T.C
İSTANBUL GEDİK ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

OPTİMİZASYON VE YAPAY ZEKÂ ALGORİTMALARI KULLANARAK MENÜ PLANLAMA YAZILIMI GELİŞTİRİLMESİ

YÜKSEK LİSANS TEZİ

Shahmirzali HUSEYNOV

Yapay Zeka Mühendisliği Anabilim Dalı

Yapay Zeka Mühendisliği Tezli Yüksek Lisans Programı

MAYIS 2024
İSTANBUL

T.C
İSTANBUL GEDİK ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

OPTİMİZASYON VE YAPAY ZEKÂ ALGORİTMALARI KULLANARAK MENÜ PLANLAMA YAZILIMI GELİŞTİRİLMESİ

YÜKSEK LİSANS TEZİ

Shahmirzali HUSEYNOV
210039004
0009-0000-2525-3029

Yapay Zeka Mühendisliği Anabilim Dalı

Yapay Zeka Mühendisliği Tezli Yüksek Lisans Programı

Tez Danışmanı: Doç. Dr. Fatih TARLAK

İstanbul 2024

T.C.
İSTANBUL GEDİK ÜNİVERSİTESİ
Lisansüstü Eğitim Enstitüsü Müdürlüğü

Jüri Tez Onay Formu

27.05.2024

LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ MÜDÜRLÜĞÜ

Bu çalışma 27.05.2024 tarihinde aşağıdaki jüri tarafından Yapay Zeka Mühendisliği Anabilim Dalı, Yapay Zeka Mühendisliği (Tezli Yüksek Lisans) Programı Yüksek Lisans Tezi olarak kabul edilmiştir.

TEZ JÜRİSİ

Doç. Dr. Fatih TARLAK
Danışman
Gebze Teknik Üniversitesi

Dr. Öğr. Üyesi Aytaç Uğur YERDEN
Üye (İmza)
İstanbul Gedik Üniversitesi

Prof. Dr. Bahaddin SİNSOYSAL
Üye (İmza)
İstanbul Gedik Üniversitesi

YEMİN METNİ

Yüksek Lisans Tezi olarak sunduğum “Optimizasyon ve Yapay Zekâ Algoritmaları Kullanarak Menü Planlama Yazılımı Geliştirilmesi’’ başlıklı bu çalışmanın, bilimsel ahlak ve geleneklere uygun şekilde tarafımdan yazıldığını, bu tezdeki bütün bilgileri akademik ve etik kurallar içinde elde ettiğimi, yararlandığım eserlerin tamamının kaynaklarda ve dipnotlarda gösterildiğini ve çalışmamın içinde kullanıldıkları her yerde bunlara atıf yapıldığını, patent ve telif haklarını ihlal edici bir davranışımın olmadığını belirtir ve bunu onurumla doğrularım. (27/05/2024)

Shahmirzali HUSEYNOV

ÖNSÖZ

Bu çalışmanın tüm süreçlerinde yol gösteren değerli tez danışmanım Doç. Dr. Fatih TARLAK'a ve desteğini her zaman hissettiğim sevgili aileme teşekkür ederim.

Mayıs 2024 Shahmirzali HUSEYNOV

İÇİNDEKİLER

Sayfa No:

KISALTMALAR

DP	: Doğrusal Programlama
KTDP	: Karışık Tamsayılı Doğrusal Programlama
DVM	: Destek Vektör Makineleri
REST	: Representational State Transfer (Temsili Durum Aktarım)
API	: Application Programming Interface (Uygulama programlama arayüzü)
http	: Hypertext Transfer Protocol (Hiper Metin Transfer Protokolü)
WEB	: World Wide Web (Dünya Çapında Ağ)
SQL	: Structured Query Language (Yapılandırılmış Sorgu Dili)
BeBİS	: Beslenme Bilişim Sistemleri Programı

ŞEKİL LİSTESİ

Sayfa No:

OPTİMİZASYON VE YAPAY ZEKÂ ALGORİTMALARI KULLANARAK MENÜ PLANLAMA YAZILIMI GELİŞTİRİLMESİ

ÖZET

Günümüzde, bireylerin sağlık ve beslenme bilincinin giderek arttığı bir dönemdeyiz. Bu durum, gıda sektöründe kişiye özel beslenme önerilerinin sağlanmasının önemini artırmakta ve yemek israfının azaltılması gerekliliğini ortaya koymaktadır. Çalışmada, demografik veriler ve yemek tercihleri gibi kullanıcı bilgileri toplanarak bu veriler yapay zeka modelleri ile analiz edilmiştir. Özellikle, Rastgele Orman algoritması kullanılarak geliştirilen model, kullanıcıların gelecekteki tercihlerini tahmin etmek ve menü planlama sürecinde eğitilmiştir. Modelin doğruluğunu ve performansını değerlendirmek amacıyla çeşitli testler gerçekleştirilmiş ve sonuçlar, yapay zeka ve optimizasyon tekniklerinin birlikte kullanılmasıyla kullanıcı odaklı menü planlarının başarıyla oluşturulabildiğini göstermiştir. Bu sayede, hem çalışan memnuniyetinin arttığı hem de yemek israfının önemli ölçüde azaldığı gözlemlenmiştir. Ayrıca, veri setinin büyüklüğüne bağlı olarak modelin performansında bazı zorluklar yaşanmış ve daha nitelikli ve temsili veri kullanmanın gerekliliği vurgulanmıştır.

Geliştirilen model, catering firmaları ve toplu beslenme hizmeti sunan diğer kuruluşlar için, yemek israfını azaltırken çalışan memnuniyetini artıran yenilikçi çözümler sunmaktadır. Gelecekte yapılacak çalışmalarla modelin daha da geliştirilmesi ve farklı sektörlerde de uygulanabilir hale getirilmesi hedeflenmektedir.

Anahtar Kelimeler: *Yapay Zeka, Optimizasyon Algoritmaları, Doğrusal Programlama, Kişiselleştirilmiş Menü, Catering*

DEVELOPING MENU PLANNING SOFTWARE USING OPTIMIZATION AND ARTIFICIAL INTELLIGENCE ALGORITHMS

ABSTRACT

In today's world, we are experiencing a period where individuals' awareness of health and nutrition is increasingly growing. This situation emphasizes the importance of providing personalized nutrition recommendations in the food industry and highlights the necessity of reducing food waste. In this study, demographic data and food preferences were collected from users and analyzed using artificial intelligence models. Specifically, a model developed using the Random Forest algorithm was trained to predict users' future preferences and guide the menu planning process. Various tests were conducted to evaluate the model's accuracy and performance, and the results demonstrated that using artificial intelligence and optimization techniques together successfully creates user-focused menu plans. Consequently, it was observed that employee satisfaction increased while food waste significantly decreased. Additionally, challenges related to the size of the dataset were encountered, underscoring the need for more qualitative and representative data.

The developed model offers innovative solutions that increase employee satisfaction while reducing food waste for catering companies and other institutions providing mass dining services. Future studies aim to further develop the model and make it applicable in different sectors.

Keywords: *Artificial Intelligence, Optimization Algorithms, Linear Programming, Personalized Menu, Catering*

1. GİRİŞ

Günümüz dünyasında bireylerin sağlık ve beslenme bilincinin artması ile birlikte, kişiye özel beslenme önerilerinin önemi giderek artmaktadır. Bu çalışmanın temel amacı, yapay zeka ve optimizasyon algoritmaları kullanarak kişiselleştirilmiş menü planları oluşturmaktır. Bu süreç, çalışanların kişisel verileri ve beslenme tercihlerine dayanarak, sağlık ve beslenme hedeflerine uygun, dengeli menü önerileri sunmayı amaçlamaktadır. Genel beslenme önerileri her bireyin özel ihtiyaçlarını tam olarak karşılamayabilir; bu nedenle, bu çalışma, bireysel beslenme gereksinimlerini doğru bir şekilde tespit ederek, bunlara uygun menü planları oluşturmayı hedeflemektedir.

Teknolojik yenilikler, sağlık ve beslenme alanında kişiselleştirilmiş çözümler sunarak bireylerin yaşam kalitesini artırmakta önemli bir role sahiptir. Özellikle yapay zeka ve optimizasyon algoritmaları, kişisel beslenme ve sağlık hedeflerine ulaşmada kritik araçlar olarak öne çıkmaktadır. Bu teknolojiler, geniş veri kümelerini analiz ederek, kullanıcıların sağlık durumlarına ve beslenme tercihlerine uygun özelleştirilmiş diyet planları sunabilir (Vesna, Antoska, Knights, 2022**)**. Yapay zeka ve optimizasyon algoritmalarının kullanılması, kişisel beslenme tercihlerine dayalı bireyselleştirilmiş menü planları sunma potansiyeline sahiptir. Bu yaklaşım, her çalışanın yaşam tarzı ve beslenme alışkanlıklarına uygun, bilimsel temellere dayanan çözümler sunabilir. Araştırma, yapay zeka ve optimizasyon tekniklerinin uygulanmasını göstererek, çalışanların beslenme planlamasını daha bilinçli ve etkili bir şekilde yapmalarına olanak tanır (G. Arvindaraj, B. Manikandan, 2023).

Catering firmaları, özellikle büyük şirketler için toplu yemek hizmeti sunan kuruluşlar olarak, çalışanların beslenme ihtiyaçlarını karşılama konusunda önemli bir role sahiptir. Ancak, bu firmalar genellikle standart menüler sunarak geniş kitlelere hitap etmeye çalışmakta, bu da çalışanların bireysel beslenme ihtiyaçlarını göz ardı etmelerine yol açmaktadır. Bu durum, hem çalışan memnuniyetinin azalmasına hem de yemek israfının artmasına neden olmaktadır (S. Hou, D. Zhu and J. Xu, 2022) (Sambudi Hamali, 2019).

Bu tez, catering firmalarının sunduğu hizmetleri daha verimli ve israfı minimuma indirgeyen bir şekilde sunmalarını sağlamayı amaçlamaktadır. Çalışanların beslenme alışkanlıkları ve tercihleri dikkate alınmadığında, bazı yemekler tüketilmemekte ve bu da hem ekonomik kayıplara hem de çevresel sorunlara yol açmaktadır (Melanie Speck, 2022) (Christopher Malefors, 2022). Catering firmalarının işleyişinde karşılaşılan temel sorunlardan biri, büyük miktarlarda yemeğin israf edilmesidir. Bu israf, hem ekonomik kayıplara yol açmakta hem de çevresel açıdan olumsuz etkiler yaratmaktadır (Erdem, M. B, 2023). Yapay zeka ve optimizasyon algoritmaları kullanılarak geliştirilen modelimiz, çalışanlar için menüler oluşturmakta ve bu sayede çalışan memnuniyetini artırırken, yemek israfını da önemli ölçüde azaltmayı ve çalışanların memnuniyetini artırmayı amaçlamaktadır.

Catering firmalarının sunduğu hizmetlerde, çalışanların bireysel ihtiyaçlarına yönelik çözümler sunulması, iş verimliliğini artırmakta ve çalışanların işyerindeki motivasyonunu olumlu yönde etkilemektedir. Bu tezde geliştirilen model, sadece catering firmaları için değil, aynı zamanda diğer toplu beslenme hizmeti sunan kuruluşlar için de uygulanabilir bir çözüm sunmaktadır. Böylece, geniş bir kitleye hitap eden toplu beslenme hizmetlerinin daha verimli ve sürdürülebilir bir şekilde sunulması mümkün olacaktır.

Sonuç olarak, bu tez çalışması, yapay zeka ve optimizasyon tekniklerinin birleşimiyle, catering firmaları için kişiselleştirilmiş menü planları oluşturmanın mümkün olduğunu göstermektedir. Çalışma, hem çalışan memnuniyetini artırmayı hem de yemek israfını azaltmayı hedefleyen yenilikçi bir yaklaşım sunmaktadır. Geliştirilen model, oluşturulan veri setleri üzerinde başarılı sonuçlar vermiş ve uygulamada etkili bir çözüm olduğunu gözlemlenmiştir. Gelecekte yapılacak çalışmalarla, modelin daha da geliştirilmesi ve farklı sektörlerde de uygulanabilir hale getirilmesi hedeflenmektedir.

2. KAVRAMSAL ÇERÇEVE

2.1 Optimizasyon Nedir?

Hepimiz günlük yaşamımızda sahip olduğumuz sınırlı kaynaklardan en iyi şekilde yararlanmaya çalışıyoruz. Bu aslında yaşam biçimidir. Elektrikli cihazlardan mekanik arabalara kadar etrafımızdaki her şey optimizasyonu kullanıyor.

Optimizasyon, belirli kısıtlamalar altında bir hedef veya amaç fonksiyonunu en iyi şekilde gerçekleştirmek amacıyla verilen parametreleri ayarlamak için kullanılan bir matematiksel süreçtir. Temel amacı, verilen bir sorun için en uygun çözümü bulmak olan optimizasyon, mühendislikten finansa, lojistikten yapay zekaya kadar birçok alanda uygulanır (Precup, R., Hedrea, 2020).

Bu süreç, genellikle bir amaç fonksiyonunun maksimize edilmesi veya minimize edilmesi gereken durumlarda kullanılır. Amaç fonksiyonu, optimizasyonun hedeflediği ana hedeftir ve bu fonksiyon, sorunun matematiksel bir ifadesi olarak düşünülebilir. Soruna bağlı olarak, bu fonksiyon bir maliyetin, sürenin, mesafenin veya diğer bir ölçütün en aza indirilmesi veya bir gelirin, verimliliğin, performansın en üst düzeye çıkarılmasını hedefleyebilir. Örneğin, bir fabrikada üretim maliyetlerini minimize etmek, bir lojistik şirketinde en kısa teslimat rotalarını belirlemek veya finansal portföy yönetiminde riski azaltırken getiriyi maksimize etmek gibi durumlarda optimizasyon teknikleri devreye girer (X. Gong, 2022).

Optimizasyon problemleri, genellikle kısıtlarla birlikte gelir. Bu kısıtlar, çözümün uygun olması gereken çeşitli şartları ifade eder ve genellikle eşitlikler veya eşitsizlikler şeklinde modellenir. Problemin karmaşıklığı, kısıtların ve değişkenlerin sayısına bağlı olarak artabilir (X. Gong, 2022).

Optimizasyon yöntemleri arasında Doğrusal Programlama (DP), Tamsayı Programlama, Dinamik Programlama, Karma Tamsayılı Doğrusal Programlama (KTDP) ve Heuristik Yaklaşımlar gibi çeşitli teknikler bulunur. Her bir teknik, özgün matematiksel yapıları ve algoritmaları ile farklı türdeki optimizasyon problemlerine çözümler sunar (A. Hasnaoui, 2022).

2.1.1 Doğrusal Programlama Tarihçesi

DP, diğer matematik alanlarına kıyasla nispeten kısa bir geçmişe sahiptir. Bu durum, çoğu problemin bilgisayar hesaplamaları olmadan çözülmesinin zor olmasından kaynaklanmaktadır. DP problemlerine yönelik uygulanabilir bir çözüm sunma girişimlerinden biri, Joseph Fourier tarafından 1827 yılında yayımlanan bir yöntem ile yapılmıştır. Ancak Fourier'in algoritması, daha büyük sistemler için en azından üssel zaman gerektirdiğinden verimsizdi; yüz veya daha fazla değişken içeren problemlerin çözümü için imkansız miktarda zaman gerektirecekti. Başka çözüm girişimleri de oldu ancak bunlar sınırlı başarı gösterdi. Alanın önemli gelişmeleri ancak 20. yüzyılın ortalarına kadar gerçekleşmedi (Chandru, V., & Rao, M, 1998).

DP alanına yapılan en bilinen ve önemli katkı, George Dantzig tarafından 1947 yılında gerçekleştirildi. Dantzig, Pentagon'da askeri danışman olarak çalışırken DP problemlerini çözmek için simpleks yöntemini geliştirdi. Simpleks yöntemi, DP problemlerine zaman açısından verimli bir çözüm sunan ilk pratik algoritmaydı. Ancak, bu yöntemin ortaya çıkışı sırasında büyük problemleri çözme kapasitesi hala sınırlıydı. Gerçekten de, algoritmanın büyük bir sistem üzerindeki ilk uygulamalarından biri, bir Kart Programlanabilir Hesaplayıcı'ya 8 saat boyunca kart beslenmesini gerektirdi. Bu sınırlamalara rağmen, simpleks algoritması, önceki yöntemlerden çok daha verimli bir şekilde DP problemlerini çözme olanağı sağladı. İlginç bir şekilde, DP adı, gerçekte bilgisayar programlarıyla pek ilgili değildir ve aslında askeri terminolojide lojistik ve harekat planlarına atıfta bulunan "program" kelimesinden türemiştir (Chandru, V., & Rao, M, 1998).

Simpleks yöntemi başlangıçta askeri lojistikte çeşitli iyileştirmeler yapmayı amaçlasa da, çok geçmeden ticari uygulamalarda, örneğin petrol rafinajı ve karışımı gibi alanlarda kullanılmaya başlandı. Takip eden on yılda simpleks yöntemi geliştirildi ve doğrusal olmayan programlama ile tam sayılı programlama gibi ilgili alanlar ortaya çıktı (Dantzig, G. 1982).

Günümüzde, DP, büyük veri miktarlarına kolay erişim sayesinde yönetim ve lojistik zorluklar için her zamankinden daha faydalı hale gelmiştir. İyileştirilmiş algoritmalar ve bilgisayar gücündeki artışlarla birlikte, DP daha büyük ve daha karmaşık problemleri çözebilmektedir. Şu anki temel zorluk, uygulanabilir durumları

doğru bir şekilde modellendirme ve mevcut verileri etkin bir şekilde kullanma yeteneğidir (Atlanta, Chakraborty, 2020).

2.1.2 Doğrusal Programlama Uygulamaları ve Metodolojisi

Matematiksel bir optimizasyon modeli, bir problemi nicel olarak ifade etmek ve en iyi çıkar için en uygun çözümü bulmak için kullanılan bir karar aracıdır. DP optimizasyonu, bir dizi doğrusal denklem ve ya eşitsizlikten istenen çözümün elde edildiği bir süreçtir. DP optimizasyonunu kullanarak bir problemi çözmek için önce bir model oluşturulur ve ardından birkaç değişken seti tanımlanır. Model, bir amaç fonksiyonu ve amaç fonksiyonunu optimize etmek için bir dizi kısıtlama fonksiyonundan oluşur. Tanımlanan değişkenler, modeli takip eden bir dizi doğrusal denklemi formüle eder ve son olarak hedef çözümü bulmak için bu denklemler çözülür. Zamandan ve iş gücünden tasarruf etmek için bu sürece yardımcı olacak birçok yazılım aracı mevcuttur (Preeti, Tewari, 2022).

İkinci Dünya Savaşı sırasında askeri harcamaları ve sonuçları planlamak ve ilişkilendirmek için DP'nın geliştirilmesi ve uygulanmasından bu yana, yıllar boyunca giderek daha fazla uygulamada kanıtlanmış ve uygulanmıştır. Günümüzde DP endüstriyel ve bilimsel araştırma alanlarında yaygın olarak kullanılmaktadır. Endüstrilerde maliyet fayda ilişkilerini analiz etmek, tarımda verimi maksimize etmek, ticari havacılıkta rotaları tasarlamak, telekomünikasyonda trafik yoğunluğunu tahmin etmek ve diğer birçok alanda etkili bir araç haline gelmiştir. Ayrıca, ulusal politikaların belirlenmesinden insan davranışlarının analizine kadar uygulama alanı bulan oyun teorisi gibi sofistike tekniklerin geliştirilmesine de katkıda bulunmuştur (Christian, Hoover, 2023).

