References

Machine Learning in Computational Biology
Assignment 2

Konstantinos Konstantinidis
Student number: 7115152400017

May 6, 2025

Repo: The repository for this assignment can be found here:
`https://github.com/KonsKons26/Assignment-2`

Contents

1 Abstract

In this assignment, we are tasked with classifying a dataset consisting of 512 samples of fine needle aspirate (FNA) measurements from breast masses. Each sample is represented by 30 features, and the goal is to predict whether the mass is malignant or benign. The dataset will be split into a training set and a holdout set; the training set will be used to train a set of classifiers, and the holdout set will be used to evaluate the performance of the best one using bootstrapping.

The classifiers will be tuned using `Optuna`, a hyperparameter optimization framework, while the features will be selected using `mRMR`, a feature selection method which minimizes the redundancy and maximizes the relevance of the features. The classifiers will be evaluated using several metrics from the `scikit-learn` library.

The classifiers used in this assignment are:

- Logistic Regression (LR) with Elastic Net regularization
- Gaussian Naive Bayes (GNB)
- Linear Discriminant Analysis (LDA)
- Support Vector Machine (SVM)
- Random Forest (RF)
- LightGBM (LGBM)

The selected features were:
• `concave_points_worst` • `perimeter_worst` • `concave_points_mean` • `radius_worst`
• `perimeter_mean` • `area_worst` • `concavity_mean` • `radius_mean` • `concavity_worst`
• `area_mean`.

All models performed exceptionally well, with the best one being the **LDA** classifier. The optimal hyperparameters for the LDA classifier were:
• `boosting_type=dart` • `num_leaves=32` • `max_depth=53` • `learning_rate=0.403`
• `n_estimators=903` • `min_child_sample=20` • `reg_alpha=0.149` • `reg_lambda=0.542`
• `bagging_freq=9` • `bagging_fraction=0.455` • `feature_fraction=0.589`.

2 Introduction

The features are extractred from a fine needle aspirate (FNA) of breast masses and they consist of the following 10 properties, each of which is described by is mean, standard error, and worst value (meaning the largest mean of the three largest values), leading to 30 features in total:
• `radius` • `texture` • `perimeter` • `area` • `smoothness` • `compactness` • `concavity`
• `concave_points` • `symmetry` • `fractal_dimension`.

2.1 Preprocessing

The dataset needed to be preprocessed before being used. Afer inspection, it was found that the dataset contained some missing values, which needed to be imputed from the rest of the data. Also, some column names contained spaces, which needed to be replaced with underscores.

I decided to use the median of each column to impute the missing values, while respecting the class labels, so that the imputation is done separately for each class. Also, for creating the holdout set, I chose only from the samples that had no missing values, so that the holdout set is not affected by the imputation process.

2.2 Data exploration

A good practice before training a model is to explore the data and see if there are any patterns or correlations between the features and the target variable. Below are some plots that show the distribution of some features, for each class of the target variable, and collectively (figures for all feature distrivutions can be found in the `data_exploration.ipynb`) notebook.

