Machine Learning in Computational Biology
Assignment 2

Konstantinos Konstantinidis
Student number: 7115152400017

May 7, 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 between features and maximizes their relevance to the target class. 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 images from a fine needle aspirate (FNA) [1] of breast masses and they consist of the following 10 **properties**, each of which is described by its 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.

The dataset was also imbalanced, with the benign class having 321 samples and the malignant class having 191 samples. I decided to leave the dataset as is, since the difference is not that big and the classifiers should be able to handle it. Also the ratio between the two classes seems to reflect the real world distribution of breast cancer cases, with the benign cases being more common than the malignant cases [2, 3].

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.

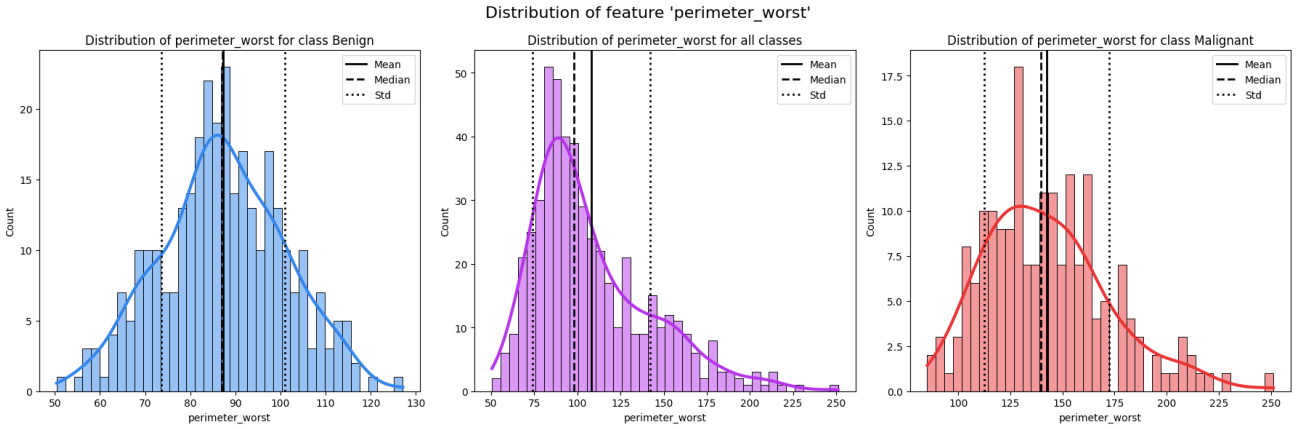

Figure 1: Distribution of the `perimeter_worst` feature for each class (left and right) and the whole dataset (center).

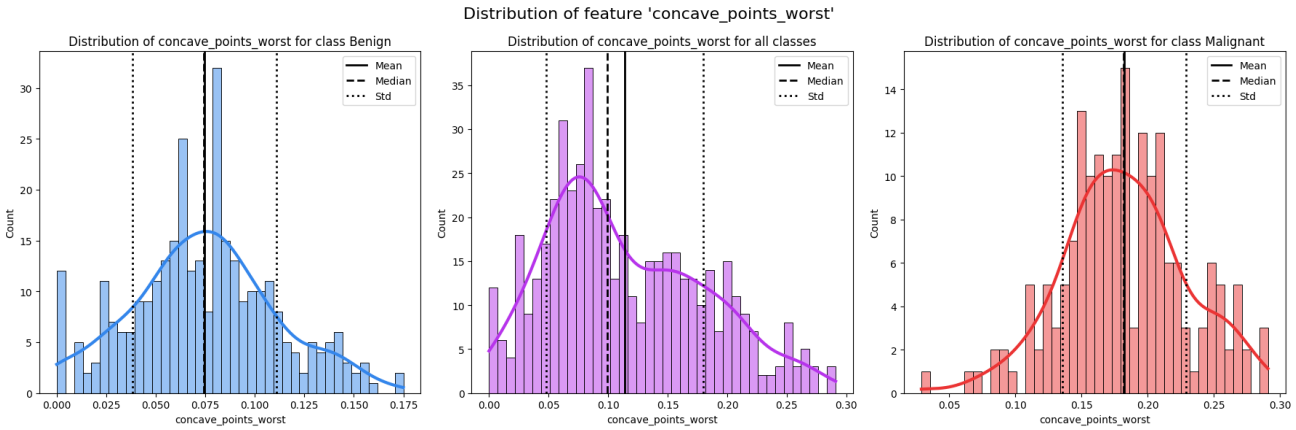

Figure 2: Distribution of the `concave_point_worst` feature for each class (left and right) and the whole dataset (center).

