Avrupa Bilim ve Teknoloji Dergisi
Sayı 23, S. 415-426, Nisan 2021

Araştırma Makalesi

www.ejosat.com ISSN:2148-2683

European Journal of Science and Technology
No. 23, pp. 415-426, April 2021

Research Article

Yazılım Projelerinin Maliyet Tahmini için WEKA'da Makine Öğrenmesi Algoritmalarının Karşılaştırmalı Analizi

Şükran Ebren Kara[1**,2], Rüya Şamlı[2]

[1*] Şırnak Üniversitesi, Cizre Meslek Yüksekokulu, Bilgisayar Teknolojileri Bölümü, Şırnak, Türkiye, (ORCID: 0000-0003-3071-6942), sebrenkara@sirnak.edu.tr
[2] İstanbul Üniversitesi-Cerrahpaşa, Bilgisayar Mühendisliği Bölümü İstanbul, Türkiye (ORCID: 0000-0002-8723-1228), ruyasamli@istanbul.edu.tr

(İlk Geliş Tarihi 9 Şubat 2021 ve Kabul Tarihi 4 Nisan 2021)

(**DOI:** 10.31590/ejosat.877296)

ATIF/REFERENCE: Ebren Kara, Ş. & Şamlı, R. (2021). Yazılım Projelerinin Maliyet Tahmini için WEKA'da Makine Öğrenmesi Algoritmalarının Karşılaştırmalı Analizi. Avrupa Bilim ve Teknoloji Dergisi, (23), 415-426.

Öz

Yazılım projelerinin en önemli sorunlarından biri, yazılım maliyet tahminidir. Yazılım maliyet tahmini, yazılımın gerçekleştirilmesi aşamasında gereken bütçenin tahmin edilmesidir. Proje yöneticisi, proje maliyetini doğru tahmin ederek projedeki belirsizlikleri azaltabilir. Aksi takdirde çok ciddi ekonomik sıkıntılar doğmaktadır. Büyük projelerin %60'ı öngörülen proje bütçelerini aşmıştır. Birtakım projelerin %15 maliyet aşımı nedeniyle hiçbir zaman tamamlanmadığı gözlemlenmiştir (Liu and Mintram, 2005). Yazılım projelerinin büyümesi ve karmaşık hale gelmesi sonucunda sürekli yeni maliyet hesaplama yöntemleri geliştirilmektedir. Bu çalışmada yazılım projelerinin maliyeti, makine öğrenmesi algoritmaları kullanılarak tahmin edilmeye çalışılmıştır. Proje maliyet tahmini, bilgi analizi için Waikato ortamında (Waikato Environment for Knowledge Analaysis - WEKA) bulunan 27 farklı makine öğrenmesi algoritmasında denenerek yapılmıştır. Algoritmalar 10 kat çapraz doğrulama tekniği ile PROMISE veri deposundan alınan COCOMO81, COCOMONASA ve COCOMONASA2 veri setlerine uygulanmış ve sonuçlar, performans ölçütü korelasyon katsayısı, hata oranları ortalama mutlak hata (mean absolute error - MAE), kök ortalama kare hata (root mean squared error - RMSE), bağıl mutlak hata (relative absolute error - RAE) ve kök bağıl kare hata (root relative squared error - RRSE) baz alınarak değerlendirilmiştir. Sonuçlar incelendiğinde bir algoritmanın her zaman en iyi sonucu üretmediği, farklı veri setleriyle farklı sonuçlar ürettiği bilgisine ulaşılmıştır. Bazı algoritmaların bazı veri setlerinde çok iyi çalışırken farklı parametrelerle ve farklı veri setlerinde kötü sonuçlar verebildiği gözlemlenmiştir. Bu çalışmayla yazılım maliyet tahmini için hangi algoritmaların kullanılabileceği, bu algoritmaların COCOMO veri setlerine uygulandığında tahmin sonuçlarının neler olabileceği ve en iyi çalışan algoritmaların hangileri olduğu bilgisine ulaşılmıştır.

Anahtar Kelimeler: Makine Öğrenmesi, COCOMO, Yazılım Maliyet Hesaplama, WEKA.

Comparative Analysis of Machine Learning Algorithms in WEKA for Cost Estimation of Software Projects

Abstract

One of the most important problems of software projects is software cost estimation. Software cost estimation is the estimation of the budget required during the actualization of the software. The project manager reduces the ambuguities in the project by estimating the project cost correctly. Otherwise, serious economic problems will arise. 60% of large projects exceeded the estimated project budgets. It has been observed that a number of projects were never completed due to a 15% cost overrun. As a result of the growth and complexity of software projects, new cost estimation methods are constantly being developed. In this study, the cost of software projects is tried to

[*] Sorumlu Yazar: sukranebren@hotmail.com

be estimated by using machine learning algorithms. Project cost estimation has been made by testing in 27 different machine learning algorithms in the waikato environment (WEKA) for information analysis. Algorithms were applied to COCOMO81, COCOMONASA and COCOMONASA2 datasets taken from PROMISE data store with 10-fold cross validation technique and results, performance criterion correlation coefficient, error rates mean absolute error (MAE), root mean square error (RMSE), relative absolute error (RAE) and root relative squared error (RRSE).When the results were examined, it was found that an algorithm does not always produce the best result, but produces different results with different data sets. It has been observed that some algorithms work very well in some data sets and may give poor results with different parameters and different data sets. Thanks to this study, the information about which algorithms can be used for software cost estimation, what the estimation results might be when these algorithms are applied to COCOMO data sets and which algorithms are the best working have been reached.

Keywords: Machine Learning, COCOMO, Software Cost Calculation, WEKA.

1. Giriş

Yazılım, kullanıcıların belirli gereksinimini karşılayacak, çoğu zaman arz talep ilişkisinden doğan, bilgisayar ve türevi teknolojik araçların istenilen işlemleri gerçekleştirebilmesi için geliştiriciler tarafından belli diller kullanılarak oluşturulan yapılardır ve bu yapıların en önemli noktalarından birisi içerdiği kod satırlarıdır. Bu kod satırları, bilgisayarlar ve türevi teknolojik araçlarla iletişimin kurulmasını sağlar. Yazılım, donanım – insan iletişimini sağladığı gibi donanım – donanım iletişimini de sağlar. Türk Dil Kurumuna göre yazılım: "Bir bilgisayarda donanıma hayat veren ve bilgi işlemde kullanılan programlar, yordamlar, programlama dilleri ve belgelemelerin tümüdür." (TDK, 2018).

Yazılım projelerinin en önemli sorunlarından biri, yazılım maliyet tahminidir. Yazılım maliyeti, yazılım geliştirme süreci için gereken kaynakların değeridir (Sommerville, 2000). Proje maliyet tahmini, proje etkinliklerinin tamamlanabilmesi için gereken parasal kaynaklara yaklaşım geliştirme sürecidir (ANSI/PMI, 2008). Adailer (2008) çalışmasında yazılım maliyet tahminini, bir yazılım sisteminin inşası için gereken çaba miktarının tahminleme süreci olarak tanımlamıştır. Bir yazılım projesinde maliyet tahminin yapılması, projenin teklif edilmesinde, kabul edilmesinde ve ilerlemesindeki birçok kararı etkilemektedir. Yazılım projelerinin maliyet tahmininin doğru gerçekleşmesi, ciddi bir yazılım maliyet hesaplama süreci gerektirmektedir. Bu süreç kapsamında önemli miktarda veri ile yapılan doğru analizler ve ölçümler, başarı oranını artırmaktadır.

Yazılım maliyetinin tahmin modelleri farklı şekillerde kategorize edilmiştir. Karataş (2011) çalışmasında algoritmik modeller ve algoritmik olmayan modeller olarak gruplandırırken, Gültekin (2019) çalışmasında model tabanlı, uzman tabanlı, öğrenme tabanlı, dinamik tabanlı, regresyon tabanlı ve karma tabanlı yöntemler olarak kategorize etmiştir.

Bu çalışmada yazılım maliyeti, Makine Öğrenmesi algoritmaları kullanılarak tahmin edilmeye çalışılmıştır. Yazılım maliyet tahmini için kullanılan COCOMO81, COCOMONASA ve COCOMONASA2 veri setleri PROMISE veri deposundan ücretsiz temin edilmiştir.

