Classification algorithms on healthcare data

A comparative analysis on Wisconsin Breast Cancer Data

Claudio Mazzi

May 29, 2024

Abstract

Non-communicable diseases are recognized as the leading cause of death and disability worldwide. Among them, oncological diseases are the most important in prevalence and gravity. Timeliness of care and accuracy of diagnosis is critical to ensure less invasive medical interventions and a greater likelihood of recovery for patients. In this report, we will focus on breast cancer, which is one of the most common cancers overall and the most prevalent cancer in the female population. Thanks to modern Imaging Data Mining techniques, databases collecting breast screening information are available for study. Therefore, we can apply Machine Learning algorithms on breast screening data to provide pre-diagnoses capable of distinguishing benign from malignant tumors. In particular, we will focus on several Supervised Learning techniques, such as logistic regression with LASSO penalization, k-Nearest Neighbours, Linear Discriminant Analysis, and Support Vector Machines. We study their goodness in classifying benign and malignant tumors from the Wisconsin Breast Cancer Database.

Contents

1 Introduction

According to the World Health Organization (WHO) breast cancer is the most common type of cancer among women worldwide; in 2022, there were 2.3 million women diagnosed with breast cancer and 670000 deaths globally [1]. Breast cancer occurs in every country of the world in women at any age after puberty but with increasing rates in later life. It is about 100 times more common in women than men. European and US women are the greater victims of getting and dying from breast cancer than the Asian, African or Native-American women. Breast cancer is the resulting of an abonormal growth of cells in the breast tissue, and the formation of a lump of mass, commonly referred to as a tumor. Initial stages tumor can be benign (not cancer), but severe form of this tumor becomes pre-malignant (precancerous) and, at the end, malignant (cancer) [2]. Tests such as Magnetic resonance imaging (MRI), mammogram, ultrasound and biopsy are commonly used to diagnose breast cancer performed. Although these techniques have good detection capability, it is critical to act early in prediagnosis. In this sense, statistical learning and, in particular, Machine Learning (ML) and Deep Learning (DL) techniques have enabled a further step forward in the treatment and prevention of breast cancer specifically, and hopefully can also be applied to other chronic diseases, which are increasing in terms of prevalence and comorbidities. This report intendeds to compare different supervised learning models in classifying bening and malignant tumors from the Wisconsin Breast Cancer Database (WBCD) obtained from Kaggle. We selected four different predicting models to work with: the logistic regression with LASSO penalization, the k-Nearest Neighbours (kNN), Linear Discriminant Analysis (LDA), and Support Vector Machines (SVM). We provided the confusion matrix of the models and compared Accuracy, Precision, Recall, F1-score, and the corresponding ROC curve. Each model resulted with a good predicting power, in particular SVM presents the best overall results as it will disccussed in the following.

2 Methods

The Breast Cancer Wisconsin (Diagnostic) DataSet is data set created in 1995 by Dr. William H. Wolberg, a physician at the University Of Wisconsin Hospital at Madison, Wisconsin, USA. To build the data set, he analyzed fluid samples taken from patients with solid breast masses and the graphical computer program *Xcyt*, which is capable of perform the analysis of cytological features based on digital scans. The program uses a curve-fitting algorithm, to compute ten features from each cells in the sample, than it calculates the mean value, the maximum value and the standard error of each feature for the image, returning a 32 valuated vector, composed by the following attributes:

1. ID number of the cell
2. Diagnosis (M = malignant, B = bening)
3. X (unnamed logical variable)

a. Radius (mean distance from the center to perimeter's points)

b. Texture (standard deviation of gray-scale values)

c. Perimeter

d. Area

e. Smoothness (local variation in radius lengths)

f. Compactness (perimeter^2/ area $-$ 1)

g. Concavity (severity of concave portions of the contour)

h. Concave points (number of concave portions of the contour)

i. Symmetry

j. Fractal dimension (coastline approximation)

Moreover, for the letter-labelled attributes (that are real-valuated features) the data set presents, in addition to the mean value, standard error and the *worst* value (mean of the three largest values). Summing up, the WBCD is composed by 33 features for 569 cells as presented in Figure 1.

2.1 Data Preprocessing

Data set preprocessing is a fundamental phase that needs to be performed before using ML methods. It entails searching within the data set for duplicate, missing data, incorrect data encodings, and redundant or unuseful features. In WBCD, the *ID* column is redundant, while the unnamed column X includes only NaN, so we dropped them. In addition, we have to scale the numerical variables, which leads to the features being adimensional. This step is fundamental to apply techniques such as Principal Component Analysis (PCA), which is scale-dependent, and to improve the performance and convergence rapidity of the learning algorithms we will use to classify tumors. After the preprocessing step, we obtain a data set with a *head* represented in Figure 2. The final data set is composed by 569 records and 31 variables, the categorical one *diagnosis* and the standardized numerical attributs of each cells nucleous.

```
'data.frame':   569 obs. of  33 variables:
 $ id                     : int  842302 842517 84300903 84348301 84358402 843786 844359 84458202 844981 84501001 ...
 $ diagnosis              : chr  "M" "M" "M" "M" ...
 $ radius_mean            : num  18 20.6 19.7 11.4 20.3 ...
 $ texture_mean           : num  10.4 17.8 21.2 20.4 14.3 ...
 $ perimeter_mean         : num  122.8 132.9 130 77.6 135.1 ...
 $ area_mean              : num  1001 1326 1203 386 1297 ...
 $ smoothness_mean        : num  0.1184 0.0847 0.1096 0.1425 0.1003 ...
 $ compactness_mean       : num  0.2776 0.0786 0.1599 0.2839 0.1328 ...
 $ concavity_mean         : num  0.3001 0.0869 0.1974 0.2414 0.198 ...
 $ concave.points_mean    : num  0.1471 0.0702 0.1279 0.1052 0.1043 ...
 $ symmetry_mean          : num  0.242 0.181 0.207 0.26 0.181 ...
 $ fractal_dimension_mean : num  0.0787 0.0567 0.06 0.0974 0.0588 ...
 $ radius_se              : num  1.095 0.543 0.746 0.496 0.757 ...
 $ texture_se             : num  0.905 0.734 0.787 1.156 0.781 ...
 $ perimeter_se           : num  8.59 3.4 4.58 3.44 5.44 ...
 $ area_se                : num  153.4 74.1 94 27.2 94.4 ...
 $ smoothness_se          : num  0.0064 0.00522 0.00615 0.00911 0.01149 ...
 $ compactness_se         : num  0.049 0.0131 0.0401 0.0746 0.0246 ...
 $ concavity_se           : num  0.0537 0.0186 0.0383 0.0566 0.0569 ...
 $ concave.points_se      : num  0.0159 0.0134 0.0206 0.0187 0.0188 ...
 $ symmetry_se            : num  0.03 0.0139 0.0225 0.0596 0.0176 ...
 $ fractal_dimension_se   : num  0.00619 0.00353 0.00457 0.00921 0.00511 ...
 $ radius_worst           : num  25.4 25 23.6 14.9 22.5 ...
 $ texture_worst          : num  17.3 23.4 25.5 26.5 16.7 ...
 $ perimeter_worst        : num  184.6 158.8 152.5 98.9 152.2 ...
 $ area_worst             : num  2019 1956 1709 568 1575 ...
 $ smoothness_worst       : num  0.162 0.124 0.144 0.21 0.137 ...
 $ compactness_worst      : num  0.666 0.187 0.424 0.866 0.205 ...
 $ concavity_worst        : num  0.712 0.242 0.45 0.687 0.4 ...
 $ concave.points_worst   : num  0.265 0.186 0.243 0.258 0.163 ...
 $ symmetry_worst         : num  0.46 0.275 0.361 0.664 0.236 ...
 $ fractal_dimension_worst: num  0.1189 0.089 0.0876 0.173 0.0768 ...
 $ X                      : logi  NA NA NA NA NA NA ...
```

Figure 1: Internal structure of the WBCD.

diagnosis	radius_mean	texture_mean	...	concave.points_worst	symmetry_worst	fractal_dimension_worst
M	1.0960995	-2.0715123	...	2.2940576	2.7482041	1.9353117
M	1.8282120	-0.3533215	...	1.0861286	-0.2436753	0.2809428
M	1.5784992	0.4557859	...	1.9532817	1.1512420	0.2012142
M	-0.7682333	0.2535091	...	2.1738732	6.0407261	4.9306719
M	1.7487579	-1.1508038	...	0.7286181	-0.8675896	-0.3967505
M	-0.4759559	-0.8346009	...	0.9050914	1.7525273	2.2398308

Figure 2: Heading data of the pre-processed WBCD.

