Avrupa Bilim ve Teknoloji Dergisi
Özel Sayı 22, S. 43-48, Ocak 2021

Araştırma Makalesi

www.ejosat.com ISSN:2148-2683

European Journal of Science and Technology
Special Issue 22, pp. 43-48, January 2021

Research Article

Yazılım Çaba Tahmininde Yapay Sinir Ağları İçin Optimum Yapının Belirlenmesi

Mehmet Kayakuş[1*]

[1*] Akdeniz Üniversitesi, Manavgat Sosyal ve Beşeri Bilimler Fakültesi, Yönetim Bilişim Sistemleri Bölümü, Antalya, Türkiye (ORCID: 0000-0003-0394-5862)

(İlk Geliş Tarihi Aralık 2020 ve Kabul Tarihi Ocak 2021)

(**DOI:** 10.31590/ejosat.847712)

ATIF/REFERENCE: Kayakuş, M. (2021). Yazılım Çaba Tahmininde Yapay Sinir Ağları İçin Optimum Yapının Belirlenmes. *Avrupa Bilim ve Teknoloji Dergisi*, (22), 43-48.

Öz

Bir yazılım projesinin çabasını tahmin etmek projenin yönetimi ve başarısı için önem teşkil etmektedir. Bu çalışmada, yazılım çaba tahminini gerçekleştirmek için yapay zekâ tekniklerinden yapay sinir ağları yöntemi kullanılmıştır. Çalışmada veri seti olarak iyi bilinen ve bu çalışmalarda sıklıkla kullanılan NASA proje veri seti kullanılmaktadır. Veri sayısının az olmasından dolayı 10 katmanlı çapraz doğrulama yöntemi kullanılmıştır. Veri seti rastgele 10 farklı gruba ayrılmış; gruplardan biri eğitim amaçlı kullanılırken geri kalanı test amaçlı kullanılmıştır. Her grup için modelde bu işlem tekrarlanarak tüm veri hem eğitilmiş hem de test edilmiştir. Böylece modelin doğruluğu arttırılmıştır. Yapay sinir ağ modelinde, geliştirme satırı ve metodoloji olmak üzere iki giriş değişkeni; çıkış değişkeni olarak yazılım çabası kullanılmıştır. Yapay sinir ağ tasarımında gizli katman sayısını ve nöron sayısı modelin başarısını etkilemektedir. Bu çalışmada 20 farklı YSA modeli geliştirilerek en başarılı model belirlenmiştir. Çalışma sonucunda R2 0,926, RMSE 0,078, MSE 0,006 ve MAE 0,058 olan 2 gizli katman ve 2 nörondan oluşan model en başarılı model olmuştur. Modeller arasında en başarılı sonucu veren model ile en başarısız modelin R2 değerleri arasında %55 fark bulunmaktadır. Bu sonuçlar yazılım çaba tahmini için parametre seçiminin önemini göstermektedir.

Anahtar Kelimeler: Yazılım çabası tahmini, Yazılım mimarileri, Yapay sinir ağları, Çapraz doğrulama, Gizli katman.

The Determination of Optimum Structure for Artificial Neural Networks in Software Effort Estimation

Abstract

Estimating the efforts of a software project possesses a great importance for the management and success of the project. In this study, artificial neural networks method, one of the artificial intelligence techniques, was used to fulfil software effort estimation. The NASA project data set, which is well known as the data set and is frequently used in these studies, is utilized in the study. Due to the low number of data, a 10-layer cross-validation method was used. The data set was randomly divided into 10 different groups. While one of the groups is used for training; others are used for testing. This process is repeated for each group in the model, and all data are both trained and tested. Thus, the accuracy of the model is increased. In the artificial neural network model, development line and methodology as input variables and software effort as output variables are used. In artificial neural network design, the number of hidden layers and neurons affect the success of the model. In this study, 20 different ANN models were developed, and the most successful model was determined. As a result of the study, the model consisting of 2 hidden layers and 2 neurons, R2 0.926, RMSE 0.078, MSE 0.006 and MAE 0.058, became the most successful model. There is a 55% difference between the R2 values of the most successful model and the least successful model. These results show the importance of parameter selection for software effort estimation.

Keywords: Software effort estimation, Software architectures, Artificial neural networks, Cross-validation, Hidden layer.

