Universidad Autónoma de Madrid

Escuela Politécnica Superior

23|24

Bachelor thesis

Comparative Analysis of Classifiers for Breast Cancer Detection with Visualizations

Iván Sotillo del Horno

Campus Internacional excelencia UAM CSIC

UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Bachelor as Ingeniería Informática (modalidad bilingüe)

BACHELOR THESIS

Comparative Analysis of Classifiers for Breast Cancer Detection with Visualizations

Author: Iván Sotillo del Horno
Advisor: Alejandro Bellogín Kouki

marzo 2024

Francisco Tomás y Valiente, n^o 1
Madrid, 28049
Spain

Iván Sotillo del Horno
Comparative Analysis of Classifiers for Breast Cancer Detection with Visualizations

Iván Sotillo del Horno

PRINTED IN SPAIN

A mi madre y a mi abuela, cuya lucha contra el cáncer de mama me ha inspirado a realizar este trabajo.

Resumen

Esta tesis presenta un análisis comparativo de clasificadores para la detección del cáncer de mama y el uso de Inteligencia Artificial Explicable (XAI) para interpretar los resultados. En la fase inicial se realizará la construcción y optimización de los modelos de clasificación, estos clasificadores analizarán los resultados de las biopsias de aguja fina y clasificarán las muestras como benignas o malignas.

Posteriormente, se realiza una comparación de rendimiento comparando métricas como la puntuación F1 o la *recall*. El objetivo es identificar el mejor clasificador de acuerdo a nuestras métricas. Una vez encontrado el mejor modelo, nos adentramos más en él para entender cómo funciona. Para esto, utilizaremos SHAP (SHapley Additive exPlanations), un método de XAI que nos permite ver la importancia de cada característica y cómo contribuyen a la decisión final del modelo. Esto nos permitirá no solo clasificar las muestras, sino también entender por qué el modelo ha tomado esa decisión, lo que puede ser un avance en la comprensión de los modelos de IA para fines médicos.

Palabras clave

Detección de Cáncer de Mama, Clasificadores, Análisis Comparativo, Interpretabilidad, SHAP, IA Explicable, Visualización

Abstract

This thesis presents a comparative analysis of base and ensemble classifiers for breast cancer detection and the use of eXplainable AI (XAI) to interpret the results. The initial phase involves constructing and optimizing the classifier models, these classifiers will analyze the results from fine needle biopsy aspirations and classify the samples as benign or malignant.

Following this, a performance comparison is conducted comparing metrics such as the F1 score or the recall. The aim is to identify the best classifier regarding our metrics. Once the best classifier model is found, we dive deeper into it to understand how it works. For this, we will use SHAP (SHapley Additive exPlanations), a method of XAI (eXplainable AI) that allows us to see the importance of each feature, and how they contribute to the final decision of the model. This will allow us to not only classify the samples but also to understand why the model has made that decision which can be a step forward in understanding AI models for medical purposes.

Keywords

Breast Cancer Detection, Classifiers, Comparative Analysis, Interpretability, SHAP, eXplainable AI, Visualization

Table of Contents

1 Introduction **1**

1.1 Motivation . . . 1

1.2 Objectives . . . 1

1.3 Structure of the document . . . 1

2 State of the art **3**

2.1 Base and Ensemble Classifiers . . . 3

2.1.1 Base Classifiers . . . 3

2.1.2 Ensemble Classifiers . . . 3

2.2 Classifier Optimization . . . 3

2.3 Evaluation of Classifiers . . . 3

2.4 Explainable AI . . . 3

2.4.1 Explainable AI . . . 3

2.4.2 SHAP . . . 3

2.5 Web Application Development . . . 4

3 Design and Implementation **5**

3.1 Project Structure . . . 5

3.2 Requirements . . . 7

3.2.1 Functional Requirements . . . 7

3.2.2 Non-Functional Requirements . . . 7

3.3 Exploratory Data Analysis . . . 8

3.3.1 Descriptive Statistics . . . 8

3.3.2 Data Visualization . . . 9

3.4 Data Preprocessing . . . 9

3.4.1 Scaling and Normalization . . . 9

3.4.2 Principal Component Analysis . . . 9

3.5 Building and Optimizing Classifiers . . . 10

3.5.1 Comparison of Classifiers . . . 11

3.5.2 Optimization of Classifiers . . . 12

3.6 SHAP Implementation . . . 13

3.7 Web Application Development . . . 14

4 Experiments and Results **17**

4.1 Exploratory Data Analysis . . . 17

4.2 Classifier Comparison 17
4.2.1 Base Classifiers Comparison 17
4.2.2 Ensemble Classifiers Comparison 17
4.2.3 Choosing the Best Classifier 17
4.3 SHAP Analysis 17
4.3.1 Global Interpretability 17
4.3.2 Local Interpretability 17
5 Conclusions and Future Work **19**
5.1 Conclusions 19
5.2 Future Work 19
Bibliography **21**
Acronyms **23**
Appendices **25**

