T.C.
FIRAT ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ

YAPAY ZEKA TABANLI YAŞ ODAKLI KİŞİSEL EGZERSİZ PLANLAMASI

Diğdem ORHAN

Tez Danışmanı
Prof. Dr. Ahmet Bedri ÖZER

BİTİRME TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

ELAZIĞ
2024

T.C.
FIRAT ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ

YAPAY ZEKA TABANLI YAŞ ODAKLI KİŞİSEL EGZERSİZ PLANLAMASI

Diğdem ORHAN

BİTİRME TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

Bu bitirme tezi … /... /2024 tarihinde aşağıda belirtilen jüri tarafından oybirliği/oyçokluğu ile başarılı/başarısız olarak değerlendirilmiştir.

Danışman	Üye
Prof. Dr. Ahmet Bedri ÖZER	Dr. Öğretim Üyesi Ertan BÜTÜN
Üye	Üye
Doç. Dr. Ebubekir ERDEM	Dr. Öğretim Üyesi Gülşah KARADUMAN

ÖZGÜNLÜK BİLDİRİMİ

Bu çalışmada, başka kaynaklardan yapılan tüm alıntıların, ilgili kaynaklar referans gösterilerek açıkça belirtildiğini, alıntılar dışındaki bölümlerin, özellikle projenin ana konusunu oluşturan teorik çalışmaların ve yazılım/donanımın bizim tarafımızdan yapıldığını bildiririz.

Fırat Üniversitesi
Bilgisayar Mühendisliği
23119 ELAZIĞ

…./…/…

Diğdem ORHAN

BENZERLİK BİLDİRİMİ

TEŞEKKÜRLER

Bu çalışmam, Fırat Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü'nde, Sayın Prof. Dr. Ahmet Bedri ÖZER'in yönlendirmesi ve gözetimi altında hazırlanmıştır. Projemin hazırlaması sürecinde bilgi, görüş ve eleştirilerinden yararlandığım hocam Sayın Ahmet Bedri ÖZER'e en içten duygularımla teşekkür ederim.

İÇİNDEKİLER

ÖZGÜNLÜK BİLDİRİMİ ..... **II**

BENZERLİK BİLDİRİMİ ..... **III**

TEŞEKKÜRLER ..... **IV**

ŞEKİLLER LİSTESİ ..... **VII**

TABLOLAR LİSTESİ ..... **IX**

SİMGELER LİSTESİ ..... **X**

KISALTMALAR LİSTESİ ..... **XI**

ÖZET ..... **XII**

ABSTRACT ..... **XIII**

1.GİRİŞ **1**

2.TEORİK ALTYAPI **2**

2.1 Python 2

2.1.1 Kullanım Alanları 2

2.1.2 Python'un Karakteristiği 2

2.2 Flask 3

2.3 Yapay Zeka ve Makine Öğrenimi 4

2.3.1 Makine Öğrenimi 4

2.3.2 Doğal Dil İşleme 6

2.3.3 Metin Sınıflandırma 9

2.3.3.1 Metin Sınıflandırma İşleminin Özellikleri 10

2.3.4 Destek Vektör Makineleri (SVM) 11

2.3.4.1 Destek Vekör Makineleri ile Sınıflandırma 11

2.3.4.1.1 Doğrusal Ayrılabilen Veriler İçin DVM 12

2.3.4.1.1 Doğrusal Ayrılamayan Veriler İçin DVM 14

3.SİSTEMİN ÇALIŞMA ŞEKLİ VE TEKNİK DETAYI **18**

3.1 Veri Setini Oluşturma İşlemi 18
3.2 Text-Classification İşlemi 21
3.3 Web Arayüzü Kullanabilmek İçin Flask 25
3.4 Web Arayüzü Tasarımı 29
3.5 Projenin Çalıştırılması 30
4.SONUÇ 33
5.KAYNAKÇA 34
6.ÖZGEÇMİŞ 36

ŞEKİLLER LİSTESİ

Şekil 2.1.2.1 Python'un Özellikleri.....3

Şekil 2.3.1.1 Makine Öğrenmesi ile Metin Sınıflandırma 4

Şekil 2.3.2.1 Doğal Dil İşleme Aşamaları 6

Şekil 2.3.4.1.1.1(a) İki Sınıflı Problem İçin Hiperdüzlem 12

Şekil 2.3.4.1.1.1(b) Optimum Hiperdüzlem ve Destek Vektörleri 12

Şekil 2.3.4.1.1.2 Hiperdüzlemin Doğrusal Olarak Ayrılabilen Veri Setleri İçin Belirlenmesi ...13

Şekil 2.3.4.1.2.1(a) Doğrusal Ayrılamayan Veri Seti 14

Şekil 2.3.4.1.2.1(b) Doğrusal Ayrılamayan Veri Seti İçin Hiperdüzlem.....14

Şekil 2.3.4.1.2.2 Veriyi Kernel Fonksiyonu İle Daha Yüksek Boyuta Dönüştürme 15

Şekil 3.1.1 Kullanıcıdan Parametre Alma İşlemi 19

Şekil 3.1.2 Veri Setini Bir Diziye Aktarma ve CSV Dosyasına Çevirme 20

Şekil 3.2.1 Uygun Kütüphanelerin Import Edilme Aşaması 21

Şekil 3.2.2 Modelin Eklenmesi ve Veri Setlerinin Yüklenmesi 22

Şekil 3.2.3 Metinlerin Sayısal Dönüşümü ve Model Eğitimi 23

Şekil 3.2.4 Modelde Tahmin Yapma ve Doğruluk Oranı Hesaplama 24

Şekil 3.2.5 Kullanıcıdan Veri Girişi Alma ve Sınıflandırma İşlemi 24

Şekil 3.3.1 Uygun Kütüphanelerin Import Edilmesi İşlemi 25

Şekil 3.3.2 Veri Setinin Yüklenmesi ve Metinlerin Sayısal Özelliklere Dönüştürülmesi 26

Şekil 3.3.3 Model Eğitimi ve Yönlendirme 27

Şekil 3.3.4 Yönlendirme Yapılan Fonksiyon ve İşlevleri 28

Şekil 3.4.1 Web Arayüzü 28

Şekil 3.4.2 Web Arayüz Kodları 29

Şekil 3.5.1 Modelin Konsolda Çalıştırılması 30

Şekil 3.5.2 Komut Ekranında Yapılacak İşlemler 30

Şekil 3.5.3 Web Arayüzü Genel Görünüm 31

Şekil 3.5.4 Sınıflandırma İşlemi 31

TABLOLAR LİSTESİ

Tablo 2.3.4.1.2.1 Destek Vektör Makinelerinde Kullanılan Temel Kernel Fonksiyonları Ve Parametreleri 17

Tablo 2.3.4.1.2.2 Destek Vektör Makineleri Çekirdek Fonksiyonları ..... .17

SİMGELER LİSTESİ

$\in$: Elemanıdır işareti

$\sum$: Toplam işareti

$\mathbf{W}$: Ağırlık Vektörü

$\boldsymbol{\xi i}$: Slack Değişkeni

∞ : Sonsuz

ϕ : Fi

σ : Sigma

$\boldsymbol{\omega}$: Omega

KISALTMALAR LİSTESİ

AI : Artifical Intelligence (Yapay Zeka)
CL : Computational Linguistics (Hesaplamalı Dilbilim)
CSV : Comma Seperated Values (Virgülle Ayrılmış Değerler)
DB : Database (Veritabanı)
DVM : Destek Vektör Makineleri
HCI : Human-Computer Interaction (İnsan-Bilgisayar Etkileşimi)
HTML : Hyper Text Markup Language (Hiper Metin İşaretleme Dili)
HTTP : Hyper-Text Transfer Protocol (Hiper-Metin Transfer Protokolü)
ML : Machine Learning (Makine Öğrenimi)
NL : Natural Language (Doğal Dil)
NLP : Natural Language Processing (Doğal Dil İşleme)
PUK : Pearson VII Kernel Function (Perason VII Çekirdek Fonksiyonu)
SQL : Structured Query Language (Yapılandırılmış Sorgu Dili)
SVC : Support Vector Classification (Destek Vektör Sınıflandırması)
SVM : Support Vector Machines (Destek Vektör Makineleri)
URL : Uniform Resource Locator (Birleşik Kaynak Konum Belirleyici)
WSGI : Web Server Gateway Interface (Web Sunucusu Ağ Geçidi Arayüzü)
XSS : Cross-Site Scripting (Siteler Arası Komut Dosyası)

ÖZET

Bu tez çalışmasında, bireylerin yaşları baz alınarak egzersiz planlaması hazırlayan bir yapay zeka modeli ortaya konulmaktadır. Bu bağlamda yaş, kilo, cinsiyet, egzersiz geçmişi, kalp ve solunum yolu rahatsızlığı geçmişi parametreleri girilerek bireye özel egzersiz planı sunan bir metin sınıflandırma modeli tasarımı yapılmıştır. Makine öğrenimi yöntemlerinden biri olan ve doğal dil işleme sürecinin bir parçası olarak metin tabanlı sınıflandırma yapmak için destek vektör makinelerinin (SVM) kullanımı tercih edilmiştir. Basitlik, optimizasyon ve yüksek performans açısından tercih edilen lineer SVM yöntemi, modelin hızlı bir şekilde eğitimini tamamlayıp yüksek seviyede doğruluk oranı vermesine katkı sağlamıştır. Geliştirilen modelin ana hedefi, yapay zeka ve spor alanlarını birbirine entegre ederek yaş odaklı kişisel egzersiz planlamasının uygulanabilirliğini ortaya koymaktadır. Tez çalışması kapsamında hazırlanmış olan bu modelin gelecekte daha büyük veri setleriyle genişletilerek doğruluk oranının arttırılması mümkündür. Sonuç olarak bu çalışma, yapay zeka ile kişiselleştirilmiş bir egzersiz planlaması ortaya koymakla beraber gelecekte bu alandaki çalışmalara zemin hazırlamaktadır.

Anahtar Kelimeler: Yapay zeka, makine öğrenimi, destek vektör makineleri, metin sınıflandırma, spor ve yapay zeka entegrasyonu.

ABSTRACT

In this thesis, an artificial intelligence model that prepares exercise plans based on the age of individuals is presented. In this context, a text classification model that provides an individualized exercise plan by entering age, weight, gender, exercise history, cardiac and respiratory disease history parameters was designed. The use of support vector machines (SVM), one of the machine learning methods, was preferred for text-based classification as part of the natural language processing process. The linear SVM method, which is preferred in terms of simplicity, optimization and high performance, contributed to the rapid training of the model and high accuracy. The main goal of the developed model is to demonstrate the feasibility of age-based personal exercise planning by integrating artificial intelligence and sports fields. It is possible to increase the accuracy rate of this model prepared within the scope of this thesis study by expanding it with larger data sets in the future. As a result, this study provides a personalized exercise planning with artificial intelligence and paves the way for future studies in this field.

Keywords: Artificial intelligence, machine learning, support vector machines, text classification, integration of sports and artificial intelligence.

1. GİRİŞ

Günümüzde sağlıklı yaşam ve fitness, bireylerin yaşam kalitesini yükseltmede önemli bir rol üstlenmektedir. Modern yaşamdaki hareketsizlik ve kötü davranışlar, çeşitli sağlık sorunlarına neden olmakta ve bu durum da kişisel sağlık yönetiminin önemini vurgulamaktadır. Egzersiz, sağlıklı bir yaşam sürmenin hayati bir bileşenidir ve her bireyin fiziksel özelliklerine ve gereksinimlerine göre özel bir yaklaşım gerektirir. Bu bağlamda, kişiye özel fitness planları oluşturmak kritik önem taşımaktadır.

Yapay zeka teknolojileri son yıllarda sağlık ve fitness alanında önemli ilerlemeler kaydetmiştir. Bu teknolojiler, büyük veri setlerinden öğrenme yetenekleri ile bireylerin özel gereksinimlerine ve hedeflerine daha doğru ve etkili yanıtlar verebilmektedir. Bu çalışma kapsamında, bireylerin yaşlarına göre sınıflandırılmış egzersiz rutinleri oluşturmak için yapay zeka algoritmaları kullanılacaktır.

Bireylerin fiziksel aktiviteye verdikleri tepkiler ve ihtiyaçları yaşlarından büyük ölçüde etkilenir. Genç insanlar tipik olarak daha yorucu egzersiz rutinleri talep ederken, yaşlı insanlar daha düşük yoğunluklu, eklem dostu egzersizlerden faydalanabilir. Mevcut egzersiz planlama stratejileri geniş kapsamlı ve kişisel olmamaktadır. Bu durum, kişilerin kendi gereksinimlerine ve isteklerine uygun olmayan egzersiz rutinleriyle karşılaşmalarına sebep olmaktadır. Bu da heves eksikliğine ve egzersiz rutinine bağlılığın zayıflamasına zemin hazırlayabilmektedir.

