ÇUKUROVA ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

Derin Öğrenmede Tekrarlayan Sinir Ağları Yaklaşımı ile Hisse Senedi Fiyat Tahmini

Güldeniz CANATAN

İstatistik Anabilim Dalı

Eylül, 2023

İÇİNDEKİLER

ÇUKUROVA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

Derin Öğrenmede Tekrarlayan Sinir Ağları Yaklaşımı ile Hisse Senedi Fiyat Tahmini

Güldeniz CANATAN

Danışman: Prof. Dr. Güzin YÜKSEL

İstatistik Anabilim Dalı

ÖZ

Zaman serisi tahmini, geçmiş veri noktalarını zaman sırasına göre düzenleyerek gelecekteki olaylar veya değerler hakkında tahminler yapmayı amaçlayan bir analiz ve modelleme yaklaşımıdır. Bu yöntem, istatistik, ekonometri, finans, meteoroloji gibi birçok alanda kullanılır. Son yıllarda, makine öğrenmesi ve derin öğrenme gibi yeni metodolojilerin yükselişi ile sinir ağları, finansal tahminlerde önemli bir araç haline gelmiştir. Bu çalışma, finansal hisse senedi verilerini tahmin etmek amacıyla tekrarlayan sinir ağı mimarilerini kullanmaktadır. Sinir ağları, karmaşık ve doğrusal olmayan ilişkileri yakalama yeteneği ile öne çıkar ve eksik verileri işleme yeteneği sayesinde bazı avantajlar sunar. Bu çalışma, sinir ağlarıyla kurulan modellerin klasik stokastik zaman serisi yöntemleri ile karşılaştırılmasına odaklanır ve sinir ağlarının finansal tahminlerdeki potansiyel üstünlüklerine vurgu yapar. Sinir ağları, geleneksel yöntemlerin ötesine geçerek, finansal piyasa verilerinin karmaşıklığını ve dinamiklerini daha iyi anlama ve daha doğru tahminler yapma yeteneği sunabilir. Bu çalışma, finansal analistler ve araştırmacılar için yeni bir bakış açısı sunarak, sinir ağlarının finansal tahminlerde nasıl kullanılabileceğini araştırır.

Anahtar Kelimeler: Zaman Serisi Tahmini, Tekrarlayan Sinir Ağları, Deri Öğrenme.

CUKUROVA UNIVERSITY

INSTITUTE OF NATURAL AND APPLIED SCIENCES

MSc THESIS

Stock Price Prediction with Recurrent Neural Networks Approach in Deep Learning

Güldeniz CANATAN

Advisor: Prof. Dr. Güzin YÜKSEL

Department of Statistics

ABSTRACT

Time series forecasting is an analysis and modeling approach that aims to make predictions about future events or values by arranging historical data points in time order. This method is used in many fields such as statistics, econometrics, finance, and meteorology. In recent years, with the rise of new methodologies such as machine learning and deep learning, neural networks have become an important tool in financial forecasting. This study uses recurrent neural network architectures to predict financial stock data. Neural networks stand out for their ability to capture complex and non-linear relationships and offer some advantages thanks to their ability to handle incomplete data. This study focuses on the comparison of models built with neural networks with classical stochastic time series methods and emphasizes the potential advantages of neural networks in financial forecasting. Going beyond traditional methods, neural networks can offer the ability to better understand the complexity and dynamics of financial market data and make more accurate predictions. This study explores how neural networks can be used in financial forecasting, offering a new perspective for financial analysts and researchers.

Keywords: Time Series Forecasting, Recurrent Neural Networks, Deep Learning.

TEŞEKKÜR

Danışmanım Prof. Dr. Güzin Yüksel'e, araştırma konumu seçmekten başlayarak tüm süreç boyunca bana sunduğu özgürlük ve rehberlik için içtenlikle teşekkür ediyorum. Kendisinin sağladığı eşsiz rehberlik ve kararlı destek, araştırmamı başarıyla tamamlamama büyük katkı sağlamıştır. Akademik gelişimime verdiği değer ve özverili çalışması için derin bir minnettarlık duyuyorum.

ÇİZELGELER DİZİNİ

ŞEKİLLER DİZİNİ

SİMGELER VE KISALTMALAR

ACF : Otokorelasyon Fonksiyonu
AR : Otoregresyon
ARIMA : Entegre Otoregresif Hareketli Ortalama
ARMA : Otoregresif Hareketli Ortalama
BRNN : Çift Yönlü Tekrarlayan Sinir Ağı
DRNN : Derin Tekrarlayan Sinir Ağı
GRU : Geçitli Tekrarlayan Birim
LSTM : Uzun-Kısa Vadeli Bellek
MA : Hareketli Ortalama
MAE : Ortalama Mutlak Hata
MSE : Hata Kareler Ortalaması
PACF : Kısmi Otokorelasyon Fonksiyonu
RMSE : Hata Kareler Ortalaması Karekökü
RNN : Tekrarlayan Sinir Ağı
SARIMA : Mevsimsel Entegre Otoregresif Hareketli Ortalama

1. GİRİŞ

Günümüzde sıkça rastlanan Fintech ifadesi dijital dönüşümün finans sektöründeki yerini temsil etmektedir. Finans endüstrisi küresel oyuncuların hakimiyetine girmesiyle teknoloji kullanımı elzem hale gelmiş ve finansal teknoloji iş modeli ve süreçleri köklü bir değişime uğramıştır. Fintech süreçlerinin bir aşaması sayılabilecek yapay zeka uygulamaları da literatürde önemli bir konuma sahiptir. Finansal verilerle regresyon, sınıflandırma ve kümeleme gibi iş problemlerini çözülebilmekte, makine öğrenmesi modelleri yaygın olarak kullanılmaktadır. Zamana bağlı olarak gözlemlenen veri noktalarını oluşturan zaman serileri de makine öğrenmesi dünyasında sıkça kullanılan bir uygulama alanıdır. En basit tabirle zamansal boyutta tanımlanan geçmiş veriler kullanılarak geleceği tahmin etme işlemi zaman serisi tahmini olarak anılmaktadır.

Zaman serisi tahmininin finans alanındaki en yaygın uygulamalarından biri hisse senedi tahminidir. Finans sektöründe yapılan analizler ve tahminler yatırım kararları vermek, portföy optimizasyonu sağlamak ve finansal stratejileri belirleme konusunda büyük önem arzetmektedir. Bu aşamada bir kuruluşa ait borsa değerinin ve hisse senedinin tahmini, çok yönlü bir fayda sağlayarak şirketin kârlılık ve üretkenlik konularında oldukça sağlam bir zemin oluşturulmasına imkan tanımaktadır. Finansal verilerin doğru analizi ve tahmini bireysel yatırımcılar için de son derece önemlidir. Finans piyasasının hareket yönü analiz edilerek yatırım stratejisi doğru bir şekilde planlanmaktadır. Tüm bu kazanımlar ışığında finansal piyasanın zamana bağlı hareket eğilimini tespit etmek büyük önem barındırmaktadır.

Zaman serisi tahmininde yaygın olarak verinin barındırdığı trend, mevsimsellik, durağanlık gibi bileşenlere göre istatistiksel-ekonometrik modeller olarak tabir edilen AR, MA, ARIMA, SARIMA vb. modeller kullanılmaktadır. Ayrıca zaman serisi analizi tahmin edilmek istenen değişkeni etkileyen parametre sayısına göre tek değişkenli ve çok değişkenli şeklinde ele alınmaktadır.

Yapay zeka algoritmalarının optimize edilmesi ve spesifik problemlere uyarlanmasıyla kullanım sıklığı her zamankinden daha yüksek bir seviyeye ulaşmıştır. Özellikle tekrarlayan sinir ağı(RNN) ve türevleri gibi sinir ağı mimarileri, sıralı dizilerde etkin bir tahmin süreci sağlamaktadır.

Bu çalışmanın amacı Anadolu Anonim Türk Sigorta Şirketine ait hisse senedi verileri istatistiksel ekonometrik modeller ve tekrarlayan sinir ağı mimarileri kullanılarak analiz etmektir. Elde edilen çıktılar üzerinden performans değerlendirmesi yapılmıştır. Özellikle veri boyutu, veri türü, doğrusallık gibi ölçütler gözetilerek sinir ağlarının etkinliği incelenmiştir. Zaman serisi tahmininde klasik yöntemlere alternatif olarak tekrarlayan sinir ağlarının kullanılmasının model performansı başta olmak üzere bir çok yönden faydalı olacağı öngörülmektedir.

Bu çalışma derin sinir ağlarının finansal zaman serisi verileri üzerindeki etkinliklerini incelemek için büyük önem arzetmektedir. Klasik istatistiksel zaman serisi metodolojilerinin sinir ağlarıyla karşılaştırılarak analize dahil edilmesi yatırımcı ve kuruluşların önemli kararlar almasını

kolaylaştıracaktır. Kurulan modellerle başarılı sonuçların elde edilmesi şirketlerin portföy optimizasyonu, yıllık bütçe çalışmaları, yatırım gelirleri gibi bir çok alana dokunan sorunlarını çözümlemeye yardımcı olmaktadır. Aynı zamanda bireysel yatırımcıların da kârlılık süreçlerinde kullanabileceği yeni yöntemler olarak değerlendirilecektir..

Zaman serisi tahmini bir çok parametre baz alınarak yapıldığı için kullanılan metodolojilerin karmaşıklığı oldukça fazladır. Verinin doğrusal olup olmama durumu, yapısı, tek veya çok değişkenden meydana gelmesi gibi özellikler, kullanılacak metodolojiyi şekillendirmektedir. Ayrıca sinir ağlarının barındırdığı gizli katmanlar sebebiyle kurulan modelin açıklanabilirliliği zordur. Bununla birlikte sinir ağları aşırı öğrenmeye eğilimlidir ve daha büyük veri setlerinde daha yüksek performans gösterir. Özetle klasik metotlar ve sinir ağı mimarileri kullanılırken bu zorluklarla karşılaşılmış, sınırlayıcı olarak değerlendirilen durumlar farklı yöntemlerle çözülmeye çalışılmıştır.

Tez çalışması 7 bölümden oluşmaktadır. İlk bölümde finans sektöründe zaman serisi tahmininin önemi açıklanmış, çalışılan konu üzerinde detaylı bilgi verilmiş, çalışmanın amacına değinilmiştir. 2. bölümde literatürde yer alan tekrarlayan sinir ağı mimarilerinin ve geleneksel yöntemlerin kullanıldığı araştırmalara yer verilmiştir. 3. bölüm klasik zaman serisi yöntemleri bileşenleri ve modelleme konusunda detaylı bir öngörü sağlamaktadır. 4. bölümde yapay sinir ağları detaylı bir şekilde açıklanmış, öğrenme işleminin nasıl gerçekleştiği ve sinir ağı mimarisinin inşa edilmesi hususunda bir çerçeve oluşturulmuştur. 5. bölümde sinir ağlarının bir türü olan RNN'ler ve farklı versiyonları tanıtılmış, klasik yapay sinir ağlarından farkı belirtilmiştir. 6. bölüm bir çok modelle kurulan analizleri içermektedir. Bu bölümde Pyhton yardımıyla elde edilen geleneksel tahmin yöntemleri ve tekrarlayan sinir ağı çıktıları incelenmiş ve yorumlanmıştır. Son olarak 7. Bölümde, elde edilen bulgular ışığında görüşler belirtilmiş, çalışmanın amacına hizmet eden sonuçlar aktarılmıştır.

2. ÖNCEKİ ÇALIŞMALAR

Hisse senedi fiyat tahminleri, borsalarda işlem gören şirketlerin yatırımcıları üzerinde doğrudan etkisi olduğu için uzun bir geçmişe dayanan ve halen büyük bir ilgi gören bir konudur. İyi bir tahmin, yatırımcıların doğru kararlar vermesine, portföy yönetimi açısından risk ve getiri dengesini sağlamaya, şirketin değerinin belirlenmesine ve yeni iş stratejilerinin doğru bir şekilde geliştirilmesine yardımcı olur.

Zamana bağlı gözlemlerin geleceğe yönelik tahmini konusu ilk olarak Yule ve Slutsky'nin 1927'de yaptığı çalışmalarla gündem olmuştur. Slutsky (1927) döngüsel olaylardan söz ederek kendinden sonraki iktisatçıları etkileyerek konjonktür teorisine katkı sağlamıştır. Yine aynı tarihte Yule'un (1927) çalışmaları otoregresif modellerin tahmin ve analizinde kullanılan bir çok yöntem geliştirmiştir. Zaman serilerinin yanısıra ekonometrik teorilerin oluşturulmasında da bir çok önemli gelişme yaşanmıştır. Frisch'in (1933) makalesi ekonometrik verilerin analizi konusunda istatistiksel, olasılıksal alt yapının kurulmasına yardımcı olmuş, ekonomik değişkenlerin ilişkilendirilmesi konusunun gündem olmasına neden olmuştur. Kolmogorov (1940) olasılık teorisiyle ilgili yaptığı çalışmalarda zaman serilerinin matematik ve istatistiksel bağımsızlık temelinde ele alınmasını sağlamıştır. Box ve Jenkins (1970) zaman serisi analizine detaylı bir şekilde odaklanmış, entegre otoregresif hareketli ortalama (ARIMA) ile tahmin yöntemlerini açıklamıştır.

Geleneksel zaman serisi tahminindeki gelişmelere paralel olarak yapay sinir ağları ve zamana bağlı gözlemlerde kullanımı da oldukça popüler hale gelmiştir. Özellikle gerçek hayat senaryolarının doğrusallık göstermemesi farklı metodolojilerin incelenmesine zemin hazırlamıştır. Sinir ağları bileşenlerinden biri olan gizli katmanlar doğrusal olmayan verilerin de analizine imkan sağlamaktadır. Lapedes ve Farmer (1987) doğrusal olmayan verilerde yapay sinir ağı kullanımına ilişkin detayları makalelerinde belirtmişlerdir. Bu çalışmalar zamanla genişletilmiş Tang ve ark (1991) ile Sharda ve Patil (1992) gibi bir çok araştırmacı bu konuya yoğunlaşmıştır. Geleneksel yöntemlerle sinir ağı tabanlı bir çok yöntemin karşılaştırmalı analizi yine yakın zamanda gerçekleşmiştir. Marquez ve ark (1992) ile Nelson ve ark (1994) bunlara örnektir.

Dizi verilerinde sıklıkla kullanılan tekrarlayan sinir ağı mimarileri hisse senedi tahmininde de oldukça yaygın kullanılmıştır. Hochreiter ve Schmidhuber(1997), tekrarlayan sinir ağlarınının bir versiyonu olan uzun kısa vadeli bellek (LSTM) ile tahmin yöntemlerinin nasıl geliştireceği konusunda detaylı bir çerçeve oluşturmuştur. Jia (2016), LSTM mimarisinin etkinliğini bir hisse senedi fiyatını tahmin ederek ortaya koymaya çalışmıştır. Tsantekidis ve ark (2017) Twitter üzerinden duygu analizi üzerine yoğunlaşarak hisse senedinin fiyat değişimlerini bir sinir ağı modeline entegre edilerek yeni bir tahmin yöntemi geliştirmiştir. Zhang ve Liu (2017), çok katmanlı algılayıcılarla klasik tahmin yöntemlerine alternatif olarak finansal haberlerin hisse senedi üzerine etkilerini araştırmış ve yeni yöntemler geliştirmiştir. Zhao ve ark (2017) LSTM mimarisini trafik tahmininde kullanarak model avantajlarına değinmişlerdir. Bouktif ve ark (2018) karşılaştırmalı bir

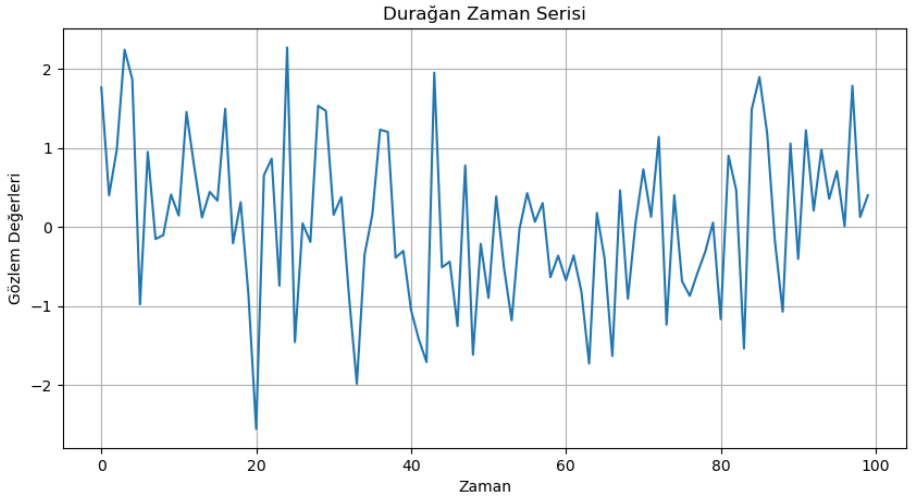
çalışma yaparak LSTM mimarisinin başarısını sınamıştır. Sonrasında bu bağlamda yapılan çalışmalar genişletilmiş klasik zaman serisi analizi yöntemleri ve tekrarlayan sinir ağlarını karşılaştıran çalışmalar yapılmıştır(Siami-Namini ve ark, 2019).

3. ZAMAN SERİSİ ANALİZİ

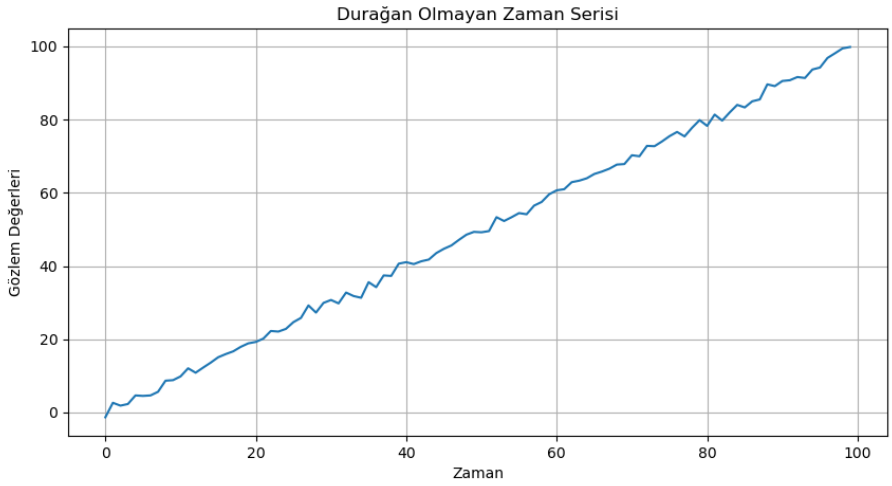
Zaman serisi, zamana göre sıralanmış gözlem değerlerinden oluşan verilerdir. Gözlem değerleri doğası gereği birbiriyle ilişkilidir. Bir hisse senedinin zamana göre değişimi bilgilerini içeren veriler zaman serisi olarak nitelendirilebilir. Zaman serileri incelenerek gereceğe yönelik tahminler yapılabilmektedir. Zaman serisi analizini bazı istatistiksel ve ekonometrik metodolojiler oluşturmaktadır.

Bir zaman serisini incelenirken durağanlık, mevsimsellik, trend, döngü gibi çeşitli kavramlar mevcuttur ve bu kavramların varlığı yapılacak analizin temelini şekillendirmektedir.

Durağanlık bir zaman serisinin çeşitli istatistiksel özelliklerinin zaman içerisinde değişmemesi olarak tanımlanmaktadır. Serinin ortalaması, varyansı ve kovaryansı zaman içinde sabit ise bu seri için durağandır diyebiliriz. Durağanlık istatistiksel testler ve grafik metodlarıyla tespit edilebilmektedir (Yaffee ve McGee, 2000).



Şekil 3.1. Durağan Zaman Serisi

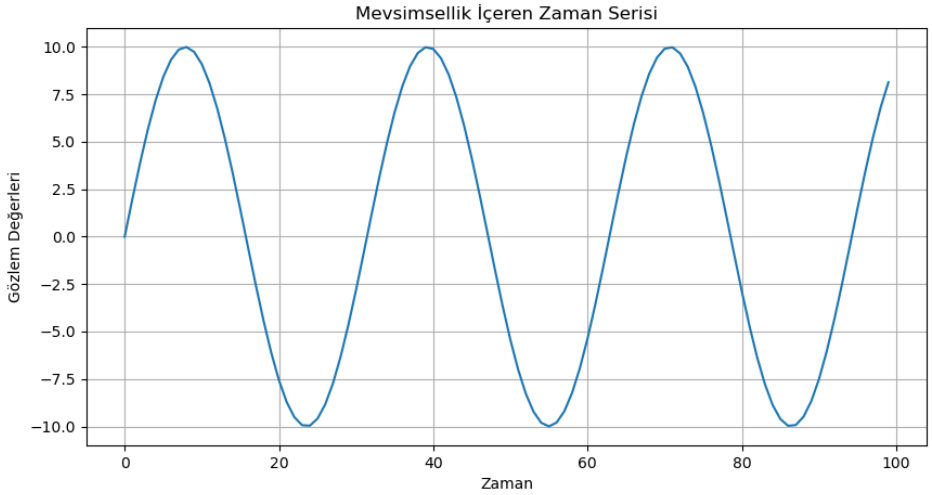


Şekil 3.2. Durağan Olmayan Zaman Serisi

Bir serinin durağan olması yapılacak tahmin için bir varsayım olarak nitelendirilmektedir. Dickey-Fuller Testi ile durağanlık varsayımı kolayca tespit edilebilmektedir. Ancak gerçek hayat problemleri genellikle durağan olmayan serilerden oluşmaktadır. Durağan olmayan serilerde yapılan tahmin sonuçları şüpheli olarak değerlendirilir. Bu yüzden durağan olmama problemini ortadan kaldıracak birkaç yöntem geliştirilmiştir. Bunlardan bazıları fark alma, trend ekleme ve logaritmik dönüşümdür.

Zaman serisi analizinde incelenen bir diğer kavram olan mevsimsellik bir seride tekrarlanan periyodik dalgalanmalar olarak tanımlanabilir. Belli frekanslarda belli dönemlerde tekrarlanan hareketlere sahip serilere örnek olarak turizm verileri verilebilir. Sıcak bir sahil şehrinde yaz ayları itibariyle deniz tatili tercih eden turistlerin bölgedeki nüfusu arttırması her sene periyodik olarak tekrarlanacak ve aynı zaman dilimine denk gelecektir. Bu şekilde bir zaman serisindeki mevsimselliği tespit etmek tahmin başarısının doğruluğu açısından oldukça önemlidir.

Bir seride mevsimsellik olup olmadığı bir çok yöntemle tespit edilebilir. Zaman serisine ait zamana bağlı gözlemlerin olduğu grafik incelenerek belli periyotlarda benzer eğilimlerin tekrarlanması mevsimsellik olarak değerlendirilir.



Şekil 3.3. Mevsimsellik Barındıran Zaman Serisi

Otokorelasyon ve Kısmi Otokorelasyon(ACF ve PCAF) grafiklerinde de belli periyotlarda yükselmeler mevcutsa yine mevsimselliğin varlığından söz edilebilir.

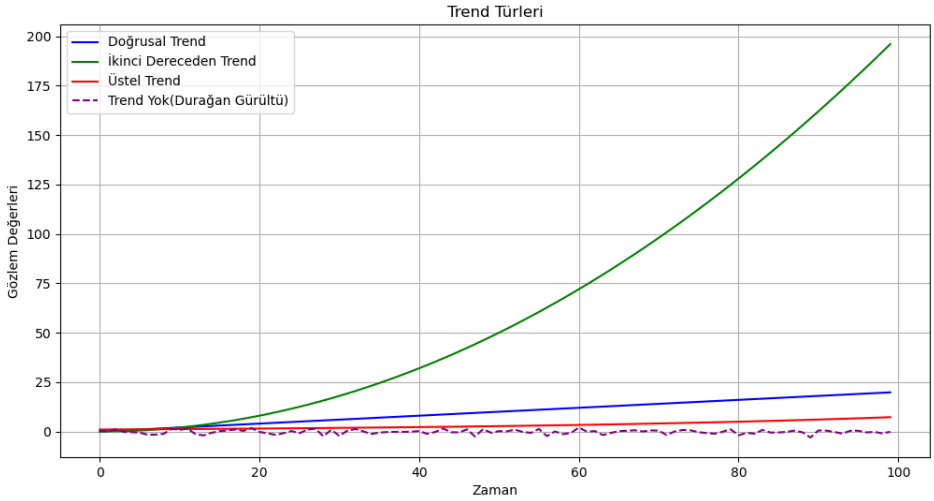
Zaman serisiyle regresyon modeli kurulurken ay yıl hafta gibi zaman periyotlarının türetilerek modele değişken olarak dahil edilmesi de verilerin görselleştirilmesiyle mevsimsel etkinin tespit edilmesine yardımcı olacaktır.

Bunlara ek olarak serideki durağanlığın tespit edilmesi de mevsimsellik hakkında bilgi sahibi olmaya yarayabilir. Mevsimselliğin olduğu bir serinin durağan olmadığı varsayımıyla grafik incelemesi yapılabilmektedir.

Bu yöntemlerin dışında çeşitli istatistiksel testlerle de mevsimsellik tespit edilebilir. Özellikle grafik yöntemler incelenirken herhangi bir tereddüt yaşanması ya da emin olmama gibi durumlarda istatistiksel testler tavsiye edilmektedir. Kruskal – Wallis testi kullanılarak periyotlar arası fark olup olmama durumu üzerinden hipotez kurulur.

Zaman serisi incelenirken kullanılan trend, bir zaman serisinin genel eğilimidir. Seri periyodik olarak artıyor ya da azalıyorsa trendin varlığından söz edilebilir. Trend analizi verinin genel yönelimini tespit etme ve doğru bir modelleme çalışması yapma açısından önemlidir. Verideki artış ve azalışlar, geleceğe yönelik tahmin, anomali tespiti ve performans değerlendirmesi trendin analizi sayesinde yapılabilir.

Zaman serisi bileşenlerinden trendin tespiti bir çok yöntemle yapılabilmektedir. Grafik inceleme ile serinin artış ya da azalış eğilimi gözlemlenebilir. Ayrıca R ve Python kütüphaneleriyle de trendin türü hakkında bilgi edinilebilmektedir.



Şekil 3.4. Trend Türleri

Yukarıdaki grafikteki renklerden kırmızı üstel, yeşil ikinci dereceden, mavi doğrusal trendi temsil etmektedir. Mor temsil edilen seride ise trendin varlığından söz edilememektedir.

Diğer bir özellik de mevsimsel dalgalanmalara benzer bir davranış gösteren döngü bileşenidir. Daha büyük periyotlarda gerçekleşen dalgalanmalar olarak tanımlanabilir. Birkaç yılda bir gerçekleşen doğal afet, salgın hastalık gibi olaylar döngüsel davranışa örnek gösterilebilir.

3.1. Zaman Serisinde Ayrıştırma

Zaman serilerinde ayrıştırma, modelin daha iyi bir başarı göstermesi için kullanılmaktadır. Toplamsal ve çarpımsal model olarak adlandırılan ayrıştırma yöntemleri, serideki genel eğilimi tespit etmek amacıyla zaman serisinin bileşenlerinin ayrıştırılmasıyla gerçekleştirilir (Brownlee, 2020).

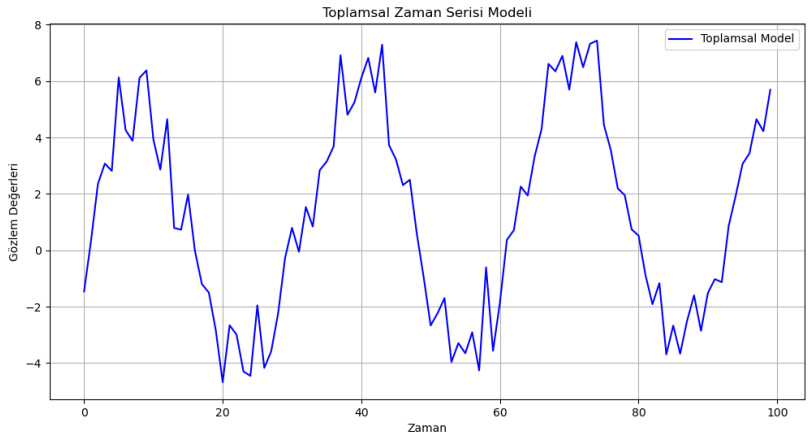
Toplamsal model, zaman serisi verilerinin ortalama değerini tahmin etmek için kullanılır. Bu model verilerin ortalama değerindeki değişimi yansıtır ve verilerin geçmişteki trendlerine odaklanır.

Çarpımsal model ise zaman serisi verilerinin belirli bir zaman aralığında değişim hızını ve dalgalanmasını tahmin etmek için kullanılır. Bu model, verilerin mevsimsel etkilerine veya döngüsel değişimlerine odaklanır.

Toplamsal model zaman serisi bileşenlerinin toplanmasıyla elde edilir.

$$Y_t = T_t + S_t + C_t + I_t \tag{3.1}$$

Y_t, t zamanındaki gözlemleri temsil etmektedir. T_t trend, S_t mevsimsellik, C_t döngü ve I_t düzensiz etki(hata) bileşenleridir. Bu bileşenlerden herhangi birinin olmaması durumunda ilgili bileşen 0 kabul edilerek analize dahil edilir.



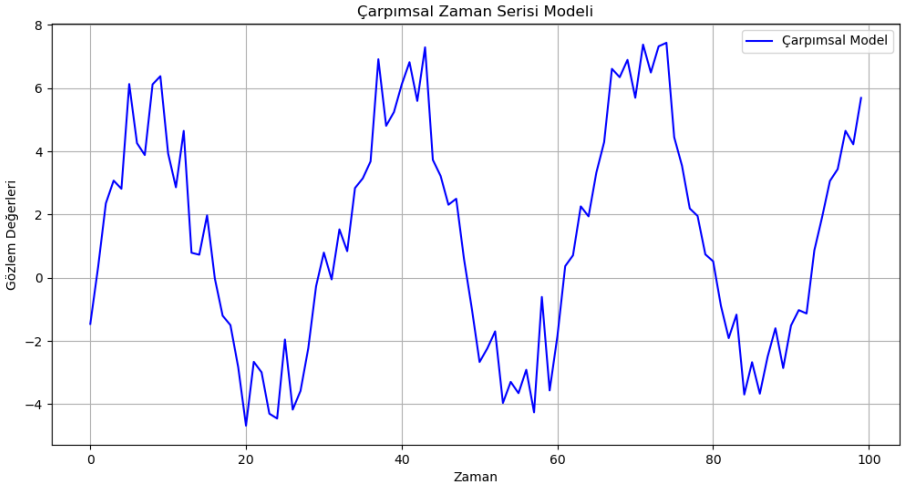
Şekil 3.5. Toplamsal Model

Toplamsal modeldeki bileşenler genellikle ayrı ayrı modellenerek seri yeniden oluşturulur. Seride uzun vadeli bir trend varsa, mevsimsel etkilerde mutlak bir değişim gözleniyorsa bu serinin toplamsal olduğundan söz edilir. Yani mevsimsel etkiler trendden bağımsız hareket etmekte, zaman içinde sabit kalmaktadır.

Çarpımsal model ise zaman serisi bileşenlerinin çarpılmasıyla elde edilir.

$$Y_t = T_t \times S_t \times C_t \times I_t \tag{3.2}$$

Y_t, t zamanındaki gözlemleri temsil etmektedir. T_t trend, S_t mevsimsellik, C_t döngü ve I_t düzensiz etki(hata) bileşenleridir. Bu bileşenlerden herhangi birinin olmaması durumunda ilgili bileşen 1 kabul edilerek analize dahil edilir.



Şekil 3.6. Çarpımsal Model

Mevsimsel bileşen zaman içinde dönemden döneme değişkenlik gösteriyorsa seri çarpımsaldır. Çarpımsal modelde serinin tüm bileşenleri birbirleriyle etkileşim halinde olarak birlikte modellenir. Çarpımsal modele uygun bir seri üzerinde logaritmik dönüşüm yapılarak toplamsal bir model elde edilebilir.

Bir zaman serisinin toplamsal ya da çarpımsal model olduğu pek çok yöntemle tespit edilebilir. İlk olarak seriye ait mevcut grafik incelenerek ani yükselme ve düşme trendlerinin varlığına göre serinin çarpımsal olduğu kanaatine varılabilir. Başka bir deyişle, benzer büyüme eğilimi gösteren gözlem noktaları toplamsal, üstel olarak değişen gözlemler çarpımsal modele işarettir. İkinci olarak serinin birim kök içerip içermediğini tespit etmek amacıyla Arttırılmış Dickey Fuller Testi ile durağanlık kontrolü yapılabilmektedir. Serinin durağan olması modelin toplamsal olduğu konusunda güçlü şüphe yaratmaktadır. Varyansın zamanla değişmesi ise serinin çarpımsal olduğunu ispatlar. Bir diğer yöntem ise serideki otokorelasyonun incelenmesidir. Otokorelasyon ve kısmi otokorelasyon grafikleri serideki gözlemler arasındaki ilişkinin boyutunu keşfetmemizi sağlar. Kısmi otokorelasyon grafiğindeki keskin değişimler modelin çarpımsal olduğunu göstermektedir. Otokorelasyon grafiğindeki daha yumuşak ve belirli bir örüntü göstermeyen değerler ise modelin toplamsal olduğunun işareti olabilir. Bütün bunların yanı sıra, zaman serisi verileriyle toplamsal ve çarpımsal olarak iki ayrı model oluşturulup performansları kıyaslanırsa, yüksek performans gösteren modelin serinin yapısını daha iyi yansıtan olduğu çıkarımında bulunulabilir.

Bir serideki bir gözlemin geçmiş ve gelecekteki değerlerinin ortalaması alınarak hareketli ortalama hesaplanmaktadır. Hareketli ortalama bir zaman serisinde gürültü azaltma, eğilimleri basitçe ortaya çıkarma ve trendi tespit etme gibi bir çok işleve sahiptir. Trend ve mevsimsellik hareketli ortalama yöntemiyle tespit edilebilmektedir. Ayrıca ilkel bir tahmin modeli oluşturmak için de kullanılabilmektedir.

3.2. Zaman Serisinde Üstel Düzleştirme

Üstel düzleştirme, zaman serileri analizinde verideki eğilim ve desenleri tahmin etmeye yardımcı olan yöntemdir. Bu yöntemle serideki geçmiş gözlem değerlerinin ortalaması alınarak gelecekteki değerler tahmin edilmeye çalışılır. Üstel düzleştirme ve hareketli ortalama yöntemlerinin bu benzerliğine karşın farklılaştığı nokta, ortalaması alınacak gözlem sayısının değişmesidir. Hareketli ortalamada serinin son *m* gözlem değerinin ortalaması alınır, düzleştirme yöntemlerinde ise tahmin için tüm serinin ortalaması dikkate alınır. Daha yakın geçmişteki verilere daha çok ağırlık verilerek tahminler üretildiği için bazı üstel düzleştirme yöntemleri trend ve mevsimsel etkileri tespit edebilmektedir.

3.2.1. Basit Üstel Düzleştirme

Basit üstel düzleştirme trend ve mevsimselliğin olmadığı zaman serilerinin tahmininde kullanılır (Pal ve Prakash, 2017). Geçmiş gerçek değerler ve geçmiş tahmin edilen değerlerin üssel olarak ağırlıklandırılmasıyla tahmin yapılır.

$$\hat{y}_t = \alpha y_{t-1} + (1-\alpha)\hat{y}_{t-1} \tag{3.3}$$

$\hat{y}_t$: t zamanı için tahmin edilen değer

y_{t-1} : bir önceki zamanın gerçek değeri

$\hat{y}_{t-1}$: bir önceki zamanın tahmin edilen değeri

α : düzleştirme(yumuşatma)parametresi($0<\alpha<1$)

Model her periyotta son gözlemden öğrenerek, son tahminden hatırlama işlemini gerçekleştirecektir.

3.2.2. İkili Üstel Düzleştirme

Bu yöntem basit üstel düzleştirmenin trend etkisi gözetilerek geliştirilmiş versiyonudur. Trend içeren ve mevsimsellik içermeyen tek değişkenli zaman serileri için uygundur.

$$l_t = \alpha y_t + (1-\alpha)(l_{t-1} + b_{t-1}) \tag{3.4}$$

$$b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1} \tag{3.5}$$

$$\hat{y}_{t+1} = l_t + b_t \tag{3.6}$$

l_t: t zamanında tahmin edilen değer(seviye)

α: mevcut değerin ağırlık faktörü

y_t : t zamanındaki gerçek değer

b_t : t zamanında önceki tahminin artış oranı(trend)

β : önceki tahminin ağırlık faktörü

$\hat{y}_{t+1}$: t+1 zamanında düzeltilmiş tahmin değeri

3.2.3. Üçlü Üstel Düzleştirme (Holt-Winters)

Holt-Winters modeli diğer üstel düzleştirme modellerinin aksine hem trend hem de mevsimsellik içeren serilerin tahmininde uygulanabilmekedir. Düzey, trend ve mevsimselliğin modellendiği durumlar aşağıdaki gibidir.

$$l_t = \alpha(y_t - s_{t-s}) + (1-\alpha)l_{t-1} \quad (3.7)$$

$$b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1} \quad (3.8)$$

$$s_t = \gamma(y_t - l_t) + (1-\gamma)s_{t-s} \quad (3.9)$$

$$\hat{y}_{t+m} = l_t + mb_t + s_{t+m-s} \quad (3.10)$$

l_t: t zamanında tahmin edilen değer(seviye)

b_t : t zamanında önceki tahminin artış oranı(trend)

s_t: t zamanında mevsimsel bileşen

$\hat{y}_{t+m}$: t+m periyot sonraki tahmin

3.3. Box-Jenkins Modelleri

Box-Jenkins modelleri literatürde sıklıkla başvurulan bir zaman serisi tahmin yöntemidir. Bu yöntemin uygulanabilmesi için serinin kesikli ve durağan olması gereklidir. Fakat çeşitli durağanlaştırma yöntemleriyle durağan olmayan seriler durağan hale getirilebilmektedir.

Daha çok ARIMA olarak anılan bu modeller trend, mevsimsellik ve durağanlık bileşenlerinin olup olmamasından etkilenmediği için diğer yöntemlere oranla daha çok tercih edilmektedir.

Durağan olmayan seriler zaman serisi tahmini için durağanlaştırılarak modellenebilir bir formata getirilmektedir. Fark alma ve logaritmik dönüşüm yaygın kullanılan durağanlaştırma yöntemlerindendir.

Fark alma, zaman serisinin gözlemleri arasındaki değişiklikleri ortadan kaldırmayı hedefler. En basit ifadeye son gözlem değerinden bir öncekinin çıkarılmasıdır. Her fark işleminde bir gözlem kaybedildiği için bu yöntemin dezavantajı olarak nitelendirilebilir. Fark alma işlemi birinci derece, ikinci derece ve mevsimsel olarak kurgulanabilmektedir.

Durağanlaştırma yöntemlerinden biri de logaritmik dönüşümdür. Gözlem değerlerinin logaritması alınarak mevsimsel etkilerin dengelenmesi hedeflenir. Fark alma ve log dönüşümü bir arada kullanılabilmektedir.

3.3.1. Otoregresif Model(AR)

AR yönteminde önceki zaman adımlarındaki gözlemlerin doğrusal bir kombinasyonu ile tahmin işlemi yapılır. Trend ve mevsimsellik içermeyen tek değişkenli zaman serileri için uygundur. Zaman gecikme sayısına göre AR(1), AR(2), AR(3)… şeklinde ifade edilir.

$AR(p)$, p.dereceden AR süreci olmak üzere şöyle ifade edilir:

$$AR(p): Y_t = \sum_{i=1}^{p} \varphi Y_{t-i} + \epsilon_t \quad (3.11)$$

p : zaman gecikmesi sayısı(p=1 ise bir önceki zaman adımı ile model kurulmuştur.)

φ_i : i.gecikme katsayısı

Y_{t-i} : serinin i.gecikmesi

$\in_t$: hata terimi

3.3.2. Hareketli Ortalama(MA)

Önceki zaman adımlarında elde edilen hataların doğrusal bir kombinasyonu ile tahmin işlemi yapılır. MA modeli trend ve mevsimsellik içermeyen tek değişkenli zaman serileri için uygundur.

$MA(q)$, q.dereceden MA süreci olmak üzere şöyle ifade edilir:

$$MA(q): Y_t = \sum_{i=1}^{q} \theta_i \epsilon_{t-i} \quad (3.12)$$

q : zaman gecikmesi sayısı

θ_i : i.gecikme katsayısı

Y_{t-i} : serinin i.gecikmesi

$\in_t$: hata terimi

3.3.3. Otoregresif Hareketli Ortalama(ARMA)

AR ve MA süreçlerinin bir birleşimi olan ARMA modeli, geçmiş gözlem değerleri ve geçmiş hataların bir kombinasyonu ile tahmin yapar. Trend ve mevsimsellik içermeyen zaman serileri için uygundur. Bu model AR ve MA'dan farklı olarak iki modelin birleşimi olduğu için daha karmaşık zaman serilerinde daha iyi performans göstermektedir.

$ARMA(p, q)$, p. dereceden AR ve q. Dereceden MA süreci şöyle ifade edilir:

$$ARMA(p, q): Y_t = \sum_{i=1}^{p} \varphi Y_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} + \epsilon_t \quad (3.13)$$

$AR(p)$: p.dereceden AR süreci

p : zaman gecikmesi sayısı(p=1 ise bir önceki zaman adımı ile model kurulmuştur).

φ_i : i.gecikme katsayısı

Y_{t-i} : serinin i.gecikmesi

$MA(q)$: q.dereceden MA süreci

q : zaman gecikmesi sayısı

θ_i : i.gecikme katsayısı

Y_{t-i} : serinin i.gecikmesi

$\in_t$: hata terimi

3.3.4. Entegre Otoregresif Hareketli Ortalama(ARIMA)

AR, MA ve ARMA modelleri durağan olmayan serilerin tahmininde kullanılmaktadır. Gerçek hayat problemlerinde sıklıkla durağan olmama durum mevcut olduğu için fark alma işlemiyle durağanlaştırılan seriye entegre otoregresif hareketli ortalama modeli uygulanabilir. ARIMA(p, d, q) sürecinde önceki zaman adımlarındaki farkı alınmış gözlemlerin ve hataların doğrusal bir kombinasyonu ile tahmin yapılır. ARIMA(p,d, q) sürecindeki p otoregresif süreç derecesini, q hareketli ortalama süreç derecesini ve d durağanlık için gerekli olan fark alma sırasını temsil etmektedir (Pal ve Prakash, 2017). Bu model tek değişkenli trendi olan fakat mevsimselliği olmayan zaman serileri tahmininde kullanılabilir.

$ARIMA(p, d, q)$, aşağıdaki gibi modellenir:

$$Y_d = \sum_{i=1}^{p} \varphi_i - Y_{d-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-q} + \epsilon_t \qquad (3.14)$$

Y_d: Y_t serisinin d farkıdır.

p : gözlem değerleri gecikme sayısı(otoregresif derece)

q : hata değerleri gecikme sayısı(hareketli ortalama derecesi)

d : fark işlemi sayısı(fark derecesi)

φ_i : serinin i.gecikme katsayısı

Y_{d-i} : serinin d farkının i.gecikmesi

θ_i : i.gecikme katsayısı

ϵ_t : hata terimi

ARIMA sürecinde modelleme iki faktör gözetilerek yapılır. İlki fark alma derecesinin belirlenmesi, ikincisi AR ve MA süreçlerindeki p ve q parametrelerinin saptanmasıdır.

4. YAPAY SİNİR AĞLARI

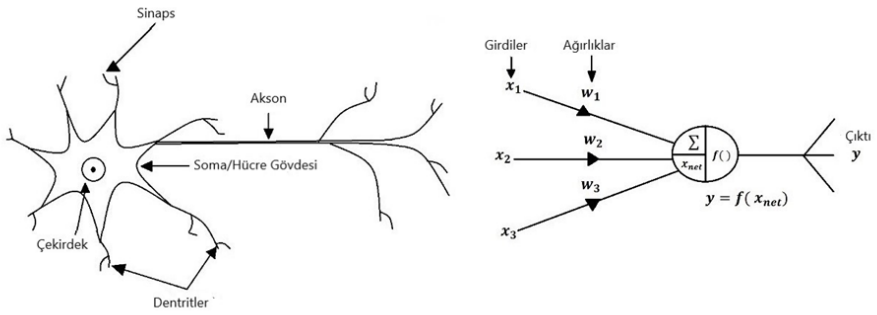
Yapay sinir ağları insan beyninden esinlenerek geliştirilmiş, nöronların öğrenme davranışlarına benzer eğilim gösteren yapılardır. Çeşitli girdiler ve ara işlemler sonucu bir çıktı üretmesi beklenen yapay sinir ağları, büyük ve karmaşık veri setleri üzerinde klasik yöntemlere oranla yaygın olarak kullanılmaktadır.

4.1. Yapay Sinir Ağları Tanımı ve Tarihçesi

1940'ların başlarında W.S. McCulloch ve W.A. Pitts'in (1943) makalesiyle ilk defa bir ilk sinir ağı modeli geliştirilerek yapay sinir ağlarının tarihi başlamıştır. McCulloch-Pitts (1943) sinir ağı modeli kullanarak çeşitli mantıksal fonksiyonların çıktılarını üretmeyi hedeflemiştir. 1949'da Donald Hebb tarafından ortaya atılan Hebbian Öğrenme stratejisiyle, sinir ağının bağlantı sayısı ile öğrenme ve uyumun ilişkili olduğu tespit edildi. Bu çalışmadan sonra ilk Hebbian ağı Farley ve Clark(1954) tarafından MIT'de uygulanmıştır. Aynı dönem psikolog Frank Rosenblatt bir sineğin gözünde bulunan ve kaçma tepkisine sebep olan basit karar sistemlerini anlamaya çalışıyordu. 1958 yılında Rosenblatt tarafından beynimizdeki nöronların nasıl çalıştığının basitleştirilmiş bir matematiksel modeli olarak perceptron tasarlandı. Widrow ve Hoff (1960) çok yönlü öğrenme yönteminin paralel erişimli bellek sistemleri ile bir araya geldiğinde mümkün olabileceğini savunmuş ve Rosenblatt'in algılayıcı modelindeki öğrenme kuralını geliştirerek 'Adaline' modelini ortaya koymuşlardır. Ayrıca Rosenblatt (1960), beynin modellenmesi hakkındaki araştırmalarının ve fikirlerinin bir kısmını içeren "Nörodinamiğin İlkeleri" kitabını yayınladı. Devrim niteliğindeki bütün bu araştırmalara rağmen algılayıcının kısıtlı bir başarı olduğu düşünülmeye başlandı. Algılayıcıların lineer olmayan problemlere yönelik çözüm üretemediğini iddia eden Pappert ve Misnky, 1969 yılında Perceptrons adında bir kitap yayınlayarak bu iddiayı ispatlamaya çalıştılar. XOR problemine çözüm bulamaması nedeniyle sinir ağları araştırmaları 80'lerin ortalarına kadar durdu. Paul Werbos (1974) doktora tezinde, sinir ağlarını optimize etmek için geri yayılımı kullanmayı öneren ilk kişi olarak sinir ağlarını optimize etmek istedi. Buna rağmen sinir ağları gereken ilgiyi halen göremiyordu. Sonrasında çok katmanlı algılayıcıların ortaya çıkmasında kilometre taşı sayılan Rummelhart'ın (1987) paralel programlama çalışmaları ile XOR probleminin çözülmesi, yapay sinir ağlarının popülaritesi yeniden artırmıştır. Yann LeCun (1989) geri yayılıma dayalı bir teknik geliştirerek el yazısıyla yazılmış rakamları tanıyan bir çalışma başlattı. Bu çalışmayla hedeflenen, posta zarflarında el yazısıyla yazılan posta kodunu bilgisayarın tanıyabilmesini sağlamaktı. Bu yöntem işe yaradı ve derin öğrenme için büyük bir başarı elde edildi. Bundan sonraki süreçte sinir ağlarına ilişkin çalışmalar hız kazandı ve "Derin Öğrenme" kavramının popüler hale gelmesine ön ayak oldu. Bu tarihten günümüze kadar yapay sinir ağlarıyla ilgili çalışmalar artan bir ivme ile devam etmektedir. Yeni öğrenme algoritmaları geliştirilmekte, siniz ağları daha kullanılabilir formata getirilmeye çalışılmaktadır.

4.2. Sinir Ağı Mimarisi ve Bileşenleri

Şekil 1 de görüldüğü gibi yapay sinir ağları, biyolojik bir sinir hücresini taklit ederek çeşitli girdiler sonucu bir çıktı üretmeyi hedefler. Biyolojik sinir ağı, canlı bir organizmanın sinir sisteminde bilgi transferinin yerine getirilmesini sağlar. Bu sinir sisteminin en küçük birimi "nöron" olarak bilinir. İnsan beyninde yaklaşık 10 ila 100 milyar nöron vardır ve her biri diğerlerine "sinapslar" ile bağlıdır. İnsan beyninde yaklaşık 100 trilyon sinaps vardır. Bu bağlantılar insan vücudunu ve düşünce süreçlerini kontrol eder. Kısaca, Yapay Sinir Ağları(YSA) insan beyninin öğrenme süreçlerini kopyalamaya çalışır. İlk YSA teorileri, insan davranışını ve düşünme sürecini insan beynini modelleyerek açıklamaya çalışan araştırmacılar tarafından açıklanmıştır. YSA alanında önde gelen araştırmacıların çoğu bugüne kadar psikoloji alanında bir geçmişe sahiptir.



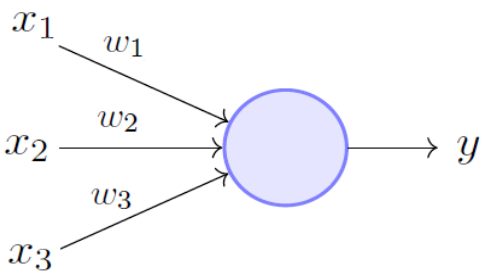
Şekil 4.1. Biyolojik ve Yapay Sinir Hücresi

Nöron: Sinir ağına gönderilen bilgileri işlemek için kullanılan en küçük birimdir.
Düğüm(Node): Verilerden gelen girdiyi birleştirip bir takım işlemler ve bir çıktı barındıran yapıdır.
Katman(Layer): Bir yapay sinir ağı giriş katmanı, gizli katman ve çıkış katmanından oluşur. Birden fazla gizli katmana sahip sinir ağları derin yapay sinir ağı olarak tanımlanır.
Ağırlık ve sapma değişkeni (Weight and biases): Sinir ağının öğrenmesi için gerekli parametrelerdir.

4.2.1. Tek Katmanlı Algılayıcılar

Algılayıcılar, 1958'de bilim adamı Frank Rosenblatt tarafından geliştirilmiştir. Tek bir yapay sinir hücresinden oluşan, gizli katmanı olmayan algılayıcılar tek katmanlı algılayıcılar olarak adlandırılır.

Algılayıcı modeline göre algılayıcının çıktısı, girdilerin ağırlık değerleriyle çarpımı ve daha sonra toplanmalarıyla belirlenir. Değişken bir eşik değeri belirlenir ve sonuç bu eşikten küçükse çıktı 0'dır, girdi-ağırlık çarpımlarının doğrusal toplamı bu eşikten büyükse çıktı sistem 1 olarak belirlenmiş olur.

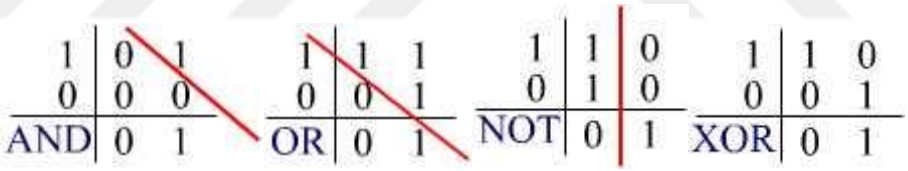


Şekil 4.2. Sinir Hücresi Girdiler, Ağırlıklar ve Çıktı

$$\sum_{i=1}^{n} w_{ij} x_j = \theta_j \tag{3.1}$$

Yapay sinir ağları ilk zamanlar tek katmanlı algılayıcılarla temsil edilmiş, ikili sınıflandırmada girdinin belli bir kategoriye ait olup olmama durumunu tahmin etmeye çalışmıştır. 1969 yılında Minksy ve Papert algılayıcıların çok sınırlı yetkinliklere sahip olduğunu, doğrusal olmayan problemlerde başarılı olamayacağını öne sürmüştür. Özellikle XOR problemi sinir ağlarının gelişimini duraklatan önemli olayların başında sıralanabilir.

Gallant (1993) AND, OR ve NOT gibi mantık operatörlerinin doğrusal olarak ayrılabilir olduğunu göstermiştir. Doğrusal ayrılabilirlik, iki sınıfa ait tüm gözlem değerlerinin 3 boyutlu uzayda tek bir düzlem ile ayrılabilir olduğunu ifade eder.



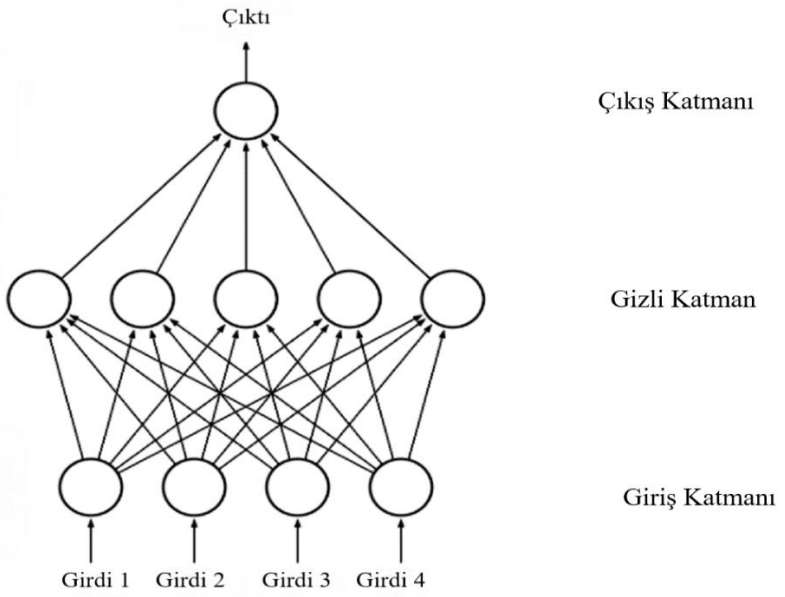
Şekil 4.3. Doğrusal Olarak Ayrılabilir Mantık Fonksiyonları

Yukarıdaki eksenlerde belli girdiler sonucu farklı çıktılar birbirlerinden ayrılabilmektedir. Fakat XOR operatörü için aynı durum söz konusu değildir. 1 veya 0 içeren çıktılar doğrusal bir düzlemle birbirinden ayrılamayacağı için XOR operatörü doğrusal ayrılabilir değildir.

4.2.2. Çok Katmanlı Algılayıcılar

İlk olarak geliştirilen ve en ilkel yapay sinir ağı olarak değerlendirilen tek katmanlı algılayıcı (single layer perceptron) doğrusal olmayan problemlerin çözümünde yeterli olmadığından çok katmanlı algılayıcılar (multi-layer perceptron) geliştirilmiştir (Fernandez ve ark., 2006).

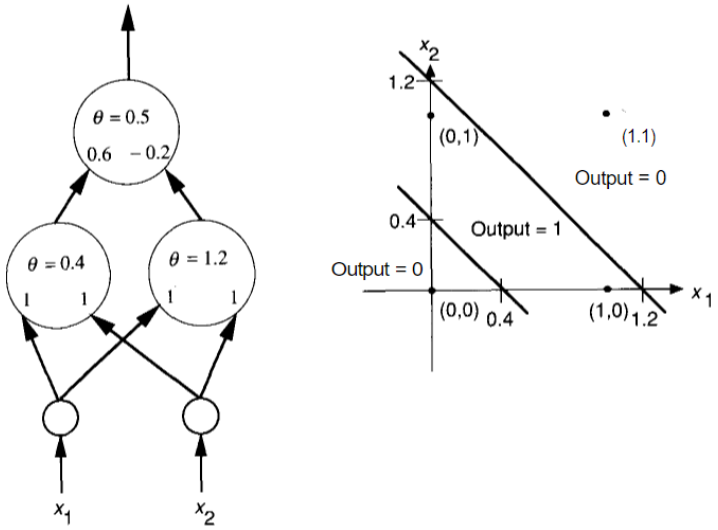
Derin sinir ağları bir veya daha fazla gizli katmana sahip sinir ağı mimarisini temsil etmektedir. Çok katmanlı algılayıcılar da derin sinir ağlarının bir alt kümesi sayılabilir. Doğrusal modellerin sınırlamalarının üstesinden gelmek için bir veya daha fazla gizli katmanı birbiri üzerine yığarak, her katmanın kendinden öncekinden beslenmesini sağlayabiliriz.



Şekil 4 4. Çok Katmanlı Algılayıcı

Geliştirilen bu mimari çok katmanlı algılayıcıdır ve kısaca MLP şeklinde ifade edilir. Problemin niteliğine göre gizli katman sayısı, bu katmanlara bağlı nöron sayısı ve çıktı sayısı değişkenlik göstermektedir. Şekildeki sinir ağında 4 girdi, 1 gizli katman ve 1 çıktı içermektedir.

Tek katmanlı algılayıcıların XOR operatörünün çözümünde etkin olmadığı için doğrusal ayrılamayan problemlerde çok katmanlı algılayıcıların kullanıldığı ifade edilmişti.



Şekil 4.5. XOR Problemini Çözen Çok Katmanlı Algılayıcı Modeli

Şekilden de anlaşılacağı üzere gizli katmandaki iki nöron sayesinde çizgiler uzayı 3 farklı bölgeye ayırabilmiş XOR problemi başarıyla çözümlenmiştir. Bu problemin çözümüyle beynin iyi tanımlanmış işlevleri yerine getiren birçok farklı paralel, dağıtılmış sistemden oluştuğu, ancak bir

veya daha fazla düzeyde bir seri işleme sisteminin kontrolü altında olduğu makul görünmektedir (Minsky ve Papert, 1969).

Çok katmanlı algılayıcılar eğitimin yönüne göre ileri beslemeli ve geri yayılımlı olarak ikiye ayrılmaktadır. İleri yayılımda sinyal, bir katmandaki hücrelerin çıkışından bir sonraki katmana giriş olarak iletilir. Geri yayılımda ise girişler ve çıkışlar arasındaki hata tespit edilerek ağırlık güncellemeleri gerçekleştirilmektedir.

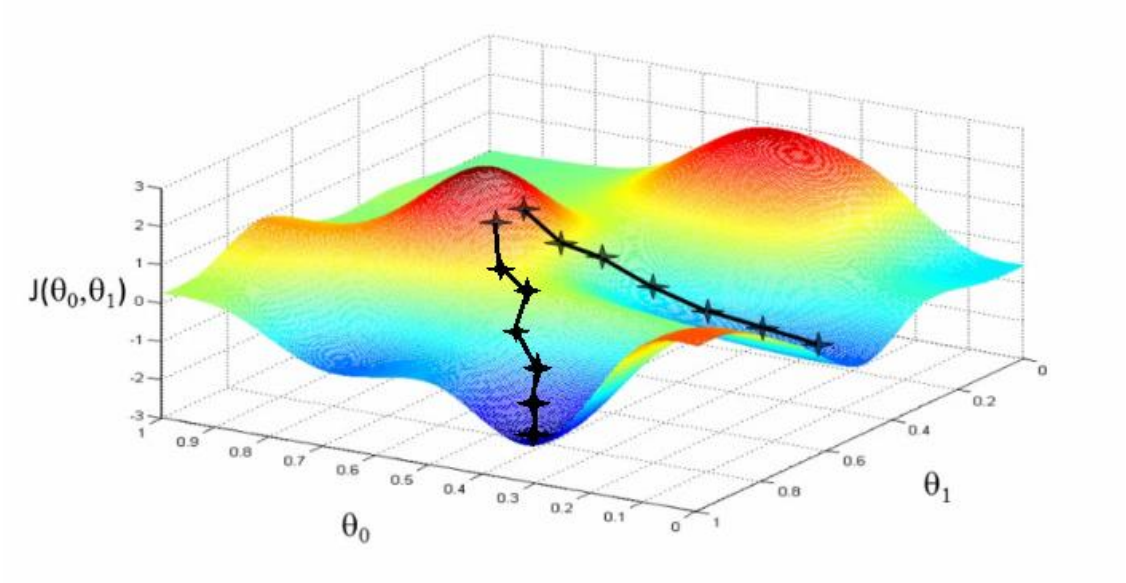
4.3. Gradyan İniş

Gradyan inişi sinir ağının ağırlıklarına göre hatayı en aza indirmek için kullanılan optimizasyon algoritmasıdır. Kademeli azalma olarak telaffuz edilen ve makine öğrenmesinde yaygın olarak kullanılan bu algoritma, kurulan modelin maliyet fonksiyonunu en aza indirmek üzere parametrelerin yinemeli olarak güncellenmesini hedefler.

Gradyan iniş, bir işlevi en aza indirmek için kullanılan bir optimizasyon algoritmasıdır. Makine öğrenimi ve diğer alanlarda yaygın olarak kullanılan bir algoritmadır ve birçok algoritma ve modelde önemli bir bileşendir.

Gradyan inişinin arkasındaki temel fikir, bir modelin parametrelerini, modelin tahminleri ile gerçek değerler arasındaki hatayı azaltacak şekilde yinelemeli olarak güncellemektir. Algoritma, her yinelemede model parametrelerine göre hata fonksiyonunun gradyanını hesaplar ve parametreleri hatayı en aza indirecek yönde ayarlar. Algoritma, minimum bir hataya veya önceden belirlenmiş bir durma noktasına ulaşana kadar bu güncellemeleri yapmaya devam eder.

Gradyan iniş yinelemeli bir süreçtir ve güncellemelerin boyutu, kullanıcı tarafından ayarlanabilen bir hiperparametre olan öğrenme hızı tarafından belirlenir. Daha büyük bir öğrenme oranı, daha büyük güncellemelere ve daha hızlı yakınsamaya yol açacaktır, ancak aynı zamanda algoritmayı minimumu aşmaya daha yatkın hale getirebilir. Daha küçük bir öğrenme oranı, daha yavaş yakınsamaya yol açacaktır, ancak algoritmanın daha kesin bir minimum bulmasına da yardımcı olabilir.



Şekil 4.6. Gradyan İniş

Şekilde bir çok lokal minimum maksimum noktası yer almaktadır. Öğrenme algoritmamız için en iyi θ1 ve θ2 parametrelerine yani global minimum noktasına ulaşmaya çalışır. Maliyet uzayını temsil eden eksenlere göre J(θ) maliyet fonksiyonudur. Global minimum noktası en düşük maliyete sahip olduğu için uygun öğrenme adımlarıyla bu noktaya erişilmesi algoritmanın öğrenme hatasının en alt seviyeye çekilmesi anlamına gelmektedir.

Genel olarak, gradyan iniş, makine öğrenimi modellerinin ve diğer karmaşık sistemlerin performansını optimize etmek için önemli bir araçtır.

4.4. İleri Beslemeli Sinir Ağları

Çok katmanlı bir sinir ağı olan ileri beslemeli sinir ağları sinyalin ileri yönlü iletilmesi sebebiyle bu şekilde isimlendirilmektedir.

İleri beslemeli ağlarda öğrenme giriş katmanından çıkışa doğru tek yönlü gerçekleşir. Düğümler arası döngü söz konusu değildir. Bilgi sırasıyla giriş katmanı, gizli katman ve çıkış katmanı şeklindekki bir sırayla iletilir. Ağın çalışma prensibi genel olarak gizli katmandaki birimlerin her girdinin değerini ilgili ağırlıkla çarparak toplamları aktivasyon fonksiyonuna göndermesi ve buradan da bir çıktı üretilmesi şeklindedir.

Herhangi bir döngünün olması durumu, çıkış katmanındaki bilgiden giriş katmanının beslenerek bir geri bildirim oluşmasını ifade etmektedir. Bu şekilde bir öğrenme sürecinin gerçekleşmesi geri yayılımın varlığını gösterir niteliktedir. İleri beslemeli ağlarda katman sayısı ve katmanlardaki düğüm sayıları değiştirilerek çoklu sınıflandırma problemleri çözülebilmektedir. Görüntü işleme çalışmalarıyla nesne tanıma, doğal dil işleme uygulamaları ileri beslemeli sinir ağları kullanılan uygulamalardır.

4.5. Geri yayılım

Geri yayılım bir sinir ağının çıktılarının doğruluğunu artırmak için kullanılan bir algoritmadır. Bir modelin değişkenlerine göre kayıp fonksiynunun gradyanını hesaplamak amacıyla kullanılır.

"Ağın gerçek çıktı vektörü ile istenen çıktı vektörü arasındaki farkın bir kısmını en aza indirmek için ağdaki bağlantıların ağırlıklarını yinelemeli olarak ayarlar." (Rumelhart ve ark., 1986).

1989'da yayınlanan makaledeki ifadeden de anlaşılacağı üzere geri yayılım hata oranını(maliyet fonksiyonunu) en aza indirmek için her ağdan ileri yayılım sonrası öğrendiklerini kullanarak geriye dönerken ağırlıklarını ve sapmalarını güncellemeyi hedefler. Ne kadarlık bir düzeltme uyguanacağı bu parametrelere ve maliyet fonksiyonunun gradyanlarına göre belirlenmektedir.

4.6. Kayıp/Maliyet Fonksiyonları

Kayıp fonksiyonu ağın tahmin ettiği değerle gerçek değerler arasındaki farkı ifade etmektedir. Bu iki parametre arasındaki mesafe değeri çeşitli yöntemlerle hesaplanarak modelin performansı hakkında yorum yapılabilmeye olanak tanır. Bu mesafeye göre düğümleri besleyen değişken ağırlıkları güncellenerek kayıp puanının düşürülmesi hedeflenir. Bu şekilde geri yönlü iletilen sinyaller geri yayılımı hatırlatmaktadır. Geri yayılım algoritması da bir nevi hatayı optimize eden bir işlemdir.

Genellikle kayıp fonksiyonu, maliyet fonksiyonu ve amaç fonksiyonu birbiriyle karıştırılmaktadır. Andrew Ng(2012) Machine Learning derslerinde kayıp fonksiyonunu tek bir örnek değeri için, maliyet fonksiyonunu ise eğitim setindeki tüm örnekler için hesaplanan hata olarak tanımlamaktadır. Amaç fonksiyonu ise Ian Goodfellow ve Ark (2015) "Deep Learning" kitabında maksimize ya da minimize edilmek istenen fonksiyon olarak tanımlamıştır. Sonuç olarak kayıp fonksiyonu, bir amaç fonksiyonunun türü olan bir maliyet fonksiyonunun bir parçasıdır. Maliyet fonksiyonu daha genel bir tanım olmasına karşın kayıp fonksiyonu tek bir gözlen değeri için hesaplanmaktadır.

Kullanılan sinir ağının veriyi ne kadar iyi modellediğini değerlendirmek için Kayıp Fonksiyonları sık sık başvurulmaktadır. Basit bir tabirle kayıp fonksiyonun verdiği değer ne kadar düşükse ağın o kadar iyi veriyi modellediğini söyleyebiliriz. Aksi takdirde modelin parametreleri güncellenmeli, kayıp en aza indirilmeye çalışılmalıdır.

Şimdi bazı kayıp fonksiyonlarını detaylı inceleyelim.

4.6.1. Regresyon Problemleri için Kayıp

En basit tabiriyle regresyon, öğrenme algoritmasına verilen bağımsız parametreler ile hedeflenen çıktıyı yani sürekli değeri tahmin etmektir.Regresyon analizinde performans değerlendirme metriği olarak kullanılan bir çok yöntem mevcuttur. İş problemine, veri setine ve daha

bir çok parametreye göre kullanılan ölçüt değişkenlik gösterebilir. Literatürde en çok gözlemlediğimiz kayıp fonksiyonları aşağıdaki gibidir.

Hata Kareler Ortalaması/İkinci Dereceden Kayıp/L2 Kaybı

En popüler kayıp fonksiyonlarından biri hata kareler ortalamasıdır. Formülden de anlaşılacağı üzere gerçek değerlerle tahmin edilen değerlerin farkının karelerinin ortalaması şeklinde belirlenir.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2 \tag{3.2}$$

$$L_2(x) = x^2 \tag{3.3}$$

$$f(y, \hat{y}) = \sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{3.4}$$

Hata kareler ortalaması bir tür öklid mesafesidir (Zhang ve ark., 2022).

Ayrıca hata kareler ortalaması derin öğrenmede genellikle L2 Kayıp Fonksiyonu şeklinde telaffuz edilmektedir. L2 kaybını kullanmanın faydalarından biri, türevlenebilir olmasıdır; bu, eğitim sırasında modelin ağırlıklarını güncellemek için gradyanları hesaplamak ve geri yayılım gerçekleştirmek için kullanılabileceği anlamına gelir.

Ortalama Mutlak Hata/L1 Kaybı

Bir diğer yaygın kayıp fonksiyonu ise ortalama mutlak hata olarak bilinen L1 kayıp fonksiyonudur.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{f}(x_i)\right| \tag{3.5}$$

$$L_1 = |x| \tag{3.6}$$

$$f(y, \hat{y}) = \sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{3.7}$$

Genel olarak gerçek değerler ve tahmin edilen değerlerin ortalama mutlak farkını ölçer. Aykırılıklara L2 kaybı kadar duyarlı değildir. Hatayı lineer olarak cezalandırır. L1 Kaybı y' = y noktasında türevlenebilir değildir.

L2 kaybı gibi gerçek ve tahmin edilen değerlerin farkının karesi alınmadığı için büyük hata değerlerinin ortaya çıkması engellenir ve olası yanlılıklar giderilmiş olur. Ayrıca türevlenebilir bir fonksiyon olmadığı için gradyan temelli gradyan inişi gibi optimizasyon yöntemlerinde kullanılamaz.

Huber Kayıp

L1 kaybı her noktada türevlenemez olduğu için L2 kaybı da aykırılıklara karşı çok duyarlı olduğu için Huber Kaybı iki fonksiyonun özelliklerini birleştirerek bir uzlaşma sağlamayı hedefler. Huber (1964) kayıp fonksiyonunu aşağıdaki gibi tanımlamıştır:

$$L_\delta\big(y, f(x)\big) = \begin{cases} \frac{1}{2}(y - f(x))^2 & |y - f(x)| \le \delta \; için \\ \delta|y - f(x)| - \frac{1}{2}\delta^2 & diğer \; durumlar \end{cases} \tag{3.8}$$

δ ayarlanmış bir parametre, y tahmin edilen değerleri ve f(x) gerçek değerleri temsil etmektedir. Kayıp küçük olduğunda L2 kayıp fonksiyonu, büyük olduğunda L1 kayıp fonksiyonu kullanılır.

Bu modelin dezavantajı eşik değeri olarak belirlenen δ hiperparametresinin model doğruluğunu en üst düzeye çıkarmak için optimize edilmesi gerektiğidir.

4.6.2. Sınıflandırma Problemleri için Kayıp

Sınıflandırma bir veri kümesini belirli kriterlere göre çeşitli kategorilere ayırmayı temsil etmektedir. Model girdilerinin hangi sınıfa ait olduğunu tespit etmek için sınıflandırma kullanırız. Sınıflandırma problemlerinde de kullanılan bir çok kayıp fonksiyonu mevcuttur.

İkili Çapraz Entropi Kaybı (LOG LOSS)

Bilgi Teorisine göre entropi, düzensizliği temsil etmektedir (MacKay, 2003). Tüm sınıflar eşit olasılıklıysa maksimum düzenden söz edilebilir. Öngörülen sınıf olasılığı gerçek olasılıktan uzaklaştıkça çapraz entropi kaybının da arttığından söz edebiliriz. Yani belirsizliğin ve rasgeleliğin artması sınıf tespitini zorlaştıracaktır.

İkili çapraz entropi kaybı da iki kategorili sınıflandırma için kullanılan bir kayıp fonksiyonudur. İlgili gözlemlerin gerçek sınıfı ile tahmin edilen sınıfın dağılımının karşılaştırılmasına imkan tanır.

$$L = -\frac{1}{m}\sum_{i=1}^{m}(y_i \,.\log(\widehat{y_i}) + (1 - y_i).\log(1 - y_i)) \tag{3.9}$$

Burda yi gerçek değerleri yi şapka tahmin edilen değerleri, m de eğitim setindeki gözlem sayısını temsil etmektedir.

Kategorik Çapraz Entropi Kaybı

Kategorik çapraz entropi kaybı, ikili çapraz entropi kaybından farklı olarak hedef değişkeninin 2'den fazla kategoriye sahip olması durumunda kullanılan bir kayıp fonksiyonudur.

$$CCE = -\sum_{c=1}^{C} t_{o,c} \log(p_{o,c}) \quad (3.10)$$

Kullback–Leibler Iraksaması

KL ıraksaması iki olasılık dağılımının arasındaki farkı belirlemek için kullanılır. Derin öğrenmede tahmin edilen değerlerle gerçek değerlerin dağılımının birbirine yakın olup olmadığının tespiti KL ıraksamasıyla mümkün olabilmektedir. Kullback ve Leibler (1951) tarafından geliştirilmiştir.

$$D_{KL}(P||Q) = \sum(P(i).\log(\frac{P(i)}{Q(i)})) \quad (3.11)$$

P ve Q karşılaştırılan iki olasılık dağılımını, i gözlem değerini ifade etmektedir.

Diğer Kayıp Fonksiyonları

Yukarıda sayılanlar dışında regresyon ve sınıflandırma problemlerinde kullanılmak üzere kayıp fonksiyonlarının bir çok varyasyonu mevcuttur.

Modele özgü kayıp fonksiyonları MinMax (GAN Loss), Discriminator Loss, Generator Loss, Focal Loss (Object Detection) gibi bir çok kayıp fonksiyonu yaygın olarak kullanılmaktadır.

4.7. Aktivasyon Fonksiyonları

Aktivasyon fonksiyonu, sinir ağındaki karmaşık yapıları öğrenmeyi ve veriyi doğrusal olmayan formatta ifade edebilmeyi sağlar. Etkinleştirme, bazı kurallara veya eşiklere bağlı olarak her katmandaki hangi nöronun ateşleneceğini söyleyerek ağda doğrusal olmamayı tanıtmaya imkan sağlar.

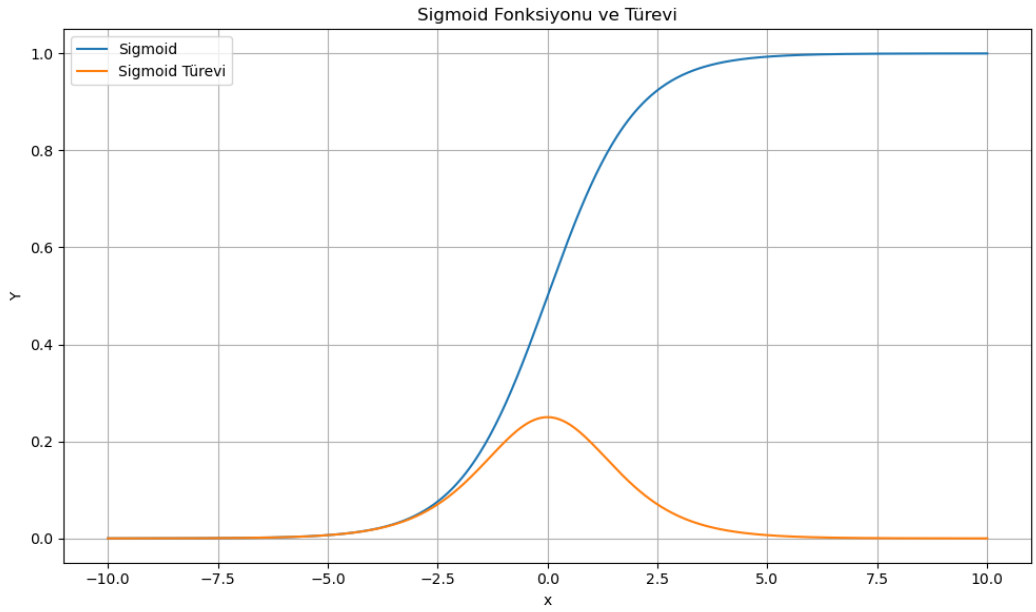
Doğrusal olmayan gerçek hayat problemlerini çözebilmemiz için böyle bir dönüşüme ihtiyaç duyulmaktadır. Aktivasyon fonksiyonu olmadan sinir ağları yalnızca doğrusal sınıflandırma gibi problemleri çözümleyebilirdi. Fakat aktivasyon fonksiyonları sayesinde verideki doğrusal olmayan yapılar tanımlanabilmekte ve buna uygun çıktılar üretilebilmektedir. Ayrıca verideki deseni iyi bir şekilde tespit etmek yine aktivasyon fonksiyonu sayesinde gerçekleşir. Bu nedenle kullanılan ağın ve problemin yapısına uygun bir aktivasyon fonksiyonu seçmek son derece önemlidir.

Sigmoid Aktivasyon Fonksiyonu

Sigmoid aktivasyon fonksiyonu girdileri 0 ile 1 arasında ölçeklendirerek nöron çıktılarını normalize etmeye yardımcı olur.

Matematiksel olarak aşağıdaki şekilde ifade edilir.

$$f(x) = \frac{1}{1+e^{-x}} \quad (3.12)$$



Şekil 4.7. Sigmoid Aktivasyon Fonksiyonu ve Türevi

Grafiğe göre x değeri 10 ile -10 arasında iken y değerleri hızlı bir şekilde değişmektedir. x'te yapılan küçük değişimlerin etkisi y'de daha büyük olacaktır. Bu durum bu fonksiyonun iyi bir sınıflandırıcı olarak kullanılabileceğini göstermektedir.

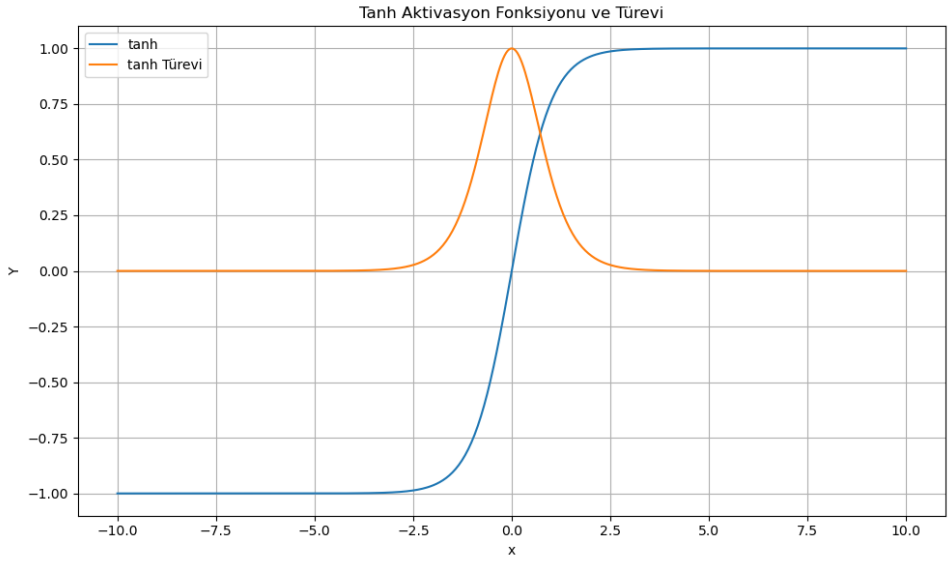
Bu fonksiyonun bir diğer avantajı da doğrusal fonksiyonda olduğu gibi -∞ ile +∞ arasında her zaman 0 ile 1 aralığında değer üretir.Yani aktivasyon fonksiyonu kaybolmaz. Fonksiyonun kuyruklarında ise y değişkeni x'teki değişikliklere çok az tepki vermektedir. Bu bölgelerde türev değerleri çok küçük olur ve sıfıra yakınsar. Buna gradyanların kaybolması denir ve öğrenme gerçekleştiğinde hatayı minimize eden optimizasyon algoritması yerel minimum noktasındaki değerlere takılı kalır ve yapay sinir ağı modelinden alınabilecek performans yetersiz kalır.

Hiperbolik Tanjant(Tanh) Aktivasyon Fonksiyonu

Tanh aktivasyon fonksiyonu sigmoid fonksiyonunun bir genişletilmiş versiyonudur. Yine çıktı değerleri 1 ile -1 arasında ölçeklendirilir.

Matematiksel olarak aşağıdaki şekilde ifade edilir:

$$tanh = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{3.13}$$



Şekil 4.8. Hiperbolik Tanjant Aktivasyon Fonksiyonu ve Türevi

Sigmoid aktivasyon fonksiyonuna göre avantajı türevinin daha dik olması yani daha çok değer alabilmesidir. Bu durum daha hızlı öğrenmeyi ve daha çok değer alabilmeyi sağlar.

ReLu ve PReLu Aktivasyon Fonksiyonları

ReLU fonksiyonu, pozitif girdileri doğrudan ileterek ve negatif girdileri sıfıra eşitleyerek çalışır. Gradyanların kaybolması bu fonksiyonda da girdi pozitifse çıktının da pozitif, negatifse çıktının sıfır olmasıyla ortaya çıkmasıdır.

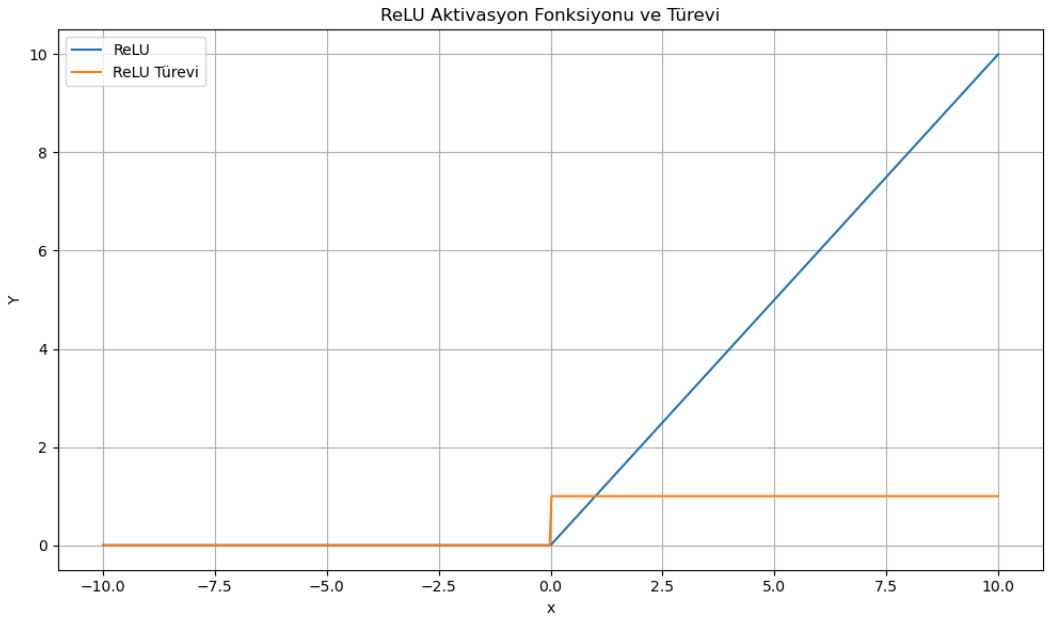
Matematiksel olarak aşağıdaki şekilde ifade edilir.

$$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \tag{3.14}$$

ReLu fonksiyonunun türevi:

$$f'(x) = \begin{cases} 1, & x > 0 \\ 0, & d.d \end{cases} \tag{3.15}$$

ReLu $[0, +\infty)$ aralığında değer alır. Çok fazla nöronlu büyük bir sinir ağında Sigmoid ve Hiperbolik Tanjant neredeyse tüm nöronların aynı şekilde ateşlenmesine(aktive olmasına) sebep olur. Bu çok fazla işlem gerektiği anlamına gelir. Ağdaki bazı nöronların aktif olması aktivasyonun seyrek yani verimli bir hesaplama sürecini doğurur. ReLu ile bu süreç sağlanmış olur.

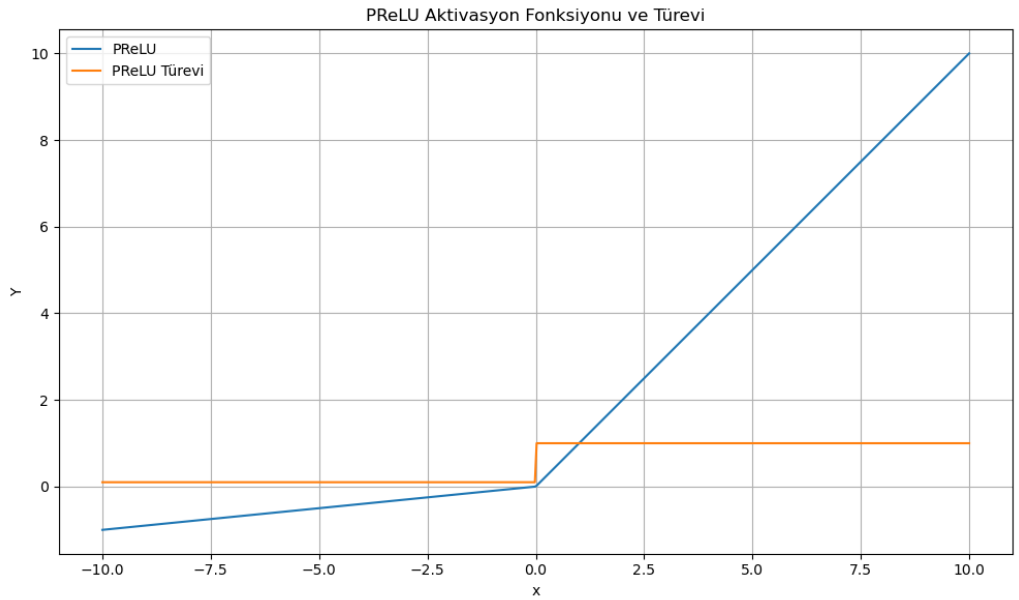


Şekil 4.9. ReLu Aktivasyon Fonksiyonu ve Türevi

PReLu, ReLu fonksiyonunun negatif girdileri sıfıra eşitleme sorununu çözmek üzere geliştirilmiştir. Negatif girdiler için pozitif bir eğim faktörü içeren ekstra bir parametreyle PReLu fonksiyonu oluşturulur. Böylece negatif değerlere esneklik sağlanır, gradyanların kaybolması sorununu hafifletir ve aşırı öğrenmenin önüne geçer.

Matematiksel olarak aşağıdaki şekilde ifade edilir:

$$f(x) = \begin{cases} x, & x > 0 \\ \alpha . x, & x \leq 0 \end{cases} \tag{3.16}$$



Şekil 4.10. PReLu Aktivasyon Fonksiyonu ve Türevi

PReLu grafiği pozitif girdilerde ReLu'ya benzerken, negatif girdilerde bir doğrı çizer. Ayrıca bu fonksiyonun türevi $x = 0$ noktasında tanımsızdır.

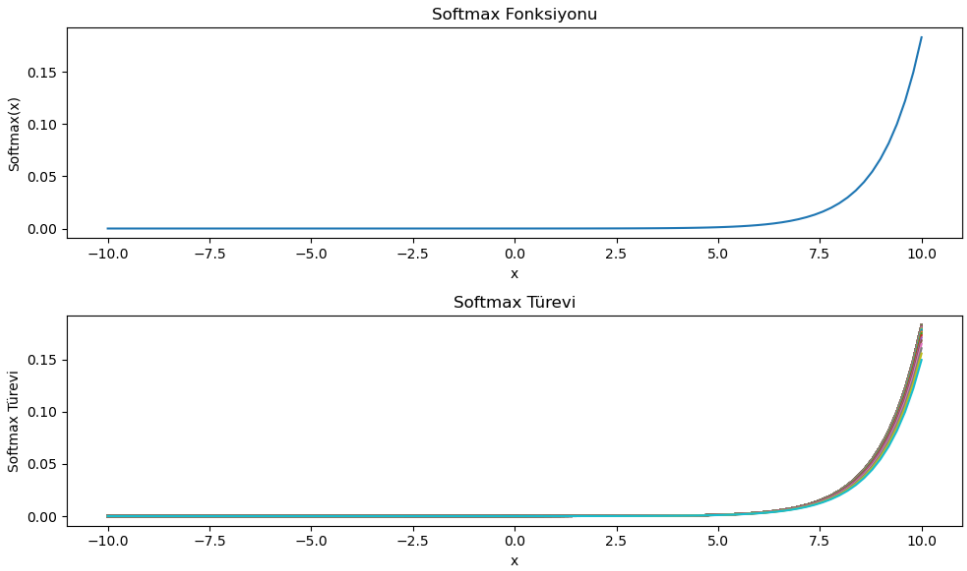
Softmax Aktivasyon Fonksiyonu

Bu fonksiyon sinir ağının çıktılarını olasılık dağılımına dönüştürerek gözlemin hangi sınıfa ait olduğunu tahmin etmeye çalışır.

Matematiksel olarak aşağıdaki şekilde ifade edilir:

$$f_i(\vec{x}) = \frac{e_{x_i}}{\sum_{j=1}^{J} e^{x_j}} \tag{3.17}$$

J: sınıf sayısı

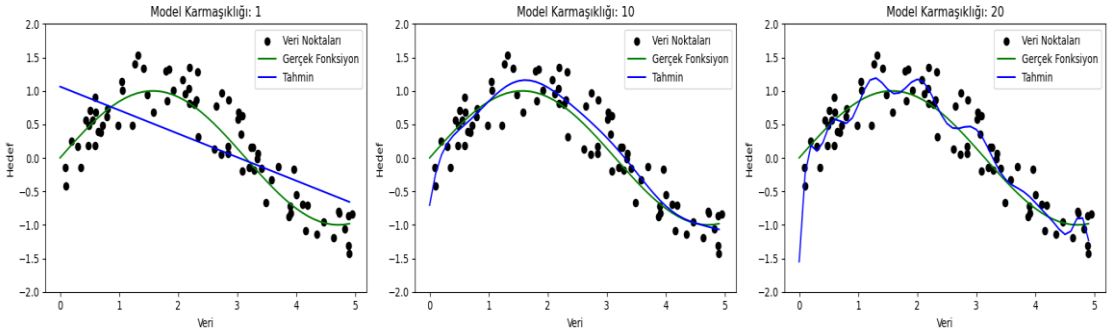


Şekil 4.11. Softmax Aktivasyon Fonksiyonu ve Türevi

Sigmoid fonksiyonuna benzer yapıda olan bu fonksiyonun en önemli farkı ikiden fazla sınıflı problemlerde özellikle derin öğrenme modellerinin çıkış katmanında tercih edilmesidir. Girdilerin belirli bir sınıfa ait olma olasılığını 0 ile 1 arasında değerler üreterek belirlenmesini sağlamaktadır. Softmax fonksiyonu, girdi değerlerini pozitif hale getirir ve ardından tüm sınıfların olasılık toplamına bölünerek normalleştirilir. Bu sayede çıktılar, 0 ile 1 arasında ve toplamı 1 olan bir olasılık dağılımını temsil eder. Ayrıca Softmax çıktıları gradyan inişle uyumlu olduğu için öğrenme işlemi kolaylaşır.

4.8. Düzenlileştirme ve Normalizasyon

Aşırı öğrenme, makine öğrenmesi modellerinde ve sinir ağlarında oldukça sık rastlanan bir problemdir. En basit tabirle modelin eğitim verilerine çok fazla uyum sağladığı, bu nedenle görmediği verilerde düşük performans göstermesi olarak açıklanabilir. Eğitim verilerindeki gürültüyü iyi öğrenmesi ve örneklem hatası, modelin aşırı öğrenmesine sebep olur.



Şekil 4.12. Yanlılık Varyans Takası

Şekil 4.12.'de ilk grafikte tahmin değerlerinin gerçek değerlerle uyumsuz olduğu görülmektedir. Bu durum modelin veriyi öğrenemediğini ve yüksek yanlılığa sahip olduğunu göstermektedir. Üçüncü grafikte modelin veriyi aşırı öğrendiği, her deseni takip ettiği ve yüksek varyansa sahip olduğu görülmektedir. İdeal bir denge noktası genellikle orta seviyede karmaşıklıkta bulunur. Ortadaki grafikte dengeli bir uyum mevcuttur.

Aşırı öğrenmenin önüne geçmek için kullanılan bir çok yöntem mevcuttur. Eğitim verisini artırma, sentetik veri oluşturma, erken durdurma gibi yöntemler aşırı öğrenmenin önüne geçmeye yardımcı olur. Ayrıca bazı düzenlileştirme ve normalleştirme yöntemleri de bu durumun üstesinden gelmek için kullanılmaktadır(Nielsen, 2015). Sinir ağları karmaşık yapılar olduğu için aşırı öğrenmeye eğilimlidirler. Bu nedenle modelleme aşamasında düzenlileştirme ve normalleştirme kullanılması oldukça önemlidir.

Model için anlamlı olmayan değişkenler öğrenmeyi zorlaştıracak bir nevi gürültü etkisi yaratacaktır. Bu gürültünün modelin aşırı öğrenmesine sebep olmaması ilgili değişkenlere ceza katsayıları uygulanarak etkilerinin azaltılması veya yok edilmesi amaçlanır. L1 ve L2 düzenlileştirme yöntemleriyle bu işlemler gerçekleştirilir.

Sinir ağı eğitim sürecinde nöronlar ağırlıklı olarak birbirine benzer olma eğilimindedir. Bu da yine aşırı öğrenmenin nedenlerinden biridir. Sönümleme yani düğüm seyreltme, öğrenme sırasında rastgele seçilen nöronların ve bağlantıların geçici olarak devre dışı bırakılması anlamına gelir. Sönümleme, her eğitim itreasyonunda ağın farklı bir alt kümesini kullanarak sinir ağının yararlı olan daha sağlam özellikleri öğrenmesini zorlar (Srivastava ve ark., 2014).

Düğüm seyreltme ağırlıkları ve bağlantıları rastgele devre dışı bırakarak aşırı öğrenmeye karşı dirençli bir model oluştururken aynı zamanda farklı özellik kombinasyonlarını öğrenmeye teşvik eder. Bu da modelin daha genelleştirilebilir formatta olmasını sağlar.

Aşırı öğrenme problemi normalizasyon ve standardizasyon yöntemleriyle de çözülebilir. Normalizasyon veri değerlerinin 0 ile 1 arasında ölçeklendirilmesidir. Ağın daha hızlı öğrenmesine yardımcı olur. Verilerin dağılımını değiştirmez, yalnızca belli bir aralıkta temsil eder. Standardizasyon ise verilerin ortalama değerinden ilgili gözlem değeri çıkarılıp standart sapmaya

bölünmesiyle elde edilir. Aynı şekilde ağın daha geniş bağlamda öğrenmesini sağlayarak aşırı öğrenmenin önüne geçmeye yardımcı olur.

Bu yöntemler dışında aşırı öğrenmeyi engellemeye yönelik toplu, katman, grup normalizasyonu gibi bir çok yöntem mevcuttur.

4.9. Optimizasyon Yöntemleri

Gradyan iniş, sinir hücrelerinin ağırlıklarını güncellemek için kullanılan genel bir yöntemdir. Tüm optimizasyon yöntemleri bu algoritmaya dayanır. Hatayı minimize etmek için kullanılan bu temel algoritmanın bir çok farklı varyantı mevcuttur.

Toplu gradyan iniş yöntemiyle bütün eğitim verisiyle ağırlıklar güncellenir. Gradyanların ortalaması alınarak parametre güncellemek için kullanılır. Bu işlem her bir döngüde iteratif olarak gerçekleştirilir. Her adımda bütün eğitim seti üzerinden hesaplama yapıldığı için çok büyük eğitim verilerinde çok yavaş çalıştığı için toplu gradyan iniş çok maliyetlidir.

Stokastik gradyan iniş, standart gradyan inişten farklı olarak her bir iterasyonda 1 örneklem alır ve bir seferde her örneklem için ağırlık güncellemesi gerçekleştirir. Yani döngü başına yineleme(iterasyon) sayısı örneklem sayısına eşittir. Bu şekilde eğitim verileri üzerindeki gradyan hesaplamaları daha hızlı gerçekleşir ve öğrenme süreci de hızlanır. Ancak, her adımda yalnızca 1 örneklem kullanıldığından gürültüsü çok olabilir. Hızlı öğrenme ve düşük bellek gereksinimi stokastik gradyan inişin avantajları arasında sayılabilirken, rasgele gradyan tahmini, ağırlık güncellemede dalgalanma ve yerel minimuma ulaşamama dezavantajları arasındadır.

Küçük toplu gradyan iniş, toplu ve stokastik gradyan inişin karması bir yöntemdir, her iki yöntemin de avantajlarını barındırır. Veri kümesinden daha az olan sabit sayıda eğitim örneğinden oluşan bir küme kullanılır ve buna mini-batch denir. Sabit boyutlu mini-batch'ler oluşturulduktan sonra iteratif bir şekilde tek bir döngüde hatanın minimizasyonu gerçekleştirilir.

Üstel ağırlıklı ortalama, modelin bir önceki ağırlık sonucu olan geçmiş gradyanları kullanılarak mevcut gradyanların güncellenmesi temeline dayanır. Bu durum gradyanların daha az gürültüye sahip olmasına yardımcı olur.

Momentum da benzer şekilde gradyanın son güncellemeye bağlı olarak yaptığı değişimleri takip eder ve buna göre günceller. Üstel ağırlıklı ortalamayı da kullanarak dikeyde salınımları indirger ve yatayda hızlanmaya yardımcı olur. En iyi optimizasyon algoritmalarından biri sayılabilir.

RMSPop, öğrenme temelli bir optimizasyon algoritmasıdır. Gradyanların kök karelerinin karesini kullanarak öğrenme hızını ayarlar. Dikeyde yayılımı minimize etmek yatayda hızı artırmak için daha fazla dalgalanma gösteren özelliklere daha düşük öğrenme oranı uygulanırken, daha az dalgalanma gösteren gradientlere sahip özelliklere daha yüksek öğrenme oranı uygulanır.

Uyarlanabilir Moment Tahmini(ADAM), RMSProp ve Momentum yöntemlerinin karmasıdır. Büyük veri setleri ve karmaşık problemler için uygundur. İki yöntemin avantajarını kullanarak gradyan inişi daha hızlı hale getirir.

Model eğitimi sırasında Adam'ı tek başına kullanmak yerine stokastik gradyan inişi kullanmak daha optimal sonuçlar ortaya çıkarmaktadır. Adam'den Stokastik Gradyan İnişe Geçiş(SWATS) yöntemiyle Adam ile derin sinir ağı eğitilmeye başlanır ve belirli kriterler gerçekleştiğinde stokastik gradyan inişe geçilir.

Ağırlık azaltma (L2 düzenlileştirilmesi) yöntemi, ağırlıkların etkisini sınırlamayı hedefleyerek modelin daha genelleştirilebilir bir formatta olmasını sağlar.

4.10. Hiperparametre Optimizasyonu ve Öğrenme Hızının Ayarlanması

Hiperparametre, eğitim sürecine başlanmadan manuel olarak ayarlanan bazı parametrelerdir. Parti boyutu, öğrenme hızı, optimize edici vs. model hiperparametreleri arasındadır.

Bir sinir ağında öğrenmenin ne kadar hızlı gerçekleşeceğini ayarlamak son derece önemlidir. Öğrenme hızı eğer çok küçükse eğitim süresi çok uzun ve yavaş ilerleyecek, model yerel minimumlara takılacaktır. Tam aksi durumda yani öğrenme hızı çok büyükse ağırlıklar hızlı bir şekilde güncelleneceğinden kayıp fonksiyonunun değeri yükselecektir. Ayrıca modelin genelleme yeteneğini kaybetmesine ve aşırı uyuma neden olabilir.

Bütün bunlar göz önünde bulundurulduğunda öğrenme hızının doğru seçilmesi üzerine bir çok yöntem geliştirilmiştir.

Adım azaltma(step decay) yöntemiyle öğrenme hızı belirli zaman aralıklarında yavaşça düşürülür. Döngüsel öğrenme hızı azaltmasıyla öğrenme hızı başlangıçta yüksektir, daha sonra daha düşük bir öğrenme hızı ayalarnır. Cosine fonksiyonu kullanılarak da benzer döngüsel öğrenme hızı ayarlama işlemleri gerçekleştirilebilir.

4.10.1. Ağırlıkların Başlatılması

Bir sinir ağı eğitilirken ağırlıkların hangi değerlerle başlatılacağı optimal sonuçların elde edilmesi açısından çok önemlidir. Sinir ağındaki her bir düğüm girdilerden elde edilen ağırlıklarla birbirini besler. Bu ağırlıklar maliyet fonksiyonunun değerini düşürmek için her seferinde güncellenir. Başlangıç ağırlıkları sinir ağlarının ilk geliştirildiği dönemlerde yaygın olarak küçük rastgele sayılarla başlatılıyordu. Günümüzde ağırlıkların başlatılmasına yönelik bir çok gelişmiş yöntem mevcuttur.

Rastgele başlatma, normal ya da uniform dağılıma sahip küçük rastgele değerler kullanılarak başlatılmasıdır. Xavier ağırlık başlatma ağın giriş ve çıkış katmanlarının toplam sayıları baz alınarak yapılır. Sigmoid ve hiperbolik tanjant aktivasyon fonksiyonlarıyla kullanılmak üzere tasarlanmıştır.

5. TEKRARLAYAN SİNİR AĞLARI(RNN)

RNN yani tekrarlayan sinir ağları, kendinden bir önceki ağdaki bilgiden beslenen bir sinir ağı türüdür. İleri beslemeli sinir ağlarının aksine geribildirim döngüsü mevcuttur. Bir diğer deyişle bellekte tutulan çıktı bilgileri bir sonraki ağa girdi olarak kullanılır. Böylece mevcut gözlem, kendinden önceki gözlemlerden etkilenerek şekillenir çıkarımında bulunulabilir.

Zaman serisi, dil çevirisi, konuşma tanıma gibi dizi formatındaki verilerin işlenmesi için elverişlidir.

5.1. Tekrarlayan Sinir Ağı Mimarisi

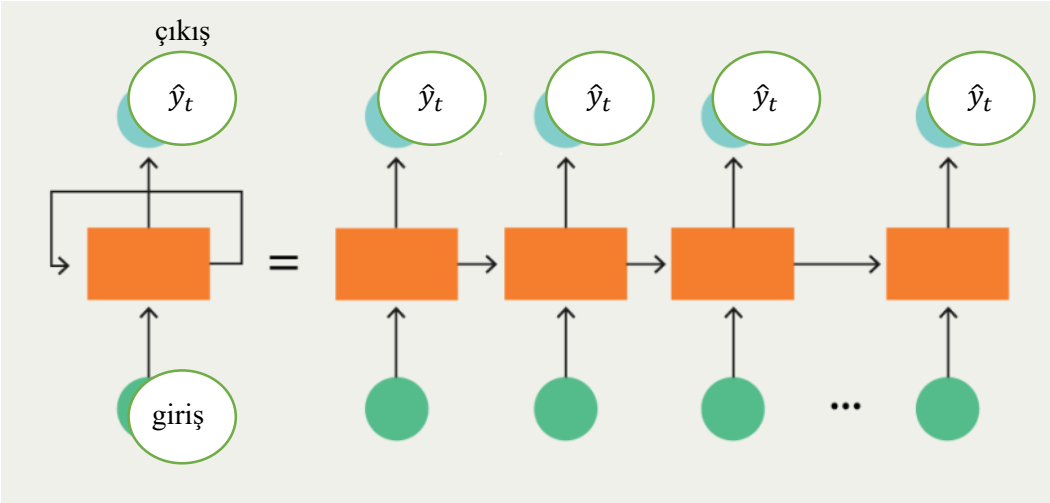
Tekrarlanan sinir ağları yani kısaca RNN'ler yinelenen ilişkileri ve ardışık verileri modellemek için kullanılan bir sinir ağıdır. Klasik sinir ağında olduğu gibi en küçük birimi olan düğümler(node) aşağıdaki bileşenlerden oluşur.

Giriş(Input): RNN'lerde giriş, mevcut zaman adımınındaki veri ve önceki zaman adımındaki gizli durumdan elde edilen bilgiden oluşur.

Gizli Durum(Hidden State): Bir önceki zaman adımınındaki bilgiyi temsil eder.

Ağırlıklar(Weights): Her giriş düğümünü güncelleyen bir ağırlık matrisi mevcuttur. Giriş verilerini gizli durum ve çıktıya dönüştürmeye yardımcı olurlar.

h Durumu Güncellemesi: Her bir zaman adımının gizli katman güncellemesinin aynı hücre içinde gerçekleştiğini ifade eder.



Şekil 5.1. Tekrarlayan Sinir Ağı Mimarisi

$$h_t = f_w(x_t, h_{t-1}) \tag{5.1}$$

Bu formülde,

h_t: mevcut durum

f_w: ağırlıkların olduğu fonksiyon

x_t: girdi

h_{t-1}:geçmiş durum

Şeklinde tanımlanır.

RNN'ler bir dizi işlenirken her zaman adımında güncellenen bir h durumuna sahiptir. Yukarıdaki formül, bir diziyi işlemek için her zaman adımınında uygulanan yineleme ilişkisini göstermektedir.

RNN'ler aynı anda hem gizli katman güncellemesi yapar hem de bir çıktı oluşturur. Gizli katmanın güncellenmesi:

$$h_t = \tanh(W_{hh}^T h_{t-1} + W_{xh}^T x_t) \tag{5.2}$$

Burada,

W_{hh}^T: ağırlık matrisi

h_{t-1}: önceki zaman adımındaki gizli durum

W_{xh}^T: diğer ağırlık matrisi

x_t: girdi

şeklinde tanımlanır.

RNN'ler bir çok farklı veri yapısını işleyebilme kabiliyetine sahiptir. Birebir, bire çok, çoklu giriş-çoklu çıkış, çoklu giriş-bire çok, bire çok-çoklu çıkış gibi farklı formlarda çeşitlenmesi ele alınan problem ve veri türü açısından çok önemlidir. Amidi (2023), aşağıdaki gibi bu formları belirtmiştir.

Bire Bir(One to One): Her bir girdi katmanı bir çıkış katmanı üretir. Sıralı olmayan veriler için uygundur. Bu yapıya geleneksel sinir ağı mimarisi örnek verilebilir.

Bire Çok(One to Many): Tek bir girdiyle birden fazla çıkış üretilir. Müzik notaları üretme, görüntü altyazılama ardışık zaman adımlarıyla mümkündür.

Çoktan Bire(Many to One): Birden fazla girdiyle tek bir çıkışın üretildiği yapılardır. Duygu sınıflandırma örnek gösterilebilir.

Çoktan Çoğa(Many to Many): Birden fazla girdiyle yine birden fazla çıktı üretilmesidir. Burada girdi boyutunun çıktıya eşit olduğu ve olmadığı iki durum söz konusudur. Girdinin çıktıya eşit olduğu durumlara adlandırılmış varlık tanıma örnek gösterilebilir. Makine çevirisi de girdi ve çıktı boyutlarının farklı olduğu çoktan çoğa yapılara örnektir.

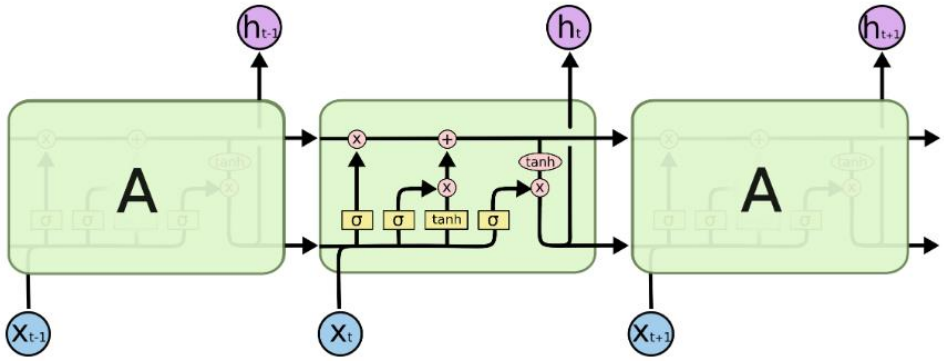
Tekrarlayıcı sinir ağlarının avantajlarının yanı sıra bir çok dezavantajı mevcuttur. En son zaman adımlarına daha çok ağırlık verildiği için uzun zaman adımlarındaki bilgiyi yakalayamayacak, yanlı sonuçlar üretebilecektir. Ayrıca veri setinin büyük olması ağırlıkların birden büyümesine ve hesaplama yükünün artmasına sebep olabilmektedir.

Ayrıca bu mimaride geri yayılım kullanıldığı için kaybolan gradyan sorunuyla karşılaşılmaktadır. Her bir zaman adımı, ileri beslemeli ağda tüm bir katmana eşdeğer olduğu için kaybolan gradyan sorunu daha sert gözlemlenir. 50 zaman adımı içeren bir veriyi eğitmek istediğimizde model 50 katmanlı ileri beslemeli bir ağı eğitmiş gibi davranacaktır. Bu durum üstel olarak daha küçük gradyanlara ve bilginin bozulmasına neden olur. Başka bir deyişle uzun süreli tekrarlayan bağlantılar ağın daha küçük türev değerlerine ulaşmasına sebep olacak ve gradyanlar yok olacaktır.

Gradyan kırpma, bu problemi çözmek için kısa vadede faydalı bir bakış açısı sağlar. Degrade vektörü yeniden ölçeklendirilerek belirli bir eşiği takip etmesi sağlanabilir. Geri yayılımda tüm ağırlıkların güncellenmemesi, ağırlık başlatma gibi yöntemler patlayan ve kaybolan gradyanlar için birer çözüm olarak değerlendirilmektedir. Ayrıca kaybolan gradyan problemini çözmek için RNN temelli bir çok gelişmiş sinir ağı mimarisi mevcuttur.

5.2. Uzun Kısa Vadeli Bellek(LSTM)

Uzun Kısa Vadeli Bellek kısaca LSTM, uzun süreli bağımlılıkları yakalayabilen bir RNN türüdür (Gers ve ark., 2000). LSTM'in RNN'den farklı olarak bir çok bileşeni mevcuttur. Bunlardan en önemlileri bazı kapı hücreleridir.



Şekil 5.2. LSTM Mimarisi

Üstteki yatay çizgi *hücre durumunu* temsil etmektedir. Hücre durumu LSTM'in izlediği yol olarak tanımlanabilir. LSTM'de kapılar hangi bilginin geçip geçmeyeceğine karar veren yapılardır. Şekildeki ilk x'i gösteren sigmoid, hücre durumundan hangi bilgilerin atılacağını ifade eder. Buna *unutma kapısı* denir. Daha sonra aday adı verilen vektör tanh yardımıyla elde edilir. *Aday hücre durumu* mevcut hücrenin durumuna eklenecek bilgi adaylarıdır. Eklenecek ve unutulacak bilgiler

yine hücre durumunda güncellenir. Buna da *güncelleme kapısı* denir. Son olarak şekildeki son sigmoid çalıştırılarak hücre durumunun hangi bölümlerinin alınacağı kararlaştırılır. *Çıkış kapısı* olarak adlandırılan bu bölümde bilginin kullanılma miktarı kontrol edilerek hücre durumu ile mevcut zaman adımındaki girdinin bir fonksiyonu oluşturularak çıktı elde edilir(Olah, 2015).

5.3. Geçitli Tekrarlayan Birim(GRU)

GRU yani Geçitli Tekrarlama Birimi, uzun süreli bağımlılıkları işleyebilme kabiliyetine sahip bir RNN türüdür. LSTM yapısına alternatif olarak geliştirilmiştir. Daha az parametre ve karmaşıklığa sahiptir (Chung ve ark., 2014)

Hücre durumu güncellemesi unutma ve giriş kapılarının bir arada yer aldığı kısıma *güncelleme kapısı* denir. *Sıfırlama kapısı* geçmiş durum ve mevcut durumu kullanarak bilginin ne kadar kullanılacağını belirler. Bu iki kapıdan gelen bilgiye göre hücrenin güncellenip güncellenmemesine karar verilerek çıkış *kapısı* ile çıktı elde edilir.

GRU, LSTM'den farklı olarak uzun süreli bağımlılıkları daha iyi yakalar, daha basit ve az bileşene sahip olduğu için hesaplama süresi nispeten azdır.

5.4. Çift Yönlü Tekrarlayan Sinir Ağları(BRNN)

Klasik RNN'lerin aksine BRNN'lerde öğrenme çift yönlü olarak gerçekleşir. Çift yönlü RNN'ler ileri ve geri yönlü olmak üzere iki bileşen içerir. İleri yönlüde klasik RNN mimarisinde olduğu gibi mevcut girdi ve önceki zaman adımından beslenerek bir sonraki adım tahmin edilir. Geri yönlüde ise sıralı veri sondan başa doğru hareket eder. Sonrasında bu iki bileşenin çıktıları birleştirilerek tahmin sonuçları üretilir. Doğal dil işleme çalışmalarında sıklıkla kullanılan BRNN'ler, mevcut zaman adımından önceki ve sonraki gözlemlerden yararlanarak bağlamsal konularda tahmine yardımcı olur.

5.5. Derin Tekrarlayan Sinir Ağları(DRNN)

Derin tekrarlayan sinir ağları daha çok gizli katman ve karmaşık yapıya sahip bir RNN varyasyonudur. Katman sayısının daha fazla olması bazı problem türleri ve veri yapılarını iyi işleyebilme avantajı sağlamaktadır.

6. MODEL ÇIKTILARI

Çalışmada geleneksel zaman serisi yaklaşımları ve sinir ağı mimarilerinden Tekrarlayan Sinir Ağları kullanılarak Anadolu Sigorta (ANSGR) şirketine ait 15.06.2022 ile 13.01.2023 tarihleri arasındaki hisse senedi verileri, Borsa İstanbul tarafından veri dağıtıcı olarak yetkilendirilen Foreks'ten elde edilerek zaman serisi tahmin modelleri geliştirilmiştir (https://www.foreks.com/sembol-detay/H2010/ansgr/anadolu-sigorta).

Özellikle Tekrarlamalı Sinir Ağı mimarisi (RNN) ve türevleri kullanılarak diğer modellerle performans kıyaslaması gerçekleştirilmiştir.

Çalışmada açık kaynak kodlu Python programlama dili kullanılarak çeşitli analizler yapılarak modeller geliştirilmiştir.

Python'da Numpy, Pandas, Matplotlib, Sklearn, Tensorflow, Keras gibi bir çok kütüphane ve metot kullanılarak veri manipülasyonu ve model kurma aşamaları gerçekleştirilmiştir.

Geleneksel zaman serisi analiziyöntemleri Python'ın Statsmodels kütüphanesi yardımıyla elde edilmiştir(Seabold, S., ve Perktold, J. 2010).

Model verisi % 80'i eğitim, % 20'si test kümesi olacak şekilde dizayn edilmiş, kayıp fonksiyonu olarak hata kareler ortalaması karekökü kullanılmıştır.

6.1. Veri Setinin İncelenmesi

```
>>> df.head()
        Tarih  Şimdi  Açılış  Yüksek  Düşük    Hacim
0  15.06.2022   5.81    5.89    5.90   5.80     1.37
1  16.06.2022   5.74    5.80    5.84   5.70   891.90
2  17.06.2022   5.72    5.73    5.73   5.66     1.06
3  20.06.2022   5.73    5.72    5.75   5.67     1.18
4  21.06.2022   5.72    5.73    5.77   5.72   597.15
```

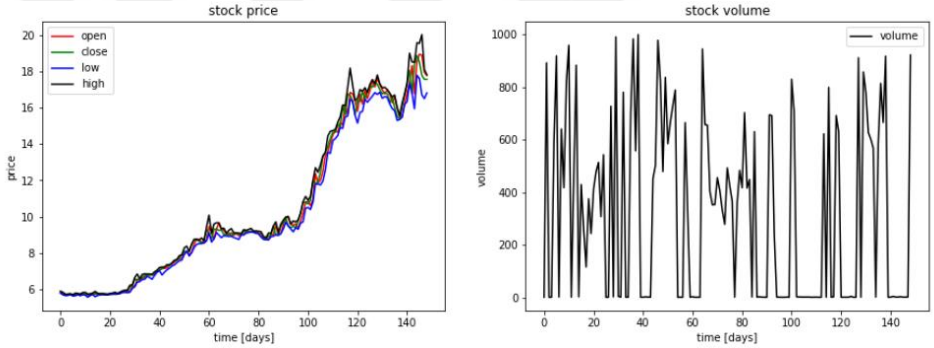
Şekil 6.1. Hisse Senedi Veri Setine Ait İlk 5 Gözlem Değeri

Veri seti hisse senedine ait Şimdi(Kapanış), Açılış, Yüksek, Düşük, Hacim ve Fark gibi bir çok değişkeni barındırmaktadır. Veri seti 5 değişken ve 150 gözlem değerinden oluşmaktadır.

```
>>> df.describe()
              Şimdi       Açılış       Yüksek        Düşük        Hacim
count   149.000000  149.000000  149.000000  149.000000  149.000000
mean     10.378322   10.343154   10.602483   10.083087  319.863356
std       4.169296    4.178519    4.342109    3.960675  342.719590
min       5.680000    5.670000    5.710000    5.570000    1.000000
25%       6.790000    6.800000    6.860000    6.720000    1.460000
50%       9.090000    9.090000    9.250000    8.940000  257.410000
75%      14.650000   14.650000   14.800000   14.300000  630.080000
max      18.870000   18.940000   20.020000   17.780000  998.980000
```

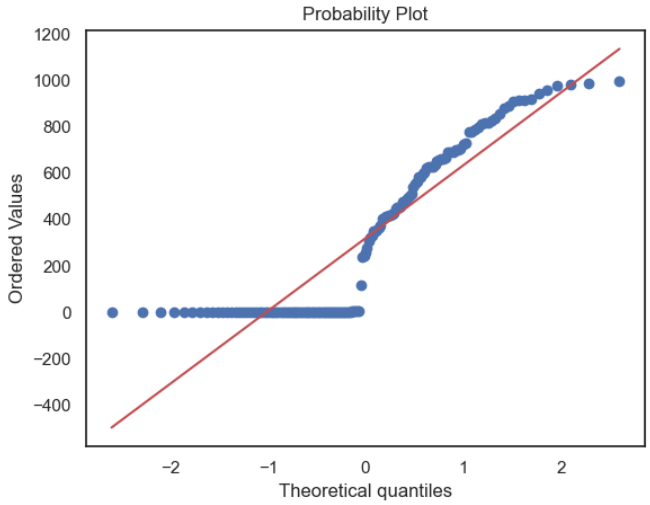
Şekil 6.2. Hisse Senedi İstatistikleri

Veri setinde yer alan değişkenlerin bazı istatistiksel bilgileri şekil 6.2'de verilmiştir. Görüldüğü üzere "Şimdi" adlı değişken yani kapanış tutarının ortalaması 10.37, standart sapması 4.16, minimum değeri 5.68, maksimum değeri 18.87'dir. Ayrıca 1, 2 ve 3. kartillere ait değerleri de kod çıktısında yer almaktadır.



Şekil 6.3. Hisse Senedi Zamana Bağlı Fiyat ve Hacmi

Şekil 6.3'de hisse senedinin değeri zaman kırılımında artış gösterirken hacminin epey inişli çıkışlı olduğu gözlemlenmektedir.



Şekil 6.4. Q-Q Grafiği

Q-Q grafiğinde gözlemler düz bir doğu etrafında dağılım göstermediği için verilerin normal dağılmadığı çıkarımında bulunulabilir.

6.2. Zaman Serisi Ayrıştırması

AR, MA, ARIMA, SARIMA gibi istatistiksel ekonometrik zaman serisi modellerinde durağanlık, mevsimsellik, trend, döngüsel ve düzensiz rastgele hareket başlıkları incelenerek veri setine uygun modeller kurulmaktadır.

6.2.1. Durağanlık

Durağanlık en basit tabirle, bir zaman serisini oluşturan sürecin istatistiksel özelliklerinin zamana bağlı olarak değişime uğramaması anlamına gelmektedir. İlgili fonksiyonun değişim şekli sabittir. Durağanlık bazı testlerle veya grafikler üzerinden kolaylıkla gözlemlenebilir.

Dickey-Fuller testi bir serinin durağan olup olmadığını tespit etmek için kullanılan bir testtir.

H_0: Seri durağan değildir.

H_1: Seri durağandır.

Python'da bu test aşağıdaki gibi bir fonksiyon tanımlanarak yapılabilir.

```python
def dicky_fuller_test(x):
    result = adfuller(x)
    print('ADF Statistic: %f' % result[0])
    print('p-value: %f' % result[1])
    print('Critical Values:')
    for key, value in result[4].items():
        print('\t%s: %.3f' % (key, value))
    if result[1]>0.05:
        print("Fail to reject the null hypothesis (H0), the data is non-stationary")
    else:
        print("Reject the null hypothesis (H0), the data is stationary.")
```

Şekil 6.5. Durağanlık Testi

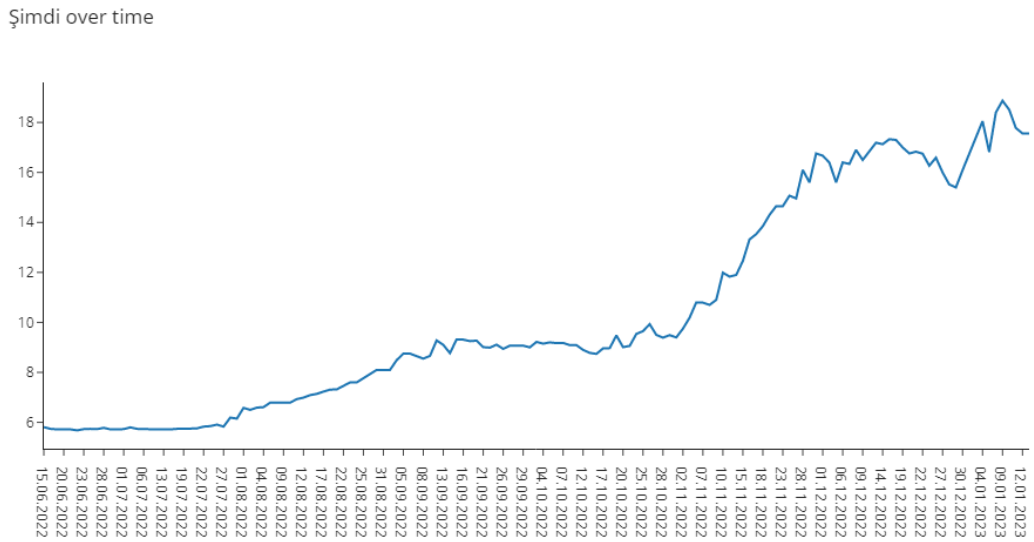
Tahmin edilmek istenen değişken veri setindeki 'Şimdi' değişkenidir.

```python
dicky_fuller_test(df['Şimdi'])
```

```
ADF Statistic: 0.514485
p-value: 0.985307
Critical Values:
        1%: -3.476
        5%: -2.881
        10%: -2.577
Fail to reject the null hypothesis (H0), the data is non-stationary
```

Şekil 6.6. Durağanlık Test Sonuçları

p-value değerinin 0,05'ten büyük olması yokluk hipotezini reddedemediğimiz anlamına gelir. Yani serinin durağan olmadığına karar verilir.



Şekil 6.7. Zamana Göre Değişim

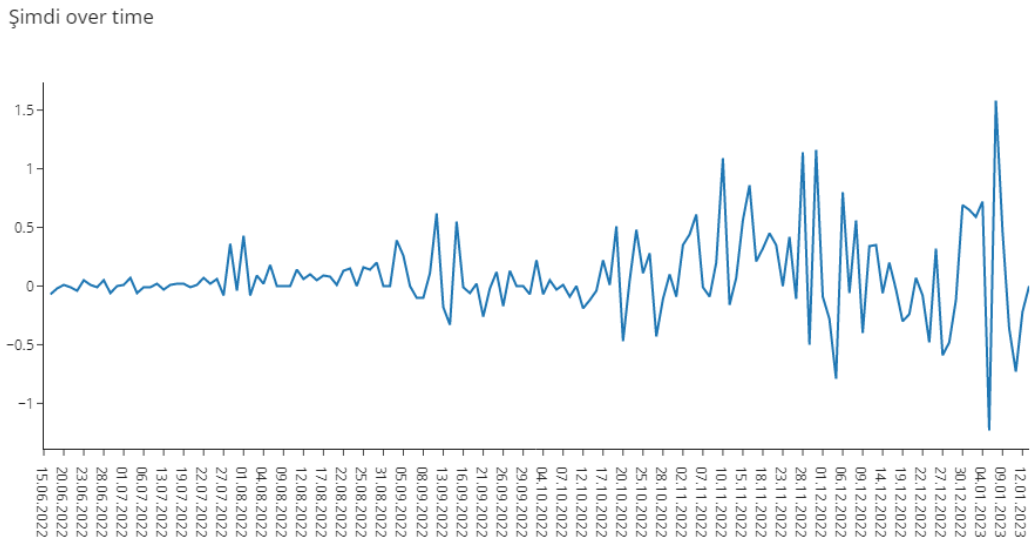
Bu durum zamana göre çizgi grafiği (Şekil 6.7) ile de gözlemlenebilir. Yine aynı şekilde zaman içinde serinin değişimi durağan değildir yorumu yapılır.

Serinin durağanlaştırılması çeşitli tahmin yöntemlerinin uygulanabilirliği ve performansı açısından büyük önem arz etmektedir. Bu nedenle serinin durağanlaştırılması fark alma işlemiyle sağlanır.

```
df['Şimdi_diff']=df['Şimdi']-df['Şimdi'].shift(1)
fig = go.Figure([go.Scatter(x=df.index,y=df.Şimdi)])
fig.update_layout(
    autosize=False,
    width=1000,
    height=500,
    template='simple_white',
    title='Şimdi over time ')
fig.show()
```

Şekil 6.8. 1 Zaman Adımı Gecikmenin Elde Edilmesi

Gecikme sayısı 1 alınarak yeni bir değişken oluşturulmuş ve aşağıdaki gibi grafiği çizdirilmiştir.

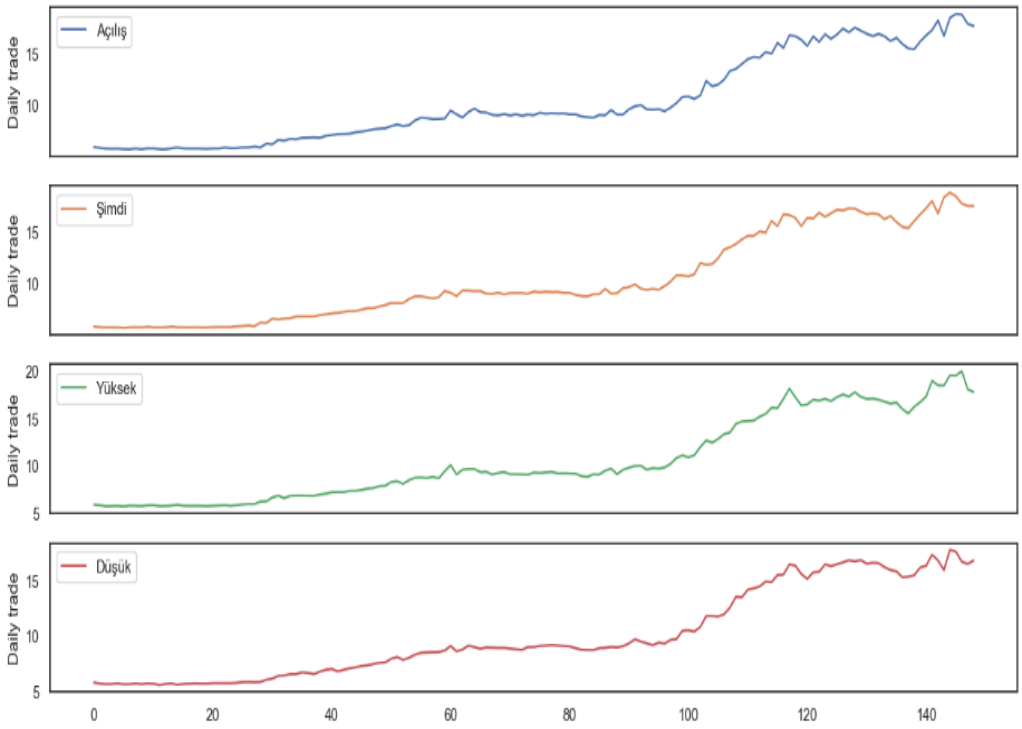


Şekil 6.9. Farkı Alınan Seri

Şekil 6.9.'da görüldüğü üzere seri, farkı alınarak trend ve mevsimsel etkilerden arındırılmıştır.

6.2.2. Mevsimsellik ve Trend

Zaman serisinin belli bir davranışı belirli periyotlarla tekrar etmesi durumu mevsimsellik olarak ifade edilir. Trend ise bir zaman serisinim uzun vadedeki artış ya da azalışıdır.

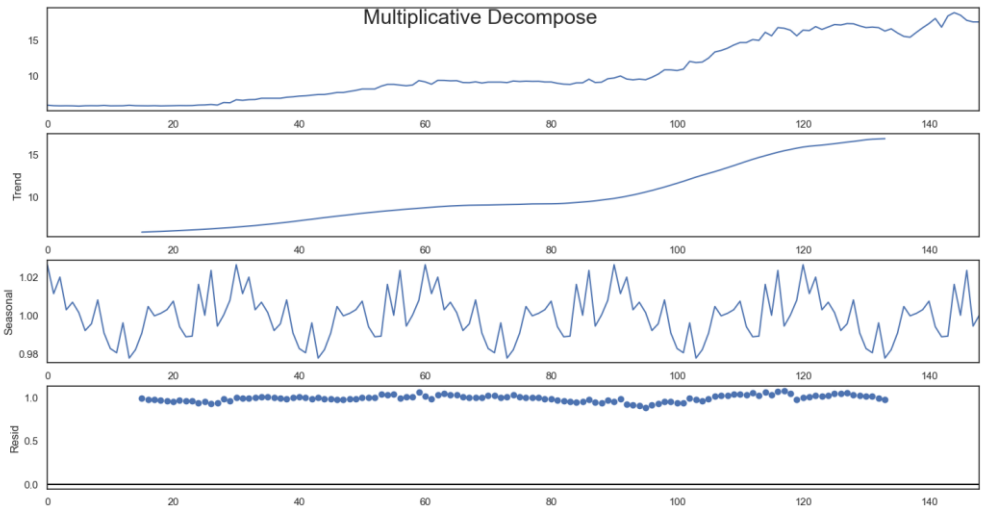


Şekil 6.10. Zamana Bağlı Değişim

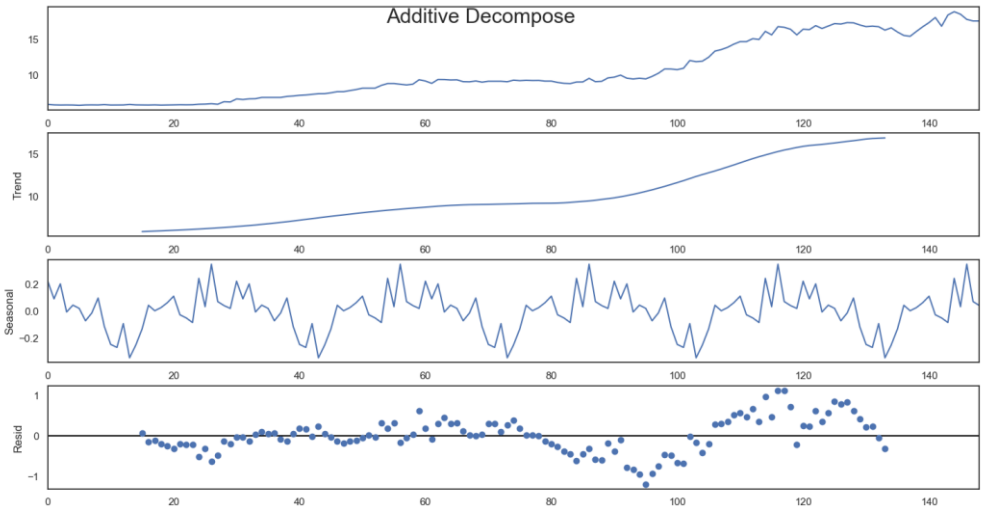
Hisse senedinin açılış, kapanış(Şimdi), yüksek ve düşük değerleri zaman periyodunda incelendiğinde artış trendinde olduğu gözlemlenmektedir. Veri seti 150 güne ait gözlem değeri içerdiği için yıllık periyotta mevsimsel etkinin gözlemlenmesi doğru olmayacaktır. Grafiğe göre 150 günlük zaman diliminde herhangi bir mevsimsel etki yoktur.

Zaman serileri modelleri, belirli bir zaman periyodunda gözlenen değişimleri anlamaya ve tahmin etmeye yarayan matematiksel yapılardır. İki tip zaman serileri modeli vardır:

- Çarpımsal model
- Toplamsal model



Şekil 6.11. Çarpımsal Model



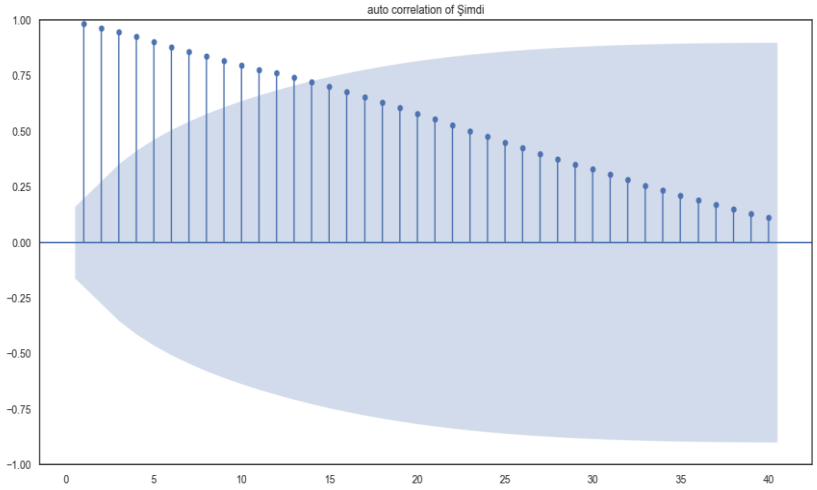
Şekil 6.12. Toplamsal Model

Statsmodels ve dateutil kütüphaneleri kullanılarak çizdirilen bu grafiklere bakılarak serinin toplamsal ya da çarpımsal olarak modellenmesine karar verilebilir. Belirli bir zaman aralığında mevsimsellik ve artıklardaki değişim trendden bağımsızsa model toplamsal, değilse çarpımsal olarak modellenebilir. Bizim veri setimize uygun modelin toplamsal olduğunu gözlemliyoruz.

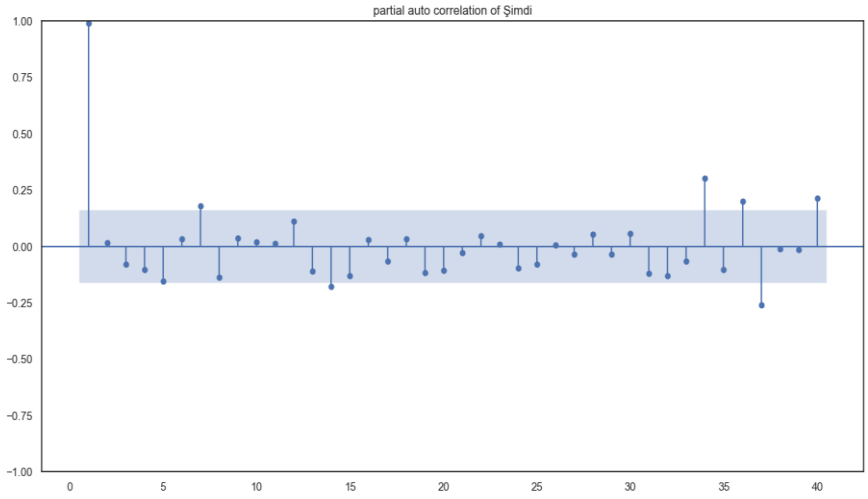
6.2.3. Otokorelasyon

Regresyon analizinde sıklıkla raslanan hata terimlerinin ilişkili olmaması varsayımı otokorelasyon olarak ifade edilmektedir.

Otokorelasyon ve kısmi otokorelasyon grafikleri incelendiğinde serinin otokorelasyona sahip olduğunu gözlemliyoruz.



Şekil 6.13. Otokorelasyon



Şekil 6.14. Kısmi Otokorelasyon

Yani hataların zaman içinde birbirine bağımlı olduğu söylenebilir. Hata terimlerinin ilişkili olması çözülmesi gereken bir problemdir. AR ve MA süreçlerinde kullanılan terimlerin modele dahil edilmesine karar vermek için başlangıç niteliğinde bilgi sağlamaktadır (Rehal, 2022).

6.2.4. AR Modeli

AR yöntemiyle önceki zaman adımlarındaki gözlemlerin doğrusal bir kombinasyonu ile tahmin işlemi yapılır. Trend ve mevsimsellik içermeyen tek değişkenli zaman serileri için uygundur.

AutoReg Model Results

Dep. Variable:	Şimdi	No. Observations:	119
Model:	AutoReg(10)	Log Likelihood	-3.470
Method:	Conditional MLE	S.D. of innovations	0.250
Date:	Mon, 07 Aug 2023	AIC	30.939
Time:	23:43:39	BIC	63.236
Sample:	10	HQIC	44.037
	119		

	coef	std err	z	P>|z|	[0.025	0.975]
const	-0.0174	0.114	-0.152	0.879	-0.241	0.206
Şimdi.L1	0.7028	0.101	6.940	0.000	0.504	0.901
Şimdi.L2	0.2476	0.122	2.022	0.043	0.008	0.487
Şimdi.L3	0.0678	0.128	0.530	0.596	-0.183	0.319
Şimdi.L4	0.1736	0.135	1.282	0.200	-0.092	0.439
Şimdi.L5	0.0469	0.148	0.317	0.752	-0.244	0.338
Şimdi.L6	-0.0808	0.157	-0.514	0.607	-0.389	0.227
Şimdi.L7	-0.1071	0.158	-0.680	0.496	-0.416	0.202
Şimdi.L8	0.2140	0.157	1.360	0.174	-0.094	0.522
Şimdi.L9	-0.2503	0.160	-1.567	0.117	-0.563	0.063
Şimdi.L10	-0.0060	0.122	-0.049	0.961	-0.245	0.233

Şekil 6.15. Otoregresif Model Özeti-1

Model çıktılarına ait yukarıdaki şekilde modellenecek değişkenin yani bağımlı değişkenin "Şimdi" olduğu görülmektedir. Model kısmındaki AutoReg(10) ifadesi de 10 adımlık bir zaman gecikmesini temsil eder(p parametresi 10'dur). Çıktıda modelleme tarihi, zamanı, tahmin yöntemi , gözlem sayısı gibi bilgiler de yer almaktadır. Akaike, Bayes ve Hannan-Quinn Bilgi kriterleri de modelin uygunluğunu değerlendirmek için birer ölçüttür. Daha düşük değerler modelin veriye daha uygun olduğunu gösterir.

İkinci kısımda gecikme sayıları baz alınarak elde edilen regresyon denklemleri yer almaktadır. L1, L2, L3.. ifadeleri gecikme sayısını ifade etmektedir. Önceki zaman adımlarından modelin ne kadar etkilendiğini gösterir. Katsayılar gecikmelerin ağırlığını temsil eder. Standart sapmanın düşük olması, modelin kesinliği konusunda olumlu bir görüş edinilmesini sağlar. Z istatistiği ise ilgili katsayının anlamlılığı yönünde görüş sağlar. P(z) değeri de katsayının anlamlılığı konusunda bilgi sağlar. 0.05'ten küçükse, katsayının anlamlı olduğu anlaşılır.

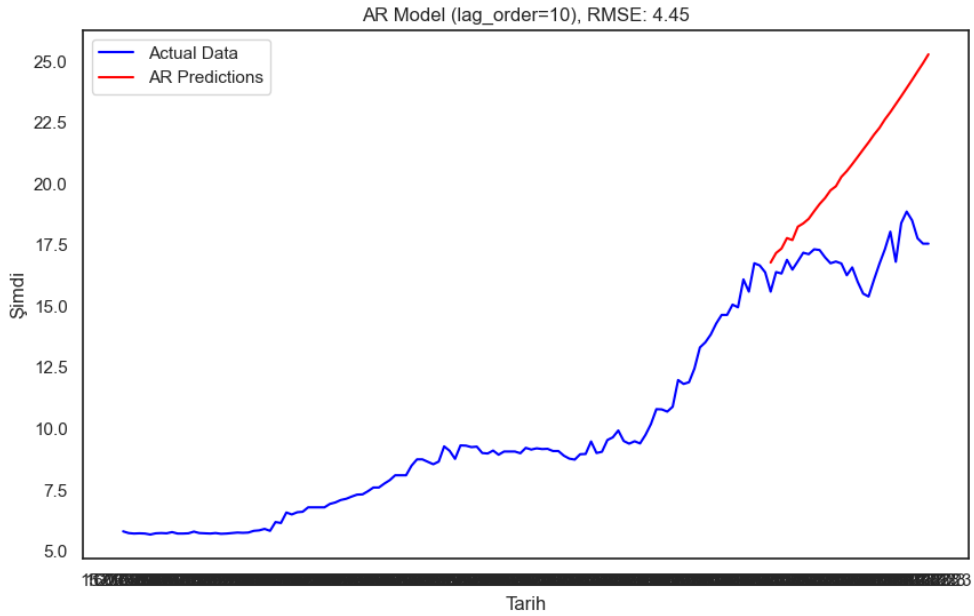
Roots

	Real	Imaginary	Modulus	Frequency
AR.1	-1.1295	-0.0000j	1.1295	-0.5000
AR.2	-0.8474	-0.7966j	1.1631	-0.3799
AR.3	-0.8474	+0.7966j	1.1631	0.3799
AR.4	0.0079	-1.1800j	1.1800	-0.2489
AR.5	0.0079	+1.1800j	1.1800	0.2489
AR.6	0.9849	-0.0000j	0.9849	-0.0000
AR.7	1.0932	-0.0000j	1.0932	-0.0000
AR.8	0.7891	-1.0427j	1.3076	-0.1469
AR.9	0.7891	+1.0427j	1.3076	0.1469
AR.10	-42.7681	-0.0000j	42.7681	-0.5000

Şekil 6.16. Otoregresif Model Özeti-2

Otoregresif model özetinin ikinci kısmında (Şekil 6.16) da modellerin karakteristik özellikleri yer almaktadır. Karakteristik kökler tablosu model hakkında bilgi sağlar. AR parametresinin 10 zaman adımı değişimi

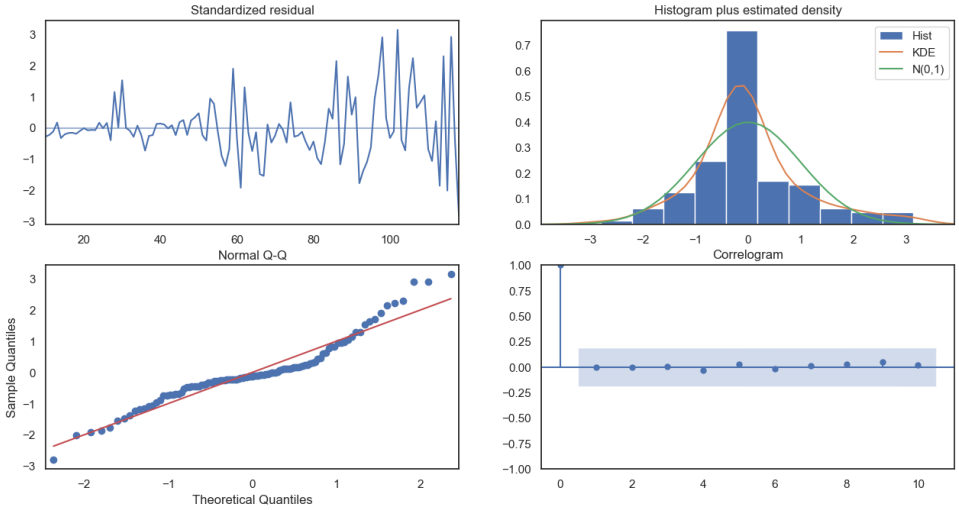
Gerçekleşen ve tahmin edilen gözlemlere ilişkin grafik aşağıdaki gibidir.



Şekil 6.17. AR Modeli Gözlem-Tahmin Değerleri

Grafikten de anlaşılacağı üzere modelin veriyi tahmin etme başarısı düşüktür. Kırmızı renkle ifade edilen tahmin değerleri, mavi renkle gösterilen gerçek değerlere

benzer eğilim göstermemiştir. Model ancak zaman serisindeki genel trendi yakabilmiştir. RMSE değeri 4.45'tir.



Şekil 6.18. AR Modeli Kalıntı Analizi

Kod çıktılarında kalıntılara ilişkin 4 grafik yer almaktadır. İlk grafikte kalıntıların sıfır etrafında rasgele dağılmaması ve huni şeklinde görülmesi varyansın sabit olmadığını (heteroskedastisite) göstermektedir. İkinci grafikte serinin sağa çarpık olduğu, ortalamanın sıfır olmadığı ve kalıntıların simetrik olmadığı gözlemlenmektedir. Q-Q grafiğinde kalıntıların normal dağılıma uygunluğu sınanmaktadır. Q-Q grafiği için kuyruklarda sapan değerlerin olduğu görülmektedir. Son grafikte kalıntıların otokorelasyonu gözlemlenebilir, kalıntıların istatistiksel olarak anlamlı olmayan mavi bölgenin dışında konumlanması korelasyonun varlığına işarettir.

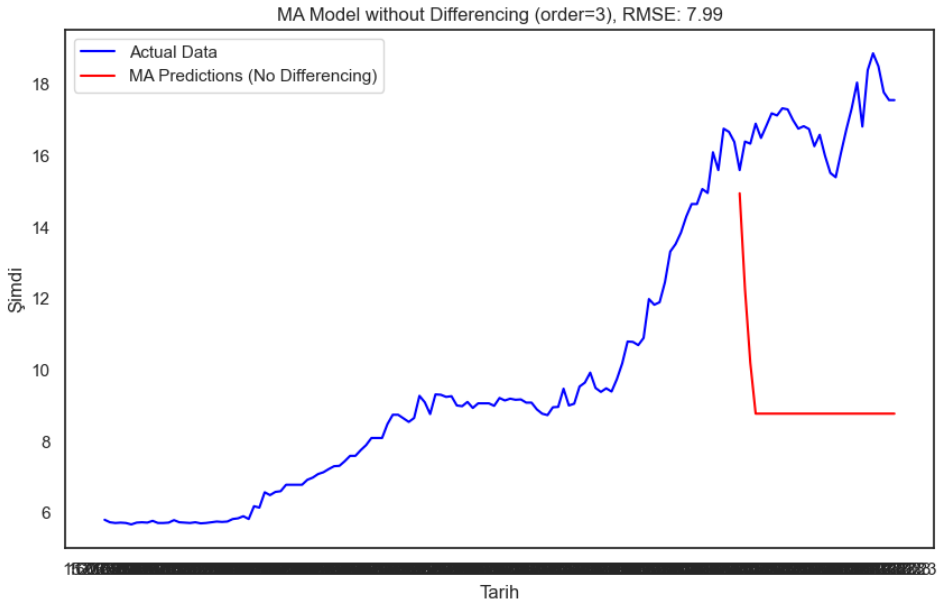
6.2.5. MA Modeli

MA modelinden elde edilen özet çıktı aşağıdaki gibidir.

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                   Şimdi   No. Observations:                  119
Model:                 ARIMA(0, 0, 3)   Log Likelihood                -131.510
Date:                Tue, 08 Aug 2023   AIC                            273.021
Time:                        01:24:20   BIC                            286.917
Sample:                             0   HQIC                           278.664
                                  - 119
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          8.7814      0.416     21.085      0.000       7.965       9.598
ma.L1          1.5459      0.068     22.856      0.000       1.413       1.678
ma.L2          1.5226      0.080     19.117      0.000       1.366       1.679
ma.L3          0.7591      0.069     11.069      0.000       0.625       0.894
sigma2         0.5113      0.071      7.195      0.000       0.372       0.651
===================================================================================
Ljung-Box (L1) (Q):                  30.60   Jarque-Bera (JB):                77.19
Prob(Q):                              0.00   Prob(JB):                         0.00
Heteroskedasticity (H):               2.68   Skew:                             1.49
Prob(H) (two-sided):                  0.00   Kurtosis:                         5.58
===================================================================================
```

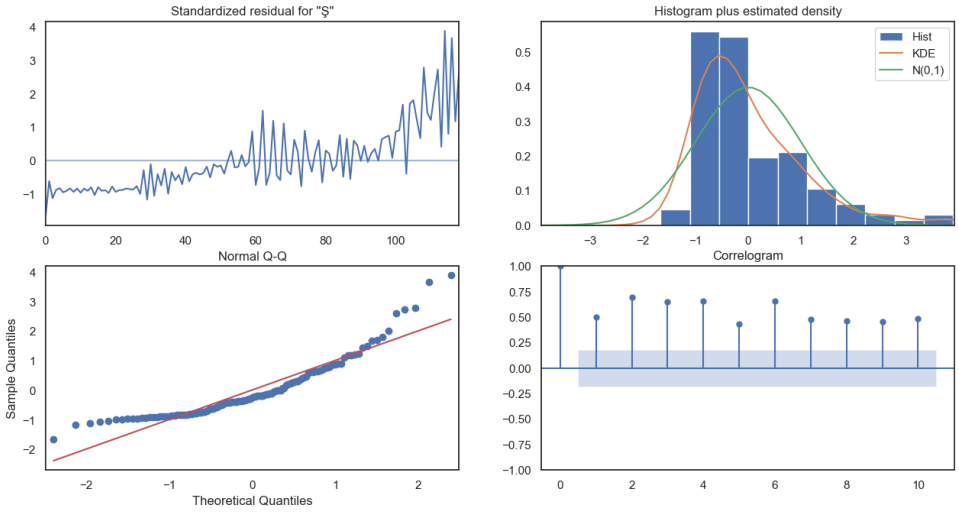
Şekil 6.19. Hareketli Ortalama Modeli Özeti

MA modelinde 3 zaman adımlı gecikme kullanılmıştır. AIC-BIC skorları AR modeline göre daha yüksek seyretmiştir. Buna göre AR modelinin daha iyi performans gösterdiği söylenebilir.



Şekil 6.20. MA Modeli Gözlem-Tahmin Değerleri

Tahmin ve mevcut gözlemlere ait grafik incelendiğinde modelin veriyle uyumlu olmadığı sonucu çıkarılabilir. MA modelindeki hata metriği(RMSE), AR modeline göre daha yüksektir.



Şekil 6.21. MA Modeli Kalıntı Analizi

Kalıntı analizinde ilk modele göre daha düşük bir performans sergilendiği açıkça gözlemlenmektedir. Varyans sabit değildir, kalıntıların doğrusallığından söz edilemez ve otokorelasyon mevcuttur.

6.2.6. ARMA Modeli

SARIMAX Results

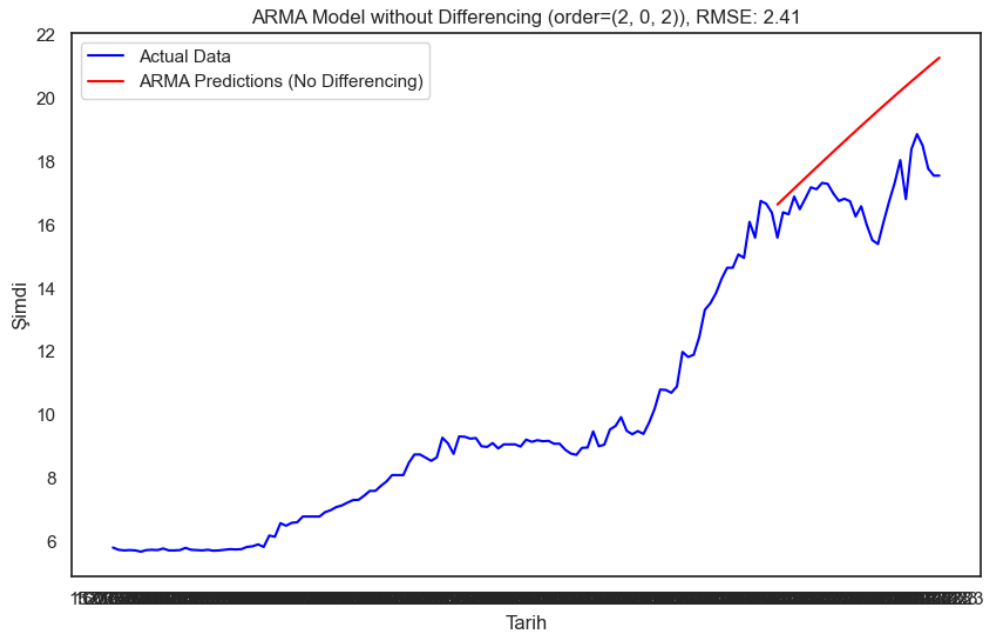
Dep. Variable:	Şimdi	No. Observations:	119
Model:	ARIMA(2, 0, 2)	Log Likelihood	-12.929
Date:	Tue, 08 Aug 2023	AIC	37.857
Time:	01:39:08	BIC	54.532
Sample:	0	HQIC	44.628
	- 119		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
const	16.0893	17.826	0.903	0.367	-18.850	51.028
ar.L1	1.9990	0.006	325.405	0.000	1.987	2.011
ar.L2	-0.9992	0.006	-179.630	0.000	-1.010	-0.988
ma.L1	-1.1816	0.105	-11.263	0.000	-1.387	-0.976
ma.L2	0.1892	0.087	2.181	0.029	0.019	0.359
sigma2	0.0676	0.008	8.020	0.000	0.051	0.084

Ljung-Box (L1) (Q):	0.09	Jarque-Bera (JB):	87.23
Prob(Q):	0.77	Prob(JB):	0.00
Heteroskedasticity (H):	12.54	Skew:	1.30
Prob(H) (two-sided):	0.00	Kurtosis:	6.29

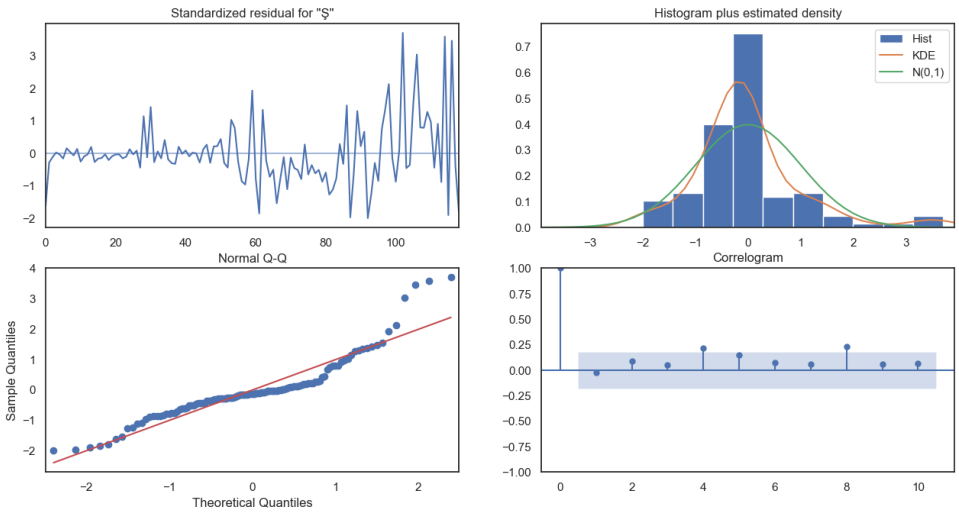
Şekil 6.22. ARMA Modeli Özeti

ARMA modelinde AR ve MA'dan gelen p ve q parametreleri birlikte kullanılır. AIC-BIC ve HQIC metrikleri AR modeliyle neredeyse benzerdir.



Şekil 6.23. ARMA Modeli Gözlem-Tahmin Değerleri

ARMA model grafiğinde modelin veriyi tahmin gücünün zayıf olduğu, yalnızca trendi yakalamada başarılı olduğu görülmektedir. Ayrıca hata metriği RMSE bu 3 model içinde en düşüktür.



Şekil 6.24. ARMA Modeli Kalıntı Analizi

Kalıntı analizinde yine kalıntıların normal dağılmadığı ve sağa çarpık bir eğilimde olduğu, otokorelasyon ve değişen varyansın mevcut olduğu görlmektedir.

6.2.7. ARIMA Modeli

ARIMA modeli, üç faktörden oluşur: otoregresif (AR) terimi, düzleştirme (I) terimi ve hareketli ortalama (MA) terimi. AR terimi verilerin geçmişteki değerlerinin etkisini dikkate alır, I terimi verilerin düzleştirilmesini sağlar ve MA terimi ise verilerdeki dalgalanmaların tahmin edilmesini sağlar.

SARIMAX Results

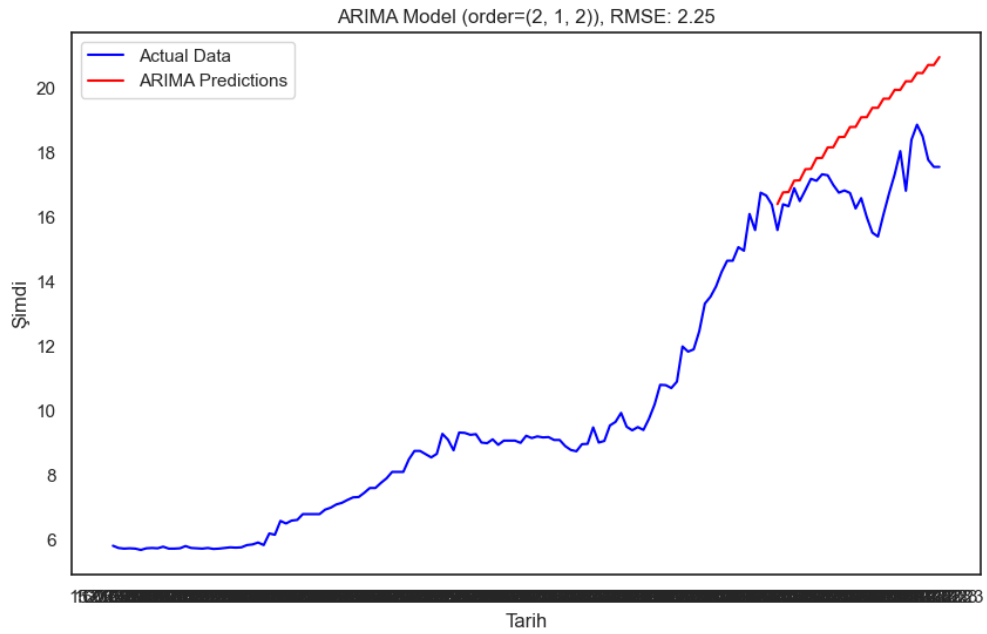
Dep. Variable:	Şimdi	No. Observations:	119
Model:	ARIMA(2, 1, 2)	Log Likelihood	-8.519
Date:	Tue, 08 Aug 2023	AIC	27.038
Time:	01:49:10	BIC	40.892
Sample:	0	HQIC	32.663
	- 119		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.0046	0.076	-0.061	0.951	-0.154	0.145
ar.L2	0.9733	0.086	11.278	0.000	0.804	1.142
ma.L1	0.0271	0.124	0.218	0.827	-0.217	0.271
ma.L2	-0.8412	0.156	-5.398	0.000	-1.147	-0.536
sigma2	0.0668	0.006	10.411	0.000	0.054	0.079

Ljung-Box (L1) (Q):	4.31	Jarque-Bera (JB):	49.55
Prob(Q):	0.04	Prob(JB):	0.00
Heteroskedasticity (H):	17.66	Skew:	0.77
Prob(H) (two-sided):	0.00	Kurtosis:	5.77

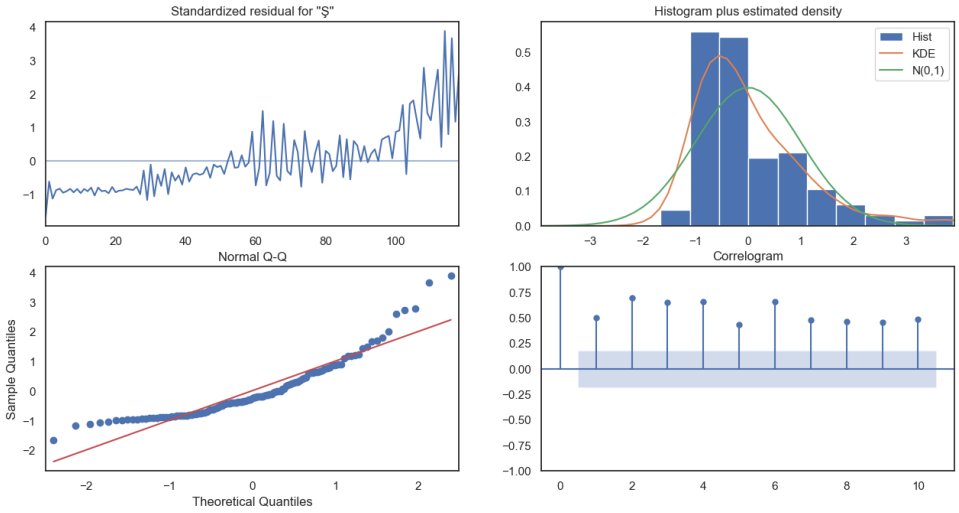
Şekil 6.25. ARIMA Modeli Özeti

ARIMA modeline ait özet bilgiler şekildeki gibidir. Model için kullanılacak optimum parametreler (2, 1, 2) olarak tespit edilmiştir.



Şekil 6.26. ARIMA Modeli Gözlem-Tahmin Değerleri

ARIMA Modeli tahmin başarısı yönünden diğer 3 modele kıyasla daha iyi performans göstermiştir. En düşük RMSE değerine sahiptir.



Şekil 6.27. ARIMA Modeli Kalıntı Analizi

ARIMA Modelinde kalıntı analizine ilişkin grafikler ARMA ile benzerlik göstermektedir.

6.2.8. SARIMA Modeli

SARIMAX Results

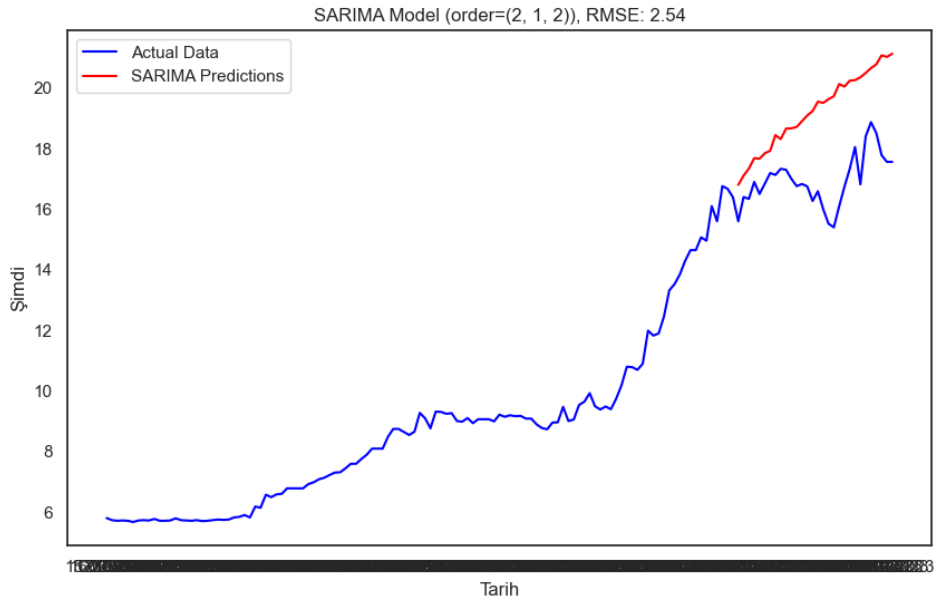
Dep. Variable:	Şimdi	No. Observations:	118
Model:	SARIMAX(2, 1, 2)x(1, 1, [1], 12)	Log Likelihood	-11.687
Date:	Tue, 08 Aug 2023	AIC	37.374
Time:	02:11:37	BIC	55.952
Sample:	0	HQIC	44.902
	- 118		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	1.3233	0.174	7.584	0.000	0.981	1.665
ar.L2	-0.3694	0.175	-2.109	0.035	-0.713	-0.026
ma.L1	-1.5914	0.130	-12.209	0.000	-1.847	-1.336
ma.L2	0.7156	0.127	5.618	0.000	0.466	0.965
ar.S.L12	0.1832	0.271	0.677	0.498	-0.347	0.714
ma.S.L12	-0.9109	0.481	-1.895	0.058	-1.853	0.031
sigma2	0.0631	0.021	2.993	0.003	0.022	0.104

Ljung-Box (L1) (Q):	0.04	Jarque-Bera (JB):	25.92
Prob(Q):	0.83	Prob(JB):	0.00
Heteroskedasticity (H):	12.86	Skew:	0.66
Prob(H) (two-sided):	0.00	Kurtosis:	5.04

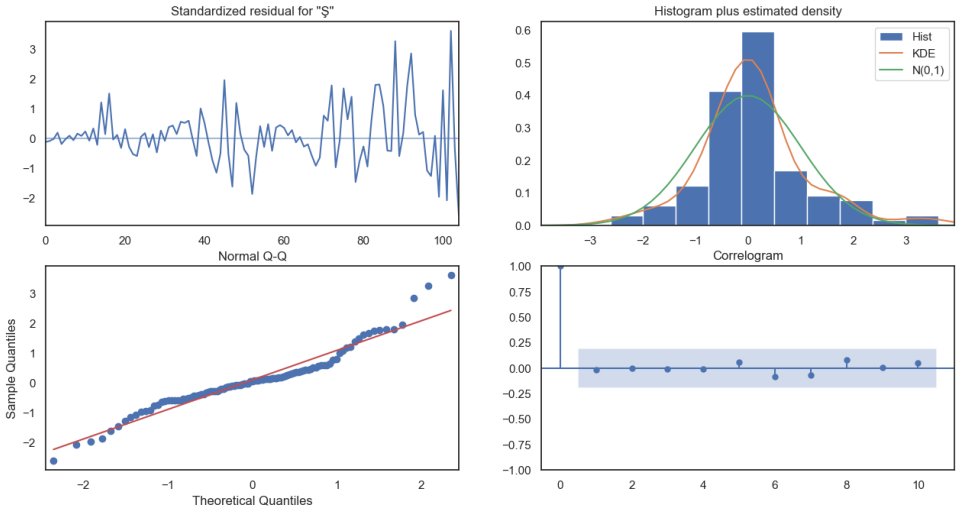
Şekil 6.28. SARIMA Modeli Özeti

SARIMA'ya ait modelleme özeti şekildeki gibidir.



Şekil 6.29. SARIMA Modeli Gözlem-Tahmin Değerleri

Hata metriği RMSE değerinin ARIMA ile aynı olduğu görülmektedir.



Şekil 6.30. SARIMA Modeli Kalıntı Analizi

Kalıntı analizinde yine sapan değerler olduğu, değişen varyans ve otokorelasyonun mevcut olduğu çıkarımlarında bulunulabilir.

6.3. Tekrarlayan Sinir Ağları ile Modelleme

Tekrarlayan sinir ağlarıyla modelleme işlemi bir çok aşamadan oluşmaktadır. Modelleme aşamaları Python'da Keras kütüphanesiyle gerçekleştirilmiştir(Chollet ve ark., 2015). Veri ön işleme aşamasında veri seti 0 ile 1 arasında MinMaxScaler metoduyla ölçeklenmiştir. Bu işlem modelin istikrarı, aşırı uyumu engelleme ve ağırlıkların optimal seviye öğrenilmesi için yapılmaktadır.

Veri seti, modellemeye uygun formata getirildikten sonra % 20 test, % 80 eğitim verisi olacak şekilde ikiye ayrılmıştır. Eğitim verisi üzerinden modelleme yapılarak test verisi üzerinden sınama gerçekleştirilmiştir.

Model kurulma aşamasında kullanılan bazı parametreler aşağıdaki gibidir.

```
batch_size = 1
timesteps = 1
units = 100
nb_epoch = 70
```

Şekil 6.31. RNN Modeli Parametreleri

Eğitim verisini daha ufak alt kümelere bölmek için batch_size kullanılır. Bu parametreye değer olarak 1 atanarak her bir öğrenme adımında(epoch) 1 veri örneği kullanılması sağlanmıştır. Timesteps de aynı şekilde 1 olarak atanmış ve her bir girdinin 1 zaman adımı ilerlemesi sağlanmıştır.

Units olarak nitelendirilen nöron sayısı parametresi her bir katmandaki nöronu ifade eder. Model karmaşıklığını artıracağı için aşırı öğrenme konusunda kontrollü davranılarak atama yapılması gerekir. Eğitim döngüleri parametrei 70 olarak atanmıştır. Bu da modelin eğitileceği eğitim dönemini belirtir.

6.3.1. Standart Tekrarlayan Sinir Ağı (RNN)

RNN modeline ilişkin öncelikle model özeti elde edilmiştir.

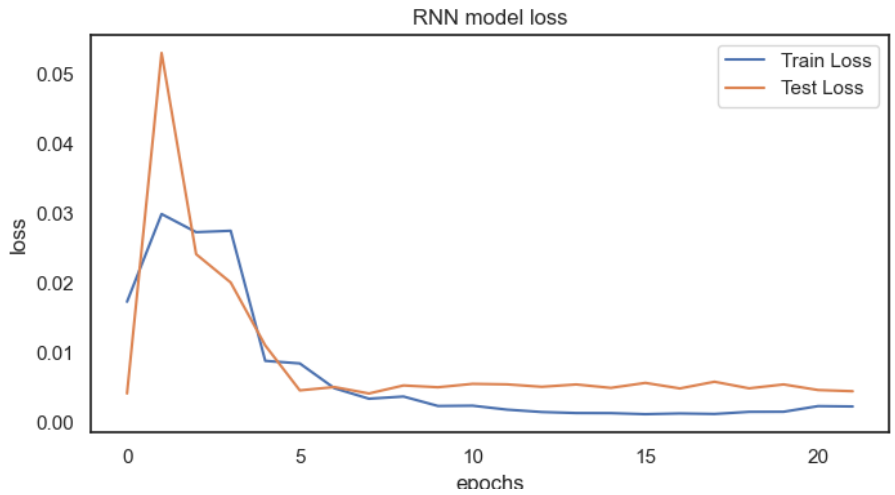
```
>>> model.summary()
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 simple_rnn (SimpleRNN)      (1, 100)                  10500

 dense (Dense)               (1, 1)                    101

=================================================================
Total params: 10,601
Trainable params: 10,601
Non-trainable params: 0
_________________________________________________________________
```

Şekil 6.32. RNN Modeli Özeti

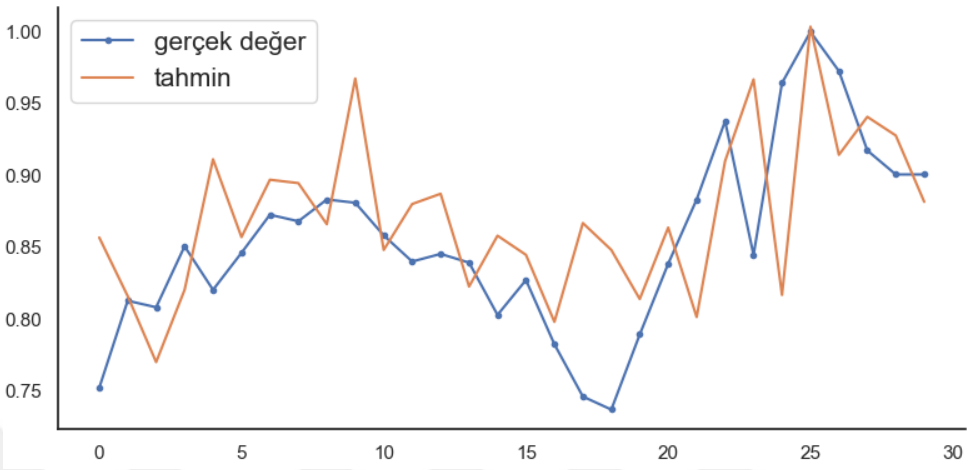
Modellemeye ilişkin özette kullanılan katman türü, çıkış şekli, parametre sayısı vs. metrikler yer almaktadır.



Şekil 6.33. RNN Modeli Eğitim ve Test Hatası

RNN ile kurulan modelin eğitim ve test kümesine ait hata fonksiyonu grafiği(MSE) şekildeki (Şekil 6.33) gibidir. Eğitim ve test hatasının birbirinden ayrıştığı noktada eğitim sürecinin

durdurulması(early stopping) aşırı öğrenmenin önüne geçecektir. RNN modelinde bu durum söz konusu değildir. Eğitim ve test hatası birlikte hareket etmektedir.



Şekil 6.34. RNN Modeli Gerçek ve Tahmin Değerleri

RNN Model performansı şuana kadar en iyi başarıyı gösteren modeldir. Şekil 6.34.'de y ekseni veri noktalarını x ekseni de zamanı temsil etmektedir. Model verideki trendi ve eğilimleri yakalama konusunda başarılıdır.

Çizelge 6.1. RNN modeline ait hata metrikleri

MSE	RMSE	MAE
0.0038	0.0620	0.0474

Modele ilişkin hata metrikleri de yukarıdaki gibidir.

6.3.2. Uzun Kısa Vadeli Bellek (LSTM)

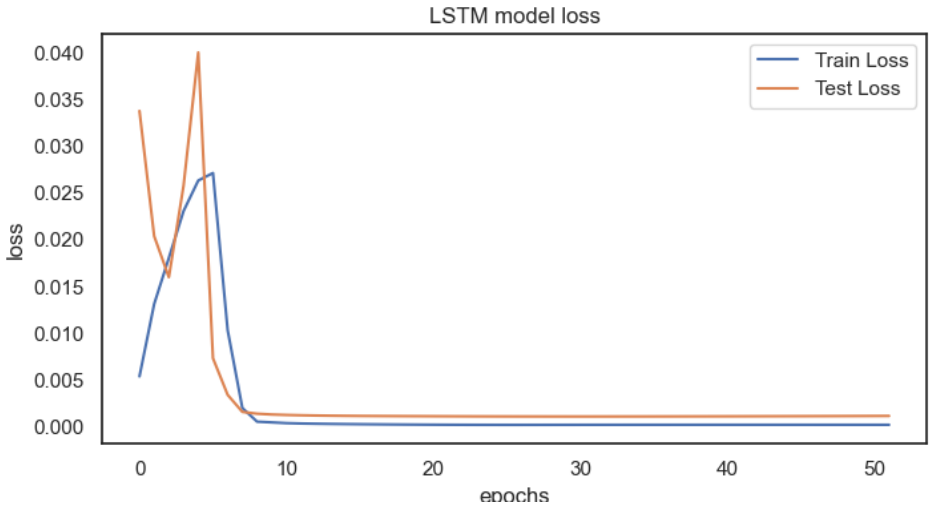
```
>>> model.summary()
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (1, 100)                  42000

 dense (Dense)               (1, 1)                    101

=================================================================
Total params: 42,101
Trainable params: 42,101
Non-trainable params: 0
_________________________________________________________________
```

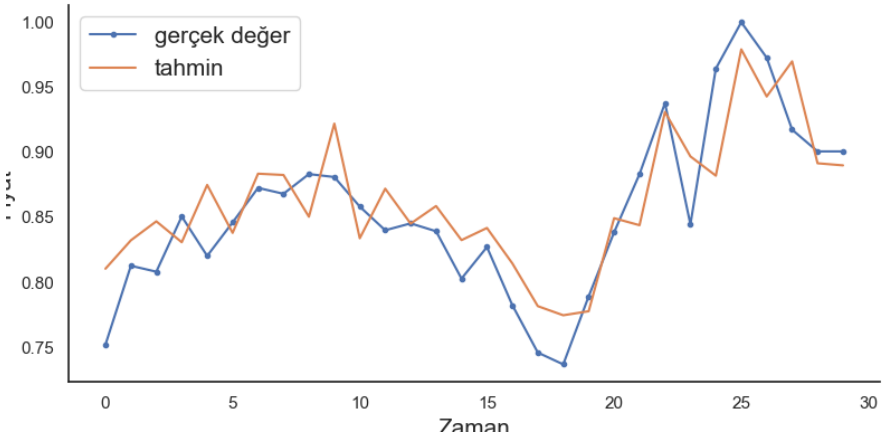
Şekil 6.35. LSTM Modeli Özeti

Modellemeye ilişkin özet şekildeki gibidir.



Şekil 6.36. LSTM Modeli Eğitim ve Test Hatası

Yaklaşık 10. döngüye kadar eğitim ve test hatası dalgalı seyretmektedir.



Şekil 6.37. LSTM Modeli Gerçek ve Tahmin Değerleri

LSTM Modeli kaybolan gradyan problemini çözebilmiş, geçmiş bağımlılıkları yakalamakta iyi bir performans göstermiştir. Bu nedenle tahmin değerleri, gerçek değerlere çok yakın bir desen çizmiştir.

Çizelge 6.2. LSTM modeline ait hata metrikleri

MSE	RMSE	MAE
0.0011	0.0338	0.0283

Hata metrikleri RNN'e kıyasla çok iyi performans göstermiştir.

6.3.3. Geçitli Tekrarlayan Birim (GRU)

GRU(Gated Recurrent Unit) mimarisi RNN (Recurrent Neural Network) gibi bir tür yineleyici sinir ağıdır ancak güncellenen ve sınırlanan hatları olan bir görünüme sahiptir ve girdilerin geçmiş çıktılar ile etkileşimine olanak tanır.

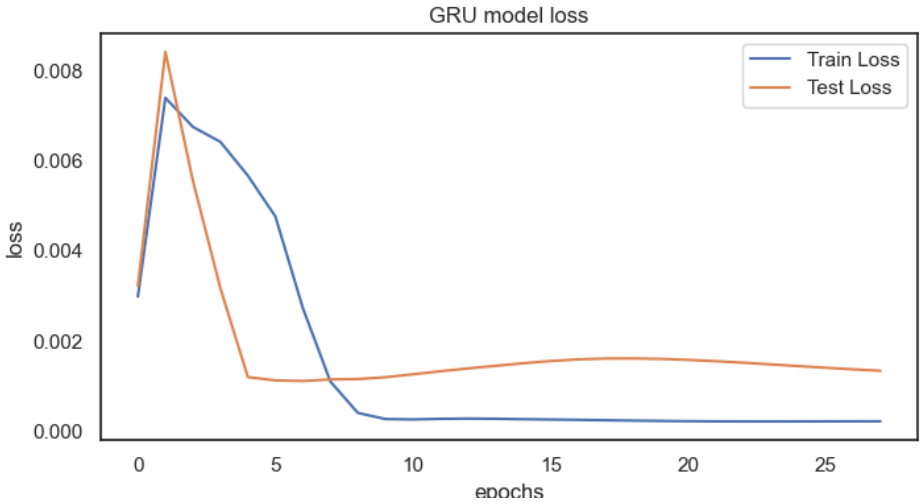
```
>>> model.summary()
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru (GRU)                   (1, 100)                  31800

 dense (Dense)               (1, 1)                    101

=================================================================
Total params: 31,901
Trainable params: 31,901
Non-trainable params: 0
_________________________________________________________________
```

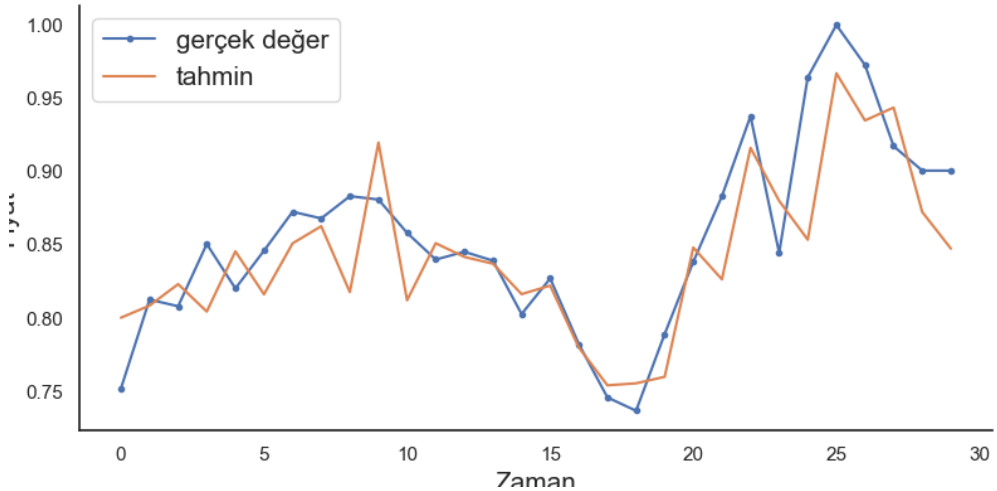
Şekil 6.38. GRU Modeli Özeti

GRU'ya ait model özeti şekilde yer almaktadır.



Şekil 6.39. GRU Modeli Eğitim ve Test Hatası

Test hatasının en düşük olduğu nokta optimal döngü sayısını göstermektedir. Eğitim ve test hatasının hızlıca birbirinden ayrışması durumu aşırı öğrenmeye işarettir. Böyle bir durumun varlığı tezimizde kurulan modellerde görülmemiştir. Bu durumun gözlenmesi halinde daha önce de ifade edildiği gibi erken durdurma ile eğitim durdurulur. Model performansını değerlendirmek ve iyileştirmek için eğitim ve test hatası grafiği iyi bir yöntemdir.



Şekil 6.40. GRU Modeli Gerçek ve Tahmin Değerleri

GRU mimarisinin de veriyle uyumunun çok iyi olduğu görülmektedir.

Çizelge 6.3. GRU modeline ait hata metrikleri

MSE	RMSE	MAE
0.0013	0.0366	0.0284

Hata metriklerinin de sinir ağları dışındaki diğer modellerden çok daha iyi performans gösterdiği görülmektedir.

7. SONUÇLAR VE ÖNERİLER

Bu çalışmayla zaman serisi analizinde klasik yöntemlerin yanısıra sinir ağı mimarilerinin kullanılabilirliği açısından önemli öngörüler elde edilmiştir. Özellikle ardışık verileri işlemede etkin bir performans sergileyen tekrarlayan sinir ağları ve türevleri kullanılarak hisse senedi verisinde iyi bir tahmin başarısı elde edilmiştir.

Yapılan analizler sonucu Anadolu Anonim Türk Sigorta Şirketi (ANSGR) hisse senedine ait en iyi tahmin sonuçlarının tekrarlayan sinir ağlarıyla olduğu tespit edilmiştir. Performans ölçütü olarak hata metriklerinden MSE, RMSE ve MAE kullanılmıştır.

Çizelge 7.1. RMSE hata metriğine göre modeller

Model	Hata Metriği: RMSE
AR	4.45
MA	7.99
ARMA	2.41
ARIMA	2.25
SARIMA	2.54
RNN	0.0620
LSTM	0.0338
GRU	0.0366

RMSE metrikleri kıyaslandığında sinir ağlarının geleneksel modellerden oldukça ayrıştığı görülmektedir. Bunun en önemli nedenlerinden biri sinir ağlarının karmaşık eğilimler ve doğrusal olmama gibi durumları işleyebilme kabiliyetidir. Bölüm 6'da yapılan analizlerde kullanılan hisse senedi veri setinin doğrusal olmadığı, artıkların da doğrusallıktan uzak olduğu, otokorelasyon ve varyansın sabit olmaması gibi problemler göze çarpmaktadır. Yine geleneksel yöntemlerde kullanılan çeşitli veri önişleme adımları ile bu problemlerin üstesinden gelinerek ilgili sürece uygun bir veri elde edilebilir. Klasik zaman serisi tahmin yöntemlerine uygun hale getirilen veriyle yapılan modelleme işlemi hem vakit alacaktır hem de performansı sinir ağlarına göre yine de kısıtlı kalacaktır. Tekrarlayan sinir ağları mimarileri daha az özellik mühendisliği ve veri ön işleme adımlarına ihtiyaç duyar. Değişkenlerin barındırdığı bilginin öğrenilme süreci otomatize bir şekilde gerçekleşir. Ham veri girdi olarak alınır, çeşitli katmanlardan geçerek öğrenme gerçekleşir. Böylece insan müdahalesine çok fazla gerek duyulmadan modellemeye ilişkin veri hazırlığı gerçekleştirilmiş olur.

AR, MA, ARMA, ARIMA, SARIMA gibi klasik metodları kullanırken bir çok farklı parametrenin ayarlanması zorunluluğu ve veri setine uygun yöntemin manuel olarak tespit edilmeye çalışılması zorunluluğu bu yöntemlerin dezavantajları niteliğindedir.

Sinir ağlarıyla geliştirilen modellerde ise özellike gizli katmanlar sayesinde bu işlemler otomatize bir şekilde yapılmaktadır. Derin sinir ağlarının barındırdığı bir takım dezavantajlar da aktivasyon fonksiyonları, optimizasyon ve düzenlileştirme teknikleriyle aşılabilmektedir.

Tekrarlayan sinir ağlarıyla zaman serisi tahmini yapmanın bir diğer avantajı ise algoritmanın hatırlama yeteneğidir. Önceki zaman adımlarındaki bilgiyi saklayarak işleyebilmesi modelleme sürecini daha verimli kılar. Özellikle hisse senedi fiyat tahmini yapılırken hisse senedinin bir önceki günkü fiyatı mevcut fiyatını doğrudan etkiler. RNN'ler bu bağımlılıkları bellekte tutarak mevcut zaman adımındaki bilgiyle birleştirir ve bu işlem tekrarlayarak devam eder. Bu nedenle zaman serisinin yapısına uygun RNN mimarisi seçilerek modelleme işlemi gerçekleştirilebilir.

Tekrarlayan sinir ağları bir çok farklı veri ve probleme hizmet etmek üzere tasarlanmış yapılardır. Hisse senedi verisinin yapısına uygun olan mimari seçilerek minimum hataya sahip olan model kullanılır. Yine 6. Bölüm'den elde edilen analiz çıktılarına göre, gerçekleşen ve tahmin edilen değer grafikleri incelendiğinde sinir ağı mimarilerinin mevcut verideki eğilimleri yakalama konusunda üstün bir başarı gösterdiği görülmektedir.

Çizelge 7.2. RNN tabanlı model hataları

Model	MSE	RMSE	MAE
RNN	0.0038	0.0620	0.0474
LSTM	0.0011	0.0338	0.0283
GRU	0.0013	0.0366	0.0284

Tabloda bulunan üç mimarinin de birbirine yakın sonuçlar ürettiği görülmektedir. LSTM mimarisi literatürde zaman serisi problemlerinde en sık kullanılan yöntemdir. LSTM için burada diğer iki modele göre 3 hata metriğinin de nispeten daha düşük seyrettiği yorumu yapılabilir.

RNN ile yapılan modellemede veri setindeki dalgalanmalar ve gürültünün de üstesinden gelinmiştir. Gereksiz yanlılıklarla başetme konusunda etkin bir rol üstlenilmiştir.

RNN mimarisinin sahip olduğu uzun kısa süreli bellek sayesinde ardışık tüm verilerde iyi bir performans göstereceği çıkarımında bulunulabilir. Enerji tüketimi tahmini, makine çevirisi, fiyat tahmini gibi ardışık sayılabilecek verilerde RNN mimarisi ve türevlerinin kullanılması faydalı olacaktır.

Sonuç olarak yatırımcılar, şirketler ve ekonomik anlamda bir çok önem barındıran hisse senedi fiyat tahmini konusu, ileri teknik ve metodolojilerle kurulan modeller sayesinde performans ve maliyet açısından yüksek oranda fayda sağlayacaktır.

KAYNAKLAR

Anonim 1: https://www.foreks.com/sembol-detay/H2010/ansgr/anadolu-sigorta [Son erişim tarihi: 13.01.2023].

Amidi, S., CS230 Deep Learning Class. Recurrent Neural Networks. https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks [Son erişim tarihi: 24.01.2023]

Box, G.E.P., ve Jenkins, G.M., 1970. Time Series Analysis: Forecasting and Control. San Francisco: Holden-Day, California, 553s.

Bouktif, S.; Fiaz, A.; Ouni, A.; Serhani, M.A., 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.

Brownlee, J., 2020. How to Decompose Time Series Data into Trend and Seasonality. https://machinelearningmastery.com/decompose-time-series-data-trend-seasonality/ [Son erişim tarihi: 11.09.2023].

Chollet, F., ve Diğerleri, 2015. Keras: https://github.com/fchollet/keras. [Son erişim tarihi: 11.09.2023]

Chung, J., Gulcehre, C., Cho, K., ve Bengio, Y., 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling, in NIPS 2014 Workshop on Deep Learning, Montreal, Canada.

Farley, B., Clark, W.A., 1954. Simulation of Self-Organizing Systems by Digital Computer. IRE Transactions on Information Theory, 4 (4): 76–84.

Fernandez, C., Soria, E., Martin, J.D., Serrano, A.J., 2006. Neural networks for animal science applications: Two case studies. Expert Systems With Applications, 31(2006):444-450.

Frisch, R., 1933. Pitfalls in the Statistical Construction of Demand and Supply Curves.

Gallant, S.I., 1993. Neural Network Learning and Expert Systems. The MIT Press, Cambridge, 127s.

Gers, F., Schmidhuber, J., ve Cummins, F., 2000. Learning to Forget: Continual Prediction with LSTM. Neural computation, 12(10), 2451-71.

Glorot, X. ve Bengio, Y., 2010. Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Sardinia, Italy.

Goodfellow, I., Bengio, Y., ve Courville, A., 2016. Deep Learning. MIT Press, 800s.

Hebb, D.O., 1949. The Organization of Behavior. Wiley & Sons, New York, 335s.

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

Huber, P., J., 1964. Robust Estimation of a Location Parameter. The Annals of Mathematical Statistics, 35(1):73-101.

Kolmogorov, A.N., 1940. The Wiener Helix and Other Interesting Curves in the

Hilbert Space, Dokl. Acad. Sci., USSR 26: 115–118.

Kullback, S., Leibler, R.A., 1951. On information and sufficiency. Annals of Mathematical Statistics. 22 (1): 79–86.

Lapedes, A., ve Farber, R., 1987. How Neural Nets Work. International Conference on Neural Information Processing Systems (NIPS'87), Massachusetts, ABD.

LeCun, Y., Boser, B.E., Denker, J.S., Henderson D., 1989. Handwritten Digit Recognition with a Back-Propagation Network. Neural Computation, 1(1989): 541-551.

MacKay, D.J.C., 2003. Information Theory, Inference and Learning Algorithms. Cambridge University Press, Cambridge, 640s.

Marquez, L., Hill, T., O'Connor, M., Remus, W., 1992. Neural network models for forecast a review. In: IEEE proceedings of the 25th Hawaii International Conference on System Sciences, Hawaii, ABD.

McCulloch, W.S., Pitts, W. A Logical Calculus of the Ideas Immanent in Nervous Activity. Bulletin of Mathematical Biophysics 5(1943):115–133.

Minsky, M. ve Papert, S., 1969. Perceptrons. M.I.T. Press., Cambridge, 258s.

Nelson, M., Hill, T., Remus, B., O'Connor, M., 1994. Can Neural Networks Be Applied to Time Series Forecasting and Learn Seasonal Patterns: An Empirical Investigation. Twenty Seventh Annual Hawaii International Conference on System Sciences, Hawaii, ABD.

Ng, A., 2023. Machine Learning Specialization. https://www.deeplearning.ai/courses/machine-learning-specialization/ [Son erişim tarihi: 01.06.2017].

Nielsen, M., 2015. Neural Networks and Deep Learning. http://neuralnetworksanddeeplearning.com/ [Son erişim tarihi: 24.01.2023].

Olah, C., 2015. Understanding LSTM Networks. https://colah.github.io/posts/2015-08-Understanding-LSTMs/ [Son erişim tarihi:11.09.2023].

Pal, A., ve Prakash, PKS., 2017. Practical Time Series Analysis : Step by Step Guide Filled with Real World Practical Examples. Packt Publishing, 244s.

Rehal, V., 2022. Interpreting ACF and PACF Plots. https://spureconomics.com/interpreting-acf-and-pacf-plots/ [Son erişim tarihi: 11.09.2023].

Rosenblatt, F., 1958. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. Psychological Review, 65(6):386-408.

Rosenblatt, F., 1960. Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Cornell Aeronautical Lab Inc, New York, 626s.

Rumelhart, D.E., 1987. Parallel Distributed Processing. The MIT Press, Cambridge, 632s.

Rumelhart, D.E., Hinton, G.E., ve Williams, R.J., 1986. Learning Representations by Back-Propagating Errors. Nature 323 (1986): 533-536.

Sharda, R., Patil, R.B., 1992. Connectionist approach to time series prediction: An empirical test. Journal of Intelligent and Manufacturing 3(5), 317–323.

Seabold, S., ve Perktold, J. 2010. statsmodels: Econometric and Statistical Modeling with Python. In 9th Python in Science Conference, Teksas, ABD.

Siami-Namini, S., Tavakoli, N., ve Siami Namin, A., 2019. A Comparison of ARIMA and LSTM in Forecasting Time Series. 17th IEEE International Conference on Machine Learning and Applications, Orlando.

Slutsky, E., 1927. The Summation of Random Causes as the Source of Cyclic Processes. In Problems of Economic Conditions, 3(1).

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., ve Salakhutdinov, R., 2014. Journal of Machine Learning Research 15(2014):1929-1958.

Tang, Z., Almeida, C., Fishwick, P.A., 1991. Time series forecasting using neural networks vs Box-Jenkins methodology simulation 57 (5), 303–310.

Tsantekidis, A., Passalis, N., Tefas, A., Kanniainen, J., Gabbouj, M., ve Iosifidis, A., 2017. Forecasting Stock Prices From Limit Order Book Using Convolutional Neural Networks. 19th IEEE International Conference on Business Informatics, Selanik, Yunanistan.

Werbos, P., 1974. Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. Ph.D. Thesis, Harvard University, Cambridge, 454s.

Widrow, B., ve Hoff, M. E., 1960. Adaptive Switching Circuits. Stanford Univ Ca Stanford Electronics Labs, California, United States.

Yaffee, R.A., ve McGee M., 2000. Introduction to Time Series Analysis and Forecasting: with Application of SAS and SPSS. Academic Press, San Diego, 555s.

Yule, G.U., 1927. On a Method of Investigating Periodicities in Disturbed Series, with Special Reference to Wolfer's Sunspot Numbers. Philosophical Trans. Royal Soc. London. Ser. A, Containing Papers of a Mathematical or Physical Character, 226(1927): 267-298.

Zhang, A., Lipton, Z., C., Li, M., ve Smola, A., J., 2022. Derin Öğrenmeye Dalış.
https://tr.d2l.ai/d2l-tr-pytorch.pdf [Son erişim tarihi: 24.01.2023]

Zhang, Y., ve Liu, W., 2017. Stock Market Prediction Using Multi-Layer Perceptrons with Financial News. IEEE Transactions on Neural Networks and Learning Systems, 28(7), 1786-1798.

Zhao, Z., Chen, W., Wu, X., Chen, P. C., ve Liu, J., 2017. LSTM Network: a Deep Learning Approach for Short-Term Traffic Forecast. IET Intelligent Transport Systems, 11(2), 68-75.

ÖZGEÇMİŞ

Güldeniz CANATAN. Ortaöğretimi Cibali Lisesinde 2011 yılında bitirdi. 2019 yılında Yıldız Teknik Üniversitesi İstatistik Bölümünü Bitirdi. Halen Anadolu Sigortatada Veri Bilimi Uzmanı olarak çalışmaktadır.