Polytechnic University of Turin

An exploration of ML methods for breast cancer classification

By

Alessandro Emmanuel Pecora

s290369

Mathematics in machine learning exam project

Master of Data science and engineering

2021/2022

ABSTRACT

With one of the most famous classification dataset "Breast Cancer (Original)", we will go trough an exploration of the main machine learning models and preprocessing methods for binary classification.

Starting from a data exploration performed with Principal Component Analysis and features correlation analysis, we will explore the possibility of generating new features, the "distances from positive and negative centroids".

Different experiments are performed with different settings and models, in order to be aware on how, the new generated features impact on results.

Key Words: [Breast Cancer; Machine Learning; Classification; Principal Component Analysis; Correlation Analysis; Feature extraction].

Contents

Introduction

Breast cancer (BC) is one of the most common cancers among women worldwide, representing the majority of new cancer cases and cancer-related deaths according to global statistics, making it a significant public health problem in today's society. The correct diagnosis of BC and classification of patients into malignant or benign groups is the subject of a lot of research. Because of its unique advantages in critical features detection from complex BC datasets, machine learning (ML) is widely recognized as the methodology of choice in BC pattern classification and forecast modelling.

The following analysis is based on the real dataset "Breast Cancer Wisconsin (Original) Data Set" [Wolberg, 1992].

Samples arrive periodically as Dr. Wolberg reports his clinical cases. The database therefore reflects this chronological grouping of the data.

Dataset Overview

The data set contains **699** instances. For each sample we have 11 cytological characteristics of fine needle aspirations of the breast that differ between benign and malignant samples, each characteristic was classified from 1 to 10 at the time of sampling.

Attribute information:

1. Sample id number: this attribute represent the index of the dataset. For our purpose we will drop it since is senseless for classification.

Type: -; *Range:* -

2. Clump Thickness: the extent to which epithelial cell aggregates were mono- or multilayered.

Type: integer; *Range:* [1 - 10]

3. Uniformity of Cell Size: indicating metastasis to lymph nodes.

Type: integer; *Range:* [1 - 10]

4. Uniformity of Cell Shape: identifying cancerous cells of varying size.

Type: integer; *Range:* [1 - 10]

5. Marginal Adhesion: cohesion of the peripheral cells of the epithelial cell aggregates. Suggesting loss of adhesion, i.e., a sign of malignancy but the cancerous cells lose this property so this retention of adhesion is an indication of malignancy

Type: integer; *Range:* [1 - 10]

6. Single Epithelial Cell Size: if the SECS become larger, it may be a malignant cell.

Type: integer; *Range:* [1 - 10]

7. Bare Nuclei: the proportion of single epithelial nuclei that were devoid of surrounding cytoplasm. Found in benign tumors.

Type: integer; *Range:* [1 - 10]

8. Bland Chromatin: usually found in benign cells describes a uniform texture of the nucleus. In cancer cell, the chromatin tends to be coarser.

Type: integer; *Range:* [1 - 10]

9. Normal Nucleoli: generally very small in benign cells.

Type: integer; *Range:* [1 - 10]

10. Mitoses: the process in cell division by which the nucleus divides.

Type: integer; *Range:* [1 - 10]

11. Class: The target class to predict.

Type: categorical; *Set:* {2('b'): benign; 4('m'): malignant}

NOTE: the value for "Class" are replaced with "b" for benign and "m" for malignant, during the preprocessing.

Dataset Exploration

In this chapter we explore the dataset in order to understand: if it is balanced, if duplicates are present, how to manage those and, if there are null values, how to handling those. Finally we explain the importance of scaling features.

3.1 Balancing

We want to find if the number of instances for benign and malignant cancers are balanced since unbalanced datasets could affects the results of classification, infact the analysis of label distribution is a crucial steps in data exploration. Figure 3.1 show the classes distribution.

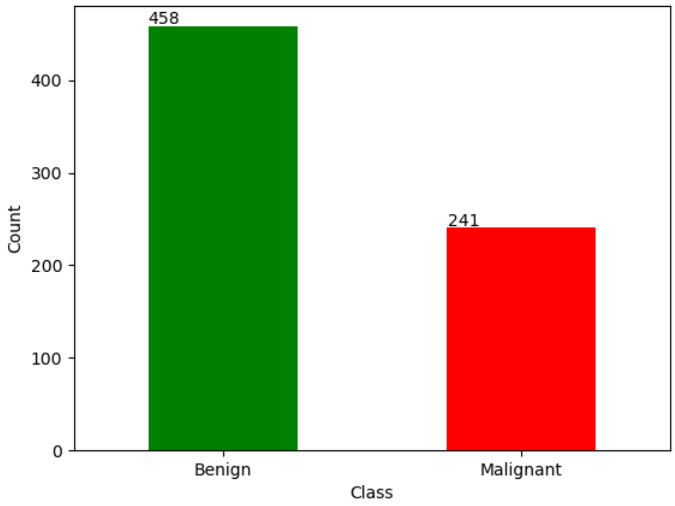

Figure 3.1: Classes distribution, the dataset is unbalanced, we have almost double benign instances.

The dataset is unbalanced in favour of benign labels, to avoid missclasification of minority class we want to balance the dataset.

Different technique are avalaible, like undersampling or oversampling, but before applying the mentioned techniques, we want to know if duplicates are present because, maybe by removing those, the dataset could become balanced.

3.2 Duplicates

By checking for duplicates we found that we have **two kinds of duplicates**, in fact, before dropping the attribute **sample id number**, we found that there are **54 duplicates**, since also the sample id is the same, probably, this duplicates are diagnosis of Dr. Wolberg about the same patients who have made several visits at different times and obtained the same results.

After drop out the sample id number attribute, we have another kind of duplicates, in fact we found other **182 duplicates** that are different from the previous, since the sample id number is different so are related to different patients with the exactly same diagnosis.

In order to balancing the dataset **we drop all the 54+182=236 duplicates**, we found that most of them are benign cancer (only 3 malignant cases are duplicated), this lead to a dataset **with 463 samples**.

Figure [3.2] shows the class distribution after we have dropped the duplicates. Now we have a bit more malign cases then benign, but the gap is very low and we can consider the **dataset** as **balanced**.

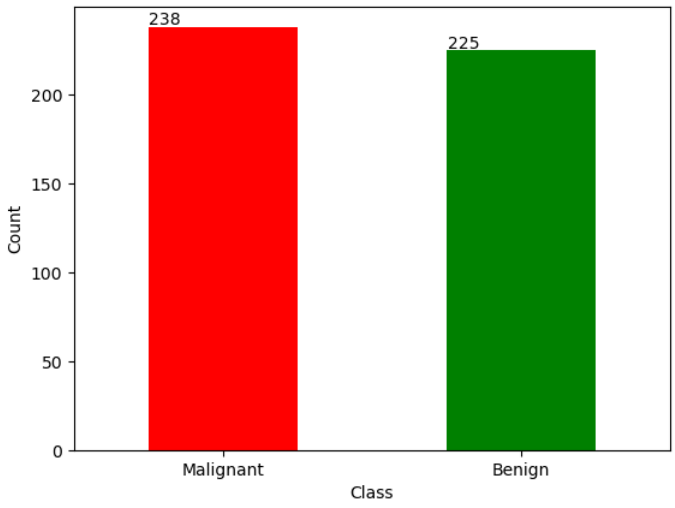

Figure 3.2: Classes distribution after dropping duplicates, the dataset now is well balanced.

3.3 Null value handling

About null value we found that are presents only for **bare nuclei** attribute, in particular we found **14** null instances.

We decide to handle the null value by **replacing it with the median of training examples** since the value are integer and we prefer to mantain the same domain, other strategies are possible but since we have few null instances the impact of different methodologies is minimal.

3.4 Standardization

A common pre-processing practice in ML pipeline is standardize the dataset. Standardizing the features with 0 mean and with a standard deviation of 1 is important when we compare measurements that have different units. Variables that are measured at different scales do not contribute equally to the analysis and might end up creating a bias. Some method and models, like PCA, SVC and KNN, require data to be standardized since the distance equations used in these are affected by the different scales of the attributes. Instead other method like: Correlation Analysis, Random Forest and Logistic Regression are not affected. An important point in our analysis is the standardization with the extracted features, that we will introduce in next chapter.