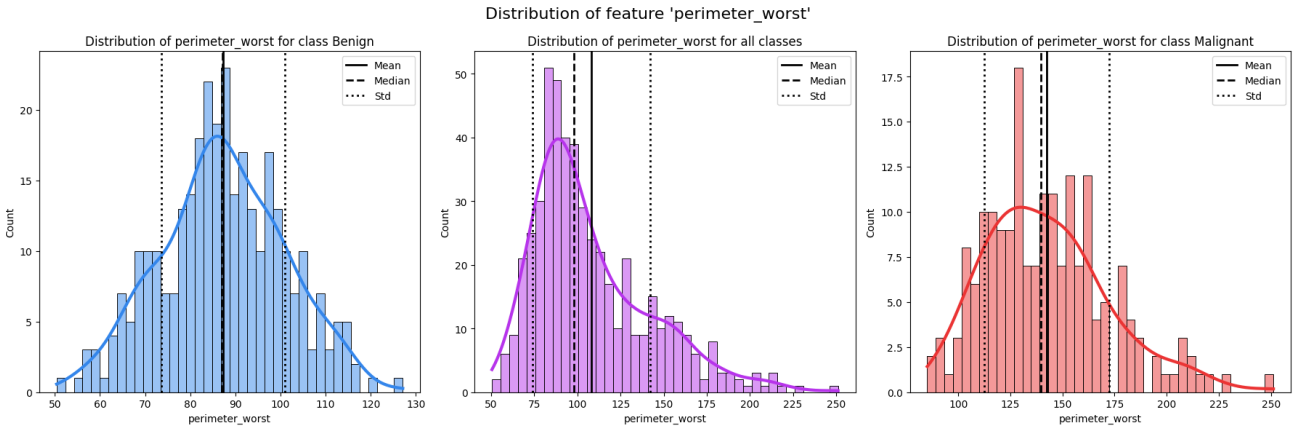

Figure 1: Distribution of the `perimeter_worst` feature for each class (left and right) and the whole dataset (center).

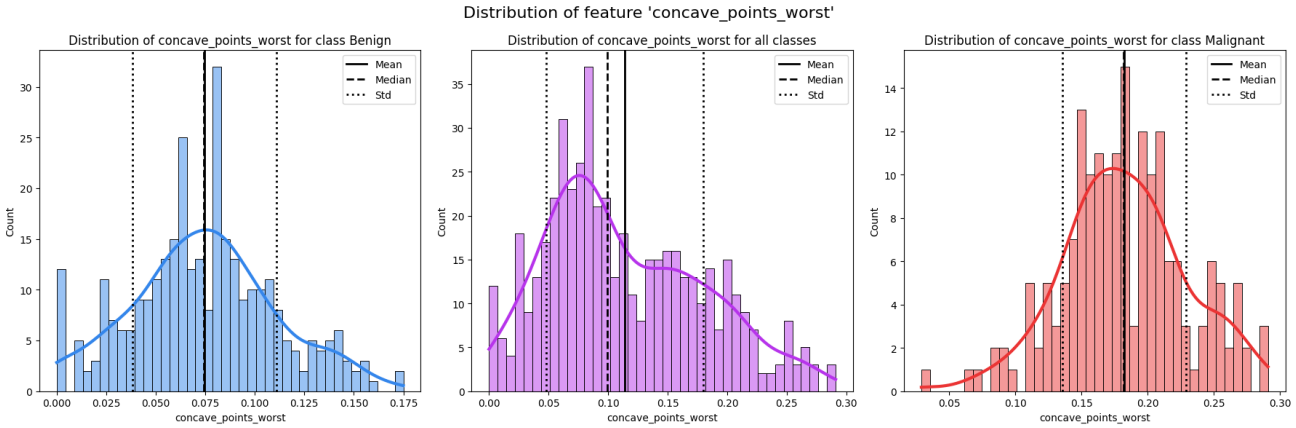

Figure 2: Distribution of the `concave_point_worst` feature for each class (left and right) and the whole dataset (center).

