DOI: 10.15439/2015F90
ACSIS, Vol. 5

Transformation of nominal features into numeric in supervised multi-class problems based on the weight of evidence parameter

Eftim Zdravevski[*], Petre Lameski[†], Andrea Kulakov[‡], Slobodan Kalajdziski[§]
Faculty of Computer Science and Engineering
Ss.Cyril and Methodius University, Skopje, Macedonia
Email: {[*]eftim.zdravevski, [†]petre.lameski, [‡]andrea.kulakov, [§]slobodan.kalajdziski}@finki.ukim.mk

***Abstract*—Machine learning has received increased interest by both the scientific community and the industry. Most of the machine learning algorithms rely on certain distance metrics that can only be applied to numeric data. This becomes a problem in complex datasets that contain heterogeneous data consisted of numeric and nominal (i.e. categorical) features. Thus the need of transformation from nominal to numeric data. Weight of evidence (WoE) is one of the parameters that can be used for transformation of the nominal features to numeric. In this paper we describe a method that uses WoE to transform the features. Although the applicability of this method is researched to some extent, in this paper we extend its applicability for multi-class problems, which is a novelty. We compared it with the method that generates dummy features. We test both methods on binary and multi-class classification problems with different machine learning algorithms. Our experiments show that the WoE based transformation generates smaller number of features compared to the technique based on generation of dummy features while also improving the classification accuracy, reducing memory complexity and shortening the execution time. Be that as it may, we also point out some of its weaknesses and make some recommendations when to use the method based on dummy features generation instead.**

***Keywords*—Weight of Evidence, WoE, dummy features, data transformation, nominal features, categorical features, heterogeneous data**

I. Introduction

CLASSIFICATION is one of the most researched problems in the data mining community. The data mining process is consisted of business and data understanding, data preparation, modeling, evaluation and deployment [1]. One of the crucial parts of this process is the data preparation which has a very large influence over the success or failure of the classification. Data preparation is not a trivial task and is dependent on the nature of the data. There are many problems that need to be addressed during the data preparation such as the existence of outliers, errors, noise, missing values etc. It is important that the data is processed and transformed correctly so that the problems that exist in the raw data are eliminated or their influence reduced as much as possible. For this we use different data transformation methods.

Depending on the data type of the features different transformations are suitable. Some of the most common methods for data transformation are well described in [2] and [3]. Transforming numeric data, be it continuous or discrete, can be done in a variety of ways. Then again, transformations of nominal and categorical data are not as extensively researched. This issue is also highlighted in [4]. A very common way for transformation of nominal features is by generating dummy features (i.e. varables). It is characterized by simplicity, independence of the data domain, and ease of implementation. Authors in [5, 3, 6] recommend to generate dummy features when there is no mapping of nominal to numeric data. After the data transformation the distance between the dummy features of the instances can be calculated in different ways, as described in [7]: Euclidean distance, Hamming distance, Jaccard distance, Levenshtein distance, etc. In this paper we treat binary dummy features as numeric features, and the distances are calculated intrinsically in the learning algorithm.

Another similarity measure for nominal features is proposed in [8]. It gives greater weight to uncommon feature value matches in similarity computations and makes no assumptions about the underlying distributions of the feature values. Application of this measure in an unsupervised setting to define the similarity metric between pairs of objects is proposed in [9].

The term frequency-inverse document frequency (TF-IDF), as described in [10] and [11], is often used in text mining problems as a numerical statistic which estimates the importance of a word to a document in a collection of documents. In a similar manner this weight can be used to transform arbitrary nominal values into numerical just as it assigns weight to words in text mining and information retrieval. This approach is mentioned in [12], but using TF-IDF as nominal data transformation technique during the preprocessing phase has not be widely researched.

The weight of evidence parameter, originally defined in [13], can be used for estimating the evidence in support of a hypothesis. Additionally it can be used for transformation of nominal data. Its applicability with examples is also discussed in [14] and [2]. There are some computational limitations of this method and we have addressed them in [15], so it can be used even when the preconditions are not met. In [16] we present an application of this transformation for calculation of the information value of features, which is consequently applied for feature selection.

One of the most serious drawbacks of this method is that it is applicable only to binary classification problems. In this paper we are providing theoretical foundations in order to extend the applicability of the WoE method to multi-class problems. In order to evaluate its advantages, limitations and drawbacks we are comparing it against the technique that generates dummy features. The reason are using this technique as benchmark is because it is most widely adopted in both literature and practice. With this in mind, we have tested both methods on binary and multi-class classification problems and we have trained Support Vector Machines (SVMs) with different types of kernels and a feed forward back-propagation neural network. Finally, we summarize the findings of our research and discuss the applicability of the proposed extension of the weight of evidence parameter and give some recommendations of when to use it.

II. Transformation with generation of dummy features

This section describes in more detail the method for transformation of nominal into numeric features based on generation of dummy features. By using this technique n new dummy features are generated from a nominal feature that has n different values, when $n > 2$. When $n = 2$ only one dummy feature is generated. Equation (1) gives the total number of generated dummy features where v_i is the number of different values of the i-th nominal features. Each of the generated features can have a value of 0 or 1, depending on the occurrence of a particular value of the original feature. This approach has been published for the first time in [17] and was mainly used in regression analysis. Also, [18] covers many aspects related to regression analysis with dummy variables. Over time, this technique has been added to many software packages as a common stage before applying various machine-learning algorithms. When the number of nominal features and the number of different values they can have is small, this transformation leads to good performance of the algorithms. The problems arise when there are a lot of nominal features that can have many different values. These kinds of situations lead to rapid increase of the number of generated dummy features, which in turn, slows down machine-learning algorithms. In fact, the memory requirements or time complexity of algorithms can expand to that degree that they cannot be executed in a reasonable time on the computers that we have today. This issue can be partially addressed by discarding some of the generated features based on their predictive power. By doing that, some potentially useful information in the discarded features is consciously thrown away.

$$z_{dummy} = \sum_{i=0}^{n} v_i \tag{1}$$

III. Weight of evidence

Decision making is mostly based on estimating the probability that one event might occur. The complexity of decision making varies from trivial decisions to some complex ones that require more involved processing of data from multiple sources. The outcome of this probabilistic decision making depends on facts that might even have inter dependencies [19]. For each decision we need to weight the influence of the facts that contribute to it. This provides us means of mapping the risk associated with a given choice or fact on a linear scale.

The concept of numerically weighting evidence was first introduced in [13] and is a result of the work performed by Alan Turing and I.J. Good during World War II. It is a statistical quantitative method for evaluating the facts (evidence) in support of a hypothesis. The weighting is performed with the parameter Weight of Evidence (WoE)[2]. It is a great tool for estimating the relative risk based on the available data. In this section we describe the mathematical background of WoE when used for binary classification problems, originally described in [15] and subsequently in [20].

