Breast cancer classification

Machine Learning project report

A.Xhyra, M.Marino, P.Tropeano

February 2021

Contents

1 Introduction

Breast cancer is the most common type of cancer affecting women and the second one affecting both sexes worldwide; cancer in general is a leading cause of death ($\sim$ 9.6 million cases in 2018) [1]. The way we can reduce the cancer burden is by improving treatments, prevention strategies, lifestyle and early detection. For this reason new technologies and automated techniques must be studied and adopted.

Machine learning and image processing provide tools that may assist pathologists in the diagnostic process but cannot replace them, an hypothesis which scares public opinion. Automated analysis of cancer cell images doesn't only speed up diagnosis, but it also makes it more objective and consistent to a wider knowledge base.

An important step in the image analysis of cancer cells is the segmentation of nuclei. Two popular approaches to nuclear segmentation are *active contour models* and *watershed method*, however learning-based approaches lead to a better generalization [2]. These techniques may be classified into two categories: **methods that use handcrafted features** (e.g. color histograms, color-texture), and **deep learning methods** [3, 4].

This report will examine the application of various techniques to a binary classification problem, in order to correctly label breast tumor cells as benign or malignant. The dataset used is the "Breast Cancer Wisconsin (Diagnostic) Data Set"[1], provided by UCI[2].

[1] Source: `https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29`

[2] University of California, Irvine

2 Features

Features have been extracted from the cell nuclei boundaries (snakes) (Figure 1), found using active contour method. More details on how these boundaries have been determined can be found in [5, 6].

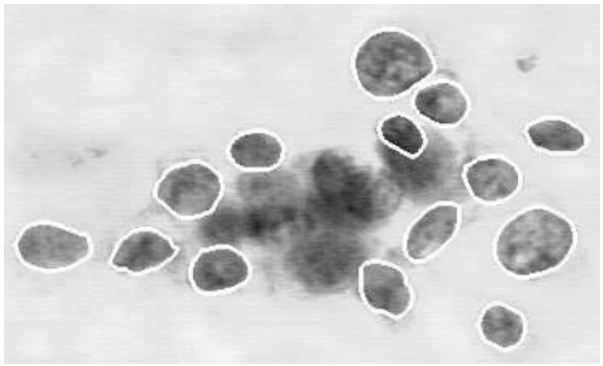

Figure 1: Cell nuclei contours found using active contour technique [6].

The ten extracted features are as follows:

1. Radius
   The nuclear radius is calculated averaging the length of the line segments given by the nucleus centroid and each point of the contour.

2. Perimeter
   The nuclear perimeter is given by the total distance between the contour points.

3. Area
   Nuclear area is the number of pixels within the boundary plus a half of the perimeter pixels.

4. Compactness
   Compactness (1) is calculated as

$$compactness = \frac{perimeter^2}{area}, \tag{1}$$

   though, in literature, this value is commonly divided by 4π. Another typical formulation is given by the reciprocal of this resulting formula.

5. Smoothness
   Smoothness (2) is calculated as the difference between a radial line length and the average length of the two lines surrounding it (Figure 2).

$$smoothness = \frac{\sum_{points} |r_i - \frac{r_{i-1}+r_{i+1}}{2}|}{perimeter} \tag{2}$$

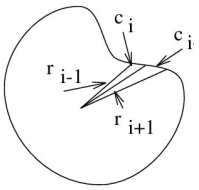

Figure 2: Radial segments used to measure smoothness [6].

6. Concavity
   Chords between non adjacent contour points are drawn, creating a more serrated shape (Figure 3). Concavity is the average distance of snake points within the chords.

Figure 3: Chords used to measure concavity [6].

7. Concave points
   This feature is the number of snake points that lie within the chords drawn to measure concavity.

8. Symmetry
   Symmetry (3) is given by the relative difference between the lengths of each pair of segments perpendicular to the major axis.

$$symmetry = \frac{\sum_i |left_i - right_i|}{\sum_i (left_i + right_i)} \tag{3}$$

9. Fractal dimension
   This features is calculated using the Mandelbrot "coastline approximation" [7].

10. Texture
    The texture is measured as the variance of the gray scale intensity values of pixels.

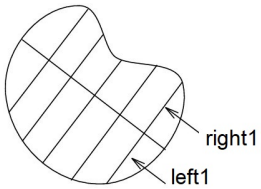

Figure 4: Line segments used to measure symmetry [6].

The dataset also provides the standard error and the worst value for each of these features, for a total of 30 variables. The worst value has been calculated as the average of the three larges values for each image.

3 Exploratory analysis

The first step was to inspect the dataset. It is made of 569 rows and 33 columns:

- an integer column representing the record id;
- a categorical column that labels the record with two possible values ("B" for "benign", "M" for "malignant");
- 30 real-valued columns that represent the features described before (Section 2);
- an extra column named X that only contains NA values.

The *id* and the X columns weren't useful for the purpose of this analysis, so they've been removed. There are no missing values within the remaining columns.

The class distribution appears slightly imbalanced: 357 benign, 212 malignant (Figure 5).

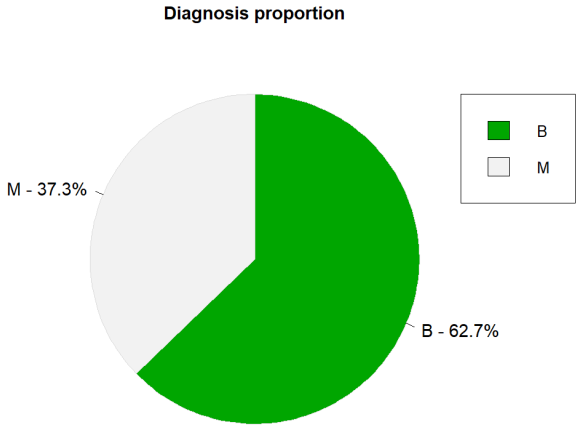

Figure 5: Pie chart of diagnosis classes ("B" 62.7%, "M" 37.3%).

3.1 Feature distribution

The next step was to represent the feature distribution through density plots, in order to gain more insight on data. It comes out that the two classes are not perfectly separable on the basis of the values of individual features (Figure 6, 7). Moreover, by a visual inspection of the feature distributions without distinguishing the two classes, it appears that only some of the features are close to be normally distributed (e.g. *texture_mean*, *smoothness_mean*) (Figure 8, 9); this can be also observed by performing the Shapiro-Wilk test (for every feature, p-value < 0.0001, so normality hypotheses are rejected). Moreover, from the boxplots, it emerges the presence of outliers and the fact that feature ranges vary significantly (Figure 10, 11). However, no transformation has been applied on data to better fit normal distributions and to reduce the number of outliers.

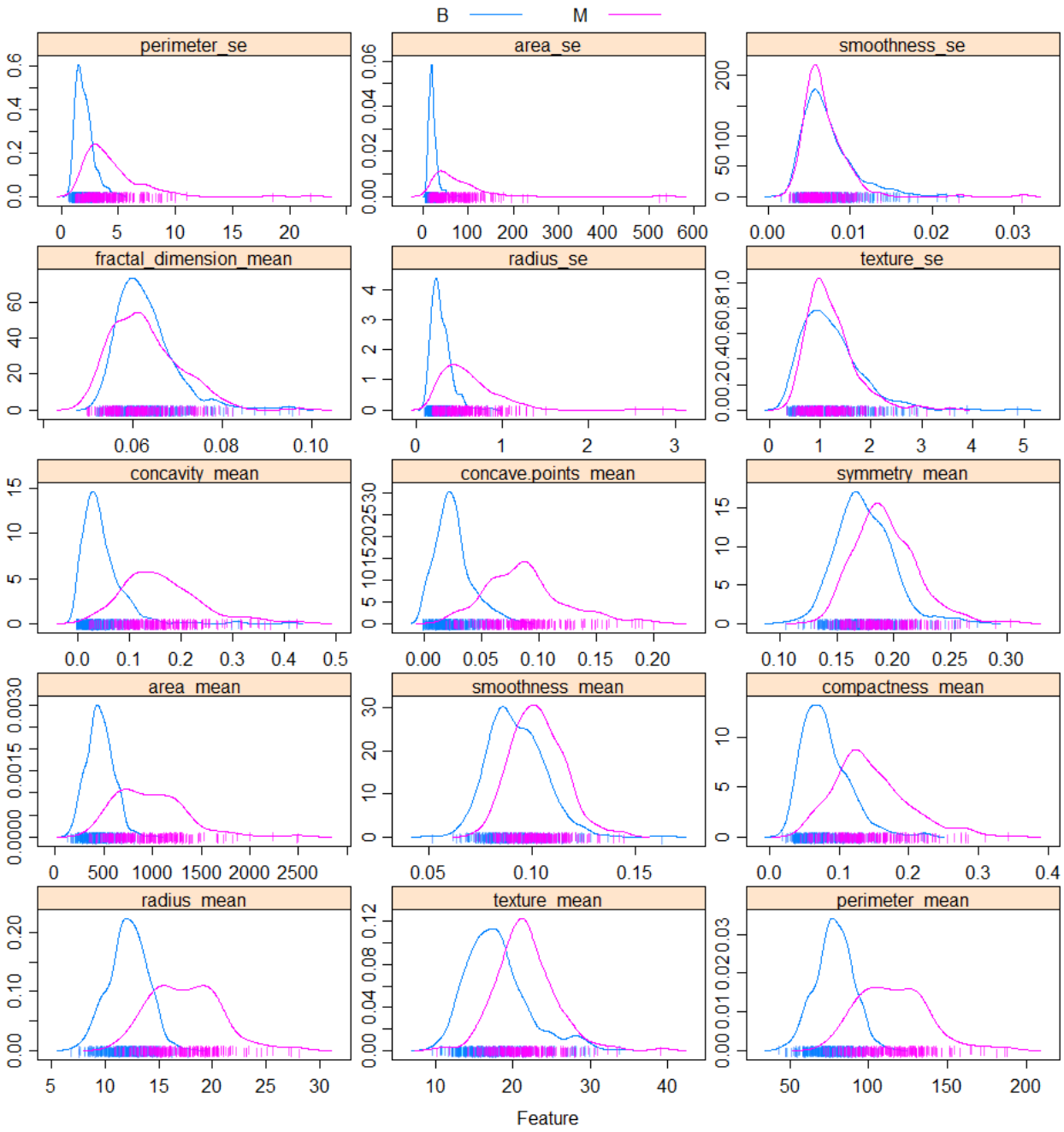

Figure 6: Density plots of the first 15 features distinguishing the two classes ("B", "M").

