SÜREÇ MADENCİLİĞİNDE TAHMİNE DAYALI SÜREÇ İZLEME TEKNİKLERİNİN KALİTESİNİN AÇIKLANABİLİR YAPAY ZEKA YÖNTEMLERİ İLE DEĞERLENDİRİLMESİ

EVALUATION OF THE QUALITY OF PREDICTIVE PROCESS MONITORING TECHNIQUES IN PROCESS MINING WITH EXPLAINABLE ARTIFICIAL INTELLIGENCE METHODS

SAMET CAN

DOÇ. DR. AYÇA KOLUKISA TARHAN

Tez Danışmanı

DR. ÖĞR. ÜYESİ TUĞBA ERDOĞAN

Eş Danışman

Hacettepe Üniversitesi

Lisansüstü Eğitim-Öğretim ve Sınav Yönetmeliğinin

Bilgisayar Mühendisliği Anabilim Dalı için Öngördüğü

YÜKSEK LİSANS TEZİ olarak hazırlanmıştır.

Eylül 2024

ÖZET

SÜREÇ MADENCİLİĞİNDE TAHMİNE DAYALI SÜREÇ İZLEME TEKNİKLERİNİN KALİTESİNİN AÇIKLANABİLİR YAPAY ZEKA YÖNTEMLERİ İLE DEĞERLENDİRİLMESİ

Samet CAN

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Danışmanı: Doç. Dr. Ayça KOLUKISA TARHAN
Eş Danışman: Dr. Öğr. Üyesi Tuğba ERDOĞAN
Eylül 2024, 94 sayfa

Süreç madenciliğinde kullanılan tahmine dayalı süreç izleme (PPM) teknikleri; sürecin sonucu, yürütülecek bir sonraki etkinlik veya kalan çalışma süresi gibi, bir sürecin gelecekteki özelliklerini tahmin etmeyi amaçlamaktadır. Mevcut birçok PPM yaklaşımı, makine öğrenimi (ML) tabanlı tahmin modellerini benimsemektedir. ML modellerinin artan verimliliği ve doğruluğu genellikle artan karmaşıklıkla birleştiğinde, bu modellerin anlaşılabilirliği tehlikeye girer. Açıklanabilir yapay zeka (XAI) yöntemleri, kullanıcılara bir ML modelinin mantık yürütme sürecine ilişkin açıklamalar sağlamak amacıyla ortaya çıkmıştır. Bununla birlikte, PPM kapsamında yapılan seçimler ve kullanılan teknikler de ortaya çıkan açıklamaları etkilemektedir. Bu çalışma, tahmine dayalı süreç izleme alanında açıklanabilir yapay zekanın temel bir anlayışını oluşturmak için literatürü sentezlemektedir. Kullanılan tahmin modellerinin türleri, uygulanan XAI teknikleri ve XAI tekniklerinin ampirik değerlendirme ve geçerliliği analiz edilmektedir. Çalışma, son

kullanıcılar için açıklanabilirliğin önemini, metinsel verilerin tahmine dayalı modelleri zenginleştirme potansiyelini ve hem kavramsal hem de ampirik çerçevelerdeki gelişmeleri göstererek PPM'de XAI'nin gelişen yönlerini vurgulamaktadır. Elde edilen içgörüler, daha yorumlanabilir ve güvenilir yapay zeka sistemlerinin tasarlanması için bilgi sağlayabilir ve bu sistemlerin kritik iş süreci yönetimi uygulamalarında benimsenmelerini kolaylaştırabilir.

Bu çalışmada, literatür sentezlendikten sonra elde edilen bilgiler ışığında XAI destekli tahmine dayalı süreç izleme adımları oluşturulmuştur. Vaka çalışması kapsamında bu adımlar kullanılarak seçilen bir olay günlüğüne belirlenen tahmine dayalı süreç izleme teknikleri uygulanarak performans metrikleriyle tahminlerin kalitesi ölçülmüş ve belirlenen açıklanabilir yapay zeka yöntemleriyle açıklamalar yapılmıştır. Daha sonra performans metriklerinin sonuçları ve açıklamalarla birlikte, modeller değerlendirilerek hiperparametre optimizasyonu ile gerekli iyileştirmeler yapılmış ve elde edilen sonuçlar paylaşılmıştır. Bu çalışma, tahmine dayalı süreç izleme modellerinin hazırlanması, eğitilmesi, değerlendirilmesi ve XAI yöntemleriyle açıklamaların oluşturulması konularında kapsamlı bir bilgi sağlamaktadır.

Keywords: Süreç Madenciliği, Tahmine Dayalı Süreç İzleme, Açıklanabilir Yapay Zeka, Sistematik Literatür Tarama

ABSTRACT

EVALUATION OF THE QUALITY OF PREDICTIVE PROCESS MONITORING TECHNIQUES IN PROCESS MINING WITH EXPLAINABLE ARTIFICIAL INTELLIGENCE METHODS

Samet CAN

Master of Science, Department of Computer Engineering
Supervisor: Assoc. Prof. Dr. Ayça KOLUKISA TARHAN
2nd Supervisor: Assist. Prof. Dr. Tuğba ERDOĞAN
September 2024, 94 pages

Predictive process monitoring (PPM) techniques used in process mining aim to predict future attributes of a process, such as process outcome, next activity to be executed, or remaining work time. Many existing PPM approaches utilize machine learning (ML) based prediction models. The increasing efficiency and accuracy of ML models is often combined with increasing complexity, compromising the understandability of these models. Explainable artificial intelligence (XAI) methods have emerged to provide users with explanations of the reasoning process of an ML model. However, the choices made and the techniques used within the scope of PPM also affect the resulting explanations. This study synthesizes the literature to construct a foundational understanding of explainable artificial intelligence in the field of predictive process monitoring. The types of predictive models used, XAI techniques applied, and the empirical evaluation and validation of XAI techniques are analyzed. The study highlights the evolving aspects of XAI in PPM, demonstrating the importance of

explainability to end-users, the potential of textual data to enrich predictive models, and advancements in both conceptual and empirical frameworks. The insights obtained can inform the design of more interpretable and trustworthy artificial intelligence systems and facilitate the adoption of these systems in critical business process management applications.

In this study, XAI supported predictive process monitoring steps were derived in the light of the information obtained after the literature was synthesized. Within the scope of the case study, by applying the determined predictive process monitoring techniques to a selected event log using these steps, the quality of the predictions was measured with performance metrics and explanations were made with the determined explainable artificial intelligence methods. Then, together with the results of the performance metrics and explanations, the models were evaluated and necessary improvements were made with hyperparameter optimization and the obtained results were shared. This study provides comprehensive information on the preparation, training, evaluation of predictive process monitoring models and the creation of explanations with XAI methods.

Keywords: Process Mining, Predictive Process Monitoring, Explainable Artificial Intelligence, Systematic Literature Review

TEŞEKKÜR

Tez konusunun belirlenmesinde yol gösteren, tez metninin yazımı ve tez çalışmasının hazırlanması sürecinde değerli katkılarını sunan ve her daim desteklerini esirgemeyen kıymetli hocalarım ve tez danışmanlarım Sayın Doç. Dr. Ayça KOLUKISA TARHAN'a ve Sayın Dr. Öğr. Üyesi Tuğba ERDOĞAN'a,

Tez savunma sınavım sırasında değerli önerileriyle bana katkı sağlayan ve tez metnini inceleyerek içeriğinin son halini almasına yardımcı olan kıymetli hocalarım Sayın Doç. Dr. Oumout CHOUSEINOGLOU'na ve Sayın Doç. Dr. Ali Seydi KEÇELİ'ye,

Hayatımın her aşamasında maddi ve manevi olarak hep yanımda olan, sevgi ve desteklerini eksik etmeyen kıymetli aileme,

En içten teşekkürlerimi sunarım.

İÇİNDEKİLER

Sayfa

ŞEKİLLER

ÇİZELGELER

Sayfa

SİMGELER VE KISALTMALAR

AI	:	**A**rtificial **I**ntelligent Yapay Zeka
BPIC	:	**B**usiness **P**rocess **I**ntelligence **C**hallenge İş Süreci Zekası Yarışması
CNN	:	**C**onvolutional **N**eural **N**etwork Evrişimsel Sinir Ağı
DARPA	:	**D**efense **A**dvanced **R**esearch **P**roject **A**gency Savunma İleri Araştırma Projeleri
DL	:	**D**eep **L**earning Derin Öğrenme
DNN	:	**D**eep **N**eural **N**etwork Derin Sinir Ağı
GAN	:	**G**enerative **A**dverserial **N**etwork Üretken Çatışmalı Ağ
GGNN	:	**G**ated **G**raph **N**eural **N**etwork Geçitli Grafik Sinir Ağı
GRU	:	**G**ated **R**ecurrent **U**nit Geçitli Yinelemeli Birim
IEEE	:	**I**nstitute of **E**lectrical and **E**lectronics **E**ngineers Elektrik ve Elektronik Mühendisleri Enstitüsü
IT	:	**I**nformation **T**echnology Bilgi Teknolojileri
LIME	:	**L**ocal **I**nterpretable **M**odel-Agnostic **E**xplanations Yerel Yorumlanabilir Model Bağımsız Açıklamalar
LRP	:	**L**ayer-wise **R**elevance **P**ropagation Katman Bazında İlgililik Yayılımı

LSTM	:	**L**ong **S**hort-**T**erm **M**emory
		Uzun Kısa Süreli Bellek
ML	:	**M**achine **L**earning
		Makine Öğrenimi
NN	:	**N**eural **N**etwork
		Sinir Ağı
PFI	:	**P**ermutation **F**eature **I**mportance
		Permütasyon Özelliği Önemi
PPM	:	**P**redictive **P**rocess Monitoring
		Tahmine Dayalı Süreç İzleme
RNN	:	**R**ecurrent **N**eural **N**etwork
		Yinelemeli Sinir Ağı
SHAP	:	**SH**apley **A**dditive ex**P**lanations
		Shapley Katkı Açıklamaları
SLR	:	**S**ystematic **L**iterature **R**eview
		Sistematik Literatür Tarama
SVM	:	**S**upport **V**ector **M**achine
		Destek Vektör Makinesi
XAI	:	E**x**plainable **A**rtificial **I**ntelligence
		Açıklanabilir Yapay Zeka
XES	:	e**X**tensible **E**vent **S**tream
		Genişletilebilir Olay Akışı
XGBoost	:	e**X**treme **G**radient **Boost**ing
		Aşırı Gradyan Artırma

1. GİRİŞ

Açıklanabilir yapay zeka (Explainable Artificial Intelligence - XAI), çeşitli alanlarda tahmine dayalı analitik için karmaşık makine öğrenimi (Machine Learning - ML) modellerinin yaygın olarak benimsenmesi nedeniyle son yıllarda büyük ilgi görmüştür [3]. Tahmine dayalı modellerin giderek daha fazla kullanıldığı bu alanlardan biri, tahmine dayalı süreç izleme (Predictive Process Monitoring - PPM) gibi tekniklerin iş süreci yürütmelerinin gelecekteki durumlarını tahmin etmeyi amaçladığı süreç izlemedir [4, 5]. Bu tahmine dayalı modeller bir yandan yüksek doğruluk gösterirken diğer yandan, karmaşık kara kutu modellerin açıklanabilirlik ve yorumlanabilirlik eksikliğine ilişkin endişeler devam etmektedir [4, 6].

Bu sorunu ele almak ve model şeffaflığını artırmak için, tahmine dayalı süreç izleme bağlamında farklı XAI yöntemleri önerilmiş ve uygulanmıştır [4]. Bununla birlikte, mevcut çalışmaların çoğu, önceden eğitilmiş tahmine dayalı modeller üzerinde modelden bağımsız açıklama yöntemlerini kullanarak post-hoc (sürecin tamamlanmasından sonra) açıklanabilirliğe odaklanmaktadır [4, 6]. Birkaç çalışma da alana özgü kural tabanlı açıklama yaklaşımlarını araştırmıştır, ancak bunlar uygulama alanı hakkında uzman bilgisi gerektirmektedir [7]. Ayrıca, tahmine dayalı süreç izleme için önerilen XAI yöntemlerinin ampirik değerlendirmesi ve geçerlemesi sınırlıdır [3, 7].

Bu çalışmada, tahmine dayalı süreç izleme için açıklanabilir yapay zeka yöntemlerinin uygulanmasına yönelik araştırmalara sistemetik literatür taraması (Systematic Literature Review - SLR) çalışmasıyla [8] kapsamlı bir genel bakış sağlanmıştır. Araştırma ve katkı türleri, PPM için XAI uygulamalarının temaları, PPM için kullanılan makine öğrenimi veya derin öğrenme modelleri, PPM için kullanılan XAI yöntemleri ile uygulama ve değerlerlendirme detayları ele alınmıştır.

Daha sonra literatüre dayalı olarak XAI destekli tahmine dayalı süreç izleme adımları oluşturulmuş ve vaka çalışması kapsamında karayolu trafik cezası yönetim süreçlerini içeren bir olay günlüğüne bu adımlar sırasıyla uygulanmıştır. Tahmine dayalı süreç

izleme tekniklerinden XGBoost ve LSTM modelleri kullanılarak performans metrikleriyle tahminlerin kalitesi ölçülmüştür. Bu modeller için açıklanabilir yapay zeka yöntemlerinden XGBoost modeline SHAP ve LSTM modeline dikkat mekanizması uygulanarak açıklamalar oluşturulmuştur.

Elde edilen performans metriklerinin sonuçları ve oluşturulan açıklamalar ile birlikte kullanılan PPM tekniklerinin kalitesi değerlendirilmiştir. Daha sonra bu modellere hiperparametre optimizasyon işlemi uygulanarak optimum hiperparametreler belirlenmiştir. Bu hiperparametrelerle birlikte açıklamalar ele alınarak PPM tekniklerinde gerekli iyileştirmeler yapılmış ve elde edilen bulgular analiz edilmiştir. Bu bulgular, tahmine dayalı süreç izleme için XAI alanındaki açık zorlukları ve fırsatları belirleyecek ve bu alanda gelecekteki araştırmalar için bir temel sağlayacaktır.

Bu tez çalışmasının diğer kısımları şu şekilde organize edilmiştir:

- İkinci bölümde, süreç madenciliği, tahmine dayalı süreç izleme, makine öğrenimi ve açıklanabilir yapay zeka konularıyla ilgili olarak literatürde yer alan genel bilgilerden bahsedilmiştir.
- Üçüncü bölümde, ilgili ikincil çalışmalar ve sistematik literatür taraması hakkında genel bir bakış sunulmuş ve ilgili çalışmaların katkıları açısından bu araştırmaya duyulan ihtiyaç vurgulanmıştır. Daha sonra sistematik inceleme süreci, arama stratejisi ve çalışma seçimi, çalışma kalitesi değerlendirmesi, veri çıkarımı ve sentezi ile potansiyel geçerlilik tehditleri dahil olmak üzere, bu araştırmayı gerçekleştirirken kullanılan metodoloji açıklanmıştır. Ayrıca çıkarılan araştırma sorularına uygun olarak araştırma sonuçları da bu bölümde sunulmuştur.
- Dördüncü bölümde, literatürden beslenerek oluşturulan XAI destekli tahmine dayalı süreç izleme adımları verilmiştir.
- Beşinci bölümde, vaka çalışması kapsamında ele alınan bir olay günlüğüne XAI destekli tahmine dayalı süreç izleme adımları uygulanarak elde edilen sonuçlarla PPM tekniklerinin kalitesi değerlendirilmiştir.

- Altıncı bölümde, vaka çalışması kapsamında karşılaşılan zorluklar ve gelecekteki çalışma planları da dahil olmak üzere diğer konular tartışılmış ve sonuçlara yer verilmiştir.

2. GENEL BİLGİLER

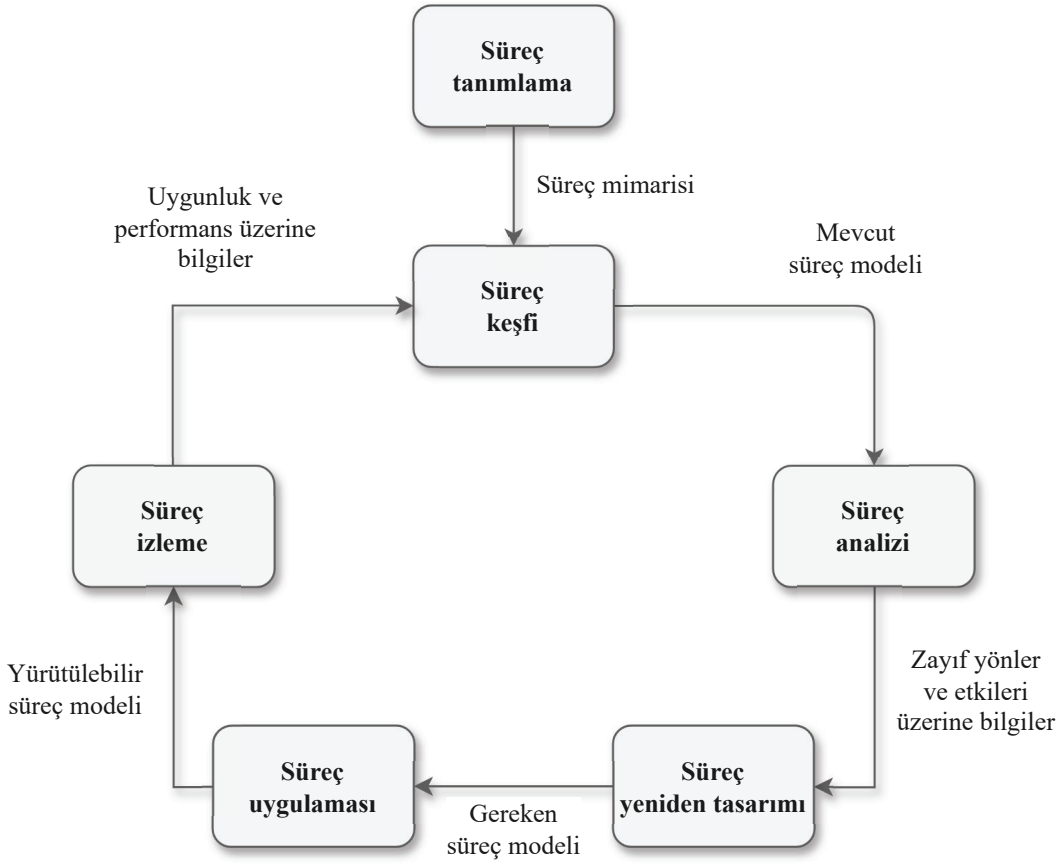
2.1. Süreç Madenciliği

Süreç madenciliği, İş Süreçleri Yönetimi (Business Process Management - BPM) ile veri biliminin kesiştiği noktada bulunan bir bilim alanıdır. Süreç madenciliği, bir sürecin etkinliklerinin nasıl gerçekleştirilmesi gerektiğine dair ideal görüşün mevcut durumla karşılaştırılmasına olanak tanır. BPM, bir şirketin iş akışlarının, süreçlerinin ve operasyonlarının performansını ve verimliliğini artırmaya yönelik sistematik bir yaklaşımdır [1]. Verimliliği optimize etmek, maliyetleri azaltmak, müşteri memnuniyetini artırmak ve organizasyonel hedeflere ulaşmak için iş süreçlerinin tanımlanmasını, tasarlanmasını, uygulanmasını, izlenmesini ve iyileştirilmesini içerir.

BPM, iş akışlarını haritalandırarak ve iyileştirilecek alanları belirleyerek iş operasyonlarını kolaylaştırmayı amaçlamaktadır. Bu, iş süreçlerinin mevcut durumunun analiz edilmesini ve anlaşılmasını, verimsizliklerin belirlenmesini ve bunları optimize etmek için bir plan geliştirilmesini içerir. BPM ayrıca manuel emeği, hataları ve maliyetleri azaltmak için teknolojiyi kullanarak belirli süreçlerin otomatikleştirilmesini de içerebilir.

Bu bağlamda, bir iş süreci; bir dizi insan aktörü, yazılım sistemi ve fiziksel ve dijital nesneleri içeren, birbirine bağlı olaylar, etkinlikler ve karar noktalarından oluşan bir bütün olarak görülür ve bu sürecin sonucunda ilgili aktörlere değer katan bir çıktı elde edilir. Şekil 3.1'de aşamaları gösterilen BPM faaliyetleri şu şekilde özetlenebilir:

- **Süreç tanımlama:** Ele alınan problemle ve ilgili iş hedefleriyle ilgili kritik süreçlerin tanımlanmasını içerir. Bu aşamanın sonucu, tanımlanan süreçlere ve bunların ilişkilerine genel bir bakış sunan bir süreç mimarisidir.
- **Süreç keşfi:** Bir organizasyonun iş sürecinin mevcut durumunun bir örneğini oluşturmayı içerir.



Şekil 2.1 BPM yaşam döngüsü modeli [1]

- **Süreç analizi:** Mevcut süreçle ilgili olası sorunları belirlemek ve böylece olası iyileştirmeleri keşfetmek amacıyla bir süreci incelemeyi içerir.
- **Süreç yeniden tasarımı:** Önceden belirlenen sorunları ele almak için mevcut süreçteki çeşitli potansiyel değişiklikleri belirlemeyi içerir. Bu işlemin sonucu, olması gereken süreç modelinin tanımlanmasıdır.
- **Süreç uygulaması:** Belirlenen değişikliklerin somut olarak uygulanmasıyla birlikte sürecin gerçekleştirilmesini içerir.
- **Süreç izleme:** Süreç yeniden tasarlandıktan sonra, süreç performansının değerlendirilmesi amacıyla sürecin yürütülmesiyle ilgili verilerin toplanması ve analiz edilmesini içerir.

Bir olay günlüğü, bir vakayla ilgili olay dizileri olan izlerin çoklu kümesinden oluşur. Bir olay, belirli bir etkinliğin yürütülmesi hakkında bilgi taşır. Her olayın temel zorunlu öğeleri; vaka tanımlayıcısı, etkinlik adı ve olayın ne zaman gerçekleştiğini gösteren bir zaman damgasıdır. Başka bir deyişle, her olay, belirli bir zamanda ve belirli bir vaka bağlamında bir etkinliğin gerçekleşmesini temsil eder. Ek olarak, bir olay diğer ilgili verileri de içerebilir. Bu ek ilgili verilere olay nitelikleri denir. Bu niteliklerin değeri iz boyunca dinamik olarak değişebilir ya da iz süresi boyunca sabit kalabilir ve hiç değişmeyebilir. Çizelge 2.1'de klasik bir olay günlüğü örneği gösterilmektedir. Olay günlüklerini temsil etmek için bir IEEE standardı olan Genişletilebilir Olay Akışı (eXtensible Event Stream - XES) [9], olay günlüklerinde izlerin, olayların ve niteliklerin yapısını düzenlemek için XML formatını tanımlar.

Çizelge 2.1 Bir olay günlüğü örneği

Olay Id	Vaka Id	Etkinlik	Zaman	Müşteri	Kaynak	Ürün	Fiyat	Adet
olay1	vaka1	Sipariş Oluşturma	2024-05-01 10:00	Ahmet	Sistem	U01	250	2
olay2	vaka1	Stok Kontrolü	2024-05-02 11:30	Ahmet	Ayşe	U01	250	2
olay3	vaka1	Sipariş Hazırlama	2024-05-03 14:15	Ahmet	Fatma	U01	250	2
olay4	vaka1	Sipariş Teslimi	2024-05-05 16:30	Ahmet	Kargo Firması	U01	250	2
olay5	vaka2	Sipariş Oluşturma	2024-05-11 18:00	Mehmet	Sistem	U02	100	5
olay6	vaka2	Stok Kontrolü	2024-05-12 12:00	Mehmet	Hakan	U02	100	5
olay7	vaka2	Sipariş İptali	2024-05-13 17:00	Mehmet	Sistem	U02	100	5

Modern kuruluşlar, olay günlükleri oluşturmak için iş süreçlerinin yürütülmesi hakkındaki bilgileri kaydeden, süreç farkındalığına sahip bilgi sistemlerini kullanır [10]. Olay günlüklerinin kullanılabilirliği, kuruluşlar arasında iş süreçlerini veri odaklı bir şekilde iyileştirme konusundaki ilgiyi artırmıştır. Süreç madenciliği, olay günlüklerinden yararlı bilgiler çıkarmakla ilgilenen BPM içerisinde bir araştırma alanıdır. Süreç madenciliği teknikleri, özellikle süreç keşfi, analiz, yeniden tasarım, uygulama ve izleme gibi iş süreci yönetimi görevlerinin çeşitli aşamalarını destekleyebilir.

