KOCAELİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

YÜKSEK LİSANS TEZİ

YAPAY SİNİR AĞLARI VE K-MEANS KULLANARAK SINIR DEĞERLERİNE GÖRE YAZILIM EFOR TAHMİNİ

ÖMER FARUK SARAÇ

KOCAELİ 2014

KOCAELİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

YÜKSEK LİSANS TEZİ

YAPAY SİNİR AĞLARI VE K-MEANS KULLANARAK SINIR DEĞERLERİNE GÖRE YAZILIM EFOR TAHMİNİ

ÖMER FARUK SARAÇ

Doç.Dr. Nevcihan DURU
Danışman, Kocaeli Üniv. .....

Doç.Dr. Banu DİRİ
Jüri Üyesi, Yıldız Teknik Üniv. .....

Yrd.Doç.Dr. Ahmet SAYAR
Jüri Üyesi, Kocaeli Üniv. .....

Tezin Savunulduğu Tarih: 02.12.2013

ÖNSÖZ VE TEŞEKKÜR

Bu çalışmada yazılım proje yönetiminde efor tahmini ile ilgili bir esneklik çalışması yapılmıştır. Yazılım proje yönetiminde, gerekli olan zamanın tespiti efor tahmini olarak ifade edilmektedir ve bu tahmin, proje planlarına, dolayısıyla proje başarısına doğrudan etki etmektedir. Mevcut yapılar genel olarak tek bir tahmin değeri üretmektedir ve bu durumda başarı oranı yeterli bilgi sunmayabilir. Tez çalışması ile amaçlanan, tahmin değerini destekleyecek olası esneklik, üst ve alt limit değerleri hesaplayarak, efor tahmin sürecinde karar destek aşamasına katkı sağlamaktır. Bu amaç için ortaya yapısal bir model konulmuş ve bu model, geliştirilen araç ile desteklenmiştir. Aynı zamanda araç üzerinde model test edilmiş ve beklentiler test sonuçlarına göre değerlendirilmiş ve olumlu neticelerin alındığı gözlemlenmiştir.

Yüksek lisans eğitimim süresince bana her konuda yardımcı olan, yol gösteren, tez kapsamında veya tez alt yapısı ile ilgili diğer unsurlarda sorunlarımı çözmemde yol gösteren, değerli zamanından bana ve sorunlarıma vakit ayıran, akademik çalışma noktasında farkındalık kazandıran değerli hocam Doç. Dr. Nevcihan DURU' ya, yine tüm yüksek lisans eğitimim boyunca kendisinden ders aldığım değerli hocalarıma, Kocaeli Üniversitesi Bilgisayar Mühendisliği Bölümüne, böyle bir imkânın oluşmasına katkıda bulunan kurumuma ve iş arkadaşlarıma, bilgi paylaşımında bulunan tüm değerli hocalar, akademisyen ve araştırmacılara ve bugüne kadar eğitim hayatımda bana katkısı olan tüm değerli büyüklerim ve arkadaşlarıma teşekkürlerimi sunarım.

Ayrıca maddi ve manevi desteklerini tüm hayatım boyunca eksiltmeden sunan aileme ve dostlarıma da şükranlarımı sunarım.

Eylül – 2013 Ömer Faruk SARAÇ

İÇİNDEKİLER

ÖNSÖZ VE TEŞEKKÜR i
İÇİNDEKİLER ii
ŞEKİLLER DİZİNİ iii
TABLOLAR DİZİNİ iv
SİMGELER DİZİNİ VE KISALTMALAR v
ÖZET vi
ABSTRACT vii
GİRİŞ 1
1. GENEL BİLGİLER 5
1.1. Yazılım Proje Yönetimi 5
1.2. Yazılım Efor Tahmini 7
1.3. COCOMO 9
1.4. Veri Madenciliği 12
1.5. Yapay Sinir Ağları 14
1.6. K-Means Algoritması 17
1.7. Yazılım Mühendisliği ve Efor Tahmini İle İlgili Çalışmalara Genel Bakış 18
1.8. Yazılım Efor Tahmininde COCOMO Modelini Baz Alan Çalışmalar 20
2. MALZEME VE YÖNTEM 24
2.1. Girdi Dosyası 26
2.2. Eğitim ve Test Kümeleri 27
2.3. Yapay Sinir Ağı Yapısı 30
2.4. K-Means Kümeleri 33
2.5. YSA ile K-Means Kümelerinin Birleştirilmesi 36
2.6. Sonuç Katmanındaki Değerler 38
2.7. Geliştirilen Araç ve Sistem İşleyişi 39
3. BULGULAR VE TARTIŞMA 48
3.1. SWR Kullanılarak Yapılan Testler 50
3.1.1. NASA 60 veri kümesi üzerinde yapılan çalışma 50
3.1.2. COCOMO 81 veri kümesi üzerinde yapılan çalışma 54
3.1.3. NASA 93 veri kümesi üzerinde yapılan çalışma 59
3.1.4. SWR Kullanarak Yapılan Testlerin Genel Değerlendirmesi 63
3.2. SWR Olmaksızın Farklı Küme Elemanları Kullanılarak Yapılan Testler 64
3.2.1. NASA 60 veri kümesi üzerinde yapılan çalışma 65
3.2.2. COCOMO 81 veri kümesi üzerinde yapılan çalışma 68
3.2.3. NASA 93 veri kümesi üzerinde yapılan çalışma 72
3.2.4. SWR Kullanılmayarak Yapılan Testlerin Genel Değerlendirmesi 76
4. SONUÇLAR VE ÖNERİLER 77
KAYNAKLAR 80
EKLER 84
KİŞİSEL YAYINLAR VE ESERLER 103
ÖZGEÇMİŞ 104

ŞEKİLLER DİZİNİ

Şekil 1.1. PMBOK Süreçleri ve Etkileşim Düzeyleri 6
Şekil 1.2. Veri Madenciliği Aşamaları 13
Şekil 1.3. Algılayıcının Genel Yapısı 14
Şekil 1.4. Temel Bir Yapay Sinir Ağı Modeli 15
Şekil 1.5. Bilinen Aktivasyon Fonksiyonları 16
Şekil 2.1. Geliştirilen Sistemin Genel Yapısı 24
Şekil 2.2. Eğitim ve Test Küme Veri Yapısı 29
Şekil 2.3. YSA Yapısının Gösterimi 31
Şekil 2.4. Program Üzerindeki Girdi Alanları 39
Şekil 2.5. Örnek Program Çıktısı 40
Şekil 2.6. Program Sınıf Özellikleri ve Metotlar 41
Şekil 2.7. Program Akışının Genel Görünümü 43
Şekil 3.1. SWR İle Nasa 60 Veri Kümesi İçin Sonuç Değerleri 54
Şekil 3.2. SWR İle COCOMO 81 Veri Kümesi İçin Sonuç Değerleri 58
Şekil 3.3. SWR İle NASA 93 Veri Kümesi İçin Sonuç Değerleri 63
Şekil 3.4. SWR Olmaksızın NASA 60 Veri Kümesi Sonuç Değerleri 68
Şekil 3.5. SWR Olmaksızın COCOMO 81 Kümesi Sonuç Değerleri 71
Şekil 3.6. SWR Olmaksızın NASA 93 Veri Kümesi Sonuç Değerleri 75

TABLOLAR DİZİNİ

Tablo 1.1. Proje Türüne Göre COCOMO Sabit Değerleri 10
Tablo 1.2. COCOMO Parametreleri 11
Tablo 3.1. SWR İle Nasa 60 Veri Kümesi İçin Hedef ve Sonuçlar 51
Tablo 3.2. SWR İle Nasa 60 Veri Kümesi İçin MRE Değerleri 52
Tablo 3.3. SWR İle COCOMO 81 Veri Kümesi İçin Hedef ve Sonuçlar 55
Tablo 3.4. SWR İle COCOMO 81 Veri Kümesi İçin MRE Değerleri 57
Tablo 3.5. SWR İle NASA 93 Veri Kümesi İçin Hedef ve Sonuçlar 59
Tablo 3.6. SWR İle NASA 93 Veri Kümesi İçin MRE Değerleri 61
Tablo 3.7. SWR Olmaksızın NASA 60 Veri Kümesi İçin Hedef ve Sonuçlar 66
Tablo 3.8. SWR Olmaksızın NASA 60 Veri Kümesi MRE Değerleri 67
Tablo 3.9. SWR Olmaksızın COCOMO 81 Veri Kümesi Hedef ve Sonuçlar 69
Tablo 3.10. SWR Olmaksızın COCOMO 81 Veri Kümesi MRE Değerleri 70
Tablo 3.11. SWR Olmaksızın NASA 93 Veri Kümesi Hedef ve Sonuçlar 72
Tablo 3.12. SWR Olmaksızın NASA 93 Veri Kümesi MRE Değerleri 74

SİMGELER DİZİNİ ve KISALTMALAR

Kısaltmalar

ANN	:	Artificial Neural Network (Yapay Sinir Ağı)
COCOMO	:	Constructive Cost Model (Yapıcı Maliyet Modeli)
EAF	:	Effort Adjustment Factor (Çaba Düzeltme Faktörü)
FPA	:	Function Point Analysis (Fonksiyon Noktası Analizi)
KNN	:	K-Nearest Neighborhood (K En Yakın Komşu)
LOC	:	Lines Of Code (Kod Satırları)
MRE	:	Magnitute of Relative Error (Bağıl Hata Büyüklüğü)
PMBOK	:	Project Management Body Of Knowledge (Proje Yönetimi Bilgi Birikimi Kılavuzu)
PMI	:	Project Management Institute (Proje Yönetimi Enstitüsü)
SEER-SEM	:	SEER for Software (Yazılım İçin SEER)
SLIM	:	Software Lifecycle Management (Yazılım Yaşam Döngüsü Yönetimi)
SWR	:	Sampling With Replacement (Yerine Koyarak Örnekleme)
YSA	:	Yapay Sinir Ağı

YAPAY SİNİR AĞLARI VE K-MEANS KULLANARAK SINIR DEĞERLERİNE GÖRE YAZILIM EFOR TAHMİNİ

ÖZET

Efor tahmini, yazılım proje yönetiminde, projenin ihtiyaç duyduğu kaynakların belirlenmesinde kullanılan yöntemlerdendir. Proje için gereken adam/zaman değeri hesaplanır ve proje planı bu bilgiler göz önünde bulundurularak yapılır. Efor tahmini için çeşitli yöntemler vardır. En bilinen yöntemlerden biri COCOMO adındaki parametrik modeldir. Esasında bir hesaplama fonksiyonu olan modelde, parametreler proje ve yazılım ortamı ile ilgili değerlendirme niteliklerine atanmış kategorik ifadelerin sayısal karşılığı olan değerlerle bir hesap yapar ve sonuçta adam/ay hesabına göre bir tahmin değeri oluşturulur. Ek olarak, efor tahmininde, önceki proje verilerinden daha etkin şekilde yararlanmak için veri madenciliği teknikleri de sürece dahil edilmiştir. En çok kullanılan yöntemlerden biri yapay sinir ağlarıdır (YSA). YSA esneklik ve kabiliyetleri ile COCOMO bazı modellerde birleştirilmiş ve daha başarılı sonuçlar alındığı görülmüştür. Bu çalışmada da benzer şekilde bir yapı söz konusudur. Ancak mevcut çalışmalarda ortaya tek bir değer çıkarılmaktadır. Bu şekilde karar aşamasında yeterli bilgi olmayabilir. Bu çalışmada önerilen, tek bir değer üretmek yerine, geçmiş bilgilere dayanılarak hesaplanan bir aralık değer kümesi oluşturmaktır. Buna göre, YSA ve COCOMO kullanılmakla beraber, K-Means algoritmasıyla olası üst ve alt limitler hesaplanmıştır. Bu şekilde daha başarılı tahminler yapılacağı, karar aşamasında yardımcı değerlerin üretilebileceği öngörülmüştür. Test verileri ile yapılan denemelerde bu öngörünün gerçekleşebileceği gözlemlenmiştir.

Anahtar Kelimeler: COCOMO, K-Means, Yapay Sinir Ağları, Yazılım Efor Tahmini.

SOFTWARE EFFORT ESTIMATION WITH BOUNDARIES USING ARTIFICIAL NEURAL NETWORKS AND K-MEANS

ABSTRACT

Effort estimation is one of the methods used to determine the resources needed for a project in software project management. Required man/time value for project is calculated and project plan is made considered to this value. There are various methods for effort estimation. One of the best-known method is a parametric model which is called COCOMO. In this model, that is basically a function, a calculation process is made via numerical equivalent values for categorical parameters which are about project and software environment and eventually a man/month estimation is calculated. In addition, data mining techniques are included in effort estimation process in order to benefit more from previous project management data. One of the most widely used methods is artificial neural networks (ANN). ANN flexibilities and capabilities are combined with COCOMO in some models and more successful results are obtained. There is a similar structure in this study. However there is only single value is outputted in these researches. In this way, there may not be enough information in decision making process. Proposed model in this study is that instead of producing single output value, a value range set is generated. In addition to ANN and COCOMO usage, possible upper and lower limits are calculated with K-Means algorithm. Thus, it is intented to make more precise estimations and give helpful values for decision making. Experimental results show that this prescience can be achieved.

Keywords: COCOMO, K-Means, Artificial Neural Networks, Software Effort Estimation.

GİRİŞ

Yazılım proje yönetimi, içerisinde farklı faz ve işlerin olduğu, planlama ve kontrol ağırlıklı bir süreçler bütünüdür. Sahip olduğu tanımlanmış aşamalarıyla planlama yapılmasını ve bu planlamaya göre işleyişin gerçekleştirilmesini sağlar. Ek olarak, proje ile ilgili kontrol ve düzenleme mekanizmaları işletilir[1].

Bir proje yönetiminde planlama en önemli aşamalardan biridir. Neticede yapılacak tüm işler, bu planlar nezdinde gerçekleştirilmektedir. Bu nedenle planlama aşamaları için yardımcı olacak çeşitli teknikler ortaya atılmıştır. Yazılım proje yönetimi söz konusu olduğunda, planlama, geliştirilecek program ya da servisin ortaya çıkması için gerekecek adam/zaman değeri üzerinden gidecektir. Bu değer hesaplandıktan-tahmin edildikten-sonra planlamaya ait diğer öğeler (bütçe, takvim gibi) de belirlenebilecektir[2]. Yazılım projesinin ihtiyaç duyduğu kaynağın tahmini konusu, yazılım efor tahmini olarak adlandırılmaktadır[3]. Efor tahmini, geliştirilecek olan sistemin ihtiyaç duyacağı kaynakları ve maliyetleri hesaplamada yardımcı olmaktadır.

Bir proje için efor tahmininin doğruluğu, proje başarısına doğrudan etki yapmaktadır. Planlar, yapılan tahminlere göre şekillenir ve buna göre diğer önemli unsurlar, bütçe, takvim, tedarik süreçleri belirlenir. Bu nedenle, bir proje yöneticisi ya da proje ekibi için en önemli konulardan biri, efor tahminini yüksek başarı yüzdesiyle yapmaktır. Gerçekleşen zaman-maliyet miktarı, planlananı aştığı durumlarda proje başarısızlığa uğrayabilir; tersi bir durumda ise, kaynakların doğru kullanılmaması gibi bir sorun ortaya çıkabilir[2]. Efor tahmininin doğru bir şekilde yapılabilmesi için proje yöneticileri yardımcı teknik ve yöntem arayışı içindedirler.

Mevcut zorluklar devam ederken, bu zorlukları kaldıracak ya da en aza indirecek teknikler de gelişmektedir. Efor tahmini değerlerinin başarı oranını artırabilecek birçok teknik ve yöntem önerilmiştir. Bunlar temelde 3 kategoriye ayrılmaktadır: formel modeller, uzman tabanlı modeller ve her iki modelin belli oranlarda kullanıldığı kombinasyon tabanlı modeller[3].

Her bir kategori içerisinde çeşitli teknikler mevcuttur. Formel modeller temelde matematiksel hesaplama yöntemlerine dayanmaktadır. Bu nedenle metodun uygulanması daha kolay olmaktadır. Bu metotlar içerisinde de, mevcut tarihi ve basitliği nedeniyle en bilinenlerden biri COCOMO' dur. Constructive Cost Model ifadesinin kısaltılmış şekli olan COCOMO, basite indirgenirse, bir hesaplama fonksiyonudur. Proje yönetimi ve yazılım geliştirme ortamına bağımlı çeşitli parametreleri vardır ve bu parametreler için de değer aralıkları bulunmaktadır. 1981 yılında B.Boehm tarafından ortaya atılmıştır[4].

Efor tahmininin başarı oranını artırmak için veri madenciliği yöntemleri de kullanılmıştır[5,6,7,8,47]. Proje yönetimi sırasında ortaya çıkan veri tabanından faydalanılarak, yeni projeler için tahminsel modeller geliştirilmiştir. Yapay sinir ağları (YSA), bu yöntemlerden çokça tercih edilenlerden biri olmuştur[5,6,8,48]. Karmaşık problemleri modelleyebilmesi ve tahmin değeri üretebilmesi gibi özelliklerinden dolayı tercih edildiği ifade edilmiştir[5]. YSA, tek başına uygulama özelinde girdi verisiyle çalışacağı gibi, birçok çalışmada da görüldüğü üzere, COCOMO modeli üzerine de uygulanmıştır. Modelin geçerliliği ve standart yapısı, YSA üzerine uygulanmasına imkân sağlamaktadır. Buna göre, YSA girdi katmanında, girdi elemanlarına COCOMO parametreleri değer olarak verilir ve sonuçta da efor tahmini değeri üretilir. Ek olarak, YSA dışında tercih edilen modeller de mevcuttur[47,49,50]. Bu şekilde farklı yaklaşımlar kullanılarak efor tahmin değerleri üretme aşamasında başarı oranı artırılmaya çalışılmıştır.

Yazılım proje yönetiminde planlama yapılırken, belirlenen kaynak ve zaman değerleri için sadece tek bir değer belirlenmemektedir. Proje planının daha sağlıklı yürütülebilmesi için güven aralıkları, değer aralıkları da belirlenir. Buna göre planlama ve destek sistemleri oluşturulur[1,2,9].

Efor tahmini söz konusu olduğunda da benzer bir durum geçerlidir. Neticede tahmin sistemleri, karar destek aşamasında yardımcı olacak önemli bir bilgi kaynağıdır. Planlama ve bu plan içerisinde yer alacak kaynaklar, proje yöneticisinin kararına bağlıdır. Yani, efor tahmini ile ortaya çıkarılan değer için belli bir esneme miktarı göz önüne bulundurulacaktır.

Proje yöneticisinin kararına yardımcı olacak tekniklerin ortaya konduğu efor tahmininde, üretilen sonuç değerinin ne derecede esnetileceği, bilgi, birikim ve tecrübeye kalabilmektedir. Neticede üretilen tek bir değer olmaktadır. Bu değerin hangi yönde ne kadar esnetileceği, tahminlerin zorluğuna yeni bir madde daha eklemektedir. Her ne kadar farklı algoritmaların kullanılarak sonuç değerlerinin tahmin edilmeye çalışıldığı çalışmalar var ise de[5,7,8,47,49] tahmin değerini destekleyecek aralıkların verildiği, ya da bu tür bir yaklaşımın önerildiği bir çalışma, tez kapsamındaki yaklaşımın yapısal özelliği göz önünde bulundurulduğunda, görülmemektedir. Oysa efor tahmin değerine ek olarak değer aralıkları da verilirse, proje yöneticisinin üretebileceği senaryolar desteklenebilecektir. Tez kapsamında ortaya konan model için temel çıkış noktası bu yaklaşımdır. Bu noktada ifade edilmelidir ki efor tahmini değeri bir sayısal değerdir ve kullanılan model, veri kümesi ve proje niteliklerine göre bir sayı ortaya çıkacaktır. Değer aralıkları da PMBOK özelinde düşünüldüğünde, bir standart olmaktan öte, proje ihtiyaçları göz önünde bulundurularak hesaplanması gereken bir değerdir[1].

Bu çalışmada amaç, tek bir sayısal efor tahmin değerinin üretilmesi yerine, efor tahmin değeri ile beraber olası en yakın sonuçları, üst ve alt limit türünden vermektir. Buna göre, diğer çalışmalarda da olduğu gibi, YSA ve COCOMO kullanılarak efor tahmin değeri bulunacaktır. Ancak çalışmada farklı olarak, veri madenciliği tekniklerinden K-Means yardımıyla, YSA ile üretilen efor tahmin değeri için en olası üst ve alt değerler hesaplanmaktadır. Bir başka ifadeyle, önerilen yöntem ile tek bir efor değeri yerine, 3 adet tahmin değeri üretilecektir. Bu şekilde proje yöneticisine daha esnek bir yapı sunulacaktır.

Tez çalışmasının ilgili alana katkısı, efor tahminine farklı bir açıdan yaklaşarak, tek bir değer üretmek yerine, karar ve planlama aşamasında daha iyi yönelimlerin olabilmesi için destek değerleri sunmak olacaktır. Bu şekilde tez çalışması ile beraber, efor tahmininde yeterli sayıda sonuç kullanıldığında daha sağlıklı yorumların yapılabileceği yapılar üretilmek hedeflenmiştir.

Tez, 4 bölümden oluşturulmuştur. Genel bilgiler bölümünde tez kapsamında değinilmesi gereken temel bilgiler, bu bilgiler ışığında literatürde yer alan çalışmalar, önceki çalışmaların yaklaşım ve içerikleri ile ilgili bilgi verilmiş, tez kapsamında

ortaya atılan çözüm yaklaşımı detaylandırılmıştır. Aynı zamanda bu çalışma ile literatürde hangi alanda katkı yapıldığı da ifade edilmiştir.

Malzeme ve yöntem bölümünde, ortaya konan model ile ilgili detaylı bilgi verilmiştir. Burada problemin çözümünün nasıl yapıldığı, hangi teknik ve yaklaşımların kullanıldığı detaylı bir şekilde anlatılmıştır. Sistem içerisindeki parametreler tanıtılmış, her bir parametrenin ne anlam ifade ettiği belirtilmiştir. Ek olarak, geliştirme ortamı ile ilgili bilgilerle beraber, çalışmada kullanılan veri kümesi ile ilgili bilgiler de sunulmuştur.

Bulgular ve tartışma bölümünde ise, önerilen metodun mevcut veri kümeleri üzerinde uygulandıktan sonra ortaya çıkan sonuçlar verilmiştir. Öncelikli olarak değerlendirme kriterleri belirtilmiş, bu kriterler ışığında ortaya çıkan sonuçlar değerlendirilmiştir. Bulguların takibinin ve değerlendirmelerinin daha kolay olması için tablo ve grafikler de sunulmuştur.

Sonuç ve öneriler bölümünde bulgulara atıf yapılarak, tez kapsamında beklenti ve bu beklentinin gerçekleşme oranı irdelenmiş, yine sonuçlar ışığında ne ölçüde başarı elde edildiği belirtilmiştir. Öneriler bölümünde de çalışma kapsamına alınmamış ya da olası gelecek çalışmaları kapsamında düşünülebilecek konular belirtilerek, tez sonuçlandırılmıştır.

1. GENEL BİLGİLER

Yazılım mühendisliği, yazılım geliştirme, ürün ve hizmetlerle ilgili süreç ve yöntemleri içeren bir disiplindir[10]. İçerisinde birçok farklı alan ve bu alanların özelinde uzmanlık konuları mevcuttur[2,10]. Tez kapsamında düşünüldüğünde, yazılım mühendisliği disiplini içerisinde yazılım geliştirme süreç yönetimi, proje yönetimi konuları öne çıkmaktadır. Diğer başlıklar bu çalışmanın kapsamında değildir.

Veri madenciliği söz konusu olduğunda da benzer bir durum ortaya çıkmaktadır. Kapsadığı bilgi işleme süreçlerinden ziyade, kullanılan algoritmalardan tez konusuna en uygun olanlar ele alınmaktadır. Çözüm yöntemi olarak belirlenen Yapay Sinir Ağları ve K-Means algoritması, bu çalışma kapsamında değerlendirilebilir.

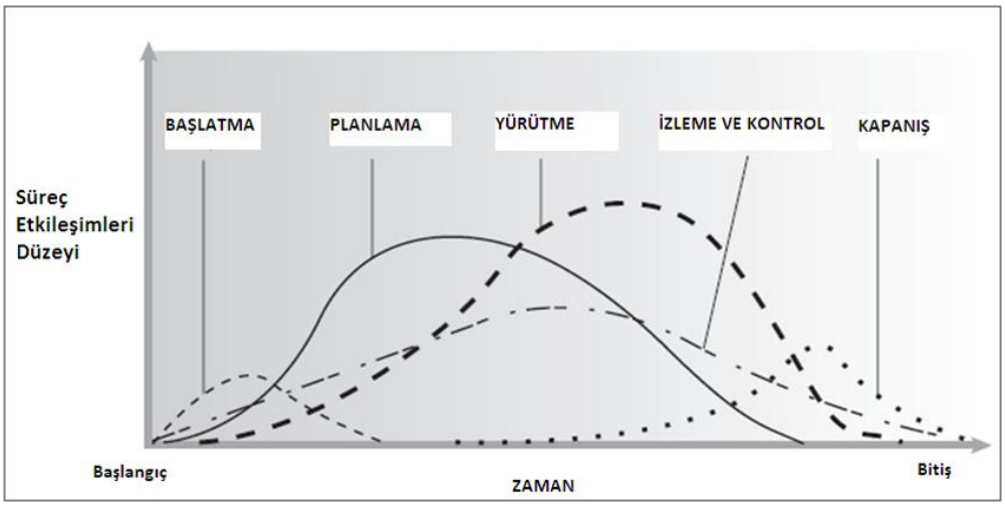
1.1. Yazılım Proje Yönetimi

Genel bir tanım verilecek olursa, proje yönetimi, bir ürün ya da hizmetin geliştirilmesi veya ortaya çıkarılması için gerekecek süreçler bütünü olarak ifade edilebilir. Proje yönetimi içerisinde hem ürüne ya da hizmete yönelik, kapsam ile ilgili süreçler yer alır, hem de proje yaşam döngüsü ile ilgili süreçler yer alır[1,2].

Yazılım proje yönetimi göz önünde bulundurulduğunda da temel çizgi aynıdır. Klasik proje yönetiminden farklı olarak, yazılımın sahip olduğu dört özellik, yazılım proje yönetiminde farklılıkların ortaya çıkmasına sebep olabilir. Bu özellikler, karmaşıklık, uygunluk, değişebilirlik ve görünmezlik olarak sıralanabilir[1,2]. Her bir özellik, yazılım proje yönetiminde farklı aşama ya da süreçlerde farklı çözümlerin ortaya çıkmasına sebep olmaktadır[2,9,10].

İster yazılım projesi olsun, isterse başka bir konu ile ilgili proje olsun, proje yönetiminde temel unsur planlamadır denilebilir. Yapılan planlar ile proje yürütülür ve kontrol edilir. Her ne kadar proje yönetimi içerisinde farklı yaşam döngüsü yaklaşımları mevcut olsa da[11,12,13], planlama, proje için esas unsur olmaya devam eder[1].

Yazılım projesi düşünüldüğünde planlamanın ehemmiyeti daha net bir şekilde ortaya çıkacaktır. Yukarıda ifade edildiği şekliyle, özellikle yazılım projelerinin karmaşık yapıları, soyut bir kavram silsilesi üzerine oturması, planlama aşamalarını da zor bir şekle sokmaktadır. Klasik bir yaşam döngüsü olan Waterfall(Çağlayan) yöntemi düşünüldüğünde, yazılım proje yönetimi içerisinde planlar, sürecin ilk safhalarında gerçekleşecektir[9,11]. Daha sonra bu planlara göre diğer iş bölümleri ve görevlendirmeler yapılacaktır. Dolayısıyla, planlamada kaynakların etkin kullanımı, proje başarısını doğrudan etkileyecektir[2]. PMI (Project Management Institute) tarafından ortaya konulan genel proje süreçleri yaklaşımı düşünüldüğünde de, planlama safhasının proje ya da faz başında temel unsurlardan biri olduğu, Şekil 1.1 üzerinden de görülecektir[1]. PMBOK (Project Management Body of Knowledge), bu yaklaşımın ifade ettiği genel tanımları içeren süreçler bütünüdür.



Şekil 1.1. PMBOK Süreçleri ve Etkileşim Düzeyleri [1]

Akademik çevrede ve sektör içerisinde, planlamanın daha etkin yapılabilmesi, proje süreçlerinin ilgili safhalarında mevcut olan bilgi kümesinin daha etkin kullanılabilmesi için çeşitli yöntemler ortaya atılmıştır[5,6,7,8]. Bu yöntemler, genel olarak, planlamanın daha kolay yapılmasını ya da planların gerçekleşenle uygunluk oranının artırılmasını hedeflemiştir.

Yazılım proje yönetiminde planlamanın etkisi, proje yöneticileri için bir engel oluşturmaktadır. En etkin yöntem ve yaklaşımların belirlenmesi, proje yönetimindeki zorluklara yeni bir madde daha eklemiş olmaktadır. Yapılan geliştirmeler, ortaya atılan yöntemler, temelde proje planlarının daha sağlıklı olabilmesi için proje

yöneticilerinin ya da daha genel ifadeyle proje katılımcılarının ihtiyaç duydukları destek sistemleri şeklini almıştır denilebilir[9].

1.2. Yazılım Efor Tahmini

Yazılım proje yönetiminde, planlama aşamalarında, ilgili proje ya da proje parçası/iş grubu için gerekli olan kaynaklar ve bu kaynakların kullanım durumları zaman ölçütü olarak tahmin edilir. Bu tahminlerden sonra takvim ve diğer planlamalar yapılır. Efor tahmini, en genel ifadesiyle, yazılım projesi için gereken kaynak ve zamanın tahmin edilmesi işlemidir[2,3,9].

Efor tahmini, planlamanın daha gerçekçi ve tutarlı yapılabilmesi için proje yöneticilerine çeşitli teknikler sunan yöntemdir. Daha net bir ifadeyle tanım tekrarlanırsa, efor tahmini, yeni bir yazılım projesi ya da mevcut bir proje üzerindeki çalışmaları, eksik-hatalı ya da kesin olmayan ön bilgi kümesi ile, en gerçekçi şekilde tahmin edilmesidir[3]. Daha önce de ifade edildiği gibi, efor tahminleri proje planına doğrudan girdi olarak dahil olduğundan, gerçekçi tahminlerin ortaya çıkarılması proje başarısı açısından oldukça önem arz etmektedir. Hatalı tahminler proje kaynaklarının verimsiz kullanımına sebebiyet verebileceği gibi, tamamıyla başarısızlığa uğramasına da etki edebilir. Efor tahmini içerisine genel tanım olarak kaynakların tahmini dahil edilmektedir. Kaynak söz konusu olduğunda temelde proje için ihtiyaç duyulacak adam ve süre ifade edilir. Bu iki niteliğin yorumlanması da kullanılan birime bağlı olarak adam-ay, adam-gün gibi formatlar alabilir.

Efor tahmininde eksik, hatalı ya da kesin olmayan bilgi ifadesiyle, proje başlangıcındaki bilgi kümesinin sınırları ifade edilmiştir denilebilir. Başlangıçta, projede, fizibilite ve genel analiz bilgileri yer almaktadır. Bu bilgilerle çoğu zaman gerçekleştirilecek işlevsellikler ve bunların gerçekleştirilmesi için gereken zamanı tahmin etmede yetersiz kalmaktadır. Benzer şekilde birçok proje için istenilen özelliklerde değişiklik talepleri yer alır ve bu talepler de bazen proje ile ilgili temel tahminlerin değişmesine etki eder. Tüm bu sorunlar, efor tahminine daha fazla ehemmiyet verilmesine neden olmuştur. Yazılım proje yönetimi söz konusu olduğunda kaynak ve zaman planlaması, proje yöneticilerinin en öncelikli konularından olmuş ve bu süreçleri başarılı bir şekilde gerçekleştirebilmek için

destekleyici yaklaşım ve yöntemleri aramışlardır. Efor tahmini de, bu beklentileri karşılamak için bir süredir üzerinde çalışma yapılan alanlardan biri olmuştur.

Efor tahmininin geçmişinin 1960 yıllarına kadar dayandığı ifade edilmiştir[3] ancak çalışmalar çoğunlukla 1990 ve sonrasında yoğunlaşmaktadır[5-8]. Her bir çalışma, daha sonrasında kategorik alanlara ayrılmış ve bu şekilde ihtiyaç duyulan çözüm yöntemi için bir üst bilgi verilmesi sağlanmış olmaktadır.

Farklı yaklaşım ve çözümlerin önerildiği efor tahmini alanında, çalışmalar belirli gruplara ayrılmış ve sınıflandırma yapılarak efor tahmini alanı belli ana kategorilere ayrılmıştır. Bu şekilde yaklaşımlar belli bir şablona göre takip edilebilmektedir. Efor tahmini için 3 temel kategoriden bahsedilmektedir[3]:

- Uzman Görüşü: çoğunlukla ilgili proje konusunda tecrübeli, ya da proje süreçlerinde benzer işleri yapmış kişi ve kurumlardan alınan bilgilendirme ve yardım ile yeni proje için ihtiyaç duyulacak kaynak ve zamanın belirlenmesi sağlanır. İş kırılım yapısı[1,14] tabanlı yaklaşımlar örnek olarak verilebilir.
- Formal(Parametrik) Modeller: belirli bir matematiksel, istatistiksel ya da geçmiş verilere dayalı formül tabanlı yöntemlerdir. Özellikle matematiksel gerçekliklere dayandırılıyor olması, uygulama açısından kolaylık sağlamaktadır. Bununla beraber, uygun formül ya da işlemin bulunması konusu bu yöntemin zorlukları arasındadır. Ek olarak, formüllere uygun veri modelinin sağlanması da proje yönetimi süreçlerine ek maliyet getirebilir. COCOMO en bilinen formal yöntemlerden biridir[4,15]. Ek olarak sıklıkla kullanılan SEER-SEM[16], SLIM[17] gibi yöntemler de mevcuttur. Kod ve fonksiyonalite tabanlı, Function Point Analysis(FPA)[18] gibi modeller de bu kategori altında sayılabilir.
- Kombinasyon Modeller: yukarıda listelenen 2 modelin belli oran ve mantıkta birleştirilerek kullanıldığı yöntemlerdir. Proje karakteristiğine göre birden çok kategori altındaki yine birden çok yöntem, ortak bir yaklaşımla birleştirilebilir.

