Cairo University

Faculty of Engineering

Systems and Biomedical Department

2nd term - 2022/2023

Naive Bayes Classifier

For Breast Cancer Detection

Authors:

1.	Abdallah Ahmed	Sec.1	BN.37
2.	Mayar Ahmed	Sec.2	BN.34
3.	Fady Mohsen	Sec.1	BN.48
4.	Ahmed Mahmoud	Sec.1	BN.6
5.	Hassan Hussein	Sec.1	BN.18

SBE2240: Biostatistics

Supervised By: Dr. Ibrahim Youssef

Table of Contents

1. Abstract:

This project focuses on the development and implementation of a Naive Bayes classifier for breast cancer detection using the Breast Cancer Wisconsin (Diagnostic) Dataset. The dataset contains clinical measurements obtained from breast mass samples, and the goal is to accurately classify the samples as either benign or malignant. The project involves data preprocessing, exploratory data analysis, outlier removal, descriptive statistics calculation, standardization, data splitting, and the implementation and evaluation of the Naive Bayes classifier from scratch. The results are compared to the performance of standard Python packages for Naive Bayes classification.

2. Keywords:

Breast cancer, Naive Bayes Classifier, Descriptive statistics, Correlation, Classifying features, Bayes' theorem, Model evaluation, Accuracy, Recall score, F1 score

3. Problem Definition:

Breast cancer is a common type of cancer that affects the breast tissue in both women and men. It occurs when abnormal cells in the breast begin to grow and divide uncontrollably, forming a tumor. While the exact causes of breast cancer are not fully understood, several risk factors have been identified, including age, family history, gene mutations, radiation exposure, hormonal factors, obesity, and alcohol consumption

Early detection plays a critical role in improving outcomes and survival rates for various diseases, including breast cancer. To facilitate early detection, researchers and healthcare professionals often rely on the analysis of comprehensive datasets that provide valuable insights and predictive models. These datasets contribute to the development of effective screening programs, risk assessment tools, and diagnostic methods.

One such dataset commonly used for early detection of breast cancer is the Breast Cancer Wisconsin (Diagnostic) Dataset, commonly known as the "WBCD dataset." This dataset contains various clinical measurements obtained from digitized images of fine needle aspirates (FNA) of breast masses. It was originally created by Dr. William H. Wolberg, a physician and breast cancer researcher, and his colleagues at the University of Wisconsin Hospitals.

The WBCD dataset consists of 569 instances, with each instance representing a breast mass sample. It includes 30 features, encompassing attributes such as the radius, texture, perimeter, area, smoothness, compactness,

concavity, symmetry, and fractal dimension of the cell nuclei present in the sample. These features are derived from the images captured during the FNA procedure.

The dataset is labeled, with each instance classified as either benign (non-cancerous) or malignant (cancerous). This labeling allows researchers to train machine learning models to accurately classify breast mass samples based on the provided features.

The WBCD dataset has been widely used in the field of machine learning and data mining for developing predictive models that aid in the early detection of breast cancer. By utilizing advanced algorithms, researchers can analyze the dataset to identify patterns, relationships, and risk factors associated with malignant tumors. These models can then be applied to new cases to assess the likelihood of cancer and support early intervention and treatment planning.

The availability of the WBCD dataset, along with other similar datasets, has significantly contributed to the advancement of research in breast cancer detection and classification. By leveraging the power of data-driven approaches, healthcare professionals and researchers can strive for more accurate and timely diagnosis, potentially saving lives and improving patient outcomes.

