Machine Learning Models for Imbalanced Quality of Life Classification for Cancer Patients

Marija Kopanja[a*], Miloš Savić[b**]
[a]Faculty of Sciences, University of Novi Sad, Serbia
[b]Faculty of Sciences, University of Novi Sad, Serbia

[*]Author: Marija, Kopanja; marija.kopanja17@gmail.com
[**]Mentor: Miloš, Savić; svc@dmi.uns.ac.rs

Abstract

Quality of life (QoL) is one of the major issues for cancer patients. Large amounts of data in medical databases with relevant QoL information, enable generation of predictive QoL models by using various machine learning techniques. However, the process of creating predictive model for binary QoL indicator from aspect of classification is challenging due to class imbalance problem. In many real world application domains, especially in medicine, classes are not represented equally. This problem, known as class imbalance, can disrupt procedure of generating machine learning model and lead to classifier that will be biased in favor of the majority class, thereby failing to identify and correctly predict instances from the minority class which are typically of the main interest. This research work investigates modifications of standard machine learning methods in tackling class imbalance problem for QoL classifiers within binary classification framework. The standard machine learning methods for classification: Naive Bayes, Support Vector Classifier, Decision Trees, Random Forest and K-Nearest Neighbors, from reference study are explored across several QoL datasets for breast cancer patients. For each baseline model obtained in the study, here is proposed modified version on the same data that should better handle skewed data distribution. Furthermore, for some methods grid search method is applied in order to tune hyperparameters. With aim to have comprehensive view about learning performance of each classifier, several evaluation measures were used. The results of our experimental evaluation indicate that each baseline model is outperformed by proposed alternative model(s) in terms of $F1$ score and support vector machine classifier with class weights emerge as best performing model.

This research paper is produced as seminar paper on PhD studies. Future aim is directed toward extending the work in order to get it published.

Machine Learning Models for Imbalanced Quality of Life Classification for Cancer Patients

Marija Kopanja
Faculty of Sciences, University of Novi Sad, Serbia
marija.kopanja17@gmail.com

ABSTRACT

Quality of life (QoL) is one of the major issues for cancer patients. Large amounts of data in medical databases with relevant QoL information, enable generation of predictive QoL models by using various machine learning techniques. However, the process of creating predictive model for binary QoL indicator from aspect of classification is challenging due to class imbalance problem. In many real world application domains, especially in medicine, classes are not represented equally. This problem, known as class imbalance, can disrupt procedure of generating machine learning model and lead to classifier that will be biased in favor of the majority class, thereby failing to identify and correctly predict instances from the minority class which are typically of the main interest. This research work investigates modifications of standard machine learning methods in tackling class imbalance problem for QoL classifiers within binary classification framework. The standard machine learning methods for classification: Naive Bayes, Support Vector Classifier, Decision Trees, Random Forest and K-Nearest Neighbors, from reference study are explored across several QoL datasets for breast cancer patients. For each baseline model obtained in the study, here is proposed modified version on the same data that should better handle skewed data distribution. Furthermore, for some methods grid search method is applied in order to tune hyperparameters. With aim to have comprehensive view about learning performance of each classifier, several evaluation measures were used. The results of our experimental evaluation indicate that each baseline model is outperformed by proposed alternative model(s) in terms of $F1$ score and support vector machine classifier with class weights emerge as best performing model.

KEYWORDS

Class imbalance, Quality of Life, Cancer Patients, Naive Bayes, Support Vector Machine, K-Nearest Neighbors, Decision Trees, Cost-sensitive Decision Trees, Random Forest, Cost-sensitive Random Forest

1 INTRODUCTION

Developments in machine learning and other areas of artificial intelligence in recent years have stimulated an increase in research on their applications within medicine. Medical records can be leveraged for creation of valuable machine learning models for doctors, specialists and patients as well. Applications of machine learning algorithms within medicine include among others postoperative prognostication, diagnostics, preoperative planning and risk stratification. Developed models can assist in the clinical decision-making process regarding postoperative treatments in order to utilize more efficiently resources, prioritize and personalize patient's care and provide improvement of QoL issues.The issues depend on type of disease and patient's status i.e. is patient in beginning disease phase, recovering from surgery, goes through some treatment process etc. In this research we will concentrate on the QoL issues for breast cancer from the study [30]. More precisely, on Depression, Anxiety, Sleep quality (Insomnia) and Pain. The study was part of ASCAPE project (https://ascape-project.eu/) where is considered QoL for breast and prostate cancer (one of the most prevalent types of cancer). The aim of the project was to employ machine learning and other artificial intelligence mechanisms to the patient's QoL data in order to enable personalised follow-up strategy for cancer patients focusing on their QoL issues [30].

In general, assessment of patient's QoL issues is based on results of different questionnaires. Patient Reported Outcome Measures (PROMs) are widely used high-quality questionnaires for checking patients' perceptions about their health and about experiences after a treatment/intervention. Generally, use of PROMs in research to gain insights relating to the medical treatments outcomes, follow-ups or post-operative prognostication, in order to evaluate the impact of treatment or QoL is well-established [9]. Such questionnaires are good instruments to minimize the risk for measurement bias and enable better QoL predictions.

In a large number of important domains, especially medicine, class imbalance problem is pervasive [24]. The class imbalance problem characterized by uneven number of examples in one class compared to other class(es), has been recognized as one of the most important problems in machine learning community [5]. From two-class classification task perspective, the class with larger number of examples is called majority class and the class with smaller number of samples is called minority class. By convention, the minority class in imbalanced data is considered as the positive class whilst the majority class is considered as negative class. Observed from binary imbalance classification framework QoL is binary indicator for some symptom, where positive class refers to patients that experience the symptom after breast cancer diagnosis, whilst patients that do not experience the symptom represent instances from negative class. When dealing with class imbalance, wrongly classifying minority class is much more undesired then wrongly classifying majority class examples, although the latter is also undesired only to lesser extent. In other words, minority class instances (patients that experience given symptom) have higher misclassification cost relative to the misclassification cost of majority class (patents that do not experience the symptom).

Supervised by Miloš Savić.

It is recognized that performance of classifiers attainable by most standard learning algorithms has significant drawback due to imbalanced class distribution [32]. Their poor performance is explained by algorithm designed to optimize measure such as accuracy or error rate, which are inappropriate when dealing with class imbalance, and assumption made by these learning algorithms, about relatively balanced class distribution thereby implicitly assuming equal misclassification costs. Most popular machine learning algorithms in classification context such as Decision Trees, Support Vector Machine, Naive Bayes and K-Nearest Neighbors are reported to be inadequate when encountering the class imbalance problem [32]. These supervised learning methods have been used in the study [30] to train models serving as baseline models in our research. It must be acknowledged that there is need for improvement of methods in order to generated models be able to better handle imbalanced data, since correctly predicting minority examples is primary concern in classifying imbalanced data [16]. Furthermore, since examples from majority class occur more frequently, discovered patterns and rules referred to minority class are undiscovered, leading to misclassification of minority samples more often. Therefore, having classifier that mostly incorrectly predicts minority class examples is rather useless as it fails to identify the examples which are of the main interest.

Many different solutions for class imbalance problem have been developed both on data and algorithm level, where the former assumes rebalancing data distribution and the latter assumes adaptation of existing classification learning algorithms to improve performance with regards to the minority class. Accordingly, methods that have been developed to cope with class imbalance problem can be categorized into algorithm level methods, data level methods and hybrid methods, as combination of the two.

Data driven methods rebalance the class distribution either randomly or deterministically. These methods are convenient and effective way to deal with class imbalance problems using standard machine learning algorithms [5]. The idea of data approach methodology is to manipulate the class distribution by using appropriate techniques in order to have more balanced distribution. Modification of data distribution is justified, as is observed in [13] that naturally occurring distribution is not always optimal. Algorithmic level approach as an alternative to data modification (or sometime combined with resampling techniques) is more complex task, as it requires knowledge about method functionality and internal structure of algorithm. Generally, it is difficult task to modify the learning scheme of an algorithm to make it favoring the minority class. This can be done by modifying the optimization function, by adjusting the decision threshold or by providing misclassification costs for each class to the learning algorithm.

One family of methods that can be used for tackling the problem are cost-sensitive learning methods, which incorporate unequal misclassification cost into the learning process. The misclassification cost plays indispensable role in biasing classifier towards rare class. The cost-sensitive learning is type of inductive, supervised learning that has attracted significant attention in recent years, as it plays important role in many data mining applications. This type of learning, beside misclassification costs, can take other types of cost into the consideration, such as test cost and other cost discussed in [26]. Cost should be thought on in most broadest sense, not just as entity that can be measured in monetary units. Potential difficulty in using cost-sensitive methods is the demand for cost-sensitive information i.e. the misclassification costs, which are typically unknown and determination of the costs requires domain knowledge. However, one possible solution is to observe the misclassification costs as hyperparameters and search for optimal cost values using some optimization technique.

Contrary to aspect in [13], where is shown that depending on the used evaluation criteria, the classifier should not be generated using natural distribution, our goal is to use improve results given in [30] without data manipulation by using modified version of each machine learning method that should be better in tackling challenging class imbalance problem. In this way, we are also avoiding data distribution modification procedure which has limits and costs associated with attempt to make distribution more balanced. The aim of our research was to provide possible extensions on algorithm level of used methods in creating benchmark classifiers that are not good in coping with explained problem present in datasets for QoL breast cancer patients.

Typically the hyperparameters of a model are tuned in order to, if it's possible, further improve performance of the model. The grid search is one of the techniques that can be used for this purpose. The choice for best parameters in grid search methods is driven by extensive search through hyperparameter space. Hence in terms of efficiency, better choice could be random search technique.

Performance of all obtained classifiers is compared with results from the study [30], by using different evaluation measures. In this framework proper measures should be chosen with caution, since evaluating classification performance in the presence of the class imbalance problem could lead to making misleading conclusions with potentially serious consequences, and that must be avoided.