1.1. Makine Öğrenmesi

Makine öğrenmesi, bilgisayarların insanlar gibi öğrenmesini ve hareket etmesini algoritmalar ve veriler yardımıyla sağlayan yapay zekânın bir alt dalıdır (Kaluza, 2016). Amerikan bilgisayar bilimcisi Arthur Samuel tarafından 1959'da "*Makine Öğrenmesi*" terimi yaygınlaştırılmıştır. Makine Öğrenmesi, bilgisayarlara açıkça programlanmadan öğrenme olanağı sağlayan algoritmaların incelenmesi, tasarlanması ve geliştirilmesi ile ilgilidir (Wikipedia, 2020; Prowmes, 2019). Makine Öğrenmesinin temelinde makineleri kendi kendine karar verebilir hale getirme fikri yatmaktadır. "Bilgi kuvvettir" prensibine dayanarak, bir sistem ne kadar çok bilgiye sahipse, o kadar çok şey öğrenebilir ve ne kadar çok şey öğrenirse o kadar çok doğru kararlar verebilir. Başka bir tanıma göre makine öğrenmesi; mantıksal işlemler gerçekleştirebilen makinelerin, gözlem ve ölçüm yöntemleriyle elde edilen verileri tecrübe olarak kabul etmesi ve bu tecrübelerden matematiksel algoritmalar aracılığıyla anlamlı ilişkiler üretmesi sürecidir (Torkul vd., 2017).

Makine öğrenmesinde üç çeşit öğrenme yöntemi vardır: denetimli öğrenme, denetimsiz öğrenme ve yarı denetimli öğrenme.

Denetimli öğrenmede; sisteme öğrenmesi için giriş değerleri ile birlikte çıkış değerleri de verilir. Giriş verilerinin değerleri işlenerek çıkış değerleri tahmin edilmeye veya öğrenilmeye çalışılır. Bu süreçte öncelikle sonuçları bilinen veriler üzerinde bir sınıflama yapılır ve sonuçları bilinmeyen veri kümesi için tahmin edilmeye çalışılır (Aydın ve Özkul, 2015). Denetimli öğrenme, ses tanıma, e-posta spam filtreleme, fotoğraflarda yüz tanıma ve kredi kartı sahtekârlıklarını tespit etme gibi işlemlerin arkasındaki ana kavramdır (Kaluza, 2016). Regresyon, karar ağaçları, sınıflandırma denetimli öğrenme algoritmalarıdır.

Denetimsiz öğrenmede; sisteme dışarıdan bir müdahale söz konusu değildir. Algoritmanın kendi kendine keşifler yapması, gözükmeyen örüntüleri keşfetmesi beklenir. Girdi verileri sisteme (öğrenme algoritmasına) verilir ancak herhangi bir işaretleme yapılmaz yani talimat verecek bir insan operatörü yoktur. Öğrenme algoritması verileri tanımlamak için inceler keşifler yapar, veriler arasındaki ilişki ağını ortaya koymaya çalışır. (Aydın ve Özkul, 2015; Alpaydın, 2011). Sistem daha fazla veriyi değerlendirdikçe, bu verilerle ilgili karar verebilme becerisi giderek artmakta ve daha fazla doğruya yaklaşmaktadır. Kümeleme, boyut azaltma bazı denetimsiz öğrenme algoritmalarıdır.

Elde az sayıda işaretlenmiş veri olmasına karşın çok daha fazla sayıda işaretlenmemiş veri varsa denetimli öğrenme de denetimsiz öğrenme de yetersiz kalabilir. Bu durumda yarı denetimli öğrenme yöntemi daha kullanışlıdır. Yarı denetimli öğrenme, az sayıdaki işaretlenmiş veriden çok sayıdaki işaretlenmemiş veriyi tahmin etmek ve sınıflandırmaktır. Yarı denetimli öğrenme sisteme denemeyi ve hatayı öğretir. Geçmiş deneyimlerden öğrenir ve mümkün olan en iyi sonucu elde etmek için duruma cevap olarak yaklaşımını adapte etmeye çalışır (Kızılkaya ve Oğuzlar, 2018).

Makine öğrenmesinin iş akışı 5 adımda toparlanabilir.

1. Veri ve problem tanımı: Bu aşamada çözmeye çalışılan problemin tanımı, önemi, bu problemin çözülmesi için kullanılacak verilerin tanımı gibi bilgiler verilmelidir.

2. Veri toplama: Bu aşamada ele alınan problemin çözümü için hangi tür verilerin elde edilmesi gerektiği belirlenir ve bu veriler toplanmaya çalışılır.
3. Veri ön işleme: Problem çözümünde kullanılacak olan veriler temiz olmalıdır. Kayıp veriler varsa belli algoritmalar kullanılarak onların yeri doldurulmalı, gürültülü veriler denetlenmeli, aykırı değerler kaldırılmalıdır. Daha sonra normalizasyon, değer depolarına uyum ve boyut sayısını azaltmak işlemleri gerçekleştirilmelidir.
4. Veri analizi ve denetimli öğrenme veya denetimsiz öğrenme ile modelleme yapma: Bu aşamada problemin çözümü için, bir model belirlenir. Bu model ile denetimli öğrenme yöntemi mi yoksa denetimsiz öğrenme yöntemi mi gerekli belirlenmelidir. Makine öğrenmesi algoritmalarının geniş bir yelpazesi vardır. Rastgele orman, karar ağaçları, destek vektör makineleri, lojistik regresyon, K-en yakın komşuluk (K-Nearest Neighbor-KNN) bu algoritmalardan sadece birkaç tanesidir.
5. Değerlendirme: Son aşama model değerlendirmedir. Bu adımın amacı modeli doğru değerlendirmek ve yeni veriler üzerinde doğru çalışacağından emin olmaktır. Değerlendirme yöntemleri arasında, çapraz doğrulama, eğitim ve test için ayrılmış veri seti, tek çıkışlı çapraz doğrulama gibi yöntemler mevcuttur (Kaluza, 2016).

Yazılım geliştirme sürecinde, maliyet, süre ve çaba gibi proje için temel kavramların yanlış hesaplanması nedeniyle çöken birçok proje mevcuttur (Karataş, 2011). Özellikle yazılım projelerinde, ölçülebilir hedefler konulmadığı için başarı sağlanamamaktadır (Ayyıldız, 2007).

Yazılım projelerinin maliyet tahmini konusunda literatürde oldukça geniş çaplı araştırmalar yürütülmektedir. Sezer (2008) çalışmasında yazılım projelerinin maliyetini tahmin etmek amacıyla YSA (Yapay Sinir Ağı) uygulaması gerçekleştirmiştir. YSA olarak çok katmanlı ileri beslemeli yapay sinir ağını, eğitim algoritması olarak delta algoritmasını seçmiştir. Eğitimi, örnek veri seti kullanarak tamamlandıktan sonra test verilerini ağa sunup hedef çıktıyı elde etmiştir. YSA uygulamasından elde ettiği verileri COCOMO 2000 verileri ile karşılaştırmıştır.

Kulter vd. (2009) yazılım maliyet tahmini için yeni bir makine öğrenme modeli önermişlerdir. Çalışmada yazılım maliyet tahmini için ilişkisel bellek ile sinir ağlarını birleştirmişlerdir. Tek bir, çok katmanlı algılayıcı (multilayer perceptrons - MLP) yerine, birden fazla MLP'yi birleştirerek kullanmışlardır. Modele tahminde bulunması için yeni bir proje verildiği zaman, her bir MLP kendi sonucunu temin etmiştir. Bu modelde birleştirme yaklaşımının kullanılmasına ek olarak, ilişkisel bellek kullanılmıştır. Bu yeni modele ENNA (İlişkilendirilebilir Hafızalı Sinir Ağları Topluluğu) denmiştir.

Kumari ve Pushkar (2013) algoritmik ve algoritmik olmayan mevcut yazılım maliyet tahmin modelleri ve teknikleri üzerinde ayrıntılı bir çalışma yapmıştır. Bu modellerin avantajlarını ve dezavantajlarını çalışmalarında sunmuştur. Çalışmada her tahmin modelinin belirli proje türüne özgü olduğu herhangi bir tahminleme modelinin diğerlerinden daha üstün olduğunu söylemenin çok zor olduğunu ve her yöntemin kendi proje türüne göre en iyisi olduğunu belirtmişlerdir.

Başka bir çalışmada (Niranjan vd., 2017) saldırı tespit sisteminin herhangi bir saldırı faaliyetini en yüksek doğrulukla sınıflandırabilmesi için makine öğrenmesi algoritmalarından Bagging, J48 ve Random Committee algoritmaları kullanılmıştır. Bagging ve Random Committee algoritmaları için Random Tree temel sınıflandırıcı olarak seçilmiştir. Çalışmada iki veri setine Bagging, J48 ve Random Committee algoritmaları çapraz doğrulama uygulayarak True Pozitif, False Pozitif ve kesinlik değerleri sonuçları tablo şeklinde verilmiştir.

