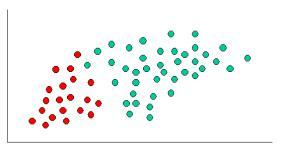
# ASSIGNMENTNO. 4

**AIM:**  Assignment on Naïve Bayes.

**PREREQUISITE**: Python programming

**THEORY:**

The Naive Bayes Classifier technique is based on the so-called Bayesian theorem and is particularly suited when the dimensionality of the inputs is high. Despite its simplicity, Naive Bayes can often outperform more sophisticated classification methods.

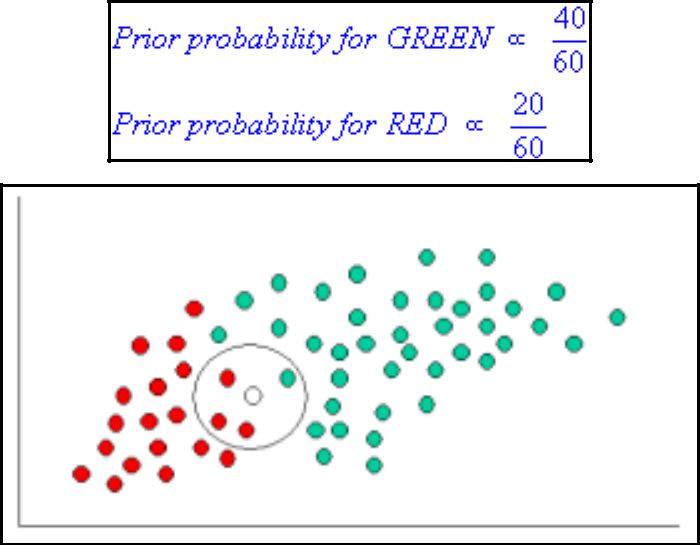


To demonstrate the concept of Naïve Bayes Classification, consider the example displayed in the illustration above. As indicated, the objects can be classified as either GREEN or RED. Our task is to classify new cases as they arrive, i.e., decide to which class label they belong, based on the currently exiting objects.

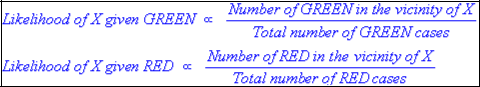
Since there are twice as many GREEN objects as RED, it is reasonable to believe that a new case (which hasn't been observed yet) is twice as likely to have membership GREEN rather than RED. In the Bayesian analysis, this belief is known as the prior probability. Prior probabilities are based on previous experience, in this case the percentage of GREEN and RED objects, and often used to predict outcomes before they actually happen.

Thus, we can write:

Since there is a total of 60 objects, 40 of which are GREEN and 20 RED, our prior probabilities for class membership are:



Having formulated our prior probability, we are now ready to classify a new object (WHITE circle). Since the objects are well clustered, it is reasonable to assume that the more GREEN (or RED) objects in the vicinity of X, the more likely that the new cases belong to that particular color. To measure this likelihood, we draw a circle around X which encompasses a number (to be chosen a priori) of points irrespective of their class labels. Then we calculate the number of points in the circle belonging to each class label. From this we calculate the likelihood:

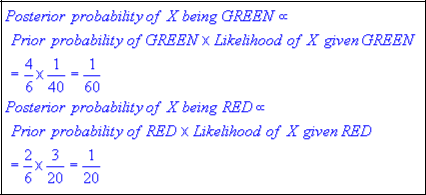


From the illustration above, it is clear that Likelihood of X given GREEN is smaller than

Likelihood of X given RED, since the circle encompasses 1 GREEN object and 3 RED ones. Thus:



Although the prior probabilities indicate that X may belong to GREEN (given that there are twice as many GREEN compared to RED) the likelihood indicates otherwise; that the class membership of X is RED (given that there are more RED objects in the vicinity of X than GREEN). In the Bayesian analysis, the final classification is produced by combining both sources of information, i.e., the prior and the likelihood, to form a posterior probability using the so-called Bayes' rule (named after Rev. Thomas Bayes 1702-1761).



Finally, we classify X as RED since its class membership achieves the largest posterior probability.

**Note.** The above probabilities are not normalized. However, this does not affect the classification outcome since their normalizing constants are the same.

Technical Naïve Bayes

In the previous section, we provided an intuitive example for understanding classification using Naive Bayes. In this section are further details of the technical issues involved. Naive Bayes classifiers can handle an arbitrary number of independent variables whether continuous or categorical. Given a set of variables, X = {x1,x2,x...,xd}, we want to construct the posterior probability for the event Cj among a set of possible outcomes C = {c1,c2,c...,cd}. In a more familiar language, X is the predictors and C is the set of categorical levels present in the dependent variable. Using Bayes' rule:



where p(Cj | x1,x2,x...,xd) is the posterior probability of class membership, i.e., the probability that X belongs to Cj. Since Naive Bayes assumes that the conditional probabilities of the independent variables are statistically independent we can decompose the likelihood to a product of terms:

and rewrite the posterior as:



Using Bayes' rule above, we label a new case X with a class level Cj that achieves the highest posterior probability.

