TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

TRAFİK SENSÖR VERİLERİ KULLANILARAK TRAFİK AKIŞ TAHMİNİ: İSTANBUL ŞEHRİ İÇİN BİR UYGULAMA

YÜKSEK LİSANS TEZİ

Nezahat SÖNMEZ

Endüstri Mühendisliği Anabilim Dalı

Tez Danışmanı: Prof. Dr. Tahir HANALİOĞLU

NİSAN 2017

Fen Bilimleri Enstitüsü Onayı

……..

Prof. Dr. Osman EROĞUL

Müdür

Bu tezin Yüksek Lisans/Doktora derecesinin tüm gereksininlerini sağladığını onaylarım.

…….

Prof.Dr. Tahir HANALİOĞLU

Anabilimdalı Başkanı

TOBB ETÜ, Fen Bilimleri Enstitüsü'nün 141311014 numaralı Yüksek Lisans Öğrencisi **Nezahat SÖNMEZ**'in ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı **"Trafik Sensör Verileri Kullanılarak Trafik Akış Tahmini: İstanbul Şehri İçin Bir Uygulama"** başlıklı tezi **10/04/2017** tarihinde aşağıda imzaları olan jüri tarafından kabul edilmiştir.

Tez Danışmanı :	**Prof.Dr. Tahir HANALİOĞLU** TOBB Ekonomive Teknoloji Üniversitesi	.....
Eş Danışman :	**Yrd.Doç.Dr. Salih TEKİN** TOBB Ekonomi ve Teknoloji Üniversitesi	.....
Jüri Üyeleri :	**Prof. Dr. Veysel YILMAZ (Başkan)** Eskişehir Osmangazi Üniversitesi	.....
	Prof. Dr. Erdoğan DOĞDU Çankaya Üniversitesi	.....
	Yrd. Doç. Dr. Ahmet Murat ÖZBAYOĞLU TOBB Ekonomi ve Teknoloji Üniversitesi	.....

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, alıntı yapılan kaynaklara eksiksiz atıf yapıldığını, referansların tam olarak belirtildiğini ve ayrıca bu tezin TOBB ETÜ Fen Bilimleri Enstitüsü tez yazım kurallarına uygun olarak hazırlandığını bildiririm.

Nezahat SÖNMEZ

ÖZET

Yüksek Lisans Tezi

Trafik Sensör Verileri Kullanılarak Trafik Akış Tahmini: İstanbul Şehri İçin Bir Uygulama

Nezahat SÖNMEZ

TOBB Ekonomi ve Teknoloji Üniversitesi
Fen Bilimleri Enstitüsü
Endüstri Mühendisliği Anabilim Dalı

Tez Danışmanları: Prof. Dr. Tahir HANALİOĞLU

Tarih: Nisan 2017

Ulaşımın insan yaşamındaki yeri her geçen gün artmakta ve toplumun neredeyse yarısı gününün yaklaşık bir saatini yolda geçirmektedir [1]. Türkiye İstatistik kurumunun açıklamasına göre 2016 Ocak ayı itibari ile İstanbul'da trafiğe kayıtlı motorlu kara taşıtlarının sayısı '3 651 166' gibi bir rakama ulaşmıştır. Trafik akışı üzerine tahminler yapmak, trafik işletme verimliliğini artırmak, trafiğe çıkacak kişilere gidilecek yolu seçmesi konusunda bilgi vermek gibi sebeplerle çalışılmaya başlanmış ve akıllı ulaşım sistemlerinin bir uygulaması olarak oldukça ilgi çekmiştir [2]. İstanbul şehrinde de dünyanın birçok şehrinde olduğu gibi trafik müdürlükleri tarafından gerçek zamanlı trafik verileri çeşitli algılayıcılardan elde edilerek toplanmaktadır. Şeritlerde bulunan araç sayısı, yön bazlı akış hızı, işgaliye miktarı ve şeritlerin hızı gibi değişkenler evrensel bazlı tipik veri seti değişkenleridir [3]. Çalışmada kullanılacak olan veri seti IBB Trafik Müdürlüğü'nden bilimsel çalışma yapmak amacı ile dilekçe yolu ile alınmış olup, İstanbul şehrinin oldukça sıkışık yolları üzerinde yoğunlaşan uzun bir rota izlemektedir. ARIMA ve Derin Çok Katmanlı Algılayıcılar bu çalışmada trafiği modellemek için kullanılmıştır. Veri seti eğitim ve test seti olarak iki parçaya ayrılmıştır. Eğitim seti modeli eğitmek, test ise eğitilen modeli test etmek amacıyla kullanılmıştır. Test set üzerinde yapılan tahminler Çok Katmanlı Algılayıcıların bu

veri seti için, ARIMA modellerine göre çok daha doğru tahminler yaptığı gözlemlenmiştir. Derin öğrenme modelleri, karmaşık sorunları çözme becerileri ile ünlüdür. Model olarak Çok Katmanlı Algılayıcı'yı seçtikten sonra, çalışmanın amacı, sadece o modelle birlikte trafik akışını tahmin etmek olmuştur.

Anahtar Kelimeler: Çok katmanlı algılayıcılar, ARIMA, Trafik akışı modelleme.

ABSTRACT

Master Thesis

Predicting the Traffic Flow with Using Traffic Sensors: An Application for Istanbul City

Nezahat SÖNMEZ

TOBB University of Economics and Technology
Institute of Natural and Applied Sciences
Department of Industrial Engineering

Supervisor: Prof. Dr. Tahir HANALİOĞLU

Date: April 2017

The place of transportation in human life is increasing day by day, and almost half of the society is spending one hour in traffic everyday [1]. According to Statistical Institute of Turkey, the number of motorized road vehicles registered in traffic in Istanbul reaches a number such as '3 651 166'. It has been tried to make estimations on traffic flow, to increase the efficiency of traffic operation, to give information about the way to go to traffic, and has attracted considerable attention as an application of intelligent transport systems [2]. Real time traffic data is collected from various sensors by the traffic directorates in Istanbul, as it is in many cities of the world. Variations such as the number of vehicles in the lines, the direction-based flow rate, the amount of occupation and the speed of the lines are typical universal data set variables [3]. The data set to be used in the study is taken from the IBB Traffic Directorate by means of a petition and an intention to carry out a scientific study and it follows a long route which focuses on the rather cramped roads of the city of Istanbul.

The models which used to modelling traffic are ARIMA and Deep Multilayer Perceptron (DMLP). Data set separated in to parts as training and test sets to train and test models. The estimates on test set showed that DMLP is much more accurate than

ARIMA for this data. DMLP is one of the deep learning algorithms. Deep learning models are famous with their ability to solve complex problems. After choosing DMLP as model, the aim of the study become predicting the traffic flow just with that model.

In the study, it is determined which factors influenced the traffic flow by using artificial neural networks trained and then estimating the traffic flow of future periods.

Keywords: Multilayer perceptrons, ARIMA, Traffic flow modelling

TEŞEKKÜR

Çalışmalarım boyunca bana her zaman destek olan ve yardım eden hocam Yrd. Doç. Dr. Salih Tekin'e, bana bu yolda destek olan hocam Prof. Dr. Tahir Hanalioğlu'na, tecrübelerinden faydalandığım ve bana yol gösteren, sabırla yardım eden hocam Yrd. Doç. Dr. Ahmet Murat Özbayoğlu'na, bana bu yolda ışık tutan ilk adımı atmamı sağlayan sayın hocam Prof. Dr. Veysel Yılmaz'a, TOBB Ekonomi ve Teknoloji Üniversitesi Endüstri Mühendisliği Bölümü Öğretim üyelerine ve arkadaşlarıma, çok teşekkür ederim. Her zaman yanımda olan yardım eden, destek olan anneme, babama, Seher Özcan'a ve Samet Başeğmez'e ayrıca teşekkür ederim.

İÇİNDEKİLER

ŞEKİL LİSTESİ

Sayfa

ÇİZELGE LİSTESİ

Sayfa

RESİM LİSTESİ

Sayfa

1. GİRİŞ

Ulaşımın insan yaşamındaki yeri her geçen gün artmakta ve toplumun neredeyse yarısı gününün yaklaşık bir saatini yolda geçirmektedir [1]. Türkiye İstatistik kurumunun açıklamasına göre 2016 Ocak ayı itibari ile İstanbul'da trafiğe kayıtlı motorlu kara taşıtlarının sayısı '3 651 166' gibi bir rakama ulaşmıştır. Trafik akışı üzerine tahminler yapmak, trafik işletme verimliliğini artırmak, trafiğe çıkacak kişilere gidilecek yolu seçmesi konusunda bilgi vermek gibi sebeplerle çalışılmaya başlanmış ve akıllı ulaşım sistemlerinin bir uygulaması olarak oldukça ilgi çekmiştir [2].

İstanbul şehrinde de dünyanın birçok şehrinde olduğu gibi trafik müdürlükleri tarafından gerçek zamanlı trafik verileri çeşitli algılayıcılardan elde edilerek toplanmaktadır. Şeritlerde bulunan araç sayısı, yön bazlı akış hızı, işgaliye miktarı ve şeritlerin hızı gibi değişkenler evrensel bazlı tipik veri seti değişkenleridir [3]. Çalışmada kullanılacak olan ver**i seti IBB Trafik Müdürlüğü'nden bilimsel çalışma yapmak amacı ile dilekçe yolu ile alınmış olup, İstanbul şehrinin oldukça sıkışık yolları üzerinde yoğunlaşan uzun bir rota izlemektedir. Belirlenen rota için veri seti, Halıcıoğlu semtinden başlayıp Boğaziçi Köprüsünü 'de içine alarak TEM Sultanbeyli Çeşme noktasına kadar uzanan yolda bulunan 23 sensör noktasından elde edilen verileri içermektedir. Çalışmanın amacı belirlenen rotada hangi sebeplerin trafik akışını etkilediğini eğitilen yapay sinir ağları ve ARIMA ile belirleyip. Daha sonra tahmin doğruluğu yüksek modeli seçip, seçilen model ile gelecek dönemlerin trafik akışı hakkında tahminlerde bulunmaktır.

Geçmiş dönemin trafik akış verisindeki eğilimden oldukça etkilenen bu trafik kontrol sistemleri akıllı ulaşım sistemlerinin oldukça önemli bir parçası haline gelmiştir [3]. Trafik akışını belirlemek ve tahminlerde bulunmak için literatürde birçok yöntem denenmiş ve birden fazla metotla %90 'nın üzerinde olasılıklarla tahminler yapılmıştır. ARIMA (Otoregresif Hareketli Ortalama) istatistiksel yöntemi trafik verileri üzerinde oldukça iyi tahminler yapmış ve birçok çalışmada kullanılmıştır [4, 5, 6, 7, 8, 9]. Bayes yaklaşımı ve regresyon gibi istatistiksel yöntemleri kullanıp trafik akışını tahmin eden

çalışmalar bulunmaktadır [10, 3, 5, 7, 9]. Aynı zamanda yapay sinir ağları ve derin öğrenme metotlarının trafik akış tahmini ve kısa süreli trafik tahmini için kullanıldığı çalışmaların sayısı literatürde hatırı sayılır büyüklüktedir. [11, 7, 12, 13], çalışmaları trafik akış tahmini için derin yapay sinir ağlarını en basit şekilde çok katmanlı algılayıcılar olarak kullanırken, diğer bazı çalışmalara daha gelişmiş derin öğrenme yöntemleri ya da basit modelleri geliştirici optimizasyon yöntemleri kullanmışlardır [14, 9, 5, 15, 16, 17].

Trafik akışının tahmini düşünüldüğünden daha karmaşık olabilir, derin öğrenme metotları ön bilgiye gereksinim duymadan bu konuda oldukça başarılı olabilirler [2]. Çalışma [18] da yazarlar oldukça açık bir şekilde sinir ağlarının ön bilgiye ihtiyaç duymayan ve modelin doğrusal olmasını gerektirmeyen yapısı ile trafik akışını modelleme konusunda en iyi seçenek olacağını belirtmişlerdir. Bizim çalışmamızda da yapay sinir ağları derin öğrenme metotları ile birlikte İstanbul şehrinde belirlenen rotanın gelecek günlerde olacak trafik akış davranışını tespit etmekte ve tahminlerde bulunmakta kullanılacaktır.

2. LİTERATÜR TARAMASI

Trafik akışı modelleme, kısa süreli trafikler için gelecek tahmininde bulunma günümüzde insan hayatında çok önemli bir yer tutmaya başlamıştır. Çünkü trafikte geçirilen zamanı azaltmak ve trafikte olması muhtemel kazaların sebebini tespit edip bunları engelleyecek önlemleri almak gibi getirileri vardır. Literatürde birçok yazar birçok farklı yönden trafik akışını ele almıştır. Bu yapılmış çalışmalarda trafik akışını modellemek için genellikle ARIMA, Yapay Sinir Ağlarının yapılandırmaları çeşitlendirilmiştir ya da geleneksel istatiksel yöntemler kullanılmıştır.

Çalışma [10], kısa süreli trafik akışını modellemek için Denetlenen Ağırlıklı Çevrimiçi Destek Vektör Regresyon (online learning weighted support vector regression) öğrenme algoritmasını kullanmışlardır. Destek vektör formülü Gauss çekirdeği ağırlığı dikkate alınarak tekrar gözden geçirilmiştir. Aynı zamanda her bir parametrenin en iyilenmesi için Karush-Kuhn-Tucker(KKT) en iyileme koşullarını kullanarak Lagrange çarpanlarını yazdıktan sonra modele uygun formülasyon elde edilmiştir. Yapılan modellemede 30 sn'lik aralıklarla işgaliye ve yoğunluk içeren trafik verisinin ilk 15 günü eğitim seti 16. gün ise test seti olarak belirlenmiştir. Sonuç olarak trafik akışının normali izlemediği durumlarda tahminleri başarılı olmuştur. Çünkü tarih olarak en yakın veriye en yüksek ağırlığı vererek hızlı bir şekilde beklenmedik değişiklikleri yakalayabilmişlerdir. Başka bir istatistiksel yöntem ise Rekabetçi Beklenti Maksimizasyonu(CEM) algoritması ile parametreleri tahmin edilen Gauss karışım modeli(GMM) ile yapılandırılan ağdaki neden ve etki düğümleri arasında ki ortak olasılık dağılımını kullanmaktır. Çalışma [3], bu yöntemi kullanarak trafik akış tahminlerinde bulunmuştur. Yoğunluk, akış hızı, işgaliye ve hız gibi parametreleri algoritmanın girdi değerleri olarak kullanıp çıktı olarak bir şeridi diğer şeritleri nasıl etkilediğini tespit etmişlerdir.

ARIMA 'nın trafik akış tahmini adına literatürdeki yerinden bahsetmemiz gerekirse, bazı çalışmalar veri setlerini bakarak sadece ARIMA kullanmayı yeterli bulurken [8], birçok çalışma hangi modelin verisine daha iyi uyduğunu anlamak adına ARIMA, Yapay Sinir Ağlarını ve bazıları bir de istatistiksel modelleri bir arada kullanmış [4, 7]

ve verisini en iyi uyan modeli seçmiştir. [4]'te çalışma Beaune, Birleşik Krallık trafik verileri için yapılmıştır. Yapay Sinir Ağları ve ARIMA ayrı ayrı trafik tahmininde kullanılmıştır. Veri setinde periyodik gruplama (günlere göre gruplama) metotları uygun olmamış daha sonra K-Ortalamalar Kümesi yöntemini kullanmışlardır ve Yapay Sinir ağı için gizli katmanda bulunan nöron sayılarına deneyerek karar verdikten sonra iyi sonuçlar almaya başlamışlardır. Başka bir çalışmada ARIMA, Tarihsel ortalama ve Geri Yayılımlı Yapay Sinir Ağları ile ayrı ayrı tahmin yapılmış ve sonuçlar karşılaştırılmıştır. En düşük tahmin hatası %4,3 ile Geri Yayılımlı Yapay Sinir Ağları ile elde edilmiştir. Bu tip karşılaştırmaları trafik akışını tahmini için kullanan başka bir çalışma ise [7] numaralı çalışmadır. Çalışmada ARIMA, En Yakın Komşu Kümelemesi, Tarihsel ortalama ve Geri Yayılımlı Yapay Sinir Ağları ile ayrı ayrı tahmin yapılmış ve sonuçlar karşılaştırılmıştır. Bu çalışmada trafik akış tahminini en iyi yapan iki algoritma En Yakın Komşu Kümelemesi ve Geri Yayılımlı Yapay Sinir Ağları olarak bulunmuştur. Her çalışmanın verisinin başka şehirlerden ya da başka durumlarda ve zamanlarda toplandığını varsayarsak, her çalışmada farklı algoritmaların daha iyi yanıt vermesi mantıklı gelecektir.

Trafik akışını modelleme adına Yapay Sinir Ağlarının kullanımı oldukça ağırlıklıdır, birçok çalışmada Derin Çok Katmanlı Algılayıcılara yer verilirken, bazı veri setleri için bu ağ yapıları yeterli gelmemiş daha karmaşık Derin Öğrenme algoritmaları kullanılmıştır. Derin Çok Katmanlı Algılayıcılar ile modelleme birçok çalışmada oldukça iyi tahminler yapmıştır [6, 7, 11, 12, 13, 16]. Veri setimize uygun Derin Öğrenme modelini eğitmeden önce verilerimizi önişlemden geçirmemiz gerekmektedir, eksik verileri doldurmak, 0-1 arasında normalize etmek, eğitim ve test seti olarak ayırmak gibi [6, 11, 12, 19, 5, 18] çalışmaları veri önişleme kısmına oldukça önem vermişlerdir. Çalışma [6]' da veri önişleme adımlarından geçirildikten sonra uygun ağırlıklar ve eğilimlerin bulunması için yapay sinir ağı 60,000 tur çalıştırılarak eğitilmiştir ve sonuçların doğruluğunu artırmak için çapraz geçerlilik ölçütünü kullanmışlardır. Çalışma [11]' de ise hız, işgaliye ve akış değişkenlerinden oluşan Rotterdam, Hollanda şehri için bulunan trafik verileri üzerinde çalışılmıştır. Bu çalışmada Geri Beslemeli Derin Çok Katmanlı Algılayıcılar ile 5, 15 ve 30 dakikalık trafik tahminleri yapılmıştır. 15 dakikalık tahminlerin oldukça ümit verici olduğu söylenebilir. [20] numaralı çalışmada gene aynı teknikler bu defa Orlando, Florida şehri için yapılan bir çalışmada kullanılmıştır. Bu sefer ki çalışmada yazarlar 1 aylık

(28 günlük) verilere sahip olup bu verilerin 14 günü ile algoritmayı eğitmişler, 4 gününü çapraz geçerlilik ölçütü için kullanmışlar ve 10 günü ise test seti olarak kullanmışlar. Bu örnek verilen çalışmalardan da görülmektedir ki Derin Öğrenme trafik akışı tahmini konusunda birçok çalışmaya ışık tutmuştur. Bu çalışmaların bir seviye üzerinde olan daha karmaşık, sadece geri beslemeli olmayan, bağlanma şekilleri ve optimizasyon algoritmaları daha farklı ve karmaşık olan çalışmalara da literatürde sık sık rastlanmakta olup, oldukça başarılı oldukları söylenebilir.