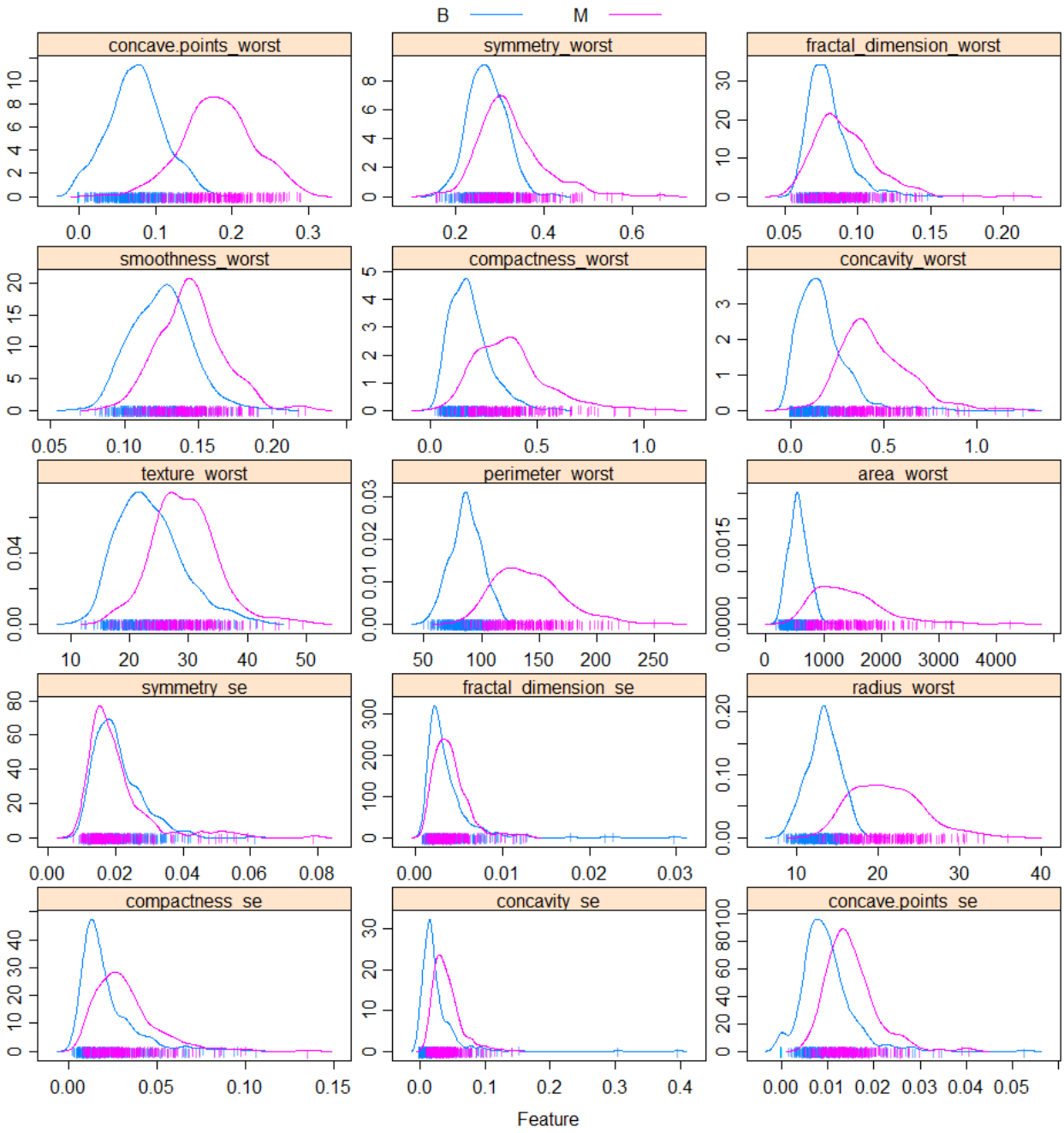

Figure 7: Density plots of the last 15 features distinguishing the two classes ("B", "M").

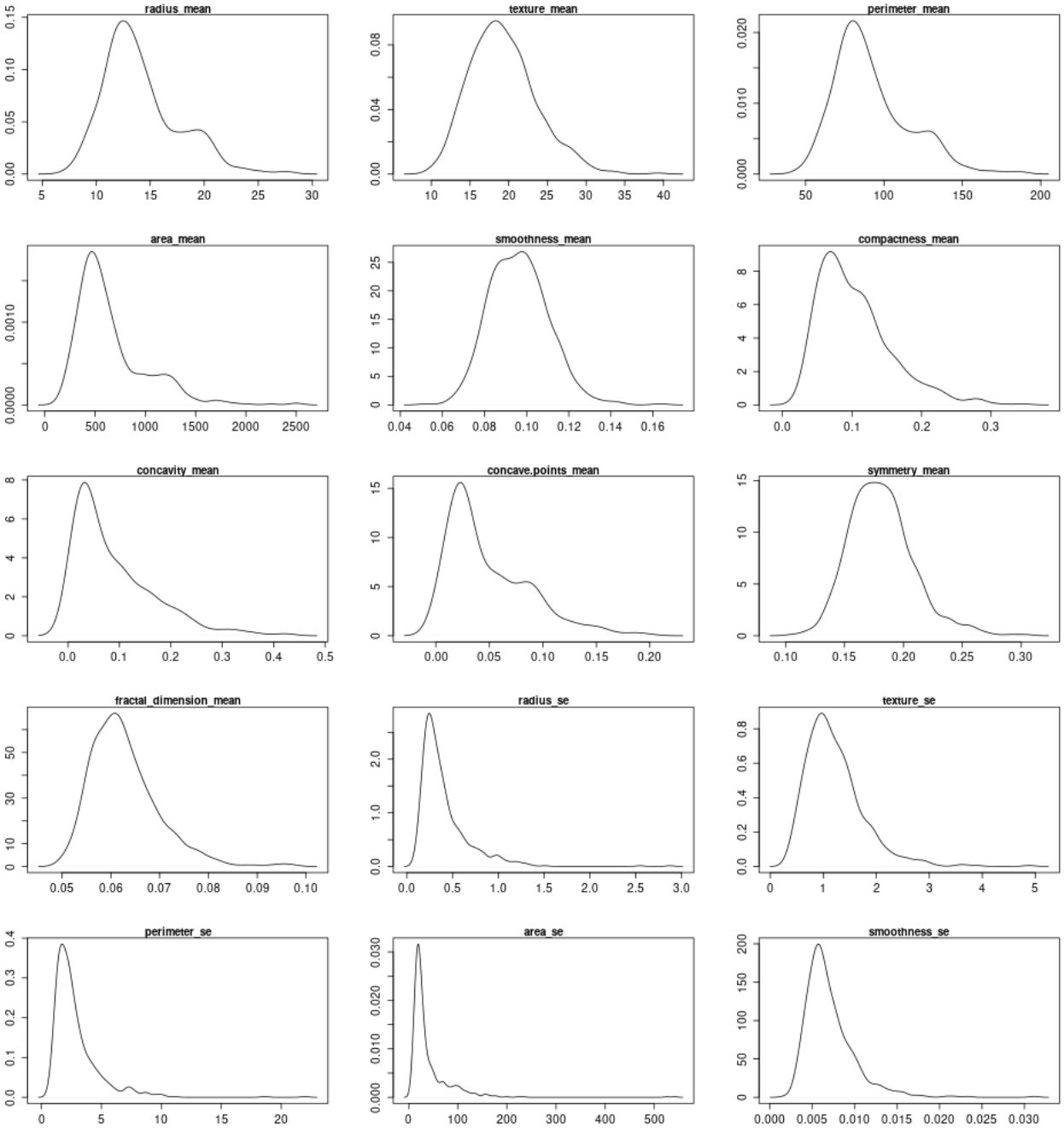

Figure 8: Density plots of the first 15 features.

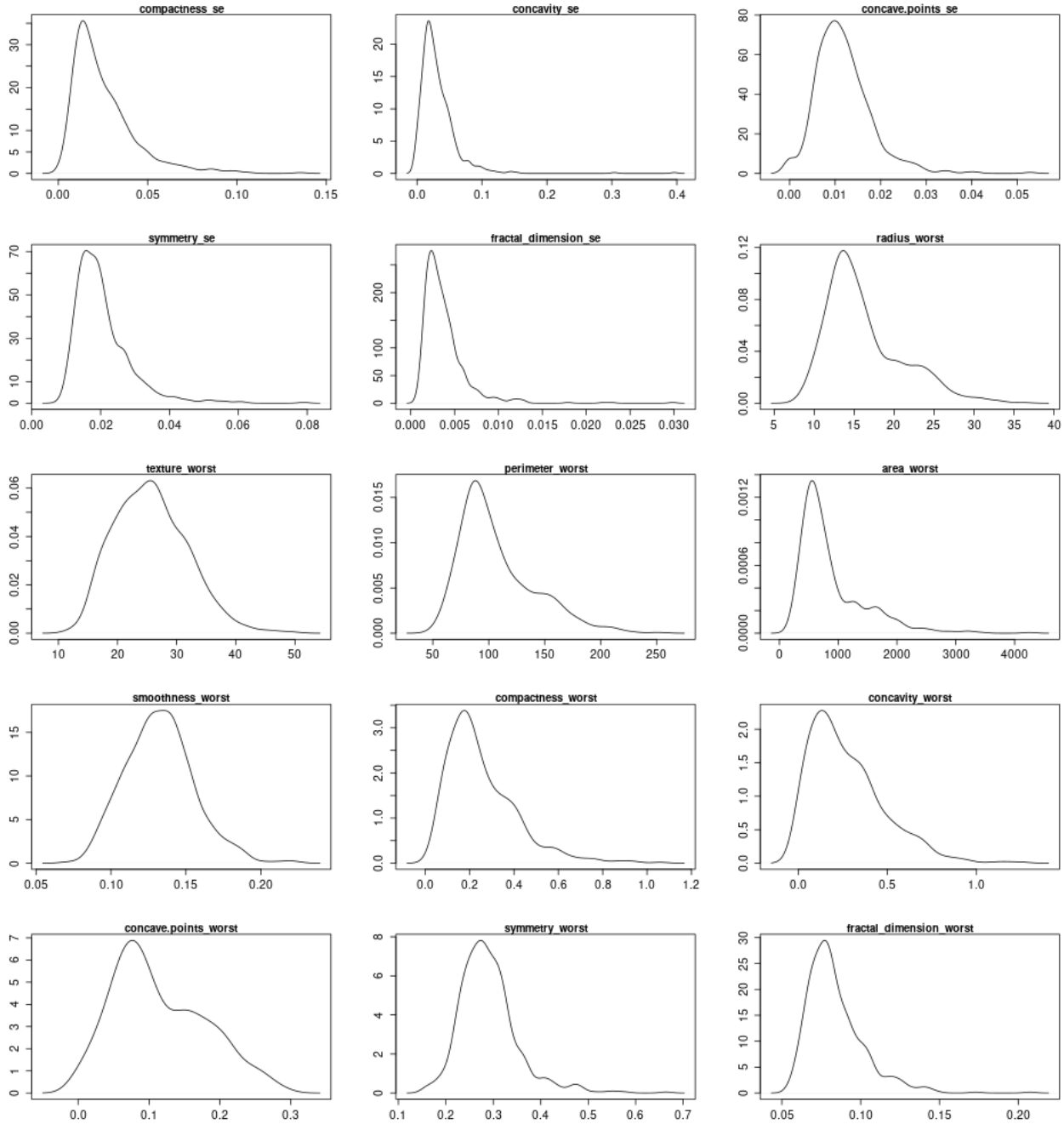

Figure 9: Density plots of the last 15 features.

