Yazılım Hata Kestiriminde Kolektif Sınıflandırma Modellerinin Etkisi

Deniz Kılınç[1], Emin Borandağ[1] , Fatih Yücalar[1],
Akın Özçift[1], Fatma Bozyiğit[1]

[1] Celal Bayar Üniversitesi Hasan Ferdi Turgutlu Teknoloji Fakültesi
Yazılım Mühendisliği Bölümü, Manisa, Türkiye
deniz.kilinc@cbu.edu.tr, emin.borandag@cbu.edu.tr,
fatih.yucalar@cbu.edu.tr, akin.ozcift@cbu.edu.tr,
fatma.bozyigit@cbu.edu.tr

Özet. Yazılım hatalarının tespiti karmaşık ve maliyetli bir süreçtir. Yazılım projelerinde ortaya çıkan hataların önceden tespit edilip düzeltilmesi, öngörülen maliyeti ve proje süresini aşma risklerini azaltır. Hataların tespiti için "yazılım ölçütleri" etkili kullanılarak, erken yazılım geliştirme aşamalarında kod analiz edilip, hata yatkınlığıyla ilgili fikir sahibi olunabilir. Yazılımların ölçümünde kullanılmak üzere çeşitli yazılım ölçütleri üretilmiştir. Bu çalışmada, "Chidamber and Kemerer (CK)" ve "Object Oriented (OO)" yazılım ölçütlerine sahip 4 veri seti üzerinde, 4 tane temel sınıflandırıcı ile bunların AdaBoost ve Rotation Forest kolektif sınıflandırma modelleri kullanılarak, yazılım hata kestirimi yapılmıştır. Temel sınıflandırıcılar seçilirken farklı kategorilerden olmasına dikkat edilmiştir. Genel olarak değerlendirildiğinde, kolektif sınıflandırma yöntemlerini kullanmak, temel sınıflandırıcıların başarımına göre yaklaşık %70 oranla daha iyi sonuç vermektedir.

Anahtar Kelimeler. Yazılım hata kestirimi, makine öğrenmesi, sınıflandırma, kolektif sınıflandırma, veri madenciliği.

1 Giriş

Yazılım testi temel olarak geliştirilen ürünün beklenen kalitede olduğunu belirlemek, değilse istenilen kaliteye ulaştırılmasını sağlamak için kullanılan bir süreçtir. Günümüz yazılım dünyasında, yazılım test yaklaşımları ve yazılım kalitesi en çok çalışılan konular haline gelmiştir. Bunun başlıca nedeni, yazılımların büyümesiyle birlikte karmaşıklıklarının ve beraberinde yazılımlarda ortaya çıkan hatalar ile bu hataları düzeltme maliyetlerinin artmasıdır [1][2].

Yazılım projelerinde ortaya çıkan hataların önceden tespit edilip düzeltilmesi öngörülen maliyeti ve proje süresini aşma risklerini azaltır. Ortaya çıkması muhtemel hataları mümkün olduğu kadar erken tespit edebilmek için, verimli ve etkili bir test planının uygulanması gerekir. Yazılım ölçütlerinin etkili kullanılmasıyla daha erken aşamalarda kod analiz edilip, hata yatkınlığıyla ilgili fikir sahibi olunabilir, gerekirse önlem alınabilir.

Yazılım hata ölçütleri, bir önceki yazılımın hatalarını kullanarak bir sonraki yazılımın hatalarını tahmin etmek için kullanılır. Yazılım hata tahmin yöntemleri sayesinde yazılımdaki hataların, kaynak kodun bazı özellikleri incelenerek tespit edilmesi mümkündür. Günümüzde yazılım hata tahmini için çeşitli sınıflandırma algoritmaları geliştirilmiştir [3][4].

