LAB 9 : Naïve Bayes Classifier

1. Binary Classification using Naïve Bayes Classifier
2. Sentiment Analysis using Naïve Bayes

Bayes' Theorem:

$$P(Y \mid X) = \frac{P(X \mid Y) \times P(Y)}{P(X)}$$

For our purposes, Y is class, X is data

Note:

- $P(Y)$ is prior probability of Y (before evidence, X, is seen)
- $P(Y|X)$ is the a posteriori probability of X, i.e. probability of the event, given the evidence X.
- Naïve Bayes assumes that the features of X are independent of each other.
- If event A is independent of event B, then $P(A \mid B, C) = P(A \mid C)$

Assume that $X = (x_1, x_2, \ldots, x_n)$. Then, since dimensions are assumed to be independent,

$$\begin{aligned} P(X|Y) &= P(x_1, x_2, \ldots, x_n \mid Y) \\ &= P(x_1 \mid Y)\, P(x_2 \mid x_1, Y) \;\ldots\; P(x_n \mid x_{n-1}, \ldots, x_1, Y) \\ &= P(x_1 \mid Y)\, P(x_2 \mid Y) \;\ldots\; P(x_n \mid Y) \end{aligned}$$

So, we have,

$$P(Y \mid X) = \frac{P(x_1 \mid Y)\, P(x_2 \mid Y) \;\ldots\; P(x_n \mid Y)\, P(Y)}{P(X)}$$

Since we compare P(Y|X) for different classes (declaring that the class Y having highest value for this as the predicted class), P(X) becomes like a scaling factor.

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

Binary Classification using Naïve 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/

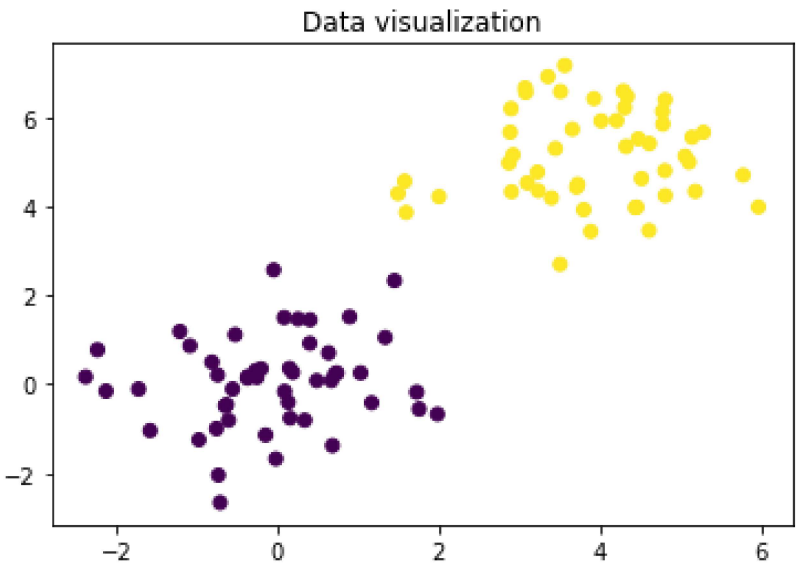
3. https://towardsdatascience.com/implementing-naive-bayes-algorithm-from-scratch-python-c6880cfc9c41

Note : The goal of this experiment is to perform and understand Naïve 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

In [ ]:
```
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')
```

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



1. Split the Dataset by Class Values (Create a Dictionary)

In [ ]:
```
def class_dictionary(data, label):
    class_dict = {}
    class_0 = []
    class_1 = []

    for ind, val in enumerate(label):
        if val == 0:
            class_0.append(data[ind])
        else:
            class_1.append(data[ind])

    class_dict = {0:class_0 , 1:class_1}
```

```
    return class_dict
```

