The dataset contains two folders, whereas one contains the data for the controls and one for the condition group. For each patient a csv file has been provided containing the actigraph data collected over time. The columns are: timestamp (one minute intervals), date (date of measurement), activity (activity measurement from the actigraph watch). In addition, the MADRS scores provided in the file "scores.csv". It contains the following columns; number (patient identifier), days (number of days of measurements), gender (1 or 2 for female or male), age (age in age groups), afftype (1: bipolar II, 2: unipolar depressive, 3: bipolar I), melanch (1: melancholia, 2: no melancholia), inpatient (1: inpatient, 2: outpatient), edu (education grouped in years), marriage (1: married or cohabiting, 2: single), work (1: working or studying, 2: unemployed/sick leave/pension), madrs1 (MADRS score when measurement started), madrs2 (MADRS when measurement stopped).

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

depression\_dataset = pd.read\_csv("scores.csv")

df = depression\_dataset.copy()

df.head()

number days gender age afftype melanch inpatient edu marriage work madrs1 madrs2

0 condition\_1 11 2 35-39 2.0 2.0 2.0 6-10 1.0 2.0 19.0 19.0

1 condition\_2 18 2 40-44 1.0 2.0 2.0 6-10 2.0 2.0 24.0 11.0

2 condition\_3 13 1 45-49 2.0 2.0 2.0 6-10 2.0 2.0 24.0 25.0

3 condition\_4 13 2 25-29 2.0 2.0 2.0 11-15 1.0 1.0 20.0 16.0

4 condition\_5 13 2 50-54 2.0 2.0 2.0 11-15 2.0 2.0 26.0 26.0

print(f" Veri setinin boyut sayısı: {df.ndim}\n",

f"Veri setinin boyut bilgisi: {df.shape}\n",

f"Veri setindeki toplam eleman sayısı: {df.size}\n")

Veri setinin boyut sayısı: 2

Veri setinin boyut bilgisi: (55, 12)

Veri setindeki toplam eleman sayısı: 660

df.info()

RangeIndex: 55 entries, 0 to 54

Data columns (total 12 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 number 55 non-null object

1 days 55 non-null int64

2 gender 55 non-null int64

3 age 55 non-null object

4 afftype 23 non-null float64

5 melanch 20 non-null float64

6 inpatient 23 non-null float64

7 edu 53 non-null object

8 marriage 23 non-null float64

9 work 23 non-null float64

10 madrs1 23 non-null float64

11 madrs2 23 non-null float64

dtypes: float64(7), int64(2), object(3)

memory usage: 5.3+ KB

df.describe().T

count mean std min 25% 50% 75% max

days 55.0 12.600000 2.491467 5.0 13.0 13.0 13.0 20.0

gender 55.0 1.454545 0.502519 1.0 1.0 1.0 2.0 2.0

afftype 23.0 1.739130 0.540824 1.0 1.0 2.0 2.0 3.0

melanch 20.0 1.950000 0.223607 1.0 2.0 2.0 2.0 2.0

inpatient 23.0 1.782609 0.421741 1.0 2.0 2.0 2.0 2.0

marriage 23.0 1.521739 0.510754 1.0 1.0 2.0 2.0 2.0

work 23.0 1.869565 0.344350 1.0 2.0 2.0 2.0 2.0

madrs1 23.0 22.739130 4.797892 13.0 18.5 24.0 26.0 29.0

madrs2 23.0 20.000000 4.729021 11.0 16.0 21.0 24.5 28.0

df.isnull().sum()

number 0

days 0

gender 0

age 0

afftype 32

melanch 35

inpatient 32

edu 2

marriage 32

work 32

madrs1 32

madrs2 32

dtype: int64

print(f"Number değerleri:\n{df['number'].value\_counts()}")

print("-" \* 20)

print(f"Days değerleri:\n{df['days'].value\_counts()}")

print("-" \* 20)

print(f"Gender değerleri:\n{df['gender'].value\_counts()}")

print("-" \* 20)

print(f"Age değerleri:\n{df['age'].value\_counts()}")

print("-" \* 20)

print(f"Afftype değerleri:\n{df['afftype'].value\_counts()}")

print("-" \* 20)

print(f"Melanch değerleri:\n{df['melanch'].value\_counts()}")