2.2. Tahmine Dayalı Süreç İzleme

Tahmine dayalı süreç izleme (PPM), geçmiş süreç yürütme izlerinin olay günlüklerinden yararlanarak devam eden süreç yürütmelerinin tamamlanıncaya kadar nasıl ilerleyeceğini

tahmin etmeyi amaçlayan bir süreç madenciliği dalıdır [11, 12]. Bir yürütme izinin geleceğine yönelik tipik tahmin örnekleri, tamamlanma süresi, belirli bir koşulun yerine getirilmesi veya yürütülecek bir sonraki etkinlik dizisiyle ilgilidir.

PPM yaklaşımları tipik olarak iki aşama ile karakterize edilir: (i) geçmiş izlerden tahmine dayalı bir modelin öğrenildiği bir eğitim aşaması ve (ii) devam eden bir olayın gelecekteki durumlarını tahmin etmek için, tahmine dayalı modelin sorgulandığı bir tahminleme aşaması. PPM zorluklarını ele alan son çalışmalar temel olarak makine öğrenimi veya istatistiksel modellerden, yani kara kutu süreç modellerinden yararlanmaktadır [2, 12–15].

PPM yaklaşımları, sağladıkları tahminlere göre şu şekilde sınıflandırılabilir [16, 17]:

- **Sayısal tahminler**, yani sürekli ölçümlerdir. Sayısal tahminlere örnek, devam eden bir yürütmenin kalan süresi veya maliyetidir.
- **Sonuca dayalı tahminler**, yani kategorik veya ikili sonuçlarla ilgili tahminlerdir. Sonuca dayalı tahminlere örnek, belirli bir yürütmenin risk sınıfı veya bir yürütme izinin yaşam döngüsü boyunca bir doğrulamanın yerine getirilmesidir.
- **Bir sonraki etkinlik tahminleri**, yani yürütülecek etkinliklerle ilgili tahminlerdir. Örnek olarak, bir süreç yürütmesinin etkinlik dizisini, mevcut zamandan itibaren tamamlanana kadar tahmin etmek mümkündür.

PPM tekniklerini uygulayan mevcut araçlar arasında en çok kullanılan ve bilinenler şunlardır: ProM [18], Apromore [19] ve Nirdizati [20]. İlk ikisi süreç madenciliği alanının farklı dallarını kapsayan ve iyi bilinen genel amaçlı araçlar olsa da sonuncusu, özel olarak PPM'ye odaklanmıştır.

2.3. Makine Öğrenimi

Makine öğrenimi (ML), istatistiksel algoritmalar ve yöntemler üzerine kurulmuş, elle yazılmış kurallar olmadan, insana benzer (zeki) kararlar almaya çalışan yapay zekanın

(Arficial Intelligence - AI) bir alt dalıdır [21]. Bunu, verilerdeki temel kalıpları öğrenerek yapar. Önemli sonuçlar elde etmek için, temel dağılımı tam olarak örnekleyen kapsamlı bir veri seti kullanılması gerekir. ML genellikle farklı öğrenme türlerine ayrılır; denetimli, denetimsiz ve pekiştirmeli öğrenme [21, 22].

Denetimli öğrenmede, bilinen giriş ve çıkış çiftlerine sahip bir veri seti, bilgisayarı eğitmek için kullanılır ve bilgisayar daha sonra yeni girdiler için çıktıyı tahmin etmeyi öğrenir. Buna karşılık, denetimsiz öğrenme, bilgisayara etiketlenmemiş bir veri kümesi verilmesini ve verilerdeki desenleri ve yapıları keşfetmeyi içerir. Pekiştirmeli öğrenme ise bir etkenin eylemlerde bulunarak ve bu eylemlere göre cezalar veya ödüller alarak bir ortamda nasıl hareket edeceğini öğrendiği başka bir yaklaşımdır.

Derin öğrenme (Deep Learning - DL), sinir ağlarından yararlanan makine öğreniminin bir alt kategorisidir. Diğer yöntemler genellikle klasik ML modelleri olarak adlandırılır. Klasik ML modelleri, derin öğrenmeden farklı olarak özellik mühendisliğine bağımlıdır. Performansı artırmak veya doğru çalışmasını sağlamak için, yüksek seviyeli özellik bilgilerini korumak amacıyla verileri elle ön işlemden geçirmek gerekir [23]. Derin öğrenmede, bu adım model mimarisine entegre edilmiştir. Sinir ağı (Neural Network - NN), yüksek seviyeli özellikleri otomatik olarak öğrenir [21]. Klasik ML'deki en yaygın yöntemler Lojistik Regresyon, Rastgele Orman, Destek Vektör Makinesi (Support Vector Machine - SVM), Aşırı Gradyan Artırma (eXtreme Gradient Boosting - XGBoost) ve Bayes Ağıdır. Derin öğrenmede, ileri beslemeli olarak Yinelemeli Sinir Ağı (Recurrent Neural Network - RNN), Derin Sinir Ağı (Deep Neural Network - DNN), Evrişimsel Sinir Ağı (Convolutional Neural Network - CNN) ve Uzun Kısa Süreli Bellek Ağı (Long Short-Term Memory - LSTM) veya Üretken Çatışmalı Ağ (Generative Adversarial Network - GAN) gibi daha gelişmiş mimariler kullanılabilir.

2.4. Açıklanabilir Yapay Zeka

Açıklanabilirlik, yorumlanabilirlik ve şeffaflık, temel olarak aynı anlama gelen ve bir ML modelinin altta yatan mekanizmalarını ve tahminlerini anlamayı ve bunlara güvenmeyi ifade

eden yaygın terimlerdir [24]. Bir açıklama, bir karar vericinin belirli bir dizi girdiyi alarak belirli bir sonuca ulaştığı sürecin insan tarafından yorumlanabilir bir tanımıdır [25].

Açıklanabilir yapay zeka (XAI), AI sistemlerinin şeffaf, kolay anlaşılabilir ve insanlara açıklanabilir olmasını vurgulayan, yapay zeka içinde özelleşmiş bir alandır. XAI, AI sistemlerinin nasıl kararlar aldığını daha iyi anlamalarını sağlamak amacıyla kullanıcılarına, AI modellerinin iç işleyişini ortaya koymayı hedefler. XAI, Savunma İleri Araştırma Projeleri Ajansı (Defense Advanced Research Project Agency - DARPA) tarafından "AI sistemlerini anlama, uygun şekilde güvenme ve etkili bir şekilde kullanma yeteneği" olarak tanımlanmıştır [26, 27]. XAI araştırmaları, AI modellerinin karar verme sürecini daha şeffaf ve yorumlanabilir hale getirmek için model yorumlama, görselleştirme ve açıklama gibi çeşitli teknikler kullanır. Shapley Katkı Açıklamaları (SHapley Additive exPlanations - SHAP), Yerel Yorumlanabilir Model Bağımsız Açıklamalar (Local Interpretable Model-Agnostic Explanations - LIME), Dikkat Mekanizması (Attention Mechanism), Permütasyon Özelliği Önemi (PFI) ve Katman Bazında İlgilik Yayılımı (Layer-wise Relevance Propagation - LRP) yaygın olarak kullanılan XAI yöntemleridir.

3. İLİŞKİLİ ÇALIŞMALAR

Bu bölümde, tahmine dayalı süreç izleme için XAI yöntemlerini kullanan çalışmaları belirlemek üzere bilimsel literatür araştırılmış ve değerlendirilmiştir. Amaç, katkıların niteliğini ve ampirik uygulamalara veya değerlendirmelere dayalı olarak sunulan XAI modellerinin olgunluğunu göz önünde bulundurarak mevcut en son teknolojiyi anlamaktır. Daha sonra, tahmine dayalı süreç izleme için XAI uygulamalarının adımlarını ortaya çıkarmaya odaklanılmıştır. Şekil 3.1'de, bu amaca ulaşmak için kullanılan araştırma metodolojisi özetlenmektedir.

Şekil 3.1 Araştırma metodolojisine genel bir bakış

XAI ve tahmine dayalı süreç izlemeyi araştıran birkaç ikincil çalışma tespit edilmiştir. Stierle ve diğerleri [28], mevcut makine öğrenimi tabanlı açıklanabilir tahmine dayalı iş süreci izleme tekniklerine genel bir bakış ve sınıflandırma sağlamak için yapılandırılmış bir literatür taraması gerçekleştirmiştir. Yazarlar, 19 çalışmayı incelemişler ve sonuçlarını, her bir çalışmanın tahmin amacı, içsel olarak yorumlanabilir veya post-hoc, değerlendirme amacı ve değerlendirme yöntemini içeren bir kavram matrisi ile sunmuşlardır. Başka bir çalışmada, Mehdiyev ve diğerleri [29], 67 çalışmayı gözden geçirerek tahmine dayalı süreç izleme için yorumlanabilir ve açıklanabilir ML modelleri üzerine bir SLR çalışması yapmıştır. Meta veri, uygulama bağlamı, etki alanı, veri kümeleri ve sınıflandırma veya regresyon gibi uygulama görevleri, XAI yöntemleri, kara kutu modelleri ve değerlendirme ayrıntılarından bibliyometri sunmuşlardır. PPM teknikleriyle birlikte XAI yöntemlerini tek başına araştıran, ayrıntılı olarak inceleyen ve değerlendiren ve tahmine dayalı süreç izleme için XAI yöntemlerinin uygulanmasının ana adımlarını sunan yayınlanmış bir çalışma yoktur.

Sonraki alt bölümlerde, araştırma soruları ve bu sorulara karşı oluşturulan cevaplara ilişkin ayrıntılı bilgiler sunulmaktadır.

3.1. Araştırma Soruları

Araştırmanın yukarıda belirtilen hedefler doğrultusunda gerçekleştirilmesi için PICOC (Nüfus, Müdahale, Karşılaştırma, Sonuçlar, Bağlam) şablonu kullanılmıştır [30].

- **Nüfus (Population):** Tahmine dayalı süreç izleme alanında kullanılan tahmine dayalı modeller ve tenikler kümesini ifade etmektedir. Bu modeller, bir sonraki faaliyet, süreç sonucu veya çalışan bir sürecin kalan süresi gibi iş süreçlerinin çeşitli yönlerini tahmin etmek için kullanılır. Ayrıca nüfus, bu modeller tarafından yapılan tahminleri anlaması ve bunlara güvenmesi gereken veri bilimcileri, iş analistleri ve alan uzmanları gibi bu tahmine dayalı modellerin kullanıcılarını da içerir.
- **Müdahale (Intervention):** XAI yöntemlerinin PPM içindeki tahmin modellerine uygulanmasıdır. Bu, yapay zeka modellerinin karar verme sürecine ilişkin içgörü sağlamayı amaçlayan SHAP, LIME veya alana özgü kural tabanlı açıklamalar gibi çeşitli açıklanabilirlik yöntemlerinin kullanımını içerir.
- **Karşılaştırma (Comparison):** PPM'deki tahminler için açıklamalar sağlamadaki etkililiği değerlendirmek amacıyla farklı XAI yöntemlerinin karşılaştırılmasını içerebilir.
- **Sonuçlar (Outcomes):** XAI yöntemleri tarafından sağlanan açıklamaların etkinliği, kalitesi ve pratikliğidir. XAI yöntemlerinin kararlılığını, yürütme süresini ve tahmin modellerinin altta yatan veri özelliklerine karşı duyarlılığını yansıtma yeteneğini değerlendirmeyi içerir. Ayrıca sonuçlar, XAI'nin kullanıcıların anlayış, güven ve karar verme süreçleri üzerindeki etkisini değerlendirebilir.
- **Bağlam (Context):** Farklı iş alanları (örneğin, sağlık, finans, üretim), iş süreçleri türleri ve bu alanlardaki açıklanabilirlik için özel gereksinimler gibi PPM ve XAI'nin

uygulandığı ortamları içerir. Bağlam ayrıca, veri özellikleri, ön işleme teknikleri ve PPM'de kullanılan belirli tahmin modelleri gibi teknik ortamı da içerir.

Özetle, bu çalışma için PICOC çerçevesi, PPM'deki çeşitli tahmin modelleri ve kullanıcı grupları (nüfus) arasında XAI yöntemlerinin kullanımını (müdahale) incelemeyi, farklı XAI yöntemlerini karşılaştırmayı (karşılaştırma), açıklamaların kalitesine ve etkisine odaklanmayı (sonuçlar) ve bunların hepsini iş süreci alanlarının ve teknik ortamların (bağlam) belirli ayarları dahilinde içermelidir.

PICOC tanımı dikkate alınarak aşağıdaki araştırma soruları ortaya koyulmuştur:

AS1: Araştırma ve katkı türleri nelerdir?

AS2: Tahmine dayalı süreç izleme için XAI uygulamalarının temaları nelerdir?

AS3: Tahmine dayalı süreç izleme için kullanılan makine öğrenimi modelleri (karar ağaçları, topluluk yöntemleri, sınıflandırma teknikleri, kümeleme teknikleri vb.) veya derin öğrenme modelleri (LSTM, CNN, RNN vb.) nelerdir?

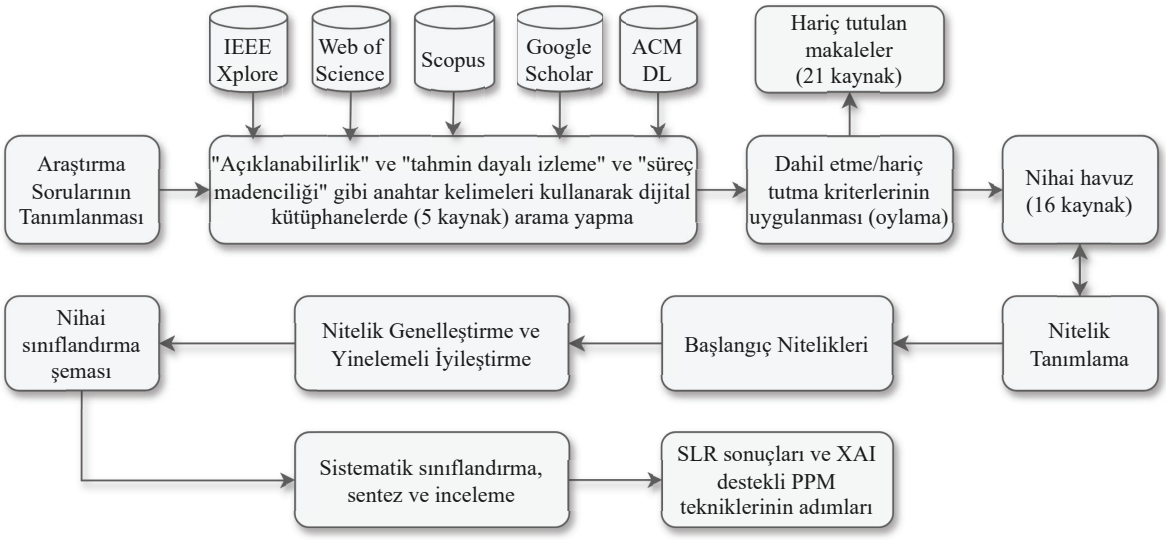
AS4: Önerilen veya kullanılan açıklanabilirlik yöntemleri (LIME, SHAP, vb.) nelerdir?

AS5: Kullanılan uygulama alanları (sağlık, bankacılık, tedarik zinciri) ve veri setleri açısından uygulamaların veya değerlendirmelerin özellikleri nelerdir?

AS6: XAI destekli tahmine dayalı süreç izleme adımları nelerdir?

3.2. Sistematik Literatür Taraması Süreci

Araştırma sorularını yanıtlamak amacıyla, sistematik literatür taraması yürütmek için kılavuzlar uygulanmıştır [30]. Sistematik literatür taraması, belirli bir konu alanı için mevcut araştırmalara dayanan kanıtları belirlemek, değerlendirmek ve yorumlamak için yapılır. SLR, boşlukları belirlemek için öneriler sunabilir veya daha fazla değerlendirme için alanlar önerebilir. Şekil 3.2'de, bu SLR gerçekleştirilirken izlenen süreç özetlenmektedir.



Şekil 3.2 SLR sürecine genel bir bakış

Birkaç ön arama ve bunların kapsamlılığının değerlendirilmesinin ardından, "açıklanabilirlik + tahmine dayalı izleme + süreç madenciliği" arama sorgusunda karar kılınmıştır. Alfabetik olarak ACM DL, Google Scholar, IEEE Xplore, Scopus ve Web of Science olarak sıralanan beş dijital kütüphanede bu araştırma gerçekleştirilmiştir. Çizelge 3.1'de bu kütüphanelerde yapılan aramaların sonuçları (i) ulaşılan ilk çalışma sayısı ve (ii) seçilen çalışma sayısı ile gösterilmektedir. Tablonun en alt satırının en sağ hücresinde görüldüğü üzere, mevcut XAI yöntemlerini kullanan ya da yeni XAI yöntemlerini tanıtan on altı birincil çalışma tespit edilmiştir. Sistematik literatür taramasına dahil edilen tüm makaleler dışarıya açık bir havuzda paylaşılmıştır: https://tinyurl.com/scXPMa

Çizelge 3.1 Başlangıçta ulaşılan ve benzersiz olarak seçilen çalışma sayısı

Dijital Kütüphane	**Başlangıçta Ulaşılan Çalışmalar**	**Benzersiz Olarak Seçilmiş Çalışmalar**
ACM DL	11	0
Google Scholar	19	4
IEEE Xplore	8	3
Scopus	21	4
Web of Science	6	5
$\sum$	65	16

3.3. Araştırma Sonuçları

Bu bölümde tahmine dayalı süreç izleme ve XAI alanında gerçekleştirilen sistematik literatür taramasına göre araştırma sorularının cevapları ele alınmıştır.

3.3.1. Araştırma ve Katkı Türleri

Çalışmaların çoğu geçerleme araştırması olmasına rağmen, değerlendirme araştırmaları oldukça azdır. Geçerleme araştırmalarının değerlendirme araştırmalarına göre baskın olması, PPM için XAI alanının gelişiminin erken bir aşamada olduğunu ve yöntemlerin değerlendirilmesi ve doğrulanması açısından daha fazla olgunluğa ihtiyaç duyulduğunu göstermektedir.

Çalışmalar katkı türü açısından incelendiğinde dört adet yeni teknik önerilmiştir. Bu teknikler vekil karar ağaçları kullanan K-LIME [6], FOX [4], geçitli RNN'nin ulaşılabilirlik grafiği [31] ve LORELEY [32]'dir. Bunların dışında mevcut XAI yöntemlerinden SHAP, LIME, PFI ve LRP gibi teknikler bu çalışmalarda kullanılmıştır.

3.3.2. Tahmine Dayalı Süreç İzleme İçin XAI Uygulamalarının Temaları

PPM'deki XAI uygulamaları, tahmine dayalı modellerin açıklanabilirliğini ve yorumlanabilirliğini artırmaya, veri türleriyle ilgili zorluklara, tahmine dayalı kalitenin iyileştirilmesine, post-hoc ve modelden bağımsız açıklamaların geliştirilmesine ve bilinçli karar almayı kolaylaştırmaya odaklanmalarıyla karakterize edilir. Bu özellikler, süreç madenciliği alanını geliştirmek ve tahmine dayalı modellerin kullanıcılar için hem etkili hem de anlaşılır olmasını sağlamak için gereklidir. PPM için XAI uygulamalarının temaları aşağıda özetlenmiştir.

3.3.2.1. Süreç Madenciliği ile Entegrasyon

XAI, süreç madenciliğinin bir alt kümesi olan PPM'e entegre edilmiştir. Bu entegrasyon, sonraki faaliyetler, nihai sonuçlar ve performans metrikleri gibi iş süreçlerinin geleceği hakkında tahminler sağlayarak değer katmayı amaçlamaktadır [3].

3.3.2.2. Kritik Alanlarda Açıklanabilirliğin Ele Alınması

PPM'deki XAI uygulamaları, tahminlerin arkasındaki mantığı anlamanın çok önemli olduğu sağlık hizmetleri gibi kritik alanlarda özellikle önemlidir. Bunun nedeni, karar alma süreçlerinde şeffaflık ve hesap verebilirlik ihtiyacıdır [5].

3.3.2.3. Veri Türleriyle İlgili Zorluklar

PPM için XAI'deki önemli bir zorluk, metinsel verilerin sınırlı olarak dikkate alınmasıdır. PPM'deki mevcut XAI yöntemlerinin çoğu öncelikle sayısal ve kategorik verilere odaklanmakta, süreç verilerindeki metinsel içerikten elde edilebilecek zengin içgörüleri göz ardı etmektedir. Bu sınırlama, açıklamaların derinliğini ve bağlamını kısıtlayarak metinde yakalanan süreç dinamiklerinin önemli yönlerinin, potansiyel olarak gözden kaçırılmasına sebep olabilir [5].

3.3.2.4. Tahmin Kalitesinin İyileştirilmesi

XAI uygulamaları, tahminlerin kalitesini artırma potansiyelleri ile karakterize edilir. Bu uygulamalar, metinsel verileri içerebilecek gelişmiş teknikler kullanarak, iş süreçlerinin izlenmesi ve yönetimi için faydalı olan daha doğru tahminler sunabilir [5].

3.3.2.5. Şeffaflık ve Yorumlanabilirlik

XAI, genellikle kara kutu olarak kabul edilen makine öğrenimi modellerinde şeffaflık sağlamayı amaçlamaktadır. Bu, kullanıcıların tahminleri yorumlamasına ve bunlara güvenmesine olanak tanır [5].

3.3.2.6. Karar Verme için Destek

PPM'de XAI, karar verme desteği ile karakterize edilir. Bu uygulamalar, tahminler için net açıklamalar sağlayarak, karar vericilerin potansiyel gelecek senaryolarını anlamalarına ve bilinçli seçimler yapmalarına yardımcı olur [6].

3.3.2.7. Modelden Bağımsız ve Modele Özgü Yöntemler

XAI yöntemleri, herhangi bir modele (model-agnostic) veya yalnızca belirli modellere (model-specific) uygulanabilmelerine göre kategorize edilir. SHAP ve LIME gibi modelden bağımsız tekniklerin kullanımı, içsel açıklanabilirliklerine bakılmaksızın çeşitli ML modelleri için açıklamalar üretilmesine olanak tanır [33] . DeepLift ve Katman Bazında İlgilik Yayılımı (LRP) gibi yöntemler, sinir ağlarına dayalı tahmin modelleri için açıklamalar sağlamak üzere tasarlanmıştır ve bu dolayısıyla modele özgüdür [31].

3.3.2.8. Ante-Hoc ve Post-Hoc Açıklama Yöntemleri

Ante-hoc yöntemler, açıklanabilirliği doğrudan tahmin modelinin mimarisine entegre ederler. Bu, modelin doğası gereği yorumlanabilir olacak şekilde tasarlandığı ve kullanıcıların modelin yapısına ve girdi verilerini işleme biçimine dayalı olarak modelin tahminlerini anlamasına olanak tanıdığı anlamına gelir. Post-hoc yöntemler, eğitildikten sonra bir modelden açıklamalar ve görselleştirmeler çıkarmayı amaçlar. Ante-hoc yöntemlerin aksine post-hoc yöntemler, modelin kendisinin yorumlanabilir olmasını gerektirmez; bunun yerine modelin çıktısını veya öğrendiği ilişkileri yorumlamaya odaklanırlar. Post-hoc yöntemler, doğası gereği yorumlanabilir olmayan karmaşık derin öğrenme mimarileri de dahil olmak üzere herhangi bir modele uygulanabilir [34].

3.3.3. Tahmine Dayalı Süreç İzleme İçin Kullanılan Makine Öğrenimi veya Derin Öğrenme Modelleri

Tahmine dayalı süreç izleme için açıklanabilir yapay zeka alanında, iş süreçleri hakkında tahminler yapmak amacıyla çeşitli makine öğrenimi ve derin öğrenme modelleri kullanılmaktadır. Bu modeller, süreç sonucu, bir sonraki faaliyet veya çalışan bir sürecin kalan süresi gibi, bir sürecin gelecekteki özelliklerini tahmin etmek için tasarlanmıştır [5]. PPM için kullanılan makine öğrenimi veya derin öğrenme modelleri aşağıda özetlenmiştir.

