HarvardX: PH125.9x Data Science
Breast Cancer Wisconsin (Diagnostic)

Wilber Acuña-Bravo

18 June, 2025

Contents

1 Introduction

Breast cancer remains a leading cause of mortality among women worldwide, with early and accurate diagnosis significantly improving survival rates. This report details a comprehensive machine learning approach to classify breast tumors as malignant or benign using the Wisconsin Diagnostic Breast Cancer (WDBC) dataset. This dataset (Street, Wolberg, and Mangasarian 1993), curated by the University of Wisconsin, comprises 569 samples derived from digitized images of fine needle aspirates (FNA) of breast masses, each characterized by 30 quantitative features describing cellular nuclei properties (e.g., radius, texture, concavity). learning models for breast cancer detection using a dedicated dataset.

1.1 The Training dataset

The data set can be downloaded from here (link). According to data description provided in the file "wdbc.names", "Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image." From each cell nucleus image a set of features was obtained:

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)

The mean, standard error, and "worst" or largest (mean of the three largest values) of these features were computed for each image

For each image, basic statistic parameters were calculated from these features, this is, the mean, standard error, and "worst" or largest (mean of the three largest values). This resulted in a data set of 30 characteristics for each cell image. The basic and the additional features were arranged, by blocks of 10 (means, standard errors, worst), as columns of the data set. A first column containing the ID number and a second one, containing a "Diagnosis" (Malignant, Benign) complete the columns in the data set.

Table 1: Summary of missing data

variable	q_zeros	p_zeros	q_na	p_na	q_inf	p_inf	type	unique
iD	0	0.00	0	0	0	0	numeric	569
diagnosis	0	0.00	0	0	0	0	factor	2
avg_radius	0	0.00	0	0	0	0	numeric	456
avg_texture	0	0.00	0	0	0	0	numeric	479
avg_perimeter	0	0.00	0	0	0	0	numeric	522
avg_area	0	0.00	0	0	0	0	numeric	539
avg_smoothness	0	0.00	0	0	0	0	numeric	474
avg_compactness	0	0.00	0	0	0	0	numeric	537
avg_concavity	13	2.28	0	0	0	0	numeric	537
avg_concave_points	13	2.28	0	0	0	0	numeric	542
avg_symmetry	0	0.00	0	0	0	0	numeric	432
avg_fractal_dim	0	0.00	0	0	0	0	numeric	499
se_radius	0	0.00	0	0	0	0	numeric	540
se_texture	0	0.00	0	0	0	0	numeric	519
se_perimeter	0	0.00	0	0	0	0	numeric	533
se_area	0	0.00	0	0	0	0	numeric	528
se_smoothness	0	0.00	0	0	0	0	numeric	547
se_compactness	0	0.00	0	0	0	0	numeric	541
se_concavity	13	2.28	0	0	0	0	numeric	533
se_concave_points	13	2.28	0	0	0	0	numeric	507
se_symmetry	0	0.00	0	0	0	0	numeric	498
se_fractal_dim	0	0.00	0	0	0	0	numeric	545
w_radius	0	0.00	0	0	0	0	numeric	457
w_texture	0	0.00	0	0	0	0	numeric	511
w_perimeter	0	0.00	0	0	0	0	numeric	514
w_area	0	0.00	0	0	0	0	numeric	544
w_smoothness	0	0.00	0	0	0	0	numeric	411
w_compactness	0	0.00	0	0	0	0	numeric	529
w_concavity	13	2.28	0	0	0	0	numeric	539
w_concave_points	13	2.28	0	0	0	0	numeric	492
w_symmetry	0	0.00	0	0	0	0	numeric	500
w_fractal_dim	0	0.00	0	0	0	0	numeric	535

The Table 1 summarizes the contents of the data set. As can be seen there are 0 missing values as well as 0 infinity values.

2 Variables preliminary exploration

This section is devoted to analyse the main properties found in the **breast cancer** data set.

2.1 The `diagnosis` variable

The `diagnosis` variable is the dependent variable for this model. The Figure 1 presents the histograms of the `diagnosis` variable. Proportions for both classes within `diagnosis` are summarized in Table 2, where clearly the `Benign` class is much more common than the `Malignant` one.

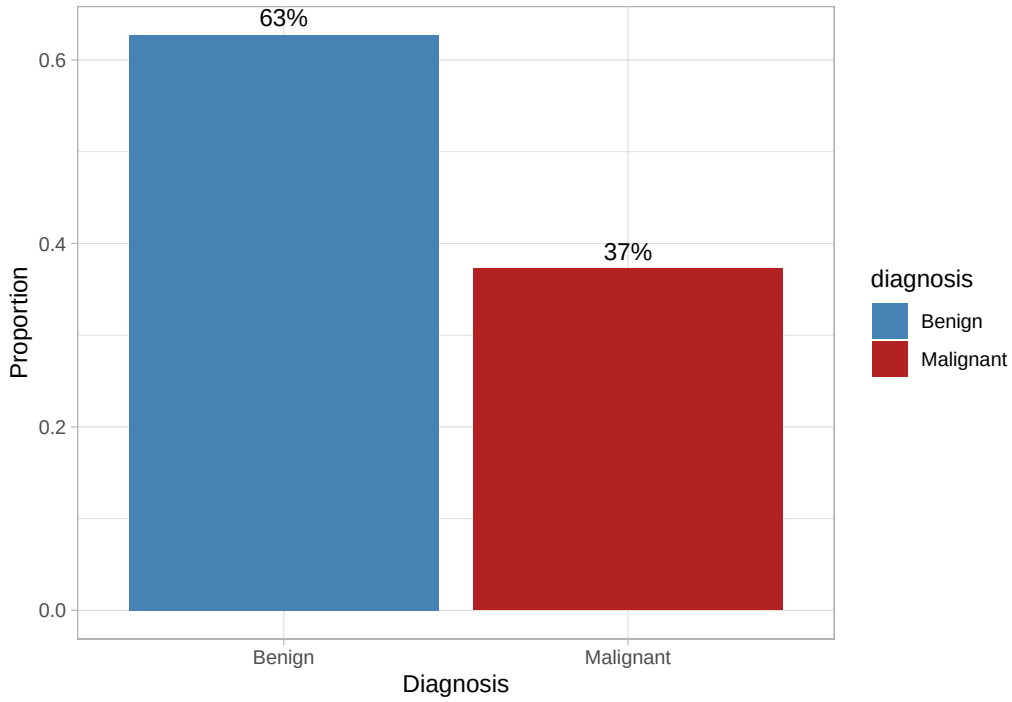

Figure 1: Diagnosis Distribution

Table 2: Summary of diagnosis proportions in data set

diagnosis	n	proportion
Benign	357	0.6274165
Malignant	212	0.3725835

2.2 Data significance

Data significance typically refers to variables that show statistically meaningful differences between classes, in this particular case, Benign/Malignant. A feature is considered significant if:

1. It shows statistically reliable differences between benign/malignant cases (low p-value)
2. The effect size is large enough to be biologically/medically meaningful
3. It improves model performance when included

Defining this characteristic is fundamental for the modelling task, if it is considered that the model must predict, from features, if a laboratory sample image conduces to a benign or a malignant diagnostic.

2.2.1 Statistical differences

The features used in the model provide useful information to the classification process which requires from them statistical differences through the classes. This is, if for a particular feature its probability distribution differs between benign and malignant classes (the difference can not be attributed to chance), then that feature must be considered in the model.

As a matter of fact, a typical test for studying the differences between probabilistic distributions is the *Welch's t-test*. However, even powerful, it is also known that its efficacy is limited to normally distributed data. This poses an additional analysis on the data set, if it is normally distributed or not.

2.2.2 Normality

There are many standard tests for testing normality in data sets. For example, are common the Shapiro-Wilk normality Test(Royston 1982), the Anderson-Darling (Stephens 1986) test for normality, among others (Thode 2002). Their usage depends on aspects like how large the sample is, or sensitivity to tail deviations, skewness, etc. Here the Anderson-Darling is used, according to the sample size and robustness of the test.