Features extraction and data analysis

In the following section we will propose a simple and efficient technique of **features extraction**, inspired from the *"Supervised compact hypersphere"* of [Tingting Mu, 2008]. We will call it *"Centroids Distances"* (*CD*), this method: improves the final results of most of the analyzed models and is a very simple way to visualize the data. Then we provide the explanation and the results of two well known analysis methods, **Principal Component Analysis (PCA)** and **correlation analysis**, those methods permits to visualize the data and the relation between the different features, including the new extracted features.

4.1 Features Extraction: Centroids Distances

The low number of features allows us to focus the study on the extrapolation of new features, starting from the original ones.

Following this idea we decide to generate 2 new features for each sample in dataset, the **distances from benign and malign centroids**. The centroids are computed, at training time, in the features space (i.e. a 9-dimensional space) as the mean for benign and for malign training samples:

$$\begin{aligned} \textbf{Centroid('b')} &= \frac{1}{|b\_samples|} \sum_{\forall \mathbf{x} \in b\_samples} \mathbf{x} \\ \textbf{Centroid('m')} &= \frac{1}{|m\_samples|} \sum_{\forall \mathbf{x} \in m\_samples} \mathbf{x} \end{aligned} \tag{4.1}$$

Then, at inference time, we extract the distances of the sample $\mathbf{x}$ from the benign and malign pre-computed centroids:

$$\begin{aligned} CD('b', \mathbf{x}) &= \sqrt{\sum_{i}^{n\_features} (Centroid('b')_i - x_i)^2} \\ CD('m', \mathbf{x}) &= \sqrt{\sum_{i}^{n\_features} (Centroid('m')_i - x_i)^2} \end{aligned} \tag{4.2}$$

Finally we use $CD('b', \mathbf{x})$ and $CD('m', \mathbf{x})$ as new features for each sample $\mathbf{x}$. We will show in next chapter that using the CD features improve results of many of analyzed models, moreover, an interesting analysis, is the scatterplot of the features in order to visualize the data.

In figure 4.1 and 4.2, the training data are plotted in a 2D space represented

by the two new features, the figures show the effect of standardization.

Figure 4.1: Training data in *"CD-Space"*, extracted **from original features**.

Figure 4.2: Training data in *CD-Space*, extracted **from standardized features**.

An important note about standardization when using this method is necessary. Since the method work with distances, we should standardize the features before extraction. Moreover if we want to use the CD features, in a method affected by the differents scales (e.g. PCA, SVM, KNN), we should standardize again after the extraction phase.

4.2 Principal Component Analysis

Principal component analysis is a mathematical technique largely used in the field of data science, in particular it finds usefull applications in data visualization and features reduction. For our purpose we use PCA analysis to:

- Visualize the data and compare the representation in PCA-space with the representation in the CD-space.
- Compare the total variance explained of PCA with and without the CD features.

PCA is defined as an orthogonal linear transformation that transforms the data to a new coordinate system such that: the greatest variance by some

scalar projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on. The main idea is therefore to seek the most accurate data representation in a lower dimensional space, to do that we can follow this passages:

- Assume each feature of the dataset is a random variable X_i.
- Compute the covariance matrix Σ of the whole dataset, i.e. the matrix in which the entry (i, j) is the covariance:

$$cov(X_i, X_j) = E[(X_i - E[X_i])(X_j - E[X_j])] \tag{4.3}$$

- Retrieve the eigenvalue λ and the associated eigenvector $\mathbf{v}$ of Σ by solving:

$$det(\Sigma - \lambda I) = 0 \tag{4.4}$$

- Sort the eigenvector by decreasing eigenvalues and choose the first $\mathbf{k}$ eigenvector to form a $(n\_features\ ,\ k)$ dimensional matrix W, this represent our new reduced space.
- Transform the samples, x_i onto the new reduced space:

$$\hat{x_i} = W^T x_i \tag{4.5}$$

Another way to look at PCA is by use an optimization problem that finds a new $k - dimensional$ subspace in such a way that the total squared distance between the projected points, $WW^T\mathbf{x}_i$ and the original points, $\mathbf{x}_i$, is minimal.

$$min_{w_1,\dots,w_k} \sum_{i}^{n\_samples} ||\mathbf{x}_i - WW^T\mathbf{x}_i||^2 \tag{4.6}$$

With the cyclic property of a trace, the equivalence $WW^T = \sum_l^k \mathbf{w}_l\mathbf{w}_l^T$ and some algebraic passages, we observe:

$$\frac{1}{n}\sum_{i}^{n} ||\mathbf{x}_i - WW^T\mathbf{x}_i||^2 = \frac{1}{n}\sum_{i}^{n} ||\mathbf{x}_i||^2 - \sum_{l}^{k} \mathbf{w}_l^T\Sigma\mathbf{w}_l \tag{4.7}$$

Where the red part is constant and Σ is the covariance matrix. In this way

the minimization problem (4.6) is equivalent to:

$$max_{w_1,\ldots,w_k} \sum_{l}^{k} \mathbf{w}_l^T \Sigma \mathbf{w}_l \tag{4.8}$$

In 4.8 the maximum is obtained exactly when **w1,...,wk** are the first k principal components of Σ.

It is fair to point out that PCA finds the most accurate data representation in a lower dimensional space, however, the directions of maximum variance may be useless for classification but for sure help the human visualization.

Figures 4.3 and 4.4 gives an hint on how PCA work to find the best projection in terms of variance:

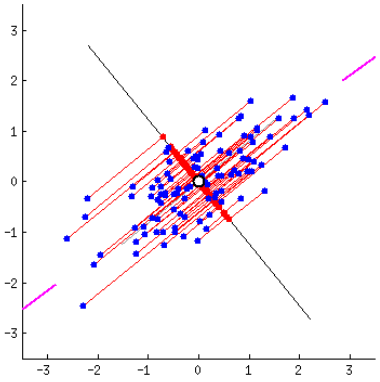

Figure 4.3: PCA, projection with **lower variance**.

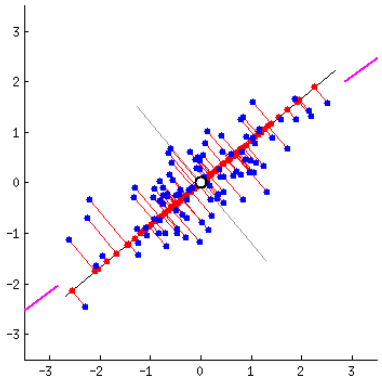

Figure 4.4: PCA, Projection with **higher variance**

Figures 4.5 and 4.6 show the results of PCA analysis computed on training data, in particular 6 representations are plotted in 3 different spaces, generated by using one, two or three principal components as basis. For each space, the results of PCA are plotted in two case: first, by reducing only the original features, and then, by reducing the original and the new extracted features togheter.

NOTE: For the first scatters the y-axis is a random jitter added to help the visualization.

