Czech Technical University in Prague

Faculty of Electrical Engeneering
Department of Cybernetics

Influence of Class Imbalance on Active Learning of Sleep EEG Classifier

Master Thesis

Supervisor: Ing. Martin Macaš, PhD.
Study programme: Biomedical Engineering and Informatics
Specialisation: Biomedical Informatics

Bc. Nela Grimová

May 2017

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

DIPLOMA THESIS ASSIGNMENT

Student: Bc. Nela G r i m o v á

Study programme: Biomedical Engineering and Informatics

Specialisation: Biomedical Informatics

Title of Diploma Thesis: Influence of Class Imbalance on Active Learning of Sleep EEG Classifier

Guidelines:

1. Make a survey, propose and implement criteria for evaluation of active learning.
2. Evaluate experimentaly active learning for data with different class imbalances. (Use synthetic data sets and observe the influence of class imbalance on performance of the active learning.)
3. Use the previous outputs for proposal, implementation and evaluation of active learning for detection of sleep stages from EEG signals (provided by the supervisor).

Bibliography/Sources:

[1] Settles, Burr. "Active learning literature survey." University of Wisconsin, Madison 52.55-66 (2010): 11.
[2] Ertekin, S., Huang, J., Bottou, L., & Giles, L. (2007, November). Learning on the border: active learning in imbalanced data classification. In Proceedings of the sixteenth ACM conference on Conference on information and knowledge management (pp. 127-136). ACM.
[3] Roederer, A. (2012). Active Learning for Classification of Medical Signals (Doctoral dissertation, University of Pennsylvania).

Diploma Thesis Supervisor: Ing. Martin Macaš, Ph.D.

Valid until: the end of the summer semester of academic year 2017/2018

L.S.

prof. Dr. Ing. Jan Kybic
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, January 11, 2017

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, 26. 5. 2017

Acknowledgements

I would like to express my gratitude to my thesis supervisor Ing. Martin Macaš, PhD. for his guidance, valuable advice and willingness.

Furthermore, I would like to thank Ing. Václav Gerla, PhD. for being helpful in discussions about EEG data, Mgr. Kyriaki-Maria Saiti for her valuable comments and Mgr. Matěj Bregant and Bc. Eliška Sekáčová for their corrections.

My thanks also go to my family and close ones for supporting and encouraging me.

Abstrakt

Tato diplomová práce se zabývá aktivním učením pro klasifikaci spánkových stavů na EEG datech. Jelikož spánková data jsou obecně nevyvážená, nejdříve sledujeme, jak nevyváženost na syntetických datech ovlivňuje aktivní učení využívající strategii nejistoty a jejích variant, které zohledňují rozmístění instancí v prostoru. Je navrhnuto několik algoritmů, z nichž jeden ušetřil více než 80 % instancí na čtyřech z pěti datasetů. Toto ušetření je významné vzhledem k tomu, že spánková data jsou anotována specialistou, který v současné době musí ke všem instancím přiřadit jejich třídu. V neposlední řadě jsou v této práci navrhnuta vyhodnocovací kritéria pro srovnání metod aktivního učení či pro zjištění, zda je navržená metoda aktivního učení lepší než náhodný výběr instancí.

Klíčová slova: aktivní učení; strategie nejistoty; EEG; spánkové EEG; vyhodnocí kritéria

Abstract

This master thesis deals with active learning used for the classification of sleep stages on EEG data. Since sleep data are in general imbalanced, we first focus on how class imbalance of synthetic data influences active learning utilizing the uncertainty sampling strategy and its density-weighted variants. Several algorithms have been proposed, one of them saves more than 80 % of instances on four out of five datasets. These savings are significant with regard to the fact that sleep data are annotated by a specialist who must currently go through all instances and classify them. Last but not least, evaluation criteria are proposed in this thesis for the comparison of active learning methods or to verify if a suggested method is better than random sampling.

Keywords: active learning; uncertainty sampling strategy; EEG; sleep EEG; evaluation criteria

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

In reality, when it comes to a lot of pattern classification problems of the real world, data instances are scarce and difficult to come by. Moreover, a situation is even worse when we are able to obtain some data but assigning them to a proper class is expensive or time-consuming. This includes applications, where a large amount of data is available but to divide them into categories is difficult (e.g. assigning of topics to documents [1]), or when a human annotator is needed.

The classification of sleep stages is a traditional task. A specialist studies a patient's record of biophysical changes during the sleep (it is a record of EEG, EMG, EOM and ECG activity in most cases) and determines in which sleep stage the patient was. It is clear that this approach is tedious so there are tendencies to use machine learning techniques and automatize the process.

The main idea of active learning is to select a few of the most informative instances and use them in order to train a classifier. This process is based on asking queries about a class of the selected instance, and then the answer is provided by an oracle (e.g. by the specialist) [2]. Three strategies of how a query can be created will be discussed in Section 2.2.

It is assumed that as a result that a smaller amount of observations to learn a classifier will be needed and thus time and cost of assigning instances to their classes will be saved.

The aim of this thesis is to take the advantage of active learning methods and use them for classification of sleep stages, so the specialist will not have to annotate the whole record of sleep but only a few instances with the same achieved level of accuracy.

Since stages of sleep are largely imbalanced, this fact hinders the traditional machine learning approaches, it is necessary to first figure out how a class imbalance influences active learning process.

1.2 Objectives

The first aim of this thesis is to conduct a survey about the current situation in active learning and also in addressing the class imbalance and as well to figure out which state-of-the-art methods are utilized.

Active learning methods are in most cases compared by the visual comparison of their values of a metric depending on the number of instances which were used for learning, so the other goal of the thesis is to propose criteria which can be utilized for an evaluation of active learning.

The next goal is to carry out experiments on synthetic datasets with different levels of class imbalance, evaluate them and observe how the class imbalance influences the performance of active learning.

The final step is to use the acquired knowledge for the application of active learning for detection of sleep stages in EEG signals, to implement chosen methods and to evaluate them.

1.3 Structure of the Thesis

We will provide the definition of active learning and mention all aspects connected to active learning in Section 2. Active learning is divided into three categories, according to which way the data used for learning are acquired. Then, we will mention strategies for selecting the most informative instances. We will especially deal with uncertainty sampling and density-weighted strategies, so these strategies will be discussed more in the last part of the second section, together with the initialization and the termination of active learning.

In Section 3, several metrics, which are used for evaluation of an algorithm performance, will be briefly introduced. Then it will be focused on evaluation of active learning, which is usually done visually. Several evaluation criteria that are suitable exactly for active learning will be proposed.

We deal with class imbalance in Section 4. The definition, what the class imbalance is, and what are the methods that are used for handling it, will be proposed. Then the connection between active learning and learning on skewed datasets will be described. We will perform experiments on synthetic datasets with different class imbalance and discuss them.

In Section 5, a brief introduction to sleep EEG will be provided and we will use results achieved in Section 4 for the utilization of active learning for detection of sleep stages. We will perform experiments on real data and discuss acquired results.

Results of the thesis will be summarized in the Chapter 6, they will be discussed and problems, on which it would be necessary to focus on in the future work, will be detected.

1.4 Types of Learning

Three categories of machine learning are often distinguished: supervised, unsupervised and reinforcement learning [3]. Let us provide a brief description of each category and also mention semi-supervised learning, which is a combination of supervised and unsupervised learning.

1.4.1 Supervised Learning

In supervised learning, we are given an instance (observation) space X, a state (class) space Y and a probability density $P_{XY}(x, y)$ which describes the joint probability that the instance x is observed and belongs to the class y. The goal of the task is to find a classifier $f : X \to Y$ which for every observation $x \in X$ assigns its state $y \in Y$. It is expected that classifier f does not estimate the correct state y for every observation x. Hence let $L : X \times X \to \mathbb{R}$ be a loss function which represents an amount of loss of the

estimated state due to the real state. Let y_R be the real state and y_E the estimated state, then $L(y_R, y_E)$ returns the corresponding amount of loss.

When the number of classes is finite, we call this task classification [3]. Let us consider a typical classification task. We are given a set S_{SL} of n pairs, each pair consists on the observation x^i and its class y^i: $S_{SL} = \{(x^1, y^1), ..., (x^n, y^n)\}$, $x^i \in X$, $y^i \in Y$. We would like to find the classifier f such that its expected value of loss $R(f)$ (also called a risk of the classifier [4]) is as small as possible:

$$f = \arg \min_{f:X \to Y} R(f) = \arg \min \sum_{x \in X} \sum_{y \in Y} L(x, f(x)) P_{XY}(x, y) \tag{1}$$

With the zero-one loss function $L_{01}(y, f(x))$, i.e. with the function which returns 0, if the estimated and the real state is equal, or 1, if the estimated and the real state differ, the risk becomes the classification error e:

$$e = \sum_{x \in X} \sum_{y \in Y} L_{01}(x, f(x)) P_{XY}(x, y) \tag{2}$$

1.4.2 Unsupervised Learning

Unlike in supervised learning, there is no knowledge about classes in unsupervised learning, only the observation space X is known. As an input, a program receives a set S_{UL} which contains n instances from X, $S_{UL} = \{x^1, ..., x^n\}$.

The goal of the task differs according to the application. It covers finding common features of data in S_{UL} and cluster these instances into k sets (e.g. k-means clustering), the estimation of the probability density P_X or the dimensionality reduction [3].

1.4.3 Semi-supervised Learning

Semi-supervised learning is a combination of supervised and unsupervised learning. Let us assume an observation space X, a class space Y and a probability density $P_{XY}(x, y)$. Semi-supervised learning deals with both unassigned and assigned observations of the class.

Let S_{SSL} be the input set of instances which is the union of sets S_L and S_U, i. e. $S_{SSL} = S_L \cup S_U$. S_L has the same meaning as the set S_{SL} in supervised learning, thus it is a set of m pairs of observations and their classes: $S_L = \{(x^1, y^1), ..., (x^m, y^m)\}$. On the other hand, S_U is similar to set S_{UL} in unsupervised learning and contains n instances from X without the assigned class: $S_{UL} = \{x^{m+1}, ..., x^{n+m}\}$. Let us call instances from set S_L labeled instances and instances from set S_U unlabeled instances. Denote that it holds $m \ll n$.

Labeled instances provide a prior knowledge about a problem and constrain the input space. A common example of semi-supervised learning is the semi-supervised variant of EM algorithm [5].

1.4.4 Reinforcement Learning

Reinforcement learning is often used in agent-based systems. Reinforcement learning is similar to supervised learning, but there is no prior knowledge about the set S_{SL}. A set of agent's actions A and a set of states T are given. An agent moves from the state t_i to the state t_j, $t_i, t_j \in T$ by taking actions and exploring an environment. Each made action gives him a reward r. Acquired knowledge helps the agent to higher ability of reasoning. [6].

1.4.5 Determination of Active Learning

Active learning can be considered as a part of semi-supervised learning (Section 1.4.3), although the term active learning also appears in literature about reinforcement learning [6], where it is used for the algorithm in which an agent actively searches through its environment. Note that active learning is also one of social psychology fields [7], but it has nothing in common with our topic.

2 Active Learning

A description of active learning will be given in this section. We will provide a brief motivation to answer the question why it should be given more attention in the field of active learning and what are its advantages.

2.1 Definition

Let X be the observation space and Y the finite state space. We are given a set of unlabeled instances S_U and a set of labeled instances S_L, which have the same meaning, as it is proposed in Section 1.4.3 and it holds that $|S_U| >> |S_L|$, the classifier $f : X \to Y$ and the oracle O for annotating selected instances.

The most informative observation $x^* \in S_U$ is chosen in each step of an active learning algorithm according to the performance of the classifier f trained on labeled instances. The methods of creating queries will be discussed later. Let $\overset{O}{\to}$: $X \to X \times Y$ be the mapping which denotes that the oracle assigns a label to the observation. Selected instances are labeled by the oracle, i.e. $x^* \overset{O}{\to} (x^*, y)$, $y \in Y$, and (x^*, y) is added to the set S_L.

It is required to answer several questions before implementing active learning. Firstly, as we have mentioned before, it can be difficult to obtain a set of unlabeled instances S_U. Processes of acquiring unlabeled instances will be described in the next section.

It is also needed to decide how the process will be initialized (i.e. how the initial set S_L will be created) and terminated. It will be mentioned in Section 2.4.3 and Section 2.4.4.

2.2 Unlabeled Instances Sampling

Sampling of unlabeled instances is divided in up to three categories: pool-based sampling, stream-based sampling and membership query synthesis [2,8]. Let us remark that pool-based and stream-based sampling can be also merged to one category called selective sampling, because of their similar features [1].

2.2.1 Pool-based Sampling

Regarding pool-based sampling many observations can be gathered at once. This amount of instances is called a pool and has the same meaning as the set S_U which is defined in Section 1.4.3.

The main advantage of this approach is that we are allowed to compute a chosen informativeness measurement on all unlabeled instances and select the most informative one.

Instances are usually chosen sequentially, i.e. one instance is labeled at each iteration and the set of labeled instances S_L is used to retrain the classifier f. However, this process often appears to be expensive and inefficient in applications where training

the classifier f consumes a great amount of time [9]. Therefore, active learning algorithms were invented in order to allow collecting more instances within one iteration. This approach is called batch-mode active learning [10].

Active learning using pool-based sampling was exploited in many applications, e.g. in the text classification [11], in the natural language processing [12] or in the artifacts detection of EEG signals [13].

2.2.2 Stream-based Sampling

Using pool-based sampling is often limited to cases when only a limited memory to store observations is available or when a data source, from which observations are gathered, is changing [2].

An algorithm samples an instance from its distribution P_U and decides if the instance will be accepted as query or rejected. Stream-based sampling is often called sequential active learning because instances are gathered from data source one by one [2].

Stream-based sampling is also used when it is not necessary to keep a whole set S_U and the easier and cheaper way is sampling observations right from their distribution.

Note that the distribution P_U does not need to be known but it is recommended that samples should come from non-uniform and real distributions [2]. If the distribution is uniform then stream-based sampling behaves like query synthesis which will be mentioned in the next section [8].

2.2.3 Query Synthesis

Only real instances are selected as queries in stream-based and pool-based sampling scenarios. Queries are generated by a learner in the query synthesis and are able to look like any instance from the observation space [14].

A learner can construct a query which is unrecognizable to human oracle, as it is often recommended to use query synthesis only for finite problems domains [2, 14]. Nevertheless, the efficient spectral algorithm has also been introduced, and it expands the possibilities to use query synthesis in more complicated settings as well [15].

2.3 Query Strategies

When active learning is adopted, one of the most important question to answer is how the most informative instances will be selected. A large number of them exist in literature [2], so the most common strategies will be mentioned in the following section. Note that x_A^* refers to the best query in accordance with the query strategy framework A.

2.3.1 Uncertainty Sampling

The uncertainty sampling is the most popular and the simplest strategy [2]. Uncertainty sampling was proposed by Lewis et al. [16]. It is suitable mostly for probabilistic models, but it might also be applied to non-probabilistic classifiers [2].

In this approach, the instance, whose label the classifier is least certain of, is queried. Three variants of uncertainty sampling are mentioned: margin uncertainty sampling, entropy sampling and the least confidence strategy [2].

