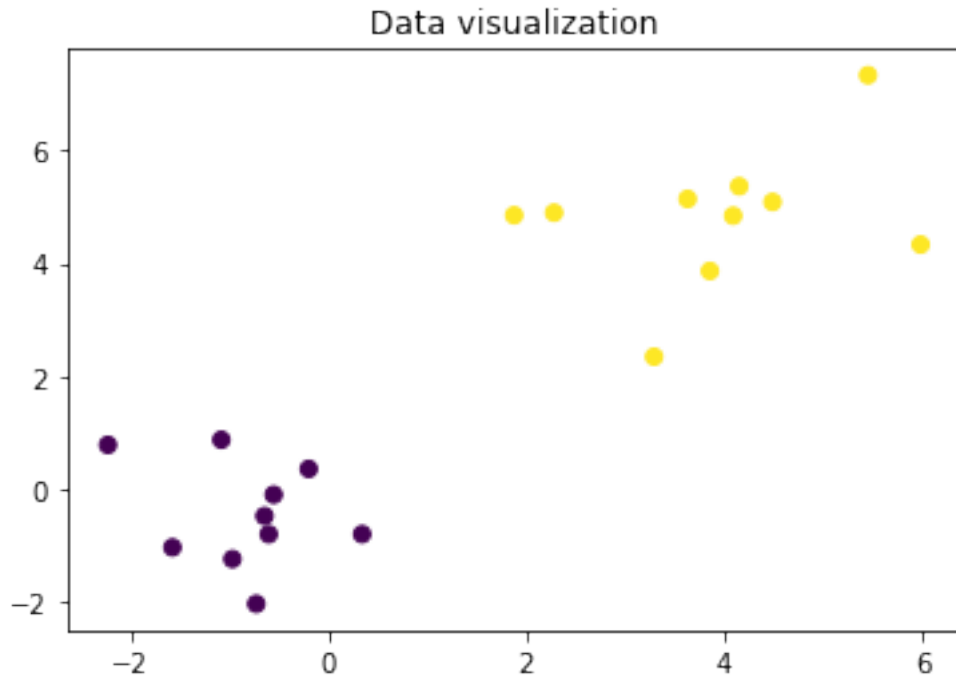
Testing for a sample point

```
[63]: class_dict = class_dictionary(data,label)
      var_dict = get_variables(class_dict)
      out = calculate_class_probabilities(var_dict,test_data[0])
      print('Class Probabilites for the first sample of test dataset : ')
      print(out)
```

```
Class Probabilites for the first sample of test dataset :
{0: 0.014197204018409897, 1: 8.332975086162198e-16}
```

As seen above the class probability for the 1st sample is given, we can observe that probability is higher for class 0 than 1 and hence imply that this datapoint belongs to class 0

Now Calculate the class probabilities for all the data points in the test dataset and calculate the accuracy by comparing the predicted labels with the true test labels

```
[ ]: ## Write your code here
```

5. Use the Sci-kit Learn library to perform Gaussian Naive Bayes classifier on the above dataset, also report the accuracy and confusion matrix for the same

```
[ ]: ## Write your code here
```

0.3 Sentiment Analysis using Naive Bayes Classifier

Go through the following article and implement the same

Keypoints :

5

1. The link to the dataset is given in the above article, download the same to perform sentiment analysis

2. Understanding how to deal with text data is very important since it requires a lot of preprocessing, you can go through this article if you are interested in learning more about it

3. Split the dataset into train-test and train the model

4. Report the accuracy metrics and try some sample prediction outside of those present in the dataset

Note : The goal of this experiment is to explore a practical use case of Naive bayes classifier as well as to understand how to deal with textual data, you can follow any other open source implemetations of sentiment analysis using naive bayes also

Other References :

1. https://towardsdatascience.com/sentiment-analysis-introduction-to-naive-bayes-algorithm-96831d77ac91

2. https://gist.github.com/CateGitau/6608912ca92733036c090676c61c13cd

```
## Write your code here
```