The rest of the paper is organized as follows. Next section takes on the overview on the ML methods developed in the research community in order to alleviate the class imbalance problem in classification framework with focus on those methods used in the study [30] as well as other studies related to QoL models. Section 3 explains the problems that arise due to class imbalance and hinder the performance of standard machine learning algorithms. For each of the described methods is presented improved version(s) for addressing the problem associated with skewed class distribution with explanation how modification influences classifier performance. Further, Section 4 describes experimental setup and different measures used in the study [30]. Discussion on used data, obtained results and comparative analysis is given in Section 5, followed by conclusion and direction for future work in last section.

2 RELATED WORK

Abundance of research articles indicate that the class imbalance problem is an active research field in machine learning. The studies differ on the aspects of class imbalance problem. Some of them are addressed on solutions for tackling class imbalance problem from data and other from algorithm perspective. Moreover, some studies direct their effort to the nature i.e. domain of the class imbalance, for example medicine. Number of research papers devoted to class imbalance problem in medical data testify about importance of this problem encountered in the field. Recently monitoring of QoL

parameters has attracted extensive research interest. Particularly in prevention and early detection of symptoms and signs for chronic diseases. The importance of proper monitoring can be seen in providing positive effect on patient's quality of life, economic impact and resource management. To our knowledge there is no research papers devoted to tackling class imbalance problem in QoL data.

Many researches [7] have tried to alleviate the class imbalance problem by using sampling techniques. Mostly, undersampling and oversampling, where the former refer to randomly/deterministically removing examples from the majority class in order to make the majority class more even with minority class, and the latter refer to randomly/deterministically adding examples from minority class with the same objective. Both techniques have their drawbacks, undersampling violate data integrity as some examples are removed and lead to loss of possibly useful information contained in those samples, and oversampling contrarily increase dataset size thereby increasing execution time and leading to overfitting problem. Other sampling techniques are based on these two, such as SMOTE (Synthetic Minority Oversampling Technique) which employs oversampling. However, in oversampling phase, examples from minority class are not copies of existing examples, rather they are created using K-nearest neighbors algorithm. A downside of the SMOTE is that synthetic examples are created without considering the majority class, resulting in possibly ambiguous examples in case the classes overlap.

Larger number of research papers related to data level methods as solution in tackling class imbalance problem might be due to potentially broader applicability of these methods as they do not require deeper knowledge in the chosen algorithm, rather make data more balanced in preprocessing step. As our goal is to provide solution for each machine learning method for tackling the imbalance on algorithm level, further discussion is concentrated on different methods internally modified to cope with class imbalance as well as on mentioned baseline methods used in other research papers in the same imbalance problem framework.

Support vector machine (SVM) is known as very successful classifier in data mining tasks. However, in class imbalance context, depending on the application domain, the results of conducted research are ambiguous. In [22], SVM outperformed Naive Bayes and Decision Tree classifiers in terms of recall on all previously preprocessed datasets with imbalance ratios from slightly to extreme imbalance. Furthermore, on large datasets the SVM performs good when G-mean was utilized as evaluation measure. However, as stated in [22] according to some other studies, SVM is not good classifier of choice on highly imbalanced data as it classifies almost all examples in majority class, thereby failing to identify examples from minority class. Nevertheless, these performance comparisons are made on modified data using some sampling technique. In [5] is proposed different framework, without data manipulation. They used so-called cost-sensitive SVM, which with optimized hyperparameters outperforms traditional SVM. The SVM is particularly well suited for classification task on complex data of small or medium size [14].

In [12] is compared performance of Multinomial Naive Bayes and Complement Naive Bayes in text classification domain, across different data size and different parameter settings. Their results indicate that there is no significant difference between performances of obtained classifiers, measured by accuracy, precision and recall using 10-fold cross-validation. According to [10], CNB better performance over MNB is attributed to improved performance over minority class. Performance of all Naive Bayes classifiers highly depends on the probabilities estimated from the training data [16]. Furthermore, the probabilities estimated from minority examples are not reliable due to small number of positive examples. This could explain outperformance of other learning algorithms over Naive Bayes classifiers in learning from imbalanced data. Naive Bayes together with tree classifier are used to predict changes (5% increase/decrease) in the QoL scores of hemodialysis patients in [29] in order to help in directing resources toward the high-risk population group, with accent on patients with the highest risk of having a drop in QoL scores in the coming month, that is shown to have positive impact on ensuring better QoL for patients as well as on reduction of financial burden.

The class imbalance problem and decision tree learning is researched in several papers. However, large number of studies is restricted on C4.5 classifier [13, 24]. In [8] is proposed weighted decision tree method based on entropy measure, where is shown variation of results by using different weights. Cost-sensitive decision tree learning method is applied within example-dependent classification framework in [1], showing better performance over standard decision tree model across several datasets. Decision trees methods are also applied in [31] with ensemble methods in prediction of 5-year lung cancer survival on the basis of QoL issues.

In [19] QoL indicator was assessed using stacked ensemble models that combines several different machine learning approaches including decision tree, random forest and support vector machine. The results showed that the ensemble model based on the stacked generalization framework is a significantly better predictor of the life satisfaction of a nation, compared to base models. Other ensemble methods like Random Forest and Bagging are successfully applied in prediction of 5-year lung cancer survival on the basis of QoL issues [31]. In general, ensemble learning is widely used supervised learning approach due to its idea of building a prediction model by combining the strengths of a collection of simpler models (base learners). Random forest generally exhibits a substantial performance improvement over the single tree classifier such as CART and C4.5 [20].

There are several attempts to make KNN method better in tackling class imbalance problem. For example, in [21] class confidence weights are introduced in order to enable correction of inherent bias in the method. Another paper [18] proposes modified KNN algorithm that attempts to provide more importance to neighbors with a higher proximity weighted confidence.

A number of techniques were shown to be effective if applied in a certain context [24]. When dealing with challenging problem like class imbalance there is demand for additional research for possible new solutions, on both data and algorithm level. If the latter is the choice, there is necessity for much deeper knowledge of internal algorithm structure and its functioning. For that reason, next section briefly explains used methods.

3 METHODOLOGY

Two types of QoL indicator can be distinguished [30]. One type are binary indicators which indicate whether the patient will experience QoL symptom like depression, anxiety, insomnia or pain after diagnosis. For this type of the indicator, most appropriate machine learning models are binary classifiers.

Within binary classification framework, the QoL indicator is binary target variable with 0 - patient does not express the symptom and 1 - patient expresses the symptom. This notation is based on convention that the class with larger number of examples (majority class) is considered as negative class, whilst the class with smaller number of samples (minority class) is considered as the positive class.

In next subsections, we will give brief explanation of the used methodological approach. We introduce standard machine learning techniques used in the benchmark research as well as our proposed modified version of the techniques intended to overcame the class imbalance problem. Additionally, for some methods is proposed more than one possible modified version with explanation way and how it should help in improving classifier performance on data where one class is much larger in size.

Before we start with explaining methods, let's first introduce notation. Given a data represented as collection of examples $\{x_i, y_i\}_{i=1}^{n}$ where x_i is vector of features with length p, $x_i = (x_{i1}, \ldots, x_{ip})$ and $y_i \in \{0, 1\}$ is a label associated with the ith example, representing the class of the example, the task of classification is to learn a function $f : X \rightarrow Y$, where $X \subseteq \mathbb{R}^p$ and $Y \subseteq \mathbb{R}$, that will have good generalization ability. That is, this function should have smallest possible error on new examples generated from the same underlaying distribution as the training data.

3.1 Naive Bayes

Bayesian methods is family of methods that learn a probabilistic model based on Bayes' theorem and (naive) assumption of independence among features (predictors) given the value of target class [7]. Here we omit using subscript i in $x_i = (x_{i1}, \ldots, x_{ip})$ for sake of simplification and compliance with definitions found in literature. Given a feature vector $x = (x_1, \ldots, x_p)$ and target value y, the algorithm finds the probability for each possible value of target variable and classify sample in class with maximum probability. Mathematically expressed:

$$\hat{y} = \underset{y}{\operatorname{argmax}} p(y) \prod_{i=j}^{p} p(x_j|y) \tag{1}$$

where $\hat{y} = p(y|x)$ is posterior probability, $p(y)$ is prior probability and $p(x_j|y)$ is conditional probability (likelihood). This is suggesting that instead of direct calculation of $p(y|x)$, we can use estimation of the prior probability, which is simply the fraction of examples in positive/negative class and estimation of conditional probability, that includes assumption about the distribution. Different Naive Bayes classifiers are distinguished based on assumptions they make regard this distribution. However, all classifiers predict class for an example with largest posterior probability.

3.1.1 ***Multinomial Naive Bayes.*** Multinomial Naive Bayes (MNB) is method that implements Naive Bayes idea with assumption that data have multinomial distribution. MNB is commonly used for text classification problems. For that reason, we will describe this method in text classification context, where the features represent words, examples are documents and data values are frequencies of words in corresponding document. That is, if ith example is represented as $x_i = (x_{i1}, \ldots, x_{ip})$, x_{ij} denotes frequency of word w_j, $j = 1, \ldots, p$. A frequency is in literature typically denoted as f_j, which is representation of data know as *bag of words*. The likelihood of an example is given by a multinomial distribution over the set of words:

$$p(x|y) = \frac{(\sum_j f_j)!}{\prod_j f_j!} \prod_j (p(w_j|y))^{f_j} \tag{2}$$

where as mentioned f_j denote the frequency count of word w_j in the document x, $(\sum_j f_j)$ represent the length of the document and $p(w_j|y)$ is class-conditioned probability of word occurrence in class y, for which is often assumed that is independent of document length [23]. Since the fraction is constant value regardless of class value y, the likelihood of an example is product of the probabilities of features that characterize that example. Therefore, for MNB we obtain following:

$$p(y|x) \infty p(y) \prod_j (p(w_j|y))^{f_j} \tag{3}$$

The priors are estimated as follows:

$$p(y) = \frac{N_y}{N}, \tag{4}$$

where N_y denotes number of examples in class y and N is total number of examples, while the conditional probabilities are estimated as:

$$p(w_j|y) = \frac{N_{yj}}{\sum_j N_{yj}} \tag{5}$$

where N_{yj} is frequency of word w_j in training examples belonging to the class y and the denominator is the total number of words in training examples belonging to the class y.

