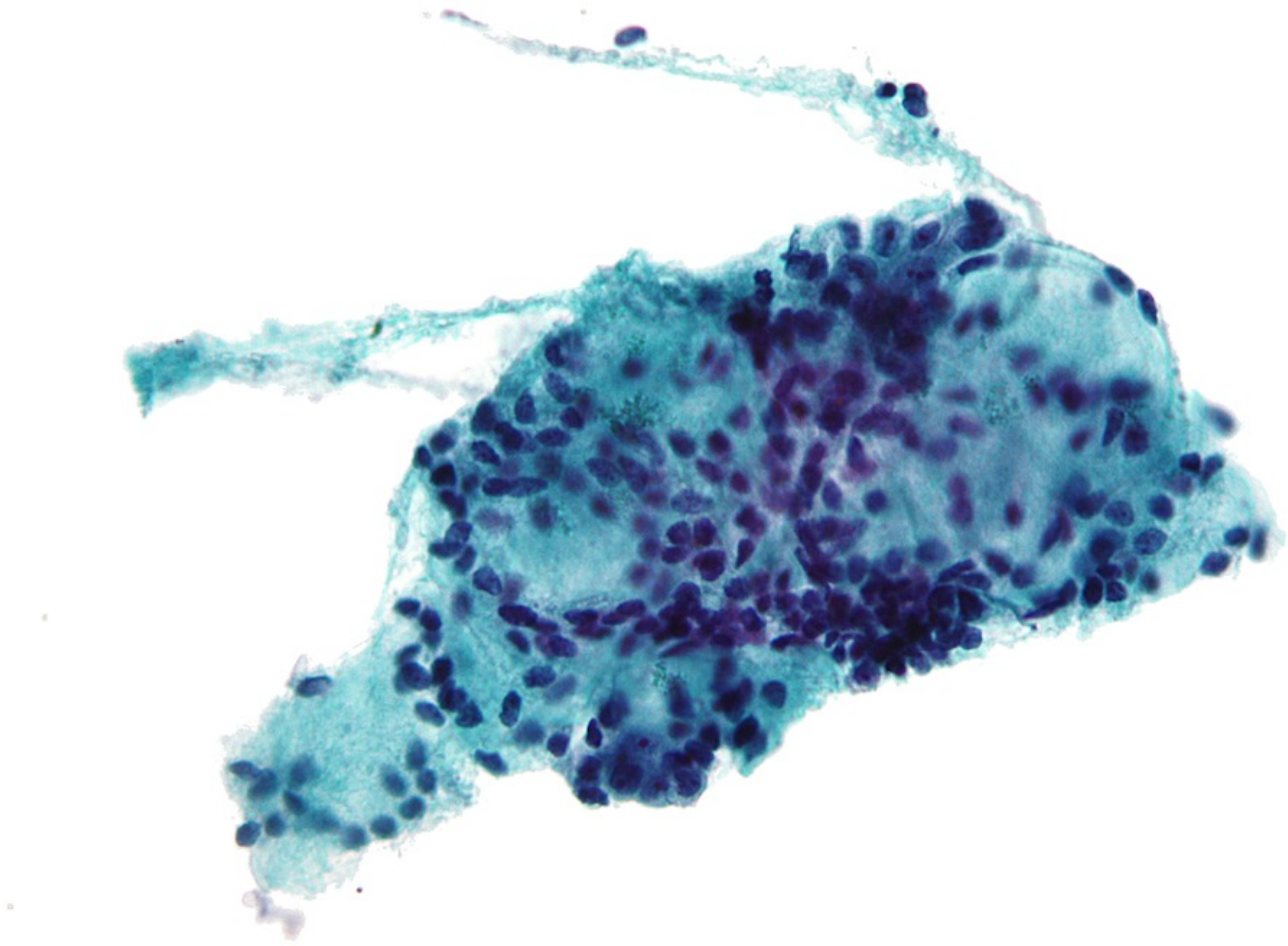
# Applying Data Mining Techniques to Breast Cancer Analysis



Luís Rei <ei05103@fe.up.pt>
José Paixão <ei04041@fe.up.pt>
Sérgio Vasconcelos <ei05074@fe.up.pt>

Knowledge Extraction and Machine Learning
Faculty of Engineering of the University of Porto
April 2011

# 1. Introduction

Fine Needle Aspirate (FNA) is a diagnostic procedure used to investigate superficial lumps or masses. In this technique, a thin, hollow needle is inserted into the mass to extract cells that, after being stained, will be examined under a microscope. The procedure serves as an alternative to usual biopsy methods that require major surgery. This work explores the use of data mining algorithms to enhance early detection of breast cancer by diagnosing breast masses from magnified images of FNAs [1].

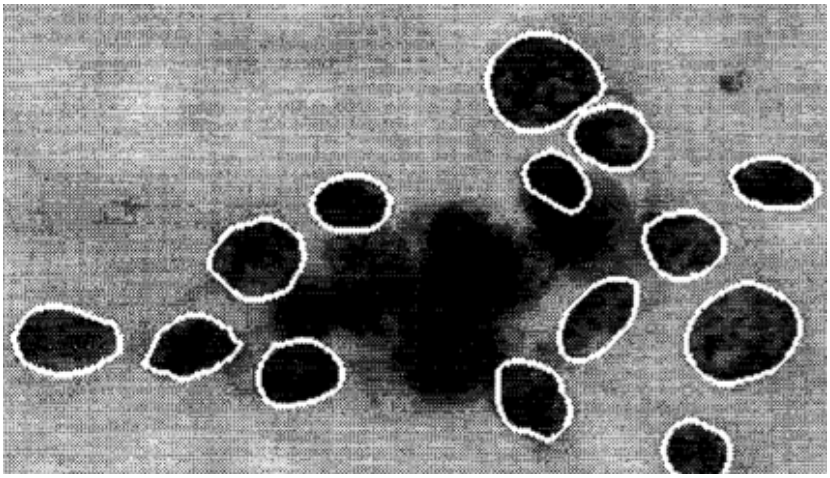


Figure 1.1 - A magnified image of a malignant breast FNA. The visible cell nuclei has been outlined with the help of a curve fitting program. Adapted from [1].

The cancer database used was the Wisconsin Breast Cancer Database (Original) dated 15 July 1992 which was obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg [1,2] and can be retrieved from UC Irvine Machine Learning Repository [3].

The purpose of the analysis is to indicate, from a set of image parameters, whether the mass represents a malignant or benign growth. This type of task is known as a **classification** problem where a model or classifier is constructed to predict categorical labels, such as “malignant” or “benign” for the FNA image data, such as “Uniformity of Cell Shape = 5”. As another example, a classifier could be also be built to define a loan as “safe” or “risky” from loan application data or to classify an e-mail message as “unsolicited” (also known as “spam”).

This work loosely follows the *CRISP-DM* (CRoss Industry Standard Process for Data Mining) process model [9], shown in Figure 1.2, that describes commonly used approaches that expert data miners use to tackle problems. It breaks down the Data Mining process into six major phases: business understanding, data understanding, data preparation, modeling,

evaluation and deployment. The main difference is that there was no deployment phase. The data preparation phase, which was small due to the fact that the work was being done on an already prepared dataset, is integrated in the “data understanding” chapter.

