CLASSIFICATION OF AN IMBALANCED DATA SET USING DECISION TREE ALGORITHMS

Ciprian-Octavian TRUICĂ[1] , Cătălin Adrian LEORDEANU[2]

Machine learning algorithms have recently become very popular for different tasks involving data analysis, classification or prediction. They can provide valuable knowledge for very large sets of data and can reach very good accuracy. However, most algorithms are sensitive to the nature of the data sets, as well as different calibrations which can lead to large differences in performance, accuracy or false positives. In this paper, a classification solution for imbalanced data sets containing information about defects of various trees is presented. The experimental results present a comparison that evaluates the classification performance of the Decision Tree, Random Forest, and Extremely Randomized Trees classifiers. The measures used in the comparison take into account weighted accuracy, precision, and recall for binary and multi-class classification.

Keywords: imbalanced data set classification, Decision Tree Classifier, Random Forest Classifier, Extremely Randomized Trees Classifier

1. **Introduction**

Machine Learning has become ubiquitous in data analysis applications. It has helped the proliferation of Big Data, as well as new technologies in different fields, such as automotive, smart homes and many others [4].

There are two types of Machine Learning: supervised and unsupervised. The difference is that in the case of supervised machine learning the algorithm needs a training data set in order to generate a model which will be subsequently used for the input. In contrast, unsupervised learning has no training data set and relies on the composition of input data to generate the output. Supervised algorithms can be divided into classification and regression algorithms, depending on their type of output.

This paper focuses on supervised classification algorithms for a data set containing information about trees in an urban environment and proposes a

1 Ph.D. student, Computer Science and Engineering Department, Faculty of Automatic Control and Computers, University Politehnica of Bucharest, Bucharest, Romania, e-mail: `ciprian.truica@cs.pub.ro`

2 Lecturer, Ph.D., Computer Science and Engineering Department, Faculty of Automatic Control and Computers, University Politehnica of Bucharest, Bucharest, Romania, e-mail: `catalin.leordeanu@cs.pub.ro`

solution for the classification of trees around the city of Grenoble. The dataset is publicly available and it was also used in the EGC (Extraction et Gestion des Connaissances) Competition in 2017 [3]. The trees are classified according to their health or various defects which were observed in their collar, crown, root or trunk.

The proposed solution is based on Decision Trees, as they are the most appropriate approach to this situation. Since the number of trees in each class is obviously highly imbalanced, the solution proposed in this paper is based on specific weights which correct the accuracy of the classification algorithms.

This paper is structured as follows. Section 2 presents the state of the art and current research in the field of supervised learning and classification and describes the algorithms used for classifying the imbalanced data set. Section 3 presents the measures used for evaluating the classification. Section 4 presents the data set, describes its attributes and the imbalance between the classes. Section 5 presents the implementation of the algorithms and their evaluation. Section 6 presents and discusses the results and finally, Section 7 presents the conclusions and hints at future work.

2. **Decision Tree Classifiers and Extensions**

Decision Tree Classifiers are one of the most used algorithms for classification because they have a good accuracy when compared to other machine learning algorithms [11]. For this approach, each node of the tree corresponds to an input value for the dataset. The Decision Tree is built by partitioning the training dataset until the subsets contain only data belonging to a single class.

The most used algorithms for building decision trees are ID3 (Iterative Dichotomiser 3) [14] and its extensions C4.5 [15] and C5.0/See5, CART (Classification and Regression Trees) [1] and CHAID (CHi-squared Automatic Interaction Detector) [10]. ID3 computes the entropy for every attribute and splits the data set using the attribute with the minimum entropy or, equivalently, the maximum information gain. C4.5 finds the attribute with the maximum normalized information gain, which it uses to split the data set. Moreover, pruning is used to minimize the tree by removing branches that bring no additional information with leaves. The C5.0/See5 algorithm improves C4.5 by adding boosting to improve the tree and give a better accuracy. CART is another decision tree learning method that recursively partitions the data space and fits a simple prediction model within each partition. The CHAID algorithm builds a decision tree that performs multi-level splits when computing the classification trees.

