Cairo University

Faculty of Engineering

Systems and Biomedical Department

2nd term - 2022/2023

# Diabetes classification

**24th June 2023**

## Team #15

**Supervised by : Dr. Ibrahim youssef**

# 1-Introduction

We will investigate the use of a diabetic dataset using the Naive Bayes classifier. Millions of individuals around the world suffer from the chronic medical illness known as diabetes. It is a chronic metabolic disorder characterized by high blood sugar levels, resulting from the body's inability to produce or effectively use insulin. It can lead to various complications affecting multiple organ systems, including the cardiovascular system, kidneys, eyes, and nerves. Our goal is to create a predictive model using the Naive Bayes classifier that can help diagnose diabetes using a collection of input features.

In this report, we aim to develop a Naive Bayes Classifier model for the prediction of diabetes. The objective is to analyze a given dataset, perform necessary data preprocessing steps, train the classifier, and evaluate its performance. The Naive Bayes Classifier is a probabilistic machine learning algorithm commonly used for classification tasks. With a dataset including relevant patient data, the Naive Bayes classifier offers a viable solution to the challenge of diabetes diagnosis. Through this study, we hope to encourage additional research and development in the area of medical data analysis while showing the value and efficiency of our algorithm in the healthcare industry.

# 2-Methods

## 2.1-Importing Libraries

In this project, we imported several Python libraries to support our analysis and implementation of the Naive Bayes Classifier for diabetes prediction. Each library serves a specific purpose and provides essential functionalities for data manipulation, visualization, model evaluation, and statistical analysis. The following libraries were imported:

- pandas (imported as pd): Pandas is a widely-used library for data manipulation and analysis. It provides data structures like DataFrames, which allow for efficient handling and manipulation of tabular data. We imported Pandas to load and preprocess the dataset, perform data cleaning operations, and extract relevant information.
- numpy (imported as np): NumPy is a fundamental library for scientific computing in Python. It provides support for large, multi-dimensional arrays and a collection of mathematical functions. We used NumPy to perform mathematical operations, compute statistical measures, and facilitate data manipulation tasks.
- seaborn (imported as sns): Seaborn is a high-level data visualization library that works in conjunction with Matplotlib. It provides a simplified interface and additional plotting

capabilities, making it easier to create visually appealing and informative plots. We imported Seaborn to enhance our data visualizations and generate informative plots.

- matplotlib.pyplot (imported as plt): Matplotlib is a popular plotting library that enables the creation of various types of visualizations. We imported the pyplot module from Matplotlib to generate informative plots, such as histograms, scatter plots, and bar charts, to visualize the data and present our findings.
- sklearn.metrics: This module from Scikit-learn provides various evaluation metrics to assess the performance of machine learning models. We imported metrics from sklearn to calculate accuracy scores, confusion matrices, F1 scores, and other metrics to evaluate the performance of our Naive Bayes Classifier.
- sklearn.model_selection: This module from Scikit-learn provides tools for model selection and evaluation. We imported train_test_split from this module to split our dataset into training and testing sets, allowing us to assess the performance of our Naive Bayes Classifier on unseen data.
- sklearn.naive_bayes.GaussianNB: This class from Scikit-learn implements the Gaussian Naive Bayes algorithm. We imported GaussianNB to instantiate an instance of the Gaussian Naive Bayes Classifier.
- scipy.stats: This module from SciPy provides various statistical functions and distributions. We imported stats from scipy to perform statistical calculations and tests if required.

By importing these libraries, we gained access to their functionalities, enabling us to load and preprocess the data, visualize the data, implement the Naive Bayes Classifier, evaluate the model's performance, and perform statistical analyses as necessary for our diabetes prediction project.

## 2.2-Choosing and classifying data set

It is crucial to take into account a few parameters while selecting a suitable dataset for classification using the Naive Bayes classifier. These variables include the accessibility of labeled data, the dataset's applicability to the problem domain, the dataset's size and quality, and the proportion of each class in the dataset.

On our project, we carefully selected a dataset that comprises comprehensive information related to diabetes, including physiological measurements such as glucose levels, blood pressure, BMI, and demographic factors like age. The dataset was chosen based on its relevance to diabetes prediction. We classified the dataset into two classes: "diabetes" and "non-diabetes," representing the presence or absence of the condition, respectively.

