Naive Bayes Classifier in Python

We have a data which classified if patients have heart disease or not according to features in it. We will try to use this data to create a model which tries predict if a patient has this disease or not.

What is Naive Bayes algorithm?

Naive Bayes is a classification technique based on Bayes' Theorem(*Probability theory*) with an assumption that all the features that predicts the target value are independent of each other. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature in determining the target value.

This assumption we just read about is very **Naive** when we are dealing with real world data because most of the times, features do depend on each other in determining the target .

> Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

Bayes theorem provides a way of calculating posterior probability P(c|x) - *(read as Probability of **c** given **x**)*, from P(c), P(x) and P(x|c). Look at the equation below:

$$\mathbf{P}(x \mid c) = \frac{\mathbf{P}(c \mid x)\,\mathbf{P}(c)}{\mathbf{P}(x)}$$

In the *above* equation,

- P(c|x) is the posterior probability of class (c, target) given predictor (x, attributes).
- P(c) is the prior probability of class **c**.
- P(x|c) is the likelihood which is the probability of predictor(the query **x**) given class.
- P(x) is the prior probability of predictor **x**.

2.Gaussian Naïve Bayes algorithm*

When we have continuous attribute values, we made an assumption that the values associated with each class are distributed according to Gaussian or Normal distribution. For example, suppose the training data contains a continuous attribute x. We first segment the data by the class, and then compute the mean and variance of x in each class. Let μi be the mean of the values and let σi be the variance of the values associated with the ith class. Suppose we have some observation value xi . Then, the probability distribution of xi given a class can be computed by the following equation –

$$p(x_i|y_j) = \frac{1}{\sqrt{2\pi\sigma_j^2}} e^{-\frac{(x_i-\mu_j)^2}{2\sigma_j^2}}$$

.

Data contains;

- age - age in years
- sex - (1 = male; 0 = female)
- cp - chest pain type
- trestbps - resting blood pressure (in mm Hg on admission to the hospital)
- chol - serum cholestoral in mg/dl
- fbs - (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
- restecg - resting electrocardiographic results
- thalach - maximum heart rate achieved
- exang - exercise induced angina (1 = yes; 0 = no)
- oldpeak - ST depression induced by exercise relative to rest
- slope - the slope of the peak exercise ST segment
- ca - number of major vessels (0-3) colored by flourosopy
- thal - 3 = normal; 6 = fixed defect; 7 = reversable defect
- output - have disease or not (1=yes, 0=no)

if the assumption of independence does not hold in the data, the Naive Bayes classifier may not perform well and other classification methods may be more appropriate.

Normalize Data

$$X_{changed} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Normalization helps ensure that the Naive Bayes algorithm(Any ML algorithm) is not biased towards any particular feature due to its scale. This can lead to more accurate predictions and a better overall performance of the algorithm. feature A ranges from 0 to 1, while feature B ranges from 0 to 1000. If we were to train a Naive Bayes model on this dataset without normalization, feature B would have a much larger impact on the model due to its larger magnitude, regardless of its actual contribution to the target variable. Normalizing the input data to the same range of values using min-max scaling would mitigate this issue, allowing the model to weigh each feature more fairly.

X: This variable contains the independent variables used to predict the output variable. In this dataset, X contains the following features: age, sex, chest pain type, resting blood pressure, cholesterol, fasting blood sugar, resting electrocardiographic results, maximum heart rate achieved, exercise induced angina, oldpeak, the slope of the peak exercise ST segment, number of major vessels colored by flourosopy, and thalassemia.

y: This variable contains the dependent variable which is to be predicted using the input features. In this dataset, y contains the output variable 'output', which indicates whether a patient has heart disease or not. '1' indicates the presence of heart disease and '0' indicates the absence of heart disease.

An accuracy score of 0.85 means that the Naive Bayes classifier correctly predicted the class label for 85% of the instances in the test dataset. In other words, out of all the instances in the test dataset, 85% were classified correctly by the classifier.

The accuracy score is a commonly used evaluation metric for classification models. It is calculated by dividing the number of correctly classified instances by the total number of instances in the test dataset.

While an accuracy score of 0.85 may seem high, it is important to consider the context in which the classifier is being used. In some applications, an accuracy score of 0.85 may be sufficient, while in others, a higher accuracy score may be required. Additionally, the accuracy score should always be interpreted in conjunction with other evaluation metrics, such as precision, recall, and F1 score, to get a more comprehensive understanding of the classifier's performance.

