Statistics Project

Submitted by: Team 01

Team members:

Hadeer Sherif

Mariam Hatem

Nada Amr

Sarah Ibrahim

Submitted to: Dr. Ibrahim Youssef

Gaussian Naive Bayes classifier

Team 01

Department of Systems and Bio-medical Engineering

Cairo university

1. Introduction

With the massive development in technology, the need to deal with large amounts of data becomes essential. Dealing with data includes describing and summarizing the data features, making predictions or classifying the data elements. Here comes the role of statistics and machine learning. Various learning algorithms are being introduced to make the computer capable of classifying large data and providing useful output as fast as possible. Binary classification is one of the most required learning algorithms. It classifies the elements of a large dataset into two classes. In our model we used a large dataset which contains the blood pressure and glucose level of 995 persons and whether they are diabetic or not. We used descriptive statistics to describe the data features, and used some statistical methods for data preprocessing, then we divided the standard data into training data and testing data and constructed a gaussian naive bayes classifier from scratch to classify the elements of the testing data into diabetic and non-diabetic ones. Finally, we compared our results with the original results, listed in the dataset, and our model showed excellent accuracy. We also used the gaussian naive classifier from the standard python packages and the results were identical to those of the classifier we made.

2.Methods

2.1. Data Description

We utilized a dataset obtained from Kaggle platform to classify whether a patient has diabetes or not.
The dataset consists of three columns and 995 entries.
The features included in the dataset are as follows:
Glucose: This feature represents the glucose levels of the patients. It is a quantitative variable.
Blood Pressure: It is also a quantitative variable
The target variable in the dataset is:
Diabetes: This is the target variable, indicating whether the patient has diabetes or not. In the original dataset.
we performed certain preprocessing steps on the dataset.
Firstly, we converted the "diabetes" column to a Boolean format to enable straightforward classification.
Next, we addressed the issue of outliers by utilizing the Interquartile Range (IQR) method.
The IQR is a statistical measure that helps identify the spread and distribution of data. It is calculated by determining the values at the 25th percentile (Q1) and the 75th percentile (Q3), representing the first and third quartiles, respectively.
Then IQR=Q3-Q1

By calculating the IQR, we determined the upper and lower bounds beyond which data points were considered outliers.
upper bounds =Q3+1.5*IQR
lower bounds =Q1-1.5*IQR
These outliers were subsequently removed from the dataset to ensure the accuracy and reliability of the analysis.

We addressed Q1,Q3 using percentile function by importing NumPy library

2.2. Descriptive Statistics

We computed several measures of central tendency to know distribution and variability of the "blood pressure" and "glucose" features in the dataset.
These measures include the mean, standard deviation, and variance.
We also standardized the data to have zero mean and unit variance. This transformation ensures that all the features are on the same scale, removing any bias that may arise from different units or ranges of the original data.

2.3. Splitting Data

We divided the dataset into 80% training set and 20% test set to teach the model using a portion of the data and then evaluate how well it could apply what it learned to unseen data.
To split the data into training and test sets, we utilized the “train_test_split” function from the “sklearn.model_selection” module in python.
This function allowed us to split the dataset into two separate subsets, one for training our model and the other for testing its performance.

2.4. Histogram plots

We generated histogram plots to visualize the distributions of the glucose and blood pressure features. we utilized the “seaborn library” by importing it into our analysis to create histogram plots.

2.5. Testing Normality

To test the normality of the blood pressure and glucose features,
We conducted the Shapiro-Wilk test by importing it from “scipy.stats”.

2.5.1. Testing Normality For the blood pressure feature

We performed the Shapiro-Wilk test.
The test evaluates the null hypothesis that the data follows a normal distribution.
Based on the results, we found that the p-value obtained was less than the significance level of 0.05, p-value=6.790084466912205e-11
Therefore, we rejected the null hypothesis, indicating that the blood pressure feature does not follow a normal distribution.