When classifying new example, it is possible that some word does not occur in training data. In this case, the probability estimate will be zero since it is proportional to frequency count of the word, leading to loss of information for all other word's probabilities since they are multiplied. Hence, in order to avoid this inconvenient situation, small smoothing constants are added to all probability estimates:

$$p(w_j|y) = \frac{N_{yj} + \alpha}{\sum_j N_{yj} + p\alpha} \tag{6}$$

where p represents the size of vocabulary i.e. the number of features for observed data. This type of regularization is known as *Lidstone smoothing* with special case when $\alpha = 1$ known as *Laplace smoothing*. Notice, adding 1 in numerator corresponds to assigning each word a frequency of one instead of zero and requires adding p in denominator in order to obtain probability distribution that sums to one.

The feature values do not need to be term frequencies, instead they can only indicate presence (value 1) or absence (value 0) of

the word in a document. In more general scenario, not only in text classification context, the categorical features [1] can be transformed to represent presence or absence of particular feature value for given example. With this transformed data we can still use MNB, as now we can assume that data have multivariate Bernoulli distributions, the special case of multinomial distribution.

3.1.2 ***Complement Naive Bayes.*** Complement Naive Bayes (CNB) is modified version of MNB, formed as solution for MNB potential deficiency in dealing with imbalanced data. Namely, when used in application domains characterized with skewed class distribution, MNB smoothing parameter biases predictions in favor of majority class since initial word frequencies have larger impact on the posterior probability when there is less data. [28] Unlike MNB where are used examples from a single class y, CNB uses examples from all classes except y. The likelihood probabilities are estimated as:

$$p(w_j|y) = \frac{N_{\overline{y}j} + \alpha}{\sum_j N_{\overline{y}j} + p\alpha} \tag{7}$$

where $N_{\overline{y}j}$ is number of times word w_j occurs in examples form classes other than y and sum in denominator is total number of all words occurrences in classes other than y. The formula for CNB is inverse function of MNB:

$$p(y|x) \infty p(y) \prod_j \frac{1}{p(w_j|\overline{y})^{f_j}} \tag{8}$$

Here we are calculating complement probability i.e. for each class $y \in \{0, 1\}$ we are calculating the probability that an example belongs to classes other than y. In other words, the method calculates the probability that an example not belongs to class y. Hence an example will be classified in class y with lowest probability (lowest probability that is not in that class, or equivalently, highest probability that is in that class).

The CNB is further augmented to alleviate one more deficiency of MNB. The independence assumption can erroneously cause production of likelihood probabilities with different magnitude. Class with strong dependencies among features will have higher likelihood, hence may be preferred relative to other class. This means that Naive Bayes method give more influence to classes that most violate the independence assumption [10]. The idea in CNB is to normalize the probabilities in order to correct for the fact that some classes have higher dependencies. If $w_{yj} = log(p(w_j|y))$, we assign:

$$w_{yj} = \frac{w_{yj}}{\sum_j w_{\overline{y}j}} \tag{9}$$

There are three other transformations that boost performance of CNB further. More information about these transformations can be found in [10]. Additionally CNB performance can be further improved by optimizing the smoothing parameter.

3.2 Support Vector Machine

Support vector machine (SVM) is group of methods based on idea of maximum margin hyperplane between two linearly separable classes. The concept of margin indicates the distance between the hyperplane and examples from both classes which the hyperplane separates. These examples are called support vectors. The aim of method is to search for the optimal hyperplane that maximizes the margin and simultaneously minimizes the classification errors. For simplicity of method description, target variable will be denoted as $y_i \in \{-1, 1\}$. Each example $x_i = (x_{i1}, \ldots, x_{ip})$ is classified based on position relative to the hyperplane which divides the feature space into two regions opposite in sign.

The examples are linearly separable if there exist vector w and scalar w_0 i.e. hyperplane (w_0, w) such that: $w_0 + wx_i \geq 1$ if $y_i = 1$ and otherwise $w_0 + wx_i \leq -1$ if $y_i = -1$. These two inequalities define the region, between two boarder hyperplanes, which is called *margin*. Notice, the term $w_0 + w_i x_i$ is the distance of the example x_i from the (central) hyperplane. Hence, the inequalities imply that the hyperplane is equidistant from the examples in opposite classes. More precisely, the two inequalities indicate that the distance of the example x_i from the hyperplane is at least one unit. Since the two boarder hyperplanes can always be scaled to unit distance from the hyperplane ([15]), further explanation continues with introduced 2 units wide margin. The former and latter inequality imply that every example above/on the marginal boundary will be classified as positive and every example below/on the marginal boundary will be classified as negative, respectively.

In case separable hyperplane can be found, it is normally not unique. Hence the algorithm searches for hyperplane which has maximal distance from examples in different classes. In other words, in learning phase, the algorithm finds a hyperplane by solving next optimization problem:

$$\begin{aligned} &\min \frac{1}{2} w w^T \\ &\text{s.t.} \quad y_i(w_0 + wx_i) \geq 1 \end{aligned} \tag{10}$$

Therefore the optimal hyperplane is unique one which maximize the margin and satisfies above constraints. Constructing the hyperplane is an optimization problem that can be solved by creating its dual form i.e. by constructing Lagrangian where $\alpha = (\alpha_1, \ldots, \alpha_n)$ is vector of non-negative Lagrange multipliers corresponding to the constraints. It can be shown [11] that the dual problem has the form:

$$\begin{aligned} &\max_\alpha \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i x_j \\ &\text{s.t.} \quad \sum_{i=1}^{n} \alpha_i y_i = 0, \quad \alpha_i \geq 0 \end{aligned} \tag{11}$$

meaning the problem is transformed into quadratic programming problem. According to the Kuhn-Tacker theorem [3], the solutions α^* must satisfy conditions: $\alpha_i(y_i(w_0 + wx) - 1) = 0$. This implies that non-zero α_i i.e. $\alpha_i \neq 0$ are only achieved if $y_i(w_0 + wx) - 1 = 0$. The vectors x_i for which $y_i(w_0 + wx) = 1$ are called *support vectors* and correspond to samples that are closest to the separating hyperplane. Furthermore, it can be shown that the optimal hyperplane solution can be written as linear combination of training examples where only support vectors have effective contribution to the sum. Then, the decision function can be expressed so it depends only on the dot product between examples. Moreover, as the $\alpha_i \neq 0$ only for support vectors, classifying new example requires fewer dot product calculations.

[1] Numerical features first should be discretized in order to enable one-hot-encoding technique to be applied.

Described method can be further expanded by enlarging the feature space using special functions, called *kernels*. The main idea of this extension is to enable linear separability of linearly inseparable data in new higher dimensional feature space. More precisely, if no linear separation is possible, a non-linear mapping into a higher dimensional feature space is realized and the hyperplane found in this higher dimensional space corresponds to a non-linear decision boundary in the original feature space. [11]

If g represents mapping function from the original feature space to the higher dimensional feature space, the same learning procedure can be applied and generate nonlinear boundary. Important concept of SVM is that there exist kernel function K in original feature space that corresponds to the dot product $g(x_i)g(x_j)$ in transformed feature space, i.e. $K(x_i, x_j) = g(x_i)g(x_j)$. Notice, for above explained initial methodology the kernel is matrix of dot products with elements: $K_{ij} = K(x_i, x_j) = x_i x_j$, i.e. the kernel is linear. Accordingly, the optimization task can be rewritten as:

$$\begin{aligned} \max_{\alpha} \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \\ \text{s.t.} \quad \sum_{i=1}^{n} \alpha_i y_i = 0, \quad \alpha_i \geq 0 \end{aligned} \tag{12}$$

Therefore we still have the quadratic convex programming problem, being particularly easy to handle. More importantly, by means of kernel function the intermediate step of creating the higher dimensional feature space is omitted and it is working only in the original space where $K(x_i, x_j)$ is defined. Beside linear, polynomial, Gaussian, Radial basis function, other kernel functions can be used [22]. Each kernel function is characterized by parameters that should be tuned for particular problem. In our implementation, we restricted out attention on RBF function: $K(x_i, x_j) = \exp(-\gamma ||x_i, x_j||^2), \gamma > 0$, as it requires only optimization of parameter γ.

Due to kernel function SVC can alleviate the problem of not linearly separable data. However, the kernel function introduces the problem of overfitting, as the method can define complex boundaries for the training set, hence not having the predictive power for unobserved samples. In other words, transformation into higher dimensional feature space decreases generalization ability of the SVM.