2.2 Exploratory Data Analysis

The second step of our study is Exploratory Data Analysis (EDA). It consists of the application of descriptive statistics and graphical tools to explore and understand the data in their entirety. The primary objectives of EDA are to summarize the main characteristics of the data, identify patterns, detect anomalies, test hypotheses, and check assumptions. After EDA, it will be possible to apply more complex modeling techniques. Fist of all we exctract the distribution of diagnosis as depicted in 357 observations, which account for 62.7% of all

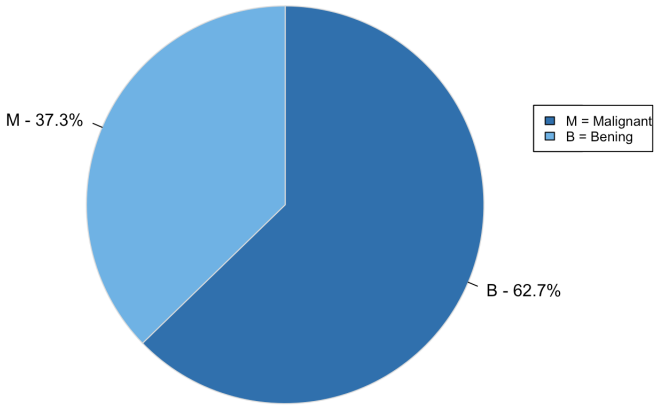

Figure 3: Pie chart of the diagnosis distribution.

observations, indicating the absence of cancer cells, and 212, which account for 37.3% of all observations, shows the presence of cancerous cell. Note that the percentage is unusually large; the dataset does not represents in this case a typical medical analysis distribution. Typically, we have a considerable large number of cases that represents negative (bening tumors) vs. a small number of cases of positives (malignant) tumors.
Now, we study features separately to observe the distributions of benign and malignant tumors. This univariate visualization technique allows us to depict possible trends and understand which features may have a better predictive value concerning the others. In Figure 4 are shown the univariate distributions for all the features in the data set. It is easy to see that most of them are Gaussian distributed. The separation between malignant and benign is not well determined, except for *concave.points_worst*, *concavity_worst*, *perimeter_worst*, *area_mean*, and *perimeter_mean*. We can guess that those features contribute more to the classification of tumors, and they will be of primary importance in the modeling we will present in the next part of this work.

We are also interested in studying possible relationships between the 30 predictors. Therefore, we compute the correlation matrix of the predictors. It shows the linear dependencies of paired features and illustrates which variables are highly correlated. This step is fundamental as it helps in building better predictive models. Indeed, highly correlated predictors

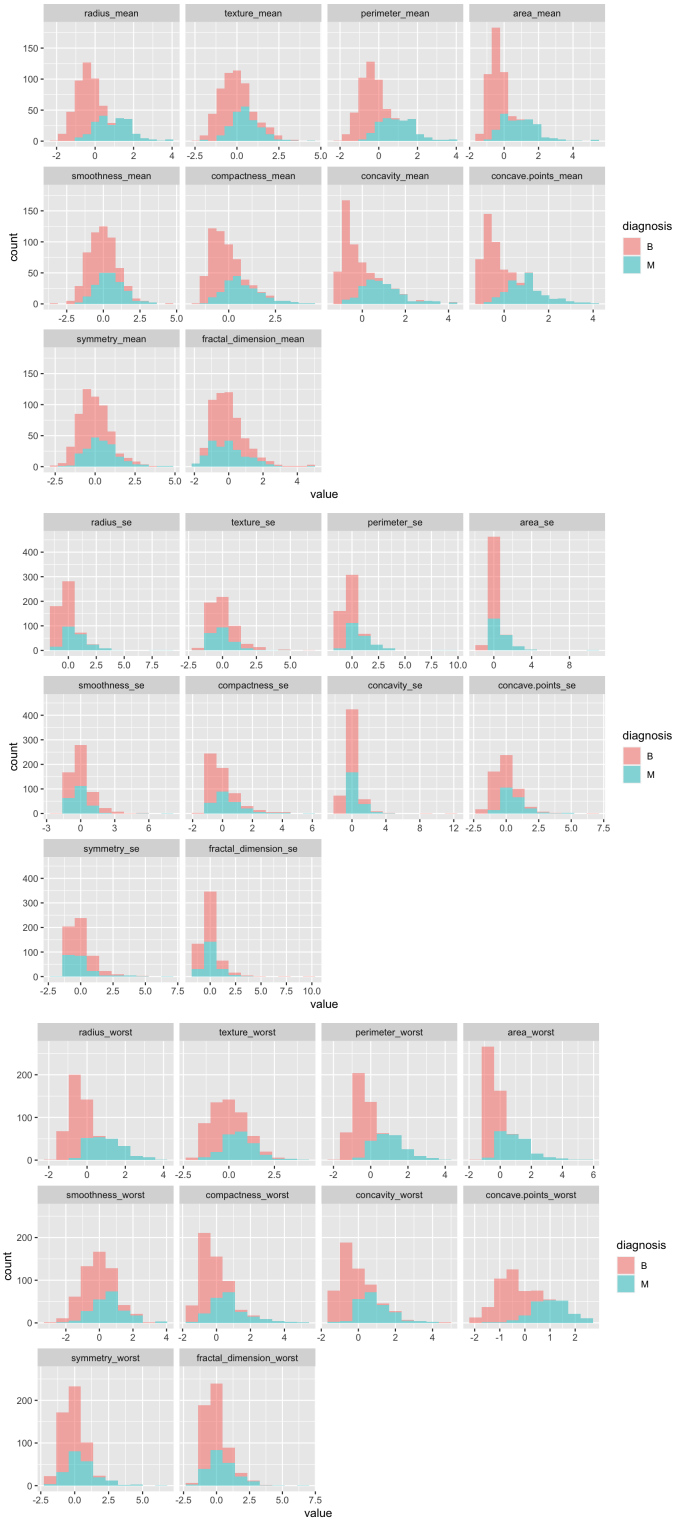

Figure 4: Pie chart of the diagnosis distribution.

can lead to multicollinearity in regression models, affecting the models' interpretability and performance. The correlation matrix of our data set is in Figure 5. As we can see, the data set presents several correlated predictors. The elimination of highly correlated features can be computed by hand, imposing a cutoff on correlation value and decoupling predictors. Doing this with a correlation cutoff of 0.9 we would delete 10 variables.

