İSTANBUL GELİŞİM ÜNİVERSİTESİ
MÜHENDİSLİK VE MİMARLIK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

YAPAY ZEKA TABANLI SU KAYNAKLARI ENTEGRE YÖNETİMİ

LİSANS BİTİRME PROJESİ

Hazırlayan
Ayşe Nur KORKMAZ
Elif SAKAL
Doğukan SÜRÜCÜ

Danışman
Doç. Dr. ELHAM PASHAEI

İstanbul – 2024

TEZ TANITIM FORMU

Yazar Adı Soyadı : Elif SAKAL, Ayşe Nur KORKMAZ, Doğukan SÜRÜCÜ

Tezin Dili : Türkçe

Tezin Adı : Yapay Zeka Tabanlı Su Kaynakları Entegre Yönetimi

Fakülte : İstanbul Gelişim Üniversitesi Mühendislik ve Mimarlık Fakültesi

Bölümü : BİLGİSAYAR MÜHENDİSLİĞİ

Tezin Türü : Lisans

Tezin Tarihi : 07.06.2024

Sayfa Sayısı : 108

Tez Danışmanları : 1. Doç.Dr. ELHAM PASHAEI

Ayşe Nur KORKMAZ
Elif SAKAL
Doğukan SÜRÜCÜ

T. C.
İSTANBUL GELİŞİM ÜNİVERSİTESİ
MÜHENDİSLİK ve MİMARLIK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

YAPAY ZEKA TABANLI SU KAYNAKLARI ENTEGRE YÖNETİMİ

Ayşe Nur KORKMAZ
Elif SAKAL
Doğukan SÜRÜCÜ

Danışman
Doç. Dr. Elham PASHAEI

İstanbul – 2024

BEYAN

Bu tezin hazırlanmasında bilimsel ahlak kurallarına uyulduğu, başkalarının eserlerinden yararlanılması durumunda bilimsel normlara uygun olarak atıfta bulunulduğu, kullanılan verilerde herhangi tahrifat yapılmadığını, tezin herhangi bir kısmının bu üniversite veya başka bir üniversitedeki başka bir tez olarak sunulmadığını beyan ederim.

Ayşe Nur KORKMAZ
Elif SAKAL
Doğukan SÜRÜCÜ
07.06.2024

İstanbul Gelişim Üniversitesi, Mühendislik ve Mimarlık Fakültesi Bilgisayar Mühendisliği Bölümü 200403623 numaralı Lisans Öğrencisi ELİF SAKAL, gerekli tüm şartları yerine getirdikten sonra hazırladığı "YAPAY ZEKA TABANLI SU KAYNAKLARI ENTEGRE YÖNETİMİ" başlıklı projesini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

İstanbul Gelişim Üniversitesi, Mühendislik ve Mimarlık Fakültesi Bilgisayar Mühendisliği Bölümü 200403599 numaralı Lisans Öğrencisi AYŞE NUR KORKMAZ, gerekli tüm şartları yerine getirdikten sonra hazırladığı "YAPAY ZEKA TABANLI SU KAYNAKLARI ENTEGRE YÖNETİMİ" başlıklı projesini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

İstanbul Gelişim Üniversitesi, Mühendislik ve Mimarlık Fakültesi Bilgisayar Mühendisliği Bölümü 200403617 numaralı Lisans Öğrencisi DOĞUKAN SÜRÜCÜ, gerekli tüm şartları yerine getirdikten sonra hazırladığı "YAPAY ZEKA TABANLI SU KAYNAKLARI ENTEGRE YÖNETİMİ" başlıklı projesini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Jüri Üyeleri		**İmza**
Tez Danışmanı :	**Doç. Dr Elham PASHAEI** (Kurumu) İGÜ Mühendislik ve Mimarlık Fakültesi Bilgisayar Mühendisliği	……..
Jüri Üyesi :	**Ünvanı Adı SOYADI** (Kurumu) İGÜ Mühendislik ve Mimarlık Fakültesi Bilgisayar Mühendisliği	……..
Jüri Üyesi :	**Ünvanı Adı SOYADI** (Kurumu) İGÜ Mühendislik ve Mimarlık Fakültesi Bilgisayar Mühendisliği	……..

Teslim Tarihi : 07.06.2024

Sunum Tarihi : 26.06.2024

ÖZET

Su kaynaklarının etkin yönetimi, giderek artan nüfus ve iklim değişikliğinin yarattığı baskılar nedeniyle kritik bir önem taşımaktadır. Bu bağlamda, doğru su talebi tahminleri, su kaynaklarının sürdürülebilir yönetimini sağlamak ve gelecekteki su ihtiyaçlarını karşılamak için büyük önem arz etmektedir. Tez çalışmam, su talebi tahminlerinin doğruluğunu artırmak amacıyla çeşitli makine öğrenimi modellerinin performansını değerlendirmekte ve karşılaştırmaktadır. Çalışmada, literatürde yer alan ve su talebi tahminleri için kullanılan farklı makine öğrenimi yaklaşımları incelenmiştir. Bunlar arasında, zaman serisi tahmininde öne çıkan ARIMA ve LSTM modelleri ile sınıflandırma ve regresyon problemlerinde sık kullanılan Random Forest ve XGBoost gibi algoritmalar bulunmaktadır. Bu modellerin performansı, mevcut araştırmalarla karşılaştırılarak ve çeşitli metriklerle (MSE, RMSE, MAE, MAPE) değerlendirilmiştir. Araştırma, özellikle LSTM modelinin, su talebi tahminlerinde zamansal bağımlılıkları etkili bir şekilde yakalayarak yüksek tahmin doğruluğu sağladığını ortaya koymuştur.

Bu model, su tüketimi verilerindeki desenleri ve zamanla değişen trendleri anlamada büyük başarı göstermiştir. XGBoost ve Random Forest gibi diğer modeller de farklı senaryolar ve veri kümeleri üzerinde etkili tahminler yapabilmişlerdir. Çalışmanın pratik bir uygulaması olarak, bu modeller Python programlama dili kullanılarak bir web uygulamasına entegre edilmiştir. Bu uygulama, kullanıcılara belirli tarihler için geçmiş ve gelecek su talebi tahminleri yapma imkânı sunmaktadır. Uygulama, kullanıcı dostu bir arayüz üzerinden erişilebilir olup, su yönetimi yetkilileri ve politika yapıcılar için değerli bir kaynak olabilir. Sonuç olarak, tez çalışması, makine öğrenimi modellerinin su talebi tahminlerindeki potansiyelini ve bu tekniklerin su kaynakları yönetimi üzerindeki etkilerini derinlemesine incelenmiştir. Bu tekniklerin daha da geliştirilmesi, su kaynakları yönetimi stratejilerinin daha etkin ve verimli hale getirilmesine olanak tanıyacak ve gelecek nesillere yönelik su güvenliğini artıracaktır.

Anahtar Kelimeler: Makine öğrenmesi, su talebi, yapay zeka, LSTM

SUMMARY

The effective management of water resources is critically important due to the pressures created by an increasing population and climate change. In this context, accurate water demand forecasts are essential for the sustainable management of water resources and meeting future water needs. This thesis evaluates and compares the performance of various machine learning models to enhance the accuracy of water demand forecasts. The study examines different machine learning approaches used for water demand forecasting that are documented in the literature. Among these, time series forecasting models like ARIMA and LSTM stand out, as well as classification and regression models frequently used such as Random Forest and XGBoost. The performance of these models has been assessed by comparing them with existing research and using various metrics (MSE, RMSE, MAE, MAPE).metrics

The research particularly highlights that the LSTM model effectively captures temporal dependencies in water demand forecasts, demonstrating high accuracy. This model has successfully interpreted patterns and evolving trends in water consumption data. Other models like XGBoost and Random Forest have also performed effectively across different scenarios and datasets. As a practical application of the study, these models have been integrated into a web application developed using the Python programming language. This application provides users with the ability to make past and future water demand forecasts for specific dates. Accessible through a user-friendly interface, the application serves as a valuable resource for water management officials and policy makers.

In conclusion, the thesis deeply investigates the potential of machine learning models in water demand forecasting and their impact on the management of water resources. The further development of these techniques will enable more effective and efficient water resource management strategies, enhancing water security for future generations.

Keywords: Machine learning, water demand, artificle intelligence, LSTM

İÇİNDEKİLER

DÖRDÜNCÜ BÖLÜM
MODEL EĞİTİMİ VE DEĞERLENDİRME

KISALTMALAR

MSE : Mean Squared Error

RMSE : Root Mean Squared Error

MAE : Mean Absolute Error

MAPE : Mean Absolute Percentage Error

R^2 : R-squared

CE : Cross Entropy

LSTM : Long Short-Term Memory

MACLA-LSTM : Multi-Attention Cross-Entropy-based LSTM

MCO-IWDF : Moving Calibration Optimization with Incremental Weighted Direct Feedback

RF : Random Forest

NN : Neural Network

NSGAII-FORAGM : Non-dominated Sorting Genetic Algorithm II for Reverse Analysis in Genetic Modification

GCRNN : Graph Convolutional Recurrent Neural Network

ANN : Artificial Neural Network

CNN-GRU : Convolutional Neural Network- Gated Recurrent Unit

WRCC : Weighted Random Correction with Cumulative Terms Regression

GBDT : Gradient Boosted Decision Trees

ZSG-BASED : Zero-Shot Learning Based Adaptive Spectral Clustering

NSE : Normalized Squared Error

SVR : Support Vector Regression

GRU : Gated Recurrent Unit

IoT : Internet of Things

AR : Autoregression

MA : Moving Average

KNN : K-Nearest Neighbors

RNN : Recurrent Neural Network

SARIMAX	**:**	Seasonal Autoregressive Integrated Moving Average with Exogenous Factors
ES	**:**	Exponential Smoothing
GOSS	**:**	Gradient-based One-Side Sampling
EFB	**:**	Interactive Feedback
API	**:**	Application Programming Interface
ASCII	**:**	American Standard Code for Information Interchange
URL	**:**	Uniform Resource Locator

TABLOLAR LİSTESİ

ŞEKİLLER LİSTESİ

ÖNSÖZ

İstanbul Gelişim Üniversitesi Mühendislik ve Mimarlık Fakültesi Bilgisayar Mühendisliği Bölümü'nde yürüttüğümüz lisans bitirme projemiz "Yapay Zekâ Tabanlı Su Kaynakları Entegre Yönetimi" üzerine çalışmalar yapma fırsatı bulduk. Bu çalışma, su kaynaklarının yönetiminde yapay zekâ ve makine öğrenimi tekniklerinin potansiyelini keşfetmek üzere tasarlanmıştır

Projenin amacı, İstanbul'un su kaynaklarının sürdürülebilir yönetimi için gelişmiş tahmin modelleri ve analitik çözümler sunmaktır. Bu süreçte, su tüketim desenlerini analiz ederek ve gelecekteki su taleplerini tahmin ederek kentimizin kaynaklarını daha verimli kullanmayı hedefledik. Çalışmalarımız, su yönetimi alanında bilgi birikimimi artırmanın yanı sıra, karşılaştığım zorluklar ve çözüm süreçleri hakkında derinlemesine tecrübe edinmemize olanak sağladı.

Bu çalışmanın, su kaynakları yönetimi konusunda yeni yaklaşımlar sunması ve bu alanda yapılacak diğer çalışmalara ilham vermesi en büyük temennimdir.

TEŞEKKÜR

Danışmanımız Doç. Dr. Elham Pashaei'nin değerli rehberliği, projenin her aşamasında bilimsel düşünme ve problem çözme becerilerimizi geliştirmemizde büyük rol oynadı. Kendisine ve projede emeği geçen tüm hocalarımıza teşekkürü bir borç biliriz.

GİRİŞ

Dünya genelinde hızla artan nüfus büyümesi ve iklim değişikliğinin yarattığı etkiler, su kaynaklarının yönetimini zorunlu kılan kritik faktörler arasında yer almaktadır. Sürdürülebilir kalkınmanın sağlanması ve gelecek nesillere yeterli su kaynaklarının bırakılabilmesi için, bu kaynakların etkin ve verimli bir şekilde yönetilmesi büyük bir önem taşımaktadır. Bu bağlamda, özellikle yüksek nüfuslu büyük şehirlerde su talebinin doğru bir şekilde tahmin edilmesi, su yönetimi stratejilerinin planlanması ve uygulanması için elzemdir. İstanbul, su kaynaklarına olan yüksek talebi ile bu şehirlerin başında gelmektedir ve bu tez, İstanbul'un su tüketim verilerini kullanarak, su talebinin daha doğru tahmin edilmesine yönelik metodolojiler geliştirmeyi amaçlamaktadır.

Yapay zekâ ve makine öğrenimi teknolojilerinin son yıllarda gösterdiği ilerleme, su kaynakları yönetimi alanında da yenilikçi çözümler sunmaktadır. Bu teknolojilerin, büyük veri kümelerinden karmaşık desenleri öğrenme ve doğru tahminlerde bulunma yeteneği, su talebi tahminlerini iyileştirme potansiyeline sahiptir. Bu tez çalışması, çeşitli makine öğrenimi modellerinin İstanbul şehri için günlük su tüketim verileri üzerindeki performansını değerlendirerek, su talebi tahminleri için en etkili modeli belirlemeye odaklanmaktadır.

Araştırma Hedefleri: Tezin temel amacı, İstanbul için su tüketim tahminlerini optimize etmeyi hedefleyen makine öğrenimi modellerini geliştirmek, test etmek ve karşılaştırmaktır. Araştırmanın özel hedefleri şunlardır:

Model Geliştirme ve Uygulama: ARIMA, LSTM, Random Forest ve XGBoost algoritmaları kullanılarak çeşitli tahmin modelleri geliştirilmesi.

Performans Analizi: Bu modellerin performansının, Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) ve Mean Absolute Percentage Error (MAPE) gibi istatistiksel metrikler kullanılarak değerlendirilmesi.

Model Karşılaştırması ve Seçimi: Elde edilen sonuçların karşılaştırılması ve İstanbul'un su talebi tahminleri için en uygun modelin seçilmesi.

Politika Önerileri ve Uygulama Stratejileri: Araştırma bulgularına dayanarak, İstanbul'un su yönetimi politikalarını destekleyecek stratejik önerilerde bulunulması.

Araştırmanın Önemi: Bu çalışma, İstanbul gibi nüfus yoğunluğu yüksek ve su kaynakları sınırlı olan bir şehir için su yönetimi stratejilerinin geliştirilmesine katkı sağlamayı hedeflemektedir. Makine öğrenimi modellerinin kullanılması, bu tür karmaşık sistemlerde karar verme süreçlerini destekleyecek veri tabanlı çözümler sunarak, yöneticilere daha bilinçli ve etkili kararlar alma imkânı verecektir. Sonuç olarak, bu tez çalışması, su kaynakları yönetimi literatürüne önemli bir katkıda bulunmayı ve teorik bulguların pratik uygulamalara dönüştürülmesini amaçlamaktadır.

BİRİNCİ BÖLÜM

LİTERATÜR TARAMASI

Su talebinin doğru tahmini, sürdürülebilir su kaynakları yönetimi için kritik önem taşır. Su yetkililerinin tahsis stratejilerini optimize etmelerine, mevcut sistemler üzerindeki baskıyı azaltmalarına ve gelecekteki ihtiyaçlara planlama yapmalarına olanak tanır. Araştırmacılar, doğru su talebi tahminleri elde etmek için çeşitli metodolojiler geliştirmiştir. Bu metodolojiler genel olarak istatistiksel ve makine öğrenimi yaklaşımlarına ayrılabilir. Bu çalışmaları incelemek için geni bir literatür taraması yapıldı.

Tablo 1. Literatür Taraması

Literatür	**Makine Öğrenmesi Algotirmaları**	**Veriler**	**Bölge**	**Bağlı Olduğu Parametreler**	**Hata Metrikleri**
Ke et al. (2023)	MACLA-LTSM	184 günlük kayıttan 72 saatlik veriler	Central China	Water demand	MAE MSE R^2
Sharma (2022)	MCO-IWDF	337 günlük kayıtlardan 15 dakikalık veriler	Saudia Arabia	Water demand amount, heat	MAE MPAE RMSE
Liu et al. (2023)	Random Forest (RF), Neural Networks (NN)	10 ve 30 haftalık kayıttan 120 dakikalık veriler	China	p-value, ADF value, Peak(m3), Trough(m3), users	RMSE MAPE R^2
Li et al. (2021)	NSGAII-FORAGM	2 yıllık kayıt	Yulin, China	Water demand sequence, Water demand sections, Mean value (100 million m3), Skewness coefficient, hurst index	MAE MAPE MSE R^2
Zubaidi et al. (2022)	PSO	9 yıllık kayıt	Melbourne	Water, Tmax, Tmin, Tmean, Rain, Eva, Srad,VP, Rhmax,FAO56	MSE MARE CE

Fiorillo et al. (2021)	Random Forest	364 günlük kayıttan 30 dakikalık veriler	Naples, Italy	Climate change, State of employment, educational level	RMSE R^2
Kavya et al. (2023)	LSTM	365 günlük kayıt	Hubli, India	Water demands	RMSE MAE MAPE R^2
Zanfei et al. (2022)	GCRNN	7 yıllık kayıttan 3 milyonluk veriler	North of Italy	Water demands	MSE R^2 MAE
Shirkoohi et al. (2021)	Artificial Neural Networks (ANN)	5 yıllık kayıttan 15 dakikalık veriler	Quebec, Canada	Total water provided	RMSE MAPE Nash-Sutcliffe efficiency (E)
Deng et al. (2022)	CNN-GRU	335 günlük kayıt	3.67 square kilometers area	Water demand, the weather and temperature data	MAE MAPE RMSE
Nasser et al. (2020)	LSTM	12 aylık kayıttan 10 dakikalık veriler	Cairo	Water consumption	RMSE MAE MAPE
Guo et al. (2022)	WRCC	25 yıllık kayıt	Yellow River Basin, China	Population, industrial and agricultural water	MAPE R^2 RMSE
Li & Fu. (2023)	LSTM	20 aylık kayıt	Shanghai, China	Weather data, COVID-19 case numbers, sensor data	MSE RMSE MAE
Shuang & Zhao (2020)	GBDT	1 yıllık kayıt	Beijing–Tianjin–Hebei, North China	Population, Climate, amount of water used, public policy	MAE MSE R^2
Liu et al. (2021)	ZSG-based	5 yıllık kayıt	Shenzhen, Guangdong, China	Population, water demand	MSE NSE MAE

Pekel (2022)	Decision tree, random forest, AdaBoost	141 aylık kayıt	Istanbul, Turkey	Temperature, water demand, humidity	MAPE R^2

İstatistiksel yöntemler, ARIMA ve Grey Model varyantları gibi, trendleri ve mevsimselliği belirlemek için geçmiş su tüketimi verilerini kullanır. Birçok çalışma, bu modellerin özellikle kısa vadeli tahminler için iyi performans gösterdiğini göstermiştir. Ancak, su tüketimi ve onu etkileyen faktörler arasındaki karmaşık ilişkileri yakalamakta zorlanabilirler.

Makine öğrenimi modelleri daha veri odaklı bir yaklaşım sunar. Çalışmalar, Yapay Sinir Ağları (ANN'ler), Destek Vektör Regresyonu (SVR), Rastgele Orman (RF) ve Uzun Kısa Süreli Hafıza (LSTM) ağları gibi çeşitli teknikleri başarıyla uygulamıştır. Bu modeller doğrusal olmayan ilişkileri ele alabilir ve hava durumu verileri, sosyo-ekonomik göstergeler ve sensör ölçümleri gibi daha geniş bir değişken yelpazesini içerebilir. Özellikle, LSTM ağları, su tüketimi verilerindeki zamansal bağımlılıkları yakalama konusunda olağanüstü yetenekler göstermiştir.

Ayrıca, bazı çalışmalar istatistiksel yöntemleri makine öğrenimi teknikleriyle birleştiren hibrit yaklaşımları incelemiştir. Örneğin, bir çalışma, yüksek tahmin doğruluğu elde etmek için Ayrık Dalgacık Dönüşümü, Başlıca Bileşen Analizi ve Parçacık Sürüsü Optimizasyonunun bir kombinasyonunu başarıyla uygulamıştır.

Su talebi tahmininin önemli bir yönü, ilgili açıklayıcı değişkenlerin seçilmesidir. Birçok çalışma, hava verilerini, özellikle sıcaklık ve yağışı tahmin modellerine dahil etmenin önemini vurgulamaktadır. Nüfus artışı, ekonomik faaliyet ve su fiyatları gibi diğer faktörler de su tüketimini önemli ölçüde etkileyebilir.

Son araştırmalar, su talebi tahmini için büyük veri analitiği ve derin öğrenme tekniklerinin potansiyelini vurgulamaktadır. Çalışmalar, büyük veri kümelerini değerlendirmek ve tahmin doğruluğunu artırmak için Konvülüsyonel Sinir Ağları (CNN'ler) ve Kapılı Yineleyen Birimler (GRU'lar) kullanımını incelemiştir. Ek olarak, araştırmacılar akıllı su sayaçları verilerinin ve Nesnelerin İnterneti (IoT) teknolojilerinin su tüketim modellerine gerçek zamanlı bilgiler sağlayarak daha dinamik ve duyarlı su yönetimi stratejileri sağlayabileceğini araştırmaktadır.

Bu makaleler, su talebi tahmininde çeşitli yöntemlerin kullanılmasını ve bu yöntemlerin su kaynakları yönetimi için sağladığı önemi ele almaktadır. Salah L. Zubaidi ve ekibinin çalışması, su talebi tahmininde yeni bir yöntem olan dalgacık dönüşümü, temel bileşen analizi ve parçacık sürü optimizasyonu gibi teknikleri bir araya getirerek, geleneksel yöntemlere göre üstün tahmin doğruluğu ve işlem süresi sunmaktadır. Qing Shuang ve RuiTingZhao'nun araştırması, makine öğrenimi modellerinin su talebi tahmininde etkinliğini vurgularken, bu modellerin ekonomik, toplumsal ve kaynak kullanımı gibi faktörleri dikkate alarak gelecekteki su talebini tahmin etmede nasıl kullanılabileceğini ortaya koymaktadır. Guoxuan Liu ve ekibinin çalışması, veri kalitesinin tahmin doğruluğunu nasıl etkilediğini inceleyerek, kentsel su talebi tahmininde makine öğrenimi yaklaşımlarının potansiyelini ortaya koymaktadır. Ahmed Abdel Nasser ve Sherif E. Hussein'in araştırması ise akıllı su sayaçlarının kullanımını ve LSTM sinir ağlarının su talebi tahminindeki etkinliğini değerlendirerek, su talebi tahminindeki teknolojik ilerlemelerin önemini vurgulamaktadır.

Bu araştırmaların toplamı, su kaynakları yönetimi ve sürdürülebilir su tedariki için kritik bir rol oynamaktadır. Doğru su talebi tahmini, su stoklarının ve altyapısının etkin bir şekilde planlanmasını ve kullanılmasını sağlayarak, su kaynaklarına olan talep artışıyla başa çıkılmasına yardımcı olabilir. Bununla birlikte, bu çalışmaların bulguları, gelecekteki su kaynakları tahminlerine dayalı politika ve uygulamaların geliştirilmesine rehberlik edebilir. Özellikle, su tedarik sistemlerinin güvenilir ve verimli bir şekilde yönetilmesi için su talebi tahmininde kullanılan tekniklerin ve modellerin geliştirilmesi, su kaynaklarının sürdürülebilirliğini sağlama açısından kritik bir öneme sahiptir. Bu nedenle, bu araştırmaların sonuçları, su kaynaklarının etkin bir şekilde yönetilmesi ve gelecek nesiller için su güvenliğinin sağlanması için önemli bir katkı sağlayabilir.

Sonuç olarak, su talebi tahmini, çok sayıda umut vadeden tekniğe sahip hızla gelişen bir alandır. Makine öğrenimi modelleri, özellikle LSTM ağları ve derin öğrenme yaklaşımları, karmaşık ilişkileri yakalamak ve yüksek tahmin doğruluğu elde etmek için güçlü araçlar olarak ortaya çıkmaktadır. Sensör teknolojileri ve veri toplama yetenekleri geliştikçe, gerçek zamanlı su talebi tahmini, verimli ve

sürdürülebilir su kaynakları yönetimini sağlamak için giderek daha önemli hale gelecektir.

İKİNCİ BÖLÜM

VERİ KÜMESİ

2.1. Veri Toplama

Veri kümesinin oluşturulması aşamasında, başlangıçta güvenilir ve kapsamlı verilere erişim sağlanması kritik öneme sahipti. Bu nedenle, veri toplama süreci titizlikle planlandı ve yürütüldü. İlk adım olarak, çeşitli kaynaklardan elde edilebilecek İstanbul'un günlük su tüketim verilerini içeren bir veri havuzu oluşturuldu. Bu kaynaklar arasında kamu kuruluşlarının raporları, su dağıtım şirketlerinin veri tabanları ve çeşitli araştırma kuruluşlarının raporları bulunmaktaydı. Bunların arasından İstanbul Büyükşehir Belediyesi'nin veri tabanından İstanbul şehri için 2011 ile 2023 yılları arasında günlük su tüketim verilerinin olduğu bir veri setinin seçimi yapıldı.

Veri seti toplandıktan sonra, verilerin doğruluğu ve güvenilirliği sağlanması için bir dizi ön işleme adımı gerçekleştirildi. Bu adımlar arasında eksik veya hatalı verilerin tespiti, aykırı değerlerin ele alınması ve tutarsızlıkların giderilmesi yer alıyordu. Özellikle, tarih ve su tüketim verilerinin doğru eşleştirilmesi ve formatlarının standartlaştırılması için özen gösterildi. Bunun için veri ön işleme adımları uygulandı. Bunlar detaylı bir şekilde aktarılacaktır.

2.2. Veri Setinin Tanımı ve Özellikleri

Veri seti, tarih, İstanbul'un günlük su tüketim ve İstanbul'daki Ömerli, Darlık, Elmalı, Terkos, Büyükçekmece, Sazlıdere, Alibey, Kazandere, Papuçdere, Istırancalar barajlarındaki doluluk oranlarının verilerinden oluşmaktadır. Tarih bilgisi "gg.aa.yyyy" formatında temsil edilmektedir, İstanbul'un günlük su tüketim verileri ise "m^3/gün" biriminde ifade edilmektedir. Baraj doluluk oranları ise yüzdelik olarak ifade edilmektedir.

Toplamda, veri setinde 4714 gözlem bulunmaktadır. Tarih verileri 1.01.2011 ile 27.11.2023 arasını kapsamaktadır. Baraj doluluk oranlarının bulunduğu satırlarda birden çok boş değer bulunmaktadır. Bu nedenle yapılacak olan model eğitimlerinde özellik seçimi yapılırken baraj doluluk oranlarının bulunduğu sütunlar çıkartılmıştır.

Tarih ve su tüketim verileri, projenin temelini oluşturan ana özelliklerdir. Ancak bu çalışmada sadece tarih ve su tüketim verileri üzerinde odaklanılmıştır. Yapılan bu özellik seçimi ile veri setinden yola çıkılarak eğitilen modellerde daha tutarlı ve doğru sonuçlara ulaşılması hedeflenmiştir. Bu özelliklerin seçimi, projenin amacına uygunluğu ve gereksinimlerini dikkate alarak yapılmıştır.

2.3. Veri Setinin İncelenmesi ve Hazırlama Süreci

Veri setinin incelenmesi aşamasında, içeriğinin doğruluğu, bütünlüğü ve uygunluğu sağlanması için çeşitli analizler yapılmıştır. İlk olarak, veri setinde herhangi bir tutarsızlık, eksiklik veya anormallik olup olmadığı kontrol edilmiştir. Bu analiz sırasında, tarih ve su tüketim verileri arasındaki ilişkiler dikkatle incelenmiş ve anlamlı desenler ve eğilimler ortaya çıkarılmıştır.

Ayrıca, zaman içinde su tüketimindeki değişimlerin gözlemlenmesi ve mevsimsel veya günlük varyasyonların belirlenmesi için çeşitli istatistiksel ve grafiksel analizler yapılmıştır. Bu analizler, projenin ilerleyen aşamalarında doğru modelleme ve tahminler yapmak için önemli bir temel oluşturmuştur.

2.3.1. Veri seti ön işleme

Öncelikli olarak veri seti İstanbul Büyükşehir Belediyesi veri tabanından alındı. Bu veri seti Excel formatında olduğu için veri ön işleme adımına geçebilmek için df = pd.read_excel("rainfall-and-daily-consumption-data-on-istanbul-dams.xlsx") şeklinde modellere eklendi. İlk olarak gerekli sütunların seçilmesi için df = df[['Tarih', 'İstanbul günlük tüketim(m³/gün)']] komutu ile veri çerçevesinden sadece “Tarih” ve "İstanbul günlük tüketim(m³/gün)" sütunları seçildi. Sonrasında veri çerçevesinin indeksini belirlemek için df = df.set_index("Tarih") ile “Tarih” sütunu veri çerçevesinin indeksi olarak ayarlandı. Ardından df.index = pd.to_datetime(df.index) ile de tarih indeksleri date time formatına dönüştürüldü. Son adım olarak veri dönüşümü ve logaritma alımı yapıldı. Bu aşamada df['İstanbul günlük tüketim(m³/gün)'] = df['İstanbul günlük tüketim(m³/gün)'] // 100 satırı ile su tüketim değerleri 100 ile bölünerek küçültüldü. Bu veri setindeki büyük sayıları daha yönetilebilir hale getirmek için yapıldı. df['İstanbul günlük tüketim(m³/gün)'] = df['İstanbul günlük tüketim(m³/gün)'].astype(float) satırı ile ise veriler float veri tipine dönüştürüldü. En sonda ise df = np.log(df) komutu ile verilerin logaritması alındı. Bu,

verilerdeki büyük dalgalanmaları azaltmak ve modelin daha iyi performans göstermesini sağlamak için yapıldı.

Sonuç olarak, titiz bir veri hazırlama süreci sonucunda, projenin gereksinimlerine uygun, doğru ve güvenilir bir veri kümesi elde edildi. Bu veri kümesi, makine öğrenmesi algoritmalarının eğitimi ve modelleme sürecinde temel bir kaynak olarak kullanıldı. Veri hazırlama aşamasında, eksik değerlerin tamamlanması, tutarsızlıkların düzeltilmesi ve veri kümesinin normalleştirilmesi gibi işlemler titizlikle gerçekleştirildi. Bu sayede, algoritmaların daha doğru ve istikrarlı tahminler üretmesi için gerekli zemin hazırlanmış oldu.

ÜÇÜNCÜ BÖLÜM

MAKİNE ÖĞRENMESİ ALGORİTMALARI

3.1. AI ve Kullanılan Algoritmaların Ayrıntılı Tanımları

Yapay zekâ, su talebi tahmini gibi karmaşık ve değişken bir problemin çözümünde insanların yapamayacağı kadar büyük veri kümelerini analiz edebilir ve bu verilerden anlamlı bilgiler çıkarabilir. Bu sayede, su kaynaklarının daha etkin ve sürdürülebilir bir şekilde yönetilmesine yardımcı olabilir. Bu proje ile yapay zekâ tekniklerinin su yönetimi alanında nasıl kullanılabileceğine dair önemli bir örnek sunuldu. Bu bölümde, su talebi tahmini için kullanılan makine öğrenmesi algoritmaları kapsamlı bir şekilde incelendi.

3.1.1. ARIMA

ARIMA (AutoRegressive Integrated Moving Average), zaman serisi verilerinde trend ve mevsimsellikleri modellemek için kullanılan istatistiksel bir yöntemdir. Ekonomi, finans ve su talebi tahmini gibi alanlarda etkili olan ARIMA, üç ana bileşen kullanır: otoregresif (AR) terimler, entegre (I) terimler ve hareketli ortalama (MA) terimler. AR terimler, serinin önceki değerlerini kullanarak gelecekteki değerleri tahmin eder; MA terimler, serideki rastgele şokların etkilerini dikkate alır; I terimi ise verideki durağan olmayan yapıları dönüştürür. Bu bileşenlerin uygun kombinasyonu, modelin durağan olmayan verileri etkili bir şekilde analiz etmesini sağlar ve ARIMA'yı, özellikle su talebi gibi zaman serisi verilerinde güçlü tahminler yapma kapasitesi ile değerli kılar.

3.1.2. XGBoost

XGBoost (eXtreme Gradient Boosting), yüksek performanslı ve ölçeklenebilir bir makine öğrenimi algoritmasıdır. Karar ağaçları temelli bir ansamble yöntemi olan bu model, sınıflandırma ve regresyon problemlerinde kullanılır. Öne çıkan özellikleri arasında düzenleme (regularization) bulunur, bu da modelin aşırı uyuma karşı dirençli olmasını sağlar. XGBoost, paralel işleme yeteneği sayesinde büyük veri setlerinde hızlı sonuçlar elde eder ve eksik değerlerle otomatik olarak başa çıkabilir.

Model, su talebi gibi zaman serisi verileri üzerinde de etkili sonuçlar verir, geçmiş verilere dayanarak gelecekteki talepleri tahmin edebilir. Bu yetenekleri, XGBoost'u su yönetimi gibi kritik alanlarda değerli bir araç yapar.

3.1.3. Decision Tree

Karar Ağacı, veri setlerini sınıflandırma ve regresyon için analiz eden, görsel olarak anlaşılması kolay bir makine öğrenimi modelidir. Bu model, verileri karar noktaları (düğümler) ve sonuçlar (yapraklar) olarak ifade eden bir ağaç yapısı kurar. Karar Ağaçları, veri üzerinde az ön işleme gerektirir ve hem kategorik hem de sayısal verilerle uyumlu çalışır. Modelin sezgiselliği ve şeffaflığı, özellikle su talebi tahmini gibi uygulamalar için değerli kılar, veriye dayalı net ve anlaşılır kararlar alınmasını sağlar. Modelin basit yapısı, karar verme süreçlerinde kolayca görselleştirilebilir ve yorumlanabilir bilgiler sunar, bu da stratejik planlama ve yönetimde önemli bir avantaj sağlar.

3.1.4. KNN

K-Nearest Neighbors (KNN), sınıflandırma ve regresyon için kullanılan basit bir makine öğrenimi algoritmasıdır. Bu model, bir veri noktasını analiz ederken, veri setindeki en yakın 'k' komşusuna bakarak sınıflandırma veya değer tahmini yapar. Algoritma, komşu sayısını (k) ve uzaklık ölçütünü (genellikle Öklid uzaklığı) temel alır. KNN, veri boyutu küçük ve boyut sayısı az olduğunda etkilidir, ancak büyük veri setleri ve yüksek boyutlulukta zorluklar yaşayabilir. Bu model, su talebi tahmini gibi uygulamalarda kullanılarak, geçmiş verilerden yararlanabilir ve doğrudan tahminler sunar.

3.1.5. Lineer Regresyon

Lineer Regresyon, bağımsız değişkenler ile bir bağımlı değişken arasındaki doğrusal ilişkiyi modelleyen bir istatistiksel yöntemdir. Bu model, verilen özelliklerin fonksiyonu olarak hedef değişkenin değerini tahmin eder ve değişkenler arasındaki ilişkiyi en iyi ifade eden doğru çizgiyi bulmaya çalışır. En Küçük Kareler Yöntemi, regresyon doğrusunu belirlemek için kullanılır. Lineer regresyonun basit yapısı, çeşitli alanlarda ve özellikle zaman serisi verilerinde, örneğin su talebi tahminlerinde

kullanımına olanak tanır. Ancak, modelin doğrusal varsayımları bazı karmaşık ilişkileri yakalayamayabilir.

3.1.6. LSTM

LSTM (Long Short-Term Memory), zaman serisi verileri için tasarlanmış bir yapay sinir ağıdır. Bu model, geleneksel RNN'lerin kısa süreli bellek sınırlılıklarını aşarak, uzun süreli bağımlılıkları öğrenebilir. LSTM, hücre durumu ve kapı yapıları (giriş, çıkış ve unutma kapıları) kullanarak bilgi akışını düzenler. Bu özellikleri ile LSTM, karmaşık veri dizilerindeki desenleri etkin bir şekilde öğrenir ve özellikle dil işleme ve zaman serisi tahminlerinde, örneğin su talebi tahmini gibi alanlarda, geniş uygulama alanı bulur.

3.1.7. AR

AutoRegressive (AR) model, zaman serisi verilerindeki önceki değerlerin gelecekteki değerler üzerindeki etkilerini modelleyen bir istatistiksel yaklaşımdır. Bu model, serinin kendisinin önceki terimlerine bağlı olarak gelecekteki değerleri tahmin etmek için kullanılır. AR modeli, özellikle finans ve ekonomik zaman serilerinde kullanılarak, verilerin içsel bağımlılıklarını açığa çıkarır. Temel bileşeni, modelin derecesi olan 'p' parametresidir, bu parametre serinin kaç önceki terimine bakılacağını belirler. AR modeli, basit yapısı ve hesaplama kolaylığı nedeniyle popülerdir ve zaman serisi analizinde temel bir araç olarak kabul edilir. Bu model, özellikle su talebi tahmini gibi uygulamalarda kullanılarak, geçmiş verilere dayanarak doğru tahminler üretir.

3.1.8. SARIMAX

SARIMAX (Seasonal AutoRegressive Integrated Moving Average with eXogenous variables) modeli, ARIMA'nın geliştirilmiş bir versiyonudur ve zaman serisi verilerindeki mevsimsel dalgalanmaları ile dışsal (exogenous) değişkenlerin etkilerini analiz eder. Bu model, ekonomi ve su talebi tahmini gibi alanlarda kullanılır, mevsimsel özellikleri ve dışsal faktörleri dikkate alarak daha kapsamlı ve doğru tahminler sağlar. SARIMAX, özellikle dış etkenlerin önemli olduğu durumlar için güçlü ve esnek bir tahmin aracıdır.

3.1.9. Exponential Smoothing

Exponential Smoothing (ES), zaman serisi verilerinde geçmiş gözlemlerin ağırlıklandırılmasıyla gelecekteki değerleri tahmin etmek için kullanılan istatistiksel bir yöntemdir. Bu model, daha yakın zamanlı verilere daha fazla ağırlık vererek, geçmiş verilerin zamanla azalan bir etkisini modellemektedir. ES, özellikle stok seviye tahmini ve kısa dönemli talep tahmini gibi alanlarda kullanılır. Modelin basitliği ve uyarlanabilir yapısı, hızlı ve etkili tahminler yapmasını sağlar, bu da onu özellikle değişken talep paternleri olan durumlar için ideal kılar. ES yöntemleri arasında Basit Exponential Smoothing, Holt'un Lineer Trend Modeli ve Winter'in Mevsimsel Ayarlama Modeli gibi çeşitler bulunur, her biri farklı türdeki veri yapısına uygun çözümler sunar.

3.1.10. N-BEATS

N-BEATS, derin öğrenme tabanlı bir zaman serisi tahmin modelidir ve özellikle karmaşık ve uzun vadeli zaman serisi verilerinde etkili tahminler yapma kapasitesine sahiptir. Bu model, seri içindeki kalıpları öğrenmek için çok katmanlı bir yapı kullanır ve tahmin sürecinde özel bir blok mimarisi benimser. Her blok, serinin farklı yönlerini modelleyerek hem trend hem de mevsimsellik gibi özellikleri ayrı ayrı ele alır. N-BEATS, esnek yapısıyla öne çıkar ve modelin öğrenme süreci boyunca herhangi bir mevsimsellik, durağanlık veya diğer öncül varsayımlara ihtiyaç duymaz. Bu özellikler, N-BEATS'i özellikle ekonomik veriler, borsa endeksleri ve talep tahmini gibi alanlarda değerli bir araç yapar. Model, tahmin performansını artırmak ve daha önce keşfedilmemiş veri desenlerini ortaya çıkarmak için geniş bir uygulama potansiyeline sahiptir.

3.1.11. LightGBM

LightGBM, gradyan artırma çerçevesinde geliştirilmiş hafif bir makine öğrenimi algoritmasıdır. Bu model, yüksek verimlilik ve hız ile büyük veri kümelerinde çalışabilme yeteneği ile öne çıkar. Özellikle sınıflandırma, regresyon ve sıralama problemlerinde kullanılır. LightGBM, veri bölünmesi için Gradient-based One-Side Sampling (GOSS) ve Exclusive Feature Bundling (EFB) gibi yenilikçi teknikler kullanarak, hesaplama sürecini optimize eder ve hafızadan tasarruf sağlar. Bu teknikler, modelin daha hızlı eğitilmesini ve daha az kaynak tüketmesini sağlarken,

model doğruluğundan ödün vermez. LightGBM, finansal analizler, telekomünikasyon sektöründeki müşteri çıkış tahminleri ve perakende stok tahminleri gibi çeşitli uygulama alanlarında etkili sonuçlar sunar.

3.1.12. Support Vector Regression (SVR)

Destek Vektör Regresyonu (SVR), veri noktaları arasında en iyi uyan hiperdüzlemi bulmayı amaçlayan bir makine öğrenimi algoritmasıdır. SVR, regresyon analizinde kullanılarak, veri noktaları arasındaki marjı maksimize eder ve bu sayede yüksek boyutlu veri setlerinde etkili tahminler sağlar. Lineer olmayan ilişkileri modellemek için çekirdek yöntemleri kullanabilir. Metin ve görüntü analizi gibi karmaşık veri yapılarıyla da başarılı tahminler gerçekleştirebilir.

3.1.13. Random Forest

Random Forest, birden fazla karar ağacını birleştirerek oluşturulan bir ansamble makine öğrenimi algoritmasıdır. Bu model, her bir ağacı rastgele seçilmiş veri alt kümeleri ve özellikler üzerinde bağımsız olarak eğiterek, sınıflandırma ve regresyon görevlerinde yüksek doğruluk sağlar. Random Forest, aşırı uymayı (overfitting) önleyen ve modelin genelleştirme yeteneğini artıran özelliklere sahiptir. Özellikle büyük ve karmaşık veri setlerinde hem hızlı hem de etkili sonuçlar elde edilmesine olanak tanır. Çeşitli endüstrilerde, risk değerlendirmesi, müşteri segmentasyonu ve biyomedikal araştırmalar gibi alanlarda geniş uygulama alanı bulur.

3.1.14. CatBoost

CatBoost, kategorik verilerle etkin bir şekilde çalışmak üzere tasarlanmış bir gradyan artırma algoritmasıdır. Bu model, özellikle sınıflandırma ve regresyon görevleri için geliştirilmiştir ve geniş veri setlerinde hızlı ve doğru tahminler sağlar. CatBoost, veri önişleme gereksinimini azaltan ve modelin aşırı uyma (overfitting) riskini düşüren özel algoritmalar kullanır. Ayrıca, veri sıralaması (ordered boosting) ve simetrik ağaç yapısı gibi yenilikçi özelliklerle donatılmıştır, bu sayede hem hesaplama süresi kısaltılır hem de tahmin doğruluğu artırılır. CatBoost, finans, reklamcılık ve tıp gibi çeşitli alanlarda, kategorik ve karmaşık veri yapıları üzerinde güçlü sonuçlar üretir.

3.1.15. BiLSTM

BiLSTM, LSTM'in çift yönlü bir versiyonudur ve verileri hem ileri hem de geri yönde işleyerek daha zengin bağlamsal anlayış sağlar. Bu model, özellikle doğal dil işleme ve konuşma tanıma gibi alanlarda tercih edilir, çünkü her iki yönden bilgiyi entegre ederek daha doğru tahminler yapabilir. BiLSTM, metin sınıflandırma, duygu analizi ve karmaşık zaman serisi tahminleri gibi uygulamalarda etkilidir.

3.1.16. MLPRegressor

MLPRegressor, yapay sinir ağlarına dayanan bir regresyon modelidir ve özellikle sürekli değerlerin tahmini için kullanılır. Bu model, çok katmanlı bir perceptron yapısına sahip olup, çeşitli gizli katmanlar ve aktivasyon fonksiyonları ile donatılmıştır. MLPRegressor, büyük ve karmaşık veri setlerindeki ilişkileri modellemek için uygun bir seçimdir, çünkü yapay sinir ağlarının sağladığı esneklik ve öğrenme kapasitesi ile non-lineer ilişkileri etkili bir şekilde yakalayabilir. Model, finans, enerji tahmini ve sağlık alanlarında geniş uygulama alanlarına sahiptir ve veriler arasındaki karmaşık desenleri ve trendleri öğrenerek doğru tahminler yapabilir.

3.2. Algoritmaların Matematiksel Temelleri

- ARIMA: ARIMA modelinin matematiksel temelleri, otoregresif (AR), entegre (I) ve hareketli ortalama (MA) terimlerinden oluşan bileşenlerine dayanır. AR terimleri, önceki zaman adımlarındaki değerler arasındaki ilişkiyi modellemek için kullanılır. İntegre terim, seriyi durağan hale getirmek için fark alma işlemiyle ilgilidir. Hareketli ortalama terimleri ise rasgele şokların etkilerini yakalamak için kullanılır.

- XGBoost: Matematiksel olarak, XGBoost, gradient boosting framework'ünde birden fazla karar ağacı oluşturur ve bu ağaçları birleştirir. Model, her ağacı oluştururken hata fonksiyonunu optimize etmek için gradyan iniş yöntemini kullanır. XGBoost'un matematiksel temelleri, karar ağaçlarının derinliklerinin ve düğüm bölme kriterlerinin optimize edilmesine dayanır.

- Decision Tree: Matematiksel olarak, karar ağacı, veri kümesini bölme kriterlerine göre ağaç benzeri bir yapıya dönüştürür. Bu bölme işlemi, veri setindeki enformatif özellikleri belirlemek için yapılan bir dizi kararın sonucunda gerçekleşir.

- KNN: Matematiksel olarak, KNN, yeni bir veri noktasının tahmin edilmesi için en yakın 'k' komşularının etiketlerini dikkate alır ve bu komşuların ağırlıklı ortalamasını hesaplar.

- Lineer Regresyon: Matematiksel olarak, lineer regresyon, verilen bağımsız değişkenlerin lineer bir kombinasyonu olarak bağımlı değişkenin tahmin edilmesini sağlayan bir model formülasyonu kullanır.

- LSTM: Matematiksel olarak, LSTM, hücre durumu ve kapı yapıları (giriş, çıkış ve unutma kapıları) kullanarak bilgi akışını düzenler. Bu yapılar, uzun vadeli bağımlılıkları öğrenmek için gereklidir ve zaman serisi verilerindeki karmaşık desenleri modellemeye yardımcı olur.

- AR: Matematiksel olarak, AR modeli, önceki zaman adımlarındaki değerler arasındaki otoregresif ilişkiyi modellemek için bir dizi katsayı kullanır.

- SARIMAX: Matematiksel olarak, SARIMAX modeli, ARIMA bileşenlerine ek olarak mevsimsel bileşenler ve dışsal değişkenlerin etkilerini içerir.

- Exponential Smoothing: Matematiksel olarak, ES, gözlemlerin ağırlıklı ortalamasını hesaplamak için bir düzeltme faktörü kullanır.

- N-BEATS: Matematiksel olarak, N-BEATS, çok katmanlı bir yapı kullanarak zaman serisi verilerindeki desenleri öğrenir ve gelecekteki değerleri tahmin eder.

- LightGBM: Matematiksel olarak, LightGBM, gradyan artırma ve özel öznitelik mühendisliği tekniklerini kullanarak hızlı ve etkili tahminler sağlar.

- Support Vector Regression: Matematiksel olarak, SVR, veri noktalarının birbirinden en iyi şekilde ayrılmasını sağlayan bir hiperdüzlemi bulmak için bir optimizasyon problemi çözer.

- Random Forest: Matematiksel olarak, Random Forest, her ağacın bağımsız olarak eğitilmesi ve tahminlerin birleştirilmesi yoluyla çalışır.

- CatBoost: CatBoost, kategorik verilerle etkin bir şekilde çalışmak üzere tasarlanmış bir gradyan artırma algoritmasıdır. Matematiksel olarak, CatBoost, gradyan artırma ve kategorik değişken işleme tekniklerini birleştirerek hızlı ve doğru tahminler yapar.

- BiLSTM: BiLSTM, LSTM'in çift yönlü bir versiyonudur ve verileri hem ileri hem de geri yönde işleyerek daha zengin bağlamsal anlayış sağlar. Matematiksel olarak, BiLSTM, iki yönlü hücre yapıları ve geçmiş verilerin işlenmesi ile çalışır.

- MLPRegressor: Matematiksel olarak, MLPRegressor, çok katmanlı perceptron yapısını kullanarak giriş özelliklerinden çıkarılan bilgileri temsil eder.

3.3. Algoritmaların Uygulanması ve Parametre Ayarları

Makine öğrenimi algoritmalarının uygulanması, Python programlama dili kullanılarak gerçekleştirildi. Her algoritmanın uygulanması için özel adımlar ve parametre ayarları şu şekilde yapıldı:

Zaman serisi verileri öncelikle uygun formata dönüştürüldü. Bu işlem için, ARIMA için statsmodels ve LSTM için keras kütüphaneleri kullanıldı. Her iki algoritma da zaman serisi verilerini etkili bir şekilde işleyerek tahminlerde bulunabilmek için öğrenildi.

sklearn kütüphanesi, bu algoritmanın eğitimi, çapraz doğrulama ile parametre optimizasyonu ve test süreçleri için gerekli araçları sağladı. Algoritmanın eğitimi

sırasında, ağaç sayısı, maksimum derinlik ve bölünme kriterleri gibi parametreler optimize edilerek en iyi performans elde edilmeye çalışıldı.

Veri setinin yüksek performanslı ve ölçeklenebilir bir şekilde işlenmesi için XGBoost algoritması tercih edildi. Bu algoritmanın uygulanması, veri setinin büyüklüğüne ve karmaşıklığına göre uygun parametrelerin seçilmesini gerektirdi. XGBoost'un sınıflandırma veya regresyon görevlerindeki başarısını artırmak için ağaç sayısı, maksimum derinlik ve öğrenme oranı gibi parametreler dikkatlice ayarlandı.

Karar ağacı modelinin uygulanması, veri setinin sınıflandırma veya regresyon amacıyla analiz edilmesini içerir. Karar ağacı oluşturulurken, veri setinin yapısına ve karmaşıklığına uygun parametreler seçildi. Ağaç derinliği, bölünme kriterleri ve düğüm sayısı gibi parametreler, modelin doğruluğunu etkilemek için ayarlandı.

K-Nearest Neighbors (KNN) algoritması, sınıflandırma veya regresyon görevlerinde veri noktalarının etkileşimlerini belirlemek için kullanılır. Bu algoritmanın uygulanması sırasında, komşu sayısı (k) ve uzaklık ölçütü gibi parametrelerin seçimi önemlidir. Veri setinin yapısına ve boyutuna bağlı olarak, en uygun k değeri ve uzaklık metriği belirlendi.

Lineer regresyon modelinin uygulanması, bağımsız değişkenler ile bir bağımlı değişken arasındaki doğrusal ilişkiyi modellemeyi içerir. Bu algoritmanın uygulanması sırasında, en küçük kareler yöntemi kullanılarak regresyon doğrusunun belirlenmesi için parametreler ayarlandı.

SARIMAX modelinin uygulanması, ARIMA'nın geliştirilmiş bir versiyonu olan bu modelin zaman serisi verilerindeki mevsimsel dalgalanmaları ve dışsal değişkenlerin etkilerini analiz etmesini içerir. Modelin uygulanması sırasında, mevsimsel bileşenlerin uygun bir şekilde modellenmesi ve dışsal değişkenlerin dahil edilmesi için parametreler ayarlandı.

Exponential Smoothing yöntemlerinin uygulanması, geçmiş gözlemlerin ağırlıklı bir şekilde kullanılarak gelecekteki değerlerin tahmin edilmesini içerir. Bu algoritmaların uygulanması sırasında, farklı smoothing parametreleri ve zaman serisi özelliklerine uygun olarak parametreler ayarlandı.

DÖRDÜNCÜ BÖLÜM

MODEL EĞİTİMİ VE DEĞERLENDİRME

4.1. LSTM Modeli

4.1.1. Gerekli Kütüphanelerin İthal Edilmesi

```
[1]: import pandas as pd
     import numpy as np
     import random
     import matplotlib.pyplot as plt
     from sklearn.metrics import mean_squared_error, mean_absolute_error, mean_absolute_percentage_error
     from sklearn.preprocessing import MinMaxScaler
     import tensorflow as tf
     from tensorflow.keras.models import Sequential
     from tensorflow.keras.layers import LSTM, Dense
     from tensorflow.keras.callbacks import EarlyStopping
```

Şekil 1. LSTM için gerekli kütüphanelerin ithal edilmesi

- pandas: Veri manipülasyonu ve analizi için kullanılır. Veri çerçeveleri oluşturmak, düzenlemek ve analiz etmek için çok sayıda araç sağlar.
- numpy: Sayısal hesaplamalar için kullanılır. Diziler ve matrislerle çalışma fonksiyonları sunar.
- matplotlib.pyplot: Grafik çizim araçları sağlar. Veri görselleştirme için kullanılır.
- sklearn.metrics: Model değerlendirme metrikleri içerir. mean_squared_error, mean_absolute_error, ve mean_absolute_percentage_error metrikleri burada kullanılır.
- sklearn.preprocessing.MinMaxScaler: Verilerin normalizasyonu için kullanılır. Verileri 0 ile 1 arasında ölçeklendirir.
- tensorflow: Derin öğrenme kütüphanesi.
- tensorflow.keras: TensorFlow'un yüksek seviyeli API'si, derin öğrenme modellerinin oluşturulması ve eğitilmesi için kullanılır.
- Sequential: Keras'ta katmanları sırayla eklemek için kullanılan bir model türüdür.

- LSTM: Long Short-Term Memory katmanı, zaman serisi verilerini modellemek için kullanılır.
- Dense: Tam bağlantılı (dense) katman, her nöronun bir önceki katmandaki tüm nöronlarla bağlantılı olduğu bir katman türüdür.
- EarlyStopping: Modelin eğitim sırasında aşırı uyumunu (overfitting) önlemek için kullanılan bir callback fonksiyonu.

4.1.2. Rastgele tohumları ayarlama

```
# Rastgele tohumları ayarlama
def set_seeds(seed=42):
    np.random.seed(seed)
    tf.random.set_seed(seed)
    random.seed(seed)
```

Şekil 2. LSTM için rastgele tohumları ayarlama

Bu kod parçacığı, rastgele sayı üretimini kontrol altına almayı amaçlayan bir fonksiyon içerir. İşlevler şu şekildedir:

- np.random.seed(seed): NumPy'de rastgele sayı üretimini kontrol eder.
- tf.random.set_seed(seed): TensorFlow'da rastgele sayı üretimini kontrol eder.
- random.seed(seed): Python'un standart random modülünde rastgele sayı üretimini kontrol eder.

Bu fonksiyon, kodun tekrarlanabilirliğini sağlamak için kullanılır. Belirli bir tohum (seed) değeri verilerek, her çalıştırmada aynı rastgele sayı dizisini elde etmek amaçlanır.

4.1.3. Veriyi yükleme ve ön işleme

```
# Veriyi yükleme ve ön işleme
df = pd.read_excel("rainfall-and-daily-consumption-data-on-istanbul-dams.xlsx")

df = df[['Tarih', 'İstanbul günlük tüketim(m³/gün)']]
df = df.set_index("Tarih")
df.index = pd.to_datetime(df.index)
df['İstanbul günlük tüketim(m³/gün)'] = df['İstanbul günlük tüketim(m³/gün)'] // 100
df['İstanbul günlük tüketim(m³/gün)'] = df['İstanbul günlük tüketim(m³/gün)'].astype(float)
df = np.log(df)
```

Şekil 3. LSTM için veri yükleme ve ön işleme

- pd.read_excel: Excel dosyasından veri yükler. Burada, İstanbul barajlarına ait günlük yağış ve su tüketimi verileri içeren bir Excel dosyası yüklenir.
- df[['Tarih', 'İstanbul günlük tüketim(m³/gün)']]: Veri çerçevesinden sadece "Tarih" ve "İstanbul günlük tüketim(m³/gün)" sütunları seçilir.
- df.set_index("Tarih"): "Tarih" sütunu veri çerçevesinin indeksi olarak ayarlanır.
- pd.to_datetime(df.index): Tarih indeksleri datetime formatına dönüştürülür.
- df['İstanbul günlük tüketim(m³/gün)'] // 100: Su tüketimi değerleri 100 ile bölünerek küçültülür. Bu, büyük sayıları daha yönetilebilir hale getirmek için yapılır.
- df['İstanbul günlük tüketim(m³/gün)'].astype(float): Veriler float tipine dönüştürülür.
- np.log(df): Verilerin logaritması alınır. Bu, verilerdeki büyük dalgalanmaları azaltmak ve modelin daha iyi performans göstermesini sağlamak için yapılır.

4.1.4. Eğitim ve test veri setlerini ayırma

```
# Eğitim ve test veri setlerini ayırma
train_size = int(len(df) * 0.80)
test_size = len(df) - train_size
train, test = df[0:train_size], df[train_size:len(df)]
```

Şekil 4. LSTM için eğitim ve test veri setlerini ayırma

- train_size = int(len(df) * 0.80): Eğitim seti için veri çerçevesinin %80'i hesaplanır.
- test_size = len(df) - train_size: Test seti için geri kalan %20 veri hesaplanır.
- train, test = df[0:train_size], df[train_size:len(df)]: Eğitim ve test setleri belirlenen boyutlara göre ayrılır.

4.1.5. Özelliklerin oluşturulması

```
# Özellikleri oluşturma
def create_features(df):
    df = df.copy()
    df['dayofweek'] = df.index.dayofweek
    df['quarter'] = df.index.quarter
    df['month'] = df.index.month
    df['year'] = df.index.year
    df['dayofyear'] = df.index.dayofyear
    return df

df = create_features(df)
train = create_features(train)
test = create_features(test)

FEATURES = ['dayofweek', 'quarter', 'month', 'year', 'dayofyear']
TARGET = 'İstanbul günlük tüketim(m³/gün)'

X_train = train[FEATURES]
y_train = train[TARGET]

X_test = test[FEATURES]
y_test = test[TARGET]
```

Şekil 5. LSTM için özelliklerin oluşturulması

- df.copy(): Veri çerçevesinin bir kopyası oluşturulur. Bu, orijinal veriyi değiştirmeden işlemler yapmayı sağlar.

- df.index.dayofweek: Haftanın günü (0: Pazartesi, 6: Pazar) eklenir.
- df.index.quarter: Yılın çeyreği (1, 2, 3, 4) eklenir.
- df.index.month: Ay (1-12) eklenir.
- df.index.year: Yıl eklenir.
- df.index.dayofyear: Yılın günü (1-365) eklenir.
- return df: Yeni özellikler eklenmiş veri çerçevesi döndürülür.
- create_features(df): Veri çerçevesine yeni özellikler eklenir.
- FEATURES: Modelin eğitiminde kullanılacak özellikler listesi.
- TARGET: Modelin tahmin etmeye çalışacağı hedef değişken.
- train[FEATURES]: Eğitim setinden özelliklerin seçilmesi.
- train[TARGET]: Eğitim setinden hedef değişkenin seçilmesi.
- test[FEATURES]: Test setinden özelliklerin seçilmesi.
- test[TARGET]: Test setinden hedef değişkenin seçilmesi.

4.1.6. Verilerin Normalize Edilmesi

```
# Veriyi normalize etme
scaler_X = MinMaxScaler()
scaler_y = MinMaxScaler()

X_train = scaler_X.fit_transform(X_train)
X_test = scaler_X.transform(X_test)

y_train = scaler_y.fit_transform(y_train.values.reshape(-1, 1)).ravel()
y_test = scaler_y.transform(y_test.values.reshape(-1, 1)).ravel()
```

Şekil 6. LSTM için verilerin normalize edilmesi

- MinMaxScaler(): Verileri 0 ile 1 arasında ölçeklendirmek için kullanılan normalizasyon yöntemi.
- fit_transform: Eğitim verilerine uygulanır ve aynı zamanda dönüştürülür.
- transform: Test verilerine uygulanır (daha önce eğitim verilerine uygulanan ölçekleme kullanılarak).
- values.reshape(-1, 1): y_train ve y_test değerleri 2 boyutlu hale getirilir.

- ravel(): Tek boyutlu hale getirilir.

4.1.7. Verilerin LSTM için yeniden şekillendirilmesi

```
# LSTM için veriyi yeniden şekillendirme
X_train = np.array(X_train).reshape((X_train.shape[0], 1, X_train.shape[1]))
X_test = np.array(X_test).reshape((X_test.shape[0], 1, X_test.shape[1]))
```

Şekil 7. LSTM için veriyi yeniden şekillendirme

- np.array(X_train): Eğitim verileri numpy dizisine dönüştürülür.
- reshape((X_train.shape[0], 1, X_train.shape[1])): Veriler 3 boyutlu hale getirilir. (örnek sayısı, zaman)

4.1.8. LSTM Modelinin Oluşturulması, Modelin Eğitimi, Tahmin Yapma, Tahminleri Orijinal Ölçeğe Döndürme

```
# LSTM modelini oluşturma
model = Sequential()
model.add(LSTM(100, activation='relu', input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')

# Erken durdurma tanımlama
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)

# Modeli eğitme
history = model.fit(X_train, y_train, epochs=100, batch_size=32, validation_data=(X_test, y_test),
```

Şekil 8. LSTM için model oluşturma, erken durdurma tanımlama, modeli eğitme

- Sequential: Keras'ta katmanları sırayla eklemek için kullanılan bir model türüdür. Bu, her katmanın bir önceki katmanın çıktısını girdi olarak aldığı anlamına gelir.
- LSTM(100): 100 LSTM birimi içerir. Bu birimler, zaman serisi verilerini işlerken kullanılan belleği temsil eder.
- activation='relu': ReLU (Rectified Linear Unit) aktivasyon fonksiyonu kullanılır. Bu fonksiyon, girdiyi pozitifse girdiyi aynen geçirir, negatifse sıfır yapar.
- input_shape=(X_train.shape[1], X_train.shape[2]): Girdi verisinin şekli (zaman adımı sayısı, özellik sayısı) burada belirtilir. X_train.shape[1] zaman adımlarını, X_train.shape[2] ise özelliklerin sayısını temsil eder.

- Dense(1): Tek bir nörondan oluşan tam bağlantılı (dense) bir katman ekler. Bu nöron, LSTM katmanından gelen bilgiyi alır ve tek bir çıkış değeri üretir. Bu, tahmin edilen su tüketimi değerini temsil eder
- optimizer='adam': Adam optimizasyon algoritmasını kullanır. Adam, öğrenme oranını dinamik olarak ayarlayarak eğitim sürecini hızlandırır ve stabilize eder.
- loss='mse': Kayıp fonksiyonu olarak MSE kullanılır. MSE, tahmin edilen ve gerçek değerler arasındaki farkların karelerinin ortalamasıdır.
- monitor='val_loss': Doğrulama setindeki kayıp değeri izlenir.
- patience=10: Doğrulama setindeki kayıp değeri 10 epoch boyunca iyileşmezse eğitim durdurulur.
- restore_best_weights=True: Eğitim durdurulduğunda, en iyi model ağırlıkları geri yüklenir.
- X_train, y_train: Eğitim verileri ve hedef değişkenler.
- epochs=100: Model 100 epoch boyunca eğitilir. Bir epoch, tüm eğitim verisinin bir kez modele verilmesidir.
- batch_size=32: Eğitim verileri 32 örneklik mini partilere bölünür ve her parti ayrı ayrı işlenir.
- validation_data=(X_test, y_test): Doğrulama verileri, modelin eğitimi sırasında doğrulama için kullanılır.
- verbose=2: Eğitim süreci sırasında ayrıntılı çıktı sağlar.
- shuffle=False: Eğitim verileri karıştırılmaz. Zaman serisi verilerinde genellikle verilerin sırası korunur.
- callbacks=[early_stopping]: Erken durdurma callback'i eğitim sürecine dahil edilir.

4.1.9. Test Verileri Üzerinde Tahmin Yapma ve Tahminleri Orijinal Ölçeğe Döndürme

```
# Test verileri üzerinde tahmin yapma
y_pred = model.predict(X_test)

# Tahminleri orijinal ölçeğe döndürme
y_test_orig = scaler_y.inverse_transform(y_test.reshape(-1, 1))
y_pred_orig = scaler_y.inverse_transform(y_pred.reshape(-1, 1))
```

Şekil 9. LSTM için tahmin yapma ve tahminleri orijinal ölçeğe döndürme

- model.predict(X_test): Test verileri kullanılarak model tahminleri yapılır.
- scaler_y.inverse_transform: Normalizasyon işlemi tersine çevrilir ve veriler orijinal ölçeğe döndürülür.

4.1.10. Hata Metric Değerlerini Hesaplama

```
# MSE ve RMSE değerlerini hesaplama
mse = mean_squared_error(y_test_orig, y_pred_orig)
rmse = np.sqrt(mse)

print("Mean Squared Error (MSE): {:.4f}".format(mse))
print("Root Mean Squared Error (RMSE): {:.4f}".format(rmse))

# MAE hesaplama
mae = mean_absolute_error(y_test_orig, y_pred_orig)
print("Mean Absolute Error (MAE):", mae)

# MAPE hesaplama
mape = mean_absolute_percentage_error(y_test_orig, y_pred_orig) * 100
print("Mean Absolute Percentage Error (MAPE):", mape)
```

Şekil 10. LSTM için hata metric değerlerini hesaplama

- mean_squared_error(y_test_orig, y_pred_orig): MSE hesaplanır.
- np.sqrt(mse): MSE'nin karekökü alınarak RMSE elde edilir.
- mean_absolute_error(y_test_orig, y_pred_orig): MAE hesaplanır. MAE, tahmin edilen ve gerçek değerler arasındaki mutlak farkların ortalamasıdır.

- mean_absolute_percentage_error(y_test_orig, y_pred_orig): MAPE hesaplanır. MAPE, tahmin hatasının yüzdesini belirtir ve genellikle % cinsinden ifade edilir.

ÇIKTI:

```
118/118 - 0s - loss: 0.0076 - val_loss: 0.0075 - 249ms/epoch - 2ms/step
Epoch 25/100
118/118 - 0s - loss: 0.0075 - val_loss: 0.0081 - 248ms/epoch - 2ms/step
Epoch 26/100
118/118 - 0s - loss: 0.0074 - val_loss: 0.0081 - 250ms/epoch - 2ms/step
Epoch 27/100
118/118 - 0s - loss: 0.0072 - val_loss: 0.0085 - 250ms/epoch - 2ms/step
30/30 [==============================] - 0s 1ms/step
Mean Squared Error (MSE): 0.0018
Root Mean Squared Error (RMSE): 0.0425
Mean Absolute Error (MAE): 0.03180024550816775
Mean Absolute Percentage Error (MAPE): 0.3080031482972641
```

Şekil 11. LSTM için çıktı

4.1.11. Gerçek ve Tahmin Değerleri Birleştirme ve Görselleştirme

```
[2]: # Gerçek ve tahmin değerleri birleştirme
results = pd.DataFrame({'Tarih': test.index, 'Gerçek': y_test_orig.flatten(), 'Tahmin': y_pred_orig.flatten()})
results = results.set_index('Tarih')

# Görselleştirme
plt.figure(figsize=(14, 7))
plt.plot(results.index, results['Gerçek'], label='Gerçek Değerler', color='blue')
plt.plot(results.index, results['Tahmin'], label='Tahmin Değerleri', color='red')
plt.title('Gerçek ve Tahmin Değerlerinin Karşılaştırılması')
plt.xlabel('Tarih')
plt.ylabel('Günlük Tüketim (m³/gün)')
plt.legend()
plt.show()
```

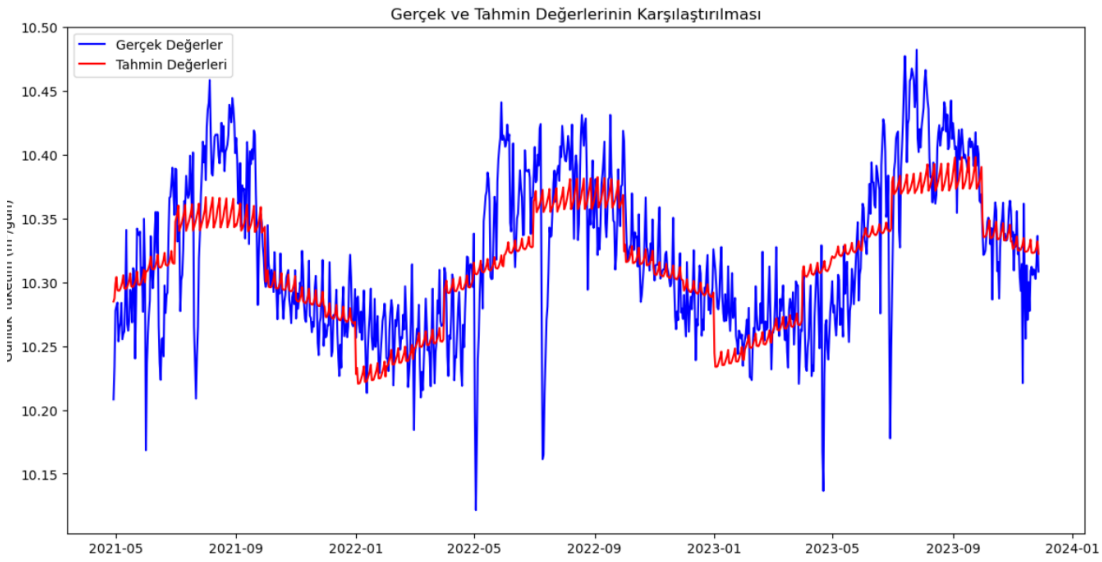
Şekil 12. LSTM için gerçek ve tahmin değerlerini birleştirme ve görselleştirme

- {'Tarih': test.index, 'Gerçek': y_test_orig.flatten(), 'Tahmin': y_pred_orig.flatten()}: DataFrame'in içeriği için bir sözlük oluşturur.
- 'Tarih': test.index: Test setinin tarihlerini içerir.
- 'Gerçek': y_test_orig.flatten(): Gerçek su tüketimi değerlerini içerir, flatten() metodu çok boyutlu diziyi tek boyutlu hale getirir.

- 'Tahmin': y_pred_orig.flatten(): Model tarafından tahmin edilen su tüketimi değerlerini içerir, flatten() metodu yine çok boyutlu diziyi tek boyutlu hale getirir.
- results.set_index('Tarih'): 'Tarih' sütununu DataFrame'in indeksi olarak ayarlar. Bu, tarihler üzerinden veri manipülasyonu ve görselleştirme yapmayı kolaylaştırır.
- plt.figure(figsize=(14, 7)): Yeni bir figür oluşturur ve boyutlarını belirler. Burada 14x7 inç boyutlarında bir grafik oluşturulur.
- plt.plot(results.index, results['Gerçek'], label='Gerçek Değerler', color='blue'): Gerçek su tüketimi değerlerini çizer.
- results.index: X ekseninde tarihleri kullanır.
- results['Gerçek']: Y ekseninde gerçek su tüketimi değerlerini kullanır.
- label='Gerçek Değerler': Grafikte bu çizginin etiketini belirler.
- color='blue': Çizginin rengini mavi yapar.
- plt.plot(results.index, results['Tahmin'], label='Tahmin Değerleri', color='red'): Tahmin edilen su tüketimi değerlerini çizer.
- results.index: X ekseninde tarihleri kullanır.
- results['Tahmin']: Y ekseninde tahmin edilen su tüketimi değerlerini kullanır.
- label='Tahmin Değerleri': Grafikte bu çizginin etiketini belirler.
- color='red': Çizginin rengini kırmızı yapar.
- plt.title('Gerçek ve Tahmin Değerlerinin Karşılaştırılması'): Grafiğe başlık ekler.
- plt.xlabel('Tarih'): X eksenine 'Tarih' etiketi ekler.
- plt.ylabel('Günlük Tüketim (m^3/gün)'): Y eksenine 'Günlük Tüketim (m^3/gün)' etiketi ekler.

- plt.legend(): Grafikteki çizgilerin neyi temsil ettiğini gösteren bir efsane (legend) ekler. Bu efsane, label parametresi ile belirtilen etiketleri kullanır.
- plt.show(): Grafiği ekranda gösterir. Bu komut olmadan grafik oluşturulsa bile görüntülenmez.

ÇIKTI:



Şekil 13. LSTM için görselleştirme çıktısı

4.1.12. Gelecek 10 günü Tahmin Etme ve Görselleştirilmesi

```
# Son 10 günlük veriyi alarak tahmin yapma
last_10_days = df[-10:]
last_10_days = create_features(last_10_days)

X_future = last_10_days[FEATURES]

# Veriyi normalize etme
X_future_normalized = scaler_X.transform(X_future)

# Veriyi LSTM modeli için uygun formata dönüştürme
X_future_formatted = np.array(X_future_normalized).reshape((X_future_normalized.shape[0], 1, X_future_normalized.shape[

# Gelecek 10 gün için tahmin yapma
future_predictions = model.predict(X_future_formatted)

# Tahminleri orijinal ölçeğe döndürme
future_predictions_orig = scaler_y.inverse_transform(future_predictions.reshape(-1, 1))
```

Şekil 14. LSTM için gelecek 10 günü tahmin etme ve görselleştirme

```
]: from datetime import datetime, timedelta

# Son tahmin edilen tarihi alınması
last_date = df.index[-1]

# Gelecek 10 günün tarihlerini oluşturma
future_dates = [last_date + timedelta(days=i) for i in range(1, 11)]

# Tahmin edilen su miktarını ekrana yazdırma
for i in range(len(future_predictions_orig)):
    print(f"Tahmin edilen su miktarı {future_dates[i].strftime('%Y-%m-%d')}: {future_predictions_orig[i][0]} m
```

Şekil 15. LSTM için gelecek 10 günü tahmin etme

- df[-10:]: DataFrame'in son 10 satırını alır. Bu, son 10 günlük veriyi içerir.
- create_features(last_10_days): create_features fonksiyonunu kullanarak son 10 günlük veri için ek özellikler oluşturur. Bu fonksiyon, tarih bilgisine dayalı olarak dayofweek, quarter, month, year, ve dayofyear gibi yeni sütunlar ekler.
- last_10_days[FEATURES]: FEATURES listesinde tanımlanan özellikleri seçer ve X_future değişkenine atar. Bu özellikler modelin girdi olarak kullanacağı verileri içerir.
- scaler_X.transform(X_future): MinMaxScaler kullanarak X_future verilerini normalize eder. Bu, modelin eğitimde gördüğü ölçekle aynı ölçekle veriyi kullanmasını sağlar.
- np.array(X_future_normalized).reshape((X_future_normalized.shape[0] , 1, X_future_normalized.shape[1])): Normalize edilmiş veriyi 3 boyutlu bir numpy dizisine dönüştürür. Bu, LSTM modelinin beklediği giriş formatıdır. (örneğin, num_samples, timesteps, num_features)
- model.predict(X_future_formatted): LSTM modelini kullanarak gelecek 10 gün için su tüketimi tahminleri yapar. Model, normalize edilmiş veriler üzerinde tahmin yapar ve normalize edilmiş tahmin değerlerini döner.
- scaler_y.inverse_transform(future_predictions.reshape(-1, 1)): Tahmin edilen değerleri orijinal ölçeğe döndürür. reshape(-1, 1) ifadesi, veriyi inverse_transform fonksiyonunun beklediği formatta yeniden şekillendirir.

- df.index[-1]: DataFrame'in indeksinin (tarih sütunu) son elemanını alır. Bu, en son veri girişinin tarihini temsil eder.
- [last_date + timedelta(days=i) for i in range(1, 11)]: Bu ifade, last_date tarihine timedelta(days=i) ekleyerek bir liste oluşturur. timedelta(days=i), last_date tarihine i gün ekler.
- range(1, 11): 1'den 10'a kadar olan sayılar için döngü oluşturur.
- last_date + timedelta(days=i): last_date tarihine i gün ekler. Örneğin, i=1 olduğunda, last_date tarihine 1 gün ekler ve bu yeni tarihi listeye ekler.
- for i in range(len(future_predictions_orig)): Tahmin edilen her bir gün için döngü başlatır. future_predictions_orig dizisinin uzunluğu kadar döngü devam eder. Bu, gelecekteki 10 gün için yapılan tahminleri içerir.
- print(f"Tahmin edilen su miktarı {future_dates[i].strftime('%Y-%m-%d')}: {future_predictions_orig[i][0]} m³/gün"):
- f"Tahmin edilen su miktarı {future_dates[i].strftime('%Y-%m-%d')}: {future_predictions_orig[i][0]} m³/gün": F-string kullanarak tahmin edilen su miktarını ve ilgili tarihi formatlar.
- {future_dates[i].strftime('%Y-%m-%d')}: future_dates listesindeki i inci tarih elemanını alır ve strftime('%Y-%m-%d') kullanarak bu tarihi 'YYYY-MM-DD' formatında bir stringe dönüştürür.
- {future_predictions_orig[i][0]}: future_predictions_orig dizisindeki i inci tahmin edilen su miktarını alır. future_predictions_orig iki boyutlu bir dizi olduğu için [i][0] kullanılarak tahmin edilen su miktarı alınır.
- print(...): Formatlanmış stringi ekrana yazdırır. Örneğin, çıktı şu şekilde olabilir: Tahmin edilen su miktarı 2024-05-29: 750.0 m³/gün

ÇIKTI:

```
    print(f"Tahmin edilen su miktarı {future_dates[i].strfti
```

```
Tahmin edilen su miktarı 2023-11-28: 10.328914642333984 m³/g
Tahmin edilen su miktarı 2023-11-29: 10.33346176147461 m³/gü
Tahmin edilen su miktarı 2023-11-30: 10.32333755493164 m³/gü
Tahmin edilen su miktarı 2023-12-01: 10.32296085357666 m³/gü
Tahmin edilen su miktarı 2023-12-02: 10.323165893554688 m³/g
Tahmin edilen su miktarı 2023-12-03: 10.32341194152832 m³/gü
Tahmin edilen su miktarı 2023-12-04: 10.32514476776123 m³/gü
Tahmin edilen su miktarı 2023-12-05: 10.327629089355469 m³/g
Tahmin edilen su miktarı 2023-12-06: 10.332171440124512 m³/g
Tahmin edilen su miktarı 2023-12-07: 10.322395324707031 m³/g
```

Şekil 16. LSTM için gelecek 10 gün tahminini görselleştirme

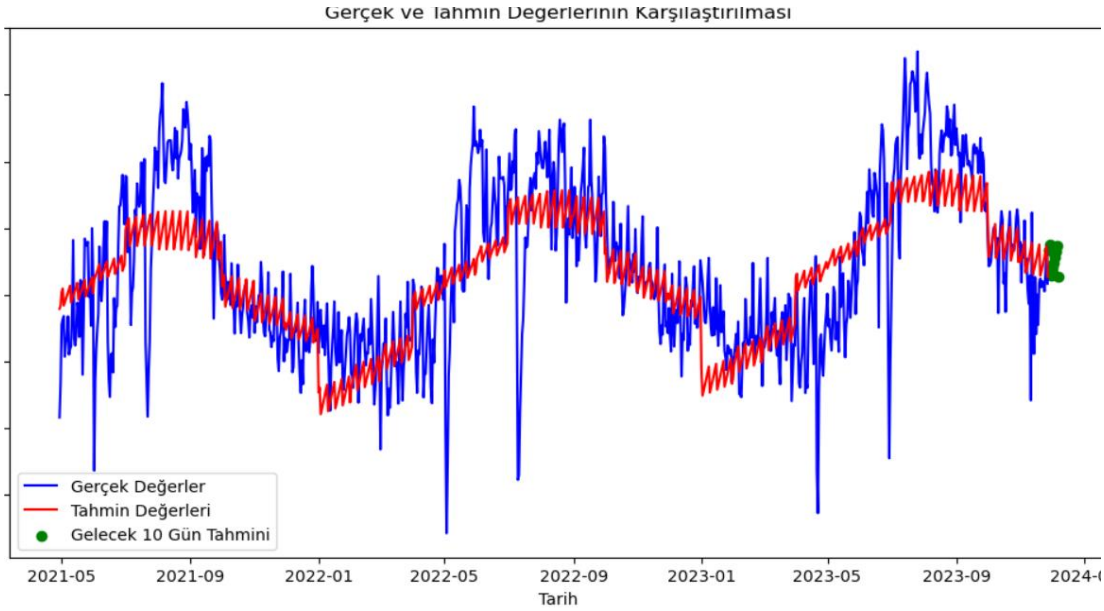
[26]:
```
# Tahmin edilen su miktarlarını ve tarihleri içeren bir DataFrame oluşturma
future_results = pd.DataFrame({'Tarih': future_dates, 'Tahmin': future_predictions_orig.flatten()})
future_results = future_results.set_index('Tarih')

# Gelecek 10 gün boyunca tahmin edilen su miktarlarının görselleştirilmesi
plt.figure(figsize=(12, 6))
plt.plot(results.index, results['Gerçek'], label='Gerçek Değerler', color='blue')
plt.plot(results.index, results['Tahmin'], label='Tahmin Değerleri', color='red')
plt.scatter(future_results.index, future_results['Tahmin'], color='green', label='Gelecek 10 Gün Tahmini')
plt.title('Gerçek ve Tahmin Değerlerinin Karşılaştırılması')
plt.xlabel('Tarih')
plt.ylabel('Günlük Tüketim (m³/gün)')
plt.legend()
plt.show()
```

Şekil 17. LSTM için gelecek 10 gün tahminini görselleştirme için gerekli kodlar

- pd.DataFrame({'Tarih': future_dates, 'Tahmin': future_predictions_orig.flatten()}): Gelecek 10 günün tarihleri (future_dates) ve modelin tahmin ettiği su tüketim miktarlarını (future_predictions_orig.flatten()) içeren bir pandas DataFrame oluşturur.
- future_dates: Gelecek 10 günün tarihlerini içerir.
- future_predictions_orig.flatten(): Tahmin edilen su miktarlarını tek boyutlu bir diziye dönüştürür.

- future_results.set_index('Tarih'): Tarih sütununu DataFrame'in indeksi olarak ayarlar. Bu, tarihlerin kolayca erişilebilir olmasını sağlar.
- plt.figure(figsize=(12, 6)): Yeni bir figure oluşturur ve boyutlarını 12x6 inç olarak ayarlar.
- plt.plot(results.index, results['Gerçek'], label='Gerçek Değerler', color='blue'): Gerçek su tüketim değerlerini çizgi grafiği olarak çizer. X ekseninde tarihleri (results.index), Y ekseninde gerçek su tüketim değerlerini (results['Gerçek']) kullanır. Çizgi rengi mavi olarak ayarlanır.
- plt.plot(results.index, results['Tahmin'], label='Tahmin Değerleri', color='red'): Modelin tahmin ettiği su tüketim değerlerini çizgi grafiği olarak çizer. X ekseninde tarihleri (results.index), Y ekseninde tahmin edilen su tüketim değerlerini (results['Tahmin']) kullanır. Çizgi rengi kırmızı olarak ayarlanır.
- plt.scatter(future_results.index, future_results['Tahmin'], color='green', label='Gelecek 10 Gün Tahmini'): Gelecek 10 gün için tahmin edilen su tüketim değerlerini nokta grafiği olarak çizer. X ekseninde gelecekteki tarihleri (future_results.index), Y ekseninde tahmin edilen su tüketim değerlerini (future_results['Tahmin']) kullanır. Nokta rengi yeşil olarak ayarlanır.
- plt.title('Gerçek ve Tahmin Değerlerinin Karşılaştırılması'): Grafiğe bir başlık ekler.
- plt.xlabel('Tarih'): X eksenine "Tarih" etiketini ekler.
- plt.ylabel('Günlük Tüketim (m^3/gün)'): Y eksenine "Günlük Tüketim (m^3/gün)" etiketini ekler.
- plt.legend(): Grafikteki çizgiler ve noktalar için bir efsane ekler, bu sayede hangi rengin neyi temsil ettiğini gösterir.
- plt.show(): Grafiği ekranda görüntüler.

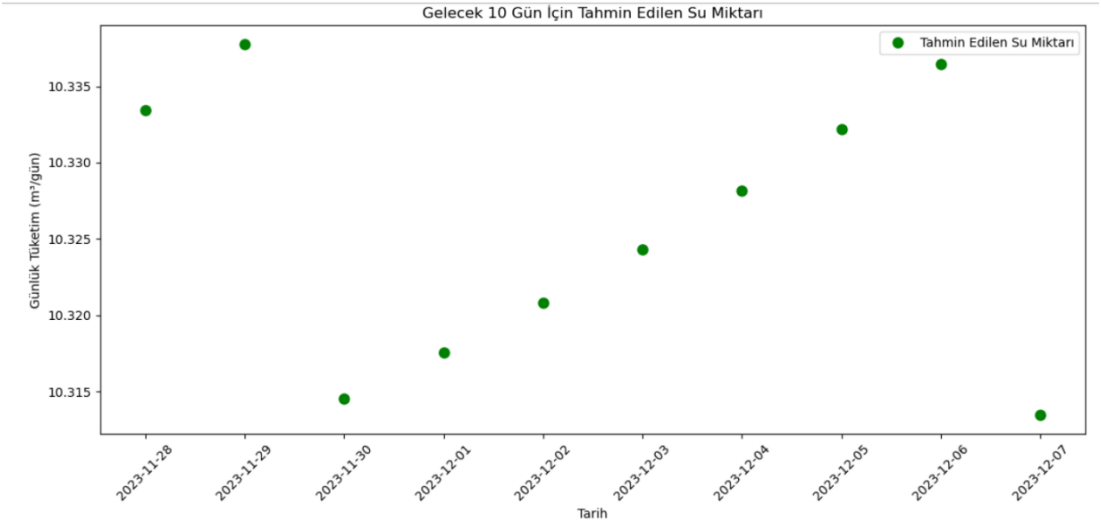
ÇIKTI:




```
[27]: # Gelecek 10 gün boyunca tahmin edilen su miktarlarının detaylı nokta görselleştirilmesi
plt.figure(figsize=(12, 6))
plt.plot(future_results.index, future_results['Tahmin'], 'go', markersize=8, label='Tahmin Edilen Su Miktarı')
plt.title('Gelecek 10 Gün İçin Tahmin Edilen Su Miktarı')
plt.xlabel('Tarih')
plt.ylabel('Günlük Tüketim (m³/gün)')
plt.xticks(rotation=45)
plt.legend()
plt.tight_layout()
plt.show()
```

Şekil 18. LSTM için gelecek 10 gün tahminini nokta tablosu oluşturma

ÇIKTI:



Şekil 19. LSTM için gelecek 10 gün tahminini nokta görselleştirmesi

4.1.13. Gelecek 1 yılı (365 günü) Tahmin Etme ve Görselleştirilmesi

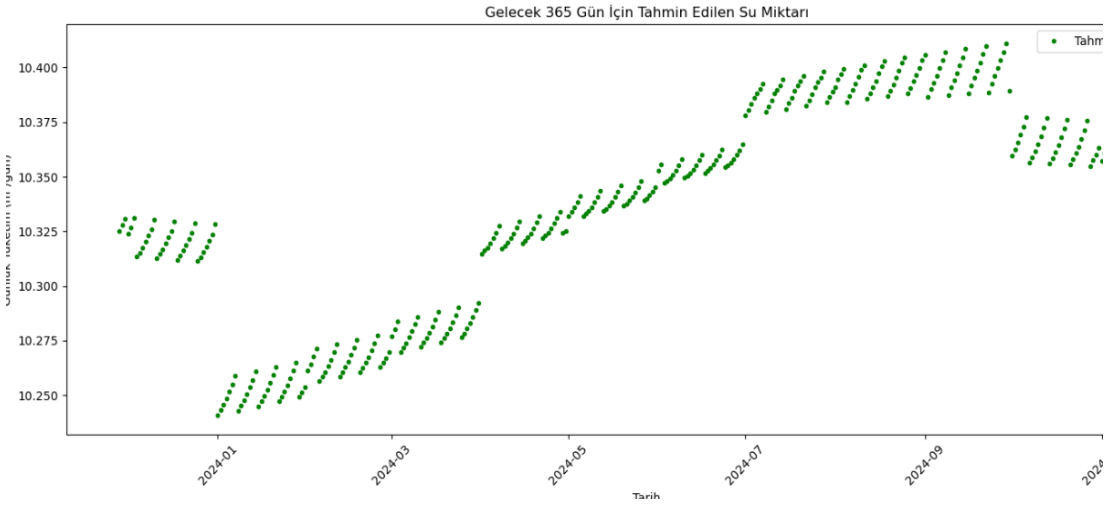
```
# Son tahmin edilen tarihi alınması
last_date = df.index[-1]

# Gelecek 365 gün boyunca tahmin yapma
future_predictions_365 = []
future_dates_365 = []
for i in range(1, 366):  # Her adımda bir sonraki günün tahmini yapılır
    # Tahmin yapılacak tarih için özelliklerin oluşturulması
    future_date = last_date + timedelta(days=i)
    future_dates_365.append(future_date)
    future_features = {'dayofweek': future_date.dayofweek,
                       'quarter': future_date.quarter,
                       'month': future_date.month,
                       'year': future_date.year,
                       'dayofyear': future_date.timetuple().tm_yday}

    # Veriyi normalize etme
    future_features_normalized = scaler_X.transform([list(future_features.values())])

    # Veriyi LSTM modeli için uygun formata dönüştürme
    future_features_formatted = np.array(future_features_normalized).reshape((1, 1, len(FEATURES)))

    # Tahmin yapma
    future_prediction = model.predict(future_features_formatted)

    # Tahmin edilen su miktarını orijinal ölçeğe döndürme ve listeye ekleme
    future_prediction_orig = scaler_y.inverse_transform(future_prediction.reshape(-1, 1))
    future_predictions_365.append(future_prediction_orig[0][0])

# Gelecek 365 gün boyunca tahmin edilen su miktarlarını içeren bir DataFrame oluşturma
future_results_365 = pd.DataFrame({'Tarih': future_dates_365, 'Tahmin': future_predictions_365})
future_results_365 = future_results_365.set_index('Tarih')
```

Şekil 20. LSTM için gelecek 1 yılı tahmin etme ve görselleştirme

- from datetime import datetime, timedelta: datetime ve timedelta sınıflarını içe aktarır. Bu sınıflar, tarih ve zaman hesaplamaları yapmak için kullanılır.
- import warnings: Python uyarılarını yönetmek için kullanılır.
- warnings.filterwarnings("ignore"): Tüm uyarıları görmezden gelir, böylece kod çalışırken gereksiz uyarılar ekrana yazdırılmaz.
- last_date = df.index[-1]: DataFrame df'nin indeksindeki (tarih sütunu) son değeri alır. Bu, mevcut veri setindeki en son tarihtir.
- future_predictions_365: Gelecek 365 gün için tahmin edilen su tüketim miktarlarını depolamak için boş bir liste.
- future_dates_365: Gelecek 365 günün tarihlerini depolamak için boş bir liste.
- for i in range(1, 366): 1'den 365'e kadar (365 dahil) bir döngü oluşturur. Her döngü iterasyonunda bir sonraki günün tahmini yapılır.
- future_date = last_date + timedelta(days=i): Son tarihe i gün ekleyerek gelecekteki tarihi hesaplar.
- future_dates_365.append(future_date): Hesaplanan tarihi future_dates_365 listesine ekler.
- future_features: Hesaplanan tarih için özellikler oluşturur (haftanın günü, çeyrek, ay, yıl ve yılın günü gibi).
- future_features_normalized = scaler_X.transform([list(future_features.values())]): Gelecek tarihe ait özellikleri normalize eder. Bu, modelin eğitiminde kullanılan aynı ölçekleme yöntemiyle yapılır.
- future_features_formatted = np.array(future_features_normalized).reshape((1, 1, len(FEATURES))): Normalize edilmiş özellikleri LSTM modelinin kabul edeceği formata dönüştürür. Bu, 3 boyutlu bir dizi haline getirilir: (örnek sayısı, zaman adımı, özellik sayısı).

- future_prediction = model.predict(future_features_formatted): LSTM modelini kullanarak gelecekteki gün için su tüketim tahmini yapar.
- future_prediction_orig = scaler_y.inverse_transform(future_prediction.reshape(-1, 1)): Normalize edilmiş tahmin değerlerini orijinal ölçeğe geri dönüştürür.
- future_predictions_365.append(future_prediction_orig[0][0]): Orijinal ölçeğe dönüştürülmüş tahmin değerini future_predictions_365 listesine ekler.
- future_results_365 = pd.DataFrame({'Tarih': future_dates_365, 'Tahmin': future_predictions_365}): Gelecek 365 günün tarihleri ve modelin tahmin ettiği su tüketim miktarlarını içeren bir pandas DataFrame oluşturur.
- future_results_365 = future_results_365.set_index('Tarih'): Tarih sütununu DataFrame'in indeksi olarak ayarlar, böylece tarihleri kolayca erişilebilir hale getirir.

ÇIKTI:



Şekil 21. LSTM için gelecek 1 yıl tahmin grafiği

4.2. MLPRegressor Modeli

4.2.1. Gerekli Kütüphanelerin İthal Edilmesi

```
[3]: import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
     from sklearn.model_selection import train_test_split, GridSearchCV
     from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error, mean_absolute_percentage_error
     from sklearn.preprocessing import MinMaxScaler
     from sklearn.neural_network import MLPRegressor
```

Şekil 22. MLPRegressor için gerekli kütüphanelerin ithal edilmesi

Kodun başında gerekli kütüphaneler ve modüller içe aktarılıyor. Bu kütüphaneler ve fonksiyonlar şunları içeriyor:

- pandas: Veri analizi ve manipülasyonu için kullanılır. DataFrame yapısıyla tabular verileri işlemede oldukça güçlüdür.
- numpy: Bilimsel hesaplamalar için kullanılır. Özellikle dizilerle çalışırken ve matematiksel fonksiyonları uygularken yararlıdır.
- matplotlib.pyplot: Grafik ve görselleştirme için kullanılır. Veriyi görselleştirmek için çeşitli grafikler çizmenize olanak tanır.
- seaborn: Matplotlib üzerine kurulu bir görselleştirme kütüphanesidir ve daha çekici ve bilgilendirici grafikler oluşturmak için kullanılır.
- sklearn.model_selection: Model eğitim ve değerlendirme süreçleri için veri setini bölmeye ve model seçmeye yarayan araçlar içerir.
- train_test_split: Veriyi eğitim ve test setlerine bölmek için kullanılır.
- GridSearchCV: Model parametrelerini optimize etmek için kullanılır.
- sklearn.metrics: Model performansını değerlendirmek için çeşitli metrikler içerir.
- r2_score: Determination coefficient, modelin açıklama gücünü ölçer.
- mean_absolute_error (MAE): Tahminlerin ortalama mutlak hatasını ölçer.

- mean_squared_error (MSE): Tahminlerin ortalama karesel hatasını ölçer.
- mean_absolute_percentage_error (MAPE): Tahminlerin ortalama mutlak yüzdelik hatasını ölçer.
- sklearn.preprocessing: Veriyi ön işleme için araçlar içerir.
- MinMaxScaler: Veriyi belirli bir aralığa (genellikle 0-1) ölçekler.
- sklearn.neural_network: Yapay sinir ağı modellerini içerir.
- MLPRegressor: Çok katmanlı perceptron regressor, regresyon problemleri için kullanılır.

4.2.2. Veri setini Yükleme ve Ön İşleme

```
# Veriyi yükleme ve ön işleme
df = pd.read_excel("rainfall-and-daily-consumption-data-on-istanbul-dams.xlsx")

df = df[['Tarih', 'İstanbul günlük tüketim(m³/gün)']]
df = df.set_index("Tarih")
df.index = pd.to_datetime(df.index)
df['İstanbul günlük tüketim(m³/gün)'] = df['İstanbul günlük tüketim(m³/gün)'] // 100
df['İstanbul günlük tüketim(m³/gün)'] = df['İstanbul günlük tüketim(m³/gün)'].astype(float)
df = np.log(df)
```

Şekil 23. MLPRegressor için veri yükleme ve ön işleme

- pd.read_excel: Excel dosyasından veri yükler. Burada, İstanbul barajlarına ait günlük yağış ve su tüketimi verileri içeren bir Excel dosyası yüklenir.
- df[['Tarih', 'İstanbul günlük tüketim(m³/gün)']]: Veri çerçevesinden sadece "Tarih" ve "İstanbul günlük tüketim(m³/gün)" sütunları seçilir.
- df.set_index("Tarih"): "Tarih" sütunu veri çerçevesinin indeksi olarak ayarlanır.
- pd.to_datetime(df.index): Tarih indeksleri datetime formatına dönüştürülür.
- df['İstanbul günlük tüketim(m³/gün)'] // 100: Su tüketimi değerleri 100 ile bölünerek küçültülür. Bu, büyük sayıları daha yönetilebilir hale getirmek için yapılır.

- df['İstanbul günlük tüketim(m³/gün)'].astype(float): Veriler float tipine dönüştürülür.
- np.log(df): Verilerin logaritması alınır. Bu, verilerdeki büyük dalgalanmaları azaltmak ve modelin daha iyi performans göstermesini sağlamak için yapılır.

4.2.3. Eğitim ve Test Veri Setlerini Ayırma

```
# Eğitim ve test veri setlerini ayırma
train_size = int(len(df) * 0.80)
test_size = len(df) - train_size
train, test = df[0:train_size], df[train_size:len(df)]
```

Şekil 24. MLPRegressor için eğitim ve test veri setlerini ayırma

- train_size = int(len(df) * 0.80): Eğitim seti için veri çerçevesinin %80'i hesaplanır.
- test_size = len(df) - train_size: Test seti için geri kalan %20 veri hesaplanır.
- train, test = df[0:train_size], df[train_size:len(df)]: Eğitim ve test setleri belirlenen boyutlara göre ayrılır.

4.2.4. Özelliklerin Oluşturulması

```
# Özellikleri oluşturma
def create_features(df):
    df = df.copy()
    df['dayofweek'] = df.index.dayofweek
    df['quarter'] = df.index.quarter
    df['month'] = df.index.month
    df['year'] = df.index.year
    df['dayofyear'] = df.index.dayofyear
    return df

df = create_features(df)
train = create_features(train)
test = create_features(test)

FEATURES = ['dayofweek', 'quarter', 'month', 'year', 'dayofyear']
TARGET = 'İstanbul günlük tüketim(m³/gün)'

X_train = train[FEATURES]
y_train = train[TARGET]

X_test = test[FEATURES]
y_test = test[TARGET]
```

Şekil 25. MLPRegressor için özellik oluşturulması

- df.copy(): Veri çerçevesinin bir kopyası oluşturulur. Bu, orijinal veriyi değiştirmeden işlemler yapmayı sağlar.
- df.index.dayofweek: Haftanın günü (0: Pazartesi, 6: Pazar) eklenir.
- df.index.quarter: Yılın çeyreği (1, 2, 3, 4) eklenir.
- df.index.month: Ay (1-12) eklenir.
- df.index.year: Yıl eklenir.
- df.index.dayofyear: Yılın günü (1-365) eklenir.
- return df: Yeni özellikler eklenmiş veri çerçevesi döndürülür.
- create_features(df): Veri çerçevesine yeni özellikler eklenir.
- FEATURES: Modelin eğitiminde kullanılacak özellikler listesi.
- TARGET: Modelin tahmin etmeye çalışacağı hedef değişken.
- train[FEATURES]: Eğitim setinden özelliklerin seçilmesi.
- train[TARGET]: Eğitim setinden hedef değişkenin seçilmesi.

- test[FEATURES]: Test setinden özelliklerin seçilmesi.
- test[TARGET]: Test setinden hedef değişkenin seçilmesi.

4.2.5. Verilerin Normalize Edilmesi

```
# Veriyi normalize etme
scaler_X = MinMaxScaler()
scaler_y = MinMaxScaler()

X_train = scaler_X.fit_transform(X_train)
X_test = scaler_X.transform(X_test)

y_train = scaler_y.fit_transform(y_train.values.reshape(-1, 1)).ravel()
y_test = scaler_y.transform(y_test.values.reshape(-1, 1)).ravel()
```

Şekil 26. MLPRegressor için verilerin normalize edilmesi

- MinMaxScaler(): Verileri 0 ile 1 arasında ölçeklendirmek için kullanılan normalizasyon yöntemi.
- fit_transform: Eğitim verilerine uygulanır ve aynı zamanda dönüştürülür.
- transform: Test verilerine uygulanır (daha önce eğitim verilerine uygulanan ölçekleme kullanılarak).
- values.reshape(-1, 1): y_train ve y_test değerleri 2 boyutlu hale getirilir.
- ravel(): Tek boyutlu hale getirilir.

4.2.6. MLPRegressor Modelinin Oluşturulması, Modelin Eğitimi, Tahmin Yapma, Tahminleri Orijinal Ölçeğe Döndürme

```
# MLPRegressor modelini oluşturma ve eğitme
mlp_reg = MLPRegressor(hidden_layer_sizes=(100,), activation='relu', solver='adam', max_iter=200, random_state=42)
mlp_reg.fit(X_train, y_train)
```

Şekil 27. MLPRegressor için modelini oluşturma ve eğitme

- MLPRegressor: Çok katmanlı perceptron regressor, regresyon problemleri için kullanılır.

- hidden_layer_sizes=(100,): Gizli katmanda 100 nöron.
- activation='relu': ReLU aktivasyon fonksiyonu.
- solver='adam': Adam optimizasyon algoritması.
- max_iter=200: Maksimum 200 iterasyon.
- random_state=42: Rastgelelik için sabit tohum değeri.
- fit: Modeli eğitim verisi ile eğitir.

4.2.7. Test Verileri Üzerinde Tahmin Yapma ve Tahminleri Orijinal Ölçeğe Döndürme

```
# Test verileri üzerinde tahmin yapma
y_pred = mlp_reg.predict(X_test)

# Tahminleri orijinal ölçeğe döndürme
y_test_orig = scaler_y.inverse_transform(y_test.reshape(-1, 1))
y_pred_orig = scaler_y.inverse_transform(y_pred.reshape(-1, 1))
```

Şekil 28. MLPRegressor için tahmin yapma ve tahminleri orijinal içeriğe döndürme

- predict: Test verisi üzerinde tahminler yapar.
- inverse_transform: Normalize edilmiş veriyi orijinal ölçeğe döndürür.

4.2.8. Hata Metrik Değerlerini Hesaplama

```
# MSE ve RMSE değerlerini hesaplama
mse = mean_squared_error(y_test_orig, y_pred_orig)
rmse = np.sqrt(mse)

print("Mean Squared Error (MSE): {:.4f}".format(mse))
print("Root Mean Squared Error (RMSE): {:.4f}".format(rmse))

# MAE hesaplama
mae = mean_absolute_error(y_test_orig, y_pred_orig)
print("Mean Absolute Error (MAE):", mae)

# MAPE hesaplama
mape = mean_absolute_percentage_error(y_test_orig, y_pred_orig) * 100
print("Mean Absolute Percentage Error (MAPE):", mape)
```

Şekil 29. MLPRegressor için hata metriklerini hesaplama

- mean_squared_error(y_test_orig, y_pred_orig): MSE hesaplanır.
- np.sqrt(mse): MSE'nin karekökü alınarak RMSE elde edilir.
- mean_absolute_error(y_test_orig, y_pred_orig): MAE hesaplanır. MAE, tahmin edilen ve gerçek değerler arasındaki mutlak farkların ortalamasıdır.
- mean_absolute_percentage_error(y_test_orig, y_pred_orig): MAPE hesaplanır. MAPE, tahmin hatasının yüzdesini belirtir ve genellikle % cinsinden ifade edilir.

ÇIKTI:

```
Mean Squared Error (MSE): 0.0024
Root Mean Squared Error (RMSE): 0.0485
Mean Absolute Error (MAE): 0.038298088621056846
Mean Absolute Percentage Error (MAPE): 0.37168875
```

Şekil 30. MLPRegressor için hata değerleri

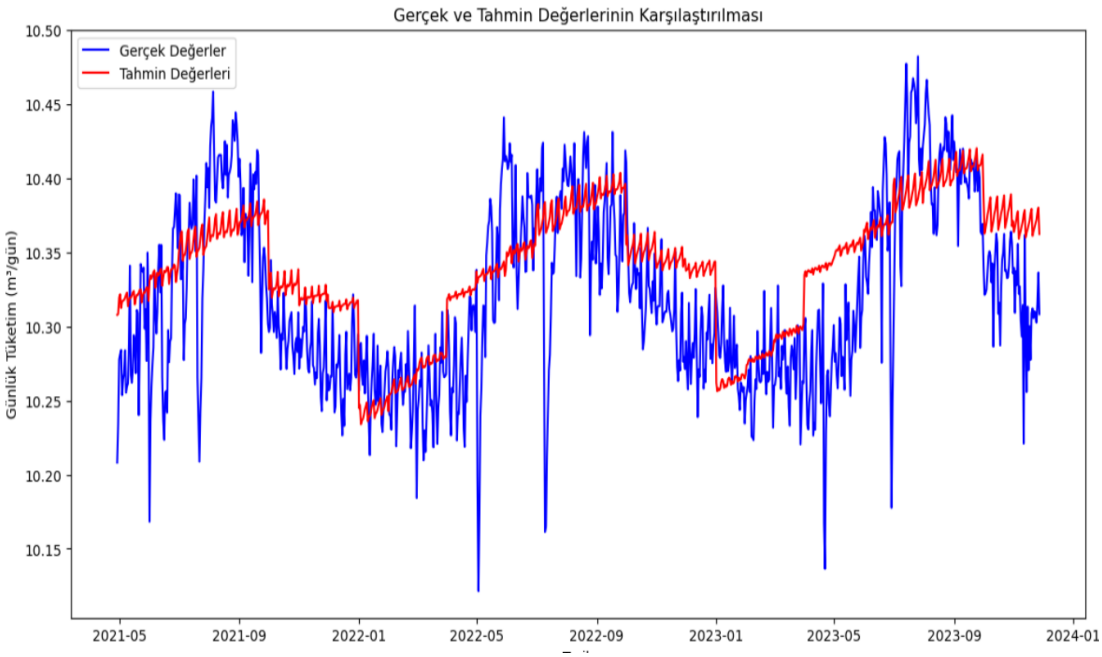
4.2.9. Gerçek ve Tahmin Değerleri Birleştirme ve Görselleştirme

```
# Gerçek ve tahmin değerleri birleştirme
results = pd.DataFrame({'Tarih': test.index, 'Gerçek': y_test_orig.flatten(), 'Tahmin': y_pred_orig.flatten
results = results.set_index('Tarih')

# Görselleştirme
plt.figure(figsize=(14, 7))
plt.plot(results.index, results['Gerçek'], label='Gerçek Değerler', color='blue')
plt.plot(results.index, results['Tahmin'], label='Tahmin Değerleri', color='red')
plt.title('Gerçek ve Tahmin Değerlerinin Karşılaştırılması')
plt.xlabel('Tarih')
plt.ylabel('Günlük Tüketim (m³/gün)')
plt.legend()
plt.show()
```

Şekil 31. MLPRegressor için gerçek ve tahmin değerlerini birleştirme ve görselleştirme kodları

- {'Tarih': test.index, 'Gerçek': y_test_orig.flatten(), 'Tahmin': y_pred_orig.flatten()}: DataFrame'in içeriği için bir sözlük oluşturur.
- 'Tarih': test.index: Test setinin tarihlerini içerir.
- 'Gerçek': y_test_orig.flatten(): Gerçek su tüketimi değerlerini içerir, flatten() metodu çok boyutlu diziyi tek boyutlu hale getirir.
- 'Tahmin': y_pred_orig.flatten(): Model tarafından tahmin edilen su tüketimi değerlerini içerir, flatten() metodu yine çok boyutlu diziyi tek boyutlu hale getirir.
- results.set_index('Tarih'): 'Tarih' sütununu DataFrame'in indeksi olarak ayarlar. Bu, tarihler üzerinden veri manipülasyonu ve görselleştirme yapmayı kolaylaştırır.
- plt.figure(figsize=(14, 7)): Yeni bir figür oluşturur ve boyutlarını belirler. Burada 14x7 inç boyutlarında bir grafik oluşturulur.
- plt.plot(results.index, results['Gerçek'], label='Gerçek Değerler', color='blue'): Gerçek su tüketimi değerlerini çizer.
- results.index: X ekseninde tarihleri kullanır.
- results['Gerçek']: Y ekseninde gerçek su tüketimi değerlerini kullanır.
- label='Gerçek Değerler': Grafikte bu çizginin etiketini belirler.
- color='blue': Çizginin rengini mavi yapar.
- plt.plot(results.index, results['Tahmin'], label='Tahmin Değerleri', color='red'): Tahmin edilen su tüketimi değerlerini çizer.
- results.index: X ekseninde tarihleri kullanır.
- results['Tahmin']: Y ekseninde tahmin edilen su tüketimi değerlerini kullanır.
- label='Tahmin Değerleri': Grafikte bu çizginin etiketini belirler.
- color='red': Çizginin rengini kırmızı yapar.

- plt.title('Gerçek ve Tahmin Değerlerinin Karşılaştırılması'): Grafiğe başlık ekler.
- plt.xlabel('Tarih'): X eksenine 'Tarih' etiketi ekler.
- plt.ylabel('Günlük Tüketim (m³/gün)'): Y eksenine 'Günlük Tüketim (m³/gün)' etiketi ekler.
- plt.legend(): Grafikteki çizgilerin neyi temsil ettiğini gösteren bir efsane (legend) ekler. Bu efsane, label parametresi ile belirtilen etiketleri kullanır.
- plt.show(): Grafiği ekranda gösterir. Bu komut olmadan grafik oluşturulsa bile görüntülenmez.



Şekil 32. MLPRegressor için gerçek ve tahmin değerlerini karşılaştırma

4.2.10. Gelecek 10 günü Tahmin Etme ve Görselleştirilmesi

```
[10]: from datetime import datetime, timedelta

# Gelecek 10 günün tarih aralığını belirleme
start_date = df.index[-1] + timedelta(days=1)
end_date = start_date + timedelta(days=9)
forecast_dates = pd.date_range(start=start_date, end=end_date, freq='D')

# Gelecek 10 gün için özellikler oluşturma
forecast_features = create_features(pd.DataFrame(index=forecast_dates))

# Özellikleri normalize etme
forecast_features_normalized = scaler_X.transform(forecast_features)

# Gelecek 10 günün su tüketimini tahmin etme
forecast_scaled = mlp_reg.predict(forecast_features_normalized)

# Tahminleri orijinal ölçeklere dönüştürme
forecast = scaler_y.inverse_transform(forecast_scaled.reshape(-1, 1)).flatten()

# Tahminleri ve tarihleri birleştirme
forecast_results = pd.DataFrame({'Tarih': forecast_dates, 'Tahmin Edilen Su Tüketimi (m³/gün)': forecast})
forecast_results = forecast_results.set_index('Tarih')

# Tahmin edilen su tüketimini görselleştirme
plt.figure(figsize=(14, 7))
plt.plot(results.index, results['Gerçek'], label='Gerçek Değerler', color='blue')
plt.plot(results.index, results['Tahmin'], label='Tahmin Değerleri', color='red')
plt.plot(forecast_results.index, forecast_results['Tahmin Edilen Su Tüketimi (m³/gün)'], label='Gelecek 10 Gün Tahmini',
plt.title('Su Tüketimi Tahmini ve Gelecek 10 Gün Tahmini')
plt.xlabel('Tarih')
plt.ylabel('Günlük Tüketim (m³/gün)')
plt.legend()
```

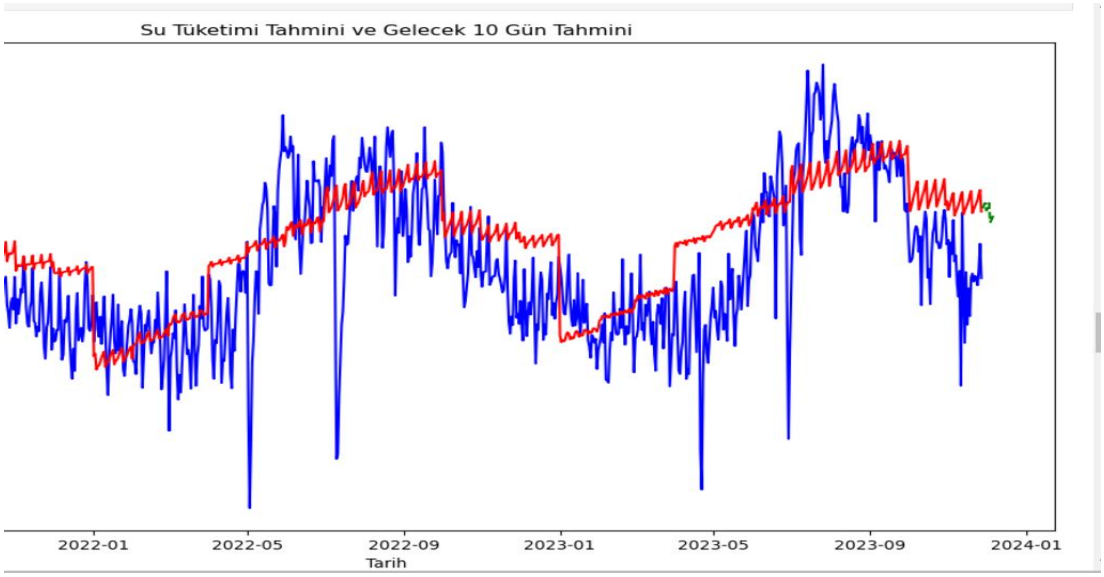
Şekil 33. MLPRegressor için gelecek 10 günü tahmin etme

- datetime ve timedelta: Tarih ve zaman işlemleri yapmak için kullanılan modüller.
- start_date: Verideki son tarihin bir gün sonrası. Bu, tahminlerin başlangıç tarihidir.
- df.index[-1]: Verideki son tarih.
- timedelta(days=1): Bir gün eklemek için kullanılır.
- end_date: start_date'den 9 gün sonrası. Bu, tahminlerin bitiş tarihidir.
- timedelta(days=9): Dokuz gün eklemek için kullanılır.
- pd.date_range: Belirtilen tarih aralığında, günlük (freq='D') bir dizi tarih oluşturur.
- start=start_date: Başlangıç tarihi.
- end=end_date: Bitiş tarihi.

- pd.DataFrame(index=forecast_dates): Gelecek 10 günü içeren, indeks olarak tarihleri kullanan boş bir DataFrame oluşturur.
- create_features: Daha önce tanımlanan fonksiyon, bu tarih aralığı için gerekli özellikleri oluşturur (haftanın günü, ay, yıl, vb.).
- scaler_X.transform: Daha önce eğitim verisi üzerinde fit edilmiş olan scaler kullanılarak, tahmin için oluşturulan özellikler normalize edilir.
- mlp_reg.predict: Normalize edilmiş özellikleri kullanarak model gelecek 10 gün için su tüketim tahminlerini yapar.
- scaler_y.inverse_transform: Normalize edilmiş tahminleri orijinal ölçeklerine geri dönüştürür.
- reshape(-1, 1): Tek boyutlu vektörü sütun vektörüne dönüştürür (inverse transform işlemi için gerekli).
- flatten(): Çok boyutlu diziyi tek boyutlu hale getirir.
- pd.DataFrame: Tahminleri ve tarihleri içeren bir DataFrame oluşturur.
- 'Tarih': forecast_dates: Tarih kolonunu oluşturur.
- 'Tahmin Edilen Su Tüketimi (m^3/gün)': forecast: Tahmin edilen su tüketim değerlerini oluşturur.
- set_index('Tarih'): "Tarih" kolonunu indeks olarak ayarlar.
- plt.figure(figsize=(14, 7)): Grafik boyutlarını ayarlar. Bu örnekte grafik 14 inç genişliğinde ve 7 inç yüksekliğinde olacaktır.
- plt.plot(results.index, results['Gerçek'], label='Gerçek Değerler', color='blue'):
- results.index: X ekseninde kullanılacak tarih verileri.
- results['Gerçek']: Y ekseninde kullanılacak gerçek su tüketimi değerleri.
- label='Gerçek Değerler': Bu çizgiye etiket verir.
- color='blue': Çizgi rengini mavi olarak ayarlar.

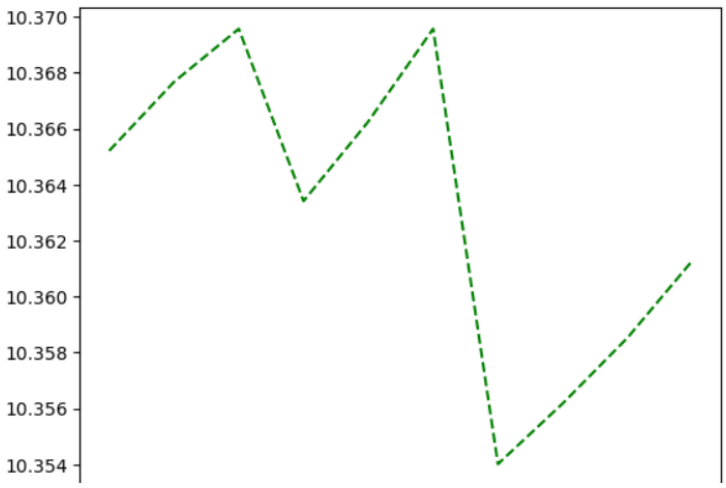
- plt.plot(results.index, results['Tahmin'], label='Tahmin Değerleri', color='red'):
- results['Tahmin']: Y ekseninde kullanılacak tahmin edilen su tüketimi değerleri.
- label='Tahmin Değerleri': Bu çizgiye etiket verir.
- color='red': Çizgi rengini kırmızı olarak ayarlar.
- plt.plot(forecast_results.index, forecast_results['Tahmin Edilen Su Tüketimi (m³/gün)'], label='Gelecek 10 Gün Tahmini', color='green', linestyle='--'):
- forecast_results.index: X ekseninde kullanılacak gelecek 10 günün tarih verileri.
- forecast_results['Tahmin Edilen Su Tüketimi (m³/gün)']: Y ekseninde kullanılacak gelecek 10 gün için tahmin edilen su tüketimi değerleri.
- label='Gelecek 10 Gün Tahmini': Bu çizgiye etiket verir.
- color='green': Çizgi rengini yeşil olarak ayarlar.
- linestyle='--': Çizgi stilini kesikli (dashed) olarak ayarlar.
- plt.title('Su Tüketimi Tahmini ve Gelecek 10 Gün Tahmini'): Grafiğe başlık ekler.
- plt.xlabel('Tarih'): X eksenine "Tarih" etiketi ekler.
- plt.ylabel('Günlük Tüketim (m³/gün)'): Y eksenine "Günlük Tüketim (m³/gün)" etiketi ekler.
- plt.legend(): Grafikte kullanılan çizgilerin açıklamalarını (etiketlerini) gösterir.
- plt.show(): Grafiği ekranda gösterir.

ÇIKTI:




```
11]: # Gelecek 10 günün tarihlerini daha iyi göstermek için x-eksenini ayarlama
     plt.xticks(rotation=45)

     # Tahmin edilen su tüketimini görselleştirme
     plt.plot(forecast_results.index, forecast_results['Tahmin Edilen Su Tüketimi (m³/gün)'], label='Gelecek 10 Gün Ta
```

```
11]: [<matplotlib.lines.Line2D at 0x18556835d90>]
```



Şekil 34. MLPRegressor için gelecek 10 günün tahmin grafikleri

4.2.11. Gelecek 1 yılı (365 günü) Tahmin Etme ve Görselleştirilmesi

```
[12]: from datetime import datetime, timedelta

# Gelecek 365 günün tarih aralığını belirleme
start_date = df.index[-1] + timedelta(days=1)
end_date = start_date + timedelta(days=364)
forecast_dates_1_year = pd.date_range(start=start_date, end=end_date, freq='D')

# Gelecek 1 yıl için özellikler oluşturma
forecast_features_1_year = create_features(pd.DataFrame(index=forecast_dates_1_year))

# Özellikleri normalize etme
forecast_features_normalized_1_year = scaler_X.transform(forecast_features_1_year)

# Gelecek 1 yılın su tüketimini tahmin etme
forecast_scaled_1_year = mlp_reg.predict(forecast_features_normalized_1_year)

# Tahminleri orijinal ölçeklere dönüştürme
forecast_1_year = scaler_y.inverse_transform(forecast_scaled_1_year.reshape(-1, 1)).flatten()

# Tahminleri ve tarihleri birleştirme
forecast_results_1_year = pd.DataFrame({'Tarih': forecast_dates_1_year, 'Tahmin Edilen Su Tüketi
forecast_results_1_year = forecast_results_1_year.set_index('Tarih')

forecast_results_1_year
```

Şekil 35. MLPRegressor için gelecek 1 yılı tahmin etme ve görselleştirme

- datetime ve timedelta: Tarih ve zaman işlemleri yapmak için kullanılan modüller.
- start_date: Verideki son tarihin bir gün sonrası. Bu, tahminlerin başlangıç tarihidir.
- df.index[-1]: Verideki son tarih.
- timedelta(days=1): Bir gün eklemek için kullanılır.
- end_date: start_date'den 364 gün sonrası. Bu, tahminlerin bitiş tarihidir.
- timedelta(days=364): Üç yüz altmış dört gün eklemek için kullanılır.
- pd.date_range: Belirtilen tarih aralığında, günlük (freq='D') bir dizi tarih oluşturur.
- start=start_date: Başlangıç tarihi.
- end=end_date: Bitiş tarihi.

- pd.DataFrame(index=forecast_dates_1_year): Gelecek 1 yılı içeren, indeks olarak tarihleri kullanan boş bir DataFrame oluşturur.
- create_features: Daha önce tanımlanan fonksiyon, bu tarih aralığı için gerekli özellikleri oluşturur (haftanın günü, ay, yıl, vb.).
- scaler_X.transform: Daha önce eğitim verisi üzerinde fit edilmiş olan scaler kullanılarak, tahmin için oluşturulan özellikler normalize edilir.
- mlp_reg.predict: Normalize edilmiş özellikleri kullanarak model gelecek 1 yıl için su tüketim tahminlerini yapar.
- scaler_y.inverse_transform: Normalize edilmiş tahminleri orijinal ölçeklerine geri dönüştürür.
- reshape(-1, 1): Tek boyutlu vektörü sütun vektörüne dönüştürür (inverse transform işlemi için gerekli).
- flatten(): Çok boyutlu diziyi tek boyutlu hale getirir
- pd.DataFrame: Tahminleri ve tarihleri içeren bir DataFrame oluşturur.
- 'Tarih': forecast_dates_1_year: Tarih kolonunu oluşturur.
- 'Tahmin Edilen Su Tüketimi (m^3/gün)': forecast_1_year: Tahmin edilen su tüketim değerlerini oluşturur.
- set_index('Tarih'): "Tarih" kolonunu indeks olarak ayarlar.
- forecast_results_1_year: Gelecek 1 yıl için tahmin edilen su tüketim değerlerini ve tarihlerini içeren DataFrame'i gösterir.

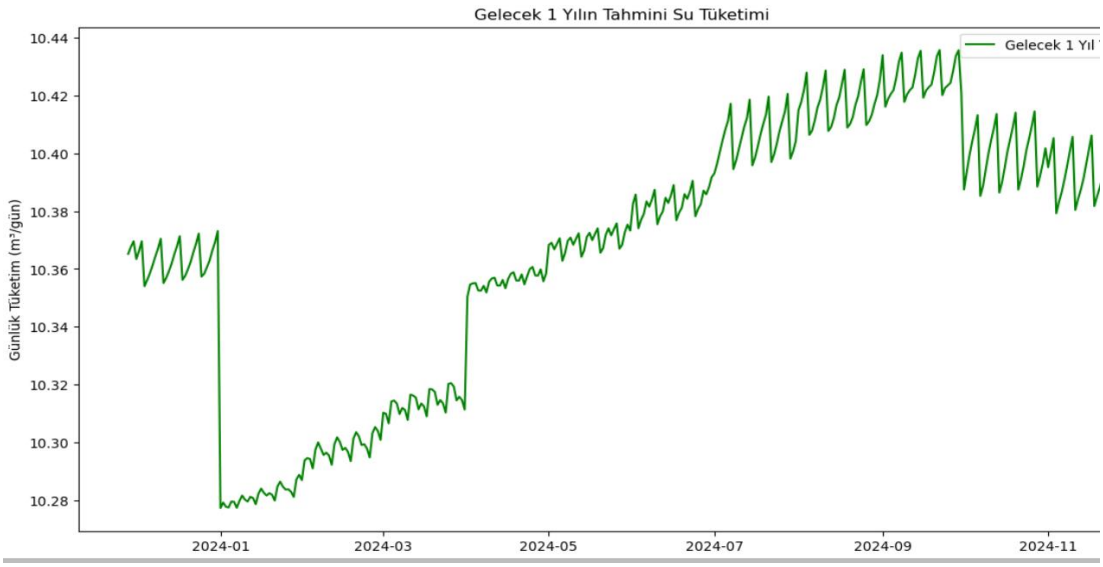
ÇIKTI:

[12]:

Tarih	Tahmin Edilen Su Tüketimi (m³/gün)
2023-11-28	10.365213
2023-11-29	10.367673
2023-11-30	10.369572
2023-12-01	10.363415
2023-12-02	10.366260
...	...
2024-11-22	10.396496
2024-11-23	10.401742
2024-11-24	10.406748
2024-11-25	10.383096
2024-11-26	10.385907

365 rows × 1 columns

[13]:
```
# Tahmin edilen su tüketimini görselleştirme
plt.figure(figsize=(14, 7))
plt.plot(forecast_results_1_year.index, forecast_results_1_year['Tahmin Edilen Su Tüketimi (m³/gün)'], label='Gelec
plt.title('Gelecek 1 Yılın Tahmini Su Tüketimi')
plt.xlabel('Tarih')
plt.ylabel('Günlük Tüketim (m³/gün)')
plt.legend()
plt.show()
```



Şekil 36. MLPRegressor için gelecek 1 yılı tahmin edilmesi ve grafiği

4.3. Destek Vektör Regresyon (Support Vector Regression) Modeli

4.3.1. Gerekli Kütüphanelerin İthal Edilmesi, Veri Setini Yükleme ve Ön İşleme İşlemleri

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, mean_absolute_error, mean_absolute_percentage_error
from sklearn.preprocessing import MinMaxScaler

# Veri setini yükleme ve hazırlama
df = pd.read_excel("rainfall-and-daily-consumption-data-on-istanbul-dams.xlsx")
df = df[['Tarih', 'İstanbul günlük tüketim(m³/gün)']]
df = df.set_index("Tarih")
df.index = pd.to_datetime(df.index)
df['İstanbul günlük tüketim(m³/gün)'] = df['İstanbul günlük tüketim(m³/gün)'] // 100
df['İstanbul günlük tüketim(m³/gün)'] = df['İstanbul günlük tüketim(m³/gün)'].astype(float)
df = np.log(df)

# Eğitim ve test veri setlerini ayırma
train_size = int(len(df) * 0.80)
test_size = len(df) - train_size
train, test = df.iloc[:train_size], df.iloc[train_size:]
```

Şekil 37. SVR için gerekli kütüphanelerin ithal edilmesi, veri setini yükleme ve ön işleme

4.3.2. Özelliklerin Oluşturulması

```
# Özellikleri oluşturma
def create_features(df):
    df = df.copy()
    df['dayofweek'] = df.index.dayofweek
    df['quarter'] = df.index.quarter
    df['month'] = df.index.month
    df['year'] = df.index.year
    df['dayofyear'] = df.index.dayofyear
    return df

df = create_features(df)
train = create_features(train)
test = create_features(test)

FEATURES = ['dayofweek', 'quarter', 'month', 'year', 'dayofyear']
TARGET = 'İstanbul günlük tüketim(m³/gün)'

X_train = train[FEATURES]
y_train = train[TARGET]

X_test = test[FEATURES]
y_test = test[TARGET]
```

Şekil 38. SVR için özelliklerin oluşturulması

4.3.3. Verilerin Normalize Edilmesi

```
# Veriyi normalize etme
scaler_X = MinMaxScaler()
scaler_y = MinMaxScaler()

X_train = scaler_X.fit_transform(X_train)
X_test = scaler_X.transform(X_test)

y_train = scaler_y.fit_transform(y_train.values.reshape(-1, 1)).ravel()
y_test = scaler_y.transform(y_test.values.reshape(-1, 1)).ravel()
```

Şekil 39. SVR için verilerin normalize edilmesi

4.3.4. SVR Modelinin Oluşturulması, Modelin Eğitimi, Tahmin Yapma, Tahminleri Orijinal Ölçeğe Döndürme

```
# SVR modelini oluşturma ve eğitme
svr_model = SVR(kernel='rbf', C=100, gamma=0.1, epsilon=.1)
svr_model.fit(X_train, y_train)

# Test verileri üzerinde tahmin yapma
y_pred = svr_model.predict(X_test)

# Tahminleri orijinal ölçeğe döndürme
y_test_orig = scaler_y.inverse_transform(y_test.reshape(-1, 1))
y_pred_orig = scaler_y.inverse_transform(y_pred.reshape(-1, 1))

# MSE ve RMSE değerlerini hesaplama
mse = mean_squared_error(y_test_orig, y_pred_orig)
rmse = np.sqrt(mse)

print("Mean Squared Error (MSE): {:.4f}".format(mse))
print("Root Mean Squared Error (RMSE): {:.4f}".format(rmse))

# MAE hesaplama
mae = mean_absolute_error(y_test_orig, y_pred_orig)
print("Mean Absolute Error (MAE): {:.4f}".format(mae))

# MAPE hesaplama
mape = mean_absolute_percentage_error(y_test_orig, y_pred_orig) * 100
print("Mean Absolute Percentage Error (MAPE): {:.4f}".format(mape))
```

Şekil 40. SVR için modelin oluşturulması, modelin eğitimi, tahmin yapma, tahminleri orijinal ölçeğe döndürme

ÇIKTI:

```
# MAE hesaplama
mae = mean_absolute_error(y_test_orig, y_pred_orig)
print("Mean Absolute Error (MAE): {:.4f}".format(mae)

# MAPE hesaplama
mape = mean_absolute_percentage_error(y_test_orig, y_
print("Mean Absolute Percentage Error (MAPE): {:.4f}"
```

```
Mean Squared Error (MSE): 0.0027
Root Mean Squared Error (RMSE): 0.0521
Mean Absolute Error (MAE): 0.0416
Mean Absolute Percentage Error (MAPE): 0.4021
```

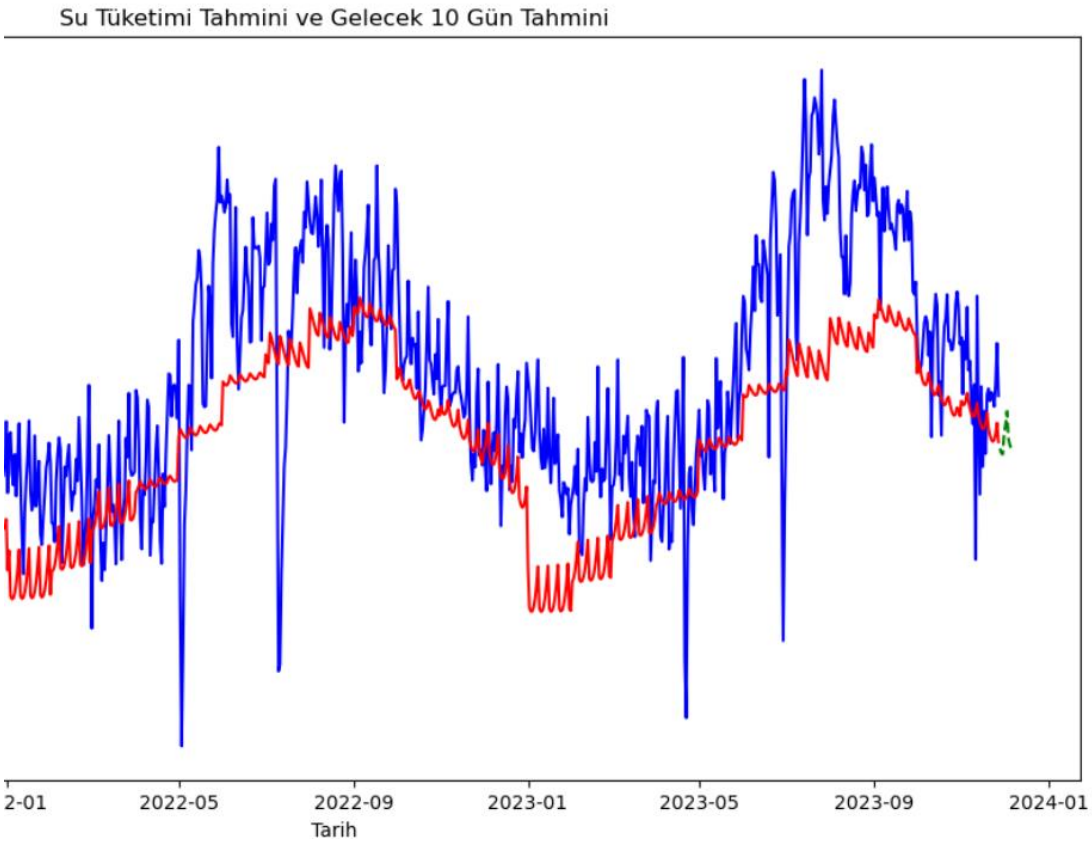
Şekil 41. SVR için hataların çıktısı

4.3.5. Gelecek 10 günü Tahmin Etme ve Görselleştirilmesi

```
[8]: from datetime import datetime, timedelta

# Gelecek 10 günün tarih aralığını belirleme
start_date = df.index[-1] + timedelta(days=1)
end_date = start_date + timedelta(days=9)
forecast_dates = pd.date_range(start=start_date, end=end_date, freq='D')

# Gelecek 10 gün için özellikler oluşturma
forecast_features = create_features(pd.DataFrame(index=forecast_dates))

# Özellikleri normalize etme
forecast_features_normalized = scaler_X.transform(forecast_features)

# Gelecek 10 günün su tüketimini tahmin etme
forecast_scaled = svr_model.predict(forecast_features_normalized)

# Tahminleri orijinal ölçeklere dönüştürme
forecast = scaler_y.inverse_transform(forecast_scaled.reshape(-1, 1)).flatten()

# Tahminleri ve tarihleri birleştirme
forecast_results = pd.DataFrame({'Tarih': forecast_dates, 'Tahmin Edilen Su Tüketimi (m³/gün)': forecast})
forecast_results = forecast_results.set_index('Tarih')

# Tahmin edilen su tüketimini görselleştirme
plt.figure(figsize=(14, 7))
plt.plot(results.index, results['Gerçek'], label='Gerçek Değerler', color='blue')
plt.plot(results.index, results['Tahmin'], label='Tahmin Değerleri', color='red')
plt.plot(forecast_results.index, forecast_results['Tahmin Edilen Su Tüketimi (m³/gün)'], label='Gelecek 10 Gün Tahmini',
plt.title('Su Tüketimi Tahmini ve Gelecek 10 Gün Tahmini')
plt.xlabel('Tarih')
plt.ylabel('Günlük Tüketim (m³/gün)')
plt.legend()
```

Şekil 42. SVR için gelecek 10 günü tahmin etme ve görselleştirme

ÇIKTI:



Şekil 43. SVR için gelecek 10 günün tahmin grafiği

4.3.6. Gelecek 1 yılı (365 günü) Tahmin Etme ve Görselleştirilmesi

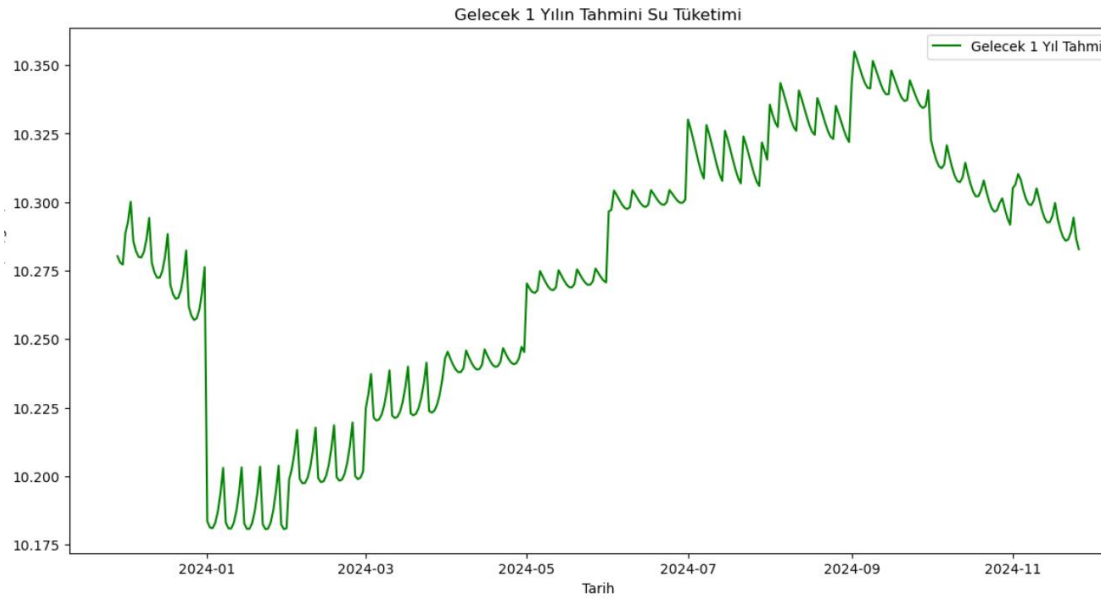
[10]:
```
from datetime import datetime, timedelta

# Gelecek 365 günün tarih aralığını belirleme
start_date = df.index[-1] + timedelta(days=1)
end_date = start_date + timedelta(days=364)
forecast_dates_1_year = pd.date_range(start=start_date, end=end_date, freq='D')

# Gelecek 1 yıl için özellikler oluşturma
forecast_features_1_year = create_features(pd.DataFrame(index=forecast_dates_1_year))

# Özellikleri normalize etme
forecast_features_normalized_1_year = scaler_X.transform(forecast_features_1_year)

# Gelecek 1 yılın su tüketimini tahmin etme
forecast_scaled_1_year = svr_model.predict(forecast_features_normalized_1_year)

# Tahminleri orijinal ölçeklere dönüştürme
forecast_1_year = scaler_y.inverse_transform(forecast_scaled_1_year.reshape(-1, 1)).flatten()

# Tahminleri ve tarihleri birleştirme
forecast_results_1_year = pd.DataFrame({'Tarih': forecast_dates_1_year, 'Tahmin Edilen Su Tüketim
forecast_results_1_year = forecast_results_1_year.set_index('Tarih')

forecast_results_1_year
```

[10]:

Tarih	Tahmin Edilen Su Tüketimi (m^3/gün)
2023-11-28	10.280274
2023-11-29	10.277941
2023-11-30	10.277192
2023-12-01	10.288719
2023-12-02	10.292950
...	...
2024-11-22	10.286374
2024-11-23	10.289057
2024-11-24	10.294379
2024-11-25	10.286559
2024-11-26	10.282824

365 rows × 1 columns

```
11]: # Tahmin edilen su tüketimini görselleştirme
plt.figure(figsize=(14, 7))
plt.plot(forecast_results_1_year.index, forecast_results_1_year['Tahmin Edilen Su Tüketimi (m³/gün)'], lab
plt.title('Gelecek 1 Yılın Tahmini Su Tüketimi')
plt.xlabel('Tarih')
plt.ylabel('Günlük Tüketim (m³/gün)')
plt.legend()
plt.show()
```



Şekil 44. SVR için gelecek 1 yılı tahmin etme ve görselleştirme

BEŞİNCİ BÖLÜM

MODEL SONUÇLARI VE TARTIŞMA

5.1. Algoritmaların Performans Karşılaştırması

Bu çalışma, su talep miktarının tahmin edilmesi amacıyla geniş bir makine öğrenimi algoritma yelpazesi kullanılarak gerçekleştirildi. Çeşitli ölçütlerle değerlendirilen algoritmaların performansları, yaygın olarak kullanılan Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) ve Mean Absolute Percentage Error (MAPE) gibi metrikler kullanılarak objektif bir biçimde değerlendirildi. Bununla birlikte, Ensemble Method in Machine Learning (Averaging) tekniği kullanılarak elde edilen sonuçlar da dikkate alındı.

MSE (Ortalama Kare Hata): Tahmin edilen değerler ile gerçek değerler arasındaki kare farkların ortalaması olarak tanımlanır. Düşük MSE değerleri, modelin verileri daha iyi tahmin ettiğini gösterir.

$$\text{MSE} = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - y_i)^2$$

Burada $\hat{y}_i$ model tarafından tahmin edilen i. örneğin değeri, y_i gerçek değer ve n örnek sayısını ifade eder.

MAE (Ortalama Mutlak Hata): Tahmin edilen değerler ile gerçek değerler arasındaki mutlak farkların ortalamasıdır. MAE, modelin hatalarının büyüklüğünü ölçer ve yüksek hata değerlerine karşı dayanıklıdır.

$$\text{MAE} = \frac{1}{n}\sum_{i=0}^{n-1}|y_i - \hat{y}_i|$$

RMSE (Karekök Ortalama Kare Hatası): RMSE, modelin tahminlerinin gerçek değerlerden ne kadar sapma gösterdiğini ölçer. RMSE′nin hesaplanması için, modelin tahmin ettiği değerler ile gerçek değerler arasındaki farkların karelerinin ortalaması alınır, ardından bu değerin karekökü alınır.

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

Burada y_i gerçek değer, $\hat{y}_i i$ model tarafından tahmin edilen değer ve nn örnek sayısını ifade eder.

MAPE (Ortalama Mutlak Yüzde Hata): MAPE, modelin tahminlerinin gerçek değerlere göre yüzde olarak ne kadar hata yaptığını ölçer. MAPE'nin hesaplanması için, her bir veri noktası için mutlak hata yüzdesinin alınması ve bunların ortalamasının bulunması gerekmektedir.

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100$$

Burada y_i gerçek değer, $\hat{y}_i$ model tarafından tahmin edilen değer ve n örnek sayısını ifade eder.

Elde edilen sonuçlar, farklı algoritmaların farklı güçlü yönlerinin olduğunu göstermektedir. Örneğin, LSTM ve MLPRegressor gibi bazı algoritmalar düşük hata değerleriyle öne çıkmış ve zaman serileri üzerindeki yetenekleriyle dikkat çekmiştir. Bu algoritmalar, özellikle uzun vadeli bağımlılıkları yakalama yetenekleriyle su talep tahmini gibi zaman serileri analizinde etkili olmuştur. Diğer yandan, SVR gibi bazı algoritmalar ise doğrusal olmayan ilişkileri modelleme yetenekleri ile önemli bir performans sergilemiştir. Bu algoritmalar, veri setindeki karmaşık ilişkileri anlamak ve modellemek için tercih edilmiştir. Bu algoritmaların detaylı incelenmesi şu şekilde gösterilmektedir:

Tablo 2. Algoritma Hata Metrikleri Tablosu

Hata Metrikleri / Algoritmalar	Mean Squared Error (MSE)	Root Mean Squared Error (RMSE)	Mean Absolute Error (MAE)	Mean Absolute Percentage Error (MAPE)	Ensemble Method in Machine Learning (Averaging)
MLPRegressor	0,0024	0,0485	0,0382	0,3716	0.00179
LSTM	0.0018	0,0425	0,0318	0,3080	
SVR (Support Vector Regression)	0,0027	0,0521	0.0416	0,4021	
KNN (K-Nearest Neighbors)	0,0033	0,0572	0,0481	0,4657	

RF (Random Forest)	0,0031	0,0554	0.0460	0,4457
BİLSTM	0,0018	0,0420	0,0318	0,3081
DECISION TREE	0,0033	0,0573	0,0469	0,4545
CATBOOST	0,0035	0,0589	0,05002	0,4841
LINEER REGRESYON	0,0049	0,0703	0,0604	0,5872
XGBoost	0,0031	0,0557	0,0475	0,4616
LightGBM	0,0035	0,0590	0,0511	0,4947
N-BEATS	0,0033	0,0576	0,0448	0,4334
ES (Exponential Smoothing)	0,0084	0,0917	0,0731	0,7050
AR	0,0227	0,1506	0,1377	1,3303
SARIMAX	0,0115	0,1074	0,0907	0,8755
ARIMA	0,0088	0,0940	0,0757	0,7308

1. MLPRegressor:

• MSE: 0.0024

• RMSE: 0.0485

• MAE: 0.0382

• MAPE: 0.3716

• Ensemble: 0.00179

• MLPRegressor, düşük MSE ve RMSE değerleri ile dikkat çeker. Bu, modelin tahminlerindeki ortalama hata miktarının oldukça düşük olduğunu gösterir. MAE ve

MAPE değerleri de rekabetçi olup, su talep tahmini konusunda oldukça doğru sonuçlar verebildiğini işaret eder.

2. LSTM:

• MSE: 0.0018

• RMSE: 0.0425

• MAE: 0.0318

• MAPE: 0.3080

• Ensemble: 0.00179

• LSTM, projedeki en düşük MSE değerine sahip olmasıyla öne çıkar. Uzun vadeli bağımlılıkları modelleme yeteneği sayesinde, su talep tahmini için ideal bir seçimdir. Düşük RMSE, MAE ve MAPE değerleri, tahminlerin doğruluğunu ve modelin güvenilirliğini gösterir.

3. SVR (Support Vector Regression):

• MSE: 0.0027

• RMSE: 0.0521

• MAE: 0.0416

• MAPE: 0.4021

• Ensemble: 0.00179

• SVR, non-lineer ilişkileri modellemede güçlü bir yeteneğe sahip olmasıyla bilinir. Projede, diğer bazı modellere göre daha yüksek hata değerleri sergilemiş olsa da karmaşık ve değişken su talebi verilerini etkili bir şekilde işleyebilmiştir.

4. KNN (K-Nearest Neighbors):

• MSE: 0.0033

• RMSE: 0.0572

• MAE: 0.0481

• MAPE: 0.4657

• KNN, lokal veri noktaları arasındaki ilişkileri kullanarak tahmin yapar. Bu model, projede diğerlerine kıyasla daha yüksek hata değerleriyle performans göstermiş, bu da modelin veri setinin özelliklerine tam olarak uyum sağlayamadığını gösterir.

5. RF (Random Forest):

• MSE: 0.0031

• RMSE: 0.0554

• MAE: 0.0460

• MAPE: 0.4457

• Random Forest, birçok karar ağacının tahminlerini birleştirerek çalışır. MSE, RMSE, MAE ve MAPE değerleri ortalama seviyededir, bu da modelin stabil ancak sınırlı performans sergilediğini belirtir.

6. BiLSTM:

• MSE: 0.0018

• RMSE: 0.0420

• MAE: 0.0318

• MAPE: 0.3081

• BiLSTM, LSTM'nin bir varyasyonu olarak, iki yönlü yapı sayesinde hem geçmiş hem de gelecek verileri dikkate alarak tahmin yapma kapasitesine sahiptir. Bu özellikleriyle, su talep tahmininde özellikle etkili olmuştur.

7. Decision Tree:

• MSE: 0.0033

• RMSE: 0.0573

• MAE: 0.0469

• MAPE: 0.4545

• Karar ağaçları, veri setindeki özellikler arasındaki karar kurallarını öğrenerek çalışır. Ancak bu modelin yüksek hata değerleri, aşırı uyum veya veri setinin karmaşıklığıyla başa çıkma konusunda sınırlı olabileceğini gösterir.

8. CatBoost:

• MSE: 0.0035

• RMSE: 0.0589

• MAE: 0.05002

• MAPE: 0.4841

• CatBoost, kategorik verilerle çalışmak için optimize edilmiş bir gradient boosting algoritmasıdır. Su talebi tahmininde, diğer bazı modellere göre daha yüksek hata değerleriyle orta düzeyde bir performans sergilemiştir.

9. Linear Regression:

• MSE: 0.0049

• RMSE: 0.0703

• MAE: 0.0604

• MAPE: 0.5872

• Lineer regresyon, en basit tahmin modellerinden biridir ve veri setindeki doğrusal ilişkileri modelleme yeteneğine sahiptir. Ancak bu proje için yüksek hata değerleri, modelin su talebi verilerinin karmaşıklığını yeterince modelleyemediğini göstermektedir.

10. XGBoost:

• MSE: 0.0031

• RMSE: 0.0557

• MAE: 0.0475

• MAPE: 0.4616

• XGBoost, güçlü bir gradient boosting modeli olup, genellikle yüksek performanslı tahminler sunar. Ancak bu projede orta seviye hata değerleriyle karışık sonuçlar vermiştir.

11. LightGBM:

• MSE: 0.0035

• RMSE: 0.0590

• MAE: 0.0511

• MAPE: 0.4947

• LightGBM, hafif bir gradient boosting modeli olup, büyük veri setlerinde hızlı çalışmasıyla bilinir. Ancak su talebi tahmininde gösterdiği performans, diğer bazı modellere kıyasla daha düşük olmuştur.

12. N-BEATS:

• MSE: 0.0033

• RMSE: 0.0576

• MAE: 0.0448

• MAPE: 0.4334

• N-BEATS, derin öğrenme tabanlı bir zaman serisi tahmin modelidir ve genellikle dikkate değer sonuçlar sunar. Ancak bu projede, beklenen düzeyde performans gösterememiştir.

13. ES (Exponential Smoothing):

• MSE: 0.0084

• RMSE: 0.0917

• MAE: 0.0731

• MAPE: 0.7050

• Üstel düzeltme, zaman serileri verileri için klasik bir tahmin yöntemi olup, bu projede diğer modellere göre oldukça yüksek hata değerleriyle düşük performans sergilemiştir.

14. AR (Autoregression):

• MSE: 0.0227

• RMSE: 0.1506

• MAE: 0.1377

• MAPE: 1.3303

• Otoregresyon, zaman serisi verilerinde geçmiş değerlerin gelecek değerleri tahmin etmede kullanılır. Bu model, projede oldukça yüksek hata değerleri ile sınırlı bir başarı göstermiştir.

15. SARIMAX:

• MSE: 0.0115

• RMSE: 0.1074

• MAE: 0.0907

• MAPE: 0.8755

• SARIMAX, mevsimsel etkileri dikkate alan gelişmiş bir ARIMA modelidir. Bu projede, diğer modellere göre daha yüksek hata değerleri ile orta düzeyde bir performans sergilemiştir.

16. ARIMA:

• MSE: 0.0088

• RMSE: 0.0940

• MAE: 0.0757

• MAPE: 0.7308

• ARIMA, zaman serisi analizinde yaygın olarak kullanılan bir modeldir. Ancak bu projede yüksek hata oranları ile sınırlı bir etkililik göstermiştir.

Bu sonuçlar, su talep tahmini gibi gerçek dünya problemlerinde farklı algoritmaların kullanılması gerektiğini vurgulamaktadır. Her bir algoritmanın kendine özgü avantajları ve dezavantajları bulunmaktadır ve doğru algoritmanın seçilmesi, doğru tahminlerin yapılması için hayati önem taşımaktadır. Bu nedenle, algoritmaların performansının objektif bir biçimde değerlendirilmesi ve uygun olanın seçilmesi, makine öğrenimi tabanlı tahmin problemlerinde başarının anahtarıdır.

5.2. En İyi Algoritmaların Seçimi ve Gerekçeleri

LSTM: LSTM, uzun vadeli bağımlılıkları yakalama yeteneği ile öne çıkar. Bu özelliği, zaman serileri üzerindeki tahmin problemleri için ideal bir seçenek olmasını sağlar. Bu çalışmada da LSTM'nin düşük MSE, RMSE ve MAPE değerleri ile dikkat

çektiği gözlemlendi. Özellikle, uzun vadeli trendleri ve desenleri tanımlama konusundaki başarısı, su talep tahmini gibi dinamik ve değişken veri setleri üzerinde etkili bir performans sergilemesini sağlar. Ayrıca, Ensemble Method kullanılarak elde edilen sonuçlar da diğer algoritmalarla benzerlik gösterir, bu da LSTM'nin stabil ve güvenilir bir tahmin modeli olarak değerlendirilmesini sağlar.

MLPRegressor: MLPRegressor, geniş veri setlerinde esneklik ve uyumluluk sağlayabilen bir yapay sinir ağı modelidir. Bu model, karmaşık ilişkileri ve non-lineer yapıları modelleme konusunda güçlüdür. Bu çalışmada da MLPRegressor'ın düşük hata değerleri ile dikkat çektiği gözlendi. Özellikle, MAE ve MAPE gibi ölçütlerde diğer algoritmaların önüne geçtiği görüldü. MLPRegressor, veri setinin özelliklerini daha iyi yakalayabilmesi ve esnek mimarisi sayesinde farklı veri yapılarına uyum sağlayabilmesiyle tercih edilen bir tahmin modeli olarak öne çıkar.

SVR: SVR, genellikle sınıflandırma problemleri için kullanılan bir algoritma olsa da bu çalışmada su talep tahmini için de etkili bir şekilde kullanıldı. SVR'in doğrusal olmayan ilişkileri modelleme yeteneği, bu algoritmayı su talep tahmini probleminde kullanışlı bir seçenek haline getirir. Diğer algoritmalara kıyasla daha yüksek hata değerlerine sahip olsa da genel olarak iyi bir performans sergilediği gözlendi. SVR, özellikle karmaşık veri setlerinde ve sınıflar arasındaki sınırları net bir şekilde belirleyebildiği durumlarda etkilidir. Bu özellikleriyle SVR, su talep tahmini gibi dinamik ve karmaşık veri setleri üzerinde güvenilir sonuçlar sağlayabilir.

Bu üç algoritma, farklı veri yapıları ve tahmin gereksinimleri göz önüne alındığında en uygun seçenekler olarak belirlendi. Ancak, son seçimin belirlenmesi, kullanılan veri setinin özelliklerine, tahmin gereksinimlerine ve hesaplama gücüne bağlı olabilir. Bu nedenle, ilgili bağlamda en uygun algoritmanın seçilmesi, doğru tahminlerin yapılması için kritik öneme sahiptir. Doğru algoritmanın seçilmesi, tahmin modelinin güvenilirliği ve başarısı açısından hayati öneme sahiptir.

ALTINCI BÖLÜM

WEB SİTESİ ENTEGRASYONU

6.1. Tahmin Modelinin Web Sitesine Entegrasyon Süreci

Projede kullanılan LSTM modeli, İstanbul şehrinin günlük su tüketim tahminleri için özel olarak seçilmiştir. LSTM, rekürrent sinir ağlarının bir alt türü olarak, zaman serisi verilerindeki uzun vadeli bağımlılıkları etkili bir şekilde modelleme yeteneği ile öne çıkar. Bu model, dinamik ve sürekli değişen veri desenleri arasındaki ilişkileri koruyarak yüksek doğrulukta tahminler sunar, bu nedenle su talebi tahminleri gibi uygulamalar için idealdir.

Teknik Detaylar ve Modelin Seçilme Gerekçeleri: LSTM modeli, veri noktaları arasındaki bağlantıları uzun süre boyunca saklayabilen hafıza hücreleri ile donatılmıştır. Modelin başlıca özellikleri şunlardır:

Hafıza Hücresi: Veriler arasındaki uzun vadeli ilişkileri depolar ve gerektiğinde bu bilgilere erişim sağlar.

Giriş Kapısı: Modelin alacağı bilgileri belirler ve gereksiz verilerin hafıza hücresine girmesini önler.

Çıkış Kapısı: Hangi bilgilerin sonraki katmana aktarılacağını ve model çıktısının ne olacağını kontrol eder.

Unutma Kapısı: Eski bilgilerin ne kadarının korunacağını veya unutulacağını yönetir, bu sayede modelin gereksiz yere bilgi yüklenmesinin önüne geçilir.

Bu yapısal özellikler, LSTM'nin su talebi gibi değişken veriler üzerinde yüksek doğrulukta tahminler yapabilmesini sağlar. Model, MSE, RMSE ve MAPE gibi hata metriklerinde düşük değerler göstermiş, bu da modelin yüksek tahmin başarısını kanıtlamaktadır.

Entegrasyon Süreci: LSTM modelinin Streamlit tabanlı web uygulamasına entegrasyon süreci şu adımları içermektedir:

Modelin Eğitimi: LSTM modeli, İstanbul'a ait tarihsel su tüketim verileri kullanılarak eğitilmiştir. Model eğitimi sırasında hücre sayısı, epoch sayısı ve batch boyutu gibi parametreler ayarlanarak optimal sonuçlar elde edilmiştir. Python

ortamında hazırlanan bu model, veri setinin zamansal dinamiklerini anlamada kritik bir rol oynamıştır.

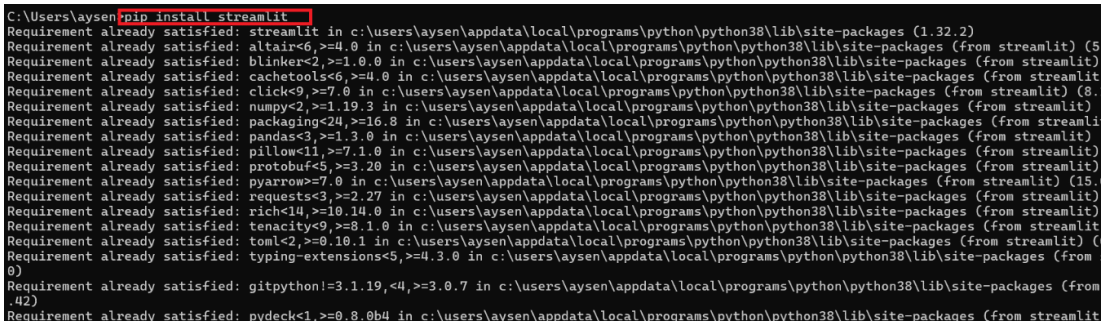
Modelin Serializasyonu: Eğitilen model, pickle veya joblib kütüphaneleri kullanılarak serialize edilmiş ve disk üzerinde saklanmıştır. Bu işlem, modelin web uygulamasına entegre edilmesi sürecini basitleştirir.

Streamlit ile Entegrasyon: Serialize edilmiş model, Streamlit uygulamasına yüklenmiştir. Uygulama, REST API aracılığıyla kullanıcı girdilerini alır ve LSTM modelini kullanarak bu girdilere dayalı olarak su tüketim tahminleri üretir. Kullanıcı arayüzü üzerinden sağlanan tarih gibi girdiler, model tarafından işlenir ve tahmin sonuçları anında kullanıcıya sunulur.

6.2. Kullanıcı Arayüzü Tasarımı ve Özellikleri

Bu bölüm, LSTM modelinin Streamlit web uygulamasına entegrasyonu sonrası oluşturulan kullanıcı arayüzünün tasarımını ve özelliklerini detaylandırmaktadır. Kullanıcı arayüzü, veri görselleştirme, interaktif kontrol elemanları ve kullanıcı dostu navigasyon özellikleri ile zenginleştirilmiştir.

6.2.1. Kurulum Süreci



Şekil 45. Web için kurulum süreci

Bu fotoğrafta görülen ekran, projenin temel bileşeni olan Streamlit kütüphanesinin kurulum sürecini göstermektedir. Streamlit, Python kullanılarak hızlı ve etkili bir şekilde interaktif web uygulamaları geliştirmek için tasarlanmış bir kütüphanedir. Bu kütüphane, veri görselleştirme ve kullanıcı etkileşimini kolaylaştırarak, veri bilimcilerin ve geliştiricilerin prototiplerini hızlıca test etmelerine olanak tanır.

Komut: pip install streamlit

Amaç: Bu komut, Streamlit kütüphanesinin ve onun bağımlılıklarının kullanıcının sistemine yüklenmesi işlemidir. Kurulum komutu, kullanıcının bilgisayarının komut istemcisine yazılarak çalıştırılır. Eğer kütüphane zaten yüklü ise, sistem "Requirement already satisfied" mesajıyla bu durumu bildirir ve ek kuruluma gerek kalmaz.

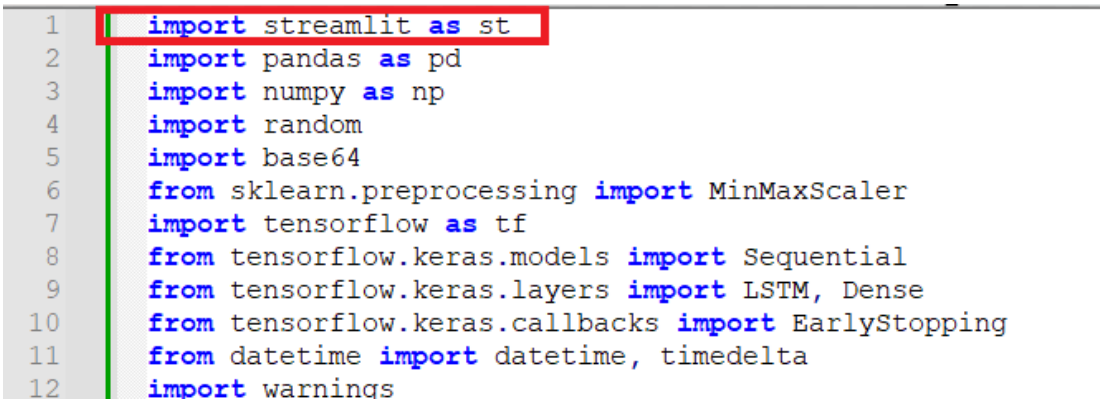
Kurulumun Önemi: Streamlit kütüphanesi, veri bilimciler ve geliştiriciler için, analitik uygulamaları ve makine öğrenimi modellerini web tabanlı bir arayüzde hızlıca sunma imkanı sağlar. Bu, özellikle dinamik veri setlerinin ve modellerin son kullanıcılarla etkileşimli bir biçimde paylaşılmasını kolaylaştırır.

Kurulum sürecinin başarılı bir şekilde tamamlanması, geliştirme sürecinin diğer aşamaları için zemin hazırlar ve projenin tüm bağımlılıklarının çözümlenmesini garantiler, böylece yazılım çakışmaları veya uyumsuzluk hatalarının önüne geçilir.

Bu adım, projenin başlangıç aşamasında önemli bir yer tutar ve projenin ilerleyen kısımlarında kullanılacak olan Streamlit özelliklerinin sorunsuz bir şekilde çalışmasını sağlar. Kurulumun başarılı bir şekilde yapılması, uygulamanın geliştirilmesi ve sonrasında kullanıcılara sunulmasındaki akıcılığın ve verimliliğin temelini oluşturur. Bu süreç, kullanıcı arayüzünün tasarımı ve işlevselliği için kritik öneme sahiptir, zira interaktif özellikler ve veri görselleştirmeleri doğrudan bu kütüphane üzerinden yönetilir ve optimize edilir.

6.2.2. Uygulama Geliştirme Süreci ve Kütüphanelerin Entegrasyonu

Streamlit kurulumu tamamlandıktan sonra, uygulamanın ana kod dosyasına çeşitli Python kütüphaneleri ve modüller import edilmiştir. Bu kütüphaneler, veri işleme, modelleme ve arayüz oluşturma süreçlerinde temel bileşenler olarak görev yapar. Her bir kütüphanenin uygulamadaki amacını ve işlevini aşağıda detaylı bir şekilde inceleyelim:



Şekil 46. Web için kütüphanelerin entegrasyonu

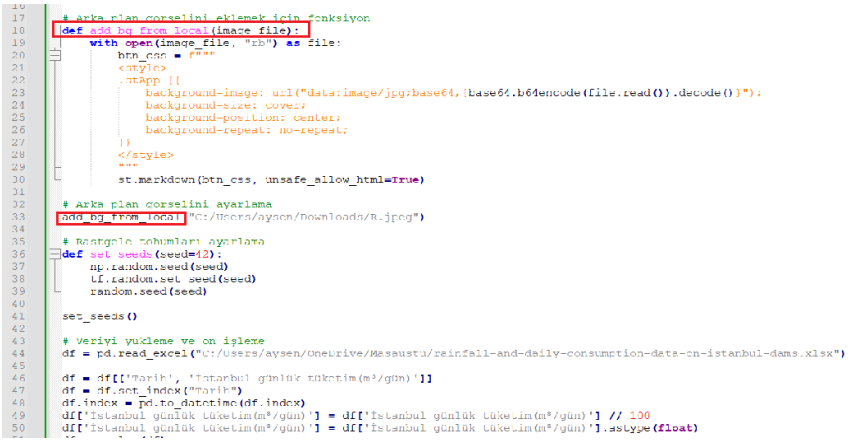
- Streamlit (import streamlit as st): Kullanıcı arayüzünün oluşturulmasında kullanılan temel kütüphane. Streamlit, veri bilimcilerin ve geliştiricilerin interaktif dashboard'lar ve uygulamalar oluşturmasını sağlar, bu da kullanıcıların uygulama içerisinde verilerle dinamik bir şekilde etkileşimde bulunmasına olanak tanır.
- Pandas (import pandas as pd): Veri manipülasyonu ve analizi için kullanılan bir kütüphane. Bu projede, su tüketim verilerinin işlenmesi, analizi ve hazırlanması süreçlerinde kullanılır.
- NumPy (import numpy as np): Bilimsel hesaplamalar için kullanılan bir kütüphane. Veri setlerindeki sayısal işlemler için gerekli olan fonksiyonaliteyi sağlar.
- Random: Rastgele sayı üretimi için kullanılan modül. Model eğitimi sırasında rastgelelik unsurlarını kontrol etmek amacıyla kullanılır.
- Base64: İkili verileri ASCII karakter dizisine çevirmek için kullanılır. Uygulamada, görsellerin veya diğer ikili dosyaların kodlanması ve streamlit arayüzüne entegre edilmesi için kullanılabilir.
- Scikit-Learn (from sklearn.preprocessing import MinMaxScaler): Veri ölçekleme ve normalleştirme işlemleri için kullanılan MinMaxScaler, modelin daha etkili öğrenmesi için girdi verilerinin 0 ile 1 arasında ölçeklenmesini sağlar.

- TensorFlow ve Keras: Derin öğrenme modellerinin geliştirilmesi ve eğitilmesi için kullanılan kütüphaneler. Bu projede, LSTM modelinin oluşturulması ve eğitilmesi (from tensorflow.keras.models import Sequential ve from tensorflow.keras.layers import LSTM, Dense) için kullanılır. Ayrıca, model eğitimi sırasında erken durdurma (from tensorflow.keras.callbacks import EarlyStopping) mekanizması gibi optimizasyonlar için de kullanılır.
- Datetime: Tarih ve zaman işlemleri için kullanılan modül. Veri setlerindeki zaman serisi verilerinin işlenmesi ve manipüle edilmesinde önemli bir role sahiptir.

Bu kütüphaneler ve modüller, projenin veri yükleme, işleme, modelleme ve kullanıcı arayüzü geliştirme gibi çeşitli aşamalarında kritik öneme sahiptir. Her biri, su tüketim tahminleri uygulamasının başarılı bir şekilde geliştirilmesi ve kullanıcıya sunulması için özel işlevler yerine getirir. Bu bütünleşik yapı, uygulamanın verimli, etkileşimli ve teknik olarak donanımlı bir şekilde çalışmasını sağlar.

6.2.3. Arka Plan Görselinin Entegrasyonu ve Estetik Katkısı

Streamlit uygulamasının kullanıcı arayüzü, görsel çekiciliği artırmak amacıyla özenle tasarlanmıştır. Bu kapsamda, uygulama arka planına bir görsel eklenmesi, estetik bir değer katmakla kalmayıp, kullanıcı deneyimini de zenginleştirir. Bu sürecin teknik detayları, add_bg_from_local fonksiyonu kullanılarak gerçekleştirilir.

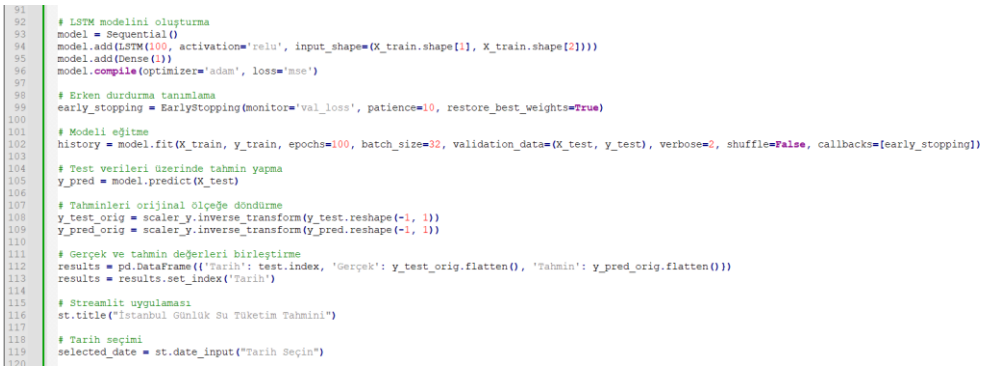


Şekil 47. Web için arka plan görsellerinin entegrasyonu

- Fonksiyon: add_bg_from_local(image_file)
    - Kod Açıklaması: Bu fonksiyon, lokal bir dosyadan arka plan için görsel okur ve bu görseli base64 formatına çevirerek Streamlit arayüzünde kullanılabilir hale getirir. Görselin base64 kodlaması sayesinde, görsel dosyanın doğrudan web uygulamasına gömülmesi mümkün olur ve harici bağlantılara gerek kalmaz.
- CSS Stilleri:
    - Kod: Görselin stil ayarları içerisinde, görselin nasıl görüneceğine dair çeşitli CSS özellikleri tanımlanır. Özellikle background-size: cover; görselin arka planı tamamen kaplamasını sağlar ve herhangi bir tekrarın olmaması için background-repeat: no-repeat; kullanılır. background-position: center; ayarı ise görselin merkezde konumlanmasını garantiler.
- Uygulama İçi Kullanım:
    - Fonksiyon Kullanımı: add_bg_from_local("C:/Users/ayşen/Downloads/R.jpeg") Bu satır, belirtilen dosya yolundan görseli okur ve uygulamanın arka planında kullanmak üzere ayarlar.
    - Görsel Katkısı: Arka plan görselinin kullanılması, kullanıcının uygulamayla etkileşimini daha keyifli ve görsel olarak tatmin edici hale getirir. Bu, özellikle uzun süreli kullanımlarda kullanıcı yorgunluğunu azaltabilir ve uygulamanın genel çekiciliğini artırır.

6.2.4. Modelin Eğitimi ve Performans Analizi

Bu bölüm, LSTM tabanlı derin öğrenme modelinin geliştirilmesi, eğitimi ve tahmin performansının değerlendirilmesi süreçlerini kapsamaktadır. Proje kapsamında kullanılan TensorFlow ve Keras kütüphaneleri sayesinde, zaman serileri verileri üzerinden su tüketim tahminleri yapmak üzere bir model oluşturulmuş ve bu model, Streamlit platformu üzerinde görselleştirilerek son kullanıcıya sunulmuştur.



```
# LSTM modelini oluşturma
model = Sequential()
model.add(LSTM(100, activation='relu', input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')

# Erken durdurma tanımlama
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)

# Modeli eğitme
history = model.fit(X_train, y_train, epochs=100, batch_size=32, validation_data=(X_test, y_test), verbose=2, shuffle=False, callbacks=[early_stopping])

# Test verileri üzerinde tahmin yapma
y_pred = model.predict(X_test)

# Tahminleri orijinal ölçeğe döndürme
y_test_orig = scaler_y.inverse_transform(y_test.reshape(-1, 1))
y_pred_orig = scaler_y.inverse_transform(y_pred.reshape(-1, 1))

# Gerçek ve tahmin değerleri birleştirme
results = pd.DataFrame({'Tarih': test.index, 'Gerçek': y_test_orig.flatten(), 'Tahmin': y_pred_orig.flatten()})
results = results.set_index('Tarih')

# Streamlit uygulaması
st.title("İstanbul Günlük Su Tüketim Tahmini")

# Tarih seçimi
selected_date = st.date_input("Tarih Seçin")
```

Şekil 48. Web için model eğitimi ve performans analizi

- Model Yapılandırması: Sequential model yapısı kullanılarak, veri akışının sıralı bir katman dizisi olarak işlenmesi sağlanmıştır. LSTM katmanı, bu yapının temel bileşeni olarak, uzun vadeli bağımlılıkları etkili bir şekilde öğrenmek üzere entegre edilmiştir. Dense (1) katmanı ile modelin çıkış katmanı tanımlanmış, bu da modelin her bir girdi için tek bir çıktı değeri üretmesini sağlar.
- Derleme ve Optimize Etme: Model, adam optimizatörü ve mean squared error kayıp fonksiyonu kullanılarak derlenmiştir. Adam optimizatörü, adaptif öğrenme oranları sağlayarak eğitim sürecinde daha hızlı ve etkili konverjans sunar. MSE, tahmin edilen değerler ile gerçek değerler arasındaki farkların karesinin ortalaması alınarak modelin ne kadar iyi performans gösterdiğini ölçer.
- Erken Durdurma Mekanizması: Eğitim sırasında EarlyStopping mekanizması devreye sokularak modelin aşırı uyum (overfitting) riskine karşı korunması amaçlanmıştır. Bu yöntem, doğrulama veri seti üzerindeki kaybın belirli bir sayıda epoch boyunca iyileşmeme durumunda model eğitiminin otomatik olarak durdurulmasını sağlar.
- Model Eğitimi: model.fit() fonksiyonu ile model, belirtilen parametreler (örneğin, batch_size=32, epochs=100) kullanılarak eğitilmiştir. Bu süreçte model, eğitim veri setini kullanarak öğrenir ve doğrulama seti üzerinden performansını sürekli olarak değerlendirir.
- Tahmin ve Görselleştirme:

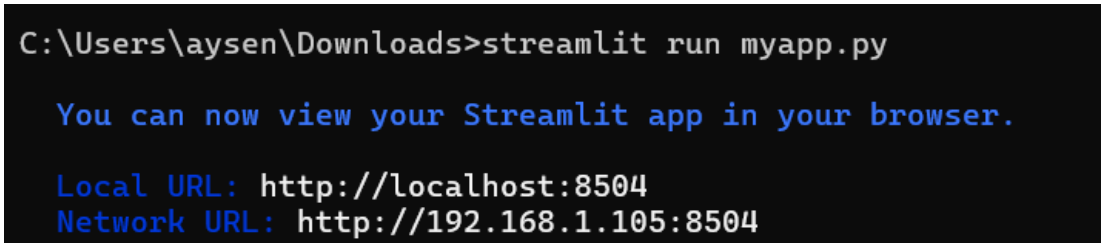
- Tahmin İşlemi: model.predict() fonksiyonu, eğitilmiş modeli kullanarak test veri seti üzerinden su tüketim tahminleri yapar. Elde edilen tahminler, gerçek değerler ile karşılaştırılır.
- Sonuçların Görselleştirilmesi: Streamlit uygulaması, tahmin sonuçlarını ve karşılaştırmalı analizleri kullanıcıya sunar. Bu sunum, pd.DataFrame kullanılarak organize edilen verilerin, zaman serisi grafikleri olarak görselleştirilmesini içerir. Kullanıcılar bu sayede, modelin günlük su tüketimi tahminlerinin doğruluğunu kolayca değerlendirebilir.

Uygulamanın Yayınlanması ve Erişim Sağlanması

Bu bölüm, geliştirilen Streamlit uygulamasının nasıl başlatıldığı ve son kullanıcılar tarafından nasıl erişilebilir hale getirildiği üzerine odaklanmaktadır. Geliştirme sürecinin son adımı olan bu aşama, uygulamanın kullanıma sunulmasını içerir ve kullanıcıların uygulamayı etkileşimli bir şekilde deneyimlemelerini sağlar.

6.2.5. Uygulamanın Yayınlanması ve Erişim Sağlanması

Bu bölüm, geliştirilen Streamlit uygulamasının nasıl başlatıldığı ve son kullanıcılar tarafından nasıl erişilebilir hale getirildiği üzerine odaklanmaktadır. Geliştirme sürecinin son adımı olan bu aşama, uygulamanın kullanıma sunulmasını içerir ve kullanıcıların uygulamayı etkileşimli bir şekilde deneyimlemelerini sağlar.



Şekil 49. Web yayınlanması ve erişim sağlanması

Uygulamanın Başlatılması:

- Komut: streamlit run myapp.py
    - Bu komut, geliştirilen Streamlit uygulamasını yerel bir sunucuda çalıştırır. Uygulama, yazılım geliştiricisi tarafından belirlenen komut

dosyası (myapp.py) kullanılarak başlatılır. Bu dosya, uygulamanın tüm işlevlerini, kullanıcı arayüzünü ve veri işleme mantığını içerir.

- Erişim Linkleri:
    - Lokal URL: http://localhost:8504
        - Bu URL, uygulamayı yerel olarak çalıştıran bilgisayarda erişim sağlamak için kullanılır. Geliştirici veya yerel ağ üzerindeki kullanıcılar bu adresi web tarayıcılarında açarak uygulamaya doğrudan erişebilirler.
    - Ağ URL'i: http://192.168.1.105:8504
        - Ağ URL'i, uygulamaya aynı yerel ağ üzerindeki diğer cihazlar tarafından erişilmesine olanak tanır. Bu, özellikle ofis ortamlarında veya birden fazla cihazın aynı ağa bağlı olduğu durumlarda uygulamayı test etmek veya demo yapmak için kullanışlıdır.

Uygulamanın Yayınlanmasının Önemi:

Bu adım, geliştirilen uygulamanın pratik kullanımının test edilmesi ve doğrulanması için kritik öneme sahiptir. Kullanıcılar, verilen URL'ler üzerinden uygulamaya erişerek, su tüketim tahminleri gibi işlevleri gerçek zamanlı olarak deneyimleyebilirler. Bu, uygulamanın kullanıcı dostu olup olmadığını, kullanıcı arayüzünün etkileşimli ve anlaşılır olup olmadığını değerlendirme fırsatı sunar. Ayrıca, gerçek kullanıcı geri bildirimleri toplamak ve uygulamanın performansını ölçmek için de bir platform sağlar.

Bu süreç, tez çalışmasının uygulamalı yönünü tamamlayarak, geliştirilen teorik modellerin ve analizlerin pratikte nasıl işlediğini gösterir. Son kullanıcıların uygulamayı kullanma deneyimleri, projenin başarısını doğrudan etkileyen faktörler arasındadır ve bu deneyimler, gelecekteki geliştirmeler için değerli veriler sunar.

6.3. Web Sitesinin Performansı ve Geri Bildirimler

Bu bölümde, geliştirilen "İstanbul Günlük Su Tüketim Tahmini" web uygulamasının genel performansı ve kullanıcı geri bildirimleri üzerine

odaklanılmaktadır. Web sitesi, kullanıcıların İstanbul için günlük su tüketim tahminlerini görebileceği interaktif bir platform olarak tasarlanmıştır.

Şekil 50. Web sitesi görseli ana ekran

6.3.1. Web Sitesinin Genel Görünümü ve Kullanıcı Etkileşimi:

- Tasarım ve Görsel Unsurlar:
    - Web sitesi, suyun vizüel temalarıyla uyumlu bir arka plana sahiptir. Temiz ve ferah bir görsel, suyun canlılığını ve önemini vurgulamak için kullanılmıştır. Bu tasarım, site ziyaretçilerine su tüketimi ve yönetimi konularının önemini sürekli hatırlatır.
    - Arayüz, kullanıcı dostu bir yapıda kurulmuş olup, tarih seçimi ve tahminleri görüntüleme gibi işlemler kolayca yapılabilecek şekilde düzenlenmiştir. Kullanıcıların istedikleri tarih için su tüketim tahminlerini birkaç tıklama ile hızlıca görebilmeleri sağlanmıştır.

6.3.2. Performans Özellikleri:

Responsive tasarım sayesinde, web sitesi farklı cihazlar ve ekran boyutlarına uyum sağlayarak geniş bir kullanıcı kitlesine erişim imkânı sunar.

6.3.3. Tarih Seçimi ve Kullanıcı Etkileşimi

Web sitesi, "İstanbul Günlük Su Tüketim Tahmini" başlığı altında, kullanıcıların İstanbul için belirli bir tarih seçerek su tüketim tahminlerini sorgulayabilecekleri

interaktif bir arayüz sunar. Bu özellik, kullanıcıların ihtiyaçlarına hızlı ve doğrudan yanıt vermek üzere tasarlanmıştır.

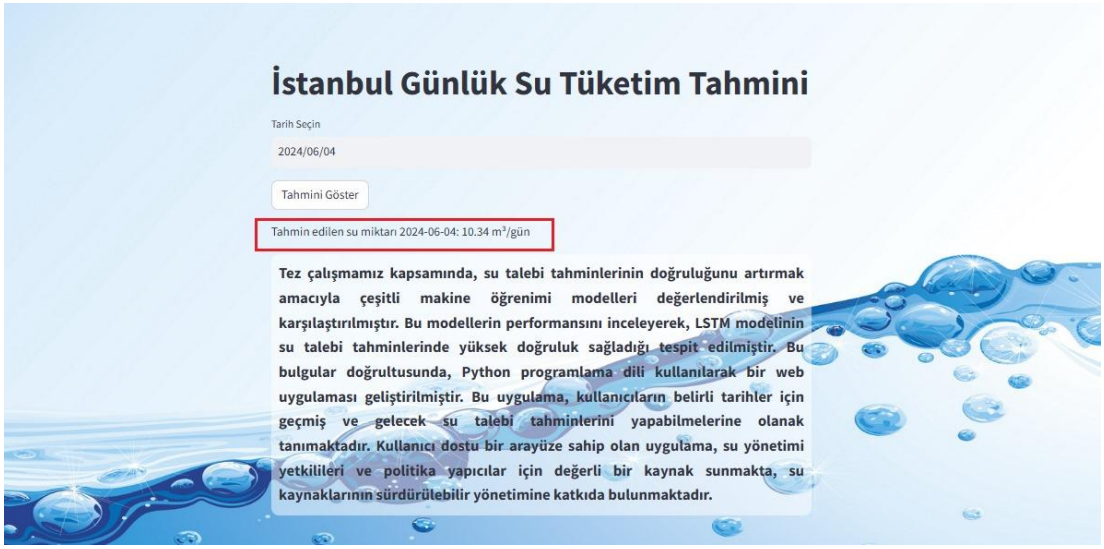
Şekil 51. Web sitesi görseli tarih seçimi

Tarih Seçim Arayüzü:

- Fonksiyonellik:
    - Web sitesi üzerinde yer alan tarih seçici, kullanıcılara geniş bir takvim arayüzü sunar. Kullanıcılar, bu arayüz yardımıyla spesifik bir tarihe kolayca erişebilir ve o gün için su tüketim tahminlerini görebilirler.
    - Tarih seçici, kullanıcıların geçmiş verilere dayalı analiz yapabilmeleri için esnek bir zaman dilimi seçimi yapmalarına olanak tanır. Bu, özellikle uzun vadeli planlama ve analizler için değerli bir araçtır.

6.3.4. Tahmin Sonuçlarının Sunumu ve Kullanıcı Etkileşimi

Son bölümde, "İstanbul Günlük Su Tüketim Tahmini" web sitesinin tahmin sonuçlarını nasıl sunduğu ve kullanıcılara bu bilgileri nasıl etkileşimli bir şekilde ilettiği incelenmektedir. Web sitesi, seçilen tarih için su tüketim tahminlerini gösteren bir arayüzle kullanıcıların bilgiye kolay ve hızlı erişimini sağlar.



Şekil 52. Web sitesi görseli sonuç gösterme

6.3.5. Tahmin Sonuçlarının Gösterimi:

Web sitesi, kullanıcının seçtiği tarih için yapılan su tüketim tahminini "Tahmin edilen su miktarı" etiketi altında gösterir. Bu sonuç, milimetreküp/gün (m^3/gün) birimiyle sunularak, kullanıcılara belirli bir tarihte İstanbul'da tahmin edilen toplam su tüketim miktarı hakkında net bir bilgi verir.

Gösterilen sonuç, tahminin yapıldığı tarihi ve saati de içerir, bu da tahminin güncelliği ve relevansı konusunda kullanıcılara güven verir.

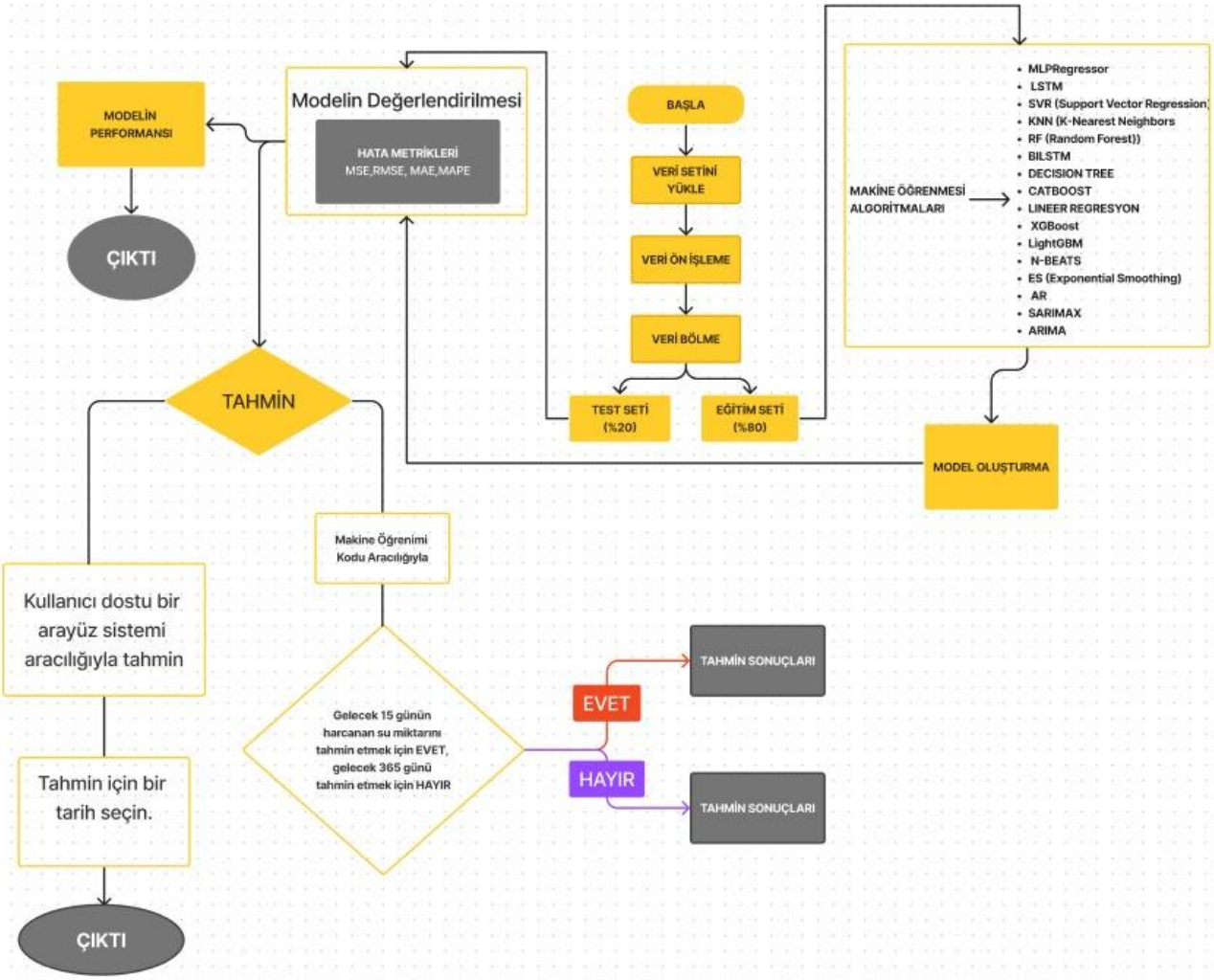
6.4. Kullanıcı Deneyimi ve Etkileşim:

Sonuçların arayüz üzerinde net ve anlaşılır bir formatla sunulması, kullanıcının deneyimini önemli ölçüde iyileştirir. Bu, kullanıcıların bilgileri hızla işlemelerine ve gerektiğinde karar verme süreçlerinde bu tahminleri kullanmalarına olanak tanır.

Tahmin sonuçları, su yönetimi ve planlama konularında ilgili bireyler ve kurumlar için değerli bir kaynak oluşturur. Kullanıcılar, bu bilgileri potansiyel su tasarrufu stratejileri geliştirmek, su kaynaklarını daha etkin kullanmak ve su yönetimi politikalarını şekillendirmek için kullanabilirler.

YEDİNCİ BÖLÜM

AKIŞ DİYAGRAMI



Şekil 53. Proje için akış diyagramı

SEKİZİNCİ BÖLÜM

SONUÇ VE ÇIKARIMLAR

8.1. Çalışmanın Özeti ve Bulguların Değerlendirilmesi

Bu tez çalışması, bir bölgedeki su talep miktarını tahmin etmek amacıyla geniş bir yelpazede makine öğrenimi algoritmaları kullanarak yapılmıştır. Toplamda 17 farklı algoritma kullanılarak gerçekleştirilen tahminler, çeşitli hata metrikleri (MSE, RMSE, MAE, MAPE) ile değerlendirilmiş ve bu süreçte özellikle LSTM, MLPRegressor ve SVR algoritmaları düşük hata değerleri ile öne çıkmıştır. Bulgular, algoritmaların su talebi tahminlerindeki etkinliğini ve potansiyel uygulama alanlarını belirlemede önemli rol oynamıştır.

Algoritmalar arasında LSTM, zaman serisi verilerinde uzun vadeli bağımlılıkları etkin bir şekilde modelleyebilmesiyle dikkat çekerken, MLPRegressor karmaşık non-lineer ilişkileri anlamada ve modellemede başarılı olmuştur. SVR ise, non-lineer ilişkileri modelleme yeteneği ile su talebi tahmininde güçlü bir alternatif sunmuştur. Bu üç algoritma, düşük hata oranları ve yüksek tahmin başarılarıyla projenin en değerli bulgularını oluşturmuştur.

8.2. Sonuçların Önemi ve Uygulama Alanları

Elde edilen sonuçlar, su kaynaklarının yönetimi ve planlaması konularında büyük önem taşımaktadır. Araştırma sonuçları, makaleler, konferans bildirileri ve kitap bölümleri aracılığıyla yayınlanarak bilimsel literatüre katkıda bulunacak ve bu süreç, su yönetimi ve yapay zeka/makine öğrenmesi alanında bilimsel ilerlemeyi destekleyecektir. Bu çalışmalar, su kaynakları yönetimi ve sürdürülebilirlik konularında önemli veri ve bulgular sağlayarak ilgili araştırmaları ilerletecek ve "Yapay Zekâ Tabanlı Su Kaynaklarının Entegre Yönetimi" projesinin geliştirilmesine katkı sağlayacaktır.

Ayrıca, bu sonuçlar, su kaynaklarının yönetimini geliştirecek politikaların oluşturulmasında ve uygulanmasında etkili olabilir. Özellikle belediyeler, su idareleri ve çevresel düzenleyici kurumlar için kritik bilgiler sunarak, su kaynaklarının daha etkin yönetilmesi, su krizlerinin önceden tahmin edilmesi ve gerekli önlemlerin zamanında alınması için stratejik planlamalarda kullanılabilir.

8.3. Akademik ve Teknolojik Etkiler

Projeden elde edilen bulgular, yüksek lisans ve doktora öğrencileri için değerli araştırma olanakları sunacak ve su kaynakları yönetimi, yapay zekâ ve makine öğrenmesi alanlarında yeni araştırma projelerinin temelini oluşturacak. Özellikle, proje sonuçlarından yararlanılarak geliştirilecek yeni modeller ve algoritmalar, akademik çevrelerde ve teknoloji sektöründe daha geniş bir etki yaratabilir. Ayrıca, bu araştırma, su yönetimi sektöründe teknolojik ilerlemeyi teşvik edebilir ve bu alandaki iş fırsatlarını genişletebilir.

8.4. Uygulama Alanları ve Toplumsal Etkiler

Su yönetimi teknolojilerindeki ilerlemeler, tarımsal sulama, kentsel planlama, altyapı yatırımları ve enerji üretimi gibi çeşitli alanlarda uygulanabilir. Özellikle hidroelektrik santrallerin operasyonlarının optimizasyonu, su talebi tahminlerinin doğruluğu ile doğrudan ilişkilidir ve bu sayede enerji üretim maliyetleri düşürülebilir. Ayrıca, bu teknolojilerin toplumsal ve ekonomik yararları, sürdürülebilir kalkınma hedeflerine ulaşmada önemli rol oynayacaktır.

Sonuç olarak, bu projenin sonuçları, su talebi tahmininin yanı sıra genel olarak zaman serisi tahmin problemleri için de geçerli olan yöntemler ve teknikler sunmaktadır. Elde edilen modeller ve algoritmalar, benzer veri yapılarına sahip diğer alanlarda da kullanılabilir, bu da çalışmanın geniş uygulama potansiyelini ortaya koymaktadır. Bu geniş kapsamlı uygulanabilirlik, makine öğreniminin gerçek dünya problemlerine çözümler sunma kapasitesini pekiştirirken, bu alanda yapılacak gelecek çalışmalar için sağlam bir temel oluşturmaktadır.

KAYNAKÇA

Deng, L., Chang, X., & Wang, P. (2022). Daily Water Demand Prediction Driven by Multi-source Data. *Procedia Computer Science*.

Fiorillo, D., Kapelan, Z., Xenochristou, M., Poala, F. D., & Giugni, M. (2021). Assessing the Impact of Climate Change on Future. *Water Resources Management*.

Guo, L., Zhu, W., Wei, J., & Wang, L. (2022). Water demand forecasting and countermeasures across the Yellow River basin: Analysis from the perspective of water resources. *Journal of Hydrology: Regional Studies*.

İstanbul Büyükşehir Belediyesi. (2023, Kasım 1). *İstanbul Yıllık Su Tüketimi Verileri*. İstanbul Büyükşehir Belediyesi: https://data.ibb.gov.tr/dataset/istanbul-yillik-su-tuketimi-verileri adresinden alındı

Kavya, M., Mathew, A., Shekar, P. R., & P, S. (2023). Short term water demand forecast modelling using artificial intelligence for smart water management. *Sustainable Cities and Society*.

Li , J., Liu, C., & Tang, L. (2021). Forecast of regional water demand based on NSGAII-FORAGM. *Water Supply*.

Li, D., & Fu, Q. (2023). Deep learning model-based demand forecasting for secondary water supply in residential communities: A Case Study of Shanghai City, China. *IEEE*.

Liu, G., Savic, D., & Fu, G. (2023). Short-term water demand forecasting using data-centric machine learning approaches. *Journal of Hydroinformatics*.

Nasser, A. A., Rashad, M. Z., & Hussein, S. E. (2021, Agust 21). A Two-Layer Water Demand Prediction System in Urban Areas Based on Micro-Services. *IEEE*.

Pekel, E. (2022). Forecasting water demand for Istanbul by applying different machine learning algorithms. *Research Square*.

Sharma, S. K. (2022). A novel approach on water resource management with Multi-Criteria . *Environmental Research* .

Shirkoohi, M. G., Doghri, M., & Duchesne, S. (2021). Short-term water demand predictions coupling an. *Corrected Proof*.

Shuang, Q., & Zhao, R. T. (2021). Water Demand Prediction Using Machine Learning Methods: A Case Study of the Beijing–Tianjin–Hebei Region in China. *Water*.

Streamlit. (2024, Haziran 5). Streamlit: https://streamlit.io/generative-ai adresinden alındı

WallpaperFlare. (2024, Haziran 5). *Blue Bubble Drop Motion" adlı masaüstü arka planı*. WallpaperFlare: https://www.wallpaperflare.com/water-desktop-backgrounds-for-winter-blue-bubble-drop-motion-wallpaper-qtfio adresinden alındı

Wang, K., Ye, Z., Wang, Z., Liu, B., & Feng, T. (2023). MACLA-LSTM:A Novel Approach for Forecasting Water Demand. *Sustainability*.

Wang, Z., Huang, Y., Liu, T., Zan, C., Ling, Y., & Guo, C. (2022). Analysis of the Water Demand-Supply Gap and Scarcity Index in Lower AmuDaryaRiver Basin, Central Asia. *Environmental Research and Public Health*.

Zanfei, A., Brentan, B. M., Menepace, A., Righetti, M., & Herrera, M. (2022). Graph Convolutional Recurrent Neural Networks for Water. *Water Resources Research*.

Zubaidi, S. L., Hashim, K., Ethaib, S., Al-Bdairi, N. S., Al-Bugharbe, H., & Gharghan, S. K. (2022). Anovel methodology to predict monthly municipal water demand based. *Journal of King Saud University– Engineering Sciences*.

EKLER

Ek 1 Bitirme Projesi Künyesi(Graduation Project Information)

BİTİRME PROJESİ KÜNYESİ (GRADUATION PROJECT INFORMATION)

1. Bitirme Tezinin Adı (Name of the Graduation Project): Yapay Zekâ Tabanlı Su Kaynaklarının Entegre Yönetimi.

2. Tezin Amacı (Purpose): "Yapay Zekâ Tabanlı Su Kaynaklarının Entegre Yönetimi" projesinin amacı, su kaynaklarının etkin yönetimi ve korunması için kapsamlı bir yapay zekâ ve makine öğrenmesi tabanlı platformun geliştirilmesidir.

3. Öğrenci Numarası, Adı ve Soyadı (Student Number, Name and Surname):

200403617, Doğukan Sürücü

200403623, Elif Sakal

200403599, Ayşe Nur Korkmaz

4. Öğrencinin Daimi E-Posta Adresi (Student's E-mail Address):

200403617@ogr.gelisim.edu.tr

200403623@ogr.gelisim.edu.tr

200403599@ogr.gelisim.edu.tr

5. Öğrencinin Bölümü (Student's Department): Bilgisayar Mühendisliği

6. Bitirme Tezinin Juri Tarihi (Jury Date): 2024

7. Bitirme Tezinin Dönemi (Period): 2023-2024 Bahar

8. Bitirme Tezinin Danışman Öğretim Üyesi (Advisor Faculty Member): Doç. Dr.

ELHAM PASHAEI

9. Yazılım, veri tabanı vb. Ortamlar / Araçlar (Software, Database, Environments / Tools):

Python, NumPy, Pandas, Seaborn, Matplotlib, Scikit-learn, TensorFlow, Keras, Kaggle, JupyterLab, PyCharm

10. Uluslararası Standartlar (International Standards): ISO/IEC 27001, GDPR

(General Data Protection Regulation)

11. Mühendislik Tasarımı (Engineering Design):

Modüler Tasarım: Proje, veri ön işleme, model oluşturma, performans değerlendirme ve tahmin yapma gibi modüllere ayrıldı.

Tasarım Adımları: Veri toplama ve ön işleme, normalizasyon ve özellik ekleme, model kurulumu ve eğitimi, model başarı ve performans değerlendirme, tahmin yapma.

12. Kısıtlar ve İterasyonlar (Constraints and Iterations): Farklı algoritmaların denenmesi, projenin tamamlanması için belirlenen süre.