Trafik akışını belirlemek için literatürdeki çeşitli Derin Öğrenme çalışmalarını inceleyecek olursak, çalışma [9] Bayes Teorisini Savunan Yapay Sinir Ağı kullanmıştır. Bu teknik için 15 dakika aralıklar ile gelen veri setleri 2 ayrı tahmin edici ile modellenmiştir. Bunlar Geri Yayılımlı Yapay Sinir Ağı ve Radyal Tabanlı fonksiyondur. Bu tahmin edicilerin ikisi de Bayesian Sinir Ağı yaklaşımı ile kombinlenmiştir. Çalışmadaki model beş üniteli bir girdi katmanı, 20 üniteli bir gizli katman ve bir üniteli bir çıktı katmanı içerir. Bu yaklaşım ile trafik akış tahminini %10 dan daha düşük hatalarla tahmin etmişlerdir. Çalışma [2]'de ünlü bir Derin Öğrenme modeli olan Otomatik Çoğaltıcı Yığını (Stack of Autoencoders) kullanılmış ve Geri Yayılım algoritması ile eğitilmiş daha sonra Gradyan Tabanlı Optimizasyon yöntemi ile en iyileme yapılmıştır. 15, 30, 45, 60 dakikalık trafik akış tahminleri için modelleri farklı sayıda gizli katman ve farklı sayıda gizli ünitelerle eğitmişler. 15 dakikalık trafik akış tahmininde %90 'ın üzerinde doğruluk elde etmeleri ile birlikte genel olarak bütün akış tahminleri için doğruluk %88 'in üstündedir. Final girdileri Genetik Algoritma ile belirlenen çalışma [5] 'de modelleme için Bölgesel Olarak Ağırlıklandırılmış Regresyon Analizi ve Zaman Gecikmeli Yapay Sinir ağı kullanılmıştır. Çalışma [14] 'de veriyi önişleme adına diğer çalışmalardan farklı olarak Üstsel Düzgünleştirme tekniği kullanılmış. Daha sonra ise Yapay Sinir ağının ağırlıklarını eğitmede ve ağı modellemede Levenberg-Marquardt(LM) algoritması kullanılmıştır. Çalışma [15]'de bizim çalışmamızda kullanacağımız Tekrarlayan Yapay Sinir Ağlarına (Recurrent Neural Network) yapı olarak çok benzeyen Durum Uzayı Yapay Sinir Ağı kullanılmıştır. Bu teknikte x:(t+1) deki durum x:(t) 'ye (t zamanı ifade etmektedir) bakarak açıklanmaktadır. Çalışma [17] Berkeley Üniversitesinde yapılmış olup Grafiksel Yapay Sinir Ağları kullanılmıştır. Los Angeles şehrinden elde edilen sensör verilerini bir grafik olarak tanımladıktan sonra bu veriler için Derin Öğrenme algoritmasını eğitmişler.

Bu çalışmada ise Tekrarlayan Yapay Sinir Ağları ile modelimizi eğittikten sonra her sensör noktasında belli gün ve saatlerde olacak şekilde trafik akış hızını aynı zamanda belli noktalar arası seyahat süresini hesaplamakta eğitilen model kullanılmıştır.

3. VERİ SETİ

Bu çalışmada kullanılan veri seti İstanbul Büyükşehir Belediyesi(İBB) Trafik Müdürlüğüne okulumuz tarafından bir dilekçe yazılarak talep edilmiştir. İBB'den gelen büyük trafik verisi, İstanbul'da birçok trafik noktasına yerleştirilmiş olan trafik sensörlerinden elde edilmiştir.

3.1 Trafik Sensörleri

İBB'den elimize ulaşan trafik verileri Uzaktan Trafik Mikrodalga Sensörleri (RTMS-Remote Traffic Microwave Sensör) ile elde edilen verilerdir. Trafiğin algılanması için tasarlanmış bu rtms sensörleri yola paralel yerleştirildiği zaman 8 şeride kadar ölçme yapabilmekteler [21]. Rtms sensörleri özellikleri [22];

- 8 şeride kadar şerit bazında ölçüm imkânı,
- Araç sayım bilgisi,
- Araç hız bilgisi,
- Trafik yoğunluk bilgisi,
- En az %90 doğru ölçüm,
- Güneş enerjili besleme sistemi ile en az 2 hafta güneş olmadan çalışabilme,
- GPRS veya 3G ile çalışabilme,
- Windows tabanlı yazılım,
- Kalibrasyon kolaylığı,
- Trafik Yoğunluk Haritasına veri sağlar.

Yolun kenarına konumlandırılan rtms sensörleri Resim 3.1'de olduğu gibi araçları tespit eder.

Resim 3.1: Rtms sensör

İstanbul'un 318 noktasına yerleştirilen rtms sensörleri, 1, 2 ve 5 dakikada bir GPRS kullanarak verileri kontrol merkezine göndermektedir. Bütün rtms sensörlerinin harita üzerindeki gösterimi Şekil 3-1'deki gibidir [23];

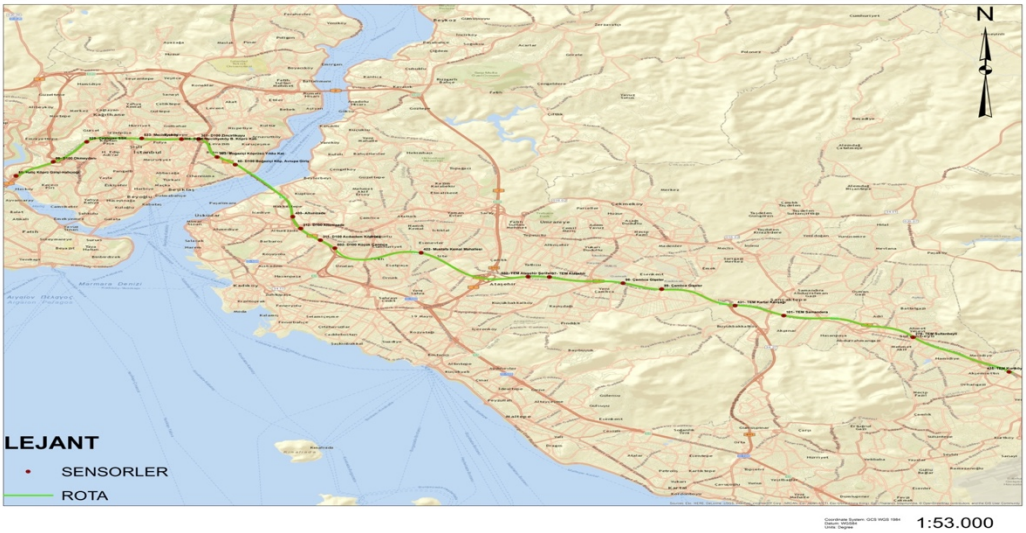
Şekil 3-1: İstanbul rtms sensörler

3.2 Çalışma İçin Seçilen Pilot Bölge

Çalışmada İstanbul trafiğinin günün birçok saatinde oldukça yoğun olduğu ve TEM otoyolu, Boğaziçi köprüsü gibi önemli kilit trafik noktalarını içine alan bir pilot bölge seçilmiştir. Trafik müdürlüğünden alınan veri yaklaşık 600 MB (1048575 satır) olup, çalışma için seçilen bölgede bulunan sensörlerin 4,5 aylık (2015-07-15 & 2015-11-30) verilerini içermektedir. Veri seti hataları verilerden ayıklandıktan ve ön işleme adımlarından sonra 428 MB'a düşmüştür. Pilot bölgede bulunan 23 sensörün rtms sensör numaraları ve sensör isimleri aşağıda ki listede olduğu gibidir.

- 585 -HALICIOĞLU
- 111-D100 Haliç2
- 147-D100 OKMEYDAN
- 97-D100 DARÜLACEZE
- 92-D100 ÇAĞLAYAN
- 115-D100 HÜRRİYET TEPESİ
- 223-MECİDİYEKÖY MEYDANI
- 549-2364 MECİDİYEKÖY
- 141-D100 MECİDİYEKÖY 1
- 564- D100 ZİNCİRLİKUYU
- 81-D100 BOĞAZİÇİ AVRUPA
- 79-BOĞAZİÇİ ANADOLU
- 68-D100 ALTUNİZADE
- 67-D100 ACIBADEM KÖP. 1
- 366-TEM LİBADİYE
- 367- TEM M.KEMAL MAH.
- 308- TEM ATAŞEHİR1
- 331-TEM DUDULLU
- 332- TEM DUDULLU KAVŞAĞI
- 321- TEM ÇAMLICA GİŞELER
- 358-TEM KARTAL KAV.
- 338-TEM SAMANDIRA
- 381- TEM S.BEYLİ ÇEŞME

Pilot bölgedeki sensörlerin İstanbul haritası üzerinde gösterimi Şekil 3-2'de olduğu gibidir.



Şekil 3-2: Pilot bölgede bulunan rtms sensörler

3.3 Rtms Sensörlerinden Gelen Ham Veri

Rtms sensörlerden elimize ulaşan ham ve 1048575 satırdan rastgele olarak seçilmiş örnek yaklaşık 10 satır veri Çizelge 3-1'deki gibidir.

Çizelge 3-1: Rtms sensöründen gelen ham veri

10,15-11-14,22:53,51,6,42,41,37,38,38,NULL,NULL,NULL,39,37,57,53,52,44,48,0,0,0,29,29,33,26,20,0,0,0,29,26,1,10,7,5,2,0,0,0,166,64
11,15-11-14,22:55,51,6,43,37,48,43,45,NULL,NULL,NULL,35,45,54,54,56,53,47,0,0,0,31,33,24,24,17,0,0,0,32,21,0,7,3,1,1,0,0,0,167,64
12,15-11-14,22:56,51,6,44,41,37,40,38,NULL,NULL,NULL,39,38,58,57,53,45,47,0,0,0,30,30,29,22,40,0,0,0,30,30,1,9,0,1,4,0,0,0,168,64
13,15-11-14,22:59,51,6,45,45,49,43,45,NULL,NULL,NULL,41,45,55,55,54,47,49,0,0,0,29,30,22,23,19,0,0,0,29,21,1,9,2,5,3,0,0,0,169,64

Çizelge 3-1: (Devam) Rtms sensöründen gelen ham veri

14,15-11-14,23:01,51,6,46,24,37,35,38,NULL,NULL,NULL,24,36,41,44,48,44,41,0,0,0,36,50,29,31,22,0,0,0,43,27,4,5,2,5,4,0,0,0,170,64
15,15-11-14,23:03,51,6,47,35,43,37,37,NULL,NULL,NULL,36,39,56,55,49,44,45,0,0,0,30,35,23,24,26,0,0,0,32,24,2,9,2,2,2,0,0,0,171,64
16,15-11-14,23:05,51,6,48,43,46,38,38,NULL,NULL,NULL,39,40,60,58,54,47,48,0,0,0,32,32,22,27,23,0,0,0,32,24,0,15,4,11,2,0,0,0,172,64
17,15-11-14,23:07,51,6,49,38,38,37,37,NULL,NULL,NULL,36,37,55,55,45,45,44,0,0,0,30,34,28,26,26,0,0,0,32,26,2,6,2,2,3,0,0,0,173,64

Veri ön işlemesinin ilk adımı olarak virgül ile ayrılmış veri (Comma separated value-CSV) formatını başlıkları da eklediğimiz sütunlara ayırmak gerekmektedir. Daha sonraki bölümde ayrıntılı görülecektir ki veride oldukça fazla geçersiz (null) değişken hatta tamamı geçersiz olan sütunlar vardır.

Çizelge 3-2: Sütunlara ayrılmış veri seti

MsgTime	RtmsNo	S1	S2	S3	S4	S5	S6	S7	S8	SGelisGraf	SGidisGraf	V1	V2	V3	V4	V5
2016-06-09 13:50:31.000	529	69	74	77	83	83	72	69	NULL	75	74	38	46	46	48	57
2016-06-09 13:48:31.000	529	64	74	75	83	82	69	64	NULL	74	71	46	35	36	44	69
2016-06-09 13:46:31.000	529	65	72	75	80	83	70	69	NULL	73	74	33	50	48	51	58
2016-06-09 13:44:32.000	529	65	64	78	83	83	75	69	NULL	72	75	43	36	45	45	55
2016-06-09 13:42:31.000	529	62	67	72	82	80	74	69	NULL	70	74	43	47	47	43	67
2016-06-09 13:40:32.000	529	67	72	77	88	82	70	64	NULL	76	72	37	37	36	49	69
2016-06-09 13:38:31.000	529	65	70	75	85	83	75	67	NULL	73	75	42	42	49	42	68
2016-06-09 13:36:32.000	529	62	67	74	80	85	75	69	NULL	70	76	40	47	47	52	57
2016-06-09 13:34:31.000	529	69	70	77	80	86	80	72	NULL	74	79	33	41	48	49	67
2016-06-09 13:32:32.000	529	67	72	78	90	86	74	67	NULL	76	75	33	41	45	32	65

Çizelge 3-2'de csv formatında gelen ham verinin tablo haline getirilmiş ve başlıkları eklenmiş halidir. Çizelge 3-2'deki veri setinde bulunan bütün sütunların isimleri, etiketleri, veri tipleri ve değişken açıklamaları Çizelge 3-3'te verilmiştir.

Çizelge 3-3: Rtms veri tanımlama tablosu

RTMS Veri Tanımı			
Değişken	**Etiket**	**Veri Tipi**	**Açıklamalar**
Date	Tarih	datetime64	Her sensörden gelen her kaydın saati ve tarihi.
rtmsno	RTMS No	int64	Veri gelen rtms sensörünün numarası.
S_i	Hız	float64	Şerit bazlı araç hızları. Şerit numaraları; i = 1,2,3,4,5,6,7,8
$S_{gelisgraf}$	Ortalama Hız	float64	Geliş yönü için ortalama hız.
$S_{gidisgraf}$	Ortalama Hız	float64	Gidiş yönü için ortalama hız.
V_i	Araç Sayısı	float64	Şerit bazlı araç sayıları. Şerit numaraları; i = 1,2,3,4,5,6,7,8
O_i	İşgaliyet	float64	Şerit bazlı ortalama işgaliyet değeri % olarak. Şerit numaraları; i = 1,2,3,4,5,6,7,8
$O_{gelisgraf}$	Toplam İşgaliyet	float64	Geliş yönü için tüm şeritlerin toplam işgaliyeti.
$O_{gidisgraf}$	Toplam İşgaliyet	float64	Gidiş yönü için tüm şeritlerin toplam işgaliyeti.
msgno	Mesaj Numarası	int64	Verinin geldiği, o sensörü ve o saati temsil eden mesaj numarası.
status code	Statü kodu	int64	Sensörün çalışıp, çalışmadığı anlaşılan statü kodu.

Veri setinin içerisinde tablodan gösterilen değişkenler dışında araçların uzunlukları (kısa-orta-uzun araç sayıları şeklinde) ile ilgili veride bulunmaktaydı, fakat yaklaşık %95 'i geçersiz(null) olarak ulaşması sebebi ile bu bilgi veri setinden çıkarıldı.

3.4 Analizler İçin Veri Önişleme Adımları

Araç uzunlukları ile ilgili geçersiz gelen veriler çıkarıldıktan sonra ilk adım olarak veriler sensör bazlı ayrı ayrı klasörler haline getirilmiştir. Bunun sebebi her yol ayrı ayrı incelenmelidir ve ortalama varış süresi hesaplanacağı zaman her olası yola bağlanma ve yoldan ayrılma durumlarını göze alabilmek için her yoldaki ortalama hızlar ayrı ayrı tahmin edilip seçilen iki varış noktası arası süre hesaplanabilir. İkinci adım olarak tarih ve saat iki ayrı sütun haline getirilmiştir. Bunun sebebi günün her saatini ayrı ayrı görebilmek ve sistemi eğitirken saat değişimini de öğrenebilmesini sağlamaktır. Tarih değişkeni ise Derin Öğrenme modelinde kullanılabilmesi için ve aynı zamanda kolaylık sağlanması için gg/aa/yyyy formatından 'Julian Tarih' denilen formata dönüştürülmüştür. Julian tarih formatı ay, gün değişkenleri yerine yılın günlerine 1'den 365'e kadar numara verip aylar ve günleri bu numaralar ile ifade etmektedir. Bu adımların tamamı R Programlama dili ile yapılmıştır. Kodları EK 1'de verilmiştir. Saat ve tarih kısmında bazı hatalı (nan) veya geçersiz (null) veriler bulunmakta idi, bu veriler Python'da aşağıdaki komut ile bir önceki var olan tarih ve saat ile doldurulmuştur.

```
df.fillna(method='pad', limit=1)
```

Veri setini daha iyi anlamak için ortalama, standart sapma, korelasyonlar gibi betimleyici istatistikler hesaplandığı zaman standart sapmanın büyüklüğünden ve bazı değerler arasında olmaması gereken aşırı değişimden görülmüştür ki data birçok hatalı veri içermektedir. Bu hatalı değişkenlere örnek Çizelge 3-4'te veri setinin içerisinden içinde hatalı değişken bulunduran ve rastgele seçilmiş olan 10 örneklem değeri gösterilmektedir.

Çizelge 3-4: Örnek hatalı veri

index	360	361	362	363	364	365	366	367	368	369
s_gelis	76	81	76	80	**14**	80	83	81	89	87

Çizelge 3-4'ün 364 numaralı indexinde bulunan 14 değeri gibi değişkenlerin hatalı olduğu Java programlama dili ile yazılan basit bir algoritma ile tespit edilip, düzeltirmiştir. Algortimanın sözde kodu işleyişini görmek adına, aşağıda açıklanmış olup, java kodu ise EK 2' dedir.

```
1:  For i = 0 to N do
2:      For t=0 to length.N[i] do
3:          List ← N[i]
4:          If length.List >9 do
5:              C ← List[i]
6:                  A ← C/10
7:                      İf (N[i] < A-45 or N[i] > A+45) → MARK
8:          List → 0
```

Bu sözde kodu daha iyi anlamak adına, burada 'N' veri setinde bulunan sütun sayısını göstermektedir. Bu algoritmada veri 10 birimlik gruplar halinde incelenmektedir. İncelemede 10 birimlik grubun ortalaması alınır daha sonra bu 10 birimden ortalamanın 45 birim altında ya da 45 birim üstünde olanlar hatalı veri olarak işaretlenir.