Sınıflandırma, nesneleri önceden tanımlanmış bilgilere ve bulunduğu duruma göre ilgili kategoriye atama anlamına gelir. Başka bir ifade ile sınıflandırma, geçmişte toplanan verilerin hangi sınıfa ait olduğu bilindiğinde, yeni gelen verinin hangi sınıfa ait olduğunu bulma işlemidir [5]. Sınıfı bilinen nesneler ile (öğrenme veri seti) bir model kurulur. Kurulan model öğrenme kümesinde yer almayan nesnele ile (test veri seti) test edilerek performansı ölçülür. Literatürde karar ağaçları, bayes sınıflandırıcıları, kural-tabanlı sınıflandırıcılar, yapay sinir ağları, k-en yakın komşu sınıflandırıcıları, destek vektör makinası ve kolektif öğrenme yöntemleri gibi sınıflandırma algoritmaları kullanılmaktadır.

Bu çalışmada, "Chidamber and Kemerer (CK)" ve "Object Oriented (OO)" yazılım ölçütlerine sahip dört veri seti üzerinde, dört temel sınıflandırıcı ile bunların AdaBoost ve Rotation Forest kolektif sınıflandırma modelleri kullanılarak yazılım hata kestirimi yapılmıştır.

Bildirinin ikinci bölümünde temel sınıflandırma algoritmaları ile ilgili bilgilere yer verilmiştir. Üçüncü bölümde, kolektif sınıflandırma içerisinde en çok kullanılan algoritmalar ele alınmıştır. Dördüncü bölümde ise deneysel çalışma, kullanılan veri seti, deneysel ölçütler ve değerlendirme sonuçları yer almaktadır. Son olarak, beşinci bölümde ise elde edilen sonuçlar değerlendirilmiştir.

2 Sınıflandırma Algoritmaları

Bütün sınıflandırma algoritmalarında hedeflenen temel nokta, minimum veri ile maksimum sayıda verinin sınıflanmasıdır. Sınıflandırma algoritmalarının kullanılmasındaki temel amaç, minimum zamanda ve minimum eğitim verisi ile tahmin yapabilmektir [6]. Çok farklı sınıflandırma türleri olsa da genel olarak yedi farklı sınıflandırma türü vardır. Bunlar; Karar Ağaçları (Decision Trees), Kural-Tabanlı (Rule-Base), En Yakın Komşu (Nearest-Neighbor), Bayes, Yapay Sinir Ağları (Artificial Neural Network), Destek Vektör Makinesi (Support Vector Machine), Kolektif Sınıflandırma (Ensemble Classifier) olarak adlandırılmaktadır.

Karar ağaçları (decision trees), veri setlerinin sınıflandırılması için kullanılan temel bir yöntemdir. Oluşturulması ve sonuçlarının yorumlanması kolay olduğu için pek çok farklı veri setinde kullanılmıştır. Dört temel adımdan oluşmaktadır. Eğitim veri setinin oluşturulması ile kurallar bütünü oluşturulur. Seçilen özellikler ile kök düğüm, iç düğüm ve yapraklar belirlenir. Seçilen her bir nitelik için beklenmeyen durumun ve belirsizliğin ortaya çıkma olasılığı kullanılarak bilgi kazancı hesaplanır. En yüksek bilgi kazancı oranı kök olarak belirlenir [7]. Karar ağaçlarındaki en önemli sorun oluşturulacak karar ağacının kök, düğüm, iç düğüm ve yapraklarının belirlemesindeki zorluktur. Bu sorunun aşılması için çeşitli algoritmalar geliştirilmiştir. Bunların başında belirsizlik olasılığının ölçümü için kullanılan entropi algoritmaları

gelmektedir. Diğerleri ise CART denilen regresyon ve sınıflandırma ağaçları içerisinde yer alan Twoing ve Gini algoritmaları ile bellek tabanlı sınıflama algoritmalarıdır [8].

Kural-tabanlı sınıflandırmada (rule-based classifier), bilgi veya bilgi bitlerini temsil etmek için kurallar kullanılmaktadır. Bir kural-tabanlı sınıflandırıcı, sınıflandırma için IF-THEN kuralları kümesini kullanır. Bir IF-THEN kuralında, IF koşulu THEN sonucu ifade etmektedir [9].