In [ ]:
```
c = class_dictionary(data, label)
print(c)
```

```
{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([ 0.340
76762, -0.80512206]), array([1.45488724, 2.3318086 ]), array([-0.14322361, -1.13894
247]), 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, -0.1058270
7]), array([0.49004011, 0.08445841]), array([ 1.73020468, -0.1749493 ]), array([-0.
04321608,  2.57402376]), array([0.67183735, 0.084099  ]), array([0.40754152, 1.4506
5429]), 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]), array([0.71376003, 0.22709168]), array([0.73990561, 0.25751442]), arr
ay([ 1.761071  , -0.55498015]), array([ 0.1379401 , -0.40110559]), array([0.8985436
8, 1.51943123]), array([0.63802444, 0.70771651]), array([-0.6272324 , -0.4541836
7]), array([-0.01665582, -1.67334851]), array([ 0.16100003, -0.75776912]), array
([1.03410461, 0.25749629]), array([0.19337668, 0.2687304 ]), array([0.25980547, 1.4
6839103]), array([ 0.68519909, -1.3758881 ]), array([-2.37551849,  0.16808376]), ar
ray([-0.75350908, -0.99490358]), array([0.08674368, 1.50143912]), array([-0.5210370
6,  1.11904511]), array([-0.80961024,  0.50068852]), array([-0.73827208,  0.2096953
]), array([-1.20578537,  1.19021132]), array([-0.36378319,  0.17554214])], 1: [arra
y([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 , 5.7
3706073]), array([3.40148307, 4.19351521]), array([2.92595151, 5.16322159]), array
([4.45833583, 3.98333488]), array([2.88905394, 5.66858736]), array([4.61548321, 5.4
1472741]), array([3.08219469, 6.56324233]), array([3.22807915, 4.77474657]), array
([4.20924569, 5.9327244 ]), array([4.4787459 , 5.52145107]), array([4.01779082, 5.9
2532132]), array([3.71211986, 4.43265775]), array([4.43956151, 3.97306561]), array
([4.81119332, 6.40069914]), array([3.45039959, 5.30238367]), array([1.59903643, 3.8
6618863]), array([3.92764978, 6.42628439]), array([5.28554342, 5.66626269]), array
([2.90126409, 6.19995457]), array([5.10671678, 4.99793313]), array([4.51808911, 4.6
2793958]), array([3.88890452, 3.44002436]), array([5.14468414, 5.55728691]), array
([3.0749813, 6.6637202]), array([3.23715953, 4.35462184]), array([4.3423914 , 6.475
22321]), array([5.77804351, 4.7055156 ]), array([2.90632196, 4.33154509]), array
([3.10326381, 4.53088183]), array([3.80127866, 3.92783617]), array([4.30978752, 6.2
2424252]), array([5.05838861, 5.1332605 ]), array([4.32610657, 5.34955732]), array
([5.18487428, 4.34018354]), array([5.96882436, 3.98581601]), array([4.61100889, 3.4
61623  ]), array([4.78403921, 5.854026  ]), array([3.51338894, 6.58356026]), array
([4.77966518, 6.13795151]), array([4.81271386, 4.24555198]), array([3.72700442, 4.4
9685086]), array([2.87520143, 4.98004819]), array([1.49913108, 4.29227149])]}
```

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

In [ ]:
```
def get_variables(class_dict):
    var_dict = {}
    for label in class_dict:
        num_rec = len(class_dict[label])
        rec = np.array(class_dict[label])
        mu_list = []
        sig_list = []
        var_dict_entry = []
        var_dict[label] = []
        for i in range(rec.shape[1]):
            m = np.mean(rec[:,i])
```

```
            s = np.sqrt(np.sum([(fv[i]-m)**2 for fv in rec])/num_rec)
            mu_list.append(m)
            sig_list.append(s)
            var_dict_entry.append((m, s, num_rec))
        var_dict[label] = var_dict_entry
        # print("vde=",var_dict_entry)
    return var_dict
```

In [ ]:
```
print(get_variables(c))
```

```
{0: [(-0.08143074021616499, 0.9927104621843587, 50), (0.05650317153356914, 1.025539
3627439435, 50)], 1: [(3.891293527338697, 1.0208299436699904, 50), (5.1364749846839
65, 1.0354673362145788, 50)]}
```

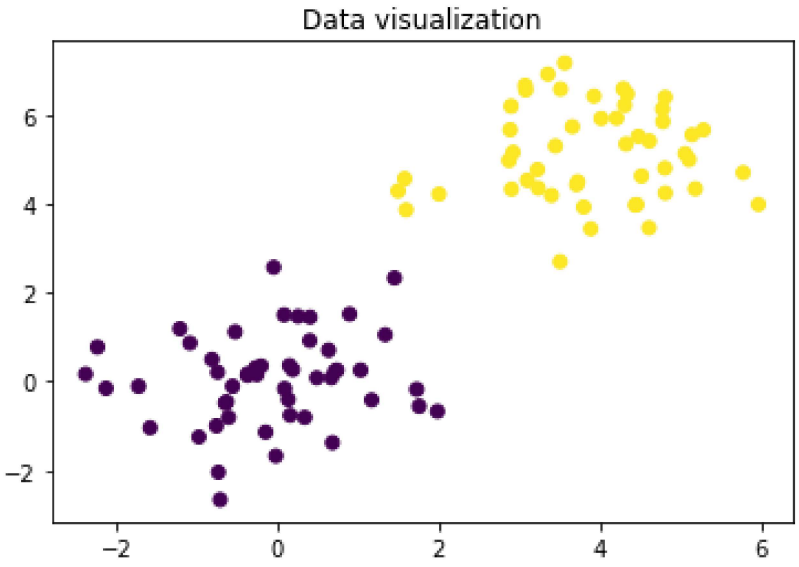
1. Calculate Class Probabilities

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

def calculate_class_probabilities(summaries, row):
  probabilities = dict()
  # summaries contains dictionary of the format label:[(mean of col1, stddev of col
  # {0:[(-0.08, 0.99, 50), (0.05, 1.02, 50)],
  #  1:[(3.89, 1.02, 50) , (5.13, 1.03, 50)]}

  total_rec = float(np.sum([summaries[i][0][2] for i in summaries])) # get number o

  for key, class_arr in summaries.items():
    # firstly, set P(k) to be fraction of records that are contained in 'k' class.
    probabilities[key] = summaries[key][0][2]/total_rec
    for i in range(len(class_arr)):
      mean, sig, count = class_arr[i]
      probabilities[key] *= calculate_probability(row[i], mean, sig) # for kth clas
    # in naive bayes, we assume independence of dimensions, so we simply multiply t

  return probabilities
```

1. Test the model using some samples

In [ ]:
```
## 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')
```

Out[ ]:
```
Test Data Size :  20
Text(0.5, 1.0, 'Data visualization')
```



Testing for a sample point from test_data

In [ ]:
```
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

