Kosinüs Benzerliğine Dayalı Çapraz-proje Hata Tahmini

Araştırma Makalesi/Research Article

Muhammed Maruf ÖZTÜRK

Bilgisayar Mühendisliği Bölümü, Süleyman Demirel Üniversitesi, Isparta, Türkiye
muhammedozturk@sdu.edu.tr

(Geliş/Received:14.08.2018; Kabul/Accepted:22.05.2019)
DOI: 10.17671/gazibtd.453436

Özet— Çapraz-proje tahmini özellikle metrik heterojenliği açısından araştırmacıların ilgisini çekmekte, bu alanda yeni yöntemlere ihtiyaç duyulmaktadır. Hata tahmin işleminin farklı projeler üzerinden yürütülmesi geliştiricilere anlamlı bilgiler sunmaktadır. Bu çalışmada, çapraz-proje tahmini için, Kosinüs benzerliğine dayalı metrik eşleştirmesi yapan CSCDP isimli bir algoritma geliştirilmiştir. Yöntem 36 farklı veri setinde üç farklı sınıflandırıcı ile test edilmiştir. Elde edilen sonuçlara göre ortalama tahmin performansının yapay sinir ağlarında diğer sınıflandırıcılara göre daha yüksek olduğu tespit edilmiştir. Ayrıca, seyreklik analizine dayalı olarak seçilen eğitim veri setlerinin test başarısını olumlu etkilediği tespit edilmiştir. Son olarak, CSCDP kullanılarak yürütülen çapraz-proje tahmininin sınıflandırma hatasını Random Forest algoritmasında F-skor parametresi için 0.65 oranında azalttığı gözlemlenmiştir.

Anahtar Kelimeler— hata tahmini, kosinüs benzerliği, çapraz-proje

Cosine Similarity-based Cross-project Defect Prediction

Abstract— Cross-project defect prediction has been intriguing researchers in terms of metric heterogeneity and new methods are needed in this field. Performing defect prediction through different projects presents valuable information for developers. In this work, a metric matching algorithm namely CSCDP is presented for cross-project defect prediction. The method is then tested on 36 different projects via three classifiers. According to the obtained results, neural network predictor outperforms the others in terms of mean prediction values. Further, selecting training data sets using sparsity analysis creates a favorable effect on testing performance. Last, CSCDP was able to reduce classification error up to 0.65 in Random Forest for F-score.

Keywords— defect prediction, consine-similarity, cross-project

1. GİRİŞ (INTRODUCTION)

Yazılım hata tahmini, yoğun çaba gerektiren bir süreçtir [1]. Yazılım projelerin ölçeklerinin giderek büyümesi, hata tahmin sürecinin zorlaşmasına ve projelerin daha karmaşık hale gelmesine neden olmaktadır. Dolayısıyla özellikle veri seti düzeyinde tahmin performansını attırmaya yönelik yöntemler geliştirilmektedir.

Yazılım hata veri setlerinde hatalılık etiketi genelde true/false veya 0/1 ile temsil edilmektedir. Ancak bazı projelere ait veri setlerinde hata adetleri hata etiketine bölümüne kaydedilebilmektedir. Böyle durumlarda hata etiketinde yer alan değerler 0'dan farklı olma durumda 1' e çevrilerek ikili sınıflandırma yapılabilmektedir. Fakat yazılım hata veri setlerindeki uğraşılar genellikle yazılım metrikleri üzerinde yoğunlaşmaktadır.

Aynı yazılım projesinin farklı sürümleri üzerinde hata tahmini yürütülürken metriklere dayalı problemlerle daha az karşılaşılır. Çünkü aynı proje grubunda kullanılan metrik seti, değişen sürümlerde genellikle hem adet bakımından hem de tip bakımından aynıdır. Bu durum proje-içi tahmin olarak adlandırılmaktadır.

Farklı projelere ait veri setlerinde tahmin yürütülürken eğitim ve test işlemlerinin yapıldığı metrik grubu farklıdır. Bu tip tahmine heterojen veya çapraz-proje hata tahmini denmektedir [2]. Heterojen hata tahminindeki

temel problem aynı olmayan metrik gruplarındaki metriklerin eşleştirilmesidir.

İstatistiksel analizlerin sıklıkla kullanıldığı heterojen tahminde, kaynak ve hedef metriklerin dağılım benzerliğinin önemi vurgulanmaktadır [1]. Öğrenmenin başarısını arttırabilmek için heterojen tahminde transfer öğrenme yöntemleri de geliştirilmektedir. Buradaki amaç, birleştirilmiş metrik grupları üzerinden tahmin yürüterek heterojenlik problemini çözmektir. Bu sırada değişen metrik adetlerinde tahmin sonuçları yorumlanmaya çalışılır. Ayrıca eğitim ve test gruplarındaki veri setleri değiştirilerek tahmin performansına etkisi gözlemlenir [2].

Çapraz-proje tahmininde başvurulan yollardan biri eğitim veri seti seçimidir. Veriler dağılım özelliklerine bakılarak gruplanıp seçilir ve böylece tahmin performansı arttırılmaya çalışılır [3]. Buna ek olarak kümeleme gibi bazı denetimsiz öğrenme yaklaşımları ile de uygun eğitim veri seti grubu tespit edilmeye çalışılır. Kümeleme aynı zamanda yazılım metriklerinin seçiminde de kullanılmaktadır [5].

Eğitim ve test aşamasında kullanılan özellikler farklı olduğu için çapraz-proje tahmininde kullanılan sınıflandırıcıların performansları ilgili diğer deney türlerine göre düşüktür. Bu problemi çözmek için birden fazla sınıflandırıcının birleştirilmesiyle oluşturulan bileşik sınıflandırıcılara başvurulur [4]. Bu tip sınıflandırıcılar her eğitim tekrarındaki en iyi sınıflandırıcıya bakarak örnek etiketlemesi yapar. Bagging, boosting, stacking gibi türleri olan bileşik sınıflandırıcılara güvenmek çapraz-proje tahmini açısından geçici bir çözümdür. Nitekim gelişen yazılım sistemlerinde bileşik sınıflandırıcıların dahi tahmin performansını arttıramaması olasıdır. Çünkü yeni metrik gruplarının ortaya çıkması bileşik sınıflandırıcıları zorlamaktadır. Bunun yerine, metrik odaklı yöntemler geliştirmek, çapraz-proje tahmini için daha tercih edilebilirdir.

