Breast Cancer Data Seti Algoritma Uygulamaları

Support Vector Machine ve Multi-Layer Perceptron

Canan SERPER

Bilişim Sistemleri Mühendisliği

Kocaeli Üniversitesi

serpercanan@gmail.com

***Abstract*— Breast cancer is prevalent among women and develops from breast tissue. Early diagnosis and accurate treatment is vital to increase the rate of survival. Identification of genetic factors with microarray technology can make significant contributions to diagnosis and treatment process. In this study, several machine learning algorithms are used for Diagnosis of Breast Cancer and their classification performances are compared with each other. In addition, the active genes in breast cancer are identified by attribute selection methods and the conducted study show success rate 90,72 % with 139 feature.**

Keywords— Breast cancer, dataset, classification, algorithms

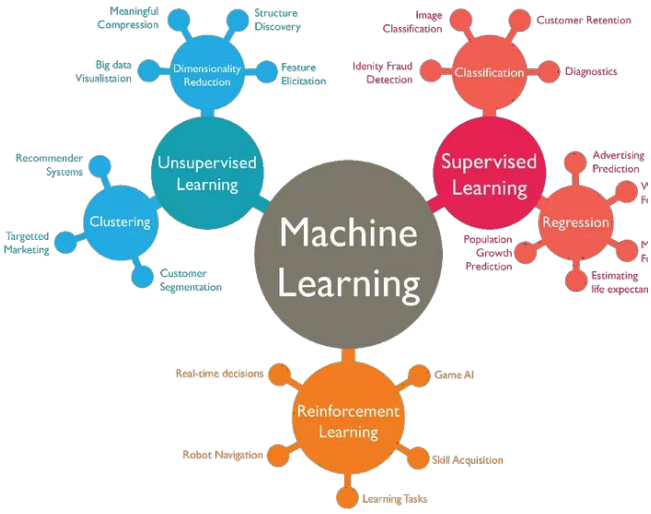
***Giriş*—Meme kanseri, sıklıkla kadınlar arasında görülen ve meme hücrelerinde başlayan bir kanser türüdür. Erken teşhis ve doğru tedavi meme kanseri hastalarının hayatta kalma oranını arttırabilme açısından son derece önemlidir. Mikrodizi teknolojisi ile hastalığın genetik faktörlerinin belirlenmesi teşhis ve tedavi sürecine önemli katkı sağlayabilir. Bu çalışmada, meme kanserinin teşhisi için birçok makine öğrenmesi algoritmaları kullanılmıştır ve bu algoritmaların birbirlerine göre sınıflandırma performansları karşılaştırılmıştır. Bunun yanı sıra öznitelik seçme yöntemleri ile meme kanserinde etkin genler belirlenip yapılan çalışma sonucunda 139 öznitelik ile % 90,72 başarı elde edilmiştir..**

Anahtar kelimeler— makine öğrenmesi, meme kanseri teşhisi, sınıflandırma, algoritmalar

1-) Makine Öğrenmesi Nedir?

Makine öğrenimi **(**ML – Machine Learning**),** yazılım programlarının açık bir şekilde programlanmadan sonuçları tahmin etmede daha doğru olmasını sağlayan bir algoritma kategorisidir. *Makine öğrenmesinin temel dayanağı*, giriş verisini alabilen algoritmalar oluşturmak ve çıktıları yeni veriler ortaya çıktıkça güncellerken bir çıktıyı tahmin etmek için istatistiksel analiz kullanmaktır.

Makine öğreniminde yer alan süreçler, *veri madenciliği* ve *tahmin modellemesi*ne benzerdir. Her ikisi de, kalıpları aramak ve program eylemlerini buna göre ayarlamak için veri aramayı gerektirir. Birçok kişi, internetten alışveriş yapmaktan ve satın alma işlemleriyle ilgili reklamlar yayınlamaktan makine öğrenimini bilmektedir. Bunun nedeni, öneri motorlarının neredeyse gerçek zamanlı olarak çevrimiçi reklam yayınını kişiselleştirmek için makine öğrenimini kullanmasıdır. Kişiselleştirilmiş pazarlamanın ötesinde, diğer yaygın makine öğrenimi kullanım durumları arasında *sahtekârlık tespiti*, *spam filtreleme*, *ağ güvenliği tehdit algılama*, *tahmini bakım* ve *bina haber beslemeleri* yer almaktadır.



Şekil1.1 Mechine Learning Uygulamalrı Şeması

A) Makine Öğrenimi Nasıl Çalışır?