Ho proposed the first algorithm that builds multiple decision trees in randomly selected subspaces of the features space [8]. Although the complexity is low due to the fact that trees tend to be overly adapted to the training data and produce over-fitting, a benefit of using a forest of decision trees is the speed of execution. This method manages to generalize and improve the classification process by using different subspaces in complementary ways. Also, trees generalize independently so a discrimination function is used to combine the classification given by each individual tree. Experimental results show that using this method a higher generalization than using simple decision trees is achieved.

Ho [9] extended his work on Random Forest with a new method to construct decision trees based classifiers that maintain the highest accuracy of training data and improves on generalization accuracy as it grows in complexity. Thus, the study aims to build classifiers whose capacity can be expanded randomly to increase the classification accuracy by constructing several decision trees in randomly selected subspaces. The experiments test different types of splits to determine which approach is better: i) axis parallel linear split, ii) oblique linear splits, and iii) piecewise linear splits (Voronoi tessellation). To improve the quality of the experimental results, the study also employs training set sub-sampling methods: bootstrapping and boosting. The article concludes that the classification is invariant for points that are different from the training points only in unselected dimensions.

Breiman argues in his work [2] that classification accuracy is significantly improved if a voting process to choose the most popular classes are used. Random vectors that manage how much each tree grows are used to generate these voting assemblies. Moreover, the error of generalization for a Random Forest Classifier depends on the strength of each individual tree and the correlation between them

Geurts et al. [5] proposed an improved algorithm for building decision tree forests called Extremely Randomized Trees. At each node, the best splitting attribute is selected from a random subset of attributes. Using this process, the algorithm builds an unpruned ensemble of decision trees.

3. Evaluation Measures

Different measures can be used to evaluate if the classification is done correctly [17]. All these measures use the confusion table (Table 1).

Table 1

Confusion Table for binary classification

Data class	Classified as Positive	Classified as Negative
Actual Positive	TP (True Positive)	FN (False Negative)
Actual Negative	FP (False Positive)	TN (True Negative)

The main three measures used to evaluate a binary classifier are Accuracy (A), Precision (P) and Recall (R) [17]. Accuracy measures the overall effectiveness of a classifier. Precision measures the class agreement of the data labels with the positive labels given by the classifier. Recall measures the effectiveness of a classifier to identify positive labels. Table 2 presents the formula for these measures.

Table 2

Confusion Table for binary classification

Measure	**Formula**
Accuracy	$A = \frac{TP+TN}{TP+TN+FP+FN}$
Precision	$P = \frac{TP}{TP+FP}$
Recall	$R = \frac{TP}{TP+FN}$

When dealing with multi-class classification, the Precision and Recall measures must be generalized to take into account the number of non-overlapping classes C_i, $i = \overline{1,N}$ where N is the number of classes. Average Accuracy ($avgA$) measures the average per-class effectiveness of a classifier. Micro Precision (μP) measures the per-class agreement of the data class labels with the positive labels given by the classifier. Micro Recall (μR) measures the per-class effectiveness of a classifier to identify positive labels. Macro Precision (MP) measures the average per-class agreement of the data class labels with those of the classifier. Macro Recall (MR) measure the average per-class effectiveness of a classifier to identify class labels. Macro-averaging treats all classes equally while micro-averaging favors bigger classes. Table 3 presents the formula for the generalized measures.

