KTO KARATAY ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ
ELEKTRİK VE BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
TEZLİ YÜKSEK LİSANS PROGRAMI

KÜÇÜKMUHSİNE ÇAYI ÜZERİNDEKİ GÖZLEM İSTASYONUNDAN ALINAN VERİLERİN DEĞERLENDİRİLMESİ

Nazlı Nida GÜLERYÜZ

Yüksek Lisans Tezi

KONYA

Mart 2023

KÜÇÜKMUHSİNE ÇAYI ÜZERİNDEKİ GÖZLEM İSTASYONUNDAN ALINAN VERİLERİN DEĞERLENDİRİLMESİ

Nazlı Nida GÜLERYÜZ

KTO Karatay Üniversitesi
Lisansüstü Eğitim Enstitüsü
Elektrik Elektronik Mühendisliği Anabilim Dalı
Elektrik ve Bilgisayar Mühendisliği Tezli Yüksek Lisans Programı

Yüksek Lisans Tezi

Tez Danışmanı: Dr. Öğr. Üyesi H. Oktay ALTUN
Tez Eş Danışmanı: Dr. Öğr. Üyesi Şule ERYÜRÜK

Konya
Mart 2023

BİLDİRİM

Enstitü tarafından onaylanan Yüksek Lisans tezimin tamamını veya herhangi bir kısmını basılı veya dijital biçimde arşivleme ve aşağıda belirtilen koşullar dahilinde erişime açma iznini KTO Karatay Üniversitesine verdiğimi bildiririm. Bu izinle, Üniversiteye verilen kullanım hakları dışındaki tüm fikri mülkiyet haklarım bende kalacak ve gelecekteki çalışmalar (makale, kitap, lisans, patent vb.) için tezimin tamamının veya bir bölümünün kullanım hakları yalnızca bana ait olacaktır.

Tezimin bütünüyle kendi çalışmam olduğunu, başkalarının haklarını ihlal etmediğimi ve tezimin tek yetkili sahibi olduğumu beyan ve taahhüt ederim. Telif hakkı bulunan ve sahiplerinden yazılı izinle kullanılması zorunlu olan kaynakları, yazılı izin alarak kullandığımı ve istenildiğinde izinlerin suretlerini Üniversiteye teslim etmeyi taahhüt ederim.

Yükseköğretim Kurulu tarafından yayımlanan "Lisansüstü Tezlerin Elektronik Ortamda Toplanması, Düzenlenmesi ve Erişime Açılmasına İlişkin Yönerge" kapsamında, tezim, aşağıda belirtilen koşullar haricince, YÖK Ulusal Tez Merkezi ve KTO Karatay Üniversitesi Açık Erişim Sisteminde erişime açılır.

☐ Enstitü / Fakülte Yönetim Kurulu kararı ile tezimin erişime açılması mezuniyet tarihimden itibaren 2 yıl ertelenmiştir.[1]

☐ Enstitü / Fakülte Yönetim Kurulunun gerekçeli kararı ile tezimin erişime açılması mezuniyet tarihimden itibaren ... ay ertelenmiştir.[2]

☐ Tezimle ilgili gizlilik kararı verilmiştir.[3][4]

17 Mart 2023

Nazlı Nida GÜLERYÜZ

[1] MADDE 6(1) Lisansüstü tezle ilgili patent başvurusu yapılması veya patent alma sürecinin devam etmesi durumunda, tez danışmanının önerisi ve enstitü anabilim dalının uygun görüşü üzerine enstitü veya fakülte yönetim kurulu iki yıl süre ile tezin erişime açılmasının ertelenmesine karar verebilir.

[2] MADDE 6(2) Yeni teknik, materyal ve metotların kullanıldığı, henüz makaleye dönüşmemiş veya patent gibi yöntemlerle korunmamış ve internetten paylaşılması durumunda 3. şahıslara veya kurumlara haksız kazanç imkanı oluşturabilecek bilgi ve bulguları içeren tezler hakkında tez danışmanının önerisi ve enstitü anabilim dalının uygun görüşü üzerine enstitü veya fakülte yönetim kurulunun gerekçeli kararı ile altı ayı aşmamak üzere tezin erişime açılması engellenebilir.

[3] MADDE 7(1) Ulusal çıkarları veya güvenliği ilgilendiren, emniyet, istihbarat, savunma ve güvenlik, sağlık vb. konulara ilişkin lisansüstü tezlerle ilgili gizlilik kararı, tezin yapıldığı kurum tarafından verilir. Kurum ve kuruluşlarla yapılan işbirliği protokolü çerçevesinde hazırlanan lisansüstü tezlere ilişkin gizlilik kararı ise, ilgili kurum ve kuruluşun önerisi ile enstitü veya fakültenin uygun görüşü üzerine üniversite yönetim kurulu tarafından verilir. Gizlilik kararı verilen tezler Yükseköğretim Kuruluna bildirilir.

[4] MADDE 7(2) Gizlilik kararı verilen tezler gizlilik süresince enstitü veya fakülte tarafından gizlilik kuralları çerçevesinde muhafaza edilir, gizlilik kararının kaldırılması halinde Tez Otomasyon Sistemine yüklenir.

ETİK BEYAN

KTO Karatay Üniversitesi Lisansüstü Eğitim Enstitüsü Tez Hazırlama ve Yazım Kurallarına uygun olarak Dr. Öğr. Üyesi H. Oktay ALTUN danışmanlığında ve Dr. Öğr. Üyesi Şule ERYÜRÜK eş danışmanlığında tarafımdan üretilen bu tez çalışmasında; sunduğum tüm veri, enformasyon, bilgi ve belgeleri bilimsel etik kuralları çerçevesinde elde ettiğimi, tüm değerlendirme, analiz, bulgu ve sonuçları bilimsel usullere uygun olarak sunduğumu, tez çalışmasında yararlandığım kaynakların tümüne bilimsel normlara uygun biçimde atıfta bulunarak kaynak gösterdiğimi, tezimin kaynak gösterilen durumlar dışında özgün olduğunu bildirir, aksi bir durumda aleyhime doğabilecek tüm hak kayıplarını kabullendiğimi beyan ederim.

17 Mart 2023

Nazlı Nida GÜLERYÜZ

TEŞEKKÜR

Tez çalışma sürecinde ve eğitim hayatım boyunca yanımda olan, değerli bilgileri ve tecrübeleri ile bana yol gösteren, her zaman iyi niyeti ve sabrıyla bana yardımcı olan danışman hocam Sayın Dr. Öğr. Üyesi Hüseyin Oktay ALTUN'a, teşekkür ve saygılarımı sunarım. Bilgi paylaşımları için değerli eş danışman hocam Sayın Dr. Öğr. Üyesi Şule ERYÜRÜK'e ve bu tez çalışmasına konu olan veriler için Sayın Dr. Öğr. Üyesi Kağan ERYÜRÜK'e ayrıca teşekkür ederim. Tezime doğrudan veya dolaylı olarak katkıda bulunan Havva YILMAZ KARAGÖZ'e ve Ömer METİN'e teşekkürü borç bilirim.

Bu çalışmayı yaparken, hem modern veri analizi tekniklerini Udemy'e koyduğu derslerden öğrenme fırsatı bulduğum, hem de paylaştığı açık kaynak kodlarından yararlanarak bu teze uyarladığım Vahit KESKİN'e teşekkürü bir borç bilirim.

Tez çalışmalarımda bütün bilgi ve birikimlerini paylaşarak, her daim yanımda olan, tezi bitirmem konusunda ısrarlarını ve desteklerini esirgemeyen, sevgili annem Nazmiye ZEREK ve babam Mustafa ZEREK'e, hayatım boyunca yüzümü güldüren, başarılarıyla her daim gururlandıran ikiz kardeşlerim Necati ve Canset ZEREK'e sonsuz teşekkürlerimi sunarım. Bu süreç içerisinde sabrı ve destekleri için sevgili eşim, Emre GÜLERYÜZ'e ayrıca teşekkür ederim.

Tez araştırmalarım ve çalışma hayatım boyunca bana destek olan, özverili çalışmaları ve etik kurallarıyla yol gösteren kıymetli AB Holding A.Ş. Satınalma Müdürü Emrah AYDEMİR'e ve aynı zamanda destekleri ve güleryüzleriyle yanımda olan çok değerli çalışma arkadaşlarıma teşekkür ederim.

Nazlı Nida GÜLERYÜZ
Mart-2023

ÖZET

Nazlı Nida GÜLERYÜZ

Küçükmuhsine Çayı Üzerindeki Gözlem İstasyonundan
Alınan Verilerin Değerlendirilmesi

Yüksek Lisans Tezi

Konya, 2023

Geçmişten itibaren su ve su kaynakları tüm canlılar için hayati bir öneme sahip olmuştur. Günümüzde ise toplumlar, nüfus artışları ve şehirleşmeden kaynaklı olarak ekolojinin tahrip edildiği bir dünyada kuraklık ve küresel ısınma gibi çeşitli sorunlarla karşı karşıya kalmaktadır. Suya erişim tüm canlılar için gittikçe zorlaşmaya başlamıştır. Bu sebeple su kaynakları büyük ölçüde önem kazanmıştır. Su kaynaklarının verimli kullanılabilmesi ve ihtiyaca göre planlanabilmesi adına makine öğrenmesi teknikleri kullanılarak karşılaştırmalar yapılmıştır. Küçükmuhsine Çayı üzerindeki gözlem istasyonundan alınan veriler öncelikle grafikler ile görselleştirilmiştir. Veri önişleme teknikleri olan veri temizleme, veri bütünleştirme, veri değiştirme ve veri azaltma vb. yöntemlerinden uygun bir ya da birkaç veri önişleme teknikleri seçilmiştir. Veri önişleme işlemleri tamamlandıktan sonra, belli bir veri aralığı eğitim veri seti, belli bir veri aralığı ise test veri seti olarak kullanılmıştır. Makine öğrenmesi teknikleri kullanılarak en iyi tahminlemeyi yapan yöntemler belirlenmiş ve karşılaştırılmıştır.

Anahtar Kelimeler

Makine öğrenmesi algoritmaları, veri önişleme, tahminleme yöntemleri

ABSTRACT

Nazlı Nida GÜLERYÜZ

Evaluation of the Data Received from the Observation Station on Küçükmuhsine Creek

Master's Thesis

Konya 2023

Water and water resources have been crucial for all living beings since ancient times. Today, however, societies are faced with various problems such as drought and global warming due to population growth and urbanization, which have resulted in ecological destruction. Access to water is becoming increasingly difficult for all living beings. Therefore, water resources have gained great importance. To efficiently use and plan water resources according to needs, machine learning techniques will be used to predict future values. The data obtained from the Kucukmuhsine Creek Observation Station is first presented using graphs. Suitable data preprocessing techniques such as data cleaning, data integration, data transformation, and data reduction etc. were applied. After data preprocessing was completed, a certain data subset was used for training various ML models, and another subset was utilized as the test data. Further analysis is performed in order to identify and compare the best methods for time series prediction.

Keywords:

Machine learning algorithms, data preprocessing, prediction methods

İÇİNDEKİLER

TABLOLAR DİZİNİ

ŞEKİLLER DİZİNİ

SİMGELER DİZİNİ

Simgeler	Açıklama
Q	Debi (m^3/s)
ρ	Korelasyon Katsayısı
km	Kilometre
m	Metre
m^3	Metreküp
μ	Ortalama
R^2	R-Kare (R-Squared)
s	Saniye
σ	Standart Sapma
σ^2	Varyans
z	Z-Skor Normalizasyon

KISALTMALAR DİZİNİ

Kısaltmalar	Açıklama
AGİ	Akım Gözlem İstasyonu
AI	Yapay Zeka (Artifical Intellengent)
ANFIS	Adaptif Ağ Tabanlı Bulanık Çıkarım Sistemi (Adaptive Neural Fuzzy Inference System)
ANN	Yapay Sinir Ağı (Artificial Neural Network)
ARIMA	Otoregresif Tümleşik Hareketli Ortalama(AutoRegressive Integrated Moving Average)
AR4	Otoregresif 4 (Autoregressive 4)
CART	Sınıflandırma ve Regresyon Ağaçları (Classification and Regression Trees)
CATBOOST	Kategorik Arttırma (Categorical Boosting)
CCNN	Basamak Korelasyon Sinir Ağı (Cascade Correlation Neural Networks)
DL	Derin Öğrenme (Deep Learning)
DSİ	Devlet Su İşleri
DVM	Destek Vektör Makineleri
EİE	Elektrik İşleri Etüt İdaresi
FFBP	İleri Beslemeli Geri Yayılımlı Sinir Ağları (Feed Forward Back Propagation)
GBM	Gradyan Arttırma Yöntemi (Gradiant Boost Method)
GBYYSA	Geri Beslemeli Yinelemeli Yapay Sinir Ağları
GFF	Genelleştirilmiş İleri Beslemeli (Generalized Feed Forward)
KDD	Veri Tabanlarında Bilgi Keşfi (Knowledge Discovery in Databases)
KNN	K-En Yakın Komşuluk (K-Nearest Neighbors)
KRA	Klasik Regresyon Analizi
LGBM	Hafif Gradyan Arttırma Yöntemi (Light Gradiant Boost Method)
LSTM	Uzun Kısa Süreli Hafıza (Long Short Term Memory)
MAE	Ortalama Mutlak Hata (Mean Absolute Error)

MFL	Mamdani Bulanık Mantık (Mamdani-Fuzzy Logic)
MLP	Çok Katmanlı Algılayıcı (Multilayer Perceptron)
MSE	Ortalama Kare Hatası (Mean square error)
NASA	Ulusal Havacılık ve Uzay Dairesi (National Aeronautics and Space Administration)
PNN	Olasılıklı Sinir Ağı (Probabista Neural Network)
RBF	Radyal Tabanlı İşlev Ağı (Radial Basis Function)
RL	Pekiştirmeli Öğrenme (Reinforcement Learning)
RMSE	Kök Ortalama Kare Hatası (Root mean square error)
RNN	Yinelemeli Sinir Ağları (Recurent Neural Networks)
RTİYSA	Radyal Temel İşlemcili Yapay Sinir Ağları
RTYSA	Radyal Tabanlı Yapay Sinir Ağları
SMRGT	Bulanık Kural Üretim Tekniği
SVM	Destek Vektör Makineleri (Support Vector Machine)
YSA	Yapay Sinir Ağları (Artificial neural networks)
XGBM	Ekstrem Gradyan Arttırma Metodu (Extreme Gradiant Boost Method)

1 GİRİŞ

Dijital teknolojilerin gelişmesi ve ucuzlamasıyla hayata dair bir çok olgu kayıt altına alınabilmektedir. Bu veriler, istatistiksel metodlarla işlenerek anlamlı bilgilere dönüştürülebilmektedir. Veri bilimi olarak da adlandırılan yeni bir multidisipliner alan oluşmuştur ve bu alan gittikçe büyümektedir. Biz bu çalışmada Konya il sınırları içinde kalan ve Apa Barajı'nı besleyen Küçük Muhsine Çayı üzerindeki gözlem istasyonundan alınan verileri analiz etmek ve bu verilerden yola çıkarak gelecek verileri tahmin etmeye çalıştık.

Kuraklık ve küresel ısınma ile birlikte su ve su kaynaklarının daha verimli kullanılabilmesi adına su kaynaklarının planlanması ve yönetimi büyük ölçüde önem kazanmıştır. Bu tür çalışmalarda geçmiş tarihlere ait veriler temin edilirken çeşitli sebeplerden dolayı eksiklikler olabilir. Bu eksikliklere ölçüm cihazlarındaki arızalar, iklimsel zorluklar ve ulaşım zorlukları gibi etkenler sebep olabilmektedir (Dursun & Karabatak, n.d.; Bakış & Göncü, 2015). Bu yüzden çalışmalarımızda veri önişleme yöntemlerinden uygun olan yöntemleri kullanarak veriyi işleme kısmına hazırlamamız gerekir.

Konya ilinde bulunan, Devlet Su İşleri (DSİ) Etüt Planlama ve Tahsisler Dairesi Başkanlığı Rasatlar Şube Müdürlüğü'nden alınan 1975-2017 yılları arasındaki günlük, aylık ve yıllık olarak debi (m^3/s) miktarları ayrı ayrı tablolarda elde edilmiştir. Yapılan çalışma çerçevesinde öncelikle ayrı olarak bulunan tablolar bir Excel içerisinde veriyi kullanabileceğimiz, tarihe göre sıralanmış olarak yeniden düzenlenmiş ve bir veri kümesi oluşturulmuştur. Veri madenciliği yöntemlerinden olan veri önişleme yöntemleri kullanılarak veri tahminleme işlemine hazırlanmıştır. Daha sonra yedi farklı metot ile tahminleme işlemi gerçekleştirilmiştir. Çıkan hata değerlerine göre sonuçlar

karşılaştırılmıştır.

Ülkemizde temiz su kaynaklarının özellikle dağlık bölgelerde dağılımları ve miktarları değişkenlik göstermektedir. Ayrıca günümüzdeki yüksek ivmeli nüfüs artışından, sanayileşmeden ve şehirleşmeden kaynaklı olarak tarımda ve içme suyu olarak kullanılan temiz su kaynaklarının kontrollü kullanılması ve planlanması gittikçe zorlaşmaya başlamıştır. Özellikle yaşanılan küresel ısınma sebebiyle yağış ve buharlaşma gibi etkenler farklılık göstermektedir. Bu sebeple bu kaynakların planlı bir şekilde kullanılabilmesi ve taşkınların önlenebilmesi gibi amaçlar için akarsuların hem nicelik hem de nitelik bakımından akım özelliklerinin tahmin edilebilmesi gerekmektedir.

Ülkemiz akarsularına ait akım ölçümleri Elektrik İşleri Etüt İdaresi (EİE) ve DSİ tarafından kurulan akım gözlem istasyonlarında yapılmaktadır. Debi birim zaman içerisindeki akışkanın hacmini ifade eder. Ayrıca ölçümü zor ve masraflı olmasından dolayı bazen akım gözlem istasyonlarında bir kesitten alınan seviye ölçümleri ile debi arasındaki eğimden tahmin edilerek debi ölçümleri yapılabilir. Fakat birim kesitten alınan ölçümler kesin doğru veriyi vermeyebilir. Ölçümlerin doğruluğu için birden fazla kesitten alınan örnek ile eğim, bitkilerin büyümesi ve zemin durumları gibi etkenlerin de göz önünde bulundurulması gerekmektedir. (Gemici, Ardıçlıoğlu, & Kocabaş, 2013)

2 İLGİLİ ÇALIŞMALAR

Küçükmuhsine Çayı üzerinde yapılan bir çalışamaya göre çok katmanlı algılayıcı (MLP), adaptif ağ tabanlı bulanık çıkarım sistemi (ANFIS) ve uzun kısa süreli hafıza (LSTM) metotları kullanılmış, en başarılı modelin LSTM modeli olduğu görülmüştür (Asaad, Eryürük, & Eryürük, 2022). Karadeniz Bölgesinde bulunan ve büyük taşkınlara sebebiyet vermiş olan Aksu Deresi üzerinde çalışmalar yapılmış, yapay sinir ağları (YSA) ve regresyon teknikleri kullanılmıştır. Kullanılan yöntemler arasında çok katmanlı YSA modeli, performansı en yüksek olan modelleme yöntemi olmuştur (Babacan & Fatih, 2022).

Bir başka çalışma Karadeniz Bölgesi'nde yer alan Haldizen Deresi Şerah Akım Gözlem İstasyonu'ndan alınan akım verileri ile çok değişkenli uyarlanabilir regresyon eğrileri kullanılarak tahmin edilmiş ve sonuçlar klasik regresyon analizi (KRA) ile karşılaştırılmıştır. Yapılan bu çalışma ile çok değişkenli uyarlanabilir regresyon eğrileri yönteminin KRA yöntemine göre daha iyi sonuçlar verdiği sonucuna ulaşılmıştır. (Nacar, Kankal, & Hınıs, 2018)

2017 yılında bir çalışmaya göre ise akım verilerinde genel olarak eksik verilerin bulunduğu görülmüş ve bu eksik verilerin doldurulması için de tahminleme metodu kullanılmıştır. Fırat Havzası'nın yukarı ve orta Fırat kısımlarında bulunan beş adet akım gözlem istasyonlarındaki (AGİ) eksik akım verileri, debi süreklilik çizgileri ve regresyon modelleri ile tahmin edilmiştir. Elde edilen bulgulara göre her iki yöntemde de yüksek determinasyon R^2'ye sahip oldukça başarılı sonuçlar elde edildiği görülmüştür. Ayrıca verilerin alındığı dört istasyon için regresyon modelleri, kalan bir istasyon için

ise debi süreklilik çizgileri yöntemi ile yapılan tahminlerin ise serilerin temel istatistiksel özelliklerini korumada daha başarılı olduğu görülmüştür. (Tosunoğlu, İspirli, Gürbüz, & Şengül, 2017)

Başka bir çalışmada ise Akdeniz Bölgesi'nde bulunan Köprüçay Akarsuyu'nun bağlı olduğu akım gözlem istasyonundan alınan veriler doğrultusunda ileri beslemeli geriye doğru hata yayılmalı yapay sinir ağları ile akım tahminlemesi yapılmıştır. Günlük akarsu verileri kullanılarak oluşturulan altı modelleme için YSA ve lineer regresyondan elde edilen sonuçlar karşılaştırılmıştır. Elde edilen sonuçlara göre tahmin süresi arttıkça modelin hassasiyetinin azaldığı görülmüştür. Bu sonuçlara göre ise YSA modelinin akarsu tahmininde kullanabileceğinden bahsedilmiştir (Demirpençe, 2015).

Kızılırmak Nehri üzerinde yapılan bir çalışmada ise nehrin yan kollarından seçilen beş farklı istasyondan alınan veriler doğrultusunda çok katmanlı yapay sinir ağları (ÇKYSA), radyal tabanlı yapay sinir ağları (RTYSA) ve ANFIS modelleri ile tahminleme yapılmıştır. Debi tahmini için YSA ve bulanık mantık modellerinin kullanılan diğer modellemelere göre daha başarılı sonuçlar verdiği gözlemlenmiştir. Modellemelerde en başarılı performansı ANFIS modeli göstermiştir (Gemici et al., 2013). Başka bir çalışmada Aşağı Sakarya Havzası'ndaki küçük akarsuların akım debileri yapay sinir ağları (YSA) yöntemi ile tahmin edilerek enerji potansiyelleri tespit edilmiştir. Tahmin için, YSA ve çoklu doğrusal regresyon analizi metotları kullanılmıştır. İleri beslemeli geriye doğru hata yayılmalı YSA ve geri beslemeli yinelemeli yapay sinir ağları (GBYYSA) algoritmaları kullanılmıştır. Özellikle küçük akış değerlerinin tahmininde yapay sinir ağlarının regresyona nazaran daha başarılı olduğu görülmüştür (Öncül, 2008).

Sakarya Havzası'nda yapılan başka bir çalışmada ise otoregresif 4 (AR4) ve YSA modelleri günlük akımların tahmini için kullanılmış ve üç farklı yapay sinir ağı algoritması seçilmiştir. Bunlar; ileri beslemeli geri yayınımlı YSA, radyal temel işlemcili YSA

(RTİYSA) ve geri beslemeli yinelenen (GBYYSA) yapay sinir ağlarıdır. Elde edilen sonuçlarda GBYYSA modeli her istasyonda da en iyi sonucu vermiştir (Toluk, 2006). Dicle Havzası'nın bir alt havzası olan Zap Suyu Havzası'nda yapılan bir çalışmada akarsu debi verilerindeki eksik verilerin tamamlanması için korelasyona bağlı regresyon analizi ve drenaj alan oranı metodu kullanılmıştır. Kullanılan yöntemlerin eksik veri sayısının az olduğu durumlarda çok iyi sonuçlar verdiği görülmüştür (Bakış & Göncü, 2015).

Mersin Lamas Nehri üzerinde yapılan çalışmalarda aylık ortalama akım verileri kullanılmış olup, YSA, destek vektör makineleri (DVM) ve derin öğrenme (DL) modellemeleri yapılmıştır. Karşılaştırma sonuçları incelendiğinde, en iyi tahmin modelinin DL olduğu görülmüştür (Çubukçu, Demir, & Sevimli, 2022). İstanbul Göksu Deresi üzerinde yapılan bir çalışmada aylık ortalama debi ve aylık maksimum yağış tahmini için çoklu resgresyon analizi ve YSA analizinde çok katmanlı algılayıcı (MLP), olasılıklı sinir ağı (PNN), radyal tabanlı işlev ağı (RBF) ve genelleştirilmiş ileri beslemeli (GFF) olmak üzere dört farklı analiz yöntemi kullanılmıştır. Göksu Deresi için ortalama debi tahmininde YSA ve GFF analiz metotlarının kullanılabileceği ve debilerin genel ortalamaları dikkate alındığında iyi sonuçlar verdiği görülmüştür (Fırat, 2019).

İsviçre'deki Birs Nehri üzerinde bulunan dört istasyondan alınan akım verileri kullanılarak yapılan bir çalışmada ise bulanık mantık genelleştirilmiş regresyon sinir ağı ve ileri beslemeli geri yayılımlı sinir ağları modelleri kullanılmıştır. Kullanılan yöntemler arasında ileriye beslemeli geriye yayılım sinir ağları (FFBP) modeli en başarılı sonucu vermiştir (Turan & Yurdusev, 2009). Çalışma alanı olarak ABD'nin Stilwater Nehri seçilmiş olan başka bir çalışmada ise çoklu doğrusal regresyon (MLR), yapay sinir ağı (ANN), M5 karar ağacı (M5T), uyarlanabilir nöro-bulanık çıkarım sistemi (ANFIS), Mamdani bulanık mantık (M-FL), basit üyelik fonksiyonları ve bulanık kural üretim tekniği (SMRGT) modelleri kullanılarak tahminlemeler yapılmıştır. M-FL ve basit üye-

lik fonksiyonları ile SMRGT modelinin nehir akış tahmininde diğer modellere göre daha iyi performans gösterdiği görülmüştür (Üneş et al., 2020).

İran'ın kuzeybatısında bulunan Zarrinehrud Nehri'nden alınan veriler kullanılarak yapılan bir çalışmada MLP, RBF ve tahmin için SVM modelleri kullanılmış, ve neticede MLP ve RBF modellerinin diğerlerine kıyasla daha iyi tahminleme yaptığı sonucu elde edilmiştir (Ghorbani, Zadeh, Isazadeh, & Terzi, 2016). Avusturalya'da Dulhunty Nehri ve Herbert Nehri üzerinde yapılan bir çalışmada basamak korelasyon sinir ağı (CCNN) ve rastgele orman (RF) modelleri kullanılmıştır. Elde edilen bulgulara göre ise CCNN modelinin daha başarılı olduğu gözlemlenmiştir (Ghorbani et al., 2020).