Makine öğrenimi algoritmaları genellikle denetlenen veya denetlenmeyen olarak kategorize edilir. Algılanan algoritmalar, *algoritma eğitimi* sırasında tahminlerin doğruluğu hakkında geri bildirim sağlamanın yanı sıra, hem girdi hem de istenen çıktıyı sağlamak için makine öğrenim becerileri ile bir veri bilimcisi veya veri analisti gerektirir. Veri bilimcileri, modelin hangi değişkenleri veya özellikleri analiz edeceğini ve tahminleri geliştirmek için kullanacağını belirler. Eğitim tamamlandığında, algoritma öğrenilenleri yeni verilere uygulayacaktır.

Denetlenmeyen algoritmaların istenen sonuç verileriyle eğitilmesi gerekmez. Bunun yerine, verileri gözden geçirmek ve sonuçlara varmak için derin öğrenme denen yinelemeli bir yaklaşım kullanırlar. Denetimsiz öğrenme algoritmaları (aynı zamanda sinir ağları olarak da adlandırılır) görüntü tanıma, konuşma-metin ve doğal dil üretimi de dâhil olmak üzere denetimli öğrenme sistemlerinden daha karmaşık işlem görevleri için kullanılır. Bu sinir ağları, milyonlarca eğitim verisi örneğini tarayarak ve birçok değişken arasında sıklıkla ince korelasyonları otomatik olarak tespit ederek çalışır. Eğitildikten sonra, algoritma yeni verileri yorumlamak için veri bankasını kullanabilir. Bu algoritmalar, büyük miktarlarda eğitim verisi gerektirdiğinden, büyük veri çağında ancak uygulanabilir hale gelmiştir.

B)Makine Öğrenmesi Süreci

Algoritmaların öğrenme sürecinde öncelikle ham veriler bulunur ve algoritmaya iletilir. Bu veriler genel olarak sayısal ölçümler, puanlamalar, metin gibi bilgilerdir.

Ardından sırası ile şu adımlar atılır:

- Özellik Çıkarımı
- Model Oluşturma
- Değerlendirme
- Tahmin

Değerlendirme süreci oldukça önemlidir ve başarılı sonuçlar elde edilebilmesi için gereklidir. Bu aşamada;

- Accuracy (Doğruluk)
- Sensitivity (Hassaslık)
- Specificity (Belirginlik) gibi testler gerçekleştirilir ve yapılan çıkarımların en iyisi olduğu kanıtlanır.

Şayet değerlendirme sonucunda istenilen sonuçlara ulaşılamaz ve modelin başarılı olmadığı düşünülürse iyileştirme yapılır. Bazı eklemeler yapılarak yeniden model oluşturulur ve oluşturulan model tekrar testlerden geçirilir. Bu süreç istenilen sonuç elde edilene kadar devam edebilir.

Makine ögrenmesi fikrinin temelinde, bilgisayarların sadece kendisine verilen görevleri yapan vekomutları isleyen bir cihaz olmasından ziyade insan gibi ögrenebilen sistemler olması yatmaktadır. Insanlarda ögrenme süreci hayatları boyunca gerek ebeveynleri, gerek ögretmenleri, gerekse yasadıkları sosyal çevrede başkalarıyla etkilesime girmesiyle gerçeklesir.

C)Makine Öğrenimi Algoritma Türleri

Makine Öğrenmesi'nin neredeyse sınırsız kullanımı olduğu gibi, makine öğrenimi algoritmalarında sıkıntı yoktur. Oldukça basit olandan son derece karmaşık olanlara kadar değişir. En yaygın kullanılan modellerden birkaçı:

- Bu makine öğrenme algoritması sınıfı, genellikle iki değişken arasında bir korelasyonun tanımlanmasını ve gelecekteki veri noktaları hakkında tahminler yapmak için bu korelasyonun kullanılmasını içerir.
- **Karar ağaçları**: Bu modeller belirli eylemler hakkındaki gözlemleri kullanır ve istenen sonuca varmak için en uygun yolu belirler.
- **K-küme**: Bu model, belirli sayıda veri noktasını benzer özelliklere dayanan belirli sayıda gruplandırmaya dâhil eder.
- **Nöral ağlar**: Bu derin öğrenme modelleri, gelecekteki veriyi işlemeyi öğrenmek için birçok

değişken arasındaki korelasyonları tanımlamak için büyük miktarda eğitim verisi kullanır.

- **Takviye öğrenme**: Bu derin öğrenme alanı, bir süreci tamamlamak için birçok girişimde yinelenen modelleri içerir. Elverişli sonuçlar üreten adımlar ödüllendirilir ve algoritma en uygun süreci öğrenene kadar istenmeyen sonuçlara yol açan adımlar cezalandırılır.

II. Breast Cancer veri Seti

Özellikler, bir göğüs kütlesinin ince iğne aspiratının (FNA) sayısallaştırılmış bir görüntüsünden hesaplanır.Çalışmada kullanılan göğüs kanseri verisi 32 adet özellik , 1 adet sınıf bilgisi içeren bir veri setidir.

