Yazılım Hata Kestirimi İçin Kaynak Kod Ölçütlerine Dayalı Bayes Sınıflandırması

Burak Turhan[1] Ayşe Bener[2]

[1,2]Bilgisayar Mühendisliği Bölümü, Boğaziçi Üniversitesi, İstanbul
[1]e-posta: turhanb@boun.edu.tr [2] e-posta: bener@boun.edu.tr

Özetçe

Bu bildiride kaynak kodlardan çıkartılmış ölçütler kullanılarak hata kestirimi yapan Bayes sınıflandırıcıları incelenmiştir. Ölçütlerin çok değişkenli veya tek değişkenli Normal dağılımlarından geldikleri varsayımlarına dayalı çeşitli karmaşıklık seviyelerinde modeller oluşturulmuştur. Bu modeller kamuya açık, gerçek projeler içeren veri kümeleri üzerinde sınanmış ve yüksek kestirim performansları ile genelleme yapabilme özellikleri gözlemlenmiştir. Hata kestirimi modellerinden elde edilen bilgilerin göz önünde bulundurulması, endüstride yazılım test planları oluştururken, test kaynaklarının verimli kullanımı açısından önemli faydalar sağlayabilir.

1. Giriş

Yazılım testi, kullanılan geliştirme modelinden bağımsız olarak yazılım geliştirme döngüsünün yaklaşık yarı zamanını alan bir aşamadır [1]. Bu nedenle verimli testler yapabilmek, değerli kaynakların doğru yerlerde kullanılması ve maliyetlerin düşürülmesi açısından çok önemlidir. Maliyeti arttıran diğer bir unsur da yazılım sürümlerinin piyasaya dağıtılmasından sonra ortaya çıkan sorunların giderildiği bakım aşamasıdır. Yapılan araştırmalar yazılım hatalarının geliştirme ve test gibi erken safhalarda bulunup düzeltilmesinin, bakım aşamasında düzeltilmesine oranla çok daha düşük maliyetlerde olduğunu göstermektedir [2].

Hata kestirim çalışmaları, yazılımda bulunan hataların mümkün olduğunca erken ve otomatik bir şekilde belirlenmesini amaçlar. Bu amaçla geliştirilen modellerin kullanılması yukarıda sayılan sorunları azaltarak yazılım geliştirme ve bakım maliyetlerinin en aza indirgenmesinde faydalı olmaktadır. Hata kestiriminin insandan bağımsız ve otomatik yapılabilmesi, olası insan kaynaklı hataları ve nesnellikleri engellemekle birlikte, insandan çok süreçlere dayalı kurumsallaşmayı kolaylaştırmaktadır. Hata kestirim modellerinde girdi olarak kaynak kod ve süreçlerden elde edilecek ölçütler kullanılabilir. Bu çalışmada kaynak kod ölçütlerine dayalı bir model önerilmektedir. Bunun en önemli sebepleri, süreç ölçütlerinin tanımlanmadığı ya da toplanmadığı kurumlarda proje yöneticilerine daha fazla yük getirilmemesi ve kaynak kod ölçütlerinin çeşitli yazılımlar aracılığı ile otomatik olarak toplanabilmesi kolaylığıdır.

Herhangi bir modelde kullanılan girdilerin niteliği model performansını ciddi şekilde etkilemektedir. Literatürde, kaynak kodlardan çıkartılabilecek çok sayıda ölçüt mevcuttur [3, 4]. Yüksek performanslı bir hata kestirim analizi için bu ölçütlerden hangilerinin girdi olarak kullanılacağına karar verilmesi önemlidir.

Bildirinin 2. bölümünde çalışmamızda kullanılan modeller ayrıntılı olarak açıklanmıştır. 3. Bölümde deney tasarımı ve deney sonuçları sunulmuştur. 4. Bölümde ise tartışmaya yer verilmiştir.