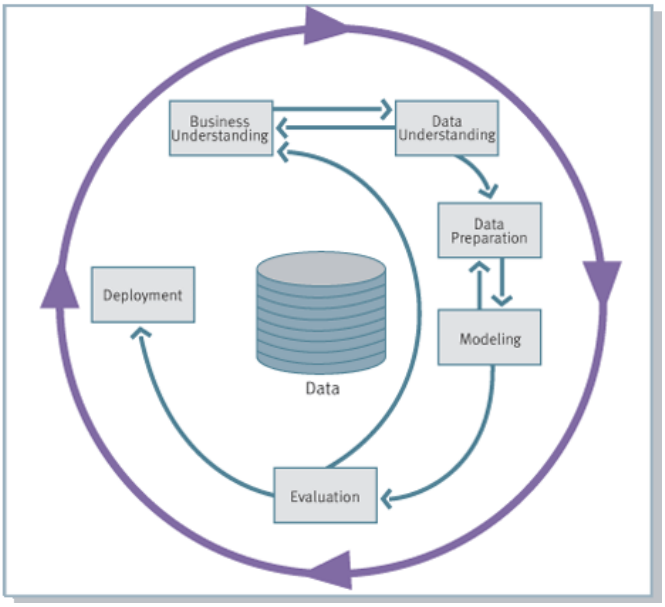



Figure 1.2 - CRISP-DM process model diagram. Adapted from [9].

The tool used to perform the analysis described in the following chapters was RapidMiner, an open-source system for data mining available as a stand-alone application for data analysis and as a data mining engine for the integration into other products [4].

This report was written for the Knowledge Extraction and Machine Learning course at the Faculty of Engineering University of Porto.

# 2. Data Understanding and Preparation

The dataset, as obtained from [3], consists of a CSV (Comma-Separated Values) file containing 699 instances with 11 attributes organized according to Table 2.1.

| Attribute Number | Attribute Name | Domain |
|---|---|---|
| 1 | Sample code number (id) | id number |
| 2 | Clump Thickness | integer [1, 10] |
| 3 | Uniformity of Cell Size | integer [1, 10] |
| 4 | Uniformity of Cell Shape | integer [1, 10] |
| 5 | Marginal Adhesion | integer [1, 10] |
| 6 | Single Epithelial Cell Size | integer [1, 10] |
| 7 | Bare Nuclei | integer [1, 10] |
| 8 | Bland Chromatin | integer [1, 10] |
| 9 | Normal Nucleoli | integer [1, 10] |
| 10 | Mitoses | integer [1, 10] |
| 11 | Class | bi-nominal<br>2 for benign, 4 for malignant |

Table 2.1 - Attributes Table

The following lines were extracted from the file to provide an example as to its contents:

```
1320141,5,1,1,1,2,1,2,1,1,2
1325309,4,1,2,1,2,1,2,1,1,2
1333063,5,1,3,1,2,1,3,1,1,2
1333495,3,1,1,1,2,1,2,1,1,2
1334659,5,2,4,1,1,1,1,1,1,2
1336798,3,1,1,1,2,1,2,1,1,2
```

There are 16 instances in Groups 1 to 6 that contain a single missing (i.e., unavailable) attribute value, denoted by "?". The dataset has a class distribution according to Table 2.2.

| Class | Number of Instances | Percentage |
|---|---|---|
| Benign | 458 | 65.5% |
| Malignant | 241 | 34.5% |

Table 2.2 - Class Distribution Table

The *id* field was, obviously, not used during the analysis, since it contains no information useful for the classification process, hence reducing the number of attributes. The 16 instances with a missing value were removed. This is reasonable since 16 instances represent only ~2% of the original dataset and a small percentage of each class. This approach is consistent with that of other authors [1,2]. Thus the pre-processed dataset contains only 683 instances with 9 attributes and the class label. The resulting distribution is present in Table 2.3:

| Class | Number of Instances | Percentage |
|---|---|---|
| Benign | 444 | 65% |
| Malignant | 239 | 35% |

Table 2.3 - Class Distribution Table after removing missing values

## 2.1. Clump Thickness

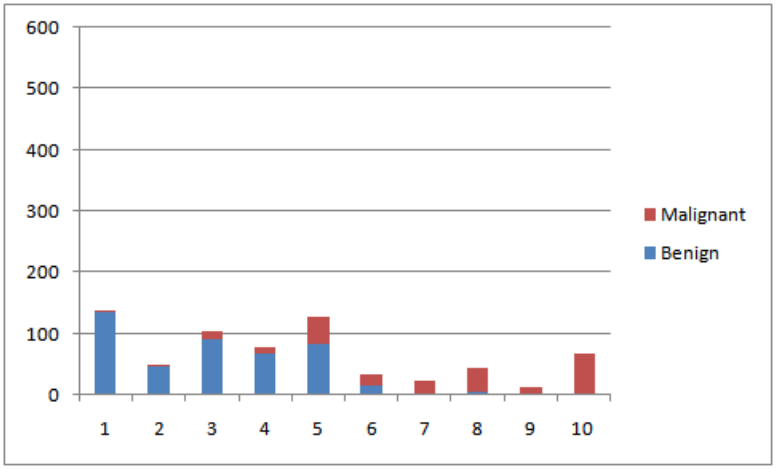


Figure 2.1 - Clump Thickness Attribute Data Bar Graph

| Clump Thickness | Benign | Malignant | Total |
|---|---|---|---|
| 1 | 136 | 3 | 139 |
| 2 | 46 | 4 | 50 |
| 3 | 92 | 12 | 104 |
| 4 | 67 | 12 | 79 |
| 5 | 83 | 45 | 128 |
| 6 | 15 | 18 | 33 |
| 7 | 1 | 22 | 23 |
| 8 | 4 | 40 | 44 |
| 9 | 0 | 14 | 14 |
| 10 | 0 | 69 | 69 |
| Total | 444 | 239 | 683 |

Table 2.4 - Clump Thickness Attribute Data Distribution in Each Class

Average = 4.442
Standard deviation = 2.821
Median = 4
Mode = 1

## 2.2. Uniformity of Cell Size

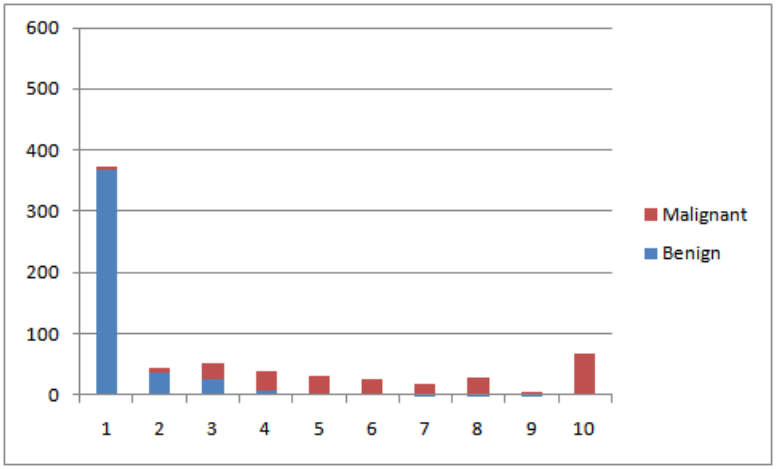


Figure 2.2 - Uniformity of Cell Size Attribute Data Bar Graph

| Uniformity of Cell Size | Benign | Malignant | Total |
|---|---|---|---|
| 1 | 369 | 4 | 373 |
| 2 | 37 | 8 | 45 |
| 3 | 27 | 25 | 52 |
| 4 | 8 | 30 | 38 |
| 5 | 0 | 30 | 30 |
| 6 | 0 | 25 | 25 |
| 7 | 1 | 18 | 19 |
| 8 | 1 | 27 | 28 |
| 9 | 1 | 5 | 6 |
| 10 | 0 | 67 | 67 |
| Total | 444 | 239 | 683 |