print("-" \* 20)

print(f"İnpatient değerleri:\n{df['inpatient'].value\_counts()}")

print("-" \* 20)

print(f"Edu değerleri:\n{df['edu'].value\_counts()}")

print("-" \* 20)

print(f"Marriage değerleri:\n{df['marriage'].value\_counts()}")

print("-" \* 20)

print(f"Work değerleri:\n{df['work'].value\_counts()}")

print("-" \* 20)

print(f"Madrs1 değerleri:\n{df['madrs1'].value\_counts()}")

print("-" \* 20)

print(f"Madrs2 değerleri:\n{df['madrs2'].value\_counts()}")

print("-" \* 20)

Number değerleri:

condition\_1 1

control\_19 1

control\_8 1

control\_9 1

control\_10 1

control\_11 1

control\_12 1

control\_13 1

control\_14 1

control\_15 1

control\_16 1

control\_17 1

control\_18 1

control\_20 1

control\_6 1

control\_21 1

control\_22 1

control\_23 1

control\_24 1

control\_25 1

control\_26 1

control\_27 1

control\_28 1

control\_29 1

control\_30 1

control\_31 1

control\_7 1

control\_5 1

condition\_2 1

condition\_14 1

condition\_3 1

condition\_4 1

condition\_5 1

condition\_6 1

condition\_7 1

condition\_8 1

condition\_9 1

condition\_10 1

condition\_11 1

condition\_12 1

condition\_13 1

condition\_15 1

control\_4 1

condition\_16 1

condition\_17 1

condition\_18 1

condition\_19 1

condition\_20 1

condition\_21 1

condition\_22 1

condition\_23 1

control\_1 1

control\_2 1

control\_3 1

control\_32 1

Name: number, dtype: int64

--------------------

Days değerleri:

13 31

14 6

11 3

9 3

16 3

8 3

12 2

18 1

7 1

5 1

20 1

Name: days, dtype: int64

--------------------

Gender değerleri:

1 30

2 25

Name: gender, dtype: int64

--------------------

Age değerleri:

45-49 9

50-54 9

35-39 7

25-29 7

30-34 7

20-24 6

40-44 5

60-64 2

65-69 2

55-59 1

Name: age, dtype: int64

--------------------

Afftype değerleri:

2.0 15

1.0 7

3.0 1

Name: afftype, dtype: int64

--------------------

Melanch değerleri:

2.0 19

1.0 1

Name: melanch, dtype: int64

--------------------

İnpatient değerleri:

2.0 18

1.0 5

Name: inpatient, dtype: int64

--------------------

Edu değerleri:

31

6-10 12

11-15 8

16-20 2

Name: edu, dtype: int64

--------------------

Marriage değerleri:

2.0 12

1.0 11

Name: marriage, dtype: int64

--------------------

Work değerleri:

2.0 20

1.0 3

Name: work, dtype: int64

--------------------

Madrs1 değerleri:

24.0 4

26.0 4

18.0 3

20.0 2

28.0 2

29.0 2

19.0 1

25.0 1

14.0 1

13.0 1

17.0 1

27.0 1

Name: madrs1, dtype: int64

--------------------

Madrs2 değerleri:

21.0 4

25.0 3

15.0 3

19.0 2

16.0 2

26.0 2

11.0 1

24.0 1

13.0 1

18.0 1

17.0 1

28.0 1

23.0 1

Name: madrs2, dtype: int64

--------------------

Aykırı değere ait bir sıkıntı yok gibi. Sonuö olarak değerlere baktığımız zaman hem az veriye sahip hem de tam olarak ilişkili değiller dolayısıyla aykırı değer yapısı için bir işlem gerçekleştirmeyeceğiz fakat null değerler için zaten veri de az olduğundan bir işlemler silsilesi gerçekleştireceğiz. Ortalam al veya sıfırla doldurmak yerine x'den bir önceki değeri al gibi temel uygulamaları gerçekleştireceğim...

