IMPLEMENTATION OF NAÏVE BAYES

STUDENT NAME – TRIPAT KAUR

STUDENT NUMBER – 0572962

Contents

GOAL

When given the three options to choose from, I chose Option 01 -> Implementing a machine learning algorithm from scratch. The algorithm chosen is Gaussian Naïve Bayes and is implemented in Python. Validation of the implementation has been done by comparing the results obtained with sci-kit's version of Gaussian Naïve Bayes. The accuracy, precision, recall of both the implementations have been compared.

DATSET USED

For validation the dataset from Option 03 -> Breast Cancer Data has been used. The dataset contains 3500 observations and 150 features.

Since the main focus of this project is the implementation of the algorithm, the dataset has been pre-processed and cleaned by doing the following steps:

1. The dataset contains multiple records per patient, and each record is then classified as being 0 or 1 i.e. whether that micro is benign or malignant. The patient id column has been dropped to focus only on the classification of micros.

2. The redundant columns have been dropped before further processing the dataset. These include - original_firstorder_Kurtosis, original_firstorder_Skewness.

3. Feature Selection is then done using SelectKBest from scikit. Its performance with different values of k has been compared.

TESTING AND VALIDATION TECHNIQUES

It was necessary to give the same set of training and testing data to both implementations to be able to get a comparable score. For training and testing two methodologies have been used and their performances compared.

1. Splitting the training and testing data from the given dataset

2. Performing a 10 – cross validation

DESCRIPTION OF ALGORITHM

For this project, Gaussian Naïve Bayes, a part of family of Naïve Bayes algorithms has been implemented.

Naïve Bayes algorithm is based on probabilistic modelling and the Bayes theorem of conditional probability. It is used for classification problems.

Applications of Naïve Bayes – Spam prediction, Classifying documents, Sentiment prediction etc

BASIC CONCEPTS OF PROBABILITY

Two events are said to be independent if and only if:

$$P(A|B) \; = \; P(A) * P(B)$$

Equation 1

According to Bayes Theorem -

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

Equation 2

i.e. P(B|A) -> Probability of B given the probability of A

P(A|B) -> Probability of A given the probability of B

P(B) -> Probability of B

P(A) -> Probability of A

PROBABILITY WITH NAÏVE BAYES

Let us assume that x is our set of input features and we have a binary classification problem at hand. We have two classes C1 and C2. In order to predict/estimate whether x belongs to C1 or C2 we can use the above Bayes Theorem i.e. we calculate P(C1|x) and P(C2|x).

$$If \; P(C1|x) \; > \; P(C2|x) \; then \; \; x \; belongs \; to \; C1 \; else \; to \; C2.$$

Equation 3

For categorical variables this is a simple process because all we need to do is count the frequencies of each input variable for every class and compute the conditional probabilities.

If for example x has multiple features

X -> {x1,x2,x3,x4}

Y -> {C1,C2,C3}

For every observation we calculate P(C1|x), P(C2|x) and P(C3|x). Whichever value is the largest is our predicted class.

$$P(C1|x) = \frac{\big(P(C1|x) * P(C1|x2) * P(C1|x3) * P(C1|x4) * P(C1)\big)}{P(x1) * P(x2) * P(x3) * P(x4)}$$

Equation 4

From the above equation we come to know one of the most important assumptions made in Naïve Bayes Classification**. All input features are independent of one another.**

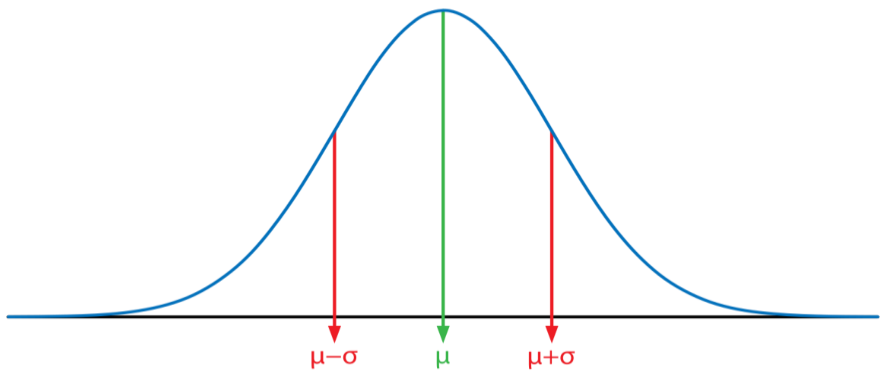
But we cannot use the above method of counting frequencies when it comes to continuous variables. For that we have the Gaussian Naïve Bayes.

