T.C.
MANİSA CELAL BAYAR ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ
YAZILIM ANABİLİM DALI
YAZILIM BİLİM DALI

DERİN ÖĞRENME YAKLAŞIMLARI İLE YAZILIM PROJE GEREKSİNİMLERİNİN SINIFLANDIRILMASI VE BAŞARIM SONUÇLARININ DEĞERLENDİRİLMESİ

Yazar
Tuğba ÇELİKTEN

Danışman
Dr. Öğr. Üyesi Fatih YÜCALAR

MANİSA–2023

Tuğba ÇELİKTEN	**DERİN ÖĞRENME YAKLAŞIMLARI İLE YAZILIM PROJE GEREKSİNİMLERİNİN SINIFLANDIRILMASI VE BAŞARIM SONUÇLARININ DEĞERLENDİRİLMESİ**	**2023**

TAAHHÜTNAME

Bu tezin Manisa Celal Bayar Üniversitesi Hasan Ferdi Turgutlu Teknoloji Fakültesi Yazılım Mühendisliği Bölümü'nde, akademik ve etik kurallara uygun olarak yazıldığını ve kullanılan tüm literatür bilgilerinin referans gösterilerek tezde yer aldığını beyan ederim.

Tuğba ÇELİKTEN

İÇİNDEKİLER

İÇİNDEKİLER I
SİMGELER VE KISALTMALAR DİZİNİ III
ŞEKİLLER DİZİNİ V
TABLO DİZİNİ VI
TEŞEKKÜR VII
ÖZET VIII
ABSTRACT IX
1. GİRİŞ 1
2. LİTERATÜR TARAMASI 3
2.1. Yazılım Gereksinim Analizi 3
2.2. Yazılım Gereksinimlerinin Sınıflandırılması 4
2.2.1. İş Gereksinimleri 4
2.2.2. Paydaş Gereksinimleri 5
2.2.3. Çözüm Gereksinimleri 5
2.2.3.1. İşlevsel Gereksinimler 6
2.2.3.2. İşlevsel Olmayan Gereksinimler 6
2.2.3.3. Alan Gereksinimleri 7
2.2.4. Geçiş Gereksinimleri 7
2.3. Gereksinim Sınıflandırması ile İlgili Yapılan Çalışmalar 8
3. GENEL BİLGİLER 13
3.1. Doğal Dil İşleme 13
3.1.1. Doğal Dil İşleme Süreçleri 16
3.2. Makine Öğrenmesi 17
3.2.1. Naive Bayes Algoritması 19
3.2.2. Multinomial Naive Bayes Algoritması 20
3.2.3. Lojistik Regresyon 20
3.2.4. Simple Lojistik 21
3.2.5. Sıralı Minimum Optimizasyon 21
3.2.6. Stokastik Gradyan Azalma 21
3.2.7. Bagging 21
3.2.8. Multiclass Sınıflandırıcı 22
3.2.9. JRip 22
3.2.10. Lojistik Regresyon Ağacı 23

3.2.11. Rastgele Orman 23
3.2.12. J48 24
3.3. Derin Öğrenme 24
3.3.1. Evrişimli Sinir Ağları 25
3.3.2. Uzun Kısa Süreli Bellek 27
3.3.3. Çift Yönlü Uzun Kısa Süreli Bellek 27
3.3.4. Geçitli Tekrarlayan Birimler 27
3.3.5. Çift Yönlü Geçitli Tekrarlayan Birimler 28
3.4. Transformer Modeller 28
3.4.1. BERT 29
3.4.2. BERTürk 30
3.4.3. DistilBERT 30
3.4.4. RoBERTa 30
3.5. Makine Öğrenmesinde Kullanılan Araçlar 31
3.5.1. Weka 31
3.5.2. Jupyter Notebook 31
4. MATERYAL VE YÖNTEM 33
4.1. Materyal 33
4.1.1. Veri Seti ve Özellikleri 33
4.2. Yöntem 35
4.2.1. Veri Ön İşleme 37
4.2.1.1. Gürültü Temizleme 37
4.2.1.2. Normalizasyon 38
4.2.2. Öznitelik Seçimi ve Özellik Çıkarımı 41
4.2.3. Modelleme 43
4.2.4. Başarım Değerlendirme Metrikleri 43
5. BULGULAR 47
5.1. Makine Öğrenmesi Deneyleri 47
5.2. Derin Öğrenme Deneyleri 49
5.3. Transformer Model Deneyleri 50
6. TARTIŞMA ve SONUÇ 52
REFERANSLAR 55

SİMGELER VE KISALTMALAR DİZİNİ

BERT Bidirectional Encoder Representations from Transformers (Çift Yönlü Kodlayıcı Temsilleri)

Bi-GRU Bidirectional Gated Recurrent Unit (Çift Yönlü Geçitli Tekrarlayan Birimler)

Bi-LSTM Bidirectional Long Short-Term Memory (Çift Yönlü Uzun Kısa Süreli Bellek)

BOW Bag of Word (Kelime Çantası)

BPSO Binary Particle Swarm Optimization (İkili Parçacık Sürü Optimizasyonu)

BRD Business Requirements Document (İş Gereksinim Dokümanı)

CBoW Continuous Bag of Words (Devamlı/Aralıksız Kelime Torbası)

CFS Correlation based Feature Selection (Korelasyon Tabanlı Özellik Seçimi)

CNN Convolutional Neural Networks (Evrişimli Sinir Ağları)

DL Deep Learning (Derin Öğrenme)

FR Functional Requirement (İşlevsel Gereksinim)

GATE General Architecture for Text Learning (Metin Öğrenme için Genel Mimari)

GNB Gaussian Naive Bayes

GR Gain Ratio (Kazanç Oranı)

GRU Gated Recurrent Unit (Geçitli Tekrarlayan Birimler)

IDE Integrated development environments (Entegre Geliştirme Ortamı)

IR Information Retrieval (Bilgi Erişim)

JAPE Java Annotation Patterns Engine (Java Ek Açıklama Modelleri Motoru)

LDARC Latent Dirichlet Allocation for Requirements Classification (Gereksinim Sınıflandırması için Gizli Dirichlet Tahsisi)

LMT Logistic Regression Tree (Lojistik Regresyon Ağacı)

LR Logistic Regression (Lojistik Regresyon)

LSTM Long Short-Term Memory (Uzun Kısa Süreli Bellek)

MCC Matthews Correlation Coefficient (Matthews Korelasyon Katsayısı)

ML Machine Learning (Makine Öğrenimi)

NB Naive Bayes

NFR Non-Functional Requirement (İşlevsel Olmayan Gereksinim)

NLP Natural Language Processing (Doğal Dil İşleme)

NLTK Natural Language Tool Kit (Doğal Dil Araç Kiti)

PIN Personal Identification Number (Kişisel Kimlik Numarası)

QP Quadratic Programming (Karesel Programlama)

RNN Recurrent Neural Network (Yinelemeli Sinir Ağı)

RoBERTa A Robustly Optimized BERT Pretraining Approach (Güçlü Bir Şekilde Optimize Edilmiş BERT Eğitim Öncesi Yaklaşımı)

ROC Receiver Operating Characteristic (İşlem Karakteristik Eğrisi)

SGD Stochastic Gradient Descent (Stokastik Gradyan Azalma)

SMO Sequential Minimal Optimization (Sıralı minimum optimizasyon)

SRC Software Requirement Classification (Yazılım Gereksinim Analizi)

SRS Software Requirement Specification (Yazılım Gereksinim Şartnamesi)

SVM Support Vector Machine (Destek Vektör Makinesi)

TF-IDF Term Frequency- Inverse Document Frequency (Terim Frekansı ve Ters Belge Frekansı)

URD User Requirements Document (Kullanıcı Gereksinim Dokümanı)

URL Uniform Resource Loader (Tekdüzen Kaynak Yükleyici)

ŞEKİLLER DİZİNİ

Şekil 1.1. Yazılım Gereksinim Belirtimi 2
Şekil 2.1. Yazılım Yaşam Döngüsü Akış Diyagramı 3
Şekil 2.2. Yazılım Gereksinimlerinin Sınıflandırılması 4
Şekil 3.1. Doğal Dil İşleme Kullanım Alanları 14
Şekil 3.2. Makine Öğrenmesi Kullanım Alanları 18
Şekil 3.3. Makine Öğrenimi Algoritmaları 19
Şekil 3.4. Ensemble Öğrenme 22
Şekil 3.5. Lojistik Regresyon Ağacı 23
Şekil 3.6. Makine Öğrenimi ve Öğrenme Mimarisi Karşılaştırılması 25
Şekil 3.7. CNN Mimarisi 26
Şekil 3.8. Transformer Modeller 29
Şekil 4.1. Veri Seti Gereksinim Tür Dağılımı 33
Şekil 4.2. Veri Setinin Eğitim-Test Veri Dağılımı 34
Şekil 4.3. Sistem Mimarisi 36
Şekil 4.4. Veri Ön İşleme Adımları 37
Şekil 4.5. Stemming 39
Şekil 4.6. Lemmatization 39
Şekil 4.7. Tokenizasyon 40
Şekil 4.8. Karmaşıklık Matrisi ve Başarım Ölçütleri 44
Şekil 4.9. ROC Eğrisi 46
Şekil 5.1. ML Algoritmaları Performans Değerleri 48
Şekil 6.1. Üç Aşamaya Ait En İyi Performans Sonuçları 52
Şekil 6.2. Tüm Algoritmaların Başarım Değerlendirmeleri 54

TABLO DİZİNİ

Tablo 4.1. Sınıflandırılmış Veri Örnekleri 34
Tablo 5.1. Makine Öğrenmesi Algoritmaları Deney Sonuçları (%) 47
Tablo 5.2. ML Algoritmaları Öznitelik Seçim ile Değerlendirme Sonuçları 48
Tablo 5.3. Derin Öğrenme Algoritmaları Deney Sonuçları (%) 49
Tablo 5.4. Transformer Modelleri Deney Sonuçları (%) 50
Tablo 5.5. Transformer Model Parameteleri 50

TEŞEKKÜR

Gerçekleştirmiş olduğum tez çalışmamın her aşamasında bir bilim insanı olarak yol gösteren, her zaman çalışmaya teşvik eden, kendisi ile çalışmaktan gurur duyduğum danışmanım Sayın Dr. Öğr. Üyesi Fatih YÜCALAR'a, tez geliştirme aşamalarında bilgi ve yardımlarını eksik etmeyen, hocam Sayın Prof. Dr. Akın ÖZÇİFT'e, tecrübeleri ve pozitif enerjisi ile destek olan Sayın Doç. Dr. Nilüfer Atman USLU hocama ve her zaman yanımda olan, yüksek lisans eğitimime başlamaya teşvik eden, rehberlikleriyle bu noktada olduğum ve başarılarımın arkasındaki en büyük güç kaynağım anne ve babama sonsuz teşekkür ederim.

Tuğba ÇELİKTEN

Manisa, 2023

ÖZET

Yüksek Lisans Tezi

Tuğba ÇELİKTEN

Manisa Celal Bayar Üniversitesi
Fen Bilimleri Enstitüsü
Yazılım Anabilim Dalı

Danışman: Dr. Öğr. Üyesi Fatih YÜCALAR

Günümüzde hızla gelişen teknoloji, yaşamımızın ayrılmaz bir parçası haline gelmiştir. Bu durum yazılım sistemlerine duyulan ihtiyacı artırmaktadır. Birçok alanda kullanılan yazılım sistemlerinin geliştirilmesinde belirli aşamalar vardır. En önemli aşamalardan biri gereksinim analizi aşamasıdır. Bu aşamada, sistemi talep eden paydaşlar gereksinimlerini belirtir. Gereksinim mühendisleri veya proje analistleri gereksinimleri amaç, işlev ve özelliklerine göre temelde işlevsel ve işlevsel olmayan gereksinimler olarak sınıflandırmaktadır. Gereksinimlerin doğru bir şekilde sınıflandırılıp, önceliklendirilmesi, yazılım yaşam döngüsünün diğer aşamalarının sorunsuz ilerlemesini sağlar. Gereksinimlerin doğal dil ile yazılıyor olması, talep sahibinin ihtiyaçlarını tam olarak aktaramaması, analiz yapacak kişilerin ilgili alana yeterince hâkim olmaması ya da gereksinimleri farklı şekilde yorumlamaları gereksinimlerin yanlış sınıflandırılmasına neden olmaktadır. Böyle bir durum yazılım geliştirme sürecine hatalı başlanılmasına -yol açmaktadır. Yazılım gereksinim analizinin otomatik olarak yürütülmesi için doğal dil işleme yöntemleri kullanılarak yazılım gereksinimleri sınıflandırılabilir. Bu amaç doğrultusunda üniversite öğrencilerinin geliştirdiği yazılım projeleri içerisinde yer alan gereksinimler toplanarak, bir Türkçe veri seti oluşturulmuştur. Toplanan gereksinimlerin 3000'ini işlevsel, 1600'ü ise işlevsel olmayan gereksinim olmak üzere manuel olarak etiketlenmiştir. %80'i eğitim, %20'si test verisi olarak ayrılan veri seti sırasıyla makine öğrenmesi algoritmaları, derin öğrenme algoritmaları ve transformer modelleri ile eğitilerek performansları test edilmiştir. Makine öğrenimi algoritmalarından Naive Bayes Multinomial, derin öğrenme algoritmalarından CNN ve transformer modellerden de BERTürk'ün en iyi performans sonuçlarını verdiği gözlemlenmiştir. Ele alınan tüm algoritmalar içinde %95 doğruluk değeri ile BERTürk en başarılı algoritma olduğu tespit edilmiştir. Elde edilen sonuçlar yazılım gereksinim sınıflandırmada geliştirilen modellerin kullanılabilirliğini açıklamaktadır.

Anahtar Kelimeler: Derin Öğrenme, Yazılım Gereksinim Analizi, Doğal Dil İşleme, Metin Sınıflandırma

2023, 60 sayfa

ABSTRACT

M.Sc. Thesis

Tuğba ÇELİKTEN

Manisa Celal Bayar University
Graduate School of Applied and Natural Sciences
Department of Software

Supervisor: Dr. Öğr. Üyesi Fatih YÜCALAR

Today, rapidly developing technology has become an integral part of our lives. This situation increases the need for software systems. There are certain stages in the development of software systems used in many fields. One of the most important stages is the requirements analysis stage. At this stage, stakeholders who request the system specify their requirements. Requirements engineers or project analysts basically classify requirements as functional and non-functional requirements according to their purpose, function, and characteristics. Accurately classifying and prioritizing requirements ensures that other phases of the software lifecycle run smoothly. However, the requirements may tend to be misclassified, if the stakeholders requesting the system cannot convey their needs clearly, the analysts do not have enough knowledge of the relevant field, or they interpret the requirements in different ways. Such a situation causes the software development process to be started incorrectly. Software requirements can be classified using natural language processing methods for the classification of software requirements automatically. For this purpose, a Turkish data set was created by collecting the requirements included in the software projects developed by university students. The collected requirements included 3000 manually labelled as functional and 1600 as non-functional requirements. The data set, of which 80% was training and 20% was test data, was trained with machine learning algorithms, deep learning algorithms, and transformer models, respectively, and their performances were tested. It was observed that Naive Bayes Multinomial from machine learning algorithms, CNN from deep learning algorithms, and BERTürk from transformer models gave the best performance results. It was found that BERTürk was the most successful algorithm with 95% accuracy among all the algorithms discussed. The obtained results explain the usability of the models developed in software requirement classification.

Keywords: Deep Learning, Software Requirement Classification, Natural Language Processing, Text Classification

2023, 60 pages

1. GİRİŞ

Teknoloji günümüzde hayatımızın her alanına dahil olmuştur ve her geçen gün hızla gelişmektedir. Bu durum teknolojiye bağımlı olmamızı da beraberinde getirmektedir. Günlük hayatta yaptığımız işlerin büyük bir kısmını teknolojinin ürünü olan yazılım uygulamaları ile yapmaktayız. Dolayısıyla bu ürünler hayatımızın bir parçası haline gelmiştir. Yazılım uygulamalarının önemi giderek arttığı için, uygulamaları geliştiren firma/kurum/kişi sayısı da artmaktadır. Bu durum aynı yazılım ürününe ait farklı versiyonların olabileceği anlamına da gelir. Bu noktada kalite faktörü yazılım ürünleri için belirleyici faktör olacaktır. Kaliteli bir yazılım projesi ortaya çıkarmak için geliştirilecek projenin ihtiyaçları doğru bir şekilde karşılaması, tutarlı ve güvenilir olması gerekmektedir. Bu aşamada yazılım geliştirme yaşam döngüsünün temeli olan gereksinim analizi aşamasının özenli bir şekilde yürütülmesi gerekmektedir. Geliştirilen projenin amacı, hedefleri, karşılayacağı ihtiyaçlar net bir şekilde belirlenmelidir. Şekil 1.1'de yazılım gereksinim belirtiminde gösterilen özellikler göz önünde bulundurularak, yazılım gereksinimi olarak ifade edilen, projenin sahip olması gereken işlevler, özellikler, kısıtlamalar gereksinim analizi aşamasında detaylı bir şekilde ele alınmalıdır. Yazılım proje geliştirmede kritik öneme sahip olan gereksinim analizi aşaması genellikle gereksinim mühendisleri tarafından yürütülen, zor ve zaman alıcı bir süreçtir. Gereksinimlerin belirlenip işlevsel ve işlevsel olmayan gereksinimler şeklinde sınıflandırılması gerekmektedir. Ayrıca işlevsel olmayan gereksinimler de kendi içinde kullanılabilirlik, performans, güvenilirlik vb. sınıflara ayrılmaktadır.

Gereksinim belirleme aşamasında gereksinim mühendisleri ya da geliştiriciler proje alanına hâkim olmayabilir. Özellikle gereksinim türlerinden biri olan alan gereksinimleri genellikle müşteri ve proje kullanıcılarının uzmanlık alanıdır. Bu durumda proje gereksinimleri kuruluşların hedef ve stratejileri göz önünde bulundurularak, müşteriler ile birlikte analiz edilmelidir. Sistemde olması gereken gereksinimler ortaya koyulurken, bu gereksinimlerin uygunluğu, diğer gereksinimler ile tutarlılığı ve geliştirme ve sistemin kullanım aşamalarında öngörülebilecek durumlar detaylı bir şekilde ele alınmalıdır. Ayrıca zaman ve insan kaynağı da dikkate alınarak gereksinimler önceliklerine göre sıralanmalıdır. Geliştirilen yazılım uygulaması için kullanıcı rolleri belirlenmelidir. Kullanıcıların rollerine göre sistem

fonksiyonlarını içeren kullanım durumları için senaryolar oluşturulmalı ve use-case diyagramları ile bu akış gösterilmelidir. Ayrıca işlevsel ve işlevsel olmayan gereksinimler (ve alt grupları) sınıflandırıldıktan sonra use-case diyagramları üzerinden örnek senaryolar ile test yapılmalı, sistem işlevleri/fonksiyonları kullanım durumlarını tetiklenecek olaylar gözden geçirilmelidir.



Şekil 1.1. Yazılım Gereksinim Belirtimi

Yazılım projesinin iskeletini oluşturan gereksinim analiz süreci, gereksinimlerin belirlenmesi, önceliklendirilmesi ve sınıflandırılması özellikle büyük çaplı projelerde oldukça zor ve hataya açık bir görevdir. Bu aşamada gereksinimlerin açık ve net bir şekilde belirlenmesi hatta paydaşlar ile mutabık kalarak tüm gereksinimlerin belgelenmesi, yazılım yaşam döngüsünün tasarım, kodlama, test ve bakım evrelerinde zaman kaybı, maliyet ve paydaşlar arasında anlaşmazlık gibi büyük sorunların önüne geçecektir.

2. LİTERATÜR TARAMASI

Bu bölümde, yazılım gereksinimleri, yazılım gereksinim analizi, yazılım gereksinim analizi yaparken yaşanan zorluklar ve yazılım gereksinim sınıflandırma ile ilgili yapılan çalışmalara yer verilmiştir.