Table 3: Arlington-Darling normality test

feature	statistic	p.value	normal
avg_radius	1.0823e+01	3.700e-24	Non-Normal
avg_texture	2.5528e+00	1.892e-06	Non-Normal
avg_perimeter	1.1502e+01	3.700e-24	Non-Normal
avg_area	2.3510e+01	3.700e-24	Non-Normal
avg_smoothness	9.8225e-01	1.352e-02	Non-Normal
avg_compactness	1.1500e+01	3.700e-24	Non-Normal
avg_concavity	2.1662e+01	3.700e-24	Non-Normal
avg_concave_points	1.9257e+01	3.700e-24	Non-Normal
avg_symmetry	2.9812e+00	1.708e-07	Non-Normal
avg_fractal_dim	8.5840e+00	7.036e-21	Non-Normal
se_radius	3.3307e+01	3.700e-24	Non-Normal
se_texture	1.0453e+01	3.700e-24	Non-Normal
se_perimeter	3.5948e+01	3.700e-24	Non-Normal
se_area	6.0462e+01	3.700e-24	Non-Normal
se_smoothness	1.7658e+01	3.700e-24	Non-Normal
se_compactness	2.2780e+01	3.700e-24	Non-Normal
se_concavity	2.8799e+01	3.700e-24	Non-Normal
se_concave_points	7.6514e+00	1.097e-18	Non-Normal
se_symmetry	2.1016e+01	3.700e-24	Non-Normal
se_fractal_dim	3.3239e+01	3.700e-24	Non-Normal
w_radius	1.6395e+01	3.700e-24	Non-Normal
w_texture	1.6880e+00	2.478e-04	Non-Normal
w_perimeter	1.5995e+01	3.700e-24	Non-Normal
w_area	3.2115e+01	3.700e-24	Non-Normal
w_smoothness	8.7444e-01	2.495e-02	Non-Normal
w_compactness	1.4326e+01	3.700e-24	Non-Normal
w_concavity	1.1028e+01	3.700e-24	Non-Normal
w_concave_points	6.6668e+00	2.346e-16	Non-Normal
w_symmetry	9.1144e+00	4.042e-22	Non-Normal
w_fractal_dim	1.3489e+01	3.700e-24	Non-Normal

Table 3 summarizes the results of the Anderson-Darling test applied to the 30 features in the **breast cancer** data set. The immediate conclusion is that the p-value for all features is uniformly smaller than 0.05. This indicates that the null hypothesis is rejected across the entire data set, meaning the data are not normally distributed.

The results in Table 3 were obtained by testing the normality of each feature across the entire data set. However, these features can also be grouped by `diagnosis` (benign or malignant) and then re-evaluated for normality within those specific groups. Figure 2 illustrates this case, clearly showing that some feature sets may indeed be normally distributed when grouped by their diagnosis class.

These normality tests provide additional insightful for future steps of the modelling process. It is clear now that **model structures assuming some statistical properties, such as normality, on the data**

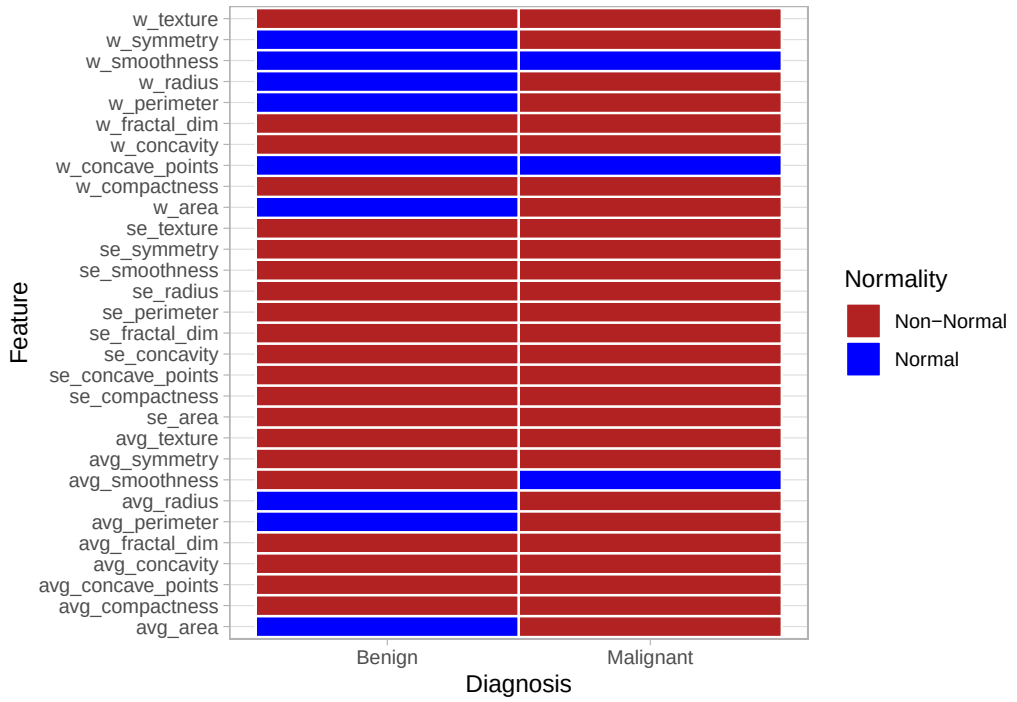

Figure 2: Normality Status by Diagnosis Group

should be avoided. The other aspect for taking into account is the so called *effect size*, and which statistical test is better for this data set.

2.2.3 Statistical tests (p-value)

For non-parametric data, the Mann-Whitney U-test or Wilcoxon Rank-Sum Test is a non-parametric alternative to the independent two samples t-test for comparing two independent groups of samples, it is a suitable tool to determine if two groups have *different distributions*. In this case the null hypothesis (H_0) indicates if the distributions of the two groups are identical versus the alternative one where the distributions are shifted (different medians). It is important to stress the fact that the p-value determines the significance of the difference (reject H_0: distributions differ), but does not quantify the magnitude of the difference. Here this test will be applied to the **breast cancer** data set through the `wilcox_test()` function of `rstatix` package.

2.2.4 Effect

The *effect size* (Tomczak and Tomczak 2014) quantifies the magnitude of the difference or relationship between a specific feature and the dependent variable, which in this case is `diagnosis` (benign or malignant). This metric helps determine if a particular feature, such as `avg_radius` or `w_concave_points`, exhibits a difference large enough to be biologically or medically meaningful. In statistical learning, effect size can serve as a feature selection criterion: features with *small effect sizes may be less useful for building predictive models* and can often be removed to reduce dimensionality and enhance model performance. Unlike a p-value, which tells you if a difference or relationship is likely due to chance (statistical significance), effect size tells how large or practically important that difference or relationship is.

A standard test for measuring effect size is Cohen's d (Cohen 1988). However, this test is limited by its requirement for normally distributed data. As noted in Section 2.2.2, the **breast cancer** data set is not normally distributed, necessitating the use of a non-parametric alternative. For non-normal data the alternative is again the Wilcoxon tests, in this case, the *Rank-Biserial Correlation* defines a measure of effect

size for the Wilcoxon rank-sum test, quantifying the strength and direction of the difference between two groups. Here this test is applied through the `wilcox_effsize()` function.

Both of these tests are complementary, so it is good statistical practice to report both statistical significance (p-value) and effect size. The Table 4 summarizes the statistical Significance and size effect for the **breast cancer** data set, as well as the Figure 3. Here the effects are classified as: 0.10 - < 0.3 (small effect), 0.30 - < 0.5 (moderate effect) and >= 0.5 (large effect).

Table 4: Statistical Significance and size effect for the breast cancer dataset

feature	group 1	group 2	p	p.adj	effsize	magnitude
w_perimeter	Benign	Malignant	2.5800e-80	7.7400e-79	0.7956185	large
w_radius	Benign	Malignant	1.1400e-78	1.7100e-77	0.7872403	large
w_area	Benign	Malignant	1.8000e-78	1.8000e-77	0.7862101	large
w_concave_points	Benign	Malignant	1.8600e-77	1.3950e-76	0.7809864	large
avg_concave_points	Benign	Malignant	1.0100e-76	6.0600e-76	0.7771936	large
avg_perimeter	Benign	Malignant	3.5500e-71	1.7750e-70	0.7478380	large
avg_area	Benign	Malignant	1.5400e-68	6.6000e-68	0.7334771	large
avg_concavity	Benign	Malignant	2.1600e-68	8.1000e-68	0.7326632	large
avg_radius	Benign	Malignant	2.6900e-68	8.9667e-68	0.7321408	large
se_area	Benign	Malignant	5.7700e-65	1.7310e-64	0.7135559	large
w_concavity	Benign	Malignant	1.7600e-63	4.8000e-63	0.7051136	large
se_perimeter	Benign	Malignant	5.1000e-51	1.2750e-50	0.6298571	large
se_radius	Benign	Malignant	6.2200e-49	1.4354e-48	0.6163697	large
avg_compactness	Benign	Malignant	8.9500e-48	1.9179e-47	0.6087527	large
w_compactness	Benign	Malignant	2.1200e-47	4.2400e-47	0.6062765	large
se_concave_points	Benign	Malignant	2.3700e-31	4.4437e-31	0.4882876	moderate
w_texture	Benign	Malignant	6.5200e-30	1.1506e-29	0.4763010	moderate
se_concavity	Benign	Malignant	3.6800e-29	6.1333e-29	0.4699247	moderate
avg_texture	Benign	Malignant	3.4300e-28	5.4158e-28	0.4615648	moderate
w_smoothness	Benign	Malignant	3.6400e-24	5.4600e-24	0.4251390	moderate
w_symmetry	Benign	Malignant	3.1500e-21	4.5000e-21	0.3964938	moderate
se_compactness	Benign	Malignant	1.1700e-19	1.5955e-19	0.3803309	moderate
avg_smoothness	Benign	Malignant	7.7900e-19	1.0161e-18	0.3715649	moderate
avg_symmetry	Benign	Malignant	2.2700e-15	2.8375e-15	0.3322751	moderate
w_fractal_dim	Benign	Malignant	1.1400e-13	1.3680e-13	0.3112029	moderate
se_fractal_dim	Benign	Malignant	1.5700e-06	1.8115e-06	0.2013145	small
se_symmetry	Benign	Malignant	2.7800e-02	3.0889e-02	0.0922223	small
se_smoothness	Benign	Malignant	2.1400e-01	2.2929e-01	0.0521472	small
avg_fractal_dim	Benign	Malignant	5.3700e-01	5.5552e-01	0.0258802	small
se_texture	Benign	Malignant	6.4400e-01	6.4400e-01	0.0194018	small

Table 4 highlights a key reason why both statistical tests were included. The rows marked in blue show an interesting scenario: an effect size below 0.3 (considered "small") alongside a p-value greater than 0.05. This combination might seem contradictory, but it often arises when large sample sizes make it possible to detect even very small differences. In such cases, it's generally more informative to prioritize the **effect size** as it indicates the practical significance of the finding. The issue can also been observed in Figure 3 where one grey point is out the dashed boundaries.

