**EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ**
**(DOKTORA TEZİ)**

# YAPAY ZEKA YÖNTEMLERİ İLE YAZILIM PROJELERİNDE MALİYET KESTİRİMİ

**Oktay ADALIER**

Bilgisayar Mühendisliği Anabilim Dalı
Bilim Dalı Kodu :619.01.00

**Sunuş Tarihi: 07.03.2008**

**Tez Danışmanı: Yrd. Doç. Dr. Aybars UĞUR**

**Bornova-İZMİR**

**Oktay ADALIER**.....tarafından .. **DOKTORA TEZİ**... tezi olarak sunulan "Yapay Zeka Yöntemleri ile Yazılım Projelerinde Maliyet Kestirimi" başlıklı bu çalışma E.Ü. Lisansüstü Eğitim ve Öğretim Yönetmeliği ile E.Ü. Fen Bilimleri Enstitüsü Eğitim ve Öğretim Yönergesi'nin ilgili hükümleri uyarınca tarafımızdan değerlendirilerek savunmaya değer bulunmuş ve 07/03/2008 tarihinde yapılan tez savunma sınavında aday oybirliği/oyçokluğu ile başarılı bulunmuştur.

**Jüri Üyeleri:**

Jüri Başkanı :Yrd. Doç. Dr. Aybars UĞUR imza .....

Raportör Üye : Prof. Dr. Serdar KORUKOĞLU imza .....

Üye : Doç. Dr. Kadir ERTAŞ imza .....

Üye : Doç. Dr. ..Mustafa Murat İNCEOĞLU imza .....

Üye : Yrd. Doç. Dr. Korhan KARABULUT imza .....

III

# ÖZET

## YAPAY ZEKA YÖNTEMLERİ İLE YAZILIM PROJELERİNDE MALİYET KESTİRİMİ

ADALIER, Oktay

Doktora Tezi, Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi: Yrd. Doç. Dr. Aybars UĞUR

Mart 2008, 96 sayfa

Yazılım maliyet tahminlemesi, yazılım geliştirmenin en önemli aşamalarından birisidir. Proje yöneticisi, proje süresini/maliyetini doğru tahminleyerek projedeki belirsizlikleri azaltır ve projenin gelişimini gerçek giderlerle, planlananları veya tahminlenenleri karşılaştırarak değerlendirir. Yazılım projeleri karmaşıklaştıkça, tahminleme yöntemlerinin önemi de artmaktadır.

Bu tezde, literatürdeki makine öğrenmesi ve yapay zeka tabanlı yazılım tahminleme teknikleri karşılaştırılmıştır. Regresyon ve çok katmanlı algılayıcı yapay sinir ağı modellerinin gerçekleştirimi yapılarak sonuçlar elde edilmiş ve bu değerlere dayalı olarak yöntemler değerlendirilmiştir. Yapay sinir ağlarının eğitim ve test aşamalarında ve regresyon katsayılarının bulunmasında ISBSG (International Software Benchmarking Standards Group) veri seti sürüm 9 kullanılmıştır. Kuvvet değerleri kullanılarak, regresyon tabanlı yeni ve başarılı bir yazılım tahminleme modeli de geliştirilmiştir. Yöntemin iyileştirilmesi için yapılabilecekler ve diğer kestirim modellerine nasıl uyarlanabileceği tartışılmıştır.

**Anahtar Sözcükler:** Yazılım maliyet kestirimi, yapay sinir ağları, çok katmanlı algılayıcı, regresyon, yapay zeka, makine öğrenmesi

# ABSTRACT

## EFFORT ESTIMATION IN SOFTWARE PROJECTS BY USING ARTIFICIAL INTELLIGENCE METHODS

ADALIER, Oktay

PhD. Thesis in Computer Engineering
Supervisor: Assistant Prof. Dr. Aybars UĞUR

March 2008, 96 pages

Software cost estimation is one of the most crucial phases of software development. A project manager can reduce the uncertainty about project by creating more accurate time/cost estimates and can evaluate project progress by comparing actual costs versus planned, or estimated costs. As the complexity of software projects grows, the role for estimating techniques will expand.

In this thesis, software estimation techniques based on machine learning and artificial intelligence in the literature are compared. Regression and multilayer perceptron neural network models are implemented and evaluated based on experimental results that are obtained. ISBSG (International Software Benchmarking Standards Group) data set Release 9 is used for training and testing the neural network, and finding regression coefficients. A new regression based software cost estimation model obtaining maximum performance is also developed using power values. There is some important discussion on how the results can be improved and how they can be applied to other estimation models.

**Keywords:** Software cost estimation, artificial neural networks, multilayer perceptron, regression, artificial intelligence, machine learning.

## TEŞEKKÜR

Tez çalışmasını yöneten, destek veren ve tüm olumsuz durumlarda bana moral verip cesaretlendiren Ege Üniversitesi Bilgisayar Mühendisliği öğretim üyesi ve danışmanım Yrd. Doç. Dr. Aybars UĞUR'a teşekkürü bir borç bilirim. Ayrıca doktora çalışmam boyunca bana gösterdiği yakın ilgi ve desteğinden dolayı Ege Üniversitesi Bilgisayar Mühendisliği Bölüm Başkan Yardımcısı Prof. Dr. Serdar KORUKOĞLU'ya ve Doç. Dr. Kadir ERTAŞ'a teşekkür ederim.

Tez çalışması süresince gösterdikleri yakın ilgi ve sağladıkları teknik ve donanım, laboratuar ve yurtdışı konferanslara katılım olanakları için TÜBİTAK-UEKAE enstitüsü müdürü Önder YETİŞ'e, iş arkadaşlarıma teşekkür ederim.

İzmir'de tez çalışmalarımda bana manevi motivasyon açısından sürekli destekli olan sayın Hulusi Ulaş bey'e teşekkürü bir borç bilirim.

Tez çalışmam süresince bana ellerinden gelen fedakarlığı esirgemeyerek maddi ve manevi destek veren, her zaman yanımda olan aileme teşekkür ederim.

İÇİNDEKİLER

**SAYFA**

# ŞEKİLLER DİZİNİ

**Şekiller** **Sayfa**

# ÇİZELGELER DİZİNİ

**Çizelgeler** **Sayfa**

# 1. GİRİŞ

Yazılım endüstrisinin geliştirme maliyeti, çalışma eforu ve zamanı konularında tam ve hassas bir sonuç bilinmesi eldeki olanaklarla olanaksız gözükmektedir. Bu durum proje yönetim danışmanlığı yapan kurumların raporlarından, örnek çalışmalardan, başarısızlıkla sonuçlanmış projelerin makalelerinden anlaşılmaktadır (Grimstad et al., 2006, Boraso et al., 1996, Jorgensen and Shepperd, 2007, Dolado, 2001).

Yazılım projelerinin geliştirme maliyetlerindeki planlanan maliyeti gerçek maliyetler çok-çok aşmaktadır. Buna örnek olarak Standish Group's Chaos raporu verilebilir (Jorgensen and Shepperd, 2007). Söz konusu raporda projelerdeki yazılım geliştirme maliyetlerinin %89 aştığı bildirilmektedir. Bunun nedenleri bölüm ikide detaylı olarak anlatılmaktadır.

Yazılım maliyetleri devlet ve özel sektör kurumlarının bütçelerinde önemli harcama kalemi olarak bulunmaktadır. Literatür ve endüstride birçok yazılım geliştirme alanında çalışanlar yeni yazılım süreçleri üzerinde çalışmaktadır. Böylelikle yazılım geliştirme maliyetlerini minimum'a indirgemeye çalışmaktadırlar. Fakat buna rağmen yazılım projelerinin başarısızlıkla tamamlanma oranları hala yüksektir. Son yıllarda başarısız yazılım projelerinin Amerikan ekonomisine getirdiği yük $75 milyon Amerikan dolarını geçmiş bulunmaktadır (Shan, 2002).

Sonuç olarak proje geliştirme maliyetlerinde planlanan kestirimi (tahminleme) aşan miktar küçümsenemeyecek boyutlardadır. Yapılan projelerin %60-%80'i planlanan maliyet ve zamanı aşmaktadır. Projelerin büyüklüğü oranında aşmaların getirdiği (maliyet /zaman) yük büyük olmaktadır. Projelerde %30-%40 oranında aşmalar (overrun)

ortalamayı oluşturmaktadır ve sektör tarafından normal kabul edilmektedir (Finnie, 1997).

Buna göre:

- 100 000$'lık bir projede aşma miktarı ~30-40 bin $ olacaktır.
- 1000 000$'lık bir projede aşma miktarı ~300-400 bin $ olacaktır.
- 10 000 000$'lık bir projede aşma miktarı ~3-4 milyon $ olacaktır.
- 100 000 000$'lık bir projede aşma miktarı ~30-40 milyon $ olacaktır.

Bu sonuçların bilişim alanında müşteri konumunda bulunan özel veya kamu kurumları tarafından kabul edilebilir olduğu mümkün görünmemektedir.

Projelerde kullanılan maliyet kestirim yöntemi genellikle uzmanların verdiği kararlara göre yapılmaktadır. Bunun başlıca nedeni formal yöntemlerle yapılan kestirimlerin daha iyi olacağına dair güvensizliktir. Ayrıca literatürde projelerde maliyet ve zaman aşımı konularında detaylı ve kapsamlı bir araştırmanın Jorgensen (Jorgensen and Shepperd, 2007) tarafından yapılmasına kadar gerçekleştirmemiş olmasından kaynaklanabilir.

Bir projenin yaklaşık maliyetinin proje başlangıcında bilinmesi projeyi başlatma gerekçeleri açısından önem taşımaktadır. Projenin müşterileri veya üst yönetim projeyi yapıp yapmamak konusunda kestirim değerlerine göre karar almaktadır. Yanlış kestirimler müşteri durumundaki kurum veya kuruluşları ekonomik ve stratejik açıdan etkilemektedir. Örneğin büyük projelerin %60'ı öngörülen proje

bütçelerini aşmışlardır. Bir takım projelerin %15 maliyet aşımı nedeniyle hiçbir zaman tamamlanmadığı gözlemlenmiştir (LIU and Mintram, 2005).

Bir yazılım projesinin en önemli maliyet etken parametresi ise çalışan çabasıdır. Dolayısıyla maliyet kestirimi diğer bir ifade ile çaba kestirimi anlamına gelmektedir. Maliyet kestirimi için birçok kaynakta çeşitli ölçütler verilmiştir (Boraso et al., 1996, Wieczorek and Ruhe, 2002). Fakat önerilen ölçütler için değerlerin toplaması oldukça zor ve zahmetli olduğundan bunlardan bir kısmının veya en etkili bir kümenin kullanılması tavsiye edilmektedir (Wieczorek and Ruhe, 2002, ISBSG, 2004).

Maliyet kestirimlerindeki etkili ölçütlerin azaltılması maliyet yöntemlerinin lineer metotlar yardımıyla kestirimine dönüşmüştür. Fakat lineer yöntemlerde değerlendirmeye alınan tüm parametrelerin lineer olmasına dikkat edilmelidir. Lineer olmama durumunun olumsuz etkilerinden kurtulmak için araştırıcılar çeşitli kestirim yöntemlerine yönelmişlerdir. Bunlara örnek olarak yapay sinir ağları, regresyon bazlı istatistiksel yöntemler ve genetik algoritmalar verilebilir (Finnie, et al., 1997). Bu çalışmalarla proje maliyet kestiriminde evrimsel yaklaşımlar dönemi başlamıştır (Chang et al., 2001).

Bu kapsamda, bölüm ikide, yazılım projelerinde maliyet kestirimine neden gerek olduğu, bu konudaki problemlerin neler olduğu anlatılmaktadır. Bölüm üçte, maliyet kestiriminin bir öğrenme yetisi sonucunda gerçekleştirilen bir faaliyet olduğu ve kestirim yöntemlerinin temelde istatistiksel öğrenme teorisine göre hareket ettikleri anlatılmaktadır. Bölümün sonraki aşamalarında regresyon temelli maliyet kestirimi, yapay sinir ağ tabanlı maliyet kestirim fonksiyonları son olarak genetik programlama tabanlı maliyet kestirim yöntemleri anlatılmaktadır.

Ayrıca aynı bölümde yöntemlerin uygulanmasından sonra sonuçların nasıl değerlendirilmesi gerektiği anlatılmaktadır. Deneysel çalışmalarla kullanılacak veri setinin analizi yine bu bölümde açıklanmaktadır. Bölüm dörtte, maliyet kestirim konusunda önceki çalışmalar sırayla ele alınarak anlatılmaktadır. Bölüm beşte, bölüm üçte anlatılan maliyet kestirim yöntemlerinin veri seti üzerine uygulaması matematiksel sonuçlarla anlatılmakta ve grafiksel öğelerle gösterilmektedir. Bölümde altıda, beşinci bölümde elde edilen sonuçlar literatürdeki benzerleri ile karlılaştırılmaktadır. Bölüm yedide, tezin sonuçları ve gelecek çalışmalar için öneriler anlatılmaktadır.

## 2. Yazılım Projelerinde Maliyet Kestirimi

Proje Yönetiminde geliştirilecek bir yazılımının ne kadar bir çaba gerektireceği ve o çabanın neye mal olacağı en önemli değişkenlerden biridir. Fakat bu çabanın belirlenmesini zorlaştıran unsurlar ve problemler nelerdir. Bunlar aşağıdaki gibi özetlenebilir (Heemstra, 1992).

Proje yapan personelin elinde önceden yapılmış benzer projelerin bilgileri bulunmamaktadır ve proje yöneticisinin kestirim yapmasını zorlaştırmaktadır.

Proje alımlarındaki acelecilikten dolayı kestirimler yüzeysel yapılmakta olup projenin sistem mühendisliği tamamlanmadan öngörümlerde bulunulmaktadır. Dolayısıyla sistemin tamamına detaylı bir şekilde hakim olunmadan proje yöneticileri maliyet kestiriminde bulunmaktadırlar.

Projenin Sistem mühendisliği tamamlanmış olmasına rağmen geliştiriciler bazı paketlerin maliyetlerini üzerinde tasarım yapmadan söylemeleri zor olmaktadır. Bundan dolayı zaman içerisinde revize edilen proje kestirimlerine gereksinim bulunmaktadır.

Yazılımların karakteristiklerinden dolayı kestirim yapmak zor olabilmektedir. Örneğin gömülü sistemlerde bazı yeni teknolojileri adapte etmek veya sağlıklı çalıştırmak zor olabildiğinden kestirimler yanlış çıkabilmektedir.

Bir yazılımın gerçekleşmesi için gerekli çaba ve zamanı birçok unsur etkileyebilir. Örneğin geliştirilecek yazılımın karmaşıklığı, geliştirmeye katılan kurum ve kuruluşların takım çalışmasına yatkınlığı ve geliştirmeyi yapan takım elemanlarının tecrübesi veya uzmanlığı yazılımın maliyetini etkileyebilir. Bu gibi unsurların zaman akışı

içerisinde kontrolü zor olduğundan maliyet kestirimleri sapabilmektedir.

Yazılım alanında teknolojilerinin hızlı değişmesi yazılım geliştirme alanındaki tecrübe birikimini azaltmaktadır. Yeni araç ve teknolojilere hakim personel az bulunmakta ve bu tür ortamlarda geliştirme yeni olduğundan birikim az olmaktadır. Dolayısıyla maliyet kestirimi zorlaşmaktadır.

Maliyet kestirimini yapan uzmanlar büyük projelerde maliyet kestiriminde bulunmak için yeterli tecrübe ve birikime sahip olamamaktadırlar. Bunun nedeni bir proje yöneticisi 10 yıl gibi bir zaman diliminde kaç tane büyük proje yapabileceği gerçeğinden ortaya çıkmaktadır. Bu sayı 2 ve 3 gibi bir değeri geçemediğinden birikim az olmakta olup projelerin maliyet kestirimleri zorlaşmaktadır.

Proje yönetimindeki işlerde doğrusal bir akış olamadığından başta yapılan kestirim eksik veya yanlış olmaktadır. Özellikle iletişim ve koordinasyon işlevleri doğrusal bir süreç izlemediğinden kestirimler yanlış olabilmektedir.

Kestirim yapan yetkili kestirim yaptığı işleri kendi yapacak gibi algıladığından işlerin uzman çalışanlar tarafından gerçekleştirileceğini kabul ederek kestirim yapmaktadır. Fakat geliştirme takımlarında bulunan geliştiricilerin bir kısmı yeni ve ilgili alanda tecrübesiz olduğundan kendilerine atanan işler planlanandan daha fazla sürmektedir. Çünkü bir zincir en zayıf halkası kadar sağlamdır. Takımda ki en tecrübesiz veya az başarılı eleman projenin ilgili paketinin geliştirme hızını belirlemektedir. Ayrıca 25 kişinin yapacağı iki yıllık bir projeyi 50 kişi bir yılda yapar demek ciddi bir hatadır. Ayrıca bir projeye ilerleyen zaman içerisinde yeni katılan çalışanlar projenin gecikmesine neden olur. Çünkü yeni katılan çalışanın projenin

mimarisini ve iş paketlerin içeriğini ve diğer paketlerle ilişkisini öğrenmek zamanını almaktadır.

İhale usulü proje alımlarında maliyet kestirimi yapan yetkili üst yönetim tarafından sürekli baskı altında tutulmakta ve proje paketlerinin zaman ve işgücü gereklerini sağlam gerekçelere dayanmadan değiştirmesi büyük bir olasılıkla kısaltılması istenmektedir. Bu tür davranışlar projenin yanlış kestirimlerle başlamasına neden olmaktadır.

Proje yönetimini etkileyen birçok unsurun dikkate alınması için bazı ön tespitlerin yapılması gerekmektedir. Bunlar aşağıdaki gibidir.

Proje kapsamında NE geliştirilecek sorusuna yanıt aranmalıdır. Geliştirilecek ürünün kalitesi ve büyüklüğü kestirim için önem taşımaktadır. Gereklerin hangi sıklıkla değiştiği, yazılım karmaşıklığı, projede tekrar kullanımın hangi oranda kullanılacağı, ne kadar doküman üretileceği ve uygulamanın türü ne olacağı proje maliyet kestirimi için önemlidir.

Projenin NE İLE geliştirileceği konusu projenin maliyetini etkilemektedir. Ürünün geliştirilmesinde yeni nesil geliştirme dilleri ve araçları kullanılacak mı? Kullanılan yazılım geliştirme süreci ve metotları projenin süresini ve kalitesini etkiler.

Projede geliştirme faaliyetlerini yürütecek personelin özellikleri, uzmanlıkları, tecrübeleri, projede tam zamanlı veya kısmi çalışmaları ve son olarak projede ulaşılması hedeflenen kalite seviyesi projenin maliyetini etkilemektedir.

Projenin NASIL geliştirileceği proje maliyeti üzerinde etkilidir. Örneğin geliştirme yapılan kurumda matris türü bir yapılanma mı bulunmakta yoksa proje tabanlı bir yapılanma mı bulunmaktadır. Geliştirme takımları arasında iletişim ve koordinasyon nasıl yapıldığı projenin maliyetini etkilemektedir. Projenin geliştirilmesinde birden

fazla kurum veya kuruluş görev almakta ise iletişim ve koordinasyon zorluğundan dolayı maliyetler artacaktır.

Geliştirilecek ürün KİMİN için yapıldığı büyük önem arz etmektedir. Örneğin geliştirilen ürün bilgisayar dünyası ile önceden iletişimi olmuş ve günlük hayatında yoğunlukla bilgisayar kullanan bir kitle için geliştiriliyorsa kullanımın karmaşıklığı fazla sorun olmayacaktır. Kullanıcılar daha çok ürünün ek özelliklerinden yararlanmaya çalışacaklardır. Fakat ürünün hedef kitlesi eğitim seviyesi düşük bir kullanıcı grubu ise ürünün kolay kullanımı çok önem taşıyacaktır. Bu iki farklı özellikte ürün geliştirilmesinin maliyetleri birbiriden farklı olacaktır.

Yazılım projelerinde maliyet kestiriminde aşağıdaki kıstaslara dikkat etmek gerekmektedir.

Yazılım projeleri adam ay maliyeti genellikle karmaşık ve pahalı olarak görünmektedir. Yazılım üretimi bir kriz içindedir. Sektör aşırı maliyetlerden yakınmaktadır. Yazılım üretimi çoğu kez kontrol dışındadır çünkü ölçülememektedir. Ölçülemeyen bir şey kontrol edilemez. Günümüzde yazılım projeleri geliştirirken kestirilebilir maliyetler koymakta başarılı olunamamaktadır. Maliyet kestirimi açık ve net olmalıdır. Kestirim ve değerlendirme yapabilmek için sistematik ölçüm yapmak gerekmektedir. Bir iş için gereken çabayı baştan bilmek çok önemlidir. Bunu bilmeden işleri yönetmek mümkün değildir. Yönetmek için bu tahminleri kullanıp projeleri baştan kabul etme veya reddetmek çok daha faydalıdır. Çok uzun sürebilecek ve çok kaynak gerektirebilecek bir yazılım geliştirme işine girmemek veya girilme kararı verilse bile baştan gerekli iş gücünü organize etmek verimi her aşamada artıracaktır.

Bugünün dünyasında yazılım maliyet kestiriminin önemi sürekli artmakta ve yayılmaktadır. Bu konuda yapılmış çalışmalar olmasına rağmen hala bu konu tam olarak aydınlatılabilmiş değildir ve üzerinde birçok çalışma yapılması gerekmektedir. Maliyet kestiriminin önemi nedir? Kestirimde problem ne kadar sapmadan sonra başlar? Gibi sorulara cevap bulunması gerekmektedir.

Yazılım projeleri zaman planına ve bütçe planına uymama yönünde kötü ün salmış projelerdir. Bunun temel nedeni gereken iş gücünü baştan yanlış tahmin etmektir. Yanlış tahminler üzerine kurulan projeler yüksek maliyetlere neden olmaktadır. Yazılım mühendisliğinde yazılım projelerinin gereken iş gücünü, zamanı ve bütçeyi tahmin etme yönünde yapılacak çalışmalar bu problemleri azaltacaktır. Yazılım maliyet tahminlemesi bir yazılım sisteminin inşası için gereken çaba miktarının tahminleme sürecidir.

Bir sonraki bölümde burada anlatılan sorunlara çözüm olabilecek maliyet kestirim yöntemleri ve kullandıkları ölçevler ele alınmaktadır. Bu çalışmada en çok kullanılan üç yöntem üzerinde detaylı bilgi verilecektir. Bunlar regresyon analizi, yapay sinir ağlar ve genetik programlama kestirim yöntemleridir.

# 3. Maliyet Kestirimi Yöntemleri

Proje Yönetiminde geliştirilecek bir yazılımının ne kadar bir çaba gerektireceği ve o çabanın neye mal olacağı en önemli değişkenlerden biridir.

Yazılım Maliyet Kestirimi (YMK)'nde genellikle klasik regresyon temelli yöntemler kullanılmaktadır. Fakat son zamanlarda başka yöntemlerin kullanımına başlanmıştır. Örneğin; Yapay Sinir Ağları, Durum Bazlı Çıkarım, bulanık (fuzzy) mantık ve uzman sistemler gibi. İstatistik temelli yöntemlere alternatif yöntemler kullanımı YMK'de değerlendirilen parametreler arasında farklı ilişkiler bularak daha doğru kestirimler yapmayı amaçlamaktadır.

Bu yöntemlerden bazıları, girdiler ile onların çıktılarını birbirleri ile ilişkilendiren maliyet kestirim fonksiyonunun sembolik olarak ifade edilmesini sağlamaktadır. Söz konusu fonksiyonlar maliyet değerlendirmesinde kullanılan parametrelerin davranışlarını doğrudan etkilemektedir. Bu fonksiyonlara maliyet fonksiyonu, ekonomi fonksiyonu veya üretim fonksiyonu adı verilmektedir. Bu fonksiyonlar, proje yöneticilerine proje maliyetinin hesaplanması için sembolik bir ifade etme dili sunarak maliyetlerin kestirimi için veri sunmaktadır (Dolado, 2001).