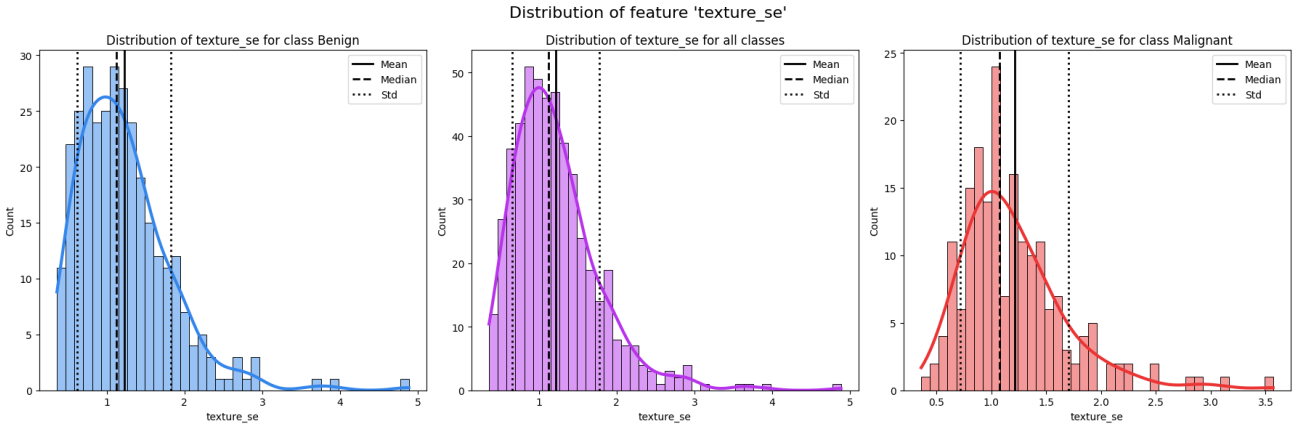

Figure 3: Distribution of the `texture_se` feature for each class (left and right) and the whole dataset (center).

In Figure 1 and Figure 2, 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 3; 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 4 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.

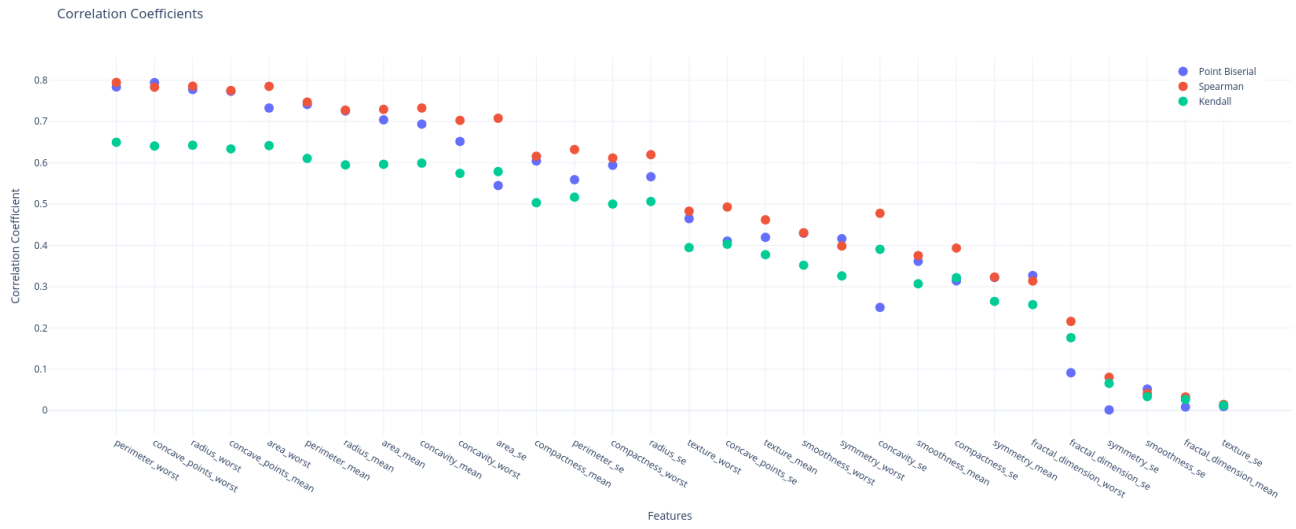

Figure 4: Distribution of the Spearman's ρ (red), Kendall's τ (green), and Point Biserial (blue) correlation coefficients of all features with the target value.

Measuring and plotting the inter-feature correlation can help us find redundant features. In Figure5 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".

