Mathematics in Machine Learning Breast Cancer Dataset

Adele de Hoffer

The importance of an early diagnosis

Breast cancer is one of the most common cancer in women in the United States
It occurs when cells in the breast begin to grow abnormaly

In many case an early stage detection is the key for the patient's surviving, for this reason it is important to have tools that are able to identify these masses at early stages.

The aim of this project is performing data mining tecniques to correctly identify these masses as benign or malign

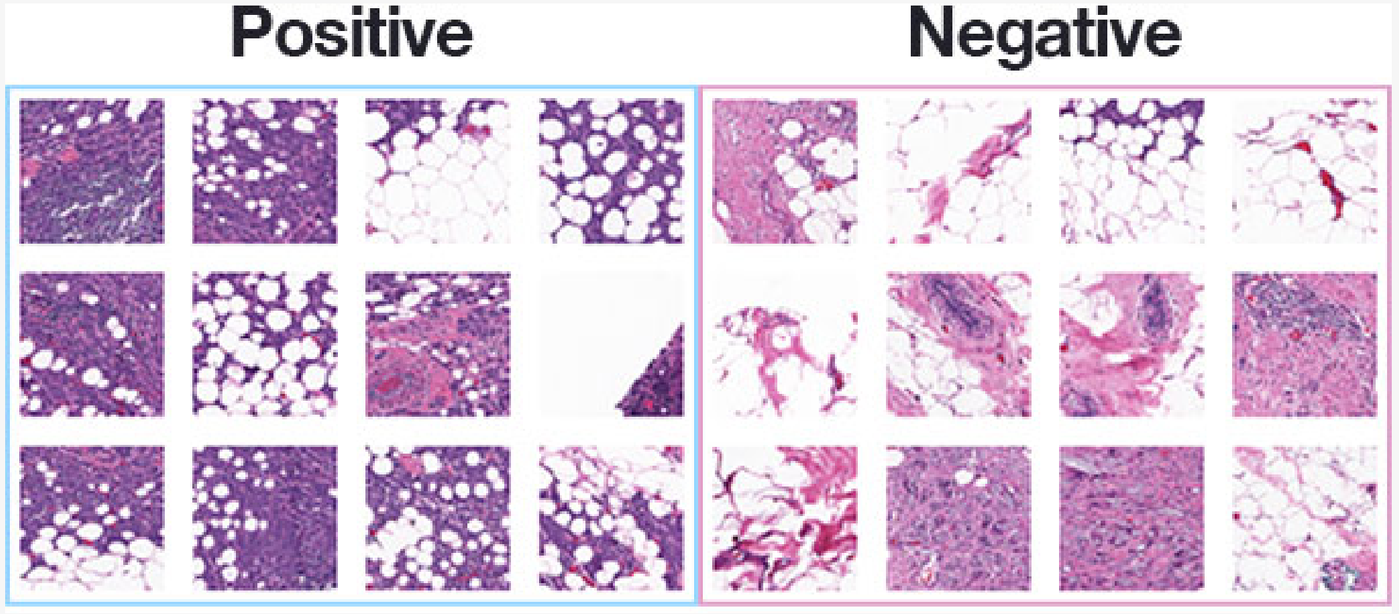
Dataset

1. **Radius** (mean of distances from center to points on the perimeter)
2. **Texture** (standard deviation of gray-scale values)
3. **Perimeter**
4. **Area**
5. **Smoothness** (local variation in radius lengths)
6. **Compactness** (perimeter^2 / area - 1.0)
7. **Concavity** (severity of concave portions of the contour)
8. **Concave points** (number of concave portions of the contour)
9. **Symmetry**
10. **Fractal dimension** ("coastline approximation" - 1)

569 digitized images of fine needle aspirate (FNA) of breast masses

For each image and measure :

Mean StandardDev Worst Measure



	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mea
0	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.300
1	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.086
2	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.197
3	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.241
4	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.198
...	...	...	...	...	...	...	...	
564	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.243
565	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.144
566	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.092
567	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.351
568	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.000

Features type

Faetures	Measures	Type
Diagnosis	/	categorical B/M
Radius	Mean Standard Erros Worst	float
Texture	Mean Standard Erros Worst	float
Perimeter	Mean Standard Erros Worst	float
Area	Mean Standard Erros Worst	float
Smoothness	Mean Standard Erros Worst	float
Compactness	Mean Standard Erros Worst	float
Concavity	Mean Standard Erros Worst	float
Concave points	Mean Standard Erros Worst	float
Symmetry	Mean Standard Erros Worst	float
Fractal dimension	Mean Standard Erros Worst	float

Null values

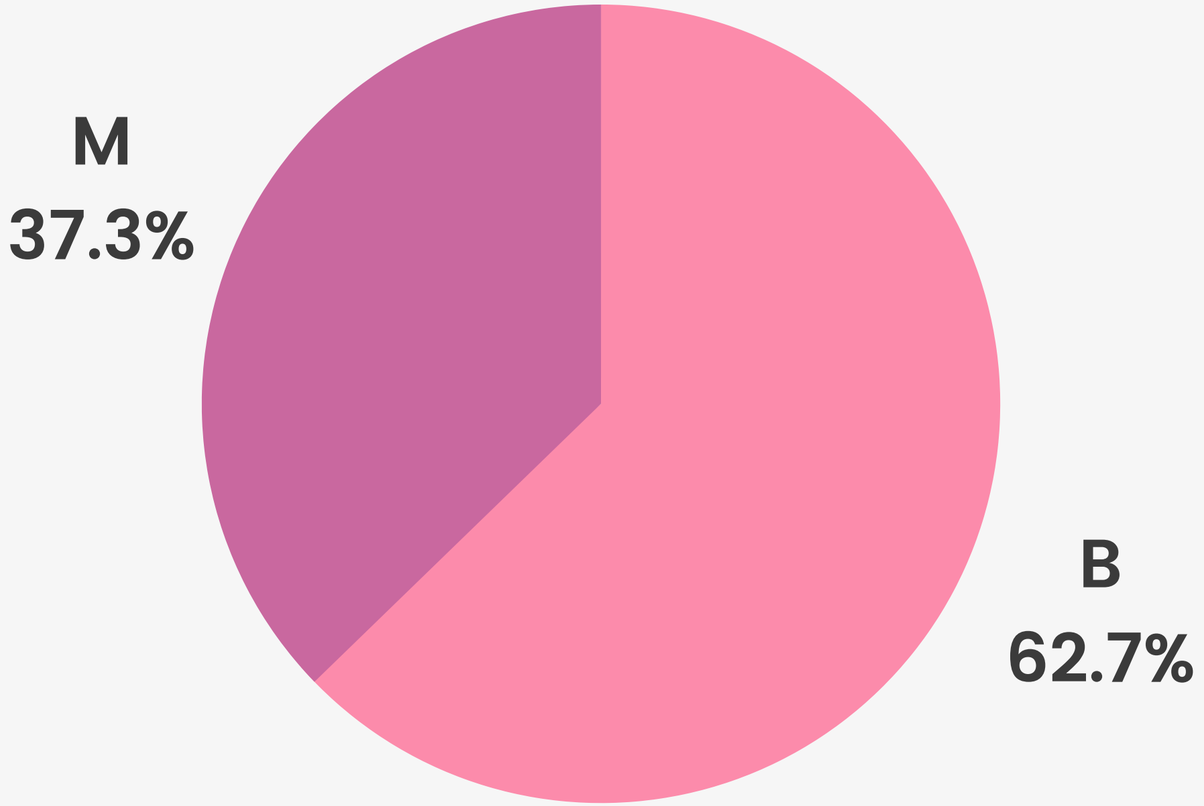
Removing null values is an important step in order to not have useless information

.

In this case we can or remove missing values or fix them using the mean of the feature associated to them.