Sistem mühendisleri proje yönetimi maliyet kestirimi için klasik regresyon yöntemleri ile birlikte yapay sinir ağları yöntemleri kullanarak eldeki verilerden maliyet kestirimi için en iyi kestirimi sağlayacak fonksiyonların seçimine çalışmaktadır.

Klasik regresyon yöntemi fonksiyonları karakterize etmek için verilerin dağılımı üzerinde bazı kabullenmeler yapmaktadır. Yapay sinir

ağları ise herhangi bir kabullenme yapmamakla birlikte daha geniş potansiyel fonksiyonlar bulmaya çalışmaktadır (Finnie et al., 1997).

Maliyet kestiriminde kullanılan modeller aşağıda verilmektedir.

Regresyon veya İstatistiksel modeller;

- COCOMO,
- SLİM,
- Estimacs,
- FPA,
- SPANS,
- Checkpoint
- COSTAR

Parametrik Modeller;

- YSA,
- GP

Yukarıdaki modellerin belirli bir ölçev kapsamında kullanılmaları gerekmektedir. Ölçev belirlemede dikkat edilmesi gereken hususlar aşağıda verilmektedir (Ayyıldız, 2006).

Bir yazılım mühendisliği ölçevi ayrık, yalıtılmış olması avantajlıdır.

Süreç, kişi, ürün bazında mümkün olduğunca çok ölçev bulup daha sonra yapılacak bir analiz ile en yararlıların bulunmasının sağlanması faydalıdır.

Ölçevlerin birbirleriyle ilişkili olmaması gerekmektedir. Birbirleri üzerinde etkisi olmamalıdır veya çok az olmalıdır. Bu sürekli izlenmelidir.

Ölçev bulma, çıkarma süreci sistematik ve düzenli olmalıdır. Sürekli iyileştirme yapmak gerekir.

Ölçevlerin doruluğu gerçek hayat ile kontrol edilebilmelidir.

İstatistiksel ve başka yöntemlerle ölçevlerin hata analizleri yapılmalıdır. Böylece gereksiz hatta zararlı ölçevler temizlenmelidir.

Ölçev bazında mantıklı ve doru karşılaştırmalar yapabilmek için kişi, süreç ve ürün benzerlikleri ve farklılıkları karşılaştırmaları iyi bilinmelidir.

Ölçevler zararlı olabilir. Daha dorusu ölçevler yanlış kullanılıyor olabilirler. Buna sürekli dikkat edilmelidir.

Ayyıldız (Ayyıldız, 2006) çalışmasında geniş bir ölçev seti tanımlamıştır. Elde ettiği veriler üzerinde dönüşüm matrisi oluşturmuştur. Matriste bulunan değişkenler arasında korelâsyonlar belirleyip en az etkili olanlarını elemiştir. Matematiksel, istatistiksel ve gözlemsel bir optimizasyon çalışmasından sonra 6 temel grupta ölçevlerini toplamıştır.

Çizelge 3-1'de ölçev setleri verilmektedir.

Çizelge 3-1 Yazılım projelerinde dikkate alınması gereken Ölçev kümesi (Ayyıldız, 2006).

| Ana Ölçev | Ölçev (Metric) |
|---|---|
| İşin Büyüklüğü | 1. Karmaşıklık |

| | |
|---|---|
| (Ürün) | 2.İşlev Puan (Function Point) |
| | 3.Önem |
| | 4. Ayrılan Bütçe |
| | 5. Ürünün beklenen özellikleri |
| Kaynak | 6. Çalışanın yeterlilikleri |
| | 7 Çalışanların Projeye katılım oranları |
| | 8. Çalışan Kişi Sayıları |
| | 9. Donanım Durumu |
| Risk | 10.Bütçe Değişme Riski |
| | 11. Çalışan Riski |
| | 12.Donanım Riski |
| | 13.Ürünün tanımının ve kapsamının değişme riski |
| Teknoloji | 14.Yaz. Geliştirme araçlarının kullanım kolaylığı |
| | 15.Yaz. Geliştirme araçlarının kullanım tecrübesi |
| | 16. Yaz. Geliştirme Araçlarının Kullanımı |
| | 17. Modern Programlama Teknikleri |
| Ortam | 18.Ortamın genel özellikleri |
| | 19.Sahiplenilme<br>(Her bir paydaş türünün projeyi |

| | |
|---|---|
| | sahiplenmesi) |
| | 20.Baskı |
| | 21.Zaman kullanım durumu |
| | 22.Verimlilik durumu |
| Planlar ve Tahminler | 23. Tahmin |
| | 24. Planlar |

Literatürde günümüzde yoğun olarak kullanılan yazılım maliyet kestirim tekniklerini dört grup altında toplamak mümkündür. Bunlar (Heemstra, 1992);

- Uzman veya uzmanlar tarafından yapılan maliyet kestirimi,
- Benzeşim veya benzetme yoluyla çıkarım yapılarak maliyet kestirimi,
- Parametrik modellerle maliyet kestirimi,
- Tümevarım – Tümdengelim maliyet kestirimi.

Uzman veya uzmanlar tarafından yapılan maliyet kestirimi: Gerçekleştirilecek projenin maliyet kestirimi yapacak uzmanın tecrübe alanına ne kadar yakın olduğu maliyet kestiriminin doğru yapılamasına en önemli etkendir. Uzmana dayalı kestirimler objektif olmaktan çok subjektifdir ve birikim kuruma miras olarak kalmaz.

Benzeşim veya benzetme yoluyla çıkarım yapılarak maliyet kestirimi: Bu tekniğin doğru olarak uygulanabilmesi için önceden yapılan projelerin verilerinin ve karakteristik özelliklerinin saklanmış olması gerekmektedir. Küçük ve orta ölçekli projelerde kestirim

sonuçları tatmin edici olabilmektedir. Fakat büyük ölçekli projelerde iyi kestirim mümkün olamamaktadır.

Parametrik modellerle maliyet kestirimi: Proje gerçekleştirim zamanı ve maliyet kestirimi çok parametreli fonksiyonlar yardımı ile belirlemesi hedeflenmektedir. Fonksiyonun değişkenleri proje için kritik olan maliyet parametrelerini ifade etmektedir. Kestirim yönteminin çekirdeği kestirim algoritması ve fonksiyonda kullanılan parametrelerdir. Fonksiyonlarda kullanılan parametrelerin ilk değerleri daha önceki projelerden elde edilen verilerden çıkarılmaktadır.

Tümevarım – Tümdengelim maliyet kestirimi: Tümevarım maliyet kestiriminde geliştirilecek ürünün genel karakteristikleri göz önünde bulundurularak bir maliyet çıkarılır. Sonra toplam maliyet ürünün bileşenleri ortaya çıktıkça onlara uygun bir şekilde paylaştırılır. Tümdengelim maliyet kestiriminde ise geliştirilecek ürünün her bileşeninin maliyeti o bileşeni geliştirecek geliştirici tarafından belirlenerek çıkarılır. Her bir bileşenin maliyeti toplanarak toplam maliyet bulunur.

Yukarıdaki kestirim yöntemlerinin kullanım yüzdeleri ise aşağıdaki gibidir.

Çizelge 3-2 Kullanılan maliyet kestirim modelleri (Heemstra, 1992)

| **Kestirim Yöntemi** | **Kullanım %** |
|---|---|
| Uzman veya uzmanlar tarafından yapılan maliyet kestirimi | 25.5 |
| Benzeşim veya benzetme yoluyla çıkarım yapılarak maliyet kestirimi | 60.8 |

| | |
|---|---|
| Parametrik modellerle maliyet kestirimi | 13.7 |
| Tümevarım – Tümdengelim maliyet kestirimi | 8.9 |
| Diğerleri | 20.8 |

Biz çalışmamızda regresyon temelli ve yapay sinir ağ temelli olmak üzere iki model kullanılmaktadır. Her iki yöntemde istatistiksel öğrenme teorisi üzerine kurulmuştur. Bu neden dolayı İstatistiksel öğrenme teorisi konusu bir sonraki bölümde ele alınmaktadır.

## 3.1 İstatistiksel Öğrenim Teorisi

İstatistiksel yöntemlerle maliyet kestirim fonksiyonunun parametrelerinin belirlenmesi çalışması aslında bir öğrenme problemidir. Bu konuya özgü detaylı çalışmalar Devroye, Vapnik ve Cristianini (Devroye et al., 1996, Vapnik, 1998, Cristianini and Taylor, 2000) tarafından litaratüre kazandırılmıştır. Söz konusu çalışmalarda öğrenim problemleri için bir çerçeve (framework) önerilmektedir. Bu çalışmada söz konusu öğrenme çerçevesi kullanılarak parametrik modelli maliyet kestirim fonksiyonları ve istatistiksel dağılımlara dayalı yöntemler analiz edilecektir.

Maliyet kestirim fonksiyonunu oluşturmak için fonksiyonun bilinmeyen parametrelerinin belirlenmesi gerekmektedir. Buda eldeki örnek verilerle söz konusu maliyet kestirim fonksiyonunun eğitilmesi gerekmektedir. Bir fonksiyonun eğitilmesi (Michael, 2004) aşağıdaki gibi olmaktadır.

F : X → Y eğitilecek fonksiyon olsun. Burada X olası fonksiyon girdileri (bizim durumuzda maliyeti etkileyen unsurlar) olmaktadır ve Y olası fonksiyon çıktıları (maliyet kestirim değerleri) olmaktadır.

Kabuller:

Eğitici örnekleri $(x_i, y_i)$, i = 1, ... m, $x_i \in X$, $y_i \in Y$ olsun.

F = $f(x_i)$ fonksiyonu aday girdileri olan $x_i$'leri uygun $y_i$'lere dönüştürmektedir. Genellenecek olunursa, bir Φ eşlemesi (mapping) her $(x, y) \in X \times Y$ elemanını $\Phi(x, y) \in \Re^n$ özellik vektörüne ilişkilendirmektedir. Kestirilmesi hedeflenen parametre değerleri parametre vector $\Theta \in \Re^n$ ile iade edilmektedir.

Kestirme işleminin bileşenleri F, Φ, Θ, girdi $x_i$'den çıktı $F(x_i)$'e bir eşleme (mapping) tanımlamaktadır.

$$F(x) = \Phi(x, y) \bullet \Theta \qquad (1)$$

$\Phi(x, y) \cdot \Theta$ ifadesi *S* uzayında iç çarpım $\sum_S \Theta_S \Phi_S(x, y)$'ı göstermektedir. Öğrenme işlemi, eğitim örneklerinin kanıt olarak kullanılarak Θ parametre değerlerinin belirlenmesi çalışmasıdır.

Denklem (1) X×Y üzerinde eşleme işlemiyle bir D(x, y) olasılık dağılımı tanımlamaktadır. Bu dağılım eğitim ve test veri girdileri için kullanılmaktadır. Fakat bu dağılım literatürde tanımlı değildir. Her araştırmacı kendi gereksinimlerine göre dağılımlar kullanabilir.

Parametrik modeller eğitim problemini birleşik dağılımlar (joint distribution) D(x,y) veya koşullu dağılımlar ile D(x|y) ile çözmeye çalışmaktadır. Bu çalışmada koşullu dağılımlara dayalı çözümler kullanılmayacaktır.

Birleşik dağılımlarda parametreli olasılık dağılımı P(x,y|Θ) bulunmaktadır. Parametre değerleri Θ değişim gösterdikçe dağılımda değişim gösterecektir (Michael, 2004). Çözüm için olası parametre değerleri olan P(x, y|Θ) için parametre uzayı Ω olsun. P(x, y|Θ) tanımlı bir dağılım olsun (x,y için P(x, y|Θ) = 1).

Parametrik kestirim yaklaşımlarında önemli bir kabul $\Theta^* \in \Omega$ olmasıdır. Diğer bir ifade ile kestirilen parametre vektörünün parametre uzayının bir elemanı olduğu ve D(x, y) = P(x, y|Θ*) olmasıdır. Detaylı olarak açıklanacak olunursa D üzerinde çalışılan dağılımlar kümesinin elamanı olsun. Elimizde bulunan eğitim seti $\{(x_1, y_1), \ldots, (x_m, y_m)\}$, D dağılımı üzerinden eşlensin. Genellikle kullanılan kestirim yönteminde parametreler maximum benzerlik (maximum likelihood) kestirimine göre hesaplanırlar.

Yukarıda yapılan kabule göre bazı $\Theta^* \in \Omega$ için D(x, y) = P(x, y|Θ*) olduğunda birçok dağılımlarda P(x, y|Θ*), eğitim setinin boyu *m* sonsuza yaklaştıkça limitte D(x, y)'e yaklaşır.

Eğitim setinin boyu pratikte sonsuz olamayacağından D(x, y) - P(x, y|Θ*) ≤ MSE olduğunda kestirimin hedefe ulaştığına karar verilecektir. Burada MSE hata ortalamasının karesini ifade etmektedir.

## 3.2 Regresyon Temelli Maliyet Kestirim Modeli

Yukarıda değinildiği üzere lineer modele dayalı maliyet kestiriminde en çok tercih edilen kestirim yöntemi regresyon analizi kullanımıdır (Shepperd and Schofield, 1997). Bu yöntemde aşağıdaki denklemin katsayıları bulunduğunda kestirim fonksiyonu tanımlanmış olur.

$$y = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 + \varepsilon \qquad (2)$$

Denklem (2) de $x_1$ kaynak kod sayısını, $x_2$ uyarlanmış fonksiyon nokta sayısını, ve $x_3$ normalize edilmiş verimlilik geri dönüş oranını ifade etmektedir. y ise adam ay cinsinden tahmini eforu ifade etmektedir. $\alpha_0$, $\alpha_1$, $\alpha_2$, ve $\alpha_3$ kestirilmesi gereken katsayılardır. $\varepsilon$ ise hata terimidir.

Denklem (2) de her iki tarafın doğal logaritması alındığında kestirim fonksiyonu denklem (3)'e dönüşmektedir.

$$y^{\frac{1}{\beta}} = \alpha_0 + \alpha_1 \log x_1 + \alpha_2 \log x_2 + \alpha_3 \log x_3 + \varepsilon \tag{3}$$

Matematiksel olarak denklem (3)'ü genelleştirildiğinde denklem (4)'e dönüşmektedir.

$$y = (\alpha_0 + \sum_{i=1}^{k} \alpha_i \log x_i + \varepsilon)^{\beta} \tag{4}$$

Seçilen veri setinde üç belirleyici veri grubu olduğundan k=3 olmaktadır ve fonksiyonun üstel kuvvet parametresi $\beta$ = 1,2,3,…., 10 şeklinde devam etmektedir. $\beta$ = 1 olduğunda maliyet kestirim fonksiyonu bilinen lineer regresyon formuna dönüşmektedir. Aynı denklem, denklem (5) de gösterildiği şekliylede ifade edilebilir.

$$y = (\alpha_0 + \log(\prod_{i=1}^{k} x_i^{\alpha_i}) + \varepsilon)^{\beta} \tag{5}$$

Tezin deneysel çalışmalar bölümünde denklem 4 üzerinde veri seti kayıtları uygulanarak göreceli hata'nın büyüklüğüne ve $R^2$ bakılmaktadır. Veri setlerinin uygulama sonuçları veri seti analizi bölümünde öngörülen kriterlere göre karşılaştırılmaktadır.

## 3.3 Yapay Sinir Ağ Temelli Maliyet Kestirim Modeli

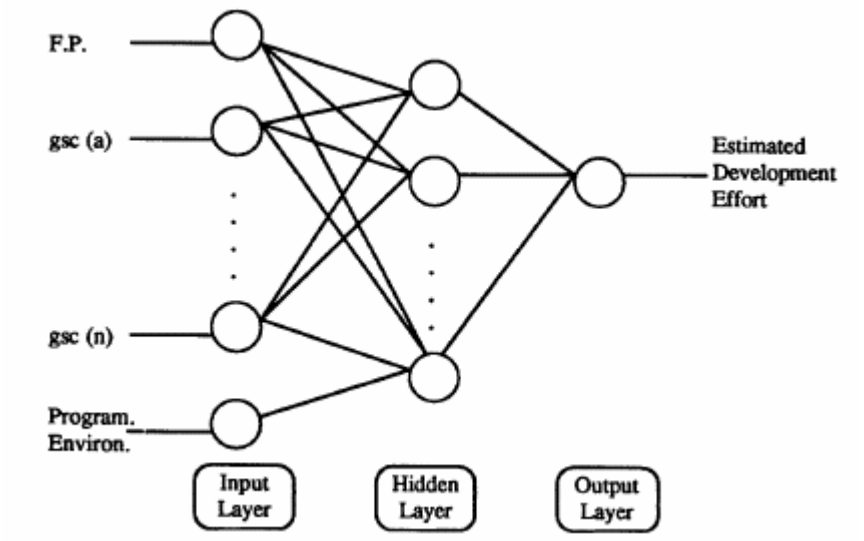
Basit tanımıyla yapay sinir ağları, birçok basit işlemci elemandan oluşan yapılardır. Bu elemanlar farklı formda ifade edilebilen nümerik verileri taşıyan bağlantılar veya ağırlıklar ile birbirlerine bağlıdırlar. Yapay Sinir Ağlarında gelişmelerin ana kaynağı, beynimizin rutin olarak gerçekleştirdiği karmaşık hesaplamaları yapabilen yapay belki zeki davranış sergileyen sistemlerin yapılabileceği ümididir. Yapay zeka ağ yapılarına göre farklı öğrenme yaklaşımı kullanılır ve bu yaklaşımlara göre ağırlıklar değiştirilir. Ağırlıkların değişimi öğrenmeyi ifade eder (Sağıroğlu, 2003).

Yapay Sinir Ağları, tahminleme, sınıflandırma, kümeleme, veri ilişkilendirme ve yorumlama ve veri filtreleme gibi birçok alanda kullanılmaktadır.

Günümüzde parametrik değişkenler kullanılarak belirli fonksiyonlar kontrolünde proje maliyet kestirimi yapılabilmektedir. Bilim ve mühendislik alanlarında Yapay zekâ yöntemleri artan bir sıklıkla kullanılmaya başlanmıştır. Proje maliyet kestiriminde en sık kullanılan parametrik yöntemler yapay sinir ağları yöntemleri ve genetik programlama teknikleri olarak sıralanabilir (Finnie et al., 1997).

Yapay Sinir Ağları modelinde kestirim için bir öğrenme yaklaşımı uygulanmaktadır (Freeman and Skapura, 1992). Yapay sinir ağ'ı belirli bir küme girdi (fonksiyon değişkenleri ve katsayıları) almaktadır. Girdilerin işlenmesinden sonra çıktı olarak (maliyet, geliştirim eforu) eğitilmiş sabit konuma yakınsama yapılacak değerler vermektedir. Başlangıç olarak yapay sinir ağı'na eğitim kümesi verileri girilir. Girilen verilerle belirli bir hata oranına kadar sinir ağ eğitilir. Ağ eğitimi tamamlanarak sabit bir duruma geldikten sonra yeni veriler için

tahminler yapmaya uygun duruma gelmiş olmaktadır. Proje maliyet kestiriminde önceden yapılmış ve hakkında veri tutulmuş projelerin verileri eğitim seti olarak bir yapay sinir ağına verilerek ağ eğitilir. Eğitim tamamlandıktan sonra yeni yapılacak projenin verileri ve parametreleri girdi olarak sinir ağına verilir. Çıkan sonuçlar maliyet kestiriminde kullanılabilecek parametreler olacaktır. Yapay sinir ağları modeli girdi ile çıktı arasında karmaşık sıkı ilişki bulunan ortamlar için oldukça uygundur. Kestirim potansiyeli oldukça iyidir. Fakat günümüzde elde edilen sonuçların yorumlanması kullanıcılar için özellikle proje yöneticilerinin konuya (yapay sinir ağı kavramı) yabancı olmalarından dolayı zor olmaktadır.




Şekil -3-1 Yazılım Geliştirme Maliyet Kestirimi için Yapay Sinir Ağ Mimarisi (Finnie et al., 1997).

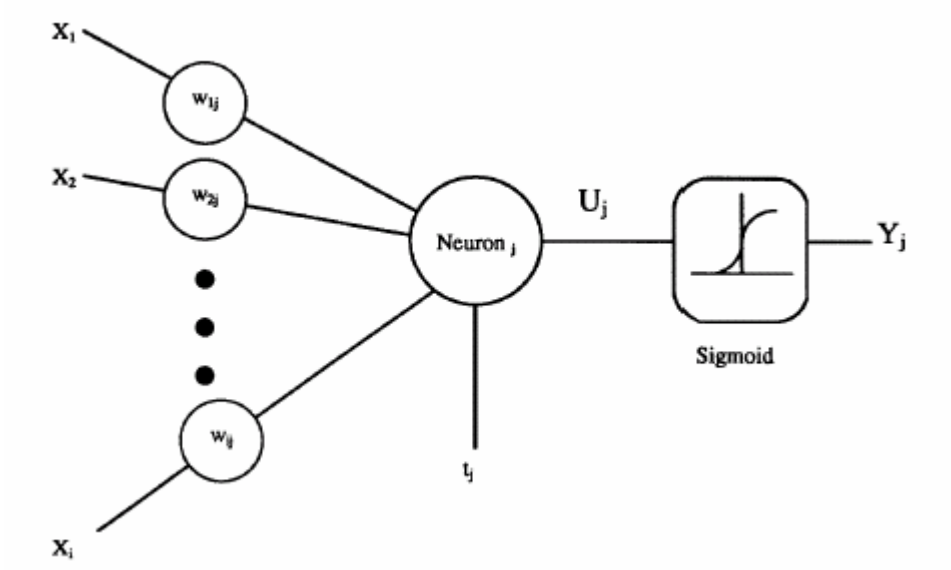
Yazılım geliştirme maliyet kestirimi için yapay sinir ağ uygulamalarından geri-yayılım öğrenme ağları kullanılmaktadır. Bu tür bir ağa örnek Şekil -3-1'de verilmektedir. Bu mimarideki ağlar girdi

olarak geliştirme ortamı ile ilgili parametreler, projenin büyüklüğü gibi genel sistem karakteristikleri verilmektedir. Çıktı olarak tahmini yazılım geliştirme çabası veya maliyet kestirimi edinilmektedir (Finnie et al., 1997).

Bir nörona giren her bir girdi değeri bir ağırlık değeri ile çarpılarak uyarlanır.

Şekil 3-2 'de gösterildiği üzere bu ağırlık değerleri w ij olarak gösterilmektedir. Ağırlıklarla çarpılan girdiler Nöronda toplanır. Referans alınan belirli bir eşik aşım değerine kadar toplama işlemi devam eder. Eşik değeri aşılmadığı durumda geri besleme ile tekrar ağırlıklarla çarpma sürecinden geçilir. Eşik değeri aşımı bir fonksiyonla kontrol edilir.

Şekil 3-2'de eşik fonksiyonu Sigmoid olarak alınmıştır ve yazımı $Y_j = (1 + e^{-U_j})^{-1}$ şeklinde ifade edilmektedir.




Şekil 3-2 Model Nöron (Finnie et al., 1997).

Sigmoid fonksiyonundan başka gauss, lineer ve diğer fonksiyonlar da kullanılabilmektedir.

Şekil 3-2 deki Nöron içerisinde toplama yapan fonksiyon

$$U_j = \sum (x_i \times w_{ij}) - t_j \qquad (6)$$

olarak verilmektedir. Buradaki Yj bir sonraki katmana geçişte girdi olarak kullanılmaktadır (Finnie et al., 1997).

Giren her girdi kümesi için sürekli çıkan tarafla fark kontrol edilmektedir. Aradaki fark hata oranını göstermektedir. Hata oranı belirli bir değerin altına düşene kadar döngü tekrar edilmektedir. Her geri beslemede hesaplamada kullanılan ağırlıklar hata oranı ile güncelleştirilirler. Hata oranı uygun değer seviyeye geldiğinde beklenen kestirim değerine ulaşılmış olunur (Finnie et al., 1997).