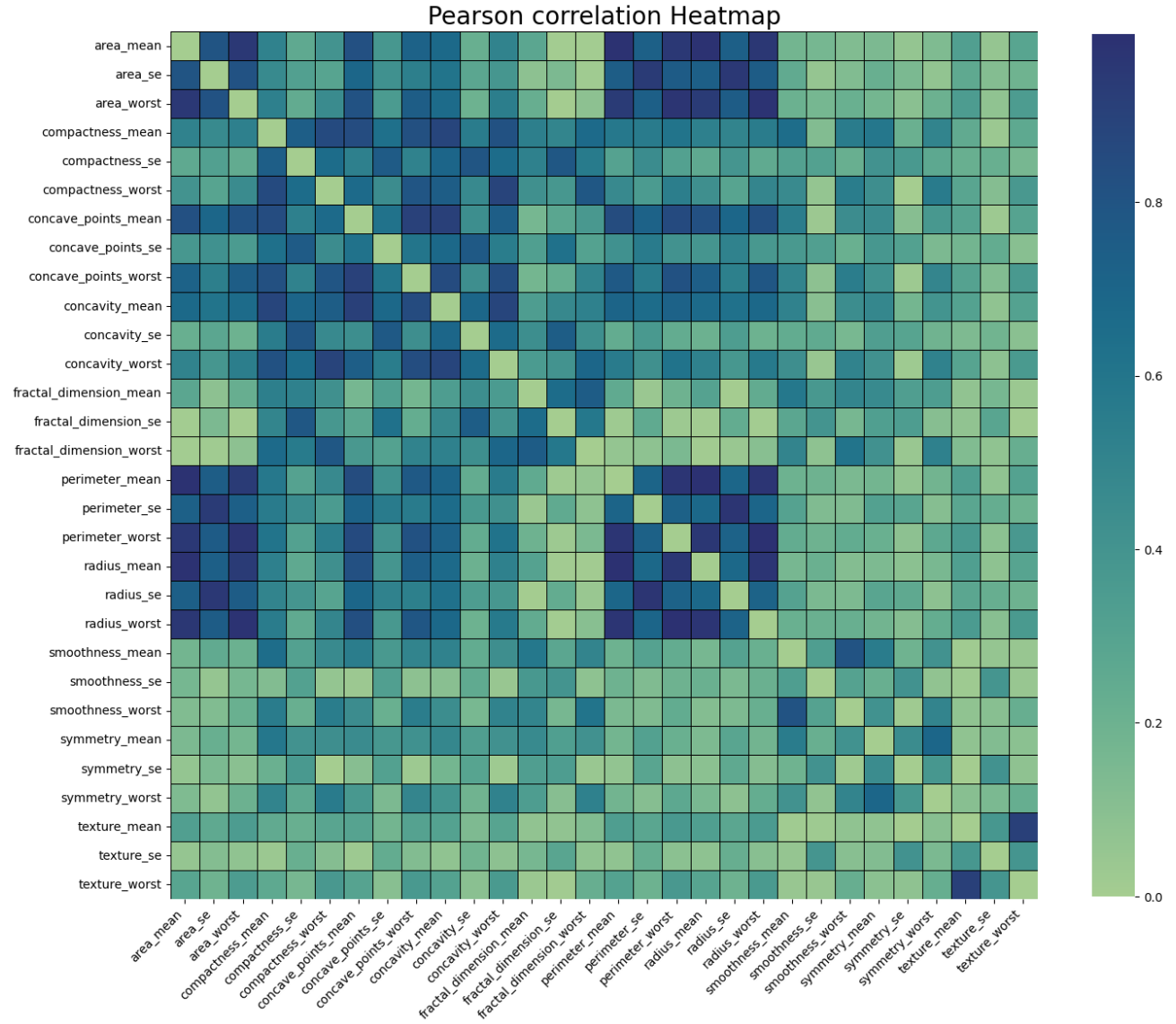

Figure 5: Distribution of the Spearman's ρ (red), Kendall's τ (green), and Point Biserial (blue) correlation coefficients of all features with the target value.

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 6 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 7, 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.

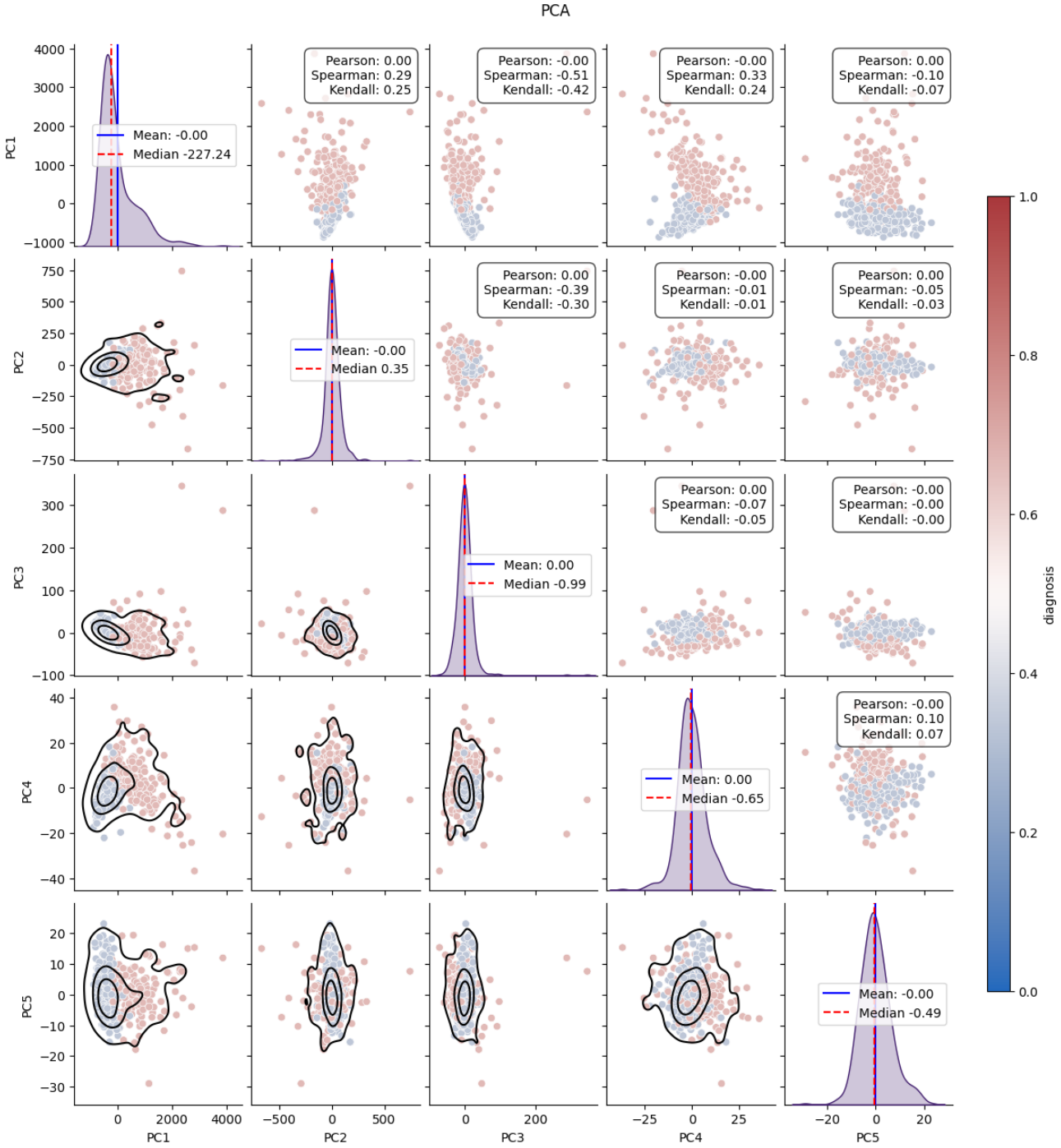

Figure 6: Pairplot of the first 5 prinicpal components after dimensionality reduction with `PCA`. The scatter plots are colored by the target's label (blue for benign and red for malignant).

