T.C.
İSTANBUL ÜNİVERSİTESİ-CERRAHPAŞA
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

DOKTORA TEZİ

YAPAY ZEKÂ YÖNTEMLERİ İLE YAZILIMLARIN MALİYETLERİNİN TAHMİN EDİLMESİ

Şükran EBREN KARA

DANIŞMAN
Prof. Dr. Rüya ŞAMLI

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

İSTANBUL-2022

Bu çalışma, Tarih girmek için burayı tıklatın. tarihinde aşağıdaki jüri tarafından ....., ...nda ..... olarak kabul edilmiştir.

Tez Jürisi

Prof. Dr. Rüya ŞAMLII(Danışman)
İstanbul Üniversitesi-Cerrahpaşa
Mühendislik Fakültesi

Prof. Dr. Ahmet SERTBAŞI
İstanbul Üniversitesi-Cerrahpaşa
Mühendislik Fakültesi

Dr. Öğr. Üyesi Ceren ÇUBUKÇU CERASİ
Gebze Teknik Üniversitesi
İşletme Fakültesi

Doç. Dr. Selçuk SEVGEN
İstanbul Üniversitesi-Cerrahpaşa
Mühendislik Fakültesi

Dr. Öğr. Üyesi Şengül BAYRAK
Sabahattin Zaim Üniversitesi
Mühendislik ve Doğa Bilimleri Fakültesi

20.04.2016 tarihli Resmi Gazete'de yayımlanan Lisansüstü Eğitim ve Öğretim Yönetmeliğinin 9/2 ve 22/2 maddeleri gereğince; Bu lisansüstü teze, İstanbul Üniversitesi-Cerrahpaşa'nın abonesi olduğu intihal yazılım programı kullanılarak Lisansüstü Eğitim Enstitüsü'nün belirlemiş olduğu ölçütlere uygun rapor alınmıştır.

ÖNSÖZ

Doktora eğitimimde danışmanlığımı üstelenen, ilk tanıştığımız günden beri beni yönlendiren, çözüm odaklı pratik zekâsı ile heyecanlandıran, kendisine hayran bırakan bana güvenerek değerli olduğumu hissettrien, toparlanmam için bana zaman tanıyan, toparlandığımda doğrulmama yardımcı olan yoluma ışık tutan kıymetli danışman hocam Prof. Dr. Rüya ŞAMLI'ya sonsuz teşekkürler.

Öncellikle bende hep bir baba şefkati hissi uyandıran Bilgisayar Mühendisliği Bölüm Başkanı değerli jüri üyesi Prof. Dr. Ahmet SERTBAŞ'a, yapıcı yönlendirmeleri ile tezimin şekillenmesine yardımcı olan değerli jüri üyesi Dr. Öğr. Üyesi Ceren ÇUBUKÇU CERASİ'ye ve bölümün diğer saygı değer hocalarına teşekkürü bir borç bilirim.

Ayrıca tüm hayatım boyunca yanımda olan aldığım kararları hep destekleyen hiçbir maddi manevi yardımlarını esirgemeyen beni sürekli cesaretlendiren başta sevgili babam Hasan EBREN'e, annem Taybet EBREN'e ve kardeşlerime çok teşekkürler. Tanıştığımız ilk günden beri beni yalnız bırakmayan aldığım her kararda yanımda duran, bütün sıkıntılarıma sabırla katlanan, destek olan tavsiyeleri ile beni cesaretlendiren hayat arkadaşım, dostum, sırdaşım canım eşim Dr. Ferhat KARA'ya ve hayattaki can damarım olan sevgili çocuklarıma sonsuz teşekkürler.

Son olarak beni sürekli destekleyen, motive eden meslektaşlarıma, çıkmaza girdiğim bir anda bana can simidi gibi yetişen Dr. Zeynep Behrin GÜVEN AYDIN'a ayrıca teşekkür ederim.

Mayıs 2022 Şükran EBREN KARA

İÇİNDEKİLER

Sayfa No

ŞEKİL LİSTESİ

Sayfa No

TABLO LİSTESİ

Sayfa No

SİMGE VE KISALTMA LİSTESİ

Kısaltmalar	Açıklama
AA	**:** Factor Which Reflects The Initial Assessment Costs Of Deciding If Software May Be Reus – Yazılımın Tekrar Kullanılacağını Dikkate Almakla Maliyetin Başlangıç Tahmini Yansıtan Etken
ABD	**:** Amerika Birleşik Devletleri
AFN	**:** Ayarlanmamış Fonksiyon Noktaları
APE	**:** Absolute Percentage Error – Mutlak Yüzde Hata
ARFF	**:** Attribute Relationship File Format – Öznitelik İlişkisi Dosyası Biçimi
ASLOC	**:** Source Line Of Code – Kaynak kodun Satır Sayısı
ASMA	**:** Australian Software Metrics Association – Avusturalya Yazılım Metrikleri Birliği
AT/100	**:** Otomatik oluşturulan kodun toplam sistem koduna yüzdesi
ATPROD	**:** Automated Production – Kod Üretiminin Üretkenlik Seviyesi
BM	Bulanık Mantık
CfsSubsetEval	**:** Corelation-based Feature Subset Selection Evaluation – Korelasyon Tabanlı Özellik Seçim Değerlendirici
COCOMO	**:** Constructive Costing Model – Yapı Maliyet Modeli
CSBSG	**:** Chinese Software Benchmarking Standards Group – Çin Yazılım Kıyaslama Standartları Grubu
DAF	**:** Değer Ayarlama Faktörü
DM	**:** Percentage Of Design Modified – Değiştirilen Tasarım Yüzdesi
EAF	**:** Effort Adjustment Factor – Maliyet Faktörü Çarpanı
EKK	**:** En Küçük Kareler Tekniği
ESLOC	**:** Effective Source Line Of Code – Etkili Kaynak Kodun Satırlarının Sayısı
FCIL	**:** Facilities – Destek Araç Gereçler
FN	**:** Fonksiyon Noktası
GA	**:** Genetik Algoritma
GPL	**:** General Public Licence – Genel Kamu lisansı
GUI	**:** Grafic User Interface – Gragik Kullanıcı Ara yüzü
IM	**:** Percentage Of The Original Integration Effort Required For Integrating The Reused Software – Tekrar Kullanılan Yazılımı Bütünleştirmek İçin Gereken Başlangıç Bütünleşme Çabası Yüzdesi
ISBSG	**:** International Software Benchmarking Standards Group – Uluslararası Yazılım Kıyaslama Standartları Grubu
KLOC	**:** Kilo Line Of Code – 1000 Kod Satır Sayısı

KSLOC **:** Kilo Source Line Of Code – 1000 Kaynak Kod Satır Sayısı

LIFE **:** Laboratory for Interchange Fuzzy Engineering – Değişim Bulanık Mühendislik Laboratuvarı

LOC **:** Lines of Code – Kod Satır Sayısı

LWL **:** Locally Weighted Learning – Yerel Ağırlıklı Öğrenme

MAE **:** Mean Absolute Error – Ortalama Mutlak Hata

MAPE Mean Absolute Percentage Error – Ortalama Mutlak Hata Yüzdesi

MLP **:** Multi Layer Perceptron – Çok Katmanlı Algılayıcı

MMRE **:** Mean Magnitude of Relative Error – Göreceli Hatanın Ortalama Büyüklüğü

MÖ **:** Makine Öğrenmesi

MRE **:** Magnitude Relative Error – Göreceli Hatanın Büyüklüğü

NASA **:**National Aeronautics and Space Administration – Ulusal Havacılık ve Uzay Dairesi

PDIF **:** Platform Difficulty – Platform Zorluğu

PERS **:** Personnel Capability – Personel Yeteneği

PRED **:** Prediction Accuracy – Tahmin Doğruluğu

PREX **:** Personal Experience – Personel Deneyimi

PROMISE **:** Predictor Models in Software Engineering – Yazılım Mühendisliğinde Tahmin Modelleri

PSO **:** Particle Swarm Optimization – Parçacık Sürü Optimizasyonu

RAE **:**Relative Absolute Error – Bağıl Mutlak Hata

RCPX **:** Product Reliability and Complexity – Ürün Doğruluğu ve Karmaşıklığı

RMSE **:** Root Mean Squared Error – Kök Ortalama Kare Hata

RRSE **:** Root Relative Squared Error – Kök Bağıl Kare Hata

SCED **:** Development Schedule Constraint – Geliştirme Takvimi Kısıtı

SDR **:** SoftLab Data Repository – Softlab Veri Deposu

SMO **:** Sequential Minimal Optimization – Sıralı Minimal Optimizasyon

SU **:** Factor Based On The Cost Of Software Understanding – Yazılımı Anlama Maliyetini Yansıtan Etken

TDI **:** Total Degree of Influence – Toplam Etki Derecesi

TSP Travelling Salesman Problem – Gezgin Satıcı Problemi

UCI **:** University of California, Irvine – Kaliforniya Univsersitesi, Irvine

USC **:** University of South California – Güney Kalforniya Ünviersitesi

WEKA **:** Waikato Environment for Knowledge Analysis – Bilgi Analizi için Waikato Ortamı

WHO **:** World Health Organization – Dünya Sağlık Örgütü

YSA **:** Yapay Sinir Ağları

DENKLEM LİSTESİ

ÖZET

DOKTORA TEZİ

Şükran EBREN KARA

YAPAY ZEKÂ YÖNTEMLERİ İLE YAZILIMLARIN MALİYETLERİNİN TAHMİN EDİLMESİ

İstanbul Üniversitesi-Cerrahpaşa

Lisansüstü Eğitim Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

Danışman : Prof. Dr. Rüya ŞAMLI

Yazılım maliyet tahmini, bir mühendisin yazılım projesini geliştirmeye başladığı esnada ihtiyaç duyduğu yaklaşık süre ve kaynakların tahminidir. Yazılım maliyet tahmini, yazılım projesinin maliyetini belirlemek ve müşteriyi ikna etmek için yazılım geliştirme sürecindeki en önemli aşamalardan birisidir. Gerçek maliyete en yakın maliyet tahminini yapmak hem yazılım geliştiricileri hem de müşteriler için çok büyük bir önem arz etmektedir. Çünkü yanlış yapılan yazılım maliyet tahminleri projelerin tamamlanamamasına ya da geniş bir zaman dilimine yayılmasına neden olmaktadır. Bu yüzden yazılım maliyet tahmini için literatürde çok farklı yöntem geliştirilmiştir. Bu tez çalışmasında, yazılım projelerinin maliyeti, Yapay Zekâ yöntemlerinden olan Makine Öğrenmesi (MÖ) kullanılarak Parçacık Sürü Optimizasyonu (PSO) ve Genetik Algoritmalarla (GA) öznitelik seçimi yapılarak tahmin edilmeye çalışılmıştır. Yazılım projesinin maliyet tahmini, WEKA (Waikato Environment for Knowledge Analaysis – Bilgi Analizi için Waikato Ortamı) veri madenciliği aracında bulunan algoritmaların çalıştırlması sonucu bulunmuştur. Algoritmalar 10 kat çapraz doğrulama tekniği ile PROMISE (Predictor Models in Software Engineering – Yazılım Mühendisliğinde Tahmin Modelleri) veri deposundan alınan 9 adet veri setine (COCOMO81, COCOMONASA, COCOMONASA2, China, Albrecht, Finnish, Kemerer, Maxwell, Miyazaki94) uygulanmış ve sonuçlar performans ölçütü korelasyon katsayısı, hata oranları MAE (Mean Absolute Error – Ortalama Mutlak Hata), RMSE (Root Mean Squared Error – Kök Ortalama Kare Hata), RAE (Relative Absolute Error – Bağıl Mutlak Hata), RRSE (Root Relative Squared Error – Kök Bağıl Kare Hata) ve MAPE (Mean Absolute Percentage Error – Ortalama Mutlak Hata Yüzdesi) baz alınarak değerlendirilmiştir.
Mayıs 2022, 128 sayfa.

Anahtar kelimeler: Yapay Zekâ, Makine Öğrenmesi, Yazılım Maliyet Tahmini, WEKA, Öznitelik Seçimi, Genetik Algoritmalar, Parçacık Sürü Optimizasyonu

SUMMARY

Ph.D. THESIS

ESTIMATING SOFTWARE COSTS BY ARTIFICIAL INTELLIGENCE METHODS

Şükran EBREN KARA

Istanbul University-Cerrahpasa

Institute of Graduate Studies

Department of Computer Engineering

Supervisor : Prof. Dr. Rüya ŞAMLI

A software cost estimation is an estimate of the time and resources an engineer will need to begin developing a software project. Software cost estimation is important to determine the cost of the software project and to convince the customer. Making a cost estimation closest to the actual cost is very important for both software developers and customers, because incorrect software cost estimations cause projects to be incomplete or spread over a large time period. Therefore, many different methods have been developed for software cost estimation. When the studies in the literature are examined, it is seen that the cost of software projects is tried to be estimated with very different methods. In this thesis, the cost of software projects has been estimated using Machine Learning from Artificial Intelligence methods by features selection with Particle Swarm Optimization (PSO) and Genetic Algorithms (GA). Project cost estimation was made by testing Machine Learning algorithms in WEKA (Waikato Environment for Knowledge Analaysis) data mining tool. Algorithms were applied to 9 datasets (COCOMO81, COCOMONASA, COCOMONASA2, China, Albrecht, Finnish, Kemerer, Maxwell, Miyazaki94) taken from PROMISE (Predictor Models in Software Engineering) data store with 10-fold cross validation technique and results, performance criterion correlation coefficient, error rates MAE (Mean Absolute Error), RMSE (Root Mean Squared Error), RAE (Relative Absolute Error), RRSE (Root Relative Squared Error), MAPE (Mean Absolute Percentage Error).
May 2022, 128 pages.

Keywords: Artificial Intelligence, Machine Learning, Software Cost Estimation, WEKA, Feature Selection, Genetic Algorithms, Particle Swarm Optimization Algorithm

1. GİRİŞ

Yazılım, insanların birçok işini kolaylaştırmak üzere farklı programlama dillerinin kullanılması sonucunda bilgisayar ve benzerî makinelerin beklenilen işlemleri gerçekleştirmesi için üretilen kod kümeleridir. Yazılımlar, makinelerin makinelerle ya da makinelerin insanlarla olan bilgi alışverişini sağlar. Yazılım bir nevi cansız bir varlığın dile gelmesi, bir donanım parçasının konuşması, derdini anlatması olarak ifade edilebilir. Bir sistemde donanımsal bir arıza oluştuğunda, bunu bir insana ya da kendi gibi bir makineye bildirmesi mümkün olmadığından bu işlemi yazılım gerçekleştirmelidir. Türk Dil Kurumu'na (TDK) göre yazılım ise "bir bilgisayarda donanıma hayat veren ve bilgi işlemde kullanılan programlar, yordamlar, programlama dilleri ve belgelemelerin tümüdür" (TDK, 2018). Yazılımları farklı şekillerde gruplandırmak mümkündür, en önemli gruplardan biri Sistem yazılımları/uygulama yazılımları olarak ifade edilebilir. Sistem yazılımları; bilgisayarın, bilgisayar üzerindeki bir donanımın veya bilgisayar üzerindeki bir yazılımın çalışması için mutlak gerekli yazılımlardır; uygulama yazılımları ise insanların kullanması için tasarlanan, çeşitli uygulamaları grçekleştiren ve bilgisayar sistemi için mutlak gerekli olmayan yazılımlardır.

Yazılımlar geliştiren kişilere genellikle Yazılım Mühendisi unvanı verilmektedir. Yazılım Mühendisi, yazılımın talep edilmesinden teslim edilmesine hatta teslim edildikten sonraki süreçler de bile aktif bir şekilde görev alan kişidir. Yazılım mühendisinin amacı; isteklere cevap veren, kaliteli, güvenli, kullanışlı bir yazılımın en az hata ile ortaya çıkmasını sağlamaktır. Sommerville'a (2000) göre Yazılım Mühendisliği, yazılım üretiminin tüm yönleriyle ilgilenen bir mühendislik disiplinidir. Yazılım projelerinde proje yöneticileri için yazılım maliyet tahmini oldukça mühim bir problemdir. Yazılım projesinin geliştirilmesi sırasında gereken kaynakların değeri yazılım maliyeti olarak tanımlanmaktadır (Sommerville, 2000). Yazılım projelerinin maliyet tahmini ise yazılımın gerçekleştirilmesi sürecinde gereken bütçenin tahmin edilmesidir. Yazılım maliyetini hesaplamak kolaydır, zor olan maliyet tahmininde bulunmaktır. Yazılım sistemlerinin oluşturulması sırasında gerekli olan eforun tahmin edilmesi aşamasına yazılım projelerinin maliyet tahmini denir (Adailer, 2008). İnsanların yazılımlara olan ihtiyaçları gün geçtikçe artmaktadır. İhtiyaçlara cevap vermek için

geliştirilen yazılımlar daha büyük ve daha karmaşık olmaktadır. Bunun sonucunda yazılım maliyetini hesaplamak için farklı yöntemler geliştirilmiştir.

Bu tez çalışmasında Yapay Zekâ yöntemleri ile yazılım maliyet tahmini gerçekleştirilmiştir. Bu kapsamda MÖ, GA, PSO ve öznitelik seçimi kullanılmıştır. Bu tez çalışmasının amaçları aşağıdaki şekilde maddelendirilebilir:

- Yazılım projelerinin maliyet tahminin PROMISE veri deposundan temin edilen literatürde sıklıkla kullanılan farklı veri setleri üzerinde MÖ algoritmalarının kullanılarak yapılması,
- Her bir veri seti üzerinde farklı MÖ algoritmasının performans analizlerinin yapılması ve sonuçlarının detaylı yorumlanması,
- Yazılım projelerinin maliyet tahmininde Genetik Programlamanın kullanılabilir olup olmadığının belirlenmesi,
- Yazılım projelerinin maliyet tahmininde GA ve PSO kullanılarak öznitelik seçiminin etkisinin araştırılması,
- Veri setlerinde yer alan hangi özniteliklerin bir arada kullanıldığında veya hangi özniteliklerin önemli hangi özniteliklerin seçilen algoritma için önemsiz olduğunun bunun sonucu olarak da hangi algoritmaların başarı oranlarının daha yüksek olduğunun belirlenmesi,
- Elde edilen bulguların literatürdeki diğer çalışmalar ile karşılaştırılması,

Bu tez çalışması şu şekilde organize edilmiştir:

- 2. Bölüm olan Genel Kısımlar Bölümü'nde yazılım maliyet tahmini, yazılım maliyet tahmininin önemi, yazılım maliyet tahminin gerçekleştirilmesi, yazılım maliyet tahminin yöntemleri, Yapay Zekâ kavramı, Yapay Zekâ yöntemleri, öznitelik seçimi ve tez konusu kapsamında literatürde bugüne kadar yapılmış çalışmalar alt başlıklar halinde sunulmuştur.
- 3. Bölüm olan Malzeme ve Yöntem Bölümü'nde; tez çalışması kapsamında kullanılan veri setleri, uygulama platformu, MÖ algoritmaları ve oluşturulan model sunulmuştur.

- 4. Bölüm olan Bulgular Bölümü'nde yapılan çalışmalar dört kısımda incelenebilir. İlk kısımda literatürdeki diğer çalışmalarla karşılaştırma yapılabilmesi amacıyla PROMISE veri deposundan temin edilen veri setlerine, WEKA programında bulunan MÖ algoritması uygulanarak yazılım maliyet tahmini gerçekleştirilmiştir. Veri setlerine uygulanan algoritmalar 10 kat çapraz doğrulama tekniği ile test edilmiş ve test sonuçları ölçüt olarak korelasyon katsayısı, hata oranları MAE, RAE, RMSE, RRSE ve MAPE baz alınarak değerlendirilmiştir. İkinci kısımda ise WEKA programında bulunan MÖ ve Evrimsel Algoritma: Genetik Programlama; Albrecht, Finnish, Kemerer, Maxwell ve Miyazaki94 veri setleri üzerinde iki şekilde çalıştırılmıştır. İlk olarak veri setleri üzerinde hiçbir öznitelik seçimi gerçekleştirilmeden MÖ algoritmaları çalıştırılmış ve yazılım maliyet tahmini yapılmıştır. İkincisinde, her bir veri seti üzerinde ilk önce GA kullanılarak öznitelik seçimi gerçekleştirilmiştir. Veri setlerine uygulanan öznitelik seçiminden sonra bazı öznitelikler veri setlerinden kaldırılmıştır. Veri setinde geri kalan öznitelikler ile MÖ algoritmaları kullanılarak yazılım maliyet tahmini yapılmıştır. Üçüncü kısımda Maxwell, China ve COCOMONASA veri setleri üzerinde öznitelik seçim metotlarından GA ve PSO kullanılarak yazılım maliyet tahmini yapılmıştır. Bu sayede hem öznitelik seçim metotlarının yazılım maliyet tahmini üzerindeki etkileri incelenmiş hemde algoritmaların performansları hesaplanıp karşılaştırılmıştır. Dördüncü kısımda yazılım maliyet tahmini için yapılan önceki çalışmalar incelenmiş ve bulgular karşılaştırılmıştır.
- 5. Bölüm olan Tartışma ve Sonuç Bölümü'nde; her bir veri setinde gerçeklenen MÖ algoritmalarının performans sonuçlarından bahsedilmiştir.

2. GENEL KISIMLAR

Bu bölümde yazılım maliyet tahmini, yazılım maliyet tahmini yöntemleri, Yapay Zekâ, Yapay Zekâ yöntemleri, öznitelik seçimi açıklanmış ve konu ile ilgili ayrıntılı bir literatür taraması sunulmuştur.

2.1. YAZILIM MALİYET TAHMİNİ

Yazılım geliştirme projelerinin en mühim kısımlarından biri, yazılım projelerinin maliyet tahminidir. Yazılım projesinin geliştirilmesi sırasında gereken kaynakların değeri yazılım maliyeti olarak tanımlanmaktadır. Sommerville (2000) yazılım maliyet bileşenlerini aşağıdaki gibi belirtmiştir:

- Ekipman ve yazılım giderleri
- Yolculuk ve eğitim giderleri
- İş gücü giderleri
  - ✓ Projeye dahil edilmiş mühendislerin ödemeleri
  - ✓ Sosyal ve sigorta giderleri
  - ✓ İş gücü giderini etkileyen diğer faktörler
- Barınma gideri
- İletişim gideri
- Birlikte kullanım giderleri

Ayyıldız'a (2007) göre ortaya çıkarılacak bir ürünün veya bir hizmetin bedelinin ne olacağına sayısal olarak tahmin edilmesi işine maliyet tahmini denir. Yazılım projelerinin maliyet tahmini ise yazılım sistemlerinin oluşturulması sırasında gerekli olan eforun tahmin edilmesi aşamasına denir (Adailer, 2008). Farklı tanımlamalar olmakla beraber tüm tanımlamalarda tahminleme işinin ortada henüz bir yazılım yokken yapılması gerektiği, bu durumun da oldukça güç bir durum olduğu fikri ortaktır.

Proje maliyet yönetiminde maliyet tahmini için gerekli girdiler, çıktılar, araçlar ve teknikler Tablo 2.1'de belirtilmiştir (PMBOK, 2000).

Tablo 2.1: Maliyet tahmini için girdiler, araçlar ve teknikler, çıktılar.

GİRDİLER	ARAÇLAR VE TEKNİKLER	ÇIKTILAR
1. İş Dağılımı Yapısı 2. Kaynak Gereksinimleri 3. Kaynak Oranları 4. Faaliyet Süresi Tahminleri 5. Tahmin Yayınları 6. Tarihi Bilgi 7. Hesap Tablosu 8. Riskler	1. Benzer Tahmin 2. Parametrik Modelleme 3. Aşağıdan Yukarıya Tahmin 4. Bilgisayarlı Aletler 5. Diğer Maliyet Tahmin Yöntemleri	1. Maliyet Tahminleri 2. Destekleyici Detaylar 3. Maliyet Yönetimi Planı

2.1.1. Yazılım Maliyet Tahmininin Önemi

Yazılım projesinin maliyet tahmininin gerçekleştirilmesi, yazılım projesinin önerilmesinde, onaylanmasında ve geliştirilmesindeki pek çok kararı olumlu yönde etkilemektedir. Proje yöneticisinin, yazılım maliyetini doğru tahmin etmesi yazılım projesindeki belirsizlikleri ortadan kaldırır. Aksi durumda çok ciddi maddî sıkıntılar baş göstermektedir. Maliyet tahmini, proje öncesinde mevduat miktarının gerçekçi bir şekilde belirlenebilmesi, bazı kararların daha mantıklı alınabilmesi, yazılım şirketlerinin teklif fiyatını doğru şekilde belirleyebilmesi ve rekabet edebilir bir fiyat saptayabilmesi gibi sebeplerden dolayı çok önemlidir. Yazılım projelerinin maliyet tahmininin ne kadar önemli olduğu aşağıdaki gibi belirtilmiştir (Leung ve Fan, 2002):

- Genel bir iş planına göre geliştirme projelerini gruplandırmaya ve önem sırasına göre belirtmeye yardım eder.
- Gerçekleştirilen değişikliklerin etkisini değerlendirmeyi sağlar ve tekrar planlama imkânı sunar.
- Kaynaklar gerçek ihtiyaçlar ile eşleştirildiğinde projelerin yönetimi ve kontrolünü kolaylaştırır.
- Müşterilerin gerçek geliştirme maliyetinin, tahmin edilen maliyetle uyumlu olmasına dair beklentisine cevap verir.

2.1.2. Yazılım Maliyet Tahminin Gerçekleştirilmesi

Yazılım sektörü sürekli gelişerek kendini yenileyen bir sektördür. Piyasanın artan talepleri daha karmaşık yazılımların geliştirilmesine neden olmaktadır. Yazılım projeleri her zaman başarılı

bir şekilde sonuçlanamamaktadır. Bu başarısızlığın sebeplerini Sezer (2008) çalışmasında; proje sınırlarının uygun bir şekilde saptanamaması, gerçeğe yakın maliyet tahmininin gerçekleştirilememesi, dönüşen müşteri isteklerinin karşılanamaması, personelin teknik donanımının eksikliği, müşterinin isteklerini tam olarak belirtememesi, bunlardan en önemlisi hiç kuşkusuz yanlış maliyet tahminlemesi olarak sıralamıştır. Dünya genelinde yazılım maliyetlerinin yanlış tahminlemesi ve projenin zaman aşımından dolayı, çoğu proje yarım bırakılmış ya da çökmüştür. Yazılım projelerinin maliyet tahmini, proje yöneticileri için çok zor bir süreçtir. Bu sürecin uygun bir şekilde işlemesi, ciddi bir yazılım maliyet tahmin yöntemi gerektirmektedir. Yazılım maliyet tahmin sürecinin başarı oranının artırmak, çok miktarda veri ile doğru analizlerin yapılmasına bağlıdır. Barry Boehm'in (Boehm, 1981) ve NASA'nın (National Aeronautics and Space Administration-Ulusal Havacılık ve Uzay Dairesi) sunduğu iki süreç bunlardan bazılarıdır (NASA, 2003). Bu süreçler Tablo 2.2 ve 2.3'te belirtilmiştir.

Tablo 2.2: Barry Boehm maliyet hesaplama süreci.

1.Adım: Maliyet tahmin adımları belirlenir.
2.Adım: Gerekli veriler ve kaynaklar için proje planı oluşturulur.
3. Adım: Yazılım gereksinimleri sabitlenir.
4. Adım: Mümkün olduğu kadar fazla detay verilir.
5.Adım: Farklı maliyet tahmin teknikleri ve kaynakları kullanılır
6.Adım: Farklı tahminler karşılaştırılır ve yinelenir.
7.Adım: İzleme gerçekleştirilir.

Tablo 2.3: Nasa'nın maliyet hesaplama süreci.

1.Adım: Yazılımı fonksiyonel ve programlı olarak analiz edip bilgi toplanır.
2. Adım: İş elemanlarını tedarikleri tanımlanır.
3.Adım: Yazılım boyutunu tahmin eder.
4.Adım: Yazılım eforunu tahmin eder.
5.Adım: Harcanacak eforu planlanır.
6.Adım: Risklerin etkisini belirlenir.
7.Adım: Tahminleri modellerle doğrulanır.
8.Adım: Tahminleri, bütçeyi ve zamanlamayı ayarlar.
9.Adım: Tahminleri inceler ve onaylar.
10.Adım: İzleme, raporlama ve bakım yapma aşamaları gerçekleşir.

2.1.3. Yazılım Maliyeti Tahmin Yöntemleri

Yazılım maliyeti tahmin modelleri literatürde farklı şekillerde kategorize edilmiştir. Attarzadeh ve Ow (2010)'un çalışmalarında yazılım projelerinin maliyet tahmin yöntemleri, algoritmik yöntemler ve algoritmik olmayan yöntemler olarak sınıflandırılmıştır. Literatürde yazılım

projelerinin maliyet tahmin modelleri için oldukça fazla araştırma yapılmıştır. Bu çalışmada yazılım maliyeti tahmin yöntemleri tüm kategorilerin birleşimi olacak şekilde aşağıdaki gibi kategorize edilmiştir:

- Algoritmik Yöntemler (Regresyon veya İstatistiksel Yöntemler)
- Algoritmik Olmayan Yöntemler
- Melez Yöntemler

Farklı yazılım maliyeti tahmin modelleri olsa da kesinlikle birinin diğerinden daha üstün olduğunu söylemek mümkün değildir. Her modelin başarılı olduğu ya da başarılı sonuçlar veremediği durumlar söz konusudur. Bazen daha doğru sonuçların elde edilebilmesi için bu modellerin bir kombinasyonuna başvurulması, her birinin tahminlerinin dikkatlice karşılaştırılması ve yinelenmesi önemlidir.

2.1.3.1. Algoritmik Yöntemler

Algoritmik yöntemler, yazılım maliyet tahmini için matematiksel bir formül kullanmaktadır (Kumari ve Pushkar 2013). Parametre olarak proje büyüklüğü, proje süresi, yazılım mühendisi sayısı, kod satır sayısı gibi girdiler ile efor ve maliyet tahminini matematiksel denklemler ve fonksiyonlar yardımı ile bulmaya çalışırlar. Putnam Modeli (SLIM), Fonksiyon Noktası Analizi ve COCOMO (Constructive Costing Model – Yapı Maliyet Modeli) bazı popüler algoritmik modeller arasında yer almaktadır.

Putnam Modeli:

Putnam modelinde zamana endeksli emek ve maliyet eğrileri mevcuttur (Putnam, 1978). Gerçek bir projede adam x ay değeri zaman içinde stabil kalmaz. Bundan dolayı bazı anlardaki kişi sayısı ihtiyacı, diğer anlara göre farklılık gösterir. Putnam modelinde emek-zaman eğrisine bakarak kişi sayısı ayarlanabilir. Aynı birim içerisinde farklı projeler arasında personel değişimi Putnam eğrilerinin bir sonucu olarak yapılabilir (Ayyıldız, 2007). Denklem 2.1'de kod satır sayısı verilmiştir.

$$S = Ex(Çaba)^{1/3} {t_d}^{4/3} \quad (2.1)$$

Burada t_d: yazılım teslim süresi, E: yazılım geliştirme yeteneğini yansıtan çevre faktörü, *S:* kaynak kodlarını ifade etmektedir. Denklem 2.2'de çaba formülü verilmiştir.

$$Çaba = Dxt^3 \tag{2.2}$$

Burada D: yeni bir yazılım ya da yeniden oluşturulmuş yazılım arasında değişen insan gücünü ifade eden bir parametredir (0 – 8, 8 – 27 arasında değer almaktadır). Yukarıdaki iki denklem birleştirilerek, yazılım maliyet tahmini için Denklem 2.3 ve Denklem 2.4 elde edilir.

$$Çaba = \left({D_0}^{4/7} xE^{-9/7}\right) xS^{9/7} \tag{2.3}$$

$$t_d = \left({D_0}^{-1} xE^{-3/7}\right) xS^{3/7} \tag{2.4}$$

Burada E çevre faktörüdür. Bu modelin diğer bir avantajı da önceden oluşturulmuş proje verilerinden boyut, efor ve süre kullanılarak kolay oluşturulabilir olmasıdır.

Fonksiyon Noktası Analizi:

Fonksiyon Nokta Analizi, kod satır sayısı yaklaşımına alternatif olarak geliştirilmiştir (Albrecht, 1979). Satır sayısı tekniğinden farklı olarak bir yazılım kurumu için direk büyüklük tahmininde bulunmanın bir zorluğu yoktur çünkü gereksinimlerin belirlenmesi faaliyetlerinde bulunan değerlerden yazılımın büyüklüğü bilgisine ulaşılabilir (Ayyıldız, 2007; Sezer 2008). Fonksiyon Nokta Analizi, yazılımdaki fonksiyonları karmaşıklıkları ve yaptıkları işlere göre sınıflandırıp saymaktadır. Bunu yapmak, yöneticilerin verimliliği takip edebilmelerini ve yazılım geliştirme maliyetlerini tahmin edebilmelerini sağlamaktadır (Keskin, 2016). Fonksiyon Nokta Analizi modelinde ilk önce AFN (Ayarlanmış Fonksiyon Noktası) hesaplanmaktadır. AFN değeri Tablo 2.4 yardımı ile hesaplanır. AFN değerini elde ettikten sonra DAF (Değer Ayarlama Faktörü) değeri hesaplanır. Denklem 2.5'de AFN formülü verilmiştir.

$$AFN = [|Haricî\ girdi|\ x\ w1] + [|Haricî\ çıktı|\ x\ w2] + [|Haricî\ sorgu|\ x\ w3] + [|Dahilî\ dosya|\ x\ w4][|Haricî\ arayüz|\ x\ w5] \tag{2.5}$$

Burada w yapılan işi göstermekedir.

Tablo 2.4: Bileşenlerin karmaşıklıklarına göre sınıflandırılması.

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

AFN değerlerinin ardından, DAF değerleriyle, son işlev puanları hesaplanabilir. DAF, 0 (en düşük) ve 5 (en yüksek) arasında seviyeleri arasında değerlendirilen 14 genel sistem özelliğinden oluşmaktadır. Bu 14 değerin toplamı toplam etki derecesini TDI (Total Degree of Influence – Toplam Etki Derecesi) vermektedir. Denklem 2.6'da TDI, Denklem 2.7'de DAF ve Denklem 2.8'de FN (Fonksiyon Noktası) verilmiştir.

$$TDI = \sum_{i=1,2,\ldots,14} Cevap_i \tag{2.6}$$

Genel sistem özellikleri (Ayyıldız, 2007) şekil 2.1'de aşağıdaki soruların cevapları şeklinde tanımlanmıştır.

- Güvenilir yedekleme ve kurtarma işlemi gerekli mi?
- Veri iletişimi gerekiyor mu?
- Dağıtık işlem ve süreçler var mı?
- Çabukluk önemli mi?
- Sistem mevcut ve fazla yüklü bir ortamda mı çalışacak?
- Çevrimiçi (on-line) veri girişi gerekecek mi?
- Çevrimiçi (on-line) giriş, fazla ekranlı veya fazla işlemli mi?
- Ana kütükler çevrimiçi (on-line) olarak mı güncellenecek?
- Girdi, çıktı, sorgulama ve kütükler karmaşık mı?
- İç süreç (internal process) karmaşık mı?
- Program yeniden kullanılabilir olarak mı tasarlanıyor?
- Dönüştürme (conversion) ve kurma (installation), tasarımın içinde yer alıyor mu?
- Değişik kuruluşlarda çoklu kurmalar tasarlanıyor mu?
- Kullanıcının kolaylığı ve uyarlamasına göre tasarlanıyor mu?

Şekil 2.1: Genel sistem özellikleri.

$$DAF = 0{,}65 + (0{,}01\ x\ TDI) \quad (2.7)$$

$$FN = AFN\ x\ DAF \quad (2.8)$$

FN, AFN ile DAF değerlerinin çarpımına eşittir.

COCOMO:

Barry W. Boehm 1981 yılında algoritmik yazılım maliyet tahmin modeli olan COCOMO modelini geliştirmiştir. Bu modele COCOMO 81 modeli de denmektedir. İlk COCOMO, yazılım geliştirme süreci olarak şelale (Waterfall) modelini ve programlama dili olarak prosedürel diller kullanmıştır. Daha sonraları ihtiyaçlara binaen COCOMO'nun farklı sürümleri çıkmıştır. Bunlar COCOMO 81, COCOMO II gibi modellerdir.

COCOMO 81: COCOMO modelinin ilk sürümüdür. COCOMO 81 modelinin, hesaplanacak yazılım maliyet tahminlerinin kapsamlarına göre Basit COCOMO, Orta Düzey COCOMO ve Detaylı COCOMO olarak üç değişik modeli geliştirilmiştir. Her modelin kullanacağı probleme göre organik, yarı ayrık ve gömülü modlar mevcuttur. Organik modda küçük bir ekip aşina oldukları bir ortamda iyi anlaşılmış proje uygulamaları geliştirir. Bu ekip belli bir deneyime sahip işlerini hızlı yapan kişilerdir. Yarı ayrık modda ekipte deneyimli ve deneyimsiz elemanlar bulunabilir. Bunlar sistemin her aşamasını bilmeyebilir ve sistemle ilgili bilgileri yetersiz olabilir. Gömülü modda projeleri katı donanım, yazılım, yönetmenlikler ve işlem kısıtlayıcılar seti içerisinde geliştirilmelidir. Yazılım üzerinde yapılacak değişiklikler o kadar maliyetlidir ki değişmez olarak kabul edilir. Bu yüzden sonradan çıkabilecek değişiklikler ve ön görülemeyen güçlükler üzerinde ciddi çalışılması gerekir. Projedeki elemanların projedeki tüm kısımlara hakim olması olanaksızlaşmıştır.

COCOMO modeli ve problem türü belirlendikten sonra ilgili formüller kullanılarak tahmin hesaplama yoluna gidilir. Basit COCOMO modelinin hızlı ve kolay kullanımından dolayı küçük ve orta ölçekli yazılım geliştirme projelerine uygulanabilir. Kullanılan formüllerin temelinde kod satır sayısı KLOC (Kilo Line OF Code – Kilo Kod Satır Sayısı) vardır. Basit COCOMO bileşenleri Tablo 2.5'te, formülü Tablo 2.6'da ve modlara uygulanışı Tablo 2.7'de belirtilmiştir.

Tablo 2.5: Basit COCOMO bileşenleri.

Problem	a	b	c	d
Organik	2,4	1,05	2,5	0,38
Yarı ayrık	3,0	1,12	2,5	0,35
Gömülü	3,6	1,2	2,5	0,32

Tablo 2.6: Basit COCOMO modelinin formülü.

Çaba (ayAdam)	$a\ x\ (KLOC)^b$
Geliştirme Zamanı (Ay)	$c\ x\ (Çaba)^d$
Verimlilik	$KLOC\ /\ Çaba$
Ortalama istihdam	$Çaba\ /\ Geliştirme\ Zamanı$

Tablo 2.7: Basit COCOMO modelinin modlara uygulanışı.