Sınıfların belirlenmesinde kullanılan diğer bir temel bir sınıflandırma türü ise En Yakın Komşu (Nearest Neighbor) yöntemidir. Hangi sınıfa ait olduğu bilinmeyen veriyi tanımlamak için kendisine en yakın olan sınıf üyesi öklit yöntemi kullanılarak belirlenir. Bu işin gerçekleştirilebilmesi için bütün veriler n boyutlu bir uzay içerisinde olacak şekilde yerleştirilir ve hangi sınıfa ait olduğu belli olmayan veri, kendisine en yakın sınıfın üyesi olarak belirlenir [10].

Bayes Sınıflandırma (Bayes Classifier), istatistiksel bir yaklaşımla hangi verinin hangi sınıfın üyesi olduğunu bulmaya çalışan bir sınıflandırma türüdür. Kendi içerisinde kural-tabanlı bir sistemi vardır. Temel kural göre verinin hangi sınıftan olması gerektiğinin olasılığı hesaplanır [11].

Yapay Sinir Ağları (Artificial Neural Networks), verilerin karmaşık yapıya sahip olduğu ve gürültü içerdiği durumlarda ve veriler arasındaki doğrusal olmayan ilişkilerin öğrenilmesinde başarılı ve güçlü bir sınıflandırma yöntemidir [12]. Yapay Sinir Ağlarının temelinde, insan beynine ait üstün karakteristikleri taklit ederek, yazılımları tanıma-öğrenme süreçlerinin performanslarını arttırmak vardır.

Destek Vektör Makinesi (Support Vector Machine), istatistiksel öğrenme teorisi alanında ortaya atılan, Cortes ve Vapnik tarafından geliştirilmiş bir öğrenme metodudur [13]. Destek Vektör Makinesi temel olarak, lineer olmayan örnek uzayının, örneklerin lineer olarak ayrılabileceği bir yüksek boyuta aktararak farklı örnekler arasındaki maksimum sınırın bulunması esasına dayanır.

Kolektif Sınıflandırmada (Ensemble Classifier) temel amaç, daha önceden farklı sınıflandırıcılar tarafından elde edilen değerlerin bir araya getirilmesi ile bir sonuç üretilmesidir. Bu işlem yapılırken diğer sınıflandırıcılara belli ağırlık puanları verilerek hesaplama yapılmaya çalışılır. Burada asıl problem farklı sınıflama algoritmalarını birleştirilmek ve hangi oranların kullanılacağına karar vermektir. En büyük avantajı diğer yöntemlerin verilerini bir arada kullandığı için daha iyi değerler elde edilebilmesidir [14]. Kolektif Sınıflandırma içerisinde yerine koyarak örnekleme (bagging), hızlandırma (boosting), rotasyon ormanı (rotation forest) ve rastgele orman (random forest) gibi çeşitli algoritmalar bulunmaktadır.

3 Kollektif Sınıflandırma Algoritmaları

Kolektif sınıflandırma içerisinde çeşitli algoritmalar bulunmaktadır. Bunlardan en çok bilinen ve kullanılanları yerine koyarak örnekleme (bagging), hızlandırma (boosting), rastgele orman (random forest) ve rotasyon ormanı (rotation forest) dır.

3.1 Yerine Koyarak Örnekleme

Breiman tarafından önerilen yerine koyarak örnekleme (bagging) algoritması, var olan bir eğitim setinden yeni eğitim setleri türeterek temel öğreniciyi yeniden eğitmeyi amaçlayan bir yöntemdir [15]. Bagging'de n adet örnekten oluşan eğitim setinden yine n örnekli bir eğitim seti yerine koymalı rastgele seçimle üretilir. Bu durumda bazı eğitim örnekleri yeni eğitim kümesinde yer almazken bazıları birden fazla kez yer alırlar. Topluluktaki her bir temel öğrenici bu şekilde üretilmiş birbirinden farklı örnekler içeren eğitim kümeleriyle eğitilirler ve sonuçları çoğunluk oylaması ile birleştirilir.