Entropy sampling is often used because of its simplicity [17]. The instance with the biggest entropy is queried:

$$x_E^* = \arg\max_x -\sum_{i=1}^{|Y|} P_{Y|X}(y_i|x) \log P_{Y|X}(y_i|x). \tag{3}$$

For problems with more than two classes, it is suitable to use least confidence strategy where the instance with the smallest posterior probability of the most probable label is sampled [18]:

$$x_{LC}^* = \arg\min_x (\arg\max_y P_{Y|X}(y|x)). \tag{4}$$

This method takes into account only information about the most probable class, as a result margin uncertainty sampling was proposed by Scheffer et al. [19]. The idea behind this strategy is that the algorithm selects the instance with the smallest difference of the highest and the second highest posterior probability.

$$x_{MS}^* = \arg\min_x P_{Y|X}(\bar{y_1}|x) - P_{Y|X}(\bar{y_2}|x), \tag{5}$$

where $\bar{y_1}$ is the most probable and $\bar{y_2}$ the second most probable label of the instance x.

2.3.2 Query by Committee

Query by committee framework have been introduced for the first time by Seung et al. [20]. A committee of c classifiers $C = \{\theta_1, \ldots, \theta_c\}$, which represents consistent hypotheses with labeled instances, are allowed to vote on the labellings of unlabeled observations. The instance, on which classifiers disagree the most, is considered as the most informative and is chosen as the query.

It is often supposed that embracing query by committee scenario results in a proper reduction of the version space [2]. The algorithm must to be able to create different hypotheses and compute some measure of disagreement among them. There have been several methods for measuring the disagreement proposed, e.g. the vote entropy [21] which is defined as following:

$$x_{VE}^* = \arg\max_x \sum_{i=1}^{|Y|} \frac{V(y_i)}{|C|} \log \frac{V(y_i)}{|C|}, \tag{6}$$

where $V(y_i)$ is the number of votes that each label y_i receives.

Other methods include Kullback-Leibler divergence [1, 11] or F-complement, where F-measures between committee members are compared [1, 22].

2.3.3 Expected Model Change

The basic idea behind expected model change is that the learner should query the instance whose labeling would cause the greatest change of the current model.

Settles et al. [23] have introduced a strategy called expected gradient length. This approach is suited for learning using gradient-based methods. The instance, whose addition to the labeled set would cause the greatest change of the gradient, is queried. Expected gradient length approach has been applied to conditional random fields models (CRFs) [12], a similar strategy for CRFs has also been introduced by Vezhnevets et al. [24].

Another algorithm called expected model change maximization was proposed by Cai et al. [25], when it was used for linear and nonlinear regression.

2.3.4 Density-Weighted Strategy

The problems of using uncertainty sampling and query by committee frameworks cover a situation when the instance, about which the classifier is most uncertain (or with the biggest disagreement among committee members), is queried, but it does not sufficiently represent other instances [2]. As a result, Settles et al. [12] proposed the information density framework.

Let $I(x)$ be the informativeness of the instance x given e.g. by uncertainty sampling or query by committee, $s(x_i, x_j)$ be the function describing the similarity between x_i and x_j and $\beta \in \mathbb{R}$ be the weight of the similarity term. Then the algorithm queries an instance for which it holds:

$$x^*_{ID} = \arg\max_x I(x) \times \left(\frac{1}{|S_U|} \sum_{j=1}^{|S_U|} s(x, x_j) \right)^{\beta} \tag{7}$$

2.3.5 Expected Error Reduction

This approach consists of computing error estimates of using $S_L \cup (x, y)$, $x \in S_U$ on remaining unlabeled instances i.e. $S_U \setminus \{x\}$ and querying the instance x^* whose addition to the set S_L will result in the smallest future error [2].

Since true labels of unlabeled instances are not known, they are estimated by using the current classifier. The label of the instance x^* is also unknown, so its expected error is computed as the average over errors computed for all possible labeling weighted by the posterior probability given by the classifier [26].

2.3.6 Variance Reduction

Another approach of selecting the most informative instance consists of variance reduction of the expected error which can be easier and less expensive than reducing the expected error as a whole [2].

Settles et al. has introduced a selection strategy for sequence models based on Fisher information framework [18]. Let $\mathcal{I}_{S_U}(\theta)$ be the Fisher information matrix of the set of unlabeled instances and $\mathcal{I}_x(\theta)$ be the Fisher information matrix of some instance x, both above the model parameters θ. Then the most informative instance derives from:

$$x^*_{FI} = \arg\min_x \operatorname{tr}(\mathcal{I}_{S_U}(\theta)\mathcal{I}_x(\theta)^{-1}), \tag{8}$$

where $\operatorname{tr}(\mathcal{A})$ denotes the trace of matrix $\mathcal{A}$.

2.4 Algorithm

We will describe all the parts of active learning process in this section. Firstly, a description of a general version of active learning using pool-based sampling, which will be used further in this text, will be provided (see Section 2.4.1). Secondly, two new active learning strategies will be introduced (see Section 2.4.2).

2.4.1 General Version

The following pseudocode describes a general version of active learning which has a set of training data TR, the classifier f and variables, which depend on selected query strategy, as inputs.

Algorithm 1 Pool-based active learning

Input: set of training data TR, classifier f, variables var
Output: set of labeled instances S_L, trained classifier f_T

```
[S_A, S_U] = initialize(TR)
S_L = get_label(S_A)
repeat
    f_T = train_classifier(S_L, f)
    x* = query_strategy(S_U, f_T, var)
    (x*, y) = get_label(x*)
    S_U = S_U \ {x*}
    S_L = S_L ∪ (x*, y)
    f = f_T
until terminal_condition = true or S_U = ∅
```

The function *initialize* divides the set of training data TR into the small set S_A and the larger set of unlabeled instances S_U. Then the oracle is asked for labels of instances in the set S_A by using the function *get_label*. Note that the input of this function can be

a single instance or a set of observations. Until the terminal condition is reached (this will be discussed more in Section 2.4.4), the algorithm loops: the classifier f is trained on the labeled set S_L, then data from the set S_U are classified, the most informative instance x^* according to some query strategy is chosen, the function *get_label* returns its label y and x^* is removed from the set S_U, while the pair (x^*, y) is added to the set S_L.

2.4.2 Strategies

Strategies that will be used in this thesis, will be described in this section. First of all, we will remind the margin uncertainty sampling strategy, we have already mentioned in Section 2.3.1, and the simplest version of the density-weighted strategy from Section 2.3.4. Furthermore, strategies called UMDN and UMDL will be proposed.

All strategies in the Algorithm 1 are called by the function *query_strategy*.

Algorithm 2 Margin Uncertainty Sampling Strategy

Input: set of unlabeled data S_U, trained classifier f_T
Output: instance x^*

```
min = ∞
for all instance x ∈ S_U do
    aposterior_prob = get_aposterior_prob(x, f_T)
    subtract the biggest and the second biggest a posterior probability, save result into
    the variable c
    if c ≤ min then
        min = c
        x* = x
    end if
end for
```

The most important part of this algorithm is calling the function *get_aposterior_prob* which returns the conditional probabilities of all classes given x. The sum of those values is equal to one.

The simplest density-weighted method is very similar to uncertainty sampling, the certainty c of an observation is multiplied by the mean distance to all other unlabeled instances. It is given by the value of β how much the certainty is affected by the distance. Let us denote this strategy as DWD*, where * is the value of β.

Density-weighted strategies called UMDN and UMDL will be introduced now. Let us describe UMDN strategy first (see Algorithm 3). Assume that we are able to get l least certain instances by calling the function *get_l_least_certain*. The least certain instances are obtained in the first part of the algorithm, then for each of these instances the nearest labeled observation is found. If the distance between them is larger than the maximal already received distance, the maximal distance is updated and the respective instance is queried. Similarly, as we have done before, let us denote this strategy as UMDN*, where * is the number l, in order to distinguish these strategies with different parameters.

In UMDL strategy (see Algorithm 4), $|Y|$ centroids of labeled instances are computed, with each centroid being the mean value of all labeled instances in the particular class. Certainty c and distance d from the nearest centroid are computed for each instance. Instance, whose product of c and d is the smallest, is queried.

Algorithm 3 UMDN

Input: set of unlabeled data S_U, set of labeled data S_L, trained classifier f_T, number l
Output: instance x^*
  $least\_certain$ = get_l_least_certain(S_U, f_T, l)
  $max = -\infty$
  for all instance $x \in least\_certain$ **do**
    d = get_distance_from_the_nearest_neighbor(S_L, x)
    if $d \geq max$ **then**
      $max = d$
      $x^* = x$
    end if
  end for

Algorithm 4 UMDL

Input: set of unlabeled data S_U, set of labeled data S_L, trained classifier f_T
Output: instance x^*
  $centroids$ = get_centroids(S_L)
  $min = \infty$
  for all instance $x \in S_U$ **do**
    $aposterior\_prob$ = get_aposterior_prob(x, f_T)
    subtract the biggest and the second biggest a posterior probability, save result into the variable c
    d = get_min_distance_from_centroid(x, $centroids$)
    $cd = c \cdot d$
    if $cd \leq min$ **then**
      $min = cd$
      $x^* = x$
    end if
  end for

2.4.3 Initialization

As we are aware of, the problem of how active learning process should be initialized, i.e. how to create the initial set of labeled instances, is not often followed up in the literature. Two different approaches exist: a random and an informative selection.

In the random selection, as the name suggests, instances to query are chosen arbitrarily. This approach is adopted e.g. from Ngai et al. [22].

Attenberg et al. [27] mentioned that researchers of active learning make one or two assumptions. First of them consists of presuming that the initial random selection

will provide a sufficiently accurate model to continue with active learning, the second reflects on the cold start problem – this means it is expected that the error of the classifier is high during several iterations at the beginning, whereas the set of labeled instances is small.

Let us demonstrate the cold start problem caused by the randomized initialization on the small example. Suppose two-dimensional data which we would like to classify into three classes. The margin uncertainty sampling strategy will be used (see Section 2.3.1) and a linear classifier [3]. The active learning process is initialized by the labeling of two randomly chosen instances. The initial situation is depicted in Figure 1.

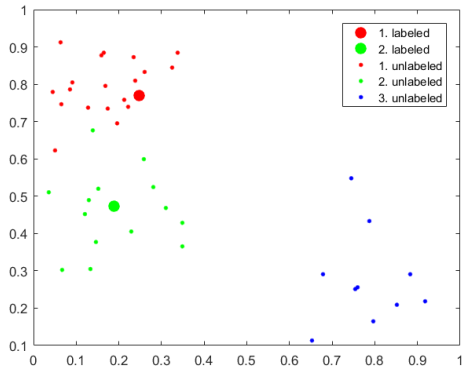


Figure 1: Example of the three-class problem

The labeled set S_L contains two instances belonging to two classes, while the third class is not represented. The initialization and first four steps of active learning are shown in Figures 2a–2e, the corresponding error on training data is depicted in Figure 2f.

The cold start problem covers the initial situation and another two steps of active learning (see Figures 2a–2c). As a result, the set of labeled instances S_L does not contain any observation from the third class, as all data from this class are classified incorrectly, so the overall error is high. If the instance from the third class is added to S_L (Figure 2d), the performance of the classifier is improved. It is remarkable that this task is very simple and the third class was detected early after three iterations, but this problem can persist for a longer time in real world applications, when the task is more challenging or the environment is more complex.

Another technique for initialization is the informative selection of labeled instances. It involves a conscious choice of typical observations by a human annotator or some preprocessing step like e.g. the k-means clustering [3], where the observations closest to the acquired centroids are annotated [28], or the k-menoids algorithm which is based on the similar principle [29].

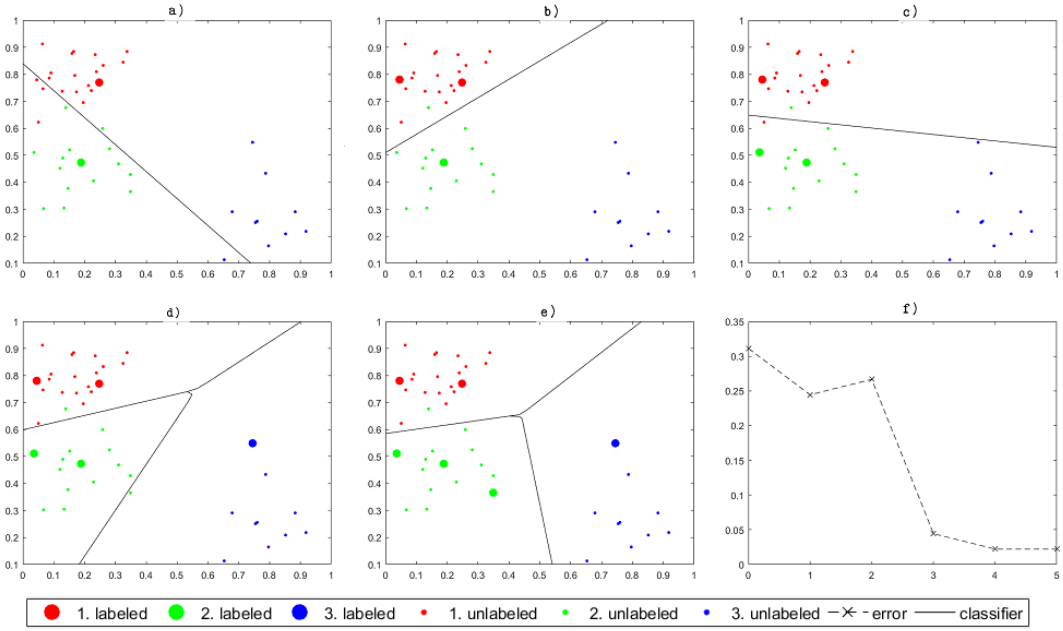
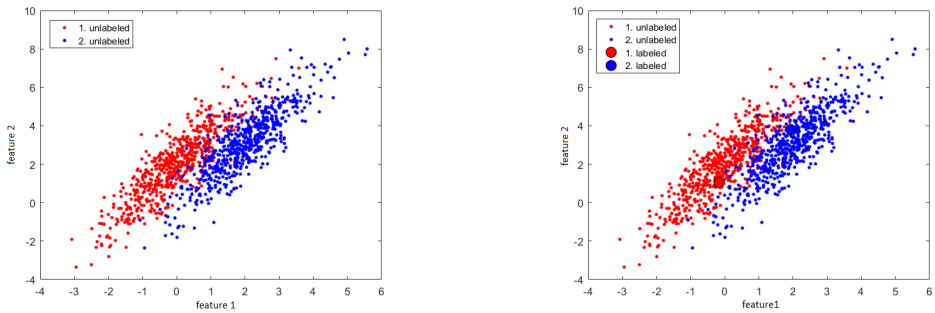


Figure 2: Initialization, four steps of the algorithm and the corresponding error, first five steps of active learning are depicted in Figures 2a–2e, the corresponding error to these situations is shown in Figure 2f

However, there are limitations of using this approach. Since this thesis deals with learning on imbalanced datasets, let us describe one of the problems caused by the informative selection on the following example. Suppose that in two-class problem, data are generated from two two-dimensional Gaussian distributions ($\mu_1 = [0, 2], \sigma_1 = [1, 1.5; 1.5, 3], \mu_2 = [2, 3], \sigma_2 = [1, 1.5; 1.5, 3]$). Assume at first that number of instances in both classes is equal, with data being depicted in Figure 3a. K-means algorithm with $k = 2$ was applied to these data and instances nearest to centroids were labeled. The result is shown in Figure 3b.

