VERİ BİLİMİ DERGİSİ
www.dergipark.gov.tr/veri

Yapay Öğrenme ile Yazılım Test Eforu Tahmini

Özgenil MERİÇ[1,2,*], Ahmet Murat ÖZBAYOĞLU[1]

[1]*TOBB ETÜ[1], Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği, Ankara*
[2]*Aselsan A.Ş., Savunma Sistem Teknolojileri Sektör Başkanlığı, Ankara*

Makale Bilgisi

Başvuru: 17/07/2020
Kabul: 03/01/2021

Özet

Yazılım Test dünyasındaki en önemli problemlerden bir tanesi yazılım test planları oluşturulurken test eforunun net bir şekilde belirlenememesidir. Yazılım test süreci, her zaman yazılımın içerisinde hatalar olduğunu önkoşul olarak kabul etmeli ve olabildiğince fazla hata bulabilmek için çabalamalıdır. Bundan dolayı projelerdeki yazılım test işçiliği için ayrılması gereken süre ve kaynak ihtiyacının doğru bir şekilde belirlenebilmesi, proje takvimlerinin oluşturulabilmesi ve kaynakların verimli bir şekilde kullanılabilmesi için önem arz etmektedir. Bu makalede makine öğrenme algoritmaları kullanarak yazılım test eforu tahmini üzerine yeni bir metot önerilmiştir. Önerilen metot ile ASELSAN bünyesinde geliştirilen, Komuta Kontrol Kullanıcı Arayüzü Yazılımları ve Gömülü ve Gerçek Zamanlı Yazılımları doğrulamak için harcanan test eforu analiz edilerek, ileride yapılması planlanan test aktiviteleri için etkin bir test eforu tahmini yapılmaktadır.

Anahtar Kelimeler: *Yazılım Test Eforu Kestirimi, Yapay Sinir Ağları, Kaynak Kullanımı, Yapay Öğrenme, SVM*

Software Test Effort Estimation with Machine Learning

Abstract

One of the main problems of software test literature is the issue of not having a sound estimation of software test effort while scheduling a plan for the whole software development. Software test process always assumes that there are errors inside within the software under test and struggle in order to find out more errors. There is a great importance of assigning correct amount of time and resources for the software test effort in order to have project calendars that are realistic with efficient resource assignments. In this study, using Machine Learning Algorithms, we propose a new method of software test effort estimation. With the proposed method, using the past experiences of ASELSAN Software Test Efforts in the areas of command control interfaces, embedded and real-time software test developments, we strive for better estimations.

Keywords: *Software Test Effort Estimation, Artificial Neural Networks, Source Optimization, Machine Learning, SVM*

[1] * İletişim e-posta: omeric@etu.edu.tr
** Bu çalışmanın bir kısmı III. International Conference on Data Science and Applications 2020'de sözlü olarak sunulmuştur.

1 Giriş

Savunma sanayinde emniyeti kritik sistemler geliştirilmektedir. Sistemlerde ve yazılımlarda çıkabilecek olası hatalar en üst seviyede önem arz etmektedir. Yazılım test faaliyetleri ile sadece geliştirilen yazılım içerisindeki hataların bulunması değil, yazılımın müşteri/sistem gereksinimlerine uygunluğunun kontrol edilmesi de amaçlanmaktadır. Dolayısıyla yazılım test faaliyetleri, yazılım geliştirme yaşam döngüsü içerisinde önemli bir parça olarak kendini göstermektedir.

Yazılım test yöneticileri açısından projelerde harcanacak olan yazılım test işgücünün doğru bir şekilde belirlenebilmesi, mevcut olan kaynak ve zamanı etkin kullanabilmek açısından oldukça önemlidir. Çoğu zaman uzman görüşüne bağlı olarak belirlenen yazılım test eforunun gerçekçi tahmin edilememesi, test faaliyetlerinin yetersiz yapılmasına sebep olabilmektedir. Bu durum, geliştirilen yazılımın kalitesini önemli ölçüde etkilemektedir. Yazılım metrikleri ve yazılım geliştirme eforu tahmini hakkında geniş bir araştırma yelpazesi olmasına rağmen yazılım test eforu tahmininde araştırmaların oldukça kısıtlı sayıda kaldığı ve gelişime açık bir alan olduğu görülmektedir.