Örnek seçimi yerine özellik seçimine dayalı yöntemlerin çapraz-proje tahmininde daha etkili olduğu tespit edilmiştir [6,7]. Bu tespit özellikle eğri altına kalan alan (AUC) ve F-skor performans parametreleri için belirgindir. Ayrıca, özellik eşleştirmenin yanında yazılım modül büyüklüklerini de dikkate almak çapraz-proje tahmini açısından olumlu etki yapmaktadır [8].

Heterojen tahminde metrik eşleştirmesi için optimizasyon yöntemlerine de başvurulmaktadır. Örneğin Hydra isimli yöntem [9] bileşik sınıflandırıcılarla beraber genetik algoritmayı kullanarak hem uygun metrikleri seçebilmiş hem de hata tahmin performansını ilgili veri setlerinde F-skor açısından ortalamada 0.544 oranında arttırabilmiştir.

Hata tahmin veri setlerindeki verilerin sayısal olması nedeniyle bu alanda istatistiksel yöntemlere sıklıkla başvurulmaktadır. Çapraz-proje tahmini açısından metrik eşleştirmesi için genellikle oran-tabanlı eşleştirme (PAnalzer), Kolmogorov-Simirnov Testi ve Spearman korelasyonu kullanılmıştır [1]. Böyle yöntemlerde genellikle Öklid mesafeye dayalı bir analiz vardır. Aynı amaca yönelik fiziksel mesafe yerine iki vektör grubunun yakınlık oranına bakan alternatif yöntemlerden biri Kosinüs benzerlik yöntemidir [10].

Bu çalışmada Kosinüs benzerliğine dayalı çapraz-proje tahmini için CSCDP isimli bir metrik eşleştirme algoritması geliştirilmiştir. Geliştirilen algoritma ile eşleştirilen metrikler sonrasında C5.0, yapay sinir ağı, Random Forest ve Destek Vektör Makinası (SVM) yöntemlerine sokularak tahmin sonuçları kaydedilmiştir. F-skor, matthew korelasyon katsayısı (MCC) ve AUC performans parametrelerinde sonuçlar karşılaştırılmıştır. CSCDP'nin Kosinüs benzerliğini kapsamasının temel nedenlerini şu şekilde sıralayabiliriz:

- Kosinüs benzerlik yönteminin karmaşıklığı düşüktür.
- Hata tahmin veri setlerinde sıfır değerlerinin çoğunlukla görülmesi verilerde seyrekliğe neden olmaktadır. Diğer taraftan Kosinüs benzerliği seyrek verilerle uyumludur.
- Yapılan çalışmalarda, hata tahmin verileri normalizasyon sürecine sokularak belirli sayısal aralıklarda çalıştırılır. Kosinüs benzerliğinde de büyüklük yerine iki vektör arasındaki açıya bakılır.

Makale şu katkıları yapmaktadır; 1- Çapraz-proje tahmin başarısını arttıran CSCDP metrik eşleştirme yöntemi sunulmuştur. 2- Öklid uzaklık yerine vektörel benzerliğe dayalı yöntemlerin tercih edilebilir olduğu tespit edilmiştir. 3- Eğitim ve test aşamasında seyreklik oranlarının hedef ve kaynak veri setlerinin belirlenmesinde tahmin başarısı açısından olumlu etki yaptığı gözlemlenmiştir. 4-Yapay sinir ağının çapraz proje tahmininde diğer alternatiflerine göre daha üstün olduğu gözlemlenmiştir. 5-Seyreklik oranı düşük olan veri setlerinin eğitim veri seti olarak belirlenmesi tahmin başarısına olumlu etki yapmıştır.

Makalenin geri kalanındaki bölümler şu şekildedir: Bölüm 2'de çapraz-proje tahmini ile ilgili çalışmalar özetlenmiştir. Bu özet özellik ve örnek seçimi açısından çalışmaları iki bölümde anlatır. Yöntemin detayları, kullanılan veri setleri, deneysel sınıflandırıcılar ve performans parametreleri Bölüm 3'te anlatılmıştır. Deneysel bulgular Bölüm 4'te tablolar ve grafiklerle sunulmaktadır. Sonuçlar Bölüm 5'te özetlenmiştir.

2. LİTERATÜR ÖZETİ (SUMMARY OF LITERATURE)

Literatürdeki çalışmalar özellik seçimi, örnek seçimi ve öğrenme yöntemleri konularında yoğunlaşmaktadır. Bu bölümde, öne çıkan ilgili çalışmalar özetlenmektedir.

Aynı veri seti grubunda yazılım hata tahmin modellerinin gelişmediğini ilk ifade eden Zimmermann ve

arkadaşlarıdır [11]. Buna ek olarak onların çalışması, eğitim ve test verilerinin kombinasyonlarının tasarımı açısından yeni bir yol açmıştır. Bağlantı-tabanlı sınıflandırma algoritması geliştirilen bir çalışmada [12], denetimsiz öğrenme yöntemlerinin denetimli öğrenme yöntemlerine göre daha yüksek AUC sonuçları ürettiği gözlemlenmiştir. Buna ek olarak, SC isimli algoritma alternatiflerine tahmin performansı açısından üstün gelmiştir. Fakat SC uygun eğitim örneklerini üretmek yerine tüm veri setini etiketlemede kullanılmıştır.

Ryu ve arkadaşları [13] maliyet-hassas bileşik bir sınıflandırıcı önermişlerdir. Yöntemde, eğitim örneklerine ağırlık verilerek sınıf dengesizliği oranına bakılır. Bu yolla tahmin kesinliği arttırılmaya çalışılır. Güvenilirliğe dayalı sınıflandırıcılara da çapraz-proje tahmini için başvurulmaktadır. Bunlardan birinde, örneklerin standart dağılımından faydalanılarak naive-Bayes sınıflandırıcısının geliştirilmiş bir sürümü sunulmuştur. Yöntem, g-ortalama açısından temel naive-Bayes yöntemine üstün gelmiştir. Ancak, veri dağılım modelleri deney sırasında dikkate alınmamıştır [14].

