SELÇUK ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ

S.Ü. Müh. Bilim ve Tekn. Derg., c.6, s.3, ss. 377-393, 2018
Selcuk Univ. J. Eng. Sci. Tech., v.6, n.3, pp. 377-393, 2018
ISSN: 2147-9364 (Electronic)
DOI: 10.15317/Scitech.2018.139

USING CLASSIFICATION ALGORITHMS FOR TURKISH MUSIC MAKAM RECOGNITION

[1]Övünç ÖZTÜRK, [2]Didem ABIDIN, [3]Tuğba ÖZACAR

[1, 2, 3]Manisa Celal Bayar Üniversitesi Bilgisayar Mühendisliği Şehit Prof. Dr. İlhan Varank Kampüsü 45140 - Yunusemre - MANİSA
[1]ovunc.ozturk@cbu.edu.tr, [2]didem.abidin@cbu.edu.tr, [3]tugba.ozacar@cbu.edu.tr

(Geliş/Received: 09.05.2017; Kabul/Accepted in Revised Form: 11.12.2017)

ABSTRACT: Turkish Music pieces are used in various studies including makam recognition in computational music domain. Turkish Music pieces offer a rich content to the researchers because of their different makam properties. SymbTr is one of the most referred Turkish Music data sets in this area. In this study, the pieces from SymbTr data set belonging to 13 makams are used to execute 10 different machine learning algorithms for makam recognition and the performances of these algorithms are evaluated. These algorithms were executed on WEKA application environment and the performances in makam recognition were obtained with F-measure and recall metrics. The machine learning algorithms performed between 82% and 88%.

Key Words: *Machine learning algorithms, Makam recognition, SymbTr, WEKA*

Klasik Türk Müziğinde Makam Tanıma İçin Veri Madenciliği Kullanımı

ÖZ: Türk Müziği eserleri veri kümeleri hesaplamalı müzik alanında başta makam tanıma çalışmaları olmak üzere çeşitli araştırmalarda kullanılmaktadır. Türk Müziği eserleri, farklı makamsal özellikler göstermeleri bakımından araştırmacılara zengin bir içerik sunmaktadır. Bu alanda en çok referans gösterilen Türk Müziği veri setlerinden biri SymbTr veri setidir. Bu çalışmada, SymbTr veri kümesinden 13 makama ait eserler üzerinde 10 farklı makine öğrenmesi algoritması çalıştırılmış ve bu algoritmaların performansları değerlendirilmiştir. Bu algoritmalar WEKA uygulama ortamı üzerinde çalıştırılarak makam tanımadaki başarım yüzdeleri f-ölçütü ve duyarlılık metrikleri üzerinden hesaplanmıştır. Makine öğrenmesi algoritmaları, %82-%88 arası performans göstermiştir.

Anahtar Kelimeler: *Makine öğrenmesi algoritmaları, Makam tanıma, SymbTr, WEKA*

INTRODUCTION

The studies on computational music using Music pieces as data are evolving rapidly. Although most of the studies focus on Western Music, Turkish Music is subject to some studies in literature. Makam system, which is the basis of Turkish Music, attracted attention of computer scientists and became a subject of research in data mining, machine learning and classification.

A study, done in 2012, about pairing sections of a score to the corresponding fragments of 44 audio recording performances in Turkish Music (Şentürk et al., 2012) has recognized final resolution pitches with a success ratio of 89%. The first study about classification of Turkish Music pieces according to their makams in the literature was done by (Gedik and Bozkurt, 2008). The authors then worked on makam recognition by automatic tone detection using frequency histograms (Gedik and Bozkurt, 2010). Another study (Şentürk et

al., 2013) used the final resolution pitch of the piece to recognize the makam. In this study, authors used 57 different recordings of 14 songs, 116 different recordings of 24 *peşrev*s, 84 different recordings of 19 *saz semai*s and they detected the makams with a success ratio of 94.9%.

(Ünal et al., 2012) is one of the most fundamental studies on makam recognition in the literature. In this study, N-gram based statistical analysis (Jurafsky and Martin, 2014) has been applied to 847 pieces from 13 different makams. The success ratio on makam recognition is 86% - 88%. A deficit of using N-gram based statistical analysis is the performance of the model remains constant with changing sizes of data. On the other hand, it is likely that the performances of the machine learning algorithms rise with increasing sizes of data. There are also studies on Dastgah recognition (Darabi et al., 2006) and Dastgah classification (Abdoli, 2011) in Iranian Music. Dastgah is a model which is similar to the makam system of Turkish Music.

In our previous work (Abidin et al., 2016), we used Random Forest algorithm for the classification of 1261 Turkish Music pieces according to their makams. We chose the following makams, which have the highest number of pieces in TRT (Turkish Radio and Television) repertoire (Bozkurt et al., 2014); Hicaz, Nihavend, Hüzzam, Rast, Kürdilihicazkar, Uşşak, Hüseyni, Mahur, Muhayyerkürdi and Hicazkar. In this study, in order to compare our work with similar studies in literature properly, we used the same makam set with these studies. The set is updated as follows: Beyati, Hicazkar, Hüseyni, Hüzzam, Kürdilihicazkar, Mahur, Muhayyer, Nihavend, Rast, Saba, Segah, Uşşak and Hicaz. We also analyzed the performances of 10 different and popular machine learning algorithms on our data set. Finally, we used feature scaling method to increase the performance of the algorithms. Table 1 shows the studies in literature about makam recognition, the final resolution pitch detection and tone detection. The data format used and algorithms applied are also given in the table.

Table 1. Studies in literature

Reference	Aim	Algorithm	Format
(Our Study)	Makam recognition	J48, MLP, Random Forest, Logistic, Bagging, SMO, NaiveBayes, BayesNet, JRip, LMT	MusicXML
(Ünal et al., 2012)	Makam recognition	N-gram model	MusicXML
(Gedik and Bozkurt, 2010)	Makam recognition	Pitch-frequency histogram	Audio files
(Darabi et al., 2006)	Dastgah and makam recognition	Pattern recognition	Audio files
(Abdoli, 2011)	Dastgah classification	Fuzzy logic	Audio files
(Şentürk et al., 2012)	Final resolution pitch detection	Hierarchical linking	Audio files
(Şentürk et al., 2013)	Tone detection	Distribution matching	MusicXML
(Abidin et al., 2016)	Makam recognition	Random Forest	MusicXML

In the next section, we describe the machine learning algorithms we used in the study. Chapter 3 explains our data set and the derived variables that we produced to increase the performance of machine learning. Chapter 4 analyzes the performance of 10 machine learning algorithms. Finally, Section 5 concludes the paper with some potential future work.