Her kuruma, projeye, çalışan personel yelpazesine uygun bir model geliştirmek oldukça zordur. Bu nedenle çalışmamızda ASELSAN SST Sektör Başkanlığı bünyesinde geliştirilen Komuta Kontrol Yazılımları ve Gömülü ve Gerçek Zamanlı Yazılımlar için yapılan fonksiyonel kara kutu testlerin efor çalışmaları değerlendirilmiştir. İkinci bölümde sırasıyla yazılım test eforu tahmini için yapılan mevcut çalışmalar ve güncel problemlere işaret edilmiştir. Çalışmamızın içerisinde kullanılan çeşitli metotlar ve teorik altyapılarından kısaca bahsedilmiştir. Ardından seçilmiş veri seti altyapı detaylarına yer verilmiş ve çalışma adımları sıralanmıştır. Üçüncü bölümde çalışma sonuçlarına, değerlendirmelere ve gelecek çalışmalar hakkındaki öngörülere yer verilmiştir.

2 Çalışma metotları

2.1 İlgili çalışmalar

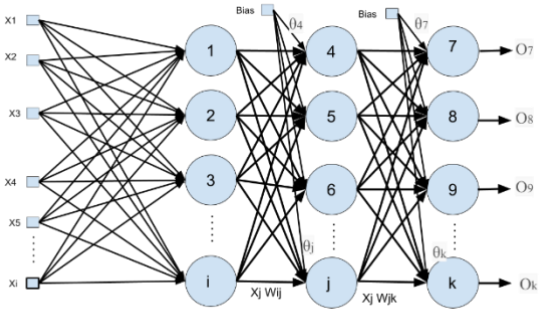
Literatürdeki yazılım test efor çalışmaları [1] genel olarak Test Nokta Analizi (Test Point Analysis), Kullanılan Durum Noktaları (Use Case Points) üzerlerine yapılan makine öğrenme iyileştirmeleri üzerine yoğunlaşmaktadır. Çözüm için çeşitli doğrusal dönüşüm algoritmaları [2] kullanılsa da farklı veri setine sahip sistemler için genel geçerli bir sonuca ulaşılamadığı [3] ve sınırlı veri setiyle beraber zor bir kestirim problemine sahip olduğumuz işaret edilmektedir. Bununla birlikte test altyapısının zorluklarına göre de kestirim metotlarında farklılıklara açık olarak ihtiyaç olduğuna işaret edilmiştir [5]. COCOMO (Yapısal Masraf Modeli) [6] ile deneye dayalı pratik çözümler de bulunmaya çalışılmış fakat endüstri düzeyindeki çalışmalar için sonuçlara yer verilememiştir [6]. Endüstri ve akademi düzeyindeki çalışmaların farklılıkları, akademik test yazılım altyapısındaki gereklilikler ve endüstri yazılım gereksinimlerindeki farklılıklar dolayısı ile iki ana çalışma alanındaki iş birliğinin gerekliliğine işaret etmektedir [9].

2.2 Yazılım Test

Yazılım testin kalitesini belirleyen temel faktör her zaman yazılımın tanımında yatmaktadır. Temel olarak nihai amaç, süreç sonunda yazılımı hata kalmamış bir hale getirmek olarak algılanmaktadır. Fakat testin temel amacı yazılımın yaratılış amacındaki tüm fonksiyonlara başarı ile sahip olup olmadığının anlaşılması olmalıdır. Yapılan testlerin amacı yazılıma ve görevlerine olan güveni arttırmaktır. Bazen bu amaçlar karıştırılabilmektedir. Yazılım test sürecinin amacı yazılımın ürün olarak kalitesine değer katmaktır. Test ile katılan değer kendini yazılım kalitesi ve sistem güvenilirliği olarak göstermektedir. Güvenilirliğin arttırılması, hataların bulunması ve ortadan kaldırılması ile gerçekleşir. Bundan dolayı hiçbir yazılım testçi yazılımın çalıştığını göstermek için test yapmamalıdır. Her zaman yazılımın içerisinde hatalar olduğunu önkoşul olarak kabul etmeli ve olabildiğince fazla hata bulabilmek için çabalamalıdır.