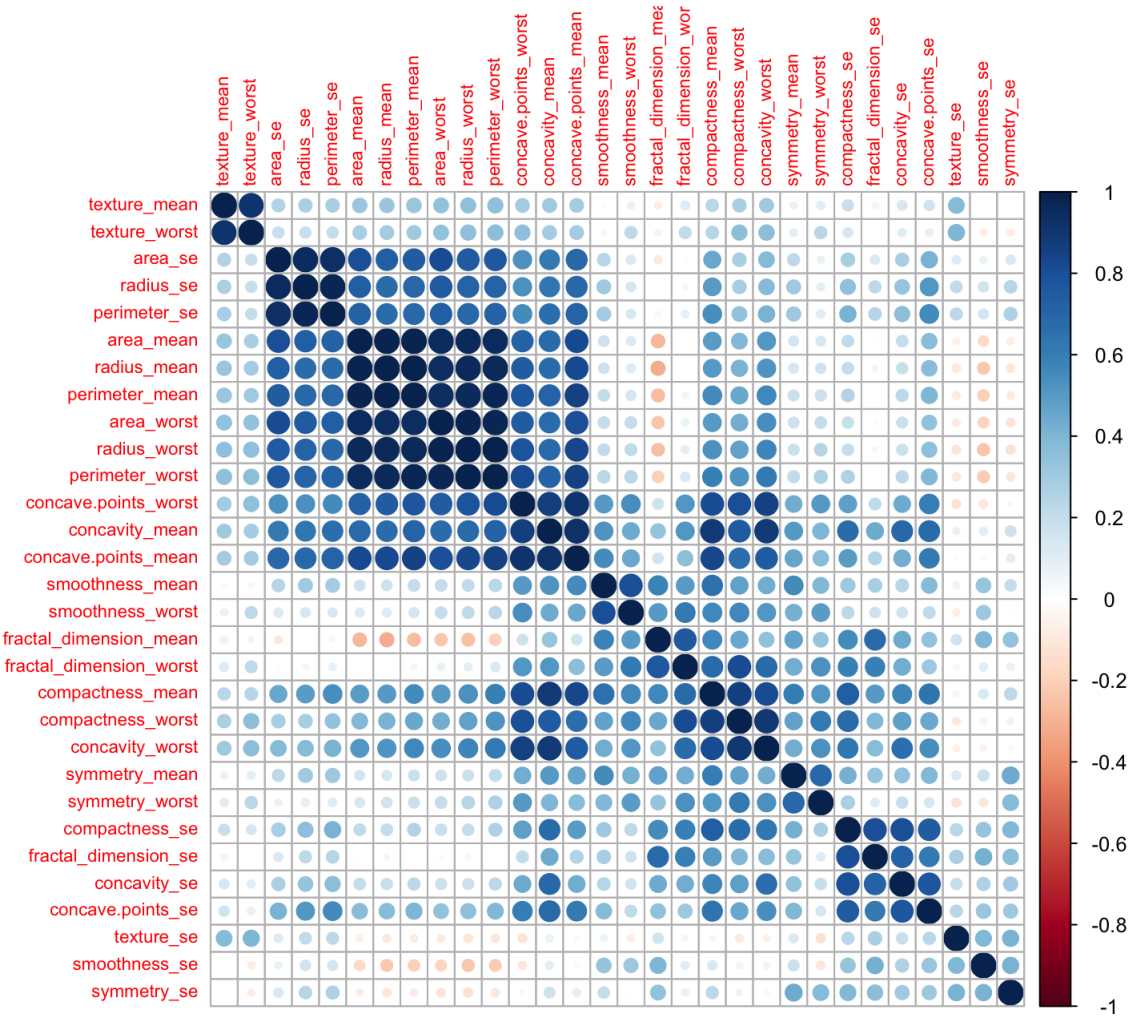

Figure 5: Correlation matrix.

A more sensitive and efficient method to perform Dimension Reduction is Principal Component Analysis (PCA).

2.3 Unsupervised Dimension Reduction with PCA

Principal Component Analysis (PCA) is a Dimension Reduction technique for unsupervised problems. It projects the data into a lower-dimensional space so that the lower-dimensional

representation retains some meaningful properties of the original data, ideally close to its intrinsic dimension. PCA allows to de-noise the data by removing negligible variation; it captures the main signals, patterns and structure in our multivariate WBCD without any prior target or specified predictive aim. PCA, finally, returns a data set reduced in size and cleaned of possible noise on which it is easier and more efficient the application of supervised classification models.

Formally PCA is a spectral decomposition of the covariace matrix associated to the data set. Eigenvectors of the decomposition are the *Principal Components* (directions), which are orthogonal and represent the new span basis of the initial data set. The associated eigenvalues, instead, represent the importance of each direction (Principal Component) in terms of variance. Principal Components are classified with respect to their importance; the first component is the one associated to the largest eigenvalues. This tells that along that direction it is projected the greater part of the variance of the date set. Rughly speaking, it explains the majority of the information and patterns of the data set. To select the right numbers of Principal Components we use *Percentage of Variance Explained* (PVE). PVE is an index that express the percentage of variance that is associated to each principal component. It is common to select a number of Principal Components that cover a PVE between 0.80 and 0.90. PVE for the i-th component is defined as follows:

$$PVE_i = \frac{\mathrm{Var}(\phi_i^T X)}{\sum_{k=i}^{p} \mathrm{Var}(\phi_k^T X)} = \frac{\lambda_i}{\sum_{k=1}^{p} \lambda_k} \tag{1}$$

where ϕ_i are the eigenvectors associated to the λ_i eigenvalue, and k goes from 1 to p = number of original predictors X.

In our specific case, we provided that the first three Principal Components explain the 73% of the variance of the system as represented by the scree plot in Figure 6 and Figure 7, which is the cumulative PVE (CPVE), defined as:

$$CPVE = \sum_{i}^{m} PVE_i \qquad m = \text{ number of eigenvectors} \tag{2}$$

To explain the 95% of the variance of the system, we need to select the first eight Principal Components, while we need 13 PCs to understand the 99% of the variance.

It is useful plotting the *correlation circle* of the first two Principal Components, PC1 and PC2. In these plot, positively correlated variables have the same direction (small angle in between). Negatively correlated variables have opposite directions. Uncorrelated variables are orthogonal. From Figure 8 $fractal\_dimension\_mean$ and *concavity_mean* well approximate the PC1 and PC2 respectively. On the other hand, *radius*-like, *area*-like, and *perimeter*-like are a linear combination of the PCs, and they are highly correlated as already seen in the correlation matrix, Figure 5.

Performed the PCA, we can show the new rotation matrix obtained from the lower dimensional space spanned by PCA construction. It shows the projection of the starting features on Principal Components; Figure 9. As already seen, the first eight PCs cover over the 95% of the variance. They concern all the information about the data set, avoiding multicollinearity

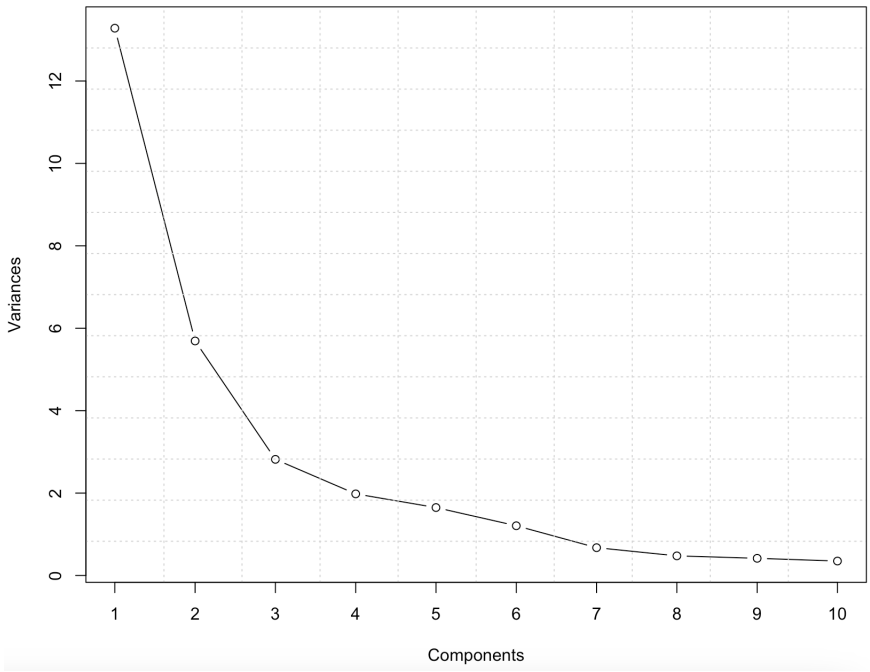

Figure 6: Scree Plot for PVE on breast cancer data. The first three Principal Components cover the 73% of the entire variance of the system. To cover the 95% we need the first eight PCs .

and inner dependencies among them. The results of the PCA indicate that dimensionality reduction can simplify the complexity of the dataset without sacrificing important information. This dimensionality reduction has reduced noise and multicollinearity among features, and now we can apply supervised machine learning models to classify breast cancer diagnoses accurately. In the next section, we will present the different algorithms we tested on WBCD to analyze their performance and differences in the breast cancer classification problem.

2.4 Supervised Classification Models

Classification models are generalization and imrpovement of the common *regression models*. In regression models, the response variable Y must be quantitative. Classification models, instead, allow predictions also for qualitative responses. We test different classification techniques to classify the categorical response variable *diagnosis* of the WBCD. In particular, we apply *k-Nearest Neighbors*, *Linear Discriminat Analysis*, *LASSO logistic regression*, and the more computer-intensive method *Support Vector Machines*.

