KÜTAHYA SAĞLIK BİLİMLERİ ÜNİVERSİTESİ

Bilgisayar Mühendisliği

Yapay Zeka Dersi Proje Tasarım Raporu

Celal ALTIN

June 14, 2024

Bu çalışma Yapay Zeka dersi final raporunu içermektedir.

1 Giriş

2019 yılında başlayan COVID-19 salgını dünya çapında ciddi sağlık, ekonomik ve sosyal etkilere yol açmıştır.(Mart 2022 de yayınlanan bir habere göre [1] dünyada ölüm sayısının 18 milyonu aştığı hesaplanıyor, bu sayı ülkemizde ise Sağlık bakanlığının verilerine göre 31.05.2022 tarihine kadar 98.965 kişinin hayatını kaybetti yönünde.)Bugün bile bu etkilerin sonuçları geçmiş değildir.

Resim 1: Dunya Covit Haritasi [2]

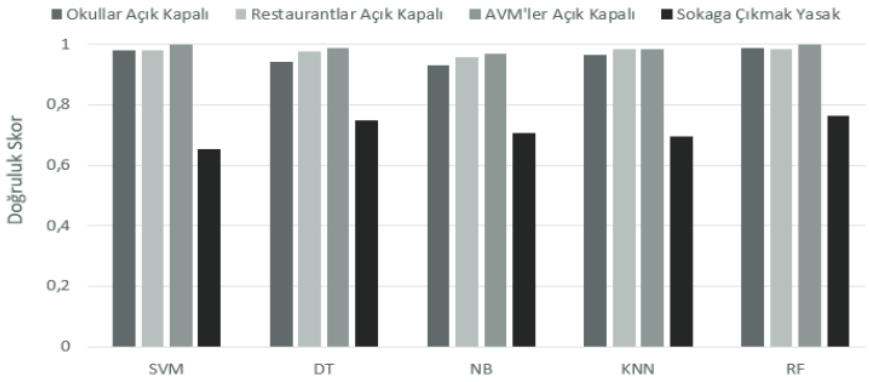
Dünya çapındaki bu problemi daha iyi anlayabilmek ve ileride olası hastalıklarda daha etkili mücadele edebilmek için bilgisayar(Makine öğrenmesi kulkanılarak yapılan bir araştırmaya göre sosyal medyanın da etkisiyle aşıya olan olumlu bakış artmıştır [3]) ve bilgisayar özelinde yapay zeka teknolojisinden faydalanmak en akılcı yöntemlerden birisidir.Bende bu projemde çeşitli kaynaklardan bulduğum Covit-19 verilerini kullanarak olası bir salgın hastalık dumunda gerçekleşebilecek seneryoyu gün yüzüne çıkartmayı amaçlıyorum.

2 Literatür Araştırması

Koronavirüs, 2019 yılının Aralık ayında ilk olarak Çin'in Wuhan kentinde ortaya çıkmış ve 11 Mart 2020'de Dünya Sağlık Örgütü tarafından pandemi olarak ilan edilmiştir. Vaka sayılarını kontrol altına almak için pek çok ülke karantina, sokağa çıkma yasağı ve sosyal alanların bir süreliğine kapatılması

gibi çeşitli önlemler almıştır. Doğrulanmış vaka tahminlemesi pandemide olası planlamalar için büyük önem taşımaktadır. Gelecek verilerinin gerçeğe en yakın bir şekilde tahminlenmesi; pandemi döneminde lojistik, tedarik, hastane personel ve malzeme planlaması için kullanılabileceği gibi aşılama senaryolarında da girdi olarak kullanılabilir. Literatürde doğrulanmış vaka tahmininde makine öğrenmesi, bölmeli model, zaman serisi analizi gibi pek çok yöntem kullanarak tahminleme yapılan çalışmalar vardır. Bu çalışmada, Amerika Birleşik Devletleri'ndeki doğrulanmış vaka sayılarını kullanarak gelecek günlerdeki vaka tahminlerini çeşitli makine öğrenmesi modelleri yapılmıştır. Python ve R programlama dili kullanılarak yapılan tahminlemeler Prophet, Polinom Regresyon, ARIMA, Doğrusal Regresyon ve Random Forest modelleri ile yapılmıştır. Test verisiyle tahmin edilen verilerin performansları ortalama mutlak yüzde hatası (MAPE), ortalama karekök sapması (RMSE) ve ortalama mutlak hata (MAE) kullanılarak değerlendirilmiştir [4].
PCA(Veri boyutunu azaltma yöntemleri sınıflandırma yapmak için harcanan zamanı ve bazı durumlarda sınıflandırma hatasını azaltmaya yardımcı olur. Zaman kritik uygulamalarda öznitelik elde etme evresinde harcanan zamanı azaltmak için, öznitelik seçme yöntemleri, tüm giriş değerlerinin ölçülmesini gerektiren boyut indirgeme yöntemlerine tercih edilir[5]) yöntemi kullanıldığında doğruluk değeri en yüksek RF algoritmasında, duyarlılık ve kesinlik değeri en yüksek SMV algoritmasında saptanmıştır. Bu yöntemin kullanılması sonucunda en düşük doğruluk NB algoritmasında, duyarlılık ve kesinlik değerleri en düşük NB ve DT algoritmalarında elde edilmiştir[6].

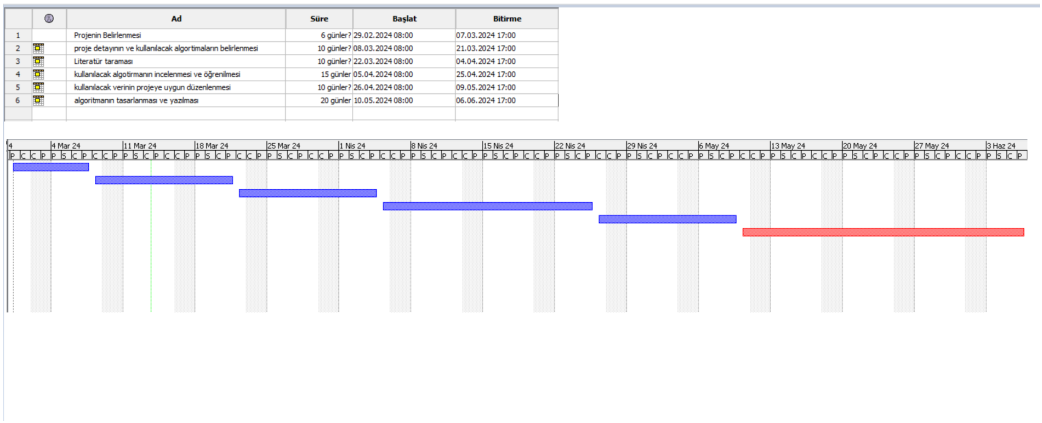
Resim 2: Dogruluk Tablosu



3 Metodoloji

Projeni şematik planı aşağıdaki şekildedir.

Resim 3: GANTT CHART



Projede kullanacağımız veriler gün veya haftalık olarak pozitif hasta sayısını, vefat edenlerin sayısını, kullandığımız verilerdeki insanların yaşadığı şehrin yada ülkenin nüfusunu, iyileşen sayısını,toplam test sayısı gibi parametreleri içermelidir.
Veri işleme Normalizasyon işlemi- Araştırmalarda veri setlerinde verilerin bütünlüğünün sağlanması, veri tekrarının önlenmesi ve veri bütünlüğünün

korunması ile performansının artırılması için normalizasyon yapılmaktadır. Daha sonra Çapraz doğrulama (Çapraz doğrulama, makine öğrenimi modellerinin başarı derecesini ortaya koymak için kullanılan yöntemdir. Çapraz doğrulama algoritma performansı hakkında bilgi verirken, verilerin daha verimli kullanılmasını sağlar.[6])-yöntemi kullanılacaktır. Daha sonra PCA yöntemi kullanılarak veri kümesini azalttıktan sonra RF(Random Forest) algoritmasına beslenecektir.

4 Kullanılacak veri

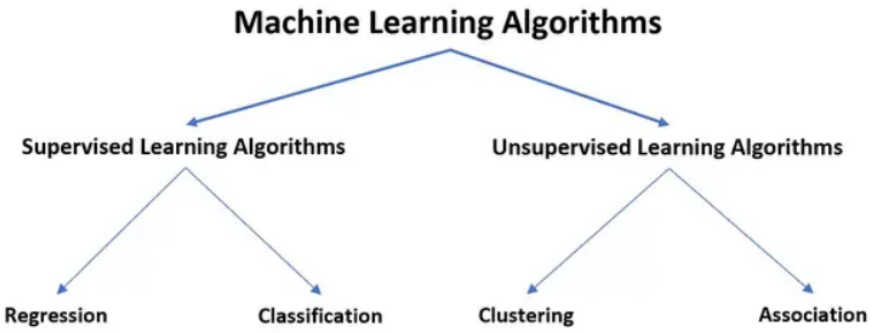
Projemde iki farklı veri kaynağından faydalandım.Bunlardan ilki T.C. Sağlık Bakanlığından alınmış verilerdir.[7].Verilerimiz 11.03.2020-31.05.2022 tarihleri arasında alınmış 813 farklı kayda sahiptir.Bu veriler Toplam Vaka, Günlük Vaka, Toplam Hasta, Gunluk Hasta, Toplam Vefat, Gunluk Vefat, Toplam iyilesen, Gunluk iyilesen, Gunluk Test şeklinde etiketlenmiştir. İkinci olarak ise Our World in Data kaynağından faydalanılmıştır[8].

5 Yöntem

Olası bir salgın hastalık durumunda oluşabilecek sonuçları tahmin etmede fikir vermesi ve projemin % 90 nın üzerinde doğrulukla sonuçlanması.

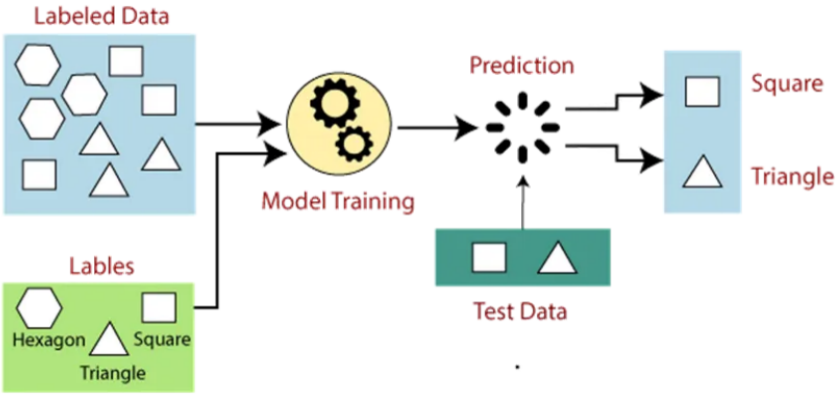
Çalışmamızda Random Forest algoritmasını kullanılacağından bahsedilmişti.Random Forest algoritması Denetimli Öğrenme algoritmaları sınıfına ait bir algoritmadır.İnternete Random Forest algoritması yazıldığında karşınıza Random Forest regresyon(Regression) ve Random Forest sınıflandırma (Classification) algoritmaları çıkacaktır.

Resim 4: Machine Learning Algorithms



Regresyon ve sınıflandırma farkından bahsetmeden önce denetimli öğrenme ve denetimsiz öğrenme algoritmaları arasındaki farktan bahsetmek daha sağlıklı olacaktır.
Denetimli Öğrenme Algoritması: Denetimli Öğrenme algoritmasındaki en önemli nokta etiketli bir veri kümesi (labeled dataset) kullanılmasıdır. Yani hangi verinin hangi bilgiye karşılık geldiği bilindiğinden bilinen bir girdi seti ile bunlara denk gelen çıktıları alıp algoritmanın daha önce hiç görmediği (eğitimde kullanılmayan) yeni verilere en uygun çıktıları üretmek için kullanılan bir makine öğrenmesi modelidir.[9]

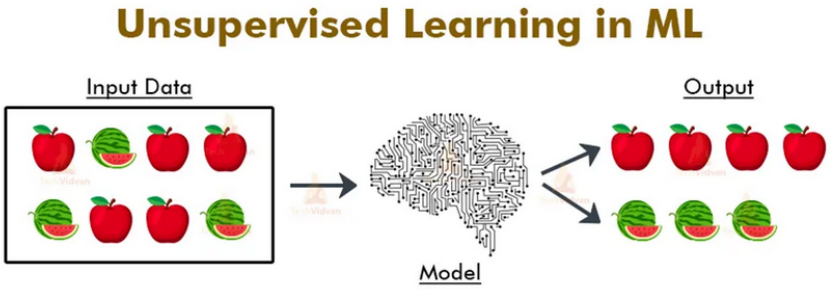
Resim 5: Denetimli Öğrenme



Denetimsiz Öğrenme Algoritması: Denetimsiz Öğrenmede ise etiketsiz veriler vardır. Bu etiketsiz veriler arasındaki gizli kalmış yapıyı/örüntüyü bulmaya çalışarak kendi kendine öğrenme biçimi sergilenir.