DP'nın en önemli avantajlarından biri esnekliğidir. Doğrusal denklemler ve eşitsizlikler çok çeşitli kısıtlamaları ve hedefleri temsil edebilir ve bu da onları çok çeşitli problemlere uygulanabilir hale getirir. Ek olarak DP problemleri, simpleks yöntemi, primal-dual yöntemi ve iç nokta yöntemi dahil olmak üzere çeşitli algoritmalar kullanılarak çözülebilir. Bu, yöntemin eldeki problemin özel ihtiyaçlarına uyarlanabileceği anlamına gelir ve bu da onu optimizasyon problemlerini çözmek için güçlü bir araç haline getirir.

İsmine rağmen DP sadece doğrusal problemlerle sınırlı değildir. Doğrusal olmayan birçok problem, doğrusallaştırma gibi belirli matematiksel teknikler

uygulanarak doğrusal forma dönüştürülebilir, böylece DP yöntemleri kullanılarak çözülebilir.

2.1.3 Doğrusal Programlama Modeli

Hedef Fonksiyonu: z = $c^T x$

Kısıtlamalar: $Ax \leq$ b

Değişkenlerin Sınırlamaları: $x \geq 0$

Öyle bir vektör x bulunmalıdır ki; $c^T x$ 'yi maksimize eder veya minimize eder, şu koşula tabidir $Ax \leq$ b ve $x \geq 0$.

Burada, x vektörünün bileşenleri $x_1, x_2, \ldots, x_n$, **karar değişkenleridir** ve bir doğrusal çözücü tarafından belirlenmelidir ki amaç fonksiyon $c^T x$ maksimize veya minimize edilsin. Doğrusal programın kısıtlamaları $Ax \leq$ b şeklinde ifade edilir. Ek olarak, vektör x'in tüm bileşenleri 0 veya daha büyük olmalıdır. Vektör x , n×1 boyutunda bir sütun vektörüdür. Vektör c de n×1 boyutunda bir sütun vektörüdür. Optimizasyon problemindeki kısıtlamaların sayısına bağlı olarak, A bir m×n boyutunda bir matristir ve vektör b ise 1×m boyutunda bir satır vektörüdür. (Neşe Yalçın, 2005)

DP, çeşitli biçimlerde ifade edilebilir. Örnek olarak standart ve kanonik formdaki yapılarını inceleyeceğiz. İkilik(Dual) teoride kanonik form genellikle tercih edilirken, standart form simpleks metodunun kullanıldığı durumlarda daha pratiktir. Her iki formun öne çıkan özellikleri ise şöyledir:

DP problemi aşağıdaki özellikleri taşıyorsa, standart formda olduğu kabul edilir:

- Eğer tüm karar değişkenleri negatif değer alamıyorsa,
- Amaç fonksiyonu ya maksimizasyon ya da minimizasyon amaçlı ise,
- Kısıtlama denklemlerinin sağ tarafındaki değerler negatif değilse,
- Ve eğer tüm kısıtlamalar denklem formunda ise, bu durumda karar değişkenlerinin yalnızca negatif olmaması şartı aranmaz.

DP probleminin kanonik formda olduğunu belirtmek için aşağıdaki özelliklerin karşılanması gerekir:

- Tüm karar değişkenlerinin negatif olmaması,
- Amaç fonksiyonunun maksimizasyon amaçlı olması,
- Tüm kısıtlamaların $\leq$ tipte olması gerekir (ancak burada değişkenlerin negatif olmaması koşulu hariçtir).

DP problemleri, uygun dönüşümler yapılarak hem standart hem de kanonik formda ifade edilebilir. Bu, problemin doğasına ve çözüm metodolojisine göre değişkenlik gösterir.

DP modelinde istenen sonuçların elde edilebilmesi için modelin amacının açık ve nicel bir şekilde ifade edilmesi zorunludur. Amaç fonksiyonu, $x_j\,(j = 1, 2, 3, \ldots, n)$ probleme ilişkin sürecin potansiyel performansını belirler ve bu performansı maksimize veya minimize etmeyi hedefler. Bu nedenle, karar değişkenlerinin değerleri belirlenir. Amaç fonksiyonu, aşağıdaki gibi ifade edilir:

$$\text{Max./Min. } Z = c_1x_1 + c_2x_2 + \ldots + c_jx_j + \ldots + c_nx_n \qquad (2..1)$$

DP modelinin kurulumu, öncelikle karar değişkenlerinin tanımlanması ile başlar. Amaç fonksiyonundaki x_j değişkenleri, karar değişkenleri olarak adlandırılır ve genellikle alınacak kararlara ilişkin faaliyetlerin düzeyini gösterir. Bu değişkenler en çok $x_j\,(j = 1, 2, 3, \ldots, n)$ simgeleriyle ifade edilir. Ayrıca, karar değişkenlerinin değerleri, kısıtlamalar kümesini tatmin etmelidir. (Robert J. Vanderbei, 2020)

Kısıtsız bir DP problemi yalnızca bir amaç fonksiyonu içerir. Amaç fonksiyonu, çok değişkenli lineer bir fonksiyon olup, amaç fonksiyonu olarak isimlendirilir. Kullanılan kaynaklar her zaman sınırlı olduğundan, kısıtsız bir DP problemi pratikte nadiren karşılaşılır. Bu nedenle, karar değişkenlerinin miktarı da bu kısıtlamalar altında sınırlıdır. Bu kısıtlamalar altında amaç fonksiyonunun sağlanması önemlidir. Kaynak miktarları $b_i\,(i = 1, 2, 3, \ldots, m)$ ve ürünlerin çeşitli üretim yolları veya teknoloji katsayıları a_{ij} ile ifade edilirse, kısıtlayıcı denklem takımı aşağıdaki şekilde gösterilecektir:

$$
\begin{aligned}
&a_{11}x_1 + a_{12}x_2 + \ldots + a_{1j}x_j + \ldots + a_{1n}x_n \leq b_1 \\
&a_{21}x_1 + a_{22}x_2 + \ldots + a_{2j}x_j + \ldots + a_{2n}x_n \leq b_2 \\
&\vdots \qquad\qquad \vdots \qquad\qquad \vdots \\
&a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{ij}x_j + \ldots + a_{in}x_n \leq b_i \\
&\vdots \qquad\qquad \vdots \qquad\qquad \vdots \\
&a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mj}x_j + \ldots + a_{mn}x_n \leq b_m
\end{aligned}
\tag{2.2}
$$

Karar değişkenlerinin katsayıları, DP kısıtlamalarında kullanılan teknolojiyi yansıttığından, bu katsayılara teknolojik katsayılar denir. Kaynak miktarları, kısıtlama fonksiyonları çerçevesinde her zaman sınırlı olmayabilir. Bu nedenle, bazen karar değişkenlerinin değerlerinin istenen limitlerin üstünde veya tam olarak eşit olması mantıklı olabilir.

DP'nın temel bileşenlerinden biri, negatif olmayan kısıtlamalardır. Bu kısıtlamalar, özellikle x_j olarak ifade edilen karar değişkenlerinin, birim üretim maliyeti veya birim taşıma maliyeti gibi değerler olması nedeniyle, bu değişkenlerin değerlerinin kesinlikle pozitif olmasını gerektirir.

2.1.4 Doğrusal Programlamanın Varsayımları

Doğrusallık:

Tüm DP modelleri, modelin tüm elemanlarının doğrusal olmasını gerektirir; bu duruma oransallık varsayımı denir. Bu varsayım, modelin amaç ve kısıtlayıcı fonksiyonları arasındaki ilişkiyi ifade eder, ki her bir aktivite, amaç fonksiyonuna katkısının, faaliyetin düzeyine orantılı olmasını sağlar (Kudak, 2007).

Toplanabilirlik:

DP, her bir fonksiyonun (ister amaç fonksiyonu ister kısıtlamaların sol tarafındaki fonksiyon) ilgili aktivitelerin bireysel katkılarının toplamı olduğunu varsayar. Bu, kısıtlamanın sol tarafındaki fonksiyonun, çeşitli üretim aktivitelerinden kaynaklanan girdilerin toplamının, her bir işlem için kullanılan girdilerin toplamına eşit olduğunu gösterir (Neşe Yalçın, 2005).

Bölünebilirlik:

Bölünebilirlik varsayımı, her karar değişkeninin kesirli değerler alabilmesine olanak tanır. Bu, değişkenlerin yalnızca tam sayı değerleri almakla sınırlı olmadığını gösterir (Neşe Yalçın, 2005). Her karar değişkeni belirli aktivitelerin düzeyini temsil ettiğinden, bu aktivitelerin kesirli düzeylerde yürütülmesi mümkündür.

Determinizm (Kesinlik):

Kesinlik prensibi, DP modellerinde her parametrenin, amaç fonksiyonunun katsayıları (c_j), kısıtlamaların sağ taraf değerleri (b_i) ve teknoloji katsayıları ($a_{i\,j}$) gibi, kesin bilindiği varsayımına dayanır. Bu, parametrelerin sabit değerler olduğu ve modelin deterministik (kesin) bir yapıda olduğu anlamına gelir. (Kudak, 2007).

Ancak, bu varsayım gerçek dünya uygulamalarında nadiren karşılaşılır çünkü DP modelleri, faaliyetlerin gelecekteki durumlarını öngörmek amacıyla tasarlanır ve bu durumda, parametre değerlerinin bazı belirsizlikler içermesi kaçınılmazdır.

Tarih boyunca ve günümüzde de birçok gerçek dünya problemi, bu tür modeller kullanılarak ve bilgisayar destekli çözümlerle başarıyla formüle edilip çözülmüştür. Bununla birlikte, modelin doğruluk ve bölünebilirlik varsayımlarının gerçek dünyadaki ilişkileri yeterince yansıtmadığına dair eleştiriler bulunmaktadır (Kudak, 2007).

Gerçek dünya uygulamalarında, doğruluk ve bölünebilirlik varsayımları bazen geçerliliğini yitirebilir. Pratikte, bu varsayımların sıklıkla karşılanamaması nedeniyle, bulanık DP gibi yaklaşımlar bu eksiklikleri giderme amacıyla değerlendirilmektedir.

2.1.5 Doğrusal Programlama Probleminin Çözümünde Kullanılan Genel Tanımlar

DP problemleri çözümünde sıklıkla başvurulan bazı temel kavramlar vardır:

Katsayılar

Katsayılar, DP modelindeki amaç fonksiyonu ve kısıtlama fonksiyonlarında yer alan değişkenlerin önüne yazılan sayılardır ve bu sayılar, ilgili değişkenlerin model üzerindeki etkilerini veya ağırlıklarını temsil eder. Her bir katsayı, problemdeki bir birim değişkenin amaç fonksiyonuna veya kısıtlamalara ne kadar

katkıda bulunduğunu nicel olarak ifade eder, böylece modelin çözümü sırasında değişkenlerin her birinin önemi ve rolü matematiksel olarak belirlenmiş olur. Bu katsayılar, modelin gerçek dünya durumunu doğru bir şekilde yansıtmasında kritik rol oynar (Daniel G. Espinoza, 2006), çünkü optimizasyon sürecinde hangi kararların maksimum fayda veya minimum maliyeti sağlayacağını doğrudan etkilerler.

Uygun Bölge

DP çerçevesinde, tüm kısıtlamaları karşılayan $x_1, x_2, \ldots, x_n$ karar değişkenlerinin oluşturduğu değerler kümesine uygun bölge denir. Uygun bölge, DP'nın tüm potansiyel çözümlerini barındırır ve bu bölge, problemin çözüm alanını tanımlar (Daniel G. Espinoza, 2006).

Köşe Noktaları

Uygun bölgenin yapısal özellikleri gereği, bu bölgede sınırlı sayıda özel nokta bulunur; bu noktalar köşe noktaları olarak adlandırılır. Köşe noktaları, DP çözümlerinde kritik öneme sahiptir çünkü potansiyel optimal çözümler genellikle bu noktalarda bulunur.

Uygun Çözüm

Bir DP problemi için, tüm kısıtlamaları karşılayan karar değişkenlerinin değerlerinin belirli bir kombinasyonuna uygun çözüm denir. Uygun bir çözüm, uygun bölgenin herhangi bir noktasında yer alabilir ve bu nokta bir köşe noktası ya da iç nokta olabilir. Uygun çözüm, problemi tanımlayan kısıtlamalar çerçevesinde mümkün olan çözüm setlerini ifade eder (Christian, Hoover, 2023).

Optimal Çözüm

Bir DP modelinde, amaç fonksiyonunun maksimum veya minimum değere ulaşması gereken durumlarda elde edilen çözüm, optimal çözüm olarak adlandırılır (Christian, Hoover, 2023).

Bozulan Çözüm

Genel temel çözümde, bir veya daha fazla temel değişkenin değeri sıfıra eşitse, bu durum bozulan çözüm olarak tanımlanır. Bu tür bir çözümde, temelde yer almayan diğer değişkenlerin çözüm değerleri sıfırdır.

Uygulanamayan Çözüm

Uygulanamayan Çözüm terimi, DP ve genel olarak optimizasyon problemlerinde kullanılır. Bu terim, problem için tanımlanan kısıtların hiçbirinin karşılanmadığı bir durumu ifade eder. Başka bir deyişle, kısıtlama denklemleri ve eşitsizlikleri ile belirlenen şartlar aynı anda yerine getirilemeyen herhangi bir çözüm seti, icra edilemez çözüm olarak adlandırılır (Christian, Hoover, 2023).

Uygulanamayan Çözümün Özellikleri:

Kısıtların Çatışması: Uygulanamayan çözümler genellikle, problemdeki kısıtların birbiriyle çatışması sonucu ortaya çıkar. Örneğin, bir kaynağın aynı anda iki farklı yerde kullanılması gerektiğini belirten çelişkili kısıtlar bu duruma yol açabilir.

Modelleme Hataları: Uygulanamayan çözümler bazen, problemin yanlış modellenmesinden kaynaklanır. Örneğin, gerçek dünya koşullarını yanlış yansıtan veya eksik kısıtlar içeren bir model, icra edilemez çözümlere yol açabilir.

Uygulanamayan çözüm durumları, modelin veya problem tanımının gözden geçirilmesi gerektiğini gösterir. Bu durum genellikle problemi çözen kişi için bir uyarı işareti olarak kabul edilir ve problemin kısıtlarının, amaç fonksiyonunun veya diğer parametrelerinin yeniden değerlendirilmesini gerektirebilir (Kudak, 2007).

Özetle ifade edildiğinde, DP standartlarında belirtildiği üzere, kısıtları karşılayan her bir vektör (x) için temel çözüm elde edilir. Eğer $x \geq 0$ koşulunu da sağlıyorsa, bu çözüm aynı zamanda temel uygun çözüm olarak adlandırılır ve bu tür çözümlerin oluşturduğu küme olası çözüm bölgesi olarak tanımlanır. Bu alanda, amaç fonksiyonunun en iyi performansı sağlanır şeklinde tanımlanan temel uygun çözüm, optimal çözüm vektörü olarak adlandırılır.

DP probleminin çözümü, bu tür bir dışbükey bölge içinde gerçekleşir ve problemde varsa optimal çözüm, bu dışbükey kümenin uç noktalarından birinde bulunur. n değişken ve m kısıt içeren bir DP problemi için uç noktaların sayısı:

$$C_m^{m+n} = \frac{(m+n)!}{m!\,n!} \tag{3.3}$$

formülü ile hesaplanır.

2.1.6 Doğrusal Programlamanın Çözüm Yöntemleri

DP'nın çözümünde esas olarak iki temel yöntem kullanılır: Grafik Yöntem ve Cebirsel Yöntem. Grafik yöntemi, özellikle iki veya üç değişkenli DP problemleri için uygundur. Ancak, gerçek dünya problemlerinde değişken sayısı genellikle üçten fazla olduğundan, büyük ölçekli problemler için daha karmaşık cebirsel yöntemler tercih edilir. Bu durumda, simpleks yöntemi devreye girer. Simpleks yöntemi, problemin grafiksel çözümündeki mantığı genişleterek, daha fazla değişken içeren problemleri çözebilmek için kullanılır.

Grafik yöntemi, kısıtları ve amaç fonksiyonunu bir grafik üzerinde çizerek, elde edilen uygun bölge içinde optimal çözümü bulmayı amaçlar. Bu süreçte, kısıtlar grafiği çizilir, uygun bölge belirlenir ve bu bölgedeki potansiyel optimal noktalar incelenir. Kısıtların oluşturduğu yarı uzayların kesişimi ile uygun bölge tanımlanır ve bu bölgede, amaç fonksiyonunu maksimize veya minimize eden noktalar aranır.

Simpleks yöntemi, G.B. Dantzig tarafından 1947 yılında geliştirilen ve iteratif bir cebirsel süreç kullanarak çözüme ulaşan bir yöntemdir. Bu yöntem, DP problemlerini bir başlangıç tablosundan başlayarak, kısıtlar ve amaç fonksiyonu arasındaki ilişkileri dikkate alarak adım adım çözer. Her iterasyonda, amaç fonksiyonunun değeri iyileştirilir ve en iyi çözüme doğru ilerlenir. Çözüm sürecinde, aylak ve yapay değişkenler gibi araçlar kullanılarak, kısıtların eşitlik formuna dönüştürülmesi sağlanır ve temel çözümlere ulaşılır. Aylak değişkenler, kullanılmayan kapasiteleri temsil ederken, yapay değişkenler ise çözüm matrisinin matematiksel gerekliliklerini karşılamak için eklenir.

Bu yöntemler, DP problemlerinin geniş bir yelpazesini kapsar ve problemin niteliğine, değişken sayısına ve karmaşıklığına bağlı olarak uygun yöntem seçilir. Her iki yöntem de, özellikle endüstriyel ve ekonomik analizlerde önemli bir rol oynar, sağladıkları çözümlerle karar verme süreçlerinde kritik öneme sahiptir.

2.1.7 Karışık Tamsayı Doğrusal Programlama

Karışık Tamsayı Doğrusal Programlama (KTDP), DP modellerinin bir alt dalı olarak, hem sürekli hem de tamsayı değişkenler içeren karmaşık optimizasyon problemlerini çözmek için kullanılır. KTDP, belirli bir amaç fonksiyonunu, lineer kısıtlamalar altında maksimize etmek veya minimize etmek üzere tasarlanmıştır. Bu yaklaşım, gerçek dünya senaryolarında sıkça rastlanan, hem kesikli (tamsayı) hem de

sürekli karar değişkenlerinin bir arada olduğu durumları modellemek için uygundur (Matthias Miltenberger, 2023). Örneğin, bir lojistik problemde, araçların sayısını (tamsayı) ve taşınacak malların miktarını (sürekli) aynı anda optimize etmek gibi.

KTDP, temel yapı olarak DP ile benzerlik gösterir çünkü her ikisi de amaç fonksiyonunu, belirli lineer kısıtlamalar dâhilinde optimize etmeyi hedefler. Ancak, KTDP'nın temel farkı, modelin tamsayı kısıtlamalarını da içermesi ve bu durumun çözüm stratejilerini önemli ölçüde değiştirmesidir. DP modelleri yalnızca sürekli değişkenlerle çalışırken, KTDP modelleri probleme tamsayı değişkenleri ekleyerek, karar verme sürecinde daha gerçekçi senaryoları ele alabilir.

Bu tamsayı değişkenler, genellikle "evet/hayır" kararlarını (0 veya 1 değerleri alan ikili değişkenler), sınırlı sayıda nesnenin seçimini veya belirli sayıda gruplandırma gibi senaryoları temsil eder (Matthias Miltenberger, 2023).. LP'de ise çözüm sürekli bir değerler kümesi üzerinde gezinirken, KTDP'de bu değerler belirli noktalarda sabitlenebilir, bu da modelin çözüm uzayını daha kesikli hale getirir ve genellikle çözümü daha zorlaştırır.