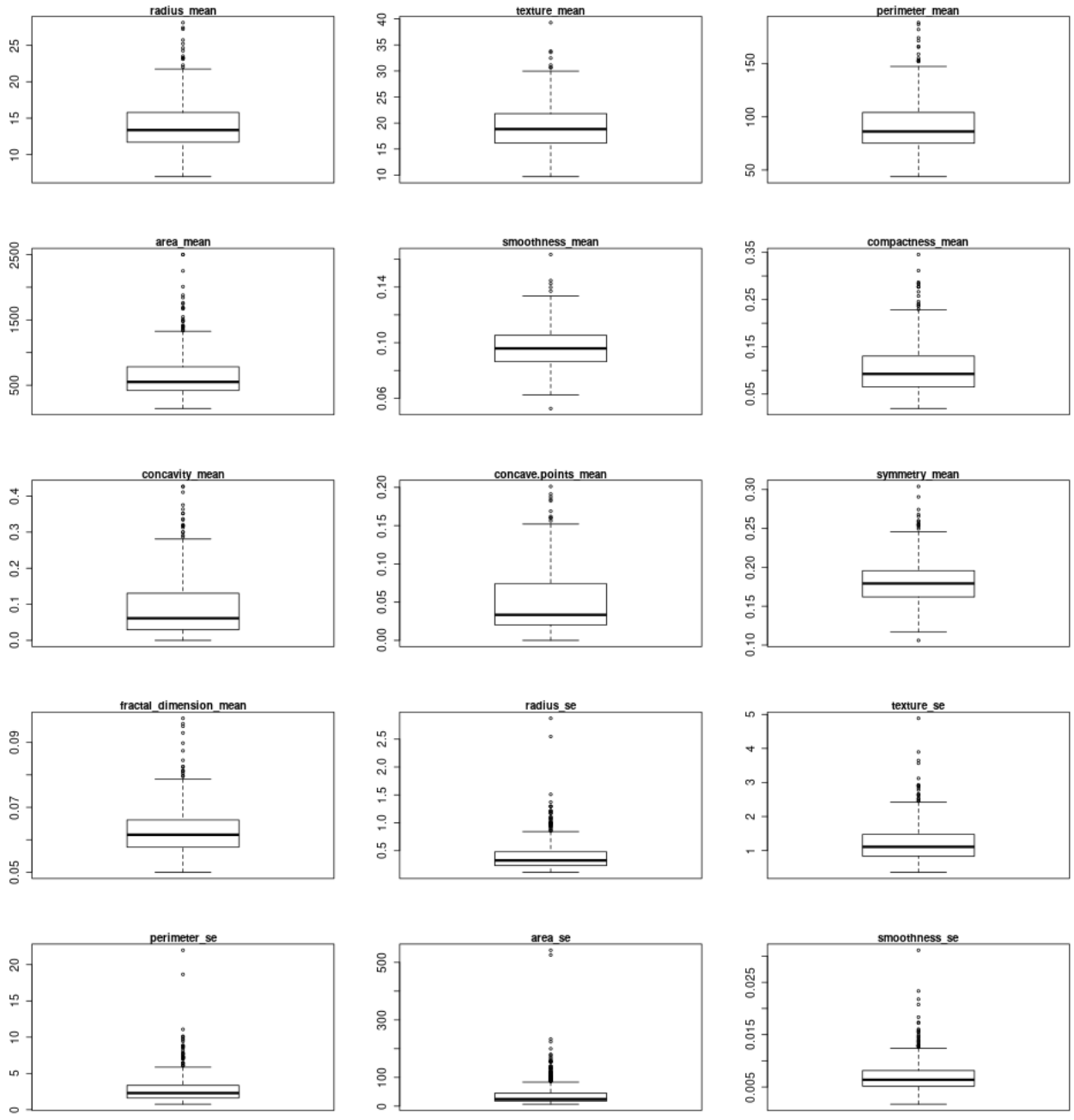

Figure 10: Boxplots of the first 15 features.

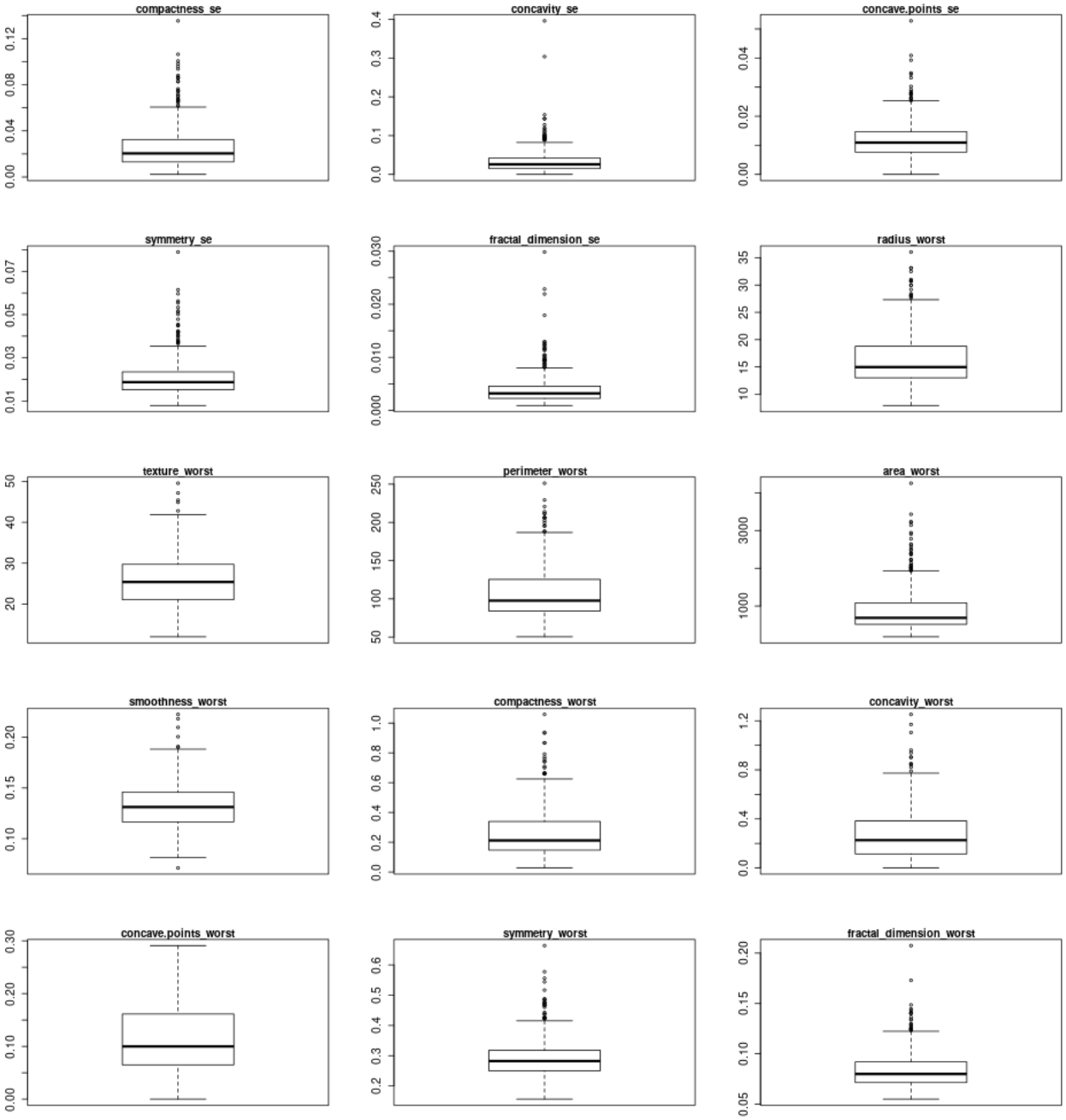

Figure 11: Boxplots of the last 15 features.

3.2 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) can be used to transform the data from a high dimensional space to a lower dimensional one. PCA is a technique used in both exploratory analysis and feature extraction to perform a linear mapping between the original data and a new set of independent features, explaining original data's variance.

How to perform PCA

1. **Standardize the data**. Since PCA is used to explain the variance of the data, non-standardized data could lead to biases relative to the domain of each feature.
2. **Calculate the covariance matrix.**
3. **Retrieve eigenvectors and eigenvalues of the covariance matrix.**

PCA results Eigenvectors with the highest eigenvalues are the most important components. Each component explains some of the initial data variance. The total variance is the sum of variances of each component (eigenvalues). The cumulative explained variance ratio of the nth component is defined as the sum of ratios of explained variance of the first n components. The main goal of PCA is to select a number of components that explains a chosen percentage of variance.

From the PCA results it appears that the first two principal components together ($PC1$, $PC2$) can explain the majority of the variance (63%), while the contribute of the last 19 components is below the 1% (Figure 13).

Another interesting result is that $PC1$ correlates with almost all the features, PC3 correlates with $*\_se$ and with some of the $*\_worst$ variables, while PC4 strongly correlates with texture features (Figure 12).

3.3 Correlation analysis

Correlation analysis is useful to determine whether there are highly correlated features that could be dropped, in order to reduce the problem complexity and to see how dropping these variables affects model performances.

In the correlation matrix below it emerges that there are highly correlated features (Figure 14). More precisely, a cutoff $t = 0.85$ was considered, thus 13 couples were identified as highly correlated (Pearson correlation coeff. $>$ t) (Table 3.3).

