1. Theory of Naïve Bayes Classifier

Bayes' Theorem in probability theory and statistics describes the probability of an event, based on prior knowledge of conditions that might be related to the event. It was named after Thomas Bayes. The formula for Bayes' theorem is given as:

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

Where:

- P(A|B) is the posterior probability, which is the probability of hypothesis A on the observed event B.

- P(B|A) is likelihood probability, which is the probability of the evidence given that the probability of a hypothesis is true.

- P(A) is prior probability, which is the probability of hypothesis before observing the evidence.

- P(B) is the marginal probability, which is the probability of Evidence.

Naive Bayes is a classification algorithm for binary (two-class) and multiclass classification problems. Its name comes from a simplification of the probabilities calculations for each class. Probabilities of each attribute value are not calculated. They are assumed to be conditionally independent given the class value. This strong assumption are rather unlikely for the real data. Despite the fact it performs surprisingly well. It is easy and quick to implement, since it does not need a model to be trained. Naïve bayes algorithm is used in machine learning e.g. for spam filtration, Sentimental analysis, classifying articles.

2. Project implementation of the algorithm and its performance

2.1 Implementation of the algorithm

In our project we implemented Gaussian based Naïve Bayes, which means that values associated with each class are distributed according to a normal (or Gaussian) distribution.

First, it is needed to divide our data by class. We do it with usage of divide_data_by_class() function. In order to do that we create a dictionary where each key is the class value and lists of records from dataset are values. We use dictionary object, because it allows easier access to the data.

Next, mean and standard deviation should be calculated for every column in dataset. In order to do that we created gather_data_params() function, which uses arithmetic_mean(), std_deviation(). The result of mentioned function is tuple of mean, standard deviation and number of rows in each column.

Further, we have to separate dataset into rows by class (FUNCION NAME) and then calculate summary statistic for each column with (FUNCTION NAME). Last but not least we use function (FUNCTION NAME) to stored list of tuples of statistics in a dictionary by their class value.

In the final step, we calculate probability of belonging of a new data to a suitable class. It is done according to given equation: P(class|data) = P(X|class) * P(class). The division given in Bayes Theorem has been removed to simplify the calculation. It leads to approximation of data belonging results. The value is still maximized, meaning that the calculation for the class that results in the largest value is taken as the prediction. Implementation is often simplified in this way, because class prediction is more interesting than probability.

The input variables are treated separately. It is a reason why this algorithm is called 'naïve'. First the total number of training records is calculated from the counts stored in the summary statistics. This is used in the calculation of the probability of a given class or P(class) as the ratio of rows with a given class of all rows in the training data. Next, probabilities are calculated for each input value in the row using the Gaussian probability density function and the statistics for that column and of that class. Probabilities are multiplied together as they accumulated. This process is repeated for each class in the dataset.

2.2 The project performance

We measured accuracy of our project with usage of train datasets like 'iris.csv' and 'pima-indians-diabetes.csv'. It is about 95%, when number of folds is equal 5 for 'iris.csv' and it is about 75% when number of folds is equal 5 folds for 'pima-indians-diabetes.csv'. In our acceptance test the coverage was about 99%. We also implemented unit test in order to check the correctness of every function work, which is implemented in class 'NaiveBayesClassifier' . The coverage of these tests is very close to 100%.

Also we have compared the performance of our algorithm with the one implemented in scikit-learn library. The results are very similar.

3. Project files description

3.1 /src

naive_bayes.py – the script consists of one class – 'NaiveBayesClassifier'. Its methods are responsible for naïve bayes algorithm implementation

3.2 /tests/unit_tests

According to the rules of writing unit test, every method of class 'NaiveBayesClassifier' is tested with known set of input and output data. The pytest framework is used to run these tests. The result of the class method is compared with the dataset, which correctness are certain. For example, for testing arithmetic_mean() method, list of some numbers and its mean value calculated by the outside tool were provided. Then with usage of 'assert' keyword results of the arithmetic_mean() function, executed on input dataset were compared with the given mean value. The pytest framework returns information about outcomes in a standardized format: test passed/test failed. In case of a failure information about the line, where the function does not work properly are displayed.

4. Source code documentation

Source code documentation can be acquired from the project itself, as it is generated by Sphinx library.

How to acquire it (tested on Ubuntu 20.04 LTS):

Go to the main project directory (e.g. */home/user/Naive-Bayes*) and run the following command: *python3 setup.py docs*

Now go to the *build/sphinx/html* subdirectory and run index.html with the Internet browser.

You will then be able to explore the project documentation generated by Sphinx.