Dauphine UNIVERSITÉ PARIS | PSL

Predictive Classification of Breast Cancer Diagnosis

Quyen Linh TA [1] and Ha Anh TRAN [1]

[1] MIDO Department, University Paris Dauphine, PSL, Pl. du Maréchal de Lattre de Tassigny, 75016 Paris

Abstract

In this project, we implemented Logistic Regression and Linear Discriminant Analysis (LDA) algorithms to classify breast cancer cases as benign or malignant. The dataset used for this study consisted of 569 samples and 30 features, which were collected from fine needle aspirates of breast mass. The performance of the two algorithms was evaluated using various metrics such as accuracy, precision, recall, and F1-score. Our results show that LDA and Logistic Regression have very good predictive results with overall accuracy of 98.12% and 98.80%. Furthermore, LDA achieved a higher precision, recall and F1-score for both benign and malignant cases. This study demonstrates the effectiveness of various machine learning algorithms in classifying breast cancer cases and highlights the potential of these algorithms for use in clinical decision-making. In addition to Logistic Regression and LDA, this project also implemented several other machine learning algorithms such as Support Vector Machine (SVM), XGBoost, AdaBoost, and CatBoost to compare the results.

Keywords: Machine Learning — Classification — Logistic Regression — Linear Discriminant Analysis

Overview of algorithms

Logistic Regression

Logistic regression is a supervised learning algorithm that is used for classification tasks. The goal of logistic regression is to model the probability of a certain class or event occurring given a set of input features.

Given a set of input features X and a binary output variable Y, logistic regression models the probability of Y being 1 (or "positive") given X as:

$$P(Y=1|X) = \frac{1}{1+e^{-\beta^T X}}$$

where β is a vector of parameters that need to be learned from the data. This function is known as the sigmoid function, and it maps the input features to a value between 0 and 1, which can be interpreted as the probability of the positive class.

In order to learn the best values for the parameters β, logistic regression uses a method called gradient descent. Gradient descent is an optimization algorithm that iteratively updates the parameters in the direction of the negative gradient of the cost function.

The cost function that is used in logistic regression is the cross-entropy loss function, which measures the difference between the predicted probability and the true label.

$$J(\beta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}log(h_\beta(x^{(i)})) + (1-y^{(i)})log(1-h_\beta(x^{(i)}))]$$

- m is the number of training examples
- $h_\beta(x^{(i)})$ is the predicted probability for the i-th example
- $y^{(i)}$ is the true label for the i-th example

The gradient of the cost function with respect to the parameters is given by:

$$\frac{\partial J(\beta)}{\partial \beta} = \frac{1}{m}\sum_{i=1}^{m}(h_\beta(x^{(i)}) - y^{(i)})x^{(i)}$$

The parameters are updated according to the following rule:

$$\beta = \beta - \alpha\frac{\partial J(\beta)}{\partial \beta}$$

where α is the learning rate, which controls the step size of the update. Gradient descent converge when the cost function (or the loss function) reaches a minimum value. This means that the algorithm has found the set of parameters that minimize the difference between the predicted output and the true labels.

In practice, the algorithm stops when the cost function (or the loss function) stops decreasing or reaches a threshold value. This threshold value is called the stopping criteria, it could be set as a small value such as 10^{-3}, or as a number of iteration reached.

Algorithm 1 Gradient Descent for Logistic Regression

```
1: Input: training set $(x^{(1)}, y^{(1)}), ..., (x^{(m)}, y^{(m)})$, learning rate $\alpha$, number of iterations $numIter$
2: Output: $\beta$
3: Initialize $\beta$ to a random value
4: for $i$ in 1 to $numIter$ do
5:     Compute the predicted probability for all examples: $h_\beta(x^{(i)}) = \frac{1}{1+e^{-\beta^T X}}$
6:     Compute the gradient: $\frac{\partial J(\beta)}{\partial \beta} = \frac{1}{m}\sum_{i=1}^{m}(h_\beta(x^{(i)}) - y^{(i)})x^{(i)}$
7:     Update the parameters: $\beta = \beta - \alpha * \frac{\partial J(\beta)}{\partial \beta}$
8: return $\beta$
```

Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is a supervised machine learning technique used for classification problems. The goal of LDA is to find a linear combination of features that separates different classes with the greatest possible margin.

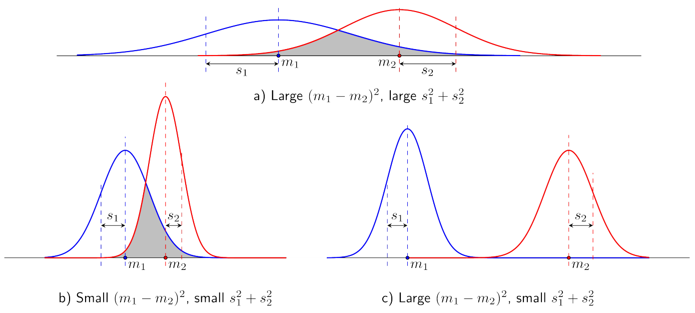

Figure 1 The distance between the expectations and the sum of the variances affects the discriminant degree of the data. a) The gap between the two expectations is large but the variance within each class is also large, causing the two distributions to overlap (gray part). b) The variance for each class is very small but the two expectations are too close, making it difficult to distinguish the two classes. c) When the variance is small enough and the gap between the two expectations is large enough, we find that the data are discriminant.

In LDA, the decision boundary is chosen to be the line that maximizes the ratio of the between-class variance to the within-class variance. Mathematically, this can be represented as:

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

where w is the projection vector, S_B is the between-class scatter matrix, and S_W is the within-class scatter matrix. The projection vector w is chosen to be the eigenvector of $S_W^{-1} S_B$ that corresponds to the largest eigenvalue.

