Making a Naive Bayes Classifier from scratch in Python

Shashank Ramesh Bhatia (UTA 1001876757) for ML 6363 Spring 2022

Background

Bayes' Theorem provides a way that we can calculate the probability of a piece of data belonging to a given class, given our prior knowledge. Bayes' Theorem is stated as:

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

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

Naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness, and diameter features.

Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample.

The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation.

1. Shuffle the dataset randomly.
2. Split the dataset into k groups
3. For each unique group:
    a. Take the group as a hold out or test data set
    b. Take the remaining groups as a training data set
    c. Fit a model on the training set and evaluate it on the test set
    d. Retain the evaluation score and discard the model
    e. Summarize the skill of the model using the sample of model evaluation scores

Data

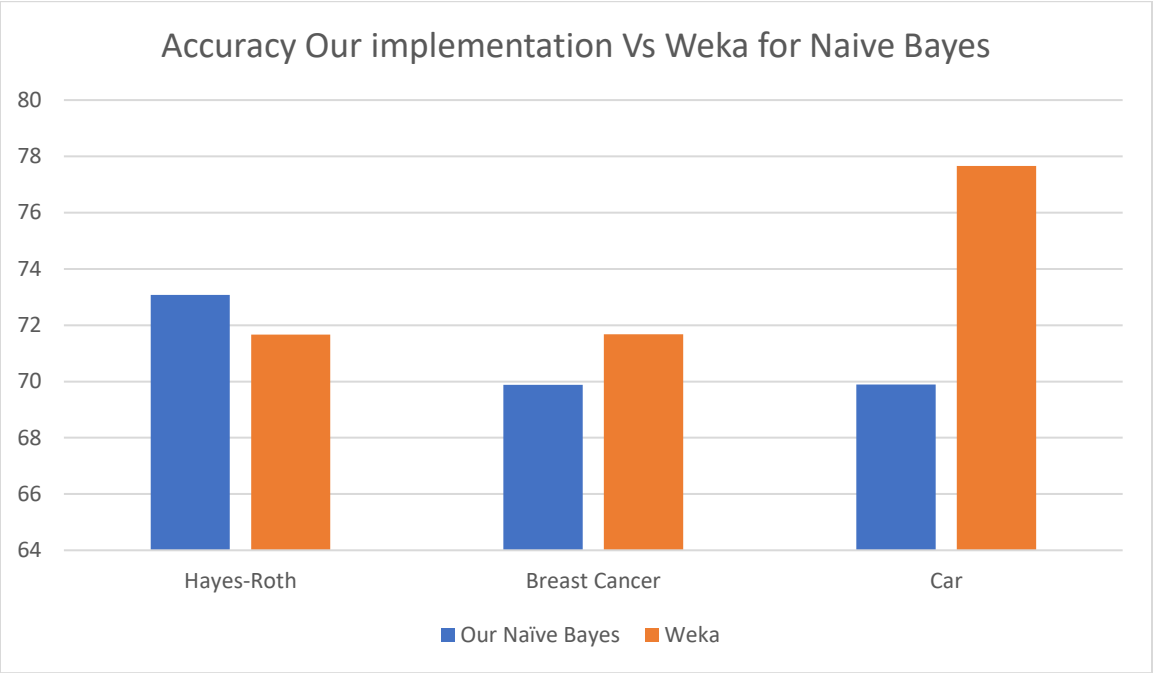
We are going to use three datasets for this - "hayes-roth", "car data", and "breast cancer data"

Implementation

```
def percentage(x, y):
    result = 0
    for i in range(len(x)):
        if x[i] == y[i]:
            result += 1
    return result / float(len(x)) * 100.0


def mean(numbers):
    return sum(numbers)/float(len(numbers))


def stdev(numbers):
    avg = mean(numbers)
    variance = sum([(x-avg)**2 for x in numbers]) / float(len(numbers)-1)
    return sqrt(variance)

# Calculate the mean, stdev and count for each column in a dataset
def summarize_data(data):
    summaries = [(mean(col), stdev(col), len(col)) for col in zip(*data)]
    del(summaries[-1])
    return summaries


def separate_by_class(data):
    separated = dict()
    for i in range(len(data)):
        each_row = data[i]
        each_class = each_row[-1]
        if (each_class not in separated):
            separated[each_class] = list()
        separated[each_class].append(each_row)
```

```
def percentage(x, y):
    result = 0
    for i in range(len(x)):
        if x[i] == y[i]:
            result += 1
    return result / float(len(x)) * 100.0

def mean(numbers):
    return sum(numbers)/float(len(numbers))

def stdev(numbers):
    avg = mean(numbers)
    variance = sum([(x-avg)**2 for x in numbers]) / float(len(numbers)-1)
    return sqrt(variance)

def summarize_data(data):
    summaries = [(mean(col), stdev(col), len(col)) for col in zip(*data)]
    del(summaries[-1])
    return summaries

def separate_by_class(data):
    separated = dict()
    for i in range(len(data)):
        each_row = data[i]
        each_class = each_row[-1]
        if (each_class not in separated):
            separated[each_class] = list()
        separated[each_class].append(each_row)
    return separated
```

Results



We find that our classifier was less accurate than the Weka implementation for the same data.

References

1. https://machinelearningmastery.com/naive-bayes-classifier-scratch-python/
2. https://en.wikipedia.org/wiki/Naive_Bayes_classifier
3. https://www.analyticsvidhya.com/blog/2022/02/k-fold-cross-validation-technique-and-its-essentials/
4. https://machinelearningmastery.com/k-fold-cross-validation/