Due to the relevance of diabetes as a chronic medical condition impacting a sizable population worldwide, the selection of the diabetes dataset for categorization is particularly important. We can create a predictive model that can aid in diagnosing diabetes based on the presented attributes by using the Naive Bayes classifier on this dataset.

To perform classification on a dataset using the Naïve Bayes classifier, you need to follow these general steps:

1. Data Preprocessing: Start by preparing the dataset for classification. This includes handling missing values, dealing with outliers, and transforming the data into a suitable format for the Naïve Bayes classifier. Ensure that the dataset is divided into features (independent variables) and the target variable (the variable you want to predict).
2. Splitting the Dataset: Divide the dataset into two parts: a training set and a test set. The training set will be used to train the Naïve Bayes classifier, while the test set will be used to evaluate its performance.
3. Training the Naïve Bayes Classifier: Using the training set, apply the Naïve Bayes algorithm to learn the probability distribution of the features given each class. The Naïve Bayes classifier assumes that the features are conditionally independent given the class variable. There are different variants of Naïve Bayes classifiers, such as Gaussian Naïve Bayes, Multinomial Naïve Bayes, and Bernoulli Naïve Bayes, each suitable for different types of data.
4. Feature Probability Calculation: Calculate the probability of each feature value occurring given each class. This involves calculating mean and variance for continuous data (Gaussian Naïve Bayes), or counting occurrences for discrete data (Multinomial or Bernoulli Naïve Bayes).
5. Class Probability Calculation: Calculate the prior probability of each class in the training set. This can be done by dividing the number of instances belonging to each class by the total number of instances.
6. Predicting the Class Labels: Using the learned probabilities and the features of the test set, predict the class labels for the test instances. This involves calculating the conditional probability of each class given the feature values using Bayes' theorem and selecting the class with the highest probability as the predicted label.
7. Evaluating the Classifier: Compare the predicted class labels with the actual class labels in the test set to assess the performance of the Naïve Bayes classifier. Common evaluation metrics include accuracy, precision, recall, and F1 score.

## 2.3-Cleaning dataset

### Dataset Source:

The dataset used for diabetes prediction is sourced from the National Institute of Diabetes and Digestive and Kidney Diseases. It contains various variables related to health and demographics.

### Variables:

- Pregnancies: Number of times a woman has been pregnant.
- Glucose: Plasma Glucose concentration measured after a 2-hour oral glucose tolerance test.
- BloodPressure: Diastolic Blood Pressure in millimeters of mercury (mmHg).
- SkinThickness: Triceps skin fold thickness in millimeters (mm).
- Insulin: 2-hour serum insulin measured in micro units per milliliter (mu U/ml).
- BMI: Body Mass Index calculated as weight in kilograms divided by height in meters squared.
- Age: Age of the individual in years.
- DiabetesPedigreeFunction: A score that estimates the likelihood of diabetes based on family history.
- Outcome: Binary variable indicating whether an individual has diabetes (1) or not (0).

### Data Cleaning Steps:

1. Checking for Null Values: Initially, the dataset was checked for null values, and it was found that there were no null values. However, it was determined that zero values for Glucose, Blood Pressure, Skin Thickness, Insulin, and BMI are not valid. Therefore, any zero values in these features were corrected to NaN (Not a Number).

2. Handling Null Values: After converting zero values to NaN, the count of null values for each feature was checked. The following counts of null values were found:

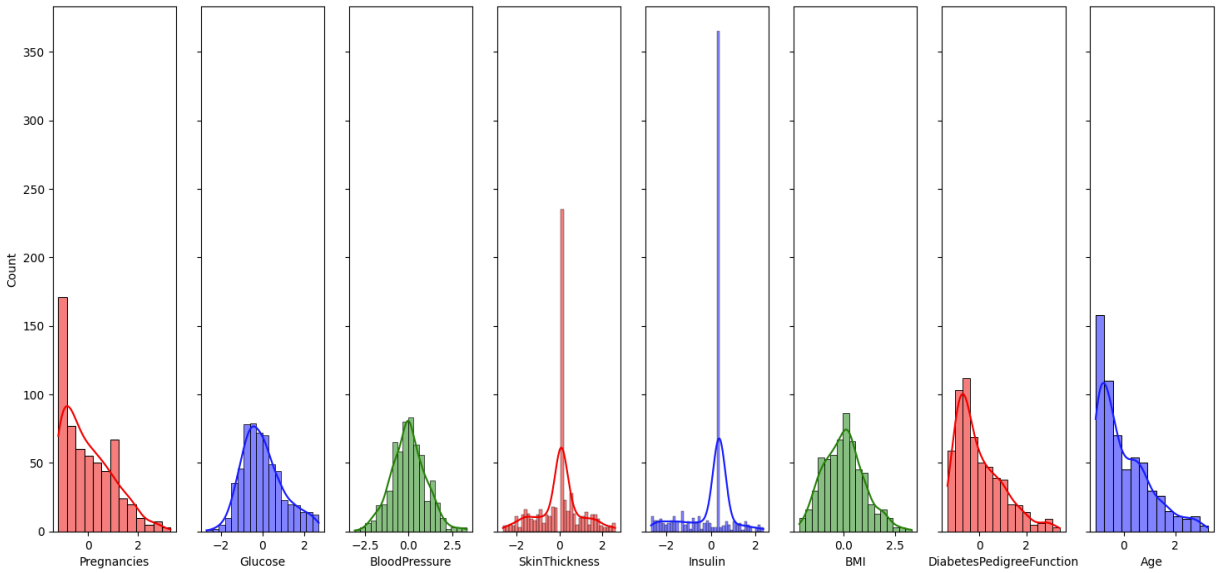
    - Pregnancies: 0
    - Glucose: 5

- Blood Pressure: 35
- Skin Thickness: 227
- Insulin: 374
- BMI: 11
- DiabetesPedigreeFunction: 0
- Age: 0
- Outcome: 0

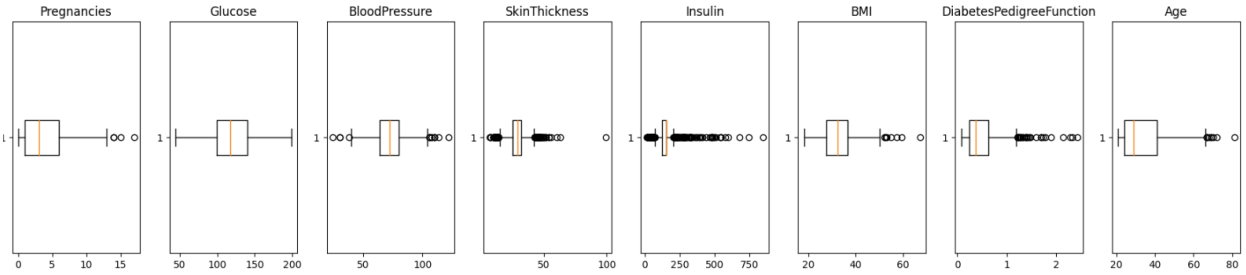
To address the null values, the missing values were imputed with the mean value of each respective feature.

3. Data Distribution Analysis: The dataframe was plotted to visualize the distributions of the variables. The observations were as follows:

    - Glucose and BMI exhibited near-normal distributions with slight deviations such as skewed tails or outliers.
    - Blood Pressure displayed a near-normal distribution with a prominent peak in the middle indicating high frequency.
    - Skin Thickness and Insulin showed distributions with some resemblance to a bell curve but had prominent peaks in the middle and skewed tails.
    - Pregnancies, Age, and DiabetesPedigreeFunction were heavily right-skewed, with a majority of the data concentrated towards lower values and a long tail extending towards higher values.




4. Outlier Detection and Removal: Boxplots were created to identify outliers within the dataset. The Shapiro-Wilk test was employed to determine whether outliers should be removed using z-scores or the interquartile range (IQR) method. After removing the outliers, the dataframe was plotted again to ensure the data appeared appropriate.
Before Removal:




In order to optimize the efficacy of our model, we shall exclusively eliminate the highly anomalous data points, considering the relatively limited size of our dataset. This strategy aims to augment the number of data points available for training, thereby facilitating the model's acquisition of a more comprehensive understanding without significantly compromising its overall accuracy in our specific context.