[*] Sorumlu Yazar: mehmetkayakus@akdeniz.edu.tr

1. Giriş

Yazılım, özellikle büyük ve karmaşık sistemler için sistem ediniminde, mühendisliğinde ve geliştirmede giderek daha önemli bir rol oynamaktadır. Bu tür sistemler için, yazılım maliyetlerinin doğru tahmin edilmesi etkili program yönetiminin kritik bir parçasıdır (Borade & Khalkar, 2013). Müşteri ve yönetim baskısı, uzman kararına dayanan eski yazılım değerlendirme yöntemleri hakkında sınırlı ve yanlış bilgi, aşırı iyimser tahminler yazılım maliyetlerinin yanlış tahmin edilmesi yönünde risk oluşturmaktadır. Bu etkenler proje sonuçlarının belirli bir zaman diliminde, bütçede ve kabul edilebilir kalitede teslim edilmesini ciddi şekilde olumsuz etkileyebilmektedir (Alami, 2016; Mieritz, 2012; Spalek, 2005; Tan, 2011).

Proje kaynaklarının tahmin edilmesi, yazılım projesi geliştirme dahil, proje yönetiminde kritik bir adımdır (Jorgensen & Shepperd, 2006). Bir yazılım projesinin maliyetini veya çabasını tahmin etme yeteneği, herhangi bir projeyi kabul etme veya reddetme yönündeki yönetim kararında doğrudan bir etkiye sahiptir. Örneğin, yazılım maliyetlerinin olduğundan yüksek tahmin edilmesi kaynak israfına ve yetersiz teslimat süresine yol açarken, eksik tahmin proje ekibinin eksik çalışmasına, bütçeleme giderlerinin fazla olmasına ve teslimat süresinin gecikmesine neden olabilmektedir (Azzeh, Nassif, & Banitaan, 2017; Heemstra, 1992).

Yazılım çaba tahmini için çeşitli tahmin yöntemleri geliştirilmiştir. COCOMO, onun güncel versiyonu COCOMO II, fonksiyon noktası (function point) ve makine öğrenimi yöntemlerine dayalı modeller bu yöntemlerden bazılarıdır. Makine öğrenmesi yöntemlerini kullanarak yazılım çabasını tahmin eden birçok çalışma bulunmaktadır. Srinivasan ve arkadaşları 15 projeden oluşan bir veritabanı üzerinde yazılım çaba tahminini tahmin etmek için yapay zekâ tekniklerinden geri yayılımlı yapay sinir ağları ve Classification and Regression Trees (CART) yöntemini kullanmışlardır (Srinivasan & Fisher, 1995). Przemyslaw ve arkadaşları ISBSG veri seti üzerinde çapraz doğrulama ve üç makine öğrenmesi yöntemini (Destek vektör makineleri, yapay sinir ağları ve lineer model) kullanarak yazılım çaba ve süre tahmini gerçekleştirmek için modeller gerçekleştirmişlerdir (Pospieszny, Czarnacka-Chrobot, & Kobylinski, 2018). Braga ve arkadaşları yazılım proje geliştirme çabasının tahmini için kullandıkları regresyon yöntemlerinin performansını iyileştirmek için bagging yöntemini kullanmışlardır. Çalışmalarında kullandıkları regresyon ağacı, destek vektör makineleri ve çok katmanlı algılayıcıları yöntemlerinin başarısını arttırmak, tahminle ilişkili varyansı azaltmak ve tahmin sürecini iyileştirmek için bagging yöntemini çalışmalarına ilave etmişlerdir (Braga, Oliveira, Ribeiro, & Meira, 2007). Nassif ve arkadaşları yazılım çabasını tahmin etmek için yeni bir yapay sinir ağı modelini önermişlerdir. Bu modelin girdileri yazılım boyutu, üretkenlik ve karmaşıklık iken çıktı tahmin edilen yazılım çabasıdır (Nassif, Capretz, & Ho, 2012). Attarzadeh ve arkadaşları yazılım geliştirmenin erken aşamalarında daha doğru yazılım tahminleri sağlamak için Constructive Cost Model (COCOMO), ANN-COCOMO II içeren yeni bir yapay sinir ağı tahmin modeli önermişlerdir. Bu model, COCOMO modelinin özelliklerini korurken, öğrenme yeteneği ve iyi yorumlanabilirlik gibi yapay sinir ağlarının avantajlarını kullanmaktadır. Elde edilen sonuçların analizi yapıldığında, orijinal COCOMO II'ye göre ANN-COCOMO II modeli tahmin doğruluğunda %8,36'lık bir iyileşme gerçekleştirmiştir (Attarzadeh, Mehranzadeh, & Barati, 2012). Dan çalışmasında yazılım çaba tahmininde doğruluğu iyileştirmek için parçacık sürüsü optimizasyonuna dayalı yapay sinir ağı modelini kullanmıştır. Modifiye model yapay sinir ağının yakınsama hızını artırmakta ve ağın ilk ağırlıklarına yüksek bağımlılığı olan yapay sinir ağının öğrenme yeteneği sorununu çözmektedir. Modifiye modeli doğrulamak için iki veri seti (COCOMO I ve NASA93) üzerinde testler yapılmış; bu testler sonucunda PSO-ANN-COCOMO II'nin orijinal yapay sinir ağına göre yazılım çaba tahminlerini %3,27'lik bir artışla tahmin ettiği görülmüştür (Dan, 2013). Tronto ve arkadaşları yazılım çaba tahmini için akıllı yapay sinir ağları ve regresyon olmak üzere iki model incelemişlerdir. Her iki yöntemin performansını karşılaştırdıklarında yapay sinir ağlarının çaba tahmininde daha etkili olduğunu sonucuna ulaşmışlardır (de Barcelos Tronto, da Silva, & Sant'Anna, 2007). Heiat çalışmasında yazılım geliştirme çabasını tahmin etmek için yapay sinir ağı ve regresyon modelleri geliştirmiş ve karşılaştırmasını yapmıştır. Çok katmanlı algılayıcı ve radyal tabanlı işlevli sinir ağlarının ortalama mutlak yüzde hatası açısından geleneksel regresyon analizine göre gelişmiş performans ürettiğini sonucuna ulaşmıştır (Heiat, 2002). Shan ve arkadaşları genetik programlama kullanarak yazılım projesi çaba tahmini gerçekleştirmişlerdir. Genetik programlama tekniği kullandıkları çalışmaları basit doğrusal regresyon ile elde edilen sonuçlara göre önemli ölçüde daha iyi olduğu görülmüştür (Shan, McKay, Lokan, & Essam, 2002). Baskeles ve arkadaşları yazılım geliştirme tahmini için dört farklı makine öğrenmesi yöntemlerini kullanmışlardır: Geri yayılımlı çok katmanlı sinir ağları, regresyon ağaçları, radyal temel fonksiyonu (RBF) ve destek vektör regresyon (SVR) sonuçlarını karşılaştırmalı analiz etmişlerdir (Baskeles, Turhan, & Bener, 2007). Huang ve arkadaşları genetik algoritmayı gri ilişkisel analize entegre ederek yazılım çaba tahmin modeli geliştirmişlerdir.