Gültekin (2019) çalışmasında yazılım maliyetinin tahmini için üç model geliştirmiştir. Bu modellerden ilki, regresyon tabanlı bir modeldir. Bu modelde, COCOMO81, COCOMONASA ve COCOMONASA2 veri setlerine ağırlıklandırılmış lineer regresyon, logaritmik regresyon, üstel regresyon, polinomsal regresyon ve çoklu polinomsal regresyon yöntemlerini uygulanmıştır. Bu modellerden ikincisi, YSA tabanlı bir modeldir. Sunulan üçüncü model ise regresyon tabanlı makine öğrenme algoritmalarıyla maliyet tahmini yapan bir modeldir. Bu modelde Destek Karar Regresyon (Suppport Vector Regression - SVR), Gradyan Artırma (Gradient Boosted Machine - GBM), Rastgele Orman (Random Forest) ve MLP gibi regresyon tabanlı algoritmalar kullanılmıştır.

Marapelli (2019) çalışmasında makine öğrenmesi yöntemiyle COCOMO81, COCOMONASA ve COCOMONASA2 veri setlerini kullanarak lineer regresyon ve K-en yakın komşu algoritmalarıyla yazılım maliyet tahminini gerçekleştirmiştir. Yazar çalışmasında, korelasyon katsayısı, MSE, MMRE performans ölçütlerini kullanarak algoritmalardan elde ettiği tahmin sonuçlarını karşılaştırmıştır.

Guven Aydın ve Samli (2020) yaptıkları çalışmalarında, PROMISE veri deposundan temin ettikleri JM1, KC1, CM1, PC1 veri setlerini incelemişlerdir. Bu veri setlerine WEKA aracı ile 40'tan fazla veri madenciliği algoritması kullanarak yazılım hatası tahmin ölçütlerini karşılaştırmışlardır. Algoritmaların doğruluk oranları incelendiğinde Bagging algoritmasının en iyi doğruluk oranına sahip olduğunu gözlemlemişlerdir.

2. Materyal ve Metot

2.1. Veri Seti

Bu çalışmada, PROMISE yazılım mühendisliği veri deposundan ücretsiz ve açık olarak indirilen COCOMO81, COCOMONASA ve COCOMONASA2 veri setleri kullanılmıştır. Bu veri setlerinde yazılım projelerinin maliyet tahmininde kullanılabilecek bağımlı ve bağımsız öz nitelikler mevcuttur. Bağımlı öz nitelik gerçek maliyet; act_effort ve bağımsız öz nitelikler; rely, data, cplx, time, stor, virt, turn, acap, aexp, pcap, vexp, lexp, modp, tool, sced, loc maliyet ile ilgili olanlardır. Veri setinin bağımsız öz nitelikleri, bağımlı öz niteliğin değerine karar vermektedir. COCOMO81 veri setinde 63, COCOMONASA veri setinde 60 ve COCOMONASA2 veri setinde 93 tane yazılım projesine ait maliyet sonuçları mevcuttur. COCOMO81 ve COCOMONASA veri setlerinde biri maliyet olmak üzere 17 tane öz nitelik bulunmaktadır. COCOMONASA2 veri setinde bu 17 öz niteliğin dışında maliyete etkisi olmayan 7 tane öz nitelik (recordnumber, projectname, cat2, forg, center, year ve mode) mevcuttur. Bu öznitelikler çalışmaya dahil edilmemiştir. COCOMO maliyet faktörleri Tablo 1'de, COCOMO maliyet faktörlerinin standart sayısal değerleri Tablo 2'de verilmiştir.

Aşağıda verilen Tablo 1'de veri setinde bulunan 15 tane öz nitelik incelenmiştir. Veri setinde bunlarında dışında kod satır sayısı (loc) ve maliyet (act_effort) öz nitelikleri de mevcuttur.

Tablo 2'de veri setinde bulunan öz niteliklerin alabildiği en büyük ve en küçük değer aralıkları incelenmiştir. COCOMONASA2 veri setinde effor tahmini için kullanılmayan

7 tane öznitelik veri setine dahil edilmemiştir. Veri setleri, eğitim seti ve test seti olarak k-kat çapraz doğrulama tekniği kullanılarak rastgele bölünmüştür.

Tablo 1. COCOMO maliyet faktörleri

Ürün Özellikleri	rely	Required software reliability	Gerekli yazılım güvenliği
	data	Database size	Veritabanı büyüklüğü
	cplx	Software product complexity	Ürün karmaşıklığı
Donanım Özellikleri	time	Execution time constraint	Çalışma süresi kısıtı
	stor	Main storage constraint	Temel depolama kısıtı
	virt	Virtual machine volatility	Sanal makine geçiciliği
	turn	Computer turn around time	Bilgisayar yanıt süresi
Personel Özellikleri	acap	Analist capability	Çalışan analistin kapasitesi
	aexp	Application experience	Proje takımının uygulama tecrübesi
	pcap	Programmer capability	Programcı kapasitesi
	wexp	Virtual machine experience	Takımın Sanal makine tecrübesi
	lexp	Language experience	Takımın programlama dili tecrübesi
Proje Özellikleri	modp	Use of modern programming practices	Modern programlama uygulamaları
	tools	Use of software tools	Kullanılan yazılım araçları
	sced	Development schedule constraint	iş takvimi kısıtı

Tablo 2. COCOMO maliyet faktörlerinin standart sayısal değerleri

Öz nitelik	**çok düşük**	**düşük**	**normal**	**yüksek**	**çok yüksek**	**ekstra yüksek**	**verimlilik aralığı**
acap	1,46	1,19	1,00	0,86	0,71		2,06
pcap	1,42	1,17	1,00	0,86	0,70		1,67
aexp	1,29	1,13	1,00	0,91	0,82		1,57
modp	1,24	1,10	1,00	0,91	0,82		1,34
tool	1,24	1,10	1,00	0,91	0,83		1,49
vexp	1,21	1,10	1,00	0,90			1,34
lexp	1,14	1,07	1,00	0,95			1,20
sced	1,23	1,08	1,00	1,04	1,10		e
stor			1,00	1,06	1,21	1,56	-1,21
data		0,94	1,00	1,08	1,16		-1,23
time			1,00	1,11	1,30	1,66	-1,30
turn		0,87	1,00	1,07	1,15		-1,32
virt		0,87	1,00	1,15	1,30		-1,49
cplx	0,70	0,85	1,00	1,15	1,30	1,65	-1,86
rely	0,75	0,88	1,00	1,15	1,40		-1,87

2.2. Model Oluşturma

WEKA aracında verileri sınıflandırmak, bölütlemek, tahmin etmek, veriler arasında ilişki kurmak için çok fazla makine öğrenmesi algoritması mevcuttur.

Bu çalışmada PROMISE veri deposundan alınan COCOMO81, COCOONASA ve COCOMONASA2 veri setleriyle WEKA aracında bulunan ilgili algoritmalar kullanılarak yazılım proje maliyet tahmini gerçekleştirilmiştir.

Weka aracında Meta başlığı altında bulunan algoritmalar ile Lazy grubunda bulunan LWL ve Rules grubunda bulunan Input Mapped Classifier algoritmaları, kendi parametrelerine ek olarak bir temel sınıflandırıcı ve onun parametrelerini alan algoritmalardır. Bu yüzden en iyi performansı almak için properties penceresinden sınıflandırıcı parametreleri değiştirilmiş, doğru bir karşılaştırma olması için hepsine aynı sınıflandırma algoritması seçilmiştir. Ayrıca parametre olarak bütün sınıflandırıcılar denenmiş en iyi sonucu veren parametre sınıflandırıcılar ayrı bir tabloda belirtilmiştir.

2.3. Modellerin Performans Değerlendirmesi

2.3.1. Korelasyon Katsayısı (Correlation Coefficient)

Korelasyon katsayısı, bağımlı değişken ile bağımsız değişkenler arasındaki istatistiksel ilişkinin yönünün ve gücünün ölçüdür. Farklı durumlar için farklı korelasyon katsayıları geliştirilmiştir. Korelasyon katsayısı -1 ile 1 arasında bir değer olabilmektedir. Korelasyon katsayısının -1 olması iki değişken arasında ters ilişki olduğunu, korelasyon katsayısının 0 olması, iki değişken arasında hiç bir ilişkinin olmadığını, korelasyon

katsayısının 1 olması ise iki değişken arasında tam ilişki olduğunu göstermektedir (Wikipedia, 2020; Marapelli, 2019).