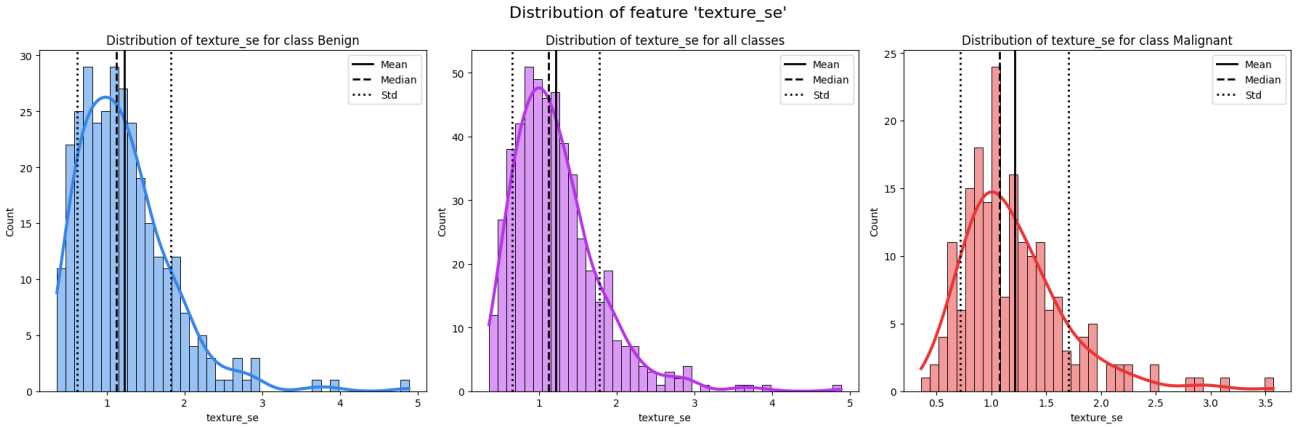

Figure 3: Distribution of the `texture_se` feature for each class (left and right) and the whole dataset (center).

In Figure ?? and Figure ??, we can see that the two classes are well separated, with the malignant class having higher values for both features. This is not the case in Figure ??; the `concave_points_se` feature has very similar distributions for both classes.

Next, we can inspect how correlated the features are with the target values by calculating and plotting the Spearman's ρ, Kendall's τ and the Point Biserial correlation coefficients. Spearman's ρ and Kendalls's τ are useful for measuring non-linear, monotonically increasing relations and the Point Biserial correlation is specifically designed for binary classification problems.

In Figure ?? we can see the absolute values of the correlation coefficients mentioned above. Eleven values show an absolute correlation of above 0.6, while only seven being below 0.3, indicating that the dataset has features that can describe the target value with high accuracy.

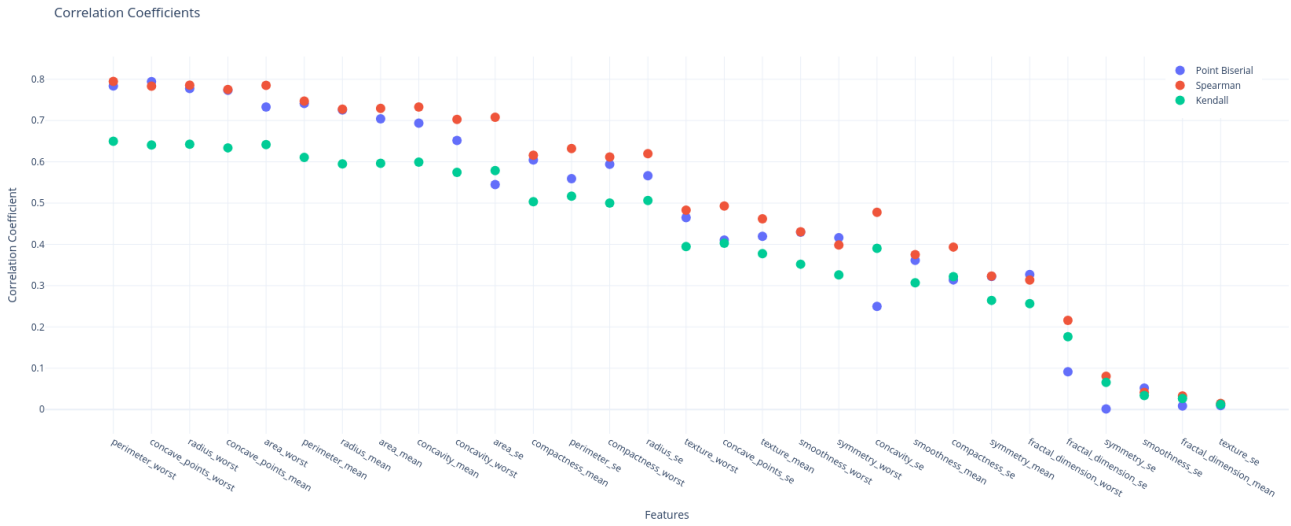

Figure 4: Distribution of the Spearman's ρ (red), Kendall's τ (green), and Point Biserial (blue) correlation coefficients of all features with the target value.