Bağımlı Değişkenler
1) Kimlik numarası
2) Teşhis (M = malign, B = iyi huylu)
3)32 Özellik

	Eğitim Kümesi		Test Kümesi		Toplam	
Sınıf	Gözlem	Yüzde	Gözlem	Yüzde	Gözlem	Yüzde
İyi Huylu	394	0,66	64	0,61	458	0,66
Kötü Huylu	200	0,34	41	0,39	241	0,34
Toplam	594		105		699	

Şekil 3. Veri Setine ait Dağılımlar

I. Kullanılan Alggoritmalar

A. Support Vector Machine (SVM)

Destek Vektör Makinesi" (SVM), sınıflandırma veya regresyon problemleri için kullanılabilen denetimli bir makine öğrenmesi algoritmasıdır. Bununla birlikte, çoğunlukla sınıflandırma problemlerinde kullanılır. Bu algoritmada, her bir veri maddesini belirli bir koordinatın değeri olan her özelliğin değeri ile birlikte n-boyutlu boşluğa (burada n sahip olduğunuz özelliklerin sayısı) bir nokta olarak çizilir. Ardından, iki sınıftan oldukça iyi ayrım yapan hiper-düzlemi bularak sınıflandırma gerçekleştirilir.

Destek Vektör Makineleri, veri setinin doğrusal olarak ayrılabilme ve ayrılamama durumuna göre ikiye ayrılmaktadır.

A. Doğrusal Destek Vektör Makineleri

Destek vektör makineleriyle sınıflandırmada, iki sınıfa ait örneklerin doğrusal olarak dağıldığını varsayalım. Bu durumda bu iki sınıfın, eğitim verisi kullanılarak elde edilen bir karar fonksiyonu yardımıyla birbirinden ayrılması amaçlanır.

Veri setini ikiye ayıran doğru karar doğrusu olarak isimlendirilmektedir. Sonsuz tane karar doğrusu çizebilme imkanı mevcut olsada önemli olan optimal yani en uygun karar doğrusunu belirlemektir.

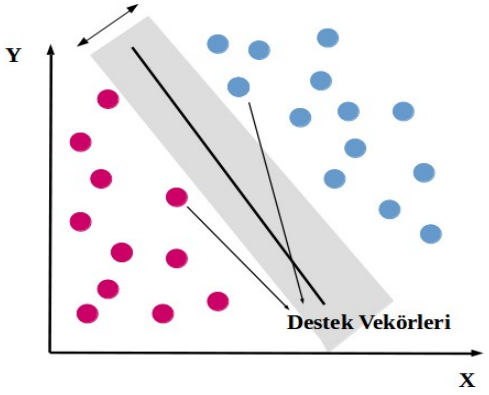
Karar doğrusunun yeni katılacak olan veriye karşı dayanıklı olabilmesi için sınır çizgisinin, iki sınıfın sınır çizgisilerine en yakın uzaklıkta olması gerekmektedir. Bu sınır çizgisine en yakın noktalar, destek noktaları olarak adlandırılmaktadır.

B . Doğrusal Olmayan Destek Vektör Makineleri

Doğrusal olarak dağılan örneklerde Destek Vektör Makineleri'nin nasıl çalıştığını ve bunları Python'da nasıl uygulayabileceğimizi konuştuktan sonra sıra doğrusal olmayanlara geldi. Doğrusal olmayan bir veri kümesinde DVM'ler doğrusal bir hiper-düzlem çizemez. Bu nedenle çekirdek numarası olarak adlandırılan *kernel trick*'ler kullanılır. Çekirdek yöntemi, doğrusal olmayan verilerde makine öğrenimini yüksek oranda arttırmaktadır.

En çok kullanılan çekirdek yöntemleri:

- Polynomial Kernel
- Gaussian RBF (Radial Basis Function) Kernel



Şekil 4.SVM Modellemesi

Avantajları:

- Yüksek boyutlu uzaylarda etkilidirler.
- Boyut sayısının, örnemlem sayısından fazla olduğu durumlarda etkilidirler.

- Karar fonksiyonunda bir takım eğitim noktaları kullanılır (“support vectors”). Dolayısıyla bellek verimli bir şekilde kullanılmış olur.

- Çok yönlü: Karar fonksiyonu için çok farklı çekirdek fonksiyonları (“kernel functions”) kullanılabilmektedir.