Table 3

Generalized multi-class classification measures

Measure	**Formula**
Average Accuracy	$avgA = \frac{1}{N} \cdot \sum_{i=1}^{N}(\frac{TP_i+TN_i}{TP_i+FN_i+FP_i+TN_i})$
Micro Precision	$\mu P = \frac{\sum_{i=1}^{N} TP_i}{\sum_{i=1}^{N}(TP_i+FP_i)}$
Micro Recall	$\mu R = \frac{\sum_{i=1}^{N} TP_i}{\sum_{i=1}^{N}(TP_i+FN_i)}$
Macro Precision	$MP = \frac{1}{N} \cdot \sum_{i=1}^{N}(\frac{TP_i}{TP_i+FP_i})$
Macro Recall	$MR = \frac{1}{N} \cdot \sum_{i=1}^{N}(\frac{TP_i}{TP_i+FN_i})$

The imbalanced data classification problem is concerned with the performance of machine learning algorithms in the presence of underrepresented data and severe class distribution skews [7]. Classification of such data sets has encountered a significant drawback of the performance attainable by most standard classifier learning algorithms which assume a relatively balanced class distribution and equal misclassification costs [18]. Moreover, classification evaluation methods fall short when dealing with such data and, for this reason, new methods must be used [6]. Therefore, the evaluation methods of a classifier must be adapted to take into account the weights of each class, and thus, in this paper, the weighted versions of Average Accuracy (wA), Precision (wP) and Recall (wR) are used. The weighted version of these measures calculates metrics for each label, and find their average, weighted by support (the number of true instances for each label - n_i). These measures alter the generalized macro versions of Precision and Recall (Table 4). These measures are also used for the binary classification.

Table 4

Generalized weighted Precision and Recall

Measure	**Formula**
Weighted Average Accuracy	$wA = \frac{1}{N} \cdot \sum_{i=1}^{N}(\frac{1}{n_i} \cdot \frac{TP_i+TN_i}{TP_i+FN_i+FP_i+TN_i})$
Weighted Precision	$wP = \frac{1}{N} \cdot \sum_{i=1}^{N}(\frac{1}{n_i} \cdot \frac{TP_i}{TP_i+FP_i})$
Weighted Recall	$wR = \frac{1}{N} \cdot \sum_{i=1}^{N}(\frac{1}{n_i} \cdot \frac{TP_i}{TP_i+FN_i})$

4. Data set description

4.1. Features description

The data set used for testing contains information about the trees around the city of Grenoble, each record has 27 features. These features can be split into four sets: general information related to the city plan, tree characteristics, location descriptors and tree health diagnostics. There is a total number of 15 375 classified entries in the data set. The data set is public and it is available on-line (Data set link `http://egc2017.imag.fr/defi/`).

Table 5 presents the city plan attributes and their description. Table 6 presents the tree characteristics. Table 7 presents the trees' location descriptors. Table 8 presents the tree health descriptors.

Table 5

City plan attributes

Attribute	**Description**
Code	unique identifier of the tree
IdentifiantPLU	unique identification number used by the local city planning comity
IntituleProtectionPLU	unique identification number used for identifying the protection category of the tree
TypeImplantionPLU	descriptor for the way the tree was planted.

Table 6

Tree description attributes

Attribute	**Description**
AnneeDePlantation	year when the tree was planted
DiametreArbreAUnMetre	tree stump diameter
Espece	tree species
Genre_Bota	botanical genus of the tree
StadeDeDeveloppenet	age descriptor with three values: young, mature and old
Variete	tree variety
Vigueur	tree vigor

Table 7

Location description attributes

Attribute	**Description**
Adr_Secteur	represents the 6 geographical areas of Grenoble
Code_Parent	each tree has associated a parent area that groups them by a geographical area and by species
Code_Parent_Desc	description of the parent area
FrequentationCible	describes if the sidewalk is used frequently
Sous_Categorie	unique identifier of the tree's category
Sous_Categorie_Desc	description of the tree's category
Trottoir	details if the tree is planted near a sidewalk

Table 8

Tree health attributes

Attribute	**Description**
AnneeRealisationDiagnostic	last year when the tree health was diagnosed
AnneeTravauxPrerecinisesDiag	recommended year for next health check
NoteDiagnostic	description of the diagnosis
PrioriteDeRenouvellement	when the next health check should occur
RaisonDePlantation	reason for the tree plantation
Remarques	remarks about the tree
StadeDeveloppementDiag	stage of development of the tree during the diagnosis
TraitementChenilles	priority of treatment for caterpillars on pine and cedar
TravauxPreconisesDiag	work recommended during the safety diagnosis

4.2. Feature engineering

Feature engineering is the process of using domain knowledge about the data set to create features that make machine learning algorithms work [16].