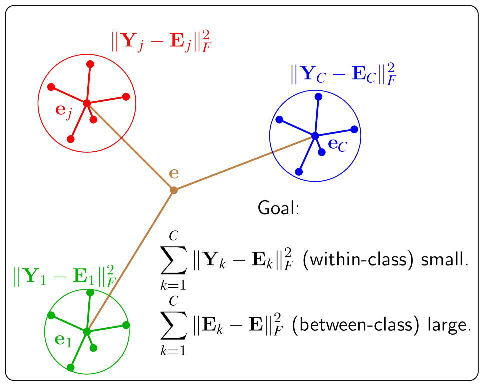
$\|\mathbf{Y}_j - \mathbf{E}_j\|_F^2$
$\mathbf{e}_j$
$\|\mathbf{Y}_C - \mathbf{E}_C\|_F^2$
$\mathbf{e}_C$
$\mathbf{e}$
$\|\mathbf{Y}_1 - \mathbf{E}_1\|_F^2$
$\mathbf{e}_1$
Goal:
$\sum_{k=1}^{C} \|\mathbf{Y}_k - \mathbf{E}_k\|_F^2$ (within-class) small.
$\sum_{k=1}^{C} \|\mathbf{E}_k - \mathbf{E}\|_F^2$ (between-class) large.

Figure 2 The purpose is also that the difference between the components in a class (within-class) is small and the difference between classes is large. Different colored data points represent different classes.

The between-class scatter matrix is defined as:

$$S_B = \sum_{i=1}^{c} N_i (\mu_i - \mu)(\mu_i - \mu)^T$$

where c is the number of classes, N_i is the number of samples in class i, μ_i is the mean vector of class i, and μ is the overall mean vector.

The within-class scatter matrix also known as the within-class variance matrix is defined as:

$$S_W = \sum_{i=1}^{c} \sum_{x \in D_i} (x - \mu_i)(x - \mu_i)^T$$

where D_i is the set of samples in class i.
Once the projection vector w is found, new samples can be projected onto the line defined by w and then classified based on which side of the line they fall on. LDA can be thought of as a technique that finds the most discriminative direction in the feature space.

Description of dataset

The Breast Cancer Wisconsin Diagnosis dataset Street *et al.* (1993) is a collection of clinical and demographic information about breast cancer patients. It was created by Dr. William H. Wolberg, a physician and researcher at the University of Wisconsin, and has been used in machine learning research for breast cancer diagnosis and prognosis.

The dataset contains a total of 569 samples, each of which represents a patient with a malignant or benign tumor. The samples are described by 30 real-valued features which are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They represent different characteristics of the cell nuclei present in the image. Some of the features include the radius, texture, perimeter, area, smoothness, concavity, and concave points of the nuclei.Details of definitions for each concept and variable names will be mentioned in the project's GitHub repository.

The dataset is available for download from the `UCI ML Repository`. It is an important dataset in the field of machine learning, as it is widely used in breast cancer research and has contributed to the development of many machine learning models for breast cancer diagnosis.

Data preprocessing and statistical analysis

Check normality and Gaussian transformation

Many statistical methods and machine learning models assume that the data is normally distributed, also known as Gaussian or bell-shaped distribution. The normal distribution is a probability distribution that is symmetric around the mean, with most of the data points concentrated around the mean and fewer data points farther away from the mean. Aliferis *et al.* (2002)

Assuming that the data is normally distributed has several advantages:

- **Simplicity**: Many statistical methods and machine learning models are based on the normal distribution, and they are relatively simple to use and interpret.
- **Robustness**: Some methods and models are robust to deviations from normality, and they can still produce accurate results even if the data is not perfectly normal.
- **Normality tests**: There are several statistical tests that can be used to check if the data is approximately normal. These tests allow us to determine whether the assumptions of normality are met, and they can help us to decide whether to use a parametric or a non-parametric method.

However, in real-world applications, the data is not always normal. Non-normal data can cause problems such as underestimating or overestimating the uncertainty in the estimates, leading to incorrect conclusions. In such cases, data transformations like log, square root, and cube root can be applied to make the data distribution more normal.

In this instance, tests show that several dataset attributes can be transformed to asymptotically a normal distribution using linear transformations such logarithms, square roots, or cube roots. Since a single value of a feature may or may not be met by linear transformations, the tests are used to check skewness or to test the probability distributions from which we get the treatment orientations for each feature.

Skewness is a measure of symmetry, or more precisely, the lack of symmetry. A distribution, or data set, is symmetric if it looks the same to the left and right of the center point. Kurtosis is a measure of whether the data are heavy-tailed or light-tailed relative to a normal distribution. Murphy (2013)

$$\tilde{\mu}_3 = \mathrm{E}\left[\left(\frac{X-\mu}{\sigma}\right)^3\right] = \frac{\mu_3}{\sigma^3} = \frac{\mathrm{E}\left[(X-\mu)^3\right]}{\left(\mathrm{E}\left[(X-\mu)^2\right]\right)^{3/2}} = \frac{\kappa_3}{\kappa_2^{3/2}}$$

It should be noted that the data we use is mostly moderately skew to the right, and not all features can fit linear transformations such as logarithms, square roots two or three.

Correlated features analysis

Low correlated features and high correlated features can affect a learning model in different ways:

Low correlated features: Features with low correlation between them provide more information and increase the model's ability to capture the underlying patterns in the data. Having a set of low correlated features helps the model to generalize well, leading to better performance on unseen data. Additionally, low correlated features can decrease the chances of multicollinearity and overfitting.

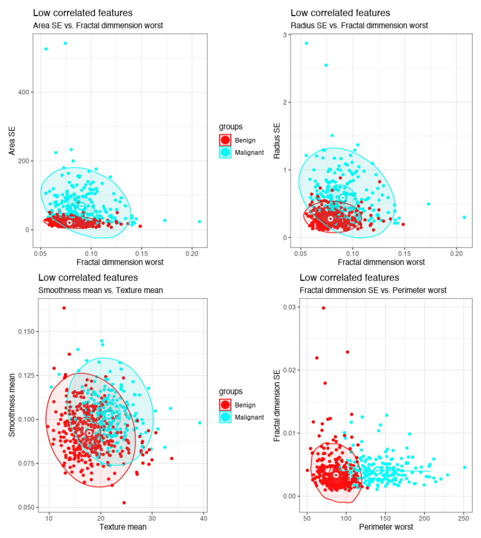

Figure 3 Some low correlated features in dataset

High correlated features: Features with high correlation between them can provide redundant information and decrease the model's ability to capture the underlying patterns in the data. High correlation between features also increases the chances of multicollinearity, which can cause unstable and unreliable estimates of the model parameters. In addition, high correlated features can lead to overfitting, making the model perform well on the training data but poorly on unseen data.