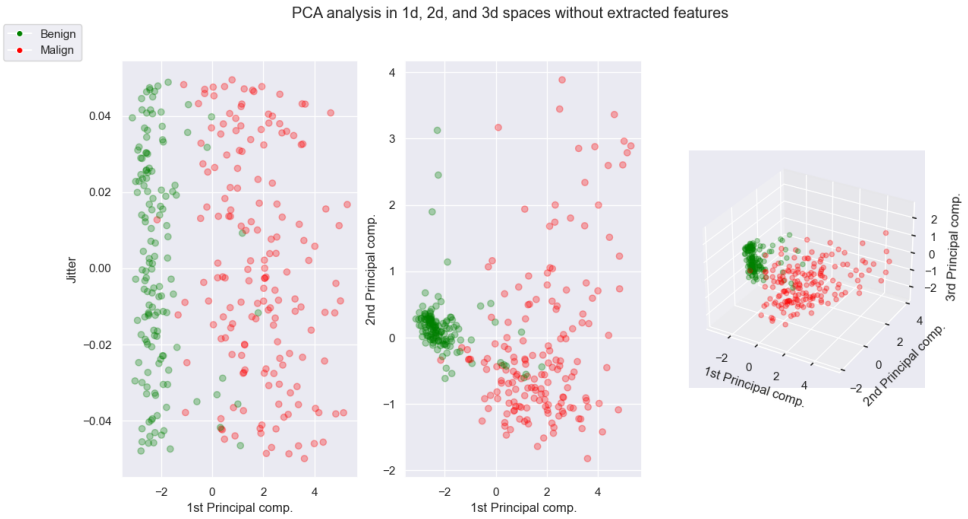

Figure 4.5: Visualization of the samples **without** the CD features, in the pca-spaces

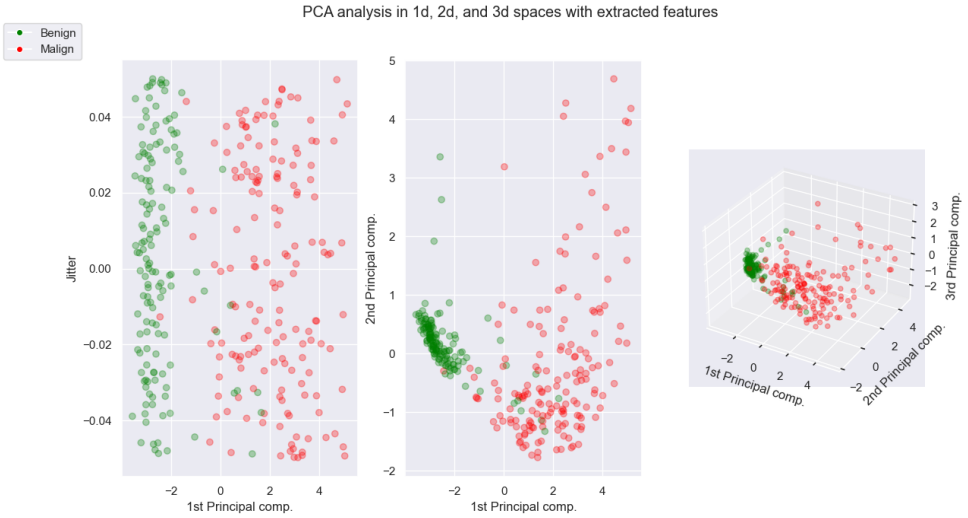

Figure 4.6: Visualization of the samples **with** the CD features, in the pca-spaces

Is interesting to compare 4.1, with the 2D scatter of 4.5. Both provides a 2D visualization method but, $CD - space$ have a simpler computation, however can be applied only for "2/3-class" classification problem, while PCA not have this limitation.

Moreover from an analytical point of view we report the Explained variance of the PCA with $k = 3$ principle components, in two cases: with and without the

Cetroids Distances features. The Explained variance $EV(\lambda_i)$ refers to the variance explained by each of the principal components (eigenvectors):

$$EV(\lambda_i) = \frac{\lambda_i}{\sum_j^{n\_features} \lambda_j} \tag{4.9}$$

Where λ_i is the $i-th$ eigenvalue associated with the $i-th$ principal components.

- Variance explained **without extracted features**

  Explained variance of each component: [0.59887984 0.0958903 0.07204125]

  Total Variance Explained: 76.68%

- Variance explained **with extracted features**

  Explained variance of each component: [0.61920395 0.11109347 0.0594775]

  Total Variance Explained: 78.98%

Adding the extracted features, permits the PCA to find a projection that can explain more variance.

4.3 Correlation Analysis

4.3.1 Pearson coefficient and Heatmap

Correlation analysis is a statistical method used to measure the "strength" of the linear relationship between two variables and compute their association. Correlation analysis calculates the level of change in one variable due to the change in the other. Different correlation index exists, for our purpose we use the **Pearson correllation coefficient**. Given a pair of random variables (X,Y), the coefficient correlation ρ is:

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \tag{4.10}$$

Where $cov()$ is the covariance and σ_X, σ_Y are the standard deviations for X and Y.

The result of 4.10 is in range [-1,1]:

- **Positive value:** means that the two variable go in the same direction, 1 means a perfect positive linear correlation.

- **Negative value:** means that two variable go in the opposite direction, -1 means a perfect negative linear correlation.
- **Weak/zero value:** means that a linear correlation not exist.

We use the Pearson correlation to compute the coefficients for each pair of features of our dataset, then we can visualize all correlations in an intuitive way, with the well known **Correlation Heatmap**. This is a 2D matrix with the features as rows and columns, each cell represents the correlation between two features, a chromatic scale is used to help in the visualization. In 4.7 we present the correlation heatmap, computed with Pearson's formula for our features:

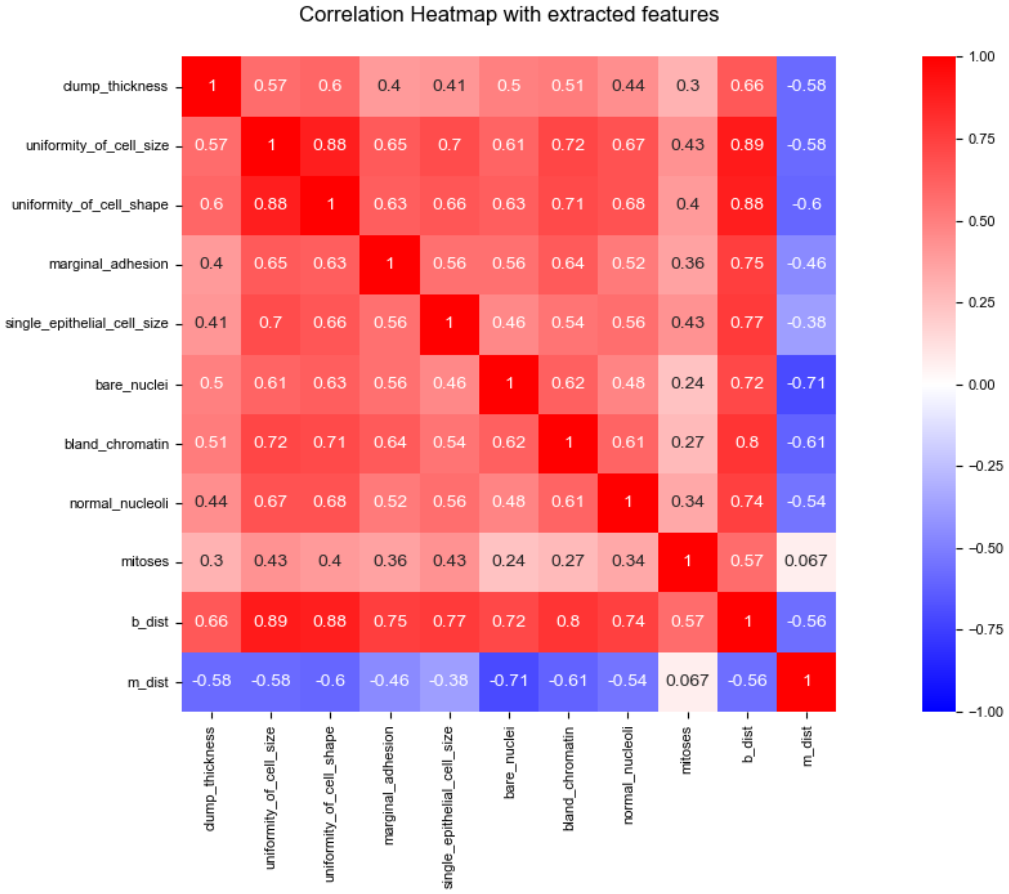

Figure 4.7: Pearson correlation heatmap of all features of dataset, including the new extracted features.

The heatmap is usually used in data science when we want to reduce the features. When a pair of feature have an high correlation, we can remove one of

those, since high correlated feature have the same impact on dependent variable, so introduce redundancy for our prediction models. However in our experiments we mantain all features, because are not too much. Is interesting to note that the **mitoses attribute** respect to others, is, in average, less correlated with all other variables, in particular obtain a 0 correlation with the **"distance from malignant centroid attribute"**, this suggest that the new feature is not redundant (respect to "mitoses"), so probably can be usefull to improve the models.