Table 2.5 - Uniformity of Cell Size Attribute Data Distribution in Each Class

Average = 3.151
Standard deviation = 3.065
Median = 1
Mode = 1

## 2.3. Uniformity of Cell Shape

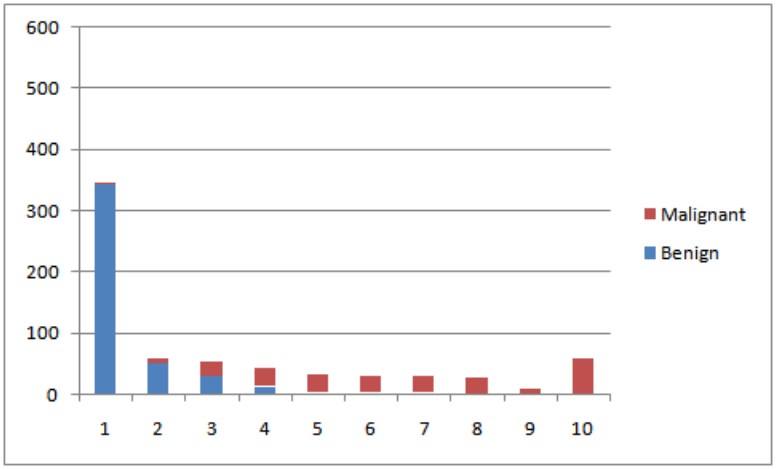



Figure 2.3 - Uniformity of Cell Shape Attribute Data Bar Graph

| Uniformity of Cell Shape | Benign | Malignant | Total |
|---|---|---|---|
| 1 | 344 | 2 | 346 |
| 2 | 51 | 7 | 58 |
| 3 | 30 | 23 | 53 |
| 4 | 12 | 31 | 43 |
| 5 | 2 | 30 | 32 |
| 6 | 2 | 27 | 29 |
| 7 | 2 | 28 | 30 |
| 8 | 1 | 26 | 27 |
| 9 | 0 | 7 | 7 |
| 10 | 0 | 58 | 58 |
| Total | 444 | 239 | 683 |

Table 2.6 - Uniformity of Cell Shape Attribute Data Distribution in Each Class

Average = 3.215
Standard deviation = 2.989
Median = 1
Mode = 1

# 2.4. Marginal Adhesion

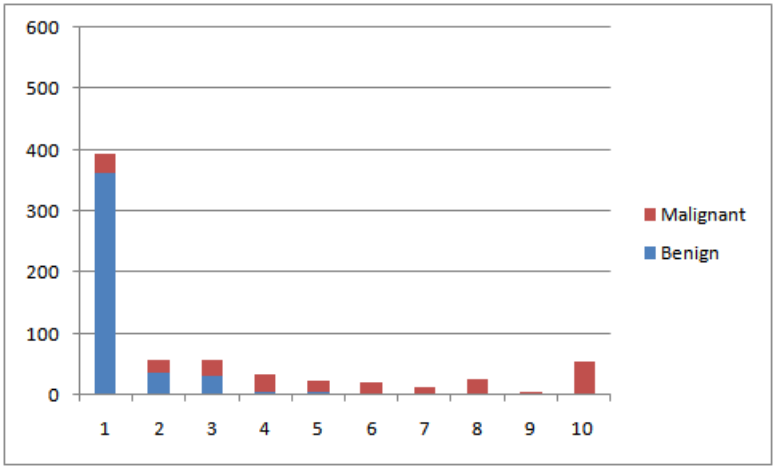


Figure 2.4 - Marginal Adhesion Attribute Data Bar Graph

| Marginal Adhesion | Benign | Malignant | Total |
|---|---|---|---|
| 1 | 363 | 30 | 393 |
| 2 | 37 | 21 | 58 |
| 3 | 31 | 27 | 58 |
| 4 | 5 | 28 | 33 |
| 5 | 4 | 19 | 23 |
| 6 | 3 | 18 | 21 |
| 7 | 0 | 13 | 13 |
| 8 | 0 | 25 | 25 |
| 9 | 0 | 4 | 4 |
| 10 | 1 | 54 | 55 |
| Total | 444 | 239 | 683 |

Table 2.7 - Marginal Adhesion Attribute Data Distribution in Each Class

Average = 2.830
Standard deviation = 2.865
Median = 1
Mode = 1

## 2.5. Single Epithelial Cell Size

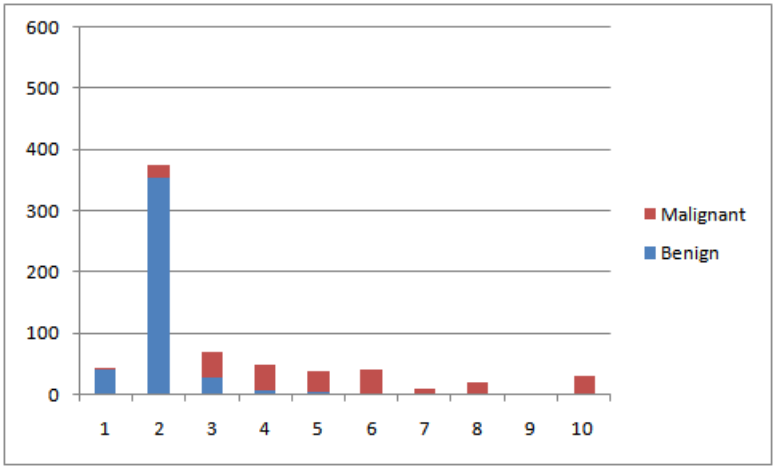



Figure 2.5 - Single Epithelial Attribute Data Bar Graph

| Single Epithelial Cell Size | Benign | Malignant | Total |
|---|---|---|---|
| 1 | 43 | 1 | 44 |
| 2 | 355 | 21 | 376 |
| 3 | 28 | 43 | 71 |
| 4 | 7 | 41 | 48 |
| 5 | 5 | 34 | 39 |
| 6 | 1 | 39 | 40 |
| 7 | 2 | 9 | 11 |
| 8 | 2 | 19 | 21 |
| 9 | 0 | 2 | 2 |
| 10 | 1 | 30 | 31 |
| Total | 444 | 239 | 683 |