Çalışmanın öncelikli hedefi, her yaştan insanın fiziksel yeteneklerine ve taleplerine uygun, kişiye özel egzersiz rutinleri geliştirmektir. Bu bağlamda yapay zekâ algoritmaları; kullanıcıların yaş, kilo, cinsiyet ve egzersiz geçmişlerine göre en uygun egzersiz planlarını ortaya koyacaktır. Kullanıcılar kendileri için en uygun egzersiz programına erişerek sağlıklı yaşam hedeflerine daha başarılı bir biçimde ulaşabileceklerdir.

Bu bağlamda tez çalışması, yapay zeka teknolojilerinin egzersiz planlamasında nasıl kullanıldığını ve bu tekniklerin bireyselleştirilmiş çözümler sunmadaki faydalarını derinlemesine inceleyecektir. Ayrıca projenin metodolojisi, veri toplama ve analitik yöntemleri, sonuçlar ve bunların değerlendirilmesiyle birlikte derinlemesine açıklanacaktır.

Bir sonraki bölümde projenin altyapısı ve teknolojileri detaylandırılmaktadır. Daha sonra modelin nasıl uygulandığı özetlenmektedir. Son bölümde ise sonuç bölümü yer almaktadır.

2. ALTYAPI

2.1 Python

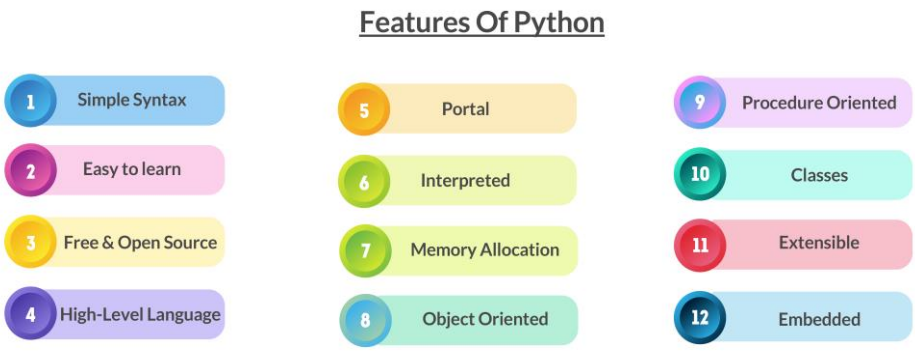
Python, diğer programlama dillerine oranla öğrenilmesi kolay ve yüksek seviyeli bir programlama dilidir. 1991 yılında Guido Van Rossum tarafından yazılmıştır. Python, çok sayıda görevi yerine getirmek için yalnızca birkaç satır kod kullanır ve bu da onu diğer programlama dillerinden ayıran en temel özellik olarak karşımıza çıkar.

2.1.1 Kullanım Alanları

Python; çevrimiçi uygulamalarda, yazılım geliştirmede, veri araştırmalarında ve makine öğreniminde (ML) yaygın olarak kullanılan bir programlama dilidir. Python; verimli olması, öğrenilmesinin kolay olması ve çeşitli platformlarda çalışabilmesi nedeniyle geliştiriciler arasında popülerdir. Python yazılımını indirmek ücretsizdir, çeşitli sistemlerle iyi çalışır ve geliştirmeyi hızlandırır.

2.1.2 Python'un Karakteristiği

Python, gerçek dünya programlaması için uygun ve iyi tasarlanmış bir dildir. Çok çeşitli uygulamalarda kullanılabilen bir yorumlayıcıya sahip, yüksek seviyeli, dinamik, nesne yönelimli, genel amaçlı bir programlama dilidir. Python'un anlaşılması ve kullanılması basit olacak şekilde tasarlanmıştır. Günümüzde nispeten kullanıcı dostu ve acemi dostu bir dil olarak kabul edilmektedir. En popüler giriş dili olan Java'nın yerini alarak yeni başlayan dostu bir dil şeklinde popülerliğini artırmıştır. Dinamik yazım özelliği sayesinde son derece esnektir. Bu dil, yapısal ve nesne yönelimli dahil olmak üzere çeşitli programlama tekniklerini destekler. Python'un esnekliği, diğer programlama dillerinde oluşturulan modüler bileşenleri kullanma yeteneğinden kaynaklanmaktadır [1].



Şekil 2.1.2.1 Python'un Özellikleri

2.2 Flask

Flask, hızlı bir şekilde web uygulaması oluşturmanızı sağlayan bir mikro çerçevedir. Sadece temel işlevleri uygular ve geliştiricilerin uygulama sırasında gerektiğinde özellikler eklemesine olanak tanır [2]. Hafif, WSGI (web server gateway interface) uygulama çatısıdır.

Bu çerçeve, gerektiğinde saf arka uç veya ön uç için kullanılabilir. İlki, etkileşimli bir hata ayıklayıcı, tam bir istek nesnesi, bir uç nokta yönlendirme sistemi, varlık etiketleri, önbellek kontrolleri, tarihler ve çerezlerle başa çıkmak için HTTP yardımcı programları gibi özellikler içerir [3].

Ayrıca yerel geliştirme için iş parçacıklı bir WSGI sunucusu ve HTTP sorgularını taklit etmek için bir test istemcisi içerir. Werkzeug ve Jinja iki temel kütüphanedir. Jinja başka bir Flask bağımlılığıdır. Tamamen işlevsel bir şablon motorudur. Sandboxed yürütme, güçlü XSS önleme, şablon kalıtımı, basit hata ayıklama ve esnek sözdizimi sayısız özellikten sadece birkaçıdır. Ek olarak, HTML şablonuna yazılan kod Python kodu olarak derlenir.

Flask genellikle bir prototip oluşturma çerçevesi olarak kabul edilir, bu nedenle bir veritabanı soyutlama katmanının yanı sıra doğrulama ve güvenlikten yoksundur. Flask böylece uygulayıcının ihtiyaçları eklemesi için tam esneklik sağlamıştır.

Flask çerçeveleri için uzantılar mevcuttur. Kütüphaneler, bunlarla sınırlı olmamak üzere, sunucu için gunicorn, veritabanı için SQLAlchemy, veritabanı taşıma yönetimi için Alembic, asenkron görev koşucusu için celery ve Redis, form doğrulama için Flask-WTF formu ve web isteklerini hız sınırlaması için Flask-limiter'ı içerir. Flask, Python 3 ve üstü

ile uyumludur; ayrıca PyPy'ye dahildir ve Python'un resmi paket yöneticisi olan pip kullanılarak kolayca kurulabilir [4].

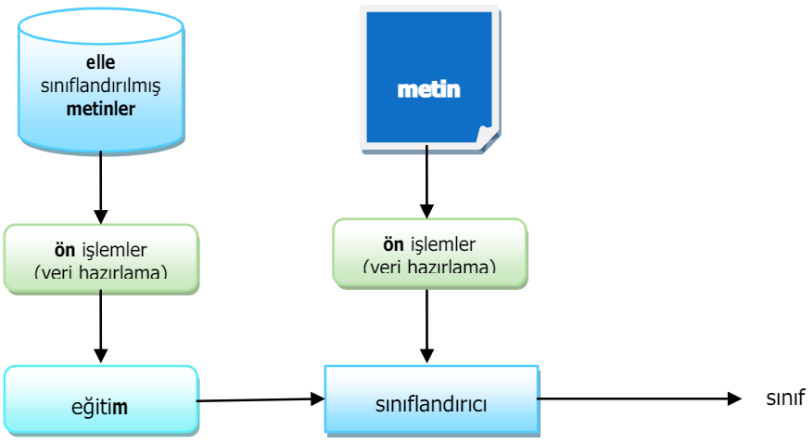
2.3 Yapay Zeka ve Makine Öğrenimi

Yapay zeka, doğada bulunan herhangi bir rasyonel davranışı taklit etmeye çalışan bir yöntemler bütünüdür ve çok çeşitli yaklaşım ve algoritmaları içerir [5]. Makine öğrenimi ve derin öğrenme, yapay zekaya yönelik iki farklı yaklaşımdır. Makine Öğrenimi ile Derin Öğrenme arasındaki en önemli fark, derin öğrenmenin kendi başına özellikler çıkarabilmesidir.

2.3.1 Makine Öğrenimi

Bir performans ölçüsünü optimize etmek için örnek verileri ve geçmiş deneyimleri kullanan bilgisayar programlama" makine öğreniminin tanımıdır [6].

Bir sınıflandırıcı, makine öğreniminin merkezi bileşenidir. Sınıf etiketleri insan eliyle belirlenmiş bir dizi belge üzerinde bir uzman tarafından eğitilir ve belgelerin hangi sınıfa ait olduğunu gösteren özellikleri otomatik olarak alır. Bu sınıflandırıcı, kendisine yeni bir belge sunulduğunda belgenin sınıfını belirlemek için öğrendiği özellikleri kullanır. "Denetimli sınıflandırma" makine öğrenimi jargonunda sınıflandırma konusudur. Aşağıdaki şekil, makine öğreniminin metin sınıflandırma için nasıl kullanıldığını göstermektedir.



Şekil 2.3.1.1 Makine Öğrenmesi ile Metin Sınıflandırma

Hiç şüphesiz, manuel olarak sınıflandırılmış eğitim seti, makine öğrenimi tekniklerinin uygulanmasındaki en önemli bileşendir. Doğru parametreleri bulmak ve en iyi performansı gösterecek sınıflandırma yöntemine karar vermek de aynı derecede önemli adımlardır. Bunu başarmak için manuel olarak sınıflandırılan veriler genellikle iki gruba ayrılır: Eğitim seti ve test seti. Bu kümeleri dengelemeye gerek yoktur. Test seti sınıflandırıcının performansını ölçmek için kullanılır ve eğitim seti algoritmaların öğrenmesini sağlar. Performansı ölçmek için kullanılan test seti, eğitim setindeki metinlerden hiçbirini içermemelidir. Adil bir değerlendirme yapılabilmesi için test setinin eğitim seti verilerini içermemesi gerektiğinden, sınıflandırıcı eğitim setindeki veriler kullanılarak oluşturulur. Bu şemada sınıflandırıcının performansı, sınıflandırma sonuçlarının daha önce görmediği metinlerin manuel sınıflandırma sonuçlarıyla karşılaştırılmasıyla belirlenir. Sınıflandırıcı daha önce görmediği test kümesindeki metinleri sınıflandırır.

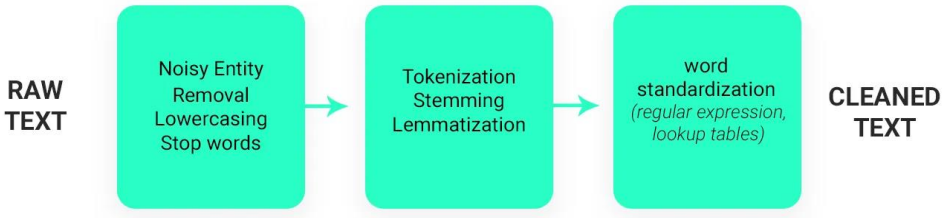
Genellikle eğitim ve test setleri rastgele oluşturulur. Yeterince temsil edici olması ve sınıflandırıcının performansını doğru bir şekilde ölçebilmesi için hem eğitim seti hem de test seti tipik metinler içermelidir. Bunu başarmak için k-kat çapraz doğrulama yaklaşımı, rastgele bölümleme ile eğit ve test prosedürlerinin yerine sıklıkla kullanılır. Bu yaklaşımı kullanarak, elle etiketlenmiş veri kümesinin tamamı, hiçbir öğeyi paylaşmayan k benzersiz gruba ayrılır. İlk aşamada ilk grup test kümesi olarak belirlenir ve kalan k-1 grubu eğitim kümesi olarak kullanılır. Bu k-1 gruplamayı içeren eğitim seti kullanılarak sınıflandırma algoritmasının eğitilmesini takiben test seti üzerinde bir performans hesaplanır. İkinci grup bir sonraki aşamada test kümesi olarak seçilir. Kalan k-1 grup eğitim kümesi olarak kullanılarak algoritma eğitilir ve test kümesi üzerindeki performansı hesaplanır. Bu çalışma mantığına göre k adet eğitim ve test adımından sonra elde edilen başarıların ortalaması alınarak nihai başarı değeri hesaplanır. Tipik olarak k=10 veya k=5 değerleri kullanılır. Parametreler genellikle eğitim seti üzerinde eğitilmiş algoritmalar tarafından kullanılır. Bu ayarları optimize etmek için, bazı programlar eğitim ve test setlerine ek olarak bir doğrulama seti (bekletme seti veya doğrulama seti olarak da bilinir) hazırlar [7].

