BURSA TEKNİK ÜNİVERSİTESİ ❖ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

ÖNGÖRÜLÜ MODEL İŞARETLEME DİLİ KULLANILARAK YAPAY ZEKA VE MAKİNE ÖĞRENMESİ MODELLERİNİN WEB SERVİSİ OLARAK SERVİS EDİLMESİ

YÜKSEK LİSANS TEZİ

Ersin YILDIZ

Akıllı Sistemler Mühendisliği Anabilim Dalı

NİSAN 2022

BURSA TEKNİK ÜNİVERSİTESİ ❖ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

ÖNGÖRÜLÜ MODEL İŞARETLEME DİLİ KULLANILARAK YAPAY ZEKA VE MAKİNE ÖĞRENMESİ MODELLERİNİN WEB SERVİSİ OLARAK SERVİS EDİLMESİ

YÜKSEK LİSANS TEZİ

Ersin YILDIZ

Akıllı Sistemler Mühendisliği Anabilim Dalı

Tez Danışmanı: Prof. Dr. Turgay Tugay BİLGİN

NİSAN 2022

BTÜ, Lisansüstü Eğitim Enstitüsü'nün 19324814019 numaralı Yüksek Lisans Öğrencisi Ersin YILDIZ, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı "ÖNGÖRÜLÜ MODEL İŞARETLEME DİLİ KULLANILARAK YAPAY ZEKA VE MAKİNE ÖĞRENMESİ MODELLERİNİN WEB SERVİSİ OLARAK SERVİS EDİLMESİ" başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı : **Prof. Dr. Turgay Tugay BİLGİN**
Bursa Teknik Üniversitesi

Jüri Üyeleri : **Dr. Volkan ALTUNTAŞ** .....
Bursa Teknik Üniversitesi

Dr. Volkan TUNALI .....
Maltepe Üniversitesi

Teslim Tarihi : ..../..../.....
Savunma Tarihi : 29 Nisan 2022

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

Bu tez, 2210-D TÜBİTAK Yurt İçi Sanayiye Yönelik Yüksek Lisans Burs Programı kapsamında, "Öngörülü Model İşaretleme Dili Kullanılarak Yapay Zeka ve Makine Öğrenmesi Modellerinin Web Servisi Olarak Servis Edilmesi" tez konusu ile desteklenmiştir.

İNTİHAL BEYANI

Bu tezde görsel, işitsel ve yazılı biçimde sunulan tüm bilgi ve sonuçların akademik ve etik kurallara uyularak tarafımdan elde edildiğini, tez içinde yer alan ancak bu çalışmaya özgü olmayan tüm sonuç ve bilgileri tezde kaynak göstererek belgelediğimi, aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul ettiğimi beyan ederim.

Ersin YILDIZ

Eşime ve oğluma,

ÖNSÖZ

Veri bilimi ve yapay zeka alanında elde edilen teknolojik gelişmeler bu alanda yapılan çalışmaların günden güne artmasını sağlamış ve çıktıları ile insanlığa yüksek fayda sağlamışlardır. İnsanlık, endüstri, finans, sağlık gibi alanlarda yapılan çalışmalar ile verimlilik artışı, yeni iş kollarının oluşması, karmaşık problemlerin çözülmesi gibi uygulamalardan hayatının hemen her noktasında yararlanmaktadır. Özellikle son dönemde pratik uygulamaların da artışıyla bu alandaki talep artmış, uygulamalar çeşitlenmiştir. Bu çalışmada, bu talebi karşılayacak olan profesyonellerin işlerini kolaylaştıracak, tekrara düştüğü ve zaman kaybettiği süreçleri optimize edecek bir uygulama ortaya konmuştur. Bu yönüyle, bu iki disiplinde gerçekleştirilecek çalışmaların uygulamalar arası entegrasyonunu hızlandıracağını düşünmekteyim.

Öncelikle, tez konusunu seçerken ilgi alanlarımı ve isteklerimi de gözeterek önerilerde bulunan, yüksek lisans eğitimim boyunca gerek derslerde, gerek tez döneminde gerekse de akademik çalışmalarımda desteğini hiç esirgemeyen saygıdeğer danışman hocam Prof. Dr. Turgay Tugay BİLGİN'e sonsuz teşekkürlerimi sunarım.

Bu tez çalışması, TÜBİTAK Bilim İnsanı Destek Programları Başkanlığı'nın 2210-D Yurt İçi Sanayiye Yönelik Yüksek Lisans Burs Programı kapsamında desteklenmiştir. Tezimin gerçekleşmesini destekleyen ve bana burs desteği sağlayan TÜBİTAK'a teşekkürü borç bilirim.

Yüksek lisans eğitimimi destekleyen ve tez çalışmasının gerçekleştirilmesinde, teknik altyapının sağlanmasında büyük katkıları olan Coşkunöz Holding, CITS Bilişim ve Yazılım A.Ş firmalarına, yöneticim Sn. Vedat DAVARCIOĞLU'na ve çalışma arkadaşlarıma teşekkür ederim.

Son olarak, tüm eğitim hayatım boyunca yanımda olan, her konuda desteğini yanımda hissettiğim sevgili aileme ve hayatımın her alanında olduğu gibi, tez çalışmamı hazırlarken de desteğini hiç eksiltmeyen değerli eşime ve oğluma sonsuz teşekkür ederim.

Nisan 2022

Ersin YILDIZ
Bilgisayar Mühendisi

İÇİNDEKİLER

Sayfa

KISALTMALAR

API : Application Programming Interface
CRUD : Create, Read, Update, Delete
CSS : Cascading Style Sheets
ER : Entity Relation
FTP : File Transfer Protocol
HTML : Hyper Text Markup Language
HTTP : Hyper Text Transfer Protocol
IoT : Internet Of Things
JDBC : Java Database Connectivity
JMS : Java Messaging Services
JSON : Javascript Object Notation
JWT : JSON Web Token
LDAP : Lightweight Directory Access Protocol
MPA : Multi Page Application
ORM : Object Relational Mapping
PMML : Predictive Model Markup Language
REST : Representational State Transfer
SaaS : Software as a Service
SMTP : Simple Mail Transfer Protocol
SOAP : Simple Object Access Protocol
SPA : Single Page Application
SQL : Structured Query Language
TCP : Transmission Control Protocol
TDA : Tahmine Dayalı Analitik
UIMA : Unstructured Information Management Architecture
URI : Uniform Resource Identifier
WSGI : Web Server Gateway Interface
XHR : XML Http Request
XML : Extensible Markup Language

SEMBOLLER

Δt	**:** Geçen süre
μ	**:** Ortalama yanıt süresi
SD	**:** Standart sapma
T	**:** Doğruluk

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

ÖNGÖRÜLÜ MODEL İŞARETLEME DİLİ KULLANILARAK YAPAY ZEKA VE MAKİNE ÖĞRENMESİ MODELLERİNİN WEB SERVİSİ OLARAK SERVİS EDİLMESİ

ÖZET

Bilgisayar teknolojisi günden güne gelişmekte ve bu gelişim yukarı yönde ivmelenerek devam etmektedir. Gelişen bilgisayar ve elektronik teknolojisi, bilgisayar özelliği taşıyan cihazların üretimindeki maliyeti düşürmüş, son kullanıcıya ulaşmasını kolaylaştırmıştır. Buna paralel olarak gelişmekte olan internet teknolojisi de son kullanıcıların bilgisayarlarla olan temas noktalarının çeşitlenmesini sağlamış ve insan bilgisayar etkileşimini artmıştır. Bunun bir sonucu olarak günümüz bilgisayar kullanıcıları üretilen ham veride önemli bir kaynak haline gelmişlerdir.

Son kullanıcı pazarının yanında endüstrideki dijital dönüşüm akımının ve buna bağlı çalışmaların bir neticesi olarak kurumların ham veriye olan bakış açısı değişmiş, veri toplama ve anlamlandırma alanındaki çalışmalar önem kazanmıştır. Kurumlar topladıkları ham veriyi depolama, işleme, anlamlandırma ve gelecekte kullanma yönünde çalışmalar yapmaktadırlar.

Bahsedilen gelişmeler neticesinde veri madenciliği ve makine öğrenmesi alanlarında yapılan çalışmalar önem kazanmış, bu alanda iş gücü ihtiyacı artmıştır. Ham verinin toplanması, analiz edilmesi, işlenmesi, model eğitilmesi ve başka bir projeye entegre edilmesi aşamalarının her biri farklı disiplinler içermekte, tek kişinin tüm sürece hakim olması güçleşmektedir.

Bu çalışmada eğitilmiş bir makine öğrenmesi modelinin öngörüleme aşamasında üçüncü taraf herhangi bir uygulama ile entegre olabilmesi için modellerin web servisi olarak servis edilmesi amaçlanmıştır. Böylelikle veri bilimcilerin veya veri mühendislerinin modelin entegrasyonuna ayıracağı zamanı en aza indirmek ve proje maliyetini düşürmek hedeflenmiştir.

Modellerin web servisine dönüştürülmesi çalışmasında model serileştirme yöntemi olarak öngörülü model işaretleme dili (Predictive Model Markup Language: PMML) tercih edilmiştir. PMML birçok veri madenciliği uygulamasınca desteklenmektedir ve model serileştirmede ortak dil olarak kabul edilmiştir. Geliştiricilerin ürettikleri PMML dosyalarını yükleyebilecekler, yönetebilecekleri ve elde edecekleri uçbirimlerin dokümantasyonunu görebilecekleri, hizmet olarak yazılım mantığında çalışan bir de arayüz geliştirilmiştir.

Yapılan çalışmalar neticesinde bir web arayüzü ve bir sunucu taraflı uygulama geliştirilmiştir. Web arayüzü kullanıcı yönetimi ve model yönetimi için arayüz sağlamaktadır. Sunucu taraflı uygulama ise önyüz isteklerini karşılamakta ve ek olarak modellerin seri durumdan çıkarılarak tahmin işleminin gerçekleştirilmesi için bir uygulama arayüzü sunmaktadır.

Çalışma sonunda, literatürde kabul görmüş örnek veri setleri kullanılarak, geliştirilen sistemin performans ölçümleri yapılmış, yük altındaki tepkisi incelenmiştir. Modelin web servisine dönüştürülmesi durumunda tahmin süresindeki değişimin araştırılması ve veri seti büyüklüğünün PMML dokümanının büyüklüğüne etkisinin araştırılması gerçekleştirilmiştir. Sonrasında ise sistemin normal yük altında ve ani yük artışı durumunda ortalama yanıt süresi, doğruluk değeri, hata oranı ve standart sapma metrikleri üzerinden değerlendirmesi yapılmıştır.

Ölçümler sonucunda, sistemin genel olarak istekleri karşılamada kararlı davrandığı görülmüş, fakat bazı sınıflandırıcı algoritmalarda düşük performans gösterirken bazılarında ise yüksek performans gösterebildiği görülmüştür. Dolayısıyla böyle bir sistemde kullanılması hedeflenen model eğitilirken PMML serileştirmesine ve hızlı öngörülemeye uygun algoritmaların seçilmesi önerilmiştir.

Anahtar kelimeler: Yapay Zeka, Makine Öğrenmesi, Veri Madenciliği, PMML, Web Servisleri, Veri Bilimi

DEPLOYING ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING MODELS AS A WEB SERVICE USING PREDICTIVE MODEL MARKUP LANGUAGE

SUMMARY

Computer technology is developing day by day and the development continues by accelerating upwards. Developments in computer and electronic technology have reduced the production costs of devices with computer features and made it easier to reach the end user. In addition, the developing internet technology has also enabled the diversity of the end users' contact points with computers and increased human-computer interaction. As a result of this, today's computer users have become an important source of raw data produced.

In addition to the end-user market, as a result of the digital transformation trend in the industry and related studies, the perspective of corporations on raw data has changed, and studies in the field of data collection and interpretation have gained importance. Corporations are working on storing, processing, interpreting and using the raw data they collect in the future.

As a result of the aforementioned developments, studies in the fields of data mining and machine learning have gained importance, and the need for labor in this field has increased. The stages of collecting, analyzing, processing, training a model and integrating it into another project involve different disciplines, making it difficult for a single person to dominate the whole process.

In this study, it is aimed to serve the models as a web service so that a trained machine learning model can be integrated with any third-party application during the prediction phase. Thus, it is aimed to minimize the time that data scientists or data engineers will devote to the integration of the model and to reduce the cost of the project.

Predictive Model Markup Language (PMML) was preferred as the model serialization method in the conversion of models to web service. PMML is supported by many data mining applications and has been adopted as the common language for model serialization. A software-as-a-service interface has also been developed, where developers can upload, manage and view the documentation of the endpoints they produce.

As a result of the studies, a web interface and a server-side application were developed. The web interface provides interface for user management and model management. The server-side application, on the other hand, handles the frontend requests and additionally provides an application interface for deserializing models and performing the prediction process.

At the end of the study, performance measurements of the developed system were made using sample data sets accepted in the literature, and its response under load was examined. In the case of converting the model to a web service, the change in the prediction time and the effect of the size of the data set on the size of the PMML

document were investigated. Afterwards, the system was evaluated on average response time, throughput, error rate and standard deviation metrics under normal load and in case of sudden load increase.

As a result of the measurements, it was seen that the system was stable in meeting the requests in general, but it was seen that it showed low performance in some classifier algorithms and high performance in others. Therefore, it has been suggested to select algorithms suitable for PMML serialization and fast prediction while training the model intended to be used in such a system.

Keywords: Artifical Intelligence, Machine Learning, Data Mining, PMML, Web Services, Data Science

1. GİRİŞ

İnternet son yıllarda teknolojisi en hızlı değişen ve gelişen kavramlardan bir tanesidir[1]. Buna sebep olan gelişmelerin en başında ise bilgisayar ve elektronik teknolojisindeki gelişmeler gelmektedirler. Bu teknolojik gelişmeler sonucunda bilgisayar niteliğinde olan ve son kullanıcıya hitap eden cihazların maliyetleri düşmüş, ulaşılabilirliği artmıştır. Kişisel bilgisayarların yaygınlaşmaya başladığı ilk dönemlerden bu yana hane başına düşen bilgisayar sayısında katlanarak devam eden bir artış yaşanmaktadır [2]. Kişisel bilgisayarların yanında gelişen çip teknolojisi ve telekomünikasyon alanındaki gelişmelerin bir neticesi olarak cep telefonları, akıllı telefonlar olarak anılmaya başlamış ve son kullanıcıların günlük yaşamında önemli bir rol oynar hale gelmiştir. Akıllı telefonlar ve internet 2.0 kavramı ile birlikte sosyal medya platformları doğmuş, ve kısa sürede çok yüksek kullanıcı sayılarına ulaşmışlardır [3] . Tüm bu gelişmelerin bütününe bakıldığında son kullanıcıların internetin yaygınlaştığı ilk dönemlerdeki "veri tüketen" rolleri günümüzde "veri üreten" rolüne dönüşmüştür. Benzer gelişmeler endüstriyel teknolojilerde de yaşanmış, endüstride üretilen verinin boyutu da önemli bir artış göstermiştir. Bir diğer bakış açısıyla, bu ve benzer gelişmeler büyük veri adı verilen kavramın doğmasına yol açmıştır. Büyük verinin kaynağı ile ilgili net bir ayrım olmasa da genellikle şu üç başlıkta ele alınmışlardır [4] :

- Sosyal Veri: Dünya genelinde en çok bilinen sosyal medya platformlarında kullanıcılarca oluşturulan veri olarak özetlenebilir. Mikrobloglardaki yazılar, paylaşımlar, yorumlar, video yüklemeleri, fotoğraf yüklemeleri, beğeniler bunlara örnek olarak gösterilebilir. Bu veri tek başına anlamsız veya sadece belirli bir kesim için anlamlı olarak görünse dahi, bütünsel olarak ele alındığında tüketici davranışlarının tahmin edilmesinde, pazarlama yöntemi olarak kullanılmasında faydalanılabilmektedir. Sosyal medya mecraları ilk zamanlarından bu yana çeşitlenmiş, toplam kullanıcı sayısına bakıldığında sürekli bir artış yaşanmıştır [3]. Bu istatistiğe bakarak bu artışın -yeni mecraların da katılacağını göz önünde bulundurulduğunda- devam edeceği

söylenebilir. Bu da türde üretilen verilerin paralel olarak artış göstermeye devam edeceğine bir işaret kabul edilebilir.

- Makine Verisi: Makinelerden toplanan verilerin tamamı bu kategoride değerlendirilebilir. Makineler üstüne konumlandırılmış duyargalar (sensörler), endüstriyel ekipmanlar bunlara iyi birer örnektirler. Üretilen makine verilerinin çeşitliliği, hacmi ve değeri Nesnelerin İnterneti (Internet Of Things : IoT) teknolojisindeki gelişime ve yaygınlaşmaya paralel olarak önemli bir artış göstermiş ve göstermeye devam etmektedir. Yapılan araştırmalar 2021 yılı itibariyle 10 milyardan fazla aktif IoT cihazı bulunduğunu ve 2030 yılına gelindiğinde bu sayının 25.4 milyar cihaza çıkmasının beklendiğini göstermektedir. 2025 yılında, IoT cihazları tarafından üretilecek verinin 73.1 Zettabyte (ZB) olacağı tahmin edilmektedir [5].
- İşlemsel Veri: İşlemsel veri, çevrimiçi veya çevrimdışı bilgisayarlarca gerçekleştirilmiş tüm işlemlerin kayıtları olarak açıklanabilir. Faturalar, ödemeler, satın alma kayıtları, kredi işlemleri, para çekme işlemleri gibi örnekler verilebilir. İşlemsel veriler iş zekası uygulamaları için anahtar kaynak niteliğindedir. Bu verilerin işlenmesi ve anlamlandırılması kurumların rekabet avantajı elde etmesinde önemli bir fırsattır. İşlemsel verilere bazı örnekler vermek gerekirse [6]:
    - Finansal İşlem Verisi: Sigorta giderleri, satış verileri, satın alma verileri, hesaptan para çekme ve para yatırma verileri vb.
    - Lojistik İşlem Verisi: Nakliye işlem verileri, nakliye durumu, gerçekleşen nakliye giderleri vb.
    - Puantaj Verisi: Çalışanların işe giriş çıkış saatleri, izin saatleri, işe alım verileri vb.

Örnekleri verilen veriler de tek başlarına anlamsız veya yapılandırılmamış görünseler dahi analiz edildiğinde ve anlamlandırıldığında kurumların ileriye dönük kararlar almasında faydalı olabilecek verilerdir.

Büyük verinin kaynakları yukarıda verildiği gibi artmaya ve çeşitlenmeye devam etmektedir. Bunun bir sonucu olarak dünya genelinde üretilen verinin hacmi neredeyse her iki yılda bir iki katına çıkmaktadır [7].

1.1 Tezin Amacı ve Önemi

Giriş kısmında verilen bilgilere de dayanarak, çok yoğun hacimde ve çeşitlilikte veri üretilen günümüz şartlarında, bu verinin işlenmesi, anlamlandırılması çok kıymet kazanmıştır. Ham verinin işlenmesinde ve anlamlandırılmasında ve bu veriden geleceğe dönük bir bilgi çıkarılmasında, uzun yıllardan beri tahmine dayalı analitik çalışmalarından faydalanılmaktadır [8].

Tahmine dayalı analitik (TDA) uygulamaları bilgisayar bilimleri tarihinde oldukça eskiye dayanmaktadır. Fakat geçerliliğini yitirmemiş, aksine pazar analizi verileri incelendiğinde potansiyelinin artarak devam ettiği görülmüştür [9]. Şirketler aşağıda verilen alanlarda yoğun olarak tahmine dayalı analitik uygulamalarını kullanmaktadırlar [10].

- Potansiyel yeni pazarların keşfedilmesi
- Risklerin işlenmesi ve azaltılması
- Özelleştirilmiş ürünler ve hizmetler sunulabilmesi
- Kalite Kontrol

Tahmine dayalı analitik uygulamalarının başlıca kullanıldığı sektörler ve örnek uygulamalar şöyle sıralanabilir:

- Perakende sektöründe kullanıcıların işlem geçmişi verilerinden faydalanarak kullanıcıları gruplama, özel kampanya oluşturma ve önceki satın almalarından faydalanarak satın alması yüksek olasılıklı ürünler önerme gibi uygulamalar [11].
- Tıp alanında toplanmış vaka verilerinden eğitilmiş modeller yardımı ile hastalık teşhisinde karar destekleyici uygulamalar [12].
- Nesnelerin İnterneti uygulamalarında uçbirimlerden toplanmış geçmişe dönük büyük verilerden faydalanarak benzer veya farklı alanlarda tahminler elde etmek. Örneğin belirli bir bölgedeki trafik verilerinin izlenip, toplanan veriden gerçek zamanlı trafik tahmini yapılması örnek uygulamalardan biridir [13].

TDA uygulamaları geliştirilirken uzun bir veri madenciliği ve analiz safhası ve devamında bir tahminci modelin eğitilmesi ve değerlendirilmesi aşamalarından geçilir. TDA alanında bu süreçleri kolaylaştıracak ve/veya hızlandıracak çok sayıda güçlü

araç, yazılım bulunabilmektedir. Fakat konu, bu modellerin gerçek bir problemi çözmek üzere mevcut bir sisteme entegre olmasına geldiğinde, bu entegrasyon süreci mevcut bilgi teknolojileri altyapıları ile çok zorlu olabilmekte, aylarca sürebilmektedir [14]. Entegre olacağı ortamın türüne göre bazen modellerin o ortamda yeniden eğitilmesi gereği dahi doğabilmekte ve bu durum projelerin maliyet hesaplarında önemli sapmalara sebep olabilmektedir.

Bu tezin amacı, eğitilmiş modellerin bilgi teknolojileri altyapısına olan bağımlılığını azaltarak, genel kabul görmüş standartların kullanımı ile birlikte entegrasyon süreçlerinin hızlandırılmasına destek olacak bir çalışma gerçekleştirmektir. Gerçekleştirilen çalışma, eğitilmiş TDA modellerini hiper metin transfer protokolü (Hyper Text Transfer Protocol: HTTP) üzerinden tahmin edilebilir hale getirmekte, bunu yaparken de tahmin hızını olabildiğince az etkilemeyi hedeflemektedir. HTTP sayesinde bu protokolü kullanabilen tüm cihazlar ve ortamlar modelin öngörüleme işlemini gerçekleştirebilecek ve kolayca entegre olabileceklerdir. Ayrıca çalışma, kuruluşların darboğaz yaşadığı bu süreci kolaylaştıracak ve modelin entegre olabilme kapasitesini artıracaktır. Böylelikle makine öğrenmesi uygulamalarına daha çeşitli uygulama alanı sağlanabilecektir. Çalışmanın çıktıları ham veriden öngörü elde edilmesini kolaylaştırması ve standartlaştırması açısından önemlidir.