2.3.2. Ortalama Mutlak Hata (Mean Absolute Error - MAE)

Ortalama mutlak hata, tahmin edilen değerlerin gerçek değerlerden ne kadar uzakta olduğunu bulmaya çalışmaktadır. Formülü Denk. (1)'de verilmiştir.

$$MAE = \frac{1}{\mathrm{n}} \sum_{i=1}^{n} (P_i - A_i) \quad (1)$$

Burada P_i = tahmini değer, A_i = gerçek değer, n = örnek sayısı

2.3.3.Kök Ortalama Kare Hata (Root Mean Squared Error - RMSE)

Kök ortalama kare hatası, tahmin edilen değerler ile gerçek değerler arasındaki farklılıkların örnek standart sapmasını verir. Formülü Denk. (2)'de verilmiştir.

$$RMSE = \sqrt{\frac{1}{\mathrm{n}} \sum_{i=1}^{n} (P_i - A_i)^2} \quad (2)$$

Burada P_i = tahmini değer, A_i= gerçek değer, n = örnek sayısı

RMSE büyük hatalara büyük önem verir çünkü hataların ortalamaları alınmadan önce kareleri alınmaktadır. Bu nedenle RMSE daha çok büyük hataların istenmediği durumlarda kullanılır.

2.3.4. Bağıl Mutlak Hata (Relative Absolute Error - RAE)

Bağıl mutlak hata, tahmin edilen değerler ile gerçek değerler arasındaki farkın bir toplamını vererek bunu gerçek değer ile gerçek değerin ortalaması arasındaki farkın toplamına böler. Formülü Denk. (3)'de verilmiştir.

$$RAE = \frac{\sum_{i=1}^{n} (\mathrm{P}_i - \mathrm{A}_i)}{\sum_{i=1}^{n} (\mathrm{A}_i - \mathrm{A}_m)} \quad (3)$$

Burada P_i = tahmini değer, A_i = gerçek değer, A_m= gerçek değerlerin toplamı, n = örnek sayısı

2.3.5. Kök Bağıl Kare Hata (Root Relative Squared Error - RRSE)

Ayrı ayrı j veri kümesinin kök bağıl kare hatası aşağıdaki gibi tanımlanmıştır (Prabhakar ve Drutta, 2013) Formülü Denk. (4)'te verilmiştir.

$$RRSE = \sqrt{\frac{\sum_{i=1}^{n} (P_{ij} - \mathrm{A}_i)^2}{\sum_{i=1}^{n} (\mathrm{A}_i - \mathrm{A}_m)^2}} \quad (4)$$

Burada P_{ij} = i'deki veri noktası için ayrı ayrı j veri kümesinin tahmini değeri, A_i = gerçek değer, A_m= gerçek değerlerin toplamı, n = örnek sayısı

2.4. Metotlar

Bu bölümde, makale içerisinde kullanılan algoritmaların ve kullanılan yöntemlerin açıklamaları sunulmuştur. Algoritmalar literatürde genel olarak orijinal isimleriyle kullanıldığı için bu makalede de bu şekilde kullanılmış ancak yanı sıra mümkün olduğunca Türkçeleştirilmiş isimleri de verilmiştir (Güven Aydın, 2021).

Gaussian Processes (Gauss Yöntemi): Bir sınıflandırma Makine Öğrenmesi algoritmasıdır. Sınıflandırma tahmini modelleme için, Gauss süreçleri kullanılabilir. Gauss Süreç modelleri çok değişkenli dağılımlı rastgele değişkenlerin sınırlı bir koleksiyonuna sahip parametrik olmayan çekirdek tabanlı olasılık modelleridir. Her doğrusal birleşim eşit dağılmıştır (Yergök ve Acı, 2019).

Linear Regression (Doğrusal Regresyon): Regresyon analizi, iki ya da daha çok değişken arasındaki sayısal ilişkiyi incelemek amacıyla kullanılan bir analiz yöntemidir. Regresyon analizinde, değişkenler arasındaki ilişkiyi fonksiyonel olarak açıklamak ve bu ilişkiyi bir modelle tanımlayabilmek amaçlanmaktadır. Regresyon analizinde bir tane bağımlı ve bir tane bağımsız iki değişken varsa buna Basit Doğrusal Regresyon, bir tane bağımlı birden fazla bağımsız değişken varsa buna Çoklu Doğrusal Regresyon denir. Çok Değişkenli Regresyon analizi, Çoklu Doğrusal Regresyon analizinin genelleştirilmiş halidir. Çok Değişkenli Regresyon analizinde birden fazla bağımlı değişken vardır.

Multilayer Perceptron (Çok Katmanlı Algılayıcı): Örnekleri sınıflandırmak ve çok katmanlı bir algılayıcı öğrenmek için geri yayılımı kullanan bir sınıflandırıcıdır.
Ağ elle oluşturulabilir veya basit bir buluşsal yöntem kullanılarak kurulabilir. Ağ parametreleri eğitim süresi boyunca izlenebilir ve değiştirilebilir. Bu ağdaki düğümlerin tümü sigmoiddir. (sınıfın sayısal olduğu durumlar hariç, bu durumda çıkış düğümleri eşiksiz doğrusal birimler haline gelir.)

SMOreg: SMOreg algoritması temel olarak destek vektör makineleri olarak bilinen yöntemleri kullanır. DVM'ler 1960'lı yılların sonunda Vladimir Vapnik ve Alexey Chervonenkis tarafından geliştirilmiş temel olarak istatiksel öğrenme teorisine dayanan bir makine öğrenmesi yöntemidir.

IBK (K-nearest neighbor – K-en yakın komşuluk): K-en yakın komşular sınıflandırıcısıdır. Çapraz doğrulamaya dayalı olarak uygun K değerini seçebilir. Mesafe ağırlıklandırması da yapabilir.

Kstar (Kyıldız – K*): K*, örnek tabanlı bir sınıflandırıcıdır, yani bir test örneğinin sınıfı, bazı benzerlik işlevleriyle belirlendiği üzere, ona benzer eğitim örneklerinin sınıfına dayanır. Entropi tabanlı bir mesafe işlevi kullanması açısından diğer örnek tabanlı öğrenicilerden farklıdır.

LWL (Locally Weighted Learning - Yerel Ağırlıklı Öğrenme) Yerel ağırlıklı bir öğrenme yöntemidir. Örnek ağırlıkları atamak için örnek tabanlı bir algoritma kullanır ve bunlar daha sonra belirli bir WeightedInstancesHandler sınıfı tarafından kullanılır. Sınıflandırma (örn. Naive Bayes kullanarak) veya regresyon (örn. Doğrusal regresyon kullanarak) yapabilir.

Additive Regression (Toplamsal Regresyon): Genellikle doğrusal olmayan gerçek yaşam etkilerinde Doğrusal Regresyon başarısız sonuçlar üretebilmektedir. Daha esnek istatistiksel modellerden olan additive regresyon, doğrusal olmayan regresyon etkilerini tanımlamak için kullanılabilmektedir.

Attribute Selected Classifier (Öznitelik Seçici Sınıflandırıcı): Eğitim ve test verilerinin boyutu, bir sınıflandırıcıya aktarılmadan önce nitelik seçimi ile azaltılır.

Bagging (Torbalama): Varyansı azaltmak için bir torbalama sınıflandırıcısıdır.

CVParameter Selection (Cross Validation Parameter Selection - Cross Validation Parameter Selection): Herhangi bir sınıflandırıcı için çapraz doğrulama ile parametre seçimi gerçekleştirme sınıfıdır.

Multi Schema (Çoklu Şema): Eğitim verileri veya eğitim verilerindeki performans üzerinde çapraz doğrulama yöntemi kullanarak birkaç sınıflandırıcı arasından bir sınıflandırıcı seçmek için kullanılan bir sınıflandırıcıdır. Performans, yüzde doğru (sınıflandırma) veya ortalama kare hataya (regresyon) göre ölçülür.

Random Comittee (Rastgele Komite): Randomize edilebilir temel sınıflandırıcılar topluluğu oluşturmak için bir sınıflandırıcıdır. Her temel sınıflandırıcı, farklı bir rasgele sayı çekirdeği kullanılarak oluşturulur (ancak aynı verilere dayalı olarak). Nihaî tahmin, bireysel temel sınıflandırıcılar tarafından üretilen tahminlerin düz bir ortalamasıdır.

