İSTANBUL ÜNİVERSİTESİ CERRAHPAŞA MÜHENDİSLİK FAKÜLTESİ BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BİTİRME PROJESİ

DERİN ÖĞRENME İLE BANKA KREDİSİ RİSK ANALİZİ

Hazırlayan : HASAN BATUHAN GÖKÇE

Danışman : DOÇ.DR.DERYA YILTAŞ KAPLAN

MAYIS 2022

ÖNSÖZ

Bu çalışmada yapay sinir ağları modelleme teknikleri kullanılarak banka kredisi risk analizi çalışması yapılmıştır.

Bu çalışmanın ortaya konmasına katkıda bulunan ve yardımlarını esirgemeyen danışman hocam Sn. Doç. Dr. Derya Yıltaş KAPLAN'a, üniversiteye başladığım günden bugüne kadar bana yol gösteren, en ufak katkıda bulunan bütün hocalarıma, eğitim hayatım boyunca yanımda olan ve desteklerini esirgemeyen aileme teşekkürlerimi sunarım.

Hasan Batuhan Gökçe

ÖZET

Büyük veri, yapay zekâ, makine öğrenmesi, derin öğrenme teknolojileri günlük hayatımızda artık sıkça duyulan kavramlar haline gelmiştir. Zaman ilerledikçe üretilen veri miktarı geleneksel veri tabanı yönetim sistemleri ile depolanması, işlenmesi, yönetilmesi mümkün olmayan bir hale gelmiştir. Yapılan bazı araştırmalara göre, insanlığın sahip olduğu veri miktarının iki katına çıkması 1900'lü yıllarda 100 yıla yakın sürerken, günümüzde bu sürenin 13 aya kadar düştüğü gözlemlenmiştir. IBM'e göre günde yaklaşık 2,5 milyar GB veri üretilmektedir. Yukarıda bahsedilen kavramlar, bu kadar veriyi işleyebilmek için üretilmiş teknolojilerden bazılarıdır.

Derin Öğrenme: Yapay sinir ağlarını temel alarak öğrenme işlemini gerçekleştiren makine öğrenmesinin alt dalı olarak isimlendirilebilir. Yapay sinir ağları: İnsan beynini oluşturan nöronları taklit edip modelleyerek makinelerin öğrenim yapmasını ve gerçek hayatta insan benzeri görevlerin makineler tarafından yerine getirilebilmesini sağlayan modellemedir.

Kredi riski bankacılıkta öne çıkan risklerden birisi olup bankaların karlılık oranlarını üzerinde önemli etkiye sahiptir. Buna bağlı olarak, bankalar ve diğer finans kuruluşları için tüketicilere kredi verme konusunda karar vermede yardımcı kredi puanlama sistemleri geliştirmek önem arz etmektedir. Finansal kuruluşlar, kredi/borç talep eden müşterilerine kredi kullandırma kararlarında izleyecekleri yolu belirleyebilmek için, kredi skoru üzerinde etkisinin olduğu düşünülen faktörler arası ilişkileri ortaya koyan çeşitli içsel kredi değerlendirme modellerine başvurmaktadır. Literatürde, kredi puanlama analizlerinde istatistik ve makine öğrenme teknikleri yaygın olarak incelenmiştir. Bu çalışmada başta bankalar olmak üzere finansal kuruluşlar ve bu kuruluşların müşterileri için de önem arz eden müşteri kredi skorunun belirlenmesi konusu ele alınmaktadır. İstatistiksel teknikler ve makine öğrenme teknikleri, son yıllarda ticari kredilerindeki büyüme ile giderek daha önemli hale gelmiştir. İstatistiksel yöntemler geniş bir yelpazede uygulanmış olmasına rağmen ticari gizlilik nedeniyle literatürde sınırlı olarak yer almaktadır. Bu çalışmada, bir bankaya başvurarak kredi talep eden bireysel müşterilerin kredi talebinin kabul edilmesi ya da reddedilmesi kararının verilmesine yönelik, yapay sinir ağları (YSA) metodolojisini temel alan bir yazılım modeli önerilmektedir. Veri seti olarak UCI açık kaynak sitedeki bir Portekiz bankasına ait 41189 adet veriye ve 21 adet niteliğe sahip veri seti kullanılmıştır.

SUMMARY

Big data, artificial intelligence, machine learning, deep learning technologies have become concepts that are frequently heard in our daily lives. As time progresses, the amount of data produced has become impossible to store, process and manage with traditional database management systems. According to some studies, it has been observed that while it took nearly 100 years in the 1900s to double the amount of data that humanity has, it has been observed that this period has decreased to 13 months today. According to IBM, approximately 2.5 billion GB of data is produced per day. The concepts mentioned above are some of the technologies produced to process this much data.

Deep Learning: It can be named as a sub-branch of machine learning that performs the learning process based on artificial neural networks. Artificial neural networks: It is the modeling that enables machines to learn and to perform human-like tasks in real life by imitating and modeling the neurons that make up the human brain.

Credit risk is one of the prominent risks in banking and has a significant impact on the profitability of banks. Accordingly, it is important to develop credit scoring systems that help banks and other financial institutions make decisions about lending to consumers. Financial institutions resort to various internal credit evaluation models that reveal the relationships among the factors that are thought to have an effect on the credit score in order to determine the path they will follow in their decision to extend credit to their customers who request credit/debt. In the literature, statistics and machine learning techniques have been widely studied in credit scoring analysis. In this study, the issue of determining the customer credit score, which is important for financial institutions, especially banks, and their customers, is discussed. Statistical techniques and machine learning techniques have become increasingly important with the growth in commercial loans in recent years. Although statistical methods have been applied in a wide range, they are limited in the literature due to commercial confidentiality. In this study, a software model based on artificial neural networks (ANN) methodology is proposed for the decision to accept or reject the loan request of individual customers who apply to a bank and request a loan. As the dataset, a dataset with 41189 data and 21 attributes belonging to a Portuguese bank on the UCI open source site was used.

1.GİRİŞ

Kredi için bankaya başvuran müşterilere kredi verme, bankaların en temel işlevlerinden biri olmakla beraber aynı zamanda bankaların en riskli faaliyet alanlarından biridir. Bu nedenle, bankaların almış olduğu en önemli risklerden biri olan kredilendirme faaliyetlerini, verimli ve verilen kredilerin geri ödenmesinden doğabilecek zararları minimum düzeyde tutabilecek şekilde yönetmeleri gerekir. Tüm bu sebeplerden ötürü, bankaların son yıllarda kredilendirme faaliyetlerinde hızlı ve sağlıklı kararlar verebilmesini sağlayan veri madenciliği, ayrıştırma analizi, sınıflandırma ve regresyon araçları, bulanık mantık ve yapay sinir ağları gibi teknikleri daha yoğun bir şekilde kullanmaya başladıkları görülmektedir.

Banka ve finans sektörlerinde risklerin doğru yönetilmesi, sahip olunan kaynakların etkin ve verimli kullanılması, potansiyel risklerin önceden tahmin edilerek gerekli önlemlerin bugünden alınmasına bağlıdır. Sorunlu kredilerin önceden tahmini bankalarda kararlılık ve kaynak verimliliği açılarından büyük önem arz etmektedir.

1.1 Literatür Taraması

Literatür araştırmaları sonucunda kredi risk analizi makine öğrenmesi, istatistik teknikleri, veri madenciliği ve birçok teknik kullanılarak çalışmalar yapılmıştır. Hasan Tahsin Oğuz "Saklı Markov Modeli ile kredi risk analizi" adlı çalışmasında saklı markov modelini kullanarak kredi risk analizinin performansını ölçmek ve sınıflandırmak için çalışmasını yürütmüştür. Gülnur Derelioğlu "KOBİ kredi risk analizinde modüler yaklaşım" adlı çalışmasıyla esnaf olan kişilerin işlerini büyütmek ve geliştirmek için kullandıkları KOBİ kredisini analiz edip, yorumlamıştır. Gül Efşan ve Bozkurt Gönen'in ortak çalışması olan "Öznitelik seçme ve transfer öğrenme algoritmaları ve kredi risk analizi üzerine uygulamaları" adlı çalışmasında öznitelik belirlemek için probit sınıflandırıcı ve çoklu çekirdek öğrenimini geliştirmişlerdir. Bu geliştirmeler sonucunda kredi risk analizi veri seti üzerinde kullanarak etkinliği verimliliğini

ölçmektedir. Erkan Çetiner "Sınıflandırma tekniklerinin kredi risk analizi üzerindeki performansı" adlı çalışmasında kredi risk analizi için kullanılan sınıflandırma ölçütlerini geliştirerek yeni bir sınıflandırma yöntemi oluşturmayı amaçlamıştır. Emine Bahçe Çizer "Makine Öğrenmesi Teknikleri ile Kredi Risk Analizi" adlı çalışmasında makine öğrenmesi algoritmaları ve veri madenciliği yöntemleri kullanarak kredi risk analizi yapmıştır. Ömer Yavuz Can "Makine Öğrenmesi Teknikleri Kullanılarak Kredi Risk Analizi" adlı çalışmasında makine öğrenmesi algoritmaları kullanarak müşterilerin krediye uygunluğunu yorumlamıştır. Amir E. Khandani, Adlar J. Kim, Andrew W. Lo ortak çalışması olan "Consumer credit-risk models via machine-learning algorithms" adlı çalışmada müşterilerin kredi taleplerini makine öğrenmesi algoritmaları kullanarak uygun olup olmadığını belirlemeye çalışmışlardır. Lean Yu, Shouyang Wang, Kin Keung Lai kişilerinin "Credit risk assessment with a multistage neural network ensemble learning approach" adlı çalışmasında çok aşamalı bir sinir ağı yapısı kullanarak kredi risk ölçümü yapmayı hedeflemişlerdir. Zhu, Li, Wu, Wang, Liang, "Balancing accuracy, complexity and interpretability in consumer credit decision making: A C-Topsis classification approach" adlı çalışmasında krediyi doğruluk, yorumlanabilirlik ve karmaşıklığını ele alarak yeni bir sınıflandırma ortaya çıkarmışlardır. Li, Shiue, Huang "The evaluation of consumer loans using support vector machines" adlı çalışmasında kredi talebinde bulunan müşterilerin taleplerini değerlendirmek için destek vektör makineleri yöntemini kullanarak bir model oluşturmuşlardır. Huang, Chen, Wang "Credit Scoring with a Data Mining Approach Based on Support Vector Machines" adlı çalışmasında kredi talebinde bulunan kişilerin özniteliklerinden bir kredi puanı oluşturup değerlendirmek için hibrid destek vektör makineleri yöntemi kullanarak kredi puanlama modeli oluşturmuşlardır. Saha, Bose, Mahanti, "A knowledge based scheme for risk assesment in loan processing by banks" adlı çalışmasında kredilendirme aşamalarını denetimini sağlamak için bir önerme gerçekleştirilmiştir. Malhotra, K. Malhotra, "Evaluating consumer loans using neural networks", adlı çalışmasında kredi talebinde bulunan kişileri değerlendirmede çoklu diskriminant analizini ve yapay sinir ağları analizini ele alarak bu iki algoritmaların performanslarını karşılaştırmaktadır. Tsai, Lin, Cheng, Lin "The consumer loan default predicting model – An application of DEA- 3 DA and neural network" başlıklı çalışmasında tüketici kredisi alan kişilerin analizlerini çıkartarak, tüketici kredisi için başvuran kişiler için bir model oluşturmuşlardır.

1.2 Tezin Amacı

Bu çalışmada içerisinde 41189 adet veri bulunan Portekiz Kredi Datası UCI veri setinden yararlanılmıştır. Dataset üzerinde derin öğrenme algoritmaları kullanılarak kredi talebinde bulunan müşterilerin krediye uygunluğu analiz edilmeye çalışılmıştır.