Measuring and plotting the inter-feature correlation can help us find redundant features. In Figure?? we can see a few examples of features that seem to carry the same information; these groups of features appear as spots of high correlation. For example the features regarding the perimeter and radius show high intra-feature correlation, as well as high correlation with the features regarding the area. Since I have ordered the features alphabetically, we expect correlations along the main diagonal to have high correlation, as they describe the are about features that describe the same property, and we see that in two main "blocks".

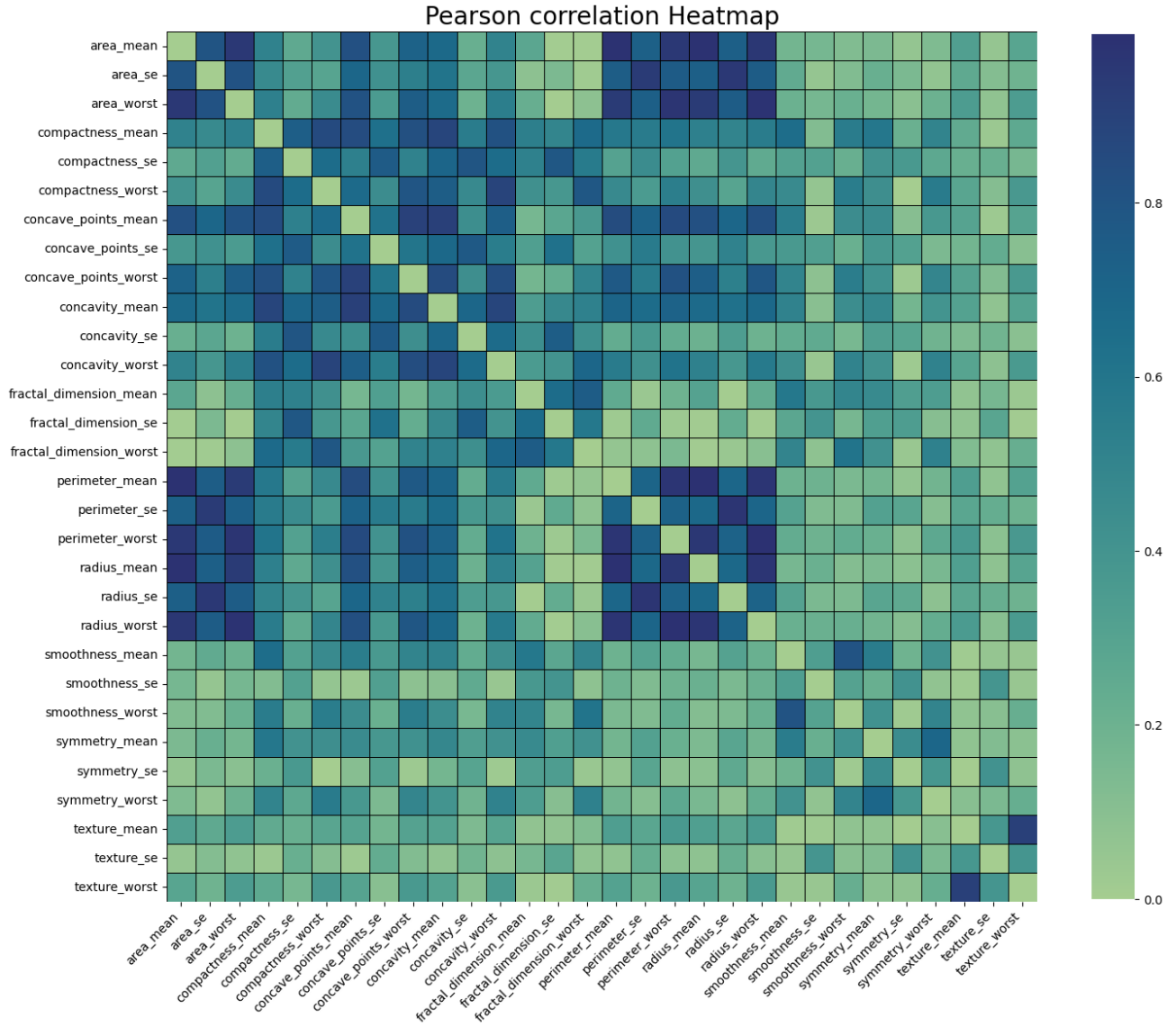

Figure 5: Distribution of the Spearman's ρ (red), Kendall's τ (green), and Point Biserial (blue) correlation coefficients of all features with the target value.

Overall, the dataset seems to be well separated, with some features being redundant. I expect that the classifiers will perform well, but I will need to use feature selection to remove the redundant features and improve the performance of the classifiers.

Latsly, we can use a dimensionality reduction technique to inspect if the projections of the data to a subspace show any meaningful separation or if they form any discernible structure. To that end I used `PCA`, `t-SNE`, and `UMAP` to map thje data points onto a lower dimension and visualize them. Below, I provide the plots for `PCA` and `UMAP`, the `t-SNE` results are very similar to the results from `UAMP` and they can be found in the `data_exploration.ipynb` notebook, I will not include them here for brevity.

In Figure **??** we can see that for the combination of the first and second principal components, the data set can be somewhat separated, with the benign class forming a more compact group and the malignant class forming a more diffuse group, but there is no physical separation between them. A similar case is true for the combination of the rest of the components. On the other hand, in Figure **??**, we can see that the data points are separated more clearly, forming a single elongated cluster of points which can be separated almost exactly at it's center, separating the two classes (in the cases of the combinations of first and second dimension and first and third). Furthermore, even from the first dimension, the data seem to from a binomial distribution,

separating the two classes.