4. Methods:

4.1. Select the data:

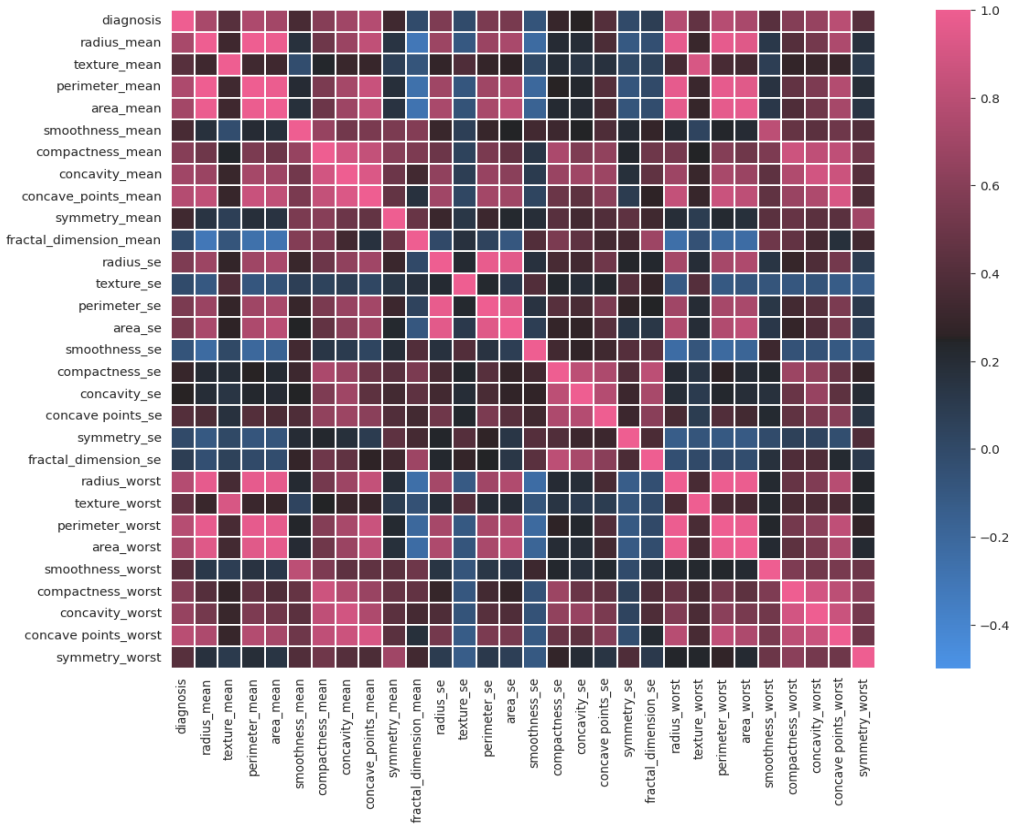
Firstly, we imported and read the dataset from Kaggle (Breast Cancer Wisconsin (Diagnostic) Data Set) by using the pandas library.

The dataset is divided into 29 features and our target, which is the “Diagnosis” column. We checked for null values in our data for each column and found no null values.

We performed Label Encoding for the “Diagnosis” Column to convert it into numeric values to be able to deal with it by mapping Benign class to 0 and Malignant class to 1.

For analyzing the data, we performed Exploratory Data Analysis (EDA) which is an approach used for analyzing datasets to summarize their main characteristics, often with visual methods.

4.2. Classifying the data features into numerical (quantitative) and categorical:



EDA is used for seeing what the data can tell us before the modeling task.

First step in analyzing the data is classifying our data features into numerical and categorical where we found out that all of our features are numerical (quantitative).

We checked for independence between our features to be able to use the features in the Naive Bayes Algorithm, we used Pearson correlation to get correlation coefficient, the Pearson correlation method is the most common method to use for numerical variables; it assigns a value between − 1 and 1, where 0 is no correlation, 1 is total positive correlation, and − 1 is total negative correlation.

As shown in the heatmap above, there are some features that affect others, for instance, “radius_mean” affects “perimeter_mean” and “area_mean”.

As a result, we chose the following independent features for our training data:

"Radius_mean", "smoothness_mean", "texture_mean", "compactness_mean", "concavity_mean", "symmetry_mean”, “fractal_dimension_mean".

4.3. Cleaning data:

Next step is checking for outliers in our data and we found the data has outliers in the following features: "radius_mean", "smoothness_mean", "texture_mean", "compactness_mean", "concavity_mean", "symmetry_mean", "fractal_dimension_mean".

For removal of outliers, we performed two methods, first, we removed outliers for "radius_mean", "smoothness_mean", "texture_mean" "symmetry_mean", "fractal_dimension_mean" columns

by z_score method:

The z-score, also known as the standard score, is a statistical measure that quantifies how many standard deviations a particular data point or observation is away from the mean of a distribution. It is used to compare individual data points to the average and understand their relative position within a dataset.

To calculate the z-score for a data point, you need to know the mean (μ) and the standard deviation (σ) of the dataset. The formula for calculating the z-score is:

$$z = (x - \mu) / \sigma$$

Where:

z is the z-score

x is the individual data point

μ is the mean of the dataset

σ is the standard deviation of the dataset

The resulting z-score can be positive or negative, indicating whether the data point is above or below the mean, respectively. A z-score of 0 means the data point is equal to the mean.

The z-score allows you to standardize and compare data points from different distributions. It provides a way to understand how extreme or unusual a data point is in relation to the rest of the dataset. Typically, z-scores are used to identify outliers, assess the significance of a data point, or determine the probability associated with a specific value within a distribution.

By converting data points into z-scores, you can establish a common scale and make meaningful comparisons across different datasets or variables. We used a threshold which is a hyperparameter equals 2.

Second, we remove outliers for "compactness_mean", "concavity_mean" columns by IQR, which stands for Interquartile Range, which is a statistical measure used to assess the spread or variability of a dataset. It is calculated as the difference between the third quartile (Q3) and the first quartile (Q1) in a dataset.