As for a linear regression model, in the classification setting, we divide the entire data set into a *training set* and a *test set*. In our case, we divide the WBCD randomly such that the training set is the 70% of the entire data set and the test set, the remaining 30%. Even if the specific architecture of the classification procedure depends on the algorithm used, the timeline of a Supervised Classification Model (SCM) is generally summarizable as follows:

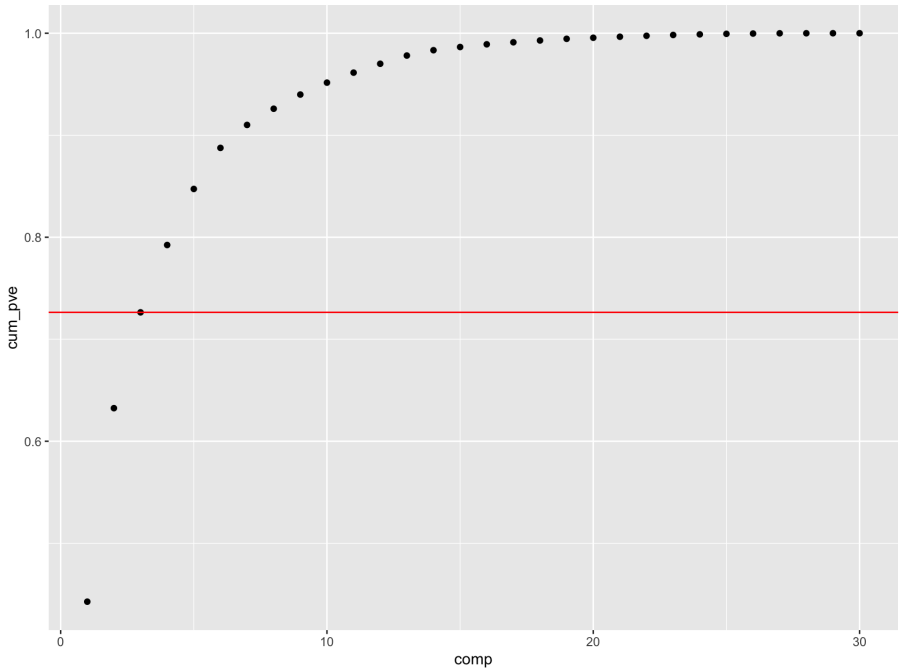

Figure 7: Scree Plot for cumulative PVE on breast cancer data. The first three Principal Components cover the 73% of the entire variance of the system. To cover the 95% we need the first eight PCs

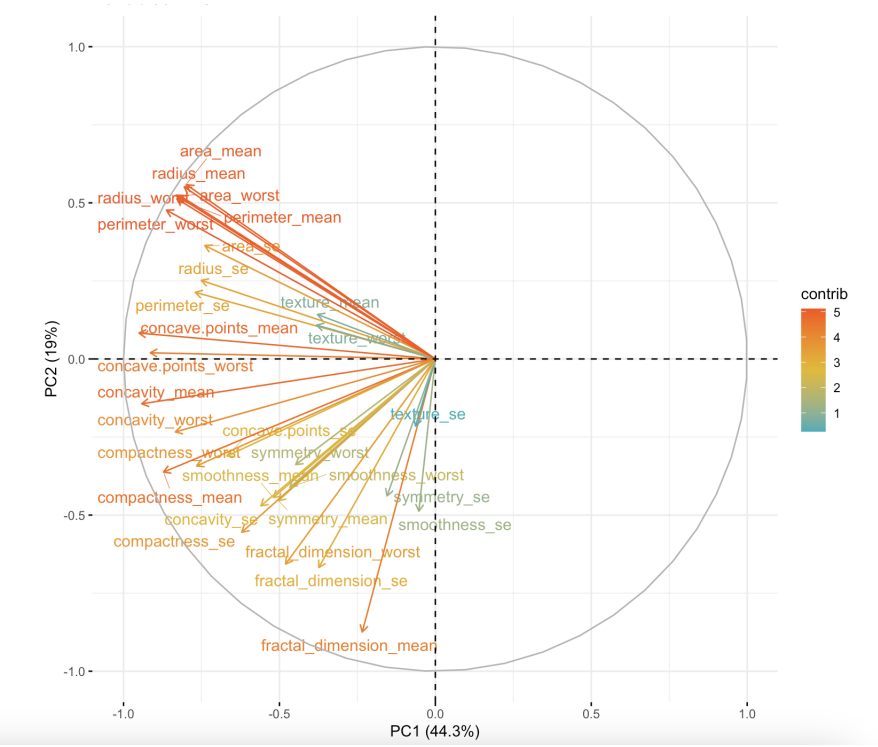

Figure 8: Correlation circle that represents the first two Principal Components of the system.

1. **Training process** consists in a optimization problems on the training set. For different learning algorithms, we provide the parameters that extremize a specific function (e.g., the log-likelihood in a Generalized Linear Model). In general, the parametric functions are *cost functions* or *residual functions*, a parametric function that measures the difference between predicted and actual labels.

2. **Validation on training set** is a process that cures possible overfitting, or selection bias of the system. In this study, we use *Cross-Validation* (CV) as a validation technique. CV consists of iterative resampling and sample splitting methods with different subsets of data to test and train a model. It estimates the accuracy of the specific predictive model and the stability of its parameters.

3. **Evaluation Metrics on Test set**. Metrics that tell the goodness of a classifier are: *accuracy*, *precision*, *recall*, and *F1-Score*. We compute the Confusion Matrix for each model to analyze all the metrics listed before.

4. **Prediction process**. Trained the algorithms and tuned by cross-validation, we can provide the value of the response variable for any labels with the same predictors.

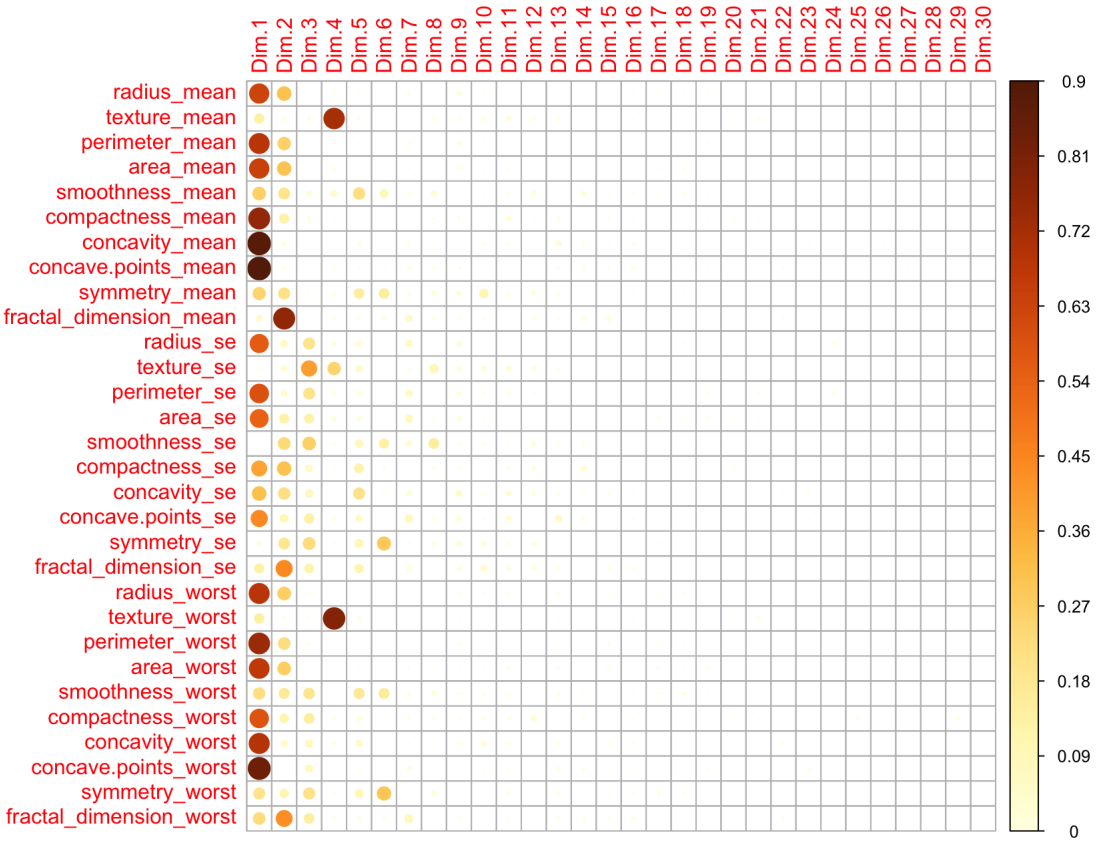

Figure 9: Totation matrix of the initial data on Principal Components.