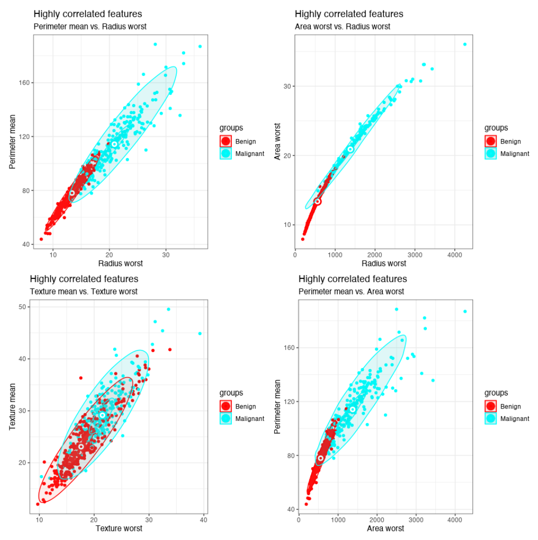

Figure 4 Some high correlated features in dataset

Therefore, it is important to identify and handle correlated features before training a model. One way to handle correlated features is by removing one of the correlated features, or by creating a new feature that is a combination of the correlated features. Additionally, techniques such as Principal Component Analysis (PCA) can be used to identify and remove correlated features.

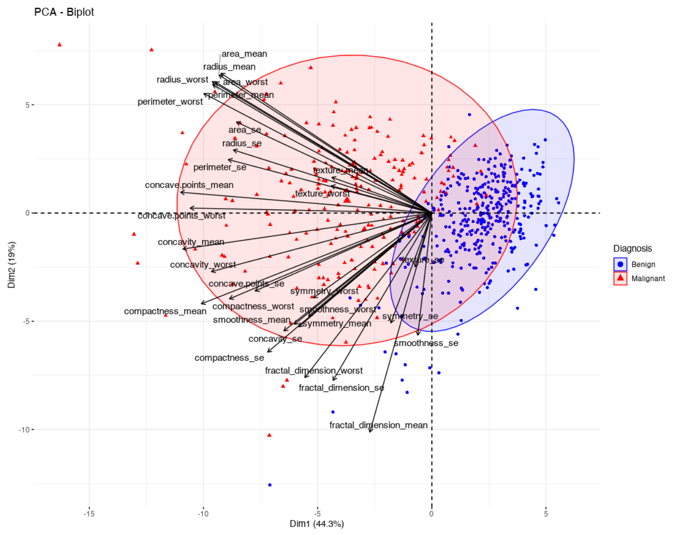

Figure 5 Plot of the principal components (PCs) of a PCA on the x- and y-axes, and it also shows the projections of the original variables (or "features") onto these PCs.

The advantage of a biplot is that it allows us to visualize the relationship between the original variables and the principal components in a two-dimensional space. This can help us to understand how the variables are related to each other, and how they contribute to the variation in the data.

Outliers detection

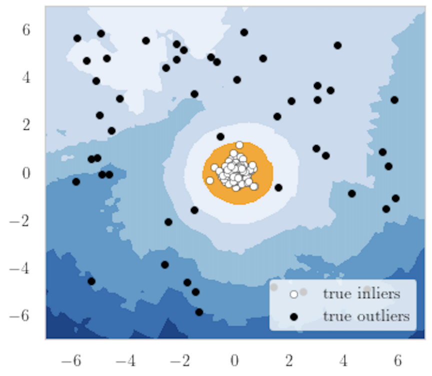

Figure 6 Local Outlier Factor (LOF) algorithm fitted on 2D

The Local Outlier Factor (LOF) algorithm is an unsupervised anomaly detection method that identifies outliers in a dataset by computing the local density deviation of a given data point with respect to its neighbors. The idea behind the LOF algorithm is that an outlier is a point that has a substantially lower density than its neighbors.

The LOF algorithm works by first defining a neighborhood around each data point, and then computing a local density for each point based on the number of points in its neighborhood. The local density of a point is defined as the reciprocal of the average distance to the k-nearest neighbors of the point. Points with a lower local density than their neighbors are considered to be outliers.

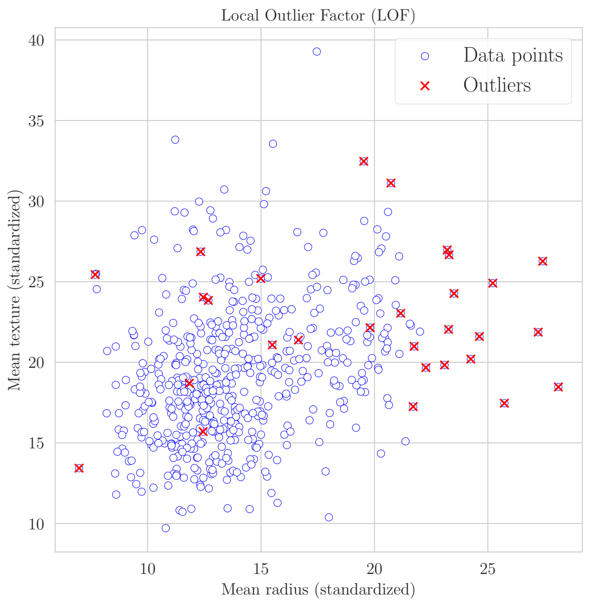

Figure 7 Local Outlier Factor algorithm result of our dataset with `threshold = 1.5`

The LOF algorithm uses two main parameters: the number of neighbors (k) and a threshold. The number of neighbors (k) is used to define the neighborhood around each data point. The threshold is used to convert the raw outlier scores to binary labels.