4.3.2 Pairs scatterplot

Another way to visualize the correlation between the data is the **Pair scatterplot**, it allow us also to see (in the diagonal) the distribution of single variables across the labels. The pairplot creates a grid of axes such that: each variable in data will by shared in the y-axis across a single row, and, in the x-axis across a single column. Figure 4.8 show the Pairs scatterplot for our dataset.

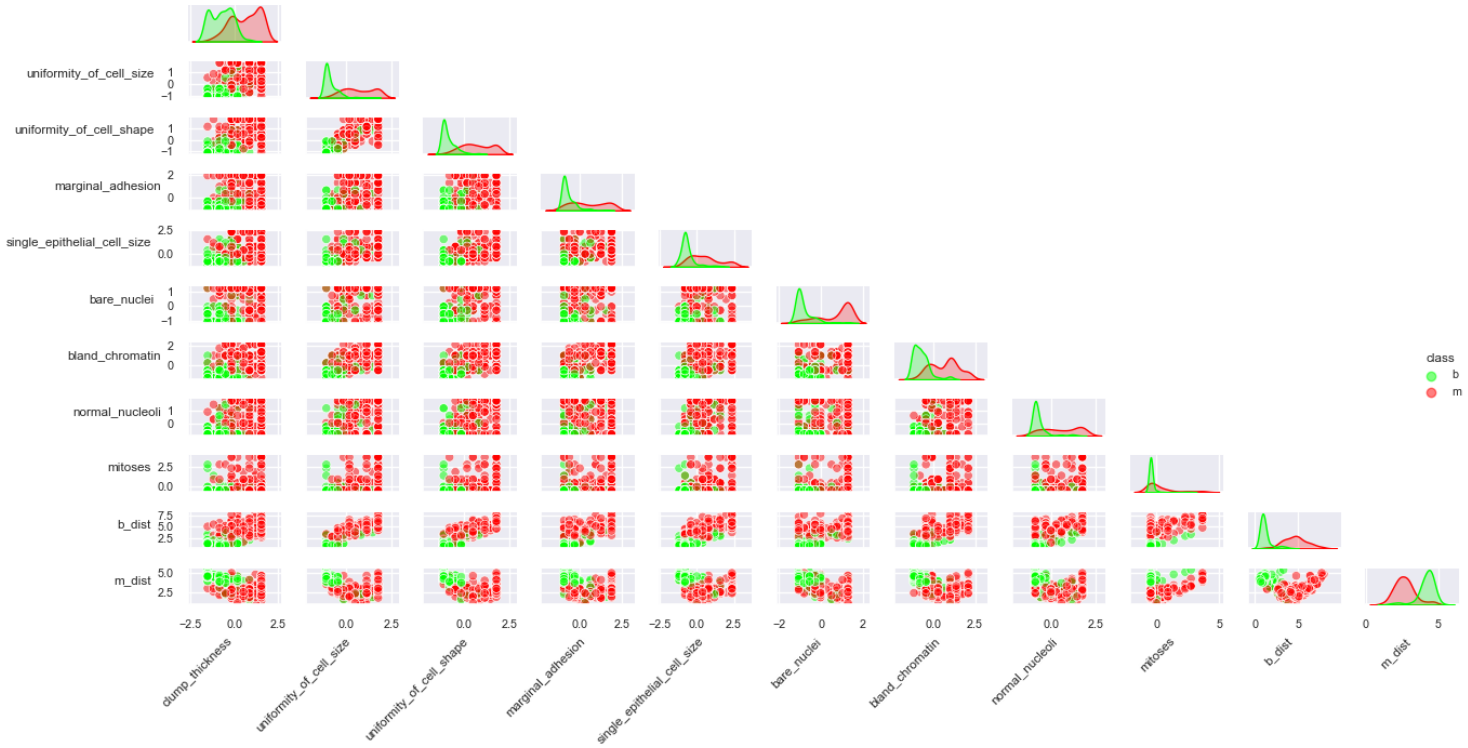

Figure 4.8: Pairs scatterplot, in the most cases malign cancer have higher values of features.

Experiments Settings

In this sections we describe the main settings, the validation method and the hyperparameters tuning technique used in the experiments. A repository, [Pecora, 2022], with the description on how to launch the code, is avalaible on github. The language used for all experiments is Python with Scikit-learn [Pedregosa et al., 2011] as library for ML models and methods.

For all experiments the **33% of the dataset is used as test-set** to get the final results, after the hyperparameters tuning. The remaining data are the train-set and are further splitted in train and validation, following the cross-validation method.

5.1 Cross validation

Cross-Validation is a statistical method of evaluating and comparing learning algorithms by dividing data into two segments: one used to learn or train a model and the other used to validate the model. In typical cross-validation, the training and validation sets must cross-over in successive rounds such that each data point has a chance of being validated against. The basic form of cross-validation is k-fold cross-validation where the training set is split into k smaller sets. For the training phase k - 1 of the folds are used, and the resulting model is validated on the remaining part of the data. The figure 5.1 helps to better understand this procedure. **For our experiments we set k=5.**

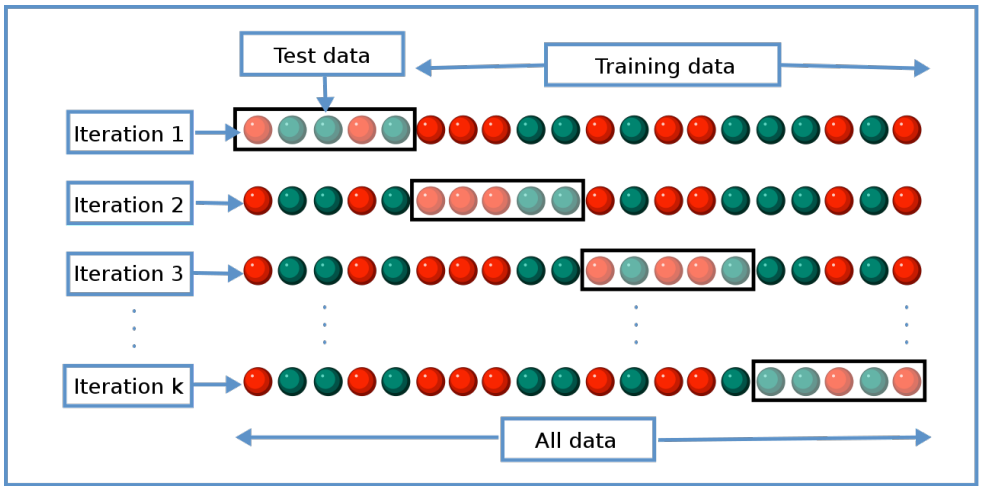

Figure 5.1: K-fold cross validation: at each iteration the model is trained on different data, as validation score the average of all iteration is used.

5.2 Grid search

A model hyperparameter is a characteristic of a model that is external to this and whose value cannot be estimated from data. The value of the hyperparameter has to be set before the learning process begins. For example, c in Support Vector Machines, k in k-Nearest Neighbors or also k in PCA. Grid-search is used to find the optimal hyperparameters of a model which results in the most 'accurate' predictions, the method consist in building a model for every combination of hyperparameters specified and evaluates each model. A more efficient technique for hyperparameter tuning is the Randomized search, where random combinations of the hyperparameters are used to find the best solution.

Models and results

In the following sections we report the experiments done with 4 different machine learning models, in order to compare the methods, explain the differences and find the best classification model for the task. For each model we repeat the experiments with and without the feature extraction, described in 4.1, to asses the usefulness of the method.

For medical applications often miss-classification on one class are more critically respect other class, a false benign cancer is more critical then a false malign, for this reason we will compute recall, precison and f1-score, both for malign and benign class. Moreover a real application should suggest to the expert the diagnosis results and these should be interpretable, to do that, for each model we explore the possibility of retrieve the probability to belong to one class, in this way, together with the labels, a sort of index of certainty is returned.

NOTE: For the following scatterplot, the visualization of the probabilities, is done only for samples that have high uncertainty, i.e. the ones, that have a difference in the class probabilities ≤ 0.8.