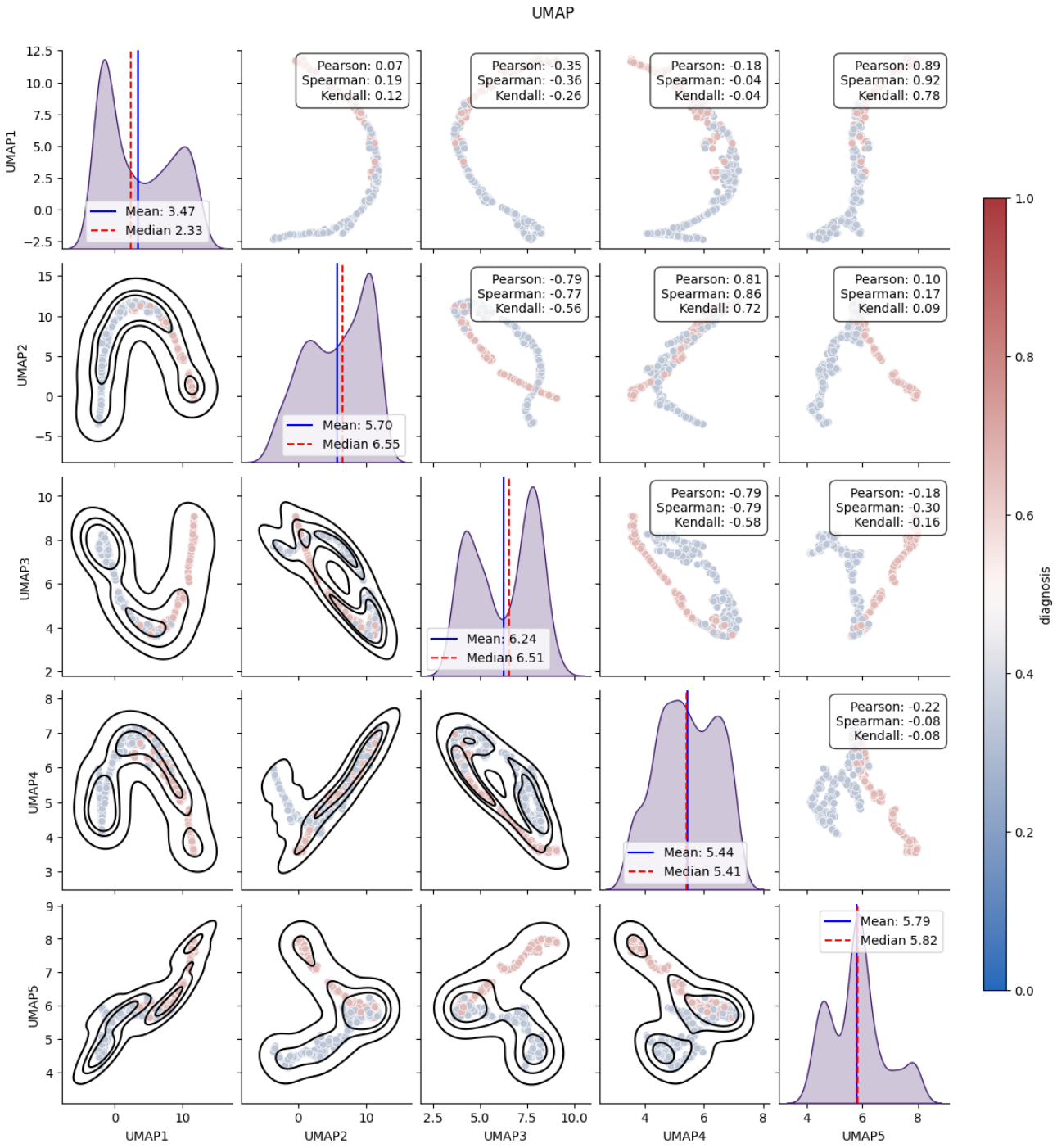

Figure 7: Pairplot of the first 5 embeddings / dimensions after dimensionality reduction with `UMAP`. The scatter plots are colored by the target's label (blue for benign and red for malignant).

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 (Figure8).

