
GAZİ ÜNİVERSİTESİ
Gazi
1926

METİN MADENCİLİĞİ YÖNTEMİ İLE ARAŞTIRMA MAKALESİ SINIFLANDIRMASI

Tuğba GÜRBÜZ

YÜKSEK LİSANS TEZİ
BİLİŞİM SİSTEMLERİ ANA BİLİM DALI

GAZİ ÜNİVERSİTESİ
BİLİŞİM ENSTİTÜSÜ

MART 2022

Tuğba GÜRBÜZ tarafından hazırlanan "METİN MADENCİLİĞİ YÖNTEMİ İLE ARAŞTIRMA MAKALESİ SINIFLANDIRMASI" adlı tez çalışması aşağıdaki jüri tarafından OY BİRLİĞİ ile Gazi Üniversitesi Bilişim Sistemleri Ana Bilim Dalında YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Danışman: Doç. Dr. Çelebi ULUYOL

Bilgisayar ve Öğretim Teknolojileri Eğitimi Bölümü, Gazi Üniversitesi

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum. ...……

Başkan: Doç. Dr. Oktay YILDIZ

Bilgisayar Mühendisliği, Gazi Üniversitesi

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum. ……...

Üye: Dr. Öğr. Üyesi Hakan ÖZKÖSE

Yönetim Bilişim Sistemleri, Bartın Üniversitesi

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum. ……...

Tez Savunma Tarihi: ...../……./ 2022

Jüri tarafından kabul edilen bu tezin Yüksek Lisans Tezi olması için gerekli şartları yerine getirdiğini onaylıyorum.

……..……

Prof. Dr. Aslıhan TÜFEKCİ

Bilişim Enstitüsü Müdürü

ETİK BEYAN

Gazi Üniversitesi Bilişim Enstitüsü Tez Yazım Kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- Tez içinde sunduğum verileri, bilgileri ve dokümanları akademik ve etik kurallar çerçevesinde elde ettiğimi,
- Tüm bilgi, belge, değerlendirme ve sonuçları bilimsel etik ve ahlak kurallarına uygun olarak sunduğumu,
- Tez çalışmasında yararlandığım eserlerin tümüne uygun atıfta bulunarak kaynak gösterdiğimi,
- Kullanılan verilerde herhangi bir değişiklik yapmadığımı,
- Bu tezde sunduğum çalışmanın özgün olduğunu,

bildirir, aksi bir durumda aleyhime doğabilecek tüm hak kayıplarını kabullendiğimi beyan ederim.

Tuğba GÜRBÜZ

...../……/2022

METİN MADENCİLİĞİ YÖNTEMİ İLE ARAŞTIRMA MAKALESİ SINIFLANDIRMASI

(Yüksek Lisans Tezi)

Tuğba GÜRBÜZ

GAZİ ÜNİVERSİTESİ

BİLİŞİM ENSTİTÜSÜ

Mart 2022

ÖZET

Bu çalışma kapsamında metin madenciliği yöntemi ile araştırma makalelerinin konularına göre sınıflandırılması amaçlanmaktadır. Çalışma için kullanılan veri seti, "Web Madenciliği" yöntemi ile Python kodu yazılarak elde edilen araştırma makalelerinden oluşmaktadır. Metin koleksiyonu, DergiPark-Akademik internet sitesindeki "Tıp", "Sosyal", "Temel Bilimler" ve "Mühendislik" konu başlıklarındaki 16 konu alanındaki makaleleri içermektedir. Makale verileri, yazım dili İngilizce ve Türkçe olan makaleler olmak üzere iki ayrı metin koleksiyonundan oluşturulmuştur. Metin madenciliği süreçlerinin ilk adımı olan metin koleksiyonu oluşturulduktan sonra, her iki metin verilerinin öz (abstract) bölümlerine metin ön işleme ve öz nitelik seçimi adımları uygulanmıştır. Sonrasında veri madenciliğinde kullanılan, denetimli (supervised) makine öğrenmesi sınıflandırma algoritmalarından Naive Bayes, Rastgele Orman ve Destek Vektör Makinesi algoritmaları kullanılarak makaleler konu alanlarına göre sınıflandırılmıştır. Metin madenciliğinin son adımı olan değerlendirme ve yorumlama aşamasında ise, uygulama ile her iki veri seti için algoritmaların çalışma performanslarının değerlendirilmesi ve makale yazım diline göre karşılaştırılması makine öğrenmesi performans ölçütlerine göre yapılmıştır. Araştırma bulgularına göre algoritmaların çalışma süreleri İngilizce ve Türkçe veri setleri için birbirine yakın süreler olup; en hızlı çalışan algoritma Naive Bayes iken en yavaş çalışan algoritma Destek Vektör Makinesi olmuştur. Her iki yazım dilindeki algoritma sonuçları birbirine yakın değerler olmakla birlikte genellikle en iyi sınıflandırma başarısına sahip algoritma Destek Vektör Makinesidir.

Bilim Kodu	:	92429
Anahtar Kelimeler	:	Metin madenciliği, web madenciliği, bilgi çıkarımı, makine öğrenimi, denetimli öğrenme
Sayfa Adedi	:	120
Danışman	:	Doç. Dr. Çelebi ULUYOL

RESEARCH ARTICLE CLASSIFICATION WITH TEXT MINING METHOD

(M. Sc. Thesis)

Tuğba GÜRBÜZ

GAZİ UNIVERSITY

INFORMATICS INSTITUTE

March 2022

ABSTRACT

Within the scope of this thesis, it is aimed to classify the research articles according to their subjects with the text mining method. The data set used for the study consists of research articles that obtained by writing Python code with a state of the art web mining method. The collection of texts includes articles in 16 subject areas under the titles of "Medicine", "Social", "Basic Sciences" and "Engineering" on the DergiPark-Academic website. The article data was created from two separate text collections: Articles written in English and Turkish. After creating the text collection, which is the first step of the text mining processes, text preprocessing and feature selection steps were applied to the abstract parts of both text data. Then, the articles were classified according to their subject areas by employing Naive Bayes, Random Forest and Support Vector Machine algorithms from supervised machine learning classification algorithms used in data mining. In the evaluation and interpretation stage, which is the last step of text mining, evaluation of the performance of algorithms for both data sets and comparison according to article writing language was made according to the machine learning performance criteria with the application. According to the research results, the working times of the algorithms were close to each other for the English and Turkish data sets. The fastest running algorithm is Naive Bayes while the slowest running algorithm is Support Vector Machine. On the other hand, Support Vector Machine dominates with the best accuracy result among others. Algorithm performance in both Turkish and English languages do not differ remarkably in terms of accuracy and speed.

Bilim Kodu	:	92429
Anahtar Kelimeler	:	Text mining, web mining, information extraction, machine learning, supervised learning
Sayfa Adedi	:	120
Danışman	:	Doç. Dr. Çelebi ULUYOL

TEŞEKKÜR

Bu tezin belirlenmesi ve hazırlanması sürecinde yönlendirme ve katkıları ile her zaman destek vererek çalışmanın akademik ve bilimsel değerlere bağlı bir şekilde sonuçlanmasını sağlayan danışman hocam Sayın Doç. Dr. Çelebi ULUYOL'a, Yüksek Lisans eğitimim süresince beni her açıdan destekleyerek maddi ve manevi yanımda olan annem, babam ve meslektaşım olan kardeşime verdikleri motivasyon ve desteklerinden dolayı teşekkür ederek şükranlarımı sunarım.

İÇİNDEKİLER

Sayfa

Sayfa

Sayfa

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil **Sayfa**

TABLOLARIN LİSTESİ

Tablo **Sayfa**

KISALTMALAR

Bu çalışmada kullanılmış kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

Kısaltmalar	Açıklamalar
ACC	Doğruluk (Accuracy)
ARFF	Öz nitelik ilişkisi dosyası biçimi (Attribute relation file format)
CRM	Müşteri ilişkileri yönetimi (Customer relationship management)
CSV	Virgülle ayrılmış değerler (Comma-Separated values)
ERR	Hata (Error)
FN	Yanlış negatif (False negative)
FP	Yanlış pozitif (False positive)
GMM	Gauss karışım modeli (Gaussian mixture model)
HTML	Hiper metin işaretleme dili **(**Hypertext markup language)
HTTP	Hiper metin transfer protokolü (Hyper text transfer protocol)
IDF	Ters doküman frekansı (Inverse document frequency)
KDD	Veri tabanlarından bilgi keşfi **(**Knowledge discovery in databases**)**
KDT	Metinden bilgi keşfi **(**Knowledge discovery from text)
KNIME	Konstanz bilgi madencisi **(**Konstanz ınformation miner)
K-NN	K-En yakın komşu algoritması **(**K-Nearest neighbours algorithm)
NLTK	Doğal dil araç seti (Natural language toolkit)
NN	En yakın komşular yöntemi (Nearest neighbors)
SaaS	Hizmet olarak yazılım **(**Software as a Service)
SGD	Stokastik gradyan inişi **(**Stochastic gradient descent algorithm)
SVM	Destek vektör makinesi **(**Support vector machine)

Kısaltmalar	Açıklamalar
TF	Terim frekansı **(**Term frequency)
URL	Standart kaynak bulucu **(**Uniform resource locator)
Web	Ağ
Weka	Bilgi analizi için waikato ortamı **(**Waikato environment for knowledge analysis)
WWW	Geniş dünya ağı **(**Word wide web)
XML	Genişletilebilir işaretleme dili **(**Extensible markup language)

1. GİRİŞ

Günümüz dünyasında bilgisayar veri tabanlarında oluşturulan ve depolanan veriler olağanüstü bir şekilde genişlerken, veri tabanlarından yararlı bilgileri elde etmek önemli bir hale gelmektedir. Geleneksel istatistiksel teknikler, küçük veri kümeleri ve yönetilebilir sayıda değişken içeren problemler için faydalı ve etkili olmaya devam etse de milyonlarca kaydın ve binlerce değişkenin bulunduğu üst problemlere uygulandığında ölçeklenebilirlik barikatıyla karşılaşırlar [1]. İstatistiklerin ötesine geçerek büyük miktardaki veriyi incelemeyi amaçlayan ve sıklıkla kullanılan bir terim olan veri madenciliği; basit veri analizinden çok daha fazla anlam içermektedir. Yaygın olarak Veri Tabanlarından Bilgi Keşfi olarak da bilinen veri madenciliği metinler, web, veri tabanları gibi veri kaynaklarındaki verilerden; örtük, önceden bilinmeyen, potansiyel olarak faydalı bilgilerin, kalıpların ve ilişkilerin çıkarılmasıdır [2]. Çok disiplinli bir alan olan veri madenciliği; yapay zekâ, veri tabanı teorisi, veri görselleştirme, pazarlama, matematik, işlem araştırması, örüntü tanıma ve istatistik gibi alanlardan yararlanır [1]. Veri madenciliği günümüzde sağlık sektörü, eğitim, güvenlik, iş dünyası, pazarlama, ticaret, astronomi, biyoloji, spor, internet, bankacılık, telekomünikasyon gibi çok çeşitli alanlarda uygulanmaktadır.

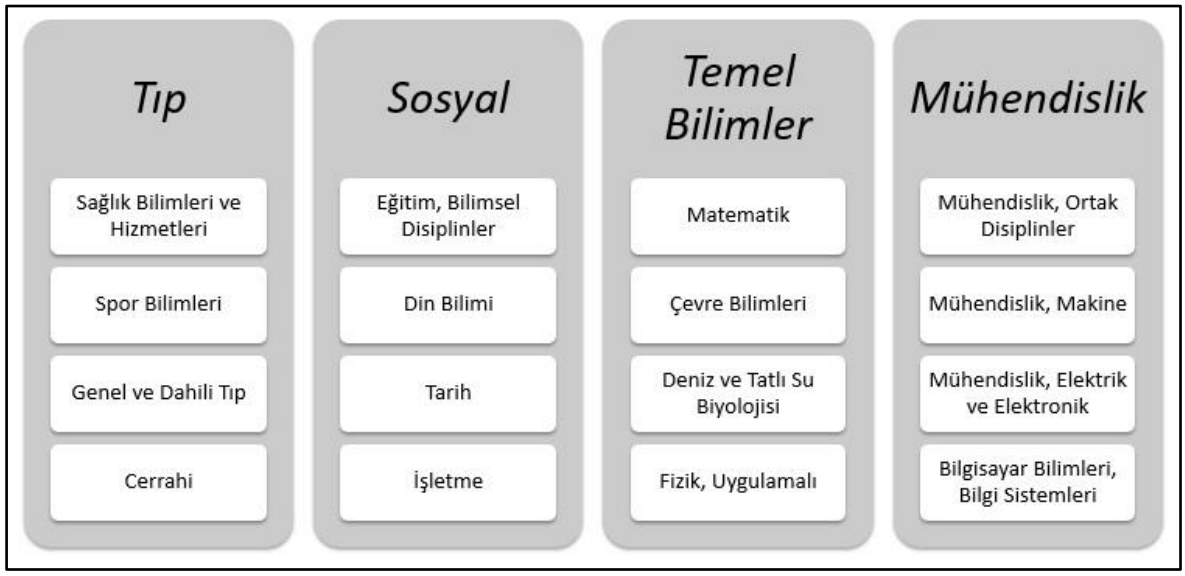
Veri madenciliğinin nasıl ve neden çalıştığını anlamak için birkaç temel kavramı anlamak önemlidir. Veri madenciliği dört temel yönteme dayanmaktadır: sınıflandırma, kümeleme, tahmin ve görselleştirme. Sınıflandırma, ilişkileri ve kümeleri tanımlar ve incelenen konuları ayırır. Kümeleme, kategorik sonuçları işlemek için kural tüme varım algoritmalarını kullanır. Tahmin, tahmine dayalı işlevleri veya olasılığı içerir ve sürekli sonuç değişkenleriyle ilgilenir. Görselleştirme, matematiksel olarak oluşturulan kuralları ve puanları göstermek için etkileşimli grafikler kullanır ve pasta veya çubuk grafiklerden çok daha karmaşıktır. Görselleştirme de öncelikle matematiksel koordinatların üç boyutlu coğrafi konumlarını göstermek için kullanılır [3].

Metin madenciliği, büyük veri tabanlarından farklı desenler bulmaya çalışan veri madenciliğinin bir varyasyonudur. Metin madenciliği farklı yazılı kaynaklardan otomatik olarak bilgi ayıklayarak önceden bilinmeyen yeni bilgilerin bilgisayar ortamı ile keşfedilmesidir. Normal veri madenciliği ile metin madenciliği arasındaki temel fark; metin madenciliğinde örüntülerin yapılandırılmış gerçek veri tabanlarından değil, doğal dil metninden çıkarılmasıdır [4].

Araştırmanın amacı

Çalışma kapsamında, DergiPark-Akademik internet sitesindeki araştırma makalelerinin metin madenciliği yöntemi ile sınıflandırılması amaçlanmaktadır. DergiPark-Akademik internet sitesindeki araştırma makaleleri, "Web Madenciliği" yöntemi ile Python kodu yazılarak elde edilmiştir.

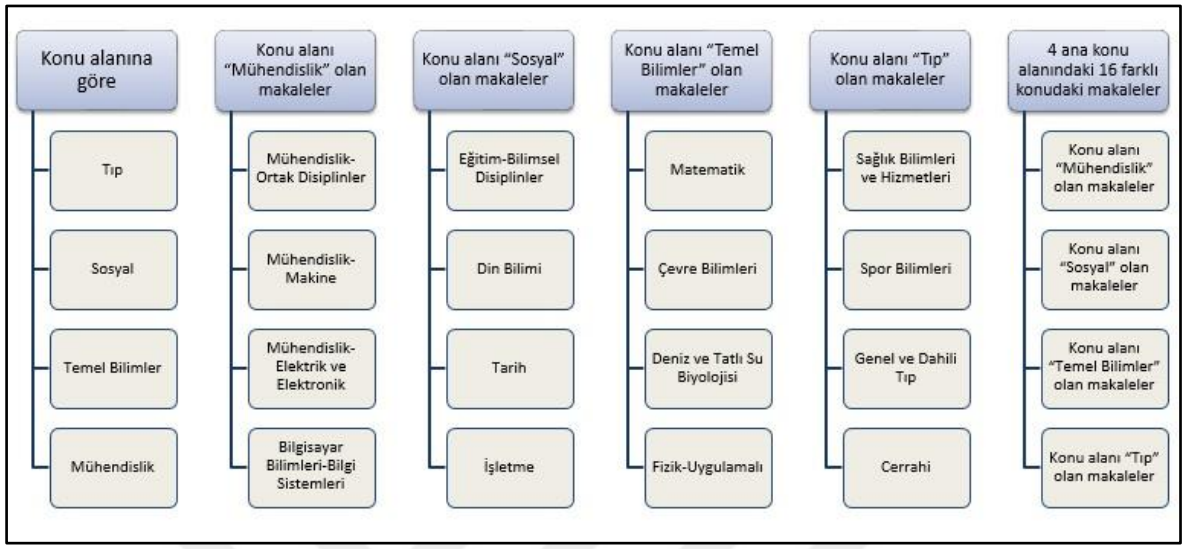
Şekil 1.1 de gösterildiği gibi veri seti, "Tıp", "Sosyal", "Temel Bilimler" ve "Mühendislik" konu başlıklarındaki 16 konu alanındaki makaleleri içermektedir:



Şekil 1.1. Konu alanlarına göre veri seti

Şekil 1.1'de konu alanlarına göre dağılımı verilen makale verileri, yazım dili İngilizce ve Türkçe olan makaleler olmak üzere iki ayrı metin koleksiyonundan oluşturulmuştur. Elde edilen İngilizce makalelerin sayısı 1897, Türkçe makalelerin sayısı 1905'tir. Metin madenciliği süreçlerinin ilk adımı olan metin koleksiyonu oluşturulduktan sonra, her iki metin verilerinin öz (abstract) bölümlerine metin ön işleme ve öz nitelik seçimi adımları uygulanmıştır. Sonrasında veri madenciliğinde kullanılan, denetimli (supervised) makine öğrenmesi sınıflandırma algoritmalarından Naive Bayes, Rastgele Orman ve Destek Vektör Makinesi algoritmaları kullanılarak makaleler konu alanlarına göre sınıflandırılmıştır. Metin madenciliğinin son adımı olan değerlendirme ve yorumlama aşamasında ise, uygulama ile her iki veri seti için algoritmaların çalışma performansları ve makale yazım diline göre karşılaştırılması makine öğrenmesi performans ölçütlerine göre yapılmıştır.

Tez kapsamında kullanılan veri seti için yapılan sınıflandırma çeşitleri Şekil 1.2'de gösterilmektedir:



Şekil 1.2. Veri seti sınıflandırma çeşitleri

Şekil 1.2'de veri seti için yapılan sınıflandırma çeşitleri gösterilmektedir. Araştırma da İngilizce ve Türkçe makale veri setleri için; 6 farklı sınıflandırma seçimi ve her bir sınıflandırma için Naive Bayes, Rastgele Orman ve Destek Vektör Makinesi sınıflandırma algoritmaları kullanılmıştır. Her bir veri setinde kullanılan sınıflandırmalardan birincisi 4 ana konu alanına (Tıp, Sosyal, Temel Bilimler, Mühendislik) göre yapılmıştır. Sonrasında her bir konu alanı ayrı ayrı 4 alt konuya göre sınıflandırılmıştır: Tıp konu alanı (Sağlık Bilimleri ve Hizmetleri, Spor Bilimleri, Genel ve Dahili Tıp, Cerrahi), Sosyal konu alanı (Eğitim-Bilimsel Disiplinler, Din Bilimi, Tarih, İşletme), Temel Bilimler konu alanı (Matematik, Çevre Bilimleri, Deniz ve Tatlı Su Biyolojisi, Fizik-Uygulamalı), Mühendislik konu alanı (Mühendislik-Ortak Disiplinler, Mühendislik-Makine, Mühendislik-Elektrik ve Elektronik, Bilgisayar Bilimleri-Bilgi Sistemleri). Sonuncusu ise metinlerin 16 alt konuya göre sınıflandırılmasıdır. Uygulanan algoritmalarda veri setinin %75'i deneme, %25'i test verisi olmak üzere ikiye ayrılmıştır.

Araştırmanın önemi

Gelişmiş bir iletişim aracı olarak kullanılan dil ile hayatın her alanında yazılı iletişim yapılabilmektedir. Kitaplar, gazeteler, reklamlar, internet siteleri, bloglar, sosyal platformlar, forumlar, anketler, elektronik postalar, mektuplar, dergiler, makaleler, sözleşmeler, kompozisyonlar ve dilekçeler yazılı iletişim örneklerinden birkaçıdır. Bilgi teknolojileri kullanımıyla yazılı iletişim verileri hızla artmaktadır. Yapısal olmayan bu verilerin büyük bir çoğunluğu metinsel verilerden oluşmaktadır. Bu metinsel veriler, kullanılan iş ile ilgili bilgilerin %85'inden daha fazlasını oluşturmaktadır [5]. Genellikle yapılandırılmamış olan bu verilerin içerisinde ise ihtiyaç olan birçok sorunun cevabı bulunmaktadır. Metin madenciliği; insan eliyle ayrıştırılamayan ya da ayrıştırılması çok zaman alacak bu verileri, insan hatalarını en aza indirerek otomatik ve hızlı bir şekilde nitelikli bir hale getirmek için kullanılmaktadır. Metin madenciliği ile yapılandırılmamış birçok metin kaynağı yapılandırılarak anlamlı bilgiler elde edilebilmektedir. Anlamlı bilgileri açığa çıkarırken, bir yandan doğal dildeki çok sayıdaki kelime ve dil yapısı ile uğraşıp diğer yandan belirsizlik ve muğlaklık ile başa çıkmayı sağlayan yöntemler uygulamaktadır [6]. Metin verilerinin anlamlı hale getirilmesi akademik çalışma yapanlar, iş hayatında bulunanlar ve yazılı iletişim kuran herkes için önem arz etmektedir. Ayrıca, metin madenciliği ile metinlerin otomatik bir şekilde özetlenmesi, dokümanlar arasında benzer olanların kümelenmesi, dokümanlardan ilgili dokümanı özetleyebilecek bazı kavramların çıkarılabilmesi gibi bazı tekniklerin uygulanabilmesi; metin madenciliğinin gerekliliğini ve önemini göstermektedir [5].

Metin verilerinden anlamlı veri elde etmek için metin madenciliği yapan herkesin sorabileceği bazı sorular aşağıda verilmiştir:

- Verileri nereden ve nasıl elde edebilirim? (Hazır veri seti mevcut değilse)
- Çalışmamda sorduğum soruların cevaplarını bulabilmek için verileri işlerken hangi metin madenciliği tekniklerini uygulamalıyım?
- Metin madenciliği sürecinde hangi programı kullanmalıyım?
- Metin ön işleme sürecinde ulaşmak istediğim sonuç için hangi algoritmaları kullanmalıyım?
- İşlediğim verileri hangi makine öğrenmesi algoritmalarını kullanarak sınıflandırabilirim?

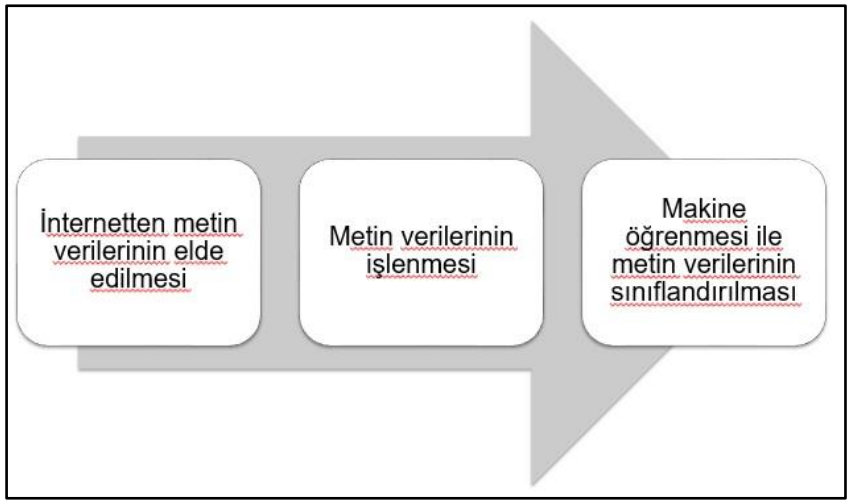
- Sınıflandırdığım metin verilerindeki anlam ve ilişkiyi nasıl değerlendirmeliyim?
- Sınıflandırma için kullandığım makine öğrenmesi algoritmalarının performanslarını nasıl ölçebilirim?

Metin verilerini anlamlı hale getirmek isteyen herkes içerisinde akademik çalışma yapan öğrenciler ya da akademisyenler de mevcuttur. Akademik çalışma yapan birisi, bilimsel araştırma sürecinde literatür araştırması yapmaktadır. Bu süreç boyunca çalıştığı konuya özgü araştırma makalelerini bulup okumaya ihtiyacı olacaktır. Bu aşamada sorabileceği bazı sorular aşağıda verilmiştir:

- Çalıştığım konuya özgü makaleleri incelemem gerektiği için bu makaleleri nasıl elde edebilirim?
- Çalışmak istediğim konu ile ilgili hangi alanda ne tür araştırmalar yapılmış?
- Hangi alandaki yayınlanan yayınlar fazlayken, hangi alandaki çalışmalarda eksiklikler mevcut?
- Çalıştığım konu ile ilgili yayınlanan makaleler ağırlıkla hangi dilde yazılmış?
- Yayınlanan makalelere özgü anahtar kelimeleri nasıl bulabilirim ve bunlar nelerdir?
- Makalelerin genel dağılımı ve konulara göre sınıflandırması nasıl?

Bu tez çalışmasında hem metin madenciliği yapmak isteyen herkesin hem de bilimsel araştırma yapan öğrenci ya da akademisyenlerin sorabileceği soruların cevabını bulabilmek için bir metin madenciliği uygulaması yapılmıştır.

Tez kapsamı Şekil 1.3'te gösterildiği gibi 3 aşamaya ayrılabilir:



Şekil 1.3 Tez kapsamı aşamaları

Şekil 1.3'te tezin aşamaları gösterilmektedir. İlk aşama internetten metin verilerinin elde edilmesi, ikinci aşama metin verilerinin işlenmesi ve üçüncü aşama ise makine öğrenmesi ile metin verilerinin sınıflandırılması sürecidir.

İlk aşama internetten verilerin elde edilmesi sürecidir. Hazır veri seti olmayan bir metin madencisi için internet çok büyük ve geniş bir kaynaktır. Akademik çalışma yapan birisi ise araştırma makalelerinin yayınlandığı internet sitelerindeki verileri kolay ve hızlı bir şekilde elde edebilir. Bu süreçte, metin madenciliği yapan birisi "Verileri nereden ve nasıl elde edebilirim?" sorusununun cevabını bulabilir. Akademik çalışma yapan birisi ise "Çalıştığım konuya özgü makaleleri incelemem gerektiği için bu makaleleri nasıl elde edebilirim?" sorusuna yanıt bulabilir.

İkinci aşama metin verilerinin yazım diline göre işlenmesi sürecidir. Çalışmada hem İngilizce hem Türkçe metin verileri için metin işleme yapılmıştır. Bir metin madencisi "Çalışmamda sorduğum soruların cevaplarını bulabilmek için verileri işlerken hangi metin madenciliği tekniklerini uygulamalıyım?" sorusunun cevabını bu tez çalışması içerisinde bulabilir ve bulduğu cevaba göre ilgili metin madenciliği tekniklerini uygulayabilir. Bu süreçte kullanılan programlar hakkında genel bir fikir sahibi olabilir. Eğer Python programlama dilini kullanmayı tercih edecek olursa çalışma için yazılan Python kodundan faydalanarak kendi problemine özgü çözüm yöntemlerini üretebilir. Akademik çalışma yapan birisi ise ilk aşamada elde ettiği verileri analiz ederek, araştırma alanı ile ilgili yapılan çalışmalar hakkında fikir sahibi olabilir. Metin işleme süreçlerinde, makalenin anlam değeri yüksek olan anahtar kelimelerini bulabilir. Ayrıca yapılan çalışmaların genellikle hangi yazım dilinde yapıldığı ve hangi alanda fazla ya da eksik olduğu hakkında bir görüşü olabilir.

Üçüncü aşama olan makine öğrenmesi ile metinlerin sınıflandırılması yapılmaktadır. Bu aşama da bir metin madencisi "İşlediğim verileri hangi makine öğrenmesi algoritmalarını kullanarak sınıflandırabilirim?" sorusuna cevap bulabilecektir. Öncelikle makine öğrenmesi algoritmaları hakkında bilgi edinerek, çalışması için hangi algoritmayı kullanması gerektiği hakkında fikir sahibi olabilir. Akademik çalışma yapan birisi de, bu çalışma da yapıldığı gibi metin kaynaklarını konusuna ya da istediği farklı bir alana göre sınıflandırabilecektir. Sınıflandırma yapan birisi, çalışmasında kullandığı algoritmanın performansını nasıl ölçebileceğini de öğrenebilecektir.

Metin madenciliği yapmak isteyen birisi, bu tez çalışmasında uygulanan aşamaları değiştirerek ya da geliştirilerek kendi çalışmasında uygulayabilir. Problemine özgü olarak, her türlü web madenciliği, metin madenciliği ve denetimli makine öğrenmesi aşamalarından gerekli olanı kullanabilir.

Tez 8 bölümden oluşmaktadır. Giriş kısmının ardından tezin ikinci bölümünde; metin madenciliğinin tanımı, metotları, uygulama alanları ve metin madenciliği süreçlerinde kullanılan programlar incelenmiştir. Tezin üçüncü bölümünde metin madenciliği süreçlerinin beş aşaması sırasıyla açıklanmıştır: metin koleksiyonu oluşturma, metin ön işleme, özellik seçme, veri madenciliği, değerlendirme ve yorumlama. Tezin dördüncü bölümünde ilgili literatür araştırması yapılmıştır. Tezin beşinci bölümü olan materyal ve yöntem kısmında, öncelikle uygulamada kullanılan programlama dili olan Python anlatılmıştır. Sonrasında ise veri elde etme ve veri seti oluşturma süreci açıklanmıştır. Tezin altıncı bölümü olan bulgular ve yorum aşamasında, metin sınıflandırma işlemleri yapılmıştır. Bu aşamada yazım dili İngilizce ve Türkçe olan makale veri setlerinin öz (abstract) bölümüne, denetimli makine öğrenmesi algoritmalarından Naive Bayes, Rastgele Orman ve Destek Vektör Makinesi yöntemleri kullanılarak; makaleler konu alanlarına göre sınıflandırılmıştır. Tezin yedinci bölümünde, yapılan tez çalışması ile ilgili benzer çalışmalar incelenip tartışılmıştır. Tezin son kısmı olan sekizinci bölüm, sonuç ve öneriler aşamasıdır. Burada, çalışmanın sonuçları ve yorumları açıklanarak, çalışmanın zorlukları ve gelecek çalışmalara dair önerilerden bahsedilmiştir. Tezin ek kısmında ise çalışma için yazılan Python kodları verilmektedir.

2. METİN MADENCİLİĞİ

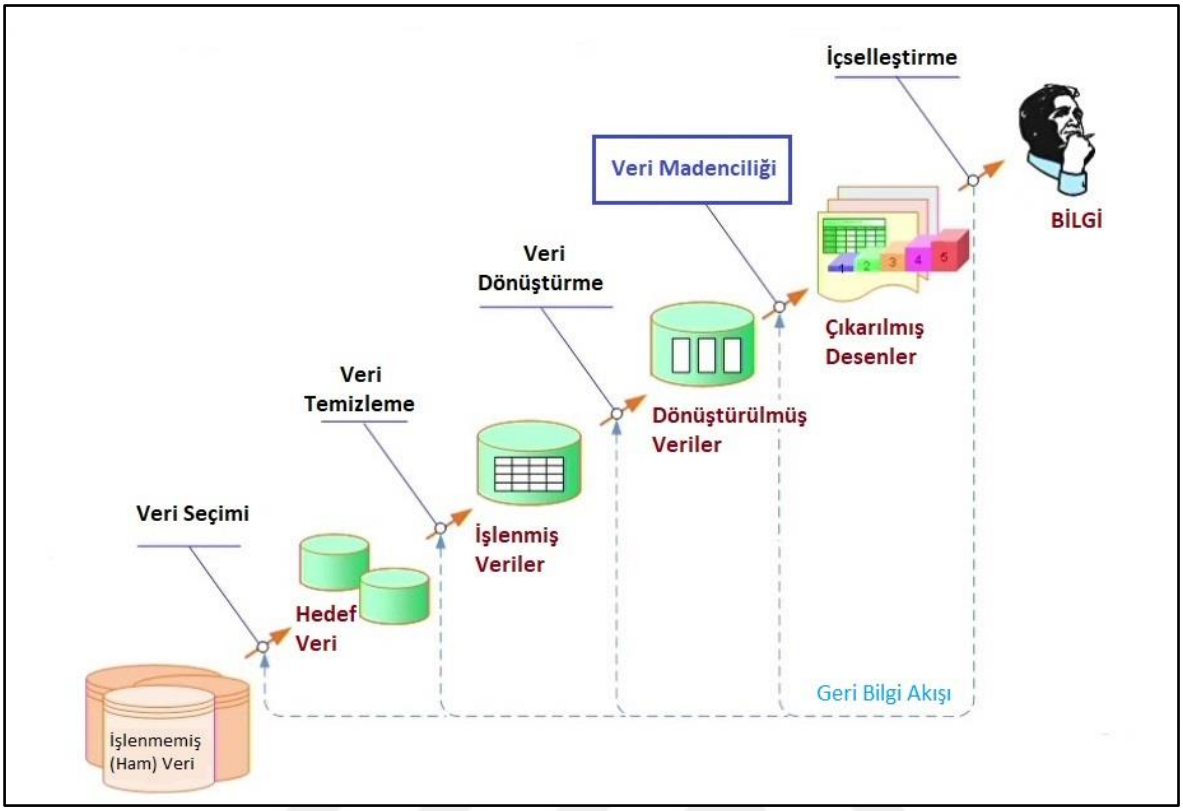
Günümüzde bilgi teknolojileri ile her geçen gün veri kaynaklarının hızla artması ile anlamsız verilerden anlamlı ve kullanışlı bilgilerin elde edilmesi önem arz etmektedir. Bu doğrultuda metin madenciliği ile düzgün veri kaynaklarını değil doğal dilde yazılmış metin verilerinden anlamlı bilgilerin çıkarılması amaçlanmaktadır.

2.1. Metin Madenciliği Tanımı

Veri kaynağının metin olduğu veri madenciliği çalışmaları metin madenciliği olarak tanımlanmaktadır. Makine öğrenmesi, doğal dil işleme, istatistik ve bilişsel bilimler gibi farklı disiplin tekniklerini kullanan metin madenciliği; yapısal olmayan verilerden daha önce bilinmeyen, ilginç ve önemli olan bilgileri keşfedip, çok sayıda dokümanı analiz edebilen bir teknolojidir [5]. Metin madenciliğinin kullandığı teknikler; bilgi alma, bilgi çıkarma ve doğal dil işleme olup bunları veri tabanlarında bilgi keşfi, istatistik, veri madenciliği ve makine öğrenimi yöntemleri ve algoritmalarıyla birleştirir [6].

Veri madenciliği Veri Tabanlarından Bilgi Keşfi (KDD- Knowledge Discovery in Databases) olarak tanımlanırken, metin madenciliği ise Metinden Bilgi Keşfi (KDT- Knowledge Discovery from Text) olarak tanımlanır.

Veri madenciliği ile ilgili KDD süreçleri Şekil 2.1’de verilmektedir:



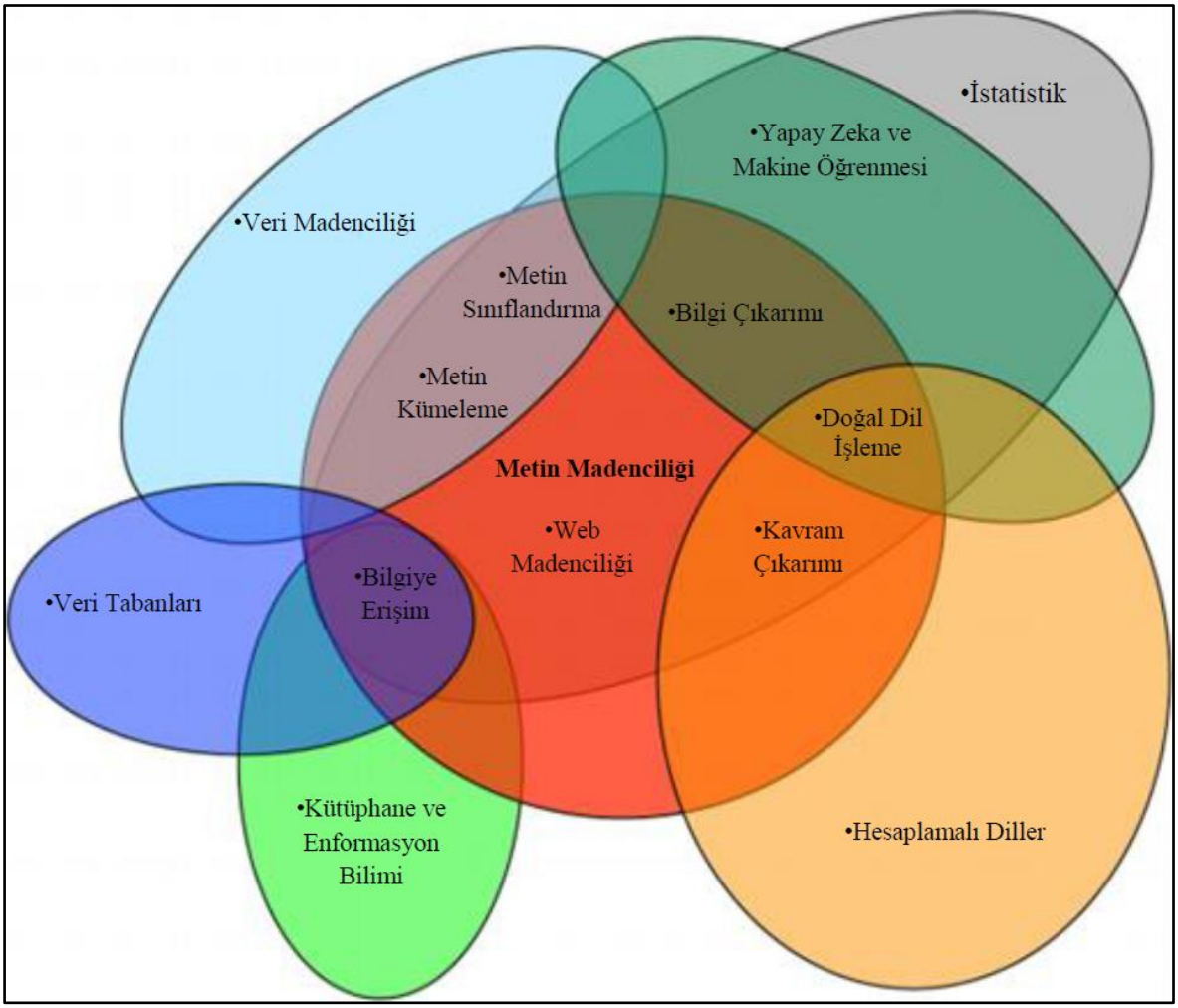
Şekil 2.1. Veri tabanlarından bilgi keşfi süreci

Şekil 2.1'de gösterilen veri madenciliği KDD süreçlerinin ilk adımı işlenmemiş (ham) veriden veri seçimi işlemidir. Sonraki adımlar sırasıyla veri temizleme ve veri dönüştürme işlemleridir. KDD sürecinin büyük bir kısmı olan veriyi hazırlama sürecinden sonra veri madenciliği süreci uygulanır. Veri madenciliği süreçleri ile anlamsız verilerden anlamlı bilgiler elde edilir. Metin madenciliği KDT süreçlerinde, KDD süreçlerinden farklı olarak veri kaynağı metinlerden oluşmaktadır.

2.2. Metin Madenciliği Metotları

Metin madenciliği; doğal dil işleme aşaması, enformasyon getirimi, adlandırılmış varlıkların tanınması, örüntüsü tanımlı olan varlıkların bulunması, eş atıf, ilişki/olay/kural çıkarımları, duygu analizi gibi problemlerle ilgilenmektedir [7].

Metin madenciliğinin ilişkili olduğu yöntemler ve disiplinler Şekil 2.2'de gösterildiği gibidir [8]:



Şekil 2.2. Metin madenciliği metotları

Metin madenciliği çalışma alanlarından olan ve Şekil 2.2'de görülen bilgiye erişimin bilgi çıkarımı ve web madenciliği metotları aşağıda açıklanmıştır.

2.2.1. Bilgiye erişim (Information retrieval)

Bilgiye erişim, cevaplarla değil, sorulara cevaplar içeren belgelerin bulunmasının yanı sıra anahtar kelimelere dayalı olarak çoğu arama motoru gibi belge erişimini gerçekleştiren sistemler olarak tanımlanır [6]. Bilgiye erişim, veri erişiminden bilgi almaya kadar tüm bilgi işleme yelpazesiyle ilgilenmek için metin verilerini otomatik olarak işlerken istatiksel yöntemleri ve ölçümleri kullanır. İlgilenilen külliyat hakkında ön bilginin toplandığı aşama olan bilgiye erişimin metin madenciliğine örnek olarak; web üzerindeki veri kaynaklarının web sayfaları, adresleri veya dosya sistemi üzerindeki dosyaların isimleri, tarihleri, dizin bilgileri ile kullanıcı bilgilerinin toplanması verilebilir [7].

2.2.2. Bilgi çıkarımı (Information extraction)

Doğal dilde yazılmış bir metin belgesinde bir bilgisayar tarafından otomatik analiz için doğrudan uygun olmayan birçok bilgi mevcut olduğundan, bilgi çıkarımı yöntemi; uygun olmayan büyük miktardaki metinleri inceleyerek sözcükler ve cümlelerden yararlı bilgiler çıkarmak için kullanılır [6]. Serbest metni etkili ve verimli bir şekilde analiz etmeye ve yapılandırılmış bilgiler biçiminde metinden ilgili ve değerli bilgileri keşfetmeye yardımcı olan bilgi çıkarımı yönteminin girdisi; elektronik posta, araştırma kâğıtları, web sayfaları, haber makaleleri, bloglar, teklifler, özgeçmişler gibi bir belge koleksiyonuyken; çıktısı ise bazı kriterlere göre kaynak belgedeki ilgili bilgilerin temsilidir [9].