MATERIAL AND METHOD

Classification is one of the most common used machine learning algorithms. Classification is a supervised method that classifies the items into several groups. The data set is divided into two groups as

training set and test set, one for building a learning model on it and the other one to test it. In this study, ten classification algorithms are applied on SymbTr data set of Turkish Music Repository (Karaosmanoğlu, 2012). The chosen classification algorithms are applied using WEKA (Waikato Environment for Knowledge Analysis) environment (Frank, 2000). The following subsections describe these algorithms in detail.

J48

J48 (Witten and Frank, 2005) is the Java implementation of C4.5 algorithm (Quinlan, 1993). C4.5 is an extension of Quinlan's earlier ID3 algorithm (Quinlan, 1986) and a classifier which accepts nominal classifiers only. C4.5 generates a decision tree from a set of labeled training data using the information entropy. C4.5 can use both discrete and continuous attributes and training data with missing attribute values. C4.5 classifiers are expressed as decision trees, but they can also construct classifiers in a more comprehensible rule set form (Salama, et. al., 2015). Given a set of cases, C4.5 first grows an initial tree. The initial tree is then pruned to avoid overfitting.

MLP

Artificial Neural Networks (ANN) is a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs (Caudill, 1989). An ANN combines artificial neurons in order to process information. An artificial neuron consists of inputs, which are multiplied by weights, and then computed by a mathematical function which determines the activation of the neuron (Figure 1). Another function computes the logical (Boolean) output of the artificial neuron.

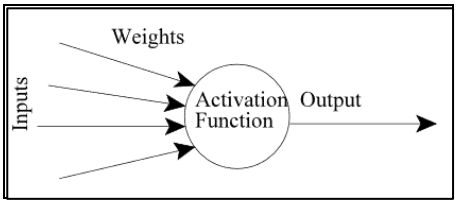

Figure 1. An artificial neuron

Figure 2 shows a typical multilayer NN (MLP) which is used to compute more complex functions than a single-layer NN. MLPs are structured in three layers: An input layer, hidden layer(s) and an output layer. The information passes through the nodes in a forward direction and the final outputs are computed (Karray and Silva, 2004). Then for each of the outputs, error values are computed and propagated to each neuron in the backward direction. In the second phase, the weights are updated to get better results. This forward-backward propagation continues until reaching minimal error values (Rojas, 1996).

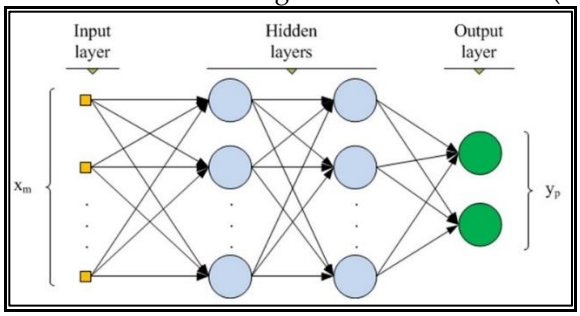

Figure 2. A multilayer neural network (MLP)

Random Forest

Random Forest is a classifier that builds many classification trees as a forest of random decision trees, each constructed using a random subset of the features (Ho, 1995). Each tree gives a classification (vote) and the algorithm chooses the classification having most votes among all the trees in the forest (Fontana et al., 2016). The generalization error for forests converges to a limit as the number of trees in the forest becomes large. The generalization error of a forest of tree classifiers depends on the strength of the individual trees in the forest and the correlation between them (Breiman, 2001).

Logistic

Logistic in WEKA builds and uses a multinomial logistic regression model (Greene, 2012). The linear regression model gets the inputs (x_j) and predicts the output by estimating initial weight values (Θ_j) for each input. This model uses the hypothesis function ($h_\Theta(x)$) (Equation 1) to compute the output.

$$h_\Theta(x) = \Theta_0 + \Theta_1 x_1 + \Theta_2 x_2 + \ldots + \Theta_n x_n \quad (1)$$

Then the cost function, which is the sum of squared difference between the actual output value and the predicted value, is computed. The algorithm tries to minimize the cost function iteratively using the Gradient Descent algorithm (Snyman, 2005). Logistic regression modifies linear regression model by generating hypothesis function ($h_\Theta(x)$) value between 0 and 1.

Bagging

Bagging is an algorithm that is designed to improve the performance of machine learning algorithms. It combines bootstrapping and aggregating (Alfaro, et. al, 2013). With a training set (Tn), B bootstrap samples (Tb) are obtained, where b = 1, 2, .., B. In these boostrap samples, noisy observations may be eliminated or reduced, so the classifiers built in these sets will have a better behavior than the classifier built in the original set. Therefore, bagging can be really useful to build a better classifier when there are noisy observations in the training set. It also helps to reduce overfitting in the algorithm. Given a standard training set, the *m* versions of this set are formed by making bootstrap replicates of the learning set (Breiman, 1996). The *m* models are fitted using the above *m* bootstrap samples and combined by averaging the output for regression or voting for classification.

Sequential Minimal Optimization (SMO)

In machine learning, support vector machines (SVMs, also support vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. SVMs divide the data into two classes. Therefore, they assign the data to the values 0 and 1 (Cortes and Vapnik, 1995). From this perspective they look like the Logistic Regression Model but in SVM problems, data cannot be classified linearly. The instances of two classes can be separated with a non-linear boundary. In this case, the hypothesis function ($h\Theta(x)$) will be calculated from complex polynomial features. An example hypothesis function is shown in Equation 2.

$$Predict \begin{cases} y = 1 \; if \; \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 + \theta_5 x_2^2 + \ldots \geq 0 \\ y = 0 \qquad otherwise \end{cases} \quad (2)$$

SMO is used to solve the Support Vector Machine (SVM) training problem. In our case, SMO is an application of multi-class SVM. In practice, multi-class classification problems are decomposed into a series of binary problems that the standard SVM can be applied (Wang and Xue, 2014). SMO uses heuristics to partition the training problem into smaller subproblems. Typically, it speeds up the training process (Platt, 1998). These small QP problems are solved analytically, which avoids using a time-consuming numerical QP optimization as an inner loop. The amount of memory required for SMO is linear in the training set size, which allows SMO to handle very large training sets. Because matrix computation is avoided, SMO scales somewhere between linear and quadratic in the training set size for various test problems, while the standard chunking SVM algorithm scales somewhere between linear and cubic in the training set size. SMO's computation time is dominated by SVM evaluation; hence SMO is fastest for linear SVMs and sparse data sets. On real-world sparse data sets, SMO can be more than 1000 times faster than the chunking algorithm.