The features that describe the data set may contain some duplicate information or to not contribute to the classification at all. Based on this reasoning, the following attributes were removed before applying any classifier:

- Code just a unique identifier that adds no information to the classifier.
- Code_Parent_Desc is just a description of the CODE_PARENT feature.
- IdentifiantPLU just a unique identifier that adds no information to the classifier.
- Sous_Categorie_Desc is just a description of the SOUS_CATEGORIE feature.

4.3. Class description

A tree can be declared healthy if it has no problems related to the Collar, Crown, Root and Trunk, otherwise, it is considered that the tree has a defect. Based on this information, there can be two ways of classifying the data set: a binary classification and a multi-class classification. Tables 9 and 10 present the number of classes broken down by each defect and the number of record in the training set for each combination of defects.

The binary classification splits the data set between trees that have defects, a total number of 5 001 records, and the ones that do not, 10 374 records. There is an imbalance of 2:1 between healthy trees and trees with at least one defect (Table 9).

Table 9

Data set binary

Binary Class	Defect	No. Records
C_b0	0	10 374
C_b1	1	5 001

The multi-class is composed of a total of 16 classes that use all the permutations of defects that appear at the Collar, Crown, Root and Trunk level. Table 10 presents all the records in the training set for each of the classes. The imbalance, in this case, is evident, for example, the imbalance between $C_{mc}0$ and $C_{mc}7$ is 1:216.

5. Implementation

The Python Scikit-learn package is used for classification [13]. The Decision Tree, Random Forest, and Extremely Randomized Trees classifiers are used.

The *GridSearchCV(estimator, param_grid, scoring, cv)* class is used for validation. This class uses an exhaustive search over specified parameter values for a classifier. The first parameter (*estimator*) is the classifier for which the cross validation is done. The second parameter (*param_grid*) is a dictionary with parameters to pass to the classifier (Code Sample 1). The *params_dtc* is the dictionary with the parameters used by the Decision Tree Classifier, while *params_rec* is the dictionary with parameters used by the Random Forest and Extremely Randomize Trees classifiers.

Table 11 presents each parameter. The third parameter is the scoring function. The *scores_bc* array is used the binary classification, while the *scores_mc* one is used for the multi-class classification (Code Sample 2). The last parameter used for Cross Validation is *cv* which determines the cross-validation splitting strategy.

Table 10

Data set multi-class

Multi Classes	**Defect**	**Collar**	**Crown**	**Root**	**Trunk**	**No. Records**
$C_{mc}0$	0	0	0	0	0	10 374
$C_{mc}1$	1	0	0	0	1	963
$C_{mc}2$	1	0	0	1	0	243
$C_{mc}3$	1	0	0	1	1	48
$C_{mc}4$	1	0	1	0	0	2 091
$C_{mc}5$	1	0	1	0	1	539
$C_{mc}6$	1	0	1	1	0	147
$C_{mc}7$	1	0	1	1	1	41
$C_{mc}8$	1	1	0	0	0	203
$C_{mc}9$	1	1	0	0	1	129
$C_{mc}10$	1	1	0	1	0	49
$C_{mc}11$	1	1	0	1	1	51
$C_{mc}12$	1	1	1	0	0	135
$C_{mc}13$	1	1	1	0	1	219
$C_{mc}14$	1	1	1	1	0	73
$C_{mc}15$	1	1	1	1	1	70

```
params_grid = { "criterion": ['gini', 'entropy'],
    "max_features": ['auto', 'log2', None],
    "max_depth": [1, 5, 10, None],
    "min_samples_split": [2, 5, 10],
    "min_samples_leaf": [1, 5, 10],
    "class_weight": class_weights }
params_dtc = params_grid
params_rec = params_grid
params_dtc["splitter"] = ['best', 'random']
params_rec["n_estimators"] = [10, 100, 1000]
params_rec["bootstrap"] = [True, False]
```