The advantages of using the LOF algorithm are: It is a density-based method, which means it can detect outliers in any shape of data distribution and also can detect outliers in high-dimensional data. It is sensitive to the local density of the data, which makes it more robust than methods that are based on global density. Finally, it can detect multiple types of outliers such as global, contextual and collective outliers.

Standardization and normalization

Standardization and normalization are techniques used to pre-process data before fitting a model. The goal is to make the data conform to certain assumptions made by the model.

- **Standardization**, also known as scaling or z-score normalization, scales the data so that it has a mean of 0 and a standard deviation of 1. This is useful for models that assume that the data is normally distributed.
- **Normalization**, scales the data to have a minimum value of 0 and a maximum value of 1. This is useful for models that assume that the data is bounded.

Both standardization and normalization help to speed up the convergence of the model during training, and can lead to better performance by making sure that the model is not trying to learn from data that is not in the appropriate scale or distribution.

Evaluating models

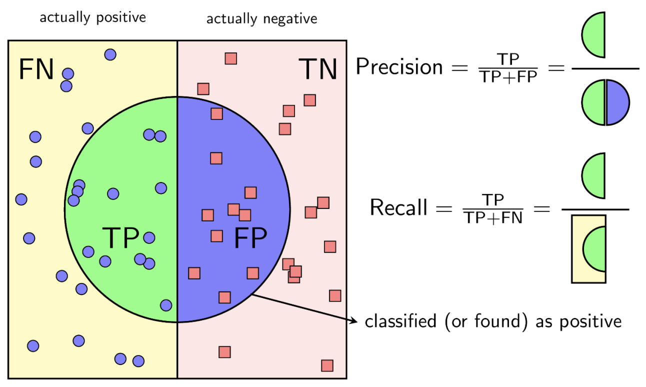

Figure 8 How to calculate Precision and Recall.

There are several metrics commonly used to evaluate classification models, including:

- **Accuracy**: This is the ratio of correctly predicted observations to the total observations. It is a commonly used metric but can be misleading when the classes are imbalanced.
- **Precision**: This is the ratio of correctly predicted positive observations to the total predicted positive observations. It is a measure of the model's ability to correctly identify positive instances.

- **Recall (Sensitivity or Hit Rate)**: This is the ratio of correctly predicted positive observations to the total actual positive observations. It is a measure of the model's ability to identify all the positive instances.
- **F1-Score**: It is the harmonic mean of precision and recall. It is a good metric to use when the classes are imbalanced.
- **AUC-ROC Curve**: The Receiver Operating Characteristic (ROC) curve is a plot of the true positive rate (sensitivity) against the false positive rate (1-specificity) for different thresholds. The Area Under the ROC Curve (AUC) represents the ability of the model to distinguish between positive and negative classes.

The learning curve is a visualization of the model's performance on the training and validation sets as the size of the training set increases. It is a useful tool for identifying overfitting and underfitting.

Hyperparameters tuning

Maximum cross-validation score is used as the evaluation metric because it provides a robust estimate of the model's performance on unseen data. Cross-validation splits the data into multiple folds, trains the model on different subsets, and evaluates the performance on the remaining data. This ensures that the model's performance is not overly influenced by the choice of a single training/validation split.

We mostly use Grid Search strategy for our models, Grid Search is an exhaustive search algorithm that trains and evaluates a model for each combination of hyperparameters in a predefined grid. It's a simple and straightforward approach that requires defining the hyperparameters and their possible values beforehand. The algorithm will then evaluate the model performance for each combination of hyperparameters, and the best combination of hyperparameters can be selected based on a performance metric, such as maximum cross-validation score.

Experiments

Below are some experimental results that we have tested with several different algorithms besides Linear Regression and LDA, each model with selected and optimized parameters and hyperparameters.

Some algorithms that we further tested like XGBoost, Linear SVM, Ridge Regression. The hyperparameters are selected by us according to the GridSearch and RandomSearch optimization and search algorithms, the reliability is after we recognize the hypotheses and check the overfitting.

	Accuracy	Precision	Recall	F1-Score
Logistic	0.98	0.97	0.97	0.97
LDA	0.97	1.0	0.91	0.95
NeuralNet	0.99	0.97	1.0	0.98
LinearSVM	0.98	0.97	0.97	0.97
Ridge	0.95	1.0	0.86	0.92
XGBoost	0.96	0.94	0.94	0.94

Figure 9 Score of classification models testing metrics

Overall, the NeuralNet model performed the best among the six models with an accuracy of 0.99 and F1-score of 0.98. The Logistic, LinearSVM, and XGBoost models also performed well with high accuracy and F1-scores around 0.97. The Ridge model had the lowest accuracy of 0.95 and F1-score of 0.92.

It is worth noting that the LDA model had a precision of 1.0, but a lower recall of 0.91. This may indicate that the model is good at correctly identifying positive examples, but not as good at identifying all positive examples.

In this case, the LDA model has a precision of 1.0, which means that all positive examples that it classified as positive are indeed positive. However, its recall of 0.91 indicates that it is not very good at identifying all positive examples, as there are some positive examples that the model failed to identify.

This trade-off between precision and recall is often encountered in machine learning and can have practical implications, depending on the specific use case. Because, in medical diagnosis, it is often better to have a model with high recall, even if it results in some false positive diagnoses, as it is better to err on the side of caution and provide additional testing, rather than miss a potential positive case.