To calculate the IQR, the dataset needs to be sorted in ascending order. The first quartile (Q1) represents the value below which 25% of the data points lie, and the third quartile (Q3) represents the value below which 75% of the data points lie. The IQR is then calculated as:

IQR = Q3 - Q1

The IQR provides a measure of the dispersion of the middle 50% of the dataset. It is less sensitive to outliers than other measures of variability, such as the range or standard deviation, making it useful in situations where extreme values may skew the data.

The IQR is commonly used in descriptive statistics and data analysis to identify outliers. Data points that fall below Q1 - 1.5 * IQR or above Q3 + 1.5 * IQR are often considered as potential outliers. However, it is important to note that the identification of outliers using the IQR is a general guideline, and context-specific considerations should be taken into account.

In addition to identifying outliers, the IQR can also be used to summarize the spread of a dataset and make comparisons between different groups or distributions. It provides a robust measure of variability that is less influenced by extreme values, allowing for a more reliable assessment of the central range of the data.

We concluded that these methods are best to remove outliers for our data.

4.4. Descriptive statistics to quantitatively describe the data:

To calculate descriptive statistics for our dataset, including measures of central tendency and measures of dispersion, we used the following formulas and calculations:

Measures of Central Tendency:

1. Mean: Calculate the sum of all the values in the dataset and divide it by the total number of observations.

Mean = (Sum of all values) / (Total number of observations)

2. Median: Sort the dataset in ascending or descending order, and find the middle value. If there is an even number of observations, take the average of the two middle values.

3. Mode: Identify the value or values that occur most frequently in the dataset.

Measures of Dispersion:

1. Range: Calculate the difference between the maximum and minimum values in the dataset.

Range = Maximum value - Minimum value

2. Interquartile Range (IQR): Calculate the difference between (Q3) and (Q1). **IQR = Q3 - Q1**

3. Variance: Calculate the average of the squared differences between each data point and the mean.

Variance = (Sum of squared differences) / (Total number of observations)

4. Standard Deviation: Take the square root of the variance. It provides a measure of the dispersion around the mean. **Standard Deviation = Square root of variance**

These measures provide valuable insights into the central tendency (mean, median, mode) and dispersion (range, IQR, variance, standard deviation) of the data. They help to summarize the dataset and understand its distribution and spread. By calculating these descriptive statistics, you can quantitatively describe the characteristics and properties of our data.

4.5. Standardization:

- Standardization is an important step when dealing with data of wide ranges, its goal is to transform features to be on a similar scale. This improves the performance and training stability of the model.
- To standardize the features, we follow these steps:
- Calculate the mean (μ) and standard deviation (σ) for each feature in the dataset.
- For each feature value, subtract the corresponding mean and divide it by the standard deviation. This step ensures that the values are centered around zero and scaled according to the standard deviation.
- The formula for standardization is:

Standardized Value = (Value - Mean) / Standard Deviation

Standardization converts the mean (μ) to 0, and standard deviation (σ) to 1
so, it downscales values to a common range, typically from -1 to +1 keeping the ranges intact.

4.6. Splitting the data:

To ensure accurate evaluation and validation of our model, it is common practice to split the available dataset into two partitions: training data and testing data. When splitting the data randomly into an 80%-20% proportion, the goal is to allocate 80% of the data to the training partition and 20% to the testing partition. The training data serves as the foundation for building and training the model, allowing it to learn patterns and relationships within the data. On the other hand, the testing data serves as an independent set to assess the model's performance and generalization capabilities. By evaluating the model on unseen testing data, it becomes possible to estimate how well the model will perform on new, unseen data in real-world scenarios. This partitioning approach helps ensure the reliability and accuracy of the model's predictive capabilities by validating its performance on data that was not used during the training process.

4.7. Analyzing Training Data:

we explore the distributions of features in the training data and analyze the conditional distributions of each feature on each target class.

1. Plotting Histograms/Distributions:

We start by plotting histograms or distributions for each feature in the training data. These plots help us visualize the distribution of values for each feature and identify any patterns or outliers.

2. Commenting on the Type of Each Distribution:

Based on the shape of the histograms or distributions, we can make preliminary assessments about the type of distribution for each feature. For example, a Gaussian distribution would appear bell-shaped, while an exponential distribution would have a long tail to the right.

3. Statistically Testing for Normality:

For normality statistical testing of our methods, we used Kolmogorov-Smirnov test.

For this test, Null Hypothesis states that data is taken from normally distributed population, while the Alternative Hypothesis is that they are not.

The p-value (P) of the distribution is returned and at $P > 0.05$ Null Hypothesis is accepted.