Denetimli öğrenme genellikle Regresyon ve Sınıflandırma problemlerine uygulanırken, denetimsiz öğrenme Kümeleme (Clustering) ve İlişkilendirme (Association) problemlerine uygulanır.

Resim 6: Denetimsiz Öğrenme



Yukarı da da söylediğimiz gibi Random Fores algoritması araştırıldığında Random Forest regresyon ve Random Forest Sınıflandırma algoritmaları ile karşılaşılacak.Regresyon ve Sınıflandırma algoritmalarına biraz daha açıklık getirmemiz gereklidir.

Regresyon(Regression) Nedir ?

Regresyon bağımlı bir değişken ile bağımsız bir değişken arasındaki ilişkinin, ortadan kaldırılması için kullanılan istatistiksel bir yöntemdir. Evet,

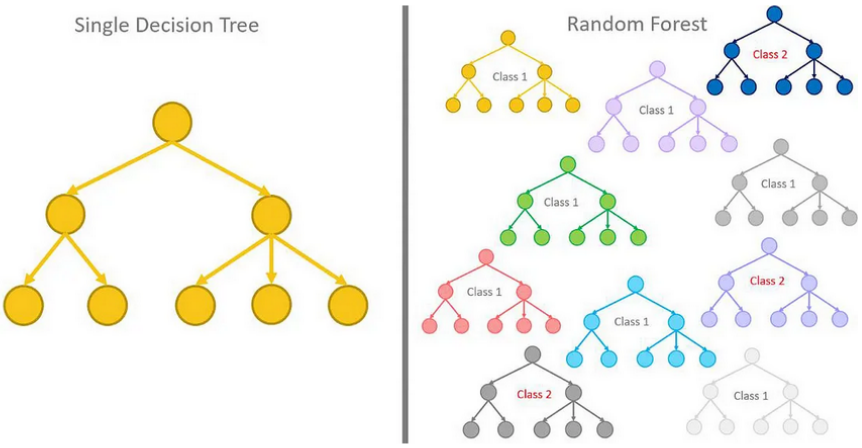
regresyonun bu teorik açıklaması size karmaşık gelmiş olabilir. Gelin biz bunu daha basit haliyle açıklayalım. Buradaki değişkenler(x ve y arasında, x → deneyim yılı, y → maaş olarak düşünebilirsiniz) arasında sebep-sonuç ilişkisi bulmaya çalışırız ve bulduğumuz bu ilişkiye göre tahminler yaparız. En basit haliyle regresyonu açıklayacak olursak, bir veri setinde sayısal tahmin yapıyorsak "regression" algoritmalarını kullanırız.
Örnek → Maaş tahmini, yaşa göre boy tahmini, çalışma süresine göre alınan not tahmini gibi örnekler verebiliriz.

Sınıflandırma(Classification) Nedir ?
Regression problemlerinde tahmin edeceğimiz y kolonunda sayısal değerler vardı. Biz x değerlerini yani giriş değerini kullanarak sayısal tahminler yapmaya çalışıyorduk. Sınıflandırma problemlerinde ise x(giriş değerleri) değerlerini kullanarak kategorik olarak y sınıfını tahmin etmeye çalışırız.[10]

Resim 7: Regresyon ve Sınıflandırma

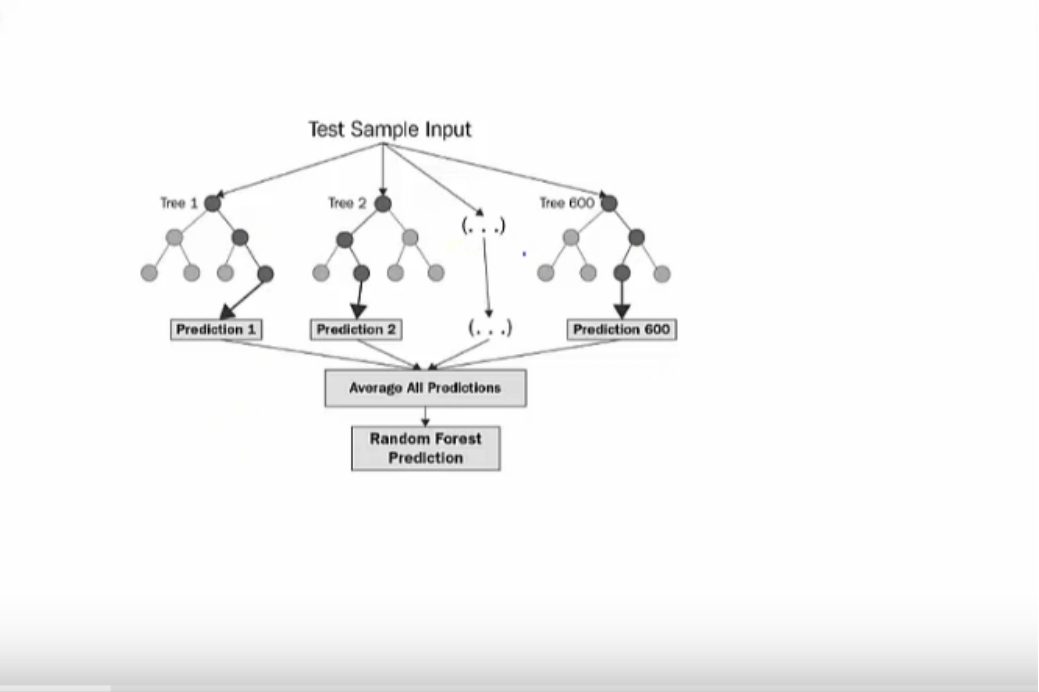
Random Forest (Rassal Orman) Algoritması: Sınıflandırma işlemi esnasında birden fazla karar ağacı üreterek sınıflandırma değerini yükseltmeyi hedefleyen bir algoritmadır. Bireysel olarak oluşturulan karar ağaçları bir araya gelerek karar ormanı oluşturur. Buradaki karar ağaçları bağlı olduğu veri setinden rastgele seçilmiş birer alt kümedir.

Resim 8: Single Decision tree and Random Forest



Fark ettiyseniz Random Forest karar ağaçlarının bir nevi birleşiminden oluşan bir algoritma.[11]

Resim 9: Random Forest



Random Forest algoritmamı uygulayacağım veriler aşağıda gösterilmiştir.

Resim 10: Covit Verileri

Tarih	Toplam Vaka	Günlük Vaka	Toplam_Hasta	Gunluk_Hasta	Toplam_Vefat	Gunluk_Vefat	Toplam_iyileser	Gunluk_iyileser	Toplam_Test	Gunluk_Test	YBU	Entube	Ağır Hasta
11.03.2020	1		1	1					940	940			
12.03.2020	1		1	0					2.470	1.530			
13.03.2020	5		2	1					4.000	1.530			
14.03.2020	6		5	3					5.000	1.000			
15.03.2020	18		18	13					6.000	1.000			
16.03.2020	47		47	29					7.000	1.000			
17.03.2020	98		98	51	1	1			8.002	1.002			
18.03.2020	191		191	93	2	1			10.000	1.998			
19.03.2020	359		359	168	4	2			11.981	1.981			
20.03.2020	670		670	311	9	5			15.637	3.656			
21.03.2020	947		947	277	21	12			18.590	2.953			
22.03.2020	1.236		1.236	289	30	9			20.345	1.755			
23.03.2020	1.529		1.529	293	37	7			24.017	3.672			
24.03.2020	1.872		1.872	343	44	7			27.969	3.952			
25.03.2020	2.433		2.433	561	59	15			33.004	5.035			
26.03.2020	3.629		3.629	1.196	75	16	0	0	40.290	7.286			
27.03.2020	5.698		5.698	2.069	92	17	42	42	47.823	7.533	344	241	
28.03.2020	7.402		7.402	1.704	108	16	70	28	55.464	7.641	445	309	

Verilerim gün bazında Türkiye için Toplam Vaka, Günlük Vaka, Toplam Hasta, Günlük Hasta, Toplam Vefat, Günlük Vefat, Toplam İyileşen, Günlük İyileşen, Toplam Test, Günlük Test, Yoğun Bakım Ünitesi ve Ağır Hasta parametrelerinden 812 kayıt içermektedir.
Öncelikler kullacağımız kütüpheneleri çalışmamıza ekliyoruz ve pandas kütüphanesinin read.csv fonksiyonuyla verilerimizi içeri alıyoruz.
Daha sonra verilerimizi incelemek ilk 5 veriyi getiriyoruz.

Resim 11: Kodlar

```
[28]: import numpy as np
      import matplotlib.pyplot as plt
      import pandas as pd

[29]: from sklearn.metrics import mean_squared_error, r2_score
      from sklearn.model_selection import train_test_split

[30]: veri = pd.read_csv("demo.csv")
      veri.head()
```

[30]:

	Unnamed: 0	Açıklamalar	Tarih	Toplam Vaka	Günlük Vaka	Toplam_Hasta	Gunluk_Hasta	Toplam_Vefat	Gunluk_Vefat	Toplam_iyilesen	Gunluk_iyilesen	Toplam_Test	Gur
0	1.0	Veri Seti	11.03.2020	1	NaN	1.0	1.0	NaN	NaN	NaN	NaN	940	
1	NaN	COVID-19 Pandemisi Türkiye Günlük Verileri	12.03.2020	1	NaN	1.0	0.0	NaN	NaN	NaN	NaN	2.470	
2	2.0	Son Güncelleme	13.03.2020	5	NaN	2.0	1.0	NaN	NaN	NaN	NaN	4.000	
3	NaN	31.05.2022	14.03.2020	6	NaN	5.0	3.0	NaN	NaN	NaN	NaN	5.000	
4	3.0	Kapsam	15.03.2020	18	NaN	18.0	13.0	NaN	NaN	NaN	NaN	6.000	

Resim 12: Kodlar2

```
[31]: veri.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 812 entries, 0 to 811
Data columns (total 19 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Unnamed: 0                      9 non-null      float64
 1   Açıklamalar                     24 non-null     object
 2   Tarih                           812 non-null    object
 3   Toplam Vaka                     812 non-null    object
 4   Günlük Vaka                     553 non-null    float64
 5   Toplam_Hasta                    481 non-null    float64
 6   Gunluk_Hasta                    481 non-null    float64
 7   Toplam_Vefat                    806 non-null    float64
 8   Gunluk_Vefat                    806 non-null    float64
 9   Toplam_iyilesen                 797 non-null    object
 10  Gunluk_iyilesen                 797 non-null    float64
 11  Toplam_Test                     812 non-null    object
 12  Gunluk_Test                     812 non-null    float64
 13  YBU                             124 non-null    float64
 14  Entube                          124 non-null    float64
 15  Ağır Hasta                      339 non-null    float64
 16  yatak_doluluk_orani             56 non-null     object
 17  eriskin_yogun_bakim_doluluk_orani  56 non-null  object
 18  ventilator_doluluk_orani        56 non-null     object
dtypes: float64(11), object(8)
memory usage: 120.7+ KB
```

Verilerimizi detaylı incelemek için veri.info() metotunu kullanıyoruz.

Resim 13: Kodlar3

```
[33]: veri = veri.drop(["Unnamed: 0", "Açıklamalar"], axis = 1)

[34]: veri.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 812 entries, 0 to 811
Data columns (total 17 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   Tarih                             812 non-null    object
 1   Toplam Vaka                       812 non-null    object
 2   Günlük Vaka                       553 non-null    float64
 3   Toplam_Hasta                      481 non-null    float64
 4   Gunluk_Hasta                      481 non-null    float64
 5   Toplam_Vefat                      806 non-null    float64
 6   Gunluk_Vefat                      806 non-null    float64
 7   Toplam_iyilesen                   797 non-null    object
 8   Gunluk_iyilesen                   797 non-null    float64
 9   Toplam_Test                       812 non-null    object
 10  Gunluk_Test                       812 non-null    float64
 11  YBU                               124 non-null    float64
 12  Entube                            124 non-null    float64
 13  Ağır Hasta                        339 non-null    float64
 14  yatak_doluluk_orani               56 non-null     object
 15  eriskin_yogun_bakim_doluluk_orani 56 non-null     object
 16  ventilator_doluluk_orani          56 non-null     object
dtypes: float64(10), object(7)
memory usage: 108.0+ KB
```

Daha sonra kullanılmayacak olan Unnamed : 0 ve Açıklamalar sütununu sililyoruz ve tekrar kontrol ediyoruz.