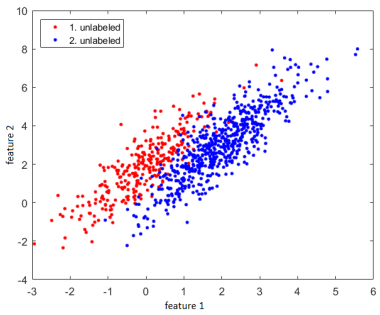


(a) Original dataset with class ratio 1:1

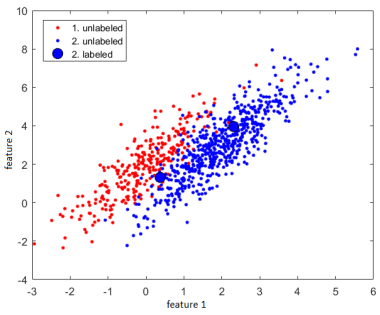
(b) K-means initialization with $k = 2$

Figure 3: Initialization of active learning using the k-means algorithm, the class ratio is 1:1

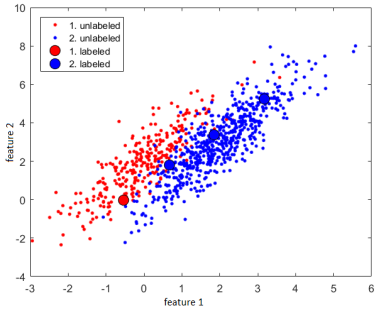
Now, suppose that there are twice less instances in the class 1 than in the second class. If we apply k-means with $k = 2$ as we did in the previous example, all selected instances would belong to the second class (see in Figure 4b). When we increase the number of centroids twice, $k = 4$, then the set of labeled observations contains data from both classes (Figure 4c). The original dataset is shown in Figure 4a.



(a) Original dataset with the class ratio 1:2



(b) K-means initialization with $k = 2$



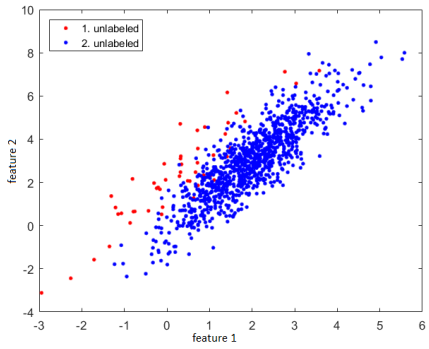
(c) K-means initialization with $k = 4$

Figure 4: Initialization of active learning using the k-means algorithm, the ratio of the first class to the second is 1:2

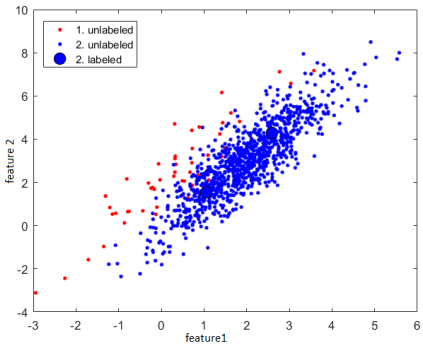
We get similar results if we use the k-means algorithm on instances where the ratio of the first class to the second one is 1:20. Given labeled instances for different k are shown in Figure 5. The original dataset is depicted in Figure 5a.

It is clear that with larger imbalance of classes it is necessary to search for more centroids to initialize active learning in the right way. We can expect that using some preclustering algorithm also causes the cold start problem in more difficult environments or when classes are very imbalanced.

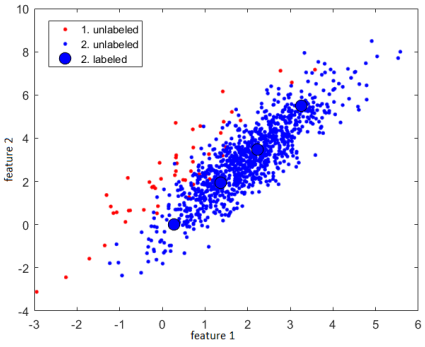
Note the initialization, in which the set of labeled instances S_L contains n randomly picked instances of each class will be used in our setting.



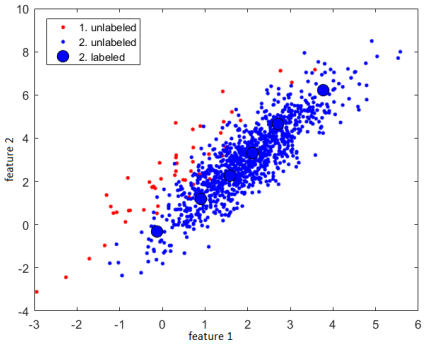
(a) Original dataset with class ratio 1:20



(b) K-means initialization with $k = 2$



(c) K-means initialization with $k = 4$



(d) K-means initialization with $k = 6$

Figure 5: Initialization of active learning using the k-means algorithm, the class ratio is 1:20.

2.4.4 Termination

Several conditions, which are used for termination of active learning, will be mentioned in this section. These conditions are:

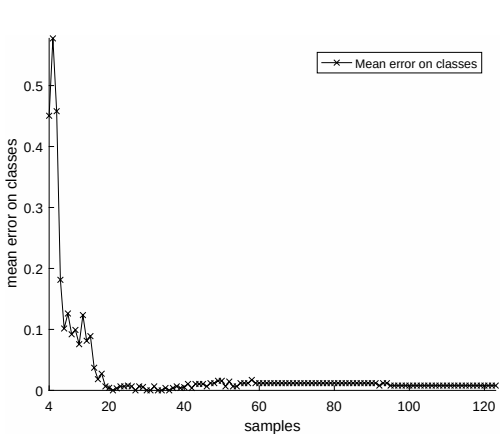
- The number of iterations, which was determined in advance, is reached.
- Required value of a metric on testing data is reached. It is assumed that the chosen metric is computed in every iteration of the algorithm.
- Given percentage of unlabeled instances has bigger certainty value than some threshold. Note that this terminal condition can be used only if uncertainty sampling strategy or its variants are used.
- All instances in the margin are exhausted.

The metric, which we mention in the second bullet point, can be the given level of accuracy, error, F-measure, g-mean etc. These metrics will be discussed in Section 3.1. The terminal condition in the fourth option is also called the early-stopping criterion and it is used in active learning which exploits the SVM classifier for classification.

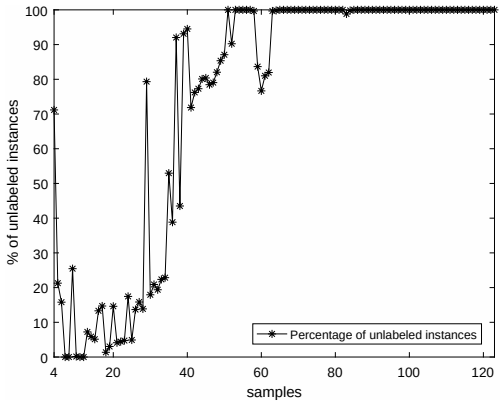
The main idea of this approach is that when all instances that lie along the margin are queried, the active learning process is terminated [30].

The third option will be discussed and shown on the example. Banknote dataset from UCI Machine Learning Repository [31] will be used. It is the two-classes dataset of $1,372$ instances with four features, which was split into two sets of approximately the same size – training and testing set. The Parzen window estimator [32] is used as the classifier, margin uncertainty sampling is used as the query strategy and there are two instances of each class in the initial set of labeled instanced S_L.

The error on testing data is depicted in Figure 6a. Assume that it is sufficient for us to be certain on the label of some instance at least on 75 %. We can see the percentage of unlabeled instances which certainty is bigger than this threshold in Figure 6b.



(a) Error on testing data versus the number of instances which are used for training



(b) Percentage of unlabeled instances, about which the classifier is certain at least of 75 %, versus number of instances for training

Figure 6: Error on testing data of Banknote dataset and the percentage of unlabeled instances whose class certainty is bigger than 75 %

The classifier is confident with at least 75% confidence about any label after forty-six iterations. There is a smaller decrease between fifty-five and fifty-eight iterations. Active learning would be stopped after forty-six iterations with the threshold of certainty set to 0.75 and threshold of percentage of instances set to 1. Note that algorithm would be stopped sooner if we set the mean error on classes to be at most 0.008 as the terminal condition. The process would be terminated after twenty iterations in that case.

3 Evaluation of Active Learning

The current situation of the active learning evaluation and the comparison of different active learning methods will be mentioned in this section. First, we will provide a introduction to several metrics that are commonly used for evaluation of standard machine learning methods.

3.1 Preliminaries

It is necessary to adopt some evaluation techniques to determine how correct is the trained model's performance. Therefore, dataset is often divided into at least two parts – to training data, on which the classifier will learn, and to testing data. The partition of the dataset can be done at random, but it is highly recommended to use stratified 10-fold cross validation in real-world problems to reduce the influence of randomness [33].

3.1.1 Confusion Matrix

Several metrics for the classifier's evaluation on testing data will be discussed in this section. All of them are based on the confusion matrix whose rows correspond to actual classes of the observation and columns to predicted states.

Consider the simplest case, i.e. the observation is either positive (belongs to the class y_+) or negative (belongs to the class y_-). If the observation's actual and predicted classes are both positive (or negative) then the observation is called true positive (negative, respectively). If an observation's actual class is positive and its predicted class is negative, then it is notated as false positive, and contrariwise, if the actual class is negative and the predicted class is positive, then the observation is called false negative (see Table 1). The example, in which the classifier classified 150 instances correctly and four observations incorrectly, is shown in Table 2.

Table 1: Visualization of the simplest confusion matrix

	Predicted y_+	Predicted y_-
Actual y_+	TP	FN
Actual y_-	FP	TN

Table 2: Example of the confusion matrix

	Predicted y_+	Predicted y_-
Actual y_+	74	3
Actual y_-	1	76

As we have mentioned before, this example is only the special case of the confusion matrix. In general, the confusion matrix is also defined for multiclass problems. Consider

a problem where observations are classified to n classes, $Y = y_1, \ldots, y_n$. Let us denote the number of observations whose actual class is y_1 and predicted class is y_n as N_{11}, the number of observations whose actual class is y_1 and predicted class is y_2 as N_{12}, etc. The corresponding confusion matrix is shown in the next table:

Table 3: The confusion matrix for a multiclass problem

	Predicted y_1	$\ldots$	Predicted y_j	$\ldots$	Predicted y_n
Actual y_1	N_{11}	$\ldots$	N_{1j}	$\ldots$	N_{1n}
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\iddots$	$\vdots$
Actual y_i	N_{i1}	$\ldots$	N_{ij}	$\ldots$	N_{in}
$\vdots$	$\vdots$	$\iddots$	$\vdots$	$\ddots$	$\vdots$
Actual y_n	N_{n1}	$\ldots$	N_{nj}	$\ldots$	N_{nn}

It is possible for the specific class y_i to convert the confusion matrix from Table 3 to the two-classes case. Let us define TP_i, FN_i, FP_i and TN_i as follows:

$$TP_i = N_{ii} \tag{9}$$

$$FN_i = \sum_{j=1,\, i \neq j}^{n} N_{ij} \tag{10}$$

$$FP_i = \sum_{j=1,\, i \neq j}^{n} N_{ji} \tag{11}$$

$$TN_i = \sum_{j=1,\, i \neq j}^{n} \sum_{k=1,\, i \neq j}^{n} N_{jk} \tag{12}$$

3.1.2 Metrics

Several metrics, which are used for the evaluation of classifiers, will be introduced in this section. Note that all described metrics take values from $< 0, 1 >$, and they might also be expressed as percentage.

Let us remind the binary case as we proposed in Section 3.1.1 i.e. the goal of the task is to classify the observation as positive or negative, in order to adopt the confusion matrix (Table 1). There are several well-known metrics used for this binary case: the accuracy acc_b, the precision $prec_b$, the sensitivity $sens_b$, the specificity $spec_b$, the miss rate $miss_b$ and the fall-out $fall_b$ of the classifier f [34]. Let us define them as:

$$acc_b = \frac{TP + TN}{TP + TN + FP + FN} \tag{13}$$

$$prec_b = \frac{TP}{TP + FP} \tag{14}$$

$$sens_b = \frac{TP}{TP + FN} \tag{15}$$

$$spec_b = \frac{TN}{TN + FP} \tag{16}$$

$$miss_b = \frac{FN}{FN + TP} \tag{17}$$

$$fall_b = \frac{FP}{FP + TN} \tag{18}$$

values TP, TN, FP and FN are defined in Table 1.

The precision of the classifier f measures how precise is the assignment to the class y_+ [35]. It is defined as the proportion of correctly classified positive observations and all positive classifications.

Since it is often inconvenient to manipulate with two metrics at the same time, several approaches, which combine two metrics into one, have been introduced. One of them is a geometric mean – g-mean which is in general defined as a square root of a two metrics' product [36]. He et al. [35] provided two definitions of g-mean:

$$gm_1 = \sqrt{sens_b \times spec_b} \tag{19}$$

$$gm_2 = \sqrt{sens_b \times prec_b} \tag{20}$$

Other, the more sophisticated metric than g-mean, is the F-measure which is defined for any $\alpha \in \mathbb{R}$, $\alpha > 0$ as follows:

$$F_\alpha = (1 + \alpha^2)\frac{prec_b \times rec_b}{(\alpha^2 \times prec_b) + rec_b}, \tag{21}$$

where rec_b is the recall, another notion for the sensitivity. α is commonly set to one, then the F-measure is called F1-score [37].

We will generalize some of these metrics for a multi-class problem, i.e. a problem where $|Y| \geq 2$.

The accuracy of classifier f is defined as:

$$acc = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n}\sum_{j=1}^{n} N_{ij}}, \tag{22}$$

where TP_i is the number of correctly estimated instances by the classifier f and N_{ij}

corresponds to values in the confusion matrix given by the classifier f described in Section 3.1.1. The same goes for TN_i, FP_i and FN_i, discussed in metrics above.

Let us remind the error of the classifier e which was defined in Section 1.4.1. It holds that the accuracy acc is the complement of the e:

$$e = 1 - acc \tag{23}$$

The precision $prec_i$, the sensitivity $sens_i$, the specificity $spec_i$, the miss rate (also called false negative rate) $miss_i$ and the fall-out (also called false positive rate) $fall_i$ of the classifier f's prediction of the class y_i are defined as:

$$prec_i = \frac{TP_i}{TP_i + FP_i} \tag{24}$$

$$sens_i = \frac{TP_i}{TP_i + FN_i} \tag{25}$$

$$spec_i = \frac{TN_i}{TN_i + FP_i} \tag{26}$$

$$miss_i = \frac{FN_i}{FN_i + TP_i} \tag{27}$$

$$fall_i = \frac{FP_i}{FP_i + TN_i} \tag{28}$$

It is noticeable that the miss rate is complement of sensitivity and also specificity and fall-out are complementary:

$$miss_i = 1 - spec_i \tag{29}$$

$$fall_i = 1 - spec_i \tag{30}$$

Let us define the mean precision on classes $mprec$ and also the mean error on classes me of the classifier as follows:

$$mprec = \frac{\sum_{i=1}^{n} prec_i}{n} \tag{31}$$

$$me = \frac{\sum_{i=1}^{n} miss_i}{n} \tag{32}$$

It is also possible to compute F-measure for a multi-class problem. Micro-averaged and macro-averaged F-measures are often used. [38].