5. Duplicate Data: The dataset was checked for duplicate entries, and it was found that there were no duplicates present.

6. Descriptive Statistics: Finally, descriptive statistics were calculated to summarize the dataset, providing insights into the central tendency, dispersion, and shape of the variables.

## 2.4-Standardizing the data set

- ### Introduction

Data standardization, is a preprocessing step in data analysis that involves transforming the features of a dataset to a common scale.
Standardization guarantees that the features have identical ranges and units, allowing for comparison and useful analysis.
When data is standardized, it is much easier to detect errors and ensure that it is accurate. This is essential for making sure that decision-makers have access to accurate and reliable information, so it is essential for preserving data quality.

- ### Importance

  Standardizing features in a dataset is a common practice in data analysis for several reasons:

**Interpretability:** Standardized features are more interpretable in the context of the dataset. Since all features have a mean of 0 and a standard deviation of 1 after standardization, the coefficients or weights obtained from models can be directly compared to assess the relative importance or impact of each feature.

**Scale normalization:** Features in a dataset can have different scales, ranges, or units of measurement. Standardizing the features brings them to a common scale, ensuring that no single feature dominates the analysis simply because of its larger magnitude. So, It allows for fairer comparisons and prevents biased results.

**Removal of units:** Standardizing features removes the units of measurement, making the data dimensionless. This can be particularly useful when dealing with algorithms that are sensitive to the scale of the features. such as distance-based methods (e.g., k-means clustering) .

**Outlier resilience:** Standardization can also enhance the robustness of the analysis against outliers. When features are standardized, extreme values that lie far outside the typical range of values are less likely to have a disproportionate impact on the analysis compared to non-standardized data.

- ## Things to be done before data standrization

  Before performing data standardization, it's important to consider a few preliminary steps:

**Data cleaning:** When the dataset is free from errors, missing values, and outliers, it ensures that the standardization process is applied to reliable and accurate data.

**Data exploration and analysis:** it's crucial to explore and analyze the dataset, understand the distribution of the features and gain insights into the relationships between variables, before standardizing the features.

**Addressing outliers:** It is important to decide how to handle outliers in your dataset, before standardization. Outliers can significantly affect the mean and standard deviation, which are crucial in the standardization process.
By following these steps, you can ensure that your dataset is prepared appropriately before applying data standardization techniques. This helps in obtaining reliable and meaningful results from your analysis.

- ## Methods of standrization

There are several methods for data standardization, each with its own advantages and applicability. Here are some commonly used methods:

**Z-Score Standardization:** Also known as standard score normalization, this method scales the data to have a mean of 0 and a standard deviation of 1.
The formula for z-score standardization is: $\mathbf{z = (x - \mu) / \sigma}$
where z is the standardized value, x is the original value of the feature, μ is the mean of the feature, and σ is the standard deviation of the feature.
Z-score standardization is widely used when the data follows a normal distribution and when you want to compare and interpret the values based on their deviation from the mean.

**Min-Max Scaling:** This method scales the data to a specific range, typically between 0 and 1.
The formula for min-max scaling is: **x_scaled = (x - min) / (max - min)**
where x_scaled is the scaled value, x is the original value of the feature, min is the minimum value of the feature, and max is the maximum value of the feature.
Min-max scaling preserves the relative relationships between the values while compressing the range to a specific interval. It is commonly used when you want to maintain the original distribution of the data and when the data does not necessarily follow a normal distribution.

**Decimal Scaling:** In this method, the data is divided by a power of 10 such that the maximum absolute value of the resulting data is between 1 and 10. For example, if the maximum absolute value in the dataset is 789, dividing all the values by 1000 would result in values ranging between 0.001 and 0.789.
Decimal scaling is useful when you want to avoid extremely small or large values in your data, particularly in situations where precision matters, such as numerical computations or machine learning algorithms.