Dezavantajları:

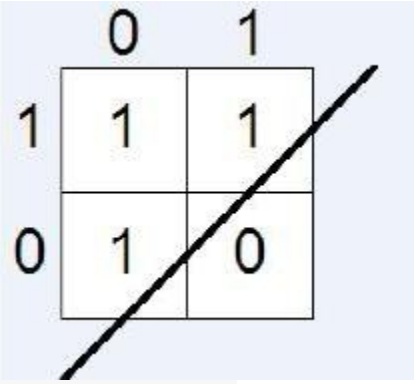
- Olasılıksal Tahmin üretememe
- Çekirdek Fonksiyonlar için mercer koşulu zorunluluğu

SVM’in kullanım alanlarına bakacak olursak:

- SVM’ler, hem standart endüktif hem de iletken ayarlarda etiketli eğitim örneklerine olan ihtiyacı önemli ölçüde azaltabileceğinden, metin ve köprü metni sınıflandırmada yardımcı olur.
- Görüntülerin sınıflandırılması da SVM’leri kullanarak yapılabilir. Deneysel sonuçlar, SVM’lerin geleneksel arama sorgulama şemalarına göre sadece üç veya dört arama sonuçları geri bildiriminden sonra önemli ölçüde daha yüksek bir arama doğruluğu sağladığını göstermektedir. Vapnik tarafından öne sürülen ayrıcalıklı yaklaşımı kullanan SVM modifiye edilmiş bir sürümünü kullananlar da dahil olmak üzere, bu, görüntü bölümlendirme sistemleri için de geçerlidir.
- Elle yazılmış karakterler SVM kullanılarak tanınabilir .
- SVM algoritması biyolojik ve diğer bilim dallarında yaygın olarak uygulanmıştır. Doğru sınıflandırılmış bileşiklerin% 90’ına kadar proteinleri sınıflandırmak için kullanılmıştır. SVM modellerinin yorumlanması için bir mekanizma olarak SVM ağırlıklarına dayanan permütasyon testleri önerilmiştir. Destek vektör makinesi ağırlıkları geçmişte SVM modellerini yorumlamak için kullanılmıştır.
- Tahmin yapmak için model tarafından kullanılan özellikleri tanımlamak için destek vektör makinesi modellerinin Posthoc yorumu, biyolojik bilimlerde özel bir önemi olan nispeten yeni bir araştırma alanıdır.

B. Multi-Layer Perceptron

Çok Katmanlı Algılayıcılar (MLP) XOR Problemi’ni çözmek için yapılan çalışmalar sonucu ortaya çıkmıştır.Bunun için öncelikle XOR problemşnş tanımak gerekir

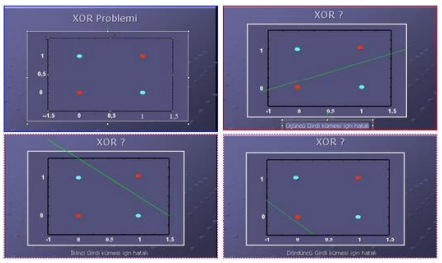


Şekil .5 Linear Seperability

A.XOR Problemi

Bir düzlemde sadece bir hat varsa iki sınıfa ait iki boyutlu örüntülerin bir kümesi doğrusal olarak ayrılabilirdir Verilen düzlemdeki örneklerin hepsi iki gruptan birine dahil olmuştur. Yani problem çözülebilir. Konuya örnek olarak and ve or fonksiyonları verilebilir. Or fonksiyonu dogrusal ayrılabilir. Eğer bir problem doğrusal ayrılabilir ise o zaman Perceptron Öğrenimi ile örüntülerin bir kümesinden ağırlıklar elde edilebilir.

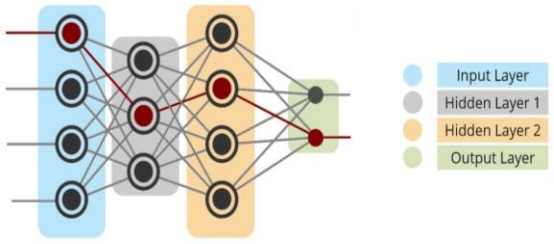
Xor fonksiyonu doğrusal ayrılabilir değildir. Düzlemdeki çıktıları tek bir hatla ikiye bölemiyoruz. Perceptronlar XOR Problemi gibi doğrusal olarak sınıflandırılamayan problemleri çözümünde başarısızdır. XOR Problemi’ni çözmek için geriye yayılımlı çok katmanlı ağlardan faydalanılabilir.



Şekil 6..Xor Modeli

Rumelhart ve arkadaşları tarafından geliştirilen bu modeli ‘Back Propogation Model’ yada hatayı ağa yaydığı için ‘Hata Yayma Modeli’ de denmektedir. Delta Öğrenme Kuralı denilen bir öğrenme metodu kullanır. MLP özellikle sınıflandırma ve genelleme yapma durumlarında etkin çalışır. Çok Katmanlı Ağ’ların yapısı aşağıdaki gibidir.