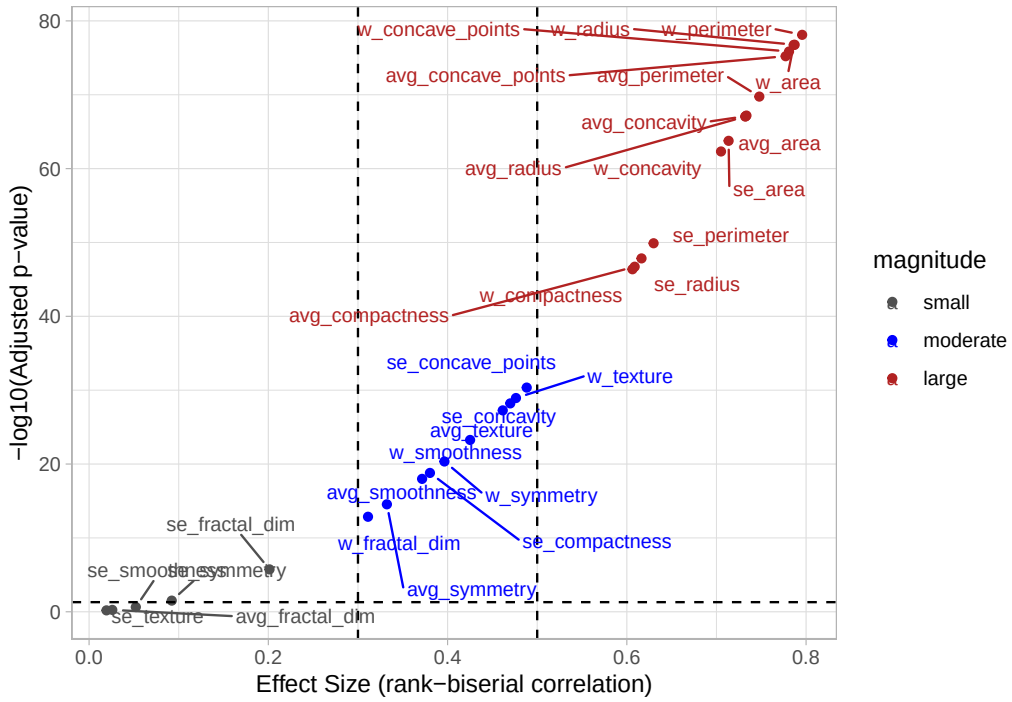

Figure 3: Feature Significance Analysis

The main conclusion of this analysis is that, if dimensionality reduction is going to be considered, features with small effect are the candidates.

2.3 Data correlation

Another interesting characteristic in **breast cancer** data set is the fact that very similar features are reported, for example the radio, perimeter and area of nucleus (in average), which presumably are highly correlated. The Figure 4 shows the correlation between features of the data set. As expected there are highly correlated features related with areas, perimeters and radios, for the case of *mean* type variables as well as *worst* type variables. This result also points to an analysis of dimension reduction.

2.3.1 Principal components and dimensionality reduction

Principal Component Analysis (PCA) is a technique used to reduce the number of features (dimensions) in a dataset while keeping as much important information as possible. It works by finding new axes (called principal components) that capture the most variation in the data. The first principal component explains the most variance, the second explains the next most, and so on.

The Figure 5 shows the percentage of variance explained versus the number of principal components used. As can be seen for reaching a 95% of variable explained, a total of 10 principal components must be considered.

The Figure 6 shows the loadings of the original variables (features) with respect to the 6 first principal components (PC) (see Figure 5). Clearly the lowest important features in the first PC coincide with the lowest size effect features in Table 4 as well as Figure 3.

The scatter plot of Figure 7 shows how the data points are spread along the first two principal components from PCA. As can be seen there are clearly two clusters in the plot that shows natural separation ("Malignant" vs. "Benign").