2.4.1 Linear Discriminant Analysis

The first classifier we test is Linear Discriminant Analysis (LDA). It is more efficient than the ordinary *logistic regression*, especially when the classes of the response variable are well-separated, and the parameter estimates for a logistic regression model are unstable [1]. The mathematical basis of LDA is the *Bayes' Theorem*:

$$\Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)} \tag{3}$$

where Y is the response variable *diagnosis* with $K = 2$ classes (M = malignant, B = bening), and X is the set of 30 predictors. π_k is the prior probability that a randomly chosen observation comes from the kth class, that is just the fraction between the overall number of records and the cardinality of each class. Finally, $f_k(x)$ is the density function of X for an observation from the kth class. We assume that $f_k(x)$ follows a Guassian ditribution with mean μ_k and variance σ^2, as it is shown by Figure 4. Under this assumption, the LDA method provides predictions by approximating the Bayes classifier 3 by the estimation of π_k, μ_k, and σ^2. The explicit form of the multivariate (predictors $p > 1$) Gaussian density $f_k(x)$ for the kth class is:

$$f_k(x) = \frac{1}{(2\pi)^{p/2}|\text{Cov}(X)|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_k)^T \text{Cov}(X)^{-1}(x-\mu_k)\right) \tag{4}$$

In the following, we show the *R*-code used to implement Linear Discriminant Analysis on WBCD:

```
# Defining TRAINING and TEST data
set.seed(42)
train_index <- createDataPartition(data_wdbc$diagnosis, p = 0.7, list = FALSE)
data_train <- data_wdbc[train_index,]
data_test <- data_wdbc[-train_index,]

# LDA model
fitControl <- trainControl(method = "cv", number = 10, classProbs = TRUE,
    summaryFunction = twoClassSummary)

ldaFit <- train(diagnosis ~ ., data = data_train, method = "lda", trControl =
    fitControl)
ldaPredict <- predict(ldaFit, newdata = data_test)
lda.class <- ldaPredict
```

We performed the analysis using the built-in package *train* with *lda method*. Otherwise, one can use the *lda* function from *class* library. In Section 3 we will discuss the results of the LDA model applied to the WBCD, showing the corresponding confusion matrix and the *ROC curve*. The next classifier we will present is the *k-Nearest Neighbors*.

[1] The explicit proof of this is outside the interest of this report, but we refer to [3] for a complete discussion.

2.4.2 k-Nearest Neighbours

In statistics, the k-nearest neighbors algorithm (kNN) is a non-parametric supervised classification method developed by Evelyn Fix and Joseph Hodges in 1951 and later expanded by Thomas Cover. In kNN classification, the output is a class membership, and an object is classified by a plurality vote of its neighbors, with the object assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). The vote is assigned through a specific distance function, e.g., the Euclidean, or the Hamming distance. Since kNN relies on distance for classification, it is scale-dependent and sensitive to the local structure of the data, such as the presence of noise or ininfluent data (not the case for WBCD). Formally, the kNN algorithm is defined as follows:

1) Compute distances $d(x_i, x_j)$ between each records x_i, x_j of the training set among all the corresponfing features.

2) Sort distances in ascending order and select the k nearest neighbors.

3) The predicted class $\hat{y}$ label for x_{test} is determined by the majority class among the k nearest neighbors:

$$\hat{y} = mode\{y_1, y_2, \ldots, y_k\} \tag{5}$$

The best choice of k depends upon the data; generally, larger values of k reduce the effect of the noise on the classification but make boundaries between classes less distinct. From cross-validation, we tuned the best value for the k neighbors; as shown in Figure 10, $k = 9$. In the following, the R-code used for the kNN classification.

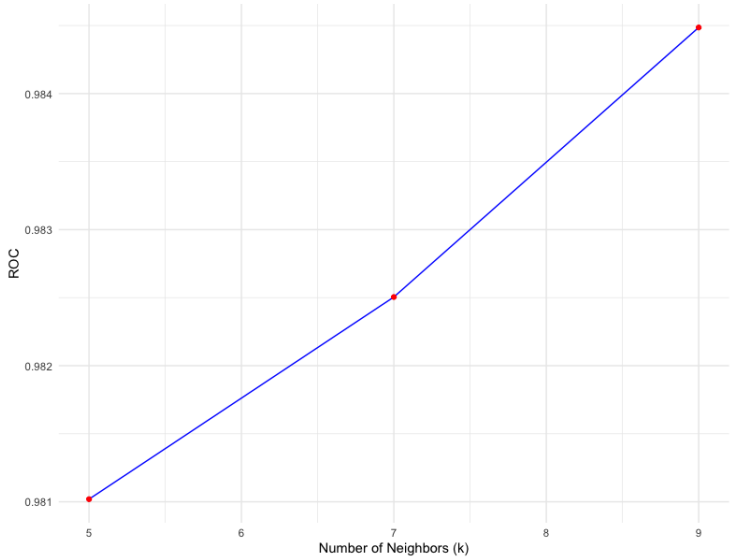

Figure 10: Cross-Validation process for the k neighbours. $k = 9$.

```r
# kNN model
fitControl <- trainControl(method = "cv", number = 10, classProbs = TRUE,
    summaryFunction = twoClassSummary)

knnFit <- train(diagnosis ~ ., data = data_train, method = "knn", trControl =
    fitControl)
knnPredict <- predict(knnFit, newdata = data_test)
knn_confusion <- confusionMatrix(knnPredict, data_test$diagnosis, positive =
    "M")

print(knnFit$bestTune)
print(knnFit$results)
```

We performed classification using the *train* function, with the *knn* method, the same analysis is available with the *knn* function of library *class*.

2.4.3 Logistic regression with LASSO penalization

Logistic regression with LASSO (Least Absolute Shrinkage and Selection Operator) penalization is a logistic regression technique that adds a penalty based on the sum of the absolute values of the coefficients in the model. This induces a sparsity in the coefficients, bringing some of them to zero and then automatically performing variable selection. It performs both variable selection and regularization to enhance the prediction accuracy and interpretability of the resulting statistical model. Formally the LASSO logistic regression is an optimization model in the family of the Generalized Linear Models (GLM). The peculiarity of the approach stands in a specific positive definite term in the residual function, and, therefore, in the GLM likelihood. This new term, called LASSO penalization, is dominated by the LASSO coefficient λ that defines the sparsity of the model. The LASSO logistic regression starts with the estimation of the best λ, by cross-validation. Then, the regression part works on the optimization of the following residual equation:

$$LASSO_{res} \equiv \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \tag{6}$$

Which is a quadratic residual plus the LASSO penalty coefficient, with n = number of records and p the number of predictors. LASSO logistic regression introduces a regularization parameter that lowers the dimensionality of the problems, simplifying the analysis and reducing the risk of overfitting. Moreover, a more sparsity model requires fewer computational resources during the prediction and training phase.

As we will see in the next Section 3, this kind of classifier works well on WBCD. We use the "train" package with the "glmnet" method and "alpha = 1" (penalty parameter for LASSO logistic regression) to perform the LASSO logistic regression:

```r
# LASSO regression model
fitControl <- trainControl(method = "cv", number = 10, classProbs = TRUE,
    summaryFunction = twoClassSummary)

lassoFit <- train(diagnosis ~ .,
                              data = data_train,
                              method = "glmnet",
                              trControl = fitControl,
                              tuneGrid = expand.grid(alpha = 1, lambda =
                                  seq(0.001, 0.1, by = 0.001)))
lassoPredict <- predict(lassoFit, newdata = data_test, type = "prob")
```

Using Cross-Validation as "fit-control" we ensure to provide prediction with the best value for the LASSO parameter λ, which is the one that balances well the complexity of the model and its ability to fit the data. The specific selection of λ results from the minimization of the *Binomial Deviance* D that is defined as follows:

$$D = -2(\log L - \log L_{saturated}) \tag{7}$$

L is the likelihood function of the LASSO-penalized logistic model 6, and $L_{saturated}$ is the likelyhood of the saturated model, $p_i = y_i$, see [4]. In Figure 11, we show the plot for the Binomial Deviance D against $\log \lambda$, from which we extract the λ value that minimizes D, which is $\lambda = 0.0049$. From Figure 11, we see that the balance between complexity/sparsity and parameters best fit introduces a regularization that switches off all the features except for nine. These features represent the more informative ones and span a new unbiased data set. The features are listed in Figure 12.