As this method is very sensitive on position of individual examples and boundaries can be quite complex, an improvement is introduced in the view of additional parameter that enables control over complexity, leading to so called *Soft margin hyperplane*. More precisely, we allow that some examples can be on the wrong side of the boundary hyperplane by introducing slack variables $\xi_i \geq 0$ and $\sum_i \xi_i \leq$ constant. As the ξ_i is the amount by which the prediction is on the wrong side of its marginal boundary, by bounding $\sum_i \xi_i$ we bound number of misclassifications. The optimization problem is changed by adding regularization parameter C:

$$\begin{aligned} \min \frac{1}{2} w w^T + C \sum_{i=1}^{p} \xi_i \\ \text{s.t.} \quad y_i(w_0 + w g(x_i)) \geq 1 - \xi_i, \\ \xi_i \geq 0, \quad i = 1, \ldots, p. \end{aligned} \tag{13}$$

where C replaces the constant from the bounding slack variables. The dual problem obtained using Lagrangian is the same as (12) with the difference in the constraints: $0 \leq \alpha_i \leq C, i = 1, \ldots, p$. More extensive and detailed explanation can be found in [3, 27]. There is described more general formulation with explanation of its special, simplified case presented here with unique solution - Soft margin hyperplane - that can be found efficiently. In case classes can not be perfectly separated, by introducing ξ_i deviations are allowed i.e. some samples can fall in region between two equidistant boarder hyperplanes or can be on the wrong side relative to the corresponding hyperplane. For example, those samples for which $\xi_i = 0$ are on the boarder hyperplane of its correct class i.e. on the boarder of the margin region, those with $\xi_i = 1$ on the separating hyperplane, and those with $\xi_i > 1$ are on the wrong side of the boarder hyperplane i.e. those are misclassified samples. Notice that the training errors occur only when $\xi_i > 1$, and other samples that are on or between margin region are support vectors.

The modification of the optimization problem allows obtaining simpler boundaries but at cost of making some errors. The objective function describes the problem of constructing a separating hyperplane which minimizes the sum of deviations of training errors and maximizes the margin for the correctly classified samples. The penalty parameter C regulates the trade-off between training error minimization and margin maximization [22]. The strength of the regularization is inversely proportional to C. Higher values of C indicate lower regularization strength and reverse otherwise. The higher this parameter is, the more significant missclassifications are and more complex the model will be. If parameter C takes very high value, overfitting may occur as the method allows smaller deviations i.e. has lower tolerance of errors and searches for more complex boundary. On the other hand, if C takes very small value, underfitting problem may arise since higher deviations from the margin are allowed and the focus is on margin maximization rather then minimizing the distance of misclassified samples from the boundary.

3.2.1 ***Weighted SVM.*** As explained, the SVM is driven by error-rate measure. However, this measure is not appropriate choice when dealing with imbalanced class distribution. In case number of samples within different classes are imbalanced, it is proposed [6] to use different regularization parameters for each class. The optimization problem formulation becomes:

$$\begin{aligned} \min \frac{1}{2} w w^T + C^+ \sum_{y_i=1} \xi_i + C^- \sum_{y_i=-1} \xi_i \\ \text{s.t.} \quad y_i(w_0 + w g(x_i)) \geq 1 - \xi_i, \\ \xi_i \geq 0, \quad i = 1, \ldots, p. \end{aligned} \tag{14}$$

where the C^+ and C^- are penalty parameters for positive and negative class, respectively. This modification introduces possibility to manage tolerance of different types of errors. That is, by setting higher penalty value for positive class then for negative class, algorithm will have smaller tolerance on misclassifying minority class then majority class. Based on this interpretation, this SVM can be seen as a cost-sensitive version of standard SVM, where C^+ and C^- are misclassification cost parameters. Therefore, by setting higher misclassification cost for positive class, the hyperplane can

be shifted away from minority class examples. Beside these regularization parameters, the kernel function has great impact on performance of classifier. Depending on the imbalance ratio and dataset domain, will differ most appropriate kernel, as is shown in [22]. Therefore, the choice of regularization parameters and kernel functions are two main choices in cost-sensitive SVM applications.

3.3 Decision Trees

Decision tree (DT) learning is process of generating classification model which can be represented with decision tree, a flowchart-like tree structure composed with three different types of nodes connected with edges. Position of each node in the diagram structure, determines the name of the node. Namely, the topmost node having only outgoing edges is known as a root node. A node that has one incoming edge and outgoing edge(s) is called internal node. A terminal node (leaf node) is a node that has only one incoming edge. This hierarchical structure is drawn upside-down, as the root node is on the top. The decision tree structure mimic human level thinking in a sense that the non-terminal nodes (root and other internal nodes) represent the questions, the edges represent possible answers and terminal nodes contain a class label in which the example will be classified. This means that classifying the example once the tree has been generated is straightforward. The decision tree model classify an example by sorting it down the tree, from the root node to some of the terminal nodes.

The learning phase consist of the feature (predictor) space stratification into a number of regions in a recursive manner. The regions are set of high-dimensional rectangles since the space is partitioned using hyperplanes that are parallel to the coordinate axes. In context of tree analogy, the regions are the nodes of the tree. Here our attention is restricted to binary partitions as that procedure is used in implemented decision tree methods.

Assume that the predictor space is divided into M distinct and non-overlaping regions R_m, $m = 1, \ldots, M$. The decision tree model will classify example x_i in the region R_m to the majority class $k(m)$:

$$\hat{y_i} = \hat{f}(x_i) = \sum_{m=1}^{M} k(m) I(x_i \in R_m) \tag{15}$$

where I is zero-one indicator variable and $k(m)$ is defined as:

$$k(m) = \underset{k}{\operatorname{argmax}} \hat{p}_{mk} \tag{16}$$

Here the probability distribution p_{mk} at each node m of the decision tree over the two classes $k \in \{0, 1\}$ is estimated as:

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k) \tag{17}$$

or shortly:

$$\hat{p}_{mk} = \frac{N_{mk}}{N_m} \tag{18}$$

where N_m denotes the number of examples in node m and N_{mk} denotes the number of examples from class k in node m. This means that p_{mk} is estimated as fraction of samples in mth node that belong to class k.

The top-down greedy search is used as it is computationally impossible to consider every possible partition of the feature space into M regions. The search begins at the top of the tree i.e. at the point where all examples belong to one region and then successively split the predictor space, creating two new branches for each indicated split. This process is greedy since at each step of the tree-building process, the best split is determined without consideration of future splits and their effect on results in future splits. To perform recursive binary splitting, the algorithm select feature j and split point s, that will split the predictor space into the two regions. For each feature $j = 1, \ldots, p$ and all possible cutpoints s for each feature, the pair of half-hyperplanes is defined as:

$$R_l(j, s) = \{x_i | x_{ij} \leq s\} \text{ and } R_r(j, s) = \{x_i | x_{ij} > s\} \tag{19}$$

The R_l and R_r represent the left and right region of feature space in which are examples with attribute j that takes on a value at least as s or bigger then s, respectively. This explanation refers to numerical features. Since all used decision tree based models can combine both numerical and categorical features, splitting for the latter amounts to assigning some of the qualitative values to one branch and remaining values to the other branch.

In order to select best feature in tree building process, an evaluation measure for the goodness of split should be provided to algorithm. There are different statistical measures suitable for classification task, defined in the terms of degree of the impurity. The degree impurity of a node is smaller, the more skewed the class distribution of the examples in the node is. In other words, the impurity of the node is smaller, the more the homogeneous the node is. One criterion that can be used to determine how well feature j splits the examples, in binary splitting framework, is the gain:

$$Gain(j) = Q_p - \left(\frac{N_{R_l}}{N_p} Q_l + \frac{N_{R_r}}{N_p} Q_r\right) \tag{20}$$

where N_p is the total number of examples in parent node[2], while N_{R_l} and N_{R_r} represent the number of samples associated with the left and right child node[3], respectively. The Q_p is the impurity measure of a parent node whilst Q_l and Q_r represent the impurity measures of the child nodes. Therefore, the sum represents the weighted average impurity measure of the child nodes. Decision tree learning algorithms typically choose a feature that maximizes the Gain (refgain). This is equivalently to minimizing the weighted average impurity of the child nodes, as the parent node has the same impurity for all features. There are different impurity measures among which most relevant include misclassification error, entropy and Gini index. Any of these measures can be used to evaluate the quality of particular split. Many different classification decision trees differ among measure they use. For example the Gini index defined as:

$$Q_m(T) = 2\hat{p}_{m1}(1 - \hat{p}_{m1}) \tag{21}$$

is used in CART method, abbreviated from *Classification and Regression Trees* [27], which is used in this research.

The process of recursive partitioning continues until some stopping criterion is met. For example, until all examples associated with the terminal node belong to the same class or until number of examples in a nodes is smaller than some predefined number. Finally, it is possible to stop growing the tree when maximum tree

[2]Parent node is a node in the tree from which an edge is descending.
[3]Child node is a node in the tree, except the root node, that have one incoming edge.

depth has been reached.[4] In our analysis minimal number of samples in terminal node and tree depth are used in hyperparameter tuning procedure.

After building the tree, a pruning procedure can be performed in order to improve generalization ability of the model, as the algorithm can generate too complex tree model that is susceptible to well-known phenomenon, overfitting. The pruning refers to the removing process of internal nodes. The post-pruning is pruning approach performed after the tree has been generated. One technique that employs this approach is *Cost-complexity pruning method*, also known as *Weakest link pruning method*, which is used in CART algorithm. The procedure successively collapse internal node (weakest link) that produces the smallest per-node increase in cost-complexity measure, where the cost-complexity measure of the tree is a function of the total sample weighted impurity of the terminal nodes and the number of terminal nodes in the tree $|T|$. The process of removing internal nodes continues until root tree is produced. This means that a finite sequence of subtrees is produced.

This pruning method is parametrized with cost-complexity parameter $\alpha > 0$. The penalty term plays a key role in controlling the model complexity, as it is used to adjust for the fact that larger trees (those with larger number of terminal nodes) tend to overfitting. More precisely, as α increases the penalty term also increases for tree with larger number of terminal nodes. Given a pre-selected α, the algorithm finds the subtree that minimizes the cost-complexity measure. One can show that for each α there is a unique smallest subtree that minimizes cost-complexity measure and sequence of subtrees obtained by pruning procedure, must contain the smallest subtree [17]. In practice, 5-fold or 10-fold cross-validation is used to estimate α and then full data set is used to obtain the subtree corresponding to α.