KTDP problemleri genellikle NP-zor kategorisine girer, bu da onların çözümünün zaman açısından DP'ye göre çok daha zor olabileceği anlamına gelir. KTDP'nın çözümünde kullanılan yaygın yöntemler arasında Branch-and-Bound, Branch-and-Cut ve Cutting Plane gibi teknikler bulunur. Bu yöntemler, problemin çözüm alanını sistematik bir şekilde keşfetmek ve potansiyel çözüm adaylarını etkili bir şekilde sınırlandırmak için tasarlanmıştır.

DP çözümleri genellikle daha hızlı ve kolay bulunurken, KTDP çözümleri, tamsayı değişkenlerinin eklenmesiyle daha karmaşık hale gelir. Bu karmaşıklık, KTDP'nın çeşitli uygulama alanlarında, özellikle endüstriyel mühendislik, lojistik, finans ve operasyon araştırmalarında tercih edilmesinin temel nedenlerinden biridir.

KTDP, bir amaç fonksiyonunu belirli lineer kısıtlamalar altında maksimize etmek veya minimize etmek için tasarlanmış bir matematiksel modeldir. Model, hem tamsayı hem de sürekli değişkenleri içerebilir, bu da KDTP'nın kapsamını ve uygulama alanlarını genişletir. Matematiksel olarak, bir KTDP problemi aşağıdaki gibi ifade edilir:

minimize (veya maximize) $\mathbf{c}^T\mathbf{x}$

Burada c katsayı vektörü ve x karar değişkenleri vektörüdür. Amaç, $\mathbf{c}^T\mathbf{x}$ ifadesini minimize etmek veya maksimize etmektir.

$$\mathbf{A}\mathbf{x} \leq \mathbf{b}$$

A matrisi, kısıtlamaların katsayılarını içerirken, b vektörü her bir kısıtlamanın sağ taraf değerlerini belirtir. Bu eşitsizlikler, çözüm uzayını tanımlar ve çözümün uygun olmasını sağlar.

$$x_j \in \mathbb{Z} \quad \text{for } j \in I, \quad x_j \geq 0 \quad \text{for } j \in J$$

Burada I kümesi, tamsayı değerleri alması gereken değişkenlerin indekslerini, J ise sürekli olabilen değişkenlerin indekslerini belirtir. Tamsayı değişkenleri genellikle belirli nesnelerin sayısını veya evet/hayır kararlarını temsil eder (Stavropoulos, A, 2012).

KTDP formülasyonu, problemin doğasına bağlı olarak değişken türleri ve kısıtlar açısından özelleştirilebilir. Örneğin, bazı durumlarda tüm değişkenler tamsayı olabilir veya bazı değişkenler ikili (0 veya 1 değerlerini alan) olabilir. Her bir özelleştirme, KDTP'nın esnekliğini ve çeşitli gerçek dünya problemlerine uygulanabilirliğini artırır.

2.2 Yapay Zeka ve Makine öğrenimi

Yapay zeka, çoğunlukla bilgisayar bilimleri ve matematik, istatistik ve veri bilimi, sinir bilimleri vb. gibi diğer bilimleri kapsayan, bilgisayar tabanlı, çok disiplinli bir çalışmadır. Yapay zekanın amacı, insan zekası gibi akıllı makineler yapmaktır. Akıl yürütme ve deneyimlerden öğrenme gibi insan zekasını kopyalayarak veya geliştirerek makinelere akıllı makineler yapma yeteneği kazandırmayı içeren genel disiplindir (Russell, 2003)

Makine öğrenimi, bilgisayarın sınıflandırma ve regresyon gibi sorunları çözmeye yönelik yöntemler geliştirmek üzere insan öğrenmesini taklit edecek şekilde programlandığı bir tür yapay zekadır. Makine öğreniminde yazılım, bir sorunun çözümü için gereken süreç ve adımların ayrıntılarını içerecek şekilde açıkça programlanmamıştır. Bunun yerine, örnek verilerden öğrenecek ve istenen sonuçların doğruluğunu giderek artıracak şekilde programlanırlar.

Makine öğrenmesi denetimli öğrenme ve denetimsiz öğrenme olmak üzere iki ana türe ayrılabilir. Denetimli öğrenme algoritmaları, kalıp bulma ve matematiksel modeller oluşturma gibi verilen görevi yerine getirebilmek için etiketlenmiş eğitim verilerine ihtiyaç duyar (Nasteski, 2017). Denetimli öğrenme hem sınıflandırma hem de regresyon problemlerinde kullanılabilir. Sınıflandırmada tahmin nominal değerleri alırken, regresyon algoritmadan sürekli sonuçlar elde etmeyi amaçlamaktadır (Harrington, 2012).

Denetimli öğrenmenin aksine, denetimsiz öğrenme algoritmaları etiketli eğitim verilerine ihtiyaç duymaz. Bu algoritmaların temel amacı verilen verilerdeki benzerlikleri aramak ve verileri bu benzerliklere göre kategorize etmektir (Nasteski, 2017). Denetimsiz sınıflandırmaya kümeleme de denir ve kategorize edilmiş gruplara kümeler adı verilir. Bu kümeler, eğer sınıf bilgisine ihtiyaç duyulursa, son işlemlerde deneyimli bir kullanıcı tarafından etiketlenebilir. Bu çalışmanın kapsamı, her piksel için ürün etiketlerini içeren ürün haritaları elde etmek olduğundan, sınıflandırma için yer gerçeği verileri kullanılarak denetimli öğrenme gerçekleştirilmiştir. Son yıllarda Rastgele Orman (Breiman, 2001), Destek Vektör Makineleri (DVM) (Vapnik et al, 1995), yapay sinir ağları ve k-en yakın komşu gibi çeşitli denetimli sınıflandırma yöntemleri geliştirilmiştir. Bu çalışmada sınıflandırma ve özellik seçimi için rastgele orman algoritması kullanılmıştır. Diğer sınıflandırıcılarla algoritma eğitiminin hesaplama süreleri. Diğer sınıflandırıcılar olan Yapay sinir ağları, DVM ve Xgboost ile algoritma eğitiminin hesaplama süreleri test edildi ve her birinin Rastgele orman'dan daha fazla hesaplama süresi aldığı, bunun da diğer yöntemlerle hiper parametre ayarlama sürecini hesaplama açısından çok maliyetli hale getireceği gözlemlenmiştir. Bu nedenle Rastgele orman, test edilen tüm sınıflandırıcılar arasında daha az hesaplama maliyeti ile yüksek doğruluk sağladığı için ürün sınıflandırmasında tercih edilmektedir.

2.2.1 Makine Öğrenmesi Türleri

Makine öğrenmesi yaklaşımlarına birkaç örnek, denetimli öğrenme, denetimsiz öğrenme, pekiştirmeli öğrenme ve derin öğrenmedir. Veri toplama işlemi bu çeşitli yaklaşımlar kullanılarak sınıflandırılır.

2.2.2 Denetimli Öğrenme

Denetimli öğrenme tarafından sağlanan eğitim örnekleri, belirli hedeflere sahiptir ve bu örnekleri kullanan algoritmalar, olası tüm girdiler için doğru sonuçlar sağlar. "Denetimli öğrenme" ve "örneklerden öğrenme" terimleri sıklıkla birbirinin yerine kullanılır. Sınıflandırma ve regresyon, denetimli öğrenme metodolojisinin uygulanabileceği iki görev örneğidir. Örneğin, sınıflandırma kullanarak "Bu kurabiye kalite standartlarına uygun mu?" sorusuna evet veya hayır cevabı verilebilir. Bu, sınıflandırmanın nasıl kullanılabileceğine bir örnek olacaktır. Regresyon ise "kaç tane" ve "ne kadar" sorularına yanıt sağlar (M.Fatima, 2017).

Denetimli öğrenme sürecinde, bir modelin istenen çıktıyı üretmesi, bir eğitim setinin yardımıyla öğretilebilir. Bu eğitim veri seti ile modelin genel performansının zamanla iyileşmesi mümkündür. Algoritma, hata kabul edilebilir bir seviyeye indirilene kadar ayarlamalar yapmaya devam eder. Bu eşik, algoritma tarafından belirlenir.

Veri madenciliği alanında denetimli öğrenme gerektiren problemler, sınıflandırma ve regresyon olmak üzere iki türe ayrılabilir.

Bir sınıflandırma sistemi kullanarak, test verilerinin kesin sınıflandırılması mümkündür. Veri kümesini tanınan öğeler açısından inceler ve bulduklarına dayanarak bu öğelerin nasıl sınıflandırılması veya tanımlanması gerektiği konusunda akıllı tahminler üretir. K-en yakın komşu analizi, rastgele orman, doğrusal sınıflandırıcılar, DVM ve karar ağaçları gibi yöntemler, yaygın sınıflandırma yaklaşımlarına örnek olarak verilebilir.

Denetimli makine öğrenmesi süreci, hedeflerine ulaşmak için çeşitli bilgisayar yaklaşımlarından ve algoritmik çerçevelerden yararlanır. Bu yaklaşımları kullanan hesaplamalar genellikle R veya Python gibi bilgisayar programları ile gerçekleştirilir. Bu makale, bilgi edinmenin en yaygın yollarından bazılarına kısa bir giriş sunmaktadır.

Denetimli makine öğrenimine yönelik rastgele orman yaklaşımı, sınıflandırma ve regresyon modellemesi için kullanılabilecek başka bir seçenektir. "Orman", her biri bağımsız olarak çalışan bir dizi bireysel karar ağacından oluşur. Bunlar, veriye dayalı daha doğru tahminler oluşturmak için birleştirilir ve bu "orman" olarak adlandırılır.

2.2.3 Rastgele Orman

Leo Breiman ve Adele Cutler tarafından geliştirilen Rastgele Orman yöntemi, birkaç karar ağacından elde edilen sonuçların sentezine dayanarak tek bir tahmin oluşturan iyi bilinen bir makine öğrenme stratejisidir. Sınıflandırma ve regresyon gibi çeşitli problemleri çözme yeteneği, popülerliğinin artmasına katkıda bulunmuştur (L. Breiman, 2001).

Rastgele orman modelini daha iyi anlamak için, öncelikle karar ağacı tekniğinin kısa bir açıklamasını sağlamak gereklidir. Bunun nedeni, her bir ağacın modelde oluşturulduğu temel bileşenin karar ağacı yönteminden kaynaklanmasıdır. Karar ağacında yapılacak ilk şey, kendinize basit bir soru sormaktır, örneğin "Bugün dışarıda yürüyüş yapmalı mıyım?" Ardından, seçenekleri daraltmak için "Hava güneşli mi?" veya "Yağmur yağıyor mu?" gibi takip soruları sormak faydalıdır. Bu tür sorular, mevcut bilgileri bölme amacı taşıyan karar ağacının dallarını oluşturur. Her bir soru dizisi, yaprak düğümünün yerini belirlemeye katkıda bulunur, bu düğüm kişinin nihai kararını temsil eder. Bu şartları karşılayan gözlemler "Evet" yolunu takip ederken, karşılamayanlar alternatif yolu seçecektir. Karar ağaçlarının görsel örneği Şekil 2.1'de verilmektedir. Verileri alt kümelere ayırmak için en iyi bölmeyi belirlemeye çalışan karar ağaçlarını eğitirken, Sınıflandırma ve Regresyon Ağacı yaklaşımı sıklıkla kullanılır. Ayrımı değerlendirmek için kullanılabilecek birkaç ölçü arasında Gini saflığı, bilgi kazancı ve ortalama kare hata bulunmaktadır (L. Breiman, J. Friedman, 1984).

Bu karar ağacının gösterdiği sınıflandırma sorunları, bu tür problemlerin uç bir örneği olarak düşünülebilir.

Karar ağaçları yaygın bir denetimli öğrenme yöntemi olmasına rağmen, önyargı ve aşırı uyum gibi paylaştıkları bazı eksiklikler de vardır. Karar ağaçları topluluğu kullanıldığında, rastgele orman tekniği daha doğru tahminler yapar ve bu, özellikle bireysel ağaçlar birbirinden bağımsız olduğunda geçerlidir (T. Kam Ho, 1995).

Topluluk öğrenme sürecinde, çeşitli sınıflandırıcılar (örneğin karar ağaçları) kullanılarak tahminler yapılır ve ardından bu tahminlerin sonuçları birleştirilerek en sık kabul edilen belirlenir. En yaygın topluluk yöntemleri, bagging ve boosting olarak bilinir. Bagging, 1996 yılında Leo Breiman tarafından yaratılmış bir tekniktir

(L. Breiman, 1996). Bu teknik, bir eğitim setinin bir kısmını rastgele seçmeyi ve ardından daha önce seçilen veri noktalarını yenileriyle değiştirmeyi içerir. Birden çok veri örneği toplandıktan sonra, bu modeller ayrı ayrı eğitilir ve eldeki göreve bağlı olarak (regresyon veya sınıflandırma), bu tahminlerin ortalaması veya çoğunluğu daha kesin bir tahmin sağlar. Bu strateji, kaotik bir veri setine bir düzen getirmeye çalışırken yaygın olarak kullanılır (R. E. Schapire, 2002).

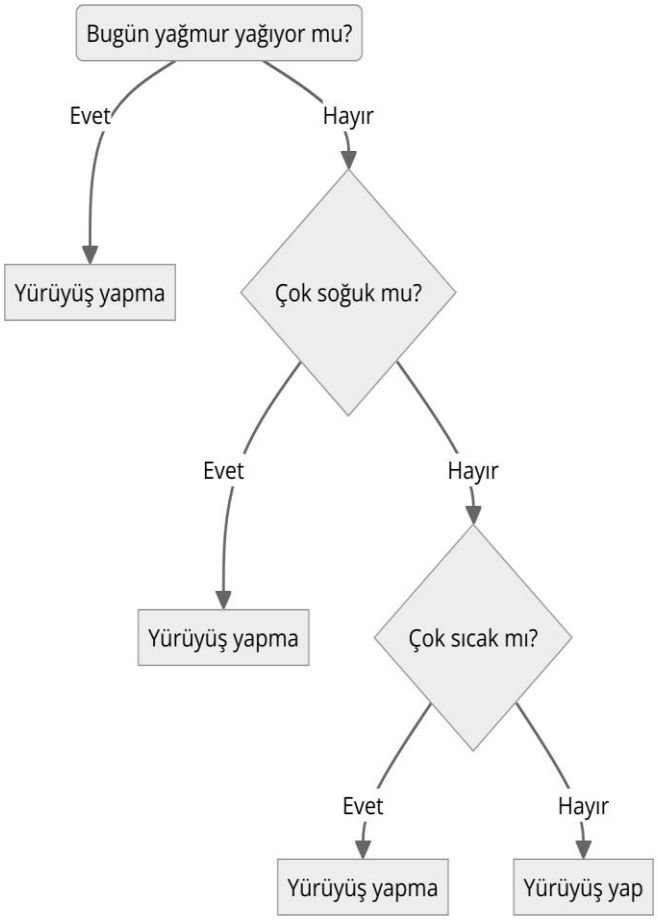
Rastgele orman yöntemi, karar ağaçlarını oluştururken kullanılan özelliklerin rastgeleleştirilmesi ile bagging yaklaşımının unsurlarını birleştirir ve bağımsız olan ve rastgele sırayla üretilen bir dizi karar ağacı oluşturur. Rastgele alt uzay tekniği, özellik toplama olarak da bilinir ve bu, karar ağaçlarının birbirleriyle çok az ortak noktasının olmasını sağlamaya yardımcı olur. Rastgele ormanlar ile karar ağaçları arasındaki fark büyük ölçüde bu özelliğe bağlanabilir. Rastgele Ormanlar, yalnızca özelliklerin bir alt kümesini seçerken, karar ağaçları tüm olası özellik bölmelerini dikkate alır (A. Liaw and M. Wiener, 2002). Rastgele Orman algoritmasının görsel örneği Şekil 2.2'de verilmektedir.

"Bugün dışarıda yürüyüş yapmalı mıyım?" gibi hipotetik bir durumda, tahmini belirlemek için sorduğum sorular, başkasının sorduğu sorular kadar geniş olmayabilir. Veri çeşitliliğinin tüm potansiyel kaynaklarını dikkate alarak, aşırı uyum, önyargı ve toplam varyans azaltılabilir, bu da daha doğru tahminlerle sonuçlanır (A. Cutler, 2012).

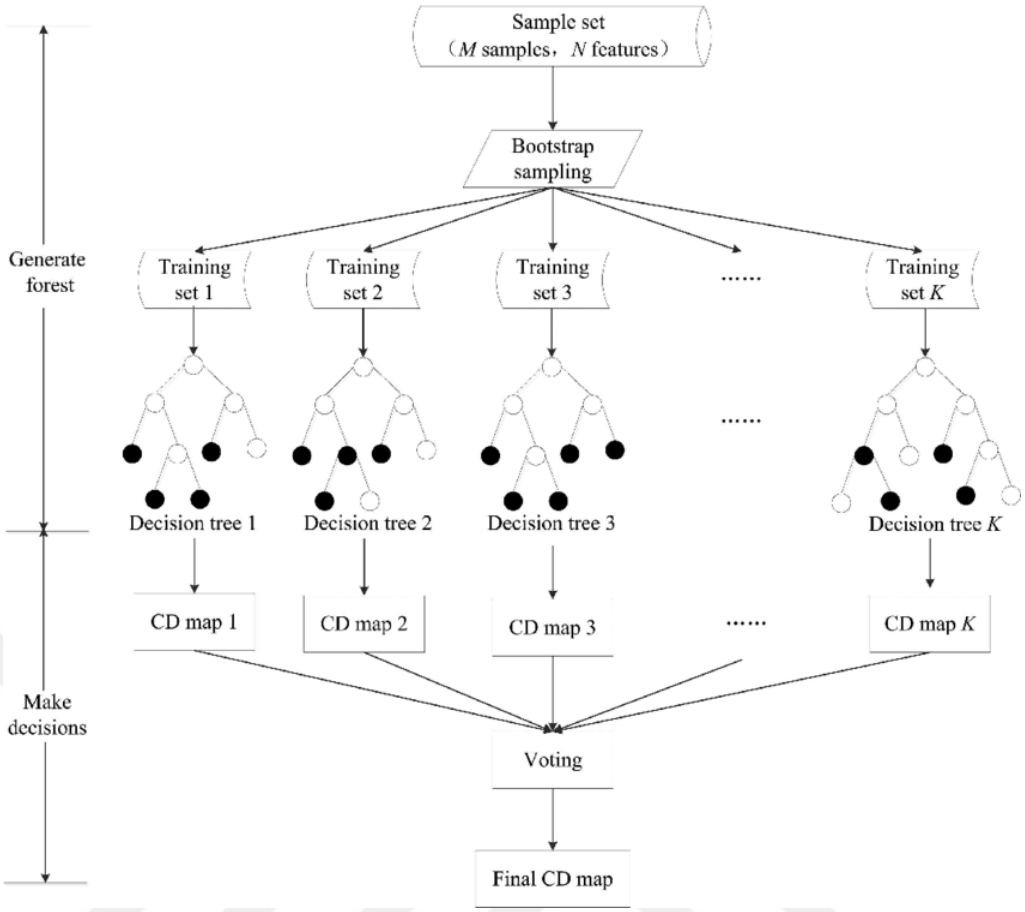
Eğitim sürecine başlamadan önce, rastgele orman yöntemlerinde kullanılan üç temel hiperparametreyi ayarlamanız gerekecektir. Temel kriterler arasında özellik örnekleme, düğüm boyutu ve ağaç derinliği bulunur. Bu adım tamamlandıktan sonra, rastgele orman sınıflandırıcısı, regresyon ve sınıflandırma ile ilgili sorunları çözmek için kullanılabilir (T. Hastie, R. Tibshirani, 2009).