**Robust Scaling:** Robust scaling is a method that standardizes the data by subtracting the median and dividing by the interquartile range (IQR). This approach is less affected by outliers and extreme values compared to z-score standardization.
Robust scaling is preferred when the dataset contains outliers or when the data does not follow a normal distribution.

The choice of which method to use depends on the specific characteristics of your dataset, the requirements of your analysis, and the algorithms you plan to use. It's important to understand the nature of your data and select the appropriate standardization method accordingly.
And for that, it was best to choose the Z-score standardization in order to standardize the features of our dataset.

So, by standardizing the data, we ensure that our analysis is based on reliable and comparable features, improving the accuracy, interpretability, and efficiency of your results. It enables more robust modeling, enhances algorithm performance, and facilitates fair comparisons between features, leading to better insights and decision-making.

## 2.5-Splitting the data set

Splitting the data is a crucial step in data analysis, especially when building machine learning models or performing statistical analysis. It involves dividing the dataset into separate subsets for training and evaluation. The most common type of data splitting is into training and testing sets. Here's a brief explanation of the different types of data splitting:

1. Training data: The training data is used to train the model or build statistical models. It is the largest subset of the data and is used to learn the patterns, relationships, and parameters of the model. The model is trained on this set by adjusting its parameters based on the input features and corresponding target variables.
2. Testing data: The testing data is used to evaluate the performance of the trained model or statistical models. It is a separate subset of the data that the model has not seen during training. The testing data is used to assess how well the model generalizes to new, unseen data. The evaluation metrics calculated on the testing dataprovide an estimate of the model's performance on real-world data.

The purpose of splitting the data is to assess the model's performance on unseen data and avoid overfitting, where the model performs well on the training data but fails to generalize to new data. It helps to evaluate the model's ability to capture underlying patterns and make accurate predictions or inferences on new observations.
It is important to maintain the integrity of the data split by ensuring that the training, and testing data are mutually exclusive and representative of the overall dataset. Common splitting ratios include 70-30, 80-20, (Training, testing) percentages, depending on the size of the dataset and specific requirements of the analysis.
And in our model, we split our data set into a 80% partition, which is the training data, and a 20% partition, which is the testing data.
So, by splitting the data we can assess the performance and reliability of our model and make informed decisions about its generalization capabilities.

## 2.6-Naive Bayes

It is a probabilistic classifier based upon Bayes theorem using features from a data set relying on an assumption about the independence between the features and another assumption that all features contribute equally in affecting the outcome. but this model cant differentiate between its features so it can't give a priority for a feature over another feature .

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

P(Y|X) x for each feature in the data set which is named posterior probability

P(Y) which is named prior probability

P(X|Y) x for each feature which is named likelihood probability

P(X) which is named evidence and in sometimes can be ignored by increasing numerator

There are several types of naive bayes models which are :