In general, we could apply a logistic regression without LASSO or RIDGE penalization in the reduced data set spanned by the features selected by LASSO penalization. For our specific simple case, we noticed that the results are invariant. In general, this is not the case, and combining two logistics models with and without penalization provides the best performance in classification problems

2.4.4 Support Vector Machines

Support Vector Machines (SVMs) are a supervised model for classification and regression analysis. Developed by Vladimir Vapnik and his colleagues at AT&T Bell Laboratories, SVMs have become among the most extensively studied models in the field, with Random Forest and Neural Networks. SVMs are not limited to linear classification; they can also perform non-linear classification efficiently using the kernel trick. This technique represents data through pairwise similarity comparisons between the original observations, transforming the data into a higher-dimensional feature space. SVMs are robust against noisy data, making them highly resilient and reliable for various applications. For a complete and detailed analysis of SVMs, we refer to the following references: [6] and [5].

In the following we attach the *R*-code for implementing the algorithm on the breast cancer data set. As before, we use cross-validation for the fine-tuning of the parameters, and a *linear kernel*.

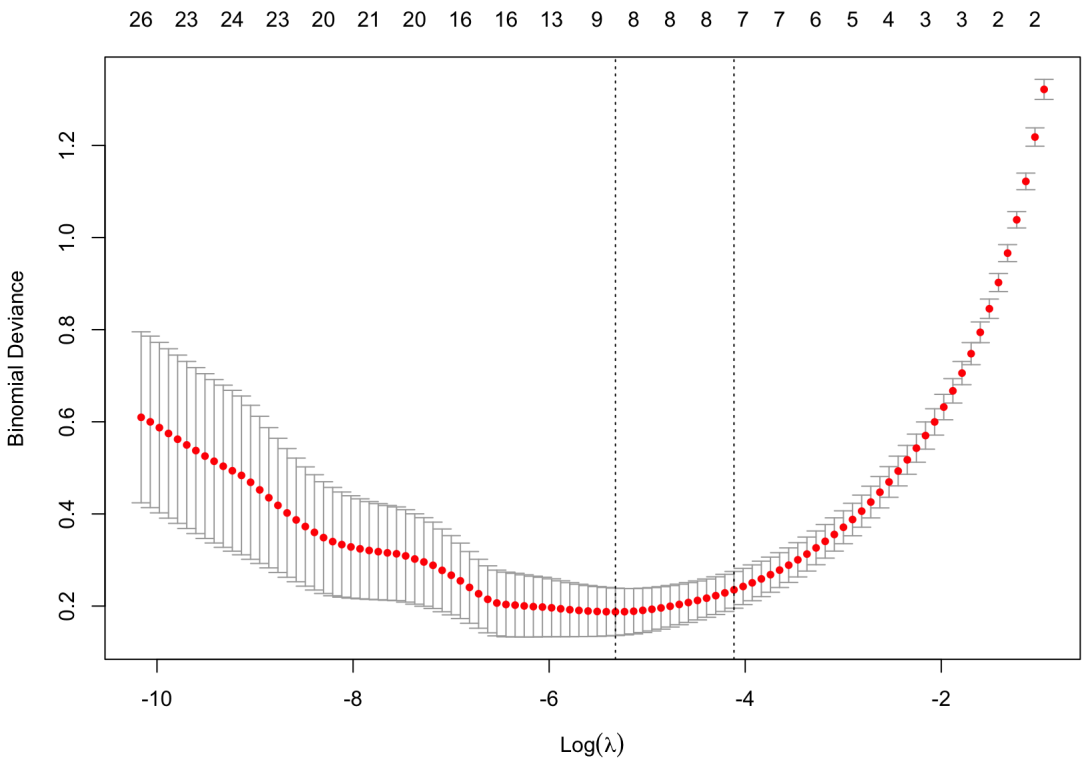

Figure 11: Plot of the Binomial Deviance D against $\log \lambda$ for LASSO logistic regression model.

```
> print(selected.variables)
[1] "concave.points_mean"  "radius_se"            "fractal_dimension_se"
[4] "radius_worst"         "texture_worst"        "smoothness_worst"
[7] "concavity_worst"      "concave.points_worst" "symmetry_worst"
```

Figure 12: Selected features of the regularization introduced by LASSO penalization.

```r
# SVMs algorithm
fitControl <- trainControl(method = "cv", number = 10, classProbs = TRUE,
    summaryFunction = twoClassSummary)

svmFit <- train(diagnosis ~ ., data = data_train, method = "svmLinear",
    trControl = fitControl)
svmPredict <- predict(svmFit, newdata = data_test)
```

Now, we discuss the specific results of the different algorithms described before.

3 Results

In this section, we illustrate the results of the classifiers. We show the confusion matrices, which show the summary of the performance of the corresponding model. In particular, the monfusion matrices display:

- The matrix with the portions of correct/incorrect predictions on positive and negative tumors against the real number of each class.
- *Accuracy*: It measures the proportion of correct prediction among the total sample.
- *Precision*: Proportion of correct *positive* prediction on all positive ones (True Positive + False Positive). We refer to M = malignant tumor for *positive*.
- *Recall*: Proportion of correct *positive* prediction on all real positive (True Positive + False Negative).
- *F1-Score*: Defined as: $\frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$
- *Specificity*: Tax of Real Negative Predictions and Tax of False Positive Prediction.

Together with the Confusion Matrix, we show the *ROC* curve (Receiver Operating Characteristic). In decision theory, ROC represents a graphical scheme for a binary classifier. The two axes represent the True Positive Rate (TPR, fraction of true positives; the Recall of Confusion Matrix = 1−Specificity) and False Positive Rate (FPR, fraction of false positives). A ROC curve is the graph of the set of pairs (FP, TP) as a classifier parameter changes (the threshold for a positive or negative decision). Therefore, through ROC curve analysis, the ability of the classifier to discern through Malignant and Benign tumors is assessed by calculating the area under the ROC curve (AUC). The value of AUC is between 0 and 1, where 0 stands for an inefficient classifier and 1 is the perfect classifier. See at [7] for a more general treatment of ROC topics.

3.1 Metrics and ROC comparison

Following the order of appearance, we show Confusion Matrix and ROC curve for all the predicting models we applied to our breast cancer data set. After the graphs, we will briefly discuss the results, underlining differences between models and comparing their metrics.

LDA Results

```
> lda.confusion <- confusionMatrix(data = lda.class, reference = data_std$diagnosis, positive = "M")
> lda.confusion
Confusion Matrix and Statistics

          Reference
Prediction   B   M
         B 355  18
         M   2 194

               Accuracy : 0.9649
                 95% CI : (0.9462, 0.9784)
    No Information Rate : 0.6274
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9236

 Mcnemar's Test P-Value : 0.0007962

            Sensitivity : 0.9151
            Specificity : 0.9944
         Pos Pred Value : 0.9898
         Neg Pred Value : 0.9517
             Prevalence : 0.3726
         Detection Rate : 0.3409
   Detection Prevalence : 0.3445
      Balanced Accuracy : 0.9547

       'Positive' Class : M
```

Figure 13: Complete Confusion Matrix for the LDA classifier.

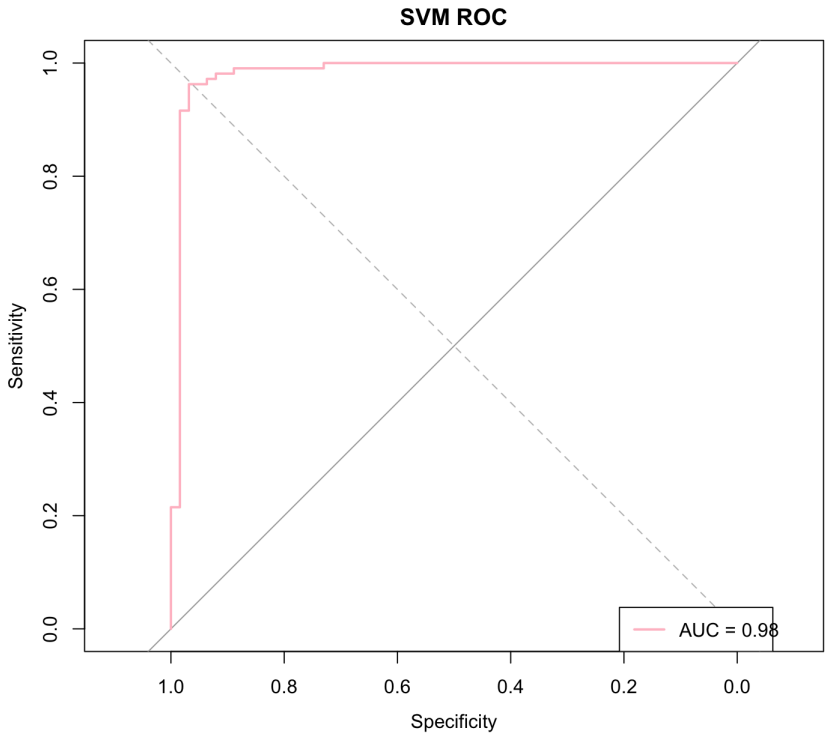

Figure 14: ROC curve plot for the LDA classifier, with the computation of the corresponding AUC.

kNN Results

```
> knn_confusion <- confusionMatrix(knnPredict, data_test$diagnosis, positive = "M")
> knn_confusion
Confusion Matrix and Statistics

          Reference
Prediction   B   M
         B 103   5
         M   4  58

               Accuracy : 0.9471
                 95% CI : (0.9019, 0.9755)
    No Information Rate : 0.6294
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8861

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9206
            Specificity : 0.9626
         Pos Pred Value : 0.9355
         Neg Pred Value : 0.9537
             Prevalence : 0.3706
         Detection Rate : 0.3412
   Detection Prevalence : 0.3647
      Balanced Accuracy : 0.9416

       'Positive' Class : M
```

Figure 15: Complete Confusion Matrix for the kNN classifier.

