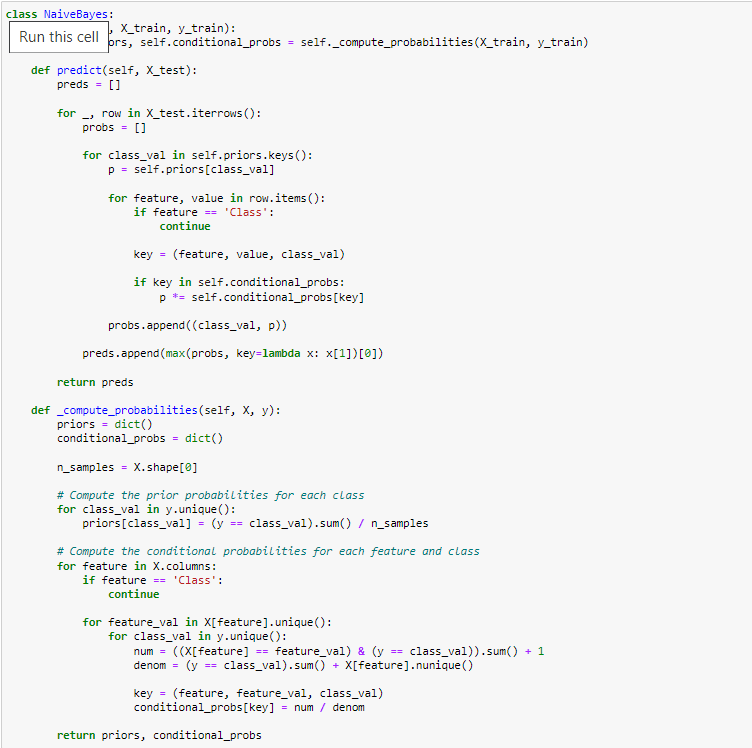
For implementation of Naïve Bayes algorithm we start by importing the necessary packages like numpy, pandas, matplotlib, scikit-learn, and seaborn. Then we read the dataset in the form of Pandas dataframe and display the first 10 rows using the head() function. We then visualize the 'Class' feature in the data using a histogram.

The function 'conversionOfCategoricalDataToNumeric' is defined to convert the dataframe's category characteristics to numerical values. Using dictionaries, this function converts each category value to a number. Each feature's mapping is contained in the dictionaries.

The dataframe's missing values '?' are replaced with NaN values, and any rows with missing values are removed from the dataframe using the dropna() function. The reset\_index() method is used to reset the dataframe's index.

Lastly, the class 'NaiveBayes' is created, which has two methods: fit() and predict(). With the training data, the fit() algorithm computes the prior probabilities and conditional probabilities. These probabilities are used by the predict() function to forecast the class of the test data. The probabilities are determined using the Naive Bayes algorithm. As the anticipated class, the class with the highest probability is chosen.



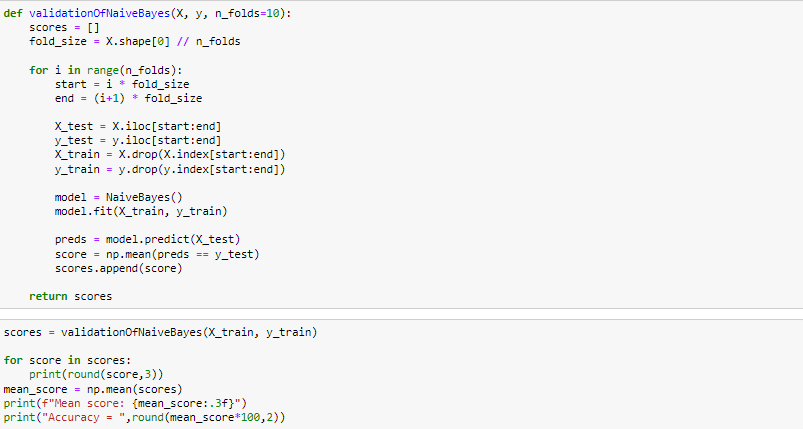
The above code defines a class NaiveBayes which implements the Naive Bayes algorithm for classification.

The fit method of the class takes in the training data X\_train and y\_train and computes the prior probabilities and the conditional probabilities of each feature given the class. The priors dictionary contains the prior probabilities for each class, and the conditional\_probs dictionary contains the conditional probabilities for each feature and class.

The predict method of the class takes in the test data X\_test and predicts the class for each sample using the computed probabilities. For each sample in X\_test, the method computes the probability of belonging to each class by multiplying the prior probability with the conditional probabilities for each feature value. The class with the highest probability is then predicted for that sample.