2.3.2 Doğal Dil İşleme

Doğal dil işleme (NLP), yapay zeka yazılımları kullanarak konuşma veya metin gibi dilsel işlevleri değerlendirerek sonuçlar üreten bir çalışma alanıdır. Semantik Ağ ve Ayrıştırma Ağacı teknikleri yapay zeka ve doğal dil işleme metodolojilerini birleştirir. Anlamsal ağ, bir cümlenin anlamsal karşılığını belirlemeye çalışır [8]. Doğal dil işlemenin birincil amacı, makinenin dili anlamasını ve uygun şekilde tepki verebilmesini sağlamaktır.
Yaygın olarak bilindiği üzere diller iki kategoriye ayrılır: Makine dili ve insanlar tarafından kullanılan doğal dil. Bilgisayarların insan dillerini anlayabilmesi ve konuşabilmesi için doğal dil işleme bilimi gerekir. Bu, özünde, bilgisayarların doğal dilleri yorumlama yöntemidir. Bu, hesaplamalı dilbilim (CL) veya insan bilgisayar etkileşimi (HCI) kapsamına girer. Bu durumda, dil bilgisayar tarafından işlenir.

Veri bilimi ve doğal dil işleme arasındaki ilişki göz önüne alındığında, aslında bu konuyu metin işleme ve metin madenciliği altında ele almak gerekir çünkü veri bilimciler genellikle bununla ilgilenirler. Bunun nedeni, veri kaynaklarının metin tabanlı olması durumunda, doğal dil işlemenin ilk analiz ve işleme için çok önemli olmasıdır.

Doğal dil işleme aşamaları aşağıdaki gibi gösterilebilir:



Şekil 2.3.2.1 Doğal Dil İşleme Aşamaları

a. **Noisy Entity Removal:** Noktalama ve anlamsız kelimeler çıkartılır.
b. **Lowercasing:** Küçük harfe dönüştürme işlemidir.
c. **Stop Words:** Metinde tekrar eden kelimeler çıkartılır.
d. **Tokenization:** Metnin parçalara ayrılması işlemini gerçekleştirir.
e. **Stemming:** Metindeki kelime kökleri ayrıştırılır.
f. **Lemmatization:** Kelimenin morfolojik kökenine inilir.
g. **Word Standardization:** Metni bir standarda indirger.

Doğal dil işleme bölümü girdilere dayanır. Konuşma diline (konuşma işleme) ve yazılı metinlere (metin işleme) dayananlar şeklinde tipik olarak iki yerden gelir. Sözlü bir soruyu yanıtlamak ya da yazılı bir materyali deşifre etmekle karşılaştırılabilir. Doğal dil işleme üzerine yapılan çalışmalar, sesin büyük ölçüde işlenemez olmasına rağmen, öncelikle yazılı sesli ifadelerden oluştuğunu ve yazılı metinlerin de sürece dahil edildiğini göstermektedir. Doğal dil işleme araştırma seviyelerinin dört ana bileşeni vardır: Anlamsal, pragmatik, morfolojik-sözlüksel ve sözdizimsel konuşma.

Sözcük anlamının incelenmesi sözcük biliminin odak noktasıdır. Bir kelimenin kökünün aldığı son eklere ve üstlendiği anlama bakarak kelimenin ne anlama geldiğini açıklığa kavuşturmaya yardımcı olur. Kelimelerin cümle içindeki dizilişi söz dizimi olarak bilinir. Bir cümle içindeki kelimelerin düzenini ele alır. Anlamsal analiz cümlelerin anlamsal içeriğini inceler. Çünkü bir bilgisayarın doğal dili doğru bir şekilde yorumlayabilmesi için öncelikle bir cümleyi doğru bir şekilde anlaması gerekir. Konuşmada kullanılan kelimelerin dağarcığı ve anlamları pragmatik söylemin odak noktasıdır. Bilgisayarın ifadelere uygun kelime dağarcığıyla yanıt verebilmesi için öncelikle konuşmayı anlaması gerekir. Üretme, doğal dil işlemenin ek bir yönüdür. Bu durumda, bir dil oluşturmak, onu anlamakla aynı miktarda zaman gerektirir.

Doğal dil işlemenin uygulama alanları aşağıdaki gibi sınıflandırılabilir:

a. **Soru Cevaplama (Question Answering):** Bu, doğal dil işleme ile ilgili ilk konular arasındadır. Sorulacak önemli soruları ve bunlara nasıl cevap verileceğini belirlemek için belirli bir kitabı incelemek veya sorulan soruların cevaplarını ortaya çıkarmak için metni kullanmak tercih edilir. Bu sorunun çözümü aynı zamanda arama motoru sorununun, yani sesli arama yetenekleri sorununun da çözümüne yol açacaktır.

b. **Otomatik Tercüme (Machine Translation):** Makine, metni bir dilden başka bir dile çevirme işlemini gerçekleştirir.
c. **Bilginin Getirilmesi (Information Retrieval):** Metin kaynakları arasında istenilen bilgi aranır, bulunur ve getirilir.
d. **Kelime işleme (Word Processing (Grammer Checking, Spell Checking, Spell Correction, Find/Replace)):** Kelime işleme, çeşitli alanlarda kullanılan çeşitli faaliyetleri içeren yaygın olarak kullanılan bir prosedürdür. Metindeki yazım hataları dilbilgisi denetimi kullanılarak kontrol edilir. Alternatif olarak, bir kelimenin yazımını doğrulamak için yazım denetimi ve yazım düzeltme kullanılabilir.
e. **Doğal Dil İşleme-Veri Tabanı Arayüzü (NL/DB interface):** Gelen soruyu sorgu cümlesine çevirerek veritabanına uyarlar.
f. **İnternet Arama Motorları (Web Search):** Arama motorlarını kullanarak arama işlemini gerçekleştirir ve sorulara uygun cevapları bulur.
g. **Anti-Intelligence (Al) Bots (Siri):** Bir makine konuşulan bir soruya yanıt olarak tekrar konuştuğunda ortaya çıkar. Bu, modern teknolojilerde yaygın olan ve istenen sorguların cevaplarını arayan ve bulan akıllı sistemlerde gözlemlenen senaryodur.
h. **Finance, Chat Rooms, Telefon Cevaplama:** Sohbet odaları, telefon cevaplama hizmetleri ve finansal raporlama oldukça yaygın olarak kullanılır. Aramaya karşılık diğer uçta sorulara cevap veren ve sorulan soruya karşılık yönlendirme yapan bir makinenin veya sohbet odalarında sohbet konuşmalarına reaksiyon veren bir makinenin bulunması şeklinde ele alınabilir.
i. **Bilgi Deposu Olarak İnternet (Internet as a Repository):** En büyük ve en uygun bilgi kaynağının internet olduğu bilinmektedir. Milyonlarca insan her gün kişisel sayfalarına, bloglarına ve web sitelerine içerik eklemektedir. İnternet ağındaki bilgisayarlar bu ortak bilgiyi anlayabilir, modelleyebilir ve kümeleyebilirse doğal dil işleme diğer insanların bu bilgiye erişimini çok daha fazla kolaylaştırmış olur.
j. **Argümanların Birleştirilmesi (Argument Aggregation):** Sosyal ağlardaki tartışmalar sonucunda bazı argümanlar ortaya çıkar. Ortaya çıkan bu argümanlar incelenmeli, gereksiz veya boş argümanlar elenmeli

ve kalan argümanlar tutarlı bir argüman oluşturmak maksadıyla ilgili verileri üretmek için kullanılmalıdır.

k. **Metin Özetleme (Text Summarization):** İnternetteki bir milyondan fazla kaynak dikkate alınarak ve bu sayfalar kullanılarak yapılan bir sonucun keşfi ve özetidir.

l. **Metin İşleme (Text Processing)(Categorization, Clustering):** Metin üzerinde kategorizasyon, kümeleme ve segmentasyon işlemlerinin yapılmasıdır [9].

2.3.3 Metin Sınıflandırma

Elektronik ortamda oluşturulan belgelerin sayısındaki artışın en önemli nedenleri; internetin ortaya çıkışı, geniş kitleler tarafından kolaylıkla erişilebilir ve kullanılabilir olması, kişisel bilgisayarların fiyatlarının düşmesi ve toplumun bilgisayar kullanımının artmasıdır. Çok sayıda bilgi varlığının getirdiği çoklu faydaların yanı sıra, ortaya çıkan bazı sorunların da ele alınması gerekmektedir. Bilgiye erişim basit olmalıdır. Bu bağlamda gündeme gelen konulardan biri de elektronik ortamdaki metinlerin sınıflandırılmasıdır. Metin sınıflandırma problemi, en geniş anlamıyla bir metnin belirtilen sınıflandırmalardan hangisine ait olduğunun belirlenmesidir. Metin sınıflandırma aynı zamanda belge sınıflandırma ve metin kategorizasyon olarak da bilinir. İngilizce'de bu problem genellikle “text classification”, “document classification”, “text categorization” veya “document categorization” olarak adlandırılır.

Metin tabanlı verileri sınıflandırmak için makine öğrenimi teknikleri ve doğal dil işleme kullanılır. İnsan dillerinin veya “doğal dilin” bilgisayarlarla nasıl etkileşime girdiğinin ve bilgisayarların doğal dil metinlerini ve konuşmalarını anlamak, yorumlamak ve değiştirmek için nasıl kullanılabileceğinin incelenmesi, dilbilim, yapay zeka ve bilgisayar biliminin bir dalı olan “doğal dil işleme” olarak bilinir [10].

2.3.3.1 Metin Sınıflandırma İşlemlerinin Özellikleri

a. Tek Etiketli/Çok Etiketli Sınıflandırma: Çeşitli uygulamaların farklı ihtiyaçları vardır ve bu da farklı sınıflandırma yöntemleri gerektirir. Bu ihtiyaçlardan biri de tek ya da çok etiketli sınıflandırma sonuçları üretmektir. Tek etiketli sınıflandırma, her bir metin için ilgili olduğu tespit edilen mevcut m sınıftan yalnızca birinin seçildiği ve bu sınıfın etiketinin metne "yapıştırıldığı" veya "atandığı" bir uygulamadır. Çok etiketli kategorizasyon ise, bir metinle ilgili olduğu düşünülen m sınıftan bir veya daha fazla etiketin seçilmesini gerektirir.

b. Sınıf Odaklı/Belge Odaklı Sınıflandırma: Normal şartlar altında, sınıflar (S kümesi) ve kategorize edilecek metinler (B kümesi) sınıflandırma işleminin başlangıcında tamamen bilinmektedir. Ancak bazı uygulamalarda, zaman içinde S veya B kümesine ek sınıflar veya belgeler eklenebilir. Bu gibi durumlarda, mesajları sınıflandırmak için kategorilere veya belgelere dayalı sınıflandırma ilkeleri kullanılmalıdır. Belge odaklı sınıflandırmanın amacı, belirli bir belgeyle bağlantılı olabilecek tüm sınıf etiketlerini bulmaktır. Sınıf odaklı sınıflandırmada amaç, o sınıfa ait olması gereken tüm belgeleri bulmaktır.

c. Sınıflandırmanın Kesinliği: Kesin sınıflandırma, sınıflandırma süreci belirli bir *bi* belgesinin belirli bir *si* sınıfına tahsis edilmesi gerektiğini belirten doğru veya yanlış kesin bir beyan verdiğinde gerçekleşir. Diğer bir yol ise, belirli bir *bi* belgesine atanacak olası sınıfların sıralı bir listesini oluşturmayı içeren sıralama kategorizasyonudur. Bu listedeki sınıflar, kullanılan yönteme göre, belgeyle en bağlantılı olduğu düşünülen sınıftan en az ilgili olan sınıfa doğru sıralanır. Listenin en üstündeki sınıfın belgeyle en alakalı olduğu düşünülse de, yöntem kesin bir yargı sonucu üretmez ve bunun yerine bu sıralı listeyi çıktı olarak döndürür. Bu sıralı liste, S kümesinde tanımlanan tüm sınıfları veya yalnızca en yüksek ilişkili değere sahip N ($N \leq m$) sınıfı içerebilir.

Makine öğrenimi uygulamalarında metin kategorizasyonu çok çeşitli algoritmalar kullanır. En popüler olanları Naive Bayes, karar ağaçları, yapay sinir ağları, örnek tabanlı sınıflandırıcılar, istatistiksel dil modeli tabanlı sınıflandırıcılar ve destek vektör makineleridir [11].

2.3.4 Destek Vektör Makineleri (DVM)

Bu sınıflandırma yaklaşımı, verileri daha yüksek bir boyuta çevirerek ve bir hiper düzlem oluşturarak iki sınıfı bölme öncülünde çalışır. Daha yüksek boyutlu dönüştürme aşaması, değişen niteliklere sahip fonksiyonları kullanır. Bu fonksiyonlar çekirdek fonksiyonları olarak adlandırılır. Kernel fonksiyonlarını kullanmak için kullanıcının matematiksel ifadelerinde belirli parametreleri belirtmesi gerekir.

Bir destek vektör makinesi, verileri en iyi şekilde iki gruba ayıran n boyutlu bir hiper düzlemdir. DVM modelleri yapay sinir ağlarıyla yakından bağlantılıdır. Bir sigmoid çekirdek fonksiyonu kullanan DVM, iki katmanlı, ileri beslemeli bir yapay sinir ağıdır [12].