6.1 K-Nearest Neighbors

The K-nearest neighbors is one of the simplest classification/regression algorithm, this assumes that similar things exist in close proximity. In other words, similar things are near to each other. The algorithm can be summarized as follow:

- Load the training dataset (with the labels)
- When a new sample arrive:
  - Compute the distances between the sample and the preloaded samples.
  - Select the first K-nearest neighbors
  - Get the distribution of the K-nearest neighbors over the labels (voting phase).
- The label with more vote is the label of the new sample.
- In case of regression the the mean of the K-nearest neighbors is used as result.

We have two main hyper-parameter:

- K: the number of the neighbors that contributes with their vote.
- Voting function weights: we can use a uniform distribution or a distribution weighted with the distances, nearest neighbors vote weight more.
- Grid used for hyper-parameter:
  - K : (1,2,3,5,10)
  - weights: (uniform, distance)

About the probability to belong or not to one class, for the KNN can be represented by the proportion of vote to one or other class.

The following table reports the results, computed on test set, of three KNN model trained: with only original features, with CD and original features together and with only CD features.

	Best params	Accuracy	Precision		Recall		F1		Support	
			Benign	Malign	Benign	Malign	Benign	Malign	Benign	Malign
KNN (original feats.)	K:3; weights: uniform	0,92	0,91	0,94	0,95	0,89	0,93	0,91	82	71
KNN (CD+Original feats.)	K:3; weights: distance	0,93	0,92	0,94	0,95	0,90	0,93	0,92	82	71
KNN (CD feats.)	K:10; distance: uniform	0,95	0,97	0,93	0,94	0,97	0,96	0,95	82	71

Table 6.1: KNN results, using the "Distance from centroids" improves the results

From the above table we note that the results are improved thanks to CD features. Use only CD features is better than using original and CD features together. This is related with the overfitting phenomenon, infact, these two models achieve a perfect score in training setting (6.2). Centroids in CD method are computed on training set, this increase the chance to overfitting. However reducing the attributes by using only CD features permits to generalize well.

The following scatterplots are generated with the model that achieve the best results.

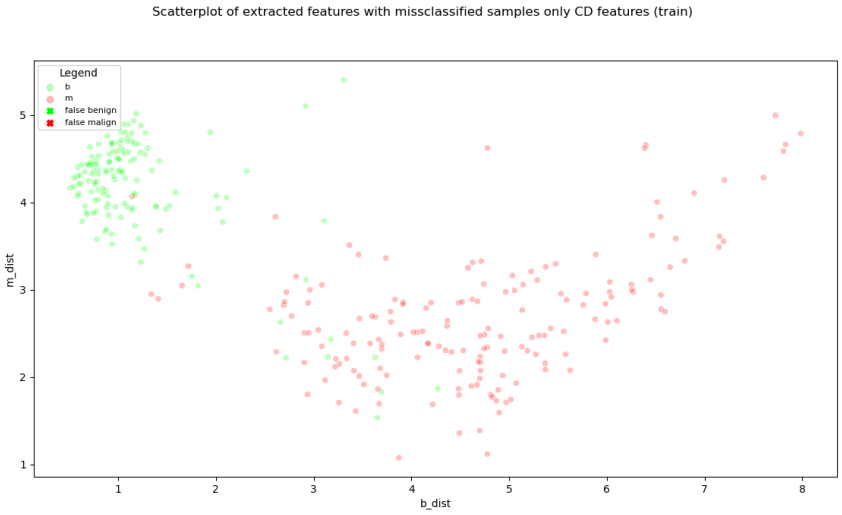

Figure 6.1: Visualization in "CD-space" of the miss-classified samples in train set with the probability to be respectively benign or malign.

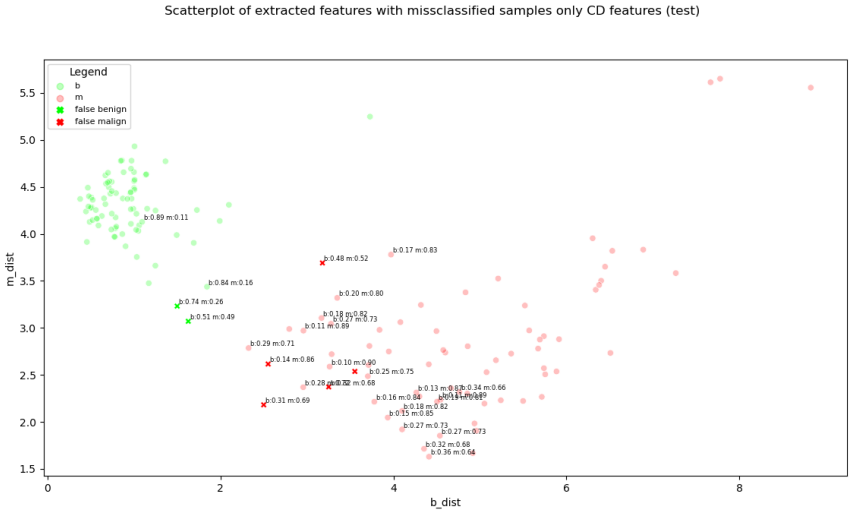

Figure 6.2: Visualization in "CD-space" of the miss-classified samples in test set with the probability to be respectively benign or malign.

6.2 Random Forest

Random forest is a supervised machine Learning algorithm that ensemble a set of **Decision trees** on different samples.

A **Decision tree** is a flowchart-like structure (6.3) in which each internal node represents a "test" on an attribute (e.g. $x_2 > tl_1$), each branch represents the outcome of the test (true or false) and splits the node, each leaf node represents a class label (decision taken after computing all attributes). In this way, the paths from root to leaf represent classification rules.

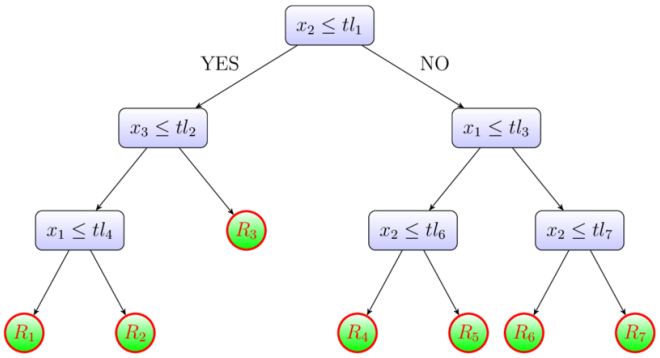

Figure 6.3: An example of a structure of decision tree.

The decision tree, in training phase, try to find the **best split**, of samples, for

each node, by tuning the "tests". Starting with all training samples, a criterion that measures the statistical dispersion of the splits, is used to retrieve the best test, i.e. the ones that generates the new sub-sets with homogeneity as high as possible. The process is reiterated for new branches recursively, until some stop condition is reached, or all attributes with all their possible values have already been tested. Note that different algorithms are avalaible and depends on the domain of the attributes, different technique are possible to tune the test. For numerical and categorical attributes, the most used criterion is the **Gini index**:

$$G = \sum_{i \in C} P(i) * (1 - P(i)) \tag{6.1}$$

C is the classes set, $P(i)$ is the probability to belong to class i, i.e. frequency of i computed on the current branch. A value for G of 0 means that the current set only have samples that belong to the same class, i.e. a perfect split, so we prefer splits that minimize the Gini index.

A random forest aggregates many decision trees with the **Bagging** (also known as **Bootstrap aggregation**) method:

It creates different training subsets from training data by sampling with replacement, each tree is trained on different training subset, the final output is based on majority voting, i.e. the class predicted by the majority of trees in the forest is the class label.

For random forest we tune 5 hyper-parameters:

- criterion: the method used to measure the homogeneity
- max_depth: the maximum depth of a tree, this is used to avoid overfitting.
- max_features: the number of features to consider when looking for the best split.
- min_samples_split: the minimum number of samples required to split an internal node.
- n_estimators: the number of tree to ensemble in the forest.
- Grid used for hyper-parameter tuning:
  - criterion: (gini, entropy)
  - max_depth: (1,2,5,8,None)
  - max_features: (None, sqrt)

- min_samples_split: (2,5,10)
- n_estimators: (1,5,10,50,100,500)

About the probability to belong or not to one class, like for the KNN, for the Random forest we can represent it with the proportion of votes, given by each tree in the forest, for one or other class.

The following table reports the results of three Random forest models, computed on test set and trained: only with original features, with original and CD features and only with CD features.