Code Sample 1. Algorithm parameters

```
scores_bc=['accuracy', 'precision_weighted', 'recall_weighted']
scores_mc=['accuracy', 'precision_weighted', 'precision_macro', '
    precision_micro', 'recall_weighted', 'recall_macro', '
    recall_micro']
```

Code Sample 2. Scoring arrays

Table 11

Parameter description

Parameter	**Description**
criterion	Measure the quality of a split. Values: - gini for the Gini impurity - entropy for the information gain.
max_features	The number of features to consider when looking for the best split. Values: - auto, max_features=sqrt(n_features) - log2, max_features=log2(n_features) - None, max_features=n_features.
max_depth	The maximum depth of the tree.
min_samples_split	The minimum number of samples required to split an internal node
min_samples_leaf	The minimum number of samples required to be at a leaf node
class_weight	The weights for each class
splitter (only for the Decision Tree Classifier)	The strategy used to choose the split at each node Values: - best, to choose the best split - random, to choose the best random split.
n_estimators (for Random Forest and Extremely Randomized Trees Classifiers)	The number of trees in the forest.
bootstrap (for Random Forest and Extremely Randomized Trees Classifiers)	Whether bootstrap samples are used when building trees.

The cross-validation strategy uses *KFold(n_splits=10, shuffle=True, random_state=seed)* class. The *n_splits* determines the number of folds used for cross-validation, in this case it was set to 10. The *shuffle* is set to True so that the data is shuffled before splitting into batches. The *random_state* is a pseudo-random number generator state used for shuffling.

The Code Sample 3 presents the *crossValidation(cls, X, Y, params, scores)* function. The input parameters for this functions are: *cls* is the classifier, X is matrix containing the data set, Y is an array with the classes for each line in X, *params* is the dictionary with parameters for the classifier and *scores* is an array with the score used for evaluating the classifier.

Code Sample 4 presents the function that initializes the classifiers and calls the *crossValidation* function. The function has as input parameter, besides the data set X and the classes Y, the number of classes n. In the case n is 2 then the score s is initialized with the binary classification scores, otherwise s is initialized with the multi-class classification scores.

Code Sample 5 presents the *main* function. This function receives from the command line, as input, two *CSV* files, the first one contains the data set and the second one contains the classes presented in Tables 9 and 10. Before calling the *callClassifiers(X, Y, n)*, two functions are used to create the X matrix and the Y array, one for the binary classification (*corpus_bc(dataset, status)*) and one for the multi-class classification (*corpus_bc(dataset, status)*) that also return the number of classes.

```python
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import StratifiedKFold
import numpy as np

seed = 7
np.random.seed(seed)

def crossValidation(cls, X, Y, params, scores):
    kfold = StratifiedKFold(n_splits=10, shuffle=True,
                  random_state=seed)
    for s in scores:
        clf = GridSearchCV(estimator=cls, param_grid=params,
              cv=kfold, scoring=s, n_jobs=-1)
        grid_result = clf.fit(X, Y)
        means = clf.cv_results_['mean_test_score']
        stds = clf.cv_results_['std_test_score']
        res_clf = clf.cv_results_['params']
        results = zip(means, stds, res_clf)
        for mean, std, params in results:
            print(s, mean, "(+/-)", std * 2, params)
```

Code Sample 3. Grid Cross Validation

```python
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier

def callClassifiers(X, Y, n):
    if n == 2:
        s = scores_bc
    else:
        s = scores_mc
    cls = DecisionTreeClassifier()
    crossValidation(cls, X, Y, params_dtc, s)
    cls = RandomForestClassifier()
    crossValidation(cls, X, Y, params_rec, s)
    cls = ExtraTreesClassifier()
    crossValidation(cls, X, Y, params_rec, s)
```