Bu çalışmada NASA veri seti kullanılarak yazılım çaba tahmini gerçekleştirilmiştir. Veri setindeki verinin azlığından dolayı başarıyı arttırmak için çapraz doğrulama yöntemi kullanılmıştır. Tahmin yöntemi olarak yapay sinir ağları modeli kullanılmış; en başarılı modeli tespit etmek için 20 farklı model oluşturulmuştur.

2. Materyal ve Metot

2.1. Veri Seti

Bu çalışmada Tablo 1'de görülen halka açık ve iyi bilinen NASA yazılım projesi verileri kullanılmıştır. Bu veri kümesi geliştirme satırı ve metodoloji olmak üzere iki bağımsız değişkenden ve çabadan oluşan bir bağımsız değişkenden oluşmaktadır. Geliştirme satırı 1000 satır kod (KLOC-Kilo lines of code), çaba kişi-ay değeri ile ifade edilmektedir.

Tablo 1. Veri seti

Geliştirme satırı	Metodoloji	Çaba
90,2	30	115,8
46,2	20	96
46,5	19	79
54,5	20	90,8
31,1	35	39,6
67,5	29	98,4
12,8	26	18,9
10,5	34	10,3
21,5	31	28,5
3,1	26	7
4,2	19	9
7,8	31	7,3
2,1	28	5
5	29	8,4
78,6	35	98,7
9,7	27	15,6
12,5	27	23,9
100,8	34	138,3

Geliştirme satırı programın kaynak kod satır sayısını gösteren bir program ölçüsüdür. Geliştirme satırı hem yeni kaynak kodu satırlarını hem de yeniden kullanılan kodu dikkate alan bir ölçüdür. Yorumlarla birlikte geliştirilen kaynak kod satırlarının sayısı (yorum içeren yeni satırlar) artı yeniden kullanılan satırların %20'si ile verilir (Bailey & Basili, 1981). Geliştirme satırı kaynak kod satırları ile ölçülmektedir. Kaynak kod satırları (SLOC) programın kaynak kodunun metnindeki satır sayısını sayarak yazılım programının boyutunu ölçmek için kullanılan yazılım ölçüsüdür. Bu metrik boş satırları, yorum satırlarını ve kitaplığı saymaz. SLOC ölçüleri programlama diline bağımlıdır.

Metodoloji değişkeni, her yazılım projesinde kullanılan geliştirme metodolojileri dikkate alınarak hesaplanmaktadır. Metodoloji test plan kabulü, belgelendirme gibi geliştirme yöntemleri kullanılarak hesaplanan bir değişkendir.