Bu alanda, görüntü tabanlı veriler için evrişimsel sinir ağları (CNN) ve sınıflandırma görevleri için destek vektör makineleri (SVM) gibi diğer derin öğrenme ve makine öğrenimi modellerini görmek de yaygındır.

3.3.3.1. Makine Öğrenimi (ML) Modelleri

Bu modeller, tahmine dayalı süreç izlemede sınıflandırma ve regresyon görevleri için yaygın olarak kullanılan karar ağaçlarını, destek vektör makinelerini, rastgele ormanları ve gradyan artırma makinelerini içerir. Çeşitli tahmin senaryolarında yorumlanabilirlikleri ve etkinlikleri ile bilinirler [3].

3.3.3.2. Derin Sinir Ağları (DNN)

Sinir ağlarının çeşitli mimarilerini içeren bu karmaşık modeller, tahmine dayalı süreç izleme görevlerinde yüksek doğrulukları nedeniyle kullanılır. Bununla birlikte, yorumlanabilirlik eksiklikleri nedeniyle genellikle kara kutu modeller olarak kabul edilirler [32].

3.3.3.3. Topluluk Yöntemleri

Tahmin performansını iyileştirmek için birden fazla modeli birleştiren torbalama (bagging) ve arttırma (boosting) gibi teknikler de PPM'de uygulanır. Bu yöntemler, izleme sisteminin genel tahmin gücünü artırarak varyansı ve yanlılığı azaltmaya yardımcı olabilir [3].

3.3.3.4. Rastgele Orman

Bu modeller, eğitim sırasında birden fazla karar ağacı oluşturarak ve sınıflandırma için sınıfların modunu veya regresyon için ortalama tahmini çıkararak çalışan bir topluluk öğrenme yöntemidir [35].

3.3.3.5. Uzun Kısa Süreli Bellek Ağları (LSTM)

Özellikle dizi tahmin problemleri için etkili olan bir tür tekrarlayan sinir ağlarıdır. Geçmiş olay günlüğü verilerinden öğrenme yeteneğine sahiptirler ve iş süreçlerinde gelecekteki faaliyetleri tahmin etmek için gerekli olan uzun vadeli bağımlılıkları yakalayabilirler [35]. Tahmine dayalı süreç izlemede genellikle diğer yöntemlerden daha iyi performans gösterdikleri gözlemlenmiştir, bu nedenle bu alanda sıklıkla işlevsel hale getirilirler [36].

3.3.3.6. Geçitli Grafik Sinir Ağları (GGNN)

Grafik yapısına sahip girdi verileri için tasarlanmış bir tür grafik sinir ağlarıdır. Standart grafik sinir ağlarını, daha verimli olmalarını sağlayan geçitli tekrarlayan birimler (Gated Recurrent Unit - GRU) kullanarak genişletirler. Tahmine dayalı iş süreci izleme bağlamında, süreç modellerini girdi olarak alır ve sonuç tahminleri yapmayı öğrenirler [34].

3.3.3.7. Catboost

Karar ağaçlarında gradyan artırmayı kullanan bir makine öğrenimi yöntemidir. Süreç yönetimi alanında yüksek performansı ve rekabetçi sonuçları ile dikkat çekmiştir. Catboost'un daha kısa sürede doğru tahminler sağlama yeteneği, onu zaman verimliliğinin önemli olduğu üretim ortamları için pratik bir seçim haline getirmektedir [7].

3.3.3.8. XGBoost (Aşırı Gradyan Artırma)

Gradyan artırmanın gelişmiş bir uygulamasıdır. Çok çeşitli tahmine dayalı modelleme görevlerinde verimliliği ve etkinliği ile bilinen makine öğreniminde yaygın olarak kullanılan

bir topluluk yöntemidir. XGBoost, her yeni ağacın bir öncekinin hatalarını düzeltmeye çalıştığı bir dizi karar ağacını sırayla oluşturarak çalışır. Daha sonra model çıktıları, herhangi bir ağaçtan daha doğru ve sağlam olan nihai tahmini yapmak için ağırlıklı bir toplam yoluyla birleştirilir. XGBoost, özellikle büyük hacimli ve çeşitli süreç verilerinin işlenmesinde çok önemli olan yüksek performansı ve hızı ile dikkat çekmektedir [3, 37].

3.3.3.9. Nöro-Bulanık model (FOX)

Bu modeller, verilerden öğrenebilen ve bilgiyi insanlar için yorumlanabilir bir şekilde ifade edebilen bir sistem sağlamak için sinir ağı mimarilerini bulanık mantıkla bütünleştirir. Nöro-bulanık sistemler, tahmin doğruluğu ve yorumlanabilirlik arasında iyi bir denge sunma yetenekleri nedeniyle tahmine dayalı süreç izlemede özellikle ilgi çekidir [4].

3.3.4. Tahmine Dayalı Süreç İzleme İçin Kullanılan XAI Yöntemleri

Derin öğrenme modellerinin kullanımındaki artış, bunların tahmin performansını artırma yeteneklerinden kaynaklanmaktadır. Bununla birlikte, bu modellerin karmaşıklığı çoğu zaman şeffaflığın eksikliğine neden olmakta ve ürettikleri tahminlerin yorumlanmasını zorlaştırmaktadır [33]. Bu durum, alanda daha doğası gereği yorumlanabilir modeller için bir çabaya veya bu karmaşık modeller tarafından yapılan tahminler için açıklamalar sağlayabilecek tekniklerin geliştirilmesine yol açmıştır [5, 33]. Tahmine dayalı süreç izleme için açıklanabilirlik yöntemleri aşağıda özetlenmiştir.

Karar ağaçları, lojistik regresyon ve doğrusal regresyon gibi doğası gereği yorumlanabilen makine öğrenimi modelleri, tahmin mantığında şeffaflık sağlar ve ekstra açıklamalara ihtiyaç duymaz [38].

3.3.4.1. SHAP (Shapley Katkı Açıklamaları)

Her özelliğin tahmine katkısını hesaplayarak makine öğrenimi modellerinin çıktısını açıklayan, modelden bağımsız bir yöntemdir. SHAP değerleri, modelin tahminini girdi

özelliklerine adil bir şekilde dağıtmanın bir yolunu sağlar ve bu, işbirlikçi oyun teorisine dayanır [7].

3.3.4.2. LIME (Yerel Yorumlanabilir Model Bağımsız Açıklamalar)

Bireysel tahminleri açıklamak için kara kutu modelini yerel olarak yorumlanabilir bir modele yaklaştırarak çalışan model bağımsız bir yaklaşımdır. Girdi veri örneklerini bozar ve çıktıdaki karşılık gelen değişiklikleri gözlemleyerek belirli bir tahmin için, hangi özelliklerin en etkili olduğuna dair içgörüler sağlar [5].

3.3.4.3. Permütasyon Özelliği Önemi (PFI)

Temel fikri bir özelliğin değerlerini permütasyona tabi tutmadan önceki ve tuttuktan sonraki tahmin hatalarının ortalamasını ölçmektir [3, 37]. Yöntem, özellik değerlerinin değiştirilmesiyle özellik ile sonuç arasındaki ilişkiyi bozmayı ve böylece tahmin hatasındaki artışı tahmin etmeyi amaçlar. Bu artış özelliğin önemini yansıtır; daha büyük bir artış, modelin tahminleri için daha anlamlı bir özelliği gösterir.

3.3.4.4. LORELEY

Tahmine dayalı süreç izleme modelleri için karşı olgusal açıklamalar üretir. Sentetik komşu veri örnekleri oluşturmak için genetik algoritmalar kullanan ve kara kutu tahmin modellerinin davranışına yerel olarak yaklaşmak için bu veriler üzerinde karar ağaçlarını eğiten LORE adı verilen mevcut bir tekniği genişletir [32].

3.3.4.5. Dikkat Mekanizması

Girdi özelliklerine dikkat ağırlıkları atayan bir yöntemdir. Tahmine dayalı süreç izleme görevlerinde, dikkat mekanizması LSTM gibi modellere uygulanarak, bir süreç örneğindeki hangi olayların veya faaliyetlerin modelin sonuca ilişkin tahmininde en etkili olduğunu vurgulamak için kullanılmıştır [33].

3.3.4.6. Katman Bazında İlgililik Yayılımı (LRP)

LSTM gibi derin öğrenme modelleriyle entegre edilebilen bir XAI yöntemidir. Özel olarak tasarlanmış bir dizi yayılım kuralı kullanarak tahminin sinir ağında geriye doğru yayılmasıyla çalışır. LRP, ilgi adı verilen değeri, çıkış sınıfı nöronlarından ilk giriş nöronlarına kadar yinelemeli bir şekilde hesaplar. Hesaplanan ilgi değeri, hangi girdi özelliklerinin en alakalı olduğunu gösterir [39].

3.3.4.7. Geçitli RNN için Erişilebilirlik Grafiği

Geçitli RNN için, durum uzayını daha yorumlanabilir hale getirmek amacıyla bir Petri ağının ulaşılabilirlik grafiğini kullanılır. Temel olarak Geçitli RNN'nin gizli durumlarını, kolayca analiz edilebilecek ve anlaşılabilecek yapılandırılmış bir gösterimle eşleştirir. Bu yaklaşım, sinir ağının açıklanmasına yardımcı olur ve iş süreçlerinin kalan süresine ilişkin tahmin doğruluğunu artırır [31].

3.3.5. Uygulama ve Değerlendirme Detayları

Uygulama alanları ve veri setleri açısından uygulama ve değerlendirmelerin özellikleri aşağıda sıralanmıştır:

- Kredi başvuru süreçleri, finansal kuruluşların verilerinin değerlendirme için kullanıldığı yaygın bir uygulama alanıdır. Örneğin, Hollanda'daki bir finans kuruluşundan alınan bir kredi onayı veri seti, açıklamaların test edilmesi için kullanılmıştır [40].
- Sağlık hizmetleri, bu tekniklerin hastane ortamından alınan veriler üzerinde test edildiği bir diğer yaygın uygulama alanıdır. Bunun bir örneği, Hollanda'daki bir akademik hastaneden alınan bir olay günlüğünün kavramsal bir çerçevenin değerlendirilmesi için kullanılmasıdır [4, 41].

- Üretimle ilgili iş süreçleri, açıklanabilirliğin önemli olduğu alanlar olarak belirtilmektedir, ancak belirli veri setlerinin ayrıntıları eksiktir [6].
- Birçok durumda, uygulamalar ve değerlendirmeler, gerçek olay günlükleri yerine sentetik olay günlüklerine veya simüle edilmiş süreçlere dayanmaktadır. Örneğin, birkaç etkinlikle basit kredi veya sipariş süreçleri yapay olarak üretilebilir [5, 6].
- Bilgi Teknolojileri (Information Technology - IT) hizmet yönetimi ve e-ticaret gibi alanlardan kamuya açık gerçek olay günlükleri de kullanılmıştır. Örneğin, bir IT bilet yönetim günlüğü, karşı olgusal açıklamaları test etmek amacıyla kullanılmıştır [32].
- Yaygın olarak kullanılan olay günlüğü veri setleri arasında İş Süreci Zekası Yarışması (Business Process Intelligence Challenge - BPIC) günlükleri de bulunmaktadır.

3.3.6. Araştırma Sonuçların Sentezi

Aşağıda araştırma sorularının sonuçları özetlenmiştir ve araştırma ve uygulamaya yönelik çıkarımlar da göz önünde bulundurularak bulgular tartışılmıştır.

Kavramsal Çerçeveler ve Yönergeler: Tahmine dayalı süreç izleme için açıklama yaklaşımlarının oluşturulmasına yönelik kavramsal çerçevenin geliştirilmesi, sistematik geliştirme süreçlerine duyulan ihtiyacı karşılayan önemli bir katkıdır. Bu çerçeve, açıklamaların son kullanıcıların ihtiyaçlarını karşıladığından ve aşırı karmaşık olmadığından emin olmak için tasarımcıların ve geliştiricilerin açıklama gereksinimlerini, tasarım aşamasının başında dikkate almalarına yardımcı olur [6].

Metinsel Verilerin Entegrasyonu: Literatür analizi, çeşitli açıklanabilir tahmine dayalı süreç izleme teknikleri mevcut olmasına rağmen, metinsel verilerden yararlanan yaklaşımların eksik olduğunu göstermektedir. Metinsel verilerin dahil edilmesinin, süreç izleme için daha zengin bir bağlam sağladığı ve açıklanabilirlikten ödün vermeden tahmin kalitesini artırdığı gösterilmiştir. Ancak bu entegrasyon daha fazla hesaplama kaynağı gerektirir [5].

Kullanıcı Anlayışı ve Kullanılabilirlik: Tahmine dayalı süreç izleme bağlamında XAI yöntemlerinin değerlendirmesi yapılmıştır; ancak katılımcı sayısı ve bütünleşik bir kullanıcı arayüzü değerlendirme metodolojisinin eksikliği gibi sınırlamalar mevcuttur. Bu, XAI yöntemleri tarafından oluşturulan açıklama grafiklerinin anlaşılabilirliğini ve kullanılabilirliğini değerlendirmek için daha kapsamlı kullanıcı çalışmalarına olan ihtiyacı vurgulamaktadır [7].

PPM Tekniklerinde Açıklanabilirlik Eksikliğinin Giderilmesi: Mevcut PPM tekniklerindeki kritik bir eksiklik, açıklanabilirliğin yetersizliği olup, kullanıcıların belirli tahminlerin arkasındaki mantığı anlayamamalarına neden olur. Literatür analizi, özellikle tahminlerin arkasında mantığın anlaşılmasının çok önemli olduğu sağlık gibi kritik alanlarda açıklanabilirlik ihtiyacını vurgulamaktadır. XAI yöntemleri, kara kutu ML modellerine şeffaflık ve yorumlanabilirlik sağlayarak bu soruna bir çözüm olarak tanımlanmaktadır. Açıklanabilirlik yalnızca tahmine dayalı modellerin davranışının anlaşılmasına yardımcı olmakla kalmaz, aynı zamanda süreç analistlerinin iş süreçleriyle ilgili karar almayı etkileyebilecek uygun müdahaleleri belirlemelerine de olanak tanır [4, 5].

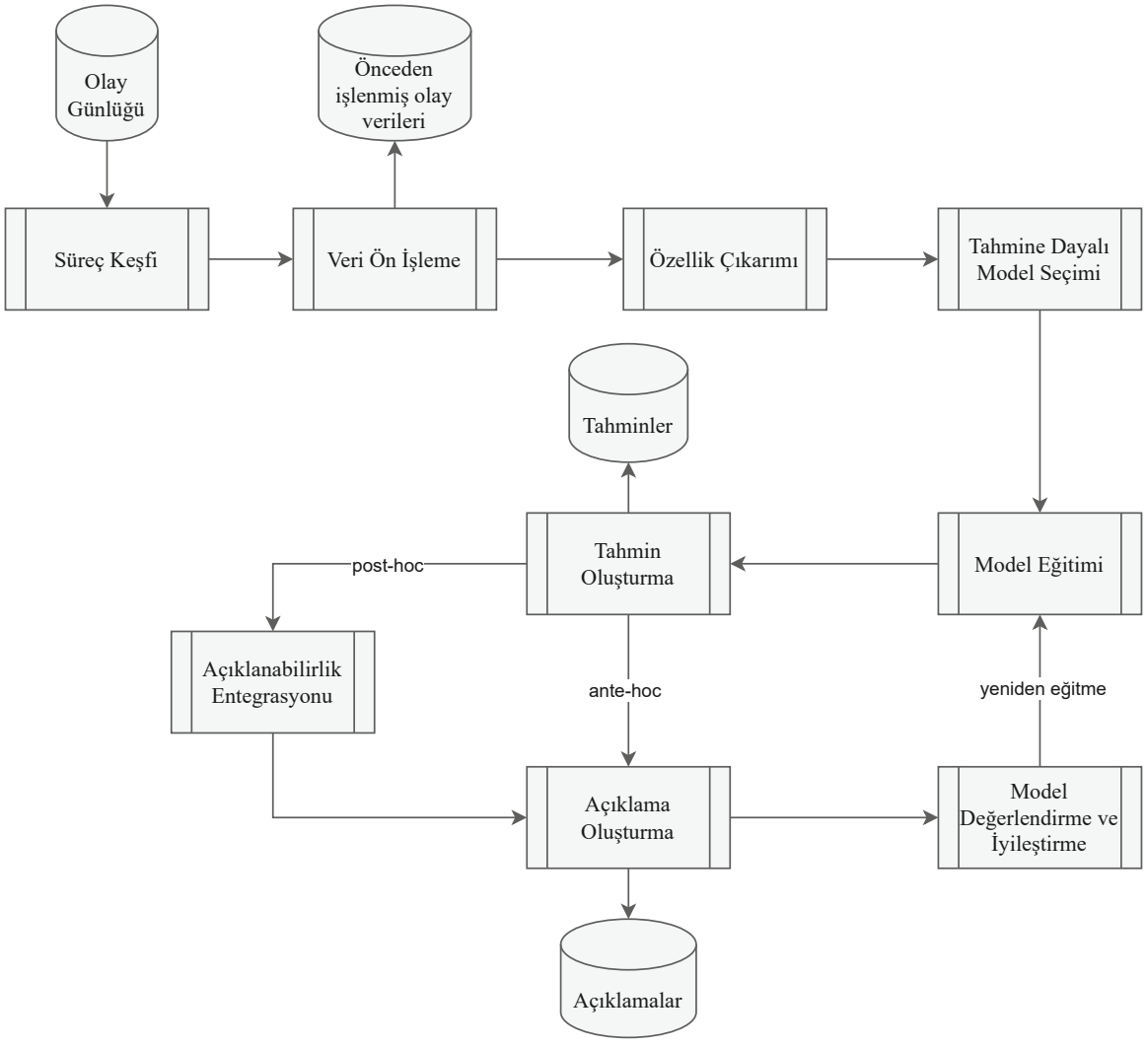
Ampirik Analiz Çerçeveleri: XAI'nin PPM'deki etkinliğini değerlendirmek için literatürde ampirik analiz çerçeveleri önerilmiştir. Bu çerçeveler, XAI yöntemlerinin gerçek dünya senaryolarındaki pratik etkilerini değerlendirmek için yapılandırılmış bir yaklaşım sağlamayı amaçlamaktadır [3].

Elde edilen sonuçlara göre bu alandaki eksiklikler göz önüne alınarak PPM teknikleriyle birlikte XAI yöntemlerinin incelenmesi ve PPM tekniklerinin XAI yöntemleri ile oluşturulan açıklamalara göre kalitesinin değerlendirilmesi amacıyla bu çalışma kapsamında aşağıdaki kısımlarda yer alan XAI destekli tahmine dayalı süreç izleme adımları çıkartılmış ve bu adımlar vaka çalışması kapsamında sırasıyla uygulanmıştır.

4. MODEL VE YÖNTEM

4.1. XAI Destekli Tahmine Dayalı Süreç İzleme Adımları

XAI destekli tahmine dayalı süreç izleme adımları, açıklanan ortak temalara ve süreçlere odaklanarak bir önceki bölümde incelenen, 16 birincil çalışmadaki bilgilerin sentezlenmesiyle elde edilmiştir. Bu adımlar Şekil 4.1'de gösterilmiştir ve aşağıda özetlenmiştir.



Şekil 4.1 XAI destekli PPM adımları

4.1.1. Süreç Keşfi

İlk adım, genellikle iş bilgi sistemleri içindeki olay günlüklerinde saklanan süreç verilerinin toplanarak süreçlerin nasıl işlediğinin keşfedilmesidir. Bu veriler, iş süreçlerinin bir parçası olarak meydana gelen olayların çeşitli özelliklerini içerirler. Bu veriler tahmine dayalı modeller için girdidir [4, 39, 40].

4.1.2. Veri Ön İşleme

Ham olay günlükleri, tahmine dayalı modeller için uygun bir formata dönüştürülmeleri amacıyla ön işleme tabi tutulabilir. Bu işlem, veri temizleme, kategorik değişkenleri kodlama ve sayısal değerleri normalleştirme gibi adımları içerebilir [37, 39, 40].

4.1.3. Özellik Çıkarımı

Özellik olarak da bilinen bağımsız değişkenler, ön işleme tabi tutulmuş verilerden çıkarılır. Bu özellikler tahminlerin yapılmasında kullanılacak bilgileri temsil eder [4, 7, 38].

4.1.4. Tahmine Dayalı Model Seçimi

Gereken karmaşıklığa ve açıklanabilirlik ihtiyacına bağlı olarak bir model seçilir. Bu, lojistik regresyon gibi basit modellerden sinir ağları gibi daha karmaşık modellere kadar değişebilir. Bazı durumlarda, doğrusal olmayan etkileşimleri yakalayabilen ve doğası gereği kendi açıklamalarını üretebilen daha basit modellerin geliştirilmiş versiyonları da dikkate alınır [33, 36, 37].

4.1.5. Model Eğitimi

Ön işleme tabi tutulmuş veriler kullanılarak seçilen makine öğrenimi veya derin öğrenme modeli, tahminler yapmak üzere eğitilir. Eğitim süreci, sürecin çeşitli yönlerini tahmin etmek

için süreç örnekleri içindeki zamansal bağımlılıkların ve kalıpların öğrenilmesini içerir [6, 31, 41].

4.1.6. Tahmin Oluşturma

Eğitilmiş model, girdi verilerine dayanarak tahminler yapmak için kullanılır. Bu tahminler süreç sonuçları, kalan yürütme süresi veya bir sonraki faaliyetle ilgili olabilir. Bu tahminlerin doğruluğu PPM'nin kullanışlılığı açısından kritik öneme sahiptir [3, 41].

4.1.7. Açıklanabilirlik Entegrasyonu

Modelin kararlarını şeffaf ve anlaşılır hale getirmek için XAI yöntemleri uygulanır. Bu, şeffaf modeller tasarlayarak ante-hoc olarak veya eğitilmiş modele açıklamalar oluşturmak için teknikler uygulayarak post-hoc olarak yapılabilir. SHAP veya LIME gibi teknikler, her bir özelliğin tahmin sonucuna katkısını vurgulayarak her bir tahmin için açıklamalar oluşturmak amacıyla kullanılabilir [3, 39].

4.1.8. Açıklama Oluşturma

Bu adım, tahmine dayalı modelin sonuçlarına nasıl ulaştığına ve karar verme sürecinde hangi özelliklerin önemli olduğuna dair içgörü sağlayan görselleştirmeler veya metinsel açıklamalar içerebilen, insanlar tarafından anlaşılabilecek açıklamalar oluşturmayı içerir [3, 37, 39].

4.1.9. Model Değerlendirme ve İyileştirme

Tahmine dayalı modelin performansı ve açıklamaların kalitesi değerlendirilir. Açıklama oluşturma sürecinden elde edilen içgörülere dayanarak tahmine dayalı modelin doğruluğunu artırmak için iyileştirmeler yapılabilir. Bu, genellikle tahmin hatalarına yol açtığı tespit edilen özelliklerin etkisinin azaltılmasını içerebilir [3, 41].

5. DENEYSEL ÇALIŞMALAR

Bu bölümde, seçilen bir olay günlüğüne önceki bölümde oluşturulan XAI destekli tahmine dayalı süreç izleme adımları uygulanarak bir vaka çalışması yapılmış ve elde edilen açıklamalardan PPM tekniklerinin kalitesini değerlendirmek amacıyla aşağıdaki araştırma sorularının cevapları araştırılmıştır. Bu vaka çalışmasında veri ön işleme, tahmine dayalı modellerin oluşturularak eğitilmesi ve XAI yöntemleri ile açıklamaların sağlanması için Python programlama dili üzerinden pandas, NumPy, TensorFlow, scikit-learn, shap ve xgboost kütüphaneleri kullanılmıştır.

AS1: Mevcut PPM teknikleri, iş süreçlerinin sonuçlarını tahmin etmede ne kadar etkilidir?

AS2: XAI yöntemleri, PPM tekniklerinin yorumlanabilirliğini nasıl etkiler?

AS3: XAI ile güçlendirilmiş PPM tekniklerinin karar verme süreçlerine etkisi nedir?