1.2 Tezin Literatüre Katkısı

Tahmine dayalı analitik çalışmaları literatürde zengin bir geçmişe sahiptir. Tahmine dayalı analitik modellerle yapılan çalışmalarda modelin eğitilmesi kadar yayımlanması da önemlidir. Özellikle pratikte kullanılması planlanan modellerin kullanılacağı ortama entegre edilmesi zorlu bir süreçtir ve bu işlemin en etkili yolunu arayan çalışmalar mevcuttur [15,16]. Modellerin eğitim amacına ve uygulanacağı probleme göre farklı yayınlama teknikleri ile karşılaşılmaktadır. Literatürde bu çalışmalara benzer birçok örnek bulunabilmektedir. Farklı bir yaklaşım olarak, TDA modellerinin genişletilebilir işaretleme dili (Extensible Markup Language : XML) ile temsilinin sağlanması amacıyla yapılan çalışmalar Öngörülü Model İşaretleme Dili (Predictive Model Markup Language : PMML)'i önermişlerdir [17]. Grossman'ın PMML kavramını ortaya koyduğu çalışma ile birlikte bu konuda yapılan çalışmaların sayısı artmıştır. Günümüze kadar birçok araştırmacı PMML yardımı ile serileştirilmiş modellerin başka platformlara entegrasyonunu kolaylaştırıcı çok sayıda çalışma

yapmıştır. Fakat bunların büyük çoğunluğu sadece mevcuttaki uygulamanın model yayınlama safhasını kolaylaştırmakta veya mevcut uygulamalara TDA modellerini entegre etmektedir. Herhangi bir modelin herhangi bir ortama kolayca entegre olabilmesine yönelik bir yayınlama yöntemi çok az sayıda araştırmacı tarafından ele alınmıştır. Yapılan başka bir çalışmada [18] PMML ile serileştirilmiş TDA modellerinin bir web servisi üstünden tahmin edilebilir olması önerilmiştir. Bulut sistemlerinde çalışacak şekilde tasarlanmış olan yöntem web haberleşmesinde servis odaklı mimari yaklaşımını tercih etmiştir. Bu çalışmada Guazzeli'nin önerisinden farklı olarak, eğitilmiş modeller bir arayüz üzerinden, hizmet olarak yazılım (Software as a Service: SaaS) yaklaşımı ile yönetilebilecek, oluşturulan uçbirimler *RESTFul* yaklaşımlı bir Web uygulama programlama arayüzü (Application Programming Interface: API) üzerinden çağırılabilecektir. Yapılan çalışmalarda servis odaklı mimari yaklaşımına kıyasla daha performanslı olduğu kanıtlanmış [19] olan *RESTFul* yaklaşımının kullanılması modellerin daha yaygın şekilde kullanıma açılabilmesine olanak sağlayacaktır. Mobil uygulamalar, tek sayfa uygulamaları gibi hızın önemli rol oynadığı, yenilikçi teknolojiler düşünüldüğünde daha az ek yük (overhead) sunan *REST* tercih edilmektedir. Ayrıca modellerin versiyonlanabilmesi, çok kullanıcılı yapı sayesinde birden fazla kullanıcıya hizmet verilebilmesi gibi ek çalışmaların tahmin performansına olan etkisi de ölçülmüş ve literatüre sunulmuştur.

1.3 Tezin Organizasyonu

Bu çalışma 8 bölüme ayrılmıştır.

Giriş bölümünde tez çalışmasının motivasyonu, amacı, literatüre katkısı ve organizasyonu verilmiştir.

İkinci bölümde literatürdeki öngörülü model işaretleme dili çalışmaları incelenmiş, bu çalışmaların konularına ait kısaca bilgiler sunulmuştur.

Üçüncü bölümde tahmine dayalı analitik ve öngörülü model işaretleme dili ile ilgili genel bilgiler verilmiş ve öngörülü model işaretleme dili alt maddeleri ile detaylandırılmıştır. Dilin ortaya çıkışından bu yana gelişimi, yapısı ve mevcut teknolojilerdeki yaygınlık durumu hakkında bilgiler verilmiştir.

Dördüncü bölümde materyal ve yöntem hakkında detaylı bilgi verilmiştir. Eğitilmiş makine öğrenmesi modellerinin öngörülü model işaretleme dili ile web servis olarak

servis edilmesi amacıyla geliştirilen uygulamalardan bahsedilmiştir. Bu uygulamalar geliştirilirken kullanılan mimari tasarım verilmiş ve faydalanılan teknolojiler detaylandırılmıştır.

Beşinci bölümde materyal ve metot bölümünde önerilen ve geliştirilen uygulamanın yayımlanması süreci ile ilgili bilgiler verilmiştir. İlişkili teknolojiler açıklanmıştır.

Altıncı bölümde geliştirilen uygulamaların kullanılabilmesi için örnek veri setlerinin belirlenmesi aşaması anlatılmış, seçilen veri setleri açıklanmıştır. Bu veri setlerinin öngörülü model işaretleme dili dokümanına dönüştürülmesi aşaması ve tercih edilen makine öğrenmesi algoritmaları hakkında bilgi verilmiştir.

Yedinci bölümde gerçekleştirilen web servisi uygulamasının performansı ile ilgili ölçümler gerçekleştirilmiş, ölçütler hakkında bilgiler verilmiştir.

Sekizinci bölümde elde edilen uygulama ve bu uygulama üzerinde gerçekleştirilen performans ölçümleri genel olarak değerlendirilmiş, böyle bir sistemin kullanılmasındaki avantaj ve dezavantajlardan bahsedilmiş ve öneriler sunulmuştur.

2. KAYNAK ARAŞTIRMASI

Tahmine dayalı analitik modellerin diğer uygulamalarla PMML sayesinde kolayca entegre olabilmesi diğer bir deyişle bu modellerin yayımlanabilmesi, bu alanda yapılan çalışmaların önünü açmış ve yapılan çalışma sayısında artışa sebep olmuştur. [17] Bir çalışmada tahmine dayalı analitik modellerin XML ile temsilini sağlayan PMML dilinin erken versiyonu yardımcı örneklerle tanıtılmıştır. PMML ve UIMA (Unstructured Information Management Architecture) kullanılarak analitik uygulamalar geliştirmeyi ve yayınlamayı açıklayan başka bir çalışmada, yapılandırılmamış verinin yapılandırılması aşamasında UIMA, yapılandırılmış verinin sınıflandırılmasında da sınıflandırma modelinin temsili olan PMML Sınıflandırıcı kullanılmıştır [20]. Bir başka çalışmada ise PMML tabanlı bir skorlama motoru tasarlanmış ve işleyişi açıklanmıştır [21]. Python ile geliştirilen sistem diğer programlama dilleri ile haberleşebilecek şekilde tasarlandığı vurgulanmıştır. PMML'in genel yapısı üzerine yapılan iki çalışmada [22,23] PMML'in genel yapısını

ve 2009 yılında duyurulan 4.0 versiyonuyla birlikte getirilen yenilikleri açıklamışlardır. [18] PMML modellerinin servis odaklı mimari ile geliştirilmiş web servisleri aracılığı ile yayımlanması amacıyla bir bulut bilişim uygulaması önermişlerdir. [24] PMML modellerinin veritabanlarına ve veri ambarlarına entegre edilerek çok yüksek hacimli verilerin tahmine dayalı analitik modeller ile skorlanmasını açıklamıştır. Bu çalışmayla modellerin serileştirildiğinde kolaylıkla ve yüksek performansla başka platformlara entegre olabildiği vurgulanmıştır. Çalışma EMC Greenplum veri ambarı üzerinde yapılmışsa da çalışmada atıf yapılan evrensel PMML (Zementis Universal PMML Plugin) eklentisi ile diğer veritabanları ile de entegrasyon sağlanabildiği vurgulanmıştır. [25] PMML ile temsil edilen tahmine dayalı analitik modellerin, kural motorlarına entegre edilmesi ile kural motorlarının kabiliyetinin genişletilmesi için bir sistem önermişlerdir. [26] PMML'in eski versiyonlarının PMML 4.0'a dönüştürülebilmesini sağlayan bir PMML Dönüştürücü önerilmiştir. Bu çalışmada PMML'in birlike-çalışabilirlik (interoperability) özelliğinin korunması amaçlanmıştır.

3. ÖNGÖRÜLÜ MODEL İŞARETLEME DİLİ

3.1 Tahmine Dayalı Analitik

Tahmine dayalı analitik, gelecekteki olayları ve davranışları tahmin etmek için geçmiş veriden faydalanılmasına dayalı bir istatistik yöntemidir. Tahmine dayalı analitiğin istatistiksel teknikleri veri modelleme, makine öğrenmesi, yapay zeka, derin öğrenme ve veri madenciliği tekniklerini kapsamaktadır. Tahmine dayalı analitiğin temelinde bağımlı ve bağımsız değişkenler arasında bir ilişki, bir örüntü tespit etmek ve bunlardan faydalanarak bilinmeyen sonucu tahmin etmek yatmaktadır. Sonuçların tahmin edilmesinde elde edilen doğruluk, ilişkilerin belirlenmesi aşamasının yani veri analizi aşamasının ne kadar iyi yapıldığı ile ilişkili olarak değişmektedir. Tahmine Dayalı Analitik modellerinin birçoğu hızlı çalışabilmekte ve çalışma zamanında hesaplama bir diğer adıyla tahmin yapabilmektedirler. Bu özellikleri sayesinde bankacılık uygulamaları, kredi risk hesaplamaları, sahtecilik denetimi gibi hızlı yanıt bekleyen uygulamalara adapte edilebilmektedirler [27].

Tahmine dayalı analitik modelleri 5 temel başlık altında toplanabilirler:

- *Sınıflandırma Modeli*: Verileri sınıflara ayırma mantığına dayanır. Verilen parametrelerin hangi sınıfa dahil olduğu bilgisini tahmin eder.
- *Kümeleme Modeli*: Benzer özellikteki verilerin tespitini ve kümelemesini gerçekleştirir.
- *Tahmin Modeli*: Sayısal değerlerle eğitilmiş ve sayısal tahminlerin beklendiği modellerdir. Örneğin geçmiş satış verisinden eğitilmiş bir model ile gelecek dönemde ne kadar ürün satılacağı bilgisinin tahmin edilmesi tahmin modeli sayesinde gerçekleştirilmektedir.
- *Aykırı Değerler Modeli*: Aykırı değerler üzerinden çalışan model türüdür. Bankacılık sektöründe, müşterinin harcama alışkanlığının dışında bir harcama işlemi gerçekleştirdiğinde bunun tespitinin sağlanmasında kullanılmaktadır.
- *Zaman Serisi Modeli*: Zamanı dayalı bir veri grubunu baz alan modeldir. Belirli bir zaman diliminden eğitilen bu model sayesinde daha geniş zaman dilimleri için tahminde bulunmak mümkündür.

3.2 Öngörülü Model İşaretleme Dili Tanımı ve Tarihsel Gelişimi

Öngörülü model işaretleme dili (PMML), Data Mining Group isimli konsorsiyum tarafından geliştirilmiş, veri madenciliği modellerinin öngörülü model işaretleme dili uyumlu uygulamalar arasında paylaşılabilmesine olanak sağlayan genişletilebilir işaretleme dili (XML) tabanlı bir dildir. PMML, modelleri geliştiriciden bağımsız bir şekilde temsil edebilmektedir. Bu sayede kullanıcılar bir veri madenciliği uygulamasında eğittikleri modelleri başka bir geliştiricinin başka bir veri işleme (görselleştirme, öngörüleme vb) uygulamasına taşıyabilmektedirler. Ayrıca bu temsil yöntemi sayesinde model eğitimlerinin uzun sürdüğü veri setlerinde model bir kez eğitilip öngörülü model işaretleme dili ile saklanarak sonraki oturumlarda tekrar eğitmeye gerek kalmadan kullanılabilir.

Öngörülü model işaretleme dili ile dönüştürülmüş bir model tahmin işlemine hazır haldedir. Dönüştürülmüş modeli temsil eden PMML dosyası, bu dosyayı yükleyecek ve tahmin işlemini gerçekleştirecek icracı bir uygulama yardımı ile tahmin işlemi

yapabilir bir model haline gelmektedir. İcracı uygulamalar öngörülü model işaretleme dili dosyalarını yorumlayabilen herhangi bir geliştirme ortamında geliştirilebilir.

Öngörülü model işaretleme dili, ilk olarak 1997 yılında ortaya çıkmıştır. İlk ortaya çıkışından bu yana sürekli geliştirilmeye devam etmiş, birçok majör versiyonu yayımlanmış, kendini ispat ederek olgunlaşma evresini tamamlamıştır. Bu kararlı gelişim, tahmine dayalı analitik ekosisteminde uygulamalar, uygulama geliştiricileri, geliştirme birimleri arasında ortak dil (*lingua franca*) olarak kabul edilmesini sağlamıştır. Bahsedilen paydaşların ortak bir dilde buluşabilmesi, öngörülü modellerin yayımlanmasına kadar olan süreçte maliyet azaltılmasına yardımcı olmuş, sürecin karmaşıklığını azaltmıştır [14].

Öngörülü Model İşaretleme Dili, her majör versiyon geçişinde önemli bir özelliği bünyesine katmış veya mevcut bir özelliği iyileştirmiştir. Bunun yanında temsil edebildiği model sayısı ilk versiyonlarında kısıtlı iken, yeni versiyonlarda yeni modeller eklenerek bu yelpaze genişletilmektedir. Ortalama her üç yılda bir güncellenen dilin en güncel versiyonu 4.4'tür ve 2019 yılının Nisan ayında yayımlanmıştır [14,28,29].

3.3 Öngörülü Model İşaretleme Dilinin Yapısı

Bir öngörülü model işaretleme dili dosyasının içeriği incelendiğinde tüm dosyanın aslında XML kodlarından oluştuğu görülebilir. PMML, bir veya birden fazla modelin XML kodlarıyla ifade edilebildiği bir yapıya sahiptir. Yazım kuralları XML kurallarıyla aynıdır ve aynı doğrulama sürecinden geçer. En güncel versiyonu ile oluşturulmuş, en sade PMML yapısı takip eden şekilde görülebilir (Şekil 3.1).

```
<?xml version="1.0"?>
<PMML version="4.4"
  xmlns="c-4_4"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <Header copyright="ersinyildiz.com"/>
  <DataDictionary> ... </DataDictionary>

  ... model ...

</PMML>
```

Şekil 3.1 : En temel PMML doküman yapısı

Her bir PMML dosyası, dört ana bölümden oluşur. Bu ana bölümler aşağıda sıralanmıştır.

- Başlık (*Header*)
- Veri Sözlüğü (*Data Dictionary*)
- Veri Dönüşümleri (*Data Transformation*)
- Model

Bu bölümde Iris veri seti [30,31] kullanılarak, Knime uygulamasında üretilen bir PMML dosyası üzerinden, PMML'in dört ana bölümü örneklerle anlatılmıştır.

3.3.1 Başlık

Başlık, PMML dosyasının genel bilgilerini ihtiva eder. Bu bilgiler, telif hakkı bilgisi, açıklama, modelin eğitildiği uygulama ve uygulamanın versiyon bilgisi, modelin üretildiği zaman bilgisi olarak çeşitlenmektedir. Bu bilgiler PMML'i okuyacak uygulamalar tarafından kullanılabilmektedir. Şekil 3.2'de Iris veri seti ile oluşturulmuş bir PMML dokümanının başlık bölümü gösterilmektedir.

```
<Header copyright="eryildiz" description="örnek açıklama">
  <Application name="KNIME" version="4.2.2"/>
  <Timestamp>2021-05-09</Timestamp>
</Header>
```

Şekil 3.2 : Örnek bir PMML doküman başlığı

3.3.2 Veri sözlüğü

Veri sözlüğü bölümü, model eğitilirken kullanılan her bir giriş değişkeninin (*DataField*) tanımını içeren bölümdür. Bu giriş değişkenleri, sürekli, kategorik veya sıralı türde olabilmektedir. Değerin hangi türde olduğu *optype* özelliği ile belirtilir. String, double gibi, girdinin veri türü ise *dataType* özelliği ile belirtilir. Decision Tree algoritması ile eğitilen Iris veri seti modeline ait veri sözlüğü aşağıdaki şekilde incelenebilir (Şekil 3.3).

```
  <DataDictionary numberOfFields="5">
    <DataField name="SepalLengthCm" optype="continuous"
dataType="double">
      <Interval closure="closedClosed" leftMargin="4.3"
rightMargin="7.9"/>
    </DataField>
    <DataField name="SepalWidthCm" optype="continuous"
dataType="double">
      <Interval closure="closedClosed" leftMargin="2.0"
rightMargin="4.4"/>
    </DataField>
    <DataField name="PetalLengthCm" optype="continuous"
dataType="double">
      <Interval closure="closedClosed" leftMargin="1.0"
rightMargin="6.9"/>
    </DataField>
    <DataField name="PetalWidthCm" optype="continuous"
dataType="double">
       <Interval closure="closedClosed" leftMargin="0.1"
rightMargin="2.5"/>
    </DataField>
    <DataField name="Species" optype="categorical"
dataType="string">
      <Value value="Iris-setosa"/>
      <Value value="Iris-versicolor"/>
      <Value value="Iris-virginica"/>
    </DataField>
```

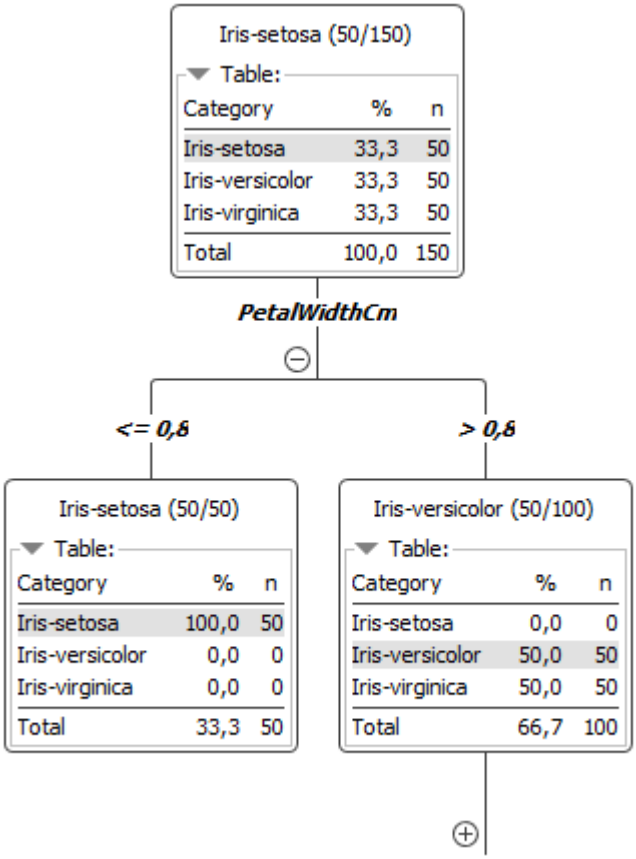
Şekil 3.3 : Iris veri seti için örnek bir veri tablosu

3.3.3 Veri dönüştürmeleri

Veri dönüştürme işlemleri girdi verilerine uygulanır. Bir girdi verisine dönüştürme işlemi uygulanır ve sonucu yeni bir girdi olarak kabul edilirse bu ifadeler türetilmiş alan (*DerivedField*) olarak veri sözlüğü içinde yer alır. Bu alanda verinin türü, hangi fonksiyon ve parametrelerle üretileceği bilgisi yer alır.

3.3.4 Model

Model bölümü, yapay sinir ağı, regresyon, karar ağacı gibi öngörülü modellerin tanımlamalarının yapıldığı bölümdür. Her algoritmaya göre modelin ifade edilmesi için kullanılan özellikler değişmektedir. Knime uygulamasında eğitilmiş olan karar ağacı modelinin düğümleri takip eden şekilden incelenebilir (Şekil 3.4).



Şekil 3.4 Knime uygulamasında oluşturulmuş karar ağacı modelinin ön izlemesi

Aynı karar ağacı modelinin PMML temsili için aşağıdaki özellikler oluşturulmuştur. Sayıları fazla olduğu için tüm düğümler gösterilmemiştir. Modelin PMML örneği ise Şekil 3.5'de incelenebilir.

- Modelin Adı (*modelName*)
- Fonksiyon (*functionName*)
- Bölme Karakteristiği (*splitCharacteristic*)
- Eksik değer stratejisi (*missingValueStrategy*)

```
  <TreeModel modelName="DecisionTree" functionName="classification"
splitCharacteristic="binarySplit"
missingValueStrategy="lastPrediction"
noTrueChildStrategy="returnNullPrediction">
    <MiningSchema>
      <MiningField name="SepalLengthCm"
invalidValueTreatment="asIs"/>
      <MiningField name="SepalWidthCm" invalidValueTreatment="asIs"/>
      <MiningField name="PetalLengthCm"
invalidValueTreatment="asIs"/>
      <MiningField name="PetalWidthCm" invalidValueTreatment="asIs"/>
      <MiningField name="floorSepalLength"
invalidValueTreatment="asIs"/>
      <MiningField name="Species" invalidValueTreatment="asIs"
usageType="target"/>
    </MiningSchema>
    <Node id="0" score="Iris-setosa" recordCount="150.0">
      <True/>
      <ScoreDistribution value="Iris-setosa" recordCount="50.0"/>
      <ScoreDistribution value="Iris-versicolor" recordCount="50.0"/>
      <ScoreDistribution value="Iris-virginica" recordCount="50.0"/>
      <Node id="1" score="Iris-setosa" recordCount="50.0">
        <SimplePredicate field="PetalWidthCm" operator="lessOrEqual"
value="0.8"/>
        <ScoreDistribution value="Iris-setosa" recordCount="50.0"/>
        <ScoreDistribution value="Iris-versicolor"
recordCount="0.0"/>
        <ScoreDistribution value="Iris-virginica" recordCount="0.0"/>
      </Node>
      <Node id="2" score="Iris-versicolor" recordCount="100.0">
        <SimplePredicate field="PetalWidthCm" operator="greaterThan"
value="0.8"/>
        <ScoreDistribution value="Iris-setosa" recordCount="0.0"/>
        <ScoreDistribution value="Iris-versicolor"
recordCount="50.0"/>
        <ScoreDistribution value="Iris-virginica"
recordCount="50.0"/>
        <Node id="3" score="Iris-versicolor" recordCount="54.0">
          ... (Detaylar Gizlenmiştir)
        </Node>
        <Node id="10" score="Iris-virginica" recordCount="46.0">
          ... (Detaylar Gizlenmiştir)
        </Node>
      </Node>
    </Node>
  </TreeModel>
```

Şekil 3.5 : Knime ile üretilen karar ağacı modelinin PMML temsili

3.3.5 Madencilik şeması

Modelde kullanılan tüm girdi alanlarının listelendiği bölümdür. Veri sözlüğü bölümünün bir alt kümesi olarak da kabul edilebilir. Girdi alanları bu bölümde madencilik alanı (Mining Field) olarak ifade edilir ve aşağıdaki özellikleri ile kullanılırlar.

- Adı (*name*): Veri sözlüğündeki girdi alanını gösterecek şekilde aynı isimle kullanılır.
- Kullanım amacı (*usageType*): Bu girdinin ne amaçla kullanıldığını belirtir. Örneğin bu veri alanı veri setinde tahmin edilmek istenen sınıfı belirtiyor ise bu özellik "target" değeri alır.
- Aykırılar (*outliers*): Aykırı verilerin hangi iyileştirmeye tabi tutulacağını belirtir.
- Düşük değer, yüksek değer (*lowValue, highValue*): Aykırılar özelliği ile aykırı kabul edilecek sınırları belirlemek için kullanılır.
- Eksik veri yerine geçecek veri (*missingValueReplacement*): Girdi türlerinde bir değer eksik olduğunda onun hangi değer ile değiştirileceği bu özellikle belirtilebilir.
- Eksik veri iyileştirme (*missingValueTreatment*): Eksik verinin yerine geçecek yeni verinin nasıl türetileceği (ortalama, medyan, özel değer) bu özellikle belirtilir.
- Geçersiz veri iyileştirme (*invalidValueTreatment*): Geçersiz değerlerin nasıl yönetileceğini belirleyen özelliktir.