Pakistan Kabil Nehri üzerinde yapılan bir çalışmaya göre ARIMA modeli kullanılarak bir makine öğrenmesi yaklaşımı sunulmuştur. ARIMA modelinin Kabil Nehri'nin su akışını tahmin etmek için etkili bir yöntem olduğu gösterilmiştir. (Musarat et al., 2021) Hindistan Mahanadi Nehri üzerinde yapılan bir çalışmaya göre nehir akışı tahmini için yapay sinir ağları (ANN) ve adaptif nöro-fuzzy çıkarım sistemleri (ANFIS) kullanılmıştır. ANN'nin ANFIS'ten daha yüksek tahmin doğruluğuna sahip olduğu görülmüştür. (Pramanik & Panda, 2009) Çin'de bulunan Hun Nehri ve Yukarı Yangtze Nehri'nde yapılan çalışmalarda LSTM ağları kullanılarak farklı nehir akışı verileri kullanılarak nehir akışı tahminleri yapılmıştır. LSTM ağlarının, nehir akışı tahmininde kullanılan diğer geleneksel yöntemlere göre daha yüksek tahmin doğruluğu elde ettiği gözlemlenmiştir. (Xu et al., 2020)

3 VERİ MADENCİLİĞİ VE YAPAY ZEKA

Teknolojinin ilerlemesiyle beraber her alanda toplanan veriler çoğalmaya başlamış ve kaydedilmeye başlanmıştır. Bu sebeple veri depolama için bigisayarların kapasiteleri de büyük oranda artış göstermiştir. Bu artışa sürekli olarak kaydedilen dijital veriler depolanmaya başlanmış ve zamanla yetersiz kalmıştır. Bu da veri kayıtları için bulut sistemlerinin ortaya çıkmasına sebep olmuştur. Bu büyük veriler kaydedilmiş fakat kullanılabilmesi için veri madenciliği yöntemleri önem kazanmıştır. Başka bir ifadeyle büyük veriler arasında gizli kalmış ve değerli verilerden bilgiler elde edilmesi amaçlanmıştır (Zaiane, 1999). Bu duruma veri tabanlarında bilgi keşfi (KDD) adı verilir. Veri madenciliği makine öğrenmesi, matematik, istatistik, görselleştirme ve örüntü tanıma gibi birçok disiplinin algoritmalarla tanımlanması anlamına da gelebilir. Süreçlere bağlı olarak geleceğe yönelik yapılan tahminleme işlemleri için yapay zeka (AI) yöntemleri kullanılır. Yapay zeka, insan zekasını taklit ederek toplanan verilere göre sürekli kendini iyileştirebilen sistemler anlamına gelir (Eren & Aksangür, 2019).

Veri madenciliğinde öncelikle elimizdeki verileri tanıma işlemlerini gerçekleştirmeliyiz. Farklı tiplerde bulunan veriler öncelikle kullanacağımız yöntemler ve amaçlarımız göz önünde bulundurularak ve elimizdeki verinin doğası doğrultusunda en uygun şekilde bir veri kümesi haline getirilmelidir. Bu aşamadan sonra veri hakkında bilgiler toplanabilir. Verinin içerisindeki eleman sayıları, verinin boyutu, indeks bilgileri, tip bilgileri, satır ve sütun sayıları, içerisindeki eksik veri bilgileri, mod, medyan, standart sapma ve varyans gibi birçok bilgiye kolaylıkla ulaşabiliriz. Verimizi tanıma işlemleri ve edindiğimiz bilgiler doğrultusunda veri görselleştirme işlemlerini yapabiliriz.

3.1 Veri Önişleme Yöntemleri

Günümüzde veriler büyük ölçekli ve yüksek boyutlara sahip olarak kaydedilmektedir. Büyük veriler, içerisinde stratejik öneme sahip bilgiler barındırmaktadır. Bu sebeple veri madenciliği bahsedilen büyük veri tabanlarındaki önemli ve gizli bilgileri açığa çıkarmak için birçok analizi temelinde barındıran bir yöntem ve bir süreçtir. (Zhou, 2003)

Veri madenciliğinde güvenilirliğin artırılması ve veri kalitesinin iyi olması için veri önişleme yapılmalıdır. Veri kalitesini düşüren sorunlar arasında, gürültü, sapan veri, eksik veri, tekrarlı veri, veri iletim hataları, teknolojik sınırlamalar, veri isimlendirme ve yapısındaki uyumsuzluklar ve anlamlandırılamayan veri gibi nedenler yer alır. Veri önişlemenin temeli bir veriyi daha iyi anlamak ve anlatmaktır. Bu süreçte verinin merkezi eğilimi ve verinin dağılımı önem arz etmektedir. (Oğuzlar, 2003)

Birçok veri önişleme teknikleri vardır. Veriyi iyi anlamak bu süreçte oldukça önemlidir. Sonrasında ise teknikler arasında uygun olan seçim veya seçimler yapılır. Veri önişleme teknikleri arasında; veri temizleme, veri bütünleştirme, veri değiştirme ve veri azaltma gibi yöntemler yer almaktadır. Veri temizleme, eksik verileri tamamlama, tutarsız verileri kaldırma, hatalı verileri düzelme gibi yöntemler ile verideki gürültü ve tutarsızlıklar için düzeltme işlemleri yapılır. Başka bir teknik olan veri birleştirme veya veri bütünleştirme, artık verilerin ortadan kaldırılmasını ve farklı veri tabanlarındaki verilerin birleştirilmesini amaçlamaktadır. Veri değiştirme tekniği ise, veriyi daha anlaşılabilir şekilde ifade etme yani normalizasyon olarak değerlendirilebilir. Veri indirgeme tekniği ise veri azaltma olarak düşünülebilir. İçerisinde veri bütünleştirme, nitelik alt kümesi seçme ve kümeleme ile boyut azaltma gibi işlevleri amaçlar. (Chakrabarti et al., 2008; Ö. Yavuz, 2021; Dogan & Birant, 2021)

3.1.1 Veri Temizleme

Veri temizlemenin amacı, eksik verilerin tamamlanması, hatalı verilerde düzeltme yapılması ve tutarsız verilerin kaldırılması olarak da özetlenebilir. Hatalı verilerin nedenleri arasında ise; ölçüm cihazlarındaki hatalar, veri giriş problemleri, veri girişi sırasında kullanıcıların hatalı yorumları, veri iletim hataları, teknolojik sınırlamalar ve veri isimlendirmesinde veya yapısında bulunan uyumsuzluklar gibi sebepler sıralanabilir. İstatistiksel olarak çıkarımlar sonucunda da hatalar bulunabilir fakat genel olarak alan uzman bilgisi gerektirmektedir.

3.1.2 Eksik Verilerin Tamamlanması

Eksik verilerin tamamlanması yönteminin ortaya çıkmasının sebebi ise günlük hayatta elimizde olan verilerin genel olarak eksiklikler ve hatalar içermesidir. Eksik verilerin tamamlanması metodu da bu eksik verilerin doldurulması, hatalı verilerin düzeltilmesi veya tutarsız verilerin kaldırılması gibi yöntemleri içerir. Eksik veri tamamlama metodu için kaydı yoksayma, elle doldurma, global bir değer ile doldurma, nitelik ortalamasıyla doldurma veya olası değerlerle doldurma (örn., regresyon ve Bayes tahminlemesi) gibi yöntemler izlenebilir.

3.1.3 Veri Bütünleştirme

Veri bütünleştirme, veri tabanlarını birleştirme ve artık verileri ortadan kaldırma olarak özetlenebilir. Scheme bütünleştirme işlemlerinde varlık tanımlama ve metadata kullanımı yer alır. Tekrarlı veri veya artık veri temizleme kısmında ise korelasyon analizi veya Chi-Square test yöntemleri kullanılabilir.

3.1.4 Veri Değiştirme

Veri değiştirme, kullanılan veriyi daha anlaşılabilir bir halde ifade etme anlamına gelir. Düzeltme (İng., smoothing), birleştirme (İng., aggregation), genelleme, normalizasyon ve nitelik oluşturma gibi işlemler yer alır. Normalizasyon işlemleri genellikle üç yöntem ile değerlendirilir. Bunlar; max-min normalizasyon, z-score normalizasyon ve ondalıklı ölçekleme ile normalizasyon olarak bilinir. Nitelik oluşturma ise elimizde bulunan veri kümesine göre kategorize edilir. Veri azaltma metodu orijinal verinin özellikleri korunarak veri boyutunu azaltmaya yönelik yapılan çalışmalar olarak değerlendirilebilir. Veri kümesi birleştirme, nitelik alt kümesi seçimi, ayrıştırma ve içerik hiyerarşisi geliştirme, veriyi modellerle veya görsel olarak ifade etme ve boyut azaltma gibi yöntemler kullanılır. Bütün yapılacak veri önişleme işlemleri için harcanan zaman, veri madenciliği yaparken kazanacağımız zamanı geçmemelidir.

4 MAKİNE ÖĞRENMESİ TEKNİKLERİ VE HATA METRİKLERİ

Makine öğrenmesi teknikleri denetimli öğrenme, denetimsiz öğrenme, yarı denetimli öğrenme ve pekiştirmeli öğrenme olarak dört başlık altında incelenebilir. Bu tezde kullanılan hata metrikleri ise R-kare (R^2), ortalama kare hatası (MSE), kök ortalama kare hatası (RMSE) ve ortalama mutlak hata (MAE) olarak dört kısımda incelenmiştir.

4.1 Makine Öğrenmesi Teknikleri

4.1.1 Denetimli Öğrenme

Denetimli öğrenme, sistemin etiketli verileri kullanarak eğitiminin sağlanması sonucu öğrenen bir modeldir. Sistem, girdi değişkenlerini çıktı değişkenlerine eşleme işlevini öğrenir. Genel olarak sınıflandırma ve regresyon için kullanılmaktadır. Basit Bayes, yapay sinir ağları, destek vektör makineleri, lojistik regresyon, multinom basit Bayes, rastgele orman ve karar ağaçları denetimli öğrenme modelinde en çok kullanılan teknikler olarak bilinir. (Caruana & Niculescu-Mizil, 2006; Yürek, Birant, & Yürek, 2021)

4.1.2 Denetimsiz Öğrenme

Denetimli öğrenme modelinin aksine, denetimsiz öğrenme modelinde sistem, etiketsiz verileri kullanarak öğrenme sağlar. Bu öğrenme çeşidinde sistem, çıkışları bilmediği için genellikle tanıma ve sınıflandırma tekniği olarak kullanılmaz. Genel kullanımında birleştirme, kümeleme, olasılık, yoğunluk tahmini, özniteliklerin arasındaki bağlantının bulunması ve boyut indirgeme gibi amaçlar yer alır (Bilgin, 2017). Denetimli öğrenme modelinde kullanılan algoritmaların bir çoğu aynı şekilde denetimsiz öğrenme mode-

linde de kullanılabilir (Chao, 2011; Teni, Ariffin, Ahmad, & Masood, 2020).

4.1.3 Yarı Denetimli Öğrenme

Bu öğrenme modeli, işaretlenmemiş veri sayısının işaretlenmiş veri sayısına göre çok daha fazla olduğu durumlarda kullanılan bir yöntemdir. Bu sebeple çok daha az sayıdaki işaretlenmiş veri kullanılarak tahminleme ve sınıflandırma yapılabilir. Çok sayıda deneme yaparak öğrenme gerçekleşir. (Şengül, 2022)

4.1.4 Pekiştirmeli Öğrenme

Eğitilecek varlık için ödülü en üst seviyeye çıkaracak şekilde davranışları deneme yanılma yöntemiyle öğrenen bir algoritmaya dayanır. Model davranışları deneme yanılma yolu ile öğrendikçe doğruluk performansı artar ve en yüksek ödüle ulaşmak için hangi eylemleri gerçekleştireceğini öğrenir. (Kaelbling, Littman, & Moore, 1996)

4.2 Hata Metrikleri

4.2.1 Ortalama Kare Hatası (MSE)

Tahminleme performansını ölçmeye yarayan kriterlerden biridir. Bu değer 0'a ne kadar yakın ise model o derece iyi performans gösterir. Her zaman pozitif değerler alır (Sevinç & Buket, 2021). Ayrıca MSE gerçek değerlerle tahmin edilen değerler arasındaki farkların karelerinin ortalama değeridir. n parametresi toplam veri noktalarının sayısını ifade ederken, y_i parametresi ise gerçek değerleri temsil eder. Tahmin edilen değerler, $\hat{y}_i$ parametresi ile gösterilir. $\sum_{i=1}^{n}$ toplam sembolü, tüm veri noktalarını kapsar (Tüzemen & Yıldız, 2018).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{4.1}$$

4.2.2 Kök Ortalama Kare Hatası (RMSE)

Tahmin edilen değer ve gerçek değer arasındaki ortalama kare farklarının karekökünün alınması ile bulunur. Daha iyi sonuç vermesi için hata değerlerinin düşük olması gerekir ve ayrıca n ifadesi örnekleme sayısını gösterirken, y_i ifadesi gerçek değerleri ve $\hat{y}_i$ ifadesi ise tahmin edilen değerleri temsil eder. (Çolakoğlu, 2020)

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{4.2}$$

4.2.3 R-Kare (R^2)

Tahmin değerleri ve gerçek değerler arasındaki ölçüm kriterlerinden bir tanesidir. Değer 1'e ne kadar yakın ise model o kadar hassastır ve formül içerisindeki n ifadesi örnekleme sayısını, y_i ifadesi gerçek değerleri, $\hat{y}_i$ ifadesi tahmin edilen değerleri ve $\bar{y}$ ifadesi gerçek değerlerin ortalamasını ifade eder. (Sevinç & Buket, 2021)

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{4.3}$$

4.2.4 Ortalama Mutlak Hata (MAE)

Her veri noktası ile tahmin arasındaki mesafenin mutlak değerlerin ortalamasını ifade eder. Değer ne kadar küçükse model o derece iyi sonuçlar elde edildiğini gösterir ve ayrıca denklemdeki n ifadesi örnekleme sayısını, y_i ifadesi gerçek değerleri ve $\hat{y}_i$ ifadesi tahmin edilen değerleri gösterir. (Çolakoğlu, 2020)

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{4.4}$$

5 YÖNTEMLER VE METOT

5.1 Veri Kümesi

Konya ili, Orta Anadolu'da 58.850 km² toprak büyüklüğü ile Türkiye'nin %7'sini oluşturmaktadır. Türkiye'nin %7'si kadar bir alana sahip olan Konya ili 36°51' ve 39°29' kuzey enlemleri ile 31°36' ve 34°52' doğu boylamları arasında yer almaktadır. (Koçak, 2019)

(a) Küçükmuhsine Çayı'nın konumu

(b) Küçükmuhsine Çayı akım gözlem istasyonu

Şekil 5.1: Küçükmuhsine Çayı'na ait görseller (Koçak, 2019)

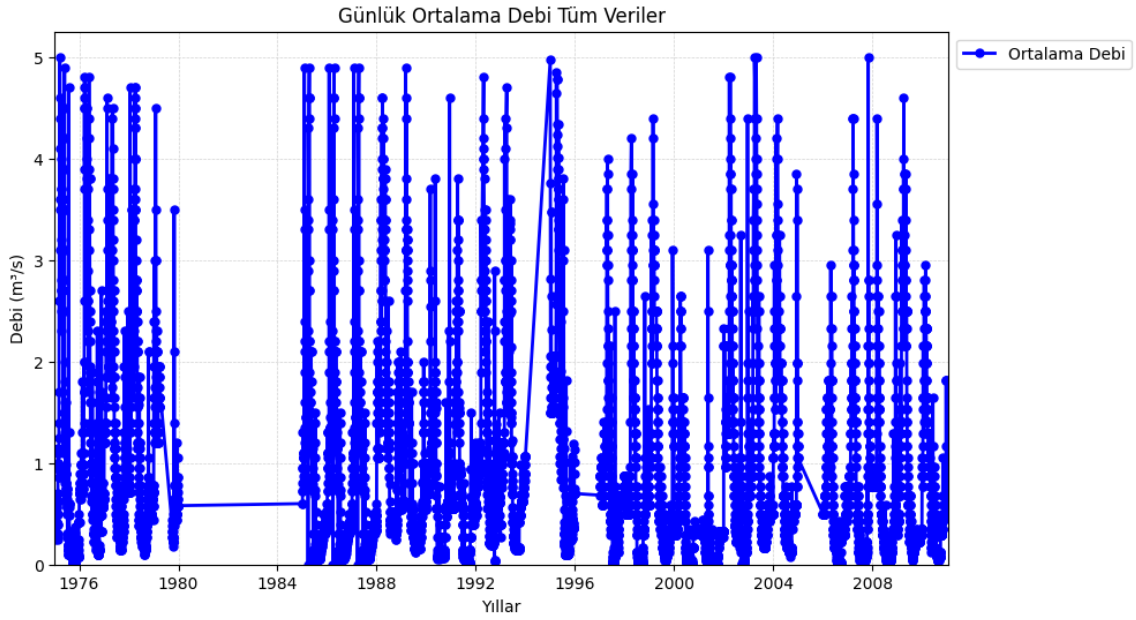
Konya ilinde bulunan, DSİ Etüt Planlama ve Tahsisler Dairesi Başkanlığı Rasatlar Şube Müdürlüğü'nden alınan Konya Beyşehir Yolu'ndan Altınapa Barajı'na varmadan sağa, Başarıkavak Yolu'ndan 8 km Ulumuhsine Köprüsü'nün 50 m membasında bulunan D16A100 istasyon numaralı enlemi 37:55:28 ve boylamı 32:16:9, Meram Ç. - Küçükmuhsine isimli akım gözlem istasyonundan alınan 1975-2017 yılları arasındaki

günlük, aylık ve yıllık olarak debi (m^3/s) miktarları ayrı ayrı tablolarda organize edilmiştir. Veri işlenmesi için veriler birleştirilmiştir ve kullanılabilecek en uygun şekilde tarihlere göre sıralanmıştır.

Bu tez kapsamında yapılan çalışmalar, günlük verilerin kullanıldığı yöntemler, aylık verilerin kullanıldığı yöntemler ve aylık verilere özellik verileri eklenerek kullanılan yöntemler olarak üç kısımdan oluşmaktadır. Aylık ortalama debi verileri kullanılırken, ortalama maximum sıcaklık, ortalama nispi nem, maximum yağış, toplam yağış, toplam buharlaşma ve ortalama güneşlenme süresi gibi veriler modele eklenmiştir.

5.1.1 Günlük Debi Verileri

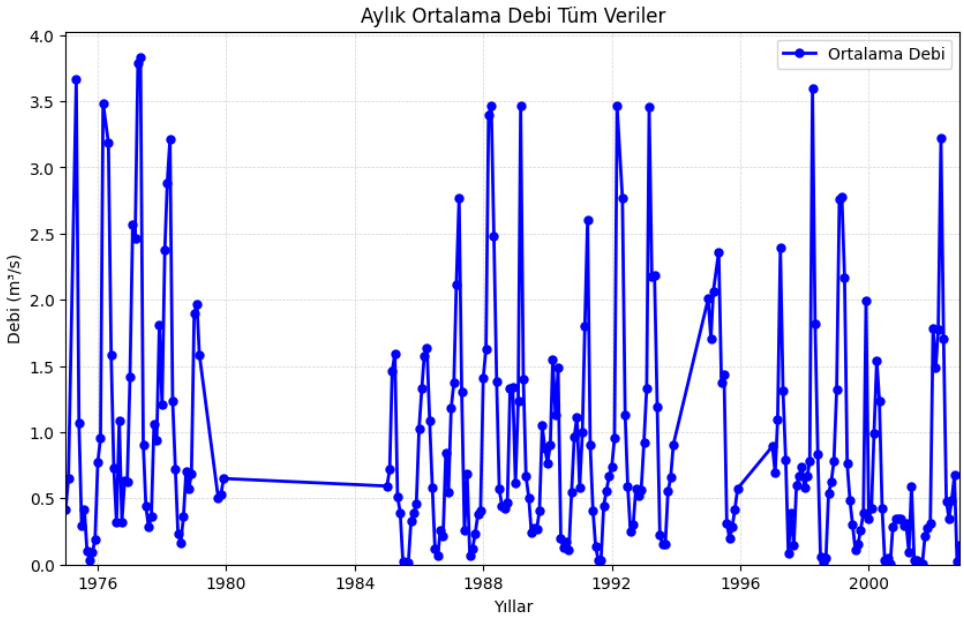
Bu çalışmada kullanılan verilerin doğasını göstermek amacıyla Şekil 5.2'de günlük ortalama debi verileri yıllara göre çizdirilmiştir. Grafikten görüleceği üzere, veriler belli seneler arasında (örn. 1980-1985, 1995-1996, 2005-2006) eksik kalmıştır. Ayrıca, verinin mevsimsel peryodik doğası grafikte açıkça görülmektedir.



Şekil 5.2: Günlük debi verilerinin zamana göre çizdirilmesi ile ortaya çıkan grafik. Görüldüğü gibi 1980 ve 1985 arası veriler eksiktir.

5.1.2 Aylık Ortalama Debi Verileri

Bu çalışmada kullanılan verilerin doğasını göstermek amacıyla Şekil 5.3'de aylık ortalama debi verileri yıllara göre çizdirilmiştir. Grafikten görüleceği üzere, veriler belli seneler arasında (örn. 1980-1985, 1995-1996, 2005-2006) eksik kalmıştır.



Şekil 5.3: Aylık ortalama debi verilerinin zamana göre çizdirilmesi ile ortaya çıkan örnek grafik. Görüldüğü gibi 1980 ve 1985 arası veriler eksiktir.

5.2 Debi Verilerinin İstatistiksel Özellikleri

Kullandığımız günlük debileri içeren veri kümesi için, veri kümesindeki veri adedi, veri kümesinin ortalaması, standart sapması, minimum ve maximum değerleri, ilk çeyrekliği, ortanca çeyrekliği ve üçüncü çeyreklik değerlerine ayrıca mod, medyan ve varyansına bakılmıştır.

5.2.1 Debi Değerlerinin Veri Önişleme Öncesi İstatistiksel Özellikleri

Veri içerisinde eksik verilerin olması veya veri değerlerinin numerik değerlere dönüştürülmemiş olması bu istatistiklerde doğruluk oranlarını değiştirebilir. Bu nedenle veri kümesinden doğru bir değerlendirme yapılabilmesi için öncelikle veri kümesinde önişlemler yapılmıştır. Eksik veriler çıkarılmış, tarihe göre sıralama yapılmış ve değerler numerik olarak ayarlanmıştır. Veri önişleme yöntemlerinden normalizasyon

uygulanmadan önce debi verilerinin istatistiksel özellikleri Tablo 5.1'de gösterilmektedir.

Günlük Debi Verilerinin İstatistiksel Özellikleri	
Metrikler	Değerler
Veri sayısı	12.600
Ortalama	1,05
Standart sapma	1,35
Min	0,00
Max	19,80
Q1	0,29
Q2	0,59
Q3	1,30
Medyan	0,59
Varyans	1,82

Tablo 5.1: Debi verilerinin istatistiksel özellikleri

5.2.2 Debi Değerlerinin Veri Önişleme Sonrası İstatistiksel Özellikleri

Max ve min değerleri arasındaki yüksek fark nedeniyle aykırı değerler (İng., outlier) veri içerisinden çıkarılmıştır. Eşik değerleri belirlenmiş ve bu değerin üstünde kalan değerler veri içerisinden çıkarılmıştır. Normalizasyon yöntemlerinden z-skor normalizasyon veri önişlemi işlemleri tamamlandıktan sonra debi verilerinin istatistiksel özellikleri Tablo 5.2'de gösterilmiştir.

Günlük Debi Verilerinin Veri Önişleme Sonrası İstatistik Özellikleri	
Metrikler	Değerler
Veri sayısı	12.250
Ortalama	0,90
Standart sapma	0,91
Min	0,00
Max	5,00
Q1	0,27
Q2	0,59
Q3	1,25
Medyan	0,59
Varyans	0,83

Tablo 5.2: Debi verilerinin veri önişleme sonrası istatistiksel özellikleri

5.3 Kullanılan Veri Önişleme Yöntemleri

Veri önişleme aşamasında, öncelikle veriler kullanılacak programa uygun bir formata dönüştürülmüş ve tarihe göre sıralanmıştır. Eksik verilerin gözlemi yapılmış, bu eksik veriler ardışık devam etmesinden dolayı tüm boş verilerin doldurulması yerine, çıkarılması uygun bulunmuştur. Eksik verilerin gözlemi yapıldıktan ve boş hücreler için çözüm sağlandıktan sonra elimizde bulunan debi verileri için veri istatistiği yapılmıştır. Daha sonra normalizasyon metotları içerisinden uygun olanı seçilmiş ve tez çalışmasında kullanılan aylık ve günlük debi verileri için ayrı ayrı kullanılmıştır.

```python
import pandas as pd

# Veri setini yükleyin
data = pd.read_csv('all_data_raw.csv')

# Boş değerleri sayın
bos_hucreler = data.isnull().sum().sum()

# Sonucu yazdırın
print("Veri setinde {} adet boş hücre var.".format(bos_hucreler))
```

```
Veri setinde 3364 adet boş hücre var.
```

Şekil 5.4: Eksik verilerin gözlemi. Verilerin yaklaşık %25'i eksiktir.

5.3.1 Veri Silme

Günlük veriler kullanılarak oluşturulan modeller üzerinde bu yöntem ile eksik verilerin tamamı silinmiştir. Eksik veriler sıralı olarak devam ettiği için bu yöntem kullanılmıştır. Fakat aylık veriler kullanılarak oluşturulan modellerdeki veri kümeleri üzerinde aynı işlem kullanılırken, eklenen özellikleri içeren veriler için başka bir veri önişleme yöntemi kullanılmıştır. Bunun sebebi ise debi verilerinde ardışık boşluklar varken, meteorolojik veri kümelerindeki boşluklar ardışık değildir. Kullanılan meteorolojik veriler ve bir önceki günün debi değeri, aylık ortalama maximum sıcaklık, aylık ortalama bağıl nem, aylık maksimum yağış, aylık toplam yağış, aylık toplam buharlaşma ve aylık ortalama güneşlenme süresi kullanılmıştır.