Naïve Bayes

Naïve Bayes (NB) is a set of supervised learning algorithms based on Bayes theorem (Bayes and Price, 1763). This algorithm is one of the most important classification algorithms because it is very easy to construct, not needing any complicated iterative parameter estimation schemes, easy to interpret, robust, fast and space efficient (Wu et al., 2008). Naïve Bayes uses Equation 3 and Equation 4 to classify instances with single and multiple attributes (Bishop, 2006; Duda et al., 2000).

$$p((c_j|d) = \frac{p(d|c_j)p(c_j)}{p(d)} \quad (3)$$

$$p(d|c_j) = p(d_1|c_j)\, p(d_2|c_j) \ \ldots\ p(d_n|c_j) \quad (4)$$

In order to classify instances with only one attribute, the algorithm uses Bayes theorem given in Equation 3, where $p(c_j \mid d)$ is the probability of instance d being in class c_j, $p(d \mid c_j)$ is the probability of generating instance d given class c_j, $p(c_j)$ is the probability of occurrence of class c_j and $p(d)$ is the probability of instance d occurring. This equation is used in datasets with one attribute and in the case of having multiple attributes (Equation 4), naive Bayesian classifiers assume that these attributes have independent distributions.

BayesNet

BayesNets improves the performance of naive Bayesian classifiers by avoiding unwarranted assumptions about independence (Friedman et al., 1997). In order to tackle this problem effectively, we need an appropriate language and efficient machinery to represent and manipulate independence assertions. Both are provided by Bayesian networks (Pearl, 1988). These networks express relationships among variables by directed acyclic graphs with probability tables stored at the nodes. Each vertex in the graph represents a random variable, and edges represent direct correlations between the variables.

Figure 3 shows a typical Bayesian network, which is taken from (Russell and Norving, 2010). In this figure, the letters B, E, A, J and M stand for *Burglary, Earthquake, Alarm, JohnCalls* and *MarryCalls* respectively. This example depicts an alarm which is fairly reliable at detecting a burglary, but also responds on occasion to minor earthquakes. We have also two neighbors in the example; John and Mary, who calls us when they hear the alarm. John always calls when he hears the alarm, but sometimes confuses the telephone ringing with the alarm and calls then, too. Mary, on the other hand, sometimes misses the alarm. Given the evidence of who has or has not called, we would like to estimate the probability of a burglary.

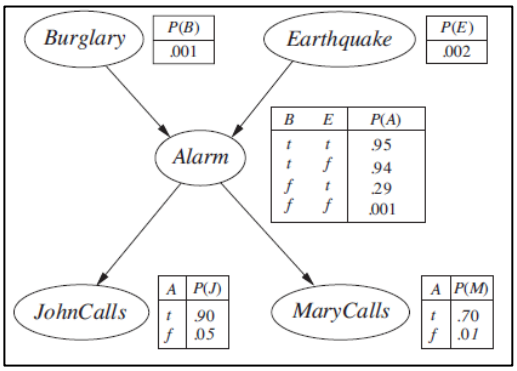

Figure 3. A Bayesian Network showing the topology and conditional probability tables

JRip

Repeated Incremental Pruning to Produce Error Reduction (RIPPER) (Cohen, 1995), is a propositional rule learner. The main idea of RIPPER approach is for seeking an initial set of rules and iteratively improving it by applying an optimization algorithm (Mehdiyev et al., 2015). To build a rule, algorithm uses the following strategy: (a) First the training data is split into a growing set and a pruning set. (b) Next a rule is grown using the growing set. In this phase, a rule is grown by greedily adding antecedents to the rule. The procedure tries every possible value of each attribute and selects the condition with highest information gain. (c) After growing a rule, the rule is immediately pruned. The instances from the pruning data set are applied to advance the performance of the obtained set by pruning it.

Logistic Model Tree (LMT)

LMT is a supervised training algorithm that combines logistic regression and decision trees. A logistic model tree basically consists of a standard decision tree structure with logistic regression functions at the leaves, much like a model tree is a regression tree with regression functions at the leaves, and the tree is pruned using a CART algorithm (Breiman et al., 1984). As in ordinary decision trees, a test on one of the attributes is associated with every inner node (Landwehr et al., 2005).

The LogitBoost algorithm uses additive logistic regression of least-squares fits for each class M_i (Doetsch et al., 2009) as in Equation 5, where β_i is the coefficient of the i^{th} component of the vector x and n is the number of factors.

$$L_M(x) = \sum_{i=1}^{n} \beta_i x_i + \beta_0 \tag{5}$$

The linear logistic regression method is used to compute the posterior probabilities of leaf nodes in the LMT model (Landwehr et al., 2005; Tien Bui et al., 2016) as in Equation 6, where D is the number of classes.

$$P(M|x) = \frac{\exp(L_M(x))}{\sum_{M=1}^{D} \exp(L_M(x))} \tag{6}$$

TURKISH MAKAM MUSIC DATA SET

In this study, we use most popular machine learning algorithms (Wu et al., 2008; Meenakshi and Geetika, 2014) for makam recognition on Turkish Music pieces from the SymbTr data set. The data is prepared according to previous studies, in which N-gram algorithm was used by the authors to recognize makams (Ünal et al., 2012). In that study, authors chose 13 different makams to apply the selected algorithm. In our study, we also chose the same 13 makams and 1246 pieces from SymbTr data set in order to compare the performances of algorithms properly. The chosen makams and the number of pieces of each makam are as follows: Beyati (62), Hicazkar (80), Hüseyni (97), Hüzzam (97), Kürdilihicazkar (70), Mahur (89), Muhayyer (67), Nihavend (128), Rast (112), Saba (70), Segah (97), Uşşak (119) and Hicaz (158).

The chosen songs in SymbTr can be found in XML format in the original data set of SymbTr. With our Java application parsing the XML files, symbolic note representations of the songs are obtained. This symbolic representation only contains the chords in letters (A, B, C, D, E, F, G) and flat (bemol) or sharp (diyez) signs of chords. The Java application also performs note transformations and adds derived variables automatically. The derived variables are determined as follows: scale frequencies, mainly emphasized note and final resolution pitch. These two processes of the application and the derived variables are described in the following subsections.

Note Transformation

In Turkish Music, the Arel – Ezgi - Uzdilek (AEU) makam system (Can, 2002) is used. This system is built on 25 tones on 24 non-equally-tempered intervals (Özkan, 1994). Unlike Western Music, the interval between two tones is divided into 9 equal pieces. Each of these 9 pieces is called "koma" and the makam characteristics are determined according to the *koma*s of notes. Since existing note writing software are designed according to Western Music standards, note representation of Turkish Music pieces must be transformed properly before the process. The corresponding letters for accidental tones of the notes are given in Table 2. In this table, seven notes (A (la), B (si), C (do), D (re), E (mi), F (Fa) and G (sol)) and 28 pitches are represented with the corresponding letter or letter groups. In the first step of our Java application, the automatic transformation of the note sequences of 1246 pieces is done according to Table 2.