Problem	Çaba	Süre
Organik	$Çaba = 2{,}4\ (KLOC)^{1{,}05}$	$Süre = 2{,}5\ (Çaba)^{0{,}38}$
Yarı ayrık	$Çaba = 3\ (KLOC)^{1{,}12}$	$Süre = 2{,}5\ (Çaba)^{0{,}35}$
Gömülü	$Çaba = 3{,}6\ (KLOC)^{1{,}20}$	$Süre = 2{,}5\ (Çaba)^{0{,}32}$

Burada, $KLOC$ 1000 Kod Satır Sayısı. a, b, c, d: Basit COCOMO Modeli Bileşenleri'dir.

Orta Düzey COCOMO'da da basit COCOMO tahmininde olduğu gibi bir çaba tahmini oluşturularak formüle yerleştirilir. Çaba tahmini oluşturulurken basit COCOMO'dan farklı olarak formülün içine EAF (Effort Adjustment Factor – Maliyet Faktörü Çarpanı) girer. EAF, her bir özniteliğin yazılım geliştirme çabası üzerindeki etkisinin çarpılması ile elde edilir. Öznitelikler, yazılım ürün öznitelikleri, bilgisayar öznitelikleri, personel öznitelikleri ve proje öznitelikleri olarak dört kategoriye ayrılır. Bu öznitelikler 15 maliyet faktörü içerir. 15 maliyet faktörü çok az, az, normal, yüksek, çok yüksek, aşırı yüksek gibi faktör değerleri alırlar. Orta Düzey COCOMO'da gerekli zaman hesabı ise Basit COCOMO modelinde olduğu gibi yapılır. Tablo 2.8'de Orta Düzey COCOMO için öznitelikler ve faktörleri, Tablo 2.9'da Orta Düzey COCOMO bileşenleri, Tablo 2.10'da Orta Düzey COCOMO formülleri belirtilmiştir.

Tablo 2.8: Orta Düzey COCOMO Modeli'nin maliyet faktörleri.

Kategori	Maliyet Faktörü	Açıklama	Çok Az	Az	Normal	Yüksek	Çok Yüksek	Aşırı Yüksek
Örün Özellikleri	Rely	Yazılımın güvenirliği	0,75	0,88	1,00	1,15	1,40	
	Size	Veri tabanın büyüklüğü		0,94	1,00	1,08	1,16	
	Cplx	Karmaşıklık	0,70	0,85	1,00	1,15	1,30	1,65
Donanım Özellikleri	Time	İşletim zamanı kısıtı			1,00	1,11	1,30	1,66
	Stor	Ana bellek kısıtı			1,00	1,06	1,21	1,56
	Virt	Sanal makine oynaklığı		0,87	1,00	1,15	1,30	
	Turn	Bilgisayar dönme zamanı		0,87	1,00	1,07	1,15	
Personel Özellikleri	Acap	Analist deneyimi	1,46	1,19	1,00	0,86	0,71	
	Aexp	Uygulama deneyimi	1,29	1,13	1,00	0,91	0,82	
	Pcap	Programcı tecrübesi	1,42	1,17	1,00	0,86	0,70	
	Vexp	Sanal makine uzmanlığı	1,21	1,10	1,00	0,90		
	Lexp	Dil tecrübesi	1,14	1,07	1,00	0,95		
Proje Özellikleri	Modp	Modern programlama deneyimi	1,24	1,10	1,00	0,91	0,82	
	Tool	Yazılım geliştirme araçları kullanımı	1,24	1,10	1,00	0,91	0,83	
	Sched	Zamanlama kısıtlamaları	1,23	1,08	1,00	1,04	1,10	

Tablo 2.9: Orta Düzey COCOMO bileşenleri.

Problem	a	b	c	d
Organik	3,2	1,05	2,5	0,38
Yarı ayrık	3,0	1,12	2,5	0,35
Gömülü	2,8	1,2	2,5	0,32

Tablo 2.10: Orta Düzey COCOMO Modelinin formülü.

EAF	$(maliyet\ faktörü1\ x\ maliyet\ faktörü2\ x \ldots x\ maliyet\ faktörü15)$
Çaba	$a\ x\ (KLOC)^b\ x\ EAF\ (problemin\ türüne\ göre\ a\ ve\ b\ değerleri\ belirlenir)$
Geliştirme zamanı	$c\ x\ (Çaba)^d$
Ortalama çalışan sayısı	$Çaba/Geliştirme\ zamanı$

Burada EAF, Maliyet Faktör Çarpanı, $KLOC$, 1000 Kod Satır Sayısı. $a, b, c, d,$ Orta Düzey COCOMO Modeli Bileşenleri'dir.

Detaylı COCOMO modeli basit COCOMO ve orta COCOMO modellerinden farklı olarak iki özellik daha barındırır. Birincisi, aşama ile ilgili işgücü çarpanları, ikincisi, yazılım maliyet kestirimidir. Bu model projenin aşamalarına göre zaman içinde oluşan farklılıkları göz önünde

bulundurarak ara ara yazılım maliyet tahmini gerçekleştirir. Detaylı COCOMO modelinde zamana bağlılık temel değişikliktir. Yapılacak işin karmaşıklığı ve çaba yoğunluğu projenin farklı aşamalarında değişmektedir (Ayyıldız, 2007).

COCOMO II: COCOMO 81 modelinden sonra modelin gelişimi devam etmiştir. 2000 yılında COCOMO II sürümü yayınlanmıştır (Boehm, 2000). COCOMO II modeli üç seviyeden oluşmuştur. Bunlar erken prototip seviyesi, erken tasarım seviyesi ve mimarîden sonraki seviyedir. Erken prototip seviyesinde değerlendirmeler nesne puanlarına göre hesaplanır ve çabayı değerlendirmek için basit formüller kullanılır. Erken prototip seviyesi, prototip projelerini ve yeniden kullanımı çok sık olan projeleri desteklemektedir. Denklem 2.9'da Çaba formülü verilmiştir.

$$Çaba = \frac{Nesne\ noktası\ sayısı\ x\ \left(1 - \%yeniden\ kullanılabilirlik\ \frac{oranı}{100}\right)}{verimlilik} \tag{2.9}$$

Erken tasarım seviyesinde değerlendirmeler işlev puanları üzerinden yapılmaktadır. Daha sonra bu LOC'a (Lines of Code – Kod Satır Sayısı) dönüştürülmektedir. Değerlendirmeler gereksinimler belirlendikten sonra da yapılabilmektedir. Çapa formülü Denklem 2.10'da ve Denklem 2.11'de verilmiştir. PMm, kod otomatik oluşturulursa kullanılan faktördür. Dolayısıyla gerekli çaba (PMm) Denklem 2.12'de hesaplanır ve çabaya eklenir.

$$Çaba = A\ x\ Size^{B}\ x\ M + PMm \tag{2.10}$$

$$M = PERS\ x\ RCPX\ x\ RUSE\ x\ PDIF\ x\ PREX\ x\ FCIL\ x\ SCED \tag{2.11}$$

$$PMm = \left(ASLOC\ x\ \frac{\frac{AT}{100}}{ATPROD} \right) \tag{2.12}$$

Burada A, 2,5 başlangıç sabiti, Size, 1000 Kod Satır Sayısı, B 1,1 ile 1,24 arasında değişir ve projenin yeniliğine, geliştirme esnekliğine, risk yönetimine, süreç olgunluğuna bağlıdır. Diğer kriterlerin açıklaması ise aşağıdaki gibi verilmiştir:

- PERS: Personnel Capability – Personel Yeteneği,
- RCPX: Product Reliability and Complexity – Ürün Doğruluğu ve Karmaşıklığı,
- PDIF: Platform Difficulty – Platform Zorluğu,
- PREX: Personal Experience – Personel Deneyimi,
- SCED: Development Schedule Constraint – Geliştirme Takvimi Kısıtı,
- FCIL: Facilities – Destek Araç Gereçler,
- ASLOC: Source Line Of Code – Kaynak kodun Satır Sayısı,
- ATPROD: Automated Production – Kod üretiminin üretkenlik seviyesi, (AT/100): Otomatik oluşturulan kodun toplam sistem koduna yüzdesi.

Mimarîden sonraki seviyede erken tasarım seviyesindeki aynı formüller kullanılabilir. Sistemin mimarîsi bitirildiğinde, yazılımın boyutu hakkında isabetli, doğru tahmin yapılabilir. Bu noktada yapılan tahmin, personelin kabiliyetini, ürün ve proje özelliklerini yansıtan daha kapsamlı çarpan kümesi kullanır. Denklem 2.13'de ESLOC (Effective Source Line Of Code – Etkili Kaynak Kodun Satırlarının Sayısı) formülü verilmiştir.

$$ESLOC = ASLOC\ x\frac{AA + SU + 0{,}4DM + 0{,}3CM + 0{,}3IM}{100} \tag{2.13}$$

Burada

- DM: Percentage Of Design Modified – Değiştirilen Tasarım Yüzdesi
- CM: Percentage Of Code Modified – Değiştirilen Kod Yüzdesi
- IM: Percentage Of The Original Integration Effort Required For Integrating The Reused Software – Tekrar Kullanılan Yazılımı Bütünleştirmek İçin Gereken Başlangıç Bütünleşme Çabası Yüzdesi
- SU: Factor Based On The Cost Of Software Understanding – Yazılımı Anlama Maliyetini Yansıtan Etken
- AA: Factor Which Reflects The Initial Assessment Costs Of Deciding If Software May Be Reus – Yazılımın Tekrar Kullanılacağını Dikkate Almakla Maliyetin Başlangıç Tahmini Yansıtan Etken

Regresyon temelli yazılım maliyet tahmini teknikleri literatürde oldukça sıklıkla ele alınan bir konudur (Adalier, 2008). Bu yöntemlerde çeşitli alanlarda, aralarında sebep sonuç ilişkisi bulunan veriler toplanılır ve tablo şekline getirilerek incelenir. Bu veriler arasındaki ilişkiyi belirlemek ve bu ilişkiyi kullanarak tahminleri ya da kestirimleri modelleyen bir fonksiyon bulunmaya çalışılır. Tahminleri ya da kestirimleri en iyi modelleyen bu fonksiyonu bulma sürecine regresyon çözümlemesi denir (Fatullayev, 2013). İstatistiksel yöntemler arasında bulunan regresyon çözümlemesi en çok tercih edilen yöntemlerden biridir. Olası birçok regresyon yönteminin dışında, genellikle matematiksel hesaplamalardaki kolaylığından dolayı, En Küçük Kareler Tekniği optimal tahmin yöntemi olarak tercih edilmektedir (Alma ve Özgül, 2008).

En Küçük Kareler Tekniği birbirine bağlı olarak değişkenlik gösteren iki fiziksel büyüklüğün arasındaki matematiksel bağlantıyı gerçeğe en yakın bir denklem olarak belirmek için kullanılan standart bir regresyon yöntemidir. Bu yöntem eldeki veri noktalarına en yakın geçecek bir fonksiyon eğrisi bulmaya çalışır. Gauss'un bulduğu bu yöntem Cres astroidinin yörüngesinin hesaplanmasında kullanılmıştır (Fatullayev, 2013). Bu hesaplama Denklem 2.14 ve Denklem 2.15'te verilmiştir.

$$y = f(x) = mx + b \tag{2.14}$$

gibi bir doğrusal fonksiyon ya da

$$y = f(x) = ax^2 + bx + c \tag{2.15}$$

gibi karesel fonksiyonda bulunması gereken değerler a, b, c, m'dir.

Burada yi değeri $f(xi)$ için olası değer, $f(xi) \approx yi$, kabul edilince yapılan hata $yi - f(xi)$ dir ve hedef, bu hatalar minimum olacak şekilde bir f fonksiyonu bulmaktır. $yi - f(xi)$ farklarının her birine bir artık denir. En Küçük Kareler Yönteminde aranan fonksiyon, ya da onun parametreleri, tüm artıkların kareleri toplamı olan Denklem 2.16'yı minimum yapacak şekilde belirlenir (Fatullayev, 2013).

$$\sum_{i=1}^{n} \big(y_i - f(x_i)\big)^2 = \big(y_1 - f(x_1)\big)^2 + \cdots + \big(y_n - f(x_n)\big)^2 \tag{2.16}$$

2.1.3.2. Algoritmik Olmayan Yöntemler

Algoritmik olmayan yöntemler maliyet tahmini için bir formül kullanmaz (Kumari ve Pushkar, 2013). Bu yöntemler 1990 yılında ortaya çıkınca yazılım araştırmacıları dikkatlerini MÖ, GA, Bulanık Mantık (BM) ve Yapay Sinir Ağları (YSA) gibi Yapay Zekâ Yöntemleri olan soft computing (esnek hesaplama) denilen yeni yaklaşımlara yöneltmiştir (Attarzadeh ve Ow, 2010). Algoritmik olmayan yöntemlerden bazıları

- Uzmanlık Temelli Yöntemler
- Uzman Görüşü

şeklindedir.

Uzmanlık temelli yöntemler, isminden de anlaşıldığı üzere uzman görüşüne ya da daha önce yapılmış projelere dayanarak yapılan tahmin yöntemleridir. Bu yöntemler daha önce benzeri görülmemiş projeler için ve daha önce yapılmış projelerin somut verileri olmadığı durumlarda kullanılmaktadır (Hihn ve Habib-agahi, 1991). Yapılan tahminin doğruluğu tahmini yapan uzmanın tecrübe alanına ne kadar hâkim olduğuna bağlıdır. Alanına hâkim olmayan, öznel görüşlerini işin içine katan bir tahmincinin objektifliği bulunmayabilir bu da yanlış tahminlerin yapılmasına neden olabilir.

Uzman görüşü bir grup uzmanın deneyimlerini kullanarak tahminde bulunma yöntemidir. Yazılım maliyet tahmini için en kullanışlı yöntemlerden biridir. Genel olarak Delphi tekniği kullanılır. Delphi Tekniği, bilhasa askeri konulara dair kestirimlerde bulunmak üzere Amerika Birleşik Devletleri'nde RAND firmasında çalışan iki araştırmacı tarafından geliştirilmiştir (Dalkey ve Helmer, 1963). Delphi Tekniği başta tıp, yönetim, askeri konular ve eğitimin çok yönlü alanlarında olmak üzere birçok alanda kapsamlı bir şekilde kullanılmaktadır (Woundenberg, 1991; Şahin, 2001). Yazılım projelerinin maliyet tahmini için kullanılan bu teknik, konuyla ilgili uzmanlardan oluşan bir grubun, rasyonalist bir yaklaşımla iki veya daha fazla turda tahmin yaparak ortak görüşlerinin yazılı olarak alınmasına dayanmaktadır. Bu teknikte uzmanların birbirlerinden etkilenmemesi ve tartışma ortamının oluşmaması için uzmanların kimlikleri gizlenir, uzman görüşlerinden ortak görüşler çıkarılır. Delphi Tekniğinin başlıca özellikleri şunlardır:

- ***Katılımda Gizlilik:*** Araştırma boyunca savunulan görüşün kime ait olduğu gizli tutulur Şahin (2001).
- ***Grup Tepkisinin İstatistiksel Analizi:*** Delphi Tekniği ile oluşturulan anketler uygulandıktan sonra her seferinde istatistiksel olarak analiz edilir.
- ***Kontrollü Geri Besleme:*** Delphi Tekniği'nde ardışık anketler uygulanır. Ankette çıkan görüşler katılımcılara yeni anket ile birlikte iletilir. Katılımcılar farklı bakış açıları ile görüşlerini yeniden gözden geçirir. Bu şekilde devam eden teknik ortak bir görüş oluşuncaya kadar devam eder.
- ***Analoji Tabanlı Yöntemler:*** Önceden yapılmış gerçek proje verilerine dayanılarak yeni bir projenin bazı özelliklerinin tahmin edilmesidir. Tamamlanmış projelerden elde edilen gerçek veriler önerilen projeyi tahmin etmek için kullanılabilir (Boehm, 1981; Ayyıldız, 2007; Kumari ve Pushkar, 2013). Bu tekniğin doğru sonuç verebilmesi için önceki proje verilerinin ve karakteristik özelliklerinin saklanmış olması gerekmektedir. Küçük ve orta ölçekli projelerdeki tahminler büyük ölçekli projelere kıyasla daha tatmin edici sonuçlar vermektedir (Adalier, 2008).
- ***Yukarıdan Aşağıya Tahmin Yöntemi:*** Makro Model olarak da adlandırılır. Bu yöntem kullanılarak, projenin global özelliklerinden proje için genel bir maliyet tahmini türetilir ve daha sonra proje çeşitli alt düzey mekanizmalara veya bileşenlere bölünür. Bu yaklaşımı kullanan öncü yöntem Putnam Modeli'dir. Bu yöntem yazılım geliştirmenin ilk evresinde ayrıntılı bilgi bulunmadığında çok yararlıdır.
- ***Aşağıdan Yukarıya Tahmin Yöntemi:*** Bu yöntem kullanılarak her bir yazılım bileşeninin maliyeti tahmin edilir ve daha sonra toplam proje maliyet tahminine ulaşmak için bu tahmin sonuçları birleştirilir. Bu tahmin yöntemi, küçük yazılım bileşenleri ve bunların etkileşimleri hakkında biriken bilgilerden bir sistemin maliyet tahminini oluşturmayı amaçlamaktadır. Bu yaklaşımı kullanan öncü model COCOMO modelidir.
- ***Parkinson Yasaları:*** Parkinson (1957)'deki "iş mevcut hacmi doldurmak için genişler" ilkesini kullanarak maliyetin objektif bir değerlendirmeye dayanmak yerine mevcut kaynaklar tarafından belirlenmesidir. Örneğin yazılım 12 ayda teslim edilmek zorunda ise ve 5 kişi mevcut ise çaba 60 kişi-ay olarak tahmin edilmektedir. Bazen iyi tahminler vermesine rağmen bu yöntem çok gerçekçi olmayan tahminler de sağlayabileceği için pek önerilmez.

- ***Kazanmak İçin Fiyat:*** Yazılım maliyetini, projeyi kazanmak için gereken en iyi fiyat olarak tahmin etmektir. Tahmin yazılımın işlevselliği yerine müşterinin bütçesine bağlıdır. Örneğin bir proje için makul bir tahmin 100 kişi-ay maliyetini oluşturur ancak müşteri 60 kişi-ay karşılayabilirse tahminin 60 kişi-ay şeklinde değiştirmesi istenir. Bu yöntem, teslimatın gecikmesi ve ekibin fazla mesai yapma olasılığını yükselttiği için iyi bir uygulama değildir (Kumari ve Pushkar, 2013).
- ***Yapay Zekâ Yöntemleri İle Yazılım Maliyet Tahmini:*** Son zamanlarda Yapay Zekâ alanında yapılan araştırmalar ve araştırmalar sonucunda elde edilen başarılardan dolayı bu alana bir yönelme olmuştur. Birçok bilim alanı Yapay Zekâ yöntemleri ile tekrar ele alınmaya başlanmış ve kayda değer sonuçlar elde edilmiştir. Bu alanlardan bir tanesi de yazılım maliyeti tahmin yöntemleridir. Yazılım projelerinin maliyet tahmini Yapay Zekâ yöntemleri ile tahmin edilmeye başlanmıştır. YSA, MÖ, BM, GA bu yöntemlerden sadece birkaç tanesidir.

2.1.3.3. Melez Sistemler

Melez sistemler, birden fazla faklı sistemin (YSA, GA, BM vb.) birlikte kullanılması ile oluşturulan sistemlerdir. Yazılım projelerinin maliyet tahmininde melez sistemler, birden fazla algoritmik yöntem, birden fazla algoritmik olmayan yöntem ya da algoritmik yöntemler ile algoritmik olmayan yöntemlerin bir arada kullanılması ile oluşturulmaktadır. Neuro-Fuzzy COCOMO modeli, Huang ve diğ. (2003) tarafından yazılım maliyet tahmini için oluşturulmuş melez bir sistemdir. Araştırmacılar, COCOMO modeli ile BM ve YSA'yı birleştirerek oluşturdukları melez sistemin yazılım maliyet tahmini başta olmak üzere yazılım mühendisliğindeki birçok probleme çözüm sağlayacağını belirtmişlerdir. Başka bir çalışmada (Molani ve diğ., 2014) YSA ve GA birlikte kullanılarak yazılım maliyet tahmini için melez bir model geliştirilmiştir. Araştırmacılar modelin optimalliğini kanıtlamak için modelin tahmin çıktılarını COCOMO 81 modelinin tahmin çıktıları ile karşılaştırmıştır. Geliştirilen modelin COCOMO 81 modelinden daha büyük doğruluk oranıyla tahmin yaptığı gözlemlenmiştir.

2.2. YAPAY ZEKÂ

Yapay Zekâ, zeki bir canlının taklit edilmesi amacı ile belli hesaplamalar sonucunda elde edilen zekâdır. Çoğu zaman akıl ve zekâ kavramları karıştırılır. Akıl somut olarak ölçülemez ve zaman içinde geliştirilebilir. Elmas (2016) aklın genetik olduğundan ve çevreden etkilenerek gelişebileceğinden söz etmekte ayrıca aklın makine, bilgisayar, yazılım gibi farklı yollarla taklit edilemeyeceğini zekânın ise geliştirilebilir, ölçülebilir ve taklit edilebilir bir yapı olduğunu savunmaktadır. TDK'nın verdiği tanıma göre zekâ, insanın düşünme, fikir yürütme, nesnel gerçekleri algılama, kavrama, yargılama, sonuç çıkarma yeteneklerinin tümüdür (TDK, 2020). Nabiyev'e (2016) göre zekâ; bireylerin amaçlı bir biçimde hareket edebilme, mantıklı düşünebilme ve çevresine uyum gösterme yetilerinin tanımıdır. Bazen zekâ; bir olayı önce anlama, ilişkileri kavrama yargıda bulunma daha sonra çözme yeteneği biçiminde de tanımlanmaktadır. Oleron (1996) zekâyı araçların duruma göre uygun kullanılması olarak tanımlar. Elmas (2016) zekânın belirli bir konuda çalışılarak, öğretilerek, edinilen bilgi ve birikimlerle, deneyimlere dayalı becerilerle geliştirilebilir olduğunu ifade etmiştir. Alan W. Turing "Makineler düşünebilir mi?" sorusuyla ilk kez insana has bir özelliğin makinelere nakledilebilme düşüncesini ortaya atmış ve Turing Testi ve Yapay Zekâ gibi kavramları literatüre kazandırmıştır (Kartal Karataş, 2011; Nabiyev, 2016). 1956 yılında Yapay Zekâ terimi ilk defa Dartmouth College'de düzenlenen bir konferansta kullanılmıştır (Nabiyev, 2016).

Çağımızda bilgisayar sistemleri hem eylemler arasında bağları öğrenebilmekte hem de eylemler hakkında kararlar alabilmektedir. Matematiksel olarak formülasyonu kurulamayan ve çözülmesi mümkün olmayan problemler sezgisel yöntemlerle bilgisayar sistemleri tarafından çözülebilmektedir. Bilgisayarlar sistemlerini bu özelliklerle donatan ve bu kabiliyetlerinin gelişmesini sağlayan çalışmalar Yapay Zekâ çalışmaları olarak bilinmektedir (Öztemel, 2016). Nabiyev (2016)'e göre Yapay Zekâ, bir bilgisayarın ya da bilgisayar denetimli bir makinenin, çoğunlukla insana has özellikler olduğu kabul edilen fikir yürütme, mana çıkarma, genelleme ve önceki tecrübelerinden öğrenme gibi yüksek zihinsel süreçlere ilişkin görevleri yerine getirme kabiliyeti olarak tanımlanmaktadır.

Yapay Zekâ, hem bilişsel sistemleri simule etmeyi hem de "akıllı" sistemleri yapılandırmayı amaçlayan bilimsel disiplindir (Görz, 2005). Başka bir tanıma göre Yapay Zekâ, mevcut verileri kullanarak doğru sonuçlar çıkaran, isabetli kararlar verebilen akıllı bilgisayar ve türevi makineleri yapma bilimidir. Bu bilimle uğraşanlar, zekânın doğasını anlamaya çalışarak bilgisayarları daha işlevsel, daha faydalı hale getirmeye çalışmaktadır. Buradaki amaç hep daha zeki programlar oluşturabilmektir (Sönmez, 2020). Yapay Zekâ teknolojisinin çok geniş bir çalışma alanı vardır (Allahverdi, 2002). Bunlardan bazıları aşağıdaki şekildedir (Tuzcuoğlu, 2003):

- İnsanın beyin işlevlerini inceleyip simülasyonunu çıkararak keşfetmek.
- İnsanların bir problem karşında geliştirdiği taktik ve tutumu model almak.
- İnsanın öğrenme yöntemlerini şekilsel duruma getirmek ve bilgi sistemlerine uygulamak.
- İnsanın, bilgisayar etkileşimini ergonomik hale getirerek ona göre kullanıcı dostu donanımlar üretmek
- Uzman kişilerden toplanan verilerden Bilgi Sistemleri ya da Uzman Sistemler geliştirmek.
- Gelecekte bilgi toplumlarının oluşturulmasına yardımcı olacak "Genel Bilgi Sistemleri" oluşturmak.
- Zeki robot timler oluşturmak
- Bilimsel buluşlarda ve araştırmalarda kullanılmak üzere araştırma yardımcıları oluşturmak.
- Askerî alanda çabuk karar verebilen sistemler geliştirmek.
- Tehlike anında insan yerine kullanılabilecek yardımcı eleman oluşturmak.
- Optik algılama bazında nesne ve renk tanımlayacak sistemler geliştirmek.
- Tıp alanında karışık durumlarda karar verebilecek sistemler geliştirmek

Yapay Zekâ teknolojisinde çeşitli yöntemler mevcuttur. Yapay Zekâ yöntemlerinden bazıları:

- Tavlama Benzetimi
- Uzman Sistemler
- Bilgisayarlı Görme
- Konuşma Tanıma
- Robotik
- Kaotik Modelleme
- YSA
- BM
- Melez Sistemler
- GA
- MÖ

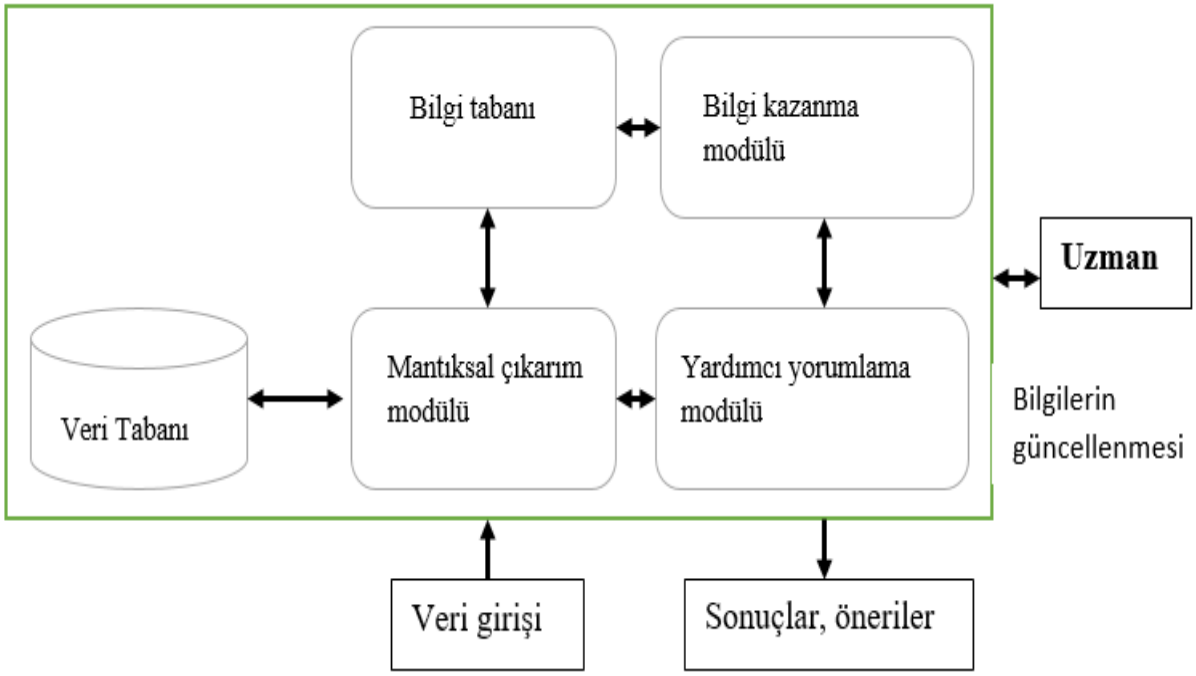
2.2.1. Tavlama Benzetimi

Tavlama Benzetimi optimizasyon problemleri için tasarlanmış ihtimallere dayalı bir algoritmadır. Amaç optimal çözümün en kısa sürede üretimini sağlamaktır. Bu algoritma özellikle hesaplama alanında bir deneyin ya da nümerik sonuçların anlık değerlerini elde etmek için kullanılır. Tavlama Benzetimi ilk olarak metallerin tavlama işlemini simüle etme amacıyla önerilmiştir ve daha sonra yinelemeli bir optimizasyon yöntemi olarak tanıtılmıştır. Tavlama Benzetimi algoritması adını, demircilerin demiri şekillendirmek için demiri döverken belirli bir sıcaklığa kadar ısıl işlemden geçirmesi olayından almıştır. Tavlama sürecinin oluşumu, oldukça yüksek bir sıcaklık değerine sahip bir çözümden başlayıp sıcaklığı dereceli olarak düşürerek iyi ve kötü çözümler arasında gezinip en nihayetinde optimal çözüme ulaşmaktır **(**Ayan, 2009**)**. Tavlama olayındaki gibi çözülmesi gereken problem ele alınır tavlama derecesi ile ısıtma aşamasından geçirilir arzu edilen noktaya gelindiğinde hedefe varıldığı kabul edilir. Tavlama Benzetimi elektronik devre tasarımı, görüntü işleme, kesme ve paketleme, akış ve iş çizelgeleme gibi problemlerin çözümlerinde kullanılmaktadır.

2.2.2. Uzman Sistemler

Uzman Sistemler, belirli konularda uzman kişilerin görüş ve önerileri ile oluşturulmuş veri tabanlarını kullanarak karar verme işlemlerini modelleyebilen, kendini geliştirebilen yazılım sistemleridir. Uzman Sistemler, tavsiyelerde bulunabilir, sorunları çözümleyebilir, bağlantı kurabilir, tasarım yapabilir, tanım yapabilir, yorumlayabilir, kestirim yapabilir, yargılayabilir, denetleyebilir, öğrenebilir ve öğretebilir, yazılımlardır (Civalek, 2003).

Uzman Sistem yaratma işlemleri "Bilgi Mühendisliği" olarak adlandırılmakta ve "Uygulamalı Yapay Zekâ" olarak kabul edilmektedir. Uzman Sistemler aynı zamanda daha geniş bir grubu oluşturan "zeki sistemler" ve "bilgiye dayalı sistemler" in alt grubunu oluşturmaktadır (Kurbanoğlu, 1992). Şekil 2.2'de bir Uzman Sistemin genel yapısı verilmiştir.



Şekil 2.2: Bir Uzman Sistemin genel yapısı.

Uzman Sistemlerin avantajları şu şekilde ifade edilebilir:

- ***Maliyet Azalması:*** Uzman Sistemler daha kısa sürede daha çok iş yapıp maliyeti düşürürler.
- ***Verimlilik:*** Uzman Sistemler daha kısa sürede daha çok çalışır ve daha az yorulurlar, verimliliği artırırlar.
- ***Kalite Artışı:*** Uzman Sistemler tutarlı ve uygun hareket ederek hata oranını düşürür, kaliteyi arttırırlar.
- ***Tutarlılık:*** Hava durumu, ekonomi, duygusal ilişkiler gibi insani durumlar Uzman Sistemleri etkilemez. Dolayısıyla aynı şartlarda aynı sonucu üretirler. Bu da tutarlılığı sağlar.
- ***Esneklik:*** Uzman Sistemlerin veri tabanları istenildiği zaman güncellenebilir bu da esnekliği sağlar.
- ***Kapsamlılık:*** Uzman Sistemler çoğunlukla birden fazla uzman bilgisiyle oluşturulmuştur dolayısıyla kapsayıcı sonuçlara sahip olurlar.
- ***Karar Alma Süresinin Kısalması:*** Uzman Sistemler uzmanlara ve diğer birçok yönteme oranla daha kısa sürede karar alırlar.
- ***Güvenilirlik:*** Uzman Sistemler veri tabanındaki bilgilerden yola çıkarak sonuçlar üretirler. Bu sonucu üretirken hiç sıkılmadan ve yorulmadan işlemi gerçekleştirirler. Bu da güvenilirliği sağlar.
- ***Tehlikeli Ortamlarda İşlem:*** Uzay keşifleri, savaş ortamları, okyanusun derinlikleri, zehirli gazlar bulunan yerler, madenler gibi insanlar ve canlılar için risk oluşturan birçok ortamda Uzman Sistemler rahatlıkla çalışabilirler.
- ***Eksiksiz ve Mutlak Olmayan Bilgi ile Çalışma:*** Uzman Sistemler sıradan bilgisayarların aksine insanlar gibi net olmayan bilgilerden çıkarımlar yapabilirler.
- ***Eğitim:*** Uzman Sistemler eğitilebilir ve eğitebilirler. Bir Uzman Sistem başka bir Uzman Sistemi ya da bir insanı eldeki verileri kullanarak eğitebilir.
- ***Problem Çözme Kabiliyeti:*** Uzman Sistemler çok karmaşık görünen problemleri insana oranla daha kısa sürede çözebilirler.

Günümüzde birçok alanda kullanılan Uzman Sistemler insanın iş yükünü azaltarak hayatı kolaylaştırmaktadır. Uzman Sistemlerin kullanıldığı bazı alanlar ve bu alanlardaki bazı örnekler şu şekilde ifade edilebilir:

- ***Yorumlama:*** Ses tanıma, görüntü analizi, denetim
- ***Tahmin:*** Hava tahmini
- ***Teşhis:*** Tıp, elektronik
- ***Tasarım:*** Devre çizimi
- ***Planlama:*** Askerî planlama
- ***Görüntüleme:*** Hastalıkların teşhisi ve tedavisi
- ***Eğitim:*** Danışma, ıslah, tedavi
- ***Tamir:*** Otomobil, bilgisayar
- ***Planlama:*** Askeri planlama, otomatik programlama
- ***Kontrol Sistemleri:*** Hava trafik kontrolü
- ***Hata Ayıklama:*** Yazılımlar

2.2.3. Bilgisayarlı Görme

Bilgisayarlı görme, 1960'ların sonlarında Yapay Zekâ çalışmalarına öncülük eden üniversitelerde başlamıştır (Szeliski, 2010). Robotlara akıllı davranışlar kazandırmayı amaçlayan araştırmacılar insan gibi görmeyi de bu çalışmanın bir basamağı olarak incelemiştir. Bilgisayarlı görme, insanın görme yeteneğinin bilgisayarlara aktarılması çalışmasıdır. Başka bir ifadeyle bilgisayarların, dijital görüntülerden veya video görüntülerinden, insan gibi sonuçlar çıkararak işlemler gerçekleştirip elde ettiği sonuca göre karar verebilir aşamaya gelebilmesidir. Bilgisayarların bunu yapabilmesi için, dijital görüntüyü oluşturma, işleme, analiz etme ve anlamlı hale getirme işlemlerini gerçekleştirebilecek yöntemleri kullanması gerekmektedir (Autonom, 2019). Bilgisayarlı görmenin, görüntü tanıma, hareket algılama, görüntü onarma, indeksleme, hareket izleme gibi alt dalları bulunmaktadır. Bilgisayarlı görme çok çeşitli alanlarda kullanılmaktadır. Bu alanlardan bazıları; medikal uygulamalar, sanayi uygulamaları, askeri uygulamalar ve otonom araç uygulamalarıdır.

2.2.4. Konuşma Tanıma

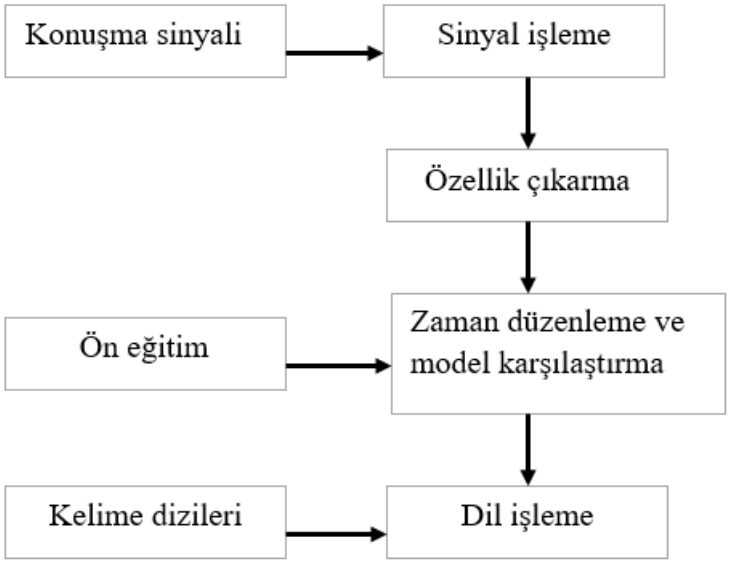
Konuşma tanıma, insan sesinin bilgisayarlar tarafından algılanmasıdır (Yalçın, 2008). Bilgisayar sistemleri, ses sinyallerinden oluşan konuşma verisini alarak veriyi işler, işlenmiş veriden yolara çıkarak tahminde bulunur ve bir metin çıktısı oluşturur buna konuşma tanıma

denir. Dede (2008) çalışmasında YSA'yı kullanarak bir konuşma tanıma uygulaması geliştirmiştir. Çalışmasında konuşma tanıma probleminin esas itibariyle bir örüntü tanıma problemi olduğunu belirtmiştir. Bir sözcüğün zamana göre frekans değerlerine yayılmış olan gösterimi o kelimenin örüntüsü olarak hesaplanır. Çünkü araştırmacıya göre ses sinyallerinde, belli sözcükler başka seslendirme kayıtlarında benzer sinyal şekilleri ortaya koymaktadır. Araştırmacı kelimelerin örüntülerinden yola çıkarak konuşma tanıma uygulaması geliştirmiştir. Öcal (2005) çalışmasında, konuşma alanında yapılan en eski çalışmanın 1936 yılında yapıldığını belirtmiştir. En başta oluşturulan konuşma tanıma sistemleri yalnızca sayıları anlamlandırabilmekteydi. Daha sonraki yıllarda bu alan üzerindeki çalışmalar artmıştır. Sovyetler Birliği, ABD, Japonya ve İngiltere'deki laboratuvarlarda 1950 ve 1960 yılları arasında ünlü ve ünsüz harfleri tanıyan donanım tabanlı konuşma tanıma sistemleri geliştirilmiştir (Yakar, 2016). Dinamik programlama ile başlayan çalışmalar saklı Markov modellerinin kullanılması ile günümüze kadar sürdürülmektedir (Gürel ve Aslan, 2008).

Konuşma Tanıma Modelini oluşturan kısımlar aşağıda belirtilmiştir (Yalçın, 2008).