Efor tahmininde mevcut yöntemlerin, belirli senaryolar için uygun çözümler ürettiği ifade edilmiştir. Bununla beraber, bazı yöntemler için sonuçların beklenen seviyede olmadığı da belirtilebilir[11]. Bu yöntemlerin genel kullanım amaçlı olması ve yazılım projelerinin karmaşık ve özgün karakteristikleri dolayısıyla genelleştirilmiş metotlar bazı durumlarda yeterli olmamaktadır. Bu durumlarda, mevcut yöntemlerin

güçlü yanlarından faydalanmakla beraber, efor tahmininin gerçekleştirilebilmesi için destekleyici yaklaşımlar da benimsenmiştir. Matematiksel analiz yöntemleri[19], ya da veri madenciliği tabanlı yöntemler bunlardan sayılabilir[5-8]. Bu tez çalışmasında da benzer şekilde, bir parametrik model olan COCOMO ile beraber veri madenciliği yöntemlerinden faydalanılmıştır. COCOMO ve model içerisinde geçen diğer yaklaşımlar, ilgili başlıklar altında detaylı olarak verilmiştir.

1.3. COCOMO

B.Boehm tarafından ortaya konan COCOMO, Constructive Cost Model ifadesinin kısaltılmış şeklidir. İlk olarak 1981 yılında tanımı yapılan model, parametrik modeller kategorisi altında sayılabilir. Temelde bir hesaplama fonksiyonu olan model içerisinde çeşitli parametreler, bu parametrelerin aralık değerleri ve ortam çarpan değerleri vardır (cost drivers)[4,15].

Model, yapı itibariyle basit, orta ve karmaşık gibi farklı türlere de ayrılmıştır. Her bir tür içerisinde farklı parametreler bulunur ve bu parametreler belli formüllerle hesaplamalara tabi tutulmaktadır. Bu çalışmada intermadiate olarak ifade edilen versiyonu ele alınmıştır. Bu versiyon içerisinde tahminlerin yapılabilmesi için temel bir formül sunulmuştur. Formülün genel yapısı Denklem (1.1) deki gibi;

$$E=a.(KLoC).b.EAF \tag{1.1}$$

bağıntısı ile verilmektedir. Formüldeki a ifadesi, yazılım geliştirme ortamına bağlı olan bir sabit değerdir. COCOMO içerisinde 3 farklı seviye belirlenmiştir. Bunlar, organik, yarı-organik ve gömülü ortamlar olarak ifade edilmiştir. Her bir seviye için formüldeki a parametresi farklı bir değer almaktadır ve bu parametre üretkenlik katsayısı olarak ifade edilmiştir. Benzer şekilde üstel olarak yazılan b değeri de bu sınıflandırma içerisindeki belli bir sabit değerdir ve yine COCOMO seviyelerine göre farklı değerler almaktadır. Bu değer de ölçek parametresi olarak tanımlanmıştır. Değerlerin kategorilere göre karşılıkları için Tablo 1.1' e bakılabilir. EAF değeri ise, COCOMO modelinin temel değerini teşkil etmektedir ve sistem niteliklerinin değerlerinin çarpımıdır. Sistem içerisinde proje faktörleri olarak ifade edilen niteliklerin çarpımıdır.

Proje türüne göre sabit değerler, model oluşturulurken ortaya konulmuş ve testlerle desteklenmiştir. Bu noktada sabitler güncellenen ya da farklı bir dönüşüm geçiren çarpanlar olmamakta, kesin değerler olarak formülde yer almaktadır.

Tablo 1.1. Proje Türüne Göre COCOMO Sabit Değerleri

Proje Türü	**A**	**b**
Organik	3,2	1,05
Yarı organik	3,0	1,12
Gömülü	2,8	1,20

Daha önce de ifade edildiği gibi, sistem içerisinde belli parametreler-nitelik değerleri- vardır. Bu parametrelerin kategorik değerlerinin karşılıkları çarpılarak EAF faktör değerini üretilir. Bu parametreler içerisinde yazılım projesi geliştirme ortamı, proje ekibinin özellikleri gibi çok farklı etmenler, parametre özellik değerleri olarak yer almaktadır. Her bir özellik için belli bir sabit değer vardır. Her ne kadar bu değerler sürekli değerler olsa da, gerçekte belli bir kategorik değerin sayısal karşılığıdır. Parametrelerin değer aralıkları genellikle 0,7 ile 1,50 arasındadır. EAF faktör değeri üretilirken bu parametrelerin sayısal değerleri birbirleri ile çarpılmaktadır. Bu parametrelerin detaylı ifadeleri Tablo 1.2' de verilmiştir. Özelliklerin ortamlara göre etkisi ve değer aralıklarının açıklamaları için COCOMO tanımına başvurulabilir[15]. Ancak burada kısaca ifade etmek gerekirse, tüm nitelik değerlerinin birbiri ile basit çarpım işleminden bahsedildiği anlaşılabilir. Her bir nitelik değeri için temelde kategorik bir aralık vardır ve bu aralıklar çok düşük, düşük, nominal gibi değerlere sahiptir. Ancak hesaplama içerisinde kullanılırken her bir nitelik ve o niteliğe ait kategorik değer için sayısal karşılık verilmiştir ve bu değerler yine tanımın öz niteliğinden kaynaklanmaktadır. Çarpım işlemi de bu niteliklerin sayısal değer çarpımlarından ibarettir. Bir başka ifadeyle Tablo 2 ile verilen 15 özelliğin, ilgili proje için belirlenmiş değerlerinin çarpımı şeklinde bir değerdir.

Niteliklerin ne olduğu konusunda da yine COCOMO tanımına başvurulabilir. Kısaca bahsetmek gerekirse, proje içerisine etki eden/edebilecek ortam, proje, donanım ve

kişisel özellik gruplarından oluşan birtakım ölçüm nitelikleri bu parametreleri oluşturmaktadır. Örneğin TOOL özelliği proje içerisinde kullanılan aracın kabiliyet/karışıklık değerini ifade eder, STOR donanım için saklama özelliğidir. CPLX, oluşturulacak, ürün/hizmetin karmaşıklığını ifade ederken, kişisel özellikler, proje içerisinde yer alanların seviyelerinin belli sınıflandırmalara göre nitelik değerleridir. Bu değerler, tahmin edilebileceği gibi, kimi niteliklerde düşük değerlerde yüksek çarpan, kimisinde ise tam tersi bir etkiye sahiptir.

Tablo 1.2. COCOMO Parametreleri

	Seviyeler					
	Çok Düşük	**Düşük**	**Nominal**	**Yüksek**	**Çok Yüksek**	**Ekstra Yüksek**
Ürün Özellikleri						
RELY	0,75	0,88	1	1,15	1,4	
DATA		0,94	1	1,08	1,16	
CPLX	0,7	0,85	1	1,15	1,3	1,65
Donanım Özellikleri						
TIME			1	1,11	1,3	1,66
STOR			1	1,06	1,21	1,56
VIRT		0,87	1	1,15	1,3	
TUM			1	1,11	1,3	1,66
Kişisel Özellikler						
ACAP	1,46	1,19	1	0,86	0,71	
AEXP	1,29	1,13	1	0,91	0,82	
PCAP	1,42	1,17	1	0,86	0,7	
VEXP	1,46	1,19	1	0,86	0,71	
LEXP	1,29	1,13	1	0,91	0,82	

Tablo 1.2. (Devam) COCOMO Parametreleri

	Seviyeler					
	Çok Düşük	**Düşük**	**Nominal**	**Yüksek**	**Çok Yüksek**	**Ekstra Yüksek**
Proje Özellikleri						
MODP	1,24	1,1	1	0,91	0,82	
TOOL	1,24	1,1	1	0,91	0,83	
SCED	1,23	1,08	1	1,04	1,1	

Formül içerisinde bir diğer önemli çarpan KLoC değeridir. Bu değer, bir yazılım ölçüm metriği olan Lines Of Code değeri üzerinden bir birimdir. KLoC değeri kısaca, yazılım içerisinde geçen/geçecek işletilen satır sayısını ifade etmektedir[20]. K değeri bu ölçümün birimi, Kilo-lines değerini ifade etmektedir. LoC değerinin ölçümü ve detayları bu çalışma kapsamında değildir ancak, efor tahmininde önemli bir yer tuttuğu ifade edilmelidir. Özellikle COCOMO modeli içerisinde temel çarpanlardan biri olduğu düşünüldüğünde, etki ve önemi daha iyi anlaşılacaktır. Aynı zamanda bu etki ve önemden dolayı da, ileri bölümlerde izah edildiği şekliyle, önerilen model içerisinde bir parametre/nitelik olarak yer almıştır.

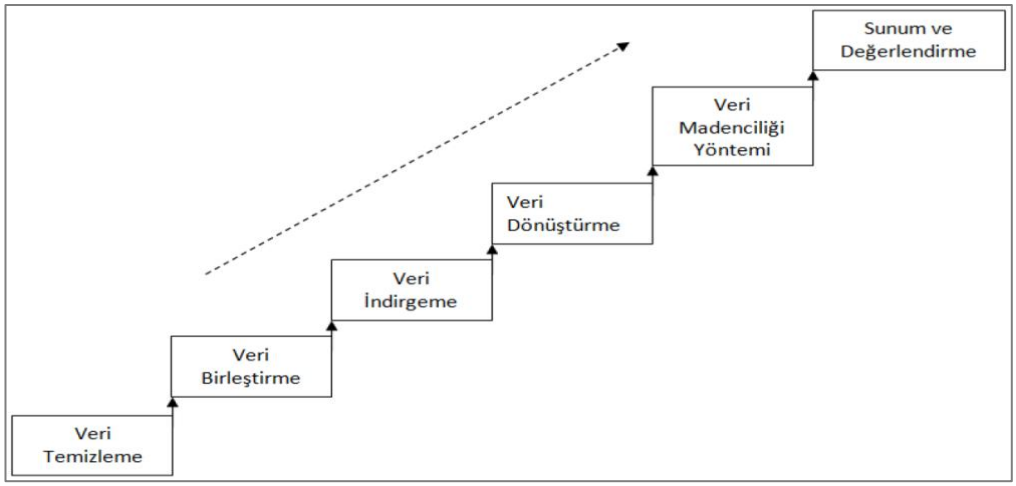
1.4. Veri Madenciliği

Gelişen teknolojik imkânlar ve elektronik ortamların yaygınlaşması ile beraber, yaygın kullanım ağı ortaya çıkmıştır. Teknolojik ve elektronik ortam ürünlerinin/hizmetlerinin yaygın kullanım ağı, beraberinde yüklü veri üretimi de ortaya çıkarmıştır. Öncelerin kâğıt üzerinde, dosya arşivlerinde tutulan yüklü bilgi, artık teknoloji yardımıyla elektronik ortamlarda saklanmaktadır.

Saklanan veriler, arşiv özelliği sunduğu gibi, bu bilgiden analizler yardımıyla yeni bilgilere, çıkarımlara ulaşılmaya da çalışılmıştır. Bu yeni yaklaşım, daha sonra veri madenciliği adıyla da anılacak olan bilgi keşfi işlemi olarak anılan yeni bir yaklaşım ve alanı ortaya çıkarmıştır. Veri madenciliği, tanım itibariyle, mevcut veriden, çeşitli yöntemler yardımıyla yeni bilgiler elde etme işlemidir[21,22]. Özellikle verinin büyük olduğu, ilişki ve istatistiksel birlikteliklerin kolay ortaya çıkmadığı ya da

tamamen ileriye dönük tahmin için mevcut verilerin kullanıldığı senaryolarda çalışma yapmaktadır.

Veri madenciliği, çeşitli aşamaları içermektedir ve bu aşamalarla beraber, kullanım alanı ve ihtiyaca uygun, çeşitli çözüm yaklaşımları, teknik ve algoritmaları vardır. Aşamaların genel gösterimi için Şekil 1.2' ye bakılabilir.



Şekil 1.2. Veri Madenciliği Aşamaları

Şekil 1.2' de görüldüğü gibi, veri madenciliği, içerisinde farklı aşamaları barındıran bir süreçler bütünüdür. Bu süreçler kısaca izah edilirse, veri madenciliğine öncelikle ilgili veri kümesinin çalışma ortamına uydurulmasıyla başlanır. Bunun için veri üzerinde temizleme, tip dönüşümleri, daha küçük parçaların anlamlı bütünlükler için birleştirilmesi ve kullanım için ihtiyaç duyulan bölümlerinin indirgenmesi gibi, veri madenciliği algoritma ya da tekniğinin uygulanması öncesi yapılması gereken aşamalar vardır. Bu aşamalar tamamlandığında, ilgili problem için en uygun çözüm yöntemi belirlenir ve temizlenmiş, bilgi keşfine hazır veri üzerinde bu yöntemler uygulanır. Ortaya çıkan sonuçlar, ihtiyacı karşılayabilecek bir gösterim türüne dönüştürülür ve bilgiye ihtiyaç duyulan alanlara sunulur.

Veri madenciliğindeki algoritmalar ise, çözüm sundukları problem türlerine göre gruplandırılır. Bu gruplandırmada, kümeleme, sınıflandırma, regresyon, tahmin, ilişki analizi ve birliktelik kuralları gibi başlıklar sayılabilir[21,22].

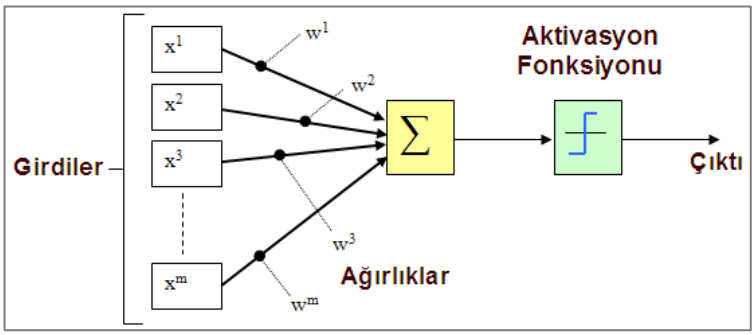
Bu çalışma kapsamında kullanılan yöntemler, Yapay Sinir Ağları (YSA) ile K-Means algoritmasıdır. YSA, kullanım amacına bağlı olarak, sınıflandırma ya da

değer tahmini grubunda sayılabilir. Bu çalışma kapsamındaki kullanım amacı düşünüldüğünde, değer tahmini grubunda olacağı anlaşılabilir. YSA konusunda gerekli bilgi devam eden bölümlerde verilmiştir. K-Means algoritması ise temelde bir kümeleme algoritmasıdır. Yine bu tez kapsamında da mevcut veri kümesini gruplandırma ve bu gruplarla destek bilgisi üretme amacı için kullanılmıştır. K-Means algoritması ile ilgili detay da ilerleyen bölümde verilmiştir.

Veri madenciliği algoritmaları, makine öğrenmesi yaklaşımı kapsamında faklı şekillerde uygulanır[22]. Tez kapsamında tercih edilen ve yaygın kullanılan türlerden biri gözetimli (supervised) öğrenmedir. Bu öğrenme türünde örnek veri kümesi vardır ve öğrenme işlemi veri kümesi ile gerçekleştirilir. Öğrenme sürecinde kontroller yapılabilir ve buna göre modelde değişiklikler gerçekleştirilebilir[22]. Modele karar verildikten sonra veri madenciliğinin gerçekleştirimi yapılabilir. YSA ve K-Means için bu çalışma kapsamında da benzer yaklaşım sergilenmiş, öncelikle örnek veriler uygulanmış ve model bu verilere göre belirlenmiştir.

1.5. Yapay Sinir Ağları

İnsan beyninin bilgiyi işleme mantığından yola çıkılarak ortaya konulan modeldir. İlk modelleri algılayıcı (perceptron) adını alan, beyin sinir hücrelerine benzer bir yapıdadır. Sonraki gelişmeler ve modeller de temelde aynı yapıdan beslenir. Burada girdiler, bu girdileri işleyen bir transfer, maliyet fonksiyonu ve sistem sonucunda üretilen, örnek gösterimi Şekil 1.3 ile gösterilen, çıktı değer(ler)i vardır. Biyolojik modelin matematiksel gösterime çevrilmiş hali olarak da ifade edilebilir[22,23].

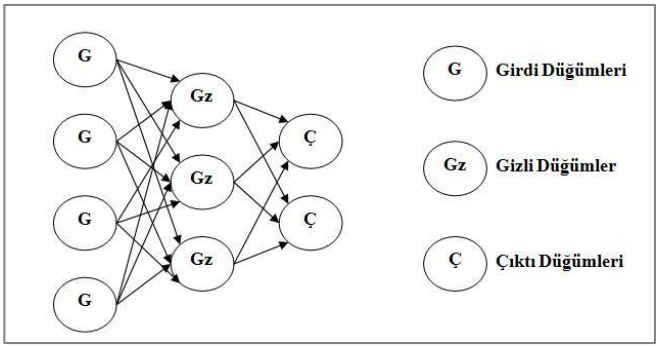


Şekil 1.3. Algılayıcının Genel Yapısı

Şekil 1.3 üzerinde gösterilen şekliyle, x değerleri sistemin girdi değerlerini oluşturur. Algılayıcı dışarıdan bu bilgileri alır. Ağırlıklar, sistemin öğrenme kapasitesini ifade

eden, genelde rastgele verilen değerlerdir. Aktivasyon fonksiyonu ise, ağırlıkların hangi formüle göre dönüştürüleceğini ifade eden matematiksel fonksiyondur. Çıktı değeri, algılayıcının ürettiği sonuçtur.

Veri madenciliği alanında düşünüldüğünde YSA, karmaşık problemlerin çözümünde, tahmin değerlerinin üretilmesinde, sınıflandırmada kullanılabilir. Şekil 1.4' te gösterildiği gibi, en bilinen versiyonlarından biri n katmanlıdır ve bu yapıda, nöron adını alan algılayıcı ve işleyici sistemler, girdi, çıktı ya da ağırlıklandırma yapan bölümler, katmanlar ve bu bölümlerde kullanılan fonksiyon ve algoritmalardan oluşur. Girdi katmanında, veri kümesinin özellikleri ilgili nöronlara, problemin gerektirdiği yapıya göre sunulur. Orta katman, YSA literatüründe ağırlık olarak adlandırılan, öğrenmenin gerçekleştirildiği katsayıları barındırır. Bu katmana gönderilen girdi değerleri, seçilen modele göre işlenir ve çıktı katmanına verilir. Çıktı katmanında da alınan değer yine belli bir fonksiyon ile dönüştürülür ve sonuç değerleri üretilir.



Şekil 1.4. Temel Bir Yapay Sinir Ağı Modeli

Katmanların sayısı, genelde girdi ve çıktı katmanı için tek, orta katman içinse bir veya birden çok olabilir. Girdi katmanında temelde öğrenme ve uygulamada kullanılacak olan veri kümesindeki seçilen özellikler değer olarak verilir. Orta katmanda, ağırlık değerleri, her bir nöron için hata değeri vardır. Aynı zamanda, ağırlıkların dönüşümünün nasıl yapılacağını belirten bir maliyet fonksiyonu vardır. Benzer şekilde çıktı katmanında hata değerleri ve dönüşüm fonksiyonları vardır.

Maliyet fonksiyonlarının seçimi, YSA açısından çok önemlidir. Doğru fonksiyon seçilmediği takdirde ağırlık değerleri, yani öğrenme mekanizması istenilen seviyede olmayabilir[22]. Genelde Sigmoid fonksiyonunun maliyet fonksiyonu olarak

seçildiği ifade edilmekle beraber, Şekil 1.5' teki gibi farklı seçenekler de mevcuttur[24].

Aktivasyon Fonksiyonları	
İsim	Fonksiyon
Birim	$A(x) = x$
Sigmoid	$A(x) = \frac{1}{1+e^{-x}}$
Tanh	$A(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$
Adım	$A(x) = \begin{matrix} -1 & if\ x < 0 \\ 1 & if\ x \geq 0 \end{matrix}$

Şekil 1.5. Bilinen Aktivasyon Fonksiyonları

YSA üzerinde, çalışma mantığına göre farklı algoritmaları kullanabilir. Bu tez çalışmasında kullanılan yaklaşım ileri beslemeli, geri yayılımlı algoritmadır. Bu algoritmada temelde iki aşama vardır. Öncelikle veri kümesiyle ileri besleme, yani ileri yönlü girdilerin işlenmesi sağlanır. Daha sonra, hata oranları hesaplanır ve geriye doğru nöronların ağırlıkları güncellenir. Bu yaklaşımda öğrenme ilgili nöronlardaki ağırlıklara sürekli olarak aktarılmış olur.

Gözetimli öğrenme yaklaşımı kapsamında, eğitim kümesi belirlenir ve ağa sunulur. Ağ öğrenmeye başladığında çeşitli kontrol mekanizmaları ile öğrenme oranı, hata oranı, denetlenir ve belirli bir noktada işlem kesilir. Kontrol mekanizmaları, ağ iterasyon sayısı, öğrenme oranı, hata eşik değeri gibi elemanlar içerir. İterasyon sayısında, ileri beslemeli algoritmanın, mevcut veri kümesi üzerine kaç sefer uygulanacağı belirlenebilir[22,23]. Öğrenme oranı, ağın hangi hızda öğrenmeyi gerçekleştireceğini ifade eder ve genelde 0,2-0,4 arası bir değer alması önerilir[22]. Hata eşik değeri de, iterasyon sayısına benzerdir. Bu değerlendirme ölçütünde bu sefer döngü sayısı değil, her bir döngü sonrasında ağın sahip olduğu toplam hata oranı kontrol edilir ve kabul edilen bir sınırın altına indiğinde işlem tamamlanmış sayılır.

Bu elemanlar dışında bir YSA üzerinde bulunabilecek birçok farklı unsur olabilir; bias nöronu, momentum değeri gibi[22]. Ancak bu tez kapsamında YSA için en

temel seçenek ele alınmıştır ve bu model içerisinde, iterasyon sayısı ve öğrenme oranı dahil edilmiş, herhangi bir bias nöron kullanılmamış, katmanlar arasında da standart ileri beslemeli yönlendirme tercih edilmiştir.

1.6. K-Means Algoritması

Tez kapsamında kullanılan veri madenciliği yöntemlerinden bir diğeri de K-Means olarak ifade edilen algoritmadır. K-Means temelde bir kümeleme algoritmasıdır. Tanım olarak, n elemandan oluşan gözlem kümesini, küme içerikleri ya da merkezlerinin en yakın(uygun) olduğu k adet alt kümeye bölme işlemi şeklinde ifade edilebilir[22,25]. Kümelere bölme işlemi için farklı yaklaşımları, farklı algoritma seçenekleri olmakla beraber, bu çalışmada standart yapı kullanılmıştır.

K-Means için ortaya atılan k sayısı, sistemin başarısı açısından ve beklenen değerlerin alınması açısından en kritik konulardan biridir. Genelde, veri kümesinin sahipliği olduğundan, bir ön aşama ile, uzman görüşü ile, olası en uygun k sayısı belirlenir[22]. K sayısının fazla seçilmiş olması durumunda etkisiz kümeler oluşabilir. Az bir sayıda olması durumda ise, heterojen kümeler ortaya çıkabilir.

K sayısının belirlenmesinden sonra, algoritma, basit olarak şu şekilde çalışır. Öncelikle rastgele küme merkezleri belirlenir ve k adet küme için bu değerler atanır. Sonrasında veri kümesi içerisindeki her bir eleman için, en uygun alt küme bulunur. Bu işlem k adet küme için de yapılır. Uygunluk için genelde, Oklid uzaklığı[26] kullanılmakla beraber, farklı yaklaşımlar da olabilir[27]. En uygun ifadesi, merkeze olan uzaklık olarak ölçüldüğünde, en küçük değer olacaktır. Bu durumda, hesaplama sonucunda en yakın küme içerisine ilgili eleman atanır. Bu işlem, tüm elemanlar için gerçekleştirilir. Elemanları belirlenen kümeler için sonrasında küme merkezi yeniden hesaplanır-bu değer basitçe, küme içerisindeki tüm elemanlar için, tüm özellikler bazında değer ortalamalarıdır. Benzer şekilde, kümeler üzerinde herhangi bir değişiklik olmayana kadar bu işlem tekrar edilir.

Kümeler oluştuktan sonra, eldeki veri kümesine göre, k adet uygun alt küme elde edilmiş olacaktır. Bu kümeler için beklenen, en uygun ortaklık özelliklerini içermesidir. Ancak, algoritma başlangıçta rastgele değerler ile çalıştığı için, her bir çalıştırmada çok farklı kümeler elde edilebilir. Bu nedenle “rastgele” değerle olan

fazla bağımlılığı kaldırmak için bazı çalışmalar yapılmaktadır[28]. Tez kapsamında bu detaylara girilmese de, önerilen metodun test aşamalarında, ortaya çıkan en iyi sonucun dikkate alındığı ifade edilmelidir.

K-Means kümeleri, uygulama ihtiyacına göre, içerik üzerinden faydalı bilgiler verebilir. Tez kapsamında düşündüğümüzde, ilgili proje için en olası efor tahmini değerlerinin bulunduğu söylenilmelidir. Ek olarak, küme içerisindeki istatistiksel değerler (en küçük, büyük efor değeri gibi) kullanılarak bazı yorumlara da ulaşılabilir. Bu değerler, ilerleyen bölümlerde daha detaylı ifade edilecektir.

1.7. Yazılım Mühendisliği ve Efor Tahmini İle İlgili Çalışmalara Genel Bakış

Yazılım mühendisliği alanında akademik olarak çeşitli gruplara ayrılabilecek çalışmalar süregelmiştir. Tez kapsamında düşündüğümüzde, yazılım mühendisliğinde iş geliştirme, proje yönetimi ve bağlı konularla ilgili doğrudan ya da dolaylı olarak çalışmalar yapılmış, veri madenciliği uygulamaları geliştirilmiş, yazılımın sahip olduğu veri kümesi, sadece sonuçları ile değil, süreci ile de incelenir olmuştur.

Menzies çalışmasında, yazılımın kendisinin de veri olduğunu ifade etmiştir[26]. Bir başka ifadeyle, yazılım sadece ürettikleri, sonuç olarak sunduklarıyla değil, yazılım geliştirme süreci, süreç yönetim yazılımları, ilgili kişiler arasında gerçekleşen yazışmalar, dokümanlar gibi süreç içi ürettikleri ile de değerlendirilmelidir. Tabii ki kodlar, test senaryoları, test sonuçları, hata veri tabanları gibi birçok farklı alanda da çok miktarda veri üretildiğini, bu verilerin hem sürecin iyileştirilmesi, hem de analizlerin yapılabilmesi için kullanılabileceğini bildirmiştir. Aynı çalışma üzerinde vurgulanan önemli bir diğer nokta, yukarıda sayılan alanlardaki verilerin hangi yöntem ve araçlarla inceleneceği, kullanıma nasıl hazır hale getirileceği ile ilgili çalışmaların elzem olduğu gerçeğidir.

Son tespitten yola çıkılarak vurgulanacak ilk nokta, yazılım projelerindeki en temel kaynak ile ilgili çalışmalardır: yazılan kodların incelenmesi. Söz konusu çalışmalar bazı yayın-alan grupları altında toplanmıştır. Bunlardan önemli bir tanesi Mining Software Repositories(MSR) adını alan, temelde kod ve kod unsurlarını irdeleyen veri tabanlarından faydalanılarak, süreç iyileştirme çalışmaları yapmaktadır[30].

MSR, hem bir yayın grubu olmuş, hem de konferans organizasyonları ile bulgular paylaşılmıştır. Bu alandaki en temel çalışmalardan biri, Ahmed Hassan tarafından yapılan doktora çalışmasıdır[31]. Bu çalışmada bir bakıma MSR konusunun konsepti ortaya konulmuştur. Çalışma, kısaca, yazılım unsurlarının aktif olarak kullanım ve incelenmesiyle hem geliştiricilere, hem de yöneticilere süreci iyileştirici adımlar önerme, tahmin çalışmalarına destek olma, yazılan kodlardaki hata kaynakları ve oranlarını irdeleme ile kaliteyi artırma gibi temel başlıklarda, faydalanabilecek veri üretmeden bahsetmiştir. Yine ilgili alanda önemli bir çalışma, MSR için bir ortak dil geliştirmeyi amaçlayan, TA-RE olarak kısaltılan çalışmadır[32]. 2006 yılında yayınlanan çalışmada, özellikle yazılım kaynaklarının analiz edilmesi ve işlenmesinde genelleşmiş bir yapının olmaması, ortaya çıkarılan temizlenmiş verinin yapısal bir altyapısı olmaması ve paylaşımın uygulama özelinde gerçekleşmesi gibi konular üzerinde durulmuş, önerilen dil ile bu konularda ortak bir yapı oluşturulması hedeflenmiştir. Standart bir mesaj yapısı belirlenmiş, kullanım yerine göre, bu mesajların iletilmesi üzerine bina edilen bir mimari sunulmuştur. Özellikle yazılımların, ilgili kurum ve problem özelinde farklılık gösteren veri kümelerindeki analizlerin irdelenmesi ve karşılıklı kullanımı için önemli bir gelişme olduğu ifade edilebilir.

Bir diğer grup, Search Based Software Engineering adını alan disiplindir. Burada da eldeki yazılım kaynaklarının belli analiz modeller ile incelenmiş ve süreç iyileştirme önerileri sunulması hedeflenmiştir[33]. İlgili referanstaki çalışmada[34], bir yazılım bakım projesinde efor tahmininin genetik algoritma ile geliştirme konusu ele alınmıştır. Konunun tez ile olan bağlantısı düşünüldüğünde, farklı bir yaklaşım ile efor tahmini yapıldığı, genetik algoritma[35] ile mevcut modellerin iyileştirilmesi hedeflendiği ve bulgularla da beklentinin gerçekleştirildiği belirtilmiştir.

Predicting Models in Software Engineering (PROMISE)[36] adını alan çalışma alanında da önemli gelişmeler mevcuttur. Minku ve Yao, çalışmalarında, toplu makine öğrenme modelleri ile efor tahmini yapılması konusunu incelemişlerdir[37]. Yayınladıkları çalışmada, farklı makine öğrenmesi yaklaşımlarını, belli parametrelere göre tasarlamış, birleştirmiş ve örnek veri kümeleri üzerinde sonuçları karşılaştırmışlardır. Yapılan seçimlerde, tercih edilen yöntemin etkisi üzerine vurgu yapılmış, doğrudan doğruya bir model önerisi, en iyi model tercihi, sunulmamış

ancak, bulgulara göre hangi yaklaşımların efor tahmininde daha etkili olabileceğine ilişkin görüş belirtilmiştir. Yine aynı araştırmacıların yayınladığı bir diğer çalışmada[38], kurumlar arası veri kümelerinin kullanımı ile efor tahmininde başarı oranının artırılıp artırılamayacağı üzerinde durulmuştur. Başlangıç önermesi olarak, mevcut bulguların, kurum için veri kümelerinin kullanımının daha iyi olduğu belirtilmekle beraber, önerilen yaklaşımla, kurumlar arası veri kümelerinin performansı artırıcı etkisi olabileceği bulgu olarak ifade edilmiştir. Burada önemli nokta, seçilen kümelerin hangi özellikleri içerdiği, hangi projeler ve özellikle hangi zaman aralıkları için incelendiği ile ilgili denemeler yapılmıştır. Yakın süreler ve küçük süre gruplarında daha etkin sonuçların alındığı ifade edilmiştir.

Farklı YSA modellerinin kullanıldığı ve bu modellerin karşılaştırmalarının yapıldığı [48], yazılım geliştirme süreci içerisinde iş paketlerine göre efor tahminin değerlendirildiği[50], ileri beslemeli ağ ile efor tahminin yapıldığı[51], yine YSA tabanlı, farklı veri kümeleri ile yapılan[54] çalışmalar da mevcuttur. Benzer şekilde, tek bir model kullanmak yerine farklı bütünleşik yapıların önerildiği[52], makine öğrenmesi tabanlı yaklaşımların önerildiği[53] çalışmalar da görülmektedir. Layman, Visual Studio ürününün ön geliştirme süreci üzerinde efor tahmini yapmış ve önemli bir veri kümesi ile çalışmıştır[55]. Efor tahmininde karşılaşılan problemlerden biri uygun verilerin eksikliğidir ve bu konu ile ilgili de çalışmalar görülmektedir[56].

1.8. Yazılım Efor Tahmininde COCOMO Modelini Baz Alan Çalışmalar

Yazılım mühendisliği genelinde, proje yönetimi, efor tahmini ile ilgili dolaylı çalışmalara ek olarak, bu tez kapsamında baz aldığımız, yukarıda açıklanan COCOMO modeline uygun efor tahmini yapılan çalışmalar da mevcuttur. Bu çalışmalar, modelin farklı parametrelerle kullanımı[6,7], model parametrelerinin özelleştirilmesi[5], modelin uygulandığı algoritmaların karşılaştırılması[39] gibi farklı konular üzerinde durmuştur. Bu bölümde, tez yazımına katkıda bulunan, önerilen metodun çıkış noktaları olarak görülebilecek, önceki literatür çalışmaları irdelenmiştir.

İlk olarak ifade edilmesi gereken noktalardan biri, modelin genelde YSA üzerinde gerçekleştirildiğidir. Farklı algoritma ve yaklaşımların olduğu görülse de, model parametrelerinin tahmin değerleri ile ilişkisi ve hesaplama formülünün yapısı,

genelde YSA üzerinde yoğunlaşılmasına sebebiyet vermiştir denilebilir. YSA' nın tahmin değerleri üretebilmesi de yine bu modelin tercih unsurlarındandır.