Resim 14: Kodlar4

```
[35]: veri.Gunluk_iyilesen[0:20]

[35]: 0      NaN
      1      NaN
      2      NaN
      3      NaN
      4      NaN
      5      NaN
      6      NaN
      7      NaN
      8      NaN
      9      NaN
      10     NaN
      11     NaN
      12     NaN
      13     NaN
      14     NaN
      15     0.0
      16    42.0
      17    28.0
      18    35.0
      19    57.0
      Name: Gunluk_iyilesen, dtype: float64

[36]: veri.loc[0:14,"Gunluk_iyilesen"]=0
```

Günlük iyileşen sütununa baktığımızda ilk 15 verimizin NaN değere sahip olduğunu görüyoruz.Algoritmamızda kullanacağımız için 2 değerini atıyoruz. Daha sonra aynı işlemi Günlük vefat ve Toplam iyileşen

Resim 15: Kodlar5

```
[37]:
veri.loc[0:6,"Gunluk_Vefat"]=0

[38]: veri["Toplam_iyilesen"][0:20]

[38]: 0     NaN
      1     NaN
      2     NaN
      3     NaN
      4     NaN
      5     NaN
      6     NaN
      7     NaN
      8     NaN
      9     NaN
      10    NaN
      11    NaN
      12    NaN
      13    NaN
      14    NaN
      15      0
      16     42
      17     70
      18    105
      19    162
      Name: Toplam_iyilesen, dtype: object

[39]: veri.loc[0:5,"Toplam_iyilesen"]=0 #nan değerlere 0 atadık
```

sütunlarımıza da yapıyoruz.

Resim 16: Kodlar6

```
[14]: X = veri.iloc[:, [8,10]].values

[15]: y = veri.iloc[:, 6].values

[16]: from sklearn.model_selection import train_test_split
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.27, random_state = 0)

[17]: #from sklearn.preprocessing import StandardScaler
      #sc_X = StandardScaler()
      #X_train = sc_X.fit_transform(X_train)
      #X_test = sc_X.transform(X_test)

[18]: from sklearn.ensemble import RandomForestRegressor
      rf=RandomForestRegressor( n_estimators=100,random_state=0)
      rf.fit(X_train,y_train )
      print("Train Başarısı = %",rf.score(X_train,y_train)*100)
      print("Test Başarısı = %",rf.score(X_test,y_test)*100)

      Train Başarısı = % 97.19048399039058
      Test Başarısı = % 83.05438190827856
```

Daha sonra eğitimde ve testte kullanılacak verilerimizi ayırıyoruz. Verilerimizin %27 sini test için ayırıyoruz.100 adet karar ağacı kullanıyoruz.Eğitim başarımını %97 test başarımını %83 olarak buluyoruz.

Resim 17: Kodlar7

```
y_pred = rf.predict(X_test)
```

```
mse=mean_squared_error(y_test,y_pred)
r2=r2_score(y_test,y_pred)
```

```
print("mean squared error",mse)
print("hata kareler",r2)
```

```
mean squared error 1400.2288695454545
hata kareler 0.8305438190827856
```

R Kare: R kare, modeldeki bağımsız değişkenlere göre bağımlı değişkenin varyasyon oranını yani bağımlı değişkendeki değişkenliğin ne kadarının model tarafından açıklanabileceğini ölçer. Korelasyon katsayısının karesidir. R Kare aşırı uyum (overfitting) sorununu dikkate almaz. Regresyon modelinin çok fazla bağımsız değişkeni varsa model eğitim verilerine çok iyi uyabilir ama testte istenen başarıyı gösteremeyebilir. Bu nedenle Düzeltilmiş R Kare kullanılır. Düzeltilmiş R Kare modele eklenen ek bağımsız değişkenleri cezalandırır ve aşırı uyum sorununu çözer.

$1 - \frac{Ssres}{SStot}$
SSres: hata kareler toplamı
SStot: toplam kareler toplamı
Ortalama Kare Hatası (Mean Squared Error (MSE)): Ortalama Kare Hatası tahmin edilen sonuçlarınızın gerçek sayıdan ne kadar farklı olduğuna dair size mutlak bir sayı verir. Tek bir sonuçtan çok fazla içgörü yorumlayamazsınız, ancak size diğer model sonuçlarıyla karşılaştırmak için gerçek bir sayı verir ve en iyi regresyon modelini seçmenize yardımcı olur. [12]
$\frac{1}{n} * \sum(y - ypred)^2$
n:Veri noktalarının sayısı.
y:Gerçek değerler.
ypred: Tahmin edilen değerler.

Daha Sonraki projemizi diğer veri setinden çektiğimiz veriler ile yapacağız.[8]

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
```

[5]:
```
veri = pd.read_csv("owid-covid-data.csv")
veri.loc[0:4,"new_cases_smoothed"]=0
veri.head(10)
```

[5]:

	iso_code	continent	location	date	total_cases	new_cases	new_cases_smoothed	total_deaths	new_deaths	new_deaths_smoothed	...	male_smokers	ha
0	AFG	Asia	Afghanistan	2020-01-05	NaN	0.0	0.0	NaN	0.0	NaN	...	NaN	
1	AFG	Asia	Afghanistan	2020-01-06	NaN	0.0	0.0	NaN	0.0	NaN	...	NaN	
2	AFG	Asia	Afghanistan	2020-01-07	NaN	0.0	0.0	NaN	0.0	NaN	...	NaN	
3	AFG	Asia	Afghanistan	2020-01-08	NaN	0.0	0.0	NaN	0.0	NaN	...	NaN	
4	AFG	Asia	Afghanistan	2020-01-09	NaN	0.0	0.0	NaN	0.0	NaN	...	NaN	
5	AFG	Asia	Afghanistan	2020-01-10	NaN	0.0	0.0	NaN	0.0	0.0	...	NaN	
6	AFG	Asia	Afghanistan	2020-01-11	NaN	0.0	0.0	NaN	0.0	0.0	...	NaN	
7	AFG	Asia	Afghanistan	2020-01-12	NaN	0.0	0.0	NaN	0.0	0.0	...	NaN	
8	AFG	Asia	Afghanistan	2020-01-13	NaN	0.0	0.0	NaN	0.0	0.0	...	NaN	
9	AFG	Asia	Afghanistan	2020-01-14	NaN	0.0	0.0	NaN	0.0	0.0	...	NaN	

Resim 18: Verilemizin Genel İncelenmesi

Yukarıda gördüğünüz kodlarda kullanıcağımız kütüphaneleri projemize ekledik ve kullanacağımız verilerden new cases smoothed sütunundaki ilk 4 veri NaN değere sahip olduğu için 0 atadık ve daha sonra ilk 10 verimizi kontrol ettik.

```
[6]: veri.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 387253 entries, 0 to 387252
Data columns (total 67 columns):
 #   Column                                    Non-Null Count   Dtype
---  ------                                    --------------   -----
 0   iso_code                                  387253 non-null  object
 1   continent                                 368726 non-null  object
 2   location                                  387253 non-null  object
 3   date                                      387253 non-null  object
 4   total_cases                               348333 non-null  float64
 5   new_cases                                 376280 non-null  float64
 6   new_cases_smoothed                        375055 non-null  float64
 7   total_deaths                              326109 non-null  float64
 8   new_deaths                                376589 non-null  float64
 9   new_deaths_smoothed                       375359 non-null  float64
 10  total_cases_per_million                   348333 non-null  float64
 11  new_cases_per_million                     376280 non-null  float64
 12  new_cases_smoothed_per_million            375050 non-null  float64
 13  total_deaths_per_million                  326109 non-null  float64
 14  new_deaths_per_million                    376589 non-null  float64
 15  new_deaths_smoothed_per_million           375359 non-null  float64
 16  reproduction_rate                         184817 non-null  float64
 17  icu_patients                              38642 non-null   float64
 18  icu_patients_per_million                  38642 non-null   float64
 19  hosp_patients                             40178 non-null   float64
 20  hosp_patients_per_million                 40178 non-null   float64
 21  weekly_icu_admissions                     10689 non-null   float64
 22  weekly_icu_admissions_per_million         10689 non-null   float64
 23  weekly_hosp_admissions                    24159 non-null   float64
 24  weekly_hosp_admissions_per_million        24159 non-null   float64
 25  total_tests                               79387 non-null   float64
 26  new_tests                                 75403 non-null   float64
 27  total_tests_per_thousand                  79387 non-null   float64
 28  new_tests_per_thousand                    75403 non-null   float64
 29  new_tests_smoothed                        103965 non-null  float64
 30  new_tests_smoothed_per_thousand           103965 non-null  float64
 31  positive_rate                             95927 non-null   float64
 32  tests_per_case                            94348 non-null   float64
 33  tests_units                               106788 non-null  object
 34  total_vaccinations                        83316 non-null   float64
 35  people_vaccinated                         79196 non-null   float64
 36  people_fully_vaccinated                   76078 non-null   float64
 37  total_boosters                            51503 non-null   float64
 38  new_vaccinations                          69060 non-null   float64
 39  new_vaccinations_smoothed                 190037 non-null  float64
 40  total_vaccinations_per_hundred            83316 non-null   float64
 41  people_vaccinated_per_hundred             79196 non-null   float64
 42  people_fully_vaccinated_per_hundred       76078 non-null   float64
 43  total_boosters_per_hundred                51503 non-null   float64
 44  new_vaccinations_smoothed_per_million     190037 non-null  float64
 45  new_people_vaccinated_smoothed            187406 non-null  float64
 46  new_people_vaccinated_smoothed_per_hundred 187406 non-null  float64
 47  stringency_index                          197292 non-null  float64
 48  population_density                        329253 non-null  float64
 49  median_age                                306058 non-null  float64
 50  aged_65_older                             295505 non-null  float64
 51  aged_70_older                             302990 non-null  float64
 52  gdp_per_capita                            300095 non-null  float64
 53  extreme_poverty                           193444 non-null  float64
 54  cardiovasc_death_rate                     300681 non-null  float64
 55  diabetes_prevalence                       316186 non-null  float64
 56  female_smokers                            225701 non-null  float64
 57  male_smokers                              222633 non-null  float64
 58  handwashing_facilities                    147326 non-null  float64
 59  hospital_beds_per_thousand                265585 non-null  float64
 60  life_expectancy                           356680 non-null  float64
 61  human_development_index                   291642 non-null  float64
 62  population                                387253 non-null  float64
 63  excess_mortality_cumulative_absolute      13172 non-null   float64
 64  excess_mortality_cumulative               13172 non-null   float64
 65  excess_mortality                          13172 non-null   float64
 66  excess_mortality_cumulative_per_million   13172 non-null   float64
dtypes: float64(62), object(5)
memory usage: 198.0+ MB
```

Resim 19: Verilerimize Detaylı İncelenmesi

Hangi türden ne kadar veri olduğunu ve hangi sütun adı altında olduklarını inceledik.

```
4]: df1=pd.DataFrame(veri)
    newCasesAfghanistan = df1[df1['location'] == 'Afghanistan']
    newCasesAfghanistan = newCasesAfghanistan['new_cases']
    newCasesAfghanistan = newCasesAfghanistan[newCasesAfghanistan.index % 7 == 0]
    newCasesAfghanistan.head(50)
```

Resim 20: Afganistan için new cases sütununun seçilmesi

İlk verilerimiz Afganistan'a ait verilerdir.Verilerimizi bir dataframe a atıp location kolonunda Afghanistan olan verilerimizi alıyoruz(Birden fazla ülkenin verileri aynı veri tabanında olduğu için).Daha sonra bağımlı değişkenlerimizde kullanmak üzere new cases verilemizi ayıklayıp indeks numarası 7 nin katları olacak şekilde tekrar alıyoruz.Bunun sebebi ise verilerimizin her 7 günde güncellenmesi aradaki günlerde veri girilmemesi.

```
df2 = pd.DataFrame(veri)
newDeathsAfghanistan = df2[df2['location'] =='Afghanistan']
newDeathsAfghanistan = newDeathsAfghanistan['new_deaths']
newDeathsAfghanistan = newDeathsAfghanistan[newDeathsAfghanistan.index % 7 == 0]
newDeathsAfghanistan.head(50)
```

Resim 21: Afganistan için new deaths sütununun seçilmesi

Aynı işlemleri new deaths, new cases smoothed, diabetes prevalence, cardiovasc death rate ve population sütunlarımız içinde yapıyoruz.