Equation (2) defines the weight of evidence (WoE) of the i-th value of the feature A, where N_i^A is the number of data points (i.e. instances) that were labeled as negative, and P_i^A is the number of data points that were labeled as positive for the i-th value of the feature A. SN is the total number of negatively labeled data points, PN is the total number of positively labeled data points in the training set, and n^A is the number of different values for the feature A.

$$WoE_i^A = ln\left(\frac{\frac{N_i^A}{SN}}{\frac{P_i^A}{SP}}\right) = ln\left(\frac{N_i^A}{P_i^A}\right) - ln\left(\frac{SN}{SP}\right) \tag{2}$$

Values of SN and SP can be calculated with (3) and (4), respectively:

$$SN = \sum_{i=1}^{n^A} N_i^A \tag{3}$$

$$SP = \sum_{i=1}^{n^A} P_i^A \tag{4}$$

WoE, as illustrated in the second part of (2), has two components: a variable component and a constant component. These numbers are independent of the machine learning algorithm that is going to be applied in the data mining process. They are calculated in the preprocessing phase. The variable component is calculated based on the data points that have a particular value of feature A and the constant component is based on the whole sample (the training data set). In real-time systems, these values should be calculated at regular tyme intervals based on the dynamic of the new data input. There are various statistics that can be monitored so we can determine the need of updating the WoE parameters.

Equation (2) implies that the values for N_i^A and P_i^A have to be different than zero, and given that they represent counts, these constraints transform to $N_i^A > 0$ and $P_i^A > 0$. However, these preconditions are not always met in real datasets thus imposing limited applicability of the WoE parameter. The following section reviews the adjustments of the WoE technique that overcome these preconditions as they were proposed in [15].

IV. Calculation of Weight of evidence for binary problems when preconditions are not met

As mentioned in the definition of WoE, in order to calculate WoE, we must satisfy the constraints $P_i^A > 0$ and $N_i^A > 0$.

These conditions must be met for any value of the feature so that the WoE can be computed. To be able to compute WoE for all values of the nominal feature that needs to be transformed, we need to make some adjustments in the way WoE is calculated for the cases where the preconditions are not met. The different types of unsatisfied preconditions as proposed in [15] are listed below.

Case 1: *The number of positively labeled data points is zero ($P_i^A = 0$) and the number of negatively labeled data points is zero ($N_i^A = 0$).* This is a trivial case when there are no data points (i.e. instances) with the i-th value of the feature A, even though this value is valid for this feature. In such case we assume that WoE_i^A is zero, meaning that this particular value of the feature A will have no impact on any transformations nor will change the calculation of some other parameters that are dependent on WoE. In fact, this value could be even deleted from the possible set of values for the current attribute. However, because it does not have effect on anything, it could be retained in the set of possible values in case some data points from the data sets have the i-th value of variable A, as well as for future analysis.

Case 2: *The number of positively labeled data points is zero ($P_i^A = 0$) and the number of negatively labeled data points is greater than zero ($N_i^A > 0$).* There are no positively labeled data points, and only negatively labeled data points with the i-th value of the feature A. In order to apply (2), we propose to use the value $P_i^A = 1$ for the positively labeled data points. At the same time, we propose to add the appropriate number of negatively labeled data points, so the overall ratio of the added positively and negatively labeled data points will be equal to the ratio of positively and negatively labeled data points in the whole data set (SN/SP). In the following equations let us denote the artificially added positive data points (in this case only one) with $\delta p_i^A = 1$ and with δn_i^A denotes the added negative data points. These artificial "additions" of data points does not involve actual additions of instances in any data set, rather it only alters the number of counted data points of particular type for the purpose of the calculations. If instead of one "added" data point ($\delta p_i^A = 1$) we were to add more, then we would need to add more negatively labeled data points δn_i^A compared to what we are adding now. This, in turn, may pose a problem because the artificial data points may become greater than the actual data points that are negatively labeled. Equations (5) and (6) define how the number of added data points will be calculated, as well as, the proposed estimate of WoE with (10).

$$\frac{\delta p_i^A}{\delta n_i^A} = \frac{SP}{SN} \tag{5}$$

After applying $\delta p_i^A = 1$ in (5), we can calculate δn_i^A with (6):

$$\delta n_i^A = \frac{SN}{SP} \tag{6}$$

Now P_i^A and N_i^A that were defined in section III can be modified to include the artificially added data points:

$$\Delta P_i^A = P_i^A + \delta p_i^A = 1 \tag{7}$$

$$\Delta N_i^A = N_i^A + \delta n_i^A = N_i^A + \frac{SN}{SP} \tag{8}$$

Then, instead of calculating WoE with (2) using N_i^A and P_i^A, we can calculate it with their modified values ΔN_i^A and ΔP_i^A, defined in the two previous equations:

$$WoE_i^A = ln\left(\frac{\Delta N_i^A}{\Delta P_i^A}\right) - ln\left(\frac{SN}{SP}\right) \tag{9}$$

And if we apply (7) and (8) in (9), finally we get the proposed estimate of WoE for this case of unsatisfied preconditions:

$$WoE_i^A = ln\left(\frac{N_i^A \times SP + SN}{SN}\right) \tag{10}$$

Case 3: *The number of negatively labeled data points is zero ($N_i^A = 0$) and the number of positively labeled data points is greater than zero ($P_i^A > 0$).* There are only positively labeled data points with the i-th value of the variable A. We propose to add one data point that is labeled as negative, so we can use $N_i^A = 1$ when applying (2), and to add the appropriate number of positively labeled data points, so the overall ratio of the added positively and negatively labeled data points will be equal to the ratio of positively and negatively labeled data points in the whole data set (SN/SP). As in the previous case, these artificial "additions" of data points are virtual because the instances in the data sets are left intact, rather we only alter the number of counted data points of a particular type. As for why we are using only one artificial "addition" the same observation as in Case 2 stands. In this case the number of artificially added negative data points is one ($\delta n_i^A = 1$), so (5) can be transformed as (11):

$$\delta p_i^A = \frac{SP}{SN} \tag{11}$$

Now P_i^A and N_i^A that were defined in section III can be modified to include the artificially added data points:

$$\Delta P_i^A = P_i^A + \delta p_i^A = P_i^A + \frac{SP}{SN} \tag{12}$$

$$\Delta N_i^A = N_i^A + \delta n_i^A = 1 \tag{13}$$

Finally, for this case of unsatisfied preconditions, instead of calculating WoE with (2) using N_i^A and P_i^A, we can calculate it with their modified values ΔN_i^A and ΔP_i^A, defined in the two previous equations:

$$WoE_i^A = ln\left(\frac{SP}{P_i^A \times SN + SP}\right) \tag{14}$$

In this section we reviewed the enhancement [15] for estimation of WoE for cases in which the it can not be calculated by using the original (2). With this approach we add some data points so that the number of positive and negative labeled data points is always positive. This has a very small influence on the data and does not change the overall distribution of the dataset. There are several benefits from using the proposed method for WoE estimation:

- It would be computable for all features and all values in the data set, meaning that WoE could be used to transform the nominal features into numeric.
- The computed WoE could be used for binning of some values of the features.
- Information value of all features could be computed, and later it could be used for feature selection.
- Many classification algorithms have preference of numeric over nominal features, and sometimes the distance between different data points cannot be estimated if the values of the features are nominal. After we calculate the WoE values, the data points can be compared in terms of WoE.

