Yazılım Maliyet Tahmininde İşlev Puanı Analizi ve Yapay Sinir Ağları Kullanımı

Mesut Keskin, Gülfem Işıklar Alptekin

Galatasaray Üniversitesi, Bilgisayar Mühendisliği Bölümü, İstanbul
mesutkskn@gmail.com,gisiklar@gsu.edu.tr

Özet. Yazılım maliyet tahmini, proje yöneticilerini her daim zorlayan işlerin başında gelmiştir. Yapılan tahmini gerçek değere yaklaştırmak, yazılım geliştirme süreci boyunca süre ve bütçe kısıtlarını daha iyi kontrol edebilmek demektir. Akademik yazında, her birinin kendine has olumlu veya olumsuz yönleri olan birden fazla tahmin yöntemi önerilmiştir. Bu makalede, özellikleri iyi bilinen bir yazılım projesinin maliyet tahmini öncelikle işlev puanı analizi (İPA) yöntemiyle gerçekleştirilmiştir. İPA'nin başlıca dezavantajları çok sayıda öznel değerlendirme içermesi ve öğrenme eğrisinin oldukça uzun olmasıdır. Ardından, aynı maliyet tahmini, yapay sinir ağları (YSA) kullanılarak yinelenmiştir. Gerçek değerlerle karşılaştırıldığında, tahminlerdeki doğruluk %90'ın üzerinde çıkmıştır. Deneysel çalışmalar sayesinde, YSA kullanılarak, gerçek sonuçlara oldukça yakın değerler bulunabileceğine inanılmıştır.

Anahtar Sözcükler Yazılım Maliyet Tahmini, Yapay Sinir Ağları, İşlev Puanı Analizi, Yazılım Proje Yönetimi.

1 Giriş

Yazılım maliyet tahmini, yazılım mühendisleri için en karmaşık problemlerden biridir. Maliyet tahmini, yazılımın geliştirilmesi sürecinde gerekecek toplam maliyetin tahmin edilmesi işlemidir. Yazılım projesinin planlanmasını ve ayrıntılı şekilde izlenmesini gerektirir. Global yazılım pazarının, 2018'de 32.4 milyar $'ı bulması beklenmektedir. Bu büyümeye paralel olarak yazılım geliştirme maliyetleri de artmaktadır. Standish Group Chaos raporuna göre, 2015 yılında gerçekleştirilen projelerin %52'si bütçe, süre veya kaynak kısıtlarını aşarak, %19'u ise tamamen başarısız şekilde sonuçlanmıştır. Büyük tahmin hatalarını engellemek amaçlı önerilen yöntemler başlıca iki sınıfa ayrılırlar: Algoritmik yöntemler (ör: Constructive Cost Model (COCOMO), Putnam yöntemi, işlev puanı analizi, doğrusal modeller, vs.) ve algoritmik olmayan yöntemler (uzman görüşü, analoji yöntemiyle tahmin, kazanmak için fiyatlama, Parkinson yasası, makine öğrenmesi yöntemleri, vs.) Algoritmik yöntemlerde, geçmiş veriler ve deneyimlerle oluşturulan matematik denklemleri kullanılır. Algoritmik olmayanlarda ise,

benzer geçmiş projelerden faydalanılır. Son dönemde yapılan bir çalışmada, global yazılım geliştirme projelerinin %28.5'unda işlev puanı analizi, %14.2'sinde sinir ağları yaklaşımı kullanıldığı belirtilmiştir [1]. Tüm bu değişik yaklaşımlar arasında, en çok kullanılanı COCOMO'dur [2][3]. Regresyon esaslı olan bu tahmin yöntemiyle, süre ve efor tahmini de yapılabilir. COCOMO seçilmiş 63 yazılım projesinin analizi sonucu ortaya çıkarılmıştır. En bilinen kısıtı, yazılım geliştirme hayat döngüsünün ilk evrelerinde yapılan proje büyüklüğü tahmininin çok belirsiz olmasıdır. Ayrıca, yöntem bu 63 projeye göre önerildiği için, tekrar ayarlamasının yapılması gereklidir. Bu bildiride, kesinliği arttırma amacıyla, yapay sinir ağları yöntemi içine COCOMO II'nin parametreleri dahil edilmiştir. Algoritmik olmayan yöntemler arasında yer alan makine öğrenmesi yöntemleri, başlıca iki sınıfa ayrılabilir: Sinir ağları ve bulanık yöntemler. Bir yapay sinir ağı sistemi, birden fazla yapay nöron ve aralarındaki ilişkilere bağlı ağırlıklardan oluşur. Geri yayılım (*back propagation*) algoritması, ağın çıktılarını gerçek çıktılarla karşılaştıran ve yazılım maliyet tahmininde kullanılmak üzere uygun olan bir algoritmadır [4]. Bildiride kullanılan iki yöntemin uygulanması için, ayrıntıları ve içeriği iyi bilinen bir yazılım projesi olan 'E-Bursum' seçilmiştir. İP analizinin gerçeğe yaklaşımı %74-%97 aralığında bulunurken, YSA ile elde edilen değerler %92-%98 aralığında olmuştur.