Table 2. Legend for tones in Turkish Music

A		B		C		D		E		F		G	
A1b	H	**B1b**	J	**C1#**	L	**D1b**	U	**E1b**	O	**F4#**	Fd	**G4b**	Gb
A4b	Ab	**B4b**	Bb	**C4#**	Cd	**D4b**	Db	**E4b**	Eb	**F5#**	Q	**G5b**	R
A5b	I	**B5b**	K	**C5#**	M	**D5b**	T	**E5b**	P			**G4#**	Gd
A4#	Ad	**B4#**	Bd			**D4#**	Dd	**E4#**	Ed			**G5#**	S
						D5#	N						

Adding Derived Variables

To perform makam recognition in WEKA on the transformed note sequences and increase machine learning performance, some derived variables are calculated in the second step of the Java application. These derived variables increase the performance of machine learning by about 40%. Each derived variable is explained in the following subsections.

Scale frequencies

In Turkish Makam Music, two scales are used for distinguishing makams: one tetra-chord (Scale1) and one penta-chord (Scale2). For each makam, these tetra-chords and penta-chords differ. For example, the scale of Hicaz makam include one Hicaz tetra chord *la - sib – do# - re* and one Rast penta-chord *re - mi – fa# - sol - la*. Since we have 13 makams in this study, the .arff file includes 13 different frequency values. The final frequency value of each makam *m* is the summation of the frequency of tetra-chord of *m* and the frequency of penta-chord of *m* in the piece. Makam attributes used in the calculation process are given in Table 3.

Table 3. Attribute Values of Makams.

Makam	**Scale1**	**Scale2**	**Makam**	**Scale1**	**Scale2**
Hicaz	AKCdD	DEFdGA	Kürdilihicazkar	GFEbDC	CBbAbG
Kürdi	ABbCD	DEFGA	Hüseyni	AJCDE	EFdGA
Hüzzam	JCDPFd	FdGAdJ	Acemaşiran	FGABbC	CDEF
Nihavend	GABbCD	DEbFG	Mahur	GQED	DCBAG
Rast	GAJCD	DEFdG	Muhayyerkürdi	AGFED	DCBbA
Uşşak	AJCD	DEFGA	Segah	JCDOFd	FdGAdJ
Hicazkar	GFdPD	DCJAbG			

Mainly emphasized note

For each makam, the end of the first scale and the beginning of the second scale is the same note. This note is called the mainly emphasized (*güçlü)* note of the makam. We derived a variable for this note as follows; for each mainly emphasized note, the frequency of that note in all note sequences are calculated and added to the .arff file. Mainly emphasized notes for thirteen makams are given in Table 4.

Table 4. Mainly emphasized notes for makams

	Hicaz	Kürdi	Hüzzam	Nihavend	Rast	Uşşak	Hicazkar	Kürdilihicazkar	Hüseyni	Acemaşiran	Mahur	Muhayyerkürdi	Segah
Note	D	D	Fd	D	D	D	D	D	E	C	D	E	Fd

Final Resolution Pitch

Every piece in Turkish Makam Music ends with a final resolution pitch, which identifies the makam of that piece. The final resolution pitch for each makam is given in Table 5. Java application writes this note to the .arff file using the note sequence automatically.

Table 5. Final resolution pitch notes for makams.

	Hicaz	Kürdi	Hüzzam	Nihavend	Rast	Uşşak	Hicazkar	Kürdilihicazkar	Hüseyni	Acemaşiran	Mahur	Muhayyerkürdi	Segah
Note	A	A	J	G	G	A	G	G	A	F	G	A	J

Feature Scaling and the .arff File

To increase the performance, the feature scaling method is applied on the values of the derived variables as follows: (1) The maximum of the first thirteen values about the frequencies of the scales are found and each of these thirteen values is divided by this maximum value. (2) The maximum of the next five values about the mainly emphasized notes are found and each of these five values is divided by this maximum value.

Figure 4 shows the final status of the .arff file. The first thirteen fields of each entry (piece) show the frequencies of the scales that are used for distinguishing makams. The next five fields represent the five mainly emphasized notes that belong to these thirteen makams (Some makams have common mainly emphasized notes.). And the next field shows the final resolution pitch of the piece.

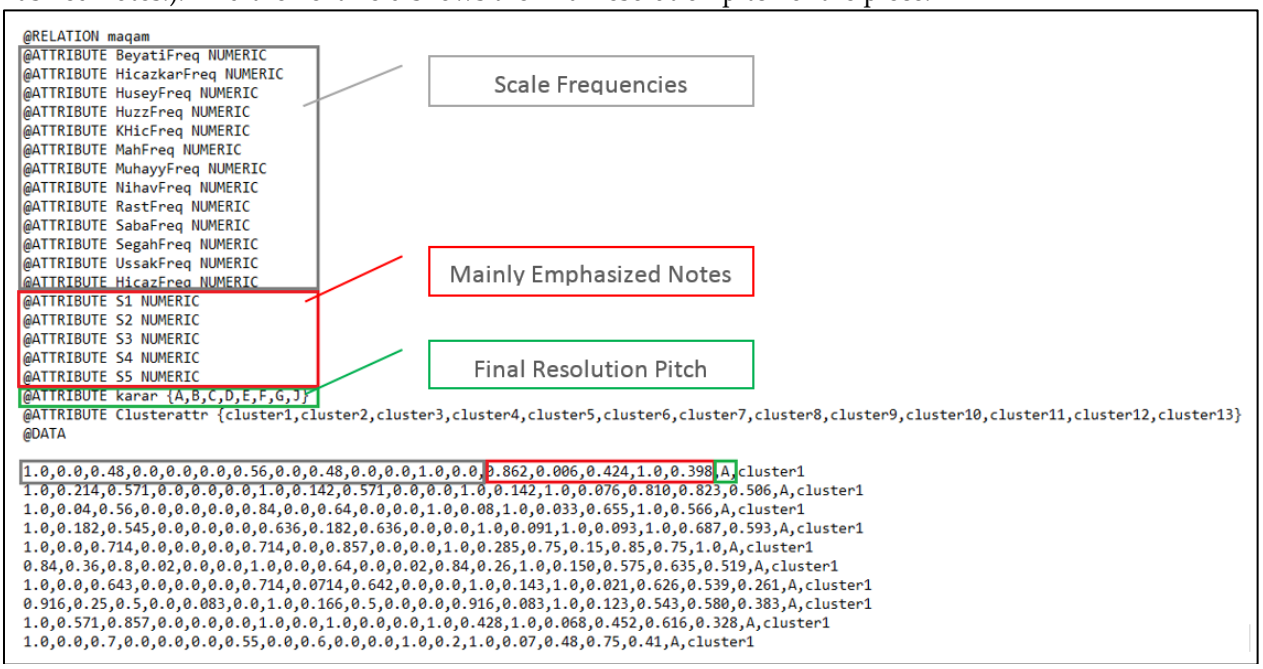

Figure 4. Definition of attributes in .arff file

EXPERIMENTS AND RESULTS

We analyzed the performance of the machine learning algorithms using WEKA. The classification operation is performed by choosing the proper algorithm under the "Classify" tab. As the test option, "Cross Validation" with 10 folds (k=10) is used. With this option, the data sets to be obtained can have different sizes and it does not need to divide the data into two sets as training and test sets. Instead, data set is divided into 10 groups. Then the algorithm uses k-1 folds for learning and the remaining one fold for the

test. For performance comparisons, recall and F-measure values are used (Powers, 2011). For further analysis, the confusion matrix of each algorithm can be downloaded from https://figshare.com/articles/Appendix_docx/5484217.