2.2.3. Web madenciliği (Web mining)

Teknolojinin gelişmesi ile birçok işlem internet üzerinden yürütüldüğünden dolayı, Geniş Dünya Ağı (WWW- World Wide Web) ortamında büyük oranda çok farklı tiplerde ve standartlarda veri yığınları oluşmuştur. Web madenciliği; kullanıcıların kayıt bilgileri, web sayfaları, oturum bilgileri, hareket bilgileri, sitelerin içeriği, sitelerin yapısı gibi farklı yapıdaki web sayfalarının kayıt bilgilerini ve dokümanlarını inceler. Ayrıca veri madenciliği tekniklerini kullanıp, web sayfalarındaki kalıpları keşfetmeye çalışır [10].

Web madenciliği; kullanılacak verilerin yapısına göre 3 gruba ayrılmaktadır: web kullanım madenciliği, web yapı madenciliği, web içerik madenciliği [11].

Web içerik madenciliği

Web içerik madenciliği ile web kaynaklarının içeriklerinden anlamlı ve yararlı bilgileri elde etmek için yapay zekâ ve akıllı yazılım programları kullanılarak, web sitesinin yapısal raporu çıkarılmaya çalışılır. Web sayfası üzerindeki metin, link, görsel, resim gibi farklı yapılardaki veriler web içerik madenciliği uygulamalarını zorlaştırmaktadır. Web içerik madenciliği ile sayfaların çok derin veya geniş yapıda olup olmadığı, kullanıcıların istedikleri bilgiyi bulup bulmadıkları, sık ziyaret edilmeyen bölümlerin yerleştirildiği yer ile sayfa düzeninin ilişkisi, içerik elemanlarının doğru yere yerleştirilip yerleştirilmediği gibi sorgulamalar hakkında bilgiler elde edilebilir [10].

Web kullanım madenciliği

Web kullanım madenciliği ile web üzerindeki farklı sunucularda tutulmuş olan kullanıcıların internete erişim hareketlerinin yer aldığı günlük kayıt dosyaları veri olarak kullanılır. Kullanılan işletim sistemine göre farklılık gösteren metin tabanlı kayıt dosyalarına kaydolarak istemcilerden gelen istekler kaydedilip, bu kayıt desenlerindeki veriler ile kullanıcılar hakkında ayrıntılı bilgiler elde edilir. Ayrıca web sunucularında erişim kayıt dosyaları, tuttukları veriler gibi farklı servislerde isteğe bağlı kayıt dosyaları tutulmaktadır [10].

Web yapı madenciliği

Web yapı madenciliği, web sayfaları arasındaki linkleri Graf Teorisi ile analiz edip izleyerek bilgi üretir. Web yapı madenciliği yapısal veri tipine göre ikiye ayrılır. Bunlardan birincisi, web sayfası dokümanlarındaki XML ya da HTML etiketleri tanım ve analizinde ağaç benzeri yapıları kullanmasıdır. İkincisi ise bir web sayfasını farklı bir lokasyona yönlendiren yapısal olan bir hiperbağlantı (hyperlink) için webdeki hiperbağlantıların modelinin çıkarılmasıdır [10].

2.3. Metin Madenciliği Uygulama Alanları

Temel metin madenciliği uygulamaları en çok aşağıdaki sektörlerde kullanılmaktadır [12]:

- Medya ve yayıncılık,
- İnternet ve bilgi teknolojisi sektörü,
- Bankalar, sigorta ve finans piyasaları,
- Sağlık hizmetleri, ilaç ve araştırma şirketleri,
- Telekomünikasyon, enerji ve diğer hizmet endüstrileri,
- Siyasi kurumlar, siyasi analistler, kamu yönetimi ve yasal belgeler.

Sektörler, denenen uygulamalara göre oldukça çeşitli şekilde analiz edilir. Metin madenciliğinin kullanımında, üretim türü ve bunları metin madenciliğine yönlendiren bilgi yönetiminin amaçları ile bağlantılı olarak bazı sektörel teknik özellikleri belirlemek mümkündür. Bankacılık ve sigorta sektörlerinde Müşteri İlişkileri Yönetimi (CRM-Customer Relationship Management) uygulamaları yaygın olup, bu sektör doğal dilde soru

soran arama motorlarını destekleyen uygulamaları ve otomatik mesaj yeniden yönlendirme sistemleri ile müşteri iletişiminin yönetimini iyileştirmeyi amaçlamaktadır. Tıp ve ilaç sektörlerinde; makalelerden, patentlerden ve bilimsel özetlerden elde edilen bilgilerin analizi, sınıflandırılması ve çıkarılması için Rekabetçi İstihbarat ve Teknoloji İzleme uygulamaları yaygın olarak kullanılır. Birden fazla uygulamanın yaygın olduğu telekomünikasyon ve hizmet şirketlerinin hedefleri ise pazar analizinden insan kaynakları yönetimine, yazım düzeltmesinden müşteri görüş anketine kadar tüm uygulamalara bir cevap bulunmasıdır [13].

Genel olarak, metin madenciliği doğal dil kaynaklarının olduğu doğal dil ile yazılmış metin içeren her türlü veriler için uygulanabilir. Bu uygulamalarla ilgili örnekler aşağıda verilmiştir:

- Kullanıcıların birbiriyle iletişim kurmalarına olanak sağlayan sosyal ağ ortamlarındaki metin verilerinden insanların duygu analizi yapılabilir.
- Birden çok yazarın olduğu kitap, dergi, öykü, şiir gibi eserler yazarlara göre sınıflandırılıp, yazarların kendilerine özgü kullandıkları anahtar kelimeler çıkarılabilir.
- İnternet siteleri türlerine göre sınıflandırılıp, hangi alanla ilgili bir site olduğu bulunabilir.
- Metin madenciliği yöntemleri ile herhangi bir alanda yazılan bir kitabın özeti çıkarılabilir.
- Borsa sektöründe; piyasanın genel analizi yapılıp, hisse senedi fiyatlarının tahmini yapılabilir.

2.4. Metin Madenciliği Sürecinde Kullanılan Programlar

Veri madenciliğinin alt dalı olan metin madenciliğinde bilgiye erişmede kullanılan farklı programlar mevcuttur. Bu yazılımlardan bazıları açık kaynak kodlu, bazıları ise ticaridir. Metin madenciliği işlemlerini Bilgi Analizi için Waikato Ortamı (Weka- Waikato Environment for Knowledge Analysis) ve Rapid Miner kendi başlarına yapabiliyorken; Konstanz Bilgi Madencisi (Knime- Konstanz Information Miner) bir modül ve R ise bir paket aracılığı ile yapabilmektedir. Python ise kütüphaneleri aracılığı ile metin madenciliği ile ilgili işlemleri yapmaktadır.

2.4.1. Weka

Weka programlama dili Waikato Üniversitesi bilim adamları tarafından Java programlama dili kullanılarak geliştirilen açık kaynak kodlu bir veri madenciliği aracıdır [14]. Weka'nın kâşif, deneyici, bilgi akışı ve basit komut satırı olmak üzere dört farklı komut satırı vardır [15]. Weka'da işlenecek veri seti için dosya formatı kısıtlı olup, dosya formatı Öznitelik İlişkisi Dosyası Biçimi (ARFF- Attribute Relationship File Format) ya da Virgülle Ayrılmış Değerler (CSV- Comma-Separated Values) formatında olmalıdır. ARFF formatı ilişki, özellik ve veri yapılarına sahiptir. CSV formatı ise özelliklerin virgül ile ayrılmış halidir. Weka'da veri ön işleme, kümeleme, sınıflandırma, ilişkilendirme, görselleştirme ve özellik seçimi gibi çeşitli standart veri madenciliği işlemleri yapılabilmektedir [16]. Weka'nın desteklediği algoritma sayısı 76 olup, yazılımda mevcut olmayan algoritmalar Weka'ya daha sonradan yüklenebilir. Ayrıca Java ile geliştirme için yazım kuralları ve özel dokümantasyonlar yazılım ekibi tarafından sunulur [17].

2.4.2. RapidMiner

Rapid Miner programlama dili Dortmund Teknoloji Üniversitesi'nin Yapay Zekâ biriminde, Java programlama dili kullanılarak geliştirilmiş bir veri madenciliği yazılımıdır [17]. Rapid Miner ticari olarak kullanılan lisanslı bir yazılımdır. Hizmet Olarak Yazılım (SaaS- Software as a Service) veya bulut alt yapılarında sunulan sunucuyla birlikte bir istemci/sunucu modeli olarak kullanılan Rapid Miner; metin madenciliği, makine öğrenimi, iş analitiği ve tahmine dayalı analiz için entegre bir ortam sağlar [18]. Rapid Miner'ın desteklediği dosya format sayısı 22 ve algoritma sayısı 129 olup, Weka'dan daha yüksek bir değere sahiptir.

2.4.3. Knime

Knime, Konstanz Üniversitesi'nin görsel veri madenciliği araştırma grubu tarafından Java ile yazılmış, kurulabilen ve yazılımı kurmadan disk üzerinde çalışabilen açık kaynak kodlu bir yazılımdır [17]. Knime'ın desteklediği algoritma sayısı 102 olup; Weka'dan daha yüksek, Rapid Miner'dan daha düşük bir değere sahiptir. Knime modüler uygulaması olan geliştirme kiti ile uygulama kullanıcılarının kendi modüllerini yazıp, geliştirmesine olanak sağlamanın yanı sıra zengin görselleştirme araçlarına da sahip olan bir yazılımdır. Metin

madenciliği işlemlerini bir modül aracılığıyla yapabilmektedir. Desteklediği dosya formatları ise ARFF ve CSV'dir.

2.4.4. R

R&R olarak da bilinen R programlama dili Auckland Üniversitesi'nin İstatistik bölümü tarafından S programlama dili ile geliştirilen açık kaynak kodlu bir programdır [19]. Veri madenciliği, makine öğrenimi ve istatistiksel teknikler sağlayan R programlama dilinin ücretsiz olarak temin edilebilen ekstra paketleri bulunmaktadır [18]. Metin madenciliği işlemlerini de bir paket aracılığıyla yapabilmektedir. R, istatistiksel hesaplama işlemlerini diğer programlara göre daha güçlü bir şekilde yapan yazılım olmakla birlikte görselleştirme açısından da zengin bir yazılımdır. R programlama dilinin güçlü yanlarından biri, gerektiğinde formüller ve matematiksel semboller dâhil olmak üzere iyi tasarlanmış yayın kalitesinde grafikleri kolay bir şekilde üretebilmesidir [18]. Büyük veri uygulamalarının son zamanlarda yaygınlaşması ile birlikte büyük veri ve yapay öğrenme alanlarında Python ile birlikte kullanımı en çok artan dillerden birisi olmuştur [20].

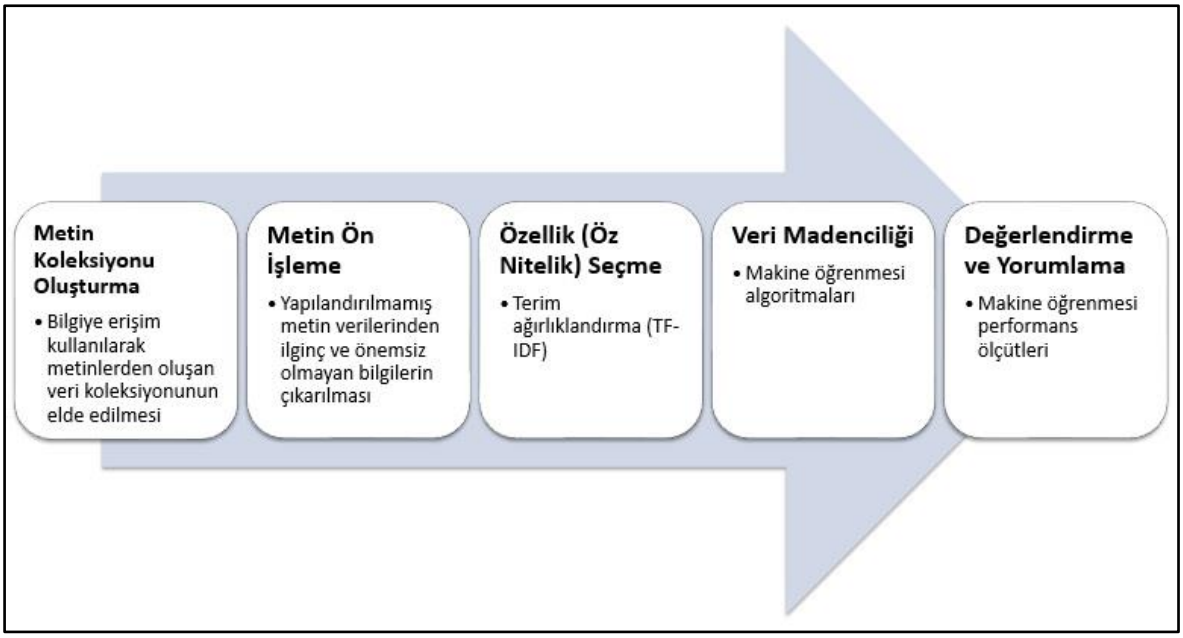
2.4.5. Python

C programlama dilinde yazılan, açık kaynak kodlu ve ücretsiz olan Python Programlama dili, 1989 yılında Guido van Rossum tarafından geliştirilmiştir [20]. Python nesne yönelimli, yorumlamalı, etkileşimli ve modüler olan yüksek seviyeli bir programlama dilidir [21]. Öğrenmesi, yazması ve okuması hem kolay hem de güçlü olan Python'un standart kütüphaneleri çok fazla ve gelişmiştir. Python kütüphaneleri sayesinde çok fazla kod yazmaya gerek duyulmadan, istenilen işlemler kolay bir şekilde yapılabilir. Metin madenciliğinin her aşaması için zengin kütüphaneleri bulunan Python, diğer programlama dillerine göre daha kullanışlı bir yazılımdır. Diğer programlarda elimizde bulunan veri setini programın istediği formatta yüklememiz gerekmektedir. Python'da veri seti elde etme süreci kütüphaneler yardımı ile yapılabiliyorken; veri seti de Python programı ile istenilen formatta elde edilip işlenebilmektedir. Python programlama dili, veri elde etme süreci gibi, diğer metin madenciliği süreçlerinin de daha kolay ve esnek bir şekilde yapılabilmesine olanak sağlamaktadır.

3. METİN MADENCİLİĞİ SÜRECİ

Metin madenciliğinin ilk aşamasında veri seti olarak metinler elde edilir. Elde edilen metinlerden metin koleksiyonu oluşturulduktan sonra metin ön işleme yapılır. Sonrasında öz nitelik seçimi, terim ağırlıklandırma ve veri madenciliği aşamaları uygulanır. Son olarak ise veri madenciliği ile elde edilen örüntü sonuçları, değerlendirilip yorumlanarak bilgi elde edilir.

Metin madenciliği süreçleri Şekil 3.1'de gösterilmektedir:



Şekil 3.1. Metin madenciliği süreçleri

Şekil 3.1'de gösterilen metin madenciliği ile ilgili süreçlerden metin koleksiyonu oluşturma, metin ön işleme, özellik seçme, veri madenciliği ile değerlendirme ve yorumlama aşamaları maddeler halinde aşağıda anlatılmaktadır.

3.1. Metin Koleksiyonu Oluşturma

Metin madenciliği metotlarından bilgiye erişim kullanılarak metinlerden oluşan veri koleksiyonu elde edilir. Oluşturulan metin koleksiyonu; internet veya bilgisayar ortamındaki metinlerden, çevrim içi ya da çevrim dışı veri tabanlarından elde edilebilmektedir. Elde edilen metin koleksiyonu genellikle doğal dil ile yazılan ve yapısal olmayan verilerden oluştuğu için, verilere metin ön işleme aşamasının uygulanması gerekebilmektedir.

3.2. Metin Ön İşleme

Metin madenciliği süreçlerinden olan metin ön işleme, yapılandırılmamış metin verilerinden ilginç ve önemsiz olmayan bilgileri çıkarmak için kullanılan önemli bir adımdır [22].

Metin ön işleme süreçleri Şekil 3.2'de gösterilmektedir:

Şekil 3.2. Metin ön işleme süreçleri

Şekil 3.2'de gösterilen metin ön işleme süreçleri için ilk önce, verilen bir metindeki tüm kelimeleri elde etmek için bir metin belgesindeki tüm noktalama işaretleri ve metin olmayan karakterlerin kaldırılıp tek beyaz boşlukla değiştirilerek bir kelime akışına bölündüğü belirtkeleme (tokenization) işlemi gereklidir [6]. Bu süreçte metin koleksiyonundan noktalama işaretleri, anlamsız (etkisiz) kelimeler ve sayılar metin koleksiyonundan çıkarılarak daha anlamlı bir veri elde edilmeye çalışılır. Gerekli olması durumunda; metin verisi büyük ya da küçük harfe de çevrilebilir.

3.3. Terim Ağırlıklandırma ile Özellik (Öz Nitelik) Seçme

Veri ön işlemenin önemli tekniklerinden biri olup sıklıkla kullanılan özellik seçimi; makine öğrenme sürecinin vazgeçilmez bir bileşenidir. İlgili özellikler tespit edilerek; seçili özelliklerden daha fazla bilgi sağlamayan gereksiz özellikleri, hiçbir bilgi sağlamayan alakasız özellikleri ve gürültülü verileri kaldırma işlemi yapılır. Bu süreç anlaşılırlığın artmasını sağlayarak tahmin doğruluğunu geliştirir ve veri madenciliği algoritmalarını hızlandırır. Özellik seçiminin Bireysel Değerlendirme ve Alt Küme Değerlendirmesi olarak iki ana yaklaşımı bulunmaktadır. İlk yaklaşım olan Bireysel Değerlendirmede, ilgililik derecesine göre bireysel bir özelliğin ağırlığı atanırken; ikinci yaklaşım olan Alt Küme Değerlendirmesinde arama stratejisi kullanılarak aday özellik alt kümeleri oluşturulur [23].

Metin madenciliğinin sınıflandırılmasında sonuçlar üzerinde direkt olarak etkili olan terim ağırlıklandırma adımında, dokümandaki bir terimin önemini belirtmek amaçlanmaktadır [24]. Terim ağırlıklandırma ile ilgili terimler aşağıda verilmiştir:

3.3.1. Terim frekansı (TF-Term frequency)

Bir metindeki seçili bir kelimenin, metin içinde bulunan toplam kelime sayısına bölümü ile elde edilen terim frekansı, Luhn tarafından ortaya çıkarılmıştır. Luhn teorisi; belge içeriğinin belirlenmesi ve ilgili bilgilerin alınması için, metinden elde edilen yüksek frekanslı terimlerin kullanımının gerekli olduğunu söyler. Fakat frekans ağırlıklandırma terimi her zaman beklendiği gibi çalışmamaktadır. Özellikle, belirli bir metin koleksiyonunda az sayıda yüksek frekanslı indeks terimi mevcut olduğunda veya yüksek frekanslı terimler belgeler arasında eşit olarak dağıtıldığında (örneğin, belirli bir terim her belgede k kez geçtiğinde); yüksek frekanslı terimlerin ağırlığının artırılması hiçbir işe yaramayacaktır [25].

3.3.2. Ters doküman frekansı (IDF-Inverse document frequency)

Terim sıklığına alternatif olarak Sparck Jones tarafından doküman frekansı teorisi ortaya atılmıştır. Doküman frekansı; toplam doküman sayısının, belirli bir kelimenin metin koleksiyonundaki geçtiği doküman sayısına bölünmesi ile elde edilir. Özellikle düşük belge sıklığına sahip terimlerin, yüksek belge sıklığına sahip terimlere göre erişim amaçları için daha önemli olduğu önerilmektedir. Çünkü karşılaştırmalı nadirlikleri, sorgu-belge

eşleştirmedeki önemlerini arttıracaktır. Ters doküman frekansı ise doküman frekansının logaritması alınarak hesaplanır [25].

3.3.3. Terim frekansı- Ters doküman frekansı (TF-IDF)

TF-IDF algoritması, bir dosya kümesindeki veya derlemdeki bir belge veya kategori için bir kelimenin önemini değerlendirmek için kullanılan istatistiksel bir yöntemdir. Bu yöntem ile bir kelime ya da kelime öbeği bir makalede sıkça geçiyorsa ve diğer makalelerde nadiren bulunuyorsa; ilgili kelime veya kelime öbeğinin iyi bir sınıf ayrımı kabiliyetine sahip olduğu ve sınıflandırmaya uygun olduğu düşünülmektedir [26]. TF-IDF algoritması ilk olarak Salton tarafından önerilmiş olup mevcut olan vektör uzayı modelinde kelime ağırlığı hesaplama fonksiyonu en sık kullanılan öz niteliktir [25]. Temel olarak TF ve IDF olmak üzere iki bölümden oluşmaktadır. TF-IDF değeri; seçili kelimenin metin içinde bulunan toplam kelime sayısına bölümü olan TF değeri ile toplam metin sayısının kelimeyi içeren metin adedine bölümünün logaritması olan IDF değerinin çarpımı sonucunda elde edilir. Metin koleksiyonundaki önem derecesi yüksek olan kelimelerin TF-IDF değeri, diğer kelimelere göre daha yüksek çıkmaktadır.

3.4. Makine Öğrenmesi ile Veri Madenciliği

Büyük veri setleri içerisinde birden çok adım ile mantığa uygun ve yeni bilgiler ortaya çıkarmak için sentezleme işlemlerinin yapıldığı veri madenciliğinde; yapay zekâ, makine öğrenmesi, örüntü tanıma ve istatistik gibi farklı yöntemler kullanılmaktadır. Kullanılan yöntemler problemin türüne, uygulama alanına ve elde bulunan veriye göre farklılıklar göstermekte olup birçok yöntem yapay zekânın alanıdır [27]. Bu aşamada eldeki veriye akıllı algoritmalar uygulanarak örüntülerin keşfedilmesi çalışılmaktadır.

Metin madenciliği süreçlerinden olan veri madenciliği aşamasında kullanılan makine öğrenimi bir yapay zekâ metodolojisi olduğundan dolayı yapay zekânın bir alt kümesidir. Makine öğrenimi algoritmaları ile metin koleksiyonu verisindeki örüntüler keşfedilerek, bu örüntüler değerlendirilip anlamlı bilgi ya da bilgiler elde edilmeye çalışılır.

3.4.1. Makine Öğrenmesi Tanımı

Geniş kapsamlı uygulamalarla bilgisayar biliminde hızla büyüyen alanlarından biri olan makine öğrenmesi; verilerdeki anlamlı kalıpların otomatik olarak algılanmasını ifade etmektedir. Makine öğrenmesi için çeşitli uygulamalar olmakla birlikte bunlar arasında en önemli olan veri madenciliğidir. İnsanlar genellikle analizler sırasında veya muhtemelen birden fazla özellik arasında ilişki kurmaya çalışırken hata yapmaya eğilimlidir. Veri madenciliği ve makine öğrenimi, uygun öğrenme algoritmaları aracılığıyla çeşitli iç görülerin elde edilebildiği siyam ikizleri olarak tanımlanır [28].

3.4.2. Makine Öğrenmesi Algoritmaları

Veri üretiminin çok fazla olması, makine öğrenimi tekniklerini zaman zaman karmaşık hale getirmiştir. Makine öğrenimi algoritmaları, algoritmadan istenilen sonucuna göre bir taksonomi düzenlemektedir [28].

Makine öğrenimi algoritmalarının performansı ve hesaplamalı analizi, hesaplamalı öğrenme teorisi olarak bilinen bir istatistik dalıdır. Makine öğrenimi, bir bilgisayarın öğrenmesini sağlayan algoritmalar tasarlamakla ilgilidir. Öğrenme, bilinçli olmayı gerektirmemekle birlikte verilerdeki istatistiksel düzenlilikleri veya diğer kalıpları bulmaya çalışır. Bu nedenle birçok makine öğrenimi algoritması, insanın bir öğrenme görevi yaklaşımına hiç benzemeyecektir. Makine öğrenmesinin denetimli öğrenme, yarı denetimli öğrenme, takviyeli öğrenme, denetimsiz öğrenme, iletim, öğrenmeyi öğrenme gibi algoritma türleri mevcuttur [29].

Denetimsiz öğrenme (Unsupervised learning)

Denetimsiz öğrenme, gruplama kararlarını eğitim örnekleri kullanmadan doğrudan verilen örnek kümenin özelliklerine dayandırmaya çalışır. Denetimsiz sınıflandırma algoritmalarının; amaç fonksiyonunun genel minimizasyonuna dayalı kümeleme, ikili mesafe kümeleme ve istatistiksel modelleme olmak üzere üç ana kategorisi bulunmaktadır. Bu tür sınıflandırma, denetimsiz öğrenmenin kapsamını kümeleme algoritmalarıyla sınırlamıştır [30].

Kümeleme (Clustering)

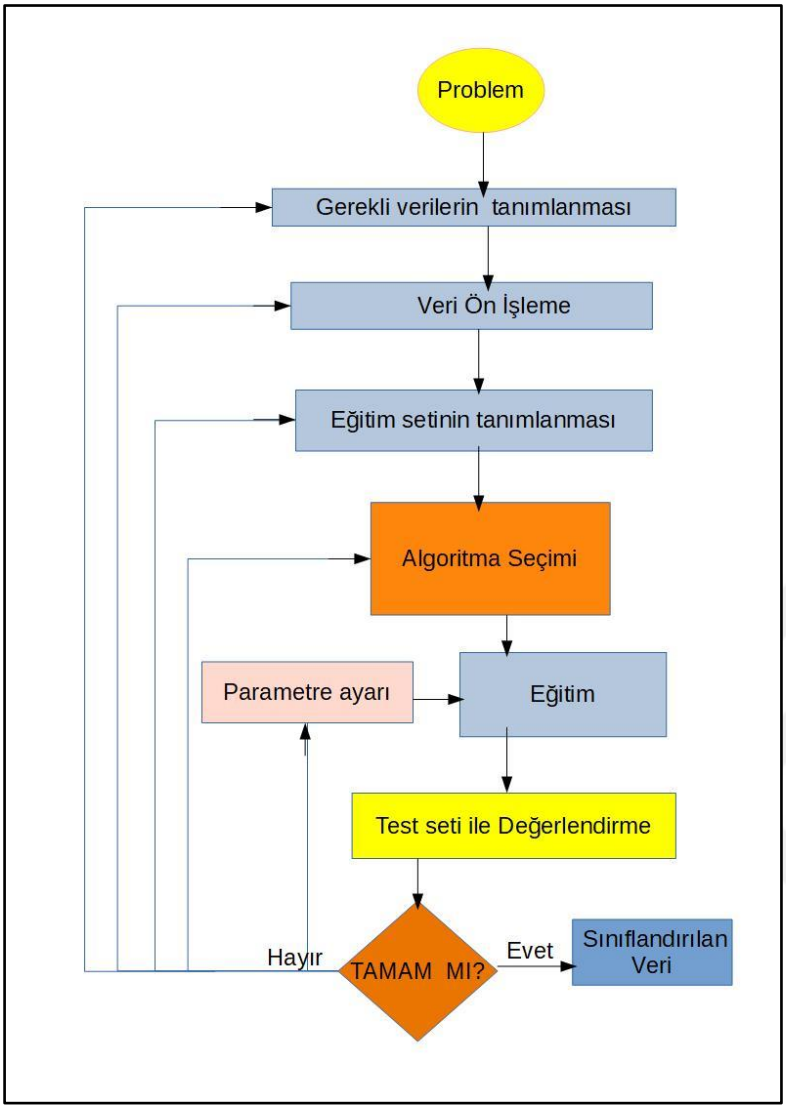
Kümeleme algoritmaları; bir kümenin örneklerini, aynı kümedeki iki örneği farklı kümelerdeki iki örnekten daha fazla birbirine benzeyecek şekilde gruplandırır. Kümeleme yöntemleri parametrik ve parametrik olmayan yöntemler olarak iki geniş sınıfa ayrılır [30].

- Parametrik olmayan kümeleme; veri kümesi örnekleri arasındaki farkın derecesinin (Öklid mesafesi gibi) bir değerlendirmesini kullanarak bir veri kümesinde doğal gruplamalar (kümeler) bulmayı içerir. Örnekler arasında bir benzerlik (farklılık) ölçüsü tanımlamayı, kümeleme için bir ölçüt işlevi tanımlamayı ve ölçüt işlevini en aza indirecek (veya en üst düzeye çıkaracak) bir algoritma tanımlamayı gerektirir. Popüler parametrik olmayan kümeleme algoritmaları arasında k-ortalamalar kümesi, hiyerarşik kümeleme algoritmaları ve spektral kümeleme bulunur.
- Parametrik kümeleme; veri kümesi örneklerinin birkaç bileşenden oluşan bir olasılık fonksiyonu ile temsil edilebileceğini varsayar. Parametrik kümeleme yöntemlerinde; bir kümedeki her örnek, aynı kümede olduğu varsayıldığı gibi, sonlu sayıda işlevin ve benzer kombinasyonlara sahip örneklerin bir kombinasyonu olarak tanımlanır. Gauss Karışım Modeli (GMM- Gaussian mixture model) ve konu tabanlı model gibi olasılıksal parametrik kümeleme yönteminin kullanılmasının; sırasıyla sürekli ve ayrık verilerin analizi ile ilgili çok çeşitli uygulamalarda başarılı olduğu gösterilmiştir.

Denetimli öğrenme (Supervised learning)

Denetimli makine öğrenimi, genel hipotezler üretmek için harici olarak sağlanan örneklerden akıl yürüten ve sonrasında gelecekteki örnekler hakkında bazı tahminlerde bulunan algoritmaların aranmasıdır. Denetimli öğrenmenin amacı, tahmin edici özellikler açısından sınıf etiketlerinin dağılımının kısa bir modelini oluşturmaktır. Ortaya çıkan sınıflandırıcı; tahmin edici özelliklerin değerlerinin bilindiği, fakat sınıf etiketinin değerinin bilinmediği sınıflandırmadaki test örneklerine sınıf etiketleri atamak için kullanılır [31].

Denetimli öğrenmenin uygulama süreci Şekil 3.3'te gösterilmektedir:



Şekil 3.3. Denetimli makine öğrenimi süreci

Şekil 3.3'te görüldüğü gibi denetimli makine öğrenimi sürecinde ilk adım veri setini toplamaktır. Konu ile ilgili bir uzman mevcutsa, uzman hangi alanların (nitelikler, özellikler) daha bilgilendirici olduğunu önerebilir. Eğer bir uzman mevcut değilse; doğru (bilgilendirici, ilgili) özelliklerin izole edilebileceği, mevcut her şeyi ölçen kaba-kuvvet (brute-force) yöntemi kullanılır [31].

İkinci adım, veri hazırlama ve veri ön işleme aşamasıdır. Koşullara bağlı olarak, araştırmacıların eksik verileri işlemek için seçebilecekleri bazı yöntemler vardır. Hodge ve Austin yakın zamanda aykırı değer (gürültü) tespiti için çağdaş teknikler üzerine bir araştırma başlatmışlardır [32]. Bu araştırmacılar, tekniklerin avantajlarını ve dezavantajlarını belirlediler. Bir örneğin seçimi hem gürültüyü ele almak için hem de çok büyük olan veri kümelerinden öğrenmenin olanaksızlığıyla başa çıkmak için kullanılır. Bu

veri kümelerinde örnek seçimi, örnek boyutunu en aza indirirken madencilik kalitesini korumaya çalışan bir optimizasyon problemidir [33]. Verileri azaltır ve bir veri madenciliği algoritmasının çok büyük veri kümeleriyle etkin bir şekilde çalışmasını sağlar. Özellik alt kümesi seçimi, mümkün olduğunca çok sayıda alakasız ve fazla özelliğin belirlenip kaldırılması işlemidir. Bu, verilerin boyutsallığını azaltır ve veri madenciliği algoritmalarının hem daha etkili hem de daha hızlı çalışmasını sağlar. Birden fazla özelliğin birbiriyle bağlı olması, denetimli makine öğrenmesinin sınıflandırma model doğruluğunu genellikle gereksiz yere etkiler [31]. Bu sorun, temel özellik kümesinden yeni özellikler oluşturularak çözülebilir. Bu tekniğe özellik oluşturma/dönüştürme denir [34]. Bu yeni oluşturulan özellikler, daha özlü ve doğru sınıflandırıcıların oluşturulmasına olanak sağlayabilir. Ayrıca anlamlı öz niteliklerin keşfi, üretilen sınıflandırıcının ve öğrenilen kavramın daha iyi anlaşılmasına katkıda bulunur [31].

Sınıflandırma (Classification)

Denetimli makine öğrenimi, gelecekteki örneklerin kaderini tahmin etmek için harici olarak sağlanan örnekleri kullanarak genel modeller ve hipotezler üretebilen algoritmaların oluşturulmasıdır. Denetimli makine öğrenimi sınıflandırma algoritmaları, önceki bilgilerden elde edilen verileri sınıflandırmayı amaçlar. Veri bilimi problemlerinde sınıflandırma çok sık yapılmaktadır. Bu tür sorunları çözmek için kural tabanlı, mantık tabanlı, örnek tabanlı ve stokastik gibi çeşitli başarılı teknikler önerilmiştir [35].

Lojistik Regresyon Algoritması (Logistic Regression Algorithm)

Lojistik Regresyon algoritması bilgisayarla görme, sosyal bilimler, pazarlama gibi birkaç disiplinde genellikle standart bir olasılıksal istatistiksel sınıflandırma modeli olarak kullanılır. Doğrusal regresyondan farklı yönü, bir örnek üzerindeki doğrusal regresyonun sonucu, olasılığın (örneğin doğrusal ölçümüne bağlı olduğu durumlarda) pozitif ya da negatif olma ihtimalidir [36].

Lojistik regresyon, kategorik bir sonuç değişkeni ile bir ya da daha fazla sürekli veya kategorik tahmin değişkeni arasındaki ilişkilerin hipotezlerini tanımlayarak, test etmek için uygun bir algoritmadır. Lojistik regresyonun altında yatan temel matematiksel kavram, logit-bir olasılık oranının doğal logaritmasıdır [37].

Naive Bayes Algoritması (Naïve Bayes Algorithm)

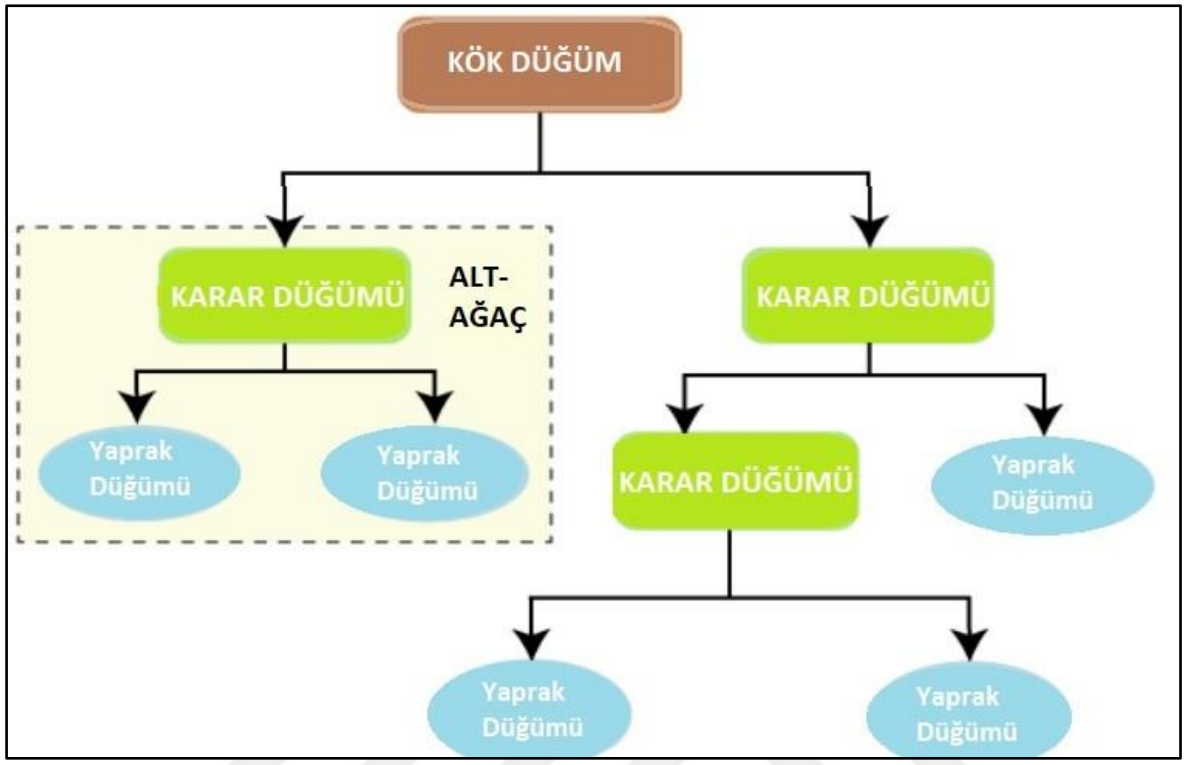
Bağımsız bir öz nitelik modeli olarak tanımlanabilecek Naive Bayes algoritması, Bayes teoreminin güçlü bağımsızlık varsayımları ile uygulanmasına dayanan basit bir olasılıksal sınıflandırıcı ile uğraşmaktadır. Naive Bayes için farklı varsayım uyumu sağlayan birkaç model vardır. En yaygın kullanılan iki modelden birisi, ikili kelime oluşumlarını kullanan ve boolean ağırlığı olarak karakterize edilen Çok Değişkenli Bernoulli Olay Modeli (Multivariate Bernoulli Event Model); diğeri ise kelime oluşum frekanslarını kullanan Çok Terimli Model'dir (Improved Multinomial Event Model). Çok Terimli Model'de bir belgedeki kelime sayımlarının oluşu kaydedilir. Bazı belirli belgeler için, bir türden sözcükler daha yüksek sözcük sayılarına neden olabilir. Bu nedenle, bir belgede geçen kelime sayısını toplamak, bir belgede meydana gelen özellik sayılarını ihmal eden Çok Değişkenli Bernoulli Olay Modelinden daha etkilidir [38].

Çok Terimli Naive Bayes Modeli; bir belgedeki bir sözcüğün her geçişini, aynı sözcüğün diğer oluşumlarından bağımsız olarak ele alır. Ancak, bir belgede aynı kelimenin birden çok tekrarı bağımsız değildir. Bir kelime bir kez geçtiğinde, tekrar ortaya çıkması muhtemeldir [39]. Çok Terimli Model bu fenomeni hesaba katmaz. Birden çok olay, çok fazla yararsız veri getiren yüzdelerden oluşur. Bunun sonucunda, dokümanlardaki birden çok geçen aynı kelimenin olasılığının büyük ölçüde küçümsenmesine neden olur [40].

Karar Ağacı Algoritması (Decision Tree Algorithm)

Karar ağacı; temel olarak model sınıflandırma, tahmin için kullanılan tüme varım araştırması ve veri madenciliği için önemli bir yöntemdir. Karar ağacı yönteminde, genellikle oluşturulan bir karar ağacının her bir düğümü için uygun özelliği belirleyerek bilgi kazanımı yaklaşımı kullanılır. Böylece, mevcut düğümün test öz niteliği olarak en yüksek bilgi kazancına (maksimum düzeyde entropi azalması) sahip öz nitelik seçilebilmektedir. Böylece, eğitim örnek alt kümesini sınıflandırmak için gereken bilgiler bölümlenerek en küçük bilgi haline gelecektir. Başka bir deyişle, bu özelliğin mevcut düğümde bulunan numune setini bölmek için kullanılması, oluşturulan tüm numune alt kümeleri için farklı tiplerin karışım derecesini minimuma indirecektir. Böyle bir bilgi teorisi yaklaşımının kullanılması, nesne sınıflandırmasının gerekli bölme sayısını etkili bir şekilde azaltacaktır [41].

Karar ağacının yapısı Şekil 3.4'te gösterilmektedir.



Şekil 3.4. Karar ağacı yapısı

Karar Ağacı Sınıflandırma algoritmaları, karar ağacı oluşturmak için kullanılan ve Şekil 3.4'te gösterilen birkaç türden oluşmaktadır. Bu; kayıp değerlerin hem sürekli hem de periyodik öz niteliklerinin kontrolü ile olur. Karar ağacı, tipik olarak istatistiksel bir sınıflandırıcı olarak temsil edilen ve kümeleme için kullanılabilen bir form tarafından oluşturulur. Karar ağacına düğümler (nodes) ve dallar (branches) dahil edilir. Her düğüm, bir veya daha fazla özelliğe dayanan; yani, bir öz nitelik değerini bir sabitle karşılaştırmak veya birden fazla özelliği karşılaştırmak için diğer işlevleri kullanmak gibi problemler gerektirir. Karar ağacının amacı için öğrenme verilerinin toplanması bazen sonuç ağacı olarak anılır [42].

K-En Yakın Komşu Algoritması (K-Nearest Neighbours Algorithm)

En yakın komşular yöntemi (NN- Nearest neighbors), veri keşfi ve tahmin rehberliği için popüler bir algoritmadır. NN yöntemine dayalı modeller, benzer koşullara sahip geçmiş günleri aramak için girdi olarak kar-meteorolojik değişkenleri (snow-meteorological variables) kullanır ve ardından tahmin oluşturmak için geçmiş benzer günlerle ilişkili

olayları analiz eder. Yüksek tahmin doğruluğu elde etmek için, çeşitli girdi değişkenlerine farklı ağırlıklar atanarak bir NN modelinin kalibre edilmesi gerekir. Bu nedenle model kalibrasyonu, tahmin doğruluğunu en üst düzeye çıkarmak amacıyla bir optimizasyon problemi olarak ele alınabilir [43].