Let us define the micro-averaged F-measure MiF_α as:

$$MiF_\alpha = (1+\alpha^2)\frac{p \times r}{(\alpha^2 \times p) + r} \tag{33}$$

$$p = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n}(TP_i + FP_i)} \tag{34}$$

$$r = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n}(TP_i + FN_i)} \tag{35}$$

Macro-averaged F-measure MaF_α is defined as:

$$MaF_\alpha = \frac{(1+\alpha^2)\sum_{i=1}^{n} \frac{prec_i \times rec_i}{(\alpha^2 \times prec_i) + rec_i}}{n} \tag{36}$$

Micro-averaged F-measure gives priority to the performance of a classifier to the larger of classes and macro-average F-measure to the minority class [38].

3.2 Active Learning Evaluation

In this section we will mention how active learning methods are compared and we will propose new criteria for evaluation of active learning.

3.2.1 Visual Comparison

As we mentioned before, active learning consists in the informative selection of one or more instance to query in each iteration of the algorithm. It is necessary to determine, if active learning will provide better results than if it had be done randomly. The most frequently used technique for evaluation, comparison to random sampling, or comparison of multiple active learning approaches is the visual comparison.

It typically consists of plotting a chosen metric as a function of the number of labeled instances, i.e. the number of instances which were used for training of the classifier. The metric is chosen according to the application, e.g. F-measure [1, 12], Area Under Received Operating Characteristic Curve [23, 27], accuracy [13, 28, 39] or the g-mean [40]. We will mostly use the error, the miss rate of individual classes and the mean error of classes.

Due the fact that random sampling and also some active learning methods are stochastic, it is necessary to run the process several times in order to get more reliable results and use the average of the chosen metric through all runs of the algorithm for the comparison [23]. Let us show it on the example. We can see the comparison of active learning using the margin uncertainty strategy with random sampling launched on the

Banknote dataset in Figure 7, where is shown the mean error of classes depending on the number of labeled instances, results curve are achieved by averaging of 150 runs. We can see in Figure 7 that the active learning distinctly dominates over random sampling in all expect the first few iterations. It should be noted that it is not always straight-forward to make such clear conclusion about dominance of one algorithm.

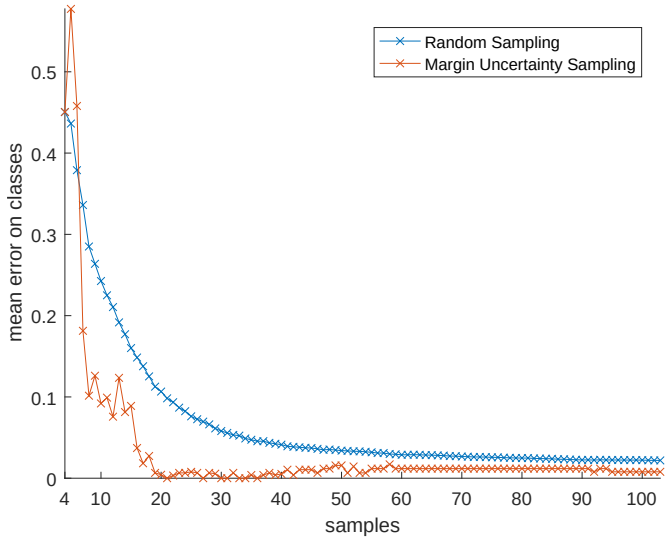


Figure 7: Mean error on classes on testing data of Banknote dataset of random sampling and margin uncertainty sampling

3.2.2 Proposed Evaluation Criteria

Several criteria which can be embraced for evaluation of active learning and comparison of methods will be proposed. Let us denote that, as in the visual comparison part, it is necessary to compare an active learning method to random sampling in order to figure out if active learning leads to better results than a random sequential choice of instances.

Reaching Threshold

This criterion is the simplest one of all suggested criteria. It consists in defining the threshold (of the error, the mean error on classes, the accuracy, etc. on testing data) and determining how many iterations algorithm needs to reach this threshold. Then, all method are sort out in ascending order, the algorithm that has reached the threshold first is the best.

Let us show this criterion on example. Consider that if three methods are compared, the number of iterations is 10 and the corresponding error of all methods is shown in Table 4. Let us assume that the threshold is set at 0.15. Method 1 reaches this threshold after eight iterations, method 2 after three iterations and method 3 after two iterations.

Table 4: Error of three methods in each iteration

	Number of iterations									
	1	2	3	4	5	6	7	8	9	10
Method 1	0.84	0.32	0.40	0.28	0.27	0.21	0.21	0.15	0.12	0.15
Method 2	0.66	0.28	0.15	0.14	0.13	0.06	0.05	0.05	0.04	0.05
Method 3	0.41	0.12	0.12	0.31	0.28	0.27	0.26	0.19	0.18	0.14

This approach is straightforward, easy to implement and obviously reduces the number of labeled instances which are needed in order to train the classifier. The biggest disadvantage of this criterion lies in the choice of the threshold.

If active learning is compared to random sampling, we can interpret this criterion as the question: "How many instances to training would we save if we exploited active learning?" The answer is the result of this criterion, when active learning is adopted and subtracted from the number of training instances.

Rank Comparison

Rank of all methods according to the chosen metric is computed in each iteration, i.e. if the mean error on classes is chosen as the metric, then the method with the smallest mean error on classes has the rank 1, the method with the second smallest mean error on classes has the rank 2, etc. The mean rank (computed over all iterations) of a method is computed for the overall comparison of all methods. Alternatively, the most frequent rank (mode) of the method can be used.

Let us remind the previous example and corresponding values of the error of the three methods in each iteration (see in Table 4). Matching ranks are shown in Table 5.

Table 5: Ranks of three methods in each iteration

	Number of iterations											
	1	2	3	4	5	6	7	8	9	10	Mean	Mode
Method 1	3	3	3	2	2	2	2	2	2	3	3	2
Method 2	2	2	2	1	1	1	1	1	1	1	1.3	1
Method 3	1	1	1	3	3	3	3	3	3	2	2.1	3

Method 2 is the best of all methods according to both criteria, method 3 has the second best mean rank, however the most frequent value of its rank is 3, and the most frequent rank of the method 1 is 2, but the mean rank is 3.

The biggest advantage of this evaluation is that no external input is required as the threshold used in the previous evaluation technique and each method is directly compared to others. As we have shown in the abovementioned example, the mean of ranks and the mode of ranks produce different results so the decision which method is better rests with the user.

Breaking Point

The main idea of this evaluation criteria is in finding a point (iteration), after which a certain percentage of metric values would be bigger (e.g. if accuracy is adopted) or smaller (e.g. if error is computed) than the given threshold.

Let us consider the example again, the corresponding mean error of the three methods in each iteration is shown in Table 4 above. Let us assume that the point, after which 95 % of values of the error would be less than or equal 0.15 is being found. According to this criterion, method 2 is the best, as 95 % of error values are equal to 0.15 or smaller after three iterations. Method 1 reaches this point after eight iterations, method 3 achieves this after 10 iterations.

Table 6: Percentage of values which are less than or equal to 0.15 in each iteration. Iterations after which 95 % of values are less than or equal to 0.15 are bold.

	Number of iterations									
	1	2	3	4	5	6	7	8	9	10
Method 1	0.30	0.33	0.38	0.43	0.50	0.60	0.75	**1**	1	1
Method 2	0.80	0.89	**1**	1	1	1	1	1	1	1
Method 3	0.30	0.33	0.25	0.14	0.17	0.20	0.25	0.33	0.50	**1**

Summarization

Let us summarize the results of our example in Table 7. The best results are marked in bold, the second best results are displayed in italics.

Table 7: Summarization of determined criteria

	Reaching the threshold	Mean Rank	Mode of Ranks	Breaking Point
Method 1	in 8 it.	3	*2*	*in 8 it.*
Method 2	*in 3 it.*	**1.3**	**1**	**in 3 it.**
Method 3	**in 2 it.**	*2.1*	3	in 10. it.

Method 2 reaches the best results, as it proves itself best in three out of four criteria. It is not easy to compare method 1 and 3, as method 1 is the second best in two out of four criteria, whereas method 3 is the best in the reaching the threshold criterion and the second best in the mean rank criterion.

3.3 Experiments

Several experiments on different datasets were conducted in order to evaluate some of the active learning strategies. We compare random sampling to these strategies: margin uncertainty sampling, the density-weighted strategy using the mean distance

to all instances as the similarity function with two different parameters and algorithms UMDN10 and UMDL from Section 2.4.2).

3.3.1 Data

Five datasets from UCI Machine Learning Repository [31] are used. We can see descriptions of the used datasets in Table 8, whereas the number of classes is stored in the column Classes, the number of features is in the column Features, the column Training instances denotes the number of instances, which are available to training, and the similar goes for the column Testing instances.

Table 8: Description of datasets

Dataset name	Classes	Features	Training instances	Testing instances
Banknote	2	4	684	684
Pima	2	8	384	384
Spambase	2	57	2281	2279
Statlog	7	19	1152	1151

3.3.2 Results

Selected stochastic algorithms (random sampling and margin uncertainty sampling) were launched 150 times. The number of iterations is 150. The mean error on classes was chosen as metric, the threshold for reaching the threshold and breaking point evaluation criteria was determined as the mean error on classes on testing data, when the classifier was trained on whole training data. The given percentage for breaking point criterion is set to 95 %.

All results are shown in the following tables, and one table corresponds to one dataset. The column RtT stands for the reaching the threshold criterion, MeanRC for the mean rank comparison, ModeRC for the mode of ranks comparison, BP for the breaking point comparison. The row RS stands for random sampling, MUS for margin uncertainty sampling, DWD1 for density-weighted method with distance function weighted by 1, DWD0.5 for density-weighted method with distance function weighted by 0.5. The best results are highlighted in bold and the second best in italics.

Table 9: Evaluation criteria for Banknote dataset, threshold is set to 0.06.

Methods	TRtT	MeanRC	ModeRC	BP
RS	27	5.27	*2*	21
MUS	**13**	**1.43**	5	**6**
DMD1	**13**	*2.38*	3	*7*
DWD0.5	**13**	4.26	4	*7*
UMDN10	*14*	2.68	4	*7*
UMDL	**13**	4.97	**1**	**6**

Table 10: Evaluation criteria for Pima dataset, threshold is set to 0.3.

Methods	TRtT	MeanRC	ModeRC	BP
RS	75	*2.04*	*2*	*94*
MUS	**31**	**1.39**	**1**	**66**
DMD1	116	4.38	5	115
DWD0.5	x	5.21	6	x
UMDN10	*72*	3.36	3	146
UMDL	101	4.62	4	139

Table 11: Evaluation criteria for Spambase dataset, threshold is set to 0.27.

Methods	TRtT	MeanRC	ModeRC	BP
RS	84	3.10	6	105
MUS	128	3.72	*3*	127
DMD1	*35*	*2.15*	**1**	**29**
DWD0.5	**30**	4.09	4	143
UMDN10	x	5.85	4	x
UMDL	37	**2.07**	5	*33*

Table 12: Evaluation criteria for Statlog dataset, threshold is set to 0.15.

Methods	TRtT	MeanRC	ModeRC	BP
RS	124	4.17	*3*	121
MUS	**80**	**2.15**	**2**	**82**
DMD1	103	3.39	4	102
DWD0.5	90	*2.45*	5	88
UMDN10	*87*	4.17	6	*84*
UMDL	122	4.67	*3*	122

The similar results on Banknote dataset are achieved by all active learning methods, margin uncertainty sampling is the best in three out of four criteria (TRtT, MeanRC and BP) and the same goes for UMDL algorithm (in criteria TRtT, ModeRC and BP). All active learning methods are better than random sampling expect of ModeRC criterion, where random sampling reaches the second best result.

Margin uncertainty sampling is a clear winner on Pima dataset, random sampling reaches the second best results in three out of four criteria, UMDLN10 strategy achieves the second best result in TRtT criterion. DWD0.5 strategy does not converge in 150 iterations, so it is overall at the sixth place.

DWD1 strategy is the best in two and the second best also in two criteria on Spambase dataset, DWD0.5 method reaches the best result in TRtT criterion and UMDL strategy in MeanRC criterion. Margin uncertainty sampling fails against random

sampling in three out of four criteria, UMDLN10 strategy does not converge in 150 iterations.

Margin uncertainty sampling reaches the best results in all criteria on Statlog dataset, UMDN10 is the second best in two criteria, UMDL and random sampling in one criterion.

It is very important to determine whether an active learning method is better than random sampling. The number we can see denotes when active learning methods reach better results than random sampling in Table13.

Table 13: Summarization of how many times active learning methods are better than random sampling in each criterion on all datasets.

Methods	TRtT	MeanRC	ModeRC	BP	Sum
MUS	3	3	3	3	12
DMD1	3	3	1	3	10
DWD0.5	3	2	1	2	8
UMDN10	3	1	1	2	7
UMDL	3	2	2	2	9

Margin uncertainty is better than random sampling twelve times out of sixteen cases, DWD1 strategy ten times and UMDL strategy nine times. DWD0.5 is better only if the half of cases and UDMDN10 is better only seven times.

3.4 Discussion

We have proposed four evaluation criteria as an alternative to the most used method which is the visual comparison.

As we saw in Section 3.3.2, results of these evaluation criteria do not often match each other. It is difficult to determine which evaluation criterion provides the most reliable performance, so the more thorough analysis is needed. Despite this fact, we can recommend using these evaluation criteria in cases when it is not possible to evaluate which method is better according to the visual comparison.

Proposed criteria were used to identify in how many criteria some method reaches better results than random sampling. We used five methods on four different datasets, margin uncertainty sampling achieved the best results of all active learning methods compared to random sampling. It is still not possible to determine if margin uncertainty sampling is better than other proposed methods,as more datasets with the different number of instances, classes and features will be needed in order to do so.

4 Active Learning on Imbalanced Data

In this section we will provide the introduction to what imbalanced data are and what techniques are commonly used for handling the class imbalance.

4.1 Introduction to Class Imbalance

Machine learning can run into problems which are caused by the imbalance of classes. It often concerns real-world applications like biomedical, financial or security data analysis, text classification, face recognition, anomaly detection, etc. [35, 41, 42]

4.1.1 Characterization of Class Imbalance

The characterization of imbalanced datasets occurs when the number of instances in each class is not approximately equal, but some class (or classes) contains much more instances than other classes [41]. Class imbalance is often described by using the ratio of numbers of instances in the most common class to the rarest one. According to He et al. [35], if the ratio is 1:2, then datasets are marginally imbalanced, whereas they are modestly imbalanced with the ratio 1:10 and extremely imbalanced with the ratio 1:1,000.

The problem of imbalanced data can be described in detail on the example. Consider a dataset which contains data belonging to two classes, e.g. the class y_1 includes 1,000 instances and the class y_2 10 instances. The corresponding confusion matrix is shown in the following table.