2.1. Yazılım Gereksinim Analizi

Gereksinim, bir sistemin gerçekleştirilmesi için gereken prosedürler, durumlar, sistem girdileri ve çıktıları olarak tanımlanabilir. Bir yazılım ürünü geliştirmek için müşterinin üründen beklediği tüm gereksinimler toplanır. Şekil 2.1'de gösterilen yazılım yaşam döngüsü akış diyagramında, gereksinim analizi aşamasında bu gereksinimler değerlendirilir. Özelliklerine göre kategorilerine ayrılır ve analiz tamamlandığında tüm paydaşların ortak kararla belirlemiş olduğu bu gereksinimler sözleşme niteliğinde belgelenir.



Şekil 2.1. Yazılım Yaşam Döngüsü Akış Diyagramı

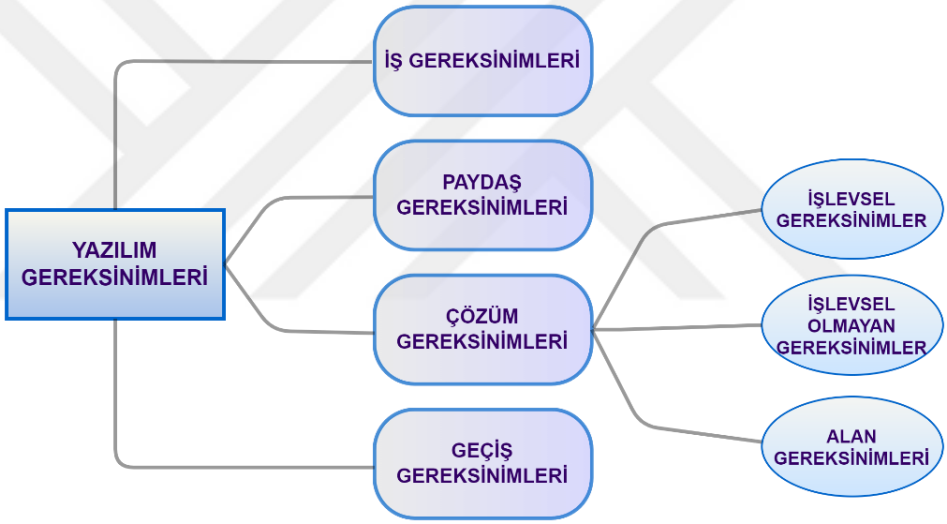
Gereksinim analizi aşaması; yazılım geliştirme sürecinin temeli olan tüm yazılım gereksinimlerinin belirlenip, ayrıntılı bir biçimde detaylandırılıp işlevlerine göre sınıflandırıldığı ve önceliklendirildiği önemli bir aşamadır. Hassas ve özenle ele alınması gereken bir görevdir. Çünkü bu aşamada yapılan bir hata, öngörülemeyen bir sistem ihtiyacı ya da yanlış sınıflandırma, sistemin baştan tasarlanmasına, sistemin yapması gereken görevlerin eksikliğine, belirsizliğine veya paydaşlar arasında çatışmalara neden olabilir. Gereksinim analizi süreci, gereksinim mühendisleri ve geliştiriciler için zorlu, zaman alıcı, uzmanlık, deneyim ve alan bilgisi gerektirir.

Gereksinim sınıflandırma görevinin manuel olarak yapılması özellikle gereksinimlerin çok olduğu büyük projelerde daha fazla insan çabası ve zaman

gerektirmektedir. Bu durumlarda da hata yapma olasılığı oldukça yüksektir. Bu sebeplerle yazılım gereksinimlerinin otomatik sınıflandırılması insan emeği, harcanan zamanın azaltılması, zihinsel yorgunluğun önlenmesi, maliyet vb. sistem kaynaklarının verimli bir şekilde kullanılması için faydalı bir çözüm olacaktır [1, 2].

2.2.Yazılım Gereksinimlerinin Sınıflandırılması

Bir yazılım projesi içerisinde yer alabilecek gereksinimler iş, paydaş, çözüm ve geçiş gereksinimleri olmak üzere dört farklı biçimde sınıflandırılabilir. Çözüm gereksinimleri de kendi içerisinde işlevsel, işlevsel olmayan ve alan gereksinimleri olmak üzere üç alt sınıfa ayrılmaktadır.



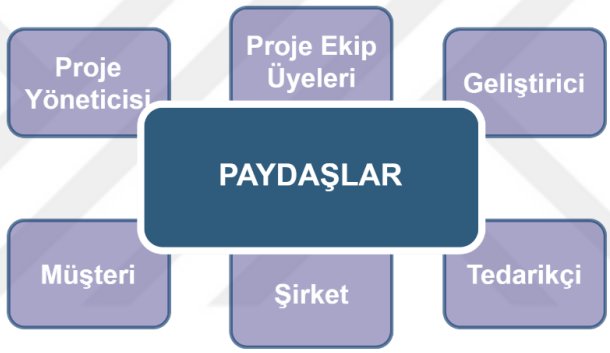
Şekil 2.2. Yazılım Gereksinimlerinin Sınıflandırılması

2.2.1. İş Gereksinimleri

İş gereksinimleri (business requirements) yazılım ürününün neden geliştirildiğini, amaç ve hedeflerini açıklar. İş gereksinimleri, tipik olarak, yazılımın geliştirilmesini talep eden müşteri veya kuruluşun amaçlarına göre belirlenir [3]. Bu nedenle müşterinin/firmanın organizasyon stratejileri ile aynı çizgide olması gerekmektedir. Ayrıca gereksinim analizi sürecinde belirlenen iş gereksinimleri "İş Gereksinim Dokümanı (BRD)" şeklinde belgelenir ve sistem dokümanları takip edilerek geliştirilir.

2.2.2. Paydaş Gereksinimleri

Paydaş gereksinimleri (stakeholder requirements), sistemin kullanıcılarına hangi hizmetleri vereceğini ifade etmektedir. Bu hizmetlerin hangi olası kısıtlamalar veya sınırlamalar altında çalışması gerektiğini açıklar. Paydaş gereksinimleri aynı zamanda kullanıcı gereksinimleri olarak da bilinir; geliştirilecek yazılım sisteminin özelliklerini, sistemin hangi işlevleri barındırdığını, Şekil 2.3'te gösterilen paydaşların sistem içindeki rollerini ve kısıtlamalarını içerir. Adından da anlaşılacağı gibi, kullanıcı gereksinimleri kullanıcıya yönelik oluşturulan gereksinimlerdir. Bu sebeple kullanıcı gereksinimleri doğal dil ile yazılmış, teknik detay içermeyen, sistem işlevlerini diyagramlar ile anlatan metinlerdir.



Şekil 2.3. Proje Paydaşları

Paydaş gereksinimlerinin analiz edilmesi, gereksinimlerin yazılım gereksinimleri belirtimi olarak belgelenmesi; belgelenmiş gereksinimlerin paydaşların gereksinimleriyle eşleştiğinin doğrulanması ve gereksinimlerin gelişiminin yönetilmesinin doğru bir biçimde yapılması gerekmektedir [3]. Kullanıcı gereksinimleri "Kullanıcı Gereksinim Dokümanı (URD)" şeklinde belgelenerek paydaşlar ile paylaşılır.

2.2.3. Çözüm Gereksinimleri

Çözüm gereksinimleri (solution requirements), geliştirilecek yazılım sisteminde olması gereken ihtiyaçları (durum, davranış vs.) karşılayacak gereksinimlerdir. Çözüm

gereksinimleri işlevsel, işlevsel olmayan ve alan gereksinimleri olmak üzere üç alt gereksinim türü olarak sınıflandırılmaktadır.

2.2.3.1. İşlevsel Gereksinimler

İşlevsel gereksinimler (functional requirements), geliştirilecek sistemin belirli koşullar için belirli girdilerle ya da belirli durumlarda nasıl davranması gerektiğini detaylı bir şekilde belirtir. Yazılım işlevselliğini tanımlayan işlevsel gereksinimler, kullanıcıların görevlerini yerine getirmelerine ve böylece iş gereksinimlerini karşılamalarına olanak tanımaktadır [3].

2.2.3.2. İşlevsel Olmayan Gereksinimler

İşlevsel olmayan gereksinimler (non-functional requirements – NFR'ler), sistem tarafından sunulan hizmetler veya işlevler üzerindeki kısıtlamaları, kullanıcının iş gereksinimlerini karşılamak için yazılım ürünlerinin sahip olması gereken nitelikleri ifade eder. Ana kaynağı, hala doğal dilde yazılmış, programlama veya zamanlama kısıtlamalarını, geliştirme ve uygulama sürecindeki kısıtlamaları içeren belgelerdir. Programlama veya zamanlama kısıtlamalarını, geliştirme ve uygulama sürecindeki kısıtlamaları içerir. NFR'lerin açıklaması genellikle farklı cümlelerde veya paragraflarda gizlenmiş, dağınık şekildedir, bu nedenle gereksinimler belgesinde gözden kaçırılmaları kolaydır. Doğal dilde yazılan NFR'ler, yazılım mühendislerinin geliştirilecek yazılım için neyin gerekli olduğunu yanlış anlamalarına yol açacak şekilde belirsiz, muğlak ve/veya belirsiz olabilir. Yazılım mimarları, analistler ve geliştiriciler, NFR'leri gereksinim belgelerinden analiz etmeyi, anlamayı ve sınıflandırmayı her zaman zorlayıcı ve zaman alıcı bulmaktadır çünkü bu görevlerin tümü uzmanlık, eğitim, deneyim ve alan bilgisi gerektirir. Farklı eğitim geçmişine, kültüre veya deneyime sahip kişiler aynı NFR'leri farklı şekilde anlayabilir. Bu nedenle, NFR'leri gereksinim belgelerinden manuel olarak sınıflandırmak, karmaşık sistemler ve büyük miktarda metin söz konusu olduğunda mümkün değildir. Şekil 2.4'te gösterildiği gibi sistem çalışmasını engellemeyen ama sistem bütünlüğünü sağlayan özellikteki gereksinimler işlevsel olmayan gereksinimlerdir. İşlevsel gereksinimler ile işlevsel olmayan gereksinimler arasındaki fark, işlevsel olmayan

gereksinimlerde kısıtlamalar fazla olduğu için tasarım ve test aşamasında genellikle daha zorlayıcıdır [1, 3].



Şekil 2.4. İşlevsel Olmayan Gereksinim Türleri

2.2.3.3.Alan Gereksinimleri

Alan gereksinimleri (domain requirements), geliştirilecek sistemin kullanım alanına özgü gereksinimleridir. Genellikle ürün sahibinin ürünü kullanacağı alan ile ilgili gereksinimlerdir. Yazılım gereksinim türü olmasına karşın teknik bir gereksinim değildir. Ürün sahibi ve alana hâkim olan kullanıcı veya müşteriler tarafından belirlenir. Bu gereksinimler, genellikle alana özgü terimler içerdiğinden, alan gereksinimlerinin geliştiriciler tarafından anlaşılması zor bir durumdur.

2.2.4. Geçiş Gereksinimleri

Geçiş gereksinimleri (transition requirements), genellikle oluşturulan sistemin bir durumdan farklı bir duruma aktarılması, farklı bir platforma taşınması ya da farklı bir sistem ile entegrasyonunda gerekli olan gereksinimlerdir. Bu gereksinimlerin analiz sürecinde belirlenmesi, yeni bir çözüm geliştirilmesi ya da sistemin farklı bir duruma geçmesinde karşılaşılacak engelleri veya projenin olumsuz etkilenmesini en aza indirmeyi sağlar.

2.3.Gereksinim Sınıflandırması ile İlgili Yapılan Çalışmalar

Li ve ark. [1] tarafından insan emeğini, harcanan zamanı azaltmak ve zihinsel yorgunluğu önlemek için işlevsel olmayan gereksinimlerin gereksinim belgelerinden otomatik olarak çıkarılması için NFRNet adlı yeni bir derin sinir ağı modeli sunan bir çalışma yapılmıştır. Çalışmada, yazılım gereksinim sınıflandırma araştırmalarında yaygın olarak kullanılan bir veri seti olan PROMISE veri seti kullanılmıştır. PROMISE veri seti içinde bulunan 11 kategoriye ait işlevsel olmayan gereksinimler 32 kategori ve 255 işlevsel olmayan gereksinim açıklama ifadeleri 6222 ifade olarak arttırılmıştır. Geliştirdikleri NFRNet sinir ağı iki bölümden oluşmaktadır. Biri, gereksinim açıklamasının bağlam temsilini öğrenmek için N-gram maskelemeye dayalı bir BERT kelime gömme modeli, diğeri ise bir Bi-LSTM sınıflandırma ağıdır. Son olarak, gereksinim açıklamasını sınıflandırmak için Softmax sınıflandırıcısı kullanılmıştır. Modelin eğitim sürecinde, eğitim için gereken yineleme sayısını azaltabilen, derin sinir ağlarının eğitimini hızlandırabilen ve eğitilen ağlarda daha düşük hata oranları elde edebilen çoklu örneklem seyreltme (multi-sample-dropout) adı verilen yeni bir düzenleme tekniği uygulanmıştır. Önerilen modelin sınıflandırma doğruluğunu değerlendirmek için SOFTWARE NFR veri setinde 10 kat çapraz doğrulama yapılmış ve elde ettikleri sonuçlara göre NFRNet modeli, kullandıkları diğer modeller içinde %91 precision, %92 recall ve %91 F1-skoru ile en yüksek performansa sahip model olmuştur [1].

Bisi ve Keskar'ın yaptıkları çalışmada [2] yazılım gereksinimlerini işlevsel ve işlevsel olmayan olarak sınıflandırmak için bir CNN modeli önerilmiştir. CNN, tahmin performansını etkileyen filtre boyutu, filtre sayısı, girdi yerleştirme boyutu ve CNN mimarisi gibi birkaç hiper parametreye sahiptir. Çalışmanın amacı İkili Parçacık Sürü Optimizasyonu (BPSO) yaklaşımını kullanarak, daha iyi tahmin doğruluğu için CNN parametrelerini optimize etmektir. Çalışmada veri seti olarak, etiketli 538 işlevsel ve işlevsel olmayan gereksinim içeren PROMISE veri seti kullanılmıştır. Öncelikle yazılım gereksinimlerini işlevsel ve işlevsel olmayan olarak sınıflandırmak için bir CNN modeli tasarlanmış, ardından kelime çantası (BOW) tekniği ve Wikipedia kullanılarak önceden eğitilmiş veri seti, ön işleme için, yani metin verilerini sayısal verilerin vektörüne dönüştürmek için kullanılmıştır. Son olarak BPSO kullanarak CNN hiper parametrelerini optimize etmek için CNN-BPSO modelini geliştirilmiştir.

Yapılan deney sonuçlarına göre, yazılım gereksinim sınıflandırmasında %80 eğitim ve %20 test verisi için önerdikleri CNN-BPSO yaklaşımı %81 doğruluk değeri ile %79 doğruluk değerine sahip CNN modelinden daha iyi performans göstermiştir [2].

Navarro ve ark.'nın yaptığı çalışmada [4], yazılım gereksinim sınıflandırması (SRC) görevinin özellikle büyük gereksinimler olduğunda çok fazla insan çabası gerektirmesi sorununa çözüm sunmak için, SRC'nin otomasyonu, Doğal Dil İşleme (NLP) ve Bilgi Erişim (IR) teknikleri kullanılarak ele alınmıştır. Bu çalışmada, yoğun emek gerektiren özellik mühendisliği olmadan yazılım gereksinimlerini sınıflandırmak için Derin Öğrenmeyi (DL) yaklaşımı kullanılmıştır. Önerdikleri model, Evrişimli Sinir Ağına (CNN) dayanmaktadır. Veri seti olarak işlevsel ve 11 farklı işlevsel olmayan gereksinim kategorilerinde bir dizi etiketli gereksinim içeren PROMISE korpusu kullanılmıştır. Kelime temsil yöntemi olarak skip-gram ile word2vec yöntemi kullanılmıştır. Önerilen modeli değerlendirmek için kullanılan precision, recall ve f1-skor metrikleri sırasıyla 0.80, 0.785 ve 0.77 olarak ölçülmüştür. Önerilen modelin, precision, recall ve f1-skorundaki düşük standart sapmayla (<0.035) iyi bir kararlılık sunduğu gösterilmiştir [4].

Singh ve ark. yaptıkları çalışmada [5], yazılım ürününün kalitesini artırmak için kural tabanlı sınıflandırıcı kullanarak NFR ve ISO 9126 kalite modelindeki alt sınıflarını sınıflandırmak için bir teknik geliştirmeye ve NFR'lerin alt sınıflarına ayrıntılı olarak tanımlanmasına odaklanmışlardır. Bu çalışmada, 265 FR ve 370 NFR ile 635 cümleye sahip 15 SRS dokümanı içeren PROMISE veri seti kullanılmıştır. Ek olarak, NFR'lerin sınıflandırılması, Java Ek Açıklama Modelleri Motoru (JAPE) kullanılarak her bir cümlenin tematik roller tarafından açıklandığı Metin Öğrenimi için Genel Mimari (GATE) ile gereksinim tematik rollerinin analizine dayanır. Kural tabanlı sınıflandırıcı ile yapılan sınıflandırma sonucunda 8 NFR alt sınıfına (doğruluk, uygunluk, güvenlik, işlevsellik, anlaşılabilirlik, tasarım, zaman davranışı, kaynak kullanımı) ait ortalama F1-skoru %97.21 olarak ölçülmüştür [5].

Halim ve Siahaan tarafından yapılan çalışmada [6] ise, doğal dillerde yazılmış yazılım gereksinimlerinde atomik olmayan gereksinimleri tespit edebilen bir model geliştirilmiştir. Atomik olmayan gereksinimler, sistemin yalnızca bir işlevinin olmadığı bir gereksinim bildirimidir. Bir sistem işlevi, özellikler, gereksinimler,

yetenekler biçiminde tam olarak tanımlanabiliyorsa, bu gereksinimin atomik olduğu söylenebilir. Bir atomik gereksinim, işlevsel veya işlevsel olmayan gereksinim olabilir. Çeşitli online kaynaklardan topladıkları gereksinimler, atomik ifadeleri içeren $Corpus_a$ ve atomik olmayan gereksinimleri içeren $Corpus_n$ olmak üzere iki korpusa ayrılmıştır. Veri seti, $Corpus_a$ (atomik) 404 gereksinim ifadesinden ve $Corpus_n$ (atomik olmayan) 196 gereksinim ifadesinden oluşan 600 gereksinim bildiriminden oluşturulmuştur ve sınıflandırma için Bayes Net, Random Forest, Multilayer Perceptron makine öğrenimi algoritmaları kullanılmıştır. Bu araştırmada en iyi model %84,25 doğru sınıflandırma oranı ile Bayes Net algoritması ile üretilmiştir. Modelin güvenilirliği, yazılım gereksinimleri belirtiminde atomik olmayan gereksinimlerin saptanmasında dengesiz veriler için makul kabul edilmiştir. Atomik olmayan gereksinimlerin saptanmasında denge verileri için modelin güvenilirliği orta düzeyde kabul edilmiştir. Modeli test etmek için 3 farklı uzman değerlendirmeleri ve önerilen model sonuçları, Cohen Kappa güvenilirlik ölçümü kullanılarak karşılaştırılmıştır. Ortalama olarak, önerilen model en yüksek güvenilirlik oranına sahiptir (0.49) [6].

Alhindawi'nin yaptığı çalışmada [7] bir dizi paralel ve sıralı adıma dayanan Gereksinim Sınıflandırması için Gizli Dirichlet Tahsisi (Latent Dirichlet Allocation for Requirements Classification – LDARC) adı verilen bir yaklaşım sunulmuştur. Sunulan yaklaşım, sınıflandırma ve haritalama sürecini gerçekleştirmek için Gizli Dirichlet Tahsisi (Latent Dirichlet Allocation – LDA) yöntemini kullanır. LDA kullanılarak, bir derlem bir dizi konu ile temsil edilebilir ve derlemdeki her bir belge de bu konulardan birden fazlası ile tanımlanabilir. Ayrıca derlemdeki her terim bu konuların birden fazlasında yer alabilir. Bu nedenle herhangi bir derlem belgesi, tek bir konuyla ilişkilendirilmekle sınırlı kalmayıp, alternatif olarak, belgenin birden çok konuyu ele alma olasılığını göz önünde bulundurarak modellenir [7]. Önerilen yaklaşımı doğrulamak için, Web ARChive sisteminden 200 gereksinimden oluşan bir dizi seçilmiş ve bu gereksinimler için bir derlem oluşturulmuştur. Derlemdeki her belge bir gereksinim mesajı açıklamasını temsil etmektedir. LDA her belgeye uygulanır. Yapılan sınıflandırma sonuçlarını doğrulamak için üç farklı deney gerçekleştirilmiştir. İlk deney, gereksinimi işlevsel ve işlevsel olmayan olarak sınıflandırmak (LDA kullanılarak yapılmıştır). İkinci deney, işlevsel ve işlevsel olmayan gereksinimleri karşılık gelen sistem gereksinimleriyle eşleştirmek ve üçüncü

deney, yeni alan gereksinimini işlevsel ve işlevsel olmayan olarak sınıflandırmaktır. Sonuçlar, sunulan yaklaşımın gereksinimleri verimli bir şekilde sınıflandırabildiğini ve haritalayabildiğini göstermiştir [7].

