Fırat Üniversitesi Müh. Bil. Dergisi
34(1), 181-192, 2022

Araştırma Makalesi
https://doi.org/10.35234/fumbd.997863

Nesne Tabanlı Ölçütlerle Yazılım Hata Kestirimi : Örnek Bir Olay İncelemesi

Begüm ERKAL[1*], Tülin ERÇELEBİ AYYILDIZ[2]
[1] Bilgisayar Mühendisliği, Mühendislik Fakültesi, Başkent Üniversitesi, Ankara, Türkiye
[2] Bilgisayar Mühendisliği, Mühendislik Fakültesi, Başkent Üniversitesi, Ankara, Türkiye
[*1] begume@baskent.edu.tr, [2] ercelebi@baskent.edu.tr

(Geliş/Received: 20/09/2021; Kabul/Accepted: 03/02/2022)

Öz: Yazılım projelerinin kalitesini ölçme işlemi sorunların erken safhada bulunmasına olanak sağlayan bir işlem olmasına rağmen çoğunlukla ihmal edilen, zaman ve bütçe ayrılmayan bir işlemdir. Yazılım kalitesini ölçebiliyor ve bunu rakamlarla ifade edebiliyor olmak bakım yapılabilirlik, yeniden kullanılabilirlik, test edilebilirlik, verimlilik, taşınabilirlik, işlevsellik ve anlaşılabilirlik gibi kalite özellikleri hakkında fikir sahibi olmamızı sağlar. Yazılımın kalitesini etkileyen önemli unsurlardan biri de kaynak koddaki hataların sayısıdır. Bu nedenle, geliştirmenin erken safhasında kaynak koddaki hataları belirlemek çok önemlidir. Çalışmada, 50 açık kaynak kodlu eğitim projesindeki yazılım hata kestirim doğruluğu analiz edilmiştir. Yazılım kalite ölçütleri "Understand" adı verilen statik kod analiz aracı kullanılarak ölçülmüş ve hata sayıları ise "SpotBugs" adı verilen araç ile belirlenmiştir. Birisi dışarıda çapraz doğrulama (LOOCV) kullanılarak sonuçların doğruluğu belirlenirken yazılım kalitesi ve hatalar arasında ilişkiyi elde etmek için adımsal (stepwise) doğrusal regresyon analizi uygulanmıştır. Sonuçlara bakıldığında kestirim doğruluğunun literatürde de geçen değerler tarafından desteklendiği görülmüştür. RFC ölçütünün hata kestiriminde hiç bir etkisinin olmadığı gözlemlenmiştir. Çalışma, yazılım hatalarının sayısının kestirimi için yazılım kalite ölçütlerinin kullanılmasının mümkün olduğunu göstermektedir.

Anahtar kelimeler: Yazılım Hata Kestirimi, Nesne Tabanlı Ölçütler, Yazılım Kalite Ölçütleri, Kestirim Doğruluğu.

Software Bug Prediction with Object-Oriented Metrics : A Case Study

Abstract: Although the process of measuring quality of software projects is a process that allows problems to be found at an early stage, it is a process that is mostly neglected and does not allocate time and budget. Being able to measure software quality and quantify it allows us to gain insight into quality features such as maintainability, reusability, testability, efficiency, portability, functionality and understandability. One of the important factors affecting the quality of software is number of bugs in source-code. Therefore, it is very important to identify bugs in the source-data early in development. In study, software bug prediction accuracy in 50 open-source education projects was analyzed. Software quality metrics were measured using static code analysis tool called "Understand", number of bugs was determined with tool called "SpotBugs". Stepwise linear-regression analysis was applied to obtain the relationship between software quality and bugs while one was determining accuracy of results determined using leave-one-out cross validation (LOOCV). Results were supported by the values of estimation accuracy, which are also mentioned in literature. It has been observed that RFC metric has no effect on bug prediction. Study shows that it is possible to use software quality metrics to estimate number of software bugs.

Key words: Software Bug Prediction, Object-Oriented Metrics, Software Quality Metrics, Prediction Accuracy.

1. Giriş

Yazılım geliştirme teknolojilerindeki önemli ilerlemelere rağmen, yazılımlardaki hatalar üretilen yazılımların kalitesini ve güvenilirliğini büyük ölçüde etkilemektedir [1]. Bu hataların yazılım geliştirilirken erken aşamalarda tespit edilmesi önemlidir. Hatalar artık çoğu yazılım kalitesi özelliği arasında fiili endüstri standardı haline gelmiştir [2]. Bu nedenle, yazılım hatalarının kestirimi, yazılım mühendisliğinde 30 yıldan fazla bir süredir ilgi çeken konu olmuştur [3]. Test araçları, genellikle yazılım ürününün uygunluğunu doğrularken hataları tespit etmek için kullanılır. Bununla birlikte, bu tür araçlar genellikle tüm olası hataları tanımlamak için yeterli değildir, bu nedenle geliştirme sürecinde olabildiğinde çok yazılım hatasını tanımak için alternatif yöntemler gereklidir [4].

Yapılan çalışmada yazılım kalitesi ve geliştirilen eğitim yazılım projelerindeki hataların sayısı arasındaki ilişki belirlenip, hata kestirimi önerisinin sunulması amaçlanmıştır. Bu amaç doğrultusunda bu çalışmada, 50 açık

* Sorumlu yazar: begume@baskent.edu.tr. Yazarların ORCID Numarası: [1] 0000-0002-2170-2162, [2] 0000-0002-7372-0223

kaynak kodlu, nesne yönelimli ve orta ölçekli eğitim projesi olmak üzere belirli bir alandaki yazılım projeleri kullanılarak yazılım hata kestiriminin doğruluğu yapılmıştır. Çalışmada oluşturulan araştırma sorusu (AS) aşağıda verilmektedir. Araştırma sorusunun yanıtlanması için söz edilmiş olan 50 adet açık kaynak kodlu eğitim projesinin yazılım kalite ölçütleriyle hata kestirim doğruluğunun istatistiksel analizi yapılmıştır.

- (AS) Yazılım hata sayısının kestirimi için yazılım kalite ölçütlerinin kullanılması mümkün mü?

Geliştirilen yazılımların teslim edilmesinden sonra içerisinde hataların bulunması, bu hataların düzeltilmesi genelde uzun bir zaman almaktadır. Bu sebeple, teslim edilmesinden önce hata kestiriminin yapılması hem projelerin başarılı olmalarına hem de kalite, maliyet ve zaman açısından önemli derecede katkı sağlamasında yararlı olmaktadır. Yapılan çalışma katkı olarak bu noktada da fayda sağlamaktadır.

1.1. Ölçüt Tabanlı Yazılımlarda Hata Kestirim Görüşleri

Yazılım ölçütleri, geliştirilen koddaki bir hata noktasını tespit etmek için kullanılabilen, yazılım süreçlerinin [5] sonuçlarına göre bazı özellikleri veya ölçütleri temsil eden hesaplanabilir değerlerdir. Böylece, yazılım kalitesini iyileştirmek için kodun hangi bölümlerinin gözden geçirilmesi veya yeniden yazılması gerektiği hakkında bilgi sağlanabilir. Literatürde hata kestirimini belirlemek için çeşitli yazılım ölçütleri ve birkaç hata kestirimi görüşü bulunmaktadır. Süreç ölçütleri [6]; yazılım değişikliklerinin hatalara neden olduğunu, Önceki hata ölçütleri [7]; önceki hataların gelecekteki hataları tahmin ettiğini, Kaynak kod ölçütleri [8] ve Değişikliklerin entropisi [9]; karmaşık değişikliklerin hatalara neden olduğunu, Kod çalkantısı ölçütleri; kaynak kod ölçütlerinin kod çalkantısıyla ilişkisinin çıkarılmasına yardımcı olduğunu, Entropi ölçütleri ise; değişikliklerin entropisinin kaynak kod ölçütleri tarafından daha iyi tanımlandığını belirtir.