Efor tahmini ile ilgili Jorgensen' in makalesinde, daha önce yapılan çalışmalar bir gözden geçirme incelemesi olarak sunmuştur[40]. Efor tahmini ve tahmin modelleri ile ilgili bu çalışma iyi bir literatür referansı olabilir. Bu makalede, yayınlanan çalışmalar kısaca ifade edilmiş, her bir çalışmanın araştırma yöntemi, öneri modeli ve sonuçları kıyaslanmıştır. Bu suretle, yayın tarihine kadar geçen süre içerisinde yapılan çalışmalar, kapsamları ve tercih edilen tekniklerdeki yönelim konusunda genel bilgiler edinilebilir.

Çalışmalarda genelde PROMISE sitesinde yayınlanan açık veri kümeleri kullanılmıştır[41]. Burada farklı modellere ait veriler bulunmakla beraber, daha çok, tez çalışmasının kapsamını oluşturan COCOMO modelini baz alan veri kümeleri ile yapılan çalışmalar dikkate alınmıştır. Bu veri kümeleri, ilgili bölümde anlatıldığı şekliyle, COCOMO modeline ait parametrelere uygun geçmiş değerler sunmaktadır. Yani, modele uygun gözlemleri içerir.

YSA ve efor tahmini ile ilgili olarak, Kaushik, COCOMO modelinin efor tahmin başarısını artırmak amacıyla, bu modeli YSA üzerinde uygulamıştır[6]. Önerilen model 3 katmandan oluşturulmuş, girdi katmanına 15 efor çarpanı, LOC değeri ve bias değeri sunulmuştur. Gizli katman 1 tanedir ve içerisinde 5 nöron bulunmaktadır. Çıktı katmanında da efor tahmin değerini veren tek nöron vardır. Bu modelde aynı zamanda, COCOMO formülü, YSA geri yayılım algoritmasının çarpımların toplamı şeklinde çalışmasını baz alarak, logaritmik forma dönüştürülmüş ve ağırlık hesaplamaları buna göre yapılmıştır. Yayında, COCOMO'63 veri kümesi kullanılmıştır[41]. Değerlendirme ölçütü olarak da Magnitute of Relative Error (MRE) kullanılmıştır[5,42]. Deneysel gerçekleştirimin sonuçları tablo olarak verilmiş, bu tabloda, gerçekleşen efor değeri ile önerilen modelin tahmin değeri karşılaştırılmıştır. MRE değerinin düşük sayılabileceği, sistemin dengeli olduğu ve gerçekleşen efor değerlerine yakın olduğu ifade edilmiştir. Özellikle efor tahmin değerlerinin gerçekleşen değerlerin büyüklük-küçüklüklerine göre eğilimlerinin uygun olduğu grafik olarak da verilmiştir.

Reddy, daha farklı bir YSA modeli kullanarak efor tahmini yapmıştır[5]. Temel varsayım, COCOMO fonksiyonunun lineer bir hale çevrilerek, daha uygun kullanımıdır. Burada da fonksiyonun logaritmik formu alınmış, çarpanlar kendi grupları içerisinde toplanmıştır. Ortaya çıkan yeni formda görülen, formülün temelde 2 parçası olduğudur: efor çarpanı ve büyüklük çarpanı değer grupları. YSA, bu modele göre kurgulanmış, standart, her bir girdi nöronun gizli nörona sunulması yaklaşımı yerine, sadece ilgili girdiler, ilgili gizli katman nöronuna sunulmuştur. Bu ifadeden de anlaşılacağı gibi, girdi katmanında model parametreleri ve 2 adet bias nöronu, tek gizli katmanda 2 nöron vardır. Yine çıktı nöronu bir tanedir. Özellikle modelin kendi çarpanlarını ilgili nöronlara sunuyor olmasının daha etkin bir yöntem olduğu ifade edilmiştir. Bu çalışmada da COCOMO'63 veri kümesi kullanılmıştır ve kümenin 80-20 oranlarında eğitim ve geçerleme alt kümelerine bölündüğü ifade edilmiştir. Yine değerlendirme ölçütü olarak MRE kullanılmıştır. Sonuçlar, efor tahmin değerleri olarak değil, tahminlerin MRE cinsinden verilmiştir. Karşılaştırma için de, COCOMO ile efor tahmin değerlerinde ortaya çıkan MRE değerleri ve önerilen metodun ortaya koyduğu değerlerde ortaya çıkan MRE değerleri sunulmuştur. Önerilen metodun daha başarılı ve dengeli olduğu ifade edilmiştir.

Kültür, çalışmasında tek model kullanımı yerine, birleşik bir yapı tercih etmiştir[8]. Bu çalışmada veri kümeleri 20 adet alt kümeye ayrılmıştır. Aynı zamanda 20 adet YSA hazırlanmış, bu veri kümeleri ilgili model unsurlarında çalıştırılmıştır. Sonuç değerleri için ağırlıklı ortalama değerleri alınmış, bu noktada kullanılan algoritma ile en uygun sonuç değeri hesaplanmıştır. Aynı zamanda geçmiş proje bilgilerinden hata değeri aşamasında da faydalanmış, K-Nearest Neighbourhood(KNN) yöntemi[22] ile gerçekleşen geçmiş efor değerlerine en uygun sonuç değeri seçilmiştir. Bu değer de daha sonra YSA' lar tarafından üretilen sonuç değerine bias değeri olarak eklenmiştir. Bu şekilde öncelikle dengelenmiş, farklı ve birden çok YSA ile çalışan bir yapı sunulmuştur. Özellikle veri kümelerinin nispeten küçük (60-90 satır) olmasından dolayı oluşabilecek problemler minimize edilmiş, ortalama alarak ve geçmiş proje değerlerinden hata oranı hesaplayarak bulunan sonuç değeri daha gerçekçi bir seviyeye çekilmiştir. Sonuçlar yine MRE olarak ifade edilmiştir. Bu çalışmada COCOMO veri kümeleri ve SDR adındaki veri kümesi kullanılmış[43],

model farklı yaklaşımlarla karşılaştırılmıştır. Sonuçta ortaya konan modelin dengeli ve etkin olduğu yorumuna varılmıştır.

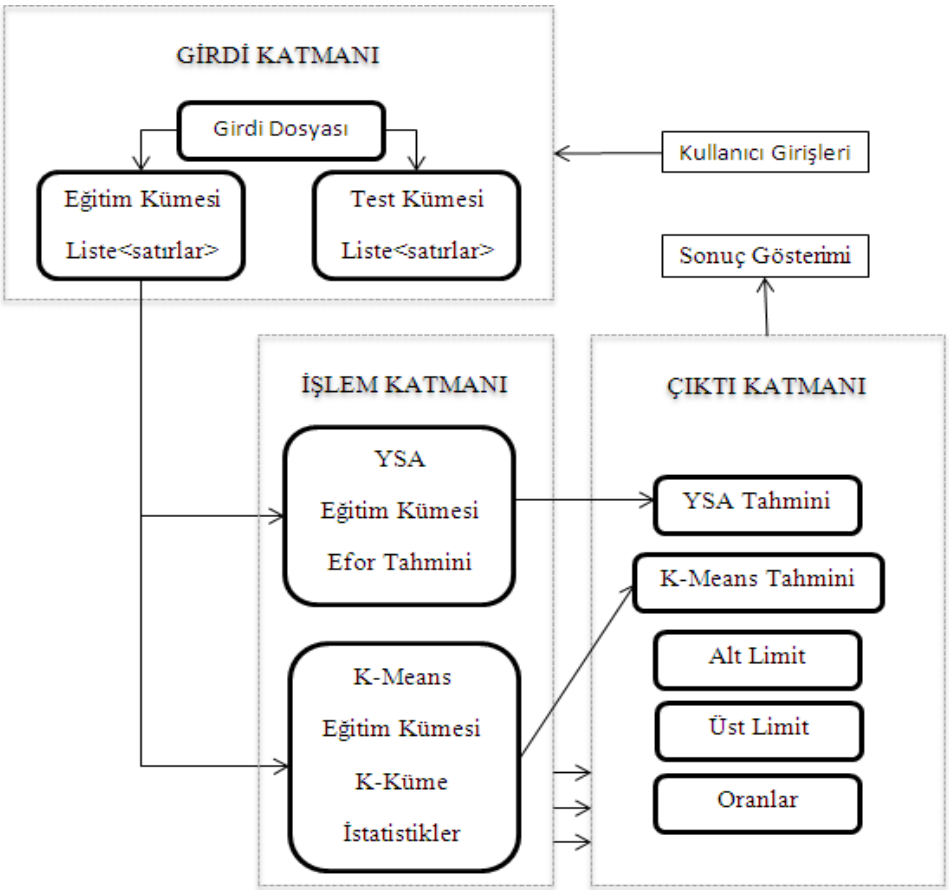
Bu bölümde listelenen çalışmalar, tez kapsamında temel unsurlardan olan COCOMO modeli üzerine kurgulandığı için tercih edilmiştir. Bu çalışmalar aynı zamanda tez çalışması için de çıkış noktası olmuş, dolayısıyla çalışmaya ve modele doğrudan katkı sağlamıştır. Ancak, ilgili çalışmaların haricinde, dolaylı olarak model geliştirmeye katkı sağlayan çalışmalar da mevcuttur. Bu çalışmalar da tez sürecinde belli ölçülerde irdelenmiş ve bu çalışmalardan da faydalanılmıştır[57-60]. Tez ve konu bütünlüğü için bu noktada birkaç örnek referans verilmiş olsa da, sayı ve hacmin daha fazla olduğu ifade edilmelidir.

Tez çalışmasının en yakın olduğu literatür Kültür ve arkadaşlarının yaptığı çalışmadır[8]. İlgili çalışmada veri kümelerinden rastgele alt kümeler üretilmiş, hata değeri, bir bakıma esneme payı için K-en yakın komşu algoritması kullanılmıştır. Bu çalışmada ise hata değeri (bias) kullanılmamış, bu değer de limit değerlerinin sağlayacağı düşünülen esnekliğe yüklenmiştir. Çalışmanın temel farkı bu limit değerlerinin hesaplanması ve bu işlemi de farklı ve yetkin bir modelle yapmış olmasıdır.

2. MALZEME VE YÖNTEM

Yazılım efor tahmininde destek değerlerinin oluşması, olası üst ve alt limitlerin belirlenebilmesi için mevcut yaklaşımların esnetilmesi gerekmektedir. Daha önce de ifade edildiği gibi, efor tahmini, genelde tek çıktısı olan bir süreçtir. Oysa karar destek aşamasında daha fazla değer ile sonucun çeşitlendirilmesi yazılım projelerinde yönetimi kolaylaştıracaktır.

Bu çalışmada temel kurgu YSA kullanımı üzerinedir. YSA, sistem için tahmin değerini üretmektedir. Bununla birlikte, önerinin temelini oluşturan esnekliği sağlayacak yapı K-Means üzerine inşa edilmiştir. YSA çıktıları, K-Means kümeleri ile irdelenir ve olası limit sonuçları da tahmin değerlerine eklenir.



Şekil 2.1. Geliştirilen Sistemin Genel Yapısı

Sistemin genel yapısı Şekil 2.1' deki gibidir. Sistem içinde bir girdi katmanı, bir işlem katmanı ve çıktı katmanı vardır. Girdi katmanında, önceki proje değerlerini COCOMO modeline uygun şekilde veren bir dosya ve bu dosya içeriğinin sistem içerisinde kullanılabilir hale getirilmiş, normalleştirilmiş satırları mevcuttur. Bu satırlar eğitim ve test kümelerini oluşturur. Satırlardaki değerler, nitelikler, sistem içerisine ayrı ayrı girdiler olarak sunulur. Yapı içerisinde birçok farklı aşama ve nitelik değerleri bulunmaktadır ancak gösterim kolaylığı için daha sade bir şekil tercih edilmiştir. Şekildeki oklar girdi ya da veri akış yönünü ifade etmektedir. İşlem katmanı ile çıktı katmanı arasındaki katman seviyesindeki çizgiler, üretilen diğer sonuç değerlerini ifade etmektedir ve bu değerler ileri bölümlerde izah edilmektedir. Ayrıca ortaya konan model bir araç üzerinde de geliştirilmiş ve dolayısıyla kullanıcı ön yüzü de olmuştur. Bu nedenle kullanıcı etkileşimini ifade eden simgesel bir giriş ve kullanıcıya sonuçların gösterildiği ara yüzü ifade eden yine simgesel bir çıkış noktası da belirtilmiştir.

İşlem katmanı, gerçekte tahminin yapıldığı yerdir. İçerisinde temelde YSA ve K-Means kümelerini barındırır. Ayrıca, girdi katmanı ile dosya nitelik değerleri üzerinden haberleşmektedir. İşlem katmanı içerisindeki 2 yapı birbiri ile ilişkilidir. Yukarıda ifade edildiği gibi, sistem YSA üzerine kurgulanmış, ek olarak sınır değerleri YSA çıktısının K-Means kümelerine sunulması ile üretilmiştir. 2 yapının birleşmesinden ortaya faklı sayıda sonuç değeri çıkmaktadır. Bu değerler de sistemin çıktı katmanını oluşturur. Çıktı katmanındaki bu değerler aynı zamanda sistemin anlamlı yüzüdür. Buradaki değerler hedeflenen bilgiyi ifade eder ve uygun şekillerle gösterilir.

Tüm sistem genelinde veri akışı tek yönlüdür. Bir başka ifadeyle girdi katmanı sadece tek yönlü olarak işlem katmanını besler, işlem katmanı gerekli hesaplama ve çalışmalarını gerçekleştirip ürettiği sonuçları çıktı katmanına aktarır. Çıktı katmanına gönderilen bilgiler sunulmak üzere kullanıcıya gösterilir. Bu aşamalarda temel unsurlardan biri, küme elemanları, küme satır değerleridir. Girdi katmanının asil öğeleri olan bu değerler, özellikle işlem katmanında da kopyalanır ve taşınır. Çıktı katmanında ise mevcut değer, beklenen değer sunumunda kullanılmak üzere referanslıdır. Bunun haricinde katmanlar bir öncekinin sonuç ya da özellik olarak

ortaya koyduğu üzerinden çalışır ve bu nedenle genel olarak tek yönlü bir akış olduğu ifade edilmiştir.

Her bir katman, katmanların içerisindeki bileşenler, bileşenlerin özellikleri, parametreleri ve sistem parametreleri ile kısıtları, takip eden bölümlerde detaylı olarak incelenmektedir.

2.1. Girdi Dosyası

Önerilen sistem COCOMO parametrelerine uygun bir yapıdadır. Bir başka ifadeyle sistem içerisindeki değerler COCOMO nitelikleridir. Bu değerler de bir dosya üzerinden okunmaktadır. İlgili dosyalar, açık olarak sunulan ve PROMISE sitesinde bulunabilen [41] dosyalardır. Bu dosyalarda her bir satırda bir proje bilgisi bulunur. Bu bilgi içerisinde COCOMO değerleri ile LOC ve gerçekleşen değer bilgileri vardır. Sistem içerisinde dosyanın kullanımının kolaylaştırılması için standart virgülle ayrılmış dosya formatı (csv) kabul edilmiştir. Bu formatta her satırda bir proje bilgisi, tekrarlanmayacak şekilde yer alır. Her bir sütun ayıraç bilgisine göre ayrılır. Dosyalarda eksik verili satırlar mevcut değildir.

Bu çalışmada bahsi geçen dosyalardan 3 tanesi kullanılmaktadır. Bu dosyalar ile ilgili içerik bazlı diğer bilgiler, bulgular ve tartışma bölümünde verilmiştir. Veri kümelerinin orijinal formatı ise, COCOMO yapısına göre kategorik değerlerin ve bu değerlerin elde edildiği proje ile ilgili bilgiler, veri kümesi ile ilgili genel bilgiler ve kategorik değerlerin sayısal karşılıklarının verildiği açıklamalarını içeren metin tabanlı dosyalardır. Dosya içerisinde önemli olan kısım, proje satırlarıdır. Diğer kısımlar açıklama ve özellikleri içerir. Veri kümesinin orjinali üzerinden bu proje satırları alınmaktadır. Proje satırlarının nasıl ifade edildiğinin küçük bir örneği ekler bölümünde EK-B içerisinde verilmiştir. Ek içerisinde sadece örnek bir bölüm vardır; dosya içeriği için yine ilgili adrese başvurulabilir[41].

Çalışmada ortaya konan model ile ilgili etkileşim bilgilerinde de ifade edileceği gibi, veri kümeleri üzerinde normalizasyon yapılmıştır. Burada veri madenciliği yaklaşımı ile ifade edilirse aslında bir ön işlem süreci de gerçekleşmektedir. Mevcut küme metin tabanlı ayrımları ifade eder. Her bir satır, bir proje değerini ifade eder ve sırasıyla her bir sütun bir niteliği anlamlandırır ve bunların içinde LOC ve

gerçekleşen değer sütunları da vardır. Öncelikle bu metin tabanlı değerler, model içerisinde verilen standartlara göre sayısal karşılıklarına çevrilir. Sonrasında ise bu değerler için hassas hesaplama yapabilmek adına normalizasyon yapılır. Kısacası veri kümeleri ön işlemden geçirilir ve ilgili satırlar çekilir ve sonra dönüştürülerek kullanılabilir csv dosyaları haline getirilir.

2.2. Eğitim ve Test Kümeleri

Sistem içerisinde girdi katmanını oluşturan temel eleman COCOMO tabanlı verileri barındıran bir dosyadır. Bu dosya içerisinde COCOMO modelinde var olan nitelikleri içeren, ek olarak bir proje numarası, bir de gerçekleşen efor değerini barındıran satırlar vardır. Dosya sistem içerisinde dinamik olarak okunabilmesi, esneklik sağlama ve dışarıdan veri alma imkânını da destekleme düşüncesi ile yukarıda da ifade edildiği gibi virgülle ayrılmış dosya formatında (csv) belirlenmiştir. Buradaki her bir sütunda bahsedildiği gibi COCOMO nitelikleri ve LOC, gerçekleşen değer gibi diğer değerler vardır. Bu değerler sistem içine (işlem sürecinde) girdi olarak sunulurlar. Yani kabul edilen dosya formatı, eğitim ve test kümeleri için bir kaynak oluşturur ve daha sonra da bu kümeler sistem girdileri olarak kullanılır.

Dosyanın içeriği her bir satır bazında, önceki proje değerlerini ihtiva eder. Yani her bir satır, sistem içerisinde kullanılabilecek geçmiş örnek proje verisidir. Bu değerler veri kümesini oluşturan, gerçek birtakım projelerden elde edilen verilerdir. Örnek veri kümeleri için bu kaynak temelde NASA projeleridir. Eğitim ve test kümeleri de bu dosyadaki satırlardan oluşmaktadır. Bu durumda, girdi katmanının çıktıları olarak 2 kümeden bahsedilebilir. Eğitim kümesi YSA ve K-Means kümelerinin oluşturulmasında, bir başka ifadeyle işlem katmanında kullanılmaktadır. Test kümesi de sistemin geçerliliğinin denenmesinde örnek olarak kullanılmaktadır. Bu kümelerin oluşturulması 2 temel yaklaşıma göre, kullanıcı tarafından belirlenebilecek şekilde gerçekleştirilir. Kullanıcının belirleyebileceği yaklaşım, kümelerin oluşturulması yöntemidir. Burada herhangi bir kümeye dâhil edilen satırın tekrar dosya genelinde tutulup tutulmayacağı ya da bir başka ifadeyle aynı satırın sistem içerisinde tekrar kullanım imkânının olup olmayacağıdır. Genelde bu yöntem, örneklem oluşturmada, Sampling With Replacement (Yerine Koyarak Örnekleme, SWR) olarak

geçmektedir. Kavram bütünlüğü açısından da kullanıcıya bu şekilde sunulmuş, yapılacak seçime göre SWR yöntemi kullanılıp kullanılmayacağı bilgisi alınmıştır. Eğer bu yöntemle kümeler oluşturulacaksa, rastgele küme oluşturulma işleminde herhangi bir satır bir kümeye dâhil olduğunda o satır tekrar okunan dosyaya eklenir. Eğer bu yöntem seçilmemişse herhangi bir kümeye, yani eğitim ya da test kümesinden birine eklenmiş bir satır bir daha sistem içerisinde yer alamayacaktır.

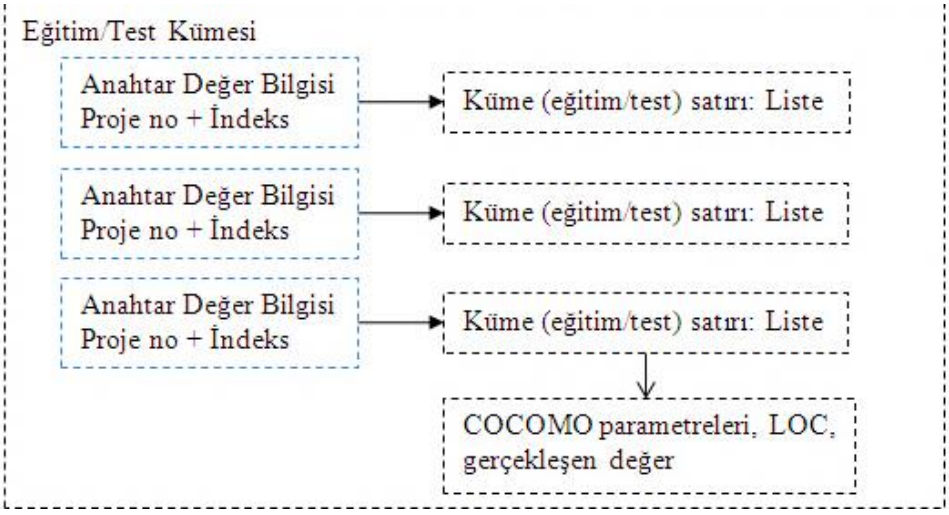
Eğitim ve test kümelerinin oluşturulmasında "rastgelelik" (randomness) esastır. Okunan dosyadaki satırlar sıra ile değerlendirilmez. Bunun yerine rastgele üretilen bir indeks değeri referans olarak kullanılır ve bu sıradaki satır ilgili kümeye eklenir. Ayrıca ilgili kümeye eklenen satırlar için sonrasında bir düzenleme işlemi, bir sıralama işlemi de yapılmaz. Rastgele seçilen satırlar yine ilgili kümede, kendi yapısı içerisinde rastgele tutulmaktadır. Bu şekilde sistemin rastgele olma durumuna sadık kalınmaktadır.

Kümeler oluşturulurken üretilen rastgele indeks değerlerinin takibi ve seçilen yönteme göre, eklenmiş bir satırı tekrar kümelere dâhil etmemek için bir liste yönetimi yapılır. Burada tüm satırlar için önce bir referans listesi oluşturulur. Daha sonra kümelerin oluşturulması aşamasında bu listeden faydalanılır. Rastgelelik de doğrudan girdi dosyası satırı yerine bu referans listesi ile yapılır. Yani rastgele üretilen indeks değeri bu listeyi kontrol eder. Bu listedeki satır numarası ile de asıl dosyaya gidilir. Buradaki rastgelelikte, sınırları belirlemede liste içerisindeki eleman sayısı esastır. Yani rastgele üretilecek değerin üst limiti eleman sayısıdır ve üretilen rastgele değer ile her zaman listede mevcut bir satıra, mevcut bir elemana ulaşma imkânı vardır.

Eğer SWR yöntemi kullanılmıyorsa, herhangi bir kümeye eklenen satır için ilgili referans bu listeden çıkarılır. Referans listesinde artık ilgili satır bulunmadığından, aynı satırın eklenmesi durumu söz konusu olmayacaktır. Aksi durumda listede bir değişiklik yapılmaz ve herhangi bir çıkarma işlemi olmadığından rasgtelelik aşamasında aynı satır tekrar bulunabilir.

Girdi katmanında 2 temel çıktı olduğu ve bu çıktıların işlem katmanında kullanıldığı ifade edilmiştir. Bunlar eğitim ve test kümelerinin tutulduğu yapılardır. Burada vurgulanması gereken nokta, model olarak sunulan yapılarda küme içeriklerinin

okunabilmesi ve takip edilebilmesi için, satırların uygun veri yapısına göre tutulduğudur. Burada tercih edilen yapı bir sözlük (dictionary) yapısıdır ve bu yapıda anahtar değer ile içerik değeri bilgileri vardır[44]. Eğitim ve test kümelerinde her bir satır için anahtar değer, o satırı ifade eden satır no ya da bir başka ifadeyle proje no bilgisidir. İçerik değeri de o satırdaki diğer tüm bilgilerdir ki bu bilgilerin de bir liste olarak tutulduğu anlaşılabilir. Şekil 2.2 eğitim ve test kümelerinin sistem içerisindeki yapısal görünümünü ifade etmektedir. Bu aynı zamanda girdi katmanının sistem açısından durumunun detayını ifade etmektedir.



Şekil 2.2. Eğitim ve Test Küme Veri Yapısı

Eğitim ve test kümeleri ile ilgili üzerinde durulması gereken bir diğer konu, buradaki orijinal değerlerin sistem içerisinde kullanılabilmesi için normalizasyon işleminden geçirilmiş olmasıdır. Veri madenciliği aşamalarından biri olan veri ön işleme aşaması sistem içerisinde bu noktada gerçekleşmektedir. Girdi dosyası normalizasyona tabi tutulmuş, oluşturulan eğitim ve test kümelerindeki tüm veriler için sistem genelinde standart değer ve değer aralıkları kullanılmıştır.

Normalizasyona tabi tutulan değerler en başta gerçekleşen efor değeri bilgisidir. Bu değer her bir proje bazında gerçekleştirim için ne kadar adam/ay geçtiğini ifade eder. Bu bilgide genelde varyasyon geniştir ve değer aralıklarına da bağlı olarak, 3 haneli rakamlara ulaşma durumu da söz konusudur(400 adam/gün gibi). Normalizasyon ile buradaki değerler 0 ile 1 aralığına çekilmiştir. Özelikle YSA için değerlerin dramatik etkisi göz önünde bulundurulduğunda normalizasyon daha da önem arz etmektedir. Bir diğer önemli normalizasyon sütunu LOC değeridir. Bu da bir bakıma sistem

büyüklüğünü ifade eder ve burada da 2 haneli yüksek rakamlar olabilmektedir (95 gibi). Bu değerler de benzer şekilde 0-1 aralığına çekilmiştir. Diğer değerler COCOMO standart değerleridir ki bunlar daha önce de ifade edildiği gibi genelde 0,70 ile 1,60 arasındadır. Buradaki normalizasyonda daha basit bir yaklaşım düşünülmüş ve her bir değer için onda bir oranı baz alınmıştır. Yani her değer 10 ile bölünmüş ve bir virgül kaydırılmıştır.

Sonuçta girdi katmanındaki eğitim ve test kümeleri, dinamik olarak bir dosya aracılığı ile oluşturulan, sözlük ve liste veri yapılarında tutulan bilgileri ihtiva eder. Bu kümeler sistem genelinde hem YSA hem de K-Means oluşumunda, ek olarak, test aşamalarında bozulmadan saklanmaktadır. Bu listelerden sadece okuma işlemi yapılmaktadır.

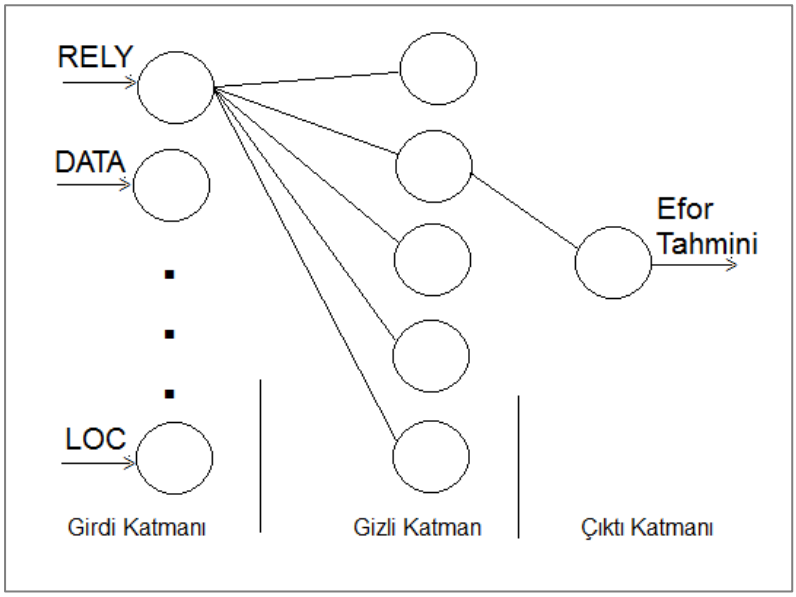
2.3. Yapay Sinir Ağı Yapısı

YSA bu çalışmanın ana temasını ve altyapısını oluşturmaktadır. Daha önce de ifade edildiği gibi, YSA tahmin değeri üretebilen bir araç olarak görülebilir. Özellikle lineer olmayan ilişkileri modellemede yardımcı olmaktadır. Yazılım efor tahmini düzleminde, eldeki parametrelere göre basit olmayan bir ilişki ağı ile tahmin değeri üretme noktasında faydalıdır.

İşlem katmanı, ilgili şekildeki gösterimden de anlaşılabileceği gibi, temelde YSA, K-Means ve bu yapılar arasındaki bağlantıyı içermektedir. Bir başka ifadeyle sistemin çalışan yönü, iş yükünü karşılayan bölümüdür. İşlem katmanına girdi olarak verilen eğitim ve test kümeleri, YSA için de girdi olmakta ve bu değerlere göre hesaplamalar yapılmaktadır.

YSA ve çeşitleri konusuna referansla bu çalışmada ileri beslemeli ve geri yayımlı algoritmasını kullanan çeşit benimsenmiştir. YSA 3 katmandan oluşturulmuştur ve katmanlarda herhangi bir bias nöron bulunmamaktadır. İlk katman, sistemin genel yapısı gibi, girdi katmanıdır. Bu katmanda COCOMO parametreleri sayısı kadar nöron, ek olarak LOC değerini karşılayacak bir nöron bulundurur. YSA içerisinde tek gizli katman vardır ve bu katmanda da 5 nöron bulunmaktadır. Girdi katmanındaki her bir nöron, tüm gizli katman nöronlarına bağlıdır. Yani girdi ve gizli katman nöronları kartezyen olarak bağlanmıştır. Ancak bu bağlantı, YSA formuna

uygun şekilde, girdi katmanından çıktı katmanı yönünedir yani tek yönlü bir bağlantı vardır. Çıktı katmanında ise tek nöron vardır ve bu nöron efor tahmin değerini ifade eder. Gizli katmandaki tüm nöronlar, 5 nöron, bu katmandaki tek nörona bağlanmıştır. Ayrıca, dış bir girdi değeri, döngüsel nöron bağlantısı ya da katman harici bir bağlantı yoktur. Düz bir ilişki mevcuttur. Şekil 2.3 genel yapıyı göstermektedir. Burada gösterim kolaylığı için girdi katmanında sadece birkaç nöron eklenmiştir. Nöron sayısı, parametre sayısı kadardır. Yine okların yönü basit bir gösterim ifade eder ve sadece belli nöronlar üzerinde, çizim karışıklığı olmaması için, belli nöronlarla bağlanmıştır. Bağlantı her bir katmanda bir önceki katmanladır. Nöronlar üzerinde ağırlık değerleri, katman bağlantıları, aktivasyon fonksiyonu ve hata değerleri vardır. Bu değerler ileride izah edilmiş, şekil içerisine eklenmemiştir.



Şekil 2.3. YSA Yapısının Gösterimi

Girdi katmanındaki sayı, doğrudan doğruya COCOMO parametreleri ile belirlenmiştir. Sistem içerisinde herhangi bir özellik indirgeme işlemi uygulanmamıştır. Bu durumda tüm COCOMO nitelik değerleri sistemde girdi olarak bulunmaktadır. Ek olarak bu çalışmada LOC değeri de girdi olarak dâhil edilmiştir. COCOMO hesaplamalarında LOC değeri kıymet ifade eden, sonuca doğrudan etkisi olan bir değerdir. Bu nedenle bu çalışmada da bir girdi olarak dâhil edilmiştir.

Gizli katmanın sayısı, yani YSA içerisinde kaç gizli katman olacağı noktasında önceki çalışmalar referans alınmıştır [5,8,9]. Bu çalışmalarda tek gizli katman

bulunmaktadır. Katmanın sayısının tahmin değeri üretmede yeterli olduğu belirtilmiştir. Katmandaki nöron sayısı da benzer şekilde önceki çalışmaların ışığında ve bazı ön denemeler neticesinde belirlenmiştir[5]. 5 adet nöron, hem özellik sayısı hem de eğitim kümesindeki eleman sayısı göz önünde bulundurularak belirlenmiştir. Her bir gizli katman nöronunda girdi katmanındaki eleman sayısını karşılayacak şekilde ağırlık değerleri vardır. Katmandaki nöronlar birbirleri ile bağlı değildir. Girdi katmanından girdi değerlerini alır ve ağırlıklara göre işleyerek çıktı katmanına iletir.

Çıktı katmanında tek nöron bulunmaktadır. Bu nöronun girdi değerleri, gizli katmandaki nöronlardan gelen ağırlıklandırılmış hesaplama değerleridir. Toplam işlemi sonucunda efor tahmin değeri ortaya çıkmaktadır. Bu değer sistemdeki en önemli sonuç değerlerindendir çünkü bu değer hem tahmin anlamını ifade eder hem de bu tahmine göre üst ve alt limitler bulunur. Eğitim ve test kümeleri normalize edilmiş olduğundan, sonuç değeri de normalizasyon formuna uygun olarak 0 ve 1 aralığında bir sayı olacaktır.

YSA nöronları içerisindeki ağırlık değerleri -1 ve 1 aralığındaki rastgele değerlere göre atanmıştır. Tüm ağ öncelikle bir başlangıç işleminden geçirilmiş, ağırlıklar verilmiştir. Aktivasyon fonksiyonu olarak Sigmoid fonksiyonu tercih edilmiştir. Bu tercih temelde diğer çalışmalar baz alınarak yapılmıştır; ilgili fonksiyonun mevcut veri yapısına uygun olduğu ifade edilebilir. Bununla beraber, kullanıcıya ilgili katmanlarda hangi fonksiyonun kullanılacağı konusunda bir liste içerisinden seçim yapma olanağı da sağlanmıştır. Sigmoid tercihine ek olarak, çalışmalar sırasında olası alternatif fonksiyonlar da denenmiş ve bu seçimin mevcut veri kümeleri için en uygun olduğu şeklinde, önceki çalışmalardan da yola çıkılarak karar kılınmıştır[5].