- Sinyal işleme modülü
- Özellik çıkarma modülü
- Zaman düzenleme ve model karşılaştırma
- Bir final kelime dizisi seçmek için dil modeli

Şekil 2.3'te konuşma tanımanın genel bir modeli verilmiştir.



Şekil 2.3: Konuşma Tanıma Modeli.

Konuşma tanıma uygulamaları günlük yaşantının birçok alanında kullanılmaktadır. Bankacılık işlemleri, akıllı ev eşyaları, sesli web tarama sistemleri, otomobiller, kamu tesisleri bunlardan sadece birkaçıdır. Bunlardan bir diğeri havalimanlarında kurulan farklı dil desteği ile rehberlik yapan Konuşma Tanıma sistemleridir. Bu sistemler farklı dilleri konuşan ama aynı cihazı kullanan yabancı gruplar için geliştirilmiştir.

Otonom araçlar, neredeyse bir insan gibi iletişim kurabilmektedir. Tek bir cümle ile birden fazla soruyu anlamaya yeteneğine sahip bu sistemler yolcular ile sesli iletişim kurabilir, navigasyon komutları verebilir, yol bilgisi alabilir, internette arama yapabilir, çalan müziği değiştirebilir.

Günlük hayatı yakından ilgilendiren giyilebilir teknoloji, müşteri hizmetleri, görme engelliler için özel cihazlar ve turizm sektörü konuşma anlama teknolojisi ile büyük bir gelişim göstermesi beklenen alanlar arasındadır (Techinside, 2017).

2.2.5. Robotik

İkinci Dünya Savaşı'ndan sonra yüksek seviyeli algoritmaların geliştirilerek akıllı oyun programlarının yapılması, otomat oyuncaklara ilgiyi azaltmıştır. Otomatların yerine, dışarıdan algıladıkları verileri alan ve aldıkları verileri kullanıp gerekli talimatları yerine getiren

sibernetik makineler geçmiştir. Bu makineler, yakınındaki ısı, ışık, gürültü kaynağını ya da bir engeli algılamakta ve ona göre hareketlerine yön verebilmektedir. Nobert Wiener, 1948'de "yönetim" anlamına gelen Sibernetik'i, insanlar ve makineler arasında iletişim ve düzenleme bilimi olarak tanımlamıştır (Nabiyev, 2016).

Robot kelimesi ilk kez 20. yüzyılın başlarında bir tiyatro oyununda kullanılmıştır. Bilim kurgu yazarı Isaac Asimov insanlık geleceğini ilgilendiren 3 önemli robotik yasası olduğunu iddia etmiştir:

- ***1. yasa:*** Robotlar hiçbir şekilde insanlara zarar vermemeli ve insanın zarar göreceği hiç bir durumda pasif kalmamalıdır.
- ***2. yasa:*** Robotlar 1. yasaya uymak kaydıyla insanlar tarafından verilen bütün komutları yerine getirmek zorundadır.
- ***3. yasa:*** 1. ve 2. yasaya uymak kaydıyla robotlar kendilerini korumak zorundadır.

Şerit üzerinde hareket eden kusurlu ürünleri tespit etme, araba parçalarını birleştirme, malzeme taşıma ya da daha kompleks davranışları yapan robotları çağımızda görmek mümkündür. Bilgisayarlar, yazılımlar aracılığı ile robotları kontrol eder ve robotlara işin nasıl yapılacağını öğretir. Bu yazılımlar robota hareketinin zamanını, yönünü, mesafesini ve benzer konularda ne yapması gerektiğini komutlar yardımıyla bildiren yazılımlardır. Bazı robotlar bir kere programlandıktan sonra tekrar tekrar programlanmalarına gerek yoktur. Rutin işlerde kullanılan, bir kere programlandıktan sonra fazla kontrol edilmeyen bu tür robotlara seç, al, yerleştir robotları denilmektedir (Civalek, 2003). Yeni nesil robotlar ise her geçen gün daha fazla zekâ yeteneği ile donatılmaktadır. Bu sayede çevrelerini daha iyi algılamakta ve hareketlerini planlamaya yönelik gelişmektedir (Kocabaş, 2013).

Robotik, makine tasarımı, kontrol kuramı, bilgisayarlı programlama ve elektronik disiplinlerinin karışımından oluşan ve Mekatronik olarak isimlendirilen mühendislik alanına girmektedir. Mekatronik biliminin temelleri 1969 yılında Japonya'da atılmıştır. Robotik, Yapay Zekânın Mekatronikle sınırında olan bir alandır (Civalek, 2003; Ozan, 2020). Robotlar üç kategoride incelenmektedir:

- **Birinci nesil robotlar:** Kendi kendine ayarlanamaz robotlardır.
- **İkinci nesil robotlar:** Dışarıdan gelen verileri alabilen en az bir alıcısı olan robotlardır.
- **Üçüncü nesil robotlar:** Kendi kendini kontrol edebilen bir Yapay Zekâ ile donatılmış robotlardır.

Robotların üstün özellikleri arasında üretim artışını sağlaması, üretim maliyetini düşürmesi, kalite artışını sağlaması, kötü şartlarda çalışabilmesi, yönetilmesinin ve kontrol edilmesinin kolay olması, çalışma sahasının geniş olması, yaşam süresinin uzun olması, daha dayanıklı ve uyumsal olması onun kullanımını cazip kılmıştır. Bu nedenle robotlar, Japonya ve ABD'de birçok araştırma ve geliştirme alanında kullanılmaktadır. Teknolojinin birçok alanında olduğu gibi robotların da gelişme göstermesinin en büyük etkenlerinden birisi askerî alanlarda kullanılmasıdır (Nabiyev, 2016).

Robotik bilimi, tıp ve sağlık alanında, endüstride, uzay araştırmalarında, askeri alanda, eğlence alanında, ulaşımda, tarım ve hayvancılık alanında, sibernetik alanı gibi daha birçok alanda kullanılmaktadır. Geçmişten günümüze farklı alanlarda robotlar tasarlanmıştır (Yamanol, 2016; Ozan, 2020).

2.2.6. Kaotik Modelleme

Fransızca'dan Türkçe'ye geçen kaos kelimesi, evrenin düzene geçmeden önceki uyumsuz, karışık hali ya da karmaşıklık, karmaşa anlamlarını içermektedir. Kaos, karmaşık ve düzensiz görünümlü başlangıç koşullarına bağlı, deterministik olmayan, zamanla değişen sistemler için kullanılan bir olgudur (Ablameyko, 2003). Kaos terimi ilk olarak 1900 yıllarında bilim adamı Jules Henri Poincaré tarafından karar verilemez ve saptanamaz olaylar için kullanılmıştır (Poincaré, 1912).

Kaos kuramı, dinamik sistemlerin beklenmedik garip davranışlarını araştıran bir bilim dalıdır (Merih, 2016). Kaotik bir sistemde kaotik işaretler elde edildikten sonra verilere uygun bir matematiksel model ile kaotik işaretler ifade edilebilmektedir. Bu direk olarak lineer olmayan bir denklem formatında olabileceği gibi BM, YSA veya Volterra Serileri gibi farklı modelleme

teknikleri ile modellenebilmektedir. Buna Kaotik Modelleme denir. Dinamik bir sistemin kaotik olarak sınıflandırılması için başlangıç koşullarına duyarlı olmalıdır, topolojik olarak karıştırılmalı ve periyodik yörüngeleri yoğun olmalıdır. Son yıllarda kaos teorisi borsa, meteoroloji, iletişim tıp, kimya, mekanik gibi çok farklı dallarda kullanılmaktadır.

2.2.7. Yapay Sinir Ağları

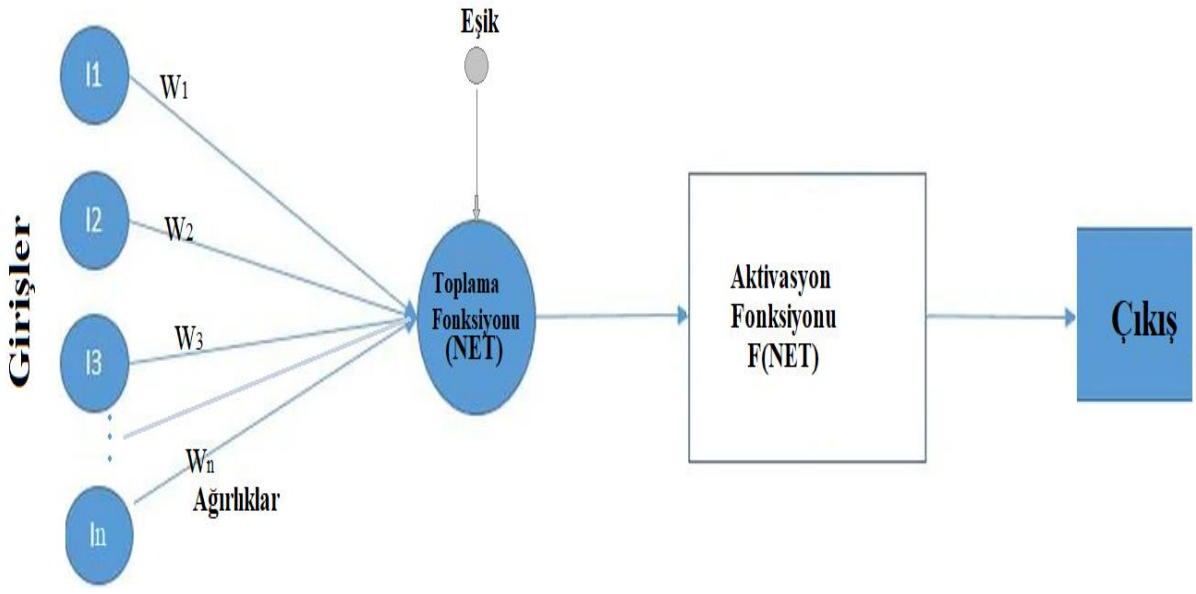
İnsan sinir hücresinin model alınarak oluşturulmuş bilgisayar sistemlerine, öğrenmeyi, öğretmeyi, sonuç çıkarmayı öğreten Yapay Zekâ yöntemine YSA denir. YSA tasarlanırken insan sinir sisteminden ilham alınmıştır. Bir YSA'yı oluşturan işlem yapıları birbirlerine ağırlıklı bağlantılar ile bağlanmış her birinin kendi belleği olan, dağıtık ve paralel yapılardır. (Elmas, 2016). Başka bir tanıma göre YSA insan beyninin biyolojik sinir sistemini temel alarak oluşturulan bu bilgisayar programları ve yapıları, algılayıcılardan aldığı veri girişleri ile daha önceden öğrenerek sınıflandırmış olduğu bilgileri kullanarak yeni bilgiler üretmektedir. Ürettiği ve oluşturduğu bilgilerden yola çıkarak yeni kararlar verebilir duruma gelmektedir (Keskenler ve Keskenler, 2017).

YSA ile ilgili ilk çalışmalar 1940'larda bugün birçok ağın önemli yapı taşı olan McCulloch-Pitts-Neuron olarak bilinen basit bir sinir hücresi modeli tasarlanması şeklindedir (Sezen, 2008; Mijwill, 2017). 1949 yılında, günümüzde bile hala pek çok öğrenme kuralının özünü oluşturan Hebbian Öğrenme oluşturulmuştur. 1957'de farklı harfleri okuyup tanıyan bir YSA modeli geliştirilmiştir. 1959 senesinde MADALINE ve ADALINE olarak adlandırılan YSA modeli oluşturulmuştur. 1960'lı yılların sonlarında YSA'nın doğrusal olmayan problemleri çözememesi ve XOR (Exclusive-Or) problemi ile bunun ispatlanması bu alandaki çalışmaları durma noktasına getirmiştir. Geleneksel Gezgin Satıcı Problemi'nin (Travelling Salesman Problem – TSP) YSA ile çözülmesi aynı zamanlarda tek katmanlı algılayıcıların çözemediği XOR probleminin çok katmanlı algılayıcıların bulunması ile çözülmesi bu alanı tekrar ilgi odağı haline getirmiştir. YSA'daki gelişmelere donanım teknolojisindeki yeniliklerin etkisi çok olmuştur. Bilgisayarların işlem hızları ve bellek kapasiteleri artmış bu da işlemlerin daha kısa sürede gerçekleştirilmesini sağlamıştır. Bu sayede YSA'nın kullanımı kolaylaşmıştır. YSA 1990'lı yıllardan sonra sadece laboratuvarlarda uğraşılan teorik çalışmalar olmanın ötesinde günlük hayatta kullanılan sistemler olmaya başlamıştır bu sayede insanların

hayatlarına pratiklik kazandıran yeni bir sürü uygulama geliştirilmiştir. YSA'daki bu gelişim günümüzde hala devam etmektedir (Öztemel, 2016).

Bir biyolojik beyin sinir hücresi akson, hücre gövdesi, dendrit ve sinapslardan oluşmaktadır. YSA, biyolojik sinir sisteminin model alınarak oluşturulmuş halidir. Sinir sistemi vücudumuzda bulunan milyarlarca sinir hücresi ve bunların bağlantılarından oluşmaktadır. Sinir hücrelerinin oluşturduğu bu bağlantılar sinir ağını meydana getirmektedir. İnsan vücudunda merkezi sinir sistemi ve çevresel sinir sistemi olarak 2 tane sinir sistemi mevcuttur. Merkezi sinir sistemini oluşturan beynimizde 10^{11} adet sinir hücresi ve bunların da $6x10^{13}$'ten fazla sayıda bağlantısının olduğu ifade edilmektedir (Öztemel, 2016).

YSA giriş seti olarak kendisine verilen bilgilere karşılık bazı matematiksel fonksiyonlar kullanarak net denilen çıkışı üretir. Bunu yapabilmesi için ağ mevcut örneklerle önceden eğitilmektedir. YSA'nın en temel elemanına yapay sinir hücresi (proses, perceptron) denmektedir. Bu yapay sinir hücresi altı temel elemandan oluşmaktadır. Bunlar, girişler, ağırlıklar, eşik, toplam fonksiyonu, aktivasyon fonksiyonu ve çıkış değeridir. Şekil 2.4'te yapay sinir hücresi verilmiştir.



Şekil 2.4: Yapay sinir hücresi.

Girişler: Girişler ($I_1, I_2,\ldots, I_n$) bir sinir hücresine dışarıdan, başka sinir hücresinden ya da kendisinden gelen bilgilerdir.

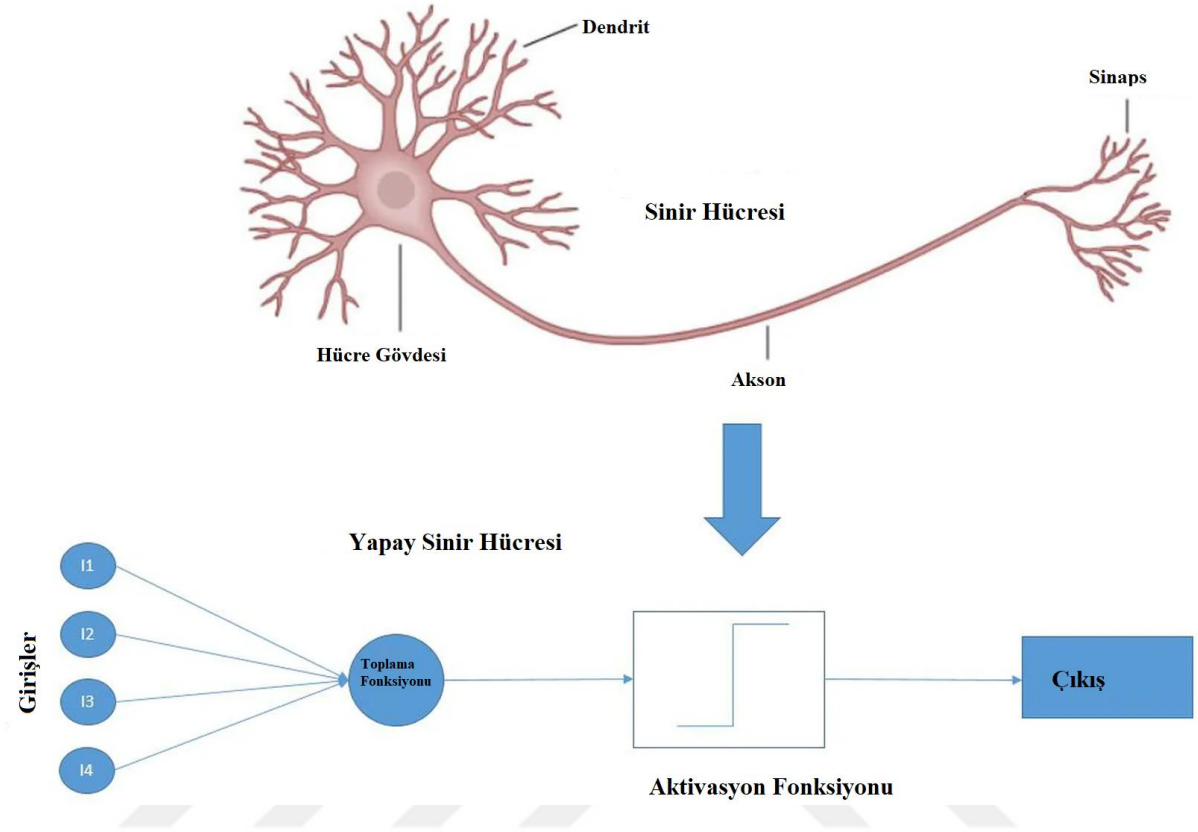
Ağırlıklar: *Ağırlıklar (W_1, W_2,..,W_n)* bir yapay sinir hücresi tarafından alınan bilginin önemini ve hücre üzerindeki etkisini gösteren katsayıdır. Ağırlıklar pozitif ya da negatif değerler alabilir. Şekildeki ağırlık W_1, girdi I_1'in yapay sinir hücresi üzerindeki etkisini göstermektedir. Ağırlık değerinin büyük olması o girdinin ağa güçlü bağlandığını, küçük olması ağa zayıf bağlandığını ve sıfır olması o ağ için herhangi bir etkisinin olmadığını göstermektedir.

Eşik: Sinir hücresinin ya da ağın çıktısının sıfır olmasını engellemek için kullanılır.

Toplama Fonksiyonu: Yapay sinir hücresine giren net bilgiyi değişik fonksiyonları kullanarak hesaplayan bir fonksiyondur. En yaygın kullanılan fonksiyon ağırlıklı toplama fonksiyonudur. Bu toplama fonksiyonu her gelen girdi değerinin kendi ağırlığıyla çarpılarak toplanmasıdır.

Aktivasyon Fonksiyonu: Bu fonksiyon, hücreye gelen net girdiye karşılık bir çıkış hesaplar. Çıkışı belirlemek için değişik fonksiyonlar kullanılır. Bu fonksiyonlardan bazıları; Doğrusal Fonksiyon, Step Fonksiyon, Sinüs Fonksiyonu, Eşik Değer Fonksiyonu, Hiperbolik Tanjant Fonksiyonu'dur.

Çıkış: Aktivasyon fonksiyonunda net girdinin işlenmesi sonucunda elde edilen çıkış değeridir. Elde edilen çıkış başka bir hücreye ya da aynı hücreye giriş olarak verilebilir. Bir hücrenin birden fazla girişi olabiliyorken sadece tek bir çıkışı vardır. Şekil 2.5'te insan sinir sistemine ait gerçek sinir hücresi ve YSA'ya ait sinir hücresi şekli bir arada verilmiştir.



Şekil 2.5: Biyolojik sinir hücresi ve yapay sinir hücresi.

YSA yapısının avantajları aşağıdaki gibi ifade edilebilir:

- Geleneksel yöntemlerle çözülemeyen birçok problemi çözebilir.
- Tam ve normal olmayan, belirsiz ve eksik bilgileri işleyebilen çok güçlü problem çözme yeteneğine sahiptir.
- Doğrusal olmayan ilişkileri de kolaylıkla modelleyebilir.
- Örnekleri kullanarak öğrenebilir ve görülmemiş örnekler hakkında bilgi üretebilir.
- Bilgiyi ağın bağlantılarında saklamaktadır. Bilgiler öteki yazılımlar gibi veri tabanında veya programın içerisinde değildir.
- Eksik bilgi ile çalışabilmektedir.
- Hata toleransı ağın bir bölümü yanlış oluşturulduğunda veya ağ zarar gördüğünde bile düzgün çalışmaya devam etmesini sağlamaktadır.

YSA yapısının dezavantajları aşağıdaki gibi ifade edilebilir:

- Ağ topolojinin belirlenmesi kesin kurallara dayanmadığından ağ deneme yanılma yoluyla belirlenmektedir.
- Ağın parametre sayısının oluşturulmasında kesin kurallar yoktur.
- Örneklerin tasarımında ve belirlenmesinde bir kural yoktur ve sadece numerik bilgiler ile çalışmaktadır.
- Donanıma bağımlı çalışmaktadır.
- Ağın eğitim sürecinin ne kadar süreceği ve ne zaman bitirileceğine karar verecek bir yöntem yoktur. Bu durumda eğitim süreci uzun sürebilmektedir.
- Ağ davranışı açıklanamaz.

2.2.8. Bulanık Mantık

Dünya sadece siyah ve beyaz renklerinden oluşmamaktadır. Siyah renginden beyaz renge geçişte çok fazla ara renk tonu mevcuttur. Benzer şekilde bilgisayar biliminde de her şey sadece 1 ve 0 ya da var ve yok değildir. 1 ve 0 arasında ara değerler mevcuttur. Bilgisayar biliminde 1 ve 0 arasındaki ara değerleri, komşuluk derecelik kavramlarına bağlı olarak değerler oluşturulmasını sağlayan bilime BM denir. BM kuramı bilgisayar ve türevi makinelere insanlara has özel verilerini işleyebilme ve onların tecrübelerinden ve öngörülerinden yararlanarak çalışabilme kabiliyeti kazandırır. Bu kabiliyeti verirken dijital ifadeler yerine simgesel ifadeler kullanır. İşte bu simgesel ifadelerin makinelere aktarılması matematiksel bir temele dayanır. Bu matematiksel temel BM Kümeler Kuramı ve buna dayanan BM'dir (Elmas, 2016).

Günlük hayatta kullandığımız birçok kavram bulanıklık içerir. Örneğin az, çok az, biraz, fazla, çok fazla, güneşli, bulutlu, ılık, soğuk, sıcak, kısa, uzun, genç, yaşlı gibi daha birçok dilsel terimler vardır. Bu terimler bulanık değişkenler olarak isimlendirilmektedir. BM kesin olmayan bilgilerin var olduğu durumlarda kullanılmaktadır. Mevcut Sistemin kompleks olduğu ve bilinen klasik yöntemlerle çözümün elde edilemediği, bilgilerin belirsiz olduğu veya bilgilerin kesin olmadığı mevcut durumlarda daha çok tercih edilmektedir.

Dünya Sağlık Örgütü'nün (World Health Organization – WHO) açıkladığı yaş dilimlerine göre 0-17 arasındaki yaşlar ergen, 18-65 arasındaki yaşlar genç, 66-79 arasındaki yaşlar orta yaşlı ve 80-99 arasındaki yaşlar yaşlı sayılmaktadır. Bu durumda 18 yaşındaki biri ve 65 yaşındaki biri genç sayılıyorken 66 yaşındaki biri orta yaşlı sayılmaktadır. Ancak 18 yaşındaki biri ile 50 yaşındaki birinin gençlik oranları aynı olmamasına rağmen klasik yöntemlerde ikisi de aynı kategoriye alınacaktır. Bu gibi durumlarda BM tanımların kullanılması çok daha uygun olmaktadır. Genç, çok genç ya da yaşlı, çok yaşlı gibi bulanık tanımlar durumun daha iyi anlaşılmasını sağlamaktadır. Gerçek dünya hayatı bu gibi pek çok örneği içermektedir. Bunlar gibi özellikleri doğru belirlenemeyen, tam tespit edilemeyen, apaçık görünmeyen, kesin olmayan şeklinde tanımlanan bulanıklık, dereceli üyelik kavramı yardımı ile teknik bilim dünyasına taşınmıştır. Bu kavram 1965 senesinde ilk defa kullanılmıştır (Zadeh, 1965). Zadeh tarafından BM ilkeleri Şekil 2.6'daki gibi belirlenmiştir (Elmas, 2007);

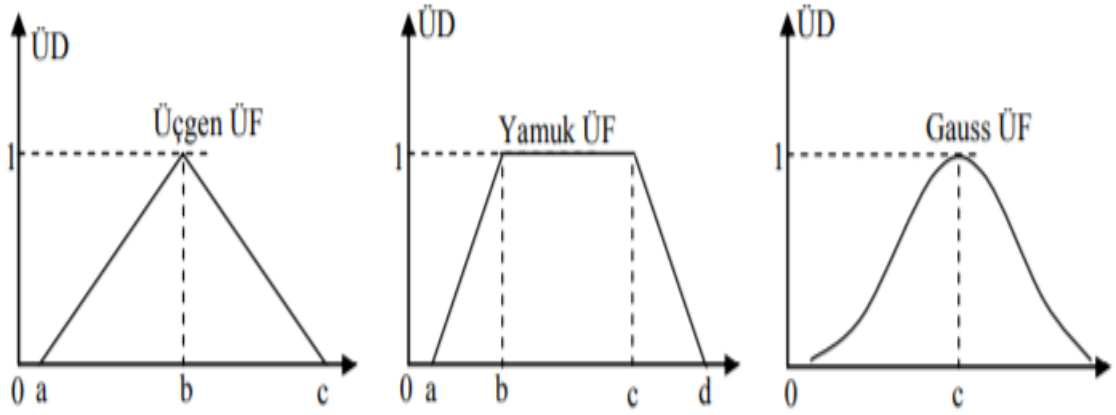
- Bulanık Mantık da kesin belli olan düşünme yerine yaklaşık düşünme kullanılır.
- Bulanık Mantık için bilgi az, çok, küçük, büyük şeklinde dilsel ifadeler ile tanımlanır.
- Bulanık Mantıkta her şey [0-1] aralığında bir üyelik derecesi ile gösterilir.
- Her mantıksal sistem bulanık halde ifadeye dönüştürülebilir.
- Matematiksel modeli çok karmaşık ve zor olan sistemler için Bulanık Mantık uygun bir yöntemdir.
- Bulanık çıkarım işlemi dilsel ifadeler arasında tanımlanan kurallar ile yapılır.

Şekil 2.6: Bulanık Mantık genel ilkeleri.

Lotfi Zadeh 1965 tarihli makalesinde, belirsizlik içeren sistemlerin yeniden gözden geçirilmesi gerektiği fikrini ortaya attıktan sonra bulanık küme kavramı 1970'li yıllarda kullanılmaya başlanmıştır (Altaş, 1999). BM kavramları aşağıda açıklanmıştır.

Üyelik Fonksiyonları: Klasik küme tanımında evrensel kümedeki bir eleman kümeye ait ise 1, değil ise 0 değerini alır. Bu değer kümeye üye olmayı ya da olmamayı ifade etmektedir. BM'de dereceli üyelik söz konusudur. Bir eleman A kümesine üye iken aynı zamanda B kümesinin de üyesi olabilmektedir. Buna dereceli üyelik denir. Elemanın küme içerisindeki

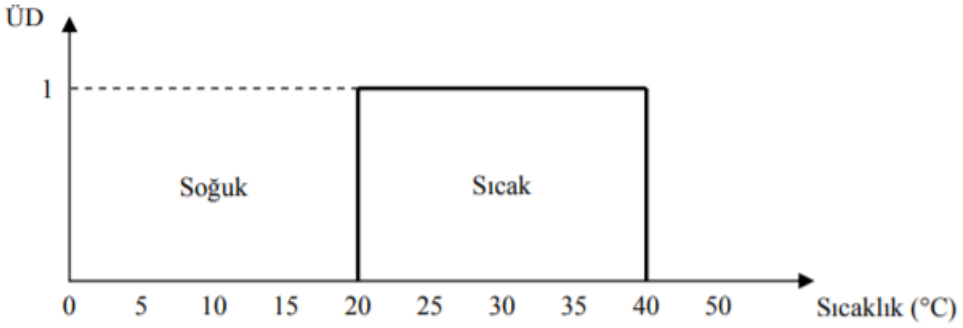
üyelik derecesini veren fonksiyonlara üyelik fonksiyonları denir. Üyelik fonksiyonlarını oluşturmak için birçok yöntem mevcuttur. Bunlardan en gelişmiş olanları aşağıda Şekil 2.7'de belirtildiği üzere üçgen, yamuk ve parabolik fonksiyonlarıdır.



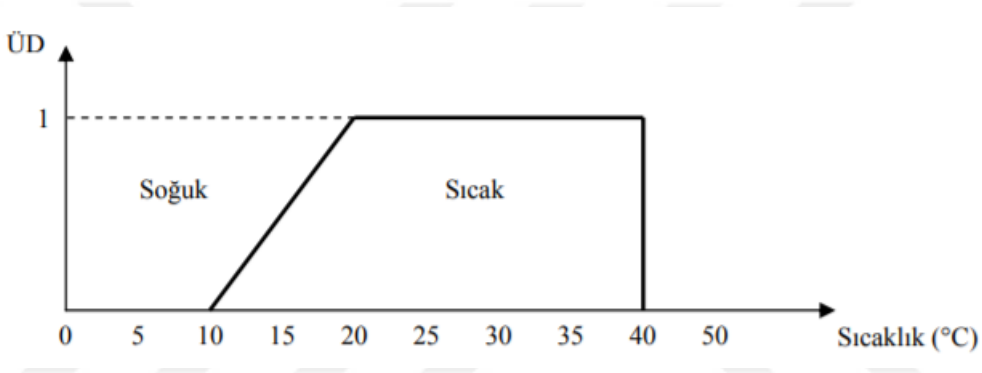
a) Üçgen Fonksiyonu b) Yamuk Fonksiyonu c) Parabolik Fonksiyon

Şekil 2.7: Üyelik fonksiyonları.

Bulanık Kümeler: Klasik kümelerde bir değer o kümenin ya elemanıdır ya da değildir. Hiçbir durumda kısmı üyelik söz konusu değildir. BM'de kısmî üyelikten bahsedilebilir. Aşağıdaki Şekil 2.8)a) incelendiğinde eğer sıcaklık 20^0C'nin altında ise sıcak değildir. Klasik mantığa göre $19,5^0$C soğuk iken 20^0C sıcaktır. Oysaki günlük hayatta suyun sıcaklığını ifade ederken çok soğuk, soğuk, sıcak ve çok sıcak gibi dereceli ifadeler kullanılmaktadır. Bulanık kümelerde üyelik dereceleri [0,1] aralığında sonsuz sayıda değişebilmektedir. Klasik kümedeki soğuk-sıcak ifadesi, BM'de az soğuk, az sıcak gibi esnek ifadelerle gerçek dünyaya benzetilmektedir. Şekil 2.8)b)'deki bulanık küme incelendiğinde tam üyelik 20^0C'de başlamaktadır ve üyelik derecesi 1'dir. 20^0C'den 40^0C'ye kadar üyelik derecesi 1'dir. 20^0C ile 10^0C arasında üyelik derecesi 0 ile 1 arasındadır ve 10^0C'de üyelik derecesi 0 olmaktadır. Yani 20^0C sıcak ise 19^0C biraz sıcaktır.



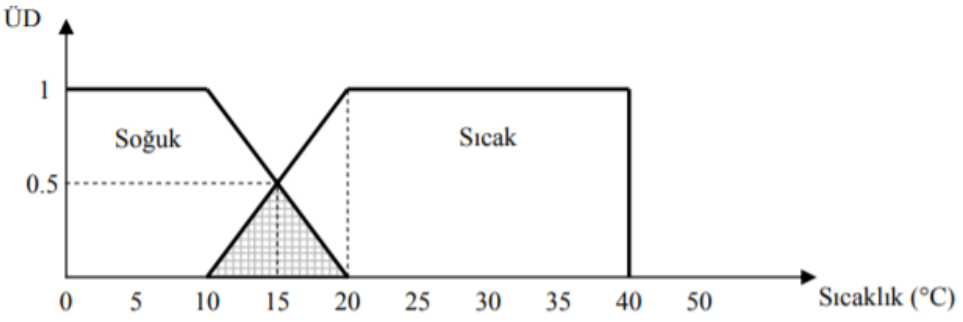
a) Sıcaklık için klasik küme örneği.



b) Sıcaklık için bulanık küme örneği.

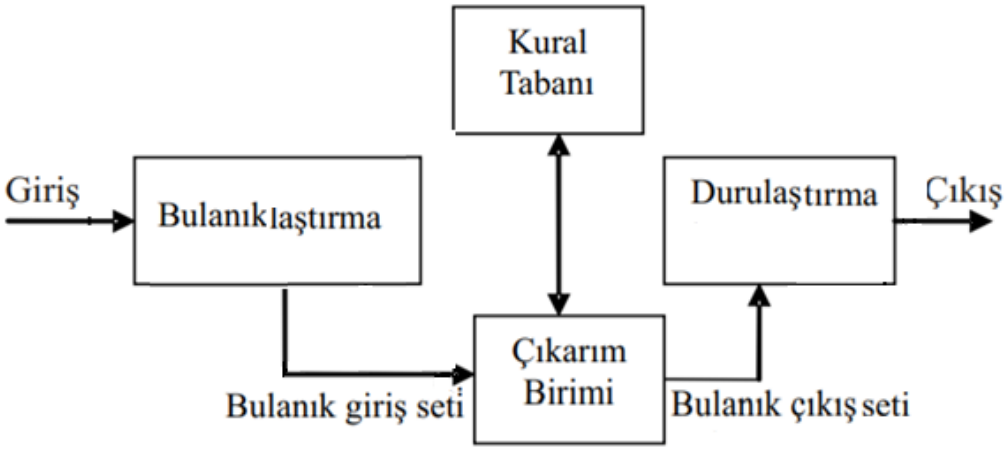
Şekil 2.8: Sıcaklık için küme örnekleri.

Şekil 2.8)a)'da üyelik dereceleri incelendiğinde 15^0C'de 0,5 noktasının hem sıcak hem de soğuk bulanık kümesine üyeliği mevcuttur. 10^0C ile 20^0C arasındaki değerler hem sıcak bulanık kümesine hem de soğuk bulanık kümesine üyedir. Şekil 2.9'da kümelerin örtüşümü olarak isimlendirilen taralı bölge bulanık kümelerin kesişim bölgesidir (Elmas, 2016).



Şekil 2.9: Bulanık kümelerde kesişim.

BM yaklaşımının kontrol sistemlerine uygulandığı ilk çalışma, 1974 senesinde bir buhar makinesinin bulanık kontrolünün yapılması ile gerçekleştirilmiştir. 1980'li yıllarda Japonlar ürünlerinde BM kullanmaya başlamıştır. Uygulandığı alanlarda performansının oldukça yüksek olması BM olan ilgiyi artırmıştır. Bu gelişmeler ışığında, 1989 yılında Değişim Bulanık Mühendislik Laboratuvarı (Laboratory for Interchange Fuzzy Engineering – LIFE) isimli laboratuvarlar kurulmuştur. Bu laboratuvarların kurulumunda Hitachi, Toshiba, Omron ve IBM gibi dünya devlerinin de aralarında bulunduğu 51 firma yer almıştır (Ertunç, 2012). BM uygulanan ürünler Japonya'da 1990 yılında tüketicilere sunulmuştur. BM, klasik yöntemlerle çözülmesi zor olan karmaşık sistemlere getirdiği kolay ve kullanışlı çözümler sayesinde çok geniş bir uygulama alanına yayılmıştır. Kullanım alanı geniş olan BM, Yapay Zekâ uygulamalarında, Robotik çalışmalarında, sağlık, mühendislik, sosyolojik ve psikolojik uygulamaların geliştirilmesinde, kavşak ve ulaştırma sorunlarının çözümünde ve bunlara benzer birçok uygulamada verimli bir şekilde uygulanmaktadır. Altaş (1999) yaptığı çalışmada Bulanık Mantığın uygulama alanlarını; En iyileme problemleri, Görüntü Tanıma, Otomatik Kontrol ve Bilgi Sistemleri olarak dört kategoride incelemiştir. Uygulama alanlarından bazıları; birçok elektronik ev eşyalarında (çamaşır makineleri, elektrik süpürgeleri, klimalar, buzdolapları) taşıma araçlarında (taşıt süspansiyonlarının kontrolü, metrolar, asansörler) inşaat sektöründe kullanılan makinelerin kontrolünde, bilgisayar donanım parçalarının kontrolünde, sembolleri, objeleri ve el yazısını tanımada, trafik lambalarında, kameraların görüntü sabitleme ayarlarında ve buna benzer birçok alanda BM kullanılmaktadır (Ertunç, 2012). BM üç aşamadan oluşur: bulanıklaştırma, bulanık çıkarım ve durulaştırma. En çok kullanılan BM sistemlerinden birisi, Mamdani çıkarım sistemidir. Mamdani yöntemi, Ebrahim Mamdani tarafından önerilmiş olup, çıkarım sistemi, Lotfi Zadeh'in önerdiği BM ilkelerini temel alarak geliştirilmiştir (Şen, 2004). Bulanık Sistemin çalışması şekil 2.10'da gösterildiği gibidir.



Şekil 2.10: Bulanık Sistemin genel yapısı.

Bulanıklaştırma: Sisteme girilen verilerin keskin değerlerini bulanık değerlere dönüştürme adımıdır. Keskin değerlerden bulanık değerlere dönüştürme işlemi belirli işlemlerin yapılması ile gerçekleştirilir.

Kural Tabanı: BM ile geliştirilen bir sistemin ikinci adımı kural tabanının oluşturulmasıdır. Kural tabanı oluşturulurken giriş ve çıkışlar arasındaki ilişkiler belirlenir.

Bulanık Çıkarım: Bulanık sistemde, sistem için gerekli olan çıkarımın elde edildiği bölümdür. Çıkarımın gerçekleştirilmesi aşamasında sisteme giriş parametreleri verilir buna bağlı olarak çıkış parametreleri oluşturulur. Oluşturulan çıkış parametrelerine göre sonuç değerleri elde edilir.

