Machine Learning Assignment 2

Rafail Adam

7115152400009

Github :

https://github.com/RafailTn/Assignment-2

Abstract

Breast cancer is the most frequent malignancy for the female population worldwide, with patients making up to 36% of oncological patients[1]. Many machine learning algorithms have been employed to aid the prognosis of breast cancer tissues[2]. Here we examine a dataset originating from measurements from fine needle aspirates and employ a variety of machine learning algorithms to discover which performs best in predicting malignancies among benign and malignant cases.

Introduction

Breast cancer is the most common type of malignant tumor in the world with an estimated 2.089 million women being diagnosed in 2018, while its incidence is increasing all regions of the world[1]. The highest incidence is found in industrialized countries, which is associated with poor diet, nicotinism, excessive stress and little physical activity.

Additional risk factors exist as well including ,but not limited to:

- Older Age (increased risk from 35 years of age)
- Family History of breast cancer
- Infection with an oncogenic virus
- Early age of first menstruation
- Late age of last menstruation
- First reported pregnancy at late age
- No pregnancy
- Use of hormone replacement therapy

Fine needle aspiration is a technique used for drawing liquid or tissue samples from patients that have new growths or masses and are in need for diagnosis. If the area of reference is superficial and palpable, inserting the needle and gently aspirating will suffice. However, if the area is not palpable or superficial/visible the aid of ultrasound is needed. More specifically, a specialized scope with an ultrasound probe attached as well as a device on the tip of the scope to deploy the needle. In general, it is used when diagnosis of a mass is needed, but it can also be used for prognostic goals such as determining genetic or molecular markers that indicate the cancer is susceptible to specific chemotherapeutic or biologic treatments, while it is also used to treat the contents of an abscess in combination with antibiotics[3].

Machine learning approaches have been proposed that predict malignancies using various different features based on the method/experiment used in the study[2], [4], [5]. In this study, we are using a dataset that originates from measurements of fine needle aspirates and examine the predictive capability of its features using a variety of machine learning algorithms including Logistic Regression, SVM, Linear Discriminant Analysis, Gaussian Naïve Bayes, LightGBM and Random Forest Classifiers from the Library Atom, to predict benign and malignant samples.[6].

Results

Preprocessing

The dataset consists of 30 features that correspond to measurements taken from masses drawn from fine needle aspirates. More specifically these features coming from cell nuclei of those masses, are based on the radius (mean distance from center to perimeter points), texture (standard deviation of gray-scale values), perimeter, area, smoothness (local variation in radius lengths), compactness (calculated as perimeter^2/area - 1.0), concavity (severity of concave portions of the contour), concave points (number of concave portions of the contour), symmetry, and fractal dimension (a measure of complexity using coastline approximation). Each of these properties is characterized in three ways: mean value, standard error, and worst (largest mean of the three largest values).

The dataset was loaded from the csv file to a python notebook and basic statistics were printed for each column in order to examine the scale of the features as well as get a rough estimate of their distributions. Given that the features are of different scales, it was decided that a scaler will be used at some point throughout the preprocessing steps. Afterwards, checks for duplicate values as well as NA values were performed where it was found that NA values exist for many of the columns. Since the dataset is small, an imputing strategy is needed in order to fill in missing values. In order to ensure minimal loss of information, an imputer based on the K-nearest neighbors method was chosen from scikit learn.

After filling in missing values, exploratory plots were made in order to examine the class distribution, the feature space their correlation with each other and the target variable. The correlation between the features is shown in the triangular heatmap below. Given the nature of the features and the dataset, linear relationships between the features are expected (for example radius with area and perimeter). This plot will help us evaluate the quality of our data as well as the redundancy of the features in it and ideally decide on which features to drop depending on the final plot of mutual information between each feature and the target variable.

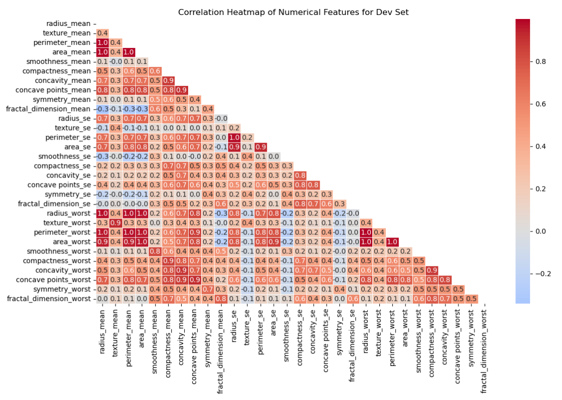

Figure 1. Feature correlation heatmap

Overall, there are no significant linear correlations between features that are not expected to be dependent on each other. Most features show mild correlation with other ones meaning that the deciding factor for dropping will be the feature selection method. The distribution of each feature in the dev set was also plotted to check visually differences between the two classes in each feature.