3.3.6 Hedefler

Hedef (*Target*) özelliği, sınıflandırma veya regresyon ile oluşturulmuş modellerin öngörü değerlerini manipüle etmek amacıyla kullanılan bir özelliktir. Örneğin bir regresyon modeli için hedef kategorilerini belirli sayısal değerlere denk düşürmek için Şekil 3.6'da görülebilecek şekilde tanımlamalar kullanılır. Başka bir örnek olarak, eğer regresyon modeli, sadece tam sayılar kümesinden değerler tahmin etmesi gerekiyorsa, gerçekleşen değer tam sayıya dönüştürülebilir. Bu işlem de castInteger özelliği ile gerçekleştirilir. Toparlanacak olursa, hedef özelliği tahmin işleminden sonra ortaya

çıkan değerler üzerinde değişiklik, düzenleme gerektiği durumlarda kullanılabilecek bir özelliktir [14,32].

```
<Targets>
  <Target field="response" optype="categorical">
    <TargetValue value="YES" displayValue="Yes"
priorProbability="0.02"/>
    <TargetValue value="NO" displayValue="No"
priorProbability="0.98"/>
  </Target>

  <!-- alternative for continuous field -->
  <Target field="amount" optype="continuous">
    <TargetValue defaultValue="432.21"/>
  </Target>
</Targets>
```

Şekil 3.6 : Örnek bir PMML hedef bölümü

3.3.7 Çıktılar

PMML modeli üzerinde bir tahmin işlemi yürütüldüğünde alınan yanıtlar olarak özetlenebilir. Şekil 3.7'de bir örneğini görebileceğiniz bu yapı öngörü işleminin sonucu için bir özet niteliğindedir ve bu sonuç elde edilirken hangi değerlerin oluştuğu bilgisi de bu özet içinde verilir.

```
<Output>
<OutputField name="Class" feature="predictedValue"
optype="categorical" dataType="string"/>

<OutputField name="Probability_Setosa" optype="continuous"
dataType="double" feature="probability" value="Iris-setosa"/>

<OutputField name="Probability_Versicolor" optype="continuous"
dataType="double" feature="probability" value="Iris-
versicolor"/>

<OutputField name="Probability_Virginica" optype="continuous"
dataType="double" feature="probability" value="Iris-
virginica"/>
</Output>
```

Şekil 3.7 : Örnek bir PMML çıktı ifadesi

Ayrıca yine Şekil 3.7'de görüleceği üzere ilk OutputField değeri tahminin sonucunu vermektedir. Bu "*feature*" değerinin "*predictedValue*" oluşundan anlaşılmaktadır. Geri kalan üç OutputField değerinin ise diğer olasılıklar olduğu yine bu yapıya bakılarak anlaşılabilmektedir. Geliştirilen sistemde bir tahminin sonucu buradaki yapı gözetilerek oluşturulmuştur.

3.4 PMML Uyumluluk Raporu

Öngörülü Model İşaretleme Dili uygulama geliştiricilerince kabul edilen bir açık standart olmuştur ve desteklendiği veri madenciliği uygulamalarının sayısı gün geçtikçe artmaktadır. Çizelge 3.1'de dilin desteklendiği yazılımların en bilinen 5 tanesi verilmiştir. Mevcut durumda ise 36 farklı şirketin 77 farklı ürününde PMML desteklenmektedir [33]. Bunlara ek olarak PMML'in desteklendiği programlama dilleri (R, Python, Scala, Java, C#) sayesinde kullanım alanı da genişlemeye müsaittir.

Çizelge 3.1: PMML destekli uygulamaların başlıcaları

Şirket / Proje	Ürünler
IBM	IBM SPSS Statistics, IBM SPSS Modeler, IBM InfoSphere Warehouse
KNIME	Knime 2.1
Spark	Apache Spark
Weka	Weka
FICO	Model Builder, Decision Optimizer, Blaze Advisor, Strategy Director

PMML geliştirilmeye devam etmekte ve ortalama her 3 yılda bir yeni majör bir sürümü çıkmaktadır. Her sürüm güncellemesinde PMML'in desteklediği algoritmalar da çeşitlenmekte ve sayısı artmaktadır. PMML 2.1 versiyonunda bu algoritmaların sayısı 8 iken en güncel versiyonu olan PMML 4.4.1 'de bu sayı 18'e çıkmıştır. En güncel versiyonda desteklenmekte olan algoritmalar ve modeller şunlardır [34]:

- Anormallik Tespit Modeli (Anomaly Detection Models)
- Birliktelik Analizi (Association Rules)
- Baseline Model
- Bayes Ağı (Bayes Network)
- Kümeleme Modelleri (Cluster Models)
- Gaussian
- Genel Regresyon
- K-En Yakın Komşuluk (K-Nearest Neighbors)

- Naive Bayes
- Yapay Sinir Ağı (Neural Network)
- Regresyon
- Kural seti (Ruleset)
- Skor Kart (Scorecard)
- Diziler (Sequences)
- Metin modelleri (Text Models)
- Zaman Serileri
- Ağaçlar
- Vektör makineleri

4. MATERYAL ve YÖNTEM

4.1 Uygulamanın Mimari Tasarımı

Uygulamanın mimari tasarımı yapılırken istemcilerin kimler olabileceği sorusundan yola çıkılmıştır. Öngörülü modeller hayatın hemen her alanında kendine kullanım alanı yaratabildiği için uygulamanın geniş bir istemci yelpazesi olabilir. Mobil uygulamalar, web uygulamaları, masaüstü uygulamalar, akıllı sistemler vb. gibi istemci türleri örnek verilebilir. Bu istemcilerin hepsine standart bir şekilde hizmet verebilecek bir sistem tasarlanmak istendiği için hemen hepsine tek bir standart ile hizmet verebilmeyi mümkün kılan Hiper-Metin Transfer Protokolü (HTTP) tercih edilmiştir. Günümüzde en basit Arduino kartlarından en performanslı bulut sistemlerine kadar geniş bir yelpazede HTTP desteklenmektedir.

Uygulamanın kapsamını olabildiğince geniş tutmak hedeflenmiştir. Bunun için hem düşük performanslı ortamlarda (kart bilgisayarlar vb.) hem de yüksek performans sunan ortamlarda (özel bulut sistemleri vb) çalışabilecek şekilde tasarlanmıştır. Bunu

yaparken farklı ortamlarda yayımlanma safhasında gerçekleştirilecek adaptasyon çalışmalarının asgari düzeyde olmasına dikkat edilmiştir.

Yukarıda anlatılan kriterler neticesinde HTTP ile hizmet veren, *RESTFul* mimaride tasarlanmış bir Web API uygulamasının geliştirilmesine karar verilmiştir. Web API gerektiğinde önyüzden herhangi bir yönetim işlemine gerek duymadan kendi başına tüm işlevlerini yerine getirebilecek şekilde tasarlanmıştır. Sistem özellikleri kısıtlı ortamlar için bu durum avantaj sağlayacaktır. Bunun yanında ayrı bir web uygulaması olarak geliştirilen yönetim önyüz istemcisi sayesinde de performanslı bulut sistemlerinde yayımlanarak çok kullanıcılı, hizmet olarak yazılım (SaaS) mantığında hizmet veren bir uygulama da geliştirilmiştir. Bu önyüz uygulaması sayesinde sistem herkese açılarak, kişilerin üye olabildiği, modellerini yönetebildiği ve tahmin uçbirimleri oluşturabildiği bir hizmet verilmesi planlanmaktadır. Uygulamanın tasarımı etkileyen, öne çıkan özellikleri, kullanılabilirlik özellikleri ve teknik özellikler başlıklarında şöyle sıralanabilir:

Kullanılabilirlik Özellikleri:

- Birbirinden izole şekilde, çok kullanıcılıdır
- Kullanıcılar proje oluşturabilir, yönetebilir.
- Her proje için birden fazla uçbirim bağlanabilir ve yönetilebilir.
- Her uçbirime modeller bağlanabilir ve yönetilebilir

Teknik Özellikler:

- Veri tabanı türüne az bağımlıdır
- Farklı işletim sistemlerini desteklemektedir
- Web altyapısının düşük sistem özelliklerinde de sorunsuz çalışabilecek şekilde hafif yapıdadır
- Yönetim arayüzünde insan bilgisayar etkileşimini yüksek tutacak şekilde tasarımlar yapılmıştır.

4.1.1 İstemciler

Uygulama, ağ üzerinden hizmet vereceği için HTTP ile haberleşebilen herhangi bir uygulamadan, herhangi bir cihazdan erişilebilir durumdadır. Bunlardan en yaygın olanlar ve bunların olası kullanım senaryoları takip eden başlıklarla sıralanmıştır.

4.1.1.1 Mobil uygulama istemcileri

Mobil uygulamalarda kullanıcıların yaptığı işlemler sırasında bir öngörülü modelden faydalanılabilir. Örneğin, kullanıcının hangi uygulamada ne kadar vakit geçirdiği bilgisi bu bilgilerden eğitilmiş bir modele verilerek kullanıcının belirli bir kümeye dahil edilmesi işlemi gerçekleştirilebilir. Böylelikle o kümeye özel dijital sağlık önerileri kullanıcıya doğru önerilebilir.

4.1.1.2 Web uygulamaları

Mobil uygulamalardaki davranışa benzer şekilde web uygulamalarında gerçekleştirilen işlemlere öngörülü modeller de entegre edilebilir. En bilinen örneklerden biri birliktelik analizi ile eğitilmiş model yardımıyla internet alışverişi kullanıcısına alması muhtemel bir ürün önermektir.

4.1.1.3 Endüstriyel uygulamalar

Endüstriyel cihazlardan alınan bilgilerin bir öngörüde kullanılması ve bunun gerçek zamanlı yapılması istendiği senaryolarda kullanılabilir. Önceki bordrolama verilerinden eğitilmiş modeller yardımıyla yeni üretilen bordroların hata ihtimallerinin anlık ölçülmesi ve hatalı bordrolama işlemlerinin engellenmesi örnek uygulama olarak verilebilir.

4.1.1.4 Akıllı sistemler

Akıllı sistemler öngörülü modeller kullanmaya yatkındır. Örneğin bir akıllı tarım uygulamasında geçmiş verilerden faydalanılarak eğitilmiş modeller sayesinde doğru gübreleme zamanının tahmin edilmesi uygulamasında modele erişim için bu uygulama kullanılabilir.

İstemciler daha fazla örnekle de çeşitlendirilebilir fakat temelde hepsi benzer amaçlarla kullanacaklardır. İletişim için ortak bir standart belirlemiş olmak tüm bu istemcilere yanıtlar fark etmeksizin hizmet verebilmeyi mümkün kılmıştır.

4.1.2 Arka uç uygulaması tasarımı

HTTP isteklerine yanıt verecek olan, Web API uygulaması kurulacak sistemin merkezinde konumlandırılmıştır ve sistemin çekirdeğini oluşturur. Uygulamanın hedeflediği çıktılar, istemcilerin türleri, uygulamanın çalışmasını beklediğimiz ortamlar gibi kriterler ele alındığında geliştirilmesi hızlı fakat yapısal olarak da hafif bir web çatısı ihtiyacı olduğu ortaya çıkmıştır. Bunun yanında, PMML modellerinin ayrıştırılabildiği ve tahmin işleminin yapılabildiği programlama dilleri araştırılmıştır. Bu araştırma yapılırken veri bilimi, makine öğrenmesi alanlarında hangi programlama dillerinin daha yoğun kullanıldığı bilgisine de başvurulmuştur. Bu araştırmalar sonucunda makine öğrenmesi alanında en çok tercih edilen dillerin başında Python programlama dilinin geldiği görülmüştür [35,36]. Buna ek olarak, PMML destekleyen az sayıda programlama dilinden biri Python'dır. Bu alanda oluşmuş olan geliştirici ekosisteminin, geliştirilen uygulamaya da katkısı olacağı gözetilerek programlama dili olarak Python tercih edilmiştir. Web uygulama çatısı olarak ise son yıllarda bu alanda en sık başvurulan çözümlerden [37,38] biri olan Flask tercih edilmiştir. Uygulamanın teknolojik incelemesi bölümünde bu teknolojilere daha detaylı değinilmiştir.

Web API uygulaması geliştirilirken tercih edilen mimari ise *RESTFul* olmuştur. Temsili Durum Transferi (Representational State Transfer: REST) yaklaşımı tercih edilirken veri madenciliği modellerinin hangi uçbirimler ile daha fazla entegre olma ihtiyacı duyacağı kriteri gözetilmiştir. Veri madenciliği modellerinin üretim sahalarında, sosyal medya uygulamalarında, web uygulamalarında, IoT sistemlerinde giderek daha fazla kullanıldığı ve bu alandaki ihtiyacın devam edeceği göz önünde bulundurulunca geliştirilecek Web API'nin sıklıkla mobil cihazlardan, kısıtlı donanım özellikleri ile çalışan IoT cihazlardan da kullanılabilmesi gerekecektir. Bu ihtiyaca göre mobil cihazlarda Basit Nesne Erişim Protokolü'ne (Simple Object Access Protocol: SOAP) göre daha performanslı çalışan ve daha az sistem kaynağı tüketen [39] REST yaklaşımı tercih edilmiştir.

4.1.2.1 Temsili durum transferi yaklaşımı

REST, Fielding tarafından geliştirilen [40] HTTP üzerinden iletişim kuran, yaygın kabul görmüş bir mimaridir. REST mimarilerde kaynaklar bir Tekdüzen Kaynak Bulucu (Uniform Resource Identifier: URI) ile temsil edilmekte ve bu kaynaklar ile yapılabilecek oluşturma, okuma, güncelleme ve silme (Create, Read, Update, Delete:

CRUD) işlemleri GET, PUT, POST, DELETE HTTP metotlarına iz düşürülmektedir. REST basit ve sade bir arayüzü olan Web servisler oluşturmayı mümkün kılar. Böylelikle sistem kaynağının kısıtlı olduğu, performansın ve hızın önemli olduğu senaryolarda tercih edilmektedir. Bu çalışmada da modellerin hızlı tahminlenmesi gerektiği durumlar için REST daha uygun bulunmuştur.

4.1.3 Önyüz uygulaması tasarımı

Geliştirilen Web API tüm fonksiyonalitesini herhangi bir önyüz çözümü gerektirmeksizin sunabilecek şekilde tasarlanmış olsa da özellikle hizmet olarak yazılım (SaaS) modelinde kullanıcıların bir yönetim arayüzüne ihtiyaçları bulunmaktadır. Bu arayüz sayesinde aşağıdaki işlemler son kullanıcılar tarafından kolaylıkla gerçekleştirilebilecektir.

- Kullanıcı İşlemleri (Kayıt, Giriş vb.)
- Proje Yönetimi
- Model Yönetimi
- Uçbirim Yönetimi

Kullanıcı arayüzü uygulaması geliştirmek için iki temel yaklaşım ve bu yaklaşımların kullanıldığı çok çeşitli teknolojiler ve yöntemler bulunmaktadır. Tek sayfalı uygulamalar (Single Page Application: SPA) ve çok sayfalı uygulamalar (Multi Page Application: MPA) olarak ikiye ayrılan bu yaklaşımlardan birini seçmek için bazı kriterler belirlenmiştir. Kriterler arasında, yenilikçilik, teknolojik olgunluk, kullanıcı deneyimi, REST API uyumu, performans, geliştirme hızı, kullanıcı ekosistemi gibi başlıklar değerlendirilmiştir. SPA ve MPA yaklaşımlarının birbirlerine karşı üstünlükleri ve eksiklikleri vardır. Fakat son dönemde oldukça popüler hale gelen, iyi bir kullanıcı deneyimi sunan, hızlı geliştirilebilen ve REST API'ler ile oldukça uyumlu çalışan, sunucu taraflı kod çalıştırma zorunluluğu olmayan SPA yaklaşımı bu proje için daha uygun görülmüştür [41].

4.1.3.1 Tek sayfa uygulama yaklaşımı

Tek-Sayfa uygulama, kullanıcıya gösterilecek sayfanın tamamen yenilenmeden sadece gerekli bölümünün dinamik değiştirilmesi mantığında çalışan bir web uygulaması türüdür. Bu tür uygulamaların ilk yüklenme aşamasında tüm Hiper Metin

İşaretleme Dili (Hyper Text Markup Language: HTML) ve Basamaklı Biçim Sayfaları (Cascading Style Sheets: CSS) kodlarının yüklemesi gerçekleştirilir. Oturumun devamında bu dosyalar tekrar yüklenmez. Bu durum ekran geçişlerinin çok daha hızlı olmasını sağlayarak iyi bir kullanıcı deneyimi yaratır. Bir ekrandan diğerine geçiş yapılacağı zaman önce yeni ekran için gerekli veri Javascript Genişletilebilir İşaretleme Dili Hiper Metin Transfer Protokolü İsteği (XMLHttpRequest: XHR) ile çekilir ve bu veri yardımıyla yeni ekran çizilerek mevcut ekrandaki görüntü tekrar oluşturulur. Bu yaklaşımda, tüm yönlendirme ve render işlemleri kullanıcıların tarayıcılarında Javascript ile gerçekleştirildiği için uygulama sunucusuna binen yük az olmaktadır.

Tek sayfa uygulamalar, bazı öne çıkan avantajları [41,42] sayesinde son yıllarda çok yaygın hale gelmişlerdir. Bu avantajlar sıralanacak olursa:

- *Sayfa Hızı*: SPA yaklaşımında bir istek sadece sayfanın ilgili kısmının yeniden oluşturulmasını gerçekleştirdiği için MPA yaklaşımına göre çok daha hızlı çalışmaktadır. MPA yaklaşımında her istekte sayfa en baştan sunucu tarafında oluşturulur ve kullanıcıya gösterilir.
- *Arka Uç Birime Gevşek Bağlılık*: SPA uygulamaları arka uç uygulamaları ile gevşek bağlıdır. İki ortam arasında sadece veri alışverişi gerçekleştirildiği için aynı veri yapısını sağlayacak başka bir uç birim ile çalışmak gerektiğinde hızlıca gerekli değişiklikler yapılabilmektedir.
- *Önbellek*: Sunucu taraflı önbellek işlemi sayfaların çok daha hızlı yanıt vermesini sağlamaktadır.

Avantajlarının yanında SPA yaklaşımının dezavantajları da mevcuttur. Arama motoru optimizasyonunun zorluğu, ölçeklenebilirlik ve güvenlik gibi konular dezavantaj yaratmakta fakat gelişen çatılar sayesinde bu konularda da iyileştirmeler gerçekleştirilmektedir.

4.1.4 Kaynak kod yönetimi

Günümüz yazılım geliştirme süreçlerinin olmazsa olmaz bir parçası olan kaynak kod yönetiminden bu çalışmada da faydalanılmıştır. Kaynak kod yönetimi, koddaki değişimlerin takip edilebildiği, ekiplerin aynı kaynak kod üstünde birlikte çalışmalarını kolaylaştıran bir süreçtir. Özellikle yaşayan projelerde kaynak kod

yönetiminin yapılması elzemdir. Kaynak kod yönetimi için farklı yönetim sistemi çözümleri mevcuttur. Git, Azure Devops, eski adıyla TFS, Subversion bunların başlıcalarıdır. Bu çalışma için, Pazar payı en yüksek [43,44] kullanıcı ekosistemi oldukça gelişmiş olan Git kaynak kod yönetim sistemi tercih edilmiştir.

4.1.4.1 Git kaynak kod yönetim sistemi

Git, dağıtık ve çevrimdışı çalışabilen, yazılım geliştirme süreçlerinde kullanılan bir sürüm kontrol ve kaynak kod yönetim aracıdır. Git, ilk olarak Linux çekirdeğinin geliştirilmesinde kullanılmak üzere 2005 yılında, Linus Torvalds tarafından geliştirilmiştir ve aktif olarak geliştirilmeye devam edilmektedir. Ücretsiz ve açık kaynak bir sistemdir [45]. Git'in en önemli avantajlarından bir tanesi kaynak kodda dallanma özelliğidir. Merkezi versiyon kontrol yönetim sistemlerindekine kıyasla çok daha kısa sürede, çok performanslı şekilde dal (branch) yönetimi yapılabilmektedir. Örneğin mevcut bir kaynak kodda yeni bir özellik geliştirileceği zaman yeni bir dal yaratmak, özelliği burada geliştirmek ve tamamlandığında ana dal ile birleştirmek en bilinen kullanım alışkanlıklarından biridir. Bu yaklaşım yeni özelliği geliştirirken yaşanabilecek sorunlardan ana daldaki kaynak kodun etkilenmemesini de sağlamaktadır [46]. Benzer yaklaşım canlı ortamda çalışan uygulamalarda hata çözme senaryosunda da kullanılmaktadır. Bu çalışmada da hem birçok deneysel geliştirme yapıldığı için, hem de kodun versiyonlanabilmesi için Git versiyon yönetim aracı kullanılmıştır.

4.2 Çalışmanın Teknolojik İncelemesi

Bu bölümde uygulama geliştirilirken, mimari bölümünde anlatılan özelliklerin hangi araçlar ile gerçeklendiği açıklanmıştır. Bu araçlar hakkında detaylı bilgi verilmiş, her aracın projeye olan katkısı ve projedeki rolü açıklanmıştır. Ayrıca kullanılan araçların alternatiflerine de değinilmiş, tercih sebepleri hakkında bilgi verilmiştir.

4.2.1 WEB API uygulaması

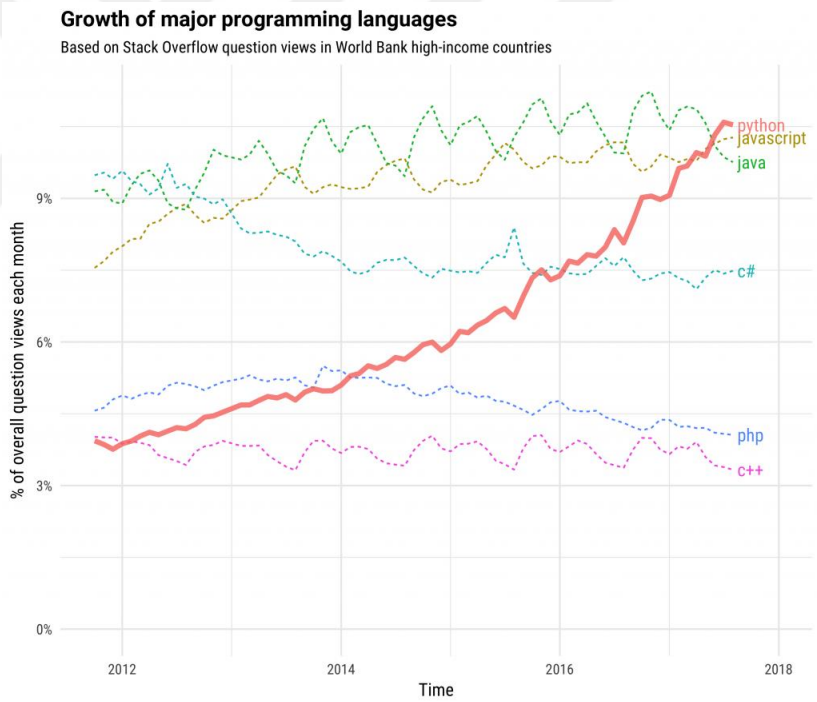
Web API uygulaması, bu çalışmanın temelini oluşturan, hedeflenen çıktıları sunabilecek en küçük yapıtaşıdır. Diğer çalışmalar kullanıcı deneyimini artıracak veya hızlandıracak yönde destekleyici çalışmalardır. Destekleyici uygulamalar, API uygulamasını kullanarak çeşitlenebilir, sayısı artabilir.

Web API uygulaması Python programlama dilinde, Flask web çatısı özelliklerinden faydalanılarak geliştirilmiştir.

4.2.1.1 Python programlama dili

Python, ilk olarak 1990 yılında Guido Van Rossum tarafından geliştirilmeye başlanmış, sonrasında Python Yazılım Vakfı tarafından geliştirilmeye devam etmiştir ve halen geliştirilmektedir. Python, yorumlanan bir dildir. Yani yazılmış olan bir kod makine diline dönüştürülmez ve bir Python yorumlayıcısı tarafından yorumlanarak icra edilir. Ayrıca Python, nesne yönelimli, modüler ve yüksek seviyeli bir programlama dilidir [47].

