T.C.

İSTANBUL AYDIN ÜNİVERSİTESİ

LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

CART MAKİNA ÖĞRENME ALGORİTMASINDA İYİLEŞTİRME VE BANKNOT DENETLEME VERİSİNDE UYGULAMA

YÜKSEK LİSANS TEZİ

BATUHAN BİLENLER

Bilgisayar Mühendisliği Ana Bilim Dalı

Bilgisayar Mühendisliği Programı

ŞUBAT, 2020

T.C.

İSTANBUL AYDIN ÜNİVERSİTESİ

LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

CART MAKİNE ÖĞRENME ALGORİTMASINDA İYİLEŞTİRMELER VE BANKNOT DENETLEME VERİSİNDE UYGULANMASI

YÜKSEK LİSANS TEZİ

BATUHAN BİLENLER

(Y1713.010075)

Bilgisayar Mühendisliği Ana Bilim Dalı

Bilgisayar Mühendisliği Programı

Tez Danışmanı : Prof. Dr. Muttalip Kutluk Özgüven

ŞUBAT, 2020

T.C.
İSTANBUL AYDIN ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ MÜDÜRLÜĞÜ

YÜKSEK LİSANS TEZ ONAY FORMU

Bilgisayar Mühendisliği Anabilim Dalı Bilgisayar Mühendisliği Tezli Yüksek Lisans Programı Y1713.010075 numaralı öğrencisi Batuhan BİLENLER'in **"CART MAKİNA ÖĞRENME ALGORİTMASINDA İYİLEŞTİRME VE BANKNOT DENETLEME VERİSİNDE UYGULAMA"** adlı tez çalışması Enstitümüz Yönetim Kurulunun 24.02.2020 tarihli ve 2020/03 sayılı kararıyla oluşturulan jüri tarafından oybirliği/oyçokluğu ile Tezli Yüksek Lisans tezi ..... tarihinde kabul edilmiştir.

	Unvan	Adı Soyadı	Üniversite	İmza
		ASIL ÜYELER		
Danışman	Prof. Dr.	Muttalip Kutluk ÖZGÜVEN	İstanbul Aydın Üniversitesi	
1. Üye	Dr. Öğr. Üyesi	Adem ÖZYAVAŞ	İstanbul Aydın Üniversitesi	
2. Üye	~~Dr. Öğr.Üyesi~~ Doç. Dr.	Ferdi SÖNMEZ	Arel Üniversitesi	
		YEDEK ÜYELER		
1. Üye	Dr. Öğr.Üyesi.	Ahmet GÜRHANLI	İstanbul Aydın Üniversitesi	
2. Üye	Doç. Dr.	Metin ZONTUL	Arel Üniversitesi	

ONAY

Prof. Dr. Ragıp Kutay KARACA
Enstitü Müdürü

YEMİN METNİ

Yüksek Lisans tezi olarak sunduğum "CART MAKİNE ÖĞRENME ALGORİTMASINDA İYİLEŞTİRMELER VE BANKNOT DENETLEME VERİSİNDE UYGULANMASI" adlı çalışmanın, tezin proje safhasından sonuçlanmasına kadarki bütün süreçlerde bilimsel ahlak ve geleneklere aykırı düşecek bir yardıma başvurulmaksızın yazıldığını ve yararlandığım eserlerin Bibliyografya'da gösterilenlerden oluştuğunu, bunlara atıf yapılarak yararlanılmış olduğunu belirtir ve onurumla beyan ederim. (30/01/2020)

Aday / İmza

ÖNSÖZ

CART algoritması ile sınıflandırma işlemi yapılmak istenildiğinde cross validation aşamasında yapılacak algoritmik geliştirmeler ile daha yüksek doğruluk oranı yakalanması hedeflenmiştir.

Bu bilimsel çalışmamın gerçekleştirilmesinde uzun süre boyunca değerli bilgilerini benimle paylaşan, kullandığı her cümlenin hayatıma kattığı önemini asla unutmayacağım saygıdeğer Prof.Dr. Kutluk ÖZGÜVEN hocama, benim için her türlü fedakarlığı yapmış rahmetli babam Demir BİLENLER'e, hayatımın tüm süreçlerinde manevi desteğini esirgemeyen canım annem YETER BİLENLER'e ve ilim yolunda rotamızı çizen, başöğretmen, büyük devlet adamı Mustafa Kemal Atatürk'ün manevi şahsiyetine şükranlarımı sunmayı borç bilirim.

Şubat 2020

Batuhan Bilenler
Bilgisayar Mühendisi

İÇİNDEKİLER

Sayfa

KISALTMALAR

CART : Classification and Regression Tree(Sınıflandırma ve Bağlam Ağacı)

kNN : K Nearest Neighborhood(En Yakın Komşuluk)

CHAID : Chi-squared Automatic Interaction Detector(Ki-Kare Etkileşimli Otonom Tespit Edici)

SVM : Support Vector Machine(Destek Vektör Makinesi)

YSA : Yapay Sinir Ağları

ID3 : Iterative Dichotomiser 3(Tekrarlı (İkiye) Ayırıcı)

RGB : Red Green Blue(Kırmızı Yeşil Mavi)

GPU : Graphics Processing Unit (Grafik İşlem Ünitesi)

CPU : Central Processing Unit(Merkezi İşlem Ünitesi)

ESA : Evrimleştirilmiş Yapay Sinir Ağları

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

CART MAKİNE ÖĞRENME ALGORİTMASINDA İYİLEŞTİRMELER VE BANKNOT DENETLEME VERİSİNDE UYGULANMASI

ÖZET

Bu bilimsel çalışmada, CART algoritması ile sınıflandırma işlemi yapılırken tüm süreçlerinin incelenmesi ve Kaggle platformu üzerinden alınan float türü veriler kullanarak algoritmik iyileştirmeler yapılması amaçlanmaktadır. Cross validation aşamasında eğitim verilerinin daha doğru seçilmesi ile ağaç yapısının daha doğru eğitilmesi beklenmektedir. Test ve eğitim verileri alt kümelere ayrılırken belirli kriterlere göre bu işlemlerin yapılması özellikle eğitim aşamasında sistemin kararlılığını artıracak ve başarı oranını yükseltecektir. Veri seti n alt bölüme ayrılma aşamasında kullanılacak olan verilerin sınıflandırma sonucuna etkisini artırmak amacıyla algoritmik geliştirmeler yapılacaktır.

Anahtar Kelimeler: *CART, Makine Öğrenmesi, Sınıflandırma, Cross Validation*

IMPROVEMENTS IN CART MACHINE LEARNING ALGORITHM AND IMPLEMENTATION IN BANKNOTE AUTHENTICATION DATA

ABSTRACT

In this scientific study, it is aimed to examine all processes while performing classification process with CART algorithm and to make algorithmic improvements by using float type data obtained from Kaggle platform. In cross validation phase, it is expected that the tree structure will be educated more accurately by selecting the training data more accurately. When testing and training data is subdivided, performing these procedures according to certain criteria will increase the stability of the system and increase the success rate, especially during the training phase. Algorithmic improvements will be made in order to increase the effect of the data to be used in the division of data set n sub-section on the classification result.

Keywords : *CART, Machine Learning, Classification, Cross Validation*

1.GİRİŞ

1.1 Tezin Amacı

CART algoritması kullanarak sınıflandırma işlemi yapılırken, Cross Validation aşamasında test ve eğitim verilerinin daha doğru seçilmesi amaçlanmaktadır. Veri setinde yer alan kayıtların, belirli kurallara bağlı olarak seçilmesi ve ayrılan kümelerin eğitim ve test aşamalarında yer almasının sınıflandırma işlemde performans artışı sağlaması hedeflenmektedir. Algoritma için sadece float veri türü kullanılarak matematiksel kurallar işletilecek, ağaç yapısında daha kararlı bir yapı oluşması sağlanacaktır. Tüm veri setinden n adet alt veriye ayrılmış kümelerin çapraz doğruluk kontrolleri yapıldıktan sonra elde edilen sonuçların toplanıp, ortalama değerinin daha yüksek bir sonuç çıkması beklenmektedir.

1.2 Literatür Araştırması

Bu bilimsel çalışmada Karar ağacı kullanarak oluşturulan CART algoritması üzerinde algoritmik geliştirmeler yaparak verileri doğru şekilde sınıflandırmak amaçlanmıştır. Günümüzde verileri sınıflandırma işlemi için birçok teknik kullanılmaktadır. Bu metodlardan en çok kullanılanları aşağıda belirtilmiştir.

- Karar ağaçları
- Lojistik regresyon
- Yapay sinir ağları
- kNN(En yakın komşuluk)
- Bayes
- Bulanık Mantık

CART'ın sahip olduğu algoritma, benzerlik gösteren değişkenlerin aynı ağaç düğümünde toplanmasına dayalı olup, bütün oluşturduğu alt dalları bağımlı değişken olan kök düğüme bağlamayla son bulmaktadır (Teng, J. , Lin, K. , Ho, B. , 2007, 741-

748). Her düğüm sürekli ikiye bölünür. Bölünecek noktaların belirlenmesinde sıklıkla Gini veya Twoing gibi ayırma ölçütleri kullanılmaktadır.

CART algoritmasında bir düğümde belirli bir kriter uygulanarak bölünme işlemi gerçekleştirilir. Her düğümden ancak 2 alt dal ayrılabilir. CART algoritması CHAID algoritmasının daha gelişmiş hali olarak yorumlanabilir. CHAID algoritması çoklu kırılım modelini benimsemektedir. Bu özelliğinden dolayı ticari faaliyetlerde kullanımı daha yaygındır. CART algoritması ise sadece ikili kırılıma izin verdiği için yüksek kestirim yapılması istenilen alanlarda daha çok tercih edilmektedir.

CART algoritması:

- Kirli verilerle çalışmaya izin verebilir.
- Gini index değerine göre kırılımları belirler.
- Sayısal değerleri input olarak çalışmaya izin verir.
- Hem sınıflandırma hem de regresyon tahminine olanak tanır.
- Eksik veriler ile çalışabilme özelliğine sahiptir.
- Ağaç budama işlemini destekler.

CART algoritması, genellikle ağaç oluşturmak için kullanılan bir algoritmadır. Ağaç yapısının daha doğru ve dengeli olması hedeflenmiştir. İkili ağaç yapısını kullanır. Bu yüzden sınıflandırma problemlerinin çözümünde önemli kolaylıklar sağlar. CART, sayısal ve nominal değerler üzerinde çalışabilir (Bozan, 2010). CART algoritması ile sınıflandırma yapılırken 3 önemli aşama bulunmaktadır.

1.Ağaç derinliğini belirleme

2.Ağaç sayısını ve verilerin kaç parçaya ayrılacağını belirleme

3.Ayrılmış test verilerini ağaca uygulama

1.3 Hipotez

Cross Validation aşamasında tüm veri setinden n adet alt veri setine verileri yerleştirirken belirli kriterlere göre ayrılıp, bazı alt kümelerde çok daha yüksek oranla başarı sağlanırken bazı alt kümelerde ise başarı oranında düşme meydana gelecektir. Mevcut algoritma yapısında n adet alt veri seti doğruluk oranları ayrı ayrı hesaplanmaktadır. N adet verinin aritmetik ortalaması bize ortalama doğruluk oranını

vermektedir. Yapılacak olan algoritmik değişiklik ile birlikte performans artışı sağlanan verilerdeki doğruluk oranındaki yükselme, performans kaybı yaratacak alt veri seti doğruluk oranlarından daha yüksek olacağı için ortalama değer daha da yükselecektir. Böylece daha doğru sınıflandırma yapılmış olacaktır.

2. YAPAY ZEKA

İnsan beyninin çalışma prensibini taklit ederek problemlerin çözümü için kolaylıklar yaratan akıllı sistemler, günlük yaşamda birçok kolaylık sağlamaktadır. YSA'lar öğrenme, hafızaya alma ve veriler arasındaki ilişkiyi ortaya çıkarma kapasitesine sahiptir. İnsana özgü davranışları taklit edebilme, duygu analizi, farklı durumlar karşısında karar alabilme gibi özelliklere sahiptir.

Son yıllarda teknolojinin ilerlemesine bağlı olarak, karşılaşılan problemlerin çözümünde daha yüksek performanslı fakat daha az maliyetli yöntemler ön plana çıkmaktadır. Kompleks görülen problemlerin çözümü için kullanılan yapay zeka algoritmaları, büyük bilgi yığınları arasından çözüm yolu bulan modern tekniklerdir. Yapay zeka birçok teknolojiyi ve disiplini içerisinde barındıran bir üst çatıdır. Yapay zeka kavramı bulanık mantık, yapay sinir ağları, derin öğrenme, karar destek makineleri gibi birçok yaklaşımı içermektedir. Bu yaklaşımlar ile yapay zeka sistemleri temel olarak sınıflandırma ve tahmin olmak üzere 2 ana amaç için kullanılır.

2.1 Makine Öğrenmesi

Bir bilgisayar programının, insan etkileşimi olmaksızın kendi kendine sorunları analiz edip, çözüm metodu geliştirmesine makine öğrenmesi denilebilir. Makine öğrenmesi yöntemlerinin toplumun geniş kesimlerini ilgilendiren problemlere çözüm bulması, bilim insanlarının bu konulara olan ilgisini daha da artıracaktır. Makine öğrenmesi yaklaşımlarının kalitesi doğru özelliklerin seçimine bağlıdır. (A. L. Blum and P. Langley, 1997)

Özellikle 1980'li yıllarda satrançtaki başarılarıyla şöhreti dünyaya yayılmış Garri Kasparov'u yenebilecek bir makinenin ortaya çıkarılabilmesi fikri, büyük teknoloji firmalarının dikkatini çekmiştir. Bir zeka oyunu olan ve karmaşık taktiklerle oynanan satranç oyununda, IBM tarafından geliştirilen bilgisayarın Kasparov'u 4-2'lik set galibiyetleriyle yenmiştir ve tüm dünyada yankı uyandırmıştır.

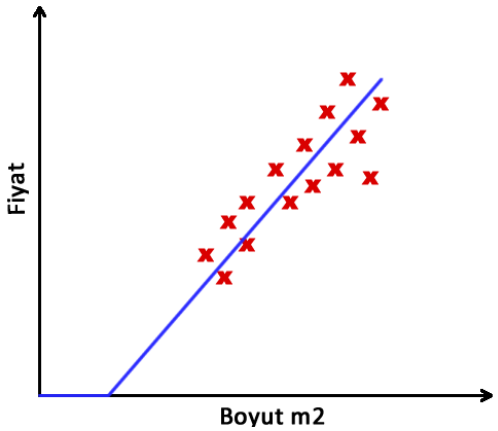
2.1.1 Denetimli öğrenme

Önceden eğitmiş olduğumuz veri setlerini kullanarak öğrenme gerçekleştirilir. Giriş verisi olarak verdiğimiz bilgiyi sistemdeki örneklerini baz alarak tahmin etmeye çalışır. Karar ağaçları, destek vektör makineleri, en yakın komşu algoritmaları (kNN) bu tip algoritmalara örnektir. Denetimli Öğrenme için genellikle sınıflandırma ve regresyon metodları kullanılmaktadır.

Sınıflandırma metodunda, daha önceden çeşitli özelliklerine göre ayrılmış sınıflardan hangisinde bulunacağının kararı verilir. Bu algoritmalar dost veya düşman kararı, kredi verilebilir/verilemez müşteri, iyi/kötü huylu tümör gibi sınıflara yerleştirir. Gelen herhangi bir mailin spam olup olmadığı yönündeki değerlendirmede bulunması bir sınıflandırma sınıflandırma metodudur. Son yıllarda oldukça popüler uygulamalardan olan yüz ve ses tanıma uygulamaları denetimli öğrenme metodlarıyla geliştirilmiş otonom sistemlerdir.

Regresyon teknikleri sürekli değişim gerektiren durumlar için dinamik çözüm önerisi sunar. Örneğin kısıtlı bir bölge için yıllarca yağış miktarlarının tutulduğu bir sistemde, ilerde yağabilecek yağış miktarı üzerinde tahminde bulunması, bir firmanın yeni kampanyasının satış miktarı üzerindeki etkisinin tahmin vermesi birer regresyon tekniğidir.

Belirli bir bölgede evlerin metrekare/fiyat bilgilerini gösteren grafik Şekil 1.'de gösterilmektedir. Metrekare bilgisi bilinen evin fiyatını tahminlemede kullanılan model regresyon tekniğidir.

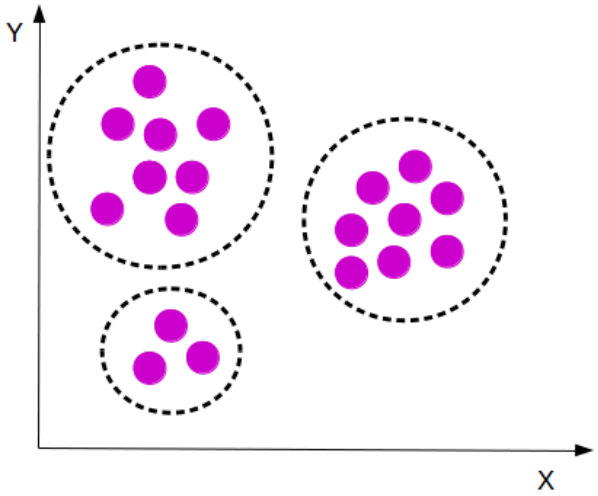


Şekil 1. Konut fiyatı belirlemede fiyat/metrekare değişimini gösteren grafik

2.1.2 Denetimsiz öğrenme

Denetimsiz öğrenme, bilgilerin içindeki ortak alanlara dayanarak gruplaştırma tabanlı veri analizine olanak veren bir yöntemdir. Bu metotta giriş olarak verinin karşısında bir çıkış bilgisi yoktur. Başlangıçta veriler hakkında bilgi verilmediği için kesin sonuç çıkarılması mümkün değildir. Verilerin çeşitli özellikleri ile ilişkilendirilerek kümelenmesi mantığına dayanır.

Şekil 2.'de benzer özellik gösteren verilerin kendi aralarında guruplaşmasına dayalı öğrenme metodu gösterilmektedir. 'X' ve 'Y' özellikleri birbirine yakın değerleri alan bilgiler kendi aralarında birliktelik kurması mantığına dayanmaktadır.



Şekil 2. Denetimsiz öğrenme metodunda kümelenme kavramının gösterimi

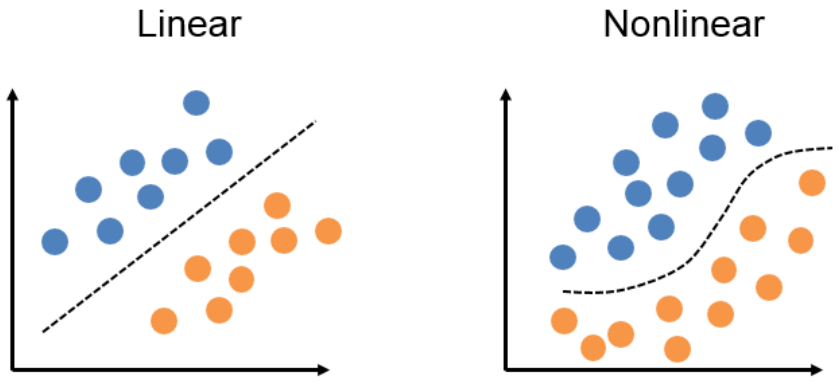
3. SINIFLANDIRMA

Sınıfı bilinmeyen, ham, işlenmemiş verilerin (unseen cases) daha önceden belirlenmiş kategorilerden birine yerleştirilmesi işlemine sınıflandırma denir. Sınıflama tahmin edici bir model olup, havanın bir sonraki gün nasıl olacağı veya bir kutuda kaç tane mavi top olduğunun tahmin edilmesi bir sınıflama işlemidir (Silahtaroğlu, G. 2008 33, 45- 47, 58). Sınıflandırma yapabilmek için mutlaka bir sınıflandırma modeline ihtiyaç duyulmaktadır. Bu model sınıflandırma yapabilen bir fonksiyon olarak görev yapar. Tahmin ve hedef değişkenleri arasındaki bağ, veri seti üzerinde deneyerek bulunur. Veri seti bu aşamada eğitim ve test verisi olmak üzere ikiye ayrılır. Eğer bir değişkenin çoğu değeri eksik ise o değişken, veri setinden çıkartılmalıdır.(Naive Bayes. ,(2009)) Eğitim olarak ayrılan verilerle model eğitilip, test verileriyle ise doğruluk kontrolü yapılmaktadır. Bu doğruluk oranı modelin başarısını ortaya koymaktadır. Kullanılacak tüm verinin yaklaşık %80'i eğitim verisi olarak, %20'si ise test verisi için ayrılmaktadır. Bu ayrım oranı, %70-%30 olarak da seçilebilir. Modelin sağlamasını yapabilmek için genellikle k-folds cross validation işlemleri gerçekleştirilir. Örneğin %20'lik bir veri test için ayrılmış ise, 5 kere farklı %20'lik alt veri setleri sırayla test verisi olur. Böylece birbirini çapraz olarak test etmiş olurlar. Bu işlemler sonucunda 5 tane ayrı oran elde edilir. Doğruluk oranını hesaplamak için ise aritmetik ortalaması alınır. Bu elde edilen ortalama sonuç modelin doğruluk oranı olarak kabul edilmektedir.

Sınıflandırma yöntemleri kullanarak son yıllarda yapılan önemli çalışmalar bulunmaktadır.

- Sağlık alanında, hastaların kan değerleri ve doku örnekleri kullanılarak tümörlü hücrelerin erken/zamanında tespiti yapılmaktadır. Önceden kanserli hastalardan alınmış numunelerle sınıflandırma modeli eğitilerek, yeni gelen bir hastanın doğru şekilde sınıflandırılması için çalışmalar yapılmaktadır.

- Bankacılık ve sigortacılık alanlarında anomali tespiti ile uygun olmayan işlemlerin önüne geçilmesi, beklenmeyen bir işlem yapıldığında erken önlem alınabilmesine izin vermektedir.
- Kimlik doğrulama işlemleri önemli olan güvenlik alanında, ses ve yüz sınıflandırma işlemleri hassas ayrımlar yapabilmeye olanak vermektedir.



Şekil 3. Lineer ve lineer olmayan sınıflandırıcı modelleri

Şekil 3'te 2 farklı sınıfın doğrusal ve doğrusal olmayan yöntemlerle birbirinden ayrılması modellenmiştir.

Sınıflandırma işlemine başlanmadan önce verinin detaylı incelenmesi sınıflandırıcının başarısını doğrudan etkileyecektir. Ham veri sınıflandırılmadan önce ön işleme tabi tutulması gerekmektedir:

- Veri dönüşümüyle, sürekli nitelik değerlerinin birbirinden ayrılması ve birbirinden ayırıcı özelliklerin belirlenmesi.
- Değerlerin normalize edilmesi ([0,1], [-1,+1])
- Kullanılmayacak olan bilgiler varsa o verilerin temizlenmesi(Data Cleaning)
- Gürültülü verinin dönüştürülmesi ve kaldırılması gibi işlemlerin yapılması gerekmektedir.

Sınıflandırma işlemi temel olarak 3 ana aşamadan oluşur.

- Modelin inşa edilmesi

- Modelin test edilip, değerlendirilmesi
- Modelin faaliyete alınması

Bir sınıflandırma modelinin başarısını değerlendirmenin birden fazla parametresi bulunmaktadır.

- Sınıflandırma işlemi için gerekli süre
- Sınıflandırma işlemi yapılırken gürültü veriler karşısında doğru sonuçlar verebilme
- Veri miktarı arttığında doğru sonuçlar elde edilebilme
- Birbiriyle ters düşen sonuçların meydana gelmemesi gibi özellikleri sağlamalıdır.

3.1 Sınıflandırma Yöntemleri

Günümüzde verileri sınıflandırmak için birçok metod kullanılmaktadır.

3.1.1 Karar ağaçları

Karar ağaçları, ağaç yapısı formunda sınıflandırma ya da regresyon modeli oluşturur. Karar ağaçları, basit karar verme adımları uygulanarak, çok sayıda kayıt içeren bir veri kümesini çok küçük kayıt gruplarına bölmek için kullanılan bir yapıdır (Berry, M. J.,Linoff, G. S., 2004). Karar ağaçları oluşturulurken kullanılan algoritmanın ne olduğu önemli bir husustur. Burada, amaç-hedef değişkene ilişkin mümkün olabilen en homojen veri alt gruplarını üretmektir.(Kurt, I. Ture, M., Kurum, A. T., 2008, 366–374). İlişkili karar ağaçları kademeli olarak oluşturulur, tüm veri kümeleri daha küçük veri kümelerine ayrılır. Karar Ağacı üzerinde düğümleri birbirine bağlayan çizgilere dal adı verilir (Gordon ve Pressman 1983: 110). Ağaç düğüm ve yaprakları olarak giderek büyür. Bölümlendirme işlemi çocuk düğümlere (child node) veya alt düğümlerin her birine ardışık olarak uygulanır (Hand, Manila ve Smyth, 2001: 147). Özellikleri belirten karar düğümleri birden fazla dala sahiptir. Bir ağaçta en üstteki karar düğümü yani kök düğümü o ağacın en belirleyici özelliğini temsil eder. En belirleyici özelliğin belirlenmesi için birçok metod bulunmaktadır. Karar ağaçları farklı veri tiplerinin üstesinden gelebilecek özelliğe sahiptir. Herhangi bir karar problemi için kullanılabilen Karar Ağacı tekniği özellikle birden fazla kararın ardışık

olarak verilmesini gerektiren karar problemlerinin gösteriminde çok kullanışlıdır (Albright, Winston ve Zappe 2006: 311).

Karar ağacı yöntemi için aşağıdaki aşamaların yapılması gerekmektedir.

1. Problemin tanımlanması,

2. Ağaç yapısının tasarlanması ve oluşturulması,

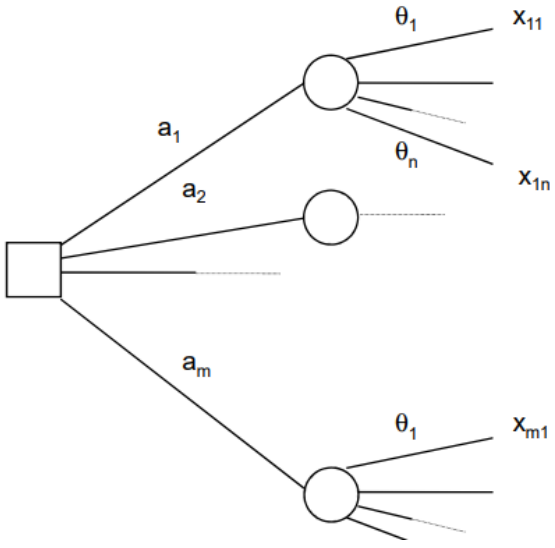
3. Bağımsız olayların oluşma ihtimallerinin belirlenmesi,

4. Elde edilecek kazancın hesaplanması,

5. Kazancın belirlenen karar noktalarına yerleştirmede göz önünde bulundurularak atamaların yapılması,

6. Tahmin işleminin yapılması

	OLAYLAR			
SEÇENEKLER	θ_1	θ_2	...	θ_n
a_1	x_{11}	x_{12}	...	x_{1n}
a_2	x_{21}	x_{22}	...	x_{2n}
.	.	.	...	.
a_m	x_{m1}	x_{m2}	...	x_{mn}



Şekil 4. Karar Ağacı modellemesi

Yukarıda Şekil 4'te olaylar ve seçenekler karşısında hangi durumların oluşacağı bilgisinin modellenerek karar ağacına yerleştirilmesi gösterilmiştir. Karar ağacı

metodu öğreterek öğrenme metodları arasında çok sık kullanılan bir yöntemdir. Sözde kod olarak algoritmayı ifade etmek gerekirse aşağıdaki aşamaları takip etmek gerekir.

Step 1: M öğrenme veri setini oluştur.

Step 2: M kümesindeki örnekleri birbirinden ayıran en önemli özelliği bul.

Step 3: Belirlenen özellik ile ağacın bir düğümünü oluştur ve bu düğümden çocuk düğümleri ve/veya yaprakları oluştur. Alt düğümlere ait alt veri kümesinin örneklerini belirle.