# pip install missingno

import missingno as ms

ms.bar(df)

plt.show()

Burada eksik değerlerin varlığına dair bir görselleştirme gerçekleştirdik. Dolayısıyla bunun için 'missingno' kütüphanesini ilk olarak indirip ardından projeye dahil ettim. Şu anda eksik değerler daha açık gözüküyor.

ms.matrix(df)

plt.show()

ms.heatmap(df)

plt.show()

Yine 'NaN' değerlerin somut olarak gösterimleri. Bu grafiklerden yola çıkarak şimdi eksik değerler üzerinden işlemler yapacağım...

Şimdi burada değişkenleri gözlemleyerek bazılarını ortalama ile bazılarını sıfır ile doldurma veya silme işlemi gerçekleştireceğim..

df.head(10)

number days gender age afftype melanch inpatient edu marriage work madrs1 madrs2

0 condition\_1 11 2 35-39 2.0 2.0 2.0 6-10 1.0 2.0 19.0 19.0

1 condition\_2 18 2 40-44 1.0 2.0 2.0 6-10 2.0 2.0 24.0 11.0

2 condition\_3 13 1 45-49 2.0 2.0 2.0 6-10 2.0 2.0 24.0 25.0

3 condition\_4 13 2 25-29 2.0 2.0 2.0 11-15 1.0 1.0 20.0 16.0

4 condition\_5 13 2 50-54 2.0 2.0 2.0 11-15 2.0 2.0 26.0 26.0

5 condition\_6 7 1 35-39 2.0 2.0 2.0 6-10 1.0 2.0 18.0 15.0

6 condition\_7 11 1 20-24 1.0 NaN 2.0 11-15 2.0 1.0 24.0 25.0

7 condition\_8 5 2 25-29 2.0 NaN 2.0 11-15 1.0 2.0 20.0 16.0

8 condition\_9 13 2 45-49 1.0 NaN 2.0 6-10 1.0 2.0 26.0 26.0

9 condition\_10 9 2 45-49 2.0 2.0 2.0 6-10 1.0 2.0 28.0 21.0

df[df.isnull().any(axis=1)]