Rtms sensörlerinden gelen veri tamamen hatalı verilerden ayıklanıp, boşluk değerleri doldurulduktan sonra veriye trafik akışının ortalama hızını etkileyebileceği düşünülen yeni veriler eklenmiştir, bunlar hava durumu, hava koşulu ve özel gün (resmî tatil) verileridir.

3.5 Veri Setine Hava Durumu Verisi Ekleme

Veri setimize hava durumu verisi eklemek için Java programlamada "OpenWeatherMap" [24] API'si kullanarak İstanbul için veri setinin tarih aralığında sorgu yapılmıştır. Yapılan sorgu ile gelen hava durumu verisi ve hava koşulu (yağmur, kar, fırtına) verisi veri setimize SQL birleştirme tekniği ile eklenmiştir.

Örnek API sorgusu:

http://history.openweathermap.org/data/2.5/history/city?q={cityID},{countrycode}&type=hour&start={start}&end={end}

Parametreler:

- q : Şehir adı, Ülke adı
- type : Sorgu tipi, örneğin tipi "hour" belirler isek saatlik sorgu yapmaktadır.
- start : Sorgunun başlangıç tarihi

- end : Sorgunun bitiş tarihi
- cnt : Dönen veri miktarı

3.6 Veri Setine Özel Gün Verisi Ekleme

Bilinmektedir ki birçok resmî tatil başlangıcında ve bitişinde yollardaki araç trafiği sonu gelmeyen kuyruklar oluşturmaktadır. Ülkemizde resmî tatillerde trafikte nasıl farklılaşmalar olduğunu görmek ve yolun ortalama hızını nasıl etkilediğini görmek adına veri setine bu verinin de eklenmesine karar verilmiştir. Veri seti 2016 yılının verisi içerdiği için sadece 2016 yılının resmî tatilleri dikkate alınmıştır. Çizelge 3-5'te 2017 yılının tüm özel günleri listelenmektedir.

Çizelge 3-5: Resmi gün ve tatiller

TATİL GÜNÜNÜN İSMİ	**SÜRE**	**AY**	**GÜN**
Yılbaşı	1. Gün	1 Ocak	Cuma
Ulusal Egemenlik Ve Çocuk Bayramı	1 Gün	23 Nisan	Cumartesi
Emek Ve Dayanışma Günü	1 Gün	1 Mayıs	Pazar
Atatürk'ü Anma Gençlik Ve Spor Bayramı	1 Gün	19 Mayıs	Perşembe
Ramazan Bayramı Arifesi	1/2 Gün	04 Temmuz	Pazartesi
Ramazan Bayramı	1. Gün	05 Temmuz	Salı
Ramazan Bayramı	2. Gün	06 Temmuz	Çarşamba
Ramazan Bayramı	3. Gün	07 Temmuz	Perşembe
Zafer Bayramı	1 Gün	30 Ağustos	Salı
Kurban Bayramı Arifesi	1/2 Gün	11 Eylül	Pazar
Kurban Bayramı	1. Gün	12 Eylül	Pazartesi
Kurban Bayramı	2. Gün	13 Eylül	Salı
Kurban Bayramı	3. Gün	14 Eylül	Çarşamba
Kurban Bayramı	4. Gün	15 Eylül	Perşembe
Cumhuriyet Bayramı	1,5 Gün	28 Ekim	Cuma
		29 Ekim	Cumartesi

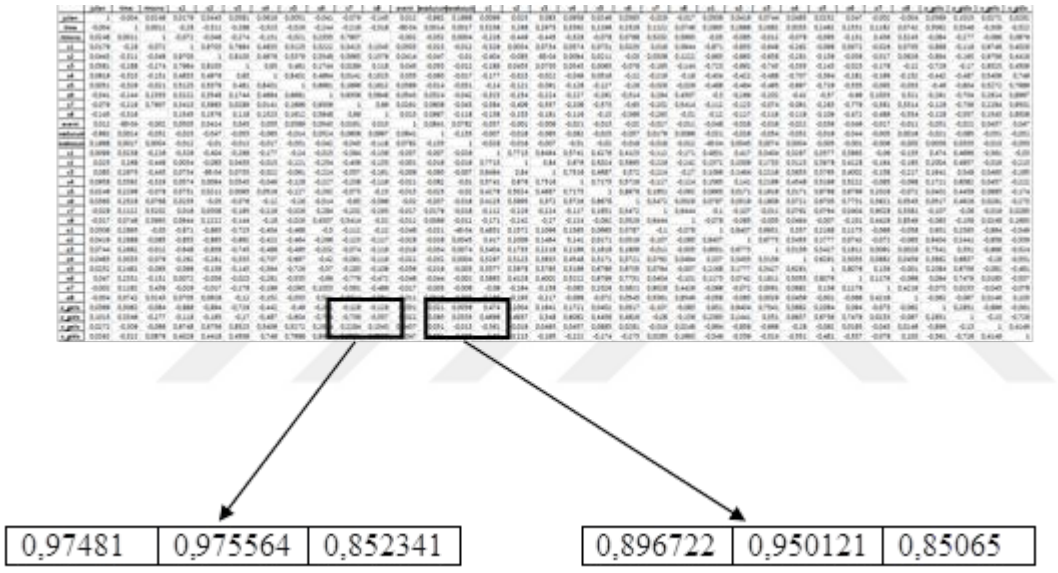
3.7 Hazırlanan Verinin Analizi Ve İncelenmesi

Verinin içerisinde rtms sensörlerinden gelen verileri ve bizim sonradan eklediğimiz verilerin birbirinden ne derece ve hangi yönlü etkilediklerini görmek adına birbirlerine olan korelasyon değerleri incelenmiştir. Aynı zamanda verinin ana hatlarını görebilmek adına büyük verimizin betimleyici istatistikleri hesaplanmıştır.

3.7.1 Korelasyonlardan bazıları

34 x 34'lük korelasyon tablosu Çizelge 3-6' da gösterildiği gibidir. Bu çizelge oldukça büyük bir çizelge olduğu için bu kısımda tamamı gösterilememiştir.

Çizelge 3-6: Korelasyon tablosu



Çizelge 3-6'da gösterildiği üzere bağımsız değişkenler arasında oldukça yüksek korelasyonlar görülmektedir. Bu gereksiz (redundant) değişkenleri veri setimizden çıkarmak, hem veri setini daha sade hale getirmek hem de aşırı uyum gösterme (overfitting) denilen durumu engellemek için gereklidir. Korelasyon analizi sırasında eklenen özellik eleme (feature elimination) filtresi ile 's1, s2, s3, s4, s5, o1, o2, o3' değişkenleri çıkartılmıştır. Çıkartılan değişkenlerden sonra tekrar korelasyonlar hesaplanmıştır, yeni korelasyonlar Çizelge 3-7'de bulunabilir.

Çizelge 3-7: Gereksiz değişkenler çıkarıldıktan sonra korelasyonlar

	julian	time	s6	s7	s8	event	h.durum	h.kosulu	v1	v2	v3	v4	v5	v6	v7	v8	o4	o5	o6	o7	o8	o_gelis	o_gidis	s_gelis	s_gidis
julian	1,000	-0,027	-0,097	-0,164	-0,130	0,022	-0,870	0,211	0,202	0,235	0,165	0,157	0,083	0,221	0,209	0,201	0,116	0,096	0,186	0,207	0,171	0,099	0,204	-0,065	-0,144
time	-0,027	1,000	-0,384	-0,423	-0,318	-0,013	0,029	-0,014	0,223	0,196	0,270	0,243	0,273	0,564	0,497	0,501	0,131	0,117	0,513	0,452	0,370	0,139	0,480	-0,167	-0,426
s6	-0,097	-0,384	1,000	0,698	0,589	0,044	0,066	-0,097	-0,109	-0,101	-0,140	-0,136	-0,143	-0,445	-0,354	-0,391	-0,074	-0,059	-0,576	-0,542	-0,534	-0,082	-0,595	0,114	0,893
s7	-0,164	-0,423	0,698	1,000	0,703	0,029	0,123	-0,123	-0,135	-0,129	-0,169	-0,123	-0,132	-0,500	-0,404	-0,428	-0,140	-0,121	-0,621	-0,644	-0,613	-0,152	-0,679	0,187	0,894
s8	-0,130	-0,318	0,589	0,703	1,000	0,021	0,093	-0,112	-0,133	-0,127	-0,157	-0,116	-0,126	-0,386	-0,317	-0,320	-0,114	-0,103	-0,504	-0,528	-0,553	-0,122	-0,572	0,148	0,848
event	0,022	-0,013	0,044	0,029	0,021	1,000	0,071	0,076	0,012	0,002	-0,037	-0,014	-0,026	-0,061	-0,027	-0,041	0,028	0,007	-0,056	-0,012	-0,034	0,033	-0,036	-0,014	0,037
havadurumu	-0,870	0,029	0,066	0,123	0,093	0,071	1,000	-0,152	-0,166	-0,197	-0,143	-0,147	-0,078	-0,194	-0,179	-0,187	-0,075	-0,059	-0,147	-0,155	-0,141	-0,057	-0,160	0,026	0,103
havakosulu	0,211	-0,014	-0,097	-0,123	-0,112	0,076	-0,152	1,000	0,041	0,044	-0,004	0,025	0,002	0,027	0,039	0,009	-0,004	-0,007	0,051	0,076	0,038	-0,004	0,061	-0,009	-0,124
v1	0,202	0,223	-0,109	-0,135	-0,133	0,012	-0,166	0,041	1,000	0,828	0,628	0,579	0,618	0,444	0,548	0,495	0,127	0,167	0,372	0,433	0,348	0,094	0,417	-0,090	-0,140
v2	0,235	0,196	-0,101	-0,129	-0,127	0,002	-0,197	0,044	0,828	1,000	0,579	0,470	0,543	0,401	0,492	0,446	0,220	0,256	0,333	0,402	0,317	0,215	0,381	-0,214	-0,132
v3	0,165	0,270	-0,140	-0,169	-0,157	-0,037	-0,143	-0,004	0,628	0,579	1,000	0,556	0,625	0,393	0,489	0,441	0,261	0,316	0,335	0,410	0,318	0,202	0,385	-0,209	-0,174
v4	0,157	0,243	-0,136	-0,123	-0,116	-0,014	-0,147	0,025	0,579	0,470	0,556	1,000	0,863	0,437	0,540	0,477	-0,288	-0,251	0,377	0,384	0,325	-0,394	0,391	0,428	-0,143
v5	0,083	0,273	-0,143	-0,132	-0,126	-0,026	-0,078	0,002	0,618	0,543	0,625	0,863	1,000	0,426	0,536	0,474	-0,178	-0,094	0,366	0,390	0,324	-0,246	0,389	0,281	-0,152
v6	0,221	0,564	-0,445	-0,500	-0,386	-0,061	-0,194	0,027	0,444	0,401	0,393	0,437	0,426	1,000	0,854	0,845	0,095	0,097	0,891	0,733	0,612	0,083	0,800	-0,089	-0,503
v7	0,209	0,497	-0,354	-0,404	-0,317	-0,027	-0,179	0,039	0,548	0,492	0,489	0,540	0,536	0,854	1,000	0,831	0,108	0,114	0,745	0,830	0,595	0,083	0,785	-0,075	-0,405
v8	0,201	0,501	-0,391	-0,428	-0,320	-0,041	-0,187	0,009	0,495	0,446	0,441	0,477	0,474	0,845	0,831	1,000	0,102	0,108	0,766	0,729	0,731	0,086	0,801	-0,078	-0,432
o4	0,116	0,131	-0,074	-0,140	-0,114	0,028	-0,075	-0,004	0,127	0,220	0,261	-0,288	-0,178	0,095	0,108	0,102	1,000	0,745	0,072	0,186	0,089	0,901	0,130	-0,849	-0,120
o5	0,096	0,117	-0,059	-0,121	-0,103	0,007	-0,059	-0,007	0,167	0,256	0,316	-0,251	-0,094	0,097	0,114	0,108	0,745	1,000	0,071	0,168	0,091	0,849	0,123	-0,810	-0,103
o6	0,186	0,513	-0,576	-0,621	-0,504	-0,056	-0,147	0,051	0,372	0,333	0,335	0,377	0,366	0,891	0,745	0,766	0,072	0,071	1,000	0,794	0,731	0,064	0,903	-0,074	-0,644
o7	0,207	0,452	-0,542	-0,644	-0,528	-0,012	-0,155	0,076	0,433	0,402	0,410	0,384	0,390	0,733	0,830	0,729	0,186	0,168	0,794	1,000	0,786	0,170	0,937	-0,166	-0,644
o8	0,171	0,370	-0,534	-0,613	-0,553	-0,034	-0,141	0,038	0,348	0,317	0,318	0,325	0,324	0,612	0,595	0,731	0,089	0,091	0,731	0,786	1,000	0,083	0,909	-0,081	-0,639
o_gelis	0,099	0,139	-0,082	-0,152	-0,122	0,033	-0,057	-0,004	0,094	0,215	0,202	-0,394	-0,246	0,083	0,083	0,086	0,901	0,849	0,064	0,170	0,083	1,000	0,119	-0,946	-0,130
o_gidis	0,204	0,480	-0,595	-0,679	-0,572	-0,036	-0,160	0,061	0,417	0,381	0,385	0,391	0,389	0,800	0,785	0,801	0,130	0,123	0,903	0,937	0,909	0,119	1,000	-0,120	-0,695
s_gelis	-0,065	-0,167	0,114	0,187	0,148	-0,014	0,026	-0,009	-0,090	-0,214	-0,209	0,428	0,281	-0,089	-0,075	-0,078	-0,849	-0,810	-0,074	-0,166	-0,081	-0,946	-0,120	1,000	0,166
s_gidis	-0,144	-0,426	0,893	0,894	0,848	0,037	0,103	-0,124	-0,140	-0,132	-0,174	-0,143	-0,152	-0,503	-0,405	-0,432	-0,120	-0,103	-0,644	-0,644	-0,639	-0,130	-0,695	0,166	1,000

Korelasyon tablosu incelendikten sonra kısaca birkaç yorum yapmak gerekirse,

- Korelasyon tablosu bütün parametreleri içerdiği durumda oldukça büyük bir matris oluşturmakta ve gereksiz, birbiri ile ilişki içerisinde olan bağımsız değişkenler içermekte idi. Bu değişkenlerin bir önce ki bölümde neden ve nasıl çıkarıldığı açıklanmıştır. Bu sebeple yeni korelasyon tablosunda 's1, s2, s3, s4, s5, o1, o2, o3' değişkenleri bulunmamaktadır.

- Geliş ve gidiş yönlerinin hem şerit bazlı hem de ortalama değerler için hız ve işgaliye değerlerinin negatif korelasyon içerisinde olduğu açık bir şekilde gözlemlenmektedir.

- Hava durumu ile yolun hızının negatif korelasyonda olması beklenirken Çizelge 3-6'de görüldüğü gibi şeritlerin bireysel hızları ve ortalama hızları ile çok düşük seviyeli de pozitif korelasyon görülürken şeritlerdeki araç sayısı ve işgaliye ile çok düşük seviyeli negatif korelasyon görülmektedir.

- Zaman yani saat değerleri ile en çok korelasyon içeresinde olan verinin altı numaralı şeridin araç sayısı ve işgaliye miktarı olduğu görülmektedir. Altıncı şerit, sensör verilerini aldığımız birçok yolun en sağ veya en sol şerididir, yani en son şerididir. Trafiğin yoğun olduğu işe gidiş ve işten çıkış saatleri gibi saatlerde bu şeritten başka yollara dönülen sapaklarda oluşan tıkanma bu korelasyona sebep verebilir.

- Çizelge 3-7'de görülen tarih(Julian) ile hava durumunun yüksek dereceli negatif korelasyonunun sebebi ise veri setinin yaz aylarının başından kış aylarına doğru seyretmesidir. Yani tarih ilerledikçe hava sıcaklıkları düşmektedir.

3.7.2 Betimleyici istatistikler

Çizelge 3-8'de veri setini betimleyen veriler gösterilmiştir. Bunlar ortalama, standart sapma, veri setinin her bir değişkenin en küçük, en büyük değerleri ve diğerleridir.

Çizelge 3-8: Betimleyici istatistikler

	s6	s7	s8	event	havadurumu	v1	v2	v3	v4	v5	v6
mean	105,486	95,450	91,760	0,062	20,468	14,761	15,519	10,517	20,453	17,393	18,651
std	11,469	8,291	8,572	0,242	4,940	7,468	9,616	6,127	10,257	10,730	12,212
min	1,000	6,000	1,000	0,000	9,000	0,000	0,000	0,000	0,000	0,000	0,000
25%	99,000	91,000	88,000	0,000	16,000	9,000	8,000	6,000	12,000	9,000	8,000
50%	106,000	94,000	91,000	0,000	22,000	14,000	14,000	10,000	20,000	16,000	19,000
75%	112,000	99,000	96,000	0,000	24,000	20,000	21,000	14,000	28,000	25,000	28,000
max	191,000	180,000	178,000	1,000	31,000	62,000	69,000	65,000	65,000	72,000	64,000
	v7	v8	o4	o5	o6	o7	o8	o_gelis	o_gidis	s_gelis	s_gidis
mean	26,965	16,712	11,453	8,399	3,520	5,973	3,982	10,309	4,159	75,489	97,235
std	11,985	8,473	16,312	12,797	2,996	3,673	3,336	16,645	3,081	30,945	8,318
min	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	3,000	11,000
25%	18,000	10,000	3,000	2,000	1,000	3,000	2,000	2,000	2,000	81,000	93,000
50%	28,000	17,000	5,000	4,000	3,000	6,000	4,000	3,000	4,000	90,000	97,000
75%	36,000	23,000	8,000	7,000	5,000	8,000	6,000	6,000	6,000	93,000	102,000
max	60,000	65,000	99,000	99,000	58,000	70,000	99,000	81,000	50,000	121,000	139,000

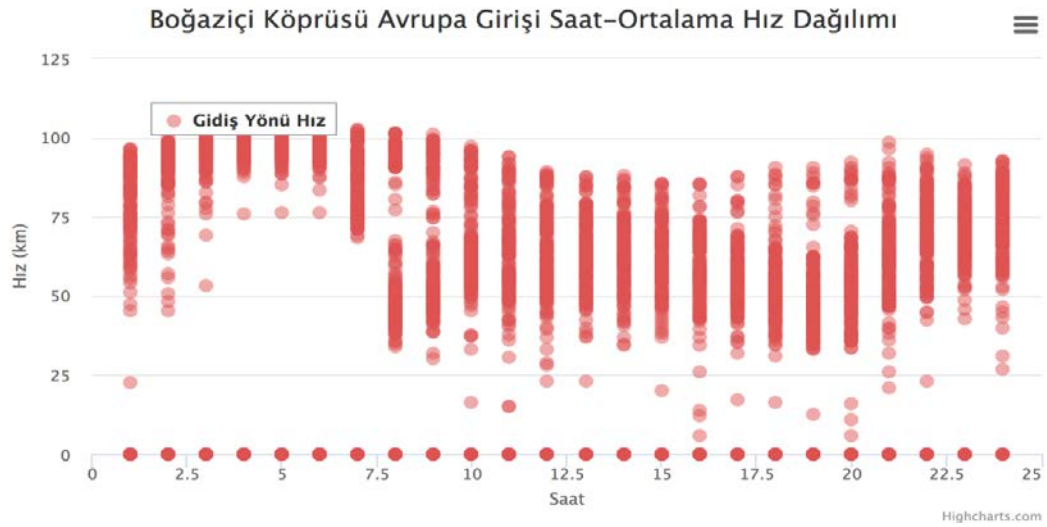
Betimleyici istatistikler veri setinin temel istatistiksel karakteristiklerini ortaya döken verilerdir. Bu verilerden veri setinin her parametresinin dağılımını ayrı ayrı inceleyebiliriz.