Limaylla-Lunarejo ve ark. [8], bir İspanyol veri kümesinde gereksinim sınıflandırması için ML algoritmalarıyla metin vektörleştirme tekniklerinin hangi kombinasyonlarının en iyi performansı gösterdiğini araştırmıştır. Bu araştırma için veri seti, A Coruna Üniversitesi'nde yürütülen 15 projeden çıkarılan 300 işlevsel ve 89 işlevsel olmayan gereksinim olmak üzere toplam 389 gereksinimden oluşmaktadır. Yapılan çalışma; ön işleme, metin vektörleştirme, ML seçimi, eğitim aşaması ve test aşaması olmak üzere beş adımdan oluşmaktadır. Çalışmada vektörleştirme tekniği olarak BOW ve TF-IDF ve n-gramlarla kombinasyon halinde NB, GNB, LR ve SVM algoritmalarına odaklanılmaktadır. Ayrıca kelime gömme teknikleriyle birleştirilmiş CNN, BETO derin öğrenme algoritmaları kullanılmıştır. BETO, bir İspanyol korpusuna dayalı BERT'li Transformers kullanan önceden eğitilmiş bir modeldir. TF-IDF ve Bigram/Trigram tekniği ile birlikte SVM, fonksiyonel sınıf için %95 oranı ile en yüksek F1 skorunu elde etmiştir. LR, CNN ve BETO algoritmalarında %93 ve NB ve GNB algoritmalarında ise %92 F1-skoru elde edilmiştir. İşlevsel olmayan gereksinimlerin sınıflandırılmasında daha düşük performansa ulaşılmıştır. Bu sınıf için en yüksek %79 F1 puanı ile TF-IDF ve Bigram/Trigram kombinasyonu ile SVM tarafından elde edilmiştir. Diğer modeller %76'nın altında F1-skor değerleri elde ederken, GNB en düşük değeri (%65) elde etmiştir [8].

Haque ve ark. tarafından ele alınan çalışmada [9], makine öğrenimi özellik çıkarma ve sınıflandırma tekniklerini birleştirilerek kaliteli yazılım geliştirme için otomatik NFR sınıflandırma yaklaşımı önerilmiştir. En iyi çifti bulmak için NFR'yi otomatik olarak sınıflandırmak üzere yedi makine öğrenimi algoritması ve dört özellik seçme yaklaşımı içeren ampirik bir çalışma uygulanmıştır. Veri seti olarak 62 gereksinim cümlesinden oluşan küçük boyutlu PROMISE veri seti kullanılmıştır. Bu cümlelerin 255'i işlevsel, 370'i işlevsel olmayan gereksinimlerdir. NFR, kullanılabilirlik, yasal, görünüm ve his, sürdürülebilirlik, operasyonel, performans, ölçeklenebilirlik, güvenlik, kullanılabilirlik, hata toleransı ve taşınabilirlik olarak 11 kategoriyle etiketlenmiştir. Bununla birlikte, veri kümesi, sınıf dağılımının dengesiz olduğunu gösteren çok sayıda işlevsel gereksinim içerir. BoW, TF-IDF (karakter

seviyesi), TF-IDF (kelime seviyesi) ve TF-IDF (n-gram) teknikleri kullanılarak veri seti önceden işlenmiştir. Daha sonra, sınıflandırıcıyı eğitmek için her bir öznitelik vektörü türü için MNB, GNB, BNB, KNN, SVM, SGD SVM ve DTree algoritmaları uygulanmıştır. Son olarak, her yöntemin sınıflandırma performansı, farklı NFR sınıflandırma sonuçları kullanılarak karşılaştırılmıştır. Ortalama sonuçlar precision, recall ve F1 puanı, TF-IDF (karakter seviyesi) için daha yüksek olmuştur. Burada SGD SVM, diğer algoritmalara kıyasla en iyi sınıflandırma sonuçlarını göstermiştir. Presicion, recall, F1 skorunu ve accuracy metrikleri sırasıyla %66, %61, %612 ve %75 olarak ölçülmüştür. Bununla birlikte, Dtree tüm durumlar için en kötü NFR sınıflandırma performansına sahiptir ve presicion, recall, F1 skorunu ve accuracy metrikleri sırasıyla %17, %17, %14 ve %18 değerlerini almıştır [9].

Quba ve ark. tarafından yapılan çalışmada [10], yazılım gereksinimleri belirtimindeki metin verilerini temsil etmek için makine öğrenimini kullanarak yazılım gereksinimlerini otomatik olarak sınıflandırmaya yönelik işlevsel ve işlevsel olmayan gereksinimleri gruplandırmak amacıyla bir teknik önerilmiştir. Veri seti olarak etiketlenmiş gereksinimleri içeren PROMISE veri setinin genişletilmiş hali olan PROMISE_exp kullanılmıştır. BoW tekniği ile SVM ve KNN algoritmaları, NFR ve FR ve NFR alt sınıflarının sınıflandırması için kullanılmıştır. Tahmin sonuçları başarım değerlendirme metriklerinden precision, recall ve F1 puanı ile değerlendirilmiştir. Yapılan çalışma sonucuna göre, SVM ve BoW kullanımı, ikili sınıflandırmada (FR veya NFR) %90'lık bir F1 puanı ile KNN ve BoW ikilisinden daha iyi performans göstermiştir. Ayrıca SVM ve BoW, işlevsel olmayan gereksinimlerin 11 alt kategori sınıflandırmasında %66 ve 12 alt kategori sınıflandırmasında %72 F1 puanına ulaşarak KNN ve BoW ikilisinden daha iyi performans göstermiştir. Elde edilen çalışma sonuçlarına göre; yazılım gereksinimlerinin otomatik olarak sınıflandırılması için BoW tekniklerinin SVM algoritmalarıyla birlikte kullanılmasının başarılı olduğu sonucuna ulaşılmıştır [10].

3. GENEL BİLGİLER

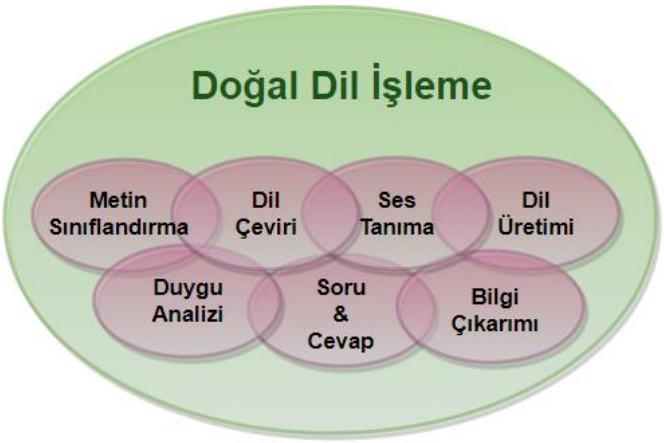
Genel bilgiler bölümü altında çalışmada kullanılan algoritmalar, teknikler ve araçlar hakkında genel bilgi verilmiştir.

3.1. Doğal Dil İşleme

Teknolojinin hızla gelişmesi, sosyal medya kullanımının artması, bilgiye ulaşımın kolaylaşması vb. etkenler ile üretilen veri miktarı her geçen gün hızla artmaktadır. Veri miktarının artması ile birlikte verinin işlenmesine duyulan ihtiyaç da artmaktadır. Örneğin büyük firmalar, veritabanlarında topladığı bilgiler ile firmanın gelişmesine katkı sağlayacak analizler yapmak isteyebilir. Bir başka örnek, bir telekominikasyon şirketi, sağladığı hizmetler, bu hizmetler karşılığında müşterilerin memnuniyet ve tutumlarını değerlendirmek isteyebilir. Bu durumda elde ettiği tüm verileri yorumlayarak durum ile ilgili önlemler almak ya da yeni hizmetler ortaya çıkarmak isteyebilir. Veri miktarının artması, veri analiz süreçlerinin de zorlaşmasına sebep olmaktadır. Yaşanan bu zorlukların üstesinden gelmek ve verileri daha fazla alanda işlemek için çeşitli yöntemler geliştirilmektedir. Doğal Dil İşleme (Natural Language Processing – NLP) bu yöntemlerden biridir.

NLP, insanların konuştuğu doğal dilin, makineler tarafından anlaşılabilir ve yorumlanabilir bir şekle dönüştürülmesi olarak tanımlanabilir. Sözlü veya yazılı metinsel ifadelerden bir yapı oluşturmayı amaçlar. Oluşturulan bu yapılar ise metni analiz etmek için yapay zeka temelli yöntemlere bir altyapı sağlar [11]. Kısacası doğal dil işleme, sözlü ya da metinsel verilerin bilgisayarlar tarafından hesaplanabilir ve anlamlandırılabilir formata dönüştürülmesi olarak ifade edilebilir. Şekil 3.1'de görüldüğü gibi günümüzde sağlık, bankacılık, telekominikasyon, eğitim gibi verinin olduğu her alanda farklı ihtiyaçlar için doğal dil işleme kullanılmaktadır. Ayrıca artan veri ile birlikte verinin yayılımı da hızlanmıştır. Verinin hızlı yayılması ile beraber, kaynağı ve doğruluğu kesin olmayan, eksik ya da yanlış verinin yayılması sorunu da ortaya çıkmıştır. Bu durum da yayılan bilginin doğruluğunu kontrol etme zorunluluğunu doğurmuştur. Bugünün dünyasında, internet ortamında günlük akan veri miktarının devasa boyutlara ulaşması göz önünde bulundurulursa verinin doğru bir şekilde yayılımını kontrol etme sürecinin insanlar tarafından kontrolünün pek de

mümkün olmayacağı açıkça görülmektedir. Yapay zekanın alt alanlarından biri olan doğal dil işleme kapsamında yapılan çalışmalardan biri de bu sorunun önüne geçmektir.



Şekil 3.1. Doğal Dil İşleme Kullanım Alanları

A. Metin Sınıflandırma

Metin sınıflandırma ya da kategorize etme bir doğal dil metninde bulunan içeriği belirli sınıflara otomatik olarak ayırma işlemidir. Doğal dil işlemenin temel alanlarından biridir. E-mail olarak gelen bir mesajın spam olup olmadığı, haber sitesinde bulunan bir haberin hangi kategoriye ait olduğu gibi sınıflandırmalar metin sınıflandırmasına örnek olarak verilebilir.

B. Duygu Analizi

Duygu analizi, insanların bir varlığa ya da duruma karşı görüşlerinin analiz edilmesi olarak tanımlanır. Amaç bir metin dokümanı içerisinden ifade edilen duyguları belirlemek ve metinleri bu duygulara göre sınıflandırmaktır [12]. Müşterilerin bir ürüne yaptığı yorumlardan yola çıkarak, müşteri memnuniyetinin belirlenmesi, bir film için yapılan yorumların pozitif ya da negatif oluşunun tespiti gibi durumlar doğal dil işleme alanında duygu analizi altında incelenmektedir. Duygusal durum ve değerlendirme teorisi çalışmalarından yola çıkan bu durum son zamanlarda doğal dil işleme alanında da çok fazla çalışma yapılan bir alan haline gelmiştir [13].

C. Dil Üretimi

Dil üretimi, bilişsel bilimi de içeren doğal dil işleme altındaki çalışma alanlarından biridir [14]. En basit tanımı ile bir yapay zekâ modeline verilen veri setini kullanarak, dil kalıplarının çıkarılıp otomatik olarak doğal dilde metinlerin oluşturulmasıdır.

D. Bilgi Çıkarımı

Metin madenciliğinde önemli çalışma alanlarından biri olan bilgi çıkarımı yapılandırılmamış veya yarı yapılandırılmış metinlerden bilgi elde etmek olarak ifade edilir [15]. Bir diğer tanım olarak varlıklar, varlıklar arasındaki ilişkiler ve varlıkları tanımlayan nitelikler gibi yapılandırılmış bilgilerin yapılandırılmamış kaynaklardan otomatik olarak çıkarılmasını ifade eder. Bilgi çıkarımı bol miktarda yapılandırılmamış kaynaklarda, yalnızca anahtar kelime aramalarıyla mümkün olandan çok daha zengin sorgu biçimleri sağlar. Yapılandırılmış ve yapılandırılmamış veriler bir arada bulunduğunda, bilgi çıkarma, iki tür kaynağı entegre etmeyi ve bunları kapsayan sorgular oluşturmayı mümkün kılar. Bilgi çıkarımı için metindeki varlıklar, anahtar ifadeler ve ilişkiler tanımlanır. Gürültülü, yapılandırılmamış kaynaklardan bilginin çıkarılması, uzun süredir zorlu bir görev devam etmektedir. Makine öğrenimi, bilgi alma, veri tabanı, web ve belge analizini kapsayan birçok farklı alan için kullanılmaktadır [16].

E. Ses Tanıma

Günümüzün akıllı dünyasında, ses tanıma birçok alanda önemli bir konu haline gelmiştir. Konuşmanın metin haline dönüştürülmesi olarak tanımlanmaktadır. Sesle etkinleştirilen bankacılık, ev otomasyonu ve sesle etkinleştirilen cihazlar, ses tanımanın birçok kullanımından yalnızca birkaçıdır [17]. Özellikle güvenlik sistemlerinde, çalınma ya da kopyalanma ihtimaline karşı şifre, kullanıcı adı, PIN kodu yerine son zamanlarda ses tanıma sistemleri daha popüler hale gelmiştir [18]. Çünkü ses, kullanıcılar için doğal olmasının yanı sıra bazı geleneksel biyometrik sistemlerle

karşılaştırıldığında daha yüksek olmasa da benzer güvenlik seviyeleri sunan bir ölçümdür [17].

F. Soru Cevaplama

Doğal dil işlemenin sık kullanıldığı alanlardan biri de soru cevaplama sistemleridir. İnsan türevi teknikler ile sanal bir asistanın sorulan sorulara cevap üretmesine dayanır. Sistem kendisine sorulmuş olan soruları ve bu sorular arasındaki ilişkileri, dahası soru metnini kelime bazında çözümleyerek optimal cevaplar üretir. Özellikle müşteri hizmetleri, telekominikasyon sektörlerinde kullanılan çözümlerden biridir.

G. Dil Çeviri

Diller arası çeviri işlemleri doğal dil işleme alanında zorlu görevlerden biridir. Tümce öğe düzeni her dilde aynı olmadığından, kelime kelime çeviri tam bir sonuç vermeyebilir ya da çeviriden sonra metni hedef dildeki karşılığında anlaşılırlığı daha net hale dönüştürmek için manuel düzenleme gerekebilir. Doğal dil işlemede, kelimeler türüne, cümle içindeki sırasına, cümlelerin birbiriyle anlamsal ilişkisine göre analiz edilmekte ve çeviri işlemi çok daha başarılı bir şekilde gerçekleşmektedir.

3.1.1. Doğal Dil İşleme Süreçleri

Geliştirilecek doğal dil modellerinde kullanılacak veri setinin anlaşılır, düzenli ve makinelerin anlayacağı şekilde olması gerekmektedir. Bu sebeple makine öğrenmesi, derin öğrenme, doğal dil işleme süreçlerinin başarılı bir şekilde tamamlanabilmesi için veri setine birtakım işlemlerin uygulanması gerekmektedir. Bu işlemlere;

- Noktalama işaretlerinin, özel karakterlerin veri setinden kaldırılması,
- URL, emoji gibi ifadelerin veri setinden kaldırılması,
- Büyük harflerin küçük harflere dönüştürülmesi,
- Tekrar eden kelimelerin kaldırılması,
- Gereksiz, anlamda bozulmaya sebep olmayacak edat ve bağlaçların veri setinden kaldırılması,

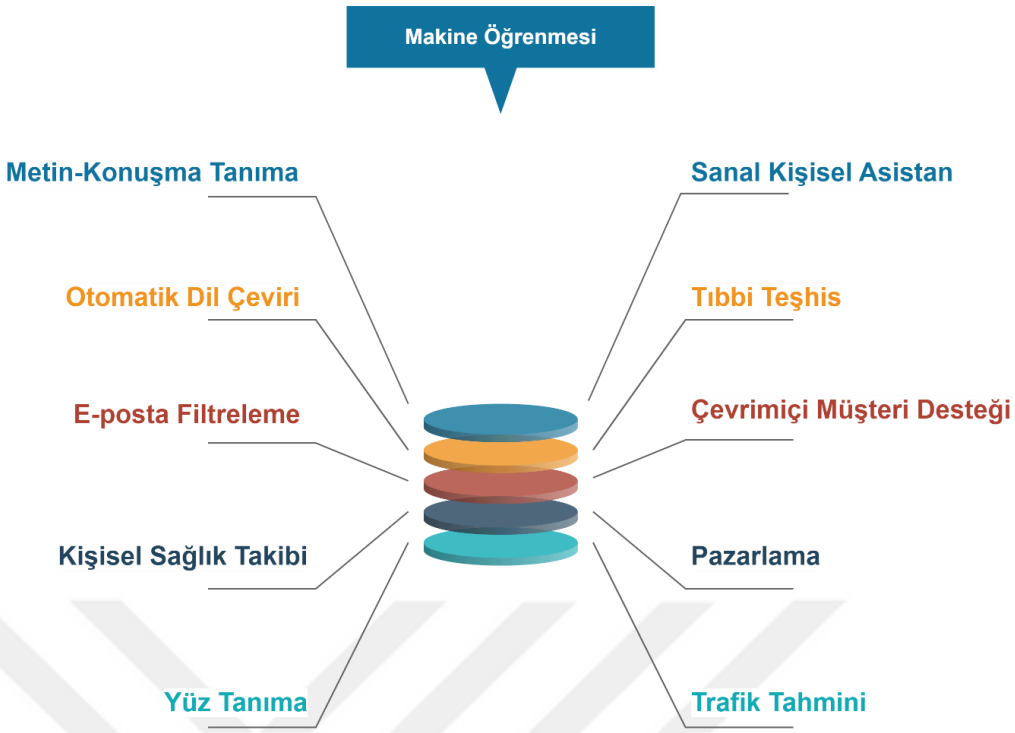
- Veri setini eğitmek için model oluşturulması,
- Eğitilen model sonucunda elde edilen tahminlerin gerçek değerler ile karşılaştırılması ve model başarımının değerlendirilmesi örnek verilebilir.

Doğal dil işleme süreçleri Bölüm 4.2.'de detaylı olarak açıklanmıştır.