3.3.1 ***Weighted Decision Tree.*** It can easily be noted that Gini index takes on small values when all proportions p_{mk} are close to zero or one, meaning that node m contains samples which are predominantly from one class. In case the classes are imbalanced, the terminal nodes will most likely contain larger number of majority class examples, leading to larger misclassification error for minority class. One possible extension of this measure is to include class weights w_k in probability p_{mk} estimation as follows:

$$\hat{p}_{mk} = \frac{w_k N_{mk}}{w_m} \tag{22}$$

where w_m represents the total weight in node m:

$$w_m = \sum_k w_k N_{mk} \tag{23}$$

This means that the probability p_{mk} is estimated as share of weight of examples from class k in node m in total weight of the node m. This modification further augments the impurity measure used to split the nodes. Therefore, the Gini index becomes weighted Gini index. Furthermore, the Gain measure (20) is modified to account for unequal class weights:

$$Gain(j) = Q_p - (\frac{w_l}{w_p} Q_l + \frac{w_r}{w_p} Q_r) \tag{24}$$

where w_l and w_r are total weight in right and left node, respectively and w_p is total weight of the parent node, all calculated using (23). Meaning, each child node impurity is multiplied by the fraction of the parent node's total weight that is in each child node. By assigning different weight to each class, the impurity of the node with larger number of majority samples will be more impure due to higher weight for minority class. Furthermore the higher weight for minority class will lead to higher decrease of impurity when number of examples for minority class increases. Notice no-weighting is equivalent to assigning unit weight for each class. Therefore different weights reflect unequal importance of each class.

3.3.2 ***Cost-sensitive Decision Tree.*** In traditional decision tree learning algorithms, all examples are assumed to have equal importance. As mentioned, these algorithms assume a balanced class distribution and implicitly assume equal misclassification cost. The cost of misclassifying an example is a function of the actual and predicted class, represented as cost matrix that corresponds to the confusion matrix: In order to define cost matrix properly,

	predicted negative	predicted positive
actual negative	C_{TN}	C_{FP}
actual positive	C_{FN}	C_{TP}

Table 1: Cost matrix

two so-called "reasonableness" conditions must be satisfied [4]: $C_{FP} > C_{TN}$ and $C_{FN} > C_{TP}$, meaning that the cost of labeling an example incorrectly should always be greater than the cost of labeling it correctly. Violation of either condition means that one column dominates the other and optimal prediction is the class corresponding to dominated column. For binary classification task, the costs of making the error for the rare class are often higher than the costs for the majority class. This means, the cost of non-detecting QoL issue can be assumed to be much larger then cost of false alarm for a patient that does not expresses the issue predicted as he/she expresses it. That is $C_{FN} > C_{FP} > 0$. Naturally, the cost is zero for correctly predicted outcomes ($C_{TN} = C_{TP} = 0$).[5] [26]

There are two types of cost matrices, example-dependent cost matrices and class-dependent cost matrices, which assume that the costs are associated with examples and classes, respectively. Later assumption, that the misclassification costs are constant is stronger assumption, widespreaded through application of most cost-sensitive learning algorithms. This assumption is used in our implementation of Cost-sensitive Decision Tree (CSDT), hence the term "cost matrix" will refer to class dependent type, and class-dependent will not be mentioned explicitly. Cost-sensitive learning assumes that a cost-matrix is known. However, in many real-life problems the value for the cost matrix are unknown and not given by experts. Setting proper misclassification cost is challenging task, which can be accomplished by using cross-validation across several different combination of values for costs during hyperparameter tuning.

Given a cost matrix with zero cost for correct classification, the CSDT method unlike traditional decision tree, will classify example

[4]The depth of decision tree is the length of the longest path from a root node to a terminal node.

[5]In general, the costs of correct classification can be non-zero.

x_i in the region R_m to the least costly class k(m):

$$k(m) = \underset{k}{\text{argmin}} cost(f_k(m)) \tag{25}$$

where $f_k(m)$ is function that assigns class label k to all examples that belong to the node m and $cost(f_k(m))$ is misclassification cost at node m, calculated as:

$$cost(f_k(m)) = \sum_{x_i \in R_m} I(y_i \neq k)\mathbf{C}(y_i \neq k, k) \tag{26}$$

In our binary framework this can be written as:

$$cost(f_0(m)) = \sum_{\substack{x_i \in R_m \\ y_i=1}} C_{FN} \quad \text{and} \quad cost(f_1(m)) = \sum_{\substack{x_i \in R_m \\ y_i=0}} C_{FP} \tag{27}$$

meaning the cost of predicting negative label in the node m is equal to misclassification costs for positive examples in the node m wrongly classified as negative. Similarly, the cost of predicting positive label in the node m is equal to misclassification costs for negative examples in the node m wrongly classified as positive. This means that terminal nodes are labeled as positive or negative to minimize the misclassification cost.

In the CSDT method the idea is to calculate the expected misclassification cost reduction if a feature is used to divide the examples, compared with the expected cost if there is no further division i.e. a terminal node is created. In other words, the CSDT algorithm chooses a feature that reduce the misclassification cost the most. The process of partitioning and sub-tree creation is recursively repeated in the same way as in cost-insensitive tree induction. The minimal cost-based impurity measure is defined as:

$$Q_m(T) = \min\{cost(f_0(m)), cost(f_1(m))\} \tag{28}$$

where $f_k(m), k \in \{0, 1\}$ is defined as in (27). The feature selection measure used is relative cost reduction having the the same notation as gain measure for cost-insensitive DT model, with Q_m being defined as above. Clearly, if the feature is intended to reduce the cost of misclassification, we would like it to have higher gain measure.

As for the cost-insensitive decision tree, the algorithm grow the tree until some stopping criterion is met. Standard pruning methods are not suitable for imbalanced data as they are based on accuracy maximization or error-rate minimization. Nevertheless, described cost-complexity pruning method can be modified to include misclassification costs. Applied pruning methods generates a series of sub-trees and then selects one by examining the cost reduction of each of them, taking into account both the costs and the complexity (size) of the tree. Each internal node in the tree is the starting point for a subtree to be pruned. If a subtree is pruned, the internal node is turned to terminal node which is allocated to least costly class. The expected cost of a terminal node is then the number of training examples which do not belong to that class. Using this pruning method, nodes of the tree that do not contribute to the minimization of the misclassification cost will be pruned, regardless of their impact of on the accuracy of the model.

Procedure starts with a fully grown tree and classifies examples, noting the number of wrong classifications occurring in each class at each node. Then it calculates the cost of unpruned decision tree and for each internal node, count the errors for each class if the subtree becomes a terminal node labeled with least costly class. Based on counted errors and given cost matrix, the costs are calculated and used to obtain the gain measure for given internal node. The positive difference between the costs is a measure of the cost reduction from pruning the subtree.[6] The weight is incorporated to include the relative tree size reduction, where tree size is measured as number of internal nodes.

The node with maximum gain is used to prune the tree and the process continues successively to collapse internal nodes that produce highest relative cost reduction weighted by the relative decrease of the tree size measured as the number of internal nodes. In this way the algorithm penalizes the estimated gain based on the subtree size.

Notice, the starting cost is the same for every possible candidate node, therefore the gain maximization is equivalent to the minimization of the pruned tree's cost. In other words, pruned internal node is the node whose removal most decreases the cost. The node is converted to a terminal node if the misclassification cost for all samples sorted to pruned tree, is less than or equal to misclassification before pruning.

As mentioned, application of cost-sensitive learning methods assumes that costs are known. The costs are an additional input to the learning and pruning process. As discussed, when the costs are unknown at training time, they can only be approximate. This means inducing different tree model for every different combination of costs in cost matrix. Therefore, the cost matrix can be considered as crucial parameter in CSDT algorithm in order to tackle class imbalance problem.

3.4 Random Forest

Random forest method is ensemble method, viewed as variant of bagging method, where base models are generated using the same decision tree learning algorithm. Bagging methods use so called bootstrap sampling technique that consists of creating bootstrap subsets by randomly sampling from original training set with replacement. In random forest, when growing each base model, instead of only bootstrap sampling procedure used to obtain the data subsets for training the models, algorithm also sample over features, keeping only a random subset of them to build the model. Therefore, the difference relative to bagging is the incorporation of randomized feature selection. During the construction of a decision tree, at each split selection step, random forest firs randomly selects a subset of features during fitting base models on bootstrap replicates of the training set.

Normally, construction of an ensemble model consists of two steps. First, multiple base learners are generated and then combined to form composite predictor. The class for a given instance is then determined by some combination process such as majority voting, where voting denotes the contribution of a single vote i.e. prediction from a base model. Random forest classifier will predict the most frequent class considering predictions of all decision trees in the ensemble. That is, the final output class is the one that receives at least half of the votes.

3.4.1 ***Weighted Random Forest***. Ensemble classifier trained by using weighted decision trees as base learners and idea of random feature subset sampling leads to another method for tackling class

[6]Negative value indicates that subtree should not be pruned, since the cost would become higher by removing the sub-tree.

imbalance problem - Weighted Random Forest. As in weighted decision trees, class weights are adjusted and instead being uniform, they are set to be inversely proportional to class frequencies.

3.4.2 ***Cost-sensitive Random Forest***. Many research utilize cost-sensitivity under multiple classifier systems. A natural extension of cost-sensitive decision tree methods are cost-sensitive ensemble methods with cost-sensitive decision trees as base learners. The general idea of Cost-sensitive Random Forest (CSRF) is to induce different CSDT classifiers by using different bootstrap subsets from the training data like in classical random forest method. The classifiers are generated to maximum size with following modification contrasting to the random forest method: instead of selecting subset of attributes for the optimal split at each node, only search through a fix subset of randomly selected attributes, as in random subspace method [33]. Random subspace algorithm like random forest requires less computational cost than bagging since the method uses random subsets of features and give more stable results compared to random forest. The difference compared to standard random forest method is that the feature subset is selected for entire decision tree rather than at each split point in the tree. After desired number of CSDT classifiers is generated, their predictions are aggregated to make the final prediction of the CSRF classifier. The aggregation is possible with two voting schemes, majority or weighted voting.