- Çizelge 3-8'den gözlemlenebilecek durumlardan birisi gidiş yönündeki ortalama işgaliye miktarının standart sapması oldukça düşükken, geliş yönü için bu standart sapma oldukça yüksektir. Bu durum aynı şekilde geliş ve dönüş yönünün hız ortalama hızları içinde geçerlidir.

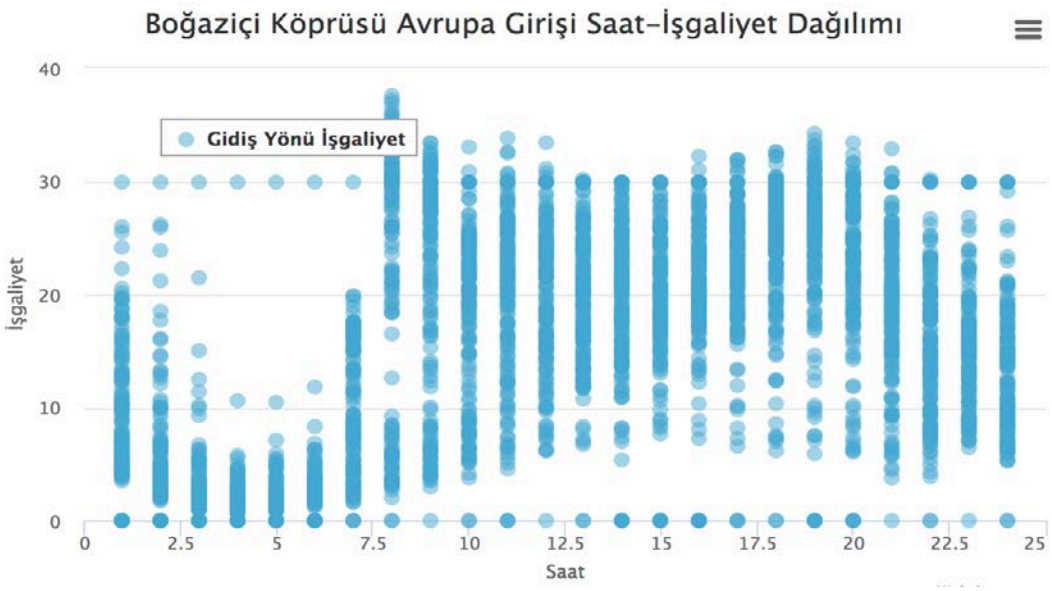
- Geliş ve gidiş yönlerinin ortalama hızları yakın seyrederken, aynı şekilde geliş ve gidiş yönlerinin ortalama işgaliye miktarları birbirlerinden farklı ortalama rakamlarda seyretmektedirler.

- Ayrıca görülmektedir ki veri setinin geldiği aralıkta çok soğuk havalar izlenmemiş en düşük 8 derece yaşanırken en yüksek 31 derece görülmüştür. Bu ve bunlar gibi veri setini betimleyici değerler Çizelge 3-8'den incelenebilir.

Veri setinin yapısında örnekler göstermek amacı ile 60 numaralı sensörün, yani Boğaziçi köprüsü Avrupa yakası giriş tarafının grafikleri Şekil 3-3 ve Şekil 3-4 gösterilmiştir. Gösterilen grafikler JavaScript programlama dili kullanılarak highcharts [25] kütüphanesinden çizilmiştir.



Şekil 3-3: 60 Numaralı sensör hız dağılımı



Şekil 3-4: 60 Numaralı sensör işgaliye dağılımı

Şekil 3-3 ve Şekil 3-4'den hız ve işgaliyenin ters yönlü ilişkisi gözlemlenmektedir. Doğal olarak işgaliyenin arttığı saatlerde hız ortalamaları düşerken, işgaliyenin az olduğu saatlerde hızda oldukça fazla bir yükselme vardır. Aynı zamanda Şekil 3-3'de görülmektedir ki sabah işe gidiş ve akşam işten çıkışların yoğun olduğu saatlerde hız düşük bir eğri izler iken işgaliye yüksek bir eğri izlemektedir.

4. METOD

4.1 Özbağlanımsal Tümleşik Hareketli Ortalama

Durağan zaman serilerini modellemenin yaygın yollarından birisi olan Özbağlanımsal Tümleşik Hareketli Ortalama (Autoregressive Integrated Moving Average (ARIMA)), George Box ve Gwilym Jenkins tarafından geliştirilmiştir. Bu yaklaşıma Box-Jenkins (BJ) yöntemi de denilmektedir. Box-Jenkins temelde zaman serilerini yalnızca kendi geçmiş değerleri ile açıklar [26]. Çalışmada ARIMA kullanılabileceğimize emin olmak adına veri setinin durağanlığı test edilmelidir. Durağanlık Dickey-Fuller testi ile belirlenebilir.

ARIMA (p, d, q) modelinde;

- AR(p), bir değişkenin değerinin önceki döneme ait değerleri ile ilişkiyi gösteren, p logları olan otoregresif bir modellerdir.
- MA(q), bir değişken ile önceki dönemlerden kalan kalıntılar arasındaki ilişki olasılığını hesaplayan, q loglu bir hareketli ortalama modelidir.
- I(d), ARMA(p,q) olarak seriyi modelleyebilmek için serinin d kez farkı alınarak durağanlaştırılmasıdır [27].

Box-Jenkins yöntemi ile tahmin 4 aşamada gerçekleşir;

- Model Belirleme: Uygun Box-Jenkins modeli belirlenir.
- Parametre Tahmini: Belirlenen modele uygun parametrelerin tahmin edildiği aşamadır.
- Uygunluk testi: İstatistiksel yöntemler ile test edilen model veri setine uygun ise son aşamaya geçilir değil ise başka bir model belirlemek için ilk aşamaya dönülür.
- Tahmin: En uygun model için tahminde bulunur [28].

ARIMA (p, d, q) modeli aşağıdaki gibi modellenir;

$$Z_t = \Phi_t Z_{t-1} + \Phi_2 Z_{t-2} + \ldots + \Phi_p Z_{t-p} + \delta + a_t - \Theta_2 a_{t-2} - \ldots - \Theta_q a_{t-q}$$

Bu modelde Zt-p'ler d dereceden farkı alınmış gözlem değerlerini, Φp'ler d dereceden farkı alınmış gözlem değerleri için katsayıları, δ sabit değeri, Θq'ler ise hata terimleri ile ilgili katsayıları göstermektedir [28].

Daha öncede bahsedildiği gibi ARIMA durağan serileri modellemek için kullanılan bir yöntemdir. Verinin durağanlığını ölçmek için literatürde önerilen birden fazla yöntem vardır. Bu çalışmada verinin durağan olup olmadığını anlamak için Dickey-Fuller testi yapılmıştır. Dickey-Fuller testini kısaca açıklamak gerekirse. AR bileşeni içeren seriler birim kök içerebilmektedir ve veri setinin birim kök içermesi durağanlığını bozmaktadır. Dickey-Fuller testide serinin birim kök içerip içermediğini sınamaya yaramaktadır. Dickey-Fuller'a göre;

y_t 'nin gözlenen değeri temsil ettiği birinci dereceden otoregresif bir modelde,

$y_t = pt_{t-1} + u_t$ iken, | p | ≥ 1 olduğu gösterilebiliyorsa birim kök vardır ve sıfır hipotezi ret edilemez denilebilir. Dickey fuller testi için kurulabilecek hipotezler aşağıdaki gibidir.

H_0: Seri durağan değildir.

H_1: Seri durağandır.

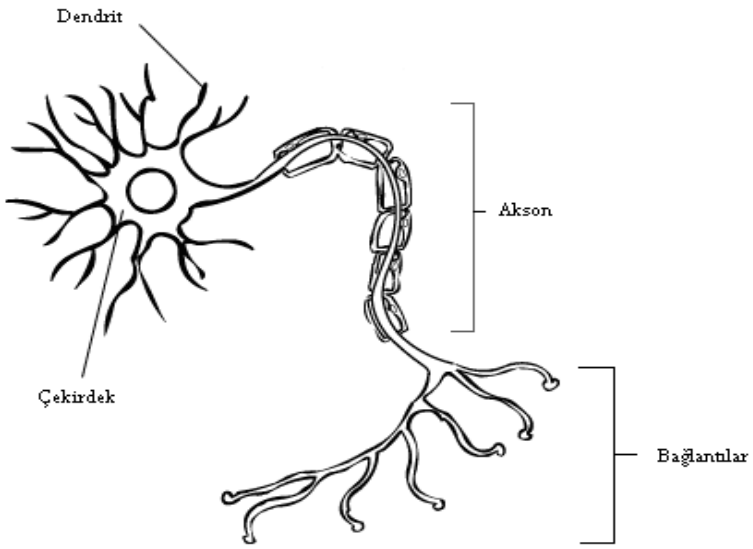
4.2 Derin Öğrenme: Çok Katmanlı Algılayıcılar

Bu çalışmada derin öğrenme modeli olarak çok katmanlı algılayıcılar kullanılmıştır. Bu bölümde çok katmanlı algılayıcıların yapay sinir ağları ile ilişkilerine, türlerine ve en iyileştirme algoritmalarına değinilecektir.

4.2.1 Yapay sinir ağları ve çok katmanlı algılayıcılar

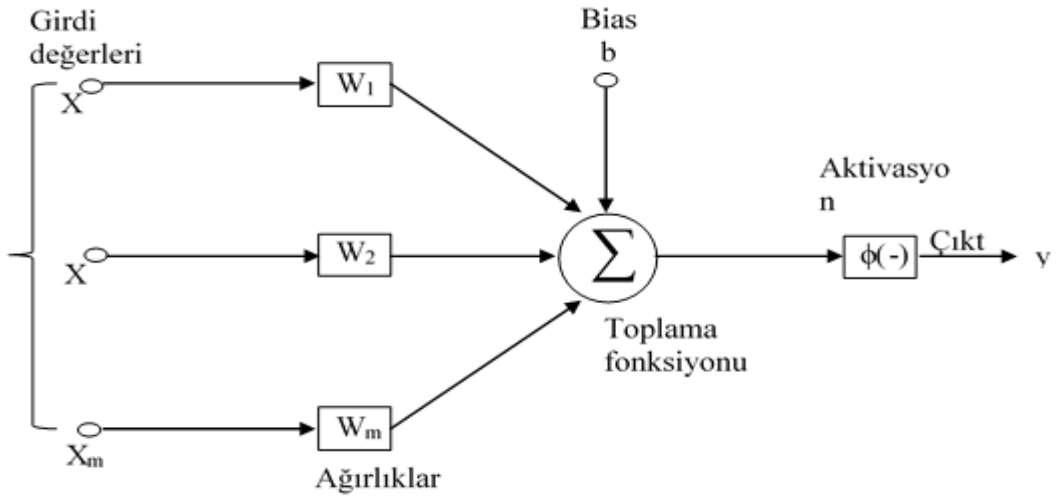
Bir bilgi işleme yaklaşımı olan Yapay Sinir Ağları, insan beynini ve sinir sistemini taklit etme yaklaşımı üzerine kurulmuş olup deneyimsel bilgileri biriktirmeye yönelik doğal bir eğilimleri vardır. Biyolojik sinir ağlarının sinir hücresi olduğu gibi yapay sinir ağlarının da yapay sinir hücreleri vardır. Şekil 4-1'de insan beyninin sinir hücresi yapısı gösterilmektedir. Biyolojik sinir siteminin temel yapı taşı olan nöronlar 3 tip bileşenden oluşur; soma, akson ve dentritlerden. Dentrit, dentritler arası sinaptik boşluklarla diğer nöronlardan iletilen elektrik tepkilerini giriş sinyalleri olarak alır.

Akson aldığı sinyali çıkış sinyali olarak diğer nöronlara iletir. Sinaps ise akson ve dentritin birleşim yerinde bulunarak aktivasyon fonksiyonu görevini görür [29].



Şekil 4-1: İnsan beyninin sinir hücresi yapısı

YSA'larında bulunan her bir nöron (sinir hücresi) aynı insan beyninde olduğu şekilde çalışır. YSA'lar nöronlardan, girdi/girdilerden, toplama fonksiyonundan, aktivasyon fonksiyonundan ve çıktı/çıktılardan oluşur. Geleneksel YSA Şekil 4-2'de olduğu gibidir.



Şekil 4-2 Yapay sinir ağ

Girdiler (X_i) n elemanlı sütun vektörü oluştururken aşağıda formülasyon (4.1)'de gösterildiği gibi, ağırlıklar (W_i) n elemanlı satır vektörü oluşturur.

$$X = \begin{bmatrix} X1 \\ \vdots \\ Xn \end{bmatrix}, \; W = [W1 \quad \cdots \quad Wn] \tag{4.1}$$

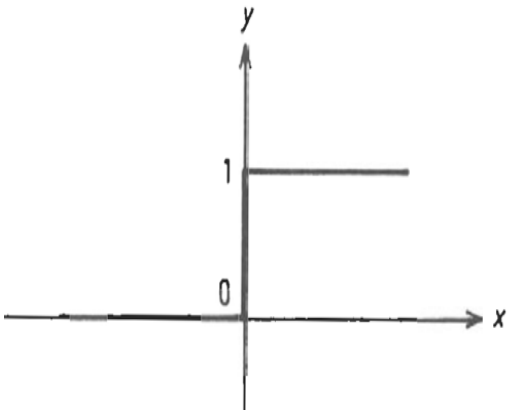
$$y = f(\sum wx + \theta)$$

$$f(.)$$

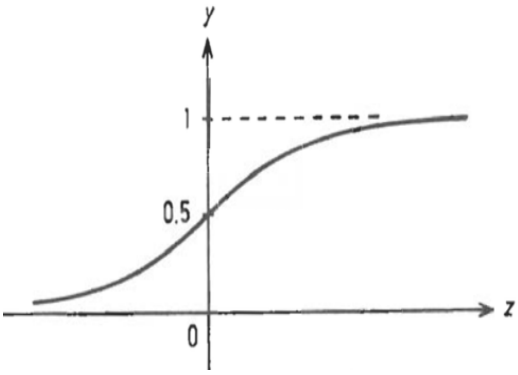
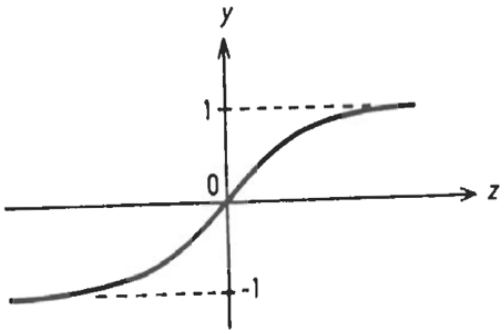
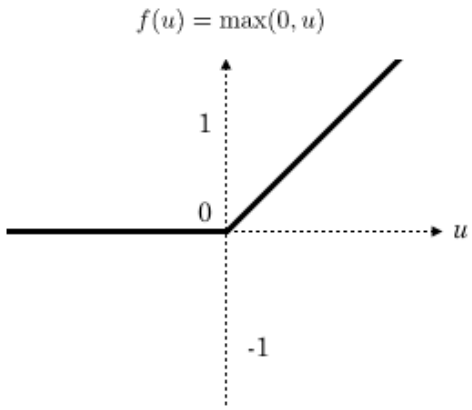
Denklemde y değeri YSA'nın çıktı değeri iken, $f(.)$ seçilen aktivasyon fonksiyonudur. Aktivasyon fonksiyonu çıktı değerini istenilen aralığa göre yeniden yapılandırır [30]. Aktivasyon fonksiyonu seçimi problemin ve veri setinin yapısına göre değişmektedir.

Yaygın olarak kullanılan aktivasyon fonksiyonları, formülasyonları ve özet bilgiler Çizelge 4-1' de gösterilmektedir.

Çizelge 4-1: Aktivasyon fonksiyonları [31]

Adım (Step) Fonksiyon 	$y = F(x)$ $= \begin{cases} 1, & x > 0 \\ -1, & x \leq 0 \end{cases}$	Sadece 1 ve -1 şeklinde iki çıktı veren bu fonksiyon değerlerin sıfırdan büyük olup olmamasına göre buna karar verir.

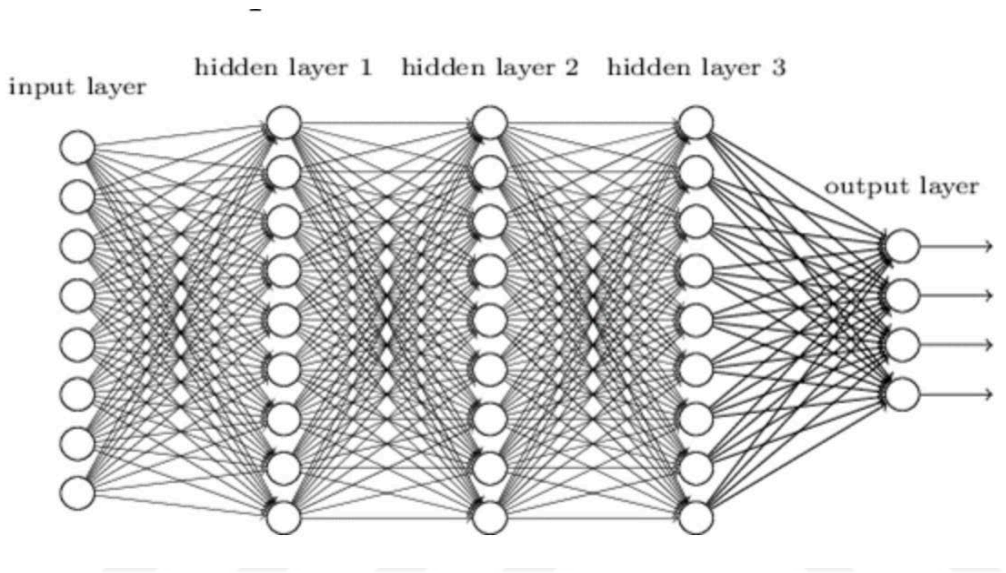
Çizelge 4-1: (Devam) Aktivasyon fonksiyonları [31]

Sigmoid Fonksiyon 	$y = F(x) = \frac{1}{1 + e^{-x}}$	Doğrusal olmayan sigmoid fonksiyonu sürekli ve türevi alınabilen bir fonksiyondur. Bu fonksiyon ile girdi değerlerinin tamamından 0 ile 1 arası değer üretilir.
Hiperbolik Tanjant Fonksiyon 	$y = F(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$	Sigmoid fonksiyonuna benzer şekilde çalışır. Girdi değerlerinden bu defa -1 ile +1 arası değer üretilir.
Rectified Linear Fonksiyon 	$y = F(x) = \max(0, x)$	Doğru hatlı birimler, bilgi detektörlerinin birden fazla katmanından geçerken göreli yoğunluklarla ilgili bilgileri korur.