Code Sample 4. Decision Tree Classifier

```
import sys

if __name__ == "__main__":
    dataset = sys.argv[1]
    classes = sys.argv[2]
    X, Y  = corpus_bc(dataset, status)
    callClassifiers(X, Y, 2)
    X, Y, no_classes = corpus_mc(dataset, status)
    callClassifiers(X, Y, no_classes)
```

Code Sample 5. Main Class

6. Experimental results

In this section the experimental results for the binary and multi-class classification are presented. For the binary classification, the classic measures, Accuracy, Precision and Recall are compared with their weighted versions. For multi-class classification, the average Accuracy and the Macro Precision and Recall are compared with their weighted versions.

Table 12 presents the best scores obtained for weighted Accuracy, Precision and Recall for binary classification.

The best scores of the evaluation methods for the Decision Tree classifier are obtained by the same parameters with the exception of *min_sample_split* which for the weighted precision differs. The Decision Trees and Random Forest Classifiers yield better results when the quality of a split is measured using the Gini impurity, while Extremely Randomized Trees present better scores when the quality of a split is measured using the entropy for information gain.

Table 12

Best measures for binary classification (with weighted features)

Measure	Decision Tree Classifier	Random Forest Classifier	Extremely Randomized Trees Classifier
wA	0.845 ± 0.021	0.874 ± 0.016	0.873 ± 0.019
wP	0.848 ± 0.018	0.874 ± 0.020	0.873 ± 0.017
wR	0.844 ± 0.021	0.874 ± 0.019	0.873 ± 0.018

Table 13 presents the best results obtained for the binary classification without using weighted measures. Accuracy and precision present better scores than their weighted versions, but these scores are wrong because, without taking into account the weights of each class, a lot of miss-classification appears and so anomalies occur, e.g. for Extremely randomized tree the Precision is equal to 1. The Recall, in this case, is lower because it measures the sensitivity of the classifier to correctly identify the right class for a data point.

Table 14 presents the best scores obtained for weighted Accuracy, Precision and Recall and micro Precision and Recall for multi-class classification.

As in the case of the binary classification, the best scores are obtained by the same algorithm by different parameters, but the difference between the scores for the same measure is very small, e.g. the weighted Precision for the Random Forest Classifier with the same parameters used for the weighted Accuracy is 0.712 ± 0.045. The scores obtained for the Extremely Randomized Trees classifiers for weighted Precision and Recall is 0.705 ± 0.040, respectively 0.760 ± 0.023 when the same parameters are used as for the weighted Accuracy.

Table 13

Best measures for binary classification (without weighted features)

Measures	Decision Tree Classifier	Random Forest Classifier	Extremely Randomized Trees Classifier
A	0.848 ± 0.007	0.876 ± 0.010	0.875 ± 0.009
P	0.846 ± 0.018	0.895 ± 0.068	1.000 ± 0.000
R	0.724 ± 0.024	0.760 ± 0.018	0.755 ± 0.020

Table 14

Best measures for multi-class classification (with weighted features)

Measure	Decision Tree Classifier	Random Forest Classifier	Extremely Randomized Trees Classifier
wA	0.732 ± 0.025	0.761 ± 0.026	0.761 ± 0.024
wP	0.692 ± 0.029	0.717 ± 0.036	0.713 ± 0.040
μP	0.733 ± 0.025	0.761 ± 0.025	0.761 ± 0.025
wR	0.733 ± 0.024	0.761 ± 0.027	0.761 ± 0.024
μR	0.763 ± 0.027	0.762 ± 0.028	0.761 ± 0.024

Table 15 presents the best results obtained for the multi-class classification without using weighted measures. In this case, the effects of weights can be seen for the macro Precision and Recall, meaning that both sensitivity and the positive predictive value are not determined correctly.