Table 2.8 - Single Epithelial Attribute Data Distribution in Each Class

Average = 3.234
Standard deviation = 2.223
Median = 2
Mode = 2

# 2.6. Bare Nuclei

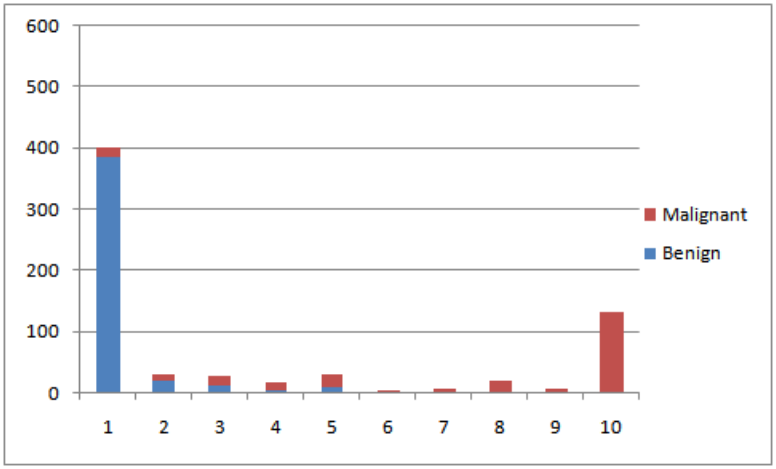



Figure 2.6 - Bare Nuclei Attribute Data Bar Graph

| Bare Nuclei | Benign | Malignant | Total |
|---|---|---|---|
| 1 | 387 | 15 | 402 |
| 2 | 21 | 9 | 30 |
| 3 | 14 | 14 | 28 |
| 4 | 6 | 13 | 19 |
| 5 | 10 | 20 | 30 |
| 6 | 0 | 4 | 4 |
| 7 | 1 | 7 | 8 |
| 8 | 2 | 19 | 21 |
| 9 | 0 | 9 | 9 |
| 10 | 3 | 129 | 132 |
| Total | 444 | 239 | 683 |

Table 2.9 - Bare Nuclei Attribute Data Distribution in Each Class

Average = 3.545
Standard deviation = 3.644
Median = 1
Mode = 1

# 2.7. Bland Chromatin

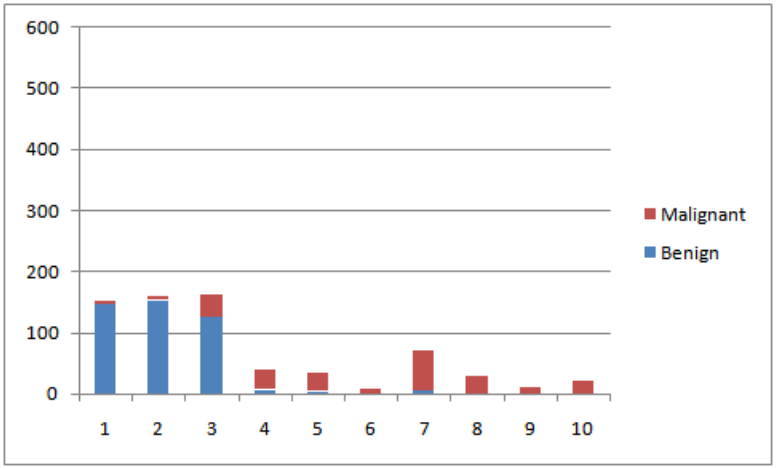


Figure 2.7 - Bland Chromatin Attribute Data Bar Graph

| Bland Chromatin | Benign | Malignant | Total |
|---|---|---|---|
| 1 | 148 | 2 | 150 |
| 2 | 153 | 7 | 160 |
| 3 | 125 | 36 | 161 |
| 4 | 7 | 32 | 39 |
| 5 | 4 | 30 | 34 |
| 6 | 1 | 8 | 9 |
| 7 | 6 | 65 | 71 |
| 8 | 0 | 28 | 28 |
| 9 | 0 | 11 | 11 |
| 10 | 0 | 20 | 20 |
| Total | 444 | 239 | 683 |

Table 2.10 - Bland Chromatin Attribute Data Distribution in Each Class

Average = 3.445
Standard deviation = 2.450
Median = 3
Mode = 3

# 2.8. Normal Nucleoli

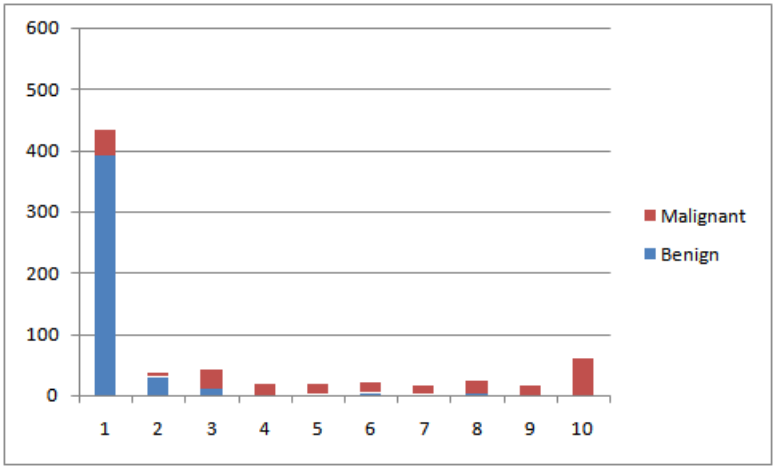


Figure 2.8 - Normal Nucleoli Attribute Data Bar Graph

| Normal Nucleoli | Benign | Malignant | Total |
|---|---|---|---|
| 1 | 391 | 41 | 432 |
| 2 | 30 | 6 | 36 |
| 3 | 11 | 31 | 42 |
| 4 | 1 | 17 | 18 |
| 5 | 2 | 17 | 19 |
| 6 | 4 | 18 | 22 |
| 7 | 2 | 14 | 16 |
| 8 | 3 | 20 | 23 |
| 9 | 0 | 15 | 15 |
| 10 | 0 | 60 | 60 |
| Total | 444 | 239 | 683 |