2.YÖNTEM

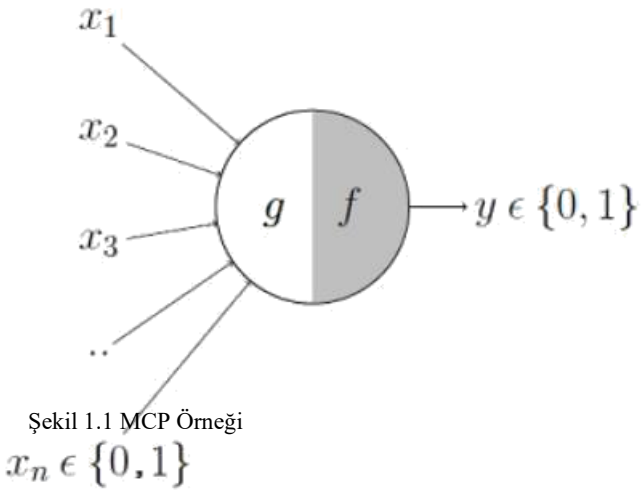
2.1 Yapay Sinir Ağları ve Derin Öğrenme

Yapay sinir ağları, makine öğrenmesi ve yapay zekâ alanları içerisinde kullanılan bir öğrenme metodudur. İnsan beynini taklit etme, simüle etme fikirleri esas alınarak yürütülen çalışmalar esnasında yapay sinir ağı modellemeleri ortaya çıkmıştır. Fikir ilk ortaya çıktığında basit matematiksel işlemleri gerçekleyen modeller üretilmiş olsa da zamanla bu modellemenin kullanımı azaldı. (1)

Ancak veri işlemede kullanılan makine öğrenmesi algoritmaları, çok fazla artan veri miktarı ile bazı problemlerin çözümünde yetersiz kaldı. 2000'li yılların başında, bu alanda emek veren insanların gayretleri ile artık sık kullanılan karar destek mekanizmalı makinelere evrilmişlerdir. 1900'lü yılların ortalarında, sinir bilimi ile ilgilenen bilim insanları nöronların bir makineye entegre olabileceğinin ve insan gibi öğrenebileceğini düşünüyorlardı. (1)

2.1.1 McCulloch-Pitts Yapay Sinir Ağı

1943 yılında sinir bilimci Warren McCulloch ve matematikçi Walter Pitts, nöronların nasıl çalışabileceğine dair bir makale yazdı ve bir yapay sinir ağının ilk modellemesini gerçekleştirdi. Elektrik devrelerini kullanarak basit bir yapay sinir ağını gerçekleştirdiler. Şekil 1.1'de basit bir MCP örneği görülmektedir. (2)



Şekil 1.1 MCP Örneği

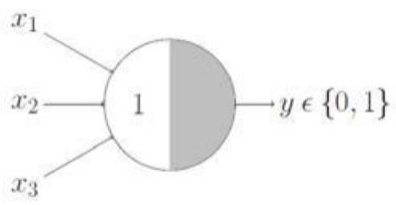
İlk kısım, bir dizi boole girdisini işler. Daha sonra bilgiler, her bir girdinin değerlerinin basit bir şekilde toplanmasını gerçekleştiren bir toplama işlemine sokulur. Ardından, f işlevi g'nin çıktısını bir eşik (threshold) veya koşulla (condition) karşılaştırır.

Bir MCP ağının çalışma mantığı basitçe aşağıda verilmiştir.

- Nöronun çıktıları 0 veya 1'dir. Y=0 ise nöron tetiklenmemiştir. Y=1 ise nöron ateşlenmiştir.
- N adet girdi vardır. Girdi elemanları da 0 ve 1'den oluşmaktadır.
- Bir adet inhibitör girişi vardır. i=1 ise nöron ateşlenmez.
- Θ eşik değerine sahiptir. Girdilerinin toplamı bu eşik değerden büyükse, nöron ateşlenir. Aksi halde nöron çalışmaz. (2)

$x = [x , x , x , \ldots, x]^T$ girişi, engelleyici giriş i ve eşiği Θ verildiğinde, y çıkışı aşağıdaki formüldeki gibi hesaplanır:

2.1.1.1 VEYA (OR) Fonksiyonu

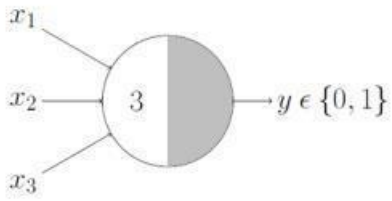


x	y	x+y
0	0	0
0	1	1
1	0	1
1	1	1

Şekil 2.1.1.1 OR Fonksiyonu

Girdilerden herhangi biri 1 ise nöron tetiklenir. (2)

2.1.1.2 VE (AND) Fonksiyonu

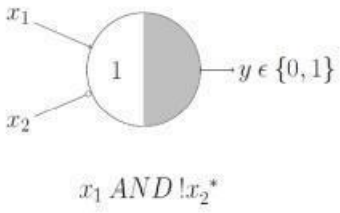


x	y	xy
0	0	0
0	1	0
1	0	0
1	1	1

Şekil 2.1.1.2 AND Fonksiyonu

Girdilerin hepsi 1 ise nöron tetiklenir. (2)

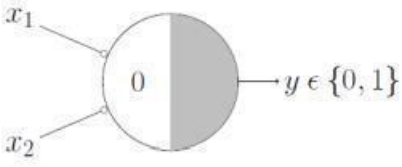
2.1.1.3 Inhibitör Girdili Fonksiyon



Şekil 2.1.1.3 Inhibitör Girdili Fonksiyon

Burada engelleyici bir girdimiz var, yani x2, yani x2=1 olduğunda çıktı 0 olacaktır. Bu durumda x1 AND !x2 işleminin yalnızca x1=1 ve x2=0 olduğunda nöronun çıktısının 1 olacağını biliyoruz. Dolayısıyla eşik değeri 1 olmalıdır. (2)

2.1.1.4 NOR Fonksiyonu

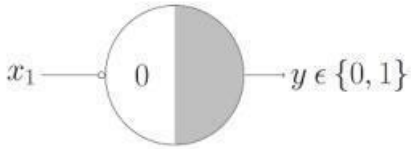


x	y	$x \downarrow y$
0	0	1
0	1	0
1	0	0
1	1	0

Şekil 2.1.1.4 NOR Fonksiyonu

Bu nöronun ateşlenmesi için tüm girdilerin 0 olması gerekmektedir. Bu nedenle eşik değeri 0 olmalıdır. Dolayısıyla tüm girdiler inhibitör girdi olarak düşünülebilir. (2)

2.1.1.5 Tersleme (NOT) Fonksiyonu



x	x'
0	1
1	0

Şekil 2.1.1.5 Tersleme Fonksiyonu

Bu nöron direk girdinin tersini çıktı olarak vermektedir. Dolayısıyla eşik değerinin 0 olması gerekmektedir.

Bu basit işlemleri gerçekleyebilen MCP nöronlarının yetersiz olduğu işlemler ortaya çıkmaya başladı. Örneğin NOR işlemi MCP nöron ile modellenemedi. İnhibitör girişinin çok fazla kısıtlayıcı olduğu düşünüldü. Boolean tipte olmayan girdiler işleme sokulamadı. Birçok nedenden ötürü yeni bir modellemeye ihtiyaç duyuldu. (2)

2.2 Rosenblatt Perceptron Modellemesi

Amerikalı psikolog Frank Rosenblatt, MCP nöronunun sınırlamalarını aşarak 1958'de klasik perceptron modelini önerdi. Rosenblatt's Perceptron, McCulloch-Pitts nöronunun çoğu sorununun üstesinden gelmek için tasarlanmıştır:

• 1 ve 0'dan farklı girdileri işleyebilir

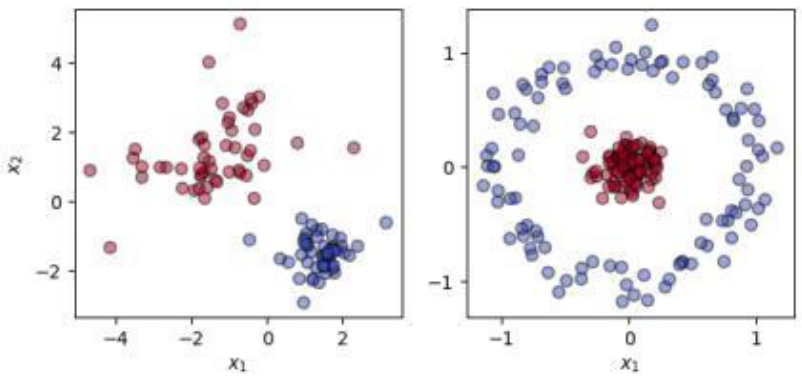
• Her girişe otomatik olarak farklı ağırlıklar atayabilir

• Eşik değer θ otomatik olarak hesaplanır

Perceptron, tek katmanlı bir sinir ağıdır. Bir algılayıcı, ağırlıklı olarak ve bir etkinleştirme işlevi uyguladığımız bir dizi girdi olarak görülebilir. Bu, bir çıktıyla sonuçlanan bir tür ağırlıklı girdi toplamı üretir. Genellikle sınıflandırma problemleri için kullanılır, fakat aynı zamanda regresyon problemleri için de kullanılabilir. Rosenblatt'in Perceptron modeli, yalnızca doğrusal olarak ayrılabilir sınıflar için sınıflandırma görevlerini işleyebilir. (2)

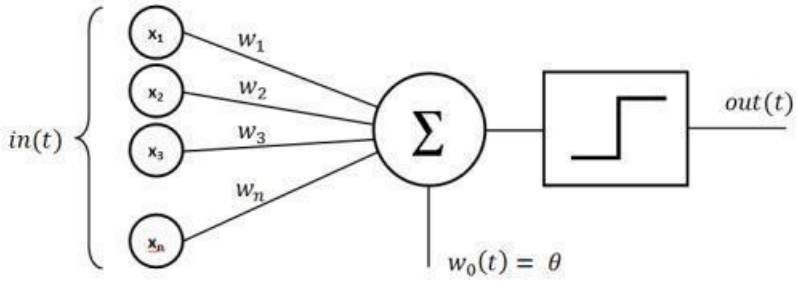
2.2.1 İkili Sınıflandırma (Binary Classification)

Bir kümenin öğelerini önceden belirlenmiş bir kurala göre iki gruba ayırma işlemidir. Örneğin, bir hayvan resminin kediyi mi yoksa köpeği mi tasvir ettiğini sınıflandırma bir ikili sınıflandırma işlemidir. (3)



Şekil 2.2.1 Lineer ve Lineer Olmayan Ayrılabilir Sınıflar

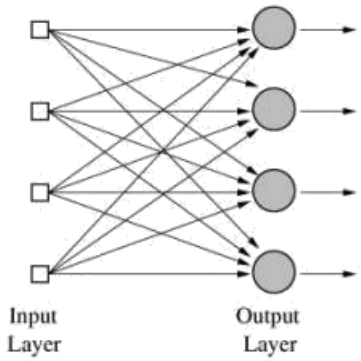
Yukarıdaki örnekte, ikinci şekli birinci şekle dönüştürmek için Özellik Mühendisliği (Feature Engineering) olarak bilinen verilerin ön işleme sokulması işleminin yapılması gerekmektedir. Perceptron'un ikili sınıflandırma problemini nasıl çözdüğünü anlamak için kullandığı yapay sinir ağı modelini inceleyelim. (3)



Şekil 2.2.2 Perceptron Modeli

Bu model, MCP modeline benzemekle beraber arada ciddi farklar bulunmaktadır.

- Nöron, bias olarakta bilinen (şekilde w_0 ile gösterilmiş) sinaptik ağırlıkla ilişkili ekstra bir girdi alır. Bias basitçe aktivasyon eşiğinin tersi olarak nitelenebilir.
- Sinaptik ağırlıklar bazı girdilerin, nöronun çıktısı üzerinde daha fazla etkiye sahip olmasına izin verir.
- Pozitif olma şartı yoktur. Bu sebeple bazı girdilere engelleyici olabilirler.
- MCP nöronunda bulunan inhibitör girişi artık kullanılmamaktadır.



2.2.2 Perceptron'un Matematiksel Gösterimi

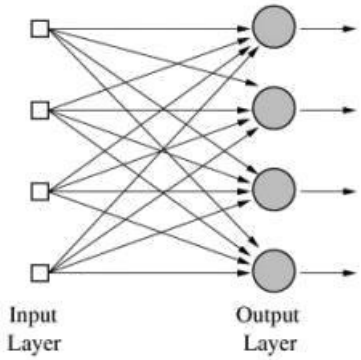
$$\sigma(\mathbf{w}^T\mathbf{x} + b) = \begin{cases} 1 \text{ if } \mathbf{w}^T\mathbf{x} + b \geq 0, \\ 0 \text{ otherwise.} \end{cases}$$

Bu Heaviside fonksiyonu aşağıda sgn(z) fonksiyonu ile tanımlanmıştır.

$$sgn(z) = \begin{cases} +1 \text{ if } z \geq 0, \\ -1 \text{ otherwise.} \end{cases}$$

Formülasyon ne olursa olsun, Perceptron'un lineer sınıflandırma çıktısı her zaman bu şekilde sonuç verecektir.

