Classification of Breast Cancer Tumors Using Mathematical Tools in Artificial Intelligence

Samuel Coriat

May 27, 2025

Contents

"Artificial intelligence is the future, and the future is now."Gary Kasparov

1 Introduction: Explanation of the Work and Basic AI

1.1 A Brief History of AI

The 1940s and 1950s witnessed the emergence of the first true computers, machines capable of performing a significant number of simple calculations per second. This era laid the groundwork for what would eventually become the field of Artificial Intelligence. Early pioneers like Alan Turing explored the theoretical possibility of thinking machines, posing the famous "Turing Test" as a benchmark for machine intelligence. The Dartmouth Workshop in 1956 is widely considered the birth of AI as an academic discipline, bringing together researchers who envisioned creating machines that could reason, solve problems, and understand language. Early enthusiasm led to the development of symbolic AI and logic-based programming, with the hope of achieving human-level intelligence relatively quickly.

1.2 Briefly, How Does an AI Work?

An AI fundamentally operates by processing vast quantities of data to discern patterns and relationships – this phase is known as "learning" or "training." During this stage, the AI algorithm adjusts its internal parameters based on the input data. Subsequently, the AI is evaluated using a "validation set." Its responses are analyzed, and its performance is measured against expected outcomes. Based on this evaluation, the AI's model is refined and corrected to improve its accuracy and reliability. Finally, a separate "test set," which the AI has never encountered during training or validation, is used to provide an unbiased assessment of the AI's generalization capabilities, ensuring that it has genuinely learned the underlying principles and can apply them to new, unseen data. This rigorous process helps confirm that the AI has not simply memorized the training data but truly understands the task at hand

2 Data set

This work utilizes the "Breast Cancer" data set (link : `https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnosticdataset`) We will develop a Python algorithm to analyze this dataset. The dataset contains 31 parameters for each patient. The data encompasses information from 569 patients. (While this dataset is relatively small, it is sufficient for our analysis.)

NB: We assume the dataset is accurate and therefore contains no false diagnostic.

2.1 Presentation of the dataset

The dataset used in this project is a CSV (Comma-Separated Values) file named `data.csv`. It consists of rows and columns: each row corresponds to a single patient, while the columns represent 31 different features extracted from medical images. These features are crucial for the analysis developed in the following sections.

To understand the structure and type of data in this dataset, we use the function `df.info()`, which provides an overview of each column, including its name, data type, and the number of non-null values. The output is shown below:

Listing 1: output of df.info

```
Data columns (total 32 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   id                      569 non-null    int64
 1   diagnosis               569 non-null    object
 2   radius_mean             569 non-null    float64
 3   texture_mean            569 non-null    float64
 4   perimeter_mean          569 non-null    float64
 5   area_mean               569 non-null    float64
 6   smoothness_mean         569 non-null    float64
 7   compactness_mean        569 non-null    float64
 8   concavity_mean          569 non-null    float64
```

```
 9   concave_points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
 16  smoothness_se            569 non-null    float64
 17  compactness_se           569 non-null    float64
 18  concavity_se             569 non-null    float64
 19  concave_points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             569 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
 26  smoothness_worst         569 non-null    float64
 27  compactness_worst        569 non-null    float64
 28  concavity_worst          569 non-null    float64
 29  concave_points_worst     569 non-null    float64
 30  symmetry_worst           569 non-null    float64
 31  fractal_dimension_worst  569 non-null    float64
dtypes: float64(30), int64(1), object(1)
memory usage: 142.4+ KB
```

We can see that the data set is fairly well balanced, with 357 healthy patients and 212 sick patients.

Listing 2: output Healthy VS Sick patient

```

Healthy patients (B): 357
Sick patients (M): 212
```

we then look at which attributes have the greatest influence on the diagnosis

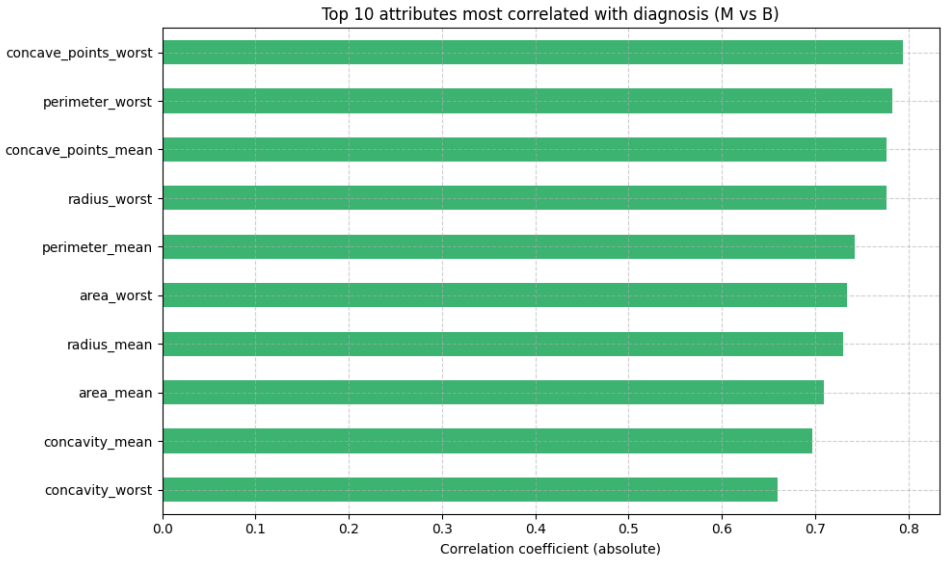

Figure 1: coef de correlation pour expliquer les liens les plus importants

Listing 3: Top 10 attributes most correlated with diagnos is (absolutevalues) :

```


concave_points_worst    0.793566
```

```
perimeter_worst          0.782914
concave_points_mean      0.776614
radius_worst             0.776454
perimeter_mean           0.742636
area_worst               0.733825
radius_mean              0.730029
area_mean                0.708984
concavity_mean           0.696360
concavity_worst          0.659610
```