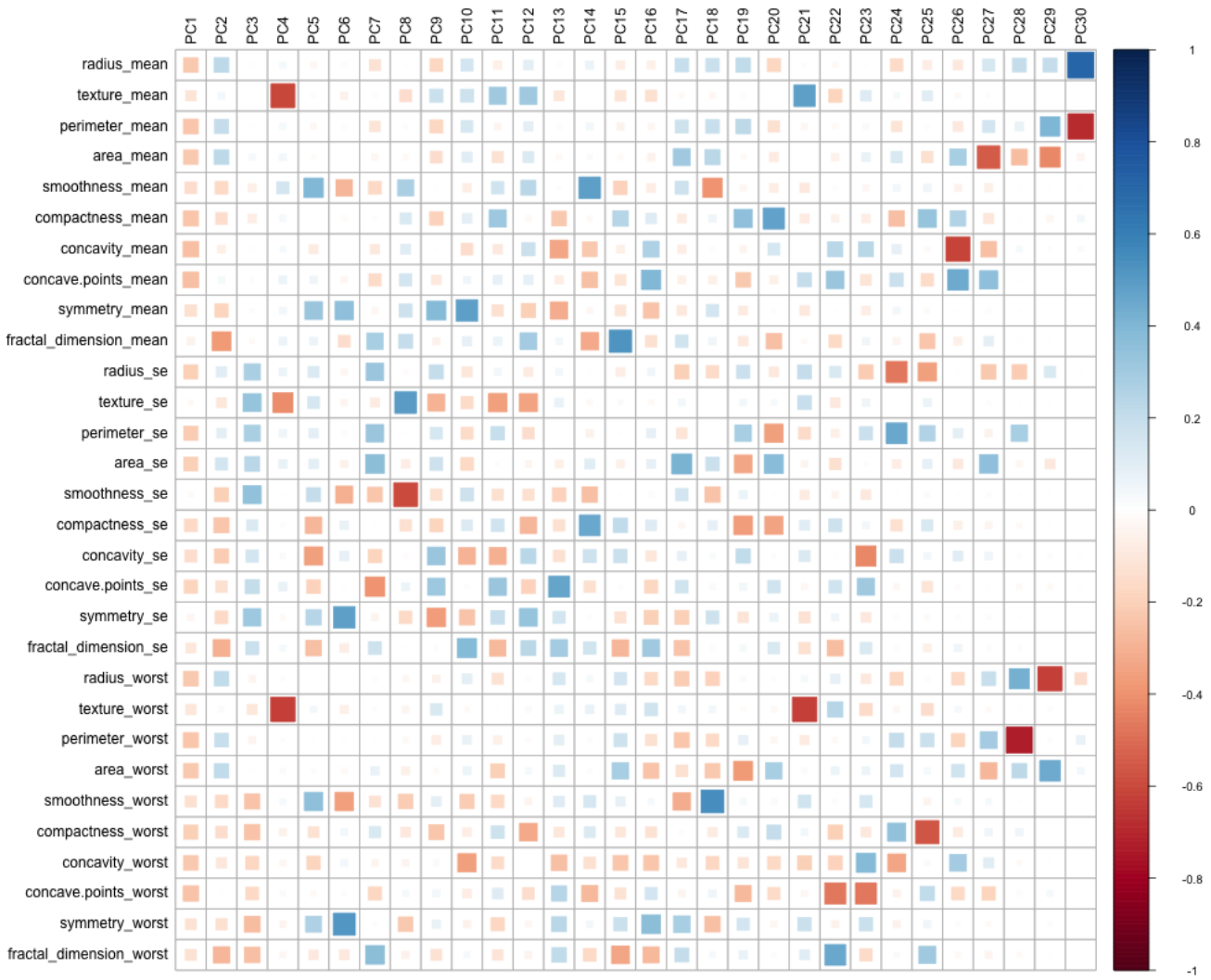

Figure 12: Correlation matrix of variable loadings (PCA).

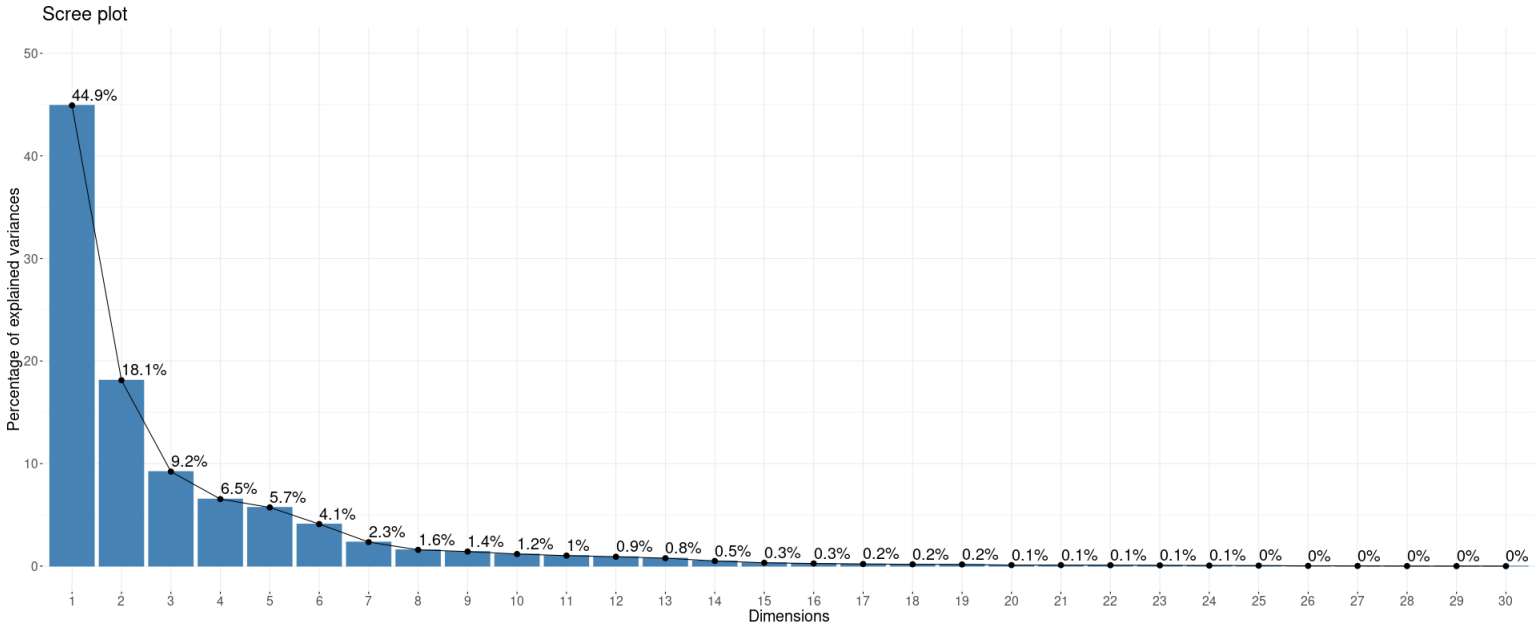

Figure 13: Scree plot.

Feature 1	Feature 2	Pearson correlation coefficient
concavity_mean	concave.points_mean	0.921
concave.points_mean	concave.points_worst	0.910
compactness_mean	compactness_worst	0.866
concave.points_worst	concavity_worst	0.855
concavity_worst	compactness_worst	0.892
perimeter_worst	radius_worst	0.994
radius_worst	perimeter_mean	0.969
perimeter_mean	area_worst	0.942
area_worst	radius_mean	0.941
radius_mean	area_mean	0.987
perimeter_se	radius_se	0.973
radius_se	area_se	0.952
texture_worst	texture_mean	0.912

Table 1: Highly correlated features.

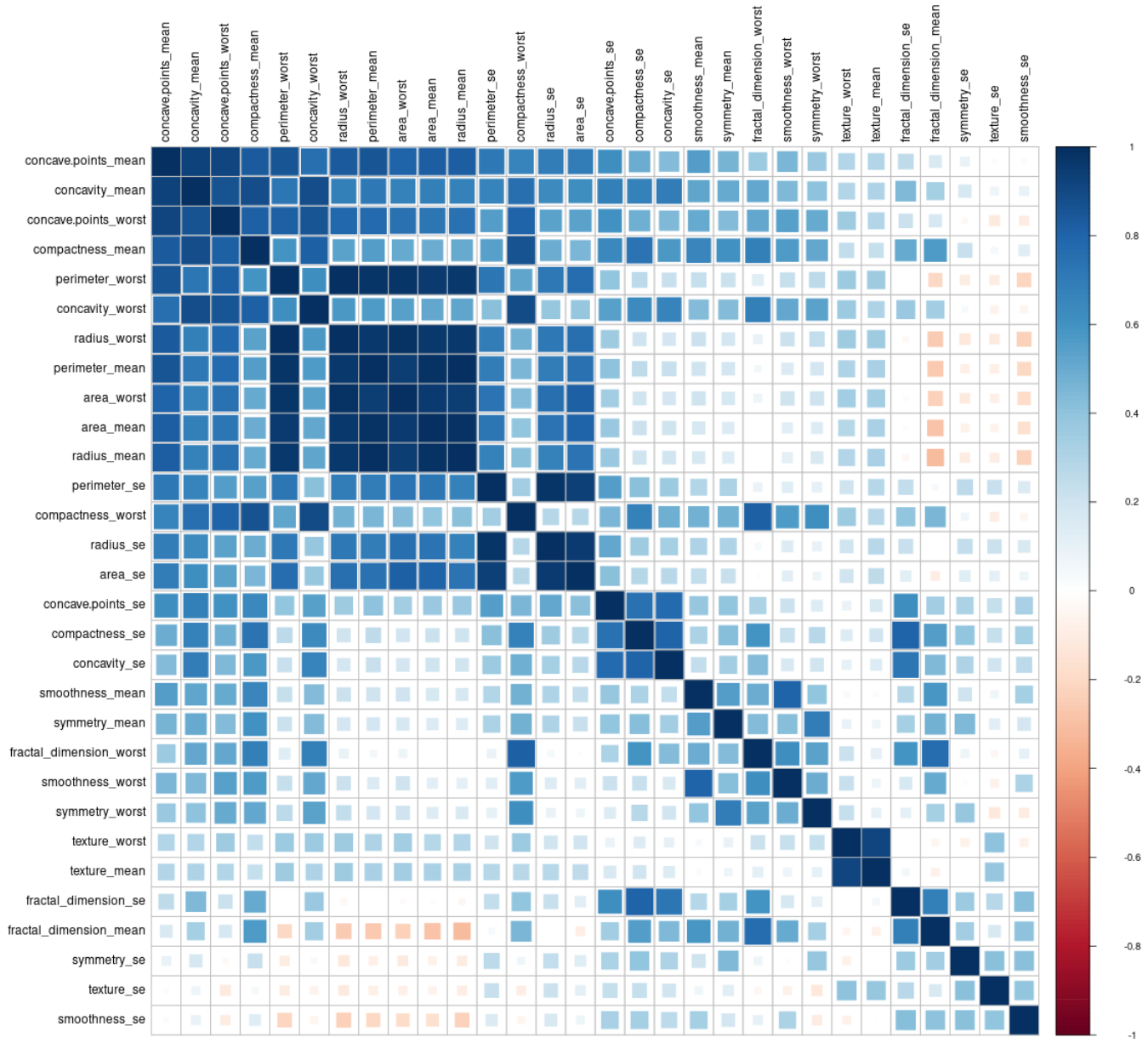

Figure 14: Correlation matrix of the features.

4 Preprocessing

To keep data unbiased and prepare the dataset for the models, different preprocessing approaches were used.

The original dataset was partitioned into a couple of subsets: train and test. Four different versions of these couples were then created: the normalized one (*trainset.norm*, *testset.norm*), the standardized one (*trainset.std*, *testset.std*), the standardized one without the highly correlated features (*trainset.corr*, *testset.corr*) and the one transformed by PCA (*trainset.pca*, *testset.pca*).

After this phase, models were trained and tested on the four couples of sets listed above.