Step 4: Step 3'teki adımda oluşturulmuş her alt veri kümesi için:

◦ Örneklerin eğer tümü aynı sınıfa aitse gelmiş ise,

◦ Örnekleri bölecek başka herhangi bir özellik bulunmuyorsa,

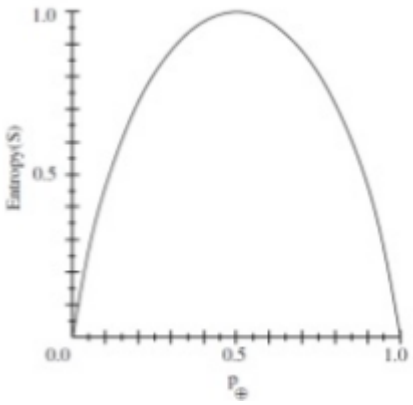
◦ Gerideki niteliklerin değerini barındıran elimizde başka örnek kalmamış ise işlemi sonlandır. Diğer kalan tüm durumlar için, Step 2 adımına dön ve döngüye devam et.

Ele alınan bağımlı değişken kategorik ise yöntem sınıflama ağaçları (Classification Tree), sürekli ise regresyon ağaçları (Regression Tree) olarak adlandırılmaktadır (Deconinck, E., Hancock, T., 2005, 91–103). Karar ağaçlarında en ayırt edici özelliği belirlemek ağacın doğruluk oranını etkileyen en önemli özelliklerden biridir. Ayırt edici özelliği belirlemek için bilgi kazancı hesaplaması yapılır. Bilgi kazancını hesaplamak için entropy metodu genellikle kullanılmaktadır. Entropy metodu, rastgeleliği ve kararlı yapıları belirlemeyi sağlayan bir formülasyondur. Aşağıdaki formülasyonda p(x) bir sınıfın gerçekleşme oranını, H ise entropi değerini göstermektedir.

$$H = - \Sigma\, p(x) \log p(x)$$

Örneğin, M veri seti için 9 adet örnek Class1, 5 adet örnek Class2 türüne ait olduğunu kabul edilirse, entropi değeri aşağıdaki gibi hesaplanabilir. 2 sınıfın entropi değeri toplamı 1 olmak zorundadır.

$$H(c1)= - (9/14)\ \mathrm{Log2}\ (9/14) - (5/14)\ \mathrm{Log2}\ (5/14) = 0.940$$



Şekil 5. Entropi skala grafiğinin gösterimi

Şekil 5'te görüleceği gibi eğer örnekler aynı sınıfa ait ise, entropi değeri 0'dır ve en kararlı olduğu durum bu durumdur. Karar ağaçlarında entropi değerini en az olması verileri ayırırken en doğru seçeneği belirlemede anahtar rol oynar. Entropiler hesaplandıktan sonra, en iyi bölünmeyi saptamak için bilgi kazancını(Information Gain) kullanılır. Bilgi kazancının bulunması ise aşağıdaki formülasyona göre yapılmaktadır.

$$Gain(S, D) = H(S) - \sum_{V \in D} \frac{|V|}{|S|} H(V)$$

Formüldeki S gerçek veri setini göstermektedir. D ise kümenin ayrılmış bir blokunu temsil etmektedir. V tüm verilerin alt kümesidir. V veri setlerinin tümü bağımsızdır. Bölünmeden önceki veri setinin entropisi ile ayrı ayrı her özelliğin entropi değeri arasındaki fark, bilgi kazancını oluşturmaktadır. Günümüzde sınıflandırma problemlerinin çözümü için en çok kullanılan karar ağacı algoritması ID3'tür. C4.5, CHART, CHAID algoritmaları da kullanılan bazı yöntemlerdendir. Entropi, belirsizliğin ölçüsünü verdiği için, ağaç tabanlı sınıflandırıcılarda, dallara ayırıcı özelliği belirlemede kullanılır. (Silahtaroğlu, G. ,(2009))

Karar ağacını oluştururken en önemli özelliği belirlemede en çok kullanılan yöntemlerden biri de Gini Index metodudur. T veri seti üzerinde n tane sınıf olduğunu kabul edersek,

$$Gini(T) = 1 - \sum_{j=1}^{n} (p_j)^2$$

T veri seti için 2 adet örnek Class1, 4 adet örnek Class2 türüne ait olduğunu kabul edilirse, aşağıdaki gibi gini index hesaplanmaktadır.

P(C1) = 2/6 P(C2) = 4/6 Gini = 1 – (2/6)2 – (4/6)2 = 0.444

Bir ağacını oluşturmak için çeşitli parametreler bulunmaktadır. Aşağıdaki temel yaklaşımlar Çizelge 1 üzerinde gösterilmektedir.

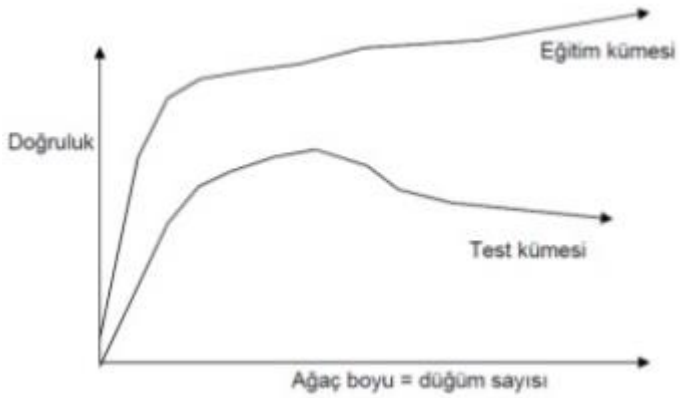
Çizelge 1 Ağaç oluştururken kullanılacak temel kriterlerin gösterimi

Bölme Kriteri	Gini Index, bilgi kazancı (entropi)
Dallanma Ölçütü	Binary dallanma (gini index), çoklu dallanma (bilgi kazancı)
Durma Kararı	Dallanmaya devam etme/etmeme kararının ne zaman alınacağının belirlenmesi
Etiketleme Kararı	Yaprak, düğüm en çok örneği olan sınıfa bakılarak benzer özellikle etiketleniyor.

Karar ağacı oluştururken öğrenme kümesindeki örneklerin azlığı veya gürültülü veri ile çalışılmış olmasından dolayı aşırı öğrenme(overfitting) meydana gelebilir. Overfitting'in oluşmasını önleyen temel 2 yaklaşım bulunmaktadır.

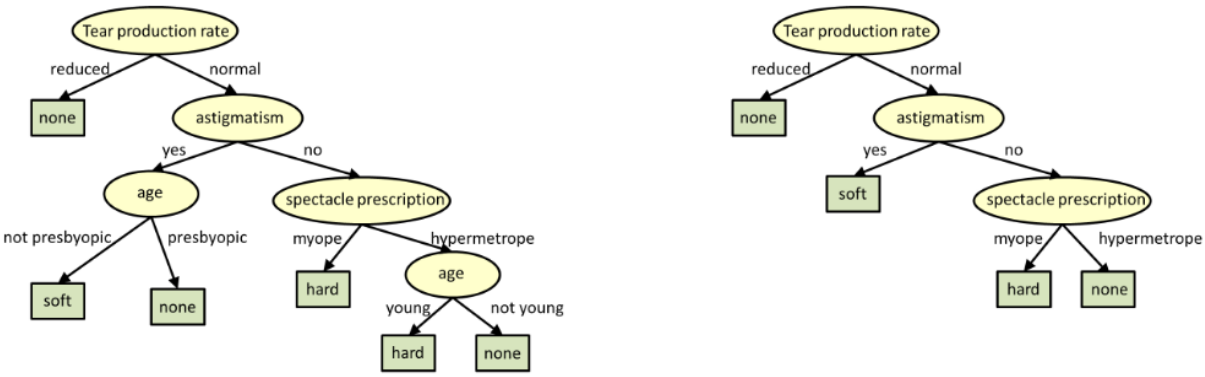
1. Eşik değeri belirleyerek, süreci erken sona erdirmek.

2. Karar ağacını oluştuktan sonra ağacı budama işlemiyle ağacı küçültmektir.



Şekil 6. Karar ağacında overffiting işleminin gösterimi

Şekil 6'da görüldüğü üzere gürültülü veri ile çalışıldığı zaman her yaprak saflaşana kadar ayrılmasına izin verilmesi, ağacın veriyi ezberlemesine yol açabilir. Yani test verisi ile ölçülen doğruluk oranı önce artar; fakat belirli aşamadan sonra oran azalmaya başlar. Genellikle bunun önüne geçmek için kullanılan yöntem ağaç budama Şekil 7'de gösterilmektedir.



Şekil 7. Ağaç budama işleminin gösterimi

Karar ağacı algoritmasını kullanmanın avantajları:

- Kolay anlaşılabilir kurallardan oluşur.

- Kolay oluşturulabilmesi ve yorumlanabilir.

- Yapısı sürekli ve farklı özellikler ile kullanılabilmeye müsaittir.

Karar ağacı algoritmasının dezavantajları:

- Öğrenme örneği az olan yapılarda yüksek başarı elde edilemeyebilir.

- Veri sayısının az ve sınıf sayısının çok olduğu durumlarda performansı düşüktür.

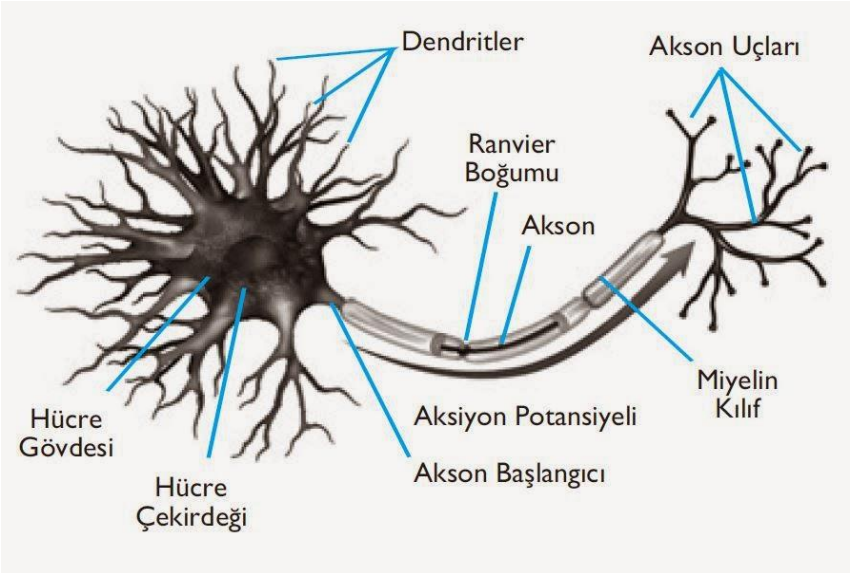
- Ağaç oluşturma ve budama için karmaşıklık fazladır.

3.1.2 Yapay sinir ağları(YSA)

Yapay sinir ağları, insan vücudundaki nöron yapısını taklit ederek oluşturulmuş bir modeldir. Farklı disiplinlerdeki problemlerin çözümlenmesinde kullanılabilen YSA için farklı ağ mimarileri ve farklı eğitim algoritmaları geliştirilmiştir. (Güler İ, Übeyli ED 2006) Bu dönemde, YSA yeniden, destek vektör makinelerine rakip olmaya başlamıştır. (J. Schmidhuber, 2015)

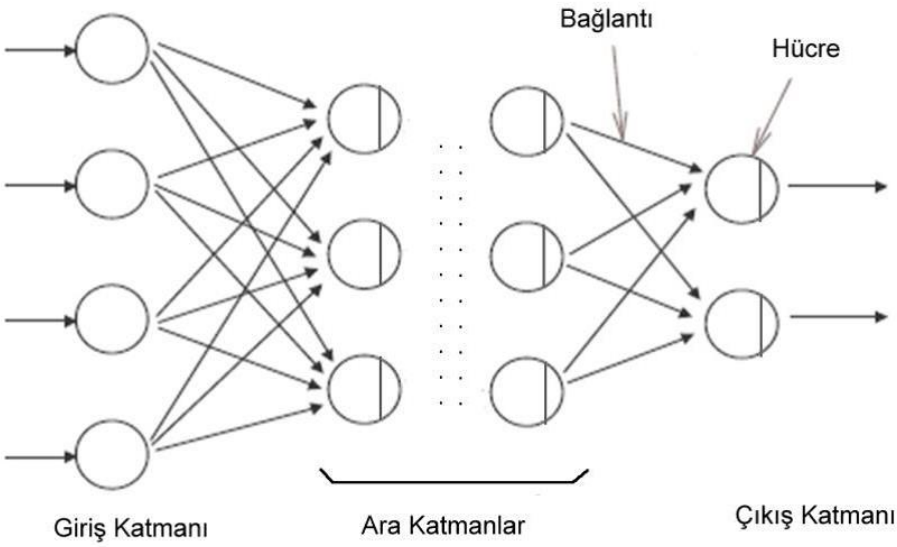
Dentrit: Diğer hücrelerden gelen işaretleri toplayan yapıdır. Sistemin girişleridir.

Akson: Çıkış darbelerinin üretildiği elektriksel aktif gövdedir ve gövde üzerinde iletim tek yönlüdür. Sistem çıkışıdır. İnsandaki sinir hücresinin yapısı Şekil 8.'de gösterilmektedir.



Şekil 8. Sinir hücresinin gösterimi

Bir giriş vektörünü işleyerek çıktılara dönüştüren, katmanlar üzerinden veriyi geçirerek çıkışa ulaştıran yapıya sinir ağı denilmektedir. Her giriş bilgiyi alır ve sıradaki katmana aktarır. Bir katmandan bir başka katmana, aradaki katmanı atlayarak geçebilmek mümkün değildir. (Çetin M, Uğur A, Bayzan Ş. ,2006) Bu aktarımlar, katmanlar arasındaki bağın ağırlığına bağlı ve değeri ölçüsünde değerlenir. Yapay sinir ağlarının 5 temel elemanı mevcuttur. Bunlar girdiler, ağırlıklar, toplam fonksiyonu, aktivasyon fonksiyonu ve çıktı şeklindedir. Şekil 9'da yapay sinir ağları gösterilmektedir.



Şekil 9. Yapay sinir ağlarının gösterimi

Girdiler: Bir yapay sinir hücresinin dışarıdan almış olduğu verilerdir. Sisteme başlangıç olarak aktarılan giriş değerleridir.

Ağırlıklar: Bir yapay hücreye gelen bilginin gücünü ve hücre üzerindeki etkisini gösterir. Şekildeki ağırlık wi, xi girdisinin hücre üzerindeki etkisini göstermektedir. Ağırlıkların değerlerinin küçük olması illaki o hücrenin önemsiz olduğunu göstermez.

Toplam Fonksiyonu: Bu fonksiyon, bir hücreye gelen net girdiyi hesaplar. Bunun için farklı farklı fonksiyonlar kullanılmaktadır. En çok kullanılan fonksiyon ağırlıklı toplam fonksiyonudur. Çizelge 2'de en çok kullanılan fonksiyonlar gösterilmiştir.

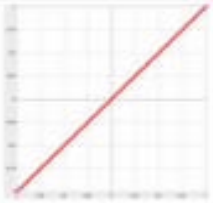
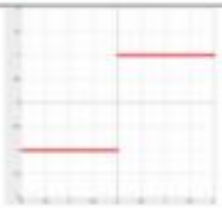
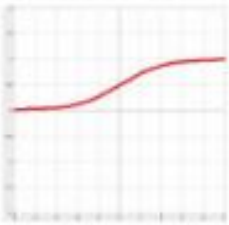
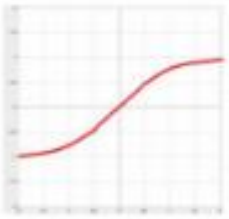
$$E= (x1*w1+x2*w2+x3*w3\ldots)$$

Çizelge 2. En sık kullanılan fonksiyonlar(Makinist,S ,2018)

Toplam $\text{Net} = \sum_{i=1}^{N} X_i * W_i$	Ağırlık değerleri girdiler ile çarpılır ve bulunan değerler birbirleriyle toplanarak Net girdi hesaplanır.
Çarpım $\text{Net} = \prod_{i=1}^{N} X_i * W_i$	Ağırlık değerleri girdiler ile çarpılır ve daha sonra bulunan değerler birbirleriyle çarpılarak Net Girdi Hesaplanır.
Maksimum $\text{Net} = \text{Max}(X_i * W_i)$	n adet girdi içinden ağırlıklar girdilerle çarpıldıktan sonra içlerinden en büyüğü Net girdi olarak kabul edilir.
Minimum $\text{Net} = \text{Min}(X_i * W_i)$	n adet girdi içinden ağırlıklar girdilerle çarpıldıktan sonra içlerinden en küçüğü Net girdi olarak kabul edilir.
Çoğunluk $\text{Net} = \sum_{i=1}^{N} Sgn(X_i * W_i)$	n adet girdi içinden girdilerle ağırlıklar çarpıldıktan sonra pozitif ile negatif olanların sayısı bulunur. Büyük olan sayı hücrenin net girdisi olarak kabul edilir.
Kumilatif Toplam $\text{Net} = \text{Net(eski)} + \sum_{i=1}^{N} X_i * W_i$	Hücreye gelen bilgiler ağırlıklı olarak toplanır. Daha önce hücreye gelen bilgilere yeni hesaplanan girdi değerleri eklenerek hücrenin net girdisi hesaplanır.

Aktivasyon Fonksiyonu: Bu fonksiyon toplam fonksiyonuyla gelen bilgiyi işleyerek, kendisinde hangi sonuca karşılık geleceğini belirler. Çoğunlukla doğrusal olmayan bir fonksiyon belirlenir. Aktivasyon fonksiyonu belirlerken en önemli özellik türevinin rahat alınabileceği bir fonksiyon olmasıdır. En sık kullanılan fonksiyon sigmoid fonksiyonudur. Çizelge 3'te genelde tercih edilen bazı fonksiyonlar gösterilmektedir. (Makinist,S ,2018)

Hücrenin Çıktısı: Aktivasyon fonksiyonun üretmiş olduğu çıktı değeridir. Bu değer tek başına bir sonuç olabildiği gibi başka bir sinir ağının girdisi olarak da kullanılabilmektedir.

Doğrusal (Lineer) Aktivasyon Fonksiyonu		F(NET)=A* NET (A sabit bir sayı)	Doğrusal problemler çözmek amacıyla aktivasyon fonksiyonu doğrusal bir fonksiyon olarak seçilebilir. Toplama fonksiyonundan çıkan sonuç, belli bir katsayı ile çarpılarak hücrenin çıktısı olarak hesaplanır.
Adım (Step) Aktivasyon Fonksiyonu		F(Net)= { 1 if Net>Eşik Değer 0 if Net<=Eşik Değer	Gelen Net girdinin belirlenen bir eşik değerin altında veya üstünde olmasına göre hücrenin çıktısı 1 veya 0 değerini alır.
Sigmoid Aktivasyon Fonksiyonu		$F(Net)=\frac{1}{1+e^{-Net}}$	Sigmoid aktivasyon fonksiyonu sürekli ve türevi alınabilir bir fonksiyondur. Doğrusal olmayışı dolayısıyla yapay sinir ağı uygulamalarında en sık kullanılan fonksiyondur. Bu fonksiyon girdi değerlerinin her biri için 0 ile 1 arasında bir değer üretir.
Tanjant Hiperbolik Aktivasyon Fonksiyonu		$F(Net)=\frac{e^{Net}+e^{-Net}}{e^{Net}-e^{-Net}}$	Tanjant hiperbolik fonksiyonu, sigmoid fonksiyonuna benzer bir fonksiyondur. Sigmoid fonksiyonunda çıkış değerleri 0 ile 1 arasında değişirken hiperbolik tanjant fonksiyonunun çıkış değerleri -1 ile 1 arasında değişmektedir.
Eşik Değer Fonksiyonu		F(Net)= { 0 if Net<=0 Net if 0<Net < 1 1 if Net>= 1	Gelen bilgilerin 0 dan küçük-eşit olduğunda 0 çıktısı, 1 den büyük-eşit olduğunda 1 çıktısı, 0 ile 1 arasında olduğunda ise yine kendisini veren çıktılar üretilebilir.
Sinüs Aktivasyon Fonksiyonu		F(Net) = Sin(Net)	Öğrenilmesi düşünülen olayların sinüs fonksiyonuna uygun dağılım gösterdiği durumlarda kullanılır.

Çizelge 3. En çok kullanılan aktivasyon fonksiyonlarının gösterimi

Yapay sinir ağını eğitebilmek için aşağıdaki aşamaları uygulamak gerekir:

- Ağırlıkları örneklere rastgele atanması,
- Input değerlerini teker teker sinir ağına uygula

- Ağırlık ve hücre değerlerini toplam fonksiyonuna uygula
- Aktivasyon fonksiyonu yardımıyla çıkış değerini hesapla
- Hata değerini hesapla
- Hatayı minimize edecek şekilde ağırlıkları değiştir ve döngüye devam et.
- Minimum hata değerinde döngüden çık.

Yapay sinir ağlarını kullanmanın avantajları:

- Yüksek doğruluk oranları elde etme.
- Öğrenme verisinde hata olduğu durumlarda bile çalışabilme, dayanıklılık.

Yapay sinir ağlarını kullanmanın dezavantajları

- Veriyi işleyecek güçlü donanım gereksinimi olduğu için öğrenme süresi uzundur.
- Öğrenilen fonksiyonun anlaşılması zor.

3.1.3 Bayes sınıflandırıcısı

Bu sınıflandırıcı, Bayes teoremini baz alarak sınıflandırma işlemlerini yapar. Modeli inşa etmesi basittir. Büyük çaplı verilerle kolayca çalışabilmektedir. Bayes sınıflandırıcısında özelliklerin önemi aynı seviyededir. Nitelikler birbirinden tamamen bağımsızdır. Naïve Bayes sınıflandırmasında öncelikle sistemin kendisine öğretilmiş veri verilmektedir. Öğretim için sunulan veriler öncesinde kesinlikle etiketlenip, kategorisi işaretlenmiş olmalıdır. Öğretilmiş veriler üzerinde yapılan olasılık işlemleri ile sisteme sunulan yeni test verileri, daha önce elde edilmiş olasılık değerlerine göre işletilir ve verilen test verisinin hangi sınıfa ait olduğu tahmin edilmeye çalışılmaktadır. Bu sınıflandırıcının doğruluk oranı öğretilmiş veri sayısı arttıkça nispete daha da artacaktır. Naïve Bayes sınıflandırma yönteminin birçok kullanım alanı bulunmaktadır. Sınıflandırma işlemi için kullanılacak veri, ikili değer veya karakter olabilmektedir. Aşağıdaki Çizelge 4'te veriler ve etiketli kategorik bilgileri verilmiştir. Bayes sınıflandırmasına göre öncelikle her bir kategorinin tüm satırlara göre oranı bulunması gerekmektedir. Naive Bayes, tüm değişkenlerin

birbirinden bağımsız ve hepsinin aynı öneme sahip olduğu varsayımlarına dayanan bir algoritmadır.(Zhang H.,(2004))

Çizelge 4. Örnek veri seti

Numara	Öğretilen cümle	Kategorisi
1	Türkiye Mısır Japonya	A
2	Türkiye Singapur Türkiye	A
3	Fransa Türkiye	A
4	Tunus Fas Türkiye	B

P(A) = 3/4 = 0.75 (A kategorisindeki satırların tüm satırlara oranı)

P(B) = 1/4 = 0.25 (B(Tunus) kategorisindeki satırların tüm satırlara oranı)

Sonrasında cümlelerin ait olduğu sınıfa göre koşullu olasılığı hesaplanır. Aşağıda formülasyonu gösterilmektedir.

P(X| Y) =(Y kategorisindeki satırlarda "X" ifadesinin tekrar sayısı +1) / (Y kategorisindeki satırlarda bulunan tüm kelimelerin sayısı + Öğretilen veri sayısı)

Mevcut örnek üzerinde toplam 6 tane durum ortaya çıkmaktadır.

P(Türkiye | A) = (5 + 1) / (8 + 6) = 0.428

P(Tunus | A) = (0 + 1) / (8 + 6) = 0.071

P(Fas| A) = (0 + 1) / (8 + 6) = 0.071

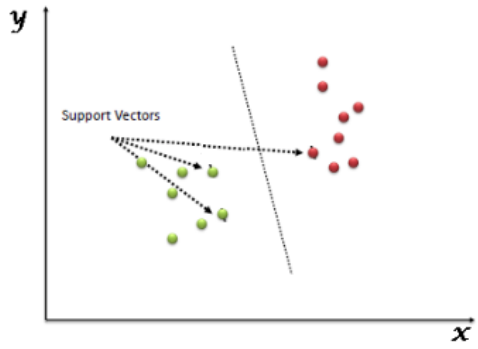
P(Türkiye | B) = (1 + 1) / (3 + 6) = 0.222

P(Tunus| B) = (1 + 1) / (3 + 6) = 0.222

P(Fas| B) = (1 + 1) / (3 + 6) = 0.222

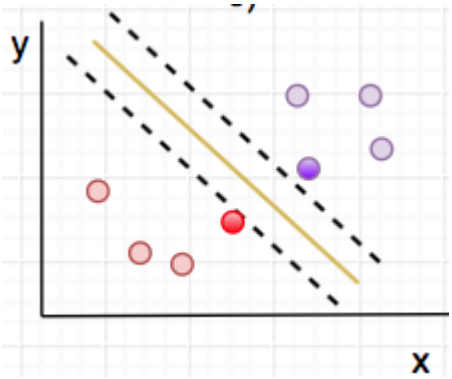
3.1.4 Destek vektör sınıflandırıcısı

SVM sınıflandırıcıları, farklı kategorideki örnekleri eğitim verisi kullanılarak elde edilmiş lineer bir karar fonksiyonu yardımıyla birbirinden ayrılması amaçlanır. Verileri tamamen birbirinden ayıran çizgiye lineer karar doğrusu denilmektedir. Şekil 10'da iki farklı sınıfın bir karar doğrusu yardımıyla ayrımı gösterilmektedir.



Şekil 10. Karar doğrusu yardımıyla verilerin sınıflandırılması

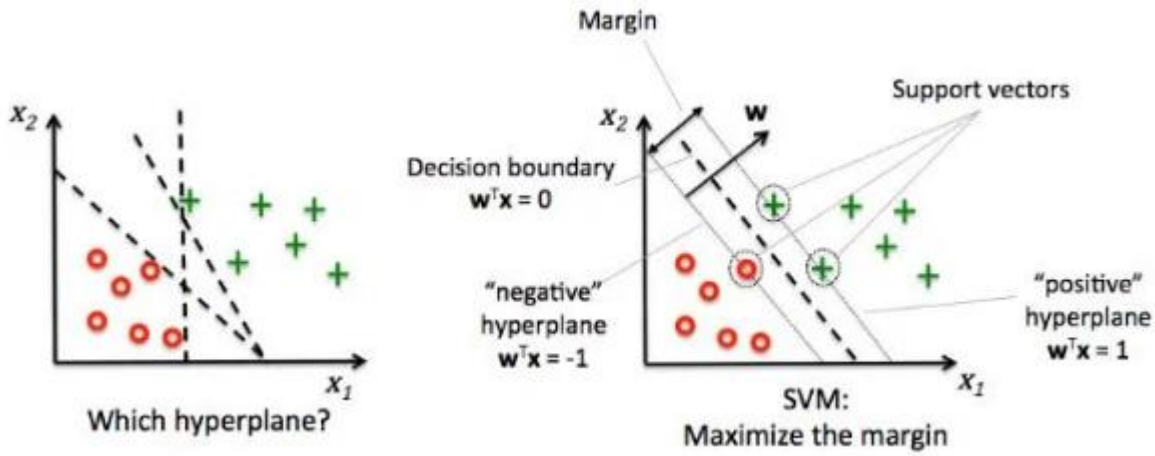
Karar doğrusunun illaki bir tane olması gerekmez. Eğer birden fazla doğru çizilebiliyorsa en optimal doğrunun kullanılması gerekir. SVM sınıflandırıcısı ile kategorik ayrım yapıldığı örneklerde genellikle [-1,+1] veya [0,+1] sınıf etiketleri kullanılmaktadır. İki sınıfın karar doğrusuna en yakın olduğu noktalara karar destek noktaları denilmektedir. Şekil 11'de sınıflandırmayı sağlayan destek vektörlerinin üzerinde bulunduğu kesikli çizgilerle gösterilmiş düzlemlere sınır düzlemleri denir.