Hangi aktivasyon fonksiyonunun seçileceği veri tipine, çalışmayı yapan kullanıcının yapacağı denemelere, kullanacağı algoritmaya bağlıdır. Örneğin, birçok derin öğrenme modeli doğrusal değildir ve türevi alınabilen fonksiyonlar kullanmayı şart

koşar [30]. Bu gibi durumlarda veri ve model incelenmeli, hala birden fazla seçenek kalıyor ise kullanıcı deneyerek karar vermelidir.

Bir makine öğrenmesi tekniği olan derin öğrenme (aynı zamanda hiyerarşik öğrenme), alt seviyedeki faktörler ile üst seviyede bulunan faktörlerin hiyerarşik özelliklerini hesaplar. Normalde YSA bir girdi katmanı, bir gizli katman bir de ara katman olabilirken derin bir ağda giriş ve çıkış arasında çok sayıda katman vardır [32]. Derin öğrenme modeline bir örnek Şekil 4-3'te gösterilmektedir.



Şekil 4-3: Çok katmanlı algılayıcı [33]

YSA'larının birçok özelliği derin öğrenme modelleri içinde geçerlidir. Bunlar:

- Doğrusal olmama: Katmanlar arası zaman bağımlılığı olmayan yani bütün sistemin eş zamanlı çalışmasına izin veren bu sistemlerde hücreler doğrusal olmadığı için bu hücrelerden oluşan YSA' da doğrusal değildir.

- Örnekten Öğrenme: Biyolojik sistemleri örnek alarak tasarlanan YSA'ları insan beynindeki hücrelerin arasında olan ve insan tecrübe ettikçe gelişen bağlantılar gibi bağlar oluşturur. Bu bağları öğrenme algoritmalarını kullanarak oluşturur.

- Paralellik: Alışılmış seri işlem gerçekleştiren bilgi işlem yöntemlerinin aksine YSA'ları paralel dağıtılmış bir yapıdadır ve bu sayede yavaş bir birim seri sistemlerde olduğu gibi tüm sisteme etkisi çok yüksek değildir.

- Genelleme yeteneği: Bir diğer özellik olan genelleme yeteneği sayesinde eğitilen bir YSA, verilen yeni bir durum için doğru tahminler yapar.

- Uyarlanabilirlik: Farklı koşullar altında eğittiğimiz bir YSA'nı değişen koşullarımıza göre tekrar eğitebiliriz.
- Hata toleransı: Geleneksel seri bağlanan ağlarda oluşan bir hata ya da etkisiz hale geçen bir hücre bütün ağın doğru bilgi üretmesini önemli derece etkilenen, bu tarz paralel bağlanan ağlarda bu durumdan bütün ağ etkilenmez. Bu sebeple paralel ağların hatayı tolere etme olasılığı daha yüksektir [30].

Çok katmanlı algılayıcılar (ÇKA) öncelikle eğitim ve test olarak iki parçaya ayrılan veri seti üzerinde eğitim ve tahmin olmak üzere iki ayrı fazda çalışır. Eğitim fazı her döngü (epoch) için tekrarlanan bir süreçtir. İlk döngü rastgele belirlenen ağırlıklar ile başlar ve her döngü ağırlıkları ve hatayı iyileştirerek yenilemeli şekilde ilerler.

4.2.2 Yapay sinir ağ türleri

Yapay sinir ağları mimarilerine, katman sayılarına ve öğrenme yöntemlerine göre gruplandırılabilir. Bu gruplandırmalara [29] numaralı kaynak referans gösterilebilir ve bu gruplandırmalara bu bölümde yer verilmiştir.

4.2.2.1 Mimarilerine göre

- İleri beslemeli yapay sinir ağları
- Geri beslemeli yapay sinir ağları

4.2.2.2 Katman sayılarına göre

- Tek Katmanlı
  - Perceptron
- Çok Katmanlı
  - Hopfield Ağı
  - Kohonen Özellik Haritası

4.2.2.3 Öğrenme yöntemine göre

- Danışmalı Öğrenme
- Danışmansız Öğrenme
- Destekleyici Öğrenme

4.2.3 Yapay sinir ağlarında öğrenme

- Geri Yayılım öğrenme algoritması
- Hebb öğrenme kuralı
- Hopfield öğrenme kuralı
- Kohonen öğrenme kuralı
- Levenberg-Marquardt öğrenme algoritması

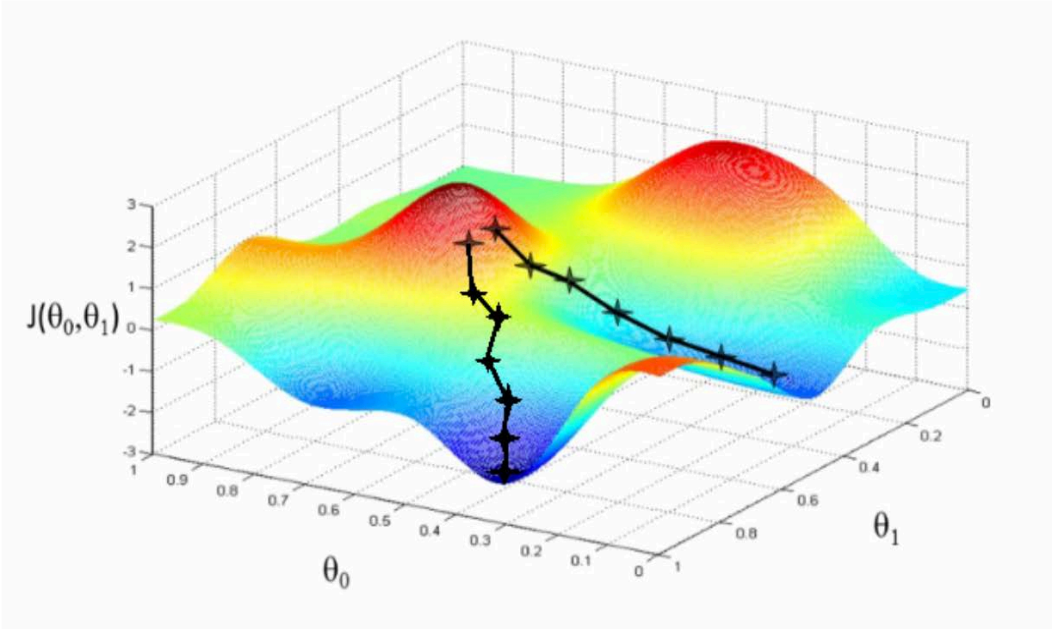
Öğrenme algoritmaları arasında en yaygın kullanılan Geri Yayılım öğrenme algoritmasıdır. Çalışmamızda da Geri Yayılım algoritması ile modelin eğitimi yapılacağı için, bu algoritmanın çalışma prensipleri ayrıntılı olarak anlatılacaktır.

4.2.3.1 Geri yayılım algoritması

Dereceli alçalma (Gradient Descent) denilen yöntemin bir formu olan Geri Yayılım (Backpropagation) algoritması çıktı için hesaplanan hatayı sinir ağ yapısına doğru geri gönderir. Hedef çıktı değeri belirli olduğu zaman Geri Yayılım algoritması kullanılabilir. Bu algoritmayı belirlemek için bir hata fonksiyonu belirlemeli ve "E" fonksiyon değerini belirtmelidir. Burada amaç hedef çıktı ile tahmin edilen çıktı arasındaki farkı temsil eden bu "E" değerini minimize etmektir. Çıktıdaki hataları tespit ettikten sonra hataların türevlerine bağlı olarak ağırlıkları hangi yönde güncelleyeceğimize karar veririz. Sebebini açıklamak gerekir ise; ileri besleme fazının sonunda elimizde üç şey bulunmaktadır. Bunlar girdi değerleri, aktivasyon fonksiyonu f(.) ve ağın ağırlıklarıdır.

Modelin girdi değerlerini ve aktivasyon fonksiyonunu değiştiremeyiz çünkü algoritma bu girdi ve aktivasyon fonksiyonu koşulları için öğrenmeyi gerçekleştirmektedir. Algoritmanın performansını artırmak için tek iyileştirebileceğimiz şey ağırlıklardır. Yani Geri Yayılım algoritması hatayı en küçüklemek için ağırlıkları eğitir [34].

Şekil 4-4'de gösterilen dereceli alçalma modelinde olduğu gibi Geri Yayılım algoritması hatanın yerel en küçük noktasına ulaşması üzerine tasarlanmıştır. Şekil 4-4'de kullanılan eksen isimler, $J(\theta)$ seçilen hata fonksiyonunu ve θ_0 ve θ_1 ise onun parametrelerini temsil etmektedir [35].



Şekil 4-4: Dereceli alçalma

Geri yayılım algoritmasının matematiksel açıklaması oldukça karışıktır. Algoritmanın ana hatlarının anlaşılması adına algoritmanın sözde kodunu incelemekte fayda vardır. Geri yayılım algoritmasını içeren yapay sinir ağı sözde kodu aşağıdaki algoritmada gösterildiği gibidir.

- Tüm ağ giriş ve çıkış değerlerini ayarla
  - Tüm ağırlık değerlerini -1 ile +1 arası küçük sayılara rastgele ata
  - Şunlar için tekrar et;
    - ✓ Eğitim setindeki her örüntü için,
    - ✓ Örüntüyü ağa sun

#Giriş ağı üzerinden ileri ileti:

- Ağdaki her katman için:
  - Katmandaki her düğüm için:
    - ✓ Düğümün girdilerinin ağırlıklarının toplamını hesapla
    - ✓ Eşik değerini toplama ekle
    - ✓ Her düğüm için aktivasyonu hesapla

#Hataları ağ üzerinden geriye doğru yayma:

- Çıktı katmanındaki her düğüm için:
  - Hata sinyalini hesapla

- Bütün gizli katmanlar için:
  - Katmandaki her düğüm için:
    - ✓ Düğümün hata sinyalini hesapla
    - ✓ Ağdaki her düğümün ağırlığını güncelle

#Genel hata hesaplama:

- Seçilen hata fonksiyonu ile ağın genel hatasını hesapla

#Döngü sayısı belirlenenden az ve hata fonksiyonu değeri belirtilenden fazla olduğu sürece yap.

Geri Yayılım Algoritmasının işleyişini matematiksel olarak incelememiz gerekirse, Algoritmada türev fonksiyonunun zincir kuralı kullanılır, E hata fonksiyonun değerini gösterirken, $\boldsymbol{w_{jk}}$ ise j'den k katmanına olan ağırlığı temsil eder.

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial h_k}\frac{\partial h_k}{\partial w_{jk}}$$

Bu formülasyonda, $h_k = \sum_l w_{lk}\, a_l$'dir. Yani h_k çıkış katmanına giren, gizli katmanların aktivasyon fonksiyonlarından çıkan değerlerin kendi ağırlıkları ile çarpılmış hallerinin toplamlarıdır. İlk olarak ikinci terim için işlem yaparsak (l = j olduğu durumlar dışında $\frac{\partial w_{lk}}{\partial w_{jk}} = 0$'dır.).

$$\frac{\partial h_k}{\partial w_{jk}} = \frac{\partial \sum_l w_{lk}\, a_l}{\partial w_{jk}}$$

$$= \sum_l \frac{\partial \sum_l w_{lk}\, a_l}{\partial w_{jk}}$$

$$= a_j$$

İkinci terim için işlemimizi bitirdikten sonra ilk terim için işlem yapmaya başlarsak,

$$\delta_0 = \frac{\partial E}{\partial h_k}$$

Çıktı için hatayı doğrudan hesaplayamayacağımız için bu noktada da zincir kuralını kullanabiliriz. Doğrudan hesaplanamamasının sebebi ise sinir hücresinin sadece çıktısını biliyor ve girdisi hakkında pek bir şey bilmiyor olmamızdır.

$$\delta_0 = \frac{\partial E}{\partial h_k} = \frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial h_k}$$

Çıktı katmanının çıktısını bir önceki gizli katmanı kullanarak hesaplar isek,

$$y_k = f\left(h_k^{output}\right) = f\left(\sum_j w_{jk} a_j^{hidden}\right)$$

f (.) aktivasyon fonksiyonunu temsil etmektedir.

$$\delta_0 = \frac{\partial E}{\partial_f} \frac{\partial_f (h_k^{output})}{\partial h_k^{output}}$$

$$= \frac{\partial E}{\partial_f \left(h_k^{output}\right)} f'\left(h_k^{output}\right)$$

$$= \frac{\partial}{\partial_f \left(h_k^{output}\right)} \left[\frac{1}{2} \sum_k (f(h^{output}) - t_k)^2\right] f'(h_k^{output})$$

$$= (f(h^{output}) - t_k) f'\left(h_k^{output}\right)$$

$$= (y_k - t_k) f'(h_k^{output})$$

İkinci katmanın ağırlıklarını güncelleme kuralını şu şekilde getirebiliriz,

$$\frac{\partial E}{\partial w_{jk}} = \delta_0 a_j$$

$$= (y_k - t_k) y_k (1 - y_k) a_j$$

Bu noktaya kadar kadarki aşamaları bütün gizli katmanlar için yapabiliriz. Gizli katmanlardan girdi katmanına doğru giderken gene zincir kuralını kullanmalıyız.

$$\partial_h = \sum_k \frac{\partial E}{\partial h_k^{output}} \frac{\partial h_k^{output}}{\partial h_j^{hidden}}$$

$$= \sum_k \partial_0 \frac{\partial h_k^{output}}{\partial h_j^{hidden}}$$

Hatırlanması gereken en önemli şeylerden birisi, veriler çıktı katmanına gelmeden önce girdi ve gizli katmanların tamamının aktivasyon fonksiyonundan geçiyor ve o katmanların ağırlıkları ile çarpılıyor.

$$h_k^{output} = f\left(\sum_l w_{lk} h_l^{hidden}\right)$$

$$\frac{\partial h_k^{output}}{\partial h_j^{hidden}} = \frac{\partial_{f\left(\sum_l w_{lk} h_l^{hidden}\right)}}{\partial h_j^{hidden}}$$

l = j olduğu durumlar dışında $\frac{\partial \mathrm{h}_l}{\partial h_j} = 0$'dır.

$$\frac{\partial h_k^{output}}{\partial h_j^{hidden}} = w_{jk} f'(a_j)$$

$$= w_{jk} a_j (1 - a_j)$$

$$\partial_h = a_j (1 - a_j) \sum_k \partial_0 w_{jk}$$

Yapılan hesaplamalar doğrultusunda aşağıdaki güncelleme kuralı ortaya çıkmaktadır.

$$\frac{\partial E}{\partial v_{ij}} = a_j (1 - a_j) \left(\sum_k \partial_0 w_{jk}\right) x_i$$

Burada not edilmelidir ki bu hesaplama modelde bir girdi katmanı, bir gizli katman ve bir çıktı katmanı olması durumu için hesaplanmıştır. Birden fazla gizli katmanı olduğu durumlar için türev alma işleme modele göre tekrar yapılabilir. Fakat unutulmamalıdır ki gizli katman sayısı arttıkça hesaplamalar git gide karışık hale gelecektir ve daha dikkatli hesaplanmalıdır.

4.3 Modellerin Tahmin Doğrulunun Test Edilmesi

İstatistiksel öğrenme yani modelleme metotlarının veri seti üzerinde ki performansını değerlendirmek için modelin ürettiği sonuçlar ile gerçek sonuçları karşılaştıran yöntemlerdir. Ortalama hata (Mean Error- E), Ortalama Karesel Hata (Mean Squared Error - MSE), Kök Ortalama Karekter Hatası (Root Mean Square Error - RMSE), Ortalama Mutlak Hata (Mean Absolute Error - MAE) performansı değerlendirmek için kullanılan metriklerdendir. ME, MSE, MAE ve RMSE formülasyonları (4.2)'de gösterildiği gibidir.

$$ME = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{4.2}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)}$$

ME, MSE, MAE ve RMSE formülasyonlarının tamamında kullanılan parametreleri açıklamak gerekirse;

n: Toplam veri sayısı

$\hat{y}$: Bilinen olması gereken çıktı değeri

y : Tahmin edilen çıktı değeri

5. SONUÇLAR

Önerilen modelleri eğitmek ve bu modeller ile yapılacak tahminlerin doğruluğunu test etmek amacı ile veri seti %80 eğitim seti %20 test seti olmak üzere ikiye ayrılmıştır. Bunun sebebi eğitim seti ile modeli geliştirdikten sonra doğrudan gelecek tahmini yapmak yanlıştır. Eğitim setine göre oluşturulan modelden elimizde hiç test seti yokmuşçasına gelecek tahmini yapıp daha sonra modelin çıktıları ile test seti daha önceden belirlenen hata fonksiyonuna göre karşılaştırılır. Anlatılan metotlardan ARIMA ve Çok Katmanlı Algılayıcılar kullanılarak, seçilen pilot bölge için sensör bazlı ortalama hız tahmin edilmiştir. Elde edilen sonuçlardan da görülmüştür ki Çok Katmanlı Algılayıcılar çok daha düşük hata payları ile tahmin yapmışlardır.

5.1 Veri Setinin ARIMA ile Modellenmesi

ARIMA modeline karar vermeden önce metot kısmında ayrıntılı olarak anlatılan Dickey-Fuller durağanlık testi yapılmıştır. Dickey- Fuller test sonuçları ve EK 3'te verilen Dickey-Fuller tablosu ile karşılaştırma Çizelge 5-1'de gösterilmiştir.

Çizelge 5-1: Dickey-Fuller test sonuçları

Dickey-Fuller Test Değeri	P-Value	Dickey-Fuller Kritik Değeri	Tablo
-135.19	0.99	2	

Dickey-Fuller durağanlık testi için kurulan hipotezler şu şekildedir,

H_0: Seri durağan değildir.

H_1: Seri durağandır.

Dickey-Fuller test istatistiği değeri, kritik değerinden daha küçük ise sıfır hipotezi ret edilir ve veri durağandır.

- 135.19 <2.00 olması sebebi ile sıfır hipotezi ret edilir. Veri seti durağandır diyebiliriz.