We will consider an ensemble of B cost-sensitive decision trees each induced on a random subset S_j of the training set S by applying the CSDT learning algorithm. Each cost-sensitive decision tree classifier for simplicity will be denoted as $h_i(S_j) \equiv h_i$, $i = 1, \ldots, B$, where h_i outputs the class prediction, not the probability (hard-voting). Recall that h_i^k is the output for the class label $k \in \{0, 1\}$. Then, cost-sensitive ensemble model H is defined as:

$$H_B(S) = \underset{k \in \{0,1\}}{\operatorname{argmax}} \sum_{i=1}^{B} h_i^k(S) \tag{29}$$

in case the strategy for aggregating the outputs is majority voting. This means that after each of base models has been generated and has made prediction of class, the final prediction from ensemble model is class predicted from larger number of base learners, where if the result is tied, the final class is chosen arbitrarily.

In case the strategy for aggregating the outputs is weighted voting, each CSDT model h_i is given weight w_i during voting phase. The idea of weighted voting is to use unequal performance of the base learners by giving more power to the stronger learners. Then, cost-sensitive ensemble model H is defined as:

$$H_B(S) = \underset{k \in \{0,1\}}{\operatorname{argmax}} \sum_{i=1}^{B} w_i \cdot h_i^k(S) \tag{30}$$

where w_i $i = 1, \ldots, B$ is the weight assigned to the classifier h_i. In practice, the weights are mostly normalized and constrained by: $w_i \geq 0$ and $\sum_{i=1}^{B} w_i = 1$, since weights could take large negative and positive values thereby leading to ensemble method that will give extreme predictions although the base learners give reasonable predictions. Each weight is calculated by taking into account the relative cost reduction of corresponding CSDT classifier learned on training subset S_i, as follows:

$$w_i = \frac{RCR(h_i(S_i^{oob})}{\sum_{i=1}^{B} RCR(h_i(S_i^{oob}))} \tag{31}$$

where S_i^{oob} represents subset of samples from training set S which is not used in learning phase of CSDT model h_i. Calculation of weights on this way guaranties that each base classifier have contribution in ensemble according to it's contribution in misclassification cost reduction. The weighted voting with adequately assigned weights can be better than both the best individual base model and ensemble model with majority voting [35].

3.5 K-Nearest Neighbor

The K-nearest neighbor (KNN) is non-parametric method that for a given sample x estimates conditional probability for each class and then classifies the sample to the class with highest probability. The method identifies K samples in the training set that are closest to the x and estimates the probability for class $k \in \{0, 1\}$ as fraction of the K samples whose have class label equal to $k \in \{0, 1\}$. In case of majority voting, the algorithm returns most likely class:

$$\hat{y} = \underset{k \in \{0,1\}}{\operatorname{argmax}} \sum_{(x_i, y_i) \in \mathcal{N}} I(y_i = k) \tag{32}$$

where $\mathcal{N}$ denotes K instances closes to x according to some distance measure, I is indicator variable and $\hat{y}$ is predicted class that correspond to the most abundant class within the K-nearest neighbors. Instead of weighting equally all closes samples, another technique can be used. The algorithm can among K-nearest neighbors with varying distances, give more weight to samples closer to x, with weights calculated as the inverse of distance:

$$\hat{y} = \underset{k \in \{0,1\}}{\operatorname{argmax}} \sum_{(x_i, y_i) \in \mathcal{N}} I(y_i = k) \frac{1}{dist(x, x_i)} \tag{33}$$

or alternatively the additive-inverse of their distances:

$$\hat{y} = \underset{k \in \{0,1\}}{\operatorname{argmax}} \sum_{(x_i, y_i) \in \mathcal{N}} I(y_i = k)(1 - \frac{dist(x, x_i)}{dist_{max}}) \tag{34}$$

where $dist$ represents distance measure and $dist_{max}$ is maximum distance used for sake of normalization. However, those modifications of KNN are insignificant if in the neighborhood of a test sample one of the classes is underrepresented, since all of the K neighbors are close to the sample and the difference among their distances is not discriminatory [21]. The study [21] with detail analysis reveals that classification mechanism of KNN is based on finding the class label that has higher prior probability, thereby suggesting that in case of skewed distribution of target variable KNN has suboptimal performance on the minority class. This holds especially in overlapping regions, where even using inverse distances as weights is ineffective in correcting this bias.

KNN is an instance-based learning algorithm which does not need to maintain a model derived from training data. The idea of the algorithm is that given a test sample, the algorithm computes the similarity measure between the test sample and all training samples to determine its K-nearest neighbors. The problem with class imbalance is in process of determining class for the test sample, since the choice is made based on the most abundant class within

the K-nearest neighbor samples, where most of the samples belong to majority class since samples of the minority class occur sparsely in the data space. Higher probability of samples from the majority class leads to higher probability that samples from minority class be incorrectly classified.

As there is inherent bias to majority class in existing KNN algorithms on any distance measurement [21], we try to generate cost-sensitive KNN by using publicly available code[7]. However, obtaining results on the size of BcBase datasets was infeasible. Therefore, as modification of baseline KNN we propose KNN with non-uniform weights, whose parameters such as number of neighbors are optimized. In results section we use KNNw to denote this approach.

4 EXPERIMENTAL EVALUATION

In evaluation protocol for data partitioning is used 10-fold cross-validation procedure as in study [30], where the model is trained on K-1 folds and the left-out fold is used in testing phase of the model. Moreover, in each fold are used the same instances with aim to decrease potential variability in results generated using different instances per fold. This is accomplished by using predefined instances per fold obtained in cooperation with researchers from the study.

Additionally, all datasets are prepared in appropriate way to meet requirements of each method. Data preparation is important and critical step in learning process of any model as the data has strong influence on the results of a generated model. Furthermore, effective data preparation can increase generalization ability of models. For SVM based classifiers, data with different scales often lead to the instability of the classifier, hence the numerical features are scaled between 0 and 1 to ensure that all features have a similar scale. For Bayesian methods, numerical features are discretized leading to datasets with only categorical and ordinal features, which are further encoded using one-hot-encoding technique. Preparation for tree-based methods (decision tree and random forest methods), included one-hot-encoding of categorical features only, as these methods can handle ordinal and numerical features. For KNN methods datasets are prepared in the same way as for decision tree based methods.

4.1 Evaluation measures

In our conducted empirical search, several performance metrics was employed. Choosing appropriate measure for evaluating is important step in conducting any research since the measure should accurately reflect the classifier's performance. There is agreement that this choice depends on the classification purpose and the domain of the dataset [22]. Different evaluation measures can reflect different aspect of classifier's performance. In general, choosing appropriate evaluation measure is challenging task, especially within class imbalance framework, as most evaluation measures assume balanced class distribution and equal misclassification costs.

When performance of the model over minority (positive) class is considered, normally two performance measures are explored, recall and precision. A recall is determined as:

$$recall = \frac{TP}{TP + FN} \tag{35}$$

where TP is number of correctly classified minority instances and FN is number of misclassified minority instances. While recall expresses the ability of model to find all relevant examples in a dataset, precision expresses the proportion of the data points our model classifies as relevant that are actually relevant. Precision is defined as:

$$precision = \frac{TP}{TP + FP} \tag{36}$$

where FP is number of misclassified majority instances. Precision and recall measures calculated by using formulae from above refer to minority class. Precision and recall metrics can be also defined for majority class, where the former is calculated as the number of instances correctly classified to negative class divided by the total number of instances classified to negative class and the latter is calculated as the number of instances correctly classified to negative class divided by the total number of instances that belong to majority (negative class). Precision and recall score for the classifier, which are reported in results section, are obtained by averaging precision and recall scores per class.

Within imbalanced framework a goal is to improve recall for minority class without decreasing precision. This goal is mostly violated since increasing number of correctly classified minority examples, increases number of wrongly classified majority examples. Reaching the goal is important since recall describes how well classifier learns minority class, while precision tells us how well a classifier removes majority class from being misclassified as minority class. Precision of classifier depends on the ratio of positive to negative class in data while recall is invariant to class imbalance [34]. As the class imbalance ratio increases, precision will decrease. Precision-recall curve is one way to summarize the performance of a classifier. However, as stated in [34], it is better to summarize precision-recall performance by using $F1$ measure. The F_1 score is the harmonic mean of precision and recall. It represents special case of F_β measure, defined as weighted harmonic average:

$$F_\beta = \frac{(1 + \beta^2) precision \cdot recall}{\beta^2 precision + recall} \tag{37}$$

Coefficient β balances relative importance between recall and precision. From the definition it can be seen that, F_1 is derived when both measures contributions are equal. Therefore F_β indicates whether classifier obtains high recall at the cost of lower precision. The attention during hyperparameter tuning was directed on the F_β metric as it beside precision include the accuracy for the minority class samples, which are the main interest when classifying imbalanced data. Moreover, parameter β enable control over weight given to the recall. In our setting, to reflect the fact that minority class examples are more important, we have been using $\beta = 2$, meaning the recall of minority class contributes twice to the score for positive class relative to precision.

In reported results, we include classification accuracy, since it is also included in [30]. Beside, it is most commonly used evaluation metric. However, in class imbalance framework it is inappropriate measure, since high accuracy of a classifier is consequence of

[7] https://github.com/DEC4F/cost-sensitive-knn

classifier bias towards the negative class. Therefore, comparing classifiers performance only based on accuracy can be missleading.

4.2 Experimental setup

All experiments were conducted in Python using the available implementations and utilities of open-source libraries, including scikit-learn and costcla library, where the former is used for generation of all non cost-sensitive models and the latter library is used for implementation of modified and adapted cost-sensitive classification methods.