Python oldukça basit bir sözdizimine dayalıdır. Girintilere dayalı sözdizimi aynı zamanda kodun okunuşunu de kolaylaştırmakdır. Bu özellikleri itibariyle yazılıma yeni başlayan kullanıcılar arasında da oldukça kabul görmüştür.



Şekil 4.1 : Stackoverflow yazılım topluluğu sitesinde yıllara göre sorulan soruların dillere göre dağılım grafiği

Şekil 4.1'de görülebileceği gibi, Python dilinin tercih edilme oranı, oraya çıktığı ilk yıllara göre son 10 yılda ciddi oranda artış göstermiştir [48]. Bu artışın önemli sebeplerinden bir tanesi de yapay zeka profesyonellerinin en çok tercih ettiği dillerin başında gelmesi gösterilebilir. Python, bilimsel çalışmalarda kullanılması amacıyla tasarlanmış bir dil olmamasına karşın bu alanda çok tercih edilmiştir. Buna sebep olarak dilin yalınlığı, yazılım altyapısı olmayan mühendislerce de kolay

öğrenilebilmesi, genişleyen geliştirici ağı ve en önemlisi bilimsel çalışmaları kolaylaştıran ve hızlandıran kütüphanelere sahip olması gösterilebilir. Bilimsel hesaplamalarda SciPy, matematik çalışmalarında NumPy kütüphaneleri bunlara iyi birer örnektir [49]. Ayrıca yapay zeka alanında ek olarak PyTorch, Pandas, Scikit-learn, TensorFlow, Keras gibi çok sık tercih edilen kütüphaneler barındırmaktadır.

Bu çalışmada Python tercih edilmesinin başlıca sebebi önceki paragrafta da anlatıldığı gibi, yapay zeka, veri bilimi, veri madenciliği alanında genel kabul görmüş bir dil olması olarak özetlenebilir. Bunun yanında, PMML dokümanını destekleyen az sayıda dilden bir tanesi oluşu da ikinci en önemli tercih sebebidir.

PMML dokümanlarının yüklenmesi ve tahminlenmesi işlemi için bir Python kütüphanesi olan PyPMML [50] kullanılmıştır. Aşağıdaki Şekil 4.2'de bir PMML dokümanını yükleyen ve tahmin eden örnek bir kod parçası verilmiştir.

```
from pypmml import Model

# The model is from
http://dmg.org/pmml/pmml_examples/KNIME_PMML_4.1_Examples/sing
le_iris_dectree.xml
model = Model.load('single_iris_dectree.xml')

>>> model.predict({'sepal_length': 5.1, 'sepal_width': 3.5,
'petal_length': 1.4, 'petal_width': 0.2})

{'probability_Iris-setosa': 1.0, 'probability_Iris-
versicolor': 0.0, 'probability': 1.0, 'predicted_class':
'Iris-setosa', 'probability Iris-virginica': 0.0, 'node id':
```

Şekil 4.2 : PMML dokümanından tahmin gerçekleştiren örnek bir Python kodu

4.2.1.2 Flask web çatısı

Flask, Python ile geliştirilmiş, bir mikro web çatısıdır. Mikro web çatısı ifadesi, alışılmış web çatılarında bulunan hazır birçok yapının Flask'te varsayılan olarak bulunmamasından dolayı kullanılmaktadır. Flask'in kendine ait bir veri tabanı soyutlama katmanı, form doğrulaması, güvenlik katmanı, önbellek katmanı bulunmamaktadır [51]. Fakat bu özellikler çatıda bulunmasa dahi bu işlemlerin yapılabilmesi için geliştirilmiş çok sayıda eklentisi vardır ve ihtiyaç halinde projeye dahil edilirler. Bu durum da Flask'in hafif yapısını korumasını sağlamaktadır. Eklentiler sayesinde nesne ilişkili haritalama, form doğrulaması, yetkilendirme, önbellek gibi birçok özellik kullanılabilmektedir.

Flask ile birkaç satırda bir Web API uygulaması ayağa kaldırılabilmektedir. Aşağıda, Şekil 4.3'de en yalın örneğiyle bir Web API uygulamasının kodunu görebilirsiniz.

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello() -> str:
    return "Hello World"

if __name__ == "__main__":
    app.run(debug=False)
```

Şekil 4.3 : Bir Web API için en temel flask kodu örneği

Bu uygulama yerel makinada çalıştırıldığında, internet tarayıcısından http://localhost/hello adresi ile istek yapılabilir ve yanıt olarak "Hello World" ifadesi verildiği görülür.

Bu çalışmada Flask'in tercih edilme sebeplerinden bazıları aşağıda başlıklar halinde verilmiştir.

- *Mikro Web Çatısı*: Geliştirilecek uygulamanın taşınabilir olması ve düşük donanım özellikli sunucularda da zorlanmadan çalışabilmesi önemlidir ve bu yüzden olabildiğince hafif yapılı bir web çatısı olması tercih sebebi olmuştur.
- *Yaygınlık*: Flask genç sayılabilecek bir çatı olmasına karşın geliştiriciler tarafından tercih edilme oranlarında her yıl artış göstererek üst sıralara çıkmıştır. Hatta istatistiklere göre Python temelli web çatıları arasında en çok tercih edilen çatı olmuştur [52]. Bu durum geliştirici ekosisteminin de büyümesini sağlamış ve bu çalışmada da tercih sebeplerinden birini oluşturmuştur.
- *Dokümantasyon*: Flask aktif olarak geliştirilmeye devam eden bir çatıdır ve iyi dokümante edilmiştir. Bunun yanında birçok eklenti desteği ve bu eklentilerin de iyi dokümante edilmesi geliştirme sürecinde çok kolaylaştırıcı rol oynamıştır.

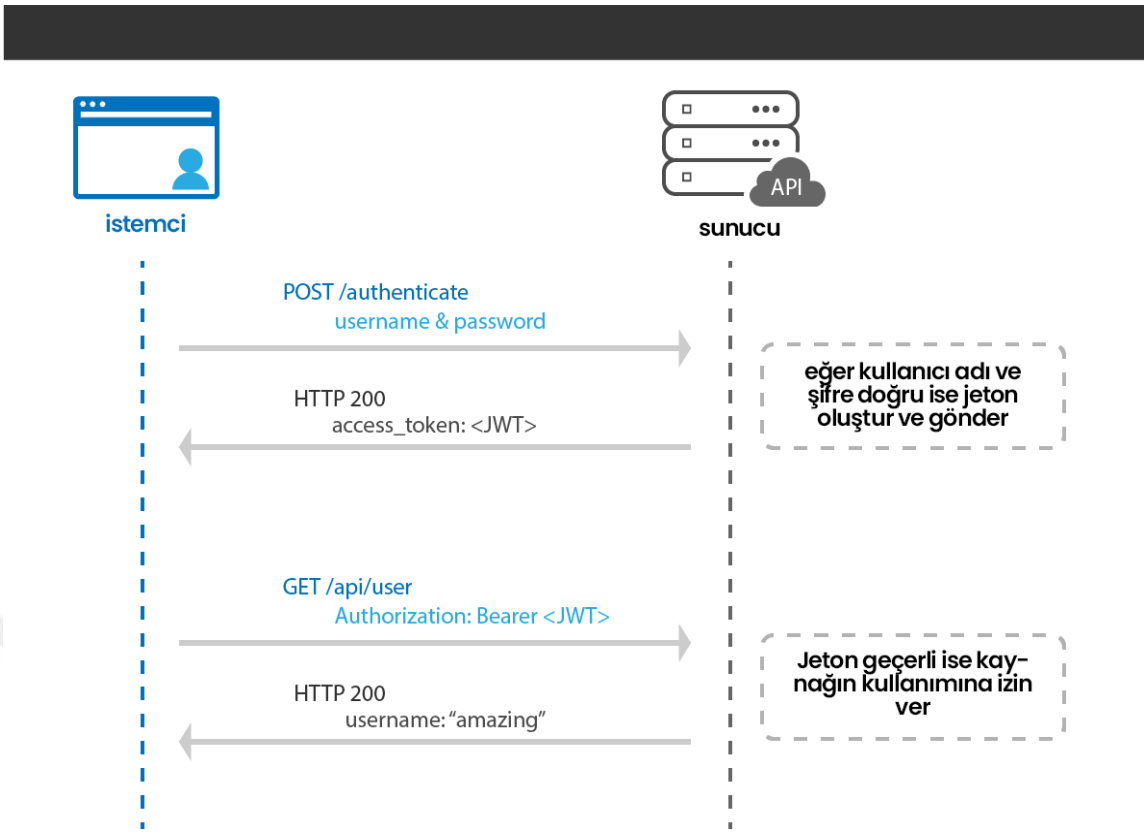
Flask ile uygulama güvenliği

API uygulamasında kullanıcıların bilgilerinin korunması hayati önem taşımaktadır. Bunu gerçekleştirmek için sadece kayıtlı kullanıcılardan gelen isteklerin kabul edilmesi gereklidir. Bunu sağlamak için kullanıcıların giriş yaptığının doğrulanması

gereklidir. Birçok kullanıcı doğrulama yöntemi mevcuttur. Bu çalışmada, durum bilgisiz (stateless) oluşu, farklı istemcilerce de kolay kullanılabilir oluşu gibi avantajları gözetilerek "Jeton Bazlı Kimlik Doğrulama" (Token Based Authentication) kimlik doğrulama yaklaşımı tercih edilmiştir [53].

Jetonlar belirli bir yaşam süresine sahip, içinde şifrelenmiş veya özütlenmiş bilgi taşıyan metin ifadelerdir [54,55]. Javascript obje notasyonu (Javascript Object Notation: JSON) Web Jetonları (JSON Web Token: JWT) bu özütleme işlemini standartlaştırmak için oluşturulmuş bir formattır ve bu çalışmada da tercih edilmiştir.

Genel akış düşünüldüğünde, önce kullanıcı sisteme kayıt olmakta ve kayıt olurken bir bilgi faktörü belirlemektedir. Bu çalışmada bilgi faktörü olarak şifre tercih edilmiştir. Sonrasında uygulamanın diğer uç birimlerini kullanabilmek için bir giriş işlemi gerçekleştirmektedir (Şekil 4.4). Giriş işleminde kullanıcının girdiği e-posta ve şifre kombinasyonu veri tabanındaki bilgiler ile karşılaştırılır ve kullanıcının iddaa ettiği bilgiler doğru ise kullanıcıya bir JWT jetonu yanıt olarak döndürülür. İstemci uygulamalar bu jetonu her yapacağı isteğe eklemek üzere saklarlar. Jetonun süresi dolduğunda yeni jeton almak gereklidir.



Şekil 4.4 : Jeton bazlı kimlik doğrulama akışı

Flask uygulamalarında Kimlik doğrulaması ve Jeton Bazlı Kimlik Doğrulama işlemlerinde kullanılmak üzere *jwt, wekzeug.security* kütüphaneleri kullanılabilmektedir. Bu çalışma için de şifrelerin veri tabanında özüt şeklinde tutulması için *werkzeug.security* kütüphanesinin *generate_password_hash* ve *check_password_hash* modülleri, jetonların oluşturulması ve geçerliliğinin kontrolü için de *jwt* kütüphanesi kullanılmıştır.

Flask ile veri kalıcılığı

Çalışmanın tasarımı gereği bazı verilerin farklı depolama türlerinde depolanması gerekmektedir. Bunlar, kullanıcı verileri, kullanıcıların oluşturduğu proje verileri, projelere bağlı belirsiz sayıda öngörülü model verisi ve modellerin PMML dokümanlarının kendisidir. Verilerin kalıcılığını sağlamak için kullanıcı verilerinde, proje verilerinde ve öngörülü modellerin detay verilerinde İlişkisel Veritabanlarından, PMML dokümanlarının depolanmasında ise daha performanslı olduğu ölçülen [56] işletim sisteminin dosya yönetim sisteminden faydalanılmıştır.

Uygulamanın veri katmanı nesne ilişkisel eşleme (Object Relational Mapping: ORM) yaklaşımı ile geliştirilmiştir.

a) SQLAlchemy ile nesne ilişkisel eşleme

Nesne ilişkisel eşleme, bir veri tabanı tablosunun uygulamanın geliştirildiği dilde tanımlanmış bir objeye denk düşürülecek şekilde kodlanması ve bu obje yardımı ile veri işlemlerinin gerçekleştirilmesine olanak sağlayan bir yaklaşımdır. Eski teknolojilerde veri tabanı işlemleri, uygulama kodları içinde çağırılan yapılandırılmış sorgu dili (Structured Query Language: SQL) sayesinde doğrudan gerçekleştirilmekteydi. Yazılım dillerinde ve veri tabanı yönetim sistemlerindeki teknolojik gelişmeler ile ortaya çıkan ORM sayesinde ise, kod içinde neredeyse hiç SQL yazma ihtiyacı duymadan tüm veri tabanı işlemleri gerçekleştirilebilir olmuştur. ORM'in tercih edilme sebeplerinin başlıcaları şöyle maddelenebilir:

- Veri tabanı yönetim sisteminin değişiminden çok az etkilenir veya hiç etkilenmez. Örneğin MySQL ile çalışan bir uygulama zahmetsizce MSSQL ile çalışır hale getirilebilir. Bu yönüyle uygulamalara oldukça esneklik kazandırmıştır.
- Yazılım geliştiricinin SQL bilme zorunluluğunu neredeyse ortadan kaldırır.
- Yazılımcıların SQL bilgisinden bağımsız olduğu için performanslı SQL sorguları üretebilir [57].
- Kodun okunabilirliğini artırır ve SQL Injection gibi birçok olası güvenlik açığını engeller.

ORM yaklaşımlarını uygulayabilmek için programlama dillerinin kendine has çözümleri mevcuttur [58]. Java programlama dili için Hibernate, C# programlama dili için EntityFramework, PHP programlama dili için CodeIgnite en bilinen ORM çatılarının başında gelmektedir. Bu çalışmada az sayıdaki Python ORM araçlarının [59] arasından SQLAlchemy tercih edilmiştir. Database şemalarının Python kodu ile tanımlanmasının yeterli oluşu, okunabilirliği yüksek fonksiyonlara sahip oluşu, modeldeki değişimleri kolaylıkla veri tabanına senkronize edebilmesi ve iyi dokümante edilmiş oluşu başlıca tercih sebepleri olmuştur. Şekil 4.5'de bir veri tabanı tablosunun SQLAlchemy kütüphanesi ile Python kodunda nasıl temsil edildiği verilmiştir.

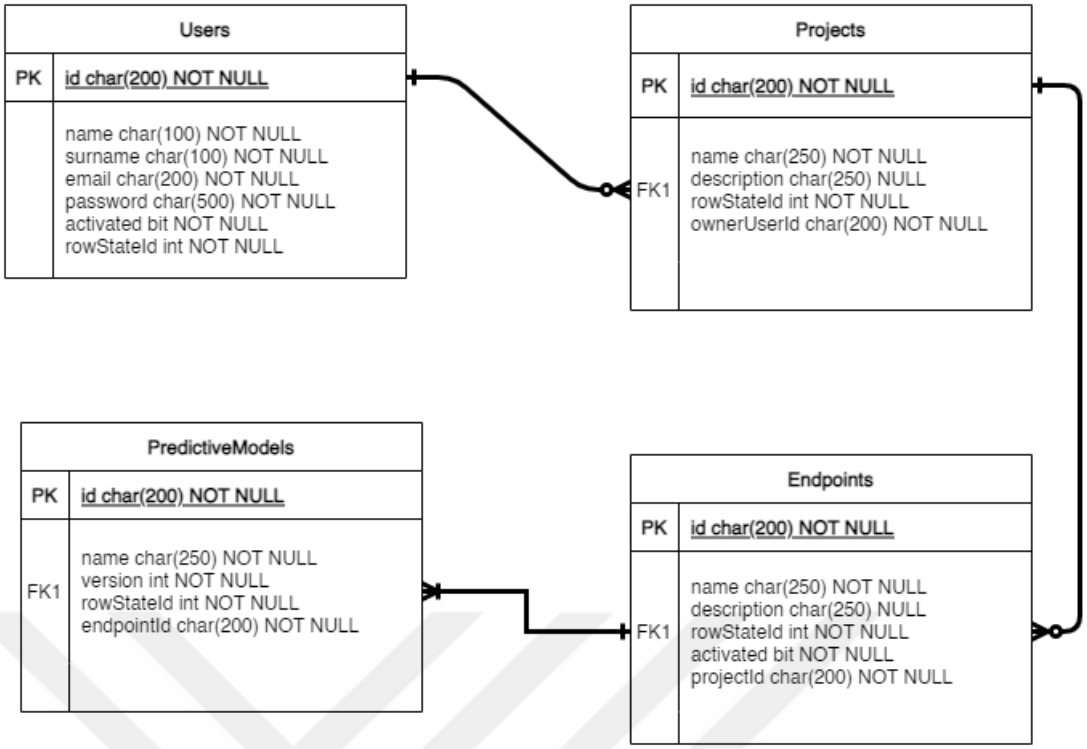
```
class Project(db.Model):
    __tablename__ = "Projects"
    id = db.Column(db.String(200), primary_key=True, nullable=False)
    name = db.Column(db.String(250), nullable=False)
    description = db.Column(db.String(250))
    rowStateId = db.Column(db.Integer, nullable=False, default=1)
    ownerUserId = db.Column(db.String(200),
db.ForeignKey('Users.id'), nullable=False)

    endpoints = db.relationship('Endpoint', backref='project',
lazy=True)

    def __repr__(self):
        return '<Project Id : {id},  Name : {name}>'.format(id =
```

Şekil 4.5 : Bir veri tabanı tablosunun SQLAlchemy ile temsili

Örnek kodda önce temsil ettiği tablo ismi verilmiş, sonra tabloların sütunları veri türleri belirteçleri ile birlikte satır satır belirtilmiştir. Son olarak da tablonun başka tablolarla olan ilişkisi ifade edilmiştir. Yönetilmek istenen tüm veri tabanı tabloları için bu tanımlama yapılmaktadır. Eğer tablolar oluşturulmamış ise SQLAlchemy buradaki ifadelere dayanarak tabloları oluşturabilmektedir.

b) Veri tabanı tasarımı

Çalışmanın başında uygulamanın amaçları net olarak belirlenmiştir ve Sonrasında genel kabul görmüş veri tabanı tasarım adımları takip edilmiştir [60]. Amaçlardan yola çıkılarak bir bilgi havuzu oluşturulmuştur. Sonrasında uygulama için belirlenen isterleri karşılayacak şekilde varlıklar belirlenmiş, bunlara karşılık gelecek tablolar tasarlanmıştır. Planlanan çalışmaya göre tablolarda tutulacak veriler yani sütunlar belirlenmiş, her sütunun amacına uygun veri türü belirlenmiştir. Son olarak varlıkların arasındaki ilişkiler tanımlanmıştır ve ortaya Şekil 4.6'da Varlık İlişki (Entity Relation: ER) diyagramı verilen veri tabanı tasarımı çıkmıştır.



Şekil 4.6 : Veri tabanı varlık ilişki diyagramı

c) Veri tabanı yönetim sistemi

Veri tabanı işlemleri için tercih edilen veri tabanı yönetim sistemi PostgreSQL olmuştur. Postgresql otuz yıldır geliştirilmeye devam eden, açık kaynak bir ilişkisel veri tabanı yönetim sistemidir. Açık kaynak oluşu, geniş kitlelerce kullanılması [61] ve iyi dokümante edilmiş olması tercih sebeplerinin başlıcalarıdır. SaaS olarak çalışması planlanan uygulama için iyi bir tercih olmakla beraber düşük donanım özellikli sunucularda çalıştırmak gerektiğinde Postgresql yerine SQLite kullanımı tercih edilebilir. Bu değişim kullanılan ORM aracı sayesinde problem olmaktan çıkmıştır ve hızlıca gerçekleştirilebilir.

4.2.1.3 Python sanal ortam kullanımı

Python uygulamaları kullanırken her proje için izole bir python ortamı oluşturmak python sürümlerinin ve paketlerinin çakışmasını engelleyerek daha verimli bir çalışma ortamı elde edilmesini sağlar. Bu projede sanal ortam yönetimi için Python 3.x versiyonları ile birlikte gelen venv komutu kullanılmıştır. Oluşturulan sanal ortamda, proje için gereken python paketlerinin yüklenmesi için ise *pip* aracı kullanılmıştır. Ayrıca projenin yayımlanma safhasında pip ile tüm bağımlılıkların bir listesinin çıkarılması işleminin getirdiği kolaylıktan da faydalanılmış, sunucu ortamında hızlıca sanal ortam ayağa kaldırılarak bağımlılıklar kurulabilmiştir. Geliştirme ortamı ve

sunucu ortamı birebir aynı yapıda hazırlanabildiği için versiyon kaynaklı hataların önüne geçilebilmiştir.

4.2.2 Web istemcisi ve önyüz uygulaması

Yapılan çalışmanın önemli bir özelliği de kod yazma tecrübesi çok olmayan, az kodla model eğitilen araçlara aşina kişilerin de zahmetsizce modellerini yayınlayabileceği bir platform sağlamasıdır. Bu sebeple ana uygulamaya entegre çalışan, kullanıcı dostu arayüze sahip bir web istemcisi geliştirilmiştir. Bu istemci sayesinde kullanıcılar sistemi kullanabilmek için üye olabilir, sonrasında projelerini yönetebilir ve modellerini yükleyip uçbirimler oluşturabilirler. Tüm bunları yaparken ana uygulamanın nasıl çalıştığı ile ilgilenmek zorunda kalmazlar.

Günümüz teknolojisinde web uygulamaları geliştirmek için birçok farklı yöntem ve araç mevcuttur. Önceki bölümde bu yaklaşım ve araçlardan bahsedilmişti. Bu bölümde Tek-sayfa uygulaması yaklaşımı ile geliştirilen uygulamanın teknolojik altyapısı incelenmiştir.

4.2.2.1 Javascript, HTML ve CSS

Hiper Metin İşaretleme Dili, web sayfaları oluşturmak amacıyla kullanılan bir metin işaretleme dilidir. 1993 yılında ilk olarak ortaya çıkmıştır [62] ve yaşayan bir standart olarak geliştirilmeye devam edilmektedir [63]. Teorik olarak bir web sitesi geliştirmek için HTML tek başına yeterli olsa da bu durağan bir web sayfasından öteye gidemez. Geliştirilen uygulamanın bir web uygulaması olabilmesi için kullanıcı girdilerine göre yeni çıktılar üretebilecek ve bu çıktıları aynı web sayfasında veya yeni bir web sayfasında gösterebilecek özelliklere sahip olması gereklidir. Bu gibi belge nesnesi modeli üzerinde değişiklikler gerçekleştirilmesinde Javascript programlama dili kullanılmaktadır. Bu çalışmada da kullanıcının ekranda girdiği bilgilerin API uygulamasına iletilmesi ve dönen yanıtın aynı ekranda görüntülenebilmesi gibi işlemlerde Javascript Programlama Dilinden faydalanılmıştır. Javascript 1995 yılında ortaya çıkmıştır ve geliştirilmeye devam edilmektedir. Javascript kodları istemcilerin bilgisayarlarında, tarayıcılar tarafından yorumlanarak çalıştırılır.

4.2.2.2 VueJS javascript çatısı

VueJS bir açık kaynak javascript çatısıdır ve tek sayfa uygulamalar geliştirmek için kullanılır [64,65]. Eklentileri yardımıyla da karmaşık uygulamalar için rotalama,

durum yönetimi gibi çözümler sunabilmektedir. VueJS'in öne çıkan bazı özellikleri şöyledir:

Komponentler
Uygulama içinde tekrar tekrar kullanılabilecek, basit HTML kodlarının kapsüllenmiş haline komponent denebilir. Örneğin uygulamanın hemen her ekranında bulunabilecek buton, yazı alanı, başlık gibi kod parçaları komponent yapısına çevrilerek çok daha az kodla uygulamanın her yerinde yeniden kullanılabilir. Bu sayede komponentteki bir değişiklik de tek noktadan yönetilmiş olur.

Rotalama
Tek sayfa uygulamalarında bulunulan sayfa yeniden yüklenmeden tamamen farklı bir görünüme geçebilmesinin temelinde yatan teknolojidir. Değişen sayfa adreslerine göre görüntülenecek içeriklerin yönetimi rotalama araçları sayesinde gerçekleşmektedir. VueJS 'in en bilinen rotalama aracının ise Vue Router olduğu söylenebilir [66].

Reaktivite
VueJS reaktif bir javascript çatısıdır. Örneğin bir metin girdi alanı bir değişken ile eşleştirilmiş ise, o alanda kullanıcı veri girerken anlık olarak değişkenin değeri de değiştirilmektedir. Ters yönde iletişime bir örnek ise, bir API çağrısı sonrası dönen değer eğer bir değişkene aktarılmış ise ve bu değişken bir HTML tablo ile ilişkilendirilmiş ise, yanıt döndüğü an tablo otomatik olarak tekrar yeni veriyle çizilmektedir. Bu işlemler için geliştiriciler fazladan çaba harcamazlar.

4.2.2.3 Kullanıcı arayüz çatısı

Günümüz şartlarında, kısa sürelerde kullanıcı dostu arayüzlere sahip uygulamalar oluşturmak rekabet edebilmek adına kıymetli bir özelliktir. Bu yüzdendir ki, her proje için uygulamanın tüm kullanıcı arayüzü elemanları baştan tasarlanmasın diye, kendini kullanıcı arayüzü ve kullanıcı deneyimi yönünden ispatlamış kullanıcı arayüzü çatıları yoğun olarak kullanılır olmuştur [67]. Mevcut teknolojide en çok tercih edilen arayüz çatılarının başında Bootstrap gelmektedir. Bu projede, temelinde bootstrap yatan, metronic isimli hazır şablon kullanıcı arayüzü ve kullanıcı deneyimi çözümü olarak tercih edilmiştir. Ayrıca hazır VueJs komponentleri sunan Vuetify arayüzü çatısı da ek olarak bu çalışmada tercih edilmiştir. Oluşan kullanıcı arayüzünün örnek görüntüleri EK A'da verilmiştir.

4.2.2.4 Bağımlılıkların yönetilmesi

Bağımlılıklar, bir uygulamada kullanılan modeller, paketler ve bunlar arasındaki ilişkiler olarak tanımlanabilir. Bir uygulamanın doğru çalışabilmesi için, birbirine bağımlı olan modüllerin dosyalarının proje dosyaları arasında eksiksiz ve doğru versiyonu ile bulunması önemlidir. Bu işlemin manuel olarak yönetilmesi, proje büyüdükçe çok zorlayıcı ve hataya müsait yapıda olacağı için bağımlılıkların yönetimi için araçlar geliştirilmiştir ve kullanılmaktadır. Npm, yarn ve bower javascript paket yöneticilerinin en bilinen üç tanesidir [68]. Bu projede npm tercih edilmiştir.

Npm komut satırından, npm komut satırı arayüzü sayesinde kullanılabilen bir bağımlılık yönetimi aracıdır. NodeJs üzerinde çalışır ve Javascript betik dili ile geliştirilmiştir. Bir çalışmaya yeni bir kütüphane, modül yüklenmek istendiğinde node komut satırı arayüzünden gerekli komutun yazılması yeterlidir. Şekil 4.7'de örnek bir vue modülü yüklemesi için gerekli npm kodu görülebilir. Bu kod önce bilinen npm kaynaklarından vuex ismindeki modülü indirir, npm paketlerinin tutulduğu klasöre taşır ve projenin artık böyle bir bağımlılığı olduğuna dair bilgiyi de packages.config ismindeki konfigürasyon dosyasına yazar.

