Implementing Naïve Bayes

# Creator: Mihir Ingole

### **Introduction:**

Naïve Bayes is a collection of classification algorithms based on Bayes’ Theorem which basically finds the probability of an event occurring given the probability of another event that has already occurred. Bayes’ theorem is stated as: -

P(X|Y) = (P(Y|X) \* P(X)) / P(Y)

In our implementation we’ll be using this as: -

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

**This project demonstrates the implementation and working on the Naive Bayes algorithm on: -**

**1). Iris Dataset.**

**2). Hayes-Roth dataset.**

**3). Car-evaluation dataset.**

**4). Breast Cancer dataset.**

### **Implementation in python:**

*Please refer the “Naive bayes.ipynb” file.*

The below methods are used to implement Naïve Bayes’ algorithm in python.

datasetSeparated: This function separates the data according to class value. It creates a dictionary where the class values are the keys, and the corresponding data are its values.

meanAndStdDevSummary: This function takes the data, belonging to one class, without class label as input and produces a summary in the form of a tuple which contains the mean, standard deviation and count of the records belonging to that class as output.

summarizedClasses: This function outputs a dictionary which contains the class labels as keys and the summarized output of meanAndStdDevSummary() function its corresponding value.

Gaussianpdf: This function returns the gaussian probability of a data element based on the mean and standard deviation of the entire dataset.

Input: Feature value of a data element, mean and standard deviation of that feature based on the entire data of its corresponding class.

Output: Gaussian probability

calculate\_class\_prob: This function implements the naive bayes algorithm. Breaking down the naive bayes idea, this function basically implements

P(class|X1, X2, ... , Xn) = P(X1|class) \* P(X2|class) \* .... \* P(Xn|class) \* P(class)

Where n is the total number of features.

Input: dataset, summarized classes (output of summarizedClasses() function) and the test data.

Output: Calculates the probabilities of the test data for each class and returns a dictionary as class labels and its probabilities.

creatingKFolds: This function implements the k fold cross validation algorithm.

Input: entire dataset with class labels and value of k.

Output: list containing k divisions of the entire dataset.

evaluateNaiveBayes: This function aggregates all the previous functions to return the accuracy after k fold cross validation.

Functions: Training for every k-1 folds and testing on the kth fold.

Calculating the probabilities for every element of the test data.

Calculating the accuracy based on the true and predicted values.

Averaging the accuracies of every fold and obtaining the final k fold cross validation accuracy.

Input: entire dataset and value of k.

Output: Prints the accuracies of all the k folds and returns the final cross validation accuracy.

testNaiveBayes: Return the predicted class of the test element.

### **Preprocessing on Datasets:**

### **Hayes-Roth dataset:**

The first attribute “name” in the Hayes-Roth dataset is a distinct attribute, hence is eliminated.

The dataset is already in label encoded format. Also, one hot encoding is performed for comparing the two methods.

### **Car-Evaluation dataset:**

Both one-hot encoding and label encoding is performed on the attributes of the dataset and the accuracy after cross validation is used for comparing the encoding methods.

### **Breast-Cancer dataset:**

Both one-hot encoding and label encoding is performed on the attributes of the dataset and the accuracy after cross validation is used for comparing the encoding methods.

### **Algorithm Evaluation:**

The implemented Naïve Bayes is compared across different datasets and on different encoding methods.

Seed value is set to 1.

Below table shows the accuracy after 10-fold cross validation on each dataset.

|  |  |  |
| --- | --- | --- |
| **Dataset** | **Label-Encoded data (%)** | **One-Hot-Encoded data (%)** |
| Hayes-Roth | 66.923 | 22.307 |
| Breast-Cancer | 77.5 | 76.428 |
| Car-Evaluation | 74.360 | 69.941 |

### **Implementation in WEKA:**

### **Preprocessing on Datasets:**

* For Hayes-Roth dataset, all the attribute values needed to be converted from numeric values to nominal values before classification.
* For Car- Evaluation dataset, the values of attributes “doors” and “persons” needed to be converted from string values to nominal values before classification.
* No preprocessing was required for Breast-Cancer dataset.