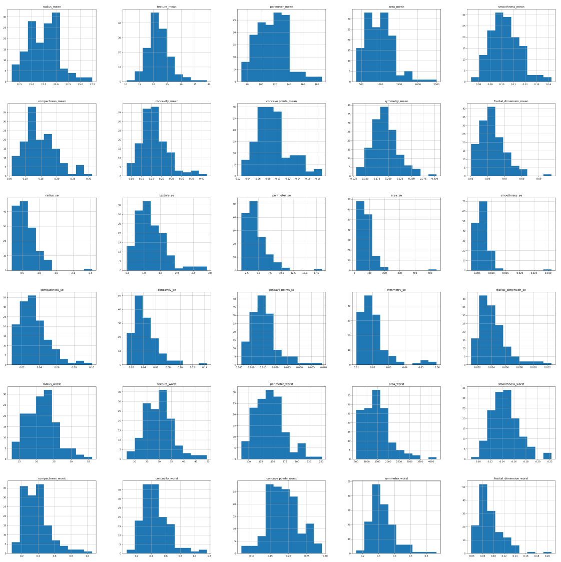

Figure 2. Distribution of each feature for Benign samples.

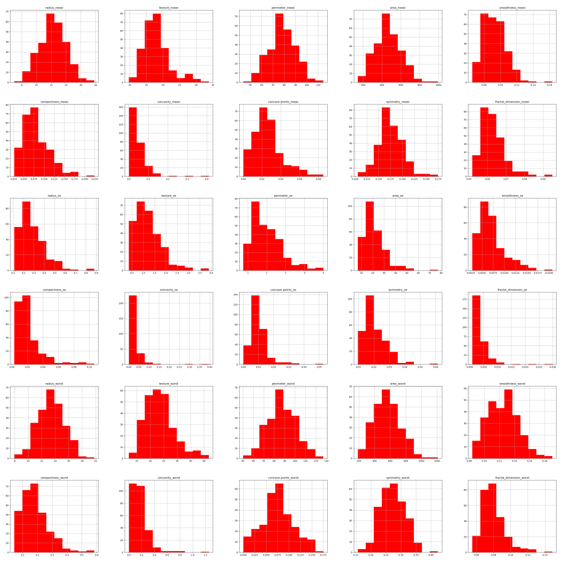
Figure 3. Distribution of each feature for Malignant samples.

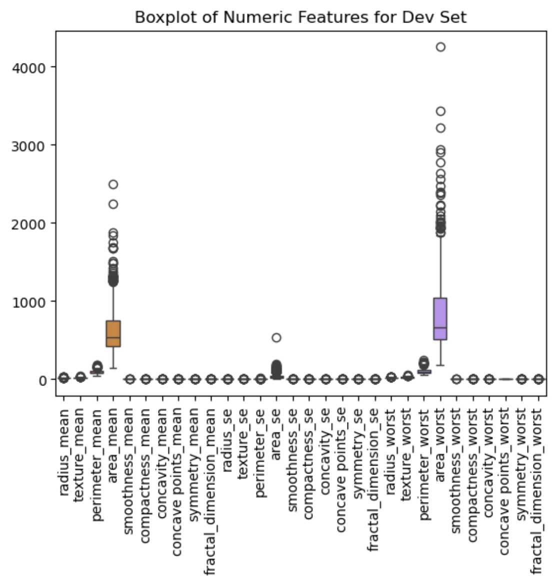

Figure 4. Boxplot of features in dev set.

In order to decide which scaling method to use, boxplots were made in order to further examine the distributions of the features and get a feeling for the number of outliers in them. Features that have higher scales tend to dominate the boxplots so many different boxplots were made sequentially to check for outliers in features of smaller scale. Here only the first boxplot that was made is shown for reference, in order to give a picture of the distributions. All of the plots except for the mutual information one were made prior to scaling in order to depict the true distribution of the features and uncover the correlations between them without any prior transformation. The correlation of each feature with the target variable was performed after scaling with the goal of seeing if the scaled features can still be used to separate the classes.

As shown in the plot, there is expected to be a significant number of outliers in the features, thus RobustScaler was utilized. The final sequence of transformations is thus the following:

1. KNN-Imputer
2. RobustScaler

Additionally, the mutual information metric of each feature with the target variable was plotted next in order to see which features are expected to be kept and be able to separate the data more efficiently. Naturally since the non-linear relationships between more important and less important features are not known, an optimal combination might not include all the features that show the highest mutual information values here, but many of them are expected to be in it.

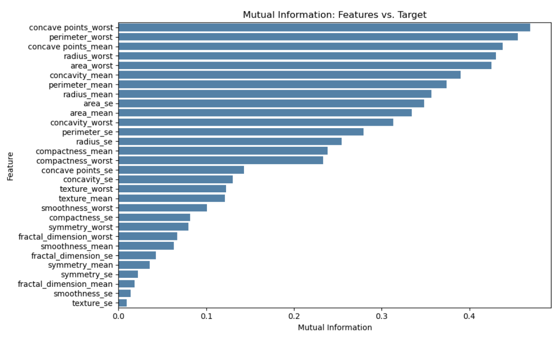

Figure 5. Barplot of mutual information of features relative to target.