K-En Yakın Komşu algoritması (K-NN), eğitim kümesindeki veriyi daha önceki verilerle ilgili en yakın kümeye göre sınıflandırmaktadır. Bu yaklaşımın üç temel unsuru vardır: bir dizi etiketli nesne, nesneler arasındaki mesafeyi hesaplamak için bir mesafe veya benzerlik metriği ve en yakın komşu sayısı olan k değeri. İlk olarak etiketlenmemiş bir nesneyi sınıflandırmak için, bu nesnenin etiketlenmiş nesnelere olan uzaklığı hesaplanır. Sonrasında K-En Yakın Komşu algoritması tanımlanır. Daha sonrasında, NN'lerin sınıf etiketleri nesnenin sınıf etiketini belirlemek için kullanılır [44].

Destek Vektör Makinesi Algoritması (SVM- Support Vector Machine Algorithm)

Destek Vektör Makinesi (SVM), örnek olarak nesnelere etiket atamayı öğrenen bir bilgisayar algoritmasıdır. SVM belirli bir veri koleksiyonuna göre belirli bir matematiksel işlevi maksimize etmek için kullanılan matematiksel bir varlıktır. Bununla birlikte; bir denklemi okumadan, SVM algoritmasının arkasındaki temel fikirler açıklanabilir. Örneğin; bir SVM, milyonlarca hileli ve hileli olmayan kredi kartı faaliyet raporunu inceleyerek hileli kredi kartı faaliyetini tanımlamayı öğrenebilir. Alternatif olarak; bir SVM, elle yazılmış sıfırlar, birler ve benzerlerinden oluşan geniş kapsamlı taranmış görüntü koleksiyonunu inceleyerek el yazısı rakamlarını tanımayı öğrenebilir. SVM; aynı zamanda, giderek genişleyen çeşitli biyolojik uygulamalara da başarıyla uygulanmaktadır [45].

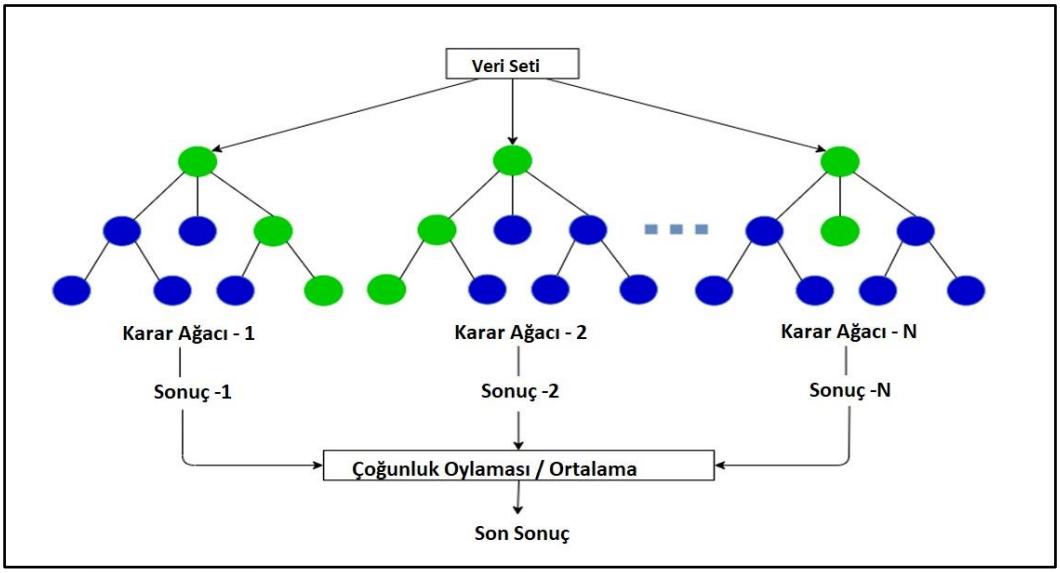
SVM algoritması dört temel kavramdan oluşmaktadır: ayırıcı hiperdüzlem (the separating hyperplane), maksimum marj hiperdüzlem (the maximum-margin hyperplane), yumuşak kenar boşluğu (the soft margin) ve çekirdek işlevi (the kernel function) [45]. Ayırıcı hiperdüzlem teoremi; optimizasyon, yöneylem araştırması, işletme ve ekonomi alanlarında sayısız uygulama ile dışbükeylik ve dışbükey programlama teorisinde en iyi bilinen teoremlerden biridir. Teorem, klasik ayırma teoremlerinden farklıdır. Sonlu bir nokta kümesinin dışbükey gövdesindeki ayırt edici bir noktanın üyeliğini test etmek için mesafe ikiliğini genelleştirir [46]. Dışbükey kümeleri ve dışbükey fonksiyonları inceleyen bir matematiksel analizin alt alanı olan konveks analizde ikilik teorisi; iki dışbükey kümenin bir

hiperdüzlemle ayrılabileceği koşulları veren bir teoremler koleksiyonuna dayanır [47]. Bu teoremler ayırıcı teoremler olarak bilinir. Ayırıcı hiperdüzlem teoremi birçok yönden bir SVM'in başarısının anahtarıdır. Yumuşak marj parametresi, hiperdüzlem ihlalleri ile marjın boyutu arasında bir dengeyi belirtir [45]. SVM verilerdeki hatalarla başa çıkabilmek için, yumuşak kenar boşluğunu; birkaç anormal ifade profilinin ayırma hiperdüzleminin yanlış tarafına düşmesine izin verip eklemektedir. Çekirdek işlevi, SVM'in bir dizi orijinal tek boyutlu verinin "iki boyutlu" bir sınıflandırmasını gerçekleştirmesine izin veren matematiksel bir numaradır. Genel olarak, bir çekirdek işlevi, verileri düşük boyutlu bir uzaydan daha yüksek boyutlu bir uzaya yansıtır [48].

Rastgele Orman Algoritması (Random Forest Algorithm)

Rastgele Orman algoritması, basitliği ve etkinliği ile bilinen makine öğreniminin temsili algoritmalarından biri olarak kabul edilebilir. Nihai sınıflandırıcının Rastgele Orman Sınıflandırması olarak en iyi sınıflandırma ağacını seçen, oylama yoluyla en çok kabul gören grup algoritması olduğundan dolayı, karar ağacı tabanlı sınıflandırıcı olarak da tanımlanabilir. Rastgele Orman, aynı şekilde dağıtılmış bağımsız rastgele vektörlere sahip bir dizi ağaç yapılı sınıflandırıcıdan oluşan bir sınıflandırıcıdır ve her bir ağaç en popüler sınıf için x girişinde bir birim oy verir. Aynı dağılımın önceki rastgele vektörlerinden, bağımsız bir rastgele vektör oluşturulur. Rastgele ormanlar için genelleme hatasını iki parametre (bireysel sınıflandırıcıların karşılıklı bağımlılığı ve kesinliği) açısından almak için bir üst sınır çıkarılarak oluşturulan bir eğitim testi ile bir ağaç oluşturulur. Binlerce girdi değişkenini hızlı ve etkili bir şekilde yönetebildiği için büyük veri tabanlarında çalışırken etkili olup doğruluk oranı yüksektir [49].

Rastgele Orman algoritmasının akış şeması Şekil 3.5'te gösterilmektedir [49]:



Şekil 3.5. Rastgele Orman algoritması akış şeması

Rastgele Orman algoritmasının Şekil 3.5'te gösterilen akış şemasına göre, veri seti belirlenen n farklı karar ağacına ayrılmaktadır. Her karar ağacından çıkan sonuçlar hesaplanarak, tahminlerin ortalaması ile nihai sonuç elde edilir.

Stokastik Gradyan İniş Algoritması (SGD- Stochastic Gradient Descent Algorithm)

Stokastik Gradyan İniş algoritması (SGD), modern makine öğreniminin beygir gücünden biri haline gelmiştir. Özellikle, sinir ağları gibi oldukça karmaşık ve dışbükey olmayan modellerin eğitimi için tercih edilen bir optimizasyon yöntemidir. Bu modeller; klasik makine öğrenmesi teorisinin önerdiğinden daha iyi genelleştiği (aşırı uyumdan daha az muzdarip olduğu) gözlemlendiğinde, bu fenomeni açıklamak için büyük bir teorik ilgi ortaya çıkmıştır. SGD'nin en iyi ihtimalle dışbükey olmayan amaç fonksiyonunun yerel bir minimumunu bulduğu göz önüne alındığında, tüm bu minimumların eşit derecede iyi olabileceği ileri sürülmüştür [50].

Stokastik yaklaşım; değerlendirmeleri gürültü tarafından bozulan bir fonksiyonun yerel minimumunu bulmak için büyük, pratik öneme sahip bir yaklaşım sağlamaktadır. Optimizasyon ve çok sayıda uygulama ile kontrol konusunda uzun bir geçmişi vardır. SGD; kontrol, sistem tanımlama ve filtreleme teorilerinde aktif makine öğrenimi topluluğundan kaynaklanan, zorlu olan yeni uygulamalar bulmuştur [51].

Çekirdek Yaklaşım Algoritması (Kernel Approximation Algorithm)

Çekirdek yöntemleri, veri madenciliği ve makine öğrenmesinde yaygın olarak kullanılmaktadır. Çekirdek yöntemlerinin performansı büyük ölçüde, çekirdek parametresi ve düzenleme parametresi gibi bazı hiper parametrelerin seçimine bağlıdır. Bu nedenle model seçimi problemi, çekirdek yöntemlerinde önemli bir konu haline gelmektedir [52]. Çekirdek seçimi, öğrenme algoritmalarının genelleme hatasıyla yakından ilgilidir. En küçük genelleme hatasına sahip çekirdek, genellikle optimal çekirdek olarak kabul edilmektedir. Genellikle verilen verilerin altında yatan dağılım bilinmediği için; optimal çekirdek, genelleme hatasını doğrudan hesaplayamaz. Çekirdek seçiminde genelleme hatasının, ampirik hatadan ve hipotez uzay karmaşıklığından oluşan teorik üst sınırlarını kullanmak yaygın bir stratejidir [53].

Regresyon (Regression)

Regresyon, iki teori için kullanılan bir tekniktir. İlk olarak regresyon analizleri; genellikle uygulamaların makine öğrenimi alanıyla büyük örtüşmelere sahip olduğu tahmin için kullanılır. İkinci olarak regresyon analizi; bazı durumlarda bağımsız ve bağımlı değişkenler arasındaki nedensel ilişkileri belirlemek için kullanılabilir. Daha da önemlisi, regresyonlar tek başına; yalnızca bağımlı bir değişken ile farklı değişkenlerden oluşan sabit bir veri kümesi koleksiyonu arasındaki ilişkileri göstermektedir [54].

Regresyon modellerine göre, bağımsız değişkenler bağımlı değişkenleri öngörmektedir. Regresyon analizi; bağımsız değişken değerlerinin 'x' aralığından dolayı bağımlı 'y' değişken değerini tahmin eder [55]. Regresyonun; basit doğrusal regresyon (Simple linear regression), polinom regresyon (Polynomial regression) veya çok değişkenli doğrusal regresyon (Multivariate linear regression) gibi modelleri bulunmaktadır. Basit doğrusal regresyon, bağımsız değişkenlerin etkisini bağımlı değişkenlerin etkileşiminden ayıran tek bağımsız değişkenli bir durum modelidir [56]. Çok değişkenli doğrusal regresyon; bir dizi açıklayıcı değişken kullanarak, bir cevap değişkeninin sonucunu tahmin etmeye yönelik istatistiksel bir tekniktir. Amacı ise bağımsız değişkenler (x) ve analiz edilecek bağımlı değişken (y) arasındaki doğrusal ilişkiyi modellemektir. Polinom regresyon; bağımsız ve bağımlı değişkenler arasındaki ilişkinin n'inci derece polinom modellemesinde bir tür regresyon analizidir. Polinom regresyon, verilerin polinom denkleminin bağımlı ve bağımsız

değişkenlerin eğrisel etkileşimi ile harmanlandığı çok değişkenli doğrusal regresyonun özel bir durumudur [57].

3.5. Makine Öğrenmesi Performans Ölçütleri ile Değerlendirme ve Yorumlama

Bir sınıflandırıcı, model eğitim sürecinden sonra hesaplanan performans ölçütlerine dayalı olarak değerlendirilmektedir. Makine öğrenimi için çeşitli sınıflandırıcılar mevcut olmakla birlikte, bir sınıflandırıcının performansını değerlendirmek için hangi performans ölçütlerinin kullanılacağı konusunda uygulayıcılar arasında genel bir fikir birliği bulunmamaktadır. Bir sınıflandırıcı; belirli bir performans ölçütüne göre değerlendirildiğinde çok iyi performans gösterirken, aynı sınıflandırıcı başka bir performans metriğine göre daha düşük performans gösterebilir. Bu durumdan dolayı bir analist, sınıflandırıcı değerlendirmesi için yaygın olarak kullanılan birden fazla metrik seçmektedir. Sezgisel olarak ele alınırsa; bir sınıflandırıcı, sınıflandırıcı performans alanının benzersiz bir yönünü ayrı ayrı yakalayan bir dizi performans ölçütüne göre değerlendirilmelidir [58].

Makine öğrenimi sınıflandırma sistemlerinin performansını ölçmek için kullanılan ölçütlerden; karışıklık/hata matrisi, doğruluk, hata, hatırlama, kesinlik ve F1- Skor aşağıda açıklanmaktadır.

3.5.1. Karışıklık matrisi (Confusion matrix) / Hata matrisi (Error matrix)

Bir sınıflandırma problemi için öz nitelik sayısını azaltmak, çok araştırılan bir alandır. Sınıflandırma için en iyi nitelik kombinasyonunu bulmada kaba kuvvet yaklaşımı, mevcut n özelliğin tüm olası kombinasyonlarının denenmesini gerektiğinden dolayı öz nitelik seçimi; özellikle çok sayıda öz niteliğe sahip veri kümeleri için gereklidir. Birçok özelliğe sahip bu tür veri alanlarının örnekleri arasında metin kategorizasyonu ve gen ifadesi analizi yer almaktadır. Veri boyutunu azaltmaya ek olarak; daha az öz nitelik seçmek, sınıflandırmayı iyileştirebilir ve bu verileri oluşturan temel sürecin daha iyi anlaşılmasını sağlayabilir [59].

Karışıklık/hata matrisi, ikili sınıflandırma görevleri için bir sınıflandırıcının performansını özetlemenin bir yoludur. Bu kare matris, örnek sayısını mutlak veya göreli "gerçek sınıf" ve "tahmin edilen sınıf" oranları olarak listeleyen sütunlardan ve satırlardan oluşmaktadır [60].

İki sınıflı sınıflandırma problemi için karışıklık/hata matrisi Tablo 3.1'de gösterilmektedir:

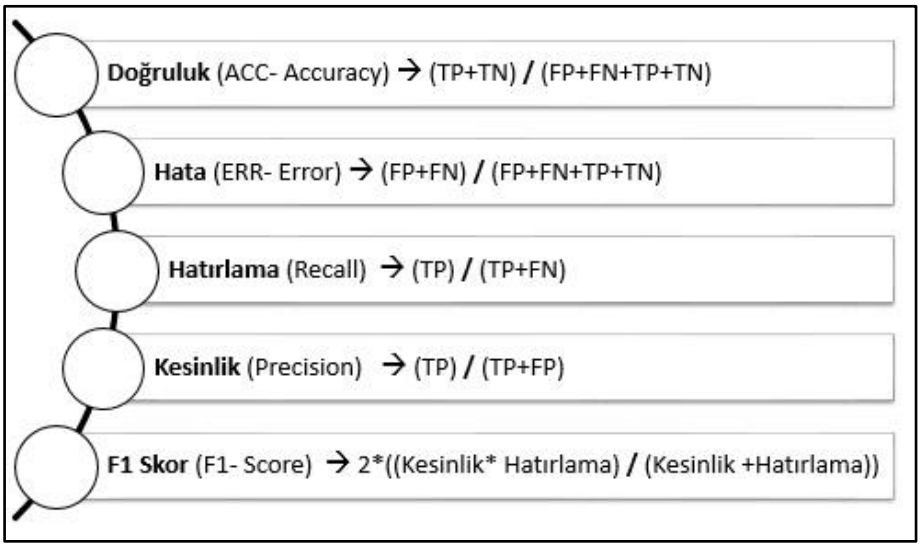
Tablo 3.1. Karışıklık/Hata matrisi

		Predicted Class (Tahmini Sınıf)	
		Negative (Negative)	*Positive (Pozitif)*
Actual Class (Gerçek Sınıf)	*Negative (Negative)*	True Negative (TN)	False Positive (FP)
	Positive (Pozitif)	False Negative (FN)	True Positive (TP)

Bir sınıflandırıcıyla ilişkili n x n boyutundaki bir karışıklık matrisi, tahmin edilen ve gerçek sınıflandırmayı gösterir. Burada n, farklı sınıfların sayısıdır. Tablo 3.1, n = 2 için bir karışıklık matrisini göstermektedir. Matristeki ifadelerin anlamları aşağıdaki gibidir [59];

- True Positive (TP): Doğru pozitif tahminlerin sayısı
- True Negative (TN): Doğru negatif tahminlerin sayısı
- False Positive (FP): Yanlış pozitif tahminlerin sayısı (gerçekte negatif olup, yanlış tahmin sonucu pozitif olarak sınıflandırılanların sayısı)
- False Negative (FN): Yanlış negatif tahminlerin sayısı (gerçekte pozitif olup, yanlış tahmin sonucu negatif olarak sınıflandırılanların sayısı)

Karışıklık matrisindeki ifadelere göre hesaplanan doğruluk, hata, hatırlama, kesinlik ve F1 skor değerleri Şekil 3.6'da gösterilmektedir:



Şekil 3.6. Performans ölçütleri

Şekil 3.6'da gösterilen ve TP, TN, FP, FN değerlerine göre hesaplanan performans değer ölçütleri aşağıda açıklanmaktadır.

3.5.2. Doğruluk (ACC- Accuracy) ve Hata (ERR- Error)

Doğruluk (ACC), makine öğreniminde en yaygın kullanılan performans metriğidir. ACC, sınıflandırıcının veri kümesinin boyutuna göre yaptığı doğru tahminlerin oranı olarak tanımlanır. Bir sınıflandırıcının sürekli çıktıları varsa (örneğin sinir ağları), bir eşik belirlenir ve bu eşiğin üzerindeki her şeyin pozitif olduğu tahmin edilir [61].

Hem tahmin doğruluğu (ACC) hem de sınıflandırma hatası (ERR), kaç örneğin yanlış sınıflandırıldığı hakkında genel bilgi sağlar. ERR, tüm yanlış tahminlerin toplamının (FP+FN) toplam tahmin sayısına (FP+FN+TP+TN) bölünmesi olarak anlaşılabilir. ACC, doğru tahminlerin toplamının (TP+TN) toplam tahmin sayısına (FP+FN+TP+TN) bölünmesiyle hesaplanır [60].

3.5.3. Hatırlama (Recall)

Hatırlama, bilgi erişiminde yaygın olarak kullanılmaktadır. Hatırlama, pozitif olarak tahmin edilen (TP) gerçek pozitiflerin (FN+TP) kesridir. Bu ölçüler, hiçbir şeyi tahmin etmeyerek veya her şeyi sırasıyla pozitif olarak tahmin ederek önemsiz bir şekilde maksimize edilir [61].

3.5.4. Kesinlik (Precision)

Kesinlik, duyarlılıktaki gibi bilgi erişiminde yaygın olarak kullanılmaktadır. Kesinlik, pozitif olarak tahmin edilen (TP) ve aslında pozitif olan örneklerin (TP+FP) oranıdır [61].

3.5.5. F1 Skor (F1- Score)

Günümüzde F1 ölçüsü; yalnızca ikili senaryoda değil, aynı zamanda çok sınıflı durumlarda da makine öğreniminin çoğu uygulama alanında yaygın olarak kullanılmaktadır. Belirli bir eşik için, F1-skoru, o eşikteki kesinlik (precision) ve hatırlamanın (recall) harmonik ortalamasıdır [61]. F1-skoru iki gerçek sayı kümesinin ayrımını değerlendiren ve genellikle SVM gibi sınıflandırıcılarla birlikte kullanılan basit bir öznitelik seçme filtre yöntemidir [62].

4. İLGİLİ LİTERATÜR

Veri madenciliğinin alt dalı olan metin madenciliği ile ilgili farklı alanlarda çeşitli çalışmalar yapılmıştır. Bilgi teknolojilerinin kullanımının artması ile birlikte veri miktarlarının hızla çoğalmasıyla farklı alanlarda yapılan çalışmalar ise hızla devam etmektedir.

Wu, Hsiao ve Nian [63] çalışmalarında metin sınıflandırması için denetimli makine öğrenmesi yöntemini kullanarak, gelişmiş istatistik dersindeki öğrencilerin ders notlarını tahmin etmeye çalışmışlardır. Öğrencilerin kişisel öğrenme ortamındaki öğrenme başarısını tahmin etmek için, kursla ilgili Facebook mesajlarından oluşan çevrim içi forumdaki 76 936 gönderi veri seti olarak kullanılmıştır. İlgili metin verileri ile ilgili üç sınıflandırma modeli oluşturulup, veriler "İstatistikle ilgili" ve "İstatistikle ilgili olmayan" gönderiler olarak sınıflandırılmıştır. Rastgele Orman algoritması, Destek Vektör Makinesi ve Yapay Sinir Ağı olmak üzere üç farklı yöntem kullanılmıştır. Destek Vektör Makinesi daha küçük eğitim kümeleriyle Yapay Sinir Ağları'ndan daha iyi performans göstermektedir. Destek Vektör Makinesinin Rastgele Orman ve Yapay Sinir Ağları Algoritmalarına göre daha tutarlı olduğu görülmüştür.

Şahin [64] çalışmasında metin madenciliği ve makine öğrenmesi yöntemleri ile internet sayfalarını 63 farklı internet sayfa kategorisine göre sınıflandırmıştır. Çalışmasında Python kodu ile verileri otomatik olarak toplamıştır. Sonrasında metin madenciliği yöntemleri uygulanarak veriler temizlenmiştir. Verilerin %83'ü eğitim seti olarak kullanılırken, %17'lik kısmı test veri seti olarak kullanılmıştır. Çalışma için toplanan internet siteleri üzerinde, iki farklı sınıflandırma yöntemi (ikili ve çoklu sınıflandırma) uygulanarak test edilmiş ve sonrasında performans karşılaştırması yapılmıştır. Performans değerlendirmesini yaparken başarım oranı, F1 skoru, kesinlik ve duyarlılık ölçütlerini kullanmıştır. Lojistik Regresyon algoritması ikili sınıflandırma yaklaşımında en iyi performans gösteren olurken, çoklu sınıflandırma yönteminde ise Destek Vektör Makineleri en yüksek performans gösteren algoritma olmuştur.

Çam [65] çalışmasında 4 farklı yazarın 6 farklı eseri ile ilgili yazarlık özelliklerini öklid uzaklığı yardımıyla çıkarmıştır. Yazarlık özellikleri elde edilirken İngilizce dili için çalışma yapmıştır. Metin madenciliği aşamalarından metinden anlamsız kelimelerin çıkarılması ve büyük harflerin küçük harfe çevrilmesi işlemlerinden sonra en sık tekrar edilen kelimeler,

zarflar ve sıfatlar bulunmuştur. Yazarı bilinmeyen bir dokümanın K-En Yakın Komşu algoritması kullanılarak tanımlanmış olan 4 farklı yazardan hangisine ait olabileceği belirlenmeye çalışılmıştır. Bu işlemler için R programlama dili kullanılmıştır. Çalışmada kullanılan veri setinin %70'lik kısmı eğitim, %30'luk kısmı test verisi olarak kullanılmıştır. Sınıflandırma başarısını ölçmek için doğruluk-hata oranı, duyarlılık, kesinlik ve F-ölçütü kullanılmıştır. Yapılan çalışma sonucu K-NN algoritmasının doğruluk oranı %85 olarak bulunmuştur.

Işık [66] çalışmasında denetimli makine öğrenmesi yöntemlerinden Naive Bayes, K-En Yakın Komşu algoritması ve Sıralı Minimal Optimizasyon algoritmalarını kullanarak sosyal medya yorumlarının duygu analizini Weka yazılımını kullanarak yapmıştır. Twitter platformundaki bazı e-ticaret firmalarına, hizmetlerine ve ürünlerine yönelik yapılan yorumlardan oluşan veri kümesi kullanılmıştır. Sosyal medya yorumları el ile olumlu, olumsuz ve nötr olarak etiketlenip üç farklı sınıfta toplanmıştır. Çalışmada; öz nitelik seçiminin ve sınıflardaki veri dağılımının sınıflandırma üzerindeki etkisi incelenmiştir. En iyi performans gösteren algoritma, öz nitelik seçimi yapıldığında K-NN olmuştur. Algoritmanın doğruluk oranı ise %93,52'dir.

Savaş ve Topaloğlu [67] çalışmalarında Twitter'da istihbarat için örnek bir uygulama geliştirmişlerdir. Türkiye İstatistik Kurumu'nun (TÜİK) 2013 yılına ait verilerine göre Türkiye'deki ceza davalarını inceleyerek çalışmalarındaki suç sınıflarını oluşturmuşlardır. Oluşturulan suç sınıfları bomba, terör, gözaltı, saldırı, gösteri, kaçakçılık, uyuşturucu, fuhuş ve tecavüz'dür. Çalışmalarında Python programlama dilinin Tweepy kütüphanesini kullanarak gerçek zamanlı tweet verilerini elde etmişlerdir. Kullanılan veri seti, 3 Mayıs 2016 ile 5 Haziran 2016 tarihleri arasında Twitter'dan elde edilen yaklaşık 900 MB boyutunda toplam 158 463 tweet verisinden oluşmaktadır. Bu tweet metin grubu 2 101 550 kelime ve 19 074 070 karakterden oluşmuştur. Türkçe dili doğal dil işleme süreçleri için Zemberek Kütüphanesi kullanılarak metin ön işleme süreçleri uygulanmıştır. Tweetlerdeki kelimeler Zemberek kullanılarak köklerine ayrılmış ve verilerde 301 690 kök bulunmuştur. Bulunan köklerin oluşturulan anahtar kelimelere göre yüzdeleri hesaplanmıştır. Bu anahtar kelimelerde "bomba" en yüksek yüzdeye (%24) sahip olurken, bunu %12, %8, %6 ve %6 oranlarıyla "terör", "saldırı", "örgüt" ve "patlama" takip etmiştir. Fakat "gösteri", "kaçakçılık" ve "fuhuş" anahtar kelimelerinin kökü bulunamamıştır. Ayrıca Zemberek analizini kontrol etmek için R programlama dili kullanılarak kelime bulutu oluşturulmuştur.

Oluşturulan kelime bulutu ile “bomba” kelimesinin kullanılma yüzdesinin yüksek olduğu görülmüştür.

Korkut [68] çalışmasında afet yönetimi ile ilgili başarı kriterlerini ortaya koymak için metin madenciliği tekniklerini kullanarak bir model oluşturmayı amaçlamıştır. Modelin uygulanmasında 2000-2016 yılları arasında yayınlanan 773 tane makaleden elde edilen 20 uzmana ait görüşler veri seti olarak kullanılmıştır. Çalışma için yapılan ön araştırma sonucu afet başarısına etki eden 9 alan tespit edilmiştir. Alt faktörler ile 122 başarı faktörü elde edilerek kritik olan faktörler belirlenmeye çalışılmıştır. Akademik uzmanların görüşlerini içeren makalelere metin ön işleme süreçleri uygulanarak metin madenciliği süreçleri R programlama dili ile yapılmıştır. Çalışmada metin madenciliğinden elde edilen verilere göre 20 kritik başarı faktörü elde edilmiştir.

Ak [69] çalışmasında metin madenciliği ile pazarlama alanında yayın yapan akademik dergileri değerlendirmiştir. Çalışmasında kullandığı veri setini SCOPUS veri tabanından elde etmiştir. Bu veri seti 25 derginin 2005-2018 yılları arasında yayınladığı 16 069 özet makaleden oluşmaktadır. Metin ön işleme süreçleri için R programlama dili kullanılmıştır. Makalelerin ana konu başlıkları belirlenmiştir. Konu başlıklarının yıl bazlı makalelerde geçme sıklığı yüzdesel olarak bulunmuştur. Çalışmasında, dergiye yayın göndermek isteyen bir araştırmacıya öneride bulunmak amacıyla, makalelerin çalışma alanlarının yüzdelik dağılımını göstermiştir.

Pasin [70] çalışmasında metin madenciliği ile Türkçe metinlerin sınıflandırmasını yapmıştır. Türk gazete şirketlerinin köşe yazılarından oluşan metin koleksiyonu (50 güne ait toplam 800 sütun), kategorize edilecek yapılandırılmamış verilerden oluşmaktadır. Çalışmada kullanılan K-En Yakın Komşu algoritması ve Naive Bayes algoritması R programı ile yazılmıştır. Sınıflandırma, “Cinsiyet Tespiti”, “Yazar Tespiti” ve “Tür Tespiti” olmak üzere üç kategoride incelenmiştir. İlk sınıflandırma olan cinsiyet tespiti sınıflandırmasında Naive Bayes yöntemi, K-NN yönteminden daha başarılıdır. Naive Bayes yöntemine göre erkek yazarların %90'ı doğru sınıflandırılırken, kadın yazarların %85'i doğru sınıflandırılmıştır. İkinci sınıflandırma olan yazar tespitinde her iki yöntemde de doğru sınıflandırma oranı %80 üzerindedir. Tür tespitinde; her iki yöntemde de diğer kategorilere göre daha belirleyici kelimeleri olduğundan dolayı, en iyi sonuç spor sütununa aittir. Uygulamasının sonuçlarına

göre Naive Bayes yöntemi, metni doğruluk oranına göre sınıflandırmak için kullanılan en başarılı algoritmalardan biridir.

Çelikel [71] çalışmasında metin madenciliği ve makine öğrenmesi yöntemlerini kullanarak hisse senedi fiyat tahminlemesi yapmaya çalışmıştır. Veri seti olarak Twitter platformundaki tweetler ile İstanbul Borsası ve Bloomberg'deki Türk ve Pegasus Hava Yolları' na ait veriler kullanılmıştır. Tweet verileri Python programlama dili ile elde edilmiştir. Borsa verilerinin, Twitter ile elde edilen tweetler ile ilişkisi R programlama dili ile incelenmiştir. Sınıflandırma algoritmaları uygulandığında modelin doğruluk oranı %67 iken yatırımcılar için oldukça verimsiz bir başarı oranı olmuştur. Bunun yerine 6 özelliğin üzerinde %95 doğruluk oranına sahip firma için yeni tahmin modeli oluşturulmuştur. Bu, regresyon modelinin yeni stok değerini %95 doğruluk oranıyla tahmin edebileceği anlamına gelir. Elde edilen sonuçlarla genelleme yapmak mümkün olmamakla birlikte yatırımcılar ve yatırım şirketleri için bir karar destek sistemi tabanı sağlamıştır.

Bilgin [72] çalışmasında metin madenciliği yöntemi ile 25 farklı divan edebiyatı şairine ait toplam 7 148 tane eserin yazarlarını belirlemeye çalışmıştır. Çalışmada Weka yazılımını kullanarak Java dilinde bir program yazmış ve modelleme için Rapid Miner programını kullanmıştır. Eserlere metin ön işleme aşamaları uygulanmıştır. Sınıflandırma için K-En Yakın Komşu, Naive Bayes, Karar Ağacı ve Destek Vektör Makinesi algoritmaları kullanılmıştır. Kurulan 20 farklı model ile yapılan analizler sonucu %91 doğruluk ve %90 F- Skor değerleri elde edilmiştir.

Tekin [73] çalışmasında yazılım geliştirme taleplerini metin madenciliği yöntemleri ile sınıflandırarak, taleplerin önem sırasını tahmin etmeye çalışmıştır. Kullanılan veri seti, bir şirkete ait talep yönetim sisteminin kayıtlarıdır. İşlenmemiş metin koleksiyonuna metin madenciliğinin ön işleme adımları uygulanıp, TF-IDF ağırlıklandırma yöntemi ile doküman-terim matrisi oluşturulmuştur. Sınıflandırma algoritmalarını uygulamaya hazır hale getirmek için açık kaynak kodlu PRETO aracı ve sınıflandırma için de Weka kullanılmıştır. Weka uygulamasında bulunan 5 farklı algoritma (Naive Bayes, Karar Ağacı, Rastgele Orman, Sıralı Minimum Optimizasyon ve Rotasyon Ormanı) ile sınıflandırma yapılmıştır. Deney sonuçlarına göre %74 F-Skor değeri ile en iyi sınıflandırmayı yapan Rastgele Orman algoritması olmuştur.

Goh ve Ubeynarayana [74] çalışmalarında metin madenciliği tekniklerini kullanarak inşaat kazası anlatı sınıflandırması yapmışlardır. Kullanılan veri seti "US OSHA (Occupational Safety and Health Administration, 2016)" internet sitesinden elde edilen 4471 inşaat sektörü vakasından oluşan verilerdir. Çalışmada Destek Vektör Makinesi, Doğrusal Regresyon, Rastgele Orman, K-En Yakın Komşu, Karar Ağacı ve Naive Bayes olmak üzere altı farklı makine öğrenme algoritması kullanılmıştır. Sınıflandırmada ortalama %73 tahmin oranı ile Destek Vektör Makinesi (SVM) en iyi performans gösteren algoritma olmuştur.

Fan, Yu, Yin, T. Wang, H. Wang [75] çalışmalarında 80 popüler proje ve GitHub'dan toplanan 252 000'den fazla sorun raporunu içeren büyük ölçekli bir veri seti üzerinde sorun sınıflandırma yaklaşımlarını araştırmışlardır. Veri seti 2014 tarihinde GitHub'dan elde edilen 1 185 konu başlıklı, 951 918 konudan oluşmaktadır. Metin ön işleme süreçlerinden sonra her konu, başlık ve açıklamalarına göre hatalı ve hatasız olmak üzere iki farklı kategoriye göre sınıflandırılmıştır. Çalışmalarında Naive Bayes, Lojistik Regresyon, Destek Vektör Makinesi ve Rastgele Orman algoritmalarını kullanmışlardır. Kullanılan sınıflandırma yöntemlerinden %69,7 ile %98,9 arasındaki değerler ile en başarılı algoritmanın Destek Vektör Makinesi olduğunu göstermişlerdir.

Dang ve Ahmad [76] çalışmalarında metin madenciliğini ve metin madenciliğinin uygulamalarını araştırmışlardır. Temel metin madenciliği teknolojilerinden bilgi alma, bilgi çıkarma, sınıflandırma ve kümeleme teknikleri açıklanmış ve tekniklerin karşılaştırmalarını yapmışlardır. Uygulama alanlarından akademik başvurular, biyoinformatik, telif hakkı ile müşteri profili analizi ve internet güvenliğinin öneminden bahsetmişlerdir. Metin madenciliği algoritmalarının zaman ve maliyeti azaltabileceği, faydalı ve yapılandırılmış veriler sağlayacağı anlatılmıştır.

Yıldırım, Saka ve Çeken [77] çalışmalarında metin madenciliği yöntemini kullanarak karaciğere ait bakteriyel enfeksiyonları için bilgi keşfi yapmışlardır. Karaciğerle ilgili olan en yüksek frekansa sahip bakterileri bulmak için Tıbbi Literatür Analizi ve Erişim Sistemi (MEDLINE) taranmıştır. Bakteri isimlerini ise "Medline MeSH data" dan almışlardır. Yüksek frekanslı bakteriler seçildikten sonra her bir bakteri için ilgili makaleler, ücretsiz bir biyomedikal veri tabanı olan PubMed'den taranmıştır. Öncelikle metin madenciliği yöntemleri uygulanmıştır. Sonrasında tedaviler için analizler (zamana bağlı değişimleri gösteren) kullanılıp ilgili tedaviler belirlenmiştir. Yapılan analiz sonuçları karaciğer

enfeksiyonları için daha iyi ilaçlara ihtiyacın olduğunu göstermiştir. Çalışmadaki metodoloji, literatürden zamana dayalı tıbbi bilgi elde etmek için bir referans modeli sunmaktadır.

Literatürde yapılan metin madenciliği çalışmaları ile farklı alanlardaki ihtiyaçlara göre farklı sorunlara çözümler getirilmiştir. Yukarıda bahsedilen çalışmaların amaçları aşağıda verilmektedir:

- Öğrencilerin kişisel öğrenme ortamındaki öğrenme başarısını tahmin etmek
- İnternet sayfalarını internet sayfa kategorisine göre sınıflandırmak
- Farklı yazarların eserlerine göre yazarlık özelliklerini çıkarmak
- Twitter platformundaki bazı e-ticaret firmalarına yapılan yorumları analiz etmek
- Twitter'da istihbarat için örnek bir uygulama yapmak
- Afet yönetimi ile ilgili başarı kriterlerini ortaya koymak
- Pazarlama alanında yayın yapan akademik dergileri değerlendirmek
- Türk gazete şirketlerinin köşe yazılarını sınıflandırmak
- Hisse senedi fiyat tahminlemesi yapmak
- Divan edebiyatı şairlerine ait eserlerin yazarlarını belirlemek
- Yazılım geliştirme taleplerinin önem sırasını tahmin etmek
- İnşaat kazası anlatı sınıflandırması yapmak
- GitHub'dan toplanan sorun raporlarını sınıflandırmak
- Metin madenciliğini ve metin madenciliğinin uygulamalarını araştırmak
- Karaciğere ait bakteriyel enfeksiyonları için bilgi keşfi yapmak

Bu tez çalışmasında ise DergiPark-Akademik internet sitesindeki araştırma makalelerinin metin madenciliği yöntemi ile sınıflandırılması amaçlanmaktadır. Çalışma kapsamındaki metin verilerinin elde edilmesi, işlenmesi ve sınıflandırılması aşamaları; metin madenciliği yapmak isteyen herkesin kendi çözümlerini bulmak için faydalanabileceği bir uygulama niteliğindedir. Tezde yapılan araştırma makalelerinin sınıflandırılması, akademik çalışma yapmak isteyen bir öğrencinin ya da akademisyenin ön araştırma süreçlerinde fayda sağlayabileceği bir çalışmadır. Bu çalışma ile akademik çalışma yapan birisinin elde edebileceği bazı faydalar aşağıdaki gibidir:

- Araştırma yaptığı konu için incelemek istediği akademik yayın verilerini internet sitesinden elde edebilir. Bu çalışmada DergiPark-Akademik internet sitesindeki araştırma makale verileri elde edilmiştir. Araştırmacı farklı bir internet sitesindeki akademik yayın verilerini elde etmek isterse, yazılan Python kodundaki link kısmına, ilgili internet sitesinin linkini yazarak kendi çalışmasına uygulayabilir.
- Elde ettiği akademik yayınlar ile ilgili anahtar kelimeleri bularak yayınların içeriği hakkında fikir sahibi olabilir. Bu çalışmada internetten elde edilen araştırma makalelerin öz (abstract) kısmına metin ön işleme yapılmıştır. Bu süreçte, TF-IDF yöntemi ile makalelerdeki önemli kelimeler bulunmuştur. Akademik çalışma yapan birisi; anahtar kelimeleri bulmak için yazılan Python kodundaki algoritmalarda değişiklik yaparak, farklı algoritmalar da deneyebilir.
- Çalışmasındaki ihtiyacına göre incelediği yayınlarda bulunan anahtar kelimeler ile farklı sınıflandırmalar yapabilir. Sınıflandırma sonucu ile incelediği spesifik kelime, konu başlığı ya da konu alanına göre elde edilen yayınları inceleyebilir. Bu yayınların sınıflandırma çeşidine göre genel dağılımlarını elde edebilir. Bu çalışma kapsamında DergiPark-Akademik internet sitesinden elde edilen araştırma makalelerine metin ön işleme aşaması uygulandıktan sonra sınıflandırma işlemi yapılmıştır. Tezde denetimli makine öğrenmesi algoritmalarından Naive Bayes, Rastgele Orman ve Destek Vektör Makinesi algoritmaları kullanılmıştır. Akademik çalışma yapan birisi yazılan Python kodundaki algoritmalarda değişiklik yaparak farklı algoritmaları kullanmayı da tercih edebilir.

5. MATERYAL VE YÖNTEM

Bu çalışma kapsamında metin madenciliği yöntemi ile denetimli makine öğrenmesi sınıflandırma algoritmaları kullanılarak araştırma makalelerinin konularına göre sınıflandırılması amaçlanmaktadır.

5.1. Uygulamada Kullanılan Programlama Dili – Python

Python programlama dili; verileri kolaylıkla analiz edebilmesi, hızlı olması, kullanışlı arayüzü olması ve geniş kütüphanelere sahip olması gibi nedenlerden dolayı uygulamada tercih edilmiştir. Python programlama dilinin öğrenilmesi, yazılması ve okunması kolay olmakla birlikte metin madenciliği için yazılan kütüphaneleri de çok gelişmiştir. Bu kütüphanelerin yardımı ile metin madenciliği süreçlerindeki verinin elde edilmesi, işlenmesi ve sınıflandırılması kolay bir şekilde yapılabilmektedir.

Uygulamada, tercih edilen Python editörlerinden PyCharm kullanılmıştır. JetBrains firması tarafından sunulan ve son derece gelişmiş olan PyCharm editörü, ticari olmakla birlikte editörün ücretsiz sürümü de bulunmaktadır [20]. Topluluk seçeneği ile kullanılan PyCharm için işletim sistemine göre MacOs, Windows ve Linux seçeneklerinden uygun olan ücretsiz sürüm indirilip kurulabilir. PyCharm; Python ile proje geliştirme, gelişmiş hata ayıklama, otomatik komut tamamlama ve sürüm kontrolü gibi özelliklere de sahiptir [20].

Çalışma kapsamında Python kütüphanelerinden Request, Beautiful Soup, Pandas, Doğal Dil Araç Seti (NLTK), Scikit-Learn kullanılmıştır. Python da “WebKazima.py” ve “MetinIsleme.py” isimli iki farklı sınıf oluşturulmuştur. “WebKazima.py” sınıfında Request, Beautiful Soup ve Pandas kütüphaneleri; “MetinIsleme.py” sınıfında ise Pandas, Doğal Dil Araç Seti (NLTK) ve Scikit-Learn kütüphaneleri kullanılmıştır.