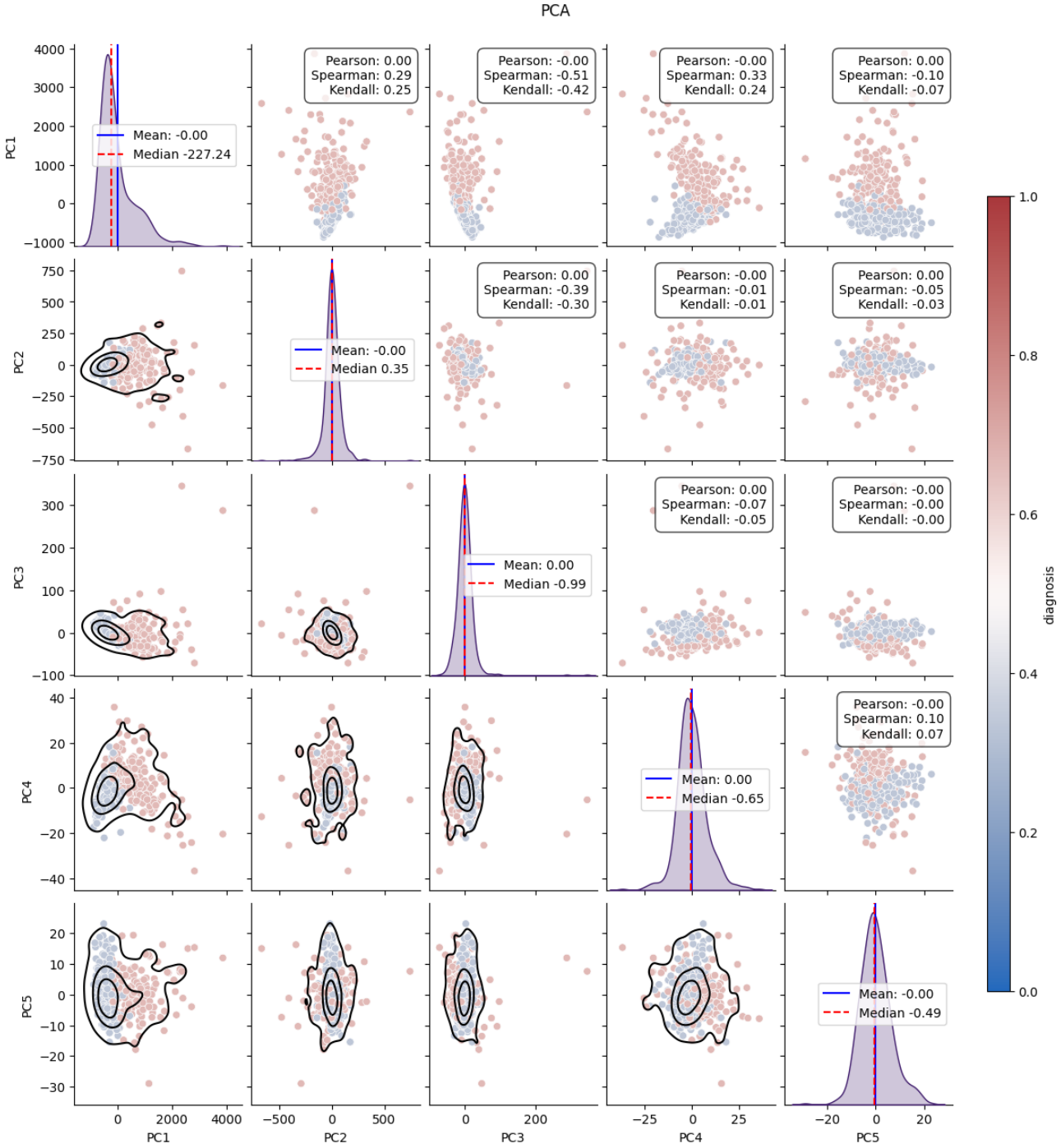

Figure 6: Pairplot of the first 5 prinicpal components after dimensionality reduction with `PCA`. The scatter plots are colored by the target's label (blue for benign and red for malignant).