Finally, PCA was performed on the feature set in order to find the number of Principal Components that explain 95% of the variance of the dataset. The total number of features is 30 and given the high linear correlations between some of them, PCA is expected to give a meaningful combination of Principal Components without sacrificing much information. After PCA the number of components needed to explain 95% of the variation is 10, with the first components being able to explain the majority of it. The first 2 principal components that cumulatively explain over 65% of the variance are plotted to check the separability of the classes.

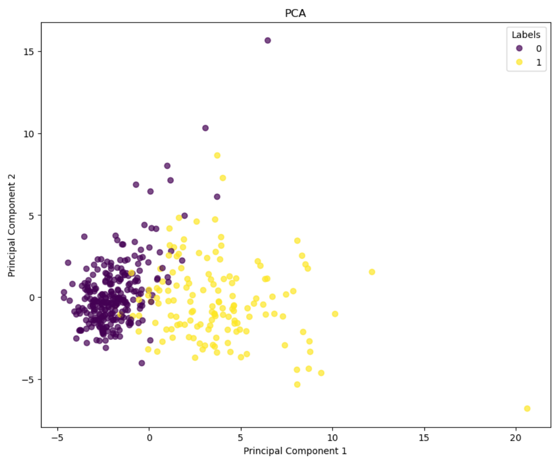

Figure 6. Scatterplot of samples based on the first 2 PCs

From the plot above it is apparent that PCA does a really good job in reducing the dimensionality for this dataset as the classes seem rather separable based on the first 2 Principal Components. Most of the overlap that is observed is expected to be taken care of by the 8 following Principal Components.

Model Development

After running the baseline models through the repeated nested kfold the model instance that had the best metrics in the inner Cross-Validation was the SVM model which was the best performing model across 22 of the 50 folds followed by the Logistic Regression model which performed better in 13 of the folds. The plots coming from the outer Cross-Validation help give a better picture regarding which model performs best since they are plotted as boxplots with means, medians and confidence intervals.

Figure 7. Baseline Outer-fold metrics

From these results we can surmise that the SVM instance seems to be the best performing model when comparing baselines in the inner fold, without feature selection. However, the outer fold results reveal that its performance is rather close, if not slightly worse, than the LR algorithm. This verdict is reached due to the fact that the median of the plots in LR are slightly higher in comparison. A final verdict for the baseline will be reached after the models are compared on their performance on the evaluation set (by using bootstrapping).

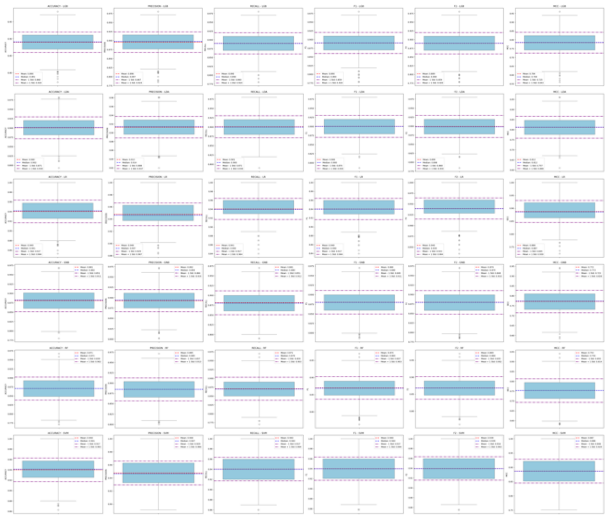
Figure 8. Bootstap evaluation for baseline models.

Based on the performance on the evaluation set, the model instance with the most consistent and generalizable results is the SVM considering it was also the model that won the inner cv voting with a significant difference. However, since performing PCA for dimensionality reduction is a necessity, all of the models' performances will be compared again after PCA due to the fact that the feature transformation performed during this technique might alter the performance of some models.

After running each algorithm through the inner Kfold, the model that won the voting across the 50 folds was the LR algorithm (16 folds), followed closely by the LGB algorithm (14 folds) with the SVM being in third place (13 folds). The results coming from the outer fold also changed significantly as shown below, giving new insight into which algorithm performs better given PCA dimensionality reduction.