4.1 Normalization and standardization

Normalization and standardization are two feature scaling techniques used in the machine learning field. They can be useful or not depending on the machine learning algorithm used: algorithms based on distance and on gradient descent can be significantly affected by different feature scales (e.g. Neural Networks, SVM, K-means), while other algorithms shouldn't be affected (e.g. Tree-based techniques, Naive Bayes). Nevertheless there's no way of knowing a priori which technique better fits the problem and leads to the best results.

Normalization Normalization scales values between a fixed range (e.g. [0,1], [-1,1]). The method used is called "Min-Max normalization" and scales values between [0,1] according to the following formula:

$$x' = \frac{x - min(x)}{max(x) - min(x),} \tag{4}$$

where x is the original value and x' is the normalized value.

Standardization Standardization doesn't scale values between a given range, but makes each feature value have zero-mean and variance equal to one, according to the following formula:

$$x' = \frac{x - average(x)}{standardDeviation(x).} \tag{5}$$

For the purpose of this analysis, both methods were used and results are compared.

4.2 Feature Reduction

In the machine learning field, a dataset with a large number of features can lead to worse results than a dataset with fewer features, and at the same time requires more computational power [8]. A high dimensional feature space is often a symptom of irrelevant or redundant features; moreover, a large number of features requires a large number of samples to perform a good analysis. The presence of problems related to the large number of features is often addressed as the curse of dimensionality or peaking phenomenon [9]. A large number of features requires an exponential computational effort and can decrease the

accuracy of a model. Feature reduction is a set of techniques used to transform a high dimensional space into a lower dimensional space preserving the intrinsic meaning of the data, allowing to work with fewer features. There are two types of feature reduction.

Feature selection It's used to select the most important features of the dataset, removing the others.

Feature extraction It's used to create a new set of features from the original dataset, using mapping algorithms.

4.2.1 Feature selection

By correlation analysis, 13 highly correlated features were identified (Table 3.3) and then removed from *trainset.std* and *testset.std*, obtaining *trainset.corr* and *testset.corr*.

Removed features concavity_mean, concave.points_mean, compactness_mean, concave.points_worst, concavity_worst, perimeter_worst, radius_worst, perimeter_mean, area_worst, radius_mean, perimeter_se, area_se, texture_mean.

4.2.2 Feature extraction

By performing PCA and selecting the most important components, the number of features used to train the models can be reduced without too much information loss.

It was chosen 99% of the explained variance as threshold, so the first 17 components were selected.

4.3 Train and test split

The training set is the dataset used to train the model; the validation set is used to tune the hyperparameters; the test set is used to evaluate the final models and get unbiased results. Stratified sampling of the dataset was not performed in order to get the same distribution of the target value for each set because it would have introduced a bias, since the benign/malignant ratio is changing.

The train set was made of the 80% of the dataset, while the test set was the remaining part of it (20%). 10-fold cross-validation with 5 repetitions was performed on the train set.

5 Models

Three different classifiers have been used:

- Naive Bayes
- Support Vector Machine
- Neural Network

Each model was trained on the four training sets: *trainset.norm*, *trainset.std*, *trainset.corr*, *trainset.pca* described in section 4.

All three classifiers have been implemented in the **R programming language** in the **caret** package (Classification And Regression Training) which contains numerous useful functions to train different models.

All models were trained using 10-fold cross-validation with 5 repetitions on the four training sets.

5.1 Naive Bayes

Naive Bayes classifiers are probabilistic classifiers based on Bayes' theorem. This model assumes all the random variables are **stochastically independent** from each other. Normal distribution of features couldn't be assumed, as explained in section 3, but, for sake of completeness, training was performed both with and without this assumption.

Naive Bayes was selected as a baseline since it's the simplest of the three classifiers.

Tuning phase

For this model two executions for each dataset were performed. In the first execution normal distribution of the features were assumed. In the second one Kernel Density Estimation (KDE) was performed to estimate feature distribution. For every dataset the KDE version had the best results.

5.2 Support Vector Machine (SVM)

Support Vector Machines are a class of classification models with the goal of finding the *best hyperplane* which divides the two classes, found by **maximizing** the margin between the two classes.

Tuning phase

Model parameters were tuned with *Grid Search*, where hyperparameters assume values as in Table 2.

Kernel degree is the degree of the kernel function used.

Scale, also known as *gamma*, is a parameter used to regularize the kernel function.

C is the parameter used for the soft margin cost function, which controls overfitting: a smaller value allows more examples to be "on the wrong side of the margin", while a larger value makes the classifier more restrictive.

Kernel degree	**Scale**	**c**
1, 2, 3, 4, 5	1/(number of features)	0.25, 0.50, 0.75, 1, 1.25, 1.50, 1.75, 2

Table 2: Tuning parameters for the Support Vector Machine.

The best configuration of hyperparameters is reported in Table 3

Dataset	Degree	Scale	C
norm	1	0.0333	0.75
std	1	0.0333	0.75
pca	2	0.0625	1.5
corr	1	0.0625	0.75

Table 3: Best tuning for the different datasets.

5.3 Neural Network

Neural Networks are a computing system inspired on human brain functioning. A neural network is based on a collection of connected nodes called neurons, grouped in layers. Each neuron in a layer is connected to all the neurons in the next layer (fully connected layers). Every connection is weighted so the input of a given neuron in a given layer (input layer excluded) is the weighted output of all the neurons in the previous layer.

During the training phase backpropagation is used to update weights.

Neural Networks are the most computationally expensive models of the three considered.

Tuning phase

Model parameters were tuned selecting the best among every possible combination expressed in Table 4.

Each neuron of a layer is connected with all of the neurons of the next layer.

More details about the weights of the best tuning nets can be found at the Github project repository[3].

Layer 1	Layer 2
3, 4, 5, 6, 7, 8, 9, 10	0, 1, 2, 3, 4, 5

Table 4: Tuning parameters for the Neural Network.

The best configuration of hyperparameters is reported in Table 5

5.4 Training time

Computations were run on a Ryzen 7 2700X, with 16GB RAM DDr4, Ubuntu 20.04. Times displayed in Table 6 referred to the average training time on *trainset.norm* and *trainset.std*.

[3]link: `https://github.com/aXhyra/breast_cancer_ML`

Dataset	Neurons in layer 1	Neurons in layer 2
norm	3	0
std	3	1
pca	4	1
corr	4	0

Table 5: Best tuning for the different datasets.

	Naive Bayes	**SVM**	**Neural Network**
Without Parallelization	3.3145 ± 0.0841	59.9235 ± 0.4066	353.3030 ± 0.5657
With Parallelization	1.3865 ± 0.1718	10.5640 ± 0.7849	58.0615 ± 0.2807

Table 6: Training time with and without parallelization (seconds)

"A cross-validation phase" = 10-fold cross-validation with 5 repetitions.
The NB classifier performed two trainings on a single dataset (both with and without kernel density estimation) with a single cross-validation phase requiring $\approx 1.7s$.
Given the SVM tuning grid, 40 models were built and trained with a single cross-validation phase requiring $\approx 1.5s$.
Given the Neural Network tuning grid, 48 models were built and trained with a single cross-validation phase requiring $\approx 7.4s$.

6 Results analysis

In this section results are reported for each of the four datasets described in section 4 and for each classification technique.

"M" ("Malignant") is considered as the positive class.

6.1 dataset.norm

Naive Bayes

The confusion matrix for training and test sets of the norm dataset using Naive Bayes are reported in Table 7.

	Reference	
Predicted	**M**	**B**
M	161	10
B	11	274

	Reference	
Predicted	**M**	**B**
M	36	6
B	4	67

Table 7: Confusion matrix on train and test set for the norm dataset (Naive Bayes).

The ROC curve is reported in Figure 15.

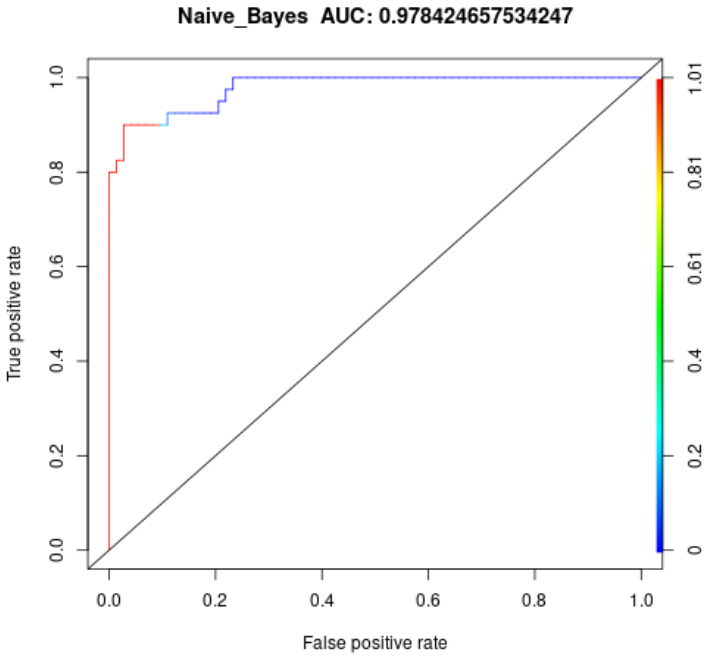

Figure 15: ROC curve on dataset.norm using Naive Bayes.

SVM

The confusion matrix for training and test sets of the norm dataset using SVM are reported in Table 8.

	Reference	
Predicted	**M**	**B**
M	167	1
B	5	283

	Reference	
Predicted	**M**	**B**
M	38	2
B	2	71

Table 8: Confusion matrix on train and test set for the norm dataset (SVM).

The ROC curve is reported in Figure 16.

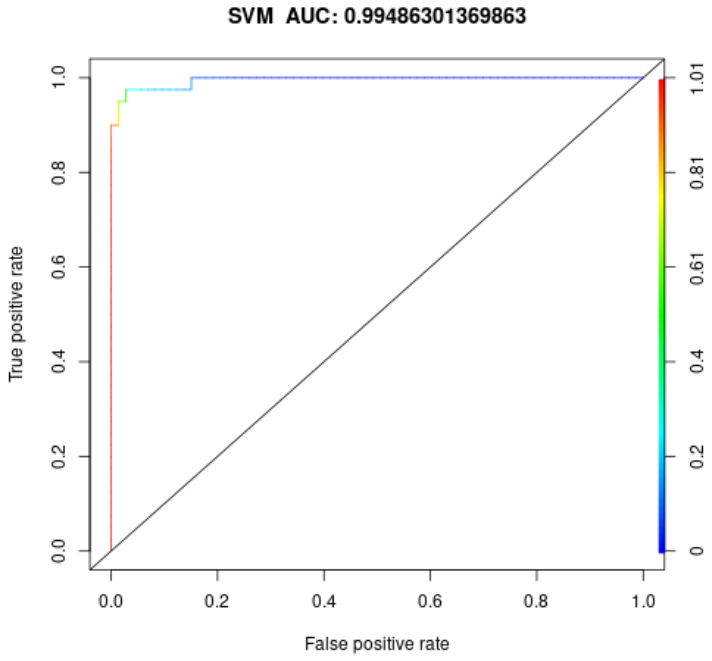

Figure 16: ROC curve on dataset.norm using SVM.