2.3 Teori

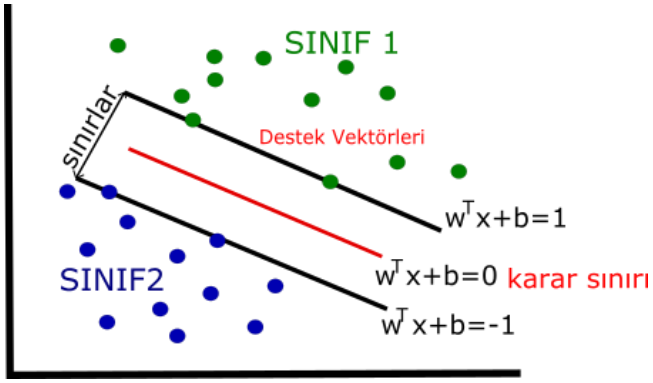
Teorik Altyapılara kısaca yer verilmeye çalışılacaktır. Gerekli altyapı ve fonksiyonlar kullanılan veri altyapısına uygun olarak düzenlenmiştir. Yapay Sinir Ağı, Geri Yayılım Algoritması [8], en basit ve en çok kullanılan denetimli metotlardan biridir. Bu algoritmadaki temel yaklaşım önce eğitilmemiş bir yapay sinir ağı ile başlayıp yapay sinir ağının değişim ağırlıklarını istenilen sonuca göre alıştırmaktan geçer.



Şekil 1. Bir gizli katmanlı Geri Yayılım Sinir Ağı örneği

Şekil 1'de Geri Yayılım Sinir Ağlarına örnek olabilecek bir taslağa yer verilmiştir. Ağ davranışı, bir veri kümesi olarak gösterilen ve bunları önceden tanımlanmış sınıflara sınıflandırması istenen bir insana benzer. Bir insan gibi, örneklerin sınıflara nasıl uyduğu hakkında hipotezler ortaya çıkacaktır. Bunlar daha sonra ağ tahminlerinin ne kadar doğru olduğunu görmek için doğru çıktılara karşı test edilir. En son hipotezdeki köklü değişiklikler, ağırlıklardaki büyük değişimlerle gösterilir ve küçük değişimler, hipotezde küçük ayarlamalar olarak görülebilir.

Rastgele Orman Algoritmaları [8], toplu öğrenim algoritmalarından biridir. Eğitim sırasında birden fazla karar ağacı ile pek çok değişken sahibi istatistiksel modellere uygun olarak sonuç kümelenmeleri yapılandırılır. Temel olarak bu sonuç kümelenmelerinin ortalaması veya küme olarak ayrımlarıyla sonuca karar verilir. Rastgele Orman Algoritması, büyük veri setlerinde etkin bir şekilde çalışır. Girdi değişmesine gerek duymadan binlerce veriyi işleyebilir, önemli değişkenlerin tahminlerini verir ve orman büyümesi ilerledikçe sabit bir genelleştirme hatası üretir. Eksik verileri tahmin etmek için etkili bir metoda sahiptir. Büyük veri oranları eksilse bile sınıf popülasyonunun dengesiz olma durumlarına karşı veri kümelerinde sınıf hatasını dengeleme yöntemleri vardır. Ayrıca; çok parçacıklı, çok çekirdekli ve paralel mimariler kullanarak paralel uygulamalara ön ayak olmuştur.