```
npm install vuex --save
```

Şekil 4.7 : Vue'de örnek bir modül yükleme komutu

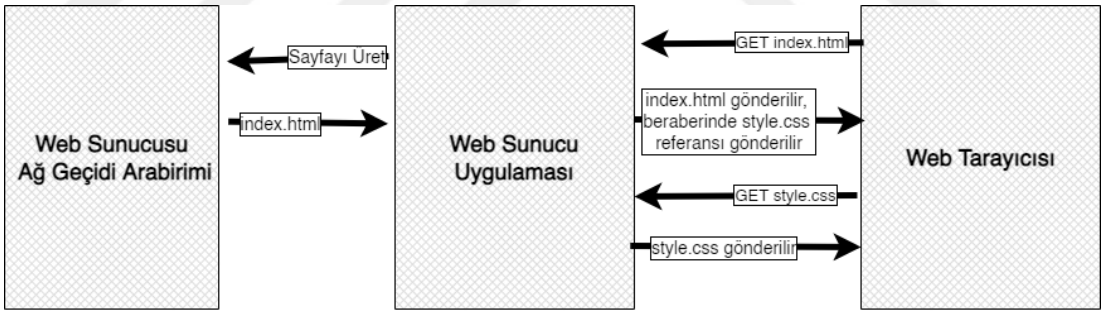
5. UYGULAMANIN YAYIMLANMASI

Uygulamanın geliştirmesi belirli bir olgunluğa ulaştığında, sunucu performansını görmek ve testlerin sunucu ortamında gerçekleştirilebilmesi için yayınlama işlemi gerçekleştirilmiştir. Yayımlanacak sunucu Ubuntu olarak tercih edilmiştir. Bir web uygulamasının linux tabanlı bir sunucu üstünden hizmet verebilmesi için bir web sunucusu uygulamasına ihtiyaç vardır. Bunun yanında Python ile geliştirilmiş web uygulamaları Nginx, Apache gibi geleneksel ve yaygın web sunucu uygulamaları ile doğrudan iletişim kuramazlar. Bu sebeple, web sunucu uygulamaları ve python uygulamaları arasında bir köprü görevi gören Web Sunucusu Ağ Geçidi Arabirimi (Web Server Gateway Interface – WSGI) uygulamaları geliştirilmiştir

5.1 Web Sunucusu Ağ Geçidi Arabirimi Sunucuları

Python kodlarının web sunucular tarafından çalıştırılabilmesi ilk olarak 1990'ların sonlarında Apache web sunucusuna bir modül geliştirerek gerçekleşmiştir. Sonraları bu özellik bir miktar daha geliştirilse de istenilen seviyeye gelememiş ve bazı güvenlik açıklarına da sebep olduğu görülmüştür [69]. Bunun yanında bu geliştirme bir standartlaşma değil, Apache sunucusuna özgü bir iyileştirme olarak kabul edilebilir. Sonraki dönemde Python geliştirici topluluğu tarafından Web Sunucusu Ağ Geçidi Arabirimi adında bir çağrı kuralı geliştirilmiştir. Bu arabirim web sunucu uygulamalarının, Python programlama dilinde geliştirilmiş olan web uygulamalarına ve çerçevelerine istek ulaştırması için bir dizi çağrı kuralından oluşur. Bu standart Python Geliştirme Önerisi PEP-333 olarak yayımlanmıştır [70]. Python 3 versiyonu ile birlikte PEP-3333 olarak ifade edilmiştir.

WSGI arayüzü Python ile web uygulaması geliştiren geliştiricilere web sunucusu seçiminde veya değişiminde esneklik sağlar. Geliştirici web sunucu uygulaması değişse dahi herhangi bir geliştirme yapmadan uygulamanın çalışmasını devam ettirebilir [71]. Örnek bir WSGI sunucusunun işleyişi aşağıda görülmektedir (Şekil 5.1).



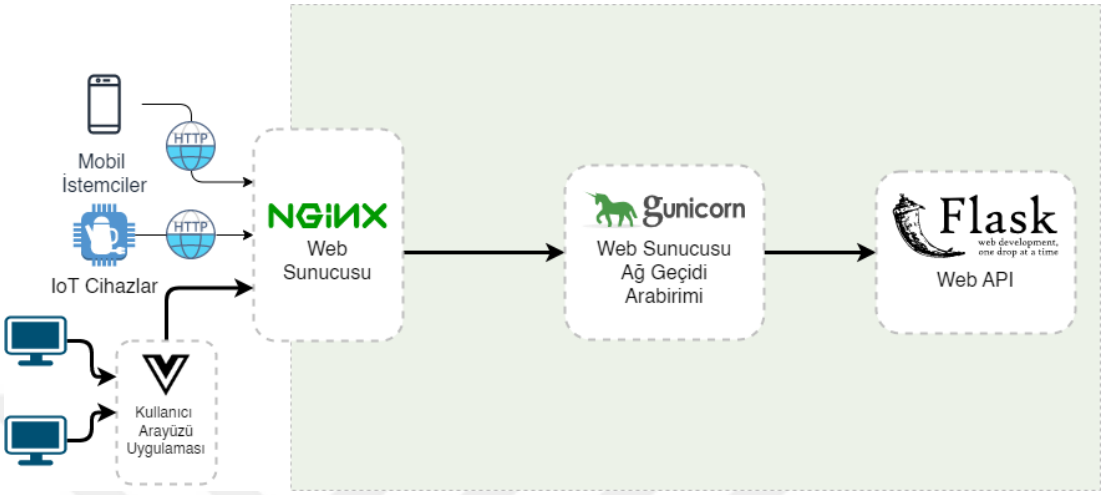
Şekil 5.1 : Web sunucusu ağ geçici arabiriminin çalışma yapısı

5.2 Web Sunucu Uygulamaları

Web uygulamalarının sunucu sistemleri üzerinden son kullanıcılara sunulabilmesini sağlayan uygulamalardır. Apache, Nginx en bilinen örneklerindendir. Bu çalışmada Web Sunucusu Ağ Geçidi Arabirimi ile iletişimde olacak olan ve son kullanıcılardan gelecek web isteklerini karşılayacak olan uygulama Nginx olarak tercih edilmiştir.

Nginx 2002 yılında Igor Sysoev tarafından geliştirilmiş, özgür ve açık kaynak kodlu bir web sunucu uygulamasıdır. Unix, Linux, Mac OSX, Microsoft gibi yaygın işletim sistemlerinde çalışabilmektedir. Nginx'in bir web sunucusu olmasının yanında, ters

vekil sunucusu ve yük dengeleyici olarak çalışması da mümkündür. Nginx'in Web Sunucusu Ağ Geçidi Arabirimi desteğinin olması ve düşük sistem gereksinimleri başlıca tercih sebebi olmuştur. Çalışma neticesinde ortaya çıkan akış ve bu akışı sağlayacak uygulamaların iletişimi Şekil 5.2'de verilmiştir.



Şekil 5.2 : Sistem akış diyagramı

6. WEB SERVİSE DÖNÜŞECEK PMML MODELLERİNİN ÜRETİLMESİ

6.1 Örnek Veri setlerinin Belirlenmesi

Gerçekleştirilen çalışmada performans kritik bir gereksinimdir [72]. Bu yüzden, geliştirilecek API uygulamasının tahmin performansını etkileyecek önceden belirlenmiş etkenlerin yanında, henüz belirlenmemiş etkenler de mevcuttur. Bu bölümde performansa etki edebileceği öngörülen etkenlerin tespiti için bazı karşılaştırmalar yapılmıştır. Veri madenciliği çalışmalarında sıklıkla tercih edilen ve literatürde yaygın kabul görmüş bazı veri setleri kullanılarak, veri setinin eğitilip PMML'e dönüştürülmesi durumunda boyut, özellik sayısı ve algoritma yönünden ne gibi farklılıklar yarattığı incelenmiştir.

API Performansına etki edeceği öngörülen ilk parametre isteklerin gövde boyutlarıdır. Bir isteğin gövde boyutu ne kadar büyük ise, sunucuya iletilmesi de benzer oranda yavaş olacaktır ve toplam yanıt süresini uzatacaktır [39,73]. İkinci parametre ise bir modelin tahmin esnasında belleğe alınması gerektiğinden, modelin boyutudur denilebilir. Büyük PMML dokümanları belleğe alınma aşamasında daha fazla zaman

harcayacağı için tahmin süresini uzatacak, toplam yanıt süresinin de uzamasını sağlayacaktır. Bu durumda PMML dokümanının boyutu önemlidir denilebilir ve bu boyutu etkileyen faktörler araştırılmıştır.

Karşılaştırma yapılırken beş farklı veri seti kullanılmıştır. Veri setleri seçilirken verinin büyüklüğü, özelliklerinin sayısı ve literatürde kabul görmüş olma durumu kriter olarak belirlenmiştir. Veri setleri ve kısa açıklamaları yazının devamında verilmiştir.

6.1.1 Iris veri seti

İlk kez R.A Fisher ‘ın 1936 yılındaki yayımlanan araştırmasında kullanılmıştır. Iris çiçeğinin üç farklı türü için, taç yaprak ve çanak yaprak uzunlukları ölçümlerinin 50 ölçüm sonucunu içerir [31].

6.1.2 Audit risk veri seti

Denetçilerin hileli firmaları mevcut geçmişe dönük risk faktörleri verileri yardımı ile tahmin edebileceği modeller üretmesini sağlayan veri setidir. 777 örnekten oluşan veri setinde farklı sektörlerden firmaların mali verileri ve geçmiş denetim verileri mevcuttur [74].

6.1.3 Heart Disease veri seti

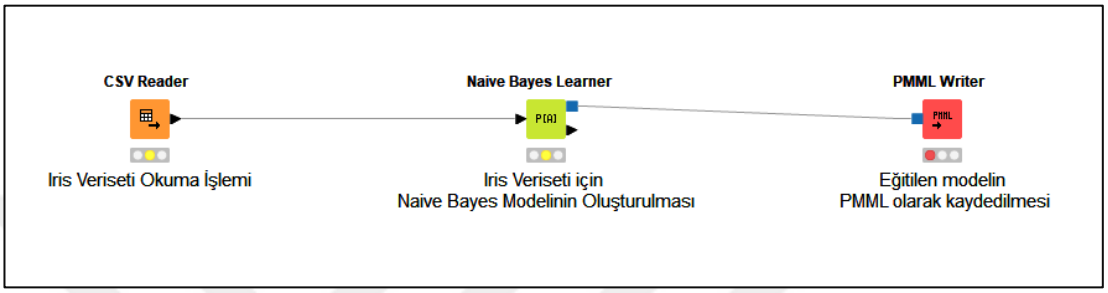
Kalp hastalığı teşhisine ilişkin, Cleveland Clinic Foundation tarafından bağışlanmış olan verilerdir [75-77]. Veri seti orijinalinde 76 farklı özellik içeriyor olsa da bu güne kadar yapılan çalışmalarda bu özelliklerin 14 tanesinin kullanıldığı görülmüştür. Veri setinde Cleveland Clinic Foundation verilerinin haricinde üç hastanenin daha verisi paylaşılmıştır fakat genele bakıldığında Cleveland Clinic Foundation verileri literatürde daha fazla atıf almıştır.

6.1.4 Adult Income veri seti

Amerika Birleşik Devletleri’nin 1994 yılındaki nüfus sayımı verisinden, Barry Becker tarafından çıkarılmış ve bağışlanmıştır. Bireylerin yaş, tahsil durumu, cinsiyet vb. 14 özelliğini içerir ve yıllık toplam gelirinin 50.000 Dolar ve üstünde olma durumu tahmin edilir [78].

6.2 Veri Setlerinin PMML'e Dönüştürülmesi

Belirlenmiş olan veri setlerinin PMML dokümanlarına dönüştürülmesi için öncelikle seçilen belirli bir yöntem ile eğitilmeleri gereklidir. Eğitim işlemi ve PMML'e dönüştürme işlemi KNIME Analytics Platform uygulaması ile gerçekleştirilmiştir.
KNIME açık kaynak kodlu bir veri madenciliği yazılımıdır. Sürükle bırak kullanıcı deneyimi ile birçok işlem kolayca gerçekleştirilebilmektedir. Şekil 6.1'de örnek bir çalışma alanı görebilirsiniz.



Şekil 6.1 : Knime uygulamasında bir veri setinin eğitilmesi ve PMML olarak kaydedilmesi için kurulmuş bir iş akışı

Şekil 6.1'de bir örneği görülen eğitim sürecinin sonucunda iki farklı algoritma ile yukarıda belirtilen veri setleri kullanılarak modeller üretilmiş ve PMML dokümanlarına dönüştürülmüştür. Kullanılan algoritmalar şunlardır:

6.2.1 Eğitilecek modellerde kullanılan algoritmalar

6.2.1.1 Karar ağacı sınıflandırıcısı

Belirlenmiş bir nominal hedef özelliğine göre verilerin belirlenmiş kurallarla sürekli daha küçük düğümlere bölünerek son karar düğümü üretilene kadar devam ettirilen bir denetimli öğrenme algoritmasıdır. Düğüm sayısının PMML dokümanının boyutuna olan etkisinin de gözlemlenmesi için tercih edilmiştir. Knime uygulaması karar ağacı öğrenmesi düğümünü C4.5 algoritması ile gerçeklemektedir.

6.2.1.2 Naive Bayes sınıflandırıcısı

Bağımsız varsayımlar yardımıyla Bayes teoremini baz alan olasılıklı bir sınıflayıcıdır. Algoritmaya önceden verilen öğretilmiş verilerden üretilen olasılık hesaplarının sonraki verilerde de kullanılarak sonucun doğru tahmin edilebilmesini hedefleyen bir yaklaşıma sahiptir.

Yukarıda açıklanmış olan veri setleri ve eğitim algoritmaları kullanılarak üretilen PMML dokümanlarının boyutları Çizelge 6.1'de verilmiştir.

Çizelge 6.1 : Farklı veri setleri ve eğitim algoritmaları için PMML doküman boyutları

Veri Seti	Model	Dosya Boyutu(kb)
Adult Income	Karar Ağacı	2165
Adult Income	Naive Bayes	35
Iris	Karar Ağacı	6
Iris	Naive Bayes	5
Audit Risk	Karar Ağacı	2039
Audit Risk	Naive Bayes	186
Heart Disease	Karar Ağacı	64
Heart Disease	Naive Bayes	16

7. PERFORMANS ÖLÇÜMLERİ, SONUÇLAR VE TARTIŞMA

7.1 Ölçümlerin Gerçekleştirildiği Ortamlar ve Kullanılan Araçlar

Uygulamanın gerçekte karşı karşıya kalacağı yüklerin benzerleri üretilerek sistemin nasıl davranacağı ve ne sonuçlar üreteceği araştırılmıştır. Bu araştırma yapılırken hizmet olarak uygulama yaklaşımı üzerinde çalışılmıştır. Bu sebeple API uygulaması gerçek hayat örneğindekine benzer bir ortamda yayımlanmıştır. Bu ortam özel bulut hizmetleri veren bir kurumun veri merkezinde yer alan bir sanal sunucu olarak seçilmiştir. Kullanıcı davranışlarını ve anlık kullanıcı sayısındaki değişimi taklit edebilmek için ise açık kaynak bir yazılım test aracı kullanılmıştır.

7.1.1 Apache JMeter

Apache Jmeter, çeşitli uygulamalarının performanslarının analiz edilmesinde, yük testlerinin yapılmasında kullanılan gelişmiş, açık kaynak bir test aracıdır. Java ile geliştirilmiştir. Gelişmiş bir kullanıcı arayüzü ve komut satırı arayüzü mevcuttur. Jmeter isteklerdeki değişkenlerin rastsallığını sağlamak, yanıt doğrulaması yapmak ve çeşitli türlerde raporlar sunmak gibi önemli özelliklere sahiptir. Ayrıca yaygın eklenti desteği sayesinde bünyesinde halihazırda bulunmayan birçok özellik hızlıca entegre edilebilmektedir. Jmeter, Java Database Connectivity (JDBC), File Transfer Protocol

(FTP), Lightweight Directory Access Protocol (LDAP), Java Messaging Services (JMS), HTTP,HTTPS, Simple Mail Transfer Protocol (SMTP) ve genel Transmission Control Protocol (TCP) protokollerinde test aracı olarak kullanılabilmektedir [79].
Bu çalışmada kullanılma amacı, API uçbirimlerinin yanıt sürelerinin ölçümlenmesi olarak genelleştirilebilir. Performans test sonuçları, Apache Jmeter'ın sunduğu performans metrikleri baz alınarak değerlendirilmiştir.

7.1.2 Uygulama sunucusu

Uygulamanın test işlemlerinde, gerçek hayat senaryolarındaki performansına en yakın performansı sunabilmesi için gerçekte çalışacağı ortamın bir benzerinde yayımlanması yöntemi tercih edilmiştir. Bunun için özel bulut hizmeti veren bir kurumun veri merkezinde konumlandırılmış bir sanal sunucu tercih edilmiştir. Sunucunun özellikleri aşağıdaki çizelgede verilmiştir (Çizelge 7.1).