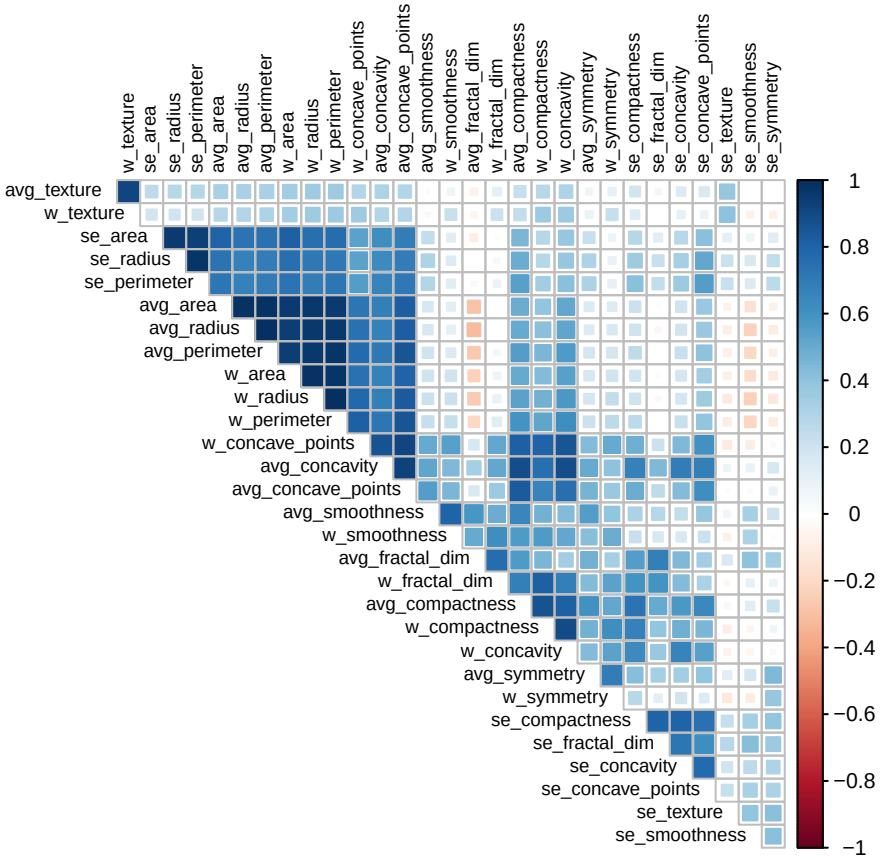

Figure 4: Correlation between features

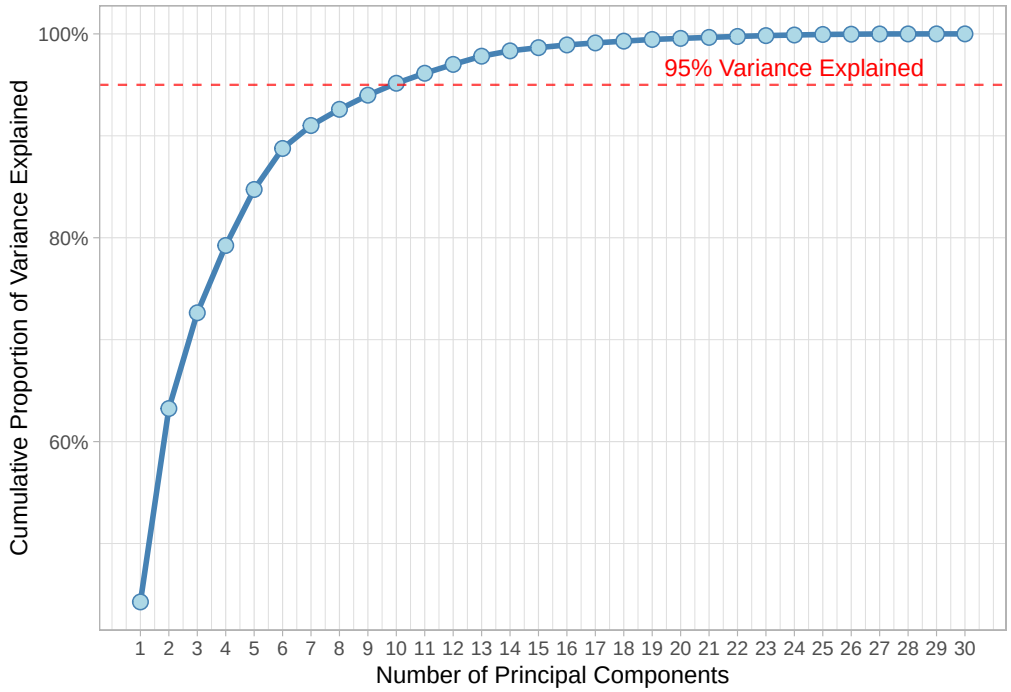

Figure 5: Cumulative Variance Explained by Principal Components

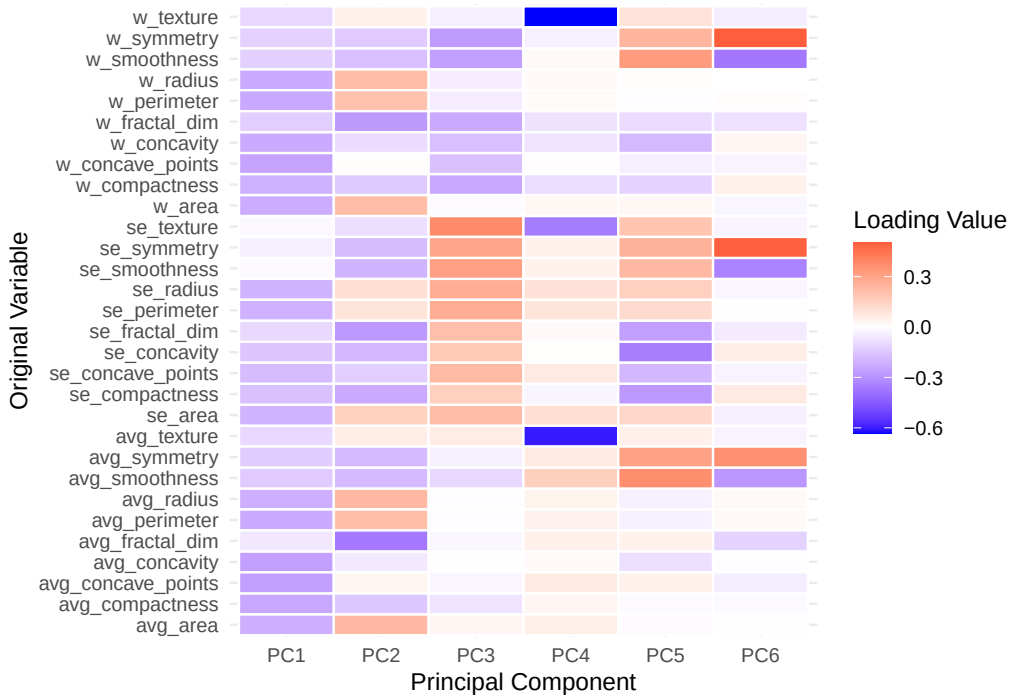

Figure 6: Loadings of Original Variables on PC1-PC6

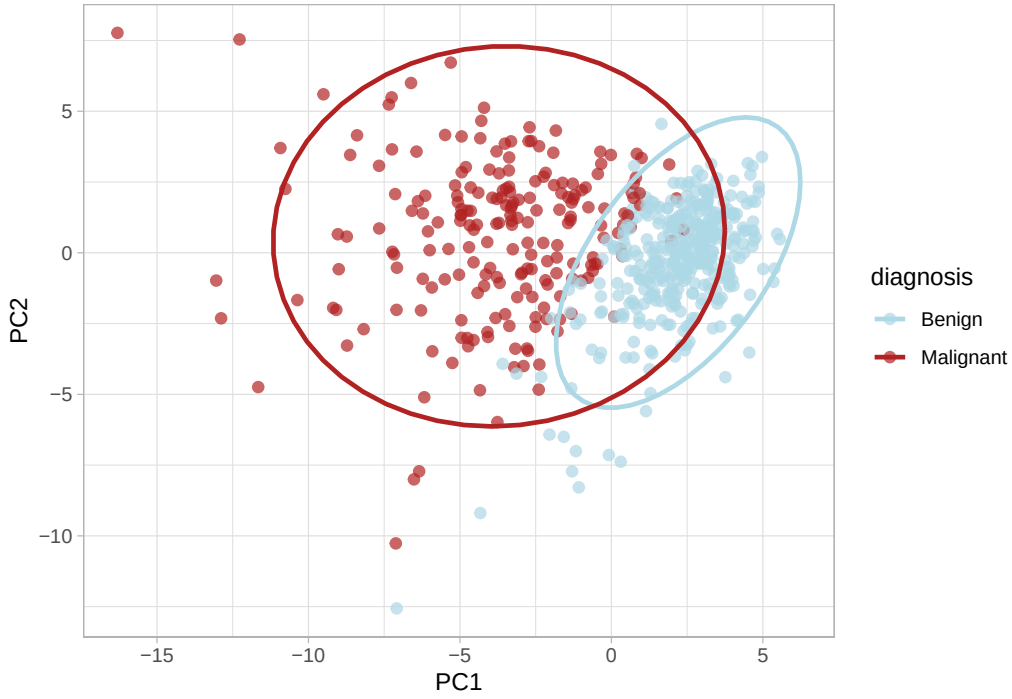

Figure 7: PC1 vs PC2

3 Methods

This section presents some insights related to the methods and reasoning for the modelling and testing process. Some basic descriptions of the model classes are presented. It is out of the scope of this report to go further in this topics, as well as many of this information is now standard on the literature.

Additionally the main code snippets and functions are presented as well as the values used for controlling the training and the tune-grids used during the optimization of each model-dependant parameter.

3.1 Preparing data for training and testing

As for this task the only available data set is **breast cancer**, it is necessary to split it into a training `train_set` and a validating `test_set` data sets. To this aim, a typical procedure based on `createDataPartition()` function was used by using a `p = 0.2` parameter (20% probability). In this case `train_set` contains 80% of the extended **breast cancer** data set whereas the `test_set` contains the remaining 20%.

3.1.1 Pre-processing data sets

Once the overall data set has been partitioned into train/test sets, it is a common task in pre-processing to centre and scale the data sets. Centering is related to subtract the mean value to the data:

$$f_{ic}^{k} = f_i - \text{mean}(f_i). \tag{1}$$

Scaling is related to scale by the standard deviation of the data:

$$f_{is}^{k} = f_{ic}/\text{sd}(f_i). \tag{2}$$

where $f_i,\ f_{ic}$ and f_{is} are $i-th$ feature and centered and scaled versions respectively and $k \in [\text{train}, \text{test}]$. Some remarks are important here:

- The center/scale process is made after partitioning the overall data set, i.e., on the `train_set` and `test_set`. The training data is supposed to be completely independent from the test data, using the average or standard deviation from the entire data set to center/scale will take information for the test into the training data, a phenomenon known as "data leakage".
- The center/scale process on the `test_set` is made by using the mean and deviation from the `train_set`.

In `R` this task is easily accomplished through the `preProcess(x, method = c("center", "scale"))` function from `caret` package.

3.2 Metrics

3.2.1 Confusion matrix

As in any modelling process, defining the proper metrics for the model performance is a crucial step. In this case, this question is about which metric is most critical for evaluating a *breast cancer diagnostic model*. This particular model corresponds to a *binary classification problem* in which predictions can be categorized into four outcomes, the basic components of a **confusion matrix**:

	Actual Positive (Cancer)	**Actual Negative (No Cancer)**
Predicted Positive	True Positive (TP)	False Positive (FP)
Predicted Negative	False Negative (FN)	True Negative (TN)

defined as:

- **True Positive (TP):** The model correctly predicted a positive case (e.g., predicted Malignant, and it was actually Malignant).

- **True Negative (TN):** The model correctly predicted a negative case (e.g., predicted Benign, and it was actually Benign).
- **False Positive (FP):** The model incorrectly predicted a positive case (e.g., predicted Malignant, but it was actually Benign). This is also known as a Type I error.
- **False Negative (FN):** The model incorrectly predicted a negative case (e.g., predicted Benign, but it was actually Malignant). This is also known as a Type II error.

From here on, it is assumed that "Malignant" is the *positive* class (the condition to be detected) and "Benign" is the *negative* class. From these components it is possible to define these metrics:

- **Accuracy:** This is the most straightforward metric. It represents the *overall proportion of correctly classified instances* (both positive and negative) out of all instances. It is given by

$$Accur = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

- **Sensitivity:** This measures the model ability to correctly identify all actual positive cases. In the context of breast cancer, it is the *proportion of actual Malignant cases that were correctly identified as Malignant*. It is also called **True Positive Rate**.

$$Sens = \frac{TP}{TP + FN} \tag{4}$$

- **Specificity:** This measures the model ability to correctly identify all actual negative cases. In the context of breast cancer, it is the *proportion of actual Benign cases that were correctly identified as Benign*. It is also called **True Negative Rate**.

$$Spec = \frac{TN}{TN + FP} \tag{5}$$

Even *accuracy* seems to be the most intuitive metric, it can be misleading, particularly in data sets characterized by class imbalance, which is common in medical contexts (e.g., breast cancer data sets where healthy cases often significantly outnumber cancer cases). This fact was reflected here in Table 2 where, for this data set, the number of Malignant cases were barely half the number of benign ones.