Birçok giriş için bir nöron yeterli olmayabilir. Paralel işlem yapan birden fazla nörona ihtiyaç duyulduğunda katman kavramı devreye girer. Görüldüğü üzere Single Perceptron Model'den farklı olarak arada gizli(hidden) katman bulunmaktadır. Giriş katmanı gelen verileri alarak ara katmana gönderir. Gelen bilgiler bir sonraki katmana aktarılırlar. Ara katman sayısı en az bir olmak üzere probleme göre değişir ve ihtiyaca göre ayarlanır. Her katmanın çıkışı bir sonraki katmanın girişi olmaktadır. Böylelikle çıkışa ulaşılmaktadır. Her işlem elemanı yani nöron bir sonraki katmanda bulunan bütün nöronlara bağlıdır. Ayrıca katmandaki nöron sayısı da probleme göre belirlenir. Çıkış katmanı önceki katmanlardan gelen verileri işleyerek ağın çıkışını belirler. Sistemin çıkış sayısı çıkış katmanında bulunan eleman sayısına eşittir. Single Perceptron Modeli incelerken bahsettiğimiz nöron yapısı burada aynen geçerlidir.



Şekil 7.MLP Şeması

Girdi Düğümleri: Giriş düğümleri dış dünyadan ağa bilgi sağlar ve birlikte "Giriş Katmanı" olarak adlandırılır. Giriş düğümlerinden hiçbirinde hesaplama yapılmaz – bunlar sadece gizli düğümlere bilgi aktarır.
Gizli Düğümler: Gizli düğümlerin dış dünyayla doğrudan bağlantısı yoktur(bundan dolayı "gizli" adı verilir.). Hesaplamalar yaparlar ve girdi düğümlerinden çıkış düğümlerine bilgi aktarırlar. Gizli düğümlerden oluşan bir koleksiyon "gizli katman" oluşturur. Bir ağ sadece tek bir giriş katmanına ve tek bir çıktı katmanına sahipken, sıfır veya çoklu gizli katmanlara sahip olabilir. Çok Katmanlı bir Perceptron'un bir veya daha fazla gizli katmanı vardır.
Çıktı Düğümleri: Çıktı düğümleri toplu olarak "Çıktı Katmanı" olarak adlandırılır ve bilgi işlemlerden ve ağdan dış dünyaya bilgi aktarımından sorumludur.

II. SUPPORT VECTOR MACHINE ALGORITMASININ ANALIZI

A. Datasetin Yüklenmesi ve Sınıf Etiketinin Gösterilmesi

```
"""
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
# %%
data = pd.read_csv("data.csv")

```

Pandas, Numpy'ın sütun adları ve homojen olmayan verilerle çalışamama gibi eksik kaldığı kısımlara ve daha fazlasına çözümler üretir. Pandas ile veri analizi yaparken kullanacağımız temel veri yapıları Seriler ve DataFrame'lerdir.

Pandas serilerini Numpy dizileri ile karşılaştırabilmek için Numpy kütüphanesini de import etmamiz gerek.İmport ettikten sonra data.csv dataseti yüklenmiştir.

B. Etiketlerin Vektöre Dönüştürülmesi

M = data[data.diagnosis == "M"]
B = data[data.diagnosis == "B"]

Data değerine atanılan veriseti içerisinden M ve B degerli verilerin ataması yapılır.

```
# %%
data.diagnosis = [1 if each == "M" else 0 for each in data.diagnosis]
y = data.diagnosis.values
x_data = data.drop(["diagnosis"],axis=1)

```

C. Normalizasyon

Normalleştirmeye ihtiyaç duymamızdaki en büyük sebep farklı niteliklerin farklı ölçü birimleri ile ölçülmesinden kaynaklanmaktadır. Makine öğrenmesinde parametrik olmayan yöntemlerin çoğu uzaklık tabanlı çalışmaktadır. Yani karar verirken iki niteliğin düzlemde birbirine göre olan uzaklıklığına göre karar vermektedirler.

```
# %%
# normalization
x = (x_data - np.min(x_data))/(np.max(x_data)-np.min(x_data))
#%%
```

D. Dataseti Train ve Test Olarak Ayırma İşlemi

Bu işlem dataset'deki ne kadar verinin test işlemi için ayrılacağını belirler.

```
#%%
# train test split
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.2,random_state=
#%% SVM
```

E. SVM Modelini Oluşturmak ve Sonuç Değerini Alma

Scikit-learn kütüphanesi svm modülü SVC sınıfından oluşturacağımız classifier nesnesi ile modeli oluşturuyoruz.

```
from sklearn.svm import SVC
svc=SVC(kernel='poly')
model_poly=svc.fit(x_train, y_train)

y_prediction=model_poly.predict(x_test)

#Başarı oranı
from sklearn.metrics import accuracy_score
accuracy_poly = accuracy_score(y_test, y_prediction)
print('Accuracy_Poly:', accuracy_poly)

from sklearn.svm import SVC
svc=SVC(kernel='linear')
model_linear=svc.fit(x_train, y_train)

y_prediction=model_linear.predict(x_test)

#Başarı oranı
from sklearn.metrics import accuracy_score
accuracy_linear = accuracy_score(y_test, y_prediction)
print('Accuracy_Linear:', accuracy_linear)

from sklearn.svm import SVC
svc=SVC(kernel='rbf')
model_rbf=svc.fit(x_train, y_train)

y_prediction=model_rbf.predict(x_test)

#Başarı oranı
from sklearn.metrics import accuracy_score
accuracy_rbf = accuracy_score(y_test, y_prediction)
print('Accuracy_Rbf:', accuracy_rbf)
#%% test
```