Tipik kaynak kodu ölçütleri arasında Chidamber ve Kemerer (CK) ölçütleri, kod satırı ölçütleri (LOC), Halstead ölçütleri ve McCabe karmaşıklık ölçütü [10] bulunmaktadır. CK ölçütleri [11], WMC (Weighted Methods Per Class - Sınıfın Ağırlıklı Metot Sayısı), DIT (Depth of Inheritance Tree - Kalıtım Ağacı Derinliği), NOC (Number of Children - Alt Sınıf Sayısı), RFC (Response for a Class - Sınıf Tarafından Tetiklenen Metot Sayısı), CBO (Coupling Between Objects - Sınıflar Arası Bağımlılık) ve LCOM (Lack of Cohesion of Methods - Uyum Eksikliği)'dur. LOC, ilk olarak 1960'larda tanıtılan ve kalite araştırmalarında yaygın olarak kullanılan en eski ölçütlerden biridir. LOC yöntemiyle, yazılmış olan kaynak kodunun satır sayısı üzerinden yazılımın büyüklüğüyle ilgili ölçümlerin yapılması mümkündür. Kullanımı kolaydır fakat, geliştirimi yapan kişilerin farklı kodlama yöntemlerini kullanması ya da tercih edilen programlama dillerinin farklılık göstermesi dezavantajları arasında yer almaktadır [12]. LOC, daha gelişmiş tasarımları ve gereksinimleri karşılayamamıştır, bu nedenle yazılım hata kestiriminde yararlı değildir [13]. Halstead ölçütleri [14], çeşitli programlarda meydana gelen operatörlerin ve işlenenlerin sayısına göre gerçek zamanlı anahtarlama sistemleri tanımlanır. McCabe karmaşıklık ölçütü [15], 1974'te Thomas McCabe tarafından önerilmiştir.

1.2. Yazılım Hata Kestirimi Konusunda İlgili Çalışmalar

Yazılım hata kestirimi alanında, birçok çalışma yapılmıştır. Yapılan çalışmalar aşağıda açıklanmaktadır:

Sarı ve Kalıpsız [16], önerdikleri modeli lojistik regresyon ile analiz etmek için Marco D'Ambros tarafından hazırlanan bir hata kestirim veri tabanını kullanmıştır. Ayrıca, modellerini bir organizasyon içinde geliştirilen yazılım kodları üzerinde test etmişler ve modelin hata kestiriminde belirli bir başarı seviyesine nasıl ulaştığı ile ilgili sonuçlarını yorumlamışlardır.

D'Ambros vd. [17], yazılım hatası kestirimi yapmak için sınıf düzeyinde veri setlerini tasarlamışlardır. Veri kümeleri, hatalar, değişiklikler ve beş yazılım sistemi için sürüm bilgilerinden oluşmaktadır. Verileri çevrimiçi olarak yayınlamışlardır ve tekrarlanabilirliğini sağlamışlardır. Dahil edilen sürümler için, her iki haftada bir ölçüt değerleri elde etmişlerdir ve her yazılım sınıfı için hata sayılarını güncellemişlerdir. Bu da hata raporlarının bir kestirim aracı olarak iyi bir yaklaşım olduğu iddiasına da yol açmıştır.

Gyimothy vd. [18], yazılım hatalarını belirlemek için daha önce Chidamber ve Kemerer (CK) tarafından sunulan nesne yönelimli ölçütler arasındaki ilişkileri incelemiştir. BUGZILLA hata izleme sistemini kullanarak yazılımdaki hataların miktarını elde ettikten sonra, C++ ile yazılmış açık kaynaklı projeler için hata eğilimi ile ölçütler arasında bir ilişkinin varlığını göstermişlerdir. Hata kestirimi için en etkili olabilecek ölçütleri araştırmışlar ve CBO ölçütünün daha üstün olduğunu, DIT ve NOC ölçütünün ise güvenilmez olduğunu gözlemlemişlerdir.

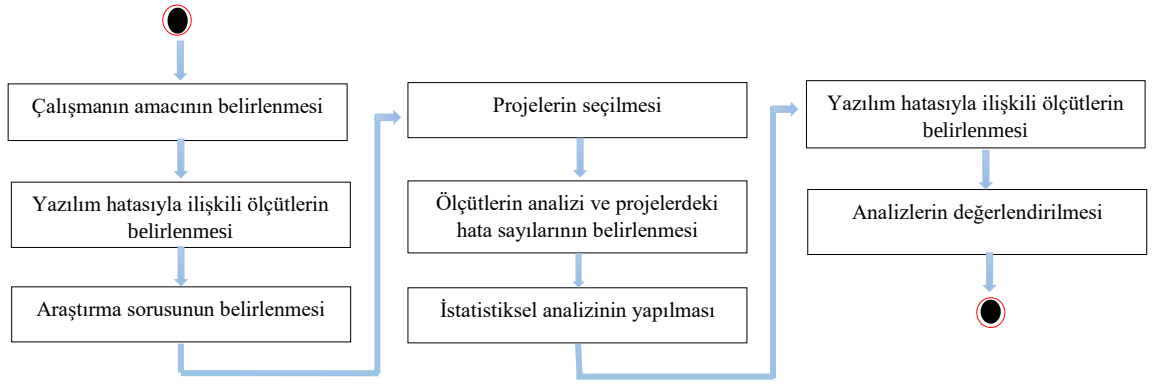
Okutan ve Yıldız [19], yazılımın ölçütleri ile hatalara yatkınlığı arasındaki ilişkileri belirlemek için Bayes ağlarını kullanmışlardır. Veri seti olarak ise 9 açık kaynak Promise deposunu dahil etmişlerdir ve RFC, LOC ve

LOCQ ölçütlerinin en etkili olduğunu gösteren modellerle sonuçlanmıştır. Diğer yandan, CBO, LCOM ve WMC ölçütlerinin daha az etkili olduğunu ve NOC ve DIT ölçütünün en güvenilmez olduğunu belirtmişlerdir.

Lamba vd. [20], hata kestirimi için yazılım modülleri kullanarak CK ve nesne yönelimli ölçütleri üzerinde çalışma yapmışlardır. Bağımlı ve bağımsız değişkenler arasındaki ilişkilerin çıkarımında regresyon analizi kullanmışlardır. Aynı zamanda R^2 değerinin karşılaştırılması için rastgele orman modelini kullanmışlardır. Sonuçların doğrulanmasında RStudio aracından faydalanmışlar ve elde edilen sonuçlarda basit doğrusal regresyon modelinin hata kestriminde rastgele orman modelinden daha iyi olduğu sonucuna ulaşmışlardır.