2.2 Principal Component Analysis (PCA)

The dataset comprises 569 observations, each with 31 features, forming a 569×31 matrix where each data point exists in a 31-dimensional space, making direct visualization impractical. One feature, the categorical diagnosis ('M' for malignant, 'B' for benign), is converted to numerical values (1 for 'M', 0 for 'B') to enable numerical analysis, resulting in a purely numerical matrix of integers and floats. Principal Component Analysis (PCA), a statistical dimensionality reduction technique, transforms this high-dimensional data into a lower-dimensional coordinate system of orthogonal principal components, ordered by the variance they capture, preserving as much of the data's structure as possible for meaningful graphical representation in two or three dimensions

Mathematical Formulation Let us denote the dataset by a matrix $X \in \mathbb{R}^{n \times m}$, where $n = 569$ is the number of samples and $m = 30$ is the number of numerical features. (We exclude the `id` and diagnosis columns from the PCA computation.)

- First, we center the data by subtracting the mean of each column:

$$\tilde{X}_{ij} = X_{ij} - \mu_j$$

  where μ_j is the empirical mean of the j-th feature across all samples.

- Next, we compute the sample covariance matrix of the centered data:

$$\Sigma = \frac{1}{n-1} \tilde{X}^\top \tilde{X}$$

  where $\Sigma \in \mathbb{R}^{m \times m}$ is a symmetric, positive semi-definite matrix.

- We then perform an eigenvalue decomposition of Σ:

$$\Sigma \mathbf{v}_i = \lambda_i \mathbf{v}_i$$

  where λ_i are the eigenvalues and $\mathbf{v}_i$ the corresponding orthonormal eigenvectors (principal components). The eigenvalues represent the variance explained by each component. they are all positive due to the fact that the matrix is semi-positive definite.

- The eigenvalues are sorted in descending order:

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m \geq 0$$

  and we define the explained variance ratio as:

$$\text{Explained Variance Ratio} = \frac{\lambda_i}{\sum_{j=1}^{m} \lambda_j}$$

The first k eigenvectors form the new basis of the reduced space, and the data is projected onto this space via:

$$Z = \tilde{X} V_k$$

where $V_k = [\mathbf{v}_1 \, \mathbf{v}_2 \, \cdots \, \mathbf{v}_k] \in \mathbb{R}^{m \times k}$ and $Z \in \mathbb{R}^{n \times k}$ is the low-dimensional representation of the dataset.

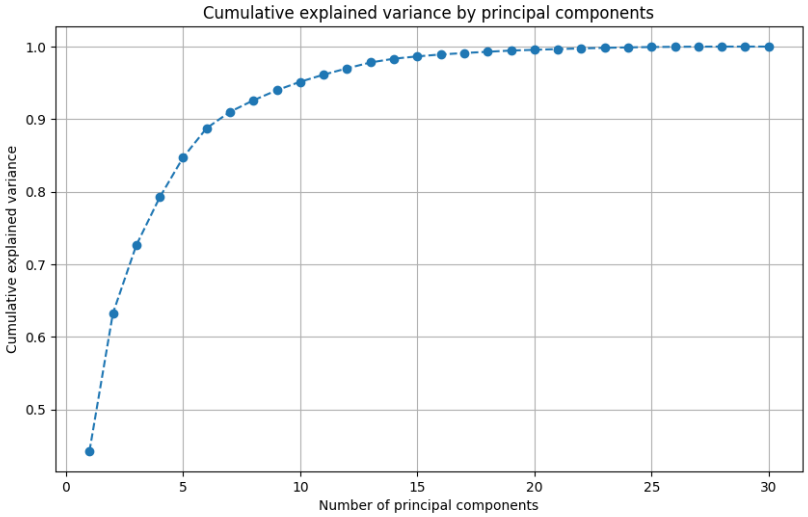

Figure 2: Cumulative explained variance as a function of the number of principal components

Variance Preservation and Choice of k In practice, we select the number k of principal components so that a desired proportion of the total variance is retained. In our analysis, we found that the first 10 principal components capture approximately 95% of the total variance, which is often considered a good trade-off between dimensionality and information retention.

Visualization and Interpretation By projecting the data onto the first three principal components, we obtain a 3-dimensional embedding that allows for graphical visualization of the dataset. Although PCA is unsupervised and does not consider class labels, we observe a partial separation between benign and malignant tumors in this space.

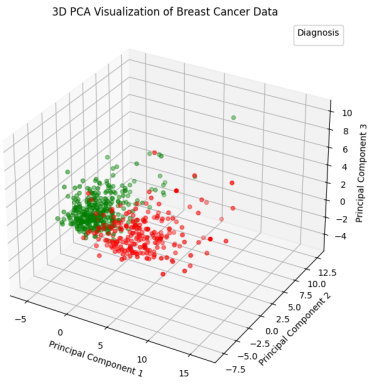

Figure 3: 3D PCA projection of the dataset, colored by diagnosis (benign vs. malignant)

This visualization reveals that the first few principal components capture structure that is relevant for distinguishing the two classes, even though PCA is purely based on variance and not class separability.

2.3 Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) is a dimensionality reduction technique primarily used in supervised classification problems. It aims to find a linear combination of features that best separates different classes by projecting the data onto a lower-dimensional space. The core idea behind LDA is to maximize the

between-class scatter while minimizing the within-class scatter. This allows for the most discriminative projection of the data.