Lists

List of algorithms

List of codes

3.1 Building the pipeline 11
3.2 Obtaining the predicted probability of the pipeline and calculating the SHAP values . . . 14
3.3 Plotting SHAP 14

List of equations

3.1 Standardization 9

List of figures

3.1 Structure of the project 5
3.2 Fine Needle Aspiration Biopsy 6
3.3 Standardization 10
3.4 PCA 10
3.5 Web Application 15
3.6 Web Application 16

List of tables

1 Introduction

1.1. Motivation

Breast cancer is the most common cancer type among women [1]; in 2020, there were more than 2.26 million women diagnosed with breast cancer [1], being the second leading cause of death among women in the United States [2]. Early detection is a crucial step for improving survival rates. With the current analysis techniques of FNAB (Fine Needle Aspiration Biopsy), we have a sensitivity (ability of a test to identify positive cases correctly) of 0.927 [3]. Therefore, there is a need for a more accurate interpretation of those tests.

Machine learning is a branch of artificial intelligence that focuses on developing algorithms that can learn from data and extract patterns from it to be then able to generalize it to unseen data. In this case, we care about classifiers, whose potential is in the ability to learn from a dataset and then on unseen data being able to classify it as one class or another; in this case, we will be able to classify as benign or malign the results of a fine needle aspiration.

The potential of classifiers in breast cancer detection is immense. However, the effectiveness of the different classifiers can vary; this is why it is crucial to understand how each classifier works, how to tweak it, and how to make them as precise and effective as possible, which is the goal of this thesis.

Finding the best possible classifier for this problem would impact cancer detection tasks, facilitating healthcare professionals in their diagnostic responsibilities and, ultimately, improving patient outcomes.

1.2. Objectives

1.3. Structure of the document

2

State of the art

2.1. Base and Ensemble Classifiers

2.1.1. Base Classifiers

2.1.2. Ensemble Classifiers

2.2. Classifier Optimization

2.3. Evaluation of Classifiers

2.4. Explainable AI

2.4.1. Explainable AI

2.4.2. SHAP

SHapley Additive exPlanations (SHAP) [4] is a model-agnostic approach to Explainable Artificial Intelligence (XAI), that is, it can be used with any model since it only uses the input and output. In short, how SHAP works, is that it uses the Shapley values from cooperative game theory to explain the output of any machine learning model. The Shapley value is a concept from cooperative game theory, it is a way to fairly distribute the loot obtained by a coalition of players. The difference is that in SHAP instead of players we have features and instead of loot, we have the predicted output as a probability. The Shapley value is the average contribution of a feature to the prediction, and it is calculated by averaging the marginal contributions of a feature to the prediction over all possible orderings of the features. The marginal contribution of a feature to the prediction is the difference between the prediction with the feature and the prediction without the feature. This is done for all possible subsets of features, and then

the Shapley value is the average of all these marginal contributions. This is done for all the features, and then we have the Shapley values, which are the importance of the features for the prediction.

2.5. Web Application Development

3

Design and Implementation

In this section the design of the project will be shown, showing the structure of the project, its requirements, and the implementation of the different parts of the project. Additionally, the implementation of each part will be explained.

3.1. Project Structure

The structure of the project is divided into four main modules corresponding to the main stages of the project: dataset, classifiers, XAI, and web application. The dataset module is responsible for loading the dataset and splitting it into a 70 % training set and a 30 % test set. Then in the classifiers module, we do the Exploratory Data Analysis (EDA), data preprocessing, and building and optimizing the classifiers. The XAI module is responsible for the implementation of the SHAP algorithm and the analysis of the results. Finally, the web application module is responsible for the development of the web application. The structure of the project is shown in Figure 3.1. Now we will describe each module in more detail.