2. Teorik Metot

Bu çalışmada yazılım hata kestirimi konusunda daha önce yapılan çalışmalar incelenmiş, daha önce yapılan çalışmadan ilham alınarak [21] proje sayılarının arttırılması hedeflenmiş ve farklı alanda geliştirilmiş Java programlama dilinde yazılmış 50 orta ölçekli açık kaynak kodlu eğitim projesi analiz edilip, veri setinin hata kestirim doğruluğu analizinin yapılması amaçlanmıştır. Bu amaçla, Java programlama dilince yazılmış 50 orta ölçekli açık kaynak kodlu eğitim projesi veri seti olarak tercih edilmiştir. Java ile yazılmış yazılımların seçilmesinin nedenleri arasında güvenilirliği, platform bağımsızlığı ve hızlı proje geliştirme imkanı sağlaması sayılabilmektedir. Veri setinde eğitim projelerinin seçilmesinin nedeni, uzaktan eğitimin son zamanlarda ön planda olması olup bu nedenle de bu yazılım platformlarının daha kullanışlı hale getirilmesi için önemli bir katkı sağlayabileceğinin düşünülmesidir. Daha önce yapılmış olan çalışmadan farklı olarak proje alanı değiştirilmiştir hem de proje sayısı arttırılmıştır. Seçilen projeler farklı alanlardan tercih edilmesinden dolayı birbirinden tamamen farklıdır. Eski çalışmada oyun projeleri üzerine çalışılırken yapılan bu çalışmada eğitim projeleri üzerinde çalışılmıştır. Analiz sonuçlarına diğer çalışmadan farklı olarak birisi dışarıda çapraz doğrulama (LOOCV) yöntemi uygulanmıştır ve kestirim doğruluğu ölçütleri olan ortalama karesel hata (MSE), kök ortalama kare hata (RMSE) ve ortalama mutlak hata (MAE) değerleri hesaplanmıştır. Çalışmanın adımları Şekil 1'de verilmektedir. Yazılım hatasıyla ilişkili ölçütlerin belirlenmesinde literatür çalışmaları incelendiğinde CK ölçütleri ve yazılım hatasıyla ilgili çok çalışma yapıldığı ve CK ölçütlerinin hatayı arttırdığına dair sonuçlar elde ettikleri görülmüştür. Ölçütler belirlenirken bu referanslardan yola çıkılmıştır ve ilerleyen paragraflarda bu referansların detaylı anlatımı yer almaktadır.



Şekil 1. Çalışmanın adımları.

Çalışmada, birçok yazılım geliştiricisinin kullandığı SourceForge sitesinden açık kaynak kodlu projeler indirilmiştir. Açık kaynak kodlu yazılımlar, hazır yazılımlarla karşılaştırıldığında en temel özelliklerinden birisi, kaynak kodların kullanıcılara açık olması ve bu kaynak kodlarını ölçebilmek için birçok ölçütün literatürde yer almasıdır [22]. SourceForge sitesi hızlı, güvenli ve aynı proje üzerinde aynı anda birden çok kişinin çalıştığı kapsamlı raporlar içermektedir [23]. Yazılımların geliştirilmesindeki yaşam döngüsü sürecinde önemli aşamalardan birisinin kodlama sırasında kaynak kodlarının statik kod analizleri araçlarıyla incelenmesi olduğu ve bu incelemelerin yazılımlardaki güvenliği artıran yöntemlerden biri olduğu söylenmektedir [24]. Statik kodu incelemek ve seçilen ölçütlerin değerlerini elde etmek için "Understand" kod analiz aracı kullanılmıştır. Bu kod analiz aracı, ayrıntılı ölçüt raporları, bağımlılık analizleri, diyagramlar ve diğer grafiksel bilgiler dahil olmak üzere birden çok yazılım dilini destekleyen kaynak kodu hakkında bilgi üretmektedir [25].

Yazılımın kalitesinin değerlendirilmesi, kodların değerlendirilmesinde önceden tanımlanan kurallara bağlı

olarak bir bakış açısı sağlayabilmek için belirlenen ölçütlerin elde edilmesini gerektirir [26]. 6 adet CK ölçütünün hatayı arttırdığıyla ilişkili referanslar incelendiğinde bu altı ölçütün analizlerde yer alması tercih edilmiştir. WMC ölçütünün değerinin yüksek olmasının hataları arttırdığı ve hata kestirim modellerinin üzerinde önemli etkisinin olduğu [27], DIT ölçütünün yüksek olmasının hataları arttırdığı [28], RFC ölçütünün birçok metodun çağrılmasının sınıfın test edilmesini ve hataların ayıklanmasını zorlaştırdığı, CBO ölçütünün sınıfların arasındaki bağımlılık artarsa hatalarında arttığı, LCOM ölçütünün uyumun düşüklüğünün karmaşıklığı arttırdığı, bu sebeple hataların yapılması olasılığının yükseldiği söylenmektedir [29]. Seçilen bu 6 CK ölçütlerine ek olarak; FANIN ölçütü, McCabe'in karmaşıklık ölçütü VG, özel metotların sayısı (NPRM) ve genel metotların sayısı (NPM) olmak üzere hatalarla ilgili dört ölçüt daha dahil edilmiştir. "SpotBugs" programı, Eclipse geliştirme platformu için bir eklenti olarak kurulabilmekte, bir kullanıcı arayüzü içermektedir. Hataların sayısını bulmak amacıyla çalışmada kullanılmıştır. "SpotBugs" ile yalnızca Java ile yazılmış mantıksal hatalar tespit edilebilmektedir [30]. Seçilen projeler Eclipse platformuna yüklendikten sonra 50 projede bulunan mantıksal hatalar "SpotBugs" analiz aracı ile tespit edilmiştir. Koddaki sözdizimi (syntax) hataları bu analizin bir parçası olarak değerlendirilmemiştir.

Analiz aracı incelendiğinde tespit ettiği hataların mantıksal hata olduğu görülmüştür ve analiz aracını test etmek için mantıksal hata içeren basit bir modül geliştirilmiştir. Test sırasında sonsuz bir döngü yazıldığında, programın bu hatayı tespit ettiği ve türünü belirttiği görülmüştür. Her eğitim projesi için kod satır sayısı ve "SpotBugs" analiz aracı ile elde edilen hata sayısı Tablo 1'de listelenmiştir.

Tablo 1. Proje numarası, kod satır sayısı ve eğitim projelerinde tespit edilen hata sayıları.

Proje Numarası	KLOC	Hata Sayısı	Proje Numarası	KLOC	Hata Sayısı
1	11K	26	26	23K	**34**
2	11K	6	27	15K	**28**
3	11K	32	28	43K	**76**
4	12K	30	29	11K	**30**
5	13K	23	30	23K	**44**
6	13K	15	31	12K	**28**
7	14K	87	32	26K	**37**
8	16K	37	33	33K	**48**
9	17K	23	34	25K	**35**
10	18K	18	35	22K	**63**
11	23K	50	36	18K	**55**
12	24K	11	37	13K	**68**
13	25K	47	38	20K	**39**
14	25K	13	39	16K	**52**
15	30K	26	40	45K	**59**
16	31K	8	41	84K	**12**
17	36K	7	42	63K	**25**
18	39K	6	43	90K	**57**
19	46K	23	44	50K	**33**
20	49K	17	45	13K	**64**
21	60K	25	46	15K	**16**
22	75K	12	47	12K	**40**
23	76K	57	48	70K	**67**
24	61K	23	49	22K	**95**
25	**76K**	**21**	**50**	**12K**	**46**

