T.C.

BİLECİK ŞEYH EDEBALİ ÜNİVERSİTESİ

Mühendislik Fakültesi

Elektrik Elektronik Mühendisliği

Veri Madenciliği Final Ödevi

Mete AKYOL

70057153042

Dr. Öğr. Üyesi Ümit Çiğdem TURHAL

BİLECİK, 2022

İÇİNDEKİLER

Bu çalışmada mahsül tipi, sıcaklık, hava durumu gibi değerlere göre su ihtiyacı tespit edilmiştir. Bu çalışmada yazılım dili olarak Python, derleyeci olarak ise Jupyter notebook kullanılmıştır.

1.VERİTABANI AÇIKLAMASI

Veri tabanı Crop type (Mahsül tipi), Soil type (Toprak tipi), Region (Alan),Temperature (Sıcaklık), Weather Condition(Hava durumu) ve Water Requirement (Su ihtiyaci) bilgilerini içeren csv uzantılı bir excel tablosudur. Verisetimizde toplam 2880 satır ve 6 adet sütün bulunmaktadır. Ayrıca verisetimizin 5 adeti kategorik 1 adeti ise sayısal değerlerden oluşmaktadır.

```
df.info() #Datasetimizde herhangi bir eksik veri yok.
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2880 entries, 0 to 2879
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   CROP TYPE          2880 non-null   object 
 1   SOIL TYPE          2880 non-null   object 
 2   REGION             2880 non-null   object 
 3   TEMPERATURE        2880 non-null   object 
 4   WEATHER CONDITION  2880 non-null   object 
 5   WATER REQUIREMENT  2880 non-null   float64
dtypes: float64(1), object(5)
memory usage: 135.1+ KB
```

Şekil 1.1 Kategorik ve sayısal değerler.

1.1 Veritabanı Kaç Nesneden Oluşuyor?

Veri setimiz Crop type (Mahsül tipi), Soil type (Toprak tipi), Region (Alan),Temperature (Sıcaklık), Weather Condition(Hava durumu) ve Water Requirement (Su ihtiyaci) olmak üzere

toplamda 6 adet nesneden oluşmaktadır.

	CROP TYPE	SOIL TYPE	REGION	TEMPERATURE	WEATHER CONDITION	WATER REQUIREMENT
0	BANANA	DRY	DESERT	10-20	NORMAL	8.750
1	BANANA	DRY	DESERT	10-20	SUNNY	10.250
2	BANANA	DRY	DESERT	10-20	WINDY	9.650
3	BANANA	DRY	DESERT	10-20	RAINY	0.750
4	BANANA	DRY	DESERT	20-30	NORMAL	9.850
...	...	...	...	...	...	...
2875	ONION	WET	HUMID	30-40	RAINY	0.100
2876	ONION	WET	HUMID	40-50	NORMAL	4.625
2877	ONION	WET	HUMID	40-50	SUNNY	6.125
2878	ONION	WET	HUMID	40-50	WINDY	5.625
2879	ONION	WET	HUMID	40-50	RAINY	0.200

2880 rows × 6 columns

Şekil 1.2 Veri seti hakkında bilgiler.

1.2 Veri Setindeki Nesnelerin Kaç Özelliği Var ?

Veri setindeki nesnelerin de özellikleri bulunmaktadır. Bu özellikler aşağıda verilmiştir.

- Crop Type (Mahsül tipi)

  Banana, soyabean, cabbage, potato, rice, melon, maize, citrus, bean, wheat, mustard, cotton, sugarcane, tomato ve onion olmak üzere toplam 15 adet özellik bulunmaktadır.

- Soil Type (Toprak tipi)

  Dry, humid ve wet olmak üzere toplam 3 adet özellik bulunmaktadır.

- Region (Alan)

  Desert, semi arid, semi humid ve humid olmak üzere toplam 4 adet özellik bulunmaktadır.

- Temperature (Sıcaklık)

  10-20, 20-30, 30-40 ve 40-50 olmak üzere toplam 4 adet özellik bulunmaktadır.

- Weather condition (Havu durumu)

  Normal, sunny, rainy ve windy olmak üzere toplam 4 adet özellik bulunmaktadır.