YSA ve nöron veri yapısı, basitçe bir nesneler ağı şeklindedir. Burada her bir nöron için, ilgili katmanın ihtiyacına göre, ağırlık değeri, nöron değeri, hata oranı ve aktivasyon fonksiyonu bulunur. Ek olarak, her bir nöron için bağlı olacağı diğer katman nöron listesi vardır.

YSA ileri beslemeli, geri yayımlı olarak çalışır. Bu durumda eğitim kümesi için her bir döngüde önce bir sonuç değeri bulunur sonra da bu sonuç değeri ile gerçek değer arasındaki fark ağa geri yansıtılır. Ağın eğitim aşamasında bu durum geçerlidir.

Ancak, test aşamasında sadece ileri yönlü bir akış gerçekleştirilmiştir. Buradaki tercihin sebebi, her bir değerin ağ içerisindeki ürettiği sonucu ayrı ayrı gözlemleyebilme isteğidir. Eğer geri yayılım gerçekleştirilecek olursa, her bir test satırı için ağ yeniden öğrenme işlemini yapacak ve muhtemelen bir sonraki satır için daha farklı bir sonuç üretecektir. Sistemin daha sağlıklı değerlendirilmesi amacıyla test aşamasında bir öğrenme işlemi yoktur.

2.4. K-Means Kümeleri

Yazılım efor tahmin değerini destekleyecek, olası üst ve alt limit değerlerini üretecek yapı, K-Means yöntemi ile gerçekleştirilmektedir. K-Means temelde bir kümeleme algoritmasıdır. Mevcut probleme uyarlanması ise, eldeki eğitim kümesinin belli sayıda alt kümelere ayrılarak, olası ortak özellikleri olan projeleri, girdi satırlarını aynı küme içinde toplayarak, daha sonra efor tahmin değerinin hangi küme içerisinde olduğunu belirleme şeklinde gerçekleştirilmiştir. Yani, önce eğitim kümesi alt kümelere bölünür. Sonrasında YSA ile üretilen efor tahmin değerinin hangi kümeye dahil edilebileceği bulunur ve bu küme içerisindeki bilgilere göre efor tahmin değerini destekleyecek diğer sonuçlar üretilir.

K-Means söz konusu olduğunda, elde bir veri kümesi vardır ve kümenin bölüneceği k adet alt küme bilinmektedir. Bunun haricinde her bir küme için başlangıç değerleri verilir. Bu değerlerin sonucu doğrudan etkilediği, farklı başlangıç değerleri ile farklı küme gruplarının ve dolayısıyla içeriklerinin oluşabileceği bilinmektedir. Bu çalışmada başlangıç değerleri YSA' da olduğu gibi, rastgele atanmıştır. Yine benzer şekilde -1 ve 1 aralığı belirlenmiş, buna göre değerler atanmıştır. Eğitim kümesi içerisinde, yine COCOMO parametreleri ve LOC değeri vardır. Hesaplamalara tüm nitelikler dâhil edilmektedir. K-Means kümelerinde gösterim düşünüldüğünde, nitelik sayısı kadar boyut olacağından, şekil ile gösterimi zor olmaktadır. Bu nedenle küme yapısı, içerik olarak şekil ile gösterilmemiş, detaylı şekilde izah edilmiştir.

Küme içerikleri belirlenirken Öklid uzaklığı esas alınmıştır. Buna göre ilgili satır için küme merkezine Öklid uzaklık değeri en küçük olan, o kümeye eklenmektedir. Bu hesaplama için her bir küme bazında, küme merkezlerinin de saklanması gerekmektedir. Küme merkezleri bir liste olarak, her bir proje için nitelik sayısı kadar eleman içermektedir. Hesaplamalar bu değerlere göre yapılır ve güncellenir.

Başlangıçta rastgele değerler de sadece bu bilgi için belirlenir. Küme merkezleri üzerinden her bir girdi satırı için hesaplama yapılır ve en yakın küme belirlenir. Öklid uzaklığı bulunurken de tüm nitelik değerleri hesaplamaya dâhildir; herhangi bir nitelik indirgeme işlemi mevcut değildir.

Eğitim aşamasında, doğal olarak ortaya mevcut tüm kümeleri kapsayacak şekilde alt kümeler çıkmaktadır. Test aşamasında ise küme yapıları üzerinde değişikliğe gidilmemiştir. Bir başka ifadeyle, test satırları için olası küme bulunmuş ancak bu veri bu kümeye dâhil edilmemiştir. Burada YSA durumunda olduğu gibi, sistem genelinde, her bir test satırı için kararlı bir durumun olması, her bir test verisi için aynı ortamın sağlanması hedeflenmiştir.

Yukarıda da ifade edildiği gibi, küme başlangıç değerleri, ortaya çıkacak küme elemanlarını dramatik olarak etkilemektedir. Algoritmanın niteliği, küme ortak noktalarının bulunması şeklinde işlediğinden, başlangıç noktası, bir sonraki aday satırın hangi kümeye dahil olabileceğini mevcut orta noktadan yola çıkarak belirler. Dolayısıyla orta noktaya yakın olan veriler ilgili küme içine gidecektir. Özellikle COCOMO ve bu yapıdaki eğitim kümesi göz önünde bulundurulduğunda, birbirine çok yakın nitelik değerleri olduğu gözlemlenmektedir. Bu durumda, başlangıçta rastgele atanmış küme merkezleri, eleman eklendikçe birbirine yaklaşacaktır. Eğitim ve test aşamalarında, küme merkezlerinin çok yakın olduğu ve küme oluşturma aşamasında bir kümenin sürekli ağırlık merkezini mevcut veri kümelerine yaklaştırdığı gözlemlenmiştir. Bunun sebebi, küme oluşturmada, belli bir sıra ile, olası en yakın küme hesabı yapılmasıdır ki bu da algoritmanın standart bir unsurudur. Bir başka ifadeyle, ilk eleman herhangi bir kümeye dâhil olduğunda, muhtemelen bir sonraki satıra yakın küme merkezi üretecektir. Gerçekten de sonuçta ortaya çıkan kümelerde, küme sayısı 2, 3, 4 ya da daha büyük olduğu durumlarda da, bazı kümelerin eğitim kümesinin çok büyük bir kısmını içerdiği, hatta bazı boş kümelerin meydana geldiği görülmüştür. Veri yapısının birbirine yakın değerler içeriyor olmasından ve algoritmadaki sıralı işlemde ilk yakın kümenin avantajlı olmasından dolayı, istenilen dengeli kümeler elde edilememektedir. Bu sorunun çözümü için yeni bir yaklaşım geliştirilmiş ve adına doluluk oranı denilmiştir. Doluluk oranı basitçe, bir küme içerisindeki maksimum eleman sayısının yüzde olarak ifadesidir. Yani sistem genelinde toplam eğitim kümesi eleman sayısından belli bir miktarı bir

küme içerisinde bulunabilir. Bir örnekle izah edilirse, doluluk oranı %40 olarak belirlenmiş bir çalışmada, eğitim kümesinin 100 örnekten olduğu düşünülürse, bir küme içerisinde en fazla 40 satır bulunabilir. Doluluk oranı, dışarıdan verilen bir sayı olmayıp, belli bir formül ile hesaplanmaktadır. Hesaplama yöntemi Denklem (2.1) deki gibi;

$$D=((T/K) + 1).C \quad (2.1)$$

şeklinde olmaktadır. Burada K sayısı, oluşacak küme sayısını ifade eder ve K-Means algoritmasının temel bileşenidir. T sayısı eğitim kümesindeki eleman sayısını ifade etmektedir. C sayısı ise bir sabit çarpan değeridir. Bu değere göre ortaya çıkan basit oran değeri esnetilebilmektedir. Hesaplama içerisindeki toplama işlemi, oran değerinin kesin sınırlarda kalmayıp, virgüllü bir sayı oluşabilmesi için verilmiştir. Bu hesaplama, veri kümesindeki niteliklerin birbirine çok yakın değerler içeriyor ve aslında sürekli sayılar olmamasından kaynaklanmaktadır. Bu tür bir yaklaşım geliştirilmediği takdirde kümeler faydasız olabilmektedir; limit değerleri yine yakın çıkacaktır ancak pratikte bir anlam olmayabilir.

K-Means için doluluk oranı, tüm kümeler için ortak bir değerdir. Yani sistem genelinde belirlenir ve bu her bir küme için, kümenin içerebileceği üst eleman limiti, satır sayısı anlamına gelir. Dolayısıyla her bir kümede bu değerin kontrolü için bilgi tutulur. İlgili kümeye bir eleman eklenirken, doluluk oranının aşılıp aşılmadığı kontrol edilmektedir. Bu kontrol için de, her bir eleman eklendiğinde, o küme için doluluk oranını yeniden hesaplanır. Bu değer sistem genelindeki sabit oran ile karıştırılmamalıdır. Her bir küme için mevcut doluluk oranını ifade eder.

Küme oluşturma aşamasında, bir satır için en uygun küme bulunduğunda, o küme için doluluk oranı aşılmışsa, artık o satır o kümeye dâhil edilmez. Bu durumda bulunan kümeden sonraki en iyi ikinci küme devreye girmektedir. Bu işlem tüm küme oluşturma aşaması için geçerlidir. Yani dolan kümeye eleman eklenmez. En yakın olası kümeye eklenir. Bu işlemin gerçekleştirilebilmesi için tüm kümelerin ilgili satır ile ilişkisi, uzaklık değerleri bilinmelidir. Sonrasında bu değerler sıralanmalı ve buna göre bir tercih oluşturulmalıdır. Zaten küme oluşturma aşamasında doğru kümenin bulunması için bir hesap yapılmakta, küme merkezine olan uzaklık değerleri hesaplanmaktadır. Burada ek olarak, her bir uzaklık değeri bir

liste olarak tutulur ve bir sıralama işleminden geçirilir. Küme oluşturma ölçütü olarak en yakın uzaklık değeri benimsendiğinden, liste, uzaklık değerlerine göre küçükten büyüğe göre sıralanmıştır. Sıralama işlemi basitçe Buble Sort(Kabarcık Sıralaması)[45] algoritması ile gerçekleştirilir. Sıralanmış uzaklık değer listesi içerinde tabii ki aynı zamanda küme referansları da bulunur. Bir elamanın hangi kümeye dâhil olacağı, bu listedeki en küçük elemandan itibaren kontrol edilir. Eğer ilgili liste elemanının, sıradaki liste satırının, dâhil olabileceği küme dolmuşsa, bir sonrakine geçilir. İşlem bu şekilde uygun küme bulunana kadar devam eder. Doluluk oranı basit bir eleman sayısı ve küme sayısı oranı olmadığından, mutlaka bir kümeye dâhil olma garanti edilmiştir. Bu değişiklikle daha dengeli ve ortak özellikleri olan kümelerin oluştuğu görüşmüştür. Bu da sistemin başarı oranını artıran önemli etkenlerden biridir.

Kümeler oluştuktan sonra, artık içeriği belli olan elemanlar ile bazı istatistiki bilgiler hesaplanabilir. Bu sayede küme ile ilgili genel bilgiler edinilebilir. Küme merkezine en yakın ve en uzak satırlar, sonuç değeri bazında en büyük ve en küçük değerler, sonuç değer ortalaması bunlardan bazılarıdır. Bir test satırı ilgili kümeye dâhil olabilir yorumunu oluşturacak sonuç üretildiğinde, küme içerisindeki veriler ve yukarıda listelenen değerlere göre daha farklı yaklaşımlar da geliştirilebilir.

2.5. YSA ile K-Means Kümelerinin Birleştirilmesi

Yazılım efor tahmin değeri, YSA ile bulunmaktadır. Ancak, bu değeri destekleyecek, sonuç üzerinden yorum yapma olanağı sağlayacak destek değerleri K-Means kümeleri ile hesaplanmaktadır. Bu iki yapının birbiri ile iletişimde olduğu açıktır. Temelde bu iki yapı, aynı eğitim kümesi ile beslenir ve bu ortak değer kümesi ile hazırlanır. Aynı şekilde test kümesindeki her bir satır için hesaplama yapılırken, her iki yapının da orijinal halinin korunması sağlanmış, her test satırı için aynı standart içyapı sunulmuştur. Aynı zamanda, karşılaştırma ve başarı seviyelerini kontrol için orijinal COCOMO hesaplaması da kullanılmış ve işlem katmanında test yapılırken ilgili satırın COCOMO hesaplamasında ne tahmin değeri ürettiği bulunmuştur. Bu değerin bulunabilmesi için COCOMO' nun ihtiyaç duyduğu nitelik değerleri zaten girdi katmanından gelmektedir. Ek olarak ortam sabiti de bu dosyada mevcuttur. Bu değer COCOMO hesaplamasında kullanılır ve çarpım işlemine dahildir. Tüm

parametrelerle hesaplama yapılır ve sonuç değeri, daha sonra, çıktı katmanında karşılaştırma unsuru olarak kullanılır.

YSA bir efor tahmin değeri üretir. K-Means ise mevcut verilere göre proje alt kümeleri oluşturur. Bu iki yapıdaki iç unsurlar, YSA efor tahmin değeri ile birleştirilmektedir. Hesaplanan efor tahmin değeri, K-Means kümeleri için girdi oluşturmaktadır. Burada sadece bir sayısal değer, tek değer olduğuna vurgu yapmak gerekmektedir. Yani, efor tahmin değeri, K-Means kümelerine sunulurken hedef, bu küme içerisindeki sonuç değerlerinden yola çıkılarak yorum yapılmasıdır. Dolayısıyla ilgili efor tahmini için doğru kümenin bulunması gerekmektedir. Bu işlem, yukarıda da izah edildiği gibi, her bir test satırı için olası küme bulunduktan sonra gerçekleşmektedir. Önce olası küme bulunur ve sonrasında bu küme içerisindeki değerlere göre, YSA tahmin değeri kullanılarak üst ve alt limit değerleri hesaplanır. Küme bulma işlemi daha önce izah edilmiştir; Öklid uzaklığı tabanlı, doluluk oranına bağlı olan, ilgili test satırı için en yakın küme bulma işlemidir. Bulunan küme içerisinde üst ve alt limit değerlerinin bulunması ise, temelde sonuç değerlerinin sıralanması ve en yakın üst ve alt komşuların bulunması ile gerçekleştirilir.

Hedef küme bulunduktan sonra, bu küme içerisindeki sonuç değerleri, önceki proje değerlerindeki gerçekleşmiş efor değerleri, artan şekilde sıralanır. Bu sıralama için de Buble Sort[45] algoritması kullanılmıştır. Bu sıralamaya, ek olarak, YSA efor tahmin değeri de eklenir. Bu tahmin değerine en yakın üst ve alt komşu, üst ve alt limit olarak belirlenir. Bu şekilde bir proje için efor tahmin değeri ve bu değere bağlı olarak üst ve alt limitler hesaplanmış olmaktadır. Sıralama işleminde, YSA değeri eğer liste içerisinde ise, bir üst ve bir alt değer bulunabilir. Bu değerler doğal olarak sınır değerleri olarak kabul edilir. Ancak, tahmin değeri, liste içerisindeki değerlerden daha küçük ya da daha büyük olabilir. Eğer tahmin değeri, listedeki elemanlardan, yani ilgili küme sonuçlarından daha küçükse, alt limit değeri kendisi, üst limit değeriyse listenin ilk elemanı olacaktır. Tersi durumunda, tahmin değerinin listedeki elemanlardan büyük olması durumunda üst limit değeri kendisi, alt limit değeri de listenin son elemanı olacaktır. Özellikle sıralama düşünüldüğünde, küçük ya da büyük olma noktasında bir yönde herhangi bir destek değeri oluşmadığından, YSA tahmin değeri kullanılmak durumundadır. Diğer durumlarda, yani liste içerisine

düştüğü durumlarda üst ve alt değerler, basitçe en yakın değerlere göre belirlenmektedir.

YSA ile efor tahmin değeri üretilir, bu değerle de limit değerleri hesaplanmaktadır. Ayrıca küme içerisinde destek olacak birçok değer de hesaplanabilmektedir. Bu değerler ve ne anlama geldikleri konusu aşağıda izah edilmiştir

2.6. Sonuç Katmanındaki Değerler

Girdi katmanı ile başlayıp, işlem katmanı ile devam eden sistem, doğal olarak bir çıktı katmanı ile neticelenmektedir. Girdi katmanında çalışma verileri, işlem katmanında hesaplama unsur ve yapıları vardır. Çıktı katmanında, bu iki katmanın çalışması sonucunda ortaya çıkan değerler ve bu değerlerin anlamlı bir bütün olarak sunumu vardır.

Sistemde temel çıktılar, YSA tahmin değeri, üst ve alt limitler ve küme bilgileridir. Bu değerler haricinde birçok çıktı değeri daha vardır. K-Means tahmin değeri, ilgili test satırı için bulunan kümedeki en yakın sonuç değeridir. Bu değer en yakın Öklid uzaklığına sahip olan satırın değeridir. En yakın küme, limit ve diğer hesaplamalarının yapıldığı kümenin hangisi olduğu belirtir bir anahtar değerdir. Küme istatistik değerleri, minimum, maksimum ve ortalama değerleridir ve bulunan küme içerisinde zaten hesaplanmış olan değerlerdir. Test satırının sonuç değerinin küme içerisinde olup olmadığını belirten bilgi, sadece sonuç değerlerine göre aralığı kontrol edilen mantıksal değerdir. YSA sonucunun ilgili küme içerisinde olup olmadığını belirten bir sonuç değeri de vardır. Ek olarak, tüm incelemelerin yapılabilmesi için, beklenen sonuç değeri, orijinal COCOMO hesaplama sonuç değeri de sunulmaktadır. Ayrıca, tahmin değerlerinin başarı oranını belirtir yüzde olarak ifade edilen alanlar da vardır. Bu alanlar için hesaplama yöntemi ve anlamı, yorum yapmada yardımcı olacak alanlar için, test sonuçlarında izah edilecektir.

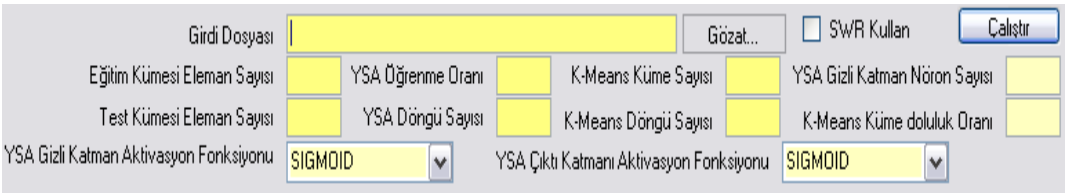
Tüm bu değerler ile beraber, sistem hakkında bilgi veren, konfigürasyon değerlerini ifade eden çıktılar da vardır. Buna göre YSA katmanları ve her bir katmandaki nöron sayısı, öğrenme oranı, YSA döngü sayısı- ağın eğitim için kaç döngü yapacağı yani öğrenme işleminde ileri besleme ve geri yayılma adımlarının art arda kaç sefer yapılacağı, test ve eğitim kümesindeki veri sayıları, SWR metodunun kullanılıp

kullanılmadığı ve küme sayısı bu değerlerdendir. Buna göre mevcut çalışma ortamı kontrol edilebilmektedir.

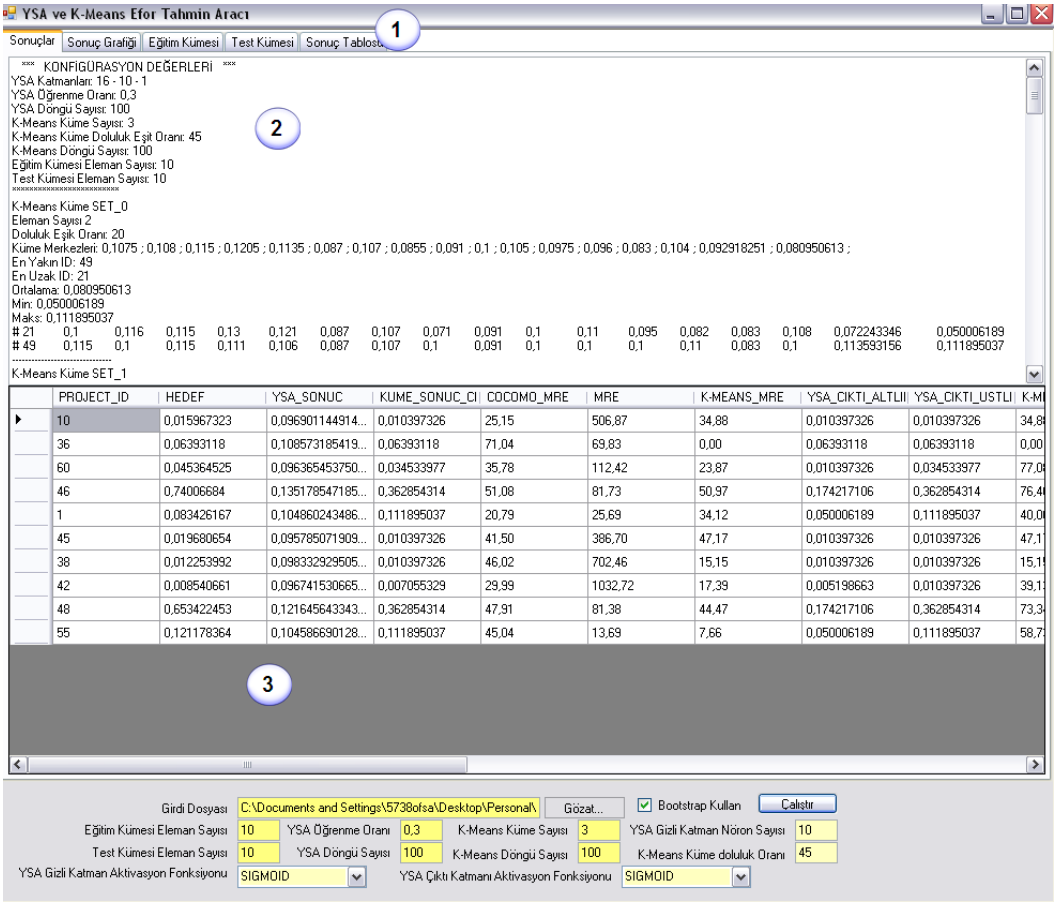
Karar destek sistemi olarak ortaya konulacak bit yapıda, tahmin sistemlerinde çokça sonuç değerinin üretilmesi okunmasını zorlaştırabilir. Ancak burada önceliğin asıl sistem sonuç değerlerine, yani YSA tahmini, üst ve alt limitler ve başarı oranlarına verildiği, diğer bilgilerin de ileri inceleme için listelendiği belirtilmelidir.

2.7. Geliştirilen Araç ve Sistem İşleyişi

Ortaya konulan modelin incelenebilmesi, sonuçlarının karşılaştırılması, bazı ayarların yapılabilmesi ve diğer sunum imkânlarının olabilmesi için bir yardımcı program geliştirilmiştir. Visual Studio 2005 ortamında, C# dili ile, Windows uygulaması[46] olarak geliştirilen programda bir girdi arayüzü, arka planda iş kurallarını ifade eden sınıflar ve sonuç değerlerinin gösterildiği tablo, grafik ve metin alanları vardır. Özellikle model geliştirilirken güçlü ve zayıf yönlerin takip edilebilmesi, gerektiği durumunda ayar yapılabilmesi ve en iyi sistemin ortaya konabilmesi için mevcut program ya da araçlar kullanılmamış, yeni bir araç geliştirilmiştir. Ancak içerisindeki K-Means ve YSA algoritmaları, standart, kabul edilen algoritmalardır ve bir değişiklik yoktur. Geliştirilen sınıflar ve diğer hesaplama unsurları, standart kütüphane (.Net) elemanlarının yardımıyla gerçekleştirilmiştir. Şekil 2.4, programın girdi alanını göstermektedir. Programın örnek bir çalışma sonucu da Şekil 2.5 ile gösterilmiştir.



Şekil 2.4. Program Üzerindeki Girdi Alanları

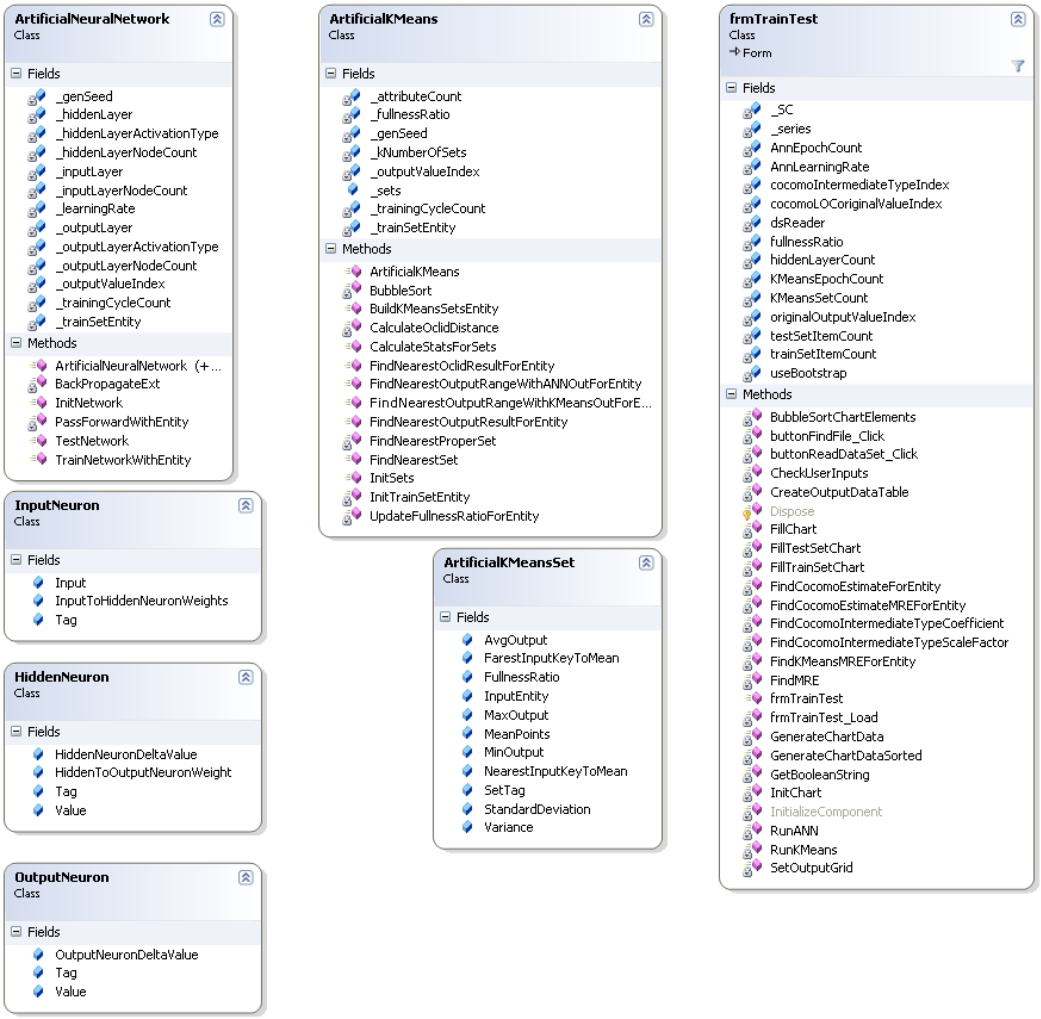


Şekil 2.5. Örnek Program Çıktısı

Şekil 2.5 üzerinde görülen 1 numaralı alanda, çalışma sonuçları ile ilgili farklı sunum türleri gösterilmektedir. Genel bilgiler şekilde görülen alanlarda sunulmaktadır ancak bunların haricinde, program içerisinde sonuç kümesi, eğitim ve test kümeleri grafikleri ve sonuçların html tabanlı gösterimi de eklenmiştir. Aynı bilgiyi gösterdiğinden, buraya eklenmemiştir ancak, basit bir grafik ara yüzünün kullanıcıya fayda sağlayacağı da bir gerçektir. 2 numaralı alanda, çalışma sonucunda ortaya çıkan kümeler, her bir test aşamasının metinsel ifadesi ve konfigürasyon değerleri gösterilmektedir. 3 numaralı alanda ise, çalışma sonuçları bir tablo olarak, karşılıklı inceleyebilmek için birleştirilerek sunulmaktadır.

Girdi olarak en temel eleman, girdi katmanını oluşturan verileri içeren dosyadır. Bu dosya, ilgili buton ile yüklenir. Diğer girdi değerleri, sistem konfigürasyonu olarak adlandırılan, sistem içerisinde değiştirilebilir tüm özellikleri içermektedir. Bu değerlerin tümü sistem içerisinde kullanıldığından, dışarıdan verilmesi gerekmektedir.

Program içerisinde hesaplama için belli sınıflar kullanılmaktadır. Bu sınıflar YSA ve K-Means öğelerini, eğitim ve test verileri ile sonuç değerlerini içermektedir. Sonuç değerleri basit bir şekilde veri tablosu şeklinde kurgulamışken, diğer unsurlarda sınıf ve sınıf içi değerler kullanılmıştır. Şekil 2.6 sınıf özelliklerinin genel görünümü sunmaktadır.

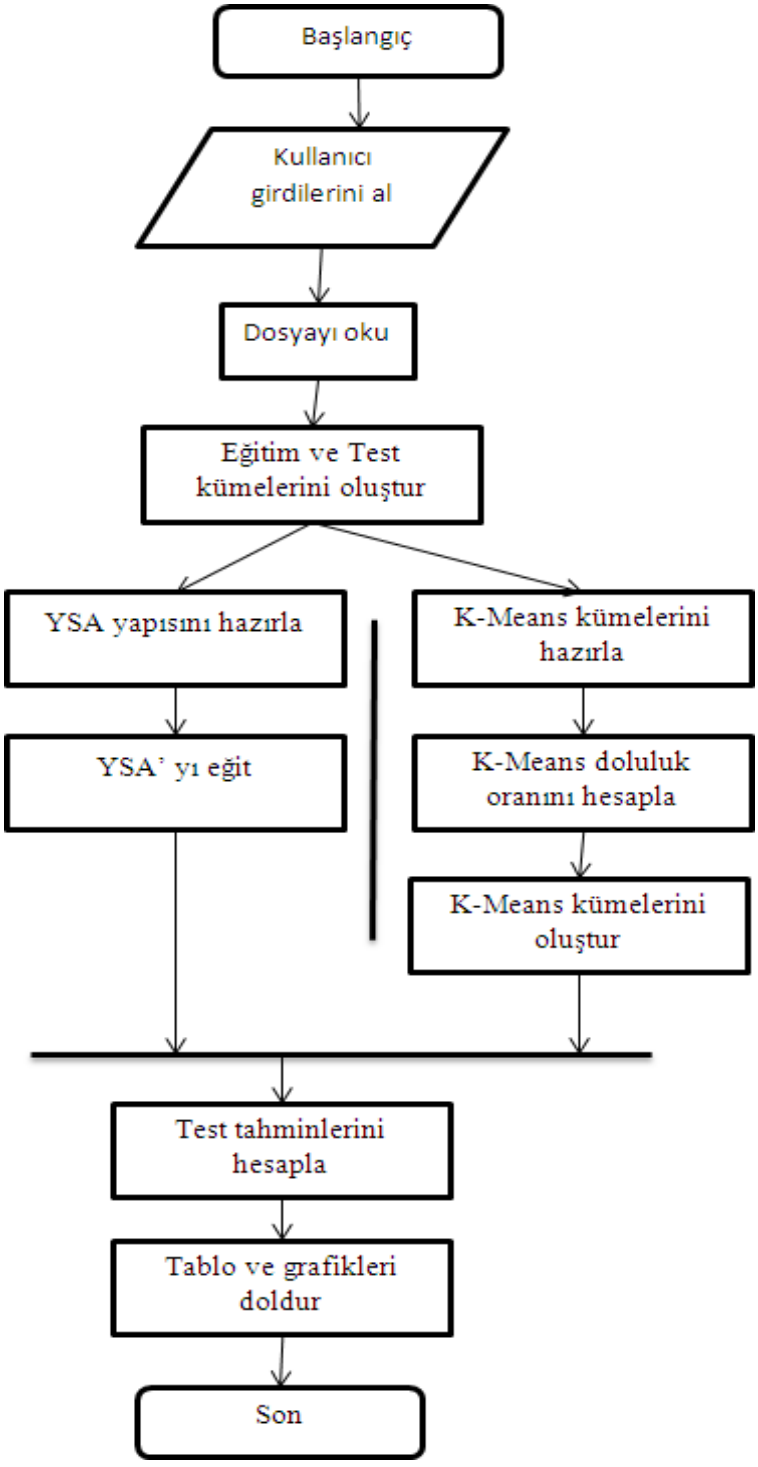


Şekil 2.6. Program Sınıf Özellikleri ve Metotlar

K-Means kümeleri için bir sınıf vardır. Bu sınıf içerisinde küme ile ilgili bilgiler bulunur ki bunlar çoğunlukla istatistiksel bilgileri ifade eder. Ayrıca, küme merkez noktalarının tutulduğu liste ve küme elemanlarının tutulduğu sözlük yapısı vardır. K-Means oluşturma, istatistikleri hesaplama, test etme, üst ve alt limit değerlerini bulma işleri ile görevli, K-Means nesnelerini içeren ayrıca bir işlem sınıfı da vardır. Benzer şekilde, YSA katmanlarındaki nöronları belirten sınıflar vardır. Her bir katman için ayrı gereksinimler olduğundan ayrı sınıflar mevcuttur. Bu sınıflar

içerisinde nöron ismi, değeri, hata oranı, ağırlıklar gibi özellikler vardır. Yine bu sınıfları içeren, YSA oluşturma, eğitme ve test etmeye, efor tahmin değeri üretmeye yarayan bir işlem sınıfı daha vardır. Tüm bu sınıflar, programın ana formu tarafından kullanılmaktadır.