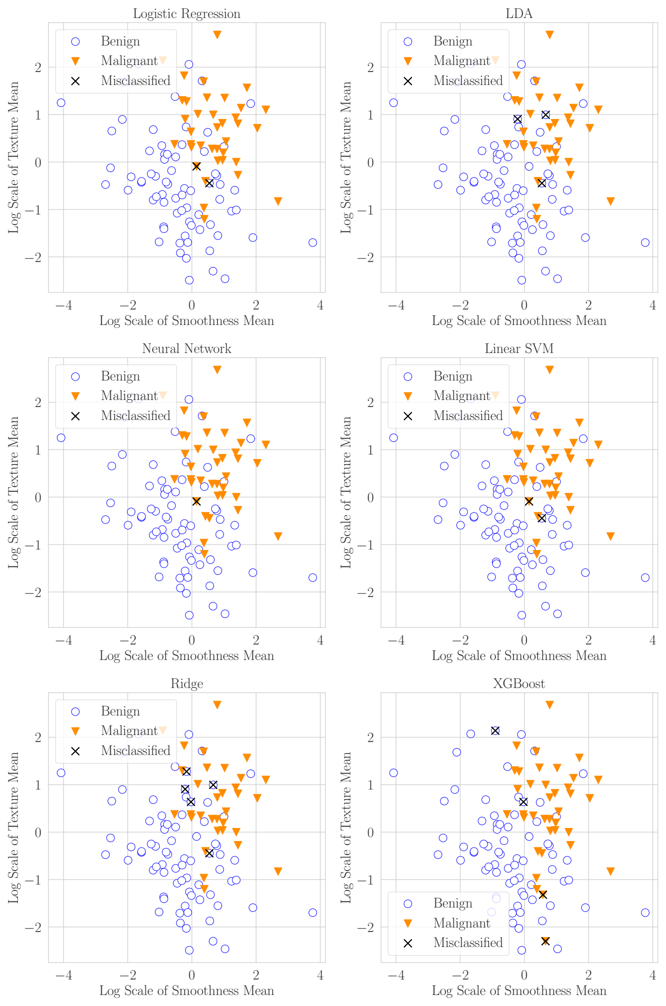

Figure 10 Plot prediction results on the experimental dataset showing that the location of the data points was incorrectly predicted for each algorithm.

Our experiments also show the efficiency in data cleaning and data processing, transformation before training models. We try to train the model with unprocessed data with Logistic Regression algorithm, that is, we do not apply linear transformations, do not delete outliers, the accuracy is about 85% to 90% based on enter custom hyperparameters.

In addition, we have carefully analyzed the composition of the predictions where the model makes mistakes to better understand, reports for each misclassified data point are generated as HTML files in the `src/` directory. This part is also shown in our notebook.

The mathematical theory behind NeuralNet is the implementation of a feedforward neural network for multi-class classification. Given a feature matrix $X \in \mathbb{R}^{n\times p}$, where n is the number of samples and p is the number of features, the goal is to predict a target vector $y \in 1, 2, \ldots, K^n$, where K is the number of classes. A feedforward neural network consists of an input layer, one or more hidden layers, and an output layer. Each layer is fully connected to the next layer, and the hidden layers apply a non-linear activation function $g(\cdot)$.

In this implementation, the neural network has two fully connected (Linear) layers, with the first layer having $p = 30$ input neurons and h hidden neurons, and the second layer having h hidden neurons and K output neurons. The activation function applied on the first layer is the Rectified Linear Unit (ReLU) function:

$$g(x) = \max(0, x)$$

The output of the network can be computed as follows:

$$\mathbf{h} = g(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)$$

$$\mathbf{o} = \mathbf{W}_2\mathbf{h} + \mathbf{b}_2$$

where $\mathbf{W}_1 \in \mathbb{R}^{h\times p}$ and $\mathbf{b}_1 \in \mathbb{R}^h$ are the weights and biases of the first layer, $\mathbf{W}_2 \in \mathbb{R}^{K\times h}$ and $\mathbf{b}_2 \in \mathbb{R}^K$ are the weights and biases of the second layer, and $\mathbf{x} \in \mathbb{R}^p$ is an input sample.

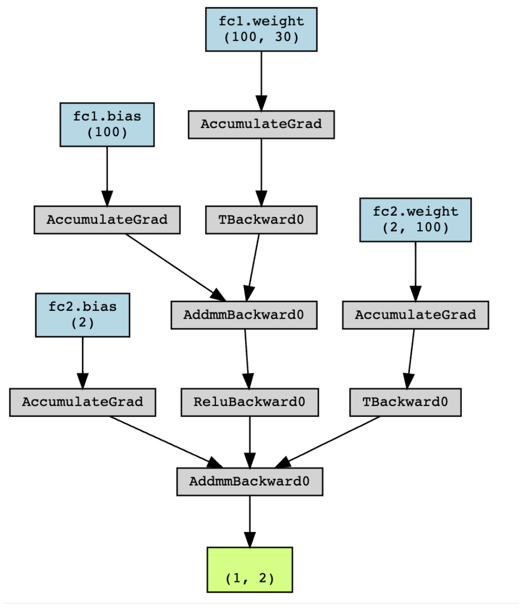

Figure 11 NeuralNet model experiment's visualisation

The training of the network is performed using Stochastic Gradient Descent (SGD) optimization and Cross-Entropy Loss as the loss function. Given the predicted output $\hat{\mathbf{y}}$ and the true target $\mathbf{y}$, the Cross-Entropy Loss is defined as:

$$\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_{i=1}^{n} y_i \log(\hat{y}_i)$$

SGD updates the network weights and biases using the gradients of the loss with respect to the parameters:

$$\mathbf{W}_1 \leftarrow \mathbf{W}_1 - \alpha \frac{\partial\mathcal{L}}{\partial\mathbf{W}_1}$$

$$\mathbf{b}_1 \leftarrow \mathbf{b}_1 - \alpha \frac{\partial\mathcal{L}}{\partial\mathbf{b}_1}$$

$$\mathbf{W}_2 \leftarrow \mathbf{W}_2 - \alpha \frac{\partial\mathcal{L}}{\partial\mathbf{W}_2}$$

It can be seen that the models give good results with negligible difference, in this project, we want to experiment with many different algorithms to realize the difference and experience. optimized for different algorithms.