5.1. Süreç Keşfi

Bu çalışmada 4TU.ResearchData'da yer alan Karayolu Trafik Cezası Yönetim Süreci (Road Traffic Fine Management Process) veri seti kullanılmıştır. Bu veri seti, İtalya'daki yerel bir polis gücü tarafından trafik cezalarının yönetim sürecinin yürütüldüğü bir bilgi sistemine ait gerçek bir olay günlüğüdür. Bu olay günlüğü, 2000 ile 2013 yılları arasında gerçekleşmiş trafik ihlalleri, trafik cezaların kesilmesi ve ödeme süreçleri gibi bilgileri içerir.

Olay günlüğünde süreç, "Create Fine" etkinliğiyle başlar. Bu etkinlik ile birlikte amount, points, payment ve dismissal olmak üzere dört değişkene değer kaydedilir. Amount (tutar) değişkeni, suçlunun ödemesi gereken tutarı ifade ederken points (puanlar) değişkeni, suçlunun ehliyetinden düşülen puan sayısını kaydeder. Payment (ödeme), suçlunun ödediği toplam tutar olup her zaman 0.0 değeriyle başlatılır. Dismissal (iptal), para cezasının olası iptalinin çeşitli nedenlerini kodlayan bir karakter içerir. Burada yer alan NIL değeri para cezasının iptal edilmediğini ve ödenmesi gerektiğini ifade eder. Ceza kesildikten sonra polis tarafından suçlunun adresine bir trafik cezası bildirimi gönderilir. Suçlunun belirli bir süre

içerisinde para cezasını ödemesi gerekir. Tüm tutar ödenirse, süreç kapatılır. Suçlu 180 gün içinde ödeme yapmazsa, genellikle para cezasının tutarı kadar bir ceza eklenir. Postayla bildirim yapıldıktan sonra suçlu, bir hakim veya valilik aracılığıyla para cezasına itiraz edebilir. Trafik cezası ödemesinin zamanında yapılmaması durumunda süreç, alacakların tahsil edilmesi için cezanın bir kredi tahsilat şirketine gönderilmesi işlemi olan "Send for Credit Collection" etkinliğiyle sona erer [42]. Olay günlüğünde yer alan nitelikler ve açıklamaları şu şekildedir:

- **amount:** cezanın miktarı
- **org:resource:** işlemi gerçekleştiren kişi veya sistem
- **dismissal:** cezanın iptal edilip edilmediği
- **concept:name:** süreç içinde gerçekleşen etkinlik
- **vehicleClass:** ihlali yapan aracın sınıfı
- **time:timestamp:** etkinliğin gerçekleştiği tarih
- **article:** ilgili trafik kuralı veya yasa maddesi
- **points:** cezayla ilişkili puanlar
- **case:concept:name:** her bir etkinliği tanımlayan benzersiz kimlik
- **expense:** ceza süreci içerinde oluşan maliyetler
- **notificationType:** cezanın bildirilme türü
- **lastSent:** en son ne zaman bildirim yapıldığı

Olay günlüğünde yer alan olayların çoğu, ortalama olarak yalnızca dört etkinlikten oluşmaktadır. Olayların %40'ı, iki etkinlikten sonra cezanın ödenmesi (payment) etkinliğiyle birlikte sona ermektedir. Olayların %50'si ise beş veya daha fazla etkinlik içermektedir. Olayların %60'ının tamamlanması 100 günden fazla sürmektedir. Bu, birçok suçlunun

cezayı zamanında ödemediğini gösterir. Toplamda 11 etkinlik gerçekleşmektedir. Çizelge 5.1'de olay günlüğündeki etkinliklerin açıklamaları ve gerçekleşme sayıları verilmiştir.

Çizelge 5.1 Olay günlüğünde yer alan etkinlikler ve sayıları

Etkinlik	**Açıklama**	**Sayı**
Create Fine	Trafik ihlali sonrası ceza işleminin başlatılması	150370
Send Fine	Kesilen cezanın ilgili kişiye gönderilmesi	103987
Insert Fine Notification	Cezanın bildirilmesinin sisteme kaydedilmesi	79860
Add Penalty	Cezaya ek yaptırımların eklenmesi	79860
Payment	Cezanın ödenmesi	77601
Send for Credit Collection	Ödenmeyen cezanın kredi tahsilat şirketine gönderilmesi	59013
Insert Date Appeal to Prefecture	İtirazın valiliğe yapıldığı tarihin sisteme girilmesi	4188
Send Appeal to Prefecture	İtirazın valiliğe gönderilmesi	4141
Receive Result Appeal from Prefecture	Valilikten itiraz sonucunun alınması	999
Notify Result Appeal to Offender	İtiraz sonucunun ihlalciye bildirilmesi	896
Appeal to Judge	Hakime itiraz işlemi	555
	Toplam:	561470

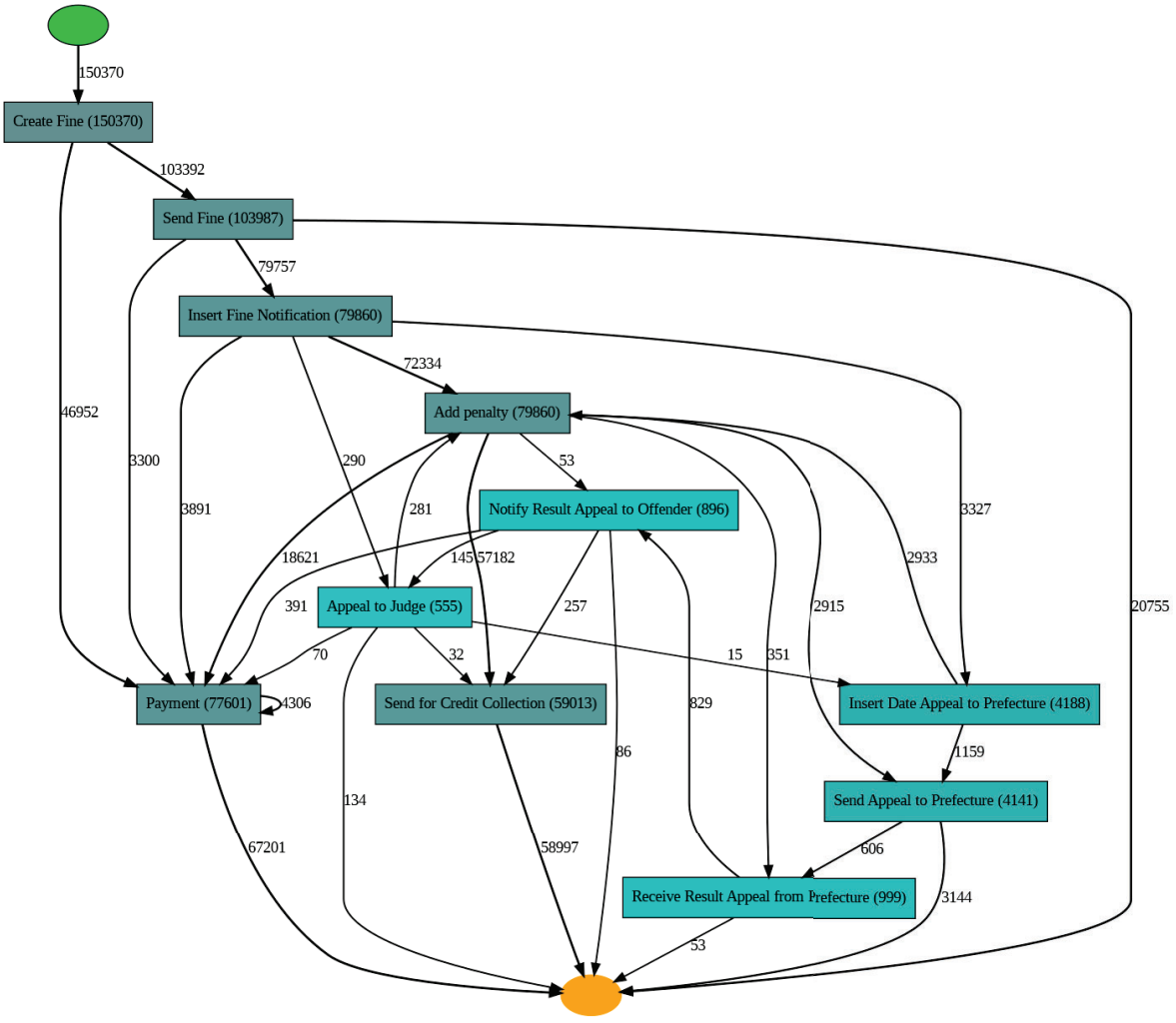
Şekil 5.1'de olay günlüğünde yer alan etkinlikler, PM4Py (Process Mining for Python) [43] kütüphanesi kullanılarak sezgisel ağlar (heuristic nets) ile gösterilmiştir. Bu gösterimde her bir etkinlik veya adım, düğüm olarak gösterilir. Düğümler arasındaki yönlü oklar, bir etkinlikten diğerine geçişi göstererek hangi etkinliğin hangi etkinlikten sonra gerçekleştiğini ifade eder.

5.2. Veri Ön İşleme

Bu çalışmada kullanılan olay günlüğünde yer alan veriler, daha anlaşılabilir ve kullanılabilir olması için veri ön işleme adımından geçirilmiştir. Bu adım, Python programlama dilinin pandas [44] ve NumPy [45] kütüphaneleri kullanılarak gerçekleştirilmiştir.

1. **Biçimin dönüştürülmesi:** İlgili olay günlüğü, xes dosya biçiminde kaydedilmiş olması sebebiyle çalışmayı daha kolay hale getirmek için csv biçimine dönüştürülmüştür.

2. **Değişkenlerin çıkarılması:** Hiç verisi olmayan ya da gereksiz olan değişkenler çıkarılmıştır. Çıkarılan değişkenler şunlardır: matricola, lifecycle:transition, paymentAmount, totalPaymentAmount.



Şekil 5.1 Olay günlüğünün sezgisel ağlar (heuristic nets) ile gösterimi

3. **Olayların çıkarılması:** ”Send Fine” etkinliğinden sonra başka bir etkinlik yoksa bu olaylar tamamanlamamış olması sebebiyle ilgili olay grubu veriden çıkarılmıştır.

4. **Hedef etkinliğin çıkarılması:** ”Send for Credit Collection” etkinliği, tahmin için kullanılacağından bu etkinlik olay günlüğünden çıkarılmıştır.

5. **Eksik değerlerin doldurulması:** Olay grubu içerisinde yer alan boş değerler, aynı olay grubu içerisindeki diğer değerlerden doldurulmuştur.

6. **Boş değerlerin doldurulması:** Kategorik olarak belirlenen değişkenlerde boş olan değerler ”missing” ifadesiyle doldurulmuştur.

Yukarıda bahsedilen adımlar uygulanarak toplamda 561470 satır veri içeren olay günlüğü 460556 satıra indirilmiş ve bununla birlikte 150370 olay grubu 129615 olaya düşürülmüştür.

Şekil 5.2'de veri ön işlemeye ait python kodu verilmiştir.

```
# özellikleri tanımla
case_id = "case:concept:name"
activity = "concept:name"
timestamp = "time:timestamp"
static_categoric = ["article", "vehicleClass"]
dynamic_categoric = [activity, "org:resource", "lastSent", "notificationType", "dismissal"]
static_numeric = ["amount", "points"]
dynamic_numeric = ["expense"]
static_columns = static_categoric + static_numeric + [case_id]
dynamic_columns = dynamic_categoric + dynamic_categoric + [timestamp]
cat_columns = dynamic_categoric + static_categoric

# 'Send Fine' etkinliği ile biten tamamlanmamış olayları çıkar
last_activities=data.sort_values([timestamp],ascending=True,kind='mergesort').groupby(case_id).last()[activity]
incomplete_cases=last_activities.index[last_activities == "Send Fine"]
data=data[~data[case_id].isin(incomplete_cases)]

# zaman damgasından çıkarılan özellikleri ekle
def extract_timestamp_features(group):
    group = group.sort_values(timestamp, ascending=False, kind='mergesort')
    temp = group[timestamp] - group[timestamp].shift(-1)
    group["timesincelastactivity"] = temp.apply(lambda x: x.total_seconds() / 60).fillna(0)
    temp = group[timestamp] - group[timestamp].iloc[-1]
    group["activity_number"] = range(1, len(group) + 1)
    return group

data = data.groupby(case_id, group_keys=False).apply(extract_timestamp_features)

# eksik değerleri olay grubundan doldur
grouped = data.sort_values(timestamp, ascending=True, kind='mergesort').groupby(case_id)
for columns in static_columns + dynamic_columns:
    data[columns] = grouped[columns].transform(lambda grp: grp.ffill())

# boş değerleri 'missing' ile doldur
data[cat_columns] = data[cat_columns].fillna('missing')
data = data.fillna(0)

# hedef etkinliği çıkar ve olay grubunu etiketle
def activity_exists(group, act):
    activity_ids = np.where(group[activity] == act)[0]
    if len(activity_ids) > 0:
        ids = activity_ids[0]
        group[label] = positive
        return group[:ids]
    else:
        group[label] = negative
        return group

data = data.sort_values([timestamp], ascending=True, kind='mergesort')
data_labeled = data.groupby(case_id).apply(activity_exists, act="Send for Credit Collection")
```

Şekil 5.2 Veri ön işlemeye ait python kodu

5.3. Özellik Çıkarımı

Özellik olarak da bilinen bağımsız değişkenler, ön işleme tabi tutulmuş verilerden oluşturulmuştur. Bu adımda tarih verisinden gün, ay, yıl değişkenleri oluşturulmuştur. Ayrıca aynı olay grubu içerisinde bir önceki etkinlik ile arada geçen süreyi ifade eden ”timesincelastactivity” değişkeni oluşturulmuştur.

”Send for Credit Collection” etkinliği, trafik cezası ödemesinin zamanında yapılmaması durumunda alacakların tahsil edilmesi için cezanın bir kredi tahsilat şirketine gönderilmesi işlemidir ve bir olay içerisinde gerçekleşebilecek en son adım olup bu adımdan sonra başka bir etkinlik gerçekleşmemektedir. Bu nedenle, bu vaka çalışmasında sonuç tahmini yapılacak olması sebebiyle bu etkinlik ilgili olaylardan çıkarılmış ve ”label” adında hedef değişkeni belirlenerek bu etkinliği içeren olay grubu ”positive”, bunların haricindeki olay grupları ise ”negative” olarak işaretlenmiştir.

Özellik çıkarımı ve çalışmanın ileriki kısmında bahsedilecek iz kodlaması ile birlikte olay günlüğünün özeti Çizelge 5.2’de verilmiştir.

Çizelge 5.2 Olay günlüğünün özeti

olay günlüğü	**etkinlik**	**olay**	**uzunluk**	**özellik (gerçek)**	**özellik (kodlama)**	**özellik (seçilen)**
trafic fines	460556	129615	10	18	249	3

5.4. Tahmine Dayalı Model Seçimi

Tahmine dayalı süreç izlemede, bir olay dizisinin belirli bir sonuca ulaşıp ulaşmayacağını tahmin etmek (sınıflandırma) veya bir olayın ne kadar süreceğini tahmin etmek (regresyon) gibi çeşitli tahmin görevleri bulunmaktadır. Bu vaka çalışması kapsamında olayların ”Send for Credit Collection” etkinliği ile sonuçlanıp sonuçlanmayacağını tahmin etmek amacıyla sınıflandırma problemi için bir makine öğrenimi modeli olan XGBoost [46] ve bir olay içerisindeki etkinliklerin sıralı olarak gerçekleşmesi sebebiyle de zaman serisi verileri üzerinde etkili olan derin öğrenme modeli LSTM [47] seçilmiştir.

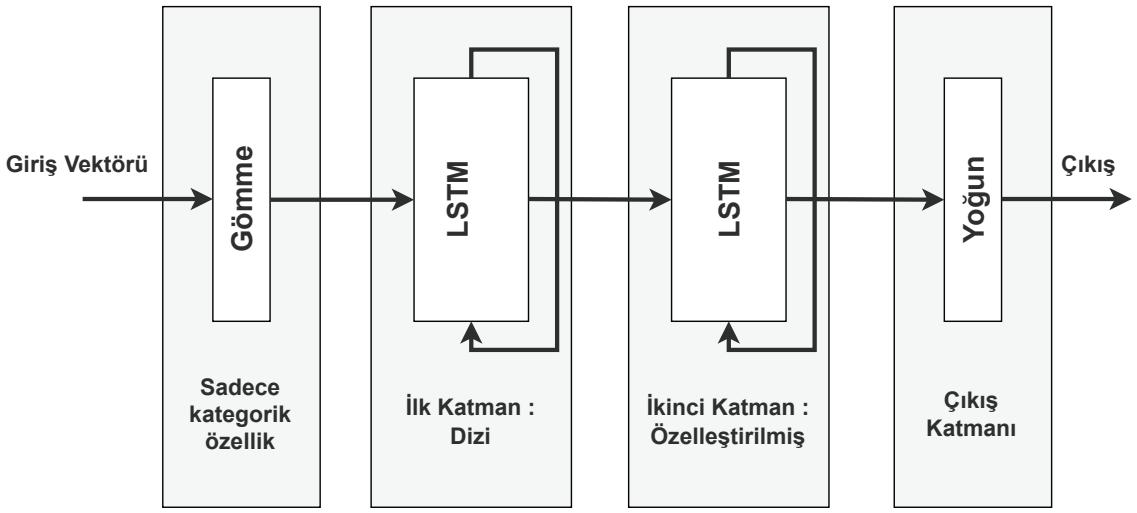
5.4.1. XGBoost (Aşırı Gradyan Artırma)

XGBoost, temel olarak Gradyan Artırma [48] algoritmasına dayanır. Aşırı Gradyan Artırma, karar ağaçları gibi zayıf öğrenicilerden oluşan bir topluluk (ensemble) oluşturur. Bu zayıf öğrenicilerden her biri, bir önceki modelin hatalarını öğrenerek bu hataların azaltılması için çalışır. Eğitim aşaması, modelin toplam hatasını azaltmak amacıyla bir dizi zayıf öğreniciler eklenerek devam eder. Bu yaklaşım, modelin tahmin doğruluğunun artırılmasına ve nihayetinde daha güçlü bir model oluşturulmasına katkı sağlar [49]. Nihai model, bir dizi zayıf öğrenicinin topluluk modeli olarak ifade edilir. XGBoost, tahmin süreci sırasında hız ve verimlilik sunması nedeniyle sınıflandırma, regresyon, sıralama, anomali tespiti gibi çeşitli makine öğrenimi görevlerinde kullanılmaktadır.

5.4.2. LSTM (Uzun Kısa Süreli Bellek)

Uzun Kısa Süreli Bellek, uzun vadeli ilişkiler ve bağımlılıkların hücre durum vektörlerine kodlandığı ve bu nedenle RNN'lerde gözlemlenen gradyan kaybolması sorununu çözen bir derin öğrenme modelidir. LSTM'lerin klasik makine öğrenimi modellerine göre avantajı, kategorik değerlerin gömme katmanında kodlandığı zamana bağlı ve ardışık veri görevlerini modelleme yeteneğidir.

LSTM, tahmine dayalı süreç izlemede bir iş sürecinin olay günlüğünde yer alan etkinlik dizilerini işlemek için kullanılan bir tahmin modelidir. Şekil 5.3'te LSTM modelinin genelleştirilmiş mimarisi gösterilmiştir. LSTM, her bir özelliği ayrı ayrı ele alan bir girdi katmanından oluşur. Bu katmanda kategorik değişkenlerin kodlanması için bir gömme (embedding) katmanı yer alır. Bunu iki LSTM katmanı ve yoğun (dense) çıkış katmanı takip eder. İlk LSTM katmanı, ikinci LSTM katmanına tek bir değer yerine bir dizi çıktısı verir. İkinci katman, her biri ayrı değişken için özelleştirilmiş bir katmandır.



Şekil 5.3 LSTM modelinin genelleştirilmiş mimarisi [2]

5.5. Model Eğitimi

Modellerin eğitilebilmesi için olay günlüğünde yer alan verilere önek uygulandıktan sonra veriler iz kodlaması ile dönüştürülmüş ve eğitim parametreleriyle model hazırlanmıştır. LSTM modelinde, XGBoost modelinden farklı olarak iz kodlamasından sonra dolgulama işlemi de gerçekleştirilmiştir.

5.5.1. Önek İşlemi

Önek, tamamlanmış bir olay dizisinden belirli sıraya kadar olan etkinlikleri alarak yeni bir etkinlik dizisi oluşturmayı ifade eder. Bu işlem sonucunda, her bir önek ve onunla ilişkili değişkenler ile hedef değişkeni içeren yeni bir veri seti oluşturulur. Bu işlem sonucunda elde edilen verilen, süreçlerin nasıl ilerlediğini anlamak ve sürecin bir sonraki adımını veya sonucunu tahmin etmek için tahmin modellerini eğitmek amacıyla kullanılır. Bu çalışmada toplamda 10 etkinlik olması sebebiyle minimum önek 1 ve maksimum önek 10 olarak seçilmiştir.

Şekil 5.4'te önek işlemine ait python kodu verilmiştir.

```python
min_prefix_len = 1
max_prefix_len = 10

def generate_prefix(data, min_len, max_len, gap=1):
    data['case_length'] = data.groupby(case_id)[activity].transform(len)
    data_prefixes = data[data['case_length'] >= min_len].groupby(case_id).head(min_len)
    data_prefixes["prefix_number"] = 1
    for number_activities in range(min_len + gap, max_len + 1, gap):
        temp = data[data['case_length'] >= number_activities].groupby(case_id).head(number_activities)
        temp[case_id] = temp[case_id].apply(lambda x: f"{x}_{number_activities}")
        temp["prefix_number"] = number_activities
        data_prefixes = pd.concat([data_prefixes, temp], axis=0)
    data_prefixes['case_length'] = data_prefixes['case_length'].clip(upper=max_len)
    return data_prefixes

train_prefixes = generate_prefix(train, min_prefix_len, max_prefix_len)
test_prefixes = generate_prefix(test, min_prefix_len, max_prefix_len)
```

Şekil 5.4 Önek işlemine ait python kodu

5.5.2. İz Kodlaması

Bu çalışmada kullanılan olay günlüğünde yer alan verilerin, makine öğrenimi ve derin öğrenme modellerine girdi olarak verilebilmesi için belirli dönüşüm işlemlerinden geçerek kodlanması gereklidir. Bu adımda statik kodlama ve toplama kodlaması işlemleri için Python programlama dilinde yer alan scikit-learn'in [50] TransformerMixin kütüphanesinden türetilen StaticTransformer ile AggregateTransformer sınıfları kullanılmıştır.

İlk olarak kategorik ve sayısal değişkenler belirlenmiş ve bunlar da kendi aralarında dinamik ve statik olmak üzere şu ayrılmıştır:

- **Statik kategorik değişkenler:** article, vehicleClass
- **Dinamik kategorik değişkenler:** concept:name, org:resource, lastSent, notificationType, dismissal
- **Statik sayısal değişkenler:** amount, points
- **Dinamik sayısal değişkenler:** expense
- **Statik değişkenler:** statik kategorik değişkenler, statik sayısal değişkenler ve case:concept:name değişkeni

- **Dinamik değişkenler:** Dinamik kategorik değişkenler, dinamik sayısal değişkenler ve time:timestamp değişkeni

Daha sonra statik kodlama ve toplama kodlaması uygulanarak dönüşüm işlemleri gerçekleştirilmiştir.

Statik Kodlama: Statik olarak tanımlanan değişkenleri dönüştürmek için kullanılır. Her bir değişkenin başlangıçtaki statik değerini alır ve bu değerleri modellemeye uygun hale getirir.