2. Kestirim Modelleri

Çalışmamızda Bayes sınıflandırması modelini kullandık. Bunun sebebi literatürde belirtilen en iyi sonuçların bu yöntemle elde edilmesidir [3]. Bayes sınıflandırmasının varsayımları aşağıda listelenmiştir:

1. Kullanılan ölçütler birbirlerinden bağımsızdır
2. Her ölçütün sınıflandırmaya etkisi eşit derecededir.

Çalışmamızda, birinci varsayımın yazılım ölçütleri açısından geçerli olmadığı gerçeğine dayanarak Bayes sınıflandırıcısında kullanılan tek değişkenli Normal dağılımı yerine çok değişkenli Normal dağılımını kullandık. Modellerin ayrıntılarından önce bu iki dağılımın temel farklarını açıklayacağız.

2.1. Tek Değişkenli ve Çok Değişkenli Normal Dağılım

Tek değişkenli normal dağılımda, $x \sim N(\mu, \sigma^2)$, veri noktası *x*, ortalama değer μ ve standart sapma σ ile dağılır. Olasılık yoğunluk fonksiyonu Formül 1'deki şekilde tanımlanır [9].

$$p(x) = \frac{1}{\sqrt{(2\pi)\sigma}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2} \right) \quad (1)$$

Formül 1'e göre, veri noktalarının ortalamadan sapmalarının standart sapmalar cinsinden değerleri üstel fonksiyon içerisinde bir uzaklık birimi olarak kullanılır. Tek değişkenli bir dağılım olduğundan, değişkenler arası ilintiler dikkate alınmaz.

Çok değişkenli normal dağılımda ise, $x \sim N(\vec{\mu}, \Sigma)$, veri noktaları bütün ölçütlere karşılık gelen değerleri içeren vektörler olarak tanımlanır. Veri noktası x, ortalama vektörü μ ve ortak değişinti matrisi Σ ile dağılır. Olasılık yoğunluk fonksiyonu Formül 2'deki şekilde tanımlanır [9].

$$p(\vec{x}) = \frac{1}{\sqrt{(2\Pi)^{d/2}\Sigma^{1/2}}} \exp\left(-\frac{1}{2}(\vec{x}-\vec{\mu})^T \Sigma^{-1} (\vec{x}-\vec{\mu}) \right) \quad (2)$$

Burada üstel fonsiyon içerisinde kullanılan uzaklık fonsiyonu Mahalanobis uzaklığıdır ve ölçüt değerlerinin doğrusal

ilintilerini dikkate alır.

Çalışmamızda kullanılan modellerin karmaşıklık seviyeleri açıklanan dağılım türlerine göre değişiklik göstermektedir. Tek değişkenli normal dağılımda eldeki veriden kestirilmesi gereken 2 parametre varken, çok değişkenli dağılımda kestirilmesi gereken parametre sayısı verinin boyutunun karesi ile orantılıdır.

2.2. Kullanılan Modeler

Hata kestiriminin amacı, hatasız ve hatalı yazılım modüllerini sırasıyla C_0 ve C_1 olarak sınıflandırmaktır. Önceki kısımda açıklanan dağılımlar ile Bayes teoreminin birleştirilmesi sonucu bu sınıflandırmayı yapacak olan sınıflandırıcılar elde edilebilir. Bayes teoremi, sonsal dağılımın önsel dağılım ve olabilirlikle orantılı olduğunu söyler. Bayes teoremi Formül 3'teki şekilde yazılır [9].

$$P(C_i \mid x) = \frac{P(x \mid C_i)P(C_i)}{P(x)} \qquad (3)$$

Paydada bulunan terim bütün örnekler için ortak olduğundan göz ardı edilebilir. Hesaplama kolaylığı açısından her iki tarafın logaritması alınarak, bir veri noktasının C_0 ve C_1 için ayıracı Formül 4'teki gibi elde edilir.

$$g_i(x) = \log(P(x \mid C_i)) + \log(P(C_i)) \qquad (4)$$

Formül 4'te önsel dağılım olarak tek değişkenli ya da çok değişkenli normal dağılım kullanılması farklı karmaşıklıkta modeller üretilmesini sağlar. Eğer her iki sınıfın da farklı ortak değişinti matrislerine sahip olduğu varsayımı yapılırsa Formül 5'te belirtilen karesel ayıraç elde edilir [9].