Rastgele orman algoritması, her biri bootstrap örnekleme aracılığıyla oluşturulan karar ağaçlarının bir topluluğudur. Bir bootstrap örneği, eğitim verilerinin tamamından periyodik olarak rastgele seçilen ve yerine konan bir alt kümedir. Bir dakika içinde, eğitim örneğinin üçte biri olan out-of-bag örneğine geri döneceğiz. Özellik toplama kullanımı, veri setinin kapsamını genişletirken, aynı anda farklı karar ağaçları kümeleri arasındaki bağlantıyı azaltan başka bir rastgelelik türünü tanıtır. Mevcut durumun ayrıntıları, tahmin yönteminin belirlenmesinde

önemli bir rol oynar. Bir sınıflandırma görevinde, tahmin edilen sınıf, en sık görülen kategorik değişken olan çoğunluk oyu ile belirlenir. Öte yandan, bir regresyon görevinde, bireysel karar ağaçları ortalaması alınır. Son aşama, örneklemin kullanımını içeren çapraz doğrulamadır (Frontiers, 2017).



Şekil 2.1: Rastgele Orman Modeli



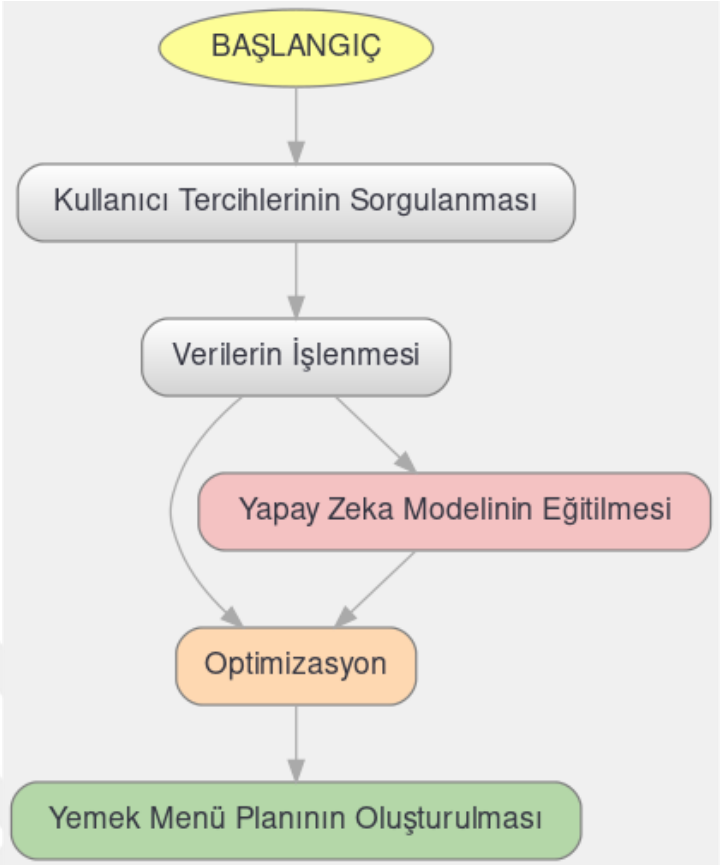
Şekil 2.2: Rastgele Orman Sınıflandırıcısının Şeması

Kaynak: (Feng, W., Sui, 2018).

3. UYGULAMA

Modern optimizasyon pratiklerinde, özellikle büyük veri setlerini içeren ve çoklu kısıtlamalar barındıran problemlerin çözümü, manuel yöntemlerle sürdürülebilir olmaktan uzaktır. Bu, literatürde de sıklıkla üzerinde durulan bir noktadır. Zira, yapılan çalışmalar, problemlerin karmaşıklığının artması ile birlikte, geleneksel hesaplama yöntemlerinin yetersiz kaldığını ve hata paylarının arttığını vurgulamaktadır (Michael, Aggrey, 2022). Bu bağlamda, bilgisayarla desteklenen optimizasyon teknikleri, sadece zaman ve maliyet açısından değil, aynı zamanda çözüm kalitesi ve uygulanabilirlik açısından da önemli avantajlar sağlamaktadır.

Modern yazılım teknolojileri ve programlama dillerinin gelişimi, Python, R, Matlab tabanlı kütüphaneler aracılığıyla optimizasyon problemlerinin daha hızlı ve doğru bir şekilde çözülmesini mümkün kılmıştır. Uygulama aşamasında de benzer bir yaklaşım benimsenmiş, Menü Planının oluşturulması sürecinde, Python programlama dili ve PuLP gibi gelişmiş optimizasyon kütüphaneleri etkin bir şekilde kullanılmıştır. Bu sayede, kullanıcı tercihleri gibi çeşitli parametreler dikkate alınarak, dinamik ve esnek çözümler üretilebilmiştir. Yapay zeka modelleri, özellikle Rastgele Orman algoritması, bu veri setlerinden öğrenerek ve tahmin yürüterek, kullanıcıların zevklerine en uygun yemek önerilerini belirlemekte kullanılmıştır. Uygulanan bu teknikler, veri büyüklüğü ve kısıtların artması gibi zorluklara rağmen, menü planlama süreçlerini optimize ederken, aynı zamanda müşteri memnuniyetini artırma ve operasyonel verimliliği maksimize etme çabalarına önemli katkılarda bulunmuştur. Elde edilen bulgular, projenin, aynı zamanda pratikte de uygulanabilir ve etkili çözümler sunduğunu göstermektedir. Uygulamanın akışı görsel örneği Şekil 3.1'de verilmektedir.



Şekil 3.1: Akış Diyagramı

3.1 Kullanılan Araçlar

Bu bölümde, uygulama aşamasında kullanılan araçlar ve teknolojiler detaylandırılacaktır. Projenin ana bileşenleri; bir web sitesi, REST API ve optimizasyon ile yapay zeka algoritmalarını içeren arka uç kodlamalarıdır. Her bir bileşen, projenin genel amacı doğrultusunda kullanıcılar için kişiselleştirilmiş yemek menüleri sunmak için bir araya getirilmiştir.

3.1.1 Web sitesi

Web sitesi, kullanıcıların yemek tercihlerini girebilecekleri ve sonrasında yetkili kişinin bu verileri yemek menüsü optimizasyonunda kullanmak üzere bir arayüz sağlamaktır. Bu arayüz, Flutter kullanılarak geliştirilmiştir. Flutter, hızlı ve etkili bir şekilde yüksek performanslı mobil ve web uygulamaları oluşturmayı sağlayan bir açık kaynaklı Kullanıcı Arayüzü yazılım geliştirme kitidir. Web sitesi şu özellikleri içermektedir:

- **Anket Modülü:** Web sitesi, çalışanlara yöneltilen spesifik anket sorularını içerir. Bu sorular, çalışanların yemek tercihlerini ve beslenme alışkanlıklarını belirlemek için tasarlanmıştır. Çalışanlar, web sitesi üzerinden bu soruları yanıtlarlar, ve bu veriler doğrudan veritabanına kaydedilir.
- **Yapay Zeka Entegrasyonu:** Toplanan kullanıcı verileri, aynı zamanda bir yapay zeka modelini eğitmek için kullanılır. Bu model, Rastgele Orman algoritması kullanılarak geliştirilmiştir ve model, gelecekteki kullanıcı tercihlerini tahmin etmek amacıyla kullanılır. Modelin tahminleri, menü planlama sürecinde rehber olarak kullanılabilir.
- **Optimizasyon Entegrasyonu**: Catering firmasındaki yetkili kişiler, toplanan verileri kullanarak özelleştirilmiş yemek menüleri oluşturabilir. Bu süreç, Python PuLP kütüphanesi kullanılarak geliştirilen bir optimizasyon modeli tarafından desteklenmektedir. Model, kullanıcı tercihlerini ve beslenme gereksinimlerini dikkate alarak optimal yemek seçimlerini belirler.

Bu web sitesi modülleri, kullanıcı deneyimini ve operasyonel verimliliği artırmak üzere entegre edilmiştir. Kullanıcı verilerinin toplanması, işlenmesi ve analiz edilmesi süreçleri, site üzerinden otomatik olarak yönetilir, bu da sürecin hızlı ve hatasız ilerlemesini sağlar.

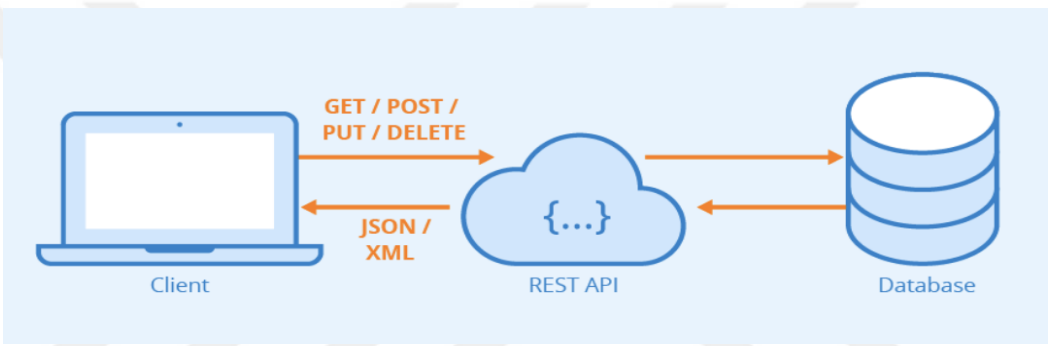
3.1.2 REST API

REST API web tabanlı sistemlerde farklı uygulamalar arasında bilgi alışverişi yapmak için kullanılan bir araçtır. REST mimarisi, kaynakların bir ağ üzerinden standart HTTP metodları ile yönetilmesini sağlar.

Projede geliştirilen REST API, kullanıcı arayüzü ve sunucu tarafı uygulamalar arasındaki veri akışını yönetmek için kullanılmaktadır. Bu API, kullanıcıdan alınan verileri toplar, işler ve sonuçları kullanıcıya sunar. Uygulama Programlama Arayüzünün görsel örneği Şekil 3.2'de verilmektedir. Ayrıca, kullanıcı tercihlerine dayalı olarak yemek menüsü önerileri üretmek için optimizasyon ve yapay zeka modellerini tetikler. Bu kapsamda, Python ve FastAPI kullanılarak bir REST API geliştirilmiştir. FastAPI, modern, hızlı (yüksek performanslı) ve Python 3.6+ sürümleri ile uyumlu bir web framework'üdür.

Temel Özellikler:

- **Veri Toplama:** Kullanıcılar web sitesi üzerinden anket yanıtları gibi veriler girer. REST API, bu verileri alır ve SQL tabanlı veritabanına kaydeder.
- **Menü Oluşturma ve Öneri:** Kullanıcıdan alınan tercihler, önceden tanımlanmış kriterlere göre değerlendirilir ve uygun yemek önerileri hazırlanır. Bu süreç, hem optimizasyon algoritmaları hem de yapay zeka modelleri tarafından desteklenir.
- **Veri Güncellemeleri ve Yönetimi:** REST API, veri güncellemelerini yönetir ve veritabanındaki değişiklikleri izler. Bu sayede, sistemdeki veriler her zaman güncel ve tutarlı kalır.



Şekil 3.2: Uygulama Programlama Arayüzü şeması

Veritabanı Yönetimi

Proje için veri saklama ve yönetim işlemleri PostgreSQL kullanılarak gerçekleştirilmektedir. PostgreSQL, açık kaynaklı ve yüksek performanslı bir ilişkisel veritabanı yönetim sistemidir. Bu sistem, büyük miktardaki verilerin etkili bir şekilde saklanması, sorgulanması ve yönetilmesi için güçlü araçlar sunar.

REST API ve veritabanı yönetimi, projenin teknik altyapısının temel taşlarını oluşturur ve sistemin etkin bir şekilde çalışmasını sağlar. Bu yapı, kullanıcı deneyimini iyileştirirken, veri yönetiminde de maksimum verimlilik ve güvenlik sunar.

Optimizasyon ve yapay zeka uygulamaları için kullanılan araçlar ve kütüphaneler bu çalışmada önemli bir yer tutuyor:

3.1.3 Python

Python, yüksek seviyeli, yorumlanabilir ve nesne yönelimli bir programlama dilidir. 1991 yılında Guido van Rossum tarafından geliştirilen Python, kolay okunabilirliği ve anlaşılır sözdizimi ile bilinir. Bu özellikler, programcıların daha az kod satırı ile daha okunabilir ve düzenli kodlar yazmalarını sağlar. Python, özellikle veri bilimi, makine öğrenimi, web geliştirme, otomasyon, veri analizi ve bilimsel hesaplamalar gibi alanlarda yaygın olarak kullanılmaktadır.

Python, açık kaynaklı bir dildir ve dünya çapında büyük bir topluluk tarafından desteklenmektedir. Bu topluluk, Python'a geniş bir kütüphane ve modül ekosistemi kazandırmıştır. Projede kullanılan Pandas, PuLP, Scikit-learn ve Joblib gibi kütüphaneler, bu dili veri işleme ve analitik uygulamalar için son derece güçlü kılar.

3.1.4 PuLP

PuLP, Python programlama dili için geliştirilmiş, açık kaynaklı bir lineer programlama kütüphanesidir. PuLP, optimizasyon problemlerini modelleme ve çözmek için kullanılır. Özellikle, tamsayılı ve sürekli karar değişkenlerini içeren karmaşık problemleri çözebilir. PuLP ile model kurulumu, kısıtların eklenmesi, hedef fonksiyonun tanımlanması ve çözüm algoritması seçimi gibi adımlar kolaylıkla gerçekleştirilebilir. PuLP, CBC (Coin-or branch and cut) gibi çözücülerle entegre çalışarak geniş çaplı problemlerin efektif bir şekilde çözülmesini sağlar.

3.1.5 Pandas

Pandas, veri analizi ve manipülasyonu için kullanılan, Python dilinde yazılmış açık kaynaklı bir kütüphanedir. Veri setlerini okuma, temizleme, transformasyon yapma, eksik verileri işleme, veri agregasyonu gibi birçok işlemi hızlı ve etkin bir şekilde gerçekleştirilebilir. Pandas, projede veri toplama ve ön işleme aşamalarında kullanıcı verilerini düzenlemek ve analiz etmek için esas alınmıştır. DataFrame ve Series gibi veri yapıları ile çalışır ve bu veri yapıları üzerinde kompleks işlemleri basit ve anlaşılır metotlarla sunar.

3.1.6 Scikit-learn

Scikit-learn, makine öğrenimi algoritmalarını içeren, Python programlama dili için geliştirilmiş açık kaynaklı bir kütüphanedir. Sınıflandırma, regresyon, kümeleme ve boyut indirgeme gibi çeşitli makine öğrenimi tekniklerini barındırır. Bu Çalışmada, Rastgele Orman algoritması bu kütüphane aracılığıyla uygulanmıştır. Scikit-learn, veri ön işleme, model eğitimi, model değerlendirme ve model iyileştirmeleri yapma konularında kapsamlı araçlar sunar.

3.1.7 Joblib

Joblib, Python'da kullanılan, özellikle büyük veri setleri üzerinde çalışırken verimliliği artırmak amacıyla geliştirilmiş bir kütüphanedir. Nesne serileştirme işlemleri, veri önbelleğe alma ve paralel hesaplama işlemleri için tasarlanmıştır. Projede, eğitilen yapay zeka modellerinin diske kaydedilmesi ve tekrar yüklenmesi işlemleri joblib kullanılarak gerçekleştirilmiştir. Bu sayede, model eğitim süreçleri arasında verimlilik sağlanmış ve modeller kolaylıkla saklanıp tekrar kullanılabilmektedir.

3.2 Veri seti ve Besin değerleri

Bu projede kullanılan besin veri seti, Beslenme Bilgi Sistemi (BeBİS) programı aracılığıyla elde edilmiştir. BeBİS, 20.000'den fazla gıda maddesinin veri tabanını ve 130'dan fazla besin ögesinin detaylı analizlerini içeren profesyonel bir bilgisayar programıdır. Türkiye Ulusal Gıda Kompozisyon Veri Tabanı kullanılarak, besinlerin girilen porsiyon miktarına göre kalori, protein, karbonhidrat, yağ, vitaminler, mineraller, amino asitler ve glisemik indeks gibi çeşitli besin değerleri hesaplanmaktadır. Bu bilgiler, yemek tarifleri, diyet planları ve menülerin besin değerlerini belirlemek için kullanılmaktadır (Şahin, S, 2022).

Projede, optimizasyon sürecinde kullanılan yemek menüleri için bu veri seti temel alınmıştır. Yemeklerin besin içerikleri, belirlenen sağlık ve beslenme standartlarına uygun olarak dengeli ve sağlıklı menüler oluşturmak üzere analiz edilmiştir. Bu analizler, gıda mühendisleri ve beslenme uzmanları tarafından yapılmıştır. Elde edilen veriler, kullanıcıların beslenme alışkanlıklarına ve tercihlerine göre özelleştirilmiş menülerin tasarlanması için esas teşkil etmiştir. Kullanılan besin değerleri Ek-1 bölümünde tablo halinde sunulmuştur.

3.3 Veri Toplama ve Ön İşleme

Projede veri toplama süreci, catering firmaları ile anlaşmalı olan şirketlerin çalışanlarına yönelik bir dizi anket uygulaması ile gerçekleştirilmiştir. Bu ankette, yaş, cinsiyet, medeni durum, aktivite durumu gibi demografik bilgilerin yanı sıra, yemek tercihlerine ilişkin detaylı sorular yer almaktadır. Toplamda yaklaşık 20 soru bulunan ankette, yemekle ilgili sorulara "Hiç", "Az", "Orta", "Sık", "Çok Sık" şeklinde beş farklı cevap seçeneği sunulmuştur.

Gerçek bir catering firması ile çalışılmadığı için, projede kullanılan veriler deneysel amaçlarla rastgele üretilmiştir. Bu süreç, Python programlama dili ve çeşitli kütüphaneleri kullanılarak gerçekleştirilmiştir. Özellikle, ***random*** kütüphanesi aracılığıyla, ankete verilecek cevapların olasılıkları ayarlanarak farklı senaryolar altında veri setleri oluşturulabilecek şeklinde yapılmıştır. Bu yöntem, projenin esnekliğini ve geniş kapsamlı testler yapabilme kapasitesini artırmaktadır.

Anket sonuçları, veri işleme ve analizi için sayısal formata dönüştürülmektedir. Bu dönüşüm, verilerin makine öğrenimi algoritmaları ve optimizasyon modelleri tarafından işlenebilir hale getirilmesini sağlar. Sayısal dönüşüm işlemi *One-Hot Encoding* fonksiyonu ile yapılmakta, bu fonksiyon her bir anket cevabını ilgili sayısal değerlere çevirmektedir. Son olarak, elde edilen sayısal veriler, veritabanına kaydedilmekte ve bu veriler üzerinden çeşitli analizler yapılabilmesi için API üzerinden erişilebilir hale getirilmektedir. Kullanıcı verilerilerinin oluşturulmasının kod örneği Şekil 3.3'de verilmektedir. Tüm sorular ve gerekli kodlar Ek-2 bölümünde sunulmuştur.

```
# Kullanıcı verisini rastgele oluşturacak fonksiyon
def generate_user_responses(questions, distributions_dict):
    user_responses = []
    for question in questions:
        dist = distributions_dict.get(question['id'])
        if question['type'] == 'range':
            age_range = question['ansers']
            answer = random.randint(int(age_range[0]), int(age_range[1]))
        else:
            if dist:
                answer = random.choices(question['ansers'], weights=dist, k=1)[0]
            else:
                answer = random.choice(question['ansers'])
        numeric_answer = convert_answers_to_numeric(question, answer)
        user_responses.append(numeric_answer)
    return user_responses
```

Şekil 3.3: Kullanıcı Verilerinin Oluşturulması

3.4 Optimizasyon modeli

Projede kullanılan optimizasyon modeli, menü planlaması için çeşitli besin değerleri ve kullanıcı tercihlerini dikkate alan bir karar verme modelidir. Model, belirli diyet kısıtlamalarını ve besin öğesi gereksinimlerini karşılayacak şekilde menüleri optimize etmek amacıyla kurulmuştur. Model, PuLP kütüphanesi kullanılarak Python ortamında kurulmuştur. Modelin çözümünde PuLP tarafından kullanılan PULP_CBC_CMD çözücüsü tercih edilmiştir. Bu çözücü, KTDP problemleri için Branch-and-Cut yöntemini kullanır.