3.2 Hızlandırma (Boosting)

Sınıflandırma açısından diğer önemli olan bir konuda hızlandırma (boosting) yöntemidir. Bu yöntem sayesinde sınıflandırıcının bulmuş olduğu doğruluk değeri arttırılabilir. Boosting yönteminde veriye ait bir önceki sınıflandırıcının doğru olarak belirleyemediği veriler kullanılır. Hatalı veriler sonradan kullanılacak eğitim seti içerisine tekrardan eklenerek daha doğru tahmin yapılmaya çalışılır. Bu yöntemde her bir veri işlemi için bir ayarlama oranının hesaplanması vardır [16, 17]. Boosting için en fazla kullanılan Adaboost algoritmasıdır.

3.3 Rastgele Orman (Random Forest)

Rastgele Orman (Random Forest - RF) bir kolektif sınıflandırma algoritmasıdır. Breiman ve Cutler [18, 19] tarafından geliştirilmiştir. Sürekli olarak yeni versiyonlarla güncellenmektedir. Temel olarak sınıflandırma ve regresyon için kullanılmaktadır. Naif karar ağacına benzer bir sisteme sahiptir. Temel farkı eğitim sürecinde birçok ağacın üretilmesine izin vermesidir. RF metodunda, ormanı oluşturan karar ağaçları Bootstrap yöntemi kullanılarak seçilen farklı örneklerden bir araya gelir. Bu yöntem için orijinal veri seti kullanılır. Veri setinin eğitimi kısmında çoklu karar ağaç yapısı kullanılır.

3.4 Rotasyon Ormanı (Rotation Forest)

Rotasyon orman algoritması son yıllarda sınıflandırıcıların performansının arttırılması amacıyla önerilen yeni nesil bir kolektif öğrenme algoritmasıdır [20]. Birden fazla ağaç kullanılmakta olan bu algoritmasının çalışma prensibi rastgele orman algoritmasına benzerdir. Rastgele orman algoritmasında olduğu gibi Bootstrap algoritması bu algoritmada temel öğretici olarak kullanılmaktadır [21]. Rastgele orman algoritmasından farklı olarak ormandaki her bir karar ağacının eğitiminde kullanılacak veri seti, ana bileşen analizi yardımıyla belirlenir. Rotasyon orman algoritması ile ormandaki karar ağaçlarının eğitimi aşamasında eğitim veri seti rastgele alt kümelere bölünür ve her bir alt kümeye ana bileşenler analizi uygulanarak özellik çıkarımı gerçekleştirilir.

4 Deneysel Çalışma

Bu bölüm içerisinde gerçekleştirilen deneysel çalışmalardan bahsedilmiştir.

4.1 Değerlendirme Kriterleri

Sınıflandırma modellerinin değerlendirilmesi için "Hata Matrisi (Confusion Matrix)" kullanılmaktadır. Tablo 1'de Hata Matrisi görülmektedir.

Tablo 1. Hata Matrisi

			Tahmin Edilen Sınıf	
			C_1 (+)	C_2 (-)
Gerçek Sınıf	C_1(+)	∑ Pozitif	TP	FN
	C_2(-)	∑ Negatif	FP	TN

Tablo 1'de görülen; TP: True Pozitif, FP: False Pozitif, FN: False Negatif, TN: True Negatif değerlerini ifade etmektedir.

Başarım oranı (Accuracy – ACC), sınıflandırıcının sınıf ayırım yeteneğini belirlemek için geniş çapta kullanılan bir ölçüttür. Algoritma tarafından doğru olarak sınıflandırılan test örneklerinin yüzdesi olarak tanımlanır. ACC, sınıflandırıcı performanslarının değerlendirilmesinde kullanılan temel ölçütlerden biridir [22]. Denklem 1'de verilen formül ile ACC oranı hesaplanır.

$$ACC = \frac{(TP+TN)}{Pozitif+Negatif} \tag{1}$$

Alıcı İşletim Karakteristiği (Receiver Operating Characteric - ROC) ya da sade biçimde ROC eğrisi; testin ayırt etme gücünün belirlenmesine, çeşitli testlerin etkinliklerinin kıyaslanmasına ve uygun pozitiflik eşiğinin belirlenmesine olanak sağlar [23]. Bir sınıflandırıcının yeterliliğini belirlemek için kullanılabilen pratik bir yöntem, performansın tek bir değer ile ifadesidir. En yaygın kullanılan ölçüm ise, ROC eğrisinin altında kalan alandır (Area Under Curve - AUC). AUC oranı ne kadar büyük ise sınıflandırıcının başarım oranı da o kadar iyidir. AUC'nin olası değerleri 0.5'ten 1.0'a kadar değişim gösterir. Denklem 2'de verilen formül ile AUC oranı hesaplanır.