3. Deneysel Çalışmalar

Araştırma sorusunun (AS) cevabını bulabilmek için 50 proje üzerinde hata kestirimiyle ilişkili ölçütlerin analizi yapılarak sonuçları incelenmiştir. Birisi dışarıda çapraz doğrulama (LOOCV) yöntemiyle, n parçaya ayrılan veri setinden her tekrarlamada biri çıkarılır ve geriye kalanlarla eğitilme aşaması yapılır [31]. Bu aşamalar tüm parçalar için tekrar edilir. LOOCV gerçekleştirilirken önce ilk proje kaldırılmıştır ve kalan 49 proje için Adımsal (Stepwise) Doğrusal Regresyon analizi uygulanmıştır. Analiz sonucu çıkarılan denklem ilk proje için yerine konulduğunda kestirilen hata sayısının hesaplanması yapılmıştır. Bu işlem 50. projeye kadar aynı olarak devam ettirilmiştir ve bağıl hata (MRE) hesaplanmıştır. MRE hesaplamaları yapılırken Pred(0,25) ve Pred(0,30) değerlerinin sonucu elde edilmiştir. Uygulanmış olan regresyon sonucu çıkan denklemin ilk projede yerine konulmasıyla LOOCV'la kestirilmiş olan hata sayısı hesaplaması yapılmıştır. İlk projenin çıkarılmasıyla oluşan denklem Denklem 1'de verilmektedir. İlk proje çıkarıldığında hesaplanan analiz ise Tablo 2'de verilmektedir.

$$\text{Hata Sayısı} = (NOC * 3{,}200 + VG * 52{,}300 + NPM * -21{,}300 + WMC * 10{,}260 + NPRM * -10{,}450 + CBO * -9{,}170 + LCOM * 0{,}458 + DIT * 10{,}700 + FANIN * 12{,}800) - 42{,}839 \quad (1)$$

Projelerdeki hata sayısı ile ortalama ölçüt değerleri arasındaki ilişki Minitab istatistiksel yazılımı ile hesaplanmıştır. Ölçütler ile toplam hata sayısı arasındaki ilişki, iki değişken arasındaki ilişkiyi ve önemini inceleyen Pearson Korelasyonu kullanılarak hesaplanmıştır [32]. Hata sayısı ile her proje için ölçütler arasında bir model oluşturmak için Adımsal (Stepwise) Doğrusal Regresyon analizi uygulanmıştır. Yaygın olarak kullanılan bir değişken seçim tekniği olan Adımsal (Stepwise) Doğrusal Regresyon, her adımda değişkenler ekleyerek veya çıkararak tekrarlı bir regresyon modeli dizisi oluşturmaktadır [33]. Bir değişken ekleme veya çıkarma kriteri tipik olarak kısmi F testi ile ifade edilmektedir [32]. Doğrusal regresyon denklemlerinde, R^2 değeri tahmin edilen değişkenin varyansını açıklayarak değişkenleri tahmin etme oranını sağlamaktadır [34]. Yüksek R^2 değeri, bağımsız değişkenlerden oluşan doğrusal denklemin bağımlı değişkeni iyi tahmin edebileceğini göstermektedir. Bu sebeple yüksek R^2 değerinin olduğu adımlar seçilerek LOOCV işlemi gerçekleştirilmiştir.

Tablo 2. İlk eğitim projesinin çıkarılması ve LOOCV kullanılarak elde edilen adımsal doğrusal regresyonun sonucu.

Adım	7	8	9
Sabit	-9,039	-26,046	**-42,839**
NOC	3,230	3,200	**3,200**
VG	51,200	52,000	**52,300**
NPM	-21,000	-21,200	**-21,300**
WMC	9,690	10,010	**10,260**
NPRM	-11,370	-10,420	**-10,450**
CBO	-8,530	-8,450	**-9,170**
LCOM	0,585	0,546	**0,458**
DIT		8,800	**10,700**
FANIN			**12,800**
S	6,900	6,270	**5,38**
R-Rq	91,400	93,060	**95,020**
R-Rq(adj)	**89,940**	**91,680**	**93,870**

Tablo 3'te verilen her proje için ortalama ölçüt değerleri, toplam ölçüt değerlerinin yazılım sınıfı sayısına bölünmesiyle hesaplanmıştır.

Tablo 3. Eğitim projelerinde kullanılan ölçütlerin hesaplanmış ortalama değerleri.

Proje Numarası	WMC	DIT	RFC	NOC	CBO	LCOM	FANIN	VG	NPRM	NPM
1	3,640	1,610	0,230	4,670	2,780	21,180	1,100	1,160	0,430	**2,620**
2	5,310	1,620	0,020	5,350	2,300	24,420	1,070	1,190	0,640	**4,490**
3	6,910	1,690	0,010	6,910	2,850	41,980	1,210	1,400	1,170	**5,020**
4	3,240	1,940	0,030	3,500	1,790	12,960	1,160	1,050	1,140	**2,090**
5	4,960	1,830	0,350	8,390	4,320	39,000	1,190	1,340	0,570	**4,080**
6	3,810	1,700	0,390	6,150	3,020	13,020	1,160	1,160	0,360	**3,410**
7	11,700	2,000	0,240	29,690	2,880	41,360	1,170	1,320	0,340	**8,810**
8	3,600	1,930	0,310	9,290	2,480	25,750	1,180	1,110	0,270	**3,260**
9	4,970	1,790	0,160	6,390	2,520	22,960	1,190	1,170	0,880	**3,630**
10	10,540	1,880	0,280	14,440	1,500	34,160	1,140	1,240	0,830	**8,810**
11	7,300	2,170	0,530	29,390	6,600	41,580	1,120	1,340	0,570	**6,480**
12	4,910	1,530	0,100	7,260	3,350	20,510	1,310	1,280	0,480	**4,240**
13	5,560	2,050	0,250	11,470	4,380	29,260	1,180	1,220	1,190	**3,590**
14	9,460	1,690	0,290	14,810	4,230	55,520	1,270	1,380	1,060	**7,830**
15	4,930	1,630	0,100	5,760	3,170	30,020	1,230	1,450	0,790	**3,950**
16	5,950	2,310	0,560	9,040	3,500	31,510	1,390	1,080	0,820	**4,960**
17	10,200	1,940	0,370	15,900	2,210	45,510	1,240	1,170	0,740	**8,850**
18	9,790	1,130	0,010	9,790	3,020	53,600	1,290	1,100	3,520	**6,200**
19	4,900	1,620	0,040	4,940	1,980	25,380	1,340	1,460	0,170	**4,670**
20	8,390	1,680	0,160	10,310	3,440	43,460	1,160	1,560	1,260	**6,780**
21	6,450	1,430	0,410	19,870	3,780	45,670	1,670	1,480	3,670	**5,670**
22	10,540	1,650	0,220	20,670	4,450	27,840	1,340	1,140	0,450	**8,540**
23	9,210	2,560	0,300	6,430	3,180	53,320	1,200	1,850	2,180	**6,210**
24	6,930	1,450	0,200	7,880	2,340	50,210	1,400	1,980	2,450	**5,670**
25	5,540	1,340	0,340	5,430	2,540	48,760	1,340	1,750	2,430	**4,780**
26	4,450	1,430	0,230	9,340	3,450	50,340	1,980	1,490	2,650	**4,440**
27	10,320	1,250	0,450	4,330	3,670	55,320	1,130	1,780	2,540	**6,450**
28	11,320	1,340	0,820	13,120	4,330	64,210	1,090	1,200	1,450	**5,330**
29	3,450	1,370	0,300	14,220	4,760	34,500	1,900	1,430	0,560	**4,200**
30	12,330	1,760	0,440	13,400	3,560	47,300	1,540	1,320	0,780	**8,400**
31	10,300	1,540	0,320	11,430	4,760	48,330	1,670	1,430	0,300	**7,410**
32	5,320	1,750	0,730	16,480	2,660	59,300	1,540	1,780	0,340	**7,320**
33	9,340	1,650	0,320	5,370	4,560	44,220	1,650	1,870	0,320	**6,450**
34	6,430	2,500	0,220	5,300	2,670	65,940	1,340	1,820	0,430	**6,700**
35	7,300	1,820	0,460	10,200	3,550	40,210	1,890	1,300	0,210	**4,540**
36	5,880	1,960	0,320	6,540	2,980	57,320	1,400	1,930	0,340	**5,540**
37	11,210	1,080	0,360	14,330	3,510	56,300	1,430	1,780	2,390	**6,400**
38	9,300	1,870	0,840	6,320	5,620	46,760	1,820	1,320	2,600	**4,210**
39	4,530	1,640	0,470	13,250	4,760	61,320	1,540	1,740	2,480	**3,560**
40	10,440	1,560	0,980	9,310	3,670	43,200	1,430	1,830	0,210	**7,230**
41	9,700	1,230	0,010	7,250	3,090	52,110	1,760	1,400	3,510	**6,320**
42	11,750	1,160	0,810	11,330	4,350	44,560	1,300	1,100	0,650	**6,880**
43	9,220	1,090	0,230	11,260	4,600	35,420	1,790	1,040	0,430	**4,200**
44	10,270	1,550	0,540	15,320	3,450	38,380	1,800	1,340	2,540	**6,890**
45	10,430	1,670	0,340	12,420	2,970	65,600	1,930	1,450	0,320	**7,350**
46	5,110	1,570	0,040	5,350	2,360	24,300	1,090	1,240	0,640	**4,210**
47	6,430	2,430	0,960	12,340	2,550	60,530	1,830	1,430	0,490	**6,880**
48	5,640	1,780	0,300	25,380	2,500	58,650	1,410	1,760	0,500	**7,460**
49	10,320	1,900	0,460	17,700	4,590	36,590	1,500	1,420	2,400	**3,560**
50	**6,300**	**1,340**	**0,320**	**19,350**	**2,520**	**39,200**	**1,630**	**1,800**	**0,510**	**7,400**