```
df3 = pd.DataFrame(veri)
newCasesSmoothedAfghanistan = df3[df3['location'] =='Afghanistan']

newCasesSmoothedAfghanistan = newCasesSmoothedAfghanistan['new_cases_smoothed']
newCasesSmoothedAfghanistan = newCasesSmoothedAfghanistan[newCasesSmoothedAfghanistan.index % 7 == 0]
newCasesSmoothedAfghanistan
```

Resim 22: Afganistan için new cases smoothes sütununun seçilmesi

```
df4 = pd.DataFrame(veri)
diabetesPrevalenceAfghanistan = df4[df4['location'] =='Afghanistan']
diabetesPrevalenceAfghanistan = diabetesPrevalenceAfghanistan['diabetes_prevalence']
diabetesPrevalenceAfghanistan = diabetesPrevalenceAfghanistan[diabetesPrevalenceAfghanistan.index % 7 == 0]
diabetesPrevalenceAfghanistan
```

Resim 23: Afganistan için diabetes prevalance sütununun seçilmesi

```
df5 = pd.DataFrame(veri)
cardiovascDeathRateAfghanistan = df5[df5['location'] =='Afghanistan']
cardiovascDeathRateAfghanistan = cardiovascDeathRateAfghanistan['cardiovasc_death_rate']
cardiovascDeathRateAfghanistan = cardiovascDeathRateAfghanistan[cardiovascDeathRateAfghanistan.index % 7 == 0]
cardiovascDeathRateAfghanistan
```

Resim 24: Afganistan için cardiovasc death rate sütununun seçilmesi

```
df6 = pd.DataFrame(veri)
PoPulationAfghanistan = df6[df6['location'] =='Afghanistan']
PoPulationAfghanistan = PoPulationAfghanistan['population']
PoPulationAfghanistan = PoPulationAfghanistan[PoPulationAfghanistan.index % 7 == 0]
PoPulationAfghanistan
```

Resim 25: Afganistan için population sütununun seçilmesi

```
X = np.column_stack((newCases, newCasesSmoothed, diabetesPrevalence, cardiovascDeathRate, PoPulation))
```

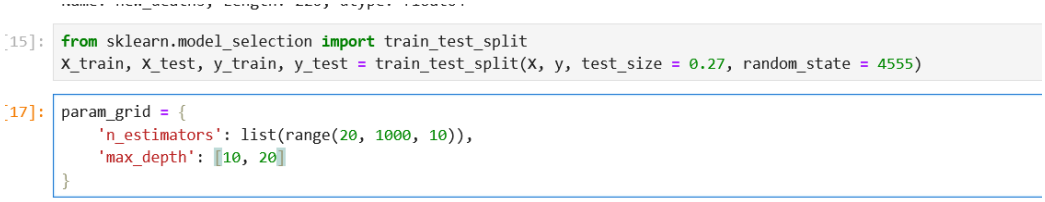
```
print(X.shape)
```

```
(218, 5)
```

```
y =  (newDeaths)
newDeaths.head(50)
y
```

Resim 26: Bağımlı ve bağımsız değişkenlerin oluşturulması

Daha sonra projemizde kullanmak üzere bağımlı X ve bağımsız y değikenlerini oluşturuyoruz.Kodda da görüldüğü gibi bağımlı değişkenimiz new cases, new cases smoothed, diabetes prevalence, cardiovasc death rate ve population verilerimiz içerirken tahim edeceğimiz yani bağımsız değişkeinimiz new deaths verilerini içeriyor.En son olarak bağımlı ve bağımsız değişkenlerin boyutlarını kontrol etmemin sebebi satır sayılarının aynı olması gerektiğindendir.



Resim 27: Test, eğitim verilerinin ayrılması ve parametre ayarı.

Yukarıdaki kodda test ve eğitim verilerimizi %27 test verisi olacak şekilde ayırdık.Daha sonra en başarılı parametre ayarını bulmak için değişkenlerimizi tanımladık.

Resim 28: Test, eğitim verilerinin ayrılması ve parametre ayarı.

Yukarıdaki kodda Resim **??** da tanımladığımız değişkenleri kullanarak her değişkenin tüm kombinasyonlarını deneyerek mse sini hesaplıyoruz ve en skoru veren parametre değerlerini buluyoruz.

```
rf=RandomForestRegressor( n_estimators=40,random_state=16,max_depth=10)
rf.fit(X_train,y_train )
print("Train Başarısı = %",rf.score(X_train,y_train)*100)
print("Test Başarısı = %",rf.score(X_test,y_test)*100)
```

```
Train Başarısı = % 93.66768559173688
Test Başarısı = % 91.83467180451275
```

```
y_pred = rf.predict(X_test)
```

```
mse=mean_squared_error(y_test,y_pred)
r2=r2_score(y_test,y_pred)
```

```
print("mean squared error",mse)
print("hata kareler",r2)
```

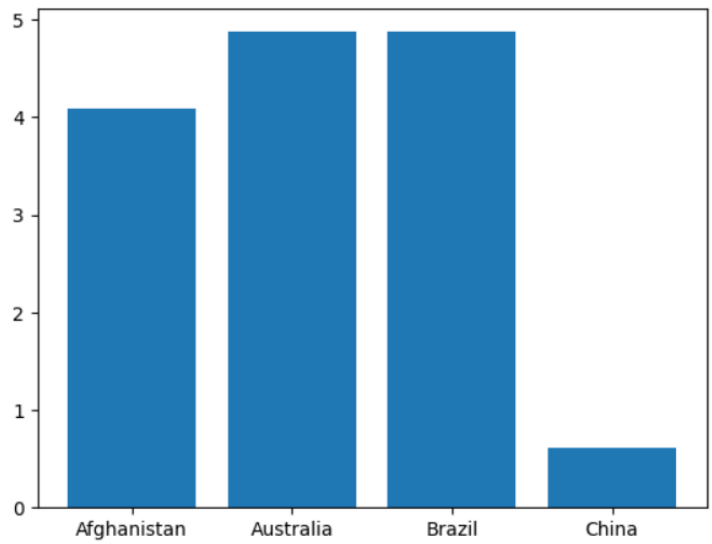
Resim 29: Modelin eğitilmesi.

En son olarak Resim ?? de elde ettiğimiz verilere göre modelimizi eğitiyoruz, mse ve r2 değerlerini hesaplıyoruz.

Aynı işlemi veri tabanımızdaki Avustralya, Brezilya, ve Çin içinde yaptık.Lakin eğitim başarımları ve test başarımları tatmin edici seviyede değildi.

```
[233]: ülkeler=['Afghanistan','Australia','Brazil','China']
       degerler=[4.09045066,4.87375132,4.8713835,0.61276007]
```