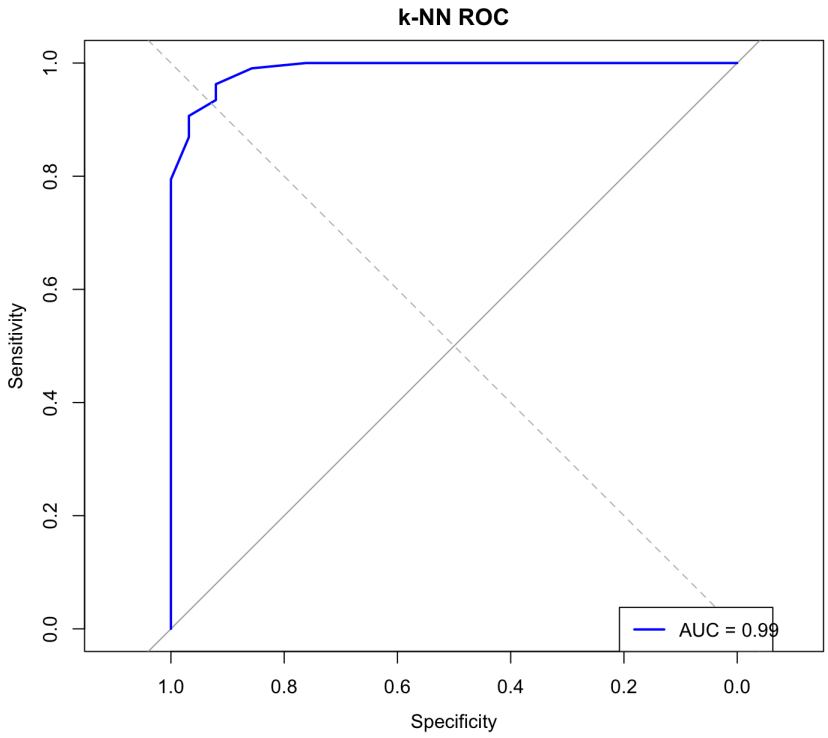

Figure 16: ROC curve plot for the kNN classifier, with the computation of the corresponding AUC.

LASSO logistic regression Results

```
> lasso.confusion <- confusionMatrix(data = as.factor(lasso.pred), as.factor(data_test$diagnosis), positive = "M")
> lasso.confusion
Confusion Matrix and Statistics

          Reference
Prediction   B   M
         B 105   3
         M   2  60

               Accuracy : 0.9706
                 95% CI : (0.9327, 0.9904)
    No Information Rate : 0.6294
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9367

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9524
            Specificity : 0.9813
         Pos Pred Value : 0.9677
         Neg Pred Value : 0.9722
             Prevalence : 0.3706
         Detection Rate : 0.3529
   Detection Prevalence : 0.3647
      Balanced Accuracy : 0.9668

       'Positive' Class : M
```

Figure 17: Complete Confusion Matrix for the LASSO-logistic regression classifier.

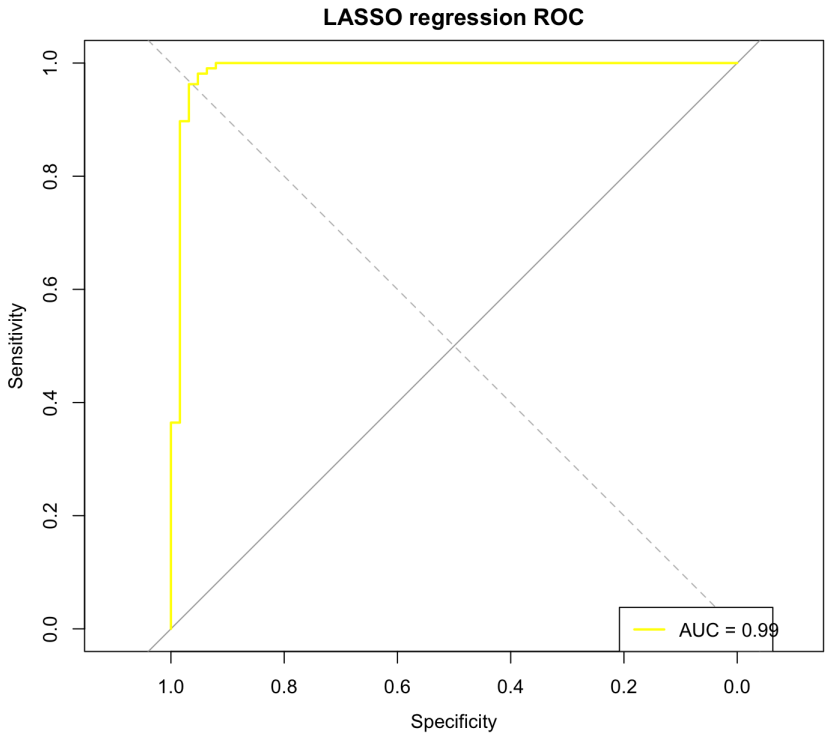

Figure 18: ROC curve plot for the LASSO logistic regression classifier, with the computation of the corresponding AUC.

SVMs Results

```
> svm_confusion <- confusionMatrix(svmPredict, data_test$diagnosis, positive = "M")
> svm_confusion
Confusion Matrix and Statistics

          Reference
Prediction   B   M
         B 106   2
         M   1  61

               Accuracy : 0.9824
                 95% CI : (0.9493, 0.9963)
    No Information Rate : 0.6294
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.962

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9683
            Specificity : 0.9907
         Pos Pred Value : 0.9839
         Neg Pred Value : 0.9815
             Prevalence : 0.3706
         Detection Rate : 0.3588
   Detection Prevalence : 0.3647
      Balanced Accuracy : 0.9795

       'Positive' Class : M
```

Figure 19: Complete Confusion Matrix for the SVMs classifierl.

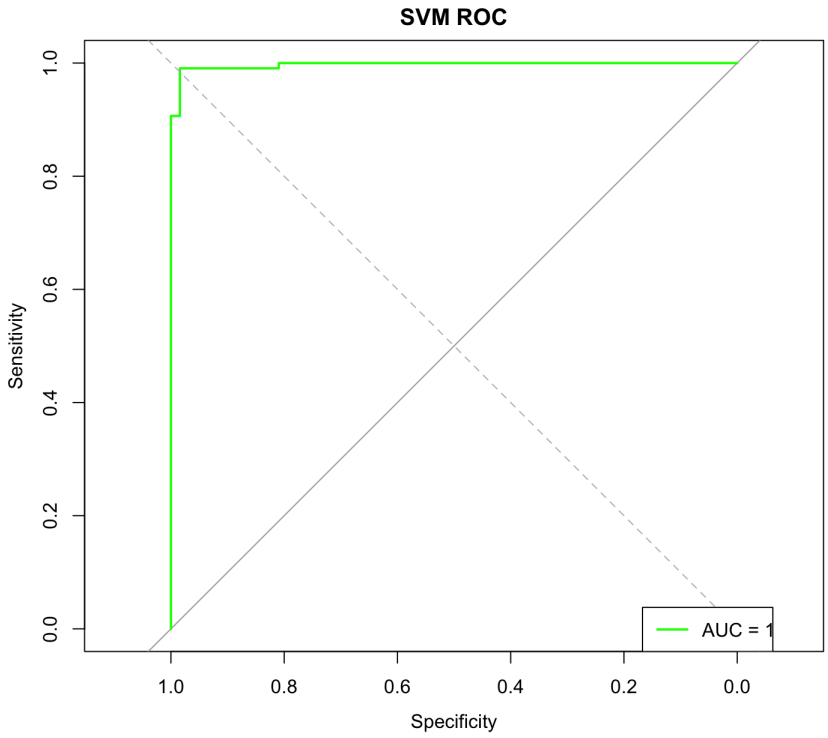

Figure 20: ROC curve plot for the SVMs classifier, with the computation of the corresponding AUC.