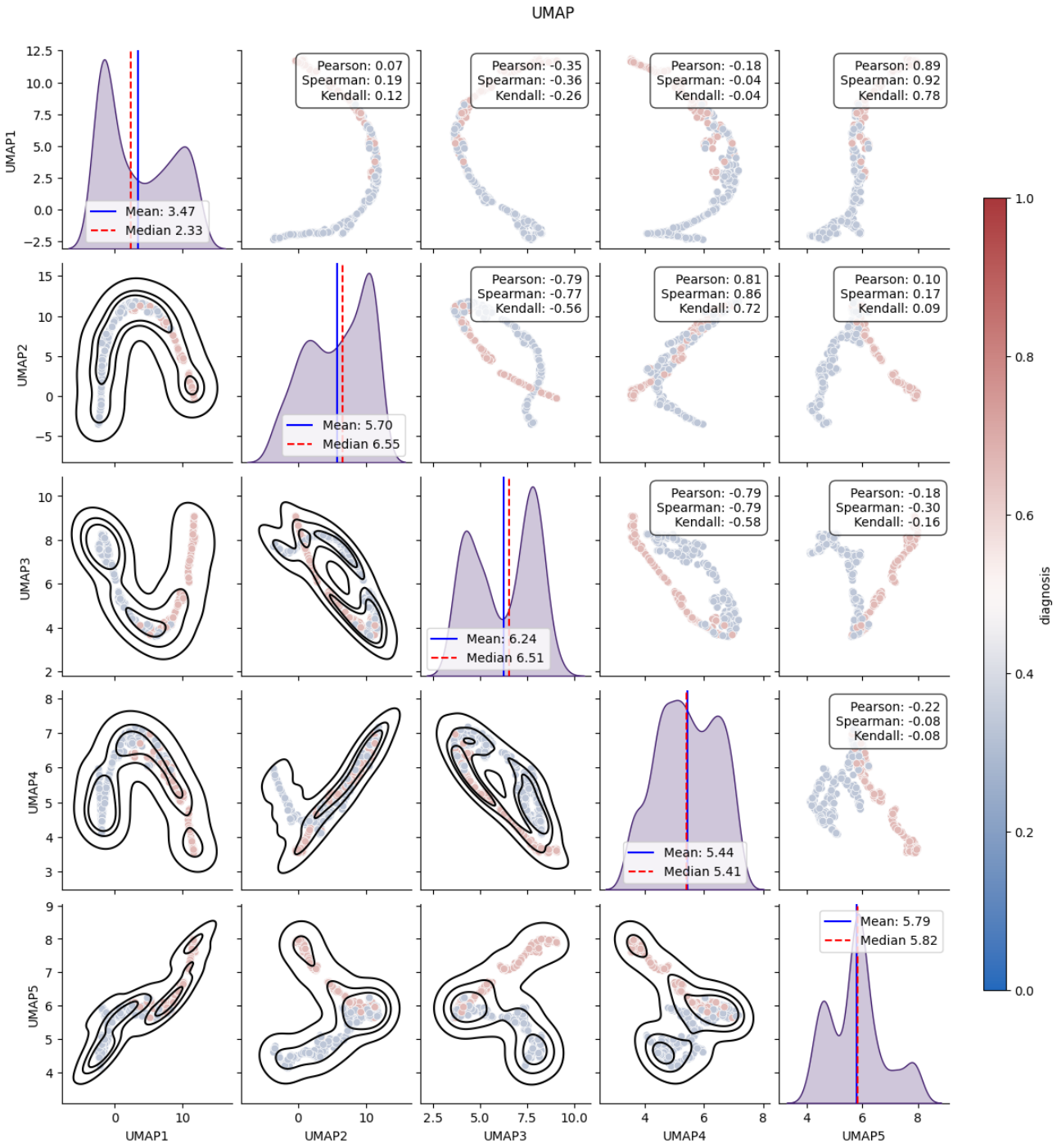

Figure 7: Pairplot of the first 5 embeddings / dimensions after dimensionality reduction with `UMAP`. The scatter plots are colored by the target's label (blue for benign and red for malignant).

With all the above in mind, I decided to set up the `mRMR` function to select 10 features, as I have shown that

3 Material and Methods

3.1 Outline

In essence, a nested cross-validation pipeline consists of two nested loops (outer and inner loops), encapsulated in a larger loop. In our case, the encapsulating loops (or rounds) will be set to $R = 10$, the outer loops will be set to $N = 5$, and the inner loops to $K = 3$.

The job of the outer fold is to split the data set and perform cross-validation, with the inner folds performing hyperparameter tuning with yet another cross-validation. The job of the enclosing rounds is to change the random number generator seeds, to get R different results, leading to better generalization ability of the model. A schematic of nested cross-validation is provided below (Figure**??**).