Recall value is the fraction of relevant instances that are successfully classified. It is calculated as in Equation 7.

$$recall = \frac{|\{relevant\ pieces\} \cap \{retrieved\ pieces\}|}{|\{retrieved\ pieces\}|} \quad (7)$$

F-measure is an evaluation technique, which is used to measure a test's accuracy and calculated as in Equation 8.

$$F = 2 * \frac{precision * recall}{precision + recall} \quad (8)$$

In the subsection 4.1, we define the parameters of algorithms we used in the experiments. Subsection 4.2 presents our experiment results using F-measure and recall metrics. This section also compares our results with the nearest study in the literature (Ünal et al., 2012, which also uses the same dataset with us. Finally, subsection 4.3 summarizes the performances of resembling works in the literature.

Algorithm Parameters

Table 6 shows the parameters that we use with each machine learning algorithm. In J48 algorithm, post pruning is implemented on a fully induced decision tree and sifts through to remove statistically insignificant nodes. In Weka, the parameter altered to test the effectiveness of post pruning is the "confidenceFactor". In WEKA J48 classifier, lowering the confidence factor decreases the amount of post pruning. In our study, the confidence is held constant at 0.25 to minimize post pruning and cross validation folds for the testing set is held at 10.

Table 6. The parameters for the machine learning algorithms

Algorithm	Parameters	Algorithm	Parameters
J48	confidence factor = 0.25 cross validation folds = 10	SMO	c = 1.0 E = 1.0 (linear kernel)
MLP	#inputs =19 #outputs =13 #hidden layers =1	Naive Bayes	useSupervisedDiscretization = true useKernelEstimator = false
Random Forest	#trees = 100 #features = 0 (log M + 1)	BayesNet	estimator = Simple Estimator searchAlgorithm = Simulated Annealing useADTree = false
Logistic	maxIts = -1	JRIP	folds= 10 minNo = 2 optimizations = 2
Bagging	classifier = REPTree bagSizePercentage = 100% numIterations = 10	LMT	minNumInstances = 15 splitOnResiduals = false weightTrimBeta = 0.01

For MLP algorithm, the number of hidden layers is selected as 1. Since default settings will often give reasonable results, we use WEKA MLP on our data set with default settings. Default setting on neurons in the hidden layer is "a". This special character ("a") defines the number of nodes in the hidden layers. "a" calculates the number of nodes in the hidden layers as "a"= (classes+attributes)/2. Therefore, in our case the nodes in the first layer is (13 + 19)/2= 16.

For the Random Forest algorithm, we pick a large number of trees (100) and according to the original paper of (Breiman) and we use 1 + $\log_2 M$ features. So, we set the number of features to consider as 0. If this parameter is a value smaller than 1, WEKA will use logM+1, where M is the number of inputs.

For Logistic algorithm, "maxIts" is set to -1, which means the number of cross validation iterations is not fixed but is continued till the feature-class weights are fully optimized. A key configuration parameter is the choice of "classifier". We choose the default classifier, namely REPTree, which is the WEKA implementation of a standard decision tree (CART). The "bagSizePercent" parameter specifies the size of each random sample. We choose 100%, which will create a new random sample with the same size as the training dataset. "numIterations" specifies the number of bags. We increase the value of this parameter until we no longer see an improvement. Then the optimum value is 10, which is also the default parameter value.

For SMO algorithm, we consider two parameters; (1) the value of c: controls the tradeoff between fitting the training data and maximizing the separating margin. We check the following values of c; 0.1, 1.0 and 10.0. The optimum value is 1.0. (2) The kernel parameters: for the polynomial kernel, the most important parameter is "exponent", which controls the degree of the polynomial. We try 1 for the linear kernel, 2 for the quadratic kernel and 3 for the cubic kernel. We get the maximum performance with the linear kernel.

NaiveBayes algorithm has two important parameters; "useKernelEstimator" and "useSupervisedDiscritization". We change the "useSupervisedDiscritization" parameter value as "True", so we automatically convert our numerical attributes to nominal attributes. The performance is changed from 78.01% to 82.99%. Then we change the "useKernelEstimator" argument that may better match the actual distribution of the attributes in our dataset. But this change has no effect on the performance.

We have three important parameters for BayesNet algorithm; "estimator", "searchAlgorithm" and "useADTree". The "searchAlgorithm" option is used to select a structure learning algorithm. If we set "useADTree" option to "True", then counts are calculated using the ADTree algorithm (Moore, 1998). Changing these two parameters does not make an improvement for our dataset. Therefore, we use default values of parameters. Finally, the estimator option is used to select the method for estimating the conditional probability distributions. We get the maximum performance when we set this parameter to "SimulatedAnnealing"

The "folds" parameter in JRIP stands for the number of folds for reduced error pruning. We get the optimum result when this parameter is set to 10. The second important parameter is "minNo", which defines the minimal weights of instances within a split. We get the maximum performance with the default value 2. The third important parameter is the "optimizations", which sets the number of runs of optimizations. The default value 2 gives the maximum performance for our study.

For LMT algorithm, the "minNumInstances" parameter specifies the minimum number of instances at which a node is considered for splitting. For our dataset, changing the value of this parameter does not affect the performance. Therefore, we use the default value of this parameter. "weightTrimBeta" is the threshold for trimming weights. We get the best results, when we set this parameter to 0.01. "splitOnResiduals" sets splitting criterion based on the residuals of LogitBoost. There are two possible splitting criteria for LMT; the default is to use the C4.5 splitting criterion that uses information gain on the class variable. The other splitting criterion tries to improve the purity in the residuals produces when fitting the logistic regression functions. The choice of the splitting criterion does not affect our results.