Our experiments also show that overfitting is well controlled by techniques that we applied such as:

- Dropout: Randomly dropping out neurons from the network during each forward pass to prevent complex co-adaptations on training data.
- Weight Initialization: Proper weight initialization can prevent overfitting. For example, using small random values for the weights can reduce the risk of overfitting.
- Regularization: Introducing a penalty term to the loss function, such as L1 or L2 regularization, to discourage overfitting by reducing the magnitude of the model parameters.

In terms of interpreting models, we adopted the implantation of probabilistic predictions to better understand the influence of each variable on the final outcome, which is depicted in the graph below.

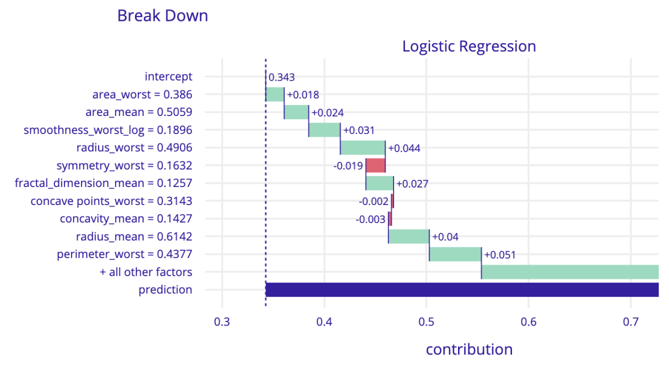

Figure 12 The contribution of each feature or predictor in a model to the overall prediction outcome.

It provides a visual representation of how each feature affects the prediction and the magnitude of its effect. The plot typically displays the prediction outcome before and after the effect of each feature is removed, allowing us to see the incremental effect of each feature on the final prediction.

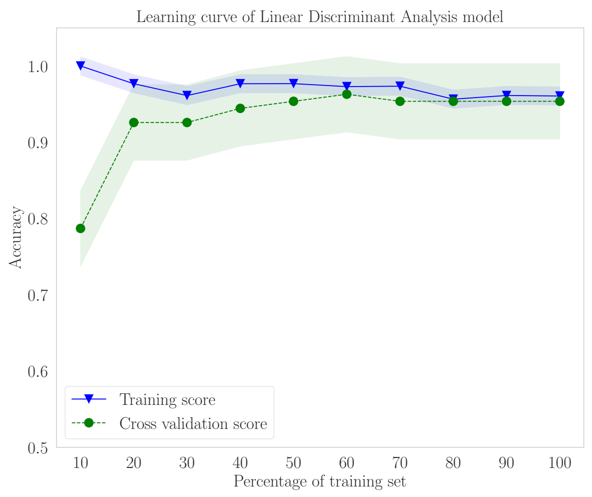

Figure 13 The contribution of each feature or predictor in a model to the overall prediction outcome.

Above we give an example of a learning curve of modes (LDA is an example), which as mentioned above, helps us a lot in understanding its performance as the amount of training data increases. It provides insight into how well the model is able to generalize to new data, how quickly it is able to learn, and whether it is suffering from overfitting or underfitting. The learning curve can be used to select a more appropriate model, adjust its hyperparameters, or provide guidance for collecting more data.

Our proposition

Based on the result, it appears that the Neural Network model has the highest accuracy and recall, as well as a high F1-score and ROC AUC. It may be the best option for the Breast Cancer classification problem. However, it is important to consider other factors such as interpretability, computational cost, and overfitting before making a final decision.

When working with medical data, it is crucial to pay attention to data quality, bias, and privacy concerns. The data should be checked for missing or incorrect values, and any bias should be addressed through techniques such as oversampling or data augmentation. Notes that we often consider and are cautious about when working with medical data sets such as:

- **Feature selection**: Medical data often contains a large number of features, and it is important to choose the most relevant features for the machine learning model Kotsiantis (2007). Techniques such as feature importance, feature correlation, and feature engineering can be used to identify the most important features.
- **Data quality**: Medical data often contains missing or incorrect values, which can negatively impact the performance of machine learning models. Therefore, it is important to check the data for missing values, outliers, and errors before applying machine learning techniques.
- **Bias**: Medical data can be biased in a number of ways, including sampling bias, measurement bias, and selection bias. This can lead to incorrect conclusions and affect the performance of machine learning models. To mitigate bias, techniques such as oversampling, data augmentation, and cross-validation can be used.
- **Model complexity**: Medical data can be high-dimensional and complex, and it is important to choose models that are able to handle this complexity without overfitting. Regularization techniques such as L1 and L2 regularization can be used to prevent overfitting.
- **Evaluation metrics**: When working with medical data, it is important to choose appropriate evaluation metrics that are sensitive to the problem at hand. For example, in a binary classification problem, accuracy is not always the best metric, as precision, recall, F1-score, and ROC AUC may be more relevant.

In terms of performance and computational cost, in cases where the amount of data is large this is something to consider, in our case the amount of data is small so the computational cost is negligible. Generally, linear algorithms such as Logistic Regression, LDA and Linear SVM tend to have lower computational costs compared to more complex algorithms such as Neural Networks and XGBoost. However, the exact cost will depend on the specific implementation and hardware being used.

Whether or not to choose one model for our problem depends on a number of factors and there is no one-size-fits-all answer.

Logistic Regression and LDA is a simple and widely used algorithm that can be a good starting point for binary classification problems. It can provide interpretable results and has a low computational cost, making it fast and easy to train. Additionally, it is a linear algorithm, which means that it can handle linear relationships between features and the target variable.

However, Logistic Regression and LDA may not be the best choice for more complex problems, or when the relationship between the features and target variable is non-linear. In these cases, more complex algorithms such as Neural Networks or XGBoost may be more appropriate.

It is also important to consider the performance of the Logistic Regression model compared to other models. From the table, the Logistic Regression model has a good accuracy (0.981481) and F1-score (0.972973), but its precision (0.972973) and recall (0.972973) are not as high as those of the Neural Network model (0.973684 and 1.0, respectively).