Şekil 2. Destek Vektör Makineleri Örneği

Destek Vektör Makineleri (SVM) [8], verilen eğitim setini, farklı boyutlarda noktalar olarak temsil edip, temsil edilen örnekler arasında açıkları kullanarak karar vermeye çalışır. Basit bir örnek olarak; iki küme arasındaki sınıflandırmayı yapmaya çalıştığınızda ikili doğrusal sınıflandırıcı ile çalışılır. Şekil 2'de kısaca örneklenmeye çalışılmıştır. Bu noktada farklı tanıları sınıflandıracak niteliksel özellikler çok önemli rol oynamaktadır. Destek Vektör Makineleri sınıflandırmalarındaki temel amaç uygun bir hesaplama zorluğu ile çok boyutlu düzlemler arasında ayırıcı hiper düzlemler oluşturabilmektir.

2.4 Veri seti

Bu çalışmada kullanılan veri seti, ASELSAN bünyesinde 2011-2019 yılları arasında geliştirilmiş olan 8 farklı projeye ait 31 yazılım konfigürasyon birimi için gerçekleştirilmiş olan test sonuçları analiz edilerek oluşturulmuştur. Veri setinde 85 adet test verisi örneği bulunmaktadır. Her bir veri örneği, bir yazılım konfigürasyon birimi için yapılmış olan kara kutu fonksiyonel test sonuçlarını içermektedir ve gerçek değerli bir test eforu cevabını tahmin etmeyi amaçlayan 5 özellikten oluşmaktadır. Çıktı değeri olarak bir test döngüsünde harcanan saat bilgisi bulunmaktadır. Nageswaran'ın [7] önerdiği parametreler, kendi verilerimize uygun olarak düzenlenmiş ve veri setinde kullanılan projeler göz önünde bulundurularak bu 15 parametrenin ağırlıkları belirlenmiştir. Veri setinde bulunan her bir veri için uzman görüşü ortalamaları alınarak parametre değerleri hesaplanmıştır.

Tablo 1'de, testlerin zorluk seviyesi ağırlıklarına ve açıklamalarına yer verilmiştir. Gerçekleştirilen testler yüzdesel olarak basit, ortalama ve zor olarak

sınıflandırılarak ağırlıklar ile çarpılıp toplam değerleri hesaplanmıştır.

Tablo 1. Testlerin zorluk seviyesi

TESTLERİN ZORLUK SEVİYESİ	AĞIRLIK	AÇIKLAMA
Basit	1	Basit kullanıcı arayüzü işlemleri vb.
Ortalama	2	Konfigürasyon dosyası işlemleri, veri tabanı kontrolü vb.
Zor	3	Simülatör ya da başka bir yazılım/sistem kullanımı, dosya yükleme işlemleri, uzun süren test adımları vb.

Tablo 2. Teknik Faktörler

TEKNİK FAKTÖRLER	AĞIRLIK
Test Otomasyonu	5
Test Verisi Kullanma	5
Donanım Üzerinde Test İhtiyacı	6
Test Başlangıcında Donanım Altyapısının Kurulması İhtiyacı	7
Dağıtık Sistemlerde Çalışma	4

Tablo 3. Test ekibinin deneyimi

TEST EKİBİNİN DENEYİMİ	AĞIRLIK
Deneyimli	3
Deneyimli ve Deneyimsiz Karışık	2
Deneyimsiz	1

Tablo 4. Yazılım Tipi

YAZILIMIN TİPİ	SINIF
Kullanıcı Arayüz Testleri	0
Gömülü Sistemler Testleri	1

Tablo 2'de görülen teknik faktörler içerisinde yer alan parametrelerin test eforunu ciddi şekilde etkilediği bilinmektedir. Özellikle Gömülü ve Gerçek Zamanlı yazılımların testlerindeki donanımsal altyapı ihtiyacı test eforunu arttırmaktadır. Bu testlerde donanımsal altyapının kurulma süresi, toplam test süresi içerisinde en büyük kalemdir. Bu sebeple bu iki parametrenin ağırlığının yüksek olması değerlendirilmiştir. Tablo 3 ve 4'te görülen test ekibi deneyimi ve yazılım tipleri de test süresinde büyük varyanslara sahip olabileceğinden önemli giriş değişkenleri arasında yerlerini almıştır.