Çaba bir kişinin belirli bir projeyi geliştirmek için ihtiyaç duyacağı ay sayısıdır ve kişi-ay birimlerinde ifade edilmektedir. Projenin başlangıcından bitimine kadar ay olarak ne kadar süre geçtiği ve projede kaç kişinin çalıştığını gösteren bir birimdir. Yazılımın maliyetine doğrudan etki eden bir faktördür. Yazılım çabasının doğru tahmin edilmesi projede beklenmeyen aksaklıkların önüne geçilmesi sağlayacaktır.

2.2. Çapraz Doğrulama

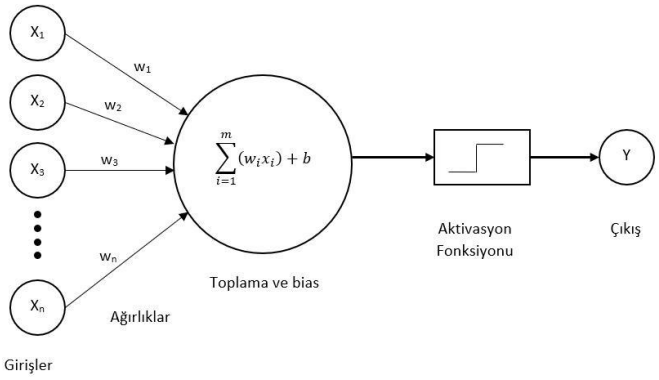
Çapraz doğrulama, sınırlı bir veri kümesinde makine öğrenimi modellerini değerlendirmek için kullanılan yeniden örnekleme tekniğidir. Modelin eğitimi sırasında kullanılmayan veriler üzerinde öngörülerde bulunmak için kullanıldığında modelin genel olarak nasıl performans göstermesi beklendiğini tahmin etmek için sınırlı bir örnek kullanmasıdır. K katmanlı ve birini dışarıda bırak çapraz doğrulama yöntemleri olmak üzere yaygın olarak iki kullanımı bulunmaktadır.

K katmanlı çapraz doğrulamada (k-fold cross validation) veri kümesi rastgele k tane gruba ayrılır. Gruplardan biri test için kullanılırken diğerleri eğitim seti olarak kullanılmaktadır. Her bir grup bu şekilde k defa tekrarlanarak modellenir. Böylece veri setindeki tüm veriler eğitilmiş olacaktır. 10-katmanlı çapraz doğrulamada veri kümesi 10 gruba ayrılmakta; tasarlanan model 10 kez eğitilip test edilmektedir. Böylece her grup test edilmişte olacaktır.

Birini dışarıda bırak yönteminde k değeri veri setindeki veri sayısına eşittir. Bu yöntemde sadece 1 gözlem verisi test verisi olarak ayrılır. Geriye kalan veriler ile model eğitilerek test verisi üzerinden değerlendirme yapılmaktadır (Şahinarslan, 2019).

2.3. Yapay Sinir Ağları

Yapay Sinir Ağları (YSA), biyolojik beyin mimarisini taklit etmeyi amaçlayan bir yapay zekâ dalıdır. Nöron adı verilen, birbiriyle bağlantılı doğrusal olmayan birçok işlem öğesinden oluşan paralel dağıtılmış bir sistemdir (Hecht-Nielsen, 1988; Lippmann, 1987; Viotti, Liuti, & Di Genova, 2002). Girişler olarak adlandırılan ve ayarlanabilir bağlantı ağırlıkları ile çarpılan sinyaller önce toplanır ve ardından çıktı üretmek üzere bir transfer fonksiyonundan geçirilir. Aktivasyon fonksiyonu, nöronun girişlerinin ağırlıklı toplamıdır ve en yaygın kullanılan transfer fonksiyonu da sigmoid fonksiyonudur(Agatonovic-Kustrin & Beresford, 2000). Temel bir YSA modeli Şekil 1'de görülmektedir.



Şekil 1. YSA Modeli

Geri bildirim, bir katmanın çıktısının önceki katmanın girişine veya aynı katmana geri döndüğü bir bağlantı türüdür. Bir ağdaki geri bildirim bağlantısının yokluğuna veya varlığına göre iki tür mimari tanımlanabilir. İleri beslemeli ağ mimarisi çıkıştan giriş nöronlarına bir bağlantı yoktur ve bu nedenle önceki çıkış değerlerinin kaydını tutmaz (Agatonovic-Kustrin & Beresford, 2000). Gizli nöronların sayısı değişebilmekte ve ideal sayı deneme yanılma yoluyla belirlenebilir; böylece hata en aza indirilebilir (Nassif et al., 2012).