$$AUC = \frac{1}{2}\left(\frac{(TP)}{TP+FN} + \frac{(TN)}{TN+FP}\right) \tag{2}$$

4.2 Veri Setleri

Bu çalışmada Ambros ve arkadaşları [24] tarafından hata kestirimi amaçlı hazırlanmış veri seti kullanılmıştır. Bu veri seti içerisinden toplam 3689 sınıfa sahip açık kaynak kodlu 4 proje seçilmiştir. Sınıfların "Chidamber and Kemerer (CK)", "Object

Oriented (OO)" ve entropi gibi farklı ölçütlerine ek olarak sınıflardaki hata sayıları da veri seti içerisinde yer almaktadır. Veri seti hakkındaki bilgiler Tablo 2'de verilmiştir.

Tablo 2. Veri setine ilişkin bilgiler

Yazılım	Sınıf Sayısı	Versiyon	Sonraki Sürüm Hataları
Eclipse JDT Core www.eclipse.org/jdt/core/	997	91	463
Eclipse PDE UI www.eclipse.org/pde/pde-ui/	1562	97	401
Equinox framework www.eclipse.org/equinox/	439	91	279
Apache Lucene lucene.apache.org	691	99	103

4.3 Kaynak kod ölçütleri

Yazılımları doğru ölçümlemek, yazılımlardaki kaliteyi arttıracaktır. Yazılımların ölçümünde kullanılmak üzere çeşitli yazılım ölçütleri üretilmiştir. "Chidamber and Kemerer (CK)" ve "Object Oriented (OO)" ölçütler bunlardan bazılarıdır. Sınıflandırma algoritmalarının dikkate aldığı her bir veri setine ait CK ve OO ölçütleri Tablo 3'de verilmiştir.

Tablo 3. CK ve OO ölçütleri

No	Tip	Ölçüt	Açıklama
1	CK	DIT	Kalıtım ağacı derinliği
2	CK	WMC	Ağırlıklı metot sayısı
3	CK	CBO	Nesneler arasındaki bağımlılık
4	CK	NOC	Sınıfları sayısı
5	CK	RFC	Sınıf yanıt sayısı
6	CK	LCOM	Metotların uyumsuzluğu
7	OO	Fan-in	Sınıfı referans eden diğer sınıfların sayısı
8	OO	Fan-out	Sınıf tarafından referans edilen diğer sınıfların sayısı
9	OO	NOA	Niteliklerin sayısı
10	OO	NOPA	Genel niteliklerin sayısı
11	OO	NOPRA	Özel niteliklerin sayısı
12	OO	NOAI	Kalıtım niteliklerinin sayısı
13	OO	LOC	Kod satır sayısı
14	OO	NOM	Metotların sayısı
15	OO	NOPM	Genel metotların sayısı
16	OO	NOPRM	Özel metotların sayısı
17	OO	NOMI	Kalıtım metotlarının sayısı

4.4 Değerlendirme Sonuçları

Tablo 4'de görüldüğü üzere literatürden dört temel sınıflandırıcı seçilmiştir. Sınıflandırma yapılırken her bir sınıflandırıcı için tekli ve kolektif sınıflandırıcılar kullanılmıştır. 4 ayrı veri seti üzerinde, seçilen bu dört temel sınıflandırıcı ile bunların AdaBoost ve Rotation Forest kolektif sınıflandırma modelleri kullanılarak, yazılım hata kestirimi yapılmıştır. Yapılan hata kestirimi sonucunda her bir proje için ayrı ayrı elde edilen ACC ve AUC oranları Tablo 4'te sunulmuştur.