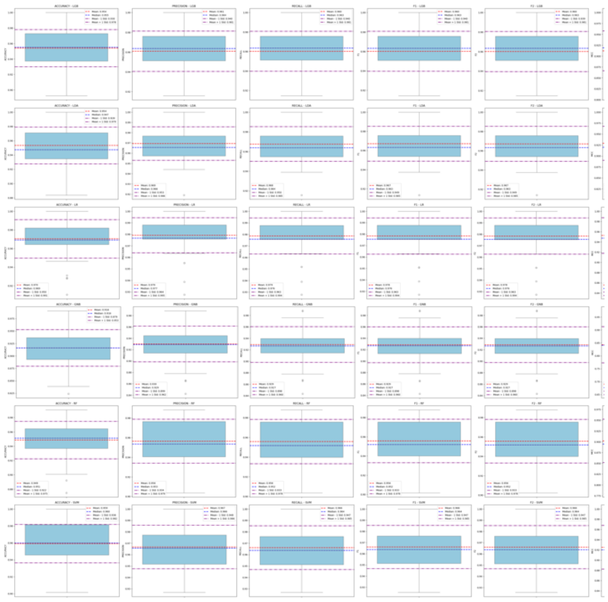
Figure 9. Baseline Outer-fold metrics after PCA.

Based on the boxplots above it can be surmised that the transformation had an important effect on the performance of the models, as many of the distributions have changed in not only values but regarding the range as well. The model that is judged to perform better in these circumstances is the LR model, as its distribution based on the outer fold in narrower and its median is higher than that of the other models. Once again, the final verdict will be reached after comparing evaluation set metrics as well.

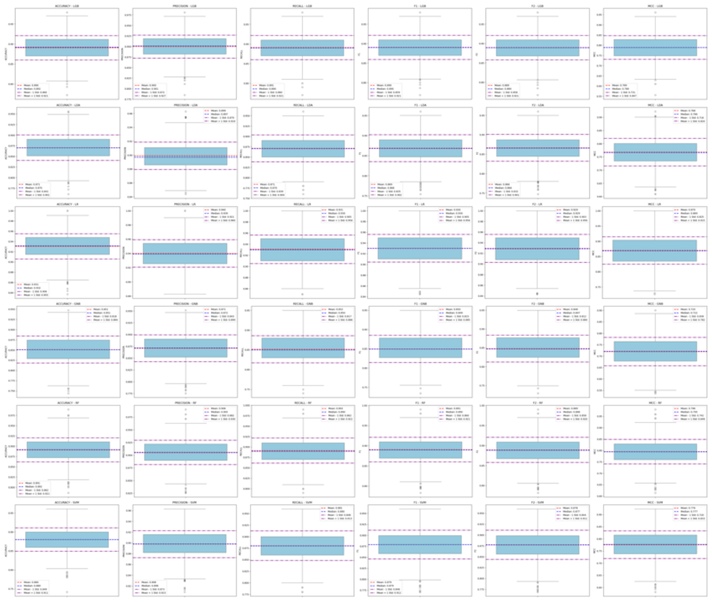
Figure 10. Bootstrap evaluation after PCA.

Based on the evaluation set performance of the models after PCA dimensionality reduction, the model instance that has the best results is the LR algorithm. Thus, this algorithm is chosen for further fine-tuning with optuna in order to further improve its performance. Due to the rnCV structure, a custom strategy is utilized for fine-tuning where the best optuna suggested models (based on the maximization of the fbeta score) coming from the inner loop are tested on the outer folds and resulting 50 combinations of the hyperparameters are plotted (for hyperparameters with continuous values) or counted (for hyperparameters with discrete values).

The metrics below correspond to the different models optuna suggested across folds and exist to make sure fine-tuning has a (positive) effect, while the histogram of the C value distribution is shown for reference. Naturally, the boxplots do not correspond to one model instance.

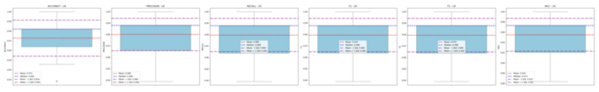
Figure 11. Boxplots of metrics for optuna suggested LR across folds.

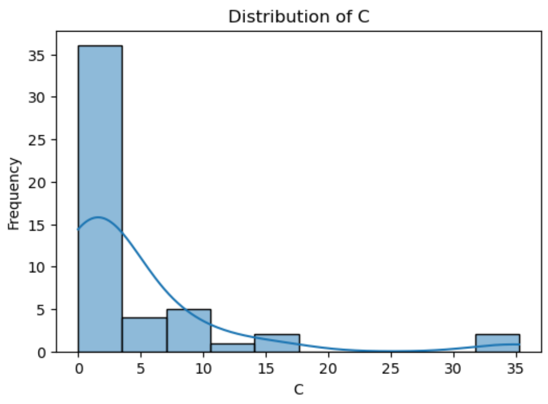

Figure 12. Histogram of suggested C value distribution.

The most frequent values for each hyperparameter are then chosen (taking into account how the model is structured and uses them) in order to create the most generalizable and high performing model instance. The best hyperparameters for the Logistic Regression model based on this workflow are the following:

- C = 1
- Penalty: L2
- Solver: liblinear

Compared to the default hyperparameters, the only change is the solver which is changed from lbfgs to liblinear. The results of fine-tuning on kfold metricsas well as the distributions of the hyperparameters across folds, can be found in the model_analysis notebook. Finally, the expected best model instance is run with the rnCV class to get the results based on the outer fold as well as bootstrap results on the evaluation set and compare them to the baseline models.