Table 14: Example of the confusion matrix of the imbalanced problem

	Predicted y_1	Predicted y_2
Actual y_1	1,000	0
Actual y_2	10	0

We will show the so-called accuracy paradox on this example [43]. Although this is a binary case, generalized metrics for a multi-class problem will be used because it does not hold that one class is positive and the other is negative. A standard classification method minimizing an error f can classify all instances as class y_1, so the accuracy of the prediction acc_b is 99 %, but it is clear that the classifier does not solve the given problem. It seems to be more natural to use another metrics in this case.

As we mentioned in Section 3.1.2, the micro-averaged F-measure MiF gives the priority to the performance on the majority class and the macro-averaged F-measure MaF on the minority class. Results correspond with this assumption, it holds that $MiF = 0.99$ and $MaF = 0.5$.

As we mentioned before, as learning on skewed datasets is the common challenge, so a large amount of approaches to deal with it exists.

4.1.2 Sampling Methods

Sampling methods are used for creating the dataset with relatively balanced classes and two kinds of them exist: undersampling and oversampling [44].

In the case of undersampling, some instances of the majority class are removed. This process can be done randomly, which is easy for implementation but is also able to cause problems by removing informative instances, or there are techniques which removes only the redundant noise [35], e.g. modifications of the k-nearest neighbor algorithm [35, 45] or using of the soft margin SVM classifier [45].

Random oversampling consists of repeatedly copying of the minority class instances. Since this approach can lead to overfitting [35], Chawla et al. [46] have introduced SMOTE algorithm, which is able to create new synthetic instances. The main idea of SMOTE is the following: until the terminal condition is reached (generally as long as the classes are imbalanced), the randomly chosen instance from the minority class is chosen. Then, k nearest neighbors to the selected instance are computed and one is randomly chosen. Values of the new instance's features are given by the difference between selected instances and its neighbor's values of features multiplicated by the number between zero and one.

Hybrid techniques, which are the combination of SMOTE algorithm and an undersampling method, are also implemented [44].

4.1.3 Ensemble Methods

The main idea of ensemble methods is training of a group of classifiers and combining their decisions [47]. The crucial condition in using ensembles is that classifiers do not need to be accurate, but have to be diverse [42].

Several approaches are used for building ensembles, e.g. boosting algorithms such as Adaboost, Random Forest [35] or SMOTEboost (which is the combination of a boosting technique and SMOTE algorithm) [46].

4.1.4 Cost Sensitive Methods

In cost sensitive methods, the loss function is exploited which will largely penalize if the minority class instances are misclassified [35].

There are cost sensitive variants of favorite machine learning approaches e.g. a cost sensitive version of the SVM algorithm [35] or cost sensitive ensemble methods [41].

4.1.5 Insensitive Classifiers

Hoens et al. in [35] have mentioned the possibility of using classifiers which are in general skew-insensitive. One of them is naive Bayes because the conditional probability of the observation x_i given the class y and the probability of the class y are computed before the conditional probability of the class y given the observation x_i [35].

Decision trees are also skew-insensitive, e.g. Cieslak et al. [48] have proposed Hellinger distance decisions trees whose splitting criterion is Hellinger distance.

4.2 Class Imbalance and Active Learning

As we showed in Section 2.4.3, highly skewed datasets can cause problems during the initialization when it is impossible for the algorithm to find sufficient number of instances of all classes. Moreover, Attenberg et al. [35] have mentioned that active learning on extremely imbalanced data can suffer from the cold start problem because it is hard to find informative instances of the minority class due to their rareness.

In any case, opinions on active learning on imbalanced datasets differ. On the one hand, active learning can be used as the sampling method (see Section 4.1.2), on the other hand, it is often mentioned that active learning on skewed data can be problematic and there are approaches which enable active learning to learn a sufficiently accurate classifier [35]. We will explore both mentioned aspects in this section.

4.2.1 Active Learning as the Sampling Method

Attenberg et al. [35] and also Tomanek et al. [1] have mentioned that active learning can be effective on not extremely imbalanced datasets because it is able to choose the most informative instances.

The motivation for using active learning as the sampling strategy is that the final set of labeled instances S_L is often more balanced than the initial pool [1]. It is supposed that it is more sufficient to use active learning to undersampling rather than some random selection of instances because random undersampling instead of active learning can discard the most informative observation.

Ertekin et al. [30] has proposed the VIRTUAL algorithm which combines active learning using SVM for classification and the SMOTE algorithm. The idea of the VIRTUAL algorithm is the following: When active learning selects the instance which comes from the minority class, then SMOTE is used, i.e. one of the k nearest neighbors of the selected observation is chosen and used for the creation of the new instance.

Zhu et al. [39] have suggested another active learning algorithm which combines both undersampling and oversampling. The set of labeled instances is resampled in each iteration of active learning; either the majority class is undersampled, or the minority class is oversampled by using the algorithm called BootOS, which iterates through all instances of the minority class and creates new instances as the combination of their neighbors.

4.2.2 Skew-sensitive Methods of Active Learning

Skew-sensitive methods of active learning in general prefer instances from minority class, so the final set of labeled instances S_L would be more balanced and its performance in the imbalanced setting would be better [35].

The favorite approach to solve this problem is the utilization of the SVM classifier [1, 35, 40]. SVM-based active learning seems to be very effective without any other skew-sensitive modification of the algorithm.

Let remind ourselves that the ensemble methods which we mentioned in Section 4.1.3 and which are in many cases used for imbalanced datasets. They consist of learning ensemble of classifiers whose combination results in better performance than using only one classifier. The similarity between ensemble methods and query by committee strategy is obvious, therefore it seems to be promising to use query by committee strategy with sufficiently diverse committees to handle the class imbalance problem.

Tomanek et al. [1] has proposed a method, which is the combination of the vote entropy (see in Section 2.3.2) and the criterion of class distribution. The instance is queried for which holds:

$$x^*_{VEU} = \arg\max_x \sum_{i=1}^{|Y|} b_i \frac{V(y_i)}{|C|} \log \frac{V(y_i)}{|C|}, \tag{37}$$

where b_i is the weight of class y_i corresponding to its importance. The larger value of b_i denotes the larger tendency to query instances which are classified as belonging to the class y_i.

Another option to deal with the class imbalance is using density-weighted strategies (see in Section 2.3.4). The advantage of using density-weighted strategies is the ability to combine one criterion given by the standard query strategy and another one representing class distribution [1].

Attenberg et al. [35] have mentioned several density-weighted heuristics. One of them is mainly used for text classification [28] and it is the combination of entropy uncertainty sampling and the average cosine similarity of k nearest neighbors:

$$x^*_{Ecos} = \arg\max_x \sum_{i=1}^{|Y|} \mathrm{ca}(\mathrm{knn}(x))\, P_{Y|X}(y|x) \log P_{Y|X}(y|x), \tag{38}$$

where $knn(x)$ is the function which returns k nearest neighbors of x and $ca()$ is the function which computes the average cosine similarity of them.

There are also methods which cover only the density of information and not standard active learning strategy. One of them is a method called sampling by clustering, in which the whole space is clustered and the nearest instances to centroids are sampled [28].

It is also possible to adopt sampling techniques which we mentioned in the previous section.

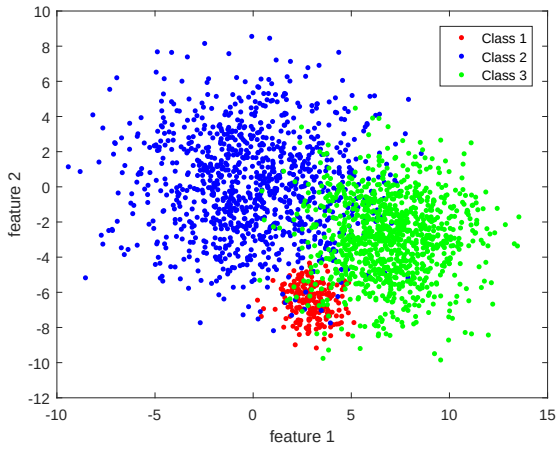
4.3 Experiments

4.3.1 Synthetic Data

We have used two different datasets which were generated from multivariate Gaussian distributions. We have already encountered the first of them in Section 2.4.3 and it consists of two two-dimensional Gaussian models: instances of class 1 are generated with parameters $\mu_1 = [0, 2]$, $\sigma_1 = [1, 1.5; 1.5, 3]$ and instances of the second class with parameters $\mu_2 = [2, 3]$, $\sigma_2 = [1, 1.5; 1.5, 3]$. To observe how the class imbalance influences the performance of active learning, the size of the first class differs in each setting – it is either 1,000, 500, 200 or 50. Number of instances of the class 2 is 1,000 for all settings.

The second dataset is made up of three two-dimensional Gaussians: The first one is generated with parameters $\mu_1 = [3, -6.5]$, $\sigma_1 = [1, 0; 0, 1]$, the second one with parameters $\mu_2 = [7, -3]$, $\sigma_2 = [5, 0; 0, 5]$ and the third one with parameters $\mu_3 = [0, 0]$, $\sigma_3 = [10, 0; 0, 10]$. The number of instances of classes 2 and 3 is always 1,000, the size of the first class differs: It is either 1,000, 500, 200, 100 or 50. We can see the proposed datasets with class sizes 20, 1,000 and 1,000 in the Figure 8.

Figure 8: Synthetic Dataset 2



We chose these two datasets because they are in general simple and obtained results are easily interpretable. Additionally to these reasons, it was necessary to use dataset which contains more than two classes, because we expect that it will cause more difficulties.

4.3.2 Influence of Imbalance on Active Learning

We have launched margin uncertainty sampling and also random sampling on proposed datasets to figure out how class imbalance influences active learning, if at all. The initial set of labeled instances S_L contained two instances of each class, number of

iterations was 1,000 and every algorithm was repeated ten times to reduce the influence of randomness.

We can see values of individual classes miss rates after 10, 100 or 1,000 iterations for the first dataset in Table 15 (for margin uncertainty sampling) and in Table 16 (for random sampling).

Table 15: Mean miss rate of individual classes after 10, 100 and 1,000 iterations with different class imbalance of training data, strategy: margin uncertainty sampling.

	After 10 it.		After 100 it.		After 1,000 it.	
Ratio	$miss_1$	$miss_2$	$miss_1$	$miss_2$	$miss_1$	$miss_2$
1:1	0.261	0.133	0.124	0.059	0.074	0.064
1:2	0.250	0.054	0.150	0.041	0.088	0.045
1:5	0.429	0.032	0.174	0.034	0.087	0.061
1:10	0.366	0.037	0.237	0.021	0.049	0.110
1:20	0.394	0.049	0.256	0.025	0.050	0.241

Table 16: Mean miss rates of individual classes after 10, 100 and 1,000 iterations with different class imbalance of training data, strategy: random sampling, dataset: 1.

	After 10 it.		After 100 it.		After 1,000 it.	
Ratio	$miss_1$	$miss_2$	$miss_1$	$miss_2$	$miss_1$	$miss_2$
1:1	0.240	0.173	0.090	0.086	0.073	0.064
1:2	0.390	0.087	0.108	0.080	0.078	0.058
1:5	0.636	0.041	0.159	0.067	0.081	0.066
1:10	0.501	0.063	0.182	0.053	0.101	0.057
1:20	0.451	0.075	0.215	0.079	0.106	0.072

There is not any obvious significant difference when random sampling is used, but it seems that larger class imbalance influences active learning, but, against expectations, the error on the majority class increases.
We can see that values of individual classes miss rate after 10, 100 or 1,000 iterations for the second dataset in Table 17 (for margin uncertainty sampling) and in Table 18 (for random sampling).

Table 17: Mean miss rates of individual classes after 10, 100 and 1,000 iterations with different class imbalance of training data, strategy: margin uncertainty sampling.

	After 10 it.			After 100 it.			After 1,000 it.		
Ratio	$miss_1$	$miss_2$	$miss_3$	$miss_1$	$miss_2$	$miss_3$	$miss_1$	$miss_2$	$miss_3$
1:1	0.037	0.718	0.058	0.020	0.112	0.186	0.008	0.111	0.174
1:2	0.166	0.404	0.026	0.009	0.150	0.144	0.012	0.105	0.175
1:5	0.006	0.480	0.095	0.011	0.183	0.147	0.021	0.110	0.150
1:10	0.039	0.271	0.064	0.090	0.162	0.076	0.035	0.107	0.137
1:20	0.072	0.394	0.017	0.022	0.191	0.023	0.277	0.089	0.091

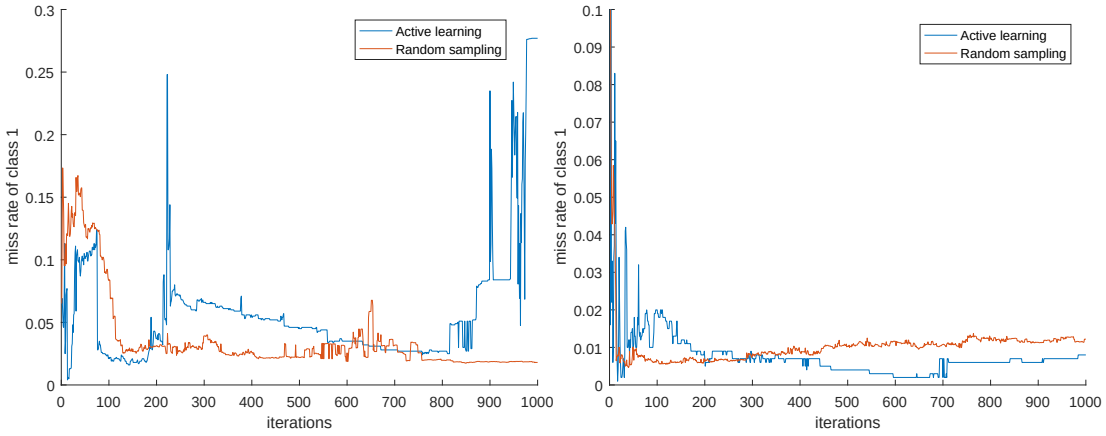
Table 18: Mean miss rates of individual classes after 10, 100 and 1,000 iterations with different class imbalance of training data, strategy: random sampling.

	After 10 it.			After 100 it.			After 1,000 it.		
Ratio	$miss_1$	$miss_2$	$miss_3$	$miss_1$	$miss_2$	$miss_3$	$miss_1$	$miss_2$	$miss_3$
1:1	0.045	0.544	0.202	0.007	0.177	0.142	0.012	0.143	0.121
1:2	0.022	0.494	0.176	0.009	0.175	0.139	0.014	0.138	0.123
1:5	0.043	0.442	0.156	0.012	0.172	0.131	0.023	0.139	0.111
1:10	0.019	0.433	0.148	0.039	0.175	0.128	0.023	0.139	0.112
1:20	0.097	0.310	0.038	0.084	0.145	0.040	0.018	0.124	0.048

Let us denote that active learning reached better results in early stages of the algorithm (after ten iterations) than random sampling. Otherwise, there is no obvious difference between them, instead of results obtained on the dataset with the largest class imbalance. The miss rate of the minority class (class 1) decreases, when random sampling is used, but te miss rate of the minority class after 1,000 iterations, when active learning is exploited, is high (0.277).