2.2.9. Melez Sistemler

Melez Sistemler, bir problemin çözümünde tek bir Yapay Zekâ yönteminin yerine birden fazla Yapay Zekâ yönteminin bir arada kullanılması ile oluşturulan sistemlere denilmektedir. Ortaya atılan problemin zorluk derecesine göre GA, MÖ, BM, YSA, Uzman Sistemler gibi Yapay Zekâ yöntemlerinin her birinin diğerine göre daha üstün özellikleri mevcuttur. Ortaya atılan problemin çözümünde Yapay Zekâ yöntemlerinin üstün niteliklerinden en üst seviyeden faydalanmak adına bu yöntemlerin bir kısmının veya hepsinin birleştirilmesi ile Melez Sistemler geliştirilmiştir. İnsanın karar verme süreci incelendiğinde onun da melez bir sistem gibi çalıştığı söylenebilir. Şöyle ki; insan bir konu hakkında karar verdiğinde ilk önce beş duyu organı ile aldığı bilgileri inceler daha sonra önceden edindiği uzman bilgileri ile bu

bilgiyi harmanlar kendi genlerindeki beceri yeteneklerine göre yoğurur ve bir sonuca ulaşır. İnsanın karar verme süreci taklit edildiğinde sinir ağlarından Bulanık Mantıka, uzman bilgisinden genetik yapıya birçok yöntemi kullandığı görülmektedir. Bu melez sistemler makinelere uyarlandığında makinelerin daha az hata payı ile sonuçlar üreteceği kaçınılmaz olacaktır. Literatürde melez sistemler ile ilgili çok fazla çalışma mevcuttur. Huang ve diğ. (2003) BM ile YSA'yı birleştirerek Bulanık Sinir Ağlarını önermiştir. Molani ve diğ., (2014) YSA ile GA bir arada kullanarak yeni bir model oluşturmuştur. Melez sistemler optimal çözümler üretmek için çok uygun sistemlerdir. Çağımız problem çözümleri için artık melez sistemlere doğru geçiş yapılmaktadır. Bilinen klasik yöntemler birçok problemin çözümü için yetersiz kalmaktadır. Melez sistemler günümüzde kullandığımız bir çok teknolojiye entegre edilmiştir. Bunlardan bir tanesi Nikko Securities şirketinin BM ve YSAnın kombinasyonundan oluşan hisse senetlerinin durumu ile ilgili kestirimde bulunması için kullandığı sinirsel-bulanık ve bir diğeri Mitsubishi firmasının sinirsel-bulanık çamaşır makinesi geliştirmesidir (Şahin, 2019).

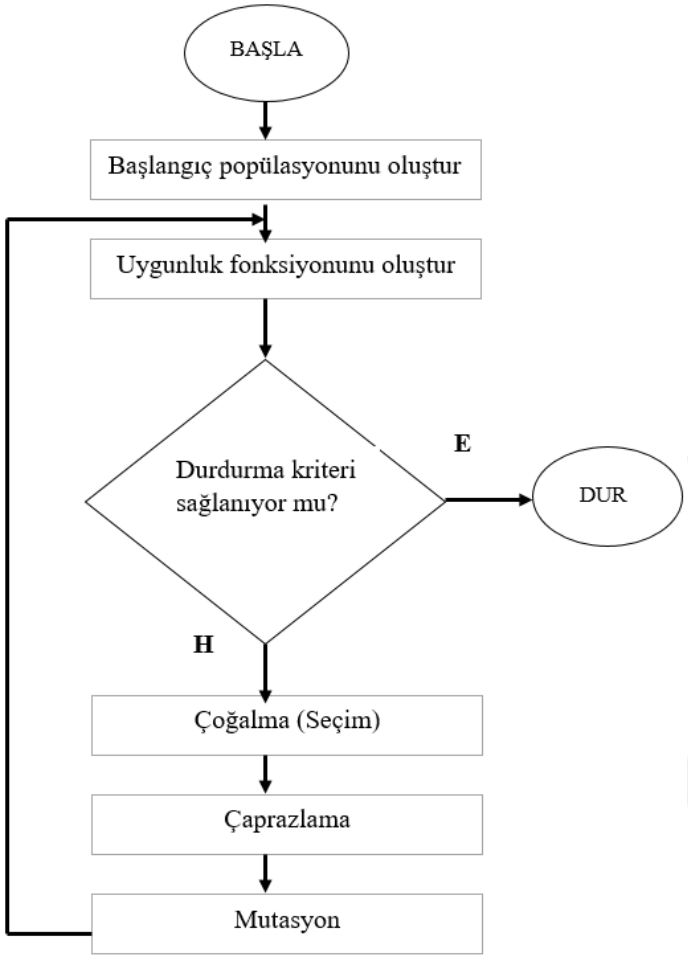
2.2.10. Genetik Algoritmalar

Charles Darwin'in ilkelerine dayanan ve evrimsel programlamanın bir alt dalı olan GA, doğada bulunan canlıların yaşadığı süreci örnek almaktadır. Buna göre en iyi nesiller kendi yaşamlarını korurken, kötü nesiller yok olmaktadır. 1960'lı yıllarda I. Rechenberg, "Evrim stratejileri" ismindeki çalışması ile evrimsel programlamayı gündeme taşımıştır. GA, evrimleşme stratejisi ve Genetik Programlama evrimsel programlama adı altında toplanmıştır. Genetik Programlama, GA'ların kodlanıp programlanmasına denmektedir. GA, doğal seçim kurallarına göre en iyi sonucu ya da en iyiye yakın sonucu elde etmeyi amaçlayan güçlü bir arama ve optimizasyon tekniğidir (Nabiyev, 2016). GA, John Holland tarafından ilk defa 1975 yılında şuan ki hali ile kullanılmıştır. Holland optimizasyon problemleri ile ilgili çalışmalarını GA kullanarak çözümlemeyi başarmıştır. Bu çalışmalarını "Adaptation in Natural and Artificial Systems" isimli kitabında yayımlaması neticesinde GA, Yapay Zekâ ve MÖ konularında bir alt alan olarak kullanılmaya başlanmıştır. GA'ların geliştirilmesi ve bilgisayar ortamına taşınması, John Holland, onun çalışma arkadaşları ve öğrencileri sayesinde gerçekleşmiştir (Kubat, 2014; Moghaddam, 2014). GA'ların en çok kullanıldığı alanlar, matematiksel formülasyonu bulunamayan, geleneksel metotlarla çözümü imkânsız olan ya da çözüm süreci sorunun

büyüklüğü ile orantılı olarak artan alanlardır. Bu alanlar farklı bilim dallarındaki optimizasyon problemleri olabilmektedir (Kubat, 2014). GA ele aldığı sorunlara birden fazla ayrı çözümden oluşan bir çözüm kümesi oluşturmaktadır. Bunun sayesinde arama uzayında aynı anda birçok nokta incelenmekte ve sonuçta bütünsel çözümü elde etme olasılığı yükselmektedir (Ebren Kara ve Şamlı, 2021). Her biri çok boyutlu uzay üzerinde bir vektör olan çözüm kümesindeki çözümler birbirinden tamamen bağımsızdır (Beasley, 1993). GA işlemleri (Nabiyev, 2016) şu şekildedir:

- Olası çözümlerin kodlandığı rastgele bir popülasyon oluşturulur.
- Toplumdaki bütün kromozomların uygunluk değerleri hesaplanır.
- Seçilen kromozomlar çiftleştirilerek tekrar klonlama ve dönüştürme operatörleri yürütülür.
- Belli büyüklükteki bir toplumu oluşturmak için mevcut kromozomlar çıkarılır yeni kromozomlar eklenir.
- Yeni toplumun başarısının bulunması için her bir kromozomun uygunluk değerleri baştan hesaplanır.
- Belirlenmiş bir sürede en iyi nesillerin oluşturulması için bu işlemler tekrarlanır.
- En uygun çözümün elde edilmesi toplumun hesaplanması esnasında en iyi bireylerin bulunmasıyla gerçekleşir.

Şekil 2.11'de GA'nın evrimleşme döngüsü verilmiştir.



Şekil 2.11: Genetik Algoritmalarda evrimleşme döngüsü.

GA, optimizasyon problemlerinde, ağ yerleşim problemlerinde, rota bulma problemlerinde, kontrol ve karar sistemlerinde, Yapay Zekâ'nın birçok alanında, Uzman Sistemelerin oluşturulmasında, MÖ, Robotik gibi daha bir çok alanda kullanılmaktadır.

2.2.11. Makine Öğrenmesi

MÖ, bilgisayarlara ve bilgisayar türevi makinelere insanlara benzer şekilde öğrenmesini ve insanlara benzer şekilde hareket etmesini veriler ve algoritmalar sayesinde öğretmektedir (Kaluza, 2016). Bilgisayar ve türevi makinelere apaçık programlanmadan öğrenme yeteneği sunan algoritmaları incelemek, tasarlamak ve geliştirmek ile ilgili olan MÖ kavramı 1959 yılında yaygınlaştırılmıştır (Prowmes, 2019). MÖ özünde, makinelerin tek başlarına doğru kararlar verebilme düşüncesi yatmaktadır. Bilgi güçtür prensibine dayanan MÖ, bir makine ne

kadar fazla eğitilirse o kadar fazla bilgilenir ne kadar fazla bilgilenirse o kadar doğru kararlar üretebilir (Ebren Kara ve Şamlı, 2021). Torkul ve diğ., (2017) çalışmasında MÖ; gözlemler ve ölçümler yapılarak sahip olunan verileri deneyim olarak alan makinelerin, bu deneyimlerden aritmetiksel algoritmalar yardımıyla mantıklı bağlantılar kurabilmesi aşamasıdır. Şekil 2.12'de bazı MÖ algoritmaları (PyCon, 2014) verilmiştir.

Makine Öğrenmesi Algoritmaları

	Eğiticisiz	**Eğiticili**
Sürekli	• Kümeleme & Boyut İndirgeme o Tekil Değer Ayrışımı o Temel Bileşen Analizi o K-Ortalamalar	• Regresyon o Lineer Regresyon o Polinomal Regresyon • Karar Ağacı • Rastgele Orman
Kategorik	• Birliktelik Analizi o Apriori o FB-Growth o Saklı Markow Modeli	• Sınıflandırma o K-en yakın komşuluk o Ağaçlar o Lojistik Regresyon o Naive-Bayes o Destek Vektör Makineleri

Şekil 2.12: Makine Öğrenmesi Algoritmaları.

MÖ'te temelde üç çeşit öğrenme yöntemi vardır:

- Eğiticili Öğrenme
- Eğiticisiz Öğrenme
- Pekiştirici, Yarı Eğiticili Öğrenme

Eğiticili Öğrenme:

Eğiticili Öğrenme, sonuç değerleri bilinen (etiketli) girdi değerlerini ele alıp girdilere bağlı olarak oluşan sonuçları inceleyerek aradaki ilişkiyi çözebilecek fonksiyonu oluşturmaya çalışır. Eğiticili Öğrenmede amaç yeni giriş verilerine karşılık gelecek sonuç değerlerini en doğru şekilde tahmin etmektir (Torkul ve diğ., 2017). Eğiticili Öğrenmede, sisteme verilen girdi değerleri işlenir sonuçta oluşacak çıktı değerlerinin ne olacağı öngörülmeye veya sistem

tarafından öğrenilmeye çalışılır. Bu işleme başlangıçta çıktı değerleri bilinen veriler üzerinde bir sınıflandırma yapılarak başlanılır daha sonra çıktı değeri bilinmeyen veriler üzerinde tahmin yapılır ve uygun sınıfa yerleştirilir (Aydın ve Özkul, 2015). Görüntü tanıma, spam olan e-postaları süzme, banka kartı sahtekârlıklarını belirleme gibi birçok MÖ işlemlerinin temelindeki sistem Eğiticili Öğrenme yöntemidir.

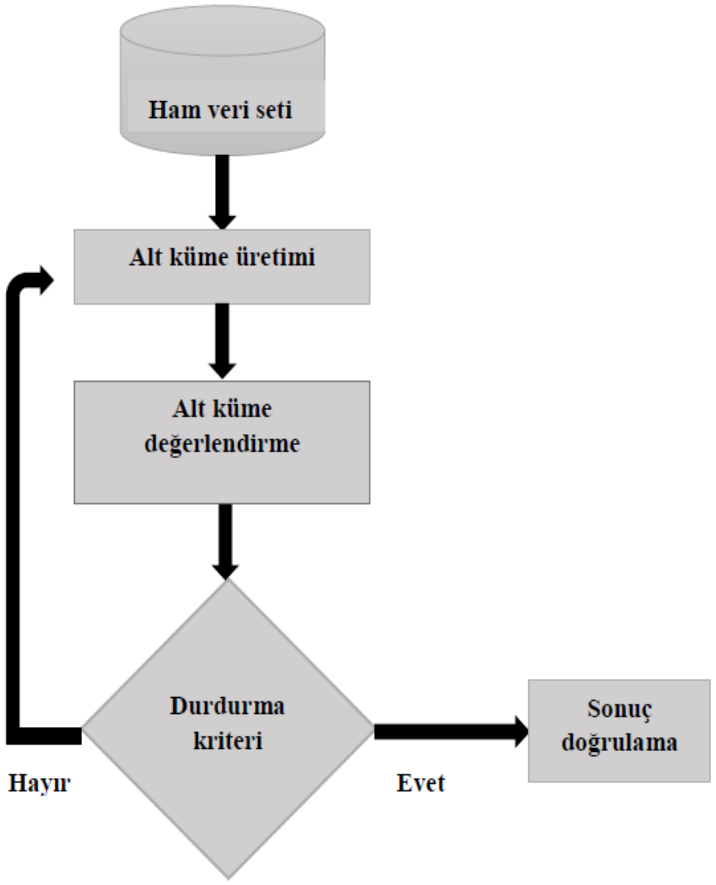
Eğiticisiz Öğrenme:

Eğiticisiz Öğrenme, sonuçları bilinmeyen (etiketsiz) verilerden öğrenmeye çalışma, veriler arasındaki saklı yapıyı bulma işlemidir. Bu öğrenme yönteminde sisteme dışarıdan herhangi bir müdahale yapılmaz. Sistemin öğrenme algoritmasını kullanarak veriler arasında var olan ama görülmeyen ilişkiyi ortaya çıkarması beklenmektedir (Alpaydın, 2011; Aydın ve Özkul, 2015). Sistem ne kadar çok veri incelerse yani sistem yeni verilerle eğitildikçe karar verme becerisi gelişmekte ve daha doğru tahminlerde bulunmaktadır.

Yarı Eğiticili Öğrenme:

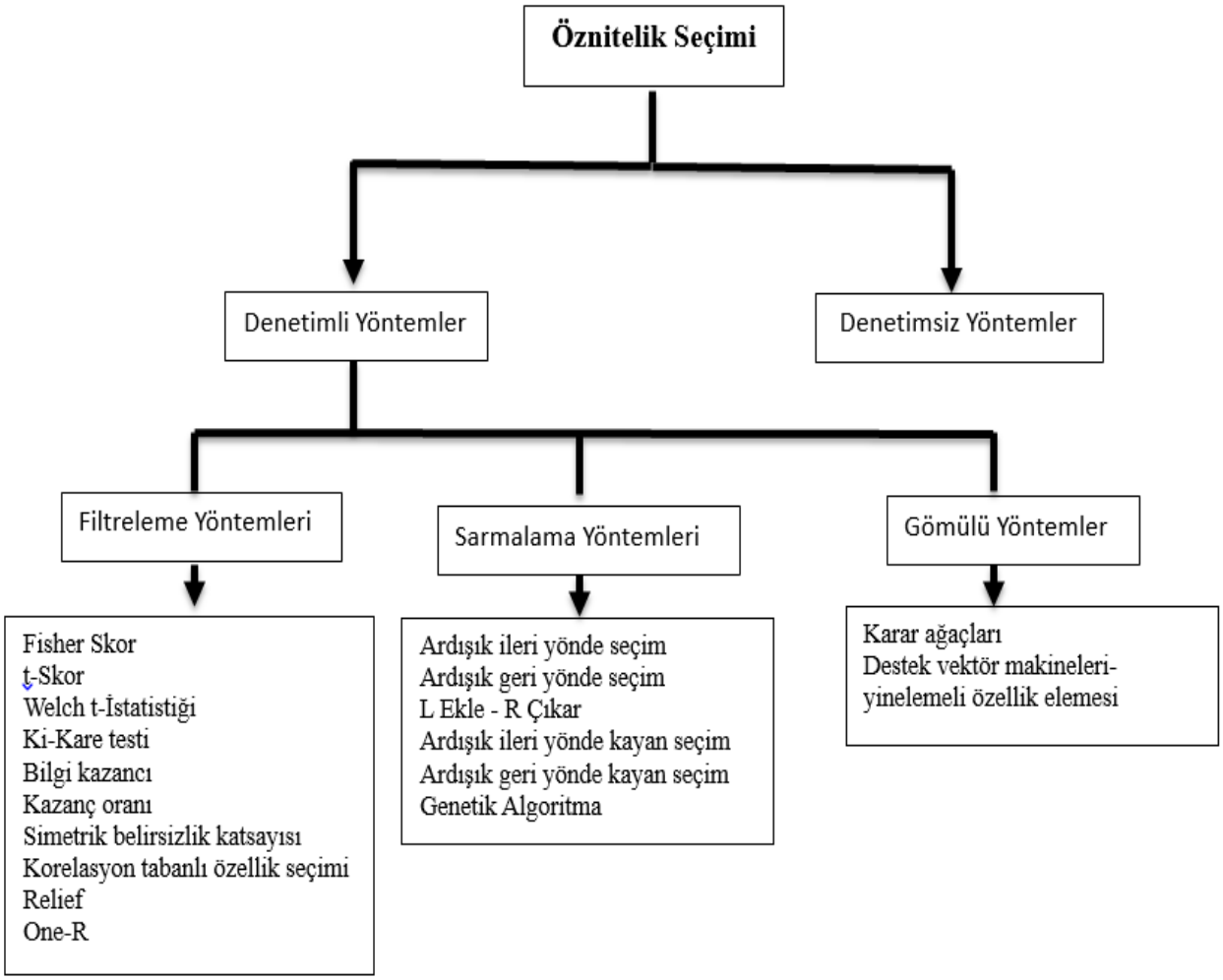
Sisteme verilen veriler arasında etiketli veri sayısı etiketsiz veri sayısından daha az ise Eğiticili Öğrenme de Eğiticisiz Öğrenme de işlevini doğru bir şekilde yapamayabilir. Böyle bir durumda Yarı Eğiticili Öğrenme yöntemi çok daha işlevsel olur. Yarı Eğiticili Öğrenme yöntemi, düşük miktardaki etiketlenmiş veriden yüksek miktardaki etiketlenmemiş veriyi tahmin etmek ve sınıflandırmaktır. Bu yöntem sisteme tecrübeyi ve yanılmayı öğretmektedir. Sistem eski tecrübelerinden öğrenir ve olabilecek en iyi sonucu bulmak için mevcut verilere göre cevabını uyarlayarak oluşturmaya çalışır (Kızılkaya ve Oğuzlar, 2018). MÖ'de, bir modelin eğitilmesi ve daha iyi öğrenmesi için çok miktarda veri toplanmaktadır. Toplanan büyük miktardaki verilerin bir kısmı model eğitimi için gereksiz olabilmektedir. Modelin eğitiminde kullanılan gereksiz veriler modelin yavaşlamasına ve doğru olmayan sonuçlar üretmesine sebebiyet vermektedir. Öznitelik seçimi, verilerin ön işlemden geçirildiği sırada modelin doğruluğunu ve performansını artırmak amacıyla ilgisiz ve gereksiz özniteliklerin atılması ve verilerin gürültüden temizlenmesi işlemidir. Sınıflandırma sistemlerinde verimli ve etkin bir biçimde kullanılan öznitelik seçimi modelin başarısını artırmaktadır (Ebren Kara ve Şamlı, 2021). Veri setine öznitelik seçimi uygulandıktan sonra veriler sınıflandırıldığında veri seti üzerindeki iş yükü azalmakta, gereksiz ve alakasız öznitelikler veri setinden atıldığı için sınıflandırmanın

doğruluk oranı artmaktadır. Sistem daha az eğitilmekte, ölçüm sayısı ve kullanılan bellek miktarı azalmaktadır. Bunun sayesinde veriler basit, hızlı ve doğruluk oranı yüksek bir şekilde sınıflandırılmış olmaktadır (Abe ve diğ., 1998; Huang ve Chow, 2005). Şekil 2.13'te öznitelik akış şeması gösterilmiştir. Akış şemasında görüldüğü gibi ham veri setinden öznitelik alt kümeleri oluşturulmaktadır. Elde edilen alt kümelerden hangilerinin kullanılacağı farklı formüller kullanılarak değerlendirilmektedir. Değerlendirmeler sonucunda bazı öznitelikler alt kümeye dâhil edilmeyerek elenmekte, elenmeyen öznitelikler seçilerek alt kümeye eklenmekte ve öznitelik belirleme işlemi kullanılan algoritmanın ölçütleri yerine getirilene kadar devam etmektedir.



Şekil 2.13: Öznitelik seçimi akış şeması.

Öznitelik Seçimi Yöntemleri: Öznitelik seçiminde kullanılan farklı algoritmalar vardır. Bular;: Filtreleme, sarmalama ve gömülü yöntemler olarak üç temel sınıfa ayrılmaktadır (Moghaddam, 2014). Filtreleme yöntemi, veri madenciliğinde kullanılan en eski öznitelik seçim yöntemi olarak bilinmektedir. Sadece istatistiksel kriterlere dayalı fonksiyonlar yardımıyla öznitelik seçimi yapan bir yöntemdir. Sarmalama yöntemi, öznitelikler üzerinde arama işlemi gerçekleştiren bir yöntemdir. Gömülü yöntem, içinde hem sınıflandırma algoritmasını hem de öznitelik seçim algoritmasını barındırdığından, sınıflandırma ve öznitelik seçim aşamalarını eşzamanlı olarak gerçekleştirebilen bir yöntemdir (Budak, 2018). Şekil 2.14'te öznitelik seçim yöntemlerinin bir kısmı gösterilmiştir.



Şekil 2.14: Öznitelik seçimi yöntemleri.

2.3. LİTERATÜR TARAMASI

Bu tez çalışması çerçevesinde, yazılım projelerinin maliyet tahmini ile ilgili geçmişten günümüze yayınlanmış Yapay Zekâ yöntemleri konularını içeren çeşitli yayınlar incelenmiştir.

Witting ve Finnie, (1997) YSA modelini kullanarak yazılım maliyet tahminlemesi yapmıştır. Araştırmacılar Avusturya Yazılım Ölçüt Birliğinin (Australian Software Metrics Association – ASMA) verilerini kullanarak YSA'nın geri-yayılım yapısını seçmiş ve 136 tane örnek proje üzerinde çalışmış ve %17 hata oranı ile tahminler elde ederek tahmin oranının iyileştirilmesi için tahmine etki edecek parametre sayısının artırılması gerektiğini savunmuştur. Finnie ve diğ. (1997) çalışmasında, üç tane kestirim yöntemi karşılaştırılmıştır. Karşılaştırma fonksiyon noktaları kullanılarak yapılmıştır. Çalışmada karşılaştıran yöntemler; Regresyon, YSA ve durum bazlı çıkarım yöntemleri olmuştur. 299 projenin veri seti üzerinde yapılan çalışmada regresyon yönteminin başarısı zayıf bulunmuştur fakat YSA ve durum bazlı çıkarım yöntemlerinden olumlu sonuçlar alınmıştır. Idri ve diğ. (2002), YSA yazılım maliyet tahmininde kolayca yorumlanabilir mi diye bir soru ile YSA'nın yazılım maliyet tahminindeki başarısını araştırmıştır. Araştırmacıların kullandıkları YSA, 17 giriş, 1 çıkış ve 1 ara katmandan oluşup veri kümesi olarak COCOMO81 veri setini almıştır.

Shan ve diğ. (2002) çalışmalarında, bir kısım araştırmacının, standart veri setlerinin çok fazla parametreye sahip olduğunu öne sürdüklerini ve bu parametre sayılarını azaltmak için çeşitli yöntemler denediklerini belirtmişlerdir. Bu araştırmada en uygun ölçüt kümelerinin belirlenmesi ve yazılım projelerinin maliyetinin tahmin doğruluğunun artırılması için eski proje veri kümelerine evrimsel bir yaklaşım olan GGGP (Grammar Guided Genetic Programming – Dilbilgisi Güdümlü Genetik Programlama) başarılı bir şekilde uygulanmıştır.

Ayyıldız (2007) çalışmasında yazılım maliyet tahmininde ölçüt kümesinin çok etkili olduğunu vurgulamış ve yaptığı araştırmalar sonucunda kendi yazılım ölçüt kümesini oluşturmuştur. Oluşturduğu ölçüt kümesini temel alarak sahip olduğu verileri kullanan bir YSA tabanlı yazılım maliyet tahmin modeli geliştirmiştir. Çalışma sonucunda geliştirilen model ile yapılan tahmin sonuçları, mevcut ölçütler kullanılarak yapılan önceki tahmin sonuçları ile karşılaştırılmıştır.

Karşılaştırmada iki tür veri seti kullanılmıştır. Birinci tür veriseti COCOMO81 veri kümesi, ikinci tür veriseti ise yeni oluşturulan ölçüt kümesine uygun belirli firmalardan toplanan veriler olmuştur. Karşılaştırma sonucunda yeni oluşturulan ölçüt kümesinin COCOMO'ya nazaran daha başarılı olduğunu gözlemlemiştir.

Başkeleş ve diğ. (2007) yaptıkları çalışmalada yazılım projelerinin maliyetini tahmin etmek için MÖ algoritmalarını çalıştıran bir model sunmuşlardır. Söz konusu modeli kamuya açık veri depolarındaki (NASA, USC) veri setleri ve Türkiye'deki yazılım şirketlerinden elde edilen veriler (SDR – SoftLab Data Repository) üzerinde denemişlerdir. Denemeler sonucunda herhangi bir veri kümesi için en iyi yöntemin değişebileceği ve sadece bir modelin kullanılmasının her zaman en iyi sonuçları üretemeyeceği gerçeğini ıspatlamışlardır. Yazılım maliyet tahmin modeli olarak YSA kullanan bir diğer çalışmada (Sezer, 2008), çok katmanlı ileri beslemeli, eğitim algoritması Delta Algoritması olan bir YSA oluşturulmuştur. YSA mevcut veri seti ile eğitilmiş eğitilen ağa test verileri sunulmuş ve hedeflenen çıktı elde edilmeye çalışılmıştır. YSA tabanlı yazılım maliyet tahmin modeli ile yapılan tahmin sonuçları COCOMO 2000 verileri ile karşılaştırılmıştır.

Adailer (2008) çalışmasında, literatürdeki MÖ ve Yapay Zekâ tabanlı yazılım tahminleme teknikleri karşılaştırılmıştır. Yazılım maliyetinin tahminini gerçekleştirmek üzere regresyon temelli ve YSA temelli iki tahminleme modeli kullanılmıştır. Her iki yöntem de istatistiksel öğrenme teorisi üzerine kurulmuştur. YSA'nın eğitilmesinde ve test aşamasında ISBSG (International Software Benchmarking Standards Group – Uluslararası – Yazılım Kıyaslama Standartları Grubu) veri seti sürüm 9 kullanılmıştır. Bu veriler istatistik ve MÖ kapsamındaki regresyon yönteminde katsayıların bulunmasında da kullanılmıştır. ISBSG veri setinde projelerin, hangi yazılım geliştirme ortamları kullanılarak gerçekleştirildiğinden, personel sayılarına, sürelerinden, kullanılan veri tabanı ve işletim sistemlerine, kaynak kod satır sayılarından, harcanan çabaya kadar birçok özellik yer almaktadır. YSA olarak çok katmanlı algılayıcı YSA modeli tercih edilmiştir. Ayrıca regresyon tabanlı yeni ve başarılı bir yazılım tahminleme modeli geliştirilmiştir. Sonuç olarak, regresyon ve çok katmanlı algılayıcı YSA modellerinin gerçekleştirimini yaparak bazı sonuçlar elde edilmiş ve bu sonuçlara dayanarak bu yöntemler değerlendirilmiştir.

Kulter ve diğ. (2009) çalışmasında, yazılım maliyet tahmininde kullanılmak üzere MÖ tabanlı bir model geliştirilmiştir. Geliştirilen modelde YSA ve ilişkisel bellek birleştirilerek bir arada kullanılmıştır. Model için kullanılan YSA'da birden fazla MLP (Multilayer Perceptrons – Çok Katmanlı Algılayıcı) bir araya getirilmiştir. Model yazılım maliyet tahmini yaptığında her bir MLP kendi tahmin sonucunu üretmiş ve tahmin sonuçları birleştirilmiştir. Kullanılan modelde hem ilişkisel bellek hem de birleştirme yapılarak yeni bir model oluşturulmuştur.

Kartal Karataş (2011) çalışmasında, yazılım projelerinin maliyet tahmini için bir YSA modeli anlatılmıştır. Çalışma için öncelikle yazılım maliyet hesaplama yöntemleri araştırılmıştır. Ardından YSA ile gerçekleştirilen yazılım maliyet tahmin çalışmaları incelenmiştir. XOR probleminin çözüm yolundan faydalanılarak maliyet tahmini için bir YSA oluşturulmuştur. YSA'nın eğitiminde ve test edilmesinde COCOMO veri kümesi kullanılmıştır. Elde edinilen tahminler; oluşturulan modelin kabul edilebilir tahminler ortaya koyduğunu göstermiştir. Tasarlanan ağın yazılım şirketleri tarafından kullanılabilmesi için, ağın da içine entegre edildiği bir web sayfası oluşturulmuştur. Bu web sayfası, yazılım şirketlerine maliyet tahmini sağladığı gibi YSA ile ilgili dokümanları içermiş ve yazılım şirketlerinin proje bilgilerini tutabileceği bir veri bankası görevi üstlenmiştir.

Singh ve Misra (2012) optimizasyon yöntemlerinden ikili genetik algoritma kullanarak COCOMO modelinin bileşenlerini yeniden ayarlayan ayrıntılı bir çalışma sunmuşlardır. Bu çalışmada daha iyi bir yazılım maliyet tahmini yapabilecek şekilde değiştirilmiş COCOMO bileşenleri ile yeni bir model önerilmiştir. Önerilen modelin performansı NASA veri seti üzerinde test edilmiş ve önceki modellerle karşılaştırılarak genetik algoritma tekniğinin sağlamlığı doğrulanmaya çalışılmıştır.

Tran ve diğ., (2015) çalışmasında çok boyutlu veriler üzerinde öznitelik inşası ve öznitelik seçimi için Genetik Programlamanın kullanımı araştırılmıştır. Çalışmada yedi veri seti üzerinde dört farklı sınıflandırma algoritması kullanılmıştır. Amaç çok boyutlu veri setleri üzerinde sınıflandırma yapmak için öznitelik oluşturma ve öznitelik seçiminde Genetik Programlamanın performansını araştırmak olmuştur. Genetik Programlama tarafından oluşturulan veya seçilen özniteliklerin, çok daha küçük bir öznitelik kümesiyle sınıflandırma algoritmalarının

performansını iyileştirebileceği gözlemlenmiştir. Sonuçlar, Genetik Programlamanın daha iyi ayırt etme yeteneğine sahip özellikleri seçme ve oluşturma potansiyeline sahip olduğunu göstermiştir. Diğer bir çalışmada (Keskin ve Alptekin, 2016), 'E–Bursum' yazılımının maliyet tahmini, ilk önce İşlev puanı analizi yöntemi ile gerçekleştirilmiştir. Daha sonra aynı maliyet tahmini, YSA kullanılarak tekrarlanmıştır. Tahmini değerler gerçek değerlerle karşılaştırıldığında, tahmindeki doğruluk oranı %90'ın üzerinde çıkmıştır. İşlev puanı analizinin gerçeğe yaklaşımı %74 – %97 aralığında bulunurken, YSA ile elde edilen sonuçlar %92 – %98 aralığında olmuştur.

Başar (2017), çalışmasında, algoritmik ve algoritmik olmayan sezgisel yöntemlerin birlikte kullanıldığı yeni bir yaklaşım geliştirilmiştir. Geliştirilen yaklaşıma "Sezgisel Bulanık İkili Karşılaştırma Tekniği" adı verilmiştir. Kullanılan teknikte yazılım projelerinin maliyet tahmininin yapılması için en uygun ölçütler uzman görüşü ile belirlenmiş daha sonra Klasik İkili Karşılaştırma yöntemi ile ölçütlerin etki dereceleri elde edilmiştir. Bilhassa önceki kesinleşmiş ölçekler kullanılarak elde edilen değerlendirmelerde, ölçütler göreli öneminin ölçekte var olan değerler ile tam olarak karşılanamaması, bulanık değerler ile karar vermenin önemini arttırmıştır. Bu yüzden, uzman görüşü yardımıyla belirlenen ölçütlerin öneminin tespiti için çalışmada ayrıca Sezgisel Bulanık İkili Karşılaştırma sunulmuştur. Sunulan tekniğin tahmin doğruluğunun belirlenmesi için yazılım alanında çalışmalar yapan bir şirketten yazılım projelerinin verileri temin edilmiş ve bu gerçek veriler üzerinden yazılım maliyet tahmini gerçekleştirilmiştir. Elde edilen tahmin sonuçları klasik yöntemlerle elde edilen tahmin sonuçları ile karşılaştırıldığında sunulan tekniğin daha yüksek doğruluk oranıyla tahmin yapabildiği belirlenmiştir.

Başka bir çalışmada (Gültekin, 2019) farklı metodolojiler kullanılarak hazırlanan yazılım projeleri için farklı yazılım maliyet tahmin yöntemi sunulmuştur. Araştırmacıya göre yazılımın geliştirildiği metodoloji yazılım maliyet tahmininde önemli bir etkendir bu yüzden yazılım projelerinin geliştirildiği metodoloji göz önünde bulundurularak tahmin yöntemi geliştirildiğinde maliyet tahmininin doğruluk oranı artmaktadır. Bu bağlamda yazılım maliyet tahmini için 3 model sunulmuştur. İlk sunulan model 6 farklı regresyon tekniği kullanılarak oluşturulan modeldir. Bu modellerde COCOMO veri setleri kullanılmıştır. İkinci sunulan

model YSA temelli bir tahmin modelidir. Bu modelde kullanılan COCOMO veri setleri özelliklerine göre guruplandırılarak kullanılmıştır. Sunulan son modelde Scrum metodolojisi ile hazırlanan yazılım projelerinin maliyet tahmini için geliştirilmiştir. Geliştirilen modelde regresyon tabanlı MÖ algoritmaları kullanılmıştır.

Diğer bir çalımada (Marepelli, 2019) WEKA programında bulunan MÖ algoritmaları kullanılarak COCOMO veri setleri üzerinde yazılım maliyet tahmini yapılmıştır. Yapılan maliyet tahmininde kullanılan iki tane MÖ algoritmasının tahmin değerlerinin hata oranları ve korelasyon sonuçları incelenmiştir.

3. MALZEME VE YÖNTEM

Bu bölümde tez çalışmasında kullanılan veri setleri, değerlendirme ölçütleri, uygulama platformu ve yöntemler alt başlıklar halinde açıklanmıştır.

3.1. VERİ SETLERİ

Bu bölümde PROMISE veri deposundan temin edilen COCOMO81, COCOMONASA, COCOMONASA2, China, Albrecht, Finnish, Kemerer, Maxwell ve Miyazaki94 veri setleri incelenmiştir. Tablo 3.1'de kullanılan veri setlerinin bilgileri verilmiştir.

Tablo 3.1: Veri setleri bilgileri.

Veri Seti	Kayıt Sayısı	Öznitelik Sayısı	Boyut (ölçü birimi)	Maliyet (ölçü birimi)
COCOMO81	63	17	LOC	Adam - Ay
COCOMONASA	60	17	LOC	Adam - Ay
COCOMONASA2	93	24	LOC	Adam - Ay
China	499	19	Fonksiyon Noktası	Adam - Saat
Albrecht	24	8	Fonksiyon Noktası	Adam - Saat
Finnish	38	9	Fonksiyon Noktası	Adam - Saat
Kemerer	15	8	KSLOC	Adam - Ay
Maxwell	62	27	Fonksiyon Noktası	Adam - Saat
Miyazaki94	48	9	KSLOC	Adam - Ay

Veri deposundan temin edilen veri setlerinde gerçek yazılım projelerinin verileri tutulmaktadır. Her biri farklı sayıda proje verisi barındıran veri setlerinde bağlı ve bağımsız öznitelikler bulunmaktadır. Bu öznitelikler maliyet tahminin gerçekleştirilmesinde kullanılmaktadır. Eğer bir öznitelik gerçek maliyet değerini veriyorsa bağlı öznitelik; maliyetle alakalı değerleri veriyorsa bağımsız öznitelik olarak adlandırılır. Veri setlerinde bulunan bağımsız öznitelikler, bağlı özniteliğin değerini belirlemektedir. Veri setlerinde bulunan bazı bağımsız özniteliklerin yazılım maliyet tahminine fazla etkisi bulunmamaktadır. USC (University of South California – Güney Kalforniya Ünviersitesi) Sistem ve Yazılım Mühendisliği Merkezine mensup olan COCOMO, USC veri seti olarak kabul edilmektedir (Kültür, 2006).

COCOMO81: COCOMO81 veri seti 1981 yılında önerilmiştir. Veri setinde 63 yazılım projesinin kaydı ile 17 öznitelik bulunmaktadır. Bu öznitelikler; projenin kod satır sayısı, projenin gerçek geliştirme maliyeti ve 15 adet maliyet çarpanıdır.

COCOMONASA: COCOMONASA veri setinde 1980'ler ve 1990'larda farklı merkezlerden toplanmış 60 NASA projesine ait kayıt ve 17 öznitelik bulunmaktadır.

COCOMONASA2: COCOMONASA2 veri seti bazı kayıtlarda NASA93 veri seti olarak geçmektedir (Bosu ve MacDonell, 2019). NASA93 veri seti NASA tarafından 1971 ile 1987 yılları arasında üretilen 93 proje verisinin beş farklı geliştirme merkezinden toplanmasıyla oluşturulmuştur. Veri seti 93 NASA projesine ait kayıt ve 24 öznitelikten oluşmaktadır. Bu 24 öznitelikten 7 tanesinin yazılım maliyetine etkisi bulunmadığından çıkarılmıştır. Bu öznitelikler; recordnumber (benzersiz bir numara), projectname (proje ismi), cat2 (uygulama kategorisi), forg (uçuş mu yer sistemi mi?), center (hangi NASA merkezi?), year (geliştirme yılı) ve mode (geliştirme modu) özniteliklerdir. Bunların dışında 17 öznitelikten 15'i COCOMO maliyet tahmininde kullanılan maliyet ölçüsü, kod satır sayısını belirten loc ve gerçek maliyeti belirten act_effort özniteliklerdir. Tablo 3.2'de maliyet tahmininde kullanılan 15 öznitelik gösterilmiştir.

Tablo 3.2: COCOMO maliyet faktörleri.

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

Promise veri deposundan alınan her biri farklı sayıda proje verisi barındıran COCOMO81, COCOMONASA, COCOMONASA2 veri setlerinde bağlı ve bağımsız öznitelikler bulunmaktadır. Eğer bir öznitelik gerçek maliyet değerini veriyorsa bağlı öznitelik; act_effort (yazılım geliştirme çabası), maliyetle alakalı değerleri veriyorsa; rely, data, cplx, time, stor, virt, turn, acap, aexp, pcap, vexp, lexp, modp, tool, sced, loc (kod satır sayısı) bağımsız öznitelik

olarak adlandırılmaktadır. Tablo 3.3'de veri setlerinde bulunan özniteliklerin alabildiği en büyük ve en küçük değer aralıkları incelenmiştir.

Tablo 3.3: COCOMO maliyet faktörlerinin standart sayısal değerleri.