The proposed transformation could degrade the performance of the classification model when significant noise is present. The estimated risk in such cases could defer from the real risk. Noisy data, however, poses a serious problem in the data mining in general and should be addressed before performing any kind of data transformation and using some machine learning algorithm.

V. One-vs-all generalization for multi-class problems of the weight of evidence parameter

One of the most constraining properties of the Weight of evidence parameter is that it is computable only for binary classification problems. On the other hand, many real data mining and machine learning applications require classification into more than two classes. In order for the WoE parameter to be applicable for such cases, the algorithm for its calculation needs to be modified accordingly. One way to achieve this is by representing the multi-class classification problem as a set of binary problems. After that, we can calculate the WoE values separately for each of the binary subproblems. In [21] and [22] is applied a similar approach, known as one-vs-all or one-vs-rest, for generalization of many machine-learning algorithms that natively support only two classes (e.g. SVMs). By applying the one-vs-all technique we can also generalize the WoE transformation for multi-class problems. We have followed this idea originally in [23], but without substantial formal definition nor significant empirical evidence. Here we explore the idea of one-vs-all generalization of the WoE transformation in more depth.

Algorithm 1 is repeated for each of the m classes. With this algorithm from a dataset with k nominal features and m classes we generate z_{woe} new numeric features, as defined with (15).

Algorithm 1 One-vs-all generalization for multi-class problems of the Weight of evidence parameter

```
for $i = 1 \rightarrow m$ do
    Temporary label with $TempClass^1$ all instances that were originally labeled with $Class^i$).
    Temporary label with $TempClass^2$ all instances that were originally labeled with some class different than $Class^i$).
    Calculate the WoE parameters for all instances and all their nominal features using the temporary labels ($TempClass^1$ and $TempClass^2$).
    Transform all $k$ nominal features using the calculated WoE parameters in the previous step. This produces $k$ new numeric features.
    Add the $k$ generated numeric features to the transformed dataset.
    Remove the temporary labels of all instances and revert them to their original labels (classes).
end for
```

$$z_{woe} = \begin{cases} m \times n, & m > 2 \\ n, & m = 2 \end{cases} \tag{15}$$

The same algorithm can be applied for transformation of the numeric attributes in the original dataset into another numeric features as well. However, given the fact that there are plenty of algorithms for transformation of numeric features, we are not focusing on that kind of application of the WoE transformation.

VI. Results

In this section we present the experiments that we conducted using the proposed method for data transformation. The first four subsections describe the performance metric and the cross-validation process that we used across all subsequent tests, how and when we performed feature selection and how we evaluated the performance. The following subsections describe how we have applied the weight of evidence transformation on the nominal features in some of the datasets obtained from the UCI Repository of Machine Learning Databases [24].

A. Performance metric

The choice of performance metric to evaluate any transformation is very important. Several research papers indicate the fact that in some cases for a given dataset, the learning method that obtains the best model according to a given measure, is not the best method if a different measure is used. In [25] is shown that Naive Bayes and pruned decision trees are very similar in predictive accuracy. Later on, applying the same algorithms, in [26] the same authors show that Naive Bayes is significantly better than pruned decision trees in terms of AUC ROC. As it is said in [27] the different results cannot be explained by slightly different implementations or variants of machine learning algorithms, but on the fact that the two measures (accuracy and AUC ROC) evaluate different things. The predictive accuracy is perhaps the most popular metric for classification problems and many researchers have been publishing papers that show the performance of various algorithms, techniques and transformations in terms of predictive accuracy on the same datasets that we also use in this paper. Another property of accuracy is that it can be calculated for multi-class problems in the same way as it is calculated for binary problems, while other metrics does not natively support multi-class problems. Because of these reasons we have also decided to compare our results in terms of predictive accuracy.

Additionally, we wanted to compare the execution times of both transformations so we can have insight of that aspect too. We have implemented all algorithms in Matlab and all tests have been performed on Windows 7 Professional SP1 64 bit running on a virtual Intel Xeon X5680 at 3.33 Ghz with 8 GB RAM. We have used the built-in implementations of FFNN and SVMs in Matlab. All listed execution times consist of the data transformation, the training of the machine learning algorithm and making the predictions.

B. Cross-validation

Some of the available datasets in the research community are consisted of training, validation and testing subset. However more often they are consisted of only one set that should be used for training, validation and testing, so the task of splitting the original dataset is left to the data analysts and researchers. The most common practice when evaluating performance in such cases is to perform cross-validation. There are few alternatives of how can cross-validation be performed and each of them has advantages and disadvantages. Stratified 10-fold cross-validation in [28] is recommended as the best model selection method, as it tends to provide less biased estimation of the accuracy. Following this recommendation we have decided to perform k-fold cross-validation while using different values for k: 2, 4, 6, 8 and 10. Usually the cross-validation is performed after the data is preprocessed. This means that all data cleaning, data transformations, and removal of outliers have to be performed, and then different subsets from the processed datasets are selected as training, validation and testing. This approach is suitable when the data transformations depend only on the actual values of features that should be transformed. Such transformations techniques are all mathematical functions that can be applied on numerical features or generation of dummy features from nominal features.

However, if the data transformations additionally depend on the relationship between the instances in the training set this approach is not suitable. Such relationship is present in the Weight of Evidence transformation because the transformed values depend on the set of values of the features that are being transformed and the set of values of all other instances in the dataset, as it is evident in (2). To review, the transformed value depends on the number of instances labeled with a particular class, and the number of instances that have a particular value of the original feature and are labeled with a particular class. These counts can vary for different subsets of the original dataset, thus a different transformed values can be obtained for the same original value. Ideally, if the dataset is large enough and if the distribution of values of the nominal features is nearly uniform, then the transformed values for different subsets of the dataset would not vary much. However, such conditions do not occur in real datasets, hence the need of to modify the cross-validation process. Algorithm 2 describes how the k-fold cross-validation is performed with repeated transformation of nominal features. By applying this algorithm for cross-validation, only the information that is present in the current set of training folds is used for data transformation.