Rosenblatt, Perceptron ile 20x20 piksel boyutundaki girdilerle basit şekilleri doğru şekilde sınıflamayı makineye öğretebildi. Ancak kompleks problemleri tek bir Perceptron katmanı ile çözemeyeceğini anladı. Çünkü tek katmanlı bir Perceptron yapısı sadece bir çıktı üretmekte idi. Tek katmana farklı Perceptron'lar eklenir ise birden çok çıktıya sahip kompleks problemlerin çözümü gerçekleştirilebilir. (3)



Şekil 2.2.2.1 Çok Katmanlı Perceptron

2.3 Widrow-Hoff ADALINE modeli

Yapay nöronlarla sinir ağlarını tasarlamak da mümkündür. Perceptron'dan farklıdır.

ADALINE, erken bir tek katmanlı yapay sinir ağıdır ve bu ağı uygulayan fiziksel cihazın adıdır. Ağ bellekleri kullanır. 1960 yılında Stanford Üniversitesi'nde Prof.Bernard Widrow ve yüksek lisans öğrencisi Ted Hoff tarafından geliştirilmiştir.

McCulloch-Pitts nöronuna dayanır. Ağırlık, sapma ve toplama fonksiyonundan oluşur. Ayrıca nöronun öğrenme mekanizmasının formal olarak hesap yoluyla hatayı minimize etmeye dayalı olabileceğini düşündüklerinden, eşik aktivasyon fonksiyonuna sahip olmamanın matematiksel olarak daha doğru olduğunu gösterdiler.

Her bir ağırlık katsayısı değiştirildiğinde hatanın ne kadar değiştiğinin ölçümü yani türevi, öğrenme sırasındaki hata miktarını aşağı çekmek ve optimal ağırlık katsayılarını bulmak için kullanılabilir.

Gradyan azalması (Gradient Descent), bir maliyet fonksiyonunu en aza indiren parametrelerinin değerlerini bulmak için kullanılan bir optimizasyon algoritmasıdır.

Burada maliyet fonksiyonu J(w)'dir. J fonksiyonu, aktivasyon fonksiyonu tarafından hesaplanan sonuç ile gerçek sınıf sonuçları arasındaki farkların karelerinin yarısı toplanarak bulunur.

$$w := w + \Delta w$$

where, $\Delta w = -\eta \nabla J(w)$

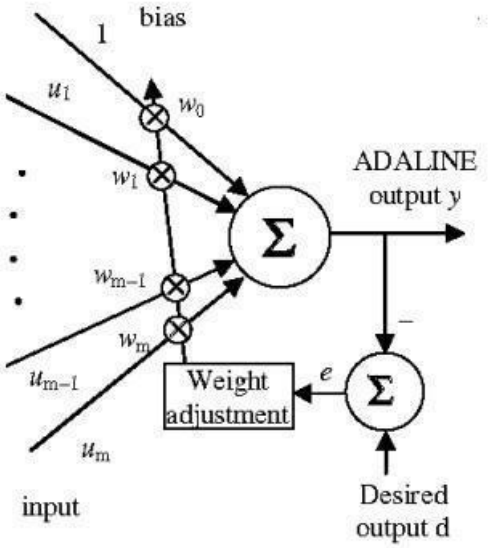
where, η = Learning rate and $\nabla J(w)$ is the gradient of J(w)

$$J(w) = \frac{1}{2} \sum_i \left(y^{(i)} - \phi(z^{(i)})\right)^2$$

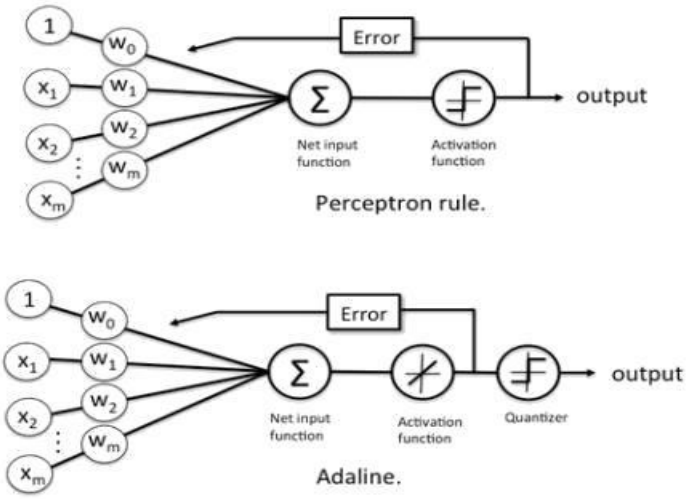
$y^{(i)}$ is the true class label

$\phi(z^{(i)})$ is the net input form the activation function.

Gradyan azalmasını kullanarak, ağırlık katsayılarını yukarıdaki kurala göre güncelleriz. Ayrıca Şekil 2.3.1'de bir ADALINE Sinir Ağı Modeli görülmektedir.



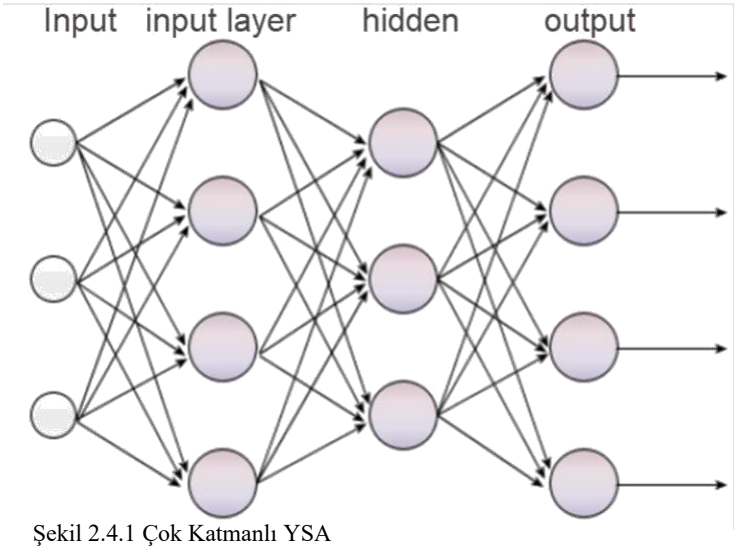
Şekil 2.3.1 ADALINE Modeli



Şekil 2.3.2 ADALINE vs Perceptron

2.4 Çok Katmanlı Yapay Sinir Ağları

Çok katmanlı yapay sinir ağları tek katmanlı perceptron'lara göre birden fazla gizli hesaplama yeteneği bulunun sinir ağından oluşmaktadır. Çok katmanlı bir YSA'da bir adet giriş, en az bir adet gizli ve bir adet çıkış katmanı bulunmalıdır. (2)



Şekil 2.4.1 Çok Katmanlı YSA

Giriş katmanında alınan değerler 1. nörondan ağırlık katsayıları ile gizli katmana iletilerek matematiksel hesaplama işlemi gerçekleştirilir. Sonucunda kullanılan aktivasyon fonksiyonları ile çıkış katmanına iletilirler. İletilen değerlere göre bir sınıflandırma yapılır ve hata oranı tespit edilir. Elde edilen hata oranına göre ağırlık ve giriş değerlerinde güncelleme yapılar aynı işlem tekrar edilir. Elde edilen sonuç en uygun seviyeye geldiğinde modelin eğitimi sonuçlandırılır ve yeni gelecek modelin daha önceden işlem yapmadığı veriler ile tahmin işlemi gerçekleştirilir.

Yapılan araştırmalar, tek katmanlı Perceptron'ların lineer olmayan problemleri çözemediği bununla beraber çok katmanlı Perceptron'larda bulunan gizli katman nöronlarının çıktı katmanında ulaşılan hata değerlerine göre kendilerini tekrar düzenleyerek doğrusal olmayan problemleri de çözebileceğini ortaya çıkarmıştır. (2)

Multilayer Perceptron'lar en küçük kareler yöntemi ile öğrenme yapmaktadırlar. Bu yöntem ise delta öğrenme kuralı olarak bilinir. İki aşamada gerçekleşir:

1-İleri Hesaplama (Feed Forward)

2-Geri Hesaplama (Back Propogation)

Modelde bulunan girdiler ile girdilerin ağırlık katsayılarının hesaplanması ile gizli katmanda kullanılan aktivasyon fonksiyonlarının belirli bir değer üretip çıktı katmanına iletilir. Çıktı katmanındaki sonuç tekrar girdi olarak verilir. Aradaki fark hata oranını vermektedir. Sonraki hesaplamalar geriye doğru yapılarak en optimum değer elde edilmeye çalışılır. (4)

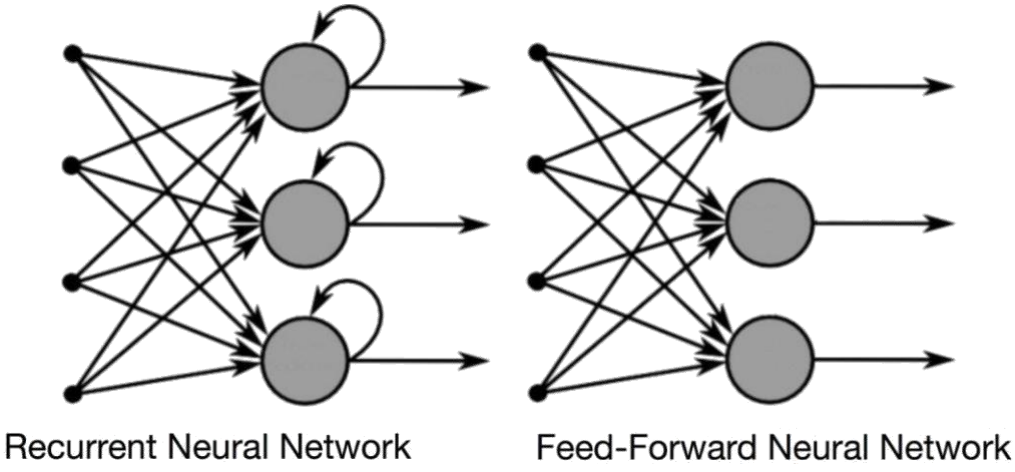
2.4.1 İleri Beslemeli Sinir Ağları

İleri beslemeli sinir ağı, birimler arasındaki bağlantıların bir döngü oluşturmadığı yapay sinir ağı tipidir. Gelişmiş yapay sinir ağının ilk ve en basit çeşididir. Bu ağ çeşidinde veri girdi düğümlerinden gizli düğümler aracılığıyla sadece tek yönde ileri çıkış düğümlerine doğru gider. Ağda döngü yoktur. (4)

2.4.2 Tekrarlayan Sinir Ağları (RNN)

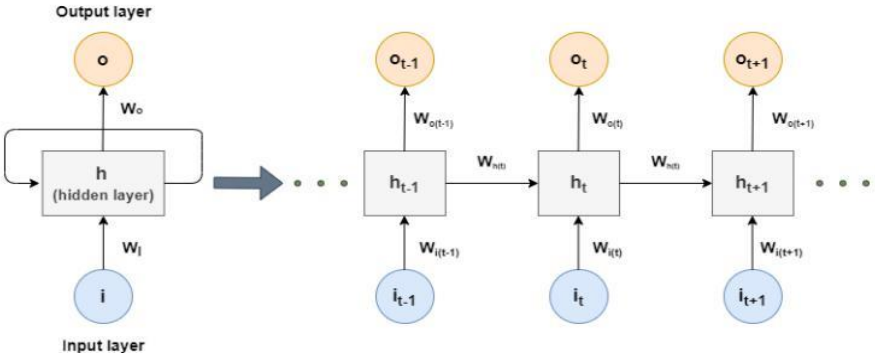
Girdiler üzerinde matematiksel işlemler yaparak çıktıların tahminlerini yapar. Bu işlemler esnasında çıkış katmanında elde edilen hata değerlerini geriye yayılım yaparak optimum seviyeye getirir. Yapılan işlem modelin eğitimi sırasında girdi olarak işlem yapılan sadece tek veri için ileri besleme yaparak sınıflandırma işleminin yapılmasıdır. (5)

Recurrent yapılar bir eğitim aşamasının çıktısını diğer eğitim aşamasının girdisi olarak verdikleri için ileri beslemeli ağlardan farklıdır. Model bir önceki işlemi bellekte tuttuğu ve bu verilere göre eğitim gerçekleştirdiği için recurrent modellerin hafızaları vardır. (5)



Şekil 2.4.2.1 RNN vs FFNN

Şekilde ileri beslemeli sinir ağı yapısı ve tekrarlayan sinir ağı yapısı arasındaki fark açık bir şekilde görülmektedir.