	Best param	Accuracy	Precision		Recall		F1		Support	
			Benign	Malign	Benign	Malign	Benign	Malign	Benign	Malign
Random Forest (Original feats.)	crit.: gini; max_dep.: 5; max_feat.: sqrt; min_samp_split: 2; n_est.: 500	0,95	0,96	0,94	0,95	0,96	0,96	0,95	82	71
Random Forest (CD+original feats.)	crit.: gini; max_dep.: None; max_feat.: sqrt; min_samp_split: 5; n_est.: 50	0,95	0,96	0,94	0,95	0,96	0,96	0,95	82	71
Random Forest (CD feats.)	crit.: gini; max_dep.: 2; max_feat.: sqrt; min_samp_split: 2; n_est.: 100	0,96	0,97	0,95	0,95	0,97	0,96	0,96	82	71

Table 6.2: Random Forest Results, using the "Distance from centroids" improves the results.

Also for Random Forest like for KNN, the CD features improve the results. In particular, when CD+Original feats. are used: the accuracy in test set is equal but, in train set (not reported), is higher, i.e. the model can't generalize anymore and could start to overfit. By using only CD features the train accuracy (see 6.5) is reduced, this permits to generalize well and to achieve the best results.

The following scatterplots are computed with the best model.

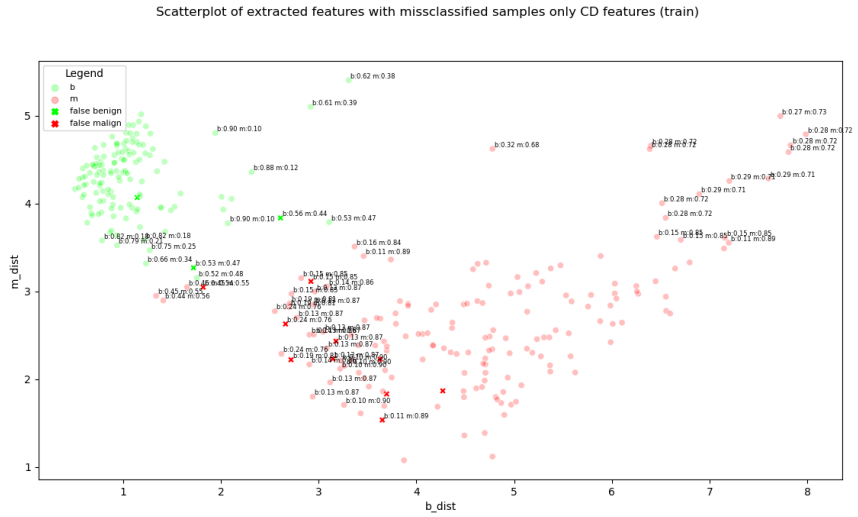

Figure 6.4: Visualization in "CD-space" of the miss-classified samples in train set with the probability to be respectively benign or malign.

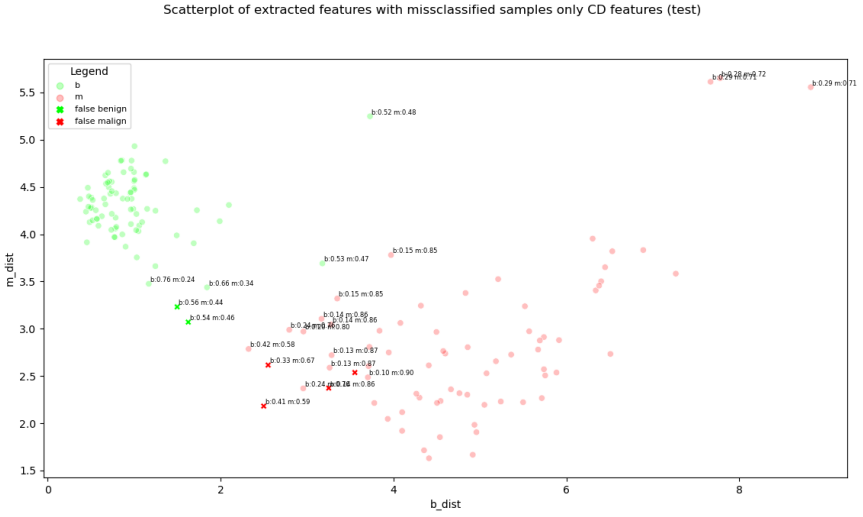

Figure 6.5: Visualization in "CD-space" of the miss-classified samples in test set with the probability to be respectively benign or malign.

6.3 Logistic Regression

Logistic regression is a statistical method which models a **linear combination** z (6.2) of one or more independent variables, the features. Then z is remapped in a probability with a **logistic function** $\sigma(z)$ (6.6). The model is fitted trough the **likelihood function** L (6.3), in which the two classes, {'benign','malign'}, are considered as two possible outcomes of an event, {0, 1}. The **Bernoulli distribution** is used in L to determinate the probability of belonging to class 1. For this reason a decision boundary is needed in prediction phase (6.5).

To estimate the probabilities, the logistic regression learn, from a training set, the coefficients of z, i.e. a vector of weights $\mathbf{w}$ and a bias term b, trough the **maximum likelihood estimation** (6.4) and numerical methods. Each weight is a real number associated with one of the input features x. The weight represents how important that input feature is to the classification decision. The bias term instead is another real number added to model the 'intercept'.

The linear combination, for an instance, is:

$$z = \sum_{i}^{n\_features} w_i x_i + b = \mathbf{wx} + b \tag{6.2}$$

Then z is remapped with the **logistic function**, hence the name of the model, in a legal probability, the most famous logistic function is called **Sigmoid**:

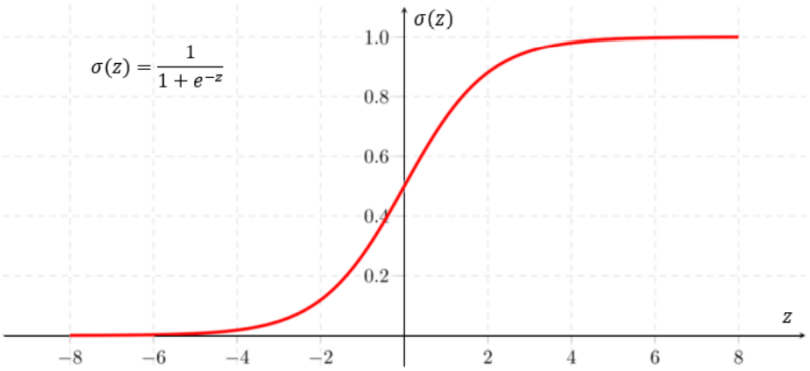

Figure 6.6: Sigmoid function as logistic function.

So the **likelihood function** is:

$$L(\mathbf{w}, b) = \prod_{i}^{n\_samples} \sigma(z)^{y_i}(1-\sigma(z))^{(1-y_i)} \;\; y \in \{0, 1\} \tag{6.3}$$

We can estimate the parameters with an optimization problem that maximize L, this method is called **maximum likelihood estimation**:

$$(\mathbf{w}, b) \leftarrow argmax_{(\mathbf{w},b)} L(\mathbf{w}, b) \tag{6.4}$$

here numerical methods are needed to find the solution.

To make a binary classification then we need to set the decision boundary:

$$\hat{y} = \begin{cases} 1(malign) \; if \; \sigma(z) \geq 0.5 \\ 0(benign) \; if \; \sigma(z) < 0.5 \end{cases} \tag{6.5}$$

For Logistic Regression we tune 4 hyper-parameters:

- Penalty: add a penalty term in the optimization problem that find the parameters. Usefull for regularization.
- C: regularization term multiplied to the penalty term.
- max_iter: maximum number of iteration of the numerical solver.
- dual: use the primal or dual optimization problem formulation.
- Grid used for hyper-parameter tuning:
  - penalty: (none, l2)
  - C: (1, 10, 100)
  - max_iter: (500,1000)
  - dual: (True only for l2, False)

About on how to retrieve the probabilities with a Logistic regression model, as described above, these are already computed by the model.

The following table reports the results of the three LR models, computed on test set, trained: with only original features, with CD and original features and with only CD features.