3.4.1 Branch-and-Cut

Branch-and-Cut algoritması, KTDP problemlerinin çözümünde kullanılan gelişmiş bir optimizasyon yöntemidir. Bu algoritma, DP ve dallanma ve sınırlandırma (Branch-and-Bound) tekniklerini birleştirerek çalışır. Algoritmanın ilk adımı, tüm tamsayı kısıtlamalarının göz ardı edildiği ve yalnızca doğrusal kısıtlamaların dikkate alındığı bir lineer relaxation probleminin çözülmesidir. Elde edilen bu çözüm, KTDP probleminin alt sınırını belirler ancak genellikle kesirli değerler içerir. Bu aşamada, kesirli çözümleri ortadan kaldırmak ve uygun bölgeyi daraltarak sadece tam sayı çözümlerine izin vermek için kesme düzlemleri (Cutting Planes) eklenir. Gomory kesme düzlemleri ve kapasiteli kesme düzlemleri gibi yöntemler, çözüm uzayını daraltmada yaygın olarak kullanılır (Maria-Florina, 2022). Eğer lineer relaxation probleminin çözümü tamsayı değerler içermiyorsa, çözüm uzayı dallanma işlemi ile alt problemlere bölünür ve her alt problem ayrı ayrı çözülür. Bu işlem, optimal çözümü bulana kadar tekrarlanır ve her iterasyonda tamsayı kısıtlamalarına uygun yeni kesme düzlemleri eklenir (Amitabh, Basu, 2022). Branch-and-Cut algoritmasının bu yapısı, onu karmaşık optimizasyon problemlerinin çözümünde son derece etkili kılar.

3.4.2 Kısıtlar

1. Dört Çeşit Yemek:

Her menüde dört farklı yemek çeşidi bulunmalıdır. Bu yemekler, dört kategori halinde sınıflandırılmıştır. Kategorilerin detaylarına ve yemeklerin tam listesine, Ek-2'de verilmiş olan GitHub adresinde bulunan "detailed_food.json"

dosyasından ulaşılabilir. Her menüde, her kategoriden yalnızca bir yemek bulunacak şekilde ayarlama yapılmalıdır.

2. Yemeklerin Belirli Bir Sıra ile Verilmesi:

Yemeklerin belirli bir sırayla verilmesi gerekmektedir.

3. Yemeklerin Benzersiz Olması:

Her menüdeki yemeklerin benzersiz olması gerekmektedir. Aynı yemek birden fazla kez kullanılmamalıdır. Bu kısıt, menülerin çeşitliliğini ve tekrar eden yemeklerden kaçınılmasını sağlamaktadır.

4. Aynı Renk ve Kıvama Sahip Yemeklerin Kullanımı:

Aynı renk ve kıvama sahip yemeklerin her menüde ikiden fazla kullanılmaması gerekmektedir.

5. Sebze Yemeklerinin Yanına Salata Verilmemesi:

Sebze yemeklerinin yanına salata ibaresi içeren yiyecekler verilmemelidir. Bu kısıt, besin çeşitliliğini artırmak ve menülerdeki sebze ağırlığını dengelemek amacıyla uygulanmaktadır.

6. Çorbaların Yanına Komposto ve Hoşaf Verilmemesi:

Çorbaların yanında komposto veya hoşaf gibi içeceklerin verilmemesi gerekmektedir.

7. Etli Dolma ve Etli Sarma Yanına Pilav Verilmemesi:

"Etli dolma" ve "etli sarma" kelimelerini içeren yemeklerin yanına ikinci kap olarak pilav çeşitlerinin verilmemesi gerekmektedir.

8. Pirinç Pilavı, Yayla Çorbası ve Sütlaç Aynı Güne Verilmemelidir:

Pirinç pilavı, yayla çorbası ve sütlaç gibi yemeklerin aynı günde verilmemesi gerekmektedir.

9. Besin Değerlerinin Sınırlandırılması:

Menüdeki yemeklerin besin değerlerinin, belirlenen minimum ve maksimum değerler arasında olması gerekmektedir. Bu değerler şunlardır:

- Karbonhidrat: 41.6 gram ile 62.4 gram arasında

- Protein: 19.31 gram ile 28.97 gram arasında

- Yağ: 16.0 gram ile 28.0 gram arasında

- Lif: 8.0 gram ile 12.0 gram arasında

- Enerji: 528.0 kalori ile 980.57 kalori arasında

Bu sınırlar, T.C. Sağlık Bakanlığı Halk Sağlığı Genel Müdürlüğü tarafından yayımlanan "Toplu Beslenme Sistemleri İçin Ulusal Menü Planlama ve Uygulama Rehberi, (2020)" kaynak alınarak belirlenmiştir.

10. Kullanıcı Tercihleri:

Kullanıcıların demografik bilgileri ve yemek tercihleri yapay zeka modelinin oluşturulması, eğitilmesinde ve kişiselleştirilmiş menülerin optimize edilerek oluşturulmasında kullanılmak. Kullanıcı tercihleri, yaş, cinsiyet, medeni durum, aktivite durumu gibi demografik bilgilerin yanı sıra, yemek tercihlerine ilişkin detaylı soruları içeren bir anket ile toplanmaktadır. Toplamda yaklaşık 20 soru bulunan ankette, yemekle ilgili sorulara "Hiç", "Az", "Orta", "Sık", "Çok Sık" şeklinde beş farklı cevap seçeneği sunulmuştur. Bu anket sonuçları, kullanıcıların besin tercihlerini ve ihtiyaçlarını belirlemek için kullanılmakta ve menülerin optimize edilmesinde önemli bir veri kaynağı olarak kullanılmaktadır.

Bu kısıtlar, diyet menüsü optimizasyonunda dikkate alınarak, kullanıcıların beslenme ihtiyaçlarını karşılayan ve çeşitli menüler oluşturmada kullanılmıştır.

3.4.3 Yapay Zeka Modeli Eğitimi

Bu bölümde, kullanıcı tercihlerini temel alarak diyet menüsü oluşturmak amacıyla kullanılan yapay zeka modelinin eğitimi açıklanmaktadır. Model, kullanıcının demografik bilgileri ve yemek tercihleri gibi verileri kullanarak kişiselleştirilmiş menüler oluşturmak için eğitilmiştir.

Modelin eğitimi için kullanılan veriler, kullanıcıların yaş, cinsiyet, aktivite durumu, medeni durumu gibi demografik bilgileri ve yemek tercihlerini içermektedir. Bu verilerin toplanması ve ön işlem süreçleri Şekil 3.4'de verilmektedir.

Çekilen veriler, bağımsız değişkenler (X) ve bağımlı değişkenler (Y) olarak ayrılmıştır. Bağımsız değişkenler, kullanıcının yaş, cinsiyet, aktivite durumu ve

medeni durumu gibi bilgilerini içerirken, bağımlı değişkenler yemek tercihlerini temsil etmektedir. Veri setinin bu bölümlere ayrılması işlemi Şekil 3.4'te gösterilmektedir.

Veri seti, modelin eğitim ve test süreçleri için ikiye bölünmüştür. Veri setinin %80'i eğitim için, %20'si ise test için ayrılmıştır. Eğitim ve test veri setlerinin ayrılma süreci Şekil 3.4'de gösterilmektedir.

Diyet menüsü oluşturmak için Rastgele Orman sınıflandırıcı modeli kullanılmıştır. Bu model, karar ağaçları yöntemini kullanarak veri setindeki örüntüleri öğrenir ve tahminler yapar. Random Forest sınıflandırıcısı 100 ağaç ile oluşturulmuş ve eğitim verileri ile eğitilmiştir. Model eğitimi süreci Şekil 3.4'de detaylı bir şekilde verilmektedir.

Eğitilen model, daha sonra kullanılmak üzere dosya olarak kaydedilmiştir. Mevcut model dosyaları kontrol edilerek yeni model dosyası, mevcut en yüksek versiyon numarasına göre adlandırılmıştır. Model dosyalarının kaydedilme süreci Şekil 3.4'te gösterilmiştir.

Model eğitiminin tamamlanmasının ardından, model başarıyla kaydedilmiş ve kullanıma hazır hale getirilmiştir. Bu süreç, kullanıcılardan toplanan verilerin işlenmesi, modelin eğitilmesi ve modelin daha sonra kullanıma hazır hale getirilmesini kapsamaktadır. Bu model, kullanıcıların demografik bilgileri ve yemek tercihleri temel alınarak kişiselleştirilmiş menülerin oluşturulmasında kullanılacaktır.

Bu sürecin tamamı Şekil 3.4'te görsel olarak sunulmuştur.

```python
def train_model(db: Session = Depends(get_db)):

    query_meals = db.query(models.UserResponses).statement
    data = pd.read_sql_query(query_meals, db.bind)

    X = data[['age','gender','activity_status','marital_status']]
    Y = data.drop(['id', 'age','gender','activity_status','marital_status', 'company_id'],axis=1)

    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)

    model = RandomForestClassifier(n_estimators=100, random_state=42)
    model.fit(X_train, Y_train)

    Y_pred = model.predict(X_test)

    directory = '.'
    files = os.listdir(directory)
    model_files = [f for f in files if re.match(r'model_\d+\.joblib', f)]

    versions = [int(re.search(r'model_(\d+)\.joblib', f).group(1)) for f in model_files]
    latest_version = max(versions) + 1 if versions else 1

    # Yeni dosya adı
    new_filename = f"model_{latest_version}.joblib"

    # Modeli kaydet
    dump(model, os.path.join(directory, new_filename))

    print(f"Model saved as {new_filename}")

    return {"Message": "Model trained and saved successfully"}
```

Şekil 3.4: Yapay zeka modelinin eğitimi

3.4.4 Optimizasyon Entegrasyonu

Diyet menüsü oluşturma sürecinde optimizasyonun entegrasyonu, kullanıcı tercihleri ve beslenme gereksinimlerine dayalı olarak gerçekleştirilmiştir. Bu süreçte kullanılan tüm kodlar, Ek-2'de detaylandırılmıştır. Optimizasyonun temel amacı, besin değerleri ve kullanıcı tercihlerini dikkate alarak dengeli ve çeşitli menüler oluşturmaktır. Kullanıcının demografik bilgileri ve yemek tercihleri, veritabanından ve dosyalardan alınarak hazırlanmış, ardından PuLP kütüphanesi kullanılarak DP problemleri tanımlanmış ve çözülmüştür.

Öncelikle, veritabanından ve dosyalardan alınan yemek verileri işlenir ve kullanıcı tercihleri hesaplanır. Kullanıcının yaş ve cinsiyet bilgileri dikkate alınarak besin sınırları belirlenir. Ardından, PuLP kullanılarak her gün için bir optimizasyon problemi tanımlanır. Bu optimizasyon problemi, kullanıcı tercihlerini maksimize edecek şekilde belirlenir. Problemin amacı, kullanıcı tercih puanlarını maksimize etmektir. Her gün için belirlenen yemeklerin besin değerleri belirlenen sınırlar içinde olmalı ve belirli kısıtlamalara uymalıdır. Örneğin, her kategoriden yalnızca bir yemek seçilmeli, aynı renk ve kıvama sahip yemekler ikiden fazla olmamalıdır.

Ayrıca, sebze yemeklerinin yanına salata verilmemeli, çorbaların yanına komposto veya hoşaf konulmamalı, etli dolma ve sarma yanına pilav verilmemeli ve pirinç pilavı, yayla çorbası ve sütlaç aynı gün içinde verilmemelidir.

Her gün için bir PuLP problemi tanımlanır ve amaç fonksiyonu belirlenir. Amaç fonksiyonu, kullanıcı tercihleri doğrultusunda yemeklerin puanlarını maksimize etmeyi hedefler. Problemin Tanımlanması sürecinin kod örneği Şekil 3.5'de verilmektedir.

```
prob = pl.LpProblem(f"Menu_Optimization_{week}_{day}", pl.LpMaximize)
dish_vars = pl.LpVariable.dicts("Dish", available_dishes, 0, 1, pl.LpBinary)

# Amaç fonksiyonu: Kullanıcı tercihlerine göre yemeklerin toplam puanını maksimize etmek
prob += pl.lpSum([dish_costs[dish_id] * dish_vars[dish_id] for dish_id in dish_vars])
```

Şekil 3.5: Problemin Tanımlanması

Besin değerlerinin belirlenen sınırlar arasında olması sağlanır. Besin Değerleri Kısıtlarının Eklenmesi sürecinin kod örneği Şekil 3.6'de verilmektedir.

```
for nutrient in ["energy", "carbohydrate", "protein", "fat", "fiber"]:
    lower_limit = float(limits[nutrient].split("-")[0])
    upper_limit = float(limits[nutrient].split("-")[1])
    prob += pl.LpConstraint(
        e=pl.lpSum([df1_indexed.loc[i][nutrient] * dish_vars[i] for i in available_dishes]),
        sense=pl.LpConstraintGE,
        name=f"{nutrient}_lower_bound",
        rhs=lower_limit,
    )
    prob += pl.LpConstraint(
        e=pl.lpSum([df1_indexed.loc[i][nutrient] * dish_vars[i] for i in available_dishes]),
        sense=pl.LpConstraintLE,
        name=f"{nutrient}_upper_bound",
        rhs=upper_limit,
    )
```

Şekil 3.6: Besin Değerleri Kısıtlarının Eklenmesi

Yemeklerin kategorilere göre dengeli bir şekilde dağıtılması sağlanır:

```
for cat, rng in categories.items():
    prob += (
        pl.lpSum([dish_vars[i] for i in available_dishes if i in rng]) == 1
    )
```

Şekil 3.7: Sıralama Kısıtlarının Eklenmesi

Diğer kısıtlamalar da benzer şekilde eklenir:

```
for color in available_colors:
    prob += (pl.lpSum([dish_vars[i] for i in available_dishes if df1_indexed.loc[i]["color"] == color]) <= 2)
for texture in available_textures:
    prob += (pl.lpSum([dish_vars[i] for i in available_dishes if df1_indexed.loc[i]["consistency"] == texture]) <= 2)
for veg_meal_id in vegetable_meals_ids:
    for salad_meal_id in salad_meals_ids:
        if veg_meal_id in available_dishes and salad_meal_id in available_dishes:
            prob += (dish_vars[veg_meal_id] + dish_vars[salad_meal_id] <= 1)
```

Şekil 3.8: Diğer Kısıtlarının Eklenmesi

PuLP kullanılarak problem çözülür ve optimal menü belirlenir:

```
prob.solve(pl.PULP_CBC_CMD(msg=False))
for v in prob.variables():
    if v.varValue == 1:
        dish_id = int(v.name.split("_")[1])
        used_dishes.add(dish_id)
```

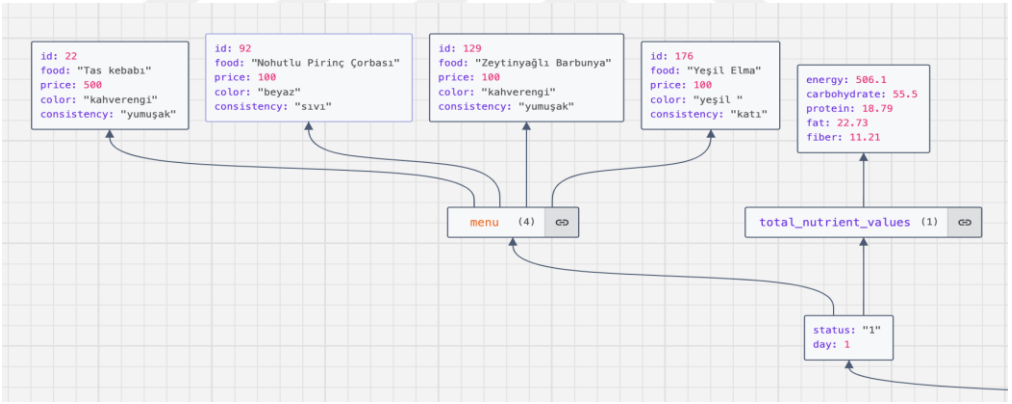
Şekil 3.9: Problemin Çözülmesi

Son olarak, her gün için oluşturulan menüler bir araya getirilerek haftalık menü oluşturulur.

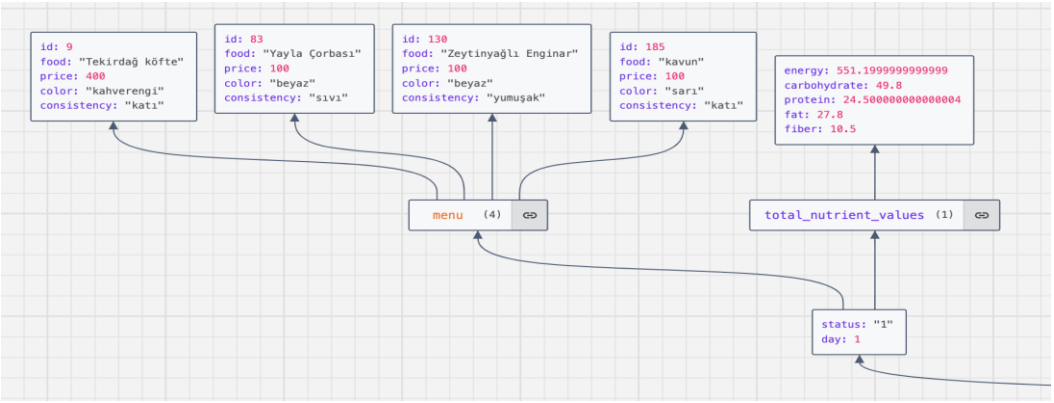
4. BULGULAR

Bu bölümde, çalışmamızda elde edilen bulgular detaylandırılmaktadır. Optimizasyon algoritmaları ve yapay zeka modelleri kullanarak kişiselleştirilmiş menü planlarının nasıl oluşturulduğu ve bu süreçlerin sonuçları sunulmaktadır. Çalışmamızda üç farklı test gerçekleştirilmiştir ve her bir testte hem yapay zeka destekli hem de yapay zeka destekli olmayan optimizasyon süreçleri değerlendirilmiştir.

1. İlk testte, 500 kullanıcı cevabı rastgele oluşturulmuş ve bu veriler üzerinde optimizasyon yapılmıştır. Optimizasyon sonuçları olumlu olup, tüm kısıtlar karşılanmıştır. İlk test için optimize edilmiş menüler ve skor metrikleri Şekil 4.1, Şekil 4.2 ve Şekil 4.3'te verilmektedir.