Table 2.11 - Normal Nucleoli Attribute Data Distribution in Each Class

Average = 2.870
Standard deviation = 3.053
Median = 1
Mode = 1

# 2.9. Mitosis

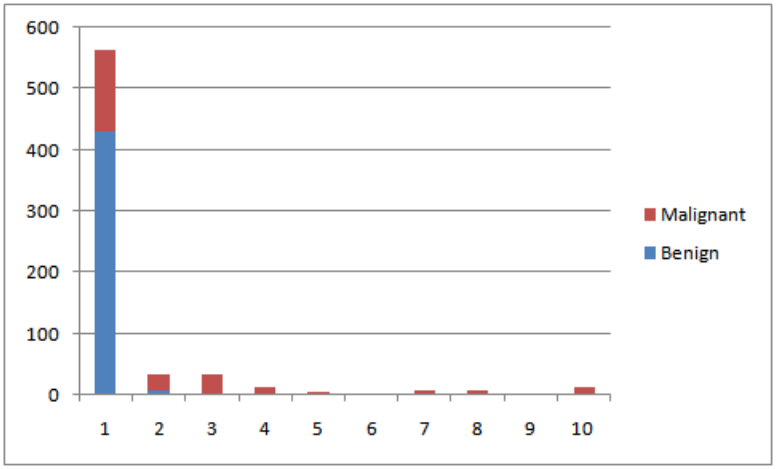



Figure 2.9 - Mitosis Attribute Data Bar Graph

| Mitosis | Benign | Malignant | Total |
|---|---|---|---|
| 1 | 431 | 132 | 563 |
| 2 | 8 | 27 | 35 |
| 3 | 2 | 31 | 33 |
| 4 | 0 | 12 | 12 |
| 5 | 1 | 5 | 6 |
| 6 | 0 | 3 | 3 |
| 7 | 1 | 8 | 9 |
| 8 | 1 | 7 | 8 |
| 9 | 0 | 0 | 0 |
| 10 | 0 | 14 | 14 |
| Total | 444 | 239 | 683 |

Table 2.12 - Mitosis Attribute Data Distribution in Each Class

Average = 1.603
Standard deviation = 1.733
Median = 1
Mode = 1

# 2.10. Data Analysis

Figure 2.10 - Scatter plot matrix

The scatter plot matrix in Figure 2.10 shows that most variables are not correlated. The exception is the high correlation between the *Uniformity of Cell Shape* and the *Uniformity of Cell Size* which has a correlation factor of 0.9. These two are also correlated, with a smaller factor of approximately 0.7, with the attribute *Single Epithelial Cell Size*. These correlations make obvious biological sense and were to be expected. The correlation between *Uniformity of Cell Size* and *Uniformity of Cell Shape* is high enough to justify considering the removal of one of these attributes from the input data if the training set was very large and training time was important. It is important to note that high correlation does not mean that the attributes do not contain useful information (i.e. it does not mean they are necessarily redundant).

An experiment was made in which *Uniformity of Cell Size* was removed from the dataset. The resulting accuracy of the algorithms did not change significantly, therefore, it would be acceptable to remove this attribute although this was not done since it also provides no advantages. Further analysis of the data, namely factor analysis, could prove useful.

# 3. Modelling

## 3.1. Introduction

Data classification is a two-step process: In the first step, a classifier is built describing a predetermined set of data classes or concepts. This is the learning step (or training phase), where a classification algorithm builds the classifier by learning from a training set made up of instances with their associated class labels. An instance, *X*, is represented by an n-dimensional attribute (or feature) vector, X = ($x_1$, $x_2$,..., $x_n$). Each instance is assumed to belong to a predefined **class** as determined its associated **class label** attribute. The class label attribute is discrete-valued, unordered and categorical, i.e., each value serves as a category or class. Because the class label of each instance in the training set is provided, this step is also known as **supervised learning**, the classifier is "supervised" in that it is told to which class each instance belongs. In the second step, the classifier is provided with additional instances, without associated class labels, and must generalize in order to decide what class that instance belongs to [5,7].

Classification problems include credit scoring which consists of assigning risk to a loan based on credit, biometric identification which could be matching fingerprints to an individual based on features extracted from an image of a fingerprint or document classification such as classifying an E-mail as "unsolicited" or not. The problem presented here consists in determining whether a mass is benign (noncancerous) or malignant (cancerous) based on features of a microscopic image of a cells extracted from that mass.

In this chapter several classification algorithms are described, applied and their results are analysed. Finally, the algorithms are compared in terms of accuracy. To do this we built the process in RapidMiner that is depicted in Figure 3.1.

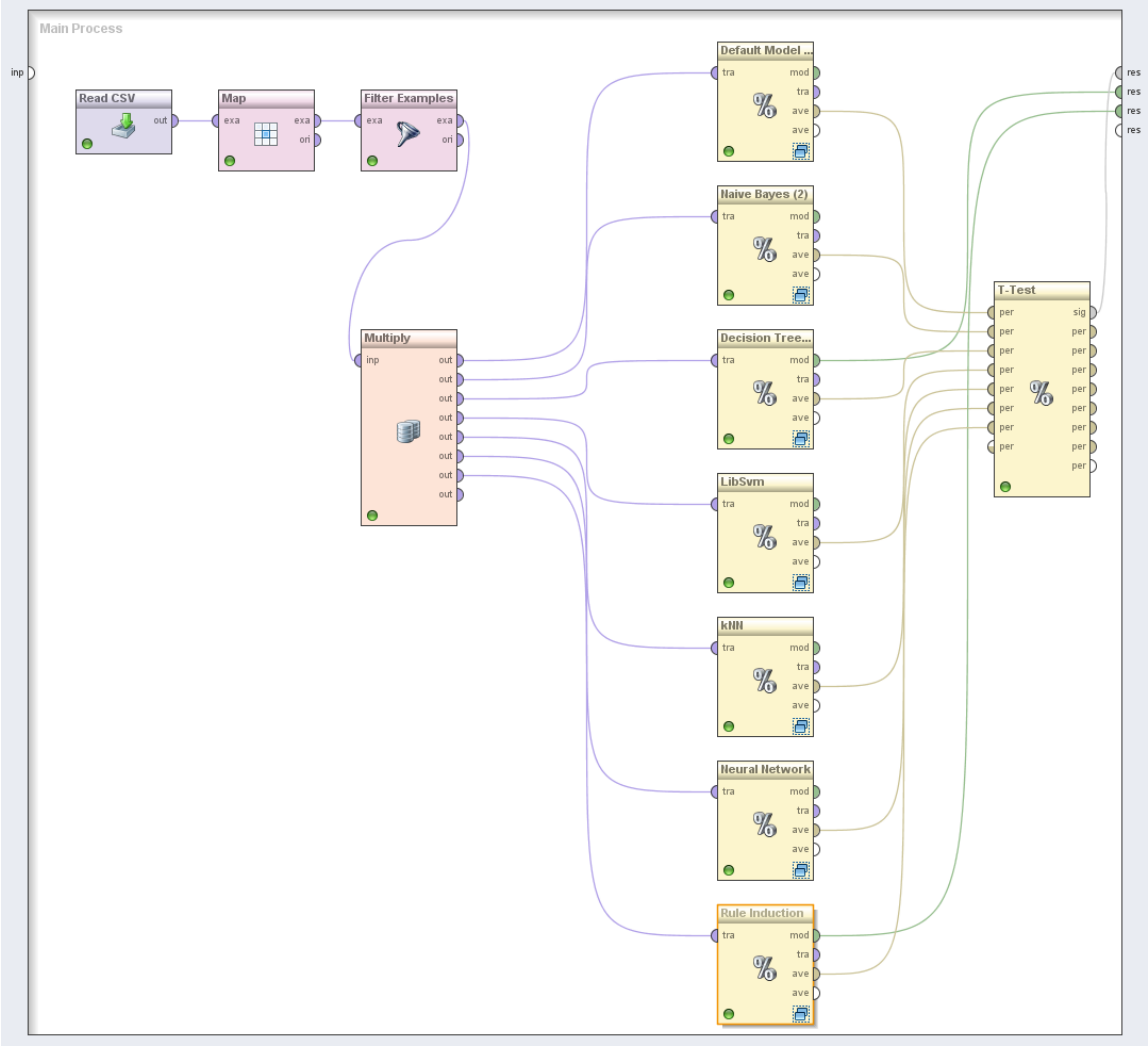



Figure 3.1 - RapidMiner process used for the experiments

The process model is divided into three main stages. The first consists of loading the data set, stored in a CSV file (Read CSV component), and mapping some attributes labels to more intuitive ones (Map component). Also, missing values had to be ignored and the Filter Examples component was used. In order to apply a number of classifiers to the dataset, we used the Multiply component. Each one of this components ports is then connected to a classifier component. In the center of the process model, the seven different classification methods are shown. Although not visible in the picture, each of the components may be expanded in order to configure the settings for the training and testing phases. The training part applies a classification algorithm, while the training part is responsible for applying the model and measuring the classifier performance.