The process involves the following steps:

2.3.1 Calculate the Mean Vectors

For a dataset with C classes, the mean vector $\boldsymbol{\mu}_i$ for the i-th class is calculated as:

$$\boldsymbol{\mu}_i = \frac{1}{N_i} \sum_{\mathbf{x}_j \in \text{class } i} \mathbf{x}_j$$

where:

- $\boldsymbol{\mu}_i$ is the mean vector of the i-th class,
- N_i is the number of data points in the i-th class,
- $\mathbf{x}_j$ represents the j-th data point belonging to the i-th class.

2.3.2 Calculate the Within-Class Scatter Matrix (S_W)

The within-class scatter matrix S_W measures the variance of data points within each class. It's calculated by summing the scatter matrices of individual classes.

The scatter matrix S_i for the i-th class is given by:

$$S_i = \sum_{\mathbf{x}_j \in \text{class } i} (\mathbf{x}_j - \boldsymbol{\mu}_i)(\mathbf{x}_j - \boldsymbol{\mu}_i)^T$$

where:
$\mathbf{x}_j$ represents the j-th data point in the i-th class, and $\boldsymbol{\mu}_i$ is the mean vector of the i-th class.
The within-class scatter matrix S_W is then the sum of the individual class scatter matrices:

$$S_W = \sum_{i=1}^{C} S_i$$

where C is the number of classes.

2.3.3 Calculate the Between-Class Scatter Matrix (S_B)

The between-class scatter matrix S_B measures the scatter between the mean vectors of different classes.

It's calculated as:

$$S_B = \sum_{i=1}^{C} N_i(\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T$$

where:
N_i is the number of data points in the i-th class,
$\boldsymbol{\mu}_i$ is the mean vector of the i-th class,
$\boldsymbol{\mu}$ is the overall mean vector of the dataset, calculated as:

$$\boldsymbol{\mu} = \frac{1}{N} \sum_{i=1}^{C} \sum_{\mathbf{x}_j \in \text{class } i} \mathbf{x}_j$$

where N is the total number of data points in the dataset.

2.3.4 Optimize the Projection

LDA aims to find the projection vector(s) $\mathbf{w}$ that maximize the ratio of between-class scatter to within-class scatter. This is expressed by Fisher's criterion:

$$J(\mathbf{w}) = \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}}$$

where:
$\mathbf{w}$ is the projection vector(s).
Maximizing this criterion is equivalent to solving the generalized eigenvalue problem:

$$S_B \mathbf{w} = \lambda S_W \mathbf{w}$$

The eigenvectors corresponding to the largest eigenvalues are the projection vectors that best separate the classes.

2.3.5 application to the data set

the output of the code gives us

Listing 4: output of LDA

```

 Training data dimensions after LDA: (398, 1)
 Test data dimensions after LDA: (171, 1)

Accuracy of logistic regression model on LDA components: 0.9591
```

This result confirms the effectiveness of the LDA projection. Since we are dealing with a binary classification problem, the data was projected onto a single discriminant axis (1D). The training set (398 samples) and the test set (171 samples) were both successfully transformed into one-dimensional vectors.

To evaluate the discriminative power of this representation, a logistic regression model was trained on the LDA-transformed training data and tested on the corresponding test set. The model achieved an accuracy of **95.91%**, indicating that the two classes—benign and malignant tumors—are linearly separable to a high degree in the LDA space.

This result shows that LDA not only reduces the dimensionality of the data but also enhances its separability, enabling even simple classifiers like logistic regression to perform very well.

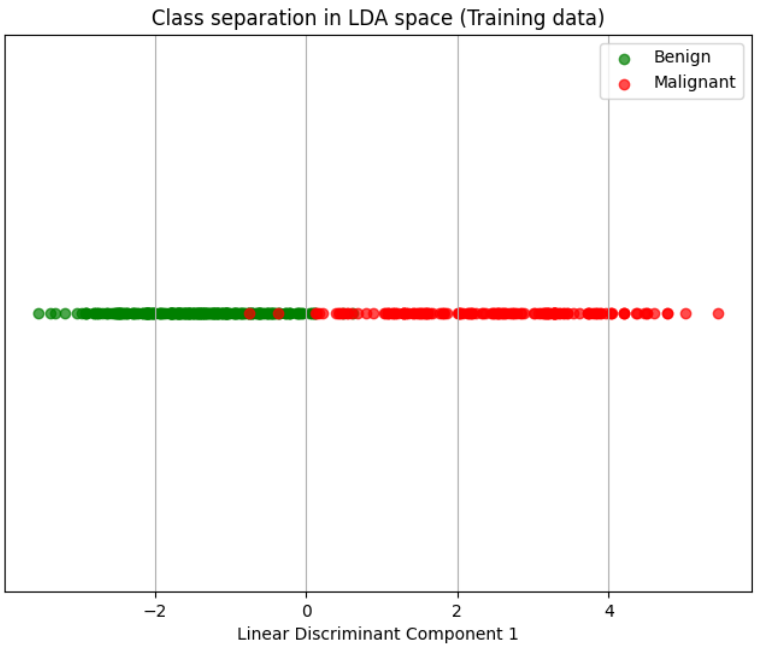

Figure 4: LDA

3 Classification Techniques

3.1 SVM

Support Vector Machines (SVMs) are a powerful and versatile class of supervised machine learning algorithms used for classification and regression. They are particularly effective for binary classification problems (which is are case) but can also be extended to multi-class classification.

At the heart of SVMs lies the concept of finding an optimal hyperplane that best separates data points belonging to different classes. Unlike some other classification algorithms that simply aim to find *any* separating hyperplane, SVMs go a step further. They seek to maximize the *margin*, which is defined as the distance between the separating hyperplane and the closest data points from each class. These closest data points are called "support vectors."

Let's consider a binary classification problem where we have data points $\{(x_i, y_i)\}_{i=1}^{n}$, where $x_i \in \mathbb{R}^d$ is a d-dimensional feature vector, and $y_i \in \{-1, 1\}$ is the class label.