	Best param	Accuracy	Precision		Recall		F1		Support	
			Benign	Malign	Benign	Malign	Benign	Malign	Benign	Malign
Log Regression (Original feat.)	C: 1; dual: False; max_iter: 500; penalty: none	0,95	0,95	0,94	0,95	0,94	0,95	0,94	82	71
Log Regression (CD+Original feat.)	C: 10; dual: False; max_iter: 500; penalty: l2	0,94	0,94	0,94	0,95	0,93	0,95	0,94	82	71
Log Regression (CD feat.)	C: 1; dual: False; max_iter: 500; penalty: none	0,95	0,96	0,94	0,95	0,96	0,96	0,95	82	71

Table 6.3: LR Results, using only CD features improve precision and recall.

In this case the CD features, permit a low improvement in terms of precision for benign cases and recall for malign cases.

The following scatterplots are computed with the model that use only CD features.

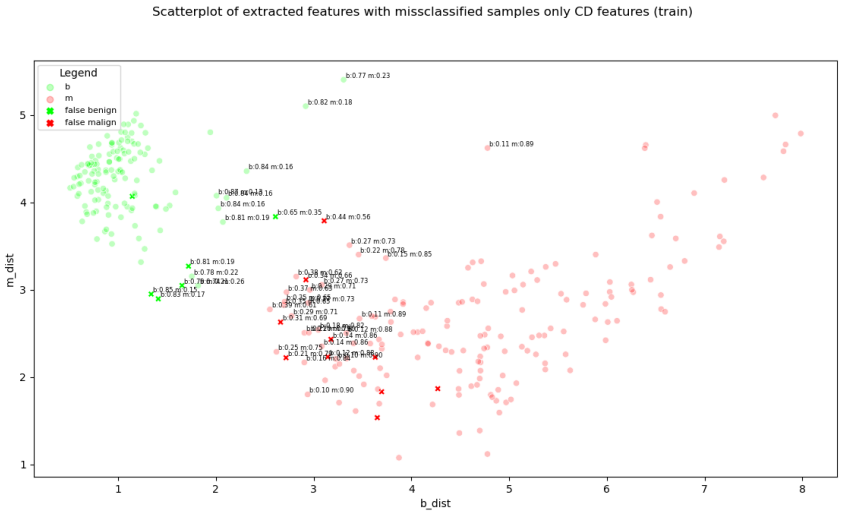

Figure 6.7: Visualization in "CD-space" of the miss-classified samples in train set with the probability to be respectively benign or malign.

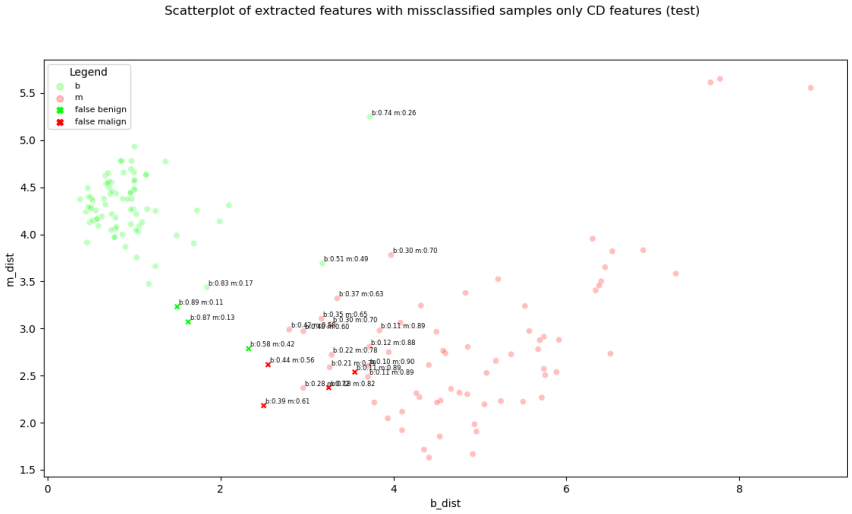

Figure 6.8: Visualization in "CD-space" of the miss-classified samples in test set with the probability to be respectively benign or malign.

6.4 Support Vector Machine

Support Vector Machine is another model of machine learning that try to find the best decision boundary, by maximize the **margin**, to separate two different classes. Margins are the (perpendicular) distances between a separating hyperplane and those dots (samples) closest to the hyperplane.

Any hyperplane can be written as the set of points x satisfying:

$$\mathbf{w}^T\mathbf{x} - b = 0 \tag{6.6}$$

Where $\mathbf{w}$ is the normal vector to the hyperplane.

If the data are linearly separable, we can select two parallel hyperplanes that separates the two classes of data, so that the margins are large as possible. With a **normalized dataset**, these hyperplanes can be described by the equations

$$\begin{aligned} \mathbf{w}^T\mathbf{x} - b &= 1 \\ \mathbf{w}^T\mathbf{x} - b &= -1 \end{aligned} \tag{6.7}$$

Geometrically, the distance between these two hyperplanes is $\frac{2}{||w||}$, so, to maximize the distance between the planes we need to minimize $||w||$.

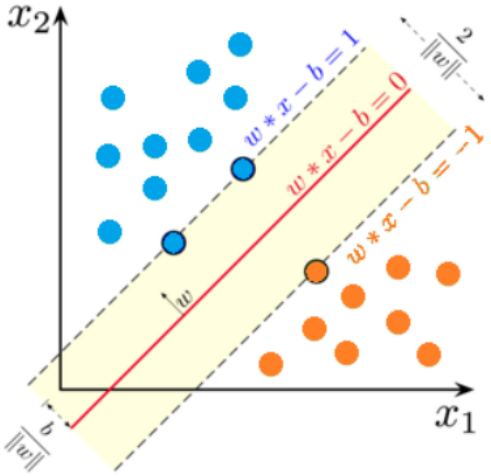

Figure 6.9: SVM graphically: the samples that lie exactly on the border of the margins are called supports, this is the reason of the name of the model.

We also have to impose that each data point must lie on the correct side of the margin:

$$y_i(\mathbf{w}^T\mathbf{x}_i - b) \geq 1 \ \forall \quad i \in [1, n\_samples] \tag{6.8}$$

Note: class labels y_i are in set {-1,1}.

We can put these together to retrieve $\mathbf{w}$ and b trough the optimization problem:

$$\begin{gathered} min||\mathbf{w}||^2 \ s.t \\ y_i(\mathbf{w}^T\mathbf{x}_i - b) \geq 1 \ \forall \quad i \in [1, n\_samples] \end{gathered} \tag{6.9}$$

An important consequence of this geometric description is that the max-margin hyperplane is completely determined by those $\mathbf{x}_i$ that lie nearest to it. These $\mathbf{x}_i$ are called **support vectors**, hence the name of the model.

At inference time we need to retrieve the sign of the output:

$$\hat{y} = sign(\mathbf{w}^T\mathbf{x} - b) \tag{6.10}$$

Since a "linear separation hyperplane" may not exist, the SVM has two extensions that allow the model to be useful in these cases as well.

The first extension is called **soft-margin**, it allow the model to makes some "mistakes" and tries to find the best compromise between, the objective of maximize the margin, and the one of minimize the "mistakes" that (in most application) are quantified trough the **Hinge-loss**, the optimization problem become:

$$min||\mathbf{w}||^2 + C \sum_{i}^{n\_sample} max[0, (1 - y_i\mathbf{w}^T\mathbf{x}_i)] \tag{6.11}$$

- The blue part maximize the margin.
- The red part is the Hinge Loss and penalize vectors that make "mistakes".