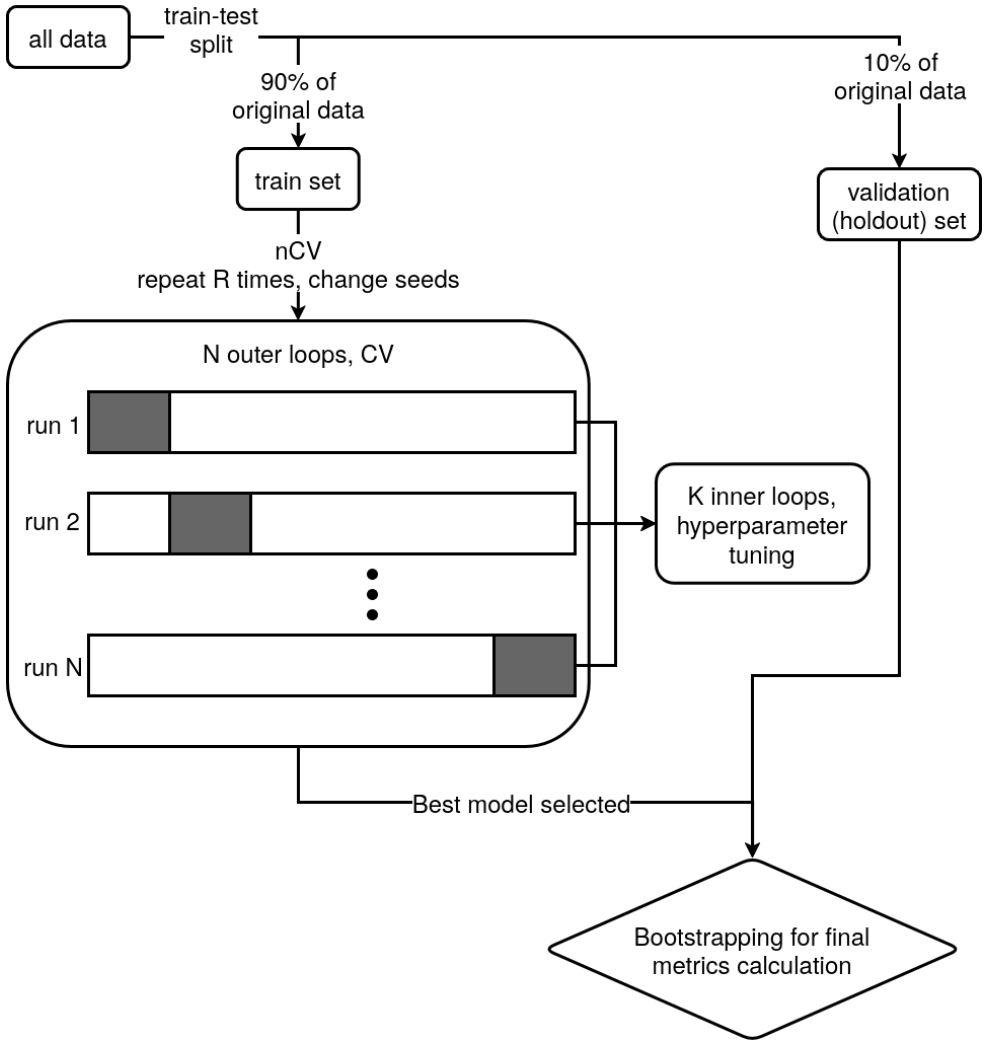

Figure 8: Nested cross-validation scheme. A holdout set is first taken from the dataset, here set to 10%. Then, R times, the model is trained using cross-validation (N splits), with an inner loop of hyperparameter tuning, which is also done using cross-validation (K splits). All training results are aggregated and the best set of hyperparameters is selected. After all models are trained, the best one can be tested against the unseen data (holdout set); here metric statistics are calculated using resampling with bootstrapping.

The implementation of a nested cross-validation is easy, but it needs extra care to avoid data leakage. The data must be scaled inside the inner loop, after their final split, otherwise the training set will carry some information of the test set (and as always the scaler must be fitted only on the test set). I decided to perform feature selection using the whole training set, ignoring the fact that it might "leak" some information to the rest of the pipeline, as I believe it will be minimal.

4 Results and Discussion

4.1 Feature Selection

5 Conclusion