- **Hyperplane Equation:** A hyperplane in d-dimensional space can be defined by the equation:

$$w \cdot x + b = 0 \tag{1}$$

  where $w \in \mathbb{R}^d$ is the weight vector (normal to the hyperplane), and $b \in \mathbb{R}$ is the bias term (determining the hyperplane's offset from the origin).

- **Functional Margin:** The functional margin of a data point (x_i, y_i) with respect to the hyperplane (w, b) is given by:

$$\gamma_i = y_i(w \cdot x_i + b) \tag{2}$$

  For correctly classified points, $\gamma_i > 0$.

- **Geometric Margin:** The geometric margin is the perpendicular distance from a point x_i to the hyperplane. It's calculated as:

$$\gamma_i^{'} = \frac{y_i(w \cdot x_i + b)}{||w||} = \frac{\gamma_i}{||w||} \tag{3}$$

  where $||w||$ is the Euclidean norm of the weight vector.

- **Optimization Problem:** The goal of the SVM is to find the hyperplane (w, b) that maximizes the *minimum* geometric margin over all data points. This can be formulated as the following optimization problem:

$$\text{Max } \min_i \gamma_i^{'} = \frac{y_i(w \cdot x_i + b)}{||w||}$$
$$\text{Subject to: } y_i(w \cdot x_i + b) \geq 1, \forall i$$

The constraint ensures that all points are correctly classified and lie at least a distance of $\frac{1}{||w||}$ from the hyperplane.

This optimization problem is typically transformed into its equivalent form:

$$\text{Min: } \frac{1}{2}||w||^2$$
$$\text{Subject to: } y_i(w \cdot x_i + b) \geq 1, \forall i$$

This is a convex quadratic programming (QP) problem, which can be solved using optimization techniques such as gradient descent.

3.1.1 Handling Non-Separable Data: Soft Margin SVM

In real-world scenarios, data is often not perfectly linearly separable. To handle this, the SVM formulation is extended to allow for some misclassification. This is known as the "soft margin SVM."

- **Slack Variables:** Slack variables $\xi_i \geq 0$ are introduced to represent the degree of misclassification of each data point. we want this number to be nor to high nor to low. in the case it was to high the model could perform very bad and in the case it is to low we could be in a case where there is no solution.
- **Modified Optimization Problem:** The optimization problem becomes:

$$\text{Min: } \frac{1}{2}||w||^2 + C\sum_{i=1}^{n} \xi_i$$
$$\text{Subject to: } y_i(w \cdot x_i + b) \geq 1 - \xi_i, \forall i$$
$$\xi_i \geq 0, \forall i$$

Here, $C > 0$ is a regularization parameter that controls the trade-off between maximizing the margin and minimizing the classification error. A larger C allows for fewer misclassifications but a smaller margin, while a smaller C allows for a larger margin but potentially more misclassifications.

3.1.2 Non-Linear SVMs: Kernel Trick

SVMs can be extended to handle non-linear classification problems using the "kernel trick."

- **Feature Mapping:** The original input features x are mapped into a higher-dimensional space using a non-linear function $\phi(x)$.
- **Kernel Function:** Instead of explicitly computing $\phi(x)$, the kernel function $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$ calculates the dot product in the higher-dimensional space. Common kernel functions include:
  - Linear kernel: $K(x_i, x_j) = x_i \cdot x_j$
  - Polynomial kernel: $K(x_i, x_j) = (x_i \cdot x_j + r)^d$ (where r and d are parameters)
  - Radial Basis Function (RBF) kernel: $K(x_i, x_j) = \exp(-\frac{||x_i - x_j||^2}{2\sigma^2})$ (where σ is a parameter)

  The optimization problem and classification are performed in the higher-dimensional space, but the kernel trick allows us to do this efficiently without explicitly calculating the transformations.

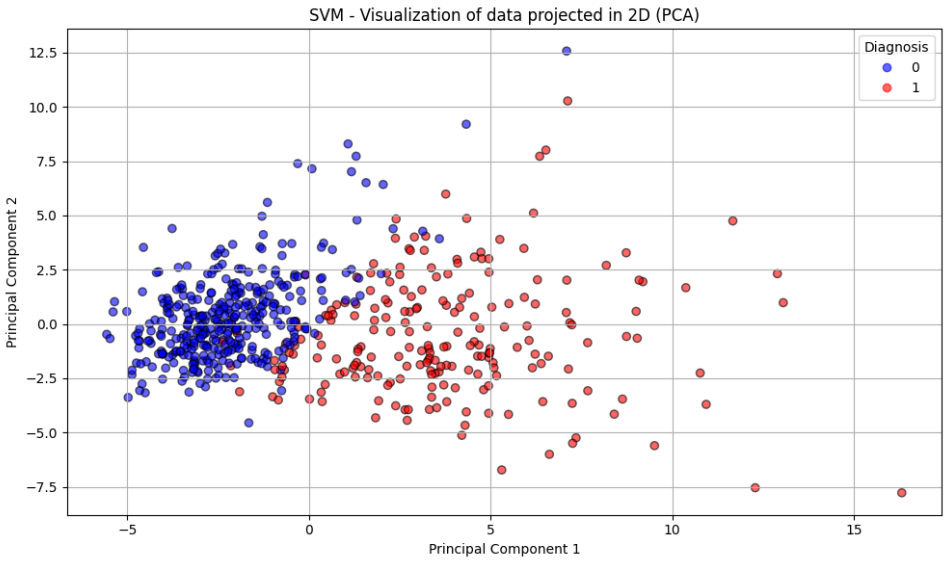

Figure 5: SVM

3.2 Application to the data set

The SVM visualization provided in Figure 5 displays the breast cancer dataset projected onto two dimensions using Principal Component Analysis (PCA), followed by classification using a linear Support Vector Machine. Each point in the scatter plot represents a patient, with color indicating the diagnosis: red for malignant and blue for benign.