Geri-yayılım öğrenme ağları çok çeşitli öğrenim algoritmaları kullanmaına rağmen kestirim yöntemlerinde en sık kullanılan algoritmalar Gradyan bazlı öğrenim algoritmalarıdır (Gradient-based learning algorithms). Bu tez çalışmasında bu sınıf algoritmaların 4 variyasyonu veri seti üzerinde kullanılarak sonuçlar karşılaştırılacaktır. Bu dört çeşit Gradyan bazlı öğrenim algoritmaları aşağıdaki gibidir (Mandic and Chambers, 2001, Matlab, 2002).

- Gradyan azalan geri-yayılım öğrenme ağları (Gradient descent backpropagation),
- Momentumlu Gradyan azalan geri-yayılım öğrenme ağları (Gradient descent with momentum backpropagation),
- Adaptasyon kabiliyetli Gradyan azalan geri-yayılım öğrenme ağları (Gradient descent with adaptive learning rate backpropagation),

- Momentumlu ve adaptasyon kabiliyetli Gradyan azalan geri-yayılım öğrenme ağları (Gradient descent with momentum and adaptive learning rate backpropagation).

Gradyan bazlı öğrenim algoritmaları dışında veri setimizin nasıl davranış sergilediğini belirlemek için ayrıca Levenberg-Marquardt geri-yayılım öğrenme ağı kullanılarak kestirim sonuçları elde edilmiştir. Sonuç bölümünde yapay sinir ağlarından elde edilen sonuçlar karşılaştırılmaktadır. Aşağıda adı geçen yapay sinir ağları uygulamaları Matlab'da gerçeklenmiştir.

### 3.3.1 Gradyan azalan öğrenme ağları

Gradyan azalan geri-yayılım öğrenme ağlarında (Gradient descent backpropagation), gradyan azaldıkça ağın ara katmanındaki ağırlık değerleri ve eğilim yönü güncellenmektedir (Mandic and Chambers, 2001, Matlab, 2002). Matlab'daki "traingd" fonksiyonu ağın öğrenimi için kullanılmaktadır. "traingd" fonksiyonu ağ ara katmanındaki ağırlık değerlerinin, ağ girdilerinin ve transfer fonksiyonlarının türevleri olduğu sürece yakınsama işlemini sürdürmektedir. Bunun için geri besleme yöntemi kullanılmaktadır. Geri besleme yönteminde öğrenme oranı (learning rate, lr) ile öğrenme performansı (training performance, perf) çarpımının ağırlık ve eğilim değişkeni x'e göre türevi alınmaktadır (Matlab, 2002).

$$dx = lr * dperf / dx \tag{7}$$

Ağın eğitimi aşağıdaki koşullardan herhangi birisi oluştuğunda durdurulmaktadır.

- Tekrar sayısının üst sınırına ulaşıldığı zaman (epochs).
- Ağ koşturma zamanının üst sınırına ulaşıldığı zaman.
- Performans, hedeflenen değere ulaştığı zaman.
- Performans gradyanı (yaklaşımı, yakınsaması) belirlenen alt değerin (mingrad) altına düştüğü zaman.
- Sonuç doğrulama performans parametresinin üst hata sayısına (max_fail) ulaştığı zaman.

Başarıyla hedeflenen durma değerlerine ulaşılınca test değerleri ile eğitilen ağ sınanır.

### 3.3.2 Momentumlu Gradyan azalan öğrenme ağları

Momentumlu Gradyan azalan geri-yayılım öğrenme ağlarında (Gradient descent with momentum backpropagation), gradyan azaldıkça ağın ara katmanındaki ağırlık değerleri ve eğilim yönü güncellenmektedir (Mandic and Chambers, 2001, Matlab, 2002). Matlab'daki "traingdm" fonksiyonu ağın öğrenimi için kullanılmaktadır. "traingdm" fonksiyonu ağ ara katmanındaki ağırlık değerlerinin, ağ girdilerinin ve transfer fonksiyonlarının türevleri olduğu sürece yakınsama işlemini sürdürmektedir. Bunun için geri besleme yöntemi kullanılmaktadır. Geri besleme yönteminde öğrenme oranı (learning rate, lr) ile öğrenme performansı (training performance, perf) çarpımının ağırlık ve eğilim değişkeni x'e göre momentumlu (mc) türevi alınmaktadır (Matlab, 2002).

$$dx = mc * dxprev + lr * (1 - mc) * dperf / dx \qquad (8)$$

Ağın eğitimi aşağıdaki koşullardan herhangi birisi oluştuğunda durdurulmaktadır.

- Tekrar sayısının üst sınırına ulaşıldığı zaman (epochs).
- Ağ koşturma zamanının üst sınırına ulaşıldığı zaman.
- Performans hedeflenen değere ulaştığı zaman.
- Performans gradyanı (yaklaşımı, yakınsaması) belirlenen alt değerin (mingrad) altına düştüğü zaman.
- Sonuç doğrulama performans parametresinin üst hata sayısına (max_fail) ulaştığı zaman.

Başarıyla hedeflenen durma değerlerine ulaşılınca test değerleri ile eğitilen ağ sınanır.

### 3.3.3 Adaptasyon kabiliyetli Gradyan azalan öğrenme ağları

Adaptasyon kabiliyetli Gradyan azalan geri-yayılım öğrenme ağlarında (Gradient descent with adaptive learning rate backpropagation), gradyan azaldıkça ağın ara katmanındaki ağırlık değerleri ve eğilim yönü güncellenmektedir (Mandic and Chambers, 2001, Matlab, 2002). Matlab'daki "traingda" fonksiyonu ağın öğrenimi için kullanılmaktadır. "traingda" fonksiyonu ağ ara katmanındaki ağırlık değerlerinin, ağ girdilerinin ve transfer fonksiyonlarının türevleri olduğu sürece yakınsama işlemini sürdürmektedir. Bunun için geri besleme yöntemi kullanılmaktadır. Geri besleme yönteminde öğrenme oranı (learning rate, lr) ile öğrenme performansı (training performance, perf) çarpımının ağırlık ve eğilim değişkeni x'e göre türevi alınmaktadır (Matlab, 2002).

$$dx = lr * dperf / dx \quad (9)$$

Adaptif Gradyan azalan geri yayılım öğrenmenin her evresinde performans parametresi hedef değer doğrultusuna yakınsayacak şekilde azaltılır. Bunun tersine olarak öğrenme oranı arttırılır. Eğer ağın performans maksimum performans artış değerinden fazla artarsa öğrenme parametre oranı düşürülme katsayısı oranında uyarlama yapılır.

Ağın eğitimi aşağıdaki koşullardan herhangi birisi oluştuğunda durdurulmaktadır.

- Tekrar sayısının üst sınırına ulaşıldığı zaman (epochs).
- Ağ koşturma zamanının üst sınırına ulaşıldığı zaman.
- Performans hedeflenen değere ulaştığı zaman.
- Performans gradyanı (yaklaşımı, yakınsaması) belirlenen alt değerin (mingrad) altına düştüğü zaman.
- Sonuç doğrulama performans parametresinin üst hata sayısına (max_fail) ulaştığı zaman.

Başarıyla hedeflenen durma değerlerine ulaşılınca test değerleri ile eğitilen ağ sınanır.

### 3.3.4 Momentumlu ve adaptasyon kabiliyetli Gradyan ağlar

Momentumlu ve adaptasyon kabiliyetli Gradyan azalan geri-yayılım öğrenme ağlarında (Gradient descent with momentum and adaptive leaning rate backpropagation), gradyan azaldıkça ağın ara

katmanındaki ağırlık değerleri ve eğilim yönü güncellenmektedir (Mandic and Chambers, 2001, Matlab, 2002). Matlab'daki "traingdx" fonksiyonu ağın öğrenimi için kullanılmaktadır. "traingdx" fonksiyonu ağ ara katmanındaki ağırlık değerlerinin, ağ girdilerinin ve transfer fonksiyonlarının türevleri olduğu sürece yakınsama işlemini sürdürmektedir. Bunun için geri besleme yöntemi kullanılmaktadır. Geri besleme yönteminde öğrenme oranı (learning rate, lr) ile öğrenme performansı (training performance, perf) çarpımının ağırlık ve eğilim değişkeni x'e göre momentumlu (mc) türevi alınmaktadır (Matlab, 2002).

$$dx = mc * dxprev + lr * mc * dperf / dx \qquad (10)$$

Adaptif Gradyan azalan geri yayılım öğrenmenin her evresinde performans parametresi hedef değer doğrultusuna yakınsayacak şekilde azaltılır. Bunun tersine olarak öğrenme oranı arttırılır. Eğer ağın performans maksimum performans artış değerinden fazla artarsa öğrenme parametre oranı düşürülme katsayısı oranında uyarlama yapılır.

Ağın eğitimi aşağıdaki koşullardan herhangi birisi oluştuğunda durdurulmaktadır.

- Tekrar sayısının üst sınırına ulaşıldığı zaman (epochs).
- Ağ koşturma zamanının üst sınırına ulaşıldığı zaman.
- Performans hedeflenen değere ulaştığı zaman.
- Performans gradyanı (yaklaşımı, yakınsaması) belirlenen alt değerin (mingrad) altına düştüğü zaman.

- Sonuç doğrulama performans parametresinin üst hata sayısına (max_fail) ulaştığı zaman.

Başarıyla hedeflenen durma değerlerine ulaşılınca test değerleri ile eğitilen ağ sınanır.

### 3.3.5 Levenberg-Marquardt öğrenme ağları

Levenberg-Marquardt öğrenim algoritması hızlı öğrenme amaçlı kullanılmakta olup Hessian matrisini hesaplamadan sonuca varmaya çalışmaktadır. Performans fonksiyonu karelerin toplamı (ağın ileri destekli (feedforward) öğrenimi) formunda olursa Hessian matrisi aşağıdaki gibi kestirilebilir (Mandic and Chambers, 2001, Matlab, 2002).

$$H = J^T J \tag{11}$$

ve gradyan (yakınsama eğilimi)

$$g = J^T e \tag{12}$$

şeklinde hesaplanır.

(11) ve (12) denklemlerindeki '*J*' Jacobian matrisi ifade etmektedir. Öğrenim sürecindeki hataların, ağdaki ağırlıklara ve eğime göre türevini ifade eder. Denklem (12) deki 'e' ağdaki hatalar vektörüdür. Jacobian matrisi geri beslemeleri öğrenim tekniğiyle hesaplanabilir.

Bunun için Levenberg-Marquardt geri yayılım öğrenme ağlarında (Levenberg-Marquardt backpropagation), performans perf'in Jacobian jx'isi hesaplanmaktadır. Ağdaki her değişkenin değeri Levenberg-Marquardt kurallarına göre değiştirilir. Matlab'daki "trainlm" fonksiyonu ağın öğrenimi için kullanılmaktadır. "trainlm" fonksiyonu ağ ara katmanındaki ağırlık değerlerinin, ağ girdilerinin ve transfer fonksiyonlarının türevleri olduğu sürece yakınsama işlemini sürdürmektedir (Matlab, 2002).

$$jj = jX * jX \tag{13}$$

$$je = jX * E \tag{14}$$

$$dX = -(jj+I*mu) \setminus je \tag{15}$$

denklem (14) de E parametresi denklemdeki tüm hataları ifade etmektedir. Denklem (15) deki I ise identity matrixini ifade etmektedir.

Denklemdeki adaptasyon değeri "mu" daha geniş ağdaki değişken mu_inc ile arttırılmaktadır. Bu arttırma işlemi performans değerlerine indirgeninceye kadar devam eder. İstenilen 'mu' saviyesi sağlanınca ağdaki ağırlıklar güncellenir ve 'mu' 'mu_dec' kadar azaltılır.

Ağın eğitimi aşağıdaki koşullardan herhangi birisi oluştuğunda durdurulmaktadır.

- Tekrar sayısının üst sınırına ulaşıldığı zaman (epochs).
- Ağ koşturma zamanının üst sınırına ulaşıldığı zaman.
- Performans hedeflenen değere ulaştığı zaman.
- Performans gradyanı (yaklaşımı, yakınsaması) belirlenen alt değerin (mingrad) altına düştüğü zaman.
- 'mu' değeri 'mu_max' değerini aştığı zaman.

- Sonuç doğrulama performans parametresinin üst hata sayısına (max_fail) ulaştığı zaman.

Başarıyla hedeflenen durma değerlerine ulaşılınca test değerleri ile eğitilen ağ sınanır.

## 3.4 Genetik Programlama

Bu bölümde Chang'ın (Chang, 2001) çalışmasından yararlanılmıştır. Genetik Programlama (GP) genetik algoritma tekniğinin bir uzantısı olup 1992 yıllarından itibaren yoğun olarak kullanılmaya başlamıştır. Evrimsel hesaplama (evolutionary computing) tekniği adı verilen ve tekrarlanarak bir döngü içerisinde üretilen aday çözümlerin en çok uyum (fitness) sağlayanının seçilmesine dayanan bir yaklaşımdır. GP bir çok alanda yaygın olarak kullanılmaktadır. Örneğin, devrelerin otomatik olarak sentezlenmesinde, kimyasal süreçlerde lineer olmayan sistemlerin belirlenmesinde, sembolik regresyon ve benzerleri gibi. GP, Proje maliyet kestiriminde ise en iyi uyum sağlayan (fitness) denklemlerin oluşturulmasında otomatik sembolik regresyon yöntemi şeklinde uygulanmaktadır. GP adı biyoloji bilimindeki doğal seçme (natural selection) veya doğal ayırım kavramlarından esinlenmektedir. GP mekanizmasının biyolojideki karşılığı, bir türün genlerinin rasgele mutasyonlar geçirerek çevre koşullarına en iyi uyum sağlayacak duruma geçiş işlemidir (Chang, 2001).

Proje Maliyet kestirim alanında ise temel fikir, girdi ve çıktıları birbirine ilişkilendiren rasgele denklemlerin oluşturulması ve aralarından en iyi uyum sağlayan denklemler üzerinde mutasyon veya çaprazlama yapılarak yeni denklemler üretilmesidir. Denklemler üzerinde mutasyon ve çaprazlama işlemlerine uygun bir çözüm

bulunana kadar devam edilir. Rapor edilen ve karşılaştırılan denklemler son döngüde en iyi uyum sağlayan denklemlerdir. Her uyum döngüsü başta nüfus (population) olarak 25-50 arası rasgele seçilmiş fonksiyon içermektedir. Her döngü sonucunda üretilen denklemlerin sayısı değişmekle birlikte en iyi sonuçlar genellikle 3-5 döngü sonunda alınmaktadır. Bir döngüden diğer döngüye değişmeden geçen denklemlerin sayısı %10 geçmemelidir.

Uygunluk (Fitness) bir denklemin çözüm için ne kadar uygun olduğunu ölçen bir tekniktir. Algoritmanın sonucu veri kümesine en iyi uyumu sağlayan denklemlerin kümesidir. GP yöntemi yazılım büyüklük kestirim problemi ve yazılım maliyet kestirim problemleri üzerine uygulanmaktadır.

Bir GP algoritması aşağıdaki gibidir.

Genetic Programlama Algoritması:

Begin

TekrarSayısı = 0;

Generate initial population P (N boyutunda) of equations;

While not Durmakoşulu do

Evaluate Herbir Denklem için Uyum (Fitness) Hesapla

For each equation in the population P select randomly one of

(a) Mutation with probability Pm

(b) Crossover with probability Pc

(c) Direct reproduction with probability (1- Pm -Pc)

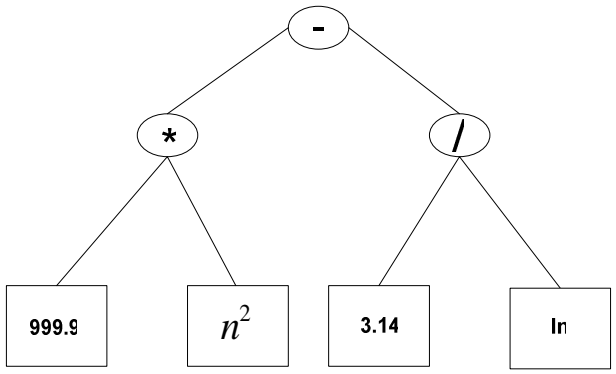
Add the new equation to the new population

endfor

endwhile

end

Çizelge 3-3(999.9 X $n^2$) – (3.14 / ln) denkleminin ağaç yapısında gösterimi (Chang, 2001).




Çizelge 3-3(999.9 X $n^2$) – (3.14 / ln) denkleminin ağaç yapısında gösterimi 'de görüldüğü üzere evrimleşecek unsurlar bir ağaç yapısı üzerinde denklemlerle ilişkilendirilmektedir. Bir algoritmanın nüfusu P (populasyonu) üzerinde mutasyon veya çaprazlama operatörlerinin uygulandığı denklemlerden oluşmaktadır. Ağacın düğüm noktalarında denklemin elemanları bulunmaktadır. Düğümlerde bulunan fonksiyonlar +,-, /, *, kare alma, kara kök alma ve logaritma alma gibi fonksiyonlardan oluşmaktadır.

GP algoritmasının en önemli bileşenlerinden olan Uygunluk (fitness) yöntemi, denklemleri verilere uyarlamak için klasik istatistiksel araçları kullanmaktadır. Uyum ölçümü için iki kriter kullanılmaktadır. Bunlardan birincisi Ortalama Kare Hata (Mean Square error = MSE) değeridir. Diğeri ise korelasyon katsayısıdır. MSE, kestirim yapılan denklemin tahmini hata oranının ölçmektedir.

Korelasyon katsayısı ise tahmin edilen ve geçerli değer arasındaki değişim farkını ölçmektedir. Her iki ölçüm kriteri fonksiyonların uyarlanmasında sık kullanılan araçlardır.

Bir döngüde seçilmiş nüfusu (population) oluşturan denklemlerin tüm parametrelerine (değişkenlere) uygun değerler atandıktan sonraki adım ikinci döngüde kullanılacak olan elemanların belirlenmesi işlemidir. Uyum (fitness) oranlı seçimde, nüfus içerisindeki fonksiyon uyum oranına göre seçim gerçekleştirilir. Seçim işlemi $f_i / F$ değeri dikkate alınarak yapılır. $f_i$ değeri N elemanlı bir nüfusun (populasyon) i'ninci denkleminin sonuç değeridir. F değeri ise N elemanlı bir nüfus için aşağıdaki gibi hesaplanarak denklemlerin sonuçlarının toplam değeridir.

$$F = \sum_{i=1}^{N} f_i \tag{16}$$

Seçilen denkleme çaprazlama veya mutasyon işlemleri uygulanır. Çaprazlama işleminde farklı iki denklemin elemanlarının değiştirilmesini öngörülür. Bu işlem ağaç daki düğüm noktalarının değiştirilmesi anlamına gelmektedir. Bu değişim sonucunda oluşturulan yeni denklemin matematiksel olarak anlamlı olmasına dikkat edilmelidir. +,-,*,/ gibi operatörlerin seçimi tamamen rasgele bir şekilde yapılır. Mutasyon işleminde ise nüfus (populasyon) içerisinden rasgele bir eleman (sabit, değişken, fonksiyon) seçilerek denklemdeki diğer elemanlarla yer değiştirilir.

Bir sonraki döngü yeni oluşturulan denklemlerle birlikte değişmeden bir önceki döngüden gelen denklemlerden oluşmaktadır. Bir önceki döngüden değişmeden gelen denklemlerin sayısı %10 geçmemelidir. Döngü sayısının da 3-5 aralığında tutulması iyi sonuçlar

almak için yeterli görülmektedir. Her uyum döngüsü başta nüfus (populasyon) olarak 20-50 arası rasgele fonksiyon içermesi yararlı olacaktır. GP genetik algoritma elemanları açısından iyi bir C++ kütüphanesi sunmaktadır. Denklemler Matlab gibi veya C++, Java gibi araçlarla kodlanarak gerçeklenebilir.

## 3.5 Sonuçları Değerlendirme Kriterleri

Bir maliyet modelinin doğrulanması için literatürde genellikle kullanılan yöntemlerin başında oluşturulan maliyet fonksiyonunun veri kümesini ne kadar ifade edebildiği göz önünde bulundurulmaktadır. Buna da çoklu korelasyon katsayısı $R^2$ veya uyarlanmış $R^2$ değerlerine bakılarak karar verilmektedir. $R^2$ değeri ne kadar yüksek ise modelin uygulandığı veri kümesini o kadar iyi ifade ettiği anlaşılmaktadır. Ayrıca kestirim modellerinde göreceli hatanın büyüklüğü de modelin başarısını ölçmektedir.

$$MRE = \left| \frac{Maliyet_{Gerçekleşen} - Maliyet_{Kestirilen}}{Maliyet_{Gerçekleşen}} \right| \quad (17)$$

MRE veri setindeki tüm projeler için hesaplanmalıdır. Ayrıca ortalama mutlak göreceli hata değeri, (Mean Absolute Relative Error - MARE) veya göreceli hata büyüklüğünün ortalamasının büyüklüğü (Mean of Magnitude of Relative Error - MMRE), modelin başarısı hakkında araştırıcılara fikir vermektedir. Başarıları karşılaştırılacak olan modellerin sayısı çok ise bu hata belirleme kriterleri kullanılabilir (Finnie et al., 1997).

$$MMRE = \left( \sum_{i=1}^{n} \left| \frac{Maliyet_{Gerçekleşen} - Maliyet_{Kestirilen}}{Maliyet_{Gerçekleşen}} \right| \right) \div N \quad (18)$$

Kestirim modelinin başarısını gösteren bir diğer değerlendirme kıstası tahmin etme seviyesidir. Önceden belirlenen doğruluk seviyesine göre (r) modelin başarısını gösterir.

$$\Pr ed(r) = k / N \quad (19)$$

Burada N toplam gözlem sayısını, k ise MRE'leri r den ufak veya eşit gözlem sayısını ifade etmektedir. Wieczorek (Wieczorek and Ruhe, 2002) ve Boraso (Boraso, 1996) bir kestirim modelin performans değerlerin Pred(0.25) $\geq 75\%$ ve $\overline{MRE} \leq 25\%$ seviyelerinde olması gerekmekte olduğunu belirtmektedir. Sonuç olarak bir kestirim modelinin doğruluk (accuracy) değerleri Pred(0.25) ile doğru orantılı fakat MRE ile ters orantılıdır.

## 3.6 Veri Seti Analizi

Dünyanın en büyük proje veri tabanına sahip ISBSG ile iletişime geçilerek yapılan bir antlaşma ile veriler tarafımıza verilmiştir. Yapılan her çalışma konusunda kendilerinin bilgilendirilmesini istenmektedir.

Önceki çalışmalarımızdan bilindiği üzere Dünyanın en büyük proje veri tabanına sahip ISBSG ile iletişime geçilerek yapılan bir

antlaşma ile veriler tarafımıza verilmiştir. Yapılan her çalışma konusunda kendilerinin bilgilendirilmesini istenmektedir (ISBSG, 2004).

Jorgensen yazılım maliyet kestirim çalışma alanında araştırma yapan araştırıcıların makalelerde "kestirim performansının ölçülmesi" ve "veri seti özellikleri" başlıklı konularda araştırma yapmasını önermektedir (Jorgensen, and Shepperd, 2007). Kendisi maliyet kestirim alanında günümüze kadar yayımlanan birçok makaledeki çalışmalar tarihsel veri setlerine dayalı olduğunu ve gerçek hayattan alınma veri setlerinin çok sınırlı olduğunu ifade etmektedir. Araştırmalarda kullanılacak veri setinin gerçek hayattan alınması durumunda geliştirilen maliyet kestirim modellerinin de gerçek hayattaki projelerde kullanımının mümkün olacağını savunmaktadır.

Bu tez çalışmamızda daha öncede belirtildiği üzere International Software Benchmarking Standards Group (ISBSG, Release 9) veri seti kullanılmıştır (ISBSG, 2004). Dolaysıyla bu çalışma sonucunda çıkan modeller gerçek hayatta kullanılma olanağı olacaktır. ISBSG sürüm 9'da yaklaşık olarak 3.024 adet proje verisi içermektedir. Bu veriler gelişmiş ülkelerdeki önemli kuruluşların gerçek projelerinden alınmış bulunmaktadır. ISBSG veri tabanına yirmi yılı aşkın bir süredir proje verisi kaydedilmektedir. ISBSG veritabanındaki projelerin %70'i son altı yılda gerçekleştirilen projelerin verileridir. ISBSG veri kütüphanesini tekil yapan özelliği de bu olmaktadır.