Şekil 5.5'te statik kodlama işlemine ait python kodu verilmiştir.

```python
class StaticTransformer(TransformerMixin):

    def __init__(self, case_id, cat_columns=None, num_columns=None, fillna=True):
        self.case_id = case_id
        self.cat_columns = cat_columns if cat_columns else []
        self.num_columns = num_columns if num_columns else []
        self.fillna = fillna
        self.columns = None

    def fit(self, X, y=None):
        return self

    def transform(self, X):
        data_first = X.groupby(self.case_id).first()  # her bir olay için ilk satırı al
        data_transformed = data_first[self.num_columns] # sayısal sütunları al

        if self.cat_columns:
            # kategorik sütunları one-hot encoding ile dönüştür
            data_cat = pd.get_dummies(data_first[self.cat_columns])
            # sayısal ve kategorik sütunları birleştir
            data_transformed = pd.concat([data_transformed, data_cat], axis=1)

        if self.fillna:
            # eksik değerleri 0 ile doldur
            data_transformed = data_transformed.fillna(0)

        if self.columns is None:
            # ilk dönüşümde sütun isimlerini kaydet
            self.columns = data_transformed.columns
        else:
            # sonraki dönüşümlerde aynı sütunlara sahip olmasını sağla
            data_transformed = data_transformed.reindex(columns=self.columns, fill_value=0)

        return data_transformed

    def get_feature_names_out(self, input_features=None):
        # özellik isimlerini döndür
        return self.columns
```

Şekil 5.5 Statik kodlama işlemine ait python kodu

Toplama Kodlaması: Dinamik olarak tanımlanan değişkenlerin belirli bir anahtar değişkene göre gruplandırılmasını ve bu gruplar üzerinde özet (aggregate) istatistiklerin hesaplanmasını sağlar. Her bir olay sırasındaki verileri özetleyerek (maksimum, ortalama, minimum, standart sapma, toplam) daha anlamlı analizlerin yapılmasına yardımcı olur.

Şekil 5.6'da toplama kodlaması işlemine ait python kodu verilmiştir.

```python
class AggregateTransformer(TransformerMixin):

    def __init__(self, case_id, cat_columns=None, num_columns=None, boolean=False, fillna=True):
        self.case_id = case_id
        self.cat_columns = cat_columns if cat_columns else []
        self.num_columns = num_columns if num_columns else []
        self.boolean = boolean # True ise 'max', False ise 'sum' kullanır
        self.fillna = fillna
        self.columns = None

    def fit(self, X, y=None):
        return self

    def transform(self, X):
        # kategorik sütunları one-hot encoding ile dönüştür
        data_transformed = pd.get_dummies(X[self.cat_columns])
        # etkinlik sütununu ekle
        data_transformed[self.case_id] = X[self.case_id]

        # kategorik verileri olay bazında topla
        aggregation = 'max' if self.boolean else 'sum'
        data_transformed = data_transformed.groupby(self.case_id).agg(aggregation)

        if self.num_columns:
            # sayısal sütunlar üzerinde gruplandırma ve özetleme (max, mean, min, std, sum)
            data_numeric = X.groupby(self.case_id)[self.num_columns].agg(['max','mean','min','std', 'sum'])
            # çok seviyeli sütun isimlerini düzleştir
            data_numeric.columns = ['_'.join(col) for col in data_numeric.columns]
            # sayısal ve kategorik sütunları birleştir
            data_transformed = pd.concat([data_transformed, data_numeric], axis=1)

        if self.fillna:
            # eksik değerleri 0 ile doldur
            data_transformed = data_transformed.fillna(0)

        if self.columns is None:
            # ilk dönüşümde sütun isimlerini kaydet
            self.columns = data_transformed.columns
        else:
            # sonraki dönüşümlerde aynı sütunlara sahip olmasını sağla
            data_transformed = data_transformed.reindex(columns=self.columns, fill_value=0)

        return data_transformed
```

Şekil 5.6 Toplama kodlaması işlemine ait python kodu

Yukarıda bahsedilen kodlama işlemleri içerisinde one-hot encoding olarak bilinen tek seferlik kodlama tekniği de uygulanmaktadır.

Tek Seferlik Kodlama (One-Hot Encoding): Kategorik değişkenler olarak tanımlanan verilerin sayısal olarak kodlanmasını sağlayan bir tekniktir. Bu teknikte bir kategorik değişkenin her değeri, i'inci bileşeni 1 ve geri kalanı 0 olarak ayarlanmış şekilde ikili bir vektörle temsil edilir. Diğer bir ifadeyle, bir değişkenin ne kadar farklı değeri varsa o kadar farklı değişken oluşturularak her bir değer için eşleşme sağlayan değişkenler 1 ve kalanları 0 olarak ifade edilir. Böylelikle kategorik değişkenler, makine öğrenimi ve derin öğrenme modelleri için kullanılabilir hale getirilir.

5.5.3. Dolgulama

LSTM ve diğer RNN modelleri, sıralı verileri (metin ve zaman serileri gibi) işlemek için kullanılan derin öğrenme modelleridir. Bu veriler, değişken uzunluklara sahip olabilir. Bu modeller ise sabit boyutlu girişlere ihtiyaç duyar. Bu nedenle, değişken uzunluktaki dizileri doğrudan bu modellere beslemek mümkün değildir. Dolgulama, farklı uzunluklarda olan veri dizilerini aynı uzunlukta olacak şekilde standart hale getirmek amacıyla kısa dizilerin sonuna ya da başına belirli bir dolgu değeri ekleyerek tüm dizilerin aynı uzunluğa getirilmesini sağlar. Burada önek işlemi sonucunda üretilen etkinlik dizilerinin boyutların farklı olması sebebiyle değişken uzunluktaki sıralı verileri sabit boyutlu hale getirmek için dolgulama işlemi uygulanmıştır.

Şekil 5.7'de dolgulama işlemine ait python kodu verilmiştir.

5.5.4. Modelin Oluşturulması

LSTM modelinin oluşturulması amacıyla kullanılan katmanlar genel olarak şu şekildedir:

- **Gömme (embedding) katmanı:** Yüksek boyutlu kategorik değişkenleri, tek seferlik kodlamaya kıyasla daha düşük boyutlu sürekli vektör gösterimlerine dönüştürür. Burada kategorik değişken olan etkinlikler için gömme katmanı oluşturulmuştur.

```
cat_columns = [activity, case_id]
cat_train = [group for _, group in train_prefixes[cat_columns].groupby(case_id, as_index=False)]
cat_test = [group for _, group in test_prefixes[cat_columns].groupby(case_id, as_index=False)]
cat_columns.remove(case_id)

# kategorik sütunları dolgula
cat_paddings_train = []
for i in cat_columns:
    padding = []
    for k in range(len(cat_train)):
        temp = list(cat_train[k][i])
        padding.append(temp)
    padded = pad_sequences(padding, maxlen=max_prefix_len, padding='pre', truncating='pre', value=0)
    padded = np.array(padded) / len(data.groupby([i]))
    cat_paddings_train.append(padded)

num_columns = [sincelastactivity, case_id]
num_train = [group for _, group in train_prefixes[num_columns].groupby(case_id, as_index=False)]
num_test = [group for _, group in test_prefixes[num_columns].groupby(case_id, as_index=False)]
num_columns.remove(case_id)

# sayısal sütunları dolgula
num_paddings_train = []
for i in num_columns:
    padding = []
    for k in range(len(num_train)):
        temp = list(num_train[k][i])
        padding.append(temp)
    padded = pad_sequences(padding, maxlen=max_prefix_len, padding='pre', truncating='pre', value=0)
    padded = np.array(padded) / data[i].max()
    num_paddings_train.append(padded)
```

Şekil 5.7 Dolgulama işlemine ait python kodu

- **LSTM katmanı:** Zaman serisi veya dizisel verilerdeki uzun dönemli bağımlılıkları yakalar. LSTM hücreleri, girdinin zaman içindeki sırasını dikkate alarak bilgiyi işler ve hatırlaması gereken bilgiyi saklar, unutması gerekeni unutur. Burada çift yönlü LSTM kullanılmıştır. Çift yönlü LSTM, veriyi hem ileri hem de geri yönde işler. Bu şekilde bir zaman serisinin hem geçmiş hem de gelecekteki değerlerini dikkate alarak öğrenme sürecini gerçekleştirir.

- **Yoğun (dense) katman:** Her bir nöronu, bir önceki katmandaki tüm nöronlara bağlayarak önceki katmanlar tarafından çıkarılan özelliklerin kombinasyonunu ve ağırlığını sağlar. Bu çalışmada dikkat mekanizması kullanılması sebebiyle modelin hangi zaman adımlarına ve özelliklere daha fazla dikkat ettiğinin gösterilmesi amacıyla alpha_dense ve beta_dense olmak üzere iki yoğun katman oluşturulmuştur. Yoğun katman, girdiyi alarak bir ağırlık matrisiyle çarpar, ardından bir sapma değeri ekler

ve son aşamada aktivasyon fonksiyonu uygular. Burada alpha_dense için softmax ve beta_dense için tanh aktivasyon fonksiyonları uygulanmıştır.

- **Çıkış (output) katmanı:** Tek bir nörondan oluşur ve sigmoid aktivasyon fonksiyonunu kullanarak bir olasılık değeri (0 ile 1 arasında) döndürür. Sigmoid aktivasyon fonksiyonu ikili sınıflandırma problemlerinde kullanılır. Bu katman modelin nihai çıktısını belirlediği için modelin doğruluğu ve performansı, çıkış katmanının doğru yapılandırılmasına bağlıdır. Yanlış bir aktivasyon fonksiyonu seçilirse, model hatalı sonuçlar üretebilir veya öğrenme süreci olumsuz etkilenebilir.

Bu çalışma kapsamında oluşturulan LSTM modelindeki katmanların işleyişi, giriş ve çıkış şekilleri, parametre sayıları ve birbirlerine nasıl bağlı oldukları Çizelge 5.3'te verilmiştir.

Çizelge 5.3 Vaka çalışması kapsamında oluşturulan LSTM modelinin özeti

Katman (tipi)	**Çıktı Şekli**	**Parametre #**	**Bağlı**
input_activity (InputLayer)	(None, 10)	0	-
embedding_activity (Embedding)	(None, 10, 10)	100	input_activity[0][0]
dropout (Dropout)	(None, 10, 10)	0	embedding_activity[0][0]
input_time (InputLayer)	(None, 10, 1)	0	-
inputs_all (Concatenate)	(None, 10, 11)	0	dropout[0][0], input_time[0][0]
alpha (Bidirectional)	(None, 10, 128)	38,912	inputs_all[0][0]
alpha_dense (TimeDistributed)	(None, 10, 1)	129	alpha[0][0]
beta (Bidirectional)	(None, 10, 128)	38,912	inputs_all[0][0]
alpha_softmax (Softmax)	(None, 10, 1)	0	alpha_dense[0][0]
beta_dense (TimeDistributed)	(None, 10, 11)	1,419	beta[0][0]
context (Multiply)	(None, 10, 11)	0	alpha_softmax[0][0], beta_dense[0][0], inputs_all[0][0]
lambda (Lambda)	(None, 11)	0	context[0][0]
dropout_1 (Dropout)	(None, 11)	0	lambda[0][0]
final_output (Dense)	(None, 1)	12	dropout_1[0][0]

1. **input_activity (InputLayer):** Bu katman, modelin ilk giriş katmanıdır. Modelin, input_activity adındaki etkinlik dizisini girdi olarak almasını sağlar. Bu girdi, 10 uzunluğunda bir vektördür.

2. **embedding_activity (Embedding):** Bu katman, input_activity için bir embedding işlemi uygular. Girdi vektörünü, 10 boyutlu bir gömme (embedding) vektörüne dönüştürür. Bu katman, sıralı veri için kullanışlıdır çünkü her etkinlik için bir gömme vektörü öğrenir.

3. **dropout (Dropout):** Dropout, aşırı uyumu önlemek için kullanılan bir tekniktir. Bu katman, her eğitim iterasyonunda rastgele seçilen bazı nöronların katkısını geçici olarak kapatır.

4. **input_time (InputLayer):** Bu, ikinci bir girdi katmanıdır ve zaman bilgisi (input_time) gibi başka bir özellik olarak modelin girdi almasını sağlar.

5. **inputs_all (Concatenate):** Bu katman, dropout ve input_time katmanlarından gelen çıktıları birleştirir. Sonuç olarak, 11 boyutlu bir birleşik girdi elde edilir.

6. **alpha (Bidirectional):** Bu katman, çift yönlü bir LSTM katmanıdır. Sıralı verileri ileri ve geri yönde işler. 128 boyutlu bir çıktı üretir. Bu şekilde girdinin geçmişi ve geleceği öğrenilir.

7. **alpha_dense (TimeDistributed):** Bu katman, her bir zaman adımında alpha katmanının çıktısını alır ve her adım için ayrı bir yoğun (dense) katman uygular. Her bir zaman adımı için tek bir çıktı elde edilir.

8. **beta (Bidirectional):** Bu katman da alpha katmanına benzer şekilde, çift yönlü bir LSTM katmanıdır. Ancak, farklı parametrelerle çalışır.

9. **alpha_softmax (Softmax):** Bu katman, alpha_dense katmanında gelen çıktılara softmax aktivasyon fonksiyonu uygular. Bu, her zaman adımındaki değerleri olasılık dağılımına dönüştürerek dikkatin (attention) dağılımını belirler.

10. **beta_dense (TimeDistributed):** Bu katman, beta katmanının çıktısını işleyerek her bir zaman adımı için 11 boyutlu bir çıktı üretir.

11. **context (Multiply):** Bu katman, alpha_softmax, beta_dense ve inputs_all katmanlarından gelen çıktıları element bazında çarpar. Bu, dikkat mekanizması ile ilgili bilgiyi modelin son işlemine dahil eder.

12. **lambda (Lambda):** Bu katman, çıktının boyutunu küçültmek için özel bir işlem uygular.

13. **dropout_1 (Dropout):** Bu katman, aşırı uyumu önlemek için tekrar dropout uygular.

14. **final_output (Dense):** Bu son katman, modelin çıktısını verir. Tek bir nörondan oluşur ve sınıflandırma probleminin sonucunu verir.

Şekil 5.8'de LSTM modelinin oluşturulması işlemine ait python kodu verilmiştir.

5.5.5. Eğitim Parametreleri

Bu çalışma kapsamında LSTM modelinin katmanları oluşturulurken kullanılan genel parametre değerleri şunlardır:

- **dropout (0.2):** aşırı uyumu (overfitting) önlemek için kullanılan bir tekniktir. Eğitim sırasında belirli bir oranda nöronlar rastgele devre dışı bırakılır. Bu ise modelin farklı veri noktalarına karşı daha esnek ve dayanıklı olmasını sağlar.

- **lstm size (64):** LSTM katmanlarının boyutlarını (nöron sayısını) belirler. Bu ise modelin sıralı verilerle ilgili kalıpları öğrenme kapasitesini etkiler. Örnek olarak 64 değeri, bu LSTM katmanında 64 hücre (nöron) olduğu anlamına gelir.

- **l2reg (0.001):** modelin karmaşıklığını kontrol etmek ve aşırı uyumu önlemek için kullanılan bir tekniktir. Modelin ağırlıklarının karelerinin toplamına bir ceza ekler. Bu ceza, ağırlıkların büyük değerlere ulaşmasını engeller.

LSTM modelinin eğitiminde Nadam optimizasyon algoritması kullanılmış olup bu algoritmaya verilen değerler Çizelge 5.4'te verilmiştir. Nadam, Adam optimizasyon

```python
# kategorik değişken (etkinlik) için indeks oluştur
list_activity = sorted(train_prefixes[activity].unique())
index_activity = {value: idx for idx, value in enumerate(list_activity)}
index_activity = {v: k for k, v in index_activity.items()}

# kategorik değişken için ağırlık matrisi oluştur
weights_activity = ku.to_categorical(sorted(index_activity.keys()), len(index_activity))

# giriş katmanlarını oluştur
input_layer_activity = Input(shape=(max_prefix_len,), name='input_activity')
input_layer_time = Input(shape=(max_prefix_len,1), name='input_time')

# ilk giriş katmanıyla gömme (embedding) katmanını oluştur
embedding_activity = Embedding(weights_activity.shape[0],
                               weights_activity.shape[1],
                               weights=[weights_activity],
                               name='embedding_activity')(input_layer_activity)

# dropout uygula
dropout = Dropout(0.2)(embedding_activity)

# gömme katmanıyla ikinci giriş katmanını birleştir
inputs_all = Concatenate(name='inputs_all')([dropout, input_layer_time])

# LSTM katmanları ile yoğun (dense) katmanları (alpha_dense, beta_dense) oluştur
alpha = Bidirectional(LSTM(64, return_sequences=True), name='alpha')(inputs_all)
alpha_dense = TimeDistributed(Dense(1, kernel_regularizer=l2(0.001)), name='alpha_dense')(alpha)
alpha_softmax = Softmax(axis=1, name='alpha_softmax')(alpha_dense)
beta = Bidirectional(LSTM(64, return_sequences=True), name='beta')(inputs_all)
beta_dense = TimeDistributed(Dense(inputs_all.shape[-1], activation='tanh'), name='beta_dense')(beta)

# context vektörünü oluştur
context = Multiply(name='context')([alpha_softmax, beta_dense, inputs_all])

# çıktı boyutunu küçült
lambda_layer = Lambda(lambda x: tf.reduce_sum(x, axis=1), name='lambda')(context)

# dropout uygula
dropout_1 = Dropout(0.2)(lambda_layer)

# çıkış katmanını oluştur
output_layer = Dense(1, activation='sigmoid', name='final_output')(dropout_1)

# modeli oluştur
model = Model(inputs=[input_layer_activity, input_layer_time], outputs=output_layer)
optimizer = Nadam(learning_rate=0.0005, beta_1=0.9, beta_2=0.999, epsilon=1e-08, clipvalue=3)
model.compile(loss={'final_output': 'binary_crossentropy'}, optimizer=optimizer)
model.summary()
```

Şekil 5.8 LSTM modelinin oluşturulması işlemine ait python kodu

algoritmasının Nesterov hızlandırılmış gradyan ile birleştirilmiş halidir. Adam algoritması, öğrenme oranını ve momentum parametrelerini kullanarak gradyan inişini hızlandırır. Nadam algoritması ise bunun üzerine Nesterov momentumunu da ekler. Bu ise gradyan inişini daha verimli hale getirebilir.

Çizelge 5.4 LSTM modelinde Nadam algoritmasına verilen değerler

learning_rate	**beta_1**	**beta_2**	**epsilon**	**clipvalue**
0.0005	0.9	0.999	1e-08	3

- **learning_rate:** modelin her adımda ağırlıklarını ne kadar güncelleyeceğini belirler. Öğrenme oranı, modelin hızlı veya yavaş öğrenmesini kontrol eder.
- **beta_1:** birinci moment tahmininin üstel hareketli ortalamasını kontrol eder. beta_1 değeri genellikle 0.9 olarak kullanılır. Bu da geçmiş gradyanların %90'ının dikkate alınacağı anlamına gelir.
- **beta_2:** ikinci moment tahmininin üstel hareketli ortalamasını kontrol eder. beta_2 değeri genellikle 0.999 olarak seçilir. Bu, gradyan karelerinin uzun bir geçmişine dayanarak daha istikrarlı bir öğrenme süreci sağlar.
- **epsilon:** sıfır bölmelerini ve sayısal kararsızlıkları önlemek için kullanılır. epsilon değeri, genellikle 1e-08 gibi çok küçük bir sayı olarak seçilir.
- **clipvalue:** gradyanların aşırı büyük değerlere ulaşmasını engellemek için kullanılır. Gradyanlar, bu değeri aştığı takdirde bu değerde sınırlandırılırlar. Bu ise eğitimin kararlı olması sağlar ve patlayan gradyan sorununu önler.

LSTM modelinin eğitim sürecini izlemek ve kontrol etmek için kullanılan geri çağırma (callbacks) fonksiyonları şunlardır:

- **ModelCheckpoint:** Modelin eğitimi sırasında belirli koşullara göre modelin durumunu (ağırlıklarını ve mimarisini) kaydetmek için kullanılır. Bu, eğitim sürecinde en iyi performansı elde eden modelin saklanmasını sağlar.
- **EarlyStopping:** Modelin performansı, belirli bir noktadan sonra iyileşmediğinde eğitimi durdurmak için kullanılır. Bu, gereksiz yere uzun süren eğitim süreçlerini önlemek ve aşırı uyumu engellemek amacıyla tercih edilir.

- **ReduceLROnPlateau:** Eğitim sırasında belirli bir metriğin iyileşmesi durduğunda öğrenme oranını azaltmak için kullanılır. Bu, modelin daha hassas bir şekilde öğrenmesini sağlar.

LSTM modelinin eğitilmesi için model.fit() fonksiyonu kullanılır. Bu fonksiyon, modelin belirtilen giriş verileri (X_train) ve hedefler (y_train) üzerinde öğrenme sürecini başlatır. Bu foksiyonda kullanılan validation_split değeri, eğitim verisinin belirli bir oranda doğrulama verisi olarak ayrılmasını sağlar. Bu doğrulama verisi, modelin eğitimi sırasında genel performansı değerlendirmek için kullanılır. Bu fonksiyonda kullanılan epoch değeri ise eğitim sürecinin kaç kez tekrar edileceğini ifade eder.

Şekil 5.9'da LSTM modelinin eğitimi işlemine ait python kodu verilmiştir.

```
# eğitim sırasında modelin durumunu kaydet
model_checkpoint = ModelCheckpoint(output_file_path, mode='auto', monitor='val_loss',
                                   save_best_only=True, save_weights_only=False)

# modelin performansı iyileşmediğinde eğitimi durdur
early_stopping = EarlyStopping(monitor='val_loss', patience=3)

# eğitim sırasında belirli bir metriğin iyileşmesi durduğunda öğrenme oranını azalt
reduce_lr = ReduceLROnPlateau(monitor='val_loss', cooldown=0, factor=0.5, min_delta=0.0001,
                              min_lr=0, mode='auto', patience=2)

# bağımsız değişkenler
X_train_cat = cat_paddings_train[0] # activity
X_train_num = Reshape((max_len, 1))(num_paddings_train[0]) # timesincelastactivity

X_test_cat = cat_paddings_test[0]
X_test_num = Reshape((max_len, 1))(num_paddings_test[0])

X_train = [X_train_cat, X_train_num]

# hedef değişken
y_train = np.array(y_train) # label
y_test = np.array(y_test)

# modeli eğit
model.fit(X_train, y_train,
          callbacks=[early_stopping, reduce_lr, model_checkpoint],
          epochs=10,
          validation_split=0.1)

X_test = [X_test_cat, X_test_num]

# tahminleri oluştur
y_pred = model.predict(X_test)
```

Şekil 5.9 LSTM modelinin eğitimi işlemine ait python kodu

XGBoost modelinin eğitimi için XGBClassifier metodu kullanılmıştır. Bu metoda verilen parametreler Çizelge 5.5'te verilmiştir.

Çizelge 5.5 XGBoost modelinin eğitiminde kullanılan parametreler

objective	n_estimators	learning_rate	subsample	max_depth	colsample_bytree	min_child_weight	seed
binary:logistic	100	0.05	0.2	10	0.1	2	0.42

- **objective:** modelin amacını belirtir. binary:logistic ifadesi, ikili sınıflandırma problemi için lojistik regresyon kullanılacağını ifade eder.
- **n_estimators:** boosting işlemi sırasında oluşturulacak toplam ağaç sayısını belirtir. Bu, modelin karmaşıklığını artıran bir parametredir.
- **learning_rate:** öğrenme oranını belirtir. Düşük bir öğrenme oranı modelin daha yavaş öğrenmesine, ancak daha iyi bir genelleme yapmasına olanak tanır.
- **subsample:** her bir ağaç için rastgele örneklenecek eğitim verisinin oranını belirtir. Bu, aşırı uyumu azaltmak için kullanılır.
- **max_depth:** her bir karar ağacının maksimum derinliğini belirtir. Daha derin ağaçlar daha karmaşık yapıları öğrenebilir, ancak aşırı uyum riskini artırır.
- **colsample_bytree:** her bir ağaç için rastgele seçilecek özelliklerin oranını belirtir. Bu da aşırı uyumu azaltmak için kullanılır.
- **min_child_weight:** bir yaprağa ayrılmadan önce gereken minimum örnek ağırlığını belirtir. Daha büyük bir değer, daha basit ağaçlar üretir.
- **seed:** rastgelelik içeren işlemler için rastgelelik tohumunu belirler. Bu, modelin tekrarlanabilirliğini sağlar.

5.6. Tahmin Oluşturma

Bu çalışma kapsamında, tahmini dayalı süreç izlemedeki tahmin yaklaşımlarından sonuca dayalı tahmin yöntemi kullanılmıştır. Burada sonuç ile ilgili olarak olay günlüğündeki bir etkinliğin "Send for Credit Collection" etkinliğine gitme riski tahmin edilmiştir.