Öznitelik	Çok Düşük	Düşük	Normal	Yüksek	Çok Yüksek	Ekstra Yüksek	Verimlilik
acap	1,46	1,19	1,00	0,86	0,71		2,06
pcap	1,42	1,17	1,00	0,86	0,70		1,67
aexp	1,29	1,13	1,00	0,91	0,82		1,57
modp	1,24	1,10	1,00	0,91	0,82		1,34
tool	1,24	1,10	1,00	0,91	0,83		1,49
vexp	1,21	1,10	1,00	0,90			1,34
lexp	1,14	1,07	1,00	0,95			1,20
sced	1,23	1,08	1,00	1,04	1,10		e
stor			1,00	1,06	1,21	1,56	-1,21
data		0,94	1,00	1,08	1,16		-1,23
time			1,00	1,11	1,30	1,66	-1,30
turn		0,87	1,00	1,07	1,15		-1,32
virt		0,87	1,00	1,15	1,30		-1,49
cplx	0,70	0,85	1,00	1,15	1,30	1,65	-1,86
rely	0,75	0,88	1,00	1,15	1,40		-1,87

Albrecht: Albrecht veri seti, IBM veri işleme hizmetlerinde gerçekleştirilen projelerden toplanan 24 kayıttan oluşur. Projeler COBOL, PL/I ve DMS programlama dilleri kullanılarak geliştirilmiştir. Projelerin boyutu ve karmaşıklığı, Albrecht tarafından önerilen fonksiyon noktası yaklaşımı kullanılarak ölçülmüştür (Albrecht ve Gaffney, 1983).

Tablo 3.4: Albrecht veri seti istatistikleri.

Sno	Öznitelik	Tanımlama	EnKüçük	EnBüyük	Ortalama
1	Input	No of inputs – Giriş sayısı	7	193	40,25
2	Output	No of outputs – Çıkış sayısı	12	150	47,25
3	Inquiry	No of inqueries – Sorgu sayısı	0	75	16,88
4	File	No of master files – Ana dosya sayısı	3	60	17,38
5	FPAdj	Function points adjustment – Fonksiyon noktaları ayarı	0,75	1,2	0,99
6	RawFPcounts	Count of raw function points – Ham fonksiyonlarının sayısı	189,52	1902	638,54
7	AdjFP	Adjusted function points – Ayarlanmış fonksiyon noktaları	199	1902	647,63
8	Effort	Person hours – Adam saat	0,5	105,2	21,88

Finnish: Finnish veri seti, TIEKE organizasyonu tarafından Finlandiya'daki dokuz firmadan toplanan 40 proje verisinden oluşmaktadır. Projelerin boyutu ve karmaşıklığı fonksiyon noktası

yaklaşımı kullanılarak ölçülmüştür (Kitchenham ve Kansala, 1993). Tablo 3.5'te Finnish veri seti istatistikleri verilmiştir.

Tablo 3.5: Finnish veri seti istatistikleri.

Sno	Öznitelik	Tanımlama	EnKüçük	EnBüyük	Ortalama
1	ID	Project no – Proje numarası	1	38	19,05
2	dev,eff,hrs	Development effort hours – Geliştirme çabası saati	460	26670	7678,29
3	hw	Hardware type – Donanım tipi	1	3	1,26
4	at	Application type – Uygulama tipi	1	5	2,24
5	FP	Function point data – Fonksiyon noktası verileri	65	1814	763,54
6	co	Application area – Uygulama alanı	2	10	6,26
7	prod	Project duration (calender months) – proje süresi (takvim ayları)	1,47	29,47	10,07
8	lnsize	System requirements size in raw Albrecht function points – Ham Albrecht fonksiyon noktalarında sistem gereksinimleri boyutu	4,17	7,5	6,36
9	lneff	Effort provided by application user – Uygulama kullanıcısı tarafından sağlanan çaba	6,13	10,19	8,40

China: China veri seti, 2010 yılında PROMISE deposuna eklenmiş diğer veri setlerine göre daha yeni bir veri setidir, 499 kayıttan oluşmaktadır (Bosu ve MacDonell, 2019). China veri seti 18'i bağımsız değişken ve 1 tanesi bağımlı değişken olmak üzere 19 öznitelikten oluşmaktadır. Tablo 3.6'da China veri seti istatistikleri verilmiştir.

Tablo 3.6: China veri seti istatistikleri.

Sno	Öznitelik	EnKüçük	EnBüyük	Ortalama
1	ID	1	499	250
2	AFP	9	17518	487
3	Input	0	9404	167
4	Output	0	2455	114
5	Enquiry	0	952	62
6	File	0	2955	91
7	Interface	0	1572	24
8	Added	0	13580	360
9	Changed	0	5193	85
10	Deleted	0	2657	12
11	PDR_AFP	0.3	83.8	12
12	PDR_UFP	0.3	96.6	12
13	NPDR_AFP	0.4	101	13
14	NPDU_UFP	0.4	108	14
15	Resource	1	4	1
16	Dev.Type	0	0	0
17	Duration	1	84	9
18	N_effort	31	54620	4278
19	Effort	26	54620	3921

Kemerer: Kemerer veri seti (Kemerer, 1987), veri işleme yazılımı geliştiren bir Amerikan firmasından toplanmıştır. Bu veri seti sekiz özniteliğe sahip 15 projeden oluşmaktadır. Veri setindeki en eski proje 1981'de başlamış olup projelerin çoğu 1983'te başlamıştır. Veri setindeki proje verileri 1985'te toplanmıştır. Tablo 3.7'de Kemerer veri seti istatistikleri verilmiştir.

Tablo 3.7: Kemerer veri seti istatistikleri.

Sno	Öznitelik	Tanımlama	EnKüçük	EnBüyük	Ortalama
1	ID	Project ID – Proje kimliği	1	15	8
2	Language	Software used – Kullanılan yazılım	1	3	1,2
3	Hardware	Hardware used – Kullanılan donanım	1	6	2,33
4	Duration	Duration – Süre	5	31	14,27
5	KSLOC	Number of source lines code in thousands – Bin olarak kaynak kod satır sayısı	39	450	186,57
6	AdjFP	Adjusted function points – Ayarlanmış fonksiyon noktaları	99,9	2306,8	999,14
7	RAWFP	Raw function points – Ham fonksiyon noktaları	97	2284	993,87
8	EffortMM	Effort Man Months – Adam ay çaba	23,2	1107,31	219,25

Miyazaki94: Miyazaki94 veri seti, Büyük Sistem Kullanıcıları Grubu tarafından toplanmıştır (Miyazaki ve diğ., 1994). Veriler, 20 farklı kuruluşta ve bu kuruluşlardaki birden fazla bölümde geliştirilen 48 COBOL projesinden elde edilmiştir. Her proje için 9 öznitelik vardır. Tablo 3.8'de Miyazaki94 veri seti istatistikleri verilmiştir.

Tablo 3.8: Miyazaki94 veri seti istatistikleri.

Sno	Öznitelik	Tanımlama	EnKüçük	EnBüyük	Ortalama
1	ID	Project ID – Proje kimliği			
2	KLOC	Number of COBOL source lines in thousands – Binlerce COBOL kaynak satırı sayısı	6,9	417,6	70,79
3	SCRN	Number of different input or output screens – Farklı giriş veya çıkış ekranlarının sayısı	0	281	33,69
4	FORM	Number of different (report) forms – Farklı (rapor) form sayısı	0	91	22,37
5	FILE	Number of different record formats – Farklı kayıt biçimlerinin sayısı	2	370	34,81
6	ESCRN	Total number of data elements in all the screens – Tüm ekranlardaki toplam veri öğesi sayısı	0	3000	525,60
7	EFORM	Total number of data elements in all the forms – Tüm formlardaki toplam veri öğesi sayısı	0	1566	460,67
8	EFILE	Total number of data elements in all the files – Tüm dosyalardaki toplam veri öğesi sayısı	57	45000	1854,58
9	MM	Effort measured in man-months - Adam-ay olarak ölçülen çaba	5,6	1586	87,47

Maxwell: Maxwell veri seti bir Fin ticari bankasından toplanmıştır. 27 nitelik ile temsil edilen 62 projeden oluşmaktadır (Maxwell, 2002). Projelerin başlangıç yılları 1985 ile 1993 yılları arasındadır. Tablo 3.9'da Maxwell veri seti istatistikleri verilmiştir.

Tablo 3.9: Maxwell veri seti istatistikleri.

Sno	Öznitelik	Tanımlama	EnKüçük	EnBüyük	Ortalama
1	Syear	Year – Geliştirme yılı	85	93	89,58
2	App	Application type – Uygulama çeşidi	1	5	2,35
3	Har	Hardware platform – Donanım platformu	1	5	2,61
4	Dba	Database – Veritabanı	0	4	1,03
5	Ifc	User interface – Kullanıcı arayüzü	1	2	1,93
6	Source	Where developed – Nerede geliştirildi	1	2	1,87
7	Telonuse	Telon use – Telon kullanımı	0	1	0,24
8	Nlan	# of development languages – Geliştirme dili	1	4	2,55
9	T01	Customer participation – Müşteri katılımı	1	5	3,05
10	T02	Development Env, adequacy – Geliştirme ortamı, yeterlilik	1	5	3,05
11	T03	Staff availability – Personel durumu	2	5	3,03
12	T04	Standards use – Standartlar kullanımı	2	5	3,19
13	T05	Methods use – Yöntem kullanımı	1	5	3,05
14	T06	Tools use – Araçlar	1	4	2,90
15	T07	Software logical complexity – Yazılım mantıksal karmaşıklığı	1	5	3,24
16	T08	Requirements volatility – Gereksinim oynaklığı	2	5	3,81
17	T09	Quality requirements – Kalite gereksinimleri	2	5	4,06
18	T10	Efficiency requirements – Verimlilik gereksinimleri	2	5	3,61
19	T11	Installation requirements – Kurulum gereksinimleri	2	5	3,42
20	T12	Staff analysis skills – Personel becerisi analizi	2	5	3,82
21	T13	Staff application knowledge – Personel uygulama becerisi	1	5	3,06
22	T14	Staff tool skills – Personel araç becerisi	1	5	3,26
23	T15	Staff team skills – Personel takım becerileri	1	5	3,34
24	Duration	Duration – Süre	4	54	17,21
25	Size	Function points – Fonksiyon noktaları	48	3643	673,30
26	Time	Time – Zaman	1	9	5,58
27	Effort	Work hours Effort – Çalışma saati çabası	583	63694	8223,21

3.2. DEĞERLENDİRME KRİTERLERİ

Bu tezde değerlendirme kriterleri olarak korelasyon katsayısı MAE, MAPE, RMSE, RAE, ve RRSE kullanılmıştır.

3.2.1. Korelasyon Katsayısı

Korelasyon katsayısı, iki farklı değişken arasındaki bağın gücünü ve yönünü belirtir. Korelasyon, değişkenler arasındaki ilişkiyi korelasyon katsayısı ise ilişkinin durumunu belirtmektedir. Korelasyon katsayısı ilişkinin durumuna göre -1 ile 1 arasında bir sayı değeri alabilmektedir. Sayı değerinin negatif bir değer olması değişkenler arasında ters bir ilişkinin olduğunu göstermektedir. Yani değişkenlerden biri artarken diğeri azalmaktadır. Sayı değerinin pozitif bir değer olması değişkenler arasında doğrusal bir ilişkinin olduğunu göstermektedir yani değişkenlerden biri artarken diğeri de artmaktadır. Korelasyon katsayısının gösteren sayı değerinin sıfır olması iki değişken arasında herhangi bir ilişkinin olmadığı anlamına gelmektedir. Korelasyon katsayısı 1'e yaklaştıkça değişkenler arasındaki ilişkinin arttığı 0'a yaklaştıkça aradaki ilişkinin azaldığı anlaşılmaktadır.

3.2.2. RMSE

RMSE (Root Mean Squared Error – Kök Ortalama Kare Hatası), gerçek değerden tahmini değer çıkarılarak karesi alınır, bulunan sonuçlar toplanarak ortalamalarının karekökü alınır yani RMSE karesel hataların ortalamasının karekökü'dür. Büyük hataların RMSE üzerinde orantısız olarak büyük bir etkisi vardır bu da RMSE'nin aykırı değerlere karşı hassas olduğunu gösterir. Bundan dolayı RMSE en fazla büyük hataların istenmediği durumlarda kullanılır. Formülü Denklem 3.1'de verilmiştir.

$$RMSE = \sqrt{\frac{1}{\mathrm{n}}\sum_{i=1}^{n}(T_i - G_i)^2} \tag{3.1}$$

Burada T_i = tahmini değer, G_i= gerçek değer, n = örnek sayısıdır.

3.2.3. MAE

MAE (Mean Absulate Error – Ortalama Mutlak Hata), gerçek değerlerin tahmin edilen değerle olan farklarının toplamının ortalamasını veren hata oranıdır. Formülü Denklem 3.2'de verilmiştir.

$$MAE = \frac{1}{\mathrm{n}}\sum_{i=1}^{n}|T_i - G_i| \tag{3.2}$$

Burada T_i = tahmini değer, G_i = gerçek değer, n = örnek sayısı'dır

3.2.4. RAE

RAE (Relative Absulate Error – Bağıl Mutlak Hata), gerçek değer ve tahmini değer arasındaki farkın toplamını bulup bunu gerçek değerlerin ortalamsından gerçek değerlerin çıkarılmasına bölmektir. Formülü Denklem 3.3'te verilmiştir.

$$RAE = \frac{\sum_{i=1}^{n}|T_i - \mathrm{G}_i|}{\sum_{i=1}^{n}|\mathrm{G}_m - \mathrm{G}_i|} \tag{3.3}$$

Burada T_i = tahmini değer, G_i = gerçek değer, G_m= gerçek değerlerin ortalaması, n = örnek sayısıdır.

3.2.5. RRSE

RRSE (Root Relative Squared Error – Kök Ortalama Kare Hata), tahmin edilen her değer ile gerçek değerin farkının karesinin; gerçek değerlerin ortalamasından her bir gerçek değerin farkının alınmasıyla karesinin alındıktan sonra bölünmesinin ve sonucun karekökünün alınmasıdır. Kareleri alınmış hatalar bölünerek normalleştirilir. Göreceli kare hatasının karekökü alınarak, hata tahmin edilen miktarda aynı boyuta indirilir. Denklem 3.4'te RRSE formülü verilmiştir.

$$RRSE = \sqrt{\frac{\sum_{i=1}^{n}(T_i - \mathrm{G}_i)^2}{\sum_{i=1}^{n}(\mathrm{G}_m - \mathrm{G}_i)^2}} \tag{3.4}$$

Burada T_i = tahmini değer, G_i = gerçek değer, G_m= gerçek değerlerin ortalaması, n = örnek sayısıdır.

3.2.6. MMRE

MMRE (Mean Magnitude Of Relative Error – Göreceli Hatanın Ortalama Büyüklüğü), çoklu tahminlerin doğruluğu değerlendirilirken bir toplama yöntemi gerekmektedir. Aritmetik ortalaması da alınan değerlendirmeye MMRE denmektedir. Bir tahmin doğruluğu ölçülürken, göreceli hatanın büyüklüğü (MRE-Magnitute Relative Error) sıklıkla kullanılır. Hatanın gerçek gözlemlenen değere oranının mutlak değeri olarak tanımlanır: $|(gerçek - tahmin\ edilen)/gerçek|$. Bu 100 ile çarpıldığında mutlak yüzde hatayı (APE-Absolute Percentage Error) verir. MMRE, kullanılan modellerin başarısı hakkında fikir vermektedir. Formülü Denklem 3.5'te verilmiştir.

$$MMRE = \frac{1}{\mathrm{n}} \sum_{i=1}^{n} \left| \frac{T_i - G_i}{G_i} \right| \tag{3.5}$$

Burada T_i = tahmini değer, G_i= gerçek değer, G_m= gerçek değerlerin ortalaması, n = örnek sayısıdır.

3.2.7. MAPE

Bazı disiplinlerde MAPE (Mean Absolute Percentage Error – Ortalama Mutlak Hata Yüzdesi), MMRE olarak bilinmektedir (Tofallis, 2015). MAPE tahmin değerlendirmelerinde en sık kullanılan ölçüdür. MAPE çoğu çalışmada %MMRE olarak kullanılmıştır. Formülü Denklem 3.6'da verilmiştir.

$$MAPE = \frac{1}{\mathrm{n}} \sum_{i=1}^{n} \left| \frac{T_i - G_i}{G_i} \right| \mathrm{x}100 \tag{3.6}$$

Burada T_i = tahmini değer, G_i= gerçek değer, G_m= gerçek değerlerin ortalaması, n = örnek sayısıdır.

3.3. UYGULAMA PLATFORMLARI

Tez çalışmasında WEKA ortamı kullanılmıştır. WEKA ortamı Yeni Zelanda'da bulunan Waikato Üniversitesindeki bir doktora öğrencisi tarafından Java dilinde geliştirilmiş, GPL (General Public Licence – Genel Kamu Lisansı) lisansına sahip açık kaynaklı ve ücretsiz bir uygulamadır. WEKA programı ticarî programlara karşılık daha çok bilimsel çalışmalarda ham verileri sınıflandırma, kümeleme, görselleştirme, bölütleme, tahminleme, veriler arasında ilişki kurma, öznitelik seçimi, veri ön işleme gibi MÖ ve Veri Madenciliği işlemlerini gerçekleştirebilecek algoritmaları barındırmaktadır (Witten ve diğ., 2011; Ebren Kara ve Şamlı, 2021). Bu tezde, Genetik Programlamanın mevcut olduğu WEKA 3.4.12 (WekaGP, 2007) ve WEKA 3.9 (WEKA, 2018) sürümleri kullanılmıştır. WEKA kurulumu sırasında weka.jar dosyası da gelmektedir. Bu jar dosyasında WEKA kütüphaneleri bulunmaktadır. Bu sayede başka bir platformdan (Java, C# gibi) WEKA sınıflarına erişilerek projeler geliştirilmektedir. WEKA'da veri setleri arff (Attribute Relationship File Format) formatında hazırlanmaktadır. Veri setlerinde bulunan değişkenler sayısal (decimal) ve kategorik (nominal) değerlerden oluşmaktadır.

WEKA programında kullanılan veri setinin tahmin edilen sütun değerinin nominal ya da sayısal olmasına göre farklı hata formülleri kullanılmaktadır. Veri seti dosyanın tahmin edilen class adlı sütunu sayısal ise hata ölçüm değerleri Şekil 3.1'deki gibi olmaktadır.

```
Time taken to build model: 0.01 seconds

=== Cross-validation ===
=== Summary ===

Correlation coefficient                  0.9889
Mean absolute percentage error           0.2253
Mean absolute error                    362.939
Root mean squared error                968.6259
Relative absolute error                  9.809  %
Root relative squared error             14.9185 %
Total Number of Instances              499
```

Şekil 3.1: Sayısal sonuç tahmininde hata ölçümleri.

Şekil 3.1 incelendiğinde hata ölçüm değerleri olarak korelasyon katsayısı MAE, MAPE, RMSE, RAE, ve RRSE değerlendirildiği görülmüştür.

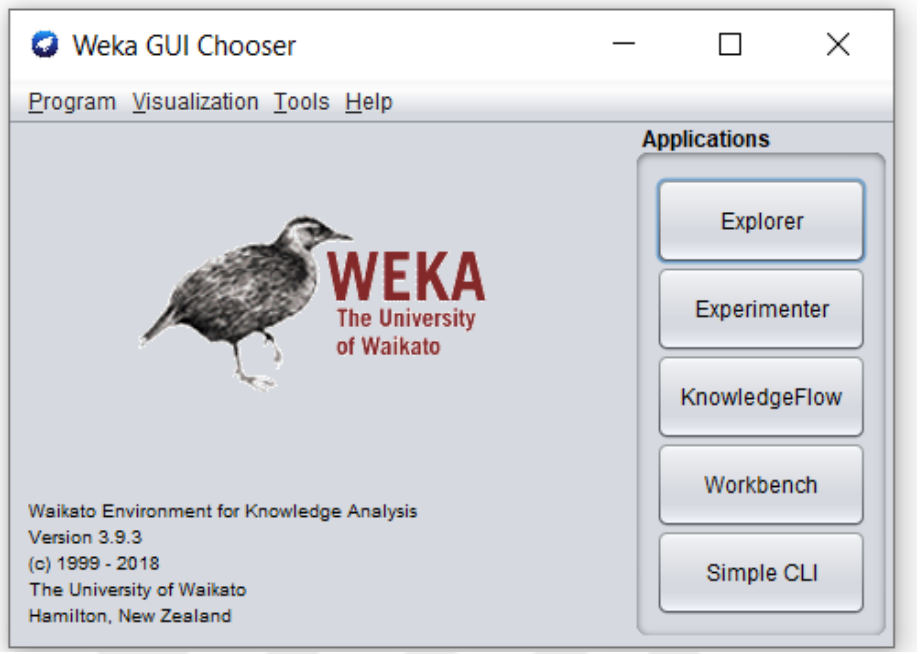
Veri seti dosyasının tahmin edilen sütunu nominal diğer adıyla kategorik ise hata ölçüm değerleri farklı olmaktadır. Şekil 3.2'de nominal bir değerin tahmin edilmesinde hata ölçümlerinin nasıl yapıldığı görülmektedir. Nominal değerlere sahip veri setlerinde veriler üzerinde sınıflandırma işlemi gerçekleştirilirken amaç kaç tanesinin doğru sınıfa kaçtanesinin yanlış sınıfa yerleştirildiğini tahmin etmektir. Şekil 3.2 incelendiğinde hata ölçüm değerleri olarak Şekil 3.1'de açıklanan ölçümlerin yanında, doğru yerleştirme başarısı (correctly classified instance), kappa istatistiği (kappa statistic) ve karışıklık matrisi (confusion matrix) değerlendirilmiştir.

```
Time taken to build model: 0.02 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances          10               71.4286 %
Kappa statistic                          0.3171
Mean absolute error                      0.4399
Root mean squared error                  0.4943
Relative absolute error                 92.3774 %
Root relative squared error            100.1998 %
Total Number of Instances               14

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0,889    0,600    0,727      0,889    0,800      0,337    0,533     0,736     yes
               0,400    0,111    0,667      0,400    0,500      0,337    0,533     0,464     no
Weighted Avg.  0,714    0,425    0,706      0,714    0,693      0,337    0,533     0,639

=== Confusion Matrix ===

 a b   <-- classified as
 8 1 | a = yes
 3 2 | b = no
```

Şekil 3.2: Nominal sonuç tahmininde hata ölçümleri.

Şekil 3.1 ve Şekil 3.2 incelendiğinde kategorik değer tahminlerinde sayısal değer tahminlerine göre daha fazla hata ölçüm değerleri gösterildiği görülmüştür. Bu hata ölçüm değerleri tahmin yapmada kullanılan modelin başarısını izlemeyi sağlamaktadır.

WEKA, arff formatlı dosyalar dışında metin tabanlı csv, dat, libsvm, json ve xrff gibi formatları da desteklemektedir. WEKA programı ilk açıldığında Şekil 3.3'deki gibi kullanıcı ara yüzü açılmaktadır.



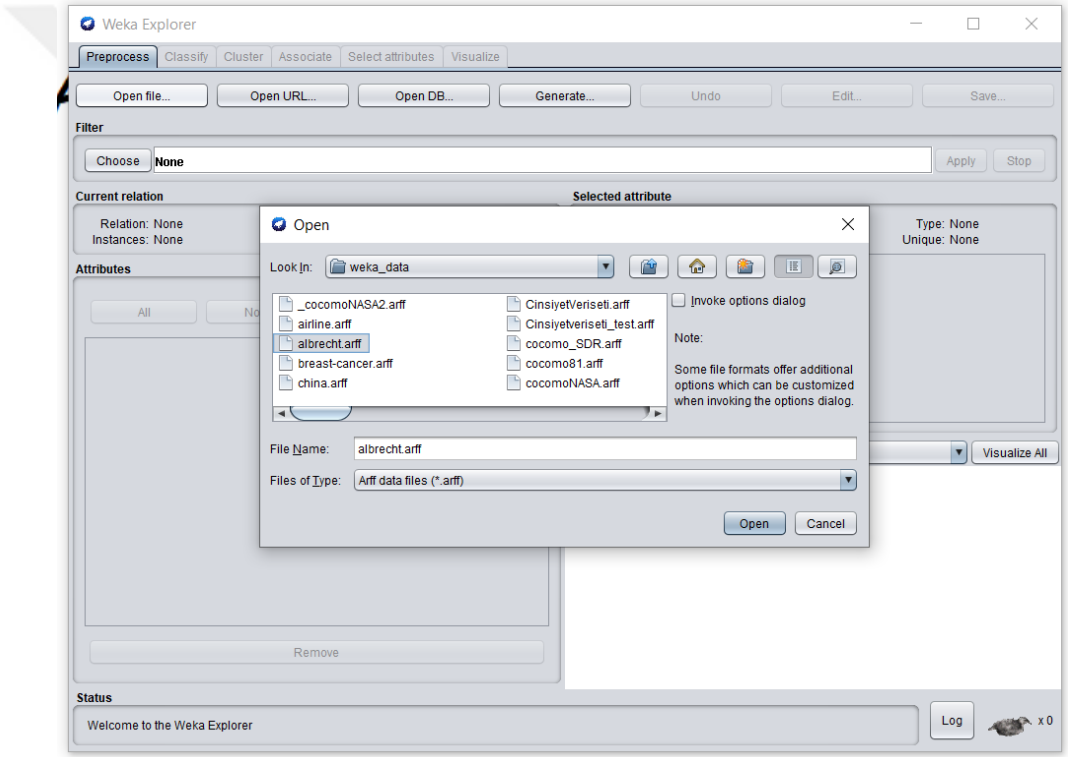
Şekil 3.3: WEKA GUI kullanıcı ara yüzü.

WEKA kullanıcı ara yüzünde çalışma alanına göre 5 faklı seçim alanı bulunmaktadır. Bunlar Şekil 3.3'te görüldüğü üzere Explorer, Experimenter, KnowledgeFlow, Workbench, Simple CLI alanlarıdır.

- *Explorer* düğmesi ile veri seti yükleme, sınıflandırma, kümeleme, ön işleme, öznitelik seçimi gibi işlemler gerçekleştirilmektedir.
- *Experimenter* düğmesi ile sınıflandırma ve regresyon yöntemlerinde en uygun metotların ve en uygun parametre değerlerinin hangisi olduğuna karar verilir.
- *KnowledgeFlow* düğmesi ile büyük boyuttaki verilerin işlemleri gerçekleştirilir. Düğmeye tıklandığında açılan ekran, öğrenme algoritmalarını ve veri kaynaklarını temsil eden kutuları sürüklemeye ve istenilen ayarlar ile birleştirerek bir işlem akışı oluşturmaya yarar böylece veriler aşamalı olarak yüklenir ve işlenir.
- *Workbench* düğmesi, diğer üç düğmenin birleşimini ve kullanıcının yüklediği eklentileri tek bir ekranda sunan bir ana akran sağlar. Kullanıcı tarafından hangi ayarların ve eklentilerin görüneceğini belirlemeye olanak tanıyacak şekilde yapılandırılabilen, veri madenciliği için veri ön işleme, sınıflandırma, kümeleme, regresyon gibi işlemleri gerçekleştiren ve işlemler arasında geçişi sağlayan bir ekrandır.

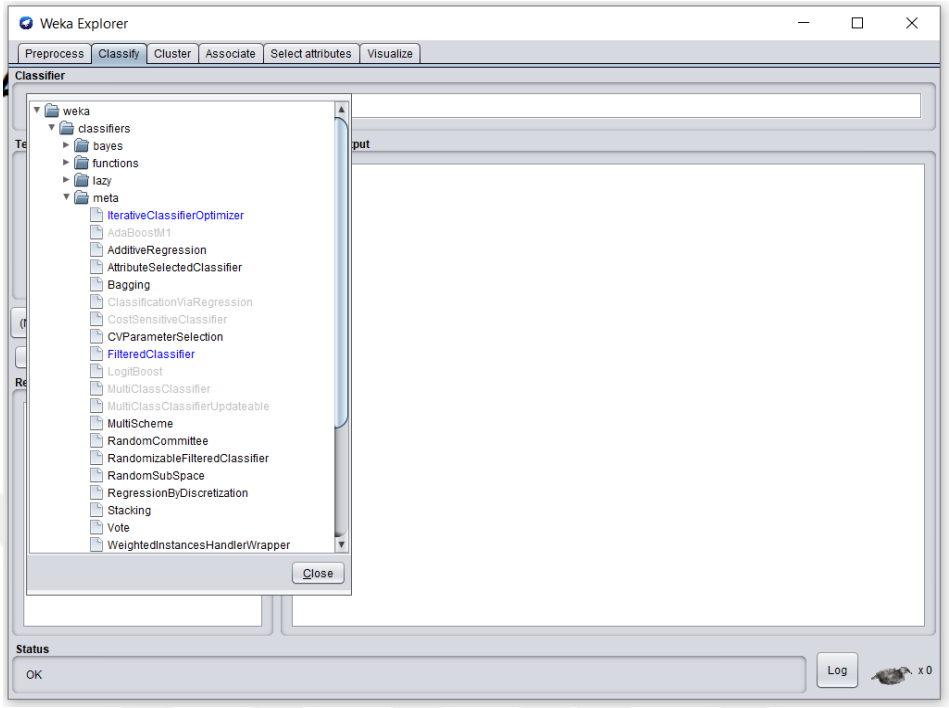
- *Simple CLI* düğmesine tıklandığında console ekranı açılır. Console ekranı ile WEKA'da gerçekleştirilen tüm işlemler metin komutları ile ham formda gerçekleştirilir (Witten, 2011; Etkin, 2017; Aydemir, 2019).

Şekil 3.3'teki ana ekrandan Explorer düğmesine tıklandığında Şekil 3.4'teki ekran açılır. Bütün menülerin aktif olması için Open File düğmesinden ilgili veri setinin WEKA programına yüklenmesi gerekir. Open file düğmesine tıklandığında Şekil 3.4'teki gibi Open penceresi açılır buradan üzerinde işlem yapılacak veri seti seçilir.

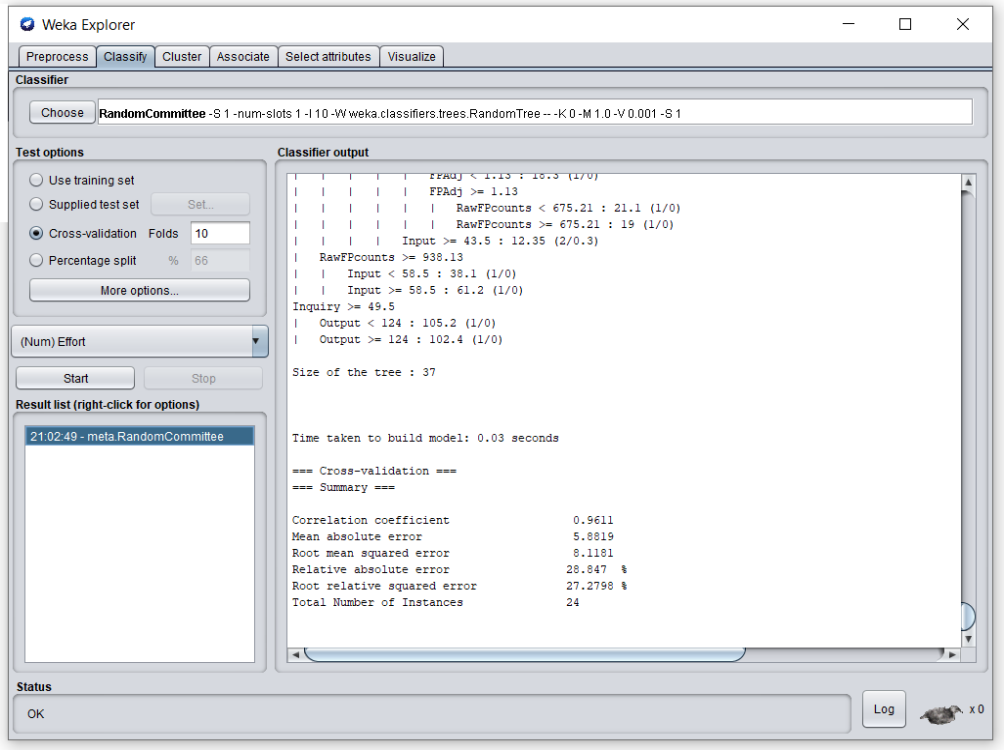


Şekil 3.4: WEKA Explorer penceresi.

Veri seti üzerinde MÖ algoritmalarının çalıştırılması için Şekil 3.5'teki pencereden ilgili algoritma seçilir ve Start düğmesine tıklanır. Varsayılan ayarlar ile veri seti üzerinde seçilen algoritma çalıştırılır ve çıktısı Şekil 3.6'daki gibi görünür.

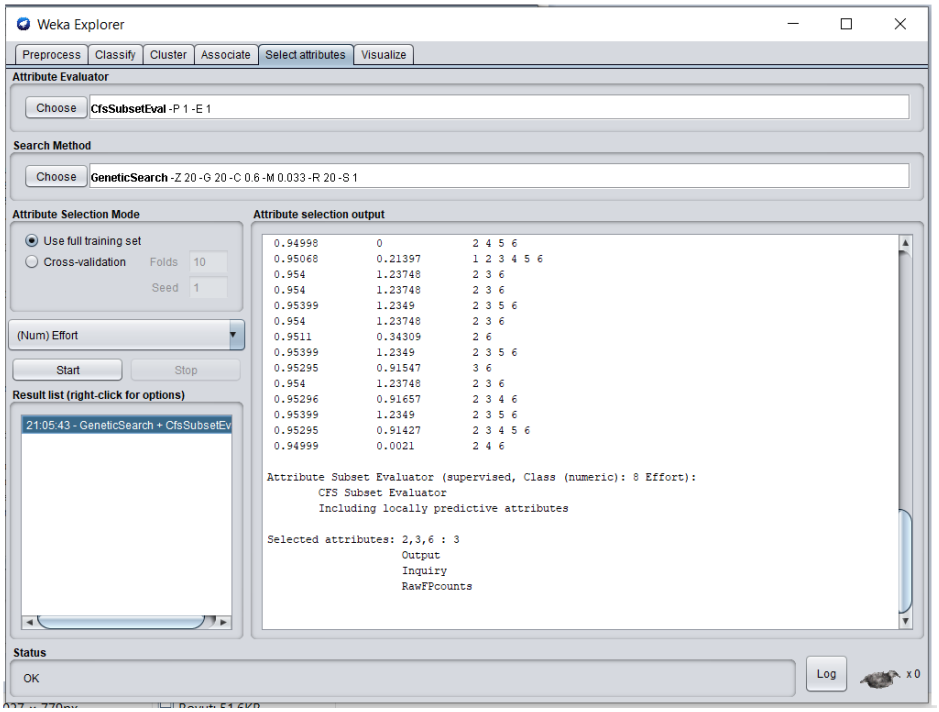


Şekil 3.5: WEKA sınıflandırma penceresi.



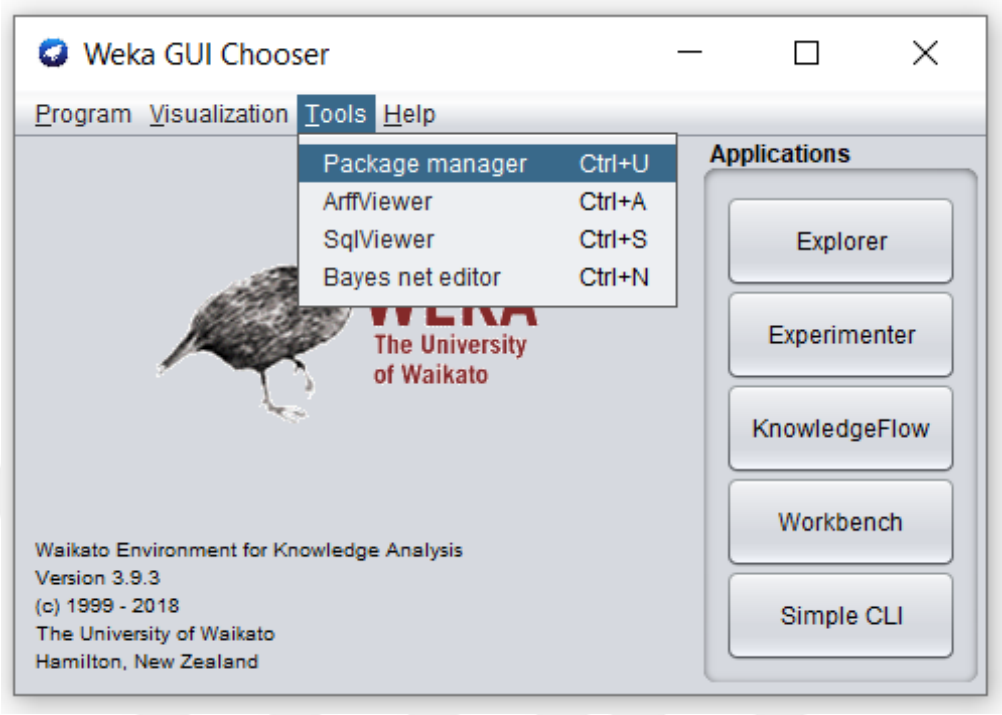
Şekil 3.6: WEKA'da çalıştırılan algoritma çıktısı.

WEKA programı kullanılarak veri setleri üzerinde öznitelik seçimini sağlayan Select attributes menüsüdür. Select attributes menüsüne tıklandığında Şekil 3.7'deki Select attributes penceresi açılır. Select attributes penceresinden öznitelik seçim yöntemi ve arama yöntemi seçilir. Bu tez çalışmasında öznitelik seçim yöntemi olarak CfsSubsetEval (Corelation-based Feature Subset Selection Evaluation – Korelasyon Tabanlı Özellik Seçim Değerlendirici) ve arama yöntemi olarak GeneticSearch ve PSOSearch arama algoritmaları seçilmiştir.