number days gender age afftype melanch inpatient edu marriage work madrs1 madrs2

6 condition\_7 11 1 20-24 1.0 NaN 2.0 11-15 2.0 1.0 24.0 25.0

7 condition\_8 5 2 25-29 2.0 NaN 2.0 11-15 1.0 2.0 20.0 16.0

8 condition\_9 13 2 45-49 1.0 NaN 2.0 6-10 1.0 2.0 26.0 26.0

23 control\_1 8 2 25-29 NaN NaN NaN NaN NaN NaN NaN NaN

24 control\_2 20 1 30-34 NaN NaN NaN NaN NaN NaN NaN

25 control\_3 12 2 30-34 NaN NaN NaN NaN NaN NaN NaN

26 control\_4 13 1 25-29 NaN NaN NaN NaN NaN NaN NaN

27 control\_5 13 1 30-34 NaN NaN NaN NaN NaN NaN NaN

28 control\_6 13 1 25-29 NaN NaN NaN NaN NaN NaN NaN

29 control\_7 13 1 20-24 NaN NaN NaN NaN NaN NaN NaN

30 control\_8 13 2 40-44 NaN NaN NaN NaN NaN NaN NaN

31 control\_9 13 2 30-34 NaN NaN NaN NaN NaN NaN NaN

32 control\_10 8 1 30-34 NaN NaN NaN NaN NaN NaN NaN

33 control\_11 13 1 45-49 NaN NaN NaN NaN NaN NaN NaN

34 control\_12 14 1 60-64 NaN NaN NaN NaN NaN NaN NaN

35 control\_13 13 1 50-54 NaN NaN NaN NaN NaN NaN NaN NaN

36 control\_14 13 1 50-54 NaN NaN NaN NaN NaN NaN NaN

37 control\_15 11 1 45-49 NaN NaN NaN NaN NaN NaN NaN

38 control\_16 13 2 40-44 NaN NaN NaN NaN NaN NaN NaN

39 control\_17 9 1 45-49 NaN NaN NaN NaN NaN NaN NaN

40 control\_18 13 2 20-24 NaN NaN NaN NaN NaN NaN NaN

41 control\_19 13 1 50-54 NaN NaN NaN NaN NaN NaN NaN

42 control\_20 13 1 35-39 NaN NaN NaN NaN NaN NaN NaN

43 control\_21 8 1 50-54 NaN NaN NaN NaN NaN NaN NaN

44 control\_22 13 1 25-29 NaN NaN NaN NaN NaN NaN NaN

45 control\_23 13 1 20-24 NaN NaN NaN NaN NaN NaN NaN

46 control\_24 13 2 20-24 NaN NaN NaN NaN NaN NaN NaN

47 control\_25 13 1 65-69 NaN NaN NaN NaN NaN NaN NaN

48 control\_26 13 1 35-39 NaN NaN NaN NaN NaN NaN NaN

49 control\_27 13 2 50-54 NaN NaN NaN NaN NaN NaN NaN

50 control\_28 16 2 45-49 NaN NaN NaN NaN NaN NaN NaN

51 control\_29 13 2 50-54 NaN NaN NaN NaN NaN NaN NaN

52 control\_30 9 2 35-39 NaN NaN NaN NaN NaN NaN NaN

53 control\_31 13 1 20-24 NaN NaN NaN NaN NaN NaN NaN

54 control\_32 14 2 25-29 NaN NaN NaN NaN NaN NaN NaN

Gözlemlediğimiz zaman eksik değerlerin sadece 'number' değişkenindeki 'control' değerlerinde var olduğunu görüyoruz. Yani bir bakıma büyük ihtimalle 'control' değerlerinde bu değişkenlere ihtiyaç duyulmamış olabilir. Yani veri hakkında daha fazla gözlem yapıp ona göre muamele gerçekleştirilmesi en doğrusu olacaktır.. Ancak ders ve öğrenme projesi olduğu için ben bu değerlere uygulama yapacağım..

df['afftype'].fillna(0, inplace=True)

df['afftype'][23:40]

23 0.0

24 0.0

25 0.0

26 0.0

27 0.0

28 0.0

29 0.0

30 0.0

31 0.0

32 0.0

33 0.0

34 0.0

35 0.0

36 0.0

37 0.0

38 0.0

39 0.0

Name: afftype, dtype: float64

İlk olarak "afftype" yani kişilerin bipolar olup olmadığını gözlemlediğimiz veri üzerinden ilerledim. Bu değerlere sıfır atamamın sebebi zaten 'condition'da bireylere teshiş'in konulmasından mütevellit. Bunun yanı sıra 0 olmasıyla birlikte ya bipolardır ya değildir sorgulamasına bilinmiyor yapısını da katabilir.

df["melanch"].fillna(0, inplace = True)

df["melanch"][25:40]

25 0.0

26 0.0

27 0.0

28 0.0

29 0.0

30 0.0

31 0.0

32 0.0

33 0.0

34 0.0

35 0.0

36 0.0

37 0.0

38 0.0

39 0.0

Name: melanch, dtype: float64

df['inpatient'].fillna(0, inplace=True)

df['inpatient'][23:40]

23 0.0

24 0.0

25 0.0

26 0.0

27 0.0

28 0.0

29 0.0

30 0.0

31 0.0

32 0.0

33 0.0

34 0.0

35 0.0

36 0.0

37 0.0

38 0.0

39 0.0

Name: inpatient, dtype: float64

df['marriage'].fillna(0, inplace=True)

df['marriage'][23:40]

23 0.0

24 0.0

25 0.0

26 0.0

27 0.0

28 0.0

29 0.0

30 0.0

31 0.0

32 0.0

33 0.0

34 0.0

35 0.0

36 0.0

37 0.0

38 0.0

39 0.0

Name: marriage, dtype: float64

df['work'].fillna(0, inplace=True)

df['work'][23:40]

23 0.0

24 0.0

25 0.0

26 0.0

27 0.0

28 0.0

29 0.0

30 0.0

31 0.0

32 0.0

33 0.0

34 0.0

35 0.0

36 0.0

37 0.0

38 0.0

39 0.0

Name: work, dtype: float64

df['madrs1'].fillna(df["madrs1"].mean(), inplace = True)

df['madrs1'][23:40]

23 22.73913

24 22.73913

25 22.73913

26 22.73913

27 22.73913

28 22.73913

29 22.73913

30 22.73913

31 22.73913

32 22.73913

33 22.73913

34 22.73913

35 22.73913

36 22.73913

37 22.73913

38 22.73913

39 22.73913

Name: madrs1, dtype: float64

df['madrs2'].fillna(df["madrs2"].mean(), inplace = True)

df['madrs2'][23:40]