Makinemizi svc.fit metoduna x_train ve y_train değerleri vererek eğittik. Result değişkenine test veri kümemizi vererek confusion matrix ve başarı oranını yazdırdık.Başarı oranlarını kernel değerlerine poly,linear,rbf vererek ayrı ayrı sonuç değerlerini yazdırdım.

D. Başarı Oranları

Kernel Fonksiyonu	Accuary Text Size		
	0.2	0.5	0.7
Poly	0.98	0.98	0.96
Linear	0.96	0.96	0.95
Rfb	0.96	0.98	0.95

En yüksek sonuç değerini poly fonk 0.96 olarak sonuç vermiştir.

III. MULTI LAYER PERCEPTRON ALGORITMASININ ANALIZI

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv('data.csv')
print(dataset.head())
print(dataset.shape)

def dataSetAnalysis(df):
```

Algoritmayı uygulamaya datasetini ve kütüphaneleri import ederek başlandı.Daha önce Spyder kullanırken şuan daha hızlı ve fonksiyonel bulut hizmeti sağlayan codlab kullanılmaya başlandı.

A. Özelliklerin ve Sınıf Etiketlerinin Ayrılması

Python'da sütunların indeks değeri 2'dan başladığından ve veri setinde 5 sütun olduğundan ilk niteliğin indeksi 24, son niteliğin 24 olacaktır. iloc[] ile sütun seçerken ilk değer dahil ikinci değer hariçtir. Bu sebeple nitelikler matrisini (X) seçerken 2'yi dahil, 24'ü hariç tutacağız. Çünkü son indeks değeri label yani hedef değişkene aittir. iloc[] içindeki İlk iki nokta (:) tüm satırları ifade ederken virgül sonrası, sütunlardan seçilecek filtreyi ifade eder. 2:24 demek 2 dahil 24 hariç indeksli sütunları seç demek.

```
dataSetAnalysis(dataset)
X = dataset.iloc[:,2:32] # [all rows, col from index 2 to the last one excluding 'Un
y = dataset.iloc[:,1] # [all rows, col one only which contains the classes of cancer
print(X)
print(y)

from sklearn.preprocessing import LabelEncoder

print("Before encoding: ")
print(y[100:110])
```

Veri Setinde iyi ve kötü huylu tümör verileri M ve B şeklinde yer almaktadır.Y değişkenine bu M ve B değerlerinin atamasını yaptım.X değişkenine M ve B değerleri hariç tüm özellik sütunları alarak atamasını yaptım.

B. Encoding İşlemi

Daha önce Y'ye alınan sınıf etiketleri M ve B olmak üzere harflerden oluşmaktadır. Harfleri 1 ve 0 dan oluşan vektöre dönüştürmek için **preprocessing ()** modülündeki **LabelEncoder ()**

sınıfını kullanarak sınıf etiketlerine encoding işlemi yapılır.

C. Dataseti Train ve Test Olarak Ayırma İşlemi

Bu işlem dataset'deki ne kadar verinin test işlemi için ayrılacağını belirler.

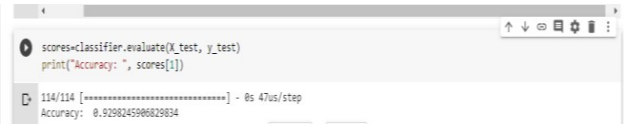
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_st

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)


from keras.layers import Dense,Dropout
from keras.models import Sequential

classifier = Sequential()
```

StandardScaler(); özellik ölçeklendirme fonksiyonunu kullanarak; datasette yer alan virgüllü sayıları (0.17 vb) hiçbir özelliğin diğerine baskın gelmemesi adına 0 ve 1 arasında yapılandırır.