Finally, the results of each classifier are then passed to the T-Test component. This last component will allow for the comparison between the accuracy of the different classification methods used.

# 3.2 Evaluation

The **accuracy** of a classifier *M*, *Acc(M)*, on a given test set is the percentage of test set instances that are correctly classified by the classifier. The **error rate** or **misclassification rate** of a classifier, M, is simply 1 − *Acc(M)*.

The **confusion matrix** is a useful tool for analyzing how well your classifier can recognize instances of different classes. An example confusion matrix for the two classes in our dataset is shown in Table 3.1.

| True: | Benign | Malignant |
|---|---|---|
| Benign: | 433 | 8 |
| Malignant: | 11 | 231 |

Table 3.1 - An example confusion matrix.

Given *m* classes, a confusion matrix is a table of at least size *m* by *m*. An entry, $CM_{i,j}$ in the first *m* rows and *m* columns indicates the number of instances of class *i* that were labeled by the classifier as class *j*. For a classifier to have good accuracy, ideally most of the instances would be represented along the diagonal of the confusion matrix, from entry $CM_{1,1}$ to entry $CM_{m,m}$, with the rest of the entries being close to zero. Thus our confusion matrix indicates that 433 Benign instances were correctly classified and 8 were incorrectly classified as Malignant while 231 malignant instances were correctly classified while 11 were incorrectly classified as Benign.

The fitting process (training) optimizes the model parameters to make the model fit the training data as well as possible. If the model created is then used to classify an independent data sample (i.e. a sample that was not part of data used in the training process), it will generally turn out that the model does not fit this new data as well as it fits the training data. This is called **overfitting**, and is particularly likely to happen when the size of the training data set is small, or when the number of parameters in the model is large. **Cross-validation** is a way to predict the fit of a model to a hypothetical validation set when an explicit validation set is not available. In **k-fold cross-validation**, the initial data are randomly partitioned into k mutually exclusive subsets or “folds,” $D_1$, $D_2$,..., $D_k$, each of approximately equal size. Training and testing is performed *k* times. In iteration *i*, partition $D_i$ is reserved as the test set, and the remaining partitions are collectively used to train the model. For classification, the accuracy estimate is the overall number of correct classifications from the *k* iterations, divided by the total number of

instances in the initial data. In general, k=10 is recommended for estimating accuracy [5].

Using overall accuracy or a similar measure makes the assumption that all errors are equally costly. This seems to be an erroneous approach for most knowledge discovery applications. In no application is that more evident than in medical diagnosis: The **cost** associated with a false negative such as, incorrectly predicting that a cancerous patient is not cancerous, is far greater than that of a false positive, incorrectly yet conservatively labeling a noncancerous patient as cancerous. It is far worse for a cancerous patient to go untreated than it is for a noncancerous patient to be subjected to further tests. In such cases, we can outweigh one type of error over another by assigning a different cost to each [5]. Misclassification costs may be represented by a **cost matrix** *C*, with $C_{i,j}$ being the cost of predicting that an example belongs to class *i* when in fact it belongs to class *j* [6]. To address this problem, the **MetaCost** solution proposed in [6] which is based on wrapping a meta-learning stage around the classifier in such a way that the classifier effectively minimizes cost while seeking to minimize the error rate (maximize accuracy). For each classification method, we manually adjusted the cost of false negatives. This adjustment was based on observation, with the objective of minimizing the occurrence of false negatives, without, however, disregarding the overall accuracy of the classifiers. The cost measure should not be understood a measure of the cost of these errors in real life, rather it is an artificial value for the algorithm that while related, is not necessarily directly related to the real life cost of misclassification. This value was arrived at through trial and error experimentation and should be considered good but not perfect (no automatic optimization process was performed on this value).

In the next sub-sections the classifiers used are described and their confusion matrices are shown, with and without the usage of cost.

## 3.3. k-Nearest-Neighbor

In the k-nearest-neighbor classifier, each instance represents a point in a n-dimensional space. When given an unknown instance, a *k*-nearest-neighbor classifier searches the pattern space for the *k* training instances that are closest to the unknown instance. These *k* training instances are the *k* “nearest neighbors” of the unknown instance. “Closeness” is defined in terms of a distance metric, such as Euclidean distance, the one used in this classification work. In our approach, we used k = 10.

**Accuracy: 97.07%**

| True: | Benign | Malignant |
|---|---|---|
| Benign: | 436 | 12 |
| Malignant: | 8 | 227 |

Table 3.2 - kNN Confusion Matrix

**False Negative Cost: 10**
**Accuracy: 96.80%**

| True: | Benign | Malignant |
|---|---|---|
| Benign: | 427 | 2 |
| Malignant: | 17 | 237 |

Table 3.3 - k-NN with MetaCost Confusion Matrix

# 3.4. Decision Tree

Decision tree induction is the learning of decision trees from class-labeled training instances. A decision tree is a flowchart-like tree structure, where each internal node (nonleaf node) denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (or terminal node) holds a class label. The topmost node in a tree is the root node. Decision trees are powerful classification methods which often can also be easily understood.

The decision tree learner used in RapidMiner works similar to Quinlan's C4.5 [10] or CART. Roughly speaking, the tree induction algorithm works as follows. Whenever a new node is created at a certain stage, an attribute is picked to maximise the discriminative power of that node with respect to the examples assigned to the particular subtree. This discriminative power is measured by a criterion which can be selected by the user (information gain, gain ratio, gini index, etc.). The used criterion in our approach was the gain ratio (default parameter).