2 Kullanılan Yazılım ve İşlev Puanı Analizi

İPA, A.J. Albrecht tarafından, kod satır sayısı (*Lines of Code*-LOC) yaklaşımına alternatif olarak önerilen, akademik yazında üzerinde çok çalışılan bir maliyet yaklaşımıdır [5]. Albrecht, işlev/ fonksiyon puanlarının satır sayısından çok daha belirleyici olduğuna inanmaktadır. İPA, yazılımdaki fonksiyonları karmaşıklıkları ve yaptıkları işlere göre sınıflandırıp saymaktadır. Bunu yapmak, yöneticilerin verimliliği takip edebilmelerini ve yazılım geliştirme maliyetlerini tahmin edebilmelerini sağlamaktadır. Yöntemin ilk aşamasında, UFP (*Unadjusted Function Points*) hesaplanmaktadır. Ardından bu değerler, VAF (*Value Adjustment Factor*) kullanılarak İP (Fonksiyon/İşlev Puanı) hesaplamakta kullanılacaktır [6]. Kullanılan yazılımdaki bileşenler, karmaşıklık seviyelerine göre beş sınıfa ayrılmıştır (Tablo 1). Projedeki her dosyanın, Tablo 1'deki verilere göre hesaplanan bir UFP değeri mevcuttur. UFP değeri, aşağıdaki denklem uyarınca hesaplanır:

$$UFP = [\,|\,Harici\ girdi\,|\,*\,w_1] + [\,|\,Harici\ çıktı\,|\,*\,w_2] + [\,|\,Harici\ sorgu\,|\,*\,w_3] + [\,|\,Dahili\ dosya\,|\,*\,w_4] + [\,|\,Harici\ arayüz\,|\,*\,w_5] \quad (1)$$

Öğrencilere burs bulmak/atamak için kullanılan 'E-Bursum' projesinin başlıca üç modülü bulunmaktadır: Finans, çevrim içi akıl hocalığı (*mentoring*), öğrenci işleri. Tablo 2 her modülün bileşenleri karmaşıklıklarına göre özetlemiştir.

Tablo 1. Bileşenlerin karmaşıklıklarına göre sınıflandırılması

		Düşük	Orta	Yüksek
1	Harici girdi	3	5	6
2	Harici çıktı	4	6	7
3	Harici sorgu	3	5	6
4	Dahili dosya	7	13	15
5	Harici arayüz	5	9	10

Tablo 2. Modüler, bileşenler ve karmaşıklıkları

Modül 1: Bileşenler		**Modül 2: Bileşenler**		**Modül 3: Bileşenler**	
Giriş ekranı	Düşük	Kayıt ekranı	Düşük	Kayıt ekranı	Düşük
Burs başvurusu	Yüksek	Bireysel test	Yüksek	Blog	Orta
Profil bilgileri	Düşük	Mentor profili	Düşük	Sponsor profili	Düşük
Başvuru ekranı	Yüksek	Öğrenci profili	Düşük	Öğrenci profili	Düşük
Eski öğrenciler	Düşük	Eski mentorler	Düşük	Belge indirme	Yüksek
Duyurular	Orta	Duyurular	Orta	Burs başvurusu	Yüksek
Sohbet	Orta	Sohbet	Orta	Sohbet	Orta
		Mentor bulma	Yüksek	News part	Orta
				Çevrim içi mentor	Yüksek
				Raporlar	Orta

Üç modüle ait UFP değerleri sırasıyla, UFP_1, UFP_2 ve UFP_3 olarak tanımlanmıştır.