5.1.1. Request

Request kütüphanesi, Python’da HTTP istekleri yapmak için kullanılan bir kütüphanedir. Güzel ve basit bir uygulamanın arkasında, istekte bulunmanın karmaşıklığını soyutladığı için hizmetlerle etkileşime ve uygulamada veri tüketmeye odaklanılabilmektedir. HTTP yöntemleri, bir HTTP isteği yaparken hangi eylemi gerçekleştirmeye çalışıldığını belirler.

En yaygın HTTP yöntemlerinden biri “GET” dir. “GET” yöntemi ile belirli bir kaynaktan veri elde etmeye veya geri almaya çalışır [78].

Çalışma kapsamında, Request modülü “WebKazima.py” sınıfında DergiPark-Akademik internet sitesine HTTP isteği yapmak için kullanılmaktadır.

5.1.2. Beautiful Soup

Beautiful Soup, HTML ve XML dosyalarından veri çeken bir Python kitaplığıdır. Ayrıştırma ağacında (parse tree) gezinmenin, aramanın ve değiştirmenin ifade tarzına uygun (idiomatic) gezinme yollarını sağlamak için ayrıştırıcı ile çalışır. Genellikle, programcıları saatlerce veya günlerce çalışmaktan kurtarır. Beautiful Soup, karmaşık bir HTML belgesini karmaşık bir Python nesnesi ağacına dönüştürür [79].

Çalışma kapsamında, Beautiful Soup kitaplığı “WebKazima.py” sınıfında Request kütüphanesi ile erişim sağlanan DergiPark-Akademik internet sitesinin HTML belgelerini ayrıştırmak için kullanılmaktadır.

5.1.3. Pandas

2008'den beri geliştirilmekte olan Pandas Kütüphanesi, Python ile çok sayıda alana özgü istatistiksel hesaplama platformu ve veri tabanı dili arasındaki veri analizi araçlarının zenginliğindeki boşluğu kapatmayı amaçlamaktadır. Başlangıçta finansal veri analizi uygulamaları için geliştirilmiştir. Sonrasında, Pandas kütüphanesinin bilimsel Python'un hem akademik hem de endüstri uygulayıcıları için daha çekici ve pratik bir istatistiksel hesaplama ortamı olmasını sağlayacağı düşünülmektedir. Kütüphanenin adı, ekonometri ve istatistikte sıklıkla karşılaşılan yüksek boyuta sahip veri kümeleri için ortak bir terim olarak ifade edilen panel veriden gelmektedir. Pandas kütüphanesi benzer şekilde adlandırılmış bir veri çerçevesi sınıfı sağlamaktadır [80].

Çalışma kapsamında, Pandas kütüphanesi hem “WebKazima.py” hem de “MetinIsleme.py” sınıfında makaleler için veri çerçevesi sınıfı oluşturma amacıyla kullanılmaktadır.

5.1.4. Doğal Dil Araç Seti (NLTK- Naturel Language Toolkit)

Doğal dil araç seti (NLTK) birçok doğal dil işleme veri türleri, işleme görevi, korpus örnekleri ve okuyucular sağlayan bir Python modülleri paketidir. Veri türleri arasında belirteçler, etiketler, parçalayıcılar, ağaçlar ve özellik yapıları bulunmaktadır. Belirteçler (tokenizers), kök ayırıcılar (stemmers), etiketleyiciler (taggers), parçalayıcılar (chunkers), ayrıştırıcılar (parsers), kümeleyiciler (clusterers) ve sınıflandırıcılar (classifiers) için ara birim tanımları ve başvuru uygulamaları sağlanır [81].

Çalışma kapsamında, NLTK kütüphanesi “MetinIsleme.py” sınıfında metin madenciliği ile ilgili süreçlerde kullanılmaktadır.

5.1.5. Scikit-Learn

Scikit-learn, makine öğrenimi algoritmalarını uygulamak için standart bir ara yüz sağlayan bir Python kitaplığıdır. Scikit-learn kitaplığı; veri ön işleme adımları, veri yeniden örnekleme teknikleri, değerlendirme parametreleri ve bir algoritmanın performansını ayarlamak/optimize etmek için arama ara yüzleri gibi makine öğrenimi ardışık düzenine entegre olan diğer yardımcı işlevleri içermektedir [82].

Çalışma kapsamında, Scikit-learn kitaplığı “MetinIsleme.py” sınıfında makine öğrenmesi sınıflandırma algoritmalarının uygulanması süreçlerinde kullanılmaktadır.

5.2. Veri Elde Etme

Çalışma için kullanılan veri seti, web madenciliği yöntemi ile Python kodu yazılarak elde edilen araştırma makalelerinden oluşmaktadır. Bu metin koleksiyonu, DergiPark-Akademik internet sitesindeki "Tıp", "Sosyal", "Temel Bilimler" ve "Mühendislik" konu başlıklarındaki; toplamda 16 konu alanındaki makaleleri içermektedir. Makale verileri, yazım dili İngilizce ve Türkçe olan makaleler olmak üzere iki ayrı metin koleksiyonundan oluşturulmuştur.

Uygulama için yazılan Python projesinde iki farklı dosya oluşturulmuştur. İlk dosya “Web Madenciliği”, ikinci dosya ise “Metin İşleme” ve “Makine Öğrenmesi Sınıflandırması” işlemlerini yapan Python kodlarını içermektedir. Web madenciliği için yazılan Python

koduna “Request”, “Beautiful Soup”, “Pandas” ve zaman işlemleri için “Time” kütüphaneleri eklenmiştir. Kütüphane ekleme ile ilgili Python kodu Şekil 5.1’de verilmiştir:

```
import requests
from bs4 import BeautifulSoup as bs
import pandas as pd
import time
```

Şekil 5.1. Web kazıma Python kod parçacığı – kütüphane ekleme

Web kazıma Python sınıfına Şekil 5.1’deki kütüphane eklentileri yapılmaktadır. Sonrasında “Request” modülü ile “https://dergipark.org.tr/tr/” sitesine HTTP isteği gönderiliyor. Sayfa erişiminin sağlanıp sağlanmadığı durum kodu ile kontrol edilmektedir. Sayfa erişimi sağlandıktan sonra HTML belgelerini ayrıştırmak için “Beautiful Soup” modülü kullanılmaktadır. “Request” ve “Beautiful Soup” kütüphanelerinin kullanıldığı Python kodu Şekil 5.2’de verilmiştir:

```
url = "https://dergipark.org.tr/tr/"
r = requests.get(url)
# r.status_code = 200 sayfa erişimi sağlandı

soup = BeautifulSoup(r.content, "lxml") # html sitesini, lxml ile parçalandı
```

Şekil 5.2. Web kazıma Python kod parçacığı- "Request" ve "Beautiful Soup" modülü kullanımı

Şekil 5.2’de verilen Python kod parçacığı ile DergiPark-Akademik internet sitesindeki veriler; XML ve HTML dosyalarının kolayca ayrıştırılmasını destekleyen “lxml” ile alınmaktadır.

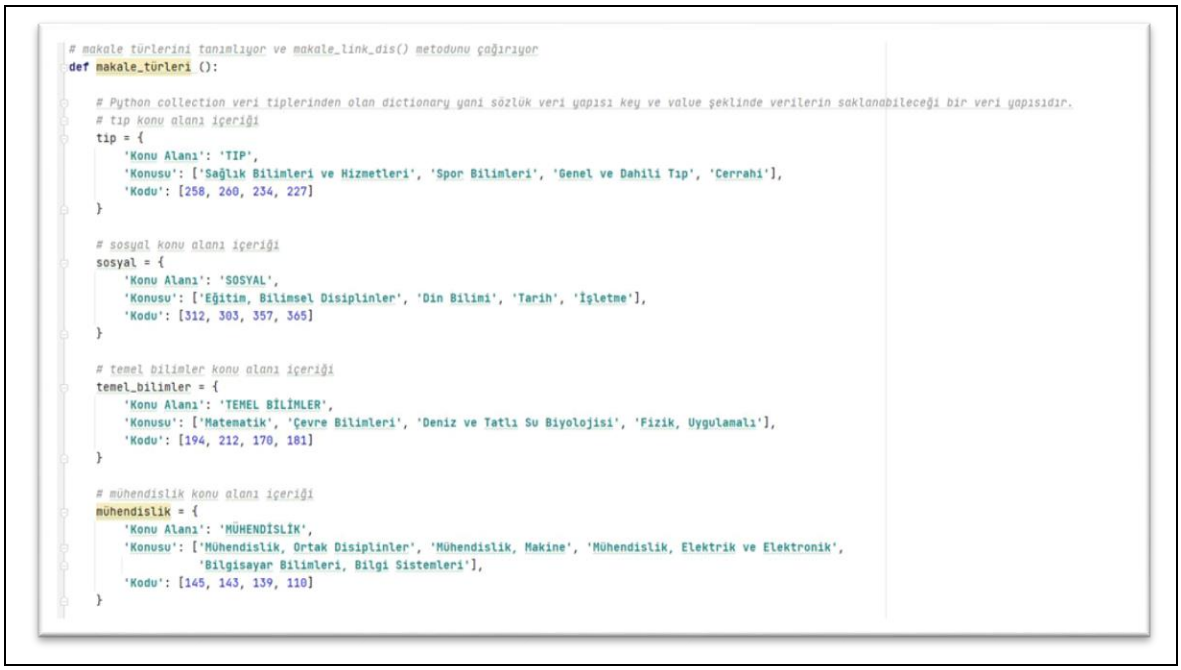
5.3. Veri Seti Oluşturma Süreci

Çalışma kapsamında, DergiPark-Akademik internet sitesindeki verilere web madenciliği yöntemi ile ulaştıktan sonra veri seti oluşturma aşamasına geçilmektedir. Veri seti oluşturma süreçleri için Python’da “WebKazima.py” sınıfı oluşturulmuştur. Veri seti; DergiPark-

Akademik internet sitesindeki en çok makale sayısının olduğu "Tıp", "Sosyal", "Temel Bilimler" ve "Mühendislik" konu alanlarındaki araştırma makalelerinden oluşturulmuştur. Her konu alanından, makale sayısı en çok olan 4 farklı konu alan seçimi yapılmıştır. Konu alanına göre seçilen konu başlıkları aşağıda verildiği gibidir:

- *Tıp Konu Alanındaki Konular*: “Sağlık Bilimleri ve Hizmetleri”, “Spor Bilimleri”, “Genel ve Dahili Tıp”, “Cerrahi”
- *Sosyal Konu Alanındaki Konular:* “Eğitim, Bilimsel Disiplinler”, “Din Bilimi”, “Tarih”, “İşletme”
- *Temel Bilimler Konu Alanındaki Konular:* “Matematik”, “Çevre Bilimleri”, “Deniz ve Tatlı Su Biyolojisi”, “Fizik, Uygulamalı”
- *Mühendislik Konu Alanındaki Konular:* “Mühendislik, Ortak Disiplinler”, “Mühendislik, Makine”, “Mühendislik, Elektrik ve Elektronik”, “Bilgisayar Bilimleri, Bilgi Sistemleri”

Konu alanlarındaki makaleler, Python sözlük veri yapısında (dictionary) saklanmaktadır. Şekil 5.3’te sözlük veri yapısı gösterilmektedir:



```
# makale türlerini tanımlıyor ve makale_link_dis() metodunu çağırıyor
def makale_türleri ():

    # Python collection veri tiplerinden olan dictionary yani sözlük veri yapısı key ve value şeklinde verilerin saklanabileceği bir veri yapısıdır.
    # tıp konu alanı içeriği
    tip = {
        'Konu Alanı': 'TIP',
        'Konusu': ['Sağlık Bilimleri ve Hizmetleri', 'Spor Bilimleri', 'Genel ve Dahili Tıp', 'Cerrahi'],
        'Kodu': [258, 260, 234, 227]
    }

    # sosyal konu alanı içeriği
    sosyal = {
        'Konu Alanı': 'SOSYAL',
        'Konusu': ['Eğitim, Bilimsel Disiplinler', 'Din Bilimi', 'Tarih', 'İşletme'],
        'Kodu': [312, 303, 357, 365]
    }

    # temel bilimler konu alanı içeriği
    temel_bilimler = {
        'Konu Alanı': 'TEMEL BİLİMLER',
        'Konusu': ['Matematik', 'Çevre Bilimleri', 'Deniz ve Tatlı Su Biyolojisi', 'Fizik, Uygulamalı'],
        'Kodu': [194, 212, 170, 181]
    }

    # mühendislik konu alanı içeriği
    mühendislik = {
        'Konu Alanı': 'MÜHENDİSLİK',
        'Konusu': ['Mühendislik, Ortak Disiplinler', 'Mühendislik, Makine', 'Mühendislik, Elektrik ve Elektronik',
                   'Bilgisayar Bilimleri, Bilgi Sistemleri'],
        'Kodu': [145, 143, 139, 110]
    }
```

Şekil 5.3. Web kazıma Python kod parçacığı- konu alanı sözlük veri yapısı

Şekil 5.3’teki Python kod parçacığında, “makale_turleri” metodu ile her konu alanı sözlük veri yapısına göre tanımlanmaktadır. Sözlük veri yapısı içeriğinde “Konu Alanı”, “Konusu”

ve “Kodu” yer almaktadır. “Konu Alanı” makalenin hangi alanda olduğunu, “Konusu” konu alanı ile ilgili tanımlanan 4 farklı konu başlığını içermektedir. “Kodu” ise web madenciliği işlemleri için kullanılmaktadır.

Makale sözlük veri yapıları, “makale_turleri” metodu içerisinde tanımlanmaktadır. Sonrasında aynı metot içerisinde, makale linklerine ulaşmak için “makale_link_dis” metodu çağrılmaktadır. “makale_link_dis” metodunun çağrıldığı Python kod parçacığı Şekil 5.4’te gösterildiği gibidir:

```
    # makale_konu_alani: tıp----------------------------------------------
    # makale_konusu: Sağlık Bilimleri ve Hizmetleri ,  makale_kodu: 258
    makale_link_dis(tip['Konu Alanı'],tip['Konusu'][0],tip['Kodu'][0] )

    # makale_konusu: Spor Bilimleri ,  makale_kodu: 260
    makale_link_dis(tip['Konu Alanı'],tip['Konusu'][1],tip['Kodu'][1] )

    # makale_konusu: Genel ve Dahili Tıp ,  makale_kodu: 234
    makale_link_dis(tip['Konu Alanı'],tip['Konusu'][2],tip['Kodu'][2] )

    # makale_konusu: Cerrahi ,  makale_kodu: 227
    makale_link_dis(tip['Konu Alanı'],tip['Konusu'][3],tip['Kodu'][3] )
    #-------------------------------------------------------------------


    # makale_konu_alani: sosyal-----------------------------------------
    # makale_konusu: Eğitim, Bilimsel Disiplinler ,  makale_kodu: 312
    makale_link_dis(sosyal['Konu Alanı'], sosyal['Konusu'][0], sosyal['Kodu'][0])

    # makale_konusu: Din Bilimi ,  makale_kodu: 303
    makale_link_dis(sosyal['Konu Alanı'], sosyal['Konusu'][1], sosyal['Kodu'][1])

    # makale_konusu: Tarih ,  makale_kodu: 357
    makale_link_dis(sosyal['Konu Alanı'], sosyal['Konusu'][2], sosyal['Kodu'][2])

    # makale_konusu: İşletme ,  makale_kodu: 365
    makale_link_dis(sosyal['Konu Alanı'], sosyal['Konusu'][3], sosyal['Kodu'][3])
    #-------------------------------------------------------------------------


    # makale_konu_alani: temel_bilimler---------------------------------------------------------------
    # makale_konusu: Matematik ,  makale_kodu: 194
    makale_link_dis(temel_bilimler['Konu Alanı'], temel_bilimler['Konusu'][0], temel_bilimler['Kodu'][0])

    # makale_konusu: Çevre Bilimleri ,  makale_kodu: 212
    makale_link_dis(temel_bilimler['Konu Alanı'], temel_bilimler['Konusu'][1], temel_bilimler['Kodu'][1])

    # makale_konusu: Deniz ve Tatlı Su Biyolojisi ,  makale_kodu: 170
    makale_link_dis(temel_bilimler['Konu Alanı'], temel_bilimler['Konusu'][2], temel_bilimler['Kodu'][2])

    # makale_konusu: Fizik, Uygulamalı ,  makale_kodu: 181
    makale_link_dis(temel_bilimler['Konu Alanı'], temel_bilimler['Konusu'][3], temel_bilimler['Kodu'][3])
    #-----------------------------------------------------------------------------------------------


    # makale_konu_alani: mühendislik------------------------------------------------------
    # makale_konusu: Mühendislik, Ortak Disiplinler ,  makale_kodu: 145
    makale_link_dis(mühendislik['Konu Alanı'], mühendislik['Konusu'][0], mühendislik['Kodu'][0])

    # makale_konusu: Mühendislik, Makine ,  makale_kodu: 143
    makale_link_dis(mühendislik['Konu Alanı'], mühendislik['Konusu'][1], mühendislik['Kodu'][1])

    # makale_konusu: Mühendislik, Elektrik ve Elektronik ,  makale_kodu: 139
    makale_link_dis(mühendislik['Konu Alanı'], mühendislik['Konusu'][2], mühendislik['Kodu'][2])

    # makale_konusu: Bilgisayar Bilimleri, Bilgi Sistemleri ,  makale_kodu: 110
    makale_link_dis(mühendislik['Konu Alanı'], mühendislik['Konusu'][3], mühendislik['Kodu'][3])
    #-----------------------------------------------------------------------------------------
```

Şekil 5.4. Web kazıma Python kod parçacığı- “makale_turleri” metodu içerisinde, "makale_link_dıs" metodunun çağrılması

Şekil 5.4'teki Python kod parçacığı "makale_turleri" metodunun içerisinde yer almaktadır. "Tıp", "Sosyal", "Temel Bilimler" ve "Mühendislik" konu alanlarındaki 4 ayrı konudaki makalelere ulaşmak için "makale_link_dis" metodu çağrılmaktadır. "makale_link_dis" metodu; "Konu Alanı", "Konusu" ve "Kodu" bilgilerini içeren üç farklı parametre almaktadır. Her konu alanı için "makale_link_dis" metoduna giden parametreler aşağıda açıklanmaktadır:

- "Tıp" konu alanı için sözlük veri yapısında dizi (array) olarak tanımlanan "Sağlık Bilimleri ve Hizmetleri", "Spor Bilimleri", "Genel ve Dahili Tıp", "Cerrahi" konu alanları ve her konu alanının kodu ayrı ayrı "makale_link_dis" metoduna gönderilmektedir.
- "Sosyal" konu alanı için "Eğitim, Bilimsel Disiplinler", "Din Bilimi", "Tarih" ve "İşletme" konu alanları ve her konu alanının kodu ayrı ayrı "makale_link_dis" metoduna gönderilmektedir.
- "Temel Bilimler" konu alanı için "Matematik", "Çevre Bilimleri", "Deniz ve Tatlı Su Biyolojisi", "Fizik, Uygulamalı" konu alanları ve her konu alanının kodu ayrı ayrı "makale_link_dis" metoduna gönderilmektedir.
- "Mühendislik" konu alanı için "Mühendislik, Ortak Disiplinler", "Mühendislik, Makine", "Mühendislik, Elektrik ve Elektronik", "Bilgisayar Bilimleri, Bilgi Sistemleri" konu alanları ve her konu alanının kodu ayrı ayrı "makale_link_dis" metoduna gönderilmektedir.

"makale_link_dis" metoduyla alınan parametreler ile hangi konu alanının hangi konusundaki makale linklerine erişileceği bilgisi alınmaktadır. İlk defa "makale_turleri" metodu çalıştırıldığında; "makale_link_dis" metoduna "Tıp" konu alanındaki "Sağlık Bilimleri ve Hizmetleri" konusu ile ilgili parametre bilgileri gönderilmektedir. "makale_link_dis" metodu ile DergiPark-Akademik internet sitesindeki araştırma makalesi linklerine ulaşılmaktadır. İlk önce, "makale_link_dis" metodu ile ulaşılan makale linklerinin kaç internet sayfasından oluştuğu bilgisi alınmaktadır. Seçilen konudaki makale linklerinin kaç sayfa olduğunu bulabilmek için "makale_link_dis" metodundaki "sayfa_sayisi" metodu çağrılmaktadır. Bir parametre alan "sayfa_sayisi" metodu Şekil 5.5'te gösterilmektedir:

```python
# seçilen konu ile ilgili toplam sayfa sayısını döndüren method, ilgili alan kodu girdi olarak alıyor
def sayfa_sayisi (alan_kodu) :
    url = "https://dergipark.org.tr/tr/search?q=&section=articles&aggs[articleType.id][0]=54&aggs[subjects.id][0]=" + str(alan_kodu)
    r = requests.get(url)
    # r.status_code = 200 sayfa erişimi sağlandı

    soup = bs(r.content, "lxml") # html sitesini, lxml ile parçalandı

    # sayfa sayılarının olduğu sondan bir önceki li yi bulma
    sayfa_sayisi_li = len(soup.find_all("ul", attrs={
        "class": "kt-pagination__links mx-auto"})[0].find_all("li")) - 2

    # sondan bir önceki li içerisine giderek toplam sayfa sayısını bulma
    sayfa_sayisi = soup.find_all("ul", attrs={
        "class": "kt-pagination__links mx-auto"})[0].find_all("li")[
        sayfa_sayisi_li].text
    return sayfa_sayisi
```

Şekil 5.5. Web kazıma Python kod parçacığı- "sayfa_sayisi" metodu

Şekil 5.5'teki "sayfa_sayisi" metodu parametre olarak "alan_kodu" bilgisini almaktadır. Makale URL adresleri, makalelerin konusuna göre değişmektedir. URL adres bilgisi, alınan "alan_kodu" parametresi ile tanımlanmaktadır. Sonrasında, internet sitesine erişim sağlanmaya çalışılmaktadır. Sayfa erişiminin sağlanıp sağlanmadığının kontrolü için "request" kütüphanesinin durum kodu (status_code) komutu çalıştırılabilir. Durum kodunun "200" olması sayfa erişiminin sağlandığını göstermektedir. Erişimi sağlanan internet sayfasının alt bölümünde, sayfa numaraları gözükmektedir. Sayfa numaralarının bilgisi Şekil 5.6'da gösterilmektedir:

Şekil 5.6. DergiPark-Akademik internet sitesindeki makale linklerinin sayfa sayısı

Şekil 5.6'da DergiPark-Akademik internet sitesindeki, seçilen konu alanındaki makale linklerinin toplamda kaç sayfa olduğu bilgisinin bir örneği gösterilmektedir. Toplam sayfa bilgisi "sayfa_sayisi" metodu ile elde edilmektedir. "makale_link_dis" metodunda çağrılan "sayfa_sayisi" metodu ile seçilen konu alanının sayfa sayısı bilgisi elde edilmektedir. Elde edilen sayfa sayısı bilgisi; "makale_link_dis" metodunda tanımlanan "toplam_sayfa" değişkenine atanmaktadır.

Belirli bir konu alanı ile ilgili ulaşılan makale linklerinin sayfa sayısı bilgisi elde edildikten sonra “for” döngüsü ile her sayfaya sırası ile erişim sağlanmaktadır. Şekil 5.7’de verilen “makale_link_dis” metodunun ilk kısmında görüldüğü gibi sayfa adedi kadar “for” döngüsü devam etmektedir:

```
# ilgili alan kodu ile ilgili sayfalardaki makale linklerini alınıyor, sayfa_sayisi() ve  makale_link_ic() metodunu çağırıyor
def makale_link_dis(makale_konu_alani, makale_konusu, makale_kodu):

    toplam_sayfa = sayfa_sayisi(makale_kodu)  # seçilen alandaki sayfa sayısı

    for sayfa_numarası in range(1, int(toplam_sayfa) + 1):  # arama sonucundaki sayfa sayısı kadar döngü devam edecek
        url_sayfa = requests.get("https://dergipark.org.tr/tr/search/"
                                 + str(sayfa_numarası)
                                 + "?q=&section=articles&aggs[articleType.id][0]=54&aggs[subjects.id][0]="
                                 + str(makale_kodu),
                                 timeout=3)  ##Error 429 Slow down! Too many requests" hatası almamak için 3 saniye delay eklendi

        #print(url_sayfa.status_code) #r.status_code = 200 sayfa erişimi sağlandı
        soup_sayfa = bs(url_sayfa.content, "lxml")
        makale_body = soup_sayfa.find_all("div", attrs={
            "class": "card-body"})  # sayfadaki bütün makalelerin body bilgisi alındı
```

Şekil 5.7. Web kazıma Python kod parçacığı- makale linklerine erişim

Şekil 5.7’deki “makale_link_dis” metodu ile bütün internet sayfalarına erişimi sağlanmaya çalışılmaktadır. Erişilen sayfalardaki her bir makalenin link bilgisi elde edilmektedir.

Dergi-Park Akademik sitesi; yazılan Pyhton kodunun, hızlı ve seri bir şekilde makale linklerine erişim sağlamaya çalışmasını saldırı olarak algılayabilmektedir. Bu nedenle makale linklerine erişim sağlanırken, her bir link erişimi için 3 saniye bekleme süresi (delay) eklenmiştir. “makale_link_dis” metodu, aldığı “makale_kodu” parametresi ile çağırdığı “sayfa_sayisi” metoduyla makalelerin toplam sayfa sayısını elde etmektedir ve “for” döngüsü ile bütün makale linklerine ulaşmaktadır. Makale linklerine erişim sağlandıktan sonra her bir link içerisine girilerek, makaleler ile ilgili detayların yer aldığı “body” bilgisi alınmaktadır. Alınan “body” bilgisi, Şekil 5.8’de verilen sözlük veri yapısı (dictionary) ile tutulmaktadır:

```
    for makale_detay in makale_body:
        makale_icerigi = {
            'link': '',
            'dil': '',
            'baslik': '',
            'konu alani' : '',
            'konusu': '',
            'oz': '',
            'anahtar kelime': ''
        }
        makale_url = makale_detay.a.get("href")  # makale linklerini makale_url değişkenine atandı
        makale_icerigi['link'] = makale_url
        makale_icerigi['konu alani'] = makale_konu_alani
        makale_icerigi['konusu'] = makale_konusu

        makale_link_ic(makale_url,makale_icerigi) # makale linklerinin içine linkleri alınan method ile girildi

        makale.append(makale_icerigi)
        time.sleep(3)  # "Error 429 Slow down! Too many requests" hatası almamak için 3 saniye delay eklendi
return
```

Şekil 5.8. Web kazıma Python kod parçacığı- makale sözlük veri yapısı (dictionary)

Şekil 5.8'de makaleler için sözlük veri yapısı tanımlanmıştır. Sözlük veri yapısı için tanımlanan anahtar (key) değerleri; "link", "dil", "baslik", "konu alani", "konusu", "oz" ve "anahtar kelime" bilgilerinden oluşmaktadır. Anahtar (key) ve değerleri (value) Tablo 5.1'de gösterilmektedir:

Tablo 5.1. Makale sözlük veri yapısının "Anahtar" ve "Değer" içerikleri

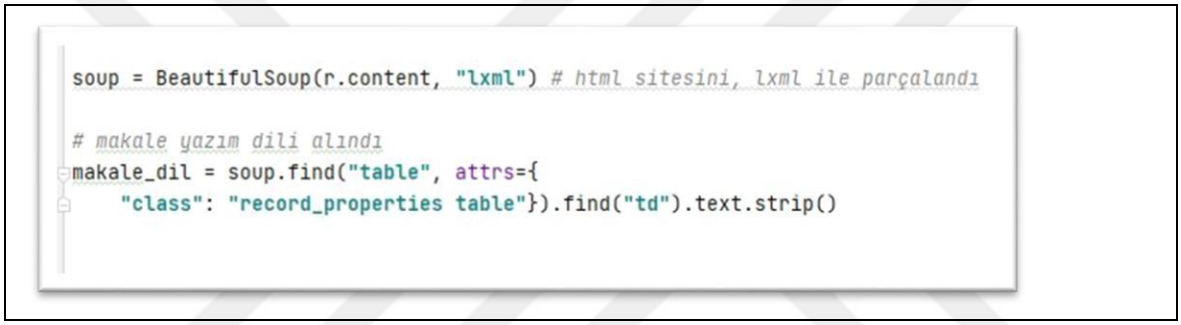
Anahtar (Key)	**Değer (Value)**
Link	Makale linkinin URL si
Dil	Makalenin yazım dili
Başlık	Makalenin konu başlığı
Konu Alanı	Makalenin konu alanı (Çalışma için "Tıp", "Sosyal", "Temel Bilimler" ve "Mühendislik" konularından birisi)
Konusu	Konu alanının konusu
Öz	Makalenin öz (abstract) bölümünün içeriği
Anahtar Kelime	Makalenin anahtar kelimeleri

Tablo 5.1'de gösterilen anahtar değerler için her bir makalenin değerleri sözlük veri yapısında tanımlanmaktadır. Makalelerin tanımlı olan link, konu alanı ve konu bilgisi alınmaktadır. Kalan anahtar değer bilgilerinin ("dil", "baslik", "oz", "anahtar kelime") elde edilmesi için "makale_link_ic" metodu çağrılmaktadır. "makale_link_ic" metodu, "makale_url" ve "makale_icerigi" parametrelerini almaktadır. "makale_link_ic" metodu ilk önce, makale linklerine ulaşım olup olmadığını kontrol eder. Bunun için alınan "makale_url" parametresi ile URL adresinin durum koduna (status_code) bakılır. Durum kodu ile ilgili 3 koşul oluşturulmuştur:

1. Makale linkine erişimin sağlandığı durum
2. Sayfa erişim hatasının alındığı durum
3. Bilinmeyen bir hatanın alındığı durum

Makale linkine erişimin sağlandığı durum

Birinci durumda, durum kodu 200 olup ulaşılmak istenilen makale linkine erişim sağlanmaktadır. Erişim sağlandıktan sonra "BeautifulSoup" ile internet sitesinden makale ile ilgili alınmak istenilen "dil", "baslik", "oz" ve "anahtar kelime" bilgileri alınmaktadır. Makale yazım dilinin bulunduğu kod parçacığı Şekil 5.9'da verilmektedir:



Şekil 5.9. Web kazıma Python kod parçacığı- makale yazım dilinin alınması

Şekil 5.9'da gösterilen kod parçacığı ile HTML kodları ayrıştırılıp, ilgili web sitesinin ögesi incelenerek makale yazım dili bulunmaktadır. Makale ile ilgili "baslik", "oz" ve "anahtar kelime" bilgilerinin bulunduğu kod parçacığı da Şekil 5.10'da gösterilmektedir:

Şekil 5.10. Web kazıma Python kod parçacığı- makale başlık, anahtar kelime ve öz bilgisinin alınması

Şekil 5.10'daki Python kod parçacığını yazabilmek için öncelikle web sitesinin ögesi incelenmiştir. Web sitesinin ögesi incelendikten sonra makalenin başlık, anahtar kelime ve öz bilgisi "BeautifulSoup" ile bulunmaktadır.

Sayfa erişim hatasının alındığı durum

İkinci durumda, durum kodu 404 olup sayfa erişim hatası alınmaktadır. Bu durumda alınmak istenilen makale bilgileri için "404 sayfa hatası"' bilgisi döndürülmektedir.

Bilinmeyen bir hatanın alındığı durum

Üçüncü durum ise durum kodunun 200 ve 404 olmadığı, bilinmeyen farklı bir hatanın alındığı durumdur.

"makale_link_dis" metodu ile "makale_link_ic" metodu çağrılmaktadır. "makale_link_ic" metodu ile makalenin "dil", "baslik", "oz" ve "anahtar kelime" bilgileri alınmaktadır. Her "for" döngüsündeki makalenin bilgileri sözlük veri yapısında tutulmaktadır. "for" döngüsünden çıkmadan önce makale bilgileri, diziye (array) atılarak kayıt altına alınmaktadır.

"WebKazima.py" sınıfında; sırası ile "makale_türleri", "makale_link_dis", "sayfa_sayisi", "makale_link_ic" metotları çalışarak; "Tıp", "Sosyal", "Temel Bilimler" ve "Mühendislik" konu alanlarındaki makalelere ulaşılmaktadır. "WebKazima.py" sınıfında en son olarak, makale dizisinde tutulan bütün makale bilgileri "makaleler.xlsx" dosyasına kaydedilmektedir.

Çalıştırılan "WebKazima.py" Python kodu ile 3 840 tane makale bilgisine ulaşılmıştır. Kaydedilen "makaleler.xlsx" dosyası 3 841 satır ve 8 sütundan oluşmaktadır. İlk sütun sıra numarası olup, diğer sütunlar sırası ile "makale linki", "makale dili", "makale başlığı", "makale konu alanı", "makale konusu", "makale özü" ve "makalenin anahtar kelimeleri" bilgilerini içermektedir.

DergiPark-Akademik internet sitesinden web madenciliği ile alınarak "makaleler.xlsx" dosyasına kaydedilen makale bilgileri; metin işleme ve makine öğrenmesi sınıflandırma işlemlerinin yapıldığı "MetinIsleme.py" Python kodunda kullanılmaktadır.

"WebKazima.py" Python kodu ile toplamda 3 840 farklı makale verisi elde edilmiştir. Elde edilen metin koleksiyonundaki makalelerin yazım dilleri; İngilizce, Türkçe, Fransızca, Arapça ve Rusçadır.

5.4. Makaleler için Metin Ön İşleme Süreçleri

Çalışma kapsamındaki makale verilerinin %99'luk kısmı (3 802 makale) yazım dili Türkçe ve İngilizce olan makalelerden oluşmaktadır. Yazım dili İngilizce olan makaleler (1 897 makale) 1 897 satır ve 6 sütundan oluşan bir veri çerçevesi (dataframe) olarak kaydedilirken; yazım dili Türkçe olan makaleler (1 905 makale) de 1 905 satır ve 6 sütundan oluşan bir veri çerçevesi olarak kaydedilmektedir.

Yazım dili İngilizce ve Türkçe olan makalelerin veri çerçevesi; "makale başlığı", "makale konu alanı", "makale konusu", "makale öz kısmı", "makale anahtar kelimeleri" ve "TD-IDF" sütunundan oluşmaktadır.

"MetinIsleme.py" Python kodu ile yapılan metin ön işleme süreçleri yazım diline göre farklı metotlar kullanılarak uygulanmaktadır. Metin işleme süreçleri, makalelerin öz (abstract) kısmına uygulanmaktadır. "MetinIsleme.py" sınıfındaki İngilizce ve Türkçe makalelerin metin ön işleme süreçleri için yazılan metotların çağrılma sırası Şekil 5.11'de gösterilmektedir:

Şekil 5.11. Metin işleme Python kod parçacığı- metotların çağrılma sırası

Metin işleme ve sınıflandırma işlemleri için ilk önce “siniflandirma_en”, sonrasında ise “siniflandirma_tr” metotları; Şekil 5.11’de gösterildiği sırayla çağrılmaktadır.

- Yazım dili İngilizce olan makaleler için metotlar; 6 defa, aşağıdaki sırayla çağrılmaktadır:
  1) I. siniflandirma_en() → II. makale_ingilizce() → III. WordNetLemmatizer() → 1. naiveBayes_randomForest_konuAlani(en_veri) → IV. konusu_siniflandirma(veri)
  2) I. siniflandirma_en() → II. makale_ingilizce() → III. WordNetLemmatizer() → 2. naiveBayes_randomForest_svm_konusu(en_veri_konusu_df) → IV. konusu_siniflandirma(veri)
  3) I. siniflandirma_en() → II. makale_ingilizce() → III. WordNetLemmatizer() → 3. naiveBayes_randomForest_svm_temelBilimler(en_veri_temel_bilimler_df) → IV. konusu_siniflandirma(veri)
  4) I. siniflandirma_en() → II. makale_ingilizce() → III. WordNetLemmatizer() → 4. naiveBayes_randomForest_svm_muhendislik(en_veri_muhendislik_df) → IV. konusu_siniflandirma(veri)
  5) I. siniflandirma_en() → II. makale_ingilizce() → III. WordNetLemmatizer() → 5. naiveBayes_randomForest_svm_tip(en_veri_tip_df) → IV. konusu_siniflandirma(veri)
  6) I. siniflandirma_en() → II. makale_ingilizce() → III. WordNetLemmatizer() → 6. naiveBayes_randomForest_svm_sosyal(en_veri_sosyal_df) → IV. konusu_siniflandirma(veri)

Sınıflandırma sürecinde, ilk önce “siniflandirma_en” metodu çalışmaktadır. Bu metot denetimli makine öğrenmesi sınıflandırmasının yapıldığı metottur. Metot içerisinde sınıflandırma işlemleri yapılmadan önce, metodun ilk satırındaki “makale_ingilizce” metodu çağrılmaktadır. Bu metot ve içerisinde çağırdığı “WordNetLemmatizer” metodu ile metin ön işleme aşamaları uygulanmaktadır. Metin ön işleme aşamaları uygulandıktan sonra “siniflandirma_en” metoduna tekrar dönülmektedir. Sonrasında, sınıflandırma metotları sırasıyla çağrılmaktadır. Her bir sınıflandırma metodu, denetimli makine öğrenmesi algoritmalarının tanımlandığı “konusu_siniflandirma(veri)” metodunu çağırmaktadır. Bu metot parametre olarak çağrıldığı metodun verilerini almaktadır.

- Yazım dili Türkçe olan makaleler için de metotlar; 6 defa, aşağıdaki sırayla çağrılmaktadır:
  1) I. siniflandirma_tr() → II. makale_turkce() → 1. naiveBayes_randomForest_svm_konuAlani_tr(tr_veri_konu_alani) → III. konusu_siniflandirma(veri)
  2) I. siniflandirma_tr() → II. makale_turkce() → 2. naiveBayes_randomForest_svm_konusu_tr(tr_veri_konusu_df) → III. konusu_siniflandirma(veri)
  3) I. siniflandirma_tr() → II. makale_turkce() → 3. naiveBayes_randomForest_svm_temelBilimler_tr(tr_veri_temel_bilimler_df) → III. konusu_siniflandirma(veri)
  4) I. siniflandirma_tr() → II. makale_turkce() → 4. naiveBayes_randomForest_svm_muhendislik_tr(tr_veri_muhendislik_df) → III. konusu_siniflandirma(veri)
  5) I. siniflandirma_tr() → II. makale_turkce() → 5. naiveBayes_randomForest_svm_tip_tr(tr_veri_tip_df) → III. konusu_siniflandirma(veri)
  6) I. siniflandirma_tr() → II. makale_turkce() → 6. naiveBayes_randomForest_svm_sosyal_tr(tr_veri_sosyal_df) → III. konusu_siniflandirma(veri)

Sınıflandırma sürecinde, ilk önce “siniflandirma_tr” metodu çalışmaktadır. Bu metot denetimli makine öğrenmesi sınıflandırmasının yapıldığı metottur. Metot içerisinde sınıflandırma işlemleri yapılmadan önce, metodun ilk satırındaki “makale_turkce” metodu çağrılmaktadır. Bu metot ile metin ön işleme aşamaları uygulanmaktadır. Metin ön işleme aşamaları uygulandıktan sonra “siniflandirma_tr” metoduna tekrar dönülmektedir. Sonrasında, sınıflandırma metotları sırasıyla çağrılmaktadır. Her bir sınıflandırma metodu, denetimli makine öğrenmesi algoritmalarının tanımlandığı “konusu_siniflandirma(veri)” metodunu çağırmaktadır. Bu metot parametre olarak çağrıldığı metodun verilerini almaktadır.

5.4.1. Belirtkeleme (Tokenization)

Kelime, bir makinenin anlayabileceği ve işleyebileceği minimum birimdir. Bu nedenle herhangi bir metin dizesi, belirtkeleme işleminden geçmeden daha fazla işlenemez. Belirtkeleme, ham dizgiyi anlamlı belirteçlere bölme işlemidir [83].

Metin ön işleme aşamasında, metinler ilk önce cümlelerine ayrılmaktadır. Bu işlem için "MetinIsleme.py" sınıfına NLTK kütüphanesi eklenmiştir. NLTK kütüphanesinin "tokenize" işlemi ile makaleler cümlelerine ayrılıyor. NLTK kütüphanesinin "tokenize" işleminde, İngilizce ve Türkçe dilleri için ayrı ayrı tanımlanan modüller bulunmaktadır. Cümlelerin ayrıştırıldığı "tokenize" işleminde; İngilizce makaleler için "english" parametresi alınmaktadır. Türkçe makaleler için ise "turkish" parametresi alınmaktadır. Cümlelerine ayrılan metin küçük harflere çevrilip, metinden noktalama işaretleri ve sayılar çıkarılmaktadır. Sonrasında durak kelimeleri de çıkarılmaktadır. Son olarak ise metin kelimelerine ayrılarak belirtkeleme işlemi tamamlanmış olmaktadır.

5.4.2. Metni küçük harfe çevirme

Metin ön işleme aşamalarından olan metinleri küçük harfe çevirme işlemleri, yazım dili İngilizce ve Türkçe olan makaleler için uygulanmaktadır. Makaleler "tokenize" ile cümlelere ayrılmaktadır. Sonrasında Python'un "lower()" metodu çağrılarak metin küçük harfe çevrilmektedir.

5.4.3. Metinden noktalama işaretleri ve sayıları çıkarma

Makaleler cümlelerine ayrılıp küçük harfe çevrildikten sonra metin ön işleme aşamalarından olan noktalama işaretleri ve sayıların çıkarılması süreçleri uygulanmaktadır. Metinden sayıların çıkarılması işlemi için "MetinIsleme.py" sınıfına Python modüllerinden olan "Re" modülü eklenmektedir. Modülün "sub" komutu kullanılarak, metinden sayılar çıkarılmaktadır. Noktalama işaretlerinin çıkarılma işlemi için ilgili sınıfa "String" modülü eklenmektedir. Modülün "punctuation" komutu kullanılıp, noktalama işaretleri metinlerden çıkarılmaktadır. Böylece hem sayılar hem de noktalama işaretleri makalelerin öz kısmından çıkarılmaktadır.