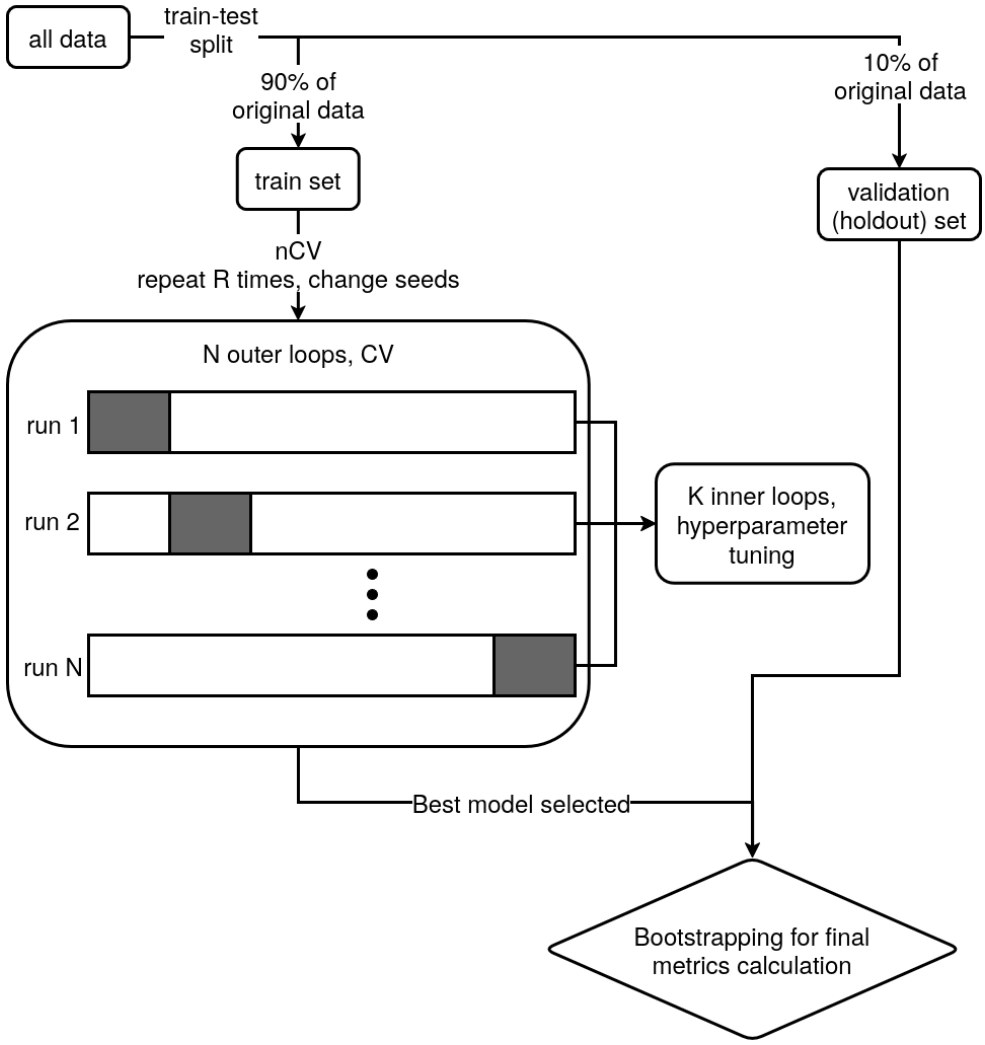

Figure 8: Nested cross-validation scheme. A holdout set is first taken from the dataset, here set to 10%. Then, R times, the model is trained using cross-validation (N splits), with an inner loop of hyperparameter tuning, which is also done using cross-validation (K splits). All training results are 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) [4]. 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 and doing otherwise would have been "overkill" in tis case.

3.2 feature selection

Feature selection was performed with the `mRMR` [5] method which aims to minimize the redundancy between features and maximize their relevance to the target value.

3.3 hyperparameter tuning

The hyperparameter tuning was performed inside the inner loop of the nested cross-validation. The hyperparameters were tuned using the `optimize` method of the `Optuna` library with a `TPE` sampler, which fits one Gaussian mixture model $l(x)$ to the set of parameters associated with the best objective values and another $g(x)$ to the remaining parameters and tries to maximize the ratio of the two, $\frac{l(x)}{g(x)}$. The objective function uses the Matthews Correlation Coefficient (MCC) [6] as the metric to evaluate the performance of the model.

3.4 model selection — best model final training

After the nested cross-validation is complete, the best model is selected manually based on the over-all statistics. The selected model is fit on the complete dataset (minus the holdout set) using the features and hyperparameters selected and tested against the holdout set.

4 results and discussion

The complete process for $R = 10$, $N = 5$, and $K = 3$, and for 1000 `Optuna` trials, for all classifiers, took around one hour, on a machine with running Ubuntu 24, with 16 GB of RAM, an AMD Ryzen 7 5800H processor, and an NVIDIA GeForce RTX 3050 Ti graphics card. All classifiers trained at relatively similar speeds, with the exception of RF and LGBM, which took significantly more time than the rest.

The complete process can be viewd in the `classification.ipynb` notebook, which also contains all outputs.

4.1 features

The selected features are shown below (Table 1).

Selected Features	
concave_points_worst	perimeter_mean
perimeter_worst	area_worst
concave_points_mean	concavity_mean
radius_worst	radius_mean
concavity_worst	area_mean

Table 1: List of selected features

These features agree with the data shown in Frigure 4 which ranks them as the features with the highest correlation out of all 30 in the dataset and from Firugre 5 we can see that although some of them are correlated, the correlations between them are not that high.

4.2 hyperparameters

The hyperparameters chosen by `Optuna` are provided in the Table below (Table 2).

Model	Parameter	Value
LogisticRegression	C	43.7051
	l1_ratio	0.0536
	max_iter	1758
	class_weight	None
GaussianNB	var_smoothing	1.94×10^{-10}
LinearDiscriminantAnalysis	solver	lsqr
	shrinkage	auto
	priors	None
SVC	C	0.8969
	gamma	auto
	coef0	0.0822
	kernel	linear
	class_weight	None
RandomForestClassifier	n_estimators	32
	criterion	gini
	min_samples_split	12
LGBMClassifier	boosting_type	dart
	num_leaves	32
	max_depth	53
	learning_rate	0.4030
	n_estimators	903
	min_child_samples	20
	reg_alpha	0.1486
	reg_lambda	0.5418
	bagging_freq	9
	bagging_fraction	0.4555
	feature_fraction	0.5883

Table 2: Best hyperparameters for each classifier after tuning. Numerical values rounded for readability.