```
[234]: plt.bar(ülkeler,degerler)
       plt.show()
```



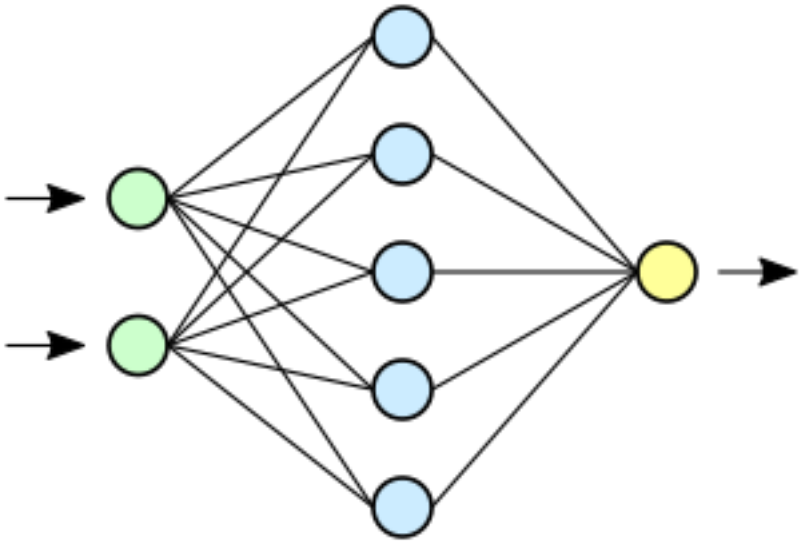
Resim 30: 4 Ülkenin Karşılaştırılması.

Yukarıdaki kodlarda 4 ülke için eğitilmiş modele beslediğimiz değerlerden

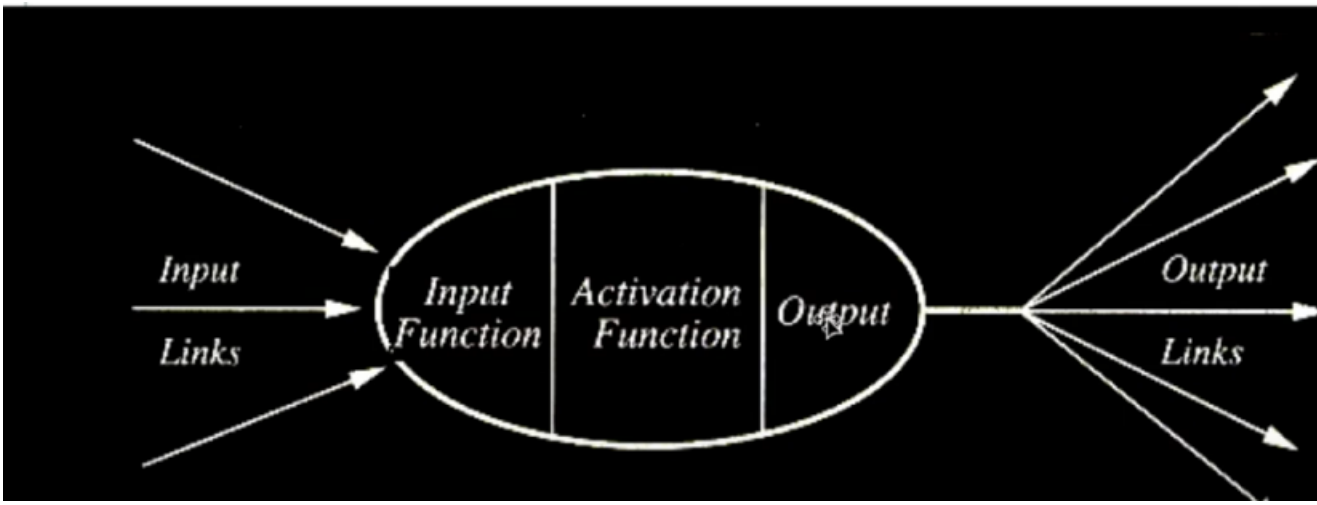
(new cases = 100 ,new cases smoothed = 20, diabetes prevalence = 5, cardiovasc death rate=150, population = 1000000) elde ettiğimiz çıktılar karşılaştırılmıştır.

Ülkeler	Eğitim başarımı	Test başarımı
Afghanistan	% 93.66	% 91.83
Australya1	% 87.35	% 35.51
Brezil	% 93.18	% 54.74
China	% 78.92	% 57.72

Yukarıdaki tabloda projemde kullandığım ülke verilerin eğitim ve test başarımları verilmiştir. Geçen haftalarda Random Forest algoritmasıyla yaptığım çalışmayı Yapay Sinir Ağları(Neural Network) ile yapmaya çalışıyıyorum. Öncelikle yapay sinir ağlarını tanımakla başlamanın doğru olacağını düşünüyorum.
Yapay sinir ağları (YSA), insan beyninin bilgi işleme tekniğinden esinlenerek geliştirilmiş bir bilgi işlem teknolojisidir. YSA ile basit biyolojik sinir sisteminin çalışma şekli taklit edilir. Yani biyolojik nöron hücrelerinin ve bu hücrelerin birbirleri ile arasında kurduğu sinaptik bağın dijital olarak modellenmesidir. Nöronlar çeşitli şekillerde birbirlerine bağlanarak ağlar oluştururlar. Bu ağlar öğrenme, hafızaya alma ve veriler arasındaki ilişkiyi ortaya çıkarma kapasitesine sahiptirler. Diğer bir ifadeyle, YSA'lar, normalde bir insanın düşünme ve gözlemlemeye yönelik doğal yeteneklerini gerektiren problemlere çözüm üretmektedir. Bir insanın, düşünme ve gözlemleme yeteneklerini gerektiren problemlere yönelik çözümler üretebilmesinin temel sebebi ise insan beyninin ve dolayısıyla insanın sahip olduğu yaşayarak veya deneyerek öğrenme yeteneğidir. [13]
Yapay sinir ağlarına diğer nöronlardan gelen ağırlıkları ile işlenmiş sinyaller giriş fonksiyonuna sokulur ve orada belirlenmiş işlemler yapılır(Toplama, çarpma, min-max bulma vb.).Daha sonra aktifleştime fonksiyonuna girer.Burada belli fonksiyonlardan geçerek eşik değerinden geçerse 1, geçmezse 0 tetiklenir.Output dan diğer nöronlara gönderilir.

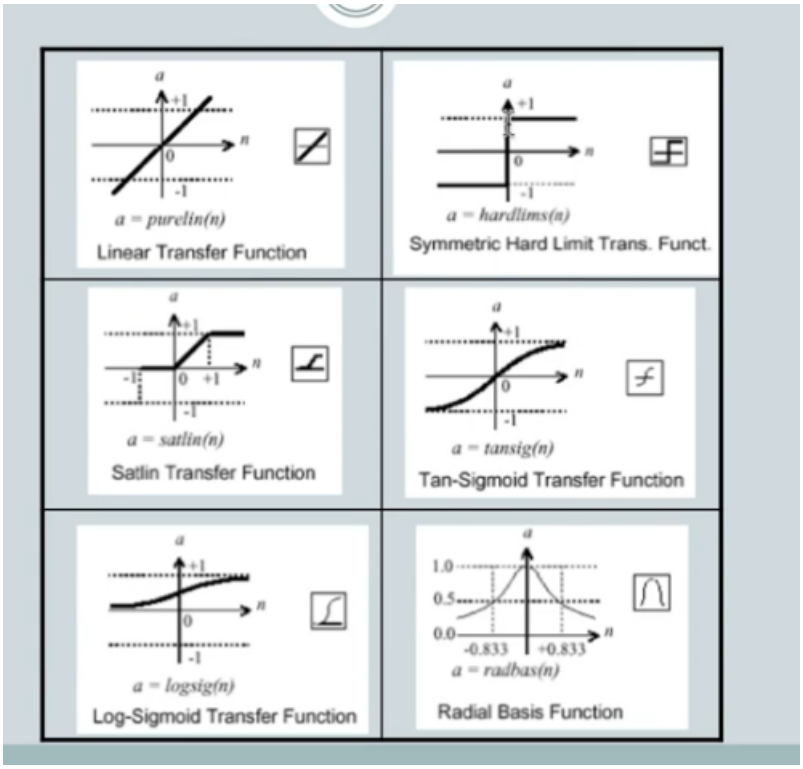


Resim 31: Yapay Sinir Ağları örnek şeması.



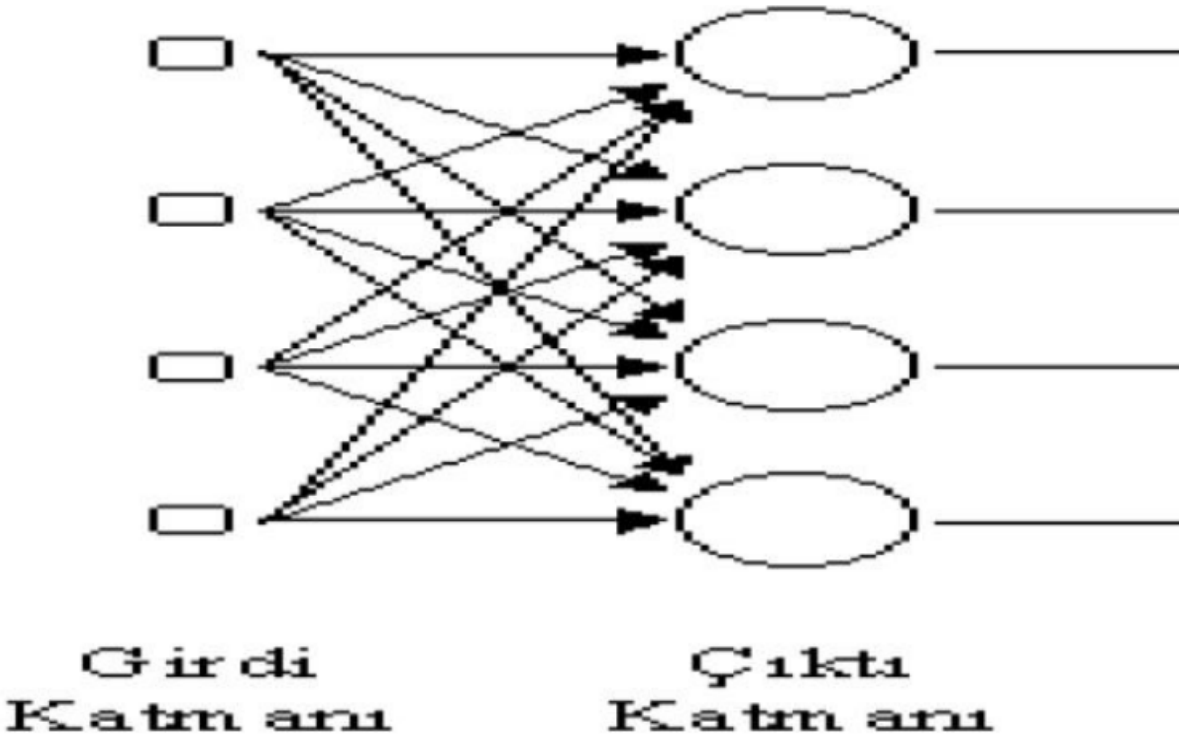
Resim 32: Nöron yapısı.
[14]

Yukarıda gördüğünüz fonksiyonlar aktivleştime fonksiyonları içinde kullanılan fonksiyonlardır.
Tek katmanlı Yapay Sinir Ağları : En basit ağ tipi olup bir çıktı katmanı ve buna bağlı bir girdi katmanından oluşmaktadır.Ara katman bulundurmaz.
Bu tür ağ tipiyle basit problemleri çözebiliriz. Örnek olarak aşağıda a,b



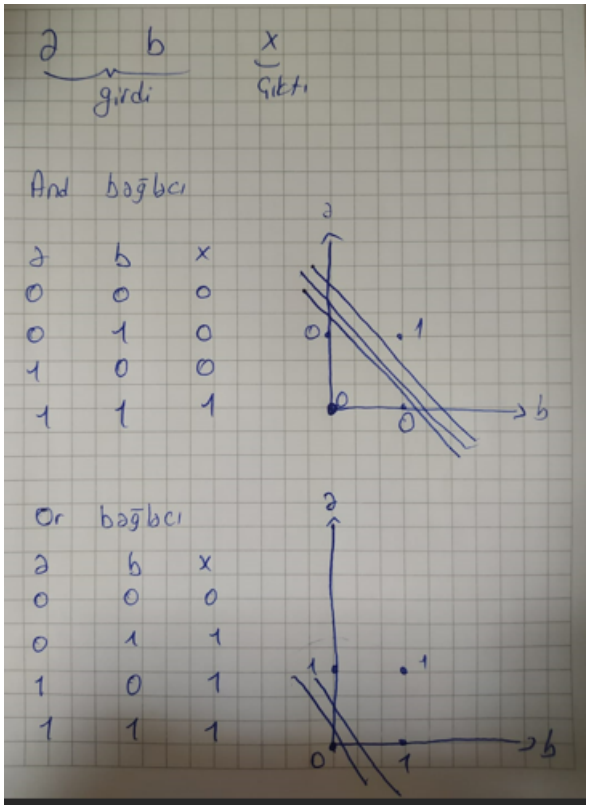
Resim 33: Aktifleştime fonksiyonları.
[14]

değişkenlerini and ve or kapısına soktuktan sonraki çıktıları inceliyoruz.



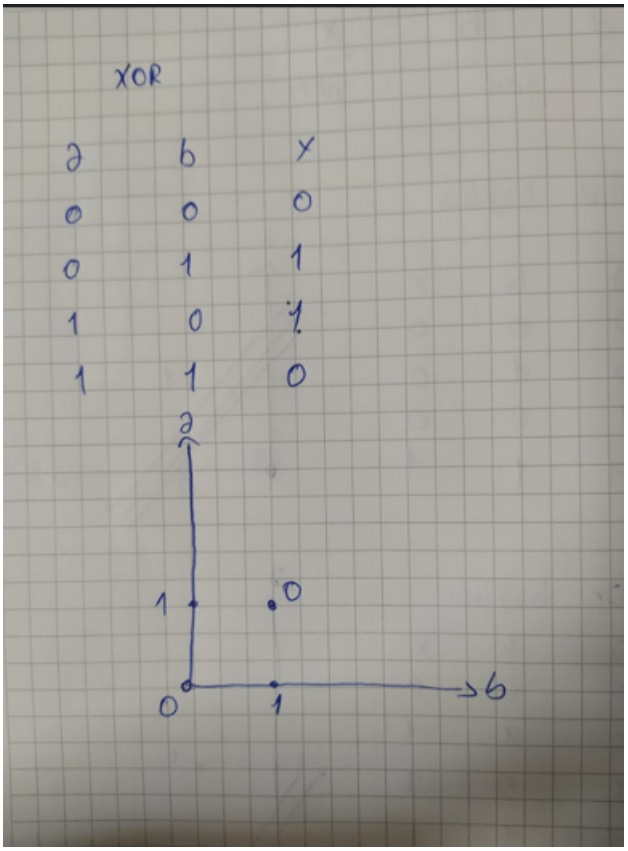
Resim 34: Tek katmanlı Yapay Sinir Ağları.
[15]

Yukarıdaki resimde de görebileceğiniz üzere a ve b değişkenlerini iki boyutlu düzleme yerleştirdiğimizde 0 ve 1 değerlerini bir soğru yardımıyla birbirinden ayırabiliyoruz.



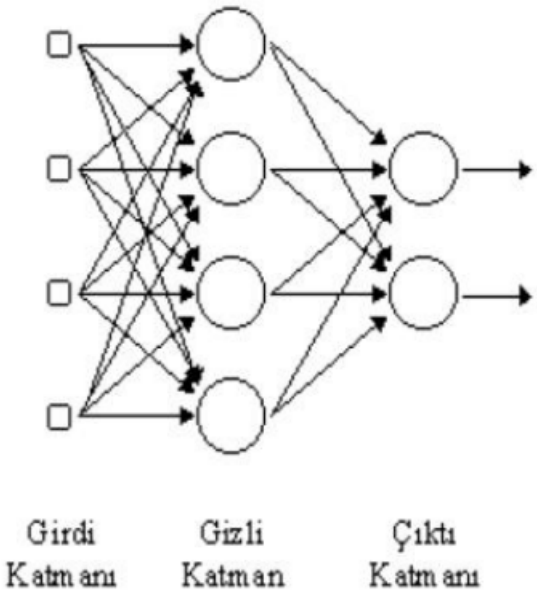
Resim 35: Tek katmanlı Yapay Sinir Ağları çözümleri.

Lakin yukarıdaki problemi (XOR kapısına sokulmuş değişkenler) tek katmanlı yapay sinir ağlarıyla çözemezsiniz.Tek bir doğru ile 0 ve 1 noktalarını birbirinden ayıramazsınız.Burada devreye Çok Katmanlı Yapay Sinir Ağları giriyor.
Çok katmanlı Yapay Sinir Ağları : Girdi katmanı ile çıktı katmanı arasında ara katman veya katmanların olmasıdır.



Resim 36: Çok katmanlı Yapay Sinir Ağları çözümleri.

Çok katmanlı Yapay Sinir Ağları daha karmaşık problemleri çözmek için tasarlanmıştır.Bundan dolayıda eğitilmesi daha zordur ve daha çok zaman alır.
Yapay Sinir Ağlarında Bias kavramı: Bias skaler bir değerdir.İnput fonksiyonunda değerlerimize eklenir.Doğrunun özelliğini değiştirmez sadece kaydırma işlemi yapar. Bu hafta geçen haftalarda Random Forest algoritmasıyla yaptığım çalışmamı Yapay Siniar ağlarına taşıdım.Yapay Sinir Ağlarına geçen hafta değindiğim için bu hafta değinmeyeceğim.
Alınan Hatalar: AttributeError: module 'tensorflow.python.distribute.input lib' has no attribute 'DistributedDatasetInterface'. Did you mean: 'DistributedDatasetSpec'? Projede Tensorflow 2.16.1 sürümü kullanılmakta. Lakin DistributedDatasetInterface özelliği 2.13 sürümlerinin altında çalışmakta.PyChar Tensorflow 2.13 öncesi sürümleri indirirken hata verdi. Daha sonra Google Colab da kodları çalıştırdık.
Yukarıdaki kodlarda kütüphanelerimi ekledik ve verilerimizi okumaya



Resim 37: Çok katmanlı Yapay Sinir Ağları.
[15]

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from scipy import stats
from sklearn.metrics import mean_absolute_error, r2_score, mean_squared_erro
veri = pd.read_csv("/owid-covid-data.csv")
```

```
veri.loc[0:4,"new_cases_smoothed"]=0
veri.head(10)
df1=pd.DataFrame(veri)
newCasesAfghanistan = df1[df1['location'] == 'Afghanistan']
newCasesAfghanistan = newCasesAfghanistan['new_cases']
newCasesAfghanistan = newCasesAfghanistan[newCasesAfghanistan.index % 7 == 0
newCasesAfghanistan.head(50)
```

Resim 38: Kütüphanelerin eklenmesi ve verilerin okunması.

başladık.

```
df2 = pd.DataFrame(veri)
newDeathsAfghanistan = df2[df2['location'] =='Afghanistan']
newDeathsAfghanistan = newDeathsAfghanistan['new_deaths']
newDeathsAfghanistan = newDeathsAfghanistan[newDeathsAfghanistan.index % 7 == 0]
newDeathsAfghanistan.head(50)

df3 = pd.DataFrame(veri)
newCasesSmoothedAfghanistan = df3[df3['location'] =='Afghanistan']
newCasesSmoothedAfghanistan = newCasesSmoothedAfghanistan['new_cases_smoothed']
newCasesSmoothedAfghanistan = newCasesSmoothedAfghanistan[newCasesSmoothedAfghanistan.inc

df4 = pd.DataFrame(veri)
diabetesPrevalenceAfghanistan = df4[df4['location'] =='Afghanistan']
diabetesPrevalenceAfghanistan = diabetesPrevalenceAfghanistan['diabetes_prevalence']
diabetesPrevalenceAfghanistan = diabetesPrevalenceAfghanistan[diabetesPrevalenceAfghanist

df5 = pd.DataFrame(veri)
cardiovascDeathRateAfghanistan = df5[df5['location'] =='Afghanistan']
cardiovascDeathRateAfghanistan = cardiovascDeathRateAfghanistan['cardiovasc_death_rate']
cardiovascDeathRateAfghanistan = cardiovascDeathRateAfghanistan[cardiovascDeathRateAfgha

df6 = pd.DataFrame(veri)
PoPulationAfghanistan = df6[df6['location'] =='Afghanistan']
PoPulationAfghanistan = PoPulationAfghanistan['population']
PoPulationAfghanistan = PoPulationAfghanistan[PoPulationAfghanistan.index % 7 == 0]
```

Resim 39: Verilerin okunması.