GAUSSIAN NAÏVE BAYES

The Gaussian Naïve Bayes method is used when there are continuous data as input features and classes as the target variable.

For this project the Gaussian Naïve Bayes has been implemented from scratch (File – naïve_bayes_implementation.py)

In this method the strong assumption is that **the continuous variables follow a normal distribution or a Gaussian distribution**. The Gaussian distribution follows a bell shaped curve as shown below.



This Photo by Unknown Author is licensed under CC BY-SA

In the Gaussian method also the Bayes theorem is followed. The difference is in the way the conditional probabilities are calculated.

Below are the steps mentioned that are followed in Gaussian Naïve Bayes along with the functions that they map to in the code.

Let us assume we have an input set of two continuous features {x1,x2} and two target classes {C1,C2}.

Step 1

Finding the class prior probabilities i.e. P(C1), P(C2).

(Function -> __calculate_prosterior_probability)

$$P(C1) = \frac{Number\ of\ observations\ in\ the\ training\ set\ that\ map\ to\ C1}{Total\ observations}$$

Equation 5

Similarly,

$$P(C2) = \frac{Number\ of\ observations\ in\ the\ training\ set\ that\ map\ to\ C2}{Total\ observations}$$

Equation 6

Step 2

Calculating the Gaussian per class for each of the features present in the input data set.

(Function -> __claculate_mean,__calculate_sd)

$$(\mu_{x_l,C_1}|\sigma^2_{x_1,C_1})\ (\mu_{x_2,C_2}|\sigma^2_{x_2,C_2})$$
$$\left(\mu_{x_2,C_1}|\sigma^2_{x_2,C_1}\right)\ (\mu_{x_2,C_2}|\sigma^2_{x_2,C_2})$$

Equation 7

$$\mu = \frac{Sum\ of\ observations\ of\ x1\ which\ are\ mapped\ to\ C1}{Total\ number\ of\ C1\ observations}$$

$$\sigma^2 = \frac{Sum\ of\ square\ of\ difference\ of\ mean\ and\ observation\ x1\ mapped\ to\ C1}{Total\ number\ of\ C1\ observations}$$

For all classes and all features we will do the above mentioned calculations. So we get 4 values as mentioned in Equation 07.

Step 3

Compute Conditional probabilities i.e.

P(x1|C1),P(x2|C1), P(x2|C1), P(x2|C2)

(Function -> __calculate_probability, __probability_test_feature)

Since we assume that the variables follow a Gaussian Distribution we can compute the probabilities using the below mentioned formulae:

$$P(x_1|C_1) = \frac{1}{\sqrt{2}\Pi^{\sigma^2}_{\varkappa_1,C_1}} \exp -\frac{1}{2}\left(\frac{(x, -\mu_{x_1,C_1})^2}{\sigma^2_{x_1,C_1}}\right)$$

Equation 8

This is done for all features for all classes for every observation.

Step 4

Combining all probabilities into Bayes theorem

(Function -> __probability_test_feature)

$$P(C1|x) = \frac{P(x1|C1) * P(x2|C1) * P(C1)}{P(x)}$$

Equation 9

$$P(C2|x) = \frac{P(x1|C2) * P(x2|C2) * P(C2)}{P(x)}$$

Equation 10

Since in both equations i.e. Equation 9 and Equation 10 the denominator is the same, we can infer that the conditional probabilities are only dependent on the numerator. So we do not calculate the denominator.

Step 5

Deciding the class based on probabilities

(Function -> predictions())

Based on the probabilities obtained from Step 5 we assign the class for which the probability has the largest value.

Once we have the predictions, we can move onto determining how good is the model.

VALIDATION

For validating how good the model is, various metrices have been used. These include ->

1) Accuracy
2) Precision
3) Recall
4) Confusion Matrix

ACCURACY

(Function -> accuracy_score)

Defined by the percentage of correct predictions.

$$Accuracy = (Correct\ Predictions/Total\ Predictions) * 100$$

Equation 11

However accuracy is usually not a good measure of how well a model performs specially when we have an imbalanced set of samples. This is because the accuracy deals with how well a model performs overall. It is possible that it performs well for one class but not for another. So we look at other measures such as Precision and Recall.