ISBSG veri kütüphanesindeki projeler çok çeşitli sektörlere ait yazılım projeleri özelliğini taşımaktadır. Finans uygulamalarından gömülü sistemlere kadar projeler çeşitlenmektedir. Ayrıca proje geliştiriminde kullanılan araç gereçler ve gereçler, geliştirme ortamları ve mimariler oldukça çeşitlilik arz etmektedir. Veri kütüphanesindeki projelerin %57'si iyileştirme projelerini oluşturmaktadır. %41 yeni

geliştirilen projeleri kapsamaktadır. %2 ise tekrardan geliştirilen projeleri ifade etmektedir. Çizelge 3-4 de geliştirme sınıflarına göre proje dağılımları görülmektedir.

Çizelge 3-4**.** Dört geliştirme türüne gore ISBSG R9 veri seti dağılım tablosu.

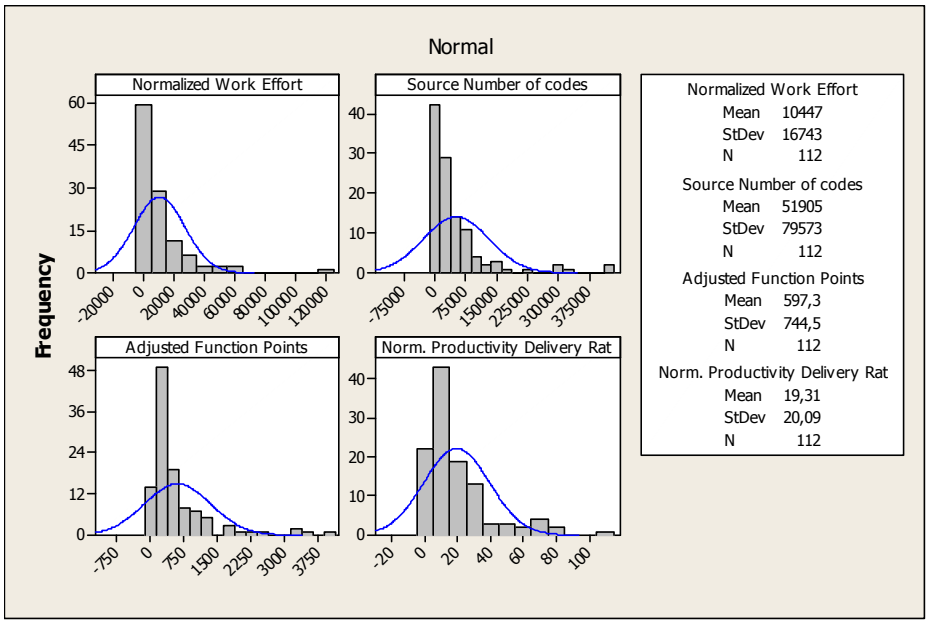
| **Geliştirme Türü** | **Projeler** |
|---|---|
| İyileştirme | 1711 |
| Yeni geliştirilen | 1246 |
| Tekrardan geliştirilen | 65 |
| Diğerleri | 2 |
| Toplam | 3024 |

ISBSG on yılı aşkın süredir proje doğruluğu ispatlanmış veri toplama yöntemleri kullanarak proje verilerini toplamaktadır. Veri kütüphanesinde gerçek hayatta gerçekleştirilip kullanılan projelerin verilerini çok sıkı bir denetleme ve sorgulama sürecinden ve kriterlerinden geçirerek toplamaktadır. Bütün proje verilerinin amacı projelerin maliyetinin adam ay cinsinden belirleme amacıyla kullanılmaktadır.

Veri kütüphanesinde tutulan projelerde tutarlılık derecesine göre sınıflanmaktadır. Bu sınıflama ‘A’ dan ‘D’ ye kadar 4 sınıfı göstermektedir. ‘A’ sınıfı gözlemleri en güvenilir olarak yapılan projelerin verilerini kapsamaktadır. ISBSG proje kütüphanesinde 734 adet ‘A’ sınıfında proje bulunmaktadır. Projeler 50 farklı açıklama parametreleri ile ifade edilmektedir. Bu parametreler proje büyüklüğü, proje zamanı, programlama dilleri gibi yazılım projelerinde etkili olan temel unsurlardan meydana gelmektedir. Veri kütüphanesinde eksik veri içeren proje kayıtları değerlendirmeden çıkarılmıştır. Geriye kalan ‘A’ sınıfı proje kayıtları 115 adet olarak gözlemlenmiştir.

Tez çalışmasında normalize çalışma eforu değeri kestirilen değişken veya bağımlı değişken olarak belirlenmiştir. Her proje gözlem kaydı 4 değerle ifade edilmektedir. Bunlardan 3 adeti bağımsız değişken olup Yazılan Kod Satır Sayısı, Uyarlanmış Fonksiyon Noktaları (Adjusted Function Points), Normalize edilmiş verimlilik oranı gibi unsurlardan oluşmaktadır.

Şeçilen bağımsız parametre değerlerin hiçbiri normal bir dağılıma sahip bulunmamaktadır. Şekil 3-3de bağımsız proje parametrelerin histogram bilgileri görülmektedir.




Şekil 3-3. Maliyet kestiriminde görev alan bağımsız verilerin histogram dağılım şemaları.

Proje maliyet kestiriminde kullanılmak üzere bağımsız değişkenlerin normal dağılıma transformasyonları gerçekleştirilmiştir. Daha sonra her verinin doğal logaritmaları alınarak Regresyon analizinde ve yapay sinir ağlarında kullanılmak üzere hazırlanmıştır.

Bağımsız değişkenler arasında çok düşük bir korelasyon olduğu gözlemlenmiştir. Fakat bunun tersine bağımlı değişkenle bire bir korelasyonlarının çok yüksek olduğu gözlemlenmiştir.

# 4. Önceki Çalışmalar

Yazılım projelerinde maliyet kestirim konusunda literatürde oldukça geniş boyutlu araştırmalar yürütülmektedir. Özellikle regresyon analizi ve yapay sinir ağ yöntemleri ile maliyet kestirimi çok yoğun olarak kullanılmaktadır. Yakın geçmişten itibaren genetik programlama yöntemi ile maliyet kestirimi kullanılmaya başlanmıştır. Bu tez çalışmasında yoğunlukla regresyon analizi ve yapay sinir ağ yaklaşımlarını konu alan araştırmalar üzerinde durulmuştur. Bu şekilde tez kapsamında yapılan çalışmaları, benzer yöntemler kullananlarla daha net olarak karşılaştırma olanağı sağlanmıştır.

Durum bazlı çıkarım, bulanık mantık (fuzzy logic) ve uzman sistemler gibi yapay zeka yöntemleri de kullanılmasına rağmen başarı durumları dikkate alınarak bu bölümde daha az yer verilmiştir. İstatistik temelli yöntemlere alternatif yöntemler kullanımı yazılım maliyet kestiriminde değerlendirilen parametreler arasında farklı ilişkiler bularak daha doğru kestirimler yapmayı amaçlamaktadır (Xu, 2004).

Bu yöntemlerden bazıları, girdiler ile onların çıktılarını birbirleri ile ilişkilendiren maliyet kestirim fonksiyonunun sembolik olarak ifade edilmesini sağlamaktadır. Söz konusu fonksiyonlar maliyet değerlendirmesinde kullanılan parametrelerin davranışlarını doğrudan etkilemektedir. Bu fonksiyonlara maliyet fonksiyonu, ekonomi fonksiyonu veya üretim fonksiyonu adı verilmektedir. Bu fonksiyonlar, proje yöneticilerine proje maliyetinin hesaplanması için sembolik bir ifade etme dili sunarak maliyetlerin kestirimi için veri sunmaktadır (Dolado, 2001).

Modeller iki aşamada sonuca ulaşmaktadır. Bunlar büyüklük tahmini ve verimlilik faktörünün belirlenmesinden oluşmaktadır. Bunlardan büyüklük tahmini aşağıdaki etmenlerden oluşmaktadır.

1. Fonksiyon sayısı,
2. Satır kod sayısı (lines of code),
3. Sınıf sayısı (class),

Verimlilik tahmini ise yazılımı yapan özel kurumun kültürü, müşteri portföyü, geliştirme ortamı, personel tecrübesi, üretim şekli gibi birçok etmene bağımlıdır. Planlanan yazılım kestirilen büyüklükte ne kadar zaman ve çabaya mal olacağı aşağıdaki formül ile bulunabilir (Moser et al., 1999).

Maliyet (adam ay) = Büyüklük X 1/(Verimlilik X Zaman) (20)

Ohlsson (Ohlsson, 1998) tecrübeye dayalı proje maliyet kestirimi üzerinde durmuştur. Önerdiği yöntemlerle tecrübeli maliyet kestirimcilerin önceki projeleri inceleyerek oldukça doğru tahmin yapabileceğini savunmaktadır. Ohlsson çalışmasında önceki projelerden hangi bilgilerin nasıl tutulması konusunda ve bu bilgilerin yorumlanması konusunda önemli yöntemler önermektedir.

Jorgensen (Jorgensen, 2004) bir başka çalışmasında yukarıdan aşağıya veya aşağıdan yukarıya analiz yöntemleri ile yazılım projelerinin maliyetlerinin kestirimi konusunda yöntemler önermektedir. Aşağıdan yukarıya yöntemde projeyi oluşturan bileşenlerin, iş paketlerinin ve alt yüklenicilerin analizi yapılarak maliyet oluşturulmaktadır. Yukarıdan aşağı olan yöntemde ise gereksinim analizinde yola çıkılarak gereklere göre maliyet kestirilmektedir. Son olarak her iki yöntemin sonuçları birleştirilerek tek bir proje maliyet kestirimi ortaya çıkmaktadır.

Maxwell (Maxwell et.al., 1999) proje maliyet kestirimi konusunda Avrupa Uzay ajansındaki 108 projeyi ele almış ve proje maliyet kestiriminde verimlilik konusu üzerinde yoğunlaşmış olup çalışanların verimliliğinin proje takvimi üzerinde etkisi olduğunu öngörmüştür. Çalışmasında projede çalışanlarının verimliğinin hangi unsurlara bağlı olduğunu araştırmıştır.

Godmann (Godmann, 1992) telekomünikasyon alanındaki proje maliyetlerinin kestirilmesinde kullanılacak yöntemler üzerine durmaktadır. İletişim teknolojileri alanında büyük bütçeli yazılım projeleri yapılmaktadır. Proje maliyet kestirimindeki hatalar bu alandaki projelerin geleceğini tehdit etmekte olduğuna değinmekte ve geliştirme iş süreçlerin iyileştirilmesi ile maliyet kestiriminde olumlu sonuçlar alınabileceğini ifade etmektedir.

Jorgensen (Jorgensen and Sjoberg, 2001) bir başka çalışmasında projelerde maliyet kestirimlerinin proje işlerine ne kadar etki ettiğini araştırması oldukça dikkate değerdir. Söz konusu çalışmasında yazar öncelikli ve sıkışık proje takvimine sahip projelerin öngörülen maliyetlerde ve zamanında tamamlandığı fakat proje takvimi geniş zaman aralıklarına göre yapılmış ve gerekleri proje başlangıcında çoğunlukla tanımlanmış olan projelerin bütçe ve zaman konusunda sıkıntıya düştükleri gözlemlenmiştir.

Bashir (Bashir and Thomson, 2001) tasarım projelerinde maliyet kestirimlerinin oldukça kötü olduğunu ifade etmektedir. Bunun nedenin geliştirilen ürünlerin pazara bir an önce çıkarılmak istenmesinden dolayı bir çok geliştirme aşamasının yüzeysel geçildiğini gözlemlemiştir. Özellikle ürün tasarım aşamasının geçiştirilmemesinin gerektiği ve başlayacak projelerin gerekler analizi ve çözüm mimarisinin iyi tanımlandıktan sonra projenin geneli hakkında maliyet

kestiriminde bulunabilineceğini savunmaktadır. Bunun için geliştirilecek ürünün fonksiyonel ayrıştırımının yapılmasına dayalı parametrik maliyet kestirim yöntemi önermektedir.

Moløkken (Moløkken et al., 2004) çalışmasında Norveç de bir çok yazılım projesi geliştiren kurum ve kuruluşların proje yapma yöntemlerini incelemiş olup proje yönetiminde maliyet kestirimi konularında sorunların asıl nedenlerini bulmaya çalışmıştır. Yazılım geliştirme yöntemlerini ve maliyet kestirim yöntemlerinin birbiri ile ilişkisini araştırmıştır. Ayrıca proje yönetiminde alınan dersler konusunda projenin izleme unsurlarının depolanmasını ve diğer projelerde istatistiksel bilgi olarak kullanmanın ülke genelinde yaygınlaştırılma sürecini incelemiştir. İlgili çalışma maliyet kestirimi konusunda çok önemli bir kaynak oluşturan rapordur.

Jorgensen (Jorgensen, 2004) bir organizasyonun Regresyon analizi yöntemi ile projelerinde maliyet kestiriminde sorunların kaynaklarının belirlenmesini hedeflemektedir. Bunun için detaylı iz takibi ve Regresyon analizi kapsamında ortaya çıkan istatistiksel değerlerin karşılaştırılması ile sorunların kaynakları belirlenmeye çalışılmıştır. Örneğin bazı proje iş paketlerinde zaman aşımının kaynağının yanlış tahminden kaynakladığı anlaşılmıştır. Bunun da nedeninin söz konusu iş paketinin maliyet kestirimini geliştirici yerine proje teknik liderinin yapmasından kaynaklandığı belirlenmiştir. Bunun gibi maliyet kestiriminde yapılan hataların kaynakları detaylı bir Regresyon analizi sonucunda belirli bir düzeye kadar belirlenebilir.

Jorgensen (Jorgensen, 2003) çalışmasında Galton'un 1800 sonlarındaki çalışmasını baz alarak çalışmasını başlatmıştır. Söz konusu çalışma ''regression toward the mean'' deyimiyle özetlenebilir. Regresyon yöntemi sonucunda elde edilen verilere göre projenin

neresinde ve hangi fazında yapılan kestirimlerin iyileştirilmesi savunulmaktadır.

Finnie (Finnie, 1997) çalışmasında üç önemli kestirim yöntemini karşılaştırmaktadır. Karşılaştırmada referans alınan unsur ise fonksiyon noktaları olarak ele alınmıştır. 299 proje verisi üzerinde yapılan çalışmada Regresyon, yapay sinir ağları ve durum bazlı çıkarım yöntemleri için standart sapma, göreceli hata oranı değerleri hesaplanarak üç yöntemin başarı durumu karşılaştırılmıştır. Regresyon yönteminin 299 adetlik veri kümesinde başarılı olduğu söylenemez. Fakat yapay sinir ağ ve durum bazlı çıkarım yöntemleri olumlu sonuç almasından dolayı Regresyon yöntemine göre daha başarılı gözükmektedir. Yapay sinir ağ ve durum bazlı çıkarım yöntemleri kullanılarak elde edilen göreceli hata büyüklüğü MMRE Regresyona göre yarı-yarıya çıkmıştır.

Oliveira (Oliveira, 2005) çalışmasındaki maliyet kestiriminde vektör Regresyon ve radial tabanlı fonksiyonlu yapay sinir ağları kullanmıştır. Modellerin eğitiminde NASA'nın veri setini kullanmıştır. Vektör tabanlı Regresyon modelinin performansı yapay sinir ağındakine göre daha yüksek çıkmıştır.

Regresyon analizi ve yapay sinir ağları ile maliyet kestirimi çalışmasının yanında Dolado (Dolado, 2000) genetik programlama yöntemi ile de maliyet kestirim alanında katkılarda bulunmuştur. Çalışmasında PRED(25) doğruluk (accuracy) sınıfında 49'luk bir veri seti için %67 bir seviyeyi üç maliyet kestirim yöntemi için başarmıştır. Ayrıca MMRE de ise birbirine çok yakın değerlerde (Regresyon = 0.28, yapay sinir ağları = 0.32, genetik programlama = 0.29) bir göreceli hata oranı değerini bulmuştur.

Yapay sinir ağları ile maliyet kestiriminde diğer istatistiksel tabanlı maliyet kestirim modellerini Wittig (Wittig and Finnie, 1997) çalışmasında karşılaştırmıştır. Yapay sinir ağı tabanlı maliyet kestirim modeli kestirime etki edecek parametre sayısı artması oranında kestirimin iyileştiğini gözlemlemiştir. Fakat Regresyon tabanlı maliyet kestirim modelinin açık farkla daha başarılı olduğunu savunmaktadır.

Huang (Huang et al., 2006) yapay zeka alanında neuro fuzzy tekniğini kullanarak maliyet kestirimi yapmaya çalışmıştır. Kestirim fonksiyonun bağımlı olduğu parametrelerin birbirine olan etkisini azaltmaya çalışılmıştır. Öğrenim kapasitesi ile önerilen model istikrarlı davranış sergileyerek beklenmedik çıktıların elimine edilmesinde veya belirsizlik oluşturan durumların tespitinde önemli avantajlar sağlamaktadır.

Genetik programlama yöntemleri yeni-yeni maliyet kestiriminde kullanılmaya başladığından bu konuda fazla çalışma bulunmamaktadır. Genetik Programlama yöntemleri ile proje maliyet kestirimi yapmak yapay zeka alanında sinir ağlarından farklı metotlar geliştirilmesine olanak tanımıştır. Başarısı kanıtlanmış bir yöntem bulunmasa da konu araştırılmaya açık bir alandır. Bu konuda bir çok araştırmacı (Chang et al., 2001, Briand et al., 2002, Dolado and Fernandez, 1998) çalışmaktadır.

# 5. Regresyon Yöntemleri ve Yapay Sinir Ağlarına İlişkin Deneysel Çalışmalar

Bu tezin Veri Seti Analizi bölümünde belirlenen ve histogramı oluşturularak normal dağılım değerleri incelenen veri seti deneysel çalışmalarda temel girdi verisi olarak kullanılacaktır. Söz konusu veri seti 115 adet veri kaydından oluşmaktadır. Bu verilerin ilk 100 adeti regresyon fonksiyonlarının oluşturulmasında ve yapay sinir ağı eğitiminde kullanılacaktır. Geriye kalan 15 adeti ise elde edilen regresyon denklemlerinin ve oluşturulan ağların doğrulamasının sağlamasında kullanılacaktır.

## 5.1 Regresyon Analizi

Bu bölümde, bölüm 3.2 de tanımlanan regresyon yöntemlerinden doğrusal regresyon analizi gerçekleştirilerek maliyet kestirim fonksiyonu elde edilecektir. Veri seti analizi bölümünde kabul edildiği üzere çalışma eforu, kot satır sayısı, fonksiyon noktaları ve verimlilik oranları parametrelerine bağlı olarak kestirilmeye çalışılacaktır. Tüm kestirim parametrelerinin p-değerleri 0.01'den küçük çıktığından ve (Delany, 1997) çalışmasında bu üç kestirim değişkenini tavsiye etmektedir. Diğer bir ifade ile bir yazılım sisteminin geliştirilmesinde minimum bu üç parametre ile kestirim yapılabileceğini önermektedir. Benzer olarak Shepperd (Shepperd, and Schofield, 1997) maliyet kestirimi yöntemleri içerisinde en çok tercih edilen yöntemin lineer model olduğunu savunmaktadır. Lineer model regresyon analizi yaparak aşağıda verilen maliyet kestirim fonksiyonunun katsayılarını kestirmektedir.

$$y = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 + \varepsilon \qquad (2)$$

Burada $x_1$ kod satır sayısını, $x_2$ uyarlanmış fonksiyon noktaların sayısını ve $x_3$ normalize verimlilik teslim oranını ifade etmektedir. $y$ değişkeni ise ay-adam cinsinden kestirilen efor değerini ifade etmektedir. Bu çalışmada uzun parametre isimleri ifade edilebilirliği ölçeklenir yapmak için kısaltmaları kullanılacaktır. Değişkenlere ait kısaltmalar Çizelge 5-1 de verilmektedir.

Çizelge 5-1 Veri seti değişken adları

| Değişken Parametre | Kısa Ad | Uzun Ad |
|---|---|---|
| $x_1$ | KSS | Kod Satır Sayısı |
| $x_2$ | UFN | Uyarlanmış Fonksiyon Noktaları |
| $x_3$ | NVO | Normalize Verimlilik Oranı |
| y | NCE | Normalize Çalışma Eforu |

$\alpha_0$, $\alpha_1$, $\alpha_2$, ve $\alpha_3$ katsayıları ise kestirilecek katsayıları ifade etmektedir. $\varepsilon$ değişkeni ise normal dağılıma sahip hata parametresini ifade etmektedir.

Bu tezin Veri Seti Analizi bölümünde belirlenen ve histogramı oluşturularak normal dağılım değerleri incelenen veri seti deneysel çalışmalarda temel girdi verisi olarak kullanılacaktır.

### 5.1.1 Ön İşlemsiz Regresyon Analizi

Ön işlemsiz regresyon analizinde veri setindeki değerlere herhangi bir matematiksel işlem yapılmadığı doğrudan regresyona sokulduğu anlaşılmalıdır.

Bu durumda elde edilen maliyet kestirim fonksiyonu aşağıdaki gibidir.

NCE = - 8684 - 0,0303 KSS + 19,3 UFN + 513 NVO

Kestirilen katsayılar ise sırayla $\alpha_0$ =- 8684 , $\alpha_1$ = - 0,0303 KSS, $\alpha_2$ = 19,3, ve $\alpha_3$ = 513 dır. Sistemin uyarlanmış çoklu korelasyon katsayısı $R^2$ = 69,1% , göreceli hata büyüklüğü MMRE = 2,39 ve doğruluk derecesi PRED(25) = 28% şeklinde elde edilmiştir. α seviyesi 0.05 de p-value değeri (0.000) olarak çıkmıştır. Bu yapılan İstatistiksel işlemin anlamlı olduğunu göstermektedir. $R^2$ = 69,1% değeri denklemi oluşturan değişkenlerin veri seti doğrultusunda iyi kapsamadığını göstermektedir. Diğer bir ifade ile elde edilen denklem veri setini yeteri kadar ifade edememektedir. Ayrıca PRED(25) = 28% düşük bir doğruluk derecesidir.

Yukarıda elde edilen sonuçlar modelimizin doğrulanmasında yetersiz kalmıştır. Dolasıyla kestirim denklemi ve veri setinin iyileştirilmesi gerekmektedir. Bunun için bölüm 3.2 verilen denklem (3)'ün kullanılması modelimizde iyileştirme yapacağı öngörülmektedir.

### 5.1.2 Ön İşlemli Regresyon Analizi

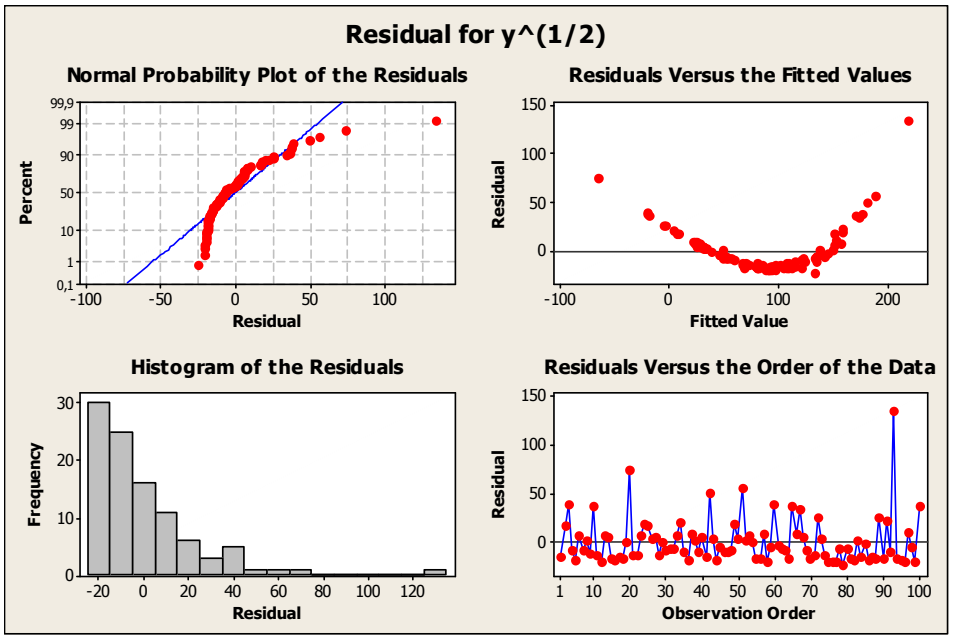
Denklem (3) üç kestirim parametre değerlerinin logaritmalarının alınarak işleme sokulmasını öngörmektedir. Bu durumda veriler daha dar uzay alanına çekilmiş olduğu ve logaritmik fonksiyonların geri dönüşümü olduğu için istenildiği zaman orijinal değerlere geri dönülebilinir.

$$y^{\frac{1}{\beta}} = \alpha_0 + \alpha_1 \log x_1 + \alpha_2 \log x_2 + \alpha_3 \log x_3 + \varepsilon \qquad (3)$$

Denklem (3) deki β üst kuvvetinin değeri arttırılarak regresyon işlemi yapılması durumunda sonuçların modeli nasıl etkileyeceği aşağıdaki çalışmada incelenecektir.