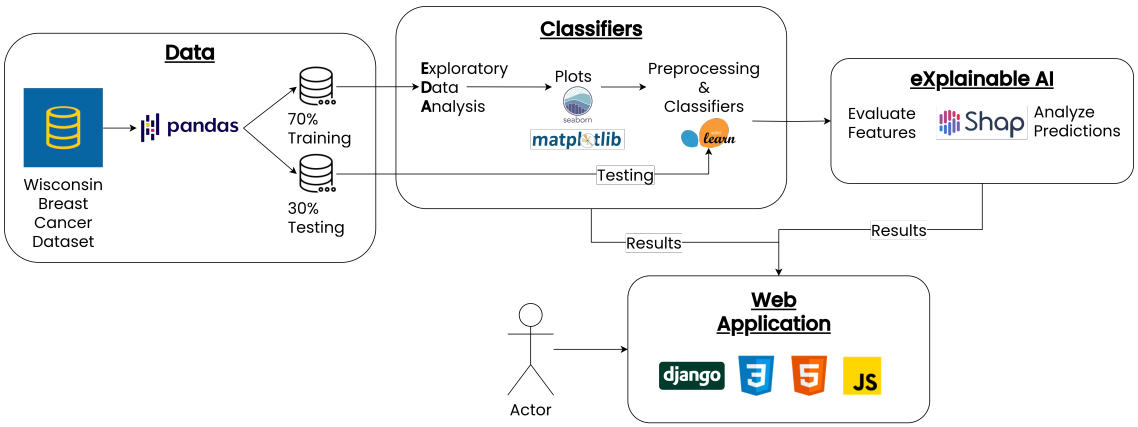

Figure 3.1: Structure of the project

1. **Dataset module**: This module is responsible for loading the dataset and splitting it into a 70 % training set and a 30 % test set. The dataset used in this project is the Breast Cancer Wisconsin dataset [5], which contains the parameters of the cell nuclei of the sample obtained by a Fine Needle Aspiration

Biopsy (FNAB) of breast masses. The dataset is loaded using the *pandas* library. The dataset is then split into a 70 % training set and a 30 % test set using *scikit-learn*. From now on, only the training will be used for the Exploratory Data Analysis, data preprocessing, and building and optimizing the classifiers. The test set will be used to evaluate the classifiers. This ensures that we avoid data leakage. An example of a FNAB is shown in Figure 3.2.

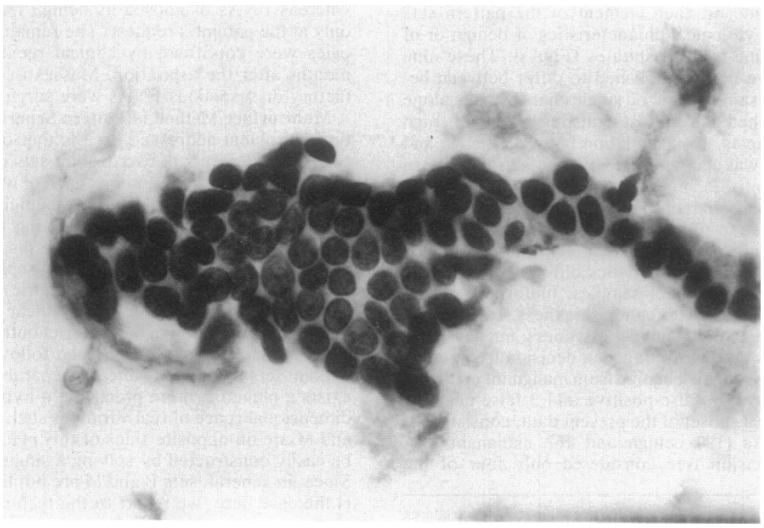

Figure 3.2: Fine Needle Aspiration Biopsy [6]

2. **Classifiers module**: This module is responsible for the EDA, data preprocessing, and building and optimizing the classifiers. The EDA is carried out using *matplotlib* and *seaborn* to obtain descriptive and visual statistics of the dataset. The data preprocessing is carried out using *scikit-learn* to scale and normalize the data and to apply Principal Component Analysis (PCA), this is made using pipelines that facilitate the process of building and optimizing the classifiers and their reproducibility. The building and optimization of the classifiers are carried out using *scikit-learn* using the training set to build and optimize the classifiers. Once the classifiers are built and optimized, they are evaluated using the test set.

3. **XAI module**: This module is responsible for the implementation of the SHAP algorithm and the analysis of the results. This is implemented using the *shap* library, which is a Python library that allows us to calculate the SHAP values of the classifiers, this values will be used to understand the importance of the features for each classifier and to understand how a prediction was made. This is what makes this project more interpretable than just using classifiers since this will allow a doctor to understand why a prediction was made.

4. **Web application module**: This module is responsible for the development of the web application. This website was made using *Django* as the backend, *HTML*, *CSS*, and *JavaScript* as the frontend. The website is mainly a way of visualizing the results of the project, it allows the user to see a list of all classifiers and their metrics, which features are the most important for that classifier, and how some predictions were made. It also allows the user to compare two classifiers and see the differences in

their metrics and the importance of the features.

3.2. Requirements

In this section, we will define the requirements of the project. We will divide them into functional (what the system should do) and non-functional (how the system should do it) requirements.

3.2.1. Functional Requirements

Dataset

- **FR-1.-** The application should allow the use of the Wisconsin breast cancer dataset or one with the same features.
- **FR-2.-** The dataset should be split into a training and testing dataset using a 70-30 ratio.

Classifiers

- **FR-3.-** The application should employ the features provided in the dataset for classifier comparison and XAI.
- **FR-4.-** The application should work with multiple classifiers for breast cancer detection.
- **FR-5.-** The application should preprocess the dataset before feeding it to the classifiers. This may include steps such as normalization, and feature selection, focusing on reproducibility and repeatability.
- **FR-6.-** The application should use a test dataset for classifier comparison.
- **FR-7.-** The application should use evaluation metrics to determine the effectiveness of the classifiers taking into account the nature of the problem.