Geri yayılım, hataların geriye doğru yayılması prensibine göre çalışan denetimli bir yapay sinir ağı algoritmasıdır. Geri yayılım algoritmasında çıktıdan giriş nöronlarına bağlantıları vardır. Ağa sunulan girdi ile ağın ürettiği çıktı karşılaştırılır ve aradaki fark hata olarak kabul edilir. Bu hatayı düşürmek için ağdaki ağırlıkların her biri güncellenmesi gerekmektedir. Bunu sağlamak için Gradient Descent optimizasyon algoritması kullanılmaktadır. Bu algoritmada yerel minimuna yakınsamak için birinci dereceden türevleri kullanılmaktadır.

2.4. Değerlendirme Yöntemleri

Tasarlanan yapay sinir ağları modellerinin başarısını ve hata varyansını ölçmek için düzeltilmiş belirleme katsayısı (R2), hata kareler ortalamasının karekökü (RMSE), ortalama kare hata (MSE) ve ortalama mutlak hata (MAE) kriterleri kullanılmıştır. Bu kriterlere ait denklemler aşağıda verilmiştir.

$$R^2 = 1 - \frac{\sum\left((y_i - x_i)\right)^2}{\sum(Y_i - Y_{ort})^2} \quad (1)$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - x_i)^2} \quad (2)$$

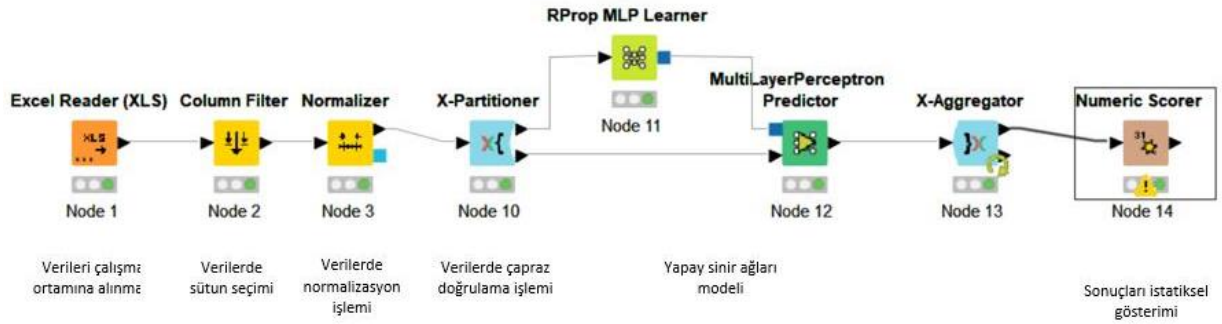
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - x_i)^2 \quad (3)$$

$$MAE = \frac{\sum_{i=1}^{n}|(y_i - x_i)|}{n} \quad (4)$$

Burada i veri sayısını, y gerçek değeri, x tahmin değerini göstermektedir. Bu kriterlere göre yüksek R2 ve en düşük RMSE, MSE ve MAE değerleri en başarılı modeli belirlemektedir.

3. Araştırma Sonuçları ve Tartışma

Çalışmada NASA veri seti kullanılarak yazılım çaba tahminini gerçekleştiren en başarılı yapay sinir ağları parametrelerinin belirlenmesi gerçekleştirilmiştir. Bunun için knime programının güçlü simülasyon özelliğinden faydalanılmıştır. Modellerin başarısını değerlendirmek için R2, RMSE, MSE ve MAE istatiksel bilgileri kullanılmıştır. Çalışmada tasarlanan Knime arayüzü Şekil 2'de görülmektedir.



Şekil 2. Knime'da yapay sinir ağları tasarımı

Çalışmada makine öğrenmesi yöntemlerinin performansının iyileştirmek ve doğruluk oranını arttırmak için normalizasyon yöntemi kullanılmıştır. Normalizasyon yöntemi olarak min-max, z-score, medyan ve sigmoid gibi yöntemler bulunmaktadır. Bu çalışmada min-max normalleştirme yöntemi kullanılarak orijinal veriler 0-1 aralığında doğrusal dönüşüm ile normalizasyon işlemi gerçekleştirilmiştir. Denklem 5'te min-max normalleştirme formülü verilmiştir.

$$x' = \frac{x_i - x_{min}}{x_{max} - x_{min}} \quad (5)$$

x' normalize edilmiş veriyi, xi girdi değerini, xmin girdi seti içerisinde yer alan en küçük sayıyı, xmax girdi seti içerisinde yer alan en büyük sayıyı göstermektedir.