β = 2 için denklem (3)'ün kestirim fonksiyonu aşağıdaki gibi çıkmaktadır.

$$y^{(1/2)} = - 239 - 3{,}15 \log x_1 + 42{,}7 \log x_2 + 41{,}0 \log x_3$$



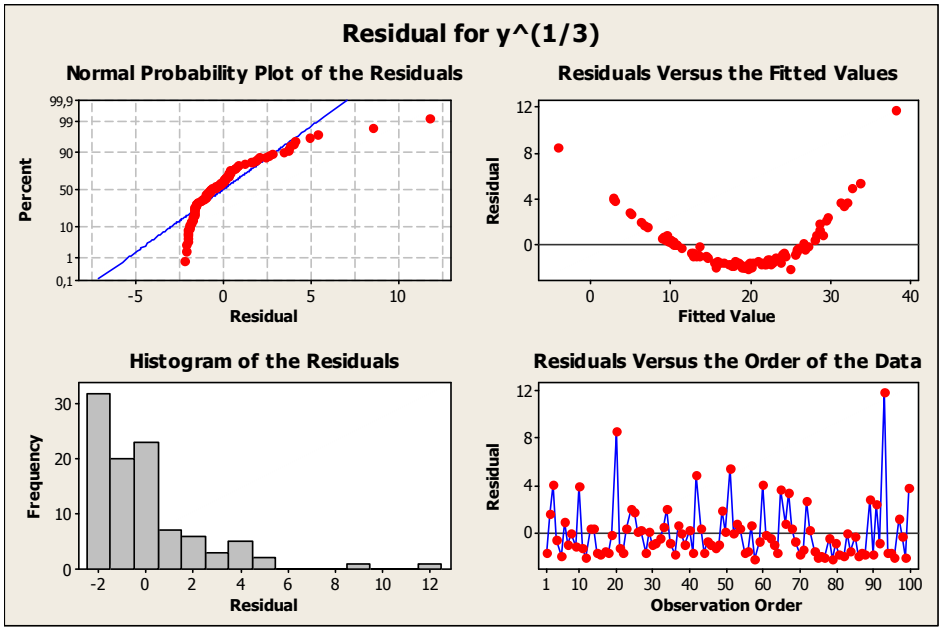

Şekil 5-1 $y^{(1/2)}$ ile kestirilen $\hat{y}^{(1/2)}$ arasındaki farklılıklar.

Regresyon denklemi analiz sonuçlarında kestirilen katsayılar ise sırayla $\alpha_0$ =- 239 , $\alpha_1$ = - 3,15 KSS, $\alpha_2$ = 42,7 ve $\alpha_3$ = 41,0 dır. Sistemin uyarlanmış $R^2$ = 84,1% , MMRE = 0,336 ve PRED(25) = 72% şeklindedir. α seviyesi 0.05 de p-value değeri (0.000) olarak çıkmıştır. Bu yapılan İstatistiksel işlemin anlamlı olduğunu göstermektedir. $R^2$ = 84,1% değeri denklemi oluşturan değişkenlerin veri seti doğrultusunda

iyi kapsadığı yönünde gelişme olduğunu göstermektedir. Diğer bir ifade ile elde edilen denklem veri setini yeteri kadar ifade etmeye başladığının göstergesidir. Ayrıca PRED(25) = 72%, eşik değeri olarak kabul edilen 75% değerine yakınsamıştır. Şekil 5-1 de denklem (3)'ün ikinci dereceden kuvveti alındığında davranış özelliği gözlemlenmektedir.

$\beta = 3$ için denklem (3)'ün kestirim fonksiyonu aşağıdaki gibi çıkmaktadır.

$$y^{(1/3)} = -30{,}2 - 0{,}28 \log x_1 + 6{,}13 \log x_2 + 6{,}02 \log x_3$$




Şekil 5-2 y(1/3) ile kestirilen $\hat{y}$ (1/3)' arasındaki farklılıklar.

Regresyon denklemi analiz sonuçlarında kestirilen katsayılar ise sırayla $\alpha_0$ =-30.22, $\alpha_1$ = - 0,28, $\alpha_2$ = 6,13 ve $\alpha_3$ = 6,02 dır. Sistemin uyarlanmış $R^2$ = 92,4% , MMRE = 0,112 ve PRED(25) = 94%

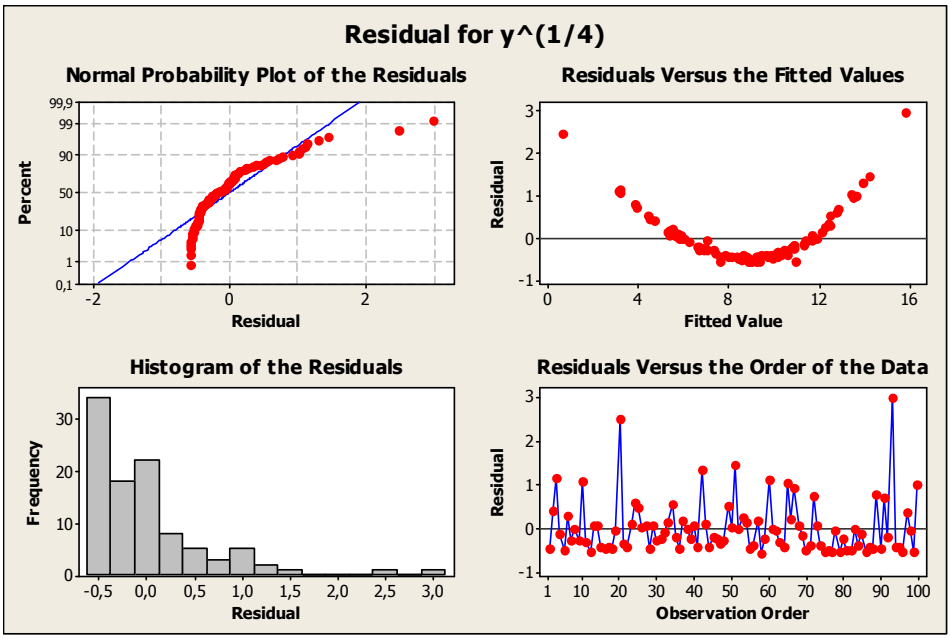
şeklindedir. α seviyesi 0.05 de p-value değeri (0.000) olarak çıkmıştır. Bu yapılan İstatistiksel işlemin anlamlı olduğunu göstermektedir.

$R^2$ = 92,4% değeri denklemi oluşturan değişkenlerin veri seti doğrultusunda iyi kapsadığı yönünde gelişme olduğunu göstermektedir. Diğer bir ifade ile elde edilen denklem veri setini yeteri kadar ifade etmeye başladığının göstergesidir. Ayrıca PRED(25) = 94%, eşik değeri olarak kabul edilen 75% değerini geçmiştir.

Şekil 5-2 de denklem (3)'ün üçüncü dereceden kuvveti alındığında davranış özelliği gözlemlenmektedir.

β = 4 için denklem (3)'ün kestirim fonksiyonu aşağıdaki gibi çıkmaktadır.

$$y^{(1/4)} = - 8{,}87 - 0{,}0711 \log x_1 + 2{,}18 \log x_2 + 2{,}16 \log x_3$$




Şekil 5-3 y(1/4) ile kestirilen ŷ (1/4)' arasındaki farklılıklar.

Regresyon denklemi analiz sonuçlarında kestirilen katsayılar ise sırayla $\alpha_0$ =-8.87, $\alpha_1$ = - 0,071, $\alpha_2$ = 2,18 ve $\alpha_3$ = 2,16 dır. Sistemin uyarlanmış $R^2$ = 95,6% , MMRE = 0,057 ve PRED(25) = 96% şeklindedir. α seviyesi 0.05 de p-value değeri (0.000) olarak çıkmıştır. Bu yapılan İstatistiksel işlemin anlamlı olduğunu göstermektedir.

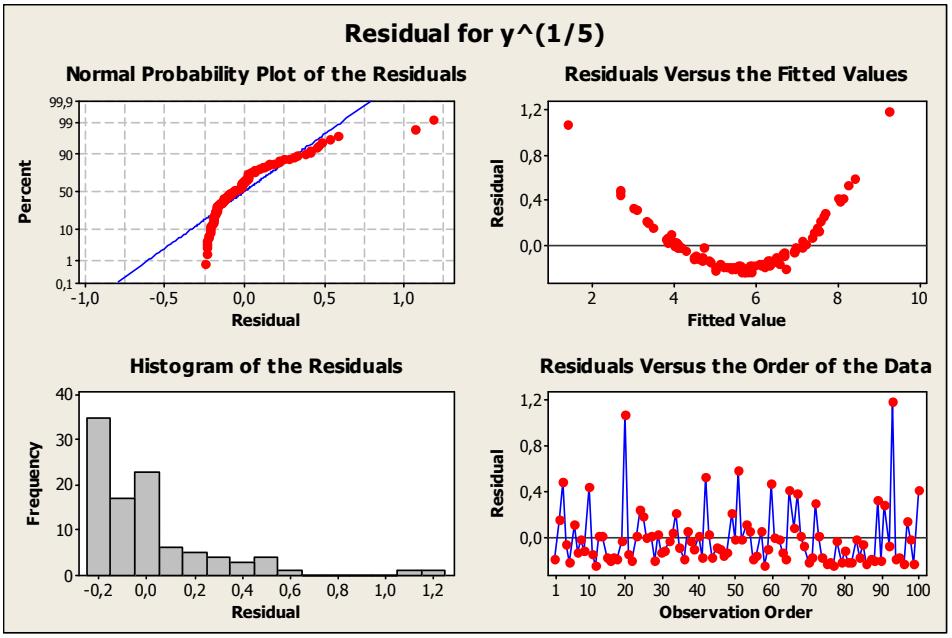
$R^2$ = 95,6% değeri denklemi oluşturan değişkenlerin veri seti doğrultusunda iyi kapsadığı yönünde gelişme olduğunu göstermektedir. Diğer bir ifade ile elde edilen denklem veri setini yeteri kadar ifade etmeye başladığının göstergesidir. Ayrıca PRED(25) = 96%, eşik değeri olarak kabul edilen 75% değerini geçmiştir.

Şekil 5-23 te denklem (3)'ün dördüncü dereceden kuvveti alındığında davranış özelliği gözlemlenmektedir.

β = 5 için denklem (3)'ün kestirim fonksiyonu aşağıdaki gibi çıkmaktadır.

$$y^{(1/5)} = - 3{,}53 - 0{,}028 \log x_1 + 1{,}12 \log x_2 + 1{,}11 \log x_3$$

Regresyon denklemi analiz sonuçlarında kestirilen katsayılar ise sırayla $\alpha_0$ =-3.53, $\alpha_1$ = - 0,028, $\alpha_2$ = 1,12 ve $\alpha_3$ = 1,11 dır. Sistemin uyarlanmış $R^2$ = 97,1% , MMRE = 0,035 ve PRED(25) = 99% şeklindedir. α seviyesi 0.05 de p-value değeri (0.000) olarak çıkmıştır. Bu yapılan İstatistiksel işlemin anlamlı olduğunu göstermektedir.




Şekil 5-4 y(1/5) ile kestirilen ŷ (1/5)' arasındaki farklılıklar.

$R^2$ = 97,1% değeri denklemi oluşturan değişkenlerin veri seti doğrultusunda iyi kapsadığı yönünde gelişme olduğunu göstermektedir. Diğer bir ifade ile elde edilen denklem veri setini yeteri kadar ifade etmeye başladığının göstergesidir. Ayrıca PRED(25) = 99%, eşik değeri olarak kabul edilen 75% değerini fazlasıyla geçmiştir.

Şekil 5-4 denklem (3)'ün besinci dereceden kuvveti alındığında davranış özelliği gözlemlenmektedir.

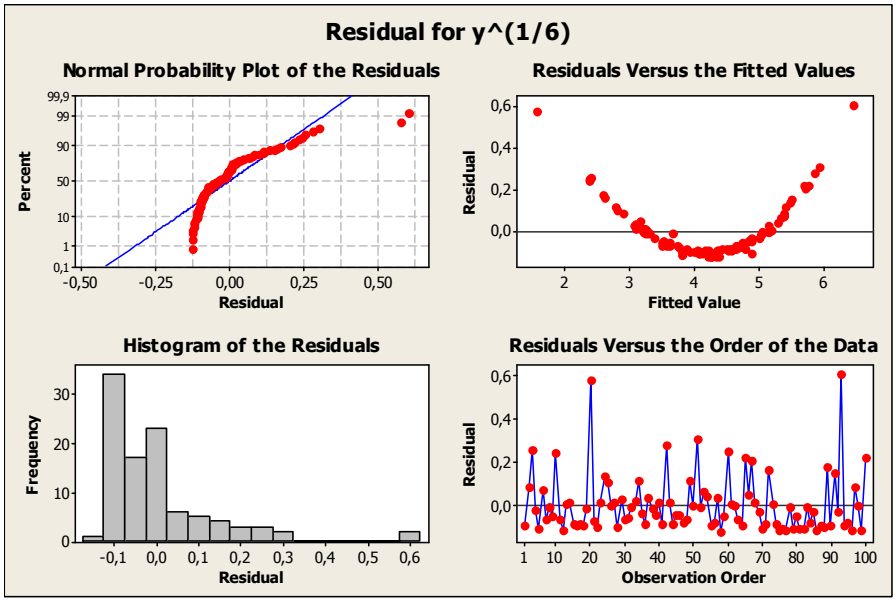
β = 6 için denklem (3)'ün kestirim fonksiyonu aşağıdaki gibi çıkmaktadır.

$$y^{(1/6)} = - 1{,}53 - 0{,}0142 \log x_1 + 0{,}696 \log x_2 + 0{,}694 \log x_3$$

Regresyon denklemi analiz sonuçlarında kestirilen katsayılar ise sırayla $\alpha_0$ =- 1,53, $\alpha 1$ = - 0,0142, $\alpha_2$ = 0,696 ve $\alpha_3$ = 0,694 dır. Sistemin uyarlanmış $R^2$ = 98% , MMRE = 0,0237 ve PRED(25) = 99%

şeklindedir. α seviyesi 0.05 de p-value değeri (0.000) olarak çıkmıştır. Bu yapılan İstatistiksel işlemin anlamlı olduğunu göstermektedir. $R^2$ = 98% değeri denklemi oluşturan değişkenlerin veri seti doğrultusunda iyi kapsadığı yönünde gelişme olduğunu göstermektedir. Diğer bir ifade ile elde edilen denklem veri setini yeteri kadar ifade etmeye başladığının göstergesidir. Ayrıca PRED(25) = 99%, eşik değeri olarak kabul edilen 75% değerini fazlasıyla geçmiştir.

Şekil 5-5 de denklem (3)'ün altıncı dereceden kuvveti alındığında davranış özelliği gözlemlenmektedir.



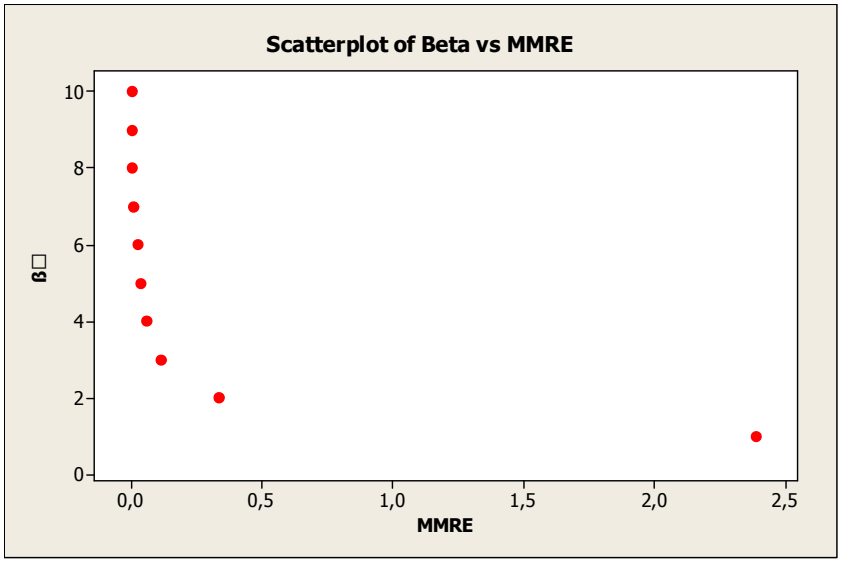

Şekil 5-5 y(1/6) ile kestirilen ŷ (1/6)' arasındaki farklılıklar.

Çizelge 5-2 Farklı üst kuvvetlerine göre önerilen maliyet kestirim fonksiyonunun sonuç değerleri

| $\beta$ | **Katsayılar** $\alpha_0, \alpha_1, \alpha_2, \alpha_3$ | **MMRE** | **$R^2$– adjusted** | **PRED( 0,25)** |
|---|---|---|---|---|
| 1 | - 8684 , - 0.0303, 19.3, 513 | 2,39 | 69,1% | 28% |
| 2 | -239, -3.15, 42.7, 41.0 | 0,336 | 84,1% | 72% |
| 3 | -30.2, -0.28, 6.13, 6.02 | 0,112 | 92,4% | 94% |
| 4 | -8.87, -0.071, 2.18, 2.16 | 0,057 | 95,6% | 96% |
| 5 | -3.53, -0.028, 1.12, 1.11 | 0,035 | 97,1% | 99% |
| 6 | -1.53, -0.0142, 0.696, 0.694 | 0,023 | 98% | 99% |
| 7 | -0.616, -0.0128, 0.494, 0.487 | 0,0064 | 98,6% | >=99,1 % |
| 8 | -0.107, –0.00829, 0.369, 0.365 | 0,0027 | 98,9% | >=99,1 % |
| 9 | 0.198, –0.00573, 0.290, 0.288 | 0,0013 | 99,1% | >=99,1 % |
| 10 | 0.393,–0.00416, 0.237, 0.235 | 0,0007 | 99,3% | >=99,1 % |

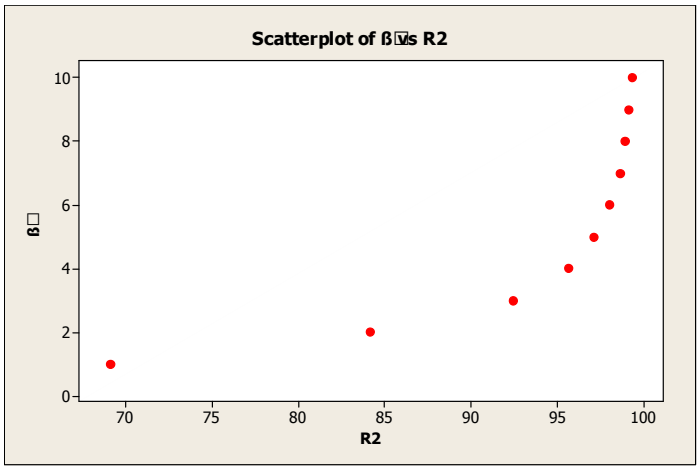
Yukarıda yapılan çalışmalar özetlenecek olursa ISBSG'den elde edilen veri setine lineer regresyon uygulanmış olup istatistiksel performans sonuçları alınmıştır. Çizelge 5-2'nin birinci satırından görüleceği üzere yüksek performans alınamamıştır. Bu telafi etmek için veri seti ve lineer kestirim fonksiyonu üzerinde matematiksel işlemler yapılmıştır. Bu işlemler sonucunda MMRE'de düşüş ve $R^2$ de yükseliş olmuştur. Ayrıca doğruluk (accuracy) derecesinin arttığı gözlemlenmiştir.

Şekil 5-6'de göreceli hata boyutunun kestirim fonksiyonun kuvvet derecesi artması oranında nasıl düştüğü görülmektedir. Bu yeni fonksiyonumuzun (denklem (3)) hızlı bir şekilde kestirim konusunda eğitildiğini göstermektedir.

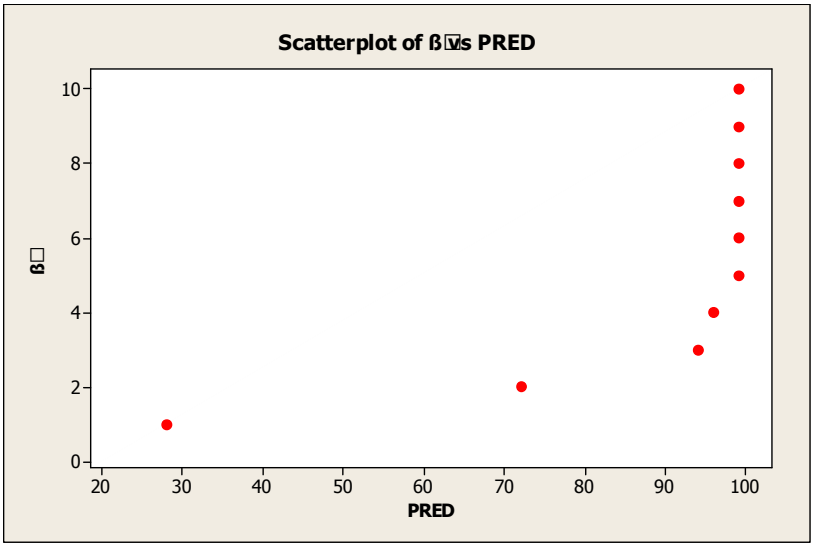



Şekil 5-6 Denklem (3) 'ün kuvvet derecesine göre hata büyüklüğünün değişimi

Şekil 5-7 de $R^2$ değeri denklem (3)'ün kuvvet derecesi arttıkça artmıştır. Diğer bir ifade ile denklemin veri setini ifade edilebilirliği veya kapsama kapasitesi dolayısıyla artmıştır.




Şekil 5-7 Denklem (3) 'ün kuvvet derecesine göre uyarlanmış $R^2$ değişimi




Şekil 5-8 Denklem (3) 'ün kuvvet derecesine göre doğruluk (accuracy) varyans hata büyüklüğünün değişimi

Şekil 5-6 de PRED (accuracy-doğruluk) değeri denklem (3)'ün kuvvet derecesi arttıkça artmıştır. Diğer bir ifade ile denklemin veri setini ifade edilebilirliği veya kapsama kapasitesi dolayısıyla artmıştır.

Önerilen bu modelin doğruluk derecesi yüksek çıkmıştır. Gerçekleştirilen örnekler gerçek hayatta kullanılabilirler.

## 5.2 Yapay Sinir Ağ Yaklaşımı

Yapay sinir ağ yaklaşımında beş ayrı eğitim algoritması kullanılmıştır. Herbir yapay sinir ağ yaklaşımı için Matlab'da kod yazılmıştır. İlgili kodlar aşağıda verilmektedir.

### 5.2.1 Gradyan azalan öğrenme ağları

Gradyan azalan geri-yayılım öğrenme ağı hızlı yakınsamasıyla bilenen bir yapay sinir ağı eğitim yöntemidir. Bu yöntemde ağ parametrelerinin deneme yanılma yöntemi ile en makul parametreleri belirlenmesi gerekmektedir.