Şekil 2.4.2.2 RNN yapısı

Yukarıdaki şekilde ‘i’ ağın giriş katmanı, ‘h’ gizli katmanı ve ‘o’ çıkış katmanıdır. Bir zaman örneği ‘t’ için iki giriş yani ‘i_t’, mevcut giriş durumu ve h_t-1, önceki durumdan çıkış yeni bir RNN nöronuna beslenir ve yeni bir gizli durum h_t üretilir. Aşağıdaki formül ağın mevcut durumudur:

$$ht = f(ht{-}1, it)$$

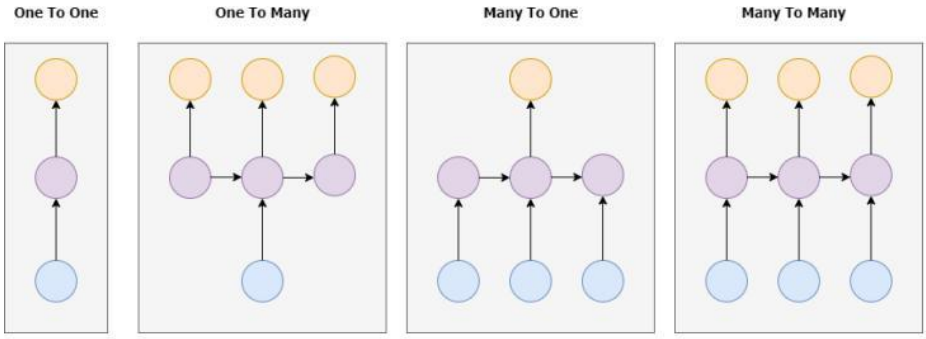
Hiperbolik tanjant aktivasyon fonksiyonunu uygulayarak, denklem, son gizli durumdan ağırlığın h olduğu ve mevcut girdinin ağırlığının olduğu aşağıdaki forma dönüşür.

$$ht = tanh(Wh * ht{-}1 + Wi * it)$$

Çıktı katmanının net formülü burada $_0$ çıktı durumunun ağırlığı ve h_t mevcut gizli durum aşağıdaki hale dönüşür.

$$ot = Wo * h\mathrm{t}$$

RNN, zaman serisi verilerini (sıralı veriler) işlediğinden, dil çevirileri, konuşma ve görüntü tanıma, duygu analizi, doğal dil işleme ve zaman serisi tahmin amaçlarına yönelik uygulamalar için kullanılır. Bir ağdaki giriş ve çıkışların sayısına bağlı olarak, RNN dört tipte sınıflandırılır. (5) (6)



Şekil 2.4.2.3 RNN Tipleri

RNN modellerine ileriye doğru yayılım yapıldıktan sonra geriye doğru yayılma eğitimi verilmelidir. Zaman Yoluyla Geri Yayılım (BPTT), geri yayılımın bir artırımıdır ve bu amaç için kullanılır. Geri yayılım tekniği, çok uzun diziler için işletim süresi ve hesaplama maliyeti açısından maliyetli olabilir. Alternatif olarak, sorunları verimli bir şekilde ele almak için LSTM olarak bilinen başka bir yapay sinir ağı mimarisi kullanılabilir.

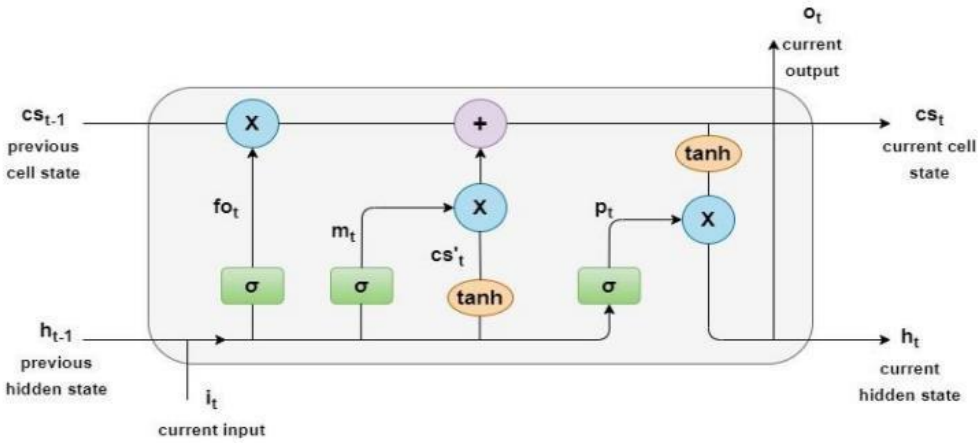
2.4.3 Uzun/Kısa Süreli Bellek (LSTM)

Uzun/kısa süreli bellek, değerleri rastgele aralıklarla hatırlayan bir tekrarlayan sinir ağı (RNN) mimarisidir. Öğrenilen ilerleme kaydedildiğinde saklanan değerler değişmez. RNN nöronlar arasında ileri ve geri bağlantılara izin verir.

Bir LSTM, önemli olaylar arasındaki bilinmeyen boyut ve sürenin gecikmeleri göz önünde alındığında zaman serilerini sınıflandırmak, işlemek ve öngörmek için oldukça uygundur. LSTM, uzun vadeli bağımlılık sorununun önüne geçmek için açık tasarlanmıştır. Uzun süre bilgileri hatırlamak varsayılan davranışlarıdır.

LSTM katmanları, veri akışını kontrol eden üç veya dört kapı içerir. Bu kapılar 0 ile 1 arası değer hesaplamak için lojistik fonksiyon kullanır. Çoğaltma,

bilginin belleğe girip çıkmak için kısmen izin vermek veya reddetmek için bu değerle uygulanır. (5)



Şekil 2.4.3.1 LSTM Yapısı

Değerlerin kararı, 0 ile 1 arasındaki değerleri sıkıştıran Sigmoid gibi, tanh aktivasyon fonksiyonu ise -1 ile 1 arasındaki değerleri sıkıştıran aktivasyon fonksiyonlarının sonuçlarına bağlıdır. Kapılara doğru ilerlerken, LSTM hücresinde üç kapı vardır. Unut kapısı, bilgilerin atılması veya kaydedilmesi gerektiğini belirler. Sigmoid fonksiyonu, mevcut giriş bilgilerinin yanı sıra önceden gizlenmiş durum verilerini de iletir. Değerler unutulacak 0 veya kaydedilecek 1'dir. Unutma kapısı denklemi, 'nin mevcut girdinin ve önceki gizli durumun ağırlıklarını temsil ettiği yerde verilmiştir.

$$fot = \sigma\,(Wfo * [ht{-}1, it] + biasfo)$$

Hücre durumunda tutulması gereken veri miktarına karar vermek için iki bölüme ayrılmıştır. 0 veya 1 döndürmek için ve h −1 birleşimini göndermek için bir sigmoid katman kullanılır. Aşağıdakiler giriş kapısı formülleridir: Önceki hücre durumunu ′ güncel hale getirmek için, ve −1'in çarpımı işe yaramaz verileri unuturken ′ ve 'nin çarpımı yeni verilere ulaşır. Her iki işlemden elde edilen sonuçlar şu şekilde toplanır:

$$mt = \sigma\,(Wm * [ht{-}1, it] + biasm)$$

$$cst' = \sigma\,(Wcs' * [ht{-}1, it] + biascs')$$

Önceki hücre durumunu cs′ güncel hale getirmek için, fot ve cst−1'in çarpımı işe yaramaz verileri unuturken cst' ve mt'nin çarpımı yeni verilere ulaşır. Her iki işlemden elde edilen sonuçlar şu şekilde toplanır: Son olarak, çıktı kapısı, girdi ve hücrenin çıktı vermeye karar verdiği her şeyi gösterir. Girdi, 0 veya 1 değerini döndüren bir sigmoid katman aracılığıyla gönderilir.

Ayrıca, hücre durumu, önemlerine göre -1 ile 1 arasında değişen puanlara ağırlık atanan bir tanh katmanı aracılığıyla işlenir. Gizli katman verileri daha sonra sigmoid katman sonucu ile çarpılır.

$$cst = (fot * cst{-}1) + (mt * cst')$$

Son olarak, çıktı kapısı, girdi ve hücrenin çıktı vermeye karar verdiği her şeyi gösterir. Girdi, 0 veya 1 değerini döndüren bir sigmoid katman aracılığıyla gönderilir. Ayrıca, hücre durumu, önemlerine göre -1 ile 1 arasında değişen puanlara ağırlık atanan bir tanh katmanı aracılığıyla işlenir. Gizli katman verileri daha sonra sigmoid katman sonucu ile çarpılır. (5)

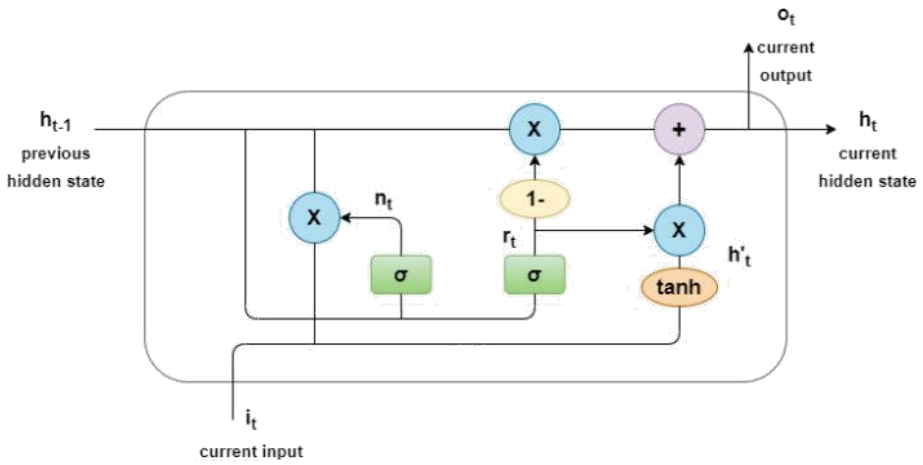
$$pt = \sigma\,(Wp * [ht{-}1,it\,] + biasp)$$

$$ht = pt * tanh(cst)$$

2.4.4 Kapılı Tekrarlayan Sinir Ağı (GRU)

Gated Recurrent Units anlamına gelen GRU, LSTM'nin genişletilmiş bir versiyonudur. Ana işlevi, RNN yapısında karşılaşılan kaybolan gradyanları ve patlayan gradyan sorunlarını ele almayı içerir.