Algorithm 2 K-fold cross-validation with repeated transformation of nominal features

```
Randomly shufle the dataset
Randomly assign $Fold_i$ ($i = 1 \to k$) to each instance in the shuffled dataset
**for** $i = 1 \to k$ **do**
    Assign all instances belonging to $Fold_i$ to $TestSet_i$.
    Assign all instances not belonging to $Fold_i$ to $TrainingSet_i$.
    Calculate $WoE_i$ parameters for all nominal features in $TrainingSet_i$.
    Transform all values of all nominal features in $TrainingSet_i$ using the transformation values $WoE_i$, thus obtaining $TransformedTrainingSet_i$.
    Transform all values of all nominal features in $TestSet_i$ using the transformation values $WoE_i$, thus obtaining $TransformedTestSet_i$.
    Train $Model_i$ using $TransformedTrainingSet_i$.
    Validate $Model_i$ using $TransformedTestSet_i$, thus obtaining $Results_i$.
**end for**
Aggregate $Results_i (i = 1 \to k)$
```

C. Feature selection

In the machine learning literature there are a lot of published papers and books about various feature selection techniques, both for nominal and numeric features. Such paper is [29], where the most popular algorithms are described and illustrated with examples. This work focuses on data transformation techniques and therefore we have performed some basic feature selection, and have not tested more advanced methods. First thing that is performed for all datasets that we have worked on, all single-value features were removed. Also if such features were generated with one of the applied data transformation techniques they were also removed. This is because these features have no information value i.e. their entropy is zero, therefore having no predictive power regardless of which machine learning algorithm would later be applied.

In the case when some nominal feature has many different values and the dummy variables generation technique is used, a lot of dummy features would be generated. This can pose a serious problem for many machine learning algorithms because they would demand enormous amount of computer power, even to the extent that they would not be usable. To overcome this problem we have used a simple feature selection that restricts the number of dummy features that can be generated. Namely, only for the values of the nominal features that occur in more than 5% of the instances a dummy feature is generated, while for all other less frequently occurring values we generate only one common dummy feature. Choosing the value of 5% as a threshold provided good balance between retaining a small number of features while not loosing too much information. This simple algorithm is very effective in helping prevent the generation of vast amount of dummy features. Of course, a more intelligent feature selection can be applied but we did not want to defocus from the main topic of the paper.

D. Performance evaluation

After applying the proposed transformation we have transformed all nominal features in the datasets into numeric and then we have trained few machine learning algorithms. In order to have basis for comparison, we have also generated dummy features from all nominal features in the original datasets and afterwards we have trained the same machine learning algorithms. To decrease the impact of randomness while making the splits of the data into cross-validation folds we have repeated the whole process multiple times and then we have aggregated the results. The following subsections describe each of the datasets, the performed data transformations and the performance in more detail. The performance is analyzed in terms of accuracy and execution time.

E. The Annealing dataset

The annealing dataset was obtained from the UCI Repository of Machine Learning Databases [24] and it contains data for a classification problem. More particularly, it contains 798 instances described with 6 numeric and 32 nominal features. All instances are labeled with one of the 5 possible classes. Only the nominal features of the dataset were subject to transformation, while the numeric features were not transformed in any way.

First, we have generated dummy features for all different values of all nominal features in the original dataset. Note that for nominal features that have only 2 different values we do not generate dummy features rather we only convert their values to 0s and 1s, because these features are already dummy features with differently encoded values. By doing that we have generated 64 dummy features. From them 7 had the same value for all instances in the training and test sets, therefore they were removed. Finally, together with the original 6 numeric features the dataset 1 is comprised of 63 numeric features.

Then we have applied the proposed WoE transformation thus obtaining $32 \times 5 = 160$ new numeric features, as defined with (15). From them 40 had the same value for all instances in the training and test sets, meaning that there is no information value in them, therefore they were removed. Finally, together with the original 6 numeric features the dataset 2 is comprised of 126 numeric features.

Both datasets were tested using a feed forward back propagation neural network (FFNN), and SVM with a linear, quadratic, polynomial and RBF kernel. The training and test partitions of the datasets were obtained using k-fold cross validation and 2, 4, 6, 8 and 10 were used as k values. An important thing to mention is that a same value of the original feature can be transformed into different values. This situation arises from the nature of the WoE values - they depend not only on the original values but also on the distribution of the other values of the same feature in the training dataset. This in turn require the WoE transformation to be performed for each training fold separately. In fact for k-fold cross validation the WoE transformation would be performed k times, once for each training fold combination. Because the instances that belong to the folds are chosen randomly, the performance can vary and may not be consistent. Therefore, the whole process was repeated 10 times for each value of k. At the end we have calculated the average, minimum, maximum and standard deviation of the accuracies and the execution times for each algorithm and each transformation type using the data from the 10 repetitions.

Table I shows the accuracies for different values of k of the 10 repetitions when a FFNN was trained with both datasets. We can see that for all values of k the WoE transformed dataset produced better average accuracy. Next, table II shows the execution times when a FFNN was trained with both datasets. We can see that both transformations together with the training and test phase needed were completed in similar time, even though the WoE transformed dataset had about twice more features.

The next algorithm we trained was SVM with linear kernel. Tables III and IV show the results of this algorithm using both datasets. For it we can note that both the accuracy and execution time are similar to each other for both transformations, but are much better than the FFNN.

After that, we have trained a SVM with RBF (i.e. Radial Basis Function) kernel. Tables V and VI show the results of this algorithm using both datasets. For this algorithm we can note that both the accuracy and the execution time are similar to each other for both transformations, are significantly better than the FFNN and worse than the SVM with linear kernel.

The following algorithm that we have trained was a SVM with Polynomial kernel. Tables VII and VIII show the results of this algorithm using both datasets. For this algorithm we can note that the accuracy is slightly better when dummy transfomation is used and execution time is slightly better when WoE transformation is used. Overall the performance is similar to the SVM with linear kernel and to the performance of SVM with Quadratic kernel, which are shown with tables IX and X.

Tables XI and XII show the results of a SVM with MLP (i.e. multilayer perceptron) kernel using both datasets. For this algorithm we can note that the accuracy is slightly better when dummy transfomation is used and execution time is slightly better when WoE transformation is used. Overall the performance is similar to the SVM with RBF kernel.

Finally, we can conclude that for this particular dataset both transformations lead to similar performance in terms of accuracy and execution time when various machine learning algorithms are applied. We want to point out again the fact that the dataset has 1000 instances that are non-uniformly distributed into 5 classes. As a consequence the execution of the transformations is very fast and therefore they are very similar. Additionally the number of different values of the nominal features is very small therefore the advantages of the WoE transformation can not be exploited. For datasets like this one, where the number of instances, the number of nominal features and the number of different values are fairly small, both transformations provide similar results. However, in such cases we recommend applying the dummy transformation because it is easier to implement and it is a lot simpler to interpret and understand.