2.5.1. Testing Normality For the glucose feature

We performed the Shapiro-Wilk test.
The test evaluates the null hypothesis that the data follows a normal distribution.
Based on the results, we found that the p-value obtained was less than the significance level of 0.05, p-value= 1.7425194464549662e-18
Therefore, we rejected the null hypothesis, indicating that the blood pressure feature does not follow a normal distribution.

2.6. Conditional Plots

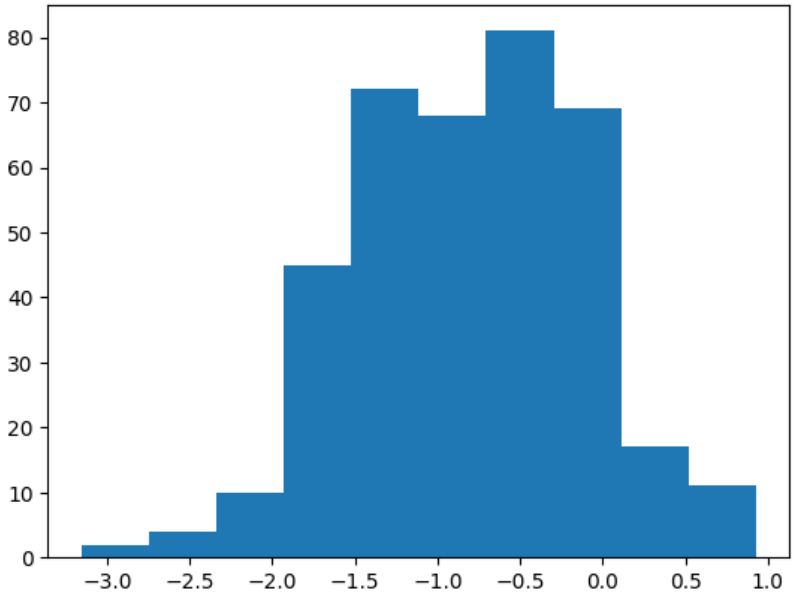
We utilized conditional plots to create separate histograms of glucose and blood pressure for diabetic(true class) versus non-diabetic(false class) individuals. This could help us identify any differences or similarities in the distributions of these variables between the two groups.
To draw conditional plots we needed to concatenate the training data with the corresponding diabetes labels, then splitting the data based on the class labels.We make concatenation Using the” pandas library”

by importing it .To visualize the conditional distributions, we drew histograms for each feature within each class by importing ‘matplotlib.pyplot”.

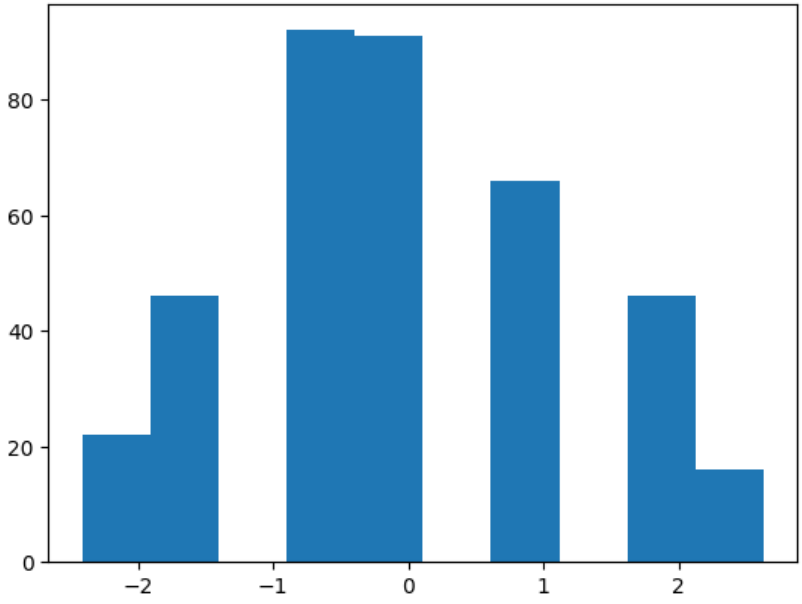
2.6.1. histogram of features for true class

We plotted histograms of the blood pressure and glucose features.