Each classifier is trained using hyperparameters tuned with grid search method. Although, the computational time required for grid search can be quite extensive, we used it for this purpose, as for each model only few hyperparameters are tuned. Having a good sense of what each hyperparameter of the model actually does can help us search in the right part of the hyperparameter space. For all Naive Bayes methods we try several different values for smoothing parameter, as is done in [12], starting with value 0.1 and increasing it by 0.1 until smoothing parameter is 5. For SVM we have chosen Radial Basis Function, which is considered as a reasonable first choice ([5]) as it has few parameters. Hence the hyperparameter tuning included only regularization parameter C and γ. For weighted SVM the optimization include the misclassification cost C^- and C^+, several values are tried, although this cost can be set as constant as is done in [5]. In our application of decision tree is used default value for all hyperparameters except max depth of the tree, minimum number of samples in leaf and cost-complexity parameter α. In grid search procedure these are sampled from the specified distributions. For weighted decision tree, additionally are explored different class weights. Similarly, for CSDT model the search space is expanded to include misclassification costs for positive and negative class. We investigated a variety of cost matrices with assumption that correct classification has no cost and without loss of generality, we impose condition $C_{FP} = 1$ [8] and vary values for C_{FN}, as is done in many studies [2, 25]. The hyperparameters to optimize in a CSRF classifier are the number of decision trees and the maximum number of features considered for splitting at each node as well as cost for misclassifying minority class. The search for optimal hyperparameters with aim to improve classifier's performance is guided by $F2$ score, to reflect the importance of correct classification of minority class.

5 RESULTS

The data sets descriptions followed by results of our experiments and conducted analysis are presented in this section.

5.1 Data description

In development process of described models are used datasets from BcBase database, received with cooperation with the authors of the study [30]. As explained in the study, BcBase is a population-based research database containing data about patients in early breast cancer stage from three healthcare regions in Sweden that account for approximately 60% of the total population. Data contain information about patient and tumor characteristics, treatment strategies and prescribed medicament(s). Although the database does not include QoL indicators, from prescribed medications the presence of given QoL issue is derived leading to creation of 4 datasets BcBase-Anxiety, BcBase-Depression, BcBase-Insomnia and BcBase-Pain. The datasets contain binary target variable denoting whether a patient is suffering from the issue i.e. anxiety, depression, insomnia and pain, after breast cancer treatment. In total, each dataset contains 18988 instances (patients) described by predictor features which are used to obtain best possible predictive model for QoL after breast cancer treatment.

One popular class-imbalance measure, imbalance ratio, can be defined by the ratio of the majority class size to minority class size. Described datasets have different imbalance ratios ranging from 1 to 2.5. More precisely, ratio of negative class instances to positive class instances for BcBase-Anxiety, BcBase-Depression, BcBase-Insomnia and BcBase-Pain is 2.309, 2.359, 1.079 and 2.499, respectively. This means that for each patient that expresses any symptom except for insomnia, there are 2 patients that do not express the symptom. According to the imbalance ratio, BcBase-Insomnia has almost perfectly balanced distribution i.e. for every patient that experiences insomnia after diagnosis there is roughly the same number of patients which do not experience this symptom.

5.2 Models performance

The performance of binary classification models trained on BcBase datasets is evaluated by the 10-fold cross-validation and results are summarized in Table 1 to Table 4. In exposed tables, asterisks (*) next to model's name denotes models from the study [30] and mentioned project, which we consider as baseline models. Hence, results for baseline models in given tables correspond to the tables from the study. Below each baseline model is proposed alternative model(s). In exposed results suffix w is used in name of each classifier that incorporates class weights or in case of Cost-sensitive Random Forest method with weighted voting strategy, w refers to the voting. To make the experiments more comprehensive and have a better idea about the learning performance of each classifier, several metrics is employed. More precisely, accuracy (Acc), f1 score (F1), macro-averaged precision (Prec) and macro-averaged recall (Rec). Other measures such as precision and recall per class are also analysed. Recall, the positive class refers to patients experiencing anxiety, depression, insomnia and pain after cancer diagnosis, whilst patients without those symptoms belong to the negative class.

From the reported results for BcBase-Anxiety datasets in Table 1, it can be noted that Complement Naive Bayes (CNB) outperforms baseline Naive Bayes (best performing model in the study [30]) in terms of F_1, precision and recall measures except for accuracy, which as explained is not appropriate measure for imbalanced data. Highest recall has Support Vector Machine model with class weights (SVMw) and CNB. The difference in precision of SVMw and CNB can be considered as insignificant as the results differ on third decimal place. Although there is slight difference between CNB and SVMw in terms of F_1 score, we can favor the latter, since by detailed analysis on precision and recall per class, we discovered that SVMw has highest recall for minority class (Rec+) which compensates the

[8] Note that, an arbitrary cost matrix can be normalized to become a cost matrix satisfying these conditions.

Table 1: Results for all classification models on BcBase-Anxiety dataset

Model	Acc	F1	Prec	Rec
RF*	0.673	0.514	0.484	0.535
RFw	0.692	0.442	0.548	0.506
CSRF	0.609	0.523	0.524	0.523
CSRFw	0.609	0.522	0.524	0.522
SVM*	0.698	0.411	0.349	0.500
SVMw	0.543	0.531	0.558	0.568
NB*	0.629	0.552	0.553	0.552
MNB	0.642	0.553	0.558	0.552
CNB	0.591	0.556	0.559	0.568
KNN*	0.682	0.458	0.529	0.507
KNNw	0.621	0.513	0.518	0.515
DT*	0.583	0.511	0.511	0.511
DTw	0.566	0.539	0.549	0.557
CSDT	0.568	0.546	0.559	0.570

difference, as is desirable to have a classifier that gives high prediction accuracy over the minority class while maintaining reasonable accuracy for the majority class. Furthermore Rec+ better reflect ability of a model to correctly classify examples form minority class. Therefore, SVMw would be preferred over both CNB and baseline NB*.

Table 2: Results for all classification models on BcBase-Depression dataset

Model	Acc	F1	Prec	Rec
RF*	0.677	0.473	0.524	0.509
RFw	0.685	0.501	0.562	0.526
CSRF	0.613	0.515	0.517	0.515
CSRFw	0.613	0.518	0.520	0.518
SVM*	0.702	0.413	0.351	0.500
SVMw	0.554	0.535	0.554	0.564
NB*	0.566	0.534	0.543	0.551
MNB	0.646	0.537	0.546	0.537
CNB	0.554	0.531	0.547	0.556
KNN*	0.688	0.462	0.536	0.509
KNNw	0.626	0.509	0.515	0.512
DT*	0.589	0.515	0.515	0.515
DTw	0.552	0.511	0.517	0.519
CSDT	0.567	0.534	0.541	0.549

According to the results from Table 2, similar conclusions can be made except for SVMw and Multinomial Naive Bayes (MNB) with even smaller difference in $F1$ score and larger difference in recall score for positive class. Rec+ of MNB is more than twice as small compared to Rec+ of SVMw. Apparently, again SVMw outperforms baseline NB* classifier in terms of $F1$, precision and recall.

Exposed results for BcBase-Insomnia in Table 3 illustrate that several classifiers has almost the same $F1$ score, with negligible difference on third decimal place. More precisely, highest $F1$ score again has SVMw followed by CNB and MNB. Furthermore, by

Table 3: Results for all classification models on BcBase-Insomnia dataset

Model	Acc	F1	Prec	Rec
RF*	0.538	0.526	0.535	0.532
RFw	0.558	0.555	0.557	0.556
CSRF	0.524	0.523	0.523	0.523
CSRFw	0.522	0.521	0.521	0.521
SVM*	0.541	0.533	0.539	0.537
SVMw	0.569	0.569	0.570	0.570
NB*	0.555	0.554	0.554	0.555
MNB	0.558	0.556	0.557	0.556
CNB	0.558	0.558	0.558	0.558
KNN*	0.521	0.502	0.516	0.514
KNNw	0.509	0.508	0.508	0.508
DT*	0.516	0.515	0.515	0.515
DTw	0.552	0.552	0.553	0.553
CSDT	0.549	0.549	0.551	0.551

comparing Rec+ of these three models, it is observed that SVMw outperforms the CNB and MNB in terms of Rec+. However, highest Rec+ has Cost-sensitive Decision Tree model (CSDT). More importantly, SVMw outperforms best model from the study, in all relevant measures, $F1$, precision and recall as well as recall for minority class.

Table 4: Results for all classification models on BcBase-Pain dataset

Model	Acc	F1	Prec	Rec
RF*	0.698	0.486	0.554	0.518
RFw	0.698	0.518	0.575	0.534
CSRF	0.628	0.523	0.526	0.523
CSRFw	0.625	0.520	0.522	0.520
SVM*	0.714	0.417	0.357	0.500
SVMw	0.579	0.553	0.567	0.582
NB*	0.530	0.517	0.553	0.564
MNB	0.661	0.543	0.554	0.543
CNB	0.568	0.538	0.552	0.564
KNN*	0.699	0.457	0.528	0.506
KNNw	0.641	0.509	0.517	0.513
DT*	0.604	0.522	0.522	0.522
DTw	0.545	0.524	0.547	0.558
CSDT	0.619	0.561	0.560	0.567

Results in Table 4 for BcBase-Pain, dataset with highest imbalance ratio, indicate that best performing model in terms of $F1$ score is CSDT followed by SVMw. Notice, both CSDT and SVMw outperform DT which is best performing model in terms of $F1$ score in the study. More importantly, analysing *Rec+* discovered that CSDT has better ability to correctly classify patients that experience the pain compared to baseline DT.