DVM'nin önemli yönü, veri seti üzerindeki ortalama hata karesini azaltarak oluşturulan ampirik risk minimizasyonu prensibi yerine, istatistiksel öğrenme teorisindeki yapısal risk minimizasyonu temelinde çalışmasıdır. DVM'nin temel kabullerinden biri, eğitim setindeki tüm örneklerin bağımsız ve benzer düzeyde dağılmasıdır [13].

DVM'ler sınıflandırma ve regresyon zorlukları için kullanılabilir. DVM regresyonunun arkasındaki temel fikir, verilen eğitim verilerinin karakteriyle yakından eşleşen ve istatistiksel öğrenme teorisine bağlı kalan doğrusal bir ayırt edici fonksiyon oluşturmaktır. Kernel fonksiyonları, sınıflandırma gibi, doğrusal olmayan durumları ele almak için regresyonda kullanılır [14].

2.3.4.1 Destek Vektör Makineleri İle Sınıflandırma

Destek Vektör Makineleri (DVM), istatistiksel öğrenmeye dayalı kontrollü bir sınıflandırma sistemidir. DVM'nin matematiksel teknikleri ilk olarak iki sınıflı doğrusal verileri sınıflandırmak için oluşturulmuş, ancak daha sonra çok sınıflı ve doğrusal olmayan verileri sınıflandırmak için genelleştirilmiştir. DVM'nin çalışma prensibi, iki sınıfı birbirinden ayırabilecek en uygun karar fonksiyonunun tahmin edilmesine veya iki sınıfı en uygun şekilde ayırabilecek hiper düzlemin tanımlanmasına dayanır [15].

DVM'ler çeşitli disiplinlerde başarıyla kullanılmıştır ve son yıllarda uzaktan algılamadaki uygulamaları üzerine birçok araştırma yapılmıştır [16].

2.3.4.1.1 Doğrusal Ayrılabilen Veriler İçin DVM

Destek vektör makineleri ile sınıflandırma, örnekleri {-1,+1} şeklinde sınıf etiketleri ile ifade edilen iki sınıfa ayırmak için eğitim verilerine dayalı bir karar işlevi kullanır. Karar fonksiyonu, eğitim verilerini en iyi şekilde ayıran hiper düzlemi keşfetmek için kullanılır.

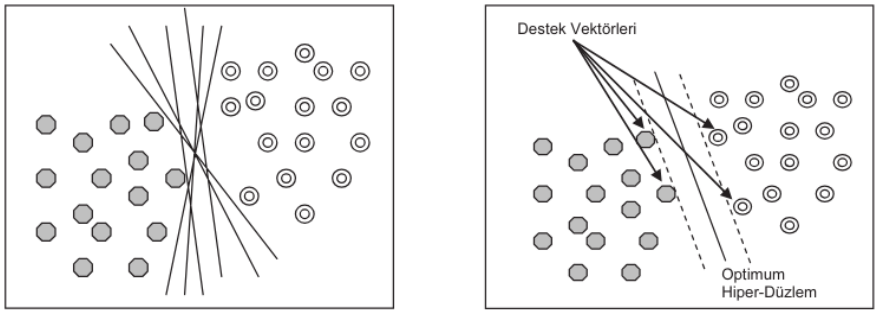
Şekil 2.3.4.1.1.1.1(a)'da gösterildiği gibi, iki veri türünü ayırmak için birden fazla hiper düzlem oluşturulabilir. Ancak DVM'nin amacı, kendisine en yakın noktalar arasındaki mesafeyi optimize eden hiper düzlemi bulmaktır. Şekil 2.3.4.1.1.1.1(b)'de gösterildiği gibi, ideal hiper düzlem sınırı en üst düzeye çıkaran hiper düzlemdir, sınırın genişliğini sınırlayan noktalar ise destek vektörleri olarak adlandırılır. Doğrusal olarak ayrılabilen iki sınıflı bir sınıflandırma probleminde, DVM'yi eğitmek için k örnekten oluşan eğitim verilerinin $\{x_i, y_i\}$, i = 1,.....,k olduğu varsayıldığında, en iyi hiper düzlem için eşitsizlikler aşağıdaki gibidir:

$w \cdot x_i + b \geq +1$ her $y = +1$ için

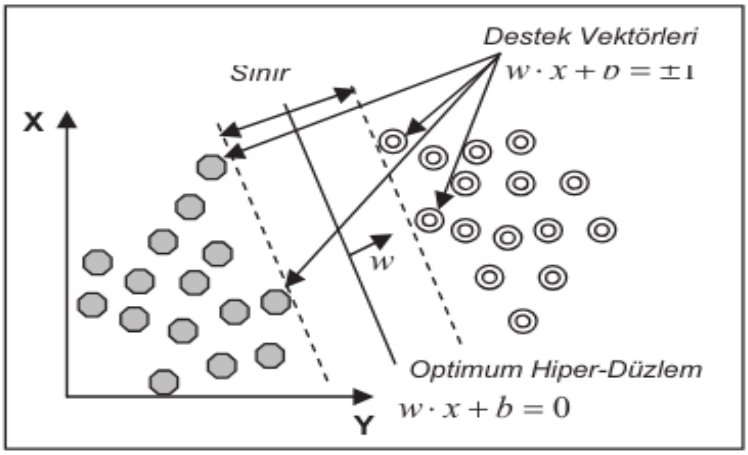
$w \cdot x_i + b \leq +1$ her $y = -1$ için

Burada $x \in R^N$ ve N boyutlu bir uzay, $y \in \{-1, +1\}$ sınıf etiketleri, w ağırlık vektörü (hiper düzlemin normali) ve b eğilim değeridir [17].

İdeal hiper-düzlemi tanımlamak için, ona paralel olan ve sınırlarını oluşturan iki hiper-düzlem tanımlanmalıdır (Şekil 2.3.4.1.1.2). Bu hiper düzlemleri oluşturan noktalar destek vektörleri olarak adlandırılır ve $w \cdot x_i + b = \pm 1$ olarak gösterilir.



Şekil 2.3.4.1.1.1(a): İki Sınıflı Problem İçin Hiperdüzlem, **(b):** Optimum Hiperdüzlem ve Destek Vektörleri



Şekil 2.3.4.1.1.2: Hiperdüzlemin Doğrusal Olarak Ayrılabilen Veri Setleri İçin Belirlenmesi

İdeal hiper düzlemin sınırını maksimize etmek için w azaltılmalıdır. Bu senaryoda en iyi hiper düzlemi belirlemek için aşağıdaki kısıtlı optimizasyon problemi çözülmelidir.

$$\min\left[\frac{1}{2}||w||^2\right]$$

Buna bağlı sınırlamalar ise;

$$y_i(w\ .\ x_i + b) - 1 \geq 0\ ve\ y_i\ \in \{1, -1\}$$

şeklinde ifade edilir [18].

Lagrange denklemleri kullanılarak da bu optimizasyon problemi çözülebilir. Bu işlemin ardından;

$$L(w,b,\alpha) = \frac{1}{2}\left||w|\right|^2 - \sum_{i=1}^{k} \alpha_i \; y_i \; (w \,.\, x_i \; + b) + \sum_{i=1}^{k} \alpha_i$$

eşitliği elde edilir. Bu bağlamda, doğrusal olarak ayrılabilen iki sınıflı bir problem için karar fonksiyonu aşağıdaki gibi formüle edilebilir:

$$f(x) = sign\,(\sum_{i=1}^{k} \lambda_i y_i (x \,.\, \mathrm{x}_i) + b)$$

2.3.4.1.2 Doğrusal Ayrılamayan Veriler İçin DVM

Uydu görüntüsü sınıflandırması gibi birçok konu, doğrusal veri ayrımına izin vermez [Şekil 2.3.4.1.2.1(a)]. Bazı eğitim verilerinin ideal hiper düzlemin karşı tarafında olması sorununu ele almak için pozitif bir sahte değişken (ξ_i) tanımlanır [Şekil 2.3.4.1.2.1(b)]. Sınırın maksimize edilmesi ile yanlış sınıflandırma hatalarının minimize edilmesi arasındaki dengeyi düzenlemek için, C ile temsil edilen ve pozitif değerler alan bir düzenlilik parametresi $0<C<\infty$ tanımlanır [19]. Yapay değişken kullanılarak doğrusal olarak ayrım yapılamayan veriler ve düzenleme parametresi için optimizasyon problemi:

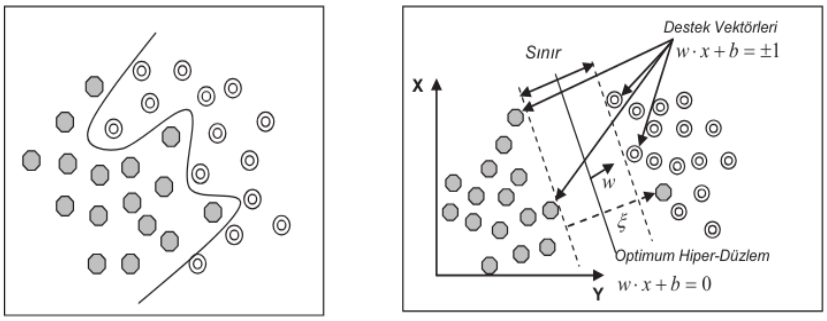
$$\min[\frac{\left||w|\right|^2}{2} + C \,.\, \sum_{i=1}^{e} \xi_i]$$

şeklini alır. Buna bağlı sınırlamalar ise;

$$y_i(w \,.\, \varphi(x_i) + b) - 1 \geq 1 - \xi_i$$

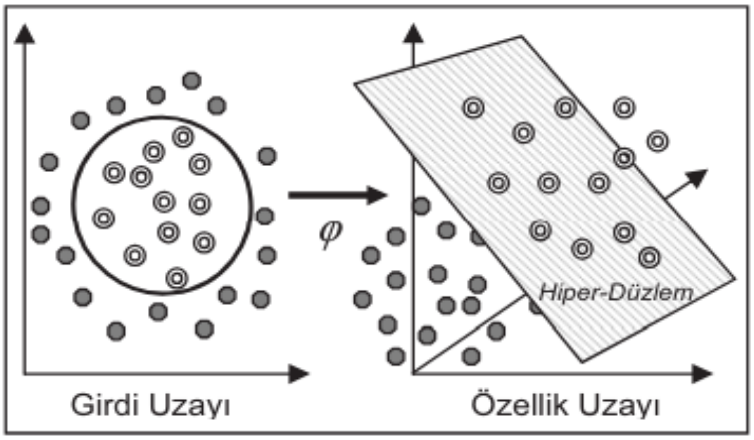
$$\xi_i \geq 0 \; ve \; i = 1, \ldots, N$$

şeklinde ifade edilir.



Şekil 2.3.4.1.2.1(a): Doğrusal Ayrılamayan Veri Seti, **(b):** Doğrusal Ayrılamayan Veri Seti İçin Hiperdüzlem

Yukarıda bahsedilen optimizasyon problemini çözmek için, Şekil 2.3.4.1.2.2'de gösterildiği gibi, giriş uzayında doğrusal olarak ayrılamayan veriler özellik uzayı olarak bilinen yüksek boyutlu bir uzayda gösterilir. Böylece veriler doğrusal olarak bölünebilir ve sınıflar arasındaki hiper düzlem tespit edilebilmektedir.



Şekil 2.3.4.1.2.2: Veriyi Kernel Fonksiyonu İle Daha Yüksek Boyuta Dönüştürme

Destek vektör makineleri doğrusal olmayan dönüşümler yapmak için bir çekirdek fonksiyonu, $K(x_i, x_j)= \phi(x) . \phi(x_j)$ kullanır. Bu, yüksek boyutluluğa sahip verilerin doğrusal olarak ayrılmasını sağlar. Sonuç olarak, kernel fonksiyonu kullanılarak doğrusal olarak ayrılamayan iki sınıflı bir problemi ele almak için karar kuralı aşağıdaki gibi ifade edilebilir:

$$f(x) = sign(\sum_i \alpha_i y_i \varphi(x).\varphi(x_i) + b)$$

Destek vektör makineleri (DVM) ile bir sınıflandırma prosedürü gerçekleştirmek için, çekirdek fonksiyonu ve optimum parametreleri belirlenmelidir.

Tablo 2.3.4.1.2.1, en sık kullanılan polinomlar için formülleri ve parametreleri göstermektedir: radyal temel fonksiyonu, Pearson VII (PUK) fonksiyonu ve normalleştirilmiş polinom çekirdeği. Tablo, kullanıcının her çekirdek fonksiyonu için bazı parametreler belirtmesi gerektiğini göstermektedir. PUK çekirdeğinin hesaplanması gereken iki parametresi varken, diğer fonksiyonlar model oluşturma sırasında sadece bir parametrenin bulunmasını gerektirir ve bu da sınıflandırma için temel teşkil eder.