Şekil 11. Sınır düzlemi doğrularının gösterimi

İki sınır düzlemine eşit mesafede olan, aynı zamanda margin mesafesinin ortasında yer alan düzleme ise hiper düzlem adı verilmektedir. Sınıflandırma işlemi sırasında

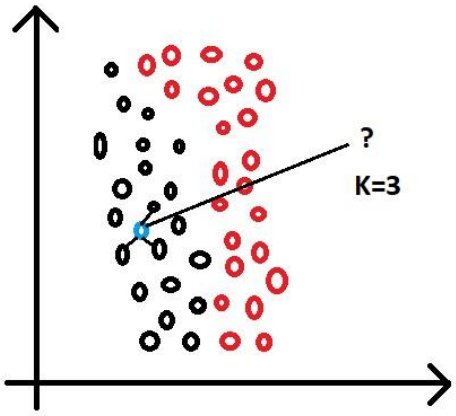
karar doğrusunu mümkün oldukça hiper düzleme yaklaştırmak sınıflandırmanın performansını olumlu yönde etkileyecektir. Şekil 12'de hiper düzlemin belirlenmesi şekil üzerinde gösterilmektedir.



Şekil 12. Hiper düzlemin belirlenmesinin gösterimi

3.1.5 En yakın komşu yöntemi (kNN)

En yakın komşu algoritması en kolay ve öğrenmesi en basit algoritmalardan biridir. Endüstriyel alanda kullanımı başarılı olduğu için çok tercih edilen sınıflandırma yöntemlerinden biridir. Hem regresyon hem sınıflandırma problemlerinde kullanılabilmesi tercih edilmesinin sebeplerinden biridir. En yakın komşu algoritması, etiketlenmiş olan verileri kullanır. Sınıflandırması bulunacak olan örneğin, verilere göre uzaklığı hesaplanıp, k kez yakın komşulukları kontrol edilir. Aşağıdaki Şekil 13'de farklı iki sınıftan hangi sınıfa dahil olabileceğini bulabilmek için 3 komşunun sınıfını kontrol edip, kendi sınıfını belirlemeye çalıştığı görülmektedir. kNN sınıflandırıcısı çoklu sınıflandırma işlemlerine uygun bir sınıflandırıcıdır. Duygu analizi ve tahmini gibi trend problemlerin çözümünde yüksek performans ile çalışabilmektedir.



Şekil 13. kNN sınıflandırıcısının uygulanması

kNN algoritmasından adım adım aşamaları:

- Bu algoritma için ilk olarak k kez sayısı belirlenir. Bu sayı verilen bir noktaya en yakın komşuların sayısıdır.
- Sınıflandırılacak olan verinin, mevcut verilere göre uzaklığı Öklid Teoremi kullanılarak ayrı ayrı hesaplanır.
- En yakın k sayıda komşu ele alınır, k adet komşunun sınıfına uygun olarak yeni verinin sınıf ataması yapılır.

Kullanımı kolay bir sınıflandırma yöntemi olmasına rağmen dezavantajlı yönleri de mevcuttur. Örneğin, uzaklık hesabı yaparken tüm verilerin konum bilgilerini bellekte tutma zorunluluğu olduğu için, büyük veri setleri üzerinde çalışıldığında geniş bellek alanına gereksinim duymaktadır.

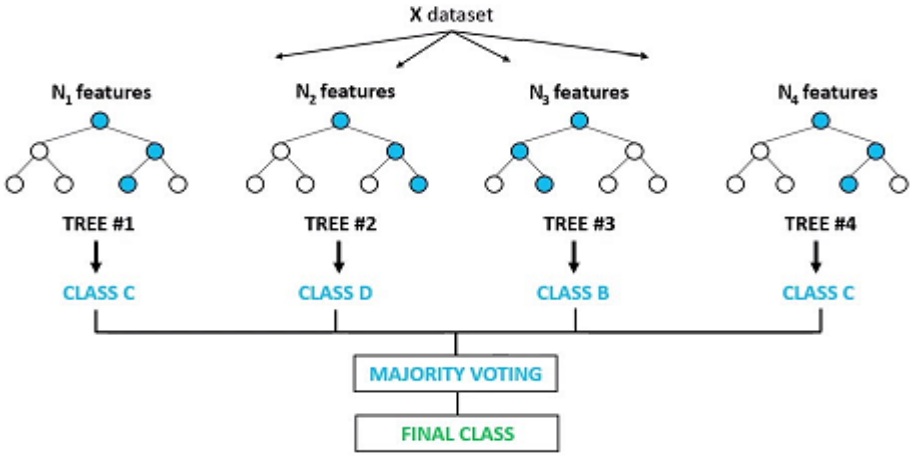
3.1.6 Rastgele orman sınıflandırıcısı

Son yıllarda özellikle sınıflandırma problemlerinde en çok kullanılan yöntemlerden biridir. Kullanımı kolay bir sınıflandırıcıdır. Regresyon ve sınıflandırma problemlerine yanıt verebilmektedir. Rastgele orman algoritması denetimli öğrenme yöntemlerinden biridir. Bu algoritma birçok rastsal olarak karar ağacından orman oluşturma mantığına dayanır. Ağaçları oluştururken en iyi özelliği belirlemek yerine,

rastgele birçok ağaç oluşturur. Bu sonuçların aritmetik ortalamasını alır. Şekil 14'te rastgele orman sınıflandırıcısının çalışma prensibi modellenerek gösterilmektedir. Karar ağaçları ile rastgele orman sınıflandırıcısını birbirinden ayıran en önemli özelliklerden biri de aşırı öğrenmedir. Rastgele orman sınıflandırıcısında çok sayıda fakat küçük ağaçlar oluşturulacağı için büyük ölçüde ezberlemenin önüne geçer ve yüksek doğruluk oranı elde etmeye yardımcı olur. En sonunda elde edilen alt ağaçları birleştirir. Yüksek işlem gücüne ihtiyaç duyulacağı için ağaçların oluşturulması uzun sürebilmektedir.

Rastgele ormanın performansını belirleyen birçok parametre mevcuttur. En önemli parametre oluşturulacak ağaç sayısıdır. Genel olarak ağaç sayısının artması, performansı yükseltir. Fakat işlem sayısını artırdığı için uzun sürmesi beklenmektedir.

Hızını artırmak için başka bir parametre ise bir ağaçta denemeye izin verilecek maksimum özellik sayısıdır.(max_features) Eğer modelin tümünün hızını artırmak isteniyorsa kullanılabilecek işlemci sayısını artırmak etkili olacaktır.(N_jobs)



Şekil 14. Random Forest sınıflandırıcısının çalışma prensibinin gösterimi

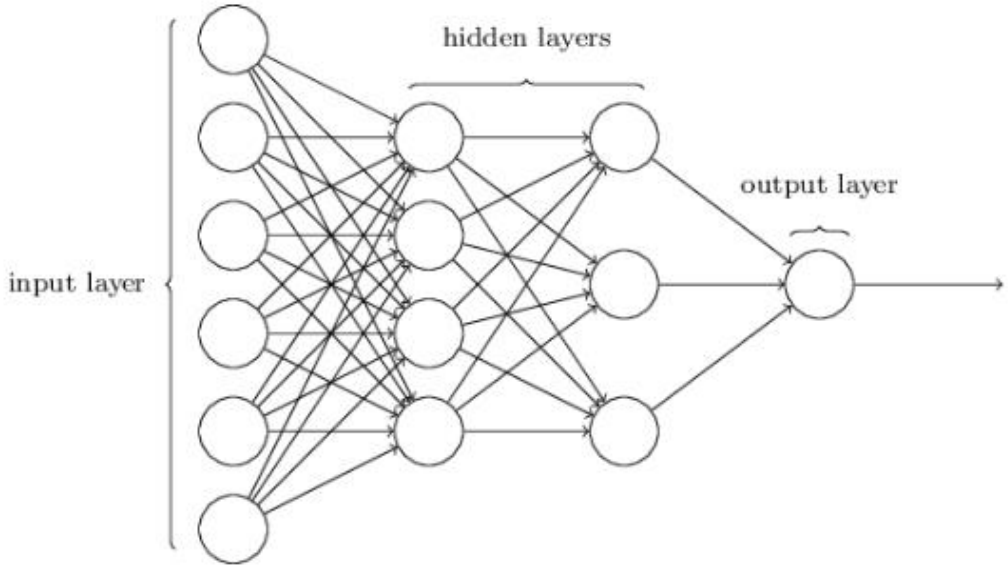
Sınıflandırmanın saflığını bulabilmek için Gini Index hesaplaması yapılmaktadır. Gini değeri ne kadar düşük ise, sınıf o kadar homojendir. Algoritmanın çalışma prensibinde bir alt düğümün gini indeks değeri, bir üst düğümün gini indeksinden küçük ise o dal başarılı olarak değerlendirilmektedir.

Rastgele orman sınıflandırıcısı çok sayıda ağaç oluşturulduğu için, gerçek zamanlı tahmin veya sınıflandırmanın gerekli olduğu endüstriyel alanlarda geri planda kalabilir. Yeteri kadar ağaç oluşturulmadığı takdirde modelde başarısızlık meydana

geleceği öngörülebilir. Bu sınıflandırıcıyı kullanmak duruma göre tercih edilebilecek bir durumdur. Günlük hayatta bankacılık, sigortacılık, e-ticaret sitelerinde oldukça kullanılmaktadır.

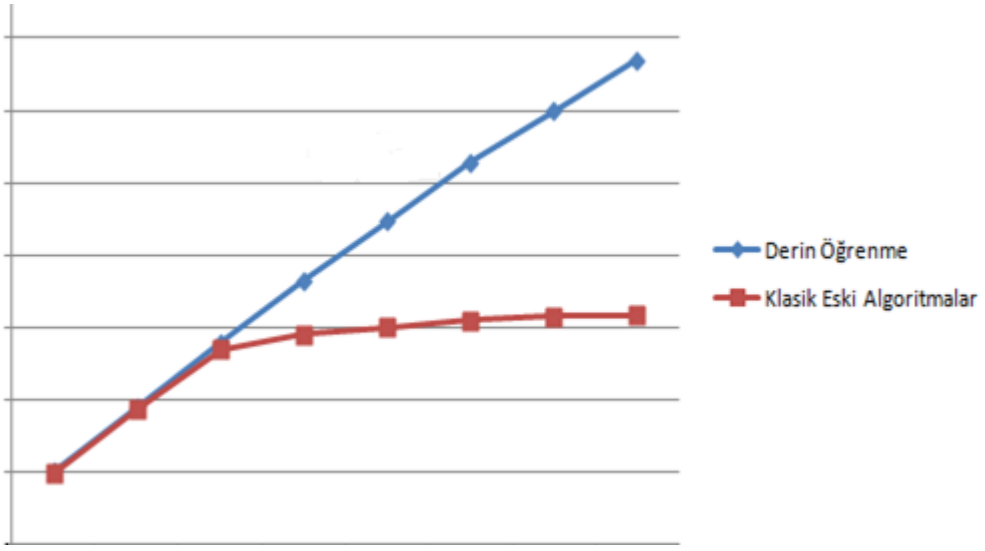
4. DERİN ÖĞRENME

2000'li yılların başlarından itibaren popülerliğini gün geçtikçe artıran derin öğrenme kavramı, makine öğrenmesinin alt dallarından biridir. Denetimli derin beslemeli çok katmanlı perceptronlar için ilk genel, öğrenme algoritması Ivakhnenko ve Lapa tarafından 1965 yılında yayınlanmıştır.(A. G. Ivakhnenko and V. G. Lapa, 1966) Derin öğrenme terimi 2000'li yılların ortalarından itibaren sıklıkla kullanılmış ve üzerine büyük akademik çalışmalar yapılmaya başlanmıştır.



Şekil 15. Katmanlı yapay sinir ağlarının gösterimi

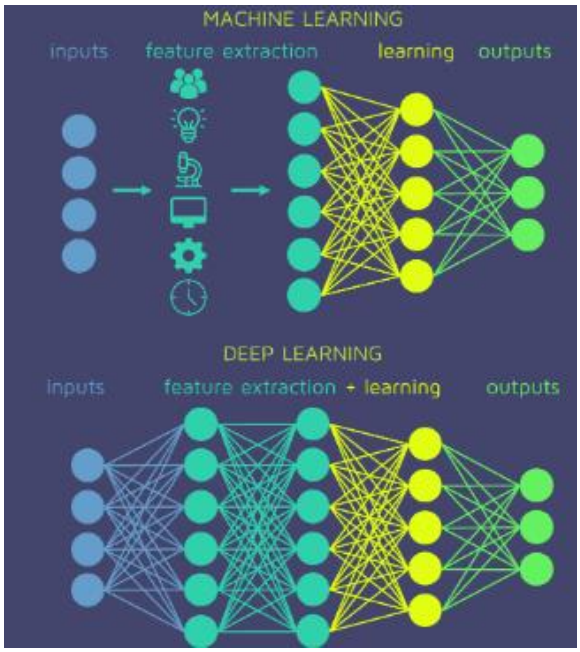
Derin öğrenme metodunda büyük bilgi yığınlarına anlam kazandırmak için çok katmanlı yapay sinir ağları kullanılmaktadır. Katmanlar öncelikle eğitilmek zorundadır. Şekil 15'te katmanlı yapay sinir ağlarının modellemesi görülmektedir. Ağın performansını üst seviyeye çıkarabilmesi kullanılan data miktarı ile doğru orantılıdır. Her ardışık katman, önceki katmandaki çıktıyı girdi olarak alır.(L. Deng and D. Yu, 2014) Derin öğrenme yaklaşımı çoklu soyutlama yapısı ile verinin temsillerini öğrenmek için bir araya getirilmiş çoklu işleme katmanlarında oluşur.(Y. LeCun, Y. Bengio, and G. Hinton, 2015)



Şekil 16. Performans ile bilgi miktarı arasındaki ilişki

Şekil 16.'da görüleceği üzere data miktarı arttıkça yapay sinir ağlarında başarı o kadar artmaktadır. Grafik İşleme Ünitesi(GPU) sayesinde yüksek işlem gücü kullanılarak katmanlı yapay sinir ağları ile tanımlama, analiz etme ve sonuç çıkarma işlemlerin yapılması kolaylaşmıştır.

Bir görüntünün sınıflandırılmasında katmaların en alt seviyesinde pixeller yer almaktadır. Önceden eğitilmiş katmanlı yapay sinir ağlarına input olarak verilen görüntü, çeşitli katmanlarda konumlandırılarak pixel bazında analizler yapılmaktadır. Bu katmanlarda köşeler ve kenarlar olarak sınıflandırılarak, diğer nesnelerden ayrılan özellikleri saptanmaya çalışılır. Diğer katmanlarda ise bu köşe ve kenar bir araya getirilerek göz, burun, kulak gibi özellikleri, başka katmanlarda yüzleri vs. bir araya getirilerek incelenmekte olan görüntünün özellikleri belirlenmektedir. Derin öğrenme metodu ile görüntü analizinin çalışma prensibi Şekil 17'de gösterilmektedir.



Şekil 17. Derin öğrenme metoduyla görüntü tanıma

Resim veya video akışındaki anlık görüntülerini farklı katmanlarda alt bölümlere ayırarak, önceden eğitilmiş ağa input olarak verilerek işlemler gerçekleştirilmektedir.

Facebook, Microsoft, Google gibi teknoloji firmaları bu tekniği, yüz tanıma, konuşma tanıma, otomobillerde nesne tanıma gibi alanlara yatırım yaparak insanlık için oldukça faydalı sonuçlar elde etmektedir. Son zamanlarda adından çok fazla söz ettiren otonom araçlar, üzerlerinde bulundurduğu onlarca sensörler vasıtasıyla aldıkları bilgileri kullanarak etraftaki araçları, ağaçları, canlıları değerlendirip kullanıcı ile paylaşabilmektedir.

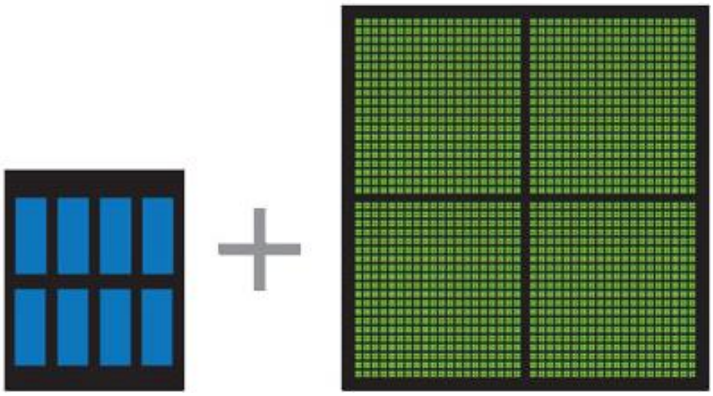
Milyonlarca otomobil, bisiklet, yaya resimleri toplayan otomobil firmaları araçlara yerleştirdikleri sensörler vasıtasıyla nesneleri tanıma, sürücüyü bilgilendirme, aracın güvenlik paketini harekete geçirme gibi ek özellikler sunmaktadır.

4.1 Grafik İşleme Ünitesinin Derin Öğrenmede Kullanımı

Derin öğrenme yönteminin yüksek performanslı çalışabilmesi için birim zamanda işlenen veri miktarının artması gerekmektedir. Merkezi İşlem Ünitesi (CPU) ile milyonlarca veriyi paralel olarak işlemek istenildiğinde performans kaybına neden olmaktadır. Derin öğrenme metodunda uygulama kodunun yaklaşık %5'i uygulamayı

çalıştırmada GPU, geriye kalan yaklaşık %95'inde ise sıralı veri işleme için kullanılmaktadır. Derin öğrenme, görüntü işlemenin bir alt dalı sayılabileceğinden, tıbbi görüntü analizi problemlerin çözümünde de oldukça etkilidir.(S. K. Zhou, H. Greenspan, and D. Shen)

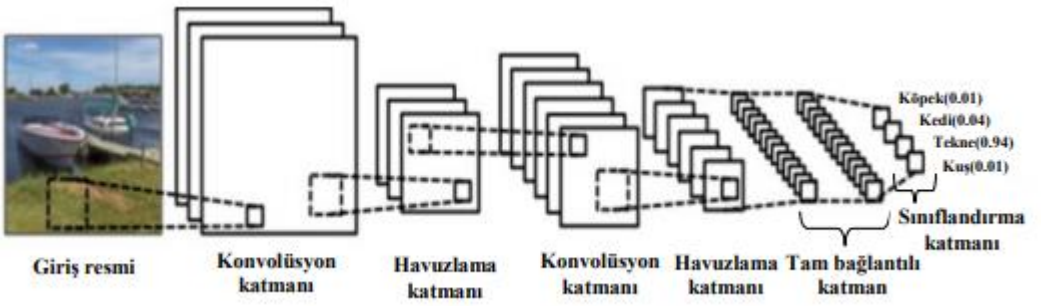
2000'li yılların sonlarına doğru yapılan çalışmalarda yapay sinir ağını eğitme kısmını ise GPU ile yapıldığında çok yüksek performans elde edildiği görülmüştür. CPU ardışık seri işlem için optimize edilen birkaç çekirdekten oluşurken, GPU birden fazla işi eşzamanlı olarak yürütmek için tasarlanan binlerce daha küçük fakat daha yüksek verimle çalışabilen bir mimariye sahiptir. Şekil 18'de CPU ve GPU çekirdeklerinin beraber çalışması gösterilmektedir.



Şekil 18. CPU ve GPU çekirdek yapısı

4.2 Evrimleştirilmiş Yapay Sinir Ağları (ESA)

Evrimleştirilmiş yapay sinir ağları, adından da anlaşılacağı üzere yapay sinir ağlarının ileri halidir. ESA mimarisi, birkaç adet Konvolüsyon (Convulation) ve Havuzlama (Pooling) katmanları, son bölümde ise tam bağlantılı katmanlar ve sınıflandırma katmanından oluşur. ESA mimarisinde, girişten alınan veri ses, video, görüntü olabilmektedir. En son aşamada bir final çıktısı oluşturulur. Oluşan final çıktısı ile istemiş olduğumuz sonuç arasındaki hata algoritmalar yardımıyla geri yönde tüm ağırlıklara dağıtımı yapılır. Bu işlemin binlerce kez tekrar edilmesiyle beraber ağlardaki ağırlıklar güncellenir. Böylece hata payı giderek azalmış olur. Şekil 19'da ESA mimarisinin modellemesi gösterilmiştir.



Şekil 19. ESA Mimarisi

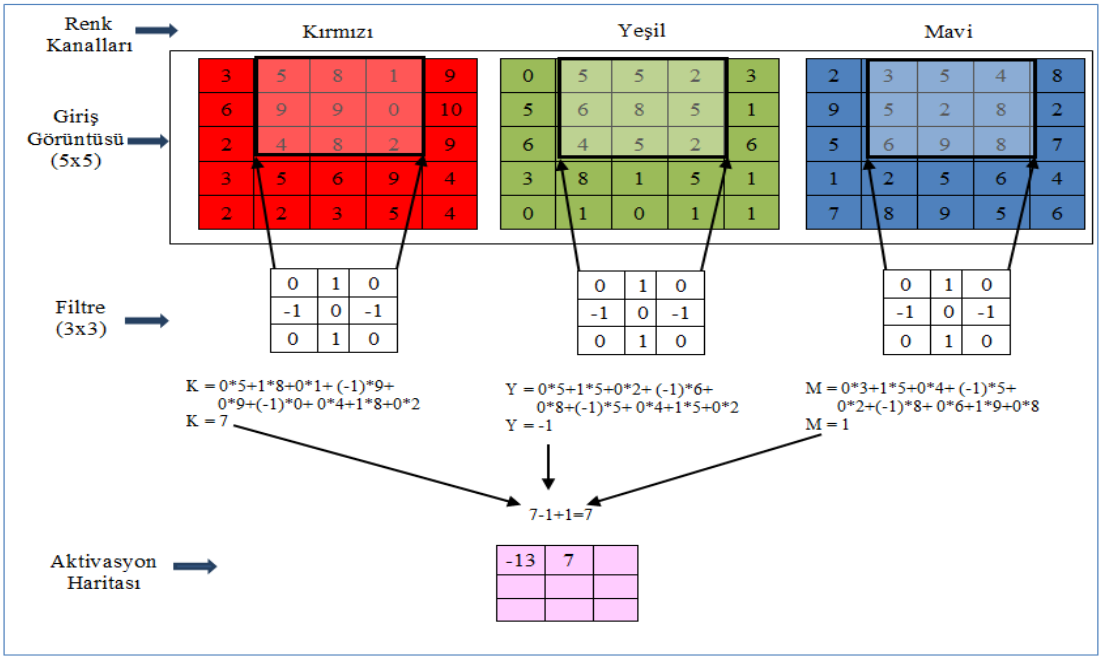
Giriş Katmanı

ESA mimarisinin ilk katmanını oluşturmaktadır. İşlenmemiş veri, ham olarak ağa verilir. Giriş olarak verilen görüntü boyutunun yüksek olmaması ağın performansını doğrudan etkileyen faktörlerdendir. Hem eğitim hem de test süresi için giriş boyutunun düşük tutulması önemlidir.

Konvolüsyon Katmanı

Bu katman, belirlenmiş olan herhangi filtrenin tüm görüntü üzerinde gezdirilmesi mantığına dayanır. Seçtiğimiz filtreler genellikle 2x2,3x3,5x5 boyutlarındadır. Önceki katmanlardan gelen görüntülere dönüşüm uygulayarak yeni bir çıkış oluştururlar. Öğrenme işlemi ilerledikçe, filtrenin katsayısı da değişir. Giriş verisinin 5x5 boyutlarında olduğunu kabul edersek ve renkli(RGB) bir görüntü üzerinde çalışılacaksa, verinin boyutu 5x5x3 olacaktır.

Seçtiğimiz filtre kırmızı, yeşil ve mavi renk kanalları için matris üzerinde gezdirilerek aktivasyon haritası oluşturulması amaçlanır. Şekil 20'de renk kanallarına filtrenin uygulanması gösterilmektedir.



Şekil 20. 5x5x3 olarak giriş yapan görüntüye 3x3 olarak uygulanan filtrenin gösterimi

Havuzlama Katmanı

Havuzlama katmanı olarak bilinen bu katmanda genellikle konvolüsyon işlemi sonrasında olan bir katmandır. (Castelluccio ve ark. 2015). Havuzlama katmanının temel görevi kendisinden bir sonraki konvolüsyon katmanına verilecek bilginin (genişlik x yükseklik) bilgilerini küçültmektir. Boyutları azalmış olan görüntü hem ezberlemeyi önler hem de kendisinden sonraki katmanların iş yükünü azaltmış olur.

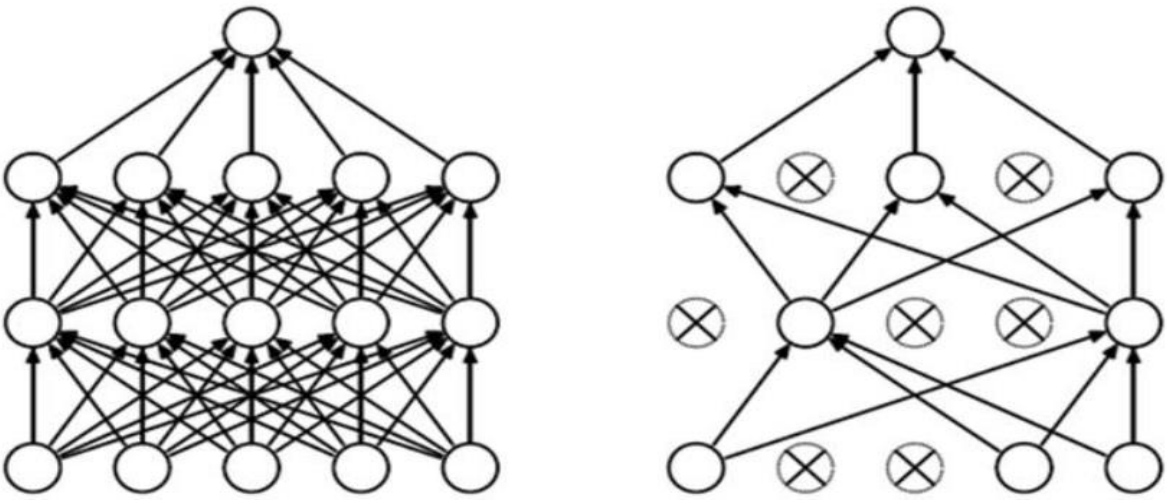
Tam Bağlantılı Katman

Konvolüsyon katmanından ve havuzlama katmanlarından geçen görüntünün hangi sınıfa olduğunu tespit etmek için kullanılan katmandır. Giriş yapan görüntünün hangi nesne olduğunun kararın verildiği bu katmanda, çıkış sayısının aktarımı sağlanır. Sınıflandırmaların yapılmasından sonra kaç tane nesne sınıflandırılacaksa aynı sayıda çıkış değeri olmak zorundadır. Şekil 19'da giriş olarak verilen resmin sınıflandırma katmanından sonra sonuç ürettiği gösterilmektedir.

DropOut Katmanı

ESA mimarisinde,ağlar bazen ezberlemeye(overfitting) yönelmektedir. Bunun önüne geçilmesi için bazı düğümlerin ağ yapısı içerisinden kaldırılmasını sağlamakla

mümkündür.Şekil 21.'de ESA ağına DropOut katmanının uygulanması gösterilmiştir. Ağın performansını artırır.(Xiao ve ark. 2016)



Şekil 21. ESA ağına DropOut katmanın uygulanması

5. CART ALGORİTMASI

5.1 Kullanılan Yazılım Dilleri ve Platformlar

Bu çalışmada Anaconda platformu üzerinde bilimsel çalışma ortamına olanak veren Spyder IDE'si yardımıyla Python yazılım dili kullanılarak yapılmıştır. Sürüm olarak Python 3.7 kullanılmıştır.