Explainable Artificial Intelligence

- **FR-8.-** The application should show which features are the most important for a classifier.
- **FR-9.-** The application should show how sure the classifier was of its prediction.
- **FR-10.-** The application should show which features determined if the sample was benign or malignant.

Web Application

- **FR-11.-** The web application should have a home page with a slideshow with information.
- **FR-12.-** The web application should display a list with all the classifiers.
- **FR-13.-** The web application should show a details page about each classifier, showing its metrics, most important features, and examples of decisions made by the classifier.
- **FR-14.-** The web application should compare any two classifiers showing their metrics and most important features.

3.2.2. Non-Functional Requirements

Classifiers

- **NFR-1.-** The programming language should be Python.
- **NFR-2.-** The classifiers should be created and trained in a Jupiter Notebook.
- **NFR-3.-** The Pandas library should be used for data manipulation.
- **NFR-4.-** The Scikit-learn library should be used for the classifiers.
- **NFR-5.-** The Matplotlib and Seaborn libraries should be used for the plots.
- **NFR-6.-** The Shap library should be used to understand the feature importance of the models and how a sample decision was made.

Web Application

- **NFR-7.-** The web application should be designed for desktop usage.
- **NFR-8.-** The web application should have a pink color palette and a modern design.
- **NFR-9.-** The web application should be implemented in Django.
- **NFR-10.-** The web application should have an intuitive interface.

3.3. Exploratory Data Analysis

As it has been previously mentioned the dataset used in this project is the Breast Cancer Wisconsin dataset [5]. This dataset is composed of 569 samples of breast masses obtained by FNAB, then these samples were parametrized, meaning that now we will work with numbers instead of images. As a result of this process, we have 30 features that describe the cell nuclei of the sample, these features are the mean, standard error, and worst of the ten different parameters of the cell nuclei.

Before using these values in the classifier we first need to understand the dataset, this is done using EDA techniques. This involves, first obtaining a general understanding of the dataset, then visualizing the dataset, and finally understanding the relationships between the features. All of this is done only using the training set, this is to avoid data leakage.

3.3.1. Descriptive Statistics

To have a general understanding of the dataset we can use the *pandas* library to load the dataset and then use the *.describe()* method to see an overview of the features in the dataset, their mean, standard deviation, minimum, and maximum values; and then use the *.skew()* method to see the skewness of the features. With this, we would have the needed descriptive statistics of the dataset.

3.3.2. Data Visualization

After seeing a statistical overview of the dataset, we can now dive into visualizing the dataset. This is done using the *matplotlib* and *seaborn* libraries. With these libraries, we will be able to see more visually the distribution of the target variable, the distribution of the features, and the relationships between the features. To achieve this, we will use histograms, correlation matrices, pair, density, and violin plots among others. This will give us a better understanding of the dataset and the relationships between the features which will allow us to make more informed decisions when preprocessing the dataset and building the classifiers.

3.4. Data Preprocessing

Once we have done the EDA and we know how the dataset is structured, how the different features are distributed, and the relationships between the features, we can start preprocessing the dataset. To achieve this we will standardize the features and then reduce the dimensionality of the features by using PCA, this will result in a dataset that is easier to work with and that will allow us to build better classifiers [7]. To facilitate this process we will use pipelines, which will allow us to easily build and optimize the classifiers and to ensure reproducibility and prevent data leakage [8], this means that we won't directly have to modify the dataset. The Code Snippet 3.1 shows how the pipeline is built.

3.4.1. Scaling and Normalization

This is done using the *scikit-learn* library. The first step is to handle the missing values, this is done using the *SimpleImputer* class, which replaces the missing values with the mean of the feature. Then we scale the features using the *StandardScaler* class, which scales the features to have a mean of 0 and a standard deviation of 1, this is done like the Equation 3.1, where μ is the mean of all the samples of the feature, σ is the standard deviation, and x is the value we want to standardize. We can see in Figure 3.3 how the distribution of the feature stays the same but the mean is 0 and the standard deviation is 1 after standardization.

$$x' = \frac{x - \mu}{\sigma} \tag{3.1}$$

3.4.2. Principal Component Analysis

Lastly, we select the most important variables using the PCA, what this does is that it takes the features and transforms them into a new set of features that are linear combinations of the original features trying to maximize the variance of the new features while the correlation between the new

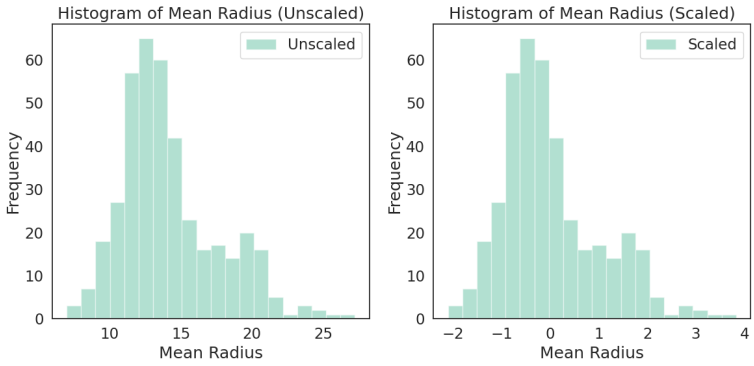

Figure 3.3: Standardization before and after of mean radius

features is 0, this is done using orthogonal transformations [9], in our case instead of specifying the number of components we want to keep, we specify the percentage of variance we want to keep, in this case, we want to keep 95 % of the variance, this is done so that we can reduce the number of features while keeping most of the information. In our case, when we plot the cumulative explained variance we can see that with 10 features we can keep 95 % of the variance, this is shown in the Figure 3.4. The Code Snippet 3.1 shows how the pipeline is built.