On the other hand, if a real scenario is assumed, this modelling process should not be only aforded as pure a numerical exercise, but actual medical considerations must be taken into account. For example, (Ong and Mandl 2015) shows that, by 2015 the cost of breast cancer overdiagnosis has been estimated to be $4 Billion a year. Human harder consequences can be observed is a more conservative approach is taken, i.e., underdiagnosis or a bias or an disproportionate number of false negatives may lead to a high cost in human lives.

3.2.2 The ROC Curve and AUC

The called "ROC" metric refers to the *Receiver Operating Characteristic* (ROC) curve, and more commonly, its associated summary statistic, the *Area Under the Curve* (AUC). It's a fundamental tool for evaluating the performance of binary classification models.

An ROC curve is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied.

- **x-axis:** False Positive Rate (FPR), also known as (1 - Specificity).
- **y-axis:** True Positive Rate (TPR), also known as Sensitivity or Recall.

Axes:

As classification threshold (from 0 to 1) is modified, a different pair of (FPR, TPR) values is obtained. Plotting all these pairs traces out the ROC curve.

- A threshold of 1.0 (very strict) would likely classify everything as negative, giving a low TPR and low FPR (closer to (0,0)).
- A threshold of 0.0 (very lenient) would likely classify everything as positive, giving a high TPR and high FPR (closer to (1,1)).

The AUC is a single scalar value that summarizes the overall performance of the classifier across all possible classification thresholds. It is simply the area underneath the ROC curve. The AUC represents the probability that a randomly chosen positive example will be ranked higher (assigned a higher probability of being positive) than a randomly chosen negative example. Values range from 0 to 1:

- 0.5: The model has no discriminatory power; it's as good as random guessing (represented by a diagonal line from (0,0) to (1,1)).
- 0.7 - 0.8: Generally considered acceptable discrimination.
- 0.8 - 0.9: Considered excellent discrimination.
- 0.9 - 1.0: Considered outstanding or exceptional discrimination.
- 1.0: Perfect discrimination (the curve goes straight up from (0,0) to (0,1) and then across to (1,1)).

In `R` it is straightforward to calculate AUC and data for ROC curve through the `pROC` package, just using functions `roc()` and `auc()` . Additionally there exist variations based on `ggplot` for plotting ROC curves by using `ggroc()`.

3.2.3 The F1 metrics

The F1-score is the *harmonic mean of Precision and Recall* (Sensitivity). It is defined as

$$F1 = 2\frac{\text{precision} \times \text{sensitivity}}{\text{precision} + \text{sensitivity}}, \tag{6}$$

where the *precision* (how many predicted as positive are actually positive) is

$$\text{precision} = \frac{TP}{TP + FP}. \tag{7}$$

Clearly it is focused on minimizing false positives. The F1-score provides a single metric that gives equal weight to Precision and Recall. A high F1-score means the model has both high Precision (few false alarms) and high Recall (misses few actual positives).

In `R` the F1-score as well as sensitivity and specificity, can be obtained from the `caret::confusionMatrix()`, by accessing list field `$byClass`.
Here the ROC, specificity and sensitivity metrics are used for training and selecting the best model during CV. The F1-score is used during the training of the *ensemble model*.

4 Modelling process

For the modelling process, four different strategies have been selected, *k-Nearest neighbour (kNN)*, *Random forest*, *radial support vector machine (SVM)* and *xgbTree*. Small summary of these models are presented next. Further theoretical details are out of the scope of this report. Most of these models where discussed in the book by(Irizarry 2020) and through the course. A total of ten cross validation (CV), each one repeated five times for each model class will be run, these parameters passed to the `train()` function of `caret` package through the `trainControl()` function. The set of parameters is summarized in Table @ref{tab:train-control-setup-table}.

Table 6: Paramenters used during training and cross validation

Parameter	Value
method	repeatedcv
number	10
repeats	5
classProbs	TRUE
returnResamp	final
savePredictions	final
allowParallel	TRUE
summaryFunction	twoClassSummary
selectionFunction	best

4.1 k-Nearest neighbour (kNN)

K-Nearest Neighbors (KNN) is a supervised machine learning algorithm used for both classification and regression tasks (Duda, Hart, and Stork 2001). It is often described as a "lazy" or "instance-based" learner because it does not build a model during a training phase, but rather memorizes the entire training data set.

KNN operates on the principle that "similar things are in close proximity" When predicting the outcome for a new, unlabeled data point, KNN looks at the k closest data points (its "neighbours") in the training data set and uses their known labels or values to make a prediction.

When training a KNN model, the key parameter is the number of neighbours k, s.t.,

- **Small k:** Highly flexible, fits training data tightly. Risk of overfitting (noisy data causes instability).
- **Large k:** Smoother decision boundaries. Risk of underfitting (may ignore local patterns).

4.2 Random forest

Random Forest is a versatile and popular ensemble learning method for both classification and regression. It operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. It is an example of a "bagging" (bootstrap aggregating) ensemble.

4.3 Radial Support Vector Machines (SVM)

Radial Support Vector Machines (SVM) use the Radial Basis Function (RBF) kernel to handle non-linear classification by transforming data into a higher-dimensional space where it becomes separable.

The RBF measures similarity between points using

$$K(x_i, x_j) = \exp(-\sigma||x_i - x_j||^2), \tag{8}$$

where σ controls flexibility, s.t.,

- **High σ:** Tight, complex boundaries (risk of overfitting)
- **Low σ:** Smoother, broader boundaries (may underfit)

Additionally there is a *Regularization parameter* C which controls the trade-off between margin width and misclassifications, s.t.,

- **High C:** Strict margin, fewer misclassifications (risk of overfitting)
- **Low C:** Wider margin, allows some errors (better generalization)

4.4 xgbTree

XGBoost, short for "Extreme Gradient Boosting," is an optimized, distributed, and highly efficient implementation of Gradient Boosting Machines (GBM). It's an ensemble learning method primarily based on boosting, which sequentially builds models that correct the errors of previous models. It is exceptionally popular for both classification and regression tasks due to its speed and performance.

4.5 Ensemble weighted probabilities

AS a last step in the training process, an ensemble model is built. There is no specific rule for building ensemble models, but the main idea is to mix the properties of first level models into a new one, trying to get additional information into an improved model. Even a common approach is to stack a model from the out-off fold probabilities of best models calculated during the CV and then train a logistic regression model, here a simpler approach will be applied.

The idea behind this ensemble is to take the out-of-fold (oof) best models from the CV process, regardless the metric used during training. For each model class, a linear combination of its set of probabilities, filtered by the positive category (in this case the positive category is set to `Malign`), with a set of weights is calculated, this is,

$$p_e = \sum_{i=1}^{n_c} w_i p_i, \tag{9}$$

where p_e and pc_i correspond to the ensemble and each one of the n_c class predictors, respectively and w_i represents the selected weights. Optimal weights are not known beforehand, so a simple solution for selecting w_i is by defining a grid and calculate an specific metric iteratively (F1 or AUC). This process can be made easily with a function like this:

```
compute_f12 <- function(w,p,yData) {
  # Inputs:
  # w: a set of weights to be tested
  # p: the predictions training data set
  # yData: diagnosis variable
  # Outputs:
  # A tibble containing the F1 and AUC metrics for the set of weights tested
  # according to: # ensemble_p = w[1]*p1 + w[2]*p2 + w[n]*pn +...
  w <- as.numeric(w)

  ensemble_p <- as.numeric(p%*%as.matrix(w))
  pred_class <- ifelse(ensemble_p >= 0.5, "Malignant", "Benign")
  pred_class <- factor(pred_class, levels = levels(yData))
  cm <- confusionMatrix(pred_class, yData, positive = "Malignant")
  roc_obj <- roc(yData, ensemble_p, levels = rev(levels(yData)))
  tibble(F1 = cm$byClass["F1"], AUC = as.numeric(auc(roc_obj)))
}
```

For this case $w_1, w_2 \in [0.005, 1]$ by steps of 0.005, $w_3 = 1 - (w_1 + w_2)$, for a ensemble of three models. For a 4 models ensemble $w_1, w_2, w_3 \in [0.05, 1]$ by steps of 0.05, $w_4 = 1 - (w_1 + w_2 + w_3)$. It should be noted that this entire procedure is performed on the training dataset. Once the "optimal" weights, which maximize F1 or AUC, are selected, new predictions are generated using the test set, allowing the quality of the ensemble model to be assessed through the linear combination. A simple code example for this task is:

```
cl <- makeCluster(detectCores() -1) # Create a cluster
registerDoParallel(cl)          # Register clusted for foreach/caret

# Prepare data set
p <- as.matrix(meta_features_train%>% select(-rowIndex))
# iterate models from predefined weights grid (parallel processing)
```

```r
f1_AUC <- foreach(i = 1:nrow(grid), .combine = bind_rows,
                  .export = c("grid", "compute_f12", "p","yENS"),
                  .packages = c("caret","pROC","dplyr")
) %dopar% {
  # current set of weights
  current_weights <- grid[i, 1:3]
  # Obtain F1 and AUC for current weight
  compute_f12(current_weights, p, yENS)
}
stopCluster(cl)
aaa <- f1_AUC %>% summarise(b_F1 = which.max(F1),b_AUC = which.max(AUC))
grid$F1 <- f1_AUC$F1

# Best weights
best_row <- grid[aaa$b_F1, ]
best_weights <- as.numeric(best_row[1:3])

# VERIFY MODEL ON TEST DATA PREDICTIONS (already calculated with model_metrics_results())
svm_p <- model_svm[[4]][, "Malignant"]
xgb_p <- model_xgbTree[[4]][, "Malignant"]
knn_p <- model_knn[[4]][, "Malignant"]
rf_p  <- model_rf[[4]][, "Malignant"]

# Use best weights to compute final ensemble prob
ensemble_p_best <- best_weights[1]*svm_p + best_weights[2]*xgb_p + best_weights[3]*knn_p
pred_class <- ifelse(ensemble_p_best >= 0.5, "Malignant", "Benign")
pred_class <- factor(pred_class, levels = levels(y_T))
# Calculate confusionMatrix and information within
cm <- confusionMatrix(pred_class, y_T, positive = "Malignant")

# calculate ROC metric
roc_obj <- roc(y_T, ensemble_p_best, levels = rev(levels(y_T)))
```

4.6 Tuning grids

According to former descriptions, the set of tune-grids used for CV training of the selected classes of models are summarized in 7.