In conclusion, Logistic Regression, LDA can be a good choice for simple binary classification problems, but other factors such as data complexity, performance, and interpretability should also be considered before making a final decision.

Closing thoughts

We learned that for each phase of the project, it was extremely important to develop the idea that it was important to harmonize the elements that make the model the best, for example, we wondered if the outliers we found by algorithms whether deletion is necessary given that we have a relatively limited amount of data.

Another question that calls for a deep grasp of data is: Out of all the algorithms, which one can we use? Which algorithm do we need to find? Algorithms that produce great prediction results but which one will be the most stable? We can, of course, reach a trustworthy conclusion by testing using various metrics, testing on various data files, and testing using various test methodologies. However, this subject will continue to be challenging because it is difficult to correlate the properties of many machine learning algorithms with the characteristics of the data.

References

Aliferis CF, Tsamardinos I, Mansion P, Statnikov A, Hardin D. 2002. Machine learning models for lung cancer classification using array comparative genomic hybridization. 16th International FLAIRS Conference. pp. 67–71.

Kotsiantis SB. 2007. Supervised machine learning: A review of classification techniques. Informatica. 31:249–268.

Murphy KP. 2013. *Machine learning : a probabilistic perspective*. MIT Press. Cambridge, Mass. [u.a.].

Street WN, Wolberg WH, Mangasarian OL. 1993. Nuclear feature extraction for breast tumor diagnosis. IST/SPIE 1993 International Symposium on Electronic Imaging: Science and Technology. 1908:861–870.

Annexes

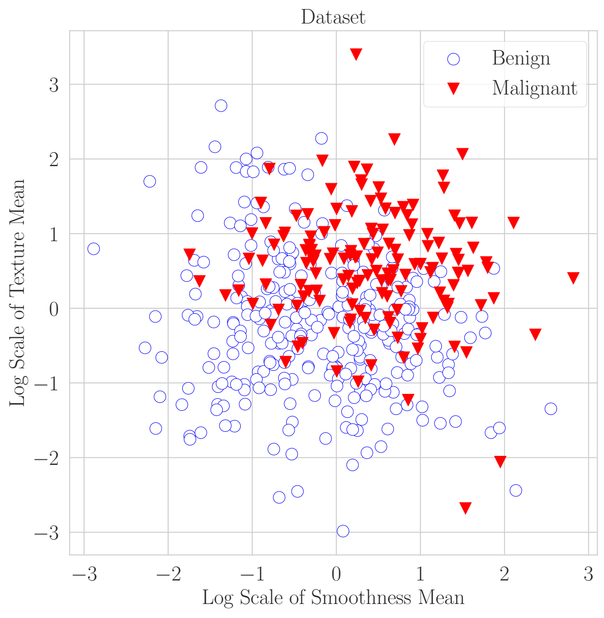

Figure 14 Dataset after transformation by axes of Log Texture Mean and Log Smoothness Mean.

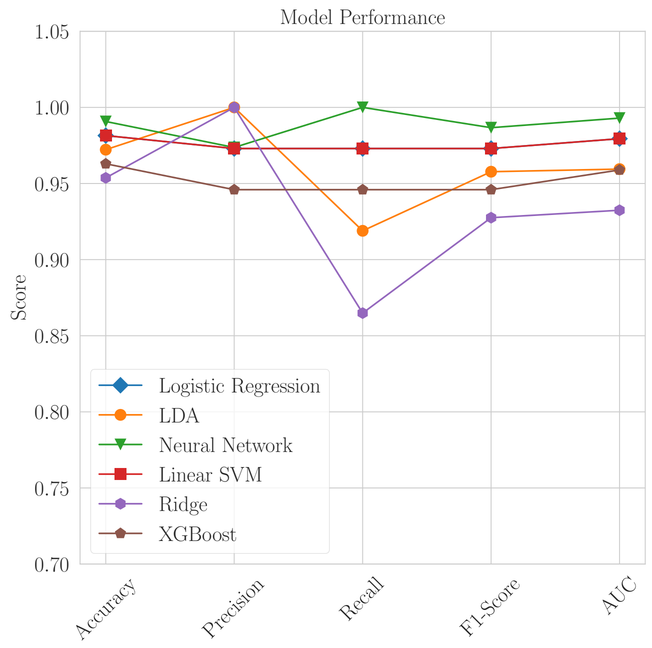

Figure 15 Decision plot from Logistic Regression model.