Table 15

Best measures for multi-class classification (without weighted measures)

Measure	Decision Tree Classifier	Random Forest Classifier	Extremely Randomized Tree Classifier
$avgA$	0.737 ± 0.013	0.767 ± 0.013	0.764 ± 0.011
MP	0.291 ± 0.057	0.417 ± 0.076	0.407 ± 0.056
μP	0.737 ± 0.013	0.767 ± 0.012	0.763 ± 0.013
MR	0.244 ± 0.029	0.261 ± 0.018	0.254 ± 0.023
μP	0.737 ± 0.012	0.767 ± 0.014	0.764 ± 0.012

The results obtained in this paper are similar to the ones in the existing literature, especially research using the same data, thus validating the design choices presented in the proposed solution. The best results for classification using the same dataset can be found in [12], which obtained 86% accuracy, 82% precision and 72% recall using binary classification. The relevance of such a comparison can be found in the fact that the model is similar with the proposed solution, being also based on Random Forest and Extremely Randomized Trees. The results previously presented in Table 12 are similar, obtaining 85% accuracy using the Decision Tree Classifier, while the results using Random Forest and Extremely Randomized Trees are superior, at 87% for both accuracy measurements. In the case of multi-class classification the paper [12] obtained 72% recall, compared to 74% recall for the model described in the proposed solution. The rest of the metrics follow a similar trend, both for binary classification, as well as multi-class classification.

7. Conclusions

Weighted measures prove to be a good choice when dealing with imbalanced data. The scores for the measures show that for all the tested algorithms the weights play an important role in classification.

The Decision Tree Classifier, although the one with the worst results from the tested algorithm, has an overall accuracy of over 84% for the binary classification and of over 73% for the multi-class one. The strategy used to choose the split at each node that yields the best scores is the 'best split'.

Random Forest Classifier yields slightly better results when bootstrap is not used, while Extremely Randomized Trees has slightly better results when bootstrap is used for both binary and multi-class classification. The results for these two classifiers are very similar for both cases.

Other algorithms can be used for classification. Ada Boost and Support Vector Machine will be tested in future work to determine if they produce better results than the Decision Tree-based algorithms.

Acknowledgement

This work has been funded by University Politehnica of Bucharest, through the Excellence Research Grants Program, UPB - GEX. Identifier: UPB - EXCELENŢĂ - 2016 Privacy and anonymity for Data Clouds, Contract number AU11-16-16.

References

[1] *L. Breiman, J.H. Friedman, Olshen and C.G. Stone*, Classification and Regression Trees, Wadsworth International Group, Belmont, California, USA., 1984.

[2] *L. Breiman*, Random Forests, Machine Learning, **45**(2001), No. 1, 5-32.

[3] *Y. Dauxais, D. Gross-Amblard, T. Guyet, T., A. Happe*, Extraction de chroniques discriminantes, Extraction et Gestion des Connaissances (2017), 165-176.

[4] *W. Fan and A. Bifet*, Mining Big Data: Current Status, and Forecast to the Future, ACM SIGKDD Journal, **14**(2013), No. 2, 1-5.

[5] *P. Geurts, D. Ernst and L. Wehenkel*, Extremly randomized trees, Machine Learning, **63**(2006), No. 1, 3-42.

[6] *Q. Gu, L. Zhu and Z. Cai*, Evaluation Measures of the Classification Performance of Imbalanced Data Sets, International Symposium on Computational Intelligence and Intelligent Systems, (2009), 461-471.

[7] *H. He and E.A. Garcia*, Learning from Imbalanced Data, IEEE Transactions on Knowledge and Data Engineering, **21**(2009), No. 9, 1263-1284.

[8] *T.K. Ho*, Random Decision Forests, International Conference on Document Analysis and Recognition, (1995), 278-282.

[9] *T.K. Ho*, The Random Subspace Method for Constructing Decision Forests, IEEE Transactions on Pattern Analysis and Machine Intelligence, **20**(1998), No. 8, 832-844.

[10] *G.V. Kass*, An Exploratory Technique for Investigating Large Quantities of Categorical Data, Journal of the Royal Statistical Society. Series C (Applied Statistics), **29**(1980), No. 2, 119-127.