Comparison

In Table 3.1 are presented the metrics' values for the corresponding classification model. Finally, we show the results in the histogram plotted in Figure 21. For each classifier, we

	Model	Accuracy	Precision	Recall	F1
1	SVM	0.98	0.98	0.97	0.98
2	kNN	0.95	0.94	0.92	0.93
3	LDA	0.96	0.99	0.92	0.95
4	Lasso	0.97	0.97	0.95	0.96

Table 1: Table of metrics results for our supervised classifiers.

display four bars representing the metrics in Table 3.1. Even for a small gap, the histogram in Figure 21 clearly shows that SVMs provide the overall best efficiency in classifying Malignant and Bening breast cancer.

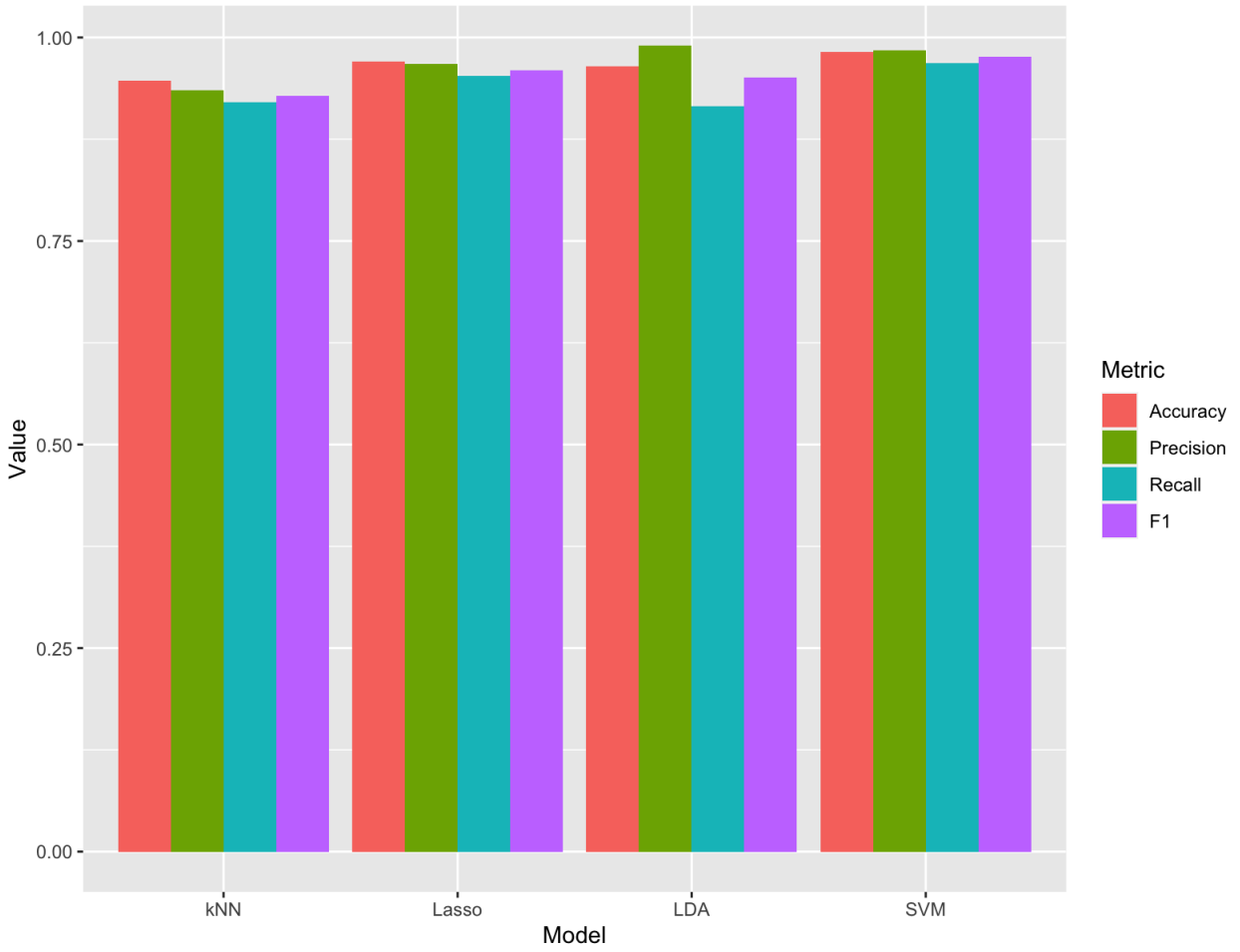

Figure 21: Histogram for metrics comparison of: LDA, kNN, logistic regression with LASSO penalization and SVMs classifier on breast cancer data.

4 Discussion

From the comparative analysis of the four supervised classifier models (LDA, kNN, LASSO logistic regression, and SVM) that we studied, we can extrapolate valuable insights into their effectiveness and reliability. SVMs stand out for their outstanding accuracy and precision of 98%, making them the most reliable model for distinguishing between benign and malignant tumors. Moreover, the F1-score, which balances accuracy and recall, suggests that SVMs are also the most balanced models, with a score of 98%. kNN and LDA show competitive results, with close to 95% accuracy, but with slight differences in other metrics. However, the LDA model generally performs better than kNN due to the possibility of a linear splitting of the response, as well as the Gaussian behavior of the predictors. Finally, LASSO logistic regression ranks just below the performance of SVMs, showing that of the many predictors, only a few are informative for tumor classifications. Regression with the LASSO penalization also provides an automatic regularization of the data set, which, in this case, proves to be a good feature over simpler models such as LDA and kNN.

However, it is fundamental to note that the choice of model also depends on the context and on the specific goal of the analysis. For example, you may need to choose a model that maximizes recall at the expense of accuracy or precision. In addition, practical considerations such as model complexity and ease of interpretation may influence the decision.

In the future, we are interested in moving the analysis outside the diagnosis prediction context and testing supervised and unsupervised learning in data sets describing the clinical pathways for generic chronic patients (IMA, diabets). Using statistical learning algorithms, we want to mine the clinical pathways for chronic patients from data concerning hospitalization, pharmaceutical prescriptions, and ambulatory services. We think that such analysis will provide several contributions to the healthcare sectors both from a pure-medical point of view, and a managing point of view.

References

[1] World Health Organization (WHO), 2024. *https://www.who.int/news-room/fact-sheets/detail/breast-cancer.*

[2] Martínez-Campa C, Menéndez-Menéndez J, Alonso-González C, González A, Álvarez-García V, Cos S. *What is known about melatonin, chemotherapy and altered gene expression in breast cancer.* Oncol Lett. 2017.

[3] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, Jonathan Taylor. *An Introduction to Statistical Learning: with Applications in R.* New York: Springer, 2013.

[4] Liu Congcong, Gao Jinsong, Liu Juntao, Wang Xietong, He Jing, Sun Jingxia, Liu Xiaowei, and Liao Shixiu. *Predictors of Failed Intrauterine Balloon Tamponade in the Management of Severe Postpartum Hemorrhage.* Frontiers in Medicine, Vol.8. 2021.

[5] Bennett Kristin P. and Campbell Colin. *Support Vector Machines: Hype or Hallelujah?* SIGKDD Explorations, Vol. 2. 2000

[6] Cortes C. and Vapnik V. *Support-vector networks.* Machine Learning, Vol. 20, 273–297. 1995.

[7] Ezio Bottarelli and Stefano Parodi. *Un approccio per la valutazione della validità dei test diagnostici: le curve R.O.C. (Receiver Operating Characteristic).* Ann. Fac. Medic. Vet. di Parma Vol. XXIII. 2003.

[8] Francesca Ferré, Chiara Seghieri, Andrea Burattin and Andrea Vandin *Process Mining and Clinical Pathways: an application to Breast cancer data in Tuscany.* Proceedings of 4th International Workshop on Process-Oriented Data Science for Healthcare. 2021