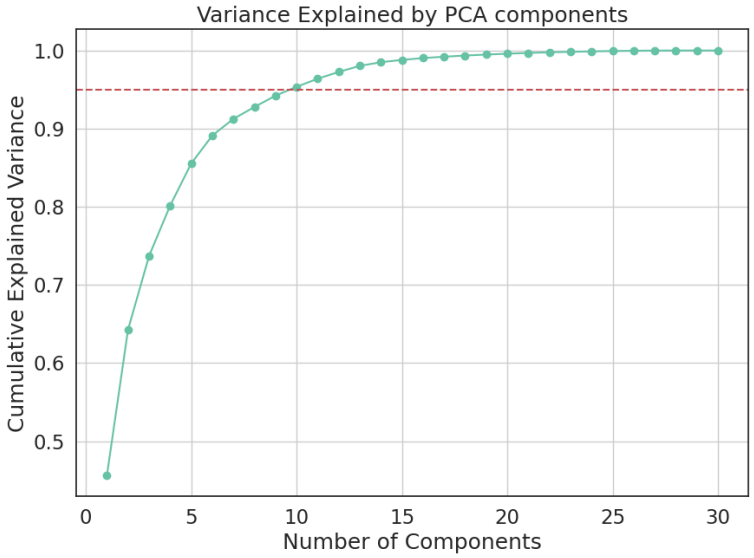

Figure 3.4: Cumulative explained variance when using PCA

3.5. Building and Optimizing Classifiers

To build the classifiers, *Scikit-learn* was used. This library was chosen because it is the most popular library for building machine learning models in *Python* and it has a wide variety of classifiers and ensemble methods. The classifiers were built in a *Jupyter Notebook* to allow reproducibility and to facilitate the process of building and optimizing the classifiers. The classifiers can be divided into two

Code 3.1: Building the pipeline with the StandardScaler and PCA

```
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('pca', PCA(n_components=0.95)),
    ('clf', Classifier())
])
```

types, base and ensemble, base classifiers are the simplest classifiers, and ensemble classifiers are a combination of base classifiers.

The base classifiers used are:

- **Logistic Regression**
- **Multilayer Perceptron**
- **Support Vector Machine**
- **Decision Tree**
- **Stochastic Gradient Descent**
- **K-Nearest Neighbors**
- **Linear Discriminant Analysis**
- **Gaussian Naive Bayes**
- **Quadratic Discriminant Analysis**

Based on the best results of the base classifiers, the ensemble classifiers were built. Some of these ensemble classifiers allow one to choose the base classifier to use, for these cases, the best base classifiers were used. The ensemble classifiers used are:

- **Random Forest**
- **AdaBoost**
- **Gradient Boosting**
- **Bagging**
- **Voting**
- **Stacking**

3.5.1. Comparison of Classifiers

To decide if a classifier is better than another, evaluation metrics were used, and to choose the appropriate metrics for this problem, we first need to understand the basic metrics used in classification problems [10]:

- **Accuracy**: Proportion of correct predictions over the total number of instances evaluated. $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$
- **Recall**: Proportion of true positive results among the number of all positive instances. $Recall = \frac{TP}{TP+FN}$
- **Precision**: The proportion of true positive results among the number of cases that were predicted to be positive.

$Precision = \frac{TP}{TP+FP}$

- **Specificity**: The proportion of true negative results among the number of cases that were actually negative.
$Specificity = \frac{TN}{TN+FP}$

Now we have to choose the appropriate metrics for this problem, the most important thing for this problem is to minimize the number of false negatives, since a false negative means that a patient with cancer was classified as healthy, and this could have fatal consequences. To minimize the number of false negatives, we will use the *Recall*, but the problem of only taking into account the recall is that to maximize it, the model could classify all samples as positive, to avoid this we will use the *Precision* since this will penalize the model for classifying a sample as positive if it was not. To combine these two metrics we will use the *F1 Score*, which is the harmonic mean of the precision and recall. We will also use the *F2 Score*, which is the weighted harmonic mean of the precision and recall, this will give more importance to the recall since we want to minimize the number of false negatives [11]. So, the evaluation metrics sorted by the importance that we will use are:

- **F1 Score**:
$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$
- **Recall**:
$$Recall = \frac{TP}{TP + FN}$$
- **F2 Score**:
$$F2 = 5 \times \frac{Precision \times Recall}{\frac{4}{Precision} + \frac{1}{Recall}}$$
- **Precision**:
$$Precision = \frac{TP}{TP + FP}$$

3.5.2. Optimization of Classifiers

To maximize these metrics in the classifiers, we have to optimize the hyperparameters of the classifiers. The hyperparameters are the parameters that are not learned by the model, and they are set before the training process, these can, for example, be the number of neighbors in the K-Nearest Neighbors classifier, the number of neurons in the Multilayer Perceptron classifier, or the maximum depth of the Decision Tree classifier. To optimize these hyperparameters, we used the *RandomizedSearchCV* followed by the *GridSearchCV*. The reason for using the *RandomizedSearchCV* first is that can achieve similar results to the *GridSearchCV* but with a lower computational cost [12], this will allow us to explore a wider range of hyperparameters and have a range of values to use in the *GridSearchCV*. Then we used the *GridSearchCV* to find the best hyperparameters in the range of values found by the *RandomizedSearchCV*.

These searches had a 5-fold Cross-Validation (CV), which means that the dataset was split into 5 parts, 4 parts were used for training and 1 part was used for testing, this process was repeated 5 times, each time with a different part for testing, and the results were averaged, using the *F1 Score* as the

metric to optimize. All the *RandomizedSearchCV* and *GridSearchCV* were done using the same seed, this means that the split of the dataset was the same for all the classifiers, this ends up in a more reliable result and also allows the reproducibility of the results. This process was done to avoid overfitting and to have a more reliable result, of course, this process was only done in the training set, to avoid data leakage.

3.6. SHAP Implementation

Building and optimizing the classifiers is a crucial step and can be helpful in a clinical situation, but it can only get us so far. We also need to know why the model made a certain decision, for this, there is a branch of Artificial Intelligence (AI) called Explainable Artificial Intelligence, this branch of AI is focused on making Machine Learning (ML) models more than a black box which is fed with an input and provides an output, it is focused on making these models transparent, allowing clinical practitioners to understand why a certain decision was made [13]. To achieve this we will use the SHapley Additive exPlanations library. The reason for using SHAP is that it has been proven to achieve great explanations for ML models in the medical field [14].

We will use SHAP to have a global understanding of the importance of the features in the models and a local understanding of how a certain decision was made. This will allow us to understand which features are the most important for a classifier, how sure the classifier was of its prediction, and which features determined if the sample was benign or malignant. This will also be done in the *Jupiter Notebook* to ensure reproducibility.

To be able to calculate the SHAP values, we need to create an *Explainer* object, this object is created with the classifier as a parameter, but in our case, we are not using just the classifier but pipelines, this means, that we will have to find a way to obtain the predicted probability of the pipeline. To do this, we will use the *predict_proba* method of the pipeline, this method will return the predicted probability of the sample being malignant no matter which classifier is used, with this, we will be able to create the *Explainer* object and calculate the SHAP values, all this process is show in the Code Snippet 3.2.

After we have the SHAP values, we will be able to use them to understand the importance of the features for the classifiers by plotting a graph with the SHAP values of all the samples and also one for each single feature which will show the behavior of that feature, meaning that we will be able to see if the higher the value of the feature the higher the probability of the sample being malignant or benign. We will also be able to use them to understand how a certain prediction was made by plotting the SHAP values of a certain sample. This will allow us to understand why a certain prediction was made and which features were the most important for that prediction, to do this we will plot the prediction of three cases where the classifier was sure the sample was benign, three where it was sure it benign, the three closest to the decision boundary and the misclassified ones. In the Code Snippet 3.3 we can see how

to code the global feature importance, the behavior of a single feature, and the local explanation of a prediction for a certain classifier, this will be done for all the classifiers, features, and the aforementioned samples

Code 3.2: Obtaining the predicted probability of the pipeline and calculating the SHAP values

```
def model_predict_proba(data_asarray):
    data_asframe = pd.DataFrame(data_asarray, columns=X_train.columns)
    proba = lr_pipeline.predict_proba(data_asframe)
    return proba[:, 1]


# explainer object
explainer = shap.Explainer(model_predict_proba, X_train)

shap_values = explainer.shap_values(X_test)

explainer_x_test = explainer(X_test)
```

Code 3.3: Plotting the SHAP global features, single feature, and local explanation

```
shap.initjs()

# global feature importance
shap.plots.bar(explainer_x_test, max_display=25)
shap.plots.beeswarm(explainer_x_test, max_display=30, alpha=0.75)

# single feature importance
shap.dependence_plot(feature, shap_values, X_test, alpha=0.65, show=False)
ax = plt.gca()
x = ax.collections[0].get_offsets()[:, 0]
y = ax.collections[0].get_offsets()[:, 1]
regression_model = LinearRegression().fit(x.reshape(-1, 1), y)
x_range = np.linspace(x.min(), x.max(), 100)
y_range = regression_model.predict(x_range.reshape(-1, 1))
plt.plot(x_range, y_range, color='red')

# prediction explanation
shap.plots.waterfall(explainers_x_test[prob][idx], show=False)
```