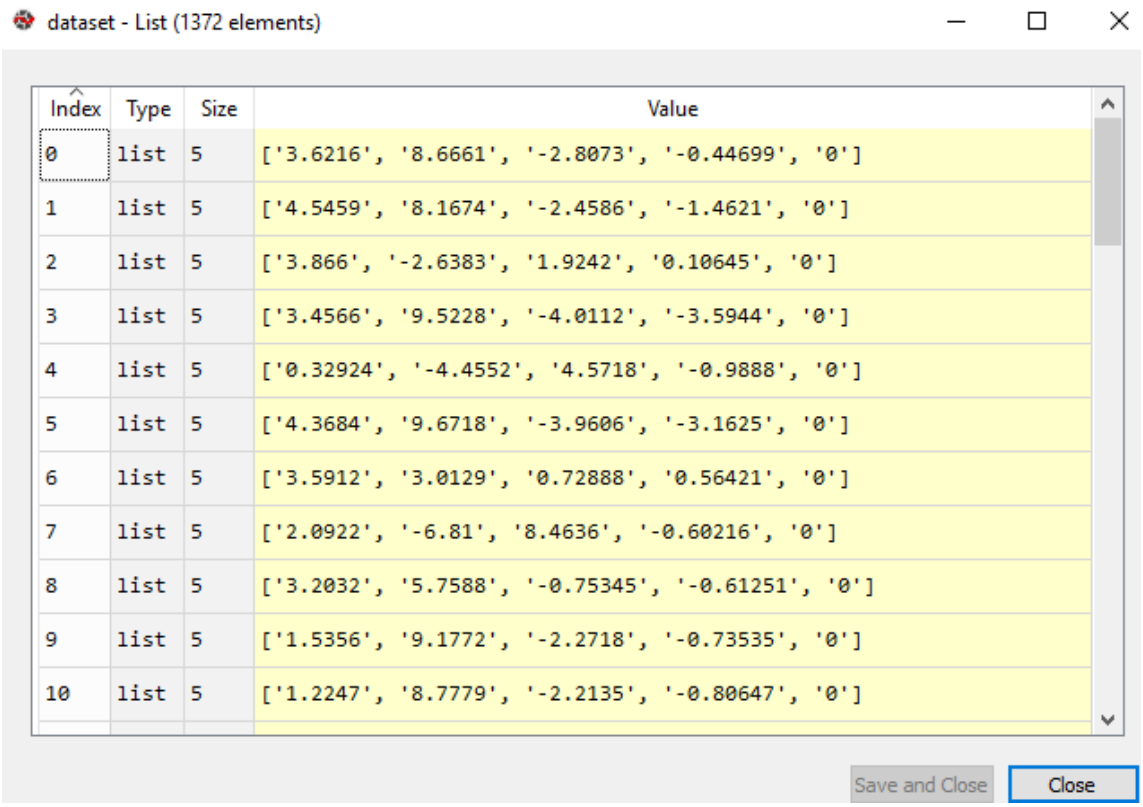
5.2 Veri Setinin İncelenmesi

Sınıflandırma çalışması için Kaggle platformu üzerinden tedarik edilen data_banknote_authentication isimli veri seti kullanılacaktır. Bu veri seti 4 input ve 1 output değeri içermektedir. orjinal ve sahte banknot benzerlerinden alınmış örnek görüntülerden faydalanılarak hazırlanmıştır. Dijitalleştirme işlemi için endüstriyel kamera ile 400*400 pixel görüntüler kullanılmıştır. Yaklaşık 660 dpi çözünürlükte gri renkli fotoğraflardan özellikler çıkarabilmek için dalgacık dönüşümü aracı kullanılarak elde edilmiştir. Çizelge 5.'de veri setinde yer alan sütunlar ile ilgili bilgiler gösterilmektedir.

Çizelge 5. Veri setindeki sütun isimleri

Input 1	Input 2	Input 3	Input 4	Output
Dalgacık Dönüştürülmüş görüntünün varyansı (sürekli).	Dalgacık Dönüştürülmüş görüntünün eğriliği (sürekli)	Dönüştürülmüş görüntünün basıklığı	Görüntünün entropisi (sürekli).	Sınıf (0-1)

Aşağıdaki Şekil 22.'de dosyadan verinin okunmuş hali gösterilmektedir.



Şekil 22. data_banknote_authentication.csv.csv isimli dosyanın okunması

CART algoritması boş veri ile çalışmaya olanak vermesine rağmen daha hassas sonuçlar elde edbilmek amacıyla algoritma aşamasına geçmeden önce NaN veri olup olmadığı kontrol edilmiştir. Mevcut veri setinde rastlanmamıştır. Geliştirilecek algoritma matematiksel işlemler gerektirdiği için tüm verilerin float türünde olmasına dikkat edilmiştir. Toplam 1372 adet veri bulunmaktadır.

5.3 Geliştirilecek CART Algoritmasının Aşamaları

Giriş Parametrelerinin Atanması

Dosya okuma işleminden sonra sınıflandırma algoritmasına geçmeden 3 adet parametre atanmıştır. Çizelge 6.'de parametre isimleri ve atanan ilk değerler görülmektedir.

Çizelge 6. Giriş parametrelerinin gösterimi

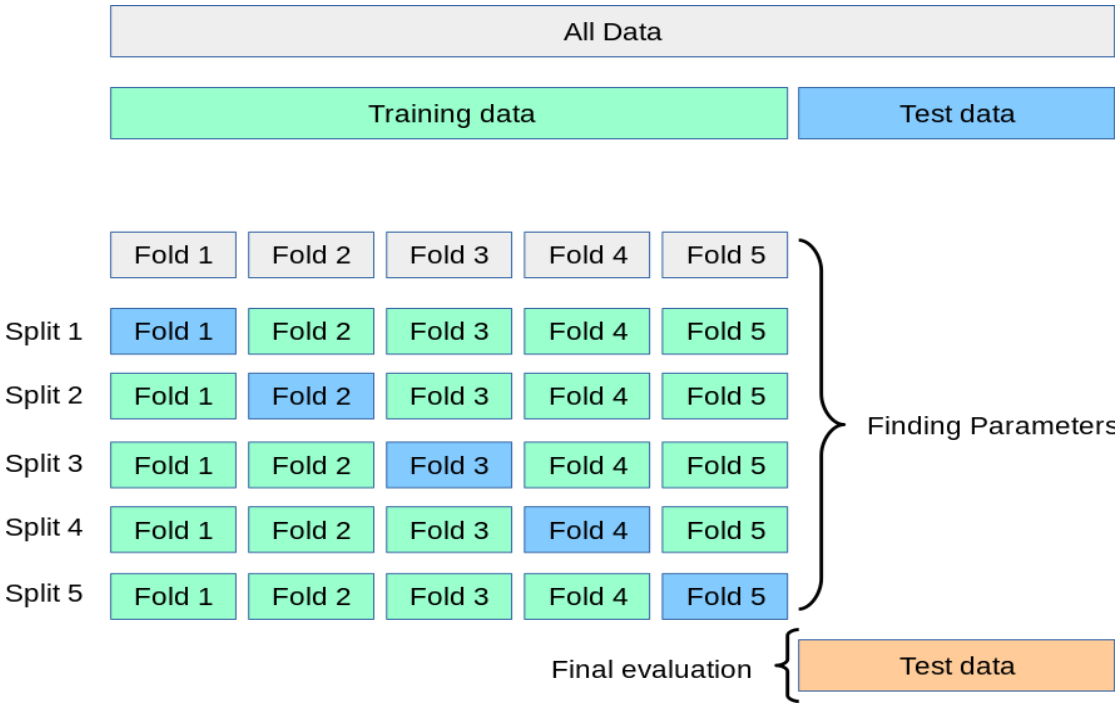
Parametre Adı	**İlk Değeri**
n_folds	5
max_depth	10
min_size	1

n_folds parametresi, cross validation aşamasında veri setinin kaç alt veri kümesine ayrılacağını gösteren parametredir. İlk değer olarak bu çalışmada 5 belirlenmiştir. 5 alt kümeye ayrıldıktan 4 tanesi eğitim için 1 tanesi ise test etmek amacıyla kullanılacaktır. max_depth parametresi, karar ağacının en fazla ulaşabileceği derinliği göstermektedir. İlk değer olarak 10 atanmıştır.min_size parametresi ise, en küçük ağacın boyutunu göstermektedir. Minimum değer olarak 1 belirlenmiştir.

Cross Validation

n_folds parametresi kullanılarak veriseti bölümlere ayrırken mevcut CART algoritmasında Random sınıfının içinde bulunan randrange() fonksiyonu kullanılarak 0 ile len(dataset) arasında bir index değer oluşturur. Bu index degere sahip eleman fold'lara yerleştirilir. n-1 adet alt küme eğitim ve 1 adet test verisi olmak üzere alt veri setleri oluşturulur. Cross Validation yöntemi mümkün olan en küçük hata ile seçimleri yapar. (Arlot S, Celisse A. ,2010:4:40-79)

Veri setini eğitim ve test set olarak ayrılmasının amacı, olası overfitting'den kaçınmak ve modelin daha önceden görmediği veri seti üzerinde nasıl performans gösterdiğini anlamak içindir. Şekil 23'te bu modelleme yapısı gösterilmektedir.



Şekil 23.'de Tüm veri setinden 5 alt veri setine ayrılarak çapraz doğrulama işleminin yapılması

Her test verisi, döngü içersinde daha sonra eğitim verisi olarak da kullanılır. n adet sonuç toplanarak ortalama değer sınıflandırma işleminin doğruluk oranını hesaplamada kullanılır.

Bu çalışmada, mevcut CART algoritmasında Random sınıfının içinde bulunan randrange() fonksiyonu kullanılarak 0 ile len(dataset) arasında üretilen bir index yerine, aşağıda belirtilen aşamalar işletilerek bir index değer üretilecektir.

Step 1 :Sınıflandırma sonucuna etkisi en yüksek sütunu belirle.

Step 2: Sınıflandırma sonucuna etkisi en düşük sütunu belirle.

Step 3: Sınıflandırma sonucuna etkisi en yüksek sütunun ortalama değerini hesapla.

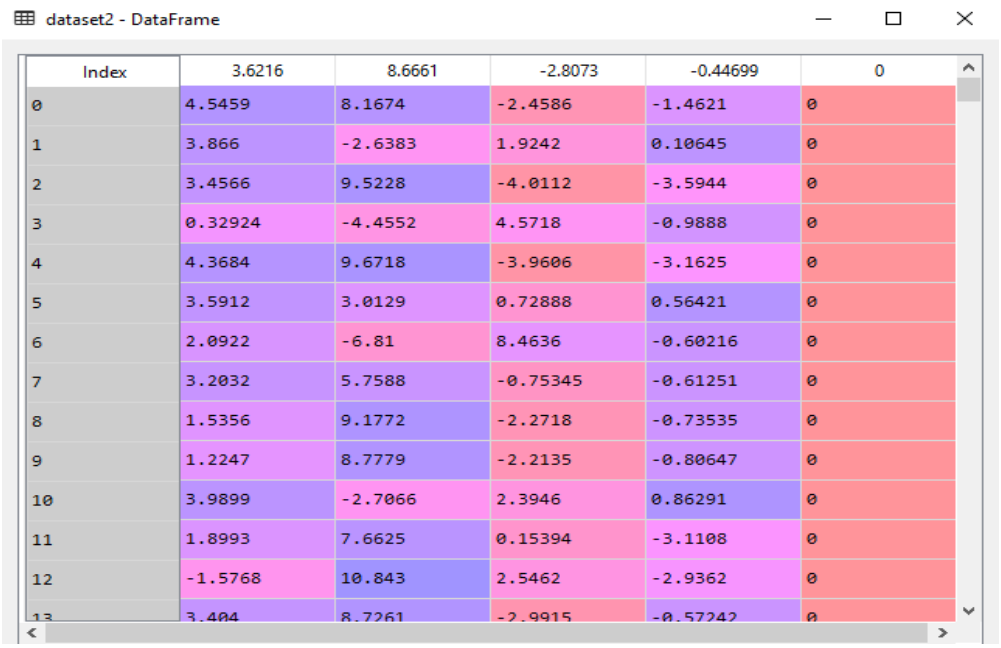
Step 4: Sınıflandırma sonucuna etkisi en düşük sütunun ortalama değerini hesapla.

Step 5: Döngü yapısı içerisinde sınıflandırma sonucuna etkisi en yüksek sütunun ortalama değeri, aynı sütun üzerindeki verileri dolaşırken sıradaki değerden büyükse indexi tut ve döngüden çık.

Step 6: Döngü yapısı içerisinde sınıflandırma sonucuna etkisi en düşük sütunun ortalama değeri, aynı sütun üzerindeki verileri dolaşırken sıradaki değerden büyükse indexi tut ve döngüden çık.

Step 7: Step 5 ve Step 6 aşamalarının ikisini de sağlamıyorsa, Random sınıfının içinde bulunan randrange() fonksiyonu kullanılarak 0 ile len(dataset) arasında bir index değer üret ve döngüden çık.

Step 1 ve Step 2 aşamalarında 4 input değerine sahip bir veri setinin sonuca en çok ve en az etkisi olan sütunları belirlemek için ayrı bir çalışma yapılacaktır. Bu ikinci çalışma için Spyder platformu üzerinde Python 3.7 sürümü kullanılacaktır. Bu python projesi içerisine Pandas ve Scikit-learn kütüphaneleri eklenmiştir. Python yazılım dili bu kütüphaneler sayesinde birçok işlemi kısa ve kolay şekilde yapmaya imkan vermektedir. Pandas kütüphanesi dosya okuma, satır ve sütunlar üzerinde ön işleme yapma, veriler üzerinde istatiksel bilgiler ortaya çıkarma gibi konularda pratik ve kullanıcı dostu bir kütüphanedir.



Index	3.6216	8.6661	-2.8073	-0.44699	0
0	4.5459	8.1674	-2.4586	-1.4621	0
1	3.866	-2.6383	1.9242	0.10645	0
2	3.4566	9.5228	-4.0112	-3.5944	0
3	0.32924	-4.4552	4.5718	-0.9888	0
4	4.3684	9.6718	-3.9606	-3.1625	0
5	3.5912	3.0129	0.72888	0.56421	0
6	2.0922	-6.81	8.4636	-0.60216	0
7	3.2032	5.7588	-0.75345	-0.61251	0
8	1.5356	9.1772	-2.2718	-0.73535	0
9	1.2247	8.7779	-2.2135	-0.80647	0
10	3.9899	-2.7066	2.3946	0.86291	0
11	1.8993	7.6625	0.15394	-3.1108	0
12	-1.5768	10.843	2.5462	-2.9362	0
13	3.404	8.7261	-2.9915	-0.57242	0

Şekil 24. Pandas kütüphanesi ile verilerin okunması

Şekil 24'te Pandas kütüphanesi kullanılarak verilerin python projesi üzerinden okunması gösterilmiştir. Sırasıyla 0,1,2,3 nolu sütunlar sisteme giriş verisi olarak verilecektir. 4 no'lu index ise sınıflandırmanın sonucu olan çıkış verisini verecektir. Scikit-learn kütüphanesi içerisinde yer alan train_test_split nesnesi yardımıyla test ve eğitim verileri otomatik şekilde ayrılacaktır. Bu ayrım sırasında en dikkat edilmesi gereken test_size parametresidir. Parametrik değer ataması istenildiği oranda yapılabilinir; fakat genellikle kullanılan %25'i test ve %75'i eğitim şeklindedir. Herhangi bir oran belirtilmeden train_test_split(X,y) şeklinde bırakılırsa, default değer olarak arka planda 0.25 olarak ayrım gerçekleşecektir. Şekil 25.'te parametrelerin belirlenmesi ve train_test_split nesnesinin kullanımı gösterilmiştir.

```
@author: Batuhan
"""

import pandas as pd
from sklearn.model_selection import train_test_split

# csv dosyamızı okuduk.
dataset2 = pd.read_csv('data_banknote_authentication.csv.csv')

X = dataset2.iloc[:, [0]].values
y = dataset2.iloc[:, 4].values

x_train, x_test, y_train, y_test = train_test_split(X,y,test_size=0.25,random_state=0)
```

Şekil 25. Input ve output parametrelerinin belirlenmesi

Scikit-learn kütüphanesi sınıflandırma işlemini ID3 algoritmasını baz alarak yapmaktadır. Karar ağacı sınıflandırması için, DecisionTreeClassifier sınıflandırıcı nesnesi kullanılacaktır.

DecisionTreeClassifer nesnesi birçok parametre ile kullanıcıdan istenilen özellikler doğrultusunda bir sınıflandırma yapmaya imkan vermektedir. En çok kullanılan parametreler:

criterion: Bölünmenin kalitesini belirlemek için kullanılır. DecisionTreeClassifer'ın desteklediği kriterler: gini safsızlığı için gini, bilgi kazanımı için ise entropi'dir. Opsiyonel bir parametredir. Belirtilmediği durumlarda arka planda default değer olarak gini kriteri üzerinden ayrım işlemlerini yapmaktadır.

splitter: Her düğümdeki bölünmeyi belirlemek için kullanılacak strateji parametresidir. Opsiyoneldir. best veya random olmak üzere 2 farklı yöntemden biri seçilebilinir. Belirtilmediği durumlarda ise arka planda best parametresi üzerinden strateji belirlenecektir.

max_depth: Bu parametre ağacın maximum gidebileceği derinliği belirlemek için kullanılır. Veri türü olarak integer değer almak zorundadır. Opsiyonel bir parametredir. Ağacın maksimum derinliği belirtilmemiş ise, tüm yapraklar saf olana kadar veya tüm yapraklar min_samples_split örneklerinden daha az içerinceye kadar düğümler genişletilir.

min_samples_split : Bir iç düğümünü bölmek için minimum örnek sayı bilgisini tutar. Arka planda parametre değeri belirtilmediği durumlarda 2 üzerinden işlemleri yapar.

Bu çalışmada, DecisionTreeClassifier nesnesi için criterion: ‘gini’ ve max_depth:10 başlangıç değerleri atanarak sınıflandırma yapılacaktır. Şekil 26.’da parametre atamalarının yapılması gösterilmiştir.

```
19 # DecisionTreeClassifier sınıfını import ettik
20 from sklearn.tree import DecisionTreeClassifier
21
22 # DecisionTreeClassifier sınıfından bir nesne ürettik
23 dtc = DecisionTreeClassifier(random_state=0,criterion='gini',max_depth=10)
24
```

Şekil 26. Sınıflandırıcının projeye eklenmesi ve parametre atamalarının yapılması

x_train ve y_train eğitim verileri fit() fonksiyonu yardımıyla ağaç eğitilmektedir. x_test verileri ise predict() fonksiyonu kullanılarak tahminleme işlemi gerçekleştirilecektir. Bu fonksiyonlar scikit-learn kütüphanesinin kullanıcılara sağladığı büyük bir kolaylıktır. Şekil 27’de ağacın eğitilmesi ve tahminde bulunulması işlemleri gösterilmiştir. Tahmin edilen değerler ile gerçek değerlerin ne kadarının gerçekleştiğini ise accuracy_score fonksiyonu yardımıyla hesaplanmaktadır.

```
25 # Makineyi eğitiyoruz
26 dtc.fit(x_train,y_train)
27
28 # Test veri kümemizi verdik ve tahmin işlemini gerçekleştirdik
29 result = dtc.predict(x_test)
30
```

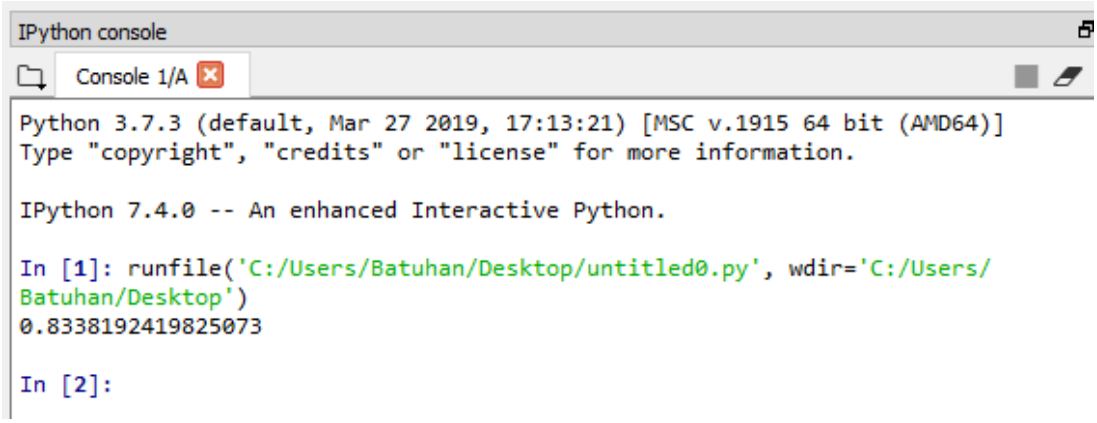
Şekil 27. Eğitim ve tahminleme işlemlerinin uygulanması

Şekil 28.’de ise makinanın tahmin ettiği değerler ile gerçek değerlerin yüzdesel olarak ne kadarının doğru çıktığını öğrenmek için accuracy_score nesnesinin kullanımı gösterilmektedir.

```
31 # Başarı Oranı
32 from sklearn.metrics import accuracy_score
33 accuracy2 = accuracy_score(y_test, result)
34
35 print(accuracy2)
```

Şekil 28. Doğruluk oranının hesaplanması

0 nolu index için sınıflandırma sonucuna tek başına etkisini bulmak için algoritma çalıştığında, Şekil 29.'da görüldüğü gibi yüzdesel bir oran elde edilmiştir. Bu sonuç diğer sütunların hiçbir katkısı olmadan elde edilmiş bir değerdir.



Şekil 29. Algoritmanın sadece 0 no'lu index için çalıştığında sınıflandırmaya olan etkisi

4 giriş sütunu için bu işlemler ayrı ayrı yapılıp, sütunların tek başına sınıflandırma sonucuna etkisi görülecektir. Bu işlemler sonucunda sınıflandırma için en önemli ve en az etkiye sahip sütunlar belirlenecek ve hipotezin Step 1 ve Step 2 aşamalarının yanıtları bulunmuş olacaktır. 4 sütun için de bu çalışma yapıldığında elde edilen sonuçlar Çizelge 7'te gösterilmiştir.Sonuçlar noktadan sonra 2 basamak olacak şekilde düzenlenmiştir. Bu yapılan ek çalışma sonucunda en önemli sütun 1 no'lu, en az etkiye sahip sütun ise 4 no'lu sütun olduğu sonucu ortaya çıkmıştır. Bu sütun index değerleri hipotezin 3.aşamasına geçmek için kullanılacaktır.

Çizelge 7. Giriş sütunlarının sınıflandırma sonucuna etkisi

Sütun No	Tek Başına Sınıflandırma Sonucuna Etkisi
1.Sütun	%83.38

2.Sütun	%70.26
3.Sütun	%65.88
4.Sütun	%53.06

Step 1 ve Step 2 aşamasında görüleceği üzere sınıflandırma sonucuna tek başına en yüksek etkiyi yapan 1.sütun, en az etkiyi yapan ise 4.sütundur. Step 3 ve Step 4 aşamalarında ise float değerlere sahip bu sütunların ortalama değerleri hesaplanacaktır. Cross Validation aşamasında, sırayla veriler ortalama değerden büyük olmasına göre karşılatırılaştırıp ona alt veri setlerine yerleştirilecektir. Aşağıda Şekil 30'da bu 2 sütun için ortalama değerleri hesaplayan fonksiyonlar gösterilmektedir.

```
def best_column_find_Average(dataset_copy):
    total=0
    for i in range(len(dataset_copy)):
         total=total+float(dataset_copy[i][0])

    return float(total/float(len(dataset_copy)))

def worst_column_find_Average(dataset_copy):
    total=0
    for i in range(len(dataset_copy)):
         total=total+float(dataset_copy[i][3])

    return float(total/float(len(dataset_copy)))
```

Şekil 30. Sınıflandırmaya etkisi en fazla ve en az olan sütunların ortalama değerinin bulunması

Cross Validation işleminde alt veri setleri oluşturulurken hangi indeksli verinin yerleşeceğine karar vermek için aşağıda Şekil 31'de belirtilen algoritma kullanılmıştır. Sırayla ana veri setindeki veriler alınıp, get_best_index isimli fonksiyon öncelikle ağırlığı en yüksek olan sütunun ortalama değerinde büyük olma durum kontrolü yapılacaktır. Bu koşulu sağlamıyorsa ağırlığı en az olan sütunun ortalama değerinden büyük olma durum kontrolü yapılacaktır. Bu iki koşulu da eğer sağlamıyorsa, Random sınıfının içerisinde bulunan randrange() fonksiyonu ile 0 ile len(dataset) arasında bir değeri üretip, return edecektir.

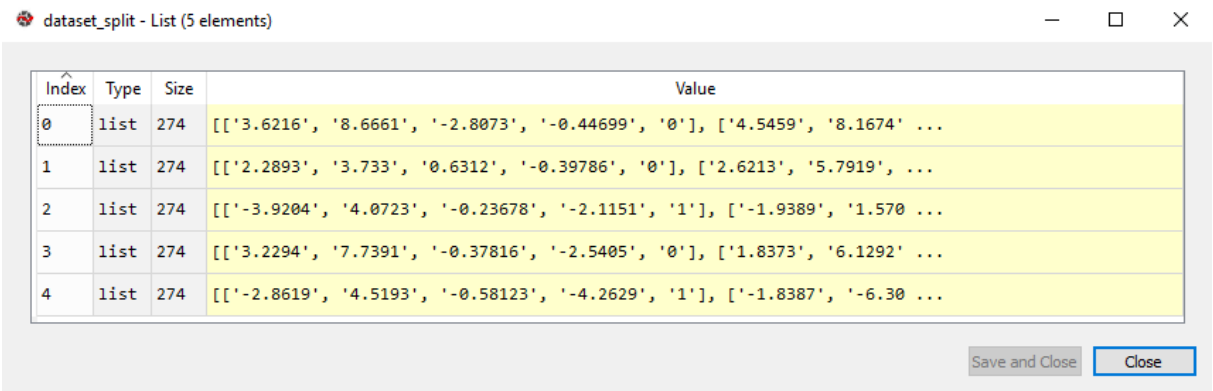
```
def get_best_index(dataset_copy):
    for i in range(len(dataset_copy)):

        if (float(dataset_copy[i][0])>best_column_find_Average(dataset_copy)):
             break
        if (float(dataset_copy[i][3])>worst_column_find_Average(dataset_copy)):
             break
        else:
             return randrange(len(dataset_copy))

    return i

# Split a dataset into k folds
def cross_validation_split(dataset, n_folds):
    dataset_split = list()
    dataset_copy = list(dataset)
    fold_size = (int(len(dataset) / n_folds))

    for i in range(n_folds):
        fold = list()
        while len(fold) < fold_size:

            index =get_best_index(dataset_copy)
            fold.append(dataset_copy.pop(index))
        dataset_split.append(fold)
    return dataset_split
```

Şekil 31. Cross Validation işleminin yapılması

Çizelge 8. Veri setinin kaç alt veriye ayrılacağının gösterilmesi

Parametre Adı	**İlk Değeri**
n_folds	5

Çizelge 8'te görüleceği üzere veri setinde bulunan verilerin geliştirilen algoritmik kurallara göre 5 adet alt veri setine doldurulması sağlanmıştır. Şekil 32'de ayrılmış veriler gösterilmektedir.



