# Naïve Bayes Classifier

Naïve Bayes Classifier Unlike k-NN which makes predictions only based on local information, Naïve Bayes Classifier is a probabilistic framework that derives the class for unknown record based on the global distribution of data. The core rationale of Bayesian Classifier is to calculate the probability of unknown record belonging to certain class, given its other attributes. The class with largest probability is assigned to the unknown record. Such probability can be denoted as following

P(C|A1, A2, …, An), where,

* P(A|C) is the probability of hypothesis A given the event C, a posterior probability.
* P(C|A) is the probability of event C given that the hypothesis A is true.
* P(A) is the probability of hypothesis A being true (regardless of any related event), or prior probability of A.
* P(C) is the probability of the event occurring (regardless of the hypothesis).

According to Bayes theorem, It is equivalent to maximize P(A1, A2, …, An|C)P(C)in which P(A1, A2,…, An |C) = P(A1|C) P(A2|C) … P(An-1|C) P(An|C) P(C). P(C) is simply the frequency of class C in global scope. To calculate P(Ai|C) in which Ai corresponds to the values of song attributes. In our case, the class consists of top song and non-top songs. The songs were assigned to one of the classes according to the attribute values. However, Naïve Bayes Classifier may not be so accurate for this dataset, because independence among different attributes cannot be proved.

# Algorithm: Gaussian Naive Bayes

A Gaussian Naive Bayes algorithm is a special type of NB algorithm. It is specifically used when the features contain continuous values other than categorical values. It is also assumed that all the features are following a gaussian distribution i.e. normal distribution.

# Implementation

## Importing Python Machine Learning Libraries

Python 3 together with Scikit-Learn are used to implement the Naive Bayes Classifier.

# To model the Gaussian Navie Bayes classifier

from sklearn.naive\_bayes import GaussianNB

import pandas as pd

## Data Slicing

We can easily perform the data slicing using sklearn’s train\_test\_split() method which randomly slices the data into training and test set

from sklearn.model\_selection import train\_test\_split

songs\_train, songs\_test = train\_test\_split(concated\_songs,

test\_size = 0.20,

random\_state = 10)

Using above code snippet, we have divided the data into training and test set, with 80% of the data used for training and 20% used for testing.

Furthermore, to improve the accuracy and data utilization, we incorporated the k-fold cross validation process to randomly partition the data set into *5* equal sized subsamples.

Of the 5 subsamples, a single subsample is retained as the validation data for testing the model, and the remaining k – 1 = 4 subsamples are used as training data. The cross-validation process is then repeated *5* times, with each of the *5* subsamples used exactly once as the validation data, such that each data is utilized for both training and testing.

concated\_songs = pd.concat([top,non\_top])

concated\_songs = shuffle(concated\_songs)

for i in range (0,5):

songs\_train1 = concated\_songs[0:120\*(i):]

songs\_train2 = concated\_songs[120\*(i+1):600:]

songs\_train = pd.concat([songs\_train1,songs\_train2])

songs\_test = concated\_songs[120\*(i):120\*(i+1):]

## Training

Training process involves finding a model for class attributes as a function of the class label. It sets the probability for each class label and counts the total times a concept occurs in the training set for each class label. Then the probability of a concept can be obtained by a simple division. After that, the conditional probability for each attribute under each concept is calculated.

gnb.fit(songs\_train[used\_features].values,

songs\_train["top"].values)

Using the above code snippets which makes use of the fit() function of the GaussianNB library, we conducted supervised training for the training data against the target value, i.e. Class 1 = top song, Class 0 = non-top song.

## Testing

The other important process in classification is to assign a class label to an unseen record. This process finds a class label for an input instance. It calculates the probability for each

testing instance under each class label. The probability is obtained by (1) multiply together each attribute’s probability with a specific value in an instance for the class label and then (2) multiply the probability of the class label. After that, the class label with largest probability will be assigned to the instance.