Programın çalışma mantığı sıralı olarak basitçe ifade edilirse, önce kullanıcıdan girdiler alınır. Belirlenen parametrelere göre eğitim ve test kümeleri belirlenir, YSA ve K-Means kümeleri ve bunlara ait diğer değerler oluşturulur. Test kümesindeki her bir satır için işlem yapılır ve her bir işlem ilgili tablo, metin ve grafik alanlarına eklenir. Tüm satırlar işlendikten sonra program sonlanır ve sonuçlar ilgili alanlarda kullanıcıya gösterilir. Bu işleyiş genel olarak Şekil 2.7' deki gibidir. Bu şekilde test işlemlerindeki döngü tek bir işlem olarak gösterilmiştir ancak bu aşamada çeşitli kontrol ve karşılaştırma metotlarının çalıştığı anlaşılabilir. Aynı zamanda, YSA ve K-Means kümelerinin oluşturulması(eğitilmesi) ayrı program alt parçaları tarafından paralel olarak gerçekleştirilmektedir. Bu nedenle çizimde bir ayrım ve sonrasında birleştirme işlemi vardır. Ek olarak, aşağıda bu işlemi detaylı şekilde izah eden sıralı maddeler vardır. Akışın daha rahat anlaşılabilmesi için metinsel, basamaklı anlatım tercih edilmiştir.



Şekil 2.7. Program Akışının Genel Görünümü

- Kullanıcı sistem parametrelerini belirtir.
  - Dosya adı ve diğer tüm parametrelerin doldurulduğu kontrol edilir.
    - Eğer eksiklik varsa kullanıcı uyarılır ve işlem kesilir.
    - Tüm bilgiler doluysa işleme devam edilir.

- Dosya konumu verilen girdi dosyası, eğitim ve test kümelerinin oluşturulması için okunur.
  - Dosya varlığı kontrol edilir.
    - Dosya bulunamazsa uyarılır ve işlem kesilir.
    - Dosya mevcutsa işleme devam edilir.
  - Dosya içeriğin okunması için açılır.
    - Dosyadaki tüm satırlar okunur.
- Her bir satır için, dosya satırındaki nitelik sayısını yönetecek şekilde listeye veriler doldurulur.
- Verileri tanımlayacak proje numarası da bu listenin tanımlayıcısı olarak belirtilir.
- Rastgele eğitim ve test kümelerinin oluşturulabilmesi için proje referans numaralarını oluşturan liste doldurulur.
  - Satırların okunması tamamlandıktan sonra dosya kapatılır.
  - Eğitim ve test kümeleri okunan listeye göre, verilen sayılarda eleman içerecek şekilde oluşturulur.
    - Referans listesinden rastgele bir sırada proje numarası bulunur
    - Bu proje satırı ilgili kümeye (eğitim/test) kopyalanır.
    - SWR metodu kullanılmıyorsa bulunan proje numarası referans listesinden çıkarılır ve listedeki boş satır silinir.
  - Oluşturulan kümeler değişmemek kaydıyla işlem katmanına girdi olarak sunulacak şekilde saklanır.
- YSA, verilen sistem parametrelerine göre kurgulanır.
  - Öğrenme oranı, döngü sayısı ve diğer parametreler belirtilir.
  - İlgili ağ sınıf yapısı oluşturulur.
    - Her bir katmandaki nöron sayısına göre sınıflar oluşturulur ve birbirine bağlanır.
  - Ağırlıklar için rastgele değerler atanır.
- YSA, girdi olarak verilen eğitim kümesi ile eğitilir.
  - Belirtilen eğitim döngü sayısı kadar,
    - Girdi katmanına değerler verilir ve ileri besleme işlemi gerçekleştirilir.
    - Sonuç ve beklenen sonuç karşılaştırılarak hata değeri ağa geri yayılır.
- K-Means kümeleri için doluluk oranı hesaplanır.

- K-Means kümeleri sistem parametrelerine göre kurgulanır.
  - Küme sayısı kadar,
    - Yeni bir küme sınıfı oluşturulur.
    - Bu kümeye doluluk oran değeri, sınırı belirtilir.
    - Merkez noktaları rastgele atanır.
    - Küme istatistik değerleri başlangıç değerlerine göre verilir.
- K-Means kümeleri verilen eğitim kümesine göre eğitilir, kümeler oluşturulur.
  - Belirtilen döngü sayısına ulaşıncaya ya da kümelerde değişiklik olmayıncaya kadar,
    - Her bir satır için uygun küme bulunur.
    - Doluluk oranına göre ilgili kümeye satır eklenir.
- Eğer önceki bir kümeye dâhilse bu kümeden silinir, hedef kümeye eklenir.
- İlk defa ekleniyorsa sadece yeni kümeye eklenir.
    - Her bir küme için doluluk oranı yeniden hesaplanır.
    - Küme merkezleri yeniden hesaplanır.
  - K-Means kümeleri için istatistikler hesaplanır.Her bir küme sayısı kadar,
    - Küme minimum, maksimum ve ortalama sonuç değerleri bulunur.
    - Öklid uzaklık bağıntısına göre en yakın, en uzak gibi diğer istatistiksel değerler hesaplanır.
- Test aşamasına geçmeden önce, mevcut durumu gösterecek şekilde, K-Means küme içerikleri ve istatistikleri formda gösterilir (gösterim için hazırlanır).
- Sistem girdi katmanından alınan test kümesine göre test edilir ve sonuçlar hesaplanır.
  - Her bir test satırı için,
    - YSA efor tahmin sonucu bulunur.
    - Bu sonuca göre hata oranı hesaplanır.
    - Orijinal COCOMO tahmin değeri hesaplanır.
    - K-Means kümelerinden hangisine dâhil edilebileceği hesaplanır.
    - YSA sonucuna göre en uygun küme bulunur, hangi kümeye dâhil olabileceği bu sefer YSA sonucuna göre hesaplanır.
    - İlgili küme sonucuna göre hata değeri hesaplanır.
    - YSA tahmini ile ilgili küme sonucuna göre hata değeri hesaplanır.
    - Test satırının ilgili küme içerisinde olup olmadığı kontrol edilir.

- YSA tahminine göre K-Means kümesi ile üst ve alt limit değerleri hesaplanır.
- Üst ve alt limit değerlerine göre hata değerleri hesaplanır.
- K-Means ve YSA sonucu ile bulunabilecek diğer hata oranları (maksimum, minimum sonuç) hesaplanır.
- YSA ve üst-alt limit hata oranları toplanır.
  - Toplam hata oranları, eleman sayısına göre bölünür ve sistem hata oranı hesaplanır.
- İşlem katmanında test süresince ortaya çıkan değerler, hesaplanan ilgili sonuçlar ve karşılaştırma değerleri ile, küme içerikleri ve sistem parametreleri kullanıcıya gösterilir.
  - Küme içerikleri gösterilir.
  - Sistem parametreleri ve konfigürasyon değerleri gösterilir.
  - Her bir test döngüsü için, hata oranı, beklenen değer, tahmin değeri gibi sonuçlar gösterilir.
  - Sonuçların tablo ve grafik sunumları hazırlanır ve gösterilir.
- İşlem tamamlanır.

YSA ve K-Means birleştirilirken ve gerçekte daha öncesinde bu yapılar oluşturulurken bazı sorunlar ortaya çıkmış ve bunlara göre birtakım çözüm yöntemleri belirtilmiştir. Sorunların en başında rastgele değer üretme sayılabilir. Neticede deterministik programlama ortamında gerçekleştirilen rastgele sonuç üretme fonksiyonu, bir sistem kütüphanesi işlemi, her durumda benzer sekanslar üretmektedir. Tabii ki bu sekansın oldukça geniş olması rastgele değerler üretilmesinde gereken seviyeyi sağlar. Ancak, YSA ve daha da önemlisi K-Means için başlangıç değerlerinin dramatik etkisi düşünüldüğünde, rastgele değerlerin daha bilinçli yönetilmesi ya da rastgele oranlarının artırılması gereklidir. Sorun bir başka şekilde ifade edilirse, programın her çalışmasında rastgele fonksiyonları aynı değerleri döndürmektedir. Gerçekte test ve eğitim kümelerinin istenilen seviye ya da sonuçlara ulaşabilmesi için birçok defa çalıştırılması gerekir. Bu nedenle rastgele değer üreten fonksiyon için, ayrıca, bir çarpan listesi hazırlanmış, önce bu liste üzerinden rastgele değer bulunmuş ve sonuç değeri bu çarpan ile çarpıldıktan sonra hesaplanmıştır. Çarpım işleminden bahsedildiği için, burada oransal olarak rastgele değeri koruyacak, değerler tercih edilmiştir. Bu şekilde yapıldığında daha sağlıklı

sonuçların alındığı, en azından ilk durumdan daha iyi sonuç üretildiği gözlemlenmiştir. Bir başka vurgulanması gereken nokta, yine rastgele değerlere bağlı olarak, K-Means kümelerinin oluşturulmasıdır. Bu problemin çözümü için, yukarıda izah edildiği gibi, doluluk oranı adında bir yaklaşım benimsenmiştir.

Önemli noktalardan bir diğeri girdi katmanıyla alakalıdır. Bilindiği gibi girdi katmanında COCOMO parametrelerine uygun, normalize edilmiş ve ileri aşamada eğitim ve test kümelerinin oluşmasını sağlayan dosya vardır. Bir ön işleme uygulaması olarak dosyadaki değerler normalize edilmektedir. Normalizasyon işleminde sonuç değerlerinin yakınlığının etkisi yüksek olduğundan, ortaya çıkan LOC ve gerçekleşen efor değerleri mümkün ölçüde çok basamaklı olarak hesaplanmıştır. Yani bu değerlerin virgülden sonraki basamak sayısı mümkün ölçüde artırılarak hassasiyet derecesi ilerletilmiştir. Bir başka durum, sonuçların sunumu, yani çıktı katmanı ile ilgilidir. Birçok farklı değerin bulunması ve bu değerlerin bir kısmının normalize edilmiş, 0 ve 1 arasında virgüllü sayı olması, sunum zorluğu çıkarmıştır. Bunu çözebilmek için hem tablo, hem metin, hem grafik ve hem de html tablolar kullanılmıştır.

3. BULGULAR VE TARTIŞMA

Bu bölümde ortaya konan modelin test verileri ile denenmiş, çeşitli konfigürasyonlara göre sonuçları belirtilmiş ve bu sonuçlar ile ilgili çeşitli yorumlar yapılmıştır. Programın farklı parametrelerle çalıştırılması sonucunda çıkan sonuçlar, parametre değerleri ile beraber, ilgili tablo ve şekillerle verilmiştir. Tüm test sonuçlarında ortaya çıkan değerler, girdi katmanından gelen dosyaların, işlem katmanında değerlendirilmesi ile üretildiğinden yine normalize edilmiş şekildedir. Bununla beraber, sonuç değerlerinin gerçek değerlere (ilgili proje için beklenen efor değerine) oranı, yüzde cinsinden verilmiştir. Bu şekilde hem sonuç değeri bazında değerlendirme yapılabilir hem de oransal anlamda yoruma gidilebilir.

Bulgular, model gerçekleştirimi için ortaya konan program aracılığı ile üretilmiştir. Bu araç ile, daha önce de ifade edildiği gibi belli değerler kullanıcı tarafından verilebilmektedir (öğrenme oranı, eğitim/test küme eleman sayısı, küme oluşturma tekniği gibi). Bu noktada özellikle kümeler oluşturulurken yerine koyma yöntemi kullanıp kullanılmadığı kritiktir ve testler de bu 2 farklı seçim üzerindedir. Yani her iki yaklaşım için de program çalıştırılmıştır. Diğer parametreler için de testlerle uygun değerler bulunmaya çalışılmıştır. Özellikle modeli geçerleme amacı ile belli bir öğrenme oranı, döngü sayısı ve eleman sayısı belirlenmeye çalışılmıştır. Bu şekilde farklı parametrelerle ne tür sonuçlar alındığı incelenebilmiş, modele etki edecek değerler gözlemlenebilmiştir.

Değerlendirme kriteri, ortaya çıkan sonuçları irdelemede önemlidir. Bu çalışmada, önceki bölümlerde referans verilen MRE değeri baz alınmıştır[5,42]. MRE Denklem (3.1) deki gibi;

$$(|\text{Gerçekleşen-Tahmin}| / \text{Gerçekleşen}).100 \qquad (3.1)$$

şeklinde ifade edilmektedir. MRE değerinin düşük çıkması, formülden de anlaşılabileceği gibi, beklenen değere daha yakın sonuçların tahmin edildiği

anlamındadır. Değerin büyük çıkması istenmeyen bir durumdur dolayısıyla MRE değeri ne kadar küçükse model o kadar iyidir ya da iyi sonuç vermiştir denilebilir. MRE için %25 ve daha aşağı değerlerin kabul edilebilir olduğu ifade edilmektedir[5,6,8].

Sonuçlar üzerinde değerlendirme yapmaya geçmeden önce belirtilmesi gereken önemli bir durum da, bu çalışmada özellikle rastgele seçimlerin ve YSA, K-Means küme atamalarının rastgeleliğinin, sonucu etkilediğidir. Bu nedenle program birçok kez çalıştırılmış ve bu çalıştırmalar sonucunda en iyi değerleri veren döngü esas alınmıştır. Bir başka çalıştırmada aynı değerleri çıkma ihtimali olmayabilir, ancak değerlerin birbirine denk çıkacağı ifade edilmelidir.

Her bir çalışma için hem sonuç tablosu verilmiş, hem de grafiklerle desteklenmiştir. Efor tahmini sürecinde ortaya çıkan değerler, çıktı katmanında izah edildiği gibi, birden çoktur. Bunlar içerisinde kimi değerler normalize edilmiş sonuç değerleri, kimileri oran ifade eden MRE yüzdelerindir. Tablo okumaları buna göre yapılabilir. Grafikler ise, ilgili tabloda sonuç ve limit değerleri arasındaki ilişkileri ifade etmektedir. Her bir tablo ve grafik üzerinden aşağıda daha detaylı geçilecektir ancak, özellikle grafik üzerinde okumanın daha rahat yapılabilmesi için temel sonuç değerlerinin gösterildiği bilinmelidir.

Bu çalışmada, açık olarak ulaşılabilen PROMISE sitesi üzerinden edinilebilen COCOMO veri kümeleri kullanılmıştır[41]. Program içerisinde NASA 60, NASA 93 ve COCOMO 81 adlarını alan veri kümeleri eğitim ve test süreçlerine dâhil edilmiştir. Bu kümelerde sırasıyla, 60, 93 ve 63 satır bulunmaktadır. Veri kümeleri ifade edilirken de vurgulandığı gibi, normalizasyon sürecinden sonra sınır değerleri olan satırlar çıkarılmıştır. Hangi küme için kaç elemanla çalışıldığı konfigürasyon bilgileri ile verilmiştir. Programın her bir çalışma sürecinde bu 3 dosya için de deneme yapılmıştır. Özetlenecek olursa, 2 farklı küme oluşturma yöntemi, 3 farklı veri kümesi ile toplam 6 sonuç üretilmiş ve bu sonuçlar gösterilmiştir. Çalışmada bu veri kümelerinin kullanılmasının temel nedeni, açık olarak ulaşılabilmeleri, diğer çalışmalarda da kullanılmış olduklarından ortak değerlendirme yapabilme olanağı ve her bir kümenin COCOMO standardında olmasıdır. Kümeler aynı yapıda olmakla beraber, farklı projeleri ihtiva etmektedir.

3.1. SWR Kullanarak Yapılan Testler

Bu bölümde, programda SWR (Yerine Koyma) yöntemi kullanılarak oluşturulan veri kümeleri ile yapılan çalışmaların sonuçları verilmiştir. Bilindiği gibi bu yöntemde, alt küme oluşturma aşamasında yerine koyma mantığı baz alındığından, kümelerde aynı elemanların, aynı proje değerlerinin olduğu gözlemlenebilir.

3.1.1. NASA 60 veri kümesi üzerinde yapılan çalışma

Konfigürasyon değerleri:

- YSA Katmanları: 16 - 5 - 1
- YSA Öğrenme Oranı: 0,2
- YSA Döngü Sayısı: 3000
- K-Means Küme Sayısı: 4
- K-Means Küme Doluluk Eşit Oranı: 30,638299089797
- K-Means Döngü Sayısı: 1000
- Eğitim Kümesi Eleman Sayısı: 47
- Test Kümesi Eleman Sayısı: 15

Yukarıda ilgili program akışı içerisindeki parametreler belirtilmiştir. Program bu bilgilere göre çalışmış ve devam eden bölümlerde tablo ve grafik olarak verilen sonuçları üretmiştir. Diğer tüm veri kümeleri ve test sonuçları için de aynı şekilde önce program parametre değerleri belirtilmiş, sonrasında sonuç tablo ve grafikleri sunulmuştur. Toplamda 6 test olduğu düşünüldüğünde, 6 sonuç içerisinde de ayrı ayrı konfigürasyon değerleri, sonuç tabloları ve grafikleri görülecektir.

Tablo 3.1, bu program döngüsü sonrasında ortaya çıkan değerler ile hedef değerlerini vermektedir. Burada özellikle okuma kolaylığı göz önünde bulundurulduğundan, önce kısa bir sonuç tablosu verilmiş, sonrasında ise detaylı bilgileri içeren tablo sunulmuştur. Tablo 3.2, aynı döngü sonucunda ortaya çıkan diğer sonuçları ifade etmektedir. Tablolardaki ID değeri, test kümesindeki proje numarasını ifade etmektedir. Hedef, o proje satırı için gerçekleşen efor değerini belirtir. YSA sonuç, YSA' nın ürettiği efor tahmin değeri ve K-Means sonuç da sadece K-Means kümeleri kullanıldığında oluşacak sonuç değeridir. Bütün değerler normalize edilmiş şekliyle verilmiştir.

Tablo 3.1. SWR İle Nasa 60 Veri Kümesi İçin Hedef ve Sonuçlar

ID	Hedef	YSA Sonuç	K-Means Sonuç
59	0,027974	0,028604	0,025251
58	0,174217	0,096461	0,145934
17	0,062384	0,045382	0,050006
23	0,019062	0,031143	0,019062
42	0,008541	0,028054	0,008541
2	0,362854	0,414734	0,383587
22	0,016586	0,029619	0,019062
36	0,063931	0,131194	0,071358
11	0,032677	0,030686	0,033791
33	0,106573	0,07705	0,108801
49	0,111895	0,05854	0,108801
10	0,015967	0,029001	0,015967
35	0,108801	0,132995	0,108801
35	0,108801	0,132995	0,108801
10	0,015967	0,029001	0,015967

Tablo 3.2 ile değerler üzerinde karşılaştırmalar yapılabilir. Bu tablodaki alanlar öncelikle MRE değerleri ve sonrasında diğer bilgilerdir. Ortaya konan modelin daha iyi yorumlanabilmesi için orijinal COCOMO hesaplaması sonrasında oluşan MRE değeri de eklenmiştir. Alt limit ve üst limit değerleri, bu çalışma kapsamında en temel öğeler olan, YSA değerini destekleyecek sonuçlardır ve bu sonuçlara ait MRE değerleri de verilmiştir. En yakın küme bilgisi de, yine temel çıkış noktası olan YSA sonucunda göre uygun küme bulma işlemi sonucunda ortaya çıkan kümenin belirtecidir. Yani limit değerleri bu küme belirtecine göre yapılmaktadır. Örnek

olarak en yakın küme bir proje test satırı için 2 olarak çıkmışsa, YSA tahmin değerine göre üst ve alt limit değerleri bu küme üzerinden hesaplanmıştır.

Tablo 3.2. SWR İle Nasa 60 Veri Kümesi İçin MRE Değerleri

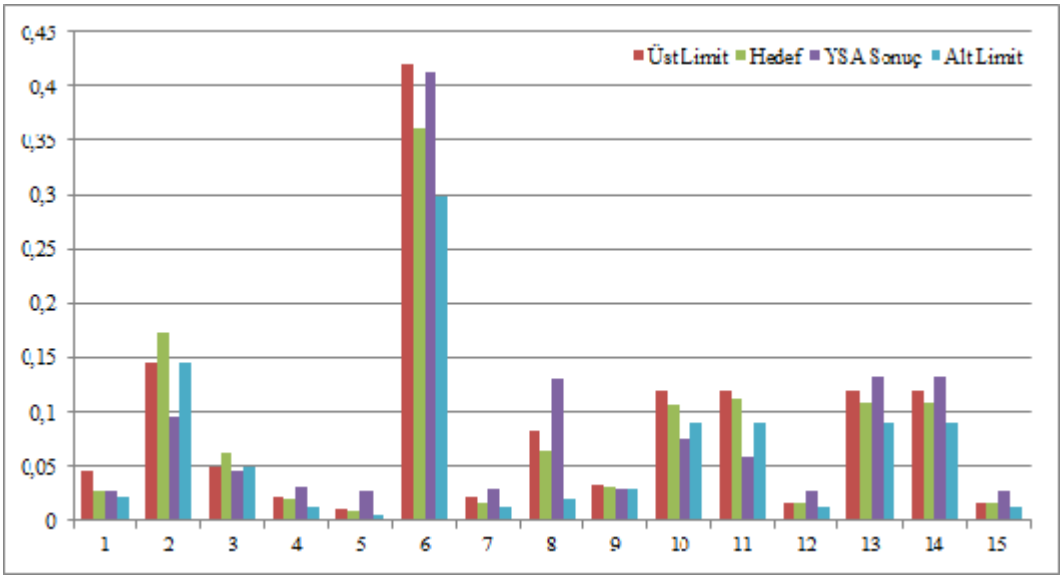
ID	COCOMO MRE	YSA MRE	K-MEANS MRE	Alt Limit	Üst Limit	Alt Limit MRE	Üst Limit MRE	En Yakın Küme
59	42,03	2,25	9,73	0,022775	0,045365	18,58	62,17	2
58	24,5	44,63	16,23	0,145934	0,145934	16,23	16,23	3
17	28,11	27,25	19,84	0,050006	0,050006	19,84	19,84	0
23	21,75	63,38	0	0,012254	0,022775	35,71	19,48	2
42	29,99	228,47	0	0,004827	0,010397	43,48	21,74	2
2	23,6	14,3	5,71	0,29849	0,42072	17,74	15,95	1
22	27,25	78,58	14,93	0,012254	0,022775	26,12	37,31	2
36	71,04	105,21	11,62	0,019681	0,083426	69,22	30,49	3
11	34,14	6,09	3,41	0,030511	0,034534	6,63	5,68	0
33	34,53	27,7	2,09	0,090234	0,121178	15,33	13,7	3
49	36,71	47,68	2,77	0,090234	0,121178	19,36	8,3	3
10	25,15	81,63	0	0,012873	0,016586	19,38	3,88	0
35	10,23	22,24	0	0,090234	0,121178	17,06	11,38	3
35	10,23	22,24	0	0,090234	0,121178	17,06	11,38	3
10	25,15	81,63	0	0,012873	0,016586	19,38	3,88	0

Tablo 3.2 üzerinden görülebilen, tüm YSA sonuçlarının ilgili küme sonuçları içerisinde olduğudur. Yani sonuç değerleri küme aralığında çıkmıştır. Bu konfigürasyonda 4 küme vardır ve bu veri kümesi ile 4 farklı kümeden sonuç üretilmiştir. Yani YSA sonucu için başvurulan üst ve alt limit değerleri tüm

kümelerden elemanlarla oluşturulmuştur. Tabloya bakıldığında, genelde COCOMO değerlerinin %30 ortalamada olduğu görülmektedir. Bununla beraber YSA değerinin daha yüksek olduğu gözlemlenmektedir. K-Means değerleri ise en başarılı sonuç olarak göze çarpmaktadır. Bazı satırlarda, 23 nolu proje örneğinde, K-Means MRE değerinin 0 olduğu görülmektedir. SWR metodu kullanıldığından, aynı proje satırının ilgili kümede birden çok geçtiği durumlarda bu sonuç ortaya çıkabilir. Yani küme içerisinde test kümesi satırı ile aynı satır mevcuttur. Burada önemli olan, ilgili satır için doğru kümenin ve doğru elemanın bulunmuş olmasıdır. Bu da bir bakıma sistemin K-Means kümeleri noktasında doğru çalıştığının bir göstergesidir. Bazı satırlar için üst ve alt limit değerleri aynı olmuştur, 58 nolu proje örneğinde olduğu gibi. Bu satırların 2 sebebi olabilir. YSA değeri kümede ya en küçük ya da en büyük değer olmuş, alt ve üst limit noktalarında tek yön bulunmuştur. Eğer küme içerisinde değilse kümenin üreteceği yine tek değer olacak ancak yönüne göre üst ya da alt limit değeri YSA sonucu olacaktır.

Tablo 3.2 incelendiğinde, alt ve üst limit değerlerinin MRE sonuçlarının yüksek olmadığı ve genelde %25 sınırından aşağıda olduğu görülebilir. Bu durumda üretilen değerlerin dengeli olduğu ifade edilebilir. 36 proje numaralı satır incelendiğinde, COCOMO MRE değeri %71, YSA MRE %105 olarak hesaplanmıştır. Hem COCOMO hem de YSA için oldukça kötü olan sonuçlar, üst ve alt limit değerleri açısından bakıldığında daha iyi olmuş, sırasıyla %30 ve %69 olarak hesaplanmıştır. Aynı zamanda ilgili proje için beklenen değer aralığının genişliği, yani limit değerleri arasındaki varyasyon da gözlemlenebilmektedir. Karar aşamasında bu değerlerin önemli olacağı açıktır. YSA MRE değerinin düşük çıktığı, örneğin 2 proje nolu satırda, K-Means değeri yine daha başarılı olmuş, ancak daha önemlisi, üst ve alt limit değerlerinde YSA MRE oranına yakın, fakat geniş değer aralığı üretilmiştir. 59 proje nolu satır incelenirse, YSA sonucunun en başarılı noktaya bu proje için geldiği, hatta K-Means değerinden daha iyi sonuç ürettiği görülebilir. Bununla beraber, üst limit noktasında yüksek bir sapma olduğu gözlemlenmektedir ve alt limit değerine çok daha yakın bir sonuç elde edilmiştir. Yani bir sonuç aralığı düşünüldüğünde bu proje için olası değerler beklenen en iyi sonuç düzlemine daha yakın olmuştur. 42 nolu satır için YSA değeri en kötü değeri üretmiş ve başarı oranı

çok düşük kalmıştır. Ancak bu değer için dahi üst ve alt limit değerleri makul denilebilecek düzeylere gelmiştir.



Şekil 3.1. SWR İle Nasa 60 Veri Kümesi İçin Sonuç Değerleri

Şekil 3.1, ve devam eden bölümlerde geçen benzer şekiller, sırasıyla 4 değeri göstermektedir: üst limit efor değeri, hedef- yani gerçekleşen efor değeri, YSA tahmin değeri ve alt limit efor değeri. Grafik okumayı kolaylaştırmak için hedef ve YSA sonuç değerleri, üst ve alt limit değerlerinin arasına alınmıştır. Grafik ile limit değerlerinin beklenen değerlere (hedef) daha yakın sonuçlar ürettiği daha iyi gözlemlenebilmektedir. Grafikte x ekseninde her bir test satırı ifade edilmiştir, proje numaraları yerine satır numaraları kullanılmıştır ancak tablo referansları ile aynı sıradadır. Y ekseninde ise, tahmin ve/veya sonuç değerleri bulunmaktadır. Bu değerlerin normalize edilmiş olduğu tekrar hatırlatılmalıdır.

3.1.2. COCOMO 81 veri kümesi üzerinde yapılan çalışma

Konfigürasyon değerleri:

- YSA Katmanları: 16 - 5 - 1
- YSA Öğrenme Oranı: 0,2
- YSA Döngü Sayısı: 40000
- K-Means Küme Sayısı: 4
- K-Means Küme Doluluk Eşit Oranı: 30,6000012159348
- K-Means Döngü Sayısı: 1000

- Eğitim Kümesi Eleman Sayısı: 200
- Test Kümesi Eleman Sayısı: 15

COCOMO 81 adlı veri kümesi ile yapılan program çalıştırmasında yukarıdaki değerler, bir önceki çalışmada gerektirdiği gibi, sistem parametreleri olarak belirlenmiştir. Bu veri kümesi için çok daha fazla YSA döngü sayısı ve eğitim kümesi eleman sayısı gerekmiştir. Bu gereksinim, veri yapısındaki dağılımından kaynaklanmıştır.

Tablolar üzerinde detaylı inceleneceği gibi, bu kümede sonuç değerleri arasında büyük bir varyasyon vardır ve YSA oldukça düşük bir performans göstermektedir ki beklenen bir durum olmadığı not edilmelidir. Özellikle COCOMO parametreleri göz önünde bulundurulduğunda, birbirine yakın nitelik değerlerinin birbirinden çok farklı sonuç değeri ürettiği durumlarda, bu çalışma için benimsenen YSA içinde yapılacak bir karşılık görülmemektedir. Yani YSA için ortaya çıkan sonuçların tutarlılık durumu sorgulanabilir seviyelerdedir. Örneğin 27 ve 60 numaralı proje örnek satırları incelendiğinde, hedef değerlerinin yakın olduğu görünmektedir. Fakat YSA sonuç değerleri ise önemli farklılıklar göstermiştir.

Tablo 3.3. SWR İle COCOMO 81 Veri Kümesi İçin Hedef ve Sonuçlar

ID	**Hedef**	**YSA Sonuç**	**K-Means Sonuç**
49	0,013173	0,025342	0,013173
48	0,111119	0,094022	0,111119
14	0,005889	0,00984	0,005626
27	0,007205	0,006984	0,007205
20	0,561176	0,561666	0,561176
26	0,033447	0,014629	0,033447
34	0,019668	0,007011	0,020282
37	0,003607	0,01508	0,003607
60	0,004485	0,008004	0,004309

Tablo 3.3. (Devam) SWR İle COCOMO 81 Veri Kümesi İçin Hedef ve Sonuçlar

ID	Hedef	YSA Sonuç	K-Means Sonuç
24	0,03924	0,050452	0,03924
42	0,003432	0,008278	0,003432
43	0,006767	0,007735	0,006767
51	0,010189	0,016135	0,010189
55	0,001062	0,004506	0,001062
53	0,000711	0,006605	0,000799

Tablo 3.3 incelendiğinde ilk göze çarpanın verilerin çok geniş bir aralıkta olduğudur. Yani küme içerisinde az sayıda eleman olmasına rağmen varyasyonu fazladır. Bu durumda üretilen tahmin değerlerinde başarı oranı fazla olmamaktadır. Özellikle YSA sonuç değerlerine bakıldığında, NASA 60 veri kümesi için geçerli olan durumun bu kümede geçerli olmadığı görülmektedir.

Tablo 3.4 MRE değerleri incelendiğinde durum daha iyi anlaşılmaktadır. Diğer veri kümesinin aksine bu kümede COCOMO MRE değerleri çok daha başarılıdır. Ancak burada da sınır değerleri olarak ifade edilecek çom büyük ve çok küçük rakamlar göze çarpmaktadır. Örneğin 42 nolu proje için MRE değeri çok yüksek çıkmıştır ancak 49 nolu projede oldukça düşük bir rakama ulaşılmıştır. Genelde COCOMO MRE için %45 ortalaması gözlemlenmektedir. Her ne kadar YSA için sonuçlar beklendiği seviyede olmamışsa da, bazı satırar için başarılı sonuçlar alınmıştır. Örneğin 27 ve 48 nolu projelerde COCOMO değerlerinden daha iyi sonuçlar üretilmiştir. Bunun yanında 53 proje nolu satırda tahmin değeri oldukça uzak kalmış ve kullanılabilirliğini kaybetmiştir. Bu satır incelendiğinde, ilgili projenin sonuç değerinin oldukça küçük olduğu gözlemlenmektedir. COCOMO modelindeki parametreler göz önüne alındığında, birbirine çok yakın değerlerle bu tür bir efor tahmin değeri üretmenin matematiksel bağıntı ile imkanı olmakla beraber, YSA üzerinde öğrenme aşamasında toplam hatada etkisinin daha az olacağı, genel eğilime daha yakın sonuçların üretileceği açıktır. YSA için geçerli olan problem K-Means

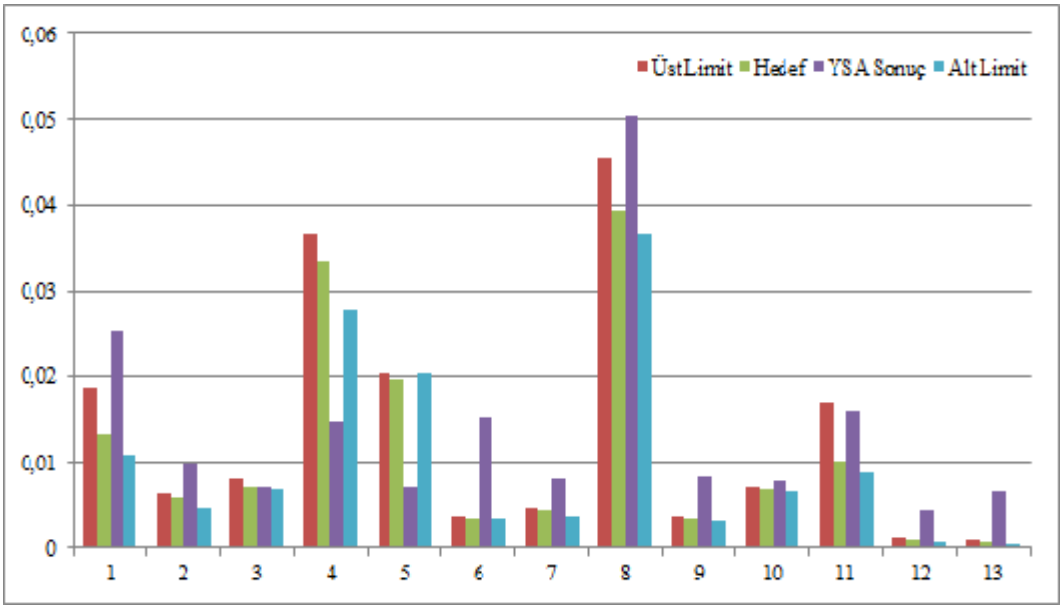
için geçerli değildir. Çoğunlukla hata oranı 0 olarak ölçülmüştür ancak bu değer, test kümesindeki elemanların büyük çoğunlukla eğitim kümesinde de yer almasından kaynaklanmaktadır. Yine de doğru satırın bulunmuş olması önemlidir. Sistem için daha anlamlı olan limit değerlerine bakıldığında, YSA değerlerinden çok daha iyi sonuçların alındığı gözlemlenmektedir. Bir önceki tablo için olduğu gibi, aynı sonuç değerleri olan satırlar vardır ve bu değerler YSA sonuç değerinin sınırları geçtiği durumları gösterir. Bu veri kümesi için de 4 kümeden sonuç üretilmiştir. Alt limit MRE değeri ortalama %25 değeri civarındadır ve kabul edilebilir sınırdadır. Üst limit değeri için bazı satırlarda yüksek değerler olsa da, üst limit değeri MRE ortalaması fazla değildir. 20 proje nolu satırda YSA en iyi sonucu üretmiştir. Limit değerleri ise aynı gerçekleşmiştir. 4 proje için YSA değerleri oldukça yüksek çıkmıştır. Bu satırlara bakıldığında, genelde beklenen değerin küme ortalamasından oldukça küçük olduğu projelere tekabül ettiği gözlemlenmektedir. Üst ve alt limit değerleri, YSA değeri için genelde yakın aralıklardadır yani YSA sonuçlarını genelde ortalar konumda çıkmıştır. Ancak 51 proje numaralı satırda da görülebileceği gibi, bazı durumlarda arada büyük farklar çıkmaktadır. Özellikle ortalamanın uç noktalarında olan değerlerin tahmininde bu durumlarla karşılaşılmaktadır.