Şekil 32. Verilerin alt veri setlerine ayrılmış hali

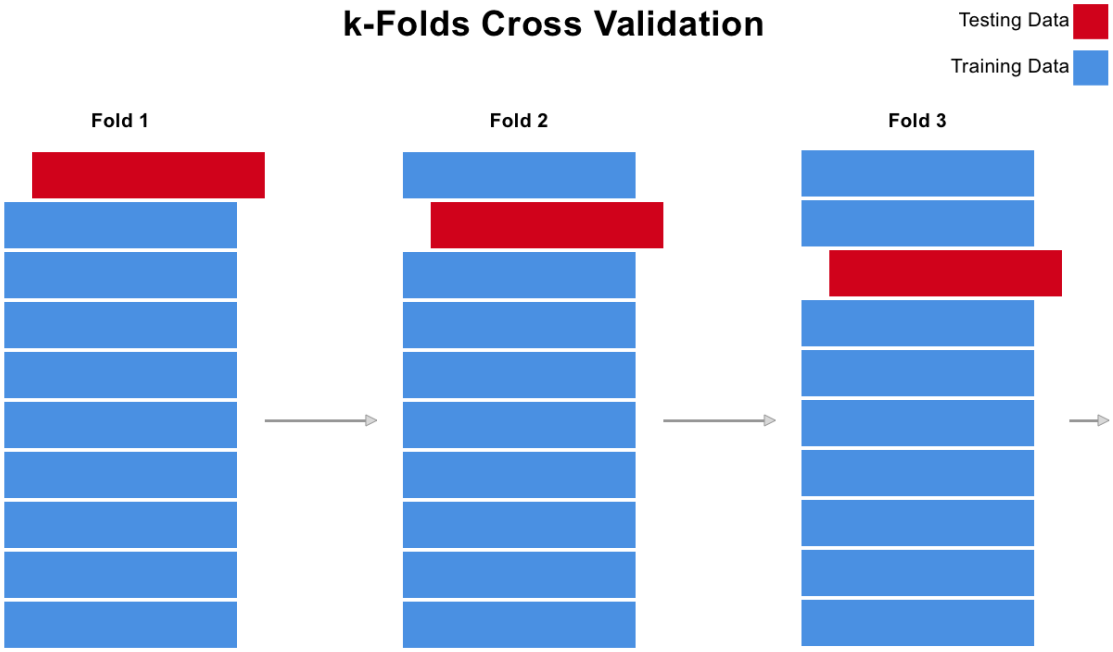
Her bir alt kümenin doğruluk sonuçlarını tutabilmek için bir liste oluşturulup, algoritmanın tamamlanma aşamasından sonra ortalama doğruluk oranını hesaplayabilmek için kullanılacaktır. Şekil 33'de önce cross_validation_split fonksiyonu yardımıyla veri setinden örnekler parametrede belirlenen değer kadar alt veri setlerine yerleştirilir.

```
# Evaluate an algorithm using a cross validation split
def evaluate_algorithm(dataset, algorithm, n_folds, *args):
    folds = cross_validation_split(dataset, n_folds)
    scores = list()
    for fold in folds:
        train_set = list(folds)
        train_set.remove(fold)
        train_set = sum(train_set, [])
        test_set = list()
        for row in fold:
            row_copy = list(row)
            test_set.append(row_copy)
            row_copy[-1] = None
        predicted = algorithm(train_set, test_set, *args)
        actual = [row[-1] for row in fold]
        accuracy = accuracy_metric(actual, predicted)
        scores.append(accuracy)
    return scores
```

Şekil 33. Veri setinden alt veri setlerine verilerin yerleştirilmesi

Üzerinde çalışılacak olan veri seti için 5 ayrı alt veri seti oluşturulacaktır. Sonraki süreçte sırayla 1. Alt veri seti test rolüne, geri kalan 4 alt veri seti ise eğitim veri seti rolünü üstlenecektir. Şekil 34'te bu çapraz kontrol işleminin modellenmiş hali gösterilmektedir.

Alt veri setleri oluşturma işlemi gerçekleştirildikten sonraki aşama ağacın oluşturulma aşamasıdır. Ağacın oluşması için bazı asgari parametrelere ihtiyaç bulunmaktadır. Karar ağacının oluşması için eğitim veri seti, test veri seti, ağacın maksimum derinlik bilgisi ve minimum oluşacak ağaç sayısı parametreleri gönderilmektedir.



Şekil 34. Çapraz doğrulama yönteminin gösterimi

Ağacın inşa edilmesi için öncelikle kök ve düğümlerinin tespit edilmesi ve hangi özelliklere bakılarak yaprakların oluşturulacağının belirlenmesi gerekmektedir. Şekil 35'te build_tree fonksiyonu kullanılarak ağacın inşa edileceği adım gösterilmiştir.

```
def decision_tree(train, test, max_depth, min_size):
    tree = build_tree(train, max_depth, min_size)
    predictions = list()
    for row in test:
        prediction = predict(tree, row)
        predictions.append(prediction)
    return(predictions)
```

Şekil 35. Karar ağacı oluşturma işleminin gösterimi

Kök düğümün hangisinin olacağının kararı get_split fonksiyonu yardımıyla yapılacaktır. Kök düğümün ve bir sonraki düğümlerin belirlenmesi ağacın oluşmasında hayati öneme sahiptir. Şekil 36'da ağaç oluşturma aşamasında öncelikle kök ve düğümleri belirleyen yapı gösterilmektedir.

```
# Build a decision tree
def build_tree(train, max_depth, min_size):
    root = get_split(train)
    split(root, max_depth, min_size, 1)
    return root
```

Şekil 36. get_split fonksiyonu ile kök belirleme işleminin yapılması

Kök düğümden sonra ilk hangi nitelikten bölüneceği ve kurallarının tespit edilmesi gerekmektedir. En iyi ayrım noktalarının tespit edilmesi için Şekil 37'de gini index yöntemi kullanılmıştır. Gini index değeri, bir sınıfın içindeki izafi olarak sıklığını ifade eder. Her bir özellik için ayrı ayrı hesaplanan Gini değerleri arasından en küçük olanı seçilir. Bölünme işlemi bu değere göre yapılmaktadır. Bu işlemler kalan veriler için de tekrar edilir ve diğer bölünmeler için hesaplanır.

```
# Select the best split point for a dataset
def get_split(dataset):
    class_values = list(set(row[-1] for row in dataset))
    b_index, b_value, b_score, b_groups = 10, 10, 10, None
    for index in range(len(dataset[0])-1):
        for row in dataset:
            groups = test_split(index, row[index], dataset)
            gini = gini_index(groups, class_values)
            if gini <= b_score:
                b_index, b_value, b_score, b_groups = index, row[index], gini, groups
    return {'index':b_index, 'value':b_value, 'groups':b_groups}
```

Şekil 37. Gini index değerlerine göre ayrım yapılması

Gini index metodu bilimsel çalışmalarda oldukça fazla kullanılmaktadır. Gini değer hesaplama işlemlerinin Python dilinde yapılması Şekil 38'de gösterilmektedir. Şekil 38'de Gini değerinin hesaplanması için fonksiyon guruplar ve sınıflar olarak 2 adet parametre aldığı görülmektedir.

```python
# Calculate the Gini index for a split dataset
def gini_index(groups, classes):
    # count all samples at split point
    n_instances = float(sum([len(group) for group in groups]))
    # sum weighted Gini index for each group
    gini = 0.0
    for group in groups:
        size = float(len(group))
        # avoid divide by zero
        if size == 0:
            continue
        score = 0.0
        # score the group based on the score for each class
        for class_val in classes:
            p = [row[-1] for row in group].count(class_val) / (size)
            score += (p * p)

        # weight the group score by its relative size
        gini += (1.0 - score) * (size / n_instances)
    return gini
```

Şekil 38. Gini değerinin Python dilinde hesaplanmasının gösterimi

Bir düğümün altında oluşturulacak alt dalların belirlenmesi için Şekil 39'daki split fonksiyonu kullanılacaktır. Bu fonksiyon maksimum derinlik, minimum ağaç sayısı parametrelerini kullanarak ağaçları oluşturacaktır. Ağacı oluşturmadan önce ayrım kontrolü ve maksimum derinliğe ulaşıp oluşmadığı kontrol edilecektir.

```python
# Create child splits for a node or make terminal
def split(node, max_depth, min_size, depth):
    left, right = node['groups']
    del(node['groups'])
    # check for a no split
    if not left or not right:
        node['left'] = node['right'] = to_terminal(left + right)
        return
    # check for max depth
    if depth >= max_depth:
        node['left'], node['right'] = to_terminal(left), to_terminal(right)
        return
    # process left child
    if len(left) <= min_size:
        node['left'] = to_terminal(left)
    else:
        node['left'] = get_split(left)
        split(node['left'], max_depth, min_size, depth+1)
    # process right child
    if len(right) <= min_size:
        node['right'] = to_terminal(right)
    else:
        node['right'] = get_split(right)
        split(node['right'], max_depth, min_size, depth+1)
```

Şekil 39. Split fonksiyonun Python yazılım dilinde oluşturulması

Split fonksiyonu ile ağaç yapısı tasarlandıktan sonra verileri sırasıyla tahmin işlemleri yapılacaktır. Öncelikle sol düğüm sonra sağ düğüm kontrol edilecektir. Şekil 40'da bu işlemlerin algoritmik olarak geliştirmesi gösterilmiştir.

```
# Make a prediction with a decision tree
def predict(node, row):
    if row[node['index']] < node['value']:
        if isinstance(node['left'], dict):
            return predict(node['left'], row)
        else:
            return node['left']
    else:
        if isinstance(node['right'], dict):
            return predict(node['right'], row)
        else:
            return node['right']
```

Şekil 40. Ağaç kullanarak tahmin işlemlerinin yapılması

Döngü içerisinde her bir alt veri seti içerisindeki örnekler için tahmin işlemleri yapıldığında sonunda 5 farklı alt veri seti için tahminler oluşmuş olacaktır. Scores isimli oluşturulacak liste içerisine sonuçlar yerleştirilecektir. Makinanın tahimn ettiği değer ile gerçekteki verilerin karşılaştırmasını yapabilmek amacıyla accuracy_metric fonksiyonu kullanılacaktır. Kolay oluşturulacak bir for döngüsü içerisinde doğru sayısı belirlenecek ve kontrol edilen veriler içerisindeki oranı hesaplanacaktır. Python yazılım dilinde doğruluk oranının bulunması için kullanılacak fonksiyon Şekil 41'de gösterilmiştir.

```
# Calculate accuracy percentage
def accuracy_metric(actual, predicted):
    correct = 0
    for i in range(len(actual)):
        if actual[i] == predicted[i]:
            correct += 1
    return correct / float(len(actual)) * 100.0
```

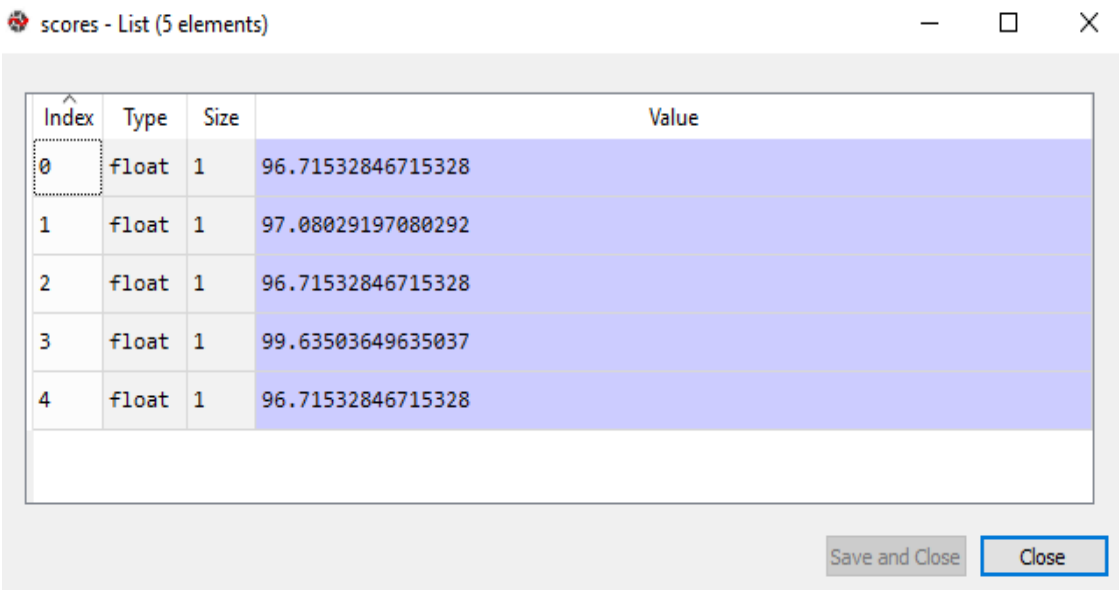
Şekil 41. Doğruluk oranının hesaplanması

accuracy_metric fonksiyonu çalıştığında float veri türünde bir değer döndürecek ve scores list yapısı içerisine 5 kere eklenecektir. Bu algoritmik yapı Şekil 42'de gösterilmiştir.

```
for fold in folds:
    train_set = list(folds)
    train_set.remove(fold)
    train_set = sum(train_set, [])
    test_set = list()
    for row in fold:
        row_copy = list(row)
        test_set.append(row_copy)
        row_copy[-1] = None
    predicted = algorithm(train_set, test_set, *args)
    actual = [row[-1] for row in fold]
    accuracy = accuracy_metric(actual, predicted)
    scores.append(accuracy)
return scores
```

Şekil 42. Doğruluk oranlarının liste yapısına atanması

Döngü yapısı sona erdiği ve yüzdelik oranların belirlenme işlemi sonra 5 farklı alt veri seti için doğruluk oranları hesaplanacaktır. Bazı alt veri setleri için daha yüksek bazı alt veri setleri için daha düşük yüzdelik oranının ortaya çıkması beklenecektir. Kullanılan örnekler, bir paranın sahte mi gerçek mi olduğunu gösteren ve float veri türündeki özellikleri içeren bir veri setidir. Şekil 43'te Scores listesinin içerisine yerleştirilen 5 farklı doğruluk oranı bilgisi gösterilmektedir.



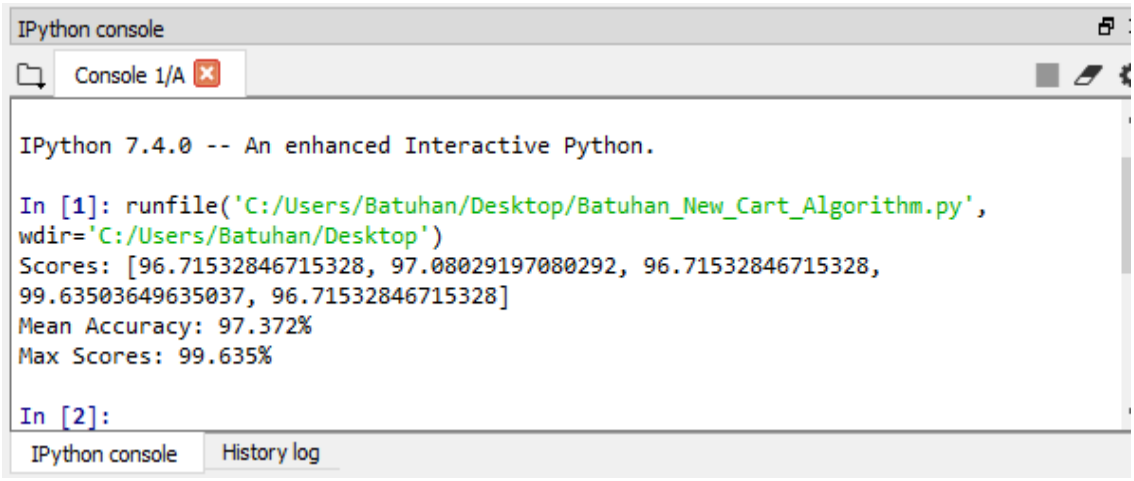
Şekil 43. Scores listesinin içerisindeki değerlerin listelenmesi

5 alt veri seti için elde edilen doğruluk oranlarının hangisinin tüm model için baz alınacağının belirlenmesi için tüm değerlerin aritmetik ortalamasının bulunması işlemi yapılacaktır. Şekil 44'te hem 5 alt veri setinin doğruluk oranlarının listelenmesi, hem modelin ortalama doğruluk değerinin bulunması, hem de 5 alt veri seti için en yüksek doğruluk oranının rapor halinde gösterilmesi için hazırlanan kod bloku görülmektedir.

```
scores = evaluate_algorithm(dataset, decision_tree, n_folds, max_depth, min_size)
print('Scores: %s' % scores)
print('Mean Accuracy: %.3f%%' % (sum(scores)/float(len(scores))))
print('Max Scores: %.3f%%' % max(scores))
```

Şekil 44. Sonuçları gösteren kod bloku

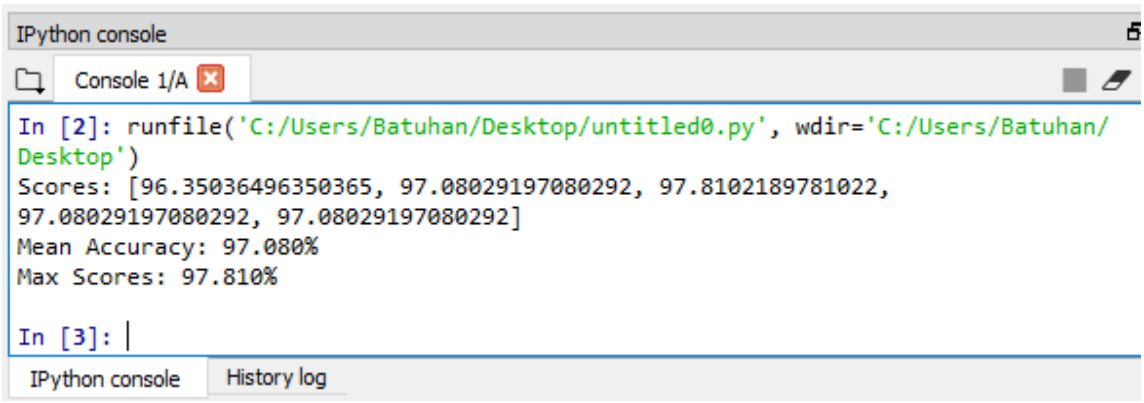
IPython Console yardımıyla sonuçlar listelenmek istenildiğinde Şekil 45'deki sonuçlar elde edilecektir.



Şekil 45. Sonuçların rapor halinde listelenmesi

6.SONUÇ VE ÖNERİLER

Geleneksel CART algoritması kullanılarak aynı veri seti ile sınıflandırma yapıldığında edilen n adet Accuracy, Max. Score, Mean Accuracy değerleri aşağıda Şekil 46'da gösterilmiştir.



Şekil 46. Geleneksel CART algoritması ile sınıflandırma sonuçlarının gösterimi

Aynı veri seti kullanılarak yapılan algoritmik geliştirmeler sonucunda yeni CART algoritmasıyla sınıflandırma işlemi yapıldığında elde edilen sonuçlar Şekil 47'de gösterilmiştir.

```
In [4]: runfile('C:/Users/Batuhan/Desktop/pandas_calısması.py', wdir='C:/Users/Batuhan/Desktop')
Scores: [96.71532846715328, 97.08029197080292, 96.71532846715328, 99.63503649635037, 96.71532846715328]
Mean Accuracy: 97.372%
Max Scores: 99.635%
```

Şekil 47. Algoritmik değişiklikler sonrası CART algoritması ile sınıflandırma sonuçları

Cross Validation aşamasında verilerin 5 alt sete ayrımı için yapılan algoritmik geliştirmeler sonucu daha doğru bir ağaç yapısı ile max score ve mean scores oranlarında performans artışı olduğu görülmektedir. Endüstriyel alanda sınıflandırma

işlemi yapılmak istenildiğinde ön çalışma yapılarak, sınıflandırmaya etkisi en az ve en çok olan özelliklerin belirlenmesinin önemi ortaya çıkmıştır. Yeni algoritmik geliştirmeler sonrasında float veri tipindeki veriler kullanıldığında özellikle finans sektöründe müşteri risk analizi, fraud detection gibi alanlarda doğruluk oranlarının oldukça artacağı öngörülmüştür. Endüstriyel alanda elde edilecek başarının sınıflandırma algoritmalarına olan ilginin ve Ar-Ge çalışmalarının ilerleyen yıllarda daha da artacağı düşünülmektedir.

Sınıflandırma algoritmaları doğru veriler ile çalışıldığı takdirde finans alanında kullanıldığında oldukça doğru sonuçlar elde edilebilmektedir. Yüzde %80 doğruluk oranı ve üzerinde bir oran ile modeller oluşturulmuş ise, %0.5'lik bir artış bile şirketlerin finansal tablolarına büyük katkı sağlayacağı aşikardır. Büyüme hedefleri olan firmaların bu alana yatırım yapmaları çarpan katsayısı çok yüksek oranda geri dönüşüme yol açacak ve şirket hafızası oluşmasında büyük faydalar yaratacaktır. Türk bankacılık sisteminde birçok kurumun veri bilimi ekipleri kurup, işlenmemiş verilerden anlamlı veriler üretmeye çabalaması ve elde ettiği somut sonuçları bilançolarına yansıtmaları veri bilimi sahasına olan ihtiyacı artıracak ve popülerliği yükselecektir.

Büyük veri yığınlarını işleyebilecek güçlü makinelerın ortaya çıkması ile problemlere özgü çözüm yolları inşa edebilecek nitelikli mühendislere olan talebi artıracak ve AR-GE yatırımların önünün açılacağı öngörülmektedir. Önümüzdeki 20 senelik zaman diliminde, dünyanın en çok talep edilen mesleklerinden biri olacağını görmek mümkün olacaktır. Gelecekte yeni sınıflandırma algoritmaları ve performans iyileştirmeleri sayesinde daha yüksek oranlarda doğruluk oranlar elde edilebilinir.

KAYNAKLAR

Begley RJ, Riege M, Rosenblum J, Tseng D. (2000), Adding intelligence to medical devices. Medical Device & Diagnostic Industry Magazine

Teng, J. , Lin, K. ,Ho, B. (2007) "Application of Classification Tree and Logistic Regression for The Management and Health İntervention Plans in A Community-Based Study", Journal of Evaluation in Clinical Practice

İnik Ö.,Ülker E. (2017). Derin Öğrenme ve Görüntü Analizinde Kullanılan Derin Öğrenme Modelleri, Gaziosmanpaşa Bilimsel Araştırma Dergisi

Berry, M. J., Linoff, G. S. (2004) "Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management 2nd ed.", Wiley, USA.

Kurt, I. , Ture, M.,Kurum, A. T. (2008) "Comparing Performances of Logistic Regression, Classification and Regression Tree, and Neural Networks for Predicting Coronary Artery Disease", Expert Systems with Applications.

Gordon, G. ve Pressman, I. (1983). Quantitative Decision-Making For Business. İkinci Baskı, USA: Prentice Hall International, Inc.

Arlot ,S. ve Celisse, A. (2010) A survey of crossvalidation procedures for model selection. Statistics Surveys.

Hand, D., Mannila H. ve Smyth P. (2001), Principles of Data Mining, MIT Press, USA.

Albright, S. C., Winston, W. L. ve Zappe, C. (2006). Data Analysis & Decision Making. Üçüncü Baskı, Australia: Thomson South-Western.

Deconinck, E., Hancock, T., Coomans, D., Massart, D.L., Heyden, Y.V. (2005) "Classification of drugs in absorption classes using the classification and regression trees (CART) methodology", Journal of Pharmaceutical and Biomedical Analysis.

Silahtaroğlu, G. (2009), An Attribute-Centre Based Decision Tree Classification Algorithm [Elektronik Sürüm]. World Academy of Science, Engineering and Technology

Zhang H. (2004), The Optimality of Naive Bayes. In FLAIRS Conference: 2004, Miami Beach, Florida, USA.

Castelluccio, M., Poggi, G., Sansone, C., & Verdoliva, L. (2015). Land use classification in remote sensing images by convolutional neural networks.

Xiao, T., Li, H., Ouyang, W., & Wang, X. (2016). Learning deep feature representations with domain guided dropout for person re-identification. In Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on (pp. 1249-1258). IEEE

Güler, İ., Übeyli E. (2006) ,Çok katmanlı perseptron sinir ağları ile diyabet hastalığının teşhisi. Gazi Üniv. Müh. Mim. Fak. Dergisi

Çetin,M., Uğur, A.,Bayzan, Ş. (2006) ,İleri beslemeli yapay sinir ağlarında backpropagation (geriye yayılım) algoritmasının sezgisel yaklaşımı. Akademik Bilişim Kongresi, Pamukkale Üniversitesi.

Silahtaroğlu, G. (2009), "Kavram ve Algoritmalarıyla Temel Veri Madenciliği", Papatya Yayıncılık Eğitim, İstanbul.

Atmaca,K. Sinir Ağları. Alındığı Tarih: 03.01.2020 Adres: https://kenanatmaca.com/yapay-sinir-aglari-nedir/attachment/2/

Bozan, F. (2010). CART(Classification and Regression Tree) Adres: http://www.farukbozan.com/2010/01/cartclassification-and-regression-tree/> Alındığı Tarih: 25.08.2019

Akın, E. (2017), K-FOLD CROSS VALIDATION (ÇAPRAZ DOĞRULAMA) Alındığı Tarih: 25.10.2019, Adres: http://cagriemreakin.com/veri-bilimi/k-fold-cross-validation-1.html

Makinist, S. ,(2018), Derin Öğrenme (Yapay Sinir Ağları-3) Alındığı Tarih: 25.10.2019, Adres: http://buyukveri.firat.edu.tr/2018/04/16/derin-ogrenme-yapay-sinir-aglari-3/

Güzel K. (2018). Geri Yayılımlı Çok Katmanlı Yapay Sinir Ağları-1. Alındığı Tarih: 03.01.2020. Adres: https://medium.com/@billmuhh/geri-yay%C4%B1l%C4%B1ml%C4%B1-%C3%A7ok-katmanl%C4%B1-yapay-sinir-a%C4%9Flar%C4%B1-1-47daa3856247

Neural Network from Scratch: Perceptron Linear Classifier. Alındığı Tarih: 03.01.2020. Adres: https://jtsulliv.github.io/perceptron/

Url-1<http://code.google.com/p/ourmine/wiki/LectureNaiveBayes#Bayes'_rule> Alındığı Tarih: 06.08.2015.