It is also worth noting that the Naive Bayes classifier assumes independence between the features, which may not always be true in real-world datasets. In such cases, other classification algorithms that do not make the same assumption, such as logistic regression or decision trees, may be more appropriate.

Confusion Matrix

A confusion matrix is a table that is used to evaluate the performance of a classification model. It is often used in supervised learning to understand how well a model is classifying instances into correct or incorrect categories. The matrix shows the true positives, true negatives, false positives, and false negatives of a model's predictions. The diagonal elements represent the correctly classified instances, while the off-diagonal elements represent the incorrectly classified instances. The number of true positives and true negatives gives the count of correctly classified instances, while the number of false positives and false negatives gives the count of incorrectly classified instances.

The four cells of the confusion matrix of naive bayes classifier represent the following:

True Positive (TP): The number of observations that are correctly classified as positive (belonging to the positive class) which is 21 here False Positive (FP): The number of observations that are incorrectly classified as positive (belonging to the positive class, but actually belonging to the negative class) which is 6 here False Negative (FN): The number of observations that are incorrectly classified as negative (belonging to the negative class, but actually belonging to the positive class) which is 3 here True Negative (TN): The number of observations that are correctly classified as negative (belonging to the negative class) which is 31 here.

LDA

Linear Discriminant Analysis (LDA) is a supervised learning algorithm used in machine learning for classification tasks. LDA is a statistical method that makes assumptions about the distribution of input data and aims to find a linear combination of input features that maximizes the separation between classes while minimizing the within-class variance.

LDA assumes that the input features have a multivariate normal distribution with equal covariance matrices for each class. The algorithm finds the projection of the input features onto a lower-dimensional space that maximizes the separation between the classes, while also minimizing the within-class variance. This projection can be used to classify new data points by projecting them onto the same lower-dimensional space and assigning them to the class with the closest centroid.

The four cells of the confusion matrix of lda represent the following:

True Positive (TP): The number of observations that are correctly classified as positive (belonging to the positive class) which is 20 here False Positive (FP): The number of observations that are incorrectly classified as positive (belonging to the positive class, but actually belonging to the negative class) which is 7 here False Negative (FN): The number of observations that are incorrectly classified as negative (belonging to the negative class, but actually belonging to the positive class) which is 3 here True Negative (TN): The number of observations that are correctly classified as negative (belonging to the negative class) which is 31 here.

Based on the given evaluation metrics, the Naive Bayes classifier has a higher F-score (0.873) than the Linear Discriminant Analysis (LDA) classifier (0.861). The F-score is a harmonic mean of precision and recall, which takes both measures into account and gives a more balanced evaluation of the classifier's performance.

The Naive Bayes classifier also has a higher precision (0.838) than the LDA classifier (0.816), indicating that the Naive Bayes classifier produces fewer false positives (i.e., instances classified as positive but actually belonging to the negative class).

However, both classifiers have the same recall (0.912), which indicates that they both correctly identify a high proportion of positive instances in the dataset.

Overall, based on these evaluation metrics, the Naive Bayes classifier appears to be performing slightly better than the LDA classifier. However, it is important to note that the choice of the best classifier depends on various factors, such as the nature of the problem, the characteristics of the dataset, and the performance requirements of the application

QDA

Quadratic Discriminant Analysis (QDA) is a statistical method used in machine learning for classification tasks. Like LDA, QDA assumes that the input features have a multivariate normal distribution, but unlike LDA, it does not assume that the covariance matrices for each class are equal.

QDA aims to find a quadratic boundary that separates the classes in the input data. To do this, QDA calculates the mean and covariance matrices for each class in the dataset, just like LDA. However, instead of assuming that the covariance matrices are equal, QDA allows each class to have its own covariance matrix.

We can see that Naive Bayes performed slightly better than LDA and QDA in classifying the data. However, the difference between the accuracies is not very significant, so it could also be that the performance difference is due to random variation in the data or model fitting process.

It is also important to note that the accuracy score alone may not be sufficient to fully evaluate the performance of a model. It is recommended to also look at other metrics such as precision, recall, and F1 score to gain a more comprehensive understanding of the model's performance.