Durağan olduğuna karar verilen veri seti için ARIMA modeli kullanılmıştır. ARIMA modelinin farklı (p, d, q) parametreleri için farklı modeller ile eğitim seti modellenmiş ve tahminler yapılmıştır. Kurulan ARIMA modelleri ve tahmin hatası değerleri Çizelge 5-2'de gösterilmiştir.

Çizelge 5-2: ARIMA modelleri ve tahmin hataları

(p, d, q)	**ME**	**RMSE**	**MAE**
(1, 0, 0)	-0.000512902	23.14302	14.42051
(2, 0, 0)	-0.000803899	22.26206	13.63462
(0, 0, 1)	-0.000163463	26.23663	19.55465
(1, 0, 1)	-0.001355201	21.90897	13.4199
(5, 1, 1)	-0.006834933	21.81542	13.48625

Çizelge 5-2'de verilen ARIMA modellerini inceledikten sonra bu modeller arasında en iyi sonucu veren modelin ARIMA (5, 1, 1) modeli olduğunu söyleyebiliriz. Bu modelin bile RMSE değeri 21.81542'dir. Bu hata değeri derin öğrenme modelleri ile karşılaştırıldığı zaman oldukça yüksektir.

5.2 Veri Setinin Çok Katmanlı Algılayıcılar ile Modellenmesi

Çalışmada Çok Katmanlı Algılayıcılar'ın tasarımında Python programlama dili ve keras [36] kütüphanesi kullanılmıştır. Çalışma için tasarlanan çok katmanlı algılayıcı metot kısmında ayrıntılı olarak bahsedilen geri yayılım algoritması ile eğitilmiştir. Geri yayılım algoritmasında tahmin fazında, girdi ağından çıktı ağına doğru akan bilgiler yardımı ile girdi vektörü kullanılarak çıktı değeri tahmin edilir. Bu çıktı değerini elimizde bulunan hedef çıktı değeri ile karşılaştırdığımızda ise tahmin modelinin kalitesine karar verebiliriz [37].

Tahmin modelini eğitmek için veri setinin daha sonra tahmin edilebilecek kısmı bağımsız değişken olarak, tahmin edilmek istenen ortalama hız değişkeni ise bağımlı değişken olarak kullanılmıştır. Modeli biraz daha açıklar isek modeli eğitmek için kullanılan girdi vektörü;

X = [Rtmsno, wend, rushHours, event, havaDurumu, havaKosulu] şeklindedir.

Burada daha önce bir kısmı açıklanan değişkenleri kısaca tekrar açıklamak gerekirse,

- Rtmsno: Verinin geldiği rtms sensörünün numarası
- wend: Veride bulunan tarih değerinden çıkarılan ve verinin geldiği günün hafta içi mi yoksa hafta sonu mu olduğunu gösteren değerdir. Bu değerin 0 olması durumunda hafta içi, 1 olması durumunda hafta sonu olduğu söylenebilir.
- rushHours: Verinin geldiği saat 'Rush Hour' olarak tanımlanacak saat grubunun içerisinde ise 1, değil ise 0 değerini alan parametredir. 'Rush Hour' olarak tanımlanan saatler işe gidiş işten çıkış ya da öğlen arası gibi trafiğin sıkışma eğiliminde olduğu saatler olarak tanımlanabilir.
- event: Verinin bakıldığı günde özel bir gün yani resmi bir tatil olup olmadığını gösteren veride bulunan tarih değişkeninden çıkarımı yapılan 0 ve 1 değerlerini alan değişkendir.
- havaDurumu: Hava durumu bakıldığı günün sıcaklığını gösteren değişkendir.
- havaKosulu: Verinin bakıldığı gün bir hava koşullu olup olmadığını gösteren değişkendir.1,2,3,4 şeklinde değerler alır.

Verilen X vektörü ile eğitilen model Y değişkeni olan yolun ortalama hızını tahmin etmeye çalışır. X vektörü oluşturulurken, daha sonra Y değişkeni için gelecek tahmini yapabileceğimiz ve bağımlı değişkeni etkileyen birbiri içerisinde körele olmayan değişkenler kullanılarak oluşturulmaya çalışılmıştır.

Tahmin modelini belirlemek için birden fazla katman ve birden fazla sinir hücresi sayısı denenmiştir. En düşük hata tahmini veren katman ve sinir hücre sayısı kombinasyonu tahmin modeli olarak seçilmiştir. Modeli eğitmekte ve tahmin yapmakta kullanılan Python kodları EK 4'te verilmiştir. Çalışma sürecinde birçok modeli eğitilmiş ve denenmiştir. Kurulan modeller arasından en düşük tahmin hatası verenler seçilmiştir. Seçilen bu modeller ve modellerin tahmin hataları Çizelge 5-3'te gösterilmektedir.

Çizelge 5-3: Çok Katmanlı algılayıcı: en iyi modeller

Model no	Gizli Katman Sayısı	Gizli katmanların sinir hücresi sayıları	Aktivasyon Fonksiyonu	R^2	MSE	RMSE	MAE
1	3	60,120,60	Rectifier	0.9542	16.07	4.008	0.056
2	3	60,120,60	Tanh	0.9613	17.29	4.158	0.058
3	4	90, 120,150,90	Tanh	0.9605	20.38	4.514	0.064
4	4	120,240,240,120	Rectifier	0.9548	15.56	3.94	0.056
5	**5**	**180,360,90,180,360**	**Rectifier**	**0.9637**	**14.48**	**3.81**	**0.053**

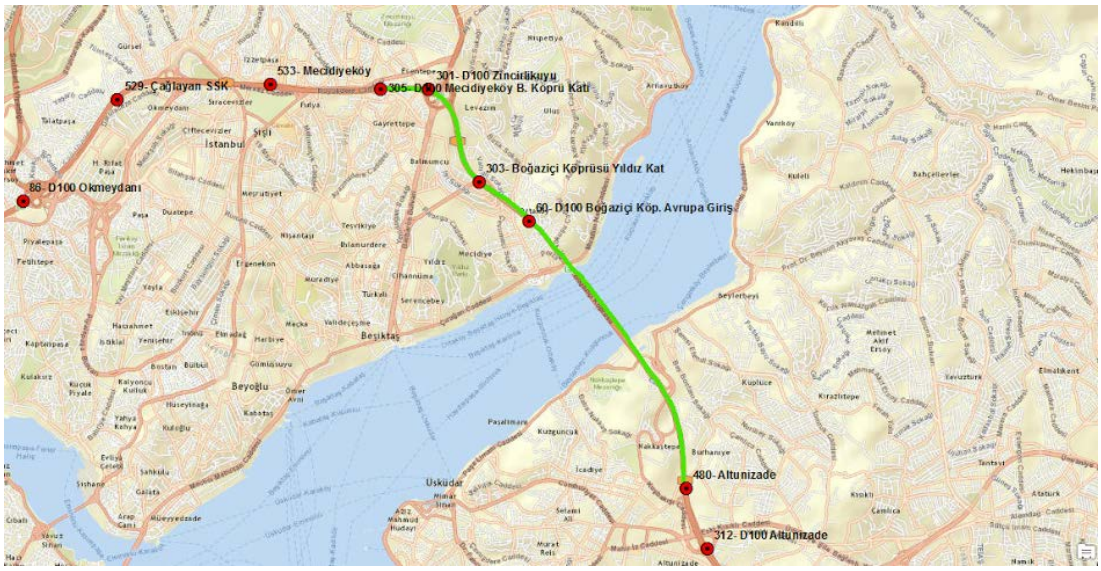
Denenen birçok yapay sinir ağı modelinden Çizelge 5-3'te tahmin hatası bazında en az hata verenleri gösterilmektedir. Bu modellerin neredeyse tamamının ARIMA ile yapılan tahminlerden daha düşük hatalar ile bu veri seti üzerinde tahminler yaptığı söylenebilir. Bu modeller arasından bir tahmin modeli seçmek gerekirse bu beş numaralı model olmalıdır. Bir sonraki bölümde beş numaralı model ile örnek teşkil etmesi açısından küçük bir uygulama yapılacaktır.

Modelin eğitiminde daha öncede bahsedildiği gibi hata en küçüklemesi için ağırlıklar her adımda güncellenmektedir. Buna ağırlıkları optimizasyonu denilmektedir. Seçilen model ile test seti üzerinde tahmin yapılmıştır. Yapılan tahminler sonucu Çizelge 5-3' te gösterilen hata değerleri elde edilebilmiştir.

5.3 Örnek Uygulama: Seçilen Derin Öğrenme Modeli ile Ortalama Hız Tahmini

Seçilen derin öğrenme modeli ile yani en az hata ile tahmin yapan çok katmanlı algılayıcı modelini kullanarak çalışmanın amacında olduğu gibi iki sensör arası ortalama hız ve varış süresi tahmininde bulunalım.

Deneme seti olarak pilot bölgeden Boğaziçi köprüsünü de içine alan kısa bir rota belirler isek, bu rota Şekil 5-1'de olduğu gibidir.



Şekil 5-1: Örnek seçilen rota

Örnek uygulama için pilot bölge içerisinden Mecidiyeköy ve Altunizade arası seçilmiş ve bu rota Şekil 5-1' de yeşil renkle gösterilmiştir. Bu örnek uygulamada amaç Mecidiyeköy'den çıkan bir kişinin güneşli günde, öğlen saat 12'de bu iki nokta arasında ki yolu kaç dakikada gideceğini tahmin edebilmektir. Bölüm 5.2 'de, 'de gösterilen 5 numaralı model kullanılacaktır. Yani 5 gizli katmandan oluşan, 180,360,90,180,360 sayıda sinir hücresi içeren, Rectifier aktivasyon fonksiyonunu kullanan model ile tahmin yapılacaktır. Eğitilen bu modelin eğitim setine %96.37 uyduğu bölüm 5.2 Çizelge 5-3'te de görülmektedir.

Tahmin modelinden çıktı yani tahmin elde edebilmek için girdi değeri olarak bir vektör verilmelidir. Bu vektör derin öğrenme modelini eğittiğimiz bağımsız değişkenlerin tamamından oluşmalıdır. Aşağıda olası binlerce durumdan birkaçı için ortalama hızlar tahmin edilmiştir.

Durum 1:

Bu modele girdi olarak salı günü için, hava durumunun 21 derece olduğu sıcaklık, saat öğlen 12, etkinlik değeri olarak özel bir gün olmadığı varsayılarak sıfır değeri, hava şartının güneşli olduğu durumda konum olarak belirlenen rota üzerinde ki noktalar sırayla verilmiştir. Rota üzerinde seçilen bölge 5 sensör noktasından oluşmaktadır. Rota üzerindeki 5 nokta için ayrı ayrı ortalama hız tahminleri elde edilmiştir. Bu hızlara göre bu noktalar arası km'ler dikkate alınarak ortalama varış süresi tahmini yapılmalıdır. Bu noktalar arası km'ler ve bu noktalarda tahmin edilen hızlar

Çizelge 5-4'de gösterildiği gibidir. Burada önemli bir nokta not edilmelidir ki, bu rota üzerindeki kırmızı ışıklar, kazalar, herhangi bir yol çalışması bu aşamada ihmal edilmiştir.

Çizelge 5-4: Örnek rota ortalama ulaşma süresi tahmini

Seçilen noktalar		**Tahmin edilen ort.**	
arası	**Uzaklık (m)**	**Hız (km/s)**	**Ulaşma (dk)**
1-2	529	70.32	0.451
2-3	1205	67.48	1.07
3-4	707	58.66	0.723
4-5	3523	62.16	3.40
		TOPLAM	**5.64 dk**

Burada toplam 5.64 seçilen kısa bir rota için salı günü güneşli bir havada öğlen 12 saatlerinde Mecidiyeköy Altunizade arasını gitmek isteyen birisinin harcaması gereken ortalama süre olarak tahmin edilmiştir.

Durum 2:

Bu modele girdi olarak durum 1 de olduğu gibi salı günü için, hava durumunun 21 derece olduğu sıcaklık, saat öğlen 12, etkinlik değeri olarak özel bir gün olmadığı varsayılarak sıfır değeri, fakat bu defa hava şartlarının yağışlı olduğu durumda, konum olarak belirlenen rota üzerindeki noktalar sırayla verilmiştir. Rota üzerinde seçilen bölge 5 sensör noktasından oluşmaktadır. Rota üzerinde ki 5 nokta için ayrı ayrı ortalama hız tahminleri elde edilmiştir. Bu hızlara göre bu noktalar arası km'ler dikkate alınarak ortalama varış süresi tahmini yapılmalıdır. Bu noktalar arası km'ler ve bu noktalarda tahmin edilen hızlar Çizelge 5-5'te gösterildiği gibidir. Burada önemli bir nokta not edilmelidir ki, bu rota üzerindeki kırmızı ışıklar, kazalar, herhangi bir yol çalışması bu aşamada ihmal edilmiştir.

Çizelge 5-5: Örnek rota ulaşma süresi hesabı 2

Seçilen noktalar		Tahmin edilen ort.	
arası	**Uzaklık (m)**	**Hız (km/s)**	**Ulaşma (dk)**
1-2	529	74.58	0.425
2-3	1205	76.65	0.943
3-4	707	80.2	0.529
4-5	3523	82.3	2.568
		TOPLAM	**4.47 dk**

Burada toplam 4.47 seçilen kısa bir rota için salı günü yağmurlu bir havada öğlen 12 saatlerinde Mecidiyeköy Altunizade arasını gitmek isteyen birisinin harcaması gereken ortalama süre olarak tahmin edilmiştir.

Bu iki durum oluşabilecek binlerce durumdan sadece ikisidir. Bu şekilde eğitilen bir modelle yola çıkmak isteyen bir kişi önceden hava durumunu bileceği için gideceği rota için ortalama hız tahmini hesaplayabilir. Bu iki tahminden gözlemlenebilecek şaşırtıcı bir durum ise aynı rota için hava durumu dışında aynı koşullar altında; yağmurlu durumda güneşli duruma göre İstanbul trafiğinde ortalama hız daha yüksek olacağı tahmin edilmektedir.

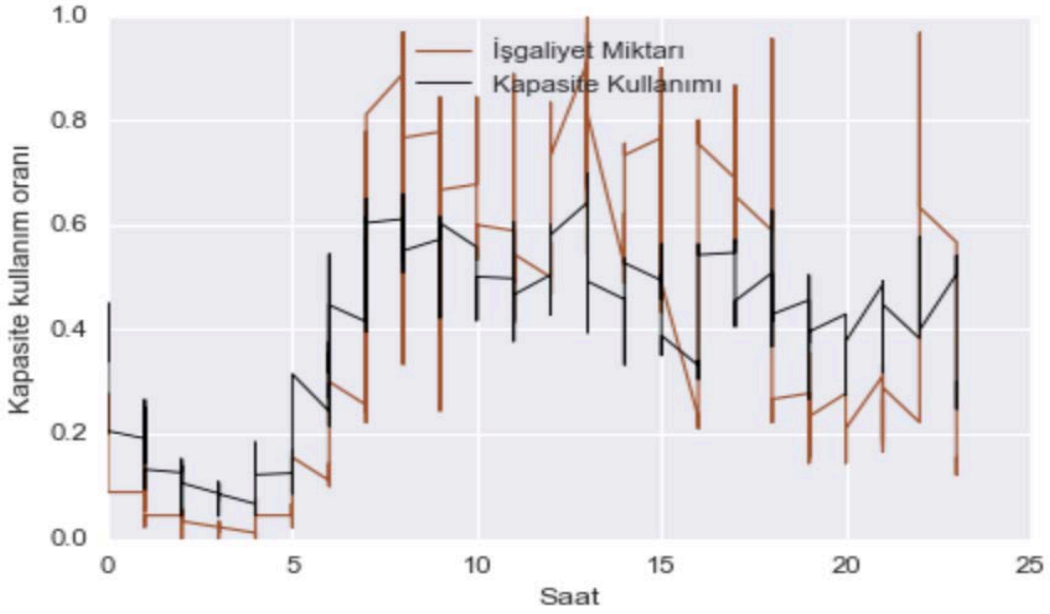
5.4 Yolun Kapasitesi

Elimizde bulunan veri seti ile hesaplanabilecek başka bir şey ise yolun maksimum kapasitesi ve hangi trafik koşullarında bu kapasitenin ne kadarının kullanıldığıdır. Kapasiteyi bildiğimiz zaman normal trafik akışında o yolun kaç araç kaldıracağını da bildiğimizi söyleyebiliriz. Bu bilgi birçok organizasyona bazlı tahminde oldukça faydalı olacaktır.

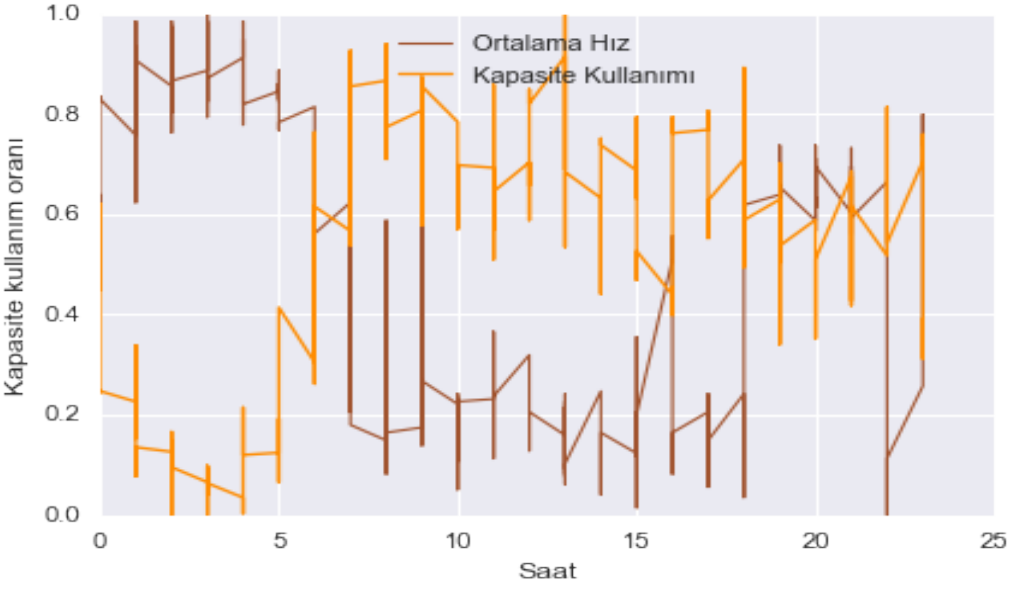
Kapasite bir yol için o ana kadar gözlemlenen hız ve işgaliyet miktarlarının çarpımlarının en büyük değeri olarak ifade edilebilir. Kapasite parametresinin yolun işgaliyet miktarı, yoldaki ortalama hız ve araç sayısı ile ilişkisi olduğu düşünülmektedir. Bu sebeple daha anlaşılır olması için birkaç sensör noktasının kapsadıkları yollar için kapasiteleri ve hangi şartlarda yolun kapasitesinin yüzde kaçının kullanıldığını gözlemlenecek, hız ve araç sayısı ile olan ilişkisi grafik üzerinde gösterilecektir.