Analizlerde projelerin çıkarılmasıyla elde edilen denklemler incelendiğinde RFC ölçütünün ilk projeden başlanarak 50. projede dahil çıkarılması aşamasında hiç bir denklem içinde yer almadığı gözlemlenmiştir. Bu sonuç aslında RFC ölçütünün hata kestiriminde hiç bir etkisinin olmadığını bize göstermektedir. Onun aksine diğer 9 ölçütün incelenen tüm denklemlerde yer aldığı görülmüştür. Tablo 3'teki bu ortalama değerler listesi ile 50 proje için LOOCV uygulanarak MRE değerleri, Pred(0,25) ve Pred(0,30) değerleri hesaplanmıştır. MRE, kestirim modellerini değerlendirmek için yaygın olarak kullanılan bir ölçüdür. Pred(x) tahmin seviyesi, literatürde belirli bir doğruluk seviyesi için yapılan gözlemlerin oranı olarak sıklıkla kullanılmaktadır [35]. MRE ve Pred(x) denklemleri sırasıyla Denklem 2 ve Denklem 3'te verilmektedir.

$$MRE = \frac{|Egerçek - Etahmin|}{Egerçek} \tag{2}$$

$$Pred(x) = \frac{k}{N} \tag{3}$$

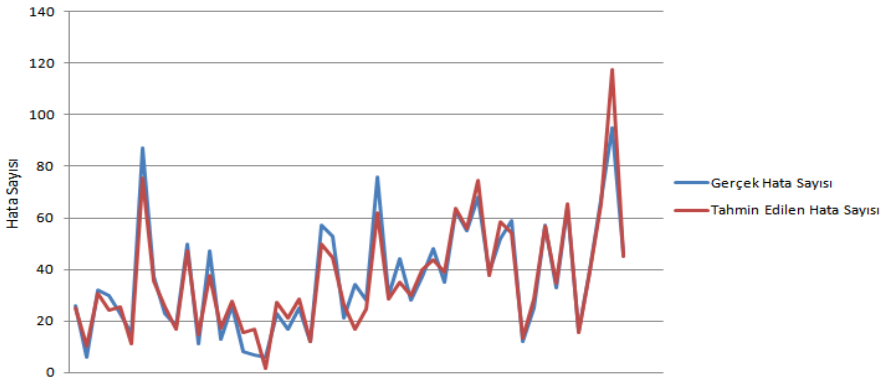
Burada k, MRE değeri 1'den küçük olan projelerin sayısını belirtmektedir. N, veri kümesindeki toplam proje sayısını temsil etmektedir. Kestirim modelleri oluştururken en sık kullanılan 0,25 [35] ve 0,30 Pred [36] değerleri dikkate alınmıştır. LOOCV ile bu hesaplamaların sonuçları, 0,25'ten küçük MRE değerlerinin koyu olarak vurgulandığı Tablo 4'te verilmektedir. Verilen Tablo 4'te LOOCV ile 50 eğitim projesinin yer aldığı görülmektedir.

Tablo 4. LOOCV ile 50 eğitim projesi için gerçek hata sayısı, kestirilen hata sayısı ve MRE değerleri.

Proje Numarası	Hata Sayısı	LOOCV ile Hata Sayısı	MRE	Proje Numarası	Hata Sayısı	LOOCV ile Hata Sayısı	MRE
1	26	25,17	**0,03**	26	34	16,74	**0,51**
2	6	10,31	0,72	27	28	24,82	**0,11**
3	32	30,89	**0,03**	28	76	62,13	**0,18**
4	30	24,19	**0,19**	29	30	28,51	**0,05**
5	23	25,55	**0,11**	30	44	35,16	**0,20**
6	15	11,25	0,26	31	28	29,96	**0,07**
7	87	75,30	**0,13**	32	37	39,70	**0,07**
8	37	35,36	**0,04**	33	48	43,66	**0,09**
9	23	25,42	**0,11**	34	35	38,96	**0,11**
10	18	16,70	**0,07**	35	63	63,54	**0,01**
11	50	47,09	**0,06**	36	55	56,03	**0,02**
12	11	14,55	0,32	37	68	74,74	**0,10**
13	47	37,68	**0,20**	38	39	37,62	**0,04**
14	13	17,09	0,31	39	52	58,60	**0,13**
15	26	27,66	**0,06**	40	59	53,98	**0,09**
16	8	15,50	0,94	41	12	13,43	**0,12**
17	7	16,82	1,40	42	25	28,57	**0,14**
18	6	1,54	0,74	43	57	56,59	**0,01**
19	23	27,08	**0,18**	44	33	34,58	**0,05**
20	17	21,09	**0,24**	45	64	65,62	**0,03**
21	25	28,38	**0,14**	46	16	15,35	**0,04**
22	12	11,99	**0,00**	47	40	39,21	**0,02**
23	57	49,80	**0,13**	48	67	65,26	**0,03**
24	23	44,45	**0,16**	49	95	117,53	**0,24**
25	21	26,42	0,26	50	46	44,97	**0,02**