**Accuracy: 95.31%**

| True: | Benign | Malignant |
|---|---|---|
| Benign: | 431 | 19 |
| Malignant: | 13 | 220 |

Table 3.4 - Decision Tree Confusion Matrix

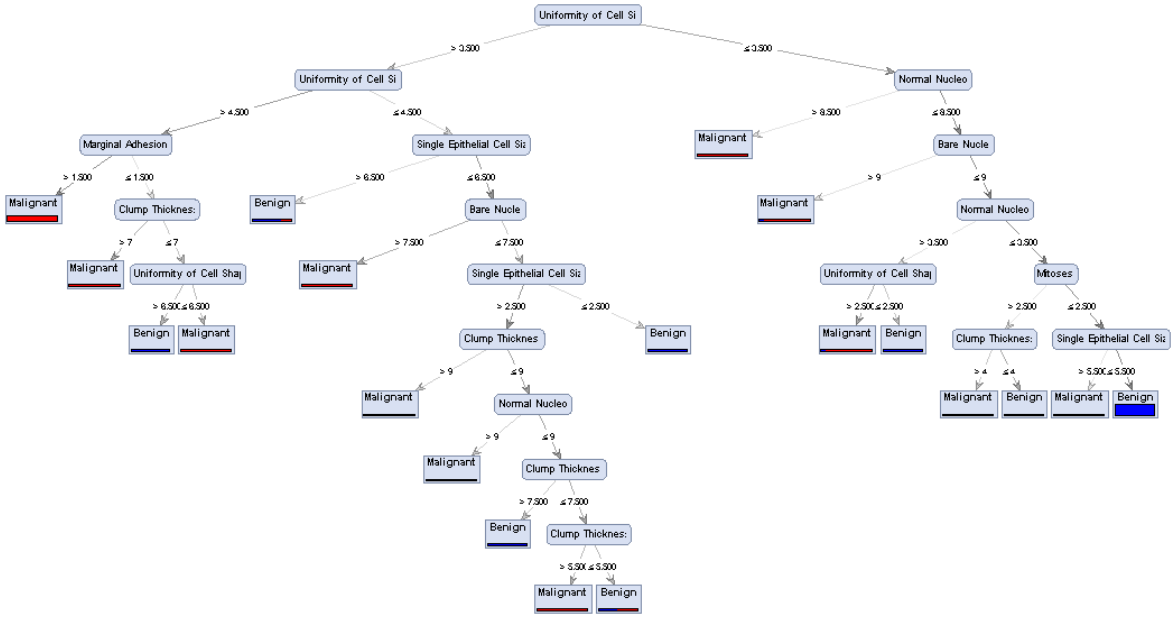



Figure 3.2 - Resulting Decision Tree

**False Negative Cost: 4**
**Accuracy: 95.10%**

| True: | Benign | Malignant |
|---|---|---|
| Benign: | 419 | 1 |
| Malignant: | 25 | 238 |

Table 3.5 - Decision Tree with MetaCost Confusion Matrix

# 3.5. Naive Bayes

A Naive Bayes classifier is a simple probabilistic classifier which applies Bayes theorem with strong independence assumptions. A Naive Bayes classifier assumes that the presence or absence of a particular feature of a class is unrelated to the presence or absence of any other feature.

**Accuracy: 96.20%**

| True: | Benign | Malignant |
|---|---|---|
| Benign: | 424 | 6 |
| Malignant: | 20 | 233 |

Table 3.6 - Naive Bayes Confusion Matrix

**False Negative Cost: 11**
**Accuracy: 95.20%**

| True: | Benign | Malignant |
|---|---|---|
| Benign: | 418 | 2 |
| Malignant: | 26 | 237 |

Table 3.7 - Naive Bayes with MetaCost Confusion Matrix

# 3.6. Support Vector Machine

A support vector machine algorithm uses a nonlinear mapping to transform the original training data into a higher dimension. Within this new dimension, it tries to find the linear optimal decision boundary, separating the instances of one class from another. In this experiment, we used a C-SVC SVM with a *rbf* kernel function. The used Gamma and C parameters were the default (0.0).

**Accuracy: 97.22%**

| True: | Benign | Malignant |
|---|---|---|
| Benign: | 433 | 8 |
| Malignant: | 11 | 231 |

Table 3.8 - Support Vector Machines Confusion Matrix

**False Negative Cost: 3**
**Accuracy: 92.24%**

| True: | Benign | Malignant |
|---|---|---|
| Benign: | 424 | 0 |
| Malignant: | 53 | 239 |

Table 3.9 - Support Vector Machines with MetaCost Confusion Matrix

## 3.7. Rule Induction (RIPPER)

*"IF-THEN"* rules can be extracted directly from the training data, using sequencial covering algorithms, without the need of generating a decision tree first. The rules are learned sequentially, where each rule for a given class will ideally cover many of the instances of that class. The sequential covering algorithm used in this problem was RIPPER [11]. This operator works similar to the propositional rule learner named Repeated Incremental Pruning to Produce Error Reduction (RIPPER, Cohen 1995). The criterion used for selecting attributes and numeric splits was based on the information gain.

```
if Uniformity of Cell Shape ≤ 2.500
then Benign

if Marginal Adhesion > 5.500
then Malignant

if Uniformity of Cell Shape > 8.500
then Malignant

else Malignant
```

Figure 3.3 - Resulting Rules

**Accuracy: 95.17%**

| True: | Benign | Malignant |
|---|---|---|
| Benign: | 430 | 19 |
| Malignant: | 14 | 220 |

Table 3.10 - RIPPER Confusion Matrix

**Accuracy: 95.00%**

| True: | Benign | Malignant |
|---|---|---|
| Benign: | 416 | 6 |
| Malignant: | 28 | 233 |

Table 3.11 - RIPPER with MetaCost Confusion Matrix

# 3.8. Backpropagation ANN

Backpropagation is a neural network learning algorithm. A neural network is a set of connected input and output units in which each connection has a weight associated with it. During the learning phase, the network learns by adjusting the weights in order to be able to predict the correct class labels of the input instances.

In our approach, we used a feed-forward neural network trained by a backpropagation algorithm (multi-layer perception). Concerning the parameters for the neural network, the default values of RapidMiner were used. The number of training cycles was 500, the learning rate was 0.3 and the momentum 0.2. No hidden layers were used since this would significantly increase the training time.

**Accuracy: 96.14%**

| True: | Benign | Malignant |
|---|---|---|
| Benign: | 430 | 13 |
| Malignant: | 14 | 226 |

Table 3.12 - Backpropagation ANN Confusion Matrix

**False Negative Cost: 2**
**Accuracy: 96.04%**

| True: | Benign | Malignant |
|---|---|---|
| Benign: | 420 | 3 |
| Malignant: | 24 | 236 |

Table 3.13 - Backpropagation ANN with MetaCost Confusion Matrix

# 3.9. Default Model: ZeroR

The ZeroR classifier simply predicts the majority category (class). Although there is no predictability power in ZeroR, it is useful for determining a baseline performance as a benchmark for other classification methods.

**Accuracy: 65.01%**

| True: | Benign | Malignant |
|---|---|---|
| Benign: | 444 | 239 |
| Malignant: | 0 | 0 |

Table 3.14 - ZeroR Confusion Matrix

**False Negative Cost: 1**
**Accuracy: 34.99%**

| True: | Benign | Malignant |
|---|---|---|
| Benign: | 0 | 0 |
| Malignant: | 444 | 239 |

Table 3.15 - ZeroR with MetaCost Confusion Matrix

# 3.10. Comparison

In order to compare the accuracy of the different classifications methods that were applied to the dataset, a simple pairwise t-test was performed. A t-test is a statistical hypothesis test in which the test statistic follows a Student's *t* distribution, if the null hypothesis is supported. It is most commonly applied when the test statistic would follow a normal distribution if the value of a scaling term in the test statistic were known. When the scaling term is unknown and is replaced by an estimate based on the data, the test statistic (under certain conditions) follows a Student's *t* distribution. This test determines the probability for the null hypothesis: “the actual means are the same”. Since a t-test can only be applied on two performance vectors, the test was applied to all possible pairs of classification methods.