5.6.1. Olay Günlüğünü Sıralama ve Bölme

Olay günlükleri genellikle iş süreçlerinden oluşturulur ve kendine özgü veri yapısı bu yapıya göre uyarlanmış bölme prosedürleri gerektirir. Literatürde birden fazla bölme tekniği mevcuttur [49]. Bu bölme teknikleri şunlardır:

- **Zamansal bölme:** Etkinlikler, başlangıç zamanlarına göre sıralanır.
- **Katı zamansal bölme:** Etkinlikler, başlangıç ve bitiş zamanlarına göre sıralanır.
- **Rastgele bölme:** Etkinlikler, zamansal sırası aktif olarak rastgeleleştirilir.

Bu çalışmada, katı zamansal bölme kullanılmıştır. Olay günlüğü sıralandıktan sonra, katı zamansal bölme prosedürüyle eğitim ve test kümesi olmak üzere veriler bölünmüştür. Olay günlüğünün ilk %80'i eğitim kümesi ve kalan %20'si ise test kümesi olarak ayrılmıştır. LSTM modeli eğitilirken eğitim kümesi içerisinden %10 luk bir kısım da doğrulama kümesi olarak ayrılmıştır. Katı zamansal bölme işleminde, sıralama ve bölme prosedürleri nedeniyle eğitim setindeki uzun süren olayların test setiyle çakışmaması için test setinin başlangıç tarihinden sonraki etkinlikler eğitim setinden çıkarılmıştır. Bu nedenle katı zamansal bölme işlemi, bölme oranından farklı olabilecek veri dağılımlarına yol açabilmektedir.

Şekil 5.10'da eğitim ve test verilerinin katı zamansal bölünmesi işlemine ait python kodu verilmiştir.

5.6.2. Performans Metrikleri

Eğitilen tahmin modellerinin performansını değerlendirmek için kullanılan çeşitli metrikler bulunmaktadır. Bu metrikler, modelin farklı yönlerini ölçer ve belirli türdeki görevler için faydalı bilgiler sunar. Sınıflandırma problemleri için yaygın olarak kullanılan değerlendirme metrikleri; doğruluk (accuracy), hassasiyet (precision), duyarlılık (recall), AUC ve F1-skorudur.

```python
train_ratio = 0.8

def split_data_strict(data, train_ratio):
    data = data.sort_values(sorting_columns, ascending=True, kind='mergesort')
    grouped = data.groupby(case_id)
    begin_time = grouped[timestamp].min().reset_index().sort_values(timestamp,ascending=True,kind='mergesort')
    train_size = int(train_ratio * len(begin_time))
    train_ids = list(begin_time[case_id])[:train_size]
    train = data[data[case_id].isin(train_ids)].sort_values(sorting_columns,ascending=True,kind='mergesort')
    test = data[~data[case_id].isin(train_ids)].sort_values(sorting_columns,ascending=True,kind='mergesort')
    split_time = test[timestamp].min()
    train = train[train[timestamp] < split_time]
    return (train, test)

train, test = split_data_strict(data, train_ratio)
```

Şekil 5.10 Katı zamansal bölme işlemine ait python kodu

Doğruluk (accuracy), tüm tahminler arasından ne kadarının doğru olduğunu gösterir. Formül 1 kullanılarak hesaplanır.

$$Doğruluk = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

Hassasiyet (precision), modelin pozitif olarak tahmin ettiği örnekler arasında gerçekten pozitif olanların oranıdır. Formül 2 kullanılarak hesaplanır.

$$Hassasiyet = \frac{TP}{TP + FP} \tag{2}$$

Duyarlılık (recall), gerçekte pozitif olan örneklerin ne kadarının model tarafından pozitif olarak doğru tahmin edildiğidir. Formül 3 kullanılarak hesaplanır.

$$Duyarlılık = \frac{TP}{TP + FN} \tag{3}$$

F1-skoru, hassasiyet ve duyarlılığın ağırlıklı ortalamasıdır. F1-skoru, hem yanlış pozitifleri hem de yanlış negatifleri dikkate aldığı için hassasiyet ve duyarlılık arasındaki dengeyi sağlar. Formül 4 kullanılarak hesaplanır.

$$F_1 = \frac{2TP}{2TP + FP + FN} \tag{4}$$

AUC, ROC (Receiver Operating Characteristic) eğrisinin altındaki alanı hesaplar. ROC eğrisi, x ve y ekseninde kartezyen koordinat sisteminde Yanlış Pozitif Oran (FPR) ve Gerçek Pozitif Oran (TPR) çizilerek elde edilir. AUC değeri, 1 değerine ne kadar yakın olursa modelin performansı o kadar iyi olarak değerlendirilir. Formül 7 kullanılarak hesaplanır.

$$TPR = \frac{TP}{TP + FN} \tag{5}$$

$$FPR = \frac{FP}{TN + FP} \tag{6}$$

$$AUC == \int TPRd(FPR) \tag{7}$$

5.6.3. Tahmin Sonuçları

AS1: Mevcut PPM teknikleri, iş süreçlerinin sonuçlarını tahmin etmede ne kadar etkilidir?

Mevcut PPM teknikleri, iş süreçleriyle ilgili olarak belirli bir iş sürecinin gerçekleşmesi durumu gibi sonuca dayalı, kalan yürütme süresi gibi sayısal ya da yürütülecek etkinliğin ne olacağını gibi bir sonraki etkinlik tahmini olarak çeşitli tahmin görevlerini yerine getirmektedir.

Bu çalışma ile tahmine dayalı süreç izleme kapsamında sonuca dayalı tahmin süreci ele alınmış olup makine öğrenimi modeli XGBoost ve derin öğrenme modeli LSTM kullanılarak bir etkinliğin gerçekleşip gerçekleşmemesiyle ilgili tahmin mekanizmaları çalıştırılmıştır. Elde edilen sonuçlar aşağıda verilmiştir.

Bu çalışma kapsamında XGBoost modeli için bağımsız değişken (X_train), etkinlikleri içeren ”case:concept:name” kategorik değişkeni olarak seçilmiştir. Hedef değişken (y_train) olarak sonuca dayalı tahmin amacının gerçekleştirebilmesi için ”Send for Credit Collection” etkinliğinin olayda yer alıp almamasından oluşturulan label değişkeni seçilmiştir. label değişkeni ile bu etkinliğin gerçekleşip gerçekleşmediği ”positive” ve ”negative” etiketleriyle veri setinde nitelendirilmiştir. Modelin eğitilebilmesi ve tahminlerin gerçekleştirilebilmesi için bu hedef değişkenin değerleri ikili sayısal değerlere dönüştürülmüştür.

LSTM modeli için bağımsız değişkenler (X_train), etkinlikleri içeren ”case:concept:name” kategorik değişkeni ile ”timesincelastactivity” sayısal değişkeni seçilmiştir. Hedef değişken (y_train) olarak da ”Send for Credit Collection” etkinliğinin olayda yer alıp almamasından oluşturulan label değişkeni seçilmiştir.

Tahmine dayalı süreç izleme modellerinin tahmin sonuçlarının değerlendirilmesi amacıyla doğruluk, hassasiyet, duyarlılık, F1-skoru ve AUC metrikleri kullanılmıştır. XGBoost ve LSTM tahmin modelleri, önceki kısımlarda tanımlanan eğitim parametreleriyle eğitilerek Çizelge 5.6’de gösterilen sonuçlar elde edilmiştir.

Çizelge 5.6 XGBoost ve LSTM modellerinin tahmin sonuçları

Metrik	**XGBoost**	**LSTM**
Doğruluk	0.66	**0.73**
Hassasiyet	0.58	**0.71**
Duyarlılık	**0.98**	0.73
F1-skoru	**0.73**	0.72
AUC	**0.79**	0.78
Geçen süre	0.5	998

Elde edilen sonuçlar, herbir metrik için ayrı ayrı ele alınarak şu değerlendirmeler yapılmıştır:

- **Doğruluk (Accuracy):** LSTM modeli, doğruluk açısından XGBoost modelinden daha iyi performans göstermektedir. LSTM’nin doğruluğu %73 iken, XGBoost’un doğruluğu %66’dır. Bu, LSTM modelinin genel olarak daha doğru sınıflandırmalar yaptığını gösterir.

- **Hassasiyet (Precision):** LSTM modeli, XGBoost'a göre kesinlik açısından da daha iyi bir performans göstermektedir. LSTM'nin hassasiyet değeri 0.71 iken, XGBoost'un değeri 0.58'dir. Bu, LSTM modelinin yanlış pozitif oranını daha iyi kontrol ederek doğru olarak tahmin ettiği pozitif sınıfların oranının daha yüksek olduğunu gösterir.

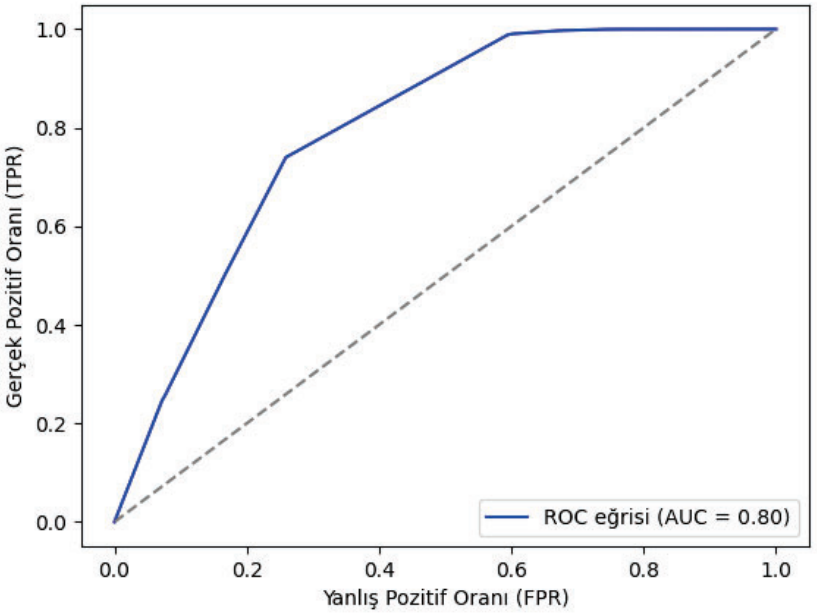
- **Duyarlılık (Recall):** XGBoost modelinin duyarlılık değeri 0.98 ile çok yüksektir. Bu da modelin pozitif sınıfları yakalama oranının oldukça iyi olduğunu gösterir. LSTM modeli ise 0.73 duyarlılık değeri ile XGBoost'a göre daha düşük bir performans göstermektedir. Bu, LSTM'nin bazı pozitif sınıfları kaçırma olasılığının daha yüksek olduğunu gösterir.

- **F1-skoru:** F1-skoru olarak her iki modelin performansı oldukça yakındır. XGBoost'un F1-skoru 0.73, LSTM'nin F1-skoru ise 0.72'dir. F1-skoru, hassasiyet ve duyarlılık arasında bir denge sağladığı için, her iki model de denge açısından benzer sonuçlar vermektedir.

- **AUC (Area Under the Curve):** AUC metriği açısından her iki modelin performansı birbirine yakındır. XGBoost'un AUC değeri 0.79 iken LSTM'nin AUC değeri 0.78'dir. Bu da her iki modelin iyi bir sınıflandırma performansına sahip olduklarını, ancak mükemmel olmadıklarını ve %80'e yakın bir olasılıkla pozitif sınıfı negatif sınıftan doğru şekilde ayırabildiklerini gösterir.

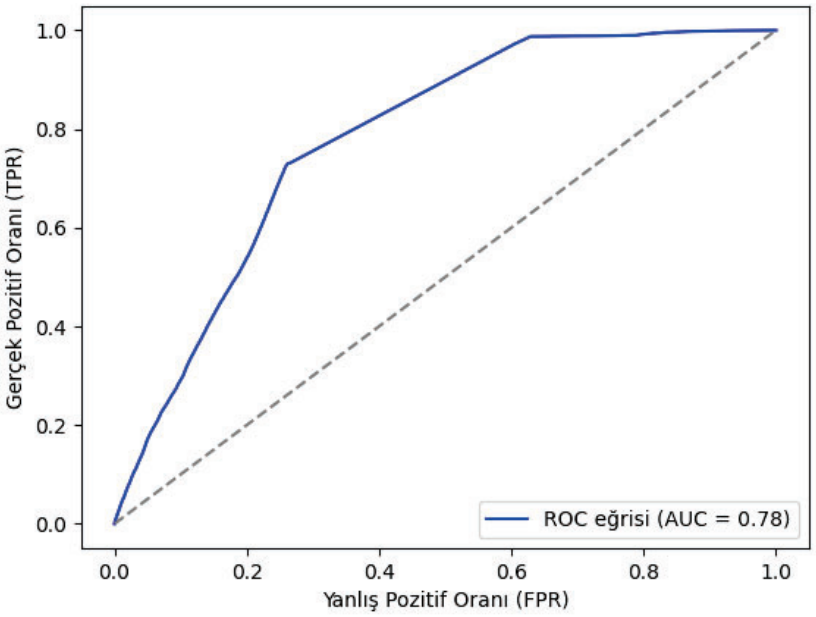
- **Geçen süre:** Burada açık bir şekilde XGBoost modeli, LSTM modeline göre çok daha hızlıdır. XGBoost, makine öğrenimi modeli olması sebebiyle tahminlerin oluşturulması yalnızca 0.5 saniye sürmüştür. LSTM ise derin öğrenme modeli olması ve çok boyutlu katman ve hücrelerden oluşması sebebiyle tahminlerin oluşturulması 998 saniye sürmüştür. Bu da LSTM'nin daha hesaplama yoğun ve zaman alıcı bir model olduğunu gösterir. XGBoost'un çok hızlı bir model olması, büyük veri kümelerinde veya hızlı sonuç gerektiren durumlarda daha avantajlı olabileceğini gösterir.

Şekil 5.11'de XGBoost modelinin ROC eğrisi ve Şekil 5.12'te LSTM modelinin ROC eğrisi gösterilmiştir. Görüldüğü üzere her iki modelin ROC eğrisi birbirine benzerdir. Mavi çizgi, modellerin ROC eğrisini temsil eder. Eğri sol üst köşeye ne kadar yakınsa, model o kadar iyidir. Grafiklerde mavi çizginin iyi bir eğriye sahip olduğunu, fakat tamamen köşeye ulaşmadığı görülmektedir. Bu, modellerin pozitif ve negatif sınıfları ayırt etme yeteneğinin makul düzeyde olduğunu, ancak bazı hata paylarının bulunduğunu gösterir.



Şekil 5.11 XGBoost modelinin ROC eğrisi

Sonuç olarak, LSTM modeli doğruluk ve hassasiyet açısından daha iyi sonuçlar verirken, XGBoost modeli duyarlılık açısından daha iyi sonuçlar vermekte ve çok daha hızlı çalışmaktadır. F1-skoru ve AUC metrikleri olarak her iki model de birbirine çok yakındır. Eğer modelin yüksek duyarlılık sağlaması kritikse (örneğin, pozitif örnekleri kaçırmamak önemliyse), XGBoost tercih edilebilir. Ancak doğruluk ve hassasiyet metrikleri daha önemliyse, LSTM daha uygun bir seçim olabilir, fakat bunun karşılığında eğitim süresi çok daha uzun olacaktır.



Şekil 5.12 LSTM modelinin ROC eğrisi

5.7. Açıklanabilirlik Entegrasyonu

Bu çalışma kapsamında açıklamalar oluşturmak amacıyla XGBoost modeline SHAP [51] yöntemi, LSTM modeline ise dikkat mekanizması [52] yöntemi uygulanmıştır.

5.7.1. SHAP

SHAP yöntemi, kara kutu olan makine öğrenimi modellerinin sonuçlarını yorumlamak için kullanılan bir post-hoc yaklaşımdır. Her bir girdi özelliğinin modelin tahmini üzerindeki etkisini değerlendirmeye yönelik bir yöntem sunar. SHAP değerleri, belirli bir özellik değeri gözlemlendiğinde beklenen model çıktısındaki değişikliği, özellik değeri bir temel değerle değiştirildiğindeki duruma kıyasla gösterir [51].

SHAP yöntemi, bir oyuncu grubunun toplam elde ettiği değeri her bir oyuncuya bireysel katkılarına göre adil bir şekilde dağıtmak için işbirlikçi oyun teorisinden Shapley değerlerini [53] kullanır. Makine öğreniminde, SHAP bağlamındaki oyuncu grubu, giriş özelliklerinin

kümesini ifade ederken elde edilen değerler modelin çıktısıdır. Shapley değerleri, her özelliğin belirli bir giriş verisi için gerçek tahmin ile beklenen tahmin arasındaki farka olan katkısını temsil eder.

SHAP, model tahminlerini detayları olarak incelemek için ayrıntılı grafikler gibi çeşitli yöntemler sunar. Özet grafikler genel yorumlanabilirlik sağlar. Bu grafikler, modelin özelliklerinin genel davranışını açıklar ve hangi özelliklerin nihai çıktı üzerinde en fazla etkiye sahip olduğunun belirlenmesine olanak tanır. Bağımlılık grafiklerinden yerel yorumlanabilirlik elde edilir ve bunlar model tahminindeki özelliklerin davranışını göstererek nihai çıktı üzerindeki bireysel etkilerin anlaşılmasına yardımcı olur. SHAP, çeşitli makine öğrenimi modellerinin çoğunu kapsayan birden fazla açıklayıcıya sahiptir.

5.7.2. Dikkat Mekanizması

Dikkat mekanizması, modelin her bir tahminde hangi zaman adımlarına veya girdilere odaklandığını açıklamak için kullanılan bir yöntemdir [52]. Bu, zaman serisi verilerinde hangi zaman adımlarının veya özelliklerin önemli olduğunu belirlemeye yardımcı olur. LSTM modelleri bir dikkat mekanizması ile birleştirildiğinde, modelin belirli bir zaman adımında hangi girdilere daha fazla dikkat ettiğini anlamak mümkün olabilir.

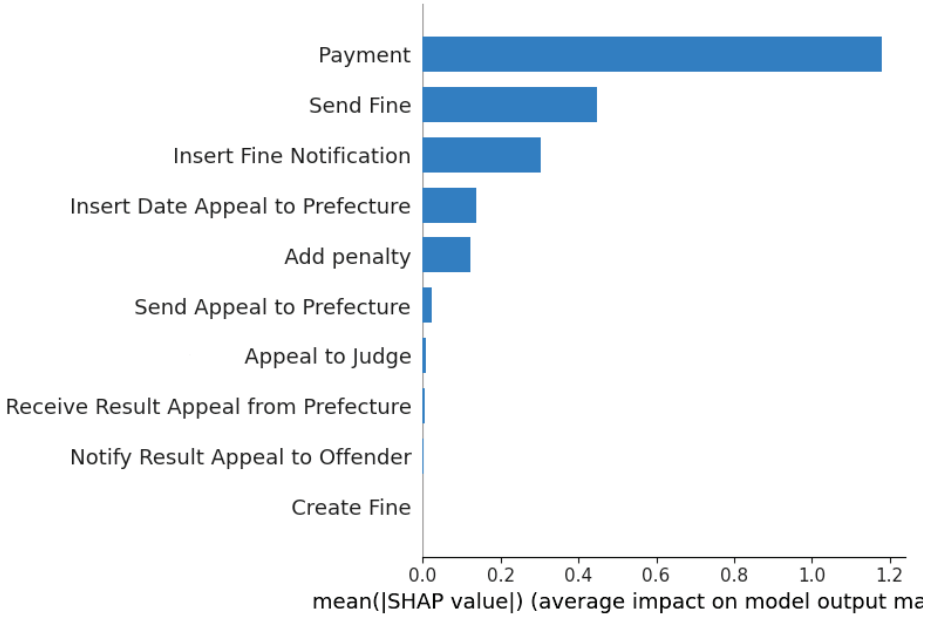
5.8. Açıklama Oluşturma

AS2: XAI yöntemleri, PPM tekniklerinin yorumlanabilirliğini nasıl etkiler?

Kara kutu makine öğrenimi ve derin öğrenme modellerini kullanan PPM teknikleri, doğası gereği karmaşık ve anlaşılması zor olması olması sebebiyle elde edilen sonuçlar üzerinde yorumlanabilirlik sağlamamaktadır. SHAP ve dikkat mekanizması gibi XAI yöntemleri bu aşamada devreye girer, modelin hangi özelliklere veya değişkenlere daha fazla önem verdiğini gösteren açıklamalar oluşturarak tahminlerin arkasındaki mantığı açıklığa kavuşturur ve bu şekilde modelin yorumlanabilirliğini önemli ölçüde artırır.

Tahmine dayalı süreç izleme için kullanılan XGBoost modeline SHAP yönteminin uygulanabilmesi için Python'da yer alan SHAP kütüphanesinin TreeExplainer metodu [54] kullanılmıştır. SHAP, her bir özelliğin bir tahmine ne kadar katkıda bulunduğunu hesaplayarak modelin belirli bir tahmini yaparken hangi özelliklere daha fazla önem verdiğinin görülmesini sağlar. Burada iki farklı gösterimle açıklamalar verilmiştir.

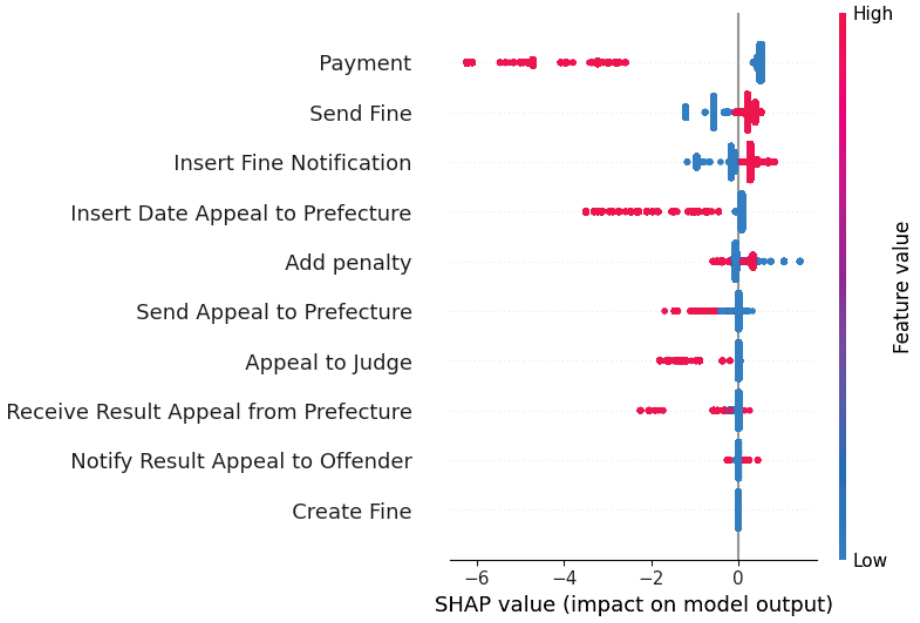
Şekil 5.13'te Bar Plot yaklaşımıyla XGBoost modelinin SHAP açıklamaları gösterilmiştir. Bu grafikte dikey eksende listelenen özellikler, modeldeki her bir özelliğin önem değerine göre sıralanmıştır. En üstte yer alan özellik, model tahminleri üzerinde en büyük etkiye sahip olan özelliktir. Bu açıklamalara göre "Payment" özelliği en yüksek ortalama SHAP değerine sahiptir ve modelin çıktısını en çok bu özellik etkilemektedir. Bu özellikten sonra sırası ile "Send Fine" ve "Insert Fine Notification" özellikleri, modelin tahminlerini belirlemede önemli rol oynamaktadır.