GRU'lar, hücrelerin içindeki ve arasındaki bilgi akışını kontrol etmek için bir geçit mekanizması kullanan LSTM'dir. GRU, unutma, giriş ve çıkış kapısı yerine güncelleme kapısı ve sıfırlama kapısı içerir. Kaybolan gradyan problemi bu kapılar yardımıyla kontrol edilebilir. LSTM'den farklı olarak GRU, hücre durumu yerine gizli bir durumu korur. Sonuç olarak, küçük veri kümeleri daha kolay ve daha hızlı eğitilebilir ve GRU kullanılarak RNN'nin bellek kapasitesi artırılabilir. (5)



Şekil 2.4.4.1 GRU Yapısı

Sıfırlama Kapısı, hücre içinde unutulacak veya yok sayılacak önceki bilgilerin miktarını tanımlar. Bir LSTM tekrarlayan birimin Unut Kapısının nasıl çalıştığına benzer. GRU'nun kısa süreli belleğini yönetir ve gizli durumunu temsil eder. Alakasız verileri kategorize eder ve kullanıcıyı bu verileri kullanmadan devam etmesi konusunda bilgilendirir. Sıfırlama kapısı denklemi aşağıdaki gibidir:

$$nt = \sigma\,(Wn * [ht{-}1, it\,] + biasn)$$

Güncelleme Kapısı, modele, gelecekteki zaman adımlarına ne kadar önceki bilginin verilmesi gerektiğine karar vermede yardımcı olur. Bu oldukça güçlüdür, çünkü model geçmişteki tüm özellikleri kopyalamayı seçebilir ve böylece gradyanların azalması riskini ortadan kaldırabilir. GRU'nun uzun süreli belleğini yönetir. Güncelleme kapısı için denklem aşağıda verilmiştir:

$$rt = \sigma\,(Wr * [ht{-}1,it\,] + biasr)$$

Mevcut gizli durumun h belirlenmesine yardımcı olan geçici bir aday gizli durum h' hesaplanır. Bu denklemin en can alıcı yönü, sıfırlama kapısının değerinin, önceki gizli durumun aday durumu ne kadar etkileyebileceğini düzenlemesidir. değeri 1 ise; O, önceki gizli durum h -1'den gelen tüm bilgilerin dikkate alındığını ima eder. Benzer şekilde, değeri 0 ise, önceki gizli durum bilgisi tamamen yok sayılır.

$$h't = tan\,h\,(W\,[\,nt * ht{-}1, it\,]\,)$$

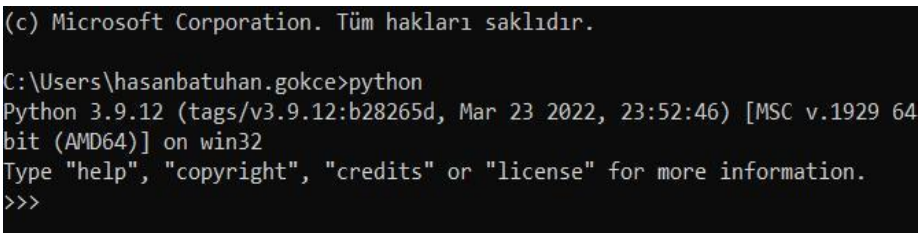
Aday gizli durum değeri kullanılarak, güncelleme geçidinin değeri dikkate alınır ve mevcut gizli durum *h* denklemi aşağıdaki gibi türetilir:

$ht = (1 - rt * ht{-}1) + (rt * ht\,') \quad (5)$

3. KULLANILAN ARAÇLAR

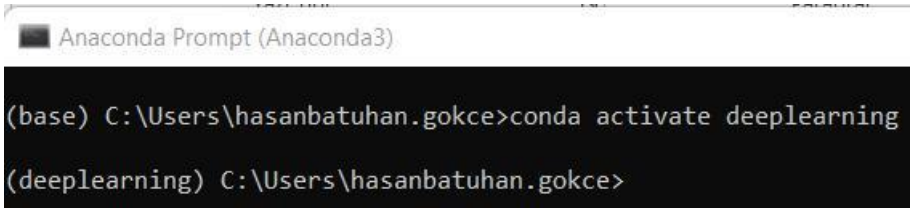
3.1. Sanal Derin Öğrenme Ortamı ve Dil Kurulumu

Geliştirme yapılacak olan dil olarak Python dili seçildi. Bunun için Python 3.9 sürümü kullanıldı.



Şekil 3.1.1 Python Sürümü

Daha sonra veri işlemesi ve algoritmaların gerçeklenmesi için sanal ortam olarak Anaconda yüklendi ve deeplearning isimli sanal ortam oluşturuldu.



Şekil 3.1.2 Anaconda Ortamı

3.2 Kullanılan Veri Seti

Derin öğrenme ile banka kredisi risk analizini yapmak için kaggle.com üzerinden elde edilen, bir Portekiz bankasının kredi kampanyası için telefon görüşmeleri ile oluşturup bunu açık kaynağa çevirdiği bank-client-data veri seti kullanıldı. Bu veri seti 41188 adet müşteri sayısı, bir tanesi krediye uygunluk olmak üzere 21 adet özellik içermektedir.

4. SİSTEMİN GERÇEKLENMESİ

4.1. Verinin Hazırlanması ve Ön İşlemesi

Bu adımda veri ile ilgili çokça düzenlemeler yapılacağı için Pandas, Numpy, Seaborn ve Matplotlib kütüphaneleri kullanılacaktır. Ayrıca analizde kullanılacak diğer teknikler için gerekli diğer kütüphaneler de import edilerek veri analizine başlandı.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import f1_score, accuracy_score, precision_score, recall_score, confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from time import time
from keras.models import Sequential
from keras.layers import Dense, Dropout
import seaborn as sns
import matplotlib.pyplot as plt
```

Şekil 4.1.1 Kütüphanelerin eklenmesi

Daha sonra veri seti okundu, kolon isimleri Türkçe hale getirildi ve bir dataframe içine aktarıldı.

```
df=pd.read_csv('bank-additional-full.csv', sep=';',)
df.columns = ['yas', 'meslek', 'medeni_hal', 'egitim_durumu','temerrud_durumu','ev_durumu','borc_durumu','iletisim_tipi', 'ay', 'gun',
              'son_gorusme','kampanya_musteri_sayisi', 'son_kampanya_gorusme', 'bu_kampanya_son_gorusme','onceki_kampanya_sonuc',
              'is_degisim_orani','tuketici_fiyat_endeksi', 'tuketici_guven_endeksi', 'euribor_oran', 'calisan_sayisi', 'uygun_mu']
df.head()
```

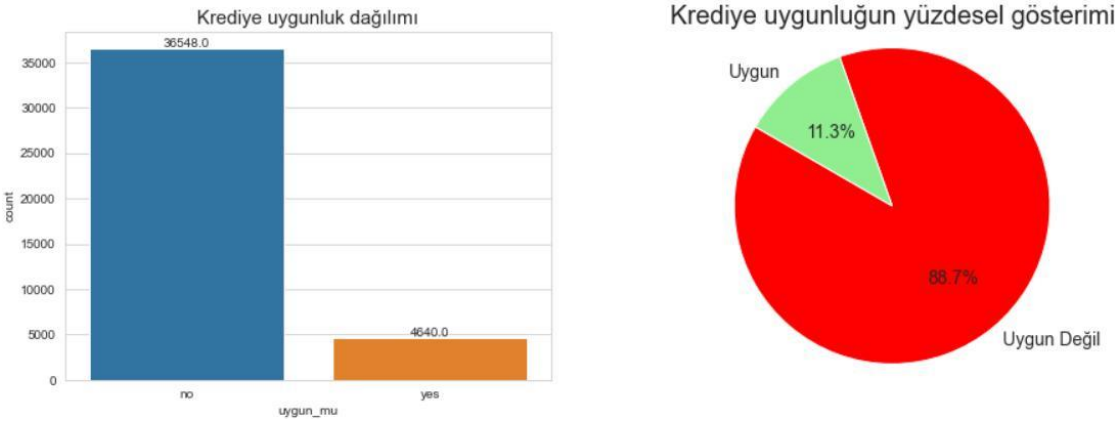
Python

	yas	meslek	medeni_hal	egitim_durumu	temerrud_durumu	ev_durumu	borc_durumu	iletisim_tipi	ay	gun	...	kampanya_musteri_sayisi	son_ka
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	...	1	
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	...	1	
2	37	services	married	high.school	no	yes	no	telephone	may	mon	...	1	
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	...	1	
4	56	services	married	high.school	no	no	yes	telephone	may	mon	...	1	

5 rows × 21 columns

Şekil 4.1.2 Veri Setinin okunması

Ardından veri setinde bulunan müşterilerin krediye uygunluğu nümerik ve yüzdesel olarak grafiğe döküldü.

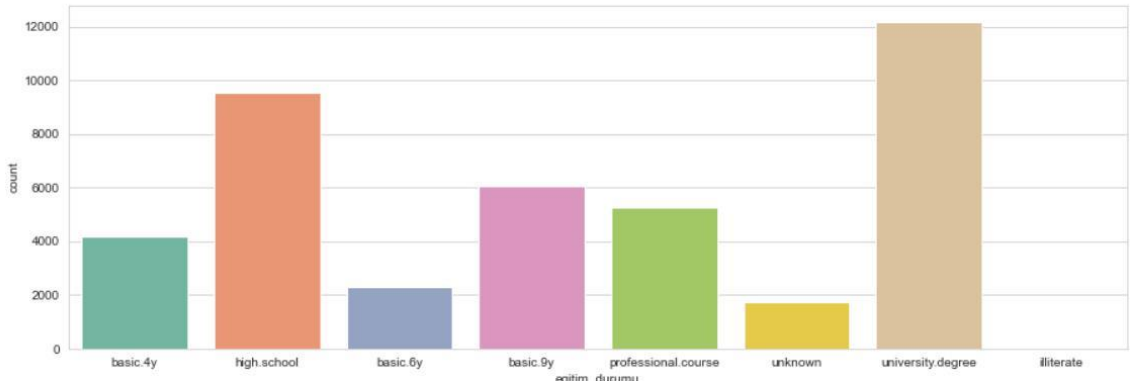


Şekil 4.1.3 Kredi Uygunluk Dağılımı

Sonraki adımda Seaborn kütüphanesinden yararlanılarak müşterilerin eğitim durumu bar grafik olarak bastırıldı.

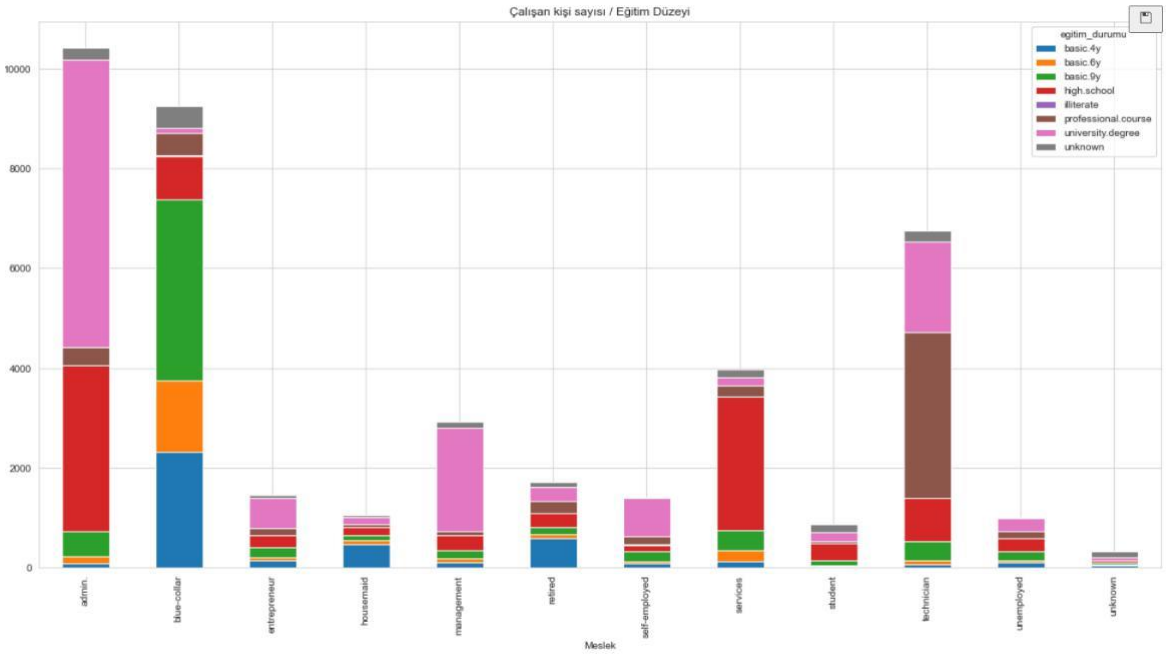
```
plt.figure()
sns.countplot(x="egitim_durumu", data=df, palette="Set2")
```

```
<AxesSubplot:xlabel='egitim_durumu', ylabel='count'>
```



Şekil 4.1.4 Eğitim durumu

Bir sonraki grafikte çalışan kişi sayısı ve eğitim düzeyinin grafiği bastırıldı.

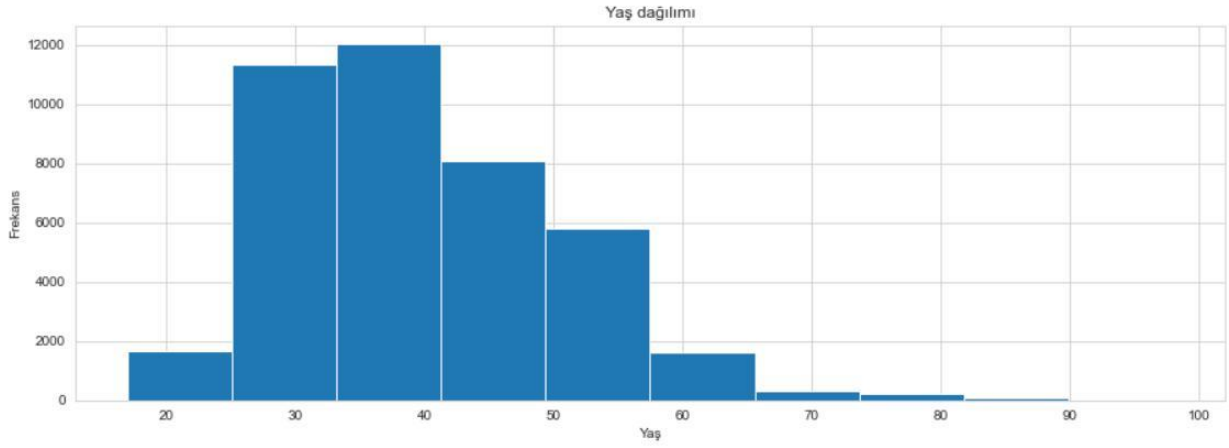


Şekil 4.1.5 Çalışan kişi sayısı / Eğitim durumu

Bir sonraki adımda yaş grubunun histogramik dağılımı bastırıldı.