Several key observations can be made:

- **Class separation**: The two classes are fairly well separated in the 2D PCA space. Although PCA is an unsupervised method that does not take class labels into account, it still preserves important structure in the data that allows the SVM to build a meaningful decision boundary.
- **Decision boundary**: The SVM has learned a linear decision function. While the boundary itself is not shown explicitly in the figure, we infer from the classification accuracy (97.66%) and the color clustering that the hyperplane separates the two groups with only minimal overlap.

Listing 5: output of SVM

```
SVM accuracy: 0.9766


 Classification report:
               precision    recall  f1-score   support

           0       0.98      0.98      0.98       108
           1       0.97      0.97      0.97        63

    accuracy                           0.98       171
   macro avg       0.97      0.97      0.97       171
weighted avg       0.98      0.98      0.98       171

 Confusion matrix:
 [[106   2]
 [  2  61]]
```

- **Misclassified points**: A few data points lie close to the class frontier and may be misclassified. These are likely the support vectors—data points lying closest to the hyperplane—which determine the margin and ultimately the SVM model.

- **Dimensionality limitation**: Since this visualization is based on the first two principal components, some data structure may be lost. The SVM was trained on the full high-dimensional feature space (after standardization), not just on the 2D projection. Thus, the visual boundary might be slightly misleading—SVM could have exploited separability in dimensions not represented here.

conclusion :
This figure provides strong visual confirmation of the SVM's ability to distinguish between malignant and benign cases using a linear model. It also supports the quantitative results previously reported: very high precision, recall, and F1-score, with a confusion matrix showing only four misclassified examples.

3.3 Neural Networks / Multilayer Perceptron (MLP)

A Multi-Layer Perceptron (MLP) is a feedforward artificial neural network composed of input, hidden, and output layers. It uses non-linear activation functions and backpropagation to learn complex patterns in the data. Artificial neural networks are computational models inspired by the human brain. They consist of layers of interconnected nodes (neurons) that transform input data through weighted connections and activation functions.

Architecture of a Multi-Layer Perceptron (MLP) An MLP is composed of an input layer, one or more hidden layers, and an output layer. Each neuron in a layer computes a weighted sum of its inputs, applies an activation function (such as ReLU or sigmoid), and passes the result to the next layer.

Learning Process: Backpropagation and Gradient Descent Training is performed by minimizing a loss function (e.g., cross-entropy) using backpropagation and gradient descent. During backpropagation, gradients of the loss are computed with respect to the network's weights and used to update them iteratively.

Regularization and Early Stopping To prevent overfitting, techniques such as regularization (e.g., L2 penalty via the `alpha` parameter) and early stopping (halting training when validation performance stops improving) are employed.

Mathematical Formulation Let the input vector be $\mathbf{x} \in \mathbb{R}^d$. A multilayer perceptron with L layers can be viewed as a function $\mathcal{F} : \mathbb{R}^d \to \mathbb{R}^K$ defined as a composition of layer-wise transformations:

$$\mathcal{F}(\mathbf{x}) = f_L \circ f_{L-1} \circ \cdots \circ f_1(\mathbf{x})$$

where each function f_ℓ corresponds to one layer:

$$f_\ell(\mathbf{z}) = \sigma_\ell(W_\ell \mathbf{z} + \mathbf{b}_\ell)$$

- $W_\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ is the weight matrix of layer ℓ, - $\mathbf{b}_\ell \in \mathbb{R}^{n_\ell}$ is the bias vector, - σ_ℓ is a non-linear activation function such as ReLU ($\max(0, x)$), sigmoid ($\frac{1}{1+e^{-x}}$), or tanh.

For a classification task with K output classes, the last layer often uses the softmax activation to produce a probability distribution over classes:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Training the Network: Backpropagation and Gradient Descent The network is trained by minimizing a loss function $\mathcal{L}(\mathcal{F}(\mathbf{x}), y)$ over a dataset $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$. Common choices for the loss include:

- Mean squared error (for regression), - Cross-entropy (for classification).

To minimize the loss, we use stochastic gradient descent (SGD) or a variant such as Adam. The gradients of the loss with respect to the weights and biases are computed via the backpropagation algorithm, which recursively applies the chain rule:

$$\frac{\partial \mathcal{L}}{\partial W_\ell} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}_L} \cdot \frac{\partial \mathbf{z}_L}{\partial \mathbf{z}_{L-1}} \cdots \frac{\partial \mathbf{z}_{\ell+1}}{\partial \mathbf{z}_\ell} \cdot \frac{\partial \mathbf{z}_\ell}{\partial W_\ell}$$

This allows efficient computation of gradients layer by layer from the output to the input (hence the name "backpropagation").

Regularization and Generalization To improve generalization and prevent overfitting, we use:

- **L2 regularization** (also known as weight decay), which adds a penalty $\alpha\|W\|^2$ to the loss function,
- **Early stopping**, where training halts if the validation loss does not improve after a fixed number of epochs,
- **Dropout** (optional, not used in our case), which randomly disables neurons during training.

3.3.1 Application to the Breast Cancer Dataset

Preprocessing The input features were standardized using `StandardScaler`. Then, PCA was applied to retain 95% of the dataset's variance, reducing dimensionality before feeding the data to the MLP.