5.4.4. Metinden durak kelimelerini çıkarma (remove stopwords)

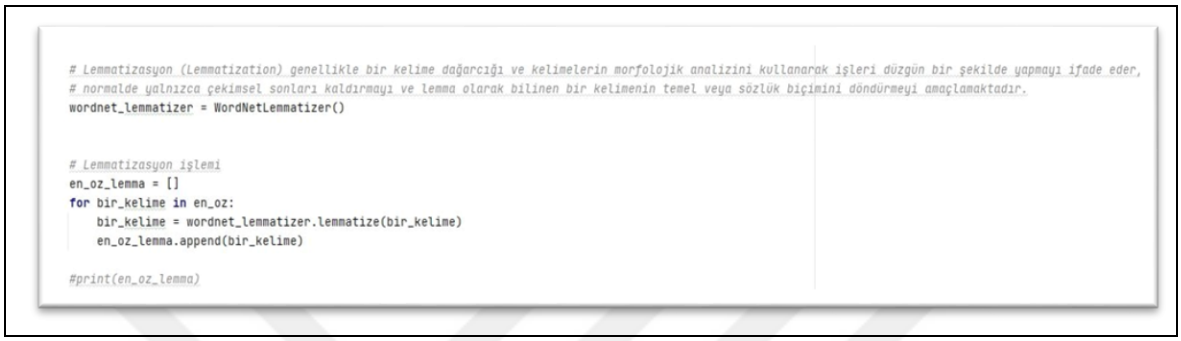
Metinlerde durak kelimeleri olarak bilinen ve çok sık geçtiği için makale sınıflandırma süreçlerini olumsuz etkileyen bazı kelimeler mevcuttur. Durak kelimeleri, NLTK kütüphanesinin külliyatında (corpus) hem İngilizce hem de Türkçe dili için tanımlanmıştır. Külliyatta İngilizce dili için 179 tane, Türkçe dili için 53 tane durak kelimesi tanımlıdır. Bu durak kelimeleri aşağıda verildiği gibidir:

- *İngilizce durak kelimeleri:* [a, above, about, after, again, against, ain, all, am, an, and, are, any, at, as, aren, aren't, because, before, then, below, these, out, you've, herself, be, being, been, once, between, both, by, but, can, yourselves, does, doing, won, do, did, wouldn, that'll, from, i, me, if, which, is, in, needn't, we, didn, ourselves, who, you'd, through until, doesn't, yourself, with, shan't, that, so, t, its, has, she, it, their, doesn, than, weren, too, ll, m, haven't, themselves, there, same, have, should've, needn, hasn, ve, over, yours, whom, this, y, isn't, haven, isn, don't, you'll, on, how, more, such, don, didn't, hadn, during, why, no, each, d, hasn't, down, few, for, will, were, should, wasn, further, mustn, now, theirs, what, couldn, having, he, them, your, him, his, mustn't, had, the, himself, most, her, wouldn't, hers, here, off, where, or, while, into, up, when, you're, myself, other, mightn, shouldn't, weren't, shouldn, only, my, to, some, itself, they, o, re, ma, it's, under, of, nor, hadn't, shan, won't, wasn't, she's, very, s, those, was, our, mightn't, couldn't, not, own, just, ours, you]
- *Türkçe durak kelimeleri:* [en, eğer, hep, nereye, aslında, gibi, mı, ile, tüm, yani, birşey, için, ne, birkaç, mu, siz, sanki, ve, az, ama, bazı, neden, mü, belki, da, hem, ise, kez, kim, ya, biz, defa, acaba, şu, nerede, veya, çünkü, bu, de, hepsi, ki, nasıl, diye, nerde, çok, şey, biri, niye, her, o, daha, niçin, hiç]

Yukarıda verilen durak kelimeleri metinlerden çıkarılmaktadır. Bu işlemle birlikte metnin küçük harfe çevrilmesi, metinden noktalama işaretlerinin ve sayıların çıkarılması, metinden durak kelimelerinin çıkarılması, metnin kelimelere ayrılması işlemleri yapılmış olmaktadır.

5.4.5. Kök çözümleme (Lemmatization)

Lemmatizasyon yöntemi isimleri tekil biçime ve fiilleri sonsuz zamana eşlemeye çalışır [6]. NLTK kütüphanesinin "WordNetLemmatizer()" metodu ile İngilizce makaleler için kök çözümleme işlemi yapılmıştır. İlgili Python kod parçacığı Şekil 5.12'de gösterilmektedir:



```
# Lemmatizasyon (Lemmatization) genellikle bir kelime dağarcığı ve kelimelerin morfolojik analizini kullanarak işleri düzgün bir şekilde yapmayı ifade eder,
# normalde yalnızca çekimsel sonları kaldırmayı ve lemma olarak bilinen bir kelimenin temel veya sözlük biçimini döndürmeyi amaçlamaktadır.
wordnet_lemmatizer = WordNetLemmatizer()


# Lemmatizasyon işlemi
en_oz_lemma = []
for bir_kelime in en_oz:
    bir_kelime = wordnet_lemmatizer.lemmatize(bir_kelime)
    en_oz_lemma.append(bir_kelime)

#print(en_oz_lemma)
```

Şekil 5.12. Metin işleme Python kod parçacığı- kök çözümleme işlemi

Kök çözümleme işlemi için Şekil 5.12'de görüldüğü gibi bir "for" döngüsü yazılmıştır. Döngüde, metin öz kısmının her kelimesine sırasıyla kök çözümleme işlemi yapılmaktadır.

Python NLTK modülü Türkçe dili için lemmatizasyon işlemini desteklemediğinden, Türkçe veri setine lemmatizasyon işlemi yapılmamıştır.

5.4.6. Kelimelerin TF-IDF değerlerini bulma

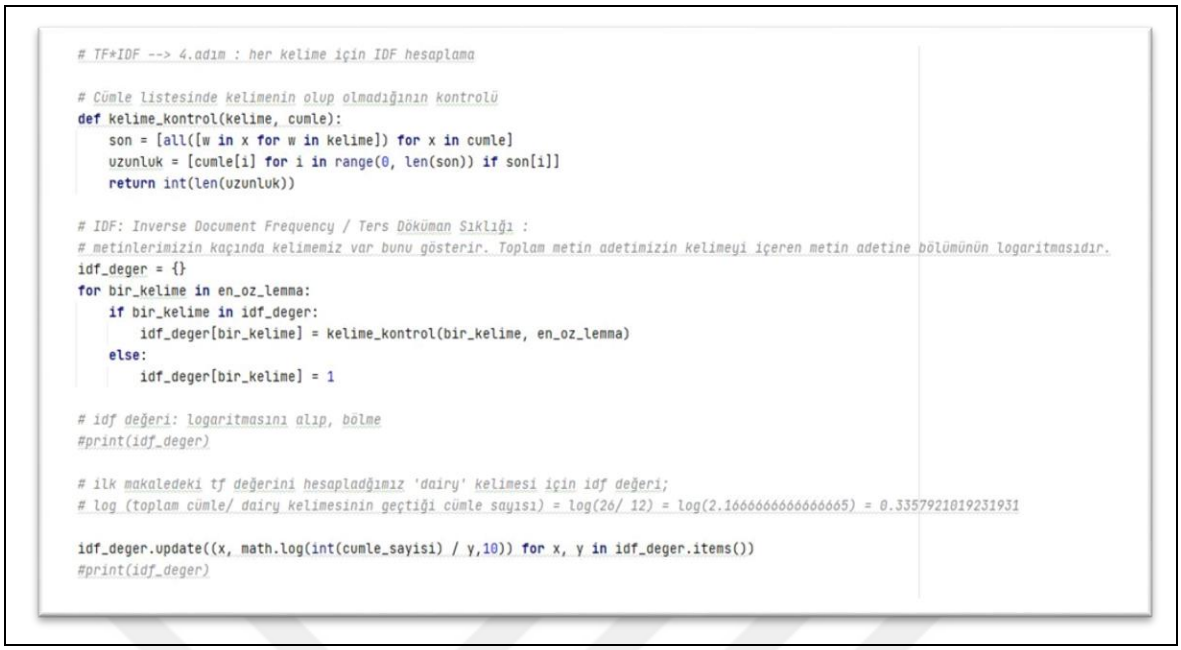
Terim sıklığı, metinde bir terimin görünme sayısının belgedeki toplam kelime sayısına oranı ile bulunmaktadır. Metinlerdeki TF değerinin bulunması ile ilgili kod parçacığı Şekil 5.13'te gösterilmektedir:

```
# TF*IDF --> 3.adım : her kelime için TF değerini hesaplama
# TF: Term Frequency / Terim Sıklığı : seçili kelimenin, metin içinde bulunan toplam kelime sayısına bölümüdür.
tf_deger = {}
for bir_kelime in en_oz_lemma:
    if bir_kelime in tf_deger:
        tf_deger[bir_kelime] += 1
    else:
        tf_deger[bir_kelime] = 1

#print(tf_deger) #her kelimenin metinde geçtiği sayı

# tf değeri : Her bir kelime için toplam kelime_sayısına bölme
tf_deger.update((x, y / int(kelime_sayisi)) for x, y in tf_deger.items())
# print(tf_deger)

# ilk makalenin toplam kelime sayısı 593, 'dairy' kelimesinin makalede toplam geçtiği sayı 12
# ilk makale için 'dairy' kelimesinin tf değeri 12/593 = 0.02023608768971332 dir.
```

Şekil 5.13. Metin işleme Python kod parçacığı- TF değerinin hesaplanması

TF değeri hesaplama sürecinde ilk önce, Şekil 5.13'te görülen "tf_deger" isimli bir sözlük yapısı tanımlanıyor. "If-else" koşulu ile kelimelerin metinde geçtiği toplam sayı bulunuyor. Kelimelerin metinde geçtiği sayı, tanımlanan "tf_deger" sözlük yapısına kaydediliyor. TF değerini hesaplamak için bulunan "tf_deger" değeri, toplam kelime sayısına bölünmektedir. Böylece her kelime için TF değeri bulunuyor. Örnek olarak, ilk makalede geçen "dairy" kelimesine bakılmaktadır. İlk makalenin toplam kelime sayısı 593'tür. "Dairy" kelimesi makalede 12 defa geçmektedir. Makaledeki "dairy" kelimesinin TF değeri 12'nin 593'e bölümü ile 0,020236 olarak bulunmaktadır.

Metinlerdeki IDF değerinin bulunması ile ilgili kod parçacığı Şekil 5.14'te gösterilmektedir:



```
# TF*IDF --> 4.adım : her kelime için IDF hesaplama

# Cümle listesinde kelimenin olup olmadığının kontrolü
def kelime_kontrol(kelime, cumle):
    son = [all([w in x for w in kelime]) for x in cumle]
    uzunluk = [cumle[i] for i in range(0, len(son)) if son[i]]
    return int(len(uzunluk))

# IDF: Inverse Document Frequency / Ters Döküman Sıklığı :
# metinlerimizin kaçında kelimemiz var bunu gösterir. Toplam metin adetimizin kelimeyi içeren metin adetine bölümünün logaritmasıdır.
idf_deger = {}
for bir_kelime in en_oz_lemma:
    if bir_kelime in idf_deger:
        idf_deger[bir_kelime] = kelime_kontrol(bir_kelime, en_oz_lemma)
    else:
        idf_deger[bir_kelime] = 1

# idf değeri: logaritmasını alıp, bölme
#print(idf_deger)

# ilk makaledeki tf değerini hesapladığımız 'dairy' kelimesi için idf değeri;
# log (toplam cümle/ dairy kelimesinin geçtiği cümle sayısı) = log(26/ 12) = log(2.1666666666666665) = 0.3357921019231931

idf_deger.update((x, math.log(int(cumle_sayisi) / y,10)) for x, y in idf_deger.items())
#print(idf_deger)
```

Şekil 5.14. Metin işleme Python kod parçacığı- IDF değerinin hesaplanması

IDF değeri hesaplama sürecinde ilk önce, Şekil 5.14'te görüldüğü gibi cümle listesinde kelimenin olup olmadığının kontrolü yapılmaktadır. Sonrasında ilgili kelimenin metindeki kaç cümlede geçtiği bulunuyor. Toplam cümle sayısının, logaritmik olarak kelimenin kaç cümlede geçtiğinin sayısına bölümü ile IDF değeri bulunmaktadır. TF değerinin hesaplandığı "dairy" kelimesinin IDF değeri ise 2,1667 değerinin logaritmik değeri olan 0,3358'dir.

TF ve IDF değerlerinin çarpımı ile TF-IDF değeri bulunmaktadır. Örnek olarak hesaplanan "dairy" kelimesinin TF-IDF değeri 0,0068'dir.

Makalelerin öz kısmındaki kelimelerin TF-IDF değeri bulunarak, her iki veri seti (Türkçe-İngilizce) için değeri en yüksek olan 100 kelime "tf_idf_kelime" sütununa eklenmektedir. Önem değeri yüksek kelimelerin bulunduğu "tf_idf_kelime" sütunundaki kelimelerle de sınıflandırma süreçleri uygulanmaktadır.

5.4.7. Metin sınıflandırma

Metin sınıflandırma için kullanılan farklı makine öğrenmesi algoritmaları bulunmaktadır. Denetimli makine öğrenmesi sınıflandırma algoritmalarının Lojistik Regresyon algoritması, Naive Bayes algoritması, Karar Ağacı algoritması, K-En Yakın Komşu algoritması, Destek

Vektör Makinesi algoritması, Rastgele Orman algoritması, Stokastik Gradyan İniş algoritması, Çekirdek Yaklaşım algoritması gibi çeşitleri bulunmaktadır. Çalışma kapsamında bu algoritmalardan üç tanesi seçilerek makalelerin öz (abstract) kısımlarının konu alanlarına göre sınıflandırılması yapılmıştır.

Sınıflandırma için Naive Bayes, Rastgele Orman ve Destek Vektör Makinesi algoritmaları seçilmiştir. Bu algoritmaların tercih edilmesinin sebepleri aşağıda verilmektedir:

- Naive Bayes algoritması güçlü bağımsızlık varsayımlarının uygulanmasıyla etkin sonuçlar üreten bir algoritma olması sebebiyle tercih edilmiştir. Naive Bayes algoritması ile hem dengesiz veri kümelerinde hem de veri seti boyutunun küçük ya da büyük olması durumlarında bile etkili sonuçlar elde edilebilmektedir. Algoritmanın yaygın olarak kullanılan iki yöntemi bulunmaktadır: Çok Değişkenli Bernoulli Olay Modeli ve Çok Terimli Model. Çalışma kapsamında; bir belgedeki bir sözcüğün her geçişini, aynı sözcüğün diğer oluşumlarından bağımsız olarak ele alan Çok Terimli (Multinomial) Model kullanılmıştır. Bir belgede geçen kelime sayısını toplayan Çok Terimli Model, bir belgede meydana gelen özellik sayılarını ihmal eden Çok Değişkenli Bernoulli Olay Modelinden daha etkili olduğundan dolayı tercih edilmiştir. Ayrıca, Naive Bayes algoritmasının Türk gazete şirketlerinin köşe yazılarını sınıflandıran Pasin'in [70] çalışmasında metni doğruluk oranına göre sınıflandıran en başarılı algoritma olması da bir diğer tercih sebebidir.
- Rastgele Orman algoritmasının binlerce girdi değişkenini basit ve etkili bir şekilde yönetebilmesi, aşırı uyum problemi ve eksik değer verilerinin kolaylıkla üstesinden gelmesi, hızlı ve esnek olması; algoritmanın tercih edilme sebebidir. Ayrıca, Rastgele Orman algoritmasının yazılım geliştirme taleplerinin önem sırasını tahmin eden Tekin'in [73] çalışmasında metni doğruluk oranına göre sınıflandıran en başarılı algoritma olması da bir diğer tercih sebebidir.
- Destek Vektör Makinesi algoritmasının yüksek boyutlu verilerde etkili olması, basit olması, bellek kullanımı açısından verimli olması, boyutların sayısının örnek sayısından fazla olduğu durumlarda etkili olması, yapılandırılmamış metin verilerinde iyi çalışması, sınıflar arasında net bir ayrım marjı olduğunda çok iyi çalışması gibi sebeplerden dolayı; Destek Vektör Makinesi tercih edilmiştir. Ayrıca, Destek Vektör Makinesi algoritmasının hem inşaat kazası anlatı sınıflandırması yapan Goh ve Ubeynarayana'nın [74] çalışmalarında hem de GitHub'dan toplanan

sorun raporlarını sınıflandıran Fan ve diğerlerinin [75] çalışmalarında metni doğruluk oranına göre sınıflandıran en başarılı algoritma olması da bir diğer tercih sebebidir.

6. BULGULAR VE YORUM

Yazım dili İngilizce ve Türkçe olan makale veri setlerinin öz (abstract) bölümüne, sınıflandırma algoritmalarının daha iyi ve doğru sonuç vermesi için metin ön işleme yapılmıştır. Python kütüphanesine makine öğrenmesi sınıflandırma işlemleri için Scikit-learn modülü eklenmektedir. Sonrasında veri madenciliğinde kullanılan denetimli makine öğrenmesi algoritmalarından Naive Bayes (Multinomial-Çok terimli model), Rastgele Orman ve Destek Vektör Makinesi yöntemleri kullanılarak makaleler konu alanlarına göre sınıflandırılmaktadır.

İngilizce ve Türkçe veri setleri için yapılan sınıflandırma çeşitleri aşağıda verildiği gibidir:

- Konu alanına göre (Tıp, Sosyal, Temel Bilimler, Mühendislik) sınıflandırma
- Konu alanı "Mühendislik" olan makalelerin 4 alt konusuna göre (Mühendislik-Ortak Disiplinler, Mühendislik-Makine, Mühendislik-Elektrik ve Elektronik, Bilgisayar Bilimleri-Bilgi Sistemleri) sınıflandırma
- Konu alanı "Sosyal" olan makalelerin 4 alt konusuna göre (Eğitim-Bilimsel Disiplinler, Din Bilimi, Tarih, İşletme) sınıflandırma
- Konu alanı "Temel Bilimler" olan makalelerin 4 alt konusuna göre (Matematik, Çevre Bilimleri, Deniz ve Tatlı Su Biyolojisi, Fizik-Uygulamalı) sınıflandırma
- Konu alanı "Tıp" olan makalelerin 4 alt konusuna göre (Sağlık Bilimleri ve Hizmetleri, Spor Bilimleri, Genel ve Dahili Tıp, Cerrahi) sınıflandırma
- 4 ana konu alanındaki 16 farklı konuya göre (Sağlık Bilimleri ve Hizmetleri, Spor Bilimleri, Genel ve Dahili Tıp, Cerrahi, Eğitim, Bilimsel Disiplinler, Din Bilimi, Tarih, İşletme, Matematik, Çevre Bilimleri, Deniz ve Tatlı Su Biyolojisi, Fizik, Uygulamalı, Mühendislik-Ortak Disiplinler, Mühendislik-Makine, Mühendislik-Elektrik ve Elektronik, Bilgisayar Bilimleri-Bilgi Sistemleri) sınıflandırma

6.1. Yazım Dili İngilizce Ve Türkçe Olan Makalelerin Sınıflandırılması

Çalışma kapsamında, web madenciliği yöntemi ile DergiPark-Akademik internet sitesinden elde edilen 3 802 makale bulunmaktadır. Bu makalelerin 1 897 tanesinin yazım dili İngilizceyken, 1 905 tanesinin yazım dili Türkçedir.

Yazım dili İngilizce olan makalelerin konu alanına göre dağılımı Tablo 6.1'de verildiği gibidir:

Tablo 6.1. İngilizce makalelerin konu alanı dağılımına göre sayıları

	Makalelerin Sayısı	**Konu Alanına Göre Dağılımı**	
Mühendislik	517	Bilgisayar Bilimleri, Bilgi Sistemleri	88
		Mühendislik, Elektrik ve Elektronik	172
		Mühendislik, Makine	136
		Mühendislik, Ortak Disiplinler	121
Sosyal	213	Din Bilimi	21
		Eğitim, Bilimsel Disiplinler	108
		Tarih	18
		İşletme	66
Temel Bilimler	689	Deniz ve Tatlı Su Biyolojisi	173
		Fizik, Uygulamalı	155
		Matematik	237
		Çevre Bilimleri	124
Tıp	478	Cerrahi	148
		Genel ve Dahili Tıp	121
		Sağlık Bilimleri ve Hizmetleri	139
		Spor Bilimleri	70

Tablo 6.1'de görüldüğü üzere, "Mühendislik" alanında 517, "Sosyal" alanında 213, "Temel Bilimler" alanında 689 ve "Tıp" alanında 478 adet makaleye ulaşılmıştır. "Mühendislik" konu alanının "Mühendislik, Elektrik ve Elektronik" alt alanında 172, "Sosyal" konu alanının "Eğitim, Bilimsel Disiplinler" alt alanında 108, "Temel Bilimler" konu alanının "Matematik" alt alanında 237 ve "Tıp" konu alanının "Cerrahi" " alt alanı ile ilgili makalelerinin sayısı 148 ile ilk sırada yer almaktadır.

Tablo 6.1'de konu alanına göre sayı dağılımı verilen makaleler, 6 farklı kombinasyonla sınıflandırılmaktadır. Her sınıflandırma için metin madenciliğinde tercih edilen Naive Bayes, Rastgele Orman ve Destek Vektör Makinesi algoritmaları kullanılmaktadır. Yazım dili İngilizce olan makaleler için 18 farklı sınıflandırma sonucu elde edilmektedir.

Yazım dili Türkçe olan makalelerin konu alanına göre dağılımı Tablo 6.2'de verildiği gibidir:

Tablo 6.2. Türkçe makalelerin konu alanı dağılımına göre sayıları

	Makalelerin Sayısı	**Konu Alanına Göre Dağılımı**	
Mühendislik	442	Bilgisayar Bilimleri, Bilgi Sistemleri	152
		Mühendislik, Elektrik ve Elektronik	68
		Mühendislik, Makine	104
		Mühendislik, Ortak Disiplinler	118
Sosyal	721	Din Bilimi	203
		Eğitim, Bilimsel Disiplinler	130
		Tarih	215
		İşletme	173
Temel Bilimler	262	Deniz ve Tatlı Su Biyolojisi	67
		Fizik, Uygulamalı	83
		Matematik	3
		Çevre Bilimleri	109
Tıp	480	Cerrahi	91
		Genel ve Dahili Tıp	119
		Sağlık Bilimleri ve Hizmetleri	100
		Spor Bilimleri	170

Tablo 6.2'de görüldüğü üzere, "Mühendislik" alanında 442, "Sosyal" alanında 721, "Temel Bilimler" alanında 262 ve "Tıp" alanında 480 adet makaleye ulaşılmıştır. "Mühendislik" konu alanının "Bilgisayar Bilimleri, Bilgi Sistemleri" alt alanında 152, "Sosyal" konu alanının "Tarih" alt alanında 215, "Temel Bilimler" konu alanının "Çevre Bilimleri" alt alanında 109 ve "Tıp" konu alanının "Genel ve Dahili Tıp" alt alanı ile ilgili makalelerinin sayısı 119 ile ilk sırada yer almaktadır.

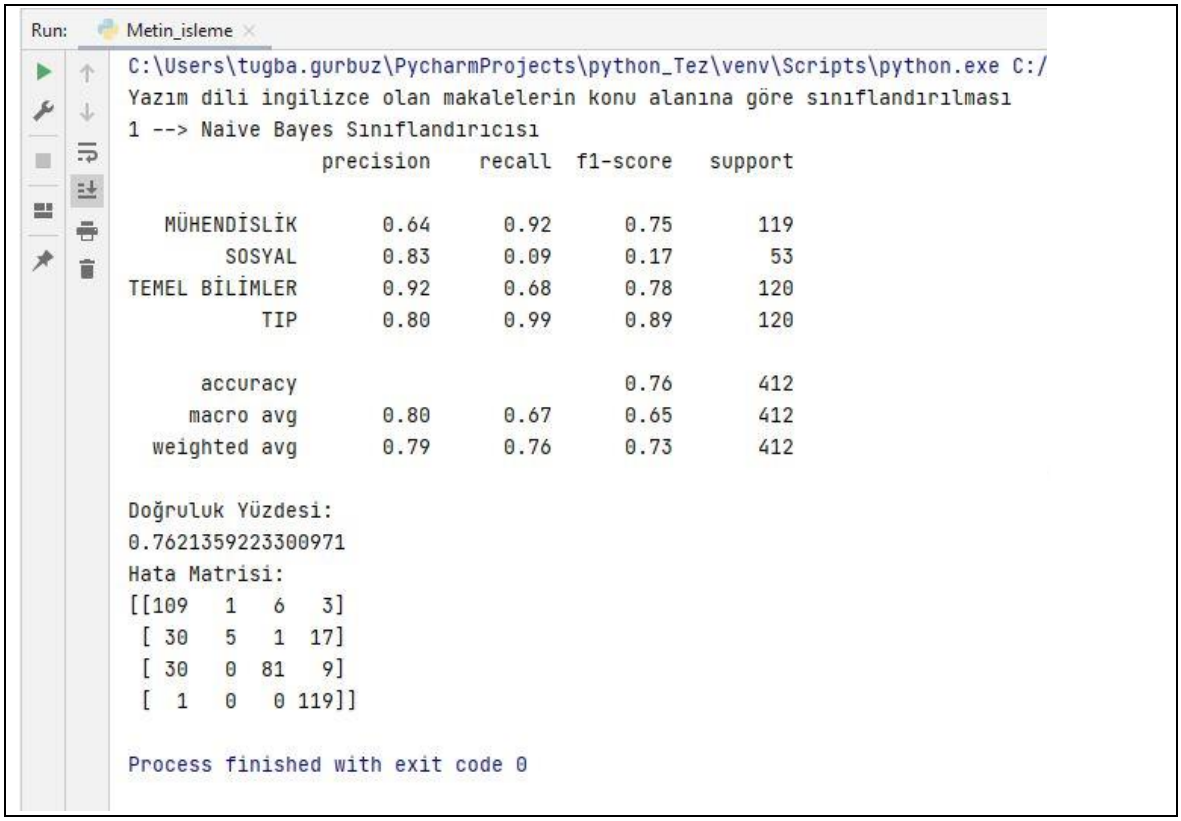
Tablo 6.2'de konu alanına göre sayı dağılımı verilen makaleler, 6 farklı kombinasyonla sınıflandırılmaktadır. Her sınıflandırma için metin madenciliğinde tercih edilen Naive Bayes, Rastgele Orman ve Destek Vektör Makinesi algoritmaları kullanılmaktadır. Yazım dili Türkçe olan makaleler için 18 farklı sınıflandırma sonucu elde edilmektedir.

Yazım dili İngilizce ve Türkçe olan makaleler için toplamda 36 farklı sınıflandırma yapılmaktadır. Sınıflandırma algoritmaları uygulanırken metinleri kapsayan bir külliyat (corpus) oluşturularak, TF-IDF modeli uygulanıyor. TF-IDF modeli uygulandıktan sonra veri setinin %75'i train (eğitim), %25'i ise test (deneme) olmak üzere ikiye ayrılmaktadır. Eğitim seti, modelin eğitildiği veri kümesiyken; deneme seti ise eğitim kümesi ile geliştirilen modelin değerlendirildiği veri kümesidir.

6.2. Makalelerin Konu Alanına Göre Sınıflandırılması

Makaleler “Mühendislik”, “Sosyal”, “Temel Bilimler” ve “Tıp” konu alanlarına göre Naive Bayes, Rastgele Orman ve Destek Vektör Makinesi algoritmaları ile sınıflandırılmaktadır.

İlk olarak, İngilizce ve Türkçe makale veri setleri Naive Bayes algoritması ile sınıflandırılmaktadır. Naive Bayes algoritmasını ile sınıflandırılan İngilizce makalelerin sonuçları Şekil 6.1’de gösterildiği gibidir:

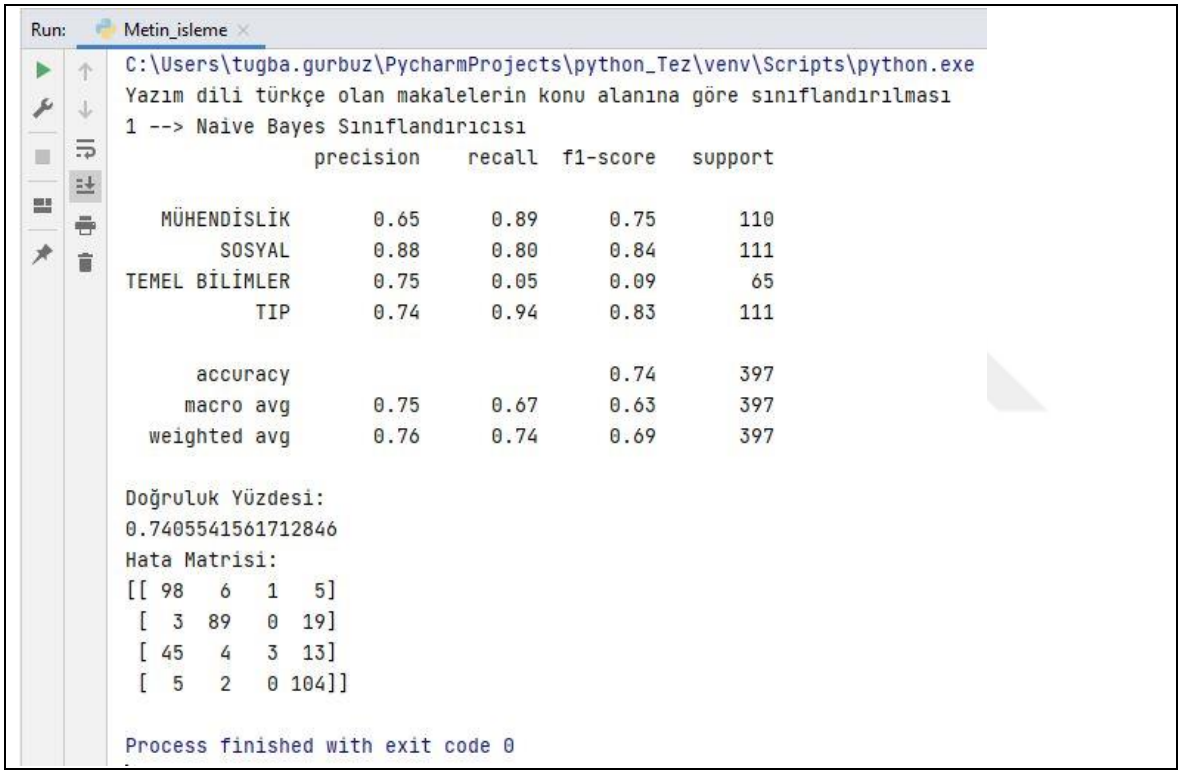


Şekil 6.1. Naive Bayes algoritması – yazım dili İngilizce olan makalelerin konu alanına göre sınıflandırılması

Şekil 6.1’de Python kodu ile yazılan Naive Bayes algoritmasının kod çıktısı gösterilmektedir. Makalelerin konu alanına göre sınıflandırılmasındaki kesinlik (precision) değerlerine göre “Temel Bilimler” konu alanı 0,92 ile en yüksek değer olurken “Mühendislik” konu alanı 0,64 ile en düşük değer olmaktadır. Hatırlama (recall) değerlerine göre ise “Tıp” konu alanı 0,99 ile en yüksek değere ve “Sosyal” konu alanı 0,09 ile en düşük değere sahiptir. Kesinlik (precision) ve hatırlamanın (recall) harmonik ortalaması olan F1 skorunda (F1-score), “Tıp” konu alanı 0,89 ile en yüksek değerdir.

Naive Bayes algoritma sonucunun hata matrisinde görüldüğü gibi "Mühendislik" konu alanındaki 119 makalenin 109 tanesi, "Sosyal" konu alanındaki 53 makalenin 5 tanesi, "Temel Bilimler" konu alanındaki 120 makalenin 81 tanesi ve "Tıp" konu alanındaki 120 makalenin 119 tanesi doğru tahmin edilmiştir. Test verisi olarak alınan 412 makaleden 314 tanesi doğru tahmin edilmiştir. Naive Bayes algoritmasının doğruluk yüzdesi (accuracy) 314'ün 412'ye oranı olan 0,76 değeridir.

Naive Bayes algoritması ile sınıflandırılan Türkçe makalelerin sonuçları Şekil 6.2'de gösterildiği gibidir:

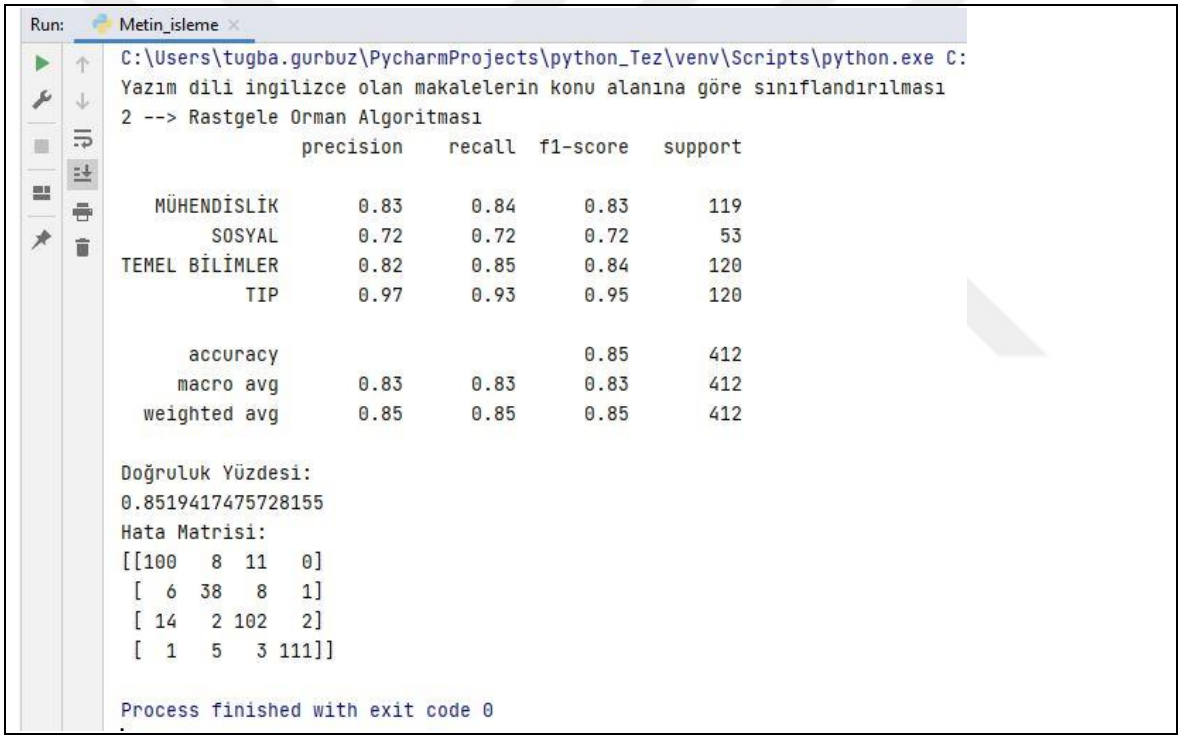


Şekil 6.2. Naive Bayes algoritması – yazım dili Türkçe olan makalelerin konu alanına göre sınıflandırılması

Şekil 6.2'de Python kodu ile yazılan Naive Bayes algoritmasının kod çıktısı gösterilmektedir. Makalelerin konu alanına göre sınıflandırılmasındaki kesinlik (precision) değerlerine göre "Sosyal" konu alanı 0,88 ile en yüksek değer olurken "Mühendislik" konu alanı 0,65 ile en düşük değer olmaktadır. Hatırlama (recall) değerlerine göre ise "Tıp" konu alanı 0,94 ile en yüksek değere ve "Temel Bilimler" konu alanı 0,05 ile en düşük değere sahiptir. Kesinlik (precision) ve hatırlamanın (recall) harmonik ortalaması olan F1 skorunda (F1-score), "Sosyal" konu alanı 0,84 ile en yüksek değerdir.

Naive Bayes algoritma sonucunun hata matrisinde görüldüğü gibi “Mühendislik” konu alanındaki 110 makalenin 98 tanesi, “Sosyal” konu alanındaki 111 makalenin 89 tanesi, “Temel Bilimler” konu alanındaki 65 makalenin 3 tanesi ve “Tıp” konu alanındaki 111 makalenin 104 tanesi doğru tahmin edilmiştir. Test verisi olarak alınan 397 makaleden 294 tanesi doğru tahmin edilmiştir. Naive Bayes algoritmasının doğruluk yüzdesi (accuracy) 294’ün 397’ye oranı olan 0,74 değeridir.

İkinci olarak, İngilizce ve Türkçe makale veri setleri Rastgele Orman algoritması ile sınıflandırılmaktadır. Rastgele Orman algoritması ile sınıflandırılan İngilizce makalelerin sonuçları Şekil 6.3’te gösterildiği gibidir:



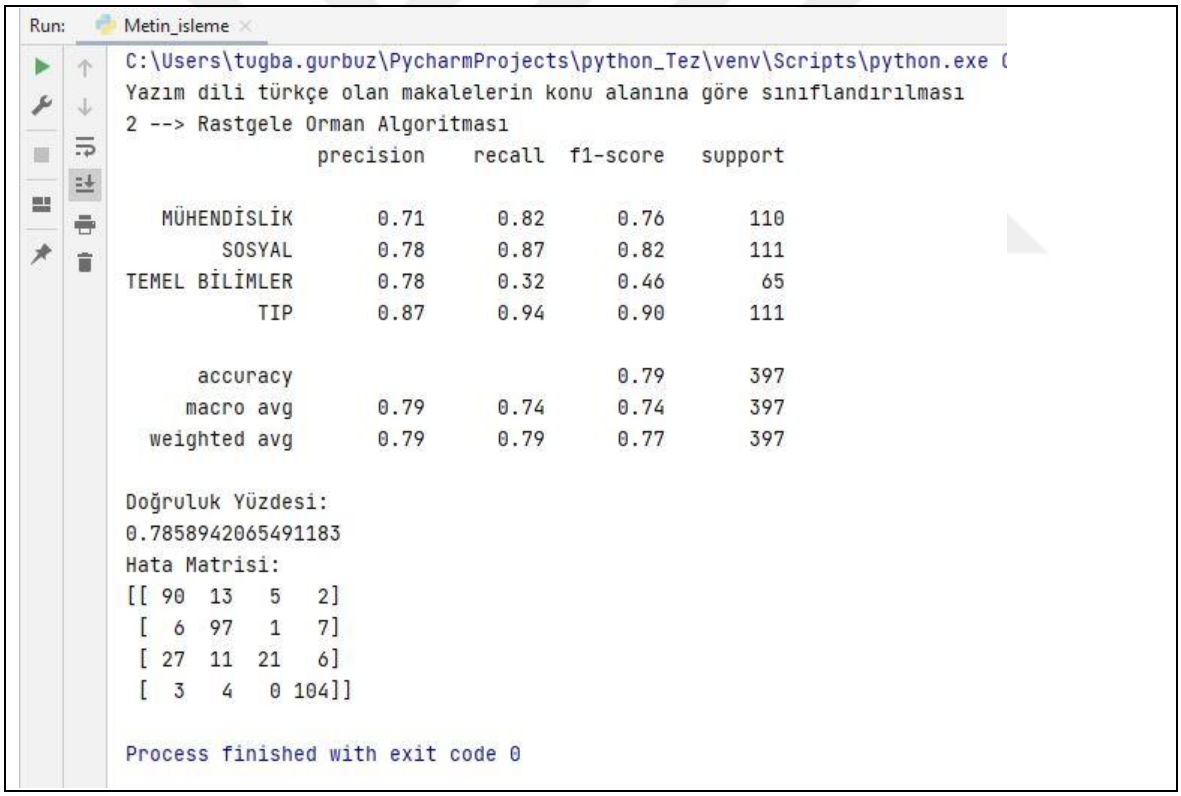
Şekil 6.3. Rastgele Orman algoritması- yazım dili İngilizce olan makalelerin konu alanına göre sınıflandırılması

Şekil 6.3’te Python kodu ile yazılan Rastgele Orman algoritmasının kod çıktısı gösterilmektedir. Makalelerin konu alanına göre sınıflandırılmasındaki kesinlik (precision) değerlerine göre “Tıp” konu alanı 0,97 ile en yüksek değer olurken “Sosyal” konu alanı 0,72 ile en düşük değer olmaktadır. Hatırlama (recall) değerlerine göre ise “Tıp” konu alanı 0,93 ile en yüksek değere ve “Sosyal” konu alanı 0,72 ile en düşük değere sahiptir. Kesinlik

(precision) ve hatırlamanın (recall) harmonik ortalaması olan F1 skorunda (F1-score), "Tıp" konu alanı 0,95 ile en yüksek değerdir.

Rastgele Orman algoritma sonucunun hata matrisinde görüldüğü gibi "Mühendislik" konu alanındaki 119 makalenin 100 tanesi, "Sosyal" konu alanındaki 53 makalenin 38 tanesi, "Temel Bilimler" konu alanındaki 120 makalenin 102 tanesi ve "Tıp" konu alanındaki 120 makalenin 111 tanesi doğru tahmin edilmiştir. Test verisi olarak alınan 412 makaleden 351 tanesi doğru tahmin edilmiştir. Rastgele Orman algoritmasının doğruluk yüzdesi (accuracy) 351'in 412'ye oranı olan 0,85 değeridir.