Yapay sinir ağını oluşturmak, eğitmek ve test ederek başarı oranlarını bulmak için yazılan Matlab kodu aşağıda verilmektedir:

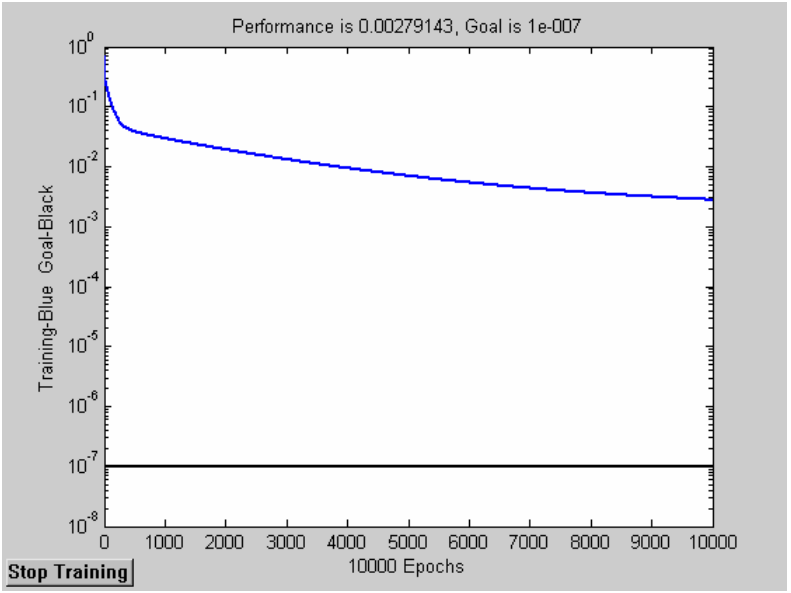
```
1. pred = 0; MMRE = 0; mse = 0; gtmp = 0; y=0; yp=0;
2. input=[data(:,2)'; data(:,3)';data(:,4)'; ];
3. target=[data(:,1)'];
4. [input,minp,maxp,target,mint,maxt] =
   premnmx(input,target);
5. net=newff(minmax(input),[10,1],{'logsig','purelin'},'train
   gd');
6. net.trainParam.goal=1e-7;
7. net.trainParam.epochs=10000;
8. [net,tr]=train(net,input,target);
9. for i = 1 : 15
```

```
10. y = testdata(i,2:4);
11. y=tramnmx(y',minp,maxp);
12. output = sim(net,y);
13. yp =    postmnmx(output,mint,maxt);
14. mse = abs((testdata(i,1)-yp)./testdata(i,1));
15. if mse <= 0.25
16. pred = pred + 1;
17. end;
18. gtmp = gtmp + mse;
19. end;
20. MMRE = gtmp / 15;
21. disp('pred='); disp(pred/15);
22. disp('MMRE='); disp(MMRE);
23. end;
```

Gradyan azalan geri-yayılım öğrenme ağının öğrenim sonuçları aşağıdaki gibi çıkmıştır.

Çizelge 5-3 Gradyan azalan geri-yayılım öğrenme ağının öğrenim sonuçları

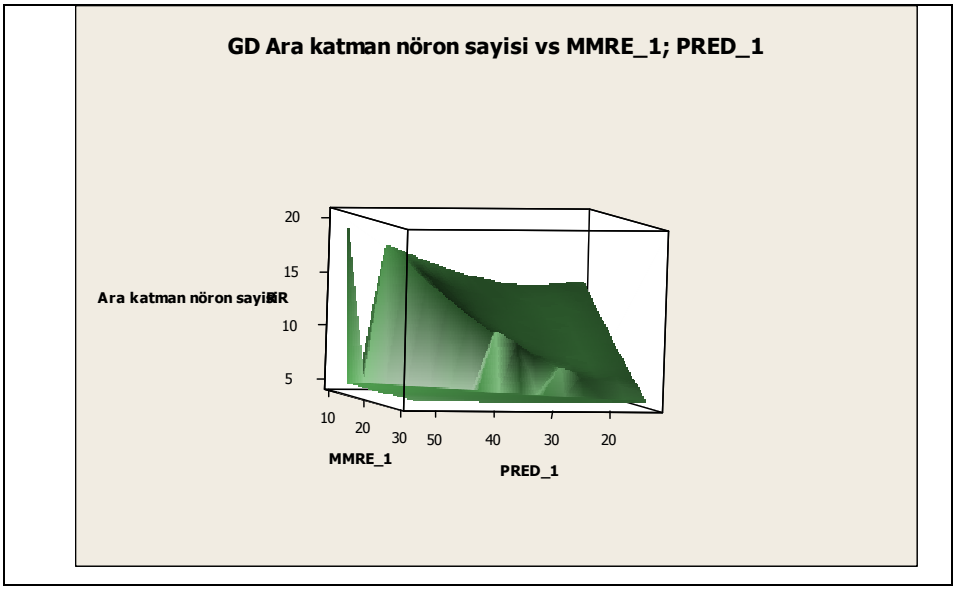
| Ara katman nöron sayısı | MMRE | PRED |
|---|---|---|
| 5 | 29,54 | 13,3 |
| 10 | 17,67 | 26,7 |
| 15 | 10,57 | 13,3 |
| 20 | 10.07 | 53,3 |




Şekil 5-9 Gradyan azalan geri-yayılım öğrenme ağının yakınsama durumu.

Şekil 5-9 da görüldüğü üzere Gradyan azalan geri-yayılım öğrenme ağı öğrenim sonucunda istenilen performans hedefine ulaşamamıştır. Bunun nedeni Şekil 5-10 da görülmektedir. Gradyan azalan geri-yayılım öğrenme ağında ara katmanda bulunan nöron sayısı azaldıkça eğitilen ağın maliyet kestirim denkleminin veri setini iyi ifade etmediği görülmektedir. Ağın performans doğruluk oranı nöron sayısı azaldıkça %95'lere asla çıkmadığı görülmektedir. Ancak %13'lerde seyretmektedir. Buna karşılık ağın göreceli hata değeri (MMRE) hızlı bir şekilde artmaktadır. Burada dikkati çeken bir başka durum ise ara katmandaki nöron sayısı 5'in altına düşünce kestirim başarısı çok kötüleşmektedir. Diğer bir ifade ile Gradyan azalan geri-yayılım öğrenme ağı 5'ten daha az bir nöron sayısını veri setimiz için kullanamayacağı görülmektedir. Ayrıca Gradyan azalan geri-yayılım öğrenme ağında ara katmandaki nöron sayısı 20'den fazla olması durumunda kestirim fonksiyonunun performans doğruluk oranı

Pred(25) başarı değerini arttırmadığı (~%50) ve ağın göreceli hata değerinin (MMRE = ~10.01) daha fazla düşmediği gözlemlenmektedir. Sonuç olarak Gradyan azalan geri-yayılım öğrenme ağı bizim veri setimizle projelerde maliyet kestiriminde kullanılamaz durumdadır.




Şekil 5-10 Gradyan azalan geri-yayılım öğrenme ağı yakınsama grafiği

### 5.2.2 Momentumlu Gradyan azalan öğrenme ağları

Momentumlu gradyan azalan geri-yayılım öğrenme ağı hızlı yakınsamasıyla bilenen bir yapay sinir ağı eğitim yöntemidir. Bu yöntemde ağ parametrelerinin deneme yanılma yöntemi ile en makul parametreleri belirlenmesi gerekmektedir.

Yapay sinir ağını oluşturmak, eğitmek ve test ederek başarı oranlarını bulmak için yazılan Matlab kodu aşağıdadır:

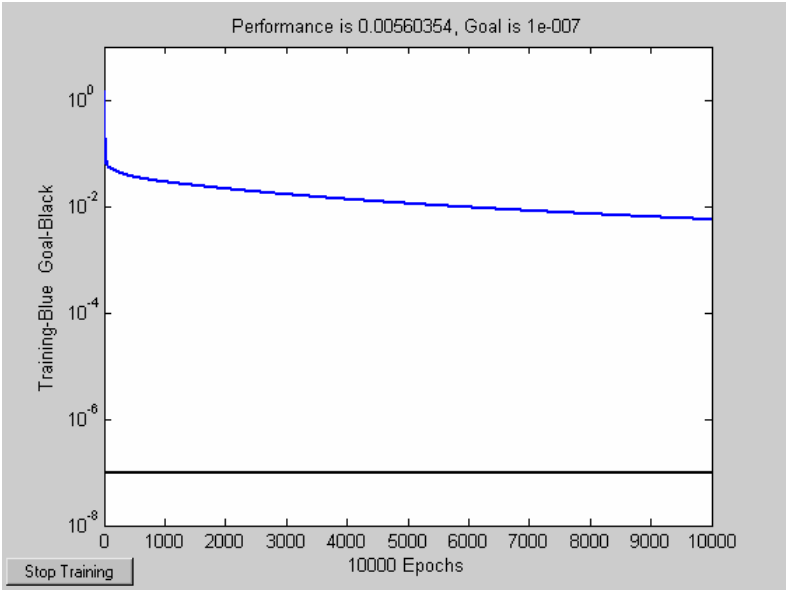
```
1. pred = 0; MMRE = 0; mse = 0; gtmp = 0; y=0; yp=0;
2. input=[data(:,2)'; data(:,3)';data(:,4)'; ];
3. target=[data(:,1)'];
4. [input,minp,maxp,target,mint,maxt] = premnmx(input,target);
5. net=newff(minmax(input),[10,1],{'logsig','purelin'},'traingdm');
6. net.trainParam.goal=1e-7;
7. net.trainParam.epochs=10000;
8. [net,tr]=train(net,input,target);
9. for i = 1 : 15
10. y = testdata(i,2:4);
11. y=tramnmx(y',minp,maxp);
12. output = sim(net,y);
13. yp =  postmnmx(output,mint,maxt);
14. mse = abs((testdata(i,1)-yp)./testdata(i,1));
15. if mse <= 0.25
16. pred = pred + 1;
17. end;
18. gtmp = gtmp + mse;
19. end;
```

```
20. MMRE = gtmp / 15;
21. disp('pred='); disp(pred/15);
22. disp('MMRE='); disp(MMRE);
23. end;
```

Momentumlu gradyan azalan geri-yayılım öğrenme ağının öğrenim sonuçları aşağıdaki gibi çıkmıştır.

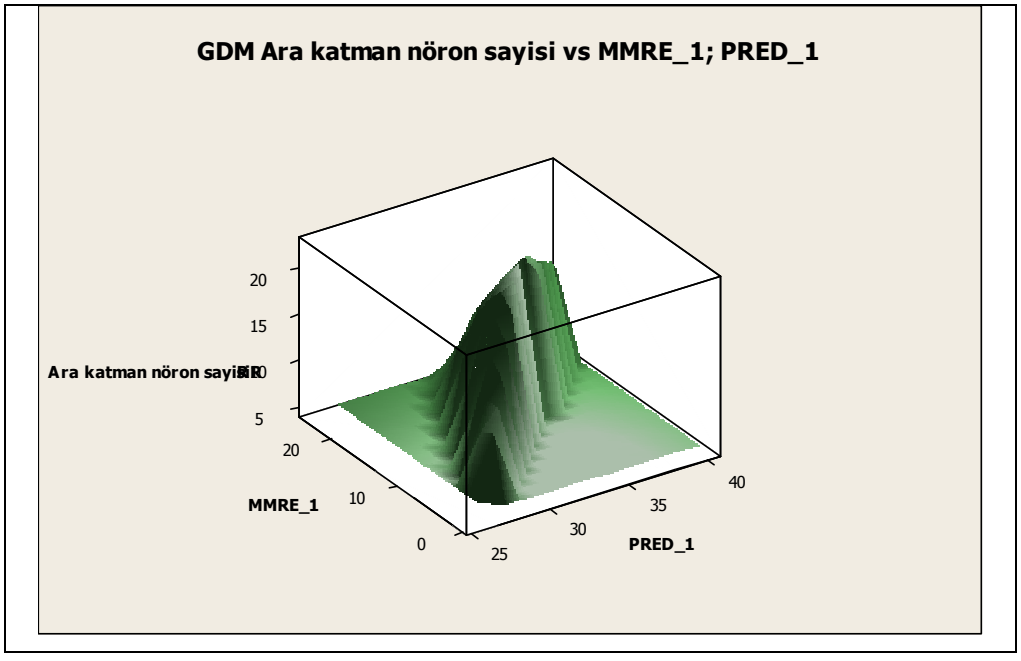
Çizelge 5-4 Momentumlu gradyan azalan geri-yayılım öğrenme ağının öğrenim sonuçları

| Ara katman nöron sayısı | MMRE | PRED |
|---|---|---|
| 5 | 23,51 | 33,3 |
| 10 | 0,79 | 26,67 |
| 15 | 20,91 | 40 |
| 20 | 14,08 | 33,33 |




Şekil 5-11 Momentumlu gradyan azalan geri-yayılım öğrenme ağının yakınsama durumu.

Şekil 5-11'de görüldüğü üzere Momentumlu gradyan azalan geri-yayılım öğrenme ağı öğrenim sonucunda istenilen performans hedefine ulaşamamıştır. Bunun nedeni Şekil 5-12 da görülmektedir. Momentumlu gradyan azalan geri-yayılım öğrenme ağında ara katmanda bulunan nöron sayısı azaldıkça eğitilen ağın maliyet kestirim denkleminin veri setini iyi ifade etmediği görülmektedir. Ağın performans doğruluk oranı nöron sayısı azaldıkça veya çoğaldıkça %30'ları asla geçemediği görülmektedir. Buna karşılık ağın göreceli hata değeri (MMRE) hızlı bir şekilde artmaktadır. Yine burada da bir önceki bölümde olduğu gibi ara katmandaki nöron sayısı 5'in altına düşünce kestirim başarısı çok kötüleşmektedir. Diğer bir ifade ile Momentumlu gradyan azalan geri-yayılım öğrenme ağı 5'ten daha az bir nöron sayısını veri setimiz için kullanamayacağı görülmektedir.




Şekil 5-12 Momentumlu gradyan azalan geri-yayılım öğrenme ağı yakınsama grafiği

Ayrıca Momentumlu gradyan azalan geri-yayılım öğrenme ağında ara katmandaki nöron sayısı 20'den fazla olması durumunda kestirim fonksiyonunun performans doğruluk oranı Pred(25) başarı değerini arttırmadığı (~%33) ve ağın göreceli hata değerinin (MMRE = ~23.0) daha fazla düşmediği gözlemlenmektedir. Sonuç olarak Momentumlu gradyan azalan geri-yayılım öğrenme ağı bizim veri setimizle projelerde maliyet kestiriminde kullanılamaz durumdadır.

### 5.2.3 Adaptasyon kabiliyetli Gradyan öğrenme ağları

Adaptasyon gradyan azalan geri-yayılım öğrenme ağı hızlı yakınsamasıyla bilenen bir yapay sinir ağı eğitim yöntemidir. Bu yöntemde ağ parametrelerinin deneme yanılma yöntemi ile en makul parametreleri belirlenmesi gerekmektedir.

Yapay sinir ağını oluşturmak, eğitmek ve test ederek başarı oranlarını bulmak için yazılan Matlab kodu aşağıda verilmektedir:

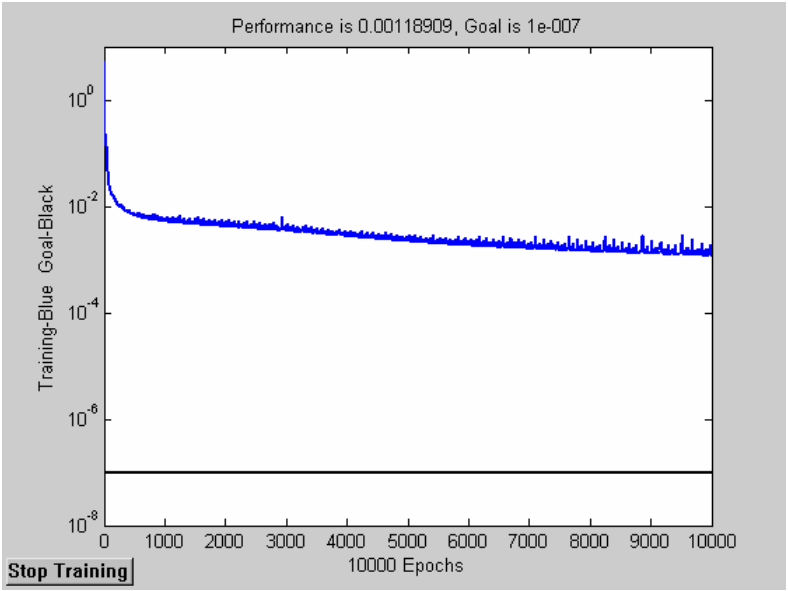
```
1. pred = 0; MMRE = 0; mse = 0; gtmp = 0; y=0; yp=0;
2. input=[data(:,2)'; data(:,3)';data(:,4)'; ];
3. target=[data(:,1)'];
4. [input,minp,maxp,target,mint,maxt]=
   premnmx(input,target);
5. net=newff(minmax(input),[10,1],{'logsig','purelin'},'train
   gda');
6. net.trainParam.goal=1e-7;
7. net.trainParam.epochs=10000;
8. [net,tr]=train(net,input,target);
```

```
9.  for i = 1 : 15
10. y = testdata(i,2:4);
11. y=tramnmx(y',minp,maxp);
12. output = sim(net,y);
13. yp =    postmnmx(output,mint,maxt);
14. mse = abs((testdata(i,1)-yp)./testdata(i,1));
15. if mse <= 0.25
16. pred = pred + 1;
17. end;
18. gtmp = gtmp + mse;
19. end;
20. MMRE = gtmp / 15;
21. disp('pred='); disp(pred/15);
22. disp('MMRE='); disp(MMRE);
23. end;
```

Adaptasyon kabiliyetli gradyan azalan geri-yayılım öğrenme ağının öğrenim sonuçları aşağıdaki gibi çıkmıştır.

Çizelge 5-5 Adaptasyon kabiliyetli gradyan azalan geri-yayılım öğrenme ağının öğrenim sonuçları

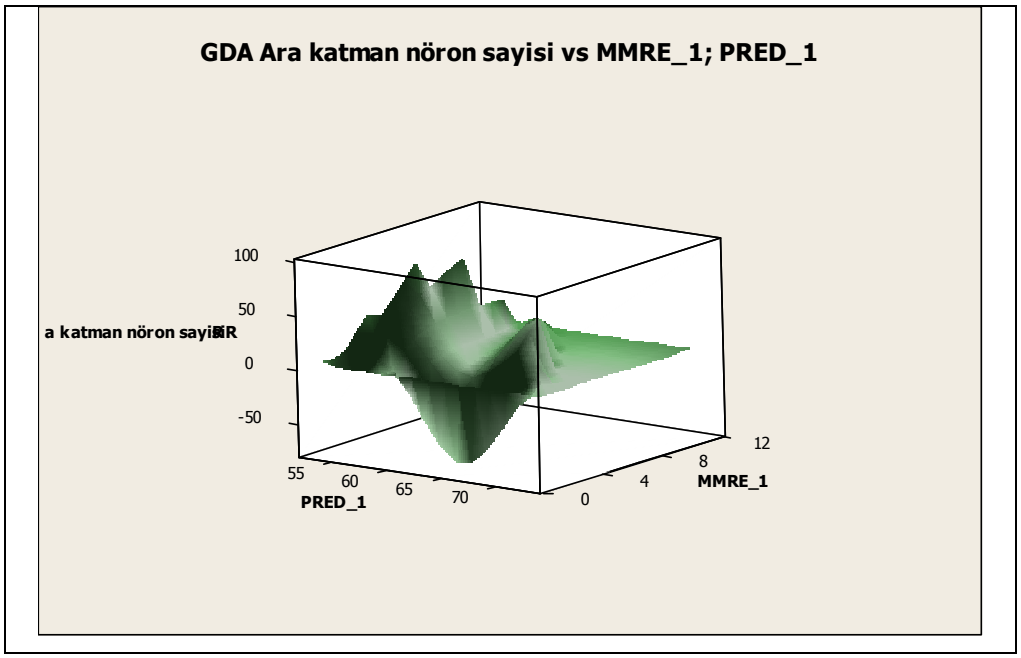
| Ara katman nöron sayısı | MMRE | PRED |
|---|---|---|
| 5 | 0,95 | 73 |
| 10 | 0,47 | 53,3 |
| 15 | 10,85 | 60 |
| 20 | 0,89 | 53,3 |




Şekil 5-13 Adaptasyon kabiliyetli gradyan azalan geri-yayılım öğrenme ağının yakınsama durumu.

Şekil 5-113 de görüldüğü üzere Adaptasyon kabiliyetli gradyan azalan geri-yayılım öğrenme ağı öğrenim sonucunda istenilen performans hedefine ulaşamamıştır. Bunun nedeni Şekil 5-14'de görülmektedir. Adaptasyon kabiliyetli gradyan azalan geri-yayılım öğrenme ağında ara katmanda bulunan nöron sayısı azaldıkça eğitilen ağın maliyet kestirim denkleminin veri setini iyi ifade etmediği görülmektedir. Ağın performans doğruluk oranı nöron sayısı azaldıkça veya çoğaldıkça %53'ları asla geçemediği görülmektedir. Buna karşılık ağın göreceli hata değeri (MMRE) hızlı bir şekilde artmaktadır. Yine burada da bir önceki bölümde olduğu gibi ara katmandaki nöron sayısı 5'in altına düşünce kestirim başarısı çok kötüleşmektedir. Diğer bir ifade ile Adaptasyon kabiliyetli gradyan azalan geri-yayılım öğrenme ağı 5'ten daha az bir nöron sayısını veri setimiz için kullanamayacağı görülmektedir. Ayrıca Adaptasyon kabiliyetli gradyan azalan geri-yayılım öğrenme ağında ara katmandaki nöron sayısı 20'den fazla

olması durumunda kestirim fonksiyonunun performans doğruluk oranı Pred(25) başarı değerini arttırmadığı (~%53) ve ağın göreceli hata değerinin (MMRE = ~10.0) daha fazla düşmediği gözlemlenmektedir. Sonuç olarak Adaptasyon kabiliyetli gradyan azalan geri-yayılım öğrenme ağı bizim veri setimizle projelerde maliyet kestiriminde kullanılamaz durumdadır.




Şekil 5-14 Adaptasyon kabiliyetli gradyan azalan geri-yayılım öğrenme ağı yakınsama grafiği

### 5.2.4 Momentumlu ve adaptasyon kabiliyetli Gradyan öğrenme ağları

Momentumlu ve adaptasyon kabiliyetli gradyan azalan geri-yayılım öğrenme ağı hızlı yakınsamasıyla bilenen bir yapay sinir ağı

eğitim yöntemidir. Bu yöntemde ağ parametrelerinin deneme yanılma yöntemi ile en makul parametreleri belirlenmesi gerekmektedir. Önceki bölümlerde tek başına kullanılan momentumlu ve adaptasyon kabiliyetli gradyan azalan geri-yayılım öğrenme ağlarında başarı elde edilememiştir. Bu bölümde her iki yöntemi aynı anda birlikte uygulayarak başarı elde edilmeye çalışılacaktır.