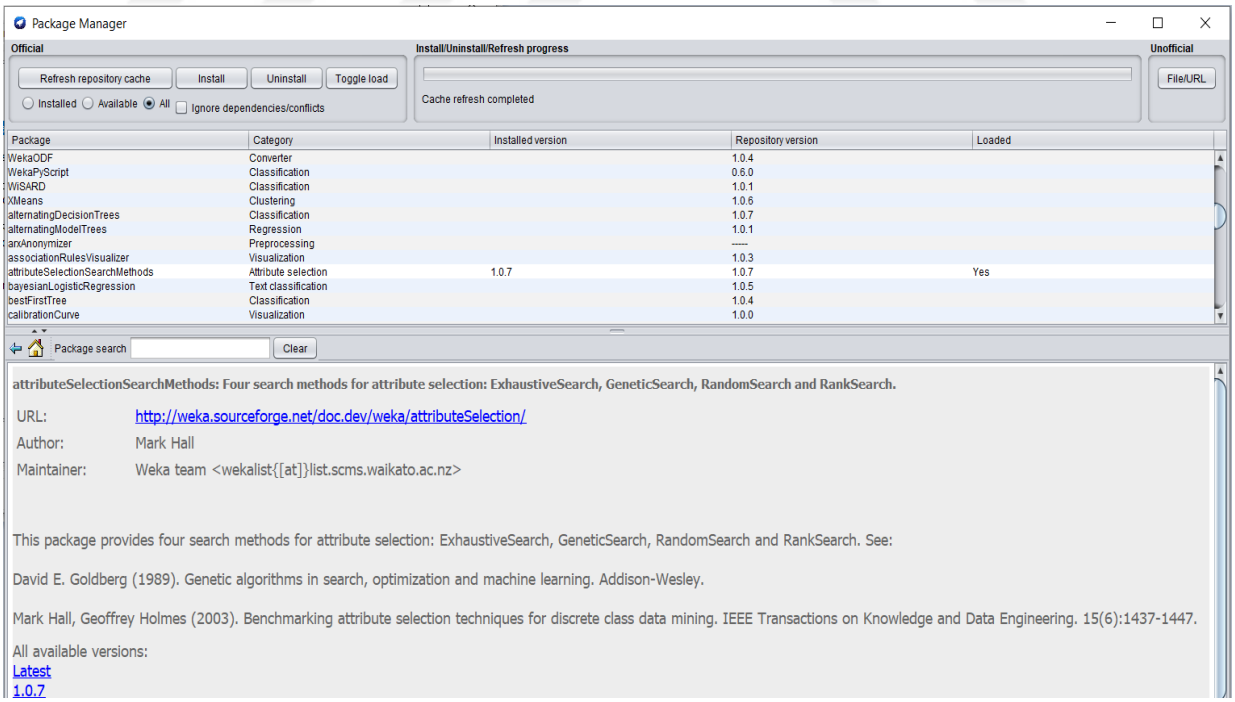


Şekil 3.7: WEKA öznitelik seçimi penceresi.

WEKA programı indirildiğinde uygulamanın içinde bütün eklentiler mevcut değildir. Kurulum esnasında gelmeyen gerekli eklentilerin ayrıca WEKA programına yüklenmesi gerekmektedir. Yükleme işlemi için Şekil 3.8'de olduğu gibi kullanıcı ara yüzünden Tools menüsünün altından Package manager seçeneği seçilir. Şekil 3.9'da görülen pencereden gerekli eklenti seçilir ve load düğesine tıklanır. Bu tez çalışmasında kullanılmak üzere GeneticSearch ve PSOSearch arama algoritmaları WEKA 3.9 sürümüne eklenmiştir. GeneticSeach ve PSOSearch arama algoritmaları yüklendikten sonra veri setleri üzerinde öznitelik seçimi yapılmıştır.



Şekil 3.8: WEKA araçlar menüsü.



Şekil 3.9: WEKA paket yöneticisi.

3.4. YÖNTEMLER

Bu bölümde tez çalışmasında kullanılan algoritmaların ve kullanılan yöntemlerin açıklamaları sunulmuştur.

WEKA programının sınıflandırma (classify) sekmesinde seç (choose) düğmesine tıklanarak MÖ algoritması seçilir. Seçilen algoritmanın parametre değerlerinde değişiklik yapılmak istendiğinde algoritmanın isminin göründüğü satıra tıklanması ve açılan ekranda gerekli değişiklilerin yapılması gerekmektedir. WEKA'da bulunan MÖ algoritmaları;

- Fonksiyonlar (Functions)
- Tembel Sınıflandırıcılar (Lazy)
- Meta
- Çeşitli Kategoriler (Misc)
- Kurallar (Rules)
- Ağaç (Tree)

olarak gruplandırılmıştır.

3.4.1. Fonksiyonlar (Functions)

Bu kategori altında aşağıdaki yöntemler yer almaktadır:

- Gauss Süreçleri (Gaussian Process)
- Doğrusal Regresyon (Linear Regression)
- Basit Doğrusal Regresyon (Simple Linear Regression)
- Ardışık Minimum Optimizasyon Regresyon (Sequential Minimal Optimisation Regression – SMOreg)
- Çok Katmanlı Algılayıcı (Multilayer Perceptron)
- Genetik Programlama (Genetic Programming)
- Parçacık Sürü Optimizasyonu (Particle Swarm Optimization – PSO)

3.4.1.1. Gauss Süreçleri

Gauss Süreçleri, regresyon tabanlı problemlerin çözümünde kullanılmaktadır. Bir Gauss Sürecini modellemek için, önceden çok değişkenli bir Gauss dağılımı gerekmektedir. Gauss Süreci modeli fazla parametreli dağıtılmış rastgele parametrelerin kısıtlı bir koleksiyonunu edinmiş parametrik olmayan çekirdek temelli olasılık modelleridir. Her doğrusal bağlantı aynı dağılmıştır (Yergök ve Acı, 2019).

3.4.1.2. Doğrusal Regresyon

Birden fazla değişken arasındaki nicel bağlantıyı gözlemleyerek değişkenlerden birinin değerini diğer bir değişkenin değerine göre tahmin etmek için kullanılan çözümleme tekniğine Regresyon Analizi denir. Değeri tahmin edilen değişken bağımlı değişken, bağımlı değişkenin değerini tahmin etmek için kullanılan değişkene de bağımsız değişken denmektedir. Bu analiz yönteminde amaç, değişkenler arasındaki bağlantıyı işlevsel bir şekilde yorumlamak ve bu bağlantıyı bir model ile tanımlamaktır. Regresyon analizinde bağımlı ve bağımsız iki değişken varsa Basit Doğrusal Regresyon, bir tane bağımlı değişkene karşı birden fazla bağımsız değişken varsa buna da Çoklu Doğrusal Regresyon denir (Ebren Kara ve Şamlı, 2021).

3.4.1.3. Basit Doğrusal Regresyon

Tek bir açıklayıcı değişkene dayanan doğrusal bir regresyon modelidir. Bir bağımsız değişken ve bir bağımlı değişken olan iki boyutlu örnek noktalarla ilgilidir. Bağımlı değişken değerlerini bağımsız değişkenin bir fonksiyonu olarak tahmin eder ve en küçük karesel hatayı veren öğeyi seçer. Nominal niteliklerle çalışmaz.

3.4.1.4. Ardışık Minimum Optimizasyon Regresyon

Regresyon için SMOreg algoritması, regresyon için Destek Vektör Makinesi'ne (Support Vector Machine – SVM) dayalı SMO algoritmasının geliştirilmiş bir uzantısıdır. SMOreg, doğrusal olmayan tahmin için etkin bir şekilde kullanılmaktadır. SMO'da verimsizlik sorununa neden olan tek bir eşik vardır. SMOreg verimsizlik sorununun üstesinden gelmek için iki eşik kullanır (Singh ve Agrawal, 2013).

3.4.1.5. Çok Katmanlı Algılayıcı

Belirli problemleri çözmek için birlikte çalışan, birbirine bağlı işlem elemanlarından (nöronlar veya düğümler) oluşan bir hesaplama sistemdir (Caudill, 1987). İnsan beyninin nasıl çalıştığından ilham alan, sinir sistemini modelleyerek oluşturulan bir algoritmadır. Çok Katmanlı Algılayıcı, giriş ve çıkış katmanı arasında bir veya daha fazla katman içeren ileri beslemeli bir sinir ağıdır. Temel olarak üç katman vardır: giriş katmanı, gizli katman ve çıkış katmanı. Gizli katman birden fazla olabilir. Her katmandaki her nöron (düğüm), bitişik katmanlardaki her nörona (düğüm) bağlıdır. Eğitim veya test vektörleri giriş katmanından verilir; gizli katmanda işlenir ve çıkış katmanından çıkış alınır (Gupta, 2015; Ebren Kara ve Şamlı, 2021).

3.4.1.6. Genetik Programlama

GA'nın kodlanması ile oluşturulan programlara Genetik Programlama denir. Genetik Programlama sınıflandırma için büyük bir potansiyel sunan; zor problemlerin çözümünde kullanılan evrimsel bir öğrenme tekniğidir. Genetik Programlama temsil biçimi olarak ağaç yapısını kullanır. Ağaç yapısında iç düğümler, işlevler ve operatörleri temsil ederken uç birimler değişkenleri ve sabitleri temsil ederler (Gupta, 2015; Ebren Kara ve Şamlı, 2021).

3.4.1.7. Parçacık Sürü Optimizasyonu

PSO algoritması ilk kez 1995 yılında geliştirilmiştir. Doğrusal olmayan ve çok boyutlu problemi optimize edebilen PSO, popülasyon tabanlı meta sezgisel ve evrimsel bir optimizasyon algoritmasıdır. Algoritma, kuş ve balık sürülerinin barınma, beslenme ve güvenlik için toplu hareketlerinden ilham alınarak geliştirilmiştir. Optimum yer aramak için çok boyutlu uzayda birlikte hareket eden sürüsünün hareketleri ve mesafeleri ayarlanarak davranışları simüle edilmiştir. PSO, GA benzer bir evrimsel hesaplama yöntemidir. Her bireye parçacık (particle), parçacıklardan oluşan topluluğa da sürü (swarm) denmektedir. Sürüler rastgele başlatılır. Sürüdeki her bir parçacığın başlangıçta rastgele oluşturulan konum ve hız bilgisi bulunmaktadır. Bu bilgiler her bir tekrarda kazanç değerlerine göre güncellenmektedir. Optimum kazanç değerine sahip olan parçacıklar sonraki kuşaklara devredilmektedir. Parçacıklardan her biri sahip olduğu konumunu daha önceki tecrübesinden

yararlanarak sürüdeki en iyi konuma göre hesaplamaktadır (Kennedy ve Eberhart, 1995). Sürüdeki her parçacık, tüm parçacıklar arasında yerel en iyi konum olarak bilinen pbest (personal best) ve küresel en iyi konum olarak bilinen gbest (global best) konumlarını korumalıdır. Parçacığın konumunu ve hızını güncellemek için kullanılan formül Denklem 3.7 ve Denklem 3.8'de verilmiştir.

$$V_i^{k+1} = V_i^k + C_1 r_1^k \left(pbest_i^k - X_i^k\right) + C_2 r_2^k \left(gbest^k - X_i^k\right) \quad (3.7)$$

$$X_i^{k+1} = X_i^k + V_i^{k+1} \quad (3.8)$$

Burada ^{k}: iterasyon sayısı, V_i^k: k'inci iterasyondaki i'inci parçacığın hızı, X_i^k: k'inci iterasyondaki i'inci parçacığın konumu, C_1, C_2: hızlandırma katsayıları, r_1, r_2: [0,1) aralığında rastgele üretilen sayılar, $pbest_i^k$: k'inci iterasyondaki i'inci parçacığın yerelde en iyi değeri, $gbest^k$: k'inci iterasyondaki sürünün en iyi değeridir. Şekil 3.10'da PSO akış şeması görünmektedir.

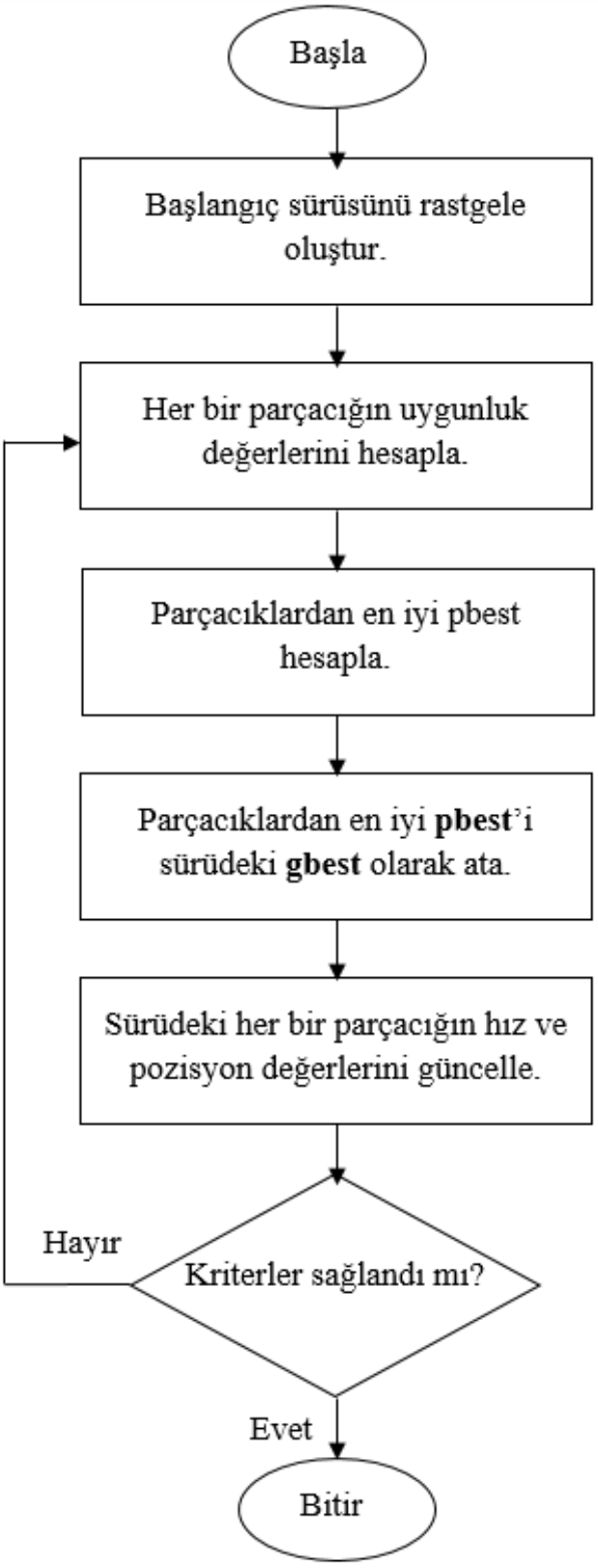
3.4.2. Tembel Sınıflandırıcılar (Lazy Classifier)

Bu kategoride aşağıdaki yöntemler yer almaktadır.

- K-En Yakın Komşu Sınıflandırıcı (K-Nearest Neighbours Classifier)
- Kyıldız (Kstar–K*)
- Yerel Ağırlıklı Öğrenme (LWL-Locally Weighted Learning)

3.4.2.1. K-En Yakın Komşu Sınıflandırıcı

Bu algoritma, sınıflandırılacak olan veriyi mevcut verilere olan yakınlık bağlantısına göre sınıflandırmaktadır. Hem sınıflandırma hem de regresyon problemlerinin çözümünde kullanılan Eğiticili Öğrenme algoritmalarındandır. K sayıda yakınlık komşuluğuna bakılarak mevcut veri setine eklenecek olan yeni verinin, eldeki verilere bakılarak uzaklığının hesaplanmasıdır.



Şekil 3.10: PSO akış şeması.

3.4.2.2. Kyıldız

Entropi tabanlı mesafe algoritması kullanan, verileri temel alan bir sınıflandırıcıdır. Kyıldız Algoritmasının amacı test verisinde bulunan öznitelik bilgisi olmayan bir örneğin, eğitim verisinde eskiden sınıflandırılmış ama ortaya çıkmamış örnekler ile karşılaştırılması esasına göre sınıflandırma gerçekleştirmektir (Aha ve diğ., 1991).

3.4.2.3. Yerel Ağırlıklı Öğrenme

LWL algoritması parametrik bir yöntem değildir ve tahmin, verilerin sadece bir alt kümesini kullanan yerel fonskiyonlar tarafından yapılır. LWL, tüm fonksiyon alanı için global bir model oluşturmak yerine, her bir ilgi noktası için, sorgu noktasının komşu verilerine dayanarak yerel bir model oluşturur. Bu amaçla, her veri noktası, tahmin için veri noktasının etkisini ifade eden bir ağırlık faktörü haline gelir. Genel olarak, mevcut sorgu noktasına yakın komşuluktaki veri noktaları, uzaktaki veri noktalarından daha fazla ağırlık almaktadır (Englert, 2012).

3.4.3. Meta

Bu kategoride aşağıdaki yöntemler yer almaktadır.

- Toplamsal Regresyon (Additive Regression)
- Öznitelik Seçici Sınıflandırıcı (Attribute Selected Classifier)
- Torbalama (Bagging)
- Çapraz Doğrulama Parametre Seçimi (CVParameterSelection)
- Çoklu Şema (Multi Scheme)
- Rastgele Komite (Random Committee)
- Randomize Edilebilir Filtreli Sınıflandırıcı (Randomizable Filtered Classifier)
- Rastgele Alt Boşluk (Random Sub Space)
- Ayrıklaştırma İle Regresyon (Regression By Discretization)
- İstifleme (Stacking)
- Oylama (Vote)
- Ağırlıklı Örnek İşleyici Sarmalayıcı (Weighted Instances Handler Wrappler)

3.4.3.1. Toplamsal Regresyon

Regresyon öğrenmesinin performansını artırır. Çoğunlukla doğrusal olmayan gerçek yaşam etkilerinde Doğrusal Regresyon başarılı olmayan çıktılar oluşturabilmektedir. Toplamsal Regresyon doğrusal olmayan regresyon etkilerini nitelendirmek için kullanılmaktadır.

3.4.3.2. Öznitelik Seçici Sınıflandırıcı

Öznitelik seçici sınıflandırıcı iki adımın bir birleşimidir. Birincisi eğitim ve test verilerinin boyutunu nitelik seçimi yoluyla azaltma, ikincisi sınıflandırmadır.

3.4.3.3. Torbalama

Temel öğrenme seçimine bağlı olarak istatistiksel sınıflandırma ve regresyonda kullanılan MÖ algoritmalarının kararlılığını ve doğruluğunu geliştirmek için tasarlanmış MÖ topluluğu meta algoritmasıdır. Varyans azaltmaya göre sınıflandırma yapar. Algoritma, karar ağacı yöntemleri başta olmak üzere her türlü yöntemle kullanılabilir.

3.4.3.4. Çapraz Doğrulama Parametre Seçimi

Her hangi bir sınıflandırıcıya göre parametre seçiminde çapraz doğrulamayı kullanarak performansı en iyi duruma getirir. Her parametreye, alt ve üst sınırlarını ve istenen artış sayısını içeren bir dize verilir.

3.4.3.5. Çoklu Şema

Yeniden yer değiştirme hatasını kullanarak bir sınıflandırıcı seçer. Performans yüzdesel doğruluk ve regresyon için hata karelerinin ortalaması kullanılarak ölçülür.

3.4.3.6. Rastgele Komite

Temel bir sınıflandırıcılar topluluğunu rastgele olacak şekilde oluşturur ve tahminlerini değerlendirir. Her bir sınıflandırıcı aynı verileri kullanırken farklı rastgele sayı çekirdeği kullanır. Kestirim sonucu her bir temel sınıflandırıcının yaptığı kestirim sonuçlarının

ortalamasıdır. Bu durum yalnızca temel sınıflandırıcı rastgele seçilmişse anlamlıdır; yoksa tüm sınıflandırıcılar aynı olur.

3.4.3.7. Randomize Edilebilir Filtreli Sınıflandırıcı

Rastgele bir filtre ile başlıca sınıflandırıcı olarak IBk ile başlayan Filtrelenmiş Sınıflandırıcının basit bir çeşididir. Ayrıca iki başlıca şekilden an az birinin rastgele ara yüzünün uygulandığını denetleyerek rastgeleleri de uygulayan Filtreli Sınıflandırıcı ile aynı fonksiyonları uygular.

3.4.3.8. Rastgele Alt Boşluk

Sınıflama yapan topluluğu meydana getirmek adına her bir giriş özniteliklerinin rastgele seçilmiş bir alt kümesini kullanan, karar ağacı tabanlı bir sınıflandırıcıdır. Algoritma öznitelik vektörünün alt kümelerinin boyutunu kontrol etmek için bir parametre üretmesinin yanında bir de tekrar sayısını ve kullanılacak rastgele adım sayısını sağlar (Ebren Kara ve Şamlı, 2021).

3.4.3.9. Ayrıklaştırma İle Regresyon

Ayrıklaştırma sayısal verilerin kategorik karşılıklarına dönüştürülmesi işlemine verilen addır. Sıcaklık değişkeninin değerlerini 0-19, 20-39 ve 40-59 gibi aralıklara gruplamak örnek olarak verilebilir. Sınıf özniteliğini eşit genişlikli ayrıklaştırma kullanarak sonlu sayıda gruba ayıran ve sonra bir sınıflandırıcı kullanan bir regresyon şemasıdır. Tahminler, her bir ayrık aralık için ortalama sınıf değerinin ağırlıklı ortalaması olup, aralıklar için öngörülen olasılıklara dayanan ağırlıklardır. Bu yöntem aykırı gözlemlerin, geçersiz veya eksik numerik değerlerin tespitini kolaylaştırır (Aydemir, 2019).

3.4.3.10. İstifleme

Birkaç sınıflandırıcıyı birleştirir. Sınıflandırma veya regresyon yapabilir. Temel sınıflandırıcılar, meta öğrenme ve çapraz doğrulama katlamalarının sayısı belirtilebilir.

3.4.3.11. Oylama

Sınıflandırmaları birleştirmek için farklı olasılık tahmin yöntemleri mevcuttur. Kabul edilen şema, sınıflandırma ve regresyon için olasılık tahminlerinin veya sayısal tahminlerinin ortalamasıdır.

3.4.3.12. Ağırlıklı Örnek İşleyici Sarmalayıcı

Bu yöntem, test verilerinin doğruluğunu kontrol etmek için eğitim verilerini kullanan eğiticili öğrenme algoritmasıdır. Bu yöntemin en büyük avantajı, ağırlıklı eğitim örnekleri için sarmalayıcı yaklaşımını kullanmasıdır. Temel sınıflandırıcı ara yüzü uygulanmadığında ve örnek ağırlıkları olduğunda, bu algoritma ağırlıklarla yeniden örnekleme uygular. Bir seçenek olarak, temel sınıflandırıcı ağırlıkları çalıştırabiliyorsa eğitim verilerini kullanır, ancak ağırlıklarla birlikte yeniden örnekleme yaklaşımlarını da uygulayabilir (Kargar ve diğ., 2021).

3.4.4. Çeşitli Kategoriler

Bu sınıflandırma algoritması, bir sınıflandırıcıyı sarmalar ve kullanılan test verilerinde mevcut olan nitelikler ile modeli eğittiğinde belirlenen nitelikler arasında bir eşleşme gerçekleştirir. Eğitim verilerinde mevcut olmayan fakat test verilerinde mevcut olan nitelikler dâhil edilmez. Test verilerinde olmayan fakat eğitim verilerinde bulunan özellikler eksik değerlere sahip olur. Bununla birlikte, eğitim verilerinde bulunmayan yeni sayısal değerler için eksik değerler kullanır (Aydemir, 2019).

3.4.5. Kurallar

Bu kategoride aşağıdaki yöntemler yer almaktadır:

- Karar Tablosu (Decision Table)
- M5 Kuralları (M5 Rules)
- ZeroR

3.4.5.1. Karar Tablosu

Bu sınıflandırma algoritması, çoğunluk sınıflandırıcısı oluşturmak için basit bir karar tablosu oluşturur ve kullanır. Karar tabloları tahmin için kullanılan sınıflandırma algoritmalarıdır. Bir karar tablosu, daha yüksek seviyeli bir tablodaki her girişin, başka bir tablo oluşturmak için bir çift ek özniteliğin değerlerine bölündüğü, hiyerarşik bir tablodan oluşur (Ebren Kara ve Şamlı, 2021).

3.4.5.2. M5 Kuralları

M5 Kuralları, regresyon problemleri için karar listeleri oluşturmak üzere böl ve yönet yöntemini kullanan bir algoritmadır. Karar listeleri hem sürekli hem de sayısal değişkenlerle çalışabilir. M5 Kuralları, bir model ağacı oluşturmak için M5 algoritmasını kullanır: en iyi yapraktan bir kural yapar daha sonra mevcut kurala bağlı olarak veri kümesinde bulunan öteki örnekler üzerinde çalışır (Omran ve diğ., 2016).

3.4.5.3. ZeroR

ZeroR, hedefe bağlı olan ve tüm tahmin edicileri yok sayan en basit sınıflandırma yöntemidir. ZeroR, basitçe çoğunluk sınıfını tahmin eder. ZeroR'da öngörülebilirlik yeteneği olmamasına rağmen, diğer sınıflandırma yöntemlerinde temel performansı belirlemek için bir kriter olarak çok kullanışlıdır (Nookala ve diğ., 2013).

3.4.6. Ağaç

Bu kategoride aşağıdaki yöntemler yer almaktadır:

- Karar Kütüğü (Decision Stump)
- M5P
- Rep Ağacı
- Rastgele Ağaç (Random Tree)
- Rastgele Orman (Random Forest)

3.4.6.1. Karar Kütüğü

Bir karar kütüğü, yalnızca uç düğümlere (yapraklar) doğrudan bağlı bir iç düğüme (kök) sahip tek seviyeli bir karar ağacından oluşan bir sınıflandırıcıdır. Bir karar kütüğü, tek bir girdinin değerine dayalı bir tahminde bulunabilir ve buna tek kural denir (Chen ve diğ., 2017).

3.4.6.2. M5P

M5 öğrenme algoritmasının yeniden yapılandırılması olan M5P ağaç algoritması regresyon tabanlı problemlerinin çözümünde kullanılan GA türüdür (Mohammed ve diğ., 2020). M5P ağaç algoritması temel olarak iki adım içerir: ağaç büyütme adımı ve ağaç budama adımı.

3.4.6.3. Rep Ağacı

Rep Ağacı, bilgi kazancını, bölme kriteri olarak kullanıp bir regresyon ağacı oluşturan ve bunu azaltılmış hata budaması kullanarak budayan hızlı bir karar ağacı türüdür. Nümerik özniteliklerin değerlerini sadece bir kez sıralar. Eksik değerleri, C4.5'in kesirli örnekleri kullanma yöntemini kullanarak inceler (WEKA, 2021).

3.4.6.4. Rastgele Ağaç

Rastgele Ağaç sınıflandırma algoritması, bir ağaç oluşturmak için her düğümde belirli sayıdaki gelişigüzel seçilen özellikleri dikkate alır. Budamayı gerçekleştirmez. Burada rastgele demek: ağaç kümesindeki her ağacın eşit örnekleme şansına sahip olduğu anlamına gelir. Rastgele ağaçlar verimli bir şekilde oluşturulabilir ve büyük Rastgele Ağaç kümelerinin birleşimi genellikle doğru modeller oluşturur. Rastgele ağaç modelleri, son yıllarda MÖ alanında kapsamlı bir şekilde geliştirilmiştir (Zhao ve Zhang, 2008).

3.4.6.5. Rastgele Orman

Rastgele Orman, rastgele birden çok tekli sınıflandırma ağacı üreterek orman inşa etmek üzere kullanılan bir sınıflandırma algoritmasıdır. Bir girişten yeni bir nesneyi sınıflandırmak için giriş vektörü ormandaki her bir ağaca yerleştirilir. Her bir ağaç kendi sonucunu üretir. Tahmin, topluluğun tahminlerinin toplanmasıyla yapılır. Rastgele Orman genellikle önemli bir performans sergilemektedir (Zhao ve Zhang, 2008).

4. BULGULAR

Bu tez çalışmasında performans analizleri PROMISE veri deposundan temin edilen veri setleri (COCOMO81, COCOMONASA, COCOMONASA2, China, Albrecht, Finnish, Kemerer, Maxwell ve Miyazaki94) üzerinde performans ölçütü olarak korelasyon katsayısı; hata oranlarını değerlendirme ölçütü olarak MAPE, MAE, RMSE, RAE ve RRSE; uygulama platformu olarak da Weka 3.9 ve Weka 3.4.12 sürümleri kullanılmıştır.

Performans analizleri aşağıdaki özelliklere sahip bilgisayarda gerçekleştirilmiştir.

- Intel (R) Core (TM) i7-6700HQ CPU @ 2.60GHz
- 16 GB RAM
- 120 GB SSD ve 1.80 TB Harddisk
- 64 bit işletim sistemi, x64 tabanlı işlemci
- Windows 10 Home Single Language

4.1. WEKA SİMÜLASYON SONUÇLARI

Tez çalışmasının bu kısmında yazılım maliyet tahmini için COCOMO81, COCOMONASA, COCOMONASA2 veri setleri kullanılmıştır. Veri setleri 10 kat çapraz doğrulama tekniği kullanılarak rastgele eğitim ve test verilerine bölünmüştür. Oluşturulan model korelasyon katsayısı, hata oranı MAE, RMSE, RAE ve RRSE'ye göre değerlendirilmiştir. WEKA ortamında bulunan MÖ algoritmaları kullanılarak yazılım projelerinin maliyet tahmini iki bölümde gerçekleştirilmiştir. İlk bölümde; WEKA ortamı algoritmalarının varsayılan ayarları tercih edilmiştir. Meta grubunda bulunan algoritmaların, LWL ve Input Mapped Classifier algoritmalarının özellikler penceresinden temel sınıflandırıcı olarak Random Forest algoritması seçilmiştir. İkinci bölümde; WEKA programında bulunan bazı algoritmalar (Meta grubunda bulunan algoritmalar, LWL ve Input Mapped Classifier algoritmaları) mevcut parametrelerine ek olarak temel bir sınıflandırıcı ve onun parametrelerini alan algoritmalardır. Söz konusu algoritmalar için temel sınıflandırıcı belirlerken olabilecek bütün algoritmalar tek tek denenmiştir ve denemeler bütün veri setleri için tekrar edilmiştir.

Tablo 4.1'de, Yazılım maliyet tahmini için COCOMO81'e uygulanan MÖ algoritmalarının performans ölçümleri verilmiştir. Tablo 4.2'de, Yazılım maliyet tahmini için COCOMONASA'ya uygulanan MÖ algoritmalarının performans ölçümleri verilmiştir. Tablo 4.3'te Yazılım maliyet tahmini için COCOMONASA2'ye uygulanan MÖ algoritmalarının performans ölçümleri verilmiştir.

Tablo 4.1: COCOMO81'de tahmin algoritmalarının performans ölçümleri.

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

Tablo 4.1 incelendiğinde, 471,6203 MAE, 1169,6529 RMSE, %51,9767 RAE, %63,7149 RRSE hata oranları ve 0,8095 korelasyon katsayısı ile en iyi tahmin sonucunu Additive

Regression algoritması gerçekleştirmiştir. REP Tree tahmin algoritması 0,0902 korelasyon katsayısı ve 787,8688 MAE, 1904,8964 RMSE, %86,8301 RAE, %103 RRSE, hata payı ile en kötü performansı sergilemiştir.

Tablo 4.2: COCOMONASA'da tahmin algoritmalarının performans ölçümleri.

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

Tablo 4.2 incelendiğinde, 150,9841 MAE, 252,8864 RMSE, %35,0178 RAE, %37,9721 RRSE hata oranları ve 0,922 korelasyon katsayısı ile en iyi tahmin sonucunu M5P algoritması gerçekleştirmiştir. Decision Table algoritması 0,4577 korelasyon katsayısı ve 261,1296 MAE, 609,2296 RMSE, %60,5639 RAE, %91,4787 RRSE hata payı ile en kötü performansı sergilemiştir.

Tablo 4.3: COCOMONASA2'de tahmin algoritmalarının performans ölçümleri.

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

Tablo 4.3 incelendiğinde, 332,7218 MAE, 652,8788 RMSE, %51,5118 RAE, %57,1464 RRSE hata oranları ve 0,8183 korelasyon katsayısı ile en iyi tahmin sonucunu LWL algoritması gerçekleştirmiştir. Decision Table algoritması 0,2525 korelasyon katsayısı ve 564,7407 MAE, 1186,1157 RMSE, 87,4329 RAE, 103,8206 RRSE hata payı ile en kötü performansı sergilemiştir. WEKA programında, Meta grubundaki bütün algoritmalar, Lazy grubundaki LWL algoritması ve Rules grubundaki Input Mapped Classifier algoritması parametre olarak bir sınıflandırma algoritması almaktadır. Bu algoritmaların özellikler penceresinden sınıflandırma özelliği değiştirilerek daha iyi tahmin sonuçları elde edilebilmektedir. Yazılım projelerinin maliyet tahmini için kullanılan MÖ algoritmalarının parametre değerleri

değiştirilerek olabilecek bütün olasılıklar denenmiştir. COCOMO81, COCOMONASA ve COCOMONASA2 veri setleri üzerinde gerçekleştirilen testlerin en iyi tahmin sonuçları korelasyon katsayısı ve RAE hata payı ile Tablo 4.4, Tablo 4.5 ve Tablo 4.6'da belirtilmiştir.

Tablo 4.4: COCOMO81'de algoritmaların en iyi tahmin sonuçları.

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

Tablo 4.4 incelendiğinde tahmin algoritmasına sınıflandırma parametresi olarak Random Committee algoritmasının verilmesi ile en iyi tahmin sonucunun elde edildiği görülmüştür. Random Committee, CVParemeter Selection ve Input Maped Classifier algoritmaları 0,8764 korelasyon katsayısı ve %48,5789 RAE hata payı ile en iyi tahmin sonucunu vermiştir. Bu durum şu şekilde elde edilmiştir: yazılım maliyet tahmini için seçilen algoritmanın özellikler penceresinden classifier özelliği için olabilecek bütün algoritmalar parametre olarak denenmiştir. COCOMO81 veri seti üzerinde en iyi tahmin sonucunun tahmin algoritması olarak Random Committee, classifier özelliğinin de Random Committee seçilmesi ile elde edildiği belirlenmiştir.

Tablo 4.5: COCOMONASA'da algoritmaların en iyi tahmin sonuçları.

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

Tablo 4.5 incelendiğinde tahmin algoritmasına sınıflandırma parametresi olarak M5P algoritmasının verilmesi ile en iyi tahmin sonucunun elde edildiği görülmüştür. Additive Regression algoritması 0,9371 korelasyon katsayısı ve %31,7645 RAE hata payı ile en iyi tahmin sonucunu vermiştir. En iyi tahmin sonucunun elde edilebilmesi için seçilen algoritmanın özellikler penceresinden classifier özelliği için olabilecek bütün algoritmalar parametre olarak denenmiştir. COCOMONASA veri seti üzerinde en iyi tahmin sonucunun tahmin algoritması olarak Additive Regression, classifier özelliğinin de M5P seçilmesi ile elde edildiği belirlenmiştir.

Tablo 4.6: COCOMONASA2 'de algoritmaların en iyi tahmin sonuçları.

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

Tablo 4.6 incelendiğinde tahmin algoritmasına sınıflandırma parametresi olarak Random Committee algoritmasının verilmesi ile en iyi tahmin sonucunun elde edildiği görülmüştür. LWL algoritması 0,8309 korelasyon katsayısı ve %50,4804 RAE hata payı ile en iyi tahmin sonucunu vermiştir. En iyi tahmin sonucunun elde edilebilmesi için seçilen algoritmanın özellikler penceresinden classifier özelliği için olabilecek bütün algoritmalar parametre olarak denenmiştir. COCOMONASA2 veri seti üzerinde en iyi tahmin sonucunun tahmin algoritması olarak LWL, classifier özelliğinin de Random Committee seçilmesi ile elde edildiği belirlenmiştir.

4.2. WEKA ÖZNİTELİK SEÇİM ALGORİTMALARI İLE PERFORMANS DEĞERLENDİRMESİ

Çalışmanın bu kısmında PROMISE veri deposundan alınan, Albrecht, Finnish, Kemerer, Maxwell, China, COCOMONASA ve Miyazaki94 veri setleri üzerinde öznitelik seçimi yapılarak yazılım maliyet tahmini yapılmıştır. Öznitelik seçimi, verilerin ön işlemden geçirildiği sırada sınıflandırıcının doğruluğunu ve performansını artırmak amacıyla ilgisiz ve gereksiz özniteliklerin atılması ve verilerin gürültüden temizlenmesi işlemidir. Öznitelik seçimi, mevcut özniteliklerin bir alt kümesini oluştururken veriler üzerinde herhangi bir dönüşüm gerçekleştirmez (Güven Aydın, 2021). Literatürde öznitelik seçimi için farklı yöntemler geliştirilmiştir. Bu yöntemlerin bazıları tezin önceki bölümlerinde açıklanmıştır. Bu çalışmada öznitelik seçim yöntemi olarak WEKA programı içerisinde bulunan CfsSubsetEval, arama metodu olarak GeneticSearch ve PSOSearch tercih edilmiştir. Oluşturulan bu modelde amaç, yazılım maliyet tahmini yapılırken kullanılan hazır veri setleri üzerinde öznitelik seçiminin maliyet tahminine olan etkisinin araştırılmasıdır.

4.2.1. CfsSubsetEval Öznitelik Seçim Algoritması

En etkili özniteliklerden oluşan öznitelik alt kümelerinin oluşturulması için WEKA ortamında bulunan CfsSubsetEval kullanılmıştır. CfsSubsetEval, öznitelik alt kümelerini korelasyona dayalı sezgisel değerlendirme işlevine göre sıralayan basit bir filtre algoritmasıdır. En iyi öznitelik alt kümesini korelasyon yardımı ile bulur. Bu algoritma sınıfla yüksek düzeyde ilişkili olan ve birbirleriyle ilişkisiz öznitelikler içeren alt kümeleri değerlendirir. Alakasız olan öznitelikler sınıfla düşük korelasyona sahip olacağından göz ardı edilir. Algoritma sınıfla

yüksek korelasyonlu, kendi aralarında düşük korelasyonlu öznitelikleri seçer. Bunu da arama algoritmalarına gönderdiği metrik ile yapar. CfsSubsetEval bir arama yöntemi değildir, bunun yerine arama algoritmalarına öznitelik alt kümesinin etkinliğini değerlendirmek için bir metrik önerir. Algoritmanın temelinde çıktı sınıfıyla yüksek oranda ilişkili ancak birbiriyle ilişkisiz özelliklere sahip iyi bir öznitelik alt kümesi oluşturmak yatmaktadır (Hall, 1999; Ebren Kara ve Şamlı, 2021). CfsSubsetEval, herhangi bir açgözlü veya meta-sezgisel arama yaklaşımıyla kullanılabilir. Bu çalışmada CfsSubsetEval, GeneticSearch ve PSOSearch algoritmaları ile kullanılmıştır.