Neural Network

The confusion matrix for training and test sets of the norm dataset using Neural Network are reported in Table 9.

	Reference	
Predicted	**M**	**B**
M	168	4
B	4	280

	Reference	
Predicted	**M**	**B**
M	39	3
B	1	70

Table 9: Confusion matrix on train and test set for the norm dataset (Neural Network).

The ROC curve is reported in Figure 17.

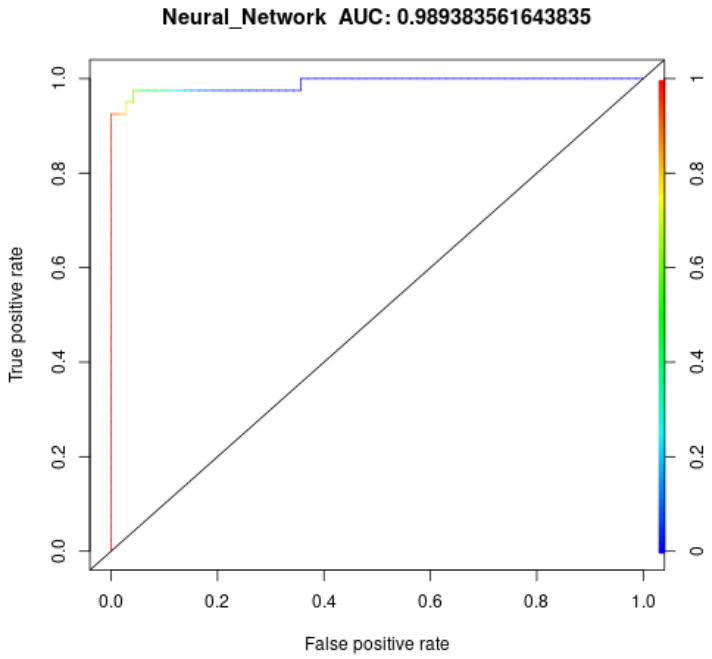

Figure 17: ROC curve on dataset.norm using a Neural Network.

Models comparison

Table 10 shows the performances obtained by the 3 models on *dataset.norm*.

	Naive Bayes	**SVM**	**Neural Network**
Accuracy	0.9115	0.9646	0.9646
Accuracy CI	(0.8433; 0.9567)	(0.9118; 0.9903)	(0.9118; 0.9903)
Precision	0.8571	0.9500	0.9286
Recall	0.9000	0.9500	0.9750
F1	0.8780	0.9500	0.9512
Sensitivity	0.9000	0.9500	0.9750
Specificity	0.9178	0.9726	0.9589

Table 10: Performances on testset.norm. Confidence Interval (CI): 95%.

The ROC curve of all the models is reported in Figure 18.

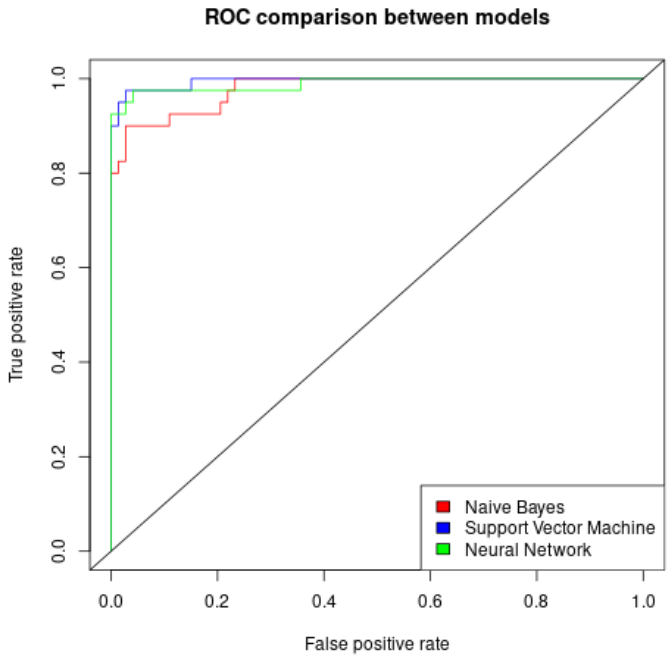

Figure 18: ROC curve of the models on dataset.norm.

6.2 dataset.std

Naive Bayes

The confusion matrix for training and test sets of the std dataset using Naive Bayes are reported in Table 11.

	Reference	
Predicted	**M**	**B**
M	161	10
B	11	274

	Reference	
Predicted	**M**	**B**
M	36	6
B	4	67

Table 11: Confusion matrix on train and test set for the std dataset (Naive Bayes).

The ROC curve is reported in Figure 19.

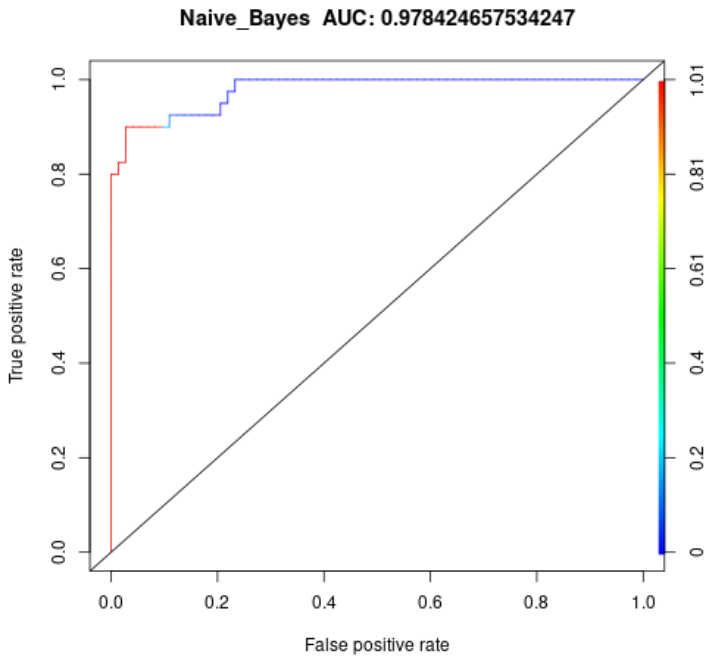

Figure 19: ROC curve on dataset.std using Naive Bayes.

SVM

The confusion matrix for training and test sets of the std dataset using SVM are reported in Table 12.

	Reference	
Predicted	**M**	**B**
M	167	1
B	5	283

	Reference	
Predicted	**M**	**B**
M	38	2
B	2	71

Table 12: Confusion matrix on train and test set for the std dataset (SVM).

The ROC curve is reported in Figure 20.

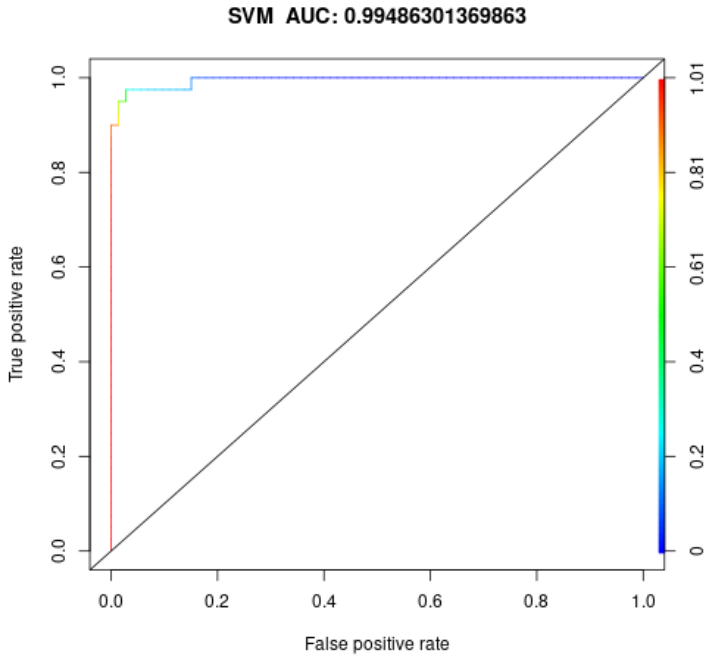

Figure 20: ROC curve on dataset.std using SVM.

Neural Network

The confusion matrix for training and test sets of the std dataset using Neural Network are reported in Table 13.

	Reference	
Predicted	**M**	**B**
M	168	0
B	4	284

	Reference	
Predicted	**M**	**B**
M	37	2
B	3	71

Table 13: Confusion matrix on train and test set for the std dataset (Neural Network).

The ROC curve is reported in Figure 21.

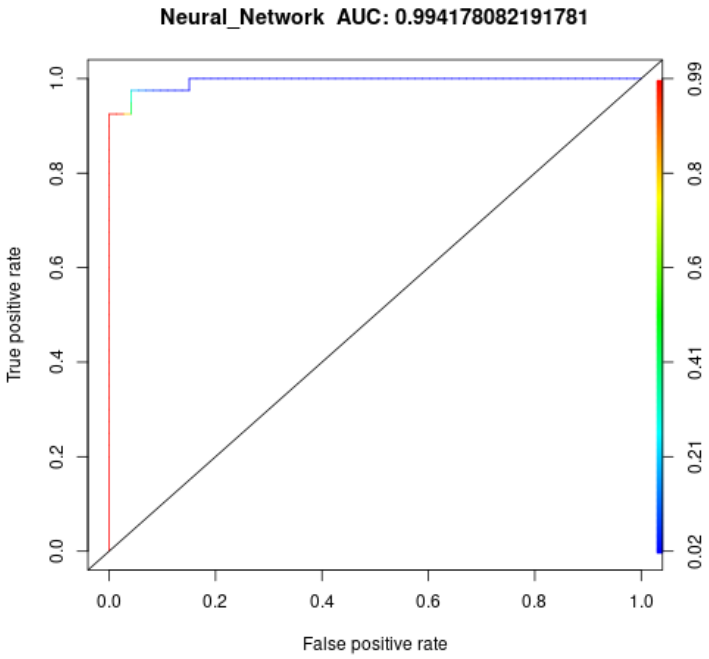

Figure 21: ROC curve on dataset.std using a Neural Network.

Models comparison

Table 14 shows the performances obtained by the 3 models on *dataset.std*.

	Naive Bayes	**SVM**	**Neural Network**
Accuracy	0.9115	0.9646	0.9558
Accuracy CI	(0.8433; 0.9567)	(0.9118; 0.9903)	(0.8998; 0.9855)
Precision	0.8571	0.9500	0.9487
Recall	0.9000	0.9500	0.9250
F1	0.8780	0.9500	0.9367
Sensitivity	0.9000	0.9500	0.9250
Specificity	0.9178	0.9726	0.9726

Table 14: Performances on testset.std. Confidence Interval (CI): 95%.

The ROC curve of all the models is reported in Figure 22.