5.4.1 51 Numaralı sensör için kapasite ve kapasite kullanımı hesabı

Bu sensör için maksimum kapasite 4410 bulunmuştur. Maksimum işgaliyet: 84, ortalama hız: 52.5 değerleri kullanılarak hesaplanmıştır. Veriler farklı skalada oldukları için normalize edilmemiş hali Şekil 5-2'de ve normalize edilmiş değerler ise Şekil 5-3'te gösterilmektedir.

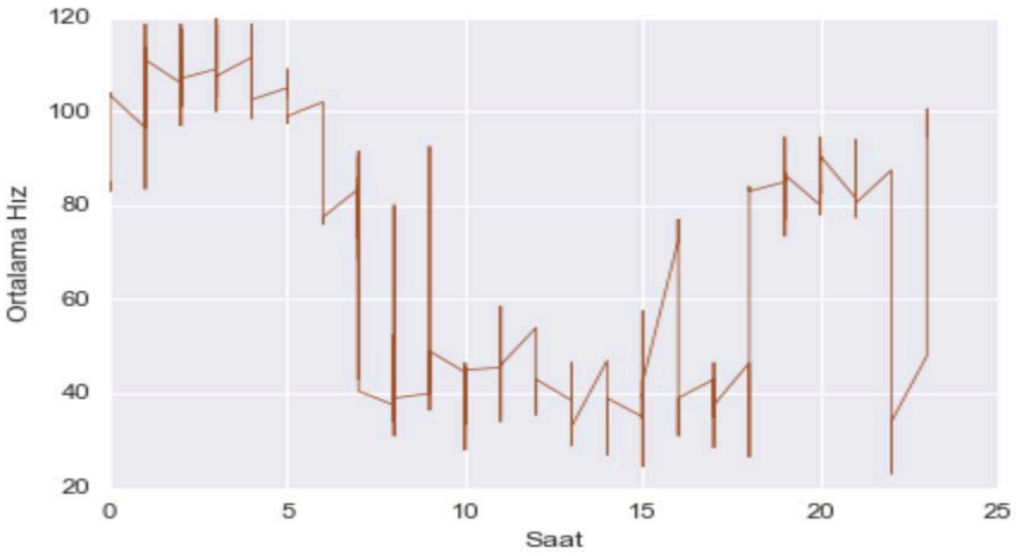


Şekil 5-2:Saat kapasite kullanım oranı- Sensör 51



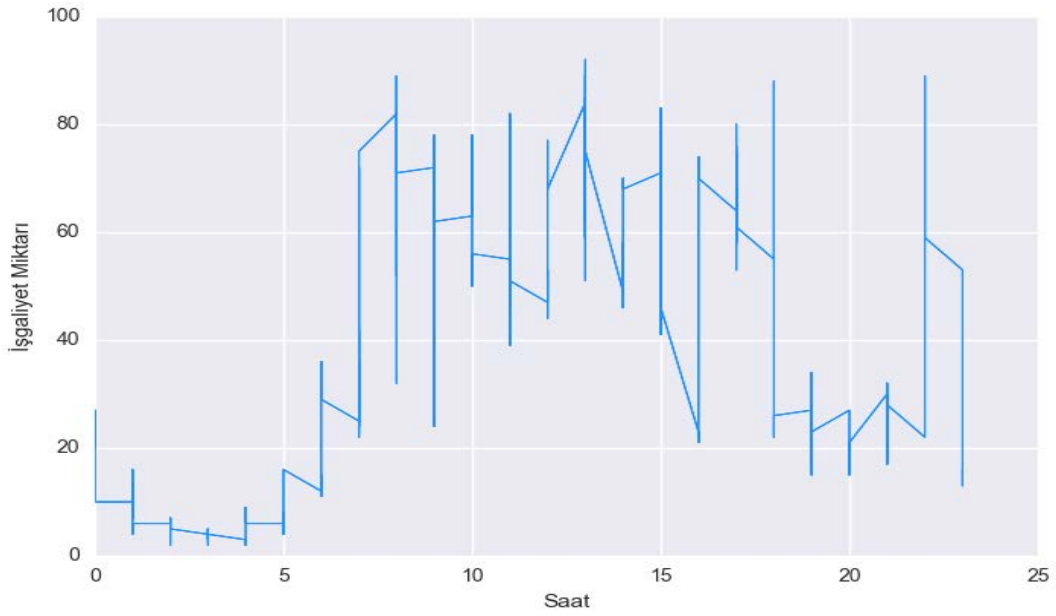
Şekil 5-3: Kapasite kullanım oranı- Sensör 51

Şekil 5-4'de 51 numaralı sensörün zamana bağlı ortalama hız miktarları gerçek değerler üzerinden gösterilmiştir. İşgaliyet miktarı, araç sayısı ve ortalama hızın zamana bağlı dağılımları sırayla grafikler üzerinde gösterilecektir. Burada amaç bu parametrelerin zamanla değişimini görerek Şekil 5-3'de gösterilen kapasite kullanımı ile ilişkilendirebiliriz.



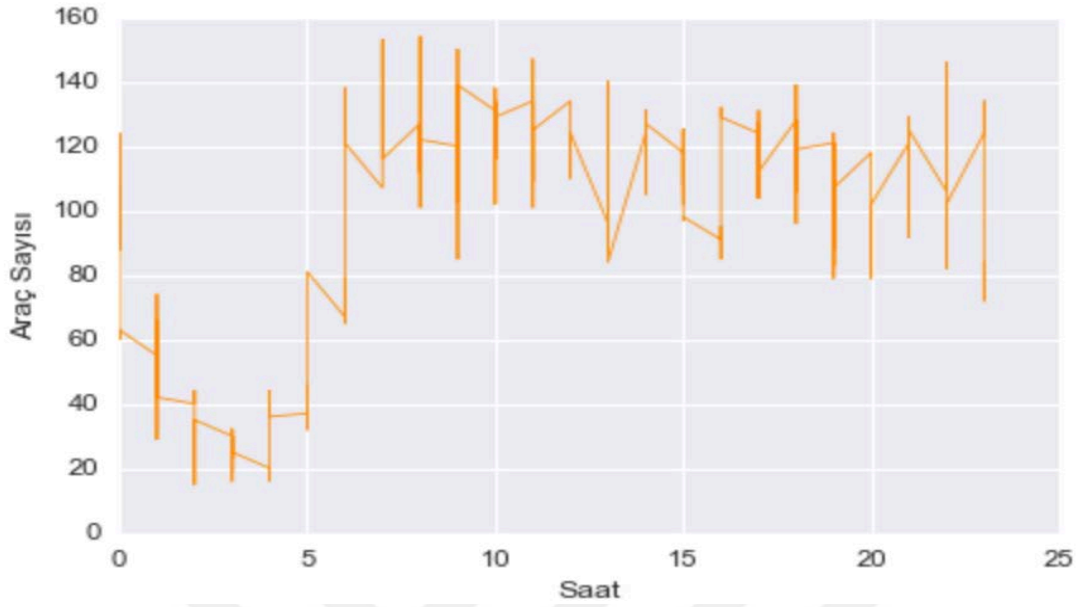
Şekil 5-4:Saat ortalama hız grafiği- Sensör 51

Şekil 5-5'de 51 numaralı sensör için zamana bağlı işgaliyet miktarı gösterilmektedir ve Şekil 5-5 'de normalize edilmemiş gerçek değerler gösterilmektedir.



Şekil 5-5: Saat işgaliyet miktarı grafiği- Sensör 51

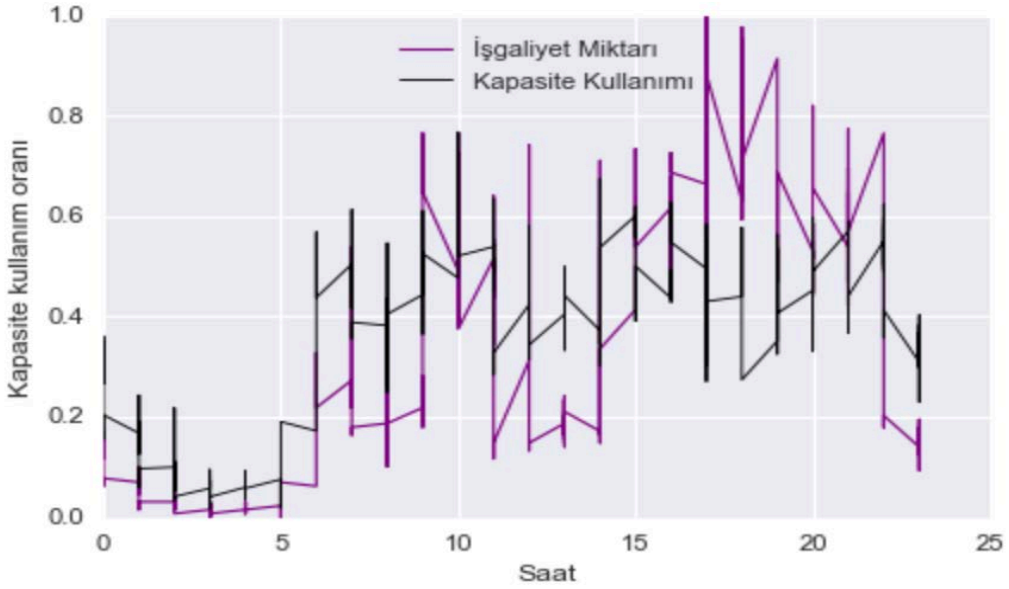
Şekil 5-6‘de zamana bağlı araç sayısı gösterilmektedir. Şekil 5-6‘de normalize edilmemiş gerçek değerleri görebiliriz.



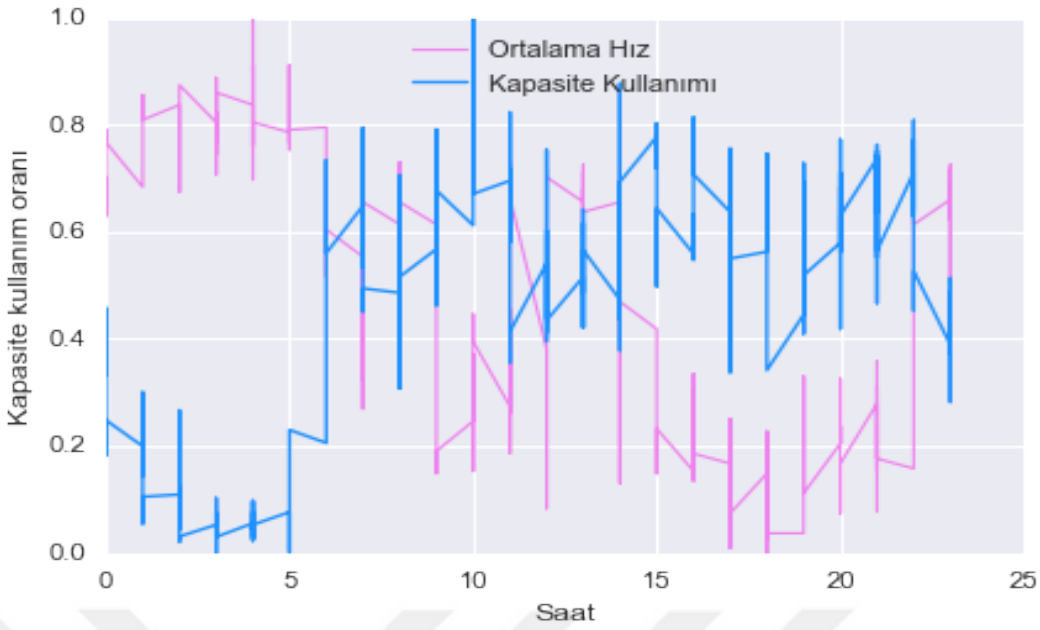
Şekil 5-6: Saat araç sayısı grafiği - Sensör 51

5.4.2 60 Numaralı sensör için kapasite ve kapasite kullanımı hesabı

Bu sensör için maksimum kapasite 5031 bulunmuştur. Maksimum işgaliyet: 78, ortalama hız: 64.5 değerleri kullanılarak hesaplanmıştır. Veriler farklı skalada oldukları için Şekil 5-7’da ve Şekil 5-8‘de normalize edilmiş değerler gösterilmektedir.

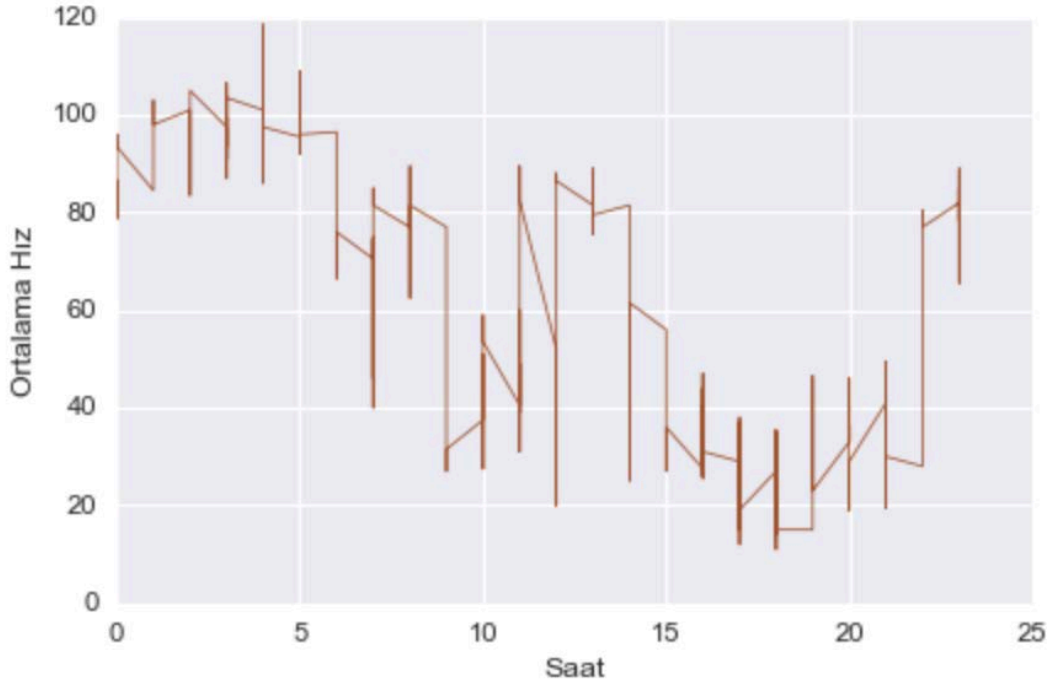


Şekil 5-7: Saat kapasite kullanım oranı-Sensör 60



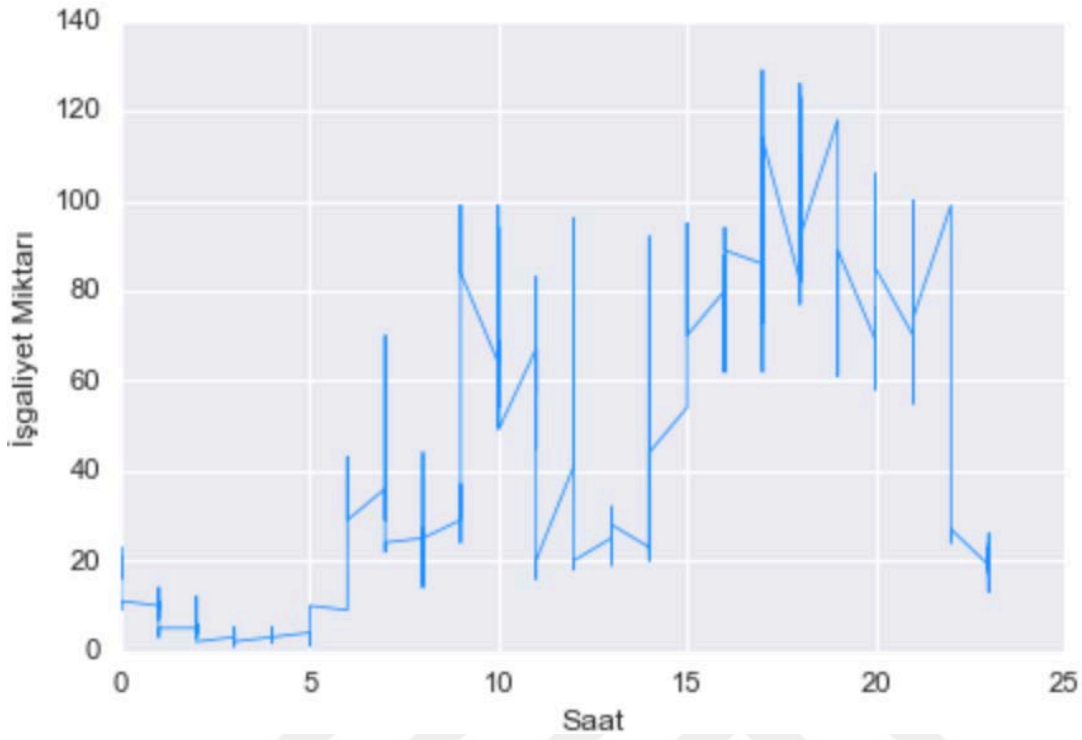
Şekil 5-8: Kapasite kullanım oranı- Sensör 60

Şekil 5-9'de 60 numaralı sensörün zamana bağlı ortalama hız gerçek değerler üzerinden gösterilmiştir. İşgaliyet miktarı, araç sayısı ve ortalama hızın zamana bağlı dağılımları sırayla grafikler üzerinde gösterilecektir. Burada amaç bu parametrelerin zamanla değişimini görerek Şekil 5-8'de gösterilen kapasite kullanımı ile ilişkilendirebiliriz.