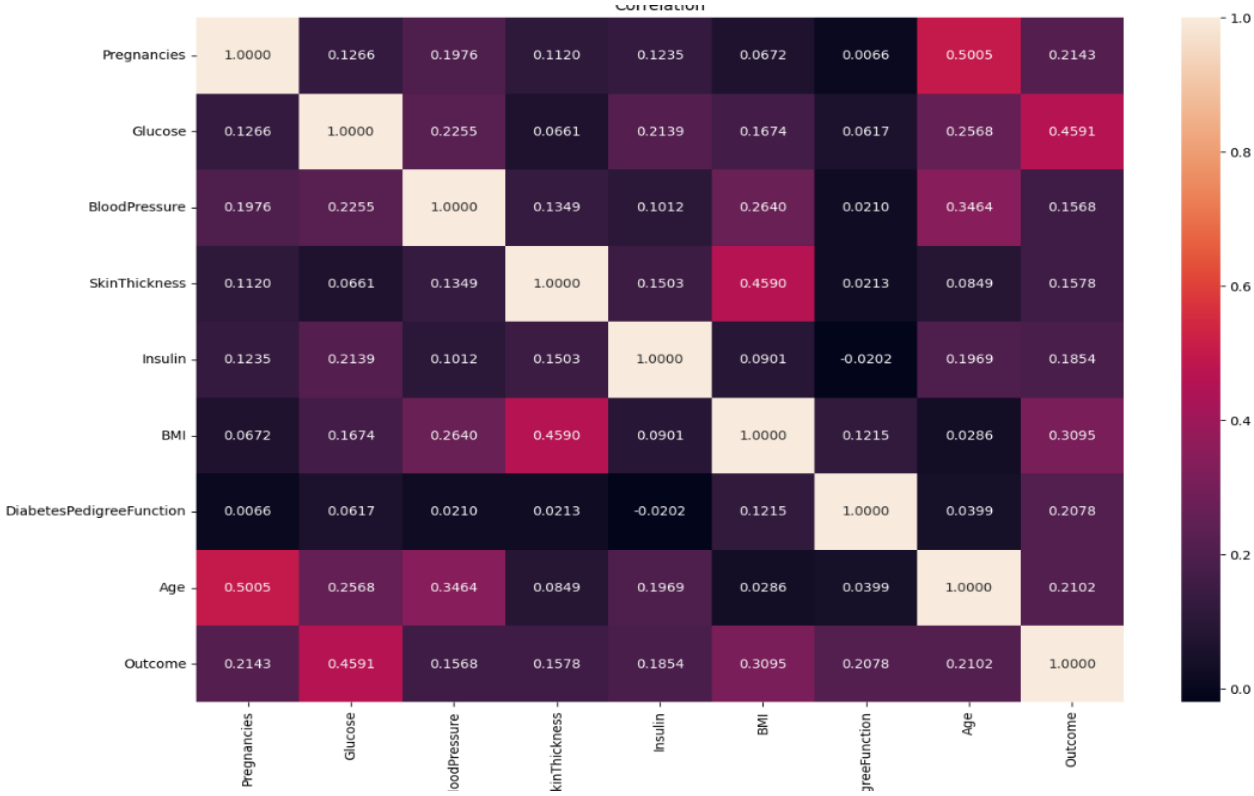
- Multinomial NB : which is a Bayesian learning approach model that is applied to countable features which are either discrete or categorical using Bayes theorem.
- Gaussian NB : which is bayesian learning approach model that is following normal distribution at which class conditional probability using mean and variance and take the higher probability as the predicted
- Categorical NB : it is suitable for categorical features and then conclude the posterior conditional probability

In our model our data is not discrete so applying multinomial NB or categorical NB is not possible but gaussian NB could be applied as our data is standardized to approximate normal distribution .

## Steps through gaussian NB in our model:

1. Remove dependent features

   In order to remove the dependent features we need to plot a heat map to see the value of correlation coefficient between each two features in our model. The relation between the features is very low so no features were needed to be removed




2. Check the distribution of each of our feature

   Most of them were approximately normally distributed

3. Implement prior probability of outcome function

   Our data set outcome is either if the person has diabetes which valued as 1 or not which is valued as 0, so in our case the outcome is binary so prior probability will be number of specific outcome divided by total number of outcomes

4. Implement likelihood probability function

- The likelihood probability is the probability of a certain feature given that the outcome is known so , in our code we calculate the probability of a certain feature but by decreasing the sample space to the needed outcome and by using the gaussian probability rule we obtain single feature probability
- Since the assumption of independency is applied the likelihood probability will be the result of multiplying each conditional probability of each feature

5. Naive bayes gaussian function

   Last but not least implementing the naive bayes function at which goes through several steps which are

   - First we calculate the the prior probability for both classes
   - Second we calculate likelihood probability conditioned on both classes
   - Third we multiply prior probability by likelihood both for the same class and we do this for both classes
   - Lastly we take the higher probability from both as our predicted value

# 3-Results and discussion

In this section, we present the results of our evaluation metrics for the Naive Bayes Classifier. We discuss the performance of the classifier based on the following metrics: **Accuracy, Precision, Recall, F1-Score, Specificity, and Balanced Accuracy.**

## Accuracy Score:

The Naive Bayes Classifier achieved an accuracy of **81.51%**. This indicates that the classifier correctly classified approximately 81.51% of the samples in our dataset. It is important to note that accuracy alone may not provide a complete picture of the classifier's performance, as it does not consider the balance between true positives and true negatives.