3.2. Makine Öğrenmesi

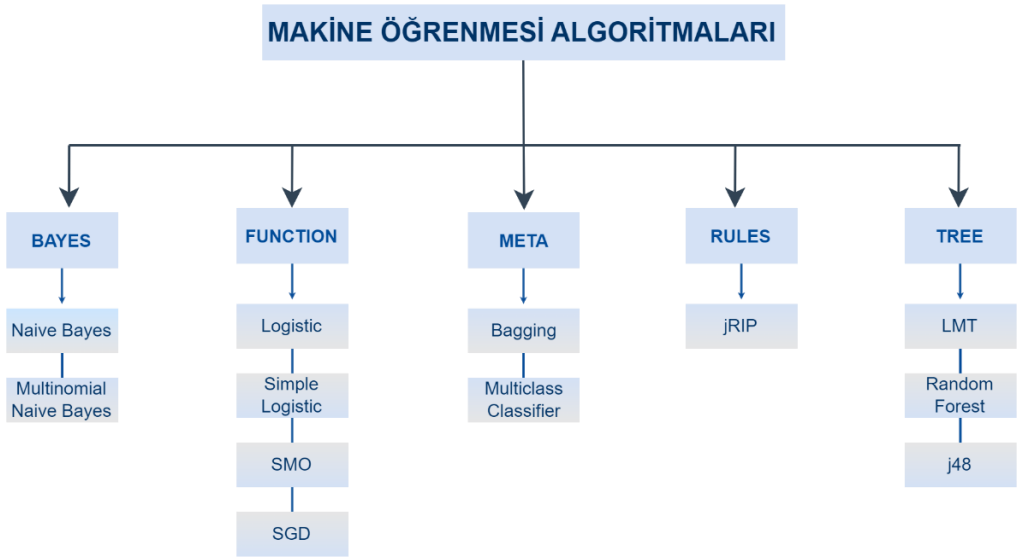
İnsanoğlu varoluşundan itibaren yaptığı işleri basitleştirmenin yollarını aramıştır. Günlük hayat işlerini ya da görevlerini basit bir şekilde gerçekleştirmek için gözlem yaparak, önceki deneyimlerini tekrar değerlendirerek çeşitli araçlar üretmiş ve üretmeye de devam etmektedir. Tüm bu araçlar aslında insanların gereksinimlerine çözüm bulmak için yaratıcıklarını kullanarak geliştirdikleri araçlardır. Diğer bir ifade ile insan yaratıcılığı ve beyninin ürünleridir denebilir. Bu ürünler, günümüzde gelişen teknoloji ile çok daha ileri seviyelere ulaşmıştır. Birçok alanda insanların yerini gittikçe makineler almakta ve makineler belirli işleri insanlar gibi yapabilmektedir. Bu ifade aslında makine öğrenmesinin de tanımıdır. Makine öğrenimi, bilgisayarların verilerden öğrenmeyi nasıl yaptığını anlamaya ve geliştirmeye çalışan bilimsel bir disiplindir. Makine öğrenmesi, verilerden ilişkileri anlayarak istatistik öğelerini bilgisayar biliminden öğelerle birleştirerek verileri yönetmek için algoritmalar geliştiren yapay zekanın bir alt dalıdır [19].

Makine öğrenmesi ile geliştirilen uygulamalar insanların işlerini kolaylaştırmanın yanı sıra, iş kazası gibi risklerin insanlara vereceği zararı minimuma indirme için de kullanılabilir. Makine öğrenmesinin kullanıldığı alanlardan bazıları Şekil 3.2'de gösterilmektedir.



Şekil 3.2. Makine Öğrenmesi Kullanım Alanları

Çoğu doğal dil işleme algoritması, makine öğrenimine dayanır. Makine öğreniminin işleyişi, dil işleme girişimlerinden farklıdır. Dil işleme görevlerinin önceki uygulamaları, tipik olarak, büyük kural kümelerinin doğrudan elle kodlanmasını içeriyordu. Makine öğrenimi ise tipik gerçek dünya örnek uzayının analizi yoluyla bu tür kuralları otomatik olarak öğrenmek için genel öğrenme algoritmasının kullanılmasını gerektirir. Doğal dil işleme görevlerine birçok Şekil 3.3'te görüldüğü gibi farklı makine öğrenimi algoritmaları uygulanabilir.



Şekil 3.3. Makine Öğrenimi Algoritmaları

3.2.1. Naive Bayes Algoritması

Naive Bayes, olasılık hesabına dayalı olan ve makine öğreniminde en sık kullanılan algoritmalardan biridir. İsmini İngiliz matematikçi Thomas Bayes'den almıştır. Bayes teoreminden yola çıkılarak, bir sınıftaki bir özelliğin olma olasılığının, diğer özelliklerin olma ya da olmama olasılığından bağımsız olduğunu varsayan bir sınıflandırma algoritmasıdır. Her bir özellik için var olma olasılığı hesaplanır ve olasılık değeri en yüksek olan özellik sınıfına göre sınıflandırma yapılır [20]. Yani bir özellik için her özelliğin olma olasılığı hesaplanır. Makine öğrenimi içindeki basit sınıflandırıcılardan biridir, çok sınıflı ya da dengesiz dağımlı veri setlerinde de yüksek performans gösterir. Bayes teoremi Denklem (1)'de verilmiştir.

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)} \qquad (1)$$

$P(A|B)$: B durumu olduğunda A durumunun olma olasılığı

$P(A)$: A durumunun olma olasılığı

$P(A|B)$: A durumu olduğunda B durumunun olma olasılığı

$P(\mathrm{A}|\mathrm{B})$: B durumunun olma olasılığı

3.2.2. Multinomial Naive Bayes Algoritması

Multinomial Navie Bayes, yüksek tahmin başarısı ve kolay modellenmesi ile metin sınıflandırmada yaygın olarak kullanılan sınıflandırıcılardan biridir. Çok terimli Naïve Bayes, kelimenin bir belgede kaç kez geçtiğini ifade eden terim sıklığı kavramı üzerinde çalışır. Bu model, kelimenin bir belgede geçip geçmediği ve o belgedeki sıklığı olan iki kavramı ele alır. Yeni bir haber makalesinin kategorisini tahmin ederken, makaledeki tüm kelimelerin olma olasılıkları ile değerlendirme yapar. Fakat bir kelimenin bir belgedeki terim sıklığının yüksek olması her zaman bu kelimenin doğru sınıflandırmaya götüreceği anlamına gelmeyebilir, kelime potansiyel olarak belgeye bir anlam katmayabilir. Dolayısıyla multinomial sınıflandırıcı uygulanırken öncelikle bu tür kelimelerin metinden ya da belgeden kaldırılması gereklidir. [21]. Denklem (2)'de gösterildiği şekilde bir belgenin w kategorisine ait olma olasılığı ($\mathrm{P}(w)$), tüm belgelerde w belirtecinin sıklığının ($\mathrm{n}(w)$), tüm belgelerdeki toplam terim sayısına ($\mathrm{n}(W)$) oranı ile hesaplanır.

$$\mathrm{P(w)} = \frac{\mathrm{n(w)}}{\mathrm{n(W)}} \qquad (2)$$

3.2.3. Lojistik Regresyon

Lojistik regresyon, hem doğrusal hem de doğrusal olmayan verileri sınıflandırmak için kullanılan gelişmiş bir regresyon tekniğidir. Verileri ikili yanıtlarla modellemek için yaygın olarak kullanılır. Yanıt ikili olduğunda, 0/1 şeklini alır, 1 genellikle bir başarıyı ve 0 bir başarısızlığı gösterir. Ancak 1 ve 0'ın alabileceği gerçek değerler, çalışmanın amacına bağlı olarak büyük ölçüde değişir. İkili sınıfladırmada sınıflardan biri 1 diğeri 0 olarak etiketlenebilir. Lojistik regresyon, verilen girdi değerini alıp girdiyi ağırlık değeri ile çarparak uygulanan bir makine öğrenme tekniğidir [22].

3.2.4. Simple Lojistik

Simple lojistik sınıflandırıcı, veriler doğrusal olduğunda çok iyi performans gösterir, ancak doğrusal olmayan veya karmaşık verilerle performans zayıf olabilir. Bir veri kümesindeki eksik verileri işleyemez [23].

3.2.5. Sıralı Minimum Optimizasyon

Sıralı Minimum Optimizasyon (Sequential Minimal Optimization – SMO), karesel programlama çözücüleri aracılığıyla denetimli makine öğrenme modeli olan destek vektör makinelerini eğitmek için yaygın olarak kullanılan bir sınıflandırma algoritmasıdır. Çeşitli özel yaklaşımlar arasında SMO, SVM'de bulunan ikinci dereceden programlama problemini çözmenin bir yoludur. SMO tarafından kullanılan yinelemeli algoritma, optimizasyon problemini bir dizi daha küçük alt probleme böler. Bu QP tabanlı alt problemler daha sonra sayısal QP optimizasyonundan tamamen kaçınılarak analitik olarak çözülür [24].

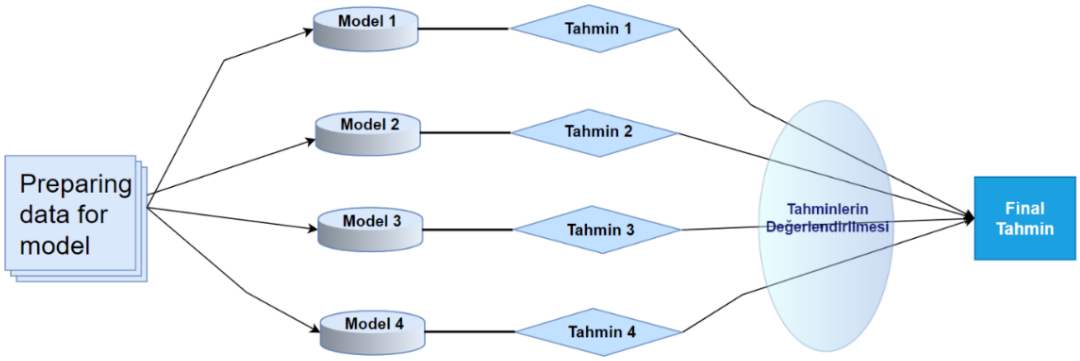
3.2.6. Stokastik Gradyan Azalma

Stokastik Gradyan Azalma (Stochastic Gradient Descent – SGD), makine öğreniminde çok çeşitli doğrusal modelleri (lojistik regresyon, doğrusal regresyon, ikili sınıf destek vektör makineleri, vb.) öğrenmek için kullanılan bir algoritmadır. Genellikle diferansiyel fonksiyonların toplamı şeklinde ifade edilen amaç fonksiyonu minimizasyonu problemini analiz etmek için yinelemeli yöntem ve istatistiksel tahmin kullanır. Algoritma, eğitim setini yakınsayana kadar araştırır ve nominal nitelikleri hedef ikili koşullara dönüştür. Sıradan gradyan inişine kıyasla SGD, tüm eğitim seti yerine sadece birkaç örnekten gerçek gradyanı ekonomik olarak değerlendirerek hesaplama maliyetini azaltır [24].

3.2.7. Bagging

Bagging birlikte (ensemble) öğrenme yöntemi ile çalışan bir algoritmadır. Şekil 3.4’te gözlemlendiği gibi birliktelik içindeki her model veri setinden bir miktar veri ile

eğitilir. Modeller birbirlerinden bağımsız oldukları için aynı anda eğitim gerçekleşebilir. Birliktelik modelinin yeni bir test örneğinin sınıflandırılması, sınıflandırıcıların kararları birleştirilerek belirlenir [25]. Amaç birden fazla sınıflandırıcıyla daha yüksek performans elde etmektir.



Şekil 3.4. Ensemble Öğrenme

3.2.8. Multiclass Sınıflandırıcı

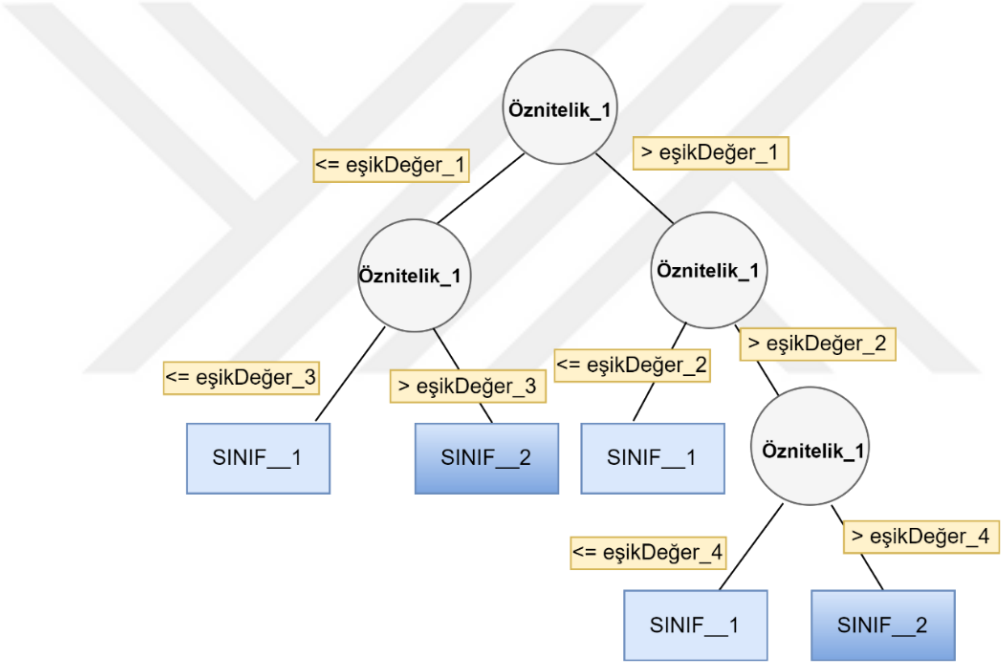
Bir sınıflandırma işleminde tahmin değerinin ikiden fazla sonucu varsa buna çok sınıflı sınıflandırıcı denir. Gözetimli sınıflandırma yöntemidir. Birliktelik kuramına göre işlemler yapılır. Birden fazla algoritmada veri seti alt kümeleri eğitilerek en yüksek başarıma sahip algoritmalar değerlendirilir. Birden fazla algoritma ile test yapıldığı için ve her birinin başarısı değerlendirildiği için makine öğrenmesi alanında genel olarak başarılı bir sınıflandırma yöntemidir.

3.2.9. JRip

JRip yaygın olarak kullanılan makine öğrenmesi algoritmalardan biridir. Çalışma prensibi, sınıflar büyüdükçe analiz edilmesi ve gittikçe azalan hata oranları kullanılarak sınıf için ilk kurallar dizisinin oluşturulmasıdır. Bu algoritma, eğitim verisindeki belirli bir veri setinin tüm örneklerini sınıflandırmak ve o sınıfın tüm üyeleri için geçerli olan bir dizi kural aramak için kullanılır. Daha sonra bir sonraki sınıfa geçer ve tüm sınıflar incelenene kadar süreci tekrarlar [17].

3.2.10. Lojistik Regresyon Ağacı

Lojistik Regresyon Ağacı (Logistic Regression Tree – LMT) temelde lojistik regresyon ile karar ağacı öğrenme metotlarının kombinasyonu şeklindedir. Standart bir karar ağacı yapısındadır. Yaprakları farklı lojistik fonksiyonlar ile ilişkilendirilir. Öznitelik seçimi ile önemli öznitelikleri seçebilme özelliğine sahiptir. Karar ağaçlarında olduğu gibi düğümlerin alt düğümleri vardır. Bu düğümlerdeki lojistik hesaplamalar neticesinde öznitelik değeri ve eşik değer karşılaştırılması ile dallanmalar yapılır. Sayısal öznitelikler için Şekil 3.5'te görüldüğü gibi öznitelik değeri eşik değerinden küçük ise sol düğüme, büyük ise sağ düğüme doğru dallanma gerçekleşir [26].



Şekil 3.5. Lojistik Regresyon Ağacı

3.2.11. Rastgele Orman

Rastgele Orman (Random Forest – RF), bir veri setindeki farklı alt kümelerde birden fazla karar ağacı kullanan ve tahmin doğruluğunu iyileştirmek için sonuçların ortalamasını alan bir makine öğrenimi algoritmasıdır. Yani tek bir karar ağacına bağlı olmak yerine, sonuçları elde etmek için birden çok ağaçtan tahminler toplar. Her ağaç düğümünde, bir koşul, giriş verilerinin bir veya daha fazla özelliği ile karşılaştırılır. Her ağaç için bir sınıf tahmini vardır ve en fazla tahmin edilen sınıf algoritma

sonucunda asıl tahmin edilen sınıftır. RF dengesiz veri setlerinde çok iyi performans gösterir ve sınıflandırma hatalarının sayısı düşüktür [23].

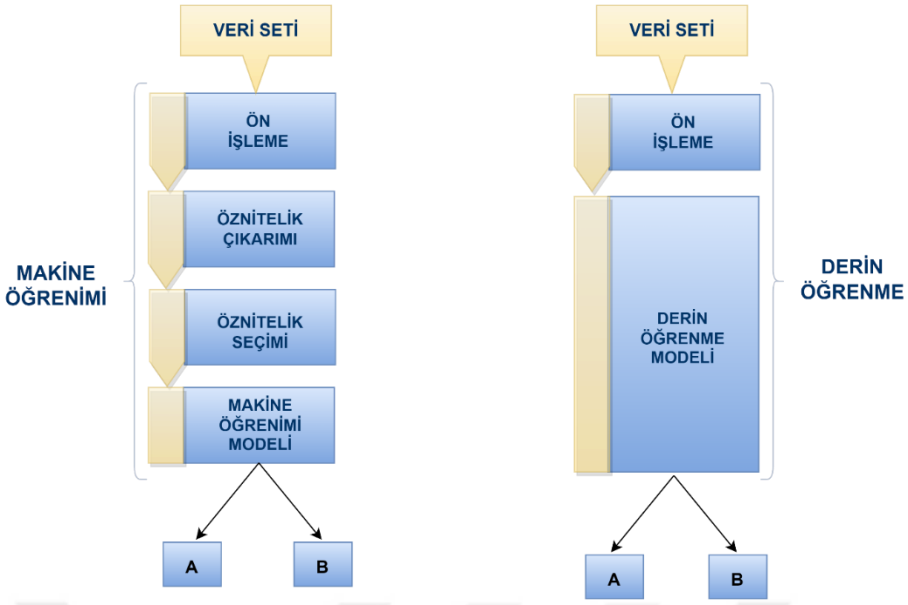
3.2.12. J48

J48, eksik değerler, kural türetme, sürekli öznitelik değer aralıkları gibi özelliklere sahip olan bir makine öğrenimi sınıflandırıcısıdır. Bu algoritma, veriye benzersiz bir kimlik kazandırma kurallarının oluşturulmasında kullanılır. Bu sınıflandırıcıyı kullanmanın amacı, çok yönlülük ve doğruluk arasında bir denge sağlayana kadar karar ağacını kademeli olarak dallandırmaktır [17].

3.3.Derin Öğrenme

Derin öğrenme, yapay zeka alanında önemli ve güncel bir konudur. Derin öğrenme yöntemleri, insan beyni, sinir sistemi ve işleyişleri örnek alınarak geliştirilir. Derin öğrenme, sinir hücrelerinden ilham alınarak geliştirilen yapay sinir ağları ve birden çok gizli katmana sahip mimarileri kullanarak verilerden öğrenme sürecidir. Bir derin öğrenme algoritmasında, veriler birden çok katmandan geçirilir ve her katman aşamalı olarak özellikleri ayıklar ve bilgileri bir sonraki katmana iletir. İlk katmanlar, daha sonra kapsamlı bir sunum oluşturmak için daha sonraki katmanlarla birleştirilen düşük seviyeli özellikleri çıkarır.

Geleneksel makine öğrenimi sınıflandırma görevi sırasıyla ön işleme, özellik çıkarma, özellik seçimi ve sınıflandırma/model oluşturma adımlarından oluşur. Makine öğrenimi yöntemlerinin tahmin başarımlarının yüksek olması büyük ölçüde doğru özellik seçimine dayanır. Buna karşılık, derin öğrenme modelleri öznitelik çıkarımı, öznitelik seçimi, öğrenme ve sınıflandırmayı Şekil 3.6'da gösterildiği gibi eş zamanlı olarak yapmaktadır. Bu özellik, derin öğrenmeyi özellikle çeşitli görevler arasında avantajlı hale getirir.



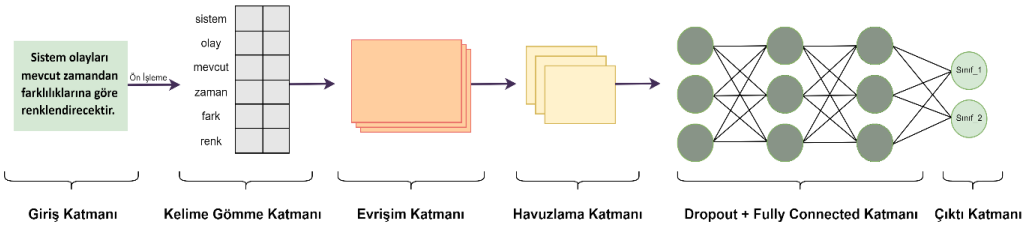
Şekil 3.6. Makine Öğrenimi ve Öğrenme Mimarisi Karşılaştırılması

Yüksek performanslı bilgi işlem olanaklarının gelişmesiyle birlikte, derin sinir ağlarını kullanan derin öğrenme teknikleri artan bir popülerlik kazanmıştır. Her geçen gün farklı yöntemler geliştirilmekte ve bu yöntemleri farklı problemler üzerinde uygulayan çalışmaların sayısı artmaktadır. Bu yöntemler, doğal dil işlemenin çeşitli alt alanlarında da yoğun bir şekilde kullanılmış ve kullanılmaya devam etmektedir. Derin öğrenmenin doğal dil işlemeye uygulanması, sinir ağlarının farklı soyutlama seviyelerine sahip temsilleri öğrenme yeteneği sayesinde dil oluşturma görevi de dahil olmak üzere farklı görevlerde son teknoloji performansının elde edilmesine yardımcı olmuştur [27]. "Evrişimli Sinir Ağları", "Uzun Kısa Süreli Bellek", "Çift Yönlü Uzun Kısa Süreli Bellek", "Geçitli Tekrarlayan Birimler" ve "Çift Yönlü Geçitli Tekrarlayan Birimler" derin öğrenme algoritmalarındandır.