TABLE I. ANNEALING DATASET. *Accuracy from 10 repetitions of k-fold cross-validation with* ***FFNN***

Dummy transformation	k=2	k=4	k=6	k=8	k=10
Min	0.7617	0.7628	0.7718	0.7862	0.7239
Max	0.8307	0.8364	0.8342	0.8273	0.8230
Mean	0.7951	0.8025	0.7981	0.8081	0.7882
StDev	0.0244	0.0229	0.0191	0.0127	0.0242

WoE transformation	k=2	k=4	k=6	k=8	k=10
Min	0.7550	0.7728	0.7030	0.7917	0.7773
Max	0.8408	0.8954	0.8731	0.8375	0.8430
Mean	0.7989	0.8258	0.8114	0.8116	0.8113
StDev	0.0258	0.0335	0.0420	0.0139	0.0207

TABLE II. ANNEALING DATASET. ***Execution time*** *in seconds from 10 repetitions of k-fold cross-validation with* ***FFNN***

Dummy transformation	k=2	k=4	k=6	k=8	k=10
Min	1.75	4.56	7.32	10.58	13.45
Max	9.48	6.25	10.99	12.66	18.95
Mean	2.83	5.43	9.02	11.62	14.88
StDev	2.23	0.53	0.96	0.63	1.49

WoE transformation	k=2	k=4	k=6	k=8	k=10
Min	1.96	5.33	8.64	12.12	14.35
Max	2.74	6.45	10.27	13.38	18.56
Mean	2.31	5.89	9.21	12.63	15.93
StDev	0.26	0.37	0.44	0.41	1.25

TABLE III. ANNEALING DATASET. *Accuracy from 10 repetitions of k-fold cross-validation with* ***SVM with linear kernel***

Dummy transformation	k=2	k=4	k=6	k=8	k=10
Min	0.9755	0.9866	0.9677	0.9732	0.9888
Max	0.9889	0.9922	0.9922	0.9922	0.9922
Mean	0.9825	0.9889	0.9884	0.9892	0.9906
StDev	0.0042	0.0017	0.0071	0.0054	0.0009

WoE transformation	k=2	k=4	k=6	k=8	k=10
Min	0.9755	0.9611	0.9866	0.9866	0.9900
Max	0.9911	0.9944	0.9944	0.9944	0.9922
Mean	0.9850	0.9882	0.9914	0.9919	0.9915
StDev	0.0041	0.0092	0.0025	0.0021	0.0007

TABLE IV. ANNEALING DATASET. ***Execution time*** *in seconds from 10 repetitions of k-fold cross-validation with* ***SVM with linear kernel***

Dummy transformation	k=2	k=4	k=6	k=8	k=10
Min	1.32	2.04	4.22	6.12	6.00
Max	2.07	3.80	6.21	10.46	13.36
Mean	1.49	2.67	5.44	7.69	9.06
StDev	0.21	0.49	0.62	1.38	1.92

WoE transformation	k=2	k=4	k=6	k=8	k=10
Min	1.55	3.34	4.74	5.75	8.22
Max	2.56	5.80	6.69	9.64	11.13
Mean	2.01	4.30	5.80	7.79	9.39
StDev	0.33	0.71	0.65	1.15	0.79

TABLE V. ANNEALING DATASET. *Accuracy from 10 repetitions of k-fold cross-validation with* ***SVM with RBF kernel***

Dummy transformation	k=2	k=4	k=6	k=8	k=10
Min	0.8664	0.8953	0.9053	0.9087	0.9064
Max	0.8953	0.9199	0.9198	0.9243	0.9243
Mean	0.8814	0.9081	0.9139	0.9180	0.9187
StDev	0.0086	0.0070	0.0044	0.0054	0.0050

WoE transformation	k=2	k=4	k=6	k=8	k=10
Min	0.8641	0.8942	0.9031	0.9076	0.9064
Max	0.8931	0.9176	0.9176	0.9242	0.9243
Mean	0.8782	0.9067	0.9126	0.9173	0.9177
StDev	0.0088	0.0067	0.0044	0.0054	0.0048

TABLE VI. ANNEALING DATASET. ***Execution time*** *in seconds from 10 repetitions of k-fold cross-validation with* ***SVM with RBF kernel***

Dummy transformation	k=2	k=4	k=6	k=8	k=10
Min	1.72	4.09	6.44	8.79	11.45
Max	2.03	4.41	7.10	11.14	15.50
Mean	1.83	4.29	6.73	9.44	12.05
StDev	0.08	0.10	0.24	0.60	1.16

WoE transformation	k=2	k=4	k=6	k=8	k=10
Min	1.77	4.15	6.26	9.08	11.14
Max	1.97	4.63	7.18	10.44	13.06
Mean	1.84	4.42	6.74	9.49	11.89
StDev	0.06	0.13	0.28	0.42	0.48

TABLE VII. ANNEALING DATASET. *Accuracy from 10 repetitions of k-fold cross-validation with* ***SVM with Polynomial kernel***

Dummy transformation	k=2	k=4	k=6	k=8	k=10
Min	0.9788	0.9889	0.9900	0.9900	0.9878
Max	0.9889	0.9933	0.9933	0.9933	0.9933
Mean	0.9860	0.9914	0.9922	0.9920	0.9916
StDev	0.0028	0.0020	0.0011	0.0010	0.0015

WoE transformation	k=2	k=4	k=6	k=8	k=10
Min	0.9577	0.9655	0.9654	0.9666	0.9655
Max	0.9755	0.9755	0.9733	0.9744	0.9710
Mean	0.9674	0.9701	0.9695	0.9698	0.9689
StDev	0.0056	0.0028	0.0024	0.0020	0.0019

TABLE VIII. ANNEALING DATASET. ***Execution time*** *in seconds from 10 repetitions of k-fold cross-validation with* ***SVM with Polynomial kernel***

Dummy transformation	k=2	k=4	k=6	k=8	k=10
Min	0.98	1.84	2.64	3.44	4.33
Max	1.11	2.14	2.89	4.35	4.93
Mean	1.04	1.97	2.78	3.71	4.50
StDev	0.04	0.08	0.09	0.24	0.19

WoE transformation	k=2	k=4	k=6	k=8	k=10
Min	0.89	1.67	2.40	3.23	3.96
Max	1.03	2.86	2.78	3.84	4.60
Mean	0.95	1.87	2.57	3.45	4.20
StDev	0.04	0.34	0.13	0.17	0.17

TABLE IX. ANNEALING DATASET. *Accuracy from 10 repetitions of k-fold cross-validation with* ***SVM with Quadratic kernel***

Dummy transformation	k=2	k=4	k=6	k=8	k=10
Min	0.9766	0.9911	0.9922	0.9900	0.9833
Max	0.9933	0.9956	0.9955	0.9956	0.9956
Mean	0.9874	0.9933	0.9948	0.9947	0.9934
StDev	0.0043	0.0014	0.0012	0.0017	0.0036

WoE transformation	k=2	k=4	k=6	k=8	k=10
Min	0.9744	0.9889	0.9922	0.9889	0.9844
Max	0.9900	0.9944	0.9944	0.9955	0.9956
Mean	0.9840	0.9918	0.9939	0.9929	0.9923
StDev	0.0043	0.0017	0.0009	0.0023	0.0034

TABLE X. ANNEALING DATASET. ***Execution time*** *in seconds from 10 repetitions of k-fold cross-validation with* ***SVM with Quadratic kernel***

Dummy transformation

	k=2	k=4	k=6	k=8	k=10
Min	0.97	1.73	2.41	3.33	4.05
Max	1.18	2.21	2.80	3.90	4.45
Mean	1.04	1.91	2.64	3.53	4.21
StDev	0.06	0.15	0.12	0.17	0.12

