

KÜTAHYA SAĞLIK BİLİMLERİ ÜNİVERSİTESİ
Bilgisayar Mühendisliği



Yapay Zeka Dersi Proje Tasarım Raporu

Celal ALTIN

April 25, 2024

Bu çalışma raporu vizeye kadar olan yapay zeka dersi çalışmalarımın detaylarını içermektedir.

1. Giriş
2. Literatür Araştırması
3. Metodoloji
4. Kullanılacak Veriler
5. Beklenen Sonuçlar
6. Kaynakça

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: Dünya Covit Haritasi [2]



Dünya çapındaki bu problemi daha iyi anlayabilmek ve ileride olası hastalıklarda daha etkili mücadele edebilmek için bilgisayar(Makine öğrenmesi kulanılarak 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.

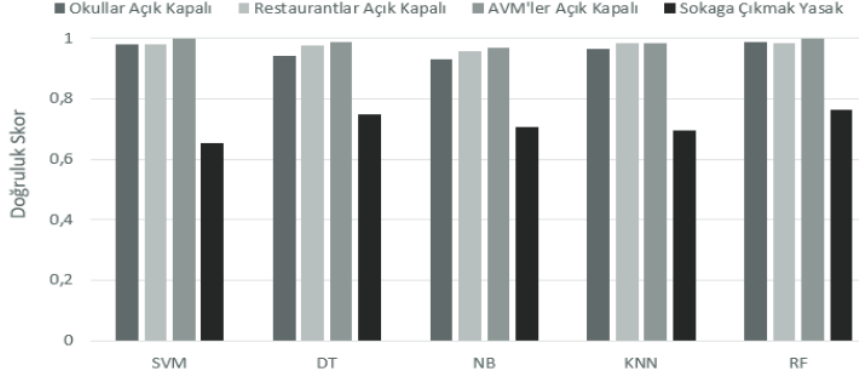
2 Literatür Araştırması

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, bölme 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 öznelik elde etme evresinde harcanan zamanı azaltmak için, öznelik 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].

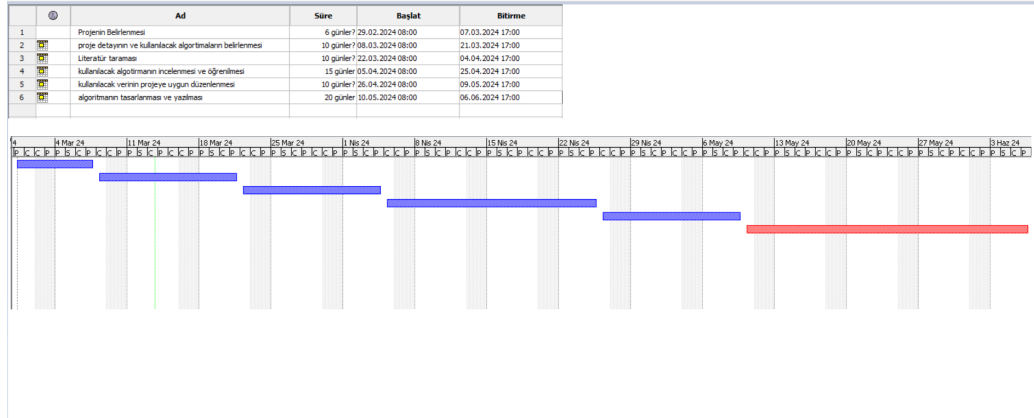
Resim 2: Doğruluk Tablosu



3 Metodoloji

Projeyi 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

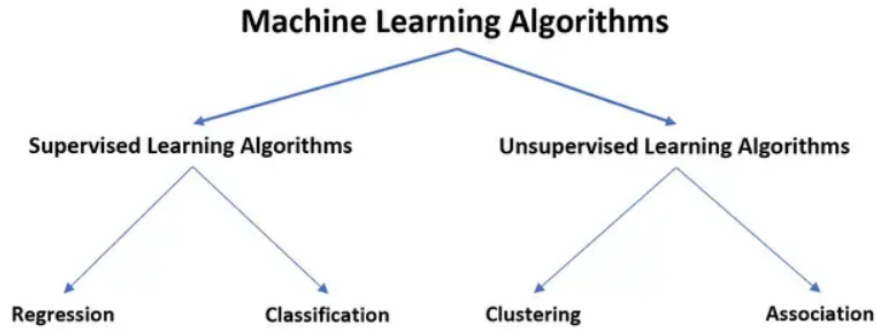
Projemde iki farklı veri kaynağından faydalandım.Bunlardan ilki T.C. Sağlık Bakanlığında alınmış verilerdir.[7].Verilerimiz 11.03.2020-31.05.2022 tarihleri arasında alınmış 813 farklı kayda sahiptir.Bu veriler Toplam Vaka, Günlük Vaka, Toplam Hasta, Günlük Hasta, Toplam Vefat, Günlük Vefat, Toplam iyileşen,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

5 Beklenen Sonuçlar

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

Çalışmamızda Random Forest algoritmasını kullanılacağından bahsedilmiştir. 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.