Yapay sinir ağını oluşturmak, eğitmek ve test ederek başarı oranlarını bulmak için yazılan Matlab kodu aşağıda verilmektedir:

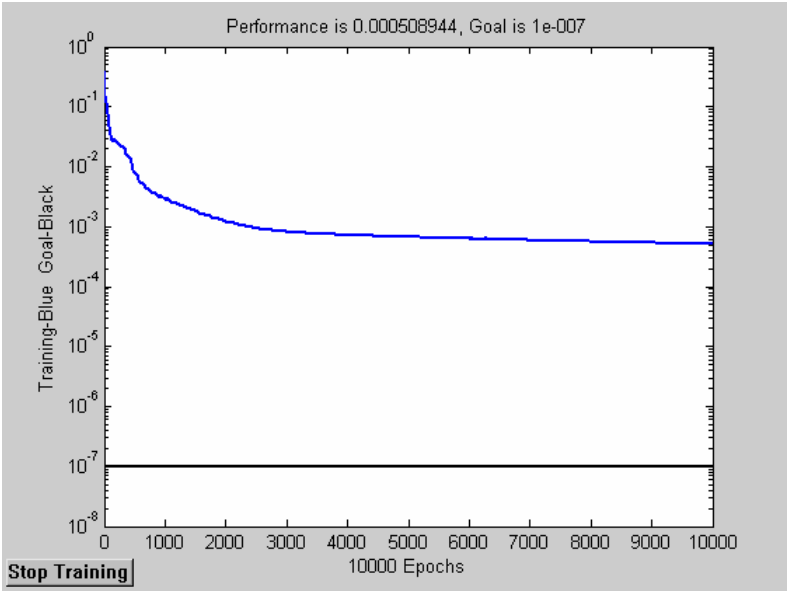
```
1. pred = 0; MMRE = 0; mse = 0; gtmp = 0; y=0; yp=0;
2. input=[data(:,2)'; data(:,3)';data(:,4)'; ];
3. target=[data(:,1)'];
4. [input,minp,maxp,target,mint,maxt] = premnmx(input,target);
5. net=newff(minmax(input),[10,1],{'logsig','purelin'},'traingdx');
6. net.trainParam.goal=1e-7;
7. net.trainParam.epochs=10000;
8. [net,tr]=train(net,input,target);
9. for i = 1 : 15
10. y = testdata(i,2:4);
11. y=tramnmx(y',minp,maxp);
12. output = sim(net,y);
13. yp =  postmnmx(output,mint,maxt);
14. mse = abs((testdata(i,1)-yp)./testdata(i,1));
```

```
15. if mse <= 0.25
16. pred = pred + 1;
17. end;
18. gtmp = gtmp + mse;
19. end;
20. MMRE = gtmp / 15;
21. disp('pred='); disp(pred/15);
22. disp('MMRE='); disp(MMRE);
23. end;
```

Momentum ve Adaptasyon kabiliyetli gradyan azalan geri-yayılım öğrenme ağının öğrenim sonuçları aşağıdaki gibi çıkmıştır.

Çizelge 5-6 Momentum ve Adaptasyon kabiliyetli gradyan azalan geri-yayılım öğrenme ağının öğrenim sonuçları

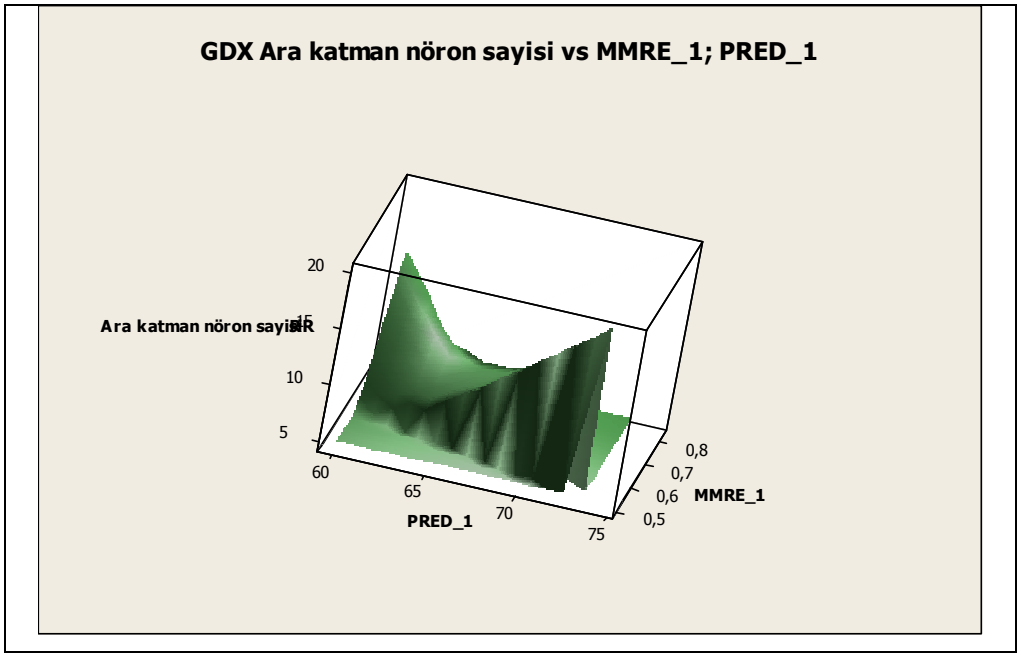
| Ara katman nöron sayısı | MMRE | PRED |
|---|---|---|
| 5 | 0,95 | 73 |
| 10 | 0,47 | 53,3 |
| 15 | 10,85 | 60 |
| 20 | 0,89 | 53,3 |




Şekil 5-15 Momentum ve Adaptasyon kabiliyetli gradyan azalan geri-yayılım öğrenme ağının yakınsama durumu.

Şekil 5-15'de görüldüğü üzere Momentum ve Adaptasyon kabiliyetli gradyan azalan geri-yayılım öğrenme ağı öğrenim sonucunda istenilen performans hedefine ulaşamamıştır. Bunun nedeni Şekil 5-16'da görülmektedir. Momentum ve Adaptasyon kabiliyetli gradyan azalan geri-yayılım öğrenme ağında ara katmanda bulunan nöron sayısı azaldıkça eğitilen ağın maliyet kestirim denkleminin veri setini iyi ifade etmediği görülmektedir. Ağın performans doğruluk oranı nöron sayısı azaldıkça veya çoğaldıkça %53'ları asla geçemediği görülmektedir. Buna karşılık ağın göreceli hata değeri (MMRE) hızlı bir şekilde artmaktadır. Yine burada da bir önceki bölümde olduğu gibi ara katmandaki nöron sayısı 5'in altına düşünce kestirim başarısı çok kötüleşmektedir. Diğer bir ifade ile Momentum ve Adaptasyon kabiliyetli gradyan azalan geri-yayılım öğrenme ağı 5'ten daha az bir nöron sayısını veri setimiz için kullanamayacağı görülmektedir. Ayrıca Momentum ve Adaptasyon kabiliyetli gradyan azalan geri-yayılım

öğrenme ağında ara katmandaki nöron sayısı 20'den fazla olması durumunda kestirim fonksiyonunun performans doğruluk oranı Pred(25) başarı değerini arttırmadığı (~%53) ve ağın göreceli hata değerinin (MMRE = ~10.0) daha fazla düşmediği gözlemlenmektedir. Sonuç olarak Momentum ve Adaptasyon kabiliyetli gradyan azalan geri-yayılım öğrenme ağı bizim veri setimizle projelerde maliyet kestiriminde kullanılamaz durumdadır.




Şekil 5-16 Momentum ve Adaptasyon kabiliyetli gradyan azalan geri-yayılım öğrenme ağı yakınsama grafiği

### 5.2.5 Levenberg-Marquardt öğrenme ağları

Levenberg-Marquardt geri yayılım öğrenme ağı hızlı yakınsamasıyla bilenen bir yapay sinir ağı eğitim yöntemidir. Bu yöntemde ağ parametrelerinin deneme yanılma yöntemi ile en makul parametreleri belirlenmesi gerekmektedir.

Yapay sinir ağını oluşturmak, eğitmek ve test ederek başarı oranlarını bulmak için yazılan Matlab kodu aşağıda verilmektedir:

```
1. pred = 0; MMRE = 0; mse = 0; gtmp = 0; y=0; yp=0;
2. input=[data(:,2)'; data(:,3)';data(:,4)'; ];
3. target=[data(:,1)'];
4. [input,minp,maxp,target,mint,maxt] = premnmx(input,target);
5. net=newff(minmax(input),[10,1],{'logsig','purelin'},'trainlm');
6. net.trainParam.goal=1e-7;
7. net.trainParam.epochs=10000;
8. [net,tr]=train(net,input,target);
9. for i = 1 : 15
10. y = testdata(i,2:4);
11. y=tramnmx(y',minp,maxp);
12. output = sim(net,y);
13. yp =  postmnmx(output,mint,maxt);
14. mse = abs((testdata(i,1)-yp)./testdata(i,1));
15. if mse <= 0.25
```

```
16. pred = pred + 1;
17. end;
18. gtmp = gtmp + mse;
19. end;
20. MMRE = gtmp / 15;
21. disp('pred='); disp(pred/15);
22. disp('MMRE='); disp(MMRE);
23. end;
```

1. pred = 0; MMRE = 0; mse = 0; gtmp = 0; y=0; yp=0;

Bu komutlarla, kullanılan değişkenlerin ilk değerleri verilmektedir.

2. input=[data(:,2)'; data(:,3)';data(:,4)'; ];

Bu komut sonunda input matrisi, 100 tane eğitim verisini içermektedir. Verilerin (data) ilk sütunu maliyet cinsinden efordur yani yapay sinir ağının çıktısını oluşturacağından kullanılmayacaktır. Matlab ortamında Yapay Sinir Ağları'ndaki bir girdiye ilişkin değerlerin satırlara yerleştirilmesi gerektiğinden, 2, 3, ve 4. sütunların transpozları alınmıştır. Bu durumda girdi matrisimiz 3 satır (ağın girdisi) ve 100 sütundan oluşmaktadır

3. target=[data(:,1)'];

Yapay sinir ağlarının eğitiminde kullanılan maliyet verilerimiz, data matrisinin 1. sütununda bulunduğundan transpozu alınarak target dizisine yerleştirilmiştir. Bu dizi maliyet cinsinden gerçek eforları tutmaktadır.

4. [input,minp,maxp,target,mint,maxt] = premnmx(input,target);

Bu komut, veriler üzerinde önişlem uygulamaktadır. Yapay Sinir ağlarına girdi ve çıktı olarak verilecek değerler, farklı tiplerde olduklarından, çok değişik ölçeklerde olabilmektedirler. Hepsinin -1 ile +1 aralığına getirilmesi, ağın etkin bir şekilde öğrenebilmesi için gerekmektedir.

5. net=newff(minmax(input),[10,1],{'logsig','purelin'},'trainlm');

Bu komut ile, ileri sürümlü çok katmanlı algılayıcı yapay sinir ağı oluşturulmaktadır. Arakatmanda bu örnek için 10 adet nöron bulunmaktadır. Arakatman aktarım fonksiyonu, genelde önerildiği gibi logsig olarak belirlenmiştir. Çıktı katmanındaki nöron sayısı da, tek sonuç (maliyet) tahminlendiğinden, 1 olarak verilmiştir. Levenberg-Marquardt geri yayılım öğrenme algoritması kullanılmıştır.

6. net.trainParam.goal=1e-7;

Hata hedefimiz, $10^{-7}$ olarak belirtilmiştir. Bu hata değerine ulaşılıncaya kadar, tüm eğitim verileri yapay sinir ağına tekrar tekrar verilmektedir. Bu değere ulaşıldığında, eğitim tamamlanmıştır ve sonlandırılmaktadır.

7. net.trainParam.epochs=10000;

Eğitimin ne kadar süreceği belirtilmektedir. Hata hedefine daha önce ulaşamazsa, eğitim verisi ağa 10000 kere verilecektir. Daha büyük değerler verildiğinde, eğitim süresi uzar ancak ağ daha iyi öğrenir.

8. [net,tr]=train(net,input,target);

Önceki satırlarda atamalarını yaptığımız girdi ve gerçek çıktı değerlerine göre, oluşturduğumuz ağ üzerinde eğitim işlemini gerçekleştirir. Ağ üzerinde, katmanlar arasındaki tüm ağırlıklar, öğrenme kalitesine göre ayarlanmaktadır.

```
9. for i = 1 : 15
10. y = testdata(i,2:4);
11. y=tramnmx(y',minp,maxp);
12. output = sim(net,y);
13. yp =    postmnmx(output,mint,maxt);
14. mse = abs((testdata(i,1)-yp)./testdata(i,1));
15. if mse <= 0.25
16. pred = pred + 1;
17. end;
18. gtmp = gtmp + mse;
19. end;
```

Bu satırlarda, eğitim verisinden ayrı olarak belirlediğimiz 15 adet test verisi, tramnmx komutu ile önişlemden geçirilmektedir. Döngü her bir test verisi için tekrarlanmaktadır. sim komutu ile, önceden eğitim verisi ile eğitmiş olduğumuz ağın ağırlıklarına göre output değişkenine tahminlenmiş maliyet değeri atanmaktadır. Ardından, elde ettiğimiz sonuç (maliyet değeri), [-1, 1] aralığından, orijinal maliyet değeri ölçeğine ayarlanacak şekilde son işlemden geçirilmektedir. Bu değer ile gerçek hayattaki maliyet arasındaki fark, yine gerçek değere bölünerek mse değeri elde edilmektedir. Bu değer 0.25'ten küçükse sayacımızı bir artırıyoruz. İç döngünün bitiminden sonra, gtmp değerine, ilgili test verisinin hesapladığımız mse değerini ekliyoruz.

```
20. MMRE = gtmp / 15;
```

21. disp('pred='); disp(pred/15);

22. disp('MMRE='); disp(MMRE);

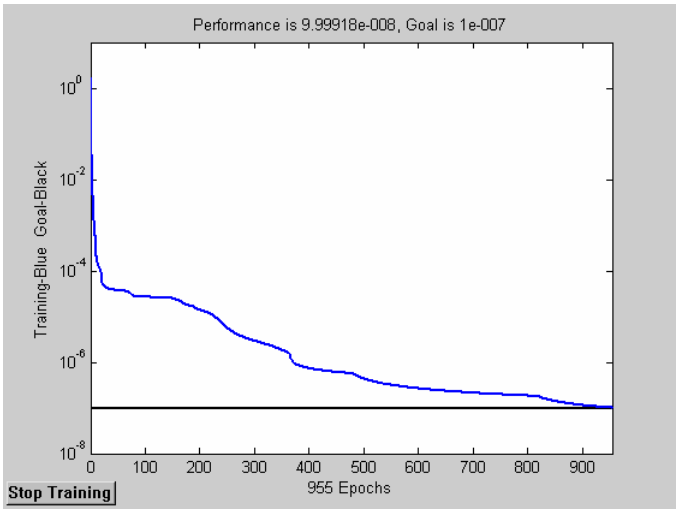
23. end;

Bu satırlarda MMRE değerini, hesapladığımız MSE toplamından yararlanarak hesaplıyoruz. Başarı oranını (pred) ve MMRE değerini ekrana yazdırıyoruz.

Levenberg-Marquardt geri-yayılım öğrenme ağının öğrenim sonuçları aşağıdaki gibi çıkmıştır.

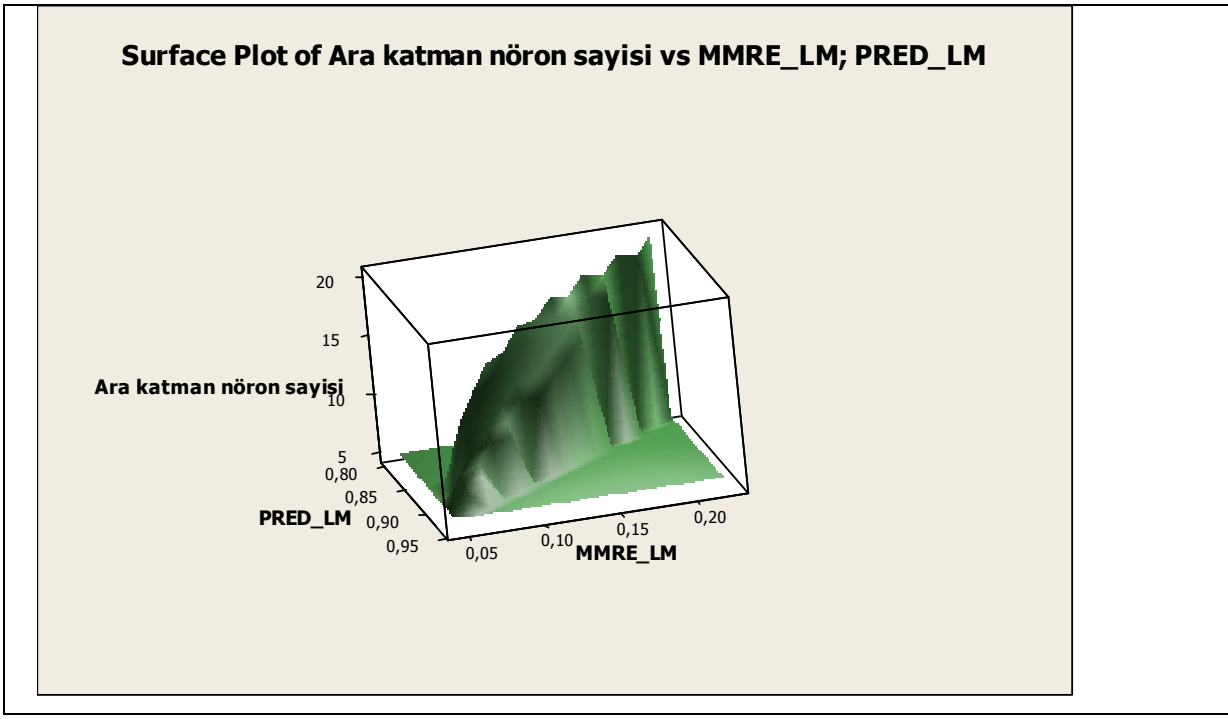
Çizelge 5-7 Levenberg-Marquardt geri-yayılım öğrenme ağının öğrenim sonuçları

| Ara katman nöron sayısı | MMRE | PRED |
|---|---|---|
| 5 | 0.0470 | 0.9333 |
| 10 | 0.0716 | 0.8667 |
| 15 | 0.1065 | 0.8667 |
| 20 | 0.2222 | 0.8000 |




Şekil 5-17 Levenberg-Marquardt geri-yayılım öğrenme ağının yakınsama durumu.

Şekil 5-17'de görüldüğü üzere Levenberg-Marquardt geri-yayılım öğrenme ağı öğrenim sonucunda istenilen performans hedefine ulaşmıştır. Bunun nedeni Şekil 5-18'de görülmektedir. Levenberg-Marquardt geri-yayılım öğrenme ağında arakatmanda bulunan nöron sayısı azaldıkça eğitilen ağın maliyet kestirim denkleminin veri setini daha iyi ifade ettiği görülmektedir. Ağın performans doğruluk oranı nöron sayısı azaldıkça %95'lere çıktığı görülmektedir. Bu karşılık ağın göreceli hata değeri (MMRE) hızlı bir şekilde azalmaktadır. Burada dikkati çeken bir başka durum ise ara katmandaki nöron sayısı 5'in altına düşünce kestirim başarısı çok kötüleşmektedir. Diğer bir ifade ile Levenberg-Marquardt geri-yayılım öğrenme ağı 5'ten daha az bir nöron sayısını veri setimiz için kullanamayacağı görülmektedir.




Şekil 5-18 Levenberg-Marquardt geri-yayılım öğrenme ağı yakınsama grafiği

Ayrıca Levenberg-Marquardt geri-yayılım öğrenme ağında ara katmandaki nöron sayısı 20'den fazla olması durumunda kestirim fonksiyonunun performans doğruluk oranı Pred(25) başarı değerini arttırmadığı (~%85) ve ağın göreceli hata değerinin (MMRE = ~0.1) daha fazla azalmadığı gözlemlenmektedir.

Bunun nedeni sinir ağının ara katmanında nöron sayısı arttıkça öğrenme kabiliyetinin artmasından daha çok unutma kabiliyetinin artması ile açıklanabilir. Çok fazla nöron olduğundan sinir ağı öğrenme gelişiminde önemli rol oynayan nöronlar arası öğrenme ilişkisi zayıflamaktadır. Bunun için sinir ağının başarısı hızla düşmektedir.

# 6. Modellerin Sonuçların Karşılaştırılması

Bu bölümde, önceki bölümde elde edilen deneysel sonuçların literatürdeki benzer çalışmalar ile karşılaştırılması yapılmaktadır. Karşılaştırma bölüm 3.5 de öngörülen çoklu korelasyon katsayısı $R^2$, göreceli hata oranının büyüklüğü ve PRED doğruluk derecesi değerleri üzerinden yapılacaktır. Fakat literatürde yapılan çalışmaların bazıları hem regresyon analizi ve yapay sinir ağ kestirim yöntemlerini desteklemektedir. Bu nedenden dolayı bazı karşılaştırmalarda regresyon analizi, bazılarında yapay sinir ağları ve bir kısmında ise her iki yöntemin sonuçları karşılaştırılacaktır. Örnek olması açısından genetik programlama veya benzetim yolu ile kestirim yöntemlerinin literatürdeki bazı önemli sonuçları karşılaştırma amacı ile bu bölümde verilecektir.

Finnie [Finnie, 1997] çalışmasında, 17 farklı organizasyondan 299 proje örneği üzerinde çalışmıştır. Kestirim fonksiyonunun parametre değişkenleri olarak normalize ve normalize olmayan fonksiyon noktaları ile iş verimliliğini temel almıştır. Regresyon analizi ve yapay sinir ağı kestirim yöntemlerini çalışmasında kestirim modelleri olarak kullanmıştır. Regresyon analizi sonucunda en düşük göreceli hata oranı büyüklüğünü MMRE = 0,623 olarak bulmuştur. Yapay sinir ağı kestirim modelinde geri beslemeli azalan sinir ağ mimarisi kullanılmıştır. Bu modelin kestirim sonucunda en düşük göreceli hata oranı büyüklüğü MMRE = 0,352 olarak bulunmuştur. Bu iki sonuç bizim tez çalışmamızda gerçekleştirdiğimiz deneysel sonuçlardan daha iyi değildir.

Jorgensen [Jorgensen, 2004] çalışmasında Norveç bilişim endüstrisinden 49 proje örneği üzerinde çalışmıştır. Kestirim fonksiyonunun parametre değişkenleri olarak kurum rolünü, katılımı ve

müşteri önceliklerini temel almıştır. Çalışmada regresyon analizi yöntemi kestirim modelini kullanmıştır. Regresyon analizi sonucunda en düşük göreceli hata oranı büyüklüğünü özel bir proje gözlemi için en düşük MMRE = 0,03 olarak bulmuştur. Fakat bu çok özel bir değerdir ve özel bir projeye aittir. 49 proje için göreceli hata oranı büyüklüğü MMRE = 0,302 olarak bulunmuştur. Bu iki sonuç bizim tez çalışmamızda gerçekleştirdiğimiz deneysel sonuçlardan daha iyi değildir.

Dolado [Dolado, 2000] çalışmasında 46 proje örneği üzerinde çalışmıştır. Regresyon analizi, genetik programlama ve yapay sinir ağı kestirim yöntemlerini çalışmasında kestirim modelleri olarak kullanmıştır. Regresyon analizi sonucunda en düşük göreceli hata oranı büyüklüğünü MMRE = 0,27 olarak bulmuştur. Doğruluk değeri Pred(25) = %65 olarak belirlemiştir. Ayrıca çoklu korelasyon katsayısı değerini $R^2$ = %64 olarak bulmuştur. Yapay sinir ağı kestirim modelinde geri beslemeli azalan sinir ağ mimarisi kullanılmıştır. Bu modelin kestirim sonucunda en düşük göreceli hata oranı büyüklüğü MMRE = 0,25 olarak bulunmuştur. Doğruluk değeri Pred(25) = %59 olarak bulunmuştur. Genetik programlama tekniği ile kestirimde en düşük göreceli hata oranı büyüklüğü MMRE = 0,29 ve doğruluk değeri Pred(25) = %65 olarak bulmuştur. Görüldüğü üzere bu sonuçlar bizim tez çalışmamızda gerçekleştirdiğimiz deneysel sonuçlardan daha iyi değildir.