Tablo 3.4. SWR İle COCOMO 81 Veri Kümesi İçin MRE Değerleri

ID	**COCOMO MRE**	**YSA MRE**	**K-MEANS MRE**	**Alt Limit**	**Üst Limit**	**Alt Limit MRE**	**Üst Limit MRE**	**En Yakın Küme**
49	9,29	92,37	0	0,010892	0,018615	17,32	41,31	2
48	73,32	15,39	0	0,061093	0,139906	45,02	25,91	3
14	31,3	67,09	4,47	0,004836	0,006416	17,88	8,94	0
27	19,42	3,07	0	0,006767	0,008083	6,09	12,18	0
20	10,57	0,09	0	0,561176	0,561176	0	0	3
26	26,12	56,26	0	0,027655	0,036607	17,32	9,45	2
34	26,08	64,35	3,12	0,020282	0,020282	3,12	3,12	0

Tablo 3.4. (Devam) SWR İle COCOMO 81 Veri Kümesi İçin MRE Değerleri

ID	COCOMO MRE	YSA MRE	K-MEANS MRE	Alt Limit	Üst Limit	Alt Limit MRE	Üst Limit MRE	En Yakın Küme
37	86,61	318,06	0	0,003432	0,00387	4,87	7,3	1
60	12,07	78,47	3,91	0,00387	0,004836	13,7	7,83	0
24	31,39	28,58	0	0,036607	0,045383	6,71	15,66	2
42	142,88	141,22	0	0,003256	0,00387	5,12	12,79	0
43	24,99	14,3	0	0,006679	0,007205	1,3	6,49	0
51	32,18	58,35	0	0,008785	0,017123	13,78	68,04	1
55	58,24	324,31	0	0,000799	0,001237	24,79	16,53	1
53	60,14	829,12	12,35	0,000535	0,001062	24,69	49,38	0



Şekil 3.2. SWR İle COCOMO 81 Veri Kümesi İçin Sonuç Değerleri

Şekil 3.2, sonuçların grafiksel gösterimini ifade etmektedir ve yaklaşım bir önceki grafik ile aynıdır. Ancak burada vurgulanması gereken bir nokta, proje satırları içerisindeki 20 ve 48 numaralı girişlerin listeden çıkarıldığıdır. Grafik gösteriminde

bu değerlerin kendisine en yakın değerden 5 kat büyük olması nedeniyle okunurluğu zorlaştırmasından dolayı çıkarılmıştır. Bu durumda 5 numaralı proje ve devam eden çizimler için referans olarak bir sonraki satırlar incelenmelidir. Bununla beraber genel eğilim mevcut satırlar üzerinden de gözlemlenebilmektedir. Yine okuma ve karşılaştırma kolaylığı için limit değerleri grafiklerin ilk ve son elemanlarıdır. Genel gözlem, üst ve alt limit değerlerinin beklenen değeri yukarı ve aşağı yönlü kapsadığı, ancak bazı durumlarda YSA sonucu için sınırlardan yüksek ya da düşük değerlerin olduğudur. Bu durumda limit değerlerinin daha sağlıklı olduğu ve hatta YSA tahmin değerinin yerine kullanılabileceği ifade edilebilir.

3.1.3. NASA 93 veri kümesi üzerinde yapılan çalışma

Konfigürasyon Değerleri:

- YSA Katmanları: 16 - 5 - 1
- YSA Öğrenme Oranı: 0,2
- YSA Döngü Sayısı: 5000
- K-Means Küme Sayısı: 4
- K-Means Küme Doluluk Eşit Oranı: 30,6666678852505
- K-Means Döngü Sayısı: 1000
- Eğitim Kümesi Eleman Sayısı: 90
- Test Kümesi Eleman Sayısı: 20

Bu küme içerisinde, diğer kümelerden daha fazla satır bulunmaktadır ve bu, örneklemenin daha farklı şekillerde yapılabilmesine olanak sağlamıştır. 20 adet test verisi üzerinde model denenmiş ve aşağıdaki sonuçlar elde edilmiştir. Diğer sunumlarda olduğu gibi, önce hedef ve sonuçlar verilmiş ve daha sonra diğer bilgiler listelenmiştir.

Tablo 3.5. SWR İle NASA 93 Veri Kümesi İçin Hedef ve Sonuçlar

ID	**Hedef**	**YSA Sonuç**	**K-Means Sonuç**
62	0,076512	0,082481	0,072123
10	0,001902	0,008814	0,001902
60	0,018726	0,029021	0,019701

Tablo 3.5. (Devam) SWR İle NASA 93 Veri Kümesi İçin Hedef ve Sonuçlar

ID	Hedef	YSA Sonuç	K-Means Sonuç
59	0,298881	0,268507	0,298881
46	0,050179	0,090428	0,050179
12	0,003365	0,009472	0,003365
35	0,004096	0,009742	0,003609
17	0,006291	0,013665	0,006291
42	0,165752	0,135482	0,165752
43	0,117597	0,136681	0,117597
29	0,006535	0,024588	0,006535
73	0,091142	0,111095	0,091142
24	0,024578	0,016936	0,028235
54	0,044084	0,031661	0,042864
55	0,142955	0,268086	0,142955
76	0,048838	0,048385	0,048838
46	0,050179	0,090428	0,050179
23	0,009948	0,01261	0,013313
65	0,022383	0,097573	0,022383
61	0,017263	0,012842	0,017263

Tablo 3.5 üzerinden de görüleceği gibi, sonuç değerleri yine geniş bir aralıktadır. Özellikle veri kümesinin 90 adet satırı olduğu, bununla beraber, yerine koyma yöntemi ile rastgele oluşturulan kümeler içerisinde her iki uçtan da eleman olacabileceği düşünüldüğünde bu değerler normal olarak ifade edilebilir. Gerçekte üretilen değerlerin anlamı ve yorumlanması Tablo 3.6 üzerinden yapılabilir. Bu

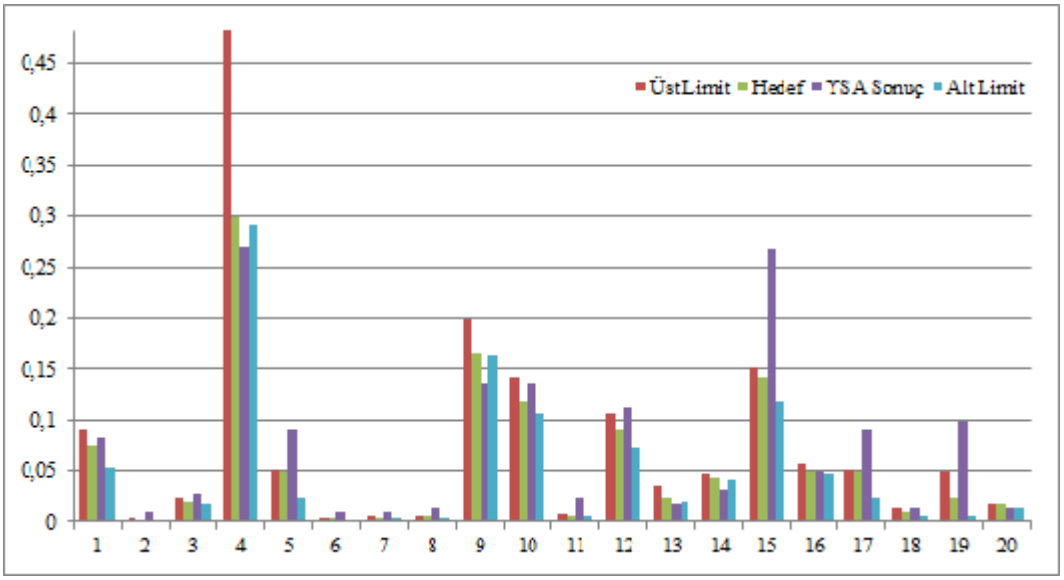
tabloda gözlemlenen, diğer veri kümelerine nazaran COCOMO hata oranının daha yüksek olduğudur. Hatta 65 nolu proje için bu rakam oldukça yüksek çıkmaktadır. Bunun yanında 46 ve 59 numaralı satırlarda da gözlemlenebildiği gibi, oldukça başarılı sonuçlar da alınabilmektedir. Benzer durum YSA için de geçerlidir. Başarı oranı, genelde belli bir ortalamada olsa da, özellikle başarısız sayılabilecek satırlarda büyük farklar ortaya çıkmıştır. Örnek olarak 10 proje numaralı satır incelendiğinde, küçük beklenen değere sahip projeler için tahmin değerlerinin büyük olarak gerçeklendiği gözlemlenebilmektedir. Diğer yüksek örneklerde de benzeri bir durum vardır. K-Means hata değerleri genelde 0 olarak gerçekleşmiştir. Önceki testlerde de ifade edildiği gibi, örnek küme içerisinde aynı satırların yer alması durumunda bu oran ortaya çıkabilmektedir. Pratik kullanım açısından artı değer sağlamasa da, limit değerlerini bulmada etkin olan, en uygun kümenin tercih edileceğine dair bir işaret olarak algılanabilir. Model için daha anlamlı olan üst ve alt limit değerleri incelendiğinde daha somut bir resim ortaya çıkmaktadır. Bu küme ile ortaya çıkan sonuçlarda limit değerleri arasındaki farklılık, diğer kümelerden daha fazla olmuştur. Bu da projenin ilgili küme içerisinde en yakın limit değerlerinde dahi büyük farklılıklar olduğunu gösterir. 59 numaralı satıra bakıldığında, YSA 10 olarak hesaplanmış, alt limit değeri de 2 civarında çıkmıştır. Ancak aynı satır için üst limit değerine bakıldığında 70 değeri gözlemlenmektedir. Bu duruma yapılacak sonuç, üretilen YSA tahmin değerinin olası en düşük değere daha yakın çıktığı, bunun yanında bir sonraki esnemede muhtemelen fazlaca bir efor süre/kaynağı gerektireceğidir. 46 proje numaralı satır incelendiğinde ise tam tersi bir durum gözlemlenmektedir. Bu satırda alt limit değeri daha uzak iken, üst limit değeri daha yakın çıkmıştır.

Tablo 3.6. SWR İle NASA 93 Veri Kümesi İçin MRE Değerleri

ID	**COCOMO MRE**	**YSA MRE**	**K-MEANS MRE**	**Alt Limit**	**Üst Limit**	**Alt Limit MRE**	**Üst Limit MRE**	**En Yakın Küme**
62	14,63	7,8	5,74	0,053105	0,091142	30,59	19,12	1
10	56,16	363,42	0	0,00117	0,003365	38,46	76,92	0
60	18,52	54,98	5,21	0,017263	0,025187	7,81	34,51	2

Tablo 3.6. (Devam) SWR İle NASA 93 Veri Kümesi İçin MRE Değerleri

ID	COCOMO MRE	YSA MRE	K-MEANS MRE	Alt Limit	Üst Limit	Alt Limit MRE	Üst Limit MRE	En Yakın Küme
59	2,68	10,16	0	0,291566	0,508351	2,45	70,08	3
46	4,03	80,21	0	0,022383	0,051642	55,39	2,92	1
12	22,23	181,49	0	0,001902	0,003609	43,48	7,25	0
35	23,13	137,83	11,9	0,003365	0,006291	17,86	53,57	0
17	24,65	117,23	0	0,003609	0,006535	42,64	3,88	0
42	16,7	18,26	0	0,163558	0,199632	1,32	20,44	1
43	39,13	16,23	0	0,106503	0,142955	9,43	21,56	1
29	18,47	276,28	0	0,006291	0,007754	3,73	18,66	2
73	43,76	21,89	0	0,072123	0,106503	20,87	16,85	1
24	13,06	31,09	14,88	0,019701	0,03555	19,84	44,64	0
54	21,98	28,18	2,77	0,041987	0,047741	4,76	8,3	2
55	4,71	87,53	0	0,117597	0,151123	17,74	5,71	1
76	5,31	0,93	0	0,047741	0,057494	2,25	17,72	2
46	4,03	80,21	0	0,022383	0,051642	55,39	2,92	1
23	30,87	26,75	33,82	0,006535	0,013605	34,31	36,76	0
65	199,07	335,92	0	0,006291	0,050179	71,9	124,18	1
61	45,37	25,61	0	0,013605	0,017872	21,19	3,53	0



Şekil 3.3. SWR İle NASA 93 Veri Kümesi İçin Sonuç Değerleri

Şekilde yine ilk ve son değerler limit değerleri olarak verilmiş ve bu şekilde okuma kolaylığı sağlanmıştır. Şekil 3.3 üzerinden daha iyi anlaşılacağı gibi, hedef değerine göre üretilen YSA tahmin değeri arasındaki yüksek farklar, üst ve alt limit değerleri ile daha aşağı çekilmiş ve daha yakın seviyelere getirilmiştir. Grafikteki büyük ve küçük değerler arasındaki farkın çokluğu, ortaya uç seviyeleri olan bir resim çıkarmış olsa da, hedeflenen çıkarımı destekleyebilecek ya da irdelenebilecek olanağı sunmaktadır.

3.1.4. SWR Kullanarak Yapılan Testlerin Genel Değerlendirmesi

Bu aşamada, eğitim ve test kümeleri, yerine koyma yöntemi ile üretilmiş, mevcut kümelerde rastgele üretilen satırlar için birden çok yer alma imkânı sağlanmıştır. Bu yöntem ile yapılan test çalışmalarında özellikle K-Means kümelerinin ürettiği değerlerin oldukça yüksek başarı oranına ulaştığı gözlemlenmektedir. Ancak bu değer, küme içerisinde aynı elemanların geçme ihtimali göz önünde bulundurulduğunda çok sağlıklı bir değer değildir. Aynı satırın tahmin olarak üretiliyor olması önemlidir ancak pratik kullanımda fazla bir katkısı mevcut değildir. YSA tahmin değerleri göz önüne alındığında, genelde çok küçük beklenen değere sahip satırlarda hata oranı daha fazla çıkmaktadır. Veri kümelerinin boyutu ve küme içerisindeki niteliklerin birbirine olan yakın değerlerine rağmen, sonuç değerlerinin aralığını çok yüksek olması, böyle bir durum ortaya çıkarmaktadır.

Yerine koyma yöntemi ile eğitim ve test kümeleri üretildiğinden, benzer satırların da gelebileceği göz önünde bulundurulursa, başarı oranı nispeten fazladır. İleri bölümde de görüleceği üzere, SWR kullanılmadığı duruma kısayla, aynı örneklem üzerinden tahmin değeri üretilmesi ve yorumlanması daha kolay olmaktadır. Önceki çalışmalarda veri kümelerinin küçük alt kümelere rastgele dağılım ile ayrıştırılıp, modelin bu verilerle çalıştırılıyor olması da yine bir bakıma yorum olanağının elde edilebilmesine yöneliktir[8]. Veri sayısının yeterli düzeyde olmadığı durumlarda bu yaklaşım faydalı olmaktadır.

Ortaya konulan modelin temel dayanağı, tek değer üretmek, yerine etkin kullanabilecek birden çok değerler, limit değerleri üretmektir. 3 veri kümesi ile yapılan çalışmadaki sonuçların bu durumu desteklediği ifade edilebilir. YSA tahmin değerindeki başarı oranı nispeten düşük olmasına rağmen, üst ve alt limit değerleri daha sağlıklı çıkmaktadır. Aynı zamanda bu değerlerle, tahmin edilen değer aralığının hangi bölüme düştüğü de anlaşılabilmektedir. Üst limit değerinin büyük, alt limit değerinin küçük olması, sıralamada bir fark noktasının, yani olası bir fazlalık değerinin çıkması ihtimalini ifade eder. Tersi durumda ise, tahmin değeri yüksek sınıra daha yakın çıkmış, en yakın düşük değerin ulaşılması daha zor olmuştur denilebilir. Elbette bulgular üzerinden tartışma noktasında, eldeki tablo ve grafikler kullanılmakta ve bu tablo/grafikler üzerindeki sayısal değerler yorumlanmaktadır. Neticede efor tahmini de bir sayısal değerdir. Bunun yanında bu sayısal değer, proje içerisinde çok farklı anlamlar da taşıyacaktır. Dolayısıyla, bir proje yöneticisi için yorumlar çok daha nitelikli olacaktır. Proje niteliklerinin bilindiği durumlarda ortaya çıkan tahmin değerlerinde olası esneklik sınırları ya da risk sınırları üzerinde yorumlar yapılabilir. Aynı durum yerine koyma metodu kullanılmadan yapılan testler için de geçerlidir. Yapılan yorumlar ve değerlendirmeler en temelde sayısal değerler üzerinden gitmek durumundadır. Her bir tablo ve sonuç bazında detaylı sayısal değer inceleme ve irdelemesi yapılmasa da, genel görünüm ifade edilmekte ve sonuç olarak ne tür yorumlar yapılabileceği konusunda görüşler bildirilmektedir.

3.2. SWR Olmaksızın Farklı Küme Elemanları Kullanılarak Yapılan Testler

Önceki kısımda, SWR yöntemi kullanılarak oluşturulan ve muhtemel aynı proje satırının eğitim ve/veya test kümesinde birden çok geçtiği kümeler üzerindeki testler

hakkında bilgi verilmiş, tablo ve grafiklerle desteklenmiştir. Bu aşamada ise, yerine koyma yöntemi olmadan, her bir proje satırının tüm kümelerde bir kere geçebildiği eğitim ve test kümeleri ile yapılan testlerin sonuçları verilmiştir. Benzer şekilde, genel tablo ve detay tablolar verilmiş, sonuç değerleri grafiksel olarak gösterilmiş ve tablo ve grafiklerle bulgular üzerinde yorumlar yapılmıştır. Yine aynı 3 veri kümesi kullanılmış ve her bir küme ilgili alt başlık ile irdelenmiştir.

3.2.1. NASA 60 veri kümesi üzerinde yapılan çalışma

Konfigürasyon Değerleri:

- YSA Katmanları: 16 - 5 - 1
- YSA Öğrenme Oranı: 0,2
- YSA Döngü Sayısı: 3000
- K-Means Küme Sayısı: 4
- K-Means Küme Doluluk Eşit Oranı: 32,4324337211815
- K-Means Döngü Sayısı: 1000
- Eğitim Kümesi Eleman Sayısı: 37
- Test Kümesi Eleman Sayısı: 10

Bu küme içerisinde aynı sonuç değerli satırlar ile sınır değerlerindeki satırlar çıkarılmıştır. Bu şekilde daha sağlıklı test yapılabildiği gözlemlenmiştir. Sonuçta eğitim kümesinde 37, test kümesinde 10 proje satırı yer almıştır. Kümelerdeki tüm elemanlar kullanılmış, rastgele seçilen elemanlar, sadece tek kümede kullanılmıştır.

Aşağıda, Tablo 3.7 üzerinde hedef değerleri ve YSA ile K-Means sonuç değerleri gösterilmektedir. Test kümesindeki hedef değerleri incelendiğinde, diğer veri kümelerindeki gibi, aralığın geniş olduğu gözlemlenmiştir. Benzer şekilde YSA ve K-Means tahmin değerlerinin de geniş bir aralıkta olduğu görülebilir. 3 veri kümesi için de geçerli olan, beklenen değer sonuçlarındaki aralığın oldukça geniş çıkmasıdır. Bu durumda özellikle YSA için öğrenme süreci daha farklı olmakta, sonuçlar nitelik değerlerinin toplam ağırlıklarına yakın olarak çok farklı çıkabilmektedir. Tablo 3.7 ve önceki diğer çalışmalardaki sonuçlar da bu durumu destekler niteliktedir.

Tablo 3.7. SWR Olmaksızın NASA 60 Veri Kümesi İçin Hedef ve Sonuçlar

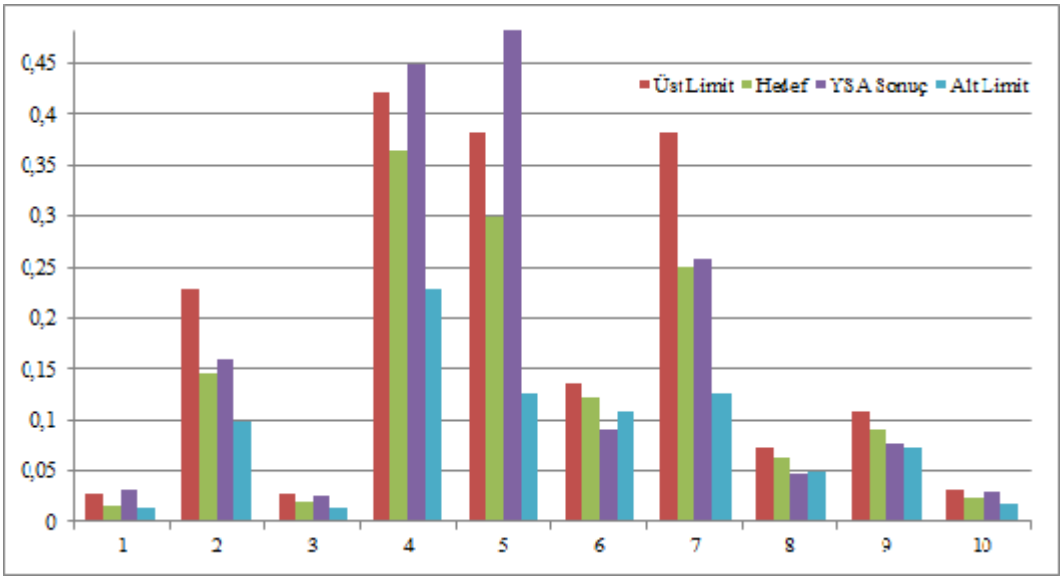
ID	Hedef	YSA Sonuç	K-Means Sonuç
10	0,015967	0,032538	0,016586
4	0,145934	0,159285	0,127367
45	0,019681	0,02697	0,016586
2	0,362854	0,449843	0,383587
56	0,29849	0,587858	0,229484
55	0,121178	0,0907	0,111895
19	0,249598	0,258382	0,229484
17	0,062384	0,047288	0,063931
7	0,090234	0,077097	0,083426
24	0,022775	0,031161	0,027974

Tablo 3.8 incelendiğinde ilk vurgulanması gereken nokta, 4 kümenin oluşturulması bekleniyorken, test sonuçlarının 3 kümeden gelmiş olmasıdır. Doluluk oranı da göz önünde bulundurulduğunda, bir küme içerisinde çok az sayıda satırın olacağı anlaşılacaktır. Özellikle eğitim kümesi içerisindeki satır sayısının az olduğu da düşünüldüğünde bu tür bir durumun çıkmış olması normal karşılanabilir. Veriler üzerinden inceleme yapıldığında COCOMO MRE değerlerinin ortalama %25 olarak gerçekleştiği görülmektedir. Bu ortalama değerini yükselten 45 ve 55 proje nolu satırlar olduğu gibi, düşük çıkmasını sağlayabilecek 4 ve 56 proje nolu satırlar da gözlemlenebilmektedir. Ancak ortalamaya yakın satırların çoğunlukta olduğu görülmektedir. YSA için MRE değeri %37 seviyesinde çıkarken, düşük değerli satırların çok daha fazla sayıda olduğu gözlemlenmektedir. Yani COCOMO için hata oranları ortalamaya yakın değerlerdeyken, YSA için düşük hata oranlı değerler daha fazla, sınırlarda olan değerler de sayısal olarak daha az ancak değer olarak daha büyüktür. Örneğin 10 ve 56 numaralı satırlara bakıldığında, ortalamayı oldukça yükselten başarısız satırların ortaya çıktığı söylenebilir. Tek başına K-Means

kullanıldığında yine daha başarılı sonuçlar alınmaktadır. K-Means MRE değeri ortalama %11 olarak gerçekleşmiştir. Limit değerlerine bakıldığında ise, sapmanın geniş olmasına rağmen, genel görünüm olarak YSA tahmin değerinden daha başarılı oldukları gözlemlenebilmektedir. Örneğin 10 proje numaralı satır için alt limit değeri çok daha makul çıkmıştır. Bununla beraber, bazı satırlarda YSA değerinin limit değerlerinden daha yüksek başarıya ulaştığı gözlemlenmektedir. Bu durum temelde normal karşılanabilir; neticede asıl tahmin YSA üzerinden yapılmaktadır. Ancak ilginç olan, üst ve alt limit değerleri arasındaki farktır ve bu fark, ilgili proje için efor tahmini konusunda yorum yapmada kritik durumun olduğu gösterebilir.

Tablo 3.8. SWR Olmaksızın NASA 60 Veri Kümesi MRE Değerleri

ID	**COCOMO MRE**	**YSA MRE**	**K-MEANS MRE**	**Alt Limit**	**Üst Limit**	**Alt Limit MRE**	**Üst Limit MRE**	**En Yakın Küme**
10	25,15	103,78	3,88	0,012873	0,027974	19,38	75,19	1
4	13,99	9,15	12,72	0,097661	0,229484	33,08	57,25	2
45	41,5	37,04	15,72	0,012873	0,027974	34,59	42,14	1
2	23,6	23,97	5,71	0,229484	0,42072	36,76	15,95	2
56	0,08	96,94	23,12	0,127367	0,383587	57,33	28,51	2
55	45,04	25,15	7,66	0,108801	0,136651	10,21	12,77	0
19	20,94	3,52	8,06	0,127367	0,383587	48,97	53,68	2
17	28,11	24,2	2,48	0,050006	0,071358	19,84	14,38	0
7	23,01	14,56	7,54	0,071358	0,108801	20,92	20,58	0
24	29,1	36,82	22,83	0,016586	0,032677	27,17	43,48	1



Şekil 3.4. SWR Olmaksızın NASA 60 Veri Kümesi Sonuç Değerleri

Şekil 3.4 ile görülen, genel olarak hedef değerinin üst ve alt limit aralığında çıktığıdır. Yani hedef, limit değerleri içinde çıkmaktadır. Bazı satırlar için YSA sonucu da bu aralıkta çıksa da, daha yüksek ya da daha düşük çıktığı durumlar da gözlemlenmektedir. Tek başına YSA değerinin kullanılmış olduğu durumlarda sağlıksız yorumların yapılabileceği anlaşılabilmekte, sınır değerlerinin katkısı gözlemlenebilmektedir.

3.2.2. COCOMO 81 veri kümesi üzerinde yapılan çalışma

Konfigürasyon Değerleri:

- YSA Katmanları: 16 - 10 - 1
- YSA Öğrenme Oranı: 0,4
- YSA Döngü Sayısı: 10000
- K-Means Küme Sayısı: 3
- K-Means Küme Doluluk Eşit Oranı: 35
- K-Means Döngü Sayısı: 1000
- Eğitim Kümesi Eleman Sayısı: 48
- Test Kümesi Eleman Sayısı: 12

Bu çalışmada yukarıda verilen değerlere göre program çalıştırılmış ve sonuçlar incelenmiştir. Burada küme sayısının 3 olarak verildiği dikkat edilmelidir. Bir önceki veri kümesinde 4 olarak verilen küme sayısının başarıya etkisinin olmadığı bu küme

içerisinde gözlemlenmiş ve 1 ya da 2 elemanlı küme içerisinden sonuç beklemenin sağlıklı olmayacağı düşünülmüştür. Bu durum sonu tabloları üzerinden daha iyi bir şekilde gözlemlenmektedir. Bu veri kümesindeki sonuçların dağılımına göre test sonuçları 2 küme içerisinden gelmektedir.

Tablo 3.9. SWR Olmaksızın COCOMO 81 Veri Kümesi Hedef ve Sonuçlar

ID	**Hedef**	**YSA Sonuç**	**K-Means Sonuç**
16	0,002993	0,030542	0,002817
34	0,019668	0,013198	0,020282
44	0,007118	0,012517	0,005626
33	0,05258	0,068325	0,046787
8	0,093829	0,037275	0,092776
58	0,010892	0,039734	0,008785
51	0,010189	0,008705	0,005626
26	0,033447	0,014182	0,036607
60	0,004485	0,018015	0,004836
13	0,006416	0,026279	0,006679
43	0,006767	0,015444	0,005626
52	0,003081	0,007905	0,003256

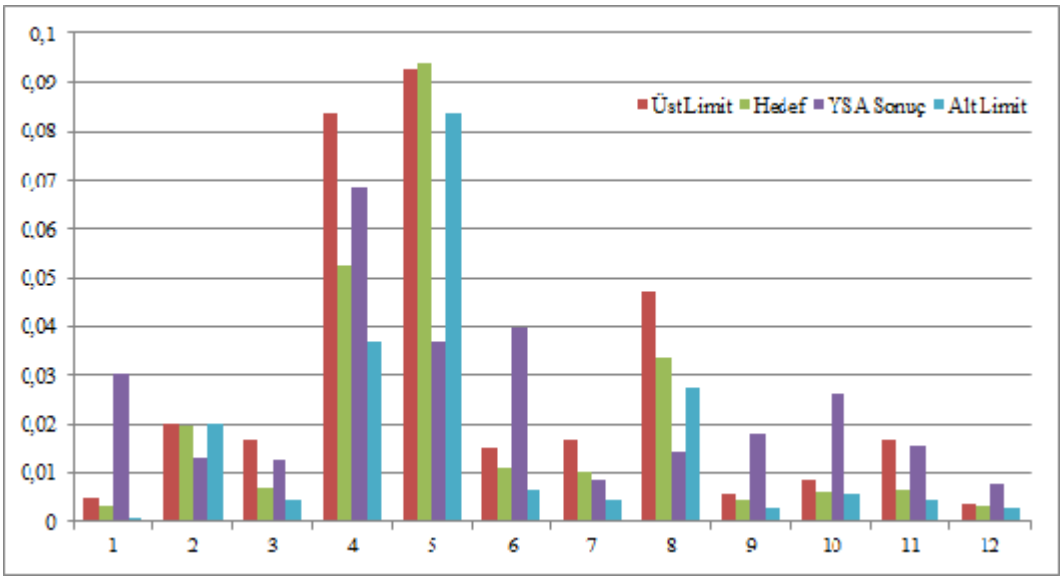
Tablo 3.9 incelendiğinde, test kümesinde küçük olarak nitelendirilebilecek satırların daha fazla olduğu görülmektedir. Bunun yanında çok büyük değerlere sahip satırlar da vardır. Bu durum tahmin değerlerine de yansımıştır. Benzer şekilde tahmin değerleri de geniş aralıkta çıkmıştır. Bu aralık için hata oranları Tablo 3.10 üzerinde gösterilmiştir.

Tablo 3.10. SWR Olmaksızın COCOMO 81 Veri Kümesi MRE Değerleri

ID	COCOMO MRE	YSA MRE	K-MEANS MRE	Alt Limit	Üst Limit	Alt Limit MRE	Üst Limit MRE	En Yakın Küme
16	45,6	920,52	5,87	0,000799	0,004836	73,31	61,58	2
34	26,08	32,9	3,12	0,020282	0,020282	3,12	3,12	0
44	52,73	75,86	20,96	0,004309	0,017123	39,46	140,57	0
33	8,98	29,95	11,02	0,036607	0,083561	30,38	58,92	2
8	41,41	60,27	1,12	0,083561	0,092776	10,94	1,12	2
58	11,69	264,82	19,34	0,006679	0,014929	38,68	37,07	2
51	32,18	14,57	44,79	0,004309	0,017123	57,71	68,04	0
26	26,12	57,6	9,45	0,027655	0,046787	17,32	39,88	2
60	12,07	301,7	7,83	0,002817	0,005889	37,18	31,31	2
13	43,79	309,61	4,1	0,005889	0,008785	8,21	36,94	2
43	24,99	128,24	16,86	0,004309	0,017123	36,32	153,05	0
52	2,16	156,62	5,7	0,002642	0,003432	14,25	11,4	0

Tablo 3.10 incelendiğinde, COCOMO tahmin değerlerinin ortalama olarak %27 olduğu ancak dağılımın, içerisinde çok büyük değerler olmasa da geniş bir aralıkta olduğu gözlemlenmiştir. Yine de, sonuçlar %40 civarı olarak bir grup ve %10 civarı olarak bir grupta toplanmış gibidir. Bu durumda ortalama değeri nispeten düşük gibi görülse de sonuçların varyasyonunu yansıtmadığı söylenebilir. YSA MRE değerlerine bakıldığında oldukça düşük bir performansın ortaya çıktığı görülmektedir. Burada ortalama çok yüksek çıkmış ve bu yüksek rakamı oluşturacak birden çok proje satırı gözlemlenmiştir. Ancak, kabul edilebilecek sınırlarda çıkan değerler de mevcuttur. Bu durumda COCOMO örneğinde olduğu gibi 2 grup sonuç oluşmuş, bir grup için başarı oranı kabul edilemez seviyede olmuşken, bir grup için nispeten iyi sonuçlar elde edilmiştir. Hata oranlarının çok yüksek çıktığı satırlar

incelendiğinde, genelde oldukça küçük beklenen değerleri içeren proje satırları olduğu görülebilmektedir. Bu durum, daha önce de ifade edildiği gibi, YSA eğitim ve dolayısıyla tahmin sürecine COCOMO parametrelerinin etkisi olarak açıklanabilir. Parametre değerleri birbirlerine çok yakın iken sonuçlar çok farklar göstermektedir. K-Means yine en başarılı sonucu vermiş ve %13 civarında bir değer ortaya çıkmıştır. Ancak diğer test sonuçlarında gözlemlenmeyen yüksek hata oranları bulunmaktadır. Alt ve üst limit değerlerine bakıldığında, özellikle YSA sonuçlarındaki başarı oranı göz önünde bulundurulursa, çok daha iyi bir grafik çizmiştir. Limit değerleri için başarı %30 ve %50 olarak gerçekleşmiştir. Üst limit üzerinde yüksek değerin çıkması, proje satırlarının küçük beklenen değere sahip olmasına karşın, tahmin değerlerinin daha büyük çıkmasındandır. Aradaki fark büyüdüğünden oran da artmıştır.