Rastgele Orman algoritması ile sınıflandırılan Türkçe makalelerin sonuçları Şekil 6.4'te gösterildiği gibidir:



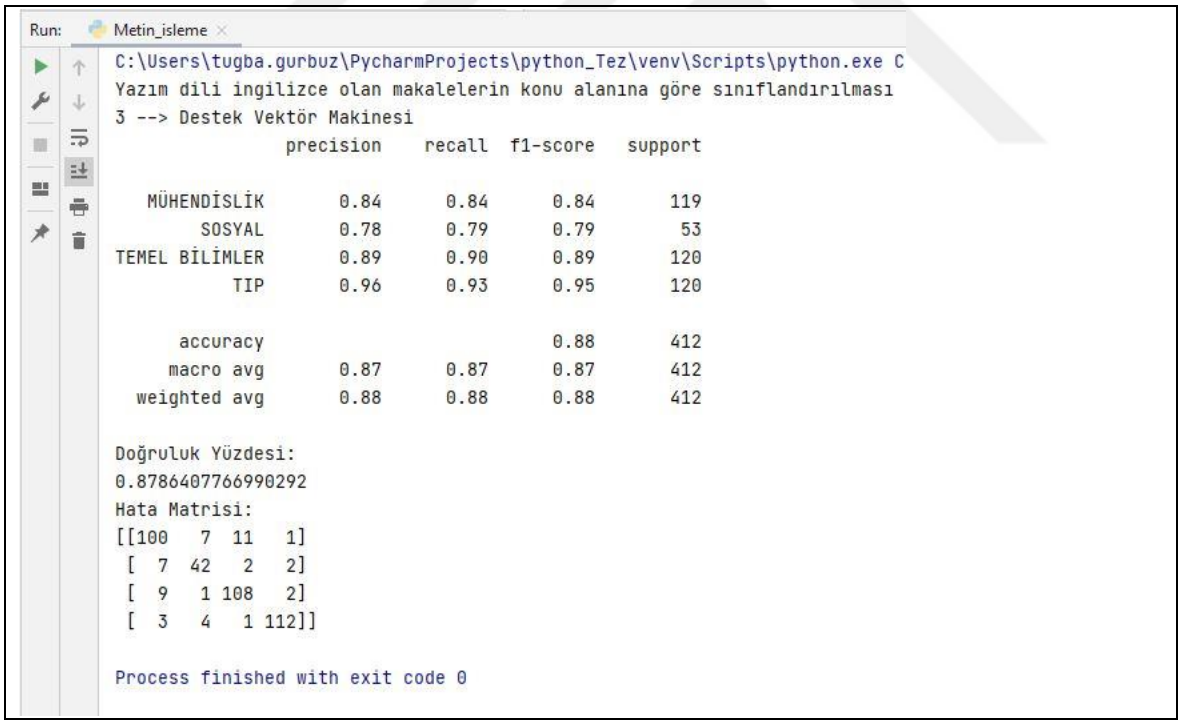
Şekil 6.4. Rastgele Orman algoritması- yazım dili Türkçe olan makalelerin konu alanına göre sınıflandırılması

Şekil 6.4'te Python kodu ile yazılan Rastgele Orman algoritmasının kod çıktısı gösterilmektedir. Makalelerin konu alanına göre sınıflandırılmasındaki kesinlik (precision) değerlerine göre "Tıp" konu alanı 0,87 ile en yüksek değer olurken "Mühendislik" konu alanı 0,71 ile en düşük değer olmaktadır. Hatırlama (recall) değerlerine göre ise "Tıp" konu

alanı 0,94 ile en yüksek değere ve "Temel Bilimler" konu alanı 0,32 ile en düşük değere sahiptir. Kesinlik (precision) ve hatırlamanın (recall) harmonik ortalaması olan F1 skorunda (F1-score), "Tıp" konu alanı 0,90 ile en yüksek değerdir.

Rastgele Orman algoritma sonucunun hata matrisinde görüldüğü gibi "Mühendislik" konu alanındaki 110 makalenin 90 tanesi, "Sosyal" konu alanındaki 111 makalenin 97 tanesi, "Temel Bilimler" konu alanındaki 65 makalenin 21 tanesi ve "Tıp" konu alanındaki 111 makalenin 104 tanesi doğru tahmin edilmiştir. Test verisi olarak alınan 397 makaleden 312 tanesi doğru tahmin edilmiştir. Rastgele Orman algoritmasının doğruluk yüzdesi (accuracy) 312'nin 412'ye oranı olan 0,79 değeridir.

Üçüncü olarak, İngilizce ve Türkçe makale veri setleri Destek Vektör Makinesi algoritması ile sınıflandırılmaktadır. Destek Vektör Makinesi algoritması ile sınıflandırılan İngilizce makalelerin sonuçları Şekil 6.5'te gösterildiği gibidir:



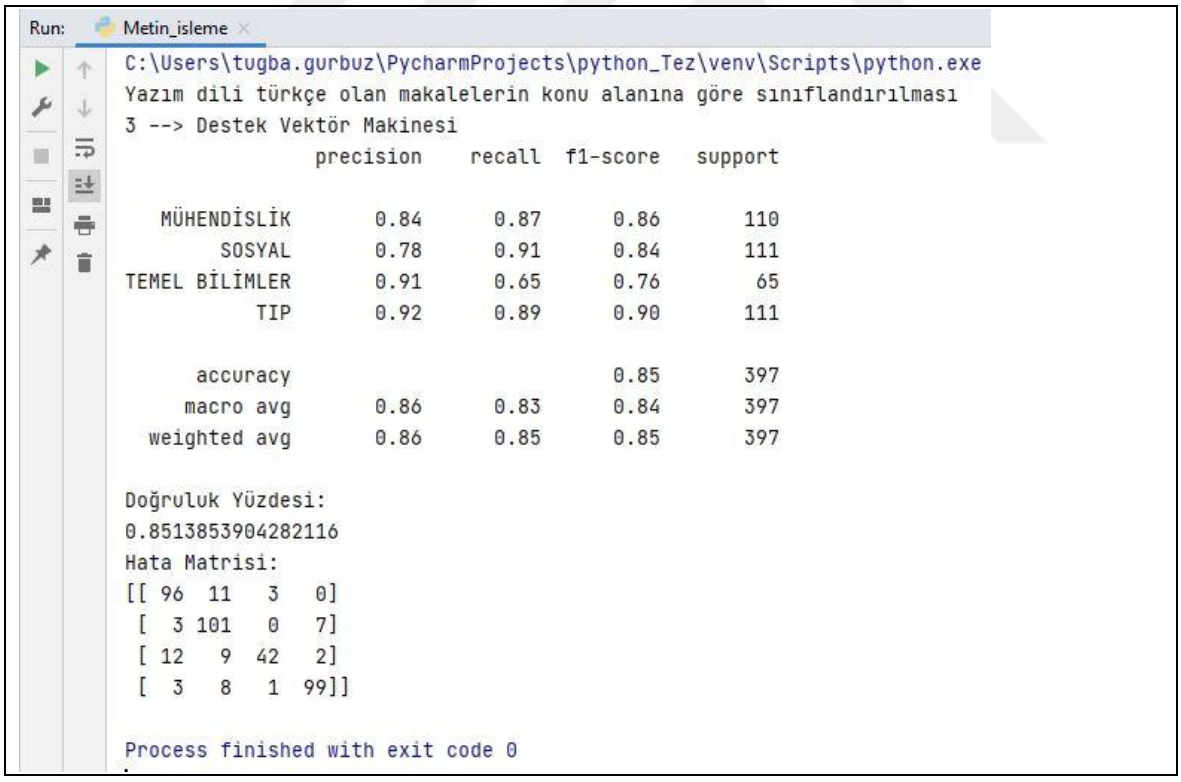
Şekil 6.5. Destek Vektör Makinesi algoritması- yazım dili İngilizce olan makalelerin konu alanına göre sınıflandırılması

Şekil 6.5'te Python kodu ile yazılan Destek Vektör Makinesi algoritmasının kod çıktısı gösterilmektedir. Makalelerin konu alanına göre sınıflandırılmasındaki kesinlik (precision) değerlerine göre "Tıp" konu alanı 0,96 ile en yüksek değer olurken "Sosyal" konu alanı 0,78

ile en düşük değer olmaktadır. Hatırlama (recall) değerlerine göre ise "Tıp" konu alanı 0,93 ile en yüksek değere ve "Sosyal" konu alanı 0,79 ile en düşük değere sahiptir. Kesinlik (precision) ve hatırlamanın (recall) harmonik ortalaması olan F1 skorunda (F1-score), "Tıp" konu alanı 0,95 ile en yüksek değerdir.

Destek Vektör Makinesi algoritma sonucunun hata matrisinde görüldüğü gibi "Mühendislik" konu alanındaki 119 makalenin 100 tanesi, "Sosyal" konu alanındaki 53 makalenin 42 tanesi, "Temel Bilimler" konu alanındaki 120 makalenin 108 tanesi ve "Tıp" konu alanındaki 120 makalenin 112 tanesi doğru tahmin edilmiştir. Test verisi olarak alınan 412 makaleden 362 tanesi doğru tahmin edilmiştir. Destek Vektör Makinesi algoritmasının doğruluk yüzdesi (accuracy) 362'nin 412'ye oranı olan 0,88 değeridir.

Destek Vektör Makinesi algoritması ile sınıflandırılan Türkçe makalelerin sonuçları Şekil 6.6'da gösterildiği gibidir:



Şekil 6.6. Destek Vektör Makinesi algoritması- yazım dili Türkçe olan makalelerin konu alanına göre sınıflandırılması

Şekil 6.6'da Python kodu ile yazılan Destek Vektör Makinesi algoritmasının kod çıktısı gösterilmektedir. Makalelerin konu alanına göre sınıflandırılmasındaki kesinlik (precision)

değerlerine göre “Tıp” konu alanı 0,92 ile en yüksek değer olurken “Sosyal” konu alanı 0,78 ile en düşük değer olmaktadır. Hatırlama (recall) değerlerine göre ise “Sosyal” konu alanı 0,91 ile en yüksek değere ve “Temel Bilimler” konu alanı 0,65 ile en düşük değere sahiptir. Kesinlik (precision) ve hatırlamanın (recall) harmonik ortalaması olan F1 skorunda (F1-score), “Tıp” konu alanı 0,90 ile en yüksek değerdir.

Destek Vektör Makinesi algoritma sonucunun hata matrisinde görüldüğü gibi “Mühendislik” konu alanındaki 110 makalenin 96 tanesi, “Sosyal” konu alanındaki 111 makalenin 101 tanesi, “Temel Bilimler” konu alanındaki 65 makalenin 42 tanesi ve “Tıp” konu alanındaki 111 makalenin 99 tanesi doğru tahmin edilmiştir. Test verisi olarak alınan 397 makaleden 338 tanesi doğru tahmin edilmiştir. Destek Vektör Makinesi algoritmasının doğruluk yüzdesi (accuracy) 397’nin 338’e oranı olan 0,85 değeridir.

Makaleler konu alanına göre Naive Bayes, Rastgele Orman ve Destek Vektör Makinesi algoritmaları ile sınıflandırılmıştır. Algoritmaların doğruluk yüzdesi değerleri Tablo 6.3’te gösterilmektedir:

Tablo 6.3. Konu alanına göre sınıflandırılan makalelerin doğruluk yüzdesi değerleri

	Naive Bayes Algoritması	Rastgele Orman Algoritması	Destek Vektör Makinesi
İngilizce Makale Verisi Doğruluk Yüzdesi (Accuracy)	**0,76**	**0,85**	**0,88**
Türkçe Makale Verisi Doğruluk Yüzdesi (Accuracy)	**0,74**	**0,79**	**0,85**

Tablo 6.3’de görülen sınıflandırma sonuçlarına göre İngilizce makale verilerinin doğruluk oranı, Türkçe makale verilerine göre daha yüksek olmuştur. Her iki veri seti için en başarılı sınıflandırma algoritması Destek Vektör Makinesi çıkmışken, en düşük performans gösteren ise Naive Bayes algoritması olmuştur.

6.3. Konu Alanı "Mühendislik" Olan Makalelerin Sınıflandırılması

Makaleler "Mühendislik" konu alanından; "Mühendislik, Ortak Disiplinler", "Mühendislik, Makine", "Mühendislik, Elektrik ve Elektronik", "Bilgisayar Bilimleri, Bilgi Sistemleri" konularına göre Naive Bayes, Rastgele Orman ve Destek Vektör Makinesi algoritmaları ile sınıflandırılmaktadır.

"Mühendislik" konu alanındaki 4 farklı konuya göre Naive Bayes, Rastgele Orman ve Destek Vektör Makinesi algoritmaları ile sınıflandırılan algoritmaların doğruluk yüzdesi değerleri Tablo 6.4'de gösterilmektedir:

Tablo 6.4. Mühendislik konu alanının 4 farklı konusuna göre sınıflandırılan makalelerin doğruluk yüzdesi değerleri

	Naive Bayes Algoritması	Rastgele Orman Algoritması	Destek Vektör Makinesi
İngilizce Makale Verisi Doğruluk Yüzdesi (Accuracy)	**0,58**	**0,53**	**0,63**
Türkçe Makale Verisi Doğruluk Yüzdesi (Accuracy)	**0,59**	**0,56**	**0,63**

Tablo 6.4'de görülen sınıflandırma sonuçlarına göre İngilizce ve Türkçe makale verileri için doğruluk oranları birbirine yakın değerler çıkmıştır. Her iki veri seti için en başarılı sınıflandırma algoritması Destek Vektör Makinesi çıkmışken, en düşük performans gösteren ise Rastgele Orman algoritması olmuştur.

6.4. Konu Alanı "Sosyal" Olan Makalelerin Sınıflandırılması

Makaleler "Sosyal" konu alanından; "Eğitim, Bilimsel Disiplinler", "Din Bilimi", "Tarih", "İşletme" konularına göre Naive Bayes, Rastgele Orman ve Destek Vektör Makinesi algoritmaları ile sınıflandırılmaktadır.

“Sosyal” konu alanındaki 4 farklı konuya göre Naive Bayes, Rastgele Orman ve Destek Vektör Makinesi algoritmaları ile sınıflandırılan algoritmaların doğruluk yüzdesi değerleri Tablo 6.5’te gösterilmektedir:

Tablo 6.5. Sosyal konu alanının 4 farklı konusuna göre sınıflandırılan makalelerin doğruluk yüzdesi değerleri

	Naive Bayes Algoritması	Rastgele Orman Algoritması	Destek Vektör Makinesi
İngilizce Makale Verisi Doğruluk Yüzdesi (Accuracy)	**0,76**	**0,81**	**0,76**
Türkçe Makale Verisi Doğruluk Yüzdesi (Accuracy)	**0,82**	**0,74**	**0,85**

Tablo 6.5’te görülen sınıflandırma sonuçlarına göre Türkçe makale verilerinin doğruluk oranı, İngilizce makale verilerine göre daha yüksek olmuştur. En başarılı sınıflandırma algoritması İngilizce veri seti için Rastgele Orman algoritması iken, Türkçe veri seti için Destek Vektör Makinesi olmuştur.

6.5. Konu Alanı “Temel Bilimler” Olan Makalelerin Sınıflandırılması

Makaleler “Temel Bilimler” konu alanından; “Matematik”, “Çevre Bilimleri”, “Deniz ve Tatlı Su Biyolojisi”, “Fizik, Uygulamalı” konularına göre Naive Bayes, Rastgele Orman ve Destek Vektör Makinesi algoritmaları ile sınıflandırılmaktadır.

“Temel Bilimler” konu alanındaki 4 farklı konuya göre Naive Bayes, Rastgele Orman ve Destek Vektör Makinesi algoritmaları ile sınıflandırılan algoritmaların doğruluk yüzdesi değerleri Tablo 6.6’da gösterilmektedir:

Tablo 6.6. Temel bilimler konu alanının 4 farklı konusuna göre sınıflandırılan makalelerin doğruluk yüzdesi değerleri

	Naive Bayes Algoritması	Rastgele Orman Algoritması	Destek Vektör Makinesi
İngilizce Makale Verisi Doğruluk Yüzdesi (Accuracy)	**0,84**	**0,84**	**0,84**
Türkçe Makale Verisi Doğruluk Yüzdesi (Accuracy)	**0,75**	**0,80**	**0,84**

Tablo 6.6'da görülen sınıflandırma sonuçlarına göre İngilizce ve Türkçe makale verileri için doğruluk oranları birbirine yakın değerler çıkmıştır. En başarılı sınıflandırma algoritması Türkçe veri seti için Destek Vektör Makinesi olmuşken, İngilizce veri seti için algoritmalar aynı performansı göstermişlerdir.

6.6. Konu Alanı "Tıp" Olan Makalelerin Sınıflandırılması

Makaleler "Tıp" konu alanından; "Sağlık Bilimleri ve Hizmetleri", "Spor Bilimleri", "Genel ve Dahili Tıp", "Cerrahi" konularına göre Naive Bayes, Rastgele Orman ve Destek Vektör Makinesi algoritmaları ile sınıflandırılmaktadır.

"Tıp" konu alanındaki 4 farklı konuya göre Naive Bayes, Rastgele Orman ve Destek Vektör Makinesi algoritmaları ile sınıflandırılan algoritmaların doğruluk yüzdesi değerleri Tablo 6.7'de gösterilmektedir:

Tablo 6.7. Tıp konu alanının 4 farklı konusuna göre sınıflandırılan makalelerin doğruluk yüzdesi değerleri

	Naive Bayes Algoritması	Rastgele Orman Algoritması	Destek Vektör Makinesi
İngilizce Makale Verisi Doğruluk Yüzdesi (Accuracy)	**0,58**	**0,61**	**0,64**
Türkçe Makale Verisi Doğruluk Yüzdesi (Accuracy)	**0,72**	**0,73**	**0,76**

Tablo 6.7'de görülen sınıflandırma sonuçlarına göre Türkçe makale verilerinin doğruluk oranı, İngilizce makale verilerine göre daha yüksek olmuştur. Her iki veri seti için en başarılı sınıflandırma algoritması Destek Vektör Makinesi çıkmışken, en düşük performans gösteren ise Naive Bayes algoritması olmuştur.

6.7. Makalelerin Konusuna Göre Sınıflandırılması

Makaleler 4 ana konu alanındaki 16 farklı konuya göre "Mühendislik, Ortak Disiplinler", "Mühendislik, Makine", "Mühendislik, Elektrik ve Elektronik", "Bilgisayar Bilimleri, Bilgi Sistemleri", "Eğitim, Bilimsel Disiplinler", "Din Bilimi", "Tarih", "İşletme", "Matematik", "Çevre Bilimleri", "Deniz ve Tatlı Su Biyolojisi", "Fizik, Uygulamalı", "Sağlık Bilimleri ve Hizmetleri", "Spor Bilimleri", "Genel ve Dahili Tıp", "Cerrahi" konularına göre Naive Bayes, Rastgele Orman ve Destek Vektör Makinesi algoritmaları ile sınıflandırılmaktadır.

16 farklı konuya göre Naive Bayes, Rastgele Orman ve Destek Vektör Makinesi algoritmaları ile sınıflandırılan algoritmaların doğruluk yüzdesi değerleri Tablo 6.8'de gösterilmektedir:

Tablo 6.8. Konularına göre sınıflandırılan makalelerin doğruluk yüzdesi değerleri

	Naive Bayes Algoritması	Rastgele Orman Algoritması	Destek Vektör Makinesi
İngilizce Makale Verisi Doğruluk Yüzdesi (Accuracy)	**0,64**	**0,58**	**0,67**
Türkçe Makale Verisi Doğruluk Yüzdesi (Accuracy)	**0,66**	**0,60**	**0,67**

Tablo 6.8'de görülen sınıflandırma sonuçlarına göre İngilizce ve Türkçe makale verileri için doğruluk oranları birbirine yakın değerler çıkmıştır. En başarılı sınıflandırma algoritması Destek Vektör Makinesi çıkmışken, en düşük performans gösteren ise Rastgele Orman algoritması olmuştur.

7. TARTIŞMA

Bu tez çalışmasında olduğu gibi Varol'da [84] metin madenciliği ile sınıflandırma çalışması yapmıştır. Çalışmasında 7 şairin her birinin 30 şiirinin bulunduğu 210 adet şiirden oluşan bir eğitim seti kullanmıştır. Bu eğitim veri seti ile kimin tarafından yazıldığı bilinmeyen bir şiirin, metin madenciliği yöntemlerini kullanarak, şairini bulmayı amaçlamıştır. Varol Rastgele Orman algoritması ile %71 en yüksek başarı oranına ulaşırken, Naive Bayes algoritması ile %68 oranında bir başarı elde etmiştir. Bu tez çalışmasındaki yazım dili İngilizce olan makaleler için konu alanlarına göre sınıflandırma işleminde Naive Bayes algoritmasının doğruluk oranı %76 ve Rastgele Orman algoritmasının doğruluk oranı %85'tir. Yazım dili Türkçe olan makaleler için konu alanlarına göre sınıflandırma işleminde ise Naive Bayes algoritmasının doğruluk oranı %74 ve Rastgele Orman algoritmasının doğruluk oranı %79'dur. Bu çalışma kapsamındaki doğruluk oranları, Varol'un çalışmasına göre daha yüksek başarı oranına sahiptir. Bunun en önemli sebebi, bu çalışmada kullanılan veri setinin daha büyük olmasıdır. Bir başka sebebi ise bu çalışmadaki metin ön işleme süreçlerinin performansının daha yüksek olması olabilir.

Bu tez çalışması ile benzer bir sınıflandırma çalışması yapan Pilavcılar [85], K-NN ve Naive Bayes algoritmaları kullanarak metin sınıflandırması yapmıştır. Metin sınıflandırmasında kullanılan veri tabanı, Anadolu Ajansı haber sitesinden 2006 ve 2007 tarihli 2 600 makale kullanılarak oluşturulmuştur. Makaleler ekonomi, politika, sağlık ve spor konu alanlarından seçilerek bu 4 alana göre sınıflandırılmıştır. Pilavcılar'ın çalışmasında Naive Bayes (Multinomial) algoritmasının başarı oranı daha yüksek çıkmıştır. Spor konu alanının tahmin yüzdeleri %96,7 ile en yüksek başarı oranına sahipken, ekonomi konu alanı %82,7 ile en düşük başarı oranına sahip olmuştur. Bu tez çalışmasının makalelerin konu alanına göre sınıflandırma kısmı, Pilavcılar'ın çalışması ile karşılaştırılabilir. Bu çalışmadaki makalelerin 1 897 tanesinin yazım dili İngilizceyken, 1905 tanesinin yazım dili Türkçedir. İngilizce ve Türkçe makalelerin konu alanına göre Naive Bayes (Multinomial) algoritması ile en iyi sınıflandırmaları sırasıyla "Temel Bilimler" alanı %92 ve "Sosyal" alanı ile %88 olmuştur. Pilavcılar'ın çalışmasındaki "Spor" alanının sınıflandırma performansı daha yüksek çıkarken, "Ekonomi" alanının performansının düşük çıkmasının sebebi; "Spor" alanın kendine özgü kelimelere sahip olması olabilir. "Spor" alanının sınıflandırma başarısının, bu tez çalışmasındaki sınıflandırma başarılarından yüksek olmasının sebebi ise bu tez çalışmasındaki veri seti boyutunun daha küçük boyutta olması olabilir.

Gelişmiş istatistik dersindeki öğrencilerin ders notlarını tahmin etmeye çalışan Wu, Hsiao ve Nian [63], çalışmalarında farklı denetimli makine öğrenmesi algoritmaları kullanmışlardır. Kullandıkları algoritmalardan en tutarlı olan algoritmanın Destek Vektör Makinesi olduğunu tespit etmişlerdir. Wu ve diğerlerinin yaptıkları çalışmada veri seti boyutuna göre de 6 farklı sınıflandırma sonucu elde etmişlerdir. Bu veri setlerinin boyutları 500, 1 000, 2 000, 5 000, 10 000 ve 20 000'dir. Veri seti boyutu arttıkça, Destek Vektör Makinesi algoritmasının doğruluk oranının da arttığını göstermişlerdir. Bu tez çalışmasında Destek Vektör Makinesi'nin doğruluk oranları; yazım dili İngilizce olan makalelerde %88 iken, yazım dili Türkçe olan makalelerde %85'dir. Bu çalışmadaki İngilizce makalelerin toplam sayısı 1 897 ve Türkçe makalelerin toplam sayısı 1 905'tir. Wu, Hsiao ve Nian'ın çalışmalarındaki Destek Vektör Makinesi algoritmasının başarı oranı; veri seti boyutu 1 000 olduğunda %81 iken, 2 000 olduğunda %84'tür. Wu ve diğerlerinin çalışmalarındaki veri setinin yazım dili Çince olurken, bu çalışmadaki veri setinin yazım dilleri İngilizce ve Türkçe'dir. Bu tez çalışmasının sonuçlarında görüldüğü gibi veri setindeki yazım dilinin farklı olması, başarı oranlarında çok büyük bir değişiklik yapmamaktadır. Başarı oranlarını asıl etkileyen faktör yazım dilleri değil, metin işleme ve sınıflandırmada kullanılan algoritmaların performanslarıdır.

İnternet sayfalarını internet sayfa kategorisine göre sınıflandıran Şahin [64], çalışmasında hem ikili hem de çoklu sınıflandırma yapmıştır. Şahin çalışmasında terim ağırlıklandırma olarak üç farklı yöntem kullanmış ve bu yöntemlerin başarı oranlarını karşılaştırmıştır. Kullandığı kelime ağırlıklandırma yöntemleri arasında, en başarılı olan yöntemin TF-IDF yöntemi olduğunu göstermiştir. Bu tez çalışmasında da başarılı bir yöntem olan TF-IDF kullanılmıştır. Ayrıca Şahin'in çoklu sınıflandırma da tercih ettiği algoritmalar, bu tezde tercih edilen algoritmalar ile aynıdır. Çalışmasında, Destek Vektör Makinesi'nin TF-IDF ağırlıklandırma yöntemine göre başarım oranının diğer algoritma ve ağırlıklandırma yöntemlerine göre daha yüksek olduğu görülmektedir. Fan, Yu, Yin, T. Wang, H. Wang [75] ise çalışmalarında GitHub'dan toplanan sorun raporlarını sınıflandırmak için Naive Bayes, Lojistik Regresyon, Destek Vektör Makinesi ve Rastgele Orman algoritmalarını kullanmışlardır. Kullanılan sınıflandırma yöntemlerinden %69,7 ile %98,9 arasındaki değerler ile en başarılı algoritmanın Destek Vektör Makinesi olduğunu göstermişlerdir. Metin madenciliği tekniklerini kullanarak inşaat kazası anlatı sınıflandırması yapan Goh ve Ubeynarayana [74], çalışmalarında Destek Vektör Makinesi, Doğrusal Regresyon, Rastgele Orman, K-En Yakın Komşu, Karar Ağacı ve Naive Bayes olmak üzere altı farklı makine

öğrenme algoritması kullanılmıştır. Sınıflandırmada ortalama %73 tahmin oranı ile Destek Vektör Makinesi en iyi performans gösteren algoritma olmuştur. Şahin'in, Fan ve diğerlerinin, Goh ve Ubeynarayana'nın çalışmalarında olduğu gibi bu çalışmada da genellikle en iyi sonuç veren algoritma Destek Vektör Makinesi'dir.

Türk gazete şirketlerinin köşe yazılarını sınıflandıran Pasin [70], çalışmasında sınıflandırma için K-En Yakın Komşu ve Naive Bayes algoritmalarını kullanmıştır. Kullandığı iki algoritma arasından en başarılı olanın Naive Bayes olduğunu belirlemiştir. Pasin çalışmasında farklı sınıflandırmalar yapmıştır. Köşe yazılarını "Ekonomi", "Sağlık", "Magazin", "Siyasi" ve "Spor" alanına göre sınıflandırmıştır. Pilavcılar'ın [85] çalışmasında olduğu gibi, Pasin'in çalışmasında da doğruluk oranının en yüksek çıktığı alan "Spor" alanı olmuştur. Bu tezde de konu alanı "Tıp" olan makaleler; "Cerrahi", "Genel ve Dahili Tıp", "Sağlık Bilimleri ve Hizmetleri" ve "Spor Bilimleri" konularına göre sınıflandırılmıştır. "Spor Bilimleri" alanının doğruluk oranı, Pilavcılar'ın ve Pasin'in çalışmalarında olduğu gibi doğruluk oranı en yüksek çıkan alan olmuştur. Diğer çalışmalarda görüldüğü gibi bu tezde de alana özgü spesifik kelimelerin fazla olmasının doğruluk oranını arttırdığı görülmektedir.

Çam [65] çalışmasında farklı yazarların eserlerine göre yazarlık özelliklerini çıkarırken R programlama dilini kullanmıştır. Her bir yazarın kullandığı kelimeler negatif, nötr ve pozitif olarak üç sınıfa ayrılmıştır. Duygu analizi için R programında yer alan "tidyr", "dplyr", "ggplot2" ve "stringr" paketlerini kullanmıştır. Işık [66] ise çalışmasında denetimli makine öğrenmesi yöntemlerinden Naive Bayes, K-En Yakın Komşu ve Sıralı Minimal Optimizasyon Algoritmalarını kullanarak sosyal medya yorumlarının duygu analizini Weka yazılımını kullanarak yapmıştır. Işık, Çam'ın çalışmasından farklı olarak sosyal medya analizlerini pozitif, negatif ve nötr olarak el yordamı ile sınıflandırmıştır. Weka yazılımını, etiketlenen verileri analiz etmek için kullanmıştır. Başka bir metin madenciliği çalışması yapan Çelikel [71] de çalışmasında hisse senedi fiyat tahminlemesi yapmıştır. Çelikel, veri seti olarak kullandığı Twitter platformundaki tweetleri Python programlama dili kullanarak elde etmiştir. Bu tez çalışmasında da internetten araştırma makale verilerini elde etmek için Python programlama dili kullanılmıştır. Nesne yönelimli, yorumlamalı ve etkileşimli yüksek seviyeli bir programlama dili olan Python; metin madenciliği yapmak isteyen herkesin internetten veri elde etmek için kullanılabileceği hem kullanışlı hem de kolay ve esnek bir şekilde yazılabilen bir programlama dilidir. Ayrıca Çam'ın çalışmasında yapılan duygu

analizi içinde Python paketleri bulunmaktadır. Bu tez çalışmasında da kolay olması, paket çeşitliliği olması gibi sebeplerden dolayı Python programlama dili tercih edilmiştir.

Metin madenciliği, yapısal olmayan metin verilerinin olduğu her alanda uygulanabilmektedir. Çalışmalarda kullanılan metin verileri internetten ya da hazır veri setlerinden elde edilmiştir. Wu, Hsiao ve Nian [63] öğrencilerin ders notlarını tahmin etmeye çalışan bir uygulama yapmak için Facebook mesajlarından oluşan çevrim içi forumdaki metin verilerini kullanmıştır. Işık [66] çalışmasında sosyal medya yorumlarının duygu analizini yapmak için Twitter platformundaki bazı e-ticaret firmalarına, hizmetlerine ve ürünlerine yönelik yapılan yorumlardan oluşan veri kümesini kullanmıştır. Savaş ve Topaloğlu [67] da çalışmalarında istihbarat için örnek bir uygulama geliştirmek için Twitter platformundaki verilerden faydalanmışlardır. Ak [69] pazarlama alanında yayın yapan akademik dergileri değerlendirmek için SCOPUS veri tabanından elde ettiği veri setini kullanmıştır. Fan ve diğerleri [75] sorun sınıflandırma yaklaşımlarını araştırmak için veri seti olarak GitHub'da toplanan sorun raporlarını kullanmışlardır. Pilavcılar [85] ekonomi, politika, sağlık ve spor konu alanlarından seçtiği makaleleri sınıflandırdığı veri setini Anadolu Ajansı haber sitesinden elde etmiştir. Bu çalışmada ise DergiPark-Akademik internet sitesindeki araştırma makaleleri veri seti olarak kullanılmıştır.

8. SONUÇ VE ÖNERİLER

Bu çalışmada DergiPark-Akademik internet sitesindeki araştırma makaleleri metin madenciliği yöntemi ile konularına göre sınıflandırılmıştır. Çalışmada öncelikle metin madenciliği kavramı, metin madenciliği süreçleri ve ilgili literatür incelemesi anlatılmıştır.

Sınıflandırma da kullanılan veri seti "Web Madenciliği" yöntemi ile Python kodu yazılarak elde edilmiştir. Tez için yapılan ön araştırma kapsamında, makalelerin öz (abstract) bölümüne göre yapılan sınıflandırmaların çalışma performanslarının daha yüksek ve iyi olduğu görülmüştür. Tez için yapılan bir diğer ön araştırma kapsamında, metin madenciliği süreçleri için ağırlıklı olarak Naive Bayes, Rastgele Orman ve Destek Vektör Makinesi algoritmalarının kullanıldığı görülmüştür. Tez için yapılan ön araştırmalar sonucu, bu çalışmada makaleler öz (abstract) bölümüne göre Naive Bayes, Rastgele Orman ve Destek Vektör Makinesi algoritmalarına göre sınıflandırılmıştır.

8.1. Sonuçlar ve Yorumlar

Araştırma bulgularına göre, algoritmaların çalışma süreleri İngilizce ve Türkçe veri setleri için birbirine yakın süreler olup; en hızlı çalışan algoritma Naive Bayes iken en yavaş çalışan algoritma Destek Vektör Makinesi olmuştur. Her iki yazım dilindeki algoritma sonuçları birbirine yakın değerler olmakla birlikte genellikle en iyi tahmin yapan algoritma Destek Vektör Makinesidir. Çalışmadaki metin ön işleme süreçlerinin, sınıflandırma algoritmalarının çalışma performansını arttırdığı görülmüştür.

Uygulama ile yapılan metin sınıflandırması doğruluk oranlarının daha iyi sonuç verebilmesi için aşağıdaki yöntemler denenebilir:

- Veri setindeki makale sayıları arttırılarak deneme veri seti daha iyi eğitilebilir.
- Metin ön işleme sürecindeki algoritmalar geliştirilerek uygulamanın çalışma performansı arttırılabilir.
- Metin sınıflandırma için kullanılan algoritmaların parametreleri değiştirilerek en iyi sonuç veren parametre bulunabilir.
- Özellik çıkarımı aşamasındaki kelime veri sayısı, arttırılıp azaltılarak en iyi performans veren veri sayısı bulunabilir.

- Metin ön işleme ve sınıflandırma aşamaları için farklı algoritmalar denenerek performans karşılaştırılması yapılabilir.

8.2. Çalışmanın Zorlukları (Eksikleri)

Algoritmaların çalışma performansını etkileyen unsurlardan bir tanesi veri setinin büyüklüğüdür. Veri setinin büyüklüğü arttıkça eğitim ve deneme veri setleri de doğru oranda artacağı için algoritmaların çalışma performansı da artacaktır. Çalışma da yazım dili Türkçe ve konu alanı "Temel Bilimler" olan makaleler bölümünde buna dair bir örnek bulunmaktadır. "Temel Bilimler" konu alanında "Matematik" konulu sadece üç adet makale yer almaktadır. Üç tanesinin ikisi eğitim, birisi test verisi olarak kullanılmıştır. Eğitim veri setinde yeterli sayıda eğitilecek veri olmadığı için; çalışmada kullanılan algoritmaların üç tanesi de makalenin "Matematik" konusunda olduğunu doğru tahmin edememiştir. Eğer veri seti daha büyük olsaydı, eğitim veri seti iyi eğitilecek ve tahmin oranı daha iyi olacaktı. Daha büyük veri setine bir örnek olarak; yazım dili İngilizce olan ve konu alanına göre sınıflandırılan makalelerden "Tıp" konu alanı verilebilir. "Tıp" konu alanında 478 makale yer almaktadır. Bunlardan 358 tanesi eğitim, kalan 120 tanesi ise test veri seti için kullanılmıştır. Naive Bayes algoritması "Tıp" konu alanındaki makalelerden 119 tanesini doğru tahmin ederek sadece 1 makaleyi yanlış tahmin etmiştir. En başarısız algoritma olan Rastgele Orman algoritması bile 120 makalenin 111 tanesini doğru tahmin etmiştir. "Tıp" konu alanının performansının yüksek çıkmasının bir diğer sebebi ise alanın kendine özgü kelimeleri olmasıdır. Bu iki örnekte görüldüğü gibi veri seti ne kadar büyükse, algoritmaların başarı oranı o kadar yüksektir.

Algoritmaların çalışma performansını etkileyen bir diğer unsur ise metin ön işleme süreçlerinin performansıdır. Metin ön işleme sürecinde kullanılan algoritmalar iyileştirilerek sınıflandırma algoritmalarının doğruluk oranı yükseltilebilir.

Çalışmadaki metin sınıflandırma için kullanılan algoritmaların deneme ve test veri seti oranları, her sınıflandırma için değiştirilerek en yüksek başarıya sahip oran bulunabilir.

8.3. Gelecek Çalışmalara Öneriler

Uygulamanın performansını ve etkinliğini arttırmada veri setinin büyüklüğü önem arz etmektedir. Bu sebeple, benzer çalışmalar için internetten “Web Madenciliği” yöntemi ile elde edilen veri seti çoğaltılabilir ya da daha büyük olan hazır veri setleri kullanılabilir.

Bu tez çalışmasında yazılan Python kodu ile farklı alanlar için metin sınıflandırma işlemleri uygulanabilir. Konusuna göre sınıflandırılmak istenilen bir haber sitesinden "Web Madenciliği" yöntemi ile veriler elde edilebilir. Sonrasında haberlerdeki metinlere, metin ön işleme yapılarak siyaset, spor, sağlık, magazin gibi konulardan hangi konudaki bir haber olduğu bulunabilir. Uygulamadaki "Web Madenciliği" yöntemi kullanılmadan hazır veri seti için de metin sınıflandırma işlemi yapılabilir. Örneğin, farklı yazarlara ait romanların ya da hikâyelerin yer aldığı hazır veri seti olabilir. Bu hazır veri setine çalışmadaki metin madenciliği işlemleri uygulanarak romanlar ya da hikâyeler yazarlarına göre sınıflandırılabilir. Bu ve benzeri uygulamalar için web madenciliği, metin madenciliği ve makine öğrenmesi sınıflandırma süreçlerinde değişiklik yapılarak en iyi performans gösteren süreçler uygulanabilir.

KAYNAKLAR

1. Hirji, Karim K. (2001). Exploring data mining implementation. *Communications of the Association for Computing Machinery (ACM)*, 44(7), 87-93.

2. Chamatkar, A. J., and Butey, P. (2014). Importance of data mining with different types of data applications and challenging areas. *Journal of Engineering Research and Applications*, *4*(5), 38-41.

3. Luan, J. (2004). Data mining applications in higher education. *Statistical Package for the Social Sciences (SPSS) Executive*, 7.

4. Hearst, M. (2003). What is text mining. *Scholar Information Meetings* (*SIMS), UC Berkeley*, *5*.

5. Karadağ, A., and Takçı, H. (2010). Metin madenciliği ile benzer haber tespiti. *Akademik Bilişim*.

6. Hotho, A., A. Nürnberger, and G. Paaß. (2005). A brief survey of text mining. In *Ldv Forum*, 19-62.

7. Seker, S. E. (2015). Metin Madenciliği (Text Mining). *YBS ansiklopedi*, *2*(3), 30-32.

8. İnternet: A.Ş., M. M. Y. (2021). *Metin Madenciliği Nedir?* URL: http://www.metinmadenciligi.com/, Son Erişim Tarihi: 04.05.2021.

9. Singh, S. (2018). Natural language processing for information extraction. arXiv preprint arXiv:1807.02383.

10. Baykal, A., & Coşkun, C. (2009). Web Madenciliği Teknikleri. *Akademik Bilişim, 2009*, 59.

11. Madria, S. K., Bhowmick, S. S., Ng, W.-K., and Lim, E.-P. (1999). Research issues in web data mining. International Conference on Data Warehousing and Knowledge Discovery, 303-312.

12. Bolasco, S., Canzonetti, A., Capo, F. M., Della Ratta-Rinaldi, F., and Singh, B. K. (2005). Understanding text mining: A pragmatic approach. In *Knowledge mining*, Springer, 31-50.

13. Gupta, V., and Lehal, G. S. (2009). A survey of text mining techniques and applications. *Journal of emerging technologies in web intelligence*, *1*(1), 60-76.

14. Silahtaroğlu, G. (2008). *Veri madenciliği*. İstanbul: Papatya Yayınları.

15. Srivastava, S. (2014). Weka: a tool for data preprocessing, classification, ensemble, clustering and association rule mining. *International Journal of Computer Applications*, *88*(10).

16. Aher, S. B., and Lobo, L. (2011). Data mining in educational system using weka. International Conference on Emerging Technology Trends (ICETT), 20-25.

17. Yıldız, M., ve Şeker, Ş. E. (2016). Veri Madenciliği Araçları (Data Mining Tools). *Yönetim Bilişim Sistemleri* (*YBS) ansiklopedi*, *3*(4).

18. Dwivedi, S., Kasliwal, P., and Soni, S. (2016). Comprehensive study of data analytics tools (RapidMiner, Weka, R tool, Knime). 2016 Symposium on Colossal Data Analysis and Networking (CDAN), 1-8.

19. Tekerek, A. (2011). Veri madenciliği süreçleri ve açık kaynak kodlu veri madenciliği araçları. *Akademik Bilişim*, *11*, 2-4.

20. Arslan, İ. (2019). *Python ile Veri Bilimi*. Pusula.

21. Kahya, A. N. (2021). Wikipedia'daki Verilere Metin Madenciliği Yöntemlerinin Uygulanması. *Eskişehir Türk Dünyası Uygulama ve Araştırma Merkezi Bilişim Dergisi*, *2*(1), 11-14.

22. Kannan, S., Gurusamy, V., Vijayarani, S., Ilamathi, J., and Nithya, M. (2014). Preprocessing techniques for text mining. *International Journal of Computer Science and Communication Networks*, *5*(1), 7-16.

23. Kumar, V., and Minz, S. (2014). Feature selection: a literature review. *SmartCR*, *4*(3), 211-229.

24. Çoban, Ö., ve ÖZYER, G. T. (2018). Twitter duygu analizinde terim ağırlıklandırma yönteminin etkisi. *Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi*, *24*(2), 283-291.

25. Salton, G., and Yu, C. T. (1973). On the construction of effective vocabularies for information retrieval. *Association for Computing Machinery (Acm) Sigplan Notices*, *10*(1), 48-60.

26. Liu, C.-z., Sheng, Y.-x., Wei, Z.-q., and Yang, Y.-Q. (2018). Research of text classification based on improved TF-IDF algorithm. 2018 The Institute of Electrical and Electronics Engineers (IEEE) International Conference of Intelligent Robotic and Control Engineering (IRCE), 218-222.