4.2.2. Albrecht, Finnish, Kemerer, Maxwell ve Miyazaki94 Veri Setleri Simülasyon Sonuçları

Burada WEKA ortamında, PROMISE veri deposundan temin edilen Albrecht, Finnish, Kemerer, Maxwell ve Miyazaki94 veri setleri kullanılmıştır. Veri setlerine WEKA'da bulunan bazı algoritmalar ve Genetik Programlama uygulanmıştır. WEKA'nın en son sürümlerinde Genetik Programlama mevcut olmadığından Genetik Programlama'nın dâhil edilebilmesi için WEKA'nın 3.4.12 sürümü kullanılmıştır. WEKA ortamında bulunan algoritmalar varsayılan ayarlar ile veri setleri üzerinde iki şekilde çalıştırılmıştır. İlkinde, herhangi bir öznitelik seçimi yapılmadan ham veri seti üzerinden algoritmalar 10 kat çapraz doğrulama ile çalıştırılmıştır. İkincisinde, her veri seti üzerinde ilk önce GA kullanılarak öznitelik seçimi gerçekleştirilmiştir. Veri setlerine uygulanan öznitelik seçiminden sonra bazı öznitelikler (bulgular bölümünde seçilen öznitelikler verilmiştir.) veri setlerinden kaldırılmıştır. Genetik Programlama her çalıştırıldığında farklı bir sonuç vermektedir bu yüzden algoritmanın performans değerleri incelenirken farklı bir yöntem uygulanmıştır. GA olasılıksal, stokastik, küresel arama algoritmasıdır; bu nedenle her popülasyonun her bireyinin, her yürütme sırasında arama alanı boyunca farklı bir yörünge gerçekleştirmesi ve popülasyonun çoklu (alt) optimal çözümlere yaklaşması beklenmektedir. Bir GA'yı tekrar tekrar yürüterek bir dizi optimal çözüm toplanabilir. Algoritmanın performans değerlendirmesinde ortalama değeri hesaplamak yanlıştır çünkü benzer uygunluk değerine sahip iki alt optimal çözüm tamamen farklı bir yapıya sahip olabilir, böylece ortalama değer arama uzayının uygun olmayan bir bölgesine karşılık gelebilir (Ebren Kara ve Şamlı, 2021). Bundan dolayı Genetik Programlama 15 defa çalıştırılarak uygunluk değerine göre en uygun sonuç seçilmiştir. Albrecht veri setine

WEKA'nın select attributes menüsü altındaki CfsSubsetEval ve GeneticSearch uygulanarak öznitelik sayısı 8'den 3'e düşürülmüştür. GA uygulanarak Output, Inquiry, RawFPcounts öznitelikleri seçilmiştir. Bağımlı öznitelik olan Effort da eklenerek öznitelik sayısı 4 olarak belirlenmiştir. Algoritmalar veri seti üzerinde öznitelik seçimi yapılmadan ve seçim yapıldıktan sonra çalıştırılmıştır ve performans değerleri Tablo 4.7'de gösterilmiştir.

Tablo 4.7: Albrecht veri setinde öznitelik seçimi.

Albrecht veri seti						
ALGORİTMALAR	**Öznitelik seçimi yapılmadan önce**			**GA ile öznitelik seçimi (öznitelik sayısı 8'den 3'e düşürüldüğünde)**		
FONKSİYONLAR	**Korelasyon katsayısı**	**MAE**	**RAE (%)**	**Korelasyon katsayısı**	**MAE**	**RAE (%)**
Gaussian Processes	0,6529	15,8737	77,8508	0,8847	21,7193	106,52
Linear Regression	0,9062	8,9952	44,116	0,9286	8,2725	40,5716
Multilayer Perceptron	0,7543	12,0558	59,1265	0,9351	7,2164	35,3922
Simple Linear Regression	0,8598	9,9098	48,6014	0,8598	9,9098	48,6014
SMOreg	0,8139	11,7115	57,4375	0,8986	10,2052	50,0501
LAZY						
IBK	0,9406	6,1917	30,3663	0,9429	6,5333	32,042
KStar	0,8518	8,5702	42,0317	0,934	6,9165	33,9211
LWL	0,8741	9,5116	46,6483	0,9003	8,0443	39,4522
META						
Additive Regression	0,8653	9,8845	48,4775	0,9162	7,1197	34,9177
Bagging	0,8263	13,2655	65,0593	0,9082	11,1858	54,8594
Random Committee	0,9611	5,8819	28,847	0,9474	5,9019	28,9454
Randomizable Fitered Classifer	0,9416	6,5083	31,9194	0,9495	6,525	32,0011
RandomSubSpace	0,4334	14,3141	70,2019	0,6672	12,7241	62,4039
RegresiyonByDiscretization	0,4337	16,1353	79,1337	0,8325	10,7695	52,818
RULES						
Decision Table	0,6937	9,9449	48,7735	0,6894	9,9155	48,6296
M5Rules	0,8921	8,0587	39,5231	0,8751	8,3984	41,189
TREE						
Desicion Stump	0,5145	13,785	67,6072	0,8877	8,9221	43,7575
M5P	0,8325	8,4905	41,6406	0,9422	7,0326	34,4907
Random Forest	0,9406	7,6828	37,6795	0,9585	6,3113	30,953
Random Tree	0,471	13,4589	66,0076	0,8631	9,2799	45,512
REP Tree	0,5834	14,9504	73,3225	0,5425	14,7469	72,3244
Genetic Programming	0,8595	11,378	15,7919	0,9035	15,277	74,927

En iyi performansı, öznitelik seçimi yapılmadan önce Random Committee, seçim yapıldıktan sonra Random Forest göstermiştir. Öznitelik seçimi yapılmadan önce en kötü performansı RegresiyonByDiscretization gösterirken öznitelik seçimi yapıldıktan sonra en kötü performansı REP Tree algoritması göstermiştir. Genel olarak algoritmaların performans değerleri incelendiğinde öznitelik seçiminin yapılması bütün algoritmalarda olumlu bir etki yaratmıştır.

Tablo 4.8: Finnish veri setinde öznitelik seçimi.

Finnish veri seti						
ALGORİTMALAR	**Öznitelik seçimi yapılmadan önce**			**GA ile öznitelik seçimi (öznitelik sayısı 9'dan 4'e düşürüldüğünde)**		
FONKSİYONLAR	**Korelasyon katsayısı**	**MAE**	**RAE (%)**	**Korelasyon katsayısı**	**MAE**	**RAE (%)**
Gaussian Processes	0,8597	0,5691	55,6165	0,8443	0,5886	57,5203
Linear Regression	0,9607	0,254	24,8268	0,9607	0,254	24,8268
Multilayer Perceptron	0,9575	0,2297	22,4464	0,9853	0,1376	13,4468
Simple Linear Regression	0,8811	0,4586	44,8219	0,8811	0,4586	44,8219
SMOreg	0,962	0,2341	22,8759	0,9702	0,2213	21,6292
LAZY						
IBK	0,7697	0,539	52,6711	0,934	0,3142	30,7074
KStar	0,9889	0,1344	13,1344	0,9923	0,1127	11,0156
LWL	0,8808	0,4707	46,0022	0,8934	0,4379	42,7989
META						
Additive Regression	0,9467	0,3197	31,2479	0,9507	0,3009	29,4097
Bagging	0,9801	0,1747	17,074	0,9857	0,1532	14,967
Random Committee	0,9797	0,1743	17,0383	0,9922	0,1072	10,4756
Randomizable Fitered Classifer	0,8736	0,4171	40,7633	0,9396	0,311	30,3962
RandomSubSpace	0,9239	0,3752	36,6703	0,9457	0,2945	28,7796
RegresiyonByDiscretization	0,9748	0,216	21,1073	0,985	0,1899	18,5577
RULES						
Decision Table	0,8788	0,3802	37,1524	0,8788	0,3802	37,1524
M5Rules	0,9876	0,1414	13,8139	0,9833	0,1475	14,4124
TREE						
Desicion Stump	0,8618	0,513	50,1362	0,8618	0,513	50,1362
M5P	0,9692	0,2146	20,9732	0,9715	0,2158	21,0941
Random Forest	0,9818	0,1649	16,1147	0,9942	0,0976	9,5354
Random Tree	0,9029	0,3654	35,7136	0,9922	0,1072	10,4756
REP Tree	0,9752	0,195	19,0542	0,9829	0,1779	17,3849
Genetic Programming	0,2286	1,5542	151,8817	0,3959	1,322	129,1927

Finnish veri setine, WEKA'nın select attributes menüsü altındaki CfsSubsetEval ve GeneticSearch uygulandığında veri setinde bulunan 9 öznitelikten 4 tanesi seçilmiştir. Bunlar dev.eff.hrs, FP, prod, lnsize öznitelikleridir. Öznitelik alt kümesine, bağımlı öznitelik olan lneff özniteliği de eklendiğinde öznitelik sayısı 5'e çıkmıştır. WEKA ortamında bulunan algoritmalar, Finnish veri setine öznitelik seçimi uygulanmadan önce ve öznitelik seçimi uygulandıktan sonra çalıştırılmıştır. Algoritmaların bulduğu tahmin sonuçları Tablo 4.8'de belirtilmiştir. Tablo 4.8 incelendiğinde, öznitelik seçiminden önce yapılan en iyi tahmin sonucunu KStar algoritması bulurken öznitelik seçiminden sonra en iyi tahmin sonucunu Random Forest algoritması bulmuştur. Veri setine uygulanan algoritmalar arasından Genetic Programming algoritması kötü bir performans sergilemiştir.

Tablo 4.9: Kemerer veri setinde öznitelik seçimi.

Kemerer veri seti						
ALGORİTMALAR	**Öznitelik seçimi yapılmadan önce**			**GA ile öznitelik seçimi (öznitelik sayısı 8'den 4'e düşürüldüğünde)**		
FONKSİYONLAR	**Korelasyon katsayısı**	**MAE**	**RAE (%)**	**Korelasyon katsayısı**	**MAE**	**RAE (%)**
Gaussian Processes	0,2401	173,4761	107,7937	0,2705	161,1668	100,145
Linear Regression	0,3692	173,2407	107,6474	0,3425	190,2161	118,1955
Multilayer Perceptron	0,3511	129,4589	80,4425	0,3277	150,4623	93,4935
Simple Linear Regression	0,3516	173,231	107,6414	0,3516	173,231	107,6414
SMOreg	0,5737	114,3301	71,0419	0,6946	96,4073	59,9051
LAZY						
IBK	0,4665	142,054	88,2688	0,336	160,028	99,4374
KStar	0,5589	134,6747	83,6835	0,6219	124,3199	77,2492
LWL	0,6673	131,4466	81,6776	0,2476	170,0038	105,6361
META						
Additive Regression	0,4744	146,6895	91,1492	0,4048	139,608	86,7489
Bagging	0,1277	185,4463	115,2317	0,117	180,8072	112,3491
Random Committee	0,3253	142,7104	88,6767	0,1869	169,0092	105,0181
Randomizable Fitered Classifer	0,689	121,1073	75,2531	0,4886	138,4607	86,036
RandomSubSpace	0,0247	144,782	89,9639	-0,0377	148,1961	92,0854
RegresiyonByDiscretization	0,2989	177,4283	110,2495	0,2962	177,4659	110,2729
RULES						
Decision Table	0,102	144,9955	90,0966	0,3026	176,223	109,5005
M5Rules	0,3244	182,0525	113,1229	0,3385	188,2263	116,9591
TREE						
Desicion Stump	0,7835	138,6838	86,1746	0,2151	177,1265	110,062
M5P	0,3291	176,3236	109,5631	0,3385	188,2263	116,9591
Random Forest	0,3532	129,0567	80,1926	0,2925	143,9352	89,4377
Random Tree	-0,0271	250,9131	155,9111	0,329	163,9313	101,8628
REP Tree	-0,3027	195,5944	121,5375	-0,3027	195,5944	121,5375
Genetic Programming	0,5299	207,0738	128,6705	0,6505	140,4516	87,2731

Kemerer veri setine, WEKA'nın select attributes menüsü altındaki CfsSubsetEval ve GeneticSearch uygulandığında veri setinde bulunan 8 öznitelikten 4 tanesi seçilmiştir. Bunlar ID, Language, KSLOC, AdjFP öznitelikleridir. Öznitelik alt kümesine, bağımlı öznitelik olan EffortMM özniteliği de eklendiğinde öznitelik sayısı 5'e çıkmıştır. WEKA ortamında bulunan algoritmalar, Kemerer veri setine öznitelik seçimi uygulanmadan ve öznitelik seçimi uygulandıktan sonra çalıştırılmıştır. Algoritmaların bulduğu tahmin sonuçları Tablo 4.9'da belirtilmiştir. Tablo 4.9 incelendiğinde, öznitelik seçiminden önce en iyi tahmin sonucunu 0,7835 korelasyon katsayısı ve %86,1746 RAE hata payı ile Desicion Stump algoritması bulmuştur. Öznitelik seçiminden sonra en iyi tahmin sonucunu SMOreg algoritması bulmuştur. Veri setine uygulanan algoritmalar arasından Random Tree algoritması öznitelik seçimi

yapılmadan önce en kötü tahmin sonucunu bulurken öznitelik seçimi yapıldıktan sonra en kötü tahmin sonucunu REP Tree algoritması bulmuştur.

Tablo 4.10: Maxwell veri setinde öznitelik seçimi.

Maxwell veri seti						
ALGORİTMALAR	**Öznitelik seçimi yapılmadan önce**			**GA ile öznitelik seçimi (öznitelik sayısı 27'den 19'a düşürüldüğünde)**		
FONKSİYONLAR	**Korelasyon katsayısı**	**MAE**	**RAE (%)**	**Korelasyon katsayısı**	**MAE**	**RAE (%)**
Gaussian Processes	0,783	3925,0493	62,4734	0,787	3933,8702	62,6138
Linear Regression	0,8085	4157,5897	66,1746	0,8544	3395,0666	54,0379
Multilayer Perceptron	0,7641	4764,3788	75,8327	0,816	4146,3094	65,9951
Simple Linear Regression	0,8023	3952,8465	62,9158	0,8023	3952,8465	62,9158
SMOreg	0,8191	3812,9653	60,6894	0,818	3522,3771	56,0642
LAZY						
IBK	0,463	5517,129	87,8139	0,7593	4494,6774	71,5399
KStar	0,7336	4618,2302	73,5065	0,8596	4078,3244	64,913
LWL	0,6089	5340,3849	85,0007	0,5866	5163,7796	82,1897
META						
Additive Regression	0,6875	5233,4509	83,2987	0,6882	5212,0729	82,9584
Bagging	0,7711	3949,1671	62,8573	0,7704	3898,5336	62,0514
Random Committee	0,7875	3991,4994	63,531	0,6924	4238,3461	67,46
Randomizable Fitered Classifer	0,7402	4411,4194	70,2147	0,8235	4289,5323	68,2747
RandomSubSpace	0,6696	4734,9648	75,3645	0,6474	4700,8192	74,821
RegresiyonByDiscretization	0,5893	5419,5948	86,2615	0,6092	5217,0333	83,0374
RULES						
Decision Table	0,3139	5355,5581	85,2422	0,417	5060,9465	80,553
M5Rules	0,7497	3853,4535	61,3338	0,6528	4334,3177	68,9875
TREE						
Desicion Stump	0,5893	5211,5585	82,9502	0,5893	5211,5585	82,9502
M5P	0,8175	3718,2692	59,1822	0,8092	3685,2814	58,6571
Random Forest	0,7612	3998,2174	63,638	0,7621	3827,5684	60,9218
Random Tree	0,569	5686,9672	90,5171	0,4398	5223,2222	83,1359
REP Tree	0,5801	4801,8161	76,4285	0,5781	4857,6179	77,3167
Genetic Programming	0,618	7700,821	122,5708	0,4508	8906,4934	141,761

Maxwell veri setine WEKA'nın select attributes menüsü altındaki CfsSubsetEval ve GeneticSearch uygulandığında veri setinde bulunan 27 öznitelikten 19 tanesi seçilmiştir. Bunlar Syear, App, Har, Dba, Source, T01, T02, T04, T06, T07, T08, T09, T10, T11, T13, T14, Duration, Size, Time özniteliklerdir. Öznitelik alt kümesine, bağımlı öznitelik olan Effort özniteliği de eklendiğinde öznitelik sayısı 20'ye çıkmıştır. WEKA ortamında bulunan algoritmalar, Maxwell veri setine öznitelik seçimi uygulanmadan önce ve öznitelik seçimi uygulandıktan sonra çalıştırılmıştır. Algoritmaların bulduğu tahmin sonuçları Tablo 4.10'da belirtilmiştir. Tablo 4.10 incelendiğinde, öznitelik seçiminden önce yapılan en iyi tahmin sonucunu SMOreg algoritması bulurken, öznitelik seçiminden sonra en iyi tahmin sonucunu

KStar algoritması bulmuştur. Veri setine uygulanan algoritmalar arasından Decision Table algoritması en kötü performansı sergilemiştir.

Tablo 4.11: Miyazaki94 veri setinde öznitelik seçimi.

Miyazaki94 veri seti						
ALGORİTMALAR	**Öznitelik seçimi yapılmadan önce**			**GA ile öznitelik seçimi (öznitelik sayısı 9'dan 3'e düşürüldüğünde)**		
FONKSİYONLAR	**Korelasyon katsayısı**	**MAE**	**RAE (%)**	**Korelasyon katsayısı**	**MAE**	**RAE (%)**
Gaussian Processes	0,5236	40,6054	107,9636	0,6259	40,1338	106,7099
Linear Regression	0,2921	35,3558	94,0057	0,7525	25,0961	66,7267
Multilayer Perceptron	0,7047	23,1496	61,5513	0,5197	39,5961	105,2801
SMOreg	0,6797	24,5578	65,2955	0,7918	21,5708	57,3536
LAZY						
IBK	0,4348	31,7511	84,4213	0,7265	29,6426	78,8151
KStar	0,4618	30,737	81,7252	0,709	26,4487	70,3232
LWL	0,5493	25,8851	68,8246	0,7979	23,9224	63,6062
META						
Additive Regression	0,6603	24,1608	64,24	0,7883	24,1991	64,3417
Bagging	-0,3738	37,3717	99,3657	0,8862	24,33	64,6898
Random Committee	0,6058	28,5917	76,0211	0,7023	26,8618	71,4216
Randomizable Fitered Classifer	0,3044	40,934	108,8375	0,6627	31,617	84,0649
RandomSubSpace	0,4785	32,9529	87,6169	0,7174	25,3419	67,3804
RegresiyonByDiscretization	0,5501	29,3266	77,9749	0,7638	26,075	69,3296
RULES						
Decision Table	0,1314	36,1975	96,2438	0,1318	35,0326	93,1465
M5Rules	-0,3381	36,6399	97,4199	0,6715	28,6709	76,2317
TREE						
Desicion Stump	0,5295	26,1019	69,4011	0,7643	25,1587	66,8931
M5P	-0,3381	36,6399	97,4199	0,7243	26,4012	70,1969
Random Forest	0,6743	29,9249	79,5658	0,7833	21,9777	58,4354
Random Tree	0,1814	40,9315	108,8307	0,6129	32,005	85,0966
REP Tree	-0,4942	37,6102	100	0,4737	26,5315	70,5432
Genetic Programming	0,7579	38,001	101,039	0,6298	46,7985	124,4302

Miyazaki94 veri setine WEKA'nın select attributes menüsü altındaki CfsSubsetEval ve GeneticSearch uygulandığında veri setinde bulunan 9 öznitelikten 3 tanesi seçilmiştir. Bunlar KLOC, FORM, FILE öznitelikleridir. Öznitelik alt kümesine, bağımlı öznitelik olan MM özniteliği de eklendiğinde öznitelik sayısı 4'e çıkmıştır. WEKA ortamında bulunan algoritmalar, Miyazaki94 veri setine öznitelik seçimi uygulanmadan önce ve öznitelik seçimi uygulandıktan sonra çalıştırılmıştır. Algoritmaların bulduğu tahmin sonuçları Tablo 4.11'de belirtilmiştir. Tablo 4.11 incelendiğinde, öznitelik seçiminden önce yapılan en iyi tahmin sonucunu Genetic Programming algoritması bulurken öznitelik seçiminden sonra en iyi tahmin sonucunu Bagging algoritması bulmuştur. Veri setine uygulanan algoritmalar arasından REP

Tree algoritması öznitelik seçiminden önce en kötü tahmin sonucunu bulurken öznitelik seçiminden sonra en kötü performansı Decision Table algoritması göstermiştir.

4.2.3. Maxwell, China ve COCOMONASA Veri Setleri Simülasyon Sonuçları

Tez çalışmasının bu kısmında, öznitelik seçim olarak GA ile PSO algoritmasının kullanılması sonucunda yazılım projelerinin maliyet tahmin sonuçları karşılaştırılmıştır. GeneticSearch ve PSOSearch evrimsel meta sezgisel arama algoritmalarıdır. Tez çalışması kapsamında kullanılan bütün veri setlerine CfsSubsetEval altında GeneticSearch ve PSOSearch algoritmaları öznitelik seçim yöntemi olarak uygulanmıştır. Amaç GA ve PSO algoritmaları sayesinde yazılım maliyet tahminini yapacak en etkili öznitelikleri belirlemektir. Veri setlerinden Albrecht, Finnish, Kemerer, Miyazaki94, COCOMO81 ve COCOMONASA2 öznitelk seçimi sonucunda aynı alt kümleri vermiştir. Bu veri setleri çalışmanın bu kısmına dâhil edilmemiştir. Maxwell, China ve COCOMONASA veri setlerine öznitelik seçim yöntemi olarak GA ile PSO algoritması uygulandığında farklı alt kümeler oluşturmuştur. Bu veri setlerine MÖ algoritmaları 3 farklı şekilde uygulanmıştır. İlk önce ham veri setleri üzerinde herhangi bir öznitelik seçimi yapılmadan, algoritmalar 10 kat çapraz doğrulama ile çalıştırılmıştır, ikincisinde öznitelik seçimi olarak GA kullanılarak elde edilen özniteliklerle, üçüncüsünde öznitelik seçimi olarak PSO algoritması uygulanarak elde edilen özniteliklerle uygulanmıştır. Uygulamalar WEKA programında varsayılan ayarlar ile yapılmıştır. Elde edilen bulgular Tablo 4.12, Tablo 4.13 ve Tablo 4.14'te belirtilmiştir.

Tablo 4.12: Maxwell veri setine farklı öznitelik yöntemlerinin uygulanması.

Maxwell veri seti									
ALGORİTMALAR	**Öznitelik seçimi yapılmadan önce**			**GA ile öznitelik seçimi (öznitelik sayısı 27'den 19'a düşürüldüğünde)**			**PSO ile öznitelik seçimi (öznitelik sayısı 27'den 15'e düşürüldüğünde)**		
FONKSİYONLAR	**Korelasyon katsayısı**	**MAPE**	**RAE (%)**	**Korelasyon katsayısı**	**MAPE**	**RAE (%)**	**Korelasyon katsayısı**	**MAPE**	**RAE (%)**
Gaussian Processes	0,783	1,0681	62,4734	0,787	1,0681	62,6138	0,8034	1,0562	60,6559
Linear Regression	0,8085	1,0848	66,1746	0,8544	0,735	54,0379	0,835	0,8264	56,5191
Multilayer Perceptron	0,7641	1,2569	75,8327	0,816	1,2322	65,9951	0,712	1,4635	82,3826
Simple Linear Regression	0,8023	0,6863	62,9158	0,8023	0,6863	62,9158	0,8023	0,6863	62,9158
SMOreg	0,8191	0,9379	60,6894	0,818	0,6903	56,0642	0,8361	0,5424	50,7562
LAZY									
IBK	0,463	1,0379	87,8139	0,7593	0,987	71,5399	0,7432	0,9711	77,1675
KStar	0,7336	0,8179	73,5065	0,8596	0,7644	64,913	0,85	0,5755	64,3137
LWL	0,6089	1,3038	85,0007	0,5866	1,26	82,1897	0,6293	1,2116	77,2602
META									
Additive Regression	0,6875	0,8649	83,2987	0,6882	0,8577	82,9584	0,6948	0,7775	79,4345
Bagging	0,7711	0,9636	62,8573	0,7704	0,9397	62,0514	0,7738	0,9119	61,1296
Random Committee	0,7875	0,8509	63,531	0,6924	0,8024	67,46	0,749	0,7249	61,4333
Randomizable Fitered Classifer	0,7402	0,7931	70,2147	0,8235	0,9866	68,2747	0,7588	0,9414	75,6192
RandomSubSpace	0,6696	1,3822	75,3645	0,6474	1,2193	74,821	0,7885	1,2611	68,5983
RegresiyonByDiscretization	0,5893	1,1059	86,2615	0,6092	1,1146	83,0374	0,4559	0,9882	92,8163
RULES									
Decision Table	0,3139	1,1923	85,2422	0,417	0,8807	80,553	0,2688	1,1403	86,5602
M5Rules	0,7497	0,9477	61,3338	0,6528	0,7843	68,9875	0,7265	0,8311	63,0235
TREE									
Desicion Stump	0,5893	1,4398	82,9502	0,5893	1,4398	82,9502	0,5893	1,4398	82,9502
M5P	0,8175	0,9478	59,1822	0,8092	0,7365	58,6571	0,834	0,8132	58,1623
Random Forest	0,7612	1,0024	63,638	0,7621	0,8315	60,9218	0,7916	0,7674	58,1876
Random Tree	0,569	1,0298	90,5171	0,4398	0,7481	83,1359	0,613	0,8036	81,9882
REP Tree	0,5801	0,9835	76,4285	0,5781	1,0389	77,3167	0,5809	0,9821	76,2101

Tablo 4.12 incelendiğinde Maxwell veri setine WEKA'nın select attributes menüsü altındaki CfsSubsetEval ve GeneticSearch uygulandığında veri setinde bulunan 27 özniteliken 19 tanesi seçilmiştir. Seçilen öznitelikler Syear, App, Har, Dba, Source, T01, T02, T04, T06, T07, T08, T09, T10, T11, T13, T14, Duration, Size, Time olmuştur. Maxwell veri setine WEKA'nın select attributes menüsü altındaki CfsSubsetEval ve PSOSearch uygulandığında veri setinde bulunan 27 öznitelikten 15 tanesi seçilmiştir. Seçilen öznitelikler Syear, Har, Dba, T02, T03, T07, T08, T09, T10, T11, T13, T14, Duration, Size, Time olmuştur. Tablo 4.12 incelendiğinde veri seti üzerinde öznitelik seçiminin yapılması algoritmaların çoğunda hata oranlarını düşürürken çok azında aynı kalmıştır. PSO algoritması uygulanarak elde edilen özniteliklerle yapılan tahmin sonuçları GA'ya göre çoğunlukla hata oranları daha düşük korelasyon katsayıları daha yüksek çıkmıştır.

Tablo 4.13: China veri setine farklı öznitelik yöntemlerinin uygulanması.

China veri seti									
ALGORİTMALAR	**Öznitelik seçimi yapılmadan önce**			**GA ile öznitelik seçimi (öznitelik sayısı 19'dan 9'a düşürüldüğünde)**			**PSO ile öznitelik seçimi (öznitelik sayısı 19'dan 8'e düşürüldüğünde)**		
FONKSİYONLAR	**Korelasyon katsayısı**	**MAPE**	**RAE(%)**	**Korelasyon katsayısı**	**MAPE**	**RAE(%)**	**Korelasyon katsayısı**	**MAPE**	**RAE(%)**
Gaussian Processes	0,9532	2,0857	43,9852	0,9534	2,1105	44,6892	0,9516	2,1247	45,5934
Linear Regression	0,9889	0,2253	9,809	0,9859	0,241	11,1281	0,986	0,2023	10,6966
Multilayer Perceptron	0,9733	0,274	12,4698	0,9746	0,2834	13,4426	0,9767	0,3436	13,8993
Simple Linear Regression	0,9833	0,1447	11,1943	0,9833	0,1447	11,1943	0,9833	0,1447	11,1943
SMOreg	0,9897	0,0967	7,3095	0,9847	0,1318	9,6896	0,9853	0,1366	9,7544
LAZY									
IBK	0,8918	0,6125	42,4638	0,9076	0,6515	39,0501	0,9081	0,7697	40,5442
KStar	0,9646	0,1902	16,9892	0,9726	0,1733	14,6867	0,9717	0,1931	14,2883
LWL	0,8351	2,1625	59,5587	0,848	2,109	57,5662	0,8484	2,1304	57,7314
META									
Additive Regression	0,9336	0,7246	31,5019	0,9426	0,6305	28,5508	0,9426	0,6305	28,5508
Bagging	0,9605	0,192	13,8374	0,9596	0,1924	14,0416	0,9597	0,1923	14,035
Random Comittee	0,9474	0,2595	18,3457	0,9555	0,2485	16,3949	0,96	0,2726	17,5369
Randomizable Fitered Classifer	0,9669	0,2735	17,9285	0,9516	0,3073	21,2019	0,9293	0,4459	27,1662
Random SubSpace	0,9383	0,4835	22,9068	0,9395	0,6165	25,5068	0,9023	0,9952	37,729
Regresiyon By Discretization	0,9519	1,5141	36,5418	0,9423	1,5147	37,3399	0,9426	1,5142	37,2662
RULES									
Decision Table	0,9292	1,4551	36,0382	0,9292	1,4551	36,0382	0,9292	1,4551	36,0382
M5 Rules	0,977	0,1237	11,1973	0,9762	0,1457	11,2756	0,9763	0,1257	10,9405
TREE									
Desicion Stump	0,8155	2,3497	62,2355	0,8155	2,3497	62,2355	0,8155	2,3497	62,2355
M5P	0,9842	0,1239	10,6158	0,984	0,1452	10,8488	0,9832	0,1243	10,5864
Random Forest	0,9591	0,2522	15,0747	0,9584	0,2557	15,6308	0,9602	0,2943	15,7784
Random Tree	0,9283	0,3468	25,4871	0,8726	0,3424	28,5588	0,9155	0,3184	23,5675
REP Tree	0,9597	0,2108	15,0204	0,9595	0,2112	15,1243	0,9595	0,2112	15,1243

Tablo 4.13 incelendiğinde China veri setine WEKA'nın select attributes menüsü altındaki CfsSubsetEval ve GeneticSearch uygulandığında veri setinde bulunan 19 öznitelikten 9 tanesi seçilmiştir. Seçilen öznitelikler ID, Input, Output, Enquiry, File, PDR_UFP, Resource, Duration, N_effort olmuştur. China veri setine WEKA'nın select attributes menüsü altındaki CfsSubsetEval ve PSOSearch uygulandığında veri setinde bulunan 19 öznitelikten 8 tanesi seçilmiştir. Seçilen öznitelikler S ID, Input, Output, Enquiry, File, Resource, Duration, N_effort olmuştur. Tablo 4.13 incelendiğinde veri seti üzerinde öznitelik seçiminin yapılması algoritmaların çoğunda hata oranlarını yükseltmiştir çok azında düşürmüştür. GA uygulanarak elde edilen özniteliklerle yapılan tahmin sonuçları PSO'ya göre çoğunlukla hata oranları daha düşük korelasyon katsayıları daha yüksek çıkmıştır.

Tablo 4.14: COCOMONASA veri setine farklı öznitelik yöntemlerinin uygulanması.

COCOMONASA veri seti									
ALGORİTMALAR	**Öznitelik seçimi yapılmadan önce**			**GA ile öznitelik seçimi yapılmış (öznitelik sayısı 17'den 9'a düşürüldüğünde)**			**PSO ile öznitelik seçimi yapılmış (öznitelik sayısı 17'den 11'e düşürüldüğünde)**		
FONKSİYONLAR	**Korelasyon katsayısı**	**MAPE**	**RAE (%)**	**Korelasyon katsayısı**	**MAPE**	**RAE (%)**	**Korelasyon katsayısı**	**MAPE**	**RAE (%)**
Gaussian Processes	0,6387	1,56	62,5047	0,6944	1,48	57,7443	0,6713	1,5587	60,4186
Linear Regression	0,7994	1,6558	57,2976	0,7648	1,7583	64,4223	0,7396	1,6606	78,7732
Multilayer Perceptron	0,8931	0,8211	41,6205	0,8879	0,7268	39,8792	0,8956	0,8134	37,607
SMOreg	0,719	1,091	57,6118	0,7508	0,9574	55,5536	0,6767	1,0759	66,3008
LAZY									
IBK	0,5768	0,8978	68,5184	0,555	0,9121	68,4662	0,5504	0,9553	69,8675
KStar	0,6772	0,4592	51,1294	0,7759	0,3837	47,6889	0,7592	0,388	45,8587
LWL	0,5768	0,8978	68,5184	0,7535	1,9057	58,8065	0,748	1,8329	62,0653
META									
Additive Regression	0,8255	0,7546	46,4632	0,7874	0,7885	52,1701	0,7843	0,8284	53,4854
Bagging	0,8083	0,8773	42,921	0,8154	0,8181	42,4936	0,8103	0,8849	43,4221
Random Comittee	0,8059	0,8793	51,5926	0,8707	0,5988	41,9377	0,8573	0,6216	43,9109
Randomizable Fitered Classifer	0,8059	0,8793	51,5926	0,8274	0,3668	43,3872	0,7833	0,4755	49,2079
Random SubSpace	0,8617	2,3585	59,786	0,7872	2,049	58,2429	0,7274	2,3334	58,8354
Regresiyon By Discretization	0,8179	1,0923	48,1966	0,7511	1,1789	56,6852	0,7514	1,1809	56,6425
RULES									
Decision Table	0,4577	0,7835	60,5639	0,5317	0,83	59,0642	0,5455	0,7426	55,4483
M5 Rules	0,9152	0,9188	36,4397	0,9064	1,0434	40,1402	0,9042	1,0982	41,5366
TREE									
Desicion Stump	0,6981	2,4527	70,3256	0,6981	2,4527	70,3256	0,6981	2,4527	70,3256
M5P	0,922	0,9282	35,0178	0,9118	1,0348	38,9148	0,9021	1,0907	40,8145
Random Forest	0,8196	0,8406	49,0968	0,8441	0,7052	45,5941	0,7992	0,7495	47,172
Random Tree	0,7029	1,0006	59,0168	0,7659	0,521	46,3355	0,674	1,0309	59,8381
REP Tree	0,594	1,4896	67,0803	0,5991	1,4835	66,3495	0,5938	1,49	67,0666

Tablo 4.14 incelendiğinde COCOMONASA veri setine WEKA'nın select attributes menüsü altındaki CfsSubsetEval ve GeneticSearch uygulandığında veri setinde bulunan 17 öznitelikten 9 tanesi seçilmiştir. Seçilen öznitelikler RELY, DATA, TIME, STOR, VIRT, TURN, LEXP, TOOL, LOC olmuştur. COCOMONASA veri setine WEKA'nın select attributes menüsü altındaki CfsSubsetEval ve PSOSearch uygulandığında veri setinde bulunan 17 öznitelikten 11 tanesi seçilmiştir. Seçilen öznitelikler RELY, DATA, TIME, STOR, VIRT, TURN, VEXP, LEXP, MODP, TOOL, LOC olmuştur. Tablo 4.13 incelendiğinde veri seti üzerinde GA uygulanarak öznitelik seçiminin yapılması algoritmaların çoğunda korelasyon katsayısını yükseltmiş ve hata oranlarını düşürmüştür. GA uygulanarak elde edilen özniteliklerle yapılan tahmin sonuçları PSO'ya göre çoğunlukla hata oranları daha düşük korelasyon katsayıları daha yüksek çıkmıştır.

4.3. BULGULARIN KARŞILAŞTIRILMASI

Bu bölümde, Yapay Zekâ tabanlı yazılım maliyet tahmini yapan çalışmaların geniş bir literatür taraması yapılmıştır. Araştırılan çalışmaların incelenmesi sonucunda elde edilen analizler Tablo 4.15'te sunulmuştur. Mevcut çalışmalar, yazılım maliyet tahmin yöntemine, kullandıkları veri setlerine, öznitelik seçimi yapıp yapmadıklarına ve değerlendirme ölçütlerine göre karşılaştırılmıştır. Yapay Zekâ tabanlı yazılım maliyet tahmini yapan çalışmalarda çoğunlukla performans değerlendirme ölçütü olarak; korelasyon katsayısı, MMRE, MAPE, MAE, RAE, RMSE, PRED'i kullandıkları tespit edilmiştir.

Bu bölümün temel amacı, araştırmacılara yazılım maliyet tahmininde hangi Yapay Zekâ yönteminin umut verici doğruluk tahmini yaptığını öğrenmesine yardımcı olmaktır. Dolayısıyla litaratür taraması sonucunda elde edilen bu analiz tablosu araştırmacılara önemli bir kaynak oluşturacaktır.

Tablo 4.15 incelendiğinde, Yapay Zekâ tabanlı yazılım maliyet tahmini çalışmalarının çok eskilere dayandığı görülmektedir. Özellikle YSA alanında yapılan çalışmalar literatürde geniş bir yer almaktadır. Regresyon tabanlı tahmin yöntemleri de yazılım maliyet tahmininde sık kullanılan yöntemler arasındadır. Yazılım maliyet tahmininde kullanılan veri setleri tahmin doğrulunu etkilemektedir. Yapılan çalışmaların çoğu yazılım maliyet tahmin yöntemini test etmek için hazır veri setlerini kullanmaktadır. Daha eski çalışmalar incelendiğinde yazılım maliyet tahmini yapıldığında öznitelik seçimine çok önem verilmediği görülmektedir. Bu tez çalışmasında yazılım maliyet tahmininde kullanılan hazır veri setleri üzerinde öznitelik seçiminin yapılması tahmin doğruluğunu olumlu yönde etkilediği tespit edilmiştir.

Tablo 4.15: Yapay Zekâ tabanlı yazılım maliyet tahmini yapan çalışmaların analizi.

Referans	Yöntem	Veri seti	ÖS	MMRE	MAE	RAE (%)	Korelasyon
Deng ve diğ., 2011	KNN	Desharnais	✓	0,36			
Wittig and Finnie, 1997	YSA	ASMA projeleri	X	0,29			
	YSA	Desharnais	X	0,17			
Marapelli, 2019	Doğrusal Regresyon	COCOMO81	X		874,477	96,3751	0,6102
		COCOMONASA	X		247,0465	57,2976	0,7994
		COCOMONASA2	X		430,7269	66,6849	0,7294
Marapelli, 2019	KNN	COCOMO81	X		782,5524	86,2442	0,0768
		COCOMONASA	X		295,4267	68,5184	0,5768
		COCOMONASA2	X		445,7796	69,0154	0,659
Finnie ve diğ., 1997	YSA	299 proje verisi	X	0,352			
	Doğrusal Regresyon	299 proje verisi	X	0,623			
Oliveira, 2005	YSA	NASA18	X	0,187			
	Destek Vektör Regresyon	NASA18	X	0,179			
Stamelosa ve diğ., 2003	Doğrusal Regresyon	ISBSG sürüm 6	X	0,23			
Sentas ve diğ., 2005	Basit Doğrusal Regresyon	ISBSG sürüm 7	X	0,3598			
Karataş, 2011	YSA	COCOMO81	X	0,41			
Idri ve diğ., 2002	YSA	COCOMO81	X	0,8435			
Ayyıldız, 2007	YSA	YEEM	X	0,09			
Sandhu ve diğ., 2008	Bulanık Model	NASA18	X	0,11943			
Adailer, 2008	Doğrusal Regresyon	ISBSG sürüm 9	X	0,023			
	YSA	ISBSG sürüm 9	X	0,047			
Malhotra ve Jain, 2011	Doğrusal Regresyon	China	✓	0,1797	1981,48	54,16	0,79
	SVM	China	✓	0,2563	1774,36	48,49	0,81
	YSA	China	✓	1,4379	2561,00	71,50	0,75
	Karar Ağacı	China	✓	0,1706	1173,43	32,02	0,93
	Torbalama	China	✓	0,7423	1668,03	45,79	0,83
Singh ve Kumar, 2020	Doğrusal Regresyon	Desharnais	✓		2013,7987		0,7673
	Multilayer Perceptron	Desharnais	✓		2742,0907		0,6843
	RastgeleAğaç	Desharnais	✓		2148,8052		0,6496
Başkeleş ve diğ., 2007	Çok Katmanlı Perseptron	COCOMO81	✓	0,9173			
	Destek Vektör Regresyon	COCOMO81	✓	0,3328			
	Karar Ağacı	COCOMO81	✓	0,3312			
Attarzadeh ve OW, 2010	Bulanık Model	COCOMO81	X	0,366			
Lefley ve Shepperd, 2003	EKK Regresyon	Finnish veri seti	X	0,469			0,846
	Genetik Programlama	Finnish veri seti	X	0,376			0,937
	YSA	Finnish veri seti	X	0,688			0,806

ÖS: Öznitelik Seçimi, gösterim ✓: evet, X: hayır.