$$UFP_1 = [(1*3)+(2*5)+(2*6)]+[(2*4)+(2*6)+(1*7)]+ [(1*5)]+[1*13] = 70$$
$$UFP_2 = [(1*3)+(2*5)+(2*6)]+[(3*4)+(2*6)+(1*7)] +[1*13] = 69$$
$$UFP_3 = [(1*3)+(2*5)+(2*6)]+[(2*4)+(2*6)+(1*7)]+[1*5] = 57$$

UFP değerlerinin hesaplanmasının ardından, VAF değerleri yardımıyla son işlev puanları (İP) hesaplanabilir. VAF, 0 (en düşük) ve 5 (en yüksek) seviyeleri arasında değerlendirilen 14 genel sistem özelliğinden oluşmaktadır. Bu 14 değerin toplamı toplam etki derecesini (*Total Degree of Influence*-TDI) vermektedir.

Tablo 3. VAFs – Genel Sistem Özellikleri (GSCs) Değerlendirmesi [7]

		Modül 1	**Modül 2**	**Modül 3**
1	Sistem güvenilir yedekleme ve kurtarma gerektiriyor mu?	5	2	5
2	Veri iletişimi gerekiyor mu?	3	3	5
3	Dağıtık fonksiyon var mı?	3	2	3
4	Performans çok önemli mi?	3	4	5
5	Sistem, yaygın kullanılan bir işletim sistemi ortamında mı çalışacak?	4	2	4
6	Sistem çevrim içi veri girişi gerektiriyor mu?	3	2	5
7	Çevrim içi veri girişi, giriş işlemlerinin birden fazla ekran ya da işlem üzerinden olmasını mı gerektiriyor?	3	2	4
8	Ana dosyalar çevrim içi mi güncelleniyor?	4	2	5
9	Girdiler, çıktılar, dosyalar ve sorgular karmaşık mı?	1	1	3
10	Kod yeniden kullanılabilir olarak mı tasarlanmış?	3	3	4
11	İç süreç karmaşık mı?	3	2	3
12	Dönüşüm ve kurulum tasarım içinde mi?	1	3	2
13	Uygulama değişik kurumlarda birden fazla kurulum gerektirecek şekilde mi tasarlanmış?	4	1	3
14	Uygulama kullanıcı tarafından kolaylıkla kullanılması ve değiştirebilmesi üzerine mi tasarlanmış?	4	5	5

Üç modülün 14 GSC açısından değerlendirmesi Tablo 3'de yapılmıştır. Sonraki adımda üç modüle ait TDI değerleri, TDI_1, TDI_2, ve TDI_3, hesaplanmıştır.

$$TDI = \sum_{i=1...14} Cevap_i \quad (2)$$

Buna göre $TDI_1 = 44$, $TD_2 = 34$, $TD_3 = 56$.

Teknik bakış açısıyla yazılımın boyutunu ölçen etmen olan *Technical Complexity Factor* (TCF) denklem (3) kullanılarak hesaplanabilir:

$$TCF = 0.65 + 0.01 * TDI \quad (3)$$

Üç modüle ait TCF değerleri şu şekildedir:

$TCF_1 = 0.65 + 0.01 * 44 = 1.09$, $TCF_2 = 0.65 + 0.01 * 34 = 0.99$,

$TCF_3 = 0.65 + 0.01 * 56 = 1.21$

Son işlev puanı UFP ve TCF değerleri kullanılarak hesaplanır:

$$İP = UFP * TCF \quad (4)$$

Buna göre: $İP_1 = 76.30$, $İP_2 = 68.31$, $İP_3 = 68.97$.

'E-Bursum' yazılımın gerçek satır sayısı (LOC) bilindiği için, yaklaşımın gerçeğe ne kadar yaklaştığı ölçülebilmiştir (Tablo 4). Bunun için, kullanılan programlama dili ve işlev puanlarını birleştiren denklem (5) kullanılmıştır:

$$LOC = İP * PHP\ dilinin\ katsayısı \quad (5)$$

Buna göre $LOC_1 = 5112.1$, $LOC_2 = 4576.77$, $LOC_3 = 4620.99$.