$$g_i(\vec{x}) = \vec{x}^T W_i \vec{x} + w_i^T \vec{x} + w_{i0} \qquad (5)$$

Ölçüt sayısının d olduğu varsayılırsa, karesel ayracı oluşturabilmek için ortak değişinti matrisi için $2d^2$, ortalama vektöreri için de $2d$ parametre kestirimi gerekmektedir. Karesel ayıraç bu çalışmada kullanılan en karmaşık modeldir.

Karmaşıklığı azaltmak için, her iki sınıfa ait örneklerin tek bir ortak değişinti matrisini paylaştıkları varsayımı yapılabilir. Bu durumda Formül 6 ile ifade edilen doğrusal ayıraç elde edilir [9].

$$g_i(\vec{x}) = \vec{w}_i^T \vec{x} + w_{i0} \qquad (6)$$

Paylaşılan ortak değişinti matrisi, her sınıfın kendi ortak değişinti matrislerinin, sınıfların önsel olasılıkları ile ağırlıklandırılmaları ile elde edilebilir. Doğrusal ayıraç modelinde paylaşılan ortak değişinti matrisi için d^2, ortalama vektörleri için de $2d$ parametre kestirimi gerekmektedir.

Karmaşıklık seviyesi en az olan son modelde ise paylaşılan ortak değişinti matrisinin ana köşengen olmayan tüm üyelerinin sıfır (0) olduğu varsıyımı yapılarak Naive Bayes sınıflandırıcısı elde edilir. Naive Bayes sınıflandırıcısı Formül 7'deki şekilde belirtilir [9].

$$g_i(x) = -\frac{1}{2}\sum_{j=1}^{d}\left(\frac{x_j^t - m_{ij}}{s_j}\right)^2 + \log(P(C_i)) \qquad (7)$$

Bu varsayımlar ile elde edilen Naive Bayes modeli, tek değişkenli normal dağılım kullanmaya karşılık gelir. Bu modelde kestirilmesi gereken parametre sayısı, $2d$ ortalama üyesi ve d köşegen matris üyeleri parametreleridir.

Modellerin karmaşıklık seviyeleri Tablo 1'de özet olarak verilmiştir.

Model	Parametre Sayısı
Karesel	(2 x (d x d)) + (2 x d)
Doğrusal	(d x d) + (2 x d)
Naive Bayes	(d) + (2 x d)

Tablo 1. Model Karmaşıklıkları

2.3. Kullanılacak Ölçütlerin Seçimi

Bu ölçütlerin veya özniteliklerin seçimi konusundaki çalışmaları iki kategoride değerlendirebiliriz:

- Öznitelik seçimi
- Öznitelik çıkarımı

Öznitelik seçiminde gözlemlenen ölçüt kümesinin en iyi alt kümesinin seçilmesi amaçlanmaktadır. Ancak bu tam kapsamlı bir arama gerektirdiğinden çeşitli buluşsal yöntemler tercih edilmektedir. Bunlar arasında ileriye doğru arama ve geriye doğru arama en sık kullanılan metodlardır [5]. İleriye doğru aramada, boş bir ölçüt kümesiyle başlayarak ve her adımda rastgele bir ölçüt eklenerek, kestirim modelinin performansı gözlemlenir. Eğer performans önemli derecede artarsa bu ölçüt kullanılacak ölçütler kümesine eklenir; aksi halde gözardı edilir ve yeni bir ölçüt için aynı işlem uygulanır. Geriye doğru aramada ise eldeki bütün ölçüt kümesi ile başlayarak her adımda rastgele bir ölçüt kümeden çıkartılır ve kestirim modelinin performansı gözlemlenir. Performans önemli derecede artarsa bu ölçüt kümeden çıkartılır ve yeni bir ölçüt denenir; aksi halde bu ölçüt kullanılacak ölçütler kümesinde kalır. Bu yaklaşımların en temel sorunu ölçütlerin kestirim modeline olan etkilerini birbirlerinden bağımsız olduklarını varsayarak ölçmeleridir. Oysa ki bir ölçüt tek başına performansa katkı sağlamazken, başka bir ölçütle veya ölçütlerle birlikte kullanıldığında performansı önemli dercede arttırabilir [6]. Öznitelik seçiminin yaygın kullanılma sebepleri arasında gözlemlenebilir öğeleri açıklamaları, eşik değerlerin belirlenebilmesi ve kolay anlaşılabilir kurallar tanımlanabilmesini sayabiliriz [7].