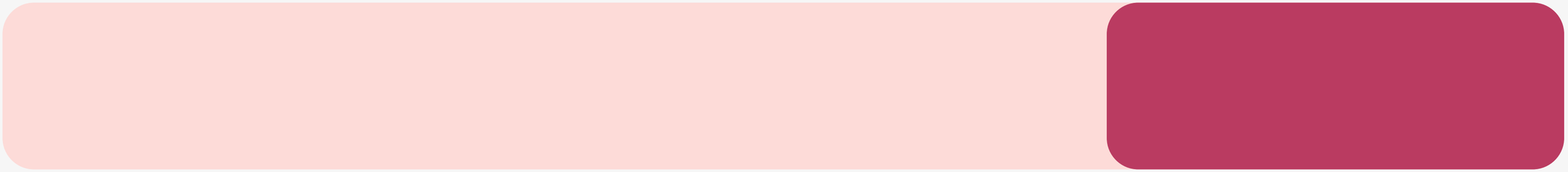
Luckily, this dataset does not present missing values, so we can proceed with the analysis!

An unbalancing problem



Train test split

Stratify + Shuffle



Train 80%

Preprocessing
Distribution
Training
Validation

Test 20%

Look at this just at the end!
Test the model

Standardization

Values for different features differ a lot !
They need to be standardized in order to not have features that weights more only because their range is bigger.

Standard Scaler:
for each sample we substract the mean and divide for the standard deviation.

$$z = \frac{x - \mu}{\sigma}$$

Now each features have
0 mean and unit variance

Normality test

It is useful to check if the data distributions come from a normal distribution, in order to use the right algorithms.

To do so, a Saphiro-Wilk test is performed

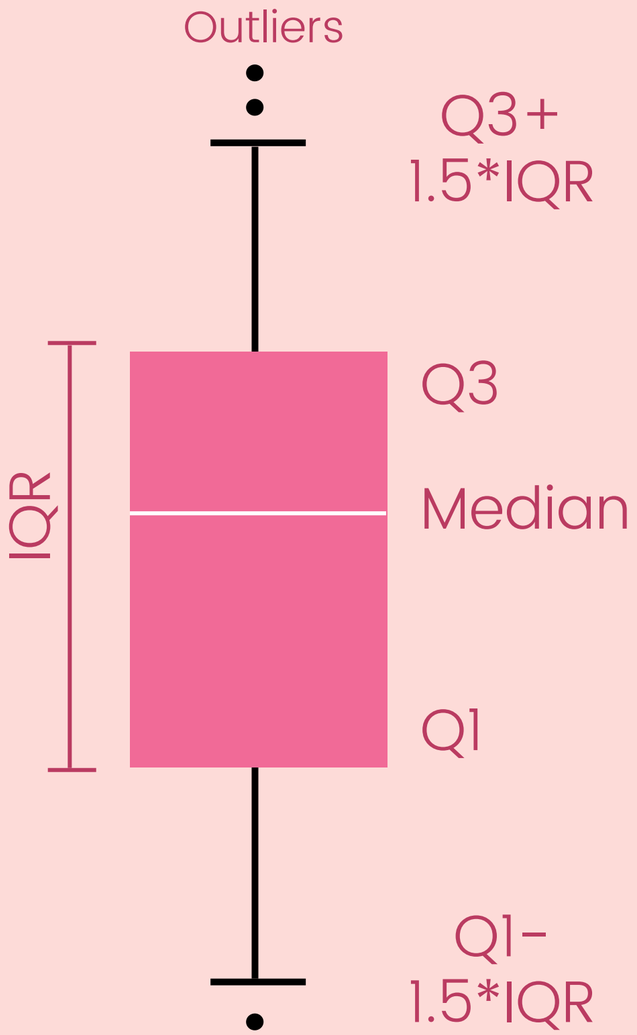
- **Null Hypotesis**: The data have a normal distribution
- **Alterntive Hypotesis**: The data have NO a normal distribution

The test statistic is: $W = \frac{(\sum_{i=1}^{N} a_i x_{(i)})^2}{\sum_{i=1}^{N}(x_i - \bar{x})}$

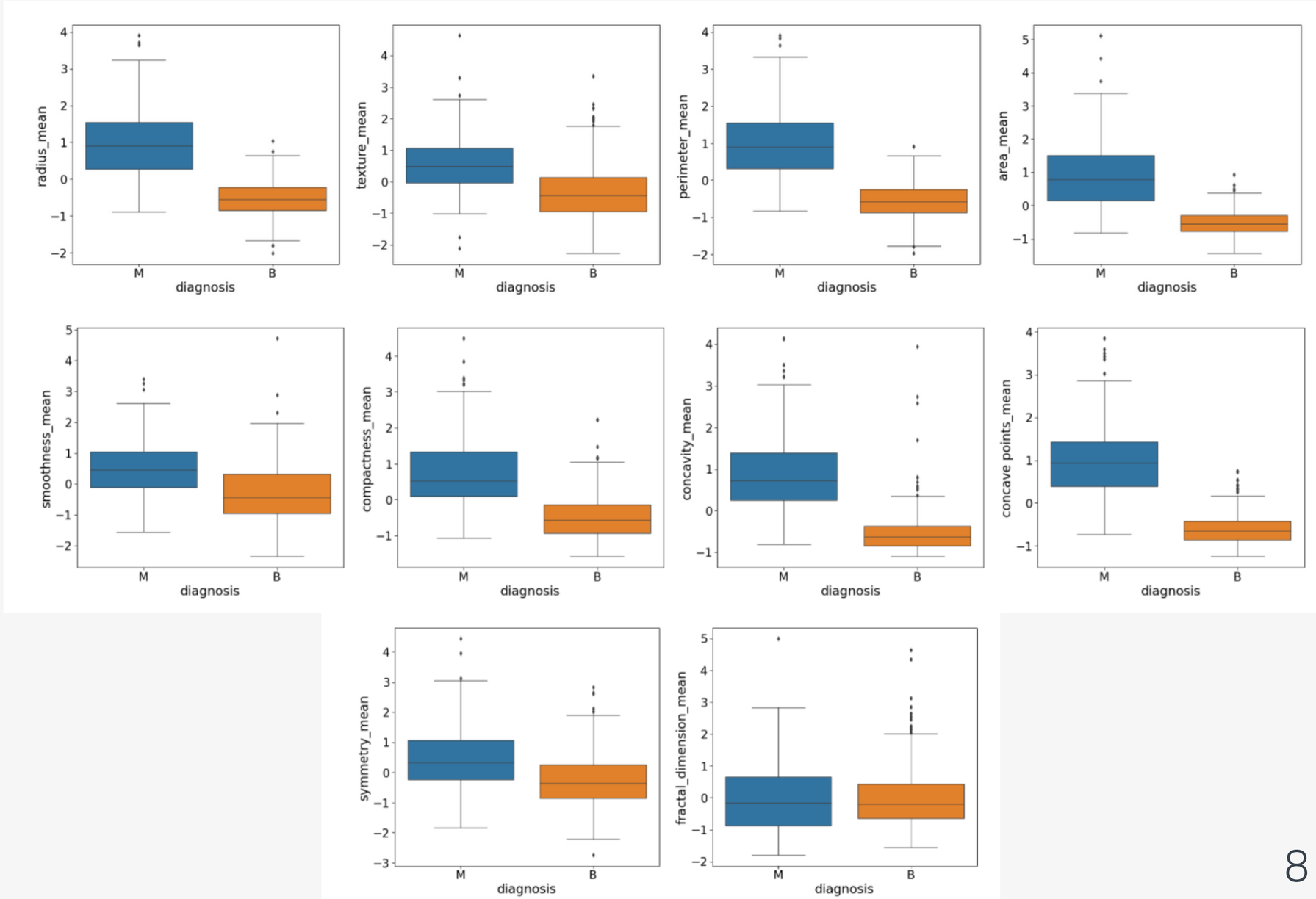
where $x_{(i)}$ is the i-th smallest number in the sample, $\bar{x}$ is the sample mean and a_i are coefficients derived from a normal distributions. The value of α is set to 0.05 in this way:

- if $p \leq \alpha$ we reject the null hypotesis that the distribution is normal
- if $p > \alpha$ we accept the null hypotesis that the distribution is normale

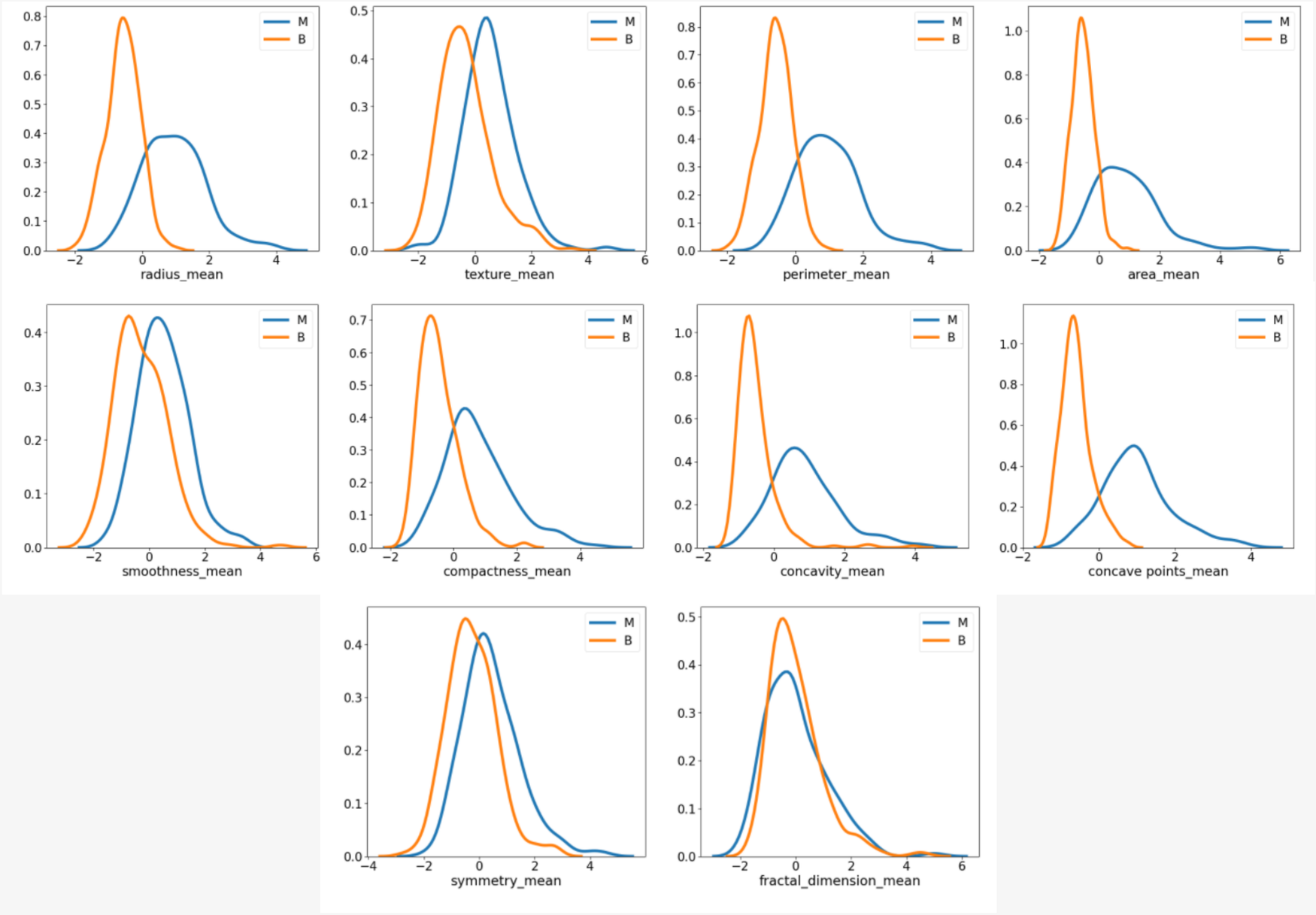
None of the features' distribution is normal



Looking at the distributions.. Boxplots!



Distributions

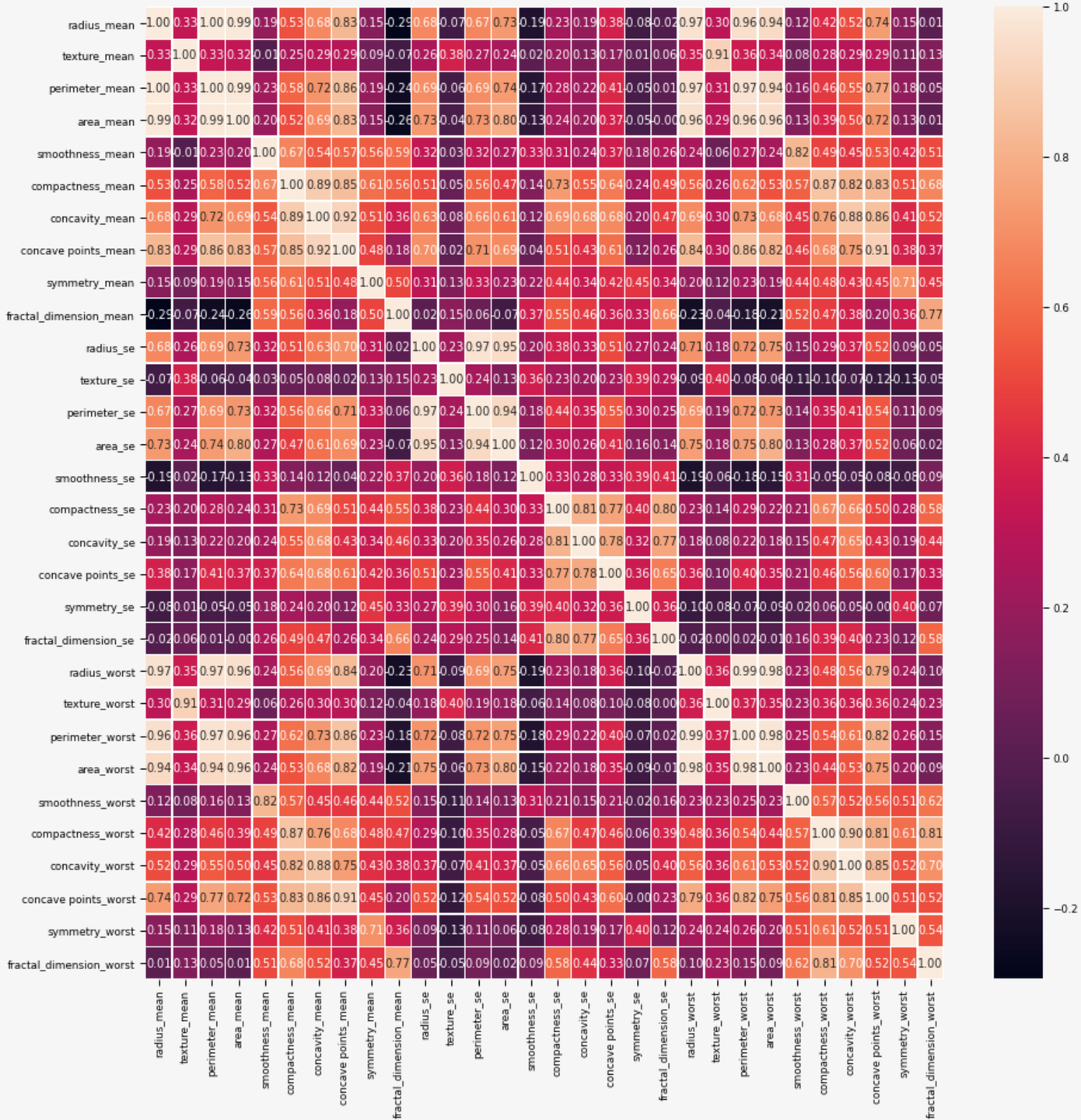


Correlation Analysis

Pearson correlation index
is a measure of linear correlation between two sets of data. Correlated features are reduntant information that may lead to overfit for this reason it is suggested to remove them

$$\rho_{x,y} = corr(X,Y) = \frac{E[(X-\mu_x)(Y-\mu_y)]}{\sigma_x \sigma_y}$$

- Positive coefficient:
as one measure increases the other one increases as well
- Negative coefficient:
that as one measure increases the one one decreases
- Null coefficient:
does not add any information