Randomizable Fitered Classifer (Randomize Edilebilir Filtreli Sınıflandırıcı): FilteredClassifier'ın, modeli randomize edilebilir bir filtreyle, daha spesifik olarak, RandomProjection ve temel sınıflandırıcı olarak IBk ile başlatan basit bir varyantıdır. Bunun dışında ve iki temel şemadan en az birinin Randomizable arayüzü uyguladığını kontrol ederek, Randomizable'ı da uygulayan FilteredClassifier ile tam olarak aynı işlevselliği uygular.

Random SubSpace (Rastgele Alt Boşluk): Bu yöntem, eğitim verilerinde en yüksek doğruluğu koruyan ve karmaşıklık arttıkça genelleme doğruluğunu iyileştiren karar ağacı tabanlı bir sınıflandırıcıdır. Sınıflandırıcı, özellik vektörünün bileşenlerinin alt kümelerinin sözde rastgele seçilmesiyle sistematik olarak oluşturulmuş birden çok ağaçtan, yani rastgele seçilen alt uzaylarda oluşturulan ağaçlardan oluşur.

Weighted Instances Handler Wrapper (Ağırlıklı Örnek İşleyici Sarmalayıcı): Ağırlıklı örnek desteğini etkinleştirmek için herhangi bir sınıflandırıcı etrafında genel bir sarmalayıcıdır. Temel sınıflandırıcı WeightedInstancesHandler arabirimini uygulamıyorsa ve 1.0'dan farklı örnek ağırlıkları varsa, ağırlıklarla yeniden örneklemeyi kullanır. Varsayılan olarak, örnek ağırlıklarını işleyebiliyorsa eğitim verileri temel sınıflandırıcıya aktarılır. Bununla birlikte, yeniden örnekleme kullanımını ağırlıklarla da zorlamak mümkündür.

Input Maped Classifier (Giriş Eşlemeli Sınıflandırıcı): Bir sınıflandırıcının birlikte oluşturulduğu eğitim verileri ile gelen test örneklerinin yapısı arasında bir eşleme oluşturarak uyumsuz eğitim ve test verilerini ele alan sarmalayıcı sınıflandırıcısıdır. Gelen örneklerde bulunmayan model öznitelikleri eksik değerleri alır, dolayısıyla sınıflandırıcının daha önce görmediği gelen nominal öznitelik değerleri de alır. Yeni bir sınıflandırıcı eğitilebilir veya bir dosyadan mevcut bir sınıflandırıcı yüklenebilir.

Decision Table (Karar Ağacı): Basit bir karar tablosu ile çoğunluk sınıflandırıcısı oluşturmak için kullanılan bir sınıflandırıcıdır.

M5 Rules (M5 Kuralları): M5 Rules algoritması regresyon problemlerinde karar listeleri oluşturmak için böl ve yönet tekniğini kullanan bir algoritmadır. M5 Rules algoritması bir model ağacı oluşturmak için böl ve yönet tekniğini kullanır, en iyi yapraktan bir kural oluşturur ve ardından oluşturulan kurula göre veri kümesinde kalan diğer örnekler üzerinde çalışır.

ZeroR: ZeroR sınıflandırıcı oluşturma ve kullanma sınıfıdır. Ortalamayı (sayısal bir sınıf için) veya modu (nominal bir sınıf için) tahmin eder.

Desicion Stump (Karar Kütüğü): Bir karar güdüsü oluşturma ve kullanma sınıfıdır. Genellikle bir yükseltme algoritması ile birlikte kullanılır. Regresyon (ortalama kare hatasına göre) veya sınıflandırma (entropiye dayalı) yapar. Eksik, ayrı bir değer olarak kabul edilir.

M5P: M5P deneysel verilerden bir regresyon ağaç modeli oluşturmak için, M5 algoritmasının yeniden yapılandırılmasıdır. Bir M5P modelinde, ağaç her dalda, veri kümesinin yaprağa ulaşan kısmının sınıf değerlerini tahmin eden doğrusal bir regresyon modelini depolar. Veri kümesi, verilerin belirli özelliklerine göre farklı bölümlere ayrılır. Standart sapma genellikle, her düğümde veri kümesini bölmek için hangi özniteliğin en iyi olduğunu belirleyen bir ölçüt olarak kullanılır. Seçilecek öznitelik, hatayı azaltmak için maksimum beklentiye sahip olandır.

Random Forest (Rastgele Orman): Rastgele ağaçlardan oluşan bir orman inşa etmek için kullanılan bir sınıflandırıcıdır.

Random Tree (Rastgele Ağaç): Her düğümde *K* rastgele seçilen öznitelikleri dikkate alan bir ağaç oluşturmak için kullanılan bir sınıflandırıcıdır. Budama yapmaz. Ayrıca, bir uzatma kümesine (geri uyum) dayalı olarak sınıf olasılıklarının (veya regresyon durumunda hedef ortalamanın) tahminine izin verme seçeneğine de sahiptir.

REP Tree (Rep Ağacı): Hızlı karar ağacı öğrenicisidir. Bilgi kazanımı / varyansını kullanarak bir karar / gerileme ağacı oluşturur ve bunu daha az hata ile budama (arka plan ile) kullanarak kurar. Sayısal özellikler için değerleri yalnızca bir kez sıralar. Eksik değerler, karşılık gelen örnekleri parçalara bölerek ele alınır.

3. Araştırma Sonuçları ve Tartışma

Bu çalışmada yazılım maliyetinin tahmini için, COCOMO81, COCOMONASA ve COCOMONASA2 veri setleri kullanılmıştır. Deneylerde veri madenciliği aracı olan WEKA programı kullanılmıştır. Veri setleri 10 – kat çapraz doğrulama tekniği ile eğitim ve test verilerine rastgele ayrılmıştır. Oluşturulan Modellerin performans ölçümleri, korelasyon katsayısı, MAE, RMSE, RAE ve RRSE baz alınarak değerlendirilmiştir.

Doğru bir analiz karşılaştırması olabilmesi için bu algoritmaların hepsine Random Forest algoritması temel sınıflandırıcı olarak seçilip testler yapılmıştır. Sonuçlar Tablo 4, Tablo 5 ve Tablo 6'da gösterilmiştir.

Bazı algoritmalar (meta başlığı altındaki bütün algoritmalar, Lazy grubundan LWL, Rules grubundan Input Mapped) kendi parametrelerine ek olarak bir temel sınıflandırıcı ve onun parametrelerini alan meta yöntemlerdir. Bu algoritmalar için bütün olasılıklar incelenmiştir. Olabilecek en iyi performans ölçümü Tablo 6, Tablo 7 ve Tablo 8'de gösterilmiştir.

Tablo 3. COCOMO81 veri setine uygulanan algoritmaların performans ölçümleri

COCOMO81 Veri Seti					
ALGORİTMALAR	**Ölçütler**				
FONKSİYONLAR	**Korelasyon katsayısı**	**MAE**	**RMSE**	**RAE(%)**	**RRSE(%)**
Gaussian Processes	0,5401	790,6207	1529,4219	87,1334	83,3127
Linear Regression	0,6102	874,477	1480,8087	96,3751	80,6645
Multilayer Perceptron	0,6739	662,3573	1651,8813	72,9976	89,9834
Simple Linear Regression	0,5803	610,8756	1556,9319	67,3239	84,8112
SMOreg	0,6598	481,4058	1414,1265	53,0552	77,0321
LAZY					
IBK (K-nearest neighbor)	0,6391	597,2745	1495,836	65,8249	81,4831
KStar	0,5621	527,3596	1707,526	58,1197	93,0146
LWL	0,7852	513,1837	1320,6153	56,5574	71,9383
META					
Additive Regression	0,8095	471,6203	1169,6529	51,9767	63,7149
Attribute Selected Classifier	0,7766	480,6095	1266,5644	52,9674	68,994
Bagging	0,6842	615,7212	1427,0346	67,8579	77,7353
CVParameter Selection	0,7624	547,2516	1288,8028	60,312	70,2054
Multi Schema	0,759	527,6654	1317,3837	58,1534	71,7622
Random Comittee	0,7722	529,8123	1303,0491	58,39	70,9814
Randomizable Fitered Classifer	0,541	570,3308	1525,4542	62,8555	83,0965
Random SubSpace	0,6095	649,4035	1470,7424	71,57	80,1162
Regresiyon By Discretization	0,7482	555,2121	1348,4807	61,1893	73,4562
Weighted Instances Handler Wrapper	0,7624	547,2516	1288,8028	60,312	70,2054
MISC					
Input Maped Classifier	0,7624	547,2516	1288,8028	60,312	70,2054
RULES					
Decision Table	0,3947	616,2634	1785,8066	67,9177	97,2788
M5 Rules	0,7657	603,709	1289,9993	66,5341	70,2705
ZeroR	-0,3757	907,3682	1835,7614	100	100
TREE					
Desicion Stump	0,4596	717,5814	1673,7058	79,0838	91,1723
M5P	0,6843	517,3589	1334,687	57,0175	72,7048
Random Forest	0,7624	547,2516	1288,8028	60,312	70,2054
Random Tree	0,37	688,7117	1840,2042	75,9021	100,242
REP Tree	0,0902	787,8688	1904,8964	86,8301	103,766