Sınıf dengesizliği problemini çapraz-proje tahmininde çözebilmek için çeşitli algoritmalar geliştirilmiştir. CDE-SMOTE [15] isimli yöntem bunlardan biridir. CDE-SMOTE, hata veri setlerindeki sınıf dağılımını tahmin ederek sınıf dengesizliği problemini hafifletmeye çalışır. Ancak, CDE-SMOTE'un yalnızca süreç metriklerinde denenmesi, statik kod metriklerinde test edilmemesi geçerlilik için bir tehdittir. Buna ek olarak, yöntem metrik heterojenliğini de dikkate almamıştır. Herbold ve arkadaşları [16] 24 farklı çapraz-proje yöntemini karşılaştırmışlardır. Sonuçta, büyük ölçekli projelerle çalışılıyorsa performansın çapraz-proje yöntemine çok bağlı olmadığını tespit etmişlerdir. Bunun yerine, eğitim safhasında kullanılan öğrenme tipinin daha etkili olduğunu vurgulamışlardır. Öğrenme tipinin araştırıldığı bir başka çalışmada [17], çapraz-proje tahmini için yarı denetimli bir öğrenme yöntemi önerilmiştir. Bu yöntem, alternatif dört yöntemi tahmin performansı açısından geride bırakmıştır. Ancak, özellikle son yıllarda çapraz-proje tahmininde heterojenlik problemine odaklanıldığı görülmektedir. Bu bağlamda metrik eşleştirme çalışmaları ağırlıklıdır [18].

Herbold ve arkadaşları [19] çapraz-proje tahminini alışılagelmedik bir şekilde ele almışlardır. Eğitim örneklerinden faydalanmak yerine yazılım modüllerinin büyüklüklerini kullanarak tahmin yürütmüşlerdir. Yöntemleri zaman açısından uygulayıcılara zaman kazandırmaktadır. Diğer taraftan, eğitim örneklerine nüfuz eden yöntemlerin eğitim sürecini tamamlamadan verileri bölümlemesi gerekir. Böyle bir çalışmada [20], çapraz-proje tahmini için kümeleme tabanlı bir özellik seçim yöntemi geliştirilmiştir. Yöntem dört farklı rakip yönteme kesinlik, f-skor ve AUC parametrelerinde üstün gelmiştir. Yöntem umut verici sonuçlar vermesine rağmen, eğitim örneklerinin seçimiyle ilgili bir yenilik getirmemiştir. Porto ve arkadaşları [25] çapraz-proje tahmin başarısını arttırabilmek için bir meta-öğrenme yöntemi önermişlerdir. Dahası, çapraz-proje tahmin yönteminin tahmin edilen projenin özelliklerine bakarak seçilmesi gerektiğine işaret etmişlerdir.

Hata tahmininde özellik seçimi ve eşleştirmesini beraber uygulayan yöntemlerden birini Nam ve arkadaşları [1] geliştirmişlerdir. Yöntem ilgili deneysel veri setlerinde AUC açısından tahmin sonuçlarını kayda değer bir biçimde arttırmıştır. Benzer bir çalışma Catal ve arkadaşları [21] tarafından özellik seçimini sağlayan yeni bir algoritma önerilerek yapılmıştır. Ancak kullanılan veri setindeki gürültü oranı ve yanlış etiketli örnekler nedeniyle sonuçlar yanlış hesaplanmış veya yorumlanmış olabilir. Diğer bir deyişle, veri setlerindeki kusurların veya eksikliklerin böyle deneylerde giderilmesi güvenilirlik açısından önemlidir. Nitekim hata etiketini metriklerin olması nedeniyle yalnızca hata etiketlerini kullanarak örnek seçimini tamamlamak yetersizdir.

Fukushima ve arkadaşları [26] eğitim ve test örneklerini eşleştirmek için bir benzerlik metriği kullanmışlardır. Dahası, daha kesin çapraz-proje tahmin sonuçları için topluluk öğrenme yöntemlerini tavsiye etmişlerdir.

Özetle, hata tahmin veri ambarlarına farklı projelerden veriler geldiği için buradaki metrik tipleri birbirinden farklıdır. Dolayısıyla genel sonuçlar çıkarmak için farklı projeleri tahmin deneylerinde kullanmak gerekir. Bu nedenle, metrik eşleştirmeye dayalı yöntemler özellikle çapraz-proje tahmini için tercih edilebilirdir.

3. YÖNTEM (METHOD)

3.1 Seyreklik Analizi (Sparsity Analysis)

Hata tahmin veri setleri sıfır değerli alanlar içerebilir. Bu değerlerin yoğun olması hesaplama karmaşıklığını arttırır ve matris hesaplama performansını olumsuz etkiler. Bir matristeki seyreklik oranı Eşitlik 1'deki gibi hesaplanır [23]. Burada *S* seyrekli oranını temsil etmektedir.

$$S = \frac{sıfır\ değerli\ alan\ sayısı}{toplam\ alan\ sayısı} \quad (1)$$

Seyrek matrislerin neden olduğu problemlerden biri alan karmaşıklığıdır. Nitekim sıfır değerleri yoğun olan matrislerin boyutları büyür ve daha fazla belleğe ihtiyaç duyarlar. Bununla beraber, seyrek matrislerle uğraşan algoritmaların zaman karmaşıklıkları yüksektir. Sıfır değerleri ihmal edilerek sıfırdan farklı değerler depolanır. Bu yolla seyreklik problemiyle baş edilmeye çalışır. CSCDP algoritmasında da seyrek matrislerdeki sıfır değerleri ihmal edilerek tahmin deneyi yürütülmüştür. Bunun için R'de "glmnet" kütüphanesinden faydalanılmıştır.

3.2 Kosinüs Benzerliği (Kosinüs Similarity)

Kosinüs bezerliği, iki veya daha fazla sayısal vektör arasındaki benzerliği ölçen istatistiksel bir yöntemdir. İki

vektör p ve q ile temsil edildiğine, bu iki vektöre ait Kosinüs benzerliği Eşitlik 2'deki formül ile hesaplanır. Formüldeki $p.q=p_1.q_1+p_2.q_2+\ldots+p_nq_n$ ile hesaplanır. Örneğin p (2,3) ile q(1,2) için (2*3)+(3*2)=12 bulunur. ||p|| ise vektördeki tüm değerlerin toplamının karekköküdür. Eşitlik 2'deki sonucun $\cos(\theta)$ ile ifade edilmesinin nedeni sonucun iki vektör arasındaki açıyı temsil etmesidir.

$$\cos(\theta) = \frac{p.q}{||p||.||q||} \quad (2)$$

3.3 CSCDP Algoritması (CSCDP Algorithm)

CSCDP, öncelikle veriler üzerinde normalizasyon işlemi yapmaktadır. Böylece her metriğin aynı standart dağılıma sahip olması sağlanır. CSCDP'de Z-normalizasyon [22] kullanılarak veriler 0-1 aralığında temsil edilir. Bu sırada "0" içeren alanlar göz ardı edilmektedir. Bunun nedeni CSCDP'nin seyreklik (Sparsity) analizini de kapsamasıdır. Nitekim çapraz-proje tahmininde kaynak ve hedef projeler eğitim ve test için belirlenirken seyreklik oranlarına bakılmaktadır. "0" değer oranları birbirine yakın olan proje veri setleri hedef veya kaynak projelerde gruplandırılmaktadır. Algoritma 1'de CSCDP'nin adımları görülmektedir.