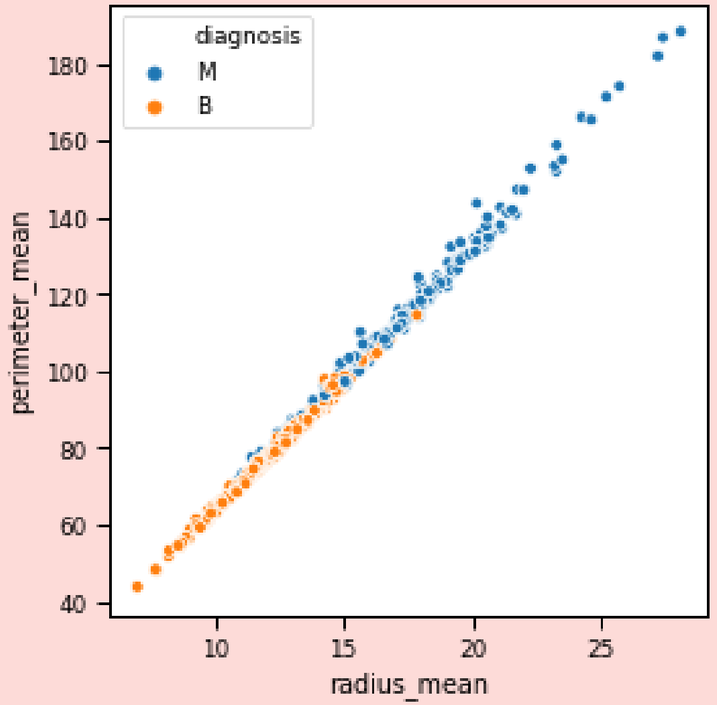
radius_mean is positive correlated with perimeter_mean,area_mean, radius_worst, perimeter_worst and area_worst

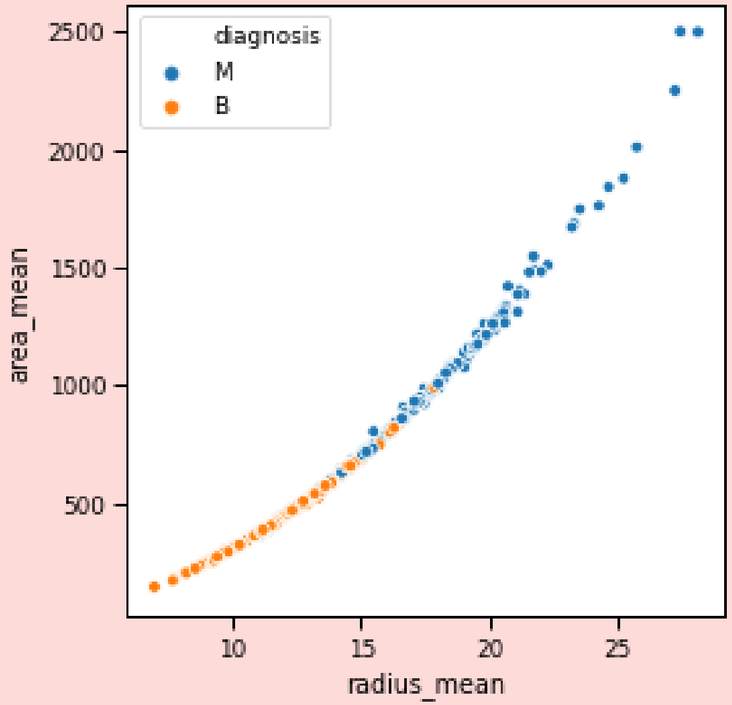
texture_mean is positive correlated with texture_worst

concavity_mean is positive correlated with concave points_mean

radius_se is positive correlated with perimeter_se and area_se

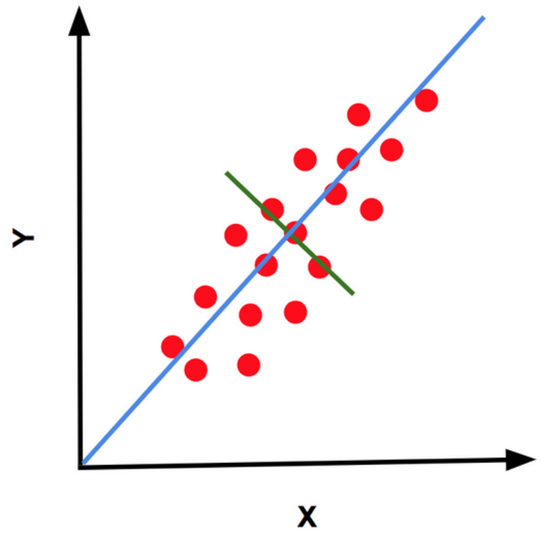
So we can discard perimeter_mean,area_mean, radius_worst, perimeter_worst and area_worst, texture_worst,concave points_mean, perimeter_se and area_se





PCA is a dimensionality reduction tool that aims to reduce the dimensionality of a matrix **with losing less information as possible.**

Concretely it is a linear dimensionality reduction approach that performs $x \rightarrow Wx$ where $W \in R^{mxk}$ and $k < m$



We can map each instance in this way:

$$z_i = a_1\phi_1 + a_2\phi_2 + .. + a_k\phi_k + a_{k+1}\phi_{k+1} + ... + a_m\phi_m$$

The new variables are a linear combination of the original features and form an ortonormal basis which define a new coordinate system

The ϕi are called loading vectors and are the component along which the data vary the most, ai are the weight associated to each leading direction and represent the projection of the original instance on that vector, and cut this mapping to k-th values
How do we define ϕ?