Daha sonra algoritmamızda kullanacağımız diğer verilerdi de okuduk.

```
#eğitim ve test veri setlerinin ayrılması
X = np.column_stack((newCasesAfghanistan, newCasesSmoothedAfghanistan, diabetesPrevalenceA
print(X.shape)


y =  (newDeathsAfghanistan)
print(y.shape)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,random_state=44)

#Ölçeklendirme-normalizasyon
from sklearn.preprocessing import MinMaxScaler

scaler=MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)
y_train = np.array(y_train)
y_test = np.array(y_test)
y_train = scaler.fit_transform(y_train.reshape(-1, 1))
y_test = scaler.fit_transform(y_test.reshape(-1, 1))
print(len(X_test))
print(len(X_train))
```

```
(220, 5)
(220,)
66
154
```

Resim 40: Verilerin ayrıştırılması ve hazırlanması

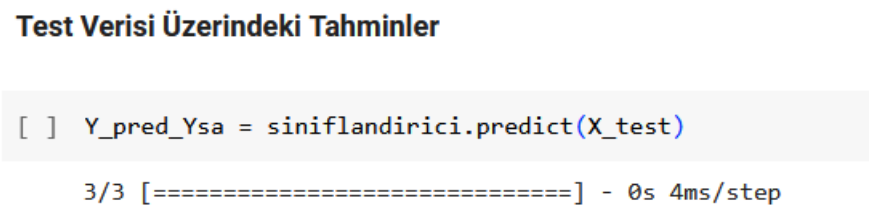
Bağımlı ve bağımsız değişkenlerimizi hazırladıktan sonra verilerimizi %30 u test verisi olacak şekilde ayırıyoruz. X_train ve y_train eğitimde kullanacağımız verilerimizi içerir.X_test ve y_test ise test verilerimizi içerir. Daha sonra verilerimiz üstündeki dönüşümleri yapıyoruz.

```
[ ]  from keras.models import Sequential
     from keras.layers import Dense
     import keras
```

```
     siniflandirici = Sequential()
     siniflandirici.add(Dense(units=6, activation='relu', input_dim=5))
     siniflandirici.add(Dense(units=6, activation='relu'))
     siniflandirici.add(Dense(units=1, activation='sigmoid'))  # Son katman, tek sınıf
     siniflandirici.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'
     siniflandirici.fit(X_train, y_train, batch_size=10, epochs=100)
```

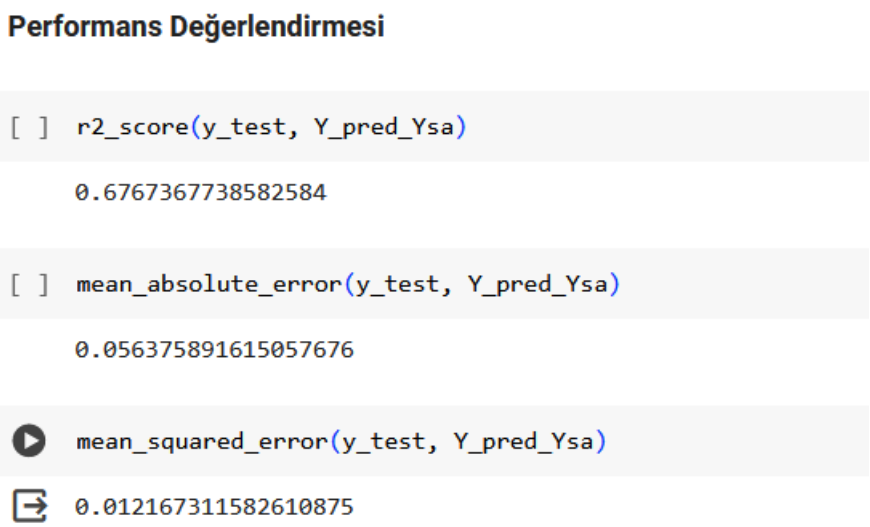
Resim 41: Modelimizin oluşturulması

Yapay Sinir ağları modelimiz için gerekli olan kütüphaneleri çalışmamıza ekliyoruz. Daha sonra siniflandirici nesnesi oluşturuyoruz.Add komutuyla katman ekliyoruz.Units parametresi nöron sayısımı activation ise aktivasyon fonksiyonumuzu temsil ediyor.En son olarak da fit metotuyla sistemimizi eğitiyoruz.



Resim 42: Modelden tahmini değerler elde etme.

Eğittiğimiz modele X_test verilerimiz besleyerek çıktıları alıyoruz.



Resim 43: Modelin performansını değerlendirme.

Daha sonra eğittiğimiz modelden elde ettiğimiz test sonuçları ile gerçek test verilerimizi karşılaştırarak r2, mean absolute error, mean squared error metriklerinde değerlendiriyoruz.

```
[ ]  for i in siniflandirici.layers:
       ilk_gizli_katman = siniflandirici.layers[0].get_weights()
       ikinci_gizli_katman = siniflandirici.layers[1].get_weights()
       cıktı_katmani = siniflandirici.layers[2].get_weights()
```

Resim 44: Ağırlıkların bulunması.

```
ilk_gizli_katman
```

```
[array([[ 0.36072823, -0.542492  , -0.4884844 , -0.6574583 , -0.45034996,
          0.57824135],
        [-0.1548088 ,  0.3305871 ,  0.25217718,  0.20984769,  0.11268872,
          1.2797422 ],
        [-0.43525657,  0.3248691 , -0.12678403,  0.52211004, -0.27542093,
         -0.22245878],
        [ 0.22686034,  0.5322649 , -0.5438212 , -0.3136623 ,  0.6457837 ,
         -0.479788  ],
        [-0.25860316,  0.01380146,  0.5381115 , -0.27385166, -0.5827503 ,
         -0.08330488]], dtype=float32),
 array([ 0.69865644,  0.        ,  0.        ,  0.        ,  0.        ,
        -0.02893527], dtype=float32)]
```

Resim 45: Ağırlıkların bulunması.

```
[ ]  ikinci_gizli_katman
```

```
[array([[-0.32305998,  1.0317196 , -0.23102528, -0.39570025, -0.2483892 ,
          0.8593963 ],
        [ 0.4728834 ,  0.7039711 , -0.5363893 , -0.3879769 , -0.17248446,
          0.4935891 ],
        [ 0.61111194, -0.59522545, -0.53340137,  0.14144784,  0.12372935,
         -0.01962644],
        [-0.3923096 ,  0.6297967 ,  0.65444463, -0.225328  ,  0.69410783,
         -0.13475037],
        [ 0.6386582 , -0.19315094,  0.02613175, -0.06956053, -0.14348543,
         -0.13212866],
        [-0.3815453 , -0.8596109 ,  1.2878914 , -0.214385  ,  0.4062121 ,
         -0.916494  ]], dtype=float32),
 array([0.        , 0.5421095 , 0.16150045, 0.        , 0.17353901,
        0.5666428 ], dtype=float32)]
```

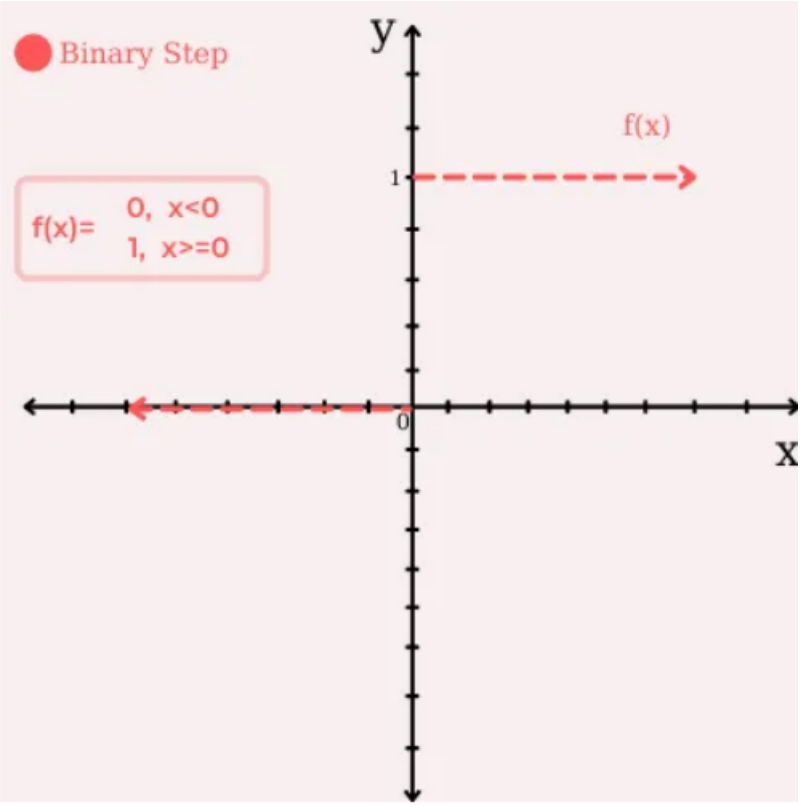
Resim 46: Ağırlıkların bulunması.

```
cıktı_katmani

[array([[ 0.1255877],
        [-1.3635423],
        [ 0.7871704],
        [ 0.3579669],
        [ 0.4277682],
        [-1.467797 ]], dtype=float32),
 array([-0.41433668], dtype=float32)]
```

Resim 47: Ağırlıkların bulunması.

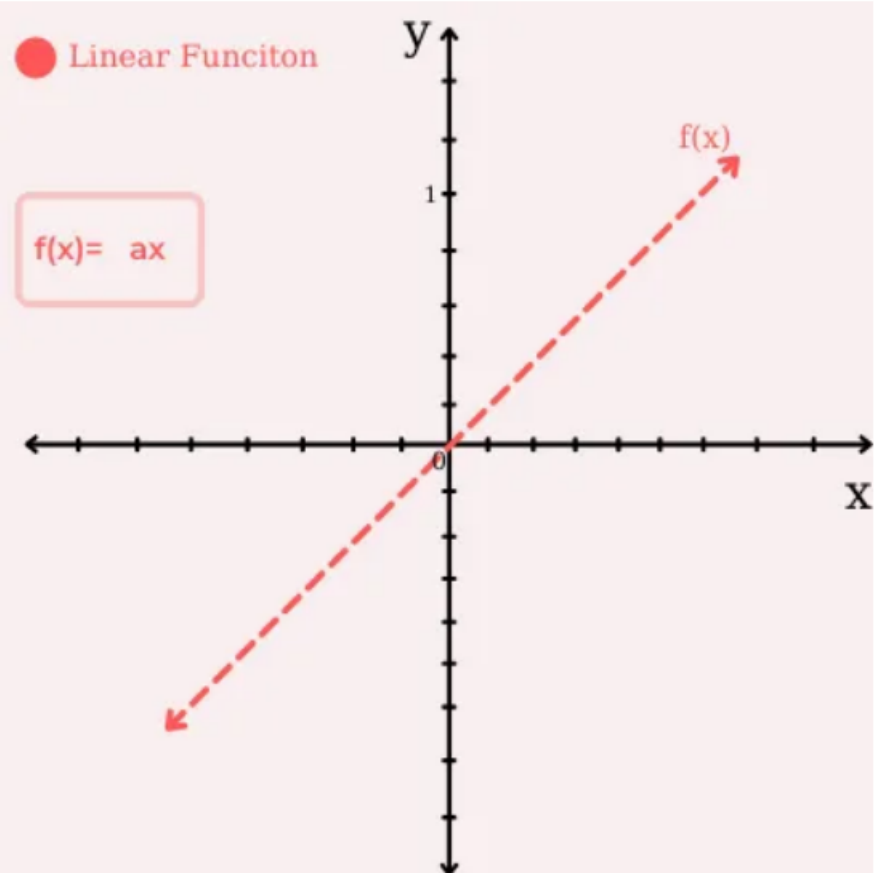
Yukarıdaki kodlarda ise nöronlar arasındaki ağırlıkları buluyoruz.Her katmanın ayrı ayrı hesaplayıp çıktısını inceliyoruz.[16] Geçen haftaki çalışmamızda yapay sinir ağları ile bir model oluşturup verimiz üzerinde çalıştırmıştık. Bu hafta ise yapar sinir ağları parametre ayarları hakkında araştırma yapıldı.Öncelikle aktivasyon fonksiyonlarından bahsedeceğiz.
Aktivasyon fonksiyonları: Yapay sinir ağlarında kullanılan aktivasyon fonksiyonları, her bir katmanın çıktılarını non-lineer hale getirerek sinir ağının daha karmaşık ilişkileri öğrenmesine olanak tanır. Aşağıda sık kullanılan Aktivasyon kodlarından bahsedilmiştir.