4.3 training results

To capture the performance of the models the following metrics were used:

`balanced accuracy` • `accuracy` • `precision` • `recall` • `specificity` • `MCC` • `F1` • `ROC AUC`

Below I include the boxplots for the Matthews correlation coefficient, recall, specificity, and balanced accuracy metrics that the models achieved during their training (Figures 9, 10, 11, 12), the rest are not included for brevity but they can be found in `classification.ipynb`.

From these metrics we can see that LDA scored the best in all but one area, recall, which was expected due to the recall/specificity trade-off, as recall captures the proportion of actual positives correctly identified, while specificity captures the proportion of actual negatives correctly identified.

$$recall = \frac{TP}{TP + FN}$$

$$specificity = \frac{TN}{TN + FP}$$

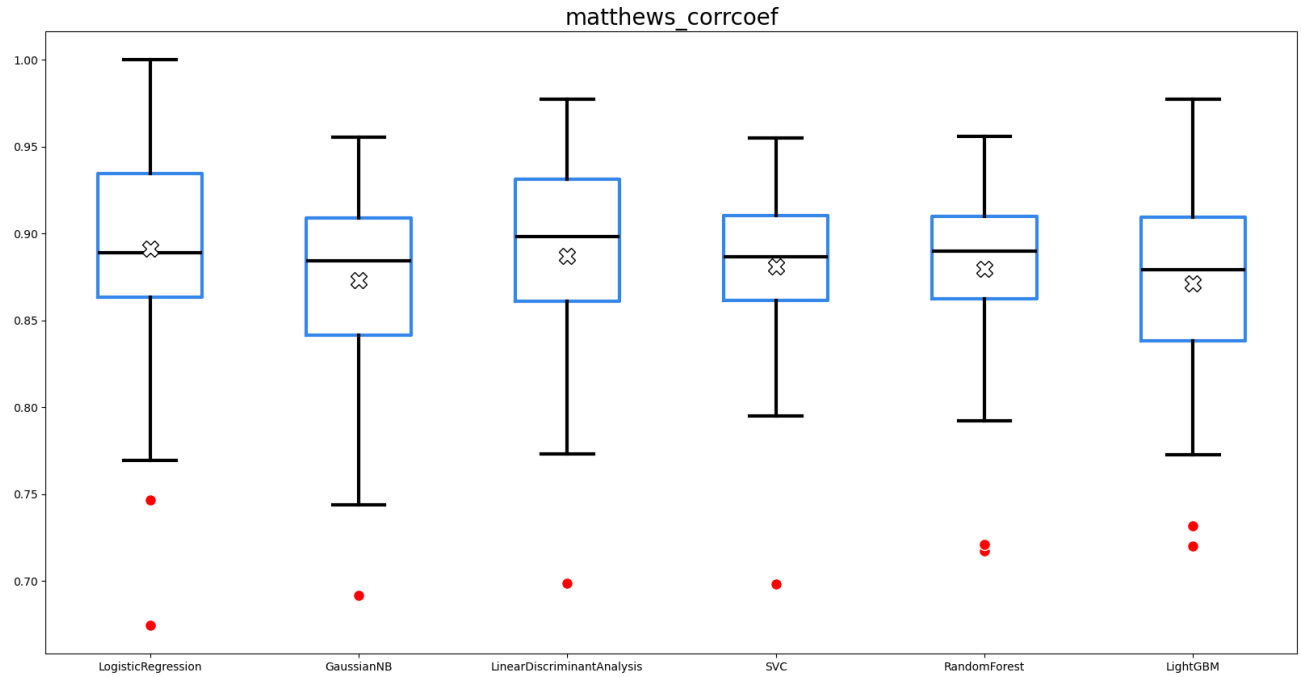

Figure 9: Matthews correlation coefficient for all models.