Şekil 5.13 XGBoost modelinin SHAP açıklamalarının Bar Plot ile gösterimi

Şekil 5.14'te Beeswarm Plot yaklaşımıyla XGBoost modelinin SHAP açıklamaları gösterilmiştir. Bu grafikte yatay eksen, SHAP değerlerini gösterir. Pozitif SHAP değerleri model tahminini artırırken, negatif değerler azaltır. "Payment" özelliğinin bazı veri noktalarında pozitif SHAP değerlerine sahip olduğu ve model tahminlerini olumlu yönde

etkilediği, bazı veri noktalarında ise negatif SHAP değerlerine sahip olduğu ve model tahminlerini olumsuz yönde etkilediği gözlenmektedir. Dikey eksen, özellikleri gösterir. En üstteki özellik, model tahmini üzerinde en büyük etkiye sahip özelliktir. ”Payment” özelliği, modelin tahminlerinde en fazla etkiye sahiptir. Noktaların renkleri, özelliğin değerini gösterir. Düşük değerler mavi ile, yüksek değerler ise kırmızı ile kodlanır. ”Payment” özelliği için kırmızı olan noktalar genellikle pozitif SHAP değerlerine sahiptir, yani yüksek değerler model tahminlerini artırma eğilimindedir. Mavi olan noktalar ise genellikle negatif SHAP değerlerine sahiptir ve bu gibi düşük değerler model tahminlerini azaltma eğilimindedir. Bu tür bir grafik, modelin tahminlerini yaparken hangi özelliklere önem verdiğini anlamak için güçlü bir araçtır ve modelin iç işleyişine dair bir öngörü sağlayarak modelin yorumlanabilirliğini artırır.



Şekil 5.14 XGBoost modelinin SHAP açıklamalarının Beeswarm Plot ile gösterimi

Şekil 5.15’te XGBoost modelinde SHAP yöntemi ile açıklamaların oluşturulması işlemine ait python kodu verilmiştir.

Tahmine dayalı süreç izleme için kullanılan LSTM modeli, zaman serileri ve sıralı veriler üzerinde çalışan ve geçmiş bilgileri uzun süreli olarak hatırlayabilen derin öğrenme modelidir. Dikkat mekanizması, LSTM modelinin tahmin görevlerini yerine getirirken giriş

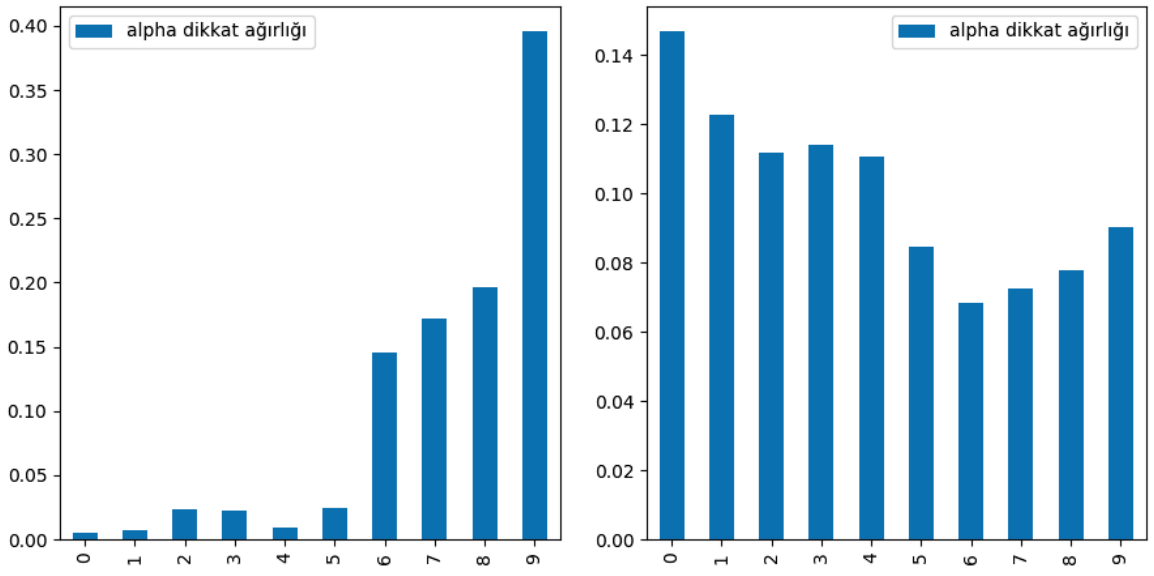
```python
model = xgb.XGBClassifier(objective='binary:logistic',
                          colsample_bytree=0.5,
                          learning_rate=0.3,
                          max_depth=15,
                          min_child_weight=4,
                          n_estimators=500,
                          subsample=0.8,
                          seed=42)

model.fit(X_train, y_train)
y_pred = model.predict_proba(X_test)[:, 1] # pozitif sınıf olasılıkları

shap_values = shap.TreeExplainer(model).shap_values(X_train)
shap.summary_plot(shap_values, X_train, plot_type="bar")

values = np.abs(shap_values).mean(0)
feature_imp = pd.DataFrame(list(zip(X_train.columns,values)),columns=['Özellik','Önem Değeri'])
feature_imp.sort_values(by=['Önem Değeri'],ascending=False,inplace=True)
print(feature_imp)

shap.summary_plot(shap_values, X_train)
```

Şekil 5.15 XGBoost'ta SHAP ile açıklamaların oluşturulması işlemine ait python kodu

dizisindeki hangi özelliklere veya zaman adımlarına daha fazla dikkat ettiğini gösterir. Bu sayede, modelin tahminlerini yaparken hangi özelliklerin veya zaman adımlarının daha etkili olduğunun anlaşılmasını sağlar.

Şekil 5.16'da LSTM modeline uygulanan dikkat mekanizması tarafından oluşturulan, zaman dikkat değerlerinin iki eğitim sonucunda oluşturulan açıklamaları gösterilmiştir. Zaman dikkat değerleri, modelin tahmin için giriş dizisindeki her bir zaman adımına ne kadar dikkat etmesi gerektiğini belirleyen ağırlık değerleridir. Bu açıklamalara göre soldaki grafikte 9. adıma doğru zaman dikkat ağırlıklarının yoğunlaşmış olduğu görülmektedir. Dikkat ağırlıkları giderek artmış ve son adımda dikkat maksimuma ulaşmıştır. Bu durum, modelin zaman serisinin sonlarına doğru daha fazla odaklandığını ve çıktıyı tahmin ederken özellikle son verilere daha fazla güvendiğini göstermektedir. Sağdaki grafikte ise daha dengeli bir dağılım olduğu görülmektedir. Dikkat değerleri zaman serisinin başında (0. adım) en yüksek değeri almış, sonraki adımlarda dikkat giderek azalmış, ancak tamamen yok olmamıştır. Bu dikkat dağılımı, modelin her zaman adımına belirli bir oranda dikkat ettiğini ve çıktıyı üretirken daha dengeli bir yaklaşım izlediğini göstermektedir.

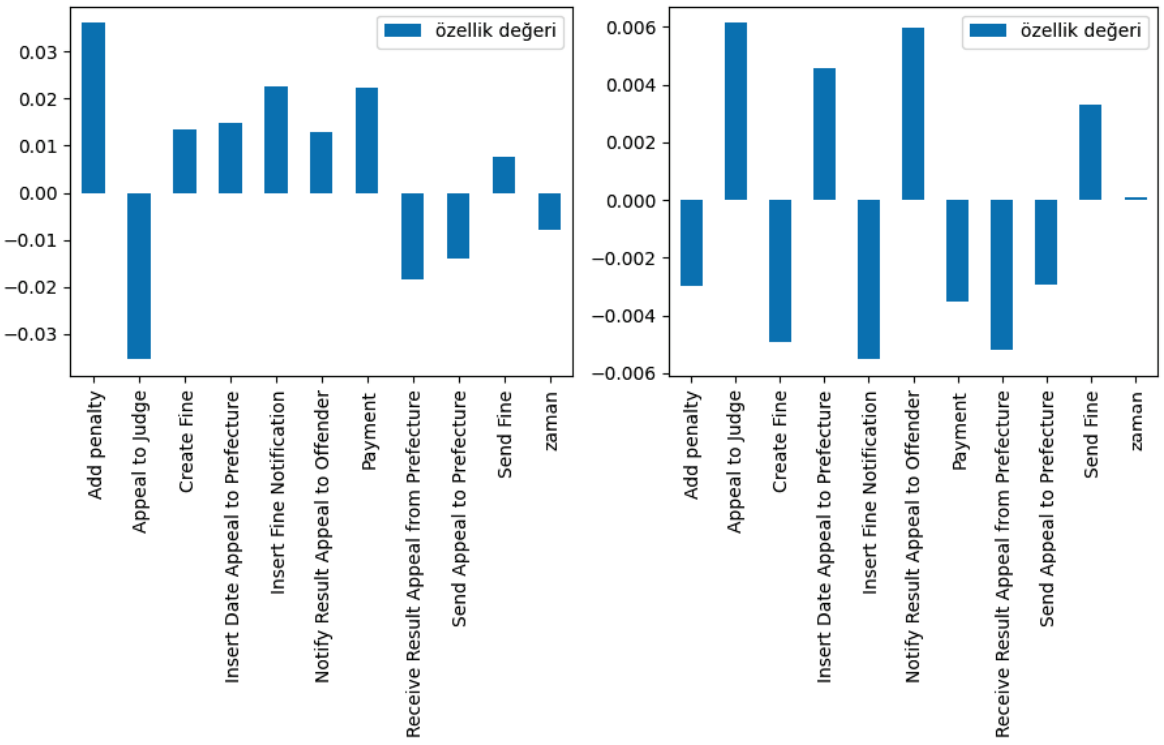
Şekil 5.17'de LSTM modeline uygulanan dikkat mekanizması tarafından oluşturulan, özellik dikkat değerlerinin iki eğitim sonucunda oluşturulan açıklamaları gösterilmiştir. Özellik



Şekil 5.16 Dikkat mekanizmasının zaman dikkat değerlerinin açıklamaları

dikkat değerleri, modelin tahmin için giriş dizisindeki farklı özelliklere ne kadar dikkat etmesi gerektiğini belirleyen ağırlıklardır. Bu grafikte dikey eksen, dikkat ağırlıklarını gösterir. Pozitif değerler, modelin bu özelliğe daha fazla dikkat ettiğini, yani bu özelliğin modelin tahminine daha fazla katkıda bulunduğunu gösterir. Negatif değerler ise, bu özelliğin modelin tahminini olumsuz yönde etkilediğini gösterir. Yatay eksen, dikkate alınan özellikleri gösterir. Soldaki grafikte dikkat pozitif yönde en çok ”Add Penalty” özelliğinde ve negatif yönde en çok ”Appeal to Judge” özelliğinde yoğunlaşmıştır. Sağdaki grafikte ise dikkat dağılımının biraz daha dengeli olduğu, hiçbir özelliğin aşırı baskın olmadığı ve böylelikle modelin her bir özelliğe belirli bir oranda dikkat ettiği görülmektedir. Sonuç olarak, bu verilen grafiklerde zaman dikkat ağırlıklarının özellik dikkat ağırlıklarını kısmen de olsa etkilediği görülmektedir.

LSTM modelinin, çok sayıda öğrenme katmanı ile hücre içeren bir derin öğreme modeli olması, hem geçmiş hem de gelecekteki değerleri dikkate alarak öğrenme sürecini gerçekleştirmesi ve modelin her eğitimde veri setindeki farklı örüntüleri keşfetmesi sebebiyle parametreler değiştirilmeden bile model tekrar eğitildiğinde elde edilen açıklamalar da değişiklik göstermektedir. Bu durum LSTM modelinin yorumlanabilirliği açısından bir



Şekil 5.17 Dikkat mekanizmasının özellik dikkat değerlerinin açıklamaları

zorluk oluşturmaktadır. Bu nedenle, makine öğrenimi modeli olan XGBoost'un eğitim parametreleri değiştirilmediğinde sabit açıklamalar oluşturması, derin öğrenme modeli olan LSTM'nin her eğitim sonunda farklı açıklamalar oluşturmasına kıyasla XGBoost modelinin LSTM modeline göre daha yorumlanabilir bir tahmine dayalı süreç izleme modeli olduğunu gösterir.

Şekil 5.18'de LSTM modelinde dikkat mekanizmasıyla açıklamaların oluşturulması işlemine ait python kodu verilmiştir.

5.9. Model Değerlendirme ve İyileştirme

Bu aşamada, tahmin modellerine verilen parametre değerleri değiştirilerek modelin performansının artırılması sağlanır. Hiperparametre optimizasyonu [55] adı verilen bu işlem, bir makine öğrenimi ya da derin öğrenme modelinin performansını artırmak için modelin hiperparametrelerini sistematik olarak arama ve ayarlama sürecidir. Hiperparametreler, modelin eğitilmesi sırasında ayarlanması gereken ancak modelin kendisi tarafından

```python
# gömme katmanını al
weights_embeding_activity = model.get_layer(name='embedding_activity').get_weights()[0]

# dikkat modelini oluştur
model_with_attention = Model(model.inputs, model.outputs + \
                             [model.get_layer(name='alpha_softmax').output, \
                              model.get_layer(name='beta_dense').output])

# dikkat modelinden tahmin değerleri (proba) ile birlikte alphas ve betas değerlerini al
proba, alphas, betas = model_with_attention.predict([X_test_cat, X_test_num])

# alphas ve betas'ı sıkıştır
alphas = np.squeeze(alphas)
betas = np.squeeze(betas)

# her örnek için en yüksek alpha değerinin indeksini bul
idx = np.argmax(alphas, axis=1)

# en yüksek alpha değerinin olduğu indekste beta değerlerini al
beta_val = betas[np.arange(len(betas)), idx]

# o indekste kullanılan etkinlik girişlerini al
activity_input = X_test_cat[np.arange(len(X_test_cat)), idx].astype(int)

# gömme katmanından etkinlik ağırlıklarını al
activity_embedding = weights_embeding_activity[activity_input]

# o indeksteki zaman değerlerini al
time_input = X_test_num[np.arange(len(X_test_num)), idx, 0]

# etkinlik ağırlıkları ile zaman değerlerini birleştir
embed = np.concatenate([activity_embedding, time_input[:, np.newaxis]], axis=1)

# beta değerleri ile çarp
beta_scaled = np.multiply(beta_val, embed)

# alpha değerleri ile çarparak değişken dikkatini hesapla
alpha_val = alphas[np.arange(len(alphas)), idx]
context = alpha_val[:, np.newaxis] * beta_scaled

# Zaman dikkat vektörlerini ve değişken dikkat vektörlerini oluştur (ortalama değerleri al)
temporal_vectors = np.mean(alphas, axis=0)
variable_vectors = np.mean(context, axis=0)

# Zaman dikkat ağırlıklarını görselleştir
pd.DataFrame(temporal_vectors, columns=['alpha dikkat ağırlığı']).plot(kind='bar')
plt.show()

# Etkinlik etiketlerini hazırla
labels_activity = [index_activity[key] for key in sorted(index_activity.keys())]
labels_activity.append('zaman')

# Değişken dikkat ağırlıklarını görselleştir
data_variable = pd.DataFrame({'özellik değeri': variable_vectors, 'özellik': labels_activity})
data_variable.plot.bar(y='özellik değeri', x='özellik')
plt.show()
```

Şekil 5.18 LSTM'de dikkat mekanizması ile açıklamaların oluşturulması işlemine ait python kodu

öğrenilmeyen parametrelerdir. Bu parametrelerin doğru şekilde seçilmesi, modelin performansını büyük ölçüde etkileyebilir.

Hiperparametre optimizasyon yöntemleri şunlardır:

- **Izgara Arama:** Belirli hiperparametre değerlerinin kapsamlı bir şekilde araştırıldığı en temel yöntemdir [55]. Her hiperparametre kombinasyonu için model eğitilir ve sonuçlar çapraz doğrulama ile değerlendirilir.
- **Rastgele Arama** [56]: Izgara aramaya göre daha verimli bir yöntemdir. Hiperparametrelerin belirli bir aralıktan rastgele olarak seçildiği ve çapraz doğrulama ile değerlendirildiği bir yaklaşımdır. Tüm kombinasyonlar tek tek denenmediği için hızlıdır, fakat en iyi hiperparametrelerin seçilmesi garanti edilemez.
- **Bayes Optimizasyonu** [57]: Hiperparametre dizisinde en iyi kombinasyonu bulmak için geçmiş denemelerden elde edilen bilgilere dayalı bir yaklaşım kullanır.

Hiperparametrelerin genel performansını doğru bir şekilde değerlendirmek için genellikle çapraz doğrulama kullanılır. Çapraz doğrulama, modelin eğitim verisine aşırı uyum sağlamadan genel performansını test etmek amacıyla veri setini birden fazla parçaya böler ve modelin her parça üzerinde nasıl performans gösterdiğini analiz eder. Hiperparametre optimizasyonunda modelin performansını değerlendirmek için kullanılacak metrik de belirlenmelidir. Bu, doğruluk (accuracy), kesinlik (precision), duyarlılık (recall), F1 skoru (f1) ve AUC (roc_auc) gibi bir metrik olabilir.

Bu çalışmada, hiperparametre optimizasyonu için XGBoost modeline rastgele arama yöntemi uygulanmıştır. Bunun için Python'da scikit-learn [50] kütüphanesinde yer alan RandomizedSearchCV metodu kullanılmıştır.

Hiperparametre optimizasyonunda rastgele arama yöntemi için kullanılan olası hiperparametre değerlerini içeren liste Çizelge 5.7'de verilmiştir.

Olası hiperparametre değerlerini içeren liste üzerinde, Çizelge 5.8'de verilen parametre değerleriyle RandomizedSearchCV metodu çalıştırılmıştır.

- **estimator:** hiperparametrelerinin optimize edilmesi istenilen modeli belirtir.

Çizelge 5.7 Modelin eğitimi için kullanılacak olası hiperparametre değerleri

Hiperparametre	**Olası Değerler**
colsample_bytree	0.1, 0.3, 0.5, 0.7, 0.9
learning_rate	0.05, 0.1, 0.15, 0.2, 0.25, 0.3
max_depth	10, 13, 15, 20, 25
min_child_weight	2, 3, 4, 5, 7
n_estimators	100, 300, 500
subsample	0.2, 0.4, 0.6, 0.8, 1.0

Çizelge 5.8 Hiperparametre optimizasyonu için rastgele arama parametreleri

estimator	**param_distributions**	**n_iter**	**cv**	**scoring**	**n_jobs**
model	param_dist	100	5	roc_auc	1

- **param_distributions:** hiperparametrelerin alabileceği olası değerlerin belirtildiği bir listedir.
- **n_iter:** rastgele aramanın kaç farklı hiperparametre kombinasyonunu deneyeceğini belirtir.
- **cv:** çapraz doğrulama (cross-validation) işleminin kaç kat (fold) üzerinde yapılacağını belirtir.
- **scoring:** modelin performansını değerlendirmek için kullanılacak metriği ifade eder.
- **n_jobs:** işlemlerin paralel olarak kaç çekirdekte çalışacağını belirler.

Hiperparametre optimizasyonu sonucunda elde edilen hiperparametre değerleri Çizelge 5.9'da verilmiştir.

Çizelge 5.9 Hiperparametre optimizasyonu sonucunda elde edilen hiperparametre değerleri

colsample_bytree	**learning_rate**	**max_depth**	**min_child_weight**	**n_estimators**	**subsample**
0.7	0.25	10	7	500	0.8

XGBoost modelinin eğitiminde kullanılan hiperparametrelerin optimizasyon öncesi ve sonrası değerleri ile birlikte bu değerlerle eğitilen modelin performans metriklerinin sonuçları Çizelge 5.10'da verilmiştir.

Çizelge 5.10 Hiperparametrelerin optimizasyon öncesi ve sonrası değerleri ile performans sonuçları

Hiperparametre	**Öncesi**	**Sonrası**
colsample_bytree	0.1	0.5
learning_rate	0.05	0.3
max_depth	10	15
min_child_weight	2	4
n_estimators	100	500
subsample	0.2	0.8
Doğruluk	0.66	**0.74**
Hassasiyet	0.58	**0.72**
Duyarlılık	**0.98**	0.74
F1-skoru	**0.73**	**0.73**
AUC	0.79	**0.80**
Geçen süre	0.5	3.5

Elde edilen sonuçlara göre hiperparametre optimizasyonundan sonra, belirlenen performans metriklerine göre tahmin doğruluğunun arttığı ve bununla birlikte modelde iyileşme sağlandığı görülmektedir. Aşağıda, hiperparametre optimizasyonundan önceki ve sonraki hiperparametre değerlerinin ve performans metriklerinin sonuçlarının karşılaştırılması yapılmıştır.

5.9.1. Hiperparametrelerin Karşılaştırılması

- **colsample_bytree:** Bu parametre, her bir ağacın eğitiminde kullanılacak özelliklerin oranını ifade eder. İlk değer olan 0.1'e kıyasla, optimizasyondan sonra bu oran 0.5'e çıkmış, yani model her bir ağaç için daha fazla özellik kullanmıştır. Bu, modelin daha fazla bilgi kullanarak daha güçlü tahminler yapmasını sağlar.

- **learning_rate:** Bu parametre, modelin öğrenme hızını belirler. İlk başta 0.05 gibi düşük bir öğrenme oranı seçilmişken, optimizasyondan sonra bu oran önemli ölçüde artarak 0.3 olmuştur. Daha yüksek bir öğrenme oranı, modelin daha hızlı öğrenmesini sağlar, ancak dikkatli ayarlanmazsa aşırı öğrenme riski doğurabilir.

- **max_depth:** Bu parametre değeri 10'dan 15'e çıkmıştır. Daha derin ağaçlar, modelin daha fazla parametre öğrenmesine olanak tanır. Ancak modelin aşırı uyum riskini ve hesaplama süresini artırabilir.

- **min_child_weight:** Bu parametre, her yaprakta bulunması gereken minimum ağırlığı tanımlar. Optimizasyondan sonra elde edilen yüksek değer, modelin daha basit ağaçlar oluşturmasını sağlar ve aşırı uyumu engelleyebilir.

- **n_estimators:** Optimizasyondan sonra n_estimators değeri 5 katına çıkmıştır. Bu, modelin daha fazla ağaç oluşturacağı anlamına gelir ve genellikle daha fazla öğrenme kapasitesi sağlar.

- **subsample:** Hiperparametre optimizasyonundan sonra subsample değeri önemli ölçüde artmıştır. Bu, modelin eğitim sırasında her ağacın alt örneklemesinde daha fazla veri kullandığını gösterir. Daha yüksek subsample değerleri genellikle modelin aşırı uyum olasılığını azaltabilir.

5.9.2. Performans Metriklerinin Sonuçlarının Karşılaştırılması

- **Doğruluk:** Optimizasyondan sonra doğruluk artmış, yani modelin genel performansı iyileşmiştir. %65 doğruluk oranı %74'e yükselmiştir.

- **Hassasiyet:** Bu metrikte de belirgin bir iyileşme görülmektedir. Optimizasyondan sonra model, doğru pozitif tahminleri artırmıştır. Bu, modelin yanlış pozitif oranını azaltma başarısını gösterir.

- **Duyarlılık:** Optimizasyondan önce oldukça yüksek bir değerde olan bu metrik azalmıştır. Duyarlılık azalması, modelin pozitif sınıfları yakalama kabiliyetinin bir miktar düştüğünü, ancak hassasiyet ve diğer metriklerin daha dengeli hale getirildiğini gösterir.

- **F1-skoru:** Bu metrik değişmemiştir. Bu ise hassasiyet ve duyarlılık arasındaki dengenin korunduğunu göstermektedir. İyileşme olmamasına rağmen dengeli bir performans sunar.
- **AUC:** Bu metrik, optimizasyondan sonra %79'dan %80'e çıkarak modelin pozitif ve negatif sınıfları ayırt etme yeteneğinin çok az da olsa arttığını göstermektedir.
- **Geçen süre:** Eğitim süresi artmış ve 0.5 saniyeden 3.5 saniyeye çıkmıştır. Bu, daha fazla ağaç oluşturulması (500) ve daha yüksek öğrenme oranı nedeniyle modelin hesaplama maliyetinin artmasına bağlıdır. Ancak, performans iyileşmesi göz önüne alındığında bu artış tolere edilebilir düzeyde olabilir.

Hiperparametre optimizasyonu sonucunda modelin doğruluk, hassasiyet ve AUC metriklerinde belirgin iyileşme sağlanmıştır. Duyarlılık metriğinde bir miktar düşüş olmasına rağmen, hassasiyet ve diğer metrikler dengelenmiştir. Eğitim süresi artmıştır, bu da modelin daha karmaşık hale geldiğini ve daha fazla hesaplama süresi gerektirdiğini göstermektedir. Genel olarak hiperparametre optimizasyonu, XGBoost modelinin daha iyi performans göstermesini sağlamıştır.

