

## MAKİNE ÖĞRENMESİ İLE ARACI KURUM HİSSE SENEDİ ÖNERİLERİNİN TUTARLILIK ANALİZİ: BORSA İSTANBUL ÖRNEĞİ

Yasin Kubilay TINMAZ<sup>1</sup>  
Dr. Öğr. Üyesi Nuri BİNGÖL<sup>2</sup>

### ÖZET

Bu araştırmanın amacı, aracı kurumların Borsa İstanbul'daki hisse senedi önerilerinin gerçekleşen piyasa verileriyle ne ölçüde örtüştüğünü analiz etmek ve bu öngörülerin makine öğrenmesi yöntemleriyle tutarlılığını değerlendirmektir. Bu doğrultuda, doğrudan fiyat seviyeleri yerine açılış fiyatına göre yüksek, düşük ve kapanış fiyatlarındaki yüzdesel değişimler tahmin edilerek, fiyatların enflasyon, hisse bölünmeleri ve uzun vadeli piyasa trendlerinden kaynaklanan etkileri minimize edilmiştir. Çalışmada kullanılan veri seti, Ziraat Yatırım'ın 2020-2021 dönemine ait günlük raporlarından Optik Karakter Tanıma (Optical Character Recognition OCR) yöntemiyle elde edilen ve doğrulama-temizleme süreçlerinden geçirilen 592 gözlemden oluşmaktadır. Eğitim sürecinde bu veri kullanılırken, metodolojinin genel çapta ele alınabilmesini test etmek amacıyla Gedik Yatırım verileri dış test seti olarak değerlendirilmiştir. Eğitim aşamasında veri dengesizliği, SMOTE-ENN yöntemiyle giderilmiş; T+1 gününe ilişkin getiriler Random Forest ve XGBoost algoritmaları kullanılarak öngörülmüştür. Sürecin genel yapısı, OCR ile hisse kodlarının çıkarılması ve doğrulanması, fiyat verilerinin eşleştirilmesi, bağımsız ve bağımlı değişkenlerin tanımlanması, veri dengesizliğinin giderilmesi, modellerin eğitilmesi ve performanslarının değerlendirilmesi adımlarından oluşmuştur. Böylece veri, rastlantısal değil sistematik biçimde işlenerek analiz için uygun hale getirilmiştir. Araştırmanın bulguları, Random Forest algoritmasının iki test setinde de XGBoost'a göre daha düşük hata metrikleri ürettiği sonucuna varmıştır.

**Anahtar Kelimeler:** Random Forest, XGBoost, SMOTEENN, OCR, Hisse Senedi

<sup>1</sup> Üsküdar Üniversitesi, Fen Bilimleri Enstitüsü, Yapay Zeka Mühendisliği, İstanbul, Türkiye – 0009-0008-1445-4931  
yasinkubilay.tinmaz@st.uskudar.edu.tr

<sup>2</sup> Üsküdar Üniversitesi, Fen Bilimleri Enstitüsü, Yapay Zeka Mühendisliği, İstanbul, Türkiye – 0000-0001-6208-7277  
nuri.bingol@uskudar.edu.tr

## CONSISTENCY ANALYSIS OF BROKERAGE FIRM STOCK RECOMMENDATIONS USING MACHINE LEARNING: THE CASE OF BORSA ISTANBUL

### ABSTRACT

The purpose of this study is to analyze the extent to which brokerage firms' stock recommendations on Borsa Istanbul align with actual market data and to evaluate the consistency of these forecasts using machine learning methods. Instead of predicting direct price levels, the study estimates the percentage changes in high, low, and closing prices relative to the opening price, thereby minimizing the effects of inflation, stock splits, and long-term market trends. The dataset consists of 592 observations extracted from Ziraat Investment's daily reports for the 2020–2021 period through Optical Character Recognition (OCR) and subsequently verified and cleaned. While this dataset was used for model training, data from Gedik Investment were employed as an external test set to assess the generalizability of the methodology. During the training phase, data imbalance was addressed using the SMOTE-ENN method, and one-day-ahead (T+1) returns were predicted using Random Forest and XGBoost algorithms. The overall process included extracting and validating stock codes via OCR, matching price data, defining independent and dependent variables, handling data imbalance, training models, and evaluating their performance. As a result, the data were systematically processed rather than randomly handled to ensure analytical suitability. The findings reveal that the Random Forest algorithm produced lower error metrics than XGBoost across both test sets.

**Keywords:** Random Forest, XGBoost, SMOTEENN, OCR, Stock Market

### GİRİŞ

Finansal piyasalar, yüksek getiri potansiyelleriyle cazip olmakla birlikte, volatilité ve öngörülemezlik gibi yapısal özellikleri nedeniyle yatırım kararlarını güçleştiren dinamik bir yapıya sahiptir. Bu nedenle hisse senedi fiyatlarının tahmini, uzun süredir akademik çalışmaların odak noktalarından biri olmuştur. Son yıllarda, bu alanda makine öğrenmesi yöntemlerinin yaygın biçimde kullanılmaya başlandığı görülmektedir. Makine öğrenmesi, veriden öğrenerek örüntüleri tanımlayan ve değişkenler arasındaki karmaşık ilişkileri modelleyebilen istatistiksel algoritmalar bütünüdür (Rouf vd., 2021). Bu yöntemler, finansal veriler üzerinden yapılam tahminlerde, klasik ekonometrik veya istatistiksel yaklaşımlara kıyasla daha istikrarlı sonuçlar üretebilme potansiyeline sahiptir.

Analist tahminlerinin doğruluğunu ölçmeye yönelik araştırmaların bir bölümü, son yıllarda makine öğrenmesi yaklaşımlarını da analize dahil etmiştir. Ou ve Wang (2024) analistlerin hedef fiyat tahminlerinin güvenilirliğini değerlendirmek amacıyla makine öğrenmesi modellerinden yararlanmış ve bu tahminlerin isabet düzeyini ampirik olarak incelemiştir. Tekin ve Çanakoğlu (2018) çalışmasında ise Borsa İstanbul'daki hisse senedi getirileri Naïve Bayes ve yapay sinir ağları gibi hem klasik hem de makine öğrenmesi temelli yöntemlerle tahmin edilmiş; sonuçlarda Naïve Bayes algoritması %92,28 doğruluk oranıyla en yüksek başarıyı göstermiştir.

Bu sonuç, Türkiye özelinde makine öğrenmesi yöntemlerinin finansal tahminlerde etkili biçimde uygulanabileceğini ortaya koyan önemli bir örnek olarak değerlendirilebilir. Ayyıldız ve İskenderoğlu (2024) çalışmasında, çeşitli algoritmaların piyasa yönü tahminindeki performansları karşılaştırılmış ve model seçiminin doğruluk üzerinde belirleyici bir faktör olduğu vurgulanmıştır.

Makine öğrenmesi yöntemleri, özellikle finansal zaman serilerinde Geleneksel istatistiksel modellerin yakalayamadığı karmaşık ve doğrusal olmayan ilişkileri temsil etme gücüyle ön plana çıkmaktadır. Subasi vd. (2021) araştırmasında, farklı sınıflandırma algoritmalarının hisse senedi piyasası yön tahminlerindeki performanslar karşılaştırılmış ve belirli koşullarda makine öğrenmesi yöntemlerinin klasik yaklaşımlara kıyasla daha yüksek doğruluk sergileyebildiği ortaya koyulmuştur. Saberironaghi vd. (2025) finansal piyasa tahminlerinde makine öğrenmesi ve derin öğrenme yöntemlerini sistematik biçimde değerlendirmiş; veri kalitesi, modelin yorumlanabilirliği ve performans göstergeleri açısından kapsamlı bir analiz sunarak bu tekniklerin avantajlarını ve sınırlılıklarını ortaya koymuştur.

Türkiye finansal piyasalarında makine öğrenmesi uygulamaları özellikle Borsa İstanbul bağlamında yön tahmini odaklı çalışmalarda yoğunlaşmaktadır. XGBoost, K-En Yakın Komşu (K-Nearest Neighbors KNN) ve Destek Vektör Makineleri (Support Vector Machine SVM) gibi algoritmalar BIST endeks hareketlerinin tahmininde kullanılmış; bu araştırmalar, model başarısının büyük ölçüde yöntemi seçimi ve veri ön işleme süreçlerinin niteliğine bağlı olduğunu ortaya koymuştur (Ayyıldız ve Iskenderoglu, 2023).

Bu çalışma kapsamında, doğrudan fiyat seviyeleri tahmin edilmek yerine, açılış fiyatına göre yüksek, düşük ve kapanış fiyatlarındaki yüzdesel değişimler öngörülmüştür. Bu yaklaşım, fiyatların zaman içinde enflasyon, hisse bölünmeleri ve uzun vadeli piyasa eğilimlerinden kaynaklanan etkilerini minimize ederek, modellerin farklı dönemlerde daha istikrarlı ve güvenilir sonuçlar üretmesine olanak sağlamaktadır. Çalışmada kullanılan veri seti, aracı kurumların günlük raporlarının görsel formatlarından OCR yöntemiyle elde edilmiştir. Böylece, görsel nitelikli raporların metinsel veriye dönüştürülmesi mümkün olmuş ve aracı kurum tavsiyelerinin makine öğrenmesi algoritmalarıyla analiz edilebilmesi için özgün bir veri işleme altyapısı geliştirilmiştir.

### 1.1. Makine Öğrenmesi

Yapay zekanın bir alt disiplini olan makine öğrenmesi, belirli bir problem ilişkin örüntüleri tanımlama, sınıflandırma ya da sonuç tahmininde bulunma süreçlerini, veriler ve algoritmalar aracılığıyla istatistiksel yöntemler kullanılarak gerçekleştiren bir bilim alanıdır. Bu yaklaşımda, verilerden elde edilen modeller sayesinde sistemin öğrenme ve tahmin yapma kabiliyeti geliştirilir. Makine öğrenmesi süreci, görüntü, ses, metin veya sayısal veri gibi farklı veri türlerinin toplanması, ön işleme aşamasından geçirilmesi, uygun öğrenme modelinin uygulanması, test edilmesi ve elde edilen sonuçların tahmin edilmesiyle tamamlanan döngüsel bir yapıya sahiptir (Shobha ve Rangaswamy, 2018).

Makine öğrenmesi yöntemleri, kullanılan eğitim verisinin niteliği ve geri bildirimin biçimine göre farklı kategorilere ayrılmaktadır. Literatürde üç temel öğrenme türü öne çıkmaktadır. Denetimli öğrenme, denetimsiz öğrenme ve pekiştirmeli öğrenme olarak nitelendirilir (Alpaydın,

2014; Murphy, 2012). Bunun yanı sıra, sınırlı sayıda etiketli verinin bulunduğu durumlarda başvuru yarı denetimli öğrenme gibi ara yaklaşımlar da geliştirilmiştir (Chapelle ve ark., 2006)

Denetimli öğrenme, doğru çıktıları önceden tanımlanmış etiketli bir eğitim verisi kullanılarak modelin eğitildiği makine öğrenmesi yaklaşımıdır (Bishop, 2006; Murphy, 2012). Bu yöntemde her bir eğitim örneği, bir girdi verisi ile ona karşılık gelen çıktı değeri çiftinden oluşur. Algoritma, girdi ile çıktı arasındaki fonksiyonel ilişkiyi öğrenmeye çalışır ve bu sayede daha önce görülmemiş girdilere doğru tahminler yapabilen bir model ortaya çıkar. Denetimli öğrenmenin amacı, verilen örneklerdeki örtük bağıntıları yakalayarak girdi uzayından çıktı uzayına bir eşleme fonksiyonu öğrenmektir (Mitchell, 1997). Bu doğrultuda modelin başarısı, ürettiği tahminlerin gerçek etiketlere ne ölçüde yakın olduğuyla değerlendirilir.

Formel olarak denetimli öğrenmede, elimizde “ $\sum_{i=1}^N L(f(x_i), y_i)$ ” biçiminde “ $N$ ” örnekten oluşan bir eğitim veri kümesi bulunmaktadır. Burada “ $x_i$ ” her bir örneğin girdi vektörünü; “ $y_i$ ” ise beklenen çıktısını ifade etmektedir. Öğrenme algoritması, belirli bir hipotez evreninden “ $(f(x_i), y_i)$ ” biçiminde bir hipotez fonksiyonu seçerek gerçek ilişkiyi mümkün olduğunca doğru biçimde yaklaştırmaya çalışır. Modelin öğrenme süreci çoğunlukla bir hata fonksiyonunun en aza indirilmesine dayanır (Hastie ve ark., 2009). Örneğin, ortalama kare hata gibi bir kayıp fonksiyonu kullanıldığında ampirik risk minimizasyonu yaklaşımı denklemde formüle edilmektedir.

$$J(f) = \frac{1}{N} \sum_{i=1}^N L(f(x_i), y_i) \quad (1)$$

Burada “ $L(f(x_i), y_i)$ ” ifadesi, modelin “ $x_i$ ” için ürettiği tahmin ile gerçek “ $y_i$ ” değeri arasındaki kaybı ölçmektedir. Eğitim algoritması, “ $J(f)$ ” fonksiyonunu en küçükleyen optimal parametreleri ya da fonksiyonu bularak modeli öğrenmektedir (Hastie ve ark., 2009). Bu süreçte gradyan inişi gibi optimizasyon yöntemleri yaygın olarak kullanılmakta ve model, eğitim verisi üzerinde mümkün olan en yüksek performansı sağlayacak biçimde uyarlanmaktadır.

Araştırmada kullanılan Random Forest ve XGBoost algoritmaları, denetimli öğrenme yaklaşımı kapsamında hem sınıflandırma hem de regresyon problemlerinde yaygın biçimde kullanılan yöntemlerle örtüşmektedir. Random Forest, çok sayıda karar ağacının birleşiminden oluşan bir topluluk modelidir. Her bir ağaç, bootstrap yöntemiyle rastgele seçilmiş örnekler üzerinde eğitilir ve her düğümde rastgele belirlenen bir alt özellik kümesi üzerinden bölünme işlemi gerçekleştirilir. Elde edilen tüm ağaç tahminleri ortalanarak nihai regresyon çıktısı elde edilir. Bu yapı, tek bir karar ağacında sıkça görülen aşırı öğrenme riskini azaltarak modelin genelleme yeteneğini güçlendirir (Breiman, 2001). Buna karşın XGBoost algoritması, gradyan artırma prensibine dayalı gelişmiş bir karar ağacı topluluğu yöntemidir. Bu yöntemde her yineleme, bir önceki modelin kalıntı hatalarını öğrenerek yeni bir ağaç ekler; böylece hatlaar kademeli biçimde azaltılır XGBoost, seyrek veri yapıları, eksik değerler ve hesaplama verimliliğini optimize eden mekanizmalarıyla hem yüksek doğruluk hem de işlem hızı sunmaktadır (Bentéjac vd., 2021; Chen ve Guestrin, 2016).

## 1.2. Araştırmanın Kurgusu

Çalışmanın ilk aşamasında, aracı kurumlara ait bültenlerden elde edilen görsel veriler işlenmiştir. Bu görsellerde yer alan hisse senedi isimlerinin otomatik olarak tanımlanabilmesi amacıyla OCR tekniği kullanılmış, bu işlemde EasyOCR kütüphanesinden yararlanılmıştır. Her bir görsele ait işlem tarihi, dosya adlarından düzenli ifadeler aracılığıyla otomatik olarak çıkarılmıştır. Metin tanıma doğruluğunu artırmak amacıyla çeşitli ön işleme adımları uygulanmıştır. Bu adımlar; gri tonlama, sabit eşikleme, keskinleştirme filtreleri, morfolojik açma ile gürültü azaltma ve adaptif Gaussian eşikleme işlemlerini içermektedir. Her bir ön işleme varyasyonu, ayrı ayrı OCR sürecinden geçirilmiş ve elde edilen metin çıktıları karşılaştırılarak en yüksek doğruluk oranı sağlayan kombinasyon belirlenmiştir.

```
def init_db(path: str):  
    conn = sqlite3.connect(path)  
    cur = conn.cursor()  
    CREATE TABLE IF NOT EXISTS ocr_ok (  
        id INTEGER PRIMARY KEY AUTOINCREMENT,  
        tarih TEXT NOT NULL,  
        hisse TEXT NOT NULL,  
        kaynak_dosya TEXT NOT NULL,  
        eklenme_zamani TEXT NOT NULL,  
        UNIQUE (tarih, hisse, kaynak_dosya)  
    )  
    CREATE TABLE IF NOT EXISTS ocr_hata (  
        id INTEGER PRIMARY KEY AUTOINCREMENT,  
        tarih TEXT,  
        kaynak_dosya TEXT NOT NULL,  
        satir_json TEXT NOT NULL,  
        hata_mesaji TEXT NOT NULL,  
        eklenme_zamani TEXT NOT NULL  
    )  
    CREATE TABLE IF NOT EXISTS ocr_duzeltme (  
        id INTEGER PRIMARY KEY AUTOINCREMENT,  
        kaynak_dosya TEXT NOT NULL,  
        tarih TEXT,  
        bulunan_hisseler_json TEXT NOT NULL,  
        yontem_ozeti_json TEXT NOT NULL,  
        eklenme_zamani TEXT NOT NULL,  
        UNIQUE (kaynak_dosya)  
    )  
    conn.commit()  
    conn.close()
```

Verilen fonksiyon, çalışmanın OCR sürecinde üretilen tüm ara çıktıları izlenebilir ve yeniden üretilebilir biçimde kaydetmek amacıyla gerekli veri tabanı altyapısını oluşturmaktadır. “*init\_db(path)*” fonksiyonu, SQLite bağlantısını kurarak üç tablo tanımlar: “*ocr\_ok*” tablosu, her doğrulanmış gözlem için tarih, hisse, kaynak dosya bilgilerinin yanı sıra eklenme zamanını saklar; burada tanımlanan “*UNIQUE*” kısıtı aynı görselin aynı hisseyle birden fazla kez kaydedilmesini engelleyerek sürecin bağımsız olmasını ve veri bütünlüğünün korunmasını sağlamaktadır. “*ocr\_hata*” fonksiyonu, başarısız OCR ya da doğrulama girişimlerini depolamaktadır. Hata bağlamı “*satir\_json*” alanında ham içerik biçiminde tutulur ve “*hata\_mesaji*” fonksiyonu ile birlikte zaman damgası eklenerek olumsuz örneklerin de izlenebilmesi, daha sonra yeniden işlenmesi veya yöntem karşılaştırmalarında kullanılabilmesini mümkün kılmaktadır. “*ocr\_duzeltme*” fonksiyonu, düzeltme ve iyileştirme denemelerini kayıt altına almaktadır. Bu fonksiyonlar sonucu oluşturulan şema, başarılı kayıtlar, hatalar ve düzeltme durumları için ayrı fakat ilişkili alanlar sağlayarak veri zincirini güçlendirmektedir.

### 1.3. Veri Kaynakları

Araştırmanın modelleme sürecinde, Ziraat Yatırım’ın 2020-2021 dönemine iat günlük hisse önerilerinden elde edilen veriler Eğitim aşamasında kullanılmış; modelin farklı veri kaynakları üzerindeki performansını değerlendirmek amacıyla Gedik Yatırım’a ait öneriler dış veri seti (external dataset) olarak test sürecinde uygulanmıştır. Ziraat Yatırım’ın uzun yıllardır Borsa İstanbul’da resmi bir aracı kurum olarak faaliyet göstermesi ve verilerini kamuya açık biçimde paylaşması, bu kurumun örneklem olarak seçilmesinde temel etken olmuştur.

**Tablo 1: OCR İşlemi Sonucu Hisse Tahmini Satırları Örneği**

| Hisse | Open  | High  | Low   | Close | High_Return | Low_Return | Close_Return |
|-------|-------|-------|-------|-------|-------------|------------|--------------|
| ARCLK | 18.15 | 18.58 | 17.96 | 18.44 | 2.39        | -1.06      | 1.57         |
| HALKB | 5.96  | 6.16  | 5.95  | 6.10  | 3.35        | -0.16      | 2.68         |
| DOAS  | 5.10  | 5.45  | 5.10  | 5.34  | 6.86        | 0.00       | 4.70         |
| KRDMD | 2.56  | 2.65  | 2.55  | 2.63  | 2.24        | -0.39      | 2.53         |
| VAKBN | 5.74  | 6.05  | 5.74  | 5.96  | 5.40        | 0.00       | 3.83         |

Çalışmada kullanılan veriler EasyOCR yöntemiyle okunmuş, OCR işlemleri sonucunda toplam 592 geçerli hisse tahmini satırı elde edilerek Tablo 1’de örneği verilmiştir.

|                                                                         |
|-------------------------------------------------------------------------|
| def get_next_trading_day(hisse, tarih):                                 |
| start = tarih + timedelta(days=1)                                       |
| end = tarih + timedelta(days=15)                                        |
| data = yf.download(hisse + ".IS", start=start, end=end, progress=False) |
| if data.empty:                                                          |
| return None, None                                                       |
| next_day = data.index[0].date()                                         |
| return next_day, data.loc[data.index[0]]                                |

Burada verilen fonksiyonlar, bir hisse kodu ve tarihe karşılık gelen sonraki ilk işlem gününü belirleyerek ilgili güne ait fiyat verilerini belirtmektedir. Öncelikle arama aralığı tarih+1 ile tarih+15 gün olarak tanımlanmakta, böylece hafta sonları ve resmi tatiller gibi işlem yapılmayan günler güvenli bir şekilde kapsam dışında bırakılmaktadır. Ardından, Borsa İstanbul sembol formatına uygun olarak hisse koduna “.IS” uzantısı eklenmekte ve Yahoo Finance üzerinden veri indirilmektedir. Zaman serisi boş döndüğünde fonksiyon (*None, None*) ile veri bulunamadığını açıkça göstermekte; veri mevcut olduğunda ise ilk kayıt “sonraki işlem günü” kabul edilmektedir. Belirlenen günün tarihi “*next\_day*” ve aynı güne ait tek satırlık fiyat ve hacim kaydı “*data.loc[data.index[0]]*” fonksiyon çıktısı ile iki öğeden oluşmaktadır. Bu yapı sayesinde ilgili kaydın “Open”, “High”, “Low”, “Close” ve “Volume” alanlarına doğrudan erişilerek T+1 bazlı getiriler ve karşılaştırmalar hesaplanabilmekte, ayrıca her zaman ilk geçerli günün alınması çalışmanın tutarlılık analizine katkı sağlamaktadır.

```

def save_to_db(results):
    conn = sqlite3.connect(db_path)
    cursor = conn.cursor()
    cursor.execute("""
CREATE TABLE IF NOT EXISTS ocr_yfinance (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    tarih TEXT NOT NULL,
    hisse TEXT NOT NULL,
    next_date TEXT,
    open REAL,
    high REAL,
    low REAL,
    close REAL,
    volume INTEGER
)
    """)
    for row in results:
        cursor.execute("""
INSERT INTO ocr_yfinance (tarih, hisse, next_date, open, high, low, close, volume)
VALUES (?, ?, ?, ?, ?, ?, ?, ?)
    """)
    conn.commit()

```

T+1 sorgularından elde edilen verileri tekilleştirilmiş ve analizlere uygun bir şema çerçevesinde SQLite veritabanına kalıcı olarak kaydetmek amacıyla söz konusu fonksiyon entegre edilmiştir. Öncelikle veritabanına bağlanılmakta ve “*ocr\_yfinance*” komutu ile mevcutsa korunan mevcut yapı, yoksa oluşturulan yeni tablo tanımlanmaktadır. Bu fonksiyonlarda her kayıt için artan birincil anahtar (id), OCR’den aktarılan tarih ve hisse kodu, ayrıca eşleştirilen sonraki işlem günü “*next\_date*” ile o güne ait OHLCV bilgileri tutulmaktadır. İşlemler “*commit*” komutuyla kalıcı hale getirilip bağlantı kapatılmaktadır.

Bu yapı sayesinde, üst akışta üretilen T+1 verileri standart bir şema altında toplanmakta, eksik değerler silinmeden işaretlenmiş biçimde veri kümesine aktarılmakta ve sonraki aşamalarda doğrudan kullanılabilir hale gelmektedir.

```

ef main(save_excel=True, save_db=True):
hisseler = fetch_hisseler()
results = []
for _, row in hisseler.iterrows():
tarih = datetime.strptime(row["tarih"], "%Y-%m-%d").date()
hisse = row["hisse"]
next_date, data = get_next_trading_day(hisse, tarih)
if data is not None:
results.append((
tarih,
hisse,
str(next_date),
float(data["Open"].iloc[0]),
float(data["High"].iloc[0]),
float(data["Low"].iloc[0]),
float(data["Close"].iloc[0]),
int(data["Volume"].iloc[0])
f"Low: {float(data['Low'].iloc[0])}, Close: {float(data['Close'].iloc[0])}, Volume: {int(data['Volume'].iloc[0])}"
))
results.append((tarih, hisse, None, None, None, None, None, None))
if save_db and results:
save_to_db(results)
if save_excel and results:
columns=["tarih","hisse","next_date","open","high","

```

Kullanılan fonksiyonlar, OCR sürecinden elde edilen tüm (tarih, hisse) çiftlerini sırasıyla işleyerek her biri için “*get\_next\_trading\_day*” fonksiyonu aracılığıyla T+1 ilk işlem gününü belirler ve sonuçları tek bir liste halinde toplar. Veri elde edilmesi durumunda, fonksiyon ilgili günün tarihini ve o güne ait tek satırlık fiyat/hacim kaydını birlikte döndürmekte, böylece bu satırın OHLCV alanları sonraki adımlarda doğrudan kullanılabilir.

Çalışmada tahmin edilmesi hedeflenen değişkenler, hisse senedi fiyatlarının açılış değerine göre günlük getiri oranları olarak tanımlanmıştır:

- $high\_return = (High - Open) / Open \times 100$
- $low\_return = (Low - Open) / Open \times 100$
- $close\_return = (Close - Open) / Open \times 100$

Bu üç oran, modelin öngörmeye çalıştığı bağımlı değişkenler olarak belirlenmiştir. Bağımsız değişkenler ise açılış fiyatı ve hisse isimlerinin one-hot encoding yöntemiyle dönüştürülmüş biçiminden oluşmaktadır. Böylelikle model, her hisseyi ayrı bir kategorik değişken olarak tanıyabilmiştir. Ayrıca, farklı ölçeklerden doğabilecek sapmaları önlemek amacıyla tüm özellikler “StandardScaler” ile standardize edilmiştir. Veri seti, %80 eğitim ve %20 test olmak üzere ikiye ayrılmış, böylece modeller yalnızca öğrenilen veriler üzerinde değil, daha önce hiç karşılaşmadıkları gözlemler üzerinde de sınanarak genelleme kabiliyetleri değerlendirilmiştir.

#### 1.4. Model Eğitimi

Bu Araştırmada modelin eğitimi yalnızca Ziraat Yatırım verileri kullanılarak gerçekleştirilmiş, Ancak modelin performansını ve kuramsal bağımsızlığını sınavabilmek amacıyla Gedik Yatırım verileri “out-of-sample test set” olarak kullanılmıştır. Bu yaklaşım, modelin yalnızca öğrenilen veri üzerinde değil, farklı aracı kurumlar ait veriler üzerinde de benzer performans gösterip göstermediğini test etme olanağı sağlamaktadır. Çalışmada iki farklı makine öğrenmesi algoritması uygulanmıştır. Random Forest Regressor, birden fazla karar ağacının birleşiminden oluşan ve aşırı öğrenme riskini azaltarak daha dengeli tahminler üreten bir yöntemdir; bu çalışmada “*n\_estimators=200*” ve “*max\_depth=6*” parametreleriyle uygulanmıştır. XGBoost Regressor ise özellikle büyük veri kümelerinde yüksek performans sergileyen, hatlar üzerinde yinelemeli öğrenme mantığına dayalı bir algoritmadır. Bu araştırmada “*n\_estimators=200*”, “*max\_depth=6*” ve “*learning\_rate=0.1*” parametreleriyle kullanılmıştır.

```
db_path = r"E:\projelerr\tez\ziraat_ocr.db"
# SQL'den veriyi oku
conn = sqlite3.connect(db_path)
df = pd.read_sql_query("SELECT hisse, open, high, low, close FROM ocr_yfinance",
conn)
conn.close()
```

Model eğitiminde veri yükleme aşaması, modelleme sürecinde kullanılacak piyasa verileri SQLite veri tabanından tek bir sorgu ile elde edilmiştir. Öncelikle veri tabanı dosyasının yolu tanımlanmış, ardından bağlantı açılarak “*ocr\_finance*” tablosundan yalnızca analiz için gerekli sütunlar seçilmiştir. Hisse kodu ile T+1 işlem gününe ait açılış, en yüksek, en düşük ve kapanış fiyatları belirlenmiştir. Sorgu, “*pandas.read\_sql\_query*” fonksiyonu kullanılarak doğrudan bir DataFrame’e aktarılmış ve bağlantı kapatılmıştır.

```
df["high_return"] = (df["high"] - df["open"]) / df["open"] * 100
df["low_return"] = (df["low"] - df["open"]) / df["open"] * 100
df["close_return"] = (df["close"] - df["open"]) / df["open"] * 100
# NaN temizliği
df = df.dropna()
```

Hedef değişkenlerin hesaplanmasında ise, modelin öngöreceği hedef değişkenler açılış fiyatına göre yüzdesel getiriler olarak tanımlanmıştır. Her bir satır için T+1 işlem gününe ait en yüksek, en düşük ve kapanış fiyatları açılış değerine oranlanarak yüzde cinsine dönüştürülmüştür. ( $high\_return = (high - open) / open \times 100$ ,  $low\_return = (low - open) / open \times 100$ ,  $close\_return = (close - open) / open \times 100$ ) Bu yöntemle farklı fiyat seviyelerine sahip hisseler ortak bir ölçeğe taşınmış; pozitif değerler gün içindeki yukarı yönlü, negatif değerler ise aşağı yönlü hareketleri ifade etmiştir. Üç hedef değişken, hem gün içi dalgalanmaları hem de gün sonu kapanış sonucunu birlikte yansıtarak modele daha zengin bir tahmin yapısı kazandırmıştır. Son aşamada, “*df.dropna*” fonksiyonu ile eksik değerlere sahip satırlar veri setinden çıkarılmış; böylece hesaplama hataları önlenmiş ve model eğitiminde sapma riskleri azaltılarak temiz ve bütünlüklü bir hedef veri kümesi oluşturulmuştur.

```
# One-hot encoding (hisse isimleri için)
df_encoded = pd.get_dummies(df, columns=["hisse"], drop_first=True)
# Bağımsız ve bağımlı değişkenler
X = df_encoded.drop(columns=["high", "low", "close", "high_return", "low_return",
"close_return"])
y = df[["high_return", "low_return", "close_return"]]
# Standardizasyon
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
# Eğitim / Test bölünmesi
X_train, X_test, y_train, y_test = train_test_split(
X_scaled, y, test_size=0.2, random_state=42)
```

Ön işleme aşamasında ise veri, modelleme sürecine uygun hale getirilmiştir. Öncelikle hisse kodu kategorik bir değişken olduğundan one-hot encoding yöntemi uygulanmış, böylece her hisse ayrı bir ikili sütunla temsil edilmiş ve gereksiz tekrarlar azaltılmıştır. Daha sonra özellik matrisi, ham fiyat sütunları ve hedef değişkenler hariç tutularak oluşturulmuştur. Dolayısıyla “X” yalnızca açılış fiyatı ve hisse kimliğini gösteren “dummy” sütunlarından oluşmaktadır. Hedef değişkenler ise üç sütundan meydana gelmektedir. Özellikler, “*StandardScaler*” kullanılarak ölçeklenmiş (her sütun yaklaşık 0 ortalama ve 1 standart sapmaya dönüştürülmüş) ve böylece farklı büyüklükteki değişkenlerin modele eşit ağırlıkla girmesi sağlanmıştır. Daha sonra veri, %80 eğitim ve %20 test seti olacak şekilde ayrılmış, “*random\_state=42*” parametresi kullanılarak aynı bölünmenin tekrarlanabilirliği güvence altına alınmıştır. Bu düzenleme ile model, açılış fiyatı ve hisse kimliği bilgisine dayalı olarak üç farklı getiri tahmin etmeye hazır hale getirilmiştir.

```

# Geçici DataFrame
train_df = pd.DataFrame(X_train)
train_df[["high_return", "low_return", "close_return"]] = y_train.values
train_df.columns = train_df.columns.astype(str)
train_df["target_class"] = (train_df["close_return"] > 0).astype(int)
# SMOTEENN uygula
smote_enn = SMOTEENN(random_state=42)
train_res, _ = smote_enn.fit_resample(
train_df.drop(columns=["target_class"]),
train_df["target_class"]
)
# Yeni X ve y
X_train_res = train_res.drop(columns=["high_return", "low_return",
"close_return"]).values
y_train_res = train_res[["high_return", "low_return", "close_return"]].values

```

SMOTEENN aşamasında eğitim verisi, kapanış getirisinin işaretine göre dengesizlikten arındırılarak modele daha dengeli örnekler sunacak şekilde yeniden yapılandırılmıştır. Öncelikle “*X\_train*” üzerinden geçici bir tablo oluşturulmuş ve bunun üzerine hedef sütunlar eklenmiş, ardından tüm sütun adları string formatına dönüştürülmüştür. Sonrasında yardımcı bir sınıf etiketi tanımlanmıştır “*target\_class = (close\_return > 0)*”. Bu etiket, günün pozitif kapanış yaptığı örnekler ile negatif kapanış yaptığı örnekleri ayırt etmek amacıyla kullanılmıştır. Dengesizliği gidermek için *SMOTEENN(random\_state=42)* yöntemi uygulanmıştır. Bu süreçte önce azınlık sınıf gözlemlerle artırılmış (SMOTE), ardından uç ya da hatalı noktalar temizlenmiştir (ENN). İşlem yalnızca eğitim verisi üzerinde gerçekleştirilmiş, “*train\_df.drop(columns=[“target\_class”])*” özellikleri ile “*train\_df[“target\_class”]*” etiketi kullanılmış, böylece test seti korunmuştur. Sonuçta elde edilen “*train\_res*” fonksiyonu, modellerde kullanılacak özellik matrisi ve hedefler ayrıştırılmıştır. Model en nihayetinde, hem pozitif hem de negatif günleri dengeli bir şekilde gözlemleyebilmiş, eğitim aşamasında tek tarafa yığılmadan kaynaklanabilecek öğrenme sorunları önlenmiştir.

```

models = {
"RandomForest": RandomForestRegressor(
n_estimators=200, max_depth=6, random_state=42
),
"XGBoost": xgb.XGBRegressor(
n_estimators=200, max_depth=6, learning_rate=0.1,
subsample=0.8, colsample_bytree=0.8, random_state=42
)
}

```

Random Forest Regressor ve XGBoost Regressor algoritmaları kullanılmıştır. Random Forest, çok sayıda karar verisini “bagging” yöntemiyle eğitmekte; “ $n\_estimators=200$ ” parametresi veri sayısını, “ $max\_depth=6$ ” ise her bir verinin karmaşıklık düzeyini sınırlamaktadır. “ $random\_state=42$ ” ise sonuçların yeniden üretilebilirliğini sağlamaktadır. XGBoost modeli ise ağaçları ardışık biçimde kurarak hataları kademeli olarak azaltmakta; “ $learning\_rate=0.1$ ” her adımın etkisini sınırlayarak aşırı uyum riskini azaltmakta, “ $subsample\_0.8$ ” ve “ $colsample\_bytree=0.8$ ” parametreleri her ağacı veri ve özelliklerin belirli bir alt kümesiyle eğiterek genelleme kabiliyetini artırmaktadır. Ayrıca “ $n\_estimators=200$ ” toplam ağaç sayısını, “ $max\_depth=6$ ” ağaçların derinlik sınırını belirlemekte ve “ $random\_state=42$ ” yine tekrarlanabilirlik sağlamaktadır. Seçilen parametreler, veri setinin boyutu ve hedef değişkenlerin yapısı dikkate alındığında hız, doğruluk ve aşırı öğrenmenin önlenmesi arasında dengeli bir çözüm sunmaktadır.

### 1.5. Performans Metrikleri

Modelin performans değerlendirmesi temel olarak kök ortalama kare hata ölçütü üzerinden gerçekleştirilmiştir. Kök Ortalama Kare Hatası (Root Mean Square Error RMSE) değerinin düşük çıkması, modelin tahminlerinin gerçek gözlemlerle daha yüksek doğrulukta örtüştüğünü göstermektedir. Bunun yanı sıra, karşılaştırmalı bir analiz elde edebilmek amacıyla Ortalama Mutlak Hata (Mean Absolute Error MAE) Ortalama Kare Hata (Mean Squared Error MSE) gibi tamamlayıcı performans göstergeleri de dikkate alınmıştır. MAE, tahmin edilen değerlerle gerçek değerler arasındaki mutlak farkların ortalamasını hesaplayarak modelin genel hata düzeyini yansıtırken; MSE, bu farkların karelerinin ortalamasını alarak özellikle büyük sapmalara karşı daha duyarlı bir değerlendirme sunmaktadır.

Sırasıyla MAE, RMSE ve MSE metriklerine ilişkin denklemler sunulmuştur (Koç, 2025).

$$MAE = \frac{1}{N} \sum_{i=1}^N |y_i - x_i| \quad (2)$$

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i^* - x_i)^2} \quad (3)$$

$$MSE = \left(\frac{1}{N}\right) \sum_{i=1}^N (x_i - x_i^*)^2 \quad (4)$$

Fiziksel tabanlı simülasyonlar veya makine öğrenmesi yöntemleri kullanıldığında, hata metrikleri, modellerin kalibrasyonu, eğitimi, doğrulanması ve geçerliliğinin değerlendirilmesinde temel ölçütler olarak öne çıkmaktadır. Bu metrikler aracılığıyla, modelin ürettiği tahminlerin doğruluk düzeyi nicel olarak belirlenebilmekte ve farklı modellerin performansları nesnel biçimde karşılaştırılabilmektedir. Böylece yalnızca hata büyüklükleri değil, aynı zamanda bu hataların dağılım özellikleri de incelenerek modelin genel başarımına ilişkin daha kapsamlı ve ayrıntılı bir değerlendirme yapılabilmektedir.

```

for name, model in models.items():
model.fit(X_train_res, y_train_res)
preds = model.predict(X_test)
rmse = np.sqrt(mean_squared_error(y_test, preds))
mae = np.mean(np.abs(y_test.values - preds))
mse = mean_squared_error(y_test, preds)

```

Performans metrikleri, her iki modeli sırasıyla eğiterek aynı test veri seti üzerinde değerlendirmektedir. “*model.fit(X\_train\_res, y\_train\_res)*” komutu ile eğitim, SMOTEENN yöntemiyle dengelenmiş eğitim verisi üzerinde gerçekleştirilmiştir, böylece modelin pozitif ve negatif günleri dengeli şekilde öğrenmesi sağlanmıştır. Daha sonra “*preds = model.predict(X\_test)*” ile tahminler, herhangi bir dengeleme uygulanmamış test kümesi üzerinde yapılmış; bu sayede eğitimdeki dengeleme etkisinin test verisine taşınmaması sağlanarak performansın daha gerçekçi ölçülmesi mümkün olmuştur. Üç hedef değişkenin aynı anda tahmin edilmesi sebebiyle metrikler çoklu çıktılar üzerinden tek bir değere indirgenmiştir. “*mean\_squared\_error(y\_test, preds)*” fonksiyonu, tüm test örnekleri ve üç sütun boyunca kare hata ortalamasını (MSE) vermektedir. RMSE bu değer karekökü olup hatayı tekrar yüzde puan ölçeğine taşır. MAE ise tahmin ile gerçek değerler arasındaki mutlak farkların ortalamasını temsil etmekte ve yine tüm test örnekleri ile üç sütun bir arada özetlenmektedir. Ekran yazdırılan sonuçlarda her model için RMSE, MAE ve MSE değerleri topluca verilmekte; böylece Random Forest ile XGBoost’un aynı koşullar altındaki performansları doğrudan karşılaştırılabilmektedir.

### 1.6. Model Performansı ve Analiz Sonuçları

Bu Araştırmada, İstanbul Menkul Kıymetler Borsası’nda işlem gören hisselerle yönelik olarak Ziraat Yatırım’ın 2020-2021 yıllarına ait günlük önerilerinden, OCR ve veri temizleme süreçleri sonucunda elde edilen 592 kayıttan oluşan veri seti kullanılarak T+1 gününe ait getiriler tahmin edilmiştir. Çalışma kapsamında, model performansının örnek düzeyinde incelenebilmesi amacıyla beş farklı hisseye ilişkin tahminler değerlendirilmiştir. Bu analizde amaç, test kümesinden rastgele seçilen beş gözleme ait gerçekleşen getiriler ile iki farklı makine öğrenmesi modelinin tahmin sonuçlarını karşılaştırmalı biçimde tablo halinde sunarak, modellerin tahmin doğruluğunu görsel ve nicel olarak ortaya koymaktır.

**Tablo 2: Ziraat Yatırım Beş Satırlık Hisse Karşılaştırma Örneği**

| Hisse | Open  | Gerçek<br>High% | RF<br>High<br>% | XGB<br>High<br>% | Gerçek<br>Low% | RF<br>Low<br>% | XGB<br>Low<br>% | Gerçek<br>Close% | RF<br>Close% | XGB<br>Close% | RF_<br>MSE |
|-------|-------|-----------------|-----------------|------------------|----------------|----------------|-----------------|------------------|--------------|---------------|------------|
| FROTO | 21.74 | 1.91            | 1.73            | 0.99             | -1.16          | -1.64          | -1.66           | 0.27             | -0.15        | -0.56         | 0.14       |
| ENKAI | 5.57  | 1.72            | 1.90            | 0.77             | -1.15          | -1.64          | -1.93           | 0.57             | -0.01        | -1.94         | 0.20       |
| TAVHL | 21,20 | 1.20            | 1.74            | 2.97             | -1.29          | -1.64          | -0.65           | -0.86            | -0.14        | 1.81          | 0.31       |
| EGEEN | 510.6 | 1.33            | 1.71            | 0.41             | -0.78          | -1.67          | -1.89           | -0.15            | -0.20        | -0.94         | 0.31       |
| YATAS | 13.71 | 2.05            | 1.91            | 1.50             | -0,68          | -1.63          | -1.16           | -0.34            | -0.01        | -0.20         | 0.34       |

Tablo 2’de sunulan beş farklı hisseye ait örnek tahminler incelendiğinde, her iki modelin de fiyat hareketlerinin yönünü genel olarak başarılı biçimde öngörebildiği, Ancak hata düzeylerinin hisse bazında değişkenlik gösterdiği görülmektedir. Özellikle FROTO ve ENKAI hisselerinde Random Forest modelinin kapanış getirisi tahminlerinde daha düşük hata değerleri ürettiği belirlenmiştir. Buna karşın, XGBoost modelinin bazı durumlarda yüksek getiri seviyelerinde daha fazla sapma eğilimi gösterdiği ve bu durumun modelin uç değerleri yakalama kapasitesindeki farklılıktan kaynaklanabileceği değerlendirilmiştir.

**Tablo 3: Gedik Yatırım Beş Satırlık Hisse Karşılaştırma Örneği**

| Hisse | Open  | Gerçek High% | RF High % | XGB High % | Gerçek Low% | RF Low % | XGB Low % | Gerçek Close% | RF Close% | XGB Close% | RF_MSE |
|-------|-------|--------------|-----------|------------|-------------|----------|-----------|---------------|-----------|------------|--------|
| CEMTS | 5.25  | 2.69         | 1.82      | 1.43       | -0.91       | -1.48    | -1.32     | 0.09          | 0.30      | 0.95       | 0.47   |
| ECILC | 5.28  | 0.94         | 1.67      | 1.25       | -1.37       | 0.00     | 0.00      | -0.86         | 0.93      | 0.00       | 0.93   |
| AGHOL | 18.80 | 2.07         | 2.02      | 1.99       | -1.36       | -1.61    | -2.88     | -1.27         | 0.01      | -2.11      | 1.44   |
| DEVA  | 25.17 | 2.62         | 1.85      | 1.70       | -0.34       | -0.02    | -0.80     | 2.34          | 0.02      | 2.98       | 2.98   |
| EKGYO | 1.86  | 3.94         | 1.81      | 1.74       | -0.24       | -1.61    | -2.05     | 2.25          | 0.01      | 1.09       | 3.80   |

Tablo 3’te, dış veri seti olarak kullanılan Gedik Yatırım’a ait beş hisse senedi önerisinde ilişkin model tahminleri yer almaktadır. Ziraat Yatırım verileriyle eğitilen modellerin, Gedik Yatırım verileri üzerinde test edilmesi, önerilen metodolojinin farklı aracı kurumlara ait veri setlerinde de geçerliliğini ve genellenebilirliğini değerlendirmek açısından önemli bir aşamayı temsil etmektedir. Bu yaklaşım, modelin yalnızca eğitildiği örnekleme özgü kalmayıp, benzer yapısal özellikler taşıyan farklı veri kaynaklarında da tutarlı performans gösterip göstermediğini ortaya koymayı amaçlamaktadır.

**Tablo 4: Modellerin Performans Metrik Sonuçları**

| Model             | RMSE   | MAE    | MSE    |
|-------------------|--------|--------|--------|
| RF (Ziraat test)  | 1.9267 | 1.3253 | 3.7121 |
| XGB (Ziraat test) | 2.3319 | 1.5859 | 5.4378 |
| RF (Gedik test)   | 1.7958 | 1.2860 | 3.2251 |
| XGB (Gedik test)  | 2.0006 | 1.4123 | 4.0024 |

Tablo 4’te, iki modelin aynı test kümesi üzerindeki hata performansları karşılaştırıldığında, Random Forest algoritmasının tüm ölçütlerde XGBoost’a kıyasla daha düşük hata değerleri ürettiği görülmektedir. Random Forest modeli, ortalama RMSE açısından 1,92 değeriyle XGBoost’un 2,33’lük sonucuna kıyasla yaklaşık 0,41 puan daha iyi performans sergilemiştir. MAE ise 1,33’e karşı 1,59 MSE ise 3,71’e karşı 5,43 olarak gerçekleşmiş ve bu da Random Forest’in yaklaşık %31,7 oranında daha düşük hata ürettiğini ortaya koymuştur.

Hedef değişkenlerin yüzde getiriler olması sebebiyle bu farklar, modelin daha dengeli ve istikrarlı tahminler yaptığını göstermektedir. Ayrıca, farklı bir kurumdan elde edilen Gedik Yatırım verileri üzerinde yapılam testte de benzer eğilim sürmüştür; Random Forest'in hata değerleri RMSE 1,80, MAE 1,29 ve MSE 3,23 iken, XGBoost'un hata değerleri sırasıyla 2,00, 1,41 ve 4,00 olarak ölçülmüştür. Bu sonuçlar, önerilen metodolojinin farklı veri kaynaklarında da tutarlı ve güvenilir biçimde uygulanabilir olduğunu kanıtlamaktadır.

## SONUÇ

Bu çalışmanın temel amacı, Borsa İstanbul'da faaliyet gösteren aracı kurumların hisse senedi önerilerinin gerçekleşen piyasa verileriyle ne derece uyumlu olduğunu analiz etmek ve bu öngörülerin makine öğrenmesi yaklaşımlarıyla tutarlılığını değerlendirmektir. Bu doğrultuda, doğrudan fiyat seviyelerinin tahmini yerine açılış fiyatına göre yüksek, düşük ve kapanış fiyatlarındaki yüzdesel değişimler öngörülmüştür; böylece fiyatların enflasyon, hisse bölünmeleri ve uzun vadeli piyasa eğilimlerinden kaynaklanan etkileri en aza indirmişmiştir. Çalışmada kullanılan veri seti, Ziraat Yatırım'ın 2020-2021 yıllarına ait günlük görsel raporlarından OCR yöntemiyle elde edilen 592 gözlemden oluşmaktadır. Modelin genel geçerliliğini değerlendirmek amacıyla eğitim aşamasında Ziraat Yatırım verileri kullanılırken, Gedik Yatırım verileri dış seti olarak analiz edilmiştir. Eğitim sürecinde veri dengesizliğini azaltmak amacıyla SMOTE-ENN yöntemi uygulanmış, ardından T+1 gününe ilişkin tahminler Random Forest ve XGBoost algoritmalarıyla gerçekleştirilmiştir.

Analiz sonuçları, Random Forest algoritmasının hem Ziraat test setinde hem de Gedik dış setinde XGBoost modeline kıyasla daha düşük hata oranlarına ulaştığını göstermiştir. Ziraat test setinde Random Forest, yaklaşık %31,7 oranında daha düşük hata üretmiş; RMSE değeri 1,92, MAE değeri 1,32 ve MSE değeri 3,71 olarak hesaplanmıştır. Buna karşılık XGBoost modelinde RMSE 2,33, MAE 1,58 ve MSE 5,43 düzeyinde gerçekleşmiştir. Benzer biçimde, Gedik test setinde de Random Forest daha başarılı sonuçlar vermiş; hata metrikleri sırasıyla RMSE 1,79, MAE 1,28 ve MSE 3,22 olarak belirlenmiştir. XGBoost modelinin aynı veri setindeki değerleri ise RMSE 2,00, MAE 1,41 ve MSE 4,00'dır. Bu bulgular, Random Forest'in farklı kurumlara ait verilerde dahi daha istikrarlı ve düşük hatalı tahminler üretebildiğini ortaya koymaktadır.

Elde edilen sonuçlar ışığında, aracı kurum raporlarının yatırımcılar açısından daha etkin ve güvenilir hale gelebilmesi için önerilerin belirli standartlara bağlanması ve geçmiş performans verilerinin şeffaf biçimde paylaşılması gerekmektedir. Yatırımcıların ise analist tavsiyelerini tek başına değerlendirmek yerine, makine öğrenmesi tabanlı öngörü modelleri ve tamamlayıcı piyasa göstergeleriyle birlikte ele alması daha rasyonel bir yaklaşım olacaktır. Akademik perspektiften bakıldığında, gelecekte yapılacak araştırmalarda daha kapsamlı veri setlerinin kullanılması, farklı algoritmaların karşılaştırmalı olarak test edilmesi ve analist raporlarının metinsel içeriklerinin doğal dil işleme yöntemleriyle analiz sürecine dahil edilmesi önerilmektedir.

## KAYNAKÇA

- Alpaydın, E. (2014). *Introduction to Machine Learning* (3. Baskı). The MIT Press.
- Ayyıldız, N., & İskenderoğlu, Ö. (2024). How effective is machine learning in stock market predictions? *Heliyon*, 10(2), e24123. <https://doi.org/10.1016/j.heliyon.2024.e24123>
- Ayyıldız, N., & Iskenderoglu, O. (2023). Prediction of stock index movement directions using machine learning methods: An application on developing countries. *Journal of Financial Economics and Banking*, 4(2), 68-78.
- Bentéjac, C., Csörgő, A., & Martínez-Muñoz, G. (2021). A comparative analysis of gradient boosting algorithms. *Artificial Intelligence Review*, 54(3), 1937-1967. <https://doi.org/10.1007/s10462-020-09896-5>
- Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1), 5-32. <https://doi.org/10.1023/A:1010933404324>
- Chapelle, O., Schölkopf, B., & Zien, A. (Ed.). (2006). *Semi-Supervised Learning*. The MIT Press.
- Chen, T., & Guestrin, C. (2016). *XGBoost: A Scalable Tree Boosting System*. 785-794. <https://doi.org/10.1145/2939672.2939785>
- Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (2. Baskı). Springer.
- Koç, D. (2025). *Düzce Üniversitesi Kampüsünün Günlük Güneş Enerjisi ile Elektrik Enerjisi Üretim Değerinin Gün Öncesinden Derin Öğrenme Yöntemi ile Tahmin Edilmesi* [Yüksek Lisans Tezi]. Düzce Üniversitesi.
- Mitchell, T. (1997). *Machine Learning*. McGraw-Hill Science/Engineering/Math.
- Murphy, K. (2012). *Machine Learning: A Probabilistic Perspective*. The MIT Press.
- Ou, R., & Wang, Q. (2024). Are analysts' Forecasts Reliable? A Machine Learning-Based Analysis of the Target Price Accuracy. *Journal of Behavioral Finance*, 1-17. <https://doi.org/10.1080/15427560.2024.2368151>
- Rouf, N., Malik, M. B., Arif, T., Sharma, S., Singh, S., Aich, S., & Kim, H.-C. (2021). Stock Market Prediction Using Machine Learning Techniques: A Decade Survey on Methodologies, Recent Developments, and Future Directions. *Electronics*, 10(21). <https://doi.org/10.3390/electronics10212717>
- Saberironaghi, M., Ren, J., & Saberironaghi, A. (2025). Stock Market Prediction Using Machine Learning and Deep Learning Techniques: A Review. *AppliedMath*, 5(3). <https://doi.org/10.3390/appliedmath5030076>

- Shobha, G., & Rangaswamy, S. (2018). Chapter 8—Machine Learning. İçinde V. N. Gudivada & C. R. Rao (Ed.), *Computational Analysis and Understanding of Natural Languages: Principles, Methods and Applications* (C. 38, ss. 197-228). Elsevier. <https://doi.org/10.1016/bs.host.2018.07.004>
- Subasi, A., Amir, F., Bagedo, K., Shams, A., & Sarirete, A. (2021). Stock Market Prediction Using Machine Learning. *Procedia Computer Science*, 194, 173-179. <https://doi.org/10.1016/j.procs.2021.10.071>
- Tekin, S., & Çanakoğlu, E. (2018). Prediction of stock returns in Istanbul stock exchange using machine learning methods. *2018 26th Signal Processing and Communications Applications Conference (SIU)*, 1-4. <https://ieeexplore.ieee.org/abstract/document/8404607/>