3.3.1. Evrişimli Sinir Ağları

Evrişimli Sinir Ağları (Convolutional Neural Networks – CNN), görüntü tanıma topluluğu içinde yaygın olarak kullanılan, ancak son zamanlarda doğal dil işlemede de dikkat çeken klasik ileri beslemeli sinir ağlarının bir çeşididir. Doğal dil işlemede CNN'ler, kelime sırasını bozulmadan tutan bir cümle gösterimi üzerinde çalışır. Bu nedenle, CNN'ler bir cümlede birden fazla kelimeyi kapsayan kelime dizilerinden oluşan kalıpları öğrenebilir ve tanıyabilir [2]. Bu da CNN'i örüntü tanımayı içeren

görevlerle uyumlu hale getirir [28]. Basit bir CNN ağı Şekil 3.7'deki gibi girdi/kelime gömme (embedding), evrişim (convolution), havuzlama (pooling), seyreltme (dropout), tam bağlantı (fully-connected) ve çıkış (output) katmanlarından oluşur.



Şekil 3.7. CNN Mimarisi

* ***Giriş/Kelime Gömme Katmanı:*** Girdi cümlesinin vektör olarak temsilinin oluşturulup 2 boyutlu matris haline dönüştürüldüğü giriş katmanıdır [28].

* ***Evrişim Katmanı:*** Evrişim katmanında, özellik haritasının çıkarılması için girdi matrisi üzerinde iki boyutlu filtre uygulanır. Filtre, girdi matrisinde evrişim adı verilen her alan üzerine uygulanır. Her evrişim, ağırlıkları ve bölgesel girdi arasındaki skaler çarpımı hesaplayan bir nöron görevi görür ve ardından aktivasyon fonksiyonu tek bir özellikle sonuçlanır. Her evrişimden gelen özellikler, her filtre için bir özellik haritasında toplanır [28].

* ***Havuzlama Katmanı***: Havuzlama katmanı evrişim katmanını takip eder ve girdi verileri üzerinde aşağı örnekleme ve boyutsallık azaltma gerçekleştirerek ağdaki bağlantı sayısını azaltır. Birincil amacı, hesaplama yükünü hafifletmek ve aşırı öğrenme sorunlarını ele almaktır. Ayrıca, havuzlama katmanı, bir görüntünün çeşitli boyutlarını birleştirerek, CNN'lerin şekilleri bozulsa veya farklı açılardan bakılsa bile nesneleri tanımasını sağlar [27].

* ***Seyreltme Katmanı:*** Yapay sinir ağlarında, seyreltme, modelin genelleştirilmesini sağlamak için yaygın olarak kullanılan tekniklerinden biridir. Ayrıca eğitim için gereken yineleme sayısını azaltabilir, eğitimi hızlandırabilir ve eğitilen ağları daha düşük hata oranları ile elde etmeyi sağlar [1].

* ***Tam-Bağlantı Katmanı:*** Dense katmanı olarak da bilinir, özellikle son tahmin için kullanılır. Bu katmanda, veriler birbirine tamamen bağlı olan bir dizi nörondan geçirilir. Sonuç olarak verinin hangi sınıfa ait olduğuna dair tahminde bulunur [2].

3.3.2. Uzun Kısa Süreli Bellek

Uzun Kısa Süreli Bellek (Long Short-Term Memory – LSTM), tekrarlayan sinir ağlarının gelişmiş bir versiyonudur. LSTM modellerinin bilgileri daha uzun dizilerde tutma ve kullanmada daha etkilidir [27]. Sinir ağı katmanlarından oluşan bir LSTM mimarisinde, işlemin yapıldığı katmandaki giriş veri bilgisi ve önceki katmandan gelen çıkış veri bilgisi bellekte tutulur. Mevcut bilgi ve önceki bilgi kullanılarak sonraki durumlar tahmin edilir. Belleğe gelecek verinin ne zaman gireceği, mevcut verinin ne zaman çıkacağı da LSTM içinde denetlenir. Temel olarak bir LSTM mimarisinde giriş, unutma ve çıkış olmak üzere üç blok bulunmaktadır ve bilgi akışı bu bloklar aracılığıyla yapılır. Uzun süreli bağımlılıkları kolayca öğrenebilir.

3.3.3. Çift Yönlü Uzun Kısa Süreli Bellek

Çift Yönlü Uzun Kısa Süreli Bellek (Bidirectional Long Short-Term Memory – Bi-LSTM), sıralı modelleme görevlerinde hem geçmiş hem de gelecek bağlamı dikkate alarak standart LSTM modellerinin sınırlamalarını ele alan LSTM mimarisinin bir uzantısıdır. Geleneksel LSTM modelleri giriş verilerini yalnızca ileri yönde işlerken, Bi-LSTM modeli iki yönde eğiterek bu sınırlamanın üstesinden gelir. Standrat bir Bi-LSTM mimarisinde giriş dizesini ileri ve geri yönde işleyen paralel iki katman bulunur. Bu çift yönlü işleme, modelin hem geçmiş hem de gelecekteki bağlamdan bilgi yakalamasını sağlayarak, dizi içindeki zamansal bağımlılıkların daha kapsamlı bir şekilde anlaşılmasını sağlar [27].

3.3.4. Geçitli Tekrarlayan Birimler

Geçitli Tekrarlayan Birimler (Gated Recurrent Unit – GRU) LSTM'nin işlevselliğinden esinlenilerek geliştirilen bir RNN türüdür. GRU, sıfırlama ve güncelleme kapıları olmak üzere iki kapıdan oluşur ve kontrol edecek bir

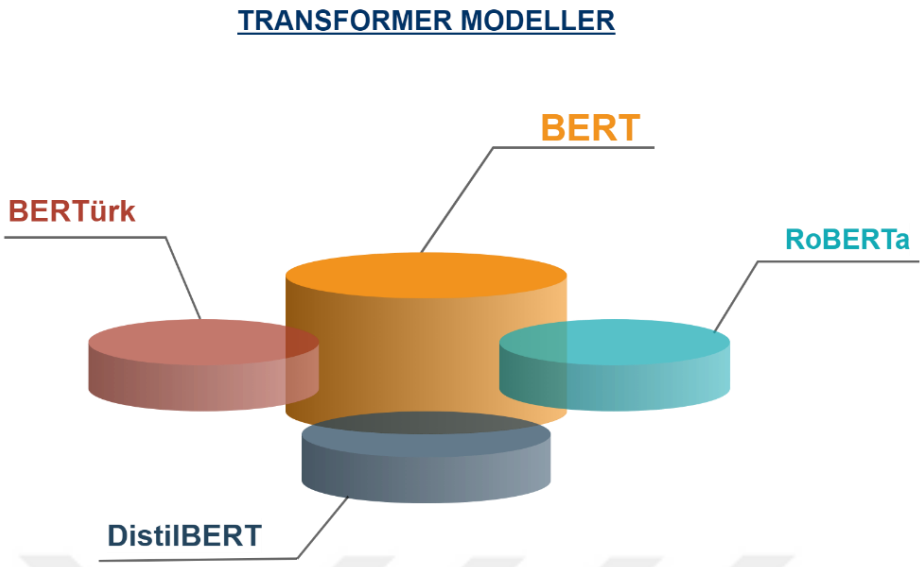
mekanizmaya sahip olmadan her seferinde tüm durumu gösterir. Sıfırlama, ihtiyaç duyulmayan bilgilerin unutmasını sağlar. Güncelleme kapısı ise önceki gizli durumdan ne kadar bilginin taşındığını kontrol eder. Gerçek aktivasyon, önceki aktivasyon ve aday aktivasyonun lineer interpolasyonu olarak hesaplanır [14].

3.3.5. Çift Yönlü Geçitli Tekrarlayan Birimler

Çift Yönlü Geçitli Tekrarlayan Birimler (Bidirectional Gated Recurrent Unit – Bi-GRU), iki GRU'dan oluşur. Bu GRU'lar zıt yönlere sahiptir ve parametreleri bağımsızdır. Bi-GRU'nun avantajı, bağlamsal cümleler arasındaki ilişkiyi analiz edebilmesidir. Bu şekilde her bir cümlenin anlamı hakkında doğru kararlar verilebilir ve orijinal anlama en yakın metin özellikleri çıkarılabilir [29].

3.4. Transformer Modeller

En gelişmiş dil modellerinin temelini oluşturan transformer tabanlı dil modelleri, dil işlemeyi insanlar gibi yapabilmek amacıyla doğal dil metinlerini analiz etmek için doğal dil işlemede kullanılan özel bir yapay zeka türüdür. Transformer tabanlı modeller, işlenmiş kelimelerin bağlamını dikkate alarak ilgili kelimelerin daha detaylı bir şekilde anlaşılmasını sağlar [30]. Bu tür modeller önceden eğitilmiş dil modelleridir yani çok sayıda doğal dil işleme görevi için test edilmiş bir çözümdür. Bu bağlamda, transformer tabanlı bir dil modeli önce çok miktarda metin üzerinde eğitilir, ardından göreve özgü veriler üzerinde ince ayar yapılır [31]. Şekil 3.8'de gösterildiği gibi BERT, BERTürk, DistilBERT ve RoBERTa sık kullanılan transformer modellerdendir.



Şekil 3.8. Transformer Modeller

3.4.1. BERT

Çift Yönlü Kodlayıcı (Bidirectional Encoder Representations from Transformers) anlamına gelen BERT, Google tarafından geliştirilen dönüştürücü tabanlı bir dil modelidir. Kodlayıcı-kod çözücü olmak üzere iki aşamadan oluşmaktadır. Kodlayıcı, kodlama katmanlarında girdiyi sırasıyla işler ve bir çıktı verir. Bu çıktı kod çözme katmanlarında tekrar işlenir. BERT, BooksCorpus veri setlerinden ve İngilizce Vikipedi'dan alınan 16 GB metinler üzerinde eğitilmiştir. BERT bir metin içindeki kelimeleri analiz ederken sadece morfolojik olarak değil, aynı zamanda bağlamsal olarak da analiz etmektedir. BERT, "Maskeli Dil Modellemesi" ve "Sonraki Cümle Tahmini" mekanizmaları ile dili analiz edebilir. Maskeli dil modellemesinde girdi cümlesi içinde herhangi bir yerdeki bir kelime yok sayılır, yok sayılan/maskelenen bu kelimenin öncesi ve sonrasındaki kelimeler ile analiz yapılarak maskelenen bu kelime tahmin edilmeye çalışılır. Sonraki cümle tahmini mekanizmasında da benzer bir mantık bulunmaktadır. Cümle içindeki herhangi bir kelime yerine, girdi metni içindeki herhangi bir cümle maskelenir ve kendisinden önceki ve sonraki cümleler analiz edilerek maskelenen cümle tahmini yapılır. Bu özelliği ile birçok dil modelinden daha yüksek performanslar göstermektedir [32].

BERT modelinin daha önceden büyük veri kümeleriyle eğitilmiş olması, kullanılacağı modele hız açısında önemli katkı sağlamaktadır. Çünkü geliştirilecek model, daha önceden var olan bir model üzerine inşa edilmektedir. Sadece ön eğitim ve ince ayar yapılması yeterlidir.

3.4.2. BERTürk

Doğal dil işlemede BERT modeli kullanılmaya başlandığında, çeşitli problemler ya da farklı diller için BERT mimarisinin farklı varyasyonları geliştirilmiştir. Türkçe metinlerin analizi için Türkçe veriler ile eğitilmiş BERTürk modeli de bu varyasyonlardan biridir. BERT ile aynı mimariye sahip olduğu için BERTürk kullanılarak oluşturulan modeller de diğer doğal dil işleme modellerine göre daha performanslı ve hızlı olmaktadır. Hazır bir model kullanıldığı için ön işleme ve ince ayar işlemlerinin doğru ve modele uygun bir şekilde yapılması performansı önemli ölçüde etkilemektedir.

3.4.3. DistilBERT

DistilBERT de BERTürk gibi BERT mimarisinin farklı bir versiyonudur. BERTürk, BERT'ten farklı bir dil için geliştirilirken DistilBERT'te model üzerinde değişikliğe gidilmiştir. BERT'in ilk sürüm mimarisi baz alınmıştır ve daha geniş parametrelere sahip daha büyük mimarileri, aynı mimarinin daha az parametreye sahip hafif bir versiyonuyla değiştirilmesiyle oluşturulmuştur. BERT tabanlı modeldeki katman sayısını iki kat azaltır ve genel amaçlı kullanım için çok daha küçük ve daha hızlı bir BERT sürümü elde etmek için belirteç yerleştirmeleri ve havuzlayıcıları kaldırır. Dinamik maskeleme uygular ve sonraki cümle tahminlerini göz ardı eder. DistilBERT'in amacı BERT'e göre daha hızlı çalışan bir sürümünü elde etmektir [32].

3.4.4. RoBERTa

Roberta, BERT'nin ilk sürümünde çeşitli parametreleri değiştirerek BERT'i optimize etmeyi amaçlayan BERT'in bir versiyonudur. Daha büyük eğitim öncesi verisi, ince ayarlar ve dinamik maskeleme ile sonraki cümle tahmin kodlaması ile

eğitilmiştir. BERT için kullanılan statik maskeleme yerine dinamik maskeleme kullanılmıştır ve bu da çoklu görevlerde iyileşme sağlamıştır [32].

3.5.Makine Öğrenmesinde Kullanılan Araçlar

3.5.1. Weka

WEKA, veri işleme görevlerinde kullanılan, açık kaynak kodlu bir makine öğrenimi aracıdır. WEKA, Yeni Zelanda Waikato Üniversitesi'nde Genel Kamu Lisansı'nda geliştirilmiştir. Java programlama dili kullanılarak yazılmıştır ve Grafik Kullanıcı Arayüzü (GUI) veya Java Uygulama Programlama Arayüzü (API) aracılığıyla erişilebilir. Weka ile veri ön işleme, görselleştirme, sınıflandırma, regresyon, kümeleme, özellik seçimi gibi çeşitli işlemler yapılabilir [23]. Ayrıca bu işlemlere ek olarak cross-validation, eğitim-test veri bölümleme gibi özellikler de Weka aracılığıyla yapılabilir.

3.5.2. Jupyter Notebook

Anaconda, Python veya R'de kod yazmak için çeşitli ortamlar sunan özel olarak tasarlanmış ücretsiz bir yazılımdır. Entegre geliştirme ortamları (Integrated Development Environments-IDE) olarak da bilinen bu ortamlar, kod geliştirmeyi büyük ölçüde kolaylaştıran platformlar veya uygulamalardır. Ayrıca Anaconda ile entegre geliştirme ortamlarını daha ergonomik hale getirmek için farklı araç setleri (toolkit) indirilebilir. Bu araç setleri ile önceden oluşturulmuş işlevler kullanılabilir. Bu işlevler, kitaplık adı verilen bölümlerde yeniden gruplandırılmıştır ve Anaconda aracılığıyla kolayca indirilebilir. Python'u bilgisayarınıza kurmanın ve kullanmanın birçok farklı yolu vardır, ancak Anaconda, kurulumunda en önemli kitaplıkları ve platformları içeren basit, grafiksel bir kullanıcı arabirimidir (Graphical User Interface – GUI). Anaconda ayrıca tüm bu kitaplıkları güncel tutma sürecini de basitleştirir. Böylece Python'u farklı platformlar, kitaplıklar ve işlevlerle ayrı ayrı kurmak yerine, Anaconda bunu tek bir kurulumda yapabilir. Jupyter Notebook bu platformlardan biridir.

Jupyter Notebook, varsayılan web tarayıcınızı kullanan web tabanlı bir IDE'dir. Jupyter Notebook'un esnek bir kullanımı vardır; bir dosyası içinde yazılan her kod bloğu ayrı ayrı çalıştırılabilir. Aynı Jupyter Notebook dosyası içinde farklı metin türleri kullanılabilir. Böylece kod çıktıları, görseller, denklemler ve normal metinler tek bir yerde bulunru. Web tabanlı olduğu için kod dosyalarını başkalarıyla paylaşmayı da kolaylaştırır [33].

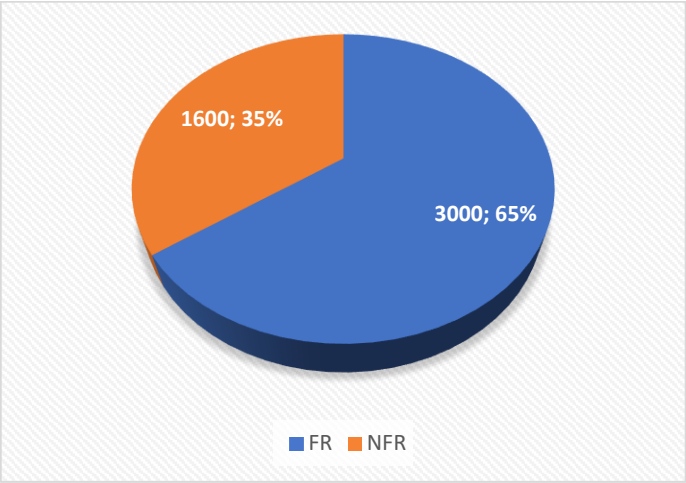
4. MATERYAL VE YÖNTEM

Bu çalışmada oluşturulan veri seti, veri setini işlemek için kullanılan teknikler ve geliştirilen algoritmalar bu bölüm altında incelenecektir.

4.1. Materyal

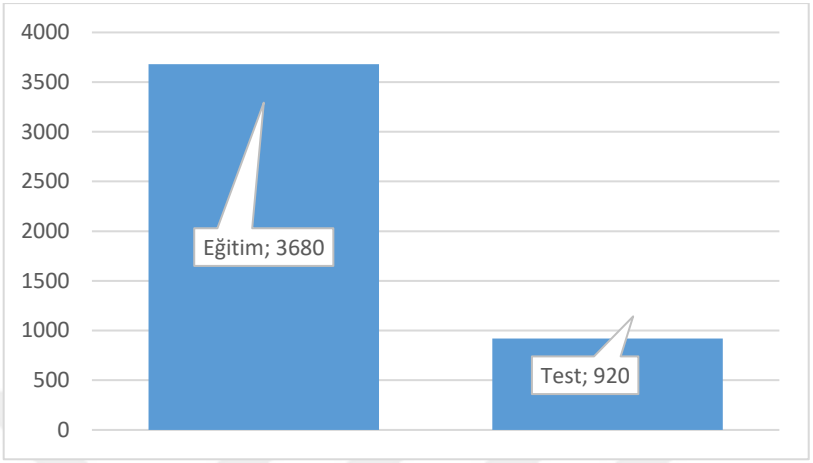
4.1.1. Veri Seti ve Özellikleri

Yazılım geliştirme süreçlerinde en önemli aşama olan gereksinimlerin tespit edilmesi ve analizi doğrudan yazılım kalitesini etkileyen unsurlardan biridir. Bu amaçla yazılım gereksinimlerinin gerçek hayata uyarlanabilir ve kullanılabilir olması gerekmektedir. Bu gerekçelerden yola çıkarak öğrenciler tarafından windows, web ve mobil tabanlı platformlar için geliştirilmiş çeşitli özgün projelere ait gereksinimler incelenerek tez çalışması kapsamında kullanılan veri seti oluşturulmuştur. Türkçe dilinde konu ile ilgili yapılan çalışmalar sınırlı olduğu için veri setinin Türkçe olması tercih edilmiştir. Veri seti Şekil 4.1'da gösterildiği üzere 3000 işlevsel gereksinim ve 1600 işlevsel olmayan gereksinim olmak üzere etiketlenmiş toplam 4600 gereksinimden oluşmaktadır. Tablo 4.1'de kullanılan veri setine ait örnekler verilmiştir.