WoE transformation

	k=2	k=4	k=6	k=8	k=10
Min	0.87	1.61	2.30	3.07	3.63
Max	0.99	1.92	2.57	3.65	4.18
Mean	0.94	1.74	2.43	3.26	3.81
StDev	0.04	0.10	0.07	0.15	0.18

TABLE XI. ANNEALING DATASET. ***Accuracy*** *from 10 repetitions of k-fold cross-validation with* ***SVM with MLP kernel***

Dummy transformation

	k=2	k=4	k=6	k=8	k=10
Min	0.8664	0.8953	0.9053	0.9087	0.9064
Max	0.8953	0.9199	0.9198	0.9243	0.9243
Mean	0.8814	0.9081	0.9139	0.9180	0.9187
StDev	0.0086	0.0070	0.0044	0.0054	0.0050

WoE transformation

	k=2	k=4	k=6	k=8	k=10
Min	0.8641	0.8942	0.9031	0.9076	0.9064
Max	0.8931	0.9176	0.9176	0.9242	0.9243
Mean	0.8782	0.9067	0.9126	0.9173	0.9177
StDev	0.0088	0.0067	0.0044	0.0054	0.0048

F. The PAKDD 2010 dataset

The 14th Pacific-Asia Knowledge Discovery and Data Mining conference (PAKDD 2010) together with NeuroTech Ltd. and the Center for Informatics of the Federal University of Pernambuco (Brazil) co-organized a data mining competition [30]. This credit risk assessment problem comes from the private label credit card operation of a major retail chain. The company has been operating its private label card for over 10 years and has applied two different methods for risk assessment with the application's acceptance rate varying from 50% to 75% within this period. Each accepted application turns the applicant into a client and gives him/her the access to credit for purchasing on the retail chain to be billed 10 to 40 days after the purchase, on a monthly basis on a fixed month day. After his/her credit acceptance, a client would take some time to make their first purchase and receive their first bill. During the first year of using the card, the set of monthly bills and payment behavior is collected and used for credit risk assessment. If the client had any monthly defaults (delays longer than the agreed payment periods) he is labeled as bad, otherwise as good client. The goal is to exploit the information that was available when the applicant applied for credit and try to predict whether he would be a good or bad client. To achieve this we have used the training data consisting of various kinds of information for the applicants like age, profession, sex, marital status, monthly income etc. and the label (good/bad) to build prediction models.

TABLE XII. ANNEALING DATASET. ***Execution time*** *in seconds from 10 repetitions of k-fold cross-validation with* ***SVM with MLP kernel***

Dummy transformation

	k=2	k=4	k=6	k=8	k=10
Min	1.74	4.19	6.38	9.13	11.33
Max	1.94	4.82	7.09	10.22	12.67
Mean	1.84	4.42	6.70	9.42	11.76
StDev	0.07	0.22	0.19	0.29	0.34

WoE transformation

	k=2	k=4	k=6	k=8	k=10
Min	1.73	4.25	6.27	8.93	11.03
Max	2.01	6.07	7.09	11.28	12.43
Mean	1.87	4.65	6.69	9.59	11.77
StDev	0.08	0.49	0.24	0.62	0.39

For the purpose of the competition there are three available datasets, but the labels (i.e. target class) of only one of them are made publicly available. This dataset is named as Modeling and has 50000 instances, distributed as 26% vs. 74% per class. The dataset is real and collected manually during a long time period therefore some of the instances have missing, invalid or inconsistent data, some of the columns have no information value or have the same values for all instances etc. We have addressed these issues by removing some of the instances and the columns. Additionally, we have generated few nominal features that capture interactions between the original features: sex, marital status, age group, profession etc. After this stage of data cleaning and preparation we ended up with a dataset with 11 numeric and 24 nominal features and a total of about 42000 instances that were similarly distributed in respect to the target class.

First, we have generated dummy features for all different values of all nominal features in the original dataset. Note that for nominal features that have only 2 different values we do not generate dummy features rather we only convert their values to 0s and 1s, because these features are already dummy features with differently encoded values. By doing that we have obtained around 4630 dummy features. Obviously a dataset with such high number of features makes it difficult to train models on it. That is why we have performed a simple feature selection as described in subsection VI-C, which resulted in a dataset with about 140 dummy features. Finally, together with the original 11 numeric features the dataset 1 is comprised of about 150 numeric features. The exact number of features varies depending on the random split during the cross-validation.

Then we have applied the proposed WoE transformation and we generated for each nominal features one numeric feature which resulted in 24 new numeric features, as defined with (15). Together with the original 11 numeric features the dataset 2 is comprised of 35 numeric features.

Both datasets were tested using a feed forward back propagation neural network, and SVM with a linear, quadratic, polynomial and RBF kernel. The training and test partitions of the datasets were obtained using k-fold cross validation and 2, 4, 6, 8 and 10 were used as k values. The whole process was repeated 10 times for each value of k, so the influence of randomness during the cross-validations can be neglected. The official performance metric of the competition is area under the receiver operating curve (AUC ROC) [26], but due to the reasons discussed in subsection VI-A, we have decided to compare the results in terms of accuracy. The first results were not very useful because the accuracy of the classifiers built from both datasets was in the most cases around 74%. Additionally in many cases the machine learning algorithms could not converge, meaning we could not compare the both transformations. The WoE transformed dataset usually provided better accuracy, but the overall results were insufficient to make firm conclusions. The fact that the accuracy was similar to the percentage of the more common class and the confusion matrices undoubtedly showed that the unbalanced dataset introduces serious problems for all machine learning algorithms. The algorithms trained on the dummy transformed dataset converged twice less often than the ones trained on the WoE transformed dataset, and in the rare cases when they both converged the WoE transformation almost always provided better results.

In order to make better comparisons of the transformations, it was obvious that a balanced dataset would be more suitable. While we can artificially generate more instances of the less common class, the most common way to balance a dataset is by "throwing away" some of the instances labeled with the more common class. When data is balanced, accuracy rates tend to decline [31]. If we balance the dataset by reducing the training set size, then this can lead to the degeneracy of the model because we are neglecting potentially useful training instances. Nevertheless, we have opted for this option because it was easier to implement the shrinking, but mostly because this way we can train algorithms much faster because of the smaller training sets. By doing this we have obtained a balanced dataset with around 22000 instances and afterwards we have performed all tests described in the previous paragraph. Important to mention here is that by repeating the whole tests 10 times we were able to select different subsets of instances of the more common class for each repetition, thus mitigating the issues of thrown-away information to some extent.

From all algorithms that were tested only the FFNN converged for all values of k and all repetitions. The achieved results are shown in tables XIII and XIV. We can conclude that when FFNN was trained the WoE transformation produced 6% to 9% better accuracy in 10% to 40% more time than the dummy transformation.