Finally, by analysing all reported results it can be observed that in most cases, each baseline model is outperformed by proposed alternative classifier(s), with few exceptions. Several conclusions

can be drawn in terms of $F1$ score. First of all, support vector machine classifier with class weights (SVMw), outperforms baseline SVM for all datasets. The same can be inferred for optimized KNN classifier with non-uniform distances (KNNw) compared to baseline KNN. Decision tree classifier performs badly when compared to all other proposed tree-based machine learning models except for BcBase-Depression, where only cost-sensitive ML model outperforms baseline DT. Regard ensemble of decision trees, we can conclude that baseline random forest is outperformed by all proposed tree-based ensemble versions for pain and depression issues, while for anxiety and insomnia this is not the case for random forest with class weights (RFw) and cost-sensitive RF models (CSRF and CSRF), respectively. Multinomial Naive Bayes (MNB) and Complement Naive Bayes (CNB) perform better than baseline Naive Bayes (NB) except for BcBase-Depression dataset, where CNB does not achieve improvement over results of NB. Nonetheless, for each individual Bc-Base dataset, results of best performing baseline model are improved what is most important in terms of *Rec+* without decrease in $F1$ score.

6 CONCLUSION AND FUTURE WORK

The purpose of this study is to extend previously conducted research study by providing more comprehensive framework for tackling situation when one class is underrepresented. The focus was on QoL datasets for breast cancer patients, with slight imbalance ratios. Obtained results show that all proposed methods achieve considerable improvement over their baseline versions. Therefore utilized techniques showed to be very promising in imbalanced framework. On basis of experimental evaluation support vector machine classifier with class weights (SVMw) stands out as best performing model over other proposed machine learning models for imbalanced data and best performing baseline model from previous study as well. Hence, we succeed to improve results of the previous study by creating SVMw, more appropriate model for breast cancer patients with imbalanced distribution of all QoL symptoms.

In future work, other cost-sensitive ensemble learning techniques should be considered such as stacking, where several different cost-sensitive base models could be combined, including support vector machine with class weights. Furthermore, instead typical methods for hyperparameter optimization, some other optimization techniques can be used, since used grid search method has extensive execution time and enlarging search space in grid method makes the computational cost to conduct the search even higher. New approach could bring most benefits and improvement in tuning misclassification cost for cost-sensitive decision trees (CSDT), which have competitive results with SVMw but are more easy to interpret then SVM models and have nice intuitive representation of decision making process in view of hierarchical structure.

REFERENCES

[1] Correa Bahnsen A. 2015. Example-dependent cost-sensitive decision trees. *Expert Systems with Applications* 42 (2015), 6609–6619. https://doi.org/10.1016/j.eswa.2015.04.042

[2] Weiss G. McCarthy K. Zabar B. 2007. Cost-Sensitive Learning vs. Sampling: Which is Best for Handling Unbalanced Classes with Unequal Error Costs? (2007), 35–41.

[3] Cortes C. and Vapnik V. 1995. Support-Vector Networks. *Mach. Learn.* 20, 3 (sep 1995), 273–297. https://doi.org/10.1023/A:1022627411411

[4] Elkan C. 2001. The Foundations of Cost-Sensitive Learning. *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence* (2001).

[5] P. Cao, Dazhe Zhao, and Osmar Zaïane. 2013. An Optimized Cost-Sensitive SVM for Imbalanced Data Learning. 280–292. https://doi.org/10.1007/978-3-642-37456-2_24

[6] Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A Library for Support Vector Machines. *ACM Trans. Intell. Syst. Technol.* 2, 3, Article 27 (may 2011), 27 pages. https://doi.org/10.1145/1961189.1961199

[7] Aridas CK., Karlos S., Kanas VG., Fazakis N., and Kotsiantis SB. 2020. Uncertainty Based Under-Sampling for Learning Naive Bayes Classifiers Under Imbalanced Data Sets. *IEEE Access* 8 (2020), 2122–2133. https://doi.org/10.1109/ACCESS.2019.2961784

[8] José Luis Polo Fernando Berzal Juan Carlos Cubero. [n. d.]. Weighted Classification Using Decision Trees for Binary Classification Problems. Department of Computer Science and Artificial Intelligence, ETSIIT, University of Granada, Spain.

[9] Field J. Holmes MM. Newell D. 2019. PROMs data: can it be used to make decisions for individual patients? A narrative review. *Patient related outcome measures* (2019), 233–241. https://doi.org/10.2147/PROM.S156291

[10] Rennie JDM. Shih L. Teevan J. Karger DR. 2003. Tackling the Poor Assumptions of Naive Bayes Text Classifiers *(ICML'03)*. AAAI Press, 616–623.

[11] Cristianini N. Ricci E. 2008. *Support Vector Machines*. Springer US, Boston, MA. 928–932 pages. https://doi.org/10.1007/978-0-387-30162-4_415

[12] Seref B. Bostanci E. 2019. Performance of Naïve and Complement Naïve Bayes Algorithms Based on Accuracy, Precision and Recall Performance Evaluation Criterions. *International Journal of Computing Academic Research* 8 (10 2019), 75–92.

[13] Weiss G. Provost F. 2001. The Effect of Class Distribution on Classifier Learning: An Empirical Study. *Tech Rep* (09 2001).

[14] Aurélien G. 2017. *Hands-on Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems* (2 ed.). O'Reilly Media. 543 pages.

[15] Brereton R. Lloyd G. 2010. Support Vector Machines for classification and regression. *The Analyst* 135 (02 2010), 230–67. https://doi.org/10.1039/b918972f

[16] Wong TT. Tsai HC. 2021. Multinomial naïve Bayesian classifier with generalized Dirichlet priors for high-dimensional imbalanced data. *Knowledge-Based Systems* 228 (2021), 107288. https://doi.org/10.1016/j.knosys.2021.107288

[17] Hastie T. Tibshirani R. Friedman J.H. 2009. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Second edition.

[18] Md Kadir, Pritom Akash, Sadia Sharmin, Amin Ali, and Mohammad Shoyaib. 2020. *A Proximity Weighted Evidential k Nearest Neighbor Classifier for Imbalanced Data*. 71–83. https://doi.org/10.1007/978-3-030-47436-2_6

[19] Maninder Kaur, Meghna Dhalaria, Pradip Kumar Sharma, and Jong Hyuk Park. 2019. Supervised Machine-Learning Predictive Analytics for National Quality of Life Scoring. *Applied Sciences* 9, 8 (2019). https://www.mdpi.com/2076-3417/9/8/1613

[20] Chen C. Breiman L. 2004. Using Random Forest to Learn Imbalanced Data.

[21] Wei Liu and Sanjay Chawla. 2011. Class Confidence Weighted kNN Algorithms for Imbalanced Data Sets. *Adv Knowl Discov Data Min* 6635, 345–356. https://doi.org/10.1007/978-3-642-20847-8_29

[22] Zhang S. Sadaoui S. Mouhoub M. 2015. An Empirical Analysis of Imbalanced Data Classification. *Computer and Information Science* 8 (02 2015), 151–162. https://doi.org/10.5539/cis.v8n1p151

[23] Alfons J. Hermann N. 2002. Reversing and Smoothing the Multinomial Naive Bayes Text Classifer. 200–212.

[24] Chawla N. Japkowicz N. and Kołcz A. 2004. Editorial: Special Issue on Learning from Imbalanced Data Sets. *SIGKDD Explorations* 6 (06 2004), 1–6. https://doi.org/10.1145/1007730.1007733

[25] Domingos P. 2002. MetaCost: A General Method for Making Classifiers Cost-Sensitive. *Proceedings of the Fifth ACM SIGKDD Int'l. Conf. on Knowledge Discovery Data Mining*. (2002). https://doi.org/10.1145/312129.312220

[26] Turney P. 2002. Types of Cost in Inductive Concept Learning.

[27] James G. Witten D. Hastie T. Tibshirani R. 2017. An Introduction to Statistical Learning: with Applications in R.

[28] Eibe F. Remco RB. 2006. Naive Bayes for Text Classification with Unbalanced Classes. In *PKDD*. 503–510.

[29] Shoab Saadat, Ayesha Aziz, Hira Ahmad, Hira Imtiaz, Zara Sohail, Alvina Kazmi, Sanaa Aslam, Naveen Naqvi, and Sidra Saadat. 2017. Predicting Quality of Life Changes in Hemodialysis Patients Using Machine Learning: Generation of an Early Warning System. *Cureus* 9 (09 2017). https://doi.org/10.7759/cureus.1713

[30] Miloš Savić, Vladimir Kurbalija, Mihailo Ilić, Mirjana Ivanovic, Dusan Jakovetic, Antonis Valachis, Serge Autexier, Johannes Rust, and Thanos Kosmidis. 2021. Analysis of Machine Learning Models Predicting Quality of Life for Cancer Patients. 35–42. https://doi.org/10.1145/3444757.3485103

[31] Jinah Sim, Young Kim, Ju Kim, Jong Lee, Moon Soo Kim, Young Shim, Jae Zo, and Young Ho Yun. 2020. The major effects of health-related quality of life on 5-year survival prediction among lung cancer survivors: applications of machine learning. *Scientific Reports* 10 (07 2020), 10693. https://doi.org/10.1038/

s41598-020-67604-3

[32] Yanmin S. Wong AKC. Mohamed SK. 2011. Classification of imbalanced data: a review. *International Journal of Pattern Recognition and Artificial Intelligence* 23 (11 2011). https://doi.org/10.1142/S0218001409007326

[33] Ho TK. 1998. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20, 8 (1998), 832–844. https://doi.org/10.1109/34.709601

[34] Christopher K. I. Williams. 2021. The Effect of Class Imbalance on Precision-Recall Curves. *Neural Computation* 33, 4 (04 2021), 853–857. https://doi.org/10.1162/neco_a_01362 arXiv:https://direct.mit.edu/neco/article-pdf/33/4/853/1908890/neco_a_01362.pdf

[35] Zhi-Hua Z. 2012. *Ensemble Methods: Foundations and Algorithms.* Chapman and Hall/CRC. 236 pages. https://doi.org/10.1201/b12207