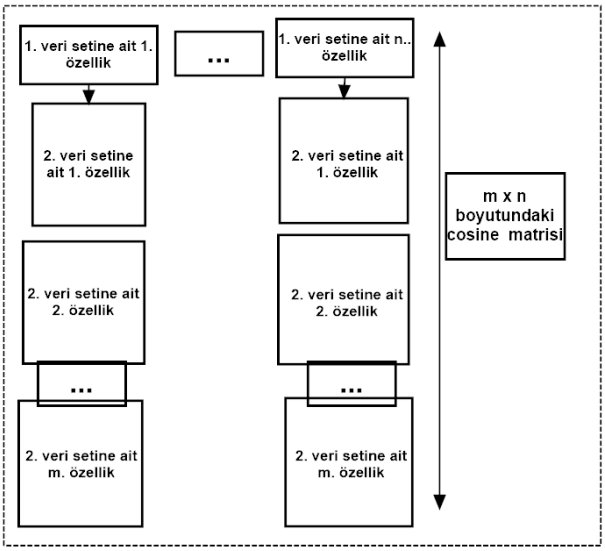
Algoritma 1. CSCDP algoritması
(CSCDP Algorithm)

Giriş: Hata tahmin veri setleri: $dp_1\ldots\ldots dp_n$
Çıkış: Eşleştirilmiş metrikler: m_{11}- $m_{xy,}$ m_{12}- $m_{zt\ldots\ldots}m_{nt}$-m_{qt}

For *1*.....*n*
Adım 1. Seyreklik analizi : $s_1\ldots\ldots s_n$
Döngü sonu
Adım 2. Z-normalizasyon ($dp_1\ldots\ldots dp_n$)
For *1*.....*n*
For *1*.....*n*
For *1*.....*m*
Adım 3. Kosinüs benzerlik analizi
Döngü sonu
Döngü sonu
Döngü sonu
Adım 4. Tüm verilerin kontrolü
Adım 5. Eşleştirilmiş metriklerin üretilmesi
Adım 6. Eşleştirilmiş metrik listesi *L* döndürülür

Algoritma 1 giriş olarak hata tahmin veri setlerini almaktadır. Bu veri setlerinin adedi n ile gösterilirse, $dp_1\ldots\ldots dp_n$ veri seri grubudur. Adım 1'de tüm veri setleri seyreklik analizine sokulmaktadır. Bu işlem için R dilinde yazılmış ve çalıştırılmıştır. Adım 2'de veri setleri z-normalizasyon sürecinden geçirilerek 0-1 aralığına daraltılmıştır. Adım 3, Kosinüs benzerlik analizi için iç içe üç döngüyü kapsamaktadır. İlk iki döngü tüm veri setlerinin birbiriyle karşılaştırılmasını, üçüncü döngü ise karşılaştırmanın tüm metrikler için ayrı ayrı yapılmasını sağlamaktadır. Üçüncü döngüdeki *m*, metrik adedini temsil etmektedir. Bu işleme ait kod parçası herhangi iki veri seti için Ek-B'de sunulmuştur. Burada "rlist" ve "coop" paketlerinden faydalanılmıştır. Kosinüs benzerliğini uygulayabilmek için iki vektörün uzunluğunu birbirine eşit olması gerekmektedir. Diğer taraftan, deneysel veri setlerinin uzunlukları birbirinden farklıdır. Böyle durumlarda, deney kapsamındaki veri setlerinde daha küçük ölçekli olanların örnek sayıları temel alınarak büyük ölçekli veri setlerinin tüm örneklerini kapsayacak şekilde tekrarlı bir Kosinüs benzerlik analizi yürütülmüş ve ortalama değerler kaydedilmiştir.

Adım 4, iki veri setindeki ölçek farkına bağlı olarak karşılaştırılmamış örnek kalmaması için bir kontrol mekanizmasını kapsar. Tüm veriler kapsandığında diğer benzerlik karşılaştırmasına geçilir. Adım 5, Kosinüs benzerlik sonuçlarına göre, eğitim için kullanılacak metriklerin en benzer test metriğine ilişkilendirilmesidir. Bunun için Kosinüs benzerliği en yüksek çıkan metrik eşleştirilir. Son adım olan Adım 6 eşleştirilmiş listeyi döndürür ve tahmin deneyinin yürütülmesine geçilir.



Şekil 1. Kosinüs benzerlik eşleştirmesi matrisi (Cosine similarity matrix)

Şekil 1'de CSCDP algoritmasının 1. ve 2. veri seti için örnek sonuçları görülmektedir. Veri setlerinin öncelikle ikili kombinasyonları (ant-cm1, ant-jm1, ant-kc1) oluşturulur. Soldaki veri setinin *m* ve sağdaki veri setinin *n* özelliği olduğu varsayılırsa *mxn* boyutunda bir kosinüs benzerlik değer matrisi üretilir. Birinci veri setindeki tüm özelliklerin ikinci veri setindeki diğer özelliklerle kombinasyonu ile kosinüs benzerlik analizi sonuçları üretilmiş olur.

3.4 Veri Setleri (Data Sets)

Deneyde kullanılan veri setleri tera-promise veri ambarından alınmıştır [24]. Bu ambar, birçok açık kaynak kodlu ve endüstriyel hata tahmin veri setini içermektedir. Veri setlerinin detayları Tablo 1'de sunulmuştur. Tabloda sunulan bazı veri setlerinde örnek sayısından fazla hata vardır. Böyle veri setlerinde ilgili yazılım bileşeninde birden fazla hata olduğu anlaşılmaktadır.