Tablo 4. Yapılan sınıflandırma sonucu elde edilen ACC ve AUC sonuçları

Sınıf	**Algoritma**	**Eclipse JDT Core**		**Eclipse PDE UI**		**Equinox Framework**		**Apache Lucene**	
		ACC	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**
Bayes	Bayesian Logistic Reg.	0,72	0,57	0,86	0,50	0,66	0,70	0,90	0,50
	Bayesian Logistic Reg.-AdaBoost	0,72	0,55	0,85	**0,73**	0,66	0,69	0,90	0,50
	Bayesian Logistic Reg.-Rotation Forest	**0,83**	**0,67**	0,86	0,56	**0,74**	**0,72**	**0,91**	**0,53**
Rules	Decision Table	0,84	0,75	0,85	0,61	0,69	0,77	0,90	0,62
	Decision Table-AdaBoost	0,84	0,75	0,85	0,70	**0,70**	0,77	0,90	**0,65**
	Decision Table-Rotation Forest	0,84	0,75	**0,86**	**0,74**	0,69	**0,78**	0,90	**0,70**
Lazy	IBk	0,80	0,69	0,79	0,60	0,69	0,75	0,86	0,56
	IBk - AdaBoost	0,80	0,69	0,79	0,58	0,69	0,75	0,86	0,54
	IBk – Rotation Forest	**0,81**	**0,77**	**0,83**	**0,70**	**0,70**	**0,79**	0,86	**0,64**
Tree	J48	0,82	0,70	0,85	0,65	0,67	0,75	0,90	0,53
	J48 - AdaBoost	0,81	**0,77**	0,84	0,69	**0,70**	0,76	0,88	**0,63**
	J48 - Rotation Forest	**0,84**	**0,77**	**0,86**	**0,72**	0,69	**0,76**	0,90	**0,67**

Dört farklı veri seti içinde ACC ve AUC oranlarına bakılmıştır. Toplam olarak 4 ayrı projenin 4 farklı kategorine baktığımızda 16 veriden 11 tanesinin kolektif sınıflandırıcıların ACC metriğinde daha iyi olduğu görülmüştür. Öte yandan AUC metriği temel alındığında kolektif öğrenme algoritmalarının diğer dört temel algoritma ile kıyaslamalarında 16 veriden 15 tanesinde daha iyi olduğu görülmüştür. Kolektif sınıflandırıcıların kendi içinde ACC oranlarına göre karşılaştırıldığında 16 veriden 11 tanesine Rotation Forest algoritmasın Adaboost algoritmasına göre daha iyi olduğu görülmüştür.

5 Sonuç

Çalışmada dört temel sınıflandırıcı ile bunların AdaBoost ve Rotation Forest kolektif sınıflandırma modelleri kullanılmıştır. 4 veri seti üzerinde toplam 12 adet sınıflandırma modeli denenmiştir. Temel sınıflandırıcılar seçilirken farklı kategorilerden olmasına dikkat edilmiştir. Çalışmada "bayes" kategorisinden "Bayesian Logistic Regression" , "rules" kategorisinden "Decision Table", "lazy" kategorisinden "IBk" ve "tree" kategorisinden "J48" sınıflandırıcıları seçilmiştir. Dört farklı veri setinden özellikle "Bayesian Logistic Regression yönteminde Ensemble Classifier yöntemleri daha başarılı olduğu görülmüştür. Genel olarak bakıldığında ACC oranlarına göre yaklaşık %70 oranla kolektif sınıflandırma yöntemlerini kullanmak daha iyi bir sonuç verdiği gözlemlenmiştir.