CONFUSION MATRIX

(Function -> confusion_matrix,precision,recall)

The confusion matrix is a nXn matrix where n is the number of classes in a target variable. It displays the number of True Positives, True Negatives, False Positives and False Negatives. So it gives us a detailed view of how well the model will perform for each class.

It can be used to derive other KPIs such as Precision and Recall.

RESULTS OBTAINED

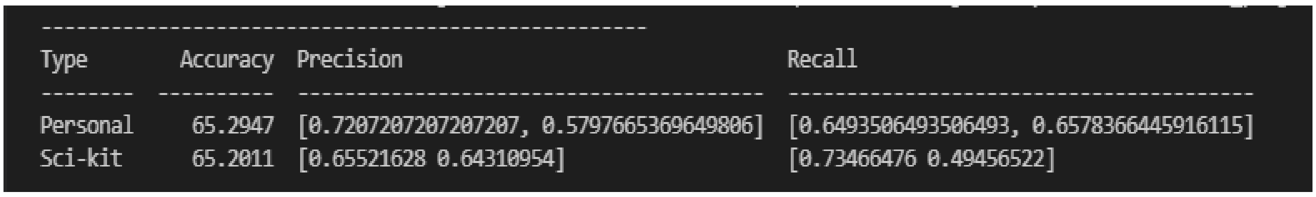
(File – main.py)

In order to validate the implementation, the various metrics mentioned in Validation have been compared with sci-kit’s version of Gaussian Naïve Bayes.

First, the data is prepared by removing columns such as Patient Id, and duplicate columns ("original_firstorder_Kurtosis", "original_firstorder_Skewness").

Next sci-kit is used to split the dataset into train and test set with a ratio of 70-30 respectively.

Without any feature selection or further pre-processing the models (personal implementation and sci-kit implementation) are trained. They are tested by performing predictions on the test dataset and their KPIs are measured.



```
----------------------------------------------------
Type       Accuracy  Precision                               Recall
--------  ---------- --------------------------------------  ----------------------------------------
Personal    65.2947  [0.7207207207207207, 0.5797665369649806] [0.6493506493506493, 0.6578366445916115]
Sci-kit     65.2011  [0.65521628 0.64310954]                  [0.73466476 0.49456522]
```

Figure 1

As we can see the accuracy matches however there is a gap between the recall obtained. Since all features are being considered the values are becoming too small which is why there is a difference between the precision and recall.

FEATURE COUNT VS ACCURACY

Next we look at the performance by selecting features using SelectKBest from sci-kit. For that we loop through various values of k (i.e. number of selected features) in SelectKBest to see whether accuracy and other metrics improve with increase in k.