2.5 Yöntem

Bu çalışmada, Kara Kutu test koşma döngüsü, yani Kara Kutu test senaryosu paketinin tam koşulabilmesi için gereken test çabasını tahmin etme problemi üzerine çalışılmaktadır. Bu problemin çözümünde geleneksel yöntemler kullanmak yerine yapay öğrenme (makine öğrenmesi) metotları ile geçmiş verileri analiz ederek yeni bir efor kestirimi yapılmaktadır.

Tablo 5. Yapay Öğrenme Giriş Değişkenleri

DEĞİŞKENLER
1) Gerçekleştirilen test senaryosu sayısı
2) Test edilen yazılımın tipi
3) Test senaryolarının zorluk seviyesi
4) Teknik faktörler
5) Test ekibinin tecrübesi

Yapay Öğrenme Giriş değişkenleri Tablo 5'te verilmiştir.

Bu çalışmada test eforu tahmini problemine çözüm olabilecek 6 model (Tablo 6) üzerinde çalışıldı ve sonuçları karşılaştırıldı:

Tablo 6. Çalışılan Modeller

MODELLER
1) Levenberg-Marquardt Geri Yayılım Algoritması (geleneksel MLP (Multilayer Perceptron) ağı ve ikinci derece Levenberg-Marquardt geri yayılım algoritması)
2) Bayesci Geri Yayılım Algoritması (geleneksel MLP ağı ve Bayesci geri yayılım algoritması)
3) Rastgele Orman Regresyonu
4) Destek Vektör Makinesi (SVM) Regresyonu
5) Gauss Çekirdek fonksiyonu ile Destek Vektör Makinesi (SVM) Regresyonu
6) Otomatik Hiper-parametre Optimizasyonu ile Destek Vektör Makinesi (SVM) Regresyonu

Yapay öğrenme tekniklerinin performansını etkileyen en önemli sorunlardan bir tanesi de kullanılan veri tabanları içerisindeki gereksiz veya aykırı verilerdir. Veri tabanları içerisindeki uygun olmayan ve gereksiz verileri veya değişkenleri ortadan kaldırarak öğrenme algoritmalarının performansını arttıracak birçok yöntem bulunmaktadır. Bizim veri setimizde olduğu gibi veri örneklerinin sayısının nispeten az olduğu durumlarda, uygun olmayan ve gereksiz değişkenlerin getirdiği gürültü nedeniyle değişken seçim sürecinin önemi daha da artmaktadır.

Bu çalışmada, veri setinde bulunan veriler içerisinde alakasız ve gereksiz verilerin ayıklanması için bir çalışma yapılmıştır. Veri setinde bulunan bütün veri örnekleri hem eğitim veri setinde hem de test veri seti içerisine eklenerek eğitim ve test veri setlerindeki performanslarına bakılmıştır. Bütün modellerde test veri setinde kötü performans sergileyen veriler, eğitim veri seti içerisine alındığında sistemin genel performansı izlenmiştir. Bu verileri veri setimizden çıkarttığımızda sistemin genel performansında artış gözlenmiştir.

ASELSAN'da test eforu tahmini için ortalama bir değer hesaplanmaktadır. Hesaplama formülü doğrusal bir sonuca tabiidir. İnsan faktörü ve yapılan iş zorluğu yazılım test efor tahminini etkilememektedir.

Ortalama olarak bir test mühendisinin günde 25 test gerçekleştirebileceği baz alınarak adam/saat olarak bir tahmin yapılmaktadır. Test sonuçlarımızı alırken Tablo 7'de bulunan adımlar uygulanmıştır.