Burgess [Burgess, 2001] çalışmasında 63 proje örneği üzerinde çalışmıştır. Genetik programlama ve yapay sinir ağı kestirim yöntemlerini çalışmasında kestirim modelleri olarak kullanmıştır. Yapay sinir ağı kestirim modelinin işlettirilmesi sonucunda en düşük göreceli hata oranı büyüklüğü MMRE = 0,59 olarak bulunmuştur. Doğruluk değeri Pred(25) = %56 olarak bulunmuştur. Ayrıca çoklu

korelasyon katsayı değeri $R^2$ = %65 olarak bulunmuştur. Genetik programlama tekniği ile kestirimde en düşük göreceli hata oranı büyüklüğü MMRE = 0,37 ve doğruluk değeri Pred(25) = %28 olarak bulunmuştur. Çoklu korelasyon katsayısı olarak $R^2$ = %82 olarak bulunmuştur. Görüldüğü üzere bu sonuçlar bizim tez çalışmamızda gerçekleştirdiğimiz deneysel sonuçlardan daha iyi değildir.

Shepperd (Shepperd, 1997) çalışmasında toplam 275 proje örneği içeren 9 veriseti üzerinde çalışmıştır. Çalışmasında regresyon analizi ve benzeşim kestirim modelleri kullanmıştır. Regresyon analizi sonucunda en düşük göreceli hata oranı büyüklüğünü MMRE = 0,29 olarak bulmuştur. Doğruluk değeri Pred(25) = %50 olarak belirlenmiştir. Benzeşim modelinin kestirim sonucunda en düşük göreceli hata oranı büyüklüğü MMRE = 0,29 olarak bulunmuştur. Doğruluk değeri Pred(25) = %70 olarak bulunmuştur. Görüldüğü üzere bu sonuçlar bizim tez çalışmamızda gerçekleştirdiğimiz deneysel sonuçlardan daha iyi değildir.

Stamelosa (Stamelosa, 2003) çalışmasında toplam 789 proje örneği içeren ISBSG sürüm 6 veri seti üzerinde çalışmıştır. Çalışmasında benzeşim modelini kullanarak kestirim sonucunda en düşük göreceli hata oranı büyüklüğü MMRE = 0,23 olarak bulunmuştur. Doğruluk değeri Pred(25) = %70,37 olarak bulunmuştur. Görüldüğü üzere bu sonuçlar bizim tez çalışmamızda gerçekleştirdiğimiz deneysel sonuçlardan daha iyi değildir.

Oliveira (Oliveira, 2005) çalışmasında NASA'nın 18 proje örneği üzerinde çalışmıştır. Regresyon analizi ve Radial bazlı fonksiyon ağı tabanlı yapay sinir ağı kestirim yöntemlerini çalışmasında kestirim modelleri olarak kullanmıştır. Regresyon analizi sonucunda en düşük göreceli hata oranı büyüklüğünü MMRE = 0,179 olarak bulmuştur. Doğruluk değeri Pred(25) = %83,3 olarak bulunmuştur. Yapay sinir ağı

kestirim modelinde kestirim sonucunda en düşük göreceli hata oranı büyüklüğü MMRE = 0,187 olarak bulunmuştur. Doğruluk değeri Pred(25) = %72,2 olarak bulunmuştur. Bu sonuçlar bizim tez çalışmamızda gerçekleştirdiğimiz deneysel sonuçlardan daha iyi değildir.

Dolado (Dolado, 2001) çalışmasında eğri tahminleme yöntemi ile genetik programlama kestirim yöntemlerini kullanmıştır. Eğri tahminleme yönteminin uygulaması sonucunda en düşük göreceli hata oranı büyüklüğünü MMRE = 0,089 olarak bulmuştur. Doğruluk değeri Pred(25) = %94,29 olarak bulunmuştur. Genetik programlama modelinde kestirim sonucunda en düşük göreceli hata oranı büyüklüğü MMRE = 0,087 olarak bulunmuştur. Doğruluk değeri Pred(25) = %94,4 olarak bulmuştur. Bu sonuçlar bizim tez çalışmamızda gerçekleştirdiğimiz deneysel sonuçlardan daha iyi değildir.

Sentas (Sentas, 2005) çalışmasında üç farklı veri seti üzerinden kestirim çalışmaları yapmıştır. Çalışmamıza uyması açısından biz ISBSG sürüm 7 veri seti üzerinde yapılan kestirim hesaplamalarını tez çalışmasındaki sonuçlarla karşılaştırdık. Regresyon analizi kestirim yöntemi çalışmalarında kestirim modeli olarak kullanılmıştır. Regresyon analizi sonucunda en düşük göreceli hata oranı büyüklüğünü MMRE = 0,33 olarak bulmuştur. Doğruluk değeri Pred(25) = %53,8 olarak bulunmuştur. Bu sonuçlar bizim tez çalışmamızda gerçekleştirdiğimiz deneysel sonuçlardan daha iyi değildir.

Liu (Liu, 2005) çalışmasında ISBSG sürüm 8 veri setini kullanmıştır. Regresyon analizi yönteminin uygulaması sonucunda uyarlanmış çoklu korelasyon katsayısı $R^2$ = %76 çıkmıştır. Bu sonuç bizim tez çalışmamızda gerçekleştirdiğimiz deneysel sonuçlardan daha iyi değildir.

Tez çalışmamız kapsamında gerçekleştirdiğimiz çalışmaların sonuçlarını literatürdeki aynı alanda yapılan benzer çalışmalar ile karşılaştırdığımızda aşağıdaki Çizelge 6-1'de ortaya çıkmaktadır. Çizelgeden görüleceği üzere tez kapsamında yaptığımız çalışmaların sonuçları literatürdeki benzer çalışmalardan çok daha iyi sonuçlar vermiştir.

Çizelge 6-1 Tez sonuçlarının literatürdeki çalışmalar ile karşılaştırılması

| **Sıra No** | **Litaratürdeki Çalışmalar** | **Regrasyon Analizi** | | | **Yapay Sinir Ağları** | |
|---|---|---|---|---|---|---|
| | | **R2** | **MMRE** | **PRED(25)** | **MMRE** | **PRED(25)** |
| 1 | Tez Çalışması | %99 | 0,023 | %98 | 0,047 | %93 |
| 3 | Wittig | ---- | -------- | ---------- | 0,29 | ------- |
| 4 | Dolado, 2000 | %64 | 0,27 | %65 | 0,25 | %59 |
| 5 | Liu'nun Çalışması [7] | %76 | ---- | ---- | ---- | ---- |
| 6 | Oliveira'nın Çalışması | ---- | 0,179 | %83,33 | 0,187 | %72,22 |
| 7 | Finnie'nin Çalışması | ---- | 0,623 | %40 | 0,352 | ---- |
| 8 | Jorgensen | ---- | 0,302 | ---- | ---- | ---- |
| 9 | Burgess | ---- | ---- | ---- | 0,59 | %56 |
| 10 | Shepperd | ---- | 0,29 | %50 | ---- | ---- |
| 11 | Sentas, 2005 | ---- | 0,33 | %53,8 | ---- | ---- |

## 7. Sonuç ve Öneriler

Yazılım maliyet kestirimi veya tahminlemesi, yazılım geliştirmenin en önemli aşamalarından birisidir. Proje yöneticisi, proje süresi/maliyeti doğru tahminleyerek projedeki belirsizlikleri azaltır ve projenin gelişimini gerçek giderlerle, planlananları veya tahminlenenleri karşılaştırarak değerlendirir. Yazılım projelerinin maliyetlerinin ve sürelerinin tahminlenmesi, birçok parametreye bağlı ve karmaşık bir işlemdir.

Zeka, karmaşık bir problemi çözmek için gerekli bilgileri toplayıp birleştirme kabiliyetidir. Yapay zeka ise, organik olmayan sistemlerdeki zekadır. Makine öğrenmesi veya deneyimlere dayalı olarak öğrenme, yapay zekanın en önemli araştırma alanlarındandır. Yazılım maliyetleri, uzman kişiler tarafından tahminlenebileceği gibi, bu alanda regresyon gibi yöntemler ve güçlü öğrenme yeteneğine sahip yapay sinir ağları ve genetik algoritmalar gibi yapay zeka yöntemleri de kullanılabilmektedir.

ISBSG (International Software Benchmarking Standards Group) veri seti sürüm 9, dünyada değişik ülkelerde yapılmış birçok projeye ilişkin bilgileri tutan bir veriler kümesidir. Projelerin, hangi yazılım geliştirme ortamları kullanılarak gerçekleştirildiğinden, personel sayılarına, sürelerinden, kullanılan veritabanı ve işletim sistemlerine, kaynak kod satır sayılarından, harcanan çabaya kadar birçok özellikleri burada yer almaktadır. Yazılım maliyetleri kestirimi alanında yapılan bilimsel çalışma ve yayınlarda bu veri seti temel oluşturmakta ve geliştirilen yöntemlerin karşılaştırılmasında da belirleyici rolü oynamaktadır. ISBSG, yazılım ölçümünde “ISO/IEC Standard 15939 Measurement Process: 2002” e dayalı taslak bir standart oluşturmuştur.

Bu tez çalışmasında, ISBSG verileri içerisinden eksik sahalara sahip olanlar ayıklanmıştır. Ardından, dağılım ve korelasyonları dikkate alınarak uygun olan 115 tanesi seçilmiştir. Bu veriler istatistik ve makine öğrenmesi kapsamındaki regresyon yönteminde katsayıların bulunmasında kullanılmıştır. Regresyon modelinin gerçekleştiriminde Minitab ortamı kullanılmıştır. Yapay zeka kapsamındaki yapay sinir ağlarının eğitim ve test aşamalarında da aynı veriler dikkate alınmıştır. Literatürde bu alandaki çalışmaların çoğunda yararlanılan çok katmanlı algılayıcı yapay sinir ağı modeli tercih edilmiş ve gerçekleştiriminde Matlab ortamı kullanılmıştır. Elde edilen sonuçlara dayalı olarak yöntemler performanslarına göre karşılaştırılmıştır.

Bu çalışmanın en önemli katkısı, kuvvet değerlerini ve logaritmaları dikkate alan matematik temelli bir regresyon yönteminin ilk olarak yazılım maliyet tahminlemesinde kullanılması ve bu alana uyarlanmasıdır. Bu yöntemle elde edilen deneysel sonuçlar, literatürdeki diğer sonuçlarla karşılaştırılmış ve en yüksek başarı değerleri elde edilmiştir. İngiltere de Birmingham üniversitesinde gerçekleştirilen IDEAL 2007 konferansında sunulmuştur (Adalıer et al., 2007).

Regresyon uzun yıllardır kullanılan, kendisini kanıtlamış güçlü bir matematik ve istatistik altyapısına sahip bir yöntemdir. Yazılım maliyet tahminlemelerinde de en çok kullanılan yaklaşımdır. Verilerin popülasyonu temsil edici özelliğinin olması, parametrelerin uygun seçilmesi, modelin uygunluğunun ve tahminlenen parametrelerin istatistiksel kalitesinin sınanması önemlidir.

Yapay sinir ağları, öğrenme yeteneği en yüksek yapay zeka yöntemlerindendir. Dolayısı ile, farklı sayıda ve değişik girdi değerlerine göre eğitilebilmekte, farklı şartlara uyum sağlayabilmektedir. Örnek olarak, herhangi bir proje yöneticisi, ISBSG

verileri ile eğitilmiş bir ağı doğrudan kullanarak, maliyet tahmini yapabileceği gibi, projenin, proje ekibinin, geliştirme ortamlarının durumlarını ve değişkenliğini dikkate alarak, sadece daha önce benzeri şirketlerde yapılan veya şirketin önceki projelerinin verilerini kullanarak da ağı eğitebilir. Bazı veriler içerisinde, birtakım sahalara ilişkin değerlerin kaydedilmemiş veya unutulmuş olması durumunda da yapay sinir ağları duyarlık kaybına rağmen doğruya yakın sonuç verebilecektir. Ancak doğru sonuç verebilmesi için yapay sinir ağlarında uygun modellerden birisinin seçilmesi, katman ve nöron sayılarının uygun şekilde ayarlanması; yeterli sayıda ve problem uzayının tüm bölgelerinden eğitim ve test verileri bulunması, eğitim yönteminin doğru seçilmesi ve parametrelerine doğru değerler verilmesi büyük önem taşımaktadır. Yapay sinir ağları, güçlü uyarlanabilme özelliğinin yanında sonucu neye dayanarak bulduğunu açıklayabilme özelliğine sahip değildir, yani kara kutu olarak düşünülmektedirler [Aybars UĞUR Ders notları, 2005]. Bu yapay zeka yöntemini kullanmaya başlayanların ilk öğrendiği bilgilerden birisi de, yapay sinir ağlarını değişik problemlere uyarlarken doğru ayarlamaların yapılmasına ilişkin evrensel kurallar olmadığı, birtakım bilgiler ışığında, tecrübelerin büyük önem taşıdığıdır.

Benzeri konularda çalışma yapmak isteyen araştırmacılar, maliyet kestiriminde kullanılan ve bu çalışma kapsamı dışında kalan genetik programlama gibi diğer yapay zeka teknikleri ile, diğer matematiksel modellerin gerçekleştirimini yaparak sonuçları karşılaştırabilirler. Burada regresyon için geliştirilen modelin diğer yöntemlere uyarlanması üzerine de çalışılmaktadır.

Birden fazla yöntemin birlikte kullanımı diğer bir ifade ile melez sistemlerin kullanımı da performansı arttırabilecek yaklaşımlardan birisidir. Örnek olarak, yapay sinir ağlarının uzman sistemlerle beraber

kullanımı, açıklanabilirliğini artıracaktır. Burada geliştirilen model dahil literatürdeki değişik yaklaşımların gerçekleştirimlerini içeren bir yazılım geliştirilmesi ve esnek bir arayüz tasarımı yapılması, pratik hayatta proje yöneticileri ve yazılım geliştiriciler için çok yararlı olabilecektir.

Ayrıca ISBSG veri setinde test ve eğitim kümesi değiştirilerek daha iyi sonuçlar alınması için yeni deneysel çalışmalar yapılabilir. Bu tez çalışmasında proje maliyet kestiriminde etkili olan unsurlar mekanik ölçeklerden alınmıştır. Diğer bir ifade ile proje yönetiminde etkili olan verimlilik oranını etkileyen diğer unsurlar olarak projedeki mimari, geliştirme ortamı ve araçları, personelin eğitim temeli, test yöntemleri gibi mekanik unsurlar temel alınmıştır. Bu, ISBSG veri setinin 50 kadar bu tür mekanik unsuru içermesinden kaynaklanmaktadır.

Fakat bir de verimliliği etkileyen psikolojik ve sosyolojik unsurlar bulunmaktadır. Örneğin çok tecrübeli ve eğitimli bir geliştiricinin motivasyonunu olumsuz etkileyen durumları ölçekleyen unsurlar psikolojide bulunmaktadır. Bu unsurlar proje yönetimindeki mekanik unsurlarla birlikte ele alınarak projelerde maliyet kestirimi daha doğruya yakın olacaktır. Böyle bir çalışmanın çok ilginç olacağı şüphesizdir.

# KAYNAKLAR DİZİNİ

**Adalier, O., Uğur, A., Korukoğlu, S., Ertaş, K.,** 2007, A New Regression Based Software Cost Estimation Model Using Power Values, 8th International Conference on Intelligent Data Engineering and Automated Learning (IDEAL'07), 16th-19th December, 2007, Birmingham, UK.

**Aybars, U.,** 2005, Yapay Zeka ve Yapay Sinir Ağları Ders notları, Ege Üniversitesi Bilgisayar Mühendisliği Bölümü.

**Ayyıldız, M., Kalıpsız, O., Yavuz, S.,** 2006, YEEM : Yazılım Projeleri Maliyet Tahminleme Ölçev Seti ve Modeli, Eleco 2006.

**Bashir, H. A. and Thomson, V.,** 2001, Models for estimating design effort and time, Design Studies Vol 22 No 2 March, Elsevier.

**Bontempi, G. And Kruijtzer, K.,** 2003, The use of intelligent data analysis techniques for system-level design: a software estimation example, Soft Computing 8 (2004) 477–490 Springer-Verlag.

**Boraso, M., Montangero, C. and Sedehi, H.,** 1996, Software Cost Estimation: an experimental study of model performances, Technical Report: TR-96-22, University of Pisa, Italy.

**Briand, L. C., Feng, J. and Labiche, Y.,** 2002, Using Genetic Algorithms and Coupling Measures to Devise Optimal Integration Test Orders, ACM - SEKE '02, July 15-19, Ischia, Italy.

**Burgess, C.J. and Lefley, M**., 2001, Can genetic programming improve software effort estimation? A comparative evaluation, Information and Software Technology 43, p: 863-873, Elsevier.

**Chang, C. K., Christensen, M. J. and Zhang, T.,** 2001, Genetic Algorithms for Project Management, Annals of Software Engineering 11, p:107–139, Kluwer Academic Publishers.

**Cristianini, N. and Shawe-Taylor, J.,** 2000, An Introduction to Support Vector Machines (and other Kernel-Based Learning Methods). Cambridge University Press.

**Delany, S.J., Cunningham, P. and Wilke, N.,** 1997, The limits of CBR in Software Project Estimation, German Workshop on Case-Based Reasoning.

**Devroye, L., Gyorfi, L., and Lugosi, G. A.,** 1996, Probabilistic Theory of Pattern Recognition. Springer.

**Dolado,J.J., and Fernandez, L.,** 1998, Genetic Programming, Neural Networks and Linear Regression in Software Project Estimation, International Conferance on Software Process Improvement, Research, Education and Trainig.

**Dolado, J.J.,** 2000, A Validation of the Component-Based Method for Software Size Estimation, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 26, NO. 10, p: 1006-1020.

**Dolado, J.J.,** 2001, On the Problem of the Software Cost Function, Elsevier Science, Information and Software Technology 43, p: 61-72.

**Goodman, P.A.,** 1992, Application of cost-estimation techniques: industrial perspective, Information and Software Technology, Vol 34 No 6.

**Grimstad, S., Jorgensen, M. and Østvold, K.M.,** 2006, Software Effort Estimation Terminology: The tower of Babel, Information and Software Technology, , Elsevier, 48,p302-310.

**Finnie, G.R., Wittig, G.E. and Desharnais, J. M.,** 1997, A Comparison of Software Effort Estimation Techniques: Using Function Points with Neural Networks, Case-Based Reasoning and Regression Models, Journal of Systems Software, v 39, p: 281-289, Elsevier Science Inc.

**Freeman, J.A. and Skapura, D.M.,** 1992, Neural Networks: Algorithms, Applications and Programming Techniques", Addison-Wesley Publishing Company.

**Heemstra, F.J**., 1992, Software cost estimation. Information and Software Technology,. 34(10): p. 627-639.

**Hu, Q., Plant, R.T. and Hertz, D.B.,** 1998, Software Cost Estimation Using Economic Production Models, Journal of Management Information System, Vol. 15, No: 1, pp-143-163.

**Huang, X., Danny, Hu, D., Ren, J., and Capretz, L.F.,** 2006, A soft computing framework for software effort estimation", Soft Computing 10, p: 170–177, Springer-Verlag.

**ISBSG,** 2004, International Software Benchmarking Standards Group. Release 9. http://www.isbsg.org.

**Jones, C.,** 1991, Applied Software Measurement: Assuring Productivity and Quality, McGraw-Hill.

**Jorgensen, M. and Sjoberg, D.I.K.,** 2001, Impact of effort estimates on software project work, Information and Software Technology 43, p: 939-948, Elsevier.

**Jorgensen, M., Indahl, U. and Sjøberg, D.,** 2003, Software effort estimation by analogy and ''regression toward the mean", The Journal of Systems and Software 68 p: 253–262, Elsevier.

**Jorgensen, M.,** 2004, Regression Models of Software Development Effort Estimation Accuracy and Bias, Empirical Software Engineering, 9, p:297–314, Kluwer Academic Puplishers.

**Jorgensen, M.,** 2004, Top-down and bottom-up expert estimation of software development effort, Information and Software Technology 46 pp: 3-16 Elsevier.

**Jorgensen, M. and Shepperd, M.,** 2007, A Systematic Review of Software Development Cost Estimation Studies, IEEE Transactions On Software Engineering, Vol. 33, No. 1.

**Liu, Q. and Mintram, R.C.,** 2005, Preliminary Data Analysis Methods in Software Estimation, Software Quality Journal, 13, p: 91-115.

**Mandic, D.P. and Chambers, J.A.,** 2001, Recurrent Neural Networks For Prediction, John Wiley & Sons, Ltd..

**Matlab,** 2002, The Language of Technical Computing, Version 6.5.180913a Release 13, Copyright 1984-2002, The MathWorks, Inc.

**Maxwell, K., Wassenhove, L.V., and Dutta, S.,** 1999, Performance Evaluation of General and Company Specific Models In Software Development Effort Estimation, Management Science, Vol. 45, No. 6, p: 787-803.

**Michael, C.,** 2004, Parameter Estimation for Statistical Parsing Models: Theory and Practice of Distribution-Free Methods. Book chapter in Harry Bunt, John Carroll and Giorgio Satta, editors, New Developments in Parsing Technology, Kluwer.

**Moløkken, K**., **Jørgensen, M., Tanilkan, S. S., Gallis, H., Lien, A.C., Siw E. Hove, S.E.,** 2004, Project Estimation in the Norwegian Software Industry – A Summary, Simula Research Laboratory, P.O.Box 134, 1325 Lysaker, Norway. March 3rd.

**Moser, S., Sellers, B.H. and Misic, V.B., 1999,** Cost estimation based on business models, The Journal of System and Software 49, p: 33-42, Elsevier.

**Ohlsson, M.S., Wohlin, c. and Regnell,** 1998, A project effort estimation study, Information and Software Technology 40, p: 831-889, Elsevier.

**Oliveira A.L.I.,** 2005, Estimation of software project effort with support vector regression, Neurocomputing, Elsevier, Article in press.

**Sağıroğlu Ş., Beştok E. Ve Erler M.,** 2003**,** Mühendislikte Yapay Zeka Uygulamaları, Ufuk Yayıncılık.

**Sentas, P., Angelis, L., Stamelos, I., Bleris, G.,** 2005, Software productivity and effort prediction with ordinal regression, Information and Software Technology 47, p: 17–29.

**Shan, Y., McKay, R.I, Lokan, C. J., Essam, D.L.,** 2002, Software Project Effort Estimation Using Genetic Programming, IEEE Proceedings of International Conference on Communications Circuits and Systems, vol.2, p: 1108- 1112.

**Shepperd, M. and Schofield, C.,** 1997, Estimating Software Project Effort Using Anologies, IEEE Transactions on Software Engineering, Vol. 23, No.12.

**Stamelosa, I., Angelisa, L., Morisiob, M., Sakellarisc, E., Blerisa, G.L.,** 2003, Estimating the development cost of custom software, Information & Management 40, p: 729–741.

**Stensrud E. and Myrtveit I.,** 1998,Human Performance Estimating with Analogy and Regression Models: An Empirical Validation, Proceedings of the 5th International Symposium on Software Metrics,

**Vapnik, V. N.,** 1998, Statistical Learning Theory. New York: Wiley.

**Wieczorek, I. and Ruhe, M.,** 2002, How Valuable is company-specific Data Compared to multi-company Data for Software Cost Estimation?, Proceedings of the Eighth IEEE Symposium on Software Metrics (METRICS.02).

**Wittig, G. And Finnie, G.,** 1997, Estimating software development effort with connectionist models, Information and Software Technology 39, pp: 469-476, Elsevier.

**Xu, Z.** and Khoshgoftaar, T.M., 2004, Identification of fuzzy models of software cost estimation, Fuzzy Sets and Systems 145, p: 141-163, Elsevier.

# ÖZGEÇMİŞ

**Adı Soyadı** : Oktay ADALIER
**Doğum Tarihi** : 1969
**Doğum Yeri** : Almanya
**Medeni Hali** : Evli
**Uyruğu** : T.C.
**Görevi** : Başuzman Araştırmacı
TUBITAK-UEKAE
Tasarım ve Geliştirme Bölümü

# EĞİTİM

Yüksek Lisans : **Orta Doğu Teknik Üniversitesi**
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı,1995

Lisans : **Orta Doğu Teknik Üniversitesi**
Fen Edebiyat Fakültesi
Matematik Bölümü, 1992

Lise : **Bursa Cumhuriyet Lisesi, 1987**

**Araştırma Alanları** : Proje Yönetimi, İstatistiksel Analiz, Yapay Zeka

**Yabancı Dil** : İngilizce, Almanca