23 20.0

24 20.0

25 20.0

26 20.0

27 20.0

28 20.0

29 20.0

30 20.0

31 20.0

32 20.0

33 20.0

34 20.0

35 20.0

36 20.0

37 20.0

38 20.0

39 20.0

Name: madrs2, dtype: float64

df.isnull().sum()

number 0

days 0

gender 0

age 0

afftype 0

melanch 0

inpatient 0

edu 2

marriage 0

work 0

madrs1 0

madrs2 0

dtype: int64

Evet böylelikle sadece bir ve ikilerden oluşan verilerin 'control' parametresine sıfır atadım. 'madrs1' ve 'madrs2' verilerini ise ortalamalar ile doldurdum. Yukarıda da bahsettiğim gibi bu işlemlerin gerekliliği tartışılabilir ve araştırmaya açıktır. Fakat uygulama ve öğrenme projesinde bunları atlamak istemedim.

from sklearn.preprocessing import LabelEncoder

lbe = LabelEncoder()

lbe.fit\_transform(df["age"])

df\_new = pd.get\_dummies(df, columns=["age"], prefix = ["age"])

df\_new.head()

number days gender afftype melanch inpatient edu marriage work madrs1 ... age\_20-24 age\_25-29 age\_30-34 age\_35-39 age\_40-44 age\_45-49 age\_50-54 age\_55-59 age\_60-64 age\_65-69

0 condition\_1 11 2 2.0 2.0 2.0 6-10 1.0 2.0 19.0 ... 0 0 0 1 0 0 0 0 0 0

1 condition\_2 18 2 1.0 2.0 2.0 6-10 2.0 2.0 24.0 ... 0 0 0 0 1 0 0 0 0 0

2 condition\_3 13 1 2.0 2.0 2.0 6-10 2.0 2.0 24.0 ... 0 0 0 0 0 1 0 0 0 0

3 condition\_4 13 2 2.0 2.0 2.0 11-15 1.0 1.0 20.0 ... 0 1 0 0 0 0 0 0 0 0

4 condition\_5 13 2 2.0 2.0 2.0 11-15 2.0 2.0 26.0 ... 0 0 0 0 0 0 1 0 0 0

5 rows × 21 columns

Evet ek olarak burada bir dönüşüm işlemi gerçekleştirdik. Kategorik değişken olarak 'age' sütununu verimizde kullanmak ve modelimize dahil etmek için numeric bir yapıya çevirdik. Şimdi burada bahsetmek istediğim bir kaç nokta var. İlk olarak dönüşüm işlemini LabelEncoder() fonksiyonuyla sağlıyoruz. Bizim 'age' kategorik değişkenimiz ikiden fazla parametre barındırdığı için bu işlem 1,2,3,4 ... şeklinde sayısal değer atamasıyla devam edecek. Fakat burada dummy tuzağını engellemek için de 'get\_dummies' yapısını uyguluyoruz. Genel mantık ise hangi değer varsa 1 olarak geçiyor diğerleri ise 0 olarak geçiyor.

df\_new.drop(["number", "edu"], axis = 1, inplace = True)

df\_new.head()

days gender afftype melanch inpatient marriage work madrs1 madrs2 age\_20-24 age\_25-29 age\_30-34 age\_35-39 age\_40-44 age\_45-49 age\_50-54 age\_55-59 age\_60-64 age\_65-69

0 11 2 2.0 2.0 2.0 1.0 2.0 19.0 19.0 0 0 0 1 0 0 0 0 0 0

1 18 2 1.0 2.0 2.0 2.0 2.0 24.0 11.0 0 0 0 0 1 0 0 0 0 0

2 13 1 2.0 2.0 2.0 2.0 2.0 24.0 25.0 0 0 0 0 0 1 0 0 0 0

3 13 2 2.0 2.0 2.0 1.0 1.0 20.0 16.0 0 1 0 0 0 0 0 0 0 0

4 13 2 2.0 2.0 2.0 2.0 2.0 26.0 26.0 0 0 0 0 0 0 1 0 0 0

Evet son olarak iki kategorisel değişkenimizi de yok ederek veri setini makine öğrenmesine uygun hale getirmiş oldum. Sadece sayısal değişkenler mevcut.. Artık bağımlı ve bağımsız değişkenlerimizi belirleyebilir ardından modellere geçebiliriz..