Öznitelik çıkarımındaki amaç ise, gözlemlenen ölçüt kümesi kullanılarak yeni ölçütler elde edebilmektir. Bu yöntemlerle gözlemlenen ölçütler ağırlıklandırılır ve doğrusal ya da doğrusal olmayan kombinasyonları elde edilir. Öznitelik çıkarım yöntemleri, öznitelik seçim yöntemlerinin dezavantajlarını içermese de, ölçütlere verilecek olan ağırlıkların belirlenmesi ve ne tür kombinasyonların kullanılacağına karar verilmesi gerekmektedir. Bu alanda en

sık kullanılan yöntem Ana Bileşenler Analizi (PCA)'dir [8]. PCA, en küçük kare hataların toplamını en aza indirgeyen, analitik çözüme sahip doğrusal bir yöntemdir. Ölçütlerin seçimi amacıyla bu çalışmada PCA kullanılmıştır.

3. Deneyler ve Sonuçlar

Hata kestirimi için önerilen modellerin sınanması için NASA tarafından kamuya açıan 4 adet veri kümesini kullandık [10]. PC1, PC2, PC3 ve PC4 kod adlı projelere karşılık gelen bu veri kümelerinin özellikleri Tablo 2'de verilmiştir.

İsim	Ölçüt Sayısı	Modül Sayısı	Hata Oranı (%)
PC1	41	1107	6
PC2	41	5589	0.6
PC3	41	1563	10
PC4	41	1458	12

Tablo 2. Veri kümeleri açıklamaları

Tablo 2'de görüldüğü gibi veri kümelerindeki örnek sayıları 1107 ile 5589 arasında değişmektedir ve oldukça geniş bir veri tabanı sağlamaktadır. Her modül için McCabe, Halstead ve satır sayısına dayalı ölçütler olmak üzere 3 ana kategoride özetlenebilecek 41 adet ölçüt mevcuttur. Kullanılan ölçütler Tablo 3'te belirtilmiştir. (Ölçütlerin tamamı için karşılık gelen Türkçe terimler literatürde mevcut olmadığından, bütünlüğün korunması için ölçüt isimleri İngilizce olarak yazılmıştır).

Projelerdeki hata oranlarına bakıldığında, toplam modül sayısının binde 6'sı ile yüzde 12'si arasında değişen hata oranları görülmektedir. Hatalı ve hatasız olarak sınıflandırılması gereken modüllerin dağılımındaki dengesizlik, modellerin performanslarının ölçümünde yaygın olarak kullanılan başarım kriterinin sorgulanmasını gerektirmektedir [3]. Örnek olarak PC1 projesinde kullanılacak olan ve bütün modüllere hiçbir analiz yapmadan hatasız diyen bir hata kestirimi modeli %94 başarım oranını sağlayacaktır. Bu sorunun üstesinden gelmek için başka performans kriterleri de değerlendirilmelidir.