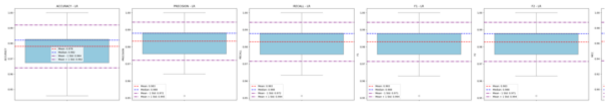
Figure 13. Fine-tuned LR Outer-fold metrics

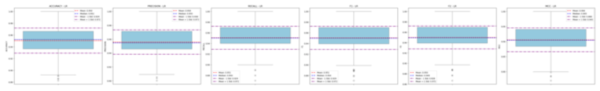
Figure 14. Fine-tuned LR bootstrap metrics

Based on the results of the outer cv, the fine-tuned LR model shows improved performance across all metrics compared to the baseline runs, increasing by almost 1%, with the exception of balanced accuracy where there is a slight drop in the median by about 0.7%. The bootstrap results show also significant improvement by as much as 2% across all metrics compared to the bootstrap metrics of previous runs showing that the change in the solver hyperparameter is significant for performance.

Methods

Due to the nature of the preprocessing steps, the functions/classes needed (ie for plotting or scaling) were called directly as they are only needed once and constitute only of a few lines of code. An object oriented approach was taken towards the model construction part of this project.

For this purpose, a class named Utils was written that contains general purpose functions that are to be utilized outside of this project as well. The functions include pipeline building and storing, plotting metrics and analyzing optuna results from kfold cross validation. Additionally, a class for repeated nested Cross Validation was built that adheres to the structure described in the requirements. This class includes a function for the afore mentioned rnCV, a function for bootstrapping on the evaluation set and one for finetuning the best model.

The rnCV structure constitutes of an inner Stratified Cross Validation of 3 folds that is used by Atom to judge the best model instance based on voting across folds. The outer CV consists of 5 folds and the whole procedure is repeated 10 times, creating 50 folds in total (since the results of the 3 inner ones are averaged by Atom by default). After the best model is decided based on the inner fold, the model is then tested on the test split from the corresponding outer fold and the metrics are kept to be plotted along with the rest of the repetitions. The metrics used for this project include:

- Balanced Accuracy
- Precision
- Recall
- F1 score
- Fbeta score (beta=2)
- Mathews Correlation Coefficient

The structure of the data flow is shown in the figure below:

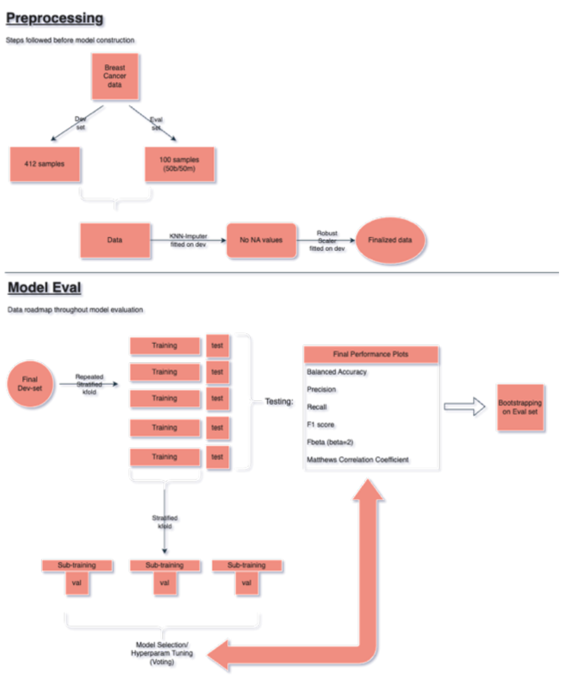

Figure 15. Data flow in the project

The models utilized and compared in the assignment are the following:

- Logistic Regression
- SVM
- Linear Discriminant Analysis
- Gaussian Naïve Bayes
- LightGBM
- Random Forest

LightGBM (Light Gradient Boosting Machine) builds an ensemble of decision trees using gradient boosting with various innovations. Firstly, it uses histogram-based bins for continuous values in order to speed up split finding. Secondly, it grows leaf-wise instead of level-wise by finding the leaf with the largest loss reduction which leads to deeper and fewer trees. Finally, it can bundle mutually exclusive features together without losing information[7].

The Random Forest classifier is a tree based model that uses decision trees and voting to perform classification. More specifically, it is trained on a random subset of the data drawn with replacement, at each split for one tree only a subset of features is considered and the final verdict is taken by the majority vote across all trees.[8]

The Logistic Regression algorithm despite its name is a classification algorithm. This algorithm expresses the output as a linear combination of the input features and uses the sigmoid function to convert it to a probability for a given class. Finally, it reaches a local minimum via the minimization of the cross-entropy loss function[9].

Linear Discriminant Analysis (LDA) is a classification method that makes specific assumptions about the classes and uses the Bayes theorem based on those in order to assign a sample to a class based on the one with the highest posterior probability. More specifically, LDA assumes that the classes are linearly separable, that they share the same covariance matrix and that each has its own multivariate normal distribution. Based on these assumptions, it maximizes (via its formula for weights) between class variance while minimizing within class variance and then solves the corresponding Bayes theorem formula[10].