df\_new["madrs3"] = df\_new["madrs1"] - df["madrs2"]

df\_new.head(10)

days gender afftype melanch inpatient marriage work madrs1 madrs2 age\_20-24 age\_25-29 age\_30-34 age\_35-39 age\_40-44 age\_45-49 age\_50-54 age\_55-59 age\_60-64 age\_65-69 madrs3

0 11 2 2.0 2.0 2.0 1.0 2.0 19.0 19.0 0 0 0 1 0 0 0 0 0 0 0.0

1 18 2 1.0 2.0 2.0 2.0 2.0 24.0 11.0 0 0 0 0 1 0 0 0 0 0 13.0

2 13 1 2.0 2.0 2.0 2.0 2.0 24.0 25.0 0 0 0 0 0 1 0 0 0 0 -1.0

3 13 2 2.0 2.0 2.0 1.0 1.0 20.0 16.0 0 1 0 0 0 0 0 0 0 0 4.0

4 13 2 2.0 2.0 2.0 2.0 2.0 26.0 26.0 0 0 0 0 0 0 1 0 0 0 0.0

5 7 1 2.0 2.0 2.0 1.0 2.0 18.0 15.0 0 0 0 1 0 0 0 0 0 0 3.0

6 11 1 1.0 0.0 2.0 2.0 1.0 24.0 25.0 1 0 0 0 0 0 0 0 0 0 -1.0

7 5 2 2.0 0.0 2.0 1.0 2.0 20.0 16.0 0 1 0 0 0 0 0 0 0 0 4.0

8 13 2 1.0 0.0 2.0 1.0 2.0 26.0 26.0 0 0 0 0 0 1 0 0 0 0 0.0

9 9 2 2.0 2.0 2.0 1.0 2.0 28.0 21.0 0 0 0 0 0 1 0 0 0 0 7.0

Burada bağımsız değişkne olarak alacağım "madrs" puanı için bir işlem gerçekleştirdim. İlk olarak tepedeki veri seti açıklamasında 'madrs1'in bu ölçüm başladığında kişiden alınan puan olduğu 'madrs2'nin ise ölçüm bittiğinde kullanıcıda var olan puan olduğu belirtiliyor. Dolayısıyla iki ölçümün farkının ölçülen kişi için niteleyici bir değer taşıdığını düşünmemden mütevellit bu işlemi gerçekleştirdim. Şimdi bu değerleri silip işlemleri 'madrs3' üzerinden ilerleteceğim...

y\_df = df\_new[["madrs3"]]

df\_new.drop(["madrs1", "madrs2", "madrs3"], axis = 1, inplace = True)

X\_df = df\_new

X\_df.head()

days gender afftype melanch inpatient marriage work age\_20-24 age\_25-29 age\_30-34 age\_35-39 age\_40-44 age\_45-49 age\_50-54 age\_55-59 age\_60-64 age\_65-69

0 11 2 2.0 2.0 2.0 1.0 2.0 0 0 0 1 0 0 0 0 0 0

1 18 2 1.0 2.0 2.0 2.0 2.0 0 0 0 0 1 0 0 0 0 0

2 13 1 2.0 2.0 2.0 2.0 2.0 0 0 0 0 0 1 0 0 0 0

3 13 2 2.0 2.0 2.0 1.0 1.0 0 1 0 0 0 0 0 0 0 0

4 13 2 2.0 2.0 2.0 2.0 2.0 0 0 0 0 0 0 1 0 0 0

y\_df.head()

madrs3

0 0.0

1 13.0

2 -1.0

3 4.0

4 0.0

from sklearn.model\_selection import train\_test\_split, GridSearchCV

from sklearn.metrics import mean\_squared\_error, r2\_score

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_df, y\_df, test\_size = 0.30, random\_state = 42)

Lasso Regresyon Modelleme

Model ve tahmin bölümü:

from sklearn.linear\_model import Lasso

lasso\_model = Lasso(alpha = 0.1).fit(X\_train, y\_train)

print(lasso\_model.coef\_) #Katsayılar..