5.3.2 Normalizasyon Metodu

Bu çalışmada veri önişleme aşamasında kullanılan yöntemlerden biri olan normalizasyon yöntemi kullanılmıştır. Normalizasyon yöntemi ile veri kümesindeki sayısal değerlerin belirli bir aralığa veya belirli bir ölçeğe dönüştürülmesine, bu şekilde daha başarılı bir modelleme kurulmasına yardımcı olur. Ayrıca normalizasyon metodu kullanılan veri kümesi içerisinde varsa eklenen özellikler için de ayrı ayrı kullanılmalıdır

(Gökçe, Sönmez, Selen, & Aladağ, 2022). Normalizasyon metotlarının min-max normalizasyon, z-skor normalizasyon, log dönüşümü, Box-Cox dönüşümü ve skaler normalizasyon gibi çeşitleri bulumaktadır (Ali, Miah, Haque, Rahman, & Islam, 2021).
Bu tez çalışması içerisinde z-skor normalizasyon metodu hem günlük hem de aylık yapılan çalışmalar için kullanılmıştır. Ayrıca aylık ortalama debi verileri üzerinde yapılan çalışmalarda z-skor metoduna ek olarak özellikleri için min-max normalizasyon yöntemi kullanılmıştır.

5.3.2.1 Z-skor Normalizasyon: Bu yöntemde veri kümesi içerisinde bulunan veriler ortalama değerinden çıkarıldıktan sonra bulunan değer veri kümesinin standart sapması ile bölünerek elde edilen değerlere z-skorları denir. Her verinin standart sapma birimleri cinsinden ne kadar uzak olduğunu ifade eder ve formül içerisindeki x parametresi normalleştirilecek olan değeri, μ parametresi veri kümesinin ortalamasını, σ veri kümesinin standart sapmasını, z parametresi ise normalleştirilmiş değer (z-skoru) olarak ifade edilebilir. (Gökçe et al., 2022)
Bu tez çalışmasında veri önişleme aşamasında aylık ve günlük veriler kullanılarak yapılan çalışmalarda z-skor normalizasyon yöntemi kullanılmıştır.

$$z = \frac{x - \mu}{\sigma} \tag{5.1}$$

5.3.2.2 Min-max Normalizasyon: Verileri belirli bir aralığa sığdırmak için kullanılır. Veriler, minimum ve maksimum değerlerine göre belirli bir aralığa indirgenir. Genellikle bu aralık [0, 1] veya [-1, 1] aralığı olarak ölçeklenir. Burada, x verileri normalizasyon yapılacak veri kümesi değerleridir. Min(x) ve max(x) ise veri kümesinin minimum ve maksimum değerleridir ve y parametresi, normalleştirilmiş verilerin yeni değerlerini temsil eder (Gökçe et al., 2022). Bu normalizasyon yöntemi sadece aylık ortalama debi değerleri için yapılan tahminleme yöntemlerinde özellik olarak eklenen; bir önceki günün debi değeri, aylık ortalama maximum sıcaklık, aylık ortalama bağıl

nem, aylık maksimum yağış, aylık toplam yağış, aylık toplam buharlaşma ve aylık ortalama güneşlenme süresini gösteren veri kümelerine uygulanmştır.

$$y = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{5.2}$$

5.3.3 Geriye Doldurma (İng., Backward Filling)

Kullandığımız bu yöntem ile aylık ortalama debi değerlerinde kullanılan özellik veri kümeleri için eksik olan veriler, bir önceki gözlem değeriyle tamamlanmıştır. Bu yöntem özellikle zaman serileri analizinde yaygın olarak kullanılır (Şadi, 2013). Bu veri önişleme yöntemi sadece aylık ortalama debi değerleri için yapılan tahminleme yöntemlerinde özellik olarak eklenen; bir önceki günün debi değeri, aylık ortalama maximum sıcaklık, aylık ortalama bağıl nem, aylık maksimum yağış, aylık toplam yağış, aylık toplam buharlaşma ve aylık ortalama güneşlenme süresini gösteren veri kümelerine uygulanmştır.

5.4 Kullanılan Tahminleme Yöntemleri

5.4.1 Gradyan Arttırma Yöntemi (İng., Gradient Boosting Method)

Adaboost zayıf sınıflandırıcı algoritmalarını bir araya getirerek güçlü bir öğrenici modeli ortaya çıkaran bir algoritmadır (Freund & Schapire, 1996). 1997 yılında, Jerome H. Friedman, GB'in ilk halini önermiştir. GB algoritmasıyla öğrenme sürecinde, önceki tahminlerin hatalarını düzeltmek ve daha iyi bir tahmin yapmak için yenilikçi bir öğrenme algoritması kullandığını açıkladı (Friedman, 2001). 2001 yılında, Friedman, Hastie ve Tibshirani, Gradient Boosting Machine ismi ile yeni bir ağaç tabanlı öğrenme algoritması tanıttı. Bu algoritma, GB yöntemini kullanarak birden fazla karar ağacının birleştirilmesi yoluyla öğrenme gerçekleştirir (Hastie, Tibshirani, Friedman, & Friedman, 2009). Adaboost'un regresyon ve sınıflandırma problemlerine entegre edilebi-

lir fakat daha genelleştirilmiş versiyonu olarak bilinir (Friedman, 2001). Metodun temelinde zayıf öğrenicileri bir araya getirerek güçlü bir öğrenim ortaya çıkarmak yer alır (Ehrenfeucht, Haussler, Kearns, & Valiant, 1989). Çalışma prensibi ise hataları azaltmaya yönelik bir şekilde ardışık olarak öğreniciler eklemektir. Bu işlem, önceki öğrenicinin hatalarını düzeltmeye çalışan yeni bir öğrenici ekleyerek gerçekleştirilir. GB modeli, birçok karar ağacını bir araya getirerek bir tahmin yapar (Mason, Baxter, Bartlett, & Frean, 1999). Her karar ağacı, veri kümesinin belirli bir alt kümesi üzerinde çalışırken model bir başlangıç tahmini yapar, bu tahminin hatalarını hesaplar ve bir sonraki karar ağacı bu hataları düzeltmek için eğitilir. Bu işlem, bir sonraki karar ağacı, önceki ağaçların tahminlerini kullanarak veri kümesinin hatalarını azaltana kadar tekrarlanarak devam eder (Chen & Guestrin, 2016; Yildirim, 2021). GB modeli, her bir ağacın ağırlığına göre toplam tahminini hesaplar. Ağaçların ağırlığı, doğruluğuna göre belirlenir. Doğru tahmin eden ağaçlar daha yüksek ağırlıklı olacak ve tahminin daha büyük bir bölümüne katkıda bulunacaktır (Kaleli & Kurtuluş, 2021; M. Öztürk & Demirtas, 2019).

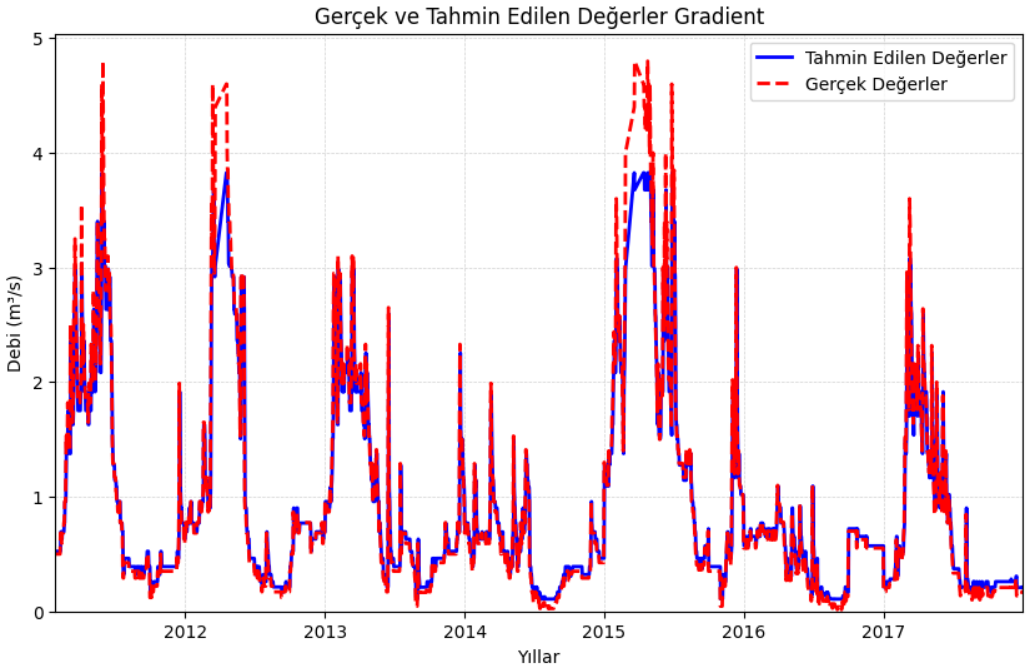
5.4.1.1 Günlük Debi Verileri Kullanılarak GB Metodu ile Tahminleme:

Günlük debi verileri kullanılarak uygulanan yöntem ile elde edilen hata değerleri Tablo 5.3'de gösterilmiştir.

Günlük Debi Verileri ile Gradient Boosting Metot	
Hata Metrikleri	Hata Değerleri
RMSE	0,21
MAE	0,10
R^2	0,94
MSE	0,04

Tablo 5.3: GB metodu ile yapılan tahminleme ve gerçek değerler arasındaki hata metrikleri

GB metodu tahminleme yöntemlerinde öncelikle günlük veriler kullanılarak elde edilen sonuçlara göre tahmin edilen ve gerçek değerler arasındaki karşılaştırma grafiği Şekil 5.5'de gösterilmektedir.



Şekil 5.5: GB metodu ile yapılan tahminleme sonuçlarının ve gerçek verilerin karşılaştırma grafiği. Bu tahminlemede günlük veriler kullanılmıştır.

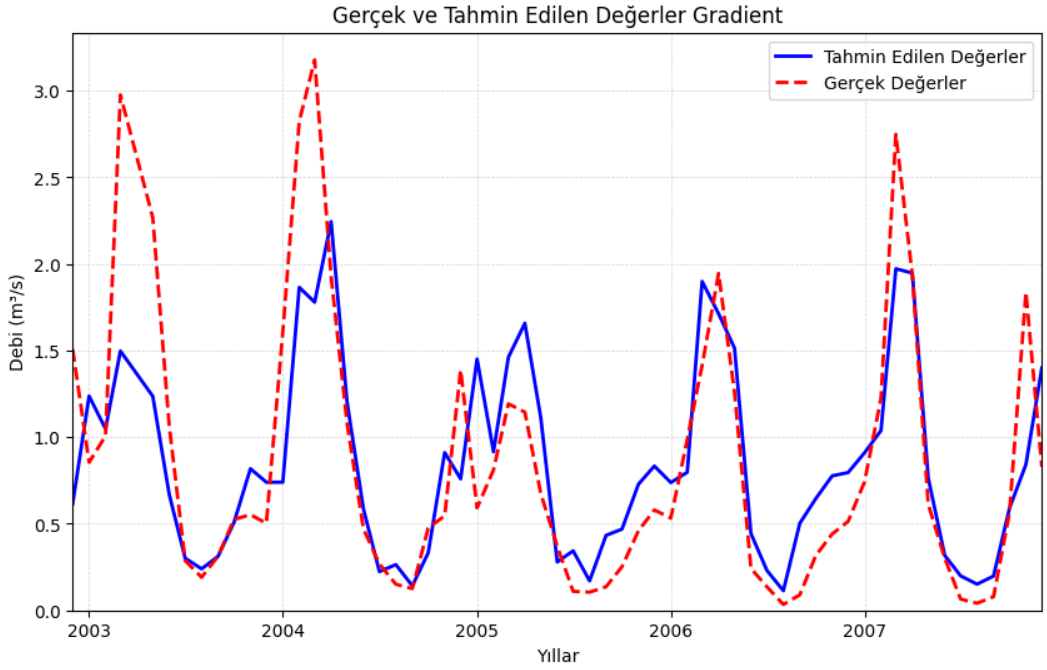
5.4.1.2 Aylık Ortalama Debi Verileri Kullanılarak Gradient Boosting Metot ile Tahminleme:

Aylık ortalama debi verileri kullanılarak uygulanan yöntem ile elde edilen hata değerleri Tablo 5.4'de gösterilmiştir.

Aylık Ortalama Debi Verileri Kullanılarak Gradient Boosting Metot	
Hata Metrikleri	Hata Değerleri
RMSE	0,47
MAE	0,33
R^2	0,64
MSE	0,22

Tablo 5.4: GB metodu ile yapılan tahminleme ve gerçek değerler arasındaki hata metrikleri

GB metodu tahminleme yöntemlerinde aylık ortalama veriler kullanılarak elde edilen sonuçlara göre tahmin edilen ve gerçek değerler arasındaki karşılaştırma grafiği Şekil 5.6'de gösterilmektedir.



Şekil 5.6: GB metodu ile yapılan tahminleme sonuçlarının ve gerçek verilerin karşılaştırma grafiği. Bu tahminlemede aylık ortalama veriler kullanılmıştır.

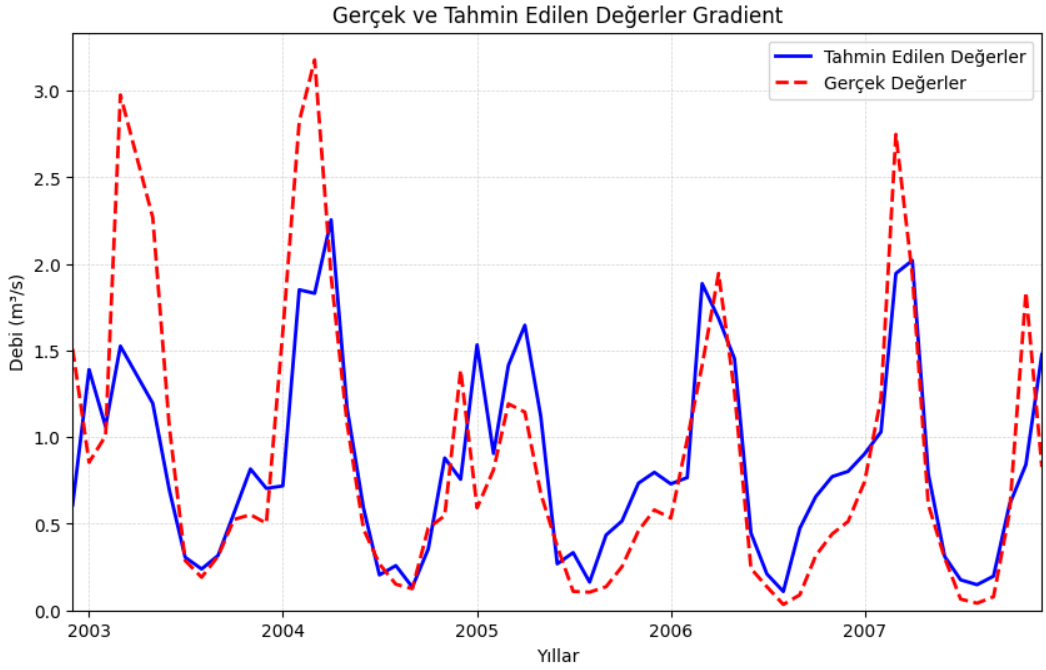
5.4.1.3 Aylık Ortalama Debi Verileri ve Özellik Veri Kümeleri Kullanılarak Gradient Boosting Metot ile Tahminleme:

Aylık ortalama debi verileri kullanılmış ve özellik veri kümeleri eklenerek uygulanan yöntem ile elde edilen hata değerleri Tablo 5.5'de gösterilmiştir.

Aylık Ortalama Debi Verileri ve Özellik Veri Kümeleri Kullanılarak Gradient Boosting Metot	
Hata Metrikleri	Hata Değerleri
RMSE	0,47
MAE	0,34
R^2	0,66
MSE	0,22

Tablo 5.5: GB metodu ile yapılan tahminleme ve gerçek değerler arasındaki hata metrikleri

GB metodu tahminleme yöntemlerinde aylık ortalama veriler kullanılmıştır ve özellik veri kümeleri eklenmiştir. Elde edilen sonuçlara göre tahmin edilen ve gerçek değerler arasındaki karşılaştırma grafiği Şekil 5.7'de gösterilmektedir.



Şekil 5.7: GB metodu ile yapılan tahminleme sonuçlarının ve gerçek verilerin karşılaştırma grafiği. Bu tahminlemede aylık ortalama veriler kullanılmıştır ve özellik veri kümeleri eklenmiştir.

5.4.2 Ekstrem Gradyan Arttırma Yöntemi (İng., Extreme Gradient Boosting Method)

XGBOOST modeli, GB metodununun tahmin ve hız performansını arttırmak için 2016 yılında Tianqi Chen tarafından geliştirilmiştir (İlya, Keser, & Yolaçan, 2021). Chen, XGBoost'u mevcut gradient boosting tekniklerinden daha yüksek performanslı ve ölçeklenebilir bir seçenek olarak tasarlamıştır (Unver & Yildiz, 2019). Açık kaynak kodlu bir proje olarak piyasaya sürülmesiyle beraber hızlı bir şekilde veri bilimciler arasında popülerlik kazanmıştır (Özsu & Koç, 2019). GBoost modeline göre ölçeklenebilir olmasından dolayı hızlı ve tahmin yetisi yüksek bir algoritmadır. Büyük veri kümeleriyle çalışırken bellek verimliliğini artırmak için özel bellek yönetimi teknikleri kullanır. Ağaç tabanlı bir algoritma olması sebebiyle XGBoost, karar ağaçları

kullanarak öğrenme gerçekleştirir, ayrıca ağaç yapısının kullanımı, modelin yüksek doğruluk seviyeleri elde etmesini sağlar. Regülarizasyon özelliği sayesinde XGBoost, L1, L2 regülarizasyon teknikleri ve dropout gibi teknikleri kullanarak aşırı öğrenmeyi (İng., overfitting) önlemek için çeşitli regülarizasyon tekniklerini içerir. Paralel işleme uygun olması ile XGBoost, CPU ve GPU'da paralel hesaplama yaparak işlem hızını artırır (Ke et al., 2017). Özelleştirilebilir olması sebebiyle XGBoost, kullanıcıların özelleştirilebilir objektif fonksiyonları, ölçütler ve optimizasyon stratejileri belirlemesine olanak tanır. Ayrıca, XGBoost, diğer makine öğrenimi modelleriyle birlikte kullanılabilir. Veri içerisindeki eksik değerlerin eğilimlerinin de tespit edilmesine olanak sağlar. Sonuç olarak ise minimum gereksinimler ile başarılı sonuçlar elde edebilir. Bu özellikler, XGBoost'un diğer makine öğrenimi modelleriyle karşılaştırıldığında yüksek performans ve ölçeklenebilirlik sağlar (Chen et al., 2018; Aydın, 2018; Aslı, 2021).

5.4.2.1 Günlük Debi Verileri Kullanılarak Extreme Gradient Boosting Metot ile Tahminleme:

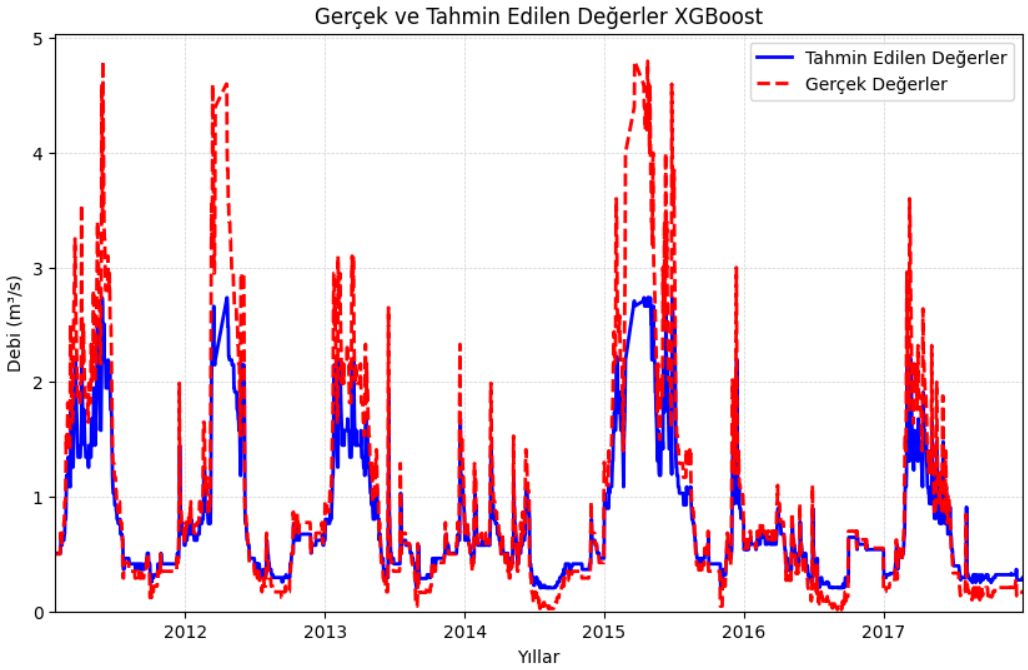
Günlük debi verileri kullanılarak uygulanan yöntem ile elde edilen hata değerleri Tablo 5.6'de gösterilmiştir.

Günlük Debi Verileri ile Extreme Gradient Boosting Metot	
Hata Metrikleri	Hata Değerleri
RMSE	0,40
MAE	0,22
R^2	0,80
MSE	0,16

Tablo 5.6: XGB metodu ile yapılan tahminleme ve gerçek değerler arasındaki hata metrikleri

XGB metodu tahminleme yöntemlerinde öncelikle günlük veriler kullanılarak elde edilen sonuçlara göre tahmin edilen ve gerçek değerler arasındaki karşılaştırma grafiği

Şekil 5.8'de gösterilmektedir.



Şekil 5.8: XGB metodu ile yapılan tahminleme sonuçlarının ve gerçek verilerin karşılaştırma grafiği. Bu tahminlemede günlük veriler kullanılmıştır

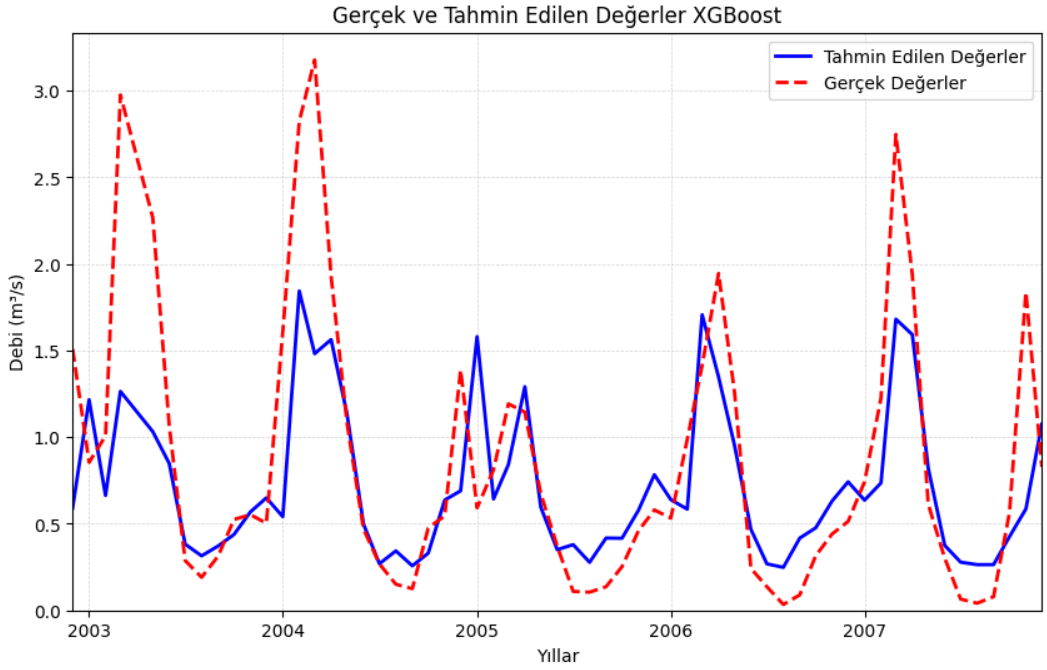
5.4.2.2 Aylık Ortalama Debi Verileri ile Extreme Gradient Boosting Metot ile Tahminleme:

Aylık ortalama debi verileri kullanılarak uygulanan yöntem ile elde edilen hata değerleri Tablo 5.7'de gösterilmiştir.

Aylık Ortalama Debi Verileri ile Extreme Gradient Boosting Metot	
Hata Metrikleri	Hata Değerleri
RMSE	0,53
MAE	0,36
R^2	0,54
MSE	0,29

Tablo 5.7: XGB metodu ile yapılan tahminleme ve gerçek değerler arasındaki hata metrikleri

XGB metodu tahminleme yöntemlerinde aylık ortalama veriler kullanılarak elde edilen sonuçlara göre tahmin edilen ve gerçek değerler arasındaki karşılaştırma grafiği Şekil 5.9'de gösterilmektedir.



Şekil 5.9: XGB metodu ile yapılan tahminleme sonuçlarının ve gerçek verilerin karşılaştırma grafiği. Bu tahminlemede aylık ortalama veriler kullanılmıştır.

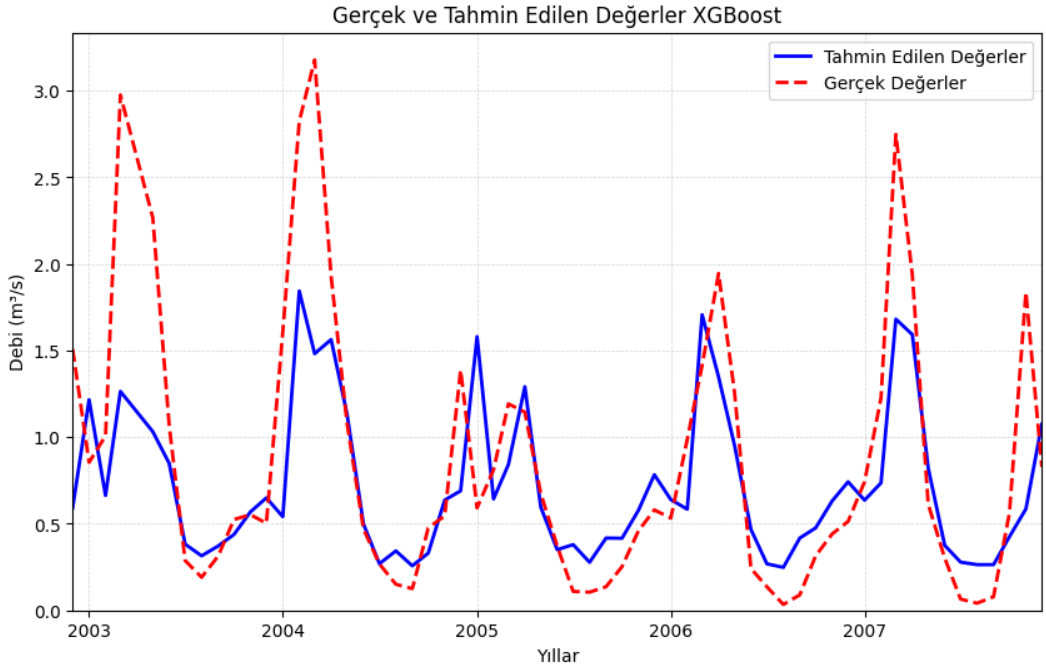
5.4.2.3 Aylık Ortalama Debi Verileri ve Özellik Veri Kümeleri Kullanılarak Extreme Gradient Boosting Metot ile Tahminleme:

Aylık ortalama debi verileri kullanılmış ve özellik veri kümeleri eklenerek uygulanan yöntem ile elde edilen hata değerleri Tablo 5.8'de gösterilmiştir.

Aylık Ortalama Debi Verileri ve Özellik Veri Kümeleri Kullanılarak XGB Tahminleme	
Hata Metrikleri	Hata Değerleri
RMSE	0,53
MAE	0,36
R^2	0,54
MSE	0,29

Tablo 5.8: XGB metodu ile yapılan tahminleme ve gerçek değerler arasındaki hata metrikleri

XGB metodu tahminleme yöntemlerinde aylık ortalama veriler kullanılmıştır ve özellik veri kümeleri eklenmiştir. Elde edilen sonuçlara göre tahmin edilen ve gerçek değerler arasındaki karşılaştırma grafiği Şekil 5.10'de gösterilmektedir.



Şekil 5.10: XGB metodu ile yapılan tahminleme sonuçlarının ve gerçek verilerin karşılaştırma grafiği. Bu tahminlemede aylık ortalama veriler kullanılmıştır ve özellik veri kümeleri eklenmiştir.

5.4.3 Kategorik Arttırma Yöntemi (İng., Categorical Boosting Method)

Boosting algoritmalarında olduğu gibi, zayıf öğrenicilerin (İng., weak learner) bir araya getirilerek güçlü bir öğrenici (İng., strong learner) elde edilmesi ve sonrasında ilk ağaç oluşturulduktan sonra hataların hesaplanması, ikinci ağaç oluşturulurken bu hataların kullanılması işlemini ve bu işlemin birçok ağacın bir araya getirilmesiyle devam ederek sonunda güçlü bir öğrenici elde edilmesi genel prensipleri ile çalışır (Freund & Schapire, 1996). XGBoost'un bir uzantısıdır ve özellikle kategorik değişkenlerin işlenmesi konusunda daha iyi performans gösterir (Prokhorenkova, Gusev, Vorobev, Dorogush, & Gulin, 2018; Kurt, 2021). Bu modelin tarihi XGBoost'un tarihine dayanır diyebiliriz. XGBoost, Tianqi Chen tarafından 2014 yılında geliştirildikten

sonra kategorik değişkenleri işlemek için XGBoost'un bir uzantısı olarak CatBoost, Yandex tarafından 2017 yılında geliştirilmiştir (Chen & Guestrin, 2016). Kategorik değişkenlerin işlenmesinde daha iyi performans gösterdiği ve kategorik değişken desteği sağlayan daha hızlı ve tahmin oranı yüksek bir modelleme olarak ortaya çıkmıştır (İlya et al., 2021). XGBoost modeline göre, kategorik değişkenleri işlemek için özel olarak tasarlanmış bir yöntem kullanır. Diğer boosting algoritmaları kategorik değişkenleri sayısal değerlere dönüştürerek işlerken, bu model kategorik değişkenleri sayısal değerlere dönüştürmeden, doğrudan işleyerek bilgi kaybını önler ve kategorik değişkenlerin sıralanması, ayrıklaştırılması gibi işlemler yapabilir (Ke et al., 2017). Bu işlemlerin yapılması, çok sayıda kategorik değişken bulunduran veri kümesi için modelin hızını ve doğruluğunu artırır. Ayrıca içerisinde bulunan kütüphane, kategorik değişkenlerin işlenmesinde yüksek performans gösterir ve verilerin büyüklüğüne, boyutuna bakılmaksızın çok çeşitli makine öğrenimi uygulamalarında kullanılabilmektedir (Li & Zou, 2018).

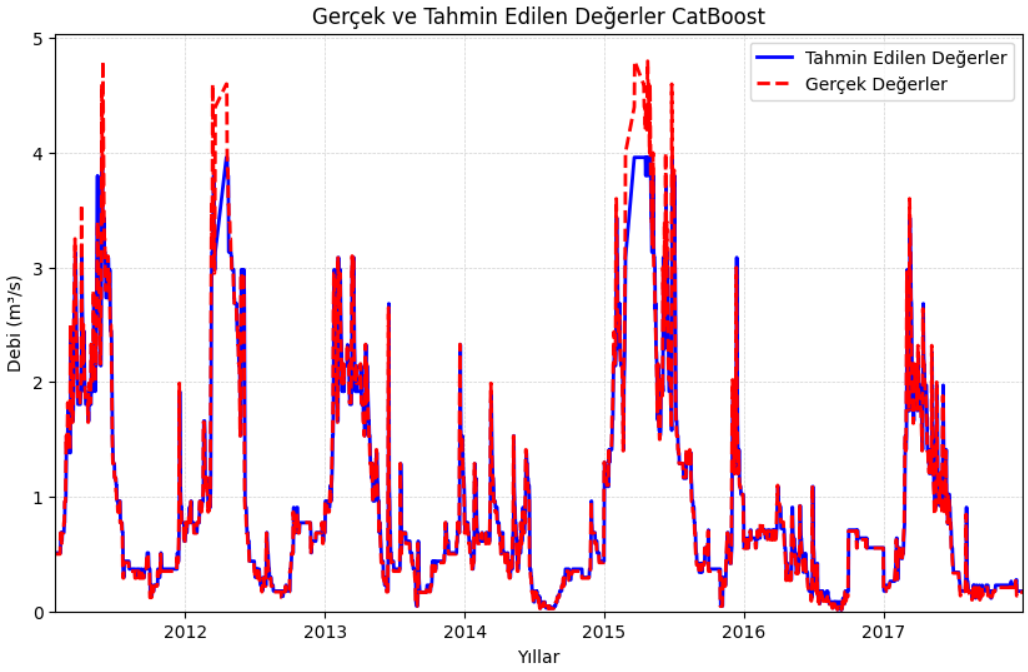
5.4.3.1 Günlük Debi Verileri ile Categorical Gradient Boosting Metot ile Tahminleme:

Günlük debi verileri kullanılarak uygulanan yöntem ile elde edilen hata değerleri Tablo 5.9'da gösterilmiştir.

Günlük Debi Verileri ile Categorical Gradient Boosting Metot	
Hata Metrikleri	Hata Değerleri
RMSE	0,20
MAE	0,08
R^2	0,95
MSE	0,04

Tablo 5.9: CATBOOST metodu ile yapılan tahminleme ve gerçek değerler arasındaki hata metrikleri

Catboost metodu tahminleme yöntemlerinde öncelikle günlük veriler kullanılarak elde edilen sonuçlara göre tahmin edilen ve gerçek değerler arasındaki karşılaştırma grafiği Şekil 5.11'de gösterilmektedir.



Şekil 5.11: Catboost metodu ile yapılan tahminleme sonuçlarının ve gerçek verilerin karşılaştırma grafiği. Bu tahminlemede günlük veriler kullanılmıştır.

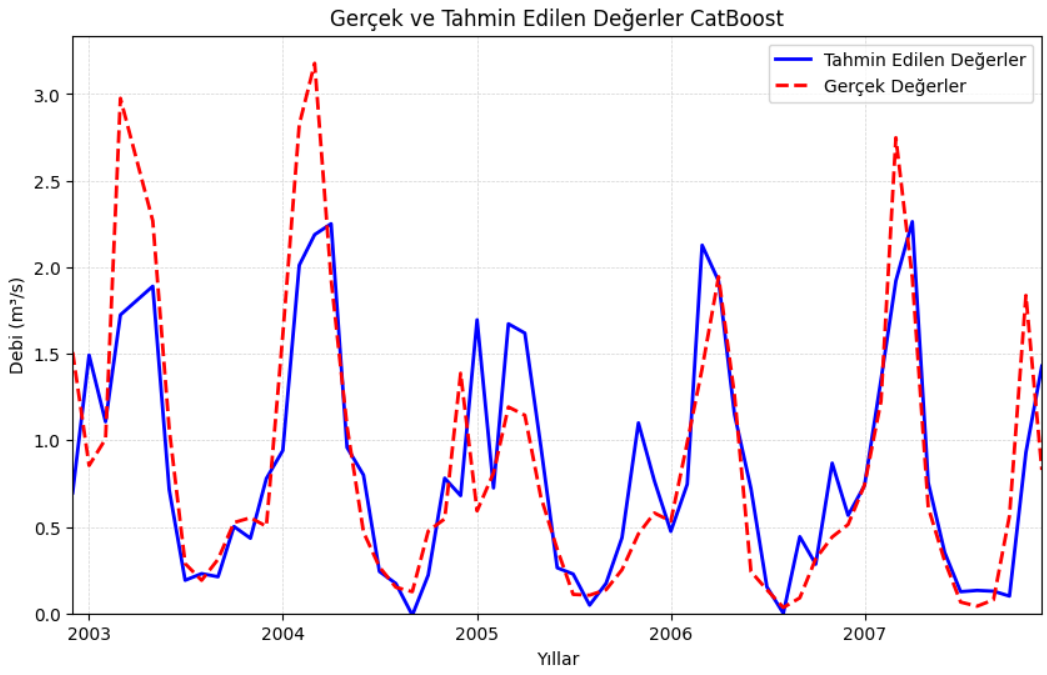
5.4.3.2 Aylık Ortalama Debi Verileri ile Categorical Gradient Boosting Metot ile Tahminleme:

Aylık ortalama debi verileri kullanılarak uygulanan yöntem ile elde edilen hata değerleri Tablo 5.10'da gösterilmiştir.

Aylık Ortalama Debi Verileri ile Categorical Gradient Boosting Metot	
Hata Metrikleri	Hata Değerleri
RMSE	0,44
MAE	0,31
R^2	0,69
MSE	0,19

Tablo 5.10: CATBOOST metodu ile yapılan tahminleme ve gerçek değerler arasındaki hata metrikleri

Catboost metodu tahminleme yöntemlerinde aylık ortalama veriler kullanılarak elde edilen sonuçlara göre tahmin edilen ve gerçek değerler arasındaki karşılaştırma grafiği Şekil 5.12'de gösterilmektedir.



Şekil 5.12: Catboost metodu ile yapılan tahminleme sonuçlarının ve gerçek verilerin karşılaştırma grafiği. Bu tahminlemede aylık ortalama veriler kullanılmıştır.

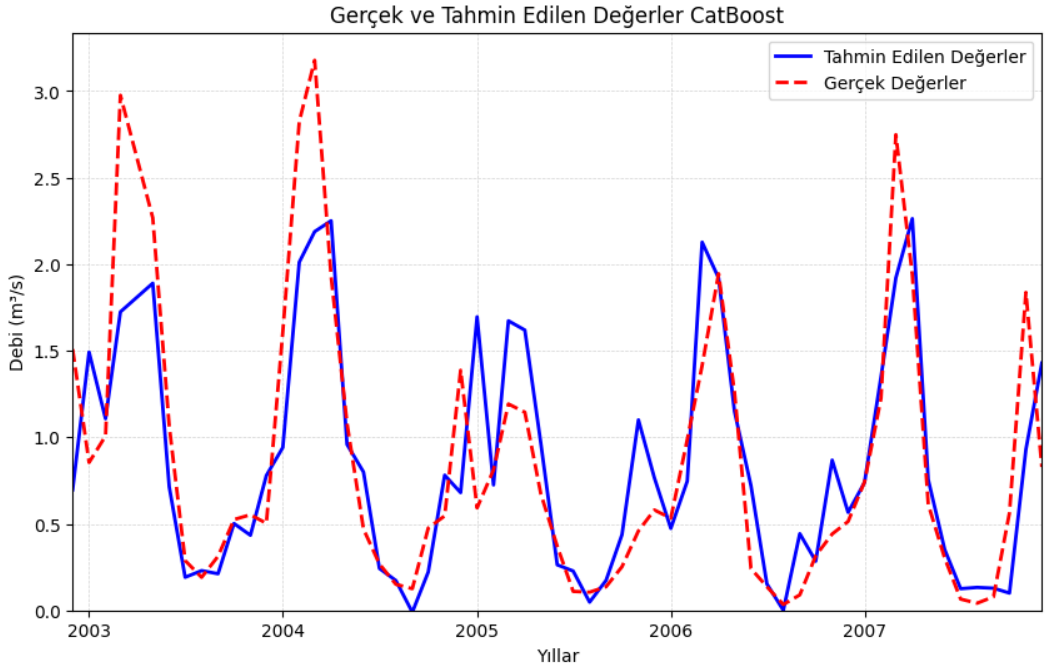
5.4.3.3 Aylık Ortalama Debi Verileri ve Özellik Veri Kümeleri Kullanılarak Categorical Gradient Boosting Metot ile Tahminleme:

Aylık ortalama debi verileri kullanılmış ve özellik veri kümeleri eklenerek uygulanan yöntem ile elde edilen hata değerleri Tablo 5.11'de gösterilmiştir.

Aylık Ortalama Debi Verileri ve Özellik Veri Kümeleri Kullanılarak Catboost Metot	
Hata Metrikleri	Hata Değerleri
RMSE	0,44
MAE	0,31
R^2	0,69
MSE	0,19

Tablo 5.11: CATBOOST metodu ile yapılan tahminleme ve gerçek değerler arasındaki hata metrikleri

Catboost metodu tahminleme yöntemlerinde aylık ortalama veriler kullanılmıştır ve özellik veri kümeleri eklenmiştir. Elde edilen sonuçlara göre tahmin edilen ve gerçek değerler arasındaki karşılaştırma grafiği Şekil 5.13'de gösterilmektedir.



Şekil 5.13: Catboost metodu ile yapılan tahminleme sonuçlarının ve gerçek verilerin karşılaştırma grafiği. Bu tahminlemede aylık ortalama veriler kullanılmıştır ve özellik veri kümeleri eklenmiştir.

5.4.4 Hafif Gradyan Arttırma Yöntemi (İng., Light Gradient Boosting Method)

Gradient boosting decision trees algoritmaları, birçok zayıf öğreniciden (İng., weak learner) oluşan güçlü bir öğrenici (İng., strong learner) oluşturmak için boosting tekniğini kullanırlar. Bu algoritmaların yavaş çalışması ve yüksek bellek kullanımı sebebiyle büyük ölçekli veri kümeleri ile çalışırken önemli bir sorun haline gelir ve bu sorunları çözmek için özellikle düşük bellek kullanımı ve yüksek hızlı hesaplama için Microsoft tarafından geliştirilen açık kaynaklı bir boosting yapısı olarak tasarlanmıştır (GuolinKe et al., 2017). Bu amaçla LightGBM, özellikle büyük veri kümeleri üzerinde çalışmak için tasarlanmış özellikler ve optimizasyonlar içerir. Bu optimizasyonlar arasında; düşük bellek kullanımı için veri sıkıştırma teknikleri, büyük veri kümeleri için dağıtılmış he-

saplama, kesin olmayan yöntemlerle hızlı hesaplama ve daha hızlı eğitim için daha fazla CPU kullanımı gibi teknikler yer almaktadır (Oral, Okatan, & Kırbaş, 2021). Yani XGboost modelinin tahmin süresinin kısaltılması üzerine geliştirilmiş bir algoritmadır (Ke et al., 2017). Kısacası bu model histogram tabanlı olup büyük boyutlar ile çalışmaya olanak sağlar (İlya et al., 2021).

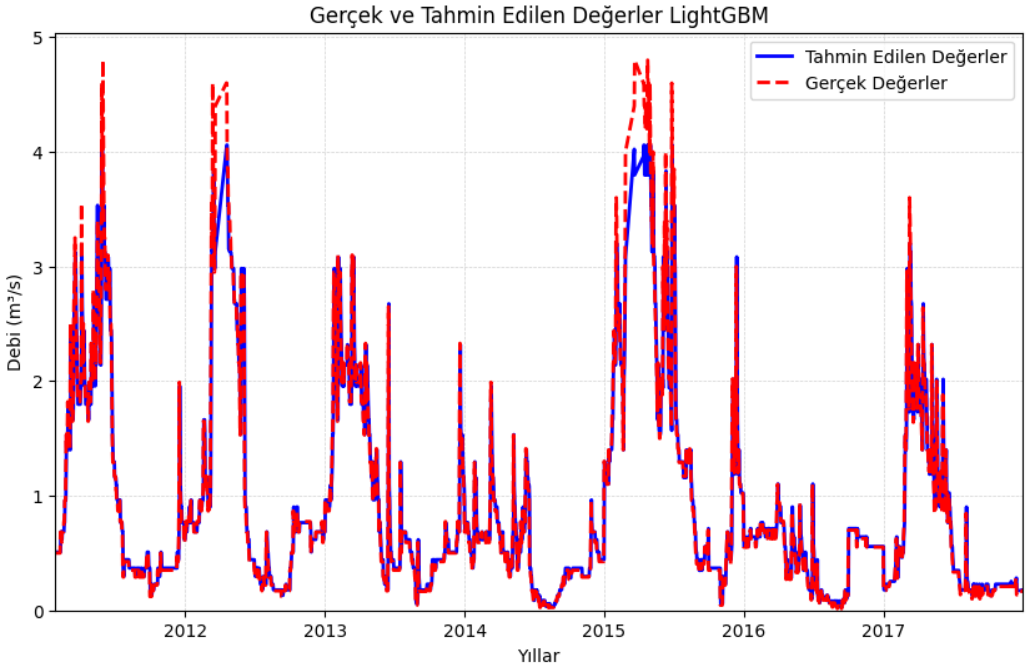
5.4.4.1 Günlük Debi Verileri ile Light Gradient Boosting Metot ile Tahminleme:

Günlük debi verileri kullanılarak uygulanan yöntem ile elde edilen hata değerleri Tablo 5.12'de gösterilmiştir.

Günlük Debi Verileri ile Light Gradient Boosting Metot	
Hata Metrikleri	Hata Değerleri
RMSE	0,20
MAE	0,08
R^2	0,95
MSE	0,04

Tablo 5.12: Light GB metodu ile yapılan tahminleme ve gerçek değerler arasındaki hata metrikleri

Light GB metodu tahminleme yöntemlerinde öncelikle günlük veriler kullanılarak elde edilen sonuçlara göre tahmin edilen ve gerçek değerler arasındaki karşılaştırma grafiği Şekil 5.14'de gösterilmektedir.



Şekil 5.14: Light GB metodu ile yapılan tahminleme sonuçlarının ve gerçek verilerin karşılaştırma grafiği. Bu tahminlemede günlük veriler kullanılmıştır.

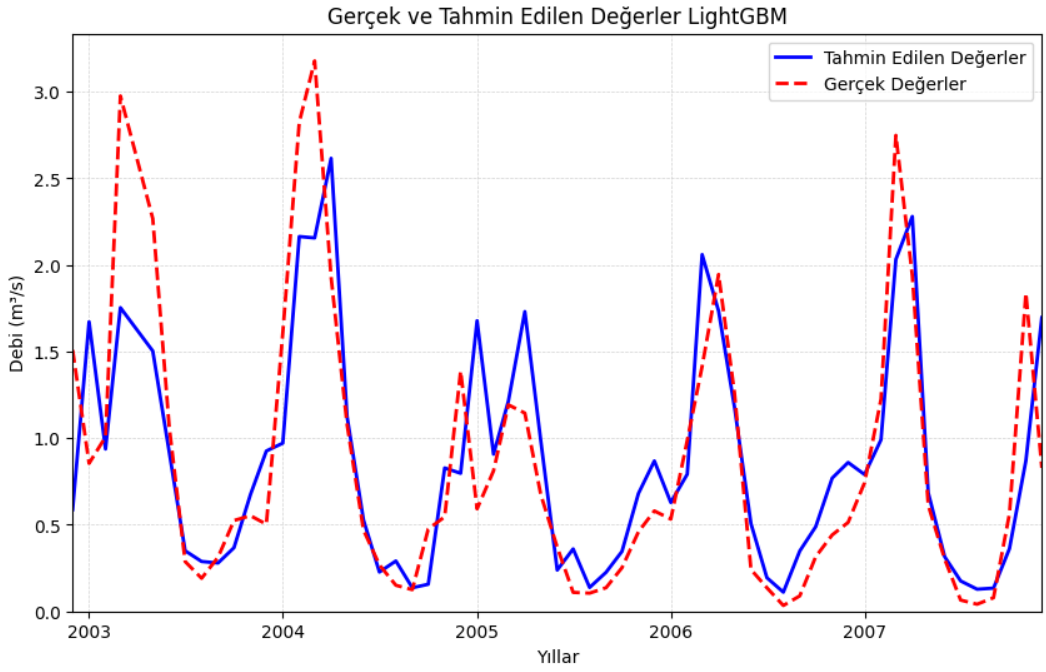
5.4.4.2 Aylık Ortalama Debi Verileri ile Light Gradient Boosting Metot ile Tahminleme:

Aylık ortalama debi verileri kullanılmış ve özellik veri kümeleri eklenerek uygulanan yöntem ile elde edilen hata değerleri Tablo 5.13'de gösterilmiştir.

Aylık Ortalama Debi Verileri ile Light Grediant Boosting Metot	
Hata Metrikleri	Hata Değerleri
RMSE	0,45
MAE	0,32
R^2	0,68
MSE	0,20

Tablo 5.13: Light GB metodu ile yapılan tahminleme ve gerçek değerler arasındaki hata metrikleri

Light GB metodu tahminleme yöntemlerinde aylık ortalama veriler kullanılarak elde edilen sonuçlara göre tahmin edilen ve gerçek değerler arasındaki karşılaştırma grafiği Şekil 5.15'de gösterilmektedir.



Şekil 5.15: Light GB metodu ile yapılan tahminleme sonuçlarının ve gerçek verilerin karşılaştırma grafiği. Bu tahminlemede aylık ortalama veriler kullanılmıştır

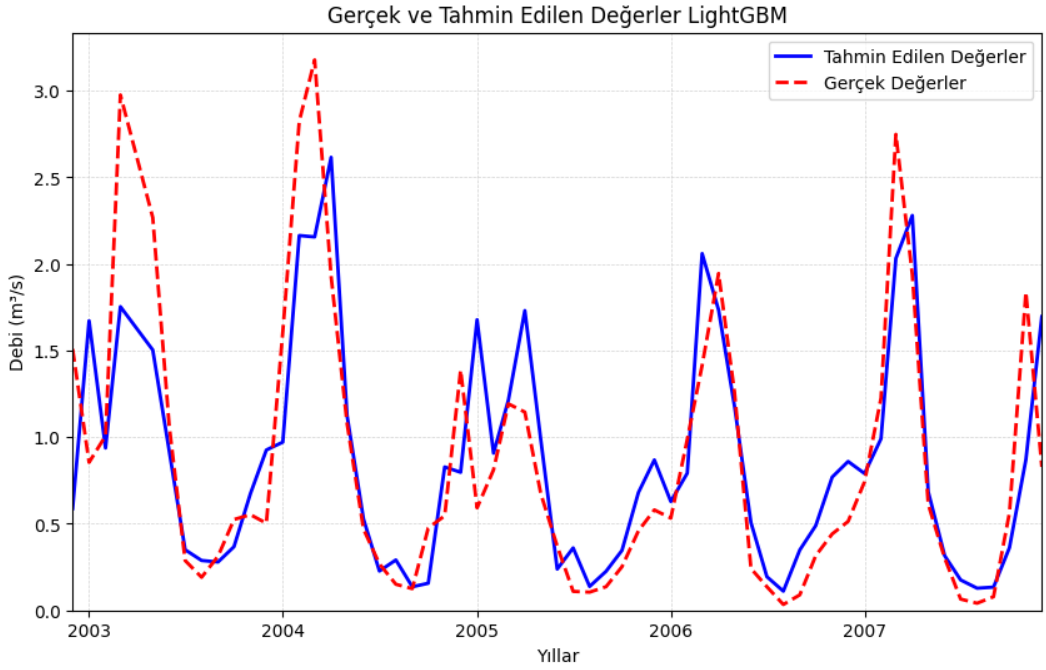
5.4.4.3 Aylık Ortalama Debi Verileri ve Özellik Veri Kümeleri Kullanılarak Light Gradient Boosting Metot:

Aylık ortalama debi verileri kullanılmış ve özellik veri kümeleri eklenerek uygulanan yöntem ile elde edilen hata değerleri Tablo 5.14'de gösterilmiştir.

Aylık Ortalama Debi Verileri ve Özellik Veri Kümeleri Kullanılarak Light GB Metot	
Hata Metrikleri	Hata Değerleri
RMSE	0,45
MAE	0,32
R^2	0,68
MSE	0,20

Tablo 5.14: Light GB metodu ile yapılan tahminleme ve gerçek değerler arasındaki hata metrikleri

Light GB metodu tahminleme yöntemlerinde aylık ortalama veriler kullanılmıştır ve özellik veri kümeleri eklenmiştir. Elde edilen sonuçlara göre tahmin edilen ve gerçek değerler arasındaki karşılaştırma grafiği Şekil 5.16'da gösterilmektedir.



Şekil 5.16: Light GB metodu ile yapılan tahminleme sonuçlarının ve gerçek verilerin karşılaştırma grafiği. Bu tahminlemede aylık ortalama veriler kullanılmıştır ve özellik veri kümeleri eklenmiştir.

5.4.5 K-En Yakın Komşuluk Regresyon Yöntemi (İng., K-Nearest Neighbors Method)

İlk kez 1967 yılında Peter N. Yianilos tarafından tanıtılmıştır. Ancak KNN'nin popülerliği, bu algoritmaya dayanan öğrenme yöntemlerinin geliştirilmesiyle 1990'lı yıllarda artmaya başlamıştır (Altman, 1992). KNN yöntemi, özellikle sınıflandırma problemlerinde kullanılmak üzere tasarlanmıştır. Ayrıca bu yöntem regresyon problemlerinde de kullanılabilir (Köklü, Karaca, & Güneş, 2016). K-NN regresyonu, KNN sınıflandırmasına benzer bir şekilde çalışır, fakat çoğunluk oylaması yerine komşu örneklerin hedef değerlerinin ortalaması alınır (Bouckaert, 1994; Coşar & Deniz, 2021; Sağbaş & Ballı, 2016). Veri kümesinin içerisideki elemanlar, oluşturulan k farklı kümeden birisine dahil edilmeye çalışılır. Bu yöntem öncelikle uzayda k noktada rasgele ola-

rak küme merkezleri oluşturarak herbir veri için uzaklıklarına bakılarak en yakın olan kümeye dahil edilir. Daha sonra kümeler için yeni merkez noktaları belirlenerek işlem tekrarlanır (A. Öztürk, Durak, & Badıllı, 2020).

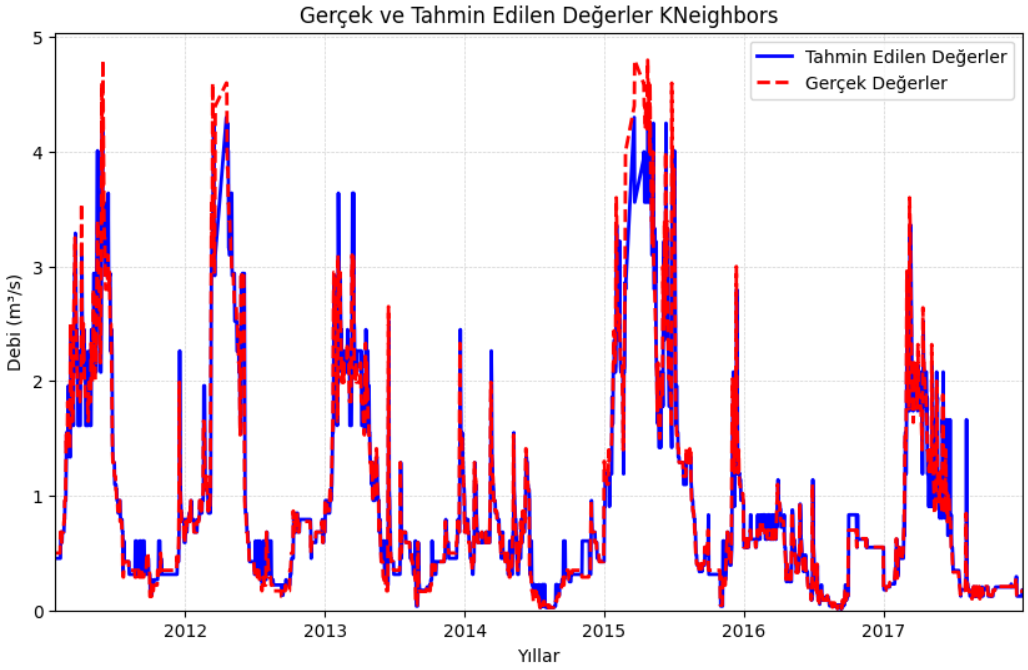
5.4.5.1 Günlük Debi Verileri Kullanılarak KNN Metodu ile Tahminleme:

Günlük debi verileri kullanılarak uygulanan yöntem ile elde edilen hata değerleri Tablo 5.15'de gösterilmiştir.

Günlük Debi Verileri ile KNN Metot	
Hata Metrikleri	Hata Değerleri
RMSE	0,24
MAE	0,12
R^2	0,93
MSE	0,06

Tablo 5.15: KNN metodu ile yapılan tahminleme ve gerçek değerler arasındaki hata metrikleri

KNN metodu tahminleme yöntemlerinde öncelikle günlük veriler kullanılarak elde edilen sonuçlara göre tahmin edilen ve gerçek değerler arasındaki karşılaştırma grafiği Şekil 5.17'de gösterilmektedir.



Şekil 5.17: KNN metodu ile yapılan tahminleme sonuçlarının ve gerçek verilerin karşılaştırma grafiği. Bu tahminlemede günlük veriler kullanılmıştır.

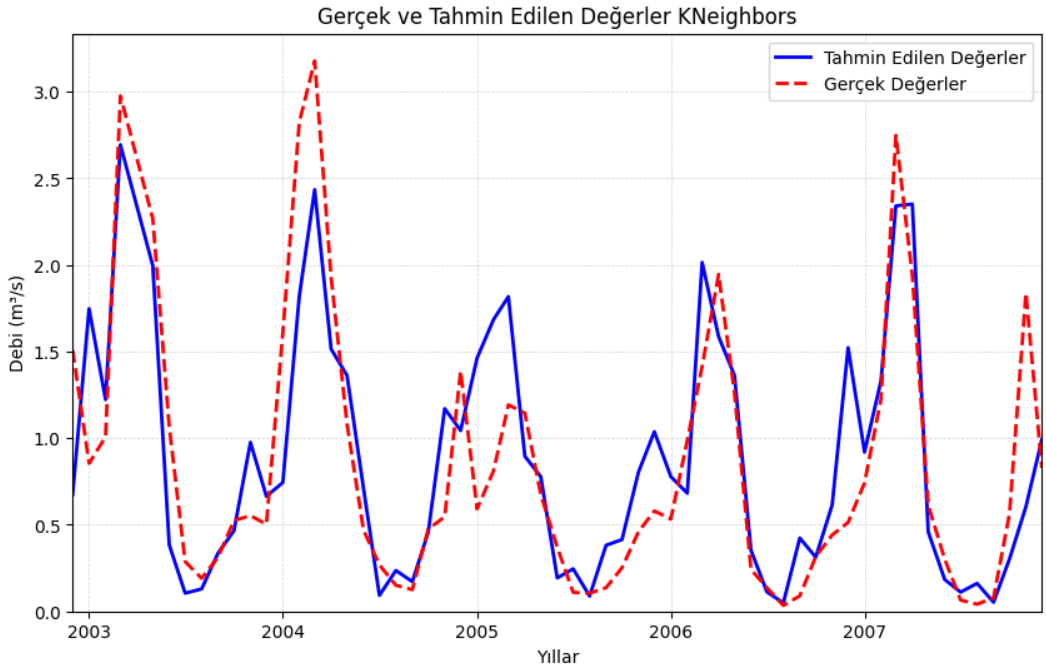
5.4.5.2 Aylık Ortalama Debi Verileri KNN Metodu ile Tahminleme:

Aylık ortalama debi verileri kullanılarak uygulanan yöntem ile elde edilen hata değerleri Tablo 5.16'da gösterilmiştir.

Aylık Ortalama Debi Verileri ile KNN Metot	
Hata Metrikleri	Hata Değerleri
RMSE	0,45
MAE	0,33
R^2	0,68
MSE	0,20

Tablo 5.16: KNN metodu ile yapılan tahminleme ve gerçek değerler arasındaki hata metrikleri

KNN metodu tahminleme yöntemlerinde aylık ortalama veriler kullanılarak elde edilen sonuçlara göre tahmin edilen ve gerçek değerler arasındaki karşılaştırma grafiği Şekil 5.18'de gösterilmektedir.



Şekil 5.18: KNN metodu ile yapılan tahminleme sonuçlarının ve gerçek verilerin karşılaştırma grafiği. Bu tahminlemede aylık ortalama veriler kullanılmıştır.

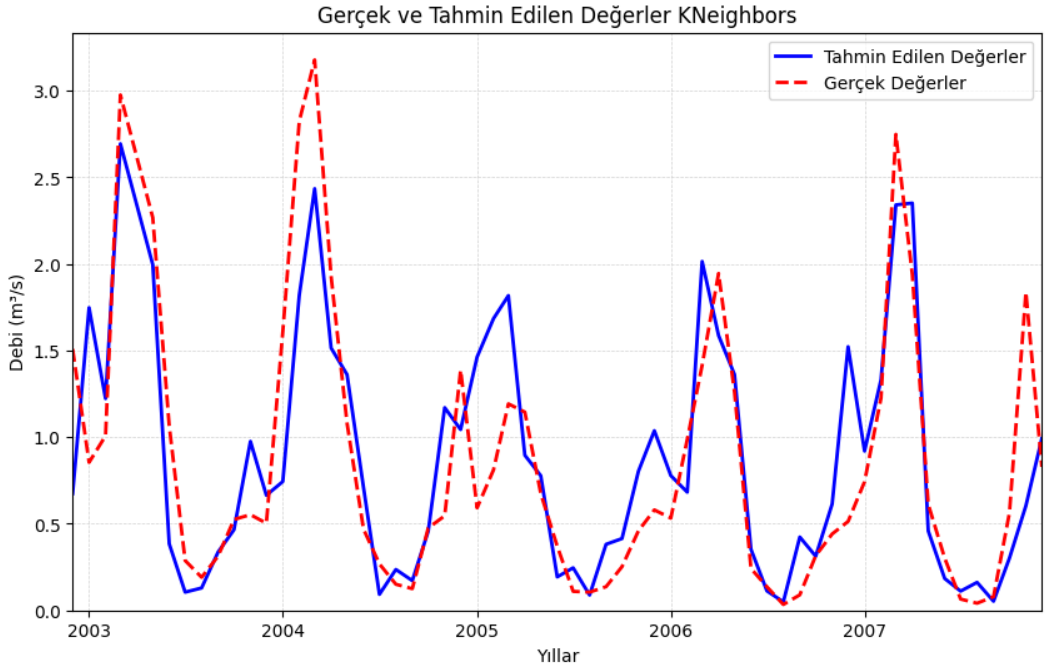
5.4.5.3 Aylık Ortalama Debi Verileri ve Özellik Veri Kümeleri Kullanılarak KNN Metodu ile Tahminleme:

Aylık ortalama debi verileri kullanılmış ve özellik veri kümeleri eklenerek uygulanan yöntem ile elde edilen hata değerleri Tablo 5.17'de gösterilmiştir.

Aylık Ortalama Debi Verileri ve Özellik Veri Kümeleri Kullanılarak KNN Metodu	
Hata Metrikleri	Hata Değerleri
RMSE	0,45
MAE	0,33
R^2	0,68
MSE	0,20

Tablo 5.17: KNN metodu ile yapılan tahminleme ve gerçek değerler arasındaki hata metrikleri

KNN metodu tahminleme yöntemlerinde aylık ortalama veriler kullanılmıştır ve özellik veri kümeleri eklenmiştir. Elde edilen sonuçlara göre tahmin edilen ve gerçek değerler arasındaki karşılaştırma grafiği Şekil 5.19'da gösterilmektedir.



Şekil 5.19: KNN metodu ile yapılan tahminleme sonuçlarının ve gerçek verilerin karşılaştırma grafiği. Bu tahminlemede aylık ortalama veriler kullanılmıştır ve özellik veri kümeleri eklenmiştir.

5.4.6 Sınıflandırma ve Karar Ağaçları Yöntemi (İng., Classification and Regression Trees Method)

Sınıflandırma ve karar ağaçları yöntemi (CART), 1980'lerin başında Breiman, Friedman, Olshen ve Stone tarafından geliştirilmiş bir makine öğrenimi yöntemidir. CART, ağaç tabanlı bir modelleme yöntemidir ve verilerin ağaç yapısı ile görselleştirilmesine izin verir. Sınıflandırma problemlerinde, yaprak düğümleri sınıf etiketlerini içerirken regresyon problemlerinde yaprak düğümleri sayısal tahminler sağlar. CART yöntemi, ağaç oluşturmak için özellik seçimi ve dal bölme kuralları gibi birçok farklı özellikleri kullanır. CART metodu sadece ikili ağaç yapımına imkan verir. Sürekli ve kategorik değişkenleri kullanabilen sınıflandırma ve regresyon ağacı algoritmasıdır (Breiman, Friedman, Olshen, & Stone, 1984; Kayri & Boysan, 2008). Bu yöntem, karar ağacı kullanarak regresyon problemlerini çözmek için kullanılır. İlk olarak veri kümesi bölünür ve her bölünmüş kümedeki varyans azaltılma yoluna gidilir. Bu işlem, her düğümde bir hedef değişkenin varyansının minimize edilmesini içerir. Veri kümesi bölündükten sonra her bölünmüş küme için bir alt ağaç oluşturulur ve bu alt ağaçlar da aynı yöntemle bölünmeye devam eder. Ağaç büyüdükçe düğümler daha da küçük alt kümeler halinde bölünür ve bu alt kümelerin ortalama hedef değişken değerleri yaprak düğümlerinde tahmin edilir. Yani her yaprak düğümü bir tahmin değeri sağlar (Loh, 2011). Bu durum, regresyon problemleri için CART yönteminin temel çalışma prensibidir.

CART yöntemi, ağaç oluşturma sırasında veri kümesinin her boyutuna odaklanır ve her boyutun en iyi ayırma noktasını belirler. Bölme kurallarının belirlenmesinde kullanılır. CART yöntemi, bölme kurallarını ve ağaç yapısını belirlemek için ayrıca düğüm puanlama ve budama tekniklerini de kullanır. CART regresyon yöntemi, veri kümesini ağaç yapısı ile temsil eder ve veri kümesindeki değişkenlerin optimum bölümlere ayrılması için en uygun dal bölme kurallarını kullanarak regresyon analizi yapar. Kısacası yapı olarak 3 adımdan oluşmaktadır. Maksimum ağacın oluşturulması ile başlar, ağaç bu-

dama ve optimum ağacın seçimi olarak tamamlanır (A. Yavuz & Çilengiroğlu, 2020; De'ath & Fabricius, 2000).

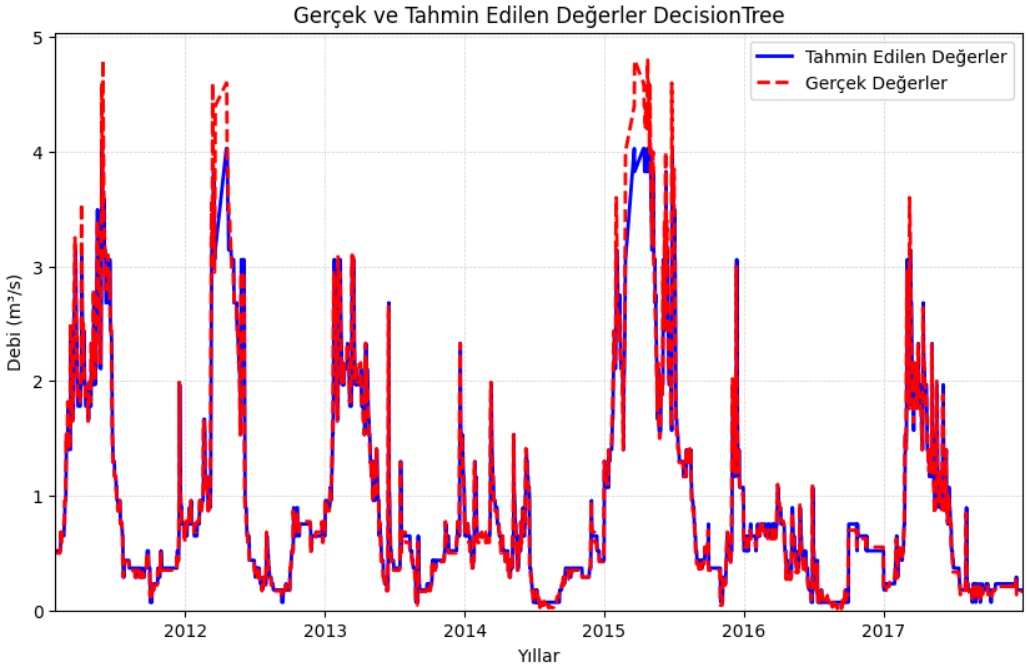
5.4.6.1 Günlük Debi Verileri Kullanılarak CART Metodu ile Tahminleme:

Günlük debi verileri kullanılarak uygulanan yöntem ile elde edilen hata değerleri Tablo 5.18'de gösterilmiştir.

Günlük Debi Verileri ile CART Metodu	
Hata Metrikleri	Hata Değerleri
RMSE	0,20
MAE	0,10
R^2	0,95
MSE	0,04

Tablo 5.18: CART metodu ile yapılan tahminleme ve gerçek değerler arasındaki hata metrikleri

CART metodu tahminleme yöntemlerinde öncelikle günlük veriler kullanılarak elde edilen sonuçlara göre tahmin edilen ve gerçek değerler arasındaki karşılaştırma grafiği Şekil 5.20'de gösterilmektedir.



Şekil 5.20: CART metodu ile yapılan tahminleme sonuçlarının ve gerçek verilerin karşılaştırma grafiği. Bu tahminlemede günlük veriler kullanılmıştır.

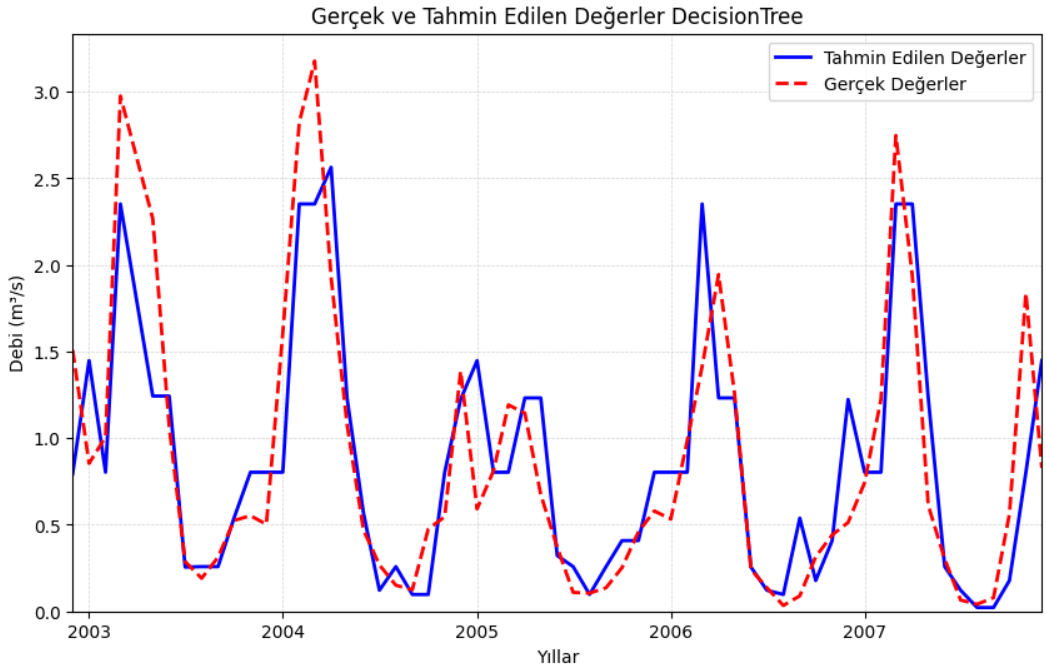
5.4.6.2 Aylık Ortalama Debi Verileri Kullanılarak CART Metodu ile Tahminleme:

Aylık ortalama debi verileri kullanılarak uygulanan yöntem ile elde edilen hata değerleri Tablo 5.19'da gösterilmiştir.

Aylık Ortalama Debi Verileri ile CART Metot	
Hata Metrikleri	Hata Değerleri
RMSE	0,42
MAE	0,31
R^2	0,71
MSE	0,18

Tablo 5.19: CART metodu ile yapılan tahminleme ve gerçek değerler arasındaki hata metrikleri

CART metodu tahminleme yöntemlerinde aylık ortalama veriler kullanılarak elde edilen sonuçlara göre tahmin edilen ve gerçek değerler arasındaki karşılaştırma grafiği Şekil 5.21'de gösterilmektedir.



Şekil 5.21: CART metodu ile yapılan tahminleme sonuçlarının ve gerçek verilerin karşılaştırma grafiği. Bu tahminlemede aylık ortalama veriler kullanılmıştır.

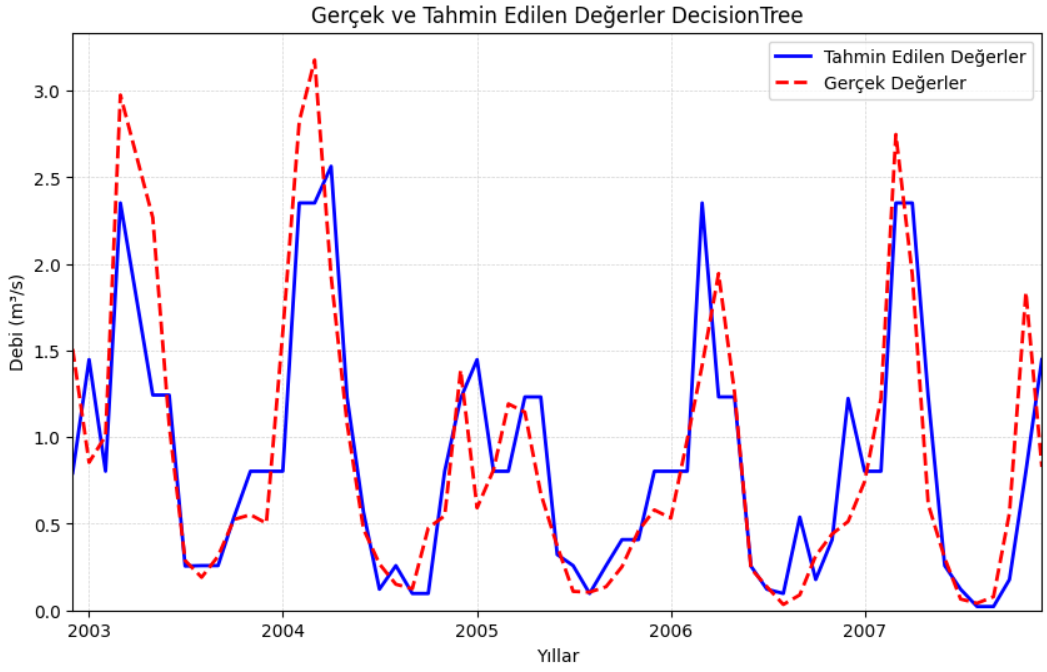
5.4.6.3 Aylık Ortalama Debi Verileri ve Özellik Veri Kümeleri Kullanılarak CART Metodu ile Tahminleme:

Aylık ortalama debi verileri kullanılmış ve özellik veri kümeleri eklenerek uygulanan yöntem ile elde edilen hata değerleri Tablo 5.20'de gösterilmiştir.

Aylık Ortalama Debi Verileri ve Özellik Veri Kümeleri ile CART Metot	
Hata Metrikleri	Hata Değerleri
RMSE	0,42
MAE	0,31
R^2	0,71
MSE	0,18

Tablo 5.20: CART metodu ile yapılan tahminleme ve gerçek değerler arasındaki hata metrikleri

CART metodu tahminleme yöntemlerinde aylık ortalama veriler kullanılmıştır ve özellik veri kümeleri eklenmiştir. Elde edilen sonuçlara göre tahmin edilen ve gerçek değerler arasındaki karşılaştırma grafiği Şekil 5.22'de gösterilmektedir.



Şekil 5.22: CART metodu ile yapılan tahminleme sonuçlarının ve gerçek verilerin karşılaştırma grafiği. Bu tahminlemede aylık ortalama veriler kullanılmıştır ve özellik veri kümeleri eklenmiştir.

5.4.7 Uzun Kısa Süreli Hafıza Yöntemi (İng., Long Short Term Method)

Uzun kısa süreli hafıza (LSTM), yinelemeli sinir ağı (RNN) türü bir derin öğrenme modelidir, zaman serisi verilerinin analizi ve tahmininde kullanılmaktadır (Graves, Fern'andez, & Gomez, 2005). LSTM, 1997 yılında Hochreiter ve Schmidhuber tarafından geliştirilmiştir. Öncesinde RNN, zaman serileri gibi sıralı verilerin işlenmesi için kullanılıyordu ancak bu modellerde zamanla ortaya çıkan problemler sebebiyle kullanımı zorlaşmıştır. RNN'de ortaya çıkan sorunlar LSTM yönteminin geliştirilmesi ile beraber ortadan kaldırılmıştır (Hochreiter & Schmidhuber, 1997). RNN modellerinde her bir giriş, geçmiş girişlerin ve mevcut durumun bir fonksiyonu olarak hesaplanırken bu durumdaki yapıyı genişletme yolu ile bellek hücreleri ekleyerek yeni bir LSTM modeli oluşturulur (Sharma, Kumar, & Patni, 2017). Girdi kapısı, çıkış kapısı ve unutma kapısı olmak üzere 3 temelden oluşan LSTM modelindeki bu kapılar, bellek hücrelerinin hangi bilgileri saklayacağı veya atacağı konusunda karar mekanizmasına sahiptir (Gündüz & Akkaya, 2019).

Girdi kapısı olarak belirtilen kısım mevcut giriş verisine dayanarak hangi bilgilerin bellek hücresine ekleneceğini belirleyen bir yapıdan oluşmaktadır. Çıkış kapısı ise bellek hücresinden hangi bilgilerin çıkarılacağını belirlemekle görevlidir. Unutma kapısı olarak belirtilen kısım ise, hangi bilgilerin bellek hücresinden silineceğini belirler. Bahsedilen bu kapılar ağırlıkların öğrenilmesi yolu ile belirlenmektedir (Xie, Xu, & Wang, 2018; Sarı & Tunalı, 2020). Bu yöntemin bir başka özelliği ise geri besleme döngüsündeki ağırlıkların, her zaman sabit kalmayıp her zaman güncellenmesidir. Bu durum ile önceki girdilerin bilgileri uzun süreli belleklerine entegre edebilebilmesine olanak tanımış olur. Model bu sayede, uzun vadeli bağımlılıkları öğrenebilir ve zaman serilerindeki trendleri ve döngüleri tanıyabilir (Hochreiter & Schmidhuber, 1997; Turhan, 2019; Asaad et al., 2022).

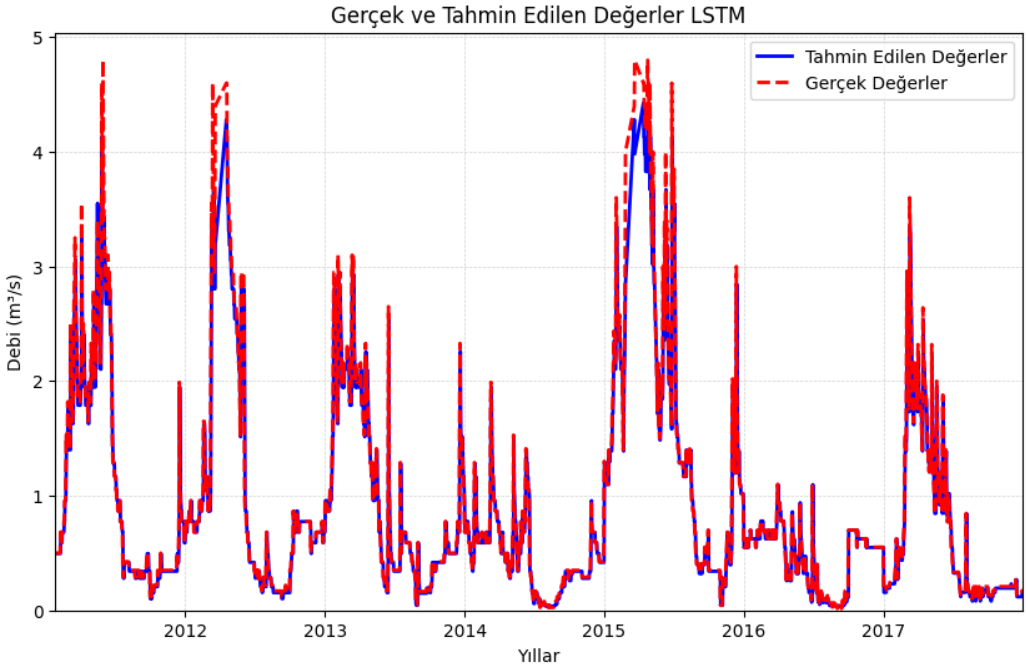
5.4.7.1 Günlük Debi Verileri Kullanılarak LSTM Metodu ile Tahminleme:

Günlük debi verileri kullanılarak uygulanan yöntem ile elde edilen hata değerleri Tablo 5.21'de gösterilmiştir.

Günlük Debi Verileri ile LSTM Metot	
Hata Metrikleri	Hata Değerleri
RMSE	0,20
MAE	0,08
R^2	0,95
MSE	0,04

Tablo 5.21: LSTM metodu ile yapılan tahminleme ve gerçek değerler arasındaki hata metrikleri

LSTM metodu tahminleme yöntemlerinde öncelikle günlük veriler kullanılarak elde edilen sonuçlara göre tahmin edilen ve gerçek değerler arasındaki karşılaştırma grafiği Şekil 5.23'de gösterilmektedir.



Şekil 5.23: LSTM metodu ile yapılan tahminleme sonuçlarının ve gerçek verilerin karşılaştırma grafiği. Bu tahminlemede günlük veriler kullanılmıştır.

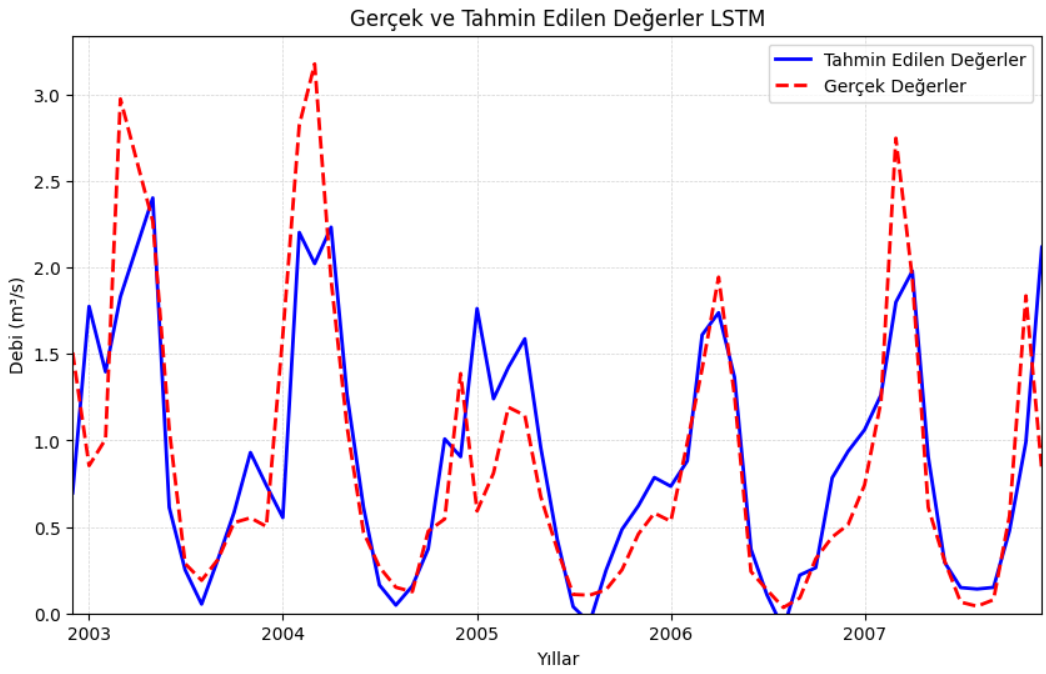
5.4.7.2 Aylık Ortalama Debi Verileri ile LSTM Metot i le Tahminleme:

Aylık ortalama debi verileri kullanılarak uygulanan yöntem ile elde edilen hata değerleri Tablo 5.22'de gösterilmiştir.

Aylık Ortalama Debi Verileri ile LSTM Metot	
Hata Metrikleri	Hata Değerleri
RMSE	0,49
MAE	0,36
R^2	0,62
MSE	0,24

Tablo 5.22: LSTM metodu ile yapılan tahminleme ve gerçek değerler arasındaki hata metrikleri

LSTM metodu tahminleme yöntemlerinde aylık ortalama veriler kullanılarak elde edilen sonuçlara göre tahmin edilen ve gerçek değerler arasındaki karşılaştırma grafiği Şekil 5.24'de gösterilmektedir.



Şekil 5.24: LSTM metodu ile yapılan tahminleme sonuçlarının ve gerçek verilerin karşılaştırma grafiği. Bu tahminlemede aylık ortalama veriler kullanılmıştır

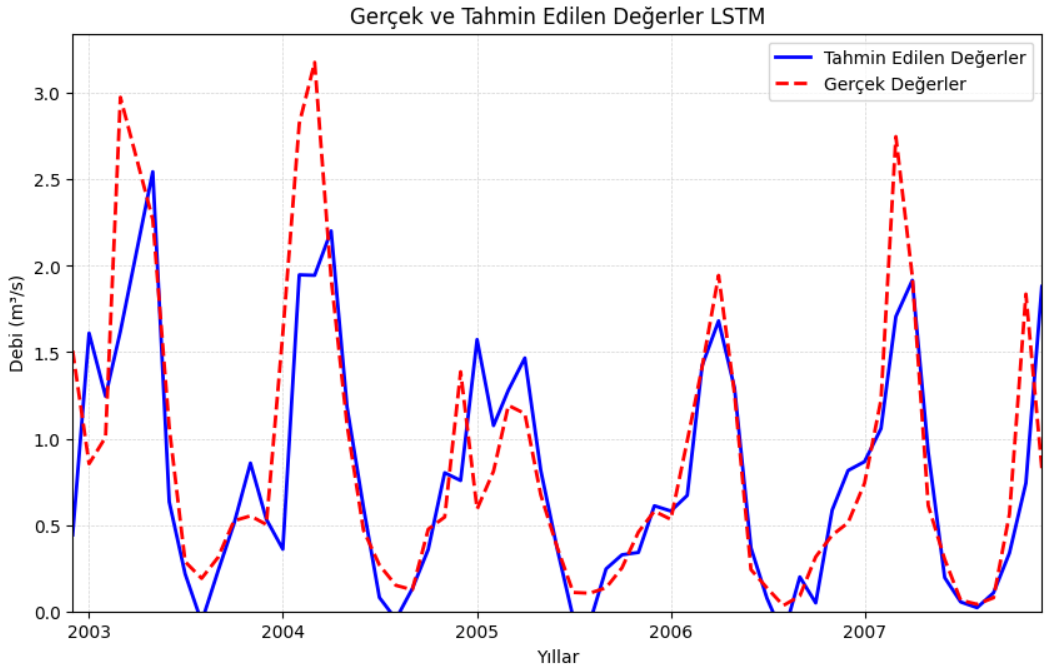
5.4.7.3 Aylık Ortalama Debi Verileri ve Özellik Veri Kümeleri Kullanılarak LSTM Metodu ile Tahminleme:

Aylık ortalama debi verileri kullanılmış ve özellik veri kümeleri eklenerek uygulanan yöntem ile elde edilen hata değerleri Tablo 5.23'de gösterilmiştir.

Aylık Ortalama Debi Verileri ile LSTM Metot	
Hata Metrikleri	Hata Değerleri
RMSE	0,48
MAE	0,33
R^2	0,63
MSE	0,23

Tablo 5.23: LSTM metodu ile yapılan tahminleme ve gerçek değerler arasındaki hata metrikleri

LSTM metodu tahminleme yöntemlerinde aylık ortalama veriler kullanılmıştır ve özellik veri kümeleri eklenmiştir. Elde edilen sonuçlara göre tahmin edilen ve gerçek değerler arasındaki karşılaştırma grafiği Şekil 5.25'de gösterilmektedir.



Şekil 5.25: LSTM metodu ile yapılan tahminleme sonuçlarının ve gerçek verilerin karşılaştırma grafiği. Bu tahminlemede aylık ortalama veriler kullanılmıştır ve özellik veri kümeleri eklenmiştir.

6 UYGULANAN METOTLARIN KARŞILAŞTIRILMASI

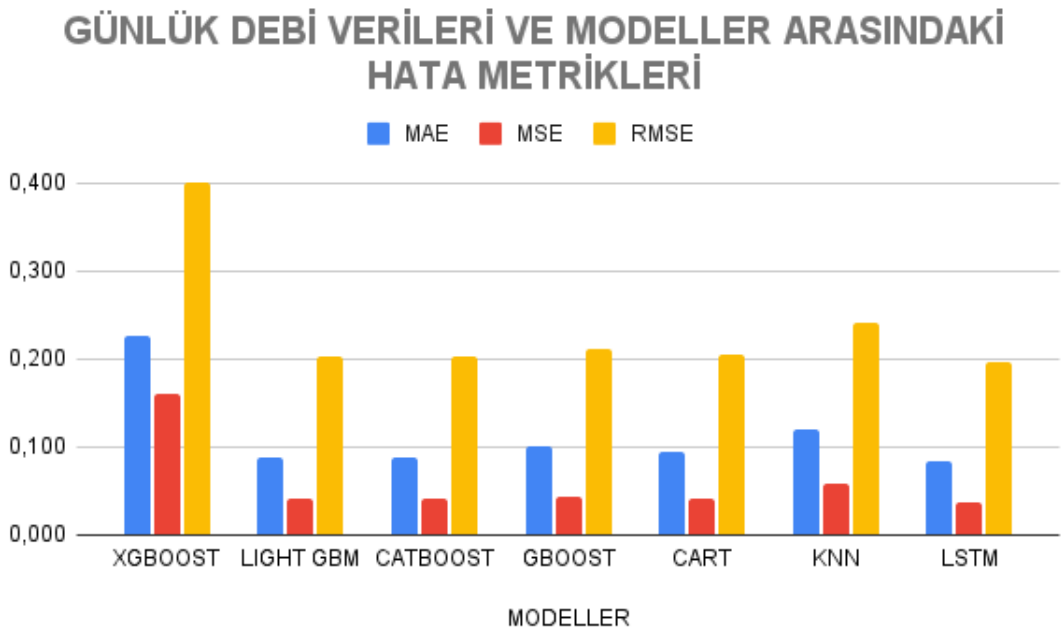
GBM, XGBM, CATBOOST, LGBM, KNN, CART ve LSTM algoritmaları kullanılmıştır. R^2, MSE, RMSE ve MAE hata metrikleri bulunmuş ve birbirleri ile kıyaslamaları yapılmıştır.

6.1 Günlük Debi Verileri Kullanılarak Yapılan Tüm Tahminleme Metotlarının Karşılaştırılması

Tüm kullanılan tahminleme metotlarına ait hata metrikleri Tablo 6.1'de gösterilmiştir. LSTM metodu değerlendirilen tüm metriklerde diğerlerine kıyasla eşit veya daha iyi bir performans sergilemiştir.

Günlük Debi Verileri Kullanılarak Hata Metrikleri Tablosu				
Metot	MAE	MSE	RMSE	R^2
XGBM	0,22	0,16	0,40	0,80
LIGHT GBM	0,09	**0,04**	**0,20**	**0,95**
CATBOOST	0,09	**0,04**	**0,20**	**0,95**
GBM	0,10	**0,04**	0,21	0,94
CART	0,10	**0,04**	**0,20**	**0,95**
KNN	0,12	0,06	0,24	0,93
LSTM	**0,08**	**0,04**	**0,20**	**0,95**

Tablo 6.1: Kullanılan metotların hata metrikleri gösterilmiştir.



Şekil 6.1: Kullanılan metotlar ve hata metrikleri

6.1.1 Günlük Debi Verileri Kullanılarak Uygulanan Tüm Metodların Değerlendirilmesi

Günlük debi verileri kullanılarak yapılan tahminleme metotları arasında en başarılı olan metodun, LSTM metodu olduğu görülmüştür. Karşılaştırma tablosu incelendiğinde Cat-Boost ve Light GBM metotlarının hata değerlerinin birbirine bu kadar yakın olmasının sebebi ise, diğer algoritmalardan farklı olarak kategorik özellikleri doğrudan işleyebilir ve eksik verileri de kullanabilir olmasıdır. Ayrıca her ikisinin de boosting tabanlı algoritmalar olması ile benzer özelliklere ve parametrelere sahip olduğu düşünülmektedir (Loshchilov & Ihler, 2017; Ke et al., 2017). Bu nedenle, bu iki algoritma aynı veri kümesi üzerinde benzer performans göstermesi ve sonuçlarının benzer olması beklenen bir durumdur.

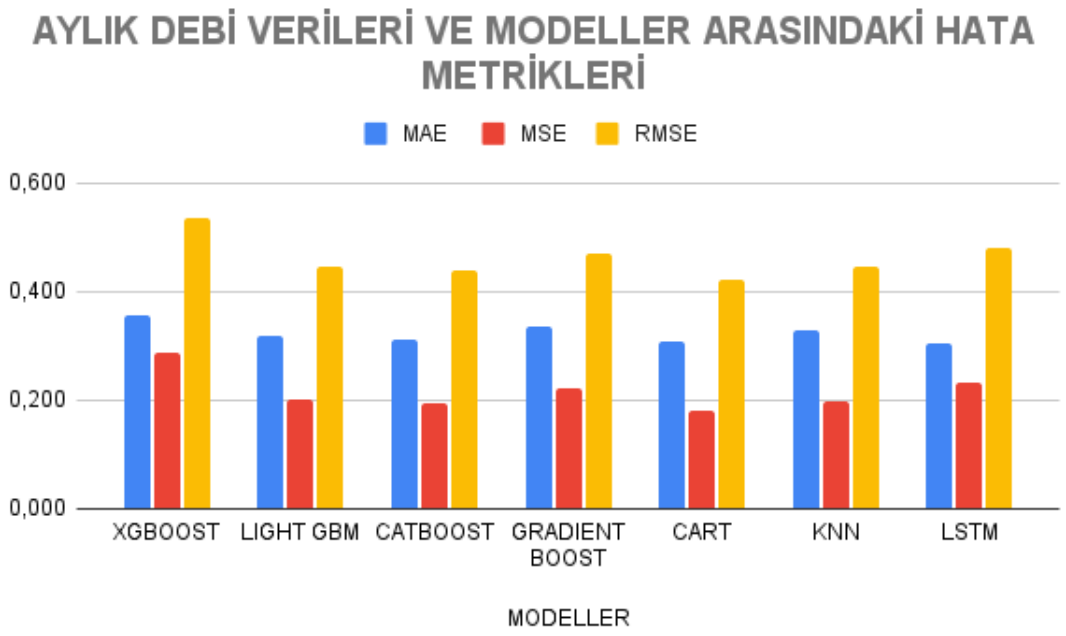
LSTM yöntemi, zaman serisi verilerinin tahmininde oldukça başarılı bir yöntemdir. Bunun nedeni ise LSTM'nin geçmiş verileri hafızasında saklayabilmesi ve bu verileri gelecekteki tahminler için kullanabilmesidir. (Hochreiter & Schmidhuber, 1997)

6.2 Aylık Debi Verileri Kullanılarak Yapılan Tüm Tahminleme Metotlarının Karşılaştırılması

Aylık ortalama veriler ile yapılan çalışmalarda uygulanan tahminleme metotlarına ait hata metrikleri Tablo 6.2'de gösterilmiştir. CART metodu değerlendirilen tüm metriklerde diğerlerine kıyasla eşit veya daha iyi bir performans sergilemiştir.

Aylık Debi Verileri Kullanılarak Hata Metrikleri Tablosu				
Metot	MAE	MSE	RMSE	R^2
XGBM	0,36	0,29	0,53	0,54
LIGHT GBM	0,32	0,20	0,45	0,68
CATBOOST	**0,31**	0,19	0,44	0,69
GBM	0,33	0,22	0,47	0,64
CART	**0,31**	**0,18**	**0,42**	**0,71**
KNN	0,33	0,20	0,45	0,68
LSTM	0,36	0,24	0,49	0,62

Tablo 6.2: Kullanılan metotlar ve hata metrikleri gösterilmiştir.



Şekil 6.2: Kullanılan metotlar ve hata metrikleri

6.2.1 Aylık Debi Verileri Kullanılarak Uygulanan Tüm Metotların Değerlendirilmesi

Sadece aylık debi verileri kullanılarak yapılan tahminleme metotları karşılaştırıldığında CART yönteminin diğer yöntemlere göre daha başarılı sonuçlar verdiği görülmüştür. CART metodunun tercih edilmesinin sebebi, ağaç yapısını oluştururken veri kümesine aşırı uyum sağlama (İng., overfitting) veya yetersiz uyum sağlama (İng., underfitting) sorunlarına karşı hassas olmasından dolayı diğer yöntemlere göre daha doğru ve istikrarlı sonuçlar üretmesidir. CART metodunun diğer yöntemlere göre daha başarılı olmasının sebebi veri kümesine en iyi entegre olabilen yöntem olmasından kaynaklanıyor olabilir. CART metodu, veri kümesi içerisindeki yapısal özellikleri tanımlama ve kategorize etme yeteneğine sahip olmasından dolayı bu yöntem özellikle kategorik veya nominal veriler içeren veri kümelerinde diğer yöntemlere göre daha başarılı olabilir (Breiman et al., 1984; Kayri & Boysan, 2008). Başka bir olası neden ise, veri

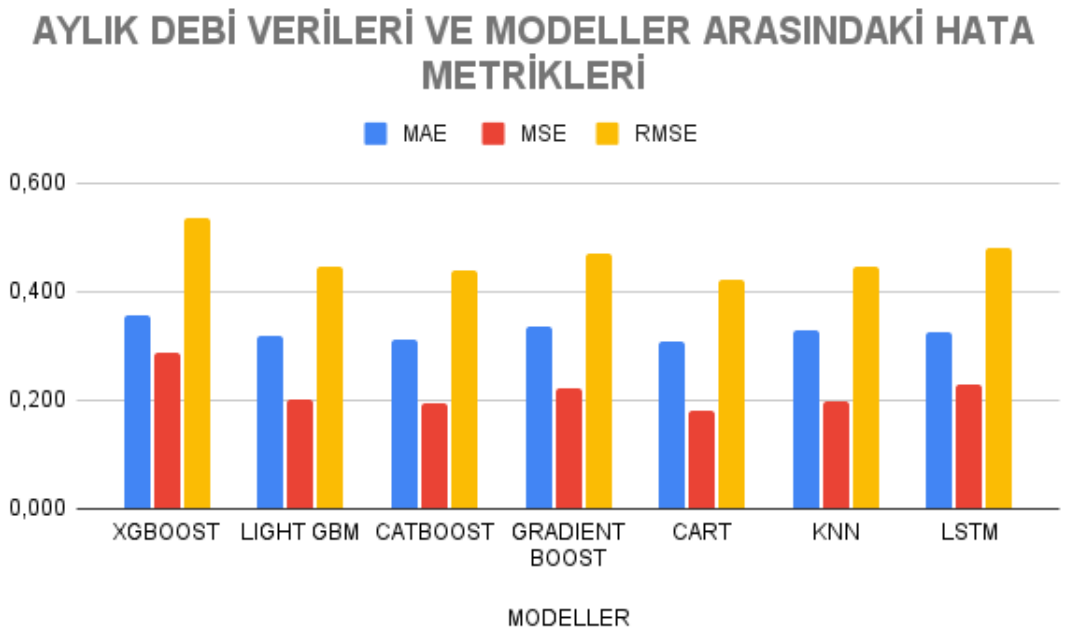
kümesindeki değişkenlerin birbirleriyle olan etkileşimleri ve doğrusallık düzeylerinden kaynaklıyor olması olabilir. CART yöntemi, değişkenler arasındaki etkileşimleri ve doğrusallık düzeylerini dikkate almaksızın, sadece her değişkenin ayrı ayrı etkisine odaklanır. Bu nedenle değişkenler arasındaki etkileşimler veya doğrusallık düzeyleri yüksek olan veri kümelerinde diğer yöntemlerin daha başarılı olması mümkündür (Hothorn, Hornik, & Zeileis, 2006; Kuhn & Johnson, 2013). Bu nedenle veri kümesine ve problem alanına özgü olarak, farklı yöntemlerin daha iyi sonuçlar verebileceği durumlar olabilir.

6.3 Aylık Debi Verileri ve Özellik Veri Kümesi Kullanılarak Yapılan Tüm Tahminleme Metotlarının Karşılaştırılması

Aylık ortalama veriler ve meteorolojik veri kümeleri ile yapılan çalışmalarda uygulanan tahminleme metotlarına ait hata metrikleri Tablo 6.3'de gösterilmiştir. CART metodu değerlendirilen tüm metriklerde diğerlerine kıyasla eşit veya daha iyi bir performans sergilemiştir.

Aylık Debi Verileri Kullanılarak Hata Metrikleri Tablosu				
Metot	MAE	MSE	RMSE	R^2
XGBM	0,36	0,29	0,53	0,54
LIGHT GBM	0,32	0,20	0,45	0,68
CATBOOST	**0,31**	0,19	0,44	0,69
GBM	0,34	0,22	0,47	0,66
CART	**0,31**	**0,18**	**0,42**	**0,71**
KNN	0,33	0,20	0,45	0,68
LSTM	0,33	0,23	0,48	0,63

Tablo 6.3: Kullanılan metotlar ve hata metrikleri gösterilmiştir.



Şekil 6.3: Kullanılan metotlar ve hata metrikleri

6.3.1 Aylık Debi Verileri ve Özellik Veri Kümeleri Kullanılarak Uygulanan Tüm Metotların Değerlendirilmesi

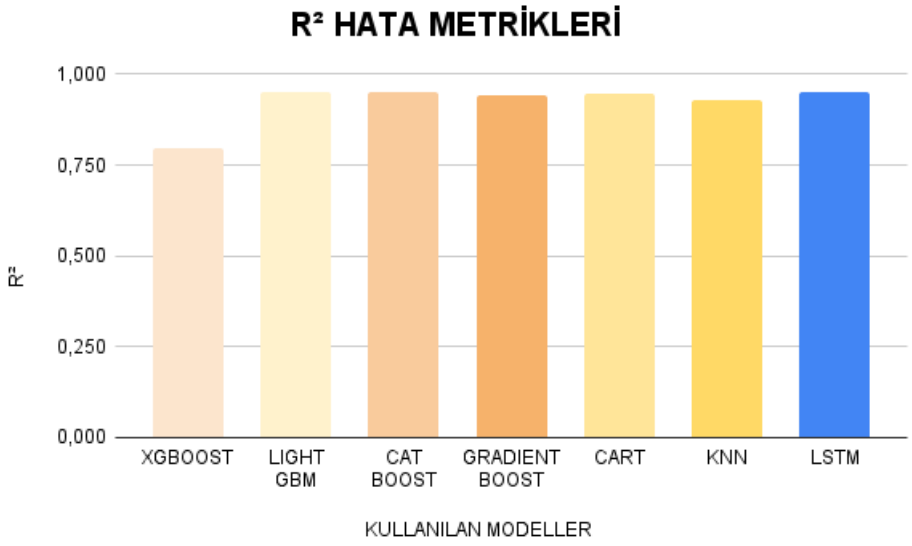
Aylık debi verileri ve meteorolojik veriler özellik verileri olarak tanımlanarak yapılan tahminleme metotları karşılaştırıldığında CART yönteminin diğer yöntemlere göre daha başarılı sonuçlar verdiği görülmüştür. Meteorolojik verilerin aylık bazda yapılan çalışmalar üzerine eklenmesinin sebebi, özellik verilerinin aslında Küçükmuhsine Çayı'nın debisini ne kadar etkileyebileceğini görmek içindir. Fakat kullanılan meteorolojik veri kümeleri her modele uymayabilir. Uygulanan tüm metotlar aylık bazda debi verileri kullanılarak yapılan çalışmaların tahminleme performansları üzerinde çok küçük etkiler göstermiştir.

6.4 Hata Metriklerine Göre Uygulanan Yöntemlerin Değerlendirilmesi

Bu tez çalışması kapsamında günlük veriler ile uygulanan metotlar için, hata metrikleri bazında yapılan karşılaştırmalar sonucunda en iyi sonucu LSTM, Light GBM, CATBOOST ve CART modellerinin verdiği Tablo 6.4'de görülmüştür.

Günlük Debi Verileri Kullanılarak Elde Edilen R^2 Değer Tablosu	
Model	R^2
XGBM	0,80
LIGHT GBM	**0,95**
CATBOOST	**0,95**
GBM	0,94
CART	**0,95**
KNN	0,93
LSTM	**0,95**

Tablo 6.4: Günlük debi verileri ile kullanılan metotlar ve R^2 gösterilmiştir.

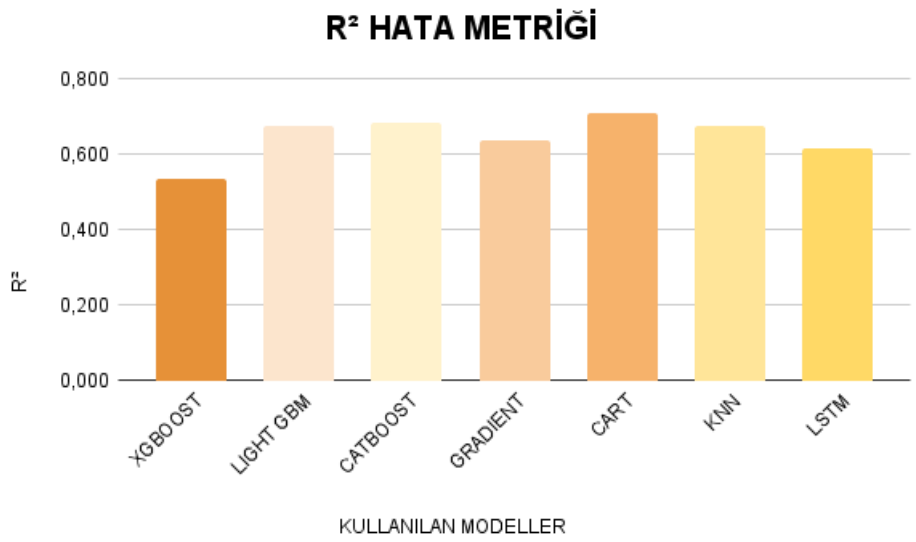


Şekil 6.4: Günlük debi verileri ile kullanılan metotlar ve bu metotlara ait R^2 değerleri gösterilmiştir.

Aylık ortalama veriler üzerinde uygulanan metotlar arasında hata metrikleri bazında yapılan karşılaştırmalar sonucu Tablo 6.5'de görüldüğü gibi en iyi sonucu CART metodu vermiştir.

Aylık Debi Verileri Kullanılarak Elde Edilen R^2 Değer Tablosu	
Model	R^2
XGBM	0,54
LIGHT GBM	0,68
CATBOOST	0,69
GBM	0,64
CART	**0,71**
KNN	0,68
LSTM	0,62

Tablo 6.5: Aylık debi verileri ile kullanılan metotlar ve R^2 değerleri gösterilmiştir.

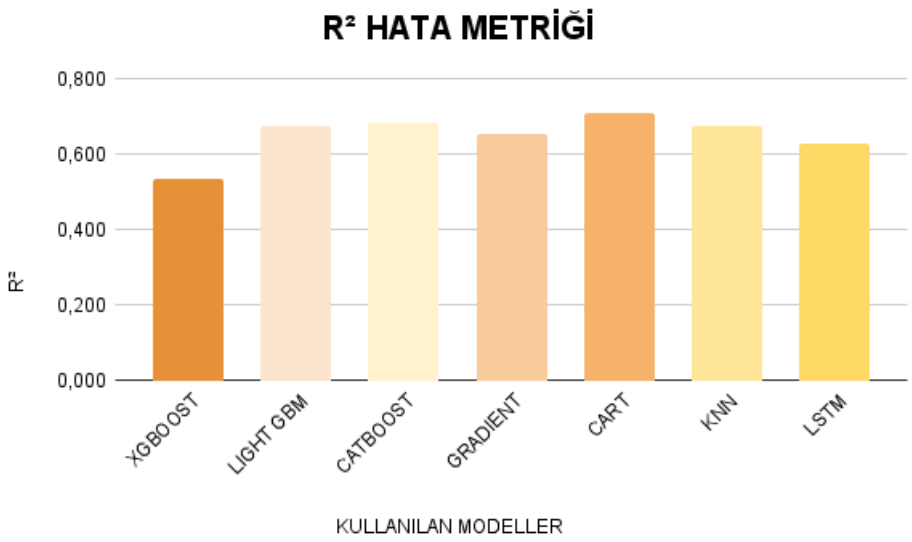


Şekil 6.5: Aylık debi verileri ile kullanılan metotlar ve R^2 değerleri gösterilmiştir.

Aylık veriler ve özellik veri kümeleri eklenerek yapılan karşılaştırmalar sonucunda ise Tablo 6.6'da görülğü gibi en iyi sonucu yine CART metodu vermiştir.

Aylık Debi Verileri ve Özellik Veri Kümeleri Kullanılarak Elde Edilen R^2 Değer Tablosu	
Model	R^2
XGBM	0,54
LIGHT GBM	0,68
CATBOOST	**0,69**
GBM	0,66
CART	0,71
KNN	0,68
LSTM	0,63

Tablo 6.6: Aylık debi verileri ve özellik veri kümeleri kullanılan metotlar ve R^2 değerleri gösterilmiştir.



Şekil 6.6: Aylık debi verileri ve özellik veri kümeleri kullanılan metotlar ve R^2 değerleri gösterilmiştir.

7 SONUÇ VE DEĞERLENDİRME

Yapılan çalışmalara göre su ve su kaynakları yönetimi ve su kaynaklarının verimli kullanılabilmesi adına gelecekteki su miktarının tahminlemesi yapılmıştır. Hata değerlerine bağlı olarak kullanılan makine öğrenmesi ve tahminleme yöntemleri ile çalışma tamamlanmıştır.

Bu tez çalışması sonucunda veri önişleme yöntemleri kullanılarak veri işleme hazır hale getirildikten sonra standart sapması, max ve min değerleri ve ortalaması bulunmuştur. Daha sonra belli yüzdelerde test ve eğitim verisi olarak ayrılmıştır. Veri kümesine uygun olacak şekilde veri önişleme yöntemleri seçilmiş ve uygulanmıştır. Eğitilen veriler sonucunda tahminleme yöntemlerinden yedi tanesi seçilmiş ve uygulanmıştır. Uygulanan yöntemlerin R^2, MAE, MSE ve RMSE değerleri bulunarak hata değerleri karşılaştırmaları yapılmıştır.

Yapılan karşılaştırmalar sonucunda günlük debi veri kümesi kullanılarak uygulanan modellerin ve aylık debi veri kümesi kullanılarak uygulanan modellerin eklenen özellikler ile uygulanan modellere göre daha iyi sonuç verdiği görülmektedir. Günlük debi veri kümesi ile uygulanan modeller arasında LSTM, Light GBM, CATBOOST ve CART modellerinin diğer modellere göre daha başarılı olduğu görülmektedir. Aylık debi veri kümesi ile yapılan çalışmalar sonucunda ve aylık veri kümesi ile beraber meteorolojik özellikler eklenerek uygulanan modeller arasında ise en başarılı sonuçlar CART modelinden alınmıştır.

EKLER

```
import pandas as pd
import numpy as np
import datetime
import os

features_data_path = "raw_data_features/all_data_raw.csv"


def min_max_scale(column):
    # Find the minimum and maximum values in the column
    min_val = column.min()
    max_val = column.max()

    # Scale the column using the min-max scaling formula
    scaled_column = (column - min_val) / (max_val - min_val)

    return scaled_column


def reset_2Dto1D(df_2d_year_month, month_dict=None):
    # Convert the table to a series of datetime and value pairs
    date_cols = df_2d_year_month.columns[1:]
    date_values = pd.melt(df_2d_year_month, id_vars=['Year'], value_vars=date_cols,
                          var_name='Month', value_name='Value')

    if month_dict is not None:
        date_values["Month"] = date_values["Month"].map(month_dict)
```

```
    date_values['Day'] = 1  # Set the day to 1 for all dates

    date_values['Date'] = pd.to_datetime(date_values[['Year', 'Month', 'Day']])

    date_values = date_values[['Date', 'Value']]

    return date_values


def detect_outliers_zscore(dataset, threshold=3):
    # Calculate the mean and standard deviation of the dataset
    mean = dataset.mean()
    std_dev = dataset.std()

    # Calculate the z-score for each data point
    z_scores = (dataset - mean) / std_dev

    # Identify the outliers in the dataset based on the z-score threshold
    outliers = dataset[z_scores.abs() < threshold]

    return outliers


def trim_replace(X, lower_percentile=0, upper_percentile=0):
    # Compute the percentile thresholds for trimming
    lower_threshold = np.percentile(X, lower_percentile)
    upper_threshold = np.percentile(X, 100 - upper_percentile)

    # Identify the values to replace
```

```
    replace_mask = np.logical_or(X < lower_threshold, X > upper_threshold)

    # Replace the values with the nearest non-outlier values
    X_trimmed = X.copy()
    for i in range(len(X)):
        if replace_mask[i]:
            if i == 0:
                X_trimmed[i] = X[np.argmin(np.abs(X[i+1:] - X[i]))]
            elif i == len(X) - 1:
                X_trimmed[i] = X[np.argmin(np.abs(X[:i] - X[i]))]
            else:
                X_trimmed[i] = X[np.argmin(
                    np.abs(np.concatenate([X[:i], X[i+1:]]) - X[i]))]

    return X_trimmed


def load_data():
    df = pd.read_csv(os.path.join(features_data_path), index_col=[0])

    values = {"Date": [], "target_flow_rate": []}
    for month, value in df.items():
        for mday_year in value.keys():
            m_day = mday_year.split('-')[0]
            year = mday_year.split('-')[1]
            if year.startswith("0") or year.startswith("1"):
                year = "20" + year
            else:
                year = "19" + year
```

```
86
87          try:
88              date = datetime.date(int(year), int(month), int(m_day))
89          except ValueError as e:
90              continue
91
92          if isinstance(value[mday_year], float):
93              river_rate = value[mday_year]
94          else:
95              try:
96                  river_rate = float(value[mday_year].replace(",", "."))
97              except ValueError as e:
98                  river_rate = np.nan
99
100         values["Date"].append(date)
101         values["target_flow_rate"].append(river_rate)
102
103     return pd.DataFrame.from_dict(values)
104
105
106 df = load_data()
107 df.sort_values("Date", inplace=True)
108 df["target_flow_rate"] = detect_outliers_zscore(
109     df["target_flow_rate"], threshold=3)
110
111 df["previous_flow_rate"] = df["target_flow_rate"].shift(1)
112 df["previous_flow_rate"] = min_max_scale(df["previous_flow_rate"])
113
114 df.dropna(inplace=True)
```

```
115 print(df)
116
117 df.to_csv("all_data_out.csv")
118 import numpy as np
119 import matplotlib.pyplot as plt
120 import os
121 from datetime import datetime
122 from settings import *
123
124
125 def image_1(model_name, months, flow_rates):
126     # Create a figure object and set size to 10x6 inches
127     fig, ax = plt.subplots(figsize=(10, 6))
128     months = [datetime.strptime(date_str, '%Y-%m-%d') for date_str in months]
129
130     # Plot the flow rates with a solid blue line and a marker for each data point
131     ax.plot(months, flow_rates, '-o', color='blue', linewidth=2,
132             markersize=5, label='average flow rate')
133
134     # Add gridlines and set the x and y-axis limits
135     ax.grid(color='lightgray', linestyle='--', linewidth=0.5)
136     ax.set_xlim(min(months), max(months))
137     ax.set_ylim(bottom=0)
138
139     # Add a title and axis labels
140     ax.set_title(f"Monthly Average Flow Rates for {model_name}")
141     ax.set_xlabel('Month')
142     ax.set_ylabel('Flow Rate')
143
```

```
144     # Add legend and adjust its position
145     ax.legend(loc='best', bbox_to_anchor=(1, 1))
146
147     # Save the plot to a file and clear the plot object
148     plt.savefig(f"images/{model_name}_image1.png", bbox_inches='tight')
149     plt.clf()
150
151
152 def image_2(model_name, dates, actual_flow_rates, predicted_flow_rates):
153     # Convert string-based dates to datetime objects
154     dates = [datetime.strptime(date_str, '%Y-%m-%d') for date_str in dates]
155
156     # Create a figure object and set size to 10x6 inches
157     fig, ax = plt.subplots(figsize=(10, 6))
158
159     # Plot the predicted and actual flow rates with different line styles and colors
160     ax.plot(dates, predicted_flow_rates, '-',
161             color='blue', linewidth=2, label='Predicted')
162     ax.plot(dates, actual_flow_rates, '--',
163             color='red', linewidth=2, label='Actual')
164
165     # Add gridlines and set the x and y-axis limits
166     ax.grid(color='lightgray', linestyle='--', linewidth=0.5)
167     ax.set_xlim(min(dates), max(dates))
168     ax.set_ylim(bottom=0)
169
170     # Add a title and axis labels
171     ax.set_title(f"Actual vs Predicted Flow Rates for {model_name}")
172     ax.set_xlabel('Month')
```

```
173     ax.set_ylabel('Flow Rate')
174
175     # Add legend and adjust its position
176     ax.legend(loc='best', bbox_to_anchor=(1, 1))
177
178     # Save the plot to a file and clear the plot object
179     plt.savefig(f"images/{model_name}_image2.png", bbox_inches='tight')
180     plt.clf()
181
182
183 def save_images(model_name, train_data, test_data, pred_y):
184     if not os.path.isdir("images"):
185         os.mkdir("images")
186     test_index = list(test_data["Date"])
187     train_index = list(train_data["Date"])
188     image_1("All Data", train_index, train_data[target_column])
189     image_1("Test Data", test_index, test_data[target_column])
190     image_2(model_name, test_index, test_data[target_column], pred_y)
191
192
193 def plot_comparison(model_names, delta_times, graph_values):
194     # Create a figure object and set size to 10x6 inches
195     fig, ax1 = plt.subplots(figsize=(10, 6))
196
197     # Create a bar chart of delta_times on the left y-axis
198     ax1.bar(model_names, delta_times, color='g', width=0.4)
199     ax1.set_ylabel("Time (s)", fontsize=14, fontweight='bold', color='g')
200
201     # Set the x-tick labels and rotate them 45 degrees
```

```
202     ax1.set_xticklabels(model_names, rotation=45, ha='right', fontsize=12)
203
204     # Create a line chart of graph_values on the right y-axis
205     ax2 = ax1.twinx()
206     ax2.plot(model_names, graph_values, color='r', linewidth=2.5)
207     ax2.set_ylabel("Score", fontsize=14, fontweight='bold', color='r')
208
209     # Set the y-axis tick labels for ax2
210     ax2.set_yticks(np.arange(0, max(graph_values)+1, 0.5*max(graph_values)))
211     ax2.tick_params(axis='y', labelcolor='r')
212     ax2.set_ylim(0, 1)
213
214     # Add a title and axis labels
215     ax1.set_title("Comparison of Model Times and Scores",
216                   fontsize=16, fontweight='bold')
217     ax1.set_xlabel("Model Name", fontsize=14, fontweight='bold')
218
219     # Save the plot to a file and show the plot
220     plt.tight_layout()
221     plt.savefig("ModelAnalyze.png")
222
223 from models.model_CatBoost import model_catboost
224 from models.model_GradientBoost import model_gradiendtBoost
225 from models.model_lightGBM import model_lightGBM
226 from models.model_XGB import model_XGBoost
227 from models.model_KNeighbors import model_KNeighbors
228 from models.model_CART import model_CART
229 from visualization import plot_comparison
230 import pandas as pd
```

```
231
232
233 functions = {
234     "CatBoost": model_catboost,
235     "GradientBoost": model_gradiendtBoost,
236     "LightGBM": model_lightGBM,
237     "XGBoost": model_XGBoost,
238     "KNeighbors": model_KNeighbors,
239     "CART": model_CART
240 }
241
242 model_names = []
243 graph_values = []
244 deltaTimes = []
245
246 results = []
247
248 for model, func in functions.items():
249     mae, mse, rmse, r2, dt = func()
250
251     results.append([model, dt, mae, mse, rmse, r2])
252
253     model_names.append(model)
254     graph_values.append(r2)
255     deltaTimes.append(dt)
256
257 result_df = pd.DataFrame(
258     results, columns=["Model", "deltaTime", "mae", "mse", "rmse", "R2"])
259
```

```
result_df.to_csv("model_analyzer_results.csv",
                 index=False, float_format="%.3f")

plot_comparison(model_names, deltaTimes, graph_values)
print(list(zip(model_names, graph_values, deltaTimes)))
```

Kod 1: Veri önişleme (Günlük veriler için) (Keskin, 2021)

```
import pandas as pd
import numpy as np
import datetime
import os

features_data_dir = "raw_data_features"
target_data_path = "target_monthly_avarage_flowrates.tsv"

data_file_paths = {
    p.split('_', 1)[1].split(".")[0]: os.path.join(features_data_dir, p)
    for p in os.listdir(features_data_dir)
}


def min_max_scale(column):
    # Find the minimum and maximum values in the column
    min_val = column.min()
    max_val = column.max()

    # Scale the column using the min-max scaling formula
    scaled_column = (column - min_val) / (max_val - min_val)
```

```
22
23      return scaled_column
24
25
26  def reset_2Dto1D(df_2d_year_month, month_dict=None):
27      # Convert the table to a series of datetime and value pairs
28      date_cols = df_2d_year_month.columns[1:]
29      date_values = pd.melt(df_2d_year_month, id_vars=['Year'], value_vars=date_cols,
30                            var_name='Month', value_name='Value')
31
32      if month_dict is not None:
33          date_values["Month"] = date_values["Month"].map(month_dict)
34      date_values['Day'] = 1  # Set the day to 1 for all dates
35
36      date_values['Date'] = pd.to_datetime(date_values[['Year', 'Month', 'Day']])
37
38      date_values = date_values[['Date', 'Value']]
39
40      return date_values
41
42
43  def detect_outliers_zscore(dataset, threshold=3):
44      # Calculate the mean and standard deviation of the dataset
45      mean = dataset.mean()
46      std_dev = dataset.std()
47
48      # Calculate the z-score for each data point
49      z_scores = (dataset - mean) / std_dev
50
```

```
    # Identify the outliers in the dataset based on the z-score threshold
    outliers = dataset[z_scores.abs() < threshold]

    return outliers


def trim_replace(X, lower_percentile=0, upper_percentile=0):
    # Compute the percentile thresholds for trimming
    lower_threshold = np.percentile(X, lower_percentile)
    upper_threshold = np.percentile(X, 100 - upper_percentile)

    # Identify the values to replace
    replace_mask = np.logical_or(X < lower_threshold, X > upper_threshold)

    # Replace the values with the nearest non-outlier values
    X_trimmed = X.copy()
    for i in range(len(X)):
        if replace_mask[i]:
            if i == 0:
                X_trimmed[i] = X[np.argmin(np.abs(X[i+1:] - X[i]))]
            elif i == len(X) - 1:
                X_trimmed[i] = X[np.argmin(np.abs(X[:i] - X[i]))]
            else:
                X_trimmed[i] = X[np.argmin(
                    np.abs(np.concatenate([X[:i], X[i+1:]]) - X[i]))]

    return X_trimmed


```

```
features_df = pd.DataFrame({"Date": []})

# Feature handling
for feature_name, feature_path in data_file_paths.items():
    df = pd.read_csv(feature_path, skiprows=7, sep="\t")

    if "YILLIK" in df.columns:
        df.drop("YILLIK", axis=1, inplace=True)
    if "Y.MAX." in df.columns:
        df.drop("Y.MAX.", axis=1, inplace=True)

    df.rename(columns={"YIL": "Year"}, inplace=True)
    month_dict = {m_name: m_id+1 for m_id, m_name in enumerate(df.columns[1:])}
    df_1D = reset_2Dto1D(df, month_dict)
    df_1D["Value"] = detect_outliers_zscore(df_1D["Value"], threshold=3)
    # df_1D["Value"] = trim_replace(df_1D["Value"], 1, 1)
    df_1D["Value"] = min_max_scale(df_1D["Value"])

    df_1D.rename(columns={"Value": feature_name}, inplace=True)

    features_df = pd.merge(features_df, df_1D, on="Date", how="outer")

# Target Handling
target_df = pd.read_csv(target_data_path, sep="\t")
target_df.rename(columns={"YILLAR": "Year"}, inplace=True)
target_df = reset_2Dto1D(target_df).sort_values("Date")
feature_as_target = target_df.copy()

feature_as_target["Value"] = target_df["Value"].shift(1)
```

```
feature_as_target.dropna(inplace=True)

feature_as_target["Value"] = min_max_scale(feature_as_target["Value"])
feature_as_target.rename(columns={"Value": "previous_flow_rate"}, inplace=True)
features_df = pd.merge(features_df, feature_as_target, on="Date", how="outer")

# target_df["Value"] = min_max_scale(target_df["Value"])
target_df["Value"] = detect_outliers_zscore(target_df["Value"], threshold=3)

target_df.rename(columns={"Value": "target_flow_rate"}, inplace=True)
all_data = pd.merge(target_df, features_df, how="left", on="Date")

notna_data = all_data[pd.notna(all_data["target_flow_rate"])]
filled_data = notna_data.fillna(method="bfill")
filled_data.ffill(inplace=True)
print(filled_data[["target_flow_rate", "previous_flow_rate"]])

filled_data.to_csv("all_data_out.csv")
import numpy as np
import matplotlib.pyplot as plt
import os
from datetime import datetime
from settings import *


def image_1(model_name, months, flow_rates):
    # Create a figure object and set size to 10x6 inches
    fig, ax = plt.subplots(figsize=(10, 6))
    months = [datetime.strptime(date_str, '%Y-%m-%d') for date_str in months]
```

```
138
139     # Plot the flow rates with a solid blue line and a marker for each data point
140     ax.plot(months, flow_rates, '-o', color='blue', linewidth=2,
141             markersize=5, label='average flow rate')
142
143     # Add gridlines and set the x and y-axis limits
144     ax.grid(color='lightgray', linestyle='--', linewidth=0.5)
145     ax.set_xlim(min(months), max(months))
146     ax.set_ylim(bottom=0)
147
148     # Add a title and axis labels
149     ax.set_title(f"Monthly Average Flow Rates for {model_name}")
150     ax.set_xlabel('Month')
151     ax.set_ylabel('Flow Rate')
152
153     # Add legend and adjust its position
154     ax.legend(loc='best', bbox_to_anchor=(1, 1))
155
156     # Save the plot to a file and clear the plot object
157     plt.savefig(f"images/{model_name}_image1.png", bbox_inches='tight')
158     plt.clf()
159
160
161 def image_2(model_name, dates, actual_flow_rates, predicted_flow_rates):
162     # Convert string-based dates to datetime objects
163     dates = [datetime.strptime(date_str, '%Y-%m-%d') for date_str in dates]
164
165     # Create a figure object and set size to 10x6 inches
166     fig, ax = plt.subplots(figsize=(10, 6))
```

```
167
168     # Plot the predicted and actual flow rates with different line styles and colors
169     ax.plot(dates, predicted_flow_rates, '-',
170             color='blue', linewidth=2, label='Predicted')
171     ax.plot(dates, actual_flow_rates, '--',
172             color='red', linewidth=2, label='Actual')
173
174     # Add gridlines and set the x and y-axis limits
175     ax.grid(color='lightgray', linestyle='--', linewidth=0.5)
176     ax.set_xlim(min(dates), max(dates))
177     ax.set_ylim(bottom=0)
178
179     # Add a title and axis labels
180     ax.set_title(f"Actual vs Predicted Flow Rates for {model_name}")
181     ax.set_xlabel('Month')
182     ax.set_ylabel('Flow Rate')
183
184     # Add legend and adjust its position
185     ax.legend(loc='best', bbox_to_anchor=(1, 1))
186
187     # Save the plot to a file and clear the plot object
188     plt.savefig(f"images/{model_name}_image2.png", bbox_inches='tight')
189     plt.clf()
190
191
192 def save_images(model_name, train_data, test_data, pred_y):
193     if not os.path.isdir("images"):
194         os.mkdir("images")
195     test_index = list(test_data["Date"])
```

```
    train_index = list(train_data["Date"])
    image_1("All Data", train_index, train_data[target_column])
    image_1("Test Data", test_index, test_data[target_column])
    image_2(model_name, test_index, test_data[target_column], pred_y)


def plot_comparison(model_names, delta_times, graph_values):
    # Create a figure object and set size to 10x6 inches
    fig, ax1 = plt.subplots(figsize=(10, 6))

    # Create a bar chart of delta_times on the left y-axis
    ax1.bar(model_names, delta_times, color='g', width=0.4)
    ax1.set_ylabel("Time (s)", fontsize=14, fontweight='bold', color='g')

    # Set the x-tick labels and rotate them 45 degrees
    ax1.set_xticklabels(model_names, rotation=45, ha='right', fontsize=12)

    # Create a line chart of graph_values on the right y-axis
    ax2 = ax1.twinx()
    ax2.plot(model_names, graph_values, color='r', linewidth=2.5)
    ax2.set_ylim(0, 1)
    ax2.set_ylabel("Score", fontsize=14, fontweight='bold', color='r')

    # Set the y-axis tick labels for ax2
    ax2.set_yticks(np.arange(0, max(graph_values)+1, 0.5*max(graph_values)))
    ax2.tick_params(axis='y', labelcolor='r')

    # Add a title and axis labels
    ax1.set_title("Comparison of Model Times and Scores", fontsize=16, fontweight='bold')
```

```
225     ax1.set_xlabel("Model Name", fontsize=14, fontweight='bold')
226
227     # Save the plot to a file and show the plot
228     plt.tight_layout()
229     plt.savefig("ModelAnalyze.png")
230 from models.model_CatBoost import model_catboost
231 from models.model_GradientBoost import model_gradiendtBoost
232 from models.model_lightGBM import model_lightGBM
233 from models.model_XGB import model_XGBoost
234 from models.model_KNeighbors import model_KNeighbors
235 from models.model_CART import model_CART
236 from visualization import plot_comparison
237 import pandas as pd
238
239
240 functions = {
241     "CatBoost": model_catboost,
242     "GradientBoost": model_gradiendtBoost,
243     "LightGBM": model_lightGBM,
244     "XGBoost": model_XGBoost,
245     "KNeighbors": model_KNeighbors,
246     "CART": model_CART
247 }
248
249 model_names = []
250 graph_values = []
251 deltaTimes = []
252
253 results = []
```

```
254
255 for model, func in functions.items():
256     mae, mse, rmse, r2, dt = func()
257
258     results.append([model, dt, mae, mse, rmse, r2])
259
260     model_names.append(model)
261     graph_values.append(r2)
262     deltaTimes.append(dt)
263
264 result_df = pd.DataFrame(
265     results, columns=["Model", "deltaTime", "mae", "mse", "rmse", "r2"])
266
267 result_df.to_csv("model_analyzer_results.csv",
268                  index=False, float_format="%.3f")
269
270 plot_comparison(model_names, deltaTimes, graph_values)
271 print(list(zip(model_names, graph_values, deltaTimes)))
```

Kod 2: Veri önişleme (Aylık veriler için) (Keskin, 2021)

```
1 # Import the necessary modules
2 from catboost import CatBoostRegressor
3 from sklearn.metrics import r2_score
4 import numpy as np
5 import pandas as pd
6 from visualization import save_images
7 import time
8 from settings import *
```

```
9
10
11 def model_catboost():
12     # Load the data into a Pandas DataFrame
13     data = pd.read_csv('all_data_out.csv')
14
15     # Split the data into training and testing sets
16     train_data = data[:int(train_test_split * len(data))]
17     test_data = data[int(train_test_split * len(data)):]
18
19     train_X = train_data[train_features]
20     train_y = train_data[target_column]
21
22     test_X = test_data[train_features]
23     test_y = test_data[target_column]
24
25     params = {
26         'border_count': 100,
27         'depth': 3,
28         'iterations': 700,
29         'l2_leaf_reg': 7,
30         'learning_rate': 0.1
31     }
32
33     t0 = time.time()
34     # Initialize the CatBoostRegressor
35     model = CatBoostRegressor(**params)
36
37     # Train the model with the training data
```

```
    model.fit(train_X, train_y)

    # Make predictions on the test data
    predictions = model.predict(test_X)

    dt = time.time() - t0

    # Evaluate the performance of the model
    mae = np.mean(abs(predictions - test_y))
    mse = np.mean((predictions - test_y)**2)
    rmse = np.sqrt(mse)
    r2 = r2_score(test_y, predictions)

    print('MAE: %.3f' % mae)
    print('MSE: %.3f' % mse)
    print('RMSE: %.3f' % rmse)
    print('R2: %.3f' % r2)

    save_images("CatBoost", train_data, test_data, predictions)

    return [mae, mse, rmse, r2, dt]
```

Kod 3: Catboost metodu (Keskin, 2021)

```
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import r2_score
import numpy as np
import pandas as pd
from visualization import save_images
```

```
import time
from settings import *


def model_gradienDtBoost():
    # Load the data into a Pandas DataFrame
    data = pd.read_csv('all_data_out.csv')

    # Split the data into training and testing sets
    train_data = data[:int(train_test_split * len(data))]
    test_data = data[int(train_test_split * len(data)):]

    train_X = train_data[train_features]
    train_y = train_data[target_column]

    test_X = test_data[train_features]
    test_y = test_data[target_column]

    params = {
        'learning_rate': 0.01,
        'max_depth': 3,
        'max_features': 'sqrt',
        'min_samples_leaf': 8,
        'min_samples_split': 6,
        'n_estimators': 300
    }

    t0 = time.time()
    # create the gradient boost model
```

```
    model = GradientBoostingRegressor(**params)

    # train the model on the training data
    model.fit(train_X, train_y)

    # make predictions on the test data
    predictions = model.predict(test_X)

    dt = time.time() - t0

    # Evaluate the performance of the model
    mae = np.mean(abs(predictions - test_y))
    mse = np.mean((predictions - test_y)**2)
    rmse = np.sqrt(mse)
    r2 = r2_score(test_y, predictions)

    print('MAE: %.3f' % mae)
    print('MSE: %.3f' % mse)
    print('RMSE: %.3f' % rmse)
    print('R2: %.3f' % r2)

    save_images("Gradient", train_data, test_data, predictions)

    return [mae, mse, rmse, r2, dt]
```

Kod 4: GB metodu (Keskin, 2021)

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import r2_score
```

```
import numpy as np
import pandas as pd
from visualization import save_images
import time
from settings import *


def model_CART():
    # Load the data into a Pandas DataFrame
    data = pd.read_csv('all_data_out.csv')

    # Split the data into training and testing sets
    train_data = data[:int(train_test_split * len(data))]
    test_data = data[int(train_test_split * len(data)):]

    train_X = train_data[train_features]
    train_y = train_data[target_column]

    test_X = test_data[train_features]
    test_y = test_data[target_column]

    # Define the hyperparameters
    params = {
        'max_depth': 5,
        'min_samples_leaf': 4,
        'min_samples_split': 10
    }

    # Initialize the DecisionTreeRegressor
```

```
    model = DecisionTreeRegressor(**params)

    t0 = time.time()
    # Train the model with the training data
    model.fit(train_X, train_y)

    # Make predictions on the test data
    predictions = model.predict(test_X)

    dt = time.time() - t0

    # Evaluate the performance of the model
    mae = np.mean(abs(predictions - test_y))
    mse = np.mean((predictions - test_y)**2)
    rmse = np.sqrt(mse)
    r2 = r2_score(test_y, predictions)

    print('MAE: %.3f' % mae)
    print('MSE: %.3f' % mse)
    print('RMSE: %.3f' % rmse)
    print('R2: %.3f' % r2)

    save_images("DecisionTree", train_data, test_data, predictions)

    return [mae, mse, rmse, r2, dt]
```

Kod 5: CART metodu (Keskin, 2021)

```
from sklearn.neighbors import KNeighborsRegressor
```

```
from sklearn.metrics import r2_score
import numpy as np
import pandas as pd
from visualization import save_images
import time
from settings import *


def model_KNeighbors():
    # Load the data into a Pandas DataFrame
    data = pd.read_csv('all_data_out.csv')

    # Split the data into training and testing sets
    train_data = data[:int(train_test_split * len(data))]
    test_data = data[int(train_test_split * len(data)):]

    train_X = train_data[train_features]
    train_y = train_data[target_column]

    test_X = test_data[train_features]
    test_y = test_data[target_column]

    params = {
        'algorithm': 'kd_tree',
        'n_neighbors': 5,
        'p': 1,
        'weights': 'uniform'
    }

```

```
    # Initialize the KNeighborsRegressor with default parameters
    model = KNeighborsRegressor(**params)

    t0 = time.time()
    # Train the model with the training data
    model.fit(train_X, train_y)

    # Make predictions on the test data
    predictions = model.predict(test_X)

    dt = time.time() - t0

    # Evaluate the performance of the model
    mae = np.mean(abs(predictions - test_y))
    mse = np.mean((predictions - test_y)**2)
    rmse = np.sqrt(mse)
    r2 = r2_score(test_y, predictions)

    print('MAE: %.3f' % mae)
    print('MSE: %.3f' % mse)
    print('RMSE: %.3f' % rmse)
    print('R2: %.3f' % r2)

    save_images("KNeighbors", train_data, test_data, predictions)

    return [mae, mse, rmse, r2, dt]
```

Kod 6: KNN metodu (Keskin, 2021)

```
import lightgbm as lgb
from sklearn.metrics import r2_score
import pandas as pd
import numpy as np
from visualization import save_images
import time
from settings import *


def model_lightGBM():
    # Load the data into a Pandas DataFrame
    data = pd.read_csv('all_data_out.csv')

    # Split the data into training and testing sets
    train_data = data[:int(train_test_split * len(data))]
    test_data = data[int(train_test_split * len(data)):]

    train_X = train_data[train_features]
    train_y = train_data[target_column]

    test_X = test_data[train_features]
    test_y = test_data[target_column]

    # Create a LightGBM dataset
    train_dataset = lgb.Dataset(train_X, label=train_y)
    val_data = lgb.Dataset(test_X, label=test_y)

    # Define your LightGBM model
```

```
params = {
    'learning_rate': 0.1,
    'max_depth': 3,
    'min_child_samples': 3,
    'n_estimators': 100,
    'num_leaves': 127,
    'reg_alpha': 0.5,
    'reg_lambda': 0.01
}

t0 = time.time()

# Train your model
model = lgb.train(params, train_dataset, valid_sets=[train_dataset, val_data],
                  num_boost_round=1000, early_stopping_rounds=10)

# Make predictions on the validation set
predictions = model.predict(test_X)
dt = time.time() - t0

# Evaluate the performance of the model
mae = np.mean(abs(predictions - test_y))
mse = np.mean((predictions - test_y)**2)
rmse = np.sqrt(mse)
r2 = r2_score(test_y, predictions)

print('MAE: %.3f' % mae)
print('MSE: %.3f' % mse)
print('RMSE: %.3f' % rmse)
```

```
    print('R2: %.3f' % r2)

    save_images("LightGBM", train_data, test_data, predictions)

    return [mae, mse, rmse, r2, dt]
```

Kod 7: Light GBM metodu (Keskin, 2021)

```
# Import the necessary libraries
import xgboost as xgb
import pandas as pd
import numpy as np
from sklearn.metrics import r2_score
from visualization import save_images
import time
from settings import *


def model_XGBoost():
    # Load the data into a Pandas DataFrame
    data = pd.read_csv('all_data_out.csv')

    # Split the data into training and testing sets
    train_data = data[:int(train_test_split * len(data))]
    test_data = data[int(train_test_split * len(data)):]

    train_X = train_data[train_features].copy()
    train_y = train_data[target_column].copy()

```

```
test_X = test_data[train_features].copy()
test_y = test_data[target_column].copy()

# Convert the data into DMatrix format, which is
# the format that XGBoost expects
dtrain = xgb.DMatrix(train_X, label=train_y)
dtest = xgb.DMatrix(test_X, label=test_y)

# Define the parameters for the XGBoost model
params = {
    'colsample_bytree': 1.0,
    'gamma': 0.1,
    'learning_rate': 0.1,
    'max_depth': 5,
    'n_estimators': 1000,
    'reg_alpha': 0.1,
    'reg_lambda': 0.5,
    'subsample': 0.5
}

t0 = time.time()
# Train the model using the XGBoost library
model = xgb.train(params, dtrain)

# Use the trained model to make predictions on the test data
predictions = model.predict(dtest)

dt = time.time() - t0

```

```
    # Evaluate the performance of the model
    mae = np.mean(abs(predictions - test_y))
    mse = np.mean((predictions - test_y)**2)
    rmse = np.sqrt(mse)
    r2 = r2_score(test_y, predictions)

    print('MAE: %.3f' % mae)
    print('MSE: %.3f' % mse)
    print('RMSE: %.3f' % rmse)
    print('R2: %.3f' % r2)

    save_images("XGBoost", train_data, test_data, predictions)

    return [mae, mse, rmse, r2, dt]
```

Kod 8: XGBM metodu (Keskin, 2021)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score
from keras.models import Sequential
from keras.layers import LSTM, Dense
from keras.optimizers import Adam
from sklearn.preprocessing import MinMaxScaler
from settings import *
from visualization import save_images

n_steps = 7
```

```
epochs = 90
batch_size = 1
learning_rate = 0.001


def model_LSTM():
    # Load the data into a Pandas DataFrame
    data = pd.read_csv('all_data_out.csv')

    # Split the data into training and testing sets
    train_data = data[:int(train_test_split * len(data))]
    test_data = data[int(train_test_split * len(data)):]

    train_X = train_data[train_features].values.reshape(-1, batch_size, len(train_features))
    train_y = train_data[target_column].values.reshape(-1, batch_size)

    test_X = test_data[train_features].values.reshape(-1, batch_size, len(train_features))
    test_y = test_data[target_column].values.reshape(-1, batch_size)

    # Build the LSTM model
    model = Sequential()
    model.add(LSTM(64, activation='relu', input_shape=(batch_size, len(train_features))))
        # n_steps, len(train_features))))
    model.add(Dense(1))
    model.compile(optimizer=Adam(learning_rate=learning_rate), loss='mse')

    history = model.fit(train_X, train_y, epochs=epochs,
                        batch_size=batch_size, verbose=0)

```

```
42  # Make predictions on the test data
43  predictions = model.predict(test_X)
44
45  # Evaluate the performance of the model
46  mae = np.mean(abs(predictions - test_y))
47  mse = np.mean((predictions - test_y)**2)
48  rmse = np.sqrt(mse)
49  r2 = r2_score(test_y, predictions)
50
51  save_images("LSTM", train_data, test_data, list(predictions))
52
53  print('MAE: %.3f' % mae)
54  print('MSE: %.3f' % mse)
55  print('RMSE: %.3f' % rmse)
56  print('R2: %.3f' % r2)
57
58  return [mae, mse, rmse, r2, 0]
```

Kod 9: LSTM metodu (Keskin, 2021)

KAYNAKLAR

Ali, M. S., Miah, M. S., Haque, J., Rahman, M. M., & Islam, M. K. (2021). An enhanced technique of skin cancer classification using deep convolutional neural network with transfer learning models. *Machine Learning with Applications*, *5*, 100036.

Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, *46*(3), 175–185.

Asaad, M. N., Eryürük, Ş., & Eryürük, K. (2022). Forecasting of streamflow and comparison of artificial intelligence methods: A case study for meram stream in konya, turkey. *Sustainability*, *14*(10), 6319.

Aslı, B. (2021). Prediction of market-clearing price using neural networks based methods and boosting algorithms. *International Advanced Researches and Engineering Journal*, *5*(2), 240–246.

Aydın, C. (2018). Makine öğrenmesi algoritmaları kullanılarak itfaiye istasyonu ihtiyacının sınıflandırılması. *Avrupa Bilim ve Teknoloji Dergisi*(14), 169–175.

Babacan, H. T., & Fatih, S. (2022). Makine öğrenmesi ile aksu deresi'nde akış tahmin modeli geliştirilmesi. *Türk Hidrolik Dergisi*, *6*(1), 1–11.

Bakış, R., & Göncü, S. (2015). Akarsu debi Ölçümlerinde eksik verilerin tamamlanması: Zap suyu havzası örneği.

Bilgin, M. (2017). Gerçek veri setlerinde klasik makine öğrenmesi yöntemlerinin performans analizi. *Breast*, *2*(9), 683.

Bouckaert, R. R. (1994). Choosing between two methods of nearest-neighbour regression. *International Journal of Remote Sensing*, *15*(3), 609–613.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees*. CRC press.

Caruana, R., & Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on machine learning* (pp. 161–168).

Chakrabarti, S., Cox, E., Frank, E., Güting, R. H., Han, J., Jiang, X., ... others (2008). *Data mining: know it all*. Morgan Kaufmann.

Chao, W.-L. (2011). Machine learning tutorial. *Digital Image and Signal Processing*.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794.

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., & Cho, H. (2018). Xgboost: A scalable and flexible gradient boosting library. *arXiv preprint arXiv 1803.05743*.

Çolakoğlu, A. A. (2020). *Makine öğrenmesi algoritmaları ile avrupa havalimanları analizi* (Unpublished master's thesis). Pamukkale Üniversitesi Sosyal Bilimleri Enstitüsü.

Coşar, M., & Deniz, E. (2021). Makine öğrenimi algoritmaları kullanarak kalp hastalıklarının tespit edilmesi. *Avrupa Bilim ve Teknoloji Dergisi*(28), 1112–1116.

Çubukçu, E. A., Demir, V., & Sevimli, M. F. (2022). Akim verilerinin makine öğrenmesi teknikleriyle tahmin edilmesi. *Gazi Mühendislik Bilimleri Dergisi*, *8*(2), 257–272.

De'ath, G., & Fabricius, K. E. (2000). Classification and regression trees: a powerful yet simple technique for ecological data analysis. *Ecology*, *81*(11), 3178–3192.

Demirpençe, H. (2015). Köprüçay akimlarinin yapay sinir ağlari ile tahmini.

Dogan, A., & Birant, D. (2021). Machine learning and data mining in manufacturing. *Expert Systems with Applications*, *166*, 114060.

Dursun, F., & Karabatak, M. (n.d.). The estimation of missing flow records by correlation and neural networks in firat basin. *Engineering Sciences*, *4*(1), 30–40.

Ehrenfeucht, A., Haussler, D., Kearns, M., & Valiant, L. (1989). A general lower bound on the number of examples needed for learning. *Information and Computation*, *82*(3), 247–261.

Eren, B., & Aksangür, İ. (2019). Çevresel veri problemleri için veri madenciliği ile veri ön işleme. *Academic Perspective Procedia*, *2*(3), 1349–1356.

Fırat, A. (2019). *Yapay sinir ağları ile ortalama debi ve maksimum yağış tahmini istanbul göksu dere örneği* (Unpublished master's thesis). Sakarya Uygulamalı Bilimler Üniversitesi.

Freund, Y., & Schapire, R. E. (1996). Game theory, on-line prediction and boosting. In *Proceedings of the ninth annual conference on computational learning theory* (pp. 325–332).

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 1189–1232.

Gemici, E., Ardıçlıoğlu, M., & Kocabaş, F. (2013). Akarsularda debinin yapay zekâ yöntemleri ile modellenmesi. *Erciyes Üniversitesi Fen Bilimleri Enstitüsü Fen Bilimleri Dergisi*, *29*(2), 135 - 143.

Ghorbani, M. A., Deo, R. C., Kim, S., Hasanpour Kashani, M., Karimi, V., & Izadkhah, M. (2020). Development and evaluation of the cascade correlation neural network and the random forest models for river stage and river flow prediction in australia. *Soft Computing*, *24*(16), 12079–12090.

Ghorbani, M. A., Zadeh, H. A., Isazadeh, M., & Terzi, O. (2016). A comparative study of artificial neural network (mlp, rbf) and support vector machine models for river flow prediction. *Environmental Earth Sciences*, *75*(6), 1–14.

Gökçe, H., Sönmez, E. F., Selen, A., & Aladağ, Z. (2022). Uygun normalizasyon tekniği ve yapay sinir ağlari analizi ile otomobil satiş tahminlemesi. *İşletme Ekonomi ve Yönetim Araştırmaları Dergisi*, *5*(1), 19–45.

Graves, A., Fern'andez, S., & Gomez, F. (2005). Framewise phoneme classification with bidirectional lstm and other neural network architectures. In *Neural information processing* (pp. 552–561).

Gündüz, G., & Akkaya, A. (2019). Derin öğrenme yöntemleri ile zaman serisi tahmin problemlerinde lstm modeli kullanımı. *Akademik Platform Mühendislik ve Fen Bilimleri Dergisi*, *7*(1), 20–25.

GuolinKe, Q. M., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Adv. Neural Inf. Process. Syst*, *30*, 52.

Hastie, T., Tibshirani, R., Friedman, J. H., & Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction* (Vol. 2). Springer.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Hothorn, T., Hornik, K., & Zeileis, A. (2006). Unbiased recursive partitioning: A conditional inference framework. *Journal of computational and graphical statistics*, *15*(3), 651–674.

İlya, K., Keser, S. B., & Yolaçan, E. (2021). Saldırı tespit sistemlerinde topluluk öğrenme yöntemlerinin kıyaslanması. *Avrupa Bilim ve Teknoloji Dergisi*(31), 725–734.

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of artificial intelligence research*, *4*, 237–285.

Kaleli, P., & Kurtuluş, B. (2021). Makine Öğrenmesi algoritmaları: Gradient boosting. *Bilim ve Teknik Dergisi*, *0*(189), 36–42.

Kayri, M., & Boysan, M. (2008). Bilişsel yatkınlık ile depresyon düzeyleri ilişkisinin

sınıflandırma ve regresyon ağacı analizi ile incelenmesi. *Hacettepe Üniversitesi Eğitim Fakültesi Dergisi*, *34*(34), 168–177.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., & Ye, Q. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, *30*, 3149–3157.

Keskin, V. (2021). *(50 saat) python a-z™: Veri bilimi ve machine learning.* Retrieved 2023-05-05, from `https://www.udemy.com/course/python-egitimi/`

Koçak, E. A. (2019). *Yarı kurak bölgelerde çevresel akış belirlenmesinde qb (minimum akış) ve ysa (yapay sinir ağları) yöntemlerinin kullanılması* (Unpublished master's thesis). Konya Teknik Üniversitesi.

Kuhn, M., & Johnson, K. (2013). *Applied predictive modeling*. Springer Science & Business Media.

Kurt, A. (2021). *Ağ tabanlı saldırı tespit sistemlerinde topluluk öğrenme yöntemlerinin karşılaştırmalı performans analizi* (Unpublished master's thesis). Sakarya Üniversitesi.

Köklü, M., Karaca, M., & Güneş, A. (2016). K-en yakın komşu regresyon (knn) yöntemi ve uygulamaları. *Anadolu Üniversitesi Bilim ve Teknoloji Dergisi A - Uygulamalı Bilimler ve Mühendislik*, *17*(2), 269–283.

Li, J., & Zou, Y. (2018). Catboost: unbiased boosting with categorical features. In *Proceedings of the 32nd international conference on neural information processing systems (nips 2018).* Montreal, Canada.

Loh, W.-Y. (2011). Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *1*(1), 14–23.

Loshchilov, I., & Ihler, A. (2017). Catboost: unbiased boosting with categorical features. In *Advances in neural information processing systems* (pp. 6638–6648).

Mason, L., Baxter, J., Bartlett, P. L., & Frean, M. (1999). Boosting algorithms as gradient descent. *Advances in neural information processing systems*, 512–518.

Musarat, M. A., Alaloul, W. S., Rabbani, M. B. A., Ali, M., Altaf, M., Fediuk, R., ... others (2021). Kabul river flow prediction using automated arima forecasting: A machine learning approach. *Sustainability*, *13*(19), 10720.

Nacar, S., Kankal, M., & Hınıs, M. A. (2018). Çok değişkenli uyarlanabilir regresyon eğrileri (çdure) ile günlük akarsu akımlarının tahmini-haldizen deresi örneği. *Gümüşhane Üniversitesi Fen Bilimleri Dergisi*, *8*(1), 38–47.

Oğuzlar, A. (2003). Veri ön işleme. *Erciyes Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi*(21).

Öncül, M. (2008). *Aşağı sakarya havzasındaki küçük akarsuların yapay sinir ağları yöntemiyle akım süreklilik eğrilerinin elde edilerek enerji potansiyellerinin tespiti* (Unpublished master's thesis). Sakarya Üniversitesi.

Oral, M., Okatan, E., & Kırbaş, İ. (2021). Makine öğrenme yöntemleri kullanarak konut fiyat tahmini üzerine bir çalışma: Madrid örneği. *Uluslararası Genç Araştırmacılar Öğrenci Kongresi, Burdur, Turkey*.

Öztürk, A., Durak, Ü., & Badıllı, F. (2020). Twitter verilerinden doğal dil işleme ve makine öğrenmesi ile hastalik tespiti. *Konya Mühendislik Bilimleri Dergisi*, *8*(4), 839–852.

Öztürk, M., & Demirtas, E. (2019). Gradient boosting machine (gbm) algoritması ve uygulamaları. In *International conference on mathematics and mathematics education* (pp. 47–56).

Pramanik, N., & Panda, R. K. (2009). Application of neural network and adaptive neuro-fuzzy inference systems for river flow prediction. *Hydrological sciences journal*, *54*(2), 247–260.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A., & Gulin, A. (2018). Catboost: unbiased boosting with categorical features. *NeurIPS*.

Şadi, Ş. (2013). İş zekası ve veri madenciliği.

Sağbaş, E. A., & Ballı, S. (2016). Akıllı telefon sensör verileri ile eylem tanımada

lojistik regresyon ve knn yöntemlerinin karşılaştırılması. In *1st international conference on engineering technology and applied science* (Vol. 21, pp. 894–899).

Sarı, , & Tunalı, Y. (2020). Zaman serisi verilerinin tahmininde derin iğrenme yöntemleri ve lstm modeli. *Ege Akademik Bakıš*, *20*(2), 759–769.

Şengül, Z. (2022). *Makine öğrenmesi algoritmalarını kullanarak bitcoin fiyat tahmini* (Unpublished master's thesis). Trakya Üniversitesi Sosyal Bilimler Enstitüsü.

Sevinç, A., & Buket, K. (2021). Derin öğrenme ve istatistiksel modelleme yöntemiyle sıcaklık tahmini ve karşılaştırılması. *Avrupa Bilim ve Teknoloji Dergisi*(28), 1222–1228.

Sharma, A., Kumar, P., & Patni, P. (2017). A review on deep learning techniques for the diagnosis of alzheimer's disease. *Journal of medical systems*, *41*(8), 129.

Teni, M. H. M., Ariffin, A. A. M., Ahmad, W. A. W., & Masood, I. (2020). Pattern recognition for manufacturing process variation using ensembled artificial neural network. *The Issues In Technologies: Application and Development*.

Toluk, T. (2006). *Akarsu akımlarının yapay sinir ağı metotları kullanılarak modellenmesi* (Unpublished master's thesis). Sakarya Üniversitesi.

Tosunoğlu, F., İspirli, M. N., Gürbüz, F., & Şengül, S. (2017). Fırat havzası'ndaki eksik akım verilerinin debi süreklilik çizgileri ve regresyon modelleri ile tahmin edilmesi. *Journal of the Institute of Science and Technology*, *7*(4), 85–94.

Turan, M. E., & Yurdusev, M. A. (2009). River flow estimation from upstream flow records by artificial intelligence methods. *Journal of Hydrology*, *369*(1-2), 71–77.

Turhan, O. (2019). Zaman serilerinde lstm ile öngörü. *Uluslararası İktisadi ve İdari İncelemeler Dergisi*, *1*(1), 16–26.

Tüzemen, A., & Yıldız, Ç. (2018). Holt-winters tahminleme yöntemlerinin karşilaştirmali analizi: Türkiye işsizlik oranlari uygulamasi. *Atatürk Üniversitesi*

İktisadi ve İdari Bilimler Dergisi, *32*(1), 1–18.

Üneş, F., Demirci, M., Zelenakova, M., Çalışıcı, M., Taşar, B., Vranay, F., & Kaya, Y. Z. (2020). River flow estimation using artificial intelligence and fuzzy techniques. *Water*, *12*(9), 2427.

Unver, Y., & Yildiz, I. (2019). Yüksek boyutlu veri kümesinde extreme gradient boosting (xgboost) modeli ile sınıflandırma. In *3rd international symposium on multidisciplinary studies and innovative technologies* (pp. 191–198).

Xie, Z., Xu, X., & Wang, Y. (2018). A novel lstm-based model for short-term traffic forecasting. *IEEE Transactions on Intelligent Transportation Systems*, *20*(3), 1037–1050.

Xu, W., Jiang, Y., Zhang, X., Li, Y., Zhang, R., & Fu, G. (2020). Using long short-term memory networks for river flow prediction. *Hydrology Research*, *51*(6), 1358–1376.

Yavuz, A., & Çilengiroğlu, Ö. V. (2020). Lojistik regresyon ve cart yöntemlerinin tahmin edici performanslarının yaşam memnuniyeti verileri için karşılaştırılması. *Avrupa Bilim ve Teknoloji Dergisi*(18), 719–727.

Yavuz, Ö. (2021). A leading indicator approach with data mining techniques in analysing bitcoin market value. *Avrupa Bilim ve Teknoloji Dergisi*(32), 20–26.

Yildirim, M. (2021). Gradient boosting machine: Teorik bir İnceleme. *Balıkesir Üniversitesi Fen Bilimleri Enstitüsü Dergisi*, *23*(1), 16–25.

Yürek, Ö., Birant, D., & Yürek, I. (2021). Wind power generation prediction using machine learning algorithms. *Dokuz Eylül Üniversitesi Mühendislik Fakültesi Fen ve Mühendislik Dergisi*, *23*(67), 107–119.

Zaiane, O. (1999). Principle of knowledge discovery in databases, university of alberta. *Department of Computer Science. CMPUT690*.

Zhou, Z.-H. (2003). *Three perspectives of data mining.*

Özsu, M. F., & Koç, E. (2019). Karar ağaçları ve xgboost: Yapay zeka uygulamaları

için Öğrenme teknikleri. In *International symposium on innovative approaches in scientific studies* (pp. 539–548).

ÖZGEÇMİŞ

Kişisel Bilgiler

Adı Soyadı : Nazlı Nida GÜLERYÜZ

Eğitim

Derece	Eğitim Birimi	Mezuniyet Tarihi
Yüksek Lisans	: KTO Karatay Üniversitesi	Mart-2023
Lisans	: KTO Karatay Üniversitesi	Şubat-2019
Lise	: Selçuklu Anadolu Lisesi	Haziran-2012

İş Deneyimi

Yıl	Yer	Görev
2015	Meram Elektrik Dağıtım A.Ş.	Stajyer Mühendis
2016	DSİ 4.Bölge Müdürlüğü	Stajyer Mühendis
2021	Anadolu Birlik Holding A.Ş.	Tedarik Zinciri Koordinatörlüğü