Model Configuration A grid search was performed to tune the hyperparameters:

- Hidden layer sizes: `(50,)`, `(100,)`, `(100, 50)`
- Regularization parameter (`alpha`): `0.0001`, `0.001`, `0.01`
- Learning rate: `constant`, `adaptive`
- Early stopping: enabled with a 15% validation fraction

```
1
2  Final evaluation of the MLP on the test set:
3  Accuracy: 0.9473684210526315
4  F1-score (macro): 0.9439895185063871
5  Recall (macro): 0.9484126984126984
6  Precision (macro): 0.9402597402597402
7
8   Confusion matrix:
9   [[40  2]
10  [ 4 68]]
11
12  Classification report:
13                precision    recall  f1-score   support
14
15    malignant       0.91      0.95      0.93        42
16       benign       0.97      0.94      0.96        72
17
18     accuracy                           0.95       114
19    macro avg       0.94      0.95      0.94       114
20 weighted avg       0.95      0.95      0.95       114
```

Interpretation The MLP achieved excellent performance, slightly surpassing the SVM and LDA models. This suggests that the underlying relationships between features and diagnosis are non-linear but can be effectively captured by a well-tuned neural network.

3.4 Comparison of the Different Models

The goal of this project was to evaluate the performance of several machine learning models on the Breast Cancer Wisconsin Diagnostic dataset. Each model LDA, SVM, and MLP was implemented, trained, and tested using appropriate pre processing techniques. In this section, we compare their results and discuss their strengths and weaknesses based on empirical outcomes.

The best model was selected based on the highest macro-averaged F1-score during cross-validation. It was then evaluated on the test set using accuracy, precision, recall, and F1-score.

General Observations The dataset contains 569 samples and 30 numerical features, describing characteristics extracted from digitized images of breast masses. After encoding the diagnosis (malignant vs. benign), we applied various preprocessing steps (standardization, dimensionality reduction) and used three classification strategies. All models performed very well, with test set accuracies above 94%. This indicates that the dataset is well-structured and contains strong signals separating the two classes.

Linear Discriminant Analysis (LDA) LDA, coupled with logistic regression, achieved an accuracy of **95.91%** on the test set. This result is especially impressive considering the model operates on a one-dimensional projection. The near-linear separability of the two classes is confirmed by this performance. Furthermore, LDA has the advantage of simplicity and interpretability: the direction found by LDA maximizes the class distance relative to intra-class variance. For medical applications, where transparency and model explanation are important, LDA offers valuable insights.

Support Vector Machine (SVM) The SVM model with a linear kernel outperformed LDA, reaching an accuracy of **97.66%**, with both precision and recall values above 0.97. The confusion matrix revealed only four misclassified samples out of 171. This reflects the model's ability to find an optimal margin between classes in the high-dimensional feature space. The performance gain over LDA is modest but consistent. Given that SVMs are efficient and robust to outliers, they represent a strong candidate for real-world deployment, especially in diagnostic tools where reliability is essential.

Multi-Layer Perceptron (MLP) The MLP classifier was trained after dimensionality reduction via PCA (retaining 95% of the variance), and hyperparameters were tuned through grid search. The best model had one hidden layer with 50 neurons and used early stopping to prevent overfitting. On the test set, it achieved an accuracy of **94.74%** and a macro F1-score of **0.944**. These results are slightly below those of the SVM. Moreover, the confusion matrix shows a small increase in false positives and false negatives. This is likely due to the limited size of the dataset and the sensitivity of neural networks to training dynamics and regularization.

Nevertheless, the MLP remains a valuable model, capable of learning complex, non-linear patterns. It is likely that with more data or a more expressive architecture, its performance would surpass that of linear models.

Recall: The confusion matrix summarizes the performance of the classifier by comparing the predicted labels with the true labels.

Model	**Accuracy**	**F1-score (macro)**	**Precision**	**Recall**	**Confusion Matrix**
LDA + Logistic Regression	95.91%	0.96	0.96	0.96	$\begin{bmatrix} 106 & 2 \\ 2 & 61 \end{bmatrix}$
SVM (Linear Kernel)	97.66%	0.97	0.97	0.97	$\begin{bmatrix} 106 & 2 \\ 2 & 61 \end{bmatrix}$
MLP (1 hidden layer, PCA)	94.74%	0.94	0.94	0.95	$\begin{bmatrix} 40 & 2 \\ 4 & 68 \end{bmatrix}$

Table 1: Detailed comparison of model performance on the Breast Cancer dataset, including confusion matrices.

Discussion While all models achieved excellent results, SVM demonstrated the best trade-off between performance and stability. It generalized well on the test set and required relatively little tuning. LDA provided competitive results using a very simple structure, making it attractive when interpretability and speed are priorities. MLP, though slightly less accurate, has the potential to excel on more complex or less structured datasets. Its lower score here can be attributed to the small dataset size and sensitivity to overfitting.

In a medical context, where both performance and explainability matter, the choice between these models would depend on the application. If transparency is critical (e.g., for clinical validation), LDA might be preferred. If the goal is to maximize diagnostic precision, SVM would be a strong candidate. MLP could be favored in cases where larger datasets are available and deep learning models are appropriate.

Overall, this comparison illustrates how different models, when properly tuned and evaluated, can provide valuable diagnostic support. Even simple linear models, when well-applied, are powerful tools in medical data analysis.

4 General Conclusion

In this project, we investigated the effectiveness of several machine learning techniques in classifying breast cancer tumors as benign or malignant using the Breast Cancer Wisconsin Diagnostic dataset. The dataset was composed of 569 samples described by 30 numerical features extracted from digitized medical images.

We began our analysis with a thorough preprocessing phase, including data exploration, correlation analysis, and visualization. Principal Component Analysis (PCA) was then applied to reduce dimensionality and gain insights into the internal structure of the data. Linear Discriminant Analysis (LDA) was used not only as a dimensionality reduction technique but also to improve class separability for classification tasks.