D. Modelin Oluşturulması

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)


from keras.layers import Dense,Dropout
from keras.models import Sequential

classifier = Sequential()

classifier.add(Dense(units = 16, kernel_initializer = 'uniform', activation = 'tanh', in
classifier.add(Dropout(0.5))
classifier.add(Dense(units = 16, kernel_initializer = 'uniform', activation = 'tanh'))
classifier.add(Dropout(0.5))
classifier.add(Dense(units = 16, kernel_initializer = 'uniform', activation = 'tanh'))
classifier.add(Dropout(0.5))
classifier.add(Dense(units = 1, kernel_initializer = 'uniform', activation = 'sigmoid'))

classifier.compile(optimizer = 'rmsprop', loss = 'binary_crossentropy', metrics = ['accu

egitim=classifier.fit(X_train, y_train, batch_size = 64, epochs = 10    , verbose=1, shu

scores=classifier.evaluate(X_test, y_test)
print("Accuracy: ", scores[1])

#MLP Accuracy Grafiği;
from matplotlib import pyplot as plt
import numpy as np
plt.figure(figsize=(17,3))
plt.subplot(1, 2, 1)
plt.plot(egitim.history['accuracy'])
plt.plot(egitim.history['val_accuracy'])
plt.title('Classifier Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(['Train', 'Test'], loc='upper left')


plt.subplot(1, 2, 2)
```

Classifier nesnesinin add() metodu içine Dense sınıfını nesne yaratarak parametre veriyoruz. Dense içindeki ilk



parametre kernel_initializer (eski init): Başlangıç ağırlıklarını belirler. Tensor'ları oluşturmak/başlatmak için uniform dağılımını
temsilen uniform parametresini
kullanıyoruz. input_dim: Yapay sinir ağı nesnesine ilk katman ekleme işlemi yapılırken mutlaka girdi katmanında kaç düğüm olacağı bildirilir. Aksi halde ilk gizli katman kendisine kaç düğümden bağlantı olacağını bilemez. Bu rakam nitelikler matrisindeki sütun sayısıdır,

E.

F. Modelin Eğitme

Yapay sinir ağı nesnemizi (classifier) eğitebiliriz. Ancak bundan önce son olarak onu derlememiz gerekiyor. Bunun için compile() metodunu kullanacağız. Tahmin y ile gerçek ye değeri arasını hesaplayıp en optimal değeri SGD'ye buldurur. Çok sınıflı bir hedef değişken olduğu

için categorical_crossentropy kullanıyoruz. metrics: İlave olarak burada model değerlendirme kriterleri belirlenir. Bir liste halinde verilir. Biz şimdilik sadece accuracy kullanıyoruz.

Batch Size	Epoch	Activation	Accuary
16	10	relu	0.938
		tanh	0.964
	70	relu	0.947
		tanh	0.956
32	10	relu	0.929
		tanh	0.964
	70	relu	0.947
		tanh	0.964
64	10	relu	0.912
		tanh	0.929
	70	relu	0.947
		tanh	0.956

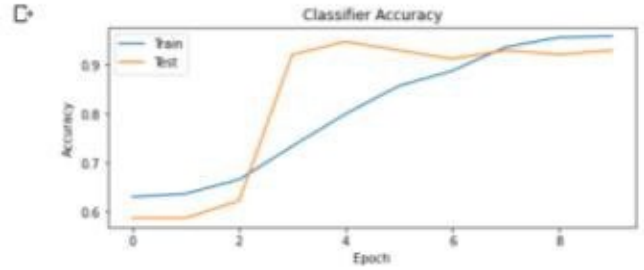
İkiden çok etiketli sınıflarda accuracy hesaplamak metrics modülü **accuracy_score**() fonksiyonunu kullanıyoruz.
En iyi sonuçları :
batch_size = 16, epochs = 10 , activation = tanh
batch_size = 32, epochs = 10 , activation = tanh
batch_size = 64, epochs = 70 , activation = tanh
sağlamıştır.
Bu sonuçlar üzerine ortak nokta activation=tanh olarak yüksek doğruluk oranı saptanmıştır.

G. Grafiklendirme

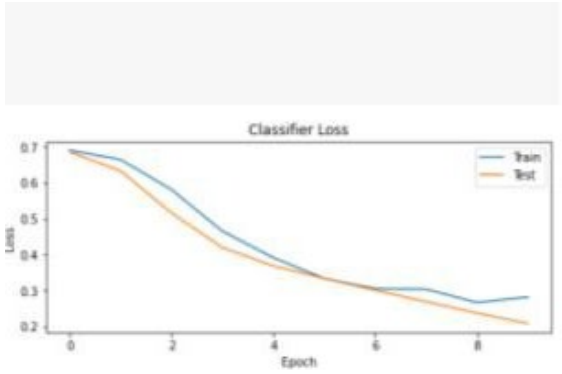
Grafiklendirme için matplotlip sınıfını kullanıyoruz.