Results of Experiments

F-measure values for each machine learning algorithm and N-gram algorithm (Ünal et al., 2012) are given in Figure 5. In the original work (Ünal et al., 2012), authors used recall values for performance comparisons. Therefore, F-measure values were calculated for comparison in our study. Figure 6 shows recall values for each algorithm in this study and N-gram of (Ünal et al., 2012). Our results are close to the results of N-gram. But the performance of the N-gram based statistical analysis remains constant with changing sizes of data. On the other hand, it is likely that the performance of the machine learning algorithms rises with increasing sizes of data. In Table 7, the percentages of correctly classified instances for each makam are given. This table also gives the percentage of correctly classified instances of N-Gram in the grey column.

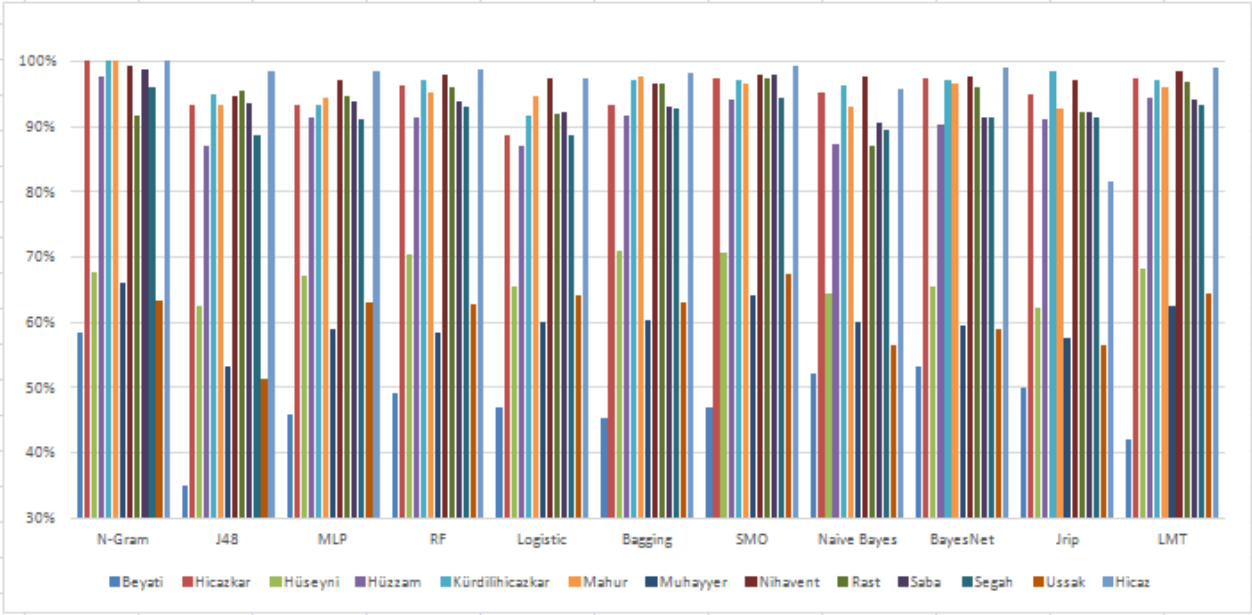

Figure 5. F-measure values for 10 machine learning algorithms and N-gram algorithm

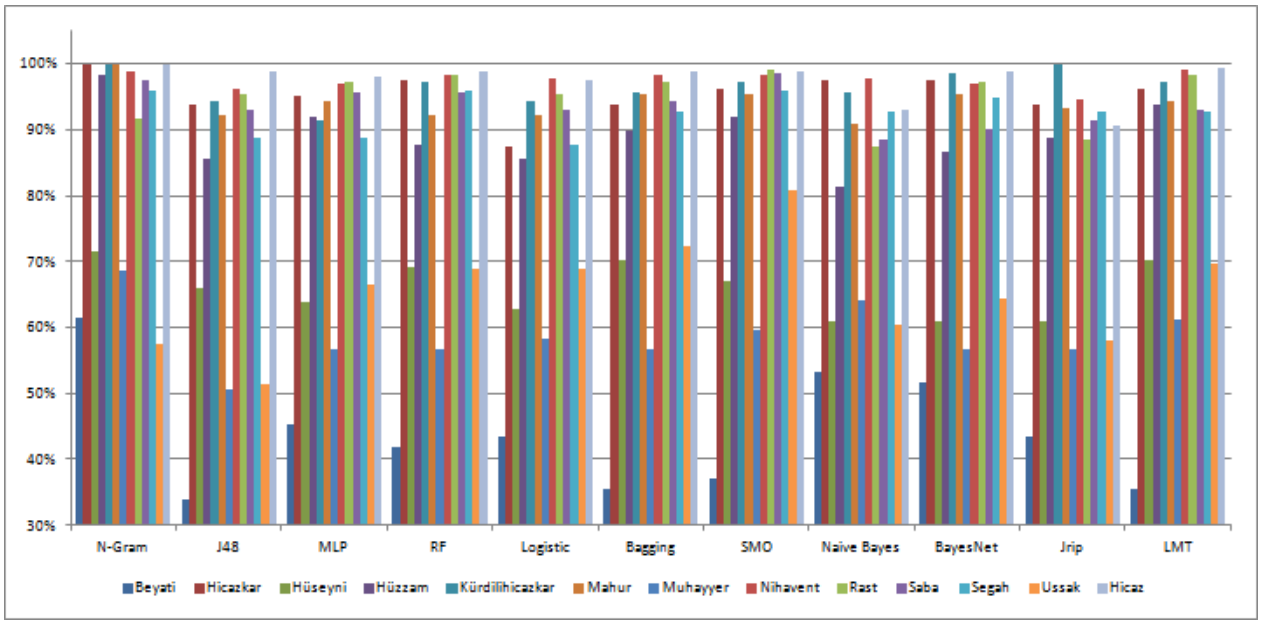

Figure 6. Recall values for 10 machine learning algorithms and N-gram algorithm

Table 7. Percentages of correctly classified instances for each machine learning algorithm

	N-Gram	J48	MLP	Random Forest	Logistic	Bagging	SMO	Naïve Bayes	BayesNet	JRip	LMT
%	87.9574	82.1027	85.1525	86.5169	83.9486	86.2761	88.122	82.9856	85.634	82.183	86.9181

According to the results in the confusion matrices (download link: https://figshare.com/articles/Appendix_docx/5484217) and F-measure graph, all machine learning algorithms are remarkably successful with a performance of 99.35% in distinguishing Hicaz makam when compared to other makams. Rast (97.37%), Nihavend (98.40%), Kürdilihicazkar (98.60%) and Saba (97.89%) makams are also recognized successfully in most of the algorithms. For these four makams, SMO algorithm gives the best performances. The biggest problem is recognizing Beyati, Hüseyni, Muhayyer and Uşşak makams. These four makams have the same mainly emphasized notes (A) and similar makam scales. Most of the machine learning algorithms classified more than half of the Beyati pieces as Uşşak, since these two makams are the most similar ones. The worst classification performance is obtained for Beyati makam (46.64%). When F-measure values are compared for four makams, Bagging algorithm is the most successful one with a performance of 70.84%. The misclassified makams are the ones that are also misclassified by domain experts. Therefore, these results show that machine learning algorithms can be applied to makam recognition effectively. When the overall results are examined, J48 algorithm has the worst performance in classifying makams with 82.10%. SMO algorithm has the highest classification percentage with 88.12%. SMO also does better in distinguishing some of the makams which are misclassified frequently including Hüseyni (%70.64) and Uşşak (%67.36).

