
GAZİ ÜNİVERSİTESİ
Gazi
1926

DERİN ÖĞRENME ALGORİTMALARINA DAYALI TAHMİN: BİST100 VE USD/TRY ÜZERİNE BİR UYGULAMA

Tuğberk Koray YÜCEL

YÜKSEK LİSANS TEZİ
İSTATİSTİK ANA BİLİM DALI

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

NİSAN 2023

ETİK BEYAN

Gazi Üniversitesi Fen Bilimleri Enstitüsü Tez Yazım Kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- Tez içinde sunduğum verileri, bilgileri ve dokümanları akademik ve etik kurallar çerçevesinde elde ettiğimi,
- Tüm bilgi, belge, değerlendirme ve sonuçları bilimsel etik ve ahlak kurallarına uygun olarak sunduğumu,
- Tez çalışmasında yararlandığım eserlerin tümüne uygun atıfta bulunarak kaynak gösterdiğimi,
- Kullanılan verilerde herhangi bir değişiklik yapmadığımı,
- Bu tezde sunduğum çalışmanın özgün olduğunu,

bildirir, aksi bir durumda aleyhime doğabilecek tüm hak kayıplarını kabullendiğimi beyan ederim.

Tuğberk Koray YÜCEL

13/04/2023

DERİN ÖĞRENME ALGORİTMALARINA DAYALI TAHMİN: BİST100 VE USD/TRY ÜZERİNE BİR UYGULAMA

(Yüksek Lisans Tezi)

Tuğberk Koray YÜCEL

GAZİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

Nisan 2023

ÖZET

Günlük hayatta zaman serisi verileri oldukça karmaşık problemler içermektedir. Bundan dolayı zaman serisi veri analizi ve tahminini tutarlı şekilde yapmak önemlidir. Teknolojik gelişmeler sayesinde günümüz bilgisayarlarının performansları artmış ve bu artışa bağlı olarak birçok amaca hizmet eden algoritmalar geliştirilmiştir. Geliştirilen bu algoritmalar her alanda olduğu gibi zaman serisi analizlerinde de uygulanmakta ve karar vericinin farklı boyutlarını destekleyen roller üstlenmektedir. Özellikle finans alanında önceden öngörülmesi önemli olan borsa işlemlerinde bu tahmin algoritmalarına çok fazla başvurulmaktadır. Farklı istatistiksel ve matematiksel analiz, makine öğrenmesi ve derin öğrenme zaman serisi analizinde ve gelecek tahmini yapabilmek için kullanılmaktadır. Bu algoritmalar geleneksel yöntemlere göre daha üstün performans sergilemiş ve kullanımları yaygınlaşmıştır. Bu çalışmada Yahoo Finans üzerinden elde edilen Bist100 ve USD/TRY endekslerine ait 5 yıl (09.03.2017-09.03.2022) geriye dönük verileri kullanılarak derin öğrenme ve makine öğrenmesi yöntemleriyle kapanış fiyat tahminleri tahmin edilebilirliği ve bu yöntemlerin uygulanabilirliği araştırılmıştır. Çalışmada veri, eğitim verisi (%80) ve test verisi (%20) olmak üzere ikiye ayrılarak, Tekrarlayan Sinir Ağı (RNN) ve Uzun-Kısa Vadeli Bellek (LSTM) yöntemleriyle tahminleri yapılmıştır. Uygulanan yöntemlerden ulaşılan sonuçlar tahminlerin dönemin olumsuzluklarından etkilenerek bazı sapmalar gerçekleştirdiği ve uygulanan yöntemlerin kısa dönemli veriler ile veya teknik/temel analiz uygulanmış verilerle daha tutarlı sonuç verebileceği düşünülmektedir.

Bilim Kodu : 20515
Anahtar Kelimeler : Lstm, rnn, derin öğrenme, makine öğrenmesi, yapay sinir ağı, zaman serisi
Sayfa Adedi : 49
Danışman : Doç.Dr. Necla GÜNDÜZ TEKİN

PREDICTION BASED ON DEEP LEARNING ALGORITHMS: AN APPLICATION ON BIST100 AND USD/TRY

(M. Sc. Thesis)

Tuğberk Koray YÜCEL

GAZİ UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

April 2023

ABSTRACT

In daily life, time series data contains very complex problems. Therefore, it is important to analyze and forecast time series data consistently. Thanks to technological advances, the performance of today's computers has increased and algorithms that serve many purposes have been developed. These algorithms are applied in time series analysis as in every field and play roles that support different dimensions of the decision maker. Especially in the field of finance, these forecasting algorithms are widely used in stock market transactions, which are important to predict in advance. Different statistical and mathematical analysis, machine learning and deep learning are used in time series analysis and future prediction. These algorithms have shown superior performance compared to traditional methods and their use has become widespread. In this study, we investigate the predictability and applicability of closing price forecasts with deep learning and machine learning methods using 5-year (09.03.2017-09.03.2022) retrospective data of BIST100 and USD/TRY indices obtained from Yahoo Finance. In the study, the data was divided into two parts as training data (80%) and test data (20%) and predictions were made with Recurrent Neural Network (RNN) and Long-Short Term Memory (LSTM) methods. The results obtained from the applied methods show that the forecasts are affected by the adverse conditions of the period and some deviations are realized, and it is thought that the applied methods can give more consistent results with short-term data or data with technical/basic analysis.

Science Code : 20515
Key Words : Lstm, rnn, machine learning, deep learning, artificial neural network, time series
Page Number : 49
Supervisor : Asst. Prof. Necla GÜNDÜZ TEKİN

TEŞEKKÜR

Çalışmalarım boyunca destek ve emeklerini esirgemeyip her daim yol gösteren, bana devamlı rol model olacak, değerli bilgi ve tecrübeleriyle ufkumu açan kıymetli hocam Sayın Prof. Dr. Bülent ALTUNKAYNAK' a sonsuz teşekkürlerimi sunar, rahmetle anarım. Hayatımda bana kattıklarıyla her daim kalbimde yer alacak. Bana kaygısız, koşulsuz güvenen ve en zor sürecimde beni destekleyen Sayın Doç. Dr. Necla GÜNDÜZ TEKİN'e teşekkürlerimi borç bilirim. Her koşulda bana sonsuz destek olan sevgili aileme ve sevdiklerime çok teşekkür ederim.

İÇİNDEKİLER

Sayfa

Sayfa

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

SİMGELER VE KISALTMALAR

Bu çalışmada kullanılmış simgeler ve kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

Kısaltmalar	Açıklamalar
ARIMA	Autoregressive integrated moving average (Otoregresif hareketli ortalama)
CNN	Convolutional neural network (Konvolüsyonel sinir ağı)
ÇKA	Çok katmanlı algılayıcı
DT	Karar ağaçları
DVM	Destek vektör makineleri
EKK	En küçük kareler
LSTM	Uzun kısa vadeli bellek
MLP	Çok katmanlı algılayıcı
NSE	Hindistan Ulusal Menkul Kıymetler Borsası
NYSE	New York Menkul Kıymetler Borsası
RELU	Doğrultulmuş lineer birim
RNN	Tekrarlayan sinir ağı
SVM	Karar-destek makinası
YSA	Yapay sinir ağı

1. GİRİŞ

Günümüz teknolojisinin gelişmesiyle birlikte çok sayıda elektronik cihazdan veri akışı sağlanmaktadır. Sensörler, kameralar, sosyal medya, finans gibi çeşitli veri kaynakları, her yönüyle analiz edilemeyecek derecede fazla veri sağlamaktadır. Bunlar içerisinde zamana bağlı olarak değişim gösterenlere, zaman serisi verileri denmektedir. Meteoroloji, finans, kontrol, video işleme gibi alanlarda zaman serisi analizlerinin önemi büyüktür ve zaman serilerinden anlamlı bilgiler çıkarmak için çok sayıda çalışmalar yapılmaktadır (Wei, 2006).

Günümüzde finans alanında önemli gayelerden biri hisse senetlerinin fiyatlarını tahmin etmek ve uygun yatırımlar yaparak para kazanmaktır. Dolayısı ile hisse senetleri ve fonlar, zaman serisi analizi ve tahmin yöntemleri kullanılarak uzun süredir yatırımcılar ve araştırmacılar için önemli bir konu haline gelmiştir.

Son yirmi yılda, zaman serisi analizlerinde, geleneksel yöntemlerin yerini, devamlı olarak gelişen MLP, SVM ve DT gibi makine öğrenmesi ve derin öğrenme modellerinin aldığını ve yaygın olarak kullanılmaya başlandığını söyleyebiliriz. Özellikle, her konuda olduğu gibi finans konusunda da oluşan büyük veriler ve bu verilerin işlenmesinde kullanılan paket programlardaki gelişmeler derin öğrenme modelleri ile analizlerin daha kolayca yapılabilir olmasını sağlamıştır. Bu nedenle, finansal verilerin tahminleri üzerine yeni çözümler arayan araştırmacılar, derin öğrenme modellerine ağırlık vererek, birçok çalışma yapmışlardır (Lachiheb ve Gouider, 2018).

Gelişen teknoloji ile birlikte öğrenmeye dayalı algoritmalar bu sürece değerli katkılar sağlamakla beraber özellikle son yıllarda makine öğrenmesi ve derin öğrenme uygulamalarının finansal alandaki tahminlerinde kullanımı ve elde edilen olumlu sonuçlar, araştırma yapanların bu alana olan ilgi ve alakasını arttırmış ancak piyasalardaki fiyat davranışlarını net tahmin edebilen bir metot bulunamadığından, bu alandaki çalışmalar sürekli olarak devam etmekte ve gelişmektedir.

Makine öğrenmesinin bir alt dalı olan Konvolüsyonel Sinir Ağlarının (CNN) çeşitli dönüşümler uygulanarak görüntü tanıma, görüntü ayıklama gibi problemlerde oldukça üstün özelliklere sahip olmasına rağmen, zaman serisi analizleri ile çok iyi çalışmadığı düşünülmektedir (Sezer ve Ozbayoglu, 2018). Öte yandan, zaman serisi problemlerinde

kullanılan Tamamıyla Bağlı Sinir Ağları (FCNN) her ne kadar iyi çalışılıyor gibi görünse de, Tekrarlayan Sinir Ağları (RNN) ve Uzun-Kısa Vadeli Bellek (LSTM) yöntemleri bu tip verilerle daha uyumlu çalışılmaktadırlar (Navon ve Keller, 2017).

Bu tez çalışmasında, zaman serileri analizlerinde kullanılan Yapay Sinir Ağı (YSA) yapılarının alt uygulaması olan Tekrarlayan Sinir Ağı (RNN) ve Uzun Kısa Vadeli Bellek (LSTM) algoritmalarının performansları finansal değerler olan BİST100 ve USD/TRY verileri için eğitim ve test verileri üzerinden değerlendirilmektedir. Algoritmaları uygulamak için Yahoo Finans üzerinden alınan veriler ön işlemeden geçirilerek analiz yapılmıştır.

Tez çalışmasının ikinci bölümünde RNN ve LSTM yöntemleri ile ilgili literatür çalışmalarına yer verilmektedir. Üçüncü bölümde ise, zaman serisi analizinde kullanılan yöntemler (ADF ve EG testi) ve verinin elde edilmesi ve ön işleme aşaması anlatılmaktadır.

Makine öğrenmesi, derin öğrenme yöntemleri, Tekrarlayan Sinir Ağı (RNN) ve Uzun-Kısa Vadeli Bellek (LSTM) algoritmaları dördüncü bölümde yer almaktadır. Beşinci bölümde, Augmented Dickey Fuller (ADF) testi, Engle-Granger Eşbütünleşme testi, Tekrarlayan Sinir Ağı (RNN) ve Uzun-Kısa Vadeli Bellek (LSTM) algoritmalarının uygulaması ve altıncı bölümde de sonuçlar özetlenmektedir.

2. LİTERATÜR TARAMASI

Yapay zekâ ve yapay zekânın alt dalları ile ilgilenen araştırmacılar her zaman daha üstün sistemlerin tasarlanmasını hedeflemektedirler. İnsan düşünce yapısını ve karar verme kabiliyetini modellemek, kuşkusuz hedeflerin en önemlisidir. Bu amaçla ilk defa McCulloch ve Pitts tarafından insanda bulunan sinir sisteminden esinlenerek beyin fonksiyonlarının işleyişini mantıksal olarak hesaplayan bir model ortaya konmuştur (McCulloch ve Pitts, 1943).

1943'den 1980 yıllarına kadar sinir ağları konusunda gelişmeler olmasına rağmen, 1980'li yıllardan sonra özellikle görüntü işleme konusunda bilgisayar teknolojinde ve programlamaya ilişkin gelişmelerle birlikte sinir ağları ile ilgili çalışmaların daha çok arttığı görülmektedir. 1980'li yıllarda yapılan çalışmalar yeniden hız kazanmış ve önemli katkılar sağlanmıştır (Rumelhart ve McClelland, 1995). Bu dönemde, yapay sinir ağlarını eğitmek için geri yayılım algoritmasının başarıyla kullanıldığı ve yaygınlaştığı görülmektedir (Rumelhart ve Hinton, 1986; LeCun ve diğerleri, 1987).

Daha sonraki yıllarda, Geoffrey Hinton, ve Hinton ve Osindero çok katmanlı ön eğitim yöntemi ile derin öğrenme algoritmalarının etkili bir şekilde eğitilebileceği üzerinde çalışmışlardır (Hinton ve Osindero, 2006). İzleyen yıllarda, derin öğrenmenin temeli olan geri yayılım algoritmaları ile ilgili çalışmalar LeCun ve diğerleri tarafından yapılmıştır (LeCun ve diğerleri 2015).

Facebook ve Google gibi büyük şirketlerinde kullanmış olduğu görüntülerin etiketlenmesi, obje tanımlanması ve görüntülerin sınıflandırılması (Kiros ve Salakhutdinov, 2014); siyah-beyaz fotoğrafların ya da görüntülerin renklendirilmesi (Hwang ve Zhou, 2015) geçmişte renksiz olarak çekilen filmlerin renklendirilmesi bir resimdeki insanlara ait bakışın değiştirilmesi (Ganin ve diğerleri, 2016); insanların anlık hareketi ve duruşunun tahmini ve insan iskelet yapısındaki hareketlerin iki boyutlu olarak tahmini (Cao ve Simon, 2016); bir resim taslağında resmin elde edilmesi, bir çizimden nesnenin şeklinin elde edilmesi veya harita taslağından gerçek haritanın elde edilmesi (Isola ve diğerleri, 2016) derin öğrenme algoritmaları ve modelleri le yapılmıştır.

Görüntü işlemenin yanı sıra, derin öğrenme zaman serileri içinde kullanılmaya başlanmıştır. Faydalı örüntüler bulma sürecinde, doğrusal ve doğrusal olmayan modeller, çok katmanlı sinir ağları, k-ortalama ve hiyerarşik kümeleme, k-en yakın komşu, karar ağacı analizi, regresyon (lojistik regresyon, genel çoklu regresyon), otoregresif hareketli ortalama modelleri (ARIMA), temel bileşen analizi ve Bayes teoremi, ilişkisel yöntemler, olasılık yöntemleri, destek vektör makinesi, gizli Markov modelleri gibi birçok veri analizi yöntemi ve tekniği finansal modellemede kullanılmıştır (Maimonand ve Rokach, 2005).

Hiransha ve diğerleri (2018) bir şirketin hisse senedi fiyatını mevcut tarihsel fiyatlara göre tahmin etmek için dört tür derin öğrenme mimarisi kullanmışlardır (Hiransha ve diğerleri, 2018). Bunlar; Çok Katmanlı Algılayıcı (MLP), Tekrarlayan Sinir Ağı (RNN), Uzun-Kısa Vadeli Bellek (LSTM), ve Konvolüsyonel Sinir Ağı (CNN) gibi yöntemleridir. Bu çalışmalarda, Hindistan'ın Ulusal Menkul Kıymetler Borsası (NSE) ve New York Menkul Kıymetler Borsası (NYSE) olarak iki farklı borsada günlük kapanış fiyatı kullanılmıştır. Sistem, NSE'den tek bir şirketin hisse senedi fiyatı ile eğitilmiştir ve hem NSE hem de NYSE'den beş farklı şirket için test edilmiştir. CNN'nin diğer modellerden daha iyi performans gösterdiği görülmüştür.

Fischer ve Krauss (2018) yaptıkları çalışmada S&P 500'in kurucu hisse senetleri için 1992'den 2015'e kadar örneklem dışı yön hareketlerini tahmin etmek için LSTM ağları kullanmışlardır. Genel pazara göre olan performans 1992'den 2009'a kadar çok açık, ancak 2010'dan itibaren, işlem maliyetlerinden sonra sıfıra yaklaşan LSTM kârlılığı ile fazla getiri tahkim edildiği gözlenmiştir. Bu bulgulardan yararlanarak, işlem maliyetlerinden önce %0.23 veren kurallara dayalı kısa vadeli bir geri dönüş stratejisi geliştirmişlerdir (Fischer ve Krauss, 2018).

Cheng ve diğerleri (2018) hisse senedi fiyat hareketini öngörmek ve alım satım stratejileri yapmak için bir LSTM modeli önermişlerdir. Çalışma iki konuya odaklanmıştır. LSTM modelini kullanarak gelecekteki hisse senedi fiyatı hareketini tahmin etmek için tarihsel hisse senedi verileri ve teknik göstergeleri kullanıldığı çalışmadaki sonuçlar geri kazanımlara göre hesaplanmıştır. Geçmiş fiyat verileri ve teknik göstergeler ile modelin eğitimi yapılmış daha sonra tahmin sonuçları bir ticaret stratejisine karar vermek için kullanılmıştır (Cheng ve diğerleri, 2018).

Chen ve diğerleri (2020) tarafından yapılan çalışmada, Bitcoin'in gelecek değer tahmini için lojistik regresyon, doğrusal ayırma analizi (LDA), rastgele orman, XGBoost (XGB), ikinci dereceden ayrım analizi, destek vektör makinesi (SVM) ve LSTM yöntemlerini karşılaştırmıştır (Chen ve diğerleri, 2020).

Zaman serisi problemlerinin analizlerinde, son yıllarda araştırmacılar tarafından tercih edilen LSTM yöntemi kompleks ve lineer olmayan verilerin tahmin tutarlılığında ciddi iyileştirmeler sağlamıştır. Sagheer ve Kotb yaptıkları çalışmada petrol fiyatlarının tahmini için LSTM algoritmasını kullanmışlardır (Sagheer ve Kotb, 2019). Bouktif ve diğerleri (2018) ise LSTM parametre optimizasyonu için genetik algoritma yaklaşımı sunmuştur (Boukif ve diğerleri, 2018). Bunların yanında finans piyasalarında hisse senetlerinin fiyat tahminleri için LSTM algoritmasının performansı incelenmiştir (Gunduz ve diğerleri, 2017), (Fischer ve Kraus, 2018).

Olvera-Juarez ve Huerta-Manzanilla (2019) BTC fiyat tahminlemesinde kullanılan bazı karma yöntemleri incelemişlerdir. Bu araştırmacılar, ARIMA, LSTM ve RNN algoritmalarının BTC fiyatlarını tahminlemeye yönelik performanslarını ve ARIMA algoritmasının durağan olmayan dönemlerde kullanıldığında, tatmin edici tahminlemelerin görülmediğini belirtilmiştlerdir. Ayrıca, LSTM ve RNN algoritmaları ile BTC fiyatlarını tahminlemek için uygun yapılar kurulabileceğini vurgulamışlardır (Olvaro-Juarez ve Huerta-Manzanilla, 2019).

Almeida, Tata, Moser ve Smit (2015) önceki günlerin kapanış fiyatlarını ve hacimlerini baz alarak sonraki gün için BTC fiyatının eğilimini tahminlemişlerdir. Çalışmalarında, tahminleme yapmak üzere yapay sinir ağları (YSA) algoritmalarını kullanmışlardır. Hacimlerin hesaba katıldığı ölçümlerin sonucu, sadece kapanış fiyatlarının yer aldığı ölçümlerin sonucundan daha düşük çıkmıştır. Bu çalışmaya ek olarak, basit yapay sinir ağ yapısı ve karmaşık yapay sinir ağ yapısını karşılaştırmışlardır. Basit ağ yapılarının tahminlemelerde daha yüksek performansa sahip olduğu görülmüştür. Farklı nöron sayıları ve geciktirme metrikleri kullanan araştırmacılar, nöron sayısı düşük olsa bile basit ağ yapılarındaki performansın pozitif bir ivme kaydettiğini vurgulamışlardır (Almeida ve diğerleri, 2015).

Borsa verilerinden asıl verimi elde etmek için kısa süreli tahmin yapmak özellikle dinamik kullanıcılar için önem arz etmektedir. Özellikle bir gün içerisindeki “Al-Sat” komutlarıyla kazanç sağlamak isteyen yatırımcılar farklı makine öğrenmesi algoritmaları kullanarak bu amaçlarına ulaşmaya çalışmışlardır (Kumar ve Ningombam, 2018; Shao ve diğerleri, 2017).

3. ZAMAN SERİSİ VERİSİNİN ÖN İŞLENMESİ VE KULLANILAN YÖNTEMLER

Zaman serisi şeklindeki veriler, her alanda karşımıza çıkmaktadır. Finans verisi, sensör verileri, titreşim verileri, hava durumu verileri gibi farklı problem verileri zamana bağlı olarak değişim göstermektedirler. Zamana bağlı değişim gösteren verilere, zaman serisi verileri denilmektedir.

Zaman serisi verileri farklı biçimlerde görülmektedir: tamamen rassal düzende, zamanla üstel bir şekilde artabilen, zamanla lineer bir şekilde artan düzende, mevsimsellik barındıran, lineer ve mevsimsellik karışımı olan düzende olabilirler.

Bir zaman serisi verisi kendisini oluşturan eğim, mevsimsellik, periyodiklik ve artık olarak 4 parçada ifade edilebilir. Bu bileşenler birleşerek zaman serisini meydana getirirler. Eğim bileşeni, uzun vadedeki yükseliş ve alçalış eğimini ifade eder. Mevsimsellik bileşeni, belirli zamanda sürekli tekrar eden kısmı ifade eder. Periyodiklik bileşeni, mevsimsellik bileşenine göre daha uzun periyodik kısımla ilgilidir. Artık bileşen ise diğer bileşenler çıkarıldığında geriye kalan artık kısmı ifade eder.

3.1. Verinin Tanımlanması

Bu çalışmada, kullanılan finans verileri, USD/TRY ve Bist100 kapanış değerleri olarak Yahoo Finance veri tabanından alınmıştır (https:// finance. yahoo. com/ quote/ XU100.IS?p =XU100.IS&.tsrc = fin-srch, https://finance.yahoo.com/quote/TRY=X?p=TRY=X&.tsrc=fin-srch). Zaman periyodu olarak 09.03.2017-09.03.2022 arasındaki tarihler kullanılmıştır. Ön işlenme sürecine geçmeden önce, bu tarihler arasında var olan 1285 adet gözlem içinde borsanın kapalı olduğu günlerin değerleri de veriden atıldıktan sonra 1255 adet gözlem üzerinden analiz yapılmıştır. Böylece, analizler açısından daha istikrarlı bir eğitim modeli oluşmasını sağlanmaktadır.

3.2. Verinin Ön İşlenmesi

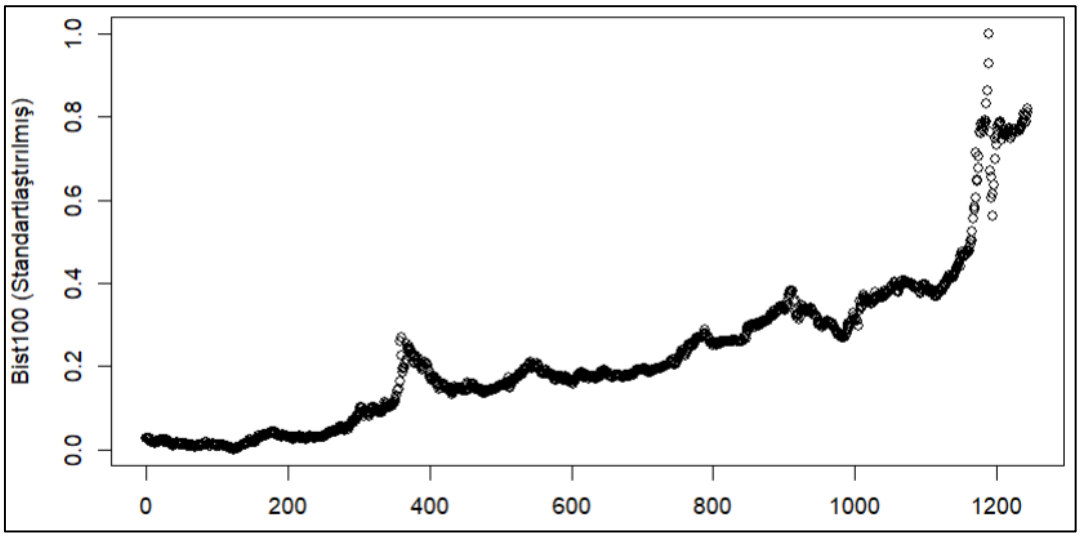
Literatürde kullanılan makine öğrenmesi ya da derin öğrenme modelleri ne kadar çok sayıda problemde insan seviyesini geçmiş olsalar da ön işlemden geçmemiş olan veriler için

çözülmeye çalışan optimizasyon probleminde sıkıntıların ortaya çıkmasına sebep olabilirler (Pal ve Sudeep, 2016). Veri setinin bu bağlamda uyumlu olması modellerin daha iyi bir öğrenme süreci geçirmesini sağlar.

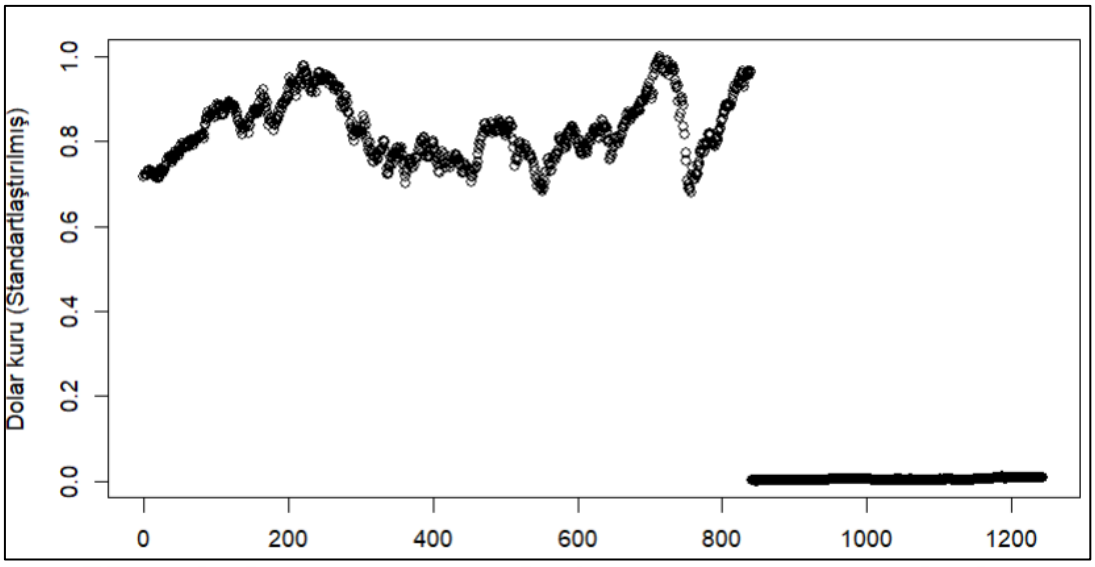
Öznitelik ölçeklendirme, en basit ön işleme yöntemlerinden biridir. Ölçeklendirme sayesinde, öznitelik uzayındaki her bir boyut 0-1 arasına sıkıştırılır. Bu standartlaştırma işlemi, her bir boyuttan minimum değerin çıkarıp, her bir boyuttaki maksimum değer ile minimum değer arasındaki farka bölünmesi ile yapılır (Eşitlik 2.1). Öznitelik ölçeklendirmesiyle yapılan ön işleme, eğer veride uç değerler var ise, uç değerler dışındaki değerleri fazlasıyla küçülteceği için probleme sebep olur. Kullanılan veride uç değer olmaması böyle bir probleme sebep olmamıştır.

Analizde kullanılan ölçeklendirilmiş olan BİST100 indeks değerleri ve USD/TRY verilerinin zaman boyutuna (5 yıllık günlük) göre grafikleri sırasıyla Şekil 3.1 ve Şekil 3.2'de verilmektedir.

$$\frac{X - X_{min}}{X_{max} - X_{min}} \tag{2.1}$$



Şekil 3.1. Standartlaştırılmış (0-1 aralığında) Bist100 verisi.



Şekil 3.2. Standartlaştırılmış (0-1 aralığında) USD/TRY verisi

3.3. Verinin Durağanlığı

Durağanlık, zaman serisinin ortalama, varyans gibi istatistiksel özelliklerinin zamana bağlı olarak değişmemesi durumudur. Verinin durağanlık durumunu öğrenmek için kullanılan test yöntemi Augmented Dickey Fuller (ADF), test edilen zaman serisi verisinin durağanlık durumunu ortaya koyar ve gecikme değerlerini gösterir. Kullanılan tahmin modelleri, bu istatistiksel özelliklere göre kendilerini adapte edebilecekleri için, durağan olamayan zaman serileri için çeşitli dönüşümler yapılması zorunludur. Bu dönüşümlerden Engle-Granger Eşbütünleşme testi veride bulunan değişkenlerin bir arada anlamlı bir sonuç verip vermeyeceğini ifade etmektedir.

3.4. Augmented Dickey Fuller (ADF) Testi

Bir serideki birim kökün varlığını araştırmak için kullanılan en popüler test Dickey ve Fuller tarafından literatüre kazandırılmıştır. Dickey ve Fuller'ın 1979 ve 1981 yıllarında yapmış oldukları çalışmalar sonucunda zaman serilerinde durağanlığı test etmek için birim kök testleri üzerinde yoğunlaşmışlardır (Dickey ve Fuller, 1979; Dickey ve Fuller, 1981).

Bir serinin uzun zaman boyutunda sahip olduğu özellik, bir önceki dönemde değişkenin aldığı değeri, gelecek dönemi nasıl etkilediğinin tespit edilmesiyle bulunmaktadır. Bu nedenden dolayı, serinin daha önceki dönemde nasıl bir sürece uğradığını görmek için, her

dönemde almış olduğu değeri geçmiş dönemlerdeki değerleriyle arasındaki ilişkinin tespiti için regresyonunun kurulması gerekmektedir. Bu sürecin tespiti için farklı metotlar geliştirilmiş, ekonometride birim kök analizi olarak adlandırılan yöntem ile, serinin durağan olup olmadığı tespit edilmektedir (Tarı, 2015).

Dickey Fuller testi zaman serisi değişkenlerinin otoregresif süreçle ifade edilip edilemeyeceğini göstermekte olup, test sonucunda hata teriminde otokorelasyon bulunması halinde zaman serileri birinci dereceden otoregressif süreçle ifade edilememektedir. Birinci dereceden otoregresif Eşitlik 3.1'deki gibidir:

$$Y_t = PY_{t-1} + u_t \tag{3.1}$$

Eşitlik 3.1'deki, u_t öngörülemeyen hata terimidir. *P* katsayısı 1'e eşitse, *t* zamanı göstermek üzere*, Yt* birim köklü zaman serisi aynı zamanda rassal yürüyüşlü zaman serisi olmaktadır. Rassal yürüyüş serisi birim köklü zaman serisi örneğidir. Model $P = 1$ eşit olduğu kabul edilirse Eşitlik 3.2'deki gibi ifade edilebilir (Gujarati, 2011):

$$Y_t = Y_{t-1} + u_t \tag{3.2}$$

Eşitlik 3.2 ile verilen modelin, daha önceki dönemlerde modeli oluşturan değişken değerinin etkisinde kaldığı söylenebilir.

Eşitlik 3.2'nin her iki tarafından $Y_t - 1$ çıkarılarak Eşitlik (3.3) ilişkisi elde edilir:

$$\Delta Y_t = (P-1)\, Y_{t-1} + u_t \tag{3.3}$$

Burada Eşitlik 3.3'ün sol tarfı $\Delta Y_t = Y_t - Y_t - 1$ birinci farktır.

Modelin birim köklü olması, orijinal bir serinin birinci farkı durağan ise orijinal seriye birinci dereceden entegre olmuş denir ve *I(1*) olarak adlandırılır. Eğer seri durağan değilse durağan olması için iki kere fark almak gerekirse *I(2)* ve d'nci defa fark almak gerekirse *I(d)* olarak yazılır. Böylece, durağan olmayan bir seri durağan olması için farkları alınarak,

durağan hale getirilebilmektedir. Bu süreç serinin durağanlaşmasını dolayısıyla serinin durağan hale gelmesini sağlamaktadır (Gujarati, 2012).

Bir serinin durağan olup olmadığını test etmek için Eşitlik 3.1 kullanılır.

H_0: $P = 1$ Ise, Y_t zaman serisi durağan değildir. Başka bir deyişle, zamana bağlı bir yapıya sahiptir ve zaman içinde sabit bir varyansa sahip değildir.

H_1: $P < 1$ Y_t ise zaman serisi durağandır ve birim kök yoktur.

3.5. Engle-Granger Eşbütünleşme Testi

Engle-Granger eşbütünleşme analizi değişkenler arasındaki ilişkinin olup olmadığını araştırmaktadır. Bu kapsamda, eşbütünleşik bir ilişkinin olabilmesi için bu ilişkinin uzun süreli ve güçlü olması gerekmektedir. Bu yöntemin ön koşulu ise analizde kullanılacak değişkenlerin düzey değerlerinde durağan olmamasıdır. Diğer bir ifadeyle, söz konusu değişkenlerin düzey değerlerinde durağan olmamaları ve bu durağanlığın her bir değişken için aynı seviyede gerçekleşmesi gerekmektedir. Bu bağlamda, seriler ilk olarak birim kök testine tabi tutulurlar (Ersin ve Eti, 2017).

Durağanlık ön koşulunu sağlayan değişkenler ile ikinci seviyeye geçilebilmektedir. Bu aşamada, söz konusu değişkenler regresyon analizine tabi tutulmaktadırlar. Daha sonra, elde edilen regresyon analizinin hata terimlerine ait seriler oluşturulmaktadır. Belirtilen hususun ardından hata terimlerine ait bu seriler tekrar birim kök testine tabi tutulmaktadır. Elde edilen analiz sonuçlarına göre, hata terimlerine ait serilerin düzeyde durağan olması durumunda, değişkenler arasında eşbütünleşme ilişkisi olduğu, aksi takdirde ise böyle bir ilişkinin bulunmadığı anlaşılmaktadır (Engle ve Granger, 1987).

İlk aşamada, incelenen serilerin eşbütünleşik olan parametrenin ya da parametre vektörünün değeri tahmin edilir. Daha sonra elde edilen bu tahmin Hata Düzeltme Modelinde (Error Correction Model) kullanılır. Her iki aşamada da en küçük kareler yöntemi kullanılarak yapılan tahminler tutarlıdır. Yöntemin birinci aşamasında,

$$Y_t = \beta_0 + \beta_1 X_t + u_t \tag{3.4}$$

regresyon modeli (eşbütünleşme regresyonu) En Küçük Kareler (EKK) ile tahmin edilir. Bu eşitlik ile ilgili tahmin edilen hata terimleri,

$$\hat{u}_t = Y_t - \hat{\beta}_0 - \hat{\beta}_1 X_t \tag{3.5}$$

olarak ifade edilir. Daha sonra bu hata terimlerinin durağan olup olmadığını belirlemek için,

$$\Delta \hat{u}_t = \hat{\beta}_1 \hat{u}_t + \varepsilon_t \tag{3.6}$$

eşitliği tahmin edilir.

Burada,

$H_0: \beta_1 = 0$ (u_t durağan değildir veya X ve Y değişkenleri arasında eşbütünleşme ilişkisi yoktur) hipotezi test edilir.

$H_0: \beta_1 \neq 0$ (u_t durağandır veya X ve Y değişkenleri arasında eşbütünleşme ilişkisi vardır) hipotezi test edilir.

Yapılan test sonucunda sıfır hipotezi red edilemezse u_t hata teriminin durağan olmadığına, dolayısıyla X_t ve Y_t serileri arasında eşbütünleşme ilişkisinin olmadığına karar verilir. Aksi halde X_t ve Y_t serileri arasında eşbütünleşme ilişkisinin olduğuna karar verilir.

4. KULLANILAN METOT

Makine öğrenmesi ve derin öğrenme konularına başlamadan önce, öğrenme ile ne söylenmek istendiğini kısaca ifade edelim. Öğrenme bir süreçtir ve süreç içerisinde kişinin davranışlarında değişiklik olması beklenir. Gerçekleşen bu değişiklikler olumlu olabileceği gibi olumsuz da gerçekleşebilir. Süreç içerisinde birey edindiği tecrübeler ışığında mevcut davranışlarını değiştirebileceği gibi yeni davranışlarda edinebilir. Öğrenme kelimesinin sözcük anlamı; öğrenim, talimat ve deneyim yoluyla beceri, anlama ve bilgi kazanımı olarak tanımlanmaktadır (Nilson,1996).

4.1. Makine Öğrenmesi

Makine öğrenmesi, sistemin geçmişteki deneyimlerinden elde edilen öğrenmelerini kullanarak bir model oluşturulmasına ve gelecekte karşılaşacağı durumlar karşısında bir tahminde bulunmasını sağlayan bir yapay zekâ alanıdır. Bilgisayarın eldeki veriler üzerinden geliştirilen bir model yardımıyla oluşturuldukları öğrenmeleri, ileride karşılaştıkları yeni veriler üzerinde kullanarak kararlar verebilme ve ilgili problemlere çözümler üretme olarak tanımlanabilir (Öğütücü, 2006). Bilgisayar oyunları ve yapay zekâ alanında öncü kişilerden sayılan Arthur Lee Samuel, 1959 yılında makine öğrenmesini bilgisayarların yeniden programlamaya ihtiyaç duymadan görev yapmasını sağlayan bilim olarak tanımlamıştır (Samuel, 1959). Michalski ve diğerleri (1986) ise daha resmi bir tanımlama yapmıştır: Bir bilgisayar yazılımının, deneyim D'yi kullanarak istenilen bir görev G'yi gerçekleştirmek için performans P ölçüsü ile eğitilmesidir. Burada performans P ölçütü ile oluşturulan görev G, deneyim D'yi geliştirmektir (Michalski ve diğerleri, 1986).

Makine öğrenmesi, bilgisayarların öğrenmelerine yönelik yöntemleri içeren veri madenciliğiyle oldukça ilgili geniş bir bilgisayar alanıdır. Makine öğrenmesi ile bilgisayar yazılımlarının, önceden yaşanılan durumlardan oluşturdukları model yardımıyla yeni durumları öğrenmeleri sağlanır. Eğitilmek istenen yazılım, öncelikle bir örnek üzerinden bazı bilgiler çıkararak bir model oluşturur. Ardından gelen diğer örnekler yardımıyla model üzerindeki iyileştirmeler gerçekleştirir. Gerçekleştirilen bu işleme tecrübelerden öğrenmede denilebilir. Makine öğrenmesi ve veri madenciliği arasında çok ince bir çizgi bulunmaktadır. Makine öğrenmesi algoritmalarının birçok uygulama alanına vardır, eğer bu algoritmaları büyük veri tabanlarına uygulanırsa buna veri madenciliği denilmektedir (Alpaydın, 2004).

Makine öğrenmesi, veri madenciliğinin uygulama kısmında yer alır ve seçilen Makine öğrenmesi algoritması ile, ilgili veri seti üzerinde çalıştırılır.

Makine öğrenmesi bir yapay zekâ alanıdır ve sadece veriler üzerinden çalıştırılmaz. Makine öğrenmesinin amacı elimizdeki veri içerisindeki değerli bilgiyi ortaya çıkarmak için algoritmaların geliştirilmesini ve bu algoritmaların iyileştirilmesi sağlamaktır. Veri madenciliğinin amacı ise elimizdeki verilerin işlenerek çıkarılan bilgilerin yorumlanmasını sağlamaktır. "İki konu arasındaki en büyük fark; makine öğrenmesi, öğrenme metotlarını geliştirerek, tahminleri ya da tanımları en iyi şekilde, yüksek performans ile nasıl çıkarabileceği ile ilgilenirken, veri madenciliğinin ortaya çıkan bilgi ile ilgilenmesidir" (Dalyan, 2006)

Bir makine öğrenmesi görevinin gerçekleştirmesi için beş temel adım şu şekilde ifade edilebilir:

Veri toplama

Çeşitli ortamlardan (veri tabanları, metin dosyaları ve web sayfaları vb.) ham verilerin toplanmasıdır. Geçmiş verilerin toplanması gelecek öğrenmelerin temelini oluşturur. İlgili verilerin çeşitliliği, yoğunluğu ve hacmi ne kadar iyi olursa, makine öğrenmesi için o derece iyi olacaktır.

Verilerin hazırlanması

Verilerin eğitim modelini oluşturmak için sisteme verilmeden önce yapılan önişlemleri kapsar. Örneğin bir doğal dil işleme için durma sözcüklerinin veriden temizlenmesi, eğer gerekli değilse noktalama işaretlerinin veriden ayıklanması vb. işlemleri kapsar.

Modeli eğitmek

Modeli oluşturmak için uygun algoritmanın seçilmesini ve verilerin temsili için uygun bir model oluşturma sürecini kapsar. Eğer danışmanlı eğitim yapılacaksa veriler bu aşama da eğitim ve test kümelerine ayrılır. Eğitim kümesi ile sistem çalıştırılarak test aşamasında kullanılmak üzere verilere uygun bir model oluşturulur.

Modelin değerlendirilmesi

Sistemin doğruluğunu test etmek için bir önceki aşamada geliştirilen model test kümesine uygulanır ve sonuçlar kaydedilir. Modelin doğruluğunu en iyi şekilde test etmek, modelin daha önce hiç görmediği veriler üzerinden başarısını ölçmektir.

Performans artırmak

Bu adım, tamamen farklı bir model seçmeyi veya verimliliği artırmak için daha fazla değişken sunmayı gerektirebilir. Bu nedenle, önemli miktarda veri toplanması ve model oluşturmak için ön-işlemden geçmesi gerekebilir.

Bilgisayarlar, makine öğrenmesi algoritmaları ile toplumun işgücüne katkı sağladıkları gibi toplumun beyin gücüne de katkılar sağlamaya başlamıştır. Çeşitli alanlarda ve çok miktardaki verinin işlenmesi, analiz edilmesi ve analiz sonuçlarının yorumlanması makine öğrenmesi algoritmaları ile kolay hale gelmiştir. Makine öğrenmesi algoritmaları, tümevarım yöntemini kullanmaktadır. Tümevarım yönteminin amacı parçaları kullanarak bütüne ulaşmaya çalışmaktadır. Parçalardan bütüne ulaşmak için oluşturulan çıkarımlar, bir sonraki adımda tahminde bulunmak için veya bir tanımlama yapmak için kullanılacaktır. Elimizdeki veri, geleceğe yönelik bir bilgiyi tahmin etmek için kullanılacaksa buna tahmin edici model denilmektedir. Tahmin edici modeller eğitim ve test aşaması olmak üzere iki bölümden oluşmaktadır. Eğitim safhasında, elimizdeki eğitim verisi denilen ve etiketli olan veriler kullanılarak bir model oluşturulur. Eğitim verisinin bir kısmı test aşamasında kullanılmak üzere ayrılır. Eğitim verisi kullanılarak sistemin bir model oluşturması sağlanır. Modelin oluşturulmasının ardından, eğitim verisi içinden test için ayrılan veri kullanılarak sistem test edilir.

Makine öğrenmesinin amaçları üç açıdan incelenebilir (Carbonell ve diğerleri, 1989):

1. Hedef Tabanlı Çalışmalar: Öğrenme sisteminin gelişimi ve analizi belirlenmiş görevlerin yerine getirmek için gerçekleştirilir. Bu yaklaşım, makine öğrenmesine mühendis yaklaşımı olarak tanımlanmıştır.

2. Bilişsel Simülasyon: İnsanın öğrenme sürecini araştırıp bilgisayar ortamında benzetimini gerçekleştirmek olarak tanımlanmıştır. Bu ise, makine öğrenmesine bilişsel modelleme yaklaşımıdır.
3. Teorik Analiz: Uygulama alanlarından bağımsız olarak teorik olabilecek öğrenme metotları ve algoritmaları incelemek içindir.

Makine öğrenmesiyle ilgilenen bilim insanları önerilen üç yaklaşımdan yalnızca birisini veya hibrit bir şekilde kullanarak çalışmalar gerçekleştirmiştir.

Carbonell makine öğrenmesinin önemini şu şekilde sıralamıştır (Carbonell ve diğerleri, 1989).

Bazı görevler için, girdi/çıktıyı belirlesek de arasındaki ilişki belirtilemeyebilir. Bu gibi durumda, makinenin kendi içyapısını ayarlayarak, büyük veri yığınlarından giriş/çıkış fonksiyonunu bulup, arasındaki ilişkiyi tahmin etmesi beklenir.

Büyük veri yığını içinde gizli kalmış önemli ilişkiler ve bağlantılar olabilir. Makine öğrenmesi metotları, bu ilişkileri seçip çıkarmak için kullanır. Bu konu, veri madenciliği olarak adlandırılmaktadır.

Bazı görevler için bilgi miktarı, insanın kodlaması için fazla olabilir. Bu gibi durumda makine insanın yapabileceğinden fazlasını yapabilir.

Makine, değişikliklere kolayca adapte olabilir. Bazı görevlerde bilgiler değişebilir. Bu gibi durumlarda, yapay zekâ sistemini tekrar tasarlamak pratik değildir. Makine öğrenmesi metotlarını kullanarak bu gibi değişiklikler gözlenebilir (Carbonell ve diğerleri, 1989).

4.2. Derin Öğrenme

Derin öğrenme nesne tanıma, konuşma tanıma, doğal dil işleme gibi alanlarda çok katmanlı yapay sinir ağlarını kullanan bir yapay zekâ yöntemi olup makine öğrenmesinin çeşitlerinden biridir. Derin öğrenme, geleneksel makine öğrenmesi yöntemlerinden farklı olarak kodlanmış kurallar ile öğrenmek yerine; resim, video, ses ve metinlere ait verilerin simgelerinden otomatik olarak öğrenebilmektedir. Esnek yapıda olduklarından, ham resim

ya da metin verisinden de öğrenebilmekte ve verinin büyüklüğüne göre tahmin doğrulukları artabilmektedir. Bununla birlikte Derin Öğrenme, örnekler üzerinden öğrenme işlemini gerçekleştirmektedir (Buduma, 2017). Makinenin çözmesi istenen bir problem için kural setleri kullanarak çözüme ulaşmak yerine örnekleri değerlendirerek probleme çözüm getirmesini sağlayan bir model verilmesi yeterlidir. Problemin çözümündeki hatayı düzeltebilmesi için de basit bir komu listesi verilerek makineni öğrenme işlemini gerçekleştirmesi beklenmektedir. Model seçimi, problemin çözümünde etkindir. Probleme uygun olarak belirlenecek model, problemin çözümüne daha fazla katkıda bulunacaktır. Derin Öğrenme kavramı ilk kez 2006 yılında Hinton tarafından çok katmanlı yapay sinir ağlarının daha verimli eğitilebileceğinin öne sürülmesiyle ortaya çıkmıştır.

Günümüzde birçok alanda tasarlanıp kodlanan ya da yazılımlar yüklenen makinelerin insan yaşamını kolaylaştırması beklenmektedir. Ancak belirli kod bloklarından oluşan makinelerin kısıtlı bellekleri ile bunu nasıl gerçekleştirecekleri sorusu, makineler üzerinde yapılan yeni çalışmalar ile cevabını bulmaya başlamaktadır. Artık insanlar tarafından elle kodlanan ya da sadece belirli işlevleri yerine getiren makineler devri, yapay zekâ kavramının ortaya çıkmasıyla kapanmaktadır.

Yapay zekâ, makinelerin insandaki öğrenme sürecini taklit etmesi olarak tanımlanmaktadır. İnsan öğrenme işlemini beynin gerçekleştirdiği göz önüne alındığında, beynin bilgi işleme kapasitesi incelenmiştir. Makinelerin de insan gibi öğrenebilmesi için, beyinde öğrenme sürecinde etkin olan ve beynin bilgiyi işleme kapasitesini belirleyen nöron adi verilen sinir hücreleri taklit edilmeye çalışılmaktadır. İnsan sinir hücreleri olan nöronların taklit edilmesiyle yapay nöron kavramı ortaya atılmıştır.

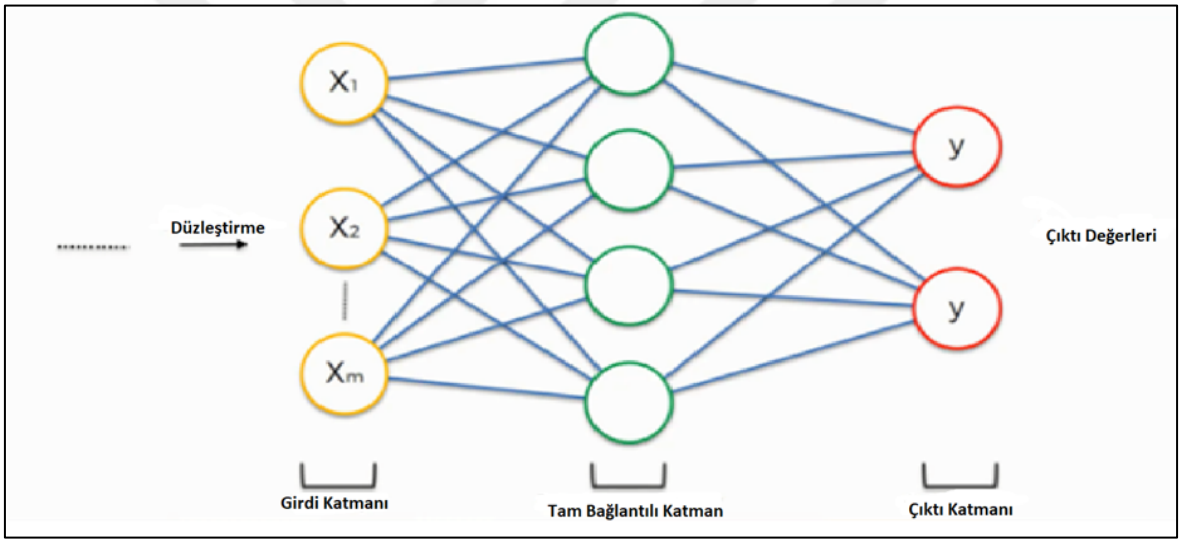
Sinir ağının temel yapısını oluşturan nöron beynimizin temel elemanıdır. Yeni bir bilgi alındığında bu bilgi işlenir ve sonra çıktıya dönüştürülür. Nöral ağda ise nörona gelen bir girdi alınır, işlenir ve sonraki işlem için diğer nöronlara gönderilen bir çıktı ya da sonuç çıktısını üretir.

Girdi nörona geldiğinde bir ağırlık ile çarpılır. İki girişli bir nöronda her bir nöron girişi, o girişe atanan bir ağırlığa sahiptir. Rastgele başlatılan bu ağırlıklar, modelin eğitimi süresince güncellenmektedir. Sinir ağı tarafından eğitim sonrasında daha önemli olduğu düşünülen girdilere daha yüksek ağırlık değeri verilmektedir. Bununla birlikte sinir ağı tarafından sıfır

ağırlık değeri verilen bir girdinin ya da özelliğin önemsiz olduğu kanaati oluşabilir. Bir a girdisinin w_1 ağrılığı ile ilişkilendirildiği varsayıldığında, düğümden geçtikten sonraki giriş $a * w_1$ olarak ifade edilir.

4.3. Tekrarlayan Sinir Ağları (RNN)

Tekrarlayan Sinir Ağları (RNN), Konvolüsyonel Sinir Ağı (CNN) ile beraber derin öğrenmenin iki temel mimarisinden biridir. Tekrarlayan Sinir Ağları zaman serisi şeklinde ifade edilen problemler için kullanılır. Örneğin ses işleme, sesten kelimeye, kelimeden sese, makine dönüştürmesi ve cümle özetlemesi gibi. Şekil 4.1'de basit Yapay Sinir Ağının yapısı görülmektedir.



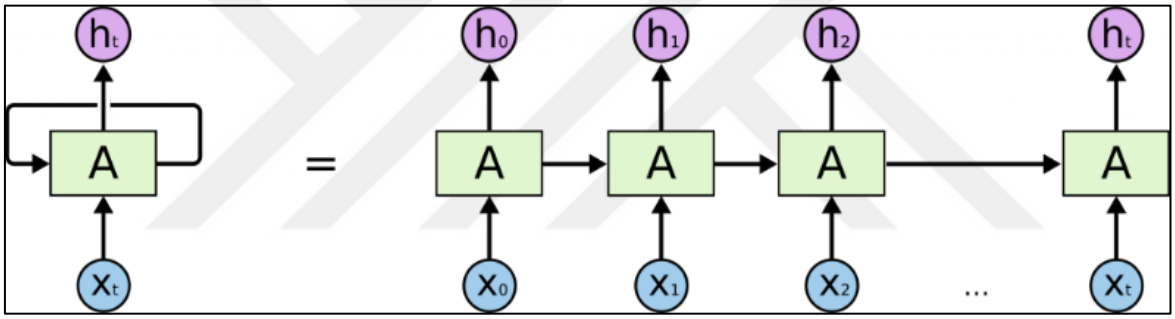
Şekil 4.1. Yapay sinir ağlarının yapısı

Konvolüsyonel Sinir Ağı mimarisinde, sinir ağı belirli bir boyutta bir veri alır. Bu veriyi sinir ağı içerisinde işlemden geçirip, işlemin sonucunda bir çıktı üretir. Örneğin girdi olarak resim verisi alıp çıktı olarak resmin sınıfının döndürülmesi gibi.

İnsan beyni belli yalnızca bir yöne doğru çalışan milyarlarca nöron hücresinden oluşmamaktadır. Bu açıdan Konvolüsyonel Sinir Ağı mimarisi insan beyninin veriyi işleme yapısıyla karşılaştırıldığında oldukça temel bir mimari olarak kalır. Bir karar verilirken yalnızca o anda ne görüp veya duymamız değil, geçmişte alınan kararların neden ve sonuçları açısından düşünülmesine de ihtiyaç olur.

Konvolüsyonel Sinir Ağı sınıflandırmaya dayalı işlemlerde çok başarılı olurken, ardışık gelen zamana göre ilerleyen konuşurken kullandığımız doğallık gibi verilerde benzer şekilde yeterince başarılı olmaz. Bir cümlede geçen kelimelerin sıralaması ve cümlelerin metin içerisindeki sıralaması çok önemlidir. Konvolüsyonel Sinir Ağında verilerdeki sıralama çok önemli değildir. Verilerdeki sıralamanın dikkate alınmaması ve geçmişteki verileri hatırlayacak bir hafızası da olmadığı için o anda alınan veri ile işlem yapılır. Yeniden bir girdi alındığında önceki işlenen verilerin bir önemi olmaz ve unutulur.

Tekrarlayan Sinir Ağı mimarisinde yapay sinir ağına daha önce işletilmiş zaman adımlarında yapılan işlem sonuçlarını hatırlaması için bir hafıza eklenir. Yapay sinir ağı yapısı sadeleştirilip Şekil 4.2'deki gibi gösterilebilir. Bu yapıya hafıza yapısını oluşturacak bir döngü eklenir ve kendi kendini beslemesini sağlar.

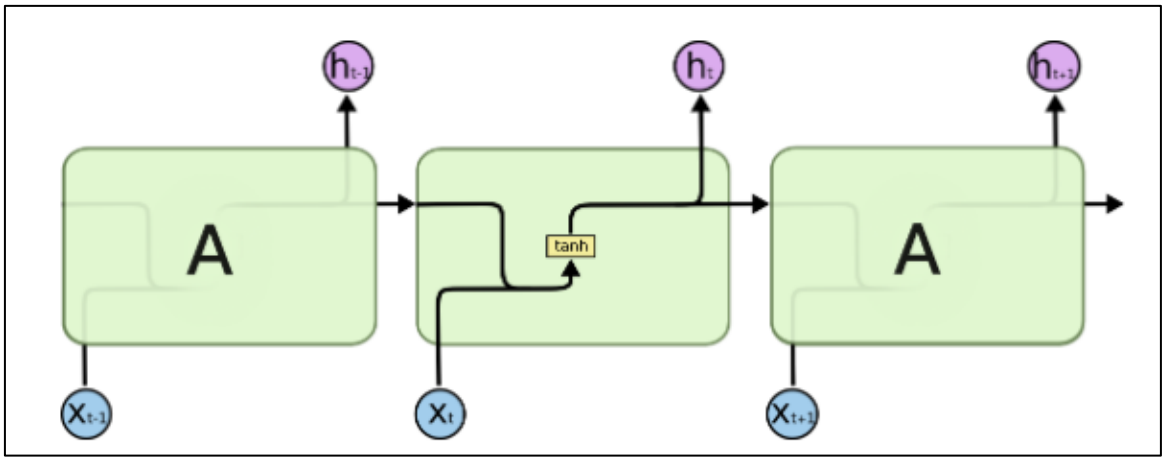


Şekil 4.2. Tekrarlayan sinir ağının yapısı (Sol kapalı – Sağ açık hali)

Eşitlik (4.1)'de basit RNN yapısının ileri besleme işlemi görülebilir. Bu eşitlikten anlaşılacağı üzere, aktivasyon fonksiyonu *Tanh (.)* olan basit bir sinir ağı kullanılmıştır. Bir önceki zaman adımının gizli hal bilgisi aktarılarak daha sonraki zaman adımındaki gizli hal bilgisi hesaplanabilir.

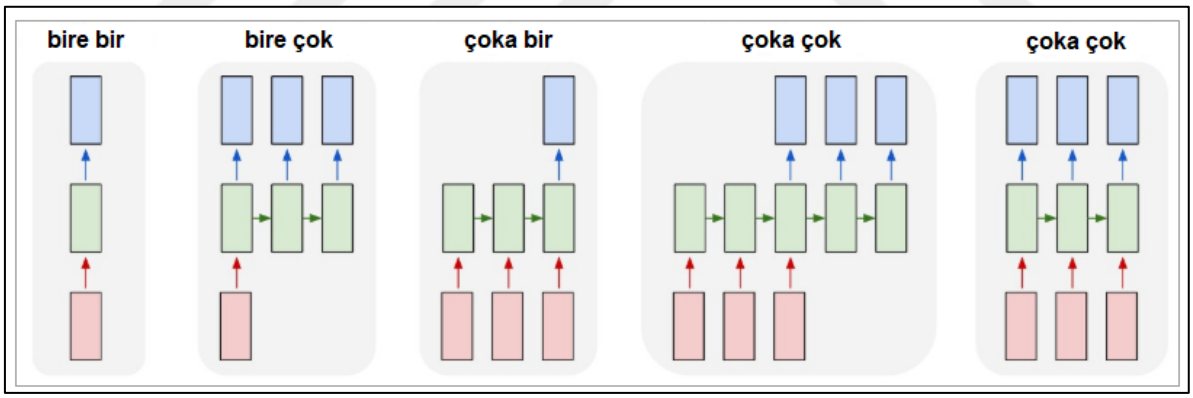
$$h_t = \tanh\left(W_x \cdot \left[x_t, h_{t-1} \right] \right) \tag{4.1}$$

$$\left(\frac{\partial L}{\partial h_t} \right) \left(\frac{\partial h_t}{\partial h_{t-1}} \right) \left(\frac{\partial h_{t-1}}{\partial h_{t-2}} \right) \cdots \left(\frac{\partial h_2}{\partial h_1} \right) \tag{4.2}$$



Şekil 4.3. Tekrarlayan sinir ağının basit iç yapısı

Tekrarlayan Sinir Ağının iç yapısı Şekil 4.3'teki gibi ifade edilebilir. Bu derin öğrenme algoritması belirli konular üzerinde çalışıldığı bilinse de problemlere göre RNN'in vereceği çıktıları çeşitli şekillerde kullanılabilir. Örnek olarak Şekil 4.4'teki gibi kırmızı nöronlar girdi vektörünü, mavi olanlar çıktı vektörünü, yeşil olanlar ise RNN'in gizli nöronlarını temsil etmektedir ve birden fazla problem çeşidine adapte olabilmektedir.



Şekil 4.4. Problem tiplerine göre tekrarlayan sinir ağı

Fakat bu yapıdaki RNN modelinin bazı sorunları ortaya çıkmaktadır. En önde gelen problem ise kaybolan gradyan (vanishing gradient) problemdir. Bu problem, model karmaşıklaştıkça (arka arkaya eklenen katmanlarının sayısı arttıkça) kendisini daha fazla göstermektedir.

İleri besleme işlemi sonuçlandıktan sonra, hata fonksiyonuyla hesaplanan modelin tahmin hatasını, modelin iç katmanlarındaki nöronlara dağıtmak gerekir. Bu dağıtılacak hata, zincir-kuralı uygulanarak modelin ağırlıklarının hesaplanan gradyanlarına göre dağıtılmasıyla olur.

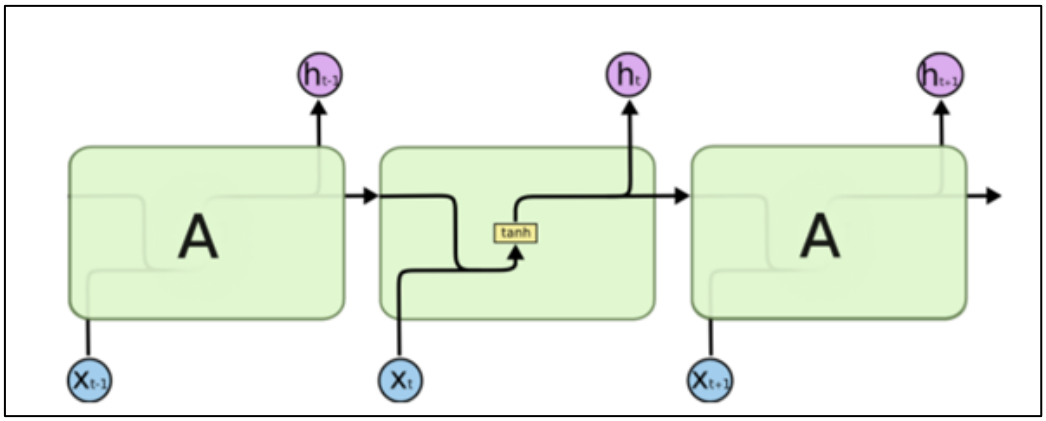
Bu işleme geri besleme adı verilir. Burada bahsedilen zincir-kuralı uygulanırken RNN modeli önce açılır, açıldığında kullanılan zaman serisi büyüklüğü alınacak olan gradyanı ya çok büyütür ya da çok küçültür. Bu yok olan ya da patlayan gradyan etkisi dizi boyunca uzun bağımlılık barındıran problemleri çözmeyi imkânsız kılar. Bu sorunun çözülmesi için Uzun-Kısa Vadeli Bellek (LSTM) modelini geliştirişmiştir (Hochreiter ve Schmidhuber, 1997).

4.4. Uzun-Kısa Vadeli Bellek (LSTM)

İnsanlar düşünmeye her şeyi bir kenara atıp yeniden sıfırdan düşünmeye başlamaz. Düşünceler kalıcıdır. Geleneksel sinir ağları bunu yapmaz ve bu büyük bir eksiklik gibi görünür. Örneğin, bir filmin her noktasında ne tür bir olay olduğunu sınıflandırmak istensin. Geleneksel bir sinir ağının, filmdeki önceki olaylarla ilgili akıl yürütmesini sonraki olayları bilgilendirmek için nasıl kullanabileceği belirsizdir. Tekrarlayan Sinir Ağları bu sorunu giderir. Bilginin kalıcı olmasını sağlayan döngüleri olan ağlardır.

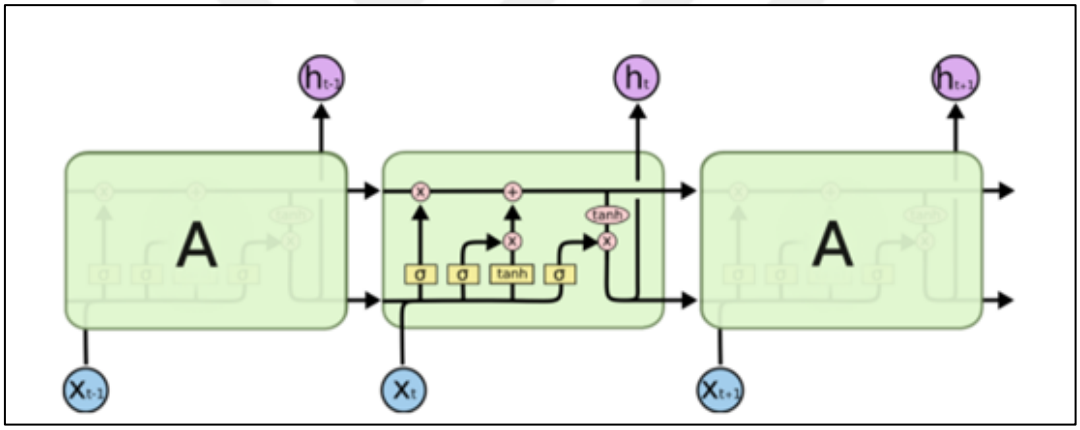
Son birkaç yılda, RNN'lerin çeşitli sorunlara uygulanma konusunda inanılmaz bir başarı elde edilmiştir. Bu başarıların temeli, birçok görev için standart versiyondan çok daha iyi çalışan çok özel bir tür tekrarlayan sinir ağı olan Uzun-Kısa Vadeli Bellek (LSTM) kullanılmasıdır. Tekrarlayan Sinir Ağlarına dayanan hemen hemen tüm sonuçlar LSTM ile elde edilir.

Uzun-Kısa Vadeli Hafıza Ağları genellikle LSTM'ler olarak adlandırılır. LSTM'ler uzun vadeli bağımlılıkları öğrenebilen özel bir RNN türüdür. Çok çeşitli sorunlar üzerine muazzam derecede çalışır ve kullanımı yaygındır. LSTM'ler uzun vadeli bağımlılık sorununun önüne geçmek için tasarlanmıştır. Uzun süre bilgi hatırlamak pratikte varsayılan davranışlarıdır, öğrenmek için uğraşmazlar. Tüm Tekrarlayan Sinir Ağları, tekrar eden sinir ağı modül zinciri biçimindedir. Standart RNN'lerde, bu yinelenen modül, tek bir katman gibi çok basit bir yapıya sahip olur.

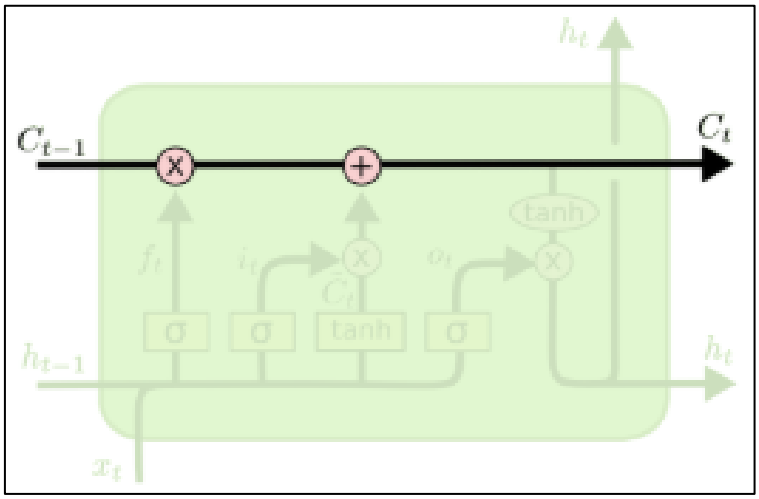


Şekil 4.5. Uzun-Kısa vadeli bellek yapısı

LSTM'lerin art arda birbirini takip eden yapıları vardır. Tek bir sinir ağı katmanı yerine, çok özel bir şekilde etkileşimde olan dört parça vardır.



Şekil 4.6. Uzun-Kısa vadeli bellek modelinin iç yapısı



Şekil 4.7. Uzun-Kısa vadeli bellek hücre bilgisi aktarımı

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \quad (4.3)$$

$$i_t = \sigma(W_f.[h_{t-1}, x_t] + b_i) \quad (4.4)$$

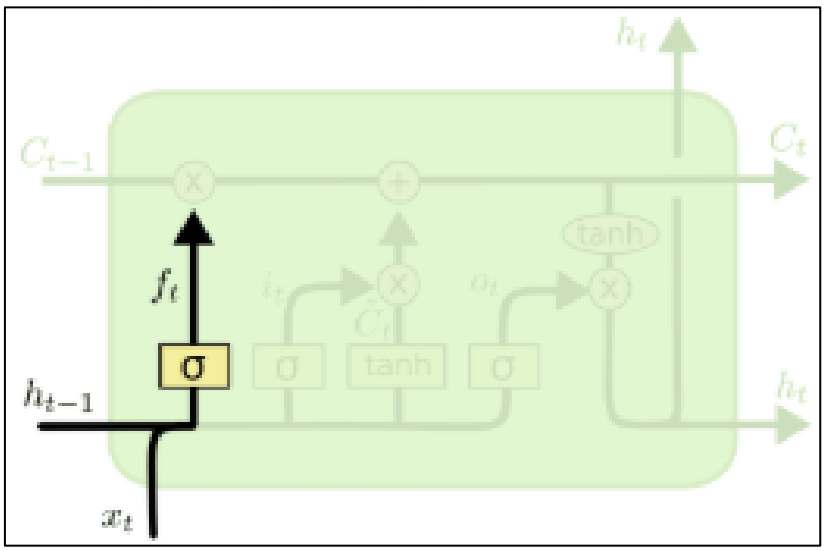
$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o) \quad (4.5)$$

$$\hat{C}_t = \tanh(W_c.[h_{t-1}, x_t] + b_c) \quad (4.6)$$

$$C_t = f_t \times C_{t-1} + i_t.\hat{C}_t \quad (4.7)$$

$$h_t = o_t \times \tanh(C_t) \quad (4.8)$$

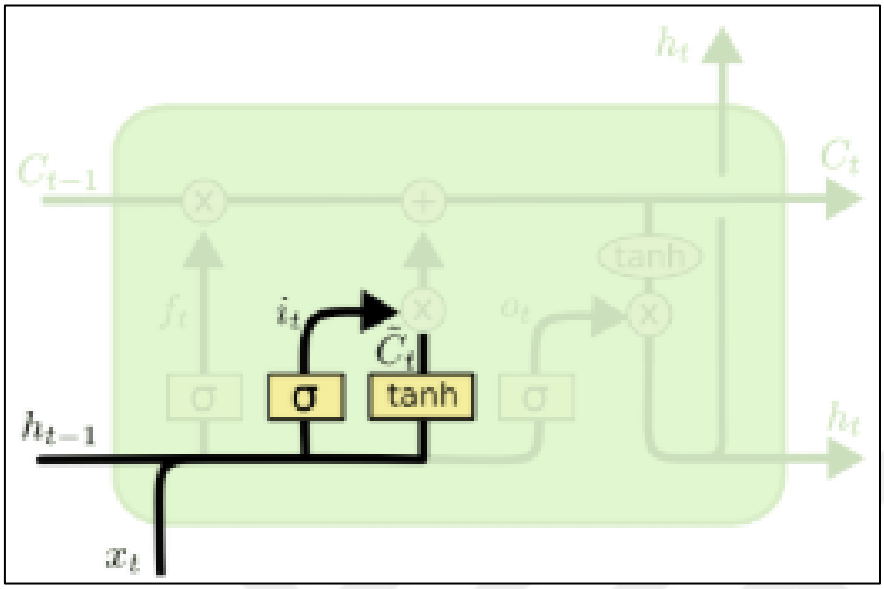
Sigmoid fonksiyonu 0 ile 1 arasında bir çıktı vermektedir. 0 çıktısı "hiçbir şeyi geçirme", 1 çıktısını ise "her şeyi geçir" demektir. LSTM'in sahip olduğu bu 3 kapı LSTM'in hücre halini korumasını, saklamasını ya da unutmasını sağlar. Şekil 4.8'de gösterilen sinir ağı, t anındaki girdi ve $t\ -1$ anındaki gizli hali girdi olarak alır ve hücre halinin ne kadar bilgiyi unutup unutmayacağına karar verir. Buradaki ft , 1'e yaklaştıkça $t-1$ anındaki hücre hali t anına aktarılır. Aynı şekilde 0'a yaklaştıkça $t-1$ anındaki hücre hali unutularak sonraki sinir ağlarından gelen bilgiler bir sonraki *t* anına aktarılır. Eşitlik (4.3)'de *t* anındaki unut kapısının nasıl hesaplandığı görülebilir.



Şekil 4.8. Uzun-Kısa vadeli bellek unut kapısı

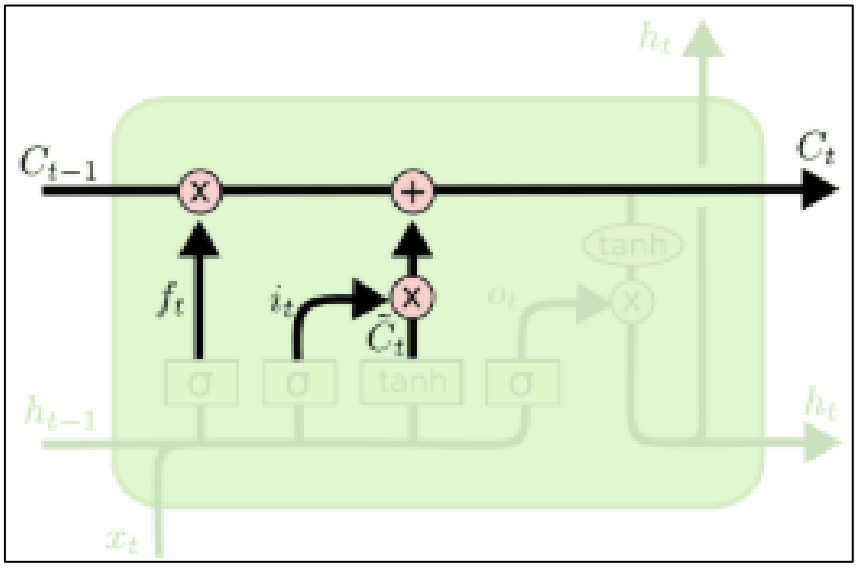
Şekil 4.9'da gösterilen aktivasyon fonksiyonu sigmoid olan sinir ağı, *t* anındaki girdi ve *t-1* anındaki gizli hali girdi olarak alır ve çıktı olarak verdiği i_t de -1 ile 1 arasına sıkıştırılmış olan kısmi yeni hücre bilgisinin ne kadarının eski hücre bilgisine ekleneceğine karar verir.

Eşitlik (4.4)'de *t* anındaki girdi kapısının, eşitlik (4.6)'de *t* anındaki kısmi yeni hücre bilgisinin nasıl hesaplandığı görülebilir.



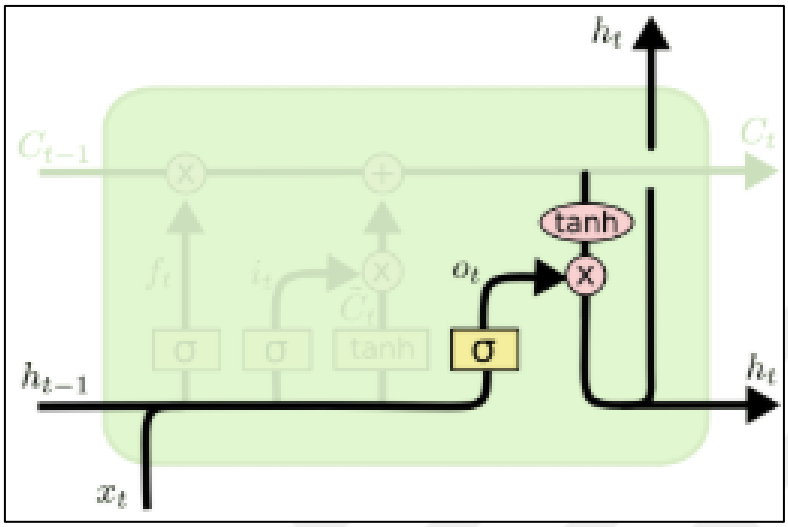
Şekil 4.9. Uzun-Kısa vadeli bellek girdi kapısı

Şekil 4.10'da unut kapısının verdiği karar $t-1$ anındaki hücre bilgisiyle çarpılıp, girdi kapısının ne kadar yeni hücre bilgisinin sisteme dahil edileceğine karar verdiği vektör ile toplanarak *t* anındaki "bir sonraki nörona aktarılacak ya da özyinelemeli şekilde kendisine tekrar girecek olan "hücre bilgisi elde edilmiş olur. Eşitlik (4.7)'de *t* anındaki hücre bilgisinin nasıl hesaplandığı görülebilir.



Şekil 4.10. Uzun-Kısa vadeli bellek hücre bilgisi güncelleme

Şekil 4.11'de -1 ile 1 arasına sıkıştırılmış *t* anındaki hücre bilgisi, *t-1* anından gelen gizli hal bilgisi ile çarpılarak *t* anındaki -bir sonraki nörona aktarılacak ya da özyinelemeli şekilde kendisine tekrar girecek olan gizli hal bilgisi elde edilmiş olur. Eşitlik (4.5)'de *t* anındaki çıktı kapısının, eşitlik (4.8)'da *t* anındaki gizli hal bilgisinin nasıl hesaplandığı görülebilir.



Şekil 4.11. Uzun-Kısa vadeli bellek gizli durum bilgisi güncelleme

LSTM, gradyanların yok olması ya da patlaması sorununu büyük ölçüde çözmüş olsa da nasıl ve ne şekilde kullanılması gerektiği problemin ve kullanılan zaman serisinin yapısına bağlı olduğu için uygulanması karışık olan bir model olmaktadır.

4.5. Aktivasyon Fonksiyonu

Giriş sinyallerini, çıkış sinyallerine çeviren aktivasyon fonksiyonunun lineer bileşime uygulanması; doğrusal bileşenin girdiye uygulanmasından sonra doğrusal olmayan fonksiyon uygulanması ile gerçekleştirilir. Aktivasyon fonksiyonu olan $f(.)$ uygulandıktan sonraki giriş $f(a * w_1 + b)$ olacaktır. Burada b sapma olarak tanımlanmaktadır.

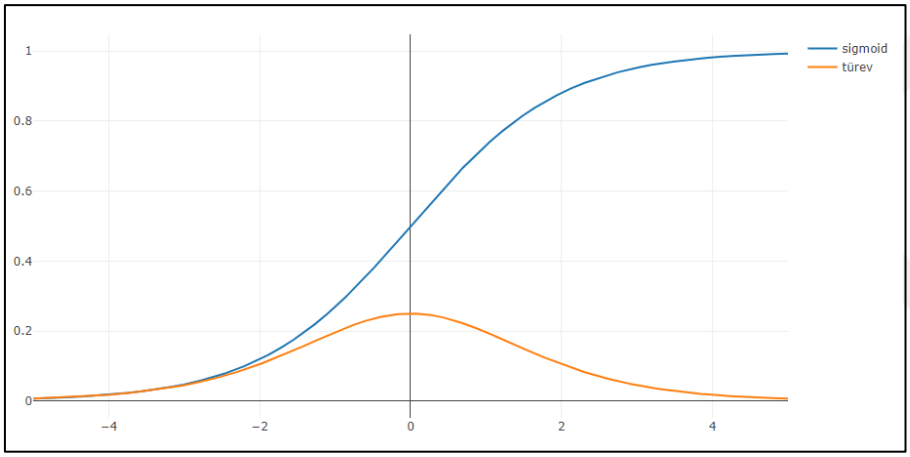
Ağırlıklar önce kendilerine karşılık gelen girdi ile çarpılır ve sonra sapma çarpım sonucuna eklenir. Ağırlık toplamlarına karşılık gelen girdi toplamlarının çarpımına sapma değeri eklenmesi sonucu oluşan ifade *u* olarak kabul edilirse Eşitlik 4.9'daki gibi ifade edilir:

$$u = \sum w * x + b \tag{4.9}$$

Aktivasyon fonksiyonunun u'ya uygulanması sonucu $f(u)$ oluşur ve nörondan elde edilen son çıktı $y_k = f(u)$ olarak ifade edilir.

4.5.1. Sigmoid

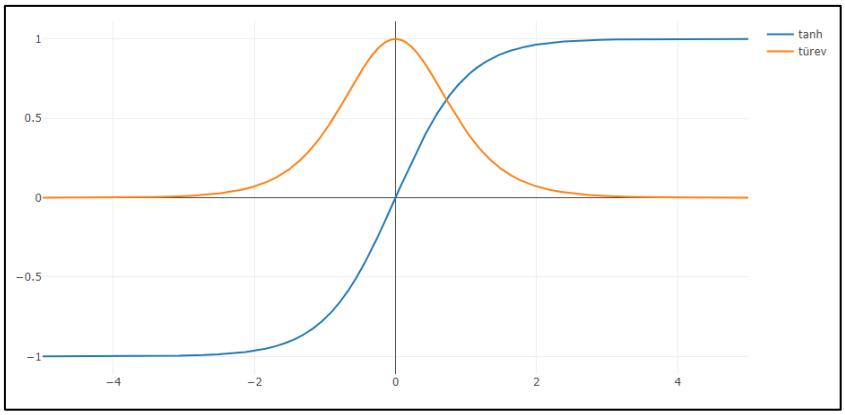
0 ile 1 arasında daha düzgün bir değer aralığı oluşturan sigmoid dönüşümü, pürüzsüz eğrilere sahiptir. Girdi değerlerindeki küçük değişikliklerin çıktı değerlerinde gözlemlenmesine imkân tanıyan pürüzsüz eğriler, adım fonksiyonlarında tercih edilir. Şekil 4.12'de Sigmoid fonksiyonunu grafiği verilmiştir.



Şekil 4.12. Sigmoid fonksiyon grafiği

4.5.2. Tanh (Hiperbolik Tanjant)

Tanh (.) fonksiyonu sigma fonksiyonunun ölçeklendirilip kaydırılmasıyla elde edilmiştir. Bu nedenle sigma fonksiyonunun tüm avantaj ve dezavantajlarına sahiptir. Yalnızca sigma fonksiyonuna göre daha büyük bir gradyan değerine sahip olduğu için gradyanların yok olması problemini daha az ortaya çıkarır. Şekil 4.13'de *Tanh (.)* fonksiyonunu grafiği verilmiştir.



Şekil 4.13. Tanh fonksiyon grafiği

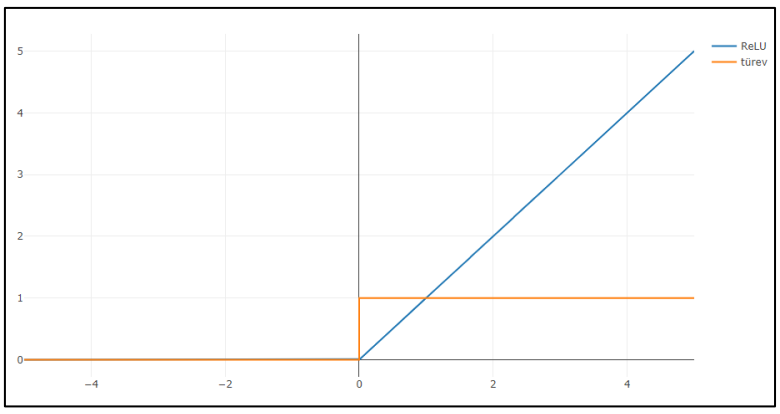
4.5.3. Doğrultulmuş lineer birim (ReLu)

Son zamanlarda ağların gizli katmanlarında sigmoid yerine Doğrultulmuş Lineer Birim (Rectified Linear Units - ReLu) aktivasyon fonksiyonu tercih edilmektedir. Bu fonksiyon Eşitlik (4.10) şekilde tanımlanmaktadır:

$$f(x) = max(x, 0) \tag{4.10}$$

Eşitlik 4.10'da verilen fonksiyon çıktısı X, sıfırdan büyükken $(X > 0)$ sonuç X alınır ve X sıfırdan küçük ve sıfıra eşit ise $(X \leq 0)$ sonuç 0 olarak alınır.

ReLu aktivasyon fonksiyonunun kullanılmasının en büyük faydası 0'dan büyük tüm girdiler için ağın daha hızlı eğitilmesini sağlayan sabit türev değerine sahip olmasıdır. Şekil 4.14 'de *ReLu* fonksiyonunu grafiği verilmiştir.



Şekil 4.14. ReLu fonksiyon grafiği

4.6. Kaybolan Eğim Problemi

Ardışık veriler için bir önceki çıktının bir sonraki çıktıyı tahmin etmesi için kullanıldığı ağlardır. Tekrarlayan sinir ağlarının içinde döngüler bulunmaktadır ve çıktının tahmin edilebilmesi için bir süre önceki bilgilerin depolanmasını sağlayan döngüler gizli nöronlar içinde bulunmaktadır. *t* anındaki zaman bilgisi için gizli katmandan elde edilen çıktı, gizli katmana geri yollanır. Tüm zaman bilgisi sağlandıktan sonra tekrarlayan nörona ait çıkış, bir sonraki katmana gönderilmektedir. Önceki çıktıya göre bu çıktı daha genel bilgileri içermektedir. Önceki bilgi daha uzun zaman saklanmaktadır. Hatanın geri yayılımı için teşhir edilmemiş ağın ağırlıklarının güncellenmesi gerekmektedir. Buna zaman içinde geri yayılım denilmektedir.

Aktivasyon fonksiyonuna ait eğimin çok küçük olması, kaybolan eğim problemine neden olmaktadır. Geri yayılım esnasında ağırlıklar, bu düşük eğimlerin çarpılmasıyla daha da küçülür ve ağ içinde daha derine ilerledikçe kaybolmaya doğru ilerler. Bu sorun ağın uzun dönemli bağımlılıklarını hatırlaması önemli olup uzun dönemli bağımlılığın unutulması bu ağlarda ciddi problemlere yol açmaktadır. Bunun çözümü için de *ReLu* gibi düşük eğimlere sahip olmayan aktivasyon fonksiyonları kullanılması gerektiğidir.

4.7. Patlayan Eğim Problemi

Kaybolan eğim probleminin tam tersi olan bu problemde aktivasyon fonksiyonunun eğimi çok büyüktür. Geri yayılım sırasında bazı düğümlerin ağırlık değerlerinin çok yüksek olması bu düğümleri diğer düğümlere göre önemsiz olarak yorumlanmasına neden olur. Bu problemin çözümlenebilmesi için eğimin belirlenen bir değerde kesilmesi ve o değeri aşmayacak şekilde belirlenmesi gerekir.

5. UYGULAMA (ANALİZ SONUÇLARI)

Bu tez çalışmasında analizlerin yapılması ve derin öğrenme algoritmalarının uygulanması için R (3.6.3) paket programından kullanılmış ve ilgili kütüphaneler konu başlıkları içinde sunulmuştur.

5.1. Augmented Dickey-Fuller Testi Analiz Sonuçları

Zaman serisi verisine yönelik yapılan analizlerde en sık karşılaşılan sorun söz konusu serilerin durağan olmamasından kaynaklanır. Değişkenler arasında yapılan analizlerde anlamlı sonuçlar elde etmek için analiz uygulanacak serilerin trend taşımamaları gerekmektedir.

Bir zaman serisinin durağan olup olmadığını test etmek için Augmented Dickey-Fuller testi kullanılır. Test işlemini gerçekleştirmek için R (3.6.3) paket programında t*imeseries* kütüphanesi kullanılmıştır.

H_0: Zaman serisi durağan değildir. (Başka bir deyişle, zamana bağlı bir yapıya sahiptir ve zaman içinde sabit bir varyansa sahip değildir.)
H_1: Zaman serisi durağandır.

Test sonuçları Çizelge 5.1'de verilmiştir. Eğer *p*-değeri anlamlılık seviyesinden düşükse o zaman sıfır hipotezi kabul edilemez ve zaman serisinin durağan olduğu sonucuna varabiliriz. Çizelge 5.1.'den görüldüğü gibi elde edilen test sonuçlarının *p*-değerinden (0.05), büyük olduğu yani durağan olmadığı sonucuna varılmaktadır.

Çizelge 5.1. Bist100 ve USD/TRY verisi için uygulanan Augmented Dickey Fuller test değerleri.

	Dickey Fuller Değeri	Gecikme Değeri	*p*-değeri	Durağanlık Durumu
USD/TRY Kapanış	-0.22868	10	0.99	Durağan değil
Bist100 Kapanış	-2.438	10	0.3929	Durağan değil

5.2. Engle-Granger Koentegrasyon (Eşbütünleşme) Testi Sonuçları

Bir zaman serisinin durağan olmadığı durumda eşbütünleşme testi uygulanır. Augmented Dickey-Fuller test istatistiği sonuçları zaman serisinin durağan olmadığını göstermektedir. Bu durumda H_0 hipotezi, alternatif hipoteze karşı test edilir. Engle Granger testi için H_0 hipotezi, eşbütünleşmenin olmadığı şeklindedir. Test işlemini gerçekleştirmek için R (3.6.3) paket programlamada *aTSA* kütüphanesi kullanılmıştır.

H_0**:** Eşbütünleşmeye sahip değildir.
H_1: Eşbütünleşmeye sahiptir.

Test sonuçları Çizelge 5.2'de verilmiştir. Eğer *p*-değeri anlamlılık seviyesinden düşükse o zaman sıfır hipotezi kabul edilemez ve eşbütünleşmenin olduğu sonucuna varılabilir. Çizelge 5.2. incelendiğinde elde edilen test sonuçlarının *p*-değerinden (0.05), küçük olduğu yani eşbütünleşmenin olduğu sonucuna varılır. Böylece iki değişkenin gelecekteki değerlerini tahmin etmek için önemli olduğu sonucuna varılır.

Çizelge 5.2. Bist100 ve USD/TRY verisinin kapanış değişkenlerinin eşbütünleşme test sonuçları.

	Gecikme Değeri	Engle-Granger Değeri	*p*-değeri
Tip 1: Trend Yok	7	-0.951	0.0100
Tip 2: Lineer Trend	7	1.62	0.010
Tip 3: Karesel Trend	7	-1.16	0.010

5.3. Tekrarlayan Sinir Ağı (RNN) Model Sonuçları

Öncelikle, fiyat ya da trend tahmini yapmak için verilerin elde edilmiş olması gerekir. Bu verileri elde etmek için Yahoo Finance (https://finance .yahoo.com/quote/XU100.IS?p =XU100.IS&.tsrc=fin-srch, https:// finance. yahoo.com /quote/ TRY=X?p=TRY =X&.tsrc = fin-srch) kullanılmıştır. Yahoo Finance'in sağladığı veriler günlük olup; tarih, açılış, kapanış, en düşük, en yüksek, işlem hacmi ve düzenlenmiş kapanış değerlerini barındırır. Düzenlenmiş kapanış değerleri, kâr payı, sermaye değişmeden hisse sayısının değiştirilmesi ya da bedelli sermaye artırımı gibi sebeplerden bozulan kapanış değerleri verisinin düzenlenmesi ile oluşturulmuştur.

Eğitim ve test verisi belirlenirken makine öğrenmesi uygulamalarında sıkça %80 eğitim ve %20 test verisi olarak dağılım yapılır (Gholamy ve diğerleri, 2018).

Finans verileri doğası gereği yüksek değişkenliğe sahiptir. Tek bir şirkete ait hisselerin değişkenliği, bunların birleşiminden oluşan borsa yatırım fonlarından daha fazladır. Finansal veriyi analiz etmek için tüm yöntemler temel analiz ve teknik analiz olmak üzere iki ana çatı altında birleşir. Temel analiz, şirketlerin söylemleri, yıllık kazançları gibi değişkenleri inceler. Bu tip değişkenlerden öznitelik çıkarmak için doğal dil işleme gibi farklı makine öğrenmesi alanlarına girilmelidir. Teknik analiz ise, finans verisindeki sayısal serilerle ilgilenir ve daha az özel bilgi gerektirir. Teknik analiz için incelenecek yöntemler, finansın zaman serisi verilerine sahip olmasından kaynaklı olarak, zaman serisi analizi altında toplanabilir ve LSTM, RNN, CNN (belirli önişlemler yapılırsa) gibi mimarilerin kullanımına olanak tanır.

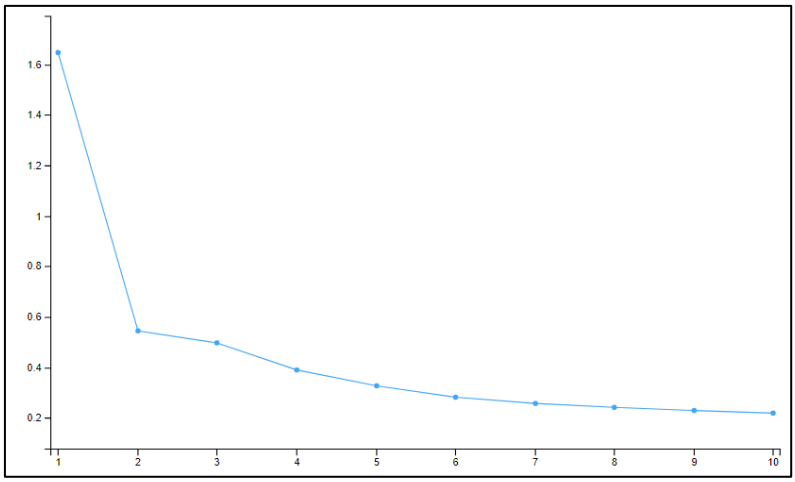
Algoritmaların uygulanabilmesi için R (3.6.3) paket programında *keras* ve *tensorflow* kütüphaneleri ve grafiklerin oluşturulması için *ggplot2* kütüphanesi kullanılmıştır.

Çizelge 5.3. Tekrarlayan Sinir Ağı ile oluşturulan model sonuçları

Katman (Tip)	Çıktı Şekli	Ağırlık Katsayısı
Düzleşmiş RNN	64	0
Yoğunluk	1	33
Yoğunluk	32	2080
Yoğunluk	64	30784

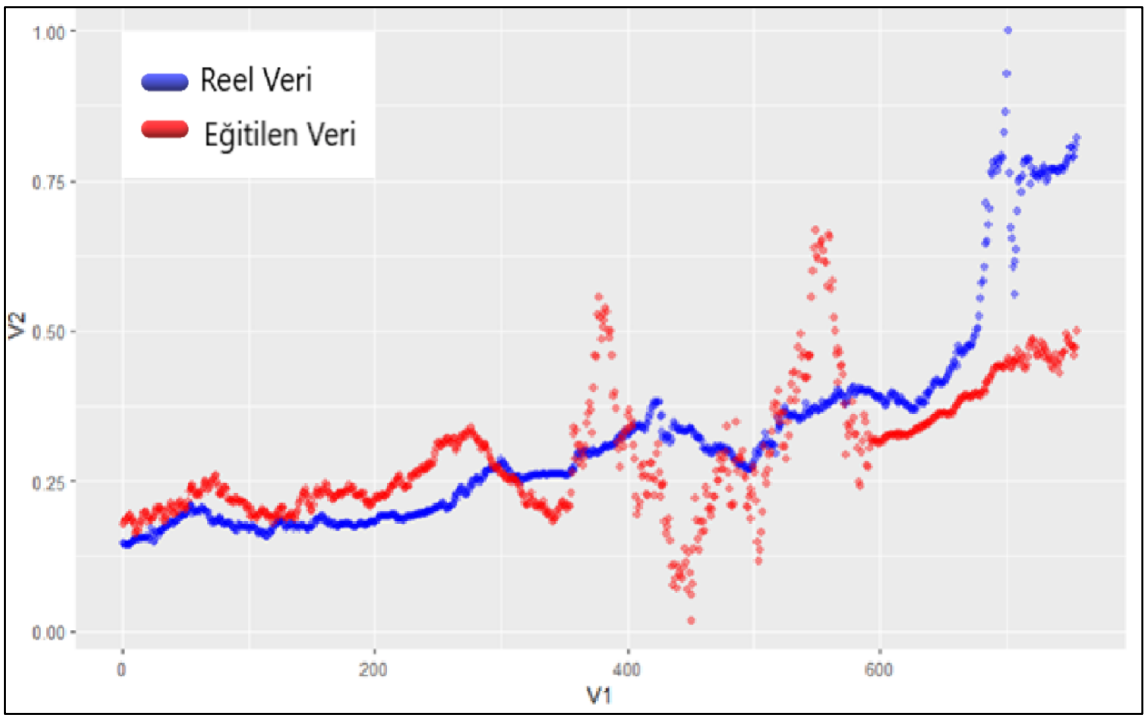
Çizelge 5.3. incelendiğinde, modele 1, 32 ve 64 adet RNN katmanı eklenmiş ve modelimiz oluşturulmuştur. Oluşan model sonucunda 32897 ağırlık katsayısına ulaşıldığı görülmüştür.

Çalışmada öğrenme iterasyon değerimiz 10 verilmiştir ve her döngüdeki hata değerleri grafiği Şekil 5.1'de gösterilmiştir. Burada model hızlı öğrenmektedir.



Şekil 5.1. Tekrarlayan sinir ağına ait öğrenme eğrisi

Şekil 5.2'de kırmızı eğri eğitilen veriyi, mavi eğri ise reel veriyi temsil etmektedir. Elde edilen sonuçlardan da anlaşılacağı üzere Tekrarlayan Sinir Ağı (RNN) ile yapılan tahminin sadece belli kesitlerde sapma yaptığı söylenebilir. Şekil 5.2. detaylı incelenirse orta kısımlarda yani 10.06.2018 tarihinden 10.10.2018 tarihleri arasında büyük sapmaların gerçekleştiği ve bu sapmanın 2018'de yapılan seçim kaynaklı olduğu, modelin o dönemde yaşanan dalgalanmalardan etkilendiği ve sonlara doğru ise bu sapmaların düzeldiği ancak Tekrarlayan Sinir Ağının etkili bir tahmin yapamadığı görülmektedir.



Şekil 5.2. Tekrarlayan sinir ağı sonucu

5.4. Uzun-Kısa Vadeli Bellek (LSTM) Model Sonuçları

Uzun-Kısa Vadeli Bellek (LSTM) hücresi, giriş, çıkış ve ayrıca giriş zaman adımları yoluyla oluşturulan parametrelere sahiptir. Giriş ağırlıkları, bulunulan zamanda girdiyi ağırlıklandırmak için kullanılır. Çıkış ağırlıkları, en son zaman adımından çıktıyı ağırlıklandırmak için kullanılır ve dahili durum, sonucun hesaplanmasında kullanılan dahili durum parametresidir. Sıfırlama periyodu daha küçük bir sayı seçilirse bilgiye ulaşmada sıfırlanma söz konusu olabilmekte, daha büyük bir sayı seçildiğinde ise gereksiz gradyan hesabı yapılarak hesaplama masrafı arttırmaktadır. Bu tarz problemler için çeşitli optimizasyon algoritmaları kullanılır.

Optimizasyon açısından çözülmesi gereken karmaşık bir problem bulunmadığı için çok kritik bir karar vermeden en yaygın olarak kullanılan, hatada meydana gelen dalgalanmayı azaltan ve kolay uyum sağlayabilen ADAM optimizasyon algoritması kullanılmıştır. ADAM yani Adaptif Moment Tahmini (Adaptive Moment Estimation) karelerin ortalamasının karekökü yayılımı yöntemine momentum ekleyen ve daha yaygın kullanılan bir yöntemdir. Momentum güncellemesi üstel hareketli ortalama ile yapılır ve β ile uğraşırken öğrenme oranını değiştirmemiz gerekmez.

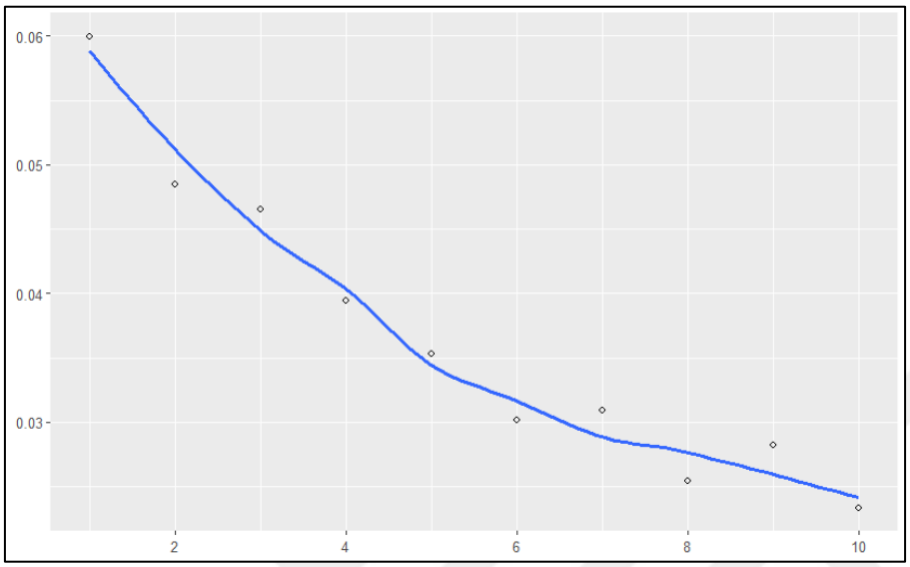
Regresyon problemi ile karşılaşma olasılığından dolayı ise masraf fonksiyonu olarak ortalama kare hata fonksiyonu kullanılmıştır. Bu çalışmada aktivasyon olarak *ReLu* ve *Tanh (.)* ile çalışılmıştır. Algoritmaların uygulanabilmesi için R (3.6.3) paket programında *keras* ve *tensorflow* kütüphaneleri ve grafiklerin oluşturulması için *ggplot2* kütüphanesi kullanılmıştır.

Çizelge 5.4. incelendiğinde, modelimize 1, 32 ve 64 adet LSTM katmanı eklenmiş ve ortalama kare hatası 0,0050 olacak şekilde oluşturulmuştur. Oluşan model sonucunda 23169 ağırlık katsayısına ulaşıldığı görülmüştür.

Çizelge 5.4. Uzun-Kısa Vadeli Bellek (ReLu) model sonuçları

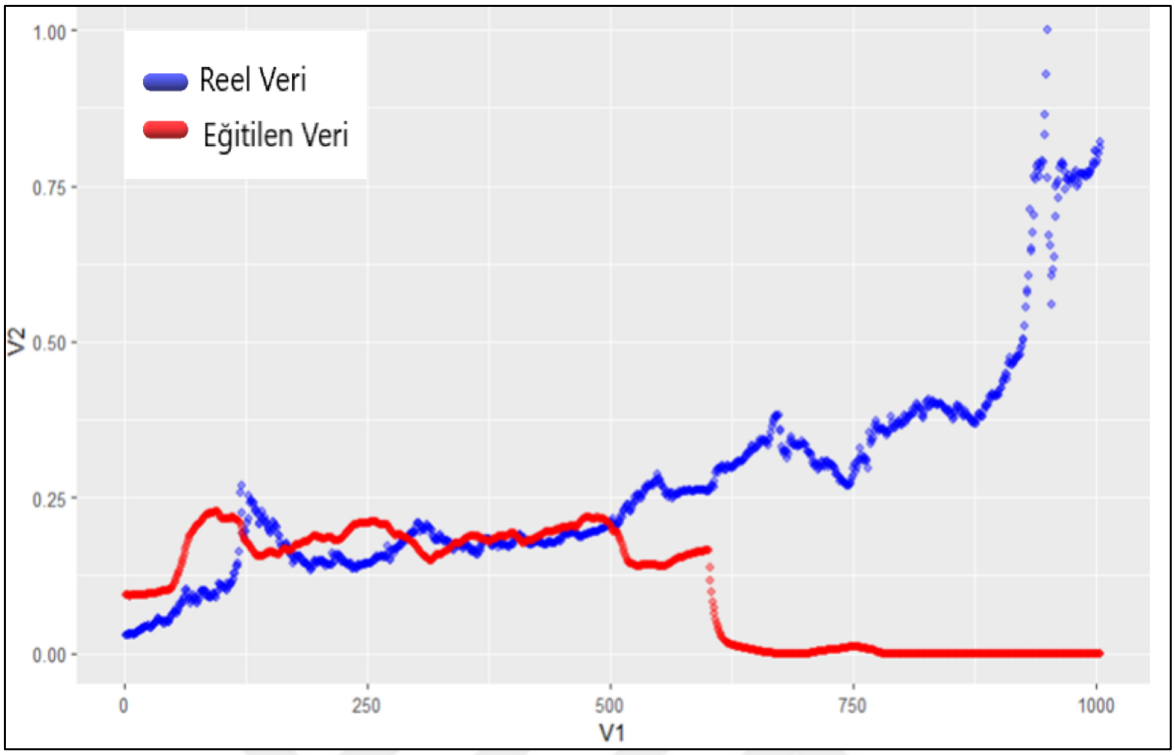
Katman (Tip)	Çıktı Şekli	Ağırlık Katsayısı
LSTM ReLu	64	16896
Yoğunluk	1	33
Yoğunluk	32	2080
Yoğunluk	64	4160

Çalışmada, öğrenme iterasyon değerimiz 10 verilmiştir ve her döngüdeki hata değerleri grafiği Şekil 5.3'de gösterilmiştir. Burada model Tekrarlayan Sinir Ağı (RNN)'ye göre daha yavaş öğrenmektedir.



Şekil 5.3. Uzun-Kısa vadeli belleğe (ReLu) ait öğrenme eğrisi

Şekil 5.4'de kırmızı eğri eğitilen veriyi, mavi eğri ise reel veriyi temsil etmektedir. Elde edilen sonuçlardan da anlaşılacağı üzere Uzun-Kısa Vadeli Bellek (LSTM-ReLu) ile yapılan tahminin sadece başlangıç kesitlerde etkili tahmin yaptığı söylenebilir. Şekil 5.4. detaylı incelenirse orta veri kesitinin yarısından sonra yani Nisan 2019 sonrası için etkili tahmin yapamadığı görülmektedir. Bu bozulmanın sebebi ise yaşanan pandemiden kaynaklanmaktadır.



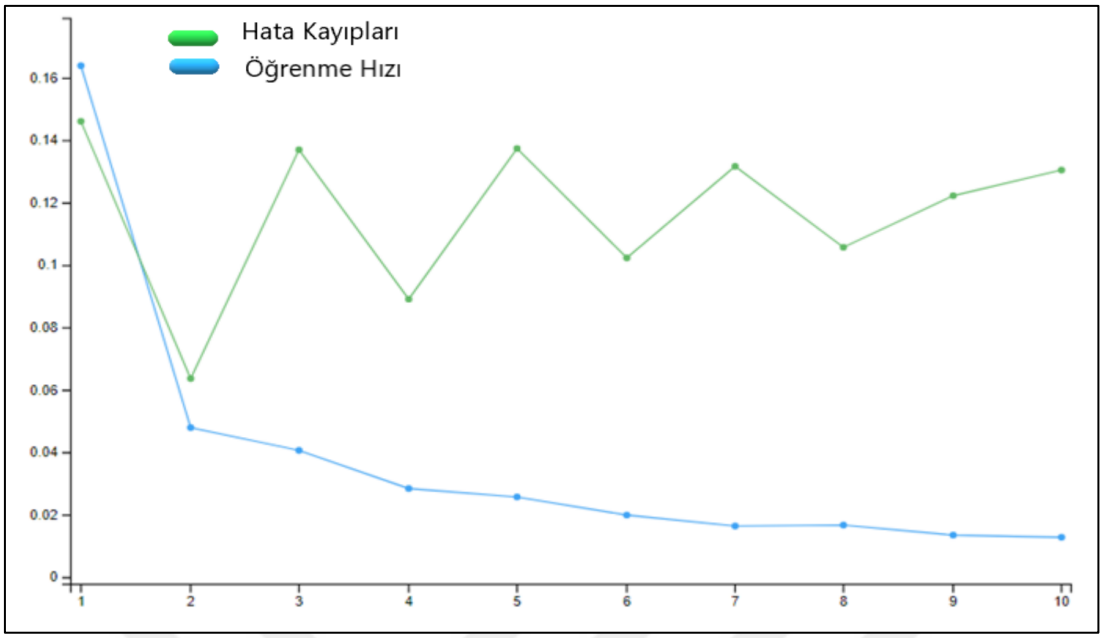
Şekil 5.4. Uzun-Kısa vadeli bellek (ReLu) sonucu

Çizelge 5.5. incelendiğinde, modelimize 1,32 ve 64 adet Uzun-Kısa Vadeli Bellek (LSTM-Tanh) katmanı eklenmiş ve ortalama kare hatası (MSE) 0,0050 olacak şekilde oluşturulmuştur. Oluşan model sonucunda 73345 ağırlık katsayısına ulaşıldığı görülmüştür.

Çizelge 5.5. Uzun-Kısa Vadeli Bellek (Tanh) model sonuçları

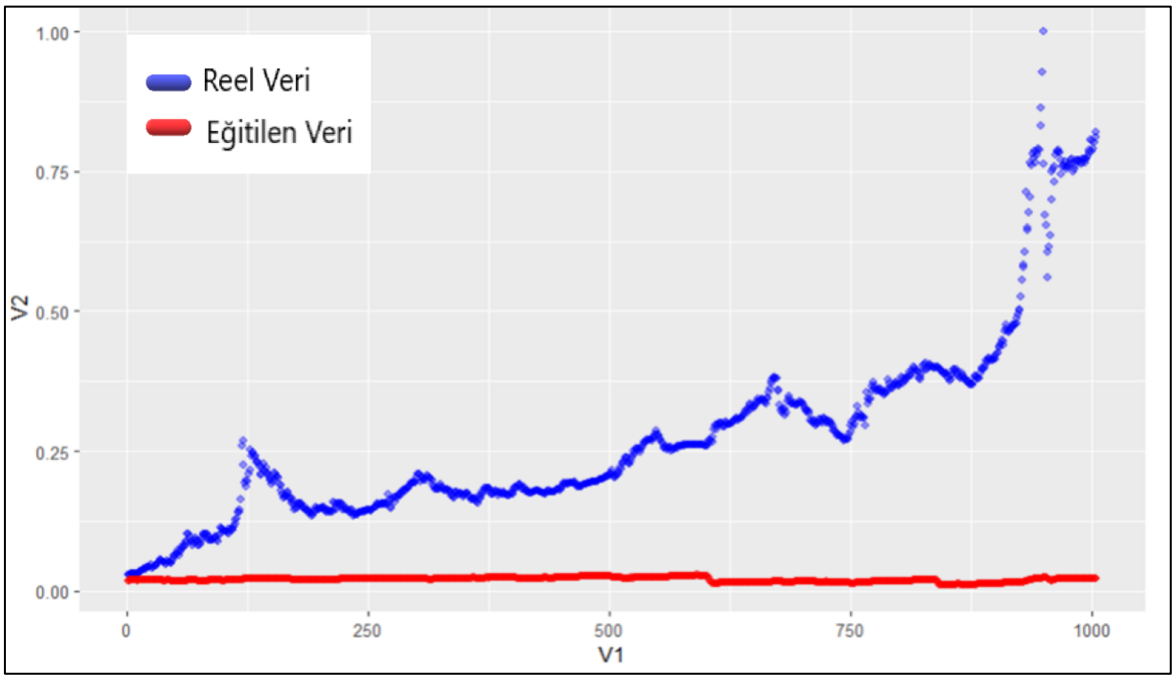
Katman (Tip)	Çıktı Şekli	Ağırlık Katsayısı
LSTM Tanh (.)	64	17152
Yoğunluk	1	33
Yoğunluk	32	2080
Yoğunluk	64	4160

Çalışmada, öğrenme iterasyon değerimiz 10 verilmiştir ve her döngüdeki hata değerleri grafiği Şekil 5.5'de gösterilmiştir. Burada model Tekrarlayan Sinir Ağı (RNN) ve Uzun-Kısa Vadeli Bellek (LSTM)'ye göre öğrenme hızı aynı olduğu fakat hata kayıplarının çok fazla olduğu görülmektedir.



Şekil 5.5. Uzun-Kısa vadeli belleğe (Tanh) ait eğitim eğrisi

Şekil 5.6'da kırmızı eğri eğitilen veriyi, mavi eğri ise reel veriyi temsil etmektedir. Elde edilen sonuçlardan da anlaşılacağı üzere Uzun-Kısa Vadeli Bellek (LSTM-Tanh) ile yapılan tahminin beklentinin dışında sonuç verdiği söylenebilir. Şekil 5.6 detaylı incelenirse tahmin eğrisi, gerçek veri eğrisine yaklaşamamıştır.



Şekil 5.6. Uzun-Kısa vadeli bellek (Tanh) sonucu

6. SONUÇLAR

Bu çalışmada, derin öğrenme modellerinden Tekrarlayan Sinir Ağı (RNN) ve Uzun-Kısa Vadeli Bellek (LSTM) mimarisi kullanılarak Bist100 ve USD/TRY kapanış fiyatlarının tahminine yönelik bir model geliştirilmiştir. 09.03.2017 ile 09.03.2022 yılları arasındaki günlük kapanış değerleri kullanılarak belirli bir döneme ait değerleri 0,0050 hata girdisi ile tahmin edilmiştir.

Verilerin yaklaşık %80'i (995 adet) eğitim için kullanılırken yaklaşık %20' si (249 adet) test verisi olarak kullanılmıştır. Uzun-Kısa Vadeli Bellek (LSTM) ve Tekrarlayan Sinir Ağı (RNN) modelinde giriş, ara ve çıkış olmak üzere her katmanda R (3.6.3) programlama dili kullanılmış ve ortalama 43.137 parametre eğitilmiştir.

Modellerin tahmin performanslarını ölçmek için Uzun-Kısa Vadeli Bellek (LSTM) kullanıldığında ve aktivasyon fonksiyonu olarak *ReLu* fonksiyonu seçildiğinde daha tutarlı sonuçlar elde edilmiştir. Modelde *Tanh (.)* aktivasyon fonksiyonuna nazaran *ReLu* aktivasyon fonksiyonu kullanıldığında verinin daha hızlı işlendiği ve hızlı öğrendiği görülmüştür. Hata fonksiyonu olarak ortalama mutlak hata fonksiyonunn aykırı değerlerin etkisini azalttığı görülmüştür.

Uzun-Kısa Vadeli Bellek (LSTM) modeli için %50,2, Tekrarlayan Sinir Ağı (RNN) modeli için ise %59,8 doğruluk oranı elde edilmiştir. Analiz sürecine geçmeden önce bu çalışmada da yapılan verilerin ön işlem sürecinden geçirilmesinin önemli olduğu düşünülmektedir.

Öğrenme iterasyon değeri arttıkça belirli zaman sonra Uzun-Kısa Vadeli Bellek (LSTM) ve Tekrarlayan Sinir Ağı (RNN) veriyi ezber yöntemiyle tekrarlama yoluna ilerleyecektir. Bu durum etkili tahmin vermemektedir ve sonuç olarak veriyle dengeli öğrenme iterasyon değerinin kullanılmasının daha uygun olduğu düşünülmektedir.

Olağanüstü durumlar gerçekleşmediği sürece herhangi bir finansal varlığın değeri ortalama olarak tahmin edilebilir. Ancak Bist100 ve USD/TRY gibi finans verilerinin günlük değerleri çok hareketli olabilmektedir ve çok küçük durumlardan bile etkilenebilmektedir. Bu çalışmada, hiçbir teknik analiz uygulanmadan veriyi incelediğinde, eğitilmiş modelin yarı yarıya bir tahmin performans gösterdiği görülmüştür. İlerleyen zamanlarda yapılacak

çalışmalarda, tahmin performansının geliştirilmesi için mevcut kullanılan teknik/temel analiz uygulanmış değerleri farklı zaman periyotları ile verdiği sonuçlar incelenebilir veya değişik kombinasyonlar incelenebilir. Bu durumların önüne geçebilmek için kısa dönemli veri kesiti veya öncelikli teknik/temel analiz ile veri üzerinde çalışılmalıdır.

KAYNAKLAR

Almeida, J., Tata, S., Moser, A. and Smit, V. (2015). Bitcoin prediciton using ANN. *Neural Networks*, 7, 1-12.

Alpaydin, E. (2020). *Introduction to machine learning*. New York: MIT Press, 26-32.

Bouktif, S., Fiaz, A., Ouni, A. and Serhani, M. (2018). Optimal deep learning lstm model for electric load forecasting using feature selection and genetic algorithm: Comparison with machine learning approaches. *Energies*, 11(7), 1636.

Buduma, N. (2017). *Fundamentals of deep learning*. London: O'Reilly Media Inc., 85-92.

Cao, Z., T. Simon, S.-E. Wei and Y. Sheikh (2016). Realtime multi-person 2d pose estimation using part affinity fields. *arXiv preprint arXiv*:1611.08050.

Carbonell, J. G., Michalski, R. S. and Mitchell, T. M. (1989). Machine learning: A historical and methodological analysis. *AI Magazine*, 4(3), 70-71.

Chen, Z., Li, C. and Sun, W. (2020). Bitcoin price prediction using machine learning: An approach to sample dimension engineering. *Journal of Computational and Applied Mathematics*, 365(2), 112395.

Cheng, L.C., Huang, Y.H. and Wu, M.E., (2018). *Applied attention-based LSTM neural networks in stock prediction,* IEEE International Conference on Big Data (Big Data), Seattle, 45-52.

Dalyan, T. (2006). *Makine öğrenmesinde 1R algoritması ve ikinci kuralın (2R) oluşturulması*. Yüksek Lisans Tezi, Kocaeli Üniversitesi, Fen Bilimleri Enstitüsü, Kocaeli.

Dickey, D. A. and Fuller, W. A. (1979). Distribution of the estimators for autoregressive time series with a unit root. *Journal of The American Statistical Association*, 74(366a), 427-431.

Dickey, D. A. and Fuller, W. A. (1981). Likelihood ratio statistics for autoregressive time series with a unit root. *The Econometric Society*, 1057-1072

Engle, R. F. and Granger, C. W. (1987). Co-integration and error correction: representation, estimation, and testing. *Econometrica: Journal of The Econometric Society*, 3(6), 251-276.

Engle, R. F. and Yoo, B. S. (1987). Forecasting and testing in co-integrated systems. *Journal of Econometrics*, 35(1), 143-159.

Engle, R. F. and White, H. (1999). *Cointegration, causality, and forecasting: a Festschrift in Honour of Clive WJ Granger*. London: Oxford University Press, 112-126.

Ersin, I. and Eti, S. (2017). Measuring the waste-conscious and saving habits of the youth in Turkey: The sample of Istanbul Medipol University. *International Journal of Islamic Economics and Finance Studies*, 3(3), 41-49.

Fischer, T. and Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2), 654-669.

Ganin, Y., Kononenko, D., Sungatullina, D. and V. Lempitsky (2016). *DeepWarp: Photorealistic image resynthesis for gaze manipulation.* Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, 26-35.

Gholamy, A., Kreinovich, V. and Kosheleva, O. (2018). Why 70/30 or 80/20 relation between training and testing sets: a explanation, *UTEP*, El Paso, 1-7.

Gujarati, D.N. (2011). *Temel Ekonometri* (Çev. Şenesen, Ü. ve Şenesen, G.), İstanbul: Literatür Yayıncılık, 36-42.

Gujarati, D.N. (2012). *Basic Econometrics.* Noida: Tata McGraw-Hill Education, 112-132.

Gunduz, H., Aslan, Y. and Cataltepe, Z. (2017). Intraday prediction of Borsa Istanbul using convolutional neural networks and feature correlations. *Knowledge-Based Systems,* 137, 138-148.

Hinton, G. E., Osindero, S. and Y.-W. Teh (2006). A fast learning algorithm for deep belief nets. *Neural Computation* 18(7), 1527–1554.

Hiransha, M., Gopalakrishnan, E.A., Vijay Krishna Menonab, K.P. Soman, (2018). NSE stock market prediction using deeplearning models, *Procedia Computer Science*, 132,1351–1362

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

Hwang, J. and Zhou, Y. (2016). Image colorization with deep convolutional neural networks, *Stanford University*, Stanford, 1-7.

Isola, P., Zhu, J.-Y., Zhou, T. and Efros, A.A. (2016). Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv*, 2(3), 12-18.

İnternet: Finance Yahoo (2022). Web: https://finance.yahoo.com/quote/TRY=X?p=TRY=X&.tsrc=fin-srch, Son Erişim Tarihi:20.05.2022

İnternet: Finance Yahoo (2022). Web: https://finance.yahoo.com/quote/XU100.IS?p=XU100.IS&.tsrc=fin-srch Son Erişim Tarihi:20.05.2022

İnternet: Machine Learning Mastery (2023). Web: https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/, Son Erişim Tarihi:12.02.2023

İnternet: Machine Learning Mastery (2023). Web: https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/, Son Erişim Tarihi:15.02.2023

İnternet: Nilson, N. J. (1996). *Introduction to machine learning an early draft of a proposed textbook.* Web: https://ai.stanford.edu/~nilsson/MLBOOK.pdf, Son Erişim Tarihi: 26/08/2022.

İnternet: Stack Overflow (2023). Web: https://stackoverflow.com/questions/49497930/categorical-variable-for-time-series-prediction-with-lstm-and-keras, Son Erişim Tarihi:18.02.2023

İnternet: Towards Data Science (2023). Web: https://towardsdatascience.com/multi-state-lstms-for-categorical-features-66cc974df1dc, Son Erişim Tarihi:18.02.2023

Kiros, R., Salakhutdinov, R. and Zemel, R. S. (2014). Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv*, 2(4), 26-32.

Kumar S., Ningombam D. (2018). *Short-Term Forecasting of stock prices using long short term memory.* 2018 International Conference on Information Technology (ICIT), Bhubaneswar, India, 182-186.

Lachiheb, O. and Gouider, M. S. (2018). A hierarchical deep neural network design for stock returns prediction. *Procedia Computer Science*, 126(7), 264-272.

LeCun, Y., Bengio, Y. and Hinton, G. (2015). Deep learning. *Nature* 521(7553), 436-444.

Maimonand, O. and Rokach, L., (2005). *Data mining and knowledge discovery handbook.* New York: Springer, 112-133.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4), 115–133.

Michalski, R. S., Carbonell, J. G. and Learning, T. M. M. M. (1986). An artificial intelligence approach. *Understanding the Nature of Learning*, 2, 3-26.

Michalski, R. S., Carbonell, J. G. and Mitchell, T. M. (1986). *Machine learning: an artificial ıntelligence approach.* New York: Springer, 26-32.

Navon, A. and Keller, Y. (2017). Financial time series prediction using deep learning. *arXiv,* 3(5), 1711.

Olvera-Juarez, D. and Huerta-Manzanilla, E. (2019). Forecasting bitcoin pricing with hybrid models: A review of the literature, *International Journal of Advanced Engineering Research and Science*, 6(9), 161-164.

Öğücü, M. O. (2006). *Yapay sinir ağları ile sistem tanıma.* Doktora Tezi, İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü, İstanbul, 122-132.

Pal, K. K. and Sudeep, K. S. (2016). *Preprocessing for image classification by convolutional neural networks.* 2016 IEEE International Conference on Recent Trends in Electronics, Bangalore, 1778–1781

Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088), 533–536.

Rumelhart, D. E., McClelland, J. L. and Group, T. P. R. (1986). *Parallel distributed processing: explorations in the microstructure of cognition.* Cambridge: MIT Press, 17-20.

Sagheer, A. and Kotb, M. (2019). Time series forecasting of petroleum production using deep LSTM recurrent networks. *Neurocomputing*, 323, 203-213.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. II—Recent progress. *IBM Journal of Research and Development*, 11(6), 601-617.

Sezer, O. B. and Ozbayoglu, A. M. (2018). Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach. *Applied Soft Computing*, 70(5), 525-538.

Shao, X., Ma, D., Liu, Y. and Yin, Q. (2017). *Short-term forecast of stock price of multi-branch LSTM based on K-means.* 4th International Conference on Systems and Informatics (ICSAI), Hangzhou, 1546-1551.

Tarı, R. (2015). *Ekonometri.* Kocaeli: Umuttepe Yayınları, Kocaeli, 8-15.

Wei, W. W. (2006). Time series analysis. In *The oxford handbook of quantitative methods in psychology.* London: Oxford University Press, 26-32.

EKLER

EK-1. R kodu: Verilerin standartlaştırılması, ADF ve EG test istatistiğinin hesaplanması

```
as.factor(Bist100$Close)
as.factor(DolarKuru$Close)
minbistclose=min(Bist100$Close, na.rm = T)
maxbistclose=max(Bist100$Close, na.rm = T)
mindolarclose=min(DolarKuru$Close, na.rm = T)
maxdolarclose=max(DolarKuru$Close, na.rm = T)
stbistclose=(Bist100$Close-minbistclose)/(maxbistclose-minbistclose)
stdolarclose=(DolarKuru$Close-mindolarclose)/(maxdolarclose-mindolarclose)
plot(stbistclose,type = "l", ylab="Close (Normalised between 0-1)")
plot(stdolarclose,type = "l", ylab="Close (Normalised between 0-1)")
close=na.omit(Bist100$Close)
adf.test(close)
dclose=na.omit(DolarKuru$Close)
adf.test(dclose)
coint.test(DolarKuru$Close,Bist100$Close)
sample=sample.split(Bist100$`Close*`, SplitRatio = 0.8)
trainbist100close=subset(Bist100$`Close*` ,sample==TRUE)
testbist100close=subset(Bist100$`Close*` ,sample==FALSE)
dim(trainbist100close)
dim(testbist100close)
sample=sample.split(DolarKuru$`Close*`, SplitRatio = 0.8)
traindolarclose=subset(DolarKuru$`Close*`, sample==TRUE)
testdolarclose=subset(DolarKuru$`Close*`, sample==FALSE)
dim(traindolarclose)
dim(testdolarclose)
```

EK-2. R kodu: Tekrarlayan Sinir Ağı model elde etme ve görselleştirme

```
data_dir <- "~/Downloads/veri"
fname <- file.path(data_dir, “veri.csv")
raw_data <- veri
glimpse(new_data)
ggplot(new_data, aes(x = 1:nrow(raw_data), y = `T (degC)`)) + geom_line()
ggplot(new_data[1:1244 ], aes(x = 1:1244, y = 'tarih')) + geom_line()
as.data.frame(new_data)
data <- data.matrix(new_data[, -1])
mean <- apply(newdata, 2, mean)
std <- apply(newdata, 2, sd)
data <- scale(newdata, center = mean, scale = std)
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}
max <- apply(newdata,2,max)
min <- apply(newdata,2,min)
data <- apply(newdata, 2, normalize)
plot(data[1:1244,1 ], ylab = "USD/TRY (Standartlaştırılmış)")
plot(data[1:1244,2 ], ylab = "Bist100 (Standartlaştırılmış)")
dotchart(data[1:1244,1],labels = "Date" ,bg="blue", pt.cex = 1.5)
dotchart(data[1:1244,2], labels = "Date", bg="red", pt.cex = 1.5)
generator <- function(newdata,  lookback,delay,min_index,   max_index,  shuffle = FALSE,   batch_size = 128,step = 6) {
  if (is.null(max_index)) max_index <- nrow(data) - delay - 1
  i <- min_index + lookback
  function() {
    if (shuffle) { rows <- sample(c((min_index + lookback):max_index), size = batch_size)  }
else
{
if (i + batch_size >= max_index)
  i <<- index + lookback rows <- c(min(i + size - 1, index))i <<- i + length(rows) }
    samples <- array(0, dim = c(length(rows),lookback / step  (data)[[-1]]))
    targets <- array(0, dim = c(length(rows)))
    for (j in 1:length(rows)) {
      indices <- seq(rows[[j]] -, rows[[j]] - 1,
                     length.out = dim(samples)[[2]])
      samples[j, , ] <- data[indices, ]
      targets[[j]] <- data[rows[[j]] + delay, 2]
lookback <- 1440
step <- 6
delay <- 44
batch_size <- 64
train_gen <- generator(data, lookback = lookback,delay = delay _index = 1,_index = 248,shuffle = FALSE, step = step,
  batch_ = _size)
lookback <- 240
step <- 1
delay <- 44
batch_ <- 64
train_gen <- generator(newdata,lookback = lookback,delay = delay index = 1,max_index = 248,shuffle = FALSE,step = step,batch
= batch)
train _data <- train_gen()
model <- keras_model_sequential() %>%
  layer_flatten(input = c(lookback / step, dim(data)[-1])) %>%
  layer_dense(units = 64, activation = "relu") %>%
  layer_dense(units = 32, activation = "relu") %>%
  layer_dense(units = 1)
summary(model)
model %>% compile(optimizer = optimizer loss = "mae")
history <- model %>% fit(train_gen_data[[1]],train_gen_data[[2]],batch = 32,epochs = 10,use_multiprocessing = T)
batch_size <- 1244
```

EK-2. (devam) R kodu: Tekrarlayan Sinir Ağı model elde etme ve görselleştirme

```
lookback_plot <- lookback
step_plot <- 1
pred_gen <- generator(lookback = lookback_plot, = 0,min_index = 249,max_index = 1244,shuffle = FALSE,step = step_plot,batch
= batch_plot)
pred_data <- pred_gen()
V1 = seq(1, length(pred_data[[2]]))
plot_data <-data.frame(cbind(V1, pred _data[[2]]))
input data <- data[[1]]
dim(data) <- c(batch_size_plot, lookback, 14)
pred_out <- model %>%predict(data)
plot_data <-cbind(data, pred_out)
p <-ggplot(plot_data, aes(x = V1, y = V2)) + geom_point(colour = "blue", size = 1.5,alpha=0.4)
p <- p + geom_point(aes(x = V1, y = pred_out), colour = "red", size = 1.5,alpha=0.4)
p
```

EK-3. R kodu: Uzun-Kısa Vadeli Bellek model elde etme ve görselleştirme (ReLu)

```
model <- keras_model_sequential() %>%
  layer_lstm(units = 64, input_shape = c(240, 1)) %>%
  layer_dense(units = 64, activation = "relu") %>%
  layer_dense(units = 32, activation = "relu") %>%
  layer_dense(units = 1, activation = "relu")
summary(model)
model %>% compile(loss = 'mean _error', optimizer = 'adam',metrics='mse')
history <-  model %>% fit (x,    y,    batch = 100,    epochs = 5,    validation = 0.1,    multiprocessing = T  )
plot(history)
batch_plot <- 1244
look back <- 240
step <- 1
pred_gen <- generator(  data,  lookback = lookback,  delay = 0,  index = 249, index = 1244,  shuffle = FALSE,  step = step_plot,
batch = batch_size)
pred_data <- pred_gen()
V1 = seq(1, length(pred_data[[2]]))
plot_data <-data.frame(cbind(V1, pred_data[[2]]))
input <- pred_data[[1]][,,2]
dim(input) <- c(batch_size_plot,lookback, 1)
pred_out <- model %>% predict(input)
plot_data <- cbind pred_out)
p <-ggplot(plot_data, aes(x = V1, y = V2)) + geom_point( colour = "blue", size = 1.5,alpha=0.4)
p <-  p + geom_point(aes(x = V1, y = pred_out), colour = "red", size = 1.5 ,alpha=0.4)
p
lookback <- 240
step <- 1
delay <- 44
batch <- 128
train_gen <- generator( data,  lookback = lookback,  min_ = 1,  max_ = 248,  shuffle = FALSE,  batch = batch_size)
train_gen <- train_gen()
model <- keras_model_sequential() %>%
  layer_lstm(units = 64, input_shape = list(NULL, dim(data)[[-1]])) %>%
  layer_dense(units = 64, activation = "relu") %>%
  layer_dense(units = 32, activation = "relu") %>%
layer_dense(units = 1, activation = "relu")
model %>% compile(optimizer = optimizer, loss = "mae")
summary(model)
history <- model %>% fit(  train_gen_data[[1]],train_gen_data[[2]],  batch = 4,  step_per_epoch = 1,  epochs = 10, )
plot(history)
batch_size = 1244
lookback <- 240
step <- 1
pred_gen <- generator(  data,  lookback = lookback_,  delay = 0,  min_ = 248,  max_ = 1244,  shuffle = FALSE,  step = step_plot,
batch = batch_size_)
pred_gen_data <- pred_gen()
V1 = seq (1, length(pred _data[[2]]))
Plot data <-  data.frame(cbind(V1, pred data[[2]]))
data <- pred_data[[1]][,,]
dim(data) <- c(batch_size_plot,lookback_plot, 14)
pred_out <- model %>%  predict(data)
plot_data <- cbind(plot_data, pred_out[])
p <- ggplot(plot_data, aes(x = V1, y = V2)) + geom_point( colour = "blue", size = 1.5,alpha=0.4)
p <-p + geom_point(aes(x = V1, y = pred_out), colour = "red", size = 1.5 ,alpha=0.4)
p
```

EK-4. R kodu: Uzun-Kısa Vadeli Bellek model elde etme ve görselleştirme (Tanh)

```
lookback <- 28 step <- 1
batch <- 1244
train_gen <- generator(data,    delay = delay,min_ = 1,  max_ = 248,  shuffle = FALSE,  batch_size = batch_size)
val_gen <- generator(  data,  lookback = lookback  min_ = 248,  max_ = 1244,  shuffle = FALSE, )
train _data <- train_gen()
val _data <- val_gen()
model <- keras_model_sequential() %>%
  layer_gru(units = 64, return_sequences = TRUE,input_shape = list(NULL, dim(data)[[-1]])) %>%
  bidirectional(layer_gru(units = 64)) %>%
  layer_dense(units = 64, activation = "tanh") %>%
  layer_dense(units = 32, activation = "tanh") %>%
  layer_dense(units = 1, activation = "tanh")
model %>% compile(optimizer = optimizer,  loss = "mae")
summary(model)
callbacks = callback_early_stopping(monitor = "val_loss", min_delta = 0,patience = 10, verbose = 0, mode = "auto",   baseline =
NULL, restore_best_weights = TRUE)
history <- model %>% fit(   train data[[1]],train_data[[2]],   batch = 1244,   epochs = 10,   validation_data = val_gen_data,
validation_steps = 5)
batch_size <- 1244
lookback <- 15
step<- 1
pred_gen <- generator(  data,  lookback = lookback_,  delay = 0,  min_ = 1,  max_ = 995,  shuffle = FALSE,  step = step_plot,
batch = batch_size_plot)
pred _data <- pred_gen()
V1 = seq(1, length(pred_gen_data[[2]]))
plot_data <-.data.frame(cbind(V1, pred _data[[2]]))
data <- pred _data[[1]][, , ]
dim(data) <- c(batch_size_plot, lookback, 2)
pred_out <- model %>%  predict(data)
plot_data <-cbind(plot_data, pred_out[])
p <- ggplot(plot_data, aes(x = V1, y = V2)) + geom_point(colour = "blue",size = 0.1, alpha = 0.4)
p <-  p + geom_point(aes(x = V1, y = pred_out), colour = "red", size = 0.1 , alpha = 0.4 )
```

Gazili olmak ayrıcalıktır…