Kernel fonksiyonları karşılaştırıldığında, polinom ve radyal kerneller daha basit ve anlaşılması daha kolaydır. Teknik olarak basit olmasına rağmen, polinomun derecesini artırmak prosedürü daha zor hale getirir. Bu, işlem süresini önemli ölçüde artırır ve belirli bir noktadan sonra sınıflandırma doğruluğunu düşürür. Çekirdek boyutu (γ) olarak bilinen radyal temel fonksiyon parametresindeki değişikliklerin sınıflandırma performansı üzerinde daha düşük bir etkisi vardır [20].

Normalleştirilmiş polinom fonksiyonu, veri toplama yerine polinom çekirdeğinin matematiksel ifadesini normalleştirmek maksadıyla ileri sürülmüştür [21].

Normalleştirilmiş polinom kerneli, polinom kernelinin genelleştirilmiş bir şekli olarak düşünülebilir. Ancak, PUK çekirdeği diğer çekirdek fonksiyonlarından daha karmaşık bir matematiksel yapıya sahiptir ve Pearson genişliği olarak bilinen iki parametreye (σ, ω) sahiptir. Bu iki faktör sınıflandırma doğruluğunu etkiler ve hangi parametre çiftinin en iyi sonuçları üreteceği önceden bilinmemektedir. Sonuç olarak, en uygun parametre çiftini bulmak PUK çekirdeğini kullanmada kritik bir adımdır.

Kernel Fonksiyonu	Matematiksel İfadesi	Parametre
Polinom Kerneli	$K(x,y)=((x\cdot y)+1)^d$	Polinom derecesi (d)
Normalleştirilmiş Polinom Kerneli	$K(x,y)=\frac{((x\cdot y)+1)^d}{\sqrt{((x\cdot x)+1)^d((y\cdot y)+1)^d}}$	Polinom derecesi (d)
Radyal Tabanlı Fonksiyon Kerneli	$K(x,y)=e^{-\gamma\|(x-x_i)\|^2}$	Kernel boyutu (γ)
Pearson VII (PUK) Kerneli	$\frac{1}{\left[1+\left(\frac{2\cdot\sqrt{\|x-y\|^2\sqrt{2^{(1/\omega)}-1}}}{\sigma}\right)^2\right]^{\omega}}$	Pearson genişliği parametreleri (σ, ω)

Tablo 2.3.4.1.2.1: Destek Vektör Makinelerinde Kullanılan Temel Kernel Fonksiyonları Ve Parametreleri

DVM'nin dört ortak çekirdek fonksiyonu vardır. Bu fonksiyonlar şunlardır:

1. Doğrusal Fonksiyon

2. Polinom Fonksiyon

3. Sigmoidal Fonksiyon

4. Radyal Taban Fonksiyon

	Formula	Parameters	Merits
Linear	$\mathrm{K}(\mathrm{x}, x_i) = \mathrm{x}\cdot x_i$	/	It is only used when the sample is separable in low dimensional space.
Polynomial	$\mathrm{K}(\mathrm{x}, x_i) = [\gamma * (\mathrm{x}\cdot x_i) + coef]^d$	γ, *coef*, d	global kernels
RBF	$\mathrm{K}(\mathrm{x}, x_i) = \exp(-\gamma * \|\|x - x_i\|\|^2)$	γ.	good local performance
Sigmoid	$\mathrm{K}(\mathrm{x}, x_i) = \tanh(\gamma(\mathrm{x}\cdot x_i) + coef)$	γ, *coef*	needs to meet certain conditions

Tablo 2.3.4.1.2.2: Destek Vektör Makineleri Çekirdek Fonksiyonları

3. SİSTEMİN ÇALIŞMA ŞEKLİ VE TEKNİK DETAYI

Bu proje; kullanıcıdan alınan yaş, kilo, boy, egzersiz geçmişi, geçmiş kalp rahatsızlığı ve geçmiş solunum rahatsızlığı parametrelerine göre o kullanıcıya en uygun egzersiz biçimini öneren bir modelden meydana gelmiştir. Yaş kitlesini baz alarak sınıflandırma yapan bu model, 5 adet kategoriye sahiptir. Kullanıcının yaş aralığına göre hangi kategoriye ait olduğunu sınıflandırır.

3.1 Veri Setini Oluşturma İşlemi

Veri seti manuel olarak Python kodu ile oluşturulmuştur. Veri setinde 5 adet kategori bulunmaktadır ve her biri bir seviyeyi içermektedir. Sınıflandırma için eğitim ve test veri seti olmak üzere iki adet veri seti kullanılmıştır. Eğitim veri seti 5101, test veri seti ise 245 adet veriden oluşmaktadır. Modelin doğruluk oranının yüksek olması için hem eğitim veri sayısı yüksek tutulmuş, hem de veri setlerindeki beş kategorinin de içerdiği veri sayısı birbirine çok yakın oranlarda tutulmuştur. Veri setleri, Python kodu ile manuel olarak oluşturulmuştur ve belirli bir formata sahiptir. Bu format; kullanıcının yaşı, kilosu, cinsiyeti ve egzersiz geçmişi parametrelerini içermektedir. Sınıflandırma işleminde yaş parametresi baz alınmıştır ve belirli yaş aralıkları için etiketleme işlemi yapılmıştır. Veri setleri için yaş aralığı 20-40, kilo aralığı 50-100 arasında tutulmuştur. 20-25 yaş aralığı için “Seviye 1”, 25-30 yaş aralığı için “Seviye 2”, 30-35 yaş aralığı için “Seviye 3”, 35-40 yaş aralığı için “Seviye 4” ve 40-45 yaş aralığı için “Seviye 5” şeklinde etiketleme işlemi yapılmıştır. Her yaş aralığı için 50-100 kilo arasındaki değerler hem erkek hem de kadın için alınmış, her cinsiyet için “egzersiz geçmişi var” ve “egzersiz geçmişi yok” şeklinde ayrımlar da yapılmış ve buna göre veriler oluşturulmuştur. Veri seti CSV formatındadır.

```python
while True:
  try:
    #Yaş aralığı 20-40, minimum:20 ve maksimum:40
    var1 = int(input('Yaş (20-40 arasında): '))
    if var1 < 20 or var1 > 40:
      raise ValueError('Yaş 20 ile 40 arasında olmalıdır!')

    #Kilo aralığı 50-100, minimum:50 ve maksimum:100
    var2 = int(input('Kilo (50-100 arasında): '))
    if var2 < 50 or var2 > 100:
      raise ValueError('Kilo 50 ile 100 arasında olmalıdır!')

    #Cinsiyet değer seçim aralığı 1-0, kadın:1 ve erkek:0
    while True:
      var3 = input('Cinsiyet (Kadın:1, Erkek:0): ')
      if var3 in ['1', '0']:
        var3 = int(var3)
        break
      else:
        print('Hatalı değer girdiniz. Lütfen 1 (Kadın) veya 0 (Erkek) giriniz: ')

    #Egzersiz geçmişi değer seçim aralığı 1-0, egzersiz geçmişi var:1 ve egzersiz geçmişi yok:0
    while True:
      var4 = input('Egzersiz Geçmişi (Var:1, Yok:0): ')
      if var4 in ['1', '0']:
        var4 = int(var4)
        break
      else:
        print('Hatalı değer girdiniz. Lütfen 1 (var) veya 0 (yok) giriniz.')

  except ValueError as e:
    print(e)
    print('Lütfen tekrar deneyin.')
  else:
    print('Girilen Değerler:')
    print('Yaş:', var1)
    print('Kilo:', var2)
    print('Cinsiyet:', 'Kadın' if var3 == 1 else 'Erkek')
    print('Egzersiz Geçmişi:', 'Var' if var4 == 1 else 'Yok')
    break
```

Şekil 3.1.1: Kullanıcıdan Parametre Alma İşlemi

Yukarıdaki kod bloğunda görüldüğü üzere veri setini oluşturmak için yaş, kilo, cinsiyet ve egzersiz geçmişi değerleri konsol üzerinden alınarak CSV dosyasına yazdırılmak üzere parametre aralıkları belirlenmiştir. Yaş değerleri 20-40, kilo değerleri 50-100 aralığında girilecek; cinsiyet ve egzersiz geçmişi değerleri ise 1 ve 0 ile ifade edilerek konsolda değer girilecektir. Aksi bir değer girildiğinde hata verilecek ve tekrardan değer girişi yapılması istenecektir.

```python
generalArr = []

#Numaralandırıcı tanımlayalım ve ilk değerini atayalım.
count = 0

for var1 in range(40, 45):
    for var2 in range(50, 101):
        for var3 in [0, 1]:
            for var4 in [0, 1]:
                if var3 == 1 and var4 == 1:
                    total = str(var1) + ' yasinda ' + str(var2) + ' kilo ' + ' erkek ' + ' egzersiz gecmisi yok '
                    obj = {"text": total, "label": 4}
                    generalArr.append(obj)
                    count += 1

#Pandas kütüphanesini import edelim.
import pandas as pd

#DataFrame'i oluşturalım.
df = pd.DataFrame(generalArr)

#Numaralandırıcıyı kaldıralım.
df.index = range(len(df))

#Değerleri CSV dosyasına yazalım.
df.to_csv('Training.csv', index=False)
```

Şekil 3.1.2: Veri Setini Bir Diziye Aktarma ve CSV Dosyasına Çevirme

Girilen parametre değerleri oluşturulan bir diziye aktarılmıştır. For döngüsündeki işlem sayısını takip edebilmek adına bir count değişkeni oluşturulmuş ve ilk değeri 0 atanmıştır. For döngüsünün "var1" değişkenine yaş aralığı, "var2" değişkenine kilo aralığı, "var3" değişkenine cinsiyet ve "var4" değişkenine ise egzersiz geçmişi değerleri tanımlanmıştır. "var1" değişkenine sırasıyla 20-25, 25-30, 30-35, 35-40, 40-45 değerleri girilerek beşer beşer arttırılmış yaş aralıkları girilmiştir. Her yaş aralığı için 50-100 kilo aralığı, kadın/erkek cinsiyetleri ve egzersiz geçmişi var/egzersiz geçmişi yok parametre değerleri alınmıştır. Bu for döngüsü ile birlikte her yaş aralığı için her cinsiyete ait egzersiz geçmişi var ve egzersiz geçmişi yok durumlarına göre veri oluşturulmuş ve etiketlenmiştir. For döngüsünün en iç katmanında bulunan if döngüsündeki "obj" nesnesi, bir text ve bir label olarak metin dizesi oluşturur. Buradan sonraki kod satırında bu "obj" nesnesi yukarıda oluşturulan "generalArr" listesine eklenmiştir. Ardından "count" değişkenini sürekli birer birer arttırarak kaç örnek oluşturulması gerektiği belirlenmiştir. Bir sonraki adımda Pandas kütüphanesi import edilerek "generalArr" listesi ile bir Pandas DataFrame'i oluşturulur. Veri setinde bulunan tüm örnek veriler, bu DataFrame'in içerisindedir. Dataframe'i sıfırdan başlatarak ardışık tam sayılar olarak numaralandırdıktan sonra bu değerleri bir CSV dosyasına yazar. "index" değerini "False" olarak tanımlaması, DataFrame indexlerinin CSV dosyasına yazılmasını engeller.

Bu "createTrain.py" dosyası ile hem eğitim veri seti, hem de test veri seti için uygun şekilde veri oluşturma işlemi tamamlanmış olur. Dosyalar CSV formatına çevrilerek eğitim için hazır hale getirilmiştir.

3.2 Text-Classification İşlemi

Bu projede, NLP (Natural Language Processing/Doğal Dil İşleme) üzerinden bir metin sınıflandırma işlemi gerçekleştirilmiştir. Bunun için SVM (Support Vector Machines) denilen destek vektör makineleri üzerinden bir model oluşturulmuş, veri setleri bu modele göre eğitilmiştir.

```
#Veri setini yükleyip analiz edebilmek için Pandas kütüphanesini import edelim.
import pandas as pd

#Kelimelerimizi sayısal matrislere dönüştüren Scikit-learn'e ait CountVectorizer ekleyelim.
from sklearn.feature_extraction.text import CountVectorizer
```

Şekil 3.2.1: Uygun Kütüphanelerin Import Edilme Aşaması

Kod bloğunda görüldüğü üzere eklenen ilk kütüphane Pandas'tır. Pandas, veri analizi ve veri manipülasyonu için kullanılan bir kütüphanedir. Veri kaynaklarından veri okuma ve veri yazma (Excel dosyaları, veritabanları, CSV dosyaları) işlemlerini kolaylıkla entegre edebilmesi açısından tercih edilir. Projemizde kullandığımız veri setleri CSV formatında olduğu için öncelikle bu veri setlerini yüklemek ve analiz edebilmek amacıyla Pandas kütüphanesi import edilmiştir.