Url-2<https://scikitlearn.org/stable/modules/cross_validation.html.Alındığı Tarih: 24.09.2019

EKLER

1.Sınıflandırma işlemi öncesinde (0-1-2-3) nolu indekslerin en güçlü ve en zayıf özelliklerin belirlenmesi için kullanılan algoritma aşağıdaki gibidir.

```
#####################################################################
import pandas as pd
from sklearn.model_selection import train_test_split

# csv dosyamızı okuduk.
dataset2 = pd.read_csv('data_banknote_authentication.csv.csv')

X = dataset2.iloc[:, [0]].values
y = dataset2.iloc[:, 4].values

x_train, x_test, y_train, y_test = train_test_split(X,y,test_size=0.25,random_state=0)

# DecisionTreeClassifier sınıfını import ettik
from sklearn.tree import DecisionTreeClassifier

# DecisionTreeClassifier sınıfından bir nesne ürettik
dtc = DecisionTreeClassifier(random_state=0,criterion='gini',max_depth=10)

# Makineyi eğitiyoruz
dtc.fit(x_train,y_train)

# Test veri kümemizi verdik ve tahmin işlemini gerçekleştirdik
result = dtc.predict(x_test)


# Başarı Oranı
from sklearn.metrics import accuracy_score
accuracy2 = accuracy_score(y_test, result)

print(accuracy2)
```

2.Sonuca etkisi en yüksek ve en az olan özellikler belirlendikten sonra sınıflandırma işleminin gerçekleştirildiği algoritma aşağıdaki gibidir.

```
# -*- coding: utf-8 -*-
"""
Created on 11.08.2019 17:29

@author: Batuhan Bilenler
"""



from random import seed
from random import randrange
from csv import reader



# Load a CSV file
def load_csv(filename):
        file = open(filename, "r")
        lines = reader(file)
        dataset = list(lines)
        return dataset

def best_column_find_Average(dataset_copy):
   total=0
   for i in range(len(dataset_copy)):
       total=total+float(dataset_copy[i][0])

   return float(total/float(len(dataset_copy)))

def worst_column_find_Average(dataset_copy):
   total=0
   for i in range(len(dataset_copy)):
       total=total+float(dataset_copy[i][3])

   return float(total/float(len(dataset_copy)))

def get_best_index(dataset_copy):
   for i in range(len(dataset_copy)):

      if (float(dataset_copy[i][0])>best_column_find_Average(dataset_copy)):
```

```
            break
        if (float(dataset_copy[i][3])>worst_column_find_Average(dataset_copy)):
            break
        else:
            return randrange(len(dataset_copy))

    return i

# Split a dataset into k folds
def cross_validation_split(dataset, n_folds):
        dataset_split = list()
        dataset_copy = list(dataset)
        fold_size = (int(len(dataset) / n_folds))

        for i in range(n_folds):
                fold = list()
                while len(fold) < fold_size:

                        index =get_best_index(dataset_copy)
                        fold.append(dataset_copy.pop(index))
                dataset_split.append(fold)
        return dataset_split

# Calculate accuracy percentage
def accuracy_metric(actual, predicted):
        correct = 0
        for i in range(len(actual)):
                if actual[i] == predicted[i]:
                        correct += 1
        return correct / float(len(actual)) * 100.0

# Evaluate an algorithm using a cross validation split
def evaluate_algorithm(dataset, algorithm, n_folds, *args):
        folds = cross_validation_split(dataset, n_folds)
        scores = list()
        for fold in folds:
                train_set = list(folds)
                train_set.remove(fold)
                train_set = sum(train_set, [])
                test_set = list()
                for row in fold:
```

```
                        row_copy = list(row)
                        test_set.append(row_copy)
                        row_copy[-1] = None
                predicted = algorithm(train_set, test_set, *args)
                actual = [row[-1] for row in fold]
                accuracy = accuracy_metric(actual, predicted)
                scores.append(accuracy)
        return scores

# Split a dataset based on an attribute and an attribute value
def test_split(index, value, dataset):
        left, right = list(), list()
        for row in dataset:
                if row[index] <= value:
                        left.append(row)
                else:
                        right.append(row)
        return left, right

# Calculate the Gini index for a split dataset
def gini_index(groups, classes):
        # count all samples at split point
        n_instances = float(sum([len(group) for group in groups]))
        # sum weighted Gini index for each group
        gini = 0.0
        for group in groups:
                size = float(len(group))
                # avoid divide by zero
                if size == 0:
                        continue
                score = 0.0
                # score the group based on the score for each class
                for class_val in classes:
                        p = [row[-1] for row in group].count(class_val) / (size)
                        score += (p * p)

                # weight the group score by its relative size
                gini += (1.0 - score) * (size / n_instances)
        return gini

# Select the best split point for a dataset
```

```
def get_split(dataset):
	class_values = list(set(row[-1] for row in dataset))
	b_index, b_value, b_score, b_groups = 10, 10, 10, None
	for index in range(len(dataset[0])-1):
		for row in dataset:
			groups = test_split(index, row[index], dataset)
			gini = gini_index(groups, class_values)
			if gini <= b_score:
				b_index, b_value, b_score, b_groups = index,
row[index], gini, groups
	return {'index':b_index, 'value':b_value, 'groups':b_groups}

# Create a terminal node value
def to_terminal(group):
	outcomes = [row[-1] for row in group]
	return max(set(outcomes), key=outcomes.count)

# Create child splits for a node or make terminal
def split(node, max_depth, min_size, depth):
	left, right = node['groups']
	del(node['groups'])
	# check for a no split
	if not left or not right:
		node['left'] = node['right'] = to_terminal(left + right)
		return
	# check for max depth
	if depth >= max_depth:
		node['left'], node['right'] = to_terminal(left), to_terminal(right)
		return
	# process left child
	if len(left) <= min_size:
		node['left'] = to_terminal(left)
	else:
		node['left'] = get_split(left)
		split(node['left'], max_depth, min_size, depth+1)
	# process right child
	if len(right) <= min_size:
		node['right'] = to_terminal(right)
	else:
		node['right'] = get_split(right)
		split(node['right'], max_depth, min_size, depth+1)
```

```
# Build a decision tree
def build_tree(train, max_depth, min_size):
	root = get_split(train)
	split(root, max_depth, min_size, 1)
	return root

# Make a prediction with a decision tree
def predict(node, row):
	if row[node['index']] < node['value']:
		if isinstance(node['left'], dict):
			return predict(node['left'], row)
		else:
			return node['left']
	else:
		if isinstance(node['right'], dict):
			return predict(node['right'], row)
		else:
			return node['right']

# Classification and Regression Tree Algorithm
def decision_tree(train, test, max_depth, min_size):
	tree = build_tree(train, max_depth, min_size)
	predictions = list()
	for row in test:
		prediction = predict(tree, row)
		predictions.append(prediction)
	return(predictions)

# Test CART on Bank Note dataset
seed(1)
# load and prepare data
filename = 'data_banknote_authentication.csv.csv'
dataset = load_csv(filename)
# convert string attributes to integers

# evaluate algorithm
n_folds = 5
max_depth = 10
min_size = 1
scores = evaluate_algorithm(dataset, decision_tree, n_folds, max_depth, min_size)
print('Scores: %s' % scores)
```

```
print('Mean Accuracy: %.3f%%' % (sum(scores)/float(len(scores))))
print('Max Scores: %.3f%%' % max(scores))
```

3.Kullanılan Veri Seti (data_banknote_authentication.csv)

```
3.6216,8.6661,-2.8073,-0.44699,0
4.5459,8.1674,-2.4586,-1.4621,0
3.866,-2.6383,1.9242,0.10645,0
3.4566,9.5228,-4.0112,-3.5944,0
0.32924,-4.4552,4.5718,-0.9888,0
4.3684,9.6718,-3.9606,-3.1625,0
3.5912,3.0129,0.72888,0.56421,0
2.0922,-6.81,8.4636,-0.60216,0
3.2032,5.7588,-0.75345,-0.61251,0
1.5356,9.1772,-2.2718,-0.73535,0
1.2247,8.7779,-2.2135,-0.80647,0
3.9899,-2.7066,2.3946,0.86291,0
1.8993,7.6625,0.15394,-3.1108,0
-1.5768,10.843,2.5462,-2.9362,0
3.404,8.7261,-2.9915,-0.57242,0
4.6765,-3.3895,3.4896,1.4771,0
2.6719,3.0646,0.37158,0.58619,0
0.80355,2.8473,4.3439,0.6017,0
1.4479,-4.8794,8.3428,-2.1086,0
5.2423,11.0272,-4.353,-4.1013,0
5.7867,7.8902,-2.6196,-0.48708,0
0.3292,-4.4552,4.5718,-0.9888,0
3.9362,10.1622,-3.8235,-4.0172,0
0.93584,8.8855,-1.6831,-1.6599,0
4.4338,9.887,-4.6795,-3.7483,0
0.7057,-5.4981,8.3368,-2.8715,0
1.1432,-3.7413,5.5777,-0.63578,0
-0.38214,8.3909,2.1624,-3.7405,0
6.5633,9.8187,-4.4113,-3.2258,0
4.8906,-3.3584,3.4202,1.0905,0
-0.24811,-0.17797,4.9068,0.15429,0
1.4884,3.6274,3.308,0.48921,0
4.2969,7.617,-2.3874,-0.96164,0
-0.96511,9.4111,1.7305,-4.8629,0
-1.6162,0.80908,8.1628,0.60817,0
2.4391,6.4417,-0.80743,-0.69139,0
2.6881,6.0195,-0.46641,-0.69268,0
3.6289,0.81322,1.6277,0.77627,0
4.5679,3.1929,-2.1055,0.29653,0
```

3.4805,9.7008,-3.7541,-3.4379,0
4.1711,8.722,-3.0224,-0.59699,0
-0.2062,9.2207,-3.7044,-6.8103,0
-0.0068919,9.2931,-0.41243,-1.9638,0
0.96441,5.8395,2.3235,0.066365,0
2.8561,6.9176,-0.79372,0.48403,0
-0.7869,9.5663,-3.7867,-7.5034,0
2.0843,6.6258,0.48382,-2.2134,0
-0.7869,9.5663,-3.7867,-7.5034,0
3.9102,6.065,-2.4534,-0.68234,0
1.6349,3.286,2.8753,0.087054,0
4.3239,-4.8835,3.4356,-0.5776,0
5.262,3.9834,-1.5572,1.0103,0
3.1452,5.825,-0.51439,-1.4944,0
2.549,6.1499,-1.1605,-1.2371,0
4.9264,5.496,-2.4774,-0.50648,0
4.8265,0.80287,1.6371,1.1875,0
2.5635,6.7769,-0.61979,0.38576,0
5.807,5.0097,-2.2384,0.43878,0
3.1377,-4.1096,4.5701,0.98963,0
-0.78289,11.3603,-0.37644,-7.0495,0
2.888,0.44696,4.5907,-0.24398,0
0.49665,5.527,1.7785,-0.47156,0
4.2586,11.2962,-4.0943,-4.3457,0
1.7939,-1.1174,1.5454,-0.26079,0
5.4021,3.1039,-1.1536,1.5651,0
2.5367,2.599,2.0938,0.20085,0
4.6054,-4.0765,2.7587,0.31981,0
2.4235,9.5332,-3.0789,-2.7746,0
1.0009,7.7846,-0.28219,-2.6608,0
0.12326,8.9848,-0.9351,-2.4332,0
3.9529,-2.3548,2.3792,0.48274,0
4.1373,0.49248,1.093,1.8276,0
4.7181,10.0153,-3.9486,-3.8582,0
4.1654,-3.4495,3.643,1.0879,0
4.4069,10.9072,-4.5775,-4.4271,0
2.3066,3.5364,0.57551,0.41938,0
3.7935,7.9853,-2.5477,-1.872,0
0.049175,6.1437,1.7828,-0.72113,0
0.24835,7.6439,0.9885,-0.87371,0
1.1317,3.9647,3.3979,0.84351,0
2.8033,9.0862,-3.3668,-1.0224,0
4.4682,2.2907,0.95766,0.83058,0
5.0185,8.5978,-2.9375,-1.281,0
1.8664,7.7763,-0.23849,-2.9634,0
3.245,6.63,-0.63435,0.86937,0

4.0296,2.6756,0.80685,0.71679,0
-1.1313,1.9037,7.5339,1.022,0
0.87603,6.8141,0.84198,-0.17156,0
4.1197,-2.7956,2.0707,0.67412,0
3.8027,0.81529,2.1041,1.0245,0
1.4806,7.6377,-2.7876,-1.0341,0
4.0632,3.584,0.72545,0.39481,0
4.3064,8.2068,-2.7824,-1.4336,0
2.4486,-6.3175,7.9632,0.20602,0
3.2718,1.7837,2.1161,0.61334,0
-0.64472,-4.6062,8.347,-2.7099,0
2.9543,1.076,0.64577,0.89394,0
2.1616,-6.8804,8.1517,-0.081048,0
3.82,10.9279,-4.0112,-5.0284,0
-2.7419,11.4038,2.5394,-5.5793,0
3.3669,-5.1856,3.6935,-1.1427,0
4.5597,-2.4211,2.6413,1.6168,0
5.1129,-0.49871,0.62863,1.1189,0
3.3397,-4.6145,3.9823,-0.23751,0
4.2027,0.22761,0.96108,0.97282,0
3.5438,1.2395,1.997,2.1547,0
2.3136,10.6651,-3.5288,-4.7672,0
-1.8584,7.886,-1.6643,-1.8384,0
3.106,9.5414,-4.2536,-4.003,0
2.9163,10.8306,-3.3437,-4.122,0
3.9922,-4.4676,3.7304,-0.1095,0
1.518,5.6946,0.094818,-0.026738,0
3.2351,9.647,-3.2074,-2.5948,0
4.2188,6.8162,-1.2804,0.76076,0
1.7819,6.9176,-1.2744,-1.5759,0
2.5331,2.9135,-0.822,-0.12243,0
3.8969,7.4163,-1.8245,0.14007,0
2.108,6.7955,-0.1708,0.4905,0
2.8969,0.70768,2.29,1.8663,0
0.9297,-3.7971,4.6429,-0.2957,0
3.4642,10.6878,-3.4071,-4.109,0
4.0713,10.4023,-4.1722,-4.7582,0
-1.4572,9.1214,1.7425,-5.1241,0
-1.5075,1.9224,7.1466,0.89136,0
-0.91718,9.9884,1.1804,-5.2263,0
2.994,7.2011,-1.2153,0.3211,0
-2.343,12.9516,3.3285,-5.9426,0
3.7818,-2.8846,2.2558,-0.15734,0
4.6689,1.3098,0.055404,1.909,0
3.4663,1.1112,1.7425,1.3388,0
3.2697,-4.3414,3.6884,-0.29829,0

5.1302,8.6703,-2.8913,-1.5086,0
2.0139,6.1416,0.37929,0.56938,0
0.4339,5.5395,2.033,-0.40432,0
-1.0401,9.3987,0.85998,-5.3336,0
4.1605,11.2196,-3.6136,-4.0819,0
5.438,9.4669,-4.9417,-3.9202,0
5.032,8.2026,-2.6256,-1.0341,0
5.2418,10.5388,-4.1174,-4.2797,0
-0.2062,9.2207,-3.7044,-6.8103,0
2.0911,0.94358,4.5512,1.234,0
1.7317,-0.34765,4.1905,-0.99138,0
4.1736,3.3336,-1.4244,0.60429,0
3.9232,-3.2467,3.4579,0.83705,0
3.8481,10.1539,-3.8561,-4.2228,0
0.5195,-3.2633,3.0895,-0.9849,0
3.8584,0.78425,1.1033,1.7008,0
1.7496,-0.1759,5.1827,1.2922,0
3.6277,0.9829,0.68861,0.63403,0
2.7391,7.4018,0.071684,-2.5302,0
4.5447,8.2274,-2.4166,-1.5875,0
-1.7599,11.9211,2.6756,-3.3241,0
5.0691,0.21313,0.20278,1.2095,0
3.4591,11.112,-4.2039,-5.0931,0
1.9358,8.1654,-0.023425,-2.2586,0
2.486,-0.99533,5.3404,-0.15475,0
2.4226,-4.5752,5.947,0.21507,0
3.9479,-3.7723,2.883,0.019813,0
2.2634,-4.4862,3.6558,-0.61251,0
1.3566,4.2358,2.1341,0.3211,0
5.0452,3.8964,-1.4304,0.86291,0
3.5499,8.6165,-3.2794,-1.2009,0
0.17346,7.8695,0.26876,-3.7883,0
2.4008,9.3593,-3.3565,-3.3526,0
4.8851,1.5995,-0.00029081,1.6401,0
4.1927,-3.2674,2.5839,0.21766,0
1.1166,8.6496,-0.96252,-1.8112,0
1.0235,6.901,-2.0062,-2.7125,0
-1.803,11.8818,2.0458,-5.2728,0
0.11739,6.2761,-1.5495,-2.4746,0
0.5706,-0.0248,1.2421,-0.5621,0
4.0552,-2.4583,2.2806,1.0323,0
-1.6952,1.0657,8.8294,0.94955,0
-1.1193,10.7271,2.0938,-5.6504,0
1.8799,2.4707,2.4931,0.37671,0
3.583,-3.7971,3.4391,-0.12501,0
0.19081,9.1297,-3.725,-5.8224,0

3.6582,5.6864,-1.7157,-0.23751,0
-0.13144,-1.7775,8.3316,0.35214,0
2.3925,9.798,-3.0361,-2.8224,0
1.6426,3.0149,0.22849,-0.147,0
-0.11783,-1.5789,8.03,-0.028031,0
-0.69572,8.6165,1.8419,-4.3289,0
2.9421,7.4101,-0.97709,-0.88406,0
-1.7559,11.9459,3.0946,-4.8978,0
-1.2537,10.8803,1.931,-4.3237,0
3.2585,-4.4614,3.8024,-0.15087,0
1.8314,6.3672,-0.036278,0.049554,0
4.5645,-3.6275,2.8684,0.27714,0
2.7365,-5.0325,6.6608,-0.57889,0
0.9297,-3.7971,4.6429,-0.2957,0
3.9663,10.1684,-4.1131,-4.6056,0
1.4578,-0.08485,4.1785,0.59136,0
4.8272,3.0687,0.68604,0.80731,0
-2.341,12.3784,0.70403,-7.5836,0
-1.8584,7.886,-1.6643,-1.8384,0
4.1454,7.257,-1.9153,-0.86078,0
1.9157,6.0816,0.23705,-2.0116,0
4.0215,-2.1914,2.4648,1.1409,0
5.8862,5.8747,-2.8167,-0.30087,0
-2.0897,10.8265,2.3603,-3.4198,0
4.0026,-3.5943,3.5573,0.26809,0
-0.78689,9.5663,-3.7867,-7.5034,0
4.1757,10.2615,-3.8552,-4.3056,0
0.83292,7.5404,0.65005,-0.92544,0
4.8077,2.2327,-0.26334,1.5534,0
5.3063,5.2684,-2.8904,-0.52716,0
2.5605,9.2683,-3.5913,-1.356,0
2.1059,7.6046,-0.47755,-1.8461,0
2.1721,-0.73874,5.4672,-0.72371,0
4.2899,9.1814,-4.6067,-4.3263,0
3.5156,10.1891,-4.2759,-4.978,0
2.614,8.0081,-3.7258,-1.3069,0
0.68087,2.3259,4.9085,0.54998,0
4.1962,0.74493,0.83256,0.753,0
6.0919,2.9673,-1.3267,1.4551,0
1.3234,3.2964,0.2362,-0.11984,0
1.3264,1.0326,5.6566,-0.41337,0
-0.16735,7.6274,1.2061,-3.6241,0
-1.3,10.2678,-2.953,-5.8638,0
-2.2261,12.5398,2.9438,-3.5258,0
2.4196,6.4665,-0.75688,0.228,0
1.0987,0.6394,5.989,-0.58277,0

4.6464,10.5326,-4.5852,-4.206,0
-0.36038,4.1158,3.1143,-0.37199,0
1.3562,3.2136,4.3465,0.78662,0
0.5706,-0.0248,1.2421,-0.5621,0
-2.6479,10.1374,-1.331,-5.4707,0
3.1219,-3.137,1.9259,-0.37458,0
5.4944,1.5478,0.041694,1.9284,0
-1.3389,1.552,7.0806,1.031,0
-2.3361,11.9604,3.0835,-5.4435,0
2.2596,-0.033118,4.7355,-0.2776,0
0.46901,-0.63321,7.3848,0.36507,0
2.7296,2.8701,0.51124,0.5099,0
2.0466,2.03,2.1761,-0.083634,0
-1.3274,9.498,2.4408,-5.2689,0
3.8905,-2.1521,2.6302,1.1047,0
3.9994,0.90427,1.1693,1.6892,0
2.3952,9.5083,-3.1783,-3.0086,0
3.2704,6.9321,-1.0456,0.23447,0
-1.3931,1.5664,7.5382,0.78403,0
1.6406,3.5488,1.3964,-0.36424,0
2.7744,6.8576,-1.0671,0.075416,0
2.4287,9.3821,-3.2477,-1.4543,0
4.2134,-2.806,2.0116,0.67412,0
1.6472,0.48213,4.7449,1.225,0
2.0597,-0.99326,5.2119,-0.29312,0
0.3798,0.7098,0.7572,-0.4444,0
1.0135,8.4551,-1.672,-2.0815,0
4.5691,-4.4552,3.1769,0.0042961,0
0.57461,10.1105,-1.6917,-4.3922,0
0.5734,9.1938,-0.9094,-1.872,0
5.2868,3.257,-1.3721,1.1668,0
4.0102,10.6568,-4.1388,-5.0646,0
4.1425,-3.6792,3.8281,1.6297,0
3.0934,-2.9177,2.2232,0.22283,0
2.2034,5.9947,0.53009,0.84998,0
3.744,0.79459,0.95851,1.0077,0
3.0329,2.2948,2.1135,0.35084,0
3.7731,7.2073,-1.6814,-0.94742,0
3.1557,2.8908,0.59693,0.79825,0
1.8114,7.6067,-0.9788,-2.4668,0
4.988,7.2052,-3.2846,-1.1608,0
2.483,6.6155,-0.79287,-0.90863,0
1.594,4.7055,1.3758,0.081882,0
-0.016103,9.7484,0.15394,-1.6134,0
3.8496,9.7939,-4.1508,-4.4582,0
0.9297,-3.7971,4.6429,-0.2957,0

4.9342,2.4107,-0.17594,1.6245,0
3.8417,10.0215,-4.2699,-4.9159,0
5.3915,9.9946,-3.8081,-3.3642,0
4.4072,-0.070365,2.0416,1.1319,0
2.6946,6.7976,-0.40301,0.44912,0
5.2756,0.13863,0.12138,1.1435,0
3.4312,6.2637,-1.9513,-0.36165,0
4.052,-0.16555,0.45383,0.51248,0
1.3638,-4.7759,8.4182,-1.8836,0
0.89566,7.7763,-2.7473,-1.9353,0
1.9265,7.7557,-0.16823,-3.0771,0
0.20977,-0.46146,7.7267,0.90946,0
4.068,-2.9363,2.1992,0.50084,0
2.877,-4.0599,3.6259,-0.32544,0
0.3223,-0.89808,8.0883,0.69222,0
-1.3,10.2678,-2.953,-5.8638,0
1.7747,-6.4334,8.15,-0.89828,0
1.3419,-4.4221,8.09,-1.7349,0
0.89606,10.5471,-1.4175,-4.0327,0
0.44125,2.9487,4.3225,0.7155,0
3.2422,6.2265,0.12224,-1.4466,0
2.5678,3.5136,0.61406,-0.40691,0
-2.2153,11.9625,0.078538,-7.7853,0
4.1349,6.1189,-2.4294,-0.19613,0
1.934,-9.2828e-06,4.816,-0.33967,0
2.5068,1.1588,3.9249,0.12585,0
2.1464,6.0795,-0.5778,-2.2302,0
0.051979,7.0521,-2.0541,-3.1508,0
1.2706,8.035,-0.19651,-2.1888,0
1.143,0.83391,5.4552,-0.56984,0
2.2928,9.0386,-3.2417,-1.2991,0
0.3292,-4.4552,4.5718,-0.9888,0
2.9719,6.8369,-0.2702,0.71291,0
1.6849,8.7489,-1.2641,-1.3858,0
-1.9177,11.6894,2.5454,-3.2763,0
2.3729,10.4726,-3.0087,-3.2013,0
1.0284,9.767,-1.3687,-1.7853,0
0.27451,9.2186,-3.2863,-4.8448,0
1.6032,-4.7863,8.5193,-2.1203,0
4.616,10.1788,-4.2185,-4.4245,0
4.2478,7.6956,-2.7696,-1.0767,0
4.0215,-2.7004,2.4957,0.36636,0
5.0297,-4.9704,3.5025,-0.23751,0
1.5902,2.2948,3.2403,0.18404,0
2.1274,5.1939,-1.7971,-1.1763,0
1.1811,8.3847,-2.0567,-0.90345,0

0.3292,-4.4552,4.5718,-0.9888,0
5.7353,5.2808,-2.2598,0.075416,0
2.6718,5.6574,0.72974,-1.4892,0
1.5799,-4.7076,7.9186,-1.5487,0
2.9499,2.2493,1.3458,-0.037083,0
0.5195,-3.2633,3.0895,-0.9849,0
3.7352,9.5911,-3.9032,-3.3487,0
-1.7344,2.0175,7.7618,0.93532,0
3.884,10.0277,-3.9298,-4.0819,0
3.5257,1.2829,1.9276,1.7991,0
4.4549,2.4976,1.0313,0.96894,0
-0.16108,-6.4624,8.3573,-1.5216,0
4.2164,9.4607,-4.9288,-5.2366,0
3.5152,6.8224,-0.67377,-0.46898,0
1.6988,2.9094,2.9044,0.11033,0
1.0607,2.4542,2.5188,-0.17027,0
2.0421,1.2436,4.2171,0.90429,0
3.5594,1.3078,1.291,1.6556,0
3.0009,5.8126,-2.2306,-0.66553,0
3.9294,1.4112,1.8076,0.89782,0
3.4667,-4.0724,4.2882,1.5418,0
3.966,3.9213,0.70574,0.33662,0
1.0191,2.33,4.9334,0.82929,0
0.96414,5.616,2.2138,-0.12501,0
1.8205,6.7562,0.0099913,0.39481,0
4.9923,7.8653,-2.3515,-0.71984,0
-1.1804,11.5093,0.15565,-6.8194,0
4.0329,0.23175,0.89082,1.1823,0
0.66018,10.3878,-1.4029,-3.9151,0
3.5982,7.1307,-1.3035,0.21248,0
-1.8584,7.886,-1.6643,-1.8384,0
4.0972,0.46972,1.6671,0.91593,0
3.3299,0.91254,1.5806,0.39352,0
3.1088,3.1122,0.80857,0.4336,0
-4.2859,8.5234,3.1392,-0.91639,0
-1.2528,10.2036,2.1787,-5.6038,0
0.5195,-3.2633,3.0895,-0.9849,0
0.3292,-4.4552,4.5718,-0.9888,0
0.88872,5.3449,2.045,-0.19355,0
3.5458,9.3718,-4.0351,-3.9564,0
-0.21661,8.0329,1.8848,-3.8853,0
2.7206,9.0821,-3.3111,-0.96811,0
3.2051,8.6889,-2.9033,-0.7819,0
2.6917,10.8161,-3.3,-4.2888,0
-2.3242,11.5176,1.8231,-5.375,0
2.7161,-4.2006,4.1914,0.16981,0

3.3848,3.2674,0.90967,0.25128,0
1.7452,4.8028,2.0878,0.62627,0
2.805,0.57732,1.3424,1.2133,0
5.7823,5.5788,-2.4089,-0.056479,0
3.8999,1.734,1.6011,0.96765,0
3.5189,6.332,-1.7791,-0.020273,0
3.2294,7.7391,-0.37816,-2.5405,0
3.4985,3.1639,0.22677,-0.1651,0
2.1948,1.3781,1.1582,0.85774,0
2.2526,9.9636,-3.1749,-2.9944,0
4.1529,-3.9358,2.8633,-0.017686,0
0.74307,11.17,-1.3824,-4.0728,0
1.9105,8.871,-2.3386,-0.75604,0
-1.5055,0.070346,6.8681,-0.50648,0
0.58836,10.7727,-1.3884,-4.3276,0
3.2303,7.8384,-3.5348,-1.2151,0
-1.9922,11.6542,2.6542,-5.2107,0
2.8523,9.0096,-3.761,-3.3371,0
4.2772,2.4955,0.48554,0.36119,0
1.5099,0.039307,6.2332,-0.30346,0
5.4188,10.1457,-4.084,-3.6991,0
0.86202,2.6963,4.2908,0.54739,0
3.8117,10.1457,-4.0463,-4.5629,0
0.54777,10.3754,-1.5435,-4.1633,0
2.3718,7.4908,0.015989,-1.7414,0
-2.4953,11.1472,1.9353,-3.4638,0
4.6361,-2.6611,2.8358,1.1991,0
-2.2527,11.5321,2.5899,-3.2737,0
3.7982,10.423,-4.1602,-4.9728,0
-0.36279,8.2895,-1.9213,-3.3332,0
2.1265,6.8783,0.44784,-2.2224,0
0.86736,5.5643,1.6765,-0.16769,0
3.7831,10.0526,-3.8869,-3.7366,0
-2.2623,12.1177,0.28846,-7.7581,0
1.2616,4.4303,-1.3335,-1.7517,0
2.6799,3.1349,0.34073,0.58489,0
-0.39816,5.9781,1.3912,-1.1621,0
4.3937,0.35798,2.0416,1.2004,0
2.9695,5.6222,0.27561,-1.1556,0
1.3049,-0.15521,6.4911,-0.75346,0
2.2123,-5.8395,7.7687,-0.85302,0
1.9647,6.9383,0.57722,0.66377,0
3.0864,-2.5845,2.2309,0.30947,0
0.3798,0.7098,0.7572,-0.4444,0
0.58982,7.4266,1.2353,-2.9595,0
0.14783,7.946,1.0742,-3.3409,0

