**Naive Bayes classifier**

**Abstract:** This project implements a Naive Bayes classifier to analyze the Pima Indians Diabetes Database. After preprocessing the text and splitting the data into training and testing sets, the model is trained and evaluated for accuracy, aiming to predict a target variable.

**Data Understanding:**

**A screenshot of a computer

Description automatically generated**

In this project, we analyzed the 'diabetes.csv' file to better understand the dataset's features and structure. The dataset comprises 768 entries and 9 columns, each representing a different aspect of patients' health relevant to diabetes prediction. The columns include:

* **Pregnancies**: The number of times the patient has been pregnant.
* **Glucose**: Plasma glucose concentration a couple of hours after a glucose challenge.
* **BloodPressure**: Diastolic blood pressure (mm Hg).
* **SkinThickness**: Triceps skin fold thickness (mm).
* **Insulin**: 2-Hour serum insulin (mu U/ml).
* **BMI**: Body mass index (weight in kg/(height in m)^2).
* **DiabetesPedigreeFunction**: A function that scores the likelihood of diabetes based on family history.
* **Age**: Age of the patient.
* **Outcome**: Indicates whether the patient has diabetes (1) or not (0).

A screenshot of a computer

Description automatically generated

Using the df.info() method, we confirmed that all columns contain 768 non-null values, indicating no missing data. The data types varied, with 7 columns as integers and 2 as floats, which is essential for understanding the numerical nature of the data and the types of analyses we can perform.

This understanding of the dataset's structure and contents is crucial for preparing the data for modeling and subsequent analysis. It allows us to identify potential relationships between features and the outcome, guiding our approach to further data preparation and modeling phases.

**Data Processing Steps:**

1. Inspect Data: Use df.info() to check data structure and missing values

2. Separate Features and Target:

• X = df.drop('Outcome', axis=1) for features

• y = df['Outcome'] for target variable

3. Train-Test Split: Use train\_test\_split to divide data, 33% for testing

**Model Choice:**

• Gaussian Naive Bayes: used when the features are continuous and it assumes that the continuous features follow a normal distribution (Gaussian distribution) within each class

**Performance Evaluation**

**Metrics:**

• Accuracy: Proportion of correct predictions

• F1 Score: Balances precision and recall

• Confusion Matrix: Visualizes performance, showing true and false classifications

A chart of different colors

Description automatically generated

**Insights Gained:**

The confusion matrix shows the model has a 66% accuracy and 80% precision, meaning it usually predicts positives correctly. However, with a 70% recall, it misses 51 real positive cases, which is worrying for health diagnoses. There are also 31 false positives that could cause unnecessary concern. Overall, the model needs to improve in finding more true positives.