Şekil 4.1. Veri Seti Gereksinim Tür Dağılımı

İşlevsel ve işlevsel olmayan olmak üzere iki sınıflı olarak sınıflandırılmış veri seti, çalışma içerisinde %80 eğitim ve %20 test verisi olacak şekilde ayrılmıştır ve Şekil 4.2'de sayısal olarak ifade edilmiştir.



Şekil 4.2. Veri Setinin Eğitim-Test Veri Dağılımı

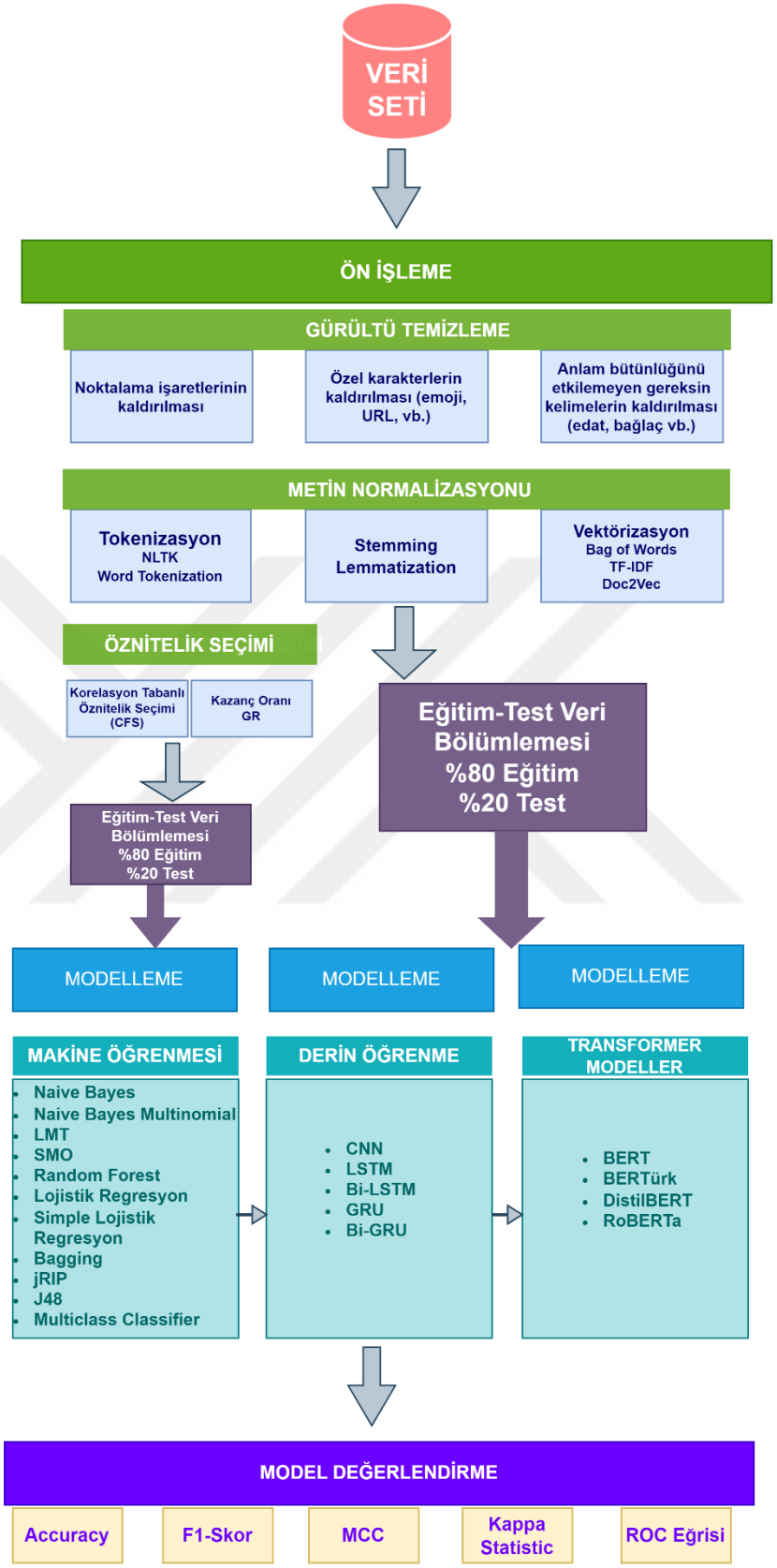
Tablo 4.1. Sınıflandırılmış Veri Örnekleri

Yazılım Gereksinimleri	Etiket
Sistem olayları mevcut zamandan farklılıklarına göre renklendirecektir.	FR
Kullanıcı Canlı Döviz Takip Uygulaması üzerinden bir bankanın döviz bilgilerini anlık takip edebilecektir.	FR
Sosyal doku analizi uygulaması üzerinde eklenen kullanıcı bilgileri saklanacaktır.	FR
RemMed uygulaması üzerinden kullanıcı sağlık günlüğünü doktoru ile paylaşabilecektir.	FR
Kullanıcı e-mail ve şifresi ile sisteme giriş yapabilecektir.	FR
Seyahatname uygulaması tarafından kullanıcının yaptığı hatalara karşın doğru hata mesajları verilmelidir.	NFR
Mobil tabanlı pazaryeri uygulaması aynı anda en az 1000 kullanıcıya hizmet verebilecektir.	NFR
Network alt yapısı sistem kaynaklarının her biri için ortalama en fazla %50'sini kullanmalıdır.	NFR
Geliştirilecek oyun programı üzerindeki ekran kontrolleri oyuncunun oyunu oynamasına engel olmayacak büyüklükte olmalıdır.	NFR
Hesabını Bil uygulaması üzerinde yer alan ekranların yenilenme süresi en fazla 5 saniye olacaktır.	NFR

4.2. Yöntem

Çalışmada kullanılan yöntem üç adımdan oluşmaktadır. Bu adımlar, oluşturulan veri seti kullanılarak yazılım gereksinim sınıflandırmasının;

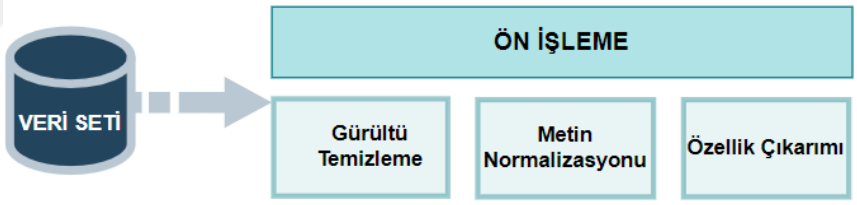
- Makine öğrenmesi algoritmaları kullanılarak değerlendirilmesi,
- Derin öğrenme algoritmaları kullanılarak değerlendirilmesi ve
- Transformer model tabanlı algoritmalar kullanılarak değerlendirilmesi olarak takip edilmektedir. Şekil 4.3'te sistem mimarisi görülmektedir.



Şekil 4.3. Sistem Mimarisi

4.2.1. Veri Ön İşleme

Veri setini oluşturduktan sonra doğal dil işlemenin ilk adımı olan veri ön işleme adımı gerçekleştirilir. Metin, resim, video vs. türündeki verilerin bir yapay zekâ modelinde girdi olarak verildiğinde bu girdilerin bilgisayarın/modelin anlayacağı bir formata dönüştürülmesi gerekir. Doğal dil işlemede veri ön işleme, doğal metin formatında olan yapılandırılmamış verinin düzenlenip belirli bir formatta yapılandırılması ve kalıplarının belirlenmesidir. Bir başka ifade ile veri setine verinin algoritmada kullanılacak model için hazırlanmasıdır. Bu adım, makine öğrenimi modeli oluşturmada temeldir çünkü modelin doğruluğu ve gerçekliği yansıtması büyük ölçüde verilerin güvenilirliğine bağlıdır. Ayrıca geliştirilecek sınıflandırma modeli performansının yüksek olması için veri ön işleme adımında kullanılan teknikler büyük öneme sahiptir. Şekil 4.4'te görüldüğü üzere yapılacak çalışmaya göre ön işleme aşamalarının sırası, uygulanacak teknikler veya prosedürler farklılık gösterebilmektedir [34, 35, 36].



Şekil 4.4. Veri Ön İşleme Adımları

4.2.1.1.Gürültü Temizleme

Veri ön işlemenin önemli adımlarından biri gürültü temizlemedir. Model veri setini, kendisine verildiği şekilde eğitir, üzerinde herhangi bir düzeltme yapmaz. Dolayısıyla metin içinde bozukluklar, gereksiz ifadeler vs. var ise bu tür ifadeler de model üzerinde işlenir ve başarım sonucunu olumsuz bir şekilde etkiler. Veri seti içinde, metnin anlamını etkilemeyen, gereksiz ifade ya da işaretler bulunan veriye gürültülü veri denmektedir. Veriyi kaliteli ve temiz bir şekilde modele vermek için birtakım işlemlerin uygulanması gerekmektedir.

* ***Removal Punctuation:*** Noktalama işaretlerinin ve fazla boşlukların kaldırılmasıdır.

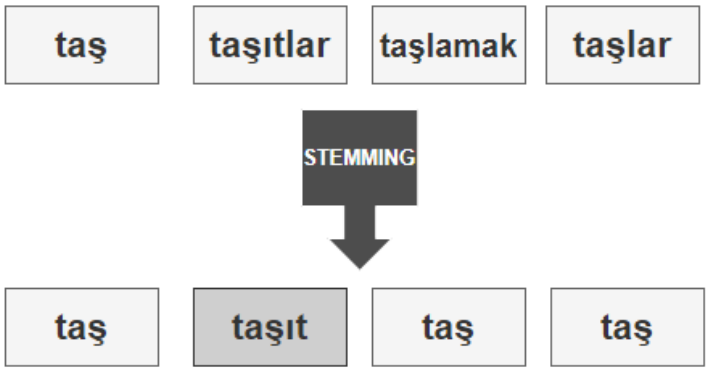
* ***Removal Special Characters:*** Özel karakterlerin kaldırılmasıdır.
* ***Remaval URL Informations:*** URL bilgilerinin kaldırılmasıdır.
* ***Converted into Lowercase Letters:*** Veriyi standart bir forma dönüştürmek için küçük/büyük harf dönüşümüdür.
* ***Removal of Repetitive Words:*** Metin içinde tekrar eden kelimelerin çıkarılmasıdır.
* ***Removal Stopwords:*** Anlamsal bütünlüğü bozmayan 've, veya, gibi, ya da vb.' bağlaçların çıkarılmasıdır. Yazılım gereksinim metinlerinden gereksiz sözcüklerin silinmesiyle, gereksiz sözcüklerin anlamsal analizlerine odaklanmanın önüne geçilmiş olur [40].

Bu işlemler doğal dil işlemenin her problemi için aynı olmak zorunda değildir, ele alınan probleme göre değerlendirilmelidir.

4.2.1.2. Normalizasyon

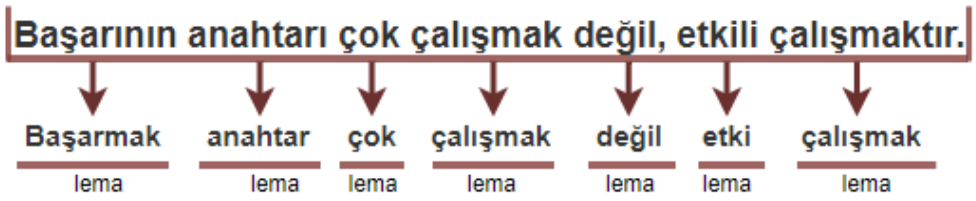
Türk dilinin yapısından dolayı bir kelimenin, tek başına ya da cümle içinde kullanılabilecek anlamsal çeşitliliğinin fazla olması sebebiyle veri ön işleme süreçleri Türkçe doğal dil işlemede ele alınan en zor süreçlerden biridir. Normalizasyon işlemi bu süreçte veriyi daha temiz ve basit bir hale getirmek için kullanılan yöntemlerden oluşur. Metnin sözdizimsel ve anlamsal ayrıştırması için gerekli olan bir ön işleme adımıdır [41, 36]. Normalizasyon aşamasında kelimelerin cümle içindeki kullanım türlerinin ve yerlerinin analizi yapılmaktadır. Aşağıda normalizasyon işlemleri açıklanmıştır.

* ***Stemming:*** Metin içinde farklı formlarda kullanılan kelimelerin farklı anlam olarak algılanmasını önlemek için ortak forma dönüştürülür. Kelimenin gövdesine kadar analiz yapılır. Sözcükler dil bilgisi açısından farklı biçimlerinde yazılsa bile aynı anlamları/bilgileri taşıyabilirler. Bu aşamada Şekil 4.5'te gösterildiği gibi kelimeden çoğulluk, durum, yer, zaman vb. ekler temizlenir ve yalnızca kelime gövdesi elde edilmiş olur.



Şekil 4.5. Stemming

* ***Lemmatization:*** Kelimenin morfolojik köküne indirgenmesidir. Şekil 4.6'da ifade edildiği gibi kelime eklerinden ayrıştırılıp yalın ya da mastar haline dönüştürülür.



Şekil 4.6. Lemmatization

* ***Tokenizasyon:*** En kısa tanımı ile metni yapısal olarak anlamlı küçük parçalara ayırmak olarak tanımlanabilir. Bu parçalar Şekil 4.7'de gösterildiği gibi bir sözcük, terim veya sembol gibi anlamlı öğeler olabilmektedir. Her bir parçaya belirteç (token) adı verilir. Tokenizasyon işlemi ile cümlenin sınırları da belirlenebilmektedir. Tokenizasyon için çeşitli araçlar bulunmaktadır. Yazılım gereksinim sınıflandırma için yapılan bu çalışmada NLTK Word Tokenization ve Word Tokenization with Python NLTK araçları kullanılmıştır. NLTK, doğal dil verileriyle çalışan Python ile yazılım geliştirmede kullanılabilen önemli bir platformdur. Metin işleme süreçleri için ssınıflandırma (classification), tokenize etme (tokenization), kelimeyi köklerine ayırma (Lemmatization), etiketleme (tagging), akıl yürütme (semantic reasoning) gibi işlemler için kullanımı kolay olan bir kütüphanedir. Word Tokenization with Python NLTK ise birden fazla tokenize etme aracı sunan NLTK içinde bir modüldür. Belirteç, metnin bağlamsal

bilgisini içeren atomik bir birim olarak hizmet ettiğinden, bir belirtecin nasıl tanımlanacağı, bir modelin performansında belirleyici bir rol oynar [36, 37].

```
array([TaggedDocument(words=['sisteme', 'giriş', 'sayfasının', 'ortalama', 'yüklenme', 'süresi', '1
5', 'saniyeden', 'az', 'olmalıdır', '.'], tags=['NFR']),
       TaggedDocument(words=['kullanıcı', 'e-hobilerim', 'uygulaması', 'üzerinden', 'diğer', 'kullanı
cıların', 'paylaştığı', 'gönderilere', 'yorum', 'yapabilir', '.'], tags=['FR']),
       TaggedDocument(words=['seyahatname', 'uygulaması', 'üzerinde', 'kullanıcılar', 'için', 'kısa',
'yollar', 've', 'hızlı', 'erişim', 'olanakları', 'olmalıdır', '.'], tags=['NFR'])],
      dtype=object)
```

Şekil 4.7. Tokenizasyon

* ***Kelime Vektörleri Oluşturma:*** Metnin bilgisayar tarafından anlaşılabilir olması için sayısal olarak ifade edilmesi gerekmektedir. Metnin anlamlı sayısal temsilleri bilgisayar tarafından okunabilirdir. Metnin makine tarafından işlenmesini sağlamak için kullanılan yöntem vektörleştirmedir (vektorization) [38]. Bu tanımdan yola çıkarak yazılım gereksinim metinlerini diğer ön işleme adımlarında ele alıp, metin üzerinde temizlemeler yaptıktan ve tokenize işlemi ile kelimelerine ayırdıktan sonra her bir token dokümandaki anlamlarına ve sözdizimsel bilgilerine bağlı kalarak sayısal değerlere dönüştürülmelidir. Metnin bu şekilde sayısal temsillerinin oluşturulması işlemine kelime vektörleri oluşturma (vektörizasyon) denir ve doğal dil işleme süreçlerinde kelimeler arası benzerlikleri ölçmek için kullanılır [38, 39]. Vektörizasyon işlemi kelime gömme (word embedding) olarak da bilinir. Vektörizasyon için kullanılan yöntemlerden bazıları Word2Vec, Fasttext, Glove, Doc2Vec ve TF-IDF'tir. Çalışma kapsamında bu yöntemlerden Word2Vec, Doc2Vec ve TF-IDF kullanılmıştır.

Vektörizasyon Yöntemleri

- ***Word2Vec***

Word2Vec, Continous Bag of Words (CBoW) veya Skip-Gram öğrenme mimarilerini kullanarak kelime ve bu kelimenin belirli bir çerçevesindeki kelimeler ile anlamsal olarak yakınlık ilişkisi kurarak, kelimeleri vektör uzayında bu ilişkiye göre konumlandırır. CBoW mimarisinde çerçeve merkezinde tahmin edilmeye çalışılan kelime bulunmaktadır ve çerçeve sınırındaki komşularına bakılarak tahmin işlemi yapılır. Skip-Gram mimarisi ise CBoW'daki durumun

tersidir. Yine çerçeve merkezindeki kelime ile çerçeve sınırları içinde bulunan komşuları tahmin etmeye çalışılır [39].

- ***TF-IDF***

TF-IDF (Term Frequency–Inverse Document Frequency) yönteminde amaç bir kelimenin bulunduğu metindeki önemini bulmaktır. TF, terim sıklığı, kelimenin bulunduğu belgede ne kadar sık geçtiğini ifade eder. TF değerinin yüksek olması kelimenin belgede sık kullanıldığı dolayısıyla önemli olduğunu göstermektedir. IDF terimi ise kelimenin tüm belgelerdeki sıklığı olarak ifade edilir [34].

Denklem (3)'te gösterilen formül ile hesaplanan TF-IDF değerinin yüksek olması genellikle kelimenin veri setiyle alakasının büyük olduğunu göstermektedir.

$$W_{i,j} = TF_{i,j} \times log\frac{N}{DF_i} \qquad (3)$$

$TF_{i,j}$= i kelimesinin j belgesinde bulunma sıklığını, DF_i, i kelimesini içeren belge sayısı ve N toplam belge sayısını gösterir [39].

- ***Doc2Vec***

Word2Vec kelime temsil yönteminin eksikliklerini gidermek amacıyla Mikolov ve Le (Le ve Mikolov, 2014) tarafından Doc2Vec geliştirilmiştir [29]. Doc2Vec yönteminin temel çalışma prensibi Word2Vec ile aynıdır. Tek fark Doc2vec vektörü, cümleler/paragraflar arasındaki benzerlikleri bulmak gibi çeşitli amaçlar için kullanılabilir [29, 42].

4.2.2. Öznitelik Seçimi ve Özellik Çıkarımı

Öznitelik ile veri seti içinde bulunan her bir nitelik/kolon olarak düşünülebilir. Yani veriyi niteleyen özelliklerdir. Metin sınıflandırması yaparken verilerin sahip olduğu öznitelikler göz önünde bulundurulur. Sınıflandırmanın başarılı bir şekilde

yapılabilmesi için veriyi en iyi şekilde tanımlayan öznitelikler ele alınmalıdır. Bu sebeple sınıflandırma problemlerinde yapılan tahminin gerçeğe yakın olması için kullanılacak özniteliklerin veriyi belirleyici özellikte olması gerekmektedir. Amaç performansı etkileyen, veriyi en çok yansıtan özellikleri seçip, boyutsallığı azaltarak öğrenme performansını artırmaktır [43]. Boyut indirgeme için öznitelik seçim (feature selection) ve öznitelik çıkarım (feature extraction) teknikleri kullanılmaktadır.