Şekil 3.5. SWR Olmaksızın COCOMO 81 Veri Kümesi Sonuç Değerleri

Şekil 3.5 ile değerler gösterilmiştir. Burada ortaya çıkan, mevcut veri kümesi içerisinde YSA değerlerinin başarısız görünümüne karşılık limit değerlerinin çok daha uygun sonuçlar ürettiğidir. Ancak, efor tahmin süreci bağlamında değerlendirildiğinde, bu veri kümesi için YSA sonuç üzerinden yorum ve çıkarım yapmanın zor olacağı açıktır. Limit değerlerinin kullanımının daha sağlıklı olacağı ifade edilebilir.

3.2.3. NASA 93 veri kümesi üzerinde yapılan çalışma

Konfigürasyon Değerleri:

- YSA Katmanları: 16 - 5 - 1
- YSA Öğrenme Oranı: 0,2
- YSA Döngü Sayısı: 7000
- K-Means Küme Sayısı: 4
- K-Means Küme Doluluk Eşit Oranı: 30,8571440832955
- K-Means Döngü Sayısı: 1000
- Eğitim Kümesi Eleman Sayısı: 70
- Test Kümesi Eleman Sayısı: 19

Bu veri kümesi ve yukarıda ifade edilen konfigürasyonlar ile modelin testleri tamamlanmaktadır. Bu şekilde 2 farklı yöntem ile 6 test yapılmış ve sonuçları paylaşılmıştır yine benzer şekilde tahmin değerleri ilgili sonuç grupları ile verilmiş, tablolar grafik ile desteklenmiştir.

Tablo 3.11. SWR Olmaksızın NASA 93 Veri Kümesi Hedef ve Sonuçlar

ID	**Hedef**	**YSA Sonuç**	**K-Means Sonuç**
26	0,00751	0,047593	0,006291
39	0,012874	0,025013	0,013313
38	0,004096	0,018514	0,004096
31	0,022383	0,055418	0,018726
87	0,057494	0,047994	0,057494
15	0,042864	0,038262	0,042864
54	0,044084	0,038739	0,042864
9	0,007754	0,024262	0,008973
73	0,091142	0,078435	0,09041

Tablo 3.11. (Devam) SWR Olmaksızın NASA 93 Veri Kümesi Hedef ve Sonuçlar

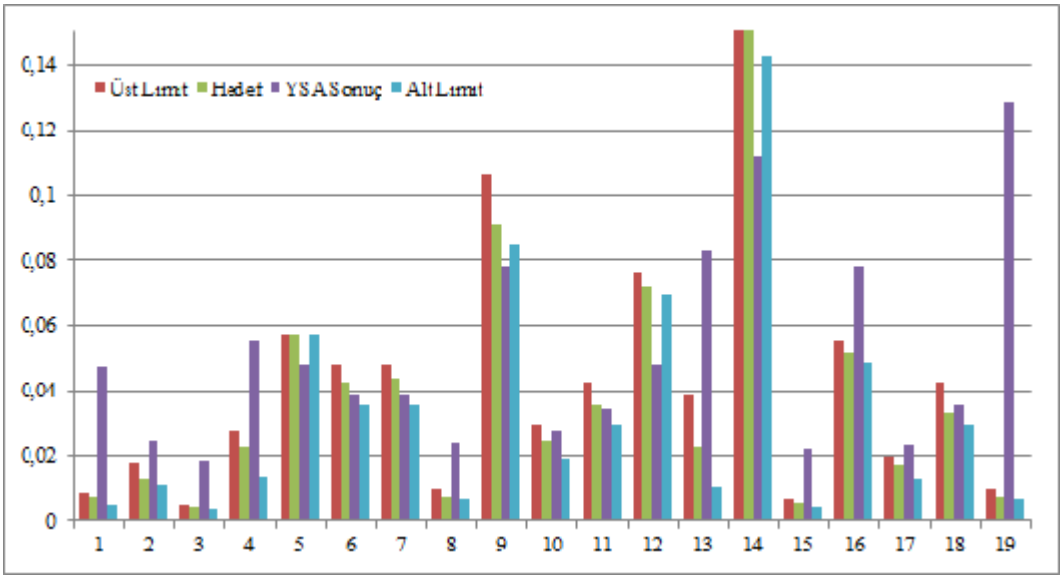
ID	Hedef	YSA Sonuç	K-Means Sonuç
24	0,024578	0,027673	0,028113
21	0,03555	0,034108	0,03555
72	0,072123	0,04784	0,072002
65	0,022383	0,082618	0,025187
42	0,165752	0,11184	0,145271
33	0,005072	0,021674	0,004828
81	0,051398	0,077936	0,053105
61	0,017263	0,023604	0,017872
56	0,032868	0,035622	0,03555
8	0,007754	0,128204	0,008973

Tablo 3.11 içerisinde, genelde bir akışa uygun değerlerin olduğu gözlemlenmektedir. Yani bir seri olarak düşünüldüğünde belli bir ivme ile ilerleyen, muhtemelen dengeli proje değerleri ve efor değerlerini içeren elemanlar mevcuttur. YSA ve K-Means sonuç değerleri de kısmen bu yapıya uygun şekilde ortaya çıkmıştır. Diğer eğitim kümelerinden daha sağlıklı bir yapısı vardır. Ancak yine de çok küçük ve büyük değerli satırlar da mevcuttur. Bu durumda, önceki kümelerde de gözlendiği gibi, dağılıma etki ettiği için yakın sonuç değerler haricinde sınır değerlerini içeren satırları da çıkmıştır. Bu satırlar MRE değerlerini, Tablo 3.12 üzerinden daha iyi şekilde gözlemlenebileceği gibi, etkilemektedir ve ölçüm içerisinde farklılıkların ortaya çıkmasına sebebiyet vermiştir. Bir başka ifadeyle, verilerin bir seri olarak düşünüldüğü durumda aykırı denilebilecek farklılıkta değerler, hem minimum hem de maksimum yönünde ortaya çıkmıştır. Neticede veri kümesini değerlendirirken ortak özellikler odaklı bakıldığı için bu bildirim ehemmiyet arz etmektedir.

Tablo 3.12. SWR Olmaksızın NASA 93 Veri Kümesi MRE Değerleri

ID	COCOMO MRE	YSA MRE	K-MEANS MRE	Alt Limit	Üst Limit	Alt Limit MRE	Üst Limit MRE	En Yakın Küme
26	11,17	533,75	16,23	0,004828	0,008973	35,71	19,48	1
39	25,04	94,29	3,41	0,011021	0,017872	14,39	38,83	1
38	15,66	351,96	0	0,003365	0,004828	17,86	17,86	1
31	17,47	147,59	16,34	0,013605	0,028113	39,22	25,6	0
87	41,14	16,52	0	0,057494	0,057494	0	0	0
15	16,17	10,74	0	0,03555	0,047741	17,06	11,38	0
54	21,98	12,12	2,77	0,03555	0,047741	19,36	8,3	0
9	54,19	212,91	15,72	0,006291	0,009948	18,87	28,3	1
73	43,76	13,94	0,8	0,08468	0,106503	7,09	16,85	3
24	13,06	12,59	14,38	0,018726	0,029698	23,81	20,83	0
21	3,16	4,06	0	0,029698	0,042864	16,46	20,58	0
72	27,33	33,67	0,17	0,069198	0,076512	4,06	6,09	3
65	199,07	269,11	12,53	0,010801	0,038476	51,74	71,9	3
42	16,7	32,53	12,36	0,142955	0,257431	13,75	55,31	2
33	27,39	327,36	4,81	0,004096	0,006291	19,23	24,04	1
81	65,94	51,63	3,32	0,048838	0,054812	4,98	6,64	0
61	45,37	36,73	3,53	0,013313	0,019701	22,88	14,12	1
56	0,02	8,38	8,16	0,029698	0,042864	9,64	30,42	0
8	36,85	1553,46	15,72	0,006291	0,009948	18,87	28,3	1

Tablo 3.12 içerisindeki COCOMO MRE değerleri verinin düzgün görünen dağılımına rağmen nispeten yüksek çıkmıştır. Bu değer ortalama olarak %35 olmuştur. Ancak bu değer içerisinde 9, 81 ve 65 proje nolu satırlardaki gibi çok yüksek MRE değerleri olan satırlar da vardır. Benzer şekilde 21, 26 ve 56 proje nolu, çok küçük MRE değerli satırlar da vardır. Aynı durum YSA MRE değerleri için çok daha dramatik şekilde gerçekleşmiştir. Bu test sonucunda YSA değerlerinin kullanılamaz durumda olduğu gözlemlenebilir. YSA için MRE ortalama değeri çok yüksek çıkmış ve bu değerin yüksek olmasını sağlayan birden çok üst değerler gözlemlenmiştir. Yine bu satırlar incelendiğinde genelde çok küçük beklenen değerlerin olduğu projeler bu tür bir yüksek MRE değerine sebep olmuştur. Ancak buna rağmen, neredeyse diğer tüm satırlarda COCOMO değeri aşılmış ve daha sağlıklı sonuçlar elde edilmiştir. K-Means söz konusu olduğunda, düzgün dağılımlı veri kümesinde en başarılı ve hatta diğer test sonuçları içerisinde de en başarılı sonuç elde edilmiş, ortalama %7 hata oranı bulunmuştur. Limit değerlerine bakıldığında da MRE değerleri özellikle YSA değerinden çok daha sağlıklı çıkmaktadır ve ortalama %18 ile %23 arasında gerçekleşmiştir. Bu noktada özellikle K-Means kümelerinin düzgün dağılmış elemanları içeriyor olmasının başarıyı artırdığı vurgulanmalıdır. Bu durum sonucun hangi kümeden geldiğini gösterir sütun ile de desteklenmektedir ve gözlemlenen, 4 kümeden de sonucun bulunduğudur.



Şekil 3.6. SWR Olmaksızın NASA 93 Veri Kümesi Sonuç Değerleri

YSA tahmin değerlerinde çok başarılı ve çok başarısız 2 grubun oluşması, Şekil 3.6 ile de görülebilecek şekilde farklı durumların ortaya çıkmasına sebebiyet vermiştir. Kimi durumlarda YSA değerleri tamamen çok farklı iken, limit değerleri beklenen değere yakın olmuş, kimi durumda ise daha dengeli bir grafik ortaya çıkmış ve YSA ile beklenen değer uyum göstermiştir. Bu durumda da yine limit değerleri ile tahmin yapmanın daha sağlıklı olabileceği varsayımı kabul edilebilir.

3.2.4. SWR Kullanılmayarak Yapılan Testlerin Genel Değerlendirmesi

Bir önceki alt bölümde yerine koyma metodu ile oluşturulan veri kümeleri ile yapılan çalışmalar incelenmiş ve genel olarak kabul edilebilir oranların çıktığı gözlemlenmiştir. Ancak, tekrarlamayan elemanlarla alt kümeler oluşturulduğunda çok daha farklı sonuçlar gözlemlenmiştir. Bu bölümde ortaya çıkan durum, COCOMO dâhil, tahmin değerlerinin başarı oranının düşüklüğüdür. Ancak K-Means için durum geçerli olmamış ve her durumda en başarılı sonuçları vermiştir. Ancak, K-Means ile gerçek bir tahmin yapılmadığından, YSA üzerinden gidilerek hesaplanan limit değerleri daha fazla odak olmuştur. Bu durumda da, K-Means kümelerinin başarısı limit değerlerine yansımış, birçok durumda limit değerleri ki olası daha düşük ve daha yüksek değerleri ifade ediyor olmalarına rağmen, daha iyi sonuçlar elde edilmiştir. Yani YSA sonucuna göre yapılacak yorumları desteklemek için kullanılacak üst ve alt limit değerleri bazı durumlarda daha az anlamlı olmuş ve sadece limit değerleri ile olası aralığın belirlenmesi daha sağlıklı bir yaklaşım olmuştur.

Bir önceki değerlendirme bölümünde de ifade edildiği gibi, efor tahmini bir sayısal değer üretme işlemidir ve dolayısıyla ortaya konan modeller, öncelikle sayısal olarak geçerlenir. Bu durumda model içerisindeki standart YSA yeterli olmamış ama esneklik olarak sunulan limit değerleri çok daha iyi sonuçlar vermiştir. Yani basitçe tek bir yöntem ve değer kullanılmış olsaydı daha düşük bir başarı elde edilecekken, yardımcı değerlerle hem daha anlamlı sonuçlar elde edilmiş, hem de daha başarılı bir değerlendirme imkânı ortaya çıkmıştır.

4. SONUÇLAR VE ÖNERİLER

Yazılım proje yönetiminde, yönetim noktasında en temel işler, gerekli kaynakların kendi birim değerleri içerisinde uygun ve etkin olarak atanmasıdır. Yani planlama yapılarak ilgili projenin gerçekleştiriminde ihtiyaç duyulacak kaynağın belirlenmesidir. Zaman, bu kaynakların en temel noktalarından biridir ve proje içerisindeki tüm temel alanlar zamana bağlıdır. Bu durumda ihtiyaç olan zamanı tahmin etme, belirleme de ehemmiyet arz etmektedir.

Proje yöneticileri tahminleri genelde önceki proje bilgilerine ve tecrübelerine göre ortaya koymaktadır. Ancak destekleyici olma mahiyetinde farklı yöntemlere de başvurabilmektedirler. COCOMO bunlardan biridir ve uygulama noktasında bir formüle dayandığından tercih edilebilir. Benzer şekilde veri madenciliği tabanlı, YSA modelleri de gözlemlenmektedir. Ancak yine de bu modellerin değerleri hem istenilen seviyede –genel olarak- değildir, hem de sadece bir sonuç değeri ürettiği için yorum noktasında eksiktir. Bir başka ifadeyle, ortaya koydukları başarı yüzdesine ek olarak dayanak değerleri üretmez. Bu çalışmada, yazılım efor tahmini üretilirken daha etkin kullanılabilecek destek değerleri sağlanması üzerinde durulmuştur. Model kısaca, efor tahmini değeri üretmekle beraber, olası üst ve alt limit değerlerini de üretmeyi hedeflemiştir.

Model ortaya konduktan ve yapısı izah edildikten sonra çeşitli varyasyonlarla test edilmiş ve bu sonuçlar verilmiştir. Test sonuçları incelendiğinde genel görünümün, tek başına efor tahmin değerlerinin çoğu durumda daha az başarılı olduğudur. Buradaki başarı faktörü, mecburen, sayısal MRE değerleri üzerinden yorumlanmıştır. Ancak, başarı durumunu asıl etkileyen, tahmin değerlerindeki sapmalar ve farklılıkları irdeleyebilecek bir başka karşılaştırma değer(ler)inin olmayışıdır. Bu noktada ortaya konan model içerisindeki değerler devreye girmekte ve çok daha olumlu sonuçların üretildiği gözlemlenmektedir.

Tüm veri kümeleri için K-Means yönteminin tek başına kullanımı durumunda daha başarılı sonuçlar alındığı gözlemlenmektedir. Ancak, çoğu durumda veri kümeleri

içerisindeki eleman sayısının az olduğu ve eğitim kümelerinde tekrarlı proje satırlarının yer alması gerektiği göz önünde bulundurulduğunda bu başarı değeri bir anlam ifade etmeyebilir. Bununla beraber, yerine koyma metodu kullanılmadan oluşturulan kümelerdeki başarı oranı ise vurgulanmalıdır. Bu kümelerde her durumda en başarılı olan K-Means tahminleridir. Ancak, tezin önermesinin tek bir değer üretmek olmadığı düşüldüğünde ve yorumların sayısal değerlendirmelerden sonra, proje için varyasyonlarında düşünülmesi gerektiği tekrar ifade edildiğinde, yine limit değerleri bir anlam ifade etmektedir.

Tüm sonuçlar için limit değerleri çok önemli bilgiler vermiştir. YSA değerinin başarılı olduğu durumlarda olası sapmaları ifade etmiştir. Ancak YSA değerlerinin başarısız olduğu durumlarda ise, olası sonuç aralığını göstermiştir. İlk bakışta test kümelerinden bu şekilde bir çıktı beklenmemiş olsa da, program varyasyonları artırıldığında ortaya çıkan senaryolarda bu tür getirilerin olduğu gözlemlenmiş ve modelin iyi bir netice verdiği görülmüştür.

Sonuç olarak, efor tahmini sürecinde destek ve sınır değerlerinin üretilmesi önemlidir ve karar aşamasında fayda sağlayacağı düşünülmektedir. Bu durumda ortaya konan modelin, bu tür yaklaşımlar için bir örnek olması umulmaktadır. Elbette, çalışma, nispeten az sayıdaki veri kümesi üzerinde gerçekleştirilmiştir. Bu durumda modelin hassasiyeti ölçülememiştir. Ancak, fikir olarak ortaya konulan ve beklenti olarak ifade edilenin genel anlamda gerçekleşme eğiliminde olduğu gözlemlenmiştir. Dolayısıyla, yaklaşımın doğruluğu da bir bakıma geçerlenmiş olmaktadır. Elbette ortaya konan model, şu aşamada bir önerme noktasındadır ve pratik etkisi henüz bilinmemektedir. Böyle bir değerlendirmenin yapılabilmesi ise, ileri aşamada, modeli gerçekleyen programın sektörde kullanımı ve proje yöneticilerinin geri bildirimleri ile gerçekleşebilir. Bu ifade en temel öneri olarak sunulabilir. Bu çalışma içerisinde dahi pratik olarak kullanılabilecek bir araç geliştirilmiş ve testler bu araç ile yapılmıştır. Bu tip bir yapının pratik olarak kullanıma sunulması, modelin değerlendirilmesine çok daha fazla katkı sağlayabilir.

Veri madenciliği ya da içerisindeki çeşitli yöntemler, temelde, büyük veri kümeleri ile çalışmaktadır ve başarıları, büyük kümelerde anlam ifade etmektedir. Bu çalışma kapsamında toplamda 200 civarında olan proje satırı ile testler

gerçekleştirilebilmiştir. Bu nedenle modelin veri kümesinden etkilenip etkilenmediği net olarak anlaşılamamaktadır. Yerine koyma yöntemi kullanılmadığında başarı oranının düşmüş olması da küçük veri kümelerinde gerçekte çok net bir yorumun yapılmasına olanak sağlamamaktadır çünkü küçük veri kümesi içerisinde neredeyse kendi sınıfı içinde birkaç veri satırı bulunmaktadır. Bu nedenle öneri noktasında bir diğer önemli aşama, veri kümelerinin sayısının artırılmasıdır. Çok daha fazla veri ile yapılacak testlerin daha iyi bir resim ortaya koyacağı beklenmektedir.

Tez kapsamında ve model içerisinde vurgulanması gereken diğer noktalar, özellikle model yapısı içindeki elemanlarla ilgili sorunlarda ortaya konan çözümlerdir. Rastgele değerlerin hassasiyetinin artırılması, normalizasyon işleminde sonuç değerlerinin daha iyi irdelenebilmesi için virgülden sonraki basamak sayısının olabildiğince fazla tutulması, K-Means kümeleri için doluluk oranı kavramının geliştirilmesi, modelin geçerlemesi noktasında birçok sonuç değerinin (COCOMO dâhil) üretilmesi, tez içerisinde yapılan birçok farklı çalışma ve incelemeyi ifade etmektedir. Aynı zamanda, daha önce de ifade edildiği gibi, sürecin bizatihi sayısal değer üretme amacı güdüyor olması, önerilen modele de kaynaklık etmiştir ancak değerlere daha fazla anlam yüklenebilmesi için mümkün olan tüm sonuçlar gösterilmiştir. Gösterimlerde ve dolayısıyla sonuç değerleri üzerinde tartışma ve çıkarımlarda sayısal değerlerin ön plana getirilmesi, aslında modelin incelenmesi ve sonuç itibariyle efor tahmin değerinin makul bir seviyede ve mümkün olan limit değerlerine göre yorumlanmasına olanak sağlamıştır. Yine de limit değerleri söz konusu olduğunda bir gelecek çalışma alanı olduğu açıktır: limit olarak hangi ölçü belirlenecektir? Bu çalışmada, sonuç değerine en yakın üst ve alt değerler alınmıştır. Ancak geliştirilebilecek daha dinamik bir model ile, proje riskine göre daha esnek veya daha az esnek bir yapı sunulabilir. Söz konusu yaklaşım, ileri aşamada üzerinde çalışmayı hak eden bir ifadedir.

Bu çalışma ile yazılım efor tahmini literatürüne farklı bir açıdan katkı sağlanmış, tahmin değerlerinin sayısı, belli bir modele göre artırılarak daha etkin ve üzerinde yorum yapılabilecek seviyeye getirilmiştir. Test sonuçları, beklentiler ışığında, limit değerlerinin başarılı olabileceğini göstermiş ve ileri aşamada benzer çalışmalara bir destek sağlayabilecek genel bir şablon ortaya koymuştur.

KAYNAKLAR

[1] Project Management Institute (PMI), *PMBOK - A guide to the project management body of knowledge*, 4th Ed., Project Management Institute, Pennsylvania, 2008.

[2] Hughes B., Cotterell M., *Software project management*, 4th Ed., McGraw Hill, Glasgow, 2004.

[3] http://en.wikipedia.org/wiki/Software_development_effort_estimation (Ziyaret tarihi: 29 Ağustos 2013).

[4] Boehm B., *Software engineering economics*, Prentice Hall, New Jersey, 1994.

[5] Reddy Ch. S., Raju KVSVN., A concise neural network model for estimating software effort, *International Journal of Recent Trends in Engineering*, 2009, **1**, 188-193.

[6] Kaushik A., Chauhan A., Mittal D., Gupta S., COCOMO estimates using neural networks, *(IJISA)*, 2012, **9**, 22-28.

[7] Idri A., Khoshgoftaar T. M., Abran A., Can neural networks be easily interpreted in software cost estimation?, *Proceedings of the IEEE International Conference on Fuzzy Systems*, Honolulu, HI, 12-17 May 2002.

[8] Kultur Y., Turhan B., Bener A., Ensemble of neural networks with associative memory (ENNA) for estimating software development costs, *Journal Knowledge-Based Systems*, 2009, **6**, 395-402.

[9] Stellman A., Greene J., *Applied software project management*, O'Reilly, Sebastopol, 2009.

[10] Vliet H., *Software engineering principles and practice*, 3rd Ed., Wiley press, Glasgow, 2007.

[11] Royce W., Managing the development of large software systems, *Proceedings of IEEE WESCON 26*, Los Angeles, USA, August 1970.

[12] Boehm B., A Spiral Model of Software Development and Enhancement, University of Maryland, http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/spiral.pdf (Ziyaret tarihi: 12 Temmuz 2013).

[13] Pressman R., *Software engineering: a practitioner's approach*, 7th Ed., McGraw Hill, Boston, 2010.

[14] Royce W., *Software project management: a unified framework*, Addison-Wesley, Reading, 1998.

[15] Brad Clark, COCOMO 81, University of Southern California, http://csse.usc.edu/csse/research/COCOMOII/cocomo81.htm (Ziyaret tarihi: 29 Haziran 2013).

[16] http://en.wikipedia.org/wiki/SEER-SEM (Ziyaret tarihi: 30 Ağustos 2013).

[17] Putnam L. H., A general empirical solution to the macro software sizing and estimating problem, *IEEE Trans. Software Eng*., 1978, **4**, 345-361.

[18] Albrecht, A. J., Gaffney, J. E., Software function, source lines of code, and development effort prediction, *IEEE Transactions on Software Engineering*, 1983, **6**, 639-648.

[19] Aljahdali S., Sheta A. F., Software effort estimation by tuning COCOMO model parameters using differential evolution, *AICCSA*, Hammamet, Tunisia, 16-19 May 2010.

[20] Park R., Goethert W., Webb J.. *Software cost and schedule estimating: a process improvement initiative*, Technical report, 1994.

[21] Fayyad U.M., Piatetsky-Shapiro G., Smyth P., From data mining to knowledge discovery: an overview, http://www.kdnuggets.com/gpspubs/aimag-kdd-overview-1996-Fayyad.pdf (Ziyaret tarihi: 29 Haziran 2013).

[22] Han J., Kamber M., *Data mining concepts and techniques*, 2nd Ed., Morgan Kauffman Publishers, San Fransisco, 2006.

[23] Carlos Gershenson, Artificial neural networks for beginners, http://arxiv.org/ftp/cs/papers/0308/0308031.pdf (Ziyaret tarihi: 22 Ağustos 2013).

[24] http://www.princeton.edu/~achaney/tmve/wiki100k/docs/Sigmoid_function.html (Ziyaret tarihi: 29 Ağustos 2013).

[25] http://en.wikipedia.org/wiki/K-means_clustering (Ziyaret tarihi: 29 Ağustos 2013).

[26] http://planetmath.org/euclideandistance (Ziyaret tarihi: 12 Nisan 2013).

[27] http://en.wikipedia.org/wiki/Minkowski_distance (Ziyaret tarihi: 29 Ağustos 2013).

[28] Tajunisha N., Saravanan V., Performance analysis of k-means with different initialization methods for high dimensional data, *International Journal of Artificial Intelligence & Applications*, 2010, **4**, 44-52.

[29] Marcus A., Menzies T., Software is data too, *Proceedings of the FSE/SDP workshop on Future of software engineering research (FoSER)*, Santa Fe, New Mexico, USA, 7-11 November 2010.

[30] http://2013.msrconf.org/ (Ziyaret tarihi: 13 Ağustos 2013).

[31] Hassan A. E., Mining Software repositories to assist developers and support managers, PhD Thesis, Waterloo, Ontario, Canada, 2004.

[32] Kim S., Zimmermann T., Kim M., Hassan A., Mockus A., Girba T., Pinzger M., Whitehead E.J., Zeller A., TA-RE: an exchange language for mining software repositories, *Proceedings of the 2006 international workshop on Mining software repositories* , Shanghai, China, 22-23 May 2006.

[33] http://ssbse.org/2013/ (Ziyaret tarihi: 13 Ağustos 2013).

[34] Balogh G., Végh A. Z., Beszédes A., Prediction of software development modification effort enhanced by genetic algorithm, *4th Symposium on Search Based Software Engineering*, Riva del Garda, Trento, Italy, 28-30 September 2012.

[35] Mitchel M., Genetic algorithms: an overview, *An introduction to genetic algorithms*, 5th Ed., MIT Press, Massachusetts, 2-24, 1996.

[36] http://promisedata.org/2013/ (Ziyaret tarihi: 13 Ağustos 2013).

[37] Minku L., Yao X., A principled evaluation of ensembles of learning machines for software effort estimation, *Proceedings of the 7th International Conference on Predictive Models in Software Engineering*, Banff, Canada, 20-21 September 2011.

[38] Minku L, Yao X., Can cross-company data improve performance in software effort estimation, *PROMISE'12*, Lund, Sweden, 21-22 September 2012.

[39] Singh J., Sahoo B., Software effort estimation with different artificial neural network, *International Journal of Computer Applications*, 2011, **1**, 13-17.

[40] Jorgensen M., Shepperd M., A systematic review of software development cost estimation studies, *IEEE transactions on software Engineering*, 2007, **1**, 33-53.

[41] http://promise.site.uottawa.ca/SERepository/datasets-page.html (Ziyaret tarihi: 29 Ağustos 2013).

[42] Briand L. C., Wieczorek I., Emam K. E., Surmann D., An assessment and comparison of common software cost estimation modeling techniques, *ISERN-1998-27*, 313-323, 1999.

[43] http://promisedata.googlecode.com/svn-history/r516/trunk/effort/cocomo-sdr/cocomo-sdr.arff (Ziyaret tarihi: 29 Ağustos 2013).

[44] http://msdn.microsoft.com/library/vstudio/hh156542 (Ziyaret tarihi: 29 Ağustos 2013).

[45] McMillan M., Basic sorting algorithms, *Data structures and algorithms using C#*, Cambridge Univesity Press, New York, 2007, 42-55, 2007.

[46] http://msdn.microsoft.com/tr-tr/library/vstudio/ms229592 (Ziyaret tarihi: 29 Ağustos 2013).

[47] Klair A. S., Kaur R. P., Software effort estimation using k-nearest neighbour (kNN), *International Conference on Artificial Intelligence and Image Processing (ICAIIP'2012)*, Dubai, UAE, 6-7 October 2012.

[48] Attarzadeh I., Ow S. H., Software development effort estimation based on a new fuzzy logic model, *International Journal of Computer Theory and Engineering*, 2009, **4**, 473-476.

[49] Malathi S., Sridhar S., A classical fuzzy approach for software effort estimation on machine learning technique, *IJCSI*, 2011, **1**, 249-253.

[50] Catal C., Aktas M. S., A composite project effort estimation approach in an enterprise software development project, *The 23rd International Conference on Software Engineering and Knowledge Engineering (SEKE)*, Miami Beach, USA, 7-9 July 2011.

[51] Dave V.S., Dutta K., Application of feed-forward neural network in estimation of software effort, *International Symposium on Devices MEMS, Intelligent Systems & Communication (ISDMISC) 2011*, Gangtok, Sikkim, India, 12-14 April 2011.

[52] Kocaguneli E., Menzies T., Keung J., On the value of ensemble effort estimation, *IEEE Transactions on Software Engineering*, 20120, **6**, 1403-1416.

[53] Srinivasan K., Machine learning approaches to estimating software development effort, *IEEE Transactions on Software Engineering*, 1995, **2**, 126-137.

[54] Finnie G. R., Wittig G. E., AI tools for software development effort estimation, *SEEP '96 Proceedings of the 1996 International Conference on Software Engineering: Education and Practice (SE:EP '96)*, Berlin, Germany, 25-30 March 1996.

[55] Layman L., Nagappan N., Guckenheimer S., Beehler J., Begel A., Mining software effort data preliminary analysis of visual studio team system data, *Proceedings of the 2008 international working conference on Mining software repositories*, Leipzig, Germany, 10 - 18 May 2008.

[56] Özkaya A., Ungan E., Demirors O., Yazılım sektöründe efor verisi toplamanın zorlukları ve yaygınlığı, *V. Ulusal Yazilim Muhendisligi Sempozyumu*, Ankara, Türkiye, 26-28 Eylül 2011.

[57] Attarzadeh I., Ow S. H., A novel algorithmic cost estimation model based on soft computing technique, *Journal of Computer Science*, 2010, **6**, 117-125.

[58] Vahid Khatibi B. V., Jawawi D., An analytical model for software development effort estimation based on investigation of project characteristics, *Malaysian Software Engineering Interest Group*, The 3rd Software Engineering Postgraduate Workshop, 34-40.

[59] Balsera J. V., Montequin V. R., Fernandez F. O., González-Fanjul C. A., Data mining applied to the improvement of project management, Editor: Associate Prof. Adem Karahoca, *Data Mining Applications in Engineering and Medicine*, 2012.

[60] Humayun M., Gang C., Estimating effort in global software development projects using machine learning techniques, *International Journal of Information and Education Technology*, 2012, **3**, 208-211.