Çizelge 7.1 : Test sunucusunun özellikleri

Birim	Özellik
İşletim Sistemi	Ubuntu 20.04
Depolama Kapasitesi	250 GB SSD
Bellek	16 GB
İşlemciler	4 Core Intel Xeon Gold 6238

7.1.3 Test Varyasyonları ve Parametreleri

Geliştirilen sistemin hizmet, mobil uygulamalar, nesnelerin interneti uygulamaları gibi hizmet vermesi hedeflenen alanlar düşünüldüğünde sistemin ani yük ve normal yük altında nasıl davrandığının ölçümlenmesi hedeflenmiştir. Apache Jmeter aracı ile bir test planı oluşturulurken tanımlanan bazı değerler yardımı ile gerçekleştirilecek testin amacı şekillendirilebilmektedir. Testlerin karakterini belirleyen bu parametreler aşağıda açıklanmıştır.

7.1.3.1 İş parçacığı sayısı

İş parçacığı sayısı her biri ayrı çalışan kaç farklı iş parçacığı oluşturulacağını belirler. Her iş parçası uygulamaya istek yapan bir kullanıcı olarak düşünülebilir. Bu iş parçacıkları eş zamanlı çalışabilmektedirler.

7.1.3.2 Artırma süresi

Artırma süresi parametresi, belirlenmiş olan iş parçacığı sayısına kaç saniyede ulaşılacağını belirler. Örneğin 100 iş parçacığı belirlenmiş ve artırma süresi 10 olarak seçilmiş ise sistem her saniyede 10 yeni kullanıcıyı teste dahil eder. Artırma süresi 1 saniye seçilmiş olsa idi sistem 1 saniye içinde 100 iş parçacığını çalıştırırdı. Diğer bir deyişle sisteme aynı anda 100 kullanıcının istek yapması taklit edilmiş olurdu. Artırma süresi yapılan testin karakteristiğini etkileyen önemli bir parametredir. Sistemin nasıl bir yük altında kalacağını belirleyen temel değişkenlerdendir.

7.1.3.3 Döngü sayısı

İş parçacığı ve artırma süresi sonunda ortaya çıkan istek örüntüsünün kaç defa tekrar edileceğinin belirlendiği değişkendir. Kullanıcıların sisteme girip birden fazla istek yapması durumunu simüle etmekte kullanılabilir. Sistemin uzun süreli kullanımlar altındaki davranışını görmek için de kullanımı mümkündür.

7.1.3.4 Test varyasyonları

Sistemin ani yük ve tahmin edilen normal yük altındaki davranışları ölçümlenecektir. Normal yük oluşturmak için parametreler 100 iş parçacığı, 100 saniye artırma süresi ve 10 döngü, ani yük oluşturmak için ise 100 iş parçacığı, 1 saniye artırma süresi ve 10 döngü olarak belirlenmiştir.

7.1.4 Değerlendirme Ölçütleri

Bir yazılımın performans testleri gerçekleştirilirken yazılımın türüne göre farklı metrikler üzerinden değerlendirmeler yapılabilir. Web uygulamalarının yük testlerinde uygulamanın kullanım senaryosuna göre değişiklikler olsa da temelde kullanıcı davranışlarını taklit eden senaryolar üzerinden metrikler kararlaştırılır. Dolayısıyla bir web uygulamasının metriklerine bakıldığında, sistemin yük altında nasıl davrandığının yorumlanabildiği metriklerin kullanıldığı görülür. Yapılan testlerin yorumlanmasında kullanılan metrikler aşağıda verilmiştir.

7.1.4.1 İstek sayısı

Çalışan test adımında sunucuya kaç adet isteğin iletildiği bilgisidir. Bu sayı test planı yapılırken belirlenir. Örneğin 100 iş parçacığı ve 10 döngü ile oluşturulmuş bir test planında istek sayısı 1000 olacaktır.

7.1.4.2 Geçen Süre

Bir isteğin yollandığı an ile isteğe gelen son yanıtın ulaştığı an arasında geçen sürenin milisaniye cinsinden değeridir.

$$\Delta t = t_f - t_0 \quad (7.1)$$

t_f, ilk isteğin gönderildiği zamanı ifade eder.
t_0, son yanıtın alındığı zamanı ifade eder.

7.1.4.3 Ortalama

Test planının tamamının icra edilmesi neticesinde oluşan ortalama yanıt süresidir. Tüm isteklerin geçen süre değerlerinin toplanıp istek sayısına bölünmesi ile elde edilir.

$$\mu = \frac{1}{n} * \sum_{i=1}^{n} x_i \quad (7.2)$$

Denklemdeki n test planındaki toplam istek sayısı, x_i ise her bir isteğin yanıt süresidir.

7.1.4.4 Standart sapma

Standart sapma değeri, ortalama değerden ne kadar sapma oluştuğunu ifade eder. Bir test icrası sonunda oluşan standart sapma değerinin ortalama değerin yarısı kadar veya daha az olması sistemin sağlıklı olduğunu gösterir [80]. Test sonucundaki standart sapma, elde edilen ortalama değere ne kadar güven duyulacağını belirler ve aynı zamanda sistem performansının ne kadar tutarlı olduğunu yorumlamada kullanılabilir.

$$SD = \sqrt{\frac{\sum |x-\mu|^2}{N}} \quad (7.3)$$

Denklemdeki x, o anki örnek isteğin yanıt süresi, μ test planının ortalama yanıt süresi, N o ana kadar yapılmış olan toplam istek sayısını ifade eder.

7.1.4.5 Doğruluk

Test icrası sonunda elde edilen doğruluk değeri, birim zamanda (saniye, dakika, saat) ne kadar isteğin sunucu tarafından yanıtlanabildiğini ifade eder.

$$T = \frac{N}{t_f - t_0} \qquad (7.4)$$

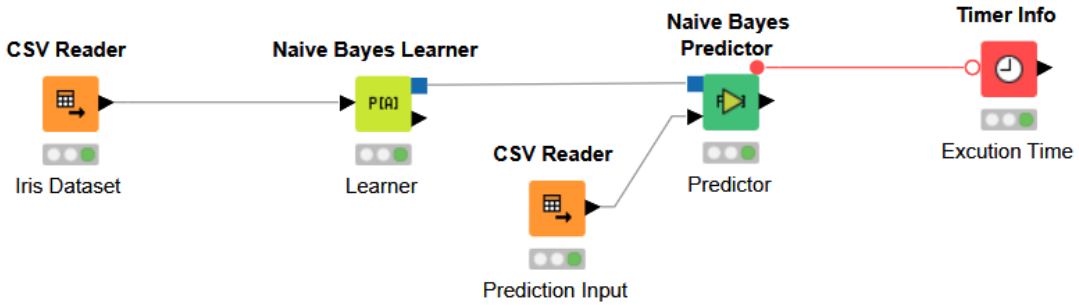
Denklemde N test planındaki toplam istek sayısını, t_f test planındaki son isteğin yanıtlandığı tarihi, t_0 ise ilk isteğin gönderildiği tarihi ifade eder.

7.1.4.6 Hata oranı

Test icrası boyunca hata ile sonlanan istek sayısının tüm isteklere oranıdır. %0.0 hata oranı tüm isteklerin başarılı şekilde yanıtlandığını ifade eder.

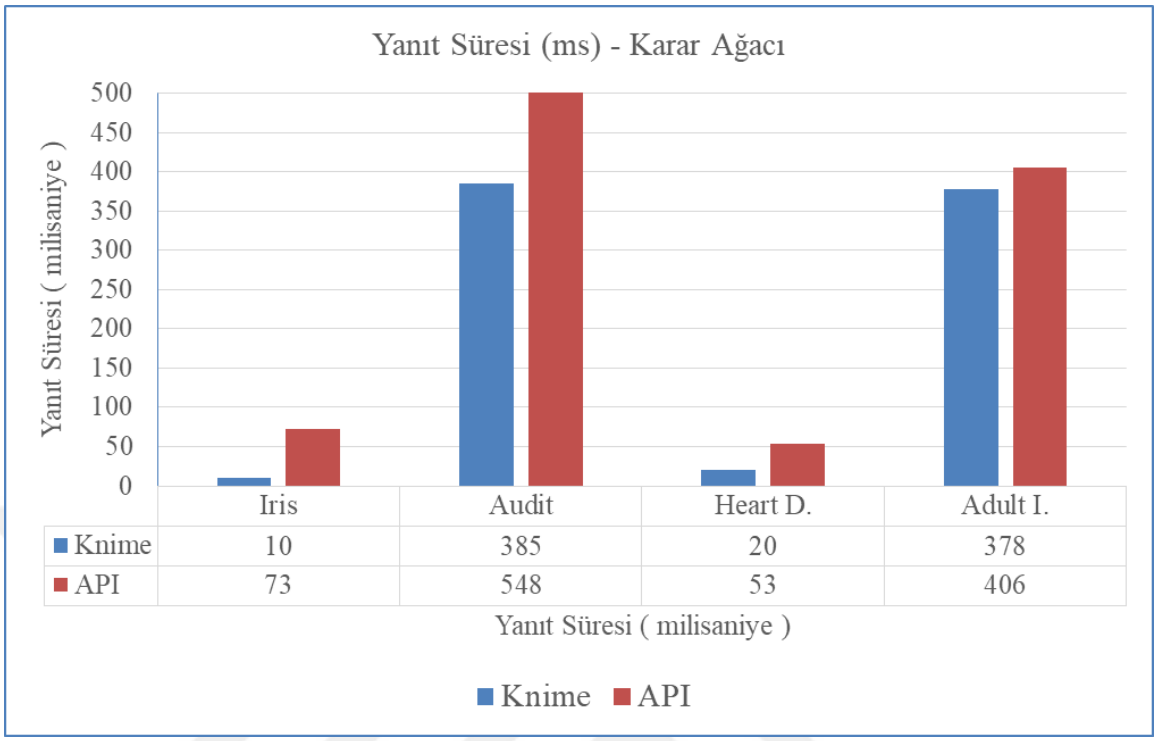
7.2 Modellerin Web Servisine Dönüştürülmesi Durumunda Öngörüleme Performansının Değişimi

Bu ölçümde, bir modelin üretildiği ortamdan kullanılabileceği ortama taşınması ve burada web servisi olarak hizmet verir hale gelmesinin, modelin tahmin süresinde ne gibi bir değişim yarattığı incelenmiştir. Ölçümler Iris, Heart Disease, Adult Income ve Audit Risk veri setleri üstünde, Naive Bayes sınıflandırıcısı ve Karar ağacı sınıflandırıcı algoritmaları ile yapılmıştır. Öncelikle bu dört model, iki farklı algoritma ile Knime ortamında eğitilmiştir. Sonrasında bir satırlık test verisi ile Knime ortamında tahmin işlemine sokulmuştur. Knime uygulamasının tahmin düğümlerinin kullanıldığı bu işlemin ne kadar süre aldığını ölçümlemek için de yine Knime uygulamasının Timer Info isimli düğümü kullanılmıştır. Timer Info düğümü Knime uygulamasında bir akıştaki tüm düğümler için zaman ve icra raporları üretir [81]. Oluşturulan akışların arasından Iris veri seti için Naive Bayes modeli ile yapılan örnek bir Knime akışı aşağıdaki şekilde verilmiştir (Şekil 7.1).

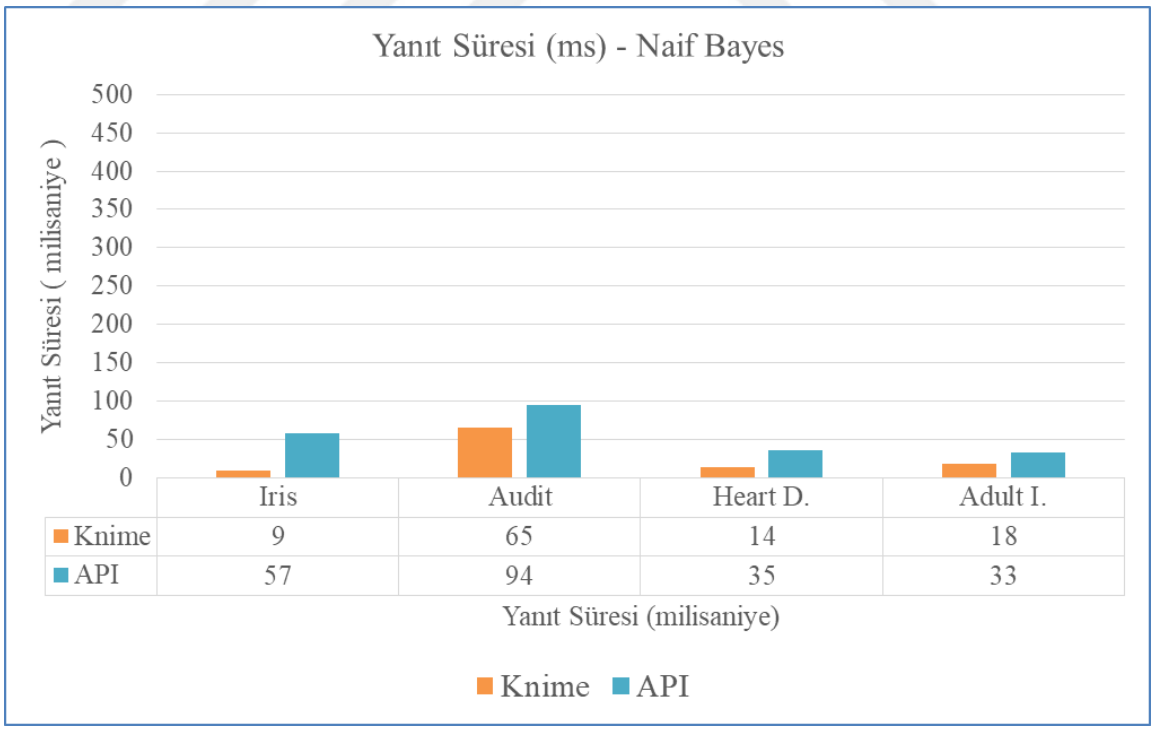


Şekil 7.1 : Knime uygulamasında, tahmin icra süresinin ölçümünü de içeren örnek bir Naive Bayes öğrenmesi

Testlerin sonucunda ortaya çıkan yanıt sürelerinin oluşturduğu grafikler aşağıda Şekil 7.2 ve Şekil 7.3 olarak verilmiştir.



Şekil 7.2 : Karar ağacı algoritması ile eğitilen modellerin dört farklı veri seti için Knime ve API ortamlarındaki yanıt süreleri grafiği



Şekil 7.3 : Naive bayes algoritması ile eğitilen modellerin dört farklı veri seti için Knime ve API ortamlarındaki yanıt süreleri grafiği

7.3 Veri seti büyüklüğünün PMML çıktısına etkisinin ölçülmesi

PMML dosyaları her tahmin işleminde dosya sisteminden belleğe aktarılarak seri durumdan çıkarılır ve öngörülemeye hazır hale getirilir. Dosya büyüklüğü bu yükleme süresini etkilediğinden mümkün olduğunca düşük tutulması sistemin performansı için önemlidir. Bu ölçümü yapabilmek için Adult Income ve Heart Disease veri setleri önce saf halleri ile karar ağacı ve naive bayes eğitimine sokularak PMML çıktıları üretilmiştir. Sonrasında veri setlerinin satır sayısı yarı yarıya azaltılarak aynı adım tekrarlanmıştır. Sonuçlar Çizelge 7.2'de verilmiştir.

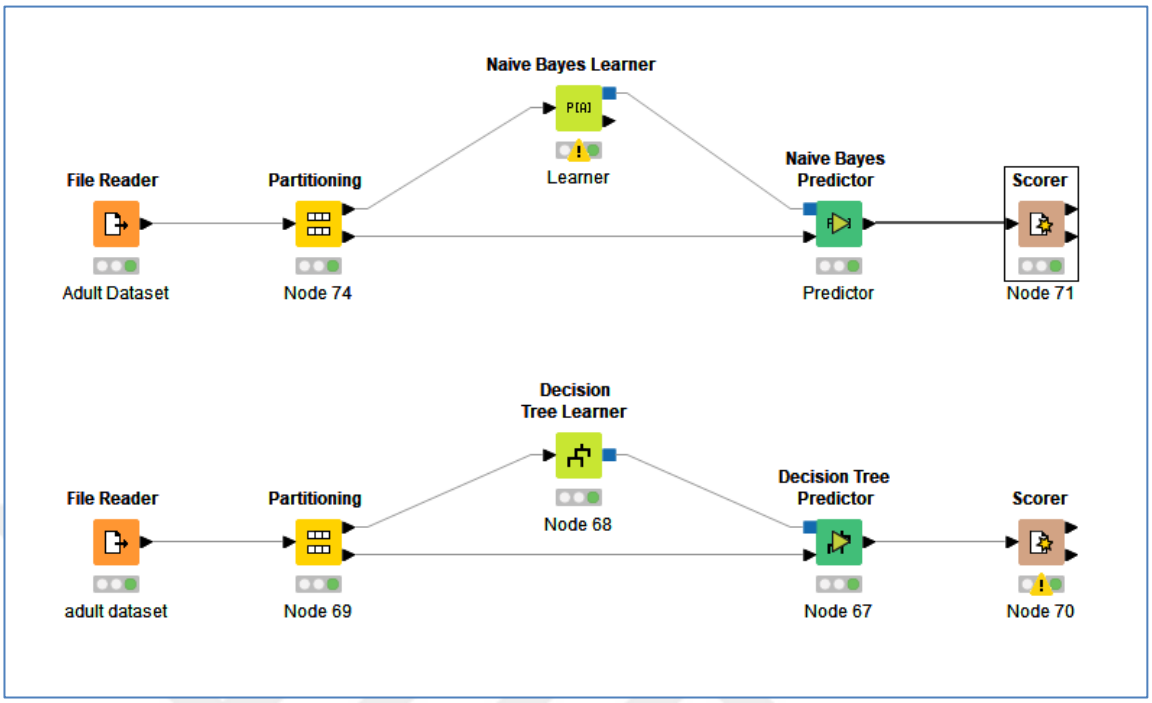
Çizelge 7.2 : Veri setleri büyüklüğünün değiştirilmesi ile elde edilen, farklı modellere ait PMML dokümanlarının dosya büyüklükleri

Veri seti	Satır sayısı	Karar Ağacı Modeli		Naive Bayes Modeli	
		Tam veri seti (KB)	Yarım veri seti (KB)	Tam veri seti (KB)	Yarım veri seti (KB)
Adult Income	32560	2165	1058	35	34
Iris	150	6	3	5	4
Audit Risk	775	2039	856	186	174
Heart Disease	300	64	34	16	16

Ölçüm sonuçlarına bakıldığında veri setinin PMML dosya boyutu üzerinde tek başına bir etkisinin olduğundan söz edilemez. Örneğin 32560 satır içeren Adult Income veri setinin PMML dokümanı 2165 KB iken 16280 satır veri ile eğitilen aynı modelin PMML doküman çıktısı 1058 KB olmaktadır. Dosya boyutundaki bu azalış diğer dört veri setinde de gözlemlenmiştir. Fakat Naive Bayes modelinden dönüştürülen PMML dokümanları, veri boyutu yarı yarıya azaltılmış da olsa büyüklük olarak çok az bir değişim göstermiştir. Dolayısıyla PMML dosyalarının boyutunda sadece veri setinin büyüklüğü değil, eğitimde kullanılan modelin türü, veri setinin bu modele uygunluğu gibi kriterlerin de etkili olduğu söylenebilir.

Yapılan ölçüm özelinde, PMML boyutu düşük olan ve veri setinin büyüklüğünden de az etkilenen yöntemin seçilmesi önerilmektedir. Fakat bu karar verilirken iki modelin tahmin skorları da incelenebilir. İki model arasında tahmin skoru çok farklı çıkmıyorsa PMML boyutu tercih sebebi olabilir. Aşağıdaki Şekil 7.4'de Adult Income veri seti için Knime ortamında geliştirilmiş bir öngörüleme akışı görülebilir. Veri seti %70

eğitim, %30 test için bölümlenmiştir. İki ayrı model için çalıştırılan bu akışın sonunda ortaya çıkan öngörüleme skoru Çizelge 7.3'de verilmiştir.



Şekil 7.4 : Adult Income veri setinin iki farklı modeldeki eğitimi sonucunda elde edilen başarı değerlerinin ölçülmesi için tasarlanmış Knime akışı

Çizelge 7.3 : Adult Income veri setinin naive bayes ve karar ağacı algoritmaları ile elde edilen modellerin doğruluk değerleri

Model Türü	Doğruluk Değeri
Naive Bayes	0.84
Karar Ağacı	0.82

Yapılan ölçümde iki farklı algoritma ile eğitilen modellerin skorlama performanslarının birbirine çok yakın çıktığı görülmüştür. Bu tür durumlarda PMML doküman çıktısı daha düşük boyutlarda olan yöntem tercih edilebilir.

7.4 Genel Yük Testleri

Test varyasyonları ve parametreler başlığında belirtilen parametreler kullanılarak iki farklı yük testi planı hazırlanmıştır. Yük testleri planları hazırlanırken önce her uçbirimin beklediği istek gövdesi oluşturulur. İstek gövdesi oluşturulurken, gerçek bir kullanıcıyı taklit edecek şekilde, yollanacak değerlerin rastgele ve veri setindeki aralıklara uyarak üretilmesi önemlidir. Bunu gerçekleştirmek için istek gövdelerinde

sabit yazılmış değerler yerine rastgele sayı üreten Jmeter fonksiyonları kullanılmıştır. Şekil 7.5'de Adult Income veri setinin istek gövdesi örneği görülebilir.