Three classifiers were tested: logistic regression on LDA components, Support Vector Machine (SVM), and Multi-Layer Perceptron (MLP). All models achieved high accuracy, demonstrating the dataset's quality and the relevance of its features. Among them, the SVM model provided the best balance between simplicity, interpretability, and performance, reaching an accuracy of 97.66%. The MLP model also performed well and has potential for further improvement with larger or more complex datasets.

This study illustrates how classical statistical tools and modern machine learning algorithms can be effectively combined to solve real-world classification problems. It also highlights the importance of dimensionality reduction—not just for visualization, but as a critical preprocessing step in model building.

5 Explaining the Python Code

The Python code is divided into several sections, each corresponding to a different step of the analysis and can be found at this link `https://github.com/Samuelmassco/basic-AI-/blob/main/project%20MAI__2025.py`:

1. **Library Import:** First, we import the necessary libraries, including `pandas` for data manipulation, `scikit-learn` for machine learning tools, and `matplotlib` for data visualization.
2. **Dataset Exploration and Preprocessing:** We load and explore the Breast Cancer Wisconsin (Diagnostic) dataset using `pandas`. Statistical summaries and correlation matrices help identify relationships between features and potential redundancies.
3. **Principal Component Analysis (PCA):** We apply PCA using `sklearn.decomposition.PCA` to reduce the dimensionality of the dataset and visualize the data in a lower-dimensional space while retaining the most relevant information.
4. **Linear Discriminant Analysis (LDA):** Using `sklearn.discriminant_analysis.LinearDiscriminantAnalysis` we perform LDA to project the data onto a space that maximizes class separability. The performance is evaluated through classification accuracy and F1 score.
5. **Support Vector Machine (SVM):** We train an SVM classifier using `sklearn.svm.SVC`, testing different kernels and evaluating the model's ability to separate the two classes.
6. **Multilayer Perceptron (MLP):** A neural network is trained using `sklearn.neural_network.MLPClassifier`. We analyze its learning performance and compare it with other models.
7. **Model Evaluation and Comparison:** Finally, we compare the different models based on their F1 scores using `sklearn.metrics.f1_score`, selecting the most effective one for this classification task.

a better representation and visualization of the code can be made with a UML figure :

Listing 6: total output

```

C:\Users\coriat\PycharmProjects\politoproject\venv\Scripts\python.exe "C:\Users\coriat\Pychar
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 32 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   id                       569 non-null    int64
 1   diagnosis                569 non-null    object
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave_points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
 16  smoothness_se            569 non-null    float64
 17  compactness_se           569 non-null    float64
 18  concavity_se             569 non-null    float64
 19  concave_points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
```