Bu amaçla sonuçlarımızı raporlarken, hata yakalama olasılığı (pd) ve yanlış ikaz olasılığı (pf) olarak tanımlanan kriterleri kullandık. Hata yakalama olasılığı yalnızca gerçekten hatalı olan modüller içerisindeki başarımı ölçerken, yanlış ikaz olasılığı yalnızca gerçekten hatalı olmayan modüller içerisindeki kestirim hatasını ölçer. Bu performans kriterlerinin tanımları başarım ile birlikte, Tablo 4'de gösterilen hata matrisinin elemanları cinsinden Formül 8, Formül 9 ve Formül 10'da belirtilmiştir. Başarılı bir modelin pd değerinin yüksek ve pf değerinin düşük olması beklenmektedir.

$$Başarım = (A+D) / (A+B+C+D) \quad (8)$$
$$Pd = (A) / (A+C) \quad (9)$$
$$Pf = (B) / (B+D) \quad (10)$$

Kullanılan modellerin genelleme yapabilme kapasitelerini ölçmek ve veri kümelerini ezberlemekten kaçınmak için bütün deneylerde 10 katlı çapraz geçerleme kullanılmıştır. Verilerin sırasının model performanslarını etkilemesini engellemek için çapraz geçerleme deneyleri, her seferinde verilerin sırası rastgele değiştirilerek 10 kez tekrarlanmıştır. Özet olarak her bir modelin her bir veri kümesi üzerindeki performansı 10x10=100 deneyin ortalama sonucu olarak raporlanmıştır. Tüm gerçekleştirimler MATLAB ortamında standart araç takımları kullanılarak yapılmıştır.

Ölçüt İsmi
Cyclometric Complexity
Cyclometric Density
Decision Density
Design Density
Essential Complexity
Essential Density
Global Data Density
Global Data Complexity
Maintenance Severity
Module Design Complexity
Normalized Cyclometric Complexity
Pathological Complexity
Halstead Length
Halstead Volume
Halstead Level
Halstead Difficulty
Halstead Intelligent Content
Halstead Programming Effort
Halstead Error Estimate
Halstead Programming Time
Branch Conut
Call Pairs
Condition Count
Decision Count
Edge Count
Formal Parameter Count
Modified Condition Count
Multiple Condition count
Node count
Number of Lines
Number of Operators
Number of Operands
Number of Unique Operators
Number of Unique Operands
Number of Executable of Lines of Code
Number of Lines of Comment
Number of Lines of Code containing Code and Comment
Number of Lines of Comment
Percent of Code that is Comments
Total Number of Blank Lines
Total Number of Lines of Code

Tablo 3. Ölçüt kümesi

Gerçek	Tahmin Edilen	
	Hatalı	Hatasız
Hatalı	A	C
Hatasız	B	D

Tablo 4. Hata Matrisi

Ölçütler modellere girdi olarak verilmeden önce PCA kullanılarak %95 değişintiyi açıklayacak şekilde boyut indirgemesinden geçirilmiştir. Bunun yanısıra ölçütlerin dağılımının normal dağılıma yakınsaması için logaritma filtrelemesi uygulanmıştır [3].

Veri	Model	pd(%)	pf(%)
PC1	N.Bayes	**68**	25
PC2	N.Bayes	72	13
PC3	Doğrusal	**76**	31
PC4	Karesel	88	20
Ortalama:		***76***	***22***
Ortalama [3]:		74,5	23,75

Tablo 5. Sonuçlar

Deneylerden elde edilen sonuçlar Tablo 5'de gösterilmiştir. Tablo 5'te kalın yazıyla belirtilen girişler t-testi sonucunda diğer modellerden daha iyi performans gösterenlerdir. Her veri kümesi için yapılan 100 çapraz geçerleme deneylerinin ortalaması raporlanmıştır. Naive Bayes modeli 2 veri kümesinde en iyi performansı gösterirken, doğrusal model ve karesel model 1'er veri kümesinde en iyi performansı göstermişlerdir. Bütün veri kümelerini doğru olarak açıklayabilen yalnızca tek bir modelin olmadığından yola çıkarak bu sonuçların beklendiğini söyleyebiliriz. Veri kümelerinin karmaşıklık yapılarına en uygun olan modellerin en iyi performansı verdiği yorumunu yapabiliriz.