LOOCV ile Eğitim Projelerinin Kestirim Doğruluğunun Değerleri
Pred (0,25) = 0,82 Pred (0,30) = 0,86

Eğitim projelerinde Pred(0,25) hesaplamak için, 0,25'ten küçük olan MRE sayısının toplam proje sayısına bölünmesiyle 0,82 elde edilmiştir. Pred(0,30) hesaplanırken ise 0,30'dan küçük olan MRE sayısının toplam proje sayısına bölünmesiyle 0,86 değeri elde edilmiştir. Bu araştırmada kullanılan kestirim denklemi 50 proje ile sınırlıdır. Gelecekte veri setine daha fazla yazılım projesi ekleyerek, karşılaştırılan modelin kestirim doğruluğunun arttırılması hedeflenmektedir. Şekil 2'de gerçek hata sayıları ve kestirilen hata sayılarının grafikselleştirildiği görsel bulunmaktadır. Bakıldığında gerçek hata sayıları ile kestirilen hata sayılarının birbirine çok yakın olduğu görülmektedir.



Şekil 2. Gerçek hata ve kestirilen hata sayılarının çizelgesi.

Bu çalışmada, 50 adet açık kaynak kodlu, nesne tabanlı eğitim projesi üzerinde çalışma yapılmıştır. Kestirim denklemi, literatürdeki önceki değerler tarafından kabul edilebilir bir doğruluk eşiği olarak desteklenen [35] Pred(0,25)=0,82 ve Pred(0,30)=0,86 oranında doğru tahminler sağlamıştır [36]. İlgili çalışmalarda kullanılan hazır veri setleri incelendiğinde; hata kestirimi ile ilgili olabilecek ölçütlerin bazılarının bu veri setlerinin içinde olmamasından dolayı açık kaynak kodlu projeler üzerinden gidilerek proje analizleri gerçekleştirilmiştir. Çalışmada ayrıca kestirim doğruluğu için sıklıkla kullanılan ölçütler arasında yer alan Ortalama Karesel Hata (Mean Squared Error - MSE), Kök Ortalama Kare Hata (Root Mean Squared Error - RMSE) ve Ortalama Mutlak Hata (Mean Absolute Error - MAE) değerlerinin sonuçları da incelenmiştir. MSE kestirim doğruluklarında en sık tercih edilen ölçütlerden biridir ve MSE değeri ne kadar düşükse hata miktarının da o kadar az olduğu söylenmektedir. Ortalama RMSE değerinin istatistiksel olarak hata diye de değerlendirildiği ve bu değerin sıfıra yakın olmasının, kestirilen değerin gerçeğe olan yakınlığını gösterdiği söylenmektedir [37]. MAE değeri, 0 ile sonsuz arasında değişebilmektedir, negatif yönelimler yani düşük değere sahip MAE değerinin daha iyi söylenmektedir. Bakıldığında MSE, RMSE ve MAE değerlerinin düşük olmasının doğruluğu daha iyi gösterdiği söylenmektedir. Bu kestirim doğruluğu ölçütlerinin denklem formülleri Tablo 5'te verilmektedir.

Tablo 5. Kestirim doğruluğu ölçütleri ve denklemleri.

Kestirim Doğruluğu Ölçütleri	Denklem
Ortalama Karesel Hata (MSE)	$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Etahmin - Egerçek)^2$$
Kök Ortalama Kare Hata (RMSE)	$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(Etahmin - Egerçek)^2}{n}}$$
Ortalama Mutlak Hata (MAE)	$$MAE = \frac{1}{n}\sum_{1}^{n}\lvert Etahmin - Egerçek\rvert$$

Tablo 5'te verilen denklemler üzerinden gerekli hesaplamalar 50 proje için yapıldığında, elde edilen sonuçlar ise Tablo 6'da verilmektedir.

Tablo 6. LOOCV ile 50 eğitim projesi için MSE, RMSE ve MAE değerleri.

Proje Numarası	**MSE (Etahmin-Egerçek)^2**	**RMSE (Etahmin-Egerçek)^2/50**	**MAE (|Etahmin- Egerçek|)**
1	0,68	0,01	**0,83**
2	18,58	0,37	**4,31**
3	1,23	0,02	**1,11**
4	33,74	0,67	**5,81**
5	6,49	0,13	**2,55**
6	14,07	0,28	**3,75**
7	137,00	2,74	**11,70**
8	2,70	0,05	**1,64**
9	5,85	0,12	**2,42**
10	1,69	0,03	**1,30**
11	8,48	0,17	**2,91**
12	12,63	0,25	**3,55**
13	86,96	1,74	**9,33**
14	16,71	0,33	**4,09**
15	2,75	0,06	**1,66**
16	56,24	1,12	**7,50**
17	96,52	1,93	**9,82**
18	19,88	0,40	**4,46**
19	16,67	0,33	**4,08**
20	16,70	0,33	**4,09**
21	11,43	0,23	**3,38**
22	0,00	0,00	**0,01**
23	51,82	1,04	**7,20**
24	73,11	1,46	**8,55**
25	29,42	0,59	**5,42**
26	298,01	5,96	**17,26**
27	10,11	0,20	**3,18**
28	192,25	3,85	**13,87**
29	2,23	0,04	**1,49**
30	78,15	1,56	**8,84**
31	3,85	0,08	**1,96**
32	7,30	0,15	**2,70**
33	18,80	0,38	**4,34**
34	15,68	0,31	**3,96**
35	0,29	0,01	**0,54**
36	1,05	0,02	**1,03**
37	45,39	0,91	**6,74**
38	1,91	0,04	**1,38**
39	43,58	0,87	**6,60**
40	25,17	0,50	**5,02**
41	2,03	0,04	**1,43**
42	12,76	0,26	**3,57**
43	0,17	0,00	**0,41**
44	2,49	0,05	**1,58**
45	2,62	0,05	**1,62**
46	0,43	0,01	**0,65**
47	0,63	0,01	**0,79**
48	3,02	0,06	**1,74**
49	507,75	10,15	**22,54**
50	1,07	0,02	**1,03**
	MSE = 39,95	**RMSE = 6,32**	**MAE = 4,51**

Değerler hesaplanırken Tablo 5'te verilen denklemlerden yararlanılmıştır. MSE hesaplanırken Tablo 6'da verilen (Etahmin-Egerçek)^2 değerlerinin toplamı proje sayısı olan 50'ye bölündüğünde 39,95 değeri, RMSE hesaplanırken (Etahmin-Egerçek)^2/50 değerlerinin toplamının karekökü alındığında 6,32 değeri ve MAE hesaplanırken |Etahmin-Egerçek| değerleri toplamı proje sayısı 50'ye bölündüğünde 4,51 değeri bulunmuştur.