Tablo 3'te, COCOMO81 veri setine uygulanan Makine Öğrenmesi algoritmalarının performans değerlendirme sonuçları belirtilmiştir. COCOMO81 veri setinde en iyi tahmini, Meta grubundan 0,8095 korelasyon katsayısı ve %51,9767 RAE hata payı ile Additive Regression algoritması gerçekleştirmiştir. Makine Öğrenmesi algoritmalarından ZeroR algoritması WEKA aracı kullanılarak COCOMO81 veri setine maliyet tahmini için uygulandığında en kötü tahmin performasını göstermiştir. Tree grubundan REP Tree sınıflandırma algoritması ZeroR algoritmasından sonra 0,0902 korelasyon katsayısı ve %86,8301 RAE hata payı ile en kötü performansı sergilemiştir.

Tablo 4. COCOMONASA veri setine uygulanan algoritmaların performans ölçümleri

COCOMONASA Veri Seti					
ALGORİTMALAR	**Ölçütler**				
FONKSİYONLAR	**Korelasyon katsayısı**	**MAE**	**RMSE**	**RAE(%)**	**RRSE(%)**
Gaussian Processes	0,6387	269,4976	513,356	62,5047	77,0828
Linear Regression	0,7994	247,0464	431,768	57,2976	64,832
Multilayer Perceptron	0,8931	179,4526	310,3657	41,6205	46,6029
SMOreg	0,719	248,4012	462,9543	57,6118	69,5148
LAZY					
IBK (K-nearest neighbor)	0,5768	295,4267	590,2186	68,5184	88,6241
KStar	0,6772	220,4516	501,335	51,1294	75,2778
LWL	0,7779	210,3535	420,0186	48,7874	63,0678
META					
Additive Regression	0,8317	200,551	367,676	46,5139	55,2083
Attribute Selected Classifier	0,8251	202,3599	392,3317	46,9334	58,9105
Bagging	0,7871	222,7612	425,0947	51,6651	63,83
CVParameter Selection	0,8196	211,6876	403,4439	49,0968	60,579
Multi Schema	0,7818	217,0315	416,7733	50,3362	62,5805
Random Comittee	0,7813	217,802	422,6492	50,5149	63,4628
Randomizable Fitered Classifer	0,8825	148,6937	313,6628	34,4866	47,0979
Random SubSpace	0,6964	255,7131	474,6142	59,3076	71,2655
Regresiyon By Discretization	0,704	251,3443	470,4945	58,2944	70,647
Weighted Instances Handler Wrapper	0,8196	211,6876	403,4439	49,0968	60,579
MISC					
Input Maped Classifier	0,8196	211,6876	403,4439	49,0968	60,579
RULES					
Decision Table	0,4577	261,1296	609,2296	60,5639	91,4787
M5 Rules	0,9152	157,1147	263,9787	36,4397	39,6376
ZeroR	-0,4382	431,164	665,9798	100	100
TREE					
Desicion Stump	0,6981	303,2187	497,9172	70,3256	74,7646
M5P	0,922	150,9841	252,8864	35,0178	37,9721
Random Forest	0,8196	211,6876	403,4439	49,0968	60,579
Random Tree	0,7029	254,4593	519,1927	59,0168	59,0168
REP Tree	0,594	289,226	544,7618	67,0803	81,7985

Tablo 4'te, COCOMONASA veri setine uygulanan Makine Öğrenmesi algoritmalarının performans değerlendirme sonuçları belirtilmiştir. COCOMONASA veri setinde en iyi tahmini, Tree grubundan 0,922 korelasyon katsayısı ve %35,0178 RAE hata payı ile M5P algoritması gerçekleştirmiştir. Rules grubundan ZeroR algoritması COCOMO81 veri setinde olduğu gibi COCOMONASA veri setinde de en kötü tahmin sonucunu üretmiştir. ZeroR algoritmasından sonra 0,4577 korelasyon katsayısı ve %60,5639 RAE hata payı ile Decision Table algoritması en kötü performası sergilemiştir.

Tablo 5. COCOMONASA2 veri setine uygulanan algoritmaların performans ölçümleri

COCOMONASA2 Veri Seti					
ALGORİTMALAR	Ölçütler				
FONKSİYONLAR	**Korelasyon katsayısı**	**MAE**	**RMSE**	**RAE(%)**	**RRSE(%)**
Gaussian Processes	0,5966	535,8033	1003,274	82,9528	87,8166
Linear Regression	0,7294	430,7269	826,1252	66,6849	72,3107
Multilayer Perceptron	0,6147	653,0797	1313,2285	101,1095	114,9468
SMOreg	0,425	737,3497	1368,5567	114,1562	119,7897
LAZY					
IBK (K-nearest neighbor)	0,659	445,7796	924,0382	69,0154	80,881
KStar	0,7091	376,3781	821,2064	58,2707	71,8801
LWL	0,8183	332,7218	652,8788	51,5118	57,1464
META					
Additive Regression	0,7974	334,6625	682,1185	51,8123	59,7058
Attribute Selected Classifier	0,7168	379,1302	788,339	58,6968	69,0033
Bagging	0,7298	365,3964	778,686	56,5705	68,1583
CVParameter Selection	0,7415	365,1982	759,6982	56,5398	66,4963
Multi Schema	0,7392	370,4636	761,003	57,355	66,6105
Random Comittee	0,7595	358,2204	739,3583	55,4595	64,716
Randomizable Fitered Classifer	0,7158	371,1652	789,8391	57,4636	69,1346
Random SubSpace	0,6729	407,627	835,7404	63,1086	73,1523
Regresiyon By Discretization	0,7069	424,2558	799,3404	65,6831	69,9662
Weighted Instances Handler Wrapper	0,7415	365,1982	759,6982	56,5398	66,4963
MISC					
Input Maped Classifier	0,7415	365,1982	759,6982	56,5398	66,4963
RULES					
Decision Table	0,2525	564,7407	1186,1157	87,4329	103,8206
M5 Rules	0,7042	360,4728	805,1669	55,8082	70,4762
ZeroR	-0,3101	645,9132	1142,4663	100	100
TREE					
Desicion Stump	0,4183	567,6411	1063,2781	87,8819	93,0687
M5P	0,7171	348,3774	788,2642	53,9356	68,9967
Random Forest	0,7415	365,1982	759,6982	56,5398	66,4963
Random Tree	0,4882	459,4538	1016,6788	71,1324	88,9898
REP Tree	0,3464	540,5725	1094,5921	83,6912	95,8096

Tablo 5'te COCOMONASA2 veri setine uygulanan Makine Öğrenmesi algoritmalarının performans değerlendirmeleri verilmiştir. Algoritmaların performansları incelendiğinde en iyi tahmin sonucunun Lazy grubundan LWL algoritmasına ait olduğu görülmektedir. LWL algoritması var olan özellikleri ile çalıştırıldığında 0,4864 korelasyon katsayısı ve %79,8692 RAE hata payı ile tahminde bulunmaktadır. Fakat WEKA aracı kullanılarak özellikler penceresinden sınıflandırma özelliği Random Forest olarak değiştirildiğinde çok daha iyi performans sergilediği görülmüştür. WEKA aracı kullanılarak her bir sınıflandırma algoritmasının parametreleri değiştirilerek farklı sonuçlar elde edilebilmektedir. Bu çalışmada doğru bir karşılaştırma analizinin olabilmesi için bütün algoritmalara parametre olarak Random Forest algoritması verilmiştir. Bu veri setinde de diğer veri setlerinde olduğu gibi en kötü performansı ZeroR algoritması göstermiştir. ZeroR algoritmasından sonra en kötü tahmin sonucu Rules grubundan Decision Table algoritmasına aittir.