Table 7: Model Tuning Grids Summary

Model	Parameters (Class)	Range
KNN	k	1 to 60 by 2
SVM	sigma, C	sigma: 0.01, 0.1, 0.25; C: 1 to 10 by 1
RF	mtry	1 to 30 by 2
XGBoost	nrounds, max_depth , eta , gamma, colsample_bytree , min_child_weight, subsample	nrounds: 100, 200; max_depth: 2 to 10 by 2; eta: 0.1 to 1 by 0.25; gamma: 0; colsample_bytree: 0.8; min_child_weight: 1; subsample: 0.8

4.7 PCA data-based models

As discussed in former sections, it is possible to improve the modelling process by including the concept of *principal components*. A full set of models will be built by considering this strategy, in such a way the

redundant or perhaps less informative features are "filtered" from the data used during CV and training.

The implementation of PCA data-based modelling is straightforward in `caret` package, it requires considering again the `preProcess()` function, just adding the key `pca` as a parameter, this is, `preProcess(x, method = c("center", "scale", "pca"), thresh = 0.95 )`. By following this procedure, the `caret::train()` function will use the PCA dataset that describes a 95% of the data variance (see Figure 5).

4.8 Multiple metrics approach

One important issue identified during preliminary training (not detailed here for simplicity) is the significant impact of the model selection metric on overall model performance during CV. To incorporate this understanding into the modelling process, each model class is evaluated using three distinct metrics: *Sensitivity*, *Specificity*, and *AUC*. This evaluation considers the tune-grids and training control settings already defined (Table 6 and Table 7). This task is straightforwardly accomplished using the following function, which trains and evaluates all models, applying multiple metrics for optimal model selection during the tuning process:

```
model_metrics_results <- function(m_method, tune_grid, params_tuned){
  # Inputs:
  # m_method: the trained method to be used ("knn", "rf", "sum", "xgb")
  # tune_grid: grid for optimizing each class method (k, mtry, eta, sigma,etc)
  # params_tuned: names of the optimized parameters (only for summarizing purposes)
  # Outputs:
  # 1: tibble summarizing metrics, 2: tuned models (ROC, Sens, Spec),
  # 3: tibble ROC data, 4: predicted probabilities on test data (pred_probs)

  # List to store trained model objects
  trained_models_list <- list()

  # List to store ROC curves for plotting
  roc_curves_list <- list()

  # Create an empty tibble to store results
  metrics_model_results <- tibble(
    method = character(), # trained method
    metric = character(), # the metric used for selecting the "best" in CV
    Accuracy = numeric(),
    Sensitivity = numeric(),
    Specificity = numeric(),
    AUC = numeric(),
    F1 = numeric(),
    best_p_cv = list(), # store the optimal cross validation value(s)
    parameter = character() #
  )

  # Metrics to optimize for during cross-validation
  # "ROC" is AUC, "Sens" is Sensitivity, "Spec" is Specificity
  optimization_metrics <- c("ROC", "Sens", "Spec")
  # iterate a for loop for training/testing a model by modifying the metrics
  # for the cross validation/folding
  for (i in 1:length(optimization_metrics)){

    opt_metric <- optimization_metrics[i]
    message(paste0("Training ", m_method, " model optimizing for: ", opt_metric))

    # Train the  model
```

```r
  # Set an specific seed, for repeatability results, can be neglected later
  set.seed(7, sample.kind = "Rounding")
  model_tuned <- train(
    xS, y,
    method = m_method,
    tuneGrid = tune_grid,
    trControl = fitControl,
    metric = opt_metric )

  # Store the trained model
  trained_models_list[[opt_metric]] <- model_tuned

  # Make predictions on the *test* set, will be used in ensemble modelling
  pred_classes <- predict(model_tuned, newdata = xS_T, type = "raw")
  pred_probs <- predict(model_tuned, newdata = xS_T, type = "prob")

  # Calculate Confusion Matrix and metrics on the TEST SET
  results_cm <- confusionMatrix(
    pred_classes,
    y_T,
    positive = positive_class_label)

  # Calculate AUC for the test set
  roc_obj <- roc(
    response = y_T,
    predictor = pred_probs[[positive_class_label]],#probabilities for the positive class
    levels = levels(y_T))

  # Store the best model AUC
  auc_value <- as.numeric(auc(roc_obj))
  roc_curves_list[[opt_metric]] <- roc_obj
  # Store the best model parameter (k_best, sigma_best, etc)
  best_param_value <- model_tuned$bestTune[params_tuned]

  # Store the results
  current_results <- tibble(
    method = m_method,
    metric = opt_metric,
    Accuracy = results_cm$overall["Accuracy"],
    Sensitivity = results_cm$byClass["Sensitivity"],
    Specificity = results_cm$byClass["Specificity"],
    AUC = auc_value,
    F1 = results_cm$byClass["F1"],
    best_p_cv = list(best_param_value),
    parameter = paste(params_tuned, collapse = ", "))

  metrics_model_results <- bind_rows(metrics_model_results, current_results)
}
# Return a 4 elements list
metrics_model_results <- list(metrics_model_results,
                              trained_models_list,
                              roc_curves_list,
                              pred_probs)
```

```
} # end of function model_metrics_results()
```

5 Results

According to the discussion on metrics from section 3.2 a first round of models and tests has been tested, here considering `knn`, `svmRadial`, `rf` and `xgbTree`. Each model has been trained by using the set of metrics selected (sensitivity, specificity and AUC)

Table 8: Summary of performance of the models classes

method	metric	Accuracy	Sensitivity	Specificity	AUC	F1	best_p_cv	parameter
svmRadial	Sens	0.9826087	0.9767442	0.9861111	0.9996770	0.9767442	0.01, 7.00	sigma, C
svmRadial	ROC	0.9739130	0.9534884	0.9861111	0.9990310	0.9647059	0.01, 4.00	sigma, C
xgbTree	Sens	0.9739130	0.9534884	0.9861111	0.9983850	0.9647059	100.00, 8.00, 0.35	nrounds, max_depth, eta
xgbTree	Spec	0.9739130	0.9534884	0.9861111	0.9983850	0.9647059	100.00, 6.00, 0.35	nrounds, max_depth, eta
xgbTree	ROC	0.9739130	0.9534884	0.9861111	0.9980620	0.9647059	100.00, 2.00, 0.35	nrounds, max_depth, eta
knn	Spec	0.9739130	0.9534884	0.9861111	0.9972545	0.9647059	5	k
rf	Spec	0.9565217	0.9534884	0.9583333	0.9951550	0.9425287	13	mtry
rf	ROC	0.9478261	0.9302326	0.9583333	0.9945090	0.9302326	1	mtry
rf	Sens	0.9478261	0.9302326	0.9583333	0.9945090	0.9302326	1	mtry
svmRadial	Spec	0.9391304	0.9534884	0.9305556	0.9919251	0.9213483	0.1, 8.0	sigma, C
knn	Sens	0.9478261	0.8837209	0.9861111	0.9917636	0.9268293	55	k
knn	ROC	0.9391304	0.8604651	0.9861111	0.9906331	0.9135802	59	k

The Table 8 summarizes the performance of the models classes discussed here for the **breast cancer** data set. As can be observed all models show high Accuracy (≥ 0.94), with most > 0.95. For the case of Sensitivity (true positive rate) and specificity (true negative rate) are both excellent across models, critical for medical diagnosis. AUC values are very high (> 0.99 in top models) models have strong discriminative power.

Some comparisons across models can be made:

- **SVM (Radial Kernel):** Achieved the highest Sensitivity (97.67%) when optimized for Sensitivity, indicating strong performance in correctly identifying positive cases (e.g., malignant tumors). The highest AUC (0.9997) suggests excellent separability between classes. However, performance varied with hyperparameters: lower specificity (93.06%) when optimized for specificity, indicating a trade-off between sensitivity and specificity.
- **XGBoost (xgbTree):** Consistently high performance across metrics, with AUC ≈ 0.998 and Specificity up to 98.61%. The model showed robustness to hyperparameter changes (e.g., max_depth from 2 to 8 had minimal impact on performance).
- **Random Forest (rf):** Lower AUC (0.994 – 0.995) compared to SVM and XGBoost, but maintained decent sensitivity (93.02 – 95.35%). The best performance was achieved with mtry=13 (number of features sampled per split), but simpler models (mtry=1) also performed well, suggesting the data may not require complex feature interactions.
- **k-Nearest Neighbors (knn):** Performance varied significantly with k. High specificity (98.61%) with k=5, but sensitivity dropped to 88.37% with k=55. Larger k (e.g., 59) improved AUC but reduced sensitivity, highlighting a sensitivity-specificity trade-off.