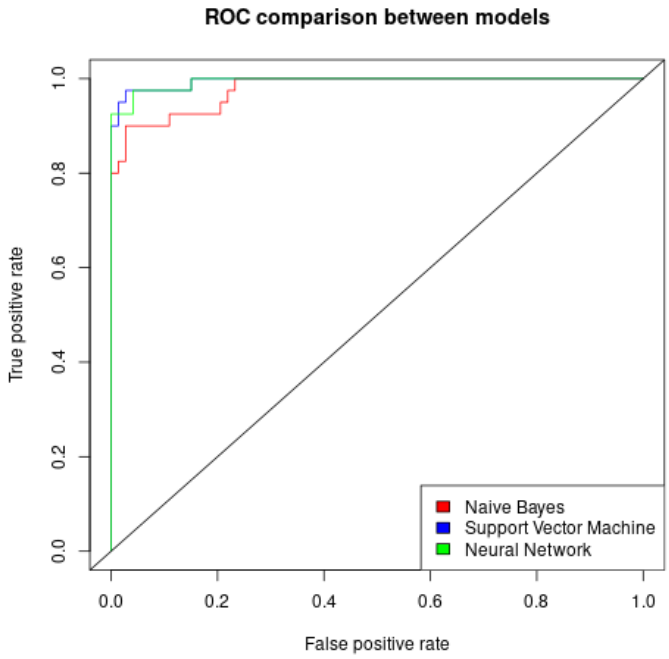

Figure 22: ROC curve of the models on dataset.std.

6.3 dataset.corr

Naive Bayes

The confusion matrix for training and test sets of the corr dataset using Naive Bayes are reported in Table 15.

	Reference	
Predicted	**M**	**B**
M	152	15
B	20	269

	Reference	
Predicted	**M**	**B**
M	37	4
B	3	69

Table 15: Confusion matrix on train and test set for the corr dataset (Naive Bayes).

The ROC curve is reported in Figure 23.

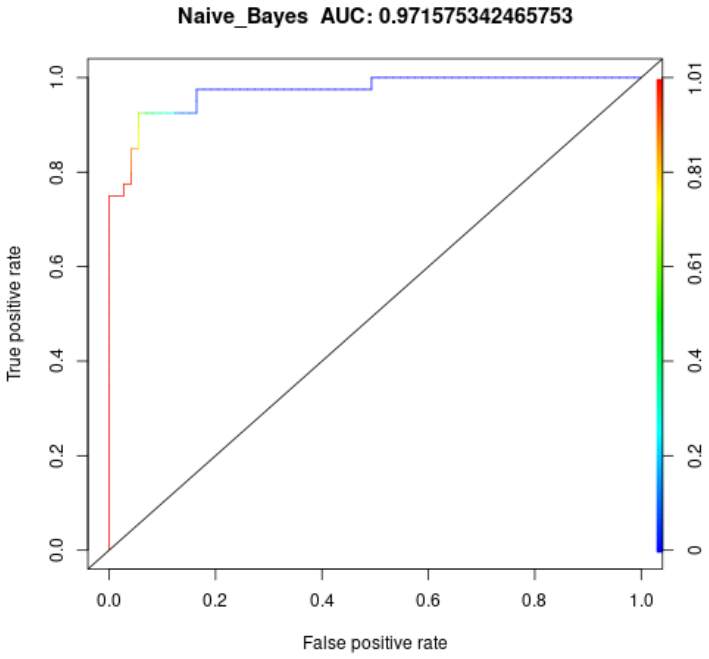

Figure 23: ROC curve on dataset.corr using Naive Bayes.

SVM

The confusion matrix for training and test sets of the corr dataset using SVM are reported in Table 16.

	Reference	
Predicted	**M**	**B**
M	166	5
B	6	279

	Reference	
Predicted	**M**	**B**
M	38	1
B	2	72

Table 16: Confusion matrix on train and test set for the corr dataset (SVM).

The ROC curve is reported in Figure 24.

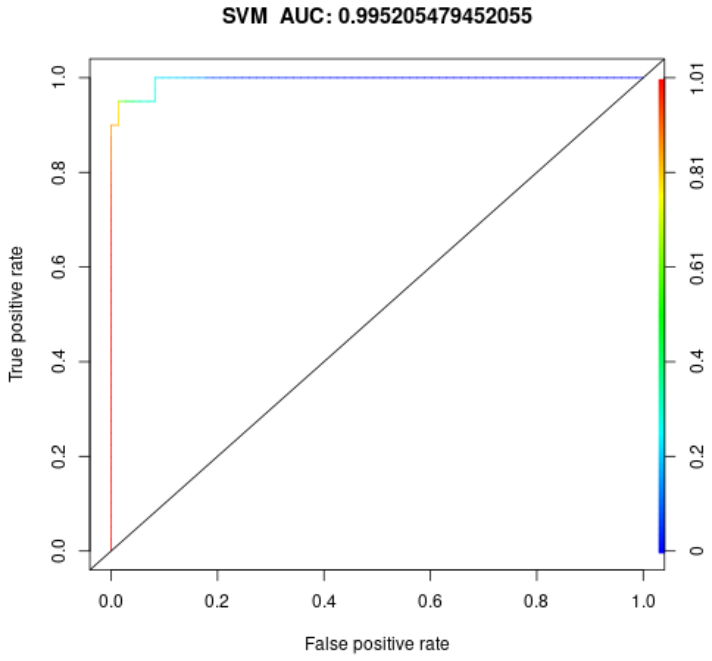

Figure 24: ROC curve on dataset.corr using SVM.

Neural Network

The confusion matrix for training and test sets of the corr dataset using Neural Network are reported in Table 17.

	Reference	
Predicted	**M**	**B**
M	168	0
B	4	284

	Reference	
Predicted	**M**	**B**
M	39	2
B	1	71

Table 17: Confusion matrix on train and test set for the corr dataset (Neural Network).

The ROC curve is reported in Figure 25.

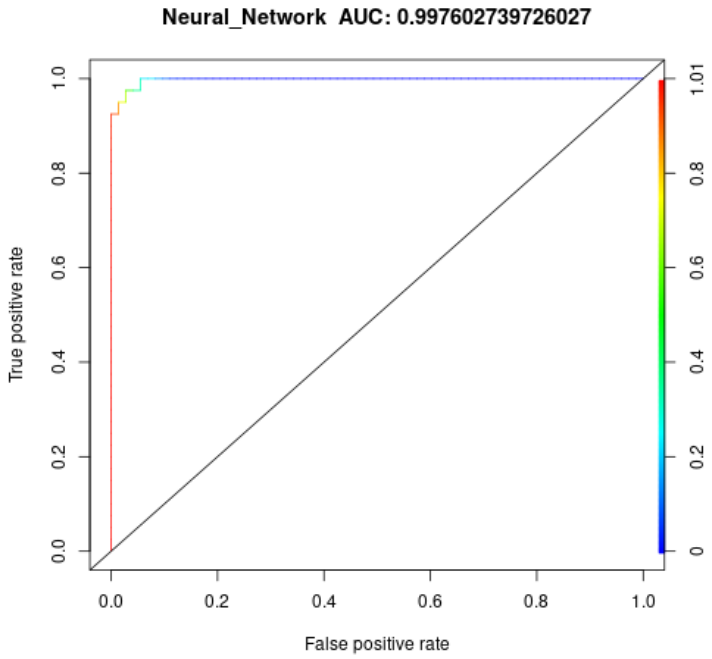

Figure 25: ROC curve on dataset.corr using a Neural Network.

Models comparison

Table 18 shows the performances obtained by the 3 models on *dataset.corr*.

	Naive Bayes	**SVM**	**Neural Network**
Accuracy	0.9381	0.9735	0.9735
Accuracy CI	(0.8765; 0.9747)	(0.9244; 0.9945)	(0.9244; 0.9945)
Precision	0.9024	0.9744	0.9512
Recall	0.9250	0.9500	0.9750
F1	0.9136	0.9620	0.9630
Sensitivity	0.9250	0.9500	0.9750
Specificity	0.9452	0.9863	0.9726

Table 18: Performances on testset.corr. Confidence Interval (CI): 95%.

The ROC curve of all the models is reported in Figure 26.

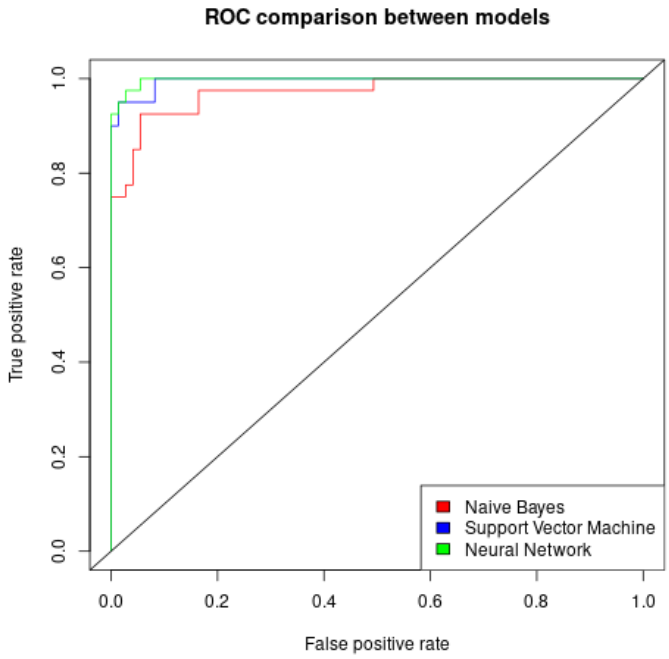

Figure 26: ROC curve of the models on dataset.corr.

6.4 dataset.pca

Naive Bayes

The confusion matrix for training and test sets of the PCA dataset using Naive Bayes are reported in Table 19.

	Reference	
Predicted	**M**	**B**
M	154	5
B	18	279

	Reference	
Predicted	**M**	**B**
M	34	5
B	6	68

Table 19: Confusion matrix on train and test set for the pca dataset (Naive Bayes).

The ROC curve is reported in Figure 27.

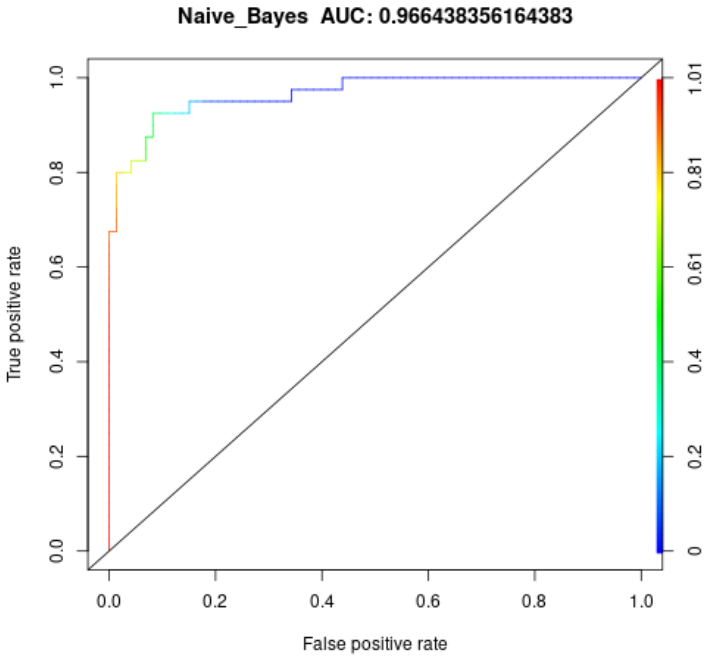

Figure 27: ROC curve on dataset.pca using Naive Bayes.