We will depict the miss rate of class 1 on dataset with the imbalance ratio 1:20 for both active learning and random sampling to see, if there is an evident difference between them. It is shown in Figure 9a. We can see miss rate of individual classes on dataset with equal class imbalance in Figure 9b for the comparison.

It is shown in Figure 9a that the miss rate of the minority class gradually decreases when the random sampling is adopted. On the other hand, when active learning is used, the miss rate of the minority class is large: between 210 and 574 iterations, and increases after 811 iterations. We can see that when classes are balanced (Figure 9b), then the miss rate of the minority class decreases and when active learning is used, it slightly increases after 700 iterations. When random sampling is used, the error of the minority class is almost constant.



(a) Dataset with the class ratio 1:20

(b) Dataset with the class ration 1:1

Figure 9: Miss rate of the minority class on the dataset with the class imbalance ratio 1:20 and on the dataset with the class imbalance ratio 1:1 for random sampling and active learning

4.3.3 Active Learning Methods for Imbalanced Data

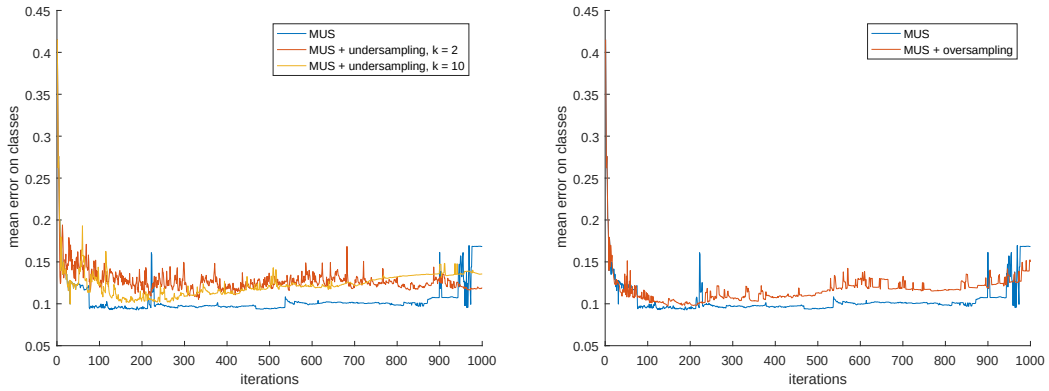
As apparent from the previous section, the favorite technique to address active learning on skewed datasets is using of the query by committee strategy and the SVM classifier. Tomanek et al. [1] and Ertekin et al. [30] have already dealt with this setting, we decided to focus on active learning using the margin uncertainty sampling strategy.

As we saw in the previous section, the extreme class imbalance on the second dataset causes the high miss rate of the minority class, so we will perform several experiments to figure out if some method that deals with the class imbalance can help.

The variant of SMOTE algorithm as oversampling method will be utilized. Every time when instance of the minority class is queried, another labeled instance also belonging to the minority class is randomly chosen and its values of features are subtracted from queried instance's features a multiplied by random number from the range between 0 and 1 in order to create a new instance which is added to the set of labeled instances S_L.

An undersampling method will be also used. When the number of instances belonging to the minority class is k times smaller than number of instances in the majority class, then the majority class is undersampled using k-means algorithm on majority class instances, k centroids replace these instances in the set of labeled observations.

We can see mean error on classes of standard margin uncertainty sampling and of margin uncertainty sampling, when the undersampling method is adopted with $k = 2$ and $k = 10$ in Figure 10a, and when oversampling is used in Figure 10b. The mean error on classes is the average of ten runs.



(a) MUS strategy with undersampling methods (b) MUS strategy with oversampling

Figure 10: Mean error on classes when undersampling methods with $k = 2$ and $k = 10$ are adopted or when oversampling is used, MUS denotes margin uncertainty sampling

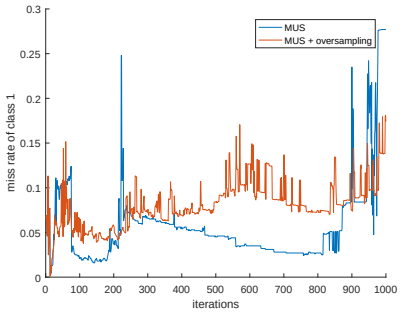
It is shown in Figure 10b that when k is small, then it causes oscillation of the mean error on classes. When k is bigger (e.g. $k = 10$), then the mean error on classes does not oscillate but is still higher than when no undersampling is utilized. The same goes for the case when the oversampling method is adopted: The mean error on classes is bigger than when oversampling is used, on the other hand, the error is not so high around the iteration 1,000.

It seems to be problematic to use undersampling, so we will focus more on oversampling. We can see error on each class depicted in Figure 11.

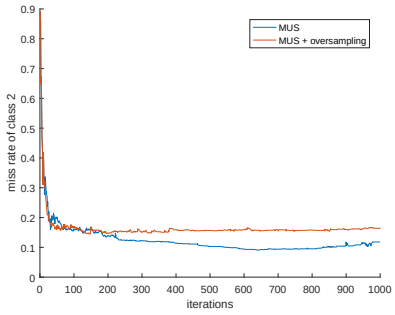
The miss rate of class 3 is smaller when the oversampling method is used, but it is overall bigger than miss rates of class 1 and class 2.

As we mentioned in Section 4.2.2, density-weighted methods are often utilized to handle the class imbalance problem. We will use DWD1, UMDN10 and UMDL methods without and with oversampling and compare them to the margin uncertainty sampling strategy.

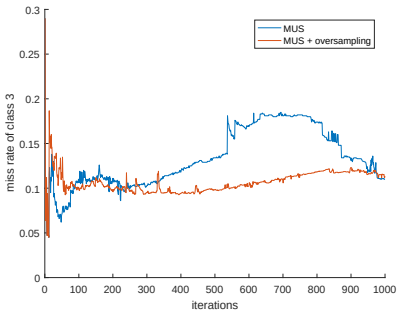
As we can see in Figure 12, where mean errors on classes of random sampling, margin uncertainty sampling and all used density-weighted methods are depicted, a clear decision which method leads to better result could not be drawn, so we will utilize evaluation criteria proposed in Section 3.2.2



(a) Miss rate of class 1



(b) Miss rate of class 2



(c) Miss rate of class 3

Figure 11: Miss rate of individual classes when only margin uncertainty sampling is used and when margin sampling together with oversampling is adopted

.

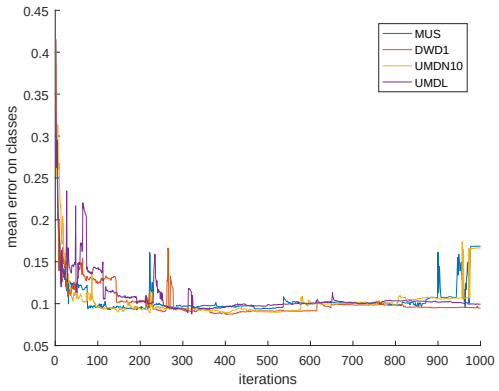


Figure 12: Mean error on class of margin uncertainty sampling strategy and three density-weighted methods: DWD1, UMDN10 and UMDL.

Achieved results are shown in Table 19. We set the threshold to 0.11, because of its estimation from Figure 12. The abbreviation OS means oversampling. The best value of a criterion is bold, the second best is in italics.

Table 19: Evaluation criteria for the second synthetic dataset with threshold set to 0.11.

Methods	TRtT	MeanRC	ModeRC	BP
RS	109	3,48	3	136
MUS	*31*	3,77	*2*	908
DWD1	**30**	**2,54**	**1**	*115*
UMDN10	38	*2,97*	5	**38**
UMDL	113	4,41	2	132
MUS + OS	216	8,38	8	982
DWD1 + OS	38	6,44	9	x
UMDN10 + OS	124	6,01	7	x
UMDL + OS	53	7,01	6	x

The clear winner is DWD1 strategy, which reaches the best results in three out of four criteria and is the second best in the remaining criterion. UMDN10 strategy is the best in one criterion and the second best also in one criterion, margin uncertainty sampling reaches the second best results in two criteria.

All methods with oversampling have failed, as their performances are always worse than their versions without oversampling.

Similarly as we did before, we will compare the results of all active learning methods to random sampling's ones. We can see the number of criteria in which the active learning method was better than random sampling in Table 20. OS denotes oversampling.

Table 20: Summarization of number of criteria in which the active learning method was better than random sampling

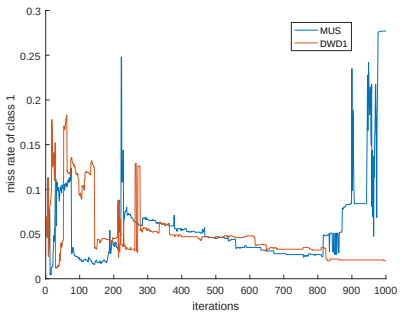
Methods	Number of criteria
MUS	3
DMD1	4
UMDN10	3
UMDL	2
MUS + OS	0
DWD1 + OS	0
UMDN10 + OS	0
UNDL + OS	0

Our conclusion has been confirmed. DWD1 strategy is better than random sampling in all criteria, UMDN10 and margin uncertainty strategies in three out of four criteria, UMDL strategy in two criteria and methods with oversampling in none.

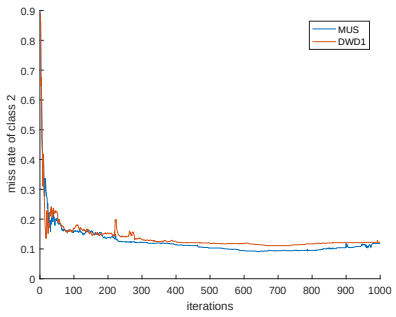
Because DWD1 strategy turns out to be the best choice on the second synthetic dataset, let us compare miss rates of individual classes of provided by DWD1 and margin uncertainty sampling strategies. Graphs are shown in Figure 13.

The miss rate of class 1, when margin uncertainty sampling is used, is smaller than

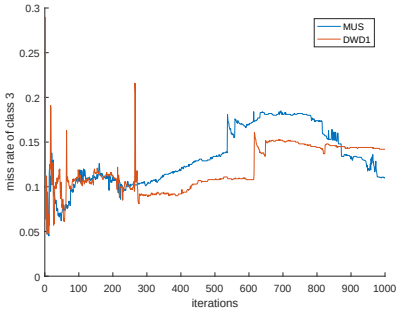
that one, when DWD1 strategy is adopted, in early iterations, but it is higher at the end of the process. The miss rate of class 2 is slightly smaller, when MUS strategy is adopted, through all iterations. The miss rate of class 3 is overall smaller, when DWD1 strategy is utilized.



(a) Miss rate of class 1



(b) Miss rate od class 2



(c) Miss rate of class3

Figure 13: Miss rate of individual classes when margin uncertainty sampling and DWD1 strategy are used

4.4 Discussion

We defined what class imbalance is and what are the most frequent approaches to address this problem. Then, we mentioned that active learning is used in many applications as a sampling method, but it is sometimes necessary to utilize some method which deals with the class imbalance and which helps active learning to learn more accurate model.

Furthermore, we focused on the margin uncertainty sampling strategy and performed several experiments using both oversampling and undersampling on a synthetic dataset.

If undersampling is embraced, then the mean error on classes starts to oscillate. That does not happen if oversampling is added, but the miss rate of only one class decreases and the error on classes overall increases. If we evaluate proposed criteria from Section 3.2.2, the performance of methods with oversampling fails and does not acquire results of methods without oversampling.

We achieved the great improvement if density-weighted methods are used. The DWD1 strategy reaches the best results. We come to a conclusion that it seems to be appropriate to use a density-weighted method on dataset with the extreme class imbalance, rather than some sampling method.

5 Active learning for Sleep EEG Analysis

We will focus on active learning on sleep EEG datasets. First, we will provide a brief introduction to this kind of data, then we will describe which problem we would solve. Furthermore, experiments will performed and obtained results will be commented.

5.1 Sleep EEG

Electroencephalogram (EEG) is a record of cumulative electrical activity of neurons in the brain through electrodes which are placed on scalp [49]. The EEG signal can be divided into five waves according to their frequency – alpha, beta, gamma, delta and theta [50]. Alpha waves occur during relaxation with closed eyes, beta waves are typical for normal consciousness, gamma waves are associated with higher functions of the brain, delta waves occur during the deep sleep and theta waves are associated with drowsiness and beginning of sleep [49].

Polysomnogram (PSG) is a record of several biological signals in sleep, typically of EEG, EOM (electrooculogram), ECG (electrocardiogram) and EMG (electromyogram). Polysomnography is able to distinguished individual stages of sleep [50].

It is distinguished between two stages of sleep that cyclically repeat 4-6 times during sleep of a normal human – non-rapid eye movement (NREM) and rapid eye movement (REM) [51].

NREM sleep is further divided into four stages. Alpha activity typically decreases and theta waves reach the highest amplitude in stage 1. Movement of eyes is slow and EMG activity decreases. Stage 2 overall occupies around fifty percent of sleep and sleep spindles and K-complexes appear during that. Stage 3 and 4 are also called slow-wave sleep and are characterized by high delta activity. During stages 2-4, EMG activity is low, compared to wakefulness and stage 1 [51].

A combination of slow alpha and theta activity appears in REM stage of sleep. We are able to distinguish between phasic and tonic stages in REM sleep. The characteristics of phasic stage are rapid eye movement, irregularities in respiration, twitching of muscles and changes in the heart rate. Tonic stage is characterized by muscular weakness and desynchronized EEG [51].

Although the term "sleep EEG" is in the title of this thesis, we typically deem the whole PSG recording because we aim to learn a classifier, which distinguishes between wakefulness, stages of NREM sleep and REM sleep, so only the record of EEG is insufficient.

5.2 Problem Statement

Currently, when it comes to classifying of sleep to the sleep stages mentioned above, a specialist has to go through all segments of this record and assign them to a proper class (annotate them). We assume that active learning is well suited to this task because

it enables to select only the most informative instances for labeling, so the specialist would classify a smaller amount of observations. The goal of our class is to exploit active learning to learn a classifier so that it would not be necessary to annotate all observations. These observation will be further discussed in Section 5.3.1. Instances will be divided into five classes in our setting: WAKE, REM, NREM1, NREM2 and NREM3.

Note that annotating of a segment is often ambiguous because individual stages are changing continuously e.g. it is difficult to determine if a segments still belongs to the REM stage or if it is already in NREM1 stage. Furthermore, labeling furnished by two experts can vary so let us assume that available annotations we posses only come from one specialist. Nevertheless, we suppose that the miss rate of NREM1 is the biggest of miss rates of other classes. We also assume that uncertainty sampling will query these instances which are tricky to classify, so density-weighted methods are expected to be more convenient for this tasks. Sleep data are also typically imbalanced (see in Section 5.3.1) and density-weighted methods are able to deal with it, as it emerges from Section 4.3.3.

5.3 Experiments

Six active learning methods (margin uncertainty sampling (MUS), DWD1, DWD0.5, UMDN10, UMDN50 and UMDL) are compared to random sampling (RS) to determine if active learning is really able to help or it is sufficient to just randomly annotate several instances. We decided to utilize density-weighted methods due to reasons mentioned above.