Bu tez çalışması kapsamında her bir veri setine uygulanan MÖ algoritmalarının bulduğu en iyi yazılım maliyet tahmin sonucu, literatüredeki çalışmalar ile karşılaştırılması için, Tablo 4.16'da verilmiştir. Bunların dışında daha kötü tahmin sonuçları da mevcuttur.

Tablo 4.16: Her bir veri seti üzerinde yapılan en iyi tahmin sonucu.

Veri Seti	Yöntem	MAPE	MAE	RAE(%)	Korelasyon
Albrecht	Random Committee	0,8487	5,8819	28,847	0,9611
China	SMOReg	0,0967	270,4561	7,3095	0,9897
COCOMO81	Additive Regression	3,1646	471,6203	51,9767	0,8095
COCOMONASA	M5P	0,9282	150,9841	35,0178	0,922
COCOMONASA2	LWL	0,9994	332,7218	51,5118	0,8183
Finnish	Random Forest	0,0124	0,0976	9,5354	0,9942
Kemerer	SMOreg	0,4229	96,4073	59,9051	0,6946
Maxwell	SMOreg	0,5424	3188,8894	50,7562	0,8361
Miyazaki94	Bagging	0,7269	24,33	64,6898	0,8862

Bu tez çalışmasında 9 farklı veri seti üzerinde çalışılmıştır. Bu sayı diğer çalışmalarda kullanılan veri setleri sayısından daha fazladır. Literatürdeki çalışmalar incelendiğinde çoğu çalışmada kullanılan veri setleri üzerinde öznitelik seçiminin dikkate alınmadığı görülmektedir. Bu çalışmada öznitelik seçiminin hazır veri setleri üzerindeki önemi vurgulanmış ve yazılım maliyet tahmini gerçekleştirilmeden önce veri setleri üzerinde öznitelik seçimi yapılmıştır. Veri setleri üzerinde öznitelik seçimi yapılmadan önce ve öznitelik seçimi yapıldıktan sonra 25 MÖ algoritması farklı senaryolarda çalıştırılmıştır. Bu sayı literatürdeki diğer çalışmalardan daha fazladır. Bu durum, bu alanda çalışacak araştırmacılara geniş bir bakış açısı sağlamaktadır. Yazılım maliyet tahmini gerçekleştirildiğinde hata oranlarının literatürdeki çalışmalar ile karşılaştırılması için birçok değerlendirme ölçütü ele alınmıştır. Bunlar Korelasyon katsayısı, MAE, RMSE, RAE, RRSE ve MAPE olmuştur. Diğer çalışmalarda bu kadar değerlendirme ölçütü ele alınmamıştır.

Bu çalışmanın amacı Yapay Zekâ alanında yazılım maliyet tahmini gerçekleştirmek olduğundan iyi ve kötü tahmin sonuçları elde etmek ve bunları değerlendirmek olası bir sonuçtur. Bu durum araştırmacılara karşılaştırma yapmaları açısından kapsamlı bir kaynak oluşturacaktır.

5. TARTIŞMA VE SONUÇ

Yazılım maliyet tahmini, yazılım geliştirme projelerinin en mühim aşamalarından biridir. Yazılımın soyut olması ve birçok bilinmeyeni içermesi yazılım geliştirme sürecini hem zorlaştırmakta hem de süreç zaman almaktadır. Günümüzde gelişen teknoloji ile paralel olarak yazılımlar daha çok önem kazanmakta ve ihtiyaçlara cevap verebilmesi için daha karmaşık yazılımlar geliştirilmektedir. Yanlış yapılan yazılım maliyeti ve zaman tahminleri yazılım projelerinin başarısızlıkla sonuçlanmasına sebep olmaktadır. Bu yüzden yazılım maliyet tahmininin doğruluğunu artırmak için birçok yazılım maliyet tahmin yöntemi geliştirilmiştir. Bu tahmin yöntemlerinden biri de Yapay Zekâ yöntemleridir.

Bu tez çalışmasında yazılım projelerinin maliyet tahmini için, Yapay Zekâ yöntemlerinden, MÖ algoritmaları kullanılarak üç farklı model geliştirilmiştir. Geliştirilen her bir model farklı veri setleri üzerinde uygulanmıştır. Yazılım maliyet tahmininde kullanılan veri setlerinin öznitelikleri, tahmin doğruluğunu önemli ölçüde etkilemektedir. Yazılım maliyetini tahminleme sürecinde öznitelik seçiminin göz ardı edilmesi tahmin sonucunu olumsuz yönde etkilediği tespit edilmiştir. Tez çalışmasında yazılım maliyet tahmini için öznitelik seçimi CfsSubsetEval ile birlikte GeneticSearch ve PSOSearch arama algoritmaları kullanılarak yapılmıştır. Bu sayede öznitelik seçiminin yazılım maliyet tahmin doğruluğunu nasıl iyileştirdiği görülmüştür.

İlk geliştirilen model COCOMO81, COCOMONASA ve COCOMONASA2 veri setleri üzerinde WEKA programında bulunan MÖ algoritmaları kullanılarak iki farklı şekilde gerçekleştirilmiştir. Birinci bölümde; WEKA programında bulunan algoritmaların varsayılan ayarları tercih edilmiş şekilde yapılan simülasyonlarda COCOMO81 veri setinde en iyi tahmini Additive Regression algoritması, en kötü tahmini REP Tree algoritması; COCOMONASA veri setinde en iyi tahmini M5P algoritması, en kötü tahmini Decision Table algoritması; COCOMONASA2 veri setinde en iyi tahmini LWL algoritması, en kötü tahmini Decision Table algoritması sunmuştur. İkinci bölüm; kendi parametrelerine ek olarak başka bir sınıflandırıcı ve onun paremetrelerini alan algoritmalar için farklı parametreler girilerek bütün olasılıklar denenmiştir. COCOMO81 veri setinde en iyi tahmini Random Committee

algoritması, en kötü tahmini Random SubSpace algoritması; COCOMONASA veri setinde en iyi tahmini M5P algoritmasının parametre olarak verilmesi ile Additive Regression algoritması, en kötü tahmini Random SubSpace algoritması; COCOMONASA2 veri setinde en iyi tahmini LWL algoritmasına parametre olarak verilen Random Committee algoritması, en kötü tahmini Random SubSpace algoritması sunmuştur. Geliştirilen ikinci modelde Albrecht, Finnish, Kemerer, Maxwell ve Miyazaki94 veri setleri kullanılmıştır. Veri setleri üzerinde WEKA programında bulunan MÖ algoritmaları iki şekilde çalıştırılmıştır. İlkinde ham veri seti üzerinde çalıştırılan algoritmalar, daha sonra veri setlerine öznitelik seçimi yapılarak tekrar çalıştırılmıştır. Sonuçlar incelendiğinde GA kullanılarak veri setleri üzerinde öznitelik seçiminin yapılması yazılım maliyet tahminini olumlu yönde etkilediği görülmüştür. Bu çalışmada ayrıca WEKA'nın eski sürümünde bulunan Genetik Programlama da yazılım maliyet tahmini için kullanılmıştır. Analiz sonuçları incelendiğinde Genetik Programlamanın yazılım maliyet tahmininde başarılı bir şekilde kullanılabildiği gözlemlenmiştir. Üçüncü geliştirilen modelde Maxwell, China ve COCOMONASA2 veri setleri üzerinde yazılım maliyet tahmini gerçekleştirilmiştir. Veri setleri üzerinde WEKA ortamında bulunan MÖ algoritmaları üç şekilde çalıştırılmıştır. İlkinde algoritmalar ham veri setleri üzerinde varsayılan ayarlarla çalıştırılmıştır. İkincisinde veri setleri üzerinde CfsSubsetEval ile birlikte GeneticSearch arama algoritmasıyla öznitelik seçimi yapılmıştır. Elde edilen öznitelik alt kümesiyle yazılım maliyet tahmini yapılmıştır. Üçünsünde veri setleri üzerinde CfsSubsetEval ile birlikte PSOSearch arama algoritmasıyla öznitelik seçimi yapılmıştır. Elde edilen öznitelik alt kümesiyle yazılım maliyet tahmini yapılmıştır. China veri seti üzerinde yapılan yazılım maliyet tahmini test sonuçları incelendeğinde ham veri seti üzerinde yapılan tahmin sonucunun hem GA hem de PSO algoritması ile elde edilen öznitelik alt kümesiyle yapılan tahmin sonuçları arasında çok büyük bir fark olmadığı görülmüştür. Maxwell ve COCOMONASA veri setleri üzerinde öznitelik seçiminin yapılması yazılım maliyet tahminindeki hata oranlarını düşürmüştür. Maxwell veri setinde GA ile elde edilen öznitelik alt kümesiyle yapılan tahmin sonuçlarının hata oranları daha düşük çıkmıştır. COCOMONASA veri seti üzerinde PSO algoritması ile elde edilen öznitelk alt kümesiyle yapılan tahmin sonuçlarının hata oranları daha düşük çıkmıştır.

Bu çalışmada WEKA programı ile MÖ algoritmalarının PROMISE veri deposunda bulunan China, COCOMO81, COCOMONASA, COCOMONASA2, Albrecht, Finnish, Kemerer, Maxwell ve Miyazaki94 veri setleri kullanılarak yazılım maliyet tahmininde gösterdikleri performanslar incelenmiştir. Tahmin sonuçları incelendiğinde, algoritmaların hata oranlarının ve korelasyon katsayılarının uygulandıkları veri setlerine göre değişkenlik gösterdiği belirlenmiştir. Bir algoritmanın her zaman en iyi sonucu üretmediği, bazı algoritmaların bazı veri setlerinde çok iyi sonuçlar üretirken farklı parametrelerle ve farklı veri setlerinde kötü sonuçlar verebileceği gözlemlenmiştir. Ayrıca veri setlerindeki özniteliklerin, öznitelikleri belirlemek için kullanılan öznitelik seçim yönteminin tahmin sonucunu çok etkilediği fark edilmiştir. Performans değerleri incelendiğinde yazılım maliyet tahmini için kullanılan veri setleri üzerinde öznitelik seçiminin yapılması, genel olarak MÖ algoritmalarında iyileştirici sonuçların elde edilmesini sağladığı görülmüştür. Yazılım projelerinin maliyet tahmini için kullanılan hazır veri setleri kullanılmadan önce veri setlerinde öznitelik seçiminin yapılması maliyet tahmininin doğruluk oranını artıracağı sonucuna varılmıştır.

Bu çalışmada, literatürde ilk kez bu kadar çok sayıda MÖ algoritmasıyla yazılım maliyet tahmini gerçekleştirilmiştir. MÖ evrimsel tabanlı algoritmalarla öznitelik seçimi yapılmış ve yazılım maliyet tahmini için Genetik Programlama kullanılmıştır. Bu çalışma sayesinde yazılım maliyet tahmini için hangi MÖ algoritmasının kullanılabileceği, bu algoritmaların ilgili veri setlerine uygulandığında tahmin sonuçlarının neler olabileceği ve en iyi çalışan algoritmaların hangileri olduğu bilgisine ulaşılmıştır.

Bu alanda çalışmak isteyen araştırmacılar; farklı metodolojide hazırlanmış yazılım projelerinin veri setleri üzerinde öznitelik seçim yöntemlerini uygulayarak yazılım maliyet tahmini gerçekleştirebilir. GA ve BM gibi Yapay Zekâ'nın diğer yöntemlerinden melez sistemler oluşturarak yazılım projelerinin maliyet tahmini için farklı modeller geliştirebilir.

KAYNAKLAR

Abe S, Thawonmas R, Kobayashi Y, 1998, Feature selection by analyzing class regions approximated by ellipsoids, *IEEE Trans. On Systems, Man, and Cybernetics-Part C: Applications and Reviews*, 28(2), 282–287.

Ablameyko S, Goras L, Gori M, Piuri V, 2003, Neural Networks for Instrumentation, Measurement and Related Industrial Applications, *IOS Press*, 120, 138.

Adalier O, 2008, *Yapay Zeka Yöntemleri İle Yazılım Projelerinde Maliyet Kestirimi*, Doktora Tezi, Ege Üniversitesi, Fen Bilimleri Enstitüsü.

Aha DW, Kipler D, Albert MK, 1991, Instance-Based Learning Algortihms, *Machine Learning*, 6, 37–66.

Albrecht AJ, 1979, Measuring application development productivity, *Proceeding of the Joint SHARE, GUIDE and IBM application development symposium*, IBM Corporation.

Albrecht AJ, Gaffney JE, 1983, Software function, source lines of code, and development effort prediction: a software science validation. *IEEE Trans. Softw. Eng.* 9, 6 (1983), 639–648.

Allahverdi N, 2002, *Uzman Sistemler Bir Yapay Zeka Uygulaması*, Atlas Yayın Dağıtım, Ankara, ISBN 975-6574-10-0.

Alma ÖG, Özgül V, 2008, Regresyon Analizinde Kullanılan En Küçük Kareler Ve En Küçük Medyan Kareler Yöntemlerinin Karşılaştırılması, *Sdü Fen Edebiyat Fakültesi Fen Dergisi*, 3(2), 219–229.

Alpaydın E, 2011, *Yapay Öğrenme*, Boğaziçi Üniversitesi Yayınevi, İstanbul, ISBN: 978-605-4238-49-1.

Altaş İH, 1999, Bulanık Mantık: Bulanıklık Kavramı, Enerji-Elektrik- Elektromekanik-3e, 62, 80–85.

Attarzadeh I, Ow SH, 2010, A novel Algorithmic Cost Estimation Model Based on Soft Computing Technique, *Journal of Computer Science*, 6(2), 117–125.

Autonom, 2019*, Bilgisayarlı Görü,* https://www.autonom.com.tr/bilgisayarli-goru-computer-vision-nedir/, [Ziyaret tarihi: 31 Ağustos 2020].

Ayan TE, 2009, Kaynak Kısıtlı Çoklu Proje Programlama Problemi İçin Tavlama Benzetimi Algoritması, *Atatürk Üniversitesi İktisadi ve İdari Bilimler Dergisi*, 23(2), 101–118.

Aydemir E, 2019, *Weka ile Yapay Zeka*, Seçkin, Ankara, ISBN: 978-975-02-5536-6.

Aydın S, Özkul AE, 2015, Veri Madenciliği Ve Anadolu Üniversitesi Açık öğretim Sisteminde Bir Uygulama, *Eğitim ve Öğretim Araştırmaları Dergisi*, Ankara, 4(3), 38.

Ayyıldız M, 2007, *Yazılım Projeleri Ölçüm Sonuçları Veri tabanının Oluşturulması ve Yeni Yazılım Projelerinin Maliyet Tahmininde Kullanımı*, Doktora, Yıldız Teknik Üniversitesi, Fen Bilimleri Enstitüsü.

Başar A, 2017, Klasik ve Sezgisel Bulanık İkili Karşılaştırma ile Yazılım Geliştirme Projelerinin Maliyet Tahmini: Uygulama Örneği, *Bilişim Teknolojileri Dergisi*, 10(2), 129–137.

Beasley D, Bull DR, Martin RR, 1993, An Overview of Genetic Algorithms: Part 1, Fundamentals. *University Computing*, 15(2), 58–69.

Başkeleş B, Turhan B, Bener A, 2007, Software Effort Estimation Using Machine Learning Methods, *Computer and information sciences*, Ankara, IEEE.

Boehm BW, 1981, *Software Engineering Economics*, Prentice Hall.

Boehm B, Abts CA, Chulani S, Clark BK, Horowitz E, Madachy R, Reifer DJ, Steece B, 2000, *Software cost estimation with COCOMO II*, Prentice Hall.

Bosu MF, MacDonell SG, 2019, Experience: Quality Benchmarking of Datasets Used in Software Effort Estimation, *CM Journal of Data and Information Quality*, 11(4), 38.

Budak H, 2018, Özellik Seçim Yöntemleri ve Yeni Bir Yaklaşım, *Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü Dergisi*.

Caudill M, 1987, Neural networks primer, *J. AI Expert*, 2(12), 46–52.

Chen H, Lin Z, Mo L, Tan C, 2017, Identification Of Colorectal Cancer Using Near-İnfrared Spectroscopy And Adaboost With Decision Stump, *Analytical Letters*, 50(16), 2608-2618.Civalek Ö, 2003, Yapay Zeka, *Türkiye Mühendislik Haberleri*, 423, 40–50

Dalkey N, Helmer O, 1963, An Experimental Application of the Delphi Method to the Use of Experts, *The RAND Corporation*, Santa Monica.

Dede G, 2008, *Yapay Sinir Ağları İle Konuşma Tanıma*, Yüksek Lisans, Fen Bilimleri Enstitüsü.

Deng JD, Purvis M, Purvis M, 2011, Software Effort Estimation: Harmonizing Algorithms And Domain Knowledge İn An İntegrated Data Mining Approach, *International Journal of Intelligent Information Technologies (IJIIT)*, *7*(3), 41-53.

Ebren Kara Ş, Şamlı R, 2021, Genetik Algoritma İle Öznitelik Seçimi Yapılarak Yazılım Projelerinin Maliyet Tahmini, *Avrupa Bilim ve Teknoloji Dergisi (EJOSAT)*, 27, 985-994.

Ebren Kara Ş, Şamlı R, 2021, Software Cost Estimation Using Machine Learning Algorithms, *Applied Stochastic Models and Data Analysis International Conference (ASMDA21)*, 1-4 June, Virtual.

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

Elmas Ç, 2016, *Yapay Zeka Uygulamaları*, 3. Baskı, Seçkin yayınları, Anakara, ISBN: 978-975-02-3686-0.

Englert P, 2012, Locally Weighted Learning, https://www.ias.informatik.tu-darmstadt.de, [Ziyaret tarihi: 26 Ocak 2022].

Ertunç HM, 2012, Introductıon To Fuzzy Logic, https:// avesis.kocaeli.edu.tr/hmertunc/deneyim, [Ziyaret tarihi: 25 Eylül 2020].

Etkin EE, 2017, *Zaman Serilerinde Veri Madenciliği Öngörü Algoritmalarının Etkinlik Ve Verimliliğinin Bıst100 Hisse Senetleri Üzerinde Gerçeklenmesi,* Yüksek Lisans, Maltepe Üniversitesi, Fen Bilimleri Enstitüsü.

Fatullayev AG, 2013, *En Küçük Kareler Yöntemi,* http://www.kocaelimakine.com/wp content/uploads/2013/04/en-kucuk-kareler-yontemi-afet-golayoglu.pdf, [Ziyaret tarihi: 15 Şubat 2019].

Finnie GR, Wittig GE, Desharnais JM, 1997, A Comparison of Software Effort Estimation Techniques: Using Function Points with Neural Networks, Case-Based Reasoning and Regression Models, *Journal of Systems Software*, 39(3), 281–289.

Görz GN, 2005, *Yapay Zeka*, İnkılap Kitabevi, İstanbul, ISBN: 975-10-2405-6.

Gupta A, 2015, Classification Of Complex UCI Datasets Using Machine Learning And Evolutionary Algorithms, *International Journal Of Scientific & Technology Research*, 4(5), 85–94.

Gültekin M, 2019, *Makine Öğrenmesi Tabanlı Yazılım Maliyet Tahmini Yöntemlerinin Karşılaştırmalı Analizi*, Doktora, Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü.

Gürel A, Aslan LM, 2008, Konuşma Tanıma için İnsan-Makine Karşılaştırması, *Dilbilim Araştırma Dergisi,* 19, 77–90.

Güven Aydın ZB, 2021, *Makine Öğrenmesi Yöntemleri İle Yazılım Hata Tahmini*, Doktora Tezi, İstanbul Üniversitesi-Cerrahpaşa Lisansüstü Eğitim Enstitüsü.

Hall Mark A, 1999, *Correlation-based Feature Selection for Machine Learning*, Doktora Tezi, University of Waikato, Bilgisayar Bilimleri.

Hihn J, Habib-agahi H, 1991, Cost estimation of software intensive projects: A survey of current *practices. In Proceedings of the 13th international conference on Software engineering*, IEEE Computer Society Press, 276–287.

Huang D, Chow TWS, 2005, Efficiently searching the important input variables using Bayesian discriminant, *IEEE Trans. on Circuits and Systems-I: Regular Papers*, 52(4), 785–793.

Huang X, Capretz LF, Ren J, Ho D, 2003, A Neuro-Fuzzy Model for Software Cost Estimation, *Proceedings of the Third International Conference On Quality Software*, 0-7695-2015-4/03 $ 17.00 © 2003 IEEE.

Idri A, Khoshgoftaar TM, Abran A, 2002, *Can Neural Networks be easily Interpreted in Software Cost Estimation?*, 0-7803-7280-8/02/$10.O0.

Kaluza B, 2016, *Machine Learning in Java*, Pact Publishing.

Kargar K, Safari MJS, Khosravi K, 2021, Weighted instances handler wrapper and rotation forest-based hybrid algorithms for sediment transport modeling. *Journal of Hydrology*, 598, 126452.

Kartal Karataş E, 2011, *Yapay Sinir Ağları İle Yazılım Projesi Maliyet Tahmini*, Yüksek Lisans Tezi, İstanbul Üniversitesi Fen Bilimleri Enstitüsü.

Kemerer CF, 1987, An Empirical Validation Of Software Cost Estimation Models. *Commun. ACM* 30, 5(1987), 416–429.

Kennedy J, Eberhart R, 1995, Particle Swarm Optimization, *Neural Networks Proceedings IEEE International Conference on*, Perth, 1942-1948.Keskenler MF, Keskenler EF, 2017, Geçmişten Günümüze Yapay Sinir Ağları ve Tarihçesi, *Takvim-i Vekayi,* 5(2), 8–18.

Keskin M, Alptekin GI, 2016, Yazılım Maliyet Tahmininde İşlev Puanı Analizi ve Yapay Sinir Ağları Kullanımı*, UYMS'16 10. Ulusal Yazılım Mühendisliği Sempozyumu*, 24–26 Ekim 2016, Çanakkale.

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

Kitchenham B, Kansala K, 1993, Inter-item correlations among function points, *In Proceedings of the 15th International Conference on Software Engineering*, 229–238.

Kubat C, 2014, *Matlab Yapay Zeka ve Mühendislik Uygulamaları*, 2. Baskı, Pusula Yayınları, İstanbul, ISBN: 978-605-5106-12-6.

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

Kumari S, Pushkar S, 2013, Performance Analysis of the Software Cost Estimation Methods: A Review, *International Journal of Advanced Research in Computer Science and Software Engineering*, Jully 2013, India, CSE & BIT Mesra, ISSN: 2277 128X, 229-238

Kurbanoğlu S, 1992, Uzman Sistemler, *Türk Kütüphaneciliği*, 6(4), 190–193.

Kültür Y, 2006, *Software Effort Estimation Using Ensable of Neural Networks with Associative Memory,* Yüksek Lisans Tezi, Boğaziçi Üniversitesi Fen Bilimleri Enstütüsü.

Lefley M, Shepperd M J, 2003, Using Genetic Programming To İmprove Software Effort Estimation Based On General Data Sets, In *Genetic and Evolutionary Computation Conference* (pp. 2477-2487). Springer, Berlin, Heidelberg.

Leung H, Fan Z, 2002, *Software Cost Estimation, Handbook of Software Engineering and Knowledge Engineering*, ftp://cs.pitt.edu/chang/handbook/42b.pdf, [Ziyaret tarihi: 10 Şubat 2019].

Malhotra, Ruchika, Ankita Jain, 2011, Software Effort Prediction Using Statistical And Machine Learning Methods, *International Journal of Advanced Computer Science and Applications*, 2(1), 145-152.

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

Maxwell K, 2002, *Applied Statistics for Software Managers, Prentice-Hall, Englewood Cliffs*, NJ. Katrina D. Maxwell and Pekka Forselius. 2000. Benchmarking software-development productivity. *IEEE Softw.* 17(1), 80–88.

Merih K, 2016, *Kaos Kuramı ve Kaotik Düşünce Sistemi,* https://datalabtr.com/index.php/2016/05/06/kaos-kurami-ve-kaotik-dusunce-sistemi/, [Ziyaret tarihi: 11 Eylül 2020].

Mijwil MM, 2017, *Yapay Sinir Ağlar Yapısı ve Fonksiyonu*, ResearchGate, https://www.researchgate.net/publication/323073864, [Ziyaret tarihi: 14 Ağustos 2020].

Miyazaki Y, Terakado M, Ozaki K, Nozaki H, 1994, Robust Regression For Developing Software Estimation Models. *J. Syst. Softw.* 27, 13–16.

Moghaddam SAV, 2014, *Etkin Sınıflandırma İçin Genetik Algoritma Tabanlı Öznitelik Alt Küme Seçimi*, Yüksek Lisans Tezi, Gazi Üniversitesi Fen Bilimleri Enstitüsü.

Mohammed A, Rafiq S, Sihag P, Kurda R, Mahmood W, Ghafor K, Sarwar W, 2020, ANN, M5P-Tree And Nonlinear Regression Approaches With Statistical Evaluations To Predict The Compressive Strength Of Cement-Based Mortar Modified With Fly Ash. *Journal of Materials Research and Technology*, *9*(6), 12416-12427.

Molani M, Ghaffari A, Jafarian A, 2014, A New Approach to Software Project Cost Estimation using a Hybrid Model of Radial Basis Function Neural Network and Genetic Algorithm, *Indian Journal of Science and Technology*, 7(6), 838–843.

Nabiyev VV, 2016, *Yapay Zeka*, 5. Baskı, Seçkin Yayınları, Ankara, ISBN: 978-975-02-3727-0.

NASA 2003, *Handbook for Software Cost Estimation*, NASA, Jet Propulsion Laboratory, Pasadena, California.

Nookala GKM, Pottumuthu BK, Orsu N, Mudunuri SB, 2013, Performance Analysis And Evaluation Of Different Data Mining Algorithms Used For Cancer Classification, International Journal of Advanced Research in Artificial Intelligence (IJARAI), 2(5), 49-55.

Oleron P, 1996, *Zeka*, Yeni Yüzyıl Kitaplığı, İletişim yayınları/ Çeviri – Ela Güngören, İstanbul, ISBN: 978-975-47-0237-8.

Oliveira AL, 2006, Estimation Of Software Project Effort With Support Vector Regression, *Neurocomputing, 69*(13-15), 1749-1753.

Omran BA, Chen Q, Jin R, 2016, Comparison Of Data Mining Techniques For Predicting Compressive Strength Of Environmentally Friendly Concrete, *Journal of Computing in Civil Engineering*, *30*(6), 04016029.

Ozan E, 2020, *Robotlar ve Uygulamaları,* Yüksek Lisans Tezi, Batman Üniversitesi Fen Bilimleri Enstitüsü.

Öcal K, 2005, *Otomatik Konuşma Tanıma Algoritmalarının Uygulamaları,* Yüksek Lisans Tezi, Ankara Üniversitesi Fen Bilimleri Enstitüsü.

Öztemel E, 2016, *Yapay Sinir Ağları*, 4. Basım, Papatya Yayıncılık, İstanbul, ISBN: 978-975-6797-39-6.

Parkinson CN, 1957, Parkinson‘s Law and Other Studies in Administration, *Houghton-Miffin, Boston*.

PMBOK 2000, *A Guide to the Project Management Body of Knowledge*, Project Management Institute.

Poincaré JH, 1912, Henry Poincaré, http://www.chaos.umd.edu./misc/poincare.html, [Ziyaret tarihi: 11 Eylül 2020].

Prowmes B, 2019, *Makine Öğrenmesi*, http://www.prowmes.com/blog/makine-ogrenmesi/, [Ziyaret tarihi: 14 Ağustos 2020].

Putnam LH, 1978, A general empirical solution to the macro software sizing and estimating problem, *IEEE transactions on Software Engineering*, 4, 345–361.

PyCon, 2014, *How To Get Started with Machine Learning – Melanie Warrick’s PyCon 2014 Talk | Hackbright Academy*, https://youtu.be/uBorfxosVYs, [Ziyaret tarihi: 14 Ağustos 2020].

Sandhu PS, Bassi P, Brar AS, 2008, Software Effort Estimation Using Soft Computing Techniques, *World Academy of Science, Engineering and Technology, 46*, 488–491.

Sentas P, Angelis L, Stamelos I, Bleris G, 2005, Software Productivity And Effort Prediction With Ordinal Regression, *Information And Software Technology*, *47*(1), 17–29.

Sezer A, 2008, *Yazılım Projelerinde Yapay Sinir Ağı Uygulaması İle Maliyet Tahmini*, Yüksek Lisans Tezi, Haliç Üniversitesi Fen Bilimleri Enstitüsü.

Shan Y, McKay CJ, Essam DL, 2002, Software Project Effort Estimation Using Genetic Programming, *International Conference on Communications Circuits and Systems.*

Singh AJ, Kumar M, 2019, Effort Estimation Using Hybridized Machine Learning Techniques for Evaluating Student's Academic Performance, In *International Conference on Information Management & Machine Intelligence* (pp. 65-75). Springer, Singapore. Singh P, Agrawal S, 2013, Node Localization in Wireless Sensor Networks Using the M5P Tree and SMOreg Algorithms, *5th International Conference on Computational Intelligence and Communication Networks.*

Singh BK, Misra AK, 2012, Software Effort Estimation by Genetic Algorithm Tuned Parameters of Modified Constructive Cost Model for NASA Software Projects, *International Journal of Computer Applications*, 59 (9) , 10.5120/9577-4053.

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

Sönmez S, 2020, Yapay Zeka (ders notları), http://bit.ly/2ieeRcT, [Ziyaret tarihi: 12 Ağustos 2020].

Stamelos I, Angelis L, Morisio M, Sakellaris E, Bleris GL, 2003, Estimating The Development Cost Of Custom Software, *Information & Management*, *40*(8), 729-741.Szeliski R, 2010, Computer Vision: Algorithms and Applications, *Springer Science & Business Media*, ISBN 978-1-84882-935-0.

Şahin AE, 2001, Eğitim Araştırmalarında Delphi Tekniği Ve Kullanımı, *Hacettepe Üniversitesi Eğitim Fakültesi Dergisi*, 20, 215–220.

Şahin M, 2019, *Melez Yapay Zeka Sistemleri*, http://mehmetsahin.web.tr/melez-yapay-zeka-sistemleri/, [Ziyaret tarihi: 26 Eylül 2020].

Şen Z, 2004, *Mühendislikte Bulanık Mantık (Fuzzy) ile modelleme prensipleri*, Su Vakfı Yayınları, İstanbul, ISBN: 975-850-923-3.

TDK, 2020, Zekâ, http://bit.ly/1CcStkR, [Ziyaret tarihi: 11 Ağustos 2020].

TDK, *Yazılım*, http://www.tdk.gov.tr, [Ziyaret tarihi: 22 Haziran 2018].

Techinside, 2017, *Ses Tanıma Teknolojisi Bu 5 Sektörü Değiştirecek,* https://www.techinside.com/ses-tanima-teknolojisi-5-sektoru-degistirecek/, [Ziyaret tarihi: 04 Eylül 2020]

Tekin BY, 2020, *Benzetimli Tavlama (Simulated Annealing) Algoritması*, https://globalaihub.com/benzetimli-tavlama-simulated-annealing-algoritmasi/ [Ziyaret tarihi: 21 Ağustos 2020].

Torkul O, Gülseçen S, UyarOğlu Y, Çağıl G, Uçar MK, 2017, *Mühendislikte Yapay Zeka Uygulamaları,* Sakarya Üniversitesi Kütüphanesi Yayınevi, Sakarya, ISBN: 978-605-4735-98-3

Tofallis C, 2015, A Better Measure Of Relative Prediction Accuracy For Model Selection And Model Estimation, *Journal of the Operational Research Society*, 66, 1352–1362.

Tran B, Xue B, Zhang M, 2015, Genetic programming for feature construction and selection in classification on high-dimensional data, *Springer-Verlag Berlin Heidelberg: Regular Papers*, 8, 3–15.

Tuzcuoğlu H, 2003,Yapay Zeka Teknikleri, Depremde Kullanılması ve Küme Kavramları, Dokuz Eylül Üniversitesi, Mühendislik Fakültesi, *Fen ve Mühendislik Dergisi*, 5(1), 73–88.

Weka last versiyon, 2018, *Downloading and installing Weka*, https://waikato.github.io/weka-wiki/downloading_weka/, [Ziyaret tarihi: 24 Mayıs 2021].

WEKA, *The Workbench for machine learning***,** http://old-www.cms.waikato.ac.nz/ml/weka, [Ziyaret tarihi: Mart 2021].

WekaGP 3.4.12, 2007, *Genetic Programming Classifier for Weka*, https://sourceforge.net/projects/wekagp/files/latest/download, [Ziyaret tarihi: 24 Mayıs 2021].

Witten IH, Frank E, Hall MA, 2011, *Data Mining Practical Machine Learning Tools and Techniques Third Edition*, Morgan Kaufmann, USA, ISBN: 978-0-12-374856-0.

Witting G, Finnie G, 1997, Estimating software development effort with connectionist models, *Information and Software Technology*, 39, 469–476.

Woundenberg F, 1991, An Evaluation of Delphi, *Technological Forecasting and Social Change*, 40: BI-ISO.

Yakar Ö, 2016, *Sözcük ve Hece Tabanlı Konuşma Tanıma Sistemlerinin Karşılaştırılması,* Yüksek Lisans, Fen Bilimleri Enstitüsü.

Yalçın N, 2008, Konuşma Tanıma Teorisi ve Teknikler, *Kastamonu Eğitim Dergisi,* 16(1), 249–266.

Yamanol İ, 2016, *Robotların Kısa Tarihi*, https://www.bilimkurgukulubu.com/genel/bilim-teknoloji/robotlarin-kisa-tarihi/, [Ziyaret tarihi: 17 Nisan 2020].

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

Zadeh LA, 1965, Fuzzy sets, *Information and Control*, 8, 338–353.

Zhao Y, Zhang Y, 2008, Comparison Of Decision Tree Methods For Finding Active Objects, *Advances in Space Research*, 41(12), 1955–1959.

EKLER

EK 1. SÖZLÜK

Additive Regression	**:** Toplamsal Regresyon
Artificial Neural Networks	**:** Yapay Sinir Ağları
Associate	**:** İlişki Kurma
Attribute Selected Classifier	**:** Öznitelik Seçici Sınıflandırıcı
Australian Software Metrics Association	**:** Avustralya Yazılım Metrikleri Birliği
Automated Production	**:** Otomatik Kod Üretimi
Bagging	**:** Torbalama
Bottom – Up	**:** Aşağıdan Yukarıya
Choose	**:** Seç
Classification	**:** Sınıflandırma
Clustering	**:** Kümeleme
Computer and Thought	**:** Bilgisayar ve Düşünce
Computer Vision	**:** Bilgisayarlı Görme
Confusion Matrix	**:** Karışıklık Matrisi
Correctly Classified Instance	**:** Doğru Yerleştirme Başarısı
Correlation Coefficient	**:** Korelasyon Katsayısı
Cross Validation Parameter Selection	**:** Çapraz Doğrulama Parametre Seçimi
Desicion Stump	**:** Karar Kütüğü
Desicion Table	**:** Karar Ağacı
Development Schedule Constraint	**:** Geliştirme Takvimi Kısıtı
Expert Systems	**:** Uzman Sistemler
Facilities	**:** Araç Gereçler
Forecasting	**:** Tahminleme
Functions	**:** Fonksiyonlar
Future Selection	**:** Öznitelik Seçimi
Fuzzy Logic	**:** Bulanık Mantık
Gaussian Process	**:** Gauss Süreçleri
General Public Licence	**:** Genel Kamu Lisansı
Genetic Algorithms	**:** Genetik Algoritmalar
Genetic Programming	**:** Genetik Programlama
Genetic Search	**:** Genetik Arama
Global Best	**:** Küresel En İyi
Heuristic	**:** Sezgisel
Hybrid Systems	**:** Melez Sistemler
Input Mapped Classifier	**:** Giriş Eşlemeli Sınıflandırıcı
International Software Benchmarking Standards Group	**:** Uluslararası Yazılım Kıyaslama standart Grubu
K-Nearest Neighbours Classifier	**:** K-En Yakın Komşu Sınıflandırıcı
Kstar	**:** Kyıldız
Laboratory For Interchange Fuzzy Engineering	**:** Değişim Bulanık Mühendislik Laboratuvarı

Lazy Classifier : Tembel Sınıflandırıcılar
Linear Regression : Doğrusal Regresyon
Locally Weighted Learning : Yerel Ağırlıklı Öğrenme
Machine Learning : Makine Öğrenmesi
Mean Absolute Error : Ortalama Mutlak Hata
Mean Absolute Percentage Error : Ortalama Mutlak Hata Yüzdesi
Mean Magnitude Of Relative Error : Göreceli Hatanın Ortalama Büyüklüğü
Multilayer Perceptron : Çok Katmanlı Algılayıcı
Multi Scheme : Çoklu Şema
Particle Swarm Optimization : Parçacık Sürü Optimizasyonu
Personal Experience : Personel Deneyimi
Personel Best : Kişisel En İyi
Personnel Capability : Personel Yeteneği
Platform Difficulty : Platform Zorluğu
Prediction Accuracy : Tahmin doğruluğu
Pre-Processing : Veri Ön İşleme
Price – To – Win : Kazanmak İçin Fiyat
Product Reliability and Complexity : Ürün Doğruluğu ve Karmaşıklığı
Randomizable Filtered Classifier : Randomize Edilebilir Filtreli Sınıflandırıcı
Random Tree : Rastgele Ağaç
Random Sub Space : Rastgele Alt Boşluk
Random Committee : Rastgele Komite
Random Forest : Rastgele Orman
Rep Tree : Rep Ağacı
Regression By Discretization : Ayrıklaştırma İle Regresyon
Reinforcement Learning : Pekiştirici, Yarı Eğiticili Öğrenme
Relative Absulate Error : Bağıl Mutlak Hata
Robotics : Robotik
Root Mean Squared Error : Kök Ortalama Kare Hata
Root Relative Squared Error : Kök Ortalama Kare Hata
Rules : Kurallar
Segmentation : Bölütleme
Simple Linear Regression : Basit Doğrusal Regresyon
Simulated Annealing : Tavlama Benzetimi
Soft Computing : Esnek Hesaplama
Speech Recognition : Konuşma Tanıma
Stacking : İstifleme
Supervised Learning : Eğiticili Öğrenme
Top – Down : Yukarıdan Aşağıya
Travelling Salesman Problem : Gezgin Satıcı Problemi
Tree : Ağaç
University of South California : Güney Kalforniya Ünviersitesi
Unsupervised Learning : Eğiticisiz Öğrenme
Visualize : Görselleştirme
Vote : Oylama

Weighted Instances Handler Wrappler **:** Ağırlıklı Örnek İşleyici Sarmalayıcı
World Health Organization **:** Dünya Sağlık Örgütü