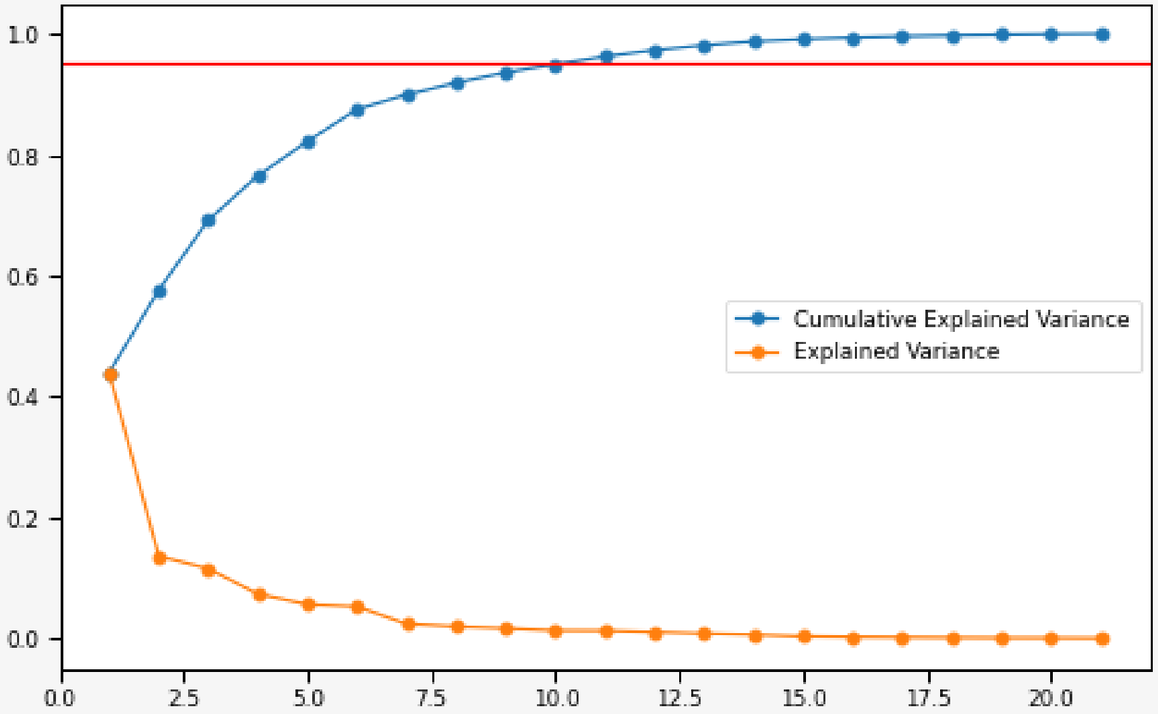
We can use the linear recovery vector y =Wx
Which means x reconstructed = Uy=UWx
So at the end it is an optimization problem

$$\arg \min_{W \in R^{n,d} U \in R^{d,n}} \sum_{i=1}^{m} ||x_i - UWx_i||^2$$

The solution is to set U to be the leading eigenvectors of A=∑x∗x which is the correlation matrix

We can decompose the total variance: we can see the percentage of the information that is expressed by each component
In fact, the first component ϕ1 retrieve most part of the information, the component ϕ2 the second part and so on.

Using only 10 dimensions we explain the 95% of dataset



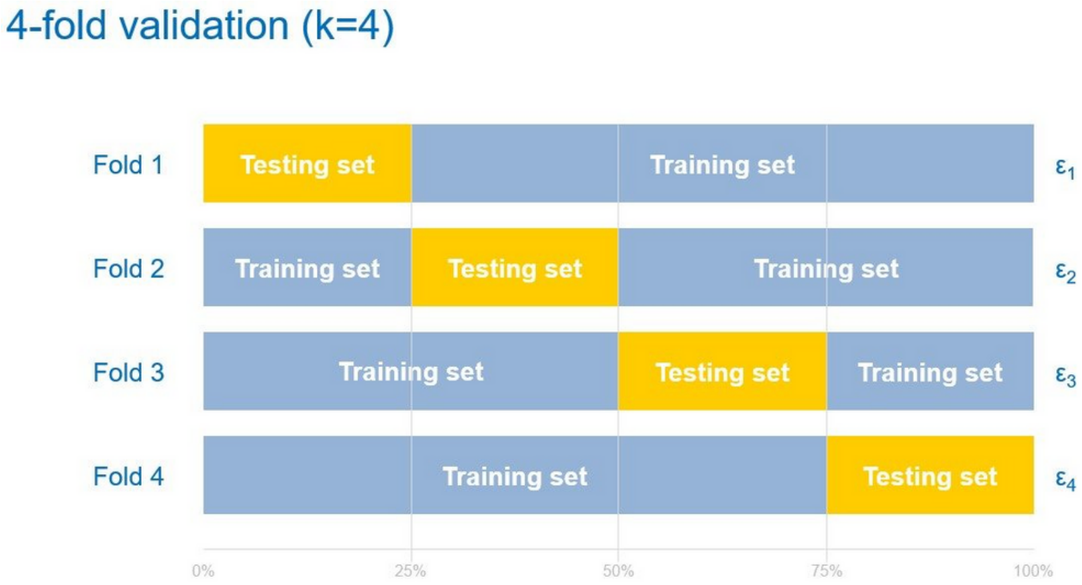
Split again but...

The train set needs to be divided again!

- a real **train set** (on which the algorithm) is trained
- a **validation set** (where the algorithm is validate in order to select the right hyperparameters)

We might risk to overfit by selecting the parameter that fits better the validation set
The model may depend too much on the dataset splitting!

Solution.. Cross Validation!



To avoid overfitting on the validation set it is better to perform
Cross Validation
The dataset is divided into k subsets of equal dimension,
For k times:

- k-1 parts are used to train the algorithm
- 1 part is used to validate the model

Result by averaging

Measures

Accuracy	Recall	Precision	f1
Number of corrected predictions over all the predictions $\frac{TP+TN}{TP+TN+FP+FN}$	Number of corrected true over all the real true $\frac{TP}{TP+FN}$	Number of corrected true over all the elements predicted as true $\frac{TP}{TP+FP}$	Harmonic average of Recall and Precision

For diagnostic data it is more useful to consider the recall it is better to predict one false positive rather than one false negative!

AN UNBALANCE PROBLEM

The percentage of the label is unbalanced
Very good results in terms of metrics but it can derive just from the biggest class

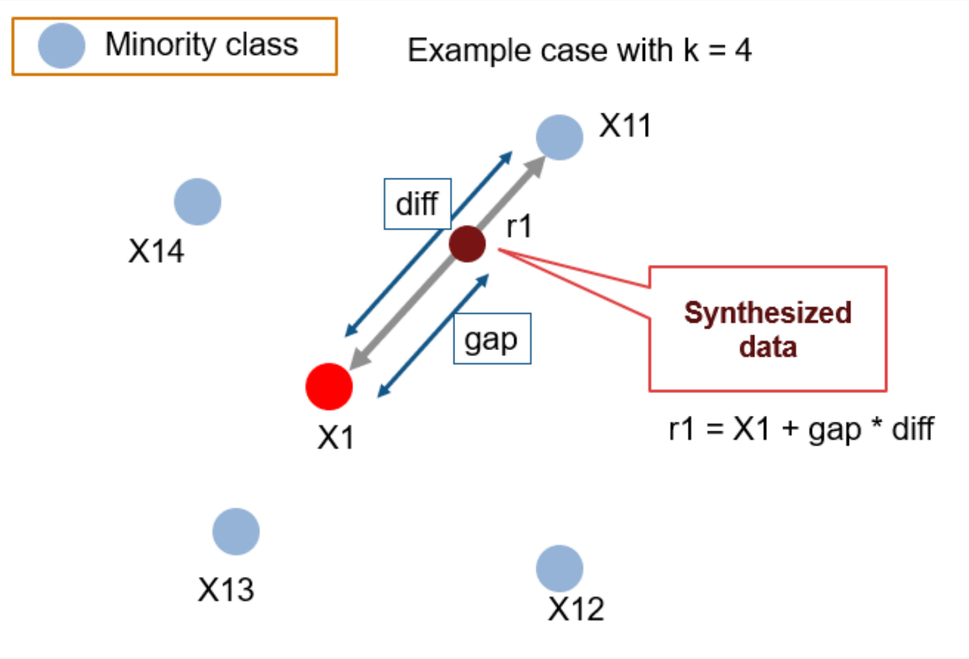
We risk to have poor performance on the minority class, although it is the most important

Undersampling: Rremoving random samples from the most populated class..
..but we are removing information!

Oversampling: Sampling randomly from the less populated class
Just a replica of information

Synthetic Minority Oversampling Technique: Data augmentation tecnique that sintetyzes new data from the less populated class. To do so we firstly select a sample and then connect it through k-lines to its k-neighbors.

The new element is selected randomly on the line that connects the original sample with each of its k-neighbors.



Undersampling + SMOTE: Undersampling firstly the most populated class and then perform SMOTE tecnique on the other one.

We perform Undersampling ratio at 0.7 and k=5 for the SMOTE tecnique.