4. Sonuçlar ve Tartışmalar

Yapılan çalışmada yazılım projelerinde olan hata sayıları ve yazılım ölçütleri arasındaki ilişkinin belirlenip, proje setine birisi dışarıda çapraz doğrulama (LOOCV) yapılarak kestirim doğruluğu değeri sonucunun elde edilmesi amaçlanmıştır. İncelenen projelerde yazılım ölçütlerinin analizi yapılırken "Understand" kod analizi aracı ve yazılımlardaki hataların tespit edilmesindeyse "SpotBugs" kod analizi aracı kullanılmıştır. 50 eğitim projesi üzerinde birisi dışarıda çapraz doğrulama (LOOCV) yöntemi kullanılarak MRE değerinin hesaplanması yapılmıştır. Hesaplamanın yapılmasından sonra Pred(0,25) ve Pred(0,30) için sonuç değerleri sırayla 0,82 ve 0,86 olarak çıkmıştır. Daha önceden yapılan oyun projesi üzerindeki çalışmada [21] Pred(0,25) ve Pred(0,30) için sonuç değerlerinin ise sırayla 0,75 ve 0,80 olarak çıktığı bilinmektedir. Bu çalışmada ek olarak LOOCV yapılmış ve farklı alanda projeler seçilerek proje sayısı arttırılmıştır. Eğitim projelerinde Pred(0,30) değerinin az farkla yüksek olduğu görülmüştür. Kestirim denkleminin, literatürde yer alan önceki değerler tarafından kabul edilebilir bir doğruluk eşik değerini sağladığı görülmüştür.

Vashisht vd. [38] yaptıkları çalışmanın amacı, yazılım geliştirme projelerindeki hataları kestirebilmek için bir model geliştirilmesidir. Veri seti olarak bir yazılım firmasından alınan 50 gerçek projeyi kullanmışlardır. 40 projeyi modelin eğitilmesinde kalan 10 proje ise sinir ağı modelin doğruluğunun onaylanmasında kullanmışlardır. Modellerinin performansını değerlendirmek için araştırmacılar tarafından sıklıkla kullanılan ölçütler olan MSE, RMSE ve MAE ölçütlerini kullanmışlardır. Sinir ağı modellerinin kestirimiyle gerçek hataların karşılaştırmasını yaptıklarında MSE, RMSE ve MAE değerlerinin, kestirim sonuçlarının gerçek sonuçlarla uyumlu olduğunu göstermektedir şeklinde yorumlamaları yer almaktadır. Test aşamasındaki sonuçlarda MSE=57,82, RMSE=7,60, MAE=2,93 değerlerini almışlardır. Vashisht vd. yaptıkları çalışmanın sonuç değerleriyle kıyaslandığında MSE ve RMSE değerlerinin yaptığımız çalışmada daha düşük sonuçlar verdiği, MAE değerinin ise çok az bir farkla yüksek olduğu görülmüştür. MSE, RMSE ve MAE değerlerinin, kestirim sonuçlarının gerçek sonuçlarla uyumlu olduğunu gösterdiği şeklinde yorumlamalarına bakıldığında ve bu değerlerin düşük olmasının doğruluğu daha iyi gösterdiği bilgisine bakıldığında elde ettiğimiz değerlerin MSE ve RMSE değerleri için daha düşük çıktığı yorumu yapılabilmektedir. Bu da kestirim sonuçlarının gerçek sonuçlarla uyumlu olduğunu göstermektedir.

Sharaf vd. [39] yaptığı çalışmada hisse senedi fiyatının kestirimi için uzun kısa süreli bellek, evrişimsel sinir ağı, destek vektör makinesi, doğrusal regresyon, lojistik regresyon gibi öğrenme modelleri kullanılarak tahmin yapabilmek için bir çerçeve önermesi yapılmaktadır. İlk bölümde destek vektör regresyonu, doğrusal regresyon, lojistik regresyon, rastgele orman ve k en yakın komşu gibi sinir ağlarından bağımsız normal makine öğrenimi modellerini kullanarak hisse senedi fiyat tahmin sürecinin tartışıldığı bölüm yer almaktadır. Gerçek ve kestirilen hatalar arasındaki karşılaştırmaları yapmışlardır. 30 günlük bir kestirim için sonuçları vermişlerdir, makine öğrenmesi modelleri kullanarak, tartışılan değerlendirme ölçütlerinin eğitilmesini ve test edilmesi aşamalarını gerçekleştirmişlerdir. Gün sayılarını 100 ve 1000 içinde tekrar ederek tablo olarak vermişlerdir. Sonuçlara bakıldığında öğrenme algoritmalarından k en yakın komşu makine öğrenimi modelinin RMSE değerinin 26,38, MSE değerinin 695,75, MAE değerinin ise 14,9 ile en iyi sonuç olarak geldiği görülmüştür. Aynı süreci sinir ağları modelleri için denediklerinde ise evrişimsel sinir ağı modelinin RMSE, MSE ve MAE için en iyi sonuçları ürettiğini gözlemlemişlerdir. Sharaf vd. yaptıkları çalışmanın sonuç değerleriyle yine kıyaslandığında üç ölçüt değerinin yaptığımız çalışmada daha düşük sonuçlar verdiği görülmüştür.

Gyimothy vd. [18], CBO ve LCOM ölçütlerinin hata kestirimi için en iyi olduğunu, NOC ölçütünün ise güvenilmez olduğunu iddia etmişlerdir. Bu çalışmada elde edilen bulgular için, CK ölçütlerinden olan RFC ölçütü dışındaki diğer tüm CK ölçütlerinin hata kestirimi ile ilişkisi olduğu gözlemlenmiştir. Kullanılan ek dört ölçütün de (FANIN, VG, NPRM ve NPM ölçütlerinin) hata kestiriminde etkili olduğu gözlemlenmiştir. Gyimothy vd. NOC ölçütünün hataların kestirimi ile ilişkisinin güvenilir olmadığını belirtirken, bu çalışmada NOC ölçütünün hata kestiriminde önemli olduğu görülmüştür. Yapılan çalışmada Gyimothy vd. yaptığı çalışmadan farklı bulgular gözlemlenmesinin nedeninin seçilmiş olan programlama dilleri farklılığından ve kullanılan hata takip sistemlerinin farklılığından kaynaklı olduğu düşünülmektedir.

Analizler incelendiğinde RFC ölçütünün ilk projeden başlanarak 50. projede dahil çıkarılması aşamasında hiç bir denklem içinde yer almadığı gözlemlenmiştir ve bu sonuç RFC ölçütünün hata kestiriminde hiç bir etkisinin olmadığını bize göstermiştir. Çalışmada ayrıca kestirim doğruluğu için sıklıkla kullanılan ölçütler arasında yer alan Ortalama Karesel Hata (Mean Squared Error - MSE), Kök Ortalama Kare Hata (Root Mean Squared Error - RMSE) ve Ortalama Mutlak Hata (Mean Absolute Error - MAE) değerlerinin sonuçları da incelenmiştir. Vashisht vd. [38] yaptıkları çalışmayla kıyaslandığında MSE ve RMSE değerlerinin yaptığımız çalışmada daha düşük sonuçlar verdiği bu değerlerin düşük olmasının doğruluğu daha iyi gösterdiği bilgisinden yola çıkılarak kestirim sonuçlarının gerçek sonuçlarla uyumlu olduğu görülmüştür. Sharaf vd. [39] yaptığı çalışmada ise üç ölçütün yaptığımız çalışmada daha düşük sonuçlar verdiği gözlemlenmiştir.

Yazılımların erken safhada hata kestirimlerinin yapılması projelerin daha başarılı olmasını, yazılımın kalitesini, maliyetini ve projelerin daha kısa sürede teslim edilmesini sağlamaktadır. Çalışma aynı zamanda yazılımlardaki hata sayılarının kestirebilmesi için yazılım kalite ölçütlerinden faydalanmanın da önemini bir kez daha göstermiştir.

