İZMİR KÂTİP ÇELEBİ ÜNİVERSİTESİ
2010
FEN BİLİMLERİ ENSTİTÜSÜ

EGE BÖLGESİ DEPREMLERİNİN MAKİNE ÖĞRENMESİ YÖNTEMLERİ İLE TAHMİN EDİLMESİ ve SONUÇLARININ KARŞILAŞTIRILMASI

Yazılım Mühendisliği Ana Bilim Dalı

Yüksek Lisans Tezi

Çağlar Özkören

ORCID 0000-0000-0000-0000

Tez Danışmanı: Prof. Dr. Femin Yalçın Küçükbayrak

Haziran 2023

Yazarlık Beyanı

Ben, **Çağlar Özkören**, başlığı **Ege Bölgesi Depremlerinin Makine Öğrenmesi Yöntemleri ile Tahmin Edilmesi ve Sonuçlarının Karşılaştırılması** olan bu bitirme projemin ve bitirme projesinin içinde sunulan bilgilerin şahsıma ait olduğunu beyan ederim. Ayrıca:

- Bu çalışmanın bütünü veya esası bu üniversitede Yüksek Lisans derecesi elde etmek üzere çalıştığım süre içinde gerçekleştirilmiştir.
- Daha önce bu bitirme projesinin herhangi bir kısmı başka bir derece veya yeterlik almak üzere bu üniversiteye veya başka bir kuruma sunulduysa bu açık biçimde ifade edilmiştir.
- Başkalarının yayımlanmış çalışmalarına başvurduğum durumlarda bu çalışmalara açık biçimde atıfta bulundum.
- Başkalarının çalışmalarından alıntıladığımda kaynağı her zaman belirttim. Bitirme projemin bu alıntılar dışında kalan kısmı tümüyle benim kendi çalışmamdır.
- Kayda değer yardım aldığım bütün kaynaklara teşekkür ettim.
- Bitirme projemde başkalarıyla birlikte gerçekleştirilen çalışmalar varsa onların katkısını ve kendi yaptıklarımı tam olarak açıkladım.

Tarih: …. / …. / 2023

_____

EGE BÖLGESİ DEPREMLERİNİN MAKİNE ÖĞRENMESİ YÖNTEMLERİ İLE TAHMİN EDİLMESİ ve SONUÇLARININ KARŞILAŞTIRILMASI

Öz

Bu proje, yapay zekâ alanında makine öğrenmesi modelleri kullanılarak, Türkiye'nin Ege Bölgesinde gerçekleşen depremlerin tahmin modelidir. Ayrıca LSTM (Uzun Kısa Vadeli Bellek) tabanlı bir model kullanarak deprem büyüklüğü tahmini yapmak için bir makine öğrenimi yaklaşımı geliştirildi. Deprem aktivitesinin zaman içindeki değişimini analiz etmek ve gelecekteki deprem büyüklüklerini tahmin etmek, deprem riskinin anlaşılması ve önlemlerin alınması açısından büyük önem taşımaktadır.

Veri seti olarak 1923-2023 yılları arasındaki Türkiye'de gerçekleşen depremlerin büyüklükleri ve bazı depremle ilgili özellikler içeren kapsamlı bir veri seti kullanıldı. İlk olarak, veri seti incelendi ve gerekiyorsa düzenlemeler yapıldı. Eksik veya anormal değerler düzeltilerek, veri seti doğrulandı ve kullanıma hazır hale getirildi. Özellik mühendisliği aşamasında, depremle ilgili bilgileri içeren özellikler seçildi ve düzenlendi. Tarih, enlem, boylam ve derinlik gibi depremle ilgili önemli faktörler, tahmin modeline dahil edildi. Ayrıca, özelliklerin uygun şekilde ölçeklendirilmesi ve gerektiğinde yeni özelliklerin türetilmesi için ön işleme adımları da uygulandı. Veri seti, eğitim ve test veri kümeleri olarak bölündü. Eğitim veri seti, modelin eğitimi için kullanıldı ve test veri seti, modelin performansını değerlendirmek için kullanıldı. LSTM tabanlı bir model oluşturuldu ve eğitim veri seti üzerinde eğitildi. Ardından, test veri seti üzerinde modelin performansı değerlendirildi. Diğer makine öğrenimi algoritmaları da uygulandı ve tahmin sonuçları ve hata metrikleri karşılaştırıldı. Çoklu Doğrusal Regresyon, Polinomal Regresyon ve Karar Ağaçları gibi popüler

algoritmalar kullanıldı. MSE (ortalama karesel hata) ve R^2 (determinasyon katsayısı) gibi hata metrikleri kullanılarak performans değerlendirmesi yapıldı.

Sonuçlar, LSTM tabanlı modelin diğer algoritmalara kıyasla daha iyi bir performans sergilediğini gösterdi. MSE değeri ve R^2 skoru açısından en düşük hata ve en yüksek açıklama gücü sağlandı. Bu sonuçlar, LSTM modelinin deprem büyüklüğü tahmininde etkili bir araç olduğunu ve gelecekteki depremleri öngörmek için potansiyel bir değer taşıdığını göstermektedir. Bu tez, LSTM tabanlı deprem büyüklüğü tahmini modelinin geliştirilmesi ve performansının değerlendirilmesiyle ilgili kapsamlı bir çalışmadır. Modelin doğruluğu ve kullanılabilirliği, deprem riskinin anlaşılması, önlemlerin alınması ve felaket yönetimi gibi alanlarda uygulanabilir ve faydalı bilgiler sağlayabilir.

Anahtar Sözcükler: Makine Öğrenmesi, LSTM (Uzun Kısa Vadeli Bellek), Depremler, Lineer Regresyon, Çoklu Doğrusal Regresyon, Karar Ağaçları

PREDICTION OF AEGEAN REGION EARTHQUAKE WITH MACHINE LEARNING METHODS AND COMPARISON OF THE RESULTS

Abstract

This project is a prediction model of earthquakes in the Aegean Region of Turkey, using machine learning models in the field of artificial intelligence. In addition, a machine learning approach was developed to estimate earthquake magnitude using an LSTM (Long Short Term Memory) based model. Analyzing the change of earthquake activity over time and estimating future earthquake magnitudes are of great importance in terms of understanding earthquake risk and taking precautions.

A comprehensive data set including the magnitudes of earthquakes and some earthquake-related features in Turkey between the years 1923-2023 was used as a data set. First, the dataset was reviewed and adjustments were made if necessary. By correcting missing or abnormal values, the data set was validated and made ready for use. During the feature engineering phase, features containing earthquake-related information were selected and edited. Important earthquake-related factors such as date, latitude, longitude, and depth were included in the forecast model. In addition, preprocessing steps were also implemented to scale features appropriately and derive new features as needed. The dataset was split into training and test datasets. The training dataset was used for training the model and the test dataset was used to evaluate the performance of the model.

An LSTM-based model was created and trained on the training dataset. Then, the performance of the model was evaluated on the test dataset. Other machine learning algorithms were also applied and prediction results and error metrics were compared. Popular algorithms such as Multiple Linear Regression, Polynomial Regression and Decision Trees were used. Performance evaluation was made using error metrics such as MSE (mean squared error) and R^2 (coefficient of determination).

The results showed that the LSTM based model outperformed other algorithms. The lowest error and highest explanatory power were provided in terms of MSE value and R^2 score. These results show that the LSTM model is an effective tool for earthquake magnitude estimation and has a potential value for predicting future earthquakes. This thesis is a comprehensive study on the development and performance evaluation of an LSTM based earthquake magnitude estimation model. The accuracy and usability of the model can provide practical and useful information in areas such as understanding earthquake risk, taking precautions, and disaster management.

Keywords: Machine Learning models, LSTM (Long Short Term Memory), Earthquakes, Multiple Linear Regression, Polynomial Regression, Decision Tree

Teşekkür

Tez çalışmasına katkıları ve destekleri sebebiyle Danışmanım Prof. Dr. Femin Yalçın Küçükbayrak' a, Yüksek Lisans öğrenimi boyunca katkılarından dolayı Üniversitemiz Fen Bilimleri Enstitüsü Yazılım Mühendisliği Anabilim Dalı Başkanımız Doç. Dr. Aytuğ Onan ve diğer tüm Öğretmenlerimize teşekkürlerimi iletirim.

İçindekiler

Yazarlık Beyanı i

Öz ii

Abstract iv

Teşekkür vi

Şekiller Listesi vii

Tablolar Listesi ix

Kısaltmalar Listesi x

Giriş 1

1 Literatür Özeti ve Çalışmanın Amacı 3

1.1 Literatür Özeti 3

1.2 Çalışmanın Amacı 4

2 Metodoloji 5

2.1 Yöntem ve Veri Seti 5

2.2 Verilerin Temizlenmesi 5

2.3 Parametrelerin Elde Edilmesi 6

3.4 Derin Öğrenme ve Makine Öğrenmesi 7

3 Model Metrik Bulguları 9

3.1 Metrikler 9

4 Proje Süreci ve Programlar 12

4.1 Kullanılan Kod Yapılarının Şekiller Aracılığı ile Gösterimi 12

Sonuç ve Değerlendirme 28

Kaynaklar 29

Özgeçmiş 31

Şekiller Listesi

Şekil 2.1 Modellerin Ortalama Kareler Hatası (MSE) 10
Şekil 2.2 Modellerin Ortalama Mutlak Hatası (MAE) 11
Şekil 2.3 Modellerin R^2 Skorları 11

Şekil 4.1 Kullanılan Kütüphaneler ve Veri Seti ile İlgili Bilgiler……13
Şekil 4.2 Veri Tipleri ve Keşifsel Veri Analizi……14
Şekil 4.3 Korelasyon Analizi……...15
Şekil 4.4 Zaman Serisi Analizi……16
Şekil 4.5 Kutu Grafiği ve Öznitelik Çıkarımı……..17
Şekil 4.6 Deprem Verileri ile İlgili İstatistiksel Analiz Değerleri Tabloları……...18
Şekil 4.7 Veri Setinin Birleştirilmesi ve Bölütlenmesi……19
Şekil 4.8 Long Short Term Memory (LSTM) Modelinin Hazırlanması…….20
Şekil 4.9 Epoch Değerleri……21
Şekil 4.10 LSTM Modeli Sonuçları ve Diğer Makine Öğrenmesi Modellerinin Hazırlanması ……22
Şekil 4.11 Polinomal ve Karar Ağaçları Regresyonları Modellerinin Analizleri…..23
Şekil 4.12 Rassal Ağaçlar Regresyonu Modelinin Analizi……24
Şekil 4.13 Metrik Sonuçlarının Görselleştirilmesi ile İlgili Kodlar……...25
Şekil 4.14 Modellerin Ortalama Kareler Hatası ve Ortalama Mutlak Hatası Skorları Karşılaştırılmaları…….26
Şekil 4.15 Modellerin R^2 Skorları Karşılaştırılmaları……27

Tablolar Listesi

Tablo 2.1 Veri Seti 5
Tablo 2.2 Veri Bölütlemesi 6
Tablo 2.3 LSTM Ağının Özellikleri 7

Kısaltmalar Listesi

LSTM	Long Short Term Memory (Uzun Kısa Süreli Bellek)
LR	Linear Regression (Lineer Regresyon)
PLR	Polynomial Regression (Çoklu Doğrusal Regresyon)
DTR	Decision Tree Regressor (Karar Ağacı Regresörü)
RFR	Random Forest Regressor (Rastgele Orman Regresörü)
MAE	Mean Absolute Error (Ortalama Mutlak Hata)
MSE	Mean Squared Error (Ortalama Karesel Hata)
R^2	Coefficient Of Determination Score (Determinasyon Katsayısı Skoru)

Giriş

Depremler, dünyanın farklı bölgelerinde yaşanan doğal afetler arasında en yıkıcı etkilere sahip olanlardan biridir. Depremlerin neden olduğu can kaybı, mal kaybı ve sosyal etkiler, toplumların güvenliğini ve sürdürülebilirliğini tehdit edebilmektedir. Bu nedenle, depremlerin önceden tahmin edilmesi ve etkilerinin minimize edilmesi büyük önem taşımaktadır.

Bu tez çalışması, deprem büyüklüğünün tahmin edilmesi konusunda yapay zeka ve makine öğrenimi tekniklerinin etkinliğini incelemeyi amaçlamaktadır. Deprem büyüklüğü, depremin enerji açığa çıkardığı ölçüsü olarak tanımlanır ve genellikle Richter ölçeği kullanılarak ifade edilir. Bu çalışma, deprem büyüklüğünün tahmininde kullanılan LSTM (Uzun Kısa Vadeli Bellek) tabanlı bir modelin performansını diğer makine öğrenimi algoritmalarıyla karşılaştırmaktadır. Bu çalışmanın amacı, deprem büyüklüğü tahmininde yapay zekâ ve makine öğrenimi tekniklerinin kullanılabilirliğini değerlendirmek ve en etkili modeli belirlemektir. Deprem verileri üzerinde gerçekleştirilen analizler ve karşılaştırmalar, deprem risk yönetimi ve felaket yönetimi gibi alanlarda daha doğru tahminler yapılmasına ve etkili önlemler alınmasına katkı sağlamayı hedeflemektedir.

Ayrıca, bu çalışma depremler hakkında genel bir anlayış sağlamak için depremler ve deprem büyüklüğü hakkında temel bilgiler sunacaktır. Depremler, yer kabuğundaki hareketlilikler sonucu meydana gelen titreşimlerdir ve büyük çoğunluğu fay hatlarında gerçekleşir. Depremlerin büyüklüğü, enerji açığa çıkardığı ölçüde ifade edilir ve büyüklük arttıkça depremin etkisi ve yıkıcılığı da artar.

Tez çalışmasının sonuçları, deprem büyüklüğü tahmini alanında yapay zekâ ve makine öğrenimi tekniklerinin kullanılabilirliğini değerlendiren bir kaynak olarak hizmet edecektir. Ayrıca, deprem risk yönetimi ve felaket yönetimi konularında karar vericilere ve uzmanlara önemli bir bilgi kaynağı sağlayacaktır. Sonuç olarak, bu çalışma depremle ilgili önemli bir konuyu ele almakta ve daha güvenli bir geleceğe doğru adımlar atmada katkıda bulunmaktadır. Bu tez çalışması, yapay zekâ ve makine öğrenimi tekniklerinin deprem büyüklüğü tahmini üzerindeki etkisini değerlendiren bir araştırma sunmaktadır. Aşağıdaki bölümlerde, kullanılan veri seti, metodoloji,

sonuçlar ve tartışma detaylı bir şekilde açıklanmaktadır. Sonuçlar, yapay zekâ tabanlı bir modelin deprem büyüklüğü tahmininde diğer geleneksel yöntemlere göre daha yüksek bir başarı oranına sahip olduğunu göstermektedir.

1. LİTERATÜR ÖZETİ VE ÇALIŞMANIN AMACI

1.1. Literatür Özeti

Kaftan ve Gök (2013) İzmir ve çevresindeki zemin özelliklerini yapay sinir ağları kullanarak incelemişler ve ivmeölçer istasyonlarından gelen verilerle zemin özelliklerini önişleme gerektirmeden belirlemişlerdir.[1]

Martínez-Álvarez ve diğerleri (2013) ise sismik göstergelerin yapay sinir ağları için girdi olarak kullanımını incelemiştir. Bu çalışmada, farklı sismik bölgelerde başarılı bir şekilde kullanılan özellik seçme tekniği, çoklu gösterge kombinasyonuyla birlikte uygulanmıştır. Orijinal girdi setleri ve farklı sınıflandırıcıların karşılaştırılmasıyla elde edilen başarı derecesi raporlanmıştır. Ayrıca, farklı sismik göstergelerden elde edilen bilgi, elde etme analizi kullanılarak dört Chile bölgesi ve iki İberia yarımadası bölgesi için karakterize edilmiştir.[2]

Reyes ve diğerleri (2013), deprem aktivitesinin yoğun olduğu Chile'de yapılan bir çalışmada, depremlerin tahmin edilmesi için yapay sinir ağı kullanılmıştır. Bu yapay sinir ağı, b değeri, Bath kanunu ve Omori Utsu kanunu gibi depremsellikle güçlü ilişkisi olan parametreleri kullanarak belirli bir eşik değerinin üstündeki depremleri tahmin etmektedir. Yapılan tahminler, istatistiksel testler aracılığıyla değerlendirilmiş ve makine öğrenme sınıflandırıcılarıyla karşılaştırılmıştır.[3]

Çelik ve diğerleri (2014), Yapay Sinir Ağları ve Destek Vektör Makineleri kullanarak sismik darbelerden deprem tahmini yapmışlardır. Yapay Sinir Ağları ile %83, Destek Vektör Makineleri ile ise %91 oranında doğru sınıflandırma elde etmişlerdir.[4]

1.2. Çalışmanın Amacı

Bu çalışmanın amaçları şu şekilde sıralanabilir.

- Ülkemizde yaşanan depremlerin tahmini için literatüre yeni bir yapı eklemek.
- Bu yapıyı kullanarak, oluşturulan modellerle, doğruluğu yüksek bir deprem tahmini yapan projeyi literatüre eklemek.
- Yapay zekâ alanında sıklıkla kullanılan makine öğrenmesi ve derin öğrenme modelleriyle bu alanda en iyi çalışan algoritmayı tespit etmek.

2. METODOLOJİ

2.1. Yöntem ve Veri Seti

01.01.1923 ile 11.02.2023 tarihleri arasında Ege Bölgesi kuzey koordinatı Kütahya ilinin en kuzeyi 40.10 ile güneyde Muğla ilinin en güneyi 35.93 enlemleri ile batı da İzmir ilinin en batısı 26.15 ve doğuda Afyonkarahisar ilinin en doğusu 31.43 boylamları arasında veri setimiz alınmıştır. Excel e aktarılmıştır. 1221 satırlık veri setidir.

Tablo 2.1 Veri Seti

Bölge Adı	Satır Sayısı	Sütun Sayısı
Ege Bölgesi	1222	15

Türkiye'nin Ege Bölgesi'nde yer alan depremlerin 15 farklı öznitelikleriyle beraber öncelikle veri önişlemesi ve keşifsel veri analizi, ardından korelasyon analizi ve zaman serisi analizleri yapılmıştır. Ardından özintelik çıkarımı ve veri ölçeklemenin ardından veriler üzerinde modelimiz için ihtiyaç duyulan veri bölünme işlemi yapılmıştır. Veri bölütleme işlemi tamamlandıktan sonra derin öğrenme ve makine öğrenmesi modellerinin tahmini hazırlanmıştır. Derin öğrenme alanında LSTM modeli, denetimli makine öğrenmesi alanında LR, Çoklu Doğrusal Regresyon, DTR, RFR gibi öğrenme yöntemleri kullanılmıştır.

2.2. Verilerin Temizlenmesi

Bu projedeki veri setine ait olan bilgilere bakıldığında 1923 yılından itibaren boylam, enlem, yıl, ay, gün, magnitüt, derinlik, saat, dakika ve süre bilgileri kaydedilmiştir. Veriler Excel dosyasına kaydedilmiş ve ardından Python programı kullanılarak veri temizliği yapılmıştır. Veri temizliği esnasında; veri okuma ve düzenleme, eksik veri doldurma, veri tipi dönüşümü, aykırı değerlerinin tespiti ve veri ölçeklendirme yapıları kullanılmıştır. İlk olarak, veri seti okunmuş ve gerekiyorsa düzenlemeler yapılmıştır. Daha sonra 'Mw' sütunundaki eksik değerler, mean fonksiyonu kullanılarak o

sütundaki tüm değerlerin ortalama değerleriyle doldurulmuştur. Bu yöntem, eksik veri noktalarının tamamlanmasına yardımcı olur. Veri setindeki 'Olus tarihi' sütunu string veri tipinde olduğundan, tarih formatına dönüştürülmüştür. Böylece zaman serisi analizleri yapılabilir hale gelmiştir. Ayrıca veri setindeki aykırı değerler, seaborn kütüphanesindeki box plot fonksiyonu kullanılarak görselleştirilmiştir. Box plot grafiği, değişkenin dağılımını ve aykırı değerlerin varlığını gösterir. Bu yöntemle, aykırı değerlerin tespiti ve analizi yapılabilir. Son olarak veri ölçeklendirme adımında 'Enlem', 'Boylam' ve 'Der(km)' sütunları ölçeklendirilmiştir. Ölçeklendirme işlemi, farklı ölçeklerdeki verilerin aynı ölçekte olmasını sağlar ve model performansını iyileştirebilir. Bizim yazdığımız algoritmada tüm ölçeklenmiş değerler 0 ile 1 arasında dağılmıştır.

2.3. Parametrelerin Elde Edilmesi

Yapılan çalışmada, verilerin önişleme adımı tamamlandıktan sonra veriyi öncelikle ikiye bölme işlemi gerçekleştirilmiştir. İlk yapıda hedef değişken olan ‘xM’ parametresi hariç tüm değerler X olarak adlandırılmış ve ardından hedef değişkenin kendisi, yani ‘xM’ değeri y olarak adlandırılmıştır. Bu aşamada verileri böyle bölmemizin sebebi verileri eğitirken, sahip olduğu özniteliklere göre bir matematiksel fonksiyon oluşturup her bir yapının sonucu olarak kendisine karşılık gelen y değişkenine eşitlemesi olacaktır. Ardından modelde kullanmamız gereken yapıları eğitim ve test olarak bölmemiz gerektiğinden bu yapı %70 eğitim ve geriye kalan %30 test haline bölünmüştür. Scikit-learn kütüphanesi ile bu yapılar X_train, X_test, y_train ve y_test olarak 4 parçaya ayrılmıştır.

Tablo 2.2 Veri Bölütlemesi

Değişken Adı	Büyüklüğü
X_train	855
X_test	367
y_train	855
y_test	367

2.4. Derin Öğrenme ve Makine Öğrenmesi Modelleri

Çalışmamızda derin öğrenme modeli ve makine öğrenmesi modelleri kullanılmıştır. Derin öğrenme modeli olarak LSTM modeli kullanılmıştır. LSTM, zaman serileri ve sıralı veriler üzerinde etkili olan bir derin öğrenme modelidir. Bu projede LSTM modeli, deprem büyüklüğünü tahmin etmek için kullanılmıştır. Model, girdi olarak veri setindeki çeşitli öznitelikleri kullanır ve ardışık katmanlarla (LSTM ve Yoğun Katman) deprem büyüklüğünü tahmin eder. LSTM modelinde çeşitli katmanlar kullanılmış ve çıktı sonucumuz girdi olarak tekrar kullanılarak modelimiz eğitilmiştir.

Tablo 2.3 LSTM Ağının Özellikleri

Parametreler	Değerler
Girdi Katmanı	64
İkinci Katmanı	32
Çıktı Nöron Sayısı	1
Optimizasyon Fonksiyonu	Adam
Kayıp Fonksiyonu	Mean Squared Error
Epoch Sayısı	50
Batch Sayısı	32

Makine öğrenmesi modellerine bakıldığında Linear Regression, Polynomial Regression, Decision Tree Regressor ve Random Forest Regressor modelleri kullanılmıştır.

Doğrusal regresyon, bağımlı ve bağımsız değişkenler arasındaki ilişkiyi ifade eden doğrusal bir model oluşturur. Bu projede, doğrusal regresyon modeli kullanılarak deprem büyüklüğünün tahmini gerçekleştirilmiştir.[5]

Polynomial Regression, doğrusal olmayan ilişkilere sahip veri setleri üzerinde kullanılan bir regresyon yöntemidir. Polinom regresyon, veri setindeki öznitelikleri

polinomik terimlerle genişleterek ve lineer regresyon kullanarak bir tahmin modeli oluşturur. Projede kullanılan polinom regresyonu modelinde 2. derece polinom kullanılmıştır.[6]

Diğer bir yöntem olan karar ağaçları, veri setindeki öznitelikleri kullanarak bir dizi karar yapısı oluşturan bir modeldir. Bu projede, karar ağaçları modeli kullanılarak deprem büyüklüğünün tahmini gerçekleştirilmiştir.[7]

Rassal ağaçlar, birden çok karar ağacının bir araya gelerek oluşturduğu bir bileşik modelidir. Rassal ağaçlar, veri setindeki öznitelikler üzerinde rastgele alt örneklemeler yaparak ve birden çok ağacın tahminlerini birleştirerek daha güçlü bir tahmin modeli oluşturur.[8]

3. Model Metrik Bulguları

Deprem büyüklüğünü tahmin etmek için makine öğrenmesi modelleri kullanılarak gerçekleştirilen bu çalışmada, çeşitli performans metrikleri kullanılarak modellerin başarısı değerlendirildi.

İlk olarak, LSTM (Long Short Term Memory) modeli kullanıldı. Bu model, veri setinin yapısını ve zamansal ilişkileri dikkate alarak deprem büyüklüğünü tahmin etmeye çalışır. Eğitim sürecinde 64 hücreli bir LSTM katmanı ve 32 hücreli bir LSTM katmanı kullanıldı. Ardından, bir yoğun katman eklenerek sonuç elde edildi. Modelin optimize edilmesi için Adam optimizer kullanıldı ve kayıp fonksiyonu olarak ortalama kare hata(MSE) seçildi. LSTM modelinin test edilmesi sonucunda, ortalama kare hata (MSE) değeri elde edildi. Ayrıca, ortalama mutlak hata (MAE) ve R-kare (R2) skorları da hesaplandı.

Diğer kullanılan modeller arasında Lineer Regresyon, Polinomial Regresyon, Karar Ağaçları ve Rassal Ağaçlar yer aldı. Bu modellerin performansı da aynı metrikler kullanılarak değerlendirildi.

3.1. Metrikler

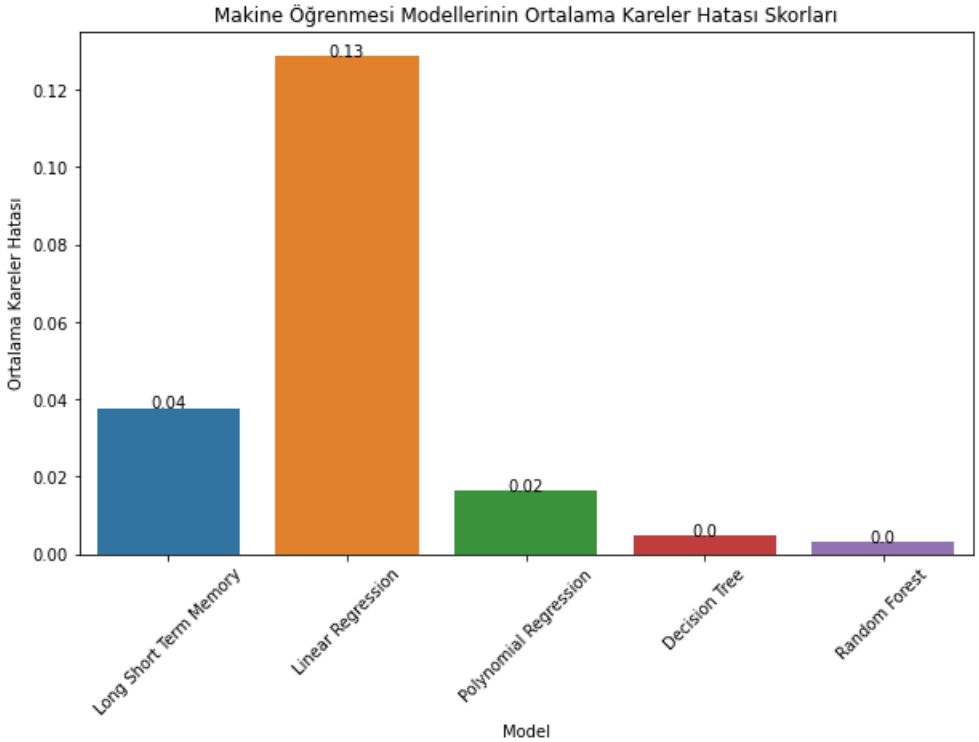
MSE (Ortalama Kareler Hatası): MSE, gerçek ve tahmin edilen değerler arasındaki farkların karesinin ortalama değerini temsil eder. MSE değeri ne kadar düşükse, modelin tahminleri gerçek değerlere o kadar yakındır ve daha iyi bir uyum sağlamış demektir. MSE, tahmin hatalarının büyüklüğünü dikkate alır ve aykırı değerlerin etkisini artırabilir.[9]

MAE (Ortalama Mutlak Hata): MAE, gerçek ve tahmin edilen değerler arasındaki mutlak farkların ortalama değerini temsil eder. MAE, tahmin hatalarının büyüklüğünü ölçerken mutlak değerler kullanır, yani negatif veya pozitif hataların etkisini aynı şekilde değerlendirir. MAE değeri ne kadar düşükse, modelin tahminleri gerçek değerlere o kadar yakındır ve daha iyi bir uyum sağlamış demektir. MSE'den farklı olarak, MAE aykırı değerlerin etkisini daha az dikkate alır.[10]

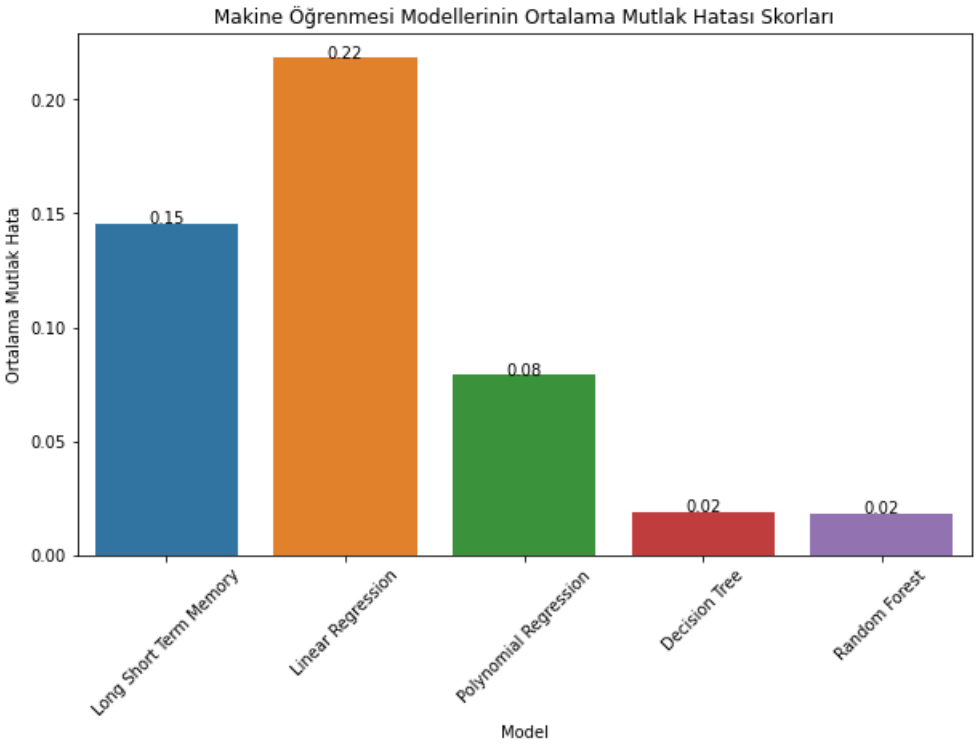
R^2 Skoru (R-Kare Skoru): R^2 skoru, modelin bağımsız değişkenler tarafından açıklanan toplam değişkenliği yüzde olarak ifade eder. R^2 skoru, modelin veri setindeki değişkenliği ne kadar iyi açıkladığını gösterir. En yüksek değer olan 1'e yaklaştıkça, modelin bağımlı değişkenin değişkenliğini tamamen açıkladığı anlamına gelir. R^2 skoru negatif olamaz ve 0 ile 1 arasında bir değer alır. R^2 skoru ne kadar yüksekse, modelin veri setindeki değişkenliği o kadar iyi açıkladığı söylenebilir.[10]

Sonuçlar, Karar Ağaçları ve Rassal Ağaçlar modellerinin en düşük MSE ve MAE değerlerine sahip olduğunu gösterdi. R^2 skoruna bakıldığında 0.98 ile Rassal Ağaçlar modeli veri setindeki değişkenliği en iyi açıklayan model olduğunu gösterdi. Diğer modellerin de iyi performans gösterdiği ancak Rassal Ağaçlar modelinin daha yüksek bir doğruluk sağladığı görüldü. Bu bulgular, deprem büyüklüğü tahmininde Rassal Ağaçlar gibi makine öğrenme modellerinin etkili bir şekilde kullanılabileceğini göstermektedir.

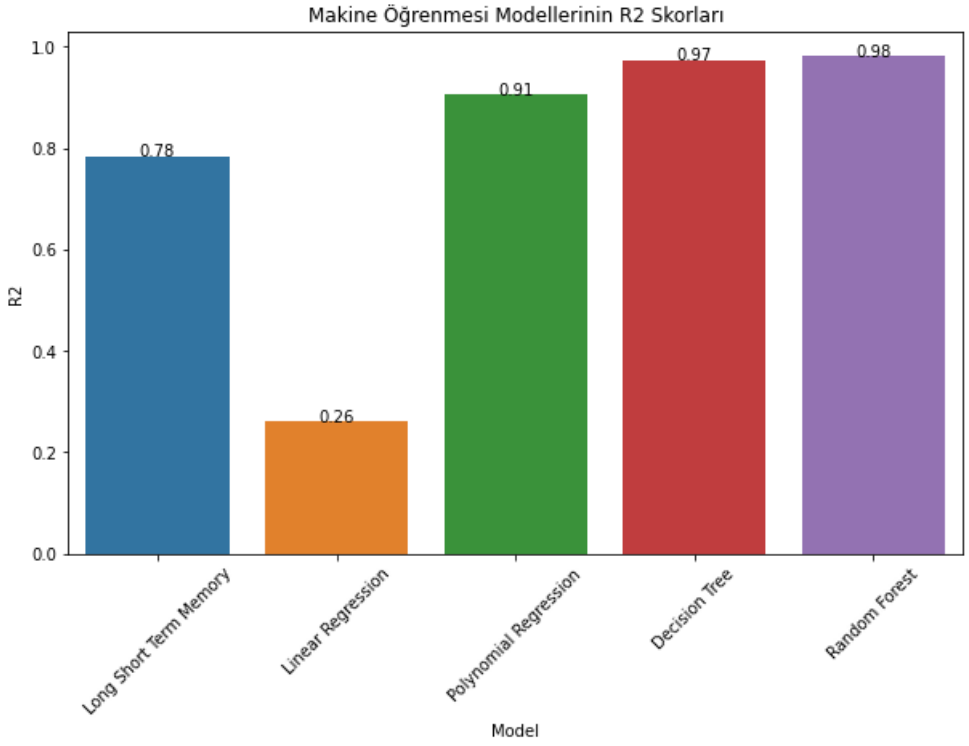
Bu çalışmanın sonuçları, deprem tahmin modellerinin geliştirilmesi ve doğal afetlerle mücadelede daha etkili önlemler alınması konusunda ilgili araştırmacılara yol gösterebilir. Ayrıca, makine öğrenmesi yöntemlerinin deprem risk analizi ve erken uyarı sistemlerinin geliştirilmesi gibi alanlarda da kullanılabilirliğini vurgulamaktadır.



Şekil 3.1 Modellerin Ortalama Kareler Hatası (MSE)



Şekil 3.2 Modellerin Ortalama Mutlak Hatası (MAE)



Şekil 3.3 Modellerin R^2 Skorları

4. Proje Süreci ve Programlar

4.1 Kullanılan Kod Yapılarının Şekiller Aracılığıyla Gösterimi

Bu bölümde, açık kaynak kodlu Jupyter Notebook Programı aracılığı ile Makine Öğrenmesi Metotları Kullanımında başarılı sonuçları almamızı sağlayan ve bunun için gereken süreyi önemli ölçüde azaltan, veri analizleri için kullanılan tam özellikli bir kütüphane hattı sunan, sade ve sezgisel bir programlama dili olan Python kullanılarak yapılan Projenin ilgili sürecinde neler yapıldığı hakkında aşamalı şekilde konu başlıklarıyla beraber açıklamalı kodları Şekiller vasıtasıyla gösterilmektedir.

1- Kütüphanenin Tanımlanması ve Verisetinin Sisteme Yüklenmesi

In [1]:

```
"""
Kütüphanelerin sisteme import edilmesi.
"""
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
import seaborn as sns
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
"""
Veri seti dosyamızı pandas kullanarak sisteme yükleyip ardından df değişkenine atıyoruz.
"""
df = pd.read_csv('19230101_20230211_4.5_9.0_43_654.csv',delimiter='\\t')
df.head()
```

Out[2]:

	"""No	Deprem Kodu	Olus tarihi	Olus zamani	Enlem	Boylam	Der(km)	xM	MD	ML	Mw	Ms	Mb	Tip	Yer"""
0	"""000001	20221104002920	2022.11.04	00:29:20.39	38.3653	27.1948	13.5	5.0	0.0	5.0	4.7	0.0	0.0	Ke	BUCA (IZMIR) [South East 3.1 km]"""
1	"""000002	20221002105802	2022.10.02	10:58:02.64	37.0608	28.8878	3.9	4.7	0.0	4.6	4.7	0.0	0.0	Ke	OTMANLAR-KOYCEGIZ (MUGLA) [North West 3.6 km]"""
2	"""000003	20220831140218	2022.08.31	14:02:18.20	37.5593	26.8283	14.4	4.7	0.0	4.6	4.7	0.0	0.0	Ke	EGE DENIZI"""
3	"""000004	20220831101010	2022.08.31	10:10:10.33	37.5827	26.8010	13.6	5.2	0.0	5.2	5.2	0.0	0.0	Ke	EGE DENIZI"""
4	"""000005	20220831095638	2022.08.31	09:56:38.73	37.5767	26.8193	13.1	4.8	0.0	4.8	4.8	0.0	0.0	Ke	EGE DENIZI"""

In [3]:

```
"""
Veri Setindeki ilk ve son sütun değerlerinin başında tırnak işaretleri mevcut bundan dolayı bu değerleri kaldırıyoruz.
"""
column_name = 'No'
df.columns = [column_name] + df.columns[1:].tolist()

df.rename(columns = {'Yer"""':'Yer'}, inplace = True)

for i in range(len(df)):
    df['No'].iloc[i] = df['No'].iloc[i].split('"')[-1]
    df['Yer'].iloc[i] = df['Yer'].iloc[i].split('"')[0]

df.head()
```

Out[3]:

	No	Deprem Kodu	Olus tarihi	Olus zamani	Enlem	Boylam	Der(km)	xM	MD	ML	Mw	Ms	Mb	Tip	Yer
0	000001	20221104002920	2022.11.04	00:29:20.39	38.3653	27.1948	13.5	5.0	0.0	5.0	4.7	0.0	0.0	Ke	BUCA (IZMIR) [South East 3.1 km]
1	000002	20221002105802	2022.10.02	10:58:02.64	37.0608	28.8878	3.9	4.7	0.0	4.6	4.7	0.0	0.0	Ke	OTMANLAR-KOYCEGIZ (MUGLA) [North West 3.6 km]
2	000003	20220831140218	2022.08.31	14:02:18.20	37.5593	26.8283	14.4	4.7	0.0	4.6	4.7	0.0	0.0	Ke	EGE DENIZI
3	000004	20220831101010	2022.08.31	10:10:10.33	37.5827	26.8010	13.6	5.2	0.0	5.2	5.2	0.0	0.0	Ke	EGE DENIZI
4	000005	20220831095638	2022.08.31	09:56:38.73	37.5767	26.8193	13.1	4.8	0.0	4.8	4.8	0.0	0.0	Ke	EGE DENIZI

Şekil 4.1 Kullanılan Kütüphaneler ve Veri Seti ile İlgili Bilgiler

In [4]:

```
"""
Veri setindeki bütün sütunların veri tiplerini ekrana basıyoruz. Burada Olus Tarihi sütunumuz object tipinde tutulmuş, bu değeri date time veri şekline çevirebiliriz.
"""
df.dtypes
```

Out[4]:

```
No              object
Deprem Kodu      int64
Olus tarihi     object
Olus zamani     object
Enlem          float64
Boylam         float64
Der(km)        float64
xM             float64
MD             float64
ML             float64
Mw             float64
Ms             float64
Mb             float64
Tip             object
Yer             object
dtype: object
```

2- Keşifsel Veri Analizi

In [5]:

```
"""
Keşifsel veri analizi için öncelikle info fonksiyonuyla görüntülüyoruz. Burada Mw sütunumuzda 403 adet boş değer olduğu görülüyor. Bu değerleri daha sonra fillna fonksiyonu ile doldurabiliriz.
"""
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1221 entries, 0 to 1220
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   No           1221 non-null   object 
 1   Deprem Kodu  1221 non-null   int64  
 2   Olus tarihi  1221 non-null   object 
 3   Olus zamani  1221 non-null   object 
 4   Enlem        1221 non-null   float64
 5   Boylam       1221 non-null   float64
 6   Der(km)      1221 non-null   float64
 7   xM           1221 non-null   float64
 8   MD           1221 non-null   float64
 9   ML           1221 non-null   float64
 10  Mw           818 non-null    float64
 11  Ms           1221 non-null   float64
 12  Mb           1221 non-null   float64
 13  Tip          1221 non-null   object 
 14  Yer          1221 non-null   object 
dtypes: float64(9), int64(1), object(5)
memory usage: 143.2+ KB
```

In [6]:

```
"""
Verisetimizde boş olan Mw değerlerini diğer sütun değerlerinin ortalaması ile dolduruyoruz.
"""
df['Mw'] = df['Mw'].fillna(df['Mw'].mean())
```


Şekil 4.2 Veri Tipleri ve Keşifsel Veri Analizi

In [7]:

```
"""
Describe fonksiyonu verilerimizin keşifsel veri analizinde sayıları, ortalamaları, standart sapmaları, minimum, maksimum ve
kuartil değerleri gibi nümerik analiz sonuçları görüntüleyebilmemizi sağlar.
"""
df.describe()
```

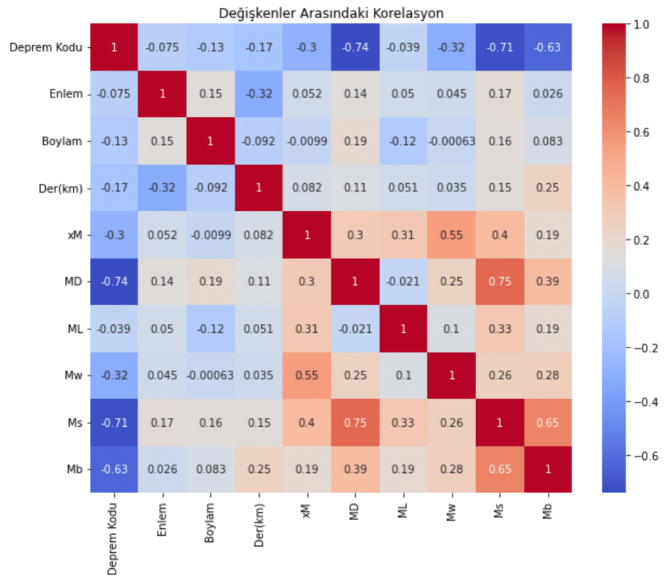
Out[7]:

	Deprem Kodu	Enlem	Boylam	Der(km)	xM	MD	ML	Mw	Ms	Mb
count	1.221000e+03	1221.000000	1221.000000	1221.000000	1221.000000	1221.000000	1221.000000	1221.000000	1221.000000	1221.000000
mean	1.977146e+13	37.719790	28.465326	32.577068	4.924161	2.875020	3.997707	4.908802	2.695168	3.621376
std	2.447121e+11	1.164982	1.358933	33.800131	0.429217	2.380028	1.771869	0.571665	2.420040	2.065666
min	1.923091e+13	35.941200	26.164300	0.000000	4.500000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	1.966012e+13	36.700000	27.200000	10.000000	4.600000	0.000000	4.300000	4.800000	0.000000	4.200000
50%	1.971052e+13	37.560000	28.440000	22.000000	4.800000	4.500000	4.600000	4.908802	4.200000	4.600000
75%	1.990090e+13	38.900000	29.560000	40.000000	5.100000	4.800000	4.900000	5.000000	4.800000	4.900000
max	2.022110e+13	40.080000	31.400000	192.000000	7.700000	7.100000	7.000000	7.200000	7.700000	6.900000

2.1- Korelasyon Analizi

In [8]:

```
"""
Veri setimizdeki sütunların birbiri arasındaki korelasyon değerlerini corr fonksiyonuyla elde edebiliyoruz. Bu korelasyon
tablosu bize, veriler arasında nasıl bir bağımlılık olduğunu gösteriyor.
"""
correlation_matrix = df.corr()

"""
Korelasyon matrisini görselleştirmek için seaborn kütüphanesi içerisinde bulunan heatmap fonksiyonunu kullanıyoruz.
"""
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm")
plt.title("Değişkenler Arasındaki Korelasyon")
plt.show()
```

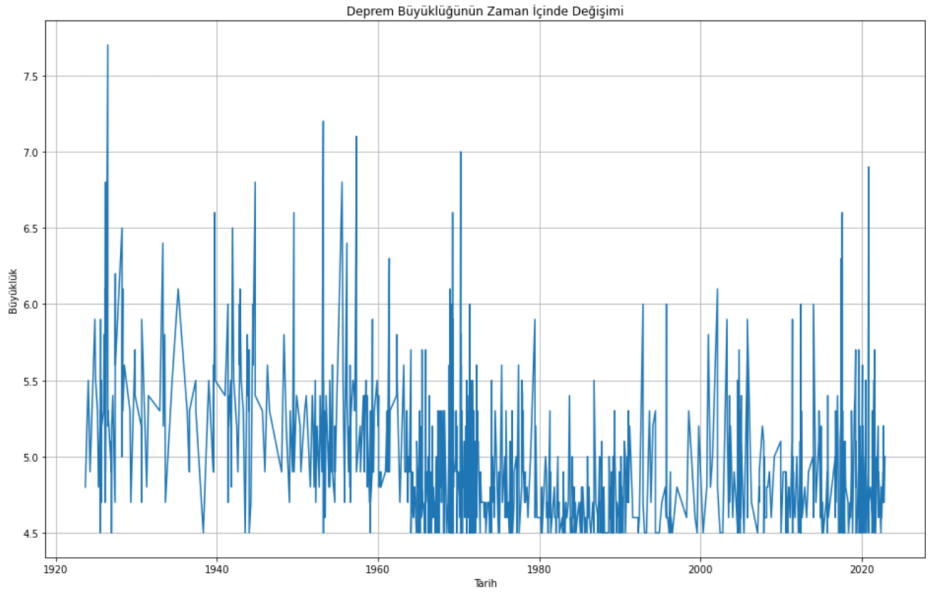


2.2- Zaman Serisi Analizi

Şekil 4.3 Korelasyon Analizi

In [9]:

```
"""
Zaman serisi analizi yapmak için öncelikle Olus Tarihi sütunumuzu ve hedef değişken değerimiz olan deprem büyüklüğü
değişkenimiz xM değerlerini time_series_data değişkenine atıyoruz. Daha sonra buradaki Olus Tarihi sütununu datetime veri
tipine dönüştürüyoruz ve çizgi grafiği şeklinde bu verilerimizi görselleştiriyoruz.
"""
time_series_data = df[["Olus tarihi", "xM"]].copy()

time_series_data["Olus tarihi"] = pd.to_datetime(time_series_data["Olus tarihi"])
time_series_data.set_index("Olus tarihi", inplace=True)

plt.figure(figsize=(16, 10))
plt.plot(time_series_data.index, time_series_data["xM"])
plt.xlabel("Tarih")
plt.ylabel("Büyüklük")
plt.title("Deprem Büyüklüğünün Zaman İçinde Değişimi")
plt.grid(True)
plt.show()
```

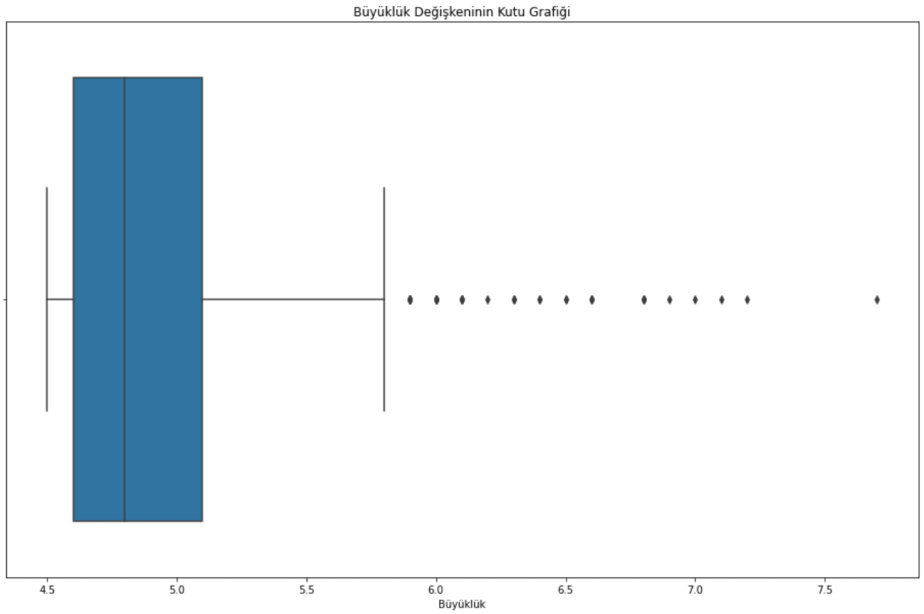


Şekil 4.4 Zaman Serisi Analizi

2.3 - Kutu Grafiği

In [10]:

```
"""
Outlier yani aykırı değer analizi yapmak adına kutu grafiği kullanarak hedef değişkenimiz olan deprem büyüklüğü değişkenini
görselleştiriyoruz
"""
plt.figure(figsize=(16,10))
sns.boxplot(x=df["xM"])
plt.xlabel("Büyüklük")
plt.title("Büyüklük Değişkeninin Kutu Grafiği")
plt.show()
```



3- Öznitelik Çıkarımı

In [11]:

```
df.columns
```

Out[11]:

```
Index(['No', 'Deprem Kodu', 'Olus tarihi', 'Olus zamani', 'Enlem', 'Boylam',
       'Der(km)', 'xM', 'MD', 'ML', 'Mw', 'Ms', 'Mb', 'Tip', 'Yer'],
      dtype='object')
```


Şekil 4.5 Kutu Grafiği ve Öznitelik Çıkarımı

In [12]:

```
df.dtypes
```

Out[12]:

```
No              object
Deprem Kodu      int64
Olus tarihi     object
Olus zamani     object
Enlem          float64
Boylam         float64
Der(km)        float64
xM             float64
MD             float64
ML             float64
Mw             float64
Ms             float64
Mb             float64
Tip             object
Yer             object
dtype: object
```

In [13]:

```
"""
No değeri depremlerin numaralarını, Deprem Kodu değeri depremlerin nümerik kodunu, Olus tarihi depremin tarihini, Olus zamani
değeri deprem saatini, Tip değeri her bir satırda aynı olan deprem tipini ve Yer sütunu konum bilgisi olduğundan bu nitelikler
eğitilen modelde kullanılmayacaklar.
"""
columns = ['Enlem','Boylam','Der(km)','xM','MD','ML','Mw','Ms','Mb']
df_new = df[columns]
```

In [14]:

```
"""
Modelimizde eğiteceğimiz değerleri ekrana basıyoruz.
"""
df_new.head()
```

Out[14]:

	Enlem	Boylam	Der(km)	xM	MD	ML	Mw	Ms	Mb
0	38.3653	27.1948	13.5	5.0	0.0	5.0	4.7	0.0	0.0
1	37.0608	28.8878	3.9	4.7	0.0	4.6	4.7	0.0	0.0
2	37.5593	26.8283	14.4	4.7	0.0	4.6	4.7	0.0	0.0
3	37.5827	26.8010	13.6	5.2	0.0	5.2	5.2	0.0	0.0
4	37.5767	26.8193	13.1	4.8	0.0	4.8	4.8	0.0	0.0

In [15]:

```
"""
Ardından eğiteceğimiz değişkenleri içeren yeni verisetimizin istatistiksel analiz değerlerini tablo şeklinde ekrana basıyoruz.
"""
df_new.describe()
```

Out[15]:

	Enlem	Boylam	Der(km)	xM	MD	ML	Mw	Ms	Mb
count	1221.000000	1221.000000	1221.000000	1221.000000	1221.000000	1221.000000	1221.000000	1221.000000	1221.000000
mean	37.719790	28.465326	32.577068	4.924161	2.875020	3.997707	4.908802	2.695168	3.621376
std	1.164982	1.358933	33.800131	0.429217	2.380028	1.771869	0.571665	2.420040	2.065666
min	35.941200	26.164300	0.000000	4.500000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	36.700000	27.200000	10.000000	4.600000	0.000000	4.300000	4.800000	0.000000	4.200000
50%	37.560000	28.440000	22.000000	4.800000	4.500000	4.600000	4.908802	4.200000	4.600000
75%	38.900000	29.560000	40.000000	5.100000	4.800000	4.900000	5.000000	4.800000	4.900000
max	40.080000	31.400000	192.000000	7.700000	7.100000	7.000000	7.200000	7.700000	6.900000

Şekil 4.6 Deprem Verileri ile İlgili İstatistiksel Analiz Değerleri Tabloları

In [16]:

```
"""
Bazı veri sütunlarımızın min ve max değerleri arasındaki fark çok fazla olduğundan bu değerlere veri ölçekleme methodu kullanı
yoruz bu yöntem verilerimizi eğitirken daha kolay ve hızlı eğitim yapılmasını sağlayacaktır.
"""
scaled_columns = ['Enlem','Boylam','Der(km)']

mms = MinMaxScaler()
scaled_data = mms.fit_transform(df[scaled_columns])

scaled_df = pd.DataFrame(scaled_data,columns=scaled_columns)
scaled_df.head()
```

Out[16]:

	Enlem	Boylam	Der(km)
0	0.585701	0.196822	0.070312
1	0.270513	0.520179	0.020312
2	0.390959	0.126822	0.075000
3	0.396613	0.121607	0.070833
4	0.395163	0.125103	0.068229

In [17]:

```
df_new = df_new.drop(['Enlem','Boylam','Der(km)'],axis=1)
"""
Ölçeklenmiş veri setini genel veri setiyle birleştirme
"""
merged_df = pd.concat([df_new, scaled_df], axis=1)

"""
Birleştirilmiş veri setini görüntüleme
"""
merged_df.head()
```

Out[17]:

	xM	MD	ML	Mw	Ms	Mb	Enlem	Boylam	Der(km)
0	5.0	0.0	5.0	4.7	0.0	0.0	0.585701	0.196822	0.070312
1	4.7	0.0	4.6	4.7	0.0	0.0	0.270513	0.520179	0.020312
2	4.7	0.0	4.6	4.7	0.0	0.0	0.390959	0.126822	0.075000
3	5.2	0.0	5.2	5.2	0.0	0.0	0.396613	0.121607	0.070833
4	4.8	0.0	4.8	4.8	0.0	0.0	0.395163	0.125103	0.068229

4- Veri Bölütleme

In [18]:

```
"""
Bir sonraki adımda verilerimizi bölüyoruz, burada öncelikle features ve labels olmak üzere veriler ikiye bölünüyor, labels yan
hedef değişkenimizde değrem büyüklüğü(xM) ve diğer features kısmında xM sütunu hariç tüm değerler olmuş oluyor. Sonrasında
verilerimizi %70 eğitim ve %30 test olarak X_train, X_test, y_train ve y_test şeklinde bölmüş oluyoruz.
"""
X = merged_df.drop('xM',axis = 1)
y = merged_df['xM']

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3)
```

5- Long Short Term Memory(LSTM) Modelinin Hazırlanması

Şekil 4.7 Veri Setinin Birleştirilmesi ve Bölütlenmesi

In [19]:

```
"""
Long Short Term Memory algoritmasında verilerimizi önce 64 nöronlu ilk ağ katmanına gönderiyoruz, ardından 32 nöronlu ikinci
ağımıza eğitim verilerini gönderiyoruz, burada yinelemeli işlemler yapılmasının ardından, son katmana yani tek kısımlı çıktı
katmanımıza verileri gönderiyoruz, burada adam optimizasyonu ve mean squared error kayıp fonksiyonu ile beraber modelimizi
çalıştırıyoruz.
"""
model = Sequential()
model.add(LSTM(64, input_shape=(X_train.shape[1], 1), return_sequences=True))
model.add(LSTM(32))
model.add(Dense(1))

model.compile(optimizer='adam', loss='mse')
model.fit(X_train, y_train, epochs=50, batch_size=32)
```


Şekil 4.8 Long Short Term Memory (LSTM) Modelinin Hazırlanması

```
Epoch 1/50
27/27 [==============================] - 5s 12ms/step - loss: 8.3033
Epoch 2/50
27/27 [==============================] - 0s 12ms/step - loss: 0.3061
Epoch 3/50
27/27 [==============================] - 0s 15ms/step - loss: 0.1888
Epoch 4/50
27/27 [==============================] - 0s 11ms/step - loss: 0.1786
Epoch 5/50
27/27 [==============================] - 0s 11ms/step - loss: 0.1772
Epoch 6/50
27/27 [==============================] - 0s 8ms/step - loss: 0.1771
Epoch 7/50
27/27 [==============================] - 0s 7ms/step - loss: 0.1770
Epoch 8/50
27/27 [==============================] - 0s 7ms/step - loss: 0.1756
Epoch 9/50
27/27 [==============================] - 0s 6ms/step - loss: 0.1739
Epoch 10/50
27/27 [==============================] - 0s 8ms/step - loss: 0.1750
Epoch 11/50
27/27 [==============================] - 0s 8ms/step - loss: 0.1731
Epoch 12/50
27/27 [==============================] - 0s 8ms/step - loss: 0.1737
Epoch 13/50
27/27 [==============================] - 0s 9ms/step - loss: 0.1726
Epoch 14/50
27/27 [==============================] - 0s 10ms/step - loss: 0.1706
Epoch 15/50
27/27 [==============================] - 0s 7ms/step - loss: 0.1673
Epoch 16/50
27/27 [==============================] - 0s 7ms/step - loss: 0.1647
Epoch 17/50
27/27 [==============================] - 1s 22ms/step - loss: 0.1611
Epoch 18/50
27/27 [==============================] - 0s 18ms/step - loss: 0.1562
Epoch 19/50
27/27 [==============================] - 0s 8ms/step - loss: 0.1511
Epoch 20/50
27/27 [==============================] - 0s 11ms/step - loss: 0.1468
Epoch 21/50
27/27 [==============================] - 0s 7ms/step - loss: 0.1498
Epoch 22/50
27/27 [==============================] - 0s 7ms/step - loss: 0.1333
Epoch 23/50
27/27 [==============================] - 0s 7ms/step - loss: 0.1234
Epoch 24/50
27/27 [==============================] - 0s 7ms/step - loss: 0.1134
Epoch 25/50
27/27 [==============================] - 0s 7ms/step - loss: 0.1005
Epoch 26/50
27/27 [==============================] - 0s 7ms/step - loss: 0.0892
Epoch 27/50
27/27 [==============================] - 0s 7ms/step - loss: 0.0831
Epoch 28/50
27/27 [==============================] - 0s 8ms/step - loss: 0.0710
Epoch 29/50
27/27 [==============================] - 0s 9ms/step - loss: 0.0690
Epoch 30/50
27/27 [==============================] - 0s 10ms/step - loss: 0.0667
Epoch 31/50
27/27 [==============================] - 0s 7ms/step - loss: 0.0729
Epoch 32/50
27/27 [==============================] - 0s 7ms/step - loss: 0.0649
Epoch 33/50
27/27 [==============================] - 0s 7ms/step - loss: 0.0631
Epoch 34/50
27/27 [==============================] - 0s 7ms/step - loss: 0.0615
Epoch 35/50
27/27 [==============================] - 0s 7ms/step - loss: 0.0608
Epoch 36/50
27/27 [==============================] - 0s 6ms/step - loss: 0.0623
Epoch 37/50
27/27 [==============================] - 0s 8ms/step - loss: 0.0625
Epoch 38/50
27/27 [==============================] - 0s 8ms/step - loss: 0.0594
Epoch 39/50
27/27 [==============================] - 0s 8ms/step - loss: 0.0588
Epoch 40/50
27/27 [==============================] - 0s 9ms/step - loss: 0.0569
Epoch 41/50
27/27 [==============================] - 0s 7ms/step - loss: 0.0589
Epoch 42/50
27/27 [==============================] - 0s 7ms/step - loss: 0.0546
```


Şekil 4.9 Epoch Değerleri

```
Epoch 43/50
27/27 [==============================] - 0s 10ms/step - loss: 0.0507
Epoch 44/50
27/27 [==============================] - 1s 20ms/step - loss: 0.0507
Epoch 45/50
27/27 [==============================] - 0s 10ms/step - loss: 0.0479
Epoch 46/50
27/27 [==============================] - 0s 8ms/step - loss: 0.0513
Epoch 47/50
27/27 [==============================] - 0s 9ms/step - loss: 0.0503
Epoch 48/50
27/27 [==============================] - 0s 7ms/step - loss: 0.0456
Epoch 49/50
27/27 [==============================] - 0s 7ms/step - loss: 0.0445
Epoch 50/50
27/27 [==============================] - 0s 7ms/step - loss: 0.0414
```

Out[19]:

```
<keras.callbacks.History at 0x2b83eb67910>
```

In [20]:

```
"""
LSTM modelimizin test hatasını değerlendiriyoruz.
"""
loss = model.evaluate(X_test, y_test)
print("LSTM Model Test Hatası:", loss)
```

```
12/12 [==============================] - 1s 3ms/step - loss: 0.0638
LSTM Model Test Hatası: 0.06381445378065109
```

In [21]:

```
"""
LSTM modelimizdeki verileri eğittikten sonra bu modeli test edip test sonuçlarını 3 farklı şekilde ekrana basıyoruz. Bu metrik
sonuçları mse, mae ve R2 skorlarıdır.
"""
y_pred_lstm = model.predict(X_test)
mse_lstm = mean_squared_error(y_test, y_pred_lstm)
print("LSTM Modeli Ortalama Kareler Hata:", mse_lstm)

mae_lstm = mean_absolute_error(y_test, y_pred_lstm)
print("LSTM Modeli Ortalama Mutlak Hata:", mae_lstm)

r2_lstm = r2_score(y_test, y_pred_lstm)
print("LSTM Modeli R^2 Skoru:", r2_lstm)
```

```
LSTM Modeli Ortalama Kareler Hata: 0.06381445208224724
LSTM Modeli Ortalama Mutlak Hata: 0.15229846398252234
LSTM Modeli R^2 Skoru: 0.5738859080900804
```

6- Diğer Makine Öğrenmesi Modelllerinin Hazırlanması

6.1- Lineer Regresyon

In [22]:

```
"""
Denetimli makine öğrenmesi algoritmalarına geldiğimizde ilk olarak modelimizde kullanacağımız Linear Regression fonksiyonunu
tanımlayıp ardından bu model fonksiyonuna eğitim verilerimizi gönderiyoruz, modelimiz eğitildikten sonra X_test değişkenimizle
model tahmini yapıyoruz.
"""
from sklearn.linear_model import LinearRegression

regression_model = LinearRegression()
regression_model.fit(X_train, y_train)
y_pred_lr = regression_model.predict(X_test)
```


Şekil 4.10 LSTM Modeli Sonuçları ve Diğer Makine Öğrenmesi Modellerinin Hazırlanması

In [23]:

```
"""
Yapılan model tahmininin ardından ne kadar başarılı bir model olduğunu y_test hedef değişken değerleri ile ölçüyoruz, yine
model ölçüm metrikleri olarak mse, mae ve R2 skorları kullanılıyor.
"""
mse_lr = mean_squared_error(y_test, y_pred_lr)
print("Lineer Regresyon Modeli Ortalama Kareler Hata:", mse_lr)

mae_lr = mean_absolute_error(y_test, y_pred_lr)
print("Lineer Regresyon Modeli Ortalama Mutlak Hata:", mae_lr)

r2_lr = r2_score(y_test, y_pred_lr)
print("Lineer Regresyon Modeli R^2 Skoru:", r2_lr)
```

```
Lineer Regresyon Modeli Ortalama Kareler Hata: 0.09846098408049965
Lineer Regresyon Modeli Ortalama Mutlak Hata: 0.22641687907873717
Lineer Regresyon Modeli R^2 Skoru: 0.3425374433060292
```

6.2- Polinomial Regresyon

In [24]:

```
"""
İkinci denetimli makine öğrenme modelimiz olan Polinomial Regresyon modelimizi verilerimiz için tamamlayıp ardından, degree
olarak 2 verip bu değerleri eğitiyoruz.
"""
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression

poly_features = PolynomialFeatures(degree=2)
X_poly = poly_features.fit_transform(X_train)

polynomial_model = LinearRegression()
polynomial_model.fit(X_poly, y_train)

X_test_poly = poly_features.transform(X_test)
y_pred_poly = polynomial_model.predict(X_test_poly)
```

In [25]:

```
"""
Eğitilen model üzerinde metrik sonuçlarına bir önceki model gibi ölçümlüyoruz.
"""
mse_poly = mean_squared_error(y_test, y_pred_poly)
print("Polinomial Regresyon Modeli Ortalama Kareler Hata:", mse_poly)

mae_poly = mean_absolute_error(y_test, y_pred_poly)
print("Polinomial Regresyon Modeli Ortalama Mutlak Hata:", mae_poly)

r2_poly = r2_score(y_test, y_pred_poly)
print("Polinomial Regresyon Modeli R^2 Skoru:", r2_poly)
```

```
Polinomial Regresyon Modeli Ortalama Kareler Hata: 0.032677143596916904
Polinomial Regresyon Modeli Ortalama Mutlak Hata: 0.07907717441496223
Polinomial Regresyon Modeli R^2 Skoru: 0.781801912957521
```

6.3- Karar Ağaçları Regresyonu

In [26]:

```
"""
Karar ağaçları modelimize geldiğimizde, öncelikle algoritma model fonksiyonunun kütüphanesini sisteme tanımlayıp ardından
tree_model değişkenimize bu modeli tanımlıyoruz. Daha sonra X_train ve y_train değişkenlerini, oluşturduğumuz modele
gönderiyoruz ve ardından modelimizi eğitiyoruz.
"""
from sklearn.tree import DecisionTreeRegressor

tree_model = DecisionTreeRegressor()
tree_model.fit(X_train, y_train)
y_pred_tree = tree_model.predict(X_test)
```


Şekil 4.11 Polinomal ve Karar Ağaçları Regresyonları Modellerinin Analizleri

In [27]:

```
"""
Sonrasında model metrik değerlendirmelerimizi karar ağacı modelimiz için ölçüyoruz.
"""
mse_tree = mean_squared_error(y_test, y_pred_tree)
print("Karar Ağaçları Modeli Ortalama Kareler Hata:", mse_tree)

mae_tree = mean_absolute_error(y_test, y_pred_tree)
print("Karar Ağaçları Modeli Ortalama Mutlak Hata:", mae_tree)

r2_tree = r2_score(y_test, y_pred_tree)
print("Karar Ağaçları Modeli R^2 Skoru:", r2_tree)
```

```
Karar Ağaçları Modeli Ortalama Kareler Hata: 0.007574931880109005
Karar Ağaçları Modeli Ortalama Mutlak Hata: 0.02506811989100906
Karar Ağaçları Modeli R^2 Skoru: 0.9494192128263983
```

6.4- Rassal Ağaçlar Regresyonu

In [28]:

```
"""
Son olarak Rassal Ağaçlar modelimizin kütüphanesini tanımlıyoruz ve sonrasında bu modelimize eğitim değerlerimizi göndererek
modelimizi eğitiyoruz.
"""
from sklearn.ensemble import RandomForestRegressor

rf_model = RandomForestRegressor()
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)
```

In [29]:

```
"""
Diğer makine öğrenmesi modelleri gibi eğittiğimiz modelimiz üzerinde metrik sonuçlarımıza ölçüyoruz.
"""
mse_rf = mean_squared_error(y_test, y_pred_rf)
print("Rassal Ağaçlar Modeli Ortalama Kareler Hata:", mse_rf)

mae_rf = mean_absolute_error(y_test, y_pred_rf)
print("Rassal Ağaçlar Modeli Ortalama Mutlak Hata:", mae_rf)

r2_rf = r2_score(y_test, y_pred_rf)
print("Rassal Ağaçlar Modeli R^2 Skoru:", r2_rf)
```

```
Rassal Ağaçlar Modeli Ortalama Kareler Hata: 0.006038871934604883
Rassal Ağaçlar Modeli Ortalama Mutlak Hata: 0.02279019073569664
Rassal Ağaçlar Modeli R^2 Skoru: 0.959676086738817
```

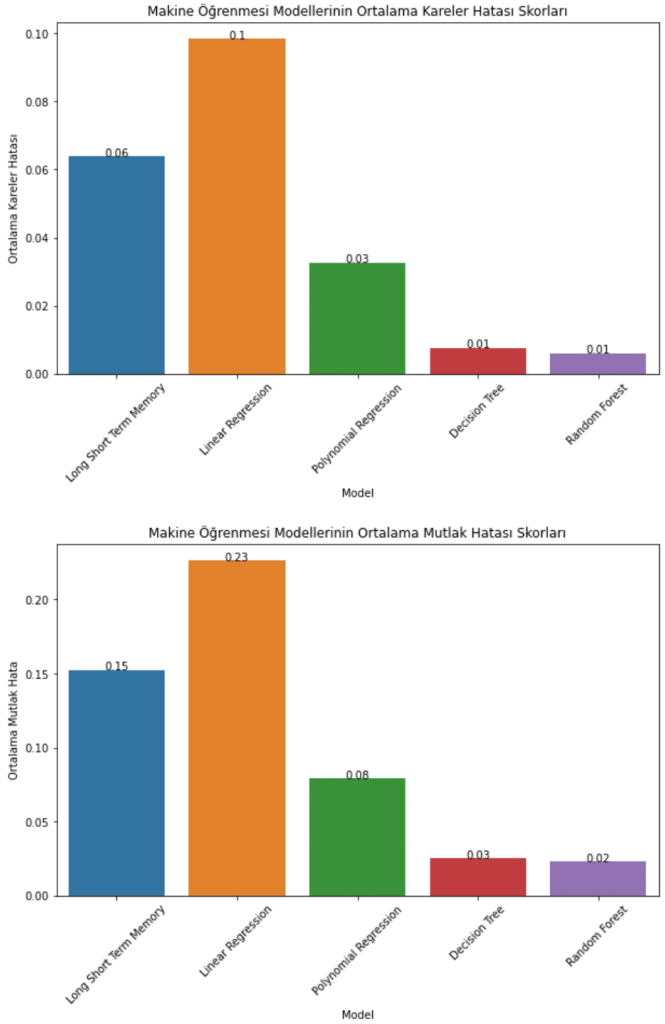
7- Metrik Sonuçlarının Görselleştirilmesi

Şekil 4.12 Rassal Ağaçlar Regresyonu Modeli Analizi

In [30]:

```
"""
Son aşamada bütün makine öğrenmesi ve derin öğrenme algoritmalarımızın metrik sonuçlarını beraberce görselleştiriyoruz.
"""
import seaborn as sns

"""
Model isimlerinin tanımlanması
"""
model_names = ['Long Short Term Memory','Linear Regression', 'Polynomial Regression', 'Decision Tree', 'Random Forest']

"""
Hata metriklerinin liste haline getirilmesi
"""
mse_scores = [mse_lstm, mse_lr, mse_poly, mse_tree, mse_rf]
mae_scores = [mae_lstm, mae_lr, mae_poly, mae_tree, mae_rf]
r2_scores = [r2_lstm, r2_lr, r2_poly, r2_tree, r2_rf]

"""
Metrik listelerimizi data frame haline getiriyoruz.
"""
df_scores = pd.DataFrame({'Model': model_names, 'MSE': mse_scores, 'MAE': mae_scores, 'R2': r2_scores})

"""
Grafikleri oluşturuyoruz
"""
plt.figure(figsize=(10, 6))
sns.barplot(x='Model', y='MSE', data=df_scores)
plt.xlabel('Model')
plt.ylabel('Ortalama Kareler Hatası')
plt.title('Makine Öğrenmesi Modellerinin Ortalama Kareler Hatası Skorları')
plt.xticks(rotation=45)

"""
Bar chartlarımızın değerlerini göstermek için matplotlib içerisindeki text fonksiyonunu kullanıyoruz.
"""
for index, row in df_scores.iterrows():
    plt.text(index, row['MSE'], str(round(row['MSE'], 2)), color='black', ha="center")

plt.show()


plt.figure(figsize=(10, 6))
sns.barplot(x='Model', y='MAE', data=df_scores)
plt.xlabel('Model')
plt.ylabel('Ortalama Mutlak Hata')
plt.title('Makine Öğrenmesi Modellerinin Ortalama Mutlak Hatası Skorları')
plt.xticks(rotation=45)

for index, row in df_scores.iterrows():
    plt.text(index, row['MAE'], str(round(row['MAE'], 2)), color='black', ha="center")

plt.show()


plt.figure(figsize=(10, 6))
sns.barplot(x='Model', y='R2', data=df_scores)
plt.xlabel('Model')
plt.ylabel('R2')
plt.title('Makine Öğrenmesi Modellerinin R2 Skorları')
plt.xticks(rotation=45)

for index, row in df_scores.iterrows():
    plt.text(index, row['R2'], str(round(row['R2'], 2)), color='black', ha="center")

plt.show()
```

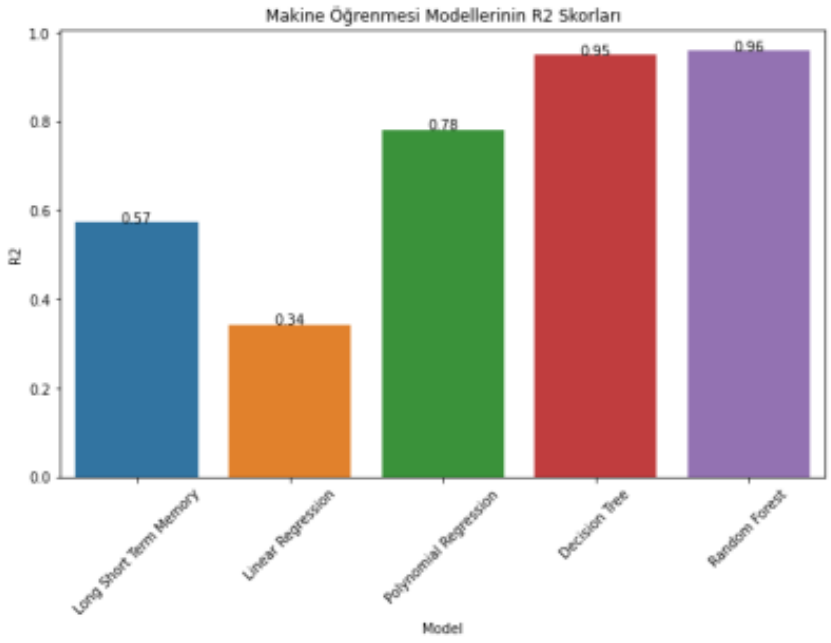

Şekil 4.13 Metrik Sonuçlarının Görselleştirilmesi ile İlgili Kodlar



Şekil 4.14 Modellerin Ortalama Kareler Hatası ve Ortalama Mutlak Hatası Skorları Karşılaştırılmaları

In [31]:

```
plt.figure(figsize=(10, 6))
sns.barplot(x='Model', y='R2', data=df_scores)
plt.xlabel('Model')
plt.ylabel('R2')
plt.title('Makine Öğrenmesi Modellerinin R2 Skorları')
plt.xticks(rotation=45)

for index, row in df_scores.iterrows():
    plt.text(index, row['R2'], str(round(row['R2'], 2)), color='black', ha="center")

plt.show()
```



In [ ]:

Şekil 4.15 Modellerin R^2 Skorları Karşılaştırılmaları

SONUÇ VE DEĞERLENDİRME

Bu çalışmamızda deprem büyüklüğünün tahmin edilmesi konusunda yapay zekâ ve makine öğrenimi tekniklerinin etkinliğini incelenmiştir. Yapay zekâ ve makine öğrenimi yöntemlerinin kullanılabilirliği ve etkilerinin değerlendirildiği bu çalışma, deprem risk yönetimi ve felaket yönetimi gibi alanlarda daha doğru tahminler yapılmasına ve etkili önlemler alınmasına katkı sağlamayı hedeflemektedir. Bu bağlamda çalışmada derin öğrenme (LSTM) ve makine öğrenmesi (LR, PLR, DTR, RFR) yöntemleri kullanılarak bir model önerilmiştir. Söz konusu model ile farklı veri seti değerleri eğitilerek gerçekleştirilmiştir. Sonuçlar %98 R^2 skoruyla Rassal Ağaçlar modelinin çalıştığını saptamıştır. Ayrıca, modelin performansı veri sayısıyla doğru orantılıdır. Daha fazla verinin kullanılması performans üzerinde olumlu etkiye sahiptir. Elde edilen bulgular, önerilen RFR modelinin hibrit yaklaşımlarla biraz daha geliştirilerek deprem bilimi alanında rahatlıkla kullanılabileceğini ortaya koymaktadır. Çalışmanın bir sonraki aşamasında, daha yüksek başarı oranları elde etmek için farklı

Algoritma veya teknikler de kullanılabilir. Ayrıca, çalışmanın konusu deprem bilimi olması nedeniyle bu alandaki uzmanların da görüşleri alınarak elde edilen sonuçlar hakkında daha doğru yorumlar yapılabilir. Sonuç olarak, önerilen modelin kısa sürede, düşük hata oranı ile ve minimum maliyetle tespit yapabilmesi, yapay zekâ tekniklerinin deprem bilimi alanında kullanılma potansiyelini ortaya koymaktadır.

Kaynaklar

[1] Kaftan, İlknur, Elçin G. İzmir ve Çevresine Ait Zemin Özelliklerinin Yapay Sinir Ağları ile İncelenmesi. 2. Türkiye Deprem Mühendisliği ve Sismoloji Konferansı; 2013 Eylül 25-27; Hatay. ss. 1-6.

[2] Martinez, Alvarez, Francisco, Jorge Reyes, Antonio Morales, Esteban, Cristina Rubio, Escudero. Determining The Best Set of Seismicity Indicators to Predict Earthquakes. Two Case Studies: Chile and The Iberian Peninsula. Knowledge-Based Systems; 2013; 50. pp. 198-210.

[3] Reyes, Juan, Antonio Morales, Esteban, Francisco Martinez, Alvarez. Neural Networks to Predict Earthquakes in Chile. Applied Soft Computing; 2013; 13(2), pp.1314-1328.

[4] Çelik, Enes, Muhammet Atalay, Harun Bayer. Yapay Sinir Ağları ve Destek Vektör Makineleri ile Deprem Tahmininde Sismik Darbelerin Kullanılması. IEEE 22nd Signal Processing And Communications Applications Conference; 2014 Nisan 23-25; Trabzon. ss. 730-733.

[5] Medium. Doğrusal Regresyon(Linear Regression)[İnternet]. 2020 [erişim tarihi 15.04.2020]. https://bernatas.medium.com/do%C4%9Frusal-regresyon-linear-regression-8f562c19aadf

[6] Medium. Polinomsal(Polynomial) Regresyon ve Python Uygulaması[İnternet]. 2020 [erişim tarihi 23.05.2020]. https://yigitsener.medium.com/polinomsal-polynomial-regresyon-ve-python-uygulamas%C4%B1-f742fb61a158

[7] Medium. Karar Ağaçları(Makine Öğrenmesi Serisi-3)[İnternet]. 2020 [erişim tarihi 07.09.2020]. https://medium.com/deep-learning-turkiye/karar-a%C4%9Fa%C3%A7lar%C4%B1-makine-%C3%B6%C4%9Frenmesi-serisi-3-a03f3ff00ba5

[8] Medium. Karar Teorisi, Karar Ağaçları, Rassal Orman ve Ensemble Learning[İnternet]. 2019 [erişim tarihi 31.12.2019]. https://medium.com/deep-learning-turkiye/karar-teorisi-karar-a%C4%9Fa%C3%A7lar%C4%B1-rassal-orman-ve-ensemble-learning-d05845dce28e

[9] IBM. Ortalama Karesi Hatası[İnternet]. 2023 [erişim tarihi 02.05.2023]. https://www.ibm.com/docs/tr/cloud-paks/cp-data/4.6.x?topic=overview-mean-squared-error

[10] Medium. Model Performansını Değerlendirmek[İnternet]. 2021 [erişim tarihi 11.02.2021]. https://medium.com/yaz%C4%B1l%C4%B1m-ve-bili%C5%9Fim-kul%C3%BCb%C3%BC/model-performans%C4%B1n%C4%B1-de%C4%9Ferlendirmek-regresyon-48b4afec8664

Özgeçmiş

Adı Soyadı: Çağlar Özkören

Eğitim:

2001–2005 Manisa Celal Bayar Üniversitesi, İnşaat Müh. Bölümü

İş Deneyimi:

2005 – 2011 Ünlü Çelik Yapı Denetim Şirketi – Elsa Yapı Denetim Şirketi

2011 – 2013 Urla Belediyesi

2013 – …… İzmir Büyükşehir Belediyesi