Gaussian Naïve Bayes classifier is an algorithm that assumes that the features follow a normal distribution within each class and that they are conditionally independent. Given conditional independence, the likelihood of the input under each class can be calculated as a product of Gaussians (usually a log is taken to convert it to sums)[11].

Support Vector Classifier is an algorithm that finds the optimal hyperplane that separates the classes with the maximum margin, the points that are found on these margins are called support vectors. The strictness of the margin can be controlled by the hyperparameter C where higher values correspond to a stricter margin and smaller ones to a softer margin. SVC applies the kernel trick whereby it uses a kernel function to project the data in a high dimensional space[12]. There is a limited number of functions that qualify as kernel functions with prominent examples being:

- Polynomial
- Sigmoid
- Radial Basis Function
- Linear

PCA transforms a combination of correlated features to a set of Principal Components that are uncorrelated. This new set of features is ordered in descending order in terms of which feature captures the most variance. Initial scaling is necessary as PCA is not scaling invariant (eg. z-scaling). After z-scaling the covariance matrix is calculated and the eigenvalues and eigenvectors are found. Finally the eigenvectors with the top k eigenvalues are kept[13].

The workflow followed is made so as to ensure that the performance of the models is representative and that it captures the degree of overfitting as much as possible. Voting in the inner fold exists to give a first view on which model is expected to perform best on the given dataset. The three folds' results are averaged by Atom automatically and the model instance with the best Fbeta score is declared the winner model of the fold. This is done as from the distribution of the classes it is apparent that the minority class is the positive one and thus maximizing recall while maintaining a good precision (secondarily) is desirable.

After the winning model instance is decided, it is used to predict the test set classes, that were created in the outer loop and the results are kept for plotting. This is a first check for each of the algorithms' generalizability as finally a boxplot created by 50 individual values is created. On that boxplot, the mean, the median as well as the confidence intervals are shown and the range of the boxplot as well as its median is used in order to decide which model performs best for each metric. Finally, the eval set is utilized by performing bootstrapping for 1000 samples and plotting the metrics based on the predictions. This set serves as a holdout set and judging it in combination with the kfold plots and the voting results gives the final verdict.

If a model performs consistently well across all folds and the eval set, it is considered as the best model instance. In terms of importance, kfold plots are used as the most important component, followed by the ones from the eval set with the voting results being the least important. It should be noted that inconsistencies across the voting results, kfold metrics and evaluation set metrics were not observed, meaning that overall if a model performed well in voting it also did so in the boxplots and competition existed mainly among the best two-three models whose results in voting were almost identical (after PCA).

For fine-tuning, a unique approach was followed. Given the structure of the rnCV, it was decided to use optuna for each of the 50 folds separately and judge the best hyperparameters using the most frequent values for discrete valued hyperparemeters, and a representative value within the most frequent range of values for continuous valued hyperparameters. For that purpose, plots were made for the 50 folds and since the number of combinations and folds is given, it is assumed that the most frequent values are expected to perform better across most folds and thus provide the most generalizable instance. Finally, that instance is used in the rnCV class and with bootstrapping to make sure this is the case.

Conclusion

Breast cancer continues to be one of the more serious and increasing in frequency malignancies worldwide. While many methods and complex machine learning models have been developed for diagnosis and prognosis of such malignancies, the need for an evaluation across the simpler machine learning algorithms remains.

In this study, we examine the use of six different machine learning approaches (LightGBM, Random Forest, Logistic Regression, Support Vector Classification, Linear Discriminant Analysis and Gaussian Naïve Bayes) in predicting malignant samples across a population of benign masses and tumors from breast tissues. The features used in training these models originate from fine needle aspirates and describe 30 physical characteristics of cell nuclei. The dataset is imbalanced in favor of the benign class.

After applying the K-nearest neighbor algorithm to fill in missing values and RobustScaler to bring all values to the same scale (as some of the models are sensitive to features not being of the same scale), exploratory plots were made to assess the quality of the data and the separability of the classes based on different criteria. Based on the distributions of the features in the different classes it is expected that the data will be easily separated by the models used while the application of PCA seems to blend in seamlessly making the models more efficient as less features are utilized.

Runs were made utilizing the original features and default model hyperparemeters, as well as with PCA as the feature selection method. Based on the outer fold plots PCA does not affect negatively the predictive capability of the models, although bootstrapping on the evaluation set shows a different picture as the medians of the metrics have slightly dropped in comparison. In general, based on both the baseline and the PCA run, the model that was decided to be the best performing moving forward was the Logistic Regression algorithm mainly due to the need for PCA. However, fine-tuning on SVC after or without PCA would be also an interesting option.