For blood pressure:

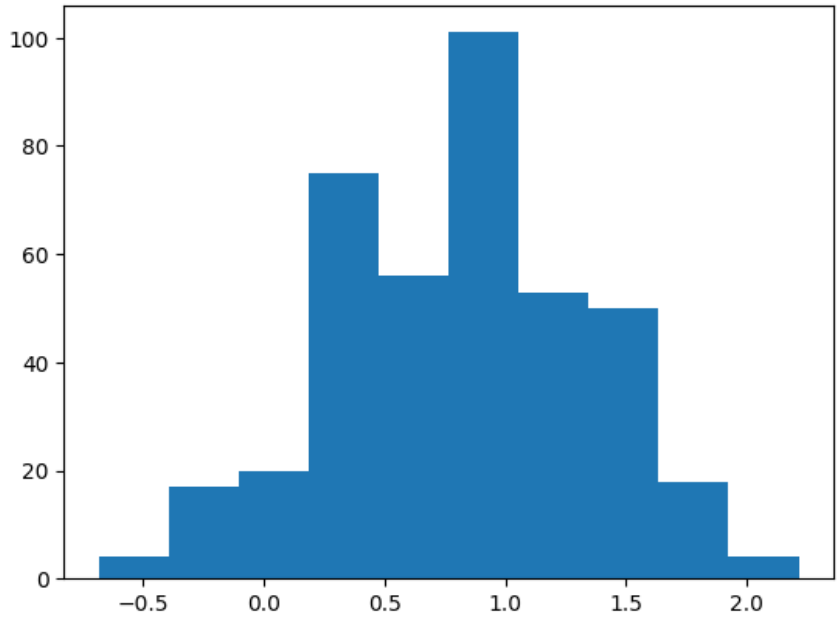


For glucose:

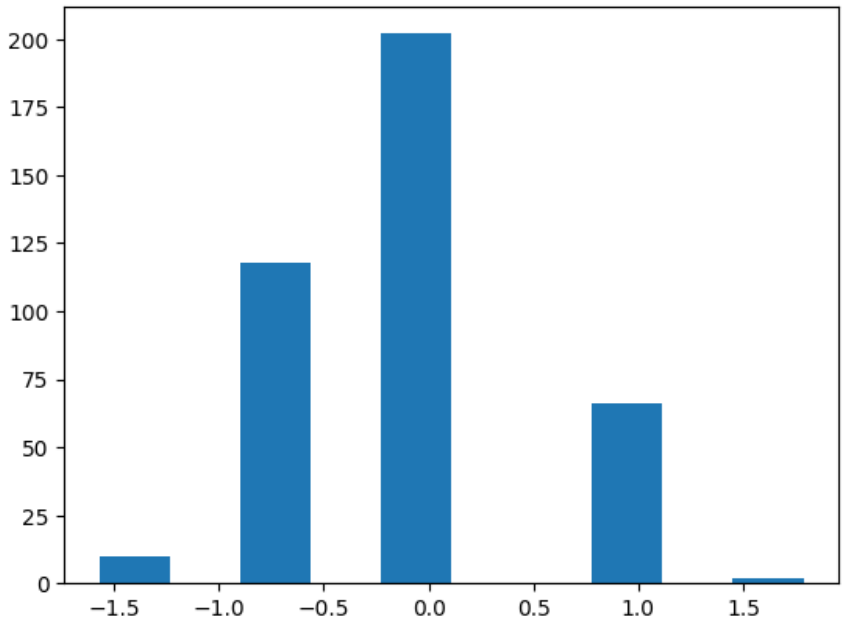


2.6.2. histogram of features for false class

For blood pressure:



For glucose:



2.7. Naive Bayes Model

We utilized a custom class called NB to implement the Naive Bayes model.