In [ ]:
```
## Write your code here
pred_labels = list()

for i in range(test_data.shape[0]):
  row = test_data[i]
  out = calculate_class_probabilities(var_dict, row)
  if(out[0] > out[1]):
    pred_labels.append(0)
  else:
    pred_labels.append(1)
pred_labels = np.array(pred_labels)

count = 0
for i, y_te in enumerate(y_test):
  if(pred_labels[i] == y_te):
    count = count + 1
test_acc = count*100/test_data.shape[0]
print("Testing Accuracy:", test_acc)
```

```
Testing Accuracy: 100.0
```

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

In [ ]:
```
## Write your code here
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix

model = GaussianNB()
model.fit(data,label)
y_pred = model.predict(test_data)
test_acc = model.score(test_data, y_test)
print("Testing accuracy = ", test_acc*100)
print(confusion_matrix(y_test, y_pred))
```

```
Testing accuracy =  100.0
[[10  0]
 [ 0 10]]
```

Sentiment Analysis using Naïve Bayes Classifier

Go through the following article and implement the same

Keypoints :

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 Naïve 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 naïve bayes also

Other References :

1. https://towardsdatascience.com/sentiment-analysis-introduction-to-naive-bayes-algorithm-96831d77ac91
2. https://gist.github.com/CateGitau/6608912ca92733036c090676c61c13cd

In [ ]:
```
## Write your code here
import pandas as pd
from sklearn.model_selection import train_test_split
import joblib
from sklearn.feature_extraction.text import CountVectorizer

data = pd.read_csv('google_play_store_apps_reviews_training.csv')

print(data.head())
```

```
          package_name                                             review  \
0  com.facebook.katana   privacy at least put some option appear offli...
1  com.facebook.katana   messenger issues ever since the last update, ...
2  com.facebook.katana   profile any time my wife or anybody has more ...
3  com.facebook.katana   the new features suck for those of us who don...
4  com.facebook.katana   forced reload on uploading pic on replying co...

   polarity
0         0
1         0
2         0
3         0
4         0
```

In [ ]:

```python
def preprocess_data(data):
    # Remove package name as it's not relevant
    data = data.drop('package_name', axis=1)

    # Convert text to lowercase
    data['review'] = data['review'].str.strip().str.lower()
    return data

data = preprocess_data(data)
# Split into training and testing data
x = data['review']
y = data['polarity']
x_train, x_test, y_train, y_test = train_test_split(x, y, stratify=y, test_size=0.2

print("training set x = \n", x_train.head())
print("_____________________________________")
print("training set y = \n", y_train.head())
print("_____________________________________")
print("testing set x = \n", x_test.head())
print("_____________________________________")
print("testing set y = \n", y_test.head())
```

```
training set x =
 485    best app i can't believe that it is free! they...
622    good good for slow connection this uc minilite...
854    #1 great game. challenging to the point where ...
536    very reliable syncing but......the web ui look...
728    didn't work couldn't use it with my note 4, it...
Name: review, dtype: object

_________________________________________
training set y =
 485    1
622    1
854    1
536    0
728    0
Name: polarity, dtype: int64

_________________________________________
testing set x =
 243    well... i just switched to a new htc phone and...
174    the new theme is not compatible with my device...
809    good as far as i have it fast and reliable. fo...
435    keeps crashing ever since i started using it m...
147    suggestion. i given 5 stars to this game,  bec...
Name: review, dtype: object

_________________________________________
testing set y =
 243    0
174    0
809    1
435    0
147    1
Name: polarity, dtype: int64
```

In [ ]:
```
vec = CountVectorizer(stop_words='english')
x_train = vec.fit_transform(x_train).toarray()
x_test = vec.transform(x_test).toarray()
```

In [ ]:
```
from sklearn.naive_bayes import MultinomialNB

model = MultinomialNB()
model.fit(x_train, y_train)
```

Out[ ]:
```
MultinomialNB()
```

In [ ]:
```
model.score(x_test, y_test)
```

Out[ ]:
```
0.8547486033519553
```

In [ ]:
```
def pred(model, str):
    print(model.predict(vec.transform([str])))
```

In [ ]:
```
pred(model, "I love this app! Though I hate the ads.")
```

```
[1]
```

In [ ]:
```
pred(model, "interesting game!")
pred(model, "interesting concept")
```

```
[1]
[0]
```

In [ ]:
```
pred(model, "I could've made better use of my time") # no negative words, but corre
```

```
[0]
```

In [ ]:
```
pred(model, "this is like angry birds!")
```

```
[1]
```

In [ ]:
```
pred(model, "this inspired me to become a game developer")
```

```
[1]
```

In [ ]:
```

```