The compute\_probabilities method is a helper method used by fit to compute the prior and conditional probabilities. It iterates over each feature in X and each class in y and computes the conditional probabilities of each feature given the class. The prior probabilities are computed by counting the number of samples in each class and dividing by the total number of samples.

Overall, this implementation of Naive Bayes assumes that the features are conditionally independent given the class. The method is commonly used for classification problems with categorical or discrete features.



The code above is an implementation of 10-fold cross-validation for a Naive Bayes classifier in Python. The function validationOfNaiveBayes takes a feature matrix X and a target vector y as inputs, and performs 10-fold cross-validation on the data. The number of folds can be specified by the n\_folds parameter, which is 10 in our case.

The function, splits data into 10 folds of approximately equal size using the fold\_size variableA Naive Bayes classifier is trained on the remaining 9 folds and tested on the current fold for each fold. The methods fit and predict are included in the NaiveBayes class.

The classifier's accuracy is calculated as the proportion of correct predictions for the current fold and is appended to a list of scores. The list of scores is returned once all ten folds have been completed.

The mean accuracy of the classifier is determined as the average of the scores using np.mean after performing validationOfNaiveBayes on the training data.

Overall, this code provides a simple and efficient method for doing 10-fold cross-validation on a Naive Bayes classifier in Python.

Breast Cancer

Accuracy of Python code:

0.773

0.773

0.636

0.818

0.636

0.864

0.818

0.773

0.727

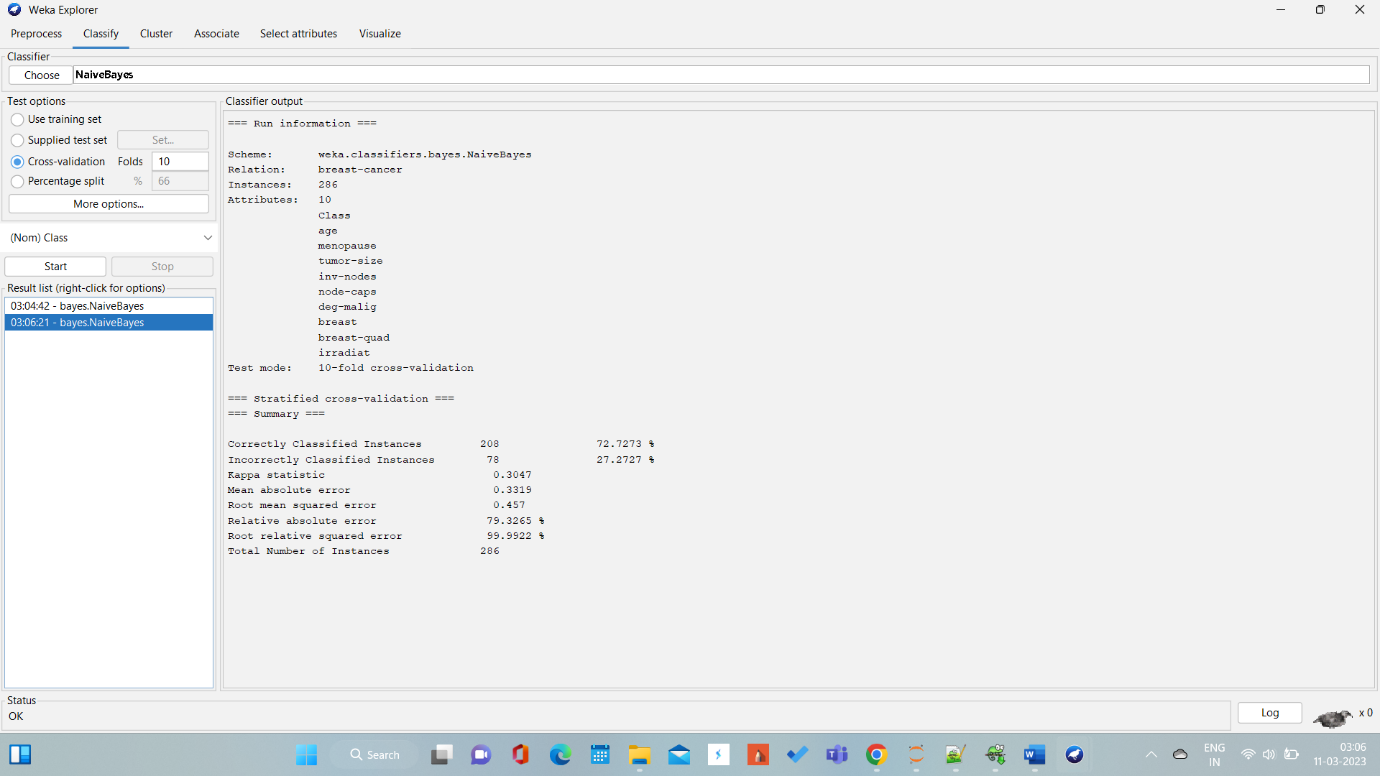
0.727

Mean score: 0.755

Accuracy = 75.45

Accuracy = 75.45

Accuracy of Weka:



Accuracy = 72.72%

Car

Accuracy of Python code:

0.848

0.87

0.855

0.87

0.812

0.862

0.884

0.826

0.804

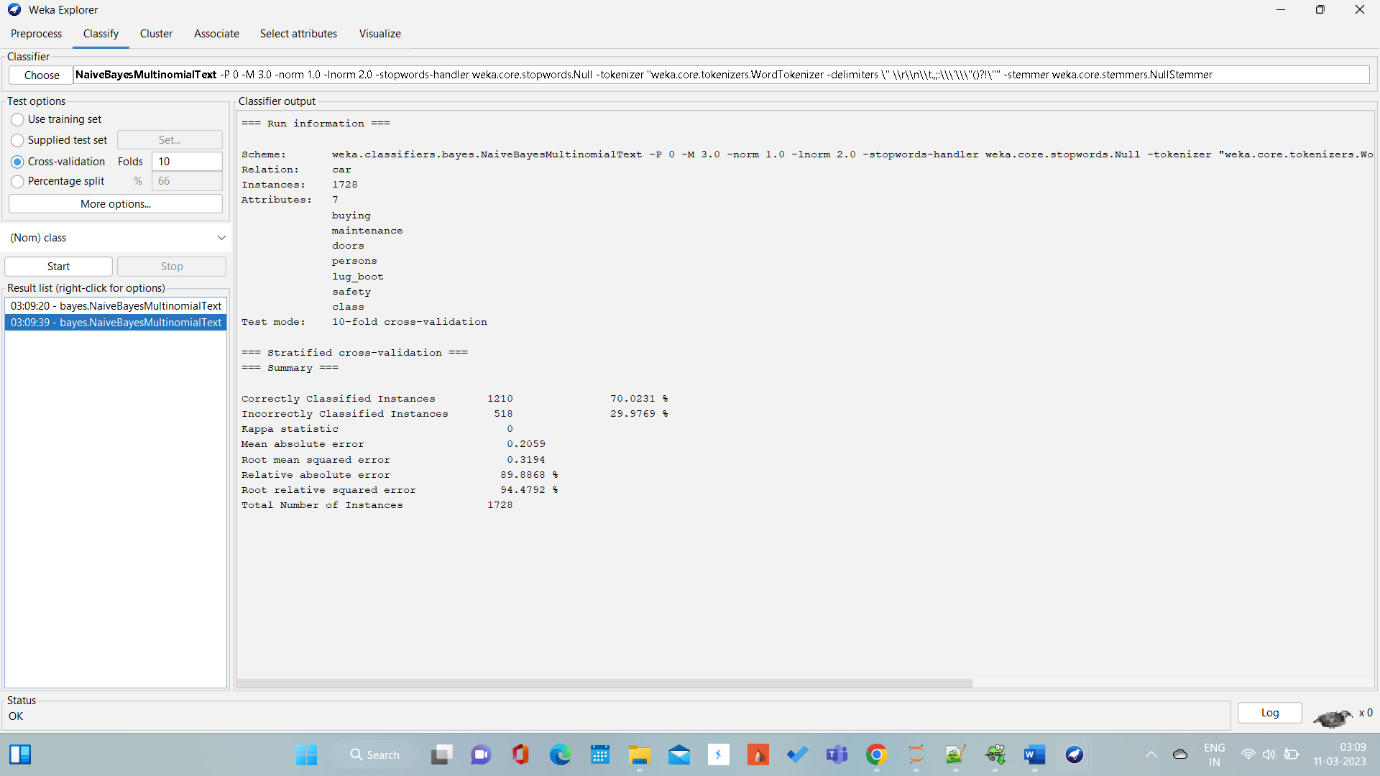
0.826

Mean score: 0.846

Accuracy = 84.57

Accuracy = 84.57

Accuracy of Weka:



Accuracy = 70.02%

Hayes Roth

Accuracy of Python code:

