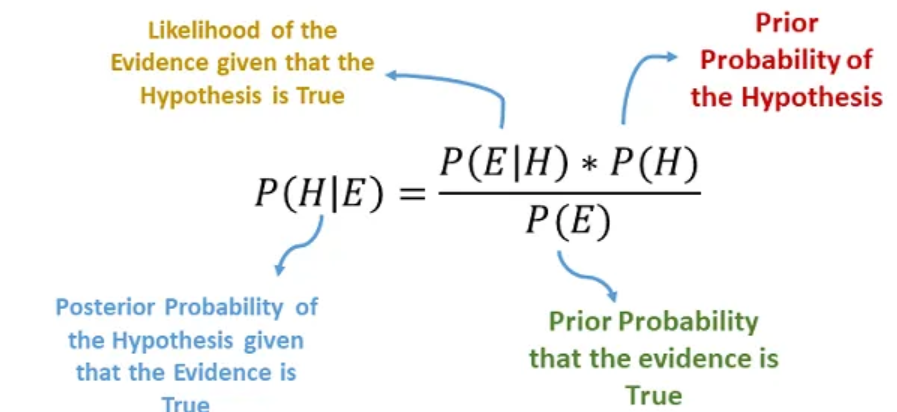
**Naïve Bayes**:   
The Naïve Bayes classifier is a supervised machine learning algorithm that is used for classification tasks such as text classification. They use principles of probability to perform classification tasks.  
Naive Bayes is a family of simple, yet powerful probabilistic classifiers based on Bayes' Theorem with an assumption of independence among predictors. Despite the simplicity of this assumption, Naive Bayes classifiers often perform surprisingly well in various applications, particularly in text classification and spam filtering.

Here’s a detailed explanation:



**Types of Naive Bayes Classifiers**

1. \*\***Gaussian Naive Bayes**\*\*: Assumes that the continuous features follow a Gaussian (normal) distribution. Likelihood of evidence is gaussian distribution.

2. \*\***Multinomial Naive Bayes**\*\*: Typically used for discrete counts, such as word frequencies in text classification. Best suited for Balanced Dataset.

3. \*\***Bernoulli Naive Bayes**\*\*: Used for binary/Boolean features, such as the presence or absence of words.

4. \*\***Categorical** **Naïve Bayes**\*\*: Here each feature are assumed as categorical.

5. \*\***Complement Naïve Bayes**\*\*: Similar to Multinomial Naïve Bayes, this is used text classification. Mainly used for Imbalanced Datasets.

**Steps in Naive Bayes Classification**

1. \*\***Training Phase**\*\*: - Calculate the prior probabilities for each class. - Calculate the likelihood of each feature given each class.

2. \*\***Prediction Phase\***\*: - Compute the posterior probability for each class using the features of the new instance. - Assign the class with the highest posterior probability to the new instance.

**Advantages**

\*\*Simplicity\*\*: Easy to implement and computationally efficient.

\*\*Scalability\*\*: Works well with large datasets. - \*\*Performs Well with High-Dimensional Data\*\*: Especially effective for text classification.

**Disadvantages**

\*\*Independence Assumption\*\*: The assumption of feature independence is rarely true in real-world data, which can affect the classifier's performance.

\*\*Zero Frequency Problem\*\*: If a categorical variable has a category not present in the training data, the probability estimation will be zero. This can be mitigated by techniques such as Laplace smoothing.

**Applications**

\*\*Spam Filtering\*\*: Classifies emails as spam or not spam.

\*\*Text Classification\*\*: Categorizes documents into predefined categories.

\*\*Sentiment Analysis\*\*: Determines the sentiment of text data, such as customer reviews.

Python Script to Naïve Bayes:

Calling the **MultinomialNB, Complement NB** with **classification report** and **confusion matrix** in Python:



Clf is just a variable to call the algorithm. Fit method to standardize the data.



A black text on a white background

Description automatically generated

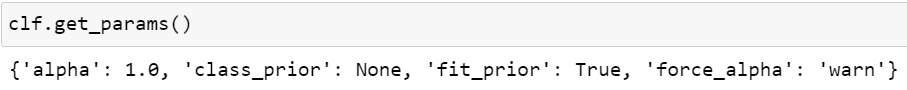
Here **fit\_transform** is used only for **train Dataset**. **Fit** will create the vocabulary and statistics from the training data. **Transform** converts the training data into matrix of features based on the learned vocabulary and statistics.

For **Test Dataset only transform** is used. It uses the vocabulary and statistics learned during the **fit** step to convert the new data into a matrix of features. It does not learn anything new from the test data; it only transforms it based on what was learned from the training data.

**Important Note**:

Assume while training the data there is no word by the name “Science”. If Test Data finds the word “Science” how does the model recognizes whether the word is ham (P(Science|Ham)) or spam (P(Science|Spam)). In this case smoothing parameter (α) is used. Each of the word is given with this parameter. So,

P(Science|Ham) <> 0 P(Science|Spam) <> 0



By Default, **class\_prior** is “none”: This is used for Prior probabilities.

**Fit\_prior** is set to True to learn prior probabilities from the model. If it is set to False, it will be uniform across the model.



A screenshot of a computer program

Description automatically generated

**Confusion Matrix**:

A chart of different colored squares

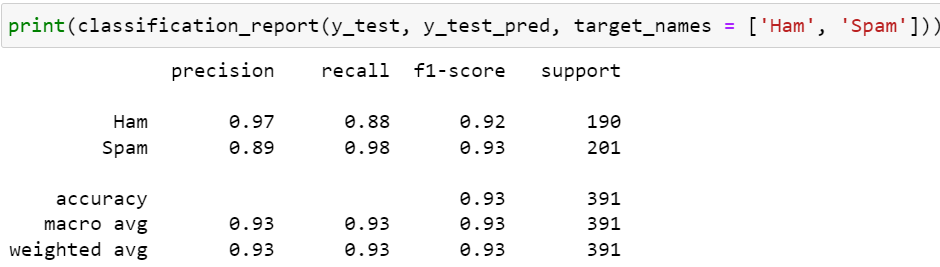
Description automatically generated

Negative: 0 (Ham) Positive: 1(Spam)

True Negative: 167 False Positive: 23

False Negative: 5 True Positive: 196

**Classification Report**:



Here Ham = 190 and spam = 201, which shows dataset is slightly imbalanced.

Accuracy : (TP + TN) / (TP + FP + FN + TP) = (167 + 196) / (167 + 23 + 5 + 196) = 0.93 (93%)

Precision (Spam) : TP/(TP + FP) = 196 / (196 + 23) = 0.89 (89%)

Precision (Ham) : TN/(TN + FN) = 167 / (167 + 5) = 0.97 (97%)

Recall (Spam) : TP/(TP + FN) = 196 / (196 + 5) = 0.98 (98%)

Recall (Ham) : TN/(TN + FP) = 167 / (167 + 23) = 0.88 (88%)

F1 Score (Ham) : Harmonic mean of precision and Recall. Harmonic mean penalizes lower values.

F1 Score (Ham) : 2 \* Precision \* Recall / (Precision + Recall)

F1 Score (Ham) : (2 \* 0.97 \* 0.88)/ ( 0.97 + 0.88) = (1.7072)/(1.85) = 0.9228 (92.3%)

F1 Score (Spam) : (2 \* 0.89 \* 0.98)/ ( 0.89 + 0.98) = (1.7442)/(1.87) = 0.9327 (92.3%)