After fine-tuning the LR model with default optuna hyperparameters a slight increase in the metrics of the kfold cross validation was observed (circa 1%) and a relatively larger one (2%) for metrics coming from bootstrapping on the evaluation set, showing that the generalization was improved. Additional feature selection methods and class balancing algorithms need to be tried (see bonus 1 and 2) as well as fine-tuning of models that came close to LR in terms of performance with and without PCA. However, the overall metrics are extremely promising and robust showing that these algorithms can be used for such purposes reliably.

Bonus-1

According to the bonus-1 requirement, 2 advanced feature selection methods implemented in Atom were examined in order to assess their capability in improving the efficiency of the models and potentially their performance.

The methods that were examined were the Particle Swarm Optimization Algorithm and the Gray Wolf Optimization Algorithm using Gaussian Naïve Bayes as the solver. Due to the nature of these algorithms in seeking the global minimum, the number of features parameter is ignored and an optimal set of features is found based on the cross-validation technique chosen. Here the rnCV structure described in the main part was utilized where in the inner cv the best model was decided and the outer cv and bootstrapping existed for evaluation purposes.

The PSO algorithm is a population based stochastic optimization algorithm based on the social behavior of birds or fish schools. It models this behavior via the use of particles. Each particle holds a set of selected features and across iterations its likelihood of dropping or maintaining features is determined based on a velocity equation that has terms representing how much a particle ‘trusts’ its own position versus that of the swarm, an inertia factor and a particle’s current position. A sigmoid is applied after updating the velocity in order to convert it to a probability used for feature selection. After a number of iterations, a single particle holds the optimal feature subset[14].

The Gray Wolf Optimization Algorithm is also a metaheuristic algorithm but it follows a different scheme than that of PSO. Instead of particles this algorithm initializes a population of possible solutions (wolves) and based on the objective function, it ranks them as Alpha, Beta, Delta and Omega. A position update function is employed that takes into consideration these solutions and has a factor for balancing the behavior between two phases (exploration, for finding new minima and exploitation, for refining solutions)[15].

After applying these two methods within the rnCV pipeline, the results were compared to the ones from the main requirements. The Gray Wolf Optimization Algorithm showed improved performance in terms of metrics compared to the main requirements with the best model still being Logistic Regression, however this came at the cost of more features being considered, more specifically this algorithm dropped only 3 features. Due to this, the final verdict is that PCA is still the superior method as the metric improvement is not enough to justify the computational burden of adding 17 more features compared to PCA’s 10 components.

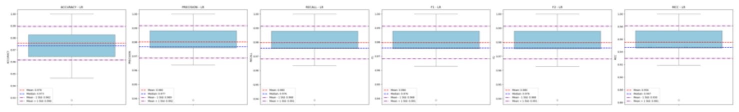

Figure 16. Fine-tuned outer-fold results of LR after GWO feature selection.

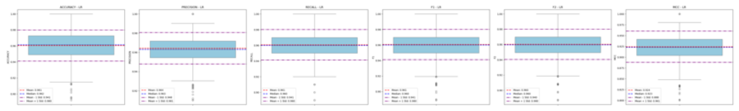

Figure 17. Fine-tuned bootstrap results of LR after GWO feature selection.

Bonus-2

According to the bonus 2 requirement, 2 class balancing methods were tried one uses undersampling of the majority class utilizing the Cluster Centroids method, while the other uses the class weight parameter in the models that allow it.

The Cluster Centroids Undersampling method works by performing K-Means clustering on the majority class and replacing specific samples with the representative of their cluster. This reduces the number of samples in the minority class making the dataset balanced and preventing the models from becoming biased to the majority class. The class weights parameter is built-in for 4/6 classifiers in scikit learn (excluding Gaussian Naïve Bayes and Linear Discriminant Analysis) and thus they were excluded from this method (although a more advanced way exists to implement a similar logic to these classifiers).

According to the resulting metric boxplots presented below, undersampling did not help the algorithms converge to a better minimum. In fact, in many models it deteriorated the performance slightly. After fine-tuning the results were similar to the ones without undersampling for bootstrapping, but worse for kfold, leading to the conclusion that another method for class balancing should be attempted as well (the rest of the plots can be found in the corresponding notebooks).

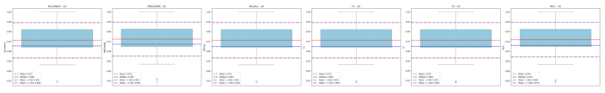
Figure 18. Fine-tuned kfold metrics of LR after undersampling.

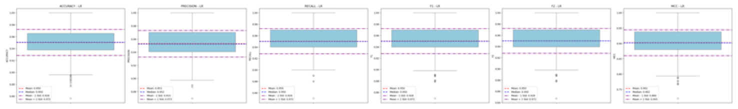
Figure 19. Fine-tuned bootstrap metrics of LR after undersampling.

The second method that was attempted was the usage of class weights in the models that immediately allow it, these being SVC, LR, LightGBM and Random Forest. Based on the results obtained, LR still was the best model and after fine-tuning, kfold results deteriorated by 1% while bootstrap results improved by 1%. Given that kfold results are more important due to the size of the data the models are tested on, the balanced weight parameter was not adopted.