Evaluation criteria proposed in Section 3.2.2 are used. The initial set of labeled instances S_L always contains two instances of each class, the process is terminated after 300 iterations and the parzen window estimator is used as the classifier.

5.3.1 Sleep EEG Data

At our disposal, we have five sleep records of a few healthy patients, which were split to 30 seconds-long segments, and these segments were distinguished by a specialist into five classes: WAKE, REM, NREM1, NREM2 and NREM3. Each observation is represented by its 82 features. Six EEG channels were chosen according to the recommendation of American Academy of Sleep Medicine (AASM): F4-M1, C4-M1, O2-M1, F3-M2, C3-M2, O1-M2 [52], on which eleven features were computed. Further, two EOG channels were used, on which five features were acquired, and also two EMG channels, on which three features were computed. We will perform experiments on observations consisting of all 82 features and also on observations containing ten features which were acquired by the principal component analysis.

Table 21: Number of instances of each class for each dataset

Dataset name	WAKE	REM	NREM1	NREM2	NREM3	Sum
Sleep1	284	53	87	309	146	879
Sleep2	200	88	64	443	53	848
Sleep3	292	124	61	316	118	911
Sleep4	362	15	103	299	17	796
Sleep5	87	107	57	482	121	854

The number of instances of each class for all datasets can be seen in Table 21. We split all datasets to two almost equally large parts – to training and testing data. We suppose that we do not know labels of training data except the ones which belong to the initial set of labeled instances. Note that the utilization of cross-validation is not well-suited for this task, because we do not know labels of training data, except for instances belonging to the initial set of labeled observations.

5.3.2 Results

First of all, we use all 82 features of observations that are classified into all five classes. The threshold, which is used in reaching the threshold and breaking point criteria, is estimated visually from the courses of the mean error on classes. The value of given percentage, which is used in the breaking point criterion, is set to 95 %.

We can see values of all criteria for all datasets in the following tables. The best results of a criterion are highlighted in bold, the second best ones are in italics.

Table 22: Evaluation criteria for Sleep1 dataset, threshold is set to 0.15.

Methods	TRtT	MeanRC	ModeRC	BP
RS	248	6.61	5	240
MUS	89	*3.54*	7	78
DWD1	100	3.73	*2*	91
DWD0.5	95	4.39	3	85
UMDN10	**46**	**1.44**	4	**50**
UMDN50	98	4.70	6	89
UMDL	*73*	3.59	**1**	*65*

Table 23: Evaluation criteria for Sleep2 dataset, threshold is set to 0.2.

Methods	TRtT	MeanRC	ModeRC	BP
RS	232	4.54	6	240
MUS	165	3.09	5	233
DWD1	262	5.37	*2*	262
DWD0.5	x	5.72	**1**	x
UMDN10	*159*	*2.10*	3	*154*
UMDN50	**105**	**1.57**	4	**100**
UMDL	x	5.61	4	x

Table 24: Evaluation criteria for Sleep3 dataset, threshold is set to 0.15.

Methods	TRtT	MeanRC	ModeRC	BP
RS	215	6.28	6	221
MUS	117	3.99	6	115
DWD1	128	3.41	3	119
DWD0.5	**93**	*3.01*	4	**86**
UMDN10	101	3.49	*2*	106
UMDN50	*98*	**1.74**	7	*88*
UMDL	148	6.09	**1**	140

Table 25: Evaluation criteria for Sleep4 dataset, threshold is set to 0.2.

Methods	TRtT	MeanRC	ModeRC	BP
RS	x	5.63	*2*	294
MUS	**135**	**2.49**	6	**254**
DWD1	x	5.98	5	x
DWD0.5	x	3.30	4	x
UMDN10	278	3.72	3	*277*
UMDN50	235	*2.78*	**1**	x
UMDL	*234*	4.10	3	x

Table 26: Evaluation criteria for Sleep5 dataset, threshold is set to 0.3.

Methods	TRtT	MeanRC	ModeRC	BP
RS	154	3.92	*2*	193
MUS	144	3.37	6	149
DWD1	*68*	5.33	**1**	239
DWD0.5	74	4.88	**1**	138
UMDN10	*71*	*2.64*	3	**108**
UMDN50	93	**2.33**	4	*117*
UMDL	165	5.53	7	162

Now, let us summarize all these results. In Table 27, it is shown how many times and in which criterion a certain method is better than random sampling, to determine if active learning works and which active learning method leads to better results.

Table 27: Summarization of how many times active learning methods are better than random sampling in each criterion on all datasets.

Methods	TRtT	MeanRC	ModeRC	BP	Sum
MUS	5	5	1	5	16
DMD1	3	2	4	2	11
DWD0.5	2	3	4	3	12
UMDN10	5	5	3	5	18
UMDN50	5	5	2	4	16
UMDL	3	3	3	3	12

The UMDN10 strategy provides better results than random sampling 18 times out of 20 cases. This strategy is beaten by random sampling only in the mode of ranks criterion in which random sampling is the second best strategy on datasets Sleep4 and Sleep5. If this strategy is used on Sleep1 dataset with the given threshold, it is able to save 833 instances for annotating (i.e. almost 95 % of all training data), 689 instances on Sleep2 dataset (i.e. 81.25 % of all training data), 813 instances on Sleep3 dataset (i.e. 89 % of all training instances), 518 instances on Sleep4 dataset (i.e. 65 % of all training data) and 783 instances on Sleep5 dataset (i.e. almost 92 % of instances).

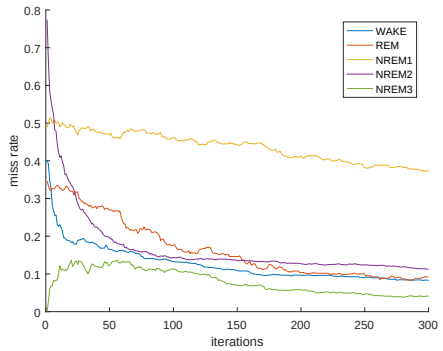
Margin uncertainty sampling and UMDN50 strategies share the second place. UMDN50 strategy beats margin uncertainty sampling strategy in thirteen out of twenty cases, but on the other hand, UMDN50 fails in the breaking point criterion on Sleep4 dataset. Both strategies save almost the same number of instances on Sleep1 datasets – 90 % of training instances in the case of margin uncertainty sampling and 89 % of training data in the case of UMDN50 strategy. MUS saves 80.5 % of training data on Sleep2 dataset, UMDN50 strategy saves almost 88 %. On the Sleep3 dataset, margin uncertainty sampling saves 80 % of instances and UMDN50 strategy 89 %. Margin uncertainty sampling saves 83 % and UMDN50 strategy 70 % of instances on Sleep4 dataset. UMDN50 saves 89 % of training data on Sleep5 dataset and MUS strategy saves 83 % of instances.

UMDL and DWD0.5 strategies reaches better results than random sampling only in twelve out of twenty criteria, DWD1 in eleven criteria. DWD1 provides good results on Sleep5 dataset where is the best in one criterion and the second best also in one criterion, it saves above that 92 % of training instances on this dataset. DWD0.5 reaches the best results on Sleep3 datasets, it saves almost 90 % of training data there and is the best in two criteria. UMDL strategy is the best in one criterion and the second best in two criteria on Sleep1 dataset where it saves almost 92 % of instances.

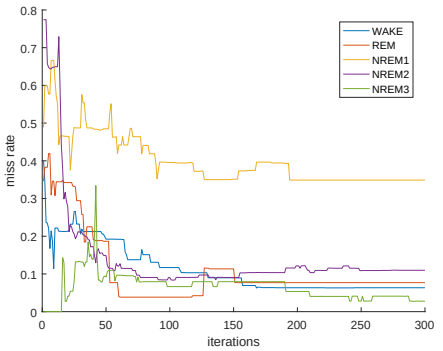
If we will summarize how many instances a particular method saves on all datasets in total, then MUS saves almost 85 % of all instances, DWD1 strategy saves less than 80 % of all instances, DWD0.5 strategy saves also less than 80 % of observations, both UMDN10 and UMDN50 save 85 % of instances and UMDL saves less than 79 % of all

observations. Random sampling saves less than 73 % of all instances. On the other hand, random sampling and DWD1 strategies do not reach the given threshold on Sleep4 dataset, UMDL strategy on Sleep2 dataset and DWD0.5 on Sleep2 and Sleep4 datasets. From this point of view, DWD0.5 achieves the worst results of all strategies.

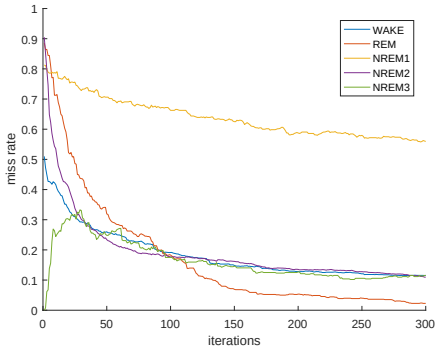
As we mentioned in Section 5.2, we suppose that the miss rate of NREM1 is the largest of miss rates of other classes. Let us depict miss rates of all classes on Sleep1 and Sleep2 datasets. The miss rates of all classes when MUS strategy is used on Sleep1 dataset is shown in Figure 14a, when UMDN10 strategy is adopted on the same dataset is depicted in Figure 14b. The miss rate of NREM1 class on Sleep2 dataset is shown in Figure 14c (in the case of using MUS strategy) and in Figure 14d (for UMDN10 strategy). Our assumption is confirmed, the miss rate of NREM1 class reaches the highest values of all other miss rates.



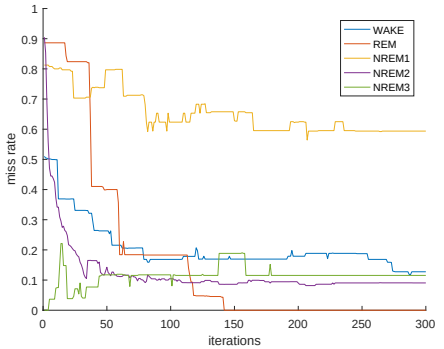
(a) Sleep1 dataset, MUS strategy



(b) Sleep1 dataset, UMDN10 strategy



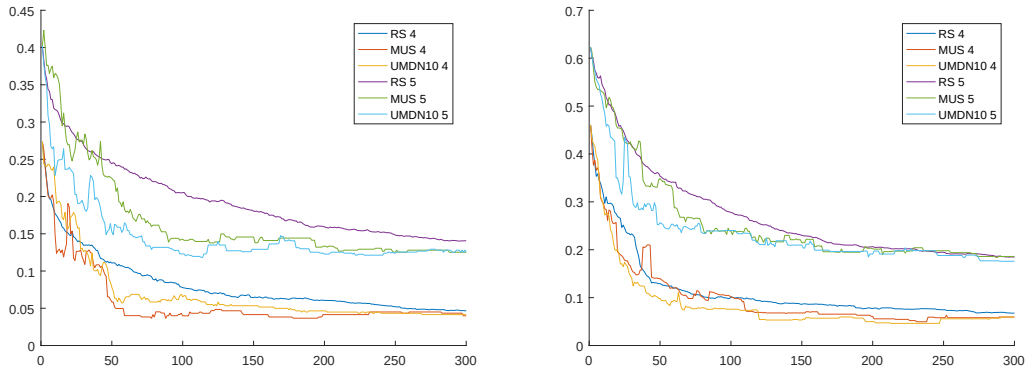
(c) Sleep2 dataset, MUS strategy



(d) Sleep2 dataset, UMDN10 strategy

Figure 14: Miss rates of all classes on Sleep1 and Sleep2 datasets

Let us perform experiments on these two datasets when NREM1 class is excluded. Results are shwon in Figure 15. Miss rates of all classes with classification to four or to five classes on Sleep1 dataset are depicted in Figure 15a, the same on Sleep2 dataset is shown in Figure 15b.



(a) Sleep1 dataset (b) Sleep2 dataset

Figure 15: Miss rates of all classes on Sleep1 and Sleep2 datasets when observations are classified into four (without NREM1 class) or into all five classes

It is clear that when instances belonging to NREM1 are removed, then the overall mean error on classes is smaller. An interesting fact is that if the classifier distinguishes between five classes, then UMDN10 strategy seems to be more accurate than margin uncertainty sampling, and if instances from NREM1 class are excluded, then margin uncertainty sampling reaches better results than UMDN10. Let us evaluate our proposed criteria on Sleep1 and Sleep2 datasets when the classifier classify into four classes, for all strategies. The threshold in both cases is set to 0.1. Best achieved result in each criterion is in bold, the second best is in italics.

Table 28: Evaluation criteria for Sleep1 dataset when NREM1 class is excluded, threshold is set to 0.1

Methods	TRtT	MeanRC	ModeRC	BP
RS	65	6.65	7	54
MUS	45	3.33	4	32
DWD1	**24**	3.56	4	*20*
DWD0.5	**24**	*3.32*	*3*	23
UMDN10	41	5.03	6	33
UMDN50	36	4.46	5	49
UMDL	*33*	**1.65**	**1**	**19**

Table 29: Evaluation criteria for Sleep2 dataset when NREM1 class is excluded , threshold is set to 0.1

Methods	TRtT	MeanRC	ModeRC	BP
RS	84	5.61	5	108
MUS	*70*	3.51	5	93
DWD1	72	4.58	*2*	64
DWD0.5	*70*	3.75	4	58
UMDN10	49	**1.71**	**1**	**37**
UMDN50	**67**	*2.51*	7	*55*
UMDL	82	6.33	7	97

Our assumption is confirmed only on Sleep1 dataset when MUS strategy is better than UMDN10 strategy in all criteria. On the other hand, UNDM10 strategy is better than margin uncertainty sampling in three out of four criteria on Sleep2 dataset.

Since 82 is a large amount of feature, the principal component analysis is used on Sleep1 dataset to reduce the number of features. In Figure 16, it is shown that with the decreasing number of features the overall mean error on classes increases. Notice that the performance of UMDN10 strategy gets worse with the decreasing number of features. The decision is reached that UMDN10 strategy might be better in our setting (when all 82 features are used), but margin uncertainty sampling is more general and suited and fits for datasets which we have not a large amount of information about.

5.4 Discussion

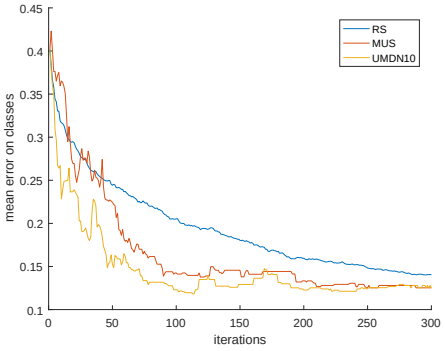
Experiments on five sleep datasets were performed in this section. Observation were classified to five classes: WAKE, REM, NREM1, NREM2 and NREM3. Since sleep data are in general imbalanced and we acquired good results using density-weighted active learning methods on skewed datasets, UMDN strategy with different parameters, DWD strategy also with different parameters and UMDL strategy were used together with the standard margin uncertainty sampling strategy and random sampling.