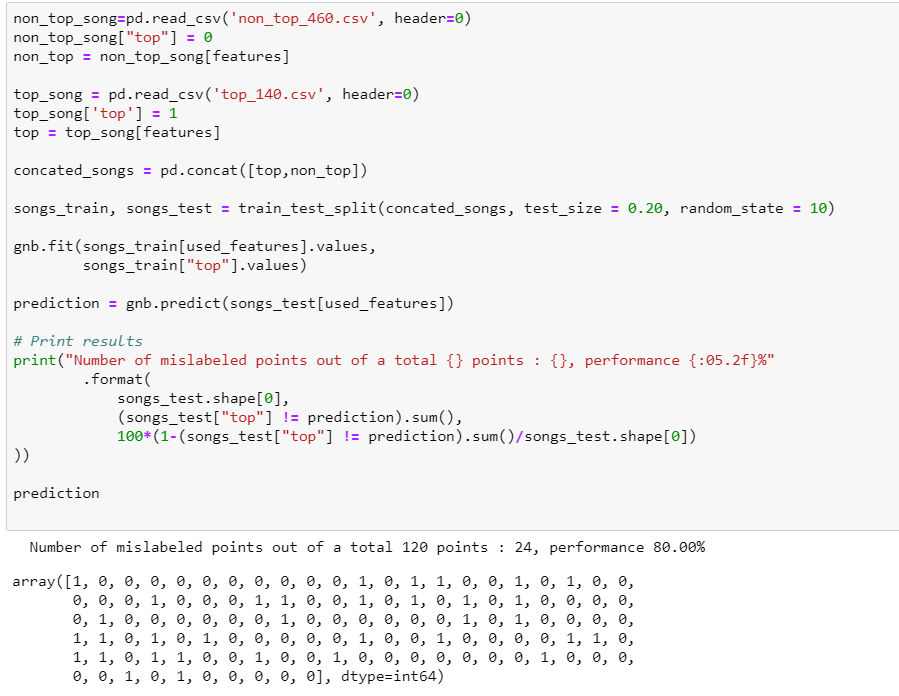
prediction = gnb.predict(songs\_test[used\_features])

Using the above code snippets which makes use of the predict() function of the GaussianNB library, we assign the respective class label to the testing data so as to predict whether the input song is a top song or a non-top song.

# Results

1. Results of *Naïve Bayes Classifier* using the train\_test\_split data slicing method:

Number of mislabeled points out of a total 120 points: 24, performance 80.00%



*Figure 1 – Implementation and results of Naïve Bayes Classifier using train\_test\_split data slicing*

1. Results of Naïve Bayes Classifier using 5-fold cross validation:

For the five iterations:

Number of mislabeled points out of a total 120 points: 19, performance 84.17%

Number of mislabeled points out of a total 120 points: 14, performance 88.33%

Number of mislabeled points out of a total 120 points: 28, performance 76.67%

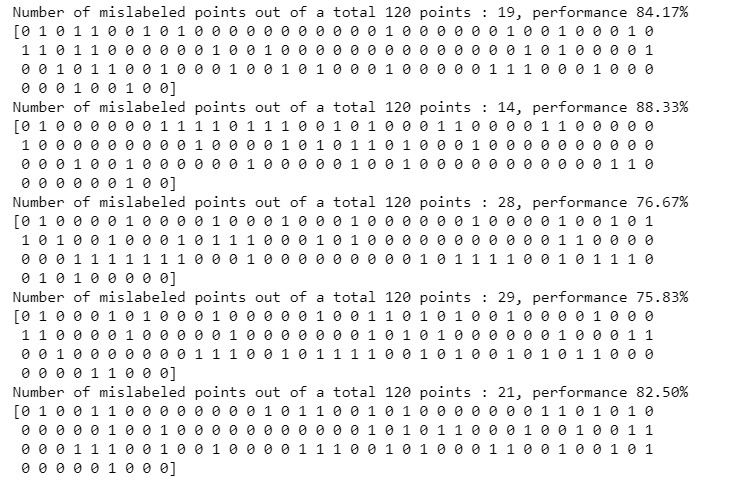
Number of mislabeled points out of a total 120 points: 29, performance 75.83%

Number of mislabeled points out of a total 120 points: 21, performance 82.50%

An average of 81.67% accuracy is obtained.



*Figure 2 – Implementation of Naïve Bayes Classifier using train\_test\_split data slicing*



*Figure 3 –Results of Naïve Bayes Classifier using train\_test\_split data slicing*

# Analysis

## Confusion Matrix and Measurement

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Predict Class | | | | |
| Actual Class |  | Class = 1 | Class = 0 | Weight Average |
| Class = 1 | 125 | 15 | 23.3% |
| Class = 0 | 95 | 365 | 76.6% |
| Accuracy | 81.67% | | | |
| Precision | 0.568 (Top=1); 0.960 (non-top=0); 0.791 (Weight Average) | | | |
| Recall | 0.893 (Top=1); 0.793 (non-top=0); 0.809 (Weight Average) | | | |
| Cost | 125\*0 + 15\*50 + 95\*50 + 365\*0 = 5500 | | | |

Naïve Bayes algorithm identified instances correctly at a rate of 81.67% within 0.04 second. This implies that the selected attributes are independent enough to have a decent level of accuracy but still have some dependences.

The inaccuracy may be caused by that the assumption of the independence of all the attributes may not hold. For example, from the results of the attributes’ inter-relationship heatmap, the loudness and the energy level are positively correlated.