However, it is important to be sure that the **test and the validation set do not contain an already known value**, for this reason this procedure is applied only on the train set
Besides it is important for the test set to be more similar to the reality as possible!!

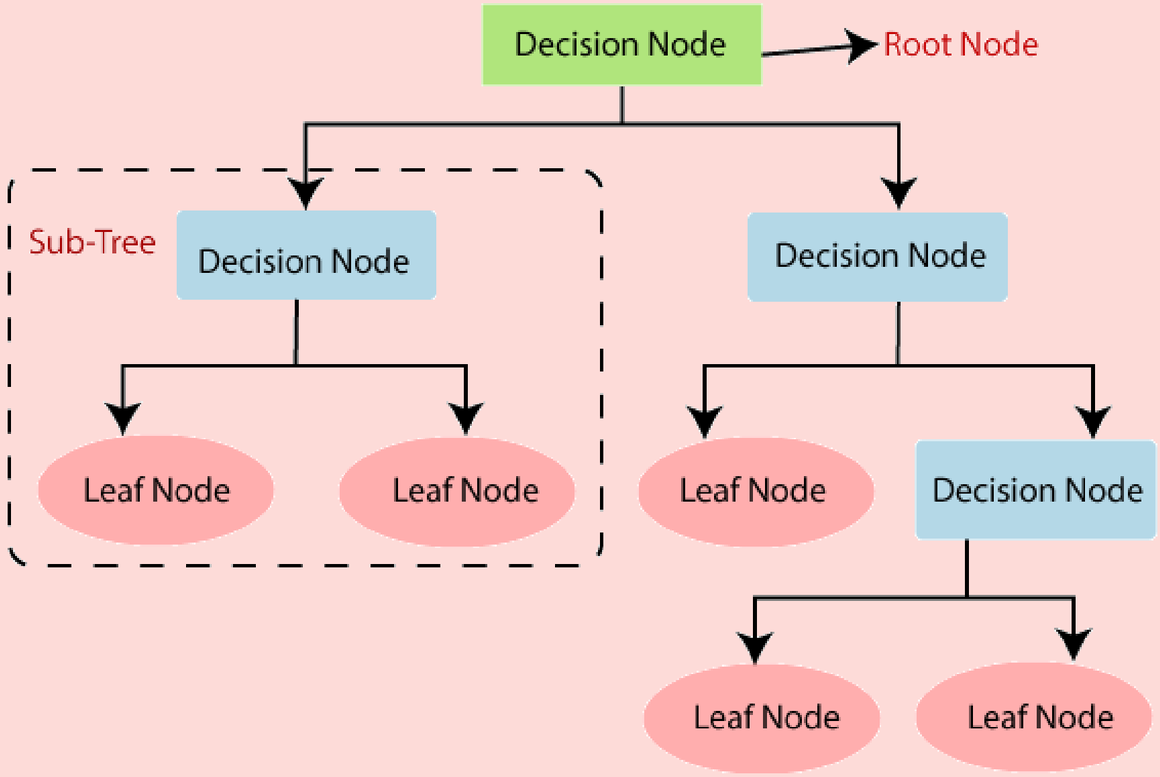
DECISION TREE

Decision tree is a classification algorithm based on decision rules.
The procedure is greedy and based on the pureness of the splitting: at each split the model select the attribute that performs the 'best split'.
The best split means that the two subgroups created after the split must maximes:

Gini index **Cross Entropy**

Building the tree is a recursively splitting based on the attribute that better divide the dataset.
It tends to overfit the data, so we need to insert some **stop condition** to the growning of the tree

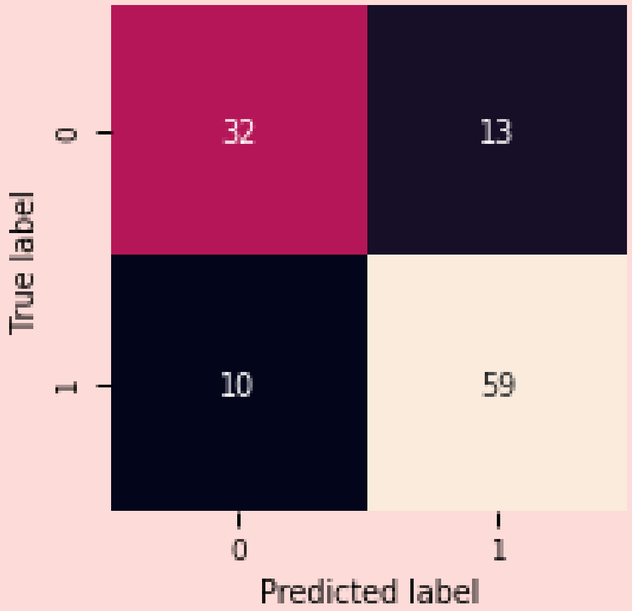
- **Maximum depth** of the tree
- **Minimum samples** in leaf nodes
- Minimum reduction in the **error metric**



The beginning of the tree is called **root node** and represents the beginning of the analysis, each internal node represents a split on an attribute, so **at each node the dataset is split in 2 parts based on one attribute at time.**
The ending nodes are called leaves and represent the predicted classes
Each outcome can be described as a chain of decision rules

```
params = {
            'criterion' :  ["gini", "entropy"],
            'splitter' :  ["best", "random"],
        'max_depth' :  [None, 1, 3, 5, 8, 10, 12, 15],
        'max_features' :  ['auto', 'sqrt', 'log2'],
    'min_samples_split' :  [2, 5, 7, 10, 11, 14, 15]
                                                    }
```

The best performaces are achieved with:
criterion: gini
max_depth: None
max_features: sqrt
min_samples_split: 10
splitter: best



Recall score:
0.86
Accuracy score:
0.80
Precision score:
0.82
f1 score:
0.84

BAGGING

Theory: averaging a set of observations reduces variance!

Given a set of B observations of variance **σ**.

The mean of the observations is

$$f_{bag} = \frac{1}{B}\sum_{b=1}^{B} f^b(x)$$

The variance of the mean of the observation is **σ**/B

k datasets are created by sampling uniformly and with replacement from the orginal dataset

Each dataset is used to train an indipendent algorithm

Random sampling allows us to train algorithms that are indipendent from each other

The output of each model is then used to make the prediction by averaging i

It can be performed in parallel

RANDOM FOREST

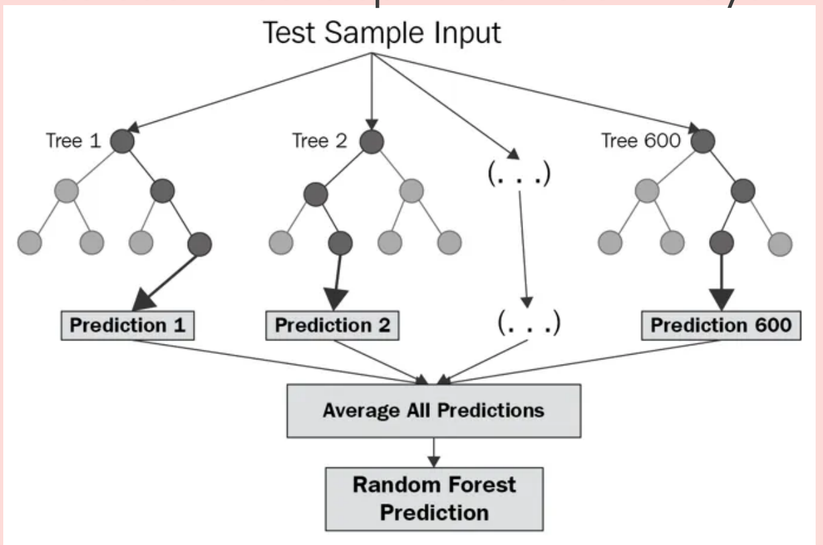
Random forest is an ensemble tecnique that uses a large number of uncorrelated decision trees.