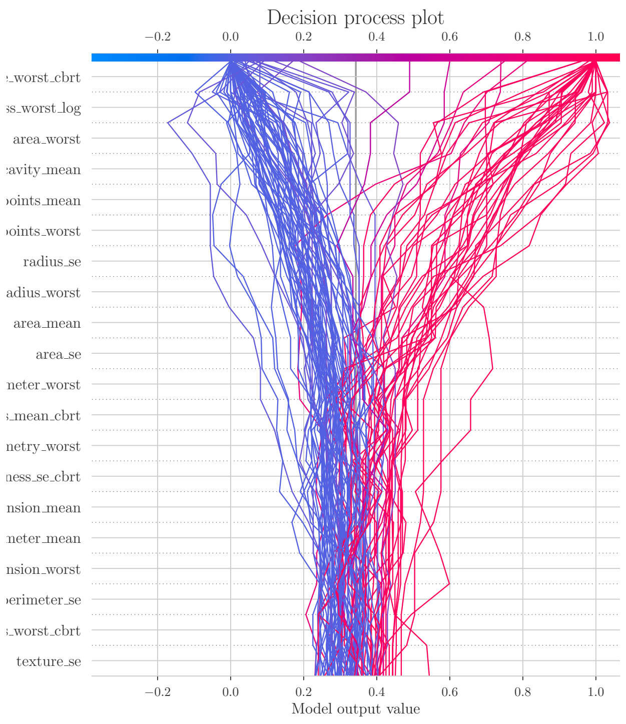

Figure 16 Metrics representation for all models

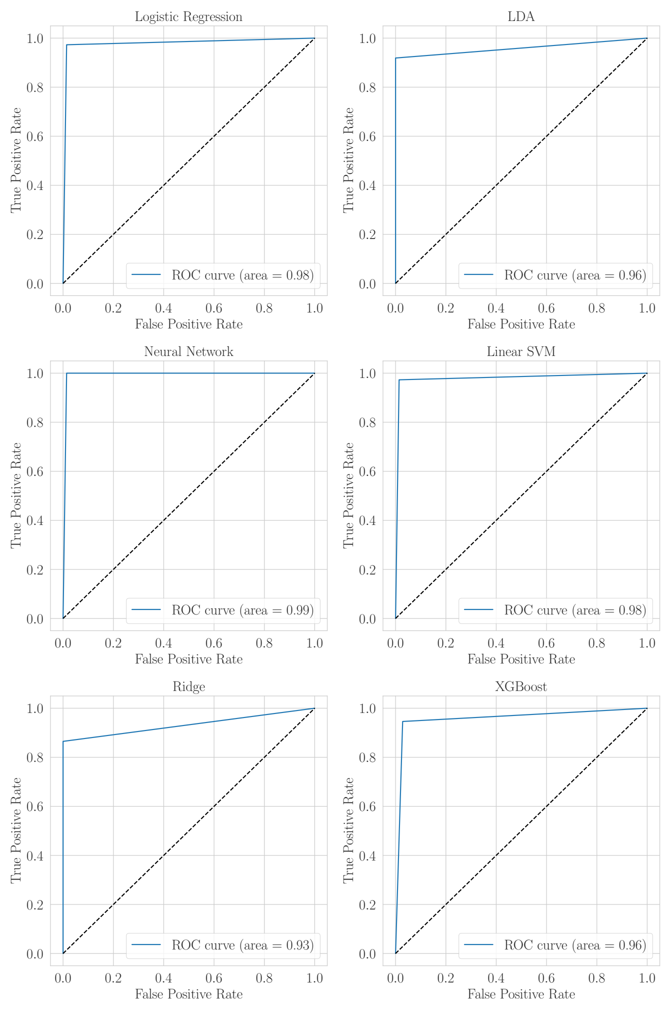

Figure 17 Confusion matrices of all models experiments.

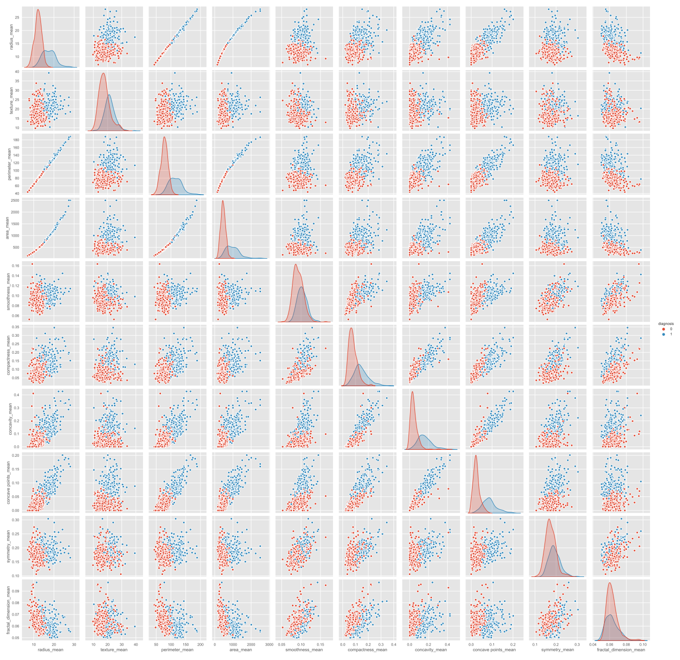

Figure 18 Pairplot of MEAN features.

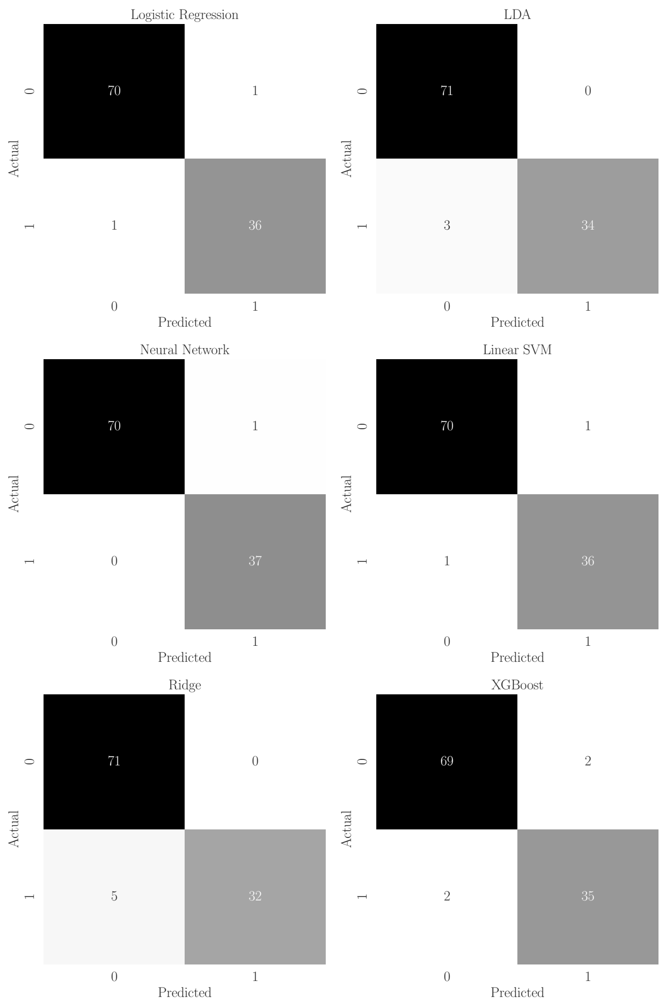

Figure 19 ROC Curves of all models experiments.