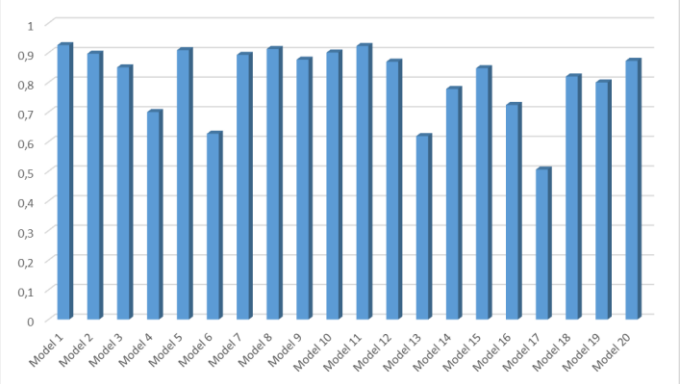
Farklı sayıda gizli katman ve nörondan oluşan toplam 20 tane yapay sinir ağı modeli tasarlanmıştır. Modellerde geri yayılım algoritması kullanılmış ve iterasyon sayısı 15000 olarak belirlenmiştir. Tablo 2'de yapay sinir ağ modellerinin başarı ve hata analizleri görülmektedir.

Tablo 2. Yapay sinir ağlarının başarı ve hata analizleri

Yapay Sinir Ağları Modeli			Başarı ve Hata Analizleri			
Model No	Gizli Katman Sayısı	Nöron Sayısı	R2	RMSE	MSE	MAE
Model 1	2	2	0,926	0,078	0,006	0,058
Model 2	2	3	0,897	0,107	0,011	0,073
Model 3	2	4	0,851	0,129	0,017	0,087
Model 4	2	5	0,700	0,183	0,033	0,124
Model 5	3	2	0,909	0,101	0,010	0,079
Model 6	3	3	0,627	0,204	0,041	0,143
Model 7	3	4	0,893	0,109	0,012	0,083
Model 8	3	5	0,913	0,099	0,010	0,077
Model 9	4	2	0,877	0,117	0,014	0,081
Model 10	4	3	0,901	0,105	0,011	0,082

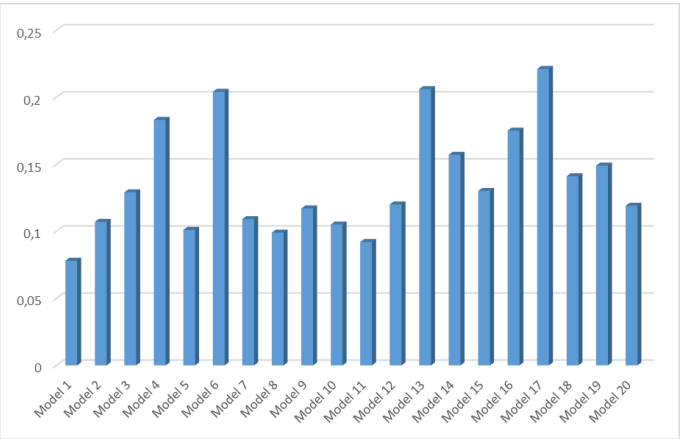
Model 11	4	4	0,923	0,092	0,009	0,072
Model 12	4	5	0,870	0,120	0,014	0,087
Model 13	5	2	0,619	0,206	0,042	0,143
Model 14	5	3	0,778	0,157	0,025	0,099
Model 15	5	4	0,848	0,130	0,017	0,091
Model 16	5	5	0,724	0,175	0,031	0,103
Model 17	6	2	0,506	0,221	0,049	0,157
Model 18	6	3	0,820	0,141	0,020	0,117
Model 19	6	4	0,800	0,149	0,022	0,106
Model 20	6	5	0,873	0,119	0,014	0,093

Modellerin R2 değerinin grafiksel gösterimim Şekil 3'te, RMSE gösterimi Şekil 4'te, MSE gösterimi Şekil 5'te, MAE gösterimi Şekil 6'da görülmektedir.



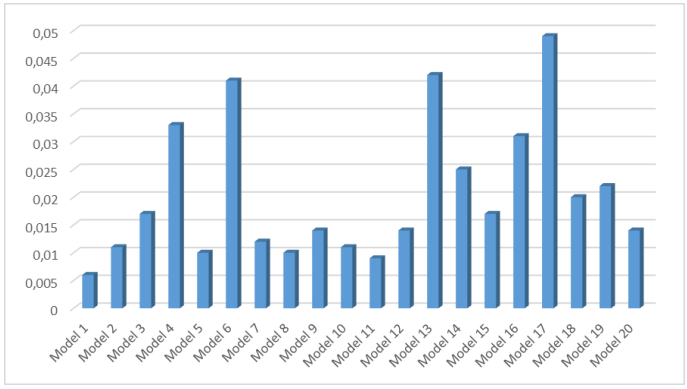
Şekil 3. R2 değerinin grafiksel gösterimi

Tablo 2 ve Şekil 3'te görüldüğü gibi R2 değerleri 0,506 ile 0,926 arasında değişmektedir. Ortalama RMSE değeri 0,813 olmuştur. En başarılı model 2 gizli katman ve 2 nörondan oluşan model 1 olmuştur. Verilerin doğrusal bir eğriye ne kadar iyi uyduğunu gösteren R2 değerinin 1 olması, test verilerinin doğrusal bir eğri sağlandığını göstermektedir. Model 1 değerinin istenen değerlere çok yaklaştığı ve doğrusal bir eğri sağlamaya yakın olduğu görülmektedir.