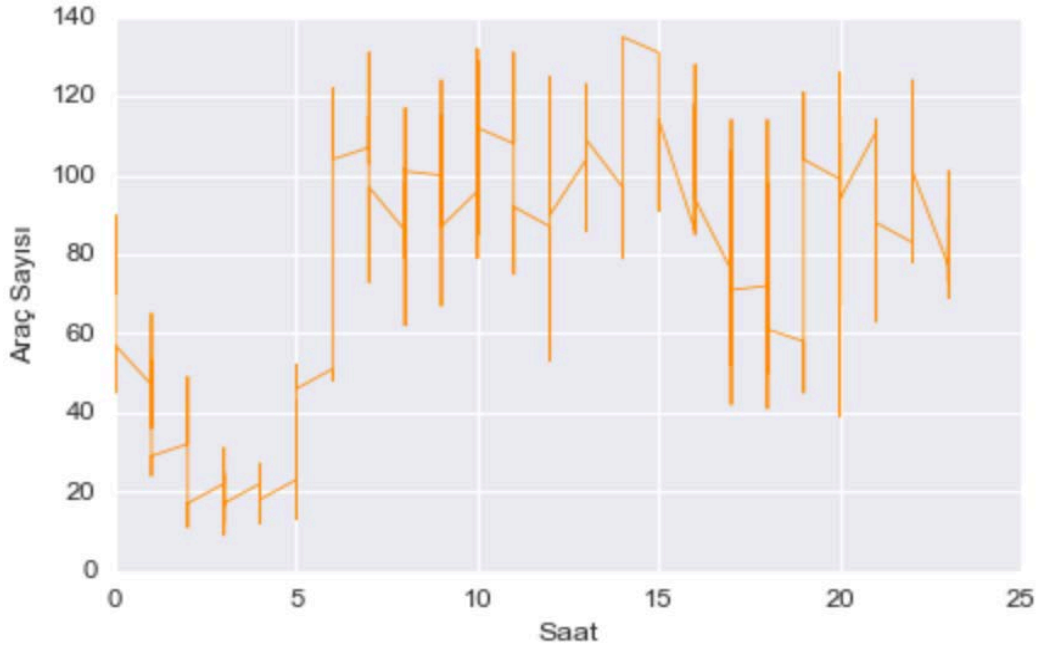
Şekil 5-9:Saat ortalama hız grafiği- Sensör 60

Şekil 5-10‘da 60 numaralı sensör için zamana bağlı işgaliyet miktarı gösterilmektedir ve Şekil 5-10 ‘da normalize edilmemiş gerçek değerler gösterilmektedir.



Şekil 5-10: Saat işgaliyet miktarı grafiği- Sensör 60

Şekil 5-11‘de zamana bağlı araç sayısı gösterilmektedir. Şekil 5-11‘de normalize edilmemiş gerçek değerleri görebiliriz.



Şekil 5-11: Saat araç sayısı grafiği - Sensör 60

İki sensör içinde grafikler incelendiği zaman kapasite kullanımının ortalama hız, araç sayısı ve işgaliyet ile benzer ilişki de olduğu gözlemlenmektedir. Örneğin; araç sayısı ve doğal olarak işgaliyetin düşmesi ile ortalama hız artmakta ve kapasite kullanım azalmaktadır. Veya kapasite kullanımının geceye göre sabah işe gidiş saatlerinde arttığı ve tüm güne oranla akşam saatlerinde oldukça arttığı grafiklerden kolayca gözlemlenebilir.

6. TARTIŞMA VE GELECEĞE YÖNELİK ÇALIŞMALAR

Trafiğin insanların hayatında özellikle büyük şehirlerde yaşayan insanların hayatında kapladığı yer her geçen gün artmaktadır, çünkü trafikte kaybedilen zaman diğer işler için ayrılan zamandan çalmaktadır. Bu sebeple trafik ile ilgili yapılacak iyileştirme çalışmaları aynı zamanda insan hayatını bir nebzede olsa iyi yönde etkileyecektir.

Elde edilen sonuçlardan görülmüştür ki Çok Katmanlı Algıyacılar İstanbul trafiğini modellemekte başarılı olmuşlardır. Fakat bu çalışmada direk ham veri ile tahmin yapılmamış ham veriden günler ve saatler özellik olarak ayrılmış aynı zamanda veriye özel gün bilgisi hava durumu ve hava koşulu bilgisi eklenmiştir. Bu şartlar altında Çok Katmanlı algılayıcılar İstanbul trafiğini modellemekte başarılıdır diyebiliriz.

Gelecek çalışmalarda bu çalışmada geliştirilen modelin parametreleri daha fazla geliştirilebilir veya trafiği etkilediği düşünülen diğer etmenler modele eklenerek modeli o şekilde eğitmek denenebilir.

Gelecekte yapılabilecek başka bir çalışma ise yeterince geliştiği düşünülen modeli otomatikleştirilmiş bir sisteme entegre ederek, halkın kullanımına açmak aynı zamanda sürekli gelen yeni veri ile sistemi besleyerek tahmin modeline her geçen gün biraz daha iyileştirmek olabilir.

KAYNAKLAR

[1] **Zhang, J., Wang, F. Y., Wang, K., Lin, W. H., Xu, X., & Chen, C**. (2011). Data-driven intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, *12*(4), 1624-1639.

[2] **Lv, Y., Duan, Y., Kang, W., Li, Z., & Wang, F. Y**. (2015). Traffic flow prediction with big data: a deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, *16*(2), 865-873.

[3] **Sun, S., Zhang, C., & Yu, G**. (2006). A Bayesian network approach to traffic flow forecasting. *IEEE Transactions on intelligent transportation systems*, *7*(1), 124-132.

[4] **Kirby, H. R., Watson, S. M., & Dougherty, M. S**. (1997). Should we use neural networks or statistical models for short-term motorway traffic forecasting. *International Journal of Forecasting*, *13*(1), 43-50.

[5] **Zhong, M., Sharma, S., & Lingras, P**. (2005). Short-term traffic prediction on different types of roads with genetically designed regression and time delay neural network models. *Journal of computing in civil engineering*, *19*(1), 94-103.

[6] **Yu, E. S., & Chen, C. R**. (1993). Traffic Prediction Using Neural Networks.

[7] **Smith, B. L., & Demetsky, M. J**. (1997). Traffic flow forecasting: comparison of modeling approaches. *Journal of transportation engineering*, *123*(4), 261-266.

[8] **Lee, S., & Fambro, D. B**. (1678). Application of Subset AutoregressiveIntergrated Moving Average Model For Short-Term Freeway Traffic Volume Forecasting. *Transportation Research Record*.

[9] **Zheng, W., Lee, D. H., & Shi, Q**. (2006). Short-term freeway traffic flow prediction: Bayesian combined neural network approach. *Journal of transportation engineering*, *132*(2), 114-121.

[10] **Jeong, Y. S., Byon, Y. J., Castro-Neto, M. M., & Easa, S. M**. (2013). Supervised weighting-online learning algorithm for short-term traffic flow prediction. *IEEE Transactions on Intelligent Transportation Systems*, *14*(4), 1700-1707.

[11] **Dougherty, M. S., & Cobbett, M. R**. (1997). Short-term inter-urban traffic forecasts using neural networks. *International journal of forecasting*, *13*(1), 21-31.

[12] **Ishak, S., Kotha, P., & Alecsandru, C**. (2003). Optimization of dynamic neural network performance for short-term traffic prediction. *Transportation Research Record: Journal of the Transportation Research Board*, (1836), 45-56.

[13] **Innamaa, S**. (2000, November). Short-term prediction of traffic situationusing MLP-neural networks. In *Proceedings of the 7th world congress on intelligent transport systems, Turin, Italy* (pp. 6-9).

[14] **Chan, K. Y., Dillon, T. S., Singh, J., & Chang, E**. (2012). Neural-network-based models for short-term traffic flow forecasting using a hybrid exponential smoothing and Levenberg–Marquardt algorithm. *IEEE Transactions on Intelligent Transportation Systems*, *13*(2), 644-654.

[15] **Van Lint, J., Hoogendoorn, S., & Van Zuylen, H**. (2002). Freeway travel time prediction with state-space neural networks: modeling state-space dynamics with recurrent neural networks. *TransportationResearch Record: Journal of the Transportation Research Board*, (1811), 30-39.

[16] **Jin, F., & Sun, S**. (2008, June). Neural network multitask learning for traffic flow forecasting. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on* (pp. 1897-1901). IEEE.

[17] **Shahsavari, B., & Abbeel, P**. (2015). Short-term traffic forecasting: Modeling and learning spatio-temporal relations in transportation networks using graph neural networks.

[18] **Vlahogianni, E. I., Karlaftis, M. G., & Golias, J. C**. (2005). Optimized and meta-optimized neural networks for short-term traffic flow prediction: A genetic approach. *Transportation Research Part C: Emerging Technologies*, *13*(3), 211-234.

[19] **Dia, H**. (2001). An object-oriented neural network approach to short-term traffic forecasting. *European Journal of Operational Research*, *131*(2), 253-261.

[20] **Zheng, W., Lee, D. H., & Shi, Q**. (2006). Short-term freeway traffic flow prediction: Bayesian combined neural network approach. Journal of transportation engineering, 132(2), 114-121.

[21] **Erkan, E., & Aydınoğlu, A. Ç**. Trafik Ölçme Sensörü Verilerinin Zamansal Analizi. arifcagdas.com adresinden 14 Mart 2017 tarihinde alınmıştır.

[22] *İsbak*. (2017, Şubat 14). http://isbak.istanbul/akilli-ulasim-sistemleri/trafik-olcum-sistemi/ adresinden alındı.

[23] İstanbul Büyükşehir Belediyesi Ulaşım Daire Başkanlığı, (2011). İstanbul Metropoliten Alanı Kentsel Ulaşım Ana Planı.

[24] *Openweathermap*. (2017, Mart 20). https://openweathermap.org/history adresinden alındı.

[25] HIGHCHARTS. (2017, Mart 2). http://www.highcharts.com/demo adresinden alındı.

[26] **Montgomery, D. C., Jennings, C. L., & Kulahci, M**. (2015). *Introduction to time series analysis and forecasting*. John Wiley & Sons, 231-250.

[27] **Akdi, Y**. (2003). *Zaman Serileri Analizi Birim Kökler ve Kointegrasyon.* Ankara: Gazi Kitapevi, 47-65, 270-279.

[28] **Kaynar, O., & Taştan, S**. (2009, Temmuz-Aralık). Zaman Serisi Analizinde MLP Yapay Sinir Ağları ve Arıma Modelinin Karşılaştırılması. *Erciyes Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi*(33), 161-172.

[29] **Demuth, H. B., Beale, M. H., Jess, O. D., & Hagan, M**., (2014). Neural Network Design. Martin Hagan, USA.

[30] **Kargı, V. S**. (2015). *Yapay Sinir Ağ Modelleri ve Bir Tekstil Firmasında Uygulama.* Bursa: Ekin Basın Yayın Dağıtım, 23-80.

[31] **Graupe, D**. (2013). *Principles of artificial neural networks* (Vol. 7). World Scientific.

[32] **Deng, L**. (2014). A tutorial survey of architectures, algorithms, and applications for deep learning. *APSIPA Transactions on Signal and Information Processing*, *3*, e2.

[33] *Rsipvision*. (2017, Mart 10). http://www.rsipvision.com/exploring-deep-learning/ adresinden alındı

[34] **Marsland, S**. (2009). *Machine Learning An Algoritmic Perspective.*Cambridge: Ralf Herbrich and Thore Graepel Microsoft Research, 47-89.

[35] **Ng, A**., (2011). CS229: Machine Learning Lecture Notes. *Stanford University.*

[36] **KERAS**. (2016, Kasım 20). https://keras.io adresinden faydalanılmıştır.

[37] **Svozil, D., Kvasnicka, V., & Pospichal, J**. (1997). Introduction to multi-layer feed-forward neural networks. *Chemometrics and intelligent laboratory systems*, *39*(1), 43-62.

EKLER

EK 1: Programlama Veri Ön Hazırlık Kodları

EK 2: Java programlama Hatalı Veri Ayıklama

EK 3: Dickey Fuller Kritik Değer Tablosu

EK 4: Python Programlama Geri Beslemeli Sinir Ağ

EK 1: R Programlama Veri Ön hazırlık Kodları

```
names(d602) <- c('index','Date', 'Hours', 'rtmsno', 'dn', 's1', 's2', 's3', 's4', 's5', 's6', 's7',
's8', 's_gelis', 's_gidis', 'v1', 'v2', 'v3', 'v4', 'v5', 'v6', 'v7', 'v8', 'o1', 'o2', 'o3', 'o4', 'o5',
'o6', 'o7', 'o8', 'o_gelis', 'o_gidis', 'vl1', 'vl2','vl3','vl4', 'vl5', 'vl6', 'vl7','vl8',
'msgno','statuscode')
d51_subset <- d602

library(lubridate)
d51_subset$Date <- as.Date(d51_subset$Date, "%y-%m-%d")
d51_subset$Julian <- yday(d51_subset$Date)
d51_subset$day <- weekdays(as.Date(d51_subset$Date))

library(dplyr)
D51_s <- left_join(D51, hava[c('Julian','havadurumu', 'havaKosulu')], by='Julian')
D311 <- D51_s[, c(1:6,9,10,7,8)]
```

EK 2: Java Programlama Hatalı Veri Ayıklama

```
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import au.com.bytecode.opencsv.CSVReader;
public class App {
public static void main(String[] args) throws IOException {
CSVReader reader = new CSVReader(new FileReader("D:\\deneme.csv"));
                String[] nextLine;
                int i = 0;
                int index = 0;
                ArrayList<Integer> list = new ArrayList<Integer>();
                while ((nextLine = reader.readNext()) != null &&  index != 1048577)
{       index++;
```

```
            i++;
             list.add(Integer.parseInt(nextLine[6]));
             if (i > 9) {
                    int k = 0;
                    for (int j = 0; j < list.size(); j++) {
k = k + list.get(j);
}
                int ort = k / 10;
for (int m = 0; m < list.size(); m++) {
            if (list.get(m) < (ort - 45) || list.get(m) > (ort + 45)) {
System.out.println("index degeri...." + (index - 9 + m) + " ...satir degeride..." +
list.get(m) + "...ortalama.." + ort);
}
}
            ort = 0;
            list = new ArrayList<Integer>();
             i = 0;
}
}
}
}
```

EK 3: Dickey Fuller Kritik Değer Tablosu

		Probability to the Right of Critical Value							
Model	Statistic	N	1%	2.5%	5%	10%	90%	95%	97.5%	99%
Model I (no constant, no trend)										
	ADF_{tr}	25	-2.66	-2.26	-1.95	-1.60	0.92	1.33	1.70	2.16
		50	-2.62	-2.25	-1.95	-1.61	0.91	1.31	1.66	2.08
		100	-2.60	-2.24	-1.95	-1.61	0.90	1.29	1.64	2.03
		250	-2.58	-2.23	-1.95	-1.61	0.89	1.29	1.63	2.01
		500	-2.58	-2.23	-1.95	-1.61	0.89	1.28	1.62	2.00
		>500	-2.58	-2.23	-1.95	-1.61	0.89	1.28	1.62	2.00

Model II (constant, no trend)

ADF_{tr}	25	-3.75	-3.33	-3.00	-2.62	-0.37	0.00	0.34	0.72
	50	-3.58	-3.22	-2.93	-2.60	-0.40	-0.03	0.29	0.66
	100	-3.51	-3.17	-2.89	-2.58	-0.42	-0.05	0.26	0.63
	250	-3.46	-3.14	-2.88	-2.57	-0.42	-0.06	0.24	0.62
	500	-3.44	-3.13	-2.87	-2.57	-0.43	-0.07	0.24	0.61
	>500	-3.43	-3.12	-2.86	-2.57	-0.44	-0.07	0.23	0.60

Model III (constant, trend)

ADF_{tr}	25	-4.38	-3.95	-3.60	-3.24	-1.14	-0.80	-0.50	-0.15
	50	-4.15	-3.80	-3.50	-3.18	-1.19	-0.87	-0.58	-0.24
	100	-4.04	-3.73	-3.45	-3.15	-1.22	-0.90	-0.62	-0.28
	250	-3.99	-3.69	-3.43	-3.13	-1.23	-0.92	-0.64	-0.31
	500	-3.98	-3.68	-3.42	-3.13	-1.24	-0.93	-0.65	-0.32
	>500	-3.96	-3.66	-3.41	-3.12	-1.25	-0.94	-0.66	-0.33

EK 4: Python Programlama Geri Beslemeli Sinir Ağ

```
def sigmoid(x):
    return 1/(1 + np.exp(-x))

class SigmoidLayer:
    def __init__(self, n_input, n_output):
        self.W = np.random.randn(n_output, n_input)
        self.b = np.random.randn(n_output, 1)
    def output(self, X):
        if X.ndim == 1:
            X = X.reshape(-1, 1)
        return sigmoid(self.W.dot(X) + self.b)
```

```
class SigmoidNetwork:

    def __init__(self, layer_sizes):
        '''
        :parameters:
            - layer_sizes : list of int
                List of layer sizes of length L+1 (including the input dimensionality)
        '''
        self.layers = []
        for n_input, n_output in zip(layer_sizes[:-1], layer_sizes[1:]):
            self.layers.append(SigmoidLayer(n_input, n_output))

    def train(self, X, y, learning_rate=0.2):
        X = np.array(X)
        y = np.array(y)
        if X.ndim == 1:
            X = X.reshape(-1, 1)
        if y.ndim == 1:
            y = y.reshape(1, -1)

        # Forward pass - compute a^n for n in {0, ... L}
        layer_outputs = [X]
        for layer in self.layers:
            layer_outputs.append(layer.output(layer_outputs[-1]))

        # Backward pass - compute \partial C/\partial z^m for m in {L, ..., 1}
        cost_partials = [layer_outputs[-1] - y]
        for layer, layer_output in zip(reversed(self.layers), reversed(layer_outputs[:-1])):
            cost_partials.append(layer.W.T.dot(cost_partials[-1])*layer_output*(1 - layer_output))
        cost_partials.reverse()
```

```
        # Compute weight gradient step
        W_updates = []
        for cost_partial, layer_output in zip(cost_partials[1:], layer_outputs[:-1]):
            W_updates.append(cost_partial.dot(layer_output.T)/X.shape[1])
        # and biases
        b_updates = [cost_partial.mean(axis=1).reshape(-1, 1) for cost_partial in
cost_partials[1:]]

        for W_update, b_update, layer in zip(W_updates, b_updates, self.layers):
            layer.W -= W_update*learning_rate
            layer.b -= b_update*learning_rate

    def output(self, X):
        a = np.array(X)
        if a.ndim == 1:
            a = a.reshape(-1, 1)
        for layer in self.layers:
            a = layer.output(a)
        return a
```

ÖZGEÇMİŞ

Ad-Soyad: Nezahat Sönmez
Uyruğu: T.C.
Doğum Tarihi ve Yeri: 05.08.1992- Adana
E-posta: sonmeznezahat@gmail.com

ÖĞRENİM DURUMU:

- **Lisans:** 2014, Eskişehir Osmangazi Üniversitesi, Fen-Edebiyat Fakültesi, İstatistik Bölümü.

MESLEKİ DENEYİM VE ÖDÜLLER:

Yıl	Yer
2016-2017	Via-Vis Bilişim Tic. Ltd. Şti.
2017-	Hugo Boss Tekstil Sanayi Ltd. Şti.

YABANCI DİL: İngilizce

TEZDEN TÜRETİLEN YAYINLAR, SUNUMLAR VE PATENTLER:

Sönmez, N., Hanalioğlu, T., Tekin, S., Predicting the Traffic Flow with Using Traffic Sensors: An Application for Istanbul City, *International Conference on Mathematics and Mathematics Education(ICMME-2017), Harran University, Şanlıurfa, 11-13 May 2017*