Aşağıdaki Tablo 6, Tablo 7 ve Tablo 8'de algoritmaların COCOMO81, COCOMONASA ve COCOMONASA2 veri setlerinde gösterebilecekleri en iyi performanslar belirtilmiştir. Bu performans ölçümlerinin elde edilebilmesi için ilgili algoritmaların özellikler penceresinden paremetre değerleri değiştirilerek bütün olasılıklar denenmiştir. En iyi tahmin sonuçları aşağıdaki tablolarda gösterilmiştir.

Tablo 6. COCOMO81 veri setine uygulanan algoritmaların en iyi performans ölçümleri

COCOMO81 veri seti			
ALGORİTMA	**En iyi Algoritma**	**Korelasyon katsayısı**	**RAE(%)**
LWL	Random Comittee	0,8331	54,4355
Additive Regression	Random Comittee	0,8282	49,6681
Attribute Selected Classifier	Random Comittee	0,8656	50,3006
Bagging	Random Comittee	0,7235	63,8393
CVParameter Selection	Random Comittee	0,8764	48,5789
Multi Schema	Random Comittee	0,8529	52,0718
Random Comittee	Random Comittee	0,8764	48,5789
Randomizable Fitered Classifer	IBK (K-nearest neighbor)	0,7722	57,5024
Random SubSpace	Random Comittee	0,7109	63,5379
Regresiyon By Discretization	Multilayer Perceptron	0,8547	56,89
Weighted Instances Handler Wrapper	Random Comittee	0,8227	51,606
Input Maped Classifier	Random Comittee	0,8764	48,5789

Tablo 6 incelendiğinde en iyi tahmin sonucunun Random Committee algoritmasına ait olduğu görülmektedir. Bu performans değerinin elde edilebilmesi için şöyle bir yol izlenmiştir: Sınıflandırma algoritması olarak Random Committe seçildikten sonra özellikler penceresinden classifier özelliği yine Random Committee seçilmiştir. Seçilen Random Comittee sınıflandırma algoritmasının classifier özelliği Random Tree olarak belirlenip algoritma çalıştırılmıştır. Buna benzer birçok değişik senaryo gerçekleştirilmiştir. En iyi performans sonucu parametre olarak Random Committee algoritmasının verilmesi ile elde edilmiştir. Input Maped Classifier algoritması ve CVParemeter Selection algoritmaları da aynı performansı sergilemiştir. Bu iki algoritmanın da özellikler penceresinden classifier özelliği Random Comittee algoritması olarak atanmıştır. Bu şekilde en iyi performansı sergilenmiştir.

Tablo 7. COCOMONASA veri setine uygulanan algoritmaların en iyi performans ölçümleri

COCOMONASA			
ALGORİTMA	**En iyi algoritma**	**Korelasyon katsayısı**	**RAE(%)**
LWL	Linear Regression	0,8945	43,6739
Additive Regression	M5P	0,9371	31,7645
Attribute Selected Classifier	Multilayer Perceptron	0,8963	36,8339
Bagging	M5P	0,9175	29,0556
CVParameter Selection	M5P	0,922	35,0178
Multi Schema	M5P	0,922	35,0178
Random Comittee	Randomizable Fitered Classifer	0,924	34,0002
Randomizable Fitered Classifer	SMOreg	0,9161	30,4218
Random SubSpace	Regresiyon By Discretization	0,817	64,1904
Regresiyon By Discretization	Randomizable Fitered Classifer	0,8309	60,419
Weighted Instances Handler Wrapper	M5P	0,922	35,0178
Input Maped Classifier	M5P	0,922	35,0178

Tablo 7 incelendiğinde en iyi tahmin sonucunun Additive Regression algoritmasına ait olduğu görülmektedir. Additive Regression algoritması Meta grubunda bulunan bir algoritmadır. Bu algoritma parametre olarak başka bir algoritma ya da yine kendisini alabilmektedir. En iyi tahmin sonucunun elde edilebilmesi için parametre olarak bütün algoritmalar denenmiştir. Parametre olarak Tree grubundan M5P algoritması verildiğinde en iyi tahmin sonucu elde edilmiştir. Genel olarak COCOMONASA veri setinde yazılım maliyet tahmini için algoritmalar uygulandığında, algoritmalara parametre olarak Tree grubundan M5P algoritması verildiğinde en iyi performans değerleri elde edilmiştir.

Tablo 8. COCOMONASA2 veri setine uygulanan algoritmaların en iyi performans ölçümleri

COCOMONASA2			
ALGORİTMA	**En iyi algoritma**	**Korelasyon katsayısı**	**RAE(%)**
LWL	Random Comittee	0,8309	50,4804
Additive Regression	Random Forest	0,7974	51,8123
Attribute Selected Classifier	Random Comittee	0,774	56,0921
Bagging	Random Tree	0,7605	53,6467
CVParameter Selection	Random Comittee	0,783	54,5614
Multi Schema	Random Comittee	0,7923	55,2471
Random Comittee	Random Tree	0,783	54,5614
Randomizable Fitered Classifer	Random Comittee	0,8043	53,5075
Random SubSpace	Random Forest	0,6729	63,1086
Regresiyon By Discretization	Random Comittee	0,7793	64,1228
Weighted Instances Handler Wrapper	Random Comittee	0,7881	54,7878
Input Maped Classifier	Random Comittee	0,783	54,5614

Tablo 8'de WEKA aracı kullanılarak COCOMONASA2 veri setine uygulanan Makine Öğrenmesi algoritmalarının performans değerlendirmeleri verilmiştir. Algoritmaların performansları incelendiğinde en iyi performansın Lazy grubundan LWL algoritmasına parametre olarak Random Committee algoritmasının verilmesi ile elde edildiği görülmüştür. Tablo 8 incelendiğinde COCOMONASA2 veri seti üzerinde parametre olarak Random Committee algoritmasının verilmesi performansı olumlu yönde etkilediği görülmüştür.

4. Sonuç

Yazılım projelerinin maliyet tahminin gerçekleştirilmesi için farklı tahmin yöntemleri geliştirilmiştir. Makine Öğrenmesi yöntemine dayalı geliştirilen modeller bu yöntemlerden bazılarıdır (Kayakuş, 2021). Bu çalışmada yazılım maliyet tahmini için PROMISE veri deposundan alınan COCOMO81, COCOMONASA ve COCOMONASA2 veri setleri kullanılarak Makine Öğrenmesi algoritmaları test edilmiştir. WEKA aracı kullanılarak yapılan tahmin sonuçları incelendiğinde, algoritmaların hata oranlarının ve korelasyon katsayılarının uygulandıkları veri setlerine göre değişkenlik gösterdiği belirlenmiştir. Bazı algoritmaların bazı veri setlerinde çok iyi çalışırken farklı parametrelerle ve farklı veri setlerinde kötü sonuçlar verebileceği gözlemlenmiştir. Ayrıca veri setlerindeki öz niteliklerin, tahmin sonucunu çok etkilediği fark edilmiştir. COCOMONASA2 veri setinde 24 tane öz nitelik bulunmaktadır. Bu 24 öz nitelik kullanılarak Makine Öğrenmesi algoritmaları veri setine uygulandığında çok daha kötü sonuçlar elde edilmiştir. COCOMONASA2 veri setinden maliyete etkisi olmayan 7 tane öz nitelik çıkarıldığında çok daha iyi performans değerleri elde edilmiştir. Ayyıldız (2007) çalışmasında, yazılım projelerinin maliyet tahmini için yeni bir ölçüt kümesi sunmuştur. Araştırmacı doğru ve nitelikli ölçüt kümesinin tahmin sonucunu ciddi biçimde etkilediğini ispatlamıştır.

Bazı algoritmalar, kendi parametrelerine ek olarak başka bir sınıflandırıcı ve onun parametrelerini alabilen algoritmalardır. Doğru bir analiz karşılaştırması olabilmesi için bu algoritmaların hepsine Random Forest algoritması temel sınıflandırıcı olarak seçilip testler yapılmıştır.