Kod bloğunun ikinci kısmında Scikit-learn kütüphanesi ve bu kütüphaneye ait CountVectorizer sınıfı eklenmiştir. Scikit-learn kütüphanesi, makine öğreniminde temel olarak tercih edilen bir kütüphanedir. Bu kütüphaneye ait olan "feature_extraction.text" modülü, metin verilerini sayısal verilere dönüştürmek için "CountVectorizer" sınıfını kullanır. Metin verilerinin sayısal verilere dönüştürülmesinin nedeni makine öğrenimi modellerinin ve algoritmalarının çoğunlukla sayısal veriler ile çalışmasından kaynaklıdır. Aynı şekilde metinlerden anlamlı özellikler çıkartmak için de sayısal verilere dönüştürme işlemi işe yarar durumdadır.

```
#Text-classification işlemi için SVC kullanalım.
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

#Eğitim veri setini yükleyelim.
train_data = pd.read_csv("Training.csv")
X_train = train_data["text"].values
y_train = train_data["label"].values

#Test veri setini yükleyelim.
test_data = pd.read_csv("Validate.csv")
X_test = test_data["text"].values
y_test = test_data["label"].values
```

Şekil 3.2.2: Modelin Eklenmesi ve Veri Setlerinin Yüklenmesi

Yukarıdaki kod bloğunu ele aldığımızda text-classification işlemi için SVC modeli kullanıldığı görülmektedir. SVC modeli, destek vektör makinelerinde kullanılan bir sınıflandırma modelidir. Scikit-learn kütüphanesinin SVM modülünden SVC sınıfı import edilerek sınıflandırma işlemi için kullanılacaktır. Aynı kütüphane üzerinden import edilmiş olan "accuracy_score", bize modelin çalıştığı zamandaki doğruluk oranını verecektir. Doğruluk oranının yüzdesi ne kadar fazlaysa, model o kadar doğru çalışmış olur. Çoklu etiket sınıflandırmalarında kullanılan bu işlev, alt küme doğruluğunu hesaplayarak modelin doğruluk oranının yüzdesini verir.

Bir sonraki blokta eğitim veri setinin nasıl yükleneceği gösterilmiştir. "Training.csv" dosyasında bulunan eğitim verileri, Pandas kütüphanesi aracılığıyla okunarak "train_data" adlı DataFrame'e yüklenir ve değişkene atama işlemi gerçekleştirilir. Bu değişkene atanan CSV dosyasındaki verilerin "text" sütunları seçilir ve bu sütuna ait değerler "x_train" değişkenine atanır. Modeli eğitecek veriler bu değişkende tutulmuş olur. Aynı şekilde eğitim veri setindeki CSV dosyasında bulunan bir diğer sütun olan "label" kısmına ait olan veriler ise "y_train" değişkenine atılarak burada tutulur. Label sütunundaki veriler, metin verilerinin hangi kategoriye ait olduğunu belirten verilerdir. Bu etiketler "y_train" değişkeninde tutulmuştur.

Veriler CSV dosyasından okunduktan ve text ile label sütunlarına ait veriler değişkenlere atandıktan sonra, sıra test verilerini yüklemeye gelir. "Validate.csv" dosyasında tutulan 245 adet test verisi, tıpkı eğitim verilerinde olduğu gibi text ve label sütunlarına ait verilerin seçilme ve değişkenlere atanma işlemlerine tabi tutulur. Bu dosyada bulunan text verileri, "x_test" adlı değişkene atılır ve modelin öğreneceği test verileri burada saklanır. Aynı şekilde test verilerine ait "label" sütununda bulunan etiket değerleri ise "y_test" değişkenine atanarak burada tutulmuştur.

```
#CountVectorizer ile metinleri sayısal özelliklere dönüştürelim.
vectorizer = CountVectorizer()
X_train_vectorized = vectorizer.fit_transform(X_train)
X_test_vectorized = vectorizer.transform(X_test)

#SVM modeli oluşturalım.
model = SVC(kernel='linear')

#Sayısal veri ve eğitim verilerini modelin öğrenmesini sağlayalım.
model.fit(X_train_vectorized, y_train)
```

Şekil 3.2.3: Metinlerin Sayısal Dönüşümü ve Model Eğitimi

CountVectorizer, Scikit-learn kütüphanesine ait bir sınıftır ve metin verilerinin sayısal verilere dönüştürülmesinde rol oynar. CountVectorizer sınıfından "vectorizer" adlı bir nesne türetilmiştir. Bu türetilen nesne, metin verilerini sayısal verilere dönüştürmek üzere bir vektörleştirme işlemi gerçekleştirir. Eğitim veri kümesine ait veriler, bu sınıftan türetilen modelin "fit_transform" yöntemiyle alınıp analiz edildikten sonra kelime dağarcıkları oluşturularak bu kelime dağarcıklarına göre vektörleştirilir. "fit()" yöntemi, vektörleştiriciyi eğitim verilerine uyarlamıştır. Sözcük haznesi yani kelime dağarcıkları bu yöntem sayesinde oluşturulur. "transform()" yöntemi ise eğitim verilerini vektörleştirici vasıtasıyla dönüştürür. Her belgeyi sayısal bir özellik vektörüne dönüştürür ve her özellik sözlükteki bir kelimenin sayısını temsil eder. Eğitim verilerinin dili bu şekilde öğrenilmiştir ve "x_train_vectorized" adlı nesnede saklanmıştır. Aynı işlem test verileri üzerinde de gerçekleştirilir. Test verileri alınıp analiz edilerek sözcük dağarcıkları oluşturulur ve oluşturulan bu sözcük dağarcıklarına göre test verileri vektörleştirilir. Yani bu sefer de test verilerine ait bir öğrenme işlemi gerçekleştirilmiş olunur ve "x_test_vectorized" adlı nesnede tutulur.

SVC (Support Vector Classification/Destek Vektör Sınıflandırması) algoritması ile bir sınıflandırma modeli oluşturulur. Oluşturulan bu model, eğitim verileri ve etiketleri kullanarak eğitilir. Burada bulunan "kernel", algoritmada kullanılan çekirdek türünü belirtir. Bir çekirdek türü belirtilmezse "rbf" kullanılır. Burada kullanılan çekirdek türü "linear" olarak belirtilmiştir. Bu durumda, doğrusal bir destek vektör makinesi kullanılmış olur. Verilerin doğrusal olarak ayrılabildiği veya çok büyük miktarda veri içeren veri setlerinde tercih edilen bir yöntemdir.

Bir sonraki satırda bulunan koda baktığımızda oluşturulmuş olan SVC modeli, eğitim veri seti ve bu eğitim veri setine ait olan etiket verilerinin tutulduğu değişken ile eğitilir. Verilerin özellikleri ve bu özelliklere ait hangi etiketin hangi veri grubunda bulunulduğu öğrenilmeye başlanır ve buna göre bir karar sınırı belirlenir. Hangi veri grubuna ne tür veri girişinin olduğunu ve bundan sonra gelebilecek olan verilerde nasıl bir

kategorizasyon yapması gerektiğini öğrenmeye başlar. Veriler arasındaki örüntü öğrenilir.

```
#Test verisi ile tahminler yapalım.
y_pred = model.predict(X_test_vectorized)

#Doğruluk oranını hesaplayalım.
accuracy = accuracy_score(y_test, y_pred)
print("Model Doğruluk Oran:", accuracy)
```

Şekil 3.2.4: Modelde Tahmin Yapma ve Doğruluk Oranı Hesaplama

Test verilerine ait öğrenme işleminden sonra bu verilerin tutulduğu "x_test_vectorized" nesnesi, eğitilmiş model kullanılarak her bir veri için örnek tahminlerde bulunur. Burada öğrendiği yeni örneklere ait sınıf tahmini yapar ve bu tahmin sonuçlarını "y_pred" değişkeninde saklar.

Modelin doğruluk oranını hesaplamak amacıyla "accuracy_score" fonksiyonu kullanılır. Bir önceki kod bloğunda yapılmış olan tahminlerin doğruluk oranının tutulduğu "y_pred" ve daha öncesinde test verilerine ait etiketlerin tutulduğu "y_test" değişkenleri bu fonksiyonun içine atılır ve modelin öğrenme doğruluğu hesaplanır. Bu iki değişkene ait verileri kendi içerisinde karıştırarak bir doğruluk oranı elde eder ve bunu "accuracy" değişkenine atar. Bu oran 0 ile 1 arasında bir değerdir. Oran ne kadar yüksek olursa model verileri o kadar iyi öğrenmiş demektir. Ardından bu doğruluk oranı konsol üzerinde ekrana yazdırılır ve eğer oran düşükse birtakım iyileştirme işlemleri yapılır.

```
#Sınıflandırma işlemi için fonksiyon oluşturalım.
def classify_exercise():
    # Kullanıcıdan veri girişi alalım.
    age = int(input("Yaş girin: "))
    weight = float(input("Kilonuzu girin (kg): "))
    gender = input("Cinsiyetinizi girin (Erkek/Kadın): ").lower()
    exercise_history = input("Egzersiz geçmişiniz var mı? (Evet/Hayır): ").lower()
    heartDisease = input("Herhangi bir kalp rahatsızlığınız var mı? (Evet/Hayır):").lower
    respiratoryDisease = input("Herhangi bir solunum yolu rahatsızlığınız var mı? (Evet/Hayır):").lower

    # Girilen verileri yazdıralım.
    input_text = f"Yaş: {age}, Kilo: {weight}, Cinsiyet: {gender}, Egzersiz Geçmişi: {exercise_history}, Kalp Rahatsızlığı: {heartDisease}, Sol

    # Kullanıcı girdisini transform yöntemiyle modelin anlayabileceği sayısal formata dönüştürelim.
    input_vectorized = vectorizer.transform([input_text])

    # Modelin anlayabileceği formatı sınıflandıralım.
    prediction = model.predict(input_vectorized)

    # Sonucu ekrana yazdıralım.
    print("\n--- Tahmin Sonucu ---")
    print("Tahmin Edilen Kategori:", prediction[0])

# Sınıflandırma işlemini başlatalım.
classify_exercise()
```

Şekil 3.2.5: Kullanıcıdan Veri Girişi Alma ve Sınıflandırma İşlemi

Yukarıdaki kod bloğunda sınıflandırma işlemi için bir fonksiyon oluşturulduğu görülmektedir. Bu fonksiyonda, ilk olarak kullanıcıdan alınacak parametreler tanımlanmıştır Yaş, kilo, cinsiyet, egzersiz geçmişi, geçmiş kalp rahatsızlığı ve geçmiş solunum yolu rahatsızlığı kullanıcıdan istenen parametre girdileridir. Kullanıcıdan

toplanan bu veriler tek bir satırda yazdırılmak üzere “input_text” değişkeninde toplanır. Tek satırda toplanan kullanıcı verileri, CountVectorizer yardımıyla bir vektörleştirme işlemine tabi tutulur ve metin verileri sayısal verilere dönüştürülür. Dönüştrülen bu veriler “input_vectorized” değişkeninde saklanır. Bu değişkende saklanan dönüştürülmüş veriler, model kullanılarak tahmin işleminden geçer ve uygun kategoriye atanır. Tahmin edilen kategori “prediction” değişkeninde saklanır. Ardından bu kategori ekrana yazdırılır. Fonksiyonun işlemleri bittikten sonra bu fonksiyon son kod satırında çağırılır.

3.3 Web Arayüzü Kullanabilmek İçin Flask

Flask, web uygulamaları oluşturmak amacı ile kullanılan bir frameworktür. Hızlı çalışması, esnek bir yapı içermesi, zaman ve maliyet tasarrufu sağlaması gibi nedenlerden ötürü tercih edilmiştir. Bu projede, model konsol üzerinde test edildikten ve doğrulandıktan sonra sonra bir web arayüzü tasarımı yapılmış ve Flask ile bağlantı sağlanmıştır.

```
#Python tabanlı web uygulamalar oluşturmak için Flask'ı ekleyelim.
from flask import Flask, render_template, request


#Veri setini yükleyip analiz edebilmek için Pandas kütüphanesini import edelim.
import pandas as pd

#Kelimelerimizi sayısal matrislere dönüştüren Scikit-learn'e ait CountVectorizer ekleyelim.
from sklearn.feature_extraction.text import CountVectorizer


#Text-classification işlemi için SVC kullanalım.
from sklearn.svm import SVC
```

Şekil 3.3.1: Uygun Kütüphanelerin Import Edilmesi İşlemi

Yukarıdaki kod bloğunu incelediğimizde uygun kütüphanelerin “app.py” dosyasına eklendiğini görürüz. Flask import edildikten sonra eklenen “render_template” fonksiyonu, eklenecek olan HTML dosyalarını sunmak için kullanılan bir fonksiyondur. “request” vasıtasıyla da HTTP isteklerinin işlenmesine olanak tanınır.