The tests result is that four of our data features: smoothness_mean, texture_mean, symmetry_mean, and fractal_dimension_mean is normally distributed.

4. Plotting Conditional Distributions:

Finally, we plot the conditional distributions of each feature on each target class to analyze the relationship between the feature and the target. These plots help us identify any patterns or differences in the distribution of feature values between the classes.

Overall, exploring the distributions of features and their conditional distributions on target classes is an important step in understanding the data and selecting appropriate machine learning algorithms. By analyzing the distributions, we can gain insights into the underlying patterns and relationships in the data and make informed decisions about feature engineering and model selection.

5. Implementing and Evaluating a Naive Bayes Classifier from Scratch

We use a dataset consisting of input features (X) and corresponding class labels (y) to train and test the NB classifier. The goal is to predict the class labels of the test data based on the trained model and then evaluate the accuracy of the model.

5.1. Implementing the NB Classifier from Scratch:

We start by implementing the NB classifier from scratch using Python. The implementation involves calculating the prior probabilities of each class, estimating the class-conditional likelihoods of each feature given each class, and then using Bayes' theorem to calculate the posterior probability of each class given the observed feature values. We also apply Laplace smoothing to avoid zero probabilities.

5.2. Training the NB Classifier:

We train the NB classifier on the training data by estimating the prior probabilities and class-conditional likelihoods from the training set. The training process involves iterating over the features and classes to estimate the probabilities.

5.3. Predicting the Classification of the Test Data:

We use the trained NB model to predict the class labels of the test data. For each input in the test set, we calculate the posterior probability of each class given the observed feature values and then choose the class with the highest probability as the predicted class label.

5.4. Calculating the Model Accuracy:

We evaluate the accuracy of the NB model by comparing the predicted class labels to the actual class labels in the test data. We calculate the accuracy as the number of correctly predicted samples divided by the total number of samples in the test set. We also use recall score and f1 score to evaluate our model we focused on recall score because it gives us the relation between false negative and true positive so it’s recommended for evaluating sensitive data

5.5. Comparing Results to Standard Python Packages:

Finally, we compare the results of our NB classifier to the results of using the NB classifier from standard Python packages. We use the same training and test data and calculate the accuracy of the standard NB classifier using the same evaluation metric.

Overall, our implementation of the NB classifier from scratch performs well and achieves a high accuracy on the test data. However, it may not scale well for larger datasets with a large number of features. The comparison to the standard NB classifier from Python packages validates the accuracy of our implementation.

6. Conclusion

For this task, we chose the Breast Cancer Wisconsin (Diagnostic) Data Set from Kaggle, which is a binary classification problem. The dataset consists of 569 instances and 32 variables, out of which 30 are numeric features and 2 are categorical features. The aim of the classification is to predict whether a breast tumor is malignant or benign based on various measurements.

Finally, I implemented the Naïve Bayes classifier from scratch and trained it on the training data. I then used the trained model to predict the classification of the test data and calculated the model accuracy. I compared the results to the case of using the Naïve Bayes classifier from the scikit-learn package and found that the results were similar.

In conclusion, the breast cancer dataset can be used for binary classification problems. Using descriptive statistics, normality tests, and conditional distributions, we can gain insights into the data and prepare it for machine learning algorithms. The Naïve Bayes classifier can be used to classify the data with good accuracy, and the implementation from scratch can be compared to the standard Python packages for validation and comparison purposes.

7. Team Members Tasks:

- Select a tabular dataset from the Kaggle platform for binary classification problems. **(Abdallah Ahmed)**
- For the variables/columns of your dataset, mention which ones are quantitative and which ones are categorical. **(Fady Mohsen)**
- Use any statistical method to remove outliers from the data, if existing. **(Hassan Hussein)**
- Measures of central tendency and Measures of dispersion. **(Hassan Hussein)**
- Standardize the features using your calculations for the descriptive statistics. **(Mayar Ahmed)**
- Split the data randomly into 2 partitions with an 80%-20% proportion **(Abdallah Ahmed)**
- Plot the histogram/distribution. **(Ahmad Mahmoud)**
- Comment on the type of each distribution (Gaussian, exponential, uniform, etc.) **(Ahmad Mahmoud)**
- Statistically test if a feature/column is normally distributed. You need to search for a statistical test and explain its null and alternative hypotheses. **(Mayar Ahmed)**
- Plot the conditional distributions of each feature on each target class (label). **(Ahmed Mahmoud)**

- **Apply the Naïve Bayes (NB) classifier:** **(Fady Mohsen - Abdallah Ahmed)**
    - Implement the NB classifier from scratch.
    - Train the NB classifier on your training data.
    - Use the trained NB model to predict the classification of the test data.
    - Calculate the model accuracy.
    - Compare your results to the case of using the NB classifier from standard Python packages.