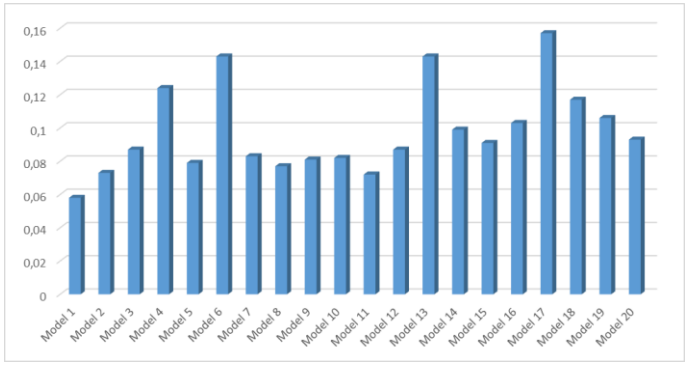
Şekil 4. RMSE değerinin grafiksel gösterimi

Şekil 4'te görüldüğü gibi RMSE değeri en düşük 0,078, en yüksek 0,221, ortalama 0,137 olduğu görülmektedir. Tahmin ettiği değerler ile gerçek değerleri arasındaki uzaklığın bulunmasında kullanılan RMSE değerinin sıfır olması modelin hiç hata yapmadığı anlamına gelir. Bu yüzden RMSE değerinin sıfır değerine yakın olması istenmektedir. Modellerin RMSE değerine bakıldığında en başarılı modelin 2 gizli katman ve 2 nörondan oluşan model 1 görülmektedir.



Şekil 5. MSE değerinin grafiksel gösterimi

Şekil 5'te MSE değerinin değişimi görülmektedir. Burada MSE değeri en düşük 0,006, en yüksek 0,049, ortalama 0,020 olmuştur. MSE değeri bir regresyon eğrisinin bir dizi noktaya ne kadar yakın olduğunu göstermek için kullanılmaktadır. Sıfırı yakın tahmin modellerinin daha başarılı söylenebilir. Modellerin MSE değerlerine bakıldığında 0,006 ile en başarılı modelin 2 gizli katman ve 2 nörondan oluşan model 1 olduğu söylenebilir.



Şekil 6. MAE değerinin grafiksel gösterimi

Şekil 6'da modellerin MAE değerleri görülmektedir. Grafikte MAE için en düşük düşük değerin 0,058, en yüksek 0,157 ve ortalama değerin 0,098 olduğu görülmektedir. MAE iki sürekli değişken arasındaki farkı gösteren bir büyüklüktür. Bir

başka deyişle gerçek değer ile veriye en iyi uyan çizgi arasındaki ortalama dikey mesafeyi göstermektedir. MAE değerinin düşük olması tahmin modelinin başarılı olduğunu göstermektedir. Bu çalışmada 0,058 ile en düşük MAE değerini 2 gizli katman ve 2 nörondan oluşan model 1 sağlamaktadır.

4. Sonuç

Bir yazılım projesinin başarılı yönetimi ve kontrolü için yazılım projesi çabasının doğru tahmini çok önemlidir. Bu çaba tahmini için çeşitli yöntemlerin geliştirilmesine yol açmıştır. COCOMO, onun güncel versiyonu COCOMO II, fonksiyon noktası (function point) ve makine öğrenimi yöntemlerine dayalı modeller bu yöntemlerden bazılarıdır.

Bu çalışmada makine öğrenmesi yöntemlerinden yapay sinir ağları kullanılarak yazılım çaba tahmini gerçekleştirilmiştir. Yazılım çaba tahmini probleminde, yazılımın boyutu ve metodoloji girdi değişkenleri olarak kullanarak bir regresyon modeli oluşturulmuştur. Çıktı hem programlama hem de yönetim faaliyetlerini göz önünde bulundurarak kişi-ay cinsinden toplam çabadır. Veri seti sayısının azlığından ve başarı doğruluğunu arttırmak için çapraz doğrulama yöntemlerinden 10 katmanlı çapraz doğrulama kullanılmıştır. Yapay sinir ağları giriş, gizli katman, gizli katmandaki nöronlardan ve çıkış katmanından oluşmaktadır. Gizli katman ve gizli katmandaki nöron sayısı modelin başarısına doğrudan etki etmektedir. Bu çalışmada en başarılı modeli tespit etmek için bir dizi test çalışması yapılmıştır. Farklı sayıda gizli katman ve nörondan oluşan toplam yirmi model üzerinde yapılan testler sonucunda R2, RMSE, MSE ve MAE değerlerine göre en başarılı model 2 gizli katman ve 2 nörondan oluşan model olmuştur. En başarı model ile en başarısız model arasındaki farka baktığımızda %54 fark olduğu görülmektedir. Bu da model seçiminin ne kadar önemli olduğunun bir göstermektedir.