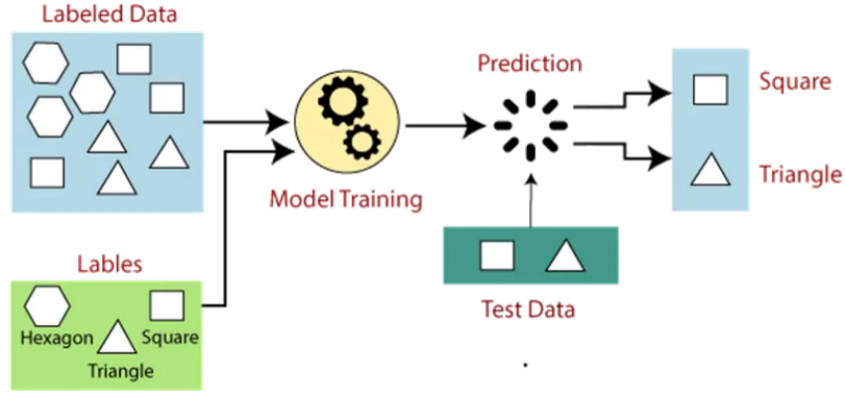
Resim 4: Machine Learning Algorithms



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]

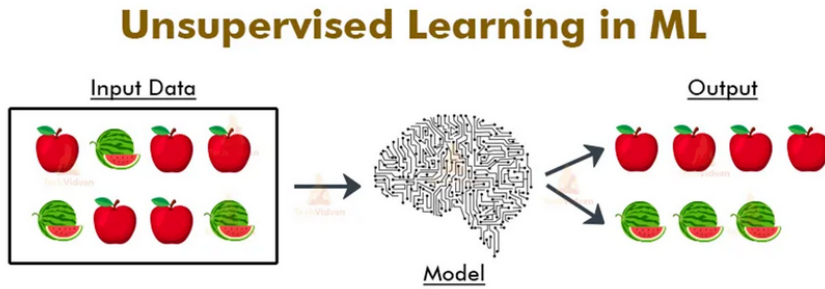
Resim 5: Denetimli Öğrenme



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 ?

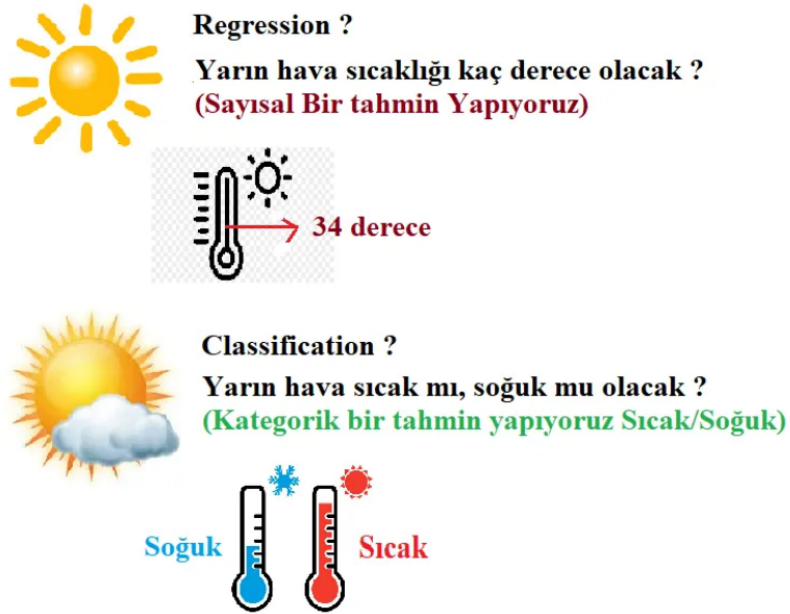
Regresyon bağımlı bir değişken ile bağımsız bir değişken arasındaki ilişkinin, ortadan kaldırılması için kullanılan istatistiksel bir yöntemdir. Evet,

regresyonun bu teorik açıklaması size karmaşık gelmiş olabilir. Gelin biz bunu daha basit haliyle açıklayalım. Buradaki değişkenler(x ve y arasında, $x \rightarrow$ deneyim yılı, $y \rightarrow$ 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 \rightarrow Maaş tahmini, yaşa göre boy tahmini, çalışma süresine göre alınan not tahmini gibi örnekler verebiliriz.