## Precision:

The precision of the classifier was found to be **70.73%**. Precision measures the proportion of true positive predictions out of the total positive predictions made by the classifier. A higher precision score indicates a lower rate of false positive predictions. In our case, the classifier achieved a precision of 70.73%, which suggests that it has a moderate ability to correctly identify positive samples.

## Recall:

The recall, also known as sensitivity, represents the proportion of true positive predictions out of the total actual positive samples in the dataset. The Naive Bayes Classifier obtained a recall score of **74.36%**, indicating that it can successfully identify a significant portion of the positive samples in the dataset.

## F1-Score:

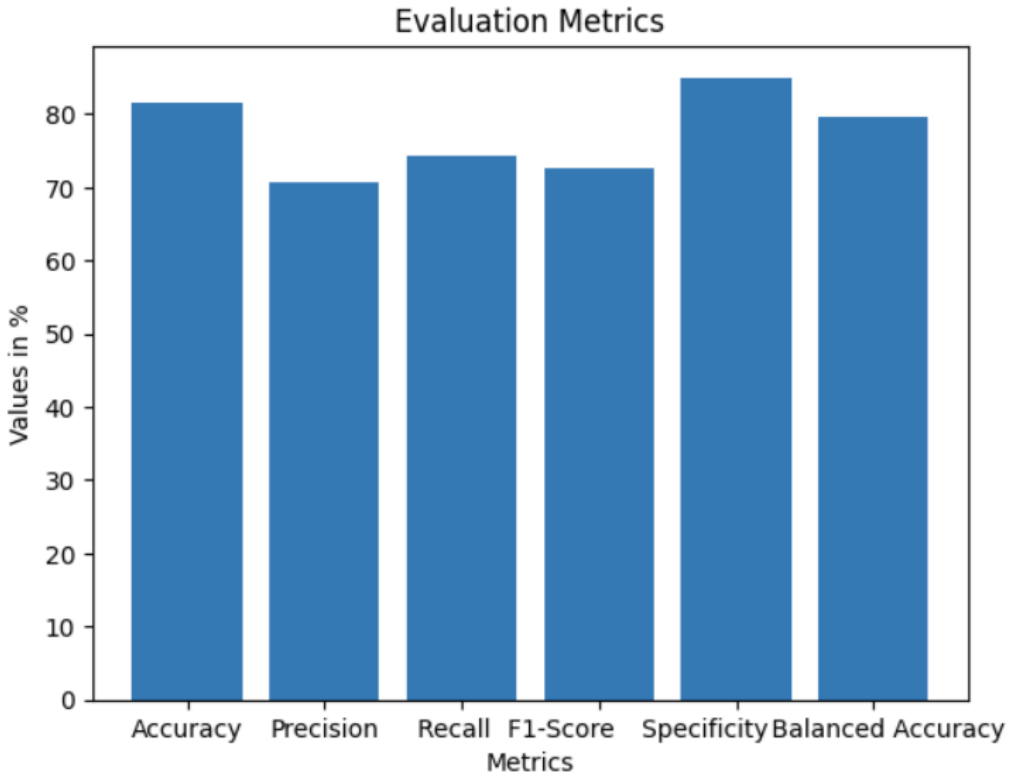
The F1-Score is a measure that combines precision and recall into a single value, providing an overall assessment of the classifier's performance. The Naive Bayes Classifier achieved an F1-Score of **72.50%**, which suggests a reasonable balance between precision and recall.

## Specificity:

Specificity measures the proportion of true negative predictions out of the total actual negative samples. In our case, the classifier obtained a specificity of **85.00%**, indicating its ability to correctly identify negative samples.

## Balanced Accuracy:

We evaluated the classifier's performance using balanced accuracy. Balanced accuracy is particularly useful when dealing with imbalanced datasets, as it calculates the average recall for all classes. The Naive Bayes Classifier achieved a balanced accuracy of **79.68%**, reflecting its ability to provide a fair evaluation across different classes.



Overall, the Naive Bayes Classifier demonstrated promising performance in our evaluation. While it achieved a relatively high accuracy of **81.51%**, it is essential to consider other metrics such as precision, recall, F1-Score, specificity, and balanced accuracy to obtain a more comprehensive understanding of its strengths and weaknesses. Further analysis and experimentation may be required to optimize the classifier's performance and address any potential limitations identified in this evaluation.