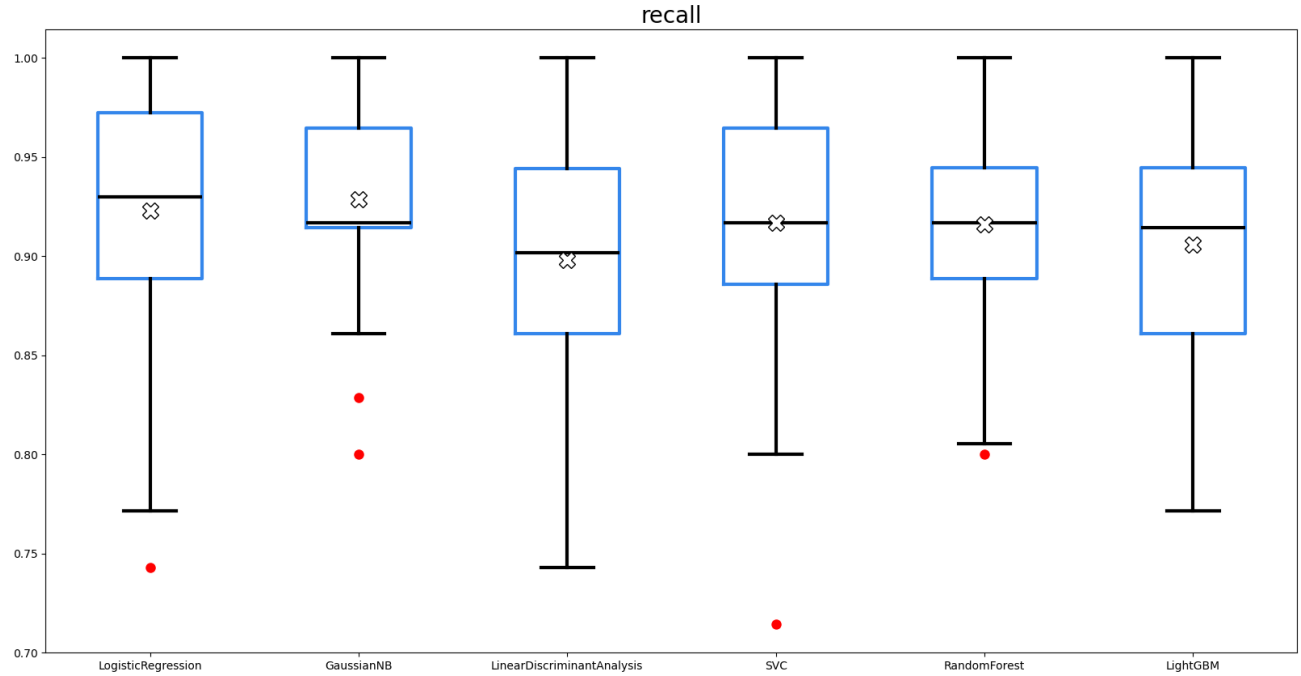

Figure 10: Recall for all models.

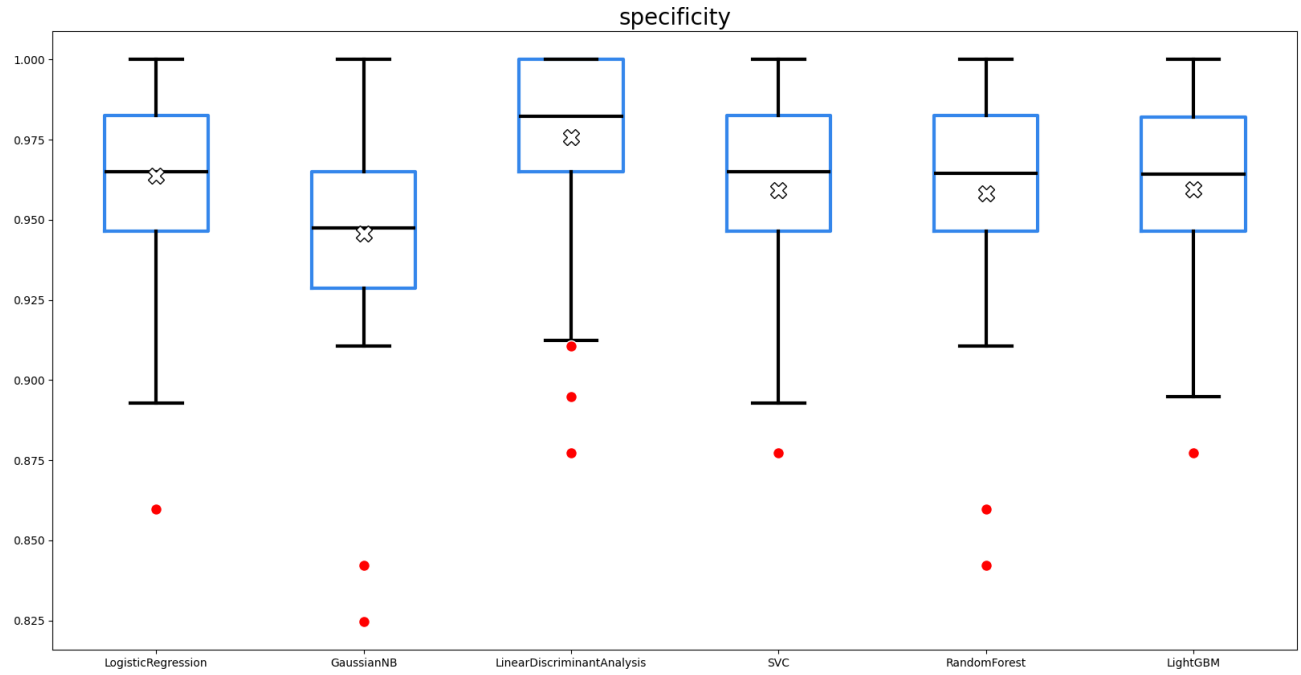

Figure 11: Specificity for all models.