```
[
${__Random(17,90)},
"${__RandomFromMultipleVars(wc1|wc2|wc3|wc4|wc5)}",
${__Random(12300,1500000)},
"${__RandomFromMultipleVars(edu1|edu2|edu3|edu4|edu5)}",
${__Random(0,17)},
"${__RandomFromMultipleVars(marry1|marry2|marry3|marry4|marry5)}"
,
"${__RandomFromMultipleVars(occ1|occ2|occ3|occ4|occ5)}",
"${__RandomFromMultipleVars(rel1|rel2|rel3|rel4|rel5)}",
"${__RandomFromMultipleVars(race1|race2|race3|race4|race5)}",
"${__RandomFromMultipleVars(gender1|gender2)}",
${__Random(0,10000)},
${__Random(0,5000)},
${__Random(1,70)},
"${__RandomFromMultipleVars(ctr1|ctr2|ctr3|ctr4|ctr5)}"
]
```

Şekil 7.5 : Gerçeğe daha yakın test senaryoları için yapılan her web isteğinin değerlerini belirli aralıkta rastgele belirlemek için hazırlanmış Jmeter istek gövdesi

Test planları tamamlandıktan sonra yük testleri çalıştırılmıştır. Apache Jmeter uygulaması yük testlerinin kullanıcı arayüzünden değil, komut satırından çalıştırılmasını önerir. Bunun sebebi de test istemcisinin kaynaklarını tam olarak kullanabilmeyi hedeflemiş olmasıdır. Test sonuçlarının anlık olarak kullanıcı arayüzüne yansıması testin performansını olumsuz etkilemektedir.

Test planları çalıştırıldıktan sonra toplanan veriler ve sonuçlar aşağıda verilmiştir.

7.4.1 Normal Yüklü Sistem Testi

100 kullanıcı, 100 saniye artırma süresi ve 10 tekrar ile gerçekleştirilen testin sonucunda farklı modeller için Çizelge 7.4 ve Çizelge 7.5'deki veriler elde edilmiştir.

Çizelge 7.4 : Karar ağacı modeli ile üretilen PMML dokümanları ile yapılan normal yüklü sistem testinin sonuçları

Veri Seti	İstek Sayısı	Hata Oranı (%)	Ortalama (ms)	Standart Sapma	Doğruluk (istek/dakika)
Adult Income	1000	0.0	15669	6567	224
Iris	1000	0.0	40	9	604
Audit Risk	1000	0.0	12927	5598	253
Heart Disease	1000	0.0	45	10	604

NaiveBayes

Çizelge 7.5 : Naive bayes modeli ile üretilen PMML dokümanları ile yapılan normal yüklü sistem testinin sonuçları

Veri Seti	İstek Sayısı	Hata Oranı (%)	Ortalama (ms)	Standart Sapma	Doğruluk (istek/dakika)
Adult Income	1000	0.0	40	9	604
Iris	1000	0.0	40	25	603
Audit Risk	1000	0.0	106	26	600
Heart Disease	1000	0.0	38	16	604

Düzenli olarak yükü artan fakat birim zamanda binen yük olarak da çok agresif davranmayan bir ölçümün sonucu değerlendirildiğinde sistemin isteklere eksiksiz olarak yanıt verebildiği görülebilir. Fakat bazı uçbirimlerin ortalama yanıt süreleri dikkate değer şekilde yüksektir. Karar ağacı modelinden üretilmiş olan uçbirimlerden Adult Income ve Audit Risk modellerinin uçbirimleri ortalama sürelerde kullanıcı deneyimini kötü etkileyecek değerlere ulaşmışlardır. Yapılan araştırmalarda kullanıcıların 10 saniye ve üzeri beklemelerde dikkat kaybı yaşadığı ve devam eden işi iptal etme veya başka bir işe geçme eğilimi gösterdiği sonucu elde edilmiştir [82]. Ek olarak ortalamaların standart sapmaları ortalamaya göre makul görünse dahi sistemin bu model türü için tutarlılığını düşük olarak değerlendirmeyi mümkün kılmaktadır. Bu iki model için sistemin stres altında kaldığı söylenebilir. Karar Ağacı modeli özelinde, Çizelge 7.4 ve Çizelge 7.5'e birlikte bakıldığında modellerin ortalama yanıt süreleri ile PMML dosya boyutları arasında bir alaka olduğu söylenebilir. Dosya boyutunun büyük oluşunun yanı sıra, modellerin çok fazla karar ağacı düğümü içermesi de tahmin icraasını yavaşlatmaktadır. Bu iki etki bu modellerin kötü performans vermesinde rol oynamıştır.

Naive Bayes ile yapılan testlerin sonucuna gelindiğinde ise doğruluk değerlerinin ve ortalamaların birbirine yakın değerlerde olduğu görülmekte ve standart sapmaların da benzer şekilde düşük olduğu görülmektedir. Ortalamaların ve doğruluk değerlerinin bu kadar yakın olması, sistemin yük testinde herhangi bir stres durumu yaşamadığını gösterir. İsteklerin yanıt ortalamalarından da anlaşılacağı üzere, isteklerin üretilme hızı tüketilme hızına göre daha yavaş kalmıştır ve sistem yeni gelen isteği hiç bekletmeden işleyebilmiştir.

7.4.2 Ani Yüklü Sistem Testi

Sisteme kısa sürede çok sayıda kullanıcının istek yapması durumunun taklit edildiği 100 kullanıcı, 1 saniye artırma süresi ve 10 tekrar ile gerçekleştirilen testin sonucunda farklı modeller için Çizelge 7.6 ve Çizelge 7.7'deki veriler elde edilmiştir.

Çizelge 7.6 : Karar ağacı modeli ile üretilen PMML dokümanları ile yapılan ani yüklü sistem testinin sonuçları

Veri Seti	İstek Sayısı	Hata Oranı (%)	Ortalama (ms)	Standart Sapma	Doğruluk (istek/dakika)
Adult Income	1000	0.0	25364	4657	224
Iris	1000	0.0	297	91	15317
Audit Risk	1000	0.0	22727	4227	250
Heart Disease	1000	0.0	968	271	5349

Çizelge 7.7 : Naive bayes modeli ile üretilen PMML dokümanları ile yapılan ani yüklü sistem testinin sonuçları

Veri Seti	İstek Sayısı	Hata Oranı (%)	Ortalama (ms)	Standart Sapma	Doğruluk (istek/dakika)
Adult Income	1000	0.0	728	217	6880
Iris	1000	0.0	227	76	17762
Audit Risk	1000	0.0	2871	552	1936
Heart Disease	1000	0.0	367	112	12295

Bu ölçümde sistemdeki bir uçbirime her 1 saniyede 100 kullanıcının aynı anda farklı parametrelerle tahmin isteği iletmesi gerçekleştirilmiştir.

Karar ağacı modelleri için sonuçlar incelendiğinde örüntü olarak normal yük testine benzediği söylenebilir. Normal yük testindeki sonuçlara benzer şekilde Adult Income ve Audit Risk modelleri kötü performans göstermişlerdir. Fakat birim zamanda daha çok isteğe maruz kaldıkları için ortalama yanıt süreleri normal yük testinin sonuçlarına göre de yüksek çıkmıştır. Bu sonuçlara göre karar ağacı modeli için sistemin stres seviyesi de ani yük testinde daha yüksek olduğu söylenebilir.

Naive Bayes modellerinin sonuçları incelendiğinde ortalama yanıt süresinin yükseldiği görülmektedir. Bunun yanında doğruluk değerlerinin yani dakikada ürettiği yanıt sayısının da büyük oranda arttığı görülmüştür. Sistem %0 hata oranıyla çalışarak tüm isteklere bir yanıt üretebilmiştir. Artırma süresinin 1 saniye oluşu test istemcisinin saniyede 100 istek üreterek sistemin tükettiğinden fazla isteğe maruz kalmasını sağlamış, bu durum doğruluk değerlerinde ayrışmalar oluşmasına sebep olmuştur. Bu

ölçümler sonunda sistemin ani yük altındayken belirli modellerde stres altında kalabildiği görülmüştür.

8. SONUÇLAR ve TARTIŞMA

Bu tez çalışmasında, ham verinin işlenmesi ve anlamlandırılması alanında çalışan profesyonellerin, eğittiği makine öğrenmesi modellerini herhangi bir kod geliştirmesi yapmadan son kullanıcıya ulaştırmasını kolaylaştırmak, ulaştırma yolunda literatürce kabul görmüş bir standarta yaklaştırmak hedeflenmiştir. Bunun yanında modellerin eğitilmesi aşamasında tecrübe sahibi olup, bir sunucu ortamında servis verir hale getirmede, ya da kısa söylemiyle entegrasyon aşamasında yetkin olmayan kişilerin de modellerini servis edebilmesini mümkün kılmak, böylece bu alanda yapılan çalışmaları da kolaylaştırıcı bir araç ortaya koymak da amaçlardan biridir. Tez çalışmasının ilk aşamasında bu ihtiyaçların tamamı göz önünde bulundurularak bir sistem tasarımı ortaya koyulmuştur.

Tasarlanan sistem, daha küçük modüllere bölünerek ve literatür araştırmalarından da faydalanılarak gerçeklenmiş ve çalışmanın amacına hizmet edebileceği yönünde bir görüş oluşmuştur. Bu aşamadan sonra tasarıma uygun olarak merkezi API uygulaması ve bunun kullanımını kolaylaştıran, kod bilgisi gerektirmeden model yayınlamaya aracı olan bir Web arayüzü geliştirilmiştir.

Sistemin geliştirilmesi tamamlandıktan sonra, amacına uygun şekilde hizmet verebilirliği kontrol edilmiştir. Bu kontrol süreci birkaç farklı ölçütle, literatürde sıkça kullanılan ölçüm araçları ile gerçekleştirilmiştir. Kontrollerin gerçekleştirilebilmesi için bazı kullanıcı senaryoları oluşturulmuştur. Bu senaryoların temel parçalarından olan veri setleri literatürdeki çalışmalarda yaygın olarak kullanılan başlıca veri setleri arasından belirlenmiştir. Bununla birlikte bu veri setleri yine en sık tercih edilen iki sınıflandırıcı algoritma ile eğitilmiş ve PMML dokümanlarına dönüştürülmüştür. Bu aşamadan sonra sistemin özellikleri test edilmiştir.

Üretilen PMML dokümanları geliştirilmiş olan web arayüzü yardımı ile API uygulamasına yüklenmiş, yine aynı arayüz yardımı ile isteklerin yapılacağı uçbirim adresleri elde edilmiştir. Web arayüzünün, yüklenmiş bir modele nasıl istek yapılacağı

ile ilgili yönlendirme yapması ile de herhangi bir kod yazma ihtiyacı duymadan, REST uyumlu bir uçbirime istek yapmanın mümkün olduğu görülmüştür. Bu yönüyle web arayüzünün tez çalışmasının amacına uygun olduğu söylenebilir.

Test senaryoları için belirlenen tüm veri setleri ve sınıflandırıcı algoritmaları ile üretilen PMML dokümanları boyut yönünden incelenmiştir. Sonrasında bu dokümanlar sisteme yüklenerek elde edilen uçbirimler üzerinden performans ölçümleri gerçekleştirilmiş ve bulgular toplanmıştır.

İlk ölçüm, bir modelin, web servisi üzerinden tahmin edilebilir hale getirilmesi işleminde, tahmin süresi olarak bir yavaşlamanın varlığını sorgulamıştır. Tüm sınıflandırıcı algoritma – veri seti varyasyonlarından elde edilen değerler incelendiğinde, modelin PMML'e dönüştürülmeden öngörülemesinin web servis üzerinden öngörülemesine kıyasla daha hızlı olduğu görülmüştür. Web servisindeki her öngörüleme işleminde HTTP'den doğan bir ek yük ve PMML dokümanının seri durumdan çıkarma işleminin getirdiği işlem yükü düşünüldüğünde bu sonuç beklenen bir sonuçtur. Değerlerdeki artış katsayı olarak yüksek görünse dahi son kullanıcıların hissedebileceği değerlerin altında kalmıştır. Bu yönüyle, modelin web servisi olarak servis edilmesinin getirdiği ek maliyetin, sağladığı faydaya kıyasla gözardı edilebilir seviyede kaldığı söylenebilir.

Veri setlerinin büyüklüğünün PMML dokümanlarının dosya boyutuna etkisinin ölçüldüğü ölçümde ise dosya büyüklüğünün seçilen model eğitim algoritmasına göre değiştiği gözlemlenmiştir. Örneğin karar ağacı sınıflandırıcısı ile üretilen modellerin PMML dosya boyutları veri setindeki satır sayısı ile doğru orantılı olarak artmıştır. Bunun yanında aynı veri setleri Naive Bayes Sınıflandırıcısı ile eğitildiğinde veri seti büyüklüğünün PMML dosya boyutunu etkilemediği görülmüştür. Modellerin doğruluğunda veri setinin büyüklüğünün de etkisi olduğu göz önünde bulundurulduğunda web servis dönüşümü yapılacak ise, veri setinden etkilenmeyen algoritmaların tercih edilmesi önerilebilmektedir.

Sistemin yük altındaki davranışlarını görmek amacıyla iki farklı yük testi yapılmıştır. İlk ölçümde düzenli olarak kullanıcı sayısı artırılmış fakat ani bir artış senaryosu uygulanmamıştır. Zamana yayılmış yoğun kullanım durumunun ölçümlenmek istendiği bu çalışmada elde edilen bulgular, sistemin böyle bir yük durumunda algoritmalara göre farklı sonuçlar ürettiğini göstermiştir. Büyük veri setlerinde karar ağacı modelinden üretilen PMML dokümanlarının öngörülemesi son kullanıcıların kabul edebileceği bekleme sürelerinin üstüne çıkmışken, aynı veri seti için Naive

Bayes modelinden üretilen PMML dokümanı öngörülemesi çok hızlı çalışmıştır. Dolayısıyla sistemin yük altındaki davranışı modelin eğitildiği sınıflandırıcının türüne bağlı olarak değişebilmektedir.

İkinci ölçümde ani yük durumunda sistemin davranışı incelenmiştir. Elde edilen bulgular, genel yük testinde, karar ağacı öngörülemesinin davranışının ani yükte artarak devam ettiğini göstermiştir. Naive Bayes sınıflandırıcının da ortalama tahmin sürelerinde artış gözlemlenmiş, fakat doğruluk değerleri de benzer şekilde artmıştır. Buradan sistemin Naive Bayes sınıflandırıcı modellerinde ani yüklerde de ortalama yanıt süresini 1 saniye altında tutmayı başararak son kullanıcıya yavaşlık hissettirmeyeceği sonucu çıkarılabilir. Bunun yanında ortalama yanıt süresi standart sapma değerlerindeki artış, ani yüklerde sistemin kararsızlık belirtileri göstermesi olarak yorumlanabilir.

Bulgular genel olarak değerlendirildiğinde, sistemin PMML oluşturulma aşamasından tahmin aşamasına kadar, amacına yönelik olarak kullanılabildiği görülmüştür. Performans yönünden incelendiğinde ise, kullanılan algoritmalara göre sistemin performansının değişebildiği görülmüştür. Bu noktada, özellikle zaman kritik uygulamalarda sistemin belirli algoritmalarda daha iyi sonuç vereceği söylenebilir.

Bu çalışma, bir makine öğrenmesi modelinin *RESTFul* web servisi üzerinden tahmin edilebilmesini sağlaması yönünden, bu alanda yapılacak benzer çalışmalar için bir ön çalışma niteliğindedir. Sistem bir model önerisi olmasının yanında dış erişime kapalı bir veri merkezinde çalışır halde bulunmaktadır. Gelecekte herkesin kullanımına açık hale getirilebilir. Mevcut yapıda, algoritma farklarından doğan performans değişimlerinin azaltılması için sunucu sistem kaynaklarının iyileştirilmesi, tek API uygulaması yerine birbirinin aynısı birden fazla API uygulama örneği çalıştırılarak bunların bir yük dengeleyici ile yük yönetimine tabi tutulması, sık kullanılan PMML dokümanlarının ön-bellek alanına taşınması gibi iyileştirmeler yapılabilir.

KAYNAKLAR

[1] **Roser M, Ritchie H, Ortiz-Ospina E.** (2015). Internet, *Our World Data*. Erişim Ocak 15, 2022. https://ourworldindata.org/internet

[2] **Alsop T**. (2021). Share of households with a computer at home worldwide from 2005 to 2019 Retrieved from https://www.statista.com/statistics/748551/worldwide-households-with-computer/

[3] **The rise of social media - Our World in Data.** (t.y.) Erişim Ocak 15, 2022. https://ourworldindata.org/rise-of-social-media

[4] **Sources of big data: Where does it come from? | CloudMoyo.** (t.y.) Erişim Ocak 15, 2022. https://www.cloudmoyo.com/blog/data-architecture/what-is-big-data-and-where-it-comes-from/

[5] **Jovanovic B.** (2021). 45 Fascinating IoT Statistics for 2021 | The State of the Industry. Erişim Ocak 15, 2022. https://dataprot.net/statistics/iot-statistics/

