Add title

Naive Bayes Classifier

The Naive Bayes Classifier (NBC) is a popular machine learning algorithm used for classification tasks. It's called "naive" because it makes some simplifying assumptions about the data, but that doesn't mean it's not powerful.

The NBC is a conditional probability model based on Bayes' theorem that assumes all features are conditionally independent of one another given the output class. This independence assumption vastly reduces the number of parameters that need to be estimated and enables the model to be trained with less data (1). Despite its simplicity and the strong feature independence assumption rarely being strictly valid in real-world data, Naive Bayes tends to perform surprisingly well across a variety of applications and has several advantages that make it a popular baseline classifier for NLP, text classification, and more (2).

Baye's Theorem:

Bayes' Theorem is the cornerstone of NBC. It describes the probability of an event based on prior knowledge of conditions that might be related to the event. Mathematically, it is expressed as:

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

Where:

- P(A|B) is the posterior probability: Probability of hypothesis A given the data B.
- P(B|A) is the likelihood: Probability of the data given the hypothesis.
- P(A) is the prior probability: Probability of hypothesis before observing the data.
- P(B) is the marginal probability: Total probability of the data.

The 'Naive' Assumption:

The term 'naive' in Naive Bayes comes from the assumption that the features (predictors) in the dataset are mutually independent. This means the presence (or absence) of one feature

does not affect the presence (or absence) of any other feature. The computations are more practical as a result of this simplification, particularly for huge datasets.

Types of Naive Bayes classifiers

There are different types of Naive Bayes classifiers, each tailored to different kinds of data distributions.

- **Gaussian Naive Bayes**: Ideal for data following normal (Gaussian) distributions, this version is particularly effective with continuous variables. It operates by calculating the mean and standard deviation for each class in the dataset. (6)
- **Multinomial Naive Bayes**: Best suited for features following multinomial distributions, this classifier excels with discrete data. It's commonly used in text analysis, such as for sorting emails into spam and non-spam categories, by working with frequency counts of words or features. (7)
- **Bernoulli Naive Bayes**: Designed for binary or Boolean data (think variables that are either 0/1, true/false), this classifier handles datasets where the features are dichotomous. (8)

Applications

- **Spam filtering**: Spam classification is one of the most popular applications of Naïve Bayes cited in literature. (3)
- **Document classification**: Document and text classification go hand in hand. Another popular use case of Naïve Bayes is content classification. (4)
- **Sentiment analysis**: While this is another form of text classification, sentiment analysis is commonly leveraged within marketing to better understand and quantify opinions and attitudes around specific products and brands.
- **Mental state predictions**: Using fMRI data, naïve bayes has been leveraged to predict different cognitive states among humans. (5)

Classification Process:

In a classification task, NBC calculates the probability of each class given a set of inputs (features). The class with the highest probability is the output of the prediction.

For a set of features $\mathbf{X}=(x_1,x_2,...,x_n)$ and a class variable C, the classifier predicts the class $\mathbf{C}_k$ for which the following is maximum:

$$P(C_k|X) = \frac{P(X|C_k)\times P(C_k)}{P(X)}$$

Since P(X) is constant for all classes, we only need to maximize $P(X|C_k) \times P(C_k)$.

OBJECTIVE:

The main aim of this blog is to build a Naive Bayes classifier to to find whether an essay is written by human or by an LLM. To design this I am going to use the data-set from LLM - Detect AI Generated Text competition in Kaggle. The data set consists of set of human written essays and LLM generated essay.

Naive Bayes provides a way to use probability to classify data points to a class.

P(class|data) = (P(data|class) * P(class)) / P(data)

Where P(class|data) is the probability of class given the provided data.

IMPLEMENTATION:

The steps involved in building Naive Bayes classifier are:

1. Clean the text
2. Normalize the text by converting it top lowercase
3. Split the data to training and testing
4. Separate the train data by classes
5. Calculate probability of data by class. Ex: If we have 4 line belonging to +ve class and 2 lines belonging to -ve class then individual class probabilities would be P(+ve) = 4/6 and P(-ve) = 2/6.

6. Tokenization:

Tokenization involves breaking down the text into individual words or tokens. Each token represents a distinct unit of meaning and is essential for further analysis.

7. Build Vocabulary:

Create a vocabulary by compiling all unique tokens from the training data. This step is crucial for calculating probabilities later in the process.

8. Count Occurrences:

Count the occurrences of each token for each class in the training data. This information is used to calculate the likelihood of each word given a specific class.

9. Calculate Class Priors:

Determine the prior probability of each class by dividing the number of documents belonging to each class by the total number of documents in the training set.

10. Calculate Likelihoods:

For each token in the vocabulary, calculate the likelihood of that token given each class. This involves computing the probability of a token occurring in documents of a particular class.

11. Smooth the Model (Optional):

To handle the issue of zero probabilities for unseen tokens, some form of smoothing may be applied, such as Laplace smoothing.

12. Make Predictions:

Apply the trained Naive Bayes model to the testing data. Calculate the probability of each class for a given document and assign the class with the highest probability as the predicted class. By following these steps, you can successfully build, train, and evaluate a Naive Bayes classifier for text classification tasks.

WORKING CODE:

I have used Kaggle data set to train my model. Since the data for llm generated is very less I have created sample essays using ChatGPT and and used it to train. My model have achieved 57% accuracy on test dataset .

Link to the notebook : https://www.kaggle.com/code/deepthipuvvada/essay-classifier

References:

(1) https://www.cs.cmu.edu/~tom/mlbook/NBayesLogReg.pdf

(2) https://www.amazon.science/blog/naive-bayes-machine-learning

3) O\'Neil, C. and Schutt, R. (2014) Doing data science Cathy O'Neil and Rachel Schutt. Sebastapol, CA: O'Reilly.

4) Mosteller, Frederick, and David L. Wallace. "Inference in an Authorship Problem." Journal of the American Statistical Association 58, no. 302 (1963): 275–309. https://doi.org/10.2307/2283270.

5) Mitchell, Tom & Hutchinson, Rebecca & Just, Marcel & Newman, Sharlene & Niculescu, Stefan & Pereira, Francisco & Wang, Xuerui. (2003). Machine learning of fMRI virtual sensors

of cognitive states.

6) Rennie, J. D., Shih, L., Teevan, J., & Karger, D. R. (2003). Tackling the poor assumptions of naive bayes text classifiers. In Proceedings of the 20th international conference on machine learning (ICML-03) (pp. 616-623).

7) McCallum, A., & Nigam, K. (1998). A comparison of event models for Naive Bayes text classification. In AAAI-98 workshop on learning for text categorization (Vol. 752, pp. 41-48).

8) Kibriya, A. M., Frank, E., Pfahringer, B., & Holmes, G. (2004, January). Multinomial naive bayes for text categorization revisited. In Australasian Joint Conference on Artificial Intelligence (pp. 488-499). Springer, Berlin, Heidelberg.