-0.062025,6.1975,1.099,-1.131,0
4.223,1.1319,0.72202,0.96118,0
0.64295,7.1018,0.3493,-0.41337,0
1.941,0.46351,4.6472,1.0879,0
4.0047,0.45937,1.3621,1.6181,0
3.7767,9.7794,-3.9075,-3.5323,0
3.4769,-0.15314,2.53,2.4495,0
1.9818,9.2621,-3.521,-1.872,0
3.8023,-3.8696,4.044,0.95343,0
4.3483,11.1079,-4.0857,-4.2539,0
1.1518,1.3864,5.2727,-0.43536,0
-1.2576,1.5892,7.0078,0.42455,0
1.9572,-5.1153,8.6127,-1.4297,0
-2.484,12.1611,2.8204,-3.7418,0
-1.1497,1.2954,7.701,0.62627,0
4.8368,10.0132,-4.3239,-4.3276,0
-0.12196,8.8068,0.94566,-4.2267,0
1.9429,6.3961,0.092248,0.58102,0
1.742,-4.809,8.2142,-2.0659,0
-1.5222,10.8409,2.7827,-4.0974,0
-1.3,10.2678,-2.953,-5.8638,0
3.4246,-0.14693,0.80342,0.29136,0
2.5503,-4.9518,6.3729,-0.41596,0
1.5691,6.3465,-0.1828,-2.4099,0
1.3087,4.9228,2.0013,0.22024,0
5.1776,8.2316,-3.2511,-1.5694,0
2.229,9.6325,-3.1123,-2.7164,0
5.6272,10.0857,-4.2931,-3.8142,0
1.2138,8.7986,-2.1672,-0.74182,0
0.3798,0.7098,0.7572,-0.4444,0
0.5415,6.0319,1.6825,-0.46122,0
4.0524,5.6802,-1.9693,0.026279,0
4.7285,2.1065,-0.28305,1.5625,0
3.4359,0.66216,2.1041,1.8922,0
0.86816,10.2429,-1.4912,-4.0082,0
3.359,9.8022,-3.8209,-3.7133,0
3.6702,2.9942,0.85141,0.30688,0
1.3349,6.1189,0.46497,0.49826,0
3.1887,-3.4143,2.7742,-0.2026,0
2.4527,2.9653,0.20021,-0.056479,0
3.9121,2.9735,0.92852,0.60558,0
3.9364,10.5885,-3.725,-4.3133,0
3.9414,-3.2902,3.1674,1.0866,0
3.6922,-3.9585,4.3439,1.3517,0
5.681,7.795,-2.6848,-0.92544,0
0.77124,9.0862,-1.2281,-1.4996,0

3.5761,9.7753,-3.9795,-3.4638,0
1.602,6.1251,0.52924,0.47886,0
2.6682,10.216,-3.4414,-4.0069,0
2.0007,1.8644,2.6491,0.47369,0
0.64215,3.1287,4.2933,0.64696,0
4.3848,-3.0729,3.0423,1.2741,0
0.77445,9.0552,-2.4089,-1.3884,0
0.96574,8.393,-1.361,-1.4659,0
3.0948,8.7324,-2.9007,-0.96682,0
4.9362,7.6046,-2.3429,-0.85302,0
-1.9458,11.2217,1.9079,-3.4405,0
5.7403,-0.44284,0.38015,1.3763,0
-2.6989,12.1984,0.67661,-8.5482,0
1.1472,3.5985,1.9387,-0.43406,0
2.9742,8.96,-2.9024,-1.0379,0
4.5707,7.2094,-3.2794,-1.4944,0
0.1848,6.5079,2.0133,-0.87242,0
0.87256,9.2931,-0.7843,-2.1978,0
0.39559,6.8866,1.0588,-0.67587,0
3.8384,6.1851,-2.0439,-0.033204,0
2.8209,7.3108,-0.81857,-1.8784,0
2.5817,9.7546,-3.1749,-2.9957,0
3.8213,0.23175,2.0133,2.0564,0
0.3798,0.7098,0.7572,-0.4444,0
3.4893,6.69,-1.2042,-0.38751,0
-1.7781,0.8546,7.1303,0.027572,0
2.0962,2.4769,1.9379,-0.040962,0
0.94732,-0.57113,7.1903,-0.67587,0
2.8261,9.4007,-3.3034,-1.0509,0
0.0071249,8.3661,0.50781,-3.8155,0
0.96788,7.1907,1.2798,-2.4565,0
4.7432,2.1086,0.1368,1.6543,0
3.6575,7.2797,-2.2692,-1.144,0
3.8832,6.4023,-2.432,-0.98363,0
3.4776,8.811,-3.1886,-0.92285,0
1.1315,7.9212,1.093,-2.8444,0
2.8237,2.8597,0.19678,0.57196,0
1.9321,6.0423,0.26019,-2.053,0
3.0632,-3.3315,5.1305,0.8267,0
-1.8411,10.8306,2.769,-3.0901,0
2.8084,11.3045,-3.3394,-4.4194,0
2.5698,-4.4076,5.9856,0.078002,0
-0.12624,10.3216,-3.7121,-6.1185,0
3.3756,-4.0951,4.367,1.0698,0
-0.048008,-1.6037,8.4756,0.75558,0
0.5706,-0.0248,1.2421,-0.5621,0

0.88444,6.5906,0.55837,-0.44182,0
3.8644,3.7061,0.70403,0.35214,0
1.2999,2.5762,2.0107,-0.18967,0
2.0051,-6.8638,8.132,-0.2401,0
4.9294,0.27727,0.20792,0.33662,0
2.8297,6.3485,-0.73546,-0.58665,0
2.565,8.633,-2.9941,-1.3082,0
2.093,8.3061,0.022844,-3.2724,0
4.6014,5.6264,-2.1235,0.19309,0
5.0617,-0.35799,0.44698,0.99868,0
-0.2951,9.0489,-0.52725,-2.0789,0
3.577,2.4004,1.8908,0.73231,0
3.9433,2.5017,1.5215,0.903,0
2.6648,10.754,-3.3994,-4.1685,0
5.9374,6.1664,-2.5905,-0.36553,0
2.0153,1.8479,3.1375,0.42843,0
5.8782,5.9409,-2.8544,-0.60863,0
-2.3983,12.606,2.9464,-5.7888,0
1.762,4.3682,2.1384,0.75429,0
4.2406,-2.4852,1.608,0.7155,0
3.4669,6.87,-1.0568,-0.73147,0
3.1896,5.7526,-0.18537,-0.30087,0
0.81356,9.1566,-2.1492,-4.1814,0
0.52855,0.96427,4.0243,-1.0483,0
2.1319,-2.0403,2.5574,-0.061652,0
0.33111,4.5731,2.057,-0.18967,0
1.2746,8.8172,-1.5323,-1.7957,0
2.2091,7.4556,-1.3284,-3.3021,0
2.5328,7.528,-0.41929,-2.6478,0
3.6244,1.4609,1.3501,1.9284,0
-1.3885,12.5026,0.69118,-7.5487,0
5.7227,5.8312,-2.4097,-0.24527,0
3.3583,10.3567,-3.7301,-3.6991,0
2.5227,2.2369,2.7236,0.79438,0
0.045304,6.7334,1.0708,-0.9332,0
4.8278,7.7598,-2.4491,-1.2216,0
1.9476,-4.7738,8.527,-1.8668,0
2.7659,0.66216,4.1494,-0.28406,0
-0.10648,-0.76771,7.7575,0.64179,0
0.72252,-0.053811,5.6703,-1.3509,0
4.2475,1.4816,-0.48355,0.95343,0
3.9772,0.33521,2.2566,2.1625,0
3.6667,4.302,0.55923,0.33791,0
2.8232,10.8513,-3.1466,-3.9784,0
-1.4217,11.6542,-0.057699,-7.1025,0
4.2458,1.1981,0.66633,0.94696,0

4.1038,-4.8069,3.3491,-0.49225,0
1.4507,8.7903,-2.2324,-0.65259,0
3.4647,-3.9172,3.9746,0.36119,0
1.8533,6.1458,1.0176,-2.0401,0
3.5288,0.71596,1.9507,1.9375,0
3.9719,1.0367,0.75973,1.0013,0
3.534,9.3614,-3.6316,-1.2461,0
3.6894,9.887,-4.0788,-4.3664,0
3.0672,-4.4117,3.8238,-0.81682,0
2.6463,-4.8152,6.3549,0.003003,0
2.2893,3.733,0.6312,-0.39786,0
1.5673,7.9274,-0.056842,-2.1694,0
4.0405,0.51524,1.0279,1.106,0
4.3846,-4.8794,3.3662,-0.029324,0
2.0165,-0.25246,5.1707,1.0763,0
4.0446,11.1741,-4.3582,-4.7401,0
-0.33729,-0.64976,7.6659,0.72326,0
-2.4604,12.7302,0.91738,-7.6418,0
4.1195,10.9258,-3.8929,-4.1802,0
2.0193,0.82356,4.6369,1.4202,0
1.5701,7.9129,0.29018,-2.1953,0
2.6415,7.586,-0.28562,-1.6677,0
5.0214,8.0764,-3.0515,-1.7155,0
4.3435,3.3295,0.83598,0.64955,0
1.8238,-6.7748,8.3873,-0.54139,0
3.9382,0.9291,0.78543,0.6767,0
2.2517,-5.1422,4.2916,-1.2487,0
5.504,10.3671,-4.413,-4.0211,0
2.8521,9.171,-3.6461,-1.2047,0
1.1676,9.1566,-2.0867,-0.80647,0
2.6104,8.0081,-0.23592,-1.7608,0
0.32444,10.067,-1.1982,-4.1284,0
3.8962,-4.7904,3.3954,-0.53751,0
2.1752,-0.8091,5.1022,-0.67975,0
1.1588,8.9331,-2.0807,-1.1272,0
4.7072,8.2957,-2.5605,-1.4905,0
-1.9667,11.8052,-0.40472,-7.8719,0
4.0552,0.40143,1.4563,0.65343,0
2.3678,-6.839,8.4207,-0.44829,0
0.33565,6.8369,0.69718,-0.55691,0
4.3398,-5.3036,3.8803,-0.70432,0
1.5456,8.5482,0.4187,-2.1784,0
1.4276,8.3847,-2.0995,-1.9677,0
-0.27802,8.1881,-3.1338,-2.5276,0
0.93611,8.6413,-1.6351,-1.3043,0
4.6352,-3.0087,2.6773,1.212,0

1.5268,-5.5871,8.6564,-1.722,0
0.95626,2.4728,4.4578,0.21636,0
-2.7914,1.7734,6.7756,-0.39915,0
5.2032,3.5116,-1.2538,1.0129,0
3.1836,7.2321,-1.0713,-2.5909,0
0.65497,5.1815,1.0673,-0.42113,0
5.6084,10.3009,-4.8003,-4.3534,0
1.105,7.4432,0.41099,-3.0332,0
3.9292,-2.9156,2.2129,0.30817,0
1.1558,6.4003,1.5506,0.6961,0
2.5581,2.6218,1.8513,0.40257,0
2.7831,10.9796,-3.557,-4.4039,0
3.7635,2.7811,0.66119,0.34179,0
-2.6479,10.1374,-1.331,-5.4707,0
1.0652,8.3682,-1.4004,-1.6509,0
-1.4275,11.8797,0.41613,-6.9978,0
5.7456,10.1808,-4.7857,-4.3366,0
5.086,3.2798,-1.2701,1.1189,0
3.4092,5.4049,-2.5228,-0.89958,0
-0.2361,9.3221,2.1307,-4.3793,0
3.8197,8.9951,-4.383,-4.0327,0
-1.1391,1.8127,6.9144,0.70127,0
4.9249,0.68906,0.77344,1.2095,0
2.5089,6.841,-0.029423,0.44912,0
-0.2062,9.2207,-3.7044,-6.8103,0
3.946,6.8514,-1.5443,-0.5582,0
-0.278,8.1881,-3.1338,-2.5276,0
1.8592,3.2074,-0.15966,-0.26208,0
0.56953,7.6294,1.5754,-3.2233,0
3.4626,-4.449,3.5427,0.15429,0
3.3951,1.1484,2.1401,2.0862,0
5.0429,-0.52974,0.50439,1.106,0
3.7758,7.1783,-1.5195,0.40128,0
4.6562,7.6398,-2.4243,-1.2384,0
4.0948,-2.9674,2.3689,0.75429,0
1.8384,6.063,0.54723,0.51248,0
2.0153,0.43661,4.5864,-0.3151,0
3.5251,0.7201,1.6928,0.64438,0
3.757,-5.4236,3.8255,-1.2526,0
2.5989,3.5178,0.7623,0.81119,0
1.8994,0.97462,4.2265,0.81377,0
3.6941,-3.9482,4.2625,1.1577,0
4.4295,-2.3507,1.7048,0.90946,0
6.8248,5.2187,-2.5425,0.5461,0
1.8967,-2.5163,2.8093,-0.79742,0
2.1526,-6.1665,8.0831,-0.34355,0

3.3004,7.0811,-1.3258,0.22283,0
2.7213,7.05,-0.58808,0.41809,0
3.8846,-3.0336,2.5334,0.20214,0
4.1665,-0.4449,0.23448,0.27843,0
0.94225,5.8561,1.8762,-0.32544,0
5.1321,-0.031048,0.32616,1.1151,0
0.38251,6.8121,1.8128,-0.61251,0
3.0333,-2.5928,2.3183,0.303,0
2.9233,6.0464,-0.11168,-0.58665,0
1.162,10.2926,-1.2821,-4.0392,0
3.7791,2.5762,1.3098,0.5655,0
0.77765,5.9781,1.1941,-0.3526,0
-0.38388,-1.0471,8.0514,0.49567,0
0.21084,9.4359,-0.094543,-1.859,0
2.9571,-4.5938,5.9068,0.57196,0
4.6439,-3.3729,2.5976,0.55257,0
3.3577,-4.3062,6.0241,0.18274,0
3.5127,2.9073,1.0579,0.40774,0
2.6562,10.7044,-3.3085,-4.0767,0
-1.3612,10.694,1.7022,-2.9026,0
-0.278,8.1881,-3.1338,-2.5276,0
1.04,-6.9321,8.2888,-1.2991,0
2.1881,2.7356,1.3278,-0.1832,0
4.2756,-2.6528,2.1375,0.94437,0
-0.11996,6.8741,0.91995,-0.6694,0
2.9736,8.7944,-3.6359,-1.3754,0
3.7798,-3.3109,2.6491,0.066365,0
5.3586,3.7557,-1.7345,1.0789,0
1.8373,6.1292,0.84027,0.55257,0
1.2262,0.89599,5.7568,-0.11596,0
-0.048008,-0.56078,7.7215,0.453,0
0.5706,-0.024841,1.2421,-0.56208,0
4.3634,0.46351,1.4281,2.0202,0
3.482,-4.1634,3.5008,-0.078462,0
0.51947,-3.2633,3.0895,-0.98492,0
2.3164,-2.628,3.1529,-0.08622,0
-1.8348,11.0334,3.1863,-4.8888,0
1.3754,8.8793,-1.9136,-0.53751,0
-0.16682,5.8974,0.49839,-0.70044,0
0.29961,7.1328,-0.31475,-1.1828,0
0.25035,9.3262,-3.6873,-6.2543,0
2.4673,1.3926,1.7125,0.41421,0
0.77805,6.6424,-1.1425,-1.0573,0
3.4465,2.9508,1.0271,0.5461,0
2.2429,-4.1427,5.2333,-0.40173,0
3.7321,-3.884,3.3577,-0.0060486,0

4.3365,-3.584,3.6884,0.74912,0
-2.0759,10.8223,2.6439,-4.837,0
4.0715,7.6398,-2.0824,-1.1698,0
0.76163,5.8209,1.1959,-0.64613,0
-0.53966,7.3273,0.46583,-1.4543,0
2.6213,5.7919,0.065686,-1.5759,0
3.0242,-3.3378,2.5865,-0.54785,0
5.8519,5.3905,-2.4037,-0.061652,0
0.5706,-0.0248,1.2421,-0.5621,0
3.9771,11.1513,-3.9272,-4.3444,0
1.5478,9.1814,-1.6326,-1.7375,0
0.74054,0.36625,2.1992,0.48403,0
0.49571,10.2243,-1.097,-4.0159,0
1.645,7.8612,-0.87598,-3.5569,0
3.6077,6.8576,-1.1622,0.28231,0
3.2403,-3.7082,5.2804,0.41291,0
3.9166,10.2491,-4.0926,-4.4659,0
3.9262,6.0299,-2.0156,-0.065531,0
5.591,10.4643,-4.3839,-4.3379,0
3.7522,-3.6978,3.9943,1.3051,0
1.3114,4.5462,2.2935,0.22541,0
3.7022,6.9942,-1.8511,-0.12889,0
4.364,-3.1039,2.3757,0.78532,0
3.5829,1.4423,1.0219,1.4008,0
4.65,-4.8297,3.4553,-0.25174,0
5.1731,3.9606,-1.983,0.40774,0
3.2692,3.4184,0.20706,-0.066824,0
2.4012,1.6223,3.0312,0.71679,0
1.7257,-4.4697,8.2219,-1.8073,0
4.7965,6.9859,-1.9967,-0.35001,0
4.0962,10.1891,-3.9323,-4.1827,0
2.5559,3.3605,2.0321,0.26809,0
3.4916,8.5709,-3.0326,-0.59182,0
0.5195,-3.2633,3.0895,-0.9849,0
2.9856,7.2673,-0.409,-2.2431,0
4.0932,5.4132,-1.8219,0.23576,0
1.7748,-0.76978,5.5854,1.3039,0
5.2012,0.32694,0.17965,1.1797,0
-0.45062,-1.3678,7.0858,-0.40303,0
4.8451,8.1116,-2.9512,-1.4724,0
0.74841,7.2756,1.1504,-0.5388,0
5.1213,8.5565,-3.3917,-1.5474,0
3.6181,-3.7454,2.8273,-0.71208,0
0.040498,8.5234,1.4461,-3.9306,0
-2.6479,10.1374,-1.331,-5.4707,0
0.37984,0.70975,0.75716,-0.44441,0

-0.95923,0.091039,6.2204,-1.4828,0
2.8672,10.0008,-3.2049,-3.1095,0
1.0182,9.109,-0.62064,-1.7129,0
-2.7143,11.4535,2.1092,-3.9629,0
3.8244,-3.1081,2.4537,0.52024,0
2.7961,2.121,1.8385,0.38317,0
3.5358,6.7086,-0.81857,0.47886,0
-0.7056,8.7241,2.2215,-4.5965,0
4.1542,7.2756,-2.4766,-1.2099,0
0.92703,9.4318,-0.66263,-1.6728,0
1.8216,-6.4748,8.0514,-0.41855,0
-2.4473,12.6247,0.73573,-7.6612,0
3.5862,-3.0957,2.8093,0.24481,0
0.66191,9.6594,-0.28819,-1.6638,0
4.7926,1.7071,-0.051701,1.4926,0
4.9852,8.3516,-2.5425,-1.2823,0
0.75736,3.0294,2.9164,-0.068117,0
4.6499,7.6336,-1.9427,-0.37458,0
-0.023579,7.1742,0.78457,-0.75734,0
0.85574,0.0082678,6.6042,-0.53104,0
0.88298,0.66009,6.0096,-0.43277,0
4.0422,-4.391,4.7466,1.137,0
2.2546,8.0992,-0.24877,-3.2698,0
0.38478,6.5989,-0.3336,-0.56466,0
3.1541,-5.1711,6.5991,0.57455,0
2.3969,0.23589,4.8477,1.437,0
4.7114,2.0755,-0.2702,1.2379,0
4.0127,10.1477,-3.9366,-4.0728,0
2.6606,3.1681,1.9619,0.18662,0
3.931,1.8541,-0.023425,1.2314,0
0.01727,8.693,1.3989,-3.9668,0
3.2414,0.40971,1.4015,1.1952,0
2.2504,3.5757,0.35273,0.2836,0
-1.3971,3.3191,-1.3927,-1.9948,1
0.39012,-0.14279,-0.031994,0.35084,1
-1.6677,-7.1535,7.8929,0.96765,1
-3.8483,-12.8047,15.6824,-1.281,1
-3.5681,-8.213,10.083,0.96765,1
-2.2804,-0.30626,1.3347,1.3763,1
-1.7582,2.7397,-2.5323,-2.234,1
-0.89409,3.1991,-1.8219,-2.9452,1
0.3434,0.12415,-0.28733,0.14654,1
-0.9854,-6.661,5.8245,0.5461,1
-2.4115,-9.1359,9.3444,-0.65259,1
-1.5252,-6.2534,5.3524,0.59912,1
-0.61442,-0.091058,-0.31818,0.50214,1

-0.36506,2.8928,-3.6461,-3.0603,1
-5.9034,6.5679,0.67661,-6.6797,1
-1.8215,2.7521,-0.72261,-2.353,1
-0.77461,-1.8768,2.4023,1.1319,1
-1.8187,-9.0366,9.0162,-0.12243,1
-3.5801,-12.9309,13.1779,-2.5677,1
-1.8219,-6.8824,5.4681,0.057313,1
-0.3481,-0.38696,-0.47841,0.62627,1
0.47368,3.3605,-4.5064,-4.0431,1
-3.4083,4.8587,-0.76888,-4.8668,1
-1.6662,-0.30005,1.4238,0.024986,1
-2.0962,-7.1059,6.6188,-0.33708,1
-2.6685,-10.4519,9.1139,-1.7323,1
-0.47465,-4.3496,1.9901,0.7517,1
1.0552,1.1857,-2.6411,0.11033,1
1.1644,3.8095,-4.9408,-4.0909,1
-4.4779,7.3708,-0.31218,-6.7754,1
-2.7338,0.45523,2.4391,0.21766,1
-2.286,-5.4484,5.8039,0.88231,1
-1.6244,-6.3444,4.6575,0.16981,1
0.50813,0.47799,-1.9804,0.57714,1
1.6408,4.2503,-4.9023,-2.6621,1
0.81583,4.84,-5.2613,-6.0823,1
-5.4901,9.1048,-0.38758,-5.9763,1
-3.2238,2.7935,0.32274,-0.86078,1
-2.0631,-1.5147,1.219,0.44524,1
-0.91318,-2.0113,-0.19565,0.066365,1
0.6005,1.9327,-3.2888,-0.32415,1
0.91315,3.3377,-4.0557,-1.6741,1
-0.28015,3.0729,-3.3857,-2.9155,1
-3.6085,3.3253,-0.51954,-3.5737,1
-6.2003,8.6806,0.0091344,-3.703,1
-4.2932,3.3419,0.77258,-0.99785,1
-3.0265,-0.062088,0.68604,-0.055186,1
-1.7015,-0.010356,-0.99337,-0.53104,1
-0.64326,2.4748,-2.9452,-1.0276,1
-0.86339,1.9348,-2.3729,-1.0897,1
-2.0659,1.0512,-0.46298,-1.0974,1
-2.1333,1.5685,-0.084261,-1.7453,1
-1.2568,-1.4733,2.8718,0.44653,1
-3.1128,-6.841,10.7402,-1.0172,1
-4.8554,-5.9037,10.9818,-0.82199,1
-2.588,3.8654,-0.3336,-1.2797,1
0.24394,1.4733,-1.4192,-0.58535,1
-1.5322,-5.0966,6.6779,0.17498,1
-4.0025,-13.4979,17.6772,-3.3202,1

-4.0173,-8.3123,12.4547,-1.4375,1
-3.0731,-0.53181,2.3877,0.77627,1
-1.979,3.2301,-1.3575,-2.5819,1
-0.4294,-0.14693,0.044265,-0.15605,1
-2.234,-7.0314,7.4936,0.61334,1
-4.211,-12.4736,14.9704,-1.3884,1
-3.8073,-8.0971,10.1772,0.65084,1
-2.5912,-0.10554,1.2798,1.0414,1
-2.2482,3.0915,-2.3969,-2.6711,1
-1.4427,3.2922,-1.9702,-3.4392,1
-0.39416,-0.020702,-0.066267,-0.44699,1
-1.522,-6.6383,5.7491,-0.10691,1
-2.8267,-9.0407,9.0694,-0.98233,1
-1.7263,-6.0237,5.2419,0.29524,1
-0.94255,0.039307,-0.24192,0.31593,1
-0.89569,3.0025,-3.6067,-3.4457,1
-6.2815,6.6651,0.52581,-7.0107,1
-2.3211,3.166,-1.0002,-2.7151,1
-1.3414,-2.0776,2.8093,0.60688,1
-2.258,-9.3263,9.3727,-0.85949,1
-3.8858,-12.8461,12.7957,-3.1353,1
-1.8969,-6.7893,5.2761,-0.32544,1
-0.52645,-0.24832,-0.45613,0.41938,1
0.0096613,3.5612,-4.407,-4.4103,1
-3.8826,4.898,-0.92311,-5.0801,1
-2.1405,-0.16762,1.321,-0.20906,1
-2.4824,-7.3046,6.839,-0.59053,1
-2.9098,-10.0712,8.4156,-1.9948,1
-0.60975,-4.002,1.8471,0.6017,1
0.83625,1.1071,-2.4706,-0.062945,1
0.60731,3.9544,-4.772,-4.4853,1
-4.8861,7.0542,-0.17252,-6.959,1
-3.1366,0.42212,2.6225,-0.064238,1
-2.5754,-5.6574,6.103,0.65214,1
-1.8782,-6.5865,4.8486,-0.021566,1
0.24261,0.57318,-1.9402,0.44007,1
1.296,4.2855,-4.8457,-2.9013,1
0.25943,5.0097,-5.0394,-6.3862,1
-5.873,9.1752,-0.27448,-6.0422,1
-3.4605,2.6901,0.16165,-1.0224,1
-2.3797,-1.4402,1.1273,0.16076,1
-1.2424,-1.7175,-0.52553,-0.21036,1
0.20216,1.9182,-3.2828,-0.61768,1
0.59823,3.5012,-3.9795,-1.7841,1
-0.77995,3.2322,-3.282,-3.1004,1
-4.1409,3.4619,-0.47841,-3.8879,1

-6.5084,8.7696,0.23191,-3.937,1
-4.4996,3.4288,0.56265,-1.1672,1
-3.3125,0.10139,0.55323,-0.2957,1
-1.9423,0.3766,-1.2898,-0.82458,1
-0.75793,2.5349,-3.0464,-1.2629,1
-0.95403,1.9824,-2.3163,-1.1957,1
-2.2173,1.4671,-0.72689,-1.1724,1
-2.799,1.9679,-0.42357,-2.1125,1
-1.8629,-0.84841,2.5377,0.097399,1
-3.5916,-6.2285,10.2389,-1.1543,1
-5.1216,-5.3118,10.3846,-1.0612,1
-3.2854,4.0372,-0.45356,-1.8228,1
-0.56877,1.4174,-1.4252,-1.1246,1
-2.3518,-4.8359,6.6479,-0.060358,1
-4.4861,-13.2889,17.3087,-3.2194,1
-4.3876,-7.7267,11.9655,-1.4543,1
-3.3604,-0.32696,2.1324,0.6017,1
-1.0112,2.9984,-1.1664,-1.6185,1
0.030219,-1.0512,1.4024,0.77369,1
-1.6514,-8.4985,9.1122,1.2379,1
-3.2692,-12.7406,15.5573,-0.14182,1
-2.5701,-6.8452,8.9999,2.1353,1
-1.3066,0.25244,0.7623,1.7758,1
-1.6637,3.2881,-2.2701,-2.2224,1
-0.55008,2.8659,-1.6488,-2.4319,1
0.21431,-0.69529,0.87711,0.29653,1
-0.77288,-7.4473,6.492,0.36119,1
-1.8391,-9.0883,9.2416,-0.10432,1
-0.63298,-5.1277,4.5624,1.4797,1
0.0040545,0.62905,-0.64121,0.75817,1
-0.28696,3.1784,-3.5767,-3.1896,1
-5.2406,6.6258,-0.19908,-6.8607,1
-1.4446,2.1438,-0.47241,-1.6677,1
-0.65767,-2.8018,3.7115,0.99739,1
-1.5449,-10.1498,9.6152,-1.2332,1
-2.8957,-12.0205,11.9149,-2.7552,1
-0.81479,-5.7381,4.3919,0.3211,1
0.50225,0.65388,-1.1793,0.39998,1
0.74521,3.6357,-4.4044,-4.1414,1
-2.9146,4.0537,-0.45699,-4.0327,1
-1.3907,-1.3781,2.3055,-0.021566,1
-1.786,-8.1157,7.0858,-1.2112,1
-1.7322,-9.2828,7.719,-1.7168,1
0.55298,-3.4619,1.7048,1.1008,1
2.031,1.852,-3.0121,0.003003,1
1.2279,4.0309,-4.6435,-3.9125,1