```
df.yas.hist()
plt.title("Yaş dağılımı")
plt.xlabel("Yaş")
plt.ylabel("Frekans")
```

```
Text(0, 0.5, 'Frekans')
```

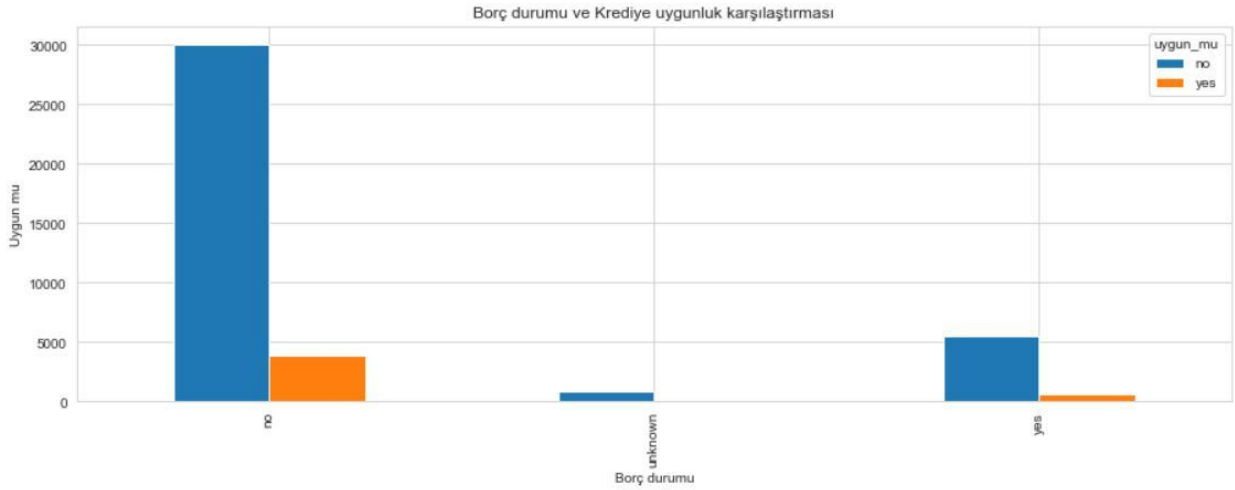


Şekil 4.1.6 Yaş grubu dağılımı

Ardından müşterilerin borç durumu ve krediye uygun olup olmadığı karşılaştırıldı.

```
pd.crosstab(df.borc_durumu, df.uygun_mu).plot(kind='bar')
plt.title("Borç durumu ve Krediye uygunluk karşılaştırması")
plt.xlabel("Borç durumu")
plt.ylabel("Uygun mu")
```

```
Text(0, 0.5, 'Uygun mu')
```

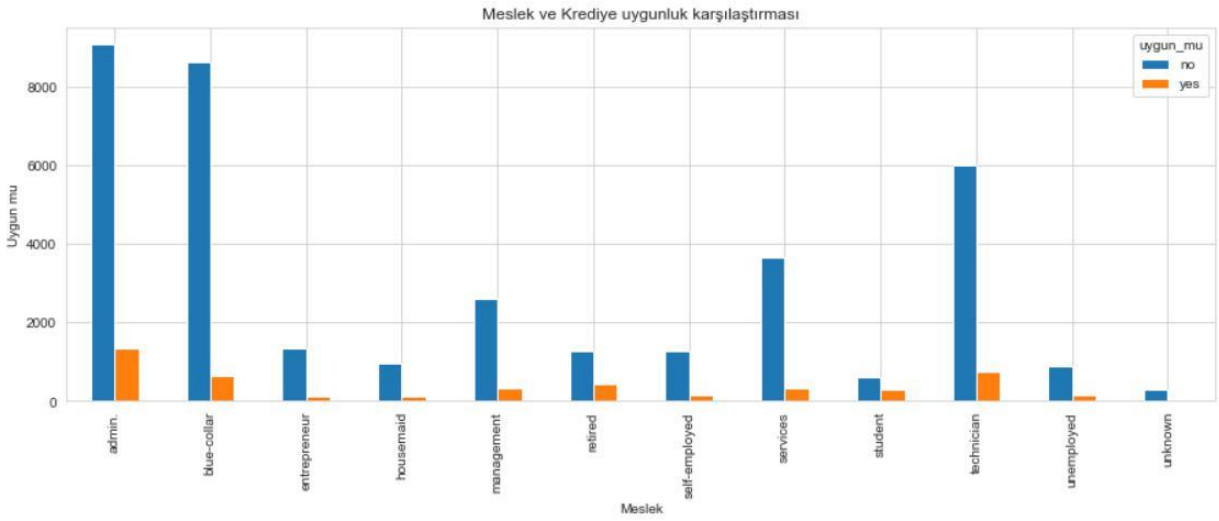


Şekil 4.1.7 Borç/ Kredi uygunluk dağılımı

Son görselleştirme olarak meslek ve krediye uygunluk arasındaki ilişki grafiğe bastırıldı.

```
pd.crosstab(df.meslek, df.uygun_mu).plot(kind='bar')
plt.title("Meslek ve Krediye uygunluk karşılaştırması")
plt.xlabel("Meslek")
plt.ylabel("Uygun mu")
```

```
Text(0, 0.5, 'Uygun mu')
```



Şekil 4.1.8 Meslek/Kredi uygunluk grafiği

Daha sonra describe() metodu ile veri setine ait histogram bilgileri bastırıldı.

```
df.describe()
```

Python

	yas	son_gorusme	kampanya_musteri_sayisi	son_kampanya_gorusme	bu_kampanya_son_gorusme	is_degisim_orani	tuketici_fiyat_endeksi	tuk
count	41188.00000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	
mean	40.02406	258.285010	2.567593	962.475454	0.172963	0.081886	93.575664	
std	10.42125	259.279249	2.770014	186.910907	0.494901	1.570960	0.578840	
min	17.00000	0.000000	1.000000	0.000000	0.000000	-3.400000	92.201000	
25%	32.00000	102.000000	1.000000	999.000000	0.000000	-1.800000	93.075000	
50%	38.00000	180.000000	2.000000	999.000000	0.000000	1.100000	93.749000	
75%	47.00000	319.000000	3.000000	999.000000	0.000000	1.400000	93.994000	
max	98.00000	4918.000000	56.000000	999.000000	7.000000	1.400000	94.767000	

Şekil 4.1.9 describe() metodu

Daha sonra encoding işlemi için python dictionary yapısı kullanılarak, veri setindeki verilere sayısal değerler atandı.

```
dict_onceki_kampanya_sonuc = {
    "onceki_kampanya_sonuc":{
        "nonexistent":0,
        "failure":0,
        "success":1
    }}
dict_uygun_mu = {
    "uygun_mu":{
        "no":0,
        "yes":1
    }}
for i in [dict_                                    mu, dict_meslek]:
    df.replace(
df.head(5)
```

HASAN BATUHAN GOKCE (hasanbatuhan.gokce@ogr.iuc.edu.tr) oturum açtı

	yas	meslek	medeni_hal	egitim_durumu	temerrud_durumu	ev_durumu	borc_durumu	iletisim_tipi	ay	gun	...	kampanya_musteri_sayisi
0	56	1.0	married	1.0	no	no	no	telephone	may	mon	...	1
1	57	3.0	married	1.0	unknown	no	no	telephone	may	mon	...	1
2	37	3.0	married	1.0	no	yes	no	telephone	may	mon	...	1
3	40	2.0	married	1.0	no	no	no	telephone	may	mon	...	1
4	56	3.0	married	1.0	no	no	yes	telephone	may	mon	...	1

5 rows × 21 columns

```
dict_meslek = {
    "meslek":{
        "housemaid":1,
        "unemployed":0,
        "entrepreneur":4,
        "blue-collar":1,
        "services":3,
        "admin.":2,
        "technician":2,
        "retired":1,
        "management":4,
        "self-employed":3,
        "unknown":1,
        "student":0.5
    }}
dict_egitim_durumu = {
    "egitim_durumu":{
        "basic.4y":1,
        "basic.6y":1,
        "basic.9y":1,
        "high.school":1,
        "professional.course":2,
        "university.degree":2,
        "illiterate":0.9,
        "unknown":0.9
    }}
```

Şekil 4.1.10 Dictionary işlemi

Label encoding işlemi yapılarak matris oluşturuldu.

```
lc_X1 = LabelEncoder()
list = ['medeni_hal','temerrud_durumu','ev_durumu','borc_durumu']

for i in list:
    df[i] = lc_X1.fit_transform(df[i])
df_cor = pd.get_dummies(df,
                        columns=['medeni_hal','temerrud_durumu','ev_durumu','borc_durumu'],
                        drop_first=True)
df.head()
```

Python

	yas	meslek	medeni_hal	egitim_durumu	temerrud_durumu	ev_durumu	borc_durumu	iletisim_tipi	ay	gun	...	kampanya_musteri_sayisi	son_kamp
0	56	1.0	1	1.0	0	0	0	telephone	may	mon	...	1	
1	57	3.0	1	1.0	1	0	0	telephone	may	mon	...	1	
2	37	3.0	1	1.0	0	2	0	telephone	may	mon	...	1	
3	40	2.0	1	1.0	0	0	0	telephone	may	mon	...	1	
4	56	3.0	1	1.0	0	0	2	telephone	may	mon	...	1	

5 rows × 21 columns

Şekil 4.1.11 Label Encoding

Ardından veriler arasındaki ilişkiyi sayısal olarak göstermek için korelasyonu alındı.