Tablo 1. Deneysel veri setleri
(Experiemntal data sets)

İsim	Sürüm	Örnek Sayısı	Hata sayısı	Hatalılık oranı (%)
ant	1.7	745	338	22
arc	1	234	33	11
berek	1	70	33	43
camel	1.0	339	14	3
camel	1.2	608	522	41
camel	1.4	872	335	16
camel	1.6	965	188	19
ckjm	1.8	10	23	50
e-learning	1	64	9	13
ivy	1.1	111	233	56
ivy	1.4	241	18	6
ivy	2.0	352	56	11
jedit	3.2	272	382	90
jedit	4.0	306	226	24
jedit	4.1	312	217	25
jedit	4.2	367	106	13
jedit	4.3	492	12	2
kalkulator	1	27	7	25
log4j	1.0	135	61	25
log4j	1.1	109	82	33
log4j	1.2	205	498	92
lucene	2.0	195	268	46
lucene	2.2	247	413	57
lucene	2.4	340	630	59
nieruchomosci	1	27	13	37
tomcat	6	858	114	8
xalan	2.4	723	155	15
cm1	1	327	33	12
jm1	1	10878	217	19
kc1	1	2107	218	15
kc2	1	521	105	20
kc3	1	458	42	9
pc1	1	1107	104	7
pc2	1	5589	4	0.4
pc3	1	1563	155	10
pc4	1	1458	150	12

3.5 Performans parametreleri ve deneysel ayarlar (Performance Parameters and Experimental Settings)

Deneyde, AUC, F-skor ve MCC performans parametreleri kullanılmıştır. Bu parametrelere ait detaylar Tablo 2'de sunulmuştur. Tahmin işleminde hatalı ve hatasız yazılım bileşenleri hakkında verilen hükümler karışıklık matrisi ile ifade edilir. Bu matriste TP, TN, FP ve FN olmak üzere dört farklı alan vardır [27, 28]. Kısaltmaların ilk harfi tahminin doğru veya yanlış (T/F) olduğunu, ikinci harfi de tahmin edilen bileşenin hatalı veya hatasız (P/N) olduğunu göstermektedir. Nitekim hatalı bir bileşen doğru tahmin edildiyse TP alanı bir arttırılır. Tablo 2'de sunulan parametrelerden AUC'nin tanımındaki *m* veri miktarını, *i* çalıştırma adedini, *j* de *n* veri noktasının çalıştırma adedini temsil etmektedir. *i* ve *j* 'nin olasılıkları sırasıyla p_i ve p_j ile temsil edildiğinde $p_i > p_j$ ise formül 1 üretir.

Tahmin işleminde SVM, yapay sinir ağı ve Random Forest algoritmaları kullanılmıştır. Sonuçlar 10*10 çapraz doğrulama ile elde edilmiştir.

Tablo 2. Performans parametreleri
(Performance parameters)

İsim	formül
MCC	$\frac{(TP * TN - FP * FN)}{\sqrt{(TP+FP) * (TP+FN) * (TN+FP) * (TN+FN)}}$
AUC	$\frac{1}{mn}\sum_{m}^{i=1}\sum_{n}^{j=1} 1\ p_i > p_j$
F-skor	$\frac{2TP}{2TP+FP+FN}$
Sınıflandırma hatası	$E=f/n\ *100$

4. DENEYSEL BULGULAR (EXPERIMENTAL FINDINGS)

Seyreklik analizine dayalı olarak kaynak ve hedef olarak gruplanan veri setlerinin seyreklik analiz sonuçları Tablo 3'teki gibidir.

Küçük ölçekli ckjm, kalkulator ve e-learning gibi veri setlerinin seyreklik oranı 0.25'in üzerindedir. Genel olarak tüm veri setlerinde seyreklik oranı 0.5'in altındadır. Ancak daha küçük ölçekli veri setlerinde seyreklik oranı daha yüksek bulunmuştur. Deneyde seyreklik eşik değeri Tablo 3 dikkate alınarak 0.24 olarak belirlenmiştir. Bu değer Tablo 3'teki değerlerin ortalamasıdır. 0.24 ve altında seyreklik değerine sahip olan veri setleri bir grup, 0.24'ün

Tablo 3. Seyreklik analizi sonuçları
(Results of sparsity analysis)

Veri seti	Seyreklik oranı
pc3.csv	0.2004607
jedit-4.1.csv	0.2004808
jedit-3.2.csv	0.2036765
cm1.csv	0.2045558
jedit-4.0.csv	0.2052288
jedit-4.2.csv	0.2055858
lucene-2.4.csv	0.2098529
jedit-4.3.csv	0.2184959

xalan-2.4.csv	0.2193638
lucene-2.0.csv	0.2230769
berek.csv	0.2232558
camel-1.0.csv	0.2249263
xalan-2.5.csv	0.2278331
xalan-2.6.csv	0.2359887
camel-1.2.csv	0.2373355
xalan-2.7.csv	0.2385589
camel-1.6.csv	0.2394301
pc4.csv	0.2406789
camel-1.4.csv	0.2419151
ivy-1.1.csv	0.2467532
log4j-1.2.csv	0.25
kc1.csv	0.2608207
kc2.csv	0.2608207
kc3.csv	0.2608207
pc1.csv	0.2608207
pc2.csv	0.2608207
ivy-1.4.csv	0.2703023
ivy-2.0.csv	0.2709686
log4j-1.1.csv	0.2729358
log4j-1.0.csv	0.2751852
jm1.csv	0.2784341
ant.csv	0.2814716
kalkulator.csv	0.2907407
arc.csv	0.2925214
ckjm.csv	0.3047619
tomcat.csv	0.3079254
e-learning.csv	0.3184524

üzerinde seyrekliğe sahip veri setleri ise ikinci gruptur. İki grup veri seti eğitim ve test sürecinde hedef ve kaynak veri setleri olarak iki yönlü bir deneyde kullanılmıştır. Nitekim birinci grup veri setleri önce hedef daha sonra kaynak olmak üzere iki yönlü hata tahmin sonuçları üretmişlerdir.

Tablo 4'te üç farklı sınıflandırıcının üç farklı performans parametresinde ürettiği sonuçlar sunulmuştur. Tablodaki numara alanı tahmin numarasını gösterirken, Kaynak=>Hedef sütununda sol taraftaki veri seti eğitim veri seti, sağ taraftaki veri setleri ise ilgili sınıflandırmanın test verisini oluşturmaktadır. Bu tahmin deneylerinin ilk 19 tanesi deneyin birinci bölümüdür. 20. Tahminden itibaren eğitim veri seti olarak kullanılan veri setleri test için kullanılarak yer değiştirilmiştir. Böylece eğitim ve test aşamasında kullanılan veri setine bağlı olarak elde edilen sonuçlar daha iyi yorumlanabilir. Eğitim ve test veri seti grupları oluşturulurken seyreklik analizi değerlerine bakılmış, 0.24 eşik değer olarak belirlenmiştir.