Each of them predicts a result, then all of them are averging in a majority voting.

Each decision tree is fed with random samples of the original dataset

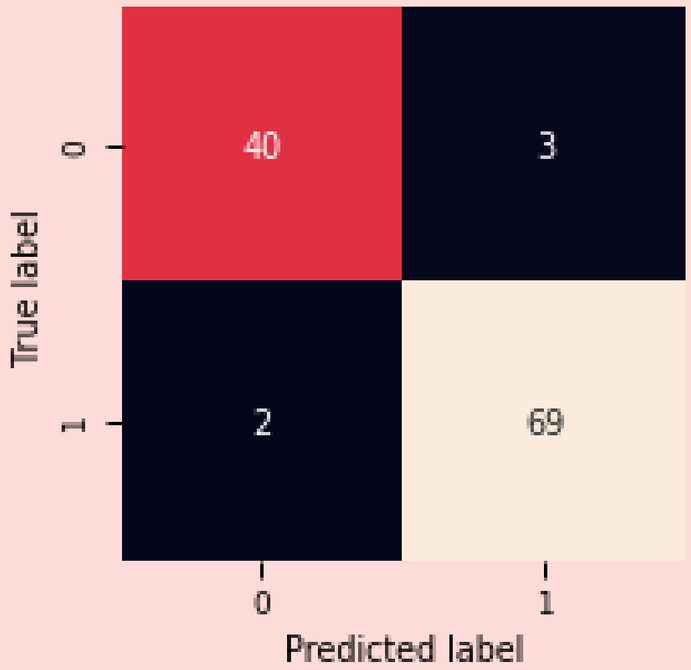
Uncorrelation: also different features are selected for the splitting procedure

Each decision tree is trained on a different set of the dataset and so each tree is split in different ways



```
params = {
        "criterion": ["gini", "entropy"],
    "n_estimators": [1, 10, 20, 50, 100],
        "max_depth": [5, 10, 15],
"max_features": ["auto", "sqrt", "log2"],
        "random_state": [1]
                                }
```

The best performaces are achieved with:
criterion: entropy
max_depth: 15
max_features: sqrt
n_estimators: 50
random_state: 1



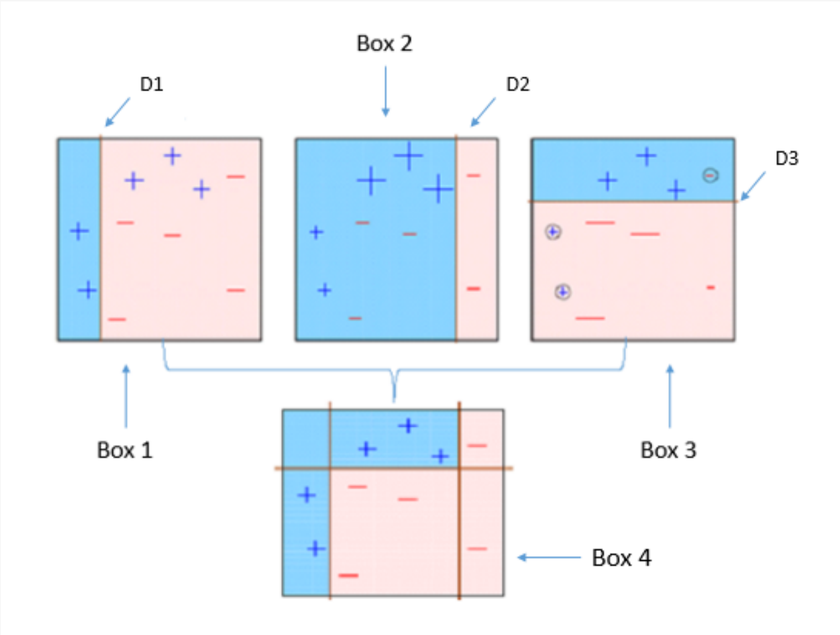
Recall score:
0.97
Accuracy score:
0.96
Precision score:
0.96
f1 score:
0.97

BOOSTING

Reduce the vaiance but also the bias

Fitting phase is consequential
NO parallel computation

At each step the model iteratively gives more importance to the samples that are misclassified in the previous step



ADABOOST

Create a strong learner from a series of weak learner
It takes as input the original dataset at time and proceeds for T iterations.

At each iteration t from 0 to T:

- A distribution D(t) is selected over the samples S
- The weak algorithm is trained to return an hypothesis
- It is assigned a weight to this distribution that is inversely proportional to the error

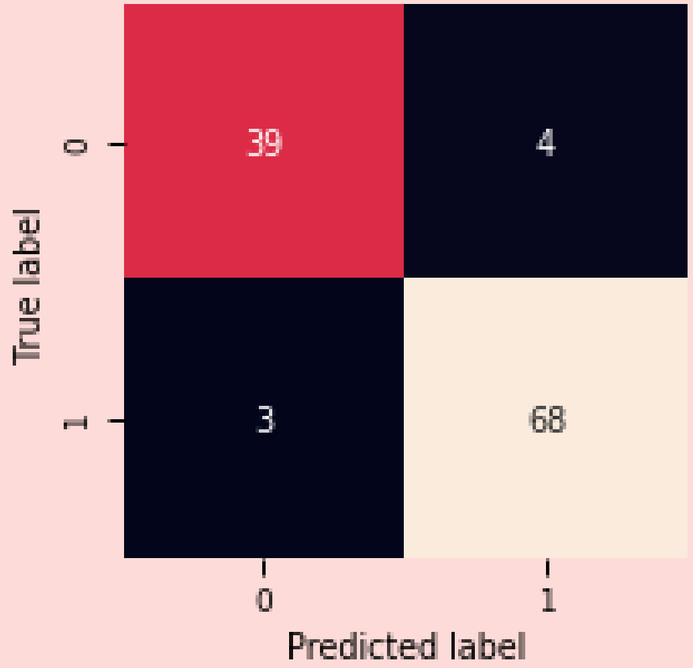
Samples that are miscassified will have a heavier weight

The learner is forced to focus its attention to the more difficult samples

```
params = {
        "criterion": ["gini", "entropy"],
    "n_estimators": [1, 10, 20, 50, 100],
        "max_depth": [5, 10, 15],
"max_features": ["auto", "sqrt", "log2"],
        "random_state": [1]
                                }
```

The best performaces are achieved with:

criterion: entropy
max_depth: 15
max_features: sqrt
n_estimators: 50
random_state: 1

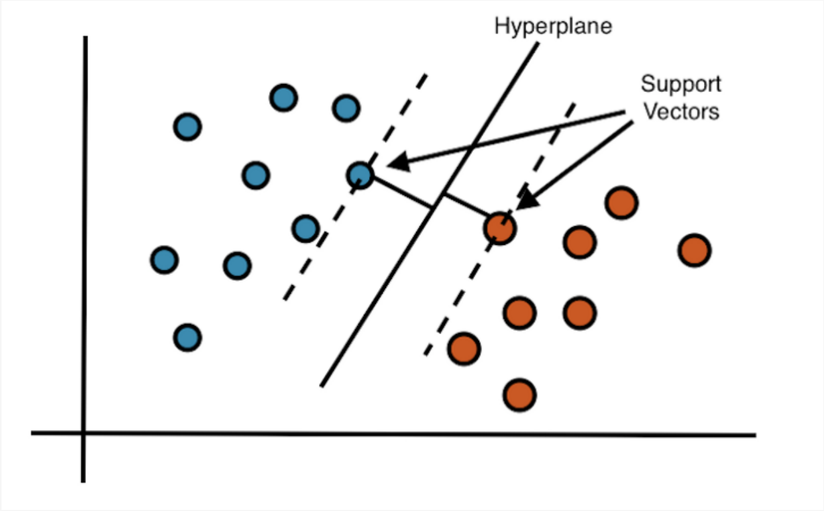


Recall score:
0.97
Accuracy score:
0.96
Precision score:
0.96
f1 score:
0.97

SVM

Support Vector Machine is a classification algorithm that aims to find the best hyperplane that divide the hyperspace containing the samples of different classes

Mazimize the margin!