Metin sınıflandırma için kullandığımız ve modeli eğittimiz ana dosyamız olan “textClassification.py” dosyasında eklediğimiz Pandas kütüphanesi, burada tekrardan kullanılmalıdır. Flask’ı çalıştırabilmek ve verileri analiz edip manipülasyon işlemini gerçekleştirebilmek için Pandas kütüphanesi burada da import edilmiştir. Scikit-learn kütüphanesinden CountVectorizer import edilerek metin verilerinin sayısal verilere dönüştürme işlemi için ilgili fonksiyon tanımlanmıştır. Aynı şekilde kullanılacak destek vektör makinesi sınıflandırması için gereken SVC sınıfı da import edilmiştir.

```
##Eğitim veri setini yükleyelim.
train_data = pd.read_csv("Training.csv")
X_train = train_data["text"].values
y_train = train_data["label"].values

#CountVectorizer ile metinleri sayısal özelliklere dönüştürelim.
vectorizer = CountVectorizer()
X_train_vectorized = vectorizer.fit_transform(X_train)
```

Şekil 3.3.2: Veri Setinin Yüklenmesi ve Metinlerin Sayısal Özelliklere Dönüştürülmesi

"Training.csv" dosyasında bulunan eğitim veri setleri, model eğitirken kullanıldığı gibi arayüzün çalıştırılması için gereken Flask dosyasında da tekrardan yüklenmelidir.

Bu dosya yüklenip Pandas kütüphanesi ile dosyanın okunması sağlanarak "train_data" adlı bir Pandas DataFrame'ine depolanmıştır. Veri setimiz text ve label olmak üzere sırasıyla metin içeriği ve etiket sütunlarından oluştuğu için, "train_data" DataFrame'ine ait text sütunlarının metin verisi ayıklanarak "x_train" değişkenine; aynı DataFrame'in label sütunlarının metin verisi ayıklanarak "y_train" değişkenine atanmıştır. CountVectorizer sınıfı yardımıyla "vectorizer" adlı bir nesne oluşturularak bu nesneyi ayıklanan metin verilerinin tutulduğu "x_train" değişkeni üzerinde eğitir. CountVectorizer tüm eğitim belgelerindeki benzersiz kelimeleri (kelime dağarcığı) tanır. Ayrıca her kelimenin her belgede kaç kez geçtiğini de sayar. Bu yöntem, metinsel materyalinin dilini öğrenmeye benzer.

```
model = SVC(kernel='linear')

#Saysal veri ve eğitim verilerini modelin öğrenmesini sağlayalım.
model.fit(X_train_vectorized, y_train)

#Flask uygulamasını başlatarak ana nesnesini oluşturalım.
app = Flask(__name__)

#/ rotası için yönlendirme yapalım.
@app.route('/')
def index():
    return render_template('index.html')

#Classify rotası için POST isteği yapalım.
@app.route('/classify', methods=['POST'])
```

Şekil 3.3.3: Model Eğitimi ve Yönlendirme

Yukarıdaki kod bloğunda ilk olarak Scikit-learn kütüphanesindeki SVC sınıfına ait lineer bir model oluşturulmuştur. Lineer model, verileri lineer bir sıralayıcı düzlemde yani hyperplane kullanarak sınıflandırır. Hyperplane kullanarak sınıflandırma yapmak; basitlik, kullanım kolaylığı, doğrusal ayrılabilirlik, overfitting (aşırı öğrenme) durumunu azaltma gibi nedenlerden dolayı tercih edilen bir yöntemdir. Ardından sayısal özelliklere

dönüştürülmüş “x_train_vectorized” ile etiketlerin yer aldığı “y_train” bu modelin içerisine parametre şeklinde verilerek model eğitimi başlatılmış olur. Model, kendisine parametre olarak verilmiş bu verileri öğrenerek gelecekteki tahminler için kullanılabilecek bir sınıflandırma elde etmiş olur.

Modele dair eğitim kısmı tamamlandıktan sonra Flask uygulamasını başlatmak için web çerçevesine ait bir örnek oluşturulur. Örnek oluşturulduktan sonra belirtilen rotaya yönlendirme yapabilmek için bir dekoratör kullanılır. Örneğin kullanıcı web sayfasına erişmek istediğinde “http://localhost:5000/”(genellikle) şeklinde bir URL girdiğinde index adlı bu fonksiyon çağırılır ve işlev görür.

Buradan sonra yazılmış olan index isimli fonksiyon, erişilecek sayfa için yönlendirme yapar. Kod bloğunda anlaşıldığı üzere yönlendirme “index.html” dosyasına yapılacaktır. Bu dosya index fonksiyonu içinde render edilerek bu şablon içeriğini döndürür ve kullanıcı bu sayfayla erişimi sağlar. Ardından “/classify” URL yoluna ait POST isteklerinin işlenebilmesi için bir dekoratör daha kullanılır. Bu URL yoluna bir istek gönderildiğinde tanımlı fonksiyonun çalışacağı anlamına gelir.

```python
def classify():
    age = int(request.form['age'])
    weight = float(request.form['weight'])
    gender = request.form['gender'].lower()
    exercise_history = request.form['exercise_history'].lower()
    heartDisease = request.form['heartDisease'].lower()
    respiratoryDisease = request.form['respiratoryDisease'].lower()


    #Girilen verileri yazdıralım.
    input_text = f"Yaş: {age}, Kilo: {weight}, Cinsiyet: {gender}, Egzersiz Geçmişi: {exercise_history}, Kalp Rahatsızlığı: {heartDisease}, Solu

    #Kullanıcı girdisini transform yöntemiyle modelin anlayabileceği sayısal formata dönüştürelim.
    input_vectorized = vectorizer.transform([input_text])

    #Modelin anlayabileceği formatı sınıflandıralım.
    prediction = model.predict(input_vectorized)

    #Flask'te HTML isteğini işleyelim.
    return render_template('index.html', prediction=prediction[0])

#Flask'in doğru çalışıp çalışmadığını kontrol edelim.
if __name__ == '__main__':
    app.run(debug=True)
```

Şekil 3.3.4: Yönlendirme Yapılan Fonksiyon ve İşlevleri

URL yoluna istek gönderildiğinde çalışacak fonksiyona ait ilgili parametreler tanımlanır. Bu parametreler; kullanıcıdan alınan yaş, kilo, cinsiyet ve egzersiz geçmişi bilgileridir. Kullanıcıdan alınan bu parametre girdileri tek bir satırda toplanarak “input_text” değişkenine atanır. Verileri tek satırda toplayan bu değişken, modelin tahmin yaparken kullanacağı veriyi temsil eder. Bir sonraki adımda bu değişken vektörize edilmek üzere kullanılır. CountVectorizer sınıfına ait oluşturulan “vectorizer” nesnesi ile “input_text” metnine ait veriler sayısal bir formata dönüştürülür ve “input_vectorized” değişkenine atanır. Bu şekilde yapılan metni sayısal formata dönüştürme işlemi, eldeki verinin model

tarafından anlaşılabilmesi ve işlenebilmesine olanak sağlar. Dönüştürülen bu metin, daha önceden eğitilmiş modele göre bir sınıflandırma işlemi yapar. Yaptığı tahmin işlemini "prediction" adlı nesnede saklar. Flask'a gönderilern HTML isteği işlenerek render edilir ve "index.html" şablonunun işlenmesi tetiklenmiş olur. Tahmin edilen kategorizasyon sonucu bu sayfada ilgili yerde gösterilir.

Flask'ın doğru çalışıp çalışmadığını kontrol etmek amacıyla yazılmış olan if bloğunda bir debug işlemi gerçekleştirilir. Hataların daha kolay yakalanabilmesi açısından "debug=True" şeklinde tetiklenir.

3.4 Web Arayüzü Tasarımı

Metin sınıflandırma işlemini gerçekleştiren modelin eğitimi tamamlandıktan ve uygun web arayüzüne bağlayacak framework yazıldıktan sonra web arayüzü için basit bir tasarım yapılmıştır.

Şekil 3.4.1: Web Arayüzü

Yaşa göre egzersiz tahminini gerçekleştirecek ve sınıflandırma yapacak bu model, konsol üzerinde başarılı bir şekilde çalışmaktadır. Flask kullanarak web arayüzüne yapılacak bağlantının kodları yazıldıktan sonra "index.html" sayfası için basit bir form ögesi oluşturularak kullanıcıdan yaş, boy, kilo, cinsiyet, egzersiz geçmişi, kalp rahatsızlığı geçmişi ve solunum yolu rahatsızlığı geçmişi bilgileri alınmaktadır. Kullanıcıdan alınan bu bilgiler doğrultusunda belirlenen beş kategoriden uygun olana sınıflandırma işlemi gerçekleştikten sonra mevcut kategoriye ait bilgi verilmiştir.

```
<body><br><br>
    <h1>Yaşa Göre Egzersiz Tahmini</h1>
    <form action="/classify" method="post" style="text-align: center;">
        <label for="age">Yaş (20-40):</label><br>
        <input type="text" id="age" name="age" min="20" max="40" required><br>
        <label for="weight">Kilo (kg):</label><br>
        <input type="text" id="weight" name="weight" min="50" max="100" required><br>
        <label for="size">Boy (cm):</label><br>
        <input type="text" id="size" name="size" required><br>
        <label for="gender">Cinsiyet (Kadın/Erkek):</label><br>
        <input type="text" id="gender" name="gender"><br>
        <label for="exercise_history">Egzersiz Geçmişiniz Var Mı? (Evet/Hayır):</label><br>
        <input type="text" id="exercise_history" name="exercise_history"><br>
        <label for="heartDisease">Herhangi bir kalp rahatsızlığınız var mı? (Evet/Hayır):</label><br>
        <input type="text" id="heartDisease" name="heartDisease"><br>
        <label for="respiratoryDisease">Herhangi bir solunum yolu rahatsızlığınız var mı? (Evet/Hayır):</label><br>
        <input type="text" id="respiratoryDisease" name="respiratoryDisease"><br><br>
        <input type="submit" value="Sınıflandır">
    </form><br><br>
    <h2>Yaşınıza Göre Uygun Egzersiz Biçimi:</h2>
    <p>Uygun Kategori: {{ prediction }}</p>
    {% if prediction == 'Seviye1' %}
    <p>20-25 yaş arasındaki yaş grubu daha enerjik ve dinamik sporları tercih eder. Koşu, yüzme, bisiklet, voleybol ve basketbol kardiyo ve daya
    {% elif prediction == 'Seviye2' %}
        <p>25-30 yaş arasındaki yaş grubu genellikle daha tecrübeli ve hedef odaklıdır. Yorucu kalp egzersizlerinin yanı sıra ağırlık antrenmanı
        </p>
    {% elif prediction == 'Seviye3' %}
        <p>30 -35 arasındaki yaş grubu; esneklik ve kuvvet antrenmanlarına ek olarak, yavaş tempolu egzersizler ve esneme rutinleri bu yaş grubu
        </p>
    {% elif prediction == 'Seviye4' %}
        <p>35-40 arasındaki yaş grubunda sakatlanma riski yaşla birlikte artar, bu nedenle düşük etkili aerobik egzersizler ve yavaş tempolu kuv
    {% elif prediction == 'Seviye5' %}
        <p>40-45 yaş aralığındaki grupta esneklik, denge ve kuvvet antrenmanı çok önemlidir. Yoga, pilates, tai chi, düşük etkili kardiyo rutinl
    {% else %}
        <p style="color: ■white;">Tanımlanamayan etiket.</p>
    {% endif %}

</body>
```

Şekil 3.4.2: Web Arayüz Kodları

Yukarıdaki görselde web arayüzüne ait HTML kodları verilmiştir. Bir form ögesi oluşturularak girilecek olan her parametre zorunlu kılınmış, buna yönelik olarak sınıflandırma işlemi gerçekleştirildikten sonra her kategoriye ait o kategoriyi açıklayan birkaç cümleden oluşan bilgi paragrafları eklenmiştir. İlgili araştırmalar sonucunda yaş aralığı arttıkça egzersiz aktivitesinin seviyesi düşürülmüş ve ileri yaşlara daha basit egzersizler verilmesi karar kılınmıştır. Dolayısıyla kategorilere ait bilgi paragrafları da aynı ölçüde belirlenmiştir.

3.5 Projenin Çalıştırılması

Proje hem web arayüzünde hem de konsol üzerinde çalıştırıldığında sorunsuz bir şekilde çalışmakta ve sınıflandırma işlemini yapmaktadır. Projeyi konsolda çalıştırmak için modelin eğitildiği "textClassification.py" dosyasına gidilerek bu dosya çalıştırılmalıdır. Modelin eğitim sonucundaki doğruluk oranı da burada görülecektir.