[11] *T.S. Lim, W.Y. Loh, Y.S. Shih*, A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty-Three Old and New Classification Algorithms, Machine Learning, **40**(2000), No. 3, 203-228.

[12] *V. Levorato, M. Lutz, M. Lagacherie*, Génération automatique de billets journalistiques: singularité et normalité d'une sélection, Extraction et Gestion des Connaissances, **RNTI-E-33**(2017), 45-56

[13] *F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M.Brucher, M. Perrot, E. Duchesnay*, Scikit-learn: Machine Learning in Python Journal of Machine Learning Research, **12**(2011), 2825-2830

[14] *J.R. Quinlan*, Induction of Decision Trees, Journal of Machine Learning, **1**(1986), No. 1, 81-106.

[15] *J.R. Quinlan*, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[16] *C.R. Turner, A.L. Wolf, A. Fuggetta and L. Lavazza*, Feature Engineering, International Workshop on Software Specification and Design, (1998), 162-172.

[17] *M. Sokolova and G. Lapalme*, A systematic analysis of performance measures for classification tasks, Information Processing & Management, **45**(2009), No. 4, 427-437.

[18] *Y. Sun, A.K.C. Wong and M.S. Kamel*, Classification of Imbalanced Data: a Review, International Journal of Pattern Recognition and Artificial Intelligence, **23**(2009), No. 4, 687-719.

Appendix A. Parameters for binary classification

Decision tree Tree Classifier

Parameter	wA	wP	wR
criterion	gini	gini	gini
max_features	None	None	None
max_depth	10	10	10
min_samples_split	2	10	2
min_samples_leaf	1	1	1
splitter	best	best	best
n_estimators	N/A	N/A	N/A
bootstrap	N/A	N/A	N/A

Random Forest Classifier

Parameter	wA	wP	wR
criterion	gini	gini	entropy
max_features	auto	auto	log2
max_depth	None	None	None
min_samples_split	5	10	5
min_samples_leaf	1	1	1
splitter	N/A	N/A	N/A
n_estimators	100	100	1000
bootstrap	False	False	True

Extremely Randomized Trees Classifier

Parameter	wA	wP	wR
criterion	entropy	gini	entropy
max_features	None	log2	None
max_depth	None	None	None
min_samples_split	5	5	5
min_samples_leaf	1	1	1
splitter	N/A	N/A	N/A
n_estimators	1000	1000	1000
bootstrap	True	False	True

Appendix B. Parameters for multi-class classification

Decision tree Tree Classifier

Parameter	wA	wP	μP	wR	μR
criterion	entropy	entropy	entropy	entropy	entropy
max_features	None	None	None	None	None
max_depth	None	None	None	None	None
min_samples_split	5	2	10	2	5
min_samples_leaf	5	1	5	5	5
splitter	best	random	best	best	best
n_estimators	N/A	N/A	N/A	N/A	N/A
bootstrap	N/A	N/A	N/A	N/A	N/A

Random Forest Classifier

Parameter	wA	wP	μP	wR	μR
criterion	gini	entropy	gini	entropy	entropy
max_features	auto	auto	auto	None	auto
max_depth	None	None	None	None	None
min_samples_split	5	2	5	5	5
min_samples_leaf	1	1	1	1	1
splitter	N/A	N/A	N/A	N/A	N/A
n_estimators	1000	1000	1000	1000	100
bootstrap	False	False	False	True	False

Extremely Randomized Trees Classifier

Parameter	wA	wP	μP	wR	μR
criterion	entropy	entropy	entropy	entropy	entropy
max_features	None	None	None	None	None
max_depth	None	None	None	None	None
min_samples_split	5	2	5	5	5
min_samples_leaf	1	1	1	1	1
splitter	N/A	N/A	N/A	N/A	N/A
n_estimators	1000	1000	1000	100	1000
bootstrap	True	True	True	True	True