Sınıflandırma(Classification) Nedir ?

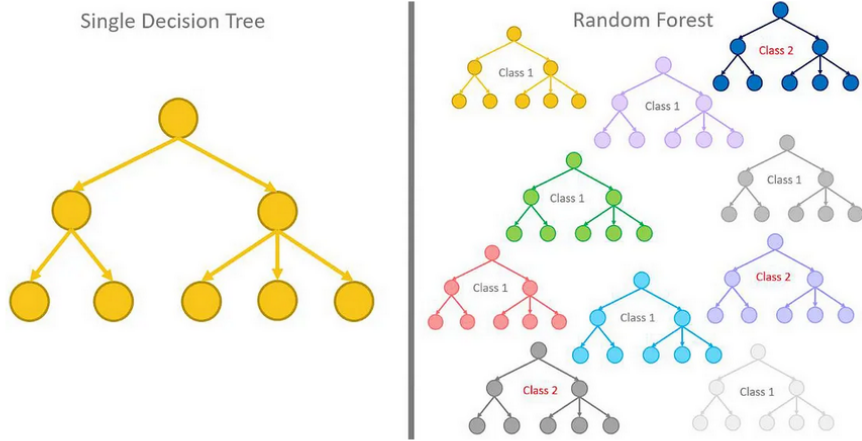
Regression problemlerinde tahmin edeceğimiz y kolonunda sayısal değerler vardı. Biz x değerlerini yani giriş değerini kullanarak sayısal tahminler yapmaya çalışıyorduk. Sınıflandırma problemlerinde ise x(giriş değerleri) değerlerini kullanarak kategorik olarak y sınıfını tahmin etmeye çalışırız.[10]

Resim 7: Regresyon ve Sınıflandırma



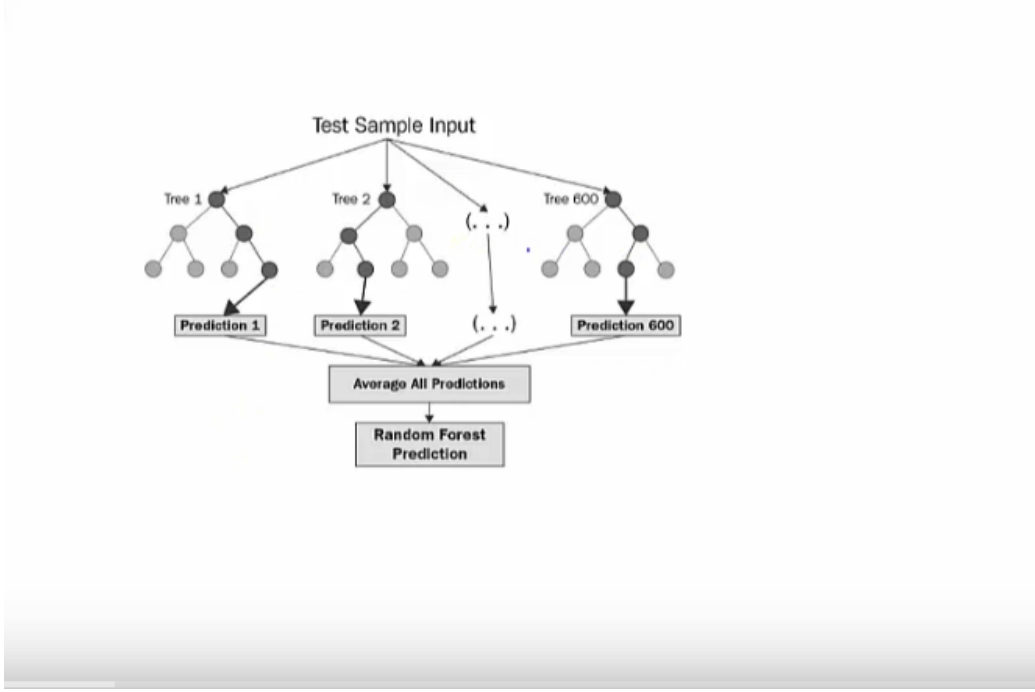
Random Forest (Rassal Orman) Algoritması: Sınıflandırma işlemi esnasında birden fazla karar ağacı üreterek sınıflandırma değerini yükseltmeyi hedefleyen bir algoritmadır. Bireysel olarak oluşturulan karar ağaçları bir araya gelerek karar ormanı oluşturur. Buradaki karar ağaçları bağlı olduğu veri setinden 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 algoritmasını uygulayacağım veriler aşağıda gösterilmiştir.