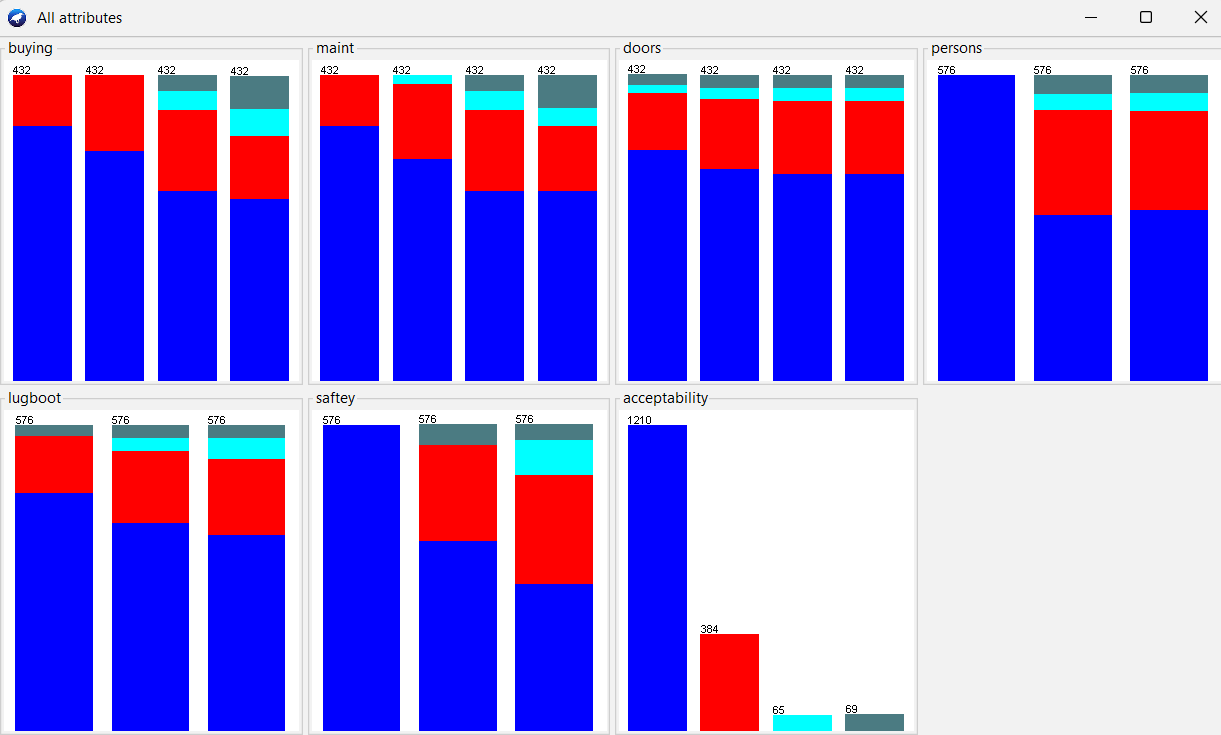
### **Visualizing the datasets:**

### **Hayes-Roth:**

Chart, bar chart

Description automatically generated

### **Car-Evaluation:**



### **Breast-Cancer:**

Chart, bar chart

Description automatically generated

### **Classification results in WEKA:**

Classification results were obtained with 10-fold cross validation on each dataset using the naïve bayes’ algorithm.

### **Hayes-Roth:**

Text

Description automatically generated with medium confidence

### **Car-Evaluation:**

A picture containing graphical user interface

Description automatically generated

### **Breast-Cancer:**

A picture containing graphical user interface

Description automatically generated

### **Hypotheses testing:**

*Please refer the “Hypotheses Testing.ipynb” file.*

Hypotheses testing is performed for accuracy comparison of Naive Bayes algorithm in Python and Weka.

* The hypotheses testing comprises of 2 samples. Sample 1 is the resulting accuracy from the 10 folds of Naive Bayes implemented in python and sample 2 is obtained from the 10 folds in Weka.
* Sample 1 contains the label encoded results of each dataset.
* Since the option to obtain the accuracy of each fold is not readily available in Weka, the accuracies presented below are obtained by setting the training-test split to 90-10% and varying the seed value from 1 through 10.

### **Results:**

### **Hayes-Roth:**

Both the methods perform differently with classifier in WEKA performing better than python.

### **Car-Evaluation:**

The performance metrics of both the methods is same.

### **Breast-Cancer:**

Both the methods perform differently with classifier in python performing better than WEKA.

### **Accuracy comparison table for Naïve Bayes’ algorithm in python and WEKA:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Dataset** | **Label-Encoded data (%)**  **Python** | **One-Hot-Encoded data (%)**  **Python** | **WEKA (%)** |
| Hayes-Roth | 66.923 | 22.307 | 80.303 |
| Breast-Cancer | 77.5 | 76.428 | 85.5324 |
| Car-Evaluation | 74.360 | 69.941 | 74.8252 |

### **Conclusion:**

In this project we learned the working and implementation of Naïve Bayes algorithm in python. We used 2 encoding algorithms namely one-hot encoding and label encoding. An accuracy comparison was performed using hypotheses testing for the naïve bayes’s algorithm on python using label encoded dataset and WEKA’s classification. From the results we can conclude that out implemented algorithm performs better than WEKA on the Breast-Cancer dataset, WEKA performs better than python algorithm on Hayes-Roth dataset and both classifiers work nearly the same on the Car-Evaluation dataset.

### **References:**

<https://www.geeksforgeeks.org/naive-bayes-classifiers/>

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

<https://www.geeksforgeeks.org/python-removing-first-element-of-list/>

<https://www.statology.org/one-hot-encoding-in-python/>

<https://datagy.io/sklearn-one-hot-encode/>

<https://www.geeksforgeeks.org/ml-one-hot-encoding-of-datasets-in-python/>

<https://www.geeksforgeeks.org/delete-a-column-from-a-pandas-dataframe/>

<https://www.geeksforgeeks.org/how-to-convert-numpy-array-to-list/>

<https://stackoverflow.com/questions/32078737/create-pandas-dataframe-manually-without-columns-name>

<https://datagy.io/sklearn-one-hot-encode/>

<https://www.geeksforgeeks.org/ml-one-hot-encoding-of-datasets-in-python/>

<https://www.geeksforgeeks.org/delete-a-column-from-a-pandas-dataframe/>

<https://www.statology.org/hypothesis-test-python/>