Şekil 4.1: Yapay Zeka Kullanmadan Kullanıcı Tercih Verileriyle Optimize Edilmiş Menü

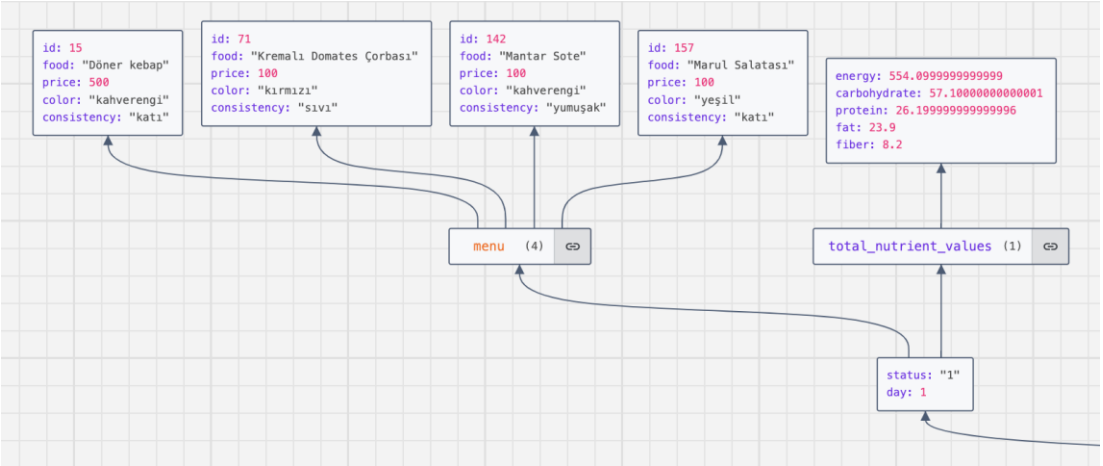


Şekil 4.2: Yapay Zeka Kullanarak Kullanıcı Tercih Verileriyle Optimize Edilmiş Menü

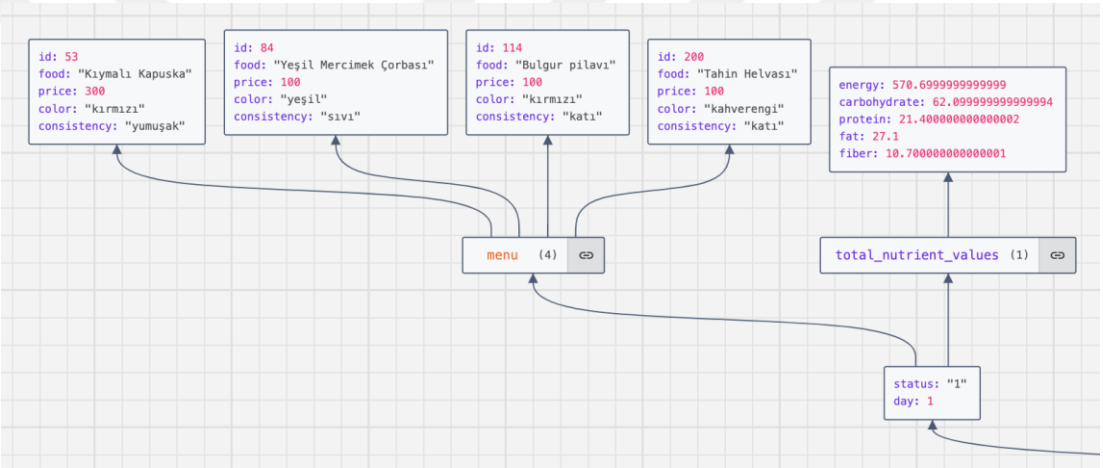
```
{
    "Accuracy": {
        "prefers_kebab_guvec": {
            "Accuracy": 0.22,
            "Report": {
                "0": {
                    "precision": 0.22727272727272727,
                    "recall": 0.25,
                    "f1-score": 0.23809523809523808,
                    "support": 20.0
                },
                "1": {
                    "precision": 0.2,
                    "recall": 0.16666666666666666,
                    "f1-score": 0.18181818181818182,
                    "support": 18.0
                },
                "2": {
                    "precision": 0.28,
                    "recall": 0.4117647058823529,
                    "f1-score": 0.3333333333333333,
                    "support": 17.0
                },
                "3": {
                    "precision": 0.2,
                    "recall": 0.10714285714285714,
                    "f1-score": 0.13953488372093023,
                    "support": 28.0
                },
                "4": {
                    "precision": 0.17391304347826086,
                    "recall": 0.23529411764705882,
                    "f1-score": 0.2,
                    "support": 17.0
                },
                "accuracy": 0.22,
                "macro avg": {
                    "precision": 0.2162371541501976,
                    "recall": 0.2341736694677871,
                    "f1-score": 0.2185563273935367,
                    "support": 100.0
                },
                "weighted avg": {
                    "precision": 0.21461976284584983,
                    "recall": 0.22,
                    "f1-score": 0.21008275445484748,
                    "support": 100.0
                }
            }
        }
    }
}
```


Şekil 4.3: Yapay Zeka Modelinin Performans Değerleri

2. İkinci testte, 5000 kullanıcı cevabı rastgele oluşturulmuş ve bu veriler üzerinde optimizasyon yapılmıştır. Optimizasyon sonuçları olumlu olup, tüm kısıtlar karşılanmıştır. İkinci test için optimize edilmiş menüler ve skor metrikleri Şekil 4.4, Şekil 4.5 ve Şekil 4.6'da verilmektedir.



Şekil 4.4: Yapay Zeka Kullanmadan Kullanıcı Tercih Verileriyle Optimize Edilmiş Menü

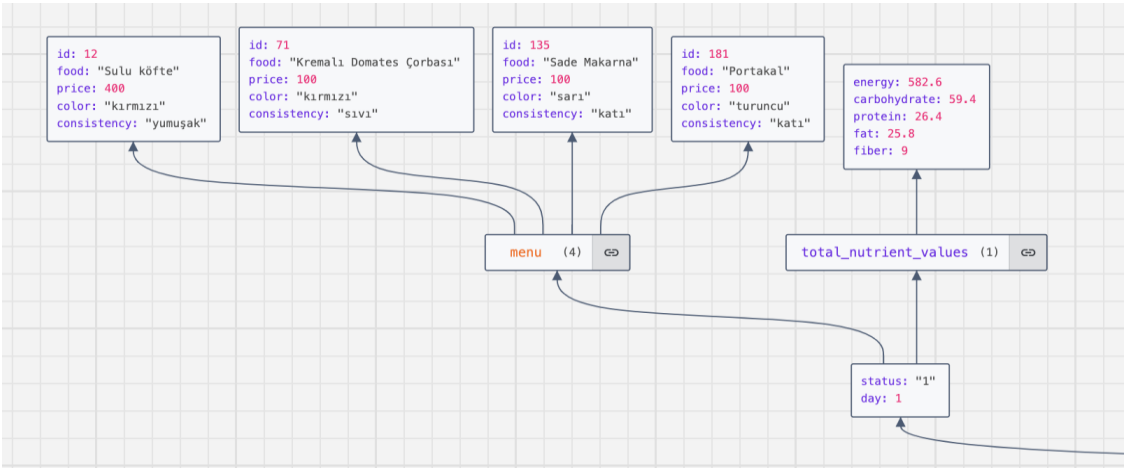


Şekil 4.5: Yapay Zeka Kullanarak Kullanıcı Tercih Verileriyle Optimize Edilmiş Menü

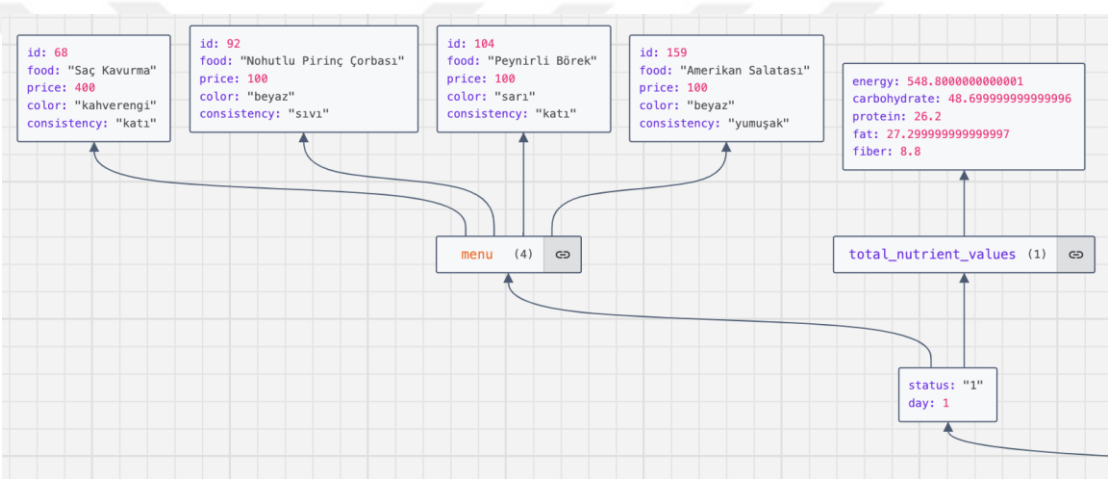
```
{
    "Accuracy": {
        "prefers_kebab_guvec": {
            "Accuracy": 0.20272727272727273,
            "Report": {
                "0": {
                    "precision": 0.17582417582417584,
                    "recall": 0.14479638009049775,
                    "f1-score": 0.1588089330024814,
                    "support": 221.0
                },
                "1": {
                    "precision": 0.20754716981132076,
                    "recall": 0.18803418803418803,
                    "f1-score": 0.19730941704035873,
                    "support": 234.0
                },
                "2": {
                    "precision": 0.16964285714285715,
                    "recall": 0.19,
                    "f1-score": 0.1792452830188679,
                    "support": 200.0
                },
                "3": {
                    "precision": 0.2328767123287671,
                    "recall": 0.2328767123287671,
                    "f1-score": 0.2328767123287671,
                    "support": 219.0
                },
                "4": {
                    "precision": 0.22053231939163498,
                    "recall": 0.25663716814159293,
                    "f1-score": 0.23721881390593047,
                    "support": 226.0
                },
                "accuracy": 0.20272727272727273,
                "macro avg": {
                    "precision": 0.20128464689975117,
                    "recall": 0.20246888971900917,
                    "f1-score": 0.2010918318592811,
                    "support": 1100.0
                },
                "weighted avg": {
                    "precision": 0.20199277836733895,
                    "recall": 0.20272727272727273,
                    "f1-score": 0.20157062393409655,
                    "support": 1100.0
                }
            }
        },
```


Şekil 4.6: Yapay Zeka Modelinin Performans Değerleri

3. Üçüncü testte, 15000 kullanıcı cevabı rastgele oluşturulmuş ve bu veriler üzerinde optimizasyon yapılmıştır. Optimizasyon sonuçları olumlu olup, tüm kısıtlar karşılanmıştır. Üçüncü test için optimize edilmiş menüler ve skor metrikleri Şekil 4.7, Şekil 4.8 ve Şekil 4.9'da verilmektedir.



Şekil 4.7: Yapay Zeka Kullanmadan Kullanıcı Tercih Verileriyle Optimize Edilmiş Menü



Şekil 4.8: Yapay Zeka Kullanarak Kullanıcı Tercih Verileriyle Optimize Edilmiş Menü

```
{
    "Accuracy": {
        "prefers_kebab_guvec": {
            "Accuracy": 0.19548387096774195,
            "Report": {
                "0": {
                    "precision": 0.21238938053097345,
                    "recall": 0.19801980198019803,
                    "f1-score": 0.2049530315969257,
                    "support": 606.0
                },
                "1": {
                    "precision": 0.19129082426127528,
                    "recall": 0.2,
                    "f1-score": 0.19554848966613672,
                    "support": 615.0
                },
                "2": {
                    "precision": 0.18,
                    "recall": 0.17647058823529413,
                    "f1-score": 0.1782178217821782,
                    "support": 612.0
                },
                "3": {
                    "precision": 0.18351477449455678,
                    "recall": 0.18819776714513556,
                    "f1-score": 0.1858267716535433,
                    "support": 627.0
                },
                "4": {
                    "precision": 0.2110939907550077,
                    "recall": 0.2140625,
                    "f1-score": 0.2125678820791311,
                    "support": 640.0
                },
                "accuracy": 0.19548387096774195,
                "macro avg": {
                    "precision": 0.19565779400836264,
                    "recall": 0.19535013147212554,
                    "f1-score": 0.19542279935558302,
                    "support": 3100.0
                },
                "weighted avg": {
                    "precision": 0.19570185135927298,
                    "recall": 0.19548387096774195,
                    "f1-score": 0.19551290180016764,
                    "support": 3100.0
                }
            }
        },
```


Şekil 4.9: Yapay Zeka Modelinin Performans Değerleri

Yapay zeka modelinin performansını değerlendirirken, modelin doğruluk ve F1-skoru metriklerine odaklandık. Testlerin sonuçları, veri setinin büyüklüğünün

artmasının her zaman performans artışı sağlamadığını göstermektedir. Aksine, veri seti büyüdükçe modelin doğruluğu ve diğer performans metriklerinde düşüş gözlemlenmiştir. Bu durum, modelin daha fazla çeşitliliğe ve karmaşıklığa maruz kalması nedeniyle tahminlerde zorluk yaşadığını göstermektedir.

Özetlemek gerekirse, modelin performansı şu şekildedir:

500 Kullanıcı: Doğruluk %22, F1-Skoru %21.86

5000 Kullanıcı: Doğruluk %20.27, F1-Skoru %20.11

15000 Kullanıcı: Doğruluk %19.55, F1-Skoru %19.54

Bu bulgular, modelçin performansını artırmak için daha fazla veri değil, daha nitelikli ve temsili veri kullanılması gerektiği anlamına geliyor. Gelecekteki çalışmalar, veri ön işleme, özellik mühendisliği ve model optimizasyonu gibi tekniklerle modelin performansını artırmayı hedefleyebilir.

5. SONUÇ

Bu tez çalışmasında, optimizasyon algoritmaları ve yapay zeka kullanarak kişiselleştirilmiş menü planları oluşturma süreci ele alınmıştır. Çalışmanın ana hedefi, kullanıcıların beslenme alışkanlıklarına ve tercihlerine göre özelleştirilmiş menüler sunarak, hem sağlık hem de memnuniyet açısından optimal çözümler elde etmektir.

Gerçekleştirilen testler sonucunda, yapay zeka ve optimizasyon tekniklerinin birlikte kullanılmasıyla kullanıcı odaklı menü planları oluşturulabildiği gözlemlenmiştir. Ancak, yapay zeka modellerinin performansı beklentilerin altında kalmış ve özellikle veri setlerinin büyüklüğüne bağlı olarak model doğruluğunda düşüşler görülmüştür. Bu durum, modelin daha fazla çeşitliliğe ve karmaşıklığa maruz kalması nedeniyle tahminlerde zorluk yaşadığını göstermektedir.

Bu çalışmanın literatüre ve pratik uygulamalara sunduğu katkılar, DP ve KTDP tekniklerinin, besin değeri kısıtlamaları ve kullanıcı tercihleri gibi faktörleri dikkate alarak optimal menüler oluşturmadaki etkili kullanımını içermektedir. Ayrıca, yapay zeka tekniklerinin entegrasyonu, kişiselleştirilmiş çözümler sunma noktasında önemli avantajlar sağlamıştır. Ancak, model performansının düşük olması, daha fazla veri ve model iyileştirme çalışmaları gerektirdiğini ortaya koymaktadır.

Gelecek çalışmalarda, model performansını artırmak için daha nitelikli ve temsili veri setlerinin kullanılması gerekmektedir. Yapay zeka modellerinin daha ileri düzey tekniklerle optimize edilmesi ve farklı algoritmaların karşılaştırılması da faydalı olacaktır. Ayrıca, sistemin kullanıcı geri bildirimleri ile sürekli iyileştirilmesi ve kullanıcı memnuniyetinin artırılması, gelecekteki çalışmalar için önemli bir odak noktası olabilir.

Sonuç olarak, bu tez çalışması, optimizasyon algoritmaları ve yapay zeka tekniklerinin kişiselleştirilmiş menü planlamada nasıl kullanılabileceğini göstermiştir. Çalışmanın sonuçları, bu tekniklerin potansiyelini ortaya koymakla

birlikte, model performansının artırılması ve daha geniş veri setlerinin kullanılması gerekliliğini de vurgulamaktadır. Gelecek çalışmalarda, bu alandaki araştırmaların daha da derinleştirilmesi ve genişletilmesi, kişiselleştirilmiş beslenme çözümlerinin geliştirilmesine önemli katkılar sağlayacaktır.

KAYNAKÇA

A. Cutler, D. R. Cutler, and J. R. Stevens, “Random Forests,” in Ensemble Machine Learning, Springer, 2012, pp. 157-175.

A. Hasnaoui, A. Omari and Z. -E. Azzouz, "Optimization of Building Energy based on Mixed Integer Linear Programming," 2022 2nd International Conference on Advanced Electrical Engineering (ICAEE), Constantine, Algeria, 2022, pp. 1-6, doi: 10.1109/ICAEE53772.2022.9962068.

A. Liaw and M. Wiener, “Classification and Regression by randomForest,” R News, vol. 2, no. 3, pp. 18-22, 2002.

Amitabh, Basu. (2022). Complexity of branch-and-bound and cutting planes in mixed-integer optimization. Mathematical Programming, 198(1), 787-810. Available from: 10.1007/s10107-022-01789-5

Atlanta, Chakraborty., Vijay, Chandru., M., R., Rao. (2020). A linear programming primer: from Fourier to Karmarkar. Annals of Operations Research, 287(2):593-616

Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5–32

Chandru, V., & Rao, M. (1998). Linear Programming

Christian, Hoover. (2023). Significance of Linear Programming for Optimization. International Journal of Advanced Research in Science, Communication and Technology, 179-18

Christopher Malefors, Luca Secondi, Stefano Marchetti, Mattias Eriksson, Food waste reduction and economic savings in times of crisis: The potential of machine learning methods to plan guest attendance in Swedish public catering during the Covid-19 pandemic, Socio-Economic Planning Sciences, Volume 82, Part A, 2022,

Daniel G. Espinoza. (2006) On Linear Programming, Integer Programming and Cutting Planes

Dantzig, G. (1982). Reminiscences about the origins of linear programming. , 78-86.

Erdem, M. B. (2023). Otel İşletmelerinde Oluşan Gıda İsrafında Tedarikçi Rolünün Değerlendirilmesi. Journal of Eurasia Tourism Research, 4(Türk Turizminin Geçmiş ve Gelecek Yüzyılı), 78-89.

Feng, W., Sui, H., Tu, J., Huang, W., & Sun, K. (2018). A novel change detection approach based on visual saliency and random forest from multi-temporal high-resolution remote-sensing images. International Journal of Remote Sensing, 39(22), 7998–8021.

Frontiers, "Random Forest Algorithm for the Classification of Neuroimaging Data in Alzheimer's Disease: A Systematic Review," 2017.

G. Arvindaraj, B. Manikandan, R. Rokesh and M. Abinaya, "Opportunities of Machine Learning Techniques in the Catering Industry- A Survey," 2023 Second International Conference on Electronics and Renewable Systems, Tuticorin, India, 2023, pp. 1405-1409, doi: 10.1109/ICEARS56392.2023.10085004.

Harrington, P. (2012). Machine learning in action. Simon and Schuster

Kudak. (2007) Doğrusal programlama ve bulanık doğrusal programlama savunma silahlarının dağıtımında matlab uygulaması

L. Breiman, "Bagging predictors," Machine Learning, vol. 24, no. 2, pp. 123–140, 1996, doi: 10.1007/BF00058655.

L. Breiman, "Random Forests," Machine Learning, vol. 45, no. 1, pp. 5–32, Oct. 2001, doi: 10.1023/A:1010933404324.

L. Breiman, J. Friedman, R. Olshen, and C. Stone, "Classification and Regression Trees," Wadsworth, 1984.

M. Fatima, M. Pasha, M. Fatima, and M. Pasha, "Survey of Machine Learning Algorithms for Disease Diagnostic," Journal of Intelligent Learning Systems and Applications, vol. 9, no. 1, pp. 1–16, Jan. 2017.

Maria-Florina, Balcan., Siddharth, Prasad., Tuomas, Sandholm., Ellen, Vitercik. (2022). Structural Analysis of Branch-and-Cut and the Learnability of Gomory Mixed Integer Cuts. abs/2204.07312 doi: 10.48550/arXiv.2204.07312

Matthias Miltenberger. (2023) Linear Programming in MILP Solving A Computational Perspective

Melanie Speck, Lynn Wagner, Felix Buchborn, Fara Steinmeier, Silke Friedrich, Nina Langen. "How public catering accelerates sustainability: a German case study." Sustainability Science, 2022.