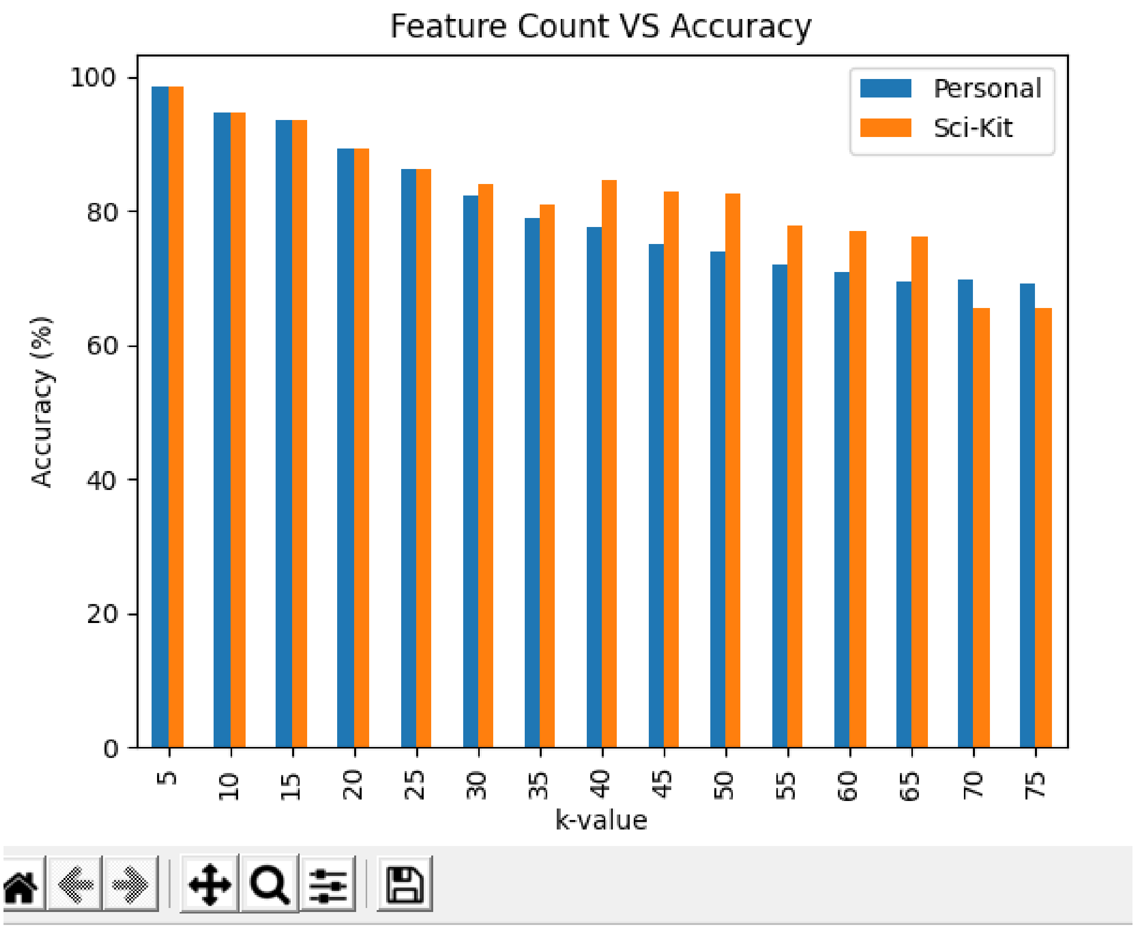


Figure 2

As we can see the accuracy decreases as the feature count increases. This is because the correlated features increase. The performance of Naïve Bayes degrades if correlated features are included because they tend to be counted multiple times for the same model.

Also for an extremely small number of feature count (i.e. 5) accuracy is highest but then that will lead to over fitting of the model.

TRAIN-TEST SPLIT VS CROSS VALIDATION

Once we have a k value we next compare the performances between the two different validation techniques i.e. Train-Test Split and Cross Validation.

(Function ->cross_validation)

For train-test split the ration of 70-30 has been maintained. Cross validation is done using 10 folds.

Cross Validation gives slight better performance than Train-Test Split as can be seen below:

Method	Personal – Accuracy	SciKit - Accuracy
10 Fold Cross Validation	94.43%	94.43%
Train-Test Split	93.63%	93.63%

Figure 3

This is because in cross validation multiple times training is done using random observations from the given dataset thus leading to a better performance.

DRAWBACKS IN THE RESULTS OBTAINED

1. The first and major drawback is the time for execution. Since in the implementation done by me I have only used simple Python lists I have multiple loops which are taking some time to execute. That is not the case with Sci-Kit.