- Water requirement (Su ihtiyacı)

Su ihtiyaci kısmında 0.1 ile 606 arasında birçok sayısal değer bulunmaktadır. İlerleyen bölümlerde bu kısma veri önişleme uygulanacak ve özellik sayısı 6'ya düşürülecektir.

```
print(df['CROP TYPE'].unique())
print(df['SOIL TYPE'].unique())
print(df['REGION'].unique())
print(df['WEATHER CONDITION'].unique())
```

```
['BANANA' 'SOYABEAN' 'CABBAGE' 'POTATO' 'RICE' 'MELON' 'MAIZE' 'CITRUS'
 'BEAN' 'WHEAT' 'MUSTARD' 'COTTON' 'SUGARCANE' 'TOMATO' 'ONION']
['DRY' 'HUMID' 'WET']
['DESERT' 'SEMI ARID' 'SEMI HUMID' 'HUMID']
['NORMAL' 'SUNNY' 'WINDY' 'RAINY']
```

Şekil 1.3 Veri setindeki nesnelerin özellikleri.

2.VERİ ÖNİŞLEME UYGULAMALARI

Veri seti şuan için eğitilmeye hazır değildir çünkü veri setinde kategorik değerlerin sayısal değerlere ve sayısal değer olan su ihtiyacı ise kategorik değere döndürmek gerekmektedir. Ayrıca su ihtiyacı altında bulunan özellikler çok fazla olduğu için modelimizin başarı oranı çok düşük kalacaktır. Bunu önlemek için özellik sayısı 6'ya indirgenecektir. Veri setindeki kategorik değerler ise One-Hot encoder ile sayısal değerlere döndürülecektir.

2.1 Sıcaklık Değerlerini Sayısal Değerlere Dönüştürme

Sıcaklık değerleri veri setimizde sayısal olarak gözükmesine ragmen sıcaklık değerlerinin tipi kategorikdir. Bu yüzden bu değerleri modelin performasını da arttırmak için integer türüne dönüştürüp oralama değeri ile değiştirdim.

```
#VERİ SETİMİZDEKİ SICAKLIK DEGERLERİNİ ORTALAMA  DEGERLER İLE DEĞİŞTİRİP INT TÜRÜNE DÖNÜŞTÜREBİLİRİZ.
df['TEMPERATURE'] = df['TEMPERATURE'].replace(['10-20'],'15')
df['TEMPERATURE'] = df['TEMPERATURE'].replace(['20-30'],'25')
df['TEMPERATURE'] = df['TEMPERATURE'].replace(['30-40'],'35')
df['TEMPERATURE'] = df['TEMPERATURE'].replace(['40-50'],'45')
df
```

	CROP TYPE	SOIL TYPE	REGION	TEMPERATURE	WEATHER CONDITION	WATER REQUIREMENT
0	BANANA	DRY	DESERT	15	NORMAL	8.750
1	BANANA	DRY	DESERT	15	SUNNY	10.250
2	BANANA	DRY	DESERT	15	WINDY	9.650
3	BANANA	DRY	DESERT	15	RAINY	0.750
4	BANANA	DRY	DESERT	25	NORMAL	9.850
...	...	...	...	...	...	...
2875	ONION	WET	HUMID	35	RAINY	0.100
2876	ONION	WET	HUMID	45	NORMAL	4.625
2877	ONION	WET	HUMID	45	SUNNY	6.125
2878	ONION	WET	HUMID	45	WINDY	5.625
2879	ONION	WET	HUMID	45	RAINY	0.200

2880 rows × 6 columns

Şekil 2.1 Sıcaklık değerini sayısallaştırma.

2.2 One-Hot Encoder ile Kategorik Değerleri Sayısal Değerlere Döndürme

One-Hot Encoder Python'da Scikit-learn kütüphanesinin Preprocessing alt kütüphanesinin altında, veri ön hazırlaması aşamasında kullanılan bir encoderdir. One-Hot encoder bu kategorik verilerin binarizasyonunu gerçekleştirmemizi sağlar. Mevcut değere "1" verip mevcut olmayanlara "0" vermektedir. Aşağıdaki Şekil 2.2'de görüldüğü üzere crop type için kod şablonun çıktısı her ayrı özellik için ayrılmıştır. Örneğin banana için (1.0.0...) değeri atanmıştır.

```
crop_encoder = OneHotEncoder()
crop_reshaped = np.array(df_categorical['CROP TYPE']).reshape(-1, 1)
crop_values = crop_encoder.fit_transform(crop_reshaped)

print(df_categorical['CROP TYPE'])
print()
print(crop_values.toarray())
print()
print(crop_encoder.inverse_transform(crop_values))
```

```
0       BANANA
1       BANANA
2       BANANA
3       BANANA
4       BANANA
         ...
2875     ONION
2876     ONION
2877     ONION
2878     ONION
2879     ONION
Name: CROP TYPE, Length: 2880, dtype: object

[[1. 0. 0. ... 0. 0. 0.]
 [1. 0. 0. ... 0. 0. 0.]
 [1. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]

[['BANANA']
 ['BANANA']
 ['BANANA']
 ...
 ['ONION']
 ['ONