İstatistikçiler Dergisi: İstatistik & Aktüerya

Journal of Statisticians: Statistics and Actuarial Sciences

IDIA 18, 2025, 1, 16-49

Geliş/Received:28.05.2025, **Kabul**/Accepted: 14.06.2025

Araştırma Makalesi / Research Article

Geleneksel yaklaşımlardan derin öğrenmeye zaman serisi öngörü yöntemleri: Pyhton ve MATLAB uygulamaları

Çağatay Bal

Muğla Sıtkı Koçman Üniversitesi, Fen Fakültesi, İstatistik Bölümü, Muğla, Türkiye
cagataybal@mu.edu.tr
ORCID: 0000-0002-7823-2712

Çağdaş Hakan Aladağ

Hacettepe Üniversitesi, Fen Fakültesi, İstatistik Bölümü, Ankara, Türkiye
aladag@hacettepe.edu.tr
ORCID: 0000-0002-3953-7601

Öz

Zaman serisi analizi birçok bilimsel alanda ihtiyaç duyulan bir araştırma alanı olarak uzun yıllardır önemini korumaktadır. Belirli bir zaman noktasındaki değişkenlerin davranışlarını anlama, modelleme ve geleceğe ilişkin öngörülerde bulunma taşıyan zaman serisi analizi geleneksel ve modern yöntemlerle gerçekleştirilmektedir. Son yıllarda büyük bir atılım ile gelişen yapay zekâ yöntemlerinin en güçlülerinden olan derin öğrenme, zaman serisi analizi için oldukça güçlü bir yaklaşım olarak kabul edilmektedir. Bu amaç doğrultusunda bu çalışmada zaman serisi analizi için kullanılan yöntemlerin geçmişten günümüze gelişimini ve birbirine olan avantaj ve dezavantajlarıyla detaylı bir şekilde incelenerek MATLAB ve Python kodlarıyla gerçek zamanlı kullanımı sunulmaktadır.

Anahtar sözcükler: Zaman serisi analizi, Box-Jenkins, Yapay sinir ağları, Derin ağlar, MATLAB, Python

Abstract

Time Series Forecasting Methods from Traditional Approaches to Deep Learning: Python and MATLAB Applications

Time series analysis has been a crucial topic in various scientific fields for many years due to its importance in predicting future observations. The forecasting of time-dependent data has been approached using both traditional and modern methods. In recent years, deep learning, one of the most advanced artificial intelligence techniques, has emerged as a powerful tool for time series analysis. In this context, this study examines the evolution of time series forecasting methods from past to present, providing a comprehensive comparison of their advantages and disadvantages. Furthermore, real-time applications are demonstrated using MATLAB and Python, offering a practical implementation of these methodologies.

Keywords: *Time series analysis, Box-Jenkins, Artificial neural networks, Deep learning, MATLAB, Python.*

1. Giriş

Zaman serileri, eşit aralıklı zaman noktalarında gözlemlenen rastgele değişkenlerin gerçekleşmiş değerlerinden oluşan dizilerdir. Zaman serisi gecikmeleri ise, bu dizinin geçmiş zaman noktalarına ait gözlemleri olarak tanımlanabilir. Zaman serisi analizlerindeki temel varsayım, geçmişte meydana gelen değişimlerin geleceği etkileyeceği ve bu değişimlerin zaman serisi gecikmeleri arasındaki sayısal ilişkide saklı olduğu varsayımıdır. Dolayısıyla zaman serisi analiz yöntemleri gecikmeler arasındaki bu ilişkileri açığa çıkarıp modellemeye odaklanmaktadır. Ayrıca zaman serisi analiz yöntemleri, verideki zamana bağlı desenleri ortaya çıkararak geleceğe dair daha isabetli çıkarımlar yapmayı da amaçlar. Bu yönüyle, öngörüdeki belirsizliği azaltmak için kullanılan birçok istatistiksel modelleme yaklaşımıyla benzer amaçlar taşımaktadır.

Modelleme aşaması, zaman serisi verisinin her bir zaman noktasında tutarlı tahminler vermesi ve bir çok açıdan gerçek verinin karakteristiğine uygun tahminler yapmasını amaçlar. Bu doğrultuda en temel ölçüt, gerçek gözlemler ile tahmin edilen gözlemler arasındaki hatanın düşük olmasıdır. Ancak bu ölçüt tek başına yeterli değildir; çünkü düşük hata değeri her zaman modelin zaman serisinin yapısal özelliklerine uygun tahminler ürettiği anlamına gelmeyebilir. Elde edilen tahminlerin "uygun" tahminler olması; hatanın minimum olmasına, geçmiş ve gelecek gözlemler arasındaki ilişkinin model tarafından doğru anlaşılmış olmasına, modelleme aşamasından sonra yapılacak "***öngörü***" 'lerin gerçek değerleri yansıtma konusunda güçlü olması gibi etkenlere bağlıdır. Dolayısıyla, başarılı bir model yalnızca hata minimizasyonu sağlayan değil, aynı zamanda zaman serisinin yapısal dinamiklerini anlamlandırabilecek yetkinlikte olmalıdır.

Geleneksel yaklaşımlarda zaman serileri ile ilgili elde edilen modelin minimum hatayı vermesi gelecek gözlemleri "***öngörmesi***" açısından yeterli olduğu kabulüne dayanmaktadır. Dolayısıyla modelin olası öngörü performansından ziyade elindeki bilgiyi ne kadar iyi öğrendiği önemlidir. Bunun için geliştirilen ilk kriter de bir "bilgi" kriteridir. Akaike, geliştirdiği bilgi kriteri ile zaman serisi modellerinin karşılaştırılmasında ya da model seçiminde kullanılan geleneksel yaklaşımı literatüre kazandırmıştır [1]. Hataya dayalı olan *Akaike Bilgi Kriterinde (**ABK**)* modeller arasında minimum ABK değerine sahip olan model daha iyi model olarak seçilmektedir.

Zaman serisi analizlerinde kullanılan ve bilinmesi gereken en önemli kavram "***otokorelasyon***" olarak adlandırılır. Otokorelasyon, bir zaman serisinin belirli bir zaman gecikmesiyle olan içsel ilişkisini ifade eder. Geleneksel modeller arasında en yaygın olarak kullanılan ve bilinen ARIMA modeli, otokorelasyon kavramına dayanmaktadır. Bir sonraki bölümde geleneksel teknikler daha detaylı incelenecektir.

Geleneksel zaman serisi analiz yöntemleri parametrik yöntemlerdir. Bu modeller, belirli varsayım ve koşullara bağlı olarak teorik bir çerçevede çalışırlar. Esnekliğin görece düşük olmasına rağmen, varsayımlar sağlandığında parametrik olmayan yöntemler kadar güçlü sonuçlar elde edebilirler. Bu bölümde detaylıca inceleyeceğimiz ***Yapay Sinir Ağları*** parametrik olmayan yöntemlere örnek olarak gösterilebilir. Herhangi bir varsayımı olmayan ve neredeyse her koşulda çalışıp sonuç üretebilen bu yöntem çoğu zaman ***ARIMA*** gibi geleneksel modellere nazaran daha iyi sonuçlar üretebilmektedir. Ancak unutulmamalıdır ki veriye dayalı olan bu yöntemler için genel-geçer bir genelleme yapmak dopru olmamaktadır. Karşılaştırmalar ve sonuçlar sadece ilgili zaman serisi verisi için geçerli olacaktır.

Yapay sinir ağları insan beyninin çalışmasını taklit etmeye yönelik geliştirilen bir makine öğrenme yöntemidir. Bahsedildiği üzere yapay sinir ağları öğrenme kapasitesine sahip olan yaklaşımlardır. Bir çok tabaka ve bu tabakalarda bulunan "***nöron***" adı verilen işlem birimlerinin birbirlerine bağlı olarak bir "***ağ***" yapısı oluşturduğu bu sistemde, girdi olarak verilen zaman serisi gecikmeleri sonraki gözlemleri hedef alarak onları tahmin etmeye çalışır. Gerçek değerlerle elde edilen tahminlerin arasındaki hatanın minimize edilmesine dayalı olan algoritmada amaç nöronlar arasındaki bağlantıların bir bütün şeklinde en iyi tahminleri veren ağ mimarisine yani modele dönüşmesini sağlamaktır. En iyi ağırlıkların elde edilmesiyle "***öğrenme***" süreci tamamlanır ve eğitilen sinir ağ artık tahmin ve öngörü yapmaya hazır bulunmaktadır.

İlk olarak McCulloch ve Pitts [2] tarafından geliştirilen ve "***Perceptron***", aynı zamanda "tekli doğrusal algılayıcı" olarak da adlandırılan sinir modeli tek bir nörondan oluşmaktadır ve doğrusal bir ayırıcı olarak

görev yapmaktadır. Werbos [3] tarafından geliştirilen "***Geri Yayılım Algoritması***" olarak bilinen yöntem ise sinir ağına öğrenme yetisini katmış ve Rumelhart [4] tarafından uygulanabilir hale getirildikten sonra çok tabakalı ve birçok nörona sahip ağ yapısı ile oldukça güçlü bir esnek hesaplama yöntemine dönüşmüştür. Yapay sinir ağları sınıflandırma, kümeleme ve modelleme gibi bir çok alanda kullanılabilen oldukça modüler bir nümerik hesaplama aracıdır. Farklı amaçlara yönelik tasarlanmış ve farklı çalışma prensiplerine sahip bir çok yapay sinir ağı literatürde mevcuttur.

Nümerik bir analiz yöntemi olduğundan yapay sinir ağları ile analiz tamamen işlem yüküne dayalı bir yapıya sahiptir ve bu işlem yükü sinir ağını oluşturan tabaka ve nöronlar arttıkça artmaktadır. Yıllar içerisindeki gelişiminde yapay sinir ağlarının önündeki en büyük engel bu işlem yükü olmuştur. Problemlerin artan boyutlarıyla başa çıkabilmesi için geliştirilmesi gereken daha büyük ve karmaşık yapıdaki sinir ağları iki büyük engel ile karşılaşmıştır. Birincisi yapay sinir ağlarının öğrenme yetisinin negatif bir sonucu olabilen "***ezberleme***" durumudur. Ezberleme kısaca sinir ağının veri yapısını öğrenmek yerine birebir taklit etmesi anlamına gelir ve ağın oldukça kötü öngörüler üretmesine neden olur. İkincisi ise karmaşıklaşan ağ yapısının eğitim sürecinin işlem yükünden dolayı çok uzun sürmesi.

Ezberleme sorunu için geliştirilen birçok yöntem olmakla birlikte karmaşık ağların eğitilmesi ve kullanışlı hale getirilmesi uzun yıllar almıştır. Gelişen teknolojiyle beraber bilgisayarların sahip olduğu ve GPU adı verilen işlem ünitelerinin yapay sinir ağları gibi karmaşık yapıların oluşturulmasında çok daha işlevsel olabileceğinin fark edilmesiyle "***Derin Ağlar***" olarak adlandırılan çok fazla tabaka ve nörona sahip karmaşık yapay sinir ağlarının eğitilmesindeki engeller ortadan kalkmıştır. Ancak işlem hızını pozitif yönde etkileyen en önemli gelişme yapay sinir ağlarının bir bileşeni olan "***aktivasyon fonksiyonu***" olarak adlandırılan ve genelde uygulanan "***Sigmoid***" fonksiyonunun yerine "***ReLU***" adında bir fonksiyonun geliştirilmesi olmuştur. "*Doğrultulmuş Doğrusal Birim (**ReLU**)*" olarak adlandırılan bu fonksiyon doğrusal olmayan bir fonksiyon karakterinde çalıştığından dolayı sigmoid fonksiyonlar kadar efektif olup, aynı zamanda basit yapısıyla işlem hızını oldukça düşürmüştür. Bu iki gelişmenin ışığında derin ağları artık uygulanabilir hale gelmiş ve sahip olduğu güçlü yapısından dolayı bir çok probleme uygulanabilir hale gelmiştir.

Otokorelasyon kavramını biraz daha açacak olursak eğer, zaman serisinin gecikmeleri arasındaki korelasyon yani ilişki olarak açıklayabiliriz yani r_1, z_t ile z_{t-1} arasındaki korelasyon iken r_2, z_t ile z_{t-2} arasındaki korelasyon değerini verir. Aşağıdaki formülde görülmek üzere;

$$r_k = \frac{\sum_{t=k+1}^{T}(y_t-\bar{y})(y_{t-k}-\bar{y})}{\sum_{t=1}^{T}(y_t-\bar{y})^2} \tag{1}$$

elde edilir. *T* zaman serisinin uzunluğudur.

Görüldüğü üzere otokorelasyon temelde zaman serisinin kendisi ile gecikmeleri arasındaki korelasyon değerlerinden oluşmaktadır. Burada sorulması gereken bir soru açığa çıkmaktadır. Peki ya gecikmeler arasındaki korelasyon değeri de önemli midir ? Konunun başında söz edilen gözlemler arasındaki ilişki herhangi iki ya da daha fazla gecikme arasında da olabilir. Bu belirsizliği çözümlemede elde edilebilecek her bilginin kullanılması modelleme açısından en doğru yol olduğu ortadadır. Yapay sinir ağları ile zaman serisi analizi yaparken sisteme girdi olarak zaman serisinin gecikmeleri verilir. Bu gecikmeler çeşitli katmanlar ve nöronlar aracılığıyla içiçe bir anlam kazanarak sonuca ilerler. Dolayısıyla sadece zaman serisi ile gecikmeler arasındaki ilişki değil bir bütün olarak gecikmeler arasındaki ilişkiler de ele alınır. Bu şekilde belirsizliği çözmek için daha fazla bilgi edinmiş oluruz ve eğittiğimiz ağ bizi daha doğru sonuçlar veren daha güçlü modellere götürür.

Peki ya aynı işlem için derin ağlar kullanılırsa ne olur ? Daha fazla tabaka ve daha fazla nöron daha fazla gecikmeyi sisteme girdi olarak alabilmek ve daha fazla ilişkinin incelenmesi anlamına gelir. Dolayısıyla belirsizliğin çözümünde elimizde daha fazla bilgi olur ve bu da bize çok daha iyi modeller elde etmemiz konusunda yardımcı olur.

Özetleyecek olursak eğer, zaman serisinin en iyi tahmini verebilen yöntem zaman serisini oluşturan gecikmeler arasındaki otokorelasyonu en iyi öğrenen modeldir. Bunun için de derin ağlar biçilmiş kaftan olarak nitelendirilebilir.

İlerleyen bölümlerde geleneksel yöntemlerden ve sinir ağlarından ***Python*** ve ***MATLAB*** uygulamalarıyla birlikte detaylı bilgiler verilecektir.

2. Python ve MATLAB kodları ile Box-Jenkins modelleri ve uygulamaları

Zaman serisi analizlerine geçmeden önce zaman serileri analiziyle ilgili temel kavram ve bileşenler bahsetmek gerekmektedir [5,6].

Trend; Zaman serisinin tanımlı olduğu dönemde süregelen artış ya da azalış hareketini yani genel eğilimini ifade etmektedir. Bu eğilim doğrusal ya da eğrisel olarak gözlemlenebilir.

Mevsimsellik; Zaman serisi belirli bir döngüde artış ya da azalışa sahip bir dalgalanma içeriyorsa eğer mevsimselliğe sahip olduğu ve bu dalgalanmanın uzunluğu da serinin periyodu olarak adlandırılır.

Rastgelelik; Zaman serisinin trend ve mevsimsellik bileşenleri dışında artan kalan kısmı ise rastgelelik olarak adlandırılmaktadır. Rastgelelikten meydana gelen kısım modellenemeyen ve belirli bir sistematiğe sahip olmayan kısımdır.

Bir diğer önemli kavram ise zaman serilerinde durağanlıktır. Durağanlık, serinin istatistiksel özelliklerinin; özellikle ortalama, varyans ve otokorelasyon yapısının zamanla değişmemesi durumudur. Yani durağan bir zaman serisinde her zaman noktasındaki gözlem, aynı dağılıma sahip rastgele değişkenlerden oluşur. Bu bağlamda, bir zaman serisinin trend içermesi, ortalamasının zamanla değiştiğini gösterir ve bu durum serinin durağan olmadığını ifade eder. Durağan olmayan seriler, analiz ve modelleme aşamasında genellikle dönüşüm işlemlerine tabi tutulur.

Geleneksel zaman serisi analizlerinin uygulanması serinin durağan olması kavramına dayanmaktadır. Durağanlıştarma işlemi ise fark alma denilen bir işlem sayesinde gerçekleştirilir. Fark alma işlemi serinin son gözlemlerinden belirli bir dönem gecikmelerinin çıkarılmasıyla yapılır. Fark alma sayesinde, stokastik (deterministik olmayan) trend ve varsa mevsimsellik etkileri ortadan kaldırılarak zaman serisi analiz için uygun hale getirilir. Ancak deterministik trend içeren serilerde bu işlem tek başına yeterli olmayabilir; bu durumda trend bileşeninin ayrıca modellenmesi veya çıkarılması gerekebilir.

Zaman serileri analizinde önceki bölümde de bahsedildiği üzere otokorelasyon kavramı çok büyük önem arz etmektedir. Bir başka önemli kavram ise kısmi otokorelasyondur. Kısmi otokorelasyon katsayısı, diğer gecikmelerin etkileri ihmal edildip , z_t ile z_{t+m} zaman serileri arasındaki ilişki miktarını verir. Aşağıdaki formülde görülmek üzere;

$$r_{kk} = \frac{r_k - \sum_{j=1}^{k-1}(r_{k-1,j})(r_{k-j})}{1 - \sum_{j=1}^{k-1}(r_{k-1,j})(r_j)} \qquad (2)$$

elde edilir. Burada r_{kk} k. gecikmeye ait otokorelasyon katsayısıdır.

Formül 1 ve 2 ile elde edilen gecikmelere ait bütün otokorelasyon ve kısmi otokorelasyon değerlerinin oluşturduğu *otokorelasyon fonksiyonu (**ACF**)* ve *kısmi otokorelasyon fonksiyonu (**PACF**)* grafikleri, serinin durağan olup olmadığının anlaşılmasında ve sonrasında doğru modelin seçilmesinde anahtar rol oynamaktadır.

2.1. Box-Jenkins Modelleri

Box-Jenkins yöntemi [5] 1960`lı yıllardan günümüze kadar ve günümüzde de oldukça çok kullanılan ve en çok bilinen zaman serisi analiz yöntemlerinden biridir. Bu modeller genel olarak *ARIMA* olarak adlandırılmaktadır. *ARIMA* modelleri mevsimsel olmayan Box-Jenkins modelleri olarak (p,d,q) parametrelerine sahiptir. Bu parametreler; *p Otoregresyon (**AR**)*, *d* fark alma işlemi sayısı ve *q Hareketli Ortalama (**MA**)* modellerinin dereceleridir. *SARIMA* modelleri ise (P,D,Q) parametreleriyle mevsimsel Box-Jenkins modelleri olarak adlandırılırlar. Bu parametreler; *P Mevsimsel Otoregresyon (**SAR**)*, *D* mevsimsel fark alma işlemi sayısı ve *Q Mevsimsel Hareketli Ortalama (**SMA**)* modellerinin dereceleridir.

Box-Jenkins modelleme yöntemi 4 ana aşamadan oluşmaktadır. Bunlar sırasıyla *model bulma, parametre tahmini, artıkların analizi* ve *öngörü* aşamalarıdır.

Bu yöntemi uygulayabilmek için modellenecek zaman serisi durağanlık koşulları sağlamalıdır. Bu nedenle ilk adım eğer seri durağan değilse fark alma işlemiyle seriyi durağan hale getirmektir. Trend ve mevsimsellikten arındırılan zaman serisi analize uygun hale getirilir.

Model Bulma; durağanlaştırılan serinin *ACF* ve *PACF* grafikleri elde edilerek uygun model belirlenmeye çalışılır. Bu kısım biraz karmaşık olabilmekle birlikte tecrübe gerektiren bir aşama olarak kabul edilebilir. En temel şekilde, eğer *ACF* grafiğindeki ilişkiler zamanla azalıyor ancak *PACF* grafiğinde azalma ani bir şekilde oluyorsa uygun model *p* parametresine sahip *otoregresyon* (*AR*) modelidir. Grafiklerde tam tersi bir hareketlilik mevcutsa ozman uygun model *q* parametresine sahip *hareketli ortalamalar* (*MA*) modelidir. Son olarak her iki grafikteki azalmalar yavaş bir şekilde gerçekleşiyorsa uygun model *p* ve *q* parametlerine sahip *otoregresif hareketli ortalamalar* (***ARMA***) modelidir.

Parametre Tahmini; aday model belirlendikten sonra parametlerin tahmini için literatürde kullanılan yaklaşımlar *olabilirlik fonksiyonunun* maksimize edilmesi veya hata kareler fonksiyonunun *en küçük kareler* yöntemiyle minimize edilmesi ile gerçekleştirilir.

Artıkların Analizi; model bulma ve parametre tahmini aşamasının ardından tahmin edilen modelin artıklarının *ak gürültü süreci* oluşturup oluşturmadığı kontrol edilir. Ak gürültü süreci ise kısaca modelin artıklarının standart normal dağılımlı rastgele değişkenlerden oluşması anlamına gelir. Artıkların ak gürültü serisi olup olmadığı *Box-Ljung* testi ile belirlenir. Artık analizinde elde edilen sonuçlar ak gürültü sürecini veriyorsa eğer *Akaike Bilgi Kriteri (**ABK**)* ve *Bayesci Bilgi Kriteri (**BBK**)* gibi ölçütlerle modelin performansına ilişkin sonuçlar elde edilir. *ABK* ve *BBK* istatistiğinin minimum olan modelin daha iyi model olduğu sonucuna varılabilir.

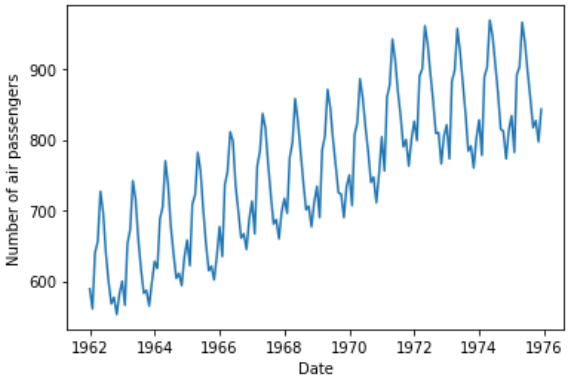
Öngörü; gelecek gözlemlerin tahmin aşaması elde edilen ve istatistiksel açıdan en uygun modelden yararlanılarak yapılmaktadır. En basit şekilde açıklamak gerekirse geçmiş gözlemleri kullanarak bir adım ötesi öngörü değeri ilgili modelin matematiksel formülü yardımıyla elde edilir. Aynı işlem tekrarlanarak istenilen zaman uzunluğunda öngörü değerleri elde edilebilir.

2.1.1. Python kodları ile Box-Jenkins modelleri

Bu bölümde gerçek hayatta da en çok karşılaşılan trend ve mevsimsellik bileşenlerini taşıyan zaman serileri analizi için Python kodları ile birlikte adım adım detaylı analiz yapılacaktır. Kullanılan veri havayolu yolcuları isminde literatürde oldukça bilinen ve yıllar boyunca analiz edilmiş bir zaman serisidir. İlgili data "https://www.kaggle.com/rakannimer/air-passengers" linki üzerinden elde edilebilir.

```
1.  import numpy as np
2.  import pandas as pd
3.  from matplotlib import pyplot as plt
4.  from pandas.plotting import register_matplotlib_converters
5.  register_matplotlib_converters()
6.  df = pd.read_csv('air_passengers.csv', parse_dates=['Month'], index_col = ['Month'])
7.  df.head()
8.  plt.xlabel('Date')
```

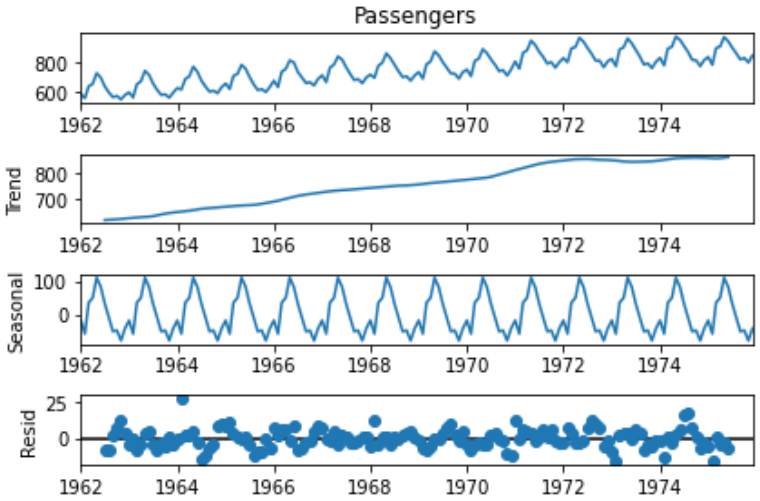
```
plt.ylabel('Number of air passengers')
plt.plot(df)
```



Şekil 1. Havayolu Yolcu Veri Grafiği

Analiz verinin grafiğini çizerek ve bunun için gerekli olan kütüphaneleri aktif ederek başlanabilir. Grafikte de görüleceği üzere zaman serisi trend ve mevsimsellik bileşenlerine sahiptir. Ancak grafik ile emin olunamayacak veriler için ya da daha net bir şekilde trend ve mevsimsellik bileşenlerini görebilmek adına ayrıştırma işlemi yapılmalıdır.

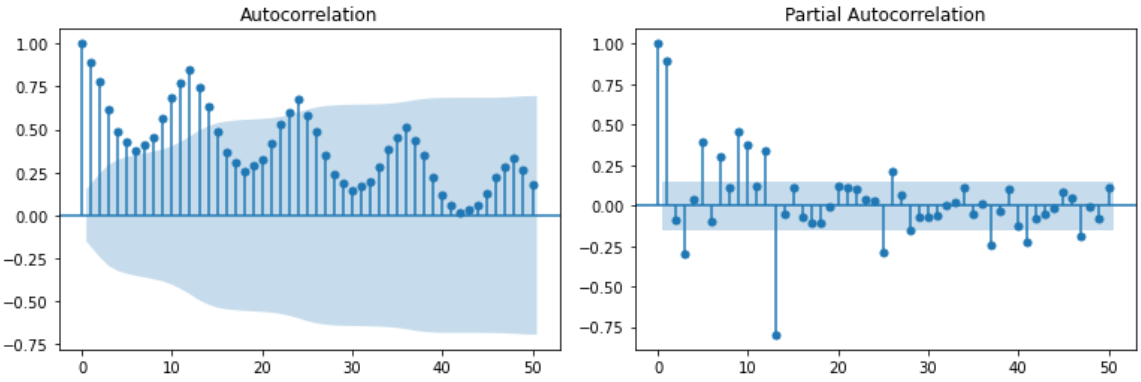
```
import statsmodels.api as sm
from statsmodels.tsa.seasonal import seasonal_decompose
decomposition = seasonal_decompose(df['Passengers'], freq=12)
decomposition.plot()
plt.show()
```



Şekil 2. Havayolu Yolcu Veri Bileşenleri

Ayrıştırma işleminin ardından verinin trend ve mevsimsellik bileşenlerine sahip olduğu çok daha net bir şekilde karşımıza çıkmaktadır. Ayrıca ayrıştırmanın ardından artıklara ilişkin grafik de yukarıda görülebilmektedir.

```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
plot_acf(df, lags=50)
plot_pacf(df, lags=50)
```



Şekil 3. Havayolu Yolcu Verisi ACF ve PACF Grafikleri

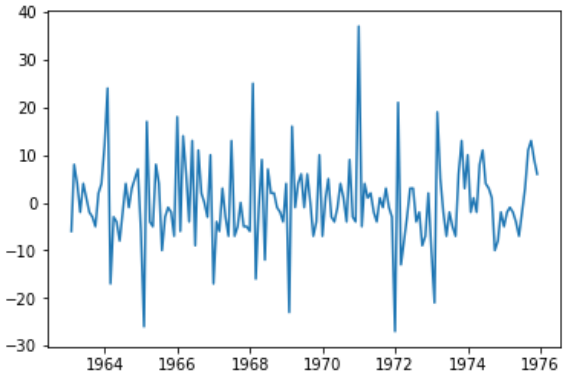
Trend ve mevsimsellik bileşenlerine sahip zaman serilerinin durağan olmadığını ve analize başlamadan önce durağanlaştırılması gerektiği bilinmektedir. Ancak verinin durağan olup olmadığına karar verilemediğinde Otokorelasyon (ACF) ve Parçalı Otokorelasyon (PACF) grafiklerine bakmak durağanlık konusunda fikir verebilmektedir. Yukarıdaki ACF ve PACF grafiklerinde de verinin durağan olmadığı açıkça görülmektedir.

```
20. from statsmodels.tsa.stattools import adfuller
21. dftest = adfuller(df['Passengers'])
22. dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-
    value','#Lags Used','Number of Observations Used'])
23. for key, value in dftest[4].items():
24.     dfoutput['Critical Value (%s)'%key] = value
25. print(dfoutput)
```

```
Test Statistic                  -1.303812
p-value                          0.627427
#Lags Used                      13.000000
Number of Observations Used    154.000000
Critical Value (1%)             -3.473543
Critical Value (5%)             -2.880498
Critical Value (10%)            -2.576878
dtype: float64
```

Elde edilen grafiklerle herhangi bir sonuç alınamadığında ise birim kök testlerine başvurulması gerekir. *Dickey-Fuller birim kök testi* sonucunda görüldüğü gibi "***p-value***" değeri *0,05* değerinin üzerinde çıkmaktadır. Bu test sonucu zaman serisinin durağan olmadığını göstermektedir.

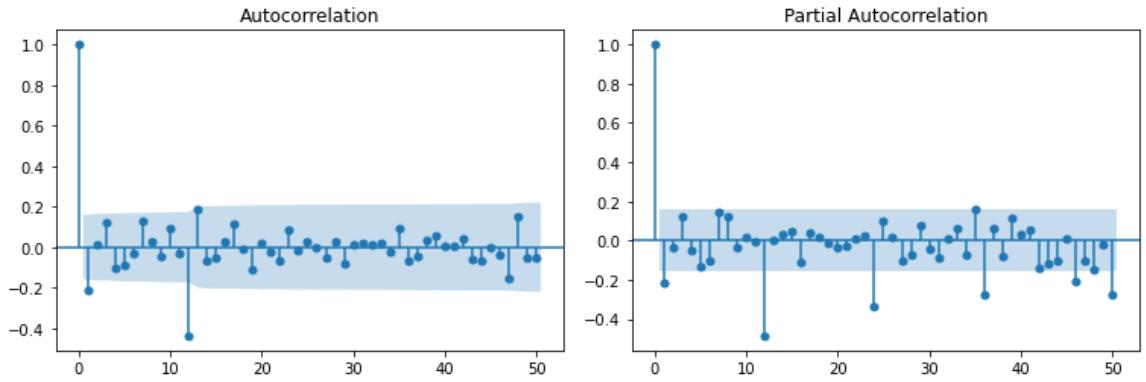
```
26. df_diff = df.diff().diff(12).dropna()
27. plt.plot(df_diff)
```



Şekil 4. Durağanlaştırılmış Havayolu Yolcu Verisi

Trend ve mevsimsellik bileşenlerinden fark alma işlemi ile ayrıştırılan ve durağan hale getirilen serinin grafiği yukarıda görülmektedir. Burada zaman serisi trend ve mevsimsellik bileşenlerine sahip ise hem birinci dereceden farkının hem de mevsimsel farkının alınması gerekmektedir.

```
28. from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
29. plot_acf(df_diff, lags=50)
30. plot_pacf(df_diff, lags=50)
```



Şekil 5. Durağanlaştırılmış Havayolu Yolcu Verisi ACF ve PACF Grafikleri

Durağanlaştırılan serinin ACF ve PACF grafiklerine bakıldığında artık serinin model seçimine hazır olduğu görülmektedir. Ancak istatistiksel olarak emin olmak adına Dickey-Fuller birim kök testinin yapılması gerekmektedir.

```
31. from statsmodels.tsa.stattools import adfuller
32. dftest = adfuller(df_diff['Passengers'])
33. dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-
    value','#Lags Used','Number of Observations Used'])
34. for key, value in dftest[4].items():
35.     dfoutput['Critical Value (%s)'%key] = value
36. print(dfoutput)
```

```
Test Statistic                  -5.038002
p-value                          0.000019
#Lags Used                      11.000000
Number of Observations Used    143.000000
Critical Value (1%)             -3.476927
Critical Value (5%)             -2.881973
Critical Value (10%)            -2.577665
dtype: float64
```

Birim kök testi sonucunda "*p*

-value" değerinin 0,05 değerinin altında olduğu görülmektedir. Yani artık zaman serisi durağan hale gelmiştir ve model seçme aşamasına geçilebilir. ACF ve PACF grafikleri yardımıyla aday modellerin parametreleri analizi yapılacak kişi tarafından belirlenebilir.

Yukarıdaki grafiklere bakıldığında p, q ve mevsimsel P, Q parametrelerinin alabilecekleri değerler 0 ile 2 arasında değişebilmektedir. Python kütüphanelerinden biri olan "***pmdarima***" kütüphanesi yardımıyla aday modellerin hepsi denenip en küçük ABK değerine sahip olan model seçimi kolaylıkla yapılabilir. Bu kütüphane, "https://pypi.org/project/pmdarima/" linkinden ulaşılarak indirilebilir ya da Python terminal kısmından "pip install pmdarima" komutuyla kolaylıkla eklenebilir.

```
37. import pmdarima as pm
38. model = pm.auto_arima(df['Passengers'], d=1, D=1,
39.                       m=12, trend='c', seasonal=True,
40.                       start_p=0, start_q=0, max_order=6, test='adf',
```

```
41.                     stepwise=True, trace=True)
```

```
Performing stepwise search to minimize aic
 ARIMA(0,1,0)(1,1,1)[12]             : AIC=1076.128, Time=0.21 sec
 ARIMA(0,1,0)(0,1,0)[12]             : AIC=1121.939, Time=0.01 sec
 ARIMA(1,1,0)(1,1,0)[12]             : AIC=1083.569, Time=0.14 sec
 ARIMA(0,1,1)(0,1,1)[12]             : AIC=1068.286, Time=0.14 sec
 ARIMA(0,1,1)(0,1,0)[12]             : AIC=1116.950, Time=0.05 sec
 ARIMA(0,1,1)(1,1,1)[12]             : AIC=1070.017, Time=0.19 sec
 ARIMA(0,1,1)(0,1,2)[12]             : AIC=1069.963, Time=0.46 sec
 ARIMA(0,1,1)(1,1,0)[12]             : AIC=1084.109, Time=0.15 sec
 ARIMA(0,1,1)(1,1,2)[12]             : AIC=inf, Time=1.17 sec
 ARIMA(0,1,0)(0,1,1)[12]             : AIC=1074.277, Time=0.09 sec
 ARIMA(1,1,1)(0,1,1)[12]             : AIC=1070.056, Time=0.20 sec
 ARIMA(0,1,2)(0,1,1)[12]             : AIC=1069.791, Time=0.20 sec
 ARIMA(1,1,0)(0,1,1)[12]             : AIC=1068.200, Time=0.13 sec
 ARIMA(1,1,0)(0,1,0)[12]             : AIC=1116.800, Time=0.05 sec
 ARIMA(1,1,0)(1,1,1)[12]             : AIC=1069.905, Time=0.21 sec
 ARIMA(1,1,0)(0,1,2)[12]             : AIC=1069.848, Time=0.43 sec
 ARIMA(1,1,0)(1,1,2)[12]             : AIC=inf, Time=1.12 sec
 ARIMA(2,1,0)(0,1,1)[12]             : AIC=1069.912, Time=0.17 sec
 ARIMA(2,1,1)(0,1,1)[12]             : AIC=1068.688, Time=0.47 sec
 ARIMA(1,1,0)(0,1,1)[12] intercept   : AIC=1068.200, Time=0.13 sec

Best model:  ARIMA(1,1,0)(0,1,1)[12] intercept
Total fit time: 5.736 seconds
```

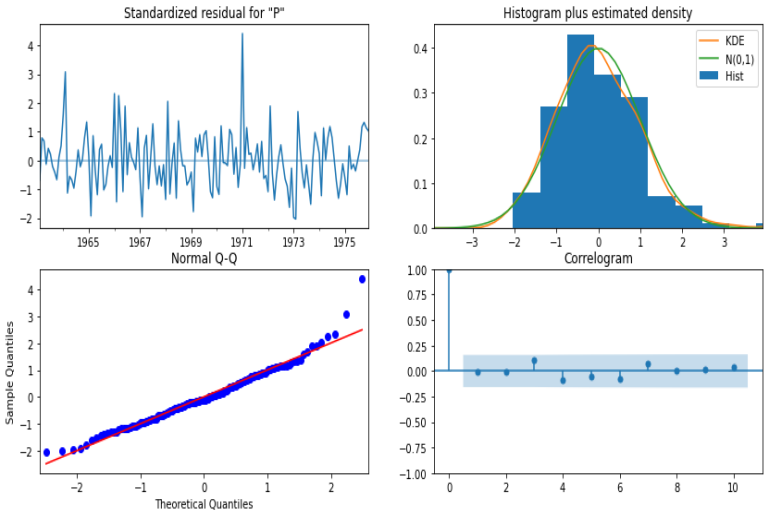
Elde edilen sonuçlar incelendiğinde en uygun modelin yani aday modeller arasından en küçük ABK değerine sahip model $ARIMA(1,1,0)(0,1,1)_{12}$ olarak belirlenmiştir.

```
42. from statsmodels.tsa.statespace.sarimax import SARIMAX
43. model = SARIMAX(df['Passengers'],
44.                 order=(1,1,0),seasonal_order=(0,1,1,12))
45. results = model.fit()
46. results.summary()
```

```
                                     SARIMAX Results
==========================================================================================
Dep. Variable:                          Passengers   No. Observations:                  168
Model:             SARIMAX(1, 1, 0)x(0, 1, [1], 12)   Log Likelihood                -530.104
Date:                              Fri, 15 Jan 2021   AIC                           1066.207
Time:                                      21:57:54   BIC                           1075.337
Sample:                                  01-01-1962   HQIC                          1069.916
                                       - 12-01-1975
Covariance Type:                               opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.2253      0.077     -2.925      0.003      -0.376      -0.074
ma.S.L12      -0.6190      0.070     -8.825      0.000      -0.757      -0.482
sigma2        52.6908      4.897     10.759      0.000      43.093      62.289
===================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):                35.11
Prob(Q):                              0.91   Prob(JB):                         0.00
Heteroskedasticity (H):               0.82   Skew:                             0.74
Prob(H) (two-sided):                  0.49   Kurtosis:                         4.80
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
"""
```

Model belirleme işleminin ardından seçilen modelin istatistiksel olarak anlamlı olup olmadığı test edilmelidir. Artık analizi olarak adlandırılan bu aşamada parametrelerinin belirlendiği modeli oluşturup ardından sonuçlara bakıldığında "*p-value*" değerinin *0,05*' ten küçük olduğu görülmektedir. Yani oluşturulan model istatistiksel olarak anlamlıdır.

```
47. results.plot_diagnostics(figsize=(14, 7))
48. plt.show()
```



Şekil 6. Havayolu Yolcu Verisi Modele İlişkin Sonuç Grafikleri

Elde edilen modelin artıklarına ilişkin grafiklere bakıldığında, artıkların standart normal dağılıma uyduğu yani bir ak gürültü süreci oluşturduğu görülmektedir.

```
49. from sklearn.model_selection import train_test_split
50. train, test = train_test_split(df, test_size=0.15, shuffle=False)
```

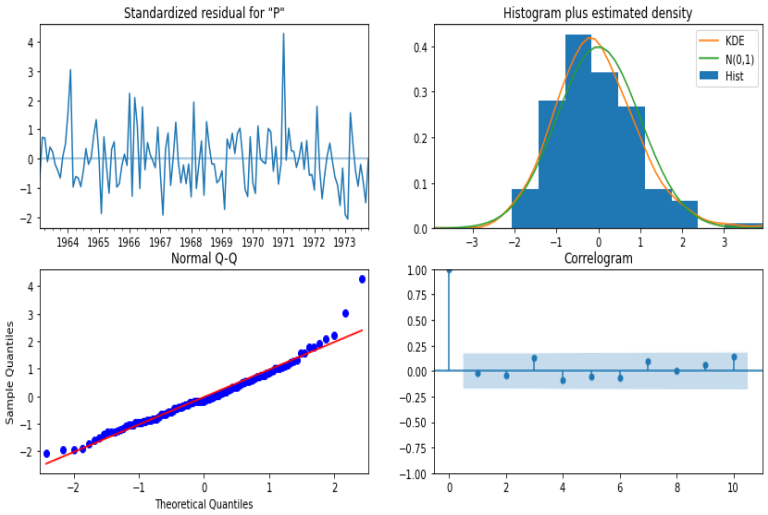
Kurulan modelin geçerliliğini sınamak için veri eğitim ve test olarak 2 kümeye bölünür. % 85 oranında eğitim, % 15 oranında da test için bölünen verinin eğitim kümesinde oluşturulan modelin test kümesinde iyi sonuç çıkarıp çıkaramadığına bakılır.

```
51. from statsmodels.tsa.statespace.sarimax import SARIMAX
52. model = SARIMAX(train['Passengers'],
53.                 order=(1,1,0),seasonal_order=(0,1,1,12))
54. results = model.fit()
55. results.summary()
```

```
                                     SARIMAX Results
==========================================================================================
Dep. Variable:                          Passengers   No. Observations:                  142
Model:             SARIMAX(1, 1, 0)x(0, 1, [1], 12)   Log Likelihood                -445.603
Date:                              Fri, 15 Jan 2021   AIC                            897.205
Time:                                      22:12:36   BIC                            905.785
Sample:                                  01-01-1962   HQIC                           900.691
                                       - 10-01-1973
Covariance Type:                               opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.2728      0.087     -3.149      0.002      -0.443      -0.103
ma.S.L12      -0.6158      0.086     -7.163      0.000      -0.784      -0.447
sigma2        56.0243      5.765      9.719      0.000      44.726      67.323
===================================================================================
Ljung-Box (L1) (Q):                   0.08   Jarque-Bera (JB):                40.04
Prob(Q):                              0.78   Prob(JB):                         0.00
Heteroskedasticity (H):               1.12   Skew:                             0.86
Prob(H) (two-sided):                  0.71   Kurtosis:                         5.12
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

Eğitim kümesi için oluşturulan modelin artık analizi sonucunda da modelin istatistiksel olarak anlamlı olduğu görülmektedir.

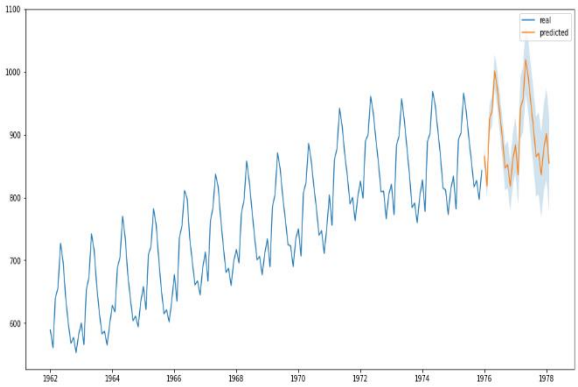
```
56. results.plot_diagnostics(figsize=(14, 7))
57. plt.show()
```



Şekil 7. Havayolu Yolcu Eğitim Kümesine İlişkin Sonuç Grafikleri

Benzer şekilde sonuçların grafiklerine bakıldığında artıkların ak gürültü sürecinde olduğu görülmektedir.

```
58. forecast_object = results.get_forecast(steps=len(test))
59. mean = forecast_object.predicted_mean
60. predictions = mean
61. conf_int = forecast_object.conf_int()
62. dates = mean.index
63.
64. plt.figure(figsize=(16,8))
65. plt.plot(df.index, df, label='real')
66. plt.plot(dates, mean, label='predicted')
67. plt.fill_between(dates, conf_int.iloc[:,0], conf_int.iloc[:,1],alpha=0.2)
68. plt.legend()
69. plt.show()
```



Şekil 8. Havayolu Yolcu Verisi Modelini Test Kümesi Tahmin Grafiği

Eğitim kümesi kullanılarak belirlenen modelin test kümesi boyunca tahminlerinin grafiği yukarıda görülmektedir. Test kümesine ilişkin tahminleri genel yapıya oldukça uygun olduğu açıkça görülmektedir.

```
70. from sklearn.metrics import r2_score
71. r2_score(test['Passengers'], predictions)
```

```
0.924043368649106
```

Sonuçlara ilişki R^2 istatistiği hesaplandığında modelin %92 lik bir uyum performansına sahip olduğu görülmektedir.

```
mean_absolute_percentage_error = np.mean(np.abs(predictions - test['Passengers'])/np.abs(test['
    Passengers']))*100
mape_result = 100-mean_absolute_percentage_error
mape_result
```

```
98.3500945328858
```

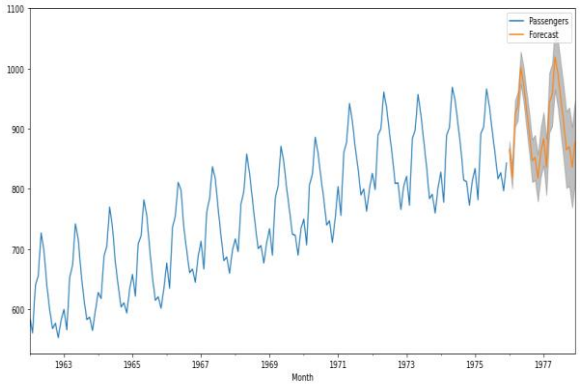
Sonuçlara ilişkin ***MAPE*** istatistiği hesaplandığında ise tahminlerin tutarlılığın %98.3 seviyesinde olduğu görülmektedir.

```
from statsmodels.tsa.statespace.sarimax import SARIMAX
model = SARIMAX(df['Passengers'],
                order=(1,1,0),seasonal_order=(0,1,1,12))
results = model.fit()
results.summary()
```

```
                                     SARIMAX Results
==========================================================================================
Dep. Variable:                          Passengers   No. Observations:                  168
Model:             SARIMAX(1, 1, 0)x(0, 1, [1], 12)   Log Likelihood                -530.104
Date:                              Fri, 15 Jan 2021   AIC                           1066.207
Time:                                      22:26:45   BIC                           1075.337
Sample:                                  01-01-1962   HQIC                          1069.916
                                       - 12-01-1975
Covariance Type:                               opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.2253      0.077     -2.925      0.003      -0.376      -0.074
ma.S.L12      -0.6190      0.070     -8.825      0.000      -0.757      -0.482
sigma2        52.6908      4.897     10.759      0.000      43.093      62.289
===================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):                35.11
Prob(Q):                              0.91   Prob(JB):                         0.00
Heteroskedasticity (H):               0.82   Skew:                             0.74
Prob(H) (two-sided):                  0.49   Kurtosis:                         4.80
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
"""
```

Modelin geçerliliği sınandıktan sonra bütün veri noktalarını içeren modellemeyi tekrar yapıp artık öngörüleri yani gelecekteki gözlemleri tahmin etmeye yönelik son adıma geçilebilir.

```
pred_f = results.get_forecast(steps=24)
pred_ci = pred_f.conf_int()
ax = df.plot(label='Rain', figsize=(14, 7))
pred_f.predicted_mean.plot(ax=ax, label='Forecast')
ax.fill_between(pred_ci.index,
                pred_ci.iloc[:, 0],
                pred_ci.iloc[:, 1], color='k', alpha=.25)

plt.legend()
plt.show()
```



Şekil 9. Havayolu Yolcu Verisi Öngörü Grafiği

Yapılan öngörü işlemi bir adım ötesi öngörü stratejisine dayanan ve 24 adım ötesi öngörüyü içeren grafik yukarıda verilmektedir. Güven aralıklarıyla birlikte elde edilen öngörülerin genel yapıya ve gidişata uygun olduğu görülebilmektedir.

Ancak tabii ki gelecekte ne olacağı konusunda net bilgi verebilen bir model yaklaşımı henüz bilinmemektedir. Geliştirilen yöntemler geçmişi daha iyi anlamaya ve bu yolla geleceği daha tutarlı tahmin etmeye çalışmaktadır. Dolayısıyla modelleme yaklaşımları her zaman yeniliklerle devam etmek ve gelişmek zorundadır.

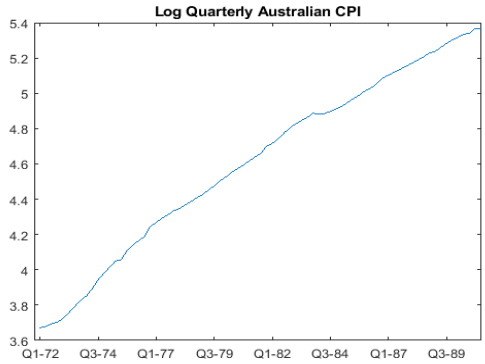
Python kodlarıyla yapılan detaylı analiz mevsimsel Box-Jenkins yani $ARIMA(p,d,q)(P,D,Q)_s$ modeli üzerine örneklendirilmiştir. Daha basit zaman serileri için farklı uygulama yapılmasına gerek duyulmamıştır çünkü analiz temel prensipte aynı olup bölümde verilen kodlar kapsayıcı niteliği taşımaktadır.

Analizler Python 3.9.12 versiyonunda numpy, pandas, matplotlib, pmdarima ve statmodels kütüphaneleri kullanılarak gerçekleştirilmiştir.

2.1.2. MATLAB kodları ile Box-Jenkins modelleri

İkinci Bu bölümde gerçek hayatta da en çok karşılaşılan trend ve mevsimsellik bileşenlerini taşıyan zaman serileri analizi için MATLAB kodları ile birlikte adım adım detaylı analiz yapılacaktır. Kullanılan veri *Avustralya Tüketici Fiyat Endeksinin* 1972 ile 1991 yılları arasında ölçülmüş üç aylık gözlemleri içeren zaman serisidir [7]. İlgili data MATLAB programı içerisinde mevcuttur.

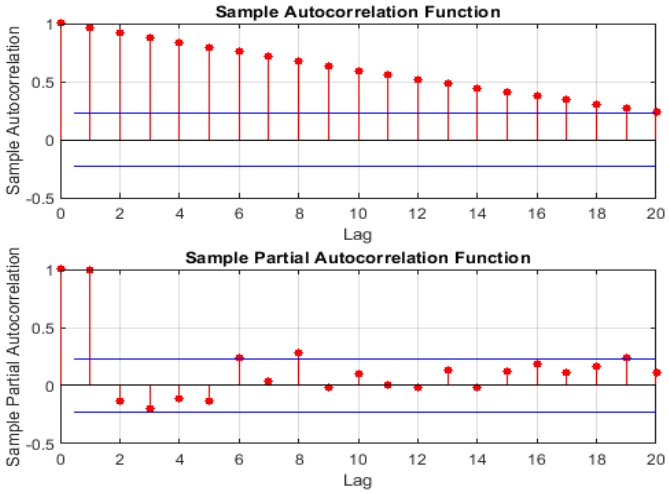
```
load Data_JAustralian
y = DataTable.PAU;
T = length(y);

figure
plot(y)
h1 = gca;
h1.XLim = [0,T];
h1.XTick = 1:10:T;
h1.XTickLabel = datestr(dates(1:10:T),17);
title('Log Quarterly Australian CPI')
```



Şekil 10. Avustralya Tüketici Fiyat Endeksi Veri Grafiği

Verinin grafiğine bakıldığında kolay bir şekilde serinin trend bileşenine sahip olduğu ama herhangi bir mevsimsellik içermediği görülmektedir.

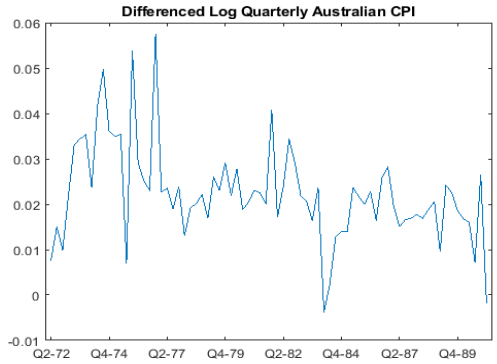
```
figure
subplot(2,1,1)
autocorr(y)
subplot(2,1,2)
parcorr(y)
```



Şekil 11. Avustralya Tüketici Fiyat Endeksi Verisi ACF ve PACF Grafikleri

Serinin ACF ve PACF grafiklerine bakıldığında ise trend bileşeninin olduğu ACF grafiğindeki doğrusal azalmadan dolayı kolayca görülebilmektedir. Ayrıca azalışta herhangi bir dalgalanma olmaması da mevsimsellik bileşeninin olmadığı konusunda bilgi vermektedir.

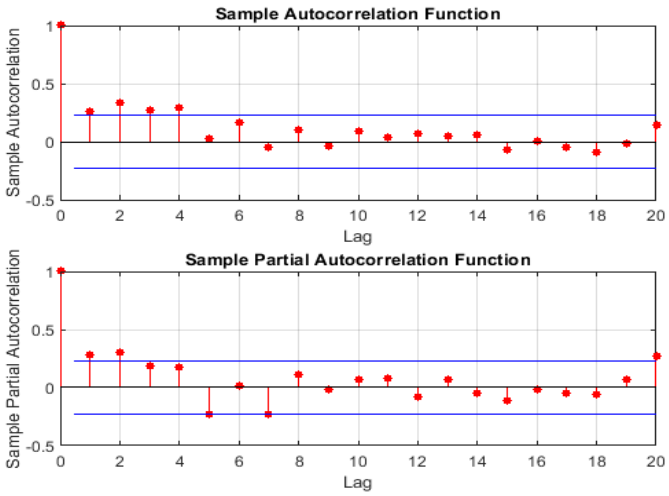
```
dY = diff(y);

figure
plot(dY)
h2 = gca;
h2.XLim = [0,T];
h2.XTick = 1:10:T;
h2.XTickLabel = datestr(dates(2:10:T),17);
title('Differenced Log Quarterly Australian CPI')
```



Şekil 12. Durağanlaştırılmış Avustralya Tüketici Fiyat Endeksi

Zaman serisinin durağanlaştırılması için fark alma işleminin yapılması gerekmektedir. Trend bileşeninden ayrıştırılan ve fark alma işlemiyle durağanlaşan serinin grafiği yukarıda görülmektedir.

```
figure
subplot(2,1,1)
autocorr(dY)
subplot(2,1,2)
parcorr(dY)
```



Şekil 13. Durağanlaştırılmış Avustralya Tüketici Fiyat Endeksi Verisi ACF ve PACF Grafikleri

Durağanlaştırılan serinin ACF ve PACF grafikleri ile uygun model parametreleri belirlenmelidir. MATLAB, Python dilinde hali hazırda olan model bulucu bir fonksiyona sahip olmadığından analizi yapan kişinin grafikler yardımıyla parametreleri belirlemesi gerekmektedir. ACF grafiğindeki düşüşün oldukça hızlı olduğu ve otokorelasyon değerlerinin güven sınırları içerisinde kaldığı kolayca görülebilmektedir. Bunun yanında PACF grafiğinde ilk iki gecikme güven sınırlarının dışında olduğu için bunun bir AR(2) süreci olduğu sonucu çıkarılabilmektedir.

Ancak farklı parametrelere sahip modellerinde anlamlı çıkabilmesi ihtimalinden dolayı uygulamada birden çok modelin analiz edilmesi her zaman faydalıdır.

```
Mdl = arima(2,1,0);
EstMdl = estimate(Mdl,y);
```

```
 ARIMA(2,1,0) Model (Gaussian Distribution):

                Value       StandardError    TStatistic      PValue


Constant      0.010072      0.0032802        3.0707       0.0021357
AR{1}          0.21206       0.095428        2.2222         0.02627
AR{2}          0.33728        0.10378        3.2499       0.0011543
Variance    9.2302e-05     1.1112e-05        8.3066      9.8491e-17
```

AR(2) sürecine ait modelin istatistiksel olarak anlamlı olduğu "p-value" değerlerinin 0,05 ten büyük olmasıyla söylenilebilir. Farklı modellere ilişkin sonuçlarda araştırılmalıdır.

```
Mdl = arima(1,1,0);
EstMdl = estimate(Mdl,y);
```

```
 ARIMA(1,1,0) Model (Gaussian Distribution):

                Value       StandardError    TStatistic      PValue
              __________    _____________    __________    __________

Constant      0.013018      0.0020367        6.3914      1.6439e-10
AR{1}          0.40511       0.077493        5.2278      1.7158e-07
Variance    0.00010516     1.1216e-05        9.3765      6.8177e-21
```

AR(1) sürecine ait modelin de istatistiksel olarak anlamlı olduğu görülmektedir.

```
Mdl = arima(0,1,1);
EstMdl = estimate(Mdl,y);
```

```
 ARIMA(0,1,1) Model (Gaussian Distribution):

                Value       StandardError    TStatistic      PValue
              __________    _____________    __________    __________

Constant      0.020772      0.0011723          17.72      2.9521e-70
MA{1}          0.20929        0.12021        1.7411        0.081668
Variance    0.00012789     1.2962e-05        9.8664      5.8207e-23
```

MA(1) sürecine ait modelin istatistiksel olarak anlamlı olmadığı "*p-value*" değerinin *0,05'* ten büyük olmasıyla açıkça ortadadır.

```
Mdl = arima(0,1,2);
EstMdl = estimate(Mdl,y);
```

```
ARIMA(0,1,2) Model (Gaussian Distribution):

                Value       StandardError    TStatistic      PValue
              __________    _____________    __________    __________

Constant      0.020399      0.0012111          16.844      1.1651e-63
MA{1}          0.18759        0.13047        1.4378          0.1505
MA{2}          0.17724        0.18968       0.93443         0.35008
Variance    0.00012258     1.2182e-05         10.063      8.0555e-24
```

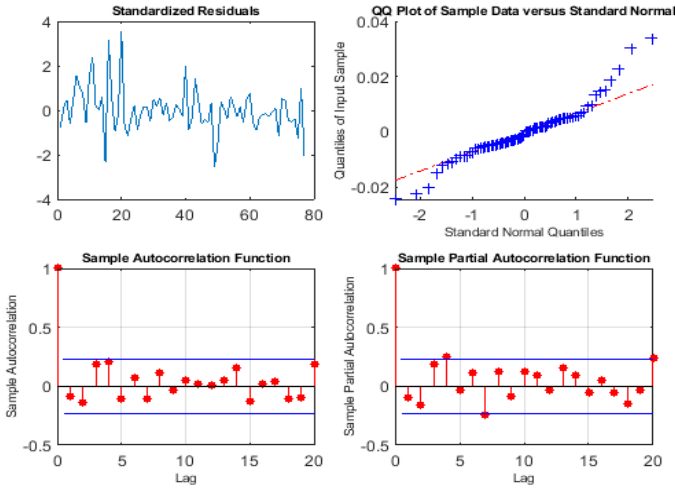
Aynı şekilde ***MA(2)*** sürecininde istatistiksel olarak anlamlı olmadığı görülmektedir.

```
Mdl = arima(1,1,1);
EstMdl = estimate(Mdl,y);
```

```
ARIMA(1,1,1) Model (Gaussian Distribution):

                Value       StandardError    TStatistic      PValue
              __________    _____________    __________    __________

Constant     0.0049721       0.003005         1.6546        0.098008
AR{1}          0.78131        0.12313        6.3456      2.2162e-10
MA{1}         -0.52528        0.15105       -3.4775      0.00050615
Variance    9.1855e-05     1.1188e-05        8.2104      2.2039e-16
```

ARMA(1,1) sürecininde istatistiksel olarak anlamlı olmadığı görülmektedir. Denenen modeller arasında AR(1) ve AR(2) modelinin anlamlı çıktığı ve bu modeller arasında artıkların ak gürültü serisi oluşturduğu model ya da modellerin kullanılabileceği bilinmektedir.

```
res = infer(EstMdl,y);

figure
subplot(2,2,1)
plot(res./sqrt(EstMdl.Variance))
title('Standardized Residuals')
subplot(2,2,2)
qqplot(res)
subplot(2,2,3)
autocorr(res)
subplot(2,2,4)
parcorr(res)

hvec = findall(gcf,'Type','axes');
set(hvec,'TitleFontSizeMultiplier',0.8,...
    'LabelFontSizeMultiplier',0.8);
```



Şekil 14. Avustralya Tüketici Fiyat Endeksi AR(2) Modeline İlişkin Sonuç Grafikleri

AR(2) sürecine ait artık analizinin sonuçlarına bakıldığında artıkların normal dağılıma yakın bir karakteristik sergiledikleri ve dolayısıyla ak gürültü serisi oluşturdukları söylenebilir.

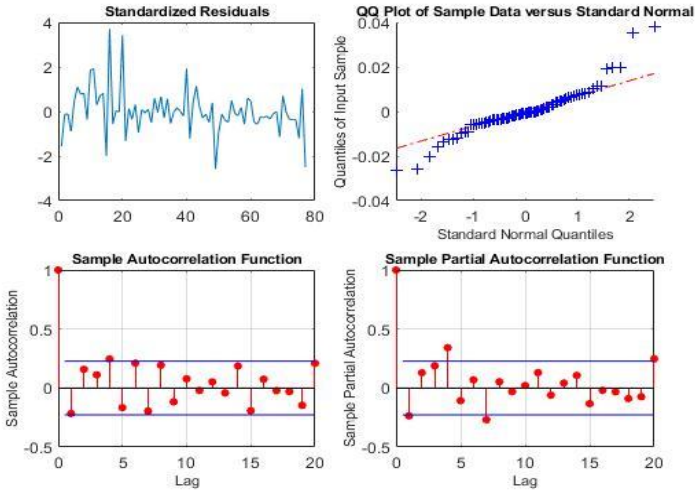
```
[EstMdl,~,logL] = estimate(Mdl,dY,'Display','off');
    results = summarize(EstMdl);
    numParam = results.NumEstimatedParameters;
[~,ic] = aicbic(logL,numParam,length(dY))
```

```
ic =

 -467.3283
```

AR(2) sürecine ait modelin ABK değeri -467.3283 olarak bulunmuştur.

```
res = infer(EstMdl,y);

figure
subplot(2,2,1)
plot(res./sqrt(EstMdl.Variance))
title('Standardized Residuals')
subplot(2,2,2)
qqplot(res)
subplot(2,2,3)
autocorr(res)
```

```
subplot(2,2,4)
parcorr(res)

hvec = findall(gcf,'Type','axes');
set(hvec,'TitleFontSizeMultiplier',0.8,...
    'LabelFontSizeMultiplier',0.8);
```



Şekil 15. Avustralya Tüketici Fiyat Endeksi AR(1) Modeline İlişkin Sonuç Grafikleri

AR(1) sürecine ait artık analizinin sonuçlarına bakıldığında artıkların normal dağılıma yakın bir karakteristik sergiledikleri ve dolayısıyla ak gürültü serisi oluşturdukları söylenebilir.

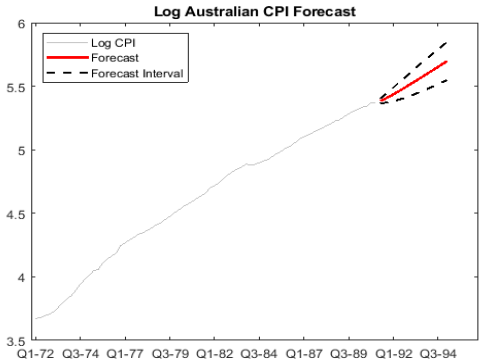
```
[EstMdl,~,logL] = estimate(Mdl,dY,'Display','off');
    results = summarize(EstMdl);
    numParam = results.NumEstimatedParameters;
[~,ic] = aicbic(logL,numParam,length(dY))
```

```
ic =

 -463.9347
```

AR(1) sürecine ait modelin ABK değeri -463.9347 olarak bulunmuştur.

Elde edilen sonuçlarda AR(1) ve AR(2) modellerinin ikiside anlamlı olmasına rağmen AR(2) modelinin ABK değeri daha küçük bulunmuştur. Bundan dolayı veri için en uygun model AR(2) sürecidir.

```
[yF,yMSE] = forecast(EstMdl,16,y);
UB = yF + 1.96*sqrt(yMSE);
LB = yF - 1.96*sqrt(yMSE);

figure
h4 = plot(y,'Color',[.75,.75,.75]);
hold on
h5 = plot(78:93,yF,'r','LineWidth',2);
h6 = plot(78:93,UB,'k--','LineWidth',1.5);
plot(78:93,LB,'k--','LineWidth',1.5);
fDates = [dates; dates(T) + cumsum(diff(dates(T-16:T)))];
h7 = gca;
h7.XTick = 1:10:(T+16);
h7.XTickLabel = datestr(fDates(1:10:end),17);
legend([h4,h5,h6],'Log CPI','Forecast',...
       'Forecast Interval','Location','Northwest')
title('Log Australian CPI Forecast')
hold off
```



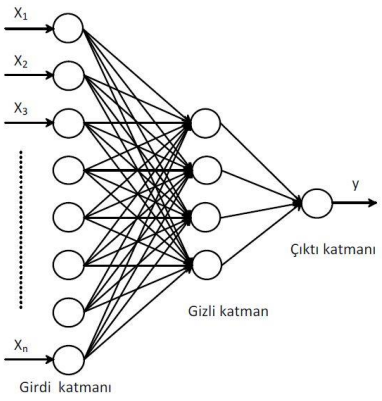
Şekil 16. Avustralya Tüketici Fiyat Endeksi Öngörü Grafiği

Analizin son adımı olan öngörü adımında ise sonraki 16 gözleme ait öngörü değeri elde edilmiş ve yukarıdaki grafikte verilmiştir. Verinin karakteristiğine ve gidişatına uygun bir yapıda olduğu görülmektedir. Analizler MATLAB 2019B sürümünde gerçekleştirilmiştir.

3. Python ve MATLAB kodları ile derin sinir ağları modelleri ve uygulamaları

3.1. Yapay sinir ağları

Genel olarak Yapay Sinir Ağları (*YSA*), insan beynindeki sinir ağlarını taklit eden sezgisel bir yaklaşımdır [8]. Şekil 17.' de *YSA*, beynin davranışlarını taklit eden birbirine bağlı bir mimari şeklinde doğrusal olmayan dinamik bir sistem olarak gösterilmektedir.



Şekil 17. Yapay Sinir Ağı Mimarisi

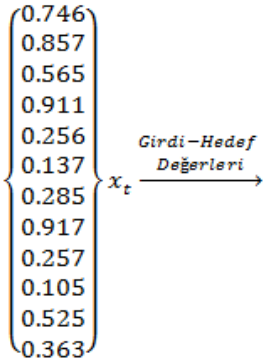
YSA nöron ya da nod olarak adlandırılan ve tabakalar halinde birbirine bağlı yapay sinir hücrelerinden oluşur. Hücrelerin birbirine olan bağlantıları sayesinde bilgi ve sinyal akışı gerçekleşir. Bağlantılar, nöronların birbirleri arasındaki bilgileri taşıyan ağırlık değerlerine sahip birimlerdir. Her bir nöronun çıktısının, sistemin veri yapısını öğrenebilmesi adına diğer nörona iletilmeden önce bir aktivasyon fonksiyonundan geçirilir.

YSA'nın güçlü desen algılayıcı olması ve doğrusal olmayan modelleme yeteneği çokça tercih edilmesinin ana sebeplerindendir. Diğer model tabanlı öngörü yöntemlerinin aksine *YSA* veri odaklı olup herhangi bir varsayım gerektirmeksizin, tahmin edilen değişkenlerle tahmin edilecek değişkenler arasındaki fonksiyonel ilişkilerle ilgilenir. Bu benzersiz niteliği sayesinde *YSA*, genellikle çok veriden oluşan fakat veri oluşum

mekanizması çoğunlukla bilinmeyen ya da test edilmesi mümkün olmayan çeşitli finansal öngörü durumlarında sıklıkla tercih edilmektedir.

Girdi tabakasındaki nöron sayısı sinir ağında kullanılacak serilerin gecikme sayısına eşit olup çıktı tabakasında bulunan tek nörona karşılık gelmesi için ise serinin kendisi hedef değerleri olarak belirlenmektedir.

Çizelge 1. Yapay Sinir Ağlarının Çalışma Prensibi



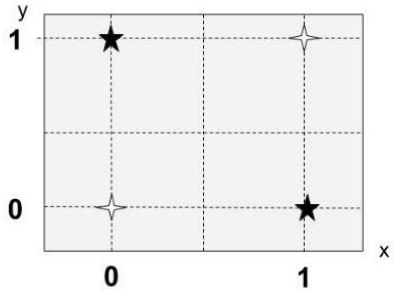
Girdi Değerleri		Hedef Değerleri
X_{t-2}	X_{t-1}	X_t
0,746	0,857	0,565
0,857	0,565	0,911
0,565	0,911	0,256
0,911	0,256	0,137
0,256	0,137	0,285
0,137	0,285	0,917
0,285	0,917	0,257
0,917	0,257	0,105
0,257	0,105	0,525
0,105	0,525	0,363

Çizelge 1.'de görüldüğü üzere 12 değerden oluşan zaman serisinin birinci ve ikinci gecikmeli serisi girdi değerleri olarak girdi tabakasındaki iki nörona denk gelmektedir. Çıktı tabakasındaki nörona denk gelen gecikmeye sahip olmayan seri ise hedef değerlerini oluşturmaktadır [9].

YSA ile zaman serisi analizi için bilinmesi gereken veri yapısı özetle bu şekildedir. Bölümün kalanında verilen uygulamalarda girdi kümesi benzer şekilde oluşturularak ağa girdi olarak verilecektir.

3.1.1. MATLAB kodları ile YSA

İlk YSA modeli olan "Perceptron" (*Tekli Doğrusal Algılayıcılar*) tek bir nöron modeli olup [10] doğrusal bir ayırıcı olarak ilk etapta basit bir *XOR* problemini bile çözemediği için oldukça eleştirilmiş ve uzun yıllar araştırmacıların odağının dışında kalmıştır [11]. Şekil 18. de *XOR* problemi görülmektedir.



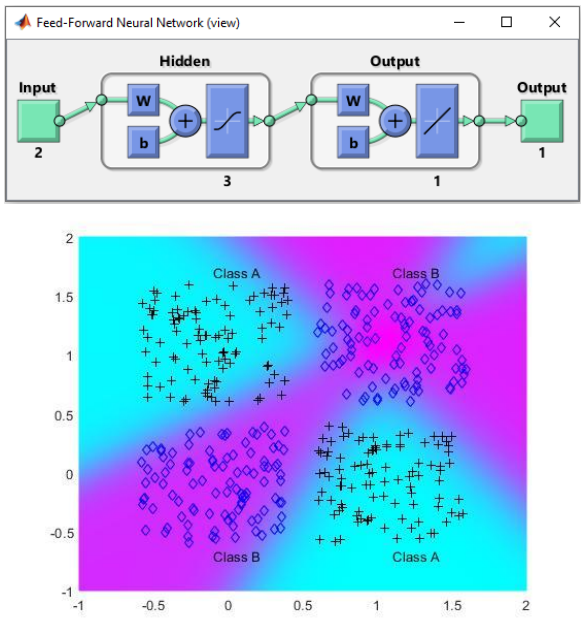
Şekil 18. XOR problemi

Görüldüğü üzere 2 farklı kümeye ait elemanların tek bir doğruyla birbirinden ayrılması olanaksızdır. Perceptron tek bir nöron içerdiği için bu problemi çözmesi mümkün değildir ancak 2 tabakalı bir Perceptron ağı bu problemi çok kolay şekilde çözebilmektedir. Anlaşıldığı üzere Perceptron bir işlem birimi yani

nörondur. Birden çok nöronun tabakalar halinde bir araya getirilerek oluşturulan YSA ise doğrusal olmayan bir yapıya sahip olduğu için *XOR* problemini çok kolay bir şekilde çözebilmektedir.

Aşağıda MATLAB platformunda *XOR* problemi için geliştirilmiş ileri beslemeli sinir ağı kodları verilmektedir.

```
% Her bir sınıfa ait gözlem değeri
K = 100;
% Girdi verisi olarak belirlenen 4 küme
q = .6; % Sınıfların birbirine uzaklıkları
A = [rand(1,K)-q; rand(1,K)+q];
B = [rand(1,K)+q; rand(1,K)+q];
C = [rand(1,K)+q; rand(1,K)-q];
D = [rand(1,K)-q; rand(1,K)-q];
% Kümelerin grafiği
figure(1)
plot(A(1,:),A(2,:),'k+')
hold on
grid on
plot(B(1,:),B(2,:),'bd')
plot(C(1,:),C(2,:),'k+')
plot(D(1,:),D(2,:),'bd')
% Kümelerin isimlendirmeleri
text(.5-q,.5+2*q,'Class A')
text(.5+q,.5+2*q,'Class B')
text(.5+q,.5-2*q,'Class A')
text(.5-q,.5-2*q,'Class B')
% Kümelerin benzer sınıflara atanması.
% a ve c bir sınıf, b ve d diğer sınıf olmak üzere.
a = -1; %  a  |  b
c = -1; %  -------
b =  1; %  d  |  c
d = 1; %
% Girdilerin tanımlanması
P = [A B C D];
% Hedef değerlerinin tanımlanması
T = [repmat(a,1,length(A)) repmat(b,1,length(B)) ...
     repmat(c,1,length(C)) repmat(d,1,length(D)) ];
% Sinir ağının oluşturulması
net = feedforwardnet(3);
% Ağın veri kümelerinin belirlenmesi
net.divideParam.trainRatio = 1; % Eğitim kümesi [%]
net.divideParam.valRatio   = 0; % Doğrulama kümesi [%]
net.divideParam.testRatio  = 0; % Test kümesi [%]
% Ağın eğitilmesi
[net,tr,Y,E] = train(net,P,T);
% Ağın gösterimi
view(net)
figure(2)
plot(T','linewidth',2)
hold on
plot(Y','r--')
grid on
legend('Targets','Network response','location','best')
ylim([-1.25 1.25])
% Koordinat düzleminin oluşturulması
span = -1:.005:2;
[P1,P2] = meshgrid(span,span);
pp = [P1(:) P2(:)]';
% Koordinatlar üzerinde sinir ağının simüle edilmesi
aa = net(pp);
% Sınıfların bölgelerinin çizimi
figure(1)
mesh(P1,P2,reshape(aa,length(span),length(span))-5);
colormap cool
view(2)
```



Şekil 19. İleri Beslemeli Sinir Ağı Mimarisi ve XOR Çözüm Grafiği

Elde edilen sonuçlarda ***2-3-1*** *mimarisine* sahip ileri beslemeli sinir ağının *XOR* problemini kolaylıkla çözdüğü görülmektedir.

3.2. Çok tabakalı perceptron

Çok tabakalı perceptron (*MLP*) genel olarak bir gizli tabakaya sahip ileri beslemeli sinir ağlarının genel adlandırılmasıdır [12]. İleri beslemeli sinir ağlarında sinyal/bilgi akışı tek yönlüdür. Girdi tabasıyla başlayıp girdilerin ağın içine alınmasıyla gizli tabakadaki nöronlarla işlenen bilgi çıktı tabakasından ağın sonucu olarak çıkar. MLP de bütün nöronlar birbirleriyle bağlantılıdır ve sinyal akışının tek yönlü olması da zaman serileri modellemesi açısından oldukça uygun bir yapı oluşturur.

Çok tabakalı ağların temel bileşenleri mimarı olarak adlandırılan model yapısı, aktivasyon fonksiyonu ve öğrenme algoritmasıdır. *YSA'*nın en büyük avantajı öğrenebilme yetisi olup bunu öğrenme algoritması yardımıyla gerçekleştirir. Türeve dayalı geri öğrenme algoritması *YSA* ile eşleştirilmiş bir algoritmadır. Temelde sinir ağının çıktısının yani tahmininin gerçek değerle arasındaki farkı yani "hatayı" küçültmeye dayalı bir strateji içermektedir. Nöronlar arasındaki bağlantı ağırlıklarının çıktı tabakasından girdi tabakasına doğru güncellenmesi prensibine dayanan öğrenme algoritması, her bir adımında hatayı küçülterek çalışır. Hatanın minimize olduğu noktada algoritma durur ve *YSA*'nın öğrenme işlemi tamamlanmış olur.

YSA oldukça esnek bir modüler yapıya sahip sistemlerdir. Dolayısıyla öğrenme algoritması için farklı geri yayılım algoritmaları ile birlikte türeve dayalı olmayan farklı optimizasyon yöntemleri de kullanılabilir. Bunlarla ilişkin örneklere yer verilecektir.

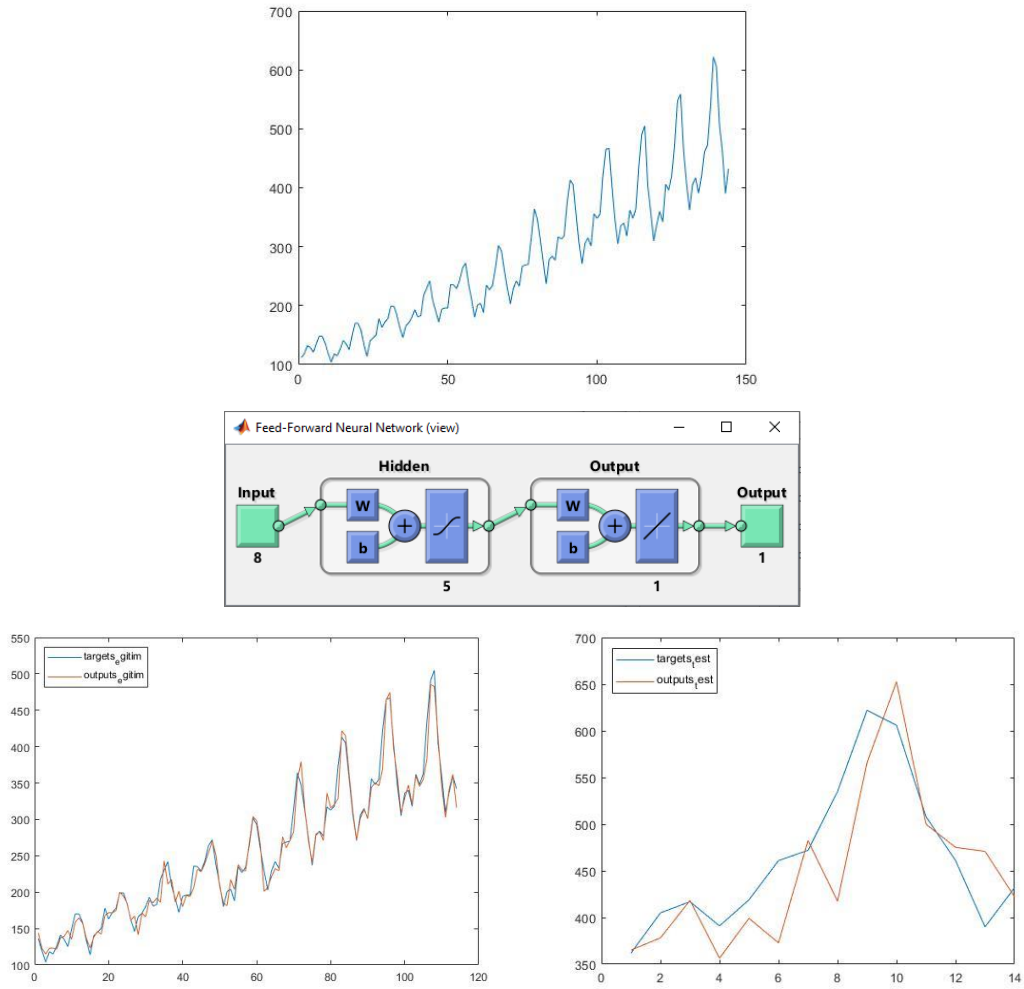
3.2.1. MATLAB Kodları ile MLP

Zaman serisi analizi için kullanılan ileri beslemeli *MLP* yaklaşımına ait MATLAB kodları aşağıda verilmektedir. Kullanılan veri seti önceki bölümde analizi yapılan havayolu yolcuları veri setidir. İndirme linki önceki bölümde paylaşılmıştır.

```
close all, clear all, clc, format compact

% Veri setinin bulunduğu dosya konumundan MATLAB'a aktarılması
```

```
[airpassengersdata]=xlsread('E:\AirPassengers.csv');
% Verinin grafiği
plot(airpassengersdata)
% Gecikme sayısını belirleyen parametre
girdi_gecikme_sayisi=8;
% Eğitim ve test kümesinin büyüklüğünü oransal olarak belirleyen parametre
egitim_kume_orani=0.85;
test_kume_orani=1-egitim_kume_orani;
% Eğitim ve test kümelerinin oluşturulması
egitim_kumesi=airpassengersdata(1:round(length(airpassengersdata)*egitim_kume_orani));
test_kumesi=airpassengersdata(round(length(airpassengersdata)*egitim_kume_orani)+1:end);
     % Eğitim kümesinin Gecikmelerden oluşan girdi matrisini ve hedef vektörünü oluşturan
     algoritma
for p=1:girdi_gecikme_sayisi
    for i=1:p
        for j=1:i
            egitim_girdi{1,j}=egitim_kumesi(j:length(egitim_kumesi)-(p-(j-1)));
        end
        egitim_girdi{2,p}(i,:)=egitim_girdi{1,j};
    end
    %Dizinin birinci kolonu girdi matrislerini içermektedir.
    egitim_kumesi_gecikmeler{p,1}=egitim_girdi{2,p};
    %Dizinin ikinci kolonu hedef vektörlerini içermektedir.
    egitim_kumesi_gecikmeler{p,2}=egitim_kumesi(p+1:length(egitim_kumesi));
end
% Test kümesinin Gecikmelerden oluşan girdi matrisini ve hedef vektörünü oluşturan algoritma
% !! Test kümesi büyüklüğü "girdi_gecikme_sayisi" parametre değerinden büyük olmalıdır.
for p=1:girdi_gecikme_sayisi
    for i=1:p
        for j=1:i
            test_girdi{1,j}=test_kumesi(j:length(test_kumesi)-(p-(j-1)));
        end
        test_girdi{2,p}(i,:)=test_girdi{1,j};
    end
    %Dizinin birinci kolonu girdi matrislerini içermektedir.
    test_kumesi_gecikmeler{p,1}=test_girdi{2,p};
    %Dizinin ikinci kolonu hedef vektörlerini içermektedir.
    test_kumesi_gecikmeler{p,2}=test_kumesi(p+1:length(test_kumesi));
end
% YSA' nın eğitim ve test kümeleri için girdi ve hedef değerlerini belirleyen parametreler
inputs_egitim=egitim_kumesi_gecikmeler{girdi_gecikme_sayisi,1};
targets_egitim=egitim_kumesi_gecikmeler{girdi_gecikme_sayisi,2}';
inputs_test=test_kumesi_gecikmeler{girdi_gecikme_sayisi,1};
targets_test=test_kumesi_gecikmeler{girdi_gecikme_sayisi,2}';

% Gizli tabakadaki nöron sayısını belirleyen parametre
gizli_tabaka_noron_sayisi=5;
% İleri beslemeli YSA'nın oluşturulması
net = feedforwardnet(gizli_tabaka_noron_sayisi);
% MATLAB'ın veri setini kendisi bölmemesi için eğitim kümesi oranı sabit kalmalı.
net.divideParam.trainRatio=1;
% YSA'nın eğitilmesi
[net,tr] = train(net,inputs_egitim,targets_egitim);
% Mimarinin görseli
view(net)
% Eğitilen ağın eğitim kümesi için tahminlerinin elde edilmesi
outputs_egitim = net(inputs_egitim);
```

```
% Eğitim kümesinin HATA KARELER ORTALAMASI cinsinden performansı
YSA_egitim_performans = sqrt(perform(net,outputs_egitim,targets_egitim));
% Eğitilen ağın test kümesi için tahminlerinin elde edilmesi
outputs_test = net(inputs_test);
% Eğitim kümesinin HATA KARELER ORTALAMASI cinsinden performansı
YSA_test_performans = sqrt(perform(net,outputs_test,targets_test));
% Eğitim kümesi ile tahminlerin grafiği
figure
plot(targets_egitim,'DisplayName','targets_egitim');hold on;
plot(outputs_egitim,'DisplayName','outputs_egitim');hold off;
% Test kümesi ile tahminlerin grafiği
figure
plot(targets_test,'DisplayName','targets_test');hold on;
plot(outputs_test,'DisplayName','outputs_test');hold off;
```



Şekil 20. Eğitilen Yapay Sinir Ağına İlişkin Sonuç Grafikleri

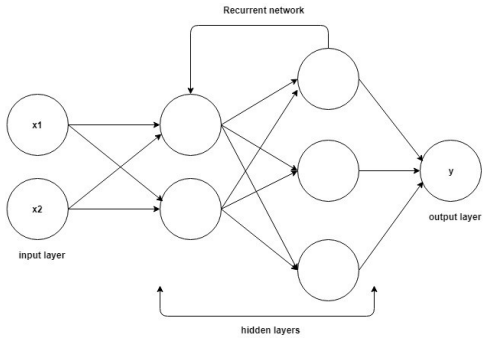
```
YSA_egitim_performans =    13.9358
YSA_test_performans =    50.8702
```

Elde edilen sonuçlarda ***8-5-1*** *mimarisine* sahip olan YSA nın oldukça başarılı sonuçlar ürettiği görülmektedir. Eğitim kümesi hata kare ortalama karekökü değeri 13.9358 ve test kümesi hata kare ortalama karekökü değeri 50.8702 olarak elde edilmiştir. Eğitim kümesi performansı ağın genelleme yeteneğini ve tahminlerin tutarlılığı konusunda belirleyici bir gösterge olmadığından test kümesi

performansına odaklanılmalıdır. Test kümesi hata kare ortalama karekökü performansının da oldukça yeterli olduğu anlaşılmaktadır.

3.3. Yinelemeli sinir ağları

Yinelemeli sinir ağları (*RNN*), *MLP*' den farklı olarak aynı tabakadaki nöronların da birbirleriyle bağlantılı olabilmesini mümkün kılan yapıdaki ağlardır [13]. Buna ek olarak yinelemeli ağlarda tam bağlantı olması zorunlu değildir. Örneğin girdi tabasından direkt çıktı tabakasına bağlantısı olan nöronlar tasarlamak mümkünken ayrıca bazı nöronlar sadece tabaka içerisinde birbirleriyle bağlantılı olup diğer tabakaya bağlantı içermeyebilir. Şekil 1.3. de yinelemeli sinir ağı yapısı görülmektedir.



Şekil 21. Yinelemeli Sinir Ağı Mimarisi

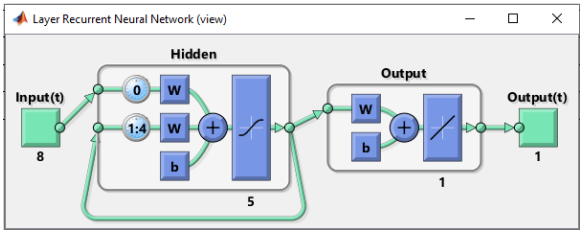
Yinelemeli ağları MLP den ayıran en büyük özelliği nöronların yinelenen bağlantısından dolayı oluşan kısa dönemli sekanslardır. Bu sekanslar kısa dönemli ilişkileri bilgi olarak ağda tutma özelliği sağladığından teoride yinelemeli ağlar zaman serisi modellemesi için oldukça uygun sistemlerdir. Ancak yapısından dolayı sahip olduğu bir dezavantajı ise "gradyan" olarak adlandırılan öğrenme algoritmasındaki türevlerin yinelenen yapıdan dolayı defalarca işlem yapılmasından kaynaklı çok küçülmesidir. Gradyanların çok küçülmesi artık bilgi taşımamaya başladıkları anlamına geldiği için ağın ürettiği sonuçlar anlamsızlaşmaya başlamaktadır. Bundan dolayı yinelenen ağların kullanımında "gradyanların yok olması" problemini önlemek için ek stratejiler uygulamak gerekebilir.

3.3.1 MATLAB kodları ile RNN

Zaman serisi analizi için kullanılan yinelemeli sinir ağları yaklaşımına ait MATLAB kodları aşağıda verilmektedir. Kullanılan veri seti bir önceki bölümde analizi yapılan havayolu yolcuları veri setidir. İndirme linki daha önceki bölümde paylaşılmıştır.

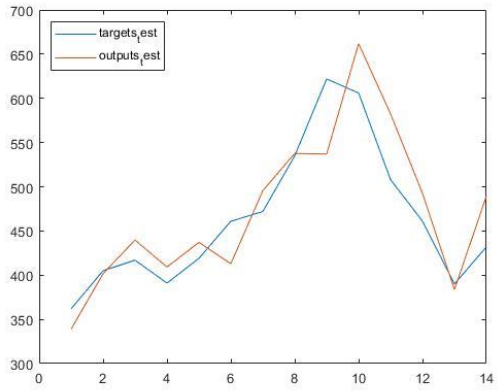
```
close all, clear all, clc, format compact
% Veri setinin bulunduğu dosya konumundan MATLAB'a aktarılması
[airpassengersdata]=xlsread('E:\AirPassengers.csv');
% Verinin grafiği
plot(airpassengersdata)
% Gecikme sayısını belirleyen parametre
girdi_gecikme_sayisi=8;
% Eğitim ve test kümesinin büyüklüğünü oransal olarak belirleyen parametre
egitim_kume_orani=0.85;
test_kume_orani=1-egitim_kume_orani;
% Eğitim ve test kümelerinin oluşturulması
egitim_kumesi=airpassengersdata(1:round(length(airpassengersdata)*egitim_kume_orani));
test_kumesi=airpassengersdata(round(length(airpassengersdata)*egitim_kume_orani)+1:end);
    % Eğitim kümesinin Gecikmelerden oluşan girdi matrisini ve hedef vektörünü oluşturan
    algoritma
for p=1:girdi_gecikme_sayisi
    for i=1:p
        for j=1:i
            egitim_girdi{1,j}=egitim_kumesi(j:length(egitim_kumesi)-(p-(j-1)));
        end
        egitim_girdi{2,p}(i,:)=egitim_girdi{1,j};
```

```
    end
    %Dizinin birinci kolonu girdi matrislerini içermektedir.
    egitim_kumesi_gecikmeler{p,1}=egitim_girdi{2,p};
    %Dizinin ikinci kolonu hedef vektörlerini içermektedir.
    egitim_kumesi_gecikmeler{p,2}=egitim_kumesi(p+1:length(egitim_kumesi));
end
% Test kümesinin Gecikmelerden oluşan girdi matrisini ve hedef vektörünü oluşturan algoritma
% !! Test kümesi büyüklüğü "girdi_gecikme_sayisi" parametre değerinden büyük olmalıdır.
for p=1:girdi gecikme sayisi
    for i=1:p
        for j=1:i
            test_girdi{1,j}=test_kumesi(j:length(test_kumesi)-(p-(j-1)));
        end
        test_girdi{2,p}(i,:)=test_girdi{1,j};
    end
    %Dizinin birinci kolonu girdi matrislerini içermektedir.
    test_kumesi_gecikmeler{p,1}=test_girdi{2,p};
    %Dizinin ikinci kolonu hedef vektörlerini içermektedir.
    test_kumesi_gecikmeler{p,2}=test_kumesi(p+1:length(test_kumesi));
end
% YSA' nın eğitim ve test kümeleri için girdi ve hedef değerlerini belirleyen parametreler
inputs_egitim=egitim_kumesi_gecikmeler{girdi_gecikme_sayisi,1};
targets_egitim=egitim_kumesi_gecikmeler{girdi_gecikme_sayisi,2}';
inputs_test=test_kumesi_gecikmeler{girdi_gecikme_sayisi,1};
targets_test=test_kumesi_gecikmeler{girdi_gecikme_sayisi,2}';

% Gizli tabakadaki nöron sayısını belirleyen parametre
gizli_tabaka_noron_sayisi=5;
% Gizli tabakadaki nöronların kendi içerisindeki bağlantı gecikme parametresi
yinelemeli_gecikme=4;
% Yinelemeli YSA'nın oluşturulması
net = layrecnet(1:yinelemeli_gecikme,gizli_tabaka_noron_sayisi);
% MATLAB'ın veri setini kendisi bölmemesi için eğitim kümesi oranı sabit kalmalı.
net.divideParam.trainRatio=1;
% YSA'nın eğitilmesi
[net,tr] = train(net,inputs_egitim,targets_egitim);
% Mimarinin görseli
view(net)
% Eğitilen ağın eğitim kümesi için tahminlerinin elde edilmesi
outputs_egitim = net(inputs_egitim);
% Eğitim kümesinin HATA KARELER ORTALAMASI cinsinden performansı
YSA_egitim_performans = sqrt(perform(net,outputs_egitim,targets_egitim));
YSA_egitim_performans
% Eğitilen ağın test kümesi için tahminlerinin elde edilmesi
outputs_test = net(inputs_test);
% Eğitim kümesinin HATA KARELER ORTALAMASI cinsinden performansı
YSA_test_performans = sqrt(perform(net,outputs_test,targets_test));
YSA_test_performans
% Eğitim kümesi ile tahminlerin grafiği
figure
    plot(targets_egitim,'DisplayName','targets_egitim');hold
    on;plot(outputs_egitim,'DisplayName','outputs_egitim');hold off;
% Test kümesi ile tahminlerin grafiği
figure
    plot(targets_test,'DisplayName','targets_test');hold
    on;plot(outputs_test,'DisplayName','outputs_test');hold off;
```



Şekil 22. Eğitilen Yinelemeli Sinir Ağı Mimarisi

Ağ yapısı incelendiğinde önceki bölümde kullanılan ileri beslemeli *YSA* ile aynı olan ***8-5-1*** *mimarisinin* kullanıldığını ve gizli tabakadaki nöronların birbirleriyle olan bağlantısına ait yineleme gecikmesinin de 4 seçildiği görülmektedir.



Şekil 23. Test Kümesi Tahminlerine İlişkin Grafik

```
YSA_egitim_performans =    10.1560
YSA_test_performans =   42.0938
```

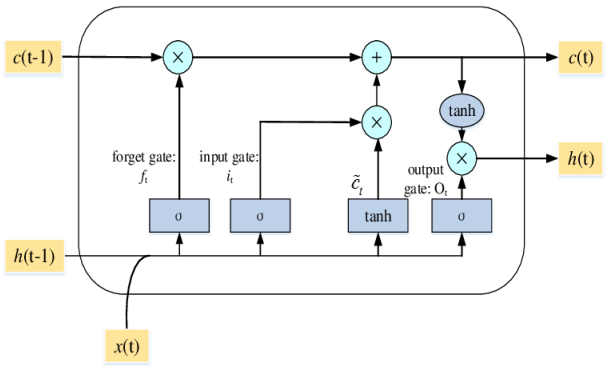
Sonuçlar incelendiğinde aynı mimarideki yinelemeli *YSA'* nın sırasıyla 10.1560, 42.0938 olarak elde edilen eğitim ve test kümesi hata kare ortalama karekökü değerleriyle, yine sırasıyla 13.9358, 50.8702 olarak elde edilen eğitim ve test kümesi hata kare ortalama karekökü değerleriyle ileri beslemeli *YSA*' dan daha iyi performans gösterdiği ayrıca tahminlere ait grafiklerden de görülebilmektedir. Ancak unutulmamalıdır ki bu sonuçlar doğru parametre değerlerinin seçilmesine oldukça bağlıdır. Bir başka deyişle yinelemeli *YSA* her zaman ileri beslemeli *YSA*' dan daha iyi değildir. Yanlış parametre değerlerinin seçilmesi bir çok çalışmada daha kötü sonuçlar elde edilmesi anlamına gelebilir. Benzer durum tüm *YSA* yöntemleri için geçerli olmakla birlikte bu dezavantaj öncelikle *YSA*'nın bağlantı değerlerinin başlangıçta rastgele belirlenmesinden kaynaklanmaktadır. Bu dezavantaj gibi görülen durum aslında model uzayında bulunan olası modellere ulaşılmasını mümkün kılabilen oldukça yararlı bir özelliktir. Fakat *YSA* yöntemlerinin parametre sayısı arttıkça ve mimari yapısı karmaşıklaştıkça en iyi modelin seçilmesi olayının gitgide zorlaştığı da gözden kaçmamalıdır.

4. Derin ağlar

Yıllar içinde değişen ve gelişen ihtiyaçlar doğrultusunda çok daha karmaşık problemler insanoğlunu daha akıllı sistemler geliştirmeye zorlamış ve bu doğrultuda *YSA* evrilerek "derin ağlar" olarak adlandırılan sistemlere genişlemiştir. Temel olarak çok fazla tabakaya sahip YSA olarak adlandırılan derin ağlar, farklı bileşenleri ve yetileriyle karmaşık ve daha yetenekli YSA' lar olarak da adlandırılabilir [14].

4.1. Uzun kısa-dönem hafıza

Uzun kısa-dönem hafıza (*LSTM*)'ler hafıza özelliği kazandırılmış karmaşık nöronlar olarak adlandırılabilir. Bu nöronların birçok tabakada birlikte kullanılmasıyla oluşan *LSTM* derin ağlar yapısı gereği uzun dönem hafızayı ağda tutarak oldukça uzun süreli ilişkileri modelleyebilmektedirler. Her bir *LSTM* nöronu sıradan nöronlardan farklı olarak 1 yerine 3 farklı işlemi yerine getirmektedir. Bu işlemleri "kapı" olarak adlandırılan işlem elemanları yardımıyla yapan *LSTM*; girdi, çıktı ve unutma kapılarıyla hafıza yapısını ağa kazandırıp uzun süre ilişkiyi ağda tutarak "*sekans*" yapısındaki verilerin modellemesinde oldukça başarılı olabilmektedir [15]. Şekil 24. te *LSTM* yapısı görülmektedir.



Şekil 24. LSTM Nöron Modeli

Bu özelliklerinden dolayı *LSTM* zaman serisi modellemesi için oldukça yararlı bir modelleme aracıdır. Oldukça fazla uygulama alanı bulunan *LSTM* ler karmaşık yapılarından dolayı tasarlanması zor derin ağlar olsalar da oldukça yararlı sonuçlar elde edebilmektedirler.

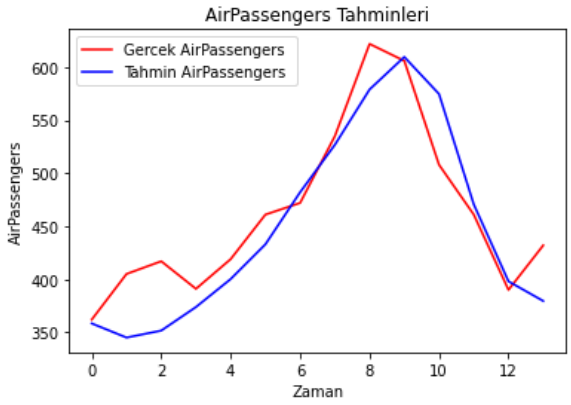
Uzun dönem cümle yapısını öğrenebilen *LSTM* derin ağlarının kullanım alanları oldukça enteresan boyutlarda olabilmektedir. Bunlara verilebilecek en enteresan örneklerden birisi de "Sunspring" olarak adlandırılan *LSTM* tarafında yazılmış yapay kısa film senaryosudur. Yüzlerce filmin senaryosu kullanılarak eğitilen özel bir LSTM sayesinde elde edilen 9 dakikalık senaryonun kısa filmi 2016 yılında yayınlanmıştır. Ayrıca geliştirilen LSTM kendine "Benjamin" ismini koymuştur. Bu bilgiler ışığında *LSTM* gibi derin ağların ne boyutta potansiyel taşıdıkları kolayca anlaşılmaktadır.

4.1.1. Python kodları ile LSTM

Önceki bölümlerde kullanılan havayolu yolcuları veri setinin Python dilinde kodları aşağıda verilmiştir. Veri seti önceki bölümlerde belirtilen linkten indirilebilir.

```
# Part 1 - Data Ön İşleme
# Kütüphanelerin çağırılması
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Eğitim kümesinin okutulması - (datanın son 14 gözlemi test için ayrılmıştır.)
dataset_train = pd.read_csv('AirPassengers_train.csv')
training_set = dataset_train.iloc[:, 1:2].values
# Verinin Ölçeklendirilmesi
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler(feature_range = (0, 1))
training set scaled = sc.fit transform(training set)
# 14 zaman noktası ve 1 çıktı içeren data yapısının oluşturulması
X_train = []
y_train = []
for i in range(14, 130):
    X_train.append(training_set_scaled[i-14:i, 0])
    y_train.append(training_set_scaled[i, 0])
X_train, y_train = np.array(X_train), np.array(y_train)
# Verinin tekrar şekillendirilip analize hazır hale getirilmesi
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))

# Part 2 - LSTM oluşturulması
# Keras kütüphanelerinin ve paketlerin çağırılması
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout

# LSTM bileşenlerinin oluşturulması
regressor = Sequential()
# Birinci LSTM tabakasının eklenmesi ve `Dropout` regularizasyonunun eklenmesi
    regressor.add(LSTM(units = 50, return_sequences = True, input_shape = (X_train.shape[1],
    1)))
```

```
regressor.add(Dropout(0.2))
# İkinci LSTM tabakasının eklenmesi ve `Dropout` regularizasyonunun eklenmesi
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))
# Üçüncü LSTM tabakasının eklenmesi ve `Dropout` regularizasyonunun eklenmesi
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))
# Dördüncü LSTM tabakasının eklenmesi ve `Dropout` regularizasyonunun eklenmesi
regressor.add(LSTM(units = 50))
regressor.add(Dropout(0.2))
# Çıktı tabakasının eklenmesi
regressor.add(Dense(units = 1))
    # LSTM ağının derlenmesi - Öğrenme algoritması `adam` ve amaç fonksiyonu `mse` olarak
    belirlenmiştir.
regressor.compile(optimizer = 'adam', loss = 'mean squared error')
# Eğitim setiyle LSTM ağının eğitilmesi
regressor.fit(X_train, y_train, epochs = 500, batch_size = 50)

# Part 3 - Tahminlerin yapilmasi ve Gorsel plotlarin olusturulmasi

# Gercek datanin olusturulmasi
dataset_test = pd.read_csv('AirPassengers_test.csv')
real_AirPassengers = dataset_test.iloc[:, 1:2].values

# Tahmin değerlerinin oluşturulması
    dataset_total = pd.concat((dataset_train['#Passengers'], dataset_test['#Passengers']),
    axis = 0)
inputs = dataset_total[len(dataset_total) - len(dataset_test) - 14:].values
inputs = inputs.reshape(-1,1)
inputs = sc.transform(inputs)
X_test = []
for i in range(14, 28):
    X_test.append(inputs[i-14:i, 0])
X_test = np.array(X_test)
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
predicted_AirPassengers = regressor.predict(X_test)
predicted_AirPassengers = sc.inverse_transform(predicted_AirPassengers)

# Sonuçların görselleştirilmesi
plt.plot(real_AirPassengers, color = 'red', label = 'Gerçek AirPassengers ')
plt.plot(predicted_AirPassengers, color = 'blue', label = 'Tahmin AirPassengers ')
plt.title('AirPassengers Tahminleri')
plt.xlabel('Zaman')
plt.ylabel('AirPassengers')
plt.legend()
plt.show()

# Test kümesinin hata kare ortalaması
from sklearn.metrics import mean_squared_error
from math import sqrt

rmse = sqrt(mean_squared_error(real_AirPassengers, predicted_AirPassengers))
```



Şekil 25. LSTM Ağı Test Kümesi Tahmin Grafiği

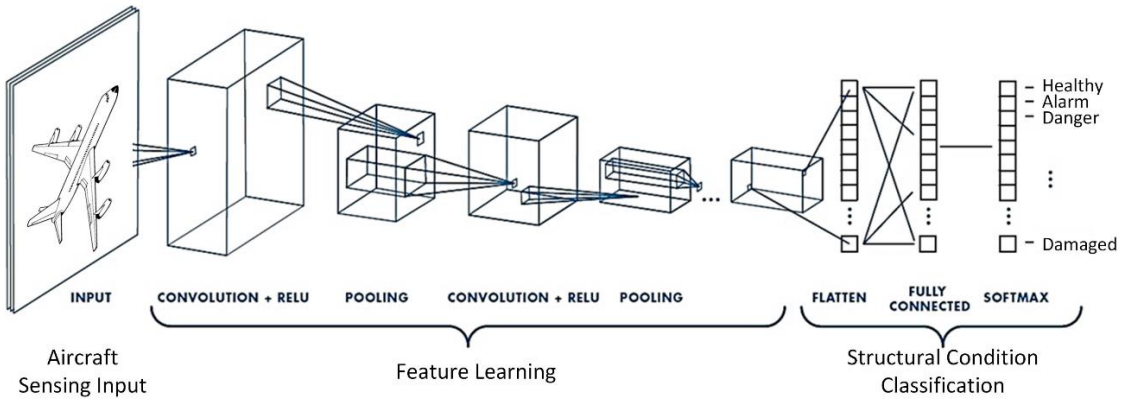
```
Out[]: 36.56466157604625
```

Python ortamında oluşturulan kodlarla eğitilen *LSTM* ağında 4 tabaka ve her bir tabakada 50 adet *LSTM* nöronu kullanılmıştır. Test kümesinin tahminlerine ilişkin grafikten ve 36.5646 olarak elde edilen hata kare ortalama karekökü değerinden görüleceği üzere *LSTM* ağı, yinelemeli *YSA*' dan elde edilen 42.0938 hata kare ortalama karekökü değerinden daha iyi bir performans göstermiştir. Ancak önceki bölümde belirtildiği gibi hangi *YSA* yöntemi kullanılırsa kullanılsın parametre seçimlerine bağlı olarak her zaman daha iyi ya da daha kötü modeller üretmek mümkündür.

Analizler Python 3.9.12 versiyonunda numpy, pandas, matplotlib, sklearn, tensorflow ve keras kütüphaneleri kullanılarak gerçekleştirilmiştir.

4.2. Konvolüsyonel sinir ağları

YSA da çok fazla tabaka kullanımı fikri eski olmamakla birlikte pratik olmayan, çok uzun süren hesaplama, ezberlemeye yatkınlık dezavantajlarından ötürü uzun yıllarca kullanımı tercih edilmemiştir. Ancak konvolüsyonel sinir ağları (*CNN*) olarak adlandırılan "desen öğrenme" odaklı tasarlanmış derin ağlar, sahip oldukları farklı tabaka konfigürasyonları, fonksiyon ve işlem bileşenleriyle başarılı derin ağlar olarak geçmişten günümüze gelişmektedirler [16]. LeCun vd. tarafından 1998 yılından başlayıp günümüze kadar birçok bilindik konvolüsyonel Sinir Ağları geliştirilmiştir. Bunlara örnek olarak *LeNet* [16]*, ImageNet* [17]*, GoogleNet* [18]*, ResNet* [19] verilebilir. Şekil 1.5. ' de konvolüsyonel sinir ağlarının yapısı görülmektedir.



Şekil 26. Konvolüsyonel Sinir Ağları Mimarisi

Temel olarak konvolüsyonel ağlar, fotoğrafın her bir pikselini girdi birimi olarak ağın içine alarak çeşitli işlemlerden geçirir ve birçok tabakadan süzülen bu bilgi ilgili fotoğrafa ait özelliklerin ağ tarafından öğrenilmesinden sonra sınıflama aşamasında belirli bir olasılık değeriyle fotoğrafın hangi cisime ait olduğu tahminini yapar. En bilindik eski ağlardan biri olan "ImageNet" [17], binlerce cismi tanıyabilen 8 tabakalı bir konvolüsyonel sinir ağıdır. Buna benzer olarak sonradan geliştirilen ağlar çok daha karmaşık yapıda daha fazla cisimi tanıyabilen, hızlı yakınsama hızına sahip ve çok daha düşük tahmin hatasıyla çalışmaktadırlar.

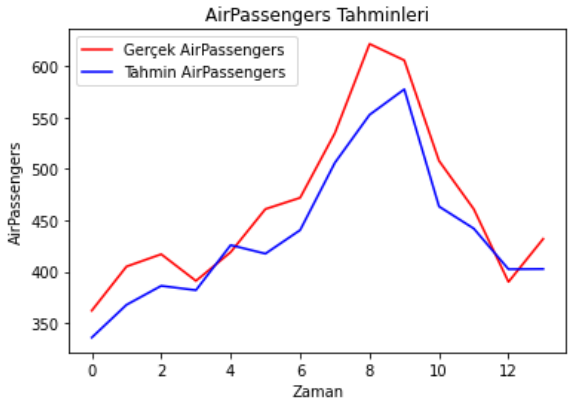
Cisimleri tanımlamak için kullanılan konvolüsyonel ağlar görsellerin ağa tanıtılması için 2 boyutlu olarak tasarlanmış olsa da, girdi yapısının özelleştirilmesiyle farklı amaçlarda da kullanılabilmektedir. Bu noktada 1 boyutlu konvolüsyonel ağlar (*1D-CNN*) zaman serisi modellemesi için kullanılabilmektedir [20].

4.2.1. Python kodları ile 1 boyutlu konvolüsyonel sinir ağları

Daha önceki bölümlerde kullanılan veri seti ile Python ortamında 1 boyutlu konvolüsyonel ağların (1D-CNN) çözümü için gerekli kodlar verilmiştir.

```
# Part 1 - Data Ön İşleme
# Kütüphanelerin çağırılması
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Eğitim kümesinin okutulması - (datanın son 14 gözlemi test için ayrılmıştır.)
dataset_train = pd.read_csv('AirPassengers_train.csv')
training set = dataset train.iloc[:, 1:2].values
# Verinin Ölçeklendirilmesi
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler(feature_range = (0, 1))
training_set_scaled = sc.fit_transform(training_set)
# 14 zaman noktası ve 1 çıktı içeren data yapısının oluşturulması
X train = []
y_train = []
for i in range(14, 130):
    X_train.append(training_set_scaled[i-14:i, 0])
    y_train.append(training_set_scaled[i, 0])
X_train, y_train = np.array(X_train), np.array(y_train)
# Verinin tekrar şekillendirilip analize hazır hale getirilmesi
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))

# Part 2 - 1D-CNN oluşturulması
# Keras kütüphanelerinin ve paketlerin çağırılması
from numpy import array
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers.convolutional import Conv1D
from keras.layers.convolutional import MaxPooling1D

# 1D-CNN bileşenlerinin oluşturulması
model = Sequential()
# Birinci 1D-CNN tabakasının eklenmesi
model.add(Conv1D(filters=64, kernel_size=2, activation='relu'))
# 1D-CNN tabakasına Pooling eklenmesi
model.add(MaxPooling1D(pool_size=2))
# İkinci 1D-CNN tabakasının eklenmesi
model.add(Conv1D(filters=32, kernel_size=2, activation='relu'))
model.add(MaxPooling1D(pool_size=2))
# Düzleştirme tabakasının eklenmesi
model.add(Flatten())
# Çıktı tabakasının oluşturulması
model.add(Dense(50, activation='relu'))
model.add(Dense(1))
    # 1D-CNN ağının derlenmesi - Öğrenme algoritması `adam` ve amaç fonksiyonu `mse` olarak
    belirlenmiştir.
model.compile(optimizer='adam', loss='mse')
# Eğitim setiyle 1D-CNN ağının eğitilmesi
model.fit(X_train, y_train, epochs=1000, verbose=0)

# Part 3 - Tahminlerin yapilmasi ve Gorsel plotlarin olusturulmasi

# Gercek datanin olusturulmasi
dataset_test = pd.read_csv('AirPassengers_test.csv')
real_AirPassengers = dataset_test.iloc[:, 1:2].values

# Tahmin değerlerinin oluşturulması
    dataset_total = pd.concat((dataset_train['#Passengers'], dataset_test['#Passengers']),
    axis = 0)
inputs = dataset_total[len(dataset_total) - len(dataset_test) - 14:].values
inputs = inputs.reshape(-1,1)
inputs = sc.transform(inputs)
X test = []
for i in range(14, 28):
    X_test.append(inputs[i-14:i, 0])
X_test = np.array(X_test)
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
predicted_AirPassengers = model.predict(X_test)
predicted AirPassengers = sc.inverse transform(predicted AirPassengers)

# Sonuçların görselleştirilmesi
plt.plot(real_AirPassengers, color = 'red', label = 'Gerçek AirPassengers ')
plt.plot(predicted_AirPassengers, color = 'blue', label = 'Tahmin AirPassengers ')
plt.title('AirPassengers Tahminleri')
plt.xlabel('Zaman')
```

```
plt.ylabel('AirPassengers')
plt.legend()
plt.show()

# Test kümesinin hata kare ortalaması
from sklearn.metrics import mean_squared_error
from math import sqrt

rmse = sqrt(mean squared error(real AirPassengers, predicted AirPassengers))

Out[]: 33.60583523107709
```



Şekil 27. 1D-CNN Ağı Test Kümesi Tahmin Grafiği

Elde edilen sonuçlar *1D-CNN* ağların zaman serisi öngörüsü için oldukça başarılı sonuçlar çıkardığını ortaya koymaktadır. Bir önceki *LSTM* ağı örneğinde 36.5646 olarak elde edilen test kümesi hata kare ortalama karekökü değeri *1D-CNN* ağında 33.6058 olarak elde edilmiştir. Dolayısıyla bu test kümesi performans değeriyle bütün ağlar arasında en iyi performans *1D-CNN* ağına aittir. Sadece görsel tanımlama değil zaman serisi öngörüsü gibi birçok farklı alanda kullanılabilen konvolüsyonel ağların yüksek potansiyele sahip ve her zaman gelişime açık yaklaşımlar olduğu ortadadır.

4. Sonuç

Zaman serisi analizi, geçmiş gözlemler arasındaki ilişkilerden yola çıkarak geleceğe dair öngörülerde bulunmayı amaçlayan, istatistiksel modellemenin önemli bir alanıdır. Bu çalışmada, zaman serisi analizine yönelik olarak hem geleneksel yöntemler hem de modern yapay zekâ tabanlı yaklaşımlar teorik ve uygulamalı boyutlarıyla ele alınmıştır. Özellikle Box-Jenkins metodolojisi kapsamında *ARIMA* ve *SARIMA* gibi parametrik modeller ile yapay sinir ağları, çok katmanlı yapay sinir ağları ve yinelemeli sinir ağları gibi parametrik olmayan, esnek öğrenme sistemleri karşılaştırmalı olarak incelenmiştir.

Araştırmanın temel bulgularına göre, geleneksel zaman serisi modelleri belirli varsayımlar altında güçlü ve tutarlı tahminler sunabilmekte; özellikle durağan ve mevsimsel özellikleri belirgin olan serilerde, düşük hata oranları ile yüksek uyum performansları elde edilebilmektedir. Bununla birlikte, bu yöntemlerin veri yapısına ilişkin katı varsayımlar içermesi, karmaşık ilişkilerin modellenmesinde sınırlayıcı bir etken olabilmektedir. Bu noktada yapay sinir ağları ve özellikle derin öğrenme tabanlı sistemler, modelleme sürecine ilişkin varsayımları minimize etmeleri ve doğrusal olmayan yapılarla çalışabilme yetkinlikleriyle ön plana çıkmaktadır.

Yapay sinir ağlarının zaman serisi analizine uygulanmasında, özellikle otokorelasyon ve gecikme yapılarının öğrenilmesi sürecinde sağladığı yüksek esneklik, daha kapsayıcı modellerin oluşturulmasına olanak sağlamıştır. Derin ağ mimarileri sayesinde, yalnızca serinin kendisi ile değil, aynı zamanda gecikmeler arasındaki ilişkilerle de anlamlı bağlantılar kurulabilmiş; bu durum tahmin performansını kayda değer şekilde artırmıştır. Python ve MATLAB platformlarında gerçekleştirilen uygulamalar aracılığıyla,

farklı model yapılarına sahip zaman serilerinde kullanılan yöntemlerin başarı düzeyleri sayısal olarak değerlendirilmiş; özellikle R^2 ve MAPE gibi istatistiksel performans ölçütleri bağlamında anlamlı sonuçlara ulaşılmıştır.

Elde edilen bulgular, tek bir yöntemin her tür zaman serisi için evrensel anlamda en iyi modelleme stratejisini sunamayacağını göstermektedir. Nitekim veri setinin yapısı, örneklem büyüklüğü, serinin bileşenleri ve tahminin amacı gibi faktörler, kullanılacak yöntem üzerinde doğrudan belirleyici olmaktadır. Bu çerçevede, çalışmanın bir diğer önemli katkısı, analiz sürecinin yalnızca modelleme adımı ile sınırlı olmadığını, aynı zamanda ön işleme (durağanlaştırma, fark alma), model seçimi (ACF ve PACF grafikleri, bilgi kriterleri), model doğrulama (artık analizi, istatistiksel anlamlılık) ve performans değerlendirme aşamalarının bütünsel olarak ele alınması gerektiğini ortaya koymasıdır.

Sonuç olarak, bu çalışma; zaman serisi analizine ilişkin yöntemsel çeşitliliği ve uygulama esnekliğini ortaya koyarak, araştırmacıların hem geleneksel istatistiksel yaklaşımları hem de modern derin öğrenme modellerini bir arada değerlendirebilmesine olanak tanıyan bütüncül bir çerçeve sunmaktadır. Gelecek çalışmalarda, farklı sektörlere (finans, sağlık, enerji vb.) özgü zaman serilerinin bu modellerle karşılaştırmalı olarak incelenmesi, model entegrasyonlarına (hibrit sistemler) yönelik çalışmalar yapılması ve ileri düzey optimizasyon algoritmalarının model öğrenme süreçlerine entegre edilmesi, literatüre önemli katkılar sağlayabilecektir.

Kaynaklar

[1] Akaike, H. (1974) A New Look at the Statistical Model Identification, *IEEE Transactions On Automatic Control*, 19: 716–723.

[2] Mcculloch, W.S. and Pitts, W.H. (1943) A logical calculus of the ideas immanent in nervous activity, *Bulletin of Mathematical Biophysics*, 5: 115 – 133.

[3] Werbos, P.J. (1974) *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*, PhD thesis, Harvard University.

[4] Rumelhart, D.E., Hinton, G.E. ve Williams, R.J. (1986) Learning representations by back-propagating errors, *Nature*, 323: 533–536.

[5] Box, G.E.P., Jenkins, G.M., ve Reinsel, G.C. (1976) *Time Series Analysis, Forecasting and Control. Third Edition*. Holden-Day, 712s.

[6] Shumway, R. H., Stoffer, D. S., & Stoffer, D. S. (2000). *Time series analysis and its applications (Vol. 3, p. 4)*. New York: Springer.

[7] Australian Bureau of Statistics. (1992). *Consumer Price Index, 1972–1991 [Data set]*. ABS Catalogue No. 6401.0. https://www.abs.gov.au/statistics.

[8] Aladag, C. H., Egrioglu, E., Gunay, S., & Basaran, M. A. (2010). Improving weighted information criterion by using optimization. *Journal of computational and applied mathematics*, 233(10), 2683-2687.

[9] Günay, S., Eğrioğlu, E., Ç.H.Aladağ (2007) *Tek Değişkenli Zaman Serileri Analizine Giriş*, Hacettepe Üniversitesi Yayınları, Ankara, 230s.

[10] Rosenblatt, F. (1958) The Perceptron: A Probabilistic Model For Information Storage And Organization In The Brain, *Psychological Review*, 65: 386 – 408.

[11] Minsky, M., Papert, S. (1969). *Perceptrons: An Introduction to Computational Geometry*. Cambridge, MA, USA: MIT Press.

[12] Bal, C., & Demir, S. (2017). Forecasting TRY/USD exchange rate with various artificial Neural Network Models. *TEM Journal*, 6(1), 11.

[13] Egrioglu, E., Yolcu, U., Aladag, C. H., & Bas, E. (2015). Recurrent multiplicative neuron model artificial neural network for non-linear time series forecasting. *Neural Processing Letters*, 41, 249-258.

[14] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, 521(7553), 436-444.

[15] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8), 1735-1780.

[16] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4), 541-551.

[17] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

[18] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. *In Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9).

[19] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *In Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

[20] Kiranyaz, S., Avci, O., Abdeljaber, O., Ince, T., Gabbouj, M., & Inman, D. J. (2021). 1D convolutional neural networks and applications: A survey. *Mechanical systems and signal processing*, 151, 107398.