Test sonuçları incelendiğinde COCOMO81 veri setinde en iyi tahmin algoritmasının Meta grubundan 0,8095 korelasyon katsayısı ve %51,9767 RAE hata payı ile Additive Regression algoritması olduğu, COCOMONASA veri setinde en iyi tahmin algoritmasının Tree grubundan 0,922 korelasyon katsayısı ve %35,0178 RAE hata payı ile M5P algoritması olduğu, COCOMONASA2 veri setinde en iyi tahmin algoritmasının Lazy grubundan 0,8183 korelasyon katsayısı ve %51,5118 RAE hata payı ile LWL algoritmasının olduğu gözlemlenmiştir.

Kendi parametrelerine ek olarak başka bir sınıflandırıcı ve onun paremetrelerini alan algoritmalar (Meta başlığı altındaki bütün algoritmalar, Lazy grubundan LWL, Rules grubundan Input Mapped) için bütün olasılıklar denenmiştir. COCOMO81 veri seti kullanılarak yapılan tahmin algoritmalarından en iyi performansı 0,8764 korelasyon katsayısı ve %48,5789 RAE hata payı ile Meta grubundan Random Committee algoritması gerçekleştirmiştir. COCOMONASA veri seti kullanılarak yapılan tahmin algoritmalarından en iyi performansı 0,9371 korelsayon katsayısı ve %31,7645 RAE hata payı ile M5P algoritmasının parametre olarak verilmesi ile Additive Regression algoritması gerçekleştirmiştir. COCOMONASA2 veri setine uygulanan algoritmalar incelendiğinde, en iyi performansın LWL algoritmasına parametre olarak Random Committee algoritmasının verilmesi ile elde edildiği belirlenmiştir.

Bu çalışmada WEKA aracı ile Makine Öğrenmesi algoritmalarının PROMISE veri deposunda bulunan COCOMO81, COCOMONASA ve COCOMONASA2 veri setleri kullanılarak yazılım maliyet tahmininde gösterdikleri performanslar incelenmiştir. Sonuçlar incelendiğinde bir algoritmanın her zaman en iyi sonucu üretmediği, farklı veri setleriyle farklı sonuçlar ürettiği bilgisine ulaşılmıştır.
Bu çalışma sayesinde yazılım maliyet tahmini için hangi Makine Öğrenmesi algoritmalarının kullanılabileceği, bu algoritmaların COCOMO81, COCOMONASA ve COCOMONASA2 veri setlerine uygulandığında tahmin sonuçlarının neler olabileceği ve en iyi çalışan algoritmaların hangileri olduğu bilgisine ulaşılmıştır.

Marapelli (2019) çalışmasında Lineer Regresyon ve K-en yakın komşuluk algoritmalarının performansalarını WEKA aracı kullanarak değerlendirmiştir. Araştırması çalışmasında COCOMO81, COCOMONASA ve COCOMONASA2 veri setleri için sadece 2 tane algoritma denemiştir. Bizim çalışmamızda 27 tane algoritmanın test sonuçları analiz edilmiştir.

Bu sayede bu çalışma daha kapsamlı bir çalışma olmuştur. Daha sonraki bir çalışmada, farklı metodolojiler ile hazırlanmış yazılım projelerinin veri setleri kullanılarak WEKA aracında testler yapılabilir. Test sonuçları analiz edilerek COCOMO veri setleri kullanılarak yapılan test sonuçları ile karşılaştırılabilir ya da bu çalışmada kullanılan Makine Öğrenmesi algoritmaları farklı bir platformda tekrar test edilerek sonuçlar karşılaştırılabilir. Genetik Algoritmalar ve Bulanık Mantık gibi Yapay Zeka'nın diğer yöntemleri de yazılım projelerinin maliyet tahmini için kullanılabilir.

Kaynakça

Adalier, O. (2008). Yapay Zekâ Yöntemleri İle Yazılım Projelerinde Maliyet Kestirimi, Doktora Tezi, Ege Üniversitesi, İzmir.

Alpaydın, E. (2011). Yapay Öğrenme, Boğaziçi Üniversitesi Yayınevi, İstanbul.

ANSI/PMI. (2008). *A Guide to the Project Management Body of Knowledge (PMBOK Guide)*, Project Management Institute, Inc., Newtown Square, Pennysylvania, ISBN: 978- 1-933890-51-7.

Aydın, S. Özkul, A. E. (2015). Veri Madenciliği Ve Anadolu Üniversitesi Açıköğretim Sisteminde Bir Uygulama, *Eğitim ve Öğretim Araştırmaları Dergisi*, 4(3), 38.

Ayyıldız, M. (2007). Yazılım Projeleri Ölçüm Sonuçları Veri Tabanının Oluşturulması ve Yeni Yazılım Projelerinin Maliyet Tahmininde Kullanımı, Doktora Tezi, Yıldız Teknik Üniversitesi, İstanbul.

Güven Aydın, Z.B. ve Samli, R. (2020). A Comparison of Software Defect Prediction Metrics Using Data Mining Algorithms, Journal of Innovative Science and Engineering. 4(1), 11-21.

Gültekin, M. (2019). Makine Öğrenmesi Tabanlı Yazılım Maliyet Tahmini Yöntemlerinin Karşılaştırmalı Analizi, Doktora Tezi, Yıldız Teknik Üniversitesi, İstanbul.

Güven Aydın, Z.B. (2021), Makine Öğrenmesi Yöntemleri İle Yazılım Hata Tahmini, Doktora Tezi, İstanbul Üniversitesi-Cerrahpaşa, İstanbul.

Kaluza, B. (2016). Machine Learning in Java, Pact Publishing.

Karataş, E.K. (2011). Yapay Sinir Ağları İle Yazılım Projesi Maliyet Tahmini, Yüksek Lisans Tezi, İstanbul Üniversitesi, İstanbul.

Kayakuş, M. (2021). Yazılım Çaba Tahmininde Yapay Sinir Ağları İçin Optimum Yapının Belirlenmesi, Avrupa Bilim ve Teknoloji Dergisi, 22, 43-48.

Kızılkaya, Y. M. ve Oğuzlar, A. (2018). Bazı Denetimli Öğrenme Algoritmalarının R Programlama Dili İle Kıyaslanması, Karadeniz Uluslararası Bilimsel Dergi, 37(37), 90 – 98.

Kulter, Y., Turhan, B. ve Baner, A. (2009). Ensemble of neural networks with associative memory (ENNA) for estimating software development costs, Knowledge-Based Systems, 395-402.

Kumari, S. ve Pushkar, S. (2013). Performance Analysis of the Software Cost Estimation Methods: A Review, International Journal of Advanced Research in Computer Science and Software Engineering, 229-238.

LIU, Q. ve Mintram, R.C. (2005). Preliminary Data Analysis Methods in Software Estimation, Software Quality Journal, 13, 91-115.

Marapelli, B. (2019). Software Development Effort Duration and Cost Estimation using Linear Regression and K-Nearest Neighbors Machine Learning Algorithms, International Journal of Innovative Technology and Exploring Engineering (IJITEE), 9(2), 2278-3075.

Niranjan, A., Prakash, A., Veena, N., Geetha, M., Deepa S. ve Venugopal, R. (2017). EBJRV: An Ensemble of Bagging, J48 and Random Committee by Voting for Efficient Classification of Intrusions, IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE), Dehradun, India.

Prabhakar ve Dutta, M. (2013). Application of machine learning techniques for predicting software effort, Elixir Comp. Sci. & Eng. 56, 13677-13682.

Prowmes blog. (2019). Makine Öğrenmesi. 14 Ağustos 2020 tarihinde http://www.prowmes.com/blog/makine-ogrenmesi adresinden erişildi.

Sezer, A. (2008). Yazılım Projelerinde Yapay Sinir Ağı Uygulaması İle Maliyet Tahmini, Yüksek Lisans Tezi, Haliç Üniversitesi, İstanbul.

Sommerville, I. (2000). *Software Engineering (6th Edition)*, Addison-Wesley .

TDK. (2018). Yazılım. 22 Haziran 2018 tarihinde http://www.tdk.gov.tr adresinden erişildi.

Torkul, O., Gülseçen, S., Uyaroğlu, Y., Çağıl, G., Uçar, M. K. (2017). Mühendislikte Yapay Zeka Uygulamaları, Sakarya Üniversitesi Kütüphanesi Yayınevi, Sakarya

Yergök, G. ve Acı, M. (2019). Toplu Yemek Üretiminde Günlük Talep Tahmini için Alternatif Bir Yaklaşım: Öğrenci Regresyon, Avrupa Bilim ve Teknoloji Dergisi, özel sayı, 64-73.

Wikipedia. (2020). Korelasyon. 25 Ocak 2021 tarihinde https://tr.wikipedia.org/wiki/ adresinden erişildi.