3.7. Web Application Development

To make the results more accessible and visual, we have built a web application using *Django*. This framework was used because *Python* has been the main programming language used in the project and *Django* was the most familiar framework. For the front-end, we used HyperText Markup Language (HTML), Cascading Style Sheets (CSS), and JavaScript (JS). The web application is hosted in *Render*

with the database in *Neon* this makes it accessible using this link, since the free hosting of *Render* is not always available, we have also made a video showing the web application in action, this video can be found in this link or if not the project can be run locally cloning the repository. The home page of the web application is shown in Figure 3.5, an example of the comparison of two classifiers is shown in Figure 3.6.

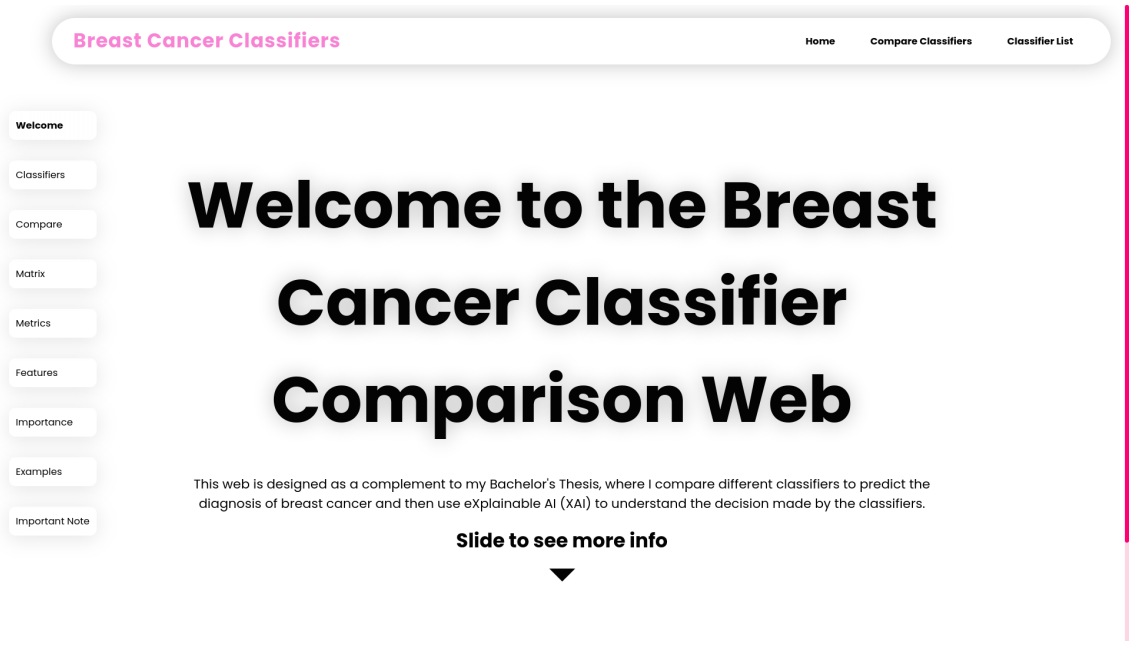

Figure 3.5: Home page of the web application

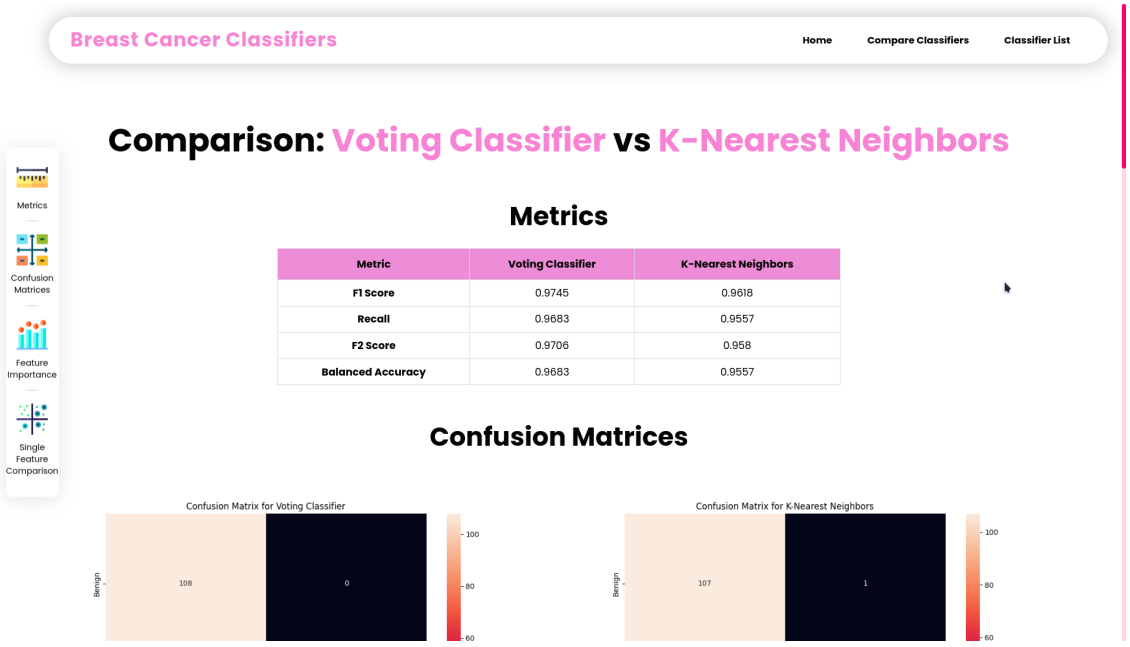

Figure 3.6: Comparison of two classifiers

4

Experiments and Results

4.1. Exploratory Data Analysis

4.2. Classifier Comparison

4.2.1. Base Classifiers Comparison

4.2.2. Ensemble Classifiers Comparison

4.2.3. Choosing the Best Classifier

4.3. SHAP Analysis

4.3.1. Global Interpretability

4.3.2. Local Interpretability

5

Conclusions and Future Work

5.1. Conclusions

5.2. Future Work

Bibliography

[1] WCRF International, “Breast cancer statistics | World Cancer Research Fund International.”

[2] American Cancer Society, “Breast Cancer Statistics | How Common Is Breast Cancer?.”

[3] Y.-H. Yu, W. Wei, and J.-L. Liu, “Diagnostic value of fine-needle aspiration biopsy for breast mass: a systematic review and meta-analysis,” *BMC Cancer*, vol. 12, p. 41, Jan. 2012.

[4] S. M. Lundberg and S.-I. Lee, “A Unified Approach to Interpreting Model Predictions,” in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[5] O. M. William Wolberg, “Breast Cancer Wisconsin (Diagnostic),” 1993.

[6] J. A. M. Sidey-Gibbons and C. J. Sidey-Gibbons, “Machine learning in medicine: a practical introduction,” *BMC Medical Research Methodology*, vol. 19, pp. 1–18, Dec. 2019. Number: 1 Publisher: BioMed Central.