These results can also be analysed straightforwardly from the heatmap of Figure 8. Something to be stressed here is the fact that the most weak metric is the *sensitivity*, providing the worst results, unlike the AUC and specificity that produce the best ones.

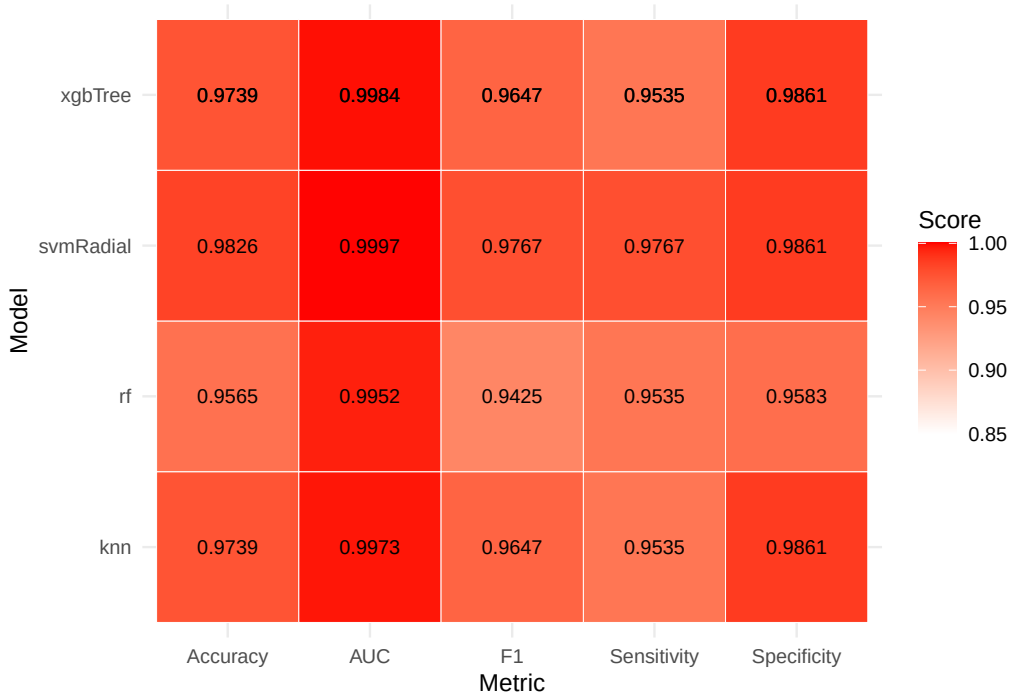

Figure 8: Best Model per Class Performance Heatmap

5.1 PCA data-based results

The results of the trained models for the case where the data set was pre-processed by using PCA are summarized in Table 9. These results allow evaluating whether dimensionality reduction improves performance.

Table 9: Summary of models with PCA preprocessed data

method	metric	Accuracy	Sensitivity	Specificity	AUC	F1	best_p_cv	parameter
svmRadial_PCA	Sens	0.9826087	0.9534884	1.0000000	0.9993540	0.9761905	0.01, 10.00	sigma, C
svmRadial_PCA	ROC	0.9826087	0.9534884	1.0000000	0.9987080	0.9761905	0.01, 9.00	sigma, C
knn_PCA	Spec	0.9739130	0.9302326	1.0000000	0.9980620	0.9638554	3	k
xgbTree_PCA	Sens	0.9739130	0.9767442	0.9722222	0.9970930	0.9655172	200.0, 10.0, 0.1	nrounds, max_depth, eta
knn_PCA	ROC	0.9826087	0.9534884	1.0000000	0.9969315	0.9761905	13	k
svmRadial_PCA	Spec	0.9826087	0.9534884	1.0000000	0.9967700	0.9761905	0.01, 2.00	sigma, C
xgbTree_PCA	ROC	0.9478261	0.9534884	0.9444444	0.9964470	0.9318182	100.00, 6.00, 0.35	nrounds, max_depth, eta
xgbTree_PCA	Spec	0.9565217	0.9534884	0.9583333	0.9964470	0.9425287	200.00, 6.00, 0.35	nrounds, max_depth, eta
knn_PCA	Sens	0.9565217	0.9069767	0.9861111	0.9924096	0.9397590	45	k
rf_PCA	Spec	0.9391304	0.9302326	0.9444444	0.9911176	0.9195402	7	mtry
rf_PCA	ROC	0.9652174	0.9534884	0.9722222	0.9907946	0.9534884	1	mtry
rf_PCA	Sens	0.9391304	0.9302326	0.9444444	0.9883721	0.9195402	25	mtry

The first thing to be stressed is the fact that there is an overall improvement nor worsening. The addition of PCA-preprocessed models shows generally comparable or improved performance across several metrics, especially in terms of Specificity and AUC, while maintaining high Sensitivity.

Some important remarks

- **SVM:** he PCA variant achieves Accuracy 0.9826, Sensitivity 0.9535, Specificity 1.000, AUC $\approx$ 0.999, F1 $\approx$ 0.976. Compared to non-PCA SVM Radial, PCA enhances Specificity (from 0.9861 to 1.000) while preserving overall Accuracy and AUC. The PCA SVM offers slightly better balance when high Specificity is critical.
- **KNN:** The best PCA KNN (ROC) model hits Accuracy 0.9826, Sensitivity 0.9535, Specificity 1.000, AUC 0.997, F1 $\approx$ 0.976. This is a clear improvement over the non-PCA KNN ROC model (Accuracy 0.9391, AUC 0.991).
- bf XGBoost: XGBoost PCA (Sens) model yields Accuracy 0.9739, Sensitivity 0.9767, AUC 0.997 — matching or slightly surpassing non-PCA XGBoost in Sensitivity and AUC. However, for ROC optimization, non-PCA XGBoost still edges out in AUC (0.998 vs 0.996).
- **Random forest:** PCA RF (ROC) achieves Accuracy 0.9652, AUC 0.991, slightly better than non-PCA RF ROC (AUC 0.995, Accuracy 0.9478). PCA seems to aid RF in Accuracy, though non-PCA RF has slightly higher AUC.

PCA preprocessing benefits KNN and SVM Radial most clearly, especially in boosting Specificity without sacrificing other metrics. For tree-based models (XGBoost, RF), PCA brings more modest or mixed impact. The top models overall are PCA SVM Radial and PCA KNN (ROC-optimized), both combining excellent Specificity (1.0) with high AUC and F1.

Figure 9: Best Model per Class Performance Heatmap (PCA)

5.2 Ensemble models

The Table 10 summarizes the results of all the ensemble models that have been built. The probabilistic weighted ensembles wp4 and wp3, both optimized for F1, achieved the highest Accuracy (0.9913043) and

perfect Specificity (1.0000000), alongside high Sensitivity (0.9767442) and AUC (0.9990310 and 0.9987080, respectively). Their F1 scores were also among the highest (0.9759036). This indicates that combining predictions from multiple models can lead to superior performance, particularly in balancing precision and recall as reflected by the F1 score.

Table 10: Summary of probabilistic weighted ensemble models with and without PCA preprocessed data

method	metric	Accuracy	Sensitivity	Specificity	AUC	F1	best_p_cv	parameter
ensemble wp3 P_OOF	F1	0.9913043	0.9767442	1.0000000	0.998708	0.9759036	0.18, 0.53, 0.29	w1,w2,w3
ensemble wp4 P_OOF	F1	0.9913043	0.9767442	1.0000000	0.999031	0.9759036	0.20, 0.55, 0.10, 0.15	w1,w2,w3,w4
ensemble wp3 P_OOF PCA	F1	0.9826087	0.9534884	1.0000000	0.998385	0.9640719	0.16, 0.63, 0.21	w1,w2,w3
ensemble wp4 P_OOF PCA	F1	0.9739130	0.9534884	0.9861111	0.996770	0.9640719	0.10, 0.75, 0.05, 0.10	w1,w2,w3,w4

The Figure 10 shows the ROC curves for the best models, in the sense of AUC, form Tables 8 and **??**tab:ensemble-data-table) (no PCA). All models exhibit high AUC values, indicating strong discriminative power, well above the 0.8 threshold for predictive value. The curves for SVM, ENS W4, ENS W3, and xgbTree are positioned closest to the top-left corner, signifying superior performance in balancing high sensitivity with low false positive rates across thresholds. The SVM (0.9997) and ENS W4 (0.999) show the highest AUCs, suggesting near-perfect discrimination. Notably, the SVM AUC of 0.9997 from this plot aligns closely with the svmRadial model optimized for Sensitivity (AUC 0.9996770) in Table 8, rather than its ROC-optimized entry (AUC 0.9739130). This suggests the plot represents the model best overall discriminative capability, which may be achieved when optimized for a metric other than "ROC" directly.