Performances of Other Studies

There are some important studies on the classification of Turkish music makams in the literature. Table 8 lists these studies and their performances (%). However, they differ in terms of the data set they use. For this reason, the comparison of performance results with our study is not appropriate.

Table 8. Percentages of correctly classified instances for each machine learning algorithm

Study	Technique	Dataset	Performance
(Kalender et al., 2012)	Combined Neural Networks	Selçuk University Dilek Sabancı State Conservatory Archives (50 pieces, 9 makams)	95.83
(Gedik and Bozkurt, 2008)	Pitch interval histogram	Monophonic instrumental recordings (9 makams)	92.00
(Kızrak et al., 2014)	Probabilistic Neural Networks	Recordings from commercial CDs and Internet (62 pieces, 6 makams)	89.40
(Kızrak and Bolat., 2014)	Neural Networks	Recordings from commercial CDs and Internet (62 pieces, 6 makams)	89.60
(Kızrak and Bolat, 2015)	Deep Belief Networks	Recordings from commercial CDs and Internet (62 pieces, 6 makams)	92.70
(Gedik and Bozkurt, 2010)	Pitch frequency histogram	Various recordings (172 pieces, 9 makams)	77.38
(Kalaycı and Korukoğlu, 2012)	K-means+ANN	Bir Şarkıdır Yaşamak - Turkish Art Music Selection Series (103 pieces, 7 makams)	70.00

DISCUSSION AND FUTURE WORK

In this study, we used ten most popular machine learning algorithms for makam recognition in Turkish Music. We used thirteen makams, which were used in some of the studies in the literature and 1246 pieces from SymbTr data set that belong to these makams. Then we compared the performances of the algorithms with similar studies in the literature.

In the first step of the study, a letter-based transformation was applied to note sequences in order to represent the sequences in a shorter and more understandable format. In the second step, derived variables were added to the data set in order to increase the performances of machine learning algorithms. Feature scaling was also applied to numerical variables for performance gain. In the evaluation phase, the machine learning algorithms had performance values between 82% (J48) and 88% (SMO). These performance values are similar to the study (Ünal et al., 2012). But the N-gram algorithm used in (Ünal et al., 2012), shows a fixed performance, independent from the size of the data set. The machine learning algorithms used in this study are expected to have better performances with the increasing size of the data set. When the f-measure graph is examined, the misclassified makams are the ones that are also misclassified by domain experts. This result shows that machine learning algorithms can be applied to makam recognition effectively.

As a future work, we plan to derive and use more makam-specific features in order to improve the skills of machine learning algorithms on makam recognition.

REFERENCES

Abdoli, S., 2011, "Iranian Traditional Music Dastgah Classification", *12. International Society for Music Information Retrieval Conference (ISMIR 2011)*, Florida, USA, 275-280, 2011.

Abidin, D., Öztürk, Ö., Özacar Öztürk, T., 2017, "Klasik Türk Müziğinde Makam Tanıma Için Veri Madenciliği Kullanımı", *Gazi Üniversitesi Mühendislik-Mimarlık Fakültesi Dergi*si, Vol. 32(4) , pp. 1221-1232.

Alfaro, E., Gamez, M., Garcia, N., 2013, "adabag: An R Package for Classification with Boosting and Bagging", *Journal of Statistical Software*, Vol. 54 (2), pp. 1 - 35.

Bayes, M., Price, M., 1763, "An Essay Towards Solving a Problem in The Doctrine of Chances. By the Late Rev. Mr. Bayes, F. R. S. Communicated by Mr. Price, in a Letter to John Canton, A. M. F. R. S.", *Philosophical Transactions*, Vol. 53, pp. 370–418.

Bishop C.M., 2006, *Pattern Recognition and Machine Learning*, Springer-Verlag.

Bozkurt, B., Ayangil, R., Holzapfel, A., 2014, "Computational Analysis of Turkish Makam Music: Review of State-of-the-Art and Challenges", *Journal of New Music Research,* Vol. 43(1), pp. 3-23.

Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J., 1984, *Classification and Regression Trees*, Belmont Wadsworth International Group, CA.

Breiman, L., 1996, "Bagging Predictors" *Machine Learning*, Vol. 24(2), pp. 123–140.

Breiman, L., 2001, "Random Forests". *Machine Learning*, 5–32. doi: 10.1023/A:1010933404324.

Can, M.C., 2002, "Geleneksel Türk Sanat Müziğinde Arel - Ezgi - Uzdilek Ses Sistemi ve Uygulamada Kullanılmayan Bazı Perdeler", *G.Ü. Gazi Eğitim Fakültesi Dergisi*, Vol. 22(1), pp. 175-181.

Caudill, M., 1989, "Neural Network Primer: Part I", *AI Expert.*

Cohen, W.W., 1995, "Fast Effective Rule Induction", *Proceedings of the 12th International Conference on Machine Learning*, 115–123.

Cortes, C., Vapnik, V., 1995, "Support Vector Networks", *Machine Learning*, Vol. 20, pp. 273-297.

Darabi, N., Azimi, N., Nojumi, H., 2006, "Recognition of Dastgah and Makam for Persian Music with Detecting Skeletal Melodic Models", *2. IEEE BENELUX/DSP Valley Signal Processing Symposium*,

Doetsch, P., Buck, C., Golik, P., Hoppe, N., Kramp, M., Laudenberg, J., Oberdörfer, C., Steingrube, P., Forster, J., Mauser, A., "Logistic Model Trees with AUC Split Criterion for the KDD Cup 2009 Small Challenge", *ACM Knowledge Discovery and Data Mining (KDD)*, France, 77-88, 28 June *2009*.

Duda, R.O., Hart, P.E., Stork, D., 2000, *Pattern Classification*, Wiley and Sons.

Fontana, F.A., Mäntylä, M.V., Zanoni, M., Marino, A., 2016, "Comparing and Experimenting Machine Learning Techniques for Code Smell Detection", *Empirical Software Engineering,* Vol. 21, pp. 1143-1191.