Tüm veri setinde aynı olan özniteliklerin ya da veriyi ayırt edici olmayan özelliklerin göz ardı edilip, veriye ait belirli özelliklerin ele alınmasına öznitelik seçimi denir. Yani veriye ait belirlenmiş öznitelik uzayından hangi özniteliklerin sınıflandırma hesaplamalarında kullanılıp hangilerinin göz ardı edileceğine karar verme işlemidir. Kısacası veriye ait özniteliklerden performansa katkısı en fazla olacak daha kaliteli veriler ile bir alt küme oluşturma işlemidir [44]. Yapılan çalışmada Korelasyon Tabanlı Özellik Seçimi (Correlation based Feature Selection – CFS) ve Kazanç Oranı (Gain Ratio – GR) öznitelik seçim teknikleri kullanılmıştır.

Korelasyon tabanlı öznitelik seçim yönteminde özniteliklerin değerini ölçen algoritmanın yanı sıra arama tabanlı algoritma da kullanılmaktadır. Her özelliğin sınıf etiketi tahmin edilirken aralarındaki iç korelasyon değerlerini de dikkate alınır. Bu yaklaşım, iyi özellik altkümeleri ilgili sınıf ile yüksek birbirleri ile düşük korelasyona sahip özelliklerden oluştuğu hipotezine dayanır [45].

Kazanç oranı tekniğinde ise öncelikle bilgi kazancı tüm özellikler için hesaplanır. Sadece, en az ortalama bilgi kazancı kadar performans gösteren özellikler baz alınır ve en iyi kazanç oranını elde eden özellik seçilir. Kazanç oranı hem doğruluk açısından hem de sınıflandırıcı karmaşıklığı açısından bilgi kazancı ölçütüne göre daha iyi performans göstermektedir [20].

Devijver ve Kittler'e göre özellik çıkarımı "sınıf içi model değişkenliğini azaltıp sınıflar arası model değişkenliğini artırarak ham veri ya da bilgiden sınıflandırma amacıyla ayırt edici özelliklerin elde edilmesi" yaklaşımıdır [46]. Öznitelik çıkarımı ile veri setine ait öznitelikler oluşturularak, veriler bu özniteliğe göre etiketlenir. Öznitelik çıkarım işlemi var olan öznitelik uzayındaki özelliklere göre ya da veriden yeni özellikler çıkarılması ile yapılır. Veriden yeni özelliklerin çıkarım

işlemi genellikle görüntü video işleme için kullanılır. Metin sınıflandırmada ise öznitelik çıkarımı çoğunlukla mevcut öznitelik kümesi kullanılarak gerçekleştirilir.

4.2.3. Modelleme

Doğal dil işlemenin önişleme sürecinde, veri seti temizlenip, doğal dilden ayrıştırılarak makinenin anlayacağı formata dönüştürülür. Bu aşamadan sonra veri seti modelde eğitmek için hazır hale gelir. Modelleme aşamasında ele alınan probleme göre makine öğrenimi algoritmaları, derin öğrenme algoritmaları ya da transformer modeller kullanılır. Veri ön işleme süreçleri benzer olsa da her birinde oluşturulması gereken model mimarisi farklıdır. Makine öğrenimi, derin öğrenme ve transformer modeller Bölüm 3'te detaylı olarak açıklanmıştır. Çalışma kapsamında makine öğrenmesi, derin öğrenme ve transformer modeller ile doğal dil işleme modelleri geliştirilmiştir.

4.2.4. Başarım Değerlendirme Metrikleri

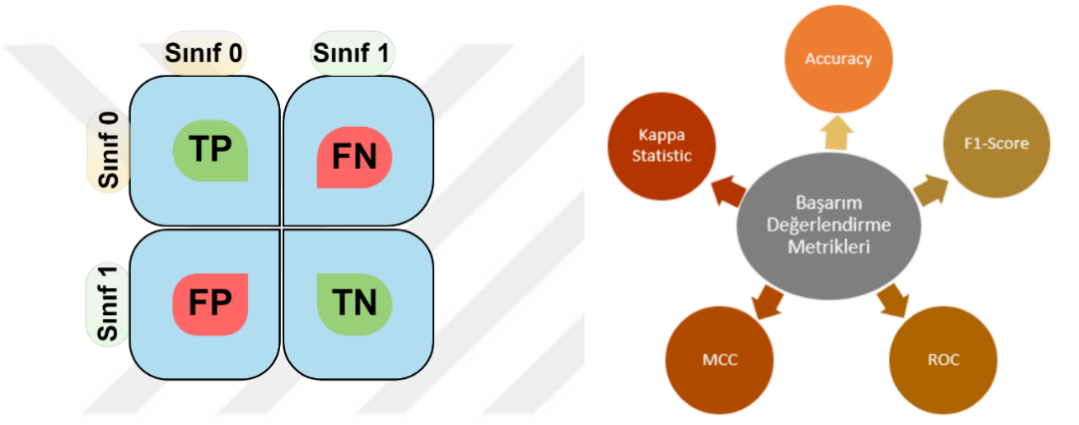
Başarım değerlendirme metrikleri sınıflandırmanın performansını ölçmek için kullanılan araçlar olarak tanımlanır [47]. Bu tanımdan yol çıkarak, başarım değerlendirme metrikleri sınıflandırma modelleri geliştirirken optimum sonuca ulaşmak için anahtar rol oynamaktadır. Ayrıca uygun metrikler ile sınıflandırma performansı yapıldığında sınıflandırmanın ne kadar genellenebilir olduğu ölçülebilir. Her değerlendirme metriği sınıflandırma performansını ölçme amacına hizmet etse de her biri farklı kriterlere göre değerlendirme yapmaktadır. Bu metrikler kullanılmadan sadece karmaşıklık matrisi (confusion matrix) ile sınıflandırmanın genel başarımına ait durum gözlenebilir. Karmaşıklık matrisi tahmin edilen değerler ile gerçek değerler arasındaki ilişkiyi gösteren bir matris olarak ifade edilir. İki sütun ve iki satırdan oluşan basit bir ikili sınıflandırma karmaşıklık matrisi için sınıflardan birinin FR, diğerinin NFR varsayıldığı durumda, karmaşıklık matrisi üzerinde;

- Gereksinimin gerçekte işlevsel gereksinim olduğu (FR), tahmin sonucunun da FR olması durumunda: True-Pozitif (TP),
- Gereksinimin gerçekte FR olduğu ve tahmin sonucunun NFR olması: False-Pozitif (FP),

- Gereksinimin gerçekte NFR ve tahmin sonucunun NFR olması: True-Negatif (TN),
- Gereksinimin gerçekte işlevsel gereksinim olduğu NFR ve tahmin sonucunun FR olması False-Negatif (FN) durumlarına karşılık gelmektedir.

Karmaşıklık matrisi kullanılarak diğer değerlendirme metrikleri hesaplanabilir (precision, recall, F1-score, accuracy, Cohen kappa, vb.)

Karmaşıklık matrisi ve yapılan çalışmada yazılım gereksinim sınıflandırma başarısını değerlendirmek kullanılan metrikler Şekil 4.8'de gösterilmiştir.



Şekil 4.8. Karmaşıklık Matrisi ve Başarım Ölçütleri

A. Doğruluk

Doğruluk (Accuracy – Acc), makine öğreniminde tahmin modellerinde genel başarı kriteri olarak kabul edilmiştir ve Denklem(4)'teki formül ile hesaplanmaktadır [48].

$$Acc = \frac{(TP + TN)}{(TP+TN+FN+FP)} \qquad (4)$$

Denklem(4)'te ifade edildiği gibi doğru yapılan tahminlerin, tüm tahminlere oranı ile hesaplanmaktadır. Genellikle veri setinin dengeli olamdığı durumlarda accuracy oranı model doğruluğunu değerlendirmek için tek başına yeterli değildir ve diğer metrikler ile de değerlendirme yapılmalıdır.

B. F1-Skor

Accuracy ile karşılaştırıldığında F1-Skoru dengesiz dağılımlı veri setlerinde daha doğru sonuç vermektedir. F1 skorunu hesaplamak için Denklem (5) ve Denklem (6)'da formülleri gösterilen recall ve precision metrikleri kullanılmaktadır. Yapılan çalışmadan yola çıkarak FR sınıfına ait gereksinimlerin kaç tanesinin doğru tahmin edildiği recall, FR olarak tahmin edilen bir durumun doğruluğu ise precision olarak tanımlanır. F1 skoru ise recall ve precision değerlerinin harmonik ortalamasıdır. F1 skoru ile uç değerlerin performans başarısını etkilemesinin önüne geçilmiş olur.

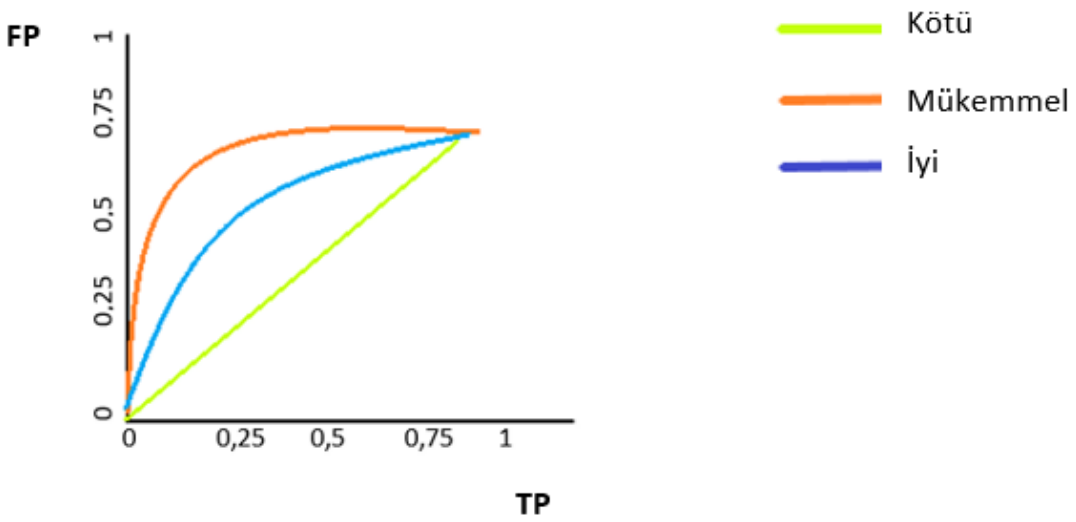
$$Recall = \frac{TP}{TP + FN} \quad (5)$$

$$Precision = \frac{TP}{TP + FP} \quad (6)$$

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \quad (7)$$

C. ROC Eğrisi

ROC eğrisi, özellikle dengesiz veri setlerinde başarılı değerlendirmeler yapan sınıflandırma performansının iki boyutlu bir ölçüsüdür. Pozitif örnekleri doğru şekilde sınıflandırma olasılığının, gerçek negatif örnekleri yanlış sınıflandırma oranına karşı bir çizimi olarak ifade edilir [49]. Yani FR sınıfına ait gereksinimleri doğru sınıflandırma olasılığı ile NFR örneklerini yanlış sınıflandırma durumlarının karşılaştırılmasıdır. Dengesiz veri setlerinde başarılı sonuçlar vermektedir. Şekil 4.9'da görüldüğü üzere ROC eğrisini bir grafik ile gösterildiğinde x ekseni FP, y ekseni ise TP değerlerine karşılık gelmektedir ve ROC eğrisi altında kalan alan 1'e yaklaştıkça model başarımı da doğru orantılı olarak artmaktadır.



Şekil 4.9. ROC Eğrisi

D. *MCC*

Bir modelin performansını MCC (Matthews Correlation Coefficient) metriği ile değerlendirmek için karmaşıklık matrisindeki tüm değerler hesaplamada kullanılmaktadır. Gerçek veriler ile tahmin verileri arasında korelasyon ilişkisine göre, Denklem (8)'de gösterilen şekilde hesplanıp, değerlendirme yapmaktadır. Minimum sınır -1, maksimum sınırı ise +1'dir ve değer 1'e yaklaştıkça performansın arttığını, tahminlerin doğru olduğunu göstermektedir.

$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \qquad (8)$$

E. Kappa

Kappa istatistiği, tahmin edilen değer ile gerçek değer arasındaki uyumu göstermektedir. Ayrıca bu uyumun şans eseri olabileceğini de değerlendirmektedir. MCC metriği gibi -1 ve +1 arasında değer alır ve 1'e ne kadar yakında modelin o kadar başarılı olduğunu gösterir [50]. 0 ise var olan uyumun şans eseri olabileceğini ifade eder.

5. BULGULAR

İki sınıftan oluşan yazılım gereksinimlere ait veri seti üç ayrı bölümde işlenmiştir. Veri seti, 1. bölümde makine öğrenmesi algoritmaları (Bölüm 5.1.), 2. bölümde derin öğrenme algoritmaları (Bölüm 5.2.) son olarak 3. bölümde ise transformer modeller üzerinde eğitilmiştir (Bölüm 5.3.). Bu başlık altında yapılan deneysel çalışmalar sonucu geliştirilen modeller ve elde edilen sonuçlara yer verilmiştir.

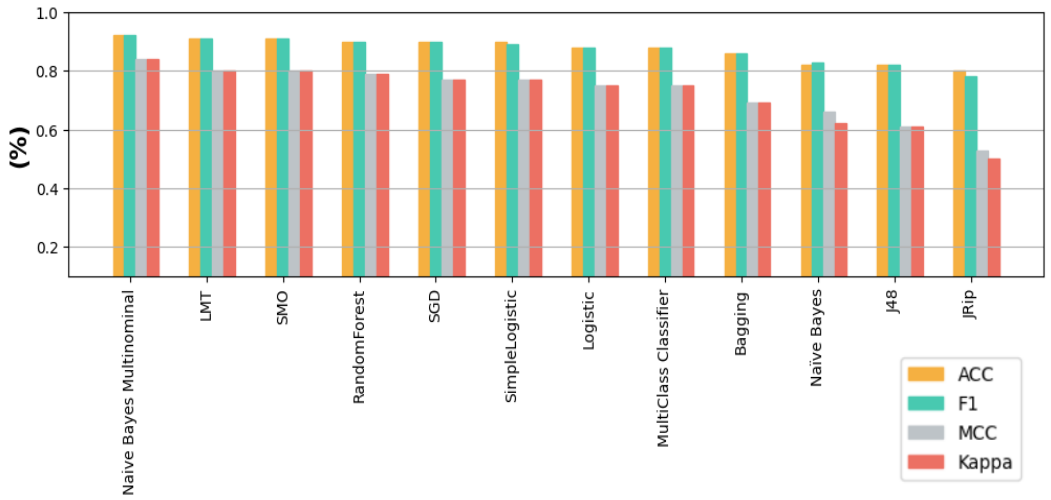
5.1. Makine Öğrenmesi Deneyleri

İlk adım olarak, %80 eğitim ve %20 test verileri ile WEKA içindeki tüm makine öğrenmesi algoritmaları ile deneyler yapılmıştır. Daha sonra aynı algoritmalara özellik seçimi tekniği uygulanarak her iki çalışmada elde edilen sonuçlar karşılaştırılmıştır. Bu algoritmalar içinden en yüksek performansa sahip 12 algoritmaya ait accuracy ve F1- skor, MCC ve Kappa değerleri Tablo 5.1'de gösterilmiştir.

Tablo 5.1. Makine Öğrenmesi Algoritmaları Deney Sonuçları (%)

Model	Accuracy	F1-Skor	MCC	Kappa
Naive Bayes Multinominal	.928	.928	.841	.84
LMT	.914	.914	.809	.808
SMO	.913	.913	.806	.806
RandomForest	.909	.909	.797	.794
SGD	.901	.901	.779	.778
SimpleLogistic	.90	.899	.775	.772
Logistic	.886	.888	.753	.752
MultiClass Classifier	.886	.888	.753	.752
Bagging	.865	.863	.694	.691
Naïve Bayes	.828	.83	.663	.627
J48	.825	.826	.616	.615
JRip	.80	.784	.539	.507

Deney sonuçlarına bakıldığında Tablo 5.1'deki algoritmalar ile geliştirilen modeller benzer sonuçlara sahiptir. Naive Bayes Multinominal ile geliştirilen model %92 accuracy ve %92 F1-skoru ile en performanslı model olmuştur. Jrip ile geliştirilen model ise %80 accuracy ve %78 F1-skoru ile 12 algoritma arasında en düşük başarıma sahip olan algoritma olmuştur. Diğer algoritmalar ile de birbirine yakın başarım elde edilmiştir. Şekil 5.1'de tüm ML algoritmalarmalarına ait başarım değerlendirme metrikleri gözlemlenebilmektedir.



Şekil 5.1. ML Algoritmaları Performans Değerleri

İlk adımın devamında aynı algoritmalara CFS ve GR öznitelik seçim yöntemleri uygulanmış ve algoritmalardaki değişim gözlenmiştir. Tablo 5.2'de öznitelik seçim yöntemleri ile geliştirilen modellere ait sonuçlar gösterilmektedir.

Tablo 5.2. ML Algoritmaları Öznitelik Seçim ile Değerlendirme Sonuçları

Model	Accuracy		F1-Skoru		MCC		Kappa	
	CFS	GR	CFS	GR	CFS	GR	CFS	GR
Naive Bayes Multinominal	.844	.927	.837	.830	.646	.628	.629	.626
LMT	.848	.817	.842	.816	.657	.817	.640	.816
SMO	.840	.915	.834	.915	.635	.811	.621	.810
RandomForest	.846	.807	.84	.805	.652	.807	.634	.805
SGD	.841	.905	.836	.905	.637	.788	.627	0
SimpleLogistic	.913	.904	.841	.904	.658	.785	.637	.783
Logistic	.852	.891	.845	.891	.665	.759	.646	.759

MultiClass Classifier	.852	.891	.845	.83	.665	.759	.646	.759
Bagging	.839	.864	.83	.862	.635	.692	.691	.688
Naïve Bayes	.748	.828	.787	.856	.558	.628	.507	.626
J48	.773	.81	.751	.82	.473	.602	.431	.601
jRIP	.798	.525	.779	.494	.541	.525	.496	.494

Tablo 5.2'ye bakıldığında genel olarak öznitelik seçim tekniklerinin accuracy ve F1 skor değerleri üzerinde belirgin bir etkisi olmamıştır. CFS öznitelik seçim tekniği sadece Simple Logistic algoritmasında accuracy değerini pozitif yönde etkilemiştir. SMO ve Logistic algoritmalarında ise GR öznitelik seçim tekniği accuracy değerinde, SMO ve Simple Logistic algoritmalarında ise F1 skoru üzerinde etkili olmuştur. MCC ve Kappa metrikleri incelendiğinde GR öznitelik seçimi SMO, Random Forest, Simple Logistic ve Logistic algoritmaları başarımında performans göstermiştir. Genel olarak CFS tekniği ile kayda değer herhangi bir değişim olmamıştır. Hatta bazı algoritmalarda ise ilk adımdaki performanslara göre küçük miktarda azalış görülmüştür.

5.2. Derin Öğrenme Deneyleri

Bölüm 5.1'de anlatılan makine öğrenmesi algoritmaları ile geliştirilen model performanslarının yapay sinir ağları kullanılarak geliştirildiğinde nasıl bir değişim olacağı değerlendirilmiştir. Derin öğrenme algoritmalarından CNN, LSTM, Bi-LSTM, GRU, Bi-GRU algoritmaları ile deneyler yapılmıştır. Makine öğrenmesi algoritmaları için kullanılan başarım değerlendirme metrikleri derin öğrenme algoritmaları için de hesaplanmıştır. Elde edilen sonuçlara Tablo 5.3'te yer verilmiştir.

Tablo 5.3. Derin Öğrenme Algoritmaları Deney Sonuçları (%)

Algoritma	**Accuracy**	**F1-Skor**	**MCC**	**Kappa**
CNN	.92	.88	.80	.82
LSTM	.91	.87	.80	.80
Bi-LSTM	.90	.85	.81	.81
GRU	.91	.87	.82	.82
Bi-GRU	.92	.88	.78	.79

Deney sonuçlarına göre yapay sinir ağları kullanılarak geliştirilen modeller benzer sonuçlar vermiştir. En yüksek accuracy ve F1 değerleri sırasıyla %92 ve %88 olarak CNN ve Bi-GRU algoritmalarına aittir. %2 oranında farkla Bi-LSTM, derin öğrenme algoritmaları içinde en düşük accuracy ve F1-skoru değerlerine sahip olsa da performans olarak başarılıdır denilebilir.