After the training was performed for the SVM with RBF kernel we noticed that convergence could not be acheived when k is 4, 6, 8 and 10 when the dummy transformed dataset is used. Similary, when the WoE transformed dataset was used the SVM converged when k was 2 and in six of the repetitions when k was 4. The results that we were able to collect are listed in tables XV and XVI. The results for SVM with MLP kernel were so similar that the averages look exactly the same despite the fact that the individual values were different. This can be seen in tables XVII and XVIII. From them we can conclude that when SVM with RBF or MLP kernel was trained the WoE transformation produced about 2% better accuracy about three times faster than the dummy transformation and additionally the SVMs were able to converge more often. Compared to the the FFNN, the SVMs produced about 20% better accuracy for $k = 2$ but were significantly slower - about 6 times.

The other SVMs with linear, polynomial and quadratic kernel could not converge for none of the repetitions and different k values. We realize that these issues might be addressed by parameter tuning, but we cannot concentrate on this issue at the moment because it is not the focus point of this paper.

TABLE XIII. **PAKDD 2010 DATASET.** ***Accuracy*** *from 10 repetitions of k-fold cross-validation with* ***FFNN***

Dummy transformation	k=2	k=4	k=6	k=8	k=10
Min	0.5122	0.5443	0.5430	0.5478	0.5536
Max	0.5944	0.5900	0.5903	0.5935	0.5860
Mean	0.5547	0.5711	0.5697	0.5739	0.5706
StDev	0.0246	0.0144	0.0147	0.0149	0.0088

WoE transformation	k=2	k=4	k=6	k=8	k=10
Min	0.5028	0.5472	0.5610	0.5868	0.5702
Max	0.6453	0.6561	0.6596	0.6377	0.6567
Mean	0.5894	0.6214	0.6158	0.6111	0.6123
StDev	0.0504	0.0328	0.0262	0.0149	0.0227

TABLE XIV. **PAKDD 2010 DATASET.** ***Execution time*** *in seconds from 10 repetitions of k-fold cross-validation with* ***FFNN***

Dummy transformation	k=2	k=4	k=6	k=8	k=10
Min	29.59	84.85	141.67	183.26	242.82
Max	45.16	103.84	309.29	223.04	272.56
Mean	35.54	93.86	169.14	207.04	257.37
StDev	4.67	5.61	47.23	12.04	8.53

WoE transformation	k=2	k=4	k=6	k=8	k=10
Min	41.54	104.24	168.24	251.99	302.14
Max	48.07	297.96	200.57	279.83	388.43
Mean	44.21	135.97	190.84	267.72	338.81
StDev	1.69	54.34	9.04	6.94	21.50

TABLE XV. **PAKDD2010 BALANCED DATASET.** ***Accuracy*** *from 10 repetitions of k-fold cross-validation with* ***SVM with RBF kernel***

Dummy trans.	k=2	WoE trans.	k=2	k=4
Min	0.7300	**Min**	0.7409	0.8044
Max	0.7400	**Max**	0.7531	0.8089
Mean	0.7336	**Mean**	0.7453	0.8063
StDev	0.0029	**StDev**	0.0035	0.0015

TABLE XVI. **PAKDD2010 BALANCED DATASET.** ***Execution times*** *in seconds from 10 repetitions of k-fold cross-validation with* ***SVM with RBF kernel***

Dummy trans.	k=2	WoE trans.	k=2	k=4
Min	713.34	**Min**	254.66	851.89
Max	968.45	**Max**	274.56	1054.31
Mean	803.70	**Mean**	262.75	954.75
StDev	82.49	**StDev**	6.22	88.40

TABLE XVII. **PAKDD2010 BALANCED DATASET.** ***Accuracy*** *from 10 repetitions of k-fold cross-validation with* ***SVM with MLP kernel***

Dummy trans.	k=2	WoE trans.	k=2	k=4
Min	0.7300	**Min**	0.7409	0.8044
Max	0.7400	**Max**	0.7531	0.8089
Mean	0.7336	**Mean**	0.7453	0.8063
StDev	0.0029	**StDev**	0.0035	0.0015

TABLE XVIII. **PAKDD2010 BALANCED DATASET.** ***Execution times*** *in seconds from 10 repetitions of k-fold cross-validation with* ***SVM with MLP kernel***

Dummy trans.	k=2	WoE trans.	k=2	k=4
Min	713.34	**Min**	254.66	851.89
Max	968.45	**Max**	274.56	1054.31
Mean	803.70	**Mean**	262.75	954.75
StDev	82.49	**StDev**	6.22	88.40

VII. Conclusion

In this study we have proposed a data transformation method based on the weight of evidence parameter. This technique is applicable in binary and multivariate supervised learning problems particularly for the nominal and categorical features. We have tested this technique on two real datasets. To verify our results, we have also generated dummy features from all nominal features in the same datasets and afterwards we have trained the same machine learning algorithms (i.e. feed forward neural networks and support vector machines with different kernels).

The analysis of the results show that in datasets in which the number of instances, the number of nominal features and the number of different values are fairly small, both transformations provide similar results in terms of predictive performance and execution time. For this reason in such cases we recommend applying the dummy transformation, because is easier to implement and it is a lot simpler to interpret and understand. However, it was the opposite case when we applied both transformations to a significantly larger dataset (with more than 10000 instances) that has more nominal features and they have more different values. Very often machine learning algorithms could not be trained on the dummy transformed dataset because of the memory complexity or because convergence could not be achieved. On the same dataset, but transformed with WoE, the same algorithms achieved convergence and produced significantly better results both in terms of predictive performance and execution time.

The presented method can be used without prior knowledge of the nature of the datasets. In our future work, we plan to compare the WoE transformation with other techniques for data transformations for nominal features. Also we will train a larger set of classifiers and will apply these transformations on other datasets as well. Additionally when we train the classifiers we also need to investigate the effect of parameter tuning, for instance when training SVMs. Our goal is to make firm and well-explained conclusions of which transformation is most suitable for what kinds of datasets, so researchers and practitioners can make apply these transformations more confidently without having to try out all possible combinations before deciding which one is most suitable.

Acknowledgment

This work was partially financed by the Faculty of Computer Science and Engineering at the Ss.Cyril and Methodius University, Skopje, Macedonia.

References

[1] C. Shearer, "The crisp-dm model: the new blueprint for data mining," *Journal of Data Warehousing*, vol. 5, no. 4, pp. 13–19, 2000.