The best results were acquired by using the UMDN10 strategy which reached the better results than random sampling in most cases.

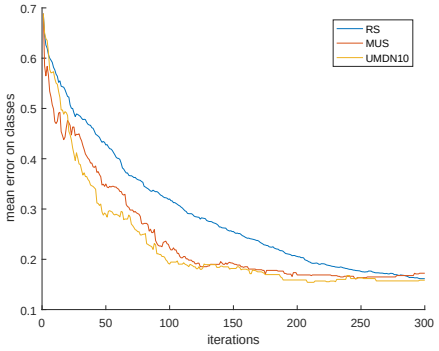
Although UMDN10 strategy's performance seemed to be the best, the quality of margin uncertainty sampling cannot be omitted. When all eighty-two features of instances were used, margin uncertainty got the second best results. When twenty or less features (acquired by PCA) were exploited, margin uncertainty sampling led to better performance than UDN10 strategy despite the fact that overall mean error of all strategies increased. To sum up, margin uncertainty sampling is well-suited for application where there is no former knowledge about data.

It was also found out that miss rate of NREM1 class is the largest. We assume that it is caused by the fact that this state is transient between REM sleep and deeper stages of sleep so it is difficult for a specialist to determine if an instances belongs to REM or to NREM1 stage.

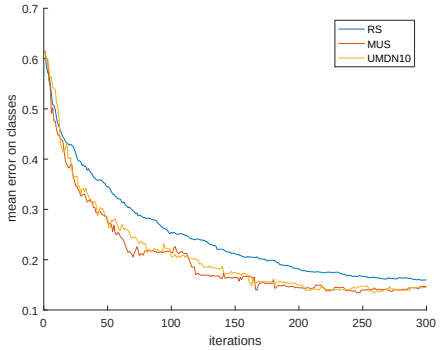
Figure 16: Mean error on classes on Sleep1 dataset for different number of features acquired by PCA



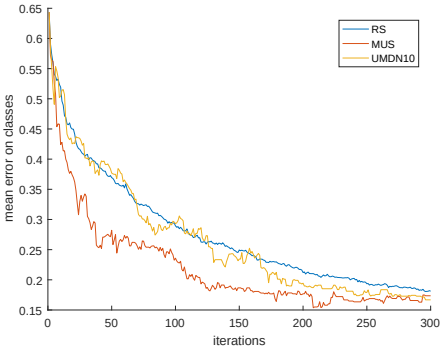
(a) All features used



(b) 50 features



(c) 20 features



(d) 10 features

6 Conclusion

The aim of this thesis was the utilization of active learning on Sleep EEG classifier. Because this kind of data is typically imbalanced, it was necessary to focus on active learning methods which are well-suited for the class imbalance problem.

Since active learning methods are mostly compared visually, it was first required to implement several evaluation methods. We proposed four criteria for the evaluation, which all are described in Section 3.2.2 and used them for evaluating active learning methods on UCI Machine Learning Repository datasets. We came to conclusion that it is still hard to compare active learning methods, even when these criteria are adopted, because one method can reach the best result in one criterion but completely fail in another. Due to this fact, we suggest to compare results acquired by criteria with these ones achieved by random sampling. If a method is worse than random sampling in more than fifty percent of each cases, then this method fails.

We then concentrated on the influence of class imbalance on active learning. Because query by committee active learning strategy is often exploited to handle the class skewness problem, we decided to focus more on uncertainty sampling methods and density-weighted methods which use uncertainty sampling as the function of informativeness. For this reason, several experiments with adaptive oversampling and undersampling were performed and we reached the decision that density-weighted strategies without any sampling method are convenient for datasets which are imbalanced.

We utilized three density-weighted methods (we proposed two of them in Section 2.4.2) with different parameters, margin uncertainty sampling and random sampling on five datasets containing sleep data. The best results were obtained by our UMDN10 strategy, which beats random sampling in 18 out of 20 cases (four criteria are computed on five datasets), and which saves more 80 % of instances for labeling on each dataset expect one, it saves even more than 90 % of instances on two datasets. Though, the UMDN10 strategy reached the best results, we cannot omit results acquired by the basic margin uncertainty sampling strategy, which lead to better results than UMDN10 strategy, where the number of features was decreased by PCA. We reached the same decision as Settles et al. [12] and that is as follows: although we can develop strategies which are well-suited for our concrete problem, the simplest least confidence strategies are not outdated.

We figured out that NREM1 class has the biggest miss rate of all other classes. It could be caused by an ambiguous transition between REM sleep and deeper stages of sleep and we assume that if these ambiguous instances were labeled by other specialist, he or she would be able to classify them differently. In active learning terminology, this is called “noisy oracles”. Although heuristics which deal with noisy oracles exist, many aspects of this problem are still open [2], so the further research might be more focused on them.

We did not take into account that sleep stages are time-dependent, i.e. the transition e.g. between WAKE and REM stage or REM and NREM1 stages is possible, but not possible between e.g. REM and NREM2. The future work should also more focus on this setting.

References

[1] Katrin Tomanek. *Resource-aware Annotation Through Active Learning.* PhD thesis, Technical University Dortmund, 2010.

[2] Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[3] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[4] Vladimir N Vapnik. An overview of statistical learning theory. *IEEE transactions on neural networks*, 10(5):988–999, 1999.

[5] Xiaojin Zhu. Semi-supervised learning literature survey. 2005.

[6] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Pearson Education, 2 edition, 2003.

[7] Michael Prince. Does active learning work? a review of the research. *Journal of engineering education*, 93(3):223–231, 2004.

[8] Alexander Roederer. *Active Learning for Classification of Medical Signals.* PhD thesis, University of Pennsylvania, 2012.

[9] Burr Settles. From theories to queries: Active learning in practice. In *JMLR Workshop and Conference Proceedings*, volume 15, pages 1–18. Microtome Publishing, 2011.

[10] Steven C. H. Hoi, Rong Jin, Jianke Zhu, and Michael R. Lyu. Batch mode active learning and its application to medical image classification. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 417–424, New York, NY, USA, 2006. ACM.

[11] Andrew McCallum and Kamal Nigam. Employing em and pool-based active learning for text classification. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 350–358, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[12] Burr Settles and Mark Craven. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 1070–1079, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

[13] V. Lawhern, D. Slayback, D. Wu, and B. J. Lance. Efficient labeling of eeg signal artifacts using active learning. In *2015 IEEE International Conference on Systems, Man, and Cybernetics*, pages 3217–3222, Oct 2015.

[14] Liantao Wang, Xuelei Hu, Bo Yuan, and Jianfeng Lu. Active learning via query synthesis and nearest neighbour search. *Neurocomputing*, 147:426–434, 2015.

[15] Ibrahim Alabdulmohsin, Xin Gao, and Xiangliang Zhang. Efficient active learning of halfspaces via query synthesis. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[16] David D. Lewis and Jason Catlett. Heterogeneous uncertainty sampling for supervised learning. In *In Proceedings of the Eleventh International Conference on Machine Learning*, pages 148–156. Morgan Kaufmann, 1994.

[17] Meng Wang and Xian-Sheng Hua. Active learning in multimedia annotation and retrieval: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(2):10, 2011.

[18] Burr Settles. *Curious Machines: Active Learning with Structured Instances*. PhD thesis, University of Wisconsin–Madison, 2008.

[19] Tobias Scheffer, Christian Decomain, and Stefan Wrobel. Active hidden markov models for information extraction. In *International Symposium on Intelligent Data Analysis*, pages 309–318. Springer, 2001.

[20] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 287–294, New York, NY, USA, 1992. ACM.

[21] Ido Dagan and Sean P. Engelson. Committee-based sampling for training probabilistic classifiers. In *In Proceedings of the Twelfth International Conference on Machine Learning*, pages 150–157. Morgan Kaufmann, 1995.

[22] Grace Ngai and David Yarowsky. Rule writing or annotation: Cost-efficient resource usage for base noun phrase chunking. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, ACL '00, pages 117–125, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

[23] Burr Settles, Mark Craven, and Soumya Ray. Multiple-instance active learning. In *Advances in neural information processing systems*, pages 1289–1296, 2008.

[24] Alexander Vezhnevets, Joachim M Buhmann, and Vittorio Ferrari. Active learning for semantic segmentation with expected change. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3162–3169. IEEE, 2012.

[25] Wenbin Cai, Ya Zhang, and Jun Zhou. Maximizing expected model change for active learning in regression. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 51–60. IEEE, 2013.

[26] Nicholas Roy and Andrew McCallum. Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 441–448, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[27] Josh Attenberg and Foster Provost. Why label when you can search?: Alternatives to active learning for applying human resources to build classification models under

extreme class imbalance. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 423–432, New York, NY, USA, 2010. ACM.

[28] Jingbo Zhu, Huizhen Wang, Tianshun Yao, and Benjamin K. Tsou. Active learning with sampling by uncertainty and density for word sense disambiguation and text classification. In *Proceedings of the 22Nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 1137–1144, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

[29] Hieu T Nguyen and Arnold Smeulders. Active learning using pre-clustering. In *Proceedings of the twenty-first international conference on Machine learning*, page 79. ACM, 2004.

[30] Seyda Ertekin. *Learning in extreme conditions: Online and active learning with massive, imbalanced and noisy data*. PhD thesis, Citeseer, 2009.

[31] M. Lichman. UCI machine learning repository, 2013.

[32] Emanuel Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962.

[33] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Stanford, CA, 1995.

[34] Wen Zhu, Nancy Zeng, Ning Wang, et al. Sensitivity, specificity, accuracy, associated confidence interval and roc analysis with practical sas implementations. *NESUG proceedings: health care and life sciences, Baltimore, Maryland*, pages 1–9, 2010.

[35] Haibo He and Yunqian Ma. *Imbalanced Learning: Foundations, Algorithms, and Applications*. Wiley-IEEE Press, 1st edition, 2013.

[36] Miroslav Kubat, Robert C Holte, and Stan Matwin. Machine learning for the detection of oil spills in satellite radar images. *Machine learning*, 30(2-3):195–215, 1998.

[37] David Martin Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. 2011.

[38] Arzucan Özgür, Levent Özgür, and Tunga Güngör. Text categorization with class-based and corpus-based keyword selection. In *International Symposium on Computer and Information Sciences*, pages 606–615. Springer, 2005.

[39] Jingbo Zhu and Eduard H Hovy. Active learning for word sense disambiguation with methods for addressing the class imbalance problem. In *EMNLP-CoNLL*, volume 7, pages 783–790, 2007.

[40] Seyda Ertekin, Jian Huang, Leon Bottou, and Lee Giles. Learning on the border: active learning in imbalanced data classification. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 127–136. ACM, 2007.

[41] Yanmin Sun, Mohamed S Kamel, Andrew KC Wong, and Yang Wang. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40(12):3358–3378, 2007.

[42] Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484, 2012.

[43] Ciza Thomas and N Balakrishnan. Improvement in minority attack detection with skewness in network traffic. In *SPIE Defense and Security Symposium*, pages 69730N–69730N. International Society for Optics and Photonics, 2008.

[44] Nitesh V Chawla, Nathalie Japkowicz, and Aleksander Kotcz. Editorial: special issue on learning from imbalanced data sets. *ACM Sigkdd Explorations Newsletter*, 6(1):1–6, 2004.

[45] Rehan Akbani, Stephen Kwek, and Nathalie Japkowicz. Applying support vector machines to imbalanced datasets. In *European conference on machine learning*, pages 39–50. Springer, 2004.

[46] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[47] Zhi-Hua Zhou and Xu-Ying Liu. Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Transactions on Knowledge and Data Engineering*, 18(1):63–77, 2006.

[48] David A Cieslak, T Ryan Hoens, Nitesh V Chawla, and W Philip Kegelmeyer. Hellinger distance decision trees are robust and skew-insensitive. *Data Mining and Knowledge Discovery*, 24(1):136–158, 2012.

[49] Ramaswamy Palaniappan. *Biological signal analysis*. BookBoon, 2011.

[50] Václav Gerla. *Automatic Analysis of Long-Term EEG Signals*. PhD thesis, Czech Technical University, 2012.

[51] Anil Natesan Rama, S Charles Cho, and Clete A Kushida. Normal human sleep. *Sleep: a comprehensive handbook*, 2006.

[52] Brett Duce, Conchita Rego, Jasmina Milosavljevic, and Craig Hukins. The aasm recommended and acceptable eeg montages are comparable for the staging of sleep and scoring of eeg arousals. *Journal of clinical sleep medicine: JCSM: official publication of the American Academy of Sleep Medicine*, 10(7):803, 2014.

[53] MATLAB. *version 7.10.0 (R2010a)*. The MathWorks Inc., Natick, Massachusetts, 2010.

[54] R.P.W. Duin, P. Juszczak, P. Paclik, E. Pekalska, D. de Ridder, D.M.J. Tax, and S. Verzakov. Pr-tools4.1, a matlab toolbox for pattern recognition, 2007.

A Contents of CD

The attached CD contains:

- The electronic version of this thesis in PDF format
- The folder called *active_learning* which contains the source code
- The folder called *data* which contains all utilized datasets

B Data and Source Code

All used data are stored in the *data* folder. UCI datasets, which were used in Section 3.3, are in corresponding mat-files: *Banknote*, *Pima*, *Spambase* and *Statlog*. Files which begin with the word *data1* contains data of the first dataset which was used in Section 4.3. The number 11 corresponds to the class ratio 1:1, the number 12 to the class ratio 1:2, etc. Data of the second dataset from Section 4.3 are stored in files starting with *data2*. Sleep datasets, which were utilized in Section 5.3, are stored in files *Sleep1*, *Sleep2*, *Sleep3*, *Sleep4* and *Sleep5*.

All experiments proposed in this thesis were performed in MATLAB environment [53] with the utilization of PRTools framework [54]. The folder *program* contains m-files – strategies which were described in this thesis are in files which start with the world *strategy*, files called *oversampling* and *undersampling* contain these sampling methods and two initialization techniques are stored in files starting with the word *init*.

The program is launched from the file *main*. The user has to specify which dataset will be used, the number of iterations and repetitions, the used classifier, how many instances will be in the initial set of labeled observations and if some sampling method will be used. Note that if it is needed to use undersampling, then the variable *under* denotes the biggest possible ratio of the minority to the majority class. When it is not necessary to exploit oversampling, then the variable *over* is empty (i.e. $over = \{\}$).

The user has to assign which initialization or learning strategies will be used. It is shown on the following example:

inits = {*'init_kmeans', 4;*}
strategies = {*'strategy_rs', [], []; 'strategy_umdn', 10', 'det';*}

The proposed entry means that the k-means initialization will be used with $k = 4$ and random sampling and UMDN10 strategies will be utilized. The second term in strategies specifies parameters of a method (e.g. it is empty in the case of random sampling, or it contains the number 10 in the case of UMDN10 strategy). Note that the term *'det'* is added to every deterministic strategies which are UMDN, UMDL and DWD strategies with different parameters. Least confidence strategies are stochastic in our setting – if there are more instances which have the same value of uncertainty, then one of them is randomly selected.

The structure all_data contains cells, each of them correspond to one strategy. Each cell contains the identification of used initialization and learning strategies, the mean error on testing data, the mean error on training data, miss rates of all classes and values of a criterion in each iteration and in each repetition of the algorithm.