1. Binary Step Fonksiyonu: Binary Step Fonksiyonu ikili sınıflandırma durumunda kullanılır. Çoklu sınıflandırmada işe yaramaz. Genellikle çıkış katmanlarında (Output Layer) kullanılır.
Türevi sıfır olduğu için de Geri Yayılım(Back Propagation) sırasında parametreler güncellenmez yani öğrenme işlemi gerçekleşmez. Bu yüzden de Gizli Katmanlarda(Hidden Layers) tercih edilmez.Fonksiyonumunuz türevi bizim için önemlidir.
Lineer Fonksiyon: Model optimizasyonu sırasında, türevi sabit bir değere eşit olacağı için öğrenme işlemi istenilen başarımı göstermeyecektir. Bu yüzden bu fonksiyon çok basit görevler dışında kullanılması tercih edilmez.
Sigmoid Fonksiyonu: En çok kullanılan aktivasyon fonksiyonlarından birisidir. Şimdiye kadar bahsettiğimiz ilk doğrusal olmayan(non-linear) fonksiyondur. Karar vermeye yönelik olasılıksal bir yaklaşımdır ve değer aralığı [0,1] arasındadır. Yani çıktının hangi sınıfa ait olduğuna dair olasılıksal bir değer verir bize. Sigmoid fonksiyonu sürekli yani türevlenebilir bir fonksiyon olduğundan öğrenme işlemi gerçekleşir.
$g(x) = 1/(1 + e^{(} - x))$



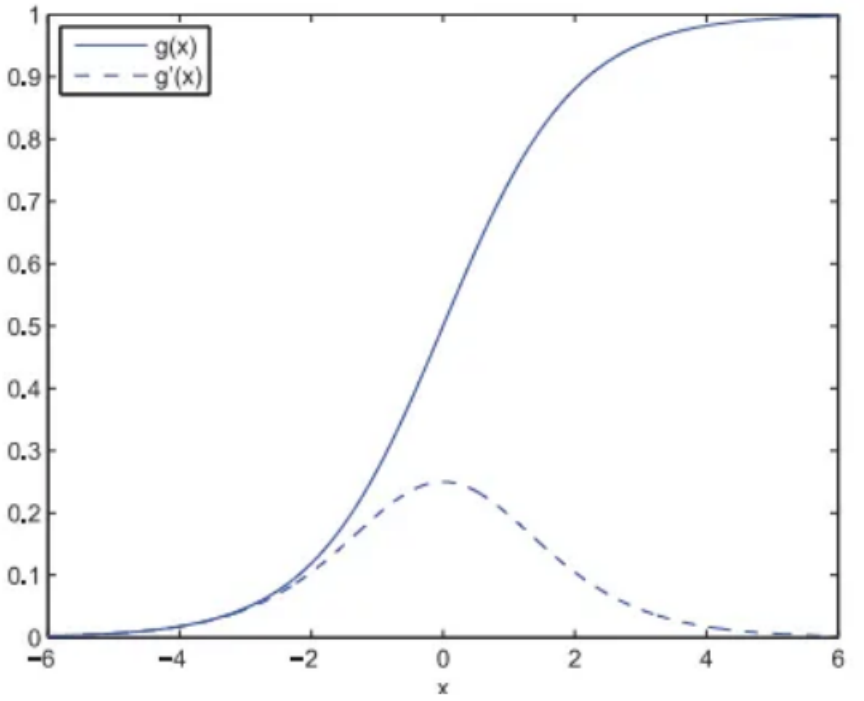
Resim 48: Binary Step fonksiyonu.

Fakat Sigmoid de mükemmel değildir. Çünkü türev değeri, tablodan da görebildiğimiz üzere, uç noktalarda sıfıra yakınsar (Vanishing Gradient).Bu durum Back Propagation sırasında öğrenmenin durmasına sebebiyet verir . Bu sebeple çok katmanlı ağlarda kullanılması tavsiye edilmez.
Hiperbolik Tanjant(tanh): Sigmoid fonksiyonu ile oldukça benzerdir. Bu fonksiyon da ikili sınıflandırma için kullanılır. Fakat tanh [-1,1] değer aralığında değerler alır ve orjin etrafında simetriktir.
$tanh(x) = 2sigmoid(2x) - 1$



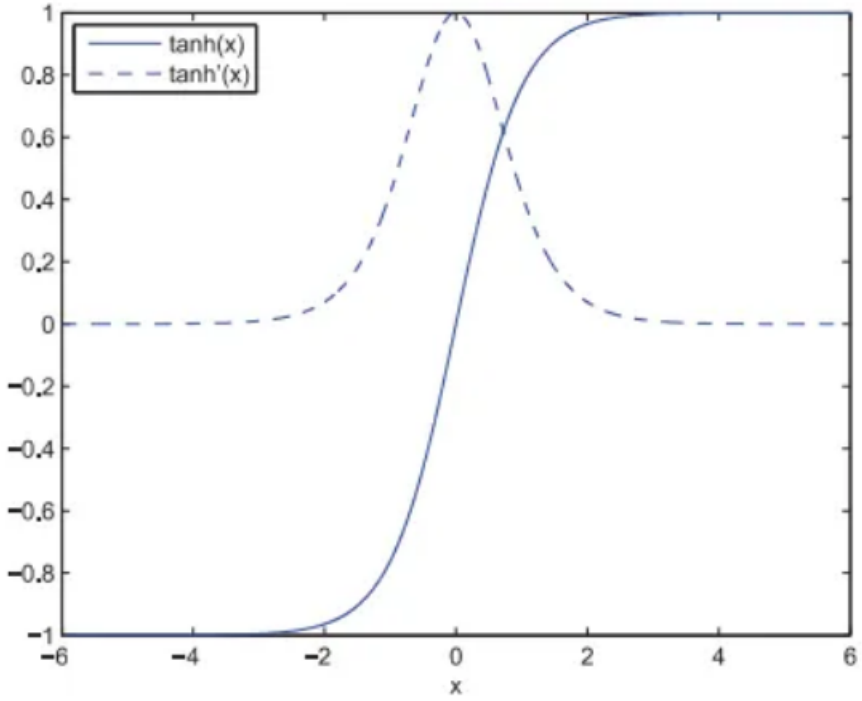
Resim 49: Lineer fonksiyon.

Grafikten gördüğümüz üzere, uç noktalarda türev sıfıra yakınsar. Yani tanh fonksiyonu da Vanishing Gradient probleminden muzdariptir. Model kurulacağı zaman katman sayısı dikkate alınmalıdır, çok katmanlı ağlarda tercih edilmemelidir.
Rectified Linear Unit (RELU): Bir diğer fonksiyonumuz Rectifier Linear Unit. Genellikle Convolutional Neural Network (CNN)'te ve ara katmanlarda çok sık kullanılan ReLU fonksiyonunun ana avantajı aynı anda tüm nöronları aktive etmemesidir. Yani bir nöron negatif değer üretirse, aktive edilmeyeceği anlamına gelir. ReLU'nun türevi sıfıra yakınsamadığı için önceki fonksiyonlarda gördüğümüz Vanishing Gradient probleminden etkilenmez. Fakat ReLU da mükemmel değildir.



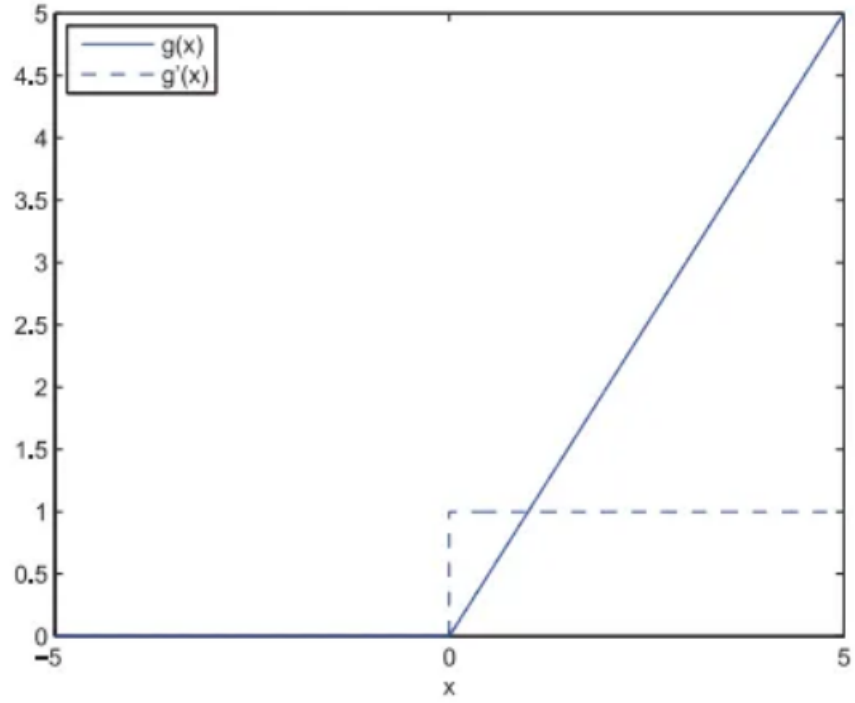
Resim 50: Sigmoid fonksiyonu.

Grafiğe baktığımızda, fonksiyonun türevinin negatif değerlerde sıfır olduğunu görürüz. Daha önceki fonksiyonlarda da bahsettiğimiz gibi bu durum Geri Yayılım (Back Propagation) esnasında parametrelerimizin güncellenmeyeceği, yani öğrenme işleminin olmayacağı anlamına gelir. Bu probleme Dying ReLU adı verilir. Bu problemi çözmek için relu fonksiyonlarının farklı varyasyonları geliştirilmiştir.[17]
Geçen haftaki çalışmamızda ölçeklendirmeyi tüm bağımlı X dizisi için yapmıştık fakat bu çok büyük değerler içeren (örneğin population) parametrelerin diğer parametreleri 0 a yakınsamasına yol açacaktı.Bunun için tüm parametreleri ayrı ayrı ölçeklendirip daha sonra X bağımlı değişken dizisine koyduk.



Resim 51: Hiperbolik Tanjant fonksiyonu.

Modelimize besleyeceğimiz değerleri bu şekilde ölçeklendirdik.Daha sonra en başarılı aktivasyon fonksiyonunu bulmak için grid search yöntemini uyguladık.



Resim 52: RELU fonksiyonu.

Bu şekilde tüm katmanlarda 3 farklı aktivasyon fonksiyonu deneyerek en düşük hata oranına sahip olanı bulmaya çalıştık. Fakat bu kodun dezavantajı tüm katmanlarda aynı aktivasyon fonksiyonunu kullandık. Farklı aktivasyon fonksiyonları için r2 score,mean absolute error, mean squared error değerleri aşağıdaki tabloda gösterilmiştir.

Tablo 1: Aktivasyon Fonksiyonlarının Performans Karşılaştırması

Akt Fonksiyonları	r2score	mae	mse	eğitim zamanı
Relu	0.732	0.054	0.010	7.72 saniye
Sigmoid	-0.042	0.092	0.039	13.45 saniye
Tanh	0.573	0.061	0.016	11.48 saniye

Tablo notları: Bu tablo, farklı aktivasyon fonksiyonlarının performans metriklerini göstermektedir.

```
#Ölçeklendirme-normalizasyon
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
newCasesAfghanistan = newCasesAfghanistan.values
newCasesAfghanistan = newCasesAfghanistan.reshape(-1, 1)
newCasesAfghanistan = scaler.fit_transform(newCasesAfghanistan)

newCasesSmoothedAfghanistan = newCasesSmoothedAfghanistan.values
newCasesSmoothedAfghanistan = newCasesSmoothedAfghanistan.reshape(-1, 1)
newCasesSmoothedAfghanistan = scaler.fit_transform(newCasesSmoothedAfghanistan)


diabetesPrevalenceAfghanistan = diabetesPrevalenceAfghanistan.values
diabetesPrevalenceAfghanistan = diabetesPrevalenceAfghanistan.reshape(-1, 1)
diabetesPrevalenceAfghanistan = scaler.fit_transform(diabetesPrevalenceAfghanistan)


cardiovascDeathRateAfghanistan = cardiovascDeathRateAfghanistan.values
cardiovascDeathRateAfghanistan = cardiovascDeathRateAfghanistan.reshape(-1, 1)
cardiovascDeathRateAfghanistan = scaler.fit_transform(cardiovascDeathRateAfghanistan)


PoPulationAfghanistan = PoPulationAfghanistan.values
PoPulationAfghanistan = PoPulationAfghanistan.reshape(-1, 1)
PoPulationAfghanistan = scaler.fit_transform(PoPulationAfghanistan)
```

Resim 53: Bağımlı değikenlerimizi ayrı ayrı ölçeklendirme.

```
param_grid = {
    'n_activation': ("relu","sigmoid","tanh")

}
```

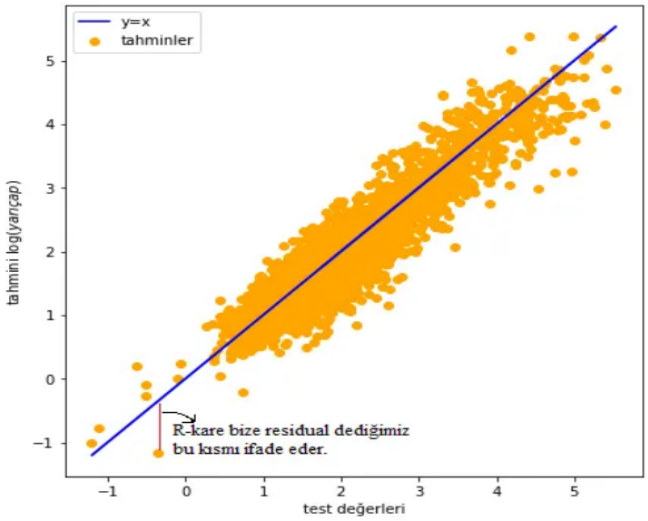
Resim 54: Grid Searh yöntemi için parametreler.