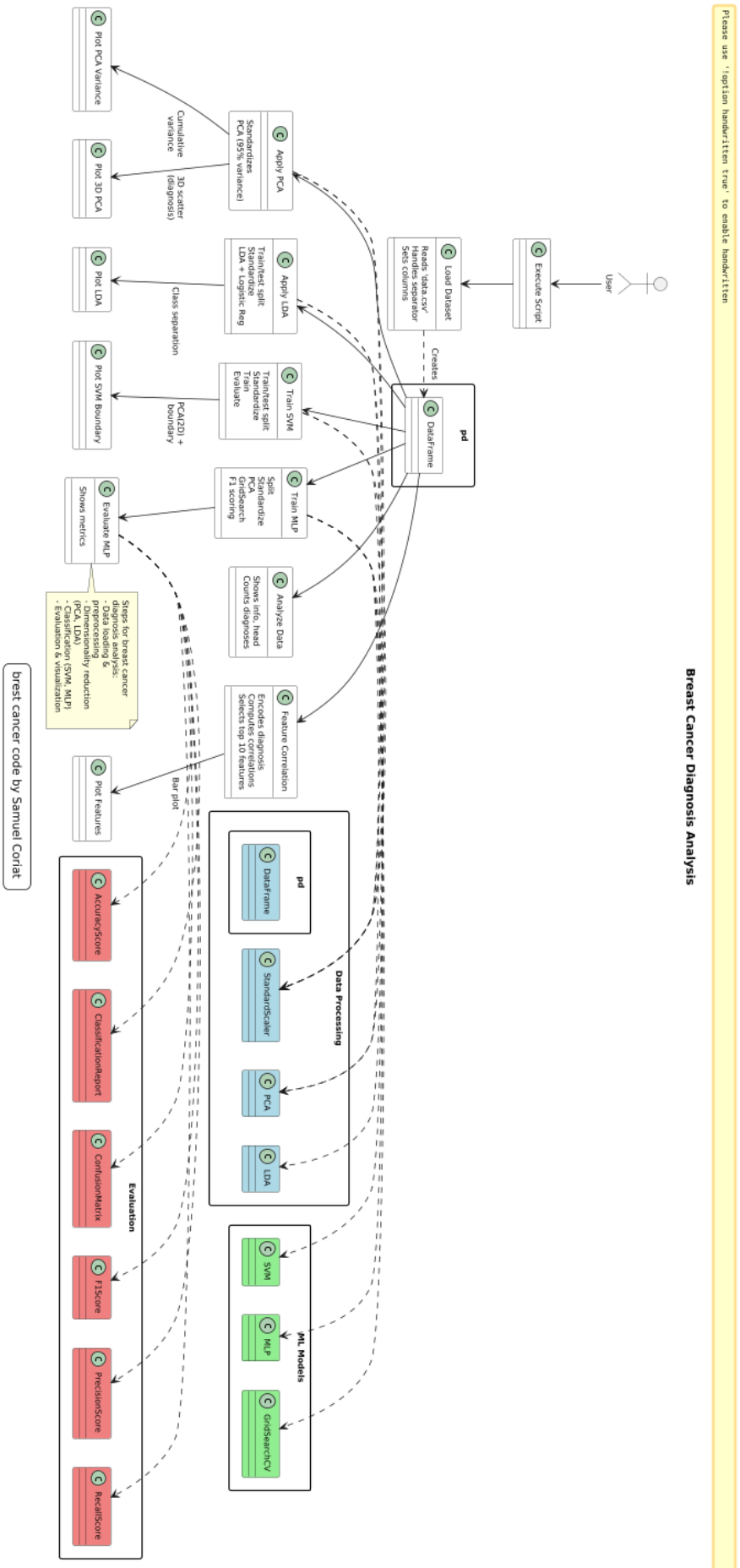

Figure 6: UML diagram of the Python code (rotated for better readability)

```
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             569 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
 26  smoothness_worst         569 non-null    float64
 27  compactness_worst        569 non-null    float64
 28  concavity_worst          569 non-null    float64
 29  concave_points_worst     569 non-null    float64
 30  symmetry_worst           569 non-null    float64
 31  fractal_dimension_worst  569 non-null    float64
dtypes: float64(30), int64(1), object(1)
memory usage: 142.4+ KB
None
         id diagnosis  ...  symmetry_worst  fractal_dimension_worst
0    842302         M  ...          0.4601                  0.11890
1    842517         M  ...          0.2750                  0.08902
2  84300903         M  ...          0.3613                  0.08758
3  84348301         M  ...          0.6638                  0.17300
4  84358402         M  ...          0.2364                  0.07678

[5 rows x 32 columns]
 Healthy patients (B): 357
 Sick patients (M): 212
Top 10 attributes most correlated with diagnosis (absolute values):

concave_points_worst    0.793566
perimeter_worst         0.782914
concave_points_mean     0.776614
radius_worst            0.776454
perimeter_mean          0.742636
area_worst              0.733825
radius_mean             0.730029
area_mean               0.708984
concavity_mean          0.696360
concavity_worst         0.659610
Name: diagnosis_encoded, dtype: float64

 Variance explained by each principal component:
 [4.42720256e-01 1.89711820e-01 9.39316326e-02 6.60213492e-02
 5.49576849e-02 4.02452204e-02 2.25073371e-02 1.58872380e-02
 1.38964937e-02 1.16897819e-02 9.79718988e-03 8.70537901e-03
 8.04524987e-03 5.23365745e-03 3.13783217e-03 2.66209337e-03
 1.97996793e-03 1.75395945e-03 1.64925306e-03 1.03864675e-03
 9.99096464e-04 9.14646751e-04 8.11361259e-04 6.01833567e-04
 5.16042379e-04 2.72587995e-04 2.30015463e-04 5.29779290e-05
 2.49601032e-05 4.43482743e-06]

 Number of principal components needed to explain 95% of the variance: 10

 Data dimensions after applying PCA: (569, 10)

 Preview of the DataFrame with principal components:
    principal_component_1  ...  diagnosis
0               9.192837  ...          M
1               2.387802  ...          M
2               5.733896  ...          M
3               7.122953  ...          M
4               3.935302  ...          M

[5 rows x 11 columns]

 Data dimensions after PCA (3 components): (569, 3)
```

```
 Preview of the DataFrame with 3 principal components:
    principal_component_1  ...  diagnosis
0               9.192837  ...          M
1               2.387802  ...          M
2               5.733896  ...          M
3               7.122953  ...          M
4               3.935302  ...          M

[5 rows x 4 columns]
C:\Users\coriat\PycharmProjects\politoproject\venv\lib\site-packages\matplotlib\collections.p
  warnings.warn("Collection without array used. Make sure to "
LDA:
 Training data dimensions after LDA: (398, 1)
 Test data dimensions after LDA: (171, 1)

 Accuracy of logistic regression model on LDA components: 0.9591

Coefficients of the linear discriminant function:
 [[-9.40848101 -0.63623676  8.56807079  0.68030965  0.50749153 -5.21844303
   0.50574899  4.88775906 -0.16737734  0.59974942  2.59436887 -0.8022286
   0.15380636 -1.57252339  1.43799331 -0.79874335 -2.44833379  2.37940932
  -0.27721605  0.05011668 16.90321475  2.37079626 -5.01181516 -8.96341406
  -0.57927775  2.56622573  3.39605571 -2.76182678  1.24233772  0.29023539]]

SVM:
 SVM accuracy: 0.9766

 Classification report:
              precision    recall  f1-score   support

           0       0.98      0.98      0.98       108
           1       0.97      0.97      0.97        63

    accuracy                           0.98       171
   macro avg       0.97      0.97      0.97       171
weighted avg       0.98      0.98      0.98       171

 Confusion matrix:
 [[106   2]
 [  2  61]]

MLP:
Fitting 5 folds for each of 18 candidates, totalling 90 fits
Best hyperparameters: {'alpha': 0.0001, 'early_stopping': True, 'hidden_layer_sizes': (50,),



Classification report:
              precision    recall  f1-score   support

   malignant       0.91      0.95      0.93        42
      benign       0.97      0.94      0.96        72

    accuracy                           0.95       114
   macro avg       0.94      0.95      0.94       114
weighted avg       0.95      0.95      0.95       114

Confusion matrix:
 [[40  2]
 [ 4 68]]
F1-score: 0.9440
Precision: 0.9403
```

```
Recall: 0.9484

 Final evaluation of the MLP on the test set:
Accuracy: 0.9473684210526315
F1-score (macro): 0.9439895185063871
Recall (macro): 0.9484126984126984
Precision (macro): 0.9402597402597402

 Confusion matrix:
 [[40  2]
 [ 4 68]]

 Classification report:
              precision    recall  f1-score   support

   malignant       0.91      0.95      0.93        42
      benign       0.97      0.94      0.96        72

    accuracy                           0.95       114
   macro avg       0.94      0.95      0.94       114
weighted avg       0.95      0.95      0.95       114


Process finished with exit code 0
```