2.7.1. Training process

for the training process we needed to calculate prior probabilities and estimate parameters for the distributions associated with each class.

Prior Probabilities:

Prior probabilities represent the likelihood of each class occurring in the dataset. We calculated the prior probabilities by dividing the number of instances in each class by the total number of instances in the training set.

i.e., prior probability for true= (number of samples in the true class)\(total number of samples)

Parameter Estimation for Distributions:

We calculate mean and standard deviation within each class.

Mean: We computed the mean of each feature (blood pressure and glucose) within each class.

Standard Deviation: We calculated the standard deviation of each feature within each class.

By estimating these parameters, we will use them during the prediction phase of the Naive Bayes algorithm.

2.7.2. Prediction process

To make predictions using the Naive Bayes algorism, we pass the test set to the predict method, which returns a list of predicted labels. In this process, we calculate the probability of each input belonging to the True or False class using the Naive Bayes algorithm.

The probability calculation involves the following steps:

1. We multiply the prior probability of the True class by the probability density function for the glucose and blood pressure features.
2. Similarly, we multiply the prior probability of the False class by the probability density function s for the same features.

The formula used for each class is as follows:

$$v_{NB}(ture) = p(ture) * \left(\frac{1}{\sigma_{g\_t}\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x_{g_i}-\mu_{g\_t}}{\sigma_{g\_t}}\right)^2}\right) * \left(\frac{1}{\sigma_{b\_t}\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x_{b_i}-\mu_{b\_t}}{\sigma_{b\_t}}\right)^2}\right)$$

$$v_{NB}(false) = p(flase) * \left(\frac{1}{\sigma_{g\_f}\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x_{g_i}-\mu_{g\_f}}{\sigma_{g\_f}}\right)^2}\right) * \left(\frac{1}{\sigma_{b\_f}\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x_{b_i}-\mu_{b\_f}}{\sigma_{b\_f}}\right)^2}\right)$$

We compare the calculated $v_{NB}(ture)$ and $v_{NB}(false)$for each input. If $v_{NB}(false)$is greater than $v_{NB}(true)$, we predict that the input belongs to the False class; otherwise, we predict it to be in the True class.

After predicting the labels for the entire test set, we compare these predictions with the true labels from the test set to evaluate the accuracy of our model's predictions.

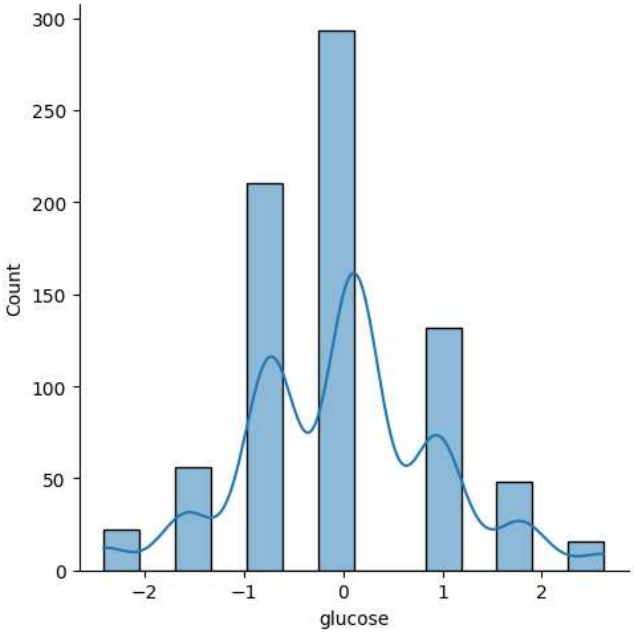
2.7.3. Prediction using Python Packages

We utilized the GaussianNB model from scikit-learn library in Python and fitting it to our training data using the fit() function.

After the model was trained, we made predictions on the test set using the predict() function, which generated a set of predicted labels Then we compared performance of our model with the GaussianNB classifier from python packages and we found that they are identical with same accuracy.