[2] R. Anderson, *The credit scoring toolkit: theory and practice for retail credit risk management and decision automation*. Oxford: Oxford University Press, 2007. ISBN 9780199226405

[3] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, Nov. 2009. doi: 10.1145/1656274.1656278. [Online]. Available: http://doi.acm.org/10.1145/1656274.1656278

[4] E. Tuv and G. Runger, "Scoring levels of categorical variables with heterogeneous data," *Intelligent Systems, IEEE*, vol. 19, no. 2, pp. 14–19, Mar 2004. doi: 10.1109/MIS.2004.1274906

[5] M. Hofmann and R. Klinkenberg, Eds., *RapidMiner: data mining use cases and business analytics applications*, ser. Chapman & Hall/CRC data mining and knowledge discovery series. Boca Raton: CRC Press, 2014, no. 33. ISBN 9781482205497

[6] T. W. Miller, *Modeling techniques in predictive analytics: business problems and solutions with R*. Upper Saddle River, New Jersey: Pearson Education, Inc, 2014. ISBN 9780133412932

[7] M. Deza, *Encyclopedia of distances*. Dordrecht : New York: Springer Verlag, 2009. ISBN 9783642002335

[8] D. W. Goodall, "A new similarity index based on probability," *Biometrics*, vol. 22, no. 4, pp. pp. 882–907, 1966. [Online]. Available: http://www.jstor.org/stable/2528080

[9] C. Li and G. Biswas, "Unsupervised learning with mixed numeric and nominal data," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 14, no. 4, pp. 673–690, Jul 2002. doi: 10.1109/TKDE.2002.1019208

[10] S. Robertson, "Understanding inverse document frequency: on theoretical arguments for idf," *Journal of Documentation*, vol. 60, no. 5, pp. 503–520, 2004. doi: 10.1108/00220410410560582. [Online]. Available: http://dx.doi.org/10.1108/00220410410560582

[11] H. C. Wu, R. W. P. Luk, K. F. Wong, and K. L. Kwok, "Interpreting tf-idf term weights as making relevance decisions," *ACM Trans. Inf. Syst.*, vol. 26, no. 3, pp. 13:1–13:37, Jun. 2008. doi: 10.1145/1361684.1361686. [Online]. Available: http://doi.acm.org/10.1145/1361684.1361686

[12] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *Machine Learning: ECML-98*, ser. Lecture Notes in Computer Science, C. Nédellec and C. Rouveirol, Eds. Springer Berlin Heidelberg, 1998, vol. 1398, pp. 137–142. ISBN 978-3-540-64417-0. [Online]. Available: http://dx.doi.org/10.1007/BFb0026683

[13] I. J. Good, *Probability and the Weighing of Evidence*. C. Griffin & Co., London, UK, 1950.

[14] E. P. Smith, I. Lipkovich, and K. Ye, "Weight-of-evidence (woe): Quantitative estimation of probability of impairment for individual and multiple lines of evidence," *Human and Ecological Risk Assessment: An International Journal*, vol. 8, no. 7, pp. 1585–1596, 2002. doi: 10.1080/20028091057493. [Online]. Available: http://dx.doi.org/10.1080/20028091057493

[15] E. Zdravevski, P. Lameski, and A. Kulakov, "Weight of evidence as a tool for attribute transformation in the preprocessing stage of supervised learning algorithms," in *Neural Networks (IJCNN), The 2011 International Joint Conference on*, July 2011. doi: 10.1109/IJCNN.2011.6033219. ISSN 2161-4393 pp. 181–188.

[16] E. Zdravevski, P. Lameski, A. Kulakov, and D. Gjorgjevikj, "Feature selection and allocation to diverse subsets for multi-label learning problems with large datasets," in *Computer Science and Information Systems (FedC-*

SIS), 2014 Federated Conference on, Sept 2014. doi: 10.15439/2014F500 pp. 387–394.

[17] D. B. Suits, “Use of dummy variables in regression equations,” *Journal of the American Statistical Association,* vol. 52, no. 280, 1957.

[18] M. A. Hardy, *Regression with dummy variables,* ser. *Sage university papers series.* Newbury Park: Sage Publications, 1993, no. no. 07-093. ISBN 0803951280

[19] N. Chater and M. Oaksford, Eds., *The probabilistic mind: prospects for Bayesian cognitive science.* Oxford ; New York: Oxford University Press, 2008. ISBN 9780199216093

[20] E. Zdravevski, P. Lameski, and A. Kulakov, “Towards a general technique for transformation of nominal features into numeric features in supervised learning,” in *Proceedings of the 9th Conference for Informatics and Information Technology (CIIT 2012).* Faculty of Computer Science and Engineering (FCSE) and Computer Society of Macedonia, 2012.

[21] E. L. Allwein, R. E. Schapire, and Y. Singer, “Reducing multiclass to binary: A unifying approach for margin classifiers,” *The Journal of Machine Learning Research,* vol. 1, pp. 113–141, 2001.

[22] R. Rifkin and A. Klautau, “In defense of one-vs-all classification,” J. Mach. Learn. Res., vol. 5, pp. 101–141, Dec. 2004. [Online]. Available: http: //dl.acm.org/citation.cfm?id=1005332.1005336

[23] E. Zdravevski, P. Lameski, and A. Kulakov, “Advanced transformations for nominal and categorical data into numeric data in supervised learning problems,” in *Proceedings of the 10th Conference for Informatics and Information Technology (CIIT 2013).* Faculty of Computer Science and Engineering (FCSE) and Computer Society of Macedonia, 2013.

[24] K. Bache and M. Lichman, “UCI machine learning repository,” 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[25] J. Huang, J. Lu, and C. Ling, “Comparing naive bayes, decision trees, and svm with auc and accuracy,” in *Data Mining,* 2003. ICDM 2003. Third IEEE International Conference on, Nov 2003. doi: 10.1109/ICDM.2003.1250975 pp. 553–556.

[26] J. Huang and C. Ling, “Using auc and accuracy in evaluating learning algorithms,” *Knowledge and Data Engineering, IEEE Transactions on,* vol. 17, no. 3, pp. 299–310, March 2005. doi: 10.1109/TKDE.2005.50

[27] C. Ferri, J. Hernndez-Orallo, and R. Modroiu, “An experimental comparison of performance measures for classification,” *Pattern Recognition Letters,* vol. 30, no. 1, pp. 27 – 38, 2009. doi: http://dx.doi.org/10.1016/j.patrec.2008.08.010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167865508002687

[28] R. Kohavi, “A study of cross-validation and bootstrap for accuracy estimation and model selection,” in *Proceedings of the 14th International Joint Conference on Artificial Intelligence* - Volume 2, ser. IJCAI’95. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995. ISBN 1-55860-363-8 pp. 1137–1143. [Online]. Available: http://dl.acm.org/citation.cfm?id=1643031.1643047

[29] I. Guyon and A. Elisseeff, “An introduction to variable and feature selection,” *J. Mach. Learn. Res.,* vol. 3, pp. 1157–1182, Mar. 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=944919.944968

[30] “Pacific-asia knowledge discovery and data mining competition 2010,” http://sede.neurotech.com.br/PAKDD2010/, accessed: 2015-06-05.

[31] D. Olson, “Data set balancing,” in *Data Mining and Knowledge Management, ser. Lecture Notes in Computer Science,* Y. Shi, W. Xu, and Z. Chen, Eds. Springer Berlin Heidelberg, 2005, vol. 3327, pp. 71–80. ISBN 978-3-540-23987-1. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-30537-8 8