Modelimizin başarımı için kullandığımız metrikleri (Ölçütleri) incelemek çıkan sonuçları yorumlamamızda yardımcı olacaktır.Bu yüzden kullandığımız ölçütleri inceleyeceğiz.

```
best_score = float('inf')
best_params = None

for n_activation in param_grid['n_activation']:

  siniflandirici = Sequential()
  siniflandirici.add(Dense(units=6, activation=n_activation, input_dim=5))#ilk katman 5 giris
  siniflandirici.add(Dense(units=6, activation=n_activation))
  siniflandirici.add(Dense(units=6, activation=n_activation))
  siniflandirici.add(Dense(units=6, activation=n_activation))
  siniflandirici.add(Dense(units=1, activation=n_activation))  # Son katman, tek sınıf

  siniflandirici.compile(optimizer='adam', loss=binary_crossentropy, metrics=['accuracy'])

  siniflandirici.fit(X_train, y_train, batch_size=10, epochs=100)

  Y_pred_Ysa = siniflandirici.predict(X_test)

  mse=mean_squared_error(y_test, Y_pred_Ysa)

  if mse < best_score:
            best_score = mse
            best_params = {'n_activation': n_activation}
print("En iyi parametreler:", best_params)

print("En iyi skor:", best_score)
```

Resim 55: Grid Searh yöntemi uygulanması.

R^2:R^2, verilerin yerleştirilmiş regresyon hattına ne kadar yakın olduğunun istatistiksel bir ölçüsüdür. Ayrıca belirleme katsayısı veya çoklu regresyon için çoklu belirleme katsayısı olarak da bilinir. Daha basit bir dilde söylemek gerekirse R-kare, doğrusal regresyon modelleri için uygunluk ölçüsüdür. [18]



Resim 56: R^2 metriği.

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \tag{1}$$

(1) Burada, SS_{res} tahmin hatalarının karelerinin toplamını, SS_{tot} ise gerçek değerlerin ortalamadan sapmalarının karelerinin toplamını temsil eder.

Düzeltilmiş(Adjusted) R^2:Düzeltilmiş R2, modelin karmaşıklığını (bağımsız değişkenlerin sayısını) göz önünde bulundurarak R2R2 değerini ayarlar. Bu, modeldeki gereksiz değişkenlerin etkisini azaltır ve modelin genel performansını daha doğru bir şekilde değerlendirmeyi sağlar.[19] Düzeltilmiş R^2 denklemi:

$$\bar{R}^2 = 1 - \left(\frac{(1 - R^2)(n - 1)}{n - k - 1} \right)$$

n = gözlem sayısı
k = bağımsız değişken sayısı

Ortalama Mutlak Hata (Mean Absolute Error - MAE: Bir tahmin modelinin gerçek değerlerle ne kadar uyumlu olduğunu ölçen bir performans metriğidir. Bu metrik, tahmin edilen değerler ile gerçek değerler arasındaki

farkların mutlak değerlerinin ortalamasını verir. Yani, her tahminin gerçek değerden ne kadar sapma gösterdiğini ölçer ve bu sapmaların ortalamasını alır.

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |x_i - \bar{x}|$$

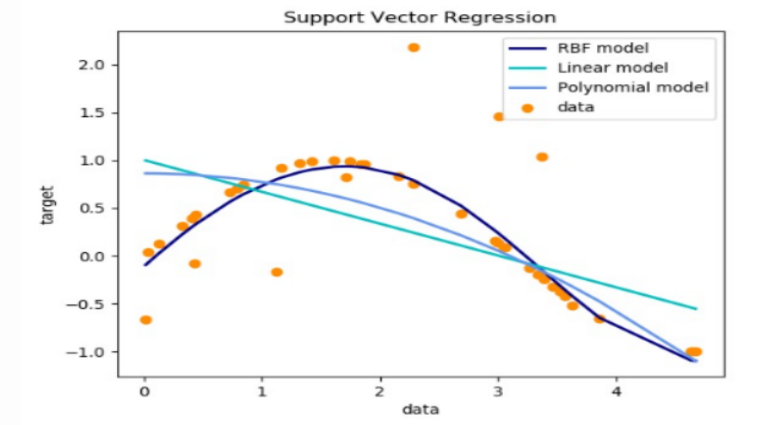
$n =$ gözlem sayısı
$x_i =$ veri noktasının değeri
$\bar{x} =$ veri noktalarının ortalaması

Ortalama Karesel Hata (MSE – Mean Squared Error):
Ortalama Karesel Hata, tahmin edilen değerler ile gerçek değerler arasındaki farkların karelerinin ortalaması alınarak hesaplanır. MSE, büyük hatalara daha fazla ağırlık verir, bu nedenle modelin büyük sapmalarını sert bir şekilde cezalandırır. Bu özellik, modelinizin büyük hatalar yapmasını özellikle önlemek istediğiniz durumlarda yararlıdır.[20]

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

$$n = \text{gözlem sayısı}$$
$$y_i = \text{gerçek değer}$$
$$\hat{y}_i = \text{tahmin edilen değer}$$

Projenin en sonunda SVR(Support Vector Regression) algoritması da çalışmalara eklenmiştir. Algoritmanın çıktılarını ve nasıl çalıştığını daha iyi anlamak için SVR makine öğrenmesi algoritması kısaca anlatılacaktır.
SVR(Support Vector Regression): Destek vektör algoritması ilk başta sınıflandırma için çıkmış bir algoritma olmasına rağmen regresyon için de kullanılmaktadır. Sınıflandırma işlemleri için SVM yani Support Vector Machine kullanılırken regresyon için de Support Vector Regression kullanılmaktadır.



Resim 57: Sigmoid fonksiyonu.

Amaç, bir marjın aralığına maksimum noktayı en küçük hata ile alabilecek şekilde doğru ya da eğriyi belirlemektir. Yani, Destek vektör regresyonu uyguladığımızda, çizeceğimiz aralığın maksimum noktayı içerisine almasını sağlamaktır. Burada doğrusal ve doğrusal olmayan olarak iki türlü SVR metodu bulunmaktadır.[21]
En son olarakta veri setimizdeki Afganistan verilerini ve tüm verileri(218 farklı ülke için toplam 55322 veri) kullanarak Random Forest , Neural Network ve Support Vector Regression algoritmaları ile modeller oluşturulmuştur. Algoritmalarla tüm veriler için model oluştururken eksik veri sorunu ile karşılaşıldı fakat bu sorun eksik verilerin bağlı oldukları sütun ortalamaları ile doldurulmasıyla kısmen çözüldü.

Tablo 2: MSE Karşılaştırması

	Random Forest	NN	SVR
Afganistan	1008.05	0.01458	0.01458
Tüm veriler	190374	0.00056	0.00330

Tablo notları: Bu tabloda, Afganistan ve tüm veriler için 3 farklı algoritmanın Mean squared error metriği karşılaştırılmıştır.

Tablo 3: MAE Karşılaştırması

	Random Forest	NN	SVR
Afganistan	18.7669.05	0.05067	0.08372
Tüm veriler	27.8525	0.00168	0.05029

Tablo notları: Bu tabloda, Afganistan ve tüm veriler için 3 farklı algoritmanın Mean absolute error metriği karşılaştırılmıştır.

Tablo 4: Eğitim süresi karşılaştırılması

	Random Forest	NN	SVR
Afganistan	0.21 saniye	8.93 saniye	0.01 saniye
Tüm veriler	12.69 saniye	863.59 saniye	1.82 saniye

Tablo notları: Bu tabloda, Afganistan ve tüm veriler için 3 farklı algoritmanın eğitim süreleri karşılaştırılmıştır.

6 Sonuç:

Sonuç olarak bu çalışmada 2 farklı veri kümesi ve 3 farklı yapay zeka algoritması kullanarak hastalık durumundaki vefat sayısı tahmin etmeye çalıştık.Yukarıdaki Tablo 2 Tablo 3 ve Tablo 4 de de görebileceğiniz üzere bizim veri setimizde en başarılı algoritma Yapay Sinir Ağları seçildi.

7 Teşekkürler:

Çalışmamın her aşamasında desteklerini hiçbir zaman esirgemeyen değerli hocam Emre GÜNGÖR'e teşekkürler.

Kaynakçalar

[1] BBC, "https://www.bbc.com/turkce/haberler-dunya-60702679," 11 Mart 2022.

[2] arcgis, "https://www.arcgis.com/apps/dashboards/bda7594740fd40299423467b48e9ecf6," 2024.

[3] C. Çılgın Et Al., "Sentiment analysis of public sensitivity to covid-19 vaccines on twitter by majority voting classifier-based machine learning twitter'da covid-19 aşılarına karşı kamu duyarlılığının çoğunluk oylama sınıflandırıcısı temelli makine öğrenmesi ile duygu analizi," *Journal of the Faculty of Engineering and Architecture of Gazi University*, pp. 1093–1104, 2023.

[4] N. S. ÖZEN, S. SARAÇ, and M. KOYUNCU, "Covid-19 vakalarının makine Öğrenmesi algoritmaları ile tahmini: Amerika birleşik devletleri Örneği," *Avrupa Bilim ve Teknoloji Dergisi*, no. 22, p. 134–139, 2021.

[5] H. M. Genc, Z. Cataltepe, and T. Pearson, "A new pca/ica based feature selection method," in *2007 IEEE 15th Signal Processing and Communications Applications*, pp. 1–4, IEEE, 2007.

[6] E. SÜTCÜ and P. SHAMS, "Türkiye'de covid-19 günlük vaka sayısının makine öğrenmesi algoritmaları ile tahmin edilmesi," *Anadolu Bil Meslek Yüksekokulu Dergisi*, vol. 16, no. 63, p. 197–213, 2021.

[7]

[8] E. Mathieu, H. Ritchie, L. Rodés-Guirao, C. Appel, C. Giattino, J. Hasell, B. Macdonald, S. Dattani, D. Beltekian, E. Ortiz-Ospina, and M. Roser, "Coronavirus pandemic (covid-19)," *Our World in Data*, 2020. https://ourworldindata.org/coronavirus.

[9] H. Candan, "Adım adım makine Öğrenmesi bölüm 4: Denetimli Öğrenme ve denetimsiz Öğrenme arasındaki fark."

[10] R. Kandar, "Regresyon (regression) - sınıflandırma(classification) nedir?."

[11] B. Daz, "Random forest algoritması (rassal orman) — machine learning [7]."

[12] B. Köseoğlu, "Model performansını değerlendirmek: Regresyon," 2021.

[13] Wikipedia, “Yapay sinir ağları,” 2024.

[14] Şadi Evren ŞEKER, “Neural network 3: Çok katmanlı yapay sinir ağları-youtube,” 2017.

[15] SlideServe, “Yapay sinir ağı (ysa),” 2014.

[16] P. D. S. Akkoyun, “Ders-8 yapay sinir ağları (python ile Örnek uygulama),” 2023.

[17] B. Ayten, “Yapay sinir ağlarında aktivasyon fonksiyonları,” 2021.

[18] Şevket Ay, “Model performansını değerlendirmek — metrikler,” 2020.

[19] E. Şirin, “R kare ve düzeltilmiş r kare,” 2017.

[20] U. CİNDİLOĞLU, “2- makine Öğrenmesinde performans metrikleri,” 2024.

[21] İlker Katkat, “Python - svr (destek vektör regresyonu),” 2020.