27. Albayrak, M. (2017). Bilimsel araştırmalarda veri madenciliği kullanımı. *International Journal of Social Sciences and Economic Review* (*IJSSER)*, *3*(3), 752-756.

28. Osisanwo, F., Akinsola, J., Awodele, O., Hinmikaiye, J., Olakanmi, O., and Akinjobi, J. (2017). Supervised machine learning algorithms: classification and comparison. *International Journal of Computer Trends and Technology (IJCTT)*, *48*(3), 128-138.

29. Ayodele, T. O. (2010). Types of machine learning algorithms. *New advances in machine learning*, *3*, 19-48.

30. Olaode, A., Naghdy, G., and Todd, C. (2014). Unsupervised classification of images: a review. *International Journal of Image Processing*, *8*(5), 325-342.

31. Kotsiantis, S. B., Zaharakis, I., and Pintelas, P. (2007). Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, *160*(1), 3-24.

32. Hodge, V., and Austin, J. (2004). A survey of outlier detection methodologies. *Artificial intelligence review*, *22*(2), 85-126.

33. Motoda, H., and Liu, H. (2002). Feature selection, extraction and construction. *Communication of Institute of Information and Computing Machinery, Taiwan (IICM) Vol*, *5*(67-72), 2.

34. Markovitch, S., and Rosenstein, D. (2002). Feature generation using general constructor functions. *Machine Learning*, *49*(1), 59-98.

35. Singh, A., Thakur, N., and Sharma, A. (2016). A review of supervised machine learning algorithms. 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), 1310-1315.

36. Feng, J., Xu, H., Mannor, S., and Yan, S. (2014). Robust logistic regression and classification. *Advances in neural information processing systems*, *27*, 253-261.

37. Peng, C.-Y. J., Lee, K. L., and Ingersoll, G. M. (2002). An introduction to logistic regression analysis and reporting. *The journal of educational research*, 96(1), 3-14.

38. Qiang, G. (2010). An effective algorithm for improving the performance of Naïve Bayes for text classification. 2010 Second international conference on computer research and development, 699-701.

39. Schneider, K.-M. (2005). Techniques for Improving the Performance of Naive Bayes for Text Classification. In A. Gelbukh, *Computational Linguistics and Intelligent Text Processing* Berlin, Heidelberg, 682-693.

40. Rennie, J. D., Shih, L., Teevan, J., and Karger, D. R. (2003). Tackling the poor assumptions of naive bayes text classifiers. Proceedings of the 20th international conference on machine learning (ICML-03), 616-623.

41. Jin, C., De-Lin, L., and Fen-Xiang, M. (2009). An improved ID3 decision tree algorithm. 2009 4th International Conference on Computer Science and Education, 127-130.

42. Charbuty, B., and Abdulazeez, A. (2021). Classification based on decision tree algorithm for machine learning. *Journal of Applied Science and Technology Trends*, *2*(01), 20-28.

43. Singh, A., Damir, B., Deep, K., and Ganju, A. (2015). Calibration of nearest neighbors model for avalanche forecasting. *Cold Regions Science and Technology*, *109*, 33-42.

44. Gök, M. (2015). An ensemble of k-nearest neighbours algorithm for detection of Parkinson's disease. *International Journal of Systems Science*, *46*(6), 1108-1112.

45. Noble, W. S. (2006). What is a support vector machine? *Nature biotechnology*, 24(12), 1565-1567.

46. Kalantari, B. (2019). An algorithmic separating hyperplane theorem and its applications. *Discrete Applied Mathematics*, *256*, 59-82.

47. Agrawal, A. (2019). Separation Theorems.

48. Patle, A., and Chouhan, D. S. (2013). SVM kernel functions for classification. 2013 International Conference on Advances in Technology and Engineering (ICATE), 1-9.

49. Abdulkareem, N. M., and Abdulazeez, A. M. (2021). Machine learning classification based on Radom Forest Algorithm: A review. *International Journal of Science and Business*, 5(2), 128-142.

50. Kuzborskij, I., and Lampert, C. (2018). Data-dependent stability of stochastic gradient descent. International Conference on Machine Learning, 2815-2824.

51. Bonnabel, S. (2013). Stochastic gradient descent on Riemannian manifolds. *The Institute of Electrical and Electronics Engineers (IEEE) Transactions on Automatic Control*, *58*(9), 2217-2229.

52. Liu, Y., Jiang, S., and Liao, S. (2014). Efficient approximation of cross-validation for kernel methods using Bouligand influence function. International Conference on Machine Learning, 324-332.

53. Ding, L., Liao, S., Liu, Y., Liu, L., Zhu, F., Yao, Y., Gao, X. (2020). Approximate kernel selection via matrix approximation. *The Institute of Electrical and Electronics Engineers (IEEE) transactions on neural networks and learning systems*, *31*(11), 4881-4891.

54. Maulud, D., and Abdulazeez, A. M. (2020). A Review on Linear Regression Comprehensive in Machine Learning. *Journal of Applied Science and Technology Trends*, *1*(4), 140-147.

55. Seber, G. A., and Lee, A. J. (2012). *Linear regression analysis* (Vol. 329). John Wiley and Sons.

56. Acharya, M. S., Armaan, A., and Antony, A. S. (2019). A comparison of regression models for prediction of graduate admissions. 2019 International Conference on Computational Intelligence in Data Science (ICCIDS), 1-5.

57. Chen, Y., He, P., Chen, W., and Zhao, F. (2018). A polynomial regression method based on Trans-dimensional Markov Chain Monte Carlo. 2018 The Institute of Electrical and Electronics Engineers (IEEE) 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), 1781-1786.

58. Seliya, N., Khoshgoftaar, T. M., and Van Hulse, J. (2009). A study on the relationships of classifier performance metrics. 2009 21st The Institute of Electrical and Electronics Engineers (IEEE international conference on tools with artificial intelligence, 59-66.

59. Visa, S., Ramsay, B., Ralescu, A. L., and Van Der Knaap, E. (2011). Confusion matrix-based feature selection. *Midwest Artificial Intelligence and Cognitive Science (MAICS)*, *710*, 120-127.

60. Raschka, S. (2014). An overview of general performance metrics of binary classifier systems. arXiv preprint arXiv:1410.5330.

61. Caruana, R., and Niculescu-Mizil, A. (2004). Data mining in metric space: an empirical analysis of supervised learning performance criteria. Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, 69-78.

62. Ding, S. (2009). Feature selection based F-score and ACO algorithm in support vector machine. 2009 Second International Symposium on Knowledge Acquisition and Modeling, 19-23.

63. Wu, J.-Y., Hsiao, Y.-C., and Nian, M.-W. (2020). Using supervised machine learning on large-scale online forums to classify course-related Facebook messages in predicting learning achievement within the personal learning environment. *Interactive Learning Environments*, *28*(1), 65-80.

64. Şahin, İ. (2019). *Metin Madenciliği ve Makine Öğrenmesi ile İnternet Sayfalarının Sınıflandırılm*ası, Yüksek Lisans Tezi, Hacettepe Üniversitesi Fen Bilimleri Enstitüsü, Ankara.

65. Çam, E.E. (2019). *Metin madenciliğinde kategorik değişkenler için benzetim katsayılarının kullanılması üzerine bir çalışma*, Yüksek Lisans Tezi, Dokuz Eylül Üniversitesi Sosyal Bilimler Enstitüsü, İzmir.

66. Işık, N. (2019). *Metin madenciliği yöntemleri ile e-ticaret markalarına yönelik sosyal medya yorumlarının analizi*, Yüksek lisans tezi, Marmara Üniversitesi Sosyal Bilimler Enstitüsü, İstanbul.

67. Savaş, S., and Topaloğlu, N. (2019). Data analysis through social media according to the classified crime. *Turkish Journal of Electrical Engineering & Computer Sciences*, 27(1), 407-420.

68. Korkut, Y. (2019). *Afet yönetiminde kritik başarı faktörlerini belirlemek için analitik hiyerarşi prosesi ve metin madenciliği destekli bir model önerisi*. Yüksek lisans tezi, Sakarya Üniversitesi İşletme Enstitüsü, Sakarya.

69. Ak, D. (2019). *Pazarlama alanındaki uluslararası akademik dergilerin metin madenciliği yöntemi ile değerlendirilmesi*. Yüksek lisans tezi, Sakarya Üniversitesi İşletme Enstitüsü, Sakarya.

70. Pasin, E. (2018). *Investigation of text mining methods on turkish text.* Yüksek lisans tezi, Dokuz Eylül Universitesi Fen Bilimleri Enstitüsü, İzmir.

71. Çelikel, A. D. (2018). *Stock value prediction using machine learning and text mining*. Yüksek lisans tezi, Kadir Has Üniversitesi, Fen ve Mühendislik Enstitüsü, İstanbul.

72. Bilgin, A.O. (2018). *Metin madenciliği yöntemleri ile yazar tanima: divan edebiyati örneği*. Yüksek lisans tezi, Karadeniz Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Trabzon.

73. Tekin, M.C. (2018). *Yazilim geliştirme taleplerinin metin madenciliği ile siniflandirilmasi ve önceliklendirilmesi*. Yüksek lisans tezi, Maltepe Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul.

74. Goh, Y. M., and Ubeynarayana, C. (2017). Construction accident narrative classification: An evaluation of text mining techniques. *Accident Analysis and Prevention*, *108*, 122-130.

75. Fan, Q., Yu, Y., Yin, G., Wang, T., and Wang, H. (2017). Where is the road for issue reports classification based on text mining? 2017 Association for Computing Machinery (ACM)/ The Institute of Electrical and Electronics Engineers (IEEE) International Symposium on Empirical Software Engineering and Measurement (ESEM), 121-130.

76. Dang, S., and Ahmad, P. H. (2014). Text mining: Techniques and its application. *International Journal of Engineering and Technology Innovations*, *1*(4), 22-25.

77. Yildirim, P., Çeken, K., and Saka, O. (2011). Knowledge discovery for the treatment of bacteria affecting the liver. Turkish Journal of Medical Sciences, 41(5), 867-875.

78. İnternet: Ronquillo, A. *Python's Requests Library* URL: https://realpython.com/python-requests/#getting-started-with-requests, Son Erişim Tarihi: 14.09.2021.

79. İnternet: Richardson, L., *Beautiful soup documentation.* URL: https://www.crummy.com/software/BeautifulSoup/bs4/doc/, Son Erişim Tarihi: 10.01.2021.

80. McKinney, W. (2011). pandas: a foundational Python library for data analysis and statistics. *Python for high performance and scientific computing*, *14*(9), 1-9.

81. Bird, S. (2006). NLTK: the natural language toolkit. Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions, 69-72.

82. Bisong, E. (2019). Introduction to Scikit-learn. In *Building Machine Learning and Deep Learning Models on Google Cloud Platform*, 215-229.

83. Hardeniya, N., Perkins, J., Chopra, D., Joshi, N., and Mathur, I. (2016). *Natural language processing: python and NLTK*. Packt Publishing Ltd.

84. Varol, M. (2011). *Metin madenciliği yöntemlerini kullanarak Türkçe dökümanlarda tür ve yazar tanıma* Yüksek lisans tezi, Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü, Isparta.

85. Pilavcılar, İ.F. (2007) *Metin madenciliği ile metin sınıflandırma*. Yüksek lisans tezi, Yıldız Teknik Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul.

EKLER

EK-1. “WebKazima.py” Python Kodu

```
import time
import requests
import pandas as pd
from bs4 import BeautifulSoup as bs


makale = [] # makale listesi (array)
makale_icerigi = {} # makale özellikleri (dictionary)


# seçilen konu ile ilgili toplam sayfa sayısını döndüren method, ilgili alan
kodu girdi olarak alıyor
def sayfa_sayisi (alan_kodu) :
    url =
"https://dergipark.org.tr/tr/search?q=&section=articles&aggs[articleType.id][0]
=54&aggs[subjects.id][0]=" + str(alan_kodu)

    r = requests.get(url)
    # r.status_code = 200 sayfa erişimi sağlandı

    soup = bs(r.content, "lxml") # html sitesini, lxml ile parçalandı

    # sayfa sayılarının olduğu sondan bir önceki li yi bulma
    sayfa_sayısı_li = len(soup.find_all("ul", attrs={
        "class": "kt-pagination__links mx-auto"})[0].find_all("li")) - 2

    # sondan bir önceki li içerisine giderek toplam sayfa sayısını bulma
    sayfa_sayısı = soup.find_all("ul", attrs={
        "class": "kt-pagination__links mx-auto"})[0].find_all("li")[
        sayfa_sayısı_li].text
    return sayfa_sayısı

# alınan makale linklerinin içeriğindeki verileri alıyor
def makale_link_ic(url, icerik):
    makale_icerigi = içerik

    r_makale_ic = requests.get(url,timeout=3) # makale_link_dıs metodundan url
adresi alındı ve 3 sn delay eklendi
    soup_makale_ic = bs(r_makale_ic.content, "lxml")  # makale html sitesini
lxml ile parçalandı

    #print(r_makale_ic.status_code)  #status_code = 200 sayfa erişimi sağlandı

    # sayfa erişim sağlandıysa
    if r_makale_ic.status_code == 200:

        makale_dil = soup_makale_ic.find("table", attrs={
            "class": "record_properties table"}).find("td").text.strip()
        makale_icerigi['dil'] = makale_dil

        # makale dili sadece türkçe ya da ingilizceyse if bloğuna
girecektir(tek dil ise)
        if soup_makale_ic.find("div", attrs={"id": "title-tr"}) == None:
            makale_baslik = soup_makale_ic.find("h2", attrs={
                "class": "title"}).text.strip()

            makale_icerigi['baslik'] = makale_baslik

            makale_anahtar_kelime = soup_makale_ic.find("div", attrs={
                "id": "articleKeywords"}).find("div").text.strip()

            makale_icerigi['anahtar kelime'] = makale_anahtar_kelime

            makale_oz = soup_makale_ic.find("div", attrs={"id":
"articleAbstract"}).find("div").text.strip()
```

EK-1.(devam) “WebKazima.py” Python Kodu

```
            makale_icerigi['oz'] = makale_oz
        # makalenin hem tr hem eng olarak iki dil seçeneği varsa elif bloğuna
girecektir
        elif soup_makale_ic.find("div", attrs={
            "id": "title-tr"}) != None:
            makale_baslik = soup_makale_ic.find("div", attrs={

                "id": "title-tr"}).text.strip()
            makale_icerigi['baslik'] = makale_baslik

            # anahtar kelime div in içinde detaylı yazılmışsa
            if soup_makale_ic.find("div", attrs={"id":
"articleKeywords"}).find("div",
attrs={"id": "keywords-tr"} != None):

                # makale iki dilde de yazılmış olup birincil dili türkçe ise
                if makale_dil == 'tr':

                    try:
                        makale_anahtar_kelime = soup_makale_ic.find("div",
attrs={
                            "id": "articleKeywords"}).find("div", attrs={
                            "id": "keywords-tr"}).text
                        makale_icerigi['anahtar kelime'] =
makale_anahtar_kelime

                        makale_oz = soup_makale_ic.find("div", attrs={
                            "id": "abstract-tab-content"}).find("div", attrs={
                            "class": "tr-tab-value tab-pane active"}).text
                        makale_icerigi["oz"] = makale_oz

                    except Exception:

                        makale_icerigi['anahtar kelime'] = ("birinci dili
türkçe olan makale anahtar kelimeleri bulunamadı")

                        makale_icerigi["oz"] = ("birinci dili türkçe olan
makale özü bulunamadı")

                 # makale iki dilde de yazılmış olup birincil dili ingilizce
ise
                elif makale_dil == 'en':
                    makale_anahtar_kelime = soup_makale_ic.find("div", attrs={
                        "id": "articleKeywords"}).find("div", attrs={
                        "class": "en-tab-value tab-pane active"}).text
                    makale_icerigi['anahtar kelime'] = makale_anahtar_kelime

                    makale_oz = soup_makale_ic.find("div", attrs={
                        "id": "abstract-tab-content"}).find("div", attrs={
                        "class":"en-tab-value tab-pane active"}).text
                    makale_icerigi["oz"] = makale_oz


            #anahtar kelime ilk div altında ise

            elif soup_makale_ic.find("div", attrs={
                "id": "articleKeywords"}).find("div", attrs={
                "id": "keywords-tr"} == None) :

                makale_anahtar_kelime = soup_makale_ic.find("div", attrs={
                    "id": "articleKeywords"}).find("div").text.strip()

                makale_icerigi['anahtar kelime'] = makale_anahtar_kelime

                makale_oz = soup_makale_ic.find("div", attrs={
                    "id": "articleAbstract"}).find("div").text.strip()
```

EK-1.(devam) “WebKazima.py” Python Kodu

```
                makale_icerigi['oz'] = makale_oz
            #anahtar kelime bulunmadıysa
            else:
                makale_icerigi['anahtar kelime'] = 'Anahtar kelime bulunamadı'
                makale_icerigi['oz'] = 'Makale özü bulunamadı'

    # sayfa erişim hatası alındıysa
    elif r_makale_ic.status_code == 404:
        makale_hata = '404 sayfa hatası'
        makale_icerigi['dil'] = makale_hata
        makale_icerigi['baslik'] = makale_hata

    # html kodu ilk iki sayfa tipinee göre farklıysa, makale ismini
bulamayacaktır
    else:
        print("Makale linkine ulaşıldı, fakat ismi bulunamadı")

    return

# ilgili alan kodu ile ilgili sayfalardaki makale linklerini alınıyor,
sayfa_sayisi() ve  makale_link_ic() metodunu çağırıyor
def makale_link_dis(makale_konu_alani, makale_konusu, makale_kodu):

    toplam_sayfa = sayfa_sayisi(makale_kodu)  # seçilen alandaki sayfa sayısı

    for sayfa_numarası in range(1, int(toplam_sayfa) + 1):  # arama sonucundaki
sayfa sayısı kadar döngü devam edecek
        url_sayfa = requests.get("https://dergipark.org.tr/tr/search/"
                                 + str(sayfa_numarası)
 + "?q=&section=articles&aggs[articleType.id][0]=54&aggs[subjects.id][0]="
                                 + str(makale_kodu),
                                 timeout=3)  ##Error 429 Slow down! Too many
requests" hatası almamak için 3 saniye delay eklendi

        #print(url_sayfa.status_code) #r.status_code = 200 sayfa erişimi
sağlandı
        soup_sayfa = bs(url_sayfa.content, "lxml")
        makale_body = soup_sayfa.find_all("div", attrs={
            "class": "card-body"})  # sayfadaki bütün makalelerin body bilgisi
alındı

        for makale_detay in makale_body:
            makale_icerigi = {
                'link': '',
                'dil': '',
                'baslik': '',
                'konu alani' : '',
                'konusu': '',
                'oz': '',
                'anahtar kelime': ''
            }

            makale_url = makale_detay.a.get("href")  # makale linklerini
makale_url değişkenine atandı
            makale_icerigi['link'] = makale_url
            makale_icerigi['konu alani'] = makale_konu_alani
            makale_icerigi['konusu'] = makale_konusu

            makale_link_ic(makale_url,makale_icerigi) # makale linklerinin
içine linkleri alınan method ile girildi

            makale.append(makale_icerigi)
            time.sleep(3)  # "Error 429 Slow down! Too many requests" hatası
almamak için 3 saniye delay eklendi
    return
```

EK-1.(devam) “WebKazima.py” Python Kodu

```
# makale türlerini tanımlıyor ve makale_link_dis() metodunu çağırıyor
def makale_türleri ():

    # Python collection veri tiplerinden olan dictionary yani sözlük veri
yapısı key ve value şeklinde verilerin saklanabileceği bir veri yapısıdır.

    # tıp konu alanı içeriği
    tip = {
        'Konu Alanı': 'TIP',
        'Konusu': ['Sağlık Bilimleri ve Hizmetleri', 'Spor Bilimleri', 'Genel
ve Dahili Tıp', 'Cerrahi'],
        'Kodu': [258, 260, 234, 227]
    }

    # sosyal konu alanı içeriği
    sosyal = {
        'Konu Alanı': 'SOSYAL',
        'Konusu': ['Eğitim, Bilimsel Disiplinler', 'Din Bilimi', 'Tarih',
'İşletme'],
        'Kodu': [312, 303, 357, 365]
    }

    # temel bilimler konu alanı içeriği
    temel_bilimler = {
        'Konu Alanı': 'TEMEL BİLİMLER',
        'Konusu': ['Matematik', 'Çevre Bilimleri', 'Deniz ve Tatlı Su
Biyolojisi', 'Fizik, Uygulamalı'],
        'Kodu': [194, 212, 170, 181]
    }

    # mühendislik konu alanı içeriği
    mühendislik = {
        'Konu Alanı': 'MÜHENDİSLİK',
        'Konusu': ['Mühendislik, Ortak Disiplinler', 'Mühendislik, Makine',
'Mühendislik, Elektrik ve Elektronik',
                   'Bilgisayar Bilimleri, Bilgi Sistemleri'],
        'Kodu': [145, 143, 139, 110]
    }

   # makale_konu_alani: tıp-----------------------------------------------

    # makale_konusu: Sağlık Bilimleri ve Hizmetleri,  makale_kodu: 258
    makale_link_dis(tip['Konu Alanı'],tip['Konusu'][0],tip['Kodu'][0] )

    # makale_konusu: Spor Bilimleri, makale_kodu: 260
    makale_link_dis(tip['Konu Alanı'],tip['Konusu'][1],tip['Kodu'][1] )

    # makale_konusu: Genel ve Dahili Tıp, makale_kodu: 234
    makale_link_dis(tip['Konu Alanı'],tip['Konusu'][2],tip['Kodu'][2] )

    # makale_konusu: Cerrahi, makale_kodu: 227
    makale_link_dis(tip['Konu Alanı'],tip['Konusu'][3],tip['Kodu'][3] )
    #-------------------------------------------------------------------------

    # makale_konu_alani: sosyal--------------------------------------------
    # makale_konusu: Eğitim, Bilimsel Disiplinler, makale_kodu: 312
    makale_link_dis(sosyal['Konu Alanı'], sosyal['Konusu'][0],
sosyal['Kodu'][0])

    # makale_konusu: Din Bilimi, makale_kodu: 303
    makale_link_dis(sosyal['Konu Alanı'], sosyal['Konusu'][1],
sosyal['Kodu'][1])
```

EK-1.(devam) “WebKazima.py” Python Kodu

```
    # makale_konusu: Tarih, makale_kodu: 357
    makale_link_dis(sosyal['Konu Alanı'], sosyal['Konusu'][2],
sosyal['Kodu'][2])

    # makale_konusu: İşletme, makale_kodu: 365
    makale_link_dis(sosyal['Konu Alanı'], sosyal['Konusu'][3],
sosyal['Kodu'][3])
    #-------------------------------------------------------------------------
--
    # makale_konu_alani: temel_bilimler-------------------------------------
------------
    # makale_konusu: Matematik ,  makale_kodu: 194
    makale_link_dis(temel_bilimler['Konu Alanı'], temel_bilimler['Konusu'][0],
temel_bilimler['Kodu'][0])

    # makale_konusu: Çevre Bilimleri ,  makale_kodu: 212
    makale_link_dis(temel_bilimler['Konu Alanı'], temel_bilimler['Konusu'][1],
temel_bilimler['Kodu'][1])

    # makale_konusu: Deniz ve Tatlı Su Biyolojisi ,  makale_kodu: 170
    makale_link_dis(temel_bilimler['Konu Alanı'], temel_bilimler['Konusu'][2],
temel_bilimler['Kodu'][2])

    # makale_konusu: Fizik, Uygulamalı ,  makale_kodu: 181
    makale_link_dis(temel_bilimler['Konu Alanı'], temel_bilimler['Konusu'][3],
temel_bilimler['Kodu'][3])
    #-------------------------------------------------------------------------
------------


    # makale_konu_alani: mühendislik----------------------------------------
------------
    # makale_konusu: Mühendislik, Ortak Disiplinler ,  makale_kodu: 145
    makale_link_dis(mühendislik['Konu Alanı'], mühendislik['Konusu'][0],
mühendislik['Kodu'][0])

    # makale_konusu: Mühendislik, Makine ,  makale_kodu: 143
    makale_link_dis(mühendislik['Konu Alanı'], mühendislik['Konusu'][1],
mühendislik['Kodu'][1])

    # makale_konusu: Mühendislik, Elektrik ve Elektronik ,  makale_kodu: 139
    makale_link_dis(mühendislik['Konu Alanı'], mühendislik['Konusu'][2],
mühendislik['Kodu'][2])

    # makale_konusu: Bilgisayar Bilimleri, Bilgi Sistemleri ,  makale_kodu: 110
    makale_link_dis(mühendislik['Konu Alanı'], mühendislik['Konusu'][3],
mühendislik['Kodu'][3])
    #-------------------------------------------------------------------------
------------


# makale_türleri --> makale_link_dis --> sayfa_sayisi --> makale_link_ic
makale_türleri()
makale_df = pd.DataFrame(makale) # makale dataframe i oluşturuldu


makale_df.to_excel('makaleler.xlsx', sheet_name='makale bilgileri')
makale_df.to_csv('makaleler.csv')
```

EK-2. “MetinIsleme.py” Python Kodu

```
import pandas as pd
from nltk.corpus import stopwords
from nltk import tokenize,word_tokenize
from nltk.stem import WordNetLemmatizer
import re
import string
import math
from operator import itemgetter
from sklearn import naive_bayes, svm
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score,
confusion_matrix, recall_score, precision_score

import warnings
#warnings.filterwarnings('always')  # always, error, modüle, ignore, default,
once
warnings.filterwarnings('ignore') # tahmin edilmeyen etiket numaralarıyla
ilgilenilmediği için uyarıları yoksayılıyor


makale = pd.read_excel("makaleler_1.xlsx")
makale_df = pd.DataFrame(makale)

makale_en = []  # ingilizce makale listesi (array)
makale_icerigi_en = {}  # ingilizce makale özellikleri (dictionary)

makale_tr = []  # türkçe makale listesi (array)
makale_icerigi_tr = {}  # türkçe makale özellikleri (dictionary)


# sınıflandırma algoritmaları --> naiveBayes_randomForest_svm_*** metodlarının
çağırdığı sınıflandırma metodu
def konusu_siniflandirma(veri):
    # metinlerin hepsini kapsayan bir corpus oluşturuluyor
    corpus = []
    for i in range(len(veri)):
        corpus.append(veri.iloc[i, 0])

    # TF-IDF modeli Corpus ile fit ediliyor
    vectorizer = TfidfVectorizer()
    vectorizer.fit(corpus)

    # corpus transform ediliyor
    X = vectorizer.transform(corpus).toarray()
    # konu alanına göre sınıflandırma yapılıyor
    y = veri.konusu

    # Stratify=y'de train ve testteki y oranının aynı olmasını sağlıyor
    # Veri setinin %75 i train, %25 i test olmak üzere 2'ye ayrılıyor
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=39, stratify=y)

    # 1. sınıflandırıcı ----------------------------------------------------
------------
    # Naive Bayes sınıflandırıcısı denetimli bir sınıflandırma algoritmasıdır.
    # Naive Bayes sınıflandırıcısı (Supervised classification algorithm) ile
veri seti sınıflandırılıyor
    print("1 --> Naive Bayes Sınıflandırıcısı")
    Naive = naive_bayes.MultinomialNB()
    Naive.fit(X_train, y_train)
    y_pred_nb = Naive.predict(X_test)
```

EK-2. (devam) “MetinIsleme.py” Python Kodu

```
    print(classification_report(y_test, y_pred_nb))
    # Doğruluk - Accuracy yüzdesini gösteriyor
    print("Doğruluk Yüzdesi: ")
    print(accuracy_score(y_test, y_pred_nb))

    # Duyarlılık - Recall
    #print(recall_score(y_test, y_pred_nb, average=None))
    # Kesinlik - Precision
    #print(precision_score(y_test, y_pred_nb , average=None))

   # hata matrisi tablo olarak gösteriliyor
    print("Hata Matrisi: ")
    print(confusion_matrix(y_test, y_pred_nb))

    # 2. sınıflandırıcı----------------------------------------------------------
------------
    # Rastgele Orman algoritması denetimli bir sınıflandırma algoritmasıdır.
    # Random Forest algoritması(Supervised classification algorithm) ile veri
seti sınıflandırılıyor
    print("2 --> Rastgele Orman Algoritması")
    rf = RandomForestClassifier(class_weight='balanced_subsample')
    rf.fit(X_train, y_train)
    y_pred_rf = rf.predict(X_test)

    # precision : tahmin yüzdesini ifade ediyor
    # f1-score : tıp alanını diğer alanlardan ayıran belirgin özellik olduğunu
gösteriyor
    print(classification_report(y_test, y_pred_rf))
    # doğruluk yüzdesini gösteriyor
    print("Doğruluk Yüzdesi: ")
    print(accuracy_score(y_test, y_pred_rf))
    # hata matrisi tablo olarak gösteriliyor
    print("Hata Matrisi: ")
    print(confusion_matrix(y_test, y_pred_rf))

    # Random forest algoritmasında hangi değişkenin modellemede önemli olduğunu
gösteriyor
    #rf_onemli = pd.DataFrame({'features': vectorizer.get_feature_names(),
'importance': rf.feature_importances_})
    #rf_onemli = rf_onemli.sort_values(by='importance', ascending=False)
    # değişken isimleri ve önem skorları
    #print(rf_onemli.head(50))

    # 3. sınıflandırıcı----------------------------------------------------------
------------

    # Destek Vektör Makinesi denetimli bir sınıflandırma algoritmasıdır.
    # Support Vector Machine (Supervised classification algorithm) ile veri
seti sınıflandırılıyor

    print("3 --> Destek Vektör Makinesi")
    SVM = svm.SVC(C=1.0, kernel='linear', degree=3, gamma='auto')
    SVM.fit(X_train, y_train)
    y_pred_svm = SVM.predict(X_test)

    print(classification_report(y_test, y_pred_svm))
    # doğruluk yüzdesini gösteriyor
    print("Doğruluk Yüzdesi: ")
    print(accuracy_score(y_test, y_pred_svm))
    # hata matrisi tablo olarak gösteriliyor
    print("Hata Matrisi: ")
    print(confusion_matrix(y_test, y_pred_svm))

    # Denetimli sınıflandırma algoritmalarının genel doğruluk yüzdeleri ;
    # Support Vector Machine > Random Forest > Naive Bayes
```

EK-2. (devam) “MetinIsleme.py” Python Kodu

```
    return
# 1 --> 'konu alanına' göre ingilizce makaleleri sınıflandırma
def naiveBayes_randomForest_svm_konuAlani(veri):
    # konu alanına göre makale sayıları eşit olmadığı için veriler dengeli hale
getiriliyor

    # print("Data'nın Bölünmeden Önceki Boyutu", len(veri)) #1897
    data_temel_bilimler = veri[veri.konu_alani == 'TEMEL BİLİMLER']
    data_muhendislik = veri[veri.konu_alani == 'MÜHENDİSLİK']
    data_tip = veri[veri.konu_alani == 'TIP']
    data_sosyal = veri[veri.konu_alani == 'SOSYAL']
    veri = pd.concat([data_temel_bilimler.sample(len(data_tip)),
                      data_muhendislik.sample(len(data_tip)), data_tip,
                      data_sosyal])
    # print("Data'nın Yeni Boyutu: ", len(veri)) #1647

    # metinlerin hepsini kapsayan bir corpus oluşturuluyor
    corpus = []
    for i in range(len(veri)):
        corpus.append(veri.iloc[i, 0])

    # Corpus ile TF-IDF modeli fit ediliyor

   vectorizer = TfidfVectorizer()
    vectorizer.fit(corpus)

    # corpus transform ediliyor
    X = vectorizer.transform(corpus).toarray()
    # konu alanına göre sınıflandırma yapılıyor
    y = veri.konu_alani

    # Stratify=y'de train ve testteki y oranının aynı olmasını sağlıyor
    # Veri setinin %75 i train, %25 i test olmak üzere 2'ye ayrılıyor
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=39, stratify=y)


    print("Yazım dili ingilizce olan makalelerin konu alanına göre
sınıflandırılması")

    # 1. sınıflandırıcı ----------------------------------------------------
------------
    # Naive Bayes sınıflandırıcısı denetimli bir sınıflandırma algoritmasıdır.
    # Naive Bayes sınıflandırıcısı (Supervised classification algorithm) ile
veri seti sınıflandırılıyor

    print("1 --> Naive Bayes Sınıflandırıcısı")
    Naive = naive_bayes.MultinomialNB()
    Naive.fit(X_train, y_train)
    y_pred_nb = Naive.predict(X_test)


    print(classification_report(y_test, y_pred_nb))
    # doğruluk yüzdesini gösteriyor
    print("Doğruluk Yüzdesi: ")
    print(accuracy_score(y_test, y_pred_nb))
    # hata matrisi tablo olarak gösteriliyor
    print("Hata Matrisi: ")
    print(confusion_matrix(y_test, y_pred_nb))

    # Duyarlılık - Recall

    #print(recall_score(y_test, y_pred_nb, average=None))

    # Kesinlik - Precision
```

EK-2. (devam) “MetinIsleme.py” Python Kodu

```
    #print(precision_score(y_test, y_pred_nb, average=None))

    # 2. sınıflandırıcı----------------------------------------------------------
------------
    # Rastgele Orman algoritması denetimli bir sınıflandırma algoritmasıdır.
    # Random Forest algoritması(Supervised classification algorithm) ile veri
seti sınıflandırılıyor
    print("2 --> Rastgele Orman Algoritması")
    rf = RandomForestClassifier(class_weight='balanced_subsample')
    rf.fit(X_train, y_train)
    y_pred_rf = rf.predict(X_test)

   # precision : tahmin yüzdesini ifade ediyor
    # f1-score : tıp alanını diğer alanlardan ayıran belirgin özellik olduğunu
gösteriyor
    print(classification_report(y_test, y_pred_rf))
    # doğruluk yüzdesini gösteriyor
    print("Doğruluk Yüzdesi: ")
    print(accuracy_score(y_test, y_pred_rf))
    # hata matrisi tablo olarak gösteriliyor
    print("Hata Matrisi: ")
    print(confusion_matrix(y_test, y_pred_rf))

    # Random forest algoritmasında hangi değişkenin modellemede önemli olduğunu
gösteriyor
    #rf_onemli = pd.DataFrame({'features': vectorizer.get_feature_names(),
'importance': rf.feature_importances_})
    #rf_onemli = rf_onemli.sort_values(by='importance', ascending=False)
    # değişken isimleri ve önem skorları
    #print(rf_onemli.head(50))

    # 3. sınıflandırıcı----------------------------------------------------------
------------
    # Destek Vektör Makinesi denetimli bir sınıflandırma algoritmasıdır.
    # Support Vector Machine (Supervised classification algorithm) ile veri
seti sınıflandırılıyor
    print("3 --> Destek Vektör Makinesi")
    SVM = svm.SVC(C=1.0, kernel='linear', degree=3, gamma='auto')
    SVM.fit(X_train, y_train)
    y_pred_svm = SVM.predict(X_test)

    print(classification_report(y_test, y_pred_svm))
    # doğruluk yüzdesini gösteriyor
    print("Doğruluk Yüzdesi: ")
    print(accuracy_score(y_test, y_pred_svm))
    # hata matrisi tablo olarak gösteriliyor
    print("Hata Matrisi: ")
    print(confusion_matrix(y_test, y_pred_svm))

    # Denetimli sınıflandırma algoritmalarının doğruluk yüzdeleri ;  Support
Vector Machine > Random Forest > Naive Bayes
    return

# 2 --> 'konusuna' göre ingilizce makaleleri sınıflandırma metodu
def naiveBayes_randomForest_svm_konusu(veri):
    # konu alanına göre makale sayıları eşit olmadığı için veriler dengeli hale
getiriliyor
    #print("Data'nın Bölünmeden Önceki Boyutu", len(veri)) #1897
    data_matematik = veri[veri.konusu == 'Matematik']
    data_deniz_ve_tatli_su_biyolojisi = veri[veri.konusu == 'Deniz ve Tatlı Su
Biyolojisi']
    data_muhendislik_elektrik_elektronik = veri[veri.konusu == 'Mühendislik,
Elektrik ve Elektronik']
    data_fizik_uygulamali = veri[veri.konusu == 'Fizik, Uygulamalı']
    data_cerrahi = veri[veri.konusu == 'Cerrahi']
```

EK-2. (devam) “MetinIsleme.py” Python Kodu

```
    data_saglik_bilimleri_ve_hizmetleri = veri[veri.konusu == 'Sağlık Bilimleri
ve Hizmetleri']
    data_muhendislik_makine = veri[veri.konusu == 'Mühendislik, Makine']
    data_cevre_bilimleri = veri[veri.konusu == 'Çevre Bilimleri']
    data_muhendislik_ortak_disiplinler = veri[veri.konusu == 'Mühendislik,
Ortak Disiplinler']
    data_genel_ve_dahili_tip = veri[veri.konusu == 'Genel ve Dahili Tıp']
    data_egitim_bilimsel_disiplinler = veri[veri.konusu == 'Eğitim, Bilimsel
Disiplinler']
    data_bilgisayar_bilimleri_bilgi_sistemleri = veri[veri.konusu ==
'Bilgisayar Bilimleri, Bilgi Sistemleri']
    data_spor_bilimleri = veri[veri.konusu == 'Spor Bilimleri']
    data_isletme = veri[veri.konusu == 'İşletme']
    data_din_bilimi = veri[veri.konusu == 'Din Bilimi']
    data_tarih = veri[veri.konusu == 'Tarih']
    veri = pd.concat([data_matematik.sample(len(data_din_bilimi)),

data_deniz_ve_tatli_su_biyolojisi.sample(len(data_din_bilimi)),

data_muhendislik_elektrik_elektronik.sample(len(data_din_bilimi)),
                      data_fizik_uygulamali.sample(len(data_din_bilimi)),
                      data_cerrahi.sample(len(data_din_bilimi)),

data_saglik_bilimleri_ve_hizmetleri.sample(len(data_din_bilimi)),
                      data_muhendislik_makine.sample(len(data_din_bilimi)),
                      data_cevre_bilimleri.sample(len(data_din_bilimi)),

data_muhendislik_ortak_disiplinler.sample(len(data_din_bilimi)),
                      data_genel_ve_dahili_tip.sample(len(data_din_bilimi)),

data_egitim_bilimsel_disiplinler.sample(len(data_din_bilimi)),

data_bilgisayar_bilimleri_bilgi_sistemleri.sample(len(data_din_bilimi)),
                      data_spor_bilimleri.sample(len(data_din_bilimi)),
                      data_isletme.sample(len(data_din_bilimi)),
data_din_bilimi,
                         data_tarih])
    #print("Data'nın Yeni Boyutu: ", len(veri)) #333
    print("Yazım dili ingilizce olan makalelerin konusuna göre
sınıflandırılması")
    konusu_siniflandirma(veri)

    return

# 3 --> Konu alanı: 'TEMEL BİLİMLER' olan ingilizce makalelerin konusuna göre
sınıflandırma metodu
def naiveBayes_randomForest_svm_temelBilimler(veri):
    # konusuna göre makale sayıları eşit olmadığı için veriler dengeli hale
getiriliyor

    #print("Data'nın Bölünmeden Önceki Boyutu", len(veri)) #689
    data_matematik = veri[veri.konusu == 'Matematik']
    data_deniz_ve_tatli_su_biyolojisi = veri[veri.konusu == 'Deniz ve Tatlı Su
Biyolojisi']
    data_fizik_uygulamali = veri[veri.konusu == 'Fizik, Uygulamalı']
    data_cevre_bilimleri = veri[veri.konusu == 'Çevre Bilimleri']
    veri = pd.concat([data_matematik.sample(len(data_fizik_uygulamali)),

data_deniz_ve_tatli_su_biyolojisi.sample(len(data_fizik_uygulamali)),
data_fizik_uygulamali,
                         data_cevre_bilimleri])
    #print("Data'nın Yeni Boyutu: ", len(veri)) #589

    # sınıflandırma algoritmalarının uygulandığı metoda gidiyor
    print("Konu alanı: 'TEMEL BİLİMLER' olan ingilizce makalelerin konusuna
göre sınıflandırılması")
```

EK-2. (devam) “MetinIsleme.py” Python Kodu

```
    konusu_siniflandirma(veri)

    return
# 4 --> Konu alanı: 'MÜHENDİSLİK' olan ingilizce makalelerin konusuna göre
sınıflandırma metodu

def naiveBayes_randomForest_svm_muhendislik(veri):
    # konusuna göre makale sayıları eşit olmadığı için veriler dengeli hale
getiriliyor

    #print("Data'nın Bölünmeden Önceki Boyutu", len(veri)) #517
    data_muhendislik_elektrik_elektronik = veri[veri.konusu == 'Mühendislik,
Elektrik ve Elektronik']
    data_muhendislik_makine = veri[veri.konusu == 'Mühendislik, Makine']
    data_muhendislik_ortak_disiplinler = veri[veri.konusu == 'Mühendislik,
Ortak Disiplinler']
    data_bilgisayar_bilimleri_bilgi_sistemleri = veri[veri.konusu ==
'Bilgisayar Bilimleri, Bilgi Sistemleri']
    veri =
pd.concat([data_muhendislik_elektrik_elektronik.sample(len(data_muhendislik_ort
ak_disiplinler)),

data_muhendislik_makine.sample(len(data_muhendislik_ortak_disiplinler)),
data_muhendislik_ortak_disiplinler,
                           data_bilgisayar_bilimleri_bilgi_sistemleri])
    #print("Data'nın Yeni Boyutu: ", len(veri)) #451

    # sınıflandırma algoritmalarının uygulandığı metoda gidiyor
    print("Konu alanı: 'MÜHENDİSLİK' olan ingilizce makalelerin konusuna göre
sınıflandırılması")
    konusu_siniflandirma(veri)

    return

# 5 --> Konu alanı: 'TIP' olan ingilizce makalelerin konusuna göre
sınıflandırma metodu

def naiveBayes_randomForest_svm_tip(veri):
    # konusuna göre makale sayıları eşit olmadığı için veriler dengeli hale
getiriliyor
    #print("Data'nın Bölünmeden Önceki Boyutu", len(veri)) #478
    data_cerrahi = veri[veri.konusu == 'Cerrahi']
    data_saglik_bilimleri_ve_hizmetleri = veri[veri.konusu == 'Sağlık Bilimleri
ve Hizmetleri']
    data_genel_ve_dahili_tip = veri[veri.konusu == 'Genel ve Dahili Tıp']
    data_spor_bilimleri = veri[veri.konusu == 'Spor Bilimleri']
    veri = pd.concat([data_cerrahi.sample(len(data_genel_ve_dahili_tip)),

data_saglik_bilimleri_ve_hizmetleri.sample(len(data_genel_ve_dahili_tip)),
data_genel_ve_dahili_tip,
                          data_spor_bilimleri])
    #print("Data'nın Yeni Boyutu: ", len(veri)) #433

    # sınıflandırma algoritmalarının uygulandığı metoda gidiyor
    print("Konu alanı: 'TIP' olan ingilizce makalelerin konusuna göre
sınıflandırılması")

    konusu_siniflandirma(veri)

    return

# 6 --> Konu alanı: 'SOSYAL' olan ingilizce makalelerin konusuna göre
sınıflandırma metodu

def naiveBayes_randomForest_svm_sosyal(veri):
```