Performans kriterlerinden pd değerinin 4 veri kümesi üzerinden ortalaması %76 ve pf değerinin ortalaması %22 olarak gözlemlenmiştir. Bunun anlamı, oluşturulan modellerin %76 oranında hata yakalama kabiliyetine sahipken, %22 oranındada yanlış ikaz verme ihtimalleri olmasıdır. Özellikle PC2 veri kümesinde 5589 modül içinde hatalı olan yaklaşık 34 modülü yakalamakta %72 performans gösterilmesi dikkat çekicidir. Bu sonuçlar literatürde raporlanan en iyi sonuçlar ile kıyaslanabilir seviyededir [3].

4. Tartışma ve Gelecek Çalışmalar

Bu çalışmada, yazılım hatalarının kestirilmesi bir sınıflandırma problemi olarak ele alınmıştır. Literatürde en iyi sonuçları verdiği raporlanan Naive Bayes sınıflandırıcısının, ölçütlerin bağımsızlığı varsayımı çok değişkenli normal dağılım kullanılarak ortadan kaldırılmıştır. Bunun sonucunda ortaya çıkan doğrusal ve karesel modeller Naive Bayes ile kamuya açık veri kümeleri üzerinde kıyaslanmıştır. Deney sonuçları doğrusal ve karesel modellerin Naive Bayes modelinden daha iyi performns verebildiğini göstermiştir. Tüm modeller girdi olarak kaynak kod ölçütlerini kullanmışlardır. Kaynak kod ölçütlerinin avantajı otomatik olarak toplanabilmeleridir. En iyi sonuçları verecek ölçüt alt kümesini aramak yerine, PCA kullanılarak ölçütlerin doğrusal kombinasyonlarıyla yeni ölçütler çıkarılmıştır.

Bu çalışmanın sonucunda ortaya konulan modellerin, yazılım endüstrisinde faaliyet gösteren firmalara, yazılım test planlarını oluştururken, test kaynakları yönetimi açısından yardımcı olmaları amaçlanmıştır. Bundan sonra sürdüreceğimiz çalışmalarımızda, Naive Bayes sınıflandırıcısının diğer varsayımını olan, ölçütlerin eşit önemde oldukları varsayımını ortadan kaldırmak üzere araştırmalara yönelmeyi planlamaktayız.

5. Teşekkür

Bu çalışma Boğaziçi Üniversitesi Bilimsel Araştırma Projeleri Fonu tarafından sağlanan BAP-06HA104 numaralı fon ile desteklenmektedir.

6. Kaynakça

[1] N. Nagappan, L. Williams, J. Osborne, M. Vouk, P. Abrahamsson, "Providing Test Quality Feedback Using Static Source Code and Automatic Test Suite Metrics", *International Symposium on Software Reliability Engineering*, 2005.

[2] B. Boehm ve V.R. Basili, "Software Defect Reduction Top 10 List", IEEE Computer, Vol. 34, No. 1, Ocak 2001, s. 135-137.

[3] T. Menzies, J. Greenwald, ve A. Frank, "Data mining static code attributes to learn defect predictors", *IEEE Transactions on Software Engineering*, 33(1), 2007, s. 2–13.

[4] S.R. Chidamber ve C.K. Kemerer, "A Metrics Suite for Object-Oriented Design", *IEEE Tans. Software Engineering*, vol. 20, no. 6, Haziran 1994, s. 476-493.

[5] T. Menzies, Z. Chen, J. Hihn, ve K. Lum, "Selecting Best Practices for Effort Estimation", *IEEE Transactions on Software Engineering*, Vol. 32, No. 11, 2006, s. 883–895.

[6] Guyon ve Elisseff, "An introduction to variable and feature selection", *Journal of Machine Learning Research*, 3, 2003, s. 1157–1182.

[7] N. E. Fenton ve M. Neil, "A critique of software defect prediction models", *IEEE Transactions. on Software. Engineering*., 25(5), 1999, s. 675–689

[8] I. T. Jolliffe, , *Principal Component Analysis*, Springer-Verlag, New York, 1986.

[9] E. Alpaydin, *Introduction to Machine Learning*, The MIT Press, Ekim 2004.

[10] NASA/WVU IV&V Facility, Metrics Data Program, internet adresi http://mdp.ivv.nasa.gov.