EKLER

EK-A GELİŞTİRİLEN PROGRAM İÇİN YAZILAN KODLAR

KOD BÖLÜMLERİ	
ReadAllDataSet	Girdi dosyasının okunduğu metot
FindRandomKeyIndex	Rastgele satır indeksi hesaplar
FindRandomKeyIndexWithBootstrap	Yerine koyma yöntemiyle rastgele satır indeksi hesaplar
MakeTrainSetWithKeyIdentifiers	Eğitim kümesini oluşturur
MakeTrainSetWithBootstrapWithKeyIdentifiers	Yerine koyma yöntemiyle eğitim kümesini oluşturur
MakeTestSetWithKeyIdentifiers	Test kümesini oluşturur
MakeTestSetWithBootstrapWithKeyIdentifiers	Yerine koyma yöntemiyle test kümesini oluşturur
InitSets	K-Means kümelere başlangıç değerlerini atar
BuildKMeansSetsEntity	K-Means kümelerini oluşturur
FindNearestProperSet	K-Means küme oluşturmada doluluk oranına göre en uygun kümeyi bulur
FindNearestOutputRangeWithANNOutForEntity	YSA sonucuna göre K-Means üst-alt limit değerlerini hesaplar
FindNearestSet	Test satırı için en yakın kümeyi bulur
InitNetwork	YSA başlangıç değerlerini atar
TrainNetworkWithEntity	YSA' yı eğitir
PassForwardWithEntity	YSA eğitim aşamasında ileri besleme kısmını gerçekleştirir
BackPropagateExt	YSA eğitim aşamasında geri yayılım kısmını gerçekleştirir
TestNetwork	Bir veri satırı ile YSA üzerinden test yapılır
FindCocomoEstimateForEntity	COCOMO formülüne göre tahmin değerini hesaplar
FindMRE	Karşılaştırma değerlerinde

	kullanılan hata oranını hesaplar
RunANN	YSA ile ilgili tüm çalışma mantığını kapsar; ayrı bir thread olarak çalışır
RunKMeans	K-Means ile ilgili tüm çalışma mantığını kapsar; ayrı bir thread olarak çalışır
buttonReadDataSet_Click	Programda Çalıştır denildiğinde arka planda gerçekleşen tüm işlemleri kapsar

ReadAllDataSet

```
        public void ReadAllDataSet(string path)
        {
            _originalInputs.Clear();
            _originalInputsKeys.Clear();
            StreamReader sr = new StreamReader(path);
            while (sr.EndOfStream == false)
            {
                string[] str = sr.ReadLine().Split(';');
                List<double> d = new List<double>();
                for (int i = 1; i < str.Length; i++)
                {
                    d.Add(Convert.ToDouble(str[i]));
                }
                int key = Convert.ToInt32(str[0]);
                _originalInputs.Add(key, d);
            }
            sr.Close();
        }
```

FindRandomKeyIndex

```
        private int FindRandomKeyIndex()
        {
            int keyIndex = 0;
            if (_originalInputsKeys.Count > 1)
                keyIndex =
RandomNumberGenerator.GetRandomNumber(0,
_originalInputsKeys.Count);
            return keyIndex;
        }
```

FindRandomKeyIndexWithBootstrap

```
        private int FindRandomKeyIndexWithBootstrap()
        {
            int keyIndex = 0;
            if (_originalInputs.Count > 1)
```

```
                keyIndex =
RandomNumberGenerator.GetRandomNumber(0, _originalInputs.Count);
            return keyIndex;
        }
```

MakeTrainSetWithKeyIdentifiers

```
        public void MakeTrainSetWithKeyIdentifiers(int
totalCount)
        {
            _trainSetWithKeyIdentifiers.Clear();
            _trainSetEntity.Inputs = null;
            while (totalCount > 0)
            {
                int keyIndex = FindRandomKeyIndex();
                int key = _originalInputsKeys[keyIndex];
                _trainSetWithKeyIdentifiers.Add(new
SetKeyIdentifier(totalCount, key), _originalInputs[key]);
                _originalInputsKeys.RemoveAt(keyIndex);
                totalCount--;
            }
            _trainSetEntity.Inputs = _trainSetWithKeyIdentifiers;
        }
```

MakeTrainSetWithBootstrapWithKeyIdentifiers

```
        public void
MakeTrainSetWithBootstrapWithKeyIdentifiers(int totalCount)
        {
            _trainSetWithKeyIdentifiers.Clear();
            _trainSetEntity.Inputs = null;
            while (totalCount > 0)
            {
                int keyIndex = FindRandomKeyIndexWithBootstrap();
                int key = _originalInputsKeys[keyIndex];
                keyIndex = FindRandomKeyIndexWithBootstrap();
                key = _originalInputsKeys[keyIndex];
                _trainSetWithKeyIdentifiers.Add(new
SetKeyIdentifier(totalCount, key), _originalInputs[key]);
                totalCount--;
            }
            _trainSetEntity.Inputs = _trainSetWithKeyIdentifiers;
        }
```

MakeTestSetWithKeyIdentifiers

```
        public void MakeTestSetWithKeyIdentifiers(int totalCount)
        {
            _testSetWithKeyIdentifiers.Clear();
            _testSetEntity.Inputs = null;
            while (totalCount > 0)
            {
                int keyIndex = FindRandomKeyIndex();
                int key = _originalInputsKeys[keyIndex];
                _testSetWithKeyIdentifiers.Add(new
SetKeyIdentifier(totalCount, key), _originalInputs[key]);
                _originalInputsKeys.RemoveAt(keyIndex);
                totalCount--;
```

```
            }
            _testSetEntity.Inputs = _testSetWithKeyIdentifiers;
        }
```

MakeTestSetWithBootstrapWithKeyIdentifiers

```
        public void
MakeTestSetWithBootstrapWithKeyIdentifiers(int totalCount)
        {
            _testSetWithKeyIdentifiers.Clear();
            _testSetEntity.Inputs = null;
            while (totalCount > 0)
            {
                int keyIndex = FindRandomKeyIndexWithBootstrap();
                int key = _originalInputsKeys[keyIndex];
                keyIndex = FindRandomKeyIndexWithBootstrap();
                key = _originalInputsKeys[keyIndex];
                _testSetWithKeyIdentifiers.Add(new
SetKeyIdentifier(totalCount, key), _originalInputs[key]);
                totalCount--;
            }
            _testSetEntity.Inputs = _testSetWithKeyIdentifiers;
        }
```

InitSets

```
        public void InitSets()
        {
            int[] randomWeightMultipliers = new int[] { -1, 1, 1,
-1, 1, 1, 1, -2, 1, -1, 1,
                                                        1, 1, -
2, 1, 1, 1, 1, 1, 1, -2,
                                                       -1, -1,
1, 1, 1, 1, 1, -2, 1, -1, 1,
                                                        1, 1, -
2, 1, 1, 1, 1, 1, 1, -2 };
            _sets.Clear();
            for (int i = 0; i < _kNumberOfSets; i++)
            {
                ArtificialKMeansSet kSet = new
ArtificialKMeansSet();
                kSet.SetTag = "SET_" + i.ToString();
                kSet.InputEntity = new DataSetBase();
                kSet.InputEntity.Inputs = new
Dictionary<SetKeyIdentifier, List<double>>();
                kSet.MeanPoints = new List<double>();
                for (int j = 0; j < _attributeCount; j++)
                {
                    RandomNumberGenerator.GetRandomNumber();
                    int carpan =
RandomNumberGenerator.GetRandomNumber(0,
                        randomWeightMultipliers.Length - 1);

kSet.MeanPoints.Add(RandomNumberGenerator.GetRandomNumber());
                }
                kSet.AvgOutput = -1.0;
                kSet.FarestInputKeyToMean = -1;
```

```
                kSet.MaxOutput = -1.0;
                kSet.MinOutput = -1.0;
                kSet.NearestInputKeyToMean = -1;
                kSet.StandardDeviation = -1.0;
                kSet.Variance = -1.0;
                _sets.Add(kSet);
            }
        }
```

BuildKMeansSetsEntity

```
        public void BuildKMeansSetsEntity(DataSetBase
originalTrainSet)
        {
            _trainSetEntity =
InitTrainSetEntity(originalTrainSet);
            bool changeOccured = true;
            bool firstRun = true;
            bool maxCycleReached = false;
            int cycleCount = 0;
            while (changeOccured && maxCycleReached == false)
            {
                changeOccured = false;
                foreach (SetKeyIdentifier var in
_trainSetEntity.Inputs.Keys)
                {
                    int kIndex =
FindNearestProperSet(_trainSetEntity[var.Key]);
                    if
(_sets[kIndex].InputEntity.ContainsIndex(var.Index) == false)
                    {
                        _sets[kIndex].InputEntity.Inputs.Add(var,
_trainSetEntity[var.Key]);
                        UpdateFullnessRatioForEntity(kIndex);
                        if (firstRun == true) firstRun = false;
                        else changeOccured = true;
                    }
                    for (int j = 0; j < _kNumberOfSets; j++)
                    {
                        if (kIndex != j &&
_sets[j].InputEntity.ContainsIndex(var.Index))
                        {

_sets[j].InputEntity.Inputs.Remove(var);
                            UpdateFullnessRatioForEntity(j);
                            changeOccured = true;
                            break;
                        }
                    }
                }
                for (int j = 0; j < _kNumberOfSets; j++)
                {
                    for (int n = 0; n < _attributeCount; n++)
                        _sets[j].MeanPoints[n] = 0.0f;
                    foreach (SetKeyIdentifier var in
_sets[j].InputEntity.Inputs.Keys)
                    {
                        for (int n = 0; n < _attributeCount; n++)
```

```
                                _sets[j].MeanPoints[n] +=
_sets[j].InputEntity[var.Key][n];
                    }
                    for (int n = 0; n < _attributeCount; n++)
                        _sets[j].MeanPoints[n] =
(double)(_sets[j].MeanPoints[n] /

_sets[j].InputEntity.Inputs.Count);
                }
                cycleCount++;
                maxCycleReached = (cycleCount >=
_trainingCycleCount);
            }
        }
```

FindNearestProperSet

```
        private int FindNearestProperSet(List<double> inputs)
        {
            double[,] oclidDistanceValues = new
double[_kNumberOfSets, 2];

            for (int i = 0; i < _kNumberOfSets; i++)
            {
                oclidDistanceValues[i, 0] = i;
                oclidDistanceValues[i, 1] =
CalculateOclidDistance(inputs, _sets[i].MeanPoints);
            }
            BubbleSort(ref oclidDistanceValues, _kNumberOfSets);
            for (int i = 0; i < _kNumberOfSets; i++)
            {
                if (_sets[Convert.ToInt32(oclidDistanceValues[i,
0])].FullnessRatio <= _fullnessRatio)
                    return Convert.ToInt32(oclidDistanceValues[i,
0]);
            }
            return
Convert.ToInt32(oclidDistanceValues[_kNumberOfSets - 1, 0]);
        }
```

FindNearestOutputRangeWithANNOutForEntity

```
        public List<double>
FindNearestOutputRangeWithANNOutForEntity(double aNNOut, int
kSetIndex)
        {
            double[,] outputValues = new
double[_sets[kSetIndex].InputEntity.Inputs.Count, 2];
            int length = 0;
            foreach (SetKeyIdentifier var in
_sets[kSetIndex].InputEntity.Inputs.Keys)
            {
                outputValues[length, 0] = var.Key;
                outputValues[length, 1] =
_sets[kSetIndex].InputEntity[var.Key][_outputValueIndex];
                length++;
            }
```

```
            BubbleSort(ref outputValues,
_sets[kSetIndex].InputEntity.Inputs.Count);
            List<double> lst = new List<double>();
            if (aNNOut < outputValues[0, 1])
            {
                lst.Add(aNNOut);
                lst.Add(outputValues[0, 1]);
                return lst;
            }
            else if (aNNOut > outputValues[length - 1, 1])
            {
                lst.Add(outputValues[length - 1, 1]);
                lst.Add(aNNOut);
                return lst;
            }
            for (int i = 1; i < length - 2; i++)
            {
                if (aNNOut == outputValues[i, 1])
                {
                    outputValues[i, 1] = outputValues[i - 1, 1];
                }
            }
            Dictionary<double, double> lstDistinct = new
Dictionary<double, double>();
            int idx = 0;
            if (length == 1)
                lstDistinct.Add(0, outputValues[0, 1]);
            for (int i = 0; i < length - 1; i++)
            {
                if (lstDistinct.ContainsValue(outputValues[i, 1])
== false)
                {
                    lstDistinct.Add(idx, outputValues[i, 1]);
                    idx++;
                }
            }
            if (lstDistinct.Count == 1)
            {
                if (lstDistinct[0] < aNNOut)
                {
                    lst.Add(lstDistinct[0]);
                    lst.Add(aNNOut);
                }
                else
                {
                    lst.Add(aNNOut);
                    lst.Add(lstDistinct[0]);
                }
                return lst;
            }
            if (aNNOut < lstDistinct[0])
            {
                lst.Add(aNNOut);
                lst.Add(lstDistinct[0]);
                return lst;
            }
            else if (aNNOut > lstDistinct[lstDistinct.Count - 1])
            {
                lst.Add(lstDistinct[lstDistinct.Count - 1]);
                lst.Add(aNNOut);
```

```
                return lst;
            }
            else if (aNNOut == lstDistinct[0])
            {
                lst.Add(aNNOut);
                lst.Add(lstDistinct[0]);
                return lst;
            }
            else if (aNNOut == lstDistinct[lstDistinct.Count -
1])
            {
                lst.Add(aNNOut);
                lst.Add(lstDistinct[lstDistinct.Count - 1]);
                return lst;
            }
            for (int i = 0; i < lstDistinct.Count - 1; i++)
            {
                if (aNNOut > lstDistinct[i] && aNNOut <
lstDistinct[i + 1])
                {
                    lst.Add(lstDistinct[i]);
                    lst.Add(lstDistinct[i + 1]);
                    return lst;
                }
            }
            return null;
        }
```

FindNearestSet

```
        public int FindNearestSet(List<double> inputs)
        {
            double oclidDistance = CalculateOclidDistance(inputs,
_sets[0].MeanPoints);
            double oclidDistanceTmp = 0;
            int kIndex = 0;
            for (int i = 1; i < _kNumberOfSets; i++)
            {
                oclidDistanceTmp = CalculateOclidDistance(inputs,
_sets[i].MeanPoints);
                if (oclidDistanceTmp < oclidDistance)
                {
                    oclidDistance = oclidDistanceTmp;
                    kIndex = i;
                }
            }
            return kIndex;
        }
```

InitNetwork

```
        public void InitNetwork()
        {
            int[] randomWeightMultipliers = new int[] { -1, 1, 1,
-1, 1, 1, 1, -2, 1, -1, 1, 1, 1,
                                                        -2, 1, 1,
1, 1, 1, 1, -2, -1, -1, 1,
```

```
                                                                    1, 1, 1,
1, -2, 1, -1, 1, 1, 1, -2,
                                                                    1, 1, 1,
1, 1, 1, -2 };
            for (int i = 0; i < _inputLayerNodeCount; i++)
            {
                InputNeuron inputNeuron = new InputNeuron();
                inputNeuron.Tag = "INPUT_" + i.ToString();
                inputNeuron.InputToHiddenNeuronWeights = new
List<double>();

                for (int j = 0; j < _hiddenLayerNodeCount; j++)
                {
                    RandomNumberGenerator.GetRandomNumber();
                    int carpan =
RandomNumberGenerator.GetRandomNumber(0,
randomWeightMultipliers.Length - 1);

inputNeuron.InputToHiddenNeuronWeights.Add(RandomNumberGenerator.
GetRandomNumber() * randomWeightMultipliers[carpan]);
                }
                _inputLayer.Add(inputNeuron);
            }
            for (int i = 0; i < _hiddenLayerNodeCount; i++)
            {
                HiddenNeuron hiddenNeuron = new HiddenNeuron();
                hiddenNeuron.Tag = "HIDDEN_" + i.ToString();
                hiddenNeuron.HiddenToOutputNeuronWeight =
RandomNumberGenerator.GetRandomNumber();
                _hiddenLayer.Add(hiddenNeuron);
            }
            _outputLayer.Tag = "OUTPUT";
        }
```

TrainNetworkWithEntity

```
        public void TrainNetworkWithEntity(DataSetBase trainSet)
        {
            _trainSetEntity = trainSet;
            for (int i = 0; i < this._trainingCycleCount; i++)
            {
                foreach (SetKeyIdentifier keyIdf in
trainSet.Inputs.Keys)
                {
                    double targOut =
trainSet[keyIdf.Key][_outputValueIndex];
                    PassForwardWithEntity(keyIdf.Key, targOut);
                    BackPropagateExt(targOut);
                }
            }
        }
```

PassForwardWithEntity

```
        private void PassForwardWithEntity(int key, double
targOut)
        {
```

```
            for (int i = 0; i < _inputLayerNodeCount; i++)
            {
                _inputLayer[i].Input = _trainSetEntity[key][i];
            }
            for (int i = 0; i < _hiddenLayerNodeCount; i++)
            {
                double sum = 0.0;
                for (int j = 0; j < _inputLayerNodeCount; j++)
                {
                    sum += _inputLayer[j].Input *
_inputLayer[j].InputToHiddenNeuronWeights[i];
                }
                _hiddenLayer[i].Value =
SigmoidActivationFunction.processHiddenValue(sum,
_hiddenLayerActivationType);
            }
            for (int i = 0; i < _outputLayerNodeCount; i++)
            {
                double sum = 0.0;
                for (int j = 0; j < _hiddenLayerNodeCount; j++)
                {
                    sum += _hiddenLayer[j].Value *
_hiddenLayer[j].HiddenToOutputNeuronWeight;
                }
                _outputLayer.Value =
SigmoidActivationFunction.processValue(sum,
_outputLayerActivationType);
            }
        }
```

BackPropagateExt

```
        private void BackPropagateExt(double targOut)
        {
            for (int i = 0; i < _hiddenLayerNodeCount; i++)
            {
                _hiddenLayer[i].HiddenNeuronDeltaValue = 0.0f;
            }
            _outputLayer.OutputNeuronDeltaValue = 0.0f;

            _outputLayer.OutputNeuronDeltaValue =
_outputLayer.Value * (1.0 - _outputLayer.Value) * (targOut -
_outputLayer.Value);
            for (int i = 0; i < _hiddenLayerNodeCount; i++)
            {
                _hiddenLayer[i].HiddenToOutputNeuronWeight +=
_learningRate * _outputLayer.OutputNeuronDeltaValue *
_hiddenLayer[i].Value;
            }
            for (int i = 0; i < _hiddenLayerNodeCount; i++)
            {
                _hiddenLayer[i].HiddenNeuronDeltaValue =
_hiddenLayer[i].Value * (1 - _hiddenLayer[i].Value) *
(_outputLayer.OutputNeuronDeltaValue *
_hiddenLayer[i].HiddenToOutputNeuronWeight);
            }
            for (int i = 0; i < _inputLayerNodeCount; i++)
            {
```

```
                for (int j = 0; j < _hiddenLayerNodeCount; j++)
                {
                    _inputLayer[i].InputToHiddenNeuronWeights[j]
+= _learningRate * _hiddenLayer[j].HiddenNeuronDeltaValue *
_inputLayer[i].Input;
                }
            }
        }
```

TestNetwork

```
        public double TestNetwork(List<double> testSetRow)
        {
            for (int i = 0; i < _inputLayerNodeCount; i++)
            {
                _inputLayer[i].Input = testSetRow[i];
            }
            for (int i = 0; i < _hiddenLayerNodeCount; i++)
            {
                double sum = 0.0;
                for (int j = 0; j < _inputLayerNodeCount; j++)
                {
                    sum += _inputLayer[j].Input *
_inputLayer[j].InputToHiddenNeuronWeights[i];
                }
                _hiddenLayer[i].Value =
SigmoidActivationFunction.processHiddenValue(sum,
_hiddenLayerActivationType);
            }
            for (int i = 0; i < _outputLayerNodeCount; i++)
            {
                double sum = 0.0;
                for (int j = 0; j < _hiddenLayerNodeCount; j++)
                {
                    sum += _hiddenLayer[j].Value *
_hiddenLayer[j].HiddenToOutputNeuronWeight;
                }
                _outputLayer.Value =
SigmoidActivationFunction.processValue(sum,
_outputLayerActivationType);
            }
            return _outputLayer.Value;
        }
```

FindCocomoEstimateForEntity

```
        private double FindCocomoEstimateForEntity(int key)
        {
            List<double> lstTestRow = new
List<double>(dsReader.TestSetEntity[key].ToArray());

            double coefficient =
FindCocomoIntermediateTypeCoefficient((int)lstTestRow[cocomoInter
mediateTypeIndex]);
            double scaleFactor =
FindCocomoIntermediateTypeScaleFactor((int)lstTestRow[cocomoInter
mediateTypeIndex]);
```

```
            double kloc =
lstTestRow[cocomoLOCoriginalValueIndex];
            lstTestRow.RemoveAt(lstTestRow.Count - 1);//actual
            lstTestRow.RemoveAt(lstTestRow.Count - 1);//loc
            lstTestRow.RemoveAt(lstTestRow.Count -
1);//intermediate type
            lstTestRow.RemoveAt(lstTestRow.Count -
1);//normalized actual
            lstTestRow.RemoveAt(lstTestRow.Count -
1);//normalized loc
            double output = 1.0f;
            foreach (double item in lstTestRow)
            {
                output = output * (item * 10);
            }
            return coefficient * Math.Pow(kloc, scaleFactor) *
output;
        }
```

FindMRE

```
        private double FindMRE(double targOut, double
annActualOut)
        {
            return (Math.Abs(targOut - annActualOut) / targOut) *
100;
        }
```

RunANN

```
        private void RunANN(object obj)
        {
            ArtificialNeuralNetwork ann =
(ArtificialNeuralNetwork)obj;
            RandomNumberGenerator.GetRandomNumber();
            ann.InitNetwork();
            ann.TrainNetworkWithEntity(dsReader.TrainSetEntity);
        }
```

RunKMeans

```
        private void RunKMeans(object obj)
        {
            ArtificialKMeans kMeans = (ArtificialKMeans)obj;
            kMeans.InitSets();

kMeans.BuildKMeansSetsEntity(dsReader.TrainSetEntity);
            kMeans.CalculateStatsForSets();
        }
```

buttonReadDataSet_Click

```
        private void buttonReadDataSet_Click(object sender,
EventArgs e)
```

```
        {
            if (CheckUserInputs() == false)
                return;
            dsReader.FillOriginalInputsRefList();
            if (useBootstrap)
            {

dsReader.MakeTrainSetWithBootstrapWithKeyIdentifiers(trainSetItem
Count);

dsReader.MakeTestSetWithBootstrapWithKeyIdentifiers(testSetItemCo
unt);
            }
            else
            {

dsReader.MakeTrainSetWithKeyIdentifiers(trainSetItemCount);

dsReader.MakeTestSetWithKeyIdentifiers(testSetItemCount);
            }

            hiddenLayerCount =
string.IsNullOrEmpty(txtANNHiddenNeuronCount.Text) == true ? 5 :
Convert.ToInt32(txtANNHiddenNeuronCount.Text);
            ArtificialNeuralNetwork ann = new
ArtificialNeuralNetwork(16, hiddenLayerCount, 16,
AnnLearningRate, AnnEpochCount,
(ActivationFunctionType)cmbANNHiddenActivationFunctionType.Select
edItem,
(ActivationFunctionType)cmbANNOutputActivationFunctionType.Select
edItem);
            fullnessRatio = 1.2f *
((double)((dsReader.TrainSetEntity.Inputs.Count / KMeansSetCount)
+ 1) / Convert.ToDouble(dsReader.TrainSetEntity.Inputs.Count)) *
100;
            fullnessRatio =
string.IsNullOrEmpty(txtKMeansFullnessRatio.Text) == true ?
fullnessRatio : Convert.ToDouble(txtKMeansFullnessRatio.Text);
            ArtificialKMeans kMeans = new
ArtificialKMeans(KMeansSetCount, 17, 16, fullnessRatio,
KMeansEpochCount);

            System.Threading.Thread thrANN = new
System.Threading.Thread(new
System.Threading.ParameterizedThreadStart(RunANN));
            System.Threading.Thread thrKMeans = new
System.Threading.Thread(new
System.Threading.ParameterizedThreadStart(RunKMeans));
            thrANN.Start(ann);
            thrKMeans.Start(kMeans);

            thrANN.Join();
            thrKMeans.Join();

            double mreANN = 0.0f;
            double mMreANN = 0.0f;
            int mMreANNCount = 0;

            double mreKMeansANN = 0.0f;
            double mMreKMeansANN = 0.0f;
```

```
            txtResults.Text = "";
            txtResults.Text += "   ***   KONFİGÜRASYON DEĞERLERİ
***   \r\n";
            txtResults.Text += "YSA Katmanları: 16 - " +
hiddenLayerCount.ToString() + " - 1 \r\n";
            txtResults.Text += "YSA Öğrenme Oranı: " +
AnnLearningRate.ToString() + " \r\n";
            txtResults.Text += "YSA Döngü Sayısı: " +
AnnEpochCount.ToString() + "  \r\n";
            txtResults.Text += "K-Means Küme Sayısı: " +
kMeans._sets.Count.ToString() + "  \r\n";
            txtResults.Text += "K-Means Küme Doluluk Eşit Oranı:
" + fullnessRatio.ToString() + "  \r\n";
            txtResults.Text += "K-Means Döngü Sayısı: " +
KMeansEpochCount.ToString() + "  \r\n";
            txtResults.Text += "Eğitim Kümesi Eleman Sayısı: " +
dsReader.TrainSetEntityCount.ToString() + "  \r\n";
            txtResults.Text += "Test Kümesi Eleman Sayısı: " +
dsReader.TestSetEntityCount.ToString() + "  \r\n";
            txtResults.Text += "************************\r\n";

            DataTable dtOut = CreateOutputDataTable();

            foreach (ArtificialKMeansSet set in kMeans._sets)
            {
                txtResults.Text += "K-Means Küme " + set.SetTag +
"\r\n";
                txtResults.Text += "Eleman Sayısı " +
set.InputEntity.Inputs.Count.ToString() + "\r\n";
                txtResults.Text += "Doluluk Eşik Oranı: " +
set.FullnessRatio.ToString() + "\r\n";
                txtResults.Text += "Küme Merkezleri: ";
                foreach (double var in set.MeanPoints)
                {
                    txtResults.Text += var.ToString() + " ; ";
                }
                txtResults.Text += "\r\n";
                txtResults.Text += "En Yakın ID: " +
set.NearestInputKeyToMean.ToString() + "\r\n";
                txtResults.Text += "En Uzak ID: " +
set.FarestInputKeyToMean.ToString() + "\r\n";
                txtResults.Text += "Ortalama: " +
set.AvgOutput.ToString() + "\r\n";
                txtResults.Text += "Min: " +
set.MinOutput.ToString() + "\r\n";
                txtResults.Text += "Maks: " +
set.MaxOutput.ToString() + "\r\n";

                foreach (SetKeyIdentifier var in
set.InputEntity.Inputs.Keys)
                {
                    txtResults.Text += "# " + var.Key + "\t";
                    foreach (double val in
set.InputEntity[var.Key])
                    {
                        txtResults.Text += val.ToString() + "\t";
                    }
                    txtResults.Text += "\r\n";
                }
```

```
                txtResults.Text += "-----------------------------
--\r\n";
            }
            txtResults.Text += "************************\r\n";

            InitChart();
            double[,] chartElements = new
double[dsReader.TestSetEntityCount, 5];
            int idx = 0;

            foreach (SetKeyIdentifier var in
dsReader.TestSetEntity.Inputs.Keys)
            {
                DataRow drow = dtOut.NewRow();
                drow["PROJECT_ID"] = var.Key;
                double annTestResult =
ann.TestNetwork(dsReader.TestSetEntity[var.Key]);
                drow["HEDEF"] =
dsReader.TestSetEntity[var.Key][16];
                drow["YSA_SONUC"] = annTestResult;

                drow["COCOMO_MRE"] =
FindCocomoEstimateMREForEntity(var.Key).ToString("0.00");

                mreANN =
FindMRE(dsReader.TestSetEntity[var.Key][16], annTestResult);
                drow["MRE"] = mreANN.ToString("0.00");
                mMreANN += mreANN;
                mMreANNCount++;
                txtResults.Text += "Hedef: " +
dsReader.TestSetEntity[var.Key][16].ToString() + "\r\n";
                txtResults.Text += "YSA Sonuç: " +
annTestResult.ToString() + "\r\n";
                txtResults.Text += "MRE: " + mreANN.ToString() +
"\r\n";

                int kIndex =
kMeans.FindNearestSet(dsReader.TestSetEntity[var.Key]);
                drow["EN_YAKIN_KUME"] = kIndex;
                drow["KUME_MIN"] =
kMeans._sets[kIndex].MinOutput;
                drow["KUME_ORT"] =
kMeans._sets[kIndex].AvgOutput;
                drow["KUME_MAKS"] =
kMeans._sets[kIndex].MaxOutput;
                txtResults.Text += "K-Means En Yakın Küme: " +
kIndex.ToString() + "\r\n";
                double oclidOut =
kMeans.FindNearestOclidResultForEntity(dsReader.TestSetEntity[var
.Key], kIndex);
                drow["KUME_OKLID_CIKTI"] = oclidOut;
                txtResults.Text += "K-Means En Yakın Oklid
Sonucu: " + oclidOut.ToString() + "\r\n";
                double resultOut =
kMeans.FindNearestOutputResultForEntity(dsReader.TestSetEntity[va
r.Key], kIndex);
                drow["KUME_SONUC_CIKTISI"] = resultOut;
                txtResults.Text += "K-Means En Yakın Sonuç: " +
resultOut.ToString() + "\r\n";
                mreKMeansANN =
```

```
FindMRE(dsReader.TestSetEntity[var.Key][16], resultOut);
                drow["K-MEANS_MRE"] =
mreKMeansANN.ToString("0.00"); ;
                drow["K-MEANS_MAX_MRE"] =
FindMRE(dsReader.TestSetEntity[var.Key][16],
kMeans._sets[kIndex].MaxOutput).ToString("0.00"); ;
                drow["K-MEANS_MIN_MRE"] =
FindMRE(dsReader.TestSetEntity[var.Key][16],
kMeans._sets[kIndex].MinOutput).ToString("0.00"); ;

                txtResults.Text += "K-Means MRE: " +
mreKMeansANN.ToString() + "\r\n";
                txtResults.Text += "Test Satırı Küme İçinde Mi: "
+ GetBooleanString(dsReader.TestSetEntity[var.Key][16] >=
kMeans._sets[kIndex].MinOutput &&
dsReader.TestSetEntity[var.Key][16] <=
kMeans._sets[kIndex].MaxOutput) + "\r\n";
                mMreKMeansANN += mreKMeansANN;

                List<double> dList =
kMeans.FindNearestOutputRangeWithKMeansOutForEntity(resultOut,
kIndex);
                txtResults.Text += "K-Means Beklenen Değer
Aralığı " + dList[0].ToString() + " ve " + dList[1].ToString() +
"\r\n";
                List<double> dListAnn =
kMeans.FindNearestOutputRangeWithANNOutForEntity(annTestResult,
kIndex);
                drow["YSA_CIKTI_ALTLIMIT"] = dList[0];
                drow["YSA_CIKTI_USTLIMIT"] = dList[1];
                drow["K-MEANS_ALTLIMIT_MRE"] =
FindMRE(dsReader.TestSetEntity[var.Key][16],
dList[0]).ToString("0.00"); ;
                drow["K-MEANS_USTLIMIT_MRE"] =
FindMRE(dsReader.TestSetEntity[var.Key][16],
dList[1]).ToString("0.00"); ;
                drow["YSA_CIKTI_KUME_ICINDEMI"] =
GetBooleanString(annTestResult >= dListAnn[0] && annTestResult <=
dListAnn[1]);
                txtResults.Text += "K-Means YSA sonucu beklenen
aralığı " + dListAnn[0].ToString() + " ve " +
dListAnn[1].ToString() + "\r\n";
                txtResults.Text += "YSA Sonucu Küme İçinde Mi:" +
GetBooleanString(annTestResult >= dListAnn[0] && annTestResult <=
dListAnn[1]) + "\r\n";

                dtOut.Rows.Add(drow);
                txtResults.Text += "----------------------------
-\r\n";

                chartElements[idx, 0] = idx;
                chartElements[idx, 1] =
dsReader.TestSetEntity[var.Key][16];
                chartElements[idx, 2] = annTestResult;
                chartElements[idx, 3] = dListAnn[0];
                chartElements[idx, 4] = dListAnn[1];
                idx++;
            }
            mMreANN = mMreANN / mMreANNCount;
            txtResults.Text += "M-MRE: " + mMreANN.ToString() +
```

```
"\r\n";
            mMreKMeansANN = mMreKMeansANN / mMreANNCount;
            txtResults.Text += "K-Means M-MRE: " +
mMreKMeansANN.ToString() + "\r\n";
            txtResults.Text += "------------------------------
\r\n";
            GenerateChartDataSorted(chartElements,
dsReader.TestSetEntity.Inputs.Count);

            FillChart();
            FillTrainSetChart();
            FillTestSetChart();
            dataGridViewOutput.DataSource = dtOut;
            SetOutputGrid();
        }
```

EK-B VERİ KÜMESİ ÖRNEK DOSYA İÇERİĞİ

Aşağıda veri kümesi içerisinde yer alan örnek proje satırları ve her bir sütun niteliği gösterilmektedir. Dosyada aynı zamanda veri kümesi ile ilgili bilgiler, kategorik değerlerin sayısal karşılıkları ve diğer açıklamalar da yer almaktadır.

RELY, DATA, CPLX, TIME, STOR, VIRT, TURN, ACAP, AEXP, PCAP, VEXP, LEXP, MODP, TOOL, SCED, LOC, ACTUAL_EFFORT

Nominal,Very_High,High,Very_High,Very_High,Low,High,Very_High,High,Nominal,Low,High,Very_High,Very_High,Low,32.6,170 % instance number: 21

Nominal,Very_High,High,Very_High,Very_High,Low,High,Very_High,High,Nominal,Low,High,Very_High,Very_High,Low,12.8,62 % instance number: 22

Nominal,Very_High,High,Very_High,Very_High,Low,High,Very_High,High,Nominal,Low,High,Very_High,Very_High,Low,15.4,70 % instance number: 23

Nominal,Very_High,High,Very_High,Very_High,Low,High,Very_High,High,Nominal,Low,High,Very_High,Very_High,Low,16.3,82 % instance number: 24

Nominal,Very_High,High,Very_High,Very_High,Low,High,Very_High,High,Nominal,Low,High,Very_High,Very_High,Low,35.5,192 % instance number: 25

KİŞİSEL YAYIN VE ESERLER

[1] **Saraç Ö. F**., Duru N., A novel method for software effort estimation: estimating with boundaries, *Innovations in Intelligent Systems and Applications (INISTA)*, Albena, Bulgaria, 19-21 June 2013.

[2] **Saraç Ö. F**., Duru N., Yazılım efor tahmininde farklı bir yaklaşım: sınır değerlerine göre tahmin, *7. Ulusal Yazılım Mühendisliği Sempozyumu (UYMS)*, İzmir, Türkiye, 25-28 Eylül 2013 (Kabul edildi).

ÖZGEÇMİŞ

1985 yılında Ordu' da doğdu. İlk, orta ve lise öğretimini Van' da tamamladı. 2002 yılında girdiği Dokuz Eylül Üniversitesi Bilgisayar Mühendisliği bölümünden 2007 yılında mezun oldu. 2011 yılında başladığı Kocaeli Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Bilimleri Anabilim Dalı' ndaki Yüksek Lisans eğitimine devam etmektedir. 6 yıldır Yazılım Mühendisi olarak özel sektörde çalışmalarına devam etmektedir.