Kaynaklar

[1] Raees M. Study of software quality improvement through reliability metrics models and root cause analysis program. International Journal of Computer Engineering and Information Technology 2020; 12(6): 42-47.

[2] Jones C. Quantifying Software: Global and Industry Perspectives. CRC Press, 2017.

[3] Fenton N, Neil M. A critique of software defect prediction models. IEEE Transactions on Software Engineering 1999; 25(5), 675-689.

[4] Delsing J. IoT Automation-Arrowhead Framework. CRC Press: Taylor & Francis Group, 2017.

[5] D'Ambros M, Lanza M, Robbes R. Evaluating defect prediction approaches: A benchmark and an extensive comparison. Emprical Software Engineering 2012; 17: 531-577.

[6] Moser R, Pedrycz W, Succi G. A comparative analysis of the efficiency of change metrics and static code attributes for defect prediction. In: ICSE '08: Proceedings of the 30th international Conference on Software Engineering; 10-18 May, 2008; Leipzig-Germany. 181-190.

[7] Kim S, Zimmermann T, Whiteas E, Zeller A. Predicting faulties from cached history. Proceedings of 29th International Conference on Software Engineering (ICSE'07); 20-26 May, 2007; Minneapolis, MN, USA. 489-498.

[8] Basili V.R, Briand L.C, Melo W.R. Validation of object-oriented design metric as quality indicators. IEEE Transactions on Software Engineering 1996; 22(10), 751- 761.

[9] Hassan A.E. Predicting faults using the complexity of code changes. International Conference on Software Engineering IEEE Computer Society; 16-24 May, 2009; Vancouver, BC, Canada. 78- 88.

[10] Nuñez-Varela A.S, Pérez-Gonzalez H.G, Martínez-Perez F.E, Soubervielle-Montalvo C. Source code metrics: a systematic mapping study. Journal of Systems and Software 2017; 128: 164-197.

[11] Chidamber S, Kemerer C. A metrics suite for object-oriented design. IEEE Transactions on Software Engineering 1994; 20(6): 476-493.

[12] Erçelebi Ayyıldız T, Koçyiğit A. Correlations between problem and solution domain measures of open source software. Journal of the Faculty of Engineering and Architecture of Gazi University 2017; 32(3): 887– 900.

[13] Catolino G, Palomba F, De Lucia A, Ferrucci F, Zaidman A. Enhancing change prediction models using developer-related factors. Journal of Systems and Software 2018; 143: 14-28.

[14] Bailey C.T, Dingee W.L. A software study using halstead metrics. ACM Sigmetrics (1981 ACM Workshop/Symp. Measurement and Evaluation of Software Quality 1981; 10: 189-197.

[15] McCabe T. A complexity measure. IEEE Transactions on Software Engineering 1976; 2(4): 308-32.

[16] Sarı Ö, Kalıpsız O. Using Data Analysis Methods for Software Bug Prediction, UYMS, https://pdfs.semanticscholar.org/f6f5/23ec28440cfe66096b4d4d1d62ca8f018db9.pdf. Yayın Tarihi 2014. Erişim Tarihi Temmuz 29, 2021.

[17] DAmbros M, Lanza M, Robbes R. On the relationship between change coupling and software defects. 16th Working Conference on Reverse Engineering. 13-16 October 2009; Lille-France. 135–144.

[18] Gyimóthy T, Ferenc R, Siket I. Empirical validation of object-oriented metrics on open source software for fault prediction. IEEE Transactions on Software Engineering 2005; 31(10): 897-910.

[19] Okutan A, Yıldız O.T. Software defect prediction using bayesian networks. Empirical Software Engineering 2014. 19(1): 154-181.

[20] Lamba T, Kumar D, Mishra A.K. Comparative study of bug prediction techniques on software metrics, International Conference on Computing for Sustainable Global Development; 01-03 March, 2017; New Delhi-(INDIA). 207-211.

[21] Erçelebi Ayyıldız T, Erkal B. The effect of object-oriented metrics on software bug prediction. Journal of Information Systems and Management Research 2019; 1(1): 1-8.

[22] Yılmaz N, Tarhan A. A two-dimensional method for evaluating maintainability and reliability of open source software. Journal of the Faculty of Engineering and Architecture of Gazi University 2019; 34(4): 1807-1829.

[23] Perez-Riverol Y., Gatto L., Wang R., Sachsenberg T., Uszkoreit J., Veiga Leprevost F. Leprevost, Fufezan C., Ternent T, et al. Ten simple rules for taking advantage of git and github. PLos Computational Biology 2016; 12(7).

[24] Durmuş G, Soğukpınar İ. A novel approach for analyzing buffer overflow vulnerabilities in binary executables by using machine learning techniques. Journal of the Faculty of Engineering and Architecture of Gazi University 2019; 34(4): 1695-1704.
[25] Understand SCI Tool, https://www.scitools.com/features. Erişim Tarihi: Temmuz 25, 2021.
[26] Gezici B, Tarhan A, Chouseinoglou O. Complexity, size and internal quality in the evolution of mobile applications: An exploratory study. Journal of the Faculty of Engineering and Architecture of Gazi University 2019; 34(3): 1483-1500.
[27] Choudhary G.R, Kumar S., Kumar K, Mishra A, Catal C. Empirical analysis of change metrics for software fault prediction. Computers & Electrical Engineering 2018; 67: 15-24.
[28] Breesam K.M. Metrics for object-oriented design focusing on class inheritance metrics. 2nd International Conference on Dependability of Computer Systems (DepCoS-RELCOMEX '07). 14-16 June, 2007; Szklarska-Poland.
[29] Subramanyam R, Krishnan M.S. Empirical analysis of ck metrics for object-oriented design complexity: Implications for software defects. IEEE Transactions on software engineering 2003; 29(4): 297-310.
[30] Spotbugs. http://spotbugs.github.io. Erişim Tarihi Temmuz 26, 2021.
[31] Wong T. Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation. Pattern Recognition 2015; 48(9): 2839-2846.
[32] Montgomery D.C, Runger G.C. Applied Statistics and Probability for Engineers. 6th Edition. John Willey & Sons, 2014.
[33] Silhavy R, Silhavy P, Prokopova Z. Analysis and selection of a regression model for the use case points method using a stepwise approach. Journal of Systems and Software 2017; 125: 1-14.
[34] Tabachnick B.G, Fidell L.S. Using Multivariate Statistics. 2nd Edition. HarperCollins, New York, 1989.
[35] Conte S, Dunsmore H.E, Shen V.Y. Software Engineering Metrics and Models. Benjamin/Cummings, Menlo Park, 1986.
[36] Tate G, Verner J. Software Costing in Practice, The Economics of Information Systems and Software. Oxford, Butterworth-Heinemann, 1991.
[37] Willmott C.J. Some comments on the evaluation of model performance. Bulletion of the American Meteorological Society 1982; 63(11): 1309–1313.
[38] Vashisht V, Lal M, Sureshchandar G.S. Defect prediction framework using neural networks for software enhancement projects. Journal of Advances in Mathematics and Computer Science 2016; 16(5): 1-12.
[39] Sharaf M, Hemdan E.E.D, El-Sayed A, El-Bahnasawy N.A. StockPred: A framework for stock price prediction. Multimedia Tools and Applications 2021; 1-32.