5.9.3. Hiperparametre Optimizasyonu Sonucunda Açıklamaların Karşılaştırılması

Hiperparametre optimizasyonu sonucunda elde edilen hiperparametrelerle model eğitildikten sonra açıklamaları oluşturan özelliklerin önem değerlerinin hiperparametre optimizasyonundan önceki değerleriyle karşılaştırılması Çizelge 5.11'de verilmiştir. Elde edilen verilere göre hiperparametre optimizasyonu sonrası özelliklerin önem değerlerinin arttığı görülmektedir. Önem değeri en çok artan özellik, "Insert Date Appeal to Prefecture" olmuştur. Ayrıca "Insert Date Appeal to Prefecture" özelliğinin önem değeri "Add Penalty" özelliğinin önem derecesini ve "Appeal to Judge" özelliğinin önem değeri de "Receive Result Appeal from Prefecture" özelliğinin önem derecesini geçmiştir. Bu ise hiperparametre optimizasyonu sonrası ilgili özelliğin hedef değişkeni tahmin etmede daha etkili hale geldiğini göstermektedir.

Çizelge 5.11 Hiperparametre optimizasyonu öncesi ve sonrası özelliklerin önem değerleri

Özellik	Öncesi	Sonrası	Artış Oranı (%)
Payment	0.3262	1.1806	261
Send Fine	0.1344	0.4488	233
Insert Fine Notification	0.1071	0.3031	183
Add Penalty	0.0332	**0.1221**	267
Insert Date Appeal to Prefecture	0.0287	**0.1378**	**380**
Send Appeal to Prefecture	0.0139	0.0244	75
Receive Result Appeal from Prefecture	0.0029	**0.0060**	106
Appeal to Judge	0.0026	**0.0076**	192
Notify Result Appeal to Offender	0.0016	0.0023	43
Create Fine	0.0000	0.0000	0

AS3: XAI ile güçlendirilmiş PPM tekniklerinin karar verme süreçlerine etkisi nedir?

Tahmine dayalı süreç izleme modellerinin sonuçlarından ne yapılması gerekeceği, son kullanıcılar tarafından anlaşılamamaktadır. XAI yöntemleri, tahmin modellerinin bu tahmini yaparken nelere dikkat ettiği ve hangi özelliğin tahminin sonucuna nasıl bir etki ettiğini göstererek bu sonuçların anlaşılmasına yardımcı olur. Elde edilen sonuçların mevcut iş süreçleri üzerinden değerlendirilmesiyle hangi özelliklere önem verilmesi ya da hangi özelliklerin gözden çıkarılması gerektiği bilgisi elde edilerek karar verme süreçlerinin daha etkin ve verimli olmasını sağlar.

Bu çalışma kapsamında tahmine dayalı süreç izleme için kullanılan XGBoost modeline uygulanan SHAP yönteminin açıklamaları ele alınarak en az etkili önem değerine sahip olan ”Create Fine” etkinliği, modele verilen özelliklerden çıkarılarak hiperparametre optimizasyonu işleminden önceki hiperparametre değerleriyle model tekrar çalıştırılmış ve tahmin sonuçlarıyla ilgili olarak Çizelge 5.12'de verilen değerler elde edilmiştir. Bu sonuçlara göre XAI yöntemi ile belirlenen en az öneme sahip özellik çıkarıldığında, hiperparametre optimizasyonu işleminden sonraki sonuçlara yakın değerler elde edilmiştir. Bu da hiperparametre optimizasyonu gibi uzun süren maliyetli işlemlerin yapılmadan daha kısa sürede istenilen tahmin sonuçlarına ulaşılabileceğini göstermektedir. Ayrıca modelin 3.5 saniyeye kıyasla 0.5 saniye gibi daha kısa sürede aynı tahmin değerlerine ulaşması, yüksek boyutlu iş süreçlerini içeren verilerde zamandan ve enerjiden tasarruf edilmesini sağlayabilir.

Çizelge 5.12 ”Create Fine” etkinliği çıkarıldıktan sonraki tahmin sonuçlarının karşılaştırılması

Metrik	**Optimazasyon Öncesi**	**Optimizasyon Sonrası**	**”Create Fine”**
Doğruluk	0.66	0.74	0.74
Hassasiyet	0.58	0.72	0.72
Duyarlılık	0.98	0.74	0.74
F1-skoru	0.73	0.73	0.73
AUC	0.79	0.80	0.80
Geçen süre	0.5	3.5	0.5

XGBoost modelinin SHAP sonuçlarından en çok etkili önem değerine sahip olan ”Payment” etkinliği, modele verilen özelliklerden çıkarılarak hiperparametre optimizasyonu işleminden önceki ve sonraki hiperparametre değerleriyle model tekrar çalıştırıldığında tahmin sonuçlarıyla ilgili olarak Çizelge 5.13’te verilen değerler elde edilmiştir. Bu verilere göre hiperparametre optimizasyonu yapılmış olsa dahi XAI sonuçlarına göre en etkili özelliğin çıkarılması ile Çizelge 5.12’de elde edilen tahmin değerlerinden daha düşük değerler elde edilmiştir. Bu da XGBoost modeline uygulanan XAI yöntemiyle oluşturulan açıklamaların karar verme süreçlerinde ne kadar etkili olduğunu ve bu açıklamaların tahmin sonuçlarına etkisinin hiperparametre optimizasyonundan daha önemli olduğunu göstermektedir.

Çizelge 5.13 ”Payment” etkinliği çıkarıldıktan sonraki tahmin sonuçlarının karşılaştırılması

Metrik	**Optimizasyon Öncesi**	**Optimizasyon Sonrası**
Doğruluk	0.69	0.69
Hassasiyet	0.65	0.65
Duyarlılık	0.74	0.75
F1-skoru	0.69	0.70
AUC	0.68	0.72
Geçen süre	0.7	3.6

Bu çalışma kapsamında tahmine dayalı süreç izleme için kullanılan LSTM modeline uygulanan dikkat mekanizmasının açıklamalarının model her eğitildikten sonra farklı olması nedeniyle XAI yöntemleri ile güçlendirilen derin öğrenme modellerinin karar verme süreçlerine etkisi ölçülememiştir. Bu da vaka çalışması kapsamında karşılaşılan zorluklardan birisi olmuştur.

6. SONUÇLAR

Bu çalışmadan elde edilen sonuçlar, PPM'de XAI'nin gelişen yapısını vurgulayarak, son kullanıcılar için açıklanabilirliğin önemini, metinsel verilerin tahmine dayalı modelleri zenginleştirme potansiyelini ve hem kavramsal hem de ampirik çerçevelerdeki ilerlemeleri göstermektedir. Bu bulgular, tahmine dayalı modellerin yorumlanabilirliğini iyileştirmeyi ve çeşitli iş alanlarında karar verme süreçlerini geliştirmeyi amaçlayan gelecekteki araştırma yönlerine zemin hazırlamaktadır.

Tahmine dayalı süreç izleme için XAI'de kaydedilen ilerlemelere rağmen, çeşitli zorluklar ve gelecekteki araştırma yönleri tespit edilmiştir. Bunların içinde, son kullanıcılar için açıklamaların anlaşılabilirliğinin artırılması, açıklama oluşturma sürecine alan bilgisinin dahil edilmesi, karmaşık olay günlükleri ve süreçleriyle başa çıkılması ve açıklamaların güvenilirliğinin değerlendirilmesi ve doğruluğunun sağlanması yer almaktadır. Bu zorlukların üstesinden gelmek için kullanıcı odaklı, etki alanına duyarlı, ölçeklenebilir ve güvenilir XAI çözümleri geliştirmek amacıyla araştırmacılar ve uygulayıcılar arasında ortak çaba gereklidir.

Bu çalışma kapsamında gerçekleştirilen vaka çalışması için kullanılan makine öğrenimi ve derin öğrenme modellerinde süreç verisiyle çalışmanın, diğer verilerle çalışmaya kıyasla çeşitli zorluklarıyla karşılaşılmıştır. Bu zorluklarında birisi bir etkinliğin öncesinde gerçekleşen etkinlikleri ele alabilmek amacıyla önek işleminin hem makine öğrenimi hem de derin öğrenme modeli için gerçekleştirilmesidir. Bununla birlikte derin öğrenme modellerinin sabit boyutlu girişlere ihtiyaç duyması sebebiyle, önek işlemiyle oluşturulan farklı uzunluklardaki etkinlik dizilerinin aynı uzunluğa getirilmesi için dolgulama adı verilen işlemin bu modeller için gerekli olduğu görülmüştür. Tüm bu işlemler için her bir olay grubunu birlikte ele alarak belirli bir sıra içerisinde işlem yapmak da süreç verileriyle çalışmanın getirdiği zorluklardan birisi olmuştur. Bunlara ek olarak, derin öğrenme modellerinde süreç verisiyle çalışmanın aynı eğitim parametreleriyle sabit açıklamalar oluşturmaması da karşılaşılan zorluklardan bir diğeri olmuştur.

Tahmine dayalı süreç izleme tekniklerinin tahmin kalitesi konusunda bu çalışma kapsamından kullanılan XGBoost modeli, hiperparametre optimizasyonu işleminden önce LSTM modelinden doğruluk ve hassasiyet metriklerinde daha düşük performans gösterirken hiperparametre optimizasyonundan sonra LSTM modeline bu metriklerde çok yakın sonuçlar vermiştir. Ayrıca tahmin kalitesi konusunda AUC metriği açısından her iki modelin performansının da %80'e yakın bir değerde olması, pozitif sınıfları negatif sınıflardan benzer doğrulukla makul bir seviyede ayırabildiklerini göstermiştir. Modellerin tahmin sonuçlarını elde etmesi için geçen süre değerlendirildiğinde ise XGBoost modelinin LSTM modeline göre çok daha hızlı çalışarak benzer tahmin kalitesini çok kısa bir sürede elde etmesi, büyük veri kümelerinde veya hızlı sonuç gerektiren durumlarda XGBoost modelinin daha avantajlı olabileceğini göstermiştir.

Tahmine dayalı süreç izleme tekniklerine XAI yöntemleri uygulandıktan sonra elde edilen açıklamalar konusunda parametre değerlerinin değiştirilmeden bu çalışma kapsamında kullanılan LSTM modelinin her eğitim sonunda farklı çıktılar vermesi ve XGBoost modelinin her eğitim sonunda sabit çıktılar oluşturarak daha istikrarlı olması, bir makine öğrenimi modeli olan XGBoost'un bir derin öğrenme modeli olan LSTM'ye göre daha yorumlanabilir bir tahmine dayalı süreç izleme modeli olduğunu göstermiştir.

Bu çalışmada gerçekleştirilen vaka çalışması kapsamında ayrıca, XGBoost modeline uygulanan SHAP yönteminin açıklamaları ele alınarak en az etkili önem değerine sahip özellik çıkarıldığında hiperparametre optimizasyonu işlemi yapılmadan da performans sonuçları olarak bu işlemden sonrakine benzer bir tahmin kalitesine ulaşıldığı görülmüştür. Bu ise makine öğrenimi modeli olan XGBoost'un açıklamalardan yararlanıldığı takdirde hiperparametre optimizasyonu gibi uzun süren maliyetli işlemlerin yapılmadan ve daha kısa sürede benzer tahmin kalitesine ulaşabildiğini göstermiştir. En çok etkili önem değerine sahip özellik çıkarıldığında ise hiperparametre optimizasyonu yapılsa dahi performans sonuçları olarak daha düşük tahmin kalitesi elde edilmiştir. Bu elde edilen sonuçlar, XGBoost modelinde SHAP yöntemiyle oluşturulan açıklamaların karar verme süreçlerinde ne kadar etkili olduğunu ve açıklamaların tahmin kalitesine etkisinin hiperparametre optimizasyonundan daha önemli olduğunu göstermiştir. LSTM modeline uygulanan dikkat

mekanizmasının her eğitim sonucunda farklı çıktılar üretmesi sebebiyle XAI yöntemiyle desteklenen derin öğrenme modellerinin karar verme süreçlerine etkisi ölçülememiştir. Bu durum ise karşılaşılan zorluklardan birisi olmuştur.

Gelecekteki çalışmaların odak noktası, özellikle derin öğrenme modellerini kullanan mevcut tahmine dayalı süreç izleme tekniklerin içsel mekanizmalarının geliştirilmesi ve buna bağlı olarak tahmin kalitesinin ve yorumlanabilirliğinin artırılması ile XAI yöntemlerinin gelişen yapay zeka teknolojileriyle desteklenerek kullanıcı odaklı, daha anlaşılabilir açıklamaların sunulması olacaktır. Açıklanabilir yapay zeka destekli tahmine dayalı iş süreçleri alanının gelecekteki başarısı, süreç paydaşlarının ihtiyaç ve beklentileriyle uyumlu, anlaşılabilir, güvenilir, bağlamsal olarak ilgili ve eyleme dönüştürülebilir açıklamalar üretme yeteneğine bağlı olacaktır.

KAYNAKLAR

[1] M. Dumas, M. La Rosa, J. Mendling, and H. A. Reijers. *Fundamentals of Business Process Management*. Springer, 2nd edition, **2018**. doi:10.1007/978-3-662-56509-4.

[2] M. Camargo, M. Dumas, and O. González-Rojas. Learning accurate lstm models of business processes. In *Business Process Management*, pages 286–302. Springer International Publishing, **2019**. doi:10.1007/978-3-030-26619-6_19.

[3] G. El-khawaga, M. Abu-Elkheir, and M. Reichert. Xai in the context of predictive process monitoring: An empirical analysis framework. *Algorithms*, 15(6), **2022**. doi:https://doi.org/10.3390/a15060199.

[4] V. Pasquadibisceglie, G. Castellano, A. Appice, and D. Malerba. Fox: a neuro-fuzzy model for process outcome prediction and explanation. In *2021 3rd International Conference on Process Mining (ICPM)*, pages 112–119. **2021**. doi:10.1109/ICPM53251.2021.9576678.

[5] C. Warmuth and H. Leopold. On the potential of textual data for explainable predictive process monitoring. In *Process Mining Workshops*, pages 190–202. Springer Nature Switzerland, **2023**. doi:10.1007/978-3-031-27815-0_14.

[6] N. Mehdiyev and P. Fettke. *Explainable Artificial Intelligence for Process Mining: A General Overview and Application of a Novel Local Explanation Approach for Predictive Process Monitoring*, page 1–28. Springer International Publishing, **2021**. doi:10.1007/978-3-030-64949-4_1.

[7] R. Galanti, M. de Leoni, M. Monaro, N. Navarin, A. Marazzi, B. Di Stasi, and S. Maldera. An explainable decision support system for predictive process analytics. *Engineering Applications of Artificial Intelligence*, 120:105904, **2023**. doi:10.1016/j.engappai.2023.105904.

[8] S. Can, T. Gurgen Erdogan, and A. Kolukısa Tarhan. Systematic review on explainable artificial intelligence for predictive process monitoring (unpublished manuscript). *Data Mining and Knowledge Discovery*, **2024**.

[9] IEEE. Ieee standard for extensible event stream (xes) for achieving interoperability in event logs and event streams. *IEEE Std 1849-2016*, pages 1–50, **2016**. doi:10.1109/IEEESTD.2016.7740858.

[10] W. Van Der Aalst. *Process Mining: Data Science in Action*. Springer, **2016**. doi:10.1007/978-3-662-49851-4.

[11] C. Di Francescomarino. *Predictive Business Process Monitoring*, pages 1–9. Springer International Publishing, **2018**. doi:10.1007/978-3-319-63962-8_105-1.

[12] F. M. Maggi, C. Di Francescomarino, M. Dumas, and C. Ghidini. Predictive monitoring of business processes. In *Advanced Information Systems Engineering*, pages 457–472. Springer International Publishing, **2014**. doi:10.1007/978-3-319-07881-6_31.

[13] C. Di Francescomarino, M. Dumas, F. M. Maggi, and I. Teinemaa. Clustering-based predictive process monitoring. *IEEE Transactions on Services Computing*, 12(6):896–909, **2019**. doi:10.1109/TSC.2016.2645153.

[14] J. Evermann, J.-R. Rehse, and P. Fettke. Predicting process behaviour using deep learning. *Decision Support Systems*, 100:129–140, **2017**. doi:10.1016/j.dss.2017.04.003.

[15] N. Tax, I. Verenich, M. L. Rosa, and M. Dumas. Predictive business process monitoring with lstm neural networks. In *Advanced Information Systems Engineering*, pages 477–492. Springer International Publishing, **2017**.

[16] C. Di Francescomarino, C. Ghidini, F. M. Maggi, and F. Milani. Predictive process monitoring methods: Which one suits me best? In *Business Process Management*, pages 462–479. Springer International Publishing, **2018**.

[17] A. E. Márquez-Chamorro, M. Resinas, and A. Ruiz-Cortés. Predictive monitoring of business processes: A survey. *IEEE Transactions on Services Computing*, 11(6):962–977, **2018**. doi:10.1109/TSC.2017.2772256.

[18] B. F. van Dongen, A. K. A. de Medeiros, H. M. W. Verbeek, A. J. M. M. Weijters, and W. M. P. van der Aalst. The prom framework: A new era in process mining tool support. In *Applications and Theory of Petri Nets 2005*, pages 444–454. Springer Berlin Heidelberg, **2005**. doi:10.1007/11494744_25.

[19] M. La Rosa, H. A. Reijers, W. M. P. van der Aalst, R. M. Dijkman, J. Mendling, M. Dumas, and L. García-Bañuelos. Apromore: An advanced process model repository. *Expert Systems with Applications*, 38(6):7029–7040, **2011**. doi:10.1016/j.eswa.2010.12.012.

[20] W. Rizzi, L. Simonetto, C. Di Francescomarino, C. Ghidini, T. Kasekamp, and F. M. Maggi. Nirdizati 2.0: New features and redesigned backend. *CEUR-WS.org*, 2420:154–158, **2019**.

[21] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, **2016**.

[22] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, **2006**.

[23] Y. Lecun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521:436–444, **2015**. doi:10.1038/nature14539.

[24] J. Zhou, A. H. Gandomi, F. Chen, and A. Holzinger. Evaluating the quality of machine learning explanations: A survey on methods and metrics. *Electronics*, 10(5), **2021**. doi:10.3390/electronics10050593.

[25] F. Doshi-Velez, M. Kortz, R. Budish, C. Bavitz, S. Gershman, D. O'Brien, S. Schieber, J. Waldo, D. Weinberger, and A. Wood. Accountability of ai under the law: The role of explanation. *CoRR*, abs/1711.01134, **2017**.

[26] Defense Advanced Research Projects Agency. Explainable artificial intelligence (xai), **2019**.

[27] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, R. Chatila, and F. Herrera. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, **2019**.

[28] M. Stierle, J. Brunk, S. Weinzierl, S. Zilker, M. Matzner, and J. Becker. Bringing light into the darkness - a systematic literature review on explainable predictive business process monitoring techniques, **2021**. Research in Progress.

[29] N. Mehdiyev, M. Majlatow, and P. Fettke. Interpretable and explainable machine learning methods for predictive process monitoring: A systematic literature review. *arXiv:2312.17584*, **2023**.

[30] B. Kitchenham and S. Charters. Guidelines for performing systematic literature reviews in software engineering. *EBSE Technical Report EBSE-2007-01*, **2007**.

[31] R. Cao, Q. Zeng, W. Ni, H. Duan, C. Liu, F. Lu, and Z. Zhao. Business process remaining time prediction using explainable reachability graph from gated rnns. *Applied Intelligence*, 53:13178–13191, **2023**. doi:10.1007/s10489-022-04192-x.

[32] T.-H. Huang, A. Metzger, and K. Pohl. Counterfactual explanations for predictive business process monitoring. In *Information Systems*, page 399–413. Springer, **2022**. doi:10.1007/978-3-030-95947-0_28.

[33] A. Stevens, J. De Smedt, and J. Peeperkorn. Quantifying explainability in outcome-oriented predictive process monitoring. In *Process Mining Workshops*, pages 194–206. Springer International Publishing, **2022**.

[34] M. Harl, S. Weinzierl, M. Stierle, and M. Matzner. Explainable predictive business process monitoring using gated graph neural networks. *Journal of Decision Systems*, 29(sup1):312–327, **2020**. doi:10.1080/12460125.2020.1780780.

[35] A. Padella, M. de Leoni, O. Dogan, and R. Galanti. Explainable process prescriptive analytics. In *2022 4th International Conference on Process Mining (ICPM)*, pages 16–23. **2022**. doi:10.1109/ICPM57379.2022.9980535.

[36] R. Galanti, B. Coma-Puig, M. de Leoni, J. Carmona, and N. Navarin. Explainable predictive process monitoring. In *2020 2nd International Conference on Process Mining (ICPM)*, pages 1–8. **2020**. doi:10.1109/ICPM49681.2020.00012.

[37] G. Elkhawaga, M. Abu-Elkheir, and M. Reichert. Explainability of predictive process monitoring results: Can you see my data issues? *Applied Sciences*, 12(16), **2022**. doi:10.3390/app12168192.

[38] A. Stevens, J. De Smedt, J. Peeperkorn, and J. De Weerdt. Assessing the robustness in predictive process monitoring through adversarial attacks. In *2022 4th International Conference on Process Mining (ICPM)*, pages 56–63. **2022**. doi:10.1109/ICPM57379.2022.9980753.

[39] S. Weinzierl, S. Zilker, J. Brunk, K. Revoredo, M. Matzner, and J. Becker. Xnap: Making lstm-based next activity predictions explainable by using lrp. In *Business Process Management Workshops*, pages 129–141. Springer International Publishing, **2020**. doi:10.1007/978-3-030-66498-5_10.

[40] R. Velioglu, J.P. Göpfert, A. Artelt, and B. Hammer. Explainable artificial intelligence for improved modeling of processes. In *Intelligent Data Engineering and Automated Learning - IDEAL 2022*, pages 313–325. Springer International Publishing, **2022**. doi:10.1007/978-3-031-21753-1_31.

[41] W. Rizzi, C. Di Francescomarino, and F.M. Maggi. Explainability in predictive process monitoring: When understanding helps improving. In *Business Process Management Forum*, pages 141–158. Springer International Publishing, **2020**. doi:10.1007/978-3-030-58638-6_9.

[42] F. Mannhardt, M. de Leoni, H. A. Reijers, and W. Van Der Aalst. Balanced multi-perspective checking of process conformance. *Computing*, 98:407–437, **2016**.

[43] A. Berti, S. van Zelst, and D. Schuster. Pm4py: A process mining library for python. *Software Impacts*, 17:100556, **2023**. doi:doi.org/10.1016/j.simpa.2023.100556.

[44] W. McKinney. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, pages 56–61. SciPy, **2010**. doi:10.25080/Majora-92bf1922-00a.

[45] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. Fernández del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with numpy. *Nature*, 585:357–362, **2020**. doi:10.1038/s41586-020-2649-2.

[46] T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, **2016**. doi:10.1145/2939672.2939785.

[47] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, **1997**.

[48] Jerome H Friedman. Greedy function approximation: A gradient boosting machine. *Annals of statistics*, pages 1189–1232, **2001**.

[49] I. Teinemaa, M. Dumas, M. La Rosa, and F. M. Maggi. Outcome-oriented predictive process monitoring: Review and benchmark. *ACM Transactions on Knowledge Discovery from Data*, 13-2(2), **2019**. doi:10.1145/3301300.

[50] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(85):2825–2830, **2011**.

[51] S. Lundberg and S. Lee. A unified approach to interpreting model predictions, **2017**. doi:10.48550/arXiv.1705.07874.

[52] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, page 6000–6010. Curran Associates Inc., **2017**.

[53] L. S. Shapley. A value for n-person games. In *Contributions to the Theory of Games II*, pages 307–317. Princeton University Press, **1953**.

[54] S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S. Lee. From local explanations to global understanding with explainable ai for trees. *Nature Machine Intelligence*, 2(1):2522–5839, **2020**.

[55] M. Feurer and F. Hutter. *Hyperparameter Optimization*, pages 3–33. Springer International Publishing, **2019**. doi:10.1007/978-3-030-05318-5_1.

[56] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, **2012**.

[57] J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., **2012**.