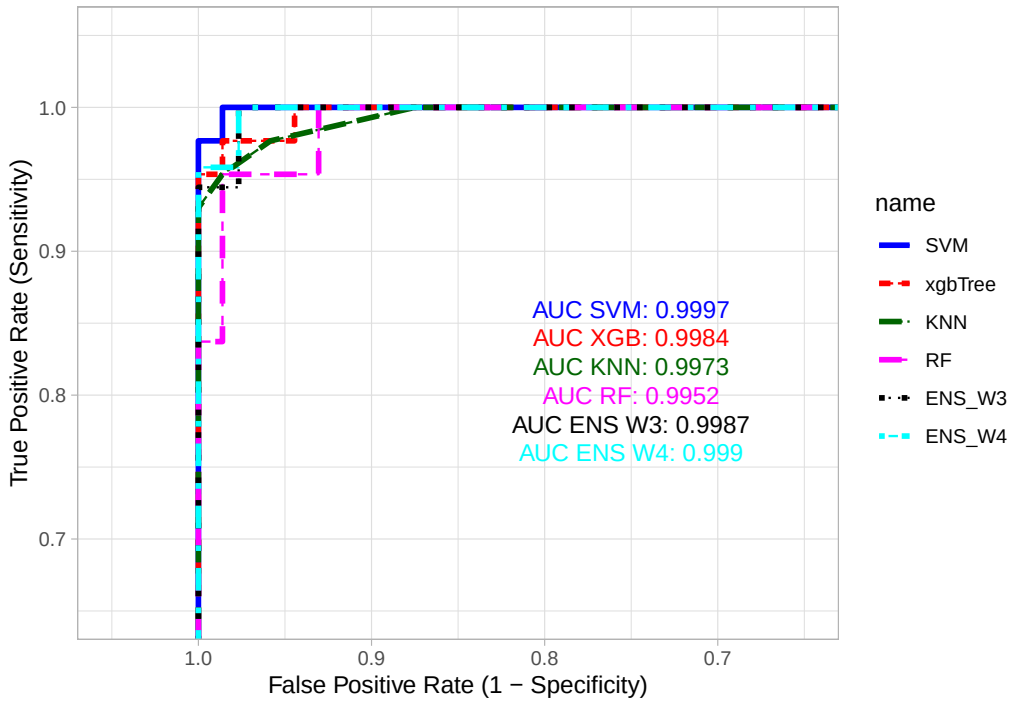

Figure 10: ROC Curves for Breast Cancer Classification Models (best models)

On the other hand, Figure 11 show the ROC curves for models trained with PCA preprocessing. Similar to the non-PCA models, all PCA-applied models maintain high AUCs, confirming their strong discriminative ability. The SVM (0.9994) again shows the highest AUC among the PCA models, closely followed by ENS

W3 (0.9984) and KNN (0.9981). The SVM AUC of 0.9994 from this plot is higher than its svmRadial_PCA ROC-optimized entry in the Table 9 (0.9826087), but aligns with its Sensitivity-optimized AUC (0.9993540) from Table 8. The ensemble models (ENS W3 and ENS W4) continue to perform strongly, with AUCs consistent with their F1-optimized entries in the table.

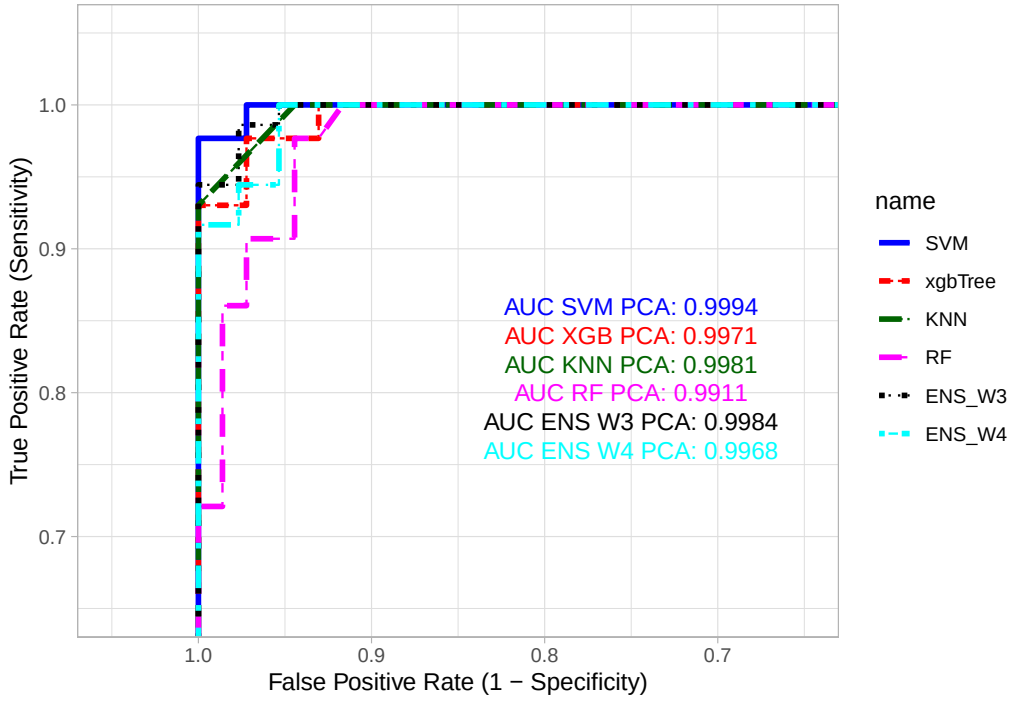

Figure 11: ROC Curves for Breast Cancer Classification Models (PCA data, best models)

6 Conclusions

This analysis presented here, showed how using different optimization goals affects the performance of machine learning models for breast cancer detection. Here were tested k-Nearest Neighbors (knn), eXtreme Gradient Boosting (xgbTree), Support Vector Machine (svmRadial), and ensemble methods, both with and without Principal Component Analysis (PCA). Tuning the models to focus on Area Under the Curve (ROC), Sensitivity, Specificity, or F1-score worked well and made sense for medical use, because it helps match the models to different diagnostic needs.

The ROC curve analysis reinforced several points from the table-based analysis. The consistently high AUC values (all above 0.99 for the top performers) across both PCA and non-PCA models indicate that the models are highly effective at distinguishing between positive and negative breast cancer cases. Visually, the ROC curves confirm that ensemble models and SVM (both with and without PCA) generally achieve the highest discriminative performance, as their curves are closest to the ideal top-left corner.

It was demonstrated the superior Performance of Ensemble and SVM Models. The ensemble methods (e.g., ensemble wp4 P_OOF, ensemble wp3 P_OOF) consistently achieved the highest overall performance, demonstrating near-perfect Accuracy (0.9913043) and Specificity (1.0000000), coupled with high Sensitivity (0.9767442) and AUC (0.9990310). This highlights the benefit of combining multiple models for robust classification. SVM models (svmRadial, svmRadial_PCA) also exhibited strong performance, particularly when optimized for Sensitivity or ROC, achieving high Accuracy, Sensitivity, Specificity, and AUC values (e.g., svmRadial Sens with AUC 0.9996770).

The PCA impact on model performance varied. For SVM and KNN, PCA generally maintained or slightly improved performance, particularly in achieving perfect Specificity in some configurations. For XGBoost and

Random Forest, PCA sometimes led to minor reductions in AUC compared to their non-PCA counterparts.

References

Cohen, J. 1988. *Statistical Power Analysis for the Behavioral Sciences*. 2nd ed. Routledge. https://doi.org/10.4324/9780203771587.

Duda, Richard O., Peter E. Hart, and David G. Stork. 2001. *Pattern Classification*. 2nd ed. New York, NY, USA: Wiley-Interscience.

Irizarry, Rafael A. 2020. *Introduction to Data Science: Data Analysis and Prediction Algorithms with r*. CRC Press.

Ong, Mei-Sing, and Kenneth D. Mandl. 2015. *Health Affairs* 34 (4). https://doi.org/10.1377/hlthaff.2014.1087.

Royston, J. P. 1982. "An Extension of Shapiro and Wilk's w Test for Normality to Large Samples." *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 31 (2): 115–24. http://www.jstor.org/stable/2347973.

Stephens, M. A. 1986. *Tests Based on EDF Statistics*. Edited by R. B. D'Agostino and M. A. Stephens. Goodness-of-Fit Techniques. Marcel Dekker.

Street, William Nick, William H. Wolberg, and Olvi L. Mangasarian. 1993. "Nuclear Feature Extraction for Breast Tumor Diagnosis." In *Electronic Imaging*. https://api.semanticscholar.org/CorpusID:14922543.

Thode, H. C. 2002. *Testing for Normality*. CRC Press. https://doi.org/doi.org/10.1201/9780203910894.

Tomczak, Maciej, and Ewa Tomczak. 2014. "The Need to Report Effect Size Estimates Revisited. An Overview of Some Recommended Measures of Effect Size." *Trends in Sport Sciences* 1 (21): 19–25.