[6] **What is Transactional Data? | TIBCO Software.** Erişim Ocak 15, 2022. https://www.tibco.com/reference-center/what-is-transactional-data

[7] **Holst A.** (2021). Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2025. Published 2021. https://www.statista.com/statistics/871513/worldwide-data-created/

[8] **Foote KD.** (2018). A Brief History of Analytics - DATAVERSITY. Dataversity.Net. Published 2018. Erişim Ekim 11, 2021. https://www.dataversity.net/brief-history-analytics/

[9] **Grand View Research. Predictive Analytics Market Size & Share | Industry Report, 2019-2025.** (2019) Erişim Ekim 11, 2021. https://www.grandviewresearch.com/industry-analysis/predictive-analytics-market

[10] **Michelle K.** (t.y). What Is Predictive Analytics? - DATAVERSITY. Erişim Ocak 15, 2022. http://www.dataversity.net/what-is-predictive-analytics/

[11] **Kumar V, L. M.** (2018). Predictive Analytics: A Review of Trends and Techniques. *Int J Comput Appl*. 2018;182(1):31–37.

[12] **Hahn T, Nierenberg AA, Whitfield-Gabrieli S.** (2017). Predictive analytics in mental health: Applications, guidelines, challenges and perspectives. *Mol Psychiatry*. 2017;22(1):37–43.

[13] **Rizwan P, Suresh K, Rajasekhara Babu M.** (2017). Real-time smart traffic management system for smart cities by using Internet of Things and big data. *Proc IEEE Int Conf Emerg Technol Trends Comput Commun Electr Eng ICETT 2016*. Published online 2017:1–7.

[14] **Guazzelli A, Lin W-C, Jena T.** (2010). *PMML in action : unleashing the power of open standards for data mining and predictive analytics*. CreateSpace Independent Publishing

[15] **Sarabia-Jacome D, Lacalle I, Palau CE, Esteve M.** (2019). Efficient Deployment of Predictive Analytics in Edge Gateways: Fall Detection

Scenario. *IEEE 5th World Forum Internet Things, WF-IoT 2019 - Conf Proc*. Published online 2019:41–46.

[16] **Guha R.** (2008). Flexible Web service infrastructure for the development and deployment of predictive models. *J Chem Inf Model*. 2008;48(2):456–464

[17] **Grossman R, Bailey S, Ramu A, vd.** (2002). Management and mining of multiple predictive models using the predictive modeling markup language. *Inf Softw Technol*. 2002;41(9):589–595.

[18] **Guazzelli A, Stathatos K, Zeller M.** (2009). Efficient deployment of predictive analytics through open standards and cloud computing. *ACM SIGKDD Explor Newsl*. 2009;11(1):32–38.

[19] **Rathod D.** (2017). Performance Evaluation of Restful Web Services and Soap / Wsdl Web Services. *Int J Adv Res Comput Sci*. 2017;8(7):415–420.

[20] **Ferrucci D, Grossman RL, Levas A.** (2006). PMML and UIMA based frameworks for deploying analytic applications and services. *4th Int Work Data Min Stand Serv Platforms*. Published online 2006:14–26.

[21] **Chaves J, Curry C, Grossman RL, Locke D, Vejcik S.** (2006). Augustus: The design and architecture of a PMML-based scoring engine. *4th Int Work Data Min Stand Serv Platforms*. Published online 2006:38–46.

[22] **Guazzelli A, Zeller M, Lin WC, Williams G.** (2009). PMML: An open standard for sharing models. *R J*. 2009;1(1):60–65.

[23] **Pechter R.** (2009). What's PMML and what's new in PMML 4.0? *ACM SIGKDD Explor Newsl*. 2009;11(1):19–25.

[24] **Das KK, Fratkin E, Gorajek A, Stathatos K, Gajjar M.** (2011). Massively parallel in-database predictions using PMML. *Proc ACM SIGKDD Int Conf Knowl Discov Data Min*. Published online 2011:22–27.

[25] **Sottara D, Mello P, Sartori C, Fry E.** (2011). Enhancing a production rule engine with predictive models using PMML. *Proc ACM SIGKDD Int Conf Knowl Discov Data Min*. Published online 2011:39–47.

[26] **Guazzelli A, Guazzelli A, Jena T, vd.** (2011). The PMML Path towards True Interoperability in Data Mining. Published online 2011:1–7.

[27] **Ali R.** (t.y.) Predictive Modeling: Types, Benefits, and Algorithms | NetSuite. Erişim Ocak 15, 2022. https://www.netsuite.com/portal/resource/articles/financial-management/predictive-modeling.shtml

[28] **Predictive Model Markup Language Release History**.(t.y.) Erişim Ocak 15, 2022. https://en.wikipedia.org/wiki/Predictive_Model_Markup_Language#Release_history

[29] **Data Mining Group - PMML 4.4 - Changes from PMML 4.3.** (t.y.) Erişim Ocak 15, 2022. http://dmg.org/pmml/v4-4/Changes.html

[30] **Anderson E.** (1935). The irises of the Gaspe Peninsula. *Bull Am Iris Soc*. 1935;59:2–5.

[31] **Fisher RA.** (1936). The Use of multiple measurements in taxonomic problems. *Ann Eugen*. 1936;7(2):179–188. doi:10.1111/J.1469-1809.1936.TB02137.X

[32] **PMML 4.1 - Target Fields and Values.** (t.y.) Erişim Ocak 15, 2022. http://dmg.org/pmml/v4-1/Targets.html

[33] **Data Mining Group - PMML Powered.** (t.y.) Erişim Ekim 11, 2021. http://dmg.org/pmml/products.html

[34] **PMML 4.4.1 - General Structure.** (t.y.) Erişim Ekim 11, 2021. http://dmg.org/pmml/v4-4-1/GeneralStructure.html

[35] **Elliott T.** (2019). The State of the Octoverse: machine learning | The GitHub Blog. Erişim Ocak 15, 2022. https://github.blog/2019-01-24-the-state-of-the-octoverse-machine-learning/

[36] **Rayome AD.** (t.y.) Why Python is the real programming language of data science, not R - TechRepublic. Erişim Ocak 15, 2022. https://www.techrepublic.com/article/why-python-is-the-real-language-of-data-science-not-r/

[37] **Most Popular Backend Frameworks – 2012/2021 - New Update - Statistics and Data.** (2021). Erişim Ocak 15, 2022. https://statisticsanddata.org/data/most-popular-backend-frameworks-2012-2021/

[38] **Kamaruzzaman M.** (2021). Top 10 In-Demand Web Development Frameworks in 2021, *Towards Data Science.* Erişim Ocak 15, 2022. https://towardsdatascience.com/top-10-in-demand-web-development-frameworks-in-2021-8a5b668be0d6

[39] **Hamad H, Saad M, Abed R.** (2010). Performance evaluation of restful web services for mobile devices. *Int Arab J e-Technology*. 2010;1(3):72–78.

[40] **Fielding RT.** (2000). Architectural Styles and the Design of Network-based Software Architectures. Published online 2000.

[41] **Chervichnik A, Varaksina S.** (t.y.). Single-Page Applications vs Multi-Page Applications: The Battle of the Web Apps - Mind Studios. Erişim Ocak 15, 2022. https://themindstudios.com/blog/spa-vs-mpa/

[42] **Mikowski MS,** (2014). Powell JC. *Single page web applications : JavaScript end-to-end*. Manning; 2014.

[43] **Favell A.** (t.y.) The History of Git: The Road to Domination. Erişim Ocak 15, 2022. https://www.welcometothejungle.com/en/articles/btc-history-git

[44] **Git (yazılım) - Vikipedi.** (t.y.). Erişim Ocak 15, 2022. https://tr.wikipedia.org/wiki/Git_(yazılım)

[45] **About - Git.** (t.y.) Erişim Ocak 15, 2022. http://git-scm.com/about

[46] **Why Git | Atlassian Git Tutorial.** (t.y.). Erişim Ocak 15, 2022. https://www.atlassian.com/git/tutorials/why-git

[47] **What is Python? Executive Summary | Python.org.** (t.y.). Erişim Ocak 15, 2022. https://www.python.org/doc/essays/blurb/

[48] **Robinson D.** (t.y.). The Incredible Growth of Python | Stack Overflow. Erişim

Ocak 15, 2022. https://stackoverflow.blog/2017/09/06/incredible-growth-python/

[49] **Millman KJ, Aivazis M.** (2011). Python for scientists and engineers. *Comput Sci Eng*. 2011;13(2):9–12.

[50] **Url-1** < https://www.pmml4s.org/>, Erişim Ocak 15, 2022.

[51] **Foreword — Flask Documentation (2.0.x).** (2010). Erişim Ocak 15, 2022. https://flask.palletsprojects.com/en/2.0.x/foreword/#what-does-micro-mean

[52] **Liu S.** (t.y.). Most used web frameworks among developers 2021 | Statista. Published 2021. Erişim Ocak 15, 2022. https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/

[53] **Cebeci C.** (2019). API İstemcisi Doğrulama Yöntemleri | Medium. Published 2019. Erişim Ocak 15, 2022. https://medium.com/@thejengo/api-istemcisi-doğrulama-yöntemleri-c29e3826daea

[54] **Huang XW, Hsieh CY, Wu CH, Cheng YC.** (2015). A token-based user authentication mechanism for data exchange in RESTful API. *Proc - 2015 18th Int Conf Network-Based Inf Syst NBiS 2015*. Published online 2015:601–606.

[55] **Ethelbert O, Moghaddam FF, Wieder P, Yahyapour R.** (2017). A JSON token-based authentication and access management schema for cloud SaaS applications. *Proc - 2017 IEEE 5th Int Conf Futur Internet Things Cloud, FiCloud 2017*. 2017;2017-Janua:47–53.

[56] **Yıldız E, Bilgin TT.** (2021). Performance Comparison of Model Storage Formats for Deploying Data Mining Models. Içinde: *International Conference on Engineering Technologies*. ; 2021:72–77.

[57] **Bostan E.** (t.y.). ORM (Object-Relational Mapping) nedir? | emrebostan.com. Erişim Ocak 15, 2022. https://www.emrebostan.com/orm-object-relational-mapping-nedir-nasil-kullanilir/

[58] **Colley D, Stanier C, Asaduzzaman M.** (2018). The Impact of Object-Relational Mapping Frameworks on Relational Query Performance. *Proc - 2018 Int Conf Comput Electron Commun Eng iCCECE 2018*. Published online 2019:47–52.

[59] **Object-relational Mappers (ORMs) - Full Stack Python.** (t.y.). Erişim Ocak 15, 2022. https://www.fullstackpython.com/object-relational-mappers-orms.html

[60] **Harrington JL.** (2016). *Relational database design and implementation*. 4. baskı.; Morgan Kauffmann;

[61] **Liu S.** (2022). Most popular database management systems 2022 | Statista. Published 2022. Erişim Ocak 15, 2022. https://www.statista.com/statistics/809750/worldwide-popularity-ranking-database-management-systems/

[62] **Lee TB.** (1989). *Information Management: A Proposal | ELMCIP*.; 1989. Erişim Ocak 15, 2022. https://www.w3.org/History/1989/proposal.html

[63] **HTML - Vikipedi.** (t.y.). Erişim Ocak 15, 2022.

https://tr.wikipedia.org/wiki/HTML

[64] **Macrae C.** (2018). *Vue.js: Up and Running : Building Accessible and Performant Web Apps.*; 2018.

[65] **Url-2** <https://vuejs.org/v2/guide/> Erişim Ocak 15, 2022.

[66] **Url-3** <https://next.router.vuejs.org/>Home. Erişim Ocak 15, 2022.

[67] **The State of CSS 2019: CSS Frameworks.** (2019). Erişim Ocak 15, 2022. https://2019.stateofcss.com/technologies/css-frameworks/

[68] **Aksan CE.** (t.y.). Paket Yöneticisi Nedir? Neden İhtiyaç Duyulur? | ceaksan | Web Analisti & Veri Bilimi Meraklısı. Erişim Ocak 15, 2022. https://ceaksan.com/tr/paket-yoneticisi

[69] **Apache Mod Python : List of security vulnerabilities.** (t.y.). Erişim Ocak 18, 2022. https://www.cvedetails.com/vulnerability-list/vendor_id-45/product_id-1882/Apache-Mod-Python.html

[70] **PEP 333 -- Python Web Server Gateway Interface v1.0 | Python.org.** (t.y.). Erişim Ocak 18, 2022. https://www.python.org/dev/peps/pep-0333/

[71] **WSGI Servers - Full Stack Python.** (t.y.). Erişim Ocak 18, 2022. https://www.fullstackpython.com/wsgi-servers.html

[72] **Grant C. Mobile API response time matters: 4 steps to speed up | TechBeacon.** (t.y.). Erişim Ocak 18, 2022. https://techbeacon.com/app-dev-testing/mobile-api-response-time-matters-4-steps-speed

[73] **Kumari S, Rath SK.** (2015). Performance comparison of SOAP and REST based Web Services for Enterprise Application Integration. *2015 Int Conf Adv Comput Commun Informatics, ICACCI 2015*. Published online 2015:1656–1660.

[74] **Hooda N, Bawa S, Rana PS.** (2018). Fraudulent Firm Classification: A Case Study of an External Audit. *Appl Artif Intell*. 2018;32(1):48–64.

[75] **Detrano R, Janosi A, Steinbrunn W, vd.** (1989). International application of a new probability algorithm for the diagnosis of coronary artery disease. *Am J Cardiol*. 1989;64(5):304–310.

[76] **Aha DW, Kibler D.** (t.y.) Instance-based prediction of heart-disease presence with the Cleveland database.

[77] **Gennari JH, Langley P, Fisher D.** (1989). Models of incremental concept formation. *Artif Intell*. 1989;40(1–3):11–61.

[78] **Kovahi R.** (1996). Scaling Up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid. *Proc Second Int Conf Knowl Discov Data Min*. Published online 1996.

[79] **Url-4** <https://jmeter.apache.org/> Erişim Ekim 11, 2021. https://jmeter.apache.org/

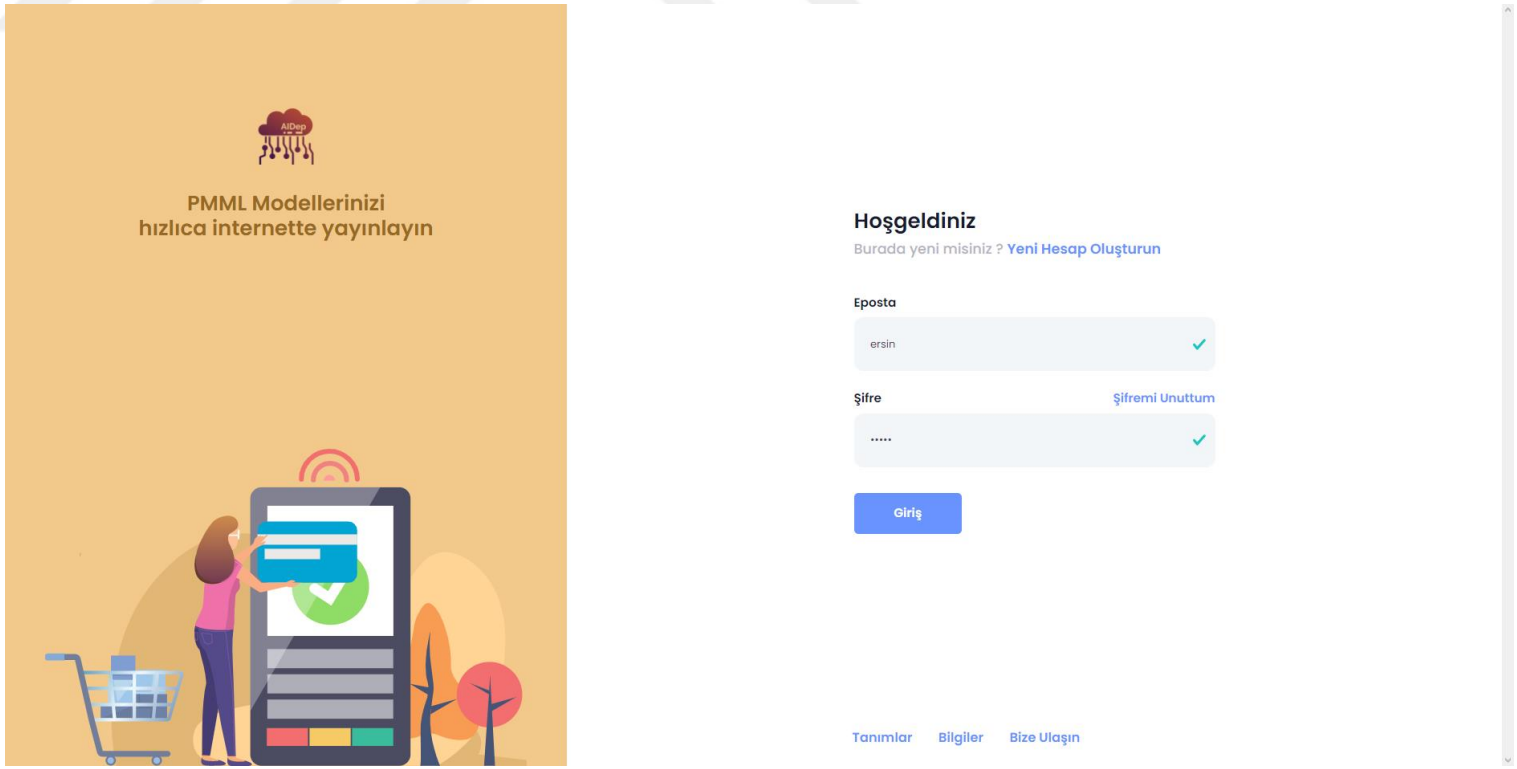
[80] **Vashista A.** (2012). Automation Performance Testing: Jmeter:Understanding Summary Report. Published 12 Ağustos 2012. Erişim Ocak 29, 2022. https://automation-performance.blogspot.com/2015/08/jmeterunderstanding-summary-report.html

[81] **Url-5** <https://hub.knime.com/knime/extensions/org.knime.features.base/latest/org.knime.base.node.util.timerinfo.TimerinfoNodeFactory> Erişim Ocak 19, 2022.

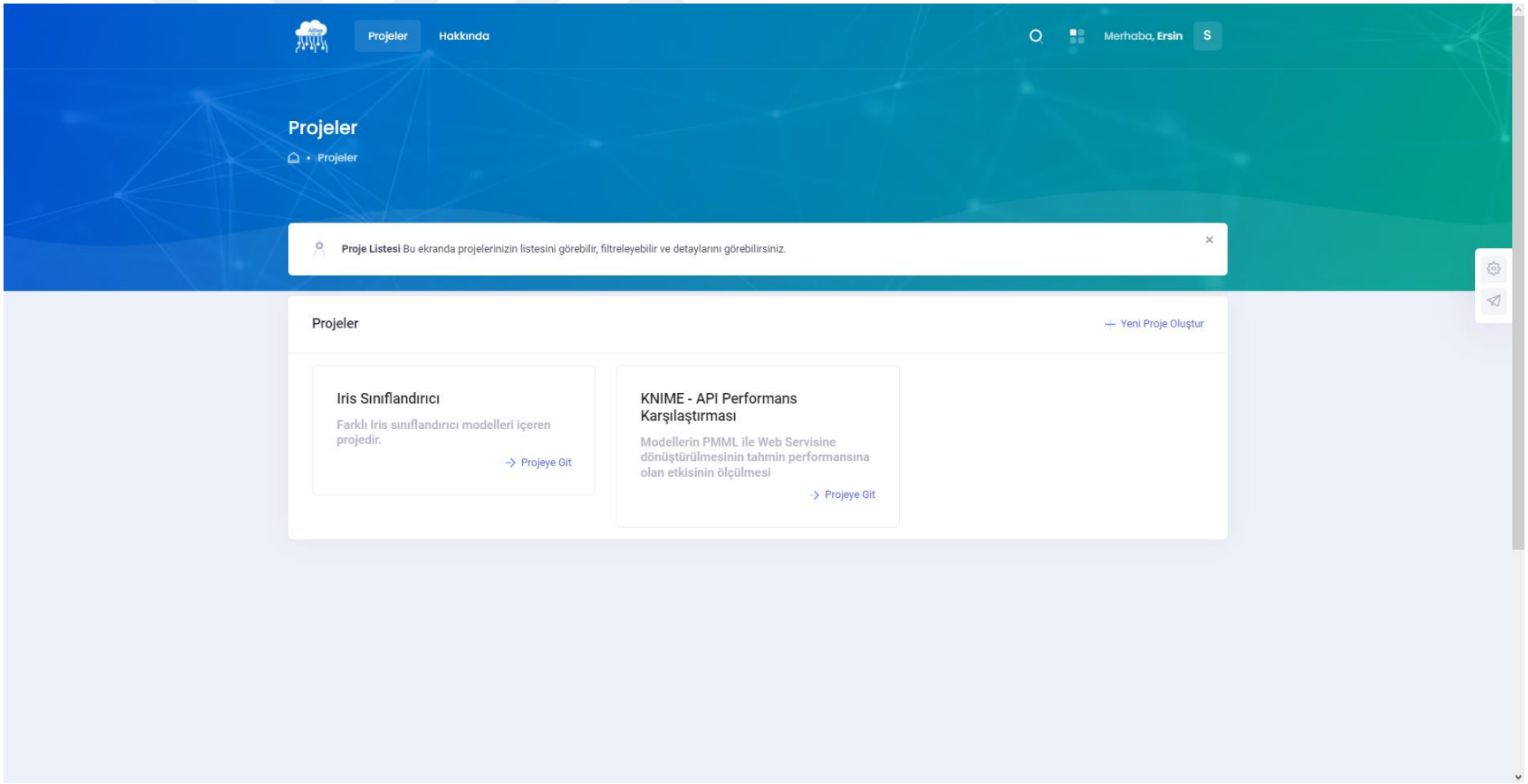
[82] **Nielsen J.** (1993)*Usability engineering*. Academic Press; 1993.

EKLER

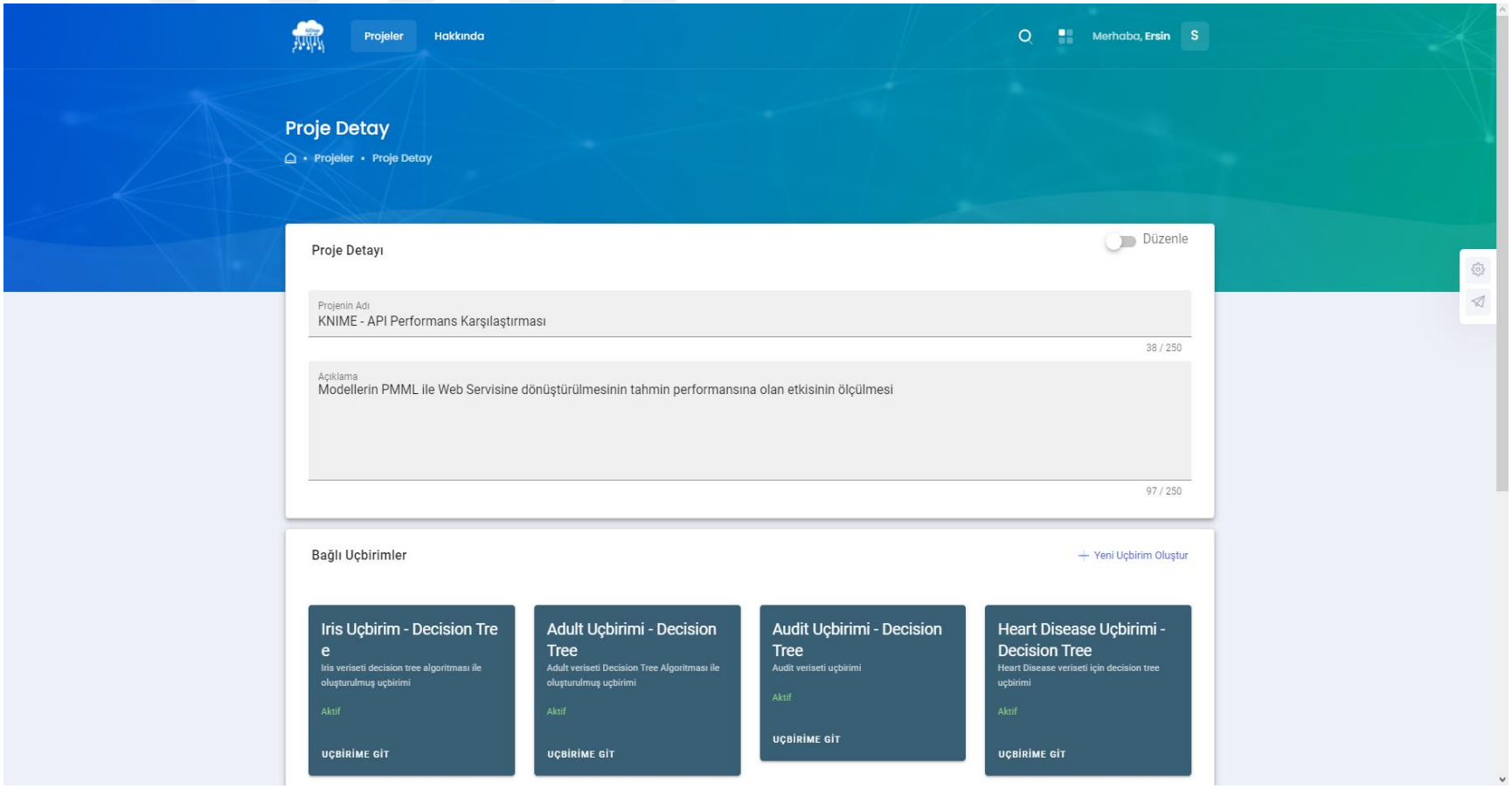
EK A : Web arayüzünün örnek ekran görüntüleri



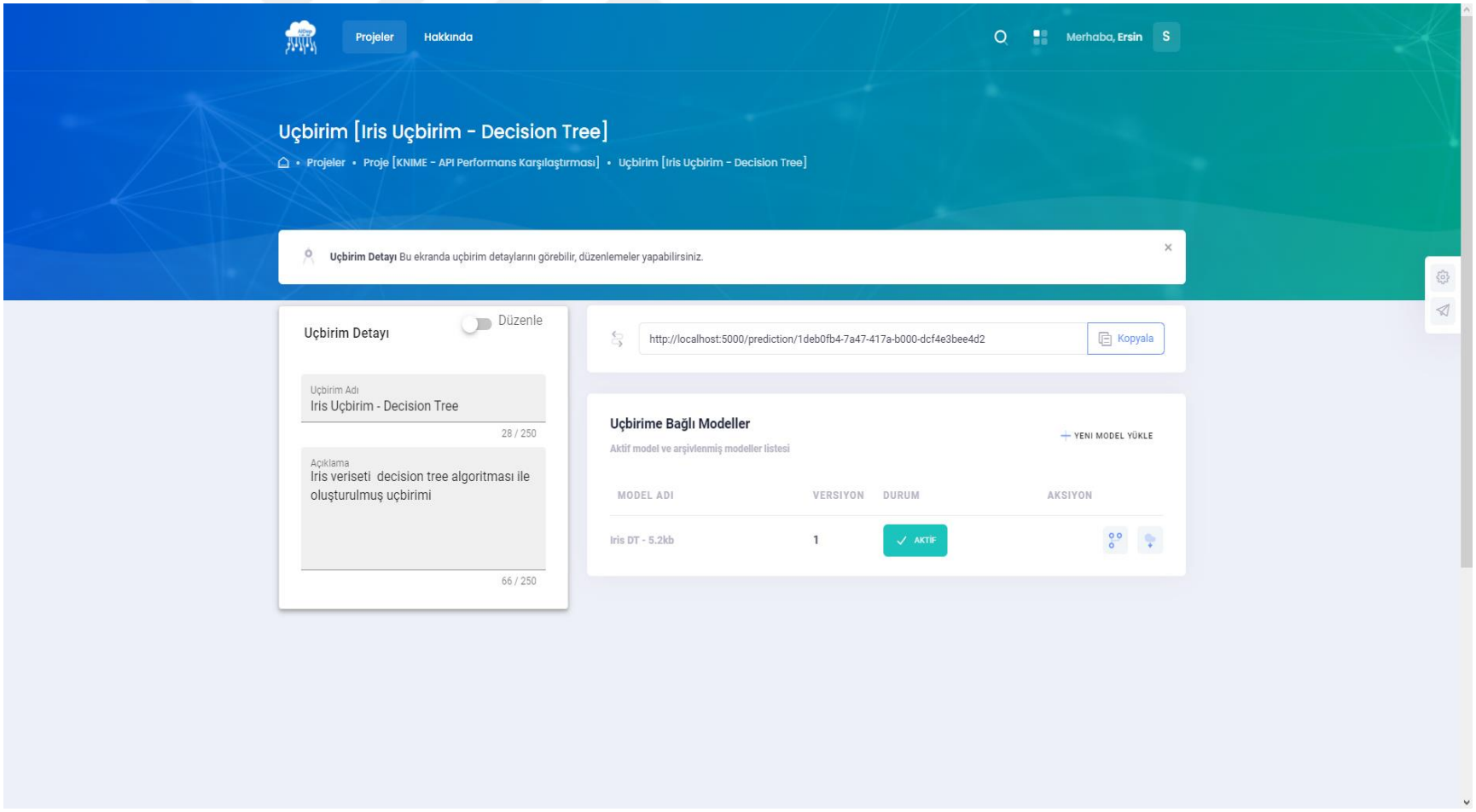
Şekil A. 1 : Kullanıcı giriş ekranının örnek görüntüsü



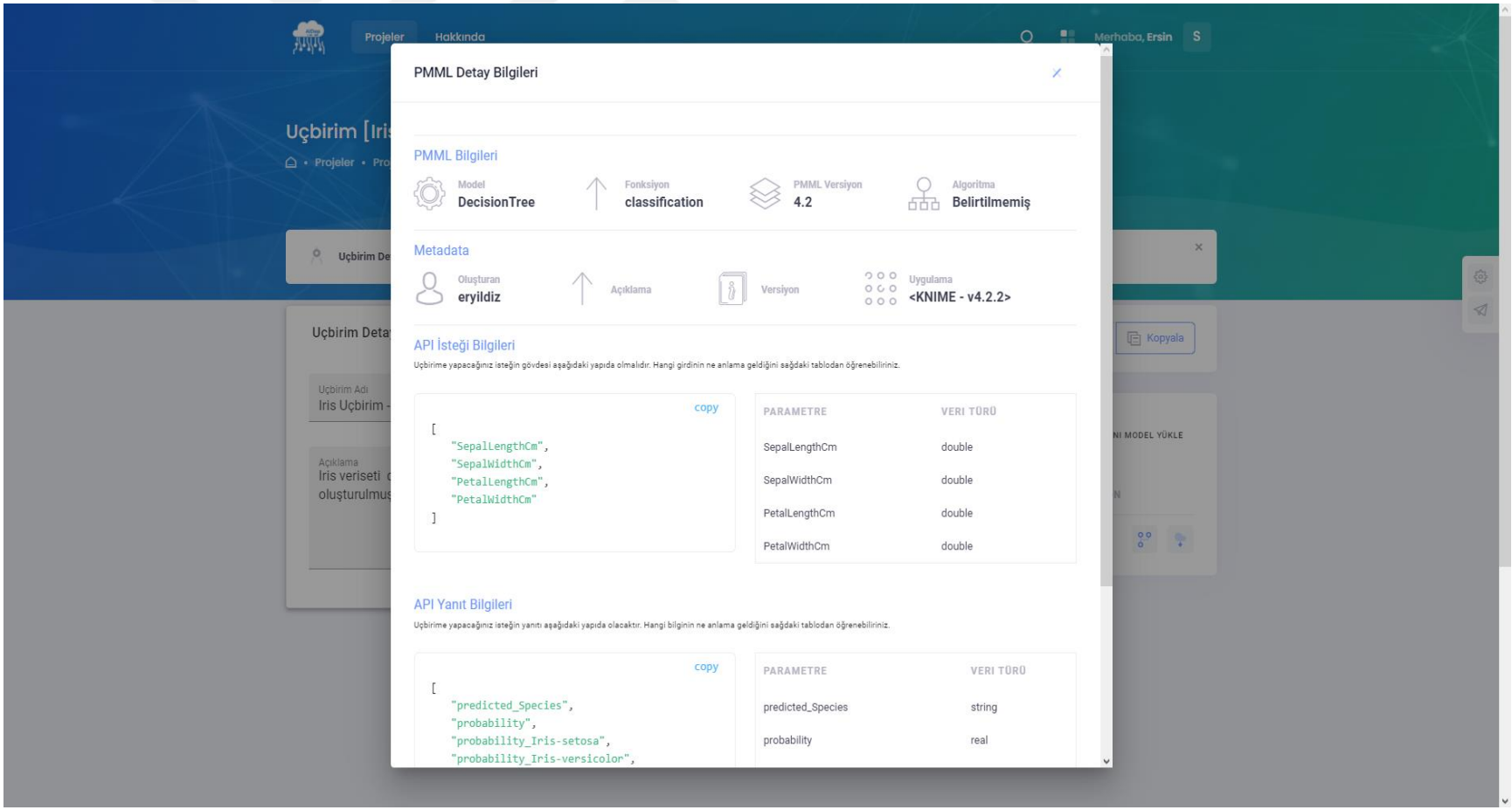
Şekil A.2 : Projeler ekranına ait örnek ekran görüntüsü



Şekil A.3: Bir projenin detay ekranının örnek ekran görüntüsü

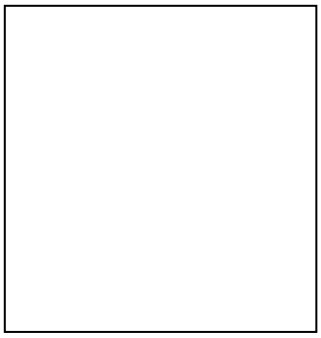


Şekil A.4 : Bir uçbirimin detay ekranının örnek ekran görüntüsü



Şekil A.5 : Uçbirime yüklenmiş bir PMML modelin detaylarının görüntülenmesine ait ekran görüntüsü

ÖZGEÇMİŞ



Ad-Soyad : Ersin YILDIZ

Doğum Tarihi ve Yeri :

E-posta :

ÖĞRENİM DURUMU:

- **Lisans :** 2013, Karadeniz Teknik Üniversitesi, Mühendislik ve Mimarlık Fakültesi, Bilgisayar Mühendisliği Bölümü

MESLEKİ DENEYİM VE ÖDÜLLER:

- 2017 - ... CITS Bilişim Hizmetleri – Yazılım Geliştirme Uzmanı ve Ar-Ge Mühendisi
- 2015-2017 Nota Yazılım – Yazılım Geliştirme Uzmanı

TEZDEN TÜRETİLEN ESERLER, SUNUMLAR VE PATENTLER:

- Yıldız E. and Bilgin T.T., "Performance Comparison of Model Storage Formats for Deploying Data Mining Models," in *International Conference on Engineering Technologies*, 2021, pp. 72–77