Tablo 7. Sonuç Adımları

ADIMLAR
1) Veri setini rastgele olarak %84 oranında eğitim seti olarak, %16 oranında test seti olarak ayırılmıştır.
2) Geri Yayılım algoritmalarında çapraz doğrulama yapıldığı için burada %70 oranında eğitim seti, %15 oranında çapraz doğrulama ve %15 oranında test verisi olarak ayrılmıştır.
3) Geri yayılım algoritmasında daha sağlıklı ölçümler elde edebilmek için her bir veri seti 10'ar kez koşturulup test eforu tahmini sonuçlarının ortalaması alınmıştır.
4) Her yapay öğrenme yöntemi için sonuçlar kayıt altına alınarak performansları ölçümlenmiştir.
5) Bütün yöntemlerin performansları karşılaştırılmıştır.

3 Deneysel Sonuçlar

Son olarak, Tablo 8'de verilen yapay öğrenme yöntemleri test eforu tahmini problemi üzerinde sonuç performansları karşılaştırılmıştır. Karşılaştırma tablosundan da görülebileceği üzere SVM regresyonu RMSE (Root Mean Square Error) ve MAE (Mean Absolute Error) için değerlendirildiğinde tüm metotlara göre en iyi performansı sergilemiştir. Bununla birlikte temel ve kullanışlı bir metot olan Bayesci Geri Yayılım Algoritması MAPE (Mean Absolute Percent Error) için değerlendirildiğinde en iyi performansı sergilemiştir. Bütün değerlendirmeler içinde en kötü tahmin Gauss Çekirdek fonksiyonu ile Destek Vektör Makinesi (SVM) Regresyonunda yapılmıştır.

Tablo 8. Sonuçlar

METOT	RMSE (saat)	MAPE (saat)	MAE (saat)
MLP-Geri Yayılım Algoritması-LM	134,554	0,446	60,044
MLP-Geri Yayılım Algoritması-BR	133,500	0,393	54,262
Rastgele Orman Algoritması	138,791	0,470	61,310
SVM Regresyonu	94,271	0,423	46,299
SVM Regresyonu-Gaussian	167,145	0,627	65,312
SVM Regresyonu-Hiperparametre Optimizasyonu	95,889	0,409	46,700
ASELSAN Tahmini	154,774	0,605	81,142

Tablo 9'da çalışmada kullanılan veri seti üzerinde ASELSAN'da yapılan mevcut tahminlere göre farklı metotların performans yüzdelerine yer verilmiştir.

Tablo 9. Mevcut Tahmine göre performans karşılaştırması

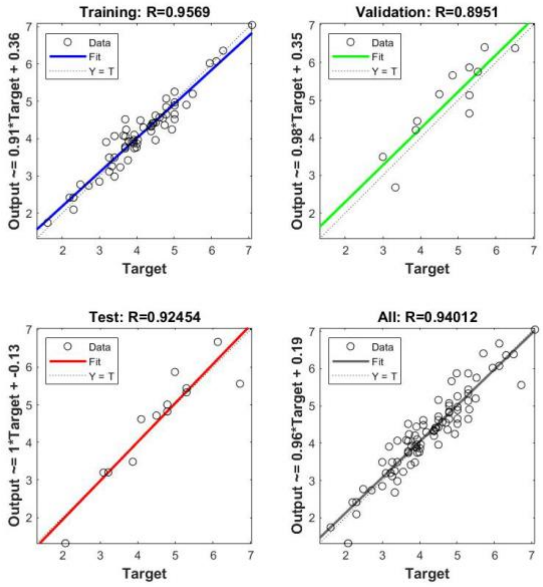
METOT	%
MLP-Geri Yayılım Algoritması-LM	67,033
MLP-Geri Yayılım Algoritması-BR	73,626
Rastgele Orman Algoritması	70,330
SVM Regresyonu	73,626
SVM Regresyonu-Gaussian	58,242
SVM Regresyonu-Hiperparametre Optimizasyonu	73,626

Verilerin %73,626'sında önerilen model (SVM ve MLP) ASELSAN'ın mevcut tahmin modelinden daha az hata ile sonucu tahmin etmiştir.