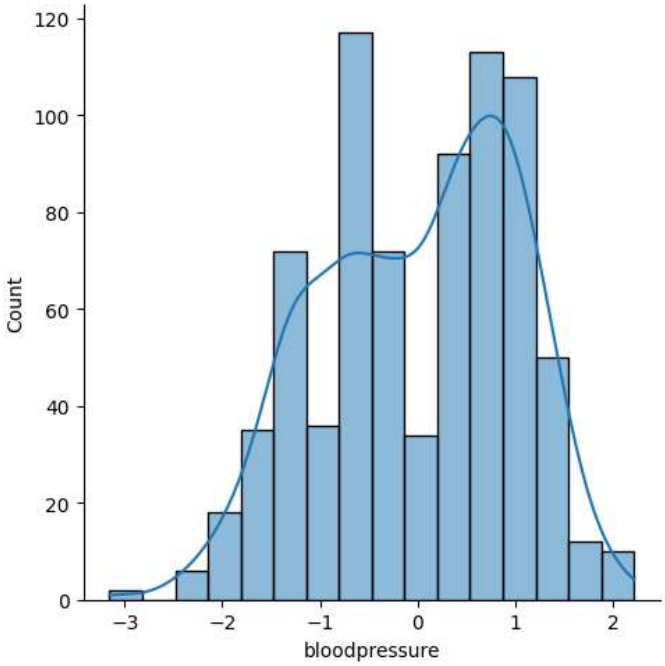
3. Results and Discussions

When we plotted the features of our training data, it was found that the glucose distribution is multimodal



Note that: The mode is the most repeated value. Multimodal distributions have multiple modes, that's why they have multiple peaks.

When we plotted the blood pressure, it was found that its distribution is bimodal.



Note that: Bimodal distributions are the distributions that have two peaks.

When we tested whether the features of the training data are normally distributed or not, it was found that both the glucose level and the blood pressure are not normally distributed

The p-value for the glucose is 1.7425194464549662e-18 which is smaller than the significance level of the test which equals 0.05. That means that we can reject the null hypothesis and that the glucose is not normally distributed.

The p-value for the blood pressure is 6.790084466912205e-11 which is smaller than the significance level of the test which equals 0.05. That means that we can reject the null hypothesis and that the blood pressure is not normally distributed.

When we used our gaussian naive classifier to classify the elements of the testing data into diabetic and non-diabetic, and compared our results with the results provided in the dataset, the accuracy of our model was 94.87%

This means that the gaussian naive classifier shows a very good accuracy in binary classifications of large datasets. The inaccuracy percentage may be caused because the gaussian naive bayes classifier assumes that the input features are completely independent while some studies showed that blood glucose and blood pressure affect each other.

When we used the gaussian naive classifier from the standard python packages and compared its results with the results of our gaussian naive bayes classifier, it was found that they are identical.This means that we built our model correctly with no errors.

4.conclution

We aimed to classify individuals as diabetic or non-diabetic using a dataset containing blood pressure and glucose level measurements. The analysis showed that the glucose distribution is multimodal, indicating multiple peaks or modes, while the blood pressure distribution is bimodal with two peaks. Both the glucose level and blood pressure features were found to deviate from a normal distribution based on the Shapiro-Wilk test.

and the study utilized a Gaussian Naive Bayes classifier to classify the testing data, and the model achieved an accuracy of 94.87%.

5. Members Contribution

Hadeer shrif : wrote the code except the removing of outliers cells, collected the report.

Nada amr : wrote the code cells to remove outliers , wrote the introduction and results and discussion in the report, participated in writing the gaussian naïve bayes from scratch code.

Sarah Ibrahim : wrote methods and conclusion in the report, participated in writing the gaussian naïve bayes from scratch code.

Mariam hatem : made the presentation slides, participated in writing the gaussian naïve bayes from scratch code.