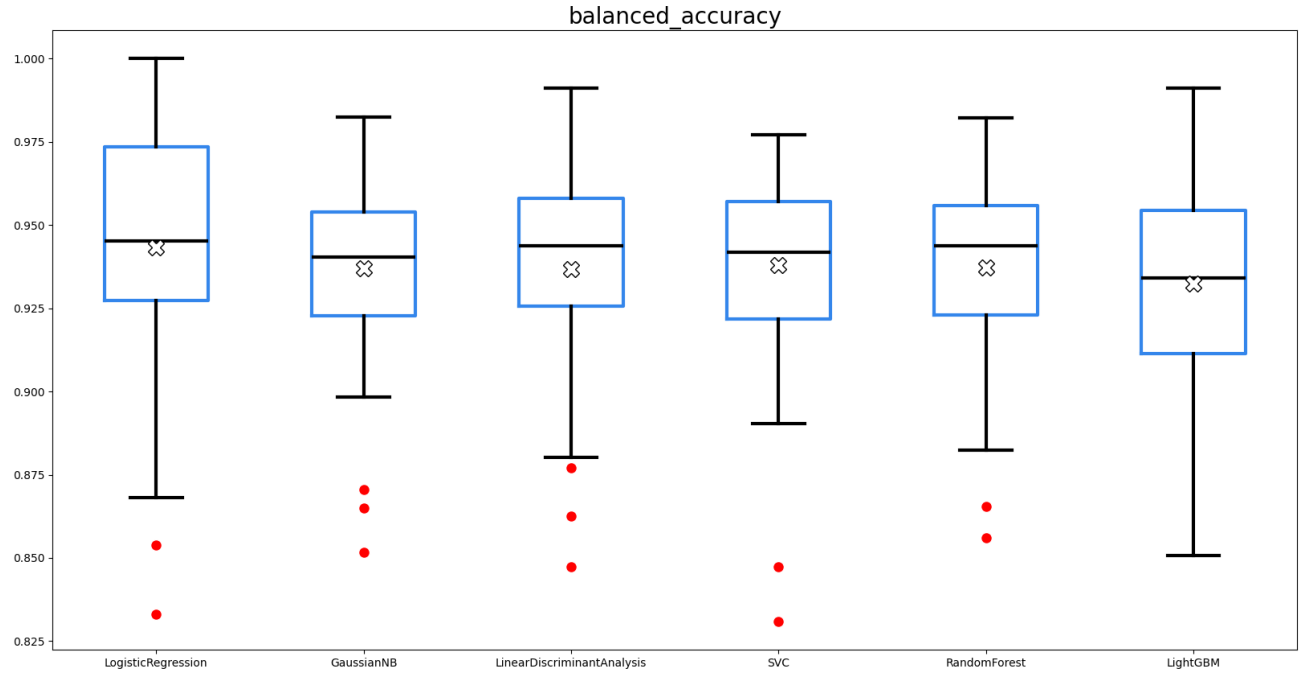

Figure 12: Balanced accuracy for all models.

4.4 best model evaluation

The best model was then tested against the holdout set, using the same metrics as above and 1000 rounds of resampling with bootstrapping to generate enough samples to calculate statistics. The results are shown in Figure 13.

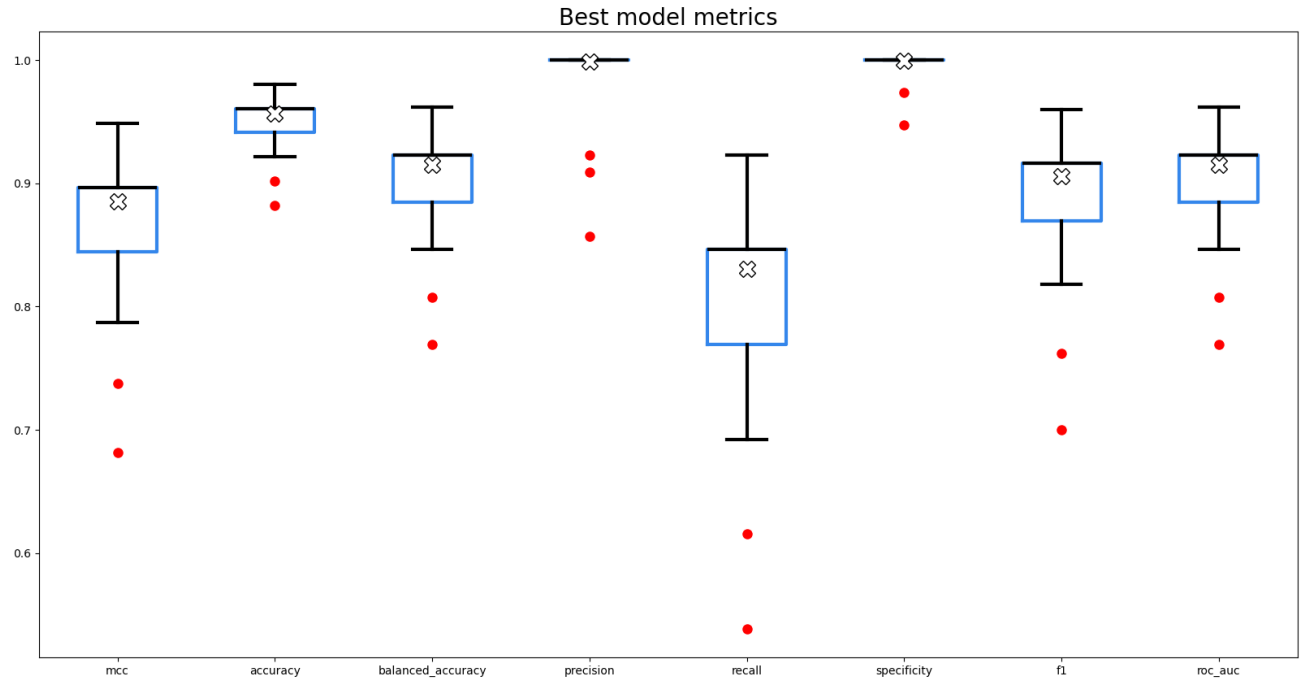

Figure 13: Best model evaluation metrics.

5 conclusion

References

[1] 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, no. 1, p. 41, 2012.

[2] E. Ugiagbe and A. Olu-Eddo, "Benign breast lesions in an african population: A 25-year histopathological review of 1864 cases," *Nigerian Medical Journal*, vol. 52, p. 211, Jan. 2011.

[3] P. S. Bhathal, R. W. Brown, G. C. Lesueur, and I. S. Russell, "Frequency of benign and malignant breast lesions in 207 consecutive autopsies in australian women," *British Journal of Cancer*, vol. 51, pp. 271–278, Feb. 1985.

[4] S. Bates, T. Hastie, and R. Tibshirani, "Cross-validation: What does it estimate and how well does it do it?," *Journal of the American Statistical Association*, vol. 119, pp. 1434–1445, May 2023.

[5] C. DING and H. PENG, "Minimum redundancy feature selection from microarray gene expression data," *Journal of Bioinformatics and Computational Biology*, vol. 03, no. 02, pp. 185–205, 2005.

[6] D. Chicco and G. Jurman, "The matthews correlation coefficient (mcc) should replace the roc auc as the standard metric for assessing binary classification," *BioData Mining*, vol. 16, no. 1, p. 4, 2023.