The second extension is based on **Kernel methods**, these consist on remapping the data in another higher-dimensional space, without ever computing the coordinates of the data in that space, but rather by simply computing the inners products between the images of all pairs of data in the new space. This operation is often computationally cheaper than the explicit computation of the coordinates. This approach is called **Kernel trick** instead the remapping functions are called **Kernel functions**. We can use the Kernel trick by working with the lagrangian relaxation of the problem, we use the fact that if α (the parameter of the constraints in the relaxed problem) is fixed the optimization problem respect to $\mathbf{w}$ is unconstrained and the objective differentiable:

$$\mathbf{w} = \sum_{i}^{m} \alpha_i y_i \mathbf{x}_i \tag{6.12}$$

The dual problem becomes a problem in which only inner products between instances are involved and we can replace these with Kernel function:

$$K(\mathbf{x}, \mathbf{x}^{'}) =< \phi(\mathbf{x}), \phi(\mathbf{x}^{'}) > \tag{6.13}$$

And at inference time we have:

$$\hat{y} = sign[(\sum_{i}^{m} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}^{'})) - b] \tag{6.14}$$

From the 6.11 we see that we have an explicit hyper-parameter C, the regularization terms, but if we want to use the kernel method we can also tune the kernel function and a parameter inside it:

- C: Regularization term, determines the trade-off between increasing the margin size and ensuring that the sample lie on the correct side of the margin, obviously is useless if the data are linearly separable.
- Kernel function: in literature there are several possible kernel functions, these remap the data in another space that hopefully separate better the samples.
- Gamma: is the Kernel coefficient for Radial Basis Function Kernel (rbf), Polynomial Kernel (poly) and Sigmoid Kernel. If setted to 'scale' uses $1/(n_{features} * Var(X))$ as value of gamma, if setted to 'auto', uses $1/n_{features}$
- Grid used for hyper-parameters tuning:
  - C: (0.1, 0.5, 1, 10, 50)
  - Kernel function: (linear, poly, rbf, sigmoid)
  - Gamma: (scale, auto)

About on how to retrieve a probability with the SVM, a common algorithm is the **Platt scaling** algorithm. This produces probabilities estimates:

$$P(y = 1|f(x)) = \frac{1}{1 + e^{(Af(x)+B)}} \tag{6.15}$$

These are **logistic transformations** of the classifier scores $f(x)$, represented by the signed distance of x from the separating hyperplane. A and B are two scalar parameters that are learned by the algorithm. Briefly, the algorithm create a new data set that has the same labels, but with one dimension (the output of the SVM). Then trains a logistic regression on this new data set, to return the probabilities.

The following table reports the results of three SVM models, computed on test set, trained: with only original features, with CD and original features, with only CD features.

	Best param	Accuracy	Precision		Recall		F1		Support	
			Benign	Malign	Benign	Malign	Benign	Malign	Benign	Malign
SVC (Original feats.)	C: 1; gamma: scale; kernel: poly	0,96	1	0,92	0,93	1	0,96	0,96	82	71
SVC (CD+Original feats.)	C: 1; gamma: scale; kernel: poly	0,96	1	0,92	0,93	1	0,96	0,96	82	71
SVC (CD feats.)	C: 0.1; gamma: scale; kernel: rbf	0,96	0,97	0,95	0,95	0,97	0,96	0,96	82	71

Table 6.4: SVC results: using the "Distance from centroids" results in equal score.

In this case the CD features, not help to improve the score, the reasons could be different. In general the new features not help to find better support vectors. Probably the Kernel method can find a space where the data is already well separated, making the two new attributes useless. However SVC is able to achieve a precision of 1 for benign, this means, as we can see from 6.11, that we don't have false benign, that are more critically for medical applications.

The following scatterplots are made with the model that use CD and original features.

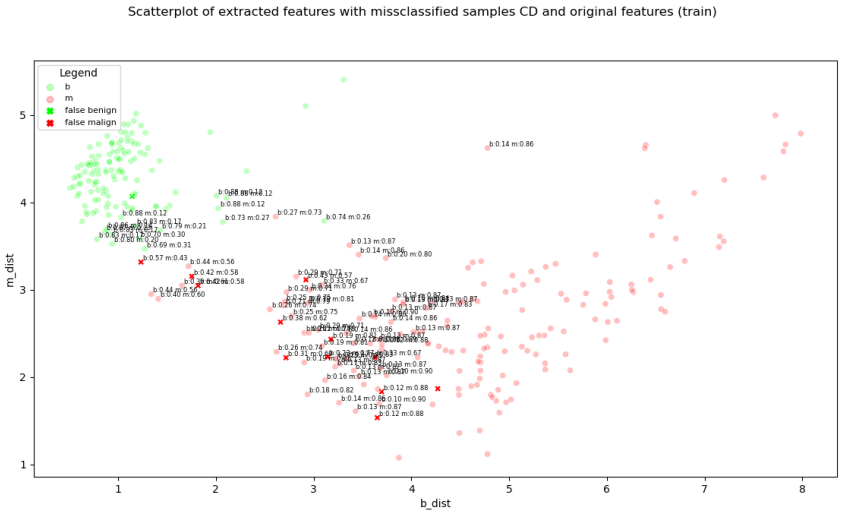

Figure 6.10: Visualization in "CD-space" of the miss-classified samples in train set with the probability to be respectively benign or malign.

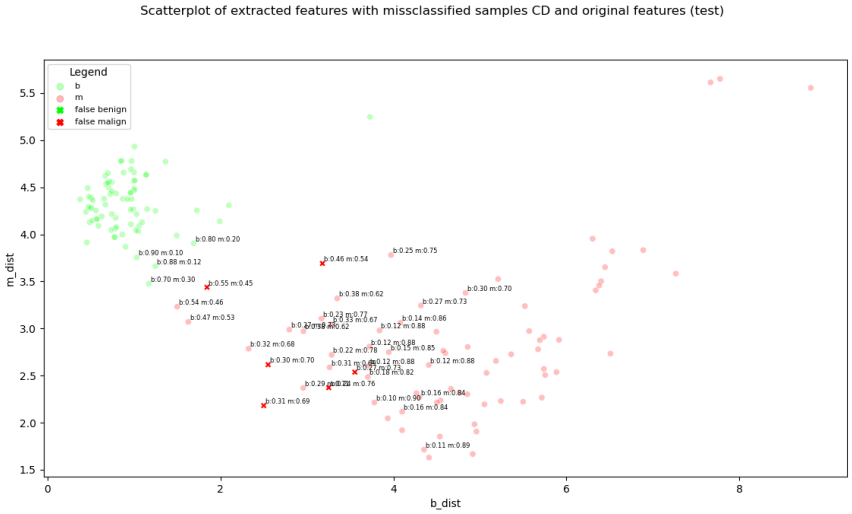

Figure 6.11: Visualization in "CD-space" of the miss-classified samples in test set with the probability to be respectively benign or malign.

Conclusion

To sum-up we have shown how to balance a dataset, manage duplicates, null values, and the importance of standardization. We have analyzed and proposed different technique of data analysis that help to understand and visualize the data, PCA and Correlation Analysis. We have proposed a simple and usefull technique of features extraction, the "Distances from centroids", that is usefull for visualization and improves the final result for the most of models. For each ML model we have explained: some mathematics behind it, and a possible technique to retrieve, with the the predictions, the probabilities, in order to have a sort of confidence index, that is of crucial importance in medical applications. In conclusion we obtain a remarkable accuracy score of 0.96% with the Random Forest classifier and the SVC. Moreover most of the missclassified samples are marked as uncertain, when these have a difference in probabilities ≤ 0.8. In this way, we have a useful instrument that, combined with human analysis, can improve the diagnosis of breast cancer. Unfortunately also many right classified sample are marked as uncertain. For sure, many improvements can be made in this direction, other more refined techniques are avaliable, such as deep learning models that are more suitable when we want to recover the probabilities. To conclude, a further work could be focus on improve the features extraction method by using others techniques to extract "centroids", for example, the "Supervised compact hypersphere" or the "Smallest enclosing ball", instead of centroids simply computed as mean of samples. Moreover others investigations can be done to study better how to avoid the overfitting introduced by the CD method.

Bibliography

Dr. WIlliam H. Wolberg. Breast cancer wisconsin (original) data set, 1992. URL https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(original).

Asoke K. Nandi Tingting Mu. Breast cancer diagnosis from fine-needle aspiration using supervised compact hypershperes and establishment of confidence of malignancy, 2008. URL https://www.eurasip.org/Proceedings/Eusipco/Eusipco2008/papers/1569100880.pdf.

Alessandro Emmanuel Pecora. An exploration of ml methods for breast cancer classification, 2022. URL https://github.com/alessandropec/ML_breast_cancer.git.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. URL https://scikit-learn.org/stable/index.html.