Hard SVM

$$\arg\max_{(w,b):||w||=1} \min |\langle w, x_i \rangle + b|$$

$$\text{s.t. } \forall i y_i(\langle w, x_i \rangle + b) > 0$$

this assuming that data are linearly separable..

NOT ALWAYS TRUE!

Soft SVM

$$\arg\min_{w,b,\xi}(\lambda||w||^2 + \frac{1}{m}\sum_{i=1}^{m}\xi_i)$$

$$\forall i \quad y_i(\langle w, x_i \rangle + b) > 1 - \xi_i \quad with \quad \xi_i > 0$$

A slack variable is added to overcome the problem

Misclassification is possible paying a price

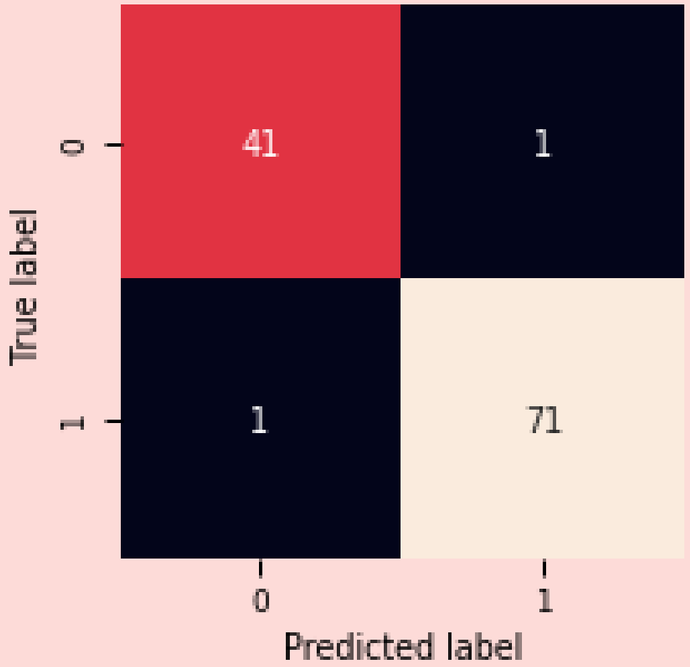
params = {
'C' : [0.0001, 0.01, 0.1, 1, 10, 100, 1000],
'gamma': [0.001,0.01,0.1,1,10,100,1000]

}

The best performaces are achieved with:

C: 0.1

gamma: 0.1

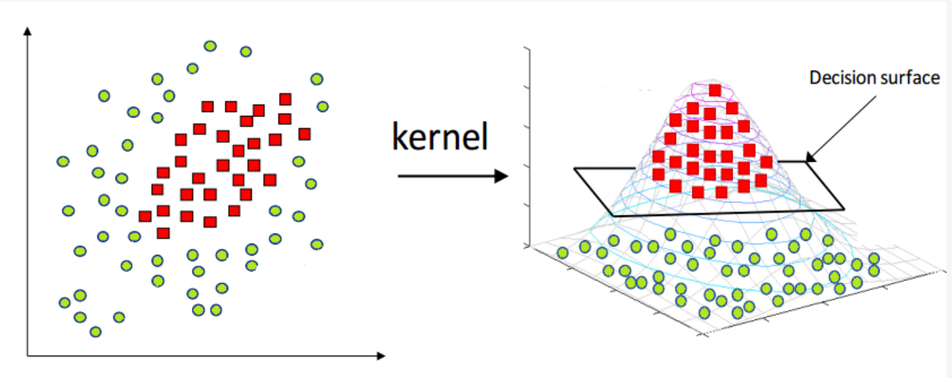


Recall score:
0.99
Accuracy score:
0.98
Precision score:
0.99
f1 score:
0.99

Kernel SVM

It is not always possible to find a hyperplane that linearly divides the classes, for this reason sometimes it is needed to find an higher dimensional feature space in where it is possible to find an hyperplane

Once this space if found, we can linearly divide our samples and then return in the previous space.



The kernel trick

Mapping the features into this higher dimensional space could be very expensive in terms of time and computational power.

The function to optimize only contain the dot product of the features in the higher dimensional space:

$$||w||^2 = \sum_{i,j=1}^{m} \alpha_i \alpha_j \langle \phi(x_i), \phi(x_j) \rangle$$

Instead of applying $\phi()$ at every sample x we can just calculate the pairwise dot product between samples.

$$K(x, x') = \langle \phi(x), \phi(x') \rangle$$

Here Gaussian:

$$K(x, x') = e^{-\gamma ||x - x'||^2}$$

params = {

'C' : [0.0001, 0.01, 0.1, 1, 10, 100, 1000],

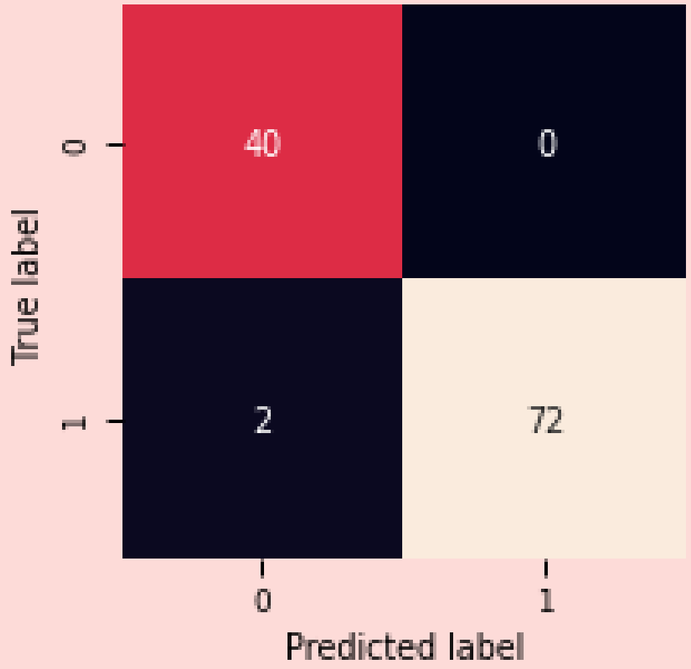
'gamma': [0.001,0.01,0.1,1,10,100,1000]

}

The best performaces are achieved with:

C: 10

gamma: 0.01



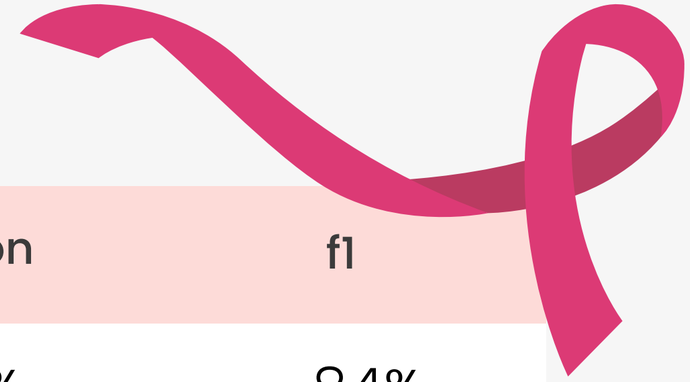
Recall score: 0.97

Accuracy score: 0.98

Precision score: 1.0

f1 score: 0.99

Conclusion



	Recall	Accuracy	Precision	f1
Decision Tree	86%	80%	82%	84%
Random Forest	97%	96%	96%	97%
AdaBoost	97%	96%	96%	97%
Linear SVM	99%	98%	99%	99%
Kernel SVM	97%	98%	100%	99%

Thank you!