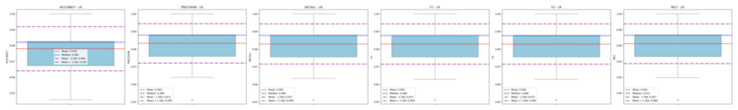
Figure 19. Fine-tuned kfold metrics of LR with class_weight balanced

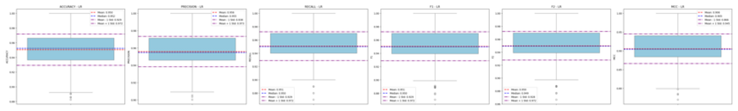
Figure 20. Fine-tuned bootstrap metrics of LR with class_weight balanced

On the Use of AI

During this exercise there was minimal use of Ai-generated code as , in big part, code was utilized from the previous exercise and refactored. More specifically the parts of the code written by AI (in this case ChatGPT) are marked with a comment above them, indicating their origin. In general, ChatGPT was used to get meaningful one-line commands that would remove the necessity of developing redundant classes/functions for their functionality otherwise.

Bibliography

[1] B. Smolarz, A. Z. Nowak, and H. Romanowicz, “Breast Cancer—Epidemiology, Classification, Pathogenesis and Treatment (Review of Literature),” *Cancers*, vol. 14, no. 10, p. 2569, May 2022, doi: 10.3390/cancers14102569.

[2] J. Xiao *et al.*, “The Application and Comparison of Machine Learning Models for the Prediction of Breast Cancer Prognosis: Retrospective Cohort Study,” *JMIR Med. Inform.*, vol. 10, no. 2, p. e33440, Feb. 2022, doi: 10.2196/33440.

[3] D. F. Sigmon and S. Fatima, “Fine Needle Aspiration,” in *StatPearls*, Treasure Island (FL): StatPearls Publishing, 2025. Accessed: Apr. 17, 2025. [Online]. Available: http://www.ncbi.nlm.nih.gov/books/NBK557486/

[4] C. Boeri *et al.*, “Machine Learning techniques in breast cancer prognosis prediction: A primary evaluation,” *Cancer Med.*, vol. 9, no. 9, pp. 3234–3243, May 2020, doi: 10.1002/cam4.2811.

[5] A. I. Penn *et al.*, “Discrimination of benign from malignant breast lesions in dense breasts with model-based analysis of regions-of-interest using directional diffusion-weighted images,” *BMC Med. Imaging*, vol. 20, no. 1, p. 61, Jun. 2020, doi: 10.1186/s12880-020-00458-3.

[6] “About - ATOM.” Accessed: Apr. 15, 2025. [Online]. Available: https://tvdboom.github.io/ATOM/latest/about/#citing-atom

[7] G. Ke *et al.*, “LightGBM: A Highly Efficient Gradient Boosting Decision Tree,” in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2017. Accessed: Apr. 16, 2025. [Online]. Available: https://papers.nips.cc/paper_files/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html

[8] L. Breiman, “Random Forests,” *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001, doi: 10.1023/A:1010933404324.

[9] D. R. Cox, “The Regression Analysis of Binary Sequences,” *J. R. Stat. Soc. Ser. B Stat. Methodol.*, vol. 20, no. 2, pp. 215–232, Jul. 1958, doi: 10.1111/j.2517-6161.1958.tb00292.x.

[10] “What Is Linear Discriminant Analysis? | IBM.” Accessed: Apr. 16, 2025. [Online]. Available: https://www.ibm.com/think/topics/linear-discriminant-analysis

[11] “(PDF) An Empirical Study of the Naïve Bayes Classifier,” ResearchGate. Accessed: Apr. 16, 2025. [Online]. Available: https://www.researchgate.net/publication/228845263_An_Empirical_Study_of_the_Naive_Bayes_Classifier

[12] N. Cristianini and E. Ricci, “Support Vector Machines,” in *Encyclopedia of Algorithms*, M.-Y. Kao, Ed., Boston, MA: Springer US, 2008, pp. 928–932. doi: 10.1007/978-0-387-30162-4_415.

[13] I. Jolliffe, “Principal Component Analysis,” in *International Encyclopedia of Statistical Science*, M. Lovric, Ed., Berlin, Heidelberg: Springer, 2011, pp. 1094–1096. doi: 10.1007/978-3-642-04898-2_455.

[14] J. Kennedy, “Particle Swarm Optimization,” in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds., Boston, MA: Springer US, 2010, pp. 760–766. doi: 10.1007/978-0-387-30164-8_630.

[15] S. Mirjalili, S. M. Mirjalili, and A. Lewis, “Grey Wolf Optimizer,” *Adv. Eng. Softw.*, vol. 69, pp. 46–61, Mar. 2014, doi: 10.1016/j.advengsoft.2013.12.007.