Although the assumption that the predictor (independent) variables are independent is not always accurate, it does simplify the classification task dramatically, since it allows the class conditional densities p(xk | Cj) to be calculated separately for each variable, i.e., it reduces a multidimensional task to a number of one-dimensional ones. In effect, Naive Bayes reduces a high-dimensional density estimation task to a one-dimensional kernel density estimation. Furthermore, the assumption does not seem to greatly affect the posterior probabilities, especially in regions near decision boundaries, thus, leaving the classification task unaffected.

When to Use Naive Bayes:

1. **Text Classification:**
   1. Spam email detection, sentiment analysis, and topic categorization often use Naive Bayes, especially when the data consists of words or features that are conditionally independent.
   2. For example, in Spam classification, the presence of certain words (e.g., "buy now", "limited offer") can be independent of other words in the email.
2. **Multinomial or Bernoulli Data:**
   1. Naive Bayes works particularly well when the input features are either discrete or count-based (like word counts in text classification). For example, Multinomial Naive Bayes is suitable for text (word frequency) data, while Bernoulli Naive Bayes is good for binary features.
3. **When the Features are Independent (or Nearly Independent):**
   1. If you suspect that the features are independent of one another, Naive Bayes can be an ideal choice, as it is built upon the assumption of independence between features given the class label.
4. **Small to Medium-Sized Datasets:**
   1. It performs efficiently on small datasets and provides a baseline for performance. It's a good starting point when you have limited data and want a quick, interpretable model.
5. **Real-time Prediction:**
   1. Naive Bayes is fast, requiring very little computation, and can make predictions in real-time. It’s ideal when you need to classify samples quickly.

Advantages of Naive Bayes:

1. **Simple and Fast:**
   * Naive Bayes is easy to implement and computationally efficient. It performs well on large datasets because it calculates probabilities quickly.
   * The algorithm is relatively fast to train and makes fast predictions due to the simplicity of the calculations involved.
2. **Works Well with High-Dimensional Data:**
   * Naive Bayes often works very well with high-dimensional data like text (e.g., bag-of-words models). Even if the number of features is large, it still performs well, especially for sparse data.
3. **Less Data Preprocessing:**
   * Naive Bayes requires less preprocessing compared to other algorithms. It doesn't require feature scaling (e.g., normalization or standardization) because the model relies on the probability of each feature.
4. **Works Well for Text and Categorical Data:**
   * The model is highly effective for categorical data, particularly when the features are conditionally independent (which is often a reasonable assumption for text features).
5. **Handles Missing Data Well:**
   * Since Naive Bayes uses probabilities, missing data can be handled by simply skipping missing values for that feature during the calculation.
6. **Good Baseline Model:**
   * It's a great choice as a baseline model due to its simplicity and interpretability. If Naive Bayes performs reasonably well, you can use it directly. If not, you can explore more sophisticated models.

Disadvantages of Naive Bayes:

1. **Assumption of Feature Independence:**
   * The key limitation is the assumption that the features are conditionally independent given the class label. In many real-world applications, this assumption is often violated (e.g., the features might have strong dependencies), which can reduce the model’s accuracy.
2. **Poor Performance with Highly Correlated Features:**
   * When the features are highly correlated, Naive Bayes doesn't perform as well because it assumes independence between the features. In such cases, other models like decision trees or SVM might perform better.
3. **Not Suitable for Regression:**
   * Naive Bayes is designed for classification problems. While extensions exist, it's not commonly used for regression tasks, as it models the class probabilities rather than predicting continuous values.
4. **Requires Large Amount of Data for Accurate Probability Estimation:**
   * Naive Bayes requires a significant amount of data to estimate the probabilities for each class and feature. If the dataset is very small or unbalanced, the model might give inaccurate predictions.
5. **Can't Model Complex Relationships Between Features:**
   * Naive Bayes assumes that each feature contributes independently to the decision boundary. This can be a problem when there are complex interactions or dependencies between the features that the model cannot capture.
6. **Zero Probability Problem:**
   * If a category (or feature value) appears in the test data but not in the training data, the probability of that category becomes zero, which would invalidate the prediction. This can be handled by smoothing techniques like Laplace Smoothing but still remains a potential limitation.

**REFERENCES:**

1. Coursera Course on “What is Data Science?” offered by IBM. Available at https://www.coursera.org/learn/what-is-datascience?specialization=ibm-data-science

2. https://www.ibm.com/in-en/topics/decision-trees

**CONCLUSION:**

We have understood and implemented the Naïve Bayes Algorithm.