0.9

0.8

1.0

0.7

0.9

0.5

0.9

0.8

0.8

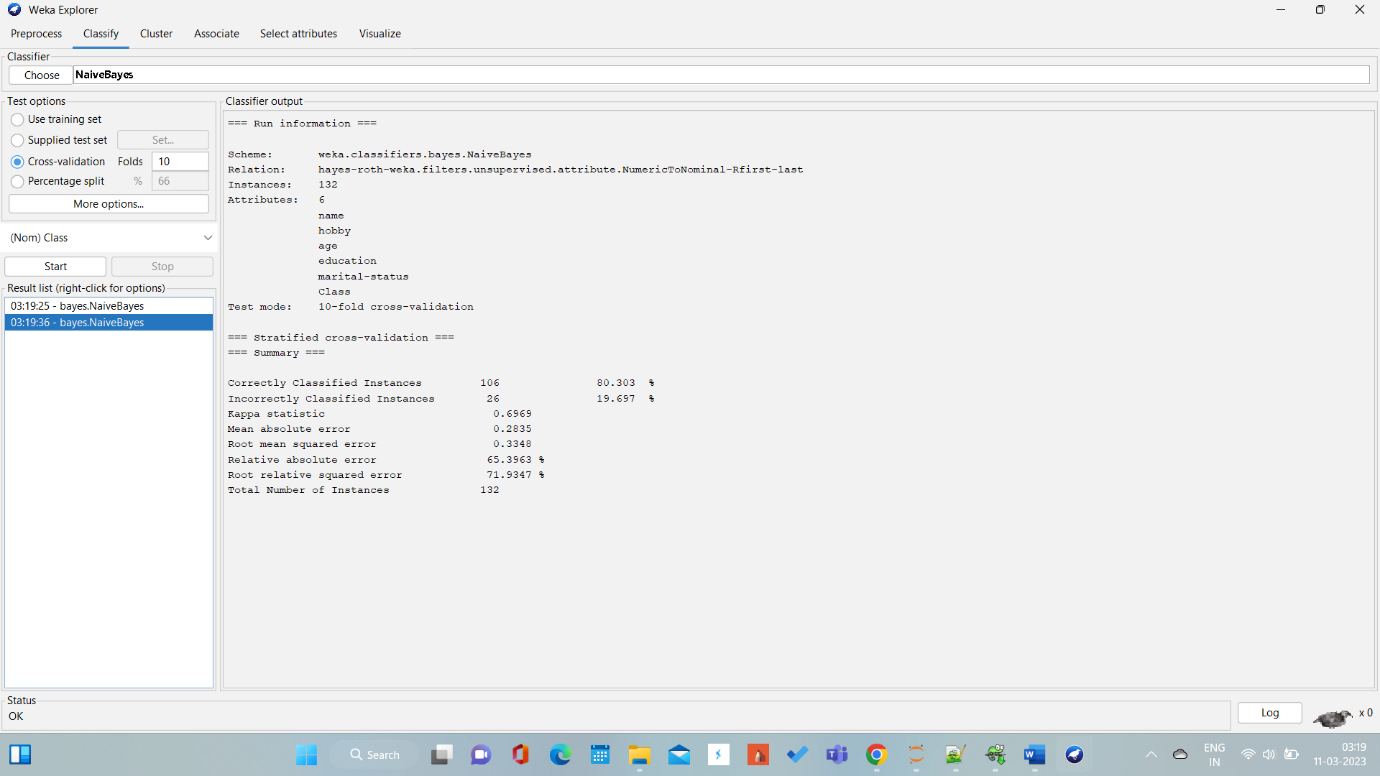
0.8

Mean score: 0.810

Accuracy = 81.0

Accuracy = 81.0

Accuracy of Weka:



Accuracy = 80.30%

**Hypothesis Testing:**

Hypothesis testing is usually performed on the two sets of results obtained from 10-fold cross-validation of the Naive Bayes classifier in Python and Weka.

However, there are a few things to consider before doing so:

The two sets of results obtained from Python and Weka are being pre-processed differently. Therefore, it may not be appropriate to directly compare the results without taking this into account.

The two sets of results may have different variance. The variance of the results obtained from Python and Weka may be different due to the differences in the implementation of the Naive Bayes classifier. Therefore, it may not be appropriate to use statistical tests that assume equal variances.

Hence, we can come to a conclusion that Hypothesis testing cannot be performed in this case.

**Reference:**

<https://machinelearningmastery.com/naive-bayes-classifier-scratch-python/>

<https://machinelearningmastery.com/k-fold-cross-validation/>

<https://medium.com/@rangavamsi5/na%C3%AFve-bayes-algorithm-implementation-from-scratch-in-python-7b2cc39268b9>

<https://towardsdatascience.com/naive-bayes-classifier-from-scratch-with-python-942708211470>

**Note:**

**I have attached a pdf as well. This is to ensure that the images are being displayed correctly and there is not change in the formatting.**