```
korelasyon = df_cor.corr()
print(korelasyon['uygun_mu'].sort_values(axis=0, ascending=True))
```

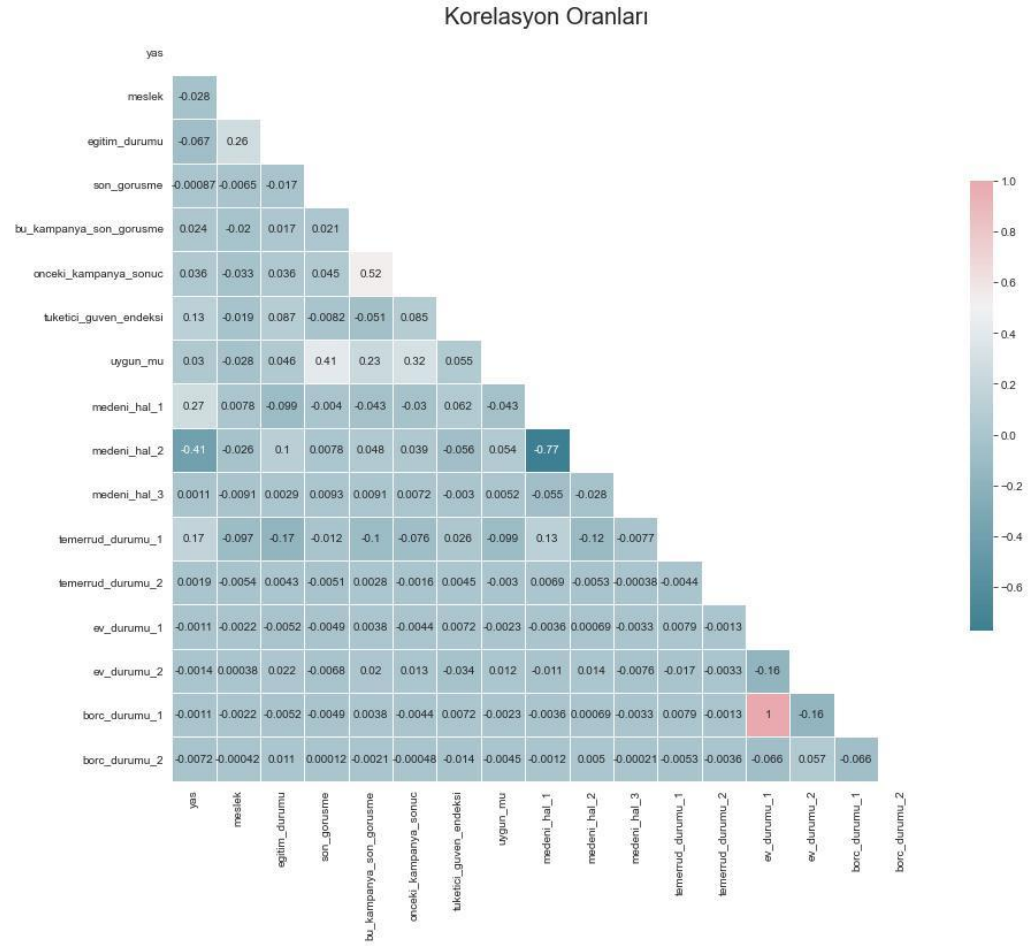
Şekil 4.1.12 Korelasyon

Daha sonra kullanılmayacak olan özellikler drop metodu ile data frame içinden çıkarıldı.

```
df_cor.drop(columns=[
    'calisan_sayisi','son_kampanya_gorusme','euribor_oran','is_degisim_orani',
    'tuketici_fiyat_endeksi','iletisim_tipi','ay','kampanya_musteri_sayisi', 'gun'
    ], axis=0, inplace=True)
```

Şekil 4.1.13 Drop metodu

Ardından Seaborn, MatPlotLib ve NumPy kütüphanelerindeki fonksiyonlar yardımı ile korelasyon matrisi bastırıldı.



Şekil 4.1.14 Korelasyon Matrisi

Sonraki adımda çıktı kolonu drop edilerek bir y değişkenine atandı. Ardından kalan ve ayrılan veriler %80-%20 oranında eğitim ve test datası olarak atandılar. Sonrasında StandartScaler() fonksiyonu kullanılarak algoritmanın kullanılabileceği şekle sokuldu.

```
X = df_cor.drop(columns = 'uygun_mu',axis=1).values
y = df_cor['uygun_mu'].values
```

+ Kodu + Markdown

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=101)
```

```
df_cor['uygun_mu'].value_counts()
```

```
0    36548
1     4640
Name: uygun_mu, dtype: int64
```

```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
```

Şekil 4.1.15 Veri setinin parçalanması

Confusion matrix'i için print_scores() metodu hazırlandı. Bu metodda ayrıca f1 skoru, precision skoru, accuracy skoru ve recall skoru da hesaplanıyor.

```
def print_scores(model, labels, pred):

    accuracy = round(accuracy_score(labels, pred), 3)
    precision = round(precision_score(labels, pred), 3)
    recall = round(recall_score(labels, pred), 3)
    f1 = round(f1_score(labels, pred), 3)

    cm = confusion_matrix(labels, pred)
    df = pd.DataFrame(cm)

    labels = [f"TP: {df[0][0]} ",f"TN: {df[1][1]} ",f"FP: {df[1][0]} ",f"FN: {df[0][1]} "]
    labels = np.asarray(labels).reshape(2,2)

    f, ax = plt.subplots(figsize=(6,5))
    sns.heatmap(cm, annot=labels, fmt='', cmap='Blues', ax=ax)
    print(f"{model}:: Accuracy: {accuracy}, Precision: {precision}, Recall: {recall}, f1_score: {f1}")

def print_cv_result(results):
    print(f"Best Params : {results.best_params_}\n")

    means = results.cv_results_['ortalama_skor']
    stds = results.cv_results_['standart_sapma_skor']

    for mean, std, params in zip(means, stds, results.cv_results_['params']):
        print(f'{round(mean, 3)} (+/-{round(std * 2, 3)}) for {params}')
```

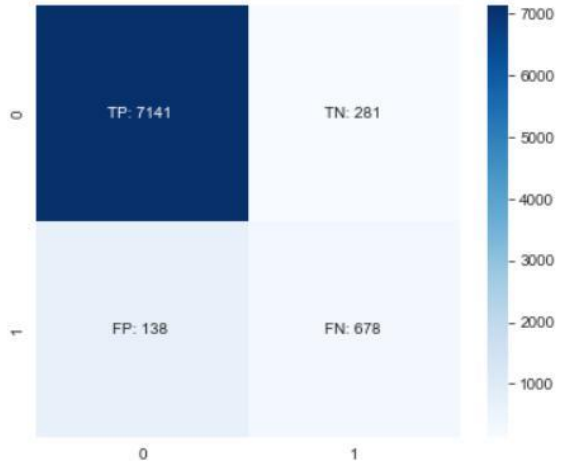
Şekil 4.1.16 Confusion Matrix fonksiyonu

4.2 Algoritmaların Gerçeklenmesi

İlk olarak çıktıların karşılaştırılabilmesi için iki adet makine öğrenmesi algoritması kullanıldı. İlk algoritma olarak Support Vector Machine algoritması çalıştırıldı ve çıktılar alındı.

```
start = time()
svc = SVC(kernel = "rbf", C=10)
svc.fit(X_train,y_train)
y_pred_svm = svc.predict(X_test)
end = time()
print(f"SVM Time:{round((end - start), 5) * 1000}")
print_scores("SVM", y_test, y_pred_svm)
```

```
SVM Time:22351.199999999997
SVM:: Accuracy: 0.901, Precision: 0.671, Recall: 0.293, f1_score: 0.408
```

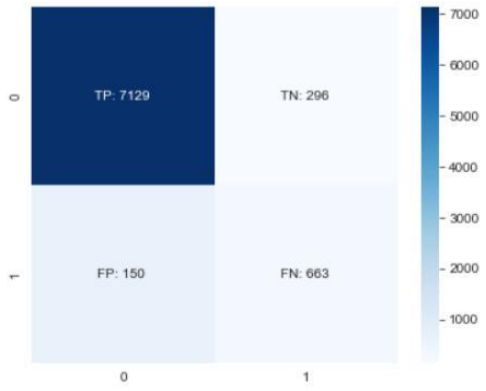


Şekil 4.2.1 SVM algoritması

İkinci algoritma olarak Logistic Regression algoritması gerçeklendi ve çıktı alındı.

```
lgr = LogisticRegression()
lgr.fit(X_train, y_train)
y_pred_lgr = lgr.predict(X_test)
print_scores("LGR", y_test, y_pred_lgr)
```

```
LGR:: Accuracy: 0.901, Precision: 0.664, Recall: 0.309, f1_score: 0.421
```



Şekil 4.2.2 Logistic Regression algoritması

Daha sonra ilk sinir ağımız olan Artificial Neural Network algoritması kullanıldı.

```
classifier = Sequential()
print(df_cor.shape)
```

```
(41188, 17)
```

```
classifier.add(Dense(units=13,
                     activation='relu',
                     kernel_initializer='uniform',
                     input_dim=16))
classifier.add(Dropout(rate=0.1))
classifier.add(Dense(units=13,
                     activation='relu',
                     kernel_initializer='uniform'))
classifier.add(Dropout(rate=0.1))
classifier.add(Dense(units=1,
                     activation='sigmoid',
                     kernel_initializer='uniform'))
classifier.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy'])
start = time()
classifier.fit(x=X_train, y=y_train, batch_size=10,epochs=5)
```

```
Epoch 1/5
3295/3295 [==============================] - 3s 924us/step - loss: 0.2623 - accuracy: 0.8955
Epoch 2/5
3295/3295 [==============================] - 3s 926us/step - loss: 0.2424 - accuracy: 0.9036
Epoch 3/5
3295/3295 [==============================] - 3s 931us/step - loss: 0.2397 - accuracy: 0.9044
Epoch 4/5
3295/3295 [==============================] - 3s 930us/step - loss: 0.2383 - accuracy: 0.9043
Epoch 5/5
3295/3295 [==============================] - 3s 933us/step - loss: 0.2385 - accuracy: 0.9039

<keras.callbacks.History at 0x15b06cf8cd0>
```

Şekil 4.2.3 Artificial Neural Network

Loss oranı ortalama %23 doğruluk oranı da ortalama %90 olarak hesaplandı.

Daha sonra gerçeklenecek olan Recurrent Neural Network algoritmaları, girdi olarak üç boyutlu veri setlerini kullanmaktadır. Ancak var olan veri seti iki boyutlu. Dolayısıyla bir zaman serisi oluşturarak bu veri seti üç boyutlu hale getirilmeye çalışıldı. Bu zaman serisi için veri setinde bulunan ‘son_gorusme’ kolonu kullanıldı.

```
df['son_gorusme']=pd.to_datetime(df['son_gorusme'],dayfirst=True)
df['son_gorusme']=pd.Series(pd.date_range(start='2015-1-1',end='2021-1-1',periods=41188))
df['son_gorusme']
```

```
0       2015-01-01 00:00:00.000000000
1       2015-01-01 01:16:38.266443295
2       2015-01-01 02:33:16.532886590
3       2015-01-01 03:49:54.799329885
4       2015-01-01 05:06:33.065773180
                    ...
41183   2020-12-31 18:53:26.934226848
41184   2020-12-31 20:10:05.200670144
41185   2020-12-31 21:26:43.467113408
41186   2020-12-31 22:43:21.733556704
41187   2021-01-01 00:00:00.000000000
Name: son_gorusme, Length: 41188, dtype: datetime64[ns]
```

Şekil 4.2.3 Zaman Serisi Oluşturulması

Daha sonra veri setindeki özellik sayısı kadar liste oluşturuldu. Ardından veri sayısının en büyük çarpanı olan 28 tane veri her bir diziye sırayla atandı. Her bir atama işlemi kaydırılarak yapıldı ve 12 adet liste veri ile dolduruldu. Daha sonra bu listeler NumPy dizilerine çevrildi.

```
x0=[]
x1=[]
x2=[]
x3=[]
x4=[]
x5=[]
x6=[]
x7=[]
x8=[]
x9=[]
x10=[]
x11=[]
y=[]
```

```
for i in range (0, df.shape[0]-28):
    x0.append(df.iloc[i:i + 28, 0])
    x1.append(df.iloc[i:i + 28, 1])
    x2.append(df.iloc[i:i + 28, 2])
    x3.append(df.iloc[i:i + 28, 3])
    x4.append(df.iloc[i:i + 28, 4])
    x5.append(df.iloc[i:i + 28, 5])
    x6.append(df.iloc[i:i + 28, 6])
    x7.append(df.iloc[i:i + 28, 7])
    x8.append(df.iloc[i:i + 28, 8])
    x9.append(df.iloc[i:i + 28, 9])
    x10.append(df.iloc[i:i + 28, 10])
    x11.append(df.iloc[i:i + 28, 11])
    y.append(df.iloc[i + 28, 11])
```

```
x0,x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,y= np.array(x0),np.array(x1),np.array(x2),np.array(x3),np.array(x4),np.array(x5), np.array(x6),np.array(
```

```
y=np.reshape(y, (len(y),1))
y.shape
```

```
(41160, 1)
```

```
scaler = MinMaxScaler(feature_range=(0,4))
x0 = scaler.fit_transform(x0)
x1 = scaler.fit_transform(x1)
x2 = scaler.fit_transform(x2)
x3 = scaler.fit_transform(x3)
x4 = scaler.fit_transform(x4)
x5 = scaler.fit_transform(x5)
x6 = scaler.fit_transform(x6)
x7 = scaler.fit_transform(x7)
x8 = scaler.fit_transform(x8)
x9 = scaler.fit_transform(x9)
x10 = scaler.fit_transform(x10)
x11 = scaler.fit_transform(x11)
y = scaler.fit_transform(y)
```

```
X = np.stack([x0,x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11], axis=2)
```

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=101)
```

Şekil 4.2.4 3D veri seti oluşturulması aşamaları

İlk RNN yapısı olarak SimpleRNN yapısı gerçeklendi.