-4.2249,6.2699,0.15822,-5.5457,1
-2.5346,-0.77392,3.3602,0.00171,1
-1.749,-6.332,6.0987,0.14266,1
-0.539,-5.167,3.4399,0.052141,1
1.5631,0.89599,-1.9702,0.65472,1
2.3917,4.5565,-4.9888,-2.8987,1
0.89512,4.7738,-4.8431,-5.5909,1
-5.4808,8.1819,0.27818,-5.0323,1
-2.8833,1.7713,0.68946,-0.4638,1
-1.4174,-2.2535,1.518,0.61981,1
0.4283,-0.94981,-1.0731,0.3211,1
1.5904,2.2121,-3.1183,-0.11725,1
1.7425,3.6833,-4.0129,-1.7207,1
-0.23356,3.2405,-3.0669,-2.7784,1
-3.6227,3.9958,-0.35845,-3.9047,1
-6.1536,7.9295,0.61663,-3.2646,1
-3.9172,2.6652,0.78886,-0.7819,1
-2.2214,-0.23798,0.56008,0.05602,1
-0.49241,0.89392,-1.6283,-0.56854,1
0.26517,2.4066,-2.8416,-0.59958,1
-0.10234,1.8189,-2.2169,-0.56725,1
-1.6176,1.0926,-0.35502,-0.59958,1
-1.8448,1.254,0.27218,-1.0728,1
-1.2786,-2.4087,4.5735,0.47627,1
-2.902,-7.6563,11.8318,-0.84268,1
-4.3773,-5.5167,10.939,-0.4082,1
-2.0529,3.8385,-0.79544,-1.2138,1
0.18868,0.70148,-0.51182,0.0055892,1
-1.7279,-6.841,8.9494,0.68058,1
-3.3793,-13.7731,17.9274,-2.0323,1
-3.1273,-7.1121,11.3897,-0.083634,1
-2.121,-0.05588,1.949,1.353,1
-1.7697,3.4329,-1.2144,-2.3789,1
-0.0012852,0.13863,-
0.19651,0.0081754,1
-1.682,-6.8121,7.1398,1.3323,1
-3.4917,-12.1736,14.3689,-0.61639,1
-3.1158,-8.6289,10.4403,0.97153,1
-2.0891,-0.48422,1.704,1.7435,1
-1.6936,2.7852,-2.1835,-1.9276,1
-1.2846,3.2715,-1.7671,-3.2608,1
-0.092194,0.39315,-0.32846,-0.13794,1
-1.0292,-6.3879,5.5255,0.79955,1
-2.2083,-9.1069,8.9991,-0.28406,1
-1.0744,-6.3113,5.355,0.80472,1
-0.51003,-0.23591,0.020273,0.76334,1

-0.36372,3.0439,-3.4816,-2.7836,1
-6.3979,6.4479,1.0836,-6.6176,1
-2.2501,3.3129,-0.88369,-2.8974,1
-1.1859,-1.2519,2.2635,0.77239,1
-1.8076,-8.8131,8.7086,-0.21682,1
-3.3863,-12.9889,13.0545,-2.7202,1
-1.4106,-7.108,5.6454,0.31335,1
-0.21394,-0.68287,0.096532,1.1965,1
0.48797,3.5674,-4.3882,-3.8116,1
-3.8167,5.1401,-0.65063,-5.4306,1
-1.9555,0.20692,1.2473,-0.3707,1
-2.1786,-6.4479,6.0344,-0.20777,1
-2.3299,-9.9532,8.4756,-1.8733,1
0.0031201,-4.0061,1.7956,0.91722,1
1.3518,1.0595,-2.3437,0.39998,1
1.2309,3.8923,-4.8277,-4.0069,1
-5.0301,7.5032,-0.13396,-7.5034,1
-3.0799,0.60836,2.7039,-0.23751,1
-2.2987,-5.227,5.63,0.91722,1
-1.239,-6.541,4.8151,-0.033204,1
0.75896,0.29176,-1.6506,0.83834,1
1.6799,4.2068,-4.5398,-2.3931,1
0.63655,5.2022,-5.2159,-6.1211,1
-6.0598,9.2952,-0.43642,-6.3694,1
-3.518,2.8763,0.1548,-1.2086,1
-2.0336,-1.4092,1.1582,0.36507,1
-0.69745,-1.7672,-0.34474,-0.12372,1
0.75108,1.9161,-3.1098,-0.20518,1
0.84546,3.4826,-3.6307,-1.3961,1
-0.55648,3.2136,-3.3085,-2.7965,1
-3.6817,3.2239,-0.69347,-3.4004,1
-6.7526,8.8172,-0.061983,-3.725,1
-4.577,3.4515,0.66719,-0.94742,1
-2.9883,0.31245,0.45041,0.068951,1
-1.4781,0.14277,-1.1622,-0.48579,1
-0.46651,2.3383,-2.9812,-1.0431,1
-0.8734,1.6533,-2.1964,-0.78061,1
-2.1234,1.1815,-0.55552,-0.81165,1
-2.3142,2.0838,-0.46813,-1.6767,1
-1.4233,-0.98912,2.3586,0.39481,1
-3.0866,-6.6362,10.5405,-0.89182,1
-4.7331,-6.1789,11.388,-1.0741,1
-2.8829,3.8964,-0.1888,-1.1672,1
-0.036127,1.525,-1.4089,-0.76121,1
-1.7104,-4.778,6.2109,0.3974,1
-3.8203,-13.0551,16.9583,-2.3052,1

-3.7181,-8.5089,12.363,-0.95518,1
-2.899,-0.60424,2.6045,1.3776,1
-0.98193,2.7956,-1.2341,-1.5668,1
-0.17296,-1.1816,1.3818,0.7336,1
-1.9409,-8.6848,9.155,0.94049,1
-3.5713,-12.4922,14.8881,-0.47027,1
-2.9915,-6.6258,8.6521,1.8198,1
-1.8483,0.31038,0.77344,1.4189,1
-2.2677,3.2964,-2.2563,-2.4642,1
-0.50816,2.868,-1.8108,-2.2612,1
0.14329,-1.0885,1.0039,0.48791,1
-0.90784,-7.9026,6.7807,0.34179,1
-2.0042,-9.3676,9.3333,-0.10303,1
-0.93587,-5.1008,4.5367,1.3866,1
-0.40804,0.54214,-0.52725,0.6586,1
-0.8172,3.3812,-3.6684,-3.456,1
-4.8392,6.6755,-0.24278,-6.5775,1
-1.2792,2.1376,-0.47584,-1.3974,1
-0.66008,-3.226,3.8058,1.1836,1
-1.7713,-10.7665,10.2184,-1.0043,1
-3.0061,-12.2377,11.9552,-2.1603,1
-1.1022,-5.8395,4.5641,0.68705,1
0.11806,0.39108,-0.98223,0.42843,1
0.11686,3.735,-4.4379,-4.3741,1
-2.7264,3.9213,-0.49212,-3.6371,1
-1.2369,-1.6906,2.518,0.51636,1
-1.8439,-8.6475,7.6796,-0.66682,1
-1.8554,-9.6035,7.7764,-0.97716,1
0.16358,-3.3584,1.3749,1.3569,1
1.5077,1.9596,-3.0584,-0.12243,1
0.67886,4.1199,-4.569,-4.1414,1
-3.9934,5.8333,0.54723,-4.9379,1
-2.3898,-0.78427,3.0141,0.76205,1
-1.7976,-6.7686,6.6753,0.89912,1
-0.70867,-5.5602,4.0483,0.903,1
1.0194,1.1029,-2.3,0.59395,1
1.7875,4.78,-5.1362,-3.2362,1
0.27331,4.8773,-4.9194,-5.8198,1
-5.1661,8.0433,0.044265,-4.4983,1
-2.7028,1.6327,0.83598,-0.091393,1
-1.4904,-2.2183,1.6054,0.89394,1
-0.014902,-1.0243,-0.94024,0.64955,1
0.88992,2.2638,-3.1046,-0.11855,1
1.0637,3.6957,-4.1594,-1.9379,1
-0.8471,3.1329,-3.0112,-2.9388,1
-3.9594,4.0289,-0.35845,-3.8957,1

-5.8818,7.6584,0.5558,-2.9155,1
-3.7747,2.5162,0.83341,-0.30993,1
-2.4198,-0.24418,0.70146,0.41809,1
-0.83535,0.80494,-1.6411,-0.19225,1
-0.30432,2.6528,-2.7756,-0.65647,1
-0.60254,1.7237,-2.1501,-0.77027,1
-2.1059,1.1815,-0.53324,-0.82716,1
-2.0441,1.2271,0.18564,-1.091,1
-1.5621,-2.2121,4.2591,0.27972,1
-3.2305,-7.2135,11.6433,-0.94613,1
-4.8426,-4.9932,10.4052,-0.53104,1
-2.3147,3.6668,-0.6969,-1.2474,1
-0.11716,0.60422,-0.38587,-0.059065,1
-2.0066,-6.719,9.0162,0.099985,1
-3.6961,-13.6779,17.5795,-2.6181,1
-3.6012,-6.5389,10.5234,-0.48967,1
-2.6286,0.18002,1.7956,0.97282,1
-0.82601,2.9611,-1.2864,-1.4647,1
0.31803,-0.99326,1.0947,0.88619,1
-1.4454,-8.4385,8.8483,0.96894,1
-3.1423,-13.0365,15.6773,-0.66165,1
-2.5373,-6.959,8.8054,1.5289,1
-1.366,0.18416,0.90539,1.5806,1
-1.7064,3.3088,-2.2829,-2.1978,1
-0.41965,2.9094,-1.7859,-2.2069,1
0.37637,-0.82358,0.78543,0.74524,1
-0.55355,-7.9233,6.7156,0.74394,1
-1.6001,-9.5828,9.4044,0.081882,1
-0.37013,-5.554,4.7749,1.547,1
0.12126,0.22347,-0.47327,0.97024,1
-0.27068,3.2674,-3.5562,-3.0888,1
-5.119,6.6486,-0.049987,-6.5206,1
-1.3946,2.3134,-0.44499,-1.4905,1
-0.69879,-3.3771,4.1211,1.5043,1
-1.48,-10.5244,9.9176,-0.5026,1
-2.6649,-12.813,12.6689,-1.9082,1
-0.62684,-6.301,4.7843,1.106,1
0.518,0.25865,-0.84085,0.96118,1
0.64376,3.764,-4.4738,-4.0483,1
-2.9821,4.1986,-0.5898,-3.9642,1
-1.4628,-1.5706,2.4357,0.49826,1
-1.7101,-8.7903,7.9735,-0.45475,1
-1.5572,-9.8808,8.1088,-1.0806,1
0.74428,-3.7723,1.6131,1.5754,1
2.0177,1.7982,-2.9581,0.2099,1
1.164,3.913,-4.5544,-3.8672,1

-4.3667,6.0692,0.57208,-5.4668,1
-2.5919,-1.0553,3.8949,0.77757,1
-1.8046,-6.8141,6.7019,1.1681,1
-0.71868,-5.7154,3.8298,1.0233,1
1.4378,0.66837,-2.0267,1.0271,1
2.1943,4.5503,-4.976,-2.7254,1
0.7376,4.8525,-4.7986,-5.6659,1
-5.637,8.1261,0.13081,-5.0142,1
-3.0193,1.7775,0.73745,-0.45346,1
-1.6706,-2.09,1.584,0.71162,1
-0.1269,-1.1505,-0.95138,0.57843,1
1.2198,2.0982,-3.1954,0.12843,1
1.4501,3.6067,-4.0557,-1.5966,1
-0.40857,3.0977,-2.9607,-2.6892,1
-3.8952,3.8157,-0.31304,-3.8194,1
-6.3679,8.0102,0.4247,-3.2207,1
-4.1429,2.7749,0.68261,-0.71984,1
-2.6864,-0.097265,0.61663,0.061192,1
-1.0555,0.79459,-1.6968,-0.46768,1
-0.29858,2.4769,-2.9512,-0.66165,1
-0.49948,1.7734,-2.2469,-0.68104,1
-1.9881,0.99945,-0.28562,-0.70044,1
-1.9389,1.5706,0.045979,-1.122,1
-1.4375,-1.8624,4.026,0.55127,1
-3.1875,-7.5756,11.8678,-0.57889,1
-4.6765,-5.6636,10.969,-0.33449,1
-2.0285,3.8468,-0.63435,-1.175,1
0.26637,0.73252,-0.67891,0.03533,1
-1.7589,-6.4624,8.4773,0.31981,1
-3.5985,-13.6593,17.6052,-2.4927,1
-3.3582,-7.2404,11.4419,-0.57113,1
-2.3629,-0.10554,1.9336,1.1358,1
-2.1802,3.3791,-1.2256,-2.6621,1
-0.40951,-0.15521,0.060545,-0.088807,1
-2.2918,-7.257,7.9597,0.9211,1
-4.0214,-12.8006,15.6199,-0.95647,1
-3.3884,-8.215,10.3315,0.98187,1
-2.0046,-0.49457,1.333,1.6543,1
-1.7063,2.7956,-2.378,-2.3491,1
-1.6386,3.3584,-1.7302,-3.5646,1
-0.41645,0.32487,-0.33617,-0.36036,1
-1.5877,-6.6072,5.8022,0.31593,1
-2.5961,-9.349,9.7942,-0.28018,1
-1.5228,-6.4789,5.7568,0.87325,1
-0.53072,-0.097265,-0.21793,1.0426,1
-0.49081,2.8452,-3.6436,-3.1004,1

-6.5773,6.8017,0.85483,-7.5344,1
-2.4621,2.7645,-0.62578,-2.8573,1
-1.3995,-1.9162,2.5154,0.59912,1
-2.3221,-9.3304,9.233,-0.79871,1
-3.73,-12.9723,12.9817,-2.684,1
-1.6988,-7.1163,5.7902,0.16723,1
-0.26654,-0.64562,-0.42014,0.89136,1
0.33325,3.3108,-4.5081,-4.012,1
-4.2091,4.7283,-0.49126,-5.2159,1
-2.3142,-0.68494,1.9833,-0.44829,1
-2.4835,-7.4494,6.8964,-0.64484,1
-2.7611,-10.5099,9.0239,-1.9547,1
-0.36025,-4.449,2.1067,0.94308,1
1.0117,0.9022,-2.3506,0.42714,1
0.96708,3.8426,-4.9314,-4.1323,1
-5.2049,7.259,0.070827,-7.3004,1
-3.3203,-0.02691,2.9618,-0.44958,1
-2.565,-5.7899,6.0122,0.046968,1
-1.5951,-6.572,4.7689,-0.94354,1
0.7049,0.17174,-1.7859,0.36119,1
1.7331,3.9544,-4.7412,-2.5017,1
0.6818,4.8504,-5.2133,-6.1043,1
-6.3364,9.2848,0.014275,-6.7844,1
-3.8053,2.4273,0.6809,-1.0871,1
-2.1979,-2.1252,1.7151,0.45171,1
-0.87874,-2.2121,-0.051701,0.099985,1
0.74067,1.7299,-3.1963,-0.1457,1
0.98296,3.4226,-3.9692,-1.7116,1
-0.3489,3.1929,-3.4054,-3.1832,1
-3.8552,3.5219,-0.38415,-3.8608,1
-6.9599,8.9931,0.2182,-4.572,1
-4.7462,3.1205,1.075,-1.2966,1
-3.2051,-0.14279,0.97565,0.045675,1
-1.7549,-0.080711,-0.75774,-0.3707,1
-0.59587,2.4811,-2.8673,-0.89828,1
-0.89542,2.0279,-2.3652,-1.2746,1
-2.0754,1.2767,-0.64206,-1.2642,1
-3.2778,1.8023,0.1805,-2.3931,1
-2.2183,-1.254,2.9986,0.36378,1
-3.5895,-6.572,10.5251,-0.16381,1
-5.0477,-5.8023,11.244,-0.3901,1
-3.5741,3.944,-0.07912,-2.1203,1
-0.7351,1.7361,-1.4938,-1.1582,1
-2.2617,-4.7428,6.3489,0.11162,1
-4.244,-13.0634,17.1116,-2.8017,1
-4.0218,-8.304,12.555,-1.5099,1

-3.0201,-0.67253,2.7056,0.85774,1
-2.4941,3.5447,-1.3721,-2.8483,1
-0.83121,0.039307,0.05369,-0.23105,1
-2.5665,-6.8824,7.5416,0.70774,1
-4.4018,-12.9371,15.6559,-1.6806,1
-3.7573,-8.2916,10.3032,0.38059,1
-2.4725,-0.40145,1.4855,1.1189,1
-1.9725,2.8825,-2.3086,-2.3724,1
-2.0149,3.6874,-1.9385,-3.8918,1
-0.82053,0.65181,-0.48869,-0.52716,1
-1.7886,-6.3486,5.6154,0.42584,1
-2.9138,-9.4711,9.7668,-0.60216,1
-1.8343,-6.5907,5.6429,0.54998,1
-0.8734,-0.033118,-0.20165,0.55774,1
-0.70346,2.957,-3.5947,-3.1457,1
-6.7387,6.9879,0.67833,-7.5887,1
-2.7723,3.2777,-0.9351,-3.1457,1
-1.6641,-1.3678,1.997,0.52283,1
-2.4349,-9.2497,8.9922,-0.50001,1
-3.793,-12.7095,12.7957,-2.825,1
-1.9551,-6.9756,5.5383,-0.12889,1
-0.69078,-0.50077,-0.35417,0.47498,1
0.025013,3.3998,-4.4327,-4.2655,1
-4.3967,4.9601,-0.64892,-5.4719,1
-2.456,-0.24418,1.4041,-0.45863,1
-2.62,-6.8555,6.2169,-0.62285,1
-2.9662,-10.3257,8.784,-2.1138,1
-0.71494,-4.4448,2.2241,0.49826,1
0.6005,0.99945,-2.2126,0.097399,1
0.61652,3.8944,-4.7275,-4.3948,1
-5.4414,7.2363,0.10938,-7.5642,1
-3.5798,0.45937,2.3457,-0.45734,1
-2.7769,-5.6967,5.9179,0.37671,1
-1.8356,-6.7562,5.0585,-0.55044,1
0.30081,0.17381,-1.7542,0.48921,1
1.3403,4.1323,-4.7018,-2.5987,1
0.26877,4.987,-5.1508,-6.3913,1
-6.5235,9.6014,-0.25392,-6.9642,1
-4.0679,2.4955,0.79571,-1.1039,1
-2.564,-1.7051,1.5026,0.32757,1
-1.3414,-1.9162,-0.15538,-0.11984,1
0.23874,2.0879,-3.3522,-0.66553,1
0.6212,3.6771,-4.0771,-2.0711,1
-0.77848,3.4019,-3.4859,-3.5569,1
-4.1244,3.7909,-0.6532,-4.1802,1
-7.0421,9.2,0.25933,-4.6832,1

-4.9462,3.5716,0.82742,-1.4957,1
-3.5359,0.30417,0.6569,-0.2957,1
-2.0662,0.16967,-1.0054,-0.82975,1
-0.88728,2.808,-3.1432,-1.2035,1
-1.0941,2.3072,-2.5237,-1.4453,1
-2.4458,1.6285,-0.88541,-1.4802,1
-3.551,1.8955,0.1865,-2.4409,1
-2.2811,-0.85669,2.7185,0.044382,1
-3.6053,-5.974,10.0916,-0.82846,1
-5.0676,-5.1877,10.4266,-0.86725,1
-3.9204,4.0723,-0.23678,-2.1151,1
-1.1306,1.8458,-1.3575,-1.3806,1
-2.4561,-4.5566,6.4534,-0.056479,1
-4.4775,-13.0303,17.0834,-3.0345,1
-4.1958,-8.1819,12.1291,-1.6017,1
-3.38,-0.7077,2.5325,0.71808,1
-2.4365,3.6026,-1.4166,-2.8948,1
-0.77688,0.13036,-0.031137,-0.35389,1
-2.7083,-6.8266,7.5339,0.59007,1
-4.5531,-12.5854,15.4417,-1.4983,1
-3.8894,-7.8322,9.8208,0.47498,1
-2.5084,-0.22763,1.488,1.2069,1
-2.1652,3.0211,-2.4132,-2.4241,1
-1.8974,3.5074,-1.7842,-3.8491,1
-0.62043,0.5587,-0.38587,-0.66423,1
-1.8387,-6.301,5.6506,0.19567,1
-3,-9.1566,9.5766,-0.73018,1
-1.9116,-6.1603,5.606,0.48533,1
-1.005,0.084831,-0.2462,0.45688,1
-0.87834,3.257,-3.6778,-3.2944,1
-6.651,6.7934,0.68604,-7.5887,1
-2.5463,3.1101,-0.83228,-3.0358,1
-1.4377,-1.432,2.1144,0.42067,1
-2.4554,-9.0407,8.862,-0.86983,1
-3.9411,-12.8792,13.0597,-3.3125,1
-2.1241,-6.8969,5.5992,-0.47156,1
-0.74324,-0.32902,-0.42785,0.23317,1
-0.071503,3.7412,-4.5415,-4.2526,1
-4.2333,4.9166,-0.49212,-5.3207,1
-2.3675,-0.43663,1.692,-0.43018,1
-2.5526,-7.3625,6.9255,-0.66811,1
-3.0986,-10.4602,8.9717,-2.3427,1
-0.89809,-4.4862,2.2009,0.50731,1
0.56232,1.0015,-2.2726,-0.0060486,1
0.53936,3.8944,-4.8166,-4.3418,1
-5.3012,7.3915,0.029699,-7.3987,1

-3.3553,0.35591,2.6473,-0.37846,1
-2.7908,-5.7133,5.953,0.45946,1
-1.9983,-6.6072,4.8254,-0.41984,1
0.15423,0.11794,-1.6823,0.59524,1
1.208,4.0744,-4.7635,-2.6129,1
0.2952,4.8856,-5.149,-6.2323,1
-6.4247,9.5311,0.022844,-6.8517,1
-3.9933,2.6218,0.62863,-1.1595,1
-2.659,-1.6058,1.3647,0.16464,1
-1.4094,-2.1252,-0.10397,-0.19225,1
0.11032,1.9741,-3.3668,-0.65259,1
0.52374,3.644,-4.0746,-1.9909,1
-0.76794,3.4598,-3.4405,-3.4276,1
-3.9698,3.6812,-0.60008,-4.0133,1
-7.0364,9.2931,0.16594,-4.5396,1
-4.9447,3.3005,1.063,-1.444,1
-3.5933,0.22968,0.7126,-0.3332,1
-2.1674,0.12415,-1.0465,-0.86208,1
-0.9607,2.6963,-3.1226,-1.3121,1
-1.0802,2.1996,-2.5862,-1.2759,1
-2.3277,1.4381,-0.82114,-1.2862,1
-3.7244,1.9037,-0.035421,-2.5095,1
-2.5724,-0.95602,2.7073,-0.16639,1
-3.9297,-6.0816,10.0958,-1.0147,1
-5.2943,-5.1463,10.3332,-1.1181,1
-3.8953,4.0392,-0.3019,-2.1836,1
-1.2244,1.7485,-1.4801,-1.4181,1
-2.6406,-4.4159,5.983,-0.13924,1
-4.6338,-12.7509,16.7166,-3.2168,1
-4.2887,-7.8633,11.8387,-1.8978,1
-3.3458,-0.50491,2.6328,0.53705,1
-1.1188,3.3357,-1.3455,-1.9573,1
0.55939,-0.3104,0.18307,0.44653,1
-1.5078,-7.3191,7.8981,1.2289,1
-3.506,-12.5667,15.1606,-0.75216,1
-2.9498,-8.273,10.2646,1.1629,1
-1.6029,-0.38903,1.62,1.9103,1
-1.2667,2.8183,-2.426,-1.8862,1
-0.49281,3.0605,-1.8356,-2.834,1
0.66365,-0.045533,-0.18794,0.23447,1
-0.72068,-6.7583,5.8408,0.62369,1
-1.9966,-9.5001,9.682,-0.12889,1
-0.97325,-6.4168,5.6026,1.0323,1
-0.025314,-0.17383,-0.11339,1.2198,1
0.062525,2.9301,-3.5467,-2.6737,1
-5.525,6.3258,0.89768,-6.6241,1

-1.2943,2.6735,-0.84085,-2.0323,1
-0.24037,-1.7837,2.135,1.2418,1
-1.3968,-9.6698,9.4652,-0.34872,1
-2.9672,-13.2869,13.4727,-2.6271,1
-1.1005,-7.2508,6.0139,0.36895,1
0.22432,-0.52147,-0.40386,1.2017,1
0.90407,3.3708,-4.4987,-3.6965,1
-2.8619,4.5193,-0.58123,-4.2629,1
-1.0833,-0.31247,1.2815,0.41291,1
-1.5681,-7.2446,6.5537,-0.1276,1
-2.0545,-10.8679,9.4926,-1.4116,1
0.2346,-4.5152,2.1195,1.4448,1
1.581,0.86909,-2.3138,0.82412,1
1.5514,3.8013,-4.9143,-3.7483,1
-4.1479,7.1225,-0.083404,-6.4172,1
-2.2625,-0.099335,2.8127,0.48662,1
-1.7479,-5.823,5.8699,1.212,1
-0.95923,-6.7128,4.9857,0.32886,1
1.3451,0.23589,-1.8785,1.3258,1
2.2279,4.0951,-4.8037,-2.1112,1
1.2572,4.8731,-5.2861,-5.8741,1
-5.3857,9.1214,-0.41929,-5.9181,1
-2.9786,2.3445,0.52667,-0.40173,1
-1.5851,-2.1562,1.7082,0.9017,1
-0.21888,-2.2038,-0.0954,0.56421,1
1.3183,1.9017,-3.3111,0.065071,1
1.4896,3.4288,-4.0309,-1.4259,1
0.11592,3.2219,-3.4302,-2.8457,1
-3.3924,3.3564,-0.72004,-3.5233,1
-6.1632,8.7096,-0.21621,-3.6345,1
-4.0786,2.9239,0.87026,-0.65389,1
-2.5899,-0.3911,0.93452,0.42972,1
-1.0116,-0.19038,-0.90597,0.003003,1
0.066129,2.4914,-2.9401,-0.62156,1
-0.24745,1.9368,-2.4697,-0.80518,1
-1.5732,1.0636,-0.71232,-0.8388,1
-2.1668,1.5933,0.045122,-1.678,1
-1.1667,-1.4237,2.9241,0.66119,1
-2.8391,-6.63,10.4849,-0.42113,1
-4.5046,-5.8126,10.8867,-0.52846,1
-2.41,3.7433,-0.40215,-1.2953,1
0.40614,1.3492,-1.4501,-0.55949,1
-1.3887,-4.8773,6.4774,0.34179,1
-3.7503,-13.4586,17.5932,-2.7771,1
-3.5637,-8.3827,12.393,-1.2823,1
-2.5419,-0.65804,2.6842,1.1952,1

ÖZGEÇMİŞ

Kişisel Bilgiler:

Adı Soyadı: Batuhan BİLENLER
Doğum Tarihi: 08.12.1992
Doğum Yeri: Malatya
E-mail: bilenlerbatuhan@gmail.com

Eğitim:

2011-2016 Bilgisayar Mühendisliği Lisans Programı, İstanbul Üniversitesi, Türkiye

2018-2020 Bilgisayar Mühendisliği Yüksek Lisans (Tezli) Programı, İstanbul Aydın Üniversitesi, Türkiye

İş Deneyimi:

2016-2017 Yazılım Geliştirme Mühendisi, Türkiye Sınai Kalkınma Bankası

2017-2019 İş Analisti, Ankara Sigorta

2019- .. Veri Bilimi Yöneticisi, Halk Sigorta

Yayınlar:

1- "BANKALARIN SAHTE PARA DENETİMİNDE KULLANDIKLARI KARAR AĞAÇLARINDA SONUÇ İYİLEŞTİRİCİ YÖNTEMLER", TIDE Academia Research, 2.cilt 1.sayı