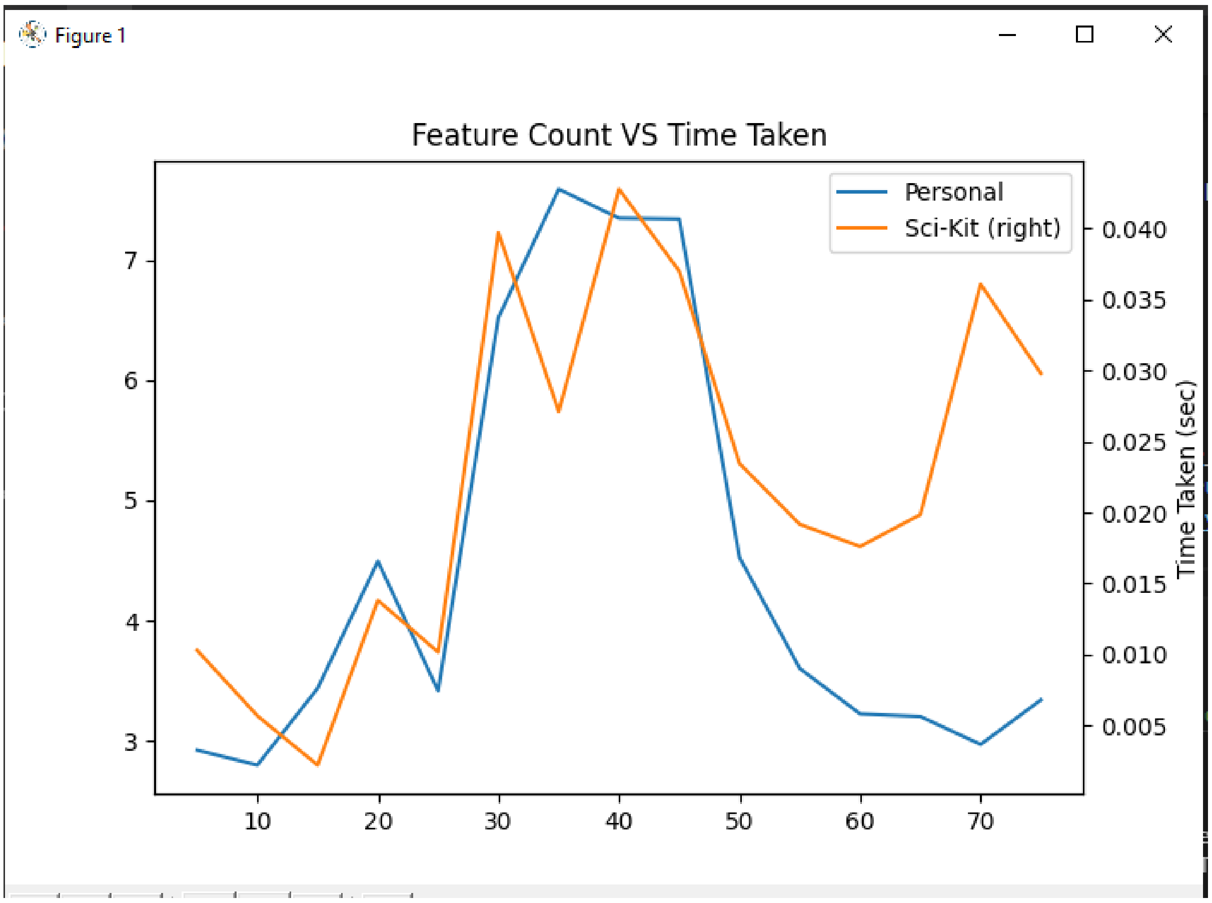
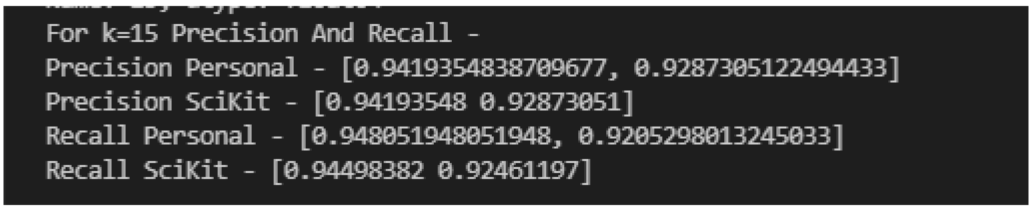


Figure 4

2. When a large number of features are being although the accuracy matches, the confusion_matrix does not match. In turn the precision and recall also do not match. As can been seen from Figure 1. But when 15 features are used the values match.



```
For k=15 Precision And Recall -
Precision Personal - [0.9419354838709677, 0.9287305122494433]
Precision SciKit - [0.94193548 0.92873051]
Recall Personal - [0.948051948051948, 0.9205298013245033]
Recall SciKit - [0.94498382 0.92461197]
```

Figure 5

REFERENCES

1. https://machinelearningmastery.com/naive-bayes-for-machine-learning/#:~:text=This%20extension%20of%20naive%20Bayes,deviation%20from%20your%20training%20data.

2. https://machinelearningmastery.com/better-naive-bayes/

3. https://medium.com/analytics-vidhya/use-naive-bayes-algorithm-for-categorical-and-numerical-data-classification-935d90ab273f

4. https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html#sklearn.naive_bayes.GaussianNB.predict

5. https://medium.com/@hackares/naive-bayes-algorithm-e565daa89eb7

6. https://prwatech.in/blog/machine-learning/naive-bayes-classifier-in-machine-learning/

STEPS TO EXECUTE THE PROJECT

1. Run pip install -r requirements.txt for downloading all packages. Or install them manually – pandas, time, matplotlib, sklearn, tabulate, copy, math and random

2. Execute the main.py file.