```
scores=classifier.evaluate(X_test, y_test)
print("Accuracy: ", scores[1])

#MLP Accuracy Grafiği;
from matplotlib import pyplot as plt
import numpy as np
plt.figure(figsize=(17,3))
plt.subplot(1, 2, 1)
plt.plot(egitim.history['accuracy'])
plt.plot(egitim.history['val_accuracy'])
plt.title('Classifier Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(['Train', 'Test'], loc='upper left')


plt.subplot(1, 2, 2)
plt.plot(egitim.history['loss'])
plt.plot(egitim.history['val_loss'])
plt.title('Classifier Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper right')

plt.show()
```

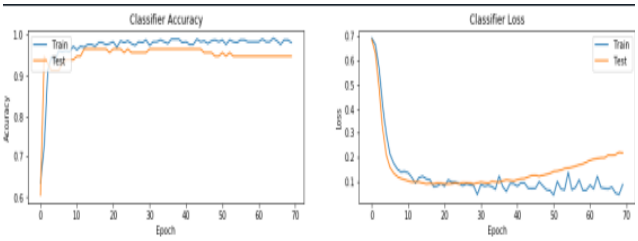
```
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper right')

plt.show()
```



Grafiklendirme sonuçları;

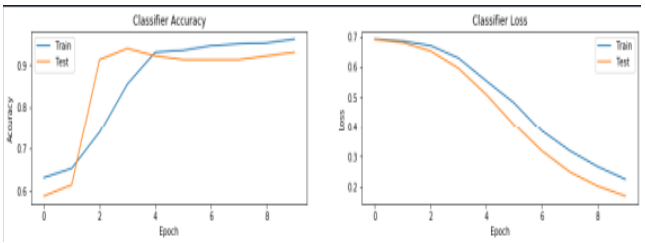


batch_size = 16, epochs = 70, activation = 'relu'

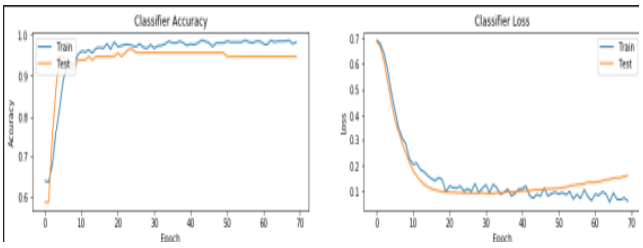


Sınıflandırıcı Algoritma	Başarı Oranı
SVM	0.98
MLP	0.96

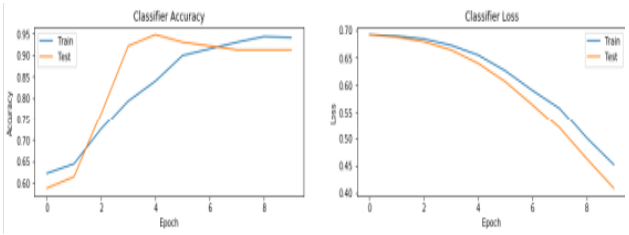
batch_size = 32, epochs = 10 , activation = 'relu'



batch_size = 32, epochs = 70 , activation = 'relu'



batch_size = 64, epochs = 10, activation = 'relu'



KAYNAKÇA

[1] “ KERAS DERIN ÖĞRENME KÜTÜPHANESI?” HTTPS://WWW.VERIBILIMIOKULU.COM/KERAS-DERIN-OGRENME-KUTUPHANESI-ILE-SINIFLANDIRMA-IRIS-VERI-SETI-UZERINDE-UYGULAMSON ERIŞIM TARIHI: 20 HAZIRAN 2020.” PYTHON ILE MAKINE ÖĞRENMESINE GIRIŞ PANDAS KÜTÜPHANESI” HTTPS://WWW.VERIBILIMIOKULU.COM/PYTHON-ILE-MAKINE- OGRENMESINE-GIRIS-PANDAS-KUTUPHANESI/SON ERIŞIM TARIHI:20 HAZIRAN 2020

[2] “ ÇOK KATMANLI ALGILAYICILAR “

http://www.nuhazginoglu.com/2018/05/15/cok-katmanli-algilayicilar-multi-layer-perceptron/Son erişim tarihi: 20 Haziran 2020

[3] ”MAKINE ÖĞRENIMI BÖLÜM4”, MEDIUM HTTPS://MEDIUM.COM/@ISIKHANELIF/MULTI-LAYER-PERCEPTRON-MLP-NEDIR-4758285A7F15SON ERIŞIM TARIHI:20 HAZIRAN 2020

[4] “DESTEK VEKTÖR MAKİNELERİ” , HTTP://WWW.VERIDEFTERI.COM/2018/12/13/TAHMIN-VE-CIKARIM-9-DESTEK-VEKTOR-MAKINELERI/, SON ERIŞIM TARIHI:20 HAZIRAN 2020

[6] “MultiLayer Perceptron nedir?, Medium https://medium.com/@isikhanelif/multi-layer-perceptron-mlp-nedir4758285a7f15

Son erişim tarihi: 20Haziran 2020

[7] “train_test_split parameters” https://scikitlearn.org/stable/modules/generated/sklearn.model_selection.train_test_split.ht ml

Son erişim tarihi: 20Haziran 2020