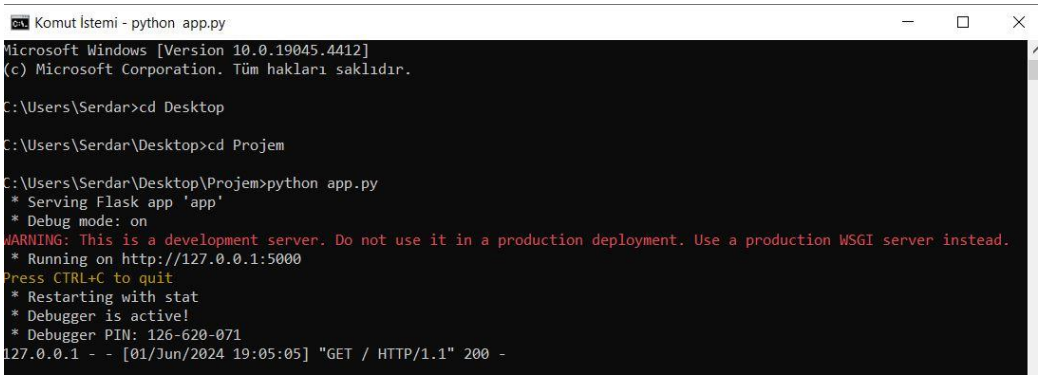
```
Model Doğruluk Oranı: 0.8073770491803278
Yaşınızı girin: 43
Kilonuzu girin (kg): 78
Cinsiyetinizi girin (Erkek/Kadın): kadın
Egzersiz geçmişiniz var mı? (Evet/Hayır): hayır
Herhangi bir kalp rahatsızlığınız var mı? (Evet/Hayır):evet
Herhangi bir solunum yolu rahatsızlığınız var mı? (Evet/Hayır):hayır

--- Tahmin Sonucu ---
Tahmin Edilen Kategori: Seviye5
```

Şekil 3.5.1: Modelin Konsolda Çalıştırılması

Yukarıdaki görselde görüldüğü üzere modelin doğruluk oranı 0.80 yani %80 seviyesindedir. 0-1 aralığındaki doğruluk oranı, 1'e ne kadar yakınsa modelin o kadar iyi çalıştığı ve doğru tahmin yaptığı anlamını taşır. %80'lik doğruluk oranıyla çalışan model, sorunsuz bir şekilde tahmin işlemini yapmakta ve ilgili kategoriye sınıflandırma işlemini gerçekleştirmektedir.

Projenin web arayüzüne erişim sağlayabilmek için öncelikle komut ekranında ilgili dosya dizinine giderek Flask kodlarının bulunduğu Python dosyasının çalıştırılması ve komut ekranında verilen URL yoluna göre web arayüzünün açılması gerekir.



Şekil 3.5.2: Komut Ekranında Yapılacak İşlemler

Komut ekranı açıldıktan sonra projenin ilgili dosyalarının bulunduğu dizine gitmek gerekir. Bu dizine eriştikten sonra Flask frameworkünün yazıldığı Python dosyasını bularak çalıştırmak gerekir. Projede kullanılan dosya "app.py" olduğu için "python app.py" şeklinde yazılarak servise erişilmeye çalışılır. Ardından aşağıda gösterilen "http://127.0.0.1:5000" linki ile web sayfasına erişim sağlanır.

Şekil 3.5.3: Web Arayüzü Genel Görünüm

Web arayüzünde basit ve sade bir tasarım tercih edilmiştir. Kullanıcı kendisine ait parametre değerlerini ilgili alanlara girmelidir. Her bir parametre değerinin girişi zorunludur ve herhangi biri atlanarak sınıflandırma işlemi yapılamaz.

Şekil 3.5.4: Sınıflandırma İşlemi

Kullanıcı bilgilerini girdikten ve butona bastıktan sonra yaş grubuna uygun egzersiz biçimi ve o gruba ait birkaç cümleden oluşan öneri paragraf metni yukarıdaki gibidir. Her yaş grubuna uygun kategori yukarıdaki gibi seviyesi belirtilerek ve ardından bir paragraflık özet şeklinde öneri metni verilerek sınıflandırma işlemi başarılı bir şekilde gerçekleştirilmiştir.

4. SONUÇ

Bu tez çalışmasında; kullanıcıların yaş ve diğer parametrelere bağlı olarak sınıflandırılması, yaş aralıkları değerlerinin baz alınarak en uygun egzersiz planının oluşturulması temel alınmıştır. Kullanıcıdan alınan yaş, kilo, cinsiyet, egzersiz geçmişi, geçmiş kalp rahatsızlığı ve geçmiş solunum yolu rahatsızlığı parametreleri alınarak yapay zeka teknolojileriyle kişiselleştirilmiş egzersiz planları oluşturulmuştur. Bu çalışmadaki temel amaç, bireylerin kendi yaş grupları çerçevesinde fiziksel kapasitelerine ve ihtiyaçlarına uygun olarak uygulaması gereken egzersiz planının oluşturulmasıdır.

Projede makine öğrenimi teknikleri kullanılarak bireylerin yaş gruplarına göre sınıflandırılması sağlanmıştır ve optimize edilmiştir. Elde edilen sonuçlara ve modelin doğruluk oranına bakıldığında tahminin çok yüksek ihtimalle doğru yapıldığı görülmektedir. Yaş grup dağılımı ve fiziksel kapasiteye göre egzersiz planı sunan bu makine öğrenimi yaklaşımı, bireylerin sağlık ve spor konusundaki hedeflerine ulaşmalarında yardımcı bir rol üstlenmektedir.

Çalışma bünyesinde elde edilen temel bulgulara bakacak olursak veri analizi ve model performansı açısından kullanıcıdan elde edilen parametreler başarılı bir şekilde analiz edilmiştir. Performansları karşılaştırılan farklı yapay zeka algoritmaları arasından modelin doğruluk oranını en yüksek seviyede veren destek vektör makineleri kullanılmıştır. Yaş bazlı sınıflandırma işlemi yapılarak bireylerin fiziksel aktiviteleri göz önünde bulundurulmuş ve uygun sınıfta yer alması sağlanmıştır. Bu çalışma, yapay zeka teknolojisinin sağlık ve spor alanlarında nasıl kullanılabileceğine ışık tutmakta ve gelecekteki araştırmalar için sağlam bir temel oluşturma niteliği barındırmaktadır.

Sonuç olarak bu tez çalışması kapsamında yapay zeka tabanlı bireyselleştirilmiş egzersiz rejimlerinin geliştirilmesi ve uygulanmasının, bireylerin daha sağlıklı yaşamalarına yardımcı olma potansiyeline sahip olduğunu göstermektedir. Bu stratejinin daha kapsamlı ve uzun vadeli çalışmalarla desteklenmesi, sağlık ve spor alanlarında daha yaratıcı ve başarılı çözümler üretilmesine yardımcı olacaktır.

KAYNAKÇA

[1] Srinath, K. R. (2017). Python–the fastest growing programming language. *International Research Journal of Engineering and Technology*, *4*(12), 354-357.

[2] Flask: Building Python Web Services. [online] O'Reilly | Safari. Available at: https://learning.oreilly.com/library/view/flask-building-python/9781787288225/ch01.html [Accessed 26 Nov. 2019].

[3] www.oreilly.com. (n.d.). 1. Installation - Flask Web Development, 2nd Edition [Book]. [online] Available at: https://learning.oreilly.com/library/view/flaskweb-development/9781491991725/ch01.html#ch_install [Accessed 2 Dec. 2019a].

[4] flask.palletsprojects.com. (n.d.). Extensions — Flask Documentation (1.1.x). [online] Available at: https://flask.palletsprojects.com/en/1.1.x/extensions/ [Accessed 5 Dec. 2019]

[5] A. Yılmaz, "Yapay Zekâ,". Kodlab Yayınevi, İstanbul, Türkiye, 2020.

[6] Alpaydin, E., Introduction to machine learning. 2004: The MIT Press.

[7] Tantuğ, A. C. (2016). Metin sınıflandırma. *Türkiye Bilişim Vakfı Bilgisayar Bilimleri ve Mühendisliği Dergisi*, *5*(2).

[8]. E. Brill, and R. J. Mooney, "An Overview of Empirical Natural Language Processing," AI Magazine, vol. 18, no. 4, 1997.

[9] SEKER, S. E. (2015). Doğal Dil İşleme (Natural Language Processing). *YBS Ansiklopedi*, *2*(4), 14-31.

[10] Taşkıran, S. F. (2021). *Doğal dil işleme ile akademik metinlerin kümelenmesi* (Master's thesis, Konya Teknik Üniversitesi).

[11] Tantuğ, A. C. (2016). Metin sınıflandırma. *Türkiye Bilişim Vakfı Bilgisayar Bilimleri ve Mühendisliği Dergisi*, *5*(2).

[12] HAYKIN S. (1999). Neural Networks: A Comprehensive Foundation [Elektronik Sürüm], Prentice Hall Inc, New Jersey.

[13] SONG, O., HU, W. ve XIE, W. (2002). "Robust Support Vector Machine with Bullet Hole Image Classification" [KurĢun Deliklerini Destek Vektör Makineleri ile Sınıflandırma], IEEE Transactions on Systems, Man and Cybernetics – Part C: Applications and Rewiews, 32(4): 440

[14] ÇOMAK, E. (2008). Destek Vektör Makinelerinin Etkin Eğitimi İçin Yeni Yaklaşımlar, Doktora Tezi, Selçuk Üniversitesi Fen Bilimleri Enstitüsü, Konya.

[15] Vapnik, V.N., 2000, The Nature of Statistical Learning Theory, 2. Baskı, Springer-Verlag, New York.

[16] Foody, G.M., Mathur, A., 2004, A Relative Evaluation of Multiclass Image Classification by Support Vector Machines, IEEE Transactions on Geoscience and Remote Sensing, 42(6): 1335–1343.

[17] Osuna, E.E., Freund, R., Girosi, F., 1997, Support Vector Machines: Training and Applications, A.I. Memo No. 1602, C.B.C.L. Paper No. 144, Massachusetts Institute of Technology and Artificial Intelligence Laboratory, Massachusetts.

[18] Vapnik, V.N., 1995, The Nature of Statistical Learning Theory, Springer-Verlag, New York.

[19] Cortes, C., Vapnik, V., 1995, Support-Vector Network, Machine Learning, 20(3): 273–297.

[20] Hsu, C.W., Chang, C.C., Lin, C.J., 2010, A Practical Guide to Support Vector Classification,http://www.csie.ntu.edu.tw/~cjli n/papers/guide/guide.pdf.

[21] Arnulf B. A. Graf, S. Borer, 2001, Normalization in Support Vector Machines, Lecture Notes in Computer Science, 2191: 277-282.

ÖZGEÇMİŞ

CONTACT

05392174859

dorhan23@gmail.com

https://www.linkedin.com/in/di%C4%9Fdem-orhan-935733275/

https://github.com/digdemorhan

EDUCATION

2020-2024

FIRAT ÜNİVERSİTESİ

- Lisans, Bilgisayar Mühendisliği

BECERİLER

- Java
- Python
- HTML
- CSS
- JavaScript
- SpringBoot
- MYSQL
- Postman API
- Dijital Pazarlama
- Girişimcilik
- E-ticaret

LANGUAGES

- English (intermediate)

DİĞDEM ORHAN

JUNIOR BACKEND DEVELOPER

HAKKIMDA

Fırat Üniversitesi Bilgisayar Mühendisliği Bölümünde son sınıf öğrencisi olan hırslı ve tutkulu bir yazılımcıyım. Temmuz 2024'te mezun olacağım. Backend developer olarak kariyerime başlamak için heyecan duyuyorum. Hedefim, yazılım geliştirme becerilerimi gerçek hayat problemlerine entegre ederek bu problemlere inovatif ve yaratıcı çözümler sağlamaktır. Hızlı öğrenen, uyumlu ve çalışkan biri olarak yeni bir ekibe kolayca adapte olabileceğime inanıyorum.

İŞ DENEYİMİM

Narveri Eğitim Bilişim — ARA 2022 - HAZ 2023
Gönüllü Stajyer

- Burada 7 aylık deneyimim boyunca Java üzerinde kendimi geliştirdim, veritabanı kullanmayı öğrendim, çeşitli küçük uygulamalar yaptım ve ofis ortamını tecrübe ettim.

Narveri Eğitim Bilişim — TEM 2023 - TEM 2023
Stajyer

- Basit bir arayüze sahip ToDoList web uygulaması geliştirdim.
- Kullanılan teknolojiler: Java, SpringBoot, HTML, CSS, JS, NetBeans IDE, Intellij IDEA, MYSQL, Postman API.

Kriptarium Ar-Ge Yazılım Danışmanlık Savunma Sanayi ve Ticaret Ltd. Şti. — EYL 2023 - EYL 2023
Stajyer

- İçerisinde çeşitli alanlardan kursların bulunduğu bir web uygulaması geliştirdim.
- Kullanılan Teknolojiler: Java, SpringBoot, HTML, CSS, JS, NetBeans IDE.

ELYSIUM SHOP/TALYAF STORE — OCA 2024 - DEVAM EDİYOR
Co-Founder

- İngiltere merkezli e-ticaret girişimi olarak Shopify platformu üzerinden çeşitli ürünlerin satışını gerçekleştirdiğimiz ve kurucu ortaklığını üstlendiğim bir şirketimiz var. Müşteri memnuniyetini en üst seviyede tutmaya gayret ediyoruz. Mağazalar, şuanda Türkiye üzerinden satışlarını gerçekleştirmekte ve daha sonrasında globale açılma hedefindedir.