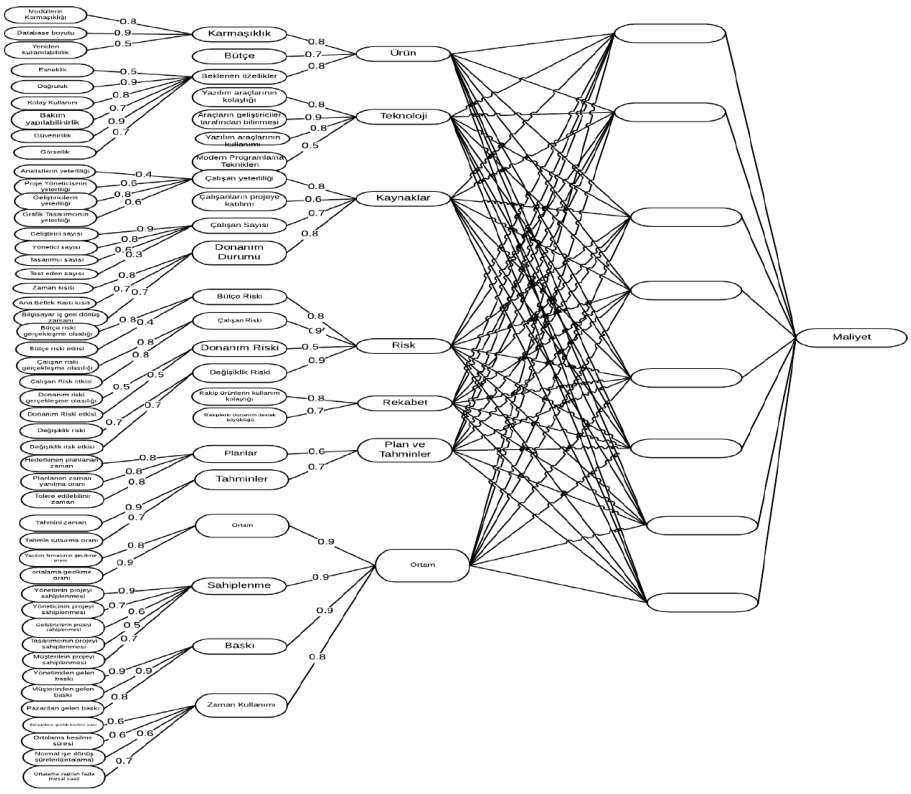
Tablo 4. İPA sonuçları ve gerçek değerlerin karşılaştırılması

	İPA'nin tahmin ettiği satır sayısı	Gerçek satır sayısı	Yaklaşık
Modül 1	5112.1	5280	96.80%
Modül 2	4576.77	5620	81.44%
Modül 3	4620.99	6228	74.19%

Proje büyüklüğünü/maliyetini satır sayısı parametresiyle tahmin etmek yaygın kullanılan bir yöntemdir. Satır sayısı esaslı tahminler, geçmiş tecrübelere dayandığı için, çoğu zaman bilimsel olmamakla eleştirilmektedirler. İPA, sade bir tahmin yöntemi olduğu için, programlama bilgisi az olan kişiler tarafından bile kullanılabilmektedir. Kodun büyük kısmının girdi ve çıktılardan oluşan tip projelere daha uygundur. Ancak, projeler her zaman böyle değildir. Buna bir çözüm olarak, İPA'ne bir ek olarak *Feature Point* yaklaşımı ortaya atılmıştır [8]. Bu yaklaşım, girdi ve çıktılarla birlikte, kullanılan algoritmaları da göz önüne alarak toplam karmaşıklığı hesaplamaktadır.

3 Yapay Sinir Ağları Yaklaşımı

Sinir ağlarını eğitmeye yarayan birden fazla algoritma olsa da, geri yayılım (*back propagation)* algoritması kullanılmıştır. YSA'daki her katman nöronlar ve bunlar arasındaki ilişkilerden oluşmaktadır [9]. Nöronlar arasındaki ağırlıklar öğrenme sürecinde kullanılmaktadır. Geri yayılım algoritması, girdiler ve elde edilmesi istenen çıktıları alarak, ağırlıkları bu iki kümeyi birbirine yaklaştırmak için ayarlar [10]. YSA yaklaşımının performansını arttırmada seçilen parametre kümesinin rolü büyüktür. Parametre kümesi seçimi için öncü çalışmalardan birinde, COCOMO'nun ölçüt kümesi alınmıştır [11]. Önerdiğimiz çalışmada, COCOMO II ve [12]'de önerilen parametre kümesi üzerine, görsel kalite ve rekabet gibi iki parametre daha eklenmiştir. Önerilen sinir ağı Şekil 1'de gösterilmiştir. YSA yaklaşımı sonuçları, gerçeğe daha yakın sonuçlar elde edilmesini sağlamıştır (Tablo 5).