Resim 10: Covit Verileri

Tarih	Toplam Vaka	Günlük Vaka	Toplam Hasta	Günlük Hasta	Toplam Vefat	Günlük Vefat	Toplam İyileşen	Günlük İyileşen	Toplam Test	Günlük 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 kütüphaneleri 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 siliyoruz 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.

$$1 - \frac{SS_{res}}{SS_{tot}}$$

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]

$$\frac{1}{n} * \sum (y - y_{pred})^2$$

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)
```

	iso_code	continent	location	date	total_cases	new_cases	new_cases_smoothed	total_deaths	new_deaths	new_deaths_smoothed	...	male_smokers	hu
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 kullanacağımız 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_olders                                                       295505 non-null float64  
51   aged_70_olders                                                       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 Afganistan 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 değişkenlerini 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 değişkenimiz new deaths verilerini içeriyor. En son olarak bağımlı ve bağımsız değişkenlerin boyutlarını kontrol etmemin sebebi satır sayılarının aynı olması gerektiğindendir.

```
name: new_cases, target: new_deaths, dtype: float64
```

```
[15]: 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 = 4555)
```

```
[17]: param_grid = {
      'n_estimators': list(range(20, 1000, 10)),
      'max_depth': [10, 20]
      }
```

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.

```
best_score = float('inf')
best_params = None
for n_estimators in param_grid['n_estimators']:
    for max_depth in param_grid['max_depth']:

        # Özel modelinizin örneğini oluşturun ve parametreleri ayarlayın
        model = RandomForestRegressor( n_estimators=n_estimators, max_depth=max_depth) # Örneğin param1 ve param2 parametreleri olsun

        # Modeli eğit
        model.fit(X_train, y_train)

        # Modeli test et ve performans değerlendire
        y_pred = model.predict(X_test)
        mse = mean_squared_error(y_test, y_pred)

        # En iyi performansı sağlayan parametreleri güncelle
        if mse < best_score:
            best_score = mse
            best_params = {'n_estimators': n_estimators, 'max_depth': max_depth} # Örneğin param1 ve param2 parametreleri olsun

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

En iyi parametreler: {'n_estimators': 40, 'max_depth': 10}
En iyi skor: 945.3551678187163

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.

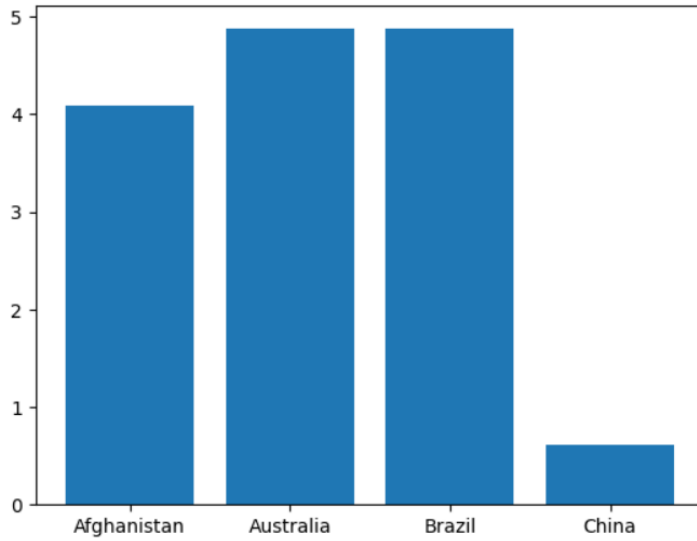
Aynı işlemi veri tabanımızdaki Avustralya, Brezilya, ve Çin içinde yaptık.Lakin eğitim başarımları ve test başarımları tatmin edici seviyede değildi.

```

[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 başarıımı	Test başarıımı
Afganistan	% 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.

5 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 aşılara 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. Rod  s-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   ğrenmesi b  l  m 4: Denetimli   ğrenme ve denetimsiz   ğrenme arasındaki fark.”
- [10] R. Kandar, “Regresyon (regression) - sınıflandırma(classification) nedir?.”
- [11] B. Daz, “Random forest algoritması (rassal orman) — machine learning [7].”
- [12] B. K  seoğlu, “Model performansını değ  rlendirmek: Regresyon,” 2021.