print("-"\*20)

print(lasso\_model.intercept\_) #Sabitler

[-0.00465593 0.73493616 -2.59659097 -0.26487977 -0.72882949 1.50801575

1.92480657 0. 0. 0. 0. 0.37100519

-0. -0. -0. 0. -0. ]

--------------------

[1.75305313]

y\_pred = lasso\_model.predict(X\_test)

np.sqrt(mean\_squared\_error(y\_test, y\_pred))

# İlkel tahmin modelimiz: 3.3752907578513645

2.311594984671114

lasso = Lasso()

lambdas = 10 \*\* np.linspace(10,-2,100) \* 0.5

katsayilar = []

for i in lambdas:

lasso.set\_params(alpha=i)

lasso.fit(X\_train,y\_train)

katsayilar.append(lasso.coef\_)

ax = plt.gca()

ax.plot(lambdas \* 2, katsayilar)

ax.set\_xscale('log')

plt.axis("tight")

plt.xlabel("alpha")

plt.ylabel("weights")

plt.show()

Model Tunning bölümü:

from sklearn.linear\_model import LassoCV

lasso = MultiTaskLassoCV().fit(X\_train, y\_train)

# ?lasso

lasso\_params = {"n\_alphas": [1,5,10,20,50,100],

"max\_iter": [1000,3000,7000,10000]}

lasso\_cv\_model = GridSearchCV(lasso, lasso\_params, cv = 10, n\_jobs=-1, verbose=2)

lasso\_cv\_model.fit(X\_train, y\_train)

print(f"En iyi parametreler: {str(lasso\_cv\_model.best\_params\_)}")

Fitting 10 folds for each of 24 candidates, totalling 240 fits

En iyi parametreler: {'max\_iter': 1000, 'n\_alphas': 5}

from sklearn.linear\_model import LassoCV

lasso\_tuned = LassoCV(max\_iter=1000, n\_alphas=5, ).fit(X\_train, y\_train)

y\_pred = lasso\_tuned.predict(X\_test)

np.sqrt(mean\_squared\_error(y\_test, y\_pred))

2.0099735403956025

Classification and Regression Trees - CART Modelleme

Model ve tahmin bölümü:

from sklearn.tree import DecisionTreeRegressor

cart\_model = DecisionTreeRegressor().fit(X\_train, y\_train)

y\_pred = cart\_model.predict(X\_test)

np.sqrt(mean\_squared\_error(y\_test, y\_pred))

2.7865221840769436

Model Tunning bölümü:

# ?cart\_model

cart\_model = DecisionTreeRegressor().fit(X\_train, y\_train)

cart\_params = {"max\_depth": [1,2,3,5,10],

"min\_samples\_split": [2, 5, 10, 20, 50],

"min\_samples\_leaf": [2, 5, 10, 20, 50]}

cart\_cv\_model = GridSearchCV(cart\_model, cart\_params, cv= 10, n\_jobs=-1, verbose=2).fit(X\_train, y\_train)

print(f"En iyi parametrelerimiz: {str(cart\_cv\_model.best\_params\_)}")

Fitting 10 folds for each of 125 candidates, totalling 1250 fits

En iyi parametrelerimiz: {'max\_depth': 1, 'min\_samples\_leaf': 2, 'min\_samples\_split': 50}

cart\_tuned = DecisionTreeRegressor(max\_depth=1, min\_samples\_split=2, min\_samples\_leaf=50).fit(X\_train, y\_train)

y\_pred = cart\_tuned.predict(X\_test)

np.sqrt(mean\_squared\_error(y\_test, y\_pred))

2.0099735403956025

Random Forests - RF Modeli

Model ve Tahmin Bölümü:

from sklearn.ensemble import RandomForestRegressor

from warnings import filterwarnings

filterwarnings('ignore')

rf\_model = RandomForestRegressor()

rf\_model.fit(X\_train, y\_train)

y\_pred = rf\_model.predict(X\_test)

np.sqrt(mean\_squared\_error(y\_test, y\_pred))

2.3601547009438626

Model Tunning bölümü:

# ?rf\_model

rf\_params = {'max\_depth': list(range(1,10)),

'max\_features': [3,5,10,20,50],

'n\_estimators': [100,200,500,1000,2000]}

rf\_model\_cv = GridSearchCV(rf\_model, rf\_params, cv=10, n\_jobs=-1, verbose=2).fit(X\_train, y\_train)

print(f"En iyi parametreler: {str(rf\_model\_cv.best\_params\_)}")

Fitting 10 folds for each of 225 candidates, totalling 2250 fits

En iyi parametreler: {'max\_depth': 1, 'max\_features': 3, 'n\_estimators': 100}

rf\_tuned = RandomForestRegressor(max\_depth=1, max\_features=3, n\_estimators = 100)

rf\_tuned.fit(X\_train, y\_train)

y\_pred = rf\_tuned.predict(X\_test)

np.sqrt(mean\_squared\_error(y\_test, y\_pred))

2.0042316448633275

Gradient Boosting Machines - GBM Modeli

Model ve Tahmin bölümü:

from sklearn.ensemble import GradientBoostingRegressor

#Modeli kurup fit ediyoruz

gbm\_model = GradientBoostingRegressor()

gbm\_model.fit(X\_train, y\_train)

#Ardından tahmin yapısını oluşturup ilkel karşılaştırma sonucunu elde ediyoruz.

y\_pred = gbm\_model.predict(X\_test)

np.sqrt(mean\_squared\_error(y\_test, y\_pred))

2.282629665508601

Model Tunning Bölümü:

gbm\_params = {"learning\_rate": [0.001,0.01,0.1,0.2],

"max\_depth": [3,5,10,50,100],

"n\_estimators": [200,500,1000,2000],

"subsample": [1,0.5,0.75]}

gbm = GradientBoostingRegressor()

gbm\_cv\_model = GridSearchCV(gbm, gbm\_params, cv = 10, n\_jobs=-1, verbose=2).fit(X\_train, y\_train)

print(f"En iyi parametrelerimiz: {str(gbm\_cv\_model.best\_params\_)}")

Fitting 10 folds for each of 240 candidates, totalling 2400 fits

En iyi parametrelerimiz: {'learning\_rate': 0.001, 'max\_depth': 100, 'n\_estimators': 200, 'subsample': 0.5}

gbm\_tuned = GradientBoostingRegressor(learning\_rate=0.001, max\_depth=100, n\_estimators=200, subsample=0.5)

gbm\_tuned.fit(X\_train, y\_train)

y\_pred = gbm\_tuned.predict(X\_test)

np.sqrt(mean\_squared\_error(y\_test, y\_pred))

2.0499438318763294

lasso\_predict = lasso\_tuned.predict(X\_test)

cart\_predict = cart\_tuned.predict(X\_test)

rf\_predict = rf\_tuned.predict(X\_test)

gbm\_predict = gbm\_tuned.predict(X\_test)

Tune edilmiş modeller üzerinde bir kaç metriğin toplu halde sorgulanması;

from sklearn.metrics import mean\_squared\_error, r2\_score, mean\_absolute\_error

predict = [lasso\_predict, cart\_predict, rf\_predict, gbm\_predict]

algoritma\_adlari = ["Lasso Regresypn","Cart Regresyon","Random Forests Regresyon", "Gradient Boosting Regresyon"]

def metrics(y\_pred):

mae = mean\_absolute\_error(y\_test, y\_pred)

mse = mean\_squared\_error(y\_test, y\_pred)

rmse = np.sqrt(mse)

r2 = r2\_score(y\_test, y\_pred)

data = [mae, mse, rmse, r2]

return data

seriler = []

metric\_s = ["Mean Absolute Error", "Mean squared Error", "Root Mean Squared Error", "R2"]

for i in predict:

data = metrics(i)

seriler.append(data)

df\_df = pd.DataFrame(data=seriler, index=algoritma\_adlari, columns = metric\_s)

pd.set\_option('display.colheader\_justify', 'center')

print(df\_df.to\_string())

Mean Absolute Error Mean squared Error Root Mean Squared Error R2

Lasso Regresypn 1.083255 4.039994 2.009974 -0.017395

Cart Regresyon 1.083255 4.039994 2.009974 -0.017395

Random Forests Regresyon 1.083853 4.016944 2.004232 -0.011591

Gradient Boosting Regresyon 1.098019 4.202270 2.049944 -0.058261