The first step was to compare the results of the first iteration of classification methods, i.e., without the use of the MetaCost algorithm.

Analyzing Table 3.16, it is valid to conclude there is no statistical evidence to claim that each one of the classification methods used is more accurate in comparison to each one of the remaining methods. This statement can be proved since all the *p* values are greater than the significance level of 0.05 (alpha=0.05).

| | | Default Model | Naive Bayes | Decision Tree | LibSVM | k-NN | Neural Network | Rule Induction |
|---|---|---|---|---|---|---|---|---|
| | | 0.650 +/- 0.009 | 0.961 +/- 0.013 | 0.962 +/- 0.024 | 0.969 +/- 0.018 | 0.969 +/- 0.028 | 0.966 +/- 0.021 | 0.902 +/- 0.034 |
| **Default Model** | 0.650 +/- 0.009 | | **0.000** | **0.000** | **0.000** | **0.000** | **0.000** | **0.000** |
| **Naive Bayes** | 0.961 +/- 0.013 | | | 0.864 | 0.311 | 0.395 | 0.469 | **0.000** |
| **Decision Tree** | 0.962 +/- 0.024 | | | | 0.457 | 0.548 | 0.672 | **0.000** |
| **LibSVM** | 0.969 +/- 0.018 | | | | | 1.000 | 0.745 | **0.000** |
| **k-NN** | 0.969 +/- 0.028 | | | | | | 0.799 | **0.000** |
| **Neural Network** | 0.966 +/- 0.021 | | | | | | | **0.000** |
| **Rule Induction** | 0.902 +/- 0.034 | | | | | | | |

Table 3.16 - T-Test significance matrix

The same t-test was applied to the results obtained using the same classification methods, this time using the MetaCost algorithm with a manual optimization of the cost of false negatives of the confusion matrix. Bold values on Table 3.17 that are smaller than the level of significance (alpha=0.05) indicate a probably significant difference between the actual mean values. In this case, it is visible there is a difference between the accuracy of LibSVM [12] algorithm and the accuracy of the Naive Bayes classifier ($p$=0.032). Also, looking at the $p$ value regarding the comparison between k-NN and LibSVM ($p$=0.003), there is statistical evidence to claim that the accuracy of the latter is worst comparing to the accuracy of the first. Moreover, there is also statistical evidence showing that the Neural Network and Rule Induction () classifiers were more accurate than LibSVM, for the experiments made with the dataset. The $p$ values are $p$=0.001 and $p$=0.036, respectively.

| | | Default Model | Naive Bayes | Decision Tree | LibSVM | k-NN | Neural Network | Rule Induction |
|---|---|---|---|---|---|---|---|---|
| | | 0.350 +/- 0.009 | 0.952 +/- 0.023 | 0.951 +/- 0.035 | 0.922 +/- 0.031 | 0.968 +/- 0.026 | 0.968 +/- 0.017 | 0.950 +/- 0.021 |
| **Default Model** | 0.350 +/- 0.009 | | **0.000** | **0.000** | **0.000** | **0.000** | **0.000** | **0.000** |
| **Naive Bayes** | 0.952 +/- 0.023 | | | 0.990 | **0.032** | 0.198 | 0.111 | 0.881 |
| **Decision Tree** | 0.951 +/- 0.035 | | | | 0.077 | 0.344 | 0.268 | 0.919 |
| **LibSVM** | 0.922 +/- 0.031 | | | | | **0.003** | **0.001** | **0.036** |
| **k-NN** | 0.968 +/- 0.026 | | | | | | 0.993 | 0.135 |
| **Neural Network** | 0.968 +/- 0.017 | | | | | | | 0.064 |
| **Rule Induction** | 0.950 +/- 0.021 | | | | | | | |

Table 3.17 - T-Test Significance using Metacost algorithm

# 4. Conclusions

FNA is a procedure in which the analysis is usually performed by human experts who make the diagnosis based on the observation of the microscope images. This work shows that this task can be performed autonomously with high accuracy by using common machine learning algorithms. Taking into account the different cost of misclassification for the different classes is a requirement of the problem. This is obvious since a false negative would put a human life at risk. With this in mind, it is fair to affirm the best results were obtained by the Support Vector Machine classifier. The number of false negatives was reduced to zero, although this caused the number of false positives to increase. However, the increase of the false positives is an acceptable price to pay in this specific context. The use of the MetaCost algorithm reduced significantly the number of false “Benign” errors in all classification methods used. In spite of causing a loss of accuracy, it is an useful algorithm to apply whenever the costs of the different classification errors are not the equal.

Algorithms such as decision tree and rule-based induction provide experts with a model that shows useful information in a simple to understand form, in this case, that information, the rules generated by rule-based induction or the decision tree created by the decision tree algorithm, should be obvious to people who perform the FNA diagnosis.

Applying optimization to algorithm parameters, should, specially in the case of complex multi-parameter algorithms such as SVM further increase their accuracy, though, it is also likely that the results are very near the limit of what can be achieved with this dataset. In that case, a new dataset, consisting of different features extracted from the microscopic images with more information could provide better results.

# 5. Acknowledgements

The authors would like to acknowledge the contributions of the course teacher, José Borges, Matko Bošnjak (Sapo Labs) and André Martins (INESC).

# 6. References


[1] - L. Mangasarian and W. H. Wolberg: "Cancer diagnosis via linear programming", SIAM News, Volume 23, Number 5, September 1990, pp 1 & 18.

[2] - Zhang, J. (1992). Selecting typical instances in instance-based learning. In Proceedings of the Ninth International Machine Learning Conference (pp. 470--479). Aberdeen, Scotland: Morgan Kaufmann.

[3] - Breast Cancer Wisconsin (Original) Data Set, http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Original) last accessed on April 18th 2011.

[4] - RapidMiner, http://rapid-i.com/ last accessed on April 18th 2011.

[5] - Data Mining Concepts and Techniques

[6] - MetaCost: A General Method for Making Classifiers Cost-Sensitive, Pedro Domingos, In Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining, 1999, 155--164, ACM Press

[7] - Machine Learning, Tom Mitchell, McGraw Hill, 1997

[8] - Introduction to Machine Learning, Ethem Aalpadyn, The MIT Press, 2004

[9] - CRISP-DM, http://www.crisp-dm.org last accessed April 18th 2011

[10] - C4. 5: programs for machine learning, Quinlan, J.R., isbn: 1558602380, 1993, Morgan Kaufmann

[11] - W. W. Cohen. Fast effective rule induction. In *Proc. of Twelfth International Conference on Machine Learning*, Lake Tahoe, California, 1995.

[12] - LIBSVM: a library for support vector machines, Chang, C.C. and Lin, C.J., 2001, Citeseer