```
model=Sequential()
model.add(keras.Input(shape=(28,12)))
model.add(layers.SimpleRNN(128, return_sequences=True, activation='relu'))
model.add(layers.SimpleRNN(128, return_sequences=False, activation='relu'))
model.add(layers.Dense(10))
model.summary()

loss = keras.losses.SparseCategoricalCrossentropy(from_logits=True)
opt=keras.optimizers.Adam(lr=0.001)
metrics = ["accuracy"]

model.compile(loss=loss, optimizer=opt, metrics=metrics)

model.fit(X_train, y_train, batch_size=64, epochs=5, verbose=2)

model.evaluate(X_test, y_test, batch_size=64, verbose=2)
```

Şekil 4.2.5 SimpleRNN yapısı gerçeklenmesi

```
Epoch 1/5

c:\Users\hasanbatuhan.gokce\.conda\envs\deeplearning\lib\site-packages\keras\optimizers\optimizer_v2\adam.py:110: UserWarning: The `lr` argument
is deprecated, use `learning_rate` instead.
  super(Adam, self).__init__(name, **kwargs)

515/515 - 7s - loss: 0.3100 - accuracy: 0.8849 - 7s/epoch - 13ms/step
Epoch 2/5
515/515 - 6s - loss: 0.2892 - accuracy: 0.8871 - 6s/epoch - 11ms/step
Epoch 3/5
515/515 - 6s - loss: 0.2863 - accuracy: 0.8875 - 6s/epoch - 11ms/step
Epoch 4/5
515/515 - 6s - loss: 0.2840 - accuracy: 0.8902 - 6s/epoch - 11ms/step
Epoch 5/5
515/515 - 6s - loss: 0.2820 - accuracy: 0.8894 - 6s/epoch - 11ms/step
129/129 - 1s - loss: 0.2844 - accuracy: 0.8897 - 869ms/epoch - 7ms/step

[0.28436288237571716, 0.8896987438201904]
```

Şekil 4.2.6 SimpleRNN yapısı çıktısı

Daha sonra GRU yapısı gerçeklendi.

```
model=Sequential()
model.add(keras.Input(shape=(28,12)))
model.add(layers.GRU(128, return_sequences=False, activation='relu'))
model.add(layers.Dense(10))
model.summary()

loss = keras.losses.SparseCategoricalCrossentropy(from_logits=True)
opt=keras.optimizers.Adam(lr=0.001)
metrics = ["accuracy"]

model.compile(loss=loss, optimizer=opt, metrics=metrics)

model.fit(X_train, y_train, batch_size=64, epochs=5, verbose=2)

model.evaluate(X_test, y_test, batch_size=64, verbose=2)
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru (GRU)                   (None, 128)               54528

 dense_3 (Dense)             (None, 10)                1290

=================================================================
Total params: 55,818
Trainable params: 55,818
Non-trainable params: 0
```

Şekil 4.2.7 GRU yapısı gerçeklenmesi

```
Epoch 1/5
515/515 - 9s - loss: 0.3231 - accuracy: 0.8839 - 9s/epoch - 17ms/step
Epoch 2/5
515/515 - 8s - loss: 0.2857 - accuracy: 0.8895 - 8s/epoch - 16ms/step
Epoch 3/5
515/515 - 8s - loss: 0.2830 - accuracy: 0.8907 - 8s/epoch - 16ms/step
Epoch 4/5
515/515 - 8s - loss: 0.2827 - accuracy: 0.8897 - 8s/epoch - 16ms/step
Epoch 5/5
515/515 - 8s - loss: 0.2824 - accuracy: 0.8908 - 8s/epoch - 16ms/step
129/129 - 1s - loss: 0.2789 - accuracy: 0.8937 - 1s/epoch - 8ms/step

[0.27891969680786133, 0.8937074542045593]
```

Şekil 4.2.8 GRU yapısı çıktısı

Loss oranı %27, accuracy skoru %89 olarak hesaplandı.

Ardından RNN yapısı gerçeklendi.

```
model = Sequential([
    layers.Flatten(input_shape=(28,12)),
    layers.Dense(128, activation='relu'),
    layers.Dense(10),
    ])
model.summary()

loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True)
opt=keras.optimizers.Adam(lr=0.001)
metrics=["accuracy"]

model.compile(loss=loss, optimizer=opt, metrics=metrics)

model.fit(X_train, y_train, batch_size=64, epochs=100, verbose=2)

model.evaluate(X_test, y_test, batch_size=64,verbose=2)
```

```
Output exceeds the size limit. Open the full output data in a text editor
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 336)               0

 dense_4 (Dense)             (None, 128)               43136

 dense_5 (Dense)             (None, 10)                1290

=================================================================
Total params: 44,426
Trainable params: 44,426
Non-trainable params: 0
```

Şekil 4.2.9 RNN yapısı gerçeklenmesi

```
Epoch 1/100
515/515 - 1s - loss: 0.3277 - accuracy: 0.8815 - 873ms/epoch - 2ms/step
Epoch 2/100
515/515 - 1s - loss: 0.2938 - accuracy: 0.8893 - 729ms/epoch - 1ms/step
Epoch 3/100
515/515 - 1s - loss: 0.2853 - accuracy: 0.8909 - 698ms/epoch - 1ms/step
Epoch 4/100
515/515 - 1s - loss: 0.2815 - accuracy: 0.8926 - 650ms/epoch - 1ms/step
Epoch 5/100
515/515 - 1s - loss: 0.2755 - accuracy: 0.8961 - 654ms/epoch - 1ms/step
...
515/515 - 1s - loss: 3.0983e-04 - accuracy: 1.0000 - 800ms/epoch - 2ms/step
Epoch 100/100
515/515 - 1s - loss: 2.4711e-04 - accuracy: 1.0000 - 654ms/epoch - 1ms/step
129/129 - 0s - loss: 1.7652 - accuracy: 0.8596 - 250ms/epoch - 2ms/step

[1.765163540840149, 0.8595724105834961]
```

Şekil 4.2.10 RNN yapısı çıktısı

Loss oranı %176,51, accuracy skoru %85 olarak hesaplandı

Son algoritma olarak LSTM yapısı gerçeklendi.

```
model = Sequential()
model.add(tf.keras.layers.LSTM(128, input_shape=(X_train.shape[1:]), activation = 'relu', return_sequences = True))
model.add(Dropout(0.2))

model.add(tf.keras.layers.LSTM(128, activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(32, activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(10, activation='softmax'))

opt = tf.keras.optimizers.Adam(lr=1e-3, decay=1e-5)

model.compile(loss='sparse_categorical_crossentropy',
optimizer=opt,
metrics=['accuracy'] )

model.fit(x=X_train, y=y_train, batch_size=10,epochs=50)
```

Şekil 4.2.11 LSTM yapısı gerçeklenmesi

```
Epoch 10/50
3293/3293 [==============================] - 72s 22ms/step - loss: 0.2867 - accuracy: 0.8861
Epoch 11/50
3293/3293 [==============================] - 69s 21ms/step - loss: 0.2881 - accuracy: 0.8874
Epoch 12/50
3293/3293 [==============================] - 68s 21ms/step - loss: 0.2865 - accuracy: 0.8867
Epoch 13/50
...
Epoch 49/50
3293/3293 [==============================] - 72s 22ms/step - loss: 0.3088 - accuracy: 0.8854
Epoch 50/50
3293/3293 [==============================] - 72s 22ms/step - loss: 0.3215 - accuracy: 0.8857

<keras.callbacks.History at 0x1f6864bb9d0>
```

Şekil 4.2.12 LSTM yapısı çıktısı

Loss oranı %32 accuracy skoru %88 olarak hesaplandı.

5.TARTIŞMA VE SONUÇ

Teknolojinin gelişmesi, insan nüfusunun artması gibi birçok nedenden ötürü dünyada bir günde üretilen veri miktarının yaklaşık 2,5 Milyar GB olduğu biliniyor (IBM). Forbes dergisine göre Büyük Veri Analitiği pazarının 2023 yılında 103 milyar dolarlık bir hacme sahip olacağı düşünülüyor. Bu kadar hızlı artan veriler artık miktarından dolayı klasik veri tabanı yönetim sistemleri ile manipüle edilememektedir. Bu verileri kontrol etmek için sürekli yeni teknolojiler geliştirilmektedir. Yapay zekâ, makine öğrenmesi, derin öğrenme gibi teknolojiler aslında çok büyük miktarda veriyi manipüle etmek için üretilmiş teknolojilerdir.

Derin Öğrenme ile yapılan banka kredi risk analizi ile birlikte, beşerî hataların minimum seviyeye indirilmesi sağlanmıştır. Hata miktarının azalması sebebi ile bankalar daha olumlu bütçe planlamalarına sahip olmakta, daha iyi bütçeler ile ülkelere daha fazla vergi ödemekte, devletler daha fazla vergi toplamakta ve işin sonunda bu vergiler halka hizmet olarak geri dönmektedir.

Yapılan analizler sonucunda da kullanılan algoritmaların çıktıları hemen hemen birbirine eşit görülmektedir. Ancak şu da bilinmelidir ki bu tip bir analiz için kullanılmış olan veri seti biraz küçük kalmaktadır. Farklı veri setleri üzerinde bu algoritmalar tekrar test edilebilir.

KAYNAKÇA

1- DERELİOĞLU, Gülnur, 2005, *A MODULAR APPROACH FOR SMEs CREDIT RISK ANALYSIS*, Master of Science, Yıldız Teknik Üniversitesi

2- OĞUZ, Hasan Tahsin, 2004, *CREDIT RISK ANALYSIS USING HIDDEN MARKOV MODEL*, Master of Science, Bilkent Üniversitesi

3- ÇİZER, Emine Bahçe, 2018, *Makine Öğrenmesi Teknikleriyle Kredi Risk Analizi*, Yüksek Lisans, Marmara Üniversitesi

4- CAN, Ömer Yavuz, 2020, *Makine Öğrenmesi Teknikleri Kullanılarak Kredi Risk Analizi*, Yüksek Lisans, İstanbul Aydın Üniversitesi

5- KARAKUŞ, Samet, 2019, *Derin Öğrenme Yöntemleri Kullanılarak Dijital Deliller Üzerinde Adli Bilişim İncelemesi*, Yüksek Lisans, Fırat Üniversitesi

6- ALAGÖZ, Gökhan, 2021, *CREDIT ANALYSIS USING MACHINE LEARNING ALGORITHMS*, Master of Science, Bahçeşehir Üniversitesi

7- ÖZTÜRK, Kadir, 2018, *Yapay Sinir Ağları ve Yapay Zekaya Genel Bir Bakış*, 596690 (dergipark.org.tr)

8- *Tekrarlayan Sinir Ağları*, Tekrarlayan Sinir Ağları (Recurrent Neural Network) – Veri Bilimcisi

9- *Yapay Sinir Ağları*, Yapay Sinir Ağları (Neural Network) – Veri Bilimcisi

10- *İleri Beslemeli Sinir Ağları*, Yapay Sinir Ağları (Neural Network) – Veri Bilimcisi

11- RAZA, Muhammed Raheel, 2022, *DEEP LEARNING BASED SENTIMENT ANALYSIS FOR CLOUD PROVIDER SELECTION*, Master of Science, Fırat Üniversitesi

ÖZGEÇMİŞ

Hasan Batuhan Gökçe. 30/07/1997 tarihinde Giresun'un Merkez ilçesinde dünyaya geldim. İlkokuldan aynı ilde bulunan Barbaros İ.Ö.O'dan 2011 yılında mezun oldum. Daha sonra Keşap Anadolu Öğretmen Lisesi'ni kazanıp 82.94 ortalama ile 2015 yılında tamamladım. Üniversite sınavına ikinci girişimde İstanbul Üniversitesi İnşaat Mühendisliği Bölümünü kazandım. Ancak bu bölümde kendime bir gelecek göremememden dolayı 2018 yılında İstanbul Üniversitesi- Cerrahpaşa Bilgisayar Mühendisliği Bölümüne yatay geçiş yaptım. Aktif bu bölümde okumaktayım.