Tablodaki sonuçlar incelendiğinde neredeyse tüm sınıflandırıcılarda seyreklik oranı 0.24'ün altındaki veri setlerinin eğitim veri seti olarak belirlendiği ilk 19 tahmin deneyinin değerlerinin 20-38 arasındaki tahmin sonuçlarına göre daha başarılı olduğu görülmektedir. Örneğin 1-19 satırları için veri setlerinin başarı değerleri toplamı 108.58 dir. Diğer taraftan 20-38 satırları için bu değer 94.66 olarak hesaplanır. 1-19 satırları için sınıflandırıcılar karşılaştırıldığında en yüksek değerleri yapay sinir ağının ürettiği görülmektedir. Bu nedenle seyreklik oranı düşük olan veri setlerinin hata tahmin deneylerinde eğitim veri grubu olarak kullanılması tahmin başarısı açısından olumlu bir etki yaratmaktadır. Diğer taraftan, sınıflandırıcılar içerisinde en başarılısı yapay sinir ağıdır. Tahmin deneyleri R dilinde "neuralnet", "randomforest" ve "e1071" kütüphanelerinden faydalanılarak gerçekleştirilmiştir.

Şekil 2 (a) ve (b)'de CSCDP kullanılmadan ve kullanılarak elde edilen sınıflandırma hata oranları sırasıyla Random Forest algoritması için çizdirilmiştir. Bu parametre Tablo 2'de sınıflandırma hatası ile verilen formüldür. Formülde E hatayı temsil ederken, f yanlış sınıflandırılan hata adedini ve n ise toplam örnek sayısını göstermektedir. Formül tüm veri setlerine uygulanıp toplandıktan sonra veri seti adedine bölünerek ortalama sınıflandırma hatası bulunur. Grafiklerdeki yeşil çizgi hatalı örnekleri temsil ederken, kırmızı çizgi hatasız örnekleri göstermektedir. Bu çizgiler sırasıyla true ve false ile etiketlenmiştir. Grafiklerdeki out-of-bag (OOB) etiketi ile temsil edilen siyah çizgi ise her eğitim gözlemi için ortalama hata oranına atıf yapar. OOB ağaç örneklemesi sırasında rastgele üretilen bir saydır. Çünkü rastgele örneklenen verilerden elde edilir.

Random Forest algoritmasında ağaç sayısına bağlı olarak elde edilen Şekil 2 (a) ve (b) grafikleri incelendiğinde, CSCDP algoritması kullanılarak elde edilen hata değerlerinin CSCDP kullanılmadan elde edilen hata değerlerine göre oldukça düşük olduğu görülmektedir. Hatalı örnekler için Şekil 2 (a)'da 0.9 olarak elde edilen hata oranı, Şekil 2 (b)'de 0.15 düzeyindedir. Ortalama hata oranı CSCDP ile Şekil 3'te 0.1'e 100 gözlemde indirilebilmişken, bu oran aynı tekrar için Şekil 2 (a)'da 0.2'de kalmıştır. Tüm veri setleri ortalamasında Random Forest 0.65 oranında sınıflandırma hatasını F-skor için azaltabilmiştir.

Deneydeki sınıflandırıcılar üzerinde parametre optimizasyonu yapılmamıştır. Otomatik ayarlar kullanılarak deney tamamlanmıştır. Örneğin SVM için radyal çekirdek, 155 destek vektörü ile gamma ayarı 0.05'tir.

Seçilen performans parametreleri benzer çalışmalarda sıklıkla kullanılmaktadır. Bununla beraber, MCC parametresi, tahmin deneylerinde örnek ve özellik düzeyinde geliştirilen yöntemlerin performans değerlendirmelerinde tercih edilmektedir. Bunların dışında, g-ortalama gibi performans parametreleri de

literatürde mevcuttur. Ancak böyle parametreler, hata tahmininde özellikle sınıf dengesizliği gibi özel problemlerde kullanılmaktadır. Bu deneyin geçerliliği için bir tehdit oluştursa da, yapay sinir ağının deneye dahil edilmesi derin öğrenme [29] açısından araştırmacıları teşvik etmektedir.

Deneyde kullanılan veri setleri ağırlıklı olarak tera-promise veri ambarında yer alan açık kaynak kodlu projelerden alınmadır. Her ne kadar açık kaynak kodlu projelerde yapılan deneyler umut verici sonuçlar üretse de endüstriyel projelerden üretilen hata tahmin veri setlerinde ihtiyaç vardır. Deneyin geçerliliğinin doğrulanması için endüstriyel veri setlerinin yöntemde test edilmesi planlanmaktadır.

Deneyde kullanılan sınıflandırıcılar yaygın olarak kullanılan sınıflandırıcılardır. Ancak son yıllarda özellikle derin öğrenmenin hata tahmininde kullanımı yaygınlaşmaktadır. Özellikle tahmin heterojenliği, sınıf dengesizliği gibi problemlerdeki derin öğrenmenin başarısı belirsizdir.

5. SONUÇLAR (RESULTS)

Yazılım hata tahmini yeni yöntemlerle gelişen ve araştırmacıların ilgisini çeken popüler bir konudur. Son yıllarda sık ele alınan çapraz-proje tahminine yönelik bu çalışmada CSCDP isimli metrik eşleştirme algoritması geliştirilmiştir. Geliştirilen algoritma 36 farklı veri setinde üç farklı sınıflandırıcıda elde edilen sonuçlar ile değerlendirilmiştir. Elde edilen sonuçlara göre;

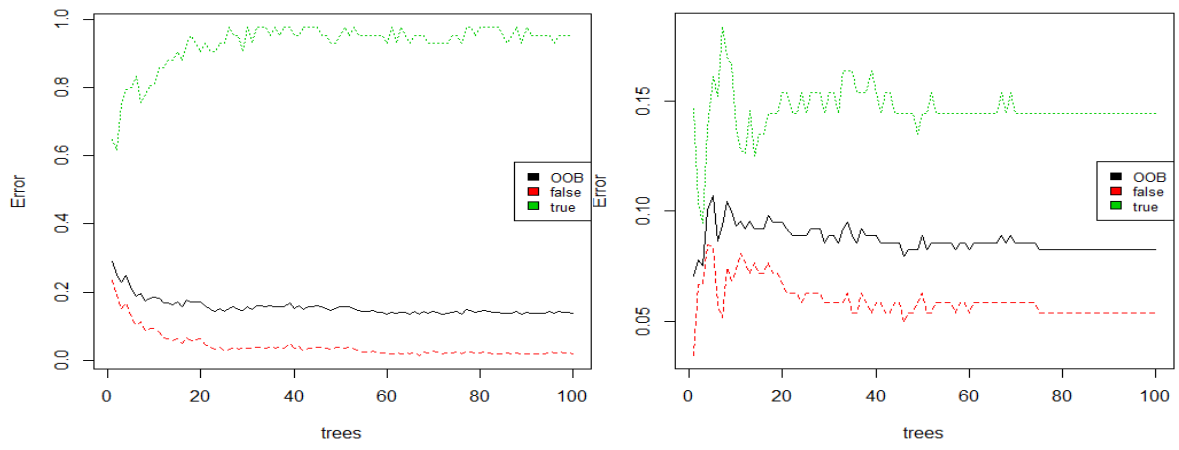
1. Çapraz proje tahmininde seyreklik oranı düşük veri setlerinin eğitim veri seti olarak belirlenmesi tahmin başarısını arttırmaktadır,
2. Yapay sinir ağı sınıflandırıcısı diğer sınıflandırıcılara göre daha yüksek sonuçlar üretmiştir.