SVM

The confusion matrix for training and test sets of the pca dataset using SVM are reported in Table **??**.

	Reference	
Predicted	**M**	**B**
M	166	0
B	6	284

	Reference	
Predicted	**M**	**B**
M	37	2
B	3	71

Table 20: Confusion matrix on train and test set for the pca dataset (SVM).

The ROC curve is reported in Figure 28.

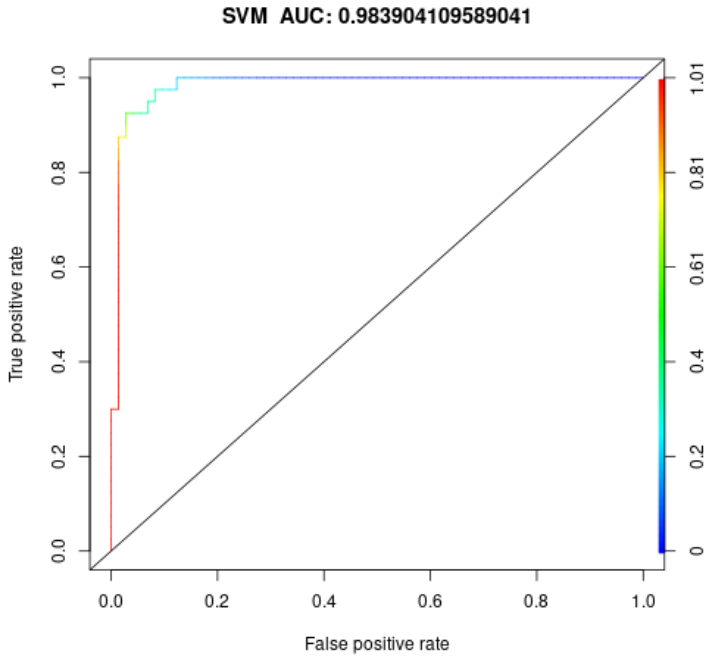

Figure 28: ROC curve on dataset.pca using SVM.

Neural Network

The confusion matrix for training and test sets of the pca dataset using Neural Network are reported in Table **??**.

	Reference	
Predicted	**M**	**B**
M	168	0
B	4	284

	Reference	
Predicted	**M**	**B**
M	37	2
B	3	71

Table 21: Confusion matrix on train and test set for the pca dataset (Neural Network).

The ROC curve is reported in Figure 29.

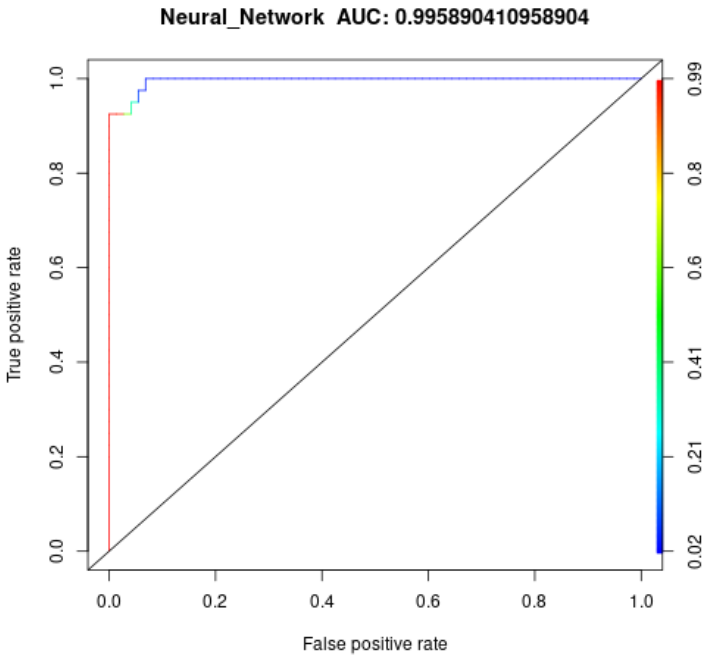

Figure 29: ROC curve on dataset.pca using a Neural Network.

Models comparison

Table 22 shows the performances obtained by the 3 models on *dataset.pca*.

	Naive Bayes	**SVM**	**Neural Network**
Accuracy	0.9027	0.9558	0.9558
Accuracy CI	(0.8325; 0.9504)	(0.8998; 0.9855)	(0.8998; 0.9855)
Precision	0.8718	0.9487	0.9487
Recall	0.8500	0.9250	0.9250
F1	0.8608	0.9367	0.9367
Sensitivity	0.8500	0.9250	0.9250
Specificity	0.9315	0.9726	0.9726

Table 22: Performances on testset.pca. Confidence Interval (CI): 95%.

The ROC curve of all the models is reported in Figure 30.

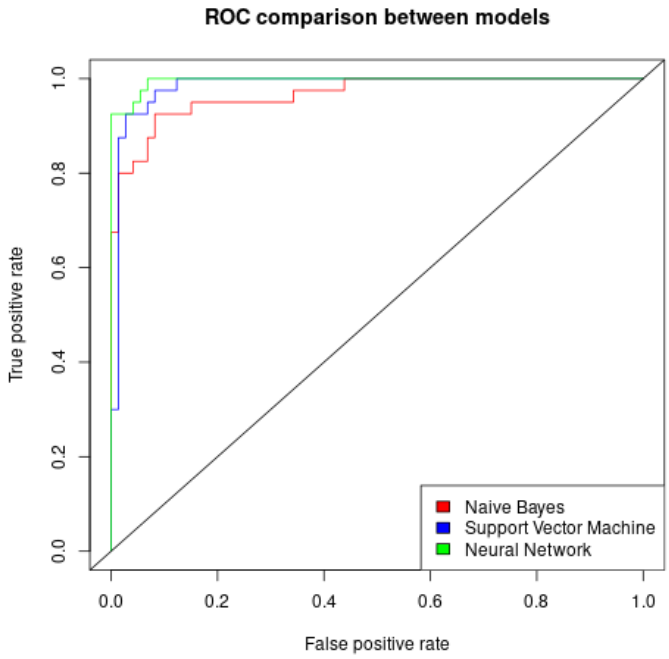

Figure 30: ROC curve of the models on dataset.pca.

7 Conclusions

This report introduced the problem of cancer classification and addressed a possible solution to the problem using an open dataset[4]. After a preliminary analysis four dataset were produced:

1. dataset.norm: normalized dataset;
2. dataset.std: standardized dataset;
3. dataset.corr: standardized dataset without the highly correlated features;
4. dataset.pca: dataset projected on its Principal Components.

On the produced datasets three classifiers were tested (Naive Bayes, SVM, Neural Network).

Each model performed the best on *dataset.corr*. Bayes performed the best on this, probably because of its assumptions on independent variables. SVM performed the best on the same dataset probably because the reduced feature space allowed the model to discriminate a better separation hyperplane.

There are no significant differences betweeen the effects of normalization and standardization. Naive Bayes performances indeed remained the same because feature probability distributions aren't affected by those scale transformations, and SVM performances also remained the same. Neural Network had slightly more benefits from normalization, probably because smaller input values make gradient descent easier.

Projecting the dataset on its Principal Components worsen the results, therefore it's not more advantageous than removing highly correlated features, which also leads to a similar complexity.

In general, Neural Networks and SVM models performed better than Naive Bayes, probably because the latter is a much simpler model and assumes feature independence, which was an unsatisfied requirement.

Pointing out that this analysis is related to the medical field, a false negative is more relevant than a false positive. In case of false positiveness further tests could be done, while, in case of a false negative, patients risk serious health complications due to not-timely treatments. For this reason, recall (= sensitivity) is the metric that needs to be maximized (from Wikipedia: sensitivity = "probability of a positive test given that the patient has the disease").

The Neural Network seems to be the best fitting classifier among the considered ones. It achieved the highest recall on *dataset.corr*, which also achieved the best accuracy out of the four datasets.

Cancer screening tests need to be done frequently in order to anticipate the onset of tumors and their progression. However a minimum delay between tests and diagnoses is allowed. Also, the supervision of a specialist is always required, so a more computationally expensive model at test time is suitable, such as Neural Network.

[4]Source: `https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29`

References

[1] J. Ferlay, M. Colombet, I. Soerjomataram, C. Mathers, D. M. Parkin, M. Piñeros, A. Znaor, and F. Bray. Estimating the global cancer incidence and mortality in 2018: Globocan sources and methods. *International Journal of Cancer*, 144(8):1941–1953, 2019.

[2] A. Mouelhi, M. Sayadi, F. Fnaiech, K. Mrad, and K. B. Romdhane. Automatic image segmentation of nuclear stained breast tissue sections using color active contour model and an improved watershed method. *Biomedical Signal Processing and Control*, 8(5):421 – 436, 2013.

[3] H. Höfener, A. Homeyer, N. Weiss, J. Molin, C. F. Lundström, and H. K. Hahn. Deep learning nuclei detection: A simple approach can deliver state-of-the-art results. *Computerized Medical Imaging and Graphics*, 70:43 – 52, 2018.

[4] N. Kumar, R. Verma, S. Sharma, S. Bhargava, A. Vahadane, and A. Sethi. A dataset and a technique for generalized nuclear segmentation for computational pathology. *IEEE Transactions on Medical Imaging*, 36:1–1, 03 2017.

[5] W. N. Street, W. H. Wolberg, and O. L. Mangasarian. Nuclear feature extraction for breast tumor diagnosis. In Raj S. Acharya and Dmitry B. Goldgof, editors, *Biomedical Image Processing and Biomedical Visualization*, volume 1905, pages 861 – 870. International Society for Optics and Photonics, SPIE, 1993.

[6] O. L. Mangasarian, W. N. Street, and W. H. Wolberg. Breast cancer diagnosis and prognosis via linear programming. *OPERATIONS RESEARCH*, 43:570–577, 1995.

[7] B. B. Mandelbrot. *The fractal geometry of nature*, volume 2. WH freeman New York, 1982.

[8] Pádraig Cunningham. Dimension reduction. In Matthieu Cord and Pádraig Cunningham, editors, *Machine Learning Techniques for Multimedia - Case Studies on Organization and Retrieval*, Cognitive Technologies, pages 91–112. Springer, 2008.

[9] Amin Zollanvari, Alex Pappachen James, and Reza Sameni. A theoretical analysis of the peaking phenomenon in classification. *J. Classif.*, 37(2):421–434, 2020.