Kaynakça

Agatonovic-Kustrin, S., & Beresford, R. (2000). Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research. Journal of pharmaceutical and biomedical analysis, 22(5), 717-727.

Alami, A. (2016). Why do information technology projects fail. Procedia Computer Science, 100(2016), 62-71.

Attarzadeh, I., Mehranzadeh, A., & Barati, A. (2012). Proposing an enhanced artificial neural network prediction model to improve the accuracy in software effort estimation. Paper presented at the 2012 Fourth International Conference on Computational Intelligence, Communication Systems and Networks.

Azzeh, M., Nassif, A. B., & Banitaan, S. (2017). Comparative analysis of soft computing techniques for predicting software effort based use case points. IET Software, 12(1), 19-29.

Bailey, J. W., & Basili, V. R. (1981). A meta-model for software development resource expenditures. Paper presented at the ICSE.

Baskeles, B., Turhan, B., & Bener, A. (2007). Software effort estimation using machine learning methods. Paper presented at the 2007 22nd international symposium on computer and information sciences.

Borade, J. G., & Khalkar, V. R. (2013). Software project effort and cost estimation techniques. International Journal of Advanced Research in Computer Science and Software Engineering, 3(8).

Braga, P. L., Oliveira, A. L., Ribeiro, G. H., & Meira, S. R. (2007). Bagging predictors for estimation of software project effort. Paper presented at the 2007 International Joint Conference on Neural Networks.

Dan, Z. (2013). Improving the accuracy in software effort estimation: Using artificial neural network model based on particle swarm optimization. Paper presented at the Proceedings of 2013 IEEE International Conference on Service Operations and Logistics, and Informatics.

De Barcelos Tronto, I. F., da Silva, J. D. S., & Sant'Anna, N. (2007). Comparison of artificial neural network and regression models in software effort estimation. Paper presented at the 2007 International Joint Conference on Neural Networks.

Hecht-Nielsen, R. (1988). Neurocomputing: picking the human brain. IEEE spectrum, 25(3), 36-41.

Heemstra, F. J. (1992). Software cost estimation. Information and software technology, 34(10), 627-639.

Heiat, A. (2002). Comparison of artificial neural network and regression models for estimating software development effort. Information and software technology, 44(15), 911-922.

Jorgensen, M., & Shepperd, M. (2006). A systematic review of software development cost estimation studies. IEEE Transactions on software engineering, 33(1), 33-53.

Lippmann, R. (1987). An introduction to computing with neural nets. IEEE Assp magazine, 4(2), 4-22.

Mieritz, L. (2012). Survey shows why projects fail. Gartner report. Retrieved from Gartner Database.

Nassif, A. B., Capretz, L. F., & Ho, D. (2012). Estimating software effort using an ANN model based on use case points. Paper presented at the 2012 11th International Conference on Machine Learning and Applications.

Pospieszny, P., Czarnacka-Chrobot, B., & Kobylinski, A. (2018). An effective approach for software project effort and duration estimation with machine learning algorithms. Journal of Systems and Software, 137, 184-196.

Shan, Y., McKay, R. I., Lokan, C. J., & Essam, D. L. (2002). Software project effort estimation using genetic programming. Paper presented at the IEEE 2002 International Conference on Communications, Circuits and Systems and West Sino Expositions.

Spalek, S. (2005). Critical Success Factors in Project Management: To Fail Or Not to Fail, that is the Question!

Srinivasan, K., & Fisher, D. (1995). Machine learning approaches to estimating software development effort. IEEE Transactions on software engineering, 21(2), 126-137.

Şahinarslan, F. V. (2019). Makine Öğrenmesi Algoritmaları İle Nüfus Tahmini: Türkiye Örneği. Sosyal Bilimler Enstitüsü,

Tan, S. (2011). How to increase your IT project success rate. Gartner Research.

Viotti, P., Liuti, G., & Di Genova, P. (2002). Atmospheric urban pollution: applications of an artificial neural network (ANN) to the city of Perugia. Ecological Modelling, 148(1), 27-46.