Frank, E., Witten, I.H., 2000, *Data Mining*, Morgan Kaufmann Publishers.

Friedman, N., Geiger, D., Goldszmidt, M., 1997, "Bayesian Network Classifiers", *Machine Learning*, Vol. 29, pp. 131–163.

Gedik, A.C., Bozkurt, B., 2008, "Automatic Classification of Turkish traditional Art Music Recordings by Arel Theory", *Conference on Interdisciplinary Musicology*, Thessaloniki, Greece.

Gedik, A.C., Bozkurt, B., 2010, "Pitch-Frequency Histogram-Based Music Information Retrieval for Turkish Music", *Signal Processing, Vol.* 90, pp. 1049-1063, Elsevier.

Greene, W.H., 2012, *Econometric Analysis* (Seventh ed.). Boston: Pearson Education, 803–806.

Ho, T.K., 1995, "Random Decision Forests", *Proceedings of the 3rd International Conference on Document Analysis and Recognition*, Montreal, QC, 278-282.

Jurafsky, D., Martin, J.H., 2014, N-Grams, *Speech and Language Processing*, Pearson Education, pp. 93-136.

Kalaycı, I., Korukoğlu, S., 2012, "Classificatıon of Turkish Maqam Music using K-Means Algorithm and Artificial Neural Networks (in Turkish)", *Proc. 20th IEEE Signal Processing and Communications Applications Conference (SIU)*, Muğla, Türkiye.

Kalender, N., Ceylan, M., Karakaya, O., 2012, "Türk Müziği Makamlarının Sınıflandırılması için Yeni Bir Yaklaşım: Kombine YSA", *ASYU 2012 Akıllı Sistemler Yenilikler ve Uygulamaları Symposium*, Trabzon, Türkiye.

Karaosmanoğlu, M.K., 2012, "A Turkish Makam Music Symbolic Database for Music Information Retrieval: SymbTr", *13. International Society for Music Information Retrieval Conference (ISMIR 2012),* Porto, Portugal, 223-228.

Karray, F.O., Silva, C.D., 2004, *Soft Computing and Intelligent Systems Design: Theory, Tools and Applications*, Addison Wesley Pearson Press, New York, USA.

Kizrak, M.A., Bayram, K.S., Bolat, B., 2014, "Classification of Classic Turkish Music Makams", *Innovations in Intelligent Systems and Applications (INISTA 2014),* Alberobello, Italy, 394-397, 23-25 June 2014.

Kizrak, M.A., Bolat, B., 2014, "Klasik Türk Müziği Makamlarının Tanınması", *Akıllı Sistemlerde Yenilikler ve Uygulamaları Sempozyumu (ASYU 2014)*, Katip Çelebi Üniversitesi, İzmir, Türkiye.

Kizrak, M.A., Bolat, B., 2015,"Classification of Turkish Music Makams bu Using Deep Belief Networks", *IEEE 23. Sinyal İşleme ve İletişim Uygulamaları Kurultayı*, Malatya, Türkiye, 2015.

Landwehr, N., Hall, M. & Frank, E., 2005, "Logistic Model Trees" *Machine Learning*, Vol. 59, pp. 161.

Meenakshi, Geetika, 2014, "Survey on Classification Methods using WEKA", *International Journal of Computer Applications,* Vol. 86(18), pp. 16-19.

Mehdiyev, N., Krumeich, J., Enke, D., Werth, D., Loos, P., 2015, "Determination of Rule Patterns in Complex Event Processing Using Machine Learning Techniques", *Procedia Computer Science*, Vol. 61, pp. 395-401.

Moore, A., Lee, M.S., 1998, "Cached Sufficient Statistics for Efficient Machine Learning with Large Datasets", *Journal of Artificial Intelligence Research* (*JAIR*), Vol. 8, pp. 67-91.

Özkan, İ.H., 1994, *Türk Musikisi Nazariyatı ve Usulleri Kudüm Velveleleri*, Ötüken Neşriyat, İstanbul, 57-58.

Pearl, J., 1988, *Probabilistic Reasoning in Intelligent Systems*, San Francisco, CA, Morgan Kaufmann.

Platt, J.C., 1998, "Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines", Technical Report MSR-TR-98-14, Microsoft Research.

Powers, D.M.W., 2011, "Evaluation: From Precision, Recall and F-measure to ROC, Informedness, Markedness & Correlation", *Journal of Machine Learning Technologies*, Vol. 2, pp. 37-63.

Quinlan, J.R., 1986, "Induction of Decision Trees", Machine Learning 1, 81-106.

Quinlan, J.R., 1993, *C4.5 Programs for Machine Learning*, Morgan Kaufmann Publishers.

Rojas,R., 1996, *Neural Networks: A Systematic Introduction*, Springer-Verlag, Berlin.

Russell, S., Norvig, P, 2010, *Artificial Intelligence, A Modern Approach*, 3rd Edition, Prentice Hall, 511-512.

Salama, A.A., Eisa, M., ELhafeez, S.A., Lotfy, M.M., 2015, "Review of Recommender Systems Algorithms Utilized in Social Networks based e-Learning Systems & Neutrosophic System", *Neutrosophic Sets and Systems*, Vol. 8, pp. 33.

Snyman, J., 2005, *Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms*, 97, XX-258.

Şentürk, S., Gulati, S., Serra, X., 2013, "Score Informed Tonic Identification for Makam Music of Turkey", *14. International Society for Music Information Retrieval Conference (ISMIR 2013),* Curitiba, Brasil.

Şentürk, S., Holzapfel, A., Serra, X., 2012, "An Approach for Linking Score and Audio Recordings in Makam Music of Turkey", *2. CompMusic Workshop*, İstanbul, Turkey, 95-106.

Tien Bui, D., Tuan, T.A., Klempe, H., Pradhan, B., Revhaug, I., 2016, "Spatial Prediction Models for Shallow Landslide Hazards: a Comparative Assessment of the Efficacy of Support Vector Machines,

Artificial Neural Networks, Kernel Logistic Regression and Logistic Model Tree", *Landslides*, Vol. 13 (2), pp. 361–378.

Ünal, E., Bozkurt, B., Karaosmanoğlu, M.K., 2012, "N-Gram Based Statistical Makam Detection on Makam Music in Turkey Using Symbolic Data", *13. International Society for Music Information Retrieval Conference (ISMIR 2012)*, Porto, Portugal.

Wang, Z., Xue, X., 2014, *Multi-class Support Vector Machine*, Chapter 2, Springer International Publishing, 23-24.

Witten, I.H., Frank, E., 2005, *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann Press, San Francisco, USA.

Wu, X., et al., 2008, "Top 10 Algorithms in Data Mining", *Knowledge and Information Systems,* Vol. 14, pp. 1-37.