5.3. Transformer Model Deneyleri

Çalışmanın son kısmında transformer modeller kullanılarak yazılım gereksinim sınıflandırması yapılmıştır. BERT, BERTürk, DistilBERT ve RoBERTa algoritmaları ile modeller geliştirilmiş ve aynı değerlendirme metrikleri ile sonuçlar karşılaştırılmıştır. Deney sonuçlarına ait değerlere Tablo 5.4'te yer verilmiştir.

Tablo 5.4. Transformer Modelleri Deney Sonuçları (%)

Algoritma	Accuracy	F1-Skor	MCC	Kappa	Roc
BERT	.94	.91	.87	.87	.97
BERTürk	.95	.93	.89	.89	.98
DistilBERT	.93	.90	.85	.85	.97
RoBERTa	.90	.85	.78	.78	.96

Tablo 5.4'teki sonuçlara göre BERTürk %95 accuracy, %93 F1-skoru ile en yüksek ve genellenebilir başarım göstermiştir. BERTürk'ü %94 accuracy, %91 F1 skoru ile BERT takip etmektedir. Ardından sırasıyla DistilBERT ve RoBERTa modelleri %90 ve üzeri oranda bir tahmin başarımı göstermiştir. Transformer modelleri geliştirilirken kullanılan parametreler ve aldıkları değerler Tablo 5.3.1'de detaylı bir şekilde belirtilmiştir.

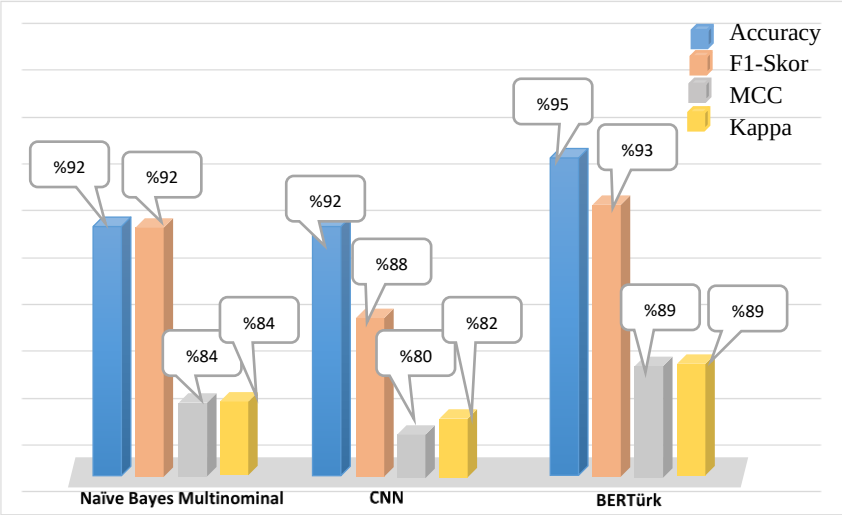
Tablo 5.5. Transformer Model Parameteleri

Algoritma	Layer	Output	Batch Size	Epoch
BERT	TFBertMainLayer, Dropout, Dense	Multiple, Multiple, Multiple	3	5
BERTürk	TFBertMainLayer, Dropout, Dense	Multiple, Multiple, Multiple	3	5

DistilBert	TFDistilBertMainLayer, Dropout, Dense, Dense, DropOut	Multiple, Multiple, Multiple, Multiple, Multiple	3	5
RoBERTa	InputLayer, InputLayer, 0InputLayer, TFRobertaModel, Dropout, Flatten, Dense	[(None, 256)], [(None, 256)], [(None, 256)], TFBaseModelOutputWithPoolingAndCrossAttentions(last_hidden_state=(None, 256, 768), pooler_output=(None, 768), past_key_values=None, hidden_states=None, attentions=None, cross_attentions=None), (None, 256, 768), (None, 196608), (None, 1)	12	3

6. TARTIŞMA ve SONUÇ

Bu çalışmada yazılım geliştirme süreçlerinde kritik bir öneme sahip olan gereksinim analizi sürecini daha hızlı ve en az hata ile yapılmasını sağlamak amacıyla yapay zekanın bir alanı olan doğal dil işleme kullanılarak modeller geliştirilmiştir. Yazılım gereksinimlerin iki sınıflı, işlevsel gereksinim ve işlevsel olmayan gereksinim, olarak sınıflandırılması için yapılan bu çalışmada başarılı ve genellenebilir sonuçlar elde edilmiştir. Makine öğrenmesi algoritmaları geliştirilen modellerde en performanslı algoritma %92 accuracy ve F1-skor tahmin yüzdeleri ile Navie Bayes Multinominal olmuştur. Yapay sinir ağları kullanarak geliştirilen derin öğrenme algoritmalarında en düşük tahmin yüzdesi %90 Bi-LSTM algoritması ile oluşturulan model olmasına karşın bu aşamada da yapılan deneyler başarıyla sonuçlanmıştır. Doğal dil işlemede son zamanlarda sıklıkla kullanılan transformer modeller ile veri seti eğitilmiş ve oldukça performanslı sonuçlara ulaşılmıştır. %95 Accuracy ve %93 F1-skor yüzdeleri ile genellenebilir sınıflandırma başarısına ulaşılmıştır. Üç aşamaya ait en yüksek başarıma sahip algoritmalar ve bu algoritmalara ait başarım değerlendirme metrikleri Şekil 6.1'de gösterilmiştir.

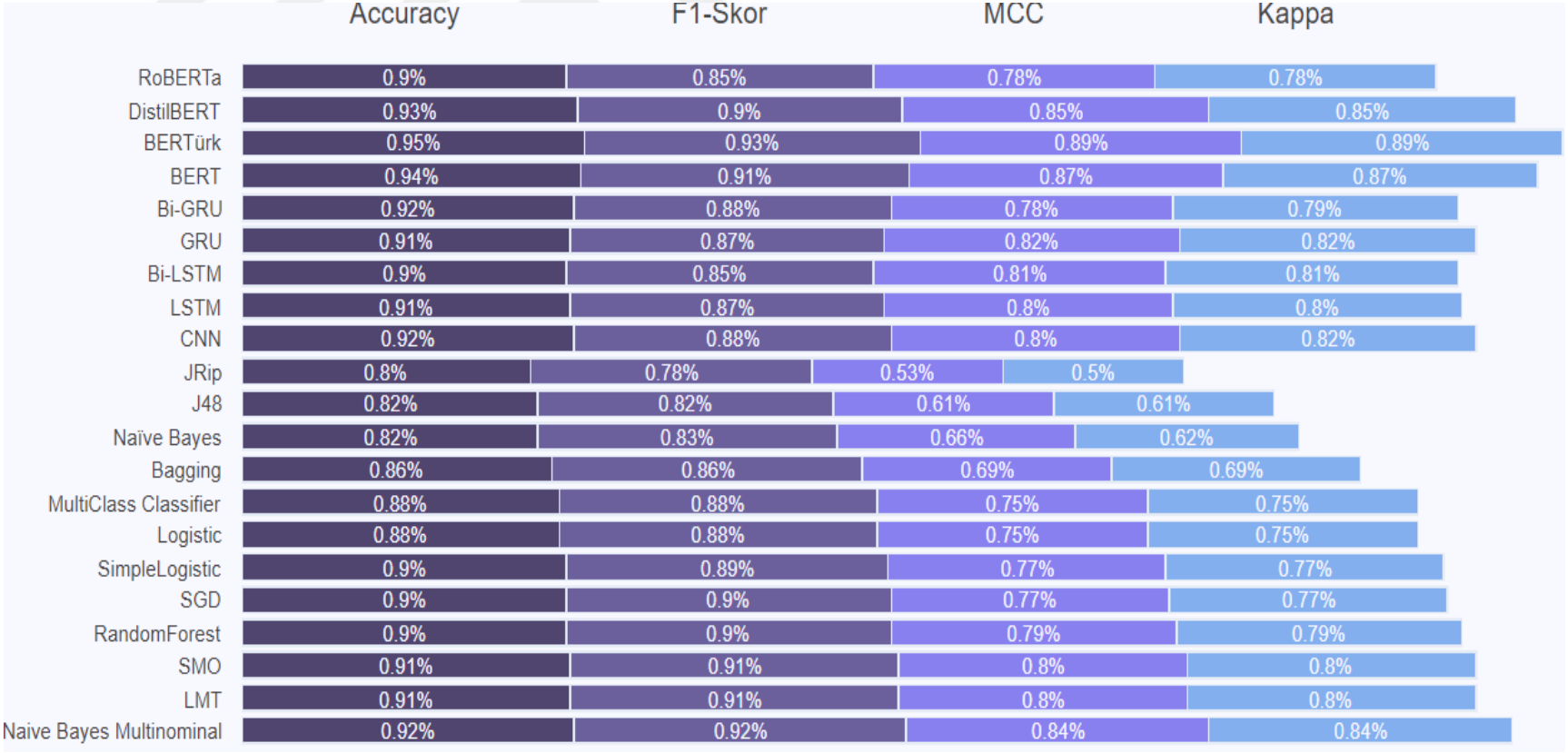


Şekil 6.1. Üç Aşamaya Ait En İyi Performans Sonuçları

Yaklaşık 36 makine öğrenmesi, 5 derin öğrenme ve 4 transformer olmak üzere toplamda 45 farklı algoritma ile yazılım gereksinim sınıflandırma konusunda kapsamlı bir çalışma yapılmıştır. En iyi sonuca sahip 12 ML algoritması, derin öğrenme

algoritmaları ve transformer modellere ait başarım değerlendirmeleri Şekil 6.2'de gözlemlenebilir.

Yazılım gereksinim sınıflandırma probleminin sınıflandırılması konusunda Türkçe çalışmaların yetersiz olması sebebiyle veri setinde bulunan örneklerin özellikle Türkçe metinlerden, farklı platformlardan (web ve mobil projeler), farklı sektörlerden ve gerçek gereksinimlerden oluşturulmuş olması konu ile ilgili yapılan çalışmalara önemli bir katkı sağlayacaktır.



Şekil 6.2. Tüm Algoritmaların Başarım Değerlendirmeleri

REFERANSLAR

[1] Li, B., Li, Z., & Yang, Y. (2021, September). NFRNet: a deep neural network for automatic classification of non-functional requirements. In 2021 IEEE 29th International Requirements Engineering Conference (RE) (pp. 434-435). IEEE.

[2] Bisi, M., & Keskar, K. (2020, December). CNN-BPSO approach to Select Optimal Values of CNN Parameters for Software Requirements Classification. In 2020 IEEE 17th India Council International Conference (INDICON) (pp. 1-6). IEEE.

[3] Alrumaih, H., Mirza, A., & Alsalamah, H. (2018, April). Toward automated software requirements classification. In 2018 21st Saudi Computer Society National Computer Conference (NCC) (pp. 1-6). IEEE.

[4] Navarro-Almanza, R., Juarez-Ramirez, R., & Licea, G. (2017, October). Towards supporting software engineering using deep learning: A case of software requirements classification. In 2017 5th International Conference in Software Engineering Research and Innovation (CONISOFT) (pp. 116-120). IEEE.

[5] Singh, P., Singh, D., & Sharma, A. (2016, December). Classification of non-functional requirements from SRS documents using thematic roles. In 2016 IEEE International Symposium on Nanoelectronic and Information Systems (iNIS) (pp. 206-207). IEEE.

[6] Halim, F., & Siahaan, D. (2019, August). Detecting non-atomic requirements in software requirements specifications using classification methods. In 2019 1st International Conference on Cybernetics and Intelligent System (ICORIS) (Vol. 1, pp. 269-273). IEEE.

[7] Alhindawi, N. T. (2018, August). Information retrieval-based solution for software requirements classification and mapping. In 2018 5th International Conference on Mathematics and Computers in Sciences and Industry (MCSI) (pp. 147-154). IEEE.

[8] Limaylla-Lunarejo, M. I., Condori-Fernandez, N., & Luaces, M. R. (2022, August). Towards an automatic requirements classification in a new Spanish dataset. In 2022 IEEE 30th International Requirements Engineering Conference (RE) (pp. 270-271). IEEE.

[9] Haque, M. A., Rahman, M. A., & Siddik, M. S. (2019, May). Non-functional requirements classification with feature extraction and machine learning: An empirical study. In 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT) (pp. 1-5). IEEE.

[10] Quba, G. Y., Al Qaisi, H., Althunibat, A., & AlZu'bi, S. (2021, July). Software requirements classification using machine learning algorithm's. In 2021 International Conference on Information Technology (ICIT) (pp. 685-690). IEEE.

[11] Young, I. J. B., Luz, S., & Lone, N. (2019). A systematic review of natural language processing for classification tasks in the field of incident reporting and adverse event analysis. International journal of medical informatics, 132, 103971.

[12] Medhat, W., Hassan, A., & Korashy, H. (2014). Sentiment analysis algorithms and applications: A survey. Ain Shams engineering journal, 5(4), 1093-1113.

[13] Mejova, Y. (2009). Sentiment analysis: An overview. University of Iowa, Computer Science Department.

[14] Santhanam, S., & Shaikh, S. (2019). A survey of natural language generation techniques with a focus on dialogue systems-past, present and future directions. arXiv preprint arXiv:1906.00500.

[15] Jiang, J. (2012). Information extraction from text. Mining text data, 11-41.

[16] Sarawagi, S. (2008). Information extraction. Foundations and Trends® in Databases, 1(3), 261-377.

[17] Ali, A. T., Abdullah, H. S., & Fadhil, M. N. (2021). Voice recognition system using machine learning techniques. Materials Today: Proceedings, 1-7.

[18] Shah, H. N. M., Ab Rashid, M. Z., Abdollah, M. F., Kamarudin, M. N., Lin, C. K., & Kamis, Z. (2014). Biometric voice recognition in security system. Indian journal of Science and Technology, 7(2), 104.

[19] Peng, G. C., Alber, M., Buganza Tepole, A., Cannon, W. R., De, S., Dura-Bernal, S., ... & Kuhl, E. (2021). Multiscale modeling meets machine learning: What can we learn?. Archives of Computational Methods in Engineering, 28, 1017-1037.

[20] Demir, M. (2021). Özellik Seçim Yöntemleri Kullanılarak Sınıflandırma Algoritmalarının Performanslarının Karşılaştırılması (Master's thesis, Fen Bilimleri Enstitüsü).

[21] Singh, G., Kumar, B., Gaur, L., & Tyagi, A. (2019, April). Comparison between multinomial and Bernoulli naïve Bayes for text classification. In 2019 International Conference on Automation, Computational and Technology Management (ICACTM) (pp. 593-596). IEEE.

[22] Aborisade, O., & Anwar, M. (2018, July). Classification for authorship of tweets by comparing logistic regression and naive bayes classifiers. In 2018 IEEE International Conference on Information Reuse and Integration (IRI) (pp. 269-276). IEEE.

[23] Sadiq, A. (2021). Intrusion Detection Using the WEKA Machine Learning Tool.

[24] Asif, A., Majid, M., & Anwar, S. M. (2019). Human stress classification using EEG signals in response to music tracks. Computers in biology and medicine, 107, 182-196.

[25] Cahya, R. A., Bachtiar, F. A., & Mahmudy, W. F. (2021). Comparison of Bagging Ensemble Combination Rules for Imbalanced Text Sentiment Analysis. Journal of Information Technology and Computer Science, 6(1), 33-49.

[26] Nematallah, H., Rajan, S., & Cretu, A. M. (2019, October). Logistic model tree for human activity recognition using smartphone-based inertial sensors. In 2019 IEEE SENSORS (pp. 1-4). IEEE.

[27] Shiri, F. M., Perumal, T., Mustapha, N., & Mohamed, R. (2023). A Comprehensive Overview and Comparative Analysis on Deep Learning Models: CNN, RNN, LSTM, GRU. arXiv preprint arXiv:2305.17473.

[28] Fong, V. L. (2018). Software requirements classification using word embeddings and convolutional neural networks.

[29] Wei, W., & Zhao, X. (2023). Fault text classification of on-board equipment in high-speed railway based on labeled-Doc2vec and BiGRU. Journal of Rail Transport Planning & Management, 26, 100372.

[30] Bouschery, S. G., Blazevic, V., & Piller, F. T. (2023). Augmenting human innovation teams with artificial intelligence: Exploring transformer-based language models. Journal of Product Innovation Management, 40(2), 139-153.

[31] Lee, J., Tang, R., & Lin, J. (2019). What would elsa do? freezing layers during transformer fine-tuning. arXiv preprint arXiv:1911.03090.

[32] Acheampong, F. A., Nunoo-Mensah, H., & Chen, W. (2021). Transformer models for text-based emotion detection: a review of BERT-based approaches. Artificial Intelligence Review, 1-41.

[33] Rolon-Mérette, D., Ross, M., Rolon-Mérette, T., & Church, K. (2016). Introduction to Anaconda and Python: Installation and setup. Quant. Methods Psychol, 16(5), S3-S11.

[34] Sjarif, N. N. A., Azmi, N. F. M., Chuprat, S., Sarkan, H. M., Yahya, Y., & Sam, S. M. (2019). SMS spam message detection using term frequency-inverse document frequency and random forest algorithm. Procedia Computer Science, 161, 509-515.

[35] Ngiam, K. Y., & Khor, W. (2019). Big data and machine learning algorithms for health-care delivery. The Lancet Oncology, 20(5), e262-e273.

[36] Kashina, M., Lenivtceva, I. D., & Kopanitsa, G. D. (2020). Preprocessing of unstructured medical data: the impact of each preprocessing stage on classification. Procedia Computer Science, 178, 284-290.

[37] Grefenstette, G. (1999). Tokenization. Syntactic Wordclass Tagging, 117-133.

[38] Sueno, H. T., Gerardo, B. D., & Medina, R. P. (2020). Converting text to numerical representation using modified Bayesian vectorization technique for multi-class classification. International Journal, 9(4), 10-30534.

[39] Çelik, Ö., & Koç, B. C. (2021). TF-IDF, Word2vec ve Fasttext Vektör Model Yöntemleri ile Türkçe Haber Metinlerinin Sınıflandırılması. Dokuz Eylül Üniversitesi Mühendislik Fakültesi Fen ve Mühendislik Dergisi, 23(67), 121-127.

[40] Etaiwi, W., & Naymat, G. (2017). The impact of applying different preprocessing steps on review spam detection. Procedia computer science, 113, 273-279.

[41] F. Gürcan, "Multi-Class Classification of Turkish Texts with Machine Learning Algorithms," 2018 2nd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), Ankara, Turkey, 2018, pp. 1-5, doi: 10.1109/ISMSIT.2018.8567307.

[42] Hendrawan, I. R., Utami, E., & Hartanto, A. D. (2022, December). Comparison of Word2vec and Doc2vec Methods for Text Classification of Product Reviews. In 2022 6th International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE) (pp. 530-534). IEEE.

[43] Remeseiro, B., & Bolon-Canedo, V. (2019). A review of feature selection methods in medical applications. Computers in biology and medicine, 112, 103375.

[44] Shah, F. P., & Patel, V. (2016, March). A review on feature selection and feature extraction for text classification. In 2016 international conference on wireless communications, signal processing and networking (WiSPNET) (pp. 2264-2268). IEEE.

[45] Budak, H. (2018). Özellik seçim yöntemleri ve yeni bir yaklaşım. Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü Dergisi, 22, 21-31.

[46] Trier, Ø. D., Jain, A. K., & Taxt, T. (1996). Feature extraction methods for character recognition-a survey. Pattern recognition, 29(4), 641-662.

[47] Hossin, M., & Sulaiman, M. N. (2015). A review on evaluation metrics for data classification evaluations. International journal of data mining & knowledge management process, 5(2), 1.

[48] Bratko, I. (1997). Machine learning: Between accuracy and interpretability (pp. 163-177). Springer Vienna.

[49] Rakotomamonjy, A. (2004, August). Optimizing Area Under Roc Curve with SVMs. In ROCAI (pp. 71-80).

[50] Özhan, E. (2020). Makine öğrenmesi yöntemleri ile web'den bilgi çıkarımı sürecinin iyileştirilmesi. Afyon Kocatepe Üniversitesi Uluslararası Mühendislik Teknolojileri ve Uygulamalı Bilimler Dergisi, 3(2), 52-59.