Şekil 1. Önerilen yapay sinir ağı topolojisi

Tablo 5. YSA sonuçları ve gerçek değerlerin karşılaştırılması

	YSA'nın tahmin ettiği satır sayısı	Gerçek satır sayısı	Yaklaşık
Modül 1	5158.5	5280	97.70%
Modül 2	5355.8	5620	95.30%
Modül 3	5751.5	6228	92.35%

4 Sonuç ve Öneriler

Bu bildiride, 'E-bursum' yazılımına ilk önce işlev puanı analizi uygulanmıştır. Elde edilen sonuçlar tatmin edici bulunmuştur. İşlev puanı analizinin avantajı, maliyet tahmininin yazılım gereksinim analizi sırasından itibaren yapılabilecek olması, uygulama araçlarından, yöntemlerinden veya programlama dillerinden bağımsız olması ve teknik dünyaya çok yakın olmayan kullanıcıların bile uygulayabilecek olmasıdır. İPA sonrasında yöntem olarak, tahmin işinin daha az öznel ve otomatik şekilde yapılabilmesini sağlayabilecek makine öğrenmesi yöntemlerinden biri olan sinir ağları seçilmiştir. Projenin büyüklüğü ve karmaşıklığı arttıkça, İPA'nin performansı da düşmektedir. YSA yöntemi, gerçeğe oldukça yakın tahminler vermiştir. Yazılım yaşam döngüsünün ilk evrelerinde bile, gerçekçi tahmin yapmaya izin verebilecektir. Sinir ağında parametre

olarak seçilen ölçütler, istendiği takdirde değiştirilebilir, ekler yapılabilir. Makine öğrenmesi yöntemlerinin, kullandıkları veri sayısı arttıkça gerçeğe daha yakın tahminler vermesi beklenir. Şirketler maliyet tahmininde bu yöntemi uyguladıkça, tahmin performansının artacağına inanılmaktadır. Bir sonraki çalışmada, kurulan bu yapay sinir ağını, bilinen başka birkaç yazılım projesine daha uygulanıp verdiği sonuçlara bakılacaktır.

5 Destek

Bu araştırma, 15.401.005 numaralı Galatasaray Üniversitesi Bilimsel Araştırma Projesi tarafından finansal olarak desteklenmektedir.

6 Kaynaklar

1. M.E. Bajta, A. Idri, J.L. Fernandez-Aleman, J.N. Ros, A. Toval, "Software cost estimation for global software development – A systematic map and review study", *International Conference on Evaluation of Novel Approaches to Software Engineering*, 2015, pp. 197-206.
2. I. Attarzadeh, S.H. Ow, "A novel soft computing model to increase the accuracy of software development cost estimation", *International Conference on Computer and Automation Engineering*, vol. 3, 2010, pp. 603-607.
3. B.W. Boehm, *Software Engineering Economics*. 1981, Englewood Cliffs, NJ.
4. V. Khatibi, D.N.A. Jawawi, "Software cost estimation methods: A review", Journal of Emerging Trends in Computing and Information Sciences, vol. 2(1), 2010, pp. 21-29.
5. A. J. Albrecht, "Measuring Application Development Productivity", Joint SHARE, GUIDE, and IBM Application Development Symposium, Monterey, California, IBM Corporation (1979), pp. 83–92.
6. R. Meli, L. Santillo, "Function point estimation methods: A comparative overview", FESMA, vol. 99, 1999, pp. 6-8.
7. L. Arthur, Measuring Programmer Productivity and Software Quality. Wiley-Interscience, 1985.
8. C. Jones, *The SPR Feature Point Method*. Software Productivity Research Inc., 1986.
9. A.R. Venkatachalam, "Software cost estimation using artificial neural networks", *International Joint Conference on Neural Networks,* 1993, pp. 987-990.
10. D.S. Touretzky, D.A. Pomerleau, "What's hidden in the hidden layers?", *Byte*, August 1989, pp. 227-233.
11. G. Wittig, G. Finnie, "Estimating software development effort with connectionist models", *Information and Software Technology*, vol. 39, 1997, pp. 469-476.
12. O. Kalıpsız, M. Ayyıldız, "Yazılım Geliştirme Projelerinde Maliyet Tahminleme Çalışmasında Kullanılabilecek Bir Ölçev Kümesi ve Bir Yapay Sinir Ağı Topolojisi Önerisi", 2009, YTU.