CSCDP algoritması ile eşleştirilen metriklerin hata tahmin performansları özellikle MCC performans parametresinde daha umut vericidir. Ayrıca, eğitim ve test veri gruplarının belirlenmesinde seyreklik analizinin kullanılması tüm veri setlerinde tahmin başarısını arttırmıştır. Özetle, vektörel büyüklüğe dayalı olmayan analizlerin metrik eşleştirmesinde tercih edilebilir olduğu görülmüştür. Nitekim hata tahmin çalışmalarında normalizasyona başvurulması vektörel büyüklüğün önemini azaltmaktadır. Çapraz projelerde tahmin başarısını iyileştirebilmek için hiperparametre optimizasyon yöntemlerinden faydalanılabilir veya bu probleme özel bir optimizasyon yöntemi geliştirilebilir.

Tablo 4. CSCDP çapraz-proje tahmin sonuçları
(Cross-project defect prediction results of CSCDP)

Numara	**Kaynak=>Hedef**	**SVM**			**Yapay sinir ağı**			**Random Forest**		
		AUC	**F-skor**	**MCC**	**AUC**	**F-skor**	**MCC**	**AUC**	**F-skor**	**MCC**
1	ivy-1.1=>**pc4**	0.55	0.59	0.57	0.58	0.64	0.74	0.59	0.60	0.59
2	camel-1.4=>**pc2**	0.57	0.63	0.56	0.56	0.71	0.75	0.58	0.61	0.52
3	camel-1.6=>**pc1**	0.56	0.62	0.61	0.63	0.74	0.76	0.54	0.63	0.63
4	xalan-2.7=>**kc1**	0.55	0.63	0.62	0.62	0.69	0.77	0.56	0.64	0.64
5	camel-1.2=>**kc2**	0.56	0.64	0.60	0.61	0.68	0.76	0.58	0.60	0.59
6	xalan-2.6=>**kc3**	0.57	0.65	0.63	0.60	0.73	0.78	0.61	0.56	0.65
7	xalan-2.5=>**jm1**	0.55	0.61	0.59	0.64	0.77	0.79	0.64	0.59	0.58
8	camel-1.0=>**e-learning**	0.49	0.67	0.58	0.69	0.76	0.80	0.59	0.62	0.51
9	berek=>**tomcat**	0.57	0.64	0.61	0.70	0.75	0.69	0.58	0.60	0.57
10	lucene-2.0=>**ckjm**	0.55	0.65	0.60	0.72	0.71	0.65	0.59	0.61	0.59
11	xalan-2.4=>**arc**	0.56	0.66	0.59	0.59	0.69	0.76	0.57	0.61	0.60
12	jedit-4.3=>**kalkulator**	0.58	0.63	0.58	0.78	0.68	0.77	0.55	0.62	0.64
13	lucene-2.4=>**ant-1.7**	0.59	0.64	0.60	0.76	0.70	0.75	0.63	0.58	0.62
14	jedit-4.2=>**log4j-1.0**	0.58	0.66	0.61	0.78	0.67	0.79	0.59	0.57	0.63
15	jedit-4.0=>**log4j-1.1**	0.61	0.65	0.63	0.77	0.68	0.74	0.62	0.59	0.58
16	jedit-3.2=>**ivy-2.0**	0.59	0.62	0.62	0.76	0.65	0.73	0.56	0.60	0.57
17	jedit-4.1=>**ivy-1.4**	0.58	0.67	0.61	0.75	0.64	0.72	0.53	0.61	0.60
18	cm1=> **log4j-1.2**	0.55	0.65	0.62	0.78	0.68	0.79	0.54	0.61	0.62
19	pc3=>**pc2**	0.56	0.59	0.60	0.80	0.60	0.76	0.56	0.52	0.63
20	**pc4**=>ivy-1.1	0.51	0.59	0.56	0.54	0.54	0.54	0.51	0.53	0.54
21	**pc2**=>camel-1.4	0.52	0.61	0.55	0.55	0.53	0.59	0.52	0.63	0.51
22	**pc1**=>camel-1.6	0.51	0.62	0.54	0.50	0.57	0.58	0.51	0.69	0.51
23	**kc1**=>xalan-2.7	0.47	0.58	0.57	0.52	0.58	0.58	0.47	0.57	0.52
24	**kc2**=>camel-1.2	0.48	0.60	0.53	0.49	0.60	0.52	0.51	0.56	0.52
25	**kc3**=>xalan-2.6	0.48	0.57	0.53	0.47	0.57	0.61	0.56	0.51	0.54
26	**jm1**=>xalan-2.5	0.49	0.59	0.59	0.49	0.59	0.63	0.56	0.53	0.54
27	**e-learning** =>camel-1.0	0.47	0.60	0.60	0.47	0.62	0.55	0.57	0.58	0.65
28	**tomcat**=> berek	0.54	0.58	0.57	0.54	0.58	0.54	0.54	0.54	0.56
29	**ckjm**=> lucene-2.0	0.53	0.60	0.59	0.53	0.60	0.59	0.53	0.55	0.52
30	**arc**=> xalan-2.4	0.53	0.59	0.52	0.53	0.59	0.51	0.52	0.56	0.53

31	**kalkulator**=>jedit-4.3	0.52	0.63	0.58	0.52	0.63	0.58	0.50	0.59	0.54
32	**ant-1.7**=>lucene-2.4	0.51	0.61	0.56	0.51	0.56	0.54	0.50	0.60	0.56
33	**log4j-1.0**=>jedit-4.2	0.56	0.62	0.59	0.56	0.59	0.59	0.56	0.61	0.59
34	**log4j-1.1**=>jedit-4.0	0.58	0.59	0.54	0.56	0.59	0.57	0.57	0.54	0.54
35	**ivy-2.0**=>jedit-3.2	0.49	0.58	0.55	0.51	0.58	0.53	0.43	0.52	0.54
36	**ivy-1.4**=> jedit-4.1	0.51	0.60	0.62	0.52	0.60	0.54	0.52	0.61	0.60
37	**log4j-1.2**=>cm1	0.54	0.62	0.61	0.49	0.58	0.59	0.59	0.60	0.58
38	**pc2**=>pc3	0.53	0.55	0.54	0.48	0.52	0.51	0.55	0.54	0.56



(a) **(b)**

Şekil 2**.** Random Forest algoritmasına ait tüm veri setlerinin ortalama tahmin hata oranları a) (CSCDP algoritması kullanılmadan elde edilmiştir) b) (CSCDP algoritması kullanılarak elde edilmiştir).
(Mean error rates of Random Forest for all the data sets a) produced without CSCDP b) produced with CSCDP

KAYNAKLAR (REFERENCES)

[1] J. Nam, W. Fu, S. Kim, T. Menzies, T., L. Tan, "Heterogeneous defect prediction", *IEEE Transactions on Software Engineering*,(1), 2017.