# 4-Conclusion

**Our objective** was to create a model to classify if a person has diabetes or not based on some given features about this person.

**Our finding key** was to first study our data set to identify the null value and the outliers after identifying both we have to handle them each one separately first the null values was substituted by the mean of the given data set and then the outliers is handled by interquartile method by increasing the threshold in order to get rid of the extreme from them .

After handling our data set we standardized it using the z-score method. Then in order to train our model we had to split our data into two groups which are test data and train data by the ratio of 0.2 and 0.8 respectively.

Finally we used gaussian naive bayes which is a probabilistic classifier by taking it step by step first a calculating the prior probability for both classes then calculating likelihood probability by calculating conditional probability of each feature on each class then by using the independence assumption we multiply all of them together to get the likelihood the predicted class will result from taking the higher value from multiplying likelihood by prior of each class.

**Implications and significance** was that our model has an accuracy same as NB classifier from python packages and it was at a value of 81.51% which is a good percent relative to our small sample of data set . There were several metrics that we did to trace our model performance which were precision ,recall ,f1-score,specificity,balanced accuracy. And then a confusion matrix was shown to show the number of true positives ,false positive ,true negative and false negative.

**Limitations** we faced were the limited number of sample we had that were reduced after cleaning the data so a bigger sample size and more info in our data set(no null value) would give a better model with a better accuracy.also there were some approximation made in the distribution as it wasn't perfectly  normal so there was another classifiers that can deal with non normal distribution like both logistic regression and SVM can handle continuous features and are

commonly used for binary classification tasks like predicting diabetes. Logistic regression models the relationship between the features and the outcome using a logistic function, while SVM tries to find a hyperplane that separates the two classes with the maximum margin.

# 5-Contributions

In this section, we provide an overview of the contributions made in different sections of our project.

**Data Wrangling**:

The data wrangling section involved several tasks to ensure the data is prepared for further analysis. This included loading the CSV data into a pandas data frame, performing data cleaning operations such as removing duplicates and outliers, renaming variables, and handling null values. Additionally, the data was split randomly into training data (80%) and testing data (20%). Finally, the features were standardized using calculated descriptive statistics.

**Visualization, EDA, and Data Description**:

The visualization, exploratory data analysis (EDA), and data description section aimed to gain insights into the dataset. The variables were categorized as quantitative or categorical. Descriptive statistics, including measures of central tendency and dispersion, were calculated to quantitatively describe the data. For each feature/column in the training data, histograms or distribution plots were generated to visualize their distributions. Statistical tests were conducted to determine if a feature/column followed a specific distribution, and conditional distributions of each feature on each target class (label) were plotted.

**Implementation of Naive Bayes**:

The implementation of the Naive Bayes classifier involved building the classifier using the preprocessed and standardized training data. This included implementing the necessary algorithms and techniques to enable accurate classification. The trained classifier was then used to predict labels for the testing data.

**Evaluation Metrics**:

The evaluation metrics section focused on assessing the performance of the Naive Bayes classifier. Various metrics, including accuracy, precision, recall, F1-Score, specificity, and balanced accuracy, were calculated. These metrics provided insights into the classifier's overall accuracy, precision in positive predictions, recall of positive samples, the harmonic mean of precision and recall, the ability to correctly identify negative samples, and a fair evaluation across different classes, respectively.

- Ahmed Mohamed Ali:

- Data wrangling
- Implementation of Naive Bayes
- Evaluation Metrics

- Ali Badran:
  - Data wrangling
  - Visualization, EDA, and Data Description
  - Implementation of Naive Bayes

- Muhannad Abdullah:
  - Implementation of Naive Bayes
  - Evaluation Metrics
  - Visualization, EDA, and Data Description

- Hesham Tamer:
  - Implementation of Naive Bayes
  - Visualization, EDA, and Data Description
  - Data wrangling

- Youssef Ashraf Mohamed:
  - Data wrangling
  - Visualization, EDA, and Data Description
  - Evaluation Metrics