EK-2. (devam) “MetinIsleme.py” Python Kodu

```
    # konusuna göre makale sayıları eşit olmadığı için veriler dengeli hale
getiriliyor

    #print("Data'nın Bölünmeden Önceki Boyutu", len(veri)) #213
    data_egitim_bilimsel_disiplinler = veri[veri.konusu == 'Eğitim, Bilimsel
Disiplinler']
    data_isletme = veri[veri.konusu == 'İşletme']
    data_din_bilimi = veri[veri.konusu == 'Din Bilimi']
    data_tarih = veri[veri.konusu == 'Tarih']
    veri =
pd.concat([data_egitim_bilimsel_disiplinler.sample(len(data_din_bilimi)),
                           data_isletme.sample(len(data_din_bilimi)),
data_din_bilimi,
                           data_tarih])

    #print("Data'nın Yeni Boyutu: ", len(veri)) #81

    # sınıflandırma algoritmalarının uygulandığı metoda gidiyor
    print("Konu alanı: 'SOSYAL' olan ingilizce makalelerin konusuna göre
sınıflandırılması")

    konusu_siniflandirma(veri)

    return


# ingilizce makale metni işleme
def makale_ingilizce():

    # etkisiz kelimeleri etkisiz_kelime listesine atama
    etkisiz_kelime = stopwords.words('english')
    # print(etkisiz_kelime)
    # print(len(etkisiz_kelime))  # 179 ingilizce stop_words var

    # Lemmatizasyon (Lemmatization) genellikle bir kelime dağarcığı ve
kelimelerin morfolojik analizini kullanarak işleri düzgün bir şekilde yapmayı
ifade eder,

    wordnet_lemmatizer = WordNetLemmatizer()

    anlamsız = ['the', 'method', 'whether', 'introduction', 'purpose', 'aim',
'objective', 'although','study','result']

    for satirsayisi in range (0,len (makale_df)): # makale boyutu kadar döngü
devam ediyor,  len (makale) = 3802

        if (makale_df.dil == 'en')[satirsayisi] == True:  # n.satırdaki
makalelerin dili ingilizceyse eğer

            makale_icerigi_en = {
                'baslik': '',
                'konu_alani': '',
                'konusu': '',
                'oz': '',
                'anahtar_kelime': '',
                'tf_idf_kelime': ''
            }

            makale_icerigi_en['baslik'] = makale_df.loc[satirsayisi, 'baslik']
            makale_icerigi_en['konu_alani'] = makale_df.loc[satirsayisi, 'konu
alani']
            makale_icerigi_en['konusu'] = makale_df.loc[satirsayisi, 'konusu']
            makale_icerigi_en['oz'] = makale_df.loc[satirsayisi, 'oz']
            makale_icerigi_en['anahtar_kelime'] = makale_df.loc[satirsayisi,
'anahtar kelime']
```

EK-2. (devam) “MetinIsleme.py” Python Kodu

```
            makale_en.append(makale_icerigi_en)

            en_oz = makale_icerigi_en['oz']  # n. ci satır, oz sütunu

            # ----------------------------------------------------------------
--
            # TF (Terim Sıklığı) = Metinde bir terimin görünme sayısı /
Belgedeki toplam kelime sayısı
            # IDF (Ters Belge Frekansı) = log (toplam cümle sayısı / t terimli
cümle sayısı)
            # TF-IDF = TF * IDF = değeri yüksek olan kelimenin önemi de
yüksektir

            # Belge -> Vectorize -> TF Bul -> IDF Bul -> TF Bul * IDF ->
Anahtar Kelimeler

            # TF*IDF -->  1.adım : makaledeki toplam cümle sayısı
            toplam_cumle = tokenize.sent_tokenize(en_oz, "english")
            cumle_sayisi = len(toplam_cumle)
            #print(cumle_sayisi)

            # metinden sayıları çıkarma
            en_oz = re.sub(r"\d+", " ", en_oz)

            # metni küçük harfe çevirme
            en_oz = en_oz.lower()

            # metinden noktalama işaretlerini çıkarma
            noktalama = str.maketrans('', '', string.punctuation)
            en_oz = en_oz.translate(noktalama)  ##32 tane noktama işareti var

            # TF*IDF --> 2.adım : makaledeki toplam kelime sayısı
            en_oz = word_tokenize(en_oz, "english", False)
            kelime_sayisi = len(en_oz)  # makaledeki toplam kelime sayısı
            #print(kelime_sayisi)

            # metinden etkisiz kelimeleri ve anlamsız kelime listesini çıkarma
            en_oz = [word for word in en_oz
                     if not word in etkisiz_kelime
                     if not word in anlamsız]

            # Lemmatizasyon işlemi
            en_oz_lemma = []
            for bir_kelime in en_oz:
                bir_kelime = wordnet_lemmatizer.lemmatize(bir_kelime)
                en_oz_lemma.append(bir_kelime)

            #print(en_oz_lemma)

            # TF*IDF --> 3.adım : her kelime için TF değerini hesaplama
            # TF: Term Frequency / Terim Sıklığı : seçili kelimenin, metin
içinde bulunan toplam kelime sayısına bölümüdür.
            tf_deger = {}

            for bir_kelime in en_oz_lemma:
                if bir_kelime in tf_deger:
                    tf_deger[bir_kelime] += 1
                else:
                    tf_deger[bir_kelime] = 1

            #print(tf_deger) #her kelimenin metinde geçtiği sayı

            # tf değeri: Her bir kelime için toplam kelime_sayısına bölme
            tf_deger.update((x, y / int(kelime_sayisi)) for x, y in
tf_deger.items())
```

EK-2. (devam) “MetinIsleme.py” Python Kodu

```
            # print(tf_deger)

            # ilk makalenin toplam kelime sayısı 593, 'dairy' kelimesinin
makalede toplam geçtiği sayı 12
            # ilk makale için 'dairy' kelimesinin tf değeri 12/593 =
0.02023608768971332 dir.
            # TF*IDF --> 4.adım: her kelime için IDF hesaplama

            # Cümle listesinde kelimenin olup olmadığının kontrolü
            def kelime_kontrol(kelime, cumle):
                son = [all([w in x for w in kelime]) for x in cumle]
                uzunluk = [cumle[i] for i in range(0, len(son)) if son[i]]
                return int(len(uzunluk))

            # IDF: Inverse Document Frequency / Ters Döküman Sıklığı :
            # metinlerimizin kaçında kelimemiz var bunu gösterir. Toplam metin
adetimizin kelimeyi içeren metin adetine bölümünün logaritmasıdır.
            idf_deger = {}
            for bir_kelime in en_oz_lemma:
                if bir_kelime in idf_deger:
                    idf_deger[bir_kelime] = kelime_kontrol(bir_kelime,
en_oz_lemma)
                else:
                    idf_deger[bir_kelime] = 1
            # idf değeri: logaritmasını alıp, bölme
            #print(idf_deger)

            # ilk makaledeki tf değerini hesapladığımız 'dairy' kelimesi için
idf değeri;
            # log (toplam cümle/ dairy kelimesinin geçtiği cümle sayısı) =
log(26/ 12) = log(2.1666666666666665) = 0.3357921019231931

            idf_deger.update((x, math.log(int(cumle_sayisi) / y,10)) for x, y
in idf_deger.items())
            #print(idf_deger)


            # TF*IDF --> 5.adım : TF*IDF değerini hesaplama
            tf_idf_deger = {key: tf_deger[key] * idf_deger.get(key, 0) for key
in tf_deger.keys()}

            #print(tf_idf_deger)

            # 'dairy' kelimesinin tf değeri = 0.02023608768971332, idf değeri =
0.3357921019231931
            # tf*idf değeri = 0.006795118420030889

            # Belgedeki en önemli ilk N kelimeyi alma
            def en_kelime(sozluk_en, n):
                sonuc = dict(sorted(sozluk_en.items(), key=itemgetter(1),
reverse=True)[:n])

                return sonuc

            onemli_kelimeler = en_kelime(tf_idf_deger, 100)
            # print(onemli_kelimeler) # kelimeler ve tf*idf değerleri
            # print(onemli_kelimeler.keys()) # kelimeler
            # print(onemli_kelimeler.values()) # kelimelerin tf*idf değerleri


            tf_idf_kelime = " ".join(onemli_kelimeler.keys())  # tf*idf yüksek
olan kelimelerin karakter dizisi

            # kelimelerin en yüksek 25 tf*idf 'e sahip olanlar tf_idf_kelime
sütunu oluşturulup eklendi
```

EK-2. (devam) “MetinIsleme.py” Python Kodu

```
            makale_icerigi_en['tf_idf_kelime'] = tf_idf_kelime
        else:
            continue

    return makale_en
# makale_ingilizce() metodundan dönen makale_en dataframe i alıyor

# makale sınıflandırma metodu
def siniflandirma_en():
    makale_en = makale_ingilizce()

    makale_en_df = pd.DataFrame(makale_en)
    #print(makale_en_df)  # [1897 rows x 6 columns] olan bir data frame
oluşturuldu

    # makalelerin öz ve konu alanına göre modellemesini yapmak için;
    # öncelikle öz kısmına metin işleme yapıp önemli kelimeler listesi olan
tf*idf sütunu ile konu alanı kısmı alınıyor

    en_veri_konu_alani = makale_en_df[['tf_idf_kelime', 'konu_alani']]
    #en_veri_konu_alani = makale_en_df[['konusu', 'konu_alani']] # doğruluk
½100 :)
    #en_veri_konu_alani = makale_en_df[['oz', 'konu_alani']]
    #en_veri_konu_alani = makale_en_df[['baslik', 'konu_alani']]

    #print (en_veri_konu_alani.head())
    #print (en_veri_konu_alani.konu_alani.value_counts()) # her konu alanında
bulunan makale sayısını gösterir

    # 1 --> konu alanına göre sınıflandırma metodu
    naiveBayes_randomForest_svm_konuAlani(en_veri_konu_alani)
    print("-------------------------------------------------")

    # 2 --> konusuna göre sınıflandırma metodu
    en_veri_konusu = makale_en_df[['tf_idf_kelime', 'konusu']]
    en_veri_konusu_df = pd.DataFrame(en_veri_konusu)

    naiveBayes_randomForest_svm_konusu(en_veri_konusu_df)
    print(en_veri_konusu_df.konusu.value_counts())

    print("-------------------------------------------------")

    #-----------------------------------------------------------------------

    # 3-->  konu alanı 'TEMEL BİLİMLER' olan makalelerin sınıflandırma metodu
    en_veri_temel_bilimler= makale_en_df.loc[makale_en_df.konu_alani=='TEMEL
BİLİMLER',['tf_idf_kelime','konusu']]
    en_veri_temel_bilimler_df = pd.DataFrame(en_veri_temel_bilimler)

    naiveBayes_randomForest_svm_temelBilimler(en_veri_temel_bilimler_df)
    print(en_veri_temel_bilimler_df.konusu.value_counts())
    print("-------------------------------------------------")

    # 4 --> konu alanı 'MÜHENDİSLİK' olan makalelerin sınıflandırma metodu
    en_veri_muhendislik = makale_en_df.loc[makale_en_df.konu_alani ==
'MÜHENDİSLİK', ['tf_idf_kelime', 'konusu']]
    en_veri_muhendislik_df = pd.DataFrame(en_veri_muhendislik)

    naiveBayes_randomForest_svm_muhendislik(en_veri_muhendislik_df)
    print(en_veri_muhendislik_df.konusu.value_counts())
    print("-------------------------------------------------")

    # 5--> konu alanı 'TIP' olan makalelerin sınıflandırma metodu
    en_veri_tip = makale_en_df.loc[makale_en_df.konu_alani == 'TIP',
```

EK-2. (devam) “MetinIsleme.py” Python Kodu

```
['tf_idf_kelime', 'konusu']]
    en_veri_tip_df = pd.DataFrame(en_veri_tip)

    naiveBayes_randomForest_svm_tip(en_veri_tip_df)
    print(en_veri_tip_df.konusu.value_counts())
    print("-------------------------------------------------")

    # 6 --> konu alanı 'SOSYAL' olan makalelerin sınıflandırma metodu
    en_veri_sosyal = makale_en_df.loc[makale_en_df.konu_alani == 'SOSYAL',
['tf_idf_kelime', 'konusu']]

    en_veri_sosyal_df = pd.DataFrame(en_veri_sosyal)

    naiveBayes_randomForest_svm_sosyal(en_veri_sosyal_df)
    print(en_veri_sosyal_df.konusu.value_counts())
    print("-------------------------------------------------")
    return

# makale_ingilizce() metodu --> siniflandirma_en() --> 2,3,4,5,6 -->
konusu_siniflandirma(veri)

# 1: NaiveBayes_randomForest_konuAlani(en_veri)
# 2: naiveBayes_randomForest_svm_konusu(en_veri_konusu_df)
# 3: naiveBayes_randomForest_svm_temelBilimler(en_veri_temel_bilimler_df)
# 4: naiveBayes_randomForest_svm_muhendislik(en_veri_muhendislik_df)
# 5: naiveBayes_randomForest_svm_tip(en_veri_tip_df)
# 6: naiveBayes_randomForest_svm_sosyal(en_veri_sosyal_df)

##############################################################################
############
siniflandirma_en()
##############################################################################
############

# 1 --> 'konu alanına' göre türkçe makaleleri sınıflandırma
def naiveBayes_randomForest_svm_konuAlani_tr(veri):
    # konu alanına göre makale sayıları eşit olmadığı için veriler dengeli hale
getiriliyor
    # print("Data'nın Bölünmeden Önceki Boyutu", len(veri)) #1905
    data_sosyal = veri[veri.konu_alani == 'SOSYAL']
    data_tip = veri[veri.konu_alani == 'TIP']
    data_muhendislik = veri[veri.konu_alani == 'MÜHENDİSLİK']
    data_temel_bilimler = veri[veri.konu_alani == 'TEMEL BİLİMLER']

    veri = pd.concat([data_sosyal.sample(len(data_muhendislik)),
                      data_tip.sample(len(data_muhendislik)), data_muhendislik,
                      data_temel_bilimler])
    # print("Data'nın Yeni Boyutu: ", len(veri)) #1588

    # metinlerin hepsini kapsayan bir corpus oluşturuluyor

    corpus = []
    for i in range(len(veri)):
        corpus.append(veri.iloc[i, 0])

    # Corpus ile TF-IDF modeli fit ediliyor
    vectorizer = TfidfVectorizer()
    vectorizer.fit(corpus)

    # corpus transform ediliyor
    X = vectorizer.transform(corpus).toarray()
    # konu alanına göre sınıflandırma yapılıyor
    y = veri.konu_alani

    # Stratify=y'de train ve testteki y oranının aynı olmasını sağlıyor
```

EK-2. (devam) “MetinIsleme.py” Python Kodu

```
    # Veri setinin %75 i train, %25 i test olmak üzere 2'ye ayrılıyor
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=39, stratify=y)


    print("Yazım dili türkçe olan makalelerin konu alanına göre
sınıflandırılması")
    # 1. sınıflandırıcı ----------------------------------------------------
--
    # Naive Bayes sınıflandırıcısı denetimli bir sınıflandırma algoritmasıdır.
    # Naive Bayes sınıflandırıcısı (Supervised classification algorithm) ile
veri seti sınıflandırılıyor
    print("1 --> Naive Bayes Sınıflandırıcısı")
    Naive = naive_bayes.MultinomialNB()
    Naive.fit(X_train, y_train)
    y_pred_nb = Naive.predict(X_test)

    print(classification_report(y_test, y_pred_nb))
    # doğruluk yüzdesini gösteriyor
    print("Doğruluk Yüzdesi: ")
    print(accuracy_score(y_test, y_pred_nb))
    # hata matrisi tablo olarak gösteriliyor
    print("Hata Matrisi: ")
    print(confusion_matrix(y_test, y_pred_nb))

    # 2. sınıflandırıcı----------------------------------------------------
--
    # Rastgele Orman algoritması denetimli bir sınıflandırma algoritmasıdır.
    # Random Forest algoritması(Supervised classification algorithm) ile veri
seti sınıflandırılıyor
    print("2 --> Rastgele Orman Algoritması")
    rf = RandomForestClassifier(class_weight='balanced_subsample')
    rf.fit(X_train, y_train)
    y_pred_rf = rf.predict(X_test)

    # precision : tahmin yüzdesini ifade ediyor
    # f1-score : tıp alanını diğer alanlardan ayıran belirgin özellik olduğunu
gösteriyor

    print(classification_report(y_test, y_pred_rf))
    # doğruluk yüzdesini gösteriyor
    print("Doğruluk Yüzdesi: ")
    print(accuracy_score(y_test, y_pred_rf))
    # hata matrisi tablo olarak gösteriliyor
    print("Hata Matrisi: ")
    print(confusion_matrix(y_test, y_pred_rf))
    # Random forest algoritmasında hangi değişkenin modellemede önemli olduğunu
gösteriyor
    #rf_onemli = pd.DataFrame({'features': vectorizer.get_feature_names(),
'importance': rf.feature_importances_})
    #rf_onemli = rf_onemli.sort_values(by='importance', ascending=False)
    # değişken isimleri ve önem skorları
    #print(rf_onemli.head(50))

    # 3. sınıflandırıcı----------------------------------------------------
--
    # Destek Vektör Makinesi denetimli bir sınıflandırma algoritmasıdır.
    # Support Vector Machine (Supervised classification algorithm) ile veri
seti sınıflandırılıyor
    print("3 --> Destek Vektör Makinesi")
    SVM = svm.SVC(C=1.0, kernel='linear', degree=3, gamma='auto')
    SVM.fit(X_train, y_train)
    y_pred_svm = SVM.predict(X_test)

    print(classification_report(y_test, y_pred_svm))
```

EK-2. (devam) “MetinIsleme.py” Python Kodu

```
    # doğruluk yüzdesini gösteriyor
    print("Doğruluk Yüzdesi: ")
    print(accuracy_score(y_test, y_pred_svm))
    # hata matrisi tablo olarak gösteriliyor
    print("Hata Matrisi: ")
    print(confusion_matrix(y_test, y_pred_svm))

    # Denetimli sınıflandırma algoritmalarının doğruluk yüzdeleri;  Support
Vector Machine > Random Forest > Naive Bayes

    return

# 2 --> 'konusuna' göre türkçe makaleleri sınıflandırma metodu
def naiveBayes_randomForest_svm_konusu_tr(veri):
    # konu alanına göre makale sayıları eşit olmadığı için veriler dengeli hale
getiriliyor
    #print("Data'nın Bölünmeden Önceki Boyutu", len(veri)) #1905
    data_tarih = veri[veri.konusu == 'Tarih']
    data_din_bilimi = veri[veri.konusu == 'Din Bilimi']
    data_isletme = veri[veri.konusu == 'İşletme']
    data_spor_bilimleri = veri[veri.konusu == 'Spor Bilimleri']
    data_bilgisayar_bilimleri_bilgi_sistemleri = veri[veri.konusu ==
'Bilgisayar Bilimleri, Bilgi Sistemleri']
    data_egitim_bilimsel_disiplinler = veri[veri.konusu == 'Eğitim, Bilimsel
Disiplinler']
    data_genel_ve_dahili_tip = veri[veri.konusu == 'Genel ve Dahili Tıp']
    data_muhendislik_ortak_disiplinler = veri[veri.konusu == 'Mühendislik,
Ortak Disiplinler']
    data_cevre_bilimleri = veri[veri.konusu == 'Çevre Bilimleri']
    data_muhendislik_makine = veri[veri.konusu == 'Mühendislik, Makine']
    data_saglik_bilimleri_ve_hizmetleri = veri[veri.konusu == 'Sağlık Bilimleri
ve Hizmetleri']
    data_cerrahi = veri[veri.konusu == 'Cerrahi']
    data_fizik_uygulamali = veri[veri.konusu == 'Fizik, Uygulamalı']
    data_muhendislik_elektrik_elektronik = veri[veri.konusu == 'Mühendislik,
Elektrik ve Elektronik']
    data_deniz_ve_tatli_su_biyolojisi = veri[veri.konusu == 'Deniz ve Tatlı Su
Biyolojisi']
    #data_matematik = veri[veri.konusu == 'Matematik'] # makale sayısı 3 tane
olduğu için tahminleme hatalı olabiliyor
    veri =
pd.concat([data_tarih.sample(len(data_muhendislik_elektrik_elektronik)),

data_din_bilimi.sample(len(data_muhendislik_elektrik_elektronik)),

data_isletme.sample(len(data_muhendislik_elektrik_elektronik)),

data_spor_bilimleri.sample(len(data_muhendislik_elektrik_elektronik)),

data_bilgisayar_bilimleri_bilgi_sistemleri.sample(len(data_muhendislik_elektrik
_elektronik)),

data_egitim_bilimsel_disiplinler.sample(len(data_muhendislik_elektrik_elektroni
k)),

data_genel_ve_dahili_tip.sample(len(data_muhendislik_elektrik_elektronik)),

data_muhendislik_ortak_disiplinler.sample(len(data_muhendislik_elektrik_elektro
nik)),

data_cevre_bilimleri.sample(len(data_muhendislik_elektrik_elektronik)),

data_muhendislik_makine.sample(len(data_muhendislik_elektrik_elektronik)),

data_saglik_bilimleri_ve_hizmetleri.sample(len(data_muhendislik_elektrik_elektr
onik)),
```

EK-2. (devam) “MetinIsleme.py” Python Kodu

```
data_cerrahi.sample(len(data_muhendislik_elektrik_elektronik)),

data_fizik_uygulamali.sample(len(data_muhendislik_elektrik_elektronik)),
data_muhendislik_elektrik_elektronik,
                          data_deniz_ve_tatli_su_biyolojisi])

    #print("Data'nın Yeni Boyutu: ", len(veri)) #1019
    konusu_siniflandirma(veri)

    return

# 3 --> Konu alanı: 'TEMEL BİLİMLER' olan türkçe makalelerin konusuna göre
sınıflandırma metodu

def naiveBayes_randomForest_svm_temelBilimler_tr(veri):
    # konusuna göre makale sayıları eşit olmadığı için veriler dengeli hale
getiriliyor
    #print("Data'nın Bölünmeden Önceki Boyutu", len(veri)) #262

    data_cevre_bilimleri = veri[veri.konusu == 'Çevre Bilimleri']
    data_fizik_uygulamali = veri[veri.konusu == 'Fizik, Uygulamalı']
    data_deniz_ve_tatli_su_biyolojisi = veri[veri.konusu == 'Deniz ve Tatlı Su
Biyolojisi']
    data_matematik = veri[veri.konusu == 'Matematik'] # matematik alanında
türkçe 3 makale olduğu için tahminleme hatası olabiliyor

    veri =
pd.concat([data_cevre_bilimleri.sample(len(data_deniz_ve_tatli_su_biyolojisi)),

data_fizik_uygulamali.sample(len(data_deniz_ve_tatli_su_biyolojisi)),
data_deniz_ve_tatli_su_biyolojisi,
                    data_matematik])

    #print("Data'nın Yeni Boyutu: ", len(veri)) #233

    # sınıflandırma algoritmalarının uygulandığı metoda gidiyor
    print("Konu alanı: 'TEMEL BİLİMLER' olan türkçe makalelerin konusuna göre
sınıflandırılması")

    konusu_siniflandirma(veri)

    return

# 4 --> Konu alanı: 'MÜHENDİSLİK' olan türkçe makalelerin konusuna göre
sınıflandırma metodu

def naiveBayes_randomForest_svm_muhendislik_tr(veri):
    # konusuna göre makale sayıları eşit olmadığı için veriler dengeli hale
getiriliyor

    #print("Data'nın Bölünmeden Önceki Boyutu", len(veri)) #442

    data_bilgisayar_bilimleri_bilgi_sistemleri = veri[veri.konusu ==
'Bilgisayar Bilimleri, Bilgi Sistemleri']
    data_muhendislik_ortak_disiplinler = veri[veri.konusu == 'Mühendislik,
Ortak Disiplinler']
    data_muhendislik_makine = veri[veri.konusu == 'Mühendislik, Makine']
    data_muhendislik_elektrik_elektronik = veri[veri.konusu == 'Mühendislik,
Elektrik ve Elektronik']

    veri =
pd.concat([data_bilgisayar_bilimleri_bilgi_sistemleri.sample(len(data_muhendisl
ik_makine)),

data_muhendislik_ortak_disiplinler.sample(len(data_muhendislik_makine)),
```

EK-2. (devam) “MetinIsleme.py” Python Kodu

```
data_muhendislik_makine,
                          data_muhendislik_elektrik_elektronik])
    #print("Data'nın Yeni Boyutu: ", len(veri)) #380

    # sınıflandırma algoritmalarının uygulandığı metoda gidiyor
    print("Konu alanı: 'MÜHENDİSLİK' olan türkçe makalelerin konusuna göre
sınıflandırılması")

    konusu_siniflandirma(veri)

    return
# 5 --> Konu alanı: 'TIP' olan türkçe makalelerin konusuna göre sınıflandırma
metodu

def naiveBayes_randomForest_svm_tip_tr(veri):
    # konusuna göre makale sayıları eşit olmadığı için veriler dengeli hale
getiriliyor
    #print("Data'nın Bölünmeden Önceki Boyutu", len(veri)) #478

    data_spor_bilimleri = veri[veri.konusu == 'Spor Bilimleri']
    data_genel_ve_dahili_tip = veri[veri.konusu == 'Genel ve Dahili Tıp']
    data_saglik_bilimleri_ve_hizmetleri = veri[veri.konusu == 'Sağlık Bilimleri
ve Hizmetleri']
    data_cerrahi = veri[veri.konusu == 'Cerrahi']
    veri =
pd.concat([data_spor_bilimleri.sample(len(data_saglik_bilimleri_ve_hizmetleri))
,

data_genel_ve_dahili_tip.sample(len(data_saglik_bilimleri_ve_hizmetleri)),
data_saglik_bilimleri_ve_hizmetleri,
                          data_cerrahi])
    #print("Data'nın Yeni Boyutu: ", len(veri)) #433

    # sınıflandırma algoritmalarının uygulandığı metoda gidiyor
    print("Konu alanı: 'TIP' olan türkçe makalelerin konusuna göre
sınıflandırılması")

    konusu_siniflandirma(veri)

    return

# 6 --> Konu alanı: 'SOSYAL' olan türkçe makalelerin konusuna göre
sınıflandırma metodu

def naiveBayes_randomForest_svm_sosyal_tr(veri):
    # konusuna göre makale sayıları eşit olmadığı için veriler dengeli hale
getiriliyor
    #print("Data'nın Bölünmeden Önceki Boyutu", len(veri)) #721

    data_tarih = veri[veri.konusu == 'Tarih']
    data_din_bilimi = veri[veri.konusu == 'Din Bilimi']
    data_isletme = veri[veri.konusu == 'İşletme']
    data_egitim_bilimsel_disiplinler = veri[veri.konusu == 'Eğitim, Bilimsel
Disiplinler']

    veri = pd.concat([data_tarih.sample(len(data_isletme)),
                          data_din_bilimi.sample(len(data_isletme)),
data_isletme,
                          data_egitim_bilimsel_disiplinler])
    #print("Data'nın Yeni Boyutu: ", len(veri)) #649

    # sınıflandırma algoritmalarının uygulandığı metoda gidiyor
    print("Konu alanı: 'SOSYAL' olan türkçe makalelerin konusuna göre
sınıflandırılması")
```

EK-2. (devam) “MetinIsleme.py” Python Kodu

```
    konusu_siniflandirma(veri)

    return

# türkçe makale metni işleme
def makale_turkce():
    # etkisiz kelimeleri etkisiz_kelime listesine atama

    etkisiz_kelime = stopwords.words('turkish')
    # print(etkisiz_kelime)
    # print(len(etkisiz_kelime))  # 53 türkçe stop_words var

    anlamsız = ['ve', 'amaç', 'amacı,', 'çalışmamız', 'bu', 'birlikte',
'çalışma', 'çalışmayı',
                'çalışmada', 'çalışmalarında', 'çalışmada,', 'çalışmanın',
'göre', 'arasında',
                'günümüz', 'günümüzde', 'günümüze', 'günümüzün', 'çoğu',
'alanda', 'alakalı', 'genellikle',
                'üzerine', 'xd', 'amacı', 'hemen', 'öz', 'öz:', 'ilgili',
'giriş', 'özetxd',
                'yöntemini', 'son', 'yıllarda,''arasında', 'amacıyla', 'özet',
'amaçtıp', 'olduğu', 'bir']


    for satirsayisi in range(0, len(makale_df)):  # makale boyutu kadar döngü
devam ediyor,  len (makale) = 3802
        if (makale_df.dil == 'tr')[satirsayisi] == True:  # n.satırdaki
makalelerin dili türkçeyse eğer

            makale_icerigi_tr = {
                'baslik': '',
                'konu_alani': '',
                'konusu': '',
                'oz': '',
                'anahtar_kelime': '',
                'tf_idf_kelime': ''
            }
            makale_icerigi_tr['baslik'] = makale_df.loc[satirsayisi, 'baslik']
            makale_icerigi_tr['konu_alani'] = makale_df.loc[satirsayisi, 'konu
alani']
            makale_icerigi_tr['konusu'] = makale_df.loc[satirsayisi, 'konusu']
            makale_icerigi_tr['oz'] = makale_df.loc[satirsayisi, 'oz']
            makale_icerigi_tr['anahtar_kelime'] = makale_df.loc[satirsayisi,
'anahtar kelime']

            makale_tr.append(makale_icerigi_tr)

            tr_oz = makale_icerigi_tr['oz']  # n. ci satır, oz sütunu
            tr_oz = tr_oz.lower()  # metni küçük harfe çevirme

            # print(dokuman_kelime[:3])
            # print(len(dokuman_kelime))  #  kelime sayısı

            # ----------------------------------------------------------------
--
            # TF (Terim Sıklığı) = Metinde bir terimin görünme sayısı /
Belgedeki toplam kelime sayısı
            # IDF (Ters Belge Frekansı) = log (toplam cümle sayısı / t terimli
cümle sayısı)
            # TF-IDF = TF * IDF = değeri yüksek olan kelimenin önemi de
yüksektir

            # Belge -> Vectorize -> TF Bul -> IDF Bul -> TF Bul * IDF ->
Anahtar Kelimeler
```

EK-2. (devam) “MetinIsleme.py” Python Kodu

```
            # TF*IDF -->  1.adım: makaledeki toplam cümle sayısı
            toplam_cumle = tokenize.sent_tokenize(tr_oz, "turkish")
            cumle_sayisi = len(toplam_cumle)
            #print(cumle_sayisi)

            # metinden sayıları çıkarma
            tr_oz = re.sub(r"\d+", " ", tr_oz)

            # metni küçük harfe çevirme
            tr_oz = tr_oz.lower()

            # metinden noktalama işaretlerini çıkarma
            noktalama = str.maketrans('', '', string.punctuation)
            tr_oz = tr_oz.translate(noktalama)  ##32 tane noktama işareti var

            # TF*IDF --> 2.adım: makaledeki toplam kelime sayısı

            tr_oz = word_tokenize(tr_oz, "turkish", False)
            kelime_sayisi = len(tr_oz)  # makaledeki toplam kelime sayısı
            #print(kelime_sayisi)

            # metinden etkisiz kelimeleri ve anlamsız kelime listesini çıkarma
            tr_oz = [word for word in tr_oz
                     if not word in etkisiz_kelime
                     if not word in anlamsız]

            # TF*IDF --> 3.adım: her kelime için TF değerini hesaplama
            # TF: Term Frequency / Terim Sıklığı: seçili kelimenin, metin
içinde bulunan toplam kelime sayısına bölümüdür.
            tf_deger = {}
            for bir_kelime in tr_oz:
                if bir_kelime in tf_deger:
                    tf_deger[bir_kelime] += 1
                else:
                    tf_deger[bir_kelime] = 1

            #print(tf_deger) #her kelimenin metinde geçtiği sayı

            # tf değeri: Her bir kelime için toplam kelime_sayısına bölme
            tf_deger.update((x, y / int(kelime_sayisi)) for x, y in
tf_deger.items())
            #print(tf_deger)

            # ilk makalenin toplam kelime sayısı 316, 'sağlık' kelimesinin
makalede toplam geçtiği sayı 2
            # ilk makale için 'sağlık' kelimesinin tf değeri 2/316 =
0.006329113924050633 dür.


            # TF*IDF --> 4.adım: her kelime için IDF hesaplama

            # Cümle listesinde kelimenin olup olmadığının kontrolü
            def kelime_kontrol(kelime, cumle):
                son = [all([w in x for w in kelime]) for x in cumle]
                uzunluk = [cumle[i] for i in range(0, len(son)) if son[i]]
                return int(len(uzunluk))


            # IDF: Inverse Document Frequency / Ters Döküman Sıklığı :
            # metinlerimizin kaçında kelimemiz var bunu gösterir. Toplam metin
adetimizin kelimeyi içeren metin adetine bölümünün logaritmasıdır.

            idf_deger = {}
            for bir_kelime in tr_oz:
                if bir_kelime in idf_deger:
                    idf_deger[bir_kelime] = kelime_kontrol(bir_kelime, tr_oz)
```

EK-2. (devam) “MetinIsleme.py” Python Kodu

```
                else:
                    idf_deger[bir_kelime] = 1

            # idf değeri: logaritmasını alıp, bölme
            #print(idf_deger)

            # ilk makaledeki tf değerini hesapladığımız 'sağlık' kelimesi için
idf değeri;
            # log (toplam cümle/ sağlık kelimesinin geçtiği cümle sayısı) =
log(20/ 2) = log(10) = 1.0 dır

            idf_deger.update((x, math.log(int(cumle_sayisi) / y, 10)) for x, y
in idf_deger.items())
            #print(idf_deger)

            # TF*IDF --> 5.adım: TF*IDF değerini hesaplama
            tf_idf_deger = {key: tf_deger[key] * idf_deger.get(key, 0) for key
in tf_deger.keys()}

            #print(tf_idf_deger)
            # 'sağlık' kelimesinin tf değeri = 0.006329113924050633, idf değeri
= 1.0
            # tf*idf değeri = 0.006329113924050633 dür

            # Belgedeki en önemli ilk N kelimeyi alma
            def en_kelime(sozluk_en, n):
                sonuc = dict(sorted(sozluk_en.items(), key=itemgetter(1),
reverse=True)[:n])
                return sonuc

            onemli_kelimeler = en_kelime(tf_idf_deger, 100)
            #print(onemli_kelimeler) # kelimeler ve tf*idf değerleri
            # print(onemli_kelimeler.keys()) # kelimeler
            # print(onemli_kelimeler.values()) # kelimelerin tf*idf değerleri

            tf_idf_kelime = " ".join(onemli_kelimeler.keys())  # tf*idf yüksek
olan kelimelerin karakter dizisi

            # kelimelerin en yüksek 100 tf*idf 'e sahip olanlar tf_idf_kelime
sütunu oluşturulup eklendi
            makale_icerigi_tr['tf_idf_kelime'] = tf_idf_kelime

        else:
            continue

    return makale_tr

# makale_turkce() metodundan dönen makale_en dataframe i alıyor
# makale sınıflandırma metodu
def siniflandirma_tr():
    makale_tr = makale_turkce()
    makale_tr_df = pd.DataFrame(makale_tr)
    #print(makale_tr_df)  # [1905 rows x 6 columns] olan bir data frame
oluşturuldu

    # makalelerin öz ve konu alanına göre modellemesini yapmak için;
    # öncelikle öz kısmına metin işleme yapıp önemli kelimeler listesi olan
tf*idf sütunu ile konu alanı kısmı alınıyor
    tr_veri_konu_alani = makale_tr_df[['tf_idf_kelime', 'konu_alani']]
    #tr_veri_konu_alani = makale_tr_df[['konusu', 'konu_alani']] # doğruluk
%100 :)
    #tr_veri_konu_alani = makale_tr_df[['oz', 'konu_alani']]
    #tr_veri_konu_alani = makale_tr_df[['baslik', 'konu_alani']]
    #print (tr_veri_konu_alani.head())
    #print (tr_veri_konu_alani.konu_alani.value_counts()) # her konu alanında
bulunan makale sayısını gösterir
```

EK-2. (devam) “MetinIsleme.py” Python Kodu

```
    # 1--> konu alanına göre sınıflandırma metodu
    naiveBayes_randomForest_svm_konuAlani_tr(tr_veri_konu_alani)
    print("-------------------------------------------------")

    # 2--> konusuna göre sınıflandırma metodu
    tr_veri_konusu = makale_tr_df[['tf_idf_kelime', 'konusu']]
    tr_veri_konusu_df = pd.DataFrame(tr_veri_konusu)

    naiveBayes_randomForest_svm_konusu_tr(tr_veri_konusu_df)
    print(tr_veri_konusu_df.konusu.value_counts())
    print("-------------------------------------------------")
    # ------------------------------------------------------------------------
    # 3--> konu alanı 'TEMEL BİLİMLER' olan makalelerin sınıflandırma metodu
    tr_veri_temel_bilimler = makale_tr_df.loc[makale_tr_df.konu_alani == 'TEMEL
BİLİMLER', ['tf_idf_kelime', 'konusu']]
    tr_veri_temel_bilimler_df = pd.DataFrame(tr_veri_temel_bilimler)

    naiveBayes_randomForest_svm_temelBilimler_tr(tr_veri_temel_bilimler_df)
    print(tr_veri_temel_bilimler_df.konusu.value_counts())
    print("-------------------------------------------------")

    # 4--> konu alanı 'MÜHENDİSLİK' olan makalelerin sınıflandırma metodu
    tr_veri_muhendislik = makale_tr_df.loc[makale_tr_df.konu_alani ==
'MÜHENDİSLİK', ['tf_idf_kelime', 'konusu']]
    tr_veri_muhendislik_df = pd.DataFrame(tr_veri_muhendislik)

    naiveBayes_randomForest_svm_muhendislik_tr(tr_veri_muhendislik_df)
    print(tr_veri_muhendislik_df.konusu.value_counts())
    print("-------------------------------------------------")

    # 5--> konu alanı 'TIP' olan makalelerin sınıflandırma metodu
    tr_veri_tip = makale_tr_df.loc[makale_tr_df.konu_alani == 'TIP',
['tf_idf_kelime', 'konusu']]
    tr_veri_tip_df = pd.DataFrame(tr_veri_tip)

    naiveBayes_randomForest_svm_tip_tr(tr_veri_tip_df)
    print(tr_veri_tip_df.konusu.value_counts())
    print("-------------------------------------------------")

    # 6--> konu alanı 'SOSYAL' olan makalelerin sınıflandırma metodu
    tr_veri_sosyal = makale_tr_df.loc[makale_tr_df.konu_alani == 'SOSYAL',
['tf_idf_kelime', 'konusu']]
    tr_veri_sosyal_df = pd.DataFrame(tr_veri_sosyal)

    naiveBayes_randomForest_svm_sosyal_tr(tr_veri_sosyal_df)
    print(tr_veri_sosyal_df.konusu.value_counts())
    print("-------------------------------------------------")
    return
# makale_turkce() metodu --> siniflandirma_tr() --> 3,4,5,6 -->
konusu_siniflandirma(veri)
# 1: naiveBayes_randomForest_svm_konuAlani_tr(tr_veri_konu_alani)
# 2: naiveBayes_randomForest_svm_konusu_tr(tr_veri_konusu_df)
# 3: naiveBayes_randomForest_svm_temelBilimler_tr(tr_veri_temel_bilimler_df)
# 4: naiveBayes_randomForest_svm_muhendislik_tr(tr_veri_muhendislik_df)
# 5: naiveBayes_randomForest_svm_tip_tr(tr_veri_tip_df)
# 6: naiveBayes_randomForest_svm_sosyal_tr(tr_veri_sosyal_df)


##########################################################################
############
siniflandirma_tr()
##########################################################################
############
```

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı : GÜRBÜZ, Tuğba

Uyruğu : T.C.

Medeni hali

Faks

E-mail

Eğitim

Derece	Eğitim Birimi	Mezuniyet Tarihi
Yüksek lisans	Gazi Üniversitesi/Bilişim Enstitüsü/Bilişim Sistemleri	Devam ediyor
Lisans	Marmara Üniversitesi/Mühendislik Fakültesi/Bilgisayar Mühendisliği (İngilizce)	17/06/2015
Lise	Şehit Oğuzhan Yaşar Anadolu Lisesi	12/06/2009

İş Deneyimi

Yıl	Yer	Görev
2018-Halen	TÜBİTAK	Bilimsel Programlar Uzman Yardımcısı
2016-2017	ASELSAN	Bilgisayar Mühendisi

Yabancı Dil

İngilizce

Tezden Üretilen Yayın

1. Gürbüz, T. (2021). Metin Madenciliği Yöntemi ile Araştırma Makalesi Sınıflandırması. 5.Uluslararası Akademik Araştırmalar Kongresi (ICAR-International Congress of Academic Research), Bildiri Özet Kitabı, 190-191.

GAZİLİ OLMAK AYRICALIKTIR…