Michael, Aggrey, Okoth., Ronghua, Shang., Licheng, Jiao., Jehangir, Arshad., Ateeq, Ur, Rehman., Habib, Hamam. (2022). A Large scale Evolutionary Algorithm based on Determinantal Point Processes for Large Scale Multi-Objective Optimization Problems. Electronics, 11(20):3317-3317. doi: 10.3390/electronics11203317

Nasteski, V. (2017). An overview of the supervised machine learning methods. HORIZONS.B, 4, 51–62. https://doi.org/10.20544/HORIZONS.B.04.1.17.P05

Neşe Yalçın. (2005) Klasik doğrusal programlama ve bulanık doğrusal programlamanın karşılaştırmalı bir analizi

Orchard-Hays, W. (1958). Evolution of Linear Programming Computing Techniques. Management Science, 4, 183-190.

Precup, R., Hedrea, E., Roman, R., Petriu, E., Szedlak-Stînean, A., & Bojan-Dragos, C. (2020). Experiment-Based Approach to Teach Optimization Techniques. IEEE Transactions on Education, 64, 88-94.

Preeti, Tewari., Bhawna, Agrawal. (2022). A study of linear programming technique. International Journal of Statistics and Applied Mathematics, 7(2):54-56. doi: 10.22271/maths.2022.v7.i2a.796

R. E. Schapire, “The boosting approach to machine learning: An overview,” MSRI Workshop on Nonlinear Estimation and Classification, 2002.

Robert J. Vanderbei. Latest edition (2020), Linear Programming Foundations and Extensions

Russell, S., & Norvig, P. (2003). Artificial intelligence - a modern approach, 2nd Edition. Prentice Hall series in artificial intelligence

S. Hou, D. Zhu and J. Xu, "Artificial Intelligence, Financial Canteen and Internal Control: A Case Study of Chinese Catering Industry," 2022 3rd International Conference on Electronic Communication and Artificial Intelligence, Zhuhai, China, 2022, pp. 310-313, doi: 10.1109/IWECAI55315.2022.00066.

Sambudi Hamali, Cecep Hidayat, Aisyah Widya Nur Shadrina, Astika Alivia Pramesti, Fransisca Fortunata Handoyo. "Improving Warehouse Layout and Allocation Optimization in Catering Services Company." 2019 IEEE International Conference on Information Management and Technology.

Stavropoulos, A. (2012) Mixed integer programming and constraint programming for production planning and scheduling at AC

Şahin, S. (2022). Orta Anadolu termal turizm projesi kapsamındaki işletmelerde menü analizi. Nevşehir Hacı Bektaş Veli Üniversitesi.

T. Hastie, R. Tibshirani, and J. Friedman, “The Elements of Statistical Learning: Data Mining, Inference, and Prediction,” Springer, 2009.

T. Kam Ho, “Random decision forests,” Proceedings of 3rd International Conference on Document Analysis and Recognition, vol. 1, pp. 278-282, 1995.

Vapnik, V., Guyon, I., & Hastie, T. (1995). Support vector machines. Mach. Learn, 20(3), 273-297.

Vesna, Antoska, Knights., Mirela, Kolak., Gordana, Markovikj., Jasenka, Gajdoš, Kljusurić. (2023). Modeling and Optimization with Artificial Intelligence in Nutrition. Applied Sciences, 13(13):7835-7835. doi: 10.3390/app13137835

X. Gong, "Optimization Algorithm of Logistics Warehousing and Distribution Path based on Artificial Intelligence Technology," 2022 International Symposium on Advances in Informatics, Electronics and Education (ISAIEE), Frankfurt, Germany, 2022, pp. 371-375, doi: 10.1109/ISAIEE57420.2022.00083.

EKLER

Ek-1: Yemeklerin Besin değerleri

id	food	portion_weight	color	consistency	energy	protein	fat	carbohydrate	fiber
1	Et haşlama	100	Kahverengi	Yumuşak	137.4	13.5	8.9	0.8	0.2
2	Et kızartma	150	Kahverengi	Katı	257.8	42.5	9.6	0.0	0.0
3	Rosto et(patatesli)	160	kahverengi	katı	193.1	16.6	8.3	12.4	1.0
4	Misket köfte	150	kahverengi	yumuşak	303.3	29.5	20.4	0.7	0.0
5	Fırın Köfte	150	kırmızı	katı	279.6	23.3	14.6	13.7	1.2
6	Kadınbudu Köfte	100	kahverengi	katı	176.6	9.5	12.2	7.5	0.4
7	Terbiyeli Köfte	250	kırmızı	yumuşak	390.2	26.2	24.3	16.6	1.1
8	İzmir Köfte	180	kahverengi	katı	264.6	15.6	12.7	21.4	1.7
9	Tekirdağ köfte	100	kahverengi	katı	184.5	16.8	11.0	4.7	0.6
10	Hindi Izgara Köfte	100	kahverengi	katı	107.3	17.7	2.3	3.8	0.4
id	food	portion_weight	color	consistency	energy	protein	fat	carbohydrate	fiber
11	Kaşarlı köfte	150	kahverengi	katı	339.5	25.8	24.4	4.6	0.5

12	Sulu köfte	180	kırmızı	yumuşak	219.4	17.5	11.8	10.9	1.3
13	Çiftlik köfte	190	kahverengi	katı	251.1	14.7	11.9	20.9	2.8
14	Izgara Köfte	100	kahverengi	katı	199.8	16.2	12.3	6.2	0.5
15	Döner kebap	300	kahverengi	katı	342.0	18.4	9.8	44.2	2.0
16	Cağ Kebabı	100	kahverengi	katı	302.3	21.1	24.2	0.0	0.0
17	Bahçıvan Kebabı	250	kırmızı	yumuşak	283.2	26.0	14.1	12.6	2.3
18	Patlıcan kebabı	150	kırmızı	katı	171.7	11.7	12.9	2.3	1.2
19	Ciğer Tas Kebabı	120	kahverengi	katı	189.9	11.6	11.2	10.6	0.7
20	İskender Kebabı	300	kahverengi	katı	522.7	25.7	24.7	48.2	3.7
id	food	portion_weight	color	consistency	energy	protein	fat	carbohydrate	fiber
21	Orman Kebabı	150	kırmızı	yumuşak	199.3	11.7	13.5	7.7	1.5
22	Tas kebabı	100	kahverengi	yumuşak	138.6	9.8	8.0	6.7	0.7
23	Tandır kebabı	150	kahverengi	katı	293.6	22.5	22.4	1.0	0.2
24	Tavuk Şiş Kebap	300	beyaz	katı	304.0	41.6	13.3	4.2	1.5
25	Kuzu Şiş Kebap	300	kahverengi	katı	375.7	21.3	24.6	17.6	3.1
26	Kuzu Pirzola	120	kahverengi	katı	316.1	30.1	21.9	0.0	0.0
27	Tavuk Pane	100	sarı	katı	207.2	16.0	7.5	18.7	1.1
28	Tavuk Haşlama	100	beyaz	katı	188.3	25.9	9.4	0.0	0.0

29	Tavuk Göğüs Şinitzel	100	sarı	katı	207.2	16.0	7.5	18.7	1.1
30	Tavuk Kanat (ızgara)	120	kahverengi	katı	273.6	26.1	19.0	0.0	0.0
id	food	portion_weight	color	consistency	energy	protein	fat	carbohydrate	fiber
31	Buğu kebabı	250	kahverengi	katı	320.9	24.4	23.1	4.0	1.7
32	Tavuk ciğeri	90	kahverengi	katı	142.7	22.9	5.0	1.3	0.0
33	Fırında tavuk	200	kahverengi	katı	211.8	37.6	6.4	0.5	0.0
34	Kremalı Mantarlı Tavuk	180	sarı	katı	185.0	27.6	6.8	3.2	1.0
35	Köri SosluTavuk	180	sarı	katı	229.7	26.2	10.8	6.8	0.3
36	Sebzeli Tavuk Sote	180	kırmızı	katı	110.6	16.0	2.0	6.6	2.0
37	Tavuk sote	180	kahverengi	katı	105.0	19.4	0.8	4.5	1.6
38	Hamsi tava	150	kahverengi	katı	180.3	35.5	4.1	0.0	0.0
39	Fırında Levrek	150	beyaz	katı	140.2	31.6	1.3	0.0	0.0
40	Patatesli Yumurta	140	sarı	katı	146.2	7.7	6.3	14.1	1.2
id	food	portion_weight	color	consistency	energy	protein	fat	carbohydrate	fiber
41	Menemen	180	kırmızı	yumuşak	216.0	8.4	18.5	4.1	1.3
42	Etli Kış Türlüsü	180	kırmızı	yumuşak	150.6	9.3	8.2	9.7	3.2
43	Etli Yaz Türlüsü	180	kırmızı	yumuşak	134.2	8.5	7.3	8.4	2.0
44	Etli Bamya	200	kırmızı	yumuşak	127.6	9.8	8.5	2.8	1.7

45	Etli Taze Fasulye	180	kırmızı	yumuşak	122.2	9.2	6.6	6.3	2.4
46	Hünkar Beğendi	300	kırmızı	yumuşak	299.7	19.9	21.0	7.9	2.6
47	Sebzeli Etli Güveç	250	kırmızı	yumuşak	143.4	14.7	4.3	10.8	3.0
48	Patlıcan Musakka	180	kahverengi	yumuşak	144.6	7.4	11.0	4.0	2.0
49	Karnıyarık	180	kahverengi	yumuşak	160.5	7.5	12.7	4.1	2.1
50	Kıymalı Ispanak	180	yeşil	yumuşak	128.2	8.9	7.3	6.1	1.6
id	food	portion_weight	color	consistency	energy	protein	fat	carbohydrate	fiber
51	Kıymalı Bezelye	180	kırmızı	yumuşak	208.2	15.3	8.7	16.7	6.6
52	Kıymalı Karnabahar	180	kırmızı	yumuşak	134.7	10.4	8.7	3.6	3.9
53	Kıymalı Kapuska	180	kırmızı	yumuşak	137.2	9.0	8.5	6.0	4.1
54	Kıymalı Patates	180	kahverengi	katı	202.6	10.2	8.9	19.7	1.8
55	Etli Nohut	200	kırmızı	yumuşak	211.3	14.9	9.3	16.6	9.1
56	Etli Kuru Fasulye	200	kırmızı	yumuşak	208.9	15.5	8.5	17.3	3.3
57	Etli Bamya	200	kırmızı	yumuşak	127.6	9.8	8.5	2.8	1.7
58	Kıymalı Kabak Dolması	200	yeşil	katı	145.8	9.9	7.3	9.9	1.7
59	Kıymalı Biber Dolması	200	yeşil	katı	183.1	11.7	9.5	12.1	1.6
60	Etli Domates Dolması	225	kırmızı	katı	162.4	5.6	3.3	27.0	1.6
id	food	portion_weight	color	consistency	energy	protein	fat	carbohydrate	fiber

61	Etli lahana sarma	120	kırmızı	katı	131.6	2.1	7.2	14.7	2.1
62	Kıymalı Yaprak Sarma	100	yeşil	katı	128.6	8.5	7.8	6.4	0.7
63	Şehriyeli Güveç	200	kahverengi	katı	291.1	20.9	13.7	21.0	1.6
64	İçli Köfte	100	kahverengi	katı	99.9	4.6	3.7	11.8	1.9
65	Patates Köftesi	100	kahverengi	katı	137.2	4.4	6.3	15.2	1.3
66	Rulo Köfte	180	kahverengi	katı	252.1	22.0	12.7	12.2	2.3
67	Şiş Köfte	150	kahverengi	katı	294.0	25.3	19.2	5.3	0.6
68	Saç Kavurma	120	kahverengi	katı	166.6	12.5	12.1	1.9	0.7
69	Kağıt Kebabı	300	kahverengi	katı	211.5	25.2	8.4	8.1	3.5
70	Kuzu Kapama	250	kahverengi	katı	415.3	31.2	30.6	4.2	1.7
id	food	portion_weight	color	consistency	energy	protein	fat	carbohydrate	fiber
71	Kremalı Domates Çorbası	200	kırmızı	sıvı	103.3	2.0	7.6	6.7	2.0
72	Tavuklu Şehriye Çorbası	200	beyaz	sıvı	99.4	7.3	5.8	4.5	0.3
73	Tavuk Etli Çorba	200	beyaz	sıvı	171.6	12.5	10.3	6.4	0.5
74	Mercimek Çorbası	200	kahverengi	sıvı	103.3	7.7	3.3	10.3	3.2
75	Bezelye Çorbası	200	yeşil	sıvı	126.7	5.9	7.7	8.3	3.1
76	Domatesli Pirinç	200	kırmızı	sıvı	82.2	1.2	4.4	9.3	0.8

	Çorbası								
77	Düğün Çorbası	200	beyaz	sıvı	132.4	8.3	6.8	9.5	0.4
78	Ezogelin Çorbası	200	kırmızı	sıvı	132.4	4.6	6.0	14.9	2.6
79	Sebze Çorbası	200	yeşil	sıvı	123.3	3.3	7.4	10.8	1.0
id	food	portion_weight	color	consistency	energy	protein	fat	carbohydrate	fiber
80	İşkembe Çorbası	200	beyaz	sıvı	99.9	7.9	6.1	2.9	0.1
81	Patates Çorbası	200	beyaz	sıvı	93.7	1.2	5.6	9.2	0.8
82	Un Çorbası	200	beyaz	sıvı	142.4	2.1	8.8	13.9	0.7
83	Yayla Çorbası	200	beyaz	sıvı	94.2	2.8	4.9	9.7	0.3
84	Yeşil Mercimek Çorbası	200	yeşil	sıvı	139.6	6.0	6.0	15.2	2.0
85	Yoğurt Çorbası	200	beyaz	sıvı	190.2	5.2	16.3	5.5	0.1
86	Balık Çorbası	200	beyaz	sıvı	86.0	11.2	2.2	5.1	1.3
87	Bulgur Çorbası	200	kahverengi	sıvı	97.0	1.9	4.6	11.8	2.4
88	Domates Çorbası	200	kırmızı	sıvı	148.7	3.4	8.6	14.3	1.5
89	Ekşili Çorba	200	kahverengi	sıvı	106.1	4.2	4.3	12.4	3.3
id	food	portion_weight	color	consistency	energy	protein	fat	carbohydrate	fiber
90	Kelle Paça Çorbası	200	beyaz	sıvı	121.9	6.1	9.9	2.1	0.1

91	Kremalı Mantar Çorbası	200	beyaz	sıvı	115.2	6.5	6.5	7.5	1.8
92	Nohutlu Pirinç Çorbası	200	beyaz	sıvı	141.0	5.7	3.0	22.4	5.0
93	Salçalı Pirinç Çorbası	200	kırmızı	sıvı	86.0	1.0	4.9	9.5	0.4
94	Pirinçli Kabak Çorba	200	yeşil	sıvı	89.4	3.6	4.5	8.4	1.0
95	Sütlü Arpa Şehriye Çorbası	200	beyaz	sıvı	74.1	2.6	3.1	9.0	0.6
96	Tarhana Çorbası	200	kırmızı	sıvı	134.8	23.3	1.8	6.0	0.5
97	Yulaf Özü Çorbası	200	beyaz	sıvı	79.3	3.9	4.1	6.7	0.6
98	Şehriye Çorbası	200	beyaz	sıvı	93.7	1.8	5.3	9.5	0.9
id	food	portion_weight	color	consistency	energy	protein	fat	carbohydrate	fiber
99	Ispanaklı Lahana Çorbası	200	yeşil	sıvı	34.4	3.0	0.2	5.0	1.3
100	Sebzeli Kırmızı Mercimek Çorbası	200	kahverengi	sıvı	125.2	4.9	5.5	13.8	2.9
101	Puf Böreği	50	sarı	katı	264.8	5.5	19.3	17.6	0.6
102	Serpme Börek	100	sarı	katı	204.8	3.7	15.7	12.5	0.6
103	Patatesli Börek	125	sarı	katı	297.0	7.1	13.7	35.7	1.9
104	Peynirli Börek	70	sarı	katı	194.6	6.5	10.7	18.0	0.9
105	Peynirli Yufka Böreği	70	sarı	katı	161.4	5.8	7.8	16.6	1.1

106	Ispanaklı Yufka Böreği	70	sarı	katı	102.2	3.1	4.7	11.7	1.1
107	Gülböreği	100	sarı	katı	132.4	5.3	6.4	13.1	0.8
id	food	portion_weight	color	consistency	energy	protein	fat	carbohydrate	fiber
108	Peynirli Su Böreği	70	sarı	katı	95.0	2.2	4.9	10.5	0.5
109	Talaş Böreği	50	sarı	katı	100.3	2.0	7.7	6.0	0.3
110	Kıymalı Tepsi Böreği	50	sarı	katı	100.5	3.6	4.4	11.4	0.7
111	Peynirli Sigara Böreği	120	sarı	katı	138.2	8.7	9.0	5.4	0.3
112	Nohutlu Bulgur Pilavı	150	kırmızı	katı	96.4	2.3	4.2	12.3	2.6
113	Pirinç Pilavı	150	beyaz	katı	166.3	2.3	6.5	24.5	0.7
114	Bulgur pilavı	150	kırmızı	katı	142.0	2.8	5.1	21.1	3.2
115	Buhara Pilavı	150	kahverengi	katı	124.8	3.9	4.8	16.4	0.6
116	Büryan Pilavı	150	beyaz	katı	164.2	6.8	7.5	17.2	0.7
117	Domatesli Pilav	150	kırmızı	katı	117.9	2.6	1.0	24.2	1.2
id	food	portion_weight	color	consistency	energy	protein	fat	carbohydrate	fiber
118	Havuçlu Pilav	150	beyaz	katı	124.0	2.5	1.2	25.5	0.9
119	İç Pilav	150	beyaz	katı	213.7	5.5	7.5	30.6	1.2
120	Sebzeli Pilav	150	beyaz	katı	156.0	5.2	2.2	28.3	3.3
121	Şehriye Pilavı	150	kahverengi	katı	171.7	2.7	6.6	25.3	0.9