Kaynaklar

1. Song, Q., Sheppard, M., Cartwright, M., and Mair, C.: Software Defect Association Mining and Defect Correction Effort Prediction. In: IEEE Transactions on Software Engineering, Vol.32, No.2, pp. 69-82 (2006).
2. Fenton, N., and Ohlsson, N.: Quantitative Analysis of Faults and Failures in a Complex Software System. In IEEE Transactions on Software Engineering, Vol.26, No.8, pp. 797-814 (2000).
3. Catal, C., Sevim, U., and Diri, B.: Software Fault Prediction of Unlabeled Program Modules. Proceedings of the World Congress on Engineering, Vol.1, London, UK (2009).
4. Catal, C., Diri, B.: Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem. Elsevier: Information Sciences, Vol.179, No.8, pp. 1040–1058 (2009).
5. Akman, M., Genç, Y., Ankaralı, H.: Random Forest Yöntemi ve Sağlık Alanında Bir Uygulama. Türkiye Klinikleri J Biostat, 3(1), ss.36-48 (2011).
6. Akman, M.: Veri Madenciliğine Genel Bakış ve Random Forests Yönteminin İncelenmesi: Sağlık Alanında Bir Uygulama. Yüksek Lisans Tezi, Ankara Üniversitesi, Ankara (2010).
7. Cha, S.H., Tappert, C.C.: A Genetic Algorithm for Constructing Compact Binary Decision Trees. Computer Science Department, Pace University 861 Bedford Road, Pleasantville, Journal of Pattern Recognition Research, New York, Vol.4, No.1, pp.1-13 (2009).
8. Özkan, Y.: Veri Madenciliği Yöntemleri. Papatya Yayıncılık Eğitim, 2. Basım (2013)
9. Han, J., Kamber, M., Pei, J.: Data Mining: Concepts and Techniques. Third Edition, Morgan Kaufmann (2011).
10. Kırmızıgül Çalışkan, S., Soğukpınar, İ.: K-means ve K en yakın komşu yöntemleri ile ağlarda nüfuz tespiti. 2. Ağ ve Bilgi Güvenliği Ulusal Sempozyumu, Girne (2008)
11. Nigam, K., McCallum, A., Thrun, S., Mitchell, T.: Learning to Classify Text from Labeled and Unlabeled Documents. In: Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence, pp. 792-799 (1998).
12. Cortés, E., Martinez, M.G., Rubio, N.G.: Multiclass Corporate Failure Prediction by Adaboost.M1. International Advances in Economic Research, 13, Issue 3, pp. 301-312 (2007).
13. Cortes, C. and Vapnik, V.: Support-vector network. Machine Learning. 20, 273–297., (1995).

14. Augusty, S. M., Izudheen, S.: Ensemble Classifiers A Survey: Evaluation of Ensemble Classifiers and Data Level Methods to Deal with Imbalanced Data Problem in Protein-Protein Interactions. Review of Bioinformatics and Biometrics, Volume 2 Issue 1 (March 2013).
15. Breiman, L.: Bagging predictors. Machine Learning, 24(2) (1996).
16. Schapire, R. E.: Theoretical Views of Boosting and Applications. In: Proceedings of the 10th International Conference on Algorithmic Learning Theory (1999).
17. Schapire, R. E.: A Brief Introduction to Boosting. In: Proceedings of the 16th International Joint Conference on Artificial Intelligence (1999).
18. Breiman, L.: Random Forests. Machine Learning. 45 (1). pp.5–32 (2001).
19. Liaw, A.: Documentation for R package Random Forest. (2012).
20. Çölkesen, İ., Yomralıoğlu, T., Kavzoğlu, T.: Rotasyon Orman Algoritması ile Yüksek Çözünürlüklü Multispektral Uydu Görüntülerinin Sınıflandırılması. V. Uzaktan Algılama ve Coğrafi Bilgi Sistemleri Sempozyumu (UZALCBS 2014). İstanbul (2014).
21. Cingiz, M. Ö., Albayrak, A., Amasyalı, M. F.: Sınıflandırıcı Topluluklarının Gürültülü Verilere Karşı Gürbüzlüğünün Değerlendirilmesi. Signal Processing and Communications Applications Conference (SIU). (2013).
22. Ozcift, A., Gulten, A.: Classifier ensemble construction with rotation forest to improve medical diagnosis performance of machine learning algorithms. Computer Methods and Programs in Biomedicine. Vol. 104. Issue 3. pp. 443–451 (2011).
23. Faraggi, D., Reiser, B.: Estimation of the area under the ROC curve. Stat Med. 21. pp. 3093-3106 (2002).
24. D'Ambros, M., Lanza, M., Robbes, R.: An extensive comparison of bug prediction approaches. In MSR '10: Proceedings of the 7th International Working Conference on Mining Software Repositories. pp. 31-41 (2010).