[2] S. Wang, L. Taiyue, L. Tan. "Automatically learning semantic features for defect prediction", **Software Engineering (ICSE), IEEE/ACM 38th International Conference** , 297-308, 2016.

[3] S. Herbold, "Training data selection for cross-project defect prediction", **Proceedings of the 9th International Conference on Predictive Models in Software Engineering**, ACM, 6, 2013.

[4] Y. Zhang, X. Lo, J. Sun, "An empirical study of classifier combination for cross-project defect prediction", **Computer Software and Applications Conference (COMPSAC)**, 2, 264-269, 2015.

[5] C. Ni, "A Cluster Based Feature Selection Method for Cross-Project Software Defect Prediction", *Journal of Computer Science and Technology*, 32(6), 2017.

[6] Q. Yu, J. Shujuan, Y. Zhang, "A feature matching and transfer approach for cross-company defect prediction", *Journal of Systems and Software*, 132, 2017.

[7] Q. Yu, S. Jiang, J. Qian, "Which is more important for cross-project defect prediction: instance or feature?", **Software Analysis, Testing and Evolution (SATE), International Conference**, 90-95, 2016.

[8] Y. Zhou, Y. Yang, H. Lu, L. Chen, Y., Zhao, "How Far We Have Progressed in the Journey?, An Examination of Cross-Project Defect Prediction", *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 27(1), 1, 2018

[9] X. Xia, L. O. David, S. J. Pan, N. Nagappan, X.Wang, "Hydra: Massively compositional model for cross-project defect prediction", *IEEE Transactions on software Engineering*, 42(10), 2016.

[10] H. V. Nguyen, L. Bai, "Kosinüs similarity metric learning for face verification", **Asian conference on computer vision**, 709-720, 2010.

[11] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, B. Murphy, "Cross-project defect prediction: a large scale experiment on data vs. domain vs. process", **Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering**, 91-100, 2009.

[12] F. Zhang, Q. Zheng, Y. Zou, A. E. Hassan, "Cross-project defect prediction using a connectivity-based unsupervised classifier",

Proceedings of the 38th International Conference on Software Engineering, 309-320, 2016.

[13] D. Ryu, J. I. Jang, J. Baik, "A transfer cost-sensitive boosting approach for cross-project defect prediction", *Software Quality Journal*, 25(1), 2017.

[14] W. N Poon, K. E. Bennin, J. Huang, P. Phannachitta, J. W. Keung, "Cross-project defect prediction using a credibility theory based naive bayes classifier", **Software Quality, Reliability and Security (QRS), IEEE International Conference**, 434-441, 2017.

[15] N. Limsettho, K. E. Bennin, J. W. Keung, H. Hata, H., & K. Matsumoto, "Cross project defect prediction using class distribution estimation and oversampling", *Information and Software Technology*, 100, 2018.

[16] S. Herbold, A. Trautsch, J. Grabowski, "A comparative study to benchmark cross-project defect prediction approaches", **Proceedings of the 40th International Conference on Software Engineering**, 1063-1063, 2017.

[17] F. Wu, X. Y. Jing, X. Dong, J., "Cross-project and within-project semi-supervised software defect prediction problems study using a unified solution", **Software Engineering Companion (ICSE-C), IEEE/ACM 39th International Conference**, 195-197, 2017.

[18] X. Jing, F. Wu, X. Dong, F. Qi, B. Xu, "Heterogeneous cross-company defect prediction by unified metric representation and CCA-based transfer learning", **Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering**, 496-507, 2015.

[19] S. Herbold, "Benchmarking cross-project defect prediction approaches with costs metrics", *arXiv preprint arXiv:1801.04107*, 2018.

[20] S. Herbold, "Crosspare: a tool for benchmarking cross-project defect predictions", **30th IEEE/ACM International Conference on Automated Software Engineering Workshop (ASEW)**, IEEE, 90-96, 2015.

[21] C. Catal, M. Song, C. Muratli, E. H. J. Kim, M. A. Tosuner, Y. Kayikci, "Cross-Cultural Personality Prediction based on Twitter Data", *Journal of Software*, 12(11), 2017.

[22] J. Hann, M. Kamber, "Data Mining: Concepts and Techniques", **Morgan Kaufman Publishers**, 2000.

[23] T. A. Davis, S. Rajamanickam, W. Sid-Lakhdar, "A survey of direct methods for sparse linear systems", *Acta Numerica*, 25, 2016.

[24] Internet: T. Menzies, R. Krishna, D. Pryor, The Promise Repository of Empirical Software Engineering Data**,** http://openscience.us/repo. North Carolina State University, Department of Computer Science, 2016.

[25] F. Porto, L. Minku, E. Mendes, A. Simao, "A systematic study of cross-project defect prediction with meta-learning", *arXiv preprint arXiv:1802.06025*, 2018.

[26] T. Fukushima, Y. Kamei, S. McIntosh, K. Yamashita, K., N. Ubayashi, "An empirical study of just-in-time defect prediction using cross-project models", **Proceedings of the 11th Working Conference on Mining Software Repositories**, 172-181, 2014.

[27] B. Karaöz, U. T. Gürsoy, "Adaptif Öğrenme Sözlüğü Temelli Duygu Analiz Algoritması Önerisi", *Bilişim Teknolojileri Dergisi*, 11(3), 2018.

[28] U. Ayvaz, H. Gürüler, "Bilgisayar Kullanıcılarına Yönelik Duygusal İfade Tespiti", *Bilişim Teknolojileri Dergisi*, 10(2), 2017.

[29] M. A. Kızrak, B. Bolat, "Derin Öğrenme ile Kalabalık Analizi Üzerine Detaylı Bir Araştırma", *Bilişim Teknolojileri Dergisi*, 11(3), 2018.