122	Şehriyeli Pirinç Pilavı	150	beyaz	katı	202.2	3.7	5.3	34.5	1.2
123	Zeytinyağlı Biber Dolma	200	yeşil	katı	176.4	2.9	8.4	21.7	1.9
124	Zeytinyağlı Yaprak Sarma	100	yeşil	katı	110.2	2.3	5.5	12.9	1.0
125	Zeytinyağlı Taze Fasulye	120	yeşil	yumuşak	84.3	2.3	5.9	5.4	2.0
126	Zeytinyağlı Bakla	120	yeşil	yumuşak	148.0	7.1	7.1	13.5	3.3
id	food	portion_weight	color	consistency	energy	protein	fat	carbohydrate	fiber
127	Zeytinyağlı Pırasa	120	beyaz	yumuşak	96.1	2.1	6.4	7.4	2.3
128	Zeytinyağlı Kereviz	120	beyaz	yumuşak	97.8	1.9	8.1	4.3	4.2
129	Zeytinyağlı Barbunya	120	kahverengi	yumuşak	153.4	2.9	11.6	9.2	3.1
130	Zeytinyağlı Enginar	160	beyaz	yumuşak	162.1	3.1	11.7	10.6	8.1
131	Zeytinyağlı Mantar	160	kahverengi	yumuşak	94.1	4.2	6.9	4.0	3.5
132	Sebze Kızartması	100	yeşil	katı	97.5	1.3	9.3	2.3	1.4
133	Sebze Yemeği	180	kırmızı	yumuşak	131.6	3.9	5.4	16.3	4.0
134	Fırında Makarna	250	sarı	katı	305.3	12.2	12.7	35.3	2.3
135	Sade Makarna	200	sarı	katı	199.3	5.5	6.1	30.2	2.6
136	Kaşar Peynirli Makarna	200	sarı	katı	254.8	8.4	9.6	33.3	2.4

id	food	portion_weight	color	consistency	energy	protein	fat	carbohydrate	fiber
137	Domates Soslu Makarna	200	kırmızı	katı	262.4	12.2	9.0	32.4	3.0
138	Havuç Soslu Makarna	200	sarı	katı	252.9	7.2	9.2	34.7	3.3
139	Mantar Soslu Makarna	200	sarı	katı	276.8	7.7	9.6	39.3	3.5
140	Patlıcan Soslu Makarna	200	sarı	katı	231.4	6.4	7.9	33.2	3.3
141	Erişte	125	sarı	katı	325.3	11.5	3.6	60.7	3.4
142	Mantar Sote	180	kahverengi	yumuşak	74.0	4.9	3.6	5.2	3.5
143	Milföy Böreği	35	sarı	katı	131.3	2.0	8.2	12.4	0.8
144	İmambayıldı	180	kırmızı	katı	140.7	2.1	12.6	4.8	2.4
145	Mercimekli Kabak Yemeği	180	yeşil	yumuşak	143.7	6.8	6.8	13.4	3.3
id	food	portion_weight	color	consistency	energy	protein	fat	carbohydrate	fiber
146	Sebze Graten	180	yeşil	katı	242.6	9.0	21.1	4.4	1.1
147	Zeytinyağlı Fava	60	beyaz	katı	54.6	4.1	2.2	4.5	4.4
148	Kabak Bayıldı	180	yeşil	katı	145.0	3.0	12.7	4.5	2.0
149	Kabak Kalye	180	yeşil	yumuşak	128.2	8.8	7.9	5.0	1.1
150	Patlıcan Musakka	180	kahverengi	yumuşak	144.6	7.4	11.0	4.0	2.0
151	Domatesli Patlıcan	150	kırmızı	yumuşak	119.7	1.7	10.5	4.5	2.0

	Salatası								
152	Domates Salatası	100	kırmızı	yumuşak	74.3	0.9	6.6	2.5	1.2
153	Havuç Salatası	100	turuncu	katı	78.2	0.8	5.5	6.1	2.7
154	Mevsim Salata	100	yeşil	katı	68.1	1.0	6.1	2.1	1.1
155	Kırmızı Lahana Salatası	100	kırmızı	katı	77.4	1.3	6.4	3.3	2.2
id	food	portion_weight	color	consistency	energy	protein	fat	carbohydrate	fiber
156	Kıvırcık Salatası	100	yeşil	katı	60.5	1.6	5.3	1.3	1.7
157	Marul Salatası	60	yeşil	katı	34.8	0.9	2.9	1.0	0.7
158	Patates Salatası	100	sarı	katı	121.2	1.7	7.1	12.1	1.4
159	Amerikan Salatası	100	beyaz	yumuşak	46.6	1.5	1.5	6.4	2.2
160	Yoğurtlu Semizotu Salatası	100	beyaz	yumuşak	46.4	2.9	2.3	3.1	1.0
161	Börülce Salatası	50	yeşil	katı	57.0	4.3	0.8	7.9	1.6
162	Gavurdağı Salatası	100	kırmızı	katı	145.8	2.5	12.0	7.0	2.0
163	HavuçTurp Salatası	100	turuncu	katı	49.5	1.0	3.0	3.8	1.7
164	Mercimek Salatası	100	yeşil	katı	140.8	7.0	4.2	18.4	3.8

id	food	portion_weight	color	consistency	energy	protein	fat	carbohydrate	fiber

165	Yoğurtlu Kereviz Salatası	150	beyaz	yumuşak	44.1	2.9	1.6	4.0	4.4
166	Çoban Salata	150	kırmızı	katı	62.7	1.2	4.5	4.0	1.7
167	Elma Kompostosu	150	beyaz	sıvı	288.2	0.3	0.0	70.4	1.9
168	Kuru Kayısı Hoşafı	150	sarı	sıvı	247.0	1.7	0.2	57.1	6.0
169	Kuru Erik Hoşafı	150	sarı	sıvı	240.2	0.8	0.2	57.0	6.1
170	Vişne Kompostosu	150	kırmızı	sıvı	241.3	0.9	0.5	56.3	1.1
171	Şeftali Kompostosu	150	sarı	sıvı	280.0	0.7	0.1	67.9	1.5
172	Yoğurt	200	beyaz	yumuşak	138.1	7.8	7.5	8.7	0.0
173	Cacık	200	beyaz	yumuşak	142.4	5.5	10.0	6.9	0.6
174	Piyaz	150	beyaz	katı	135.2	9.2	4.1	14.9	3.6
id	food	portion_weight	color	consistency	energy	protein	fat	carbohydrate	fiber
175	Kırmızı Elma	120	kırmızı	katı	73.1	0.4	0.1	17.2	2.4
176	Yeşil Elma	120	yeşil	katı	73.1	0.4	0.1	17.2	2.4
177	Tarbzon hurması	100	turuncu	katı	71.0	0.6	0.3	16.0	2.5
178	Armut	125	beyaz	katı	65.1	0.6	0.4	15.5	3.5
179	Muz	140	sarı	katı	84.0	1.1	0.2	18.8	1.9
180	Şeftali	185	turuncu	katı	75.6	1.4	0.2	16.4	3.1

181	Portakal	140	turuncu	katı	60.6	1.4	0.3	11.6	3.1
182	Mandalina	150	turuncu	katı	75.3	1.0	0.5	15.2	2.6
183	Üzüm	150	yeşil	katı	104.3	1.0	0.4	22.9	2.4
184	Karpuz	150	kırmızı	katı	57.4	0.9	0.3	12.4	0.4
id	food	portion_weight	color	consistency	energy	protein	fat	carbohydrate	fiber
185	kavun	220	sarı	katı	110.4	1.8	0.2	24.8	1.5
186	Mürdüm Eriği	175	kırmızı	katı	75.7	1.1	0.2	15.4	4.0
187	Fıstıklı Tel Kadayıf	100	kahverengi	katı	332.7	3.8	12.6	50.6	1.5
188	Cevizli Kadayıf	100	kahverengi	katı	410.4	5.5	14.4	64.1	1.8
189	Lokma	90	kahverengi	katı	204.1	2.0	6.0	35.2	0.4
190	Şekerpare	100	kahverengi	katı	160.6	2.1	7.0	22.2	0.5
191	Revani	100	sarı	katı	291.1	3.6	7.1	52.5	1.6
192	Yoğurt Tatlısı	100	sarı	katı	149.1	3.4	1.7	29.7	1.3
193	Kalburabastı	150	kahverengi	katı	254.9	2.3	12.5	33.1	0.6
194	Baklava	150	kahverengi	katı	194.8	3.6	11.2	20.0	1.0
id	food	portion_weight	color	consistency	energy	protein	fat	carbohydrate	fiber
195	Ayva Tatlısı	150	kırmızı	katı	199.3	1.3	5.6	35.0	5.5
196	Balkabağı Tatlısı	100	turuncu	katı	136.0	1.6	3.7	23.7	1.8

197	Hamur Tatlısı	150	kahverengi	katı	226.6	3.8	2.7	46.2	0.8
198	Sütlü İrmik Tatlısı	100	beyaz	yumuşak	78.5	0.7	0.1	18.5	0.5
199	İrmik Helvası	100	kahverengi	yumuşak	309.8	3.5	12.9	44.6	2.2
200	Tahin Helvası	35	kahverengi	katı	151.9	3.6	7.5	19.8	1.4
201	Tulumba Tatlısı	90	kahverengi	katı	140.7	2.9	6.7	17.3	0.4
202	Muhallebi	200	beyaz	yumuşak	190.7	5.1	4.9	31.3	0.2
203	Sütlaç	200	beyaz	yumuşak	229.4	5.0	5.1	40.3	0.2
204	Keşkül	200	beyaz	yumuşak	173.5	2.1	4.9	30.0	1.2
id	food	portion_weight	color	consistency	energy	protein	fat	carbohydrate	fiber
205	Kazandibi	200	kahverengi	yumuşak	277.2	4.5	5.1	52.7	0.2
206	Aşure	200	kahverengi	yumuşak	369.0	7.5	5.2	71.3	8.3
207	Pelte	150	beyaz	yumuşak	301.1	0.9	0.2	71.7	0.4
208	Salatalık turşusu	100	yeşil	katı	6.9	0.3	0.1	0.9	0.4
209	Lahana Turşusu	150	beyaz	katı	23.7	2.1	0.4	0.9	3.0
210	Kakaolu Puding	150	kahverengi	yumuşak	164.2	4.2	4.7	26.0	0.1

Ek-2: Optimizasyon ile ilgili sorular ve kodlar:

Sorular:

1. Yaşınız?

2. Cinsiyetiniz?

3. Aktivite durumunuz?

4. Medeni durumunuz?

5. Köfteleri ne sıklıkla tercih edersiniz?

6. Kebap veya güveç yemeklerini yemek tercihleriniz arasında ne sıklıkla yer alır?

7. Et kızartma yemeklerini ne kadar sık tüketirsiniz?

8. Tavuklu yemekleri ne sıklıkla tercih edersiniz?

9. Balık yemeklerini ne sıklıkla yersiniz?

10. Sebzeli yemekleri ne sıklıkla tercih edersiniz?

11. Zeytinyağlı yemekleri yeme sıklığınız nedir?

12. Etli sebze yemeklerini ne kadar sık tüketirsiniz?

13. Çorbaları ne sıklıkla içersiniz?

14. Pilavı ne kadar sık tüketirsiniz?

15. Börek yeme alışkanlığınız ne kadar sık?

16. Makarna ve erişte yemeklerini ne sıklıkla tercih edersiniz?

17. Salatalar ve soğuk yemekleri ne kadar sık tercih edersiniz?

18. Tatlıları ne sıklıkla tüketirsiniz?

19. Günlük içecek tüketiminiz nasıldır?

20. Meyve tüketim sıklığınız nedir?

Tüm **kodlar** bu adresten erişilebilir durumdadır: https://github.com/shahmirzali49/nutrition_backend

```
def get_diet_menu(week: int,  user_preferences: dict[str, int], db: Session):
    query_meals = db.query(app.models.Meal).statement
    df1 = pd.read_sql_query(query_meals, db.bind)
    df2 = pd.read_csv("kısıtlar.csv")
    df1_indexed = df1.set_index("id")
    user_age = 25
    user_gender = "Genel"
    week = 1

    preferences_percentages = calculate_preferences_percentages(user_preferences)

    categories = defaultdict(list)

    for item in detailed_foods:
        preference_score = user_preferences.get(str(item['id']), 0)
        categories[item['plate_category']] += [(food_id, preference_score) for food_id in item['food_ids']]

    for category, foods in categories.items():
        sorted_food_ids = [food_id for food_id, _ in sorted(foods, key=lambda x: x[1], reverse=True)]
        categories[category] = sorted_food_ids

    def parse_age_group(age_group):
        lower, upper = map(int, age_group.split("-"))
        return range(lower, upper + 1)

    limits = df2[
        (df2["Cinsiyet"] == user_gender)
```

```
            &     (df2["Yaş     Grubu"].apply(lambda     x:     user_age     in
parse_age_group(x)))
        ].iloc[0]

        def find_mandatory_dish_id(df, used_dishes):
            for dish_name in mandatory_dish_names:
                dish_row = df[df["food"] == dish_name]
                if not dish_row.empty:
                    dish_id = dish_row.iloc[0]["id"]
                    if dish_id not in used_dishes:
                        return dish_id, dish_name
            return None, None

        all_mandatory_dish_names = [
            "Etli Nohut",
            "Etli Kuru Fasulye",
            "Zeytinyağlı Barbunya",
            "Börülce Salatası",
            "Mercimek Salatası",
            "Piyaz",
        ]

        mandatory_dish_names = random.sample(all_mandatory_dish_names, 4)

        mandatory_dish_ids = [
            df1[df1["food"] == name].iloc[0]["id"]
            for name in mandatory_dish_names
            if not df1[df1["food"] == name].empty
        ]

        vegetable_meals_ids = [  123,124,125,126 ... 150 ]
        salad_meals_ids = df1[
            df1["food"].str.contains("Salata")
```

```
            & df1["id"].isin(categories["dessert_salad"])
        ]["id"].tolist()

        soup_meals_ids = [ 71,72,73,74, 75 . .. 100 ]
        compote_hosaf_meals_ids = [167, 168, 169, 170, 171]

        etli_dolma_sarma_ids = df1[
            df1["food"].str.contains("Etli")
            & (
                df1["food"].str.contains("Dolma")
                | df1["food"].str.contains("Sarma")
            )
        ]["id"].tolist()

        pilav_ids = df1[df1["food"].str.contains("Pilav")]["id"].tolist()
        specified_meal_ids = df1[
            df1["food"].str.contains(
                "Pirinç      Pilavı|Şehriyeli      Pirinç      Pilavı|Yayla
Çorbası|Sütlaç", regex=True
            )
        ]["id"].tolist()

        used_dishes = set()
        used_mandatory_dishes = set()

        df1["color"] = df1["color"].str.lower()
        df1["consistency"] = df1["consistency"].str.lower()

        available_colors = df1["color"].drop_duplicates().values.tolist()
        available_textures                                              =
df1["consistency"].drop_duplicates().values.tolist()

        weekly_menus = []
```

```
        dish_costs = {food_id: preferences_percentages.get(str(item['id']), 0)
                      for item in detailed_foods for food_id in item['food_ids']}

        def update_objective_function(prob, dish_vars, dish_costs):
            cost_function = pl.lpSum([dish_costs[dish_id] * dish_vars[dish_id] for dish_id in dish_vars])
            prob.setObjective(cost_function)

        for week in range(1, week + 1):
            mandatory_dish_day = random.choice(range(1, 6))
            mandatory_dish_id, mandatory_dish_name = None, None

            weekly_menu = WeeklyMenu(week=week, menus=[])

            for day in range(1, 6):
                prob = pl.LpProblem(f"Menu_Optimization_{week}_{day}", pl.LpMaximize)

                available_dishes = list(
                    set(df1["id"]) - used_dishes - set(mandatory_dish_ids)
                )

                if day == mandatory_dish_day:
                    available_dishes = list(
                        set(df1["id"]) - used_dishes - set(mandatory_dish_ids[week:])
                    )

                dish_vars = pl.LpVariable.dicts("Dish", available_dishes, 0, 1, pl.LpBinary)
```

```
                for nutrient in ["energy", "carbohydrate", "protein",
"fat", "fiber"]:
                    lower_limit = float(limits[nutrient].split("-")[0])
                    upper_limit = float(limits[nutrient].split("-")[1])

                    prob += pl.LpConstraint(
                        e=pl.lpSum(
                            [
                                df1_indexed.loc[i][nutrient]              *
dish_vars[i]
                                for i in available_dishes
                            ]
                        ),
                        sense=pl.LpConstraintGE,
                        name=f"{nutrient}_lower_bound",
                        rhs=lower_limit,
                    )

                    prob += pl.LpConstraint(
                        e=pl.lpSum(
                            [
                                df1_indexed.loc[i][nutrient]              *
dish_vars[i]
                                for i in available_dishes
                            ]
                        ),
                        sense=pl.LpConstraintLE,
                        name=f"{nutrient}_upper_bound",
                        rhs=upper_limit,
                    )

                for cat, rng in categories.items():
                    prob += (
```

```
                        pl.lpSum([dish_vars[i]  for  i  in  available_dishes
if i in rng]) == 1
                    )

                for color in available_colors:
                    prob += (
                        pl.lpSum(
                            [
                                dish_vars[i]
                                for i in available_dishes
                                if df1_indexed.loc[i]["color"] == color
                            ]
                        )
                        <= 2
                    )
                for texture in available_textures:
                    prob += (
                        pl.lpSum(
                            [
                                dish_vars[i]
                                for i in available_dishes
                                if   df1_indexed.loc[i]["consistency"]   ==
texture
                            ]
                        )
                        <= 2
                    )

                for veg_meal_id in vegetable_meals_ids:
                    for salad_meal_id in salad_meals_ids:
                        if (
                            veg_meal_id in available_dishes
                            and salad_meal_id in available_dishes
```

```
                ):
                    prob += dish_vars[veg_meal_id] +
dish_vars[salad_meal_id] <= 1

            for soup_meal_id in soup_meals_ids:
                for compote_hosaf_meal_id in compote_hosaf_meals_ids:
                    if ( soup_meal_id in available_dishes
                        and compote_hosaf_meal_id in available_dishes
                    ):
                        prob += ( dish_vars[soup_meal_id] +
dish_vars[compote_hosaf_meal_id]
                            <= 1
                        )

            for etli_dolma_sarma_id in etli_dolma_sarma_ids:
                for pilav_id in pilav_ids:
                    if ( etli_dolma_sarma_id in available_dishes
                        and pilav_id in available_dishes
                    ):
                        prob += ( dish_vars[etli_dolma_sarma_id] +
dish_vars[pilav_id] <= 1
                        )
            specified_meal_ids = [
                meal_id for meal_id in specified_meal_ids if meal_id
in dish_vars
            ]
            prob += ( pl.lpSum([dish_vars[meal_id] for meal_id in
specified_meal_ids]) <= 1
            )

            if day == mandatory_dish_day:
                mandatory_dish_id, mandatory_dish_name =
find_mandatory_dish_id(
                    df1, used_mandatory_dishes
                )
```

```
                    if mandatory_dish_id is None:
                        print("Uygun zorunlu yemek kalmadı.")
                    else:
                        used_mandatory_dishes.add(mandatory_dish_id)
                        prob += dish_vars[mandatory_dish_id] == 1
                        used_dishes.add(mandatory_dish_id)


                update_objective_function(prob, dish_vars, dish_costs)


                prob.solve(pl.PULP_CBC_CMD(msg=False))
                total_nutrients = defaultdict(float)


                for v in prob.variables():
                    if v.varValue == 1:
                        dish_id = int(v.name.split("_")[1])
                        used_dishes.add(dish_id)
                selected_dishes = [i for i in available_dishes if
dish_vars[i].value() == 1]
                day_menu_meals = [Meal(
                    id=int(i), food=df1_indexed.loc[i]["food"],
                    price=df1_indexed.loc[i]["price"],
                    color=df1_indexed.loc[i]["color"],
                    consistency=df1_indexed.loc[i]["consistency"]


                ) for i in selected_dishes]


                total_nutrient_values = NutrientValues(
                    energy=sum(df1_indexed.loc[i]["energy"] for i in
selected_dishes),
                    carbohydrate=sum(df1_indexed.loc[i]["carbohydrate"]
for i in selected_dishes),
                    protein=sum(df1_indexed.loc[i]["protein"] for i in
selected_dishes),
```

```
                    fat=sum(df1_indexed.loc[i]["fat"]      for      i      in
selected_dishes),
                    fiber=sum(df1_indexed.loc[i]["fiber"]    for    i    in
selected_dishes)
                )

                day_menu  =  DayMenu(status=f"{prob.status}",  day  =  day,
menu=day_menu_meals, total_nutrient_values=total_nutrient_values)
                weekly_menu.menus.append(day_menu)

            weekly_menus.append(weekly_menu)
        return weekly_menu
```

ÖZGEÇMİŞ

ÖĞRENİM DURUMU

• Lisans : 2017, Azerbaycan Devlet Petrol ve Sanayi Üniversitesi, Mekatronik ve Robotik mühendisliği

• Yüksek Lisans : (2024) T.C. İstanbul Gedik Üniversitesi, Yapay Zeka mühendisliği, Tezli Yüksek Lisans Programı