[7] I. Dinç, M. Sigdel, S. Dinç, M. S. Sigdel, M. L. Pusey, and R. S. Aygün, “Evaluation of Normalization and PCA on the Performance of Classifiers for Protein Crystallization Images,” *Proceedings of IEEE Southeastcon / IEEE Southeastcon. IEEE Southeastcon*, vol. 2014, p. 10.1109/SECON.2014.6950744, Mar. 2014.

[8] K. Zhao, “Pre-Process Data with Pipeline to Prevent Data Leakage during Cross-Validation.,” Sept. 2019.

[9] A. Daffertshofer, C. J. Lamoth, O. G. Meijer, and P. J. Beek, “PCA in studying coordination and variability: a tutorial,” *Clinical Biomechanics*, vol. 19, pp. 415–428, May 2004.

[10] H. M and S. M.N, “A Review on Evaluation Metrics for Data Classification Evaluations,” *International Journal of Data Mining & Knowledge Management Process*, vol. 5, pp. 01–11, Mar. 2015.

[11] M. Rutecki, “Best techniques and metrics for Imbalanced Dataset.”

[12] J. Bergstra and Y. Bengio, “Random Search for Hyper-Parameter Optimization,”

[13] K. Borys, Y. A. Schmitt, M. Nauta, C. Seifert, N. Krämer, C. M. Friedrich, and F. Nensa, “Explainable AI in medical imaging: An overview for clinical practitioners – Beyond saliency-based XAI approaches,” *European Journal of Radiology*, vol. 162, p. 110786, May 2023.

[14] R. Massafra, A. Fanizzi, N. Amoroso, S. Bove, M. C. Comes, D. Pomarico, V. Didonna, S. Diotaiuti, L. Galati, F. Giotta, D. La Forgia, A. Latorre, A. Lombardi, A. Nardone, M. I. Pastena, C. M. Ressa, L. Rinaldi, P. Tamborra, A. Zito, A. V. Paradiso, R. Bellotti, and V. Lorusso, “Analyzing breast cancer invasive disease event classification through explainable artificial intelligence,” *Frontiers in Medicine*, vol. 10, 2023.

Acronyms

AI Artificial Intelligence.

CSS Cascading Style Sheets.

CV Cross-Validation.

EDA Exploratory Data Analysis.

FNAB Fine Needle Aspiration Biopsy.

HTML HyperText Markup Language.

JS JavaScript.

ML Machine Learning.

PCA Principal Component Analysis.

SHAP SHapley Additive exPlanations.

XAI Explainable Artificial Intelligence.

Appendices


UÁM
Universidad Autónoma
de Madrid