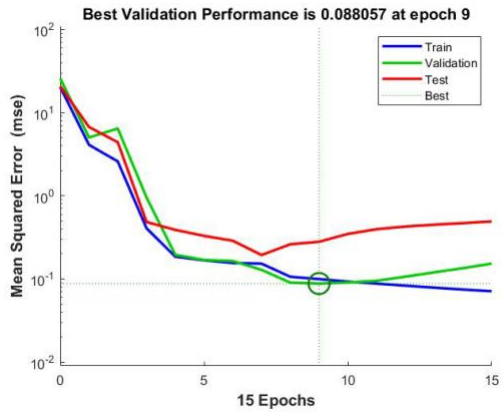
		Paired Differences					t	df	Sig. (2-tailed)
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference				
					Lower	Upper			
Pair 1	Random_Forest - BackProp_BR	9,35419	87,31347	9,47048	-9,47889	28,18726	0,988	84	0,326
Pair 2	Random_Forest - BackProp_LM	-0,30549	99,57863	10,80082	-21,7841	21,17313	-0,028	84	0,978
Pair 3	Random_Forest - SVM	4,72988	81,06887	8,79315	-12,75627	22,21603	0,538	84	0,592
Pair 4	Random_Forest - SVM_Gaussian	34,48635	122,55985	13,29348	8,05081	60,9219	2,594	84	0,011
Pair 5	Random_Forest - SVM_Opt	4,28866	85,08662	9,22894	-14,0641	22,64142	0,465	84	0,643
Pair 6	BackProp_BR - BackProp_LM	-9,65967	47,06455	5,10487	-19,81126	0,49192	-1,892	84	0,062
Pair 7	BackProp_BR - SVM	-4,6243	60,78462	6,59302	-17,73524	8,48663	-0,701	84	0,485
Pair 8	BackProp_BR - SVM_Gaussian	25,13217	107,65773	11,67712	1,91093	48,3534	2,152	84	0,034
Pair 9	BackProp_BR - SVM_Opt	-5,06552	51,66854	5,60424	-16,21017	6,07912	-0,904	84	0,369
Pair 10	BackProp_LM - SVM	5,03537	70,83956	7,68363	-10,24437	20,31511	0,655	84	0,514
Pair 11	BackProp_LM - SVM_Gaussian	34,79184	128,74718	13,96459	7,02172	62,56196	2,491	84	0,015
Pair 12	BackProp_LM - SVM_Opt	4,59415	64,90052	7,03945	-9,40457	18,59287	0,653	84	0,516
Pair 13	SVM - SVM_Gaussian	29,75647	110,79596	12,01751	5,85834	53,65461	2,476	84	0,015
Pair 14	SVM - SVM_Opt	-0,44122	19,703	2,13709	-4,69106	3,80862	-0,206	84	0,837
Pair 15	SVM_Gaussian - SVM_Opt	-30,19769	112,90032	12,24576	-54,54973	-5,84566	-2,466	84	0,016
Pair 16	ASELSAN Mevcut Tahmin - Random_Forest	-68,00685	117,04844	12,69569	-93,25361	-42,76008	-5,357	84	0
Pair 17	ASELSAN Mevcut Tahmin - BackProp_BR	-58,65266	106,15712	11,51436	-81,55022	-35,7551	-5,094	84	0
Pair 18	ASELSAN Mevcut Tahmin - BackProp_LM	-68,31233	119,71369	12,98477	-94,13398	-42,49069	-5,261	84	0
Pair 19	ASELSAN Mevcut Tahmin - SVM	-63,27697	79,82369	8,65809	-80,49454	-46,05939	-7,308	84	0
Pair 20	ASELSAN Mevcut Tahmin - SVM_Gaussian	-33,52049	82,45915	8,94395	-51,30652	-15,73446	-3,748	84	0
Pair 21	ASELSAN Mevcut Tahmin - SVM_Opt	-63,71818	89,47481	9,70491	-83,01745	-44,41892	-6,566	84	0

Bu çalışmada kullanılan metotların ortalamaları arasındaki farkın anlamlılığını test etmek için Tek Yönlü Varyans Analizi (ANOVA) kullanılmıştır. ANOVA, istatistik biliminde çoklu gruplar için kullanılan varyans analizinin başlıca tekniğidir.

ANOVA sonuçlarına göre çalışmada kullanılan yapay öğrenme metotları arasında anlamlı bir farklılığın olmadığı (Sig (p)> 0.05) fakat Gauss Çekirdek fonksiyonu ile Destek Vektör Makinesi (SVM) Regresyonu için bir farklılık olduğu görülmüştür. ASELSAN'da kullanılan mevcut tahminler ile çalışmada kullanılan metotlar arasında ise beklenildiği gibi anlamlı bir farklılık çıkmıştır (Sig (p) <0.05).



Şekil 3. Levenberg-Marquardt Geri Yayılım Algoritması için eğitim, çapraz doğrulama ve test performansından bir örnek



Şekil 4. Levenberg-Marquardt Geri Yayılım Algoritması için epoch/mse grafiği

Şekil 3 ve 4'de Levenberg-Marquardt (LM) Geri Yayılım Algoritması için eğitim, çapraz doğrulama ve test performansından örneklere yer verilmiştir. Görüldüğü gibi LM Geri Yayılım Algoritması 9 devir sonrası çalıştığımız öğrenme verileri ile birlikte optimal sonuçlara ulaşmıştır. Veriye bağımlılığı diğer geleneksel metotlara göre daha yüksek olan bu algoritma 9 devir sonrasında bunun etkilerini test veri performanslarında göstermektedir.

4 Sonuç

Sonuç olarak bundan sonraki yazılım test eforlarını daha iyi bir oranla tespit edebileceğimiz; ASELSAN içerisinde daha iyi bir test efor kestirimi için kullanılabilecek en iyi yapay öğrenme metotları belirlenmiştir. Kestirim algoritma girdilerinin sistematik olarak sağlanabileceği bir altyapı ile günümüz endüstri standartlarına uygun proje zaman kıstaslarını daha doğru sağlayabilecek dinamik temel taşlar atılmıştır. Devam eden projeler ile birlikte verilen yazılım test eforları eklenerek kullanılan veri seti genişletilecek, gerçeğe daha yakın yazılım test eforları için yapay öğrenme altyapıları iyileştirilecektir.

Kaynaklar

[1] Natália França Felipe, (2014) "A Comparative Study of Three Test Effort Estimation Methods", Revista Cubana de Ciencias Informáticas, Vol. 8, No. Especial UCIENCIA

[2] Praveen Ranjan Srivastava, (2015) "Estimation of software testing effort using fuzzy multiple linear regression", Int. J. Software Engineering, Technology and Applications, Vol. 1, Nos. 2/3/4

[3] Daniel G. e Silva, (2010) "Machine learning methods and asymmetric cost function to estimate execution effort of software testing", Third International Conference on Software Testing, Verification and Validation

[4] Praveen Ranjan Srivastava, (2012) "Software test effort estimation: a model based on cuckoo search", Int. J. Bio-Inspired Computation, Vol. 4, No. 5

[5] Dharmender Singh Kushwaha and A.K. Misra, (2008) "Software Test Effort Estimation", ACM SIGSOFT Software Engineering Notes, May 2008 Vol. 33 No. 3

[6] Prasanta Bhattacharya, (2012) "Software Test Effort Estimation Using Particle Swarm Optimization"

[7] Suresh Nageswaran, (2001) "Test Effort Estimation Using Use Case Points" Quality Week 2001, San Francisco, California, USA, June 2001

[8] Richard O. Duda, Peter E. Hart, David G. Stork, "Pattern Classification" Second Eddition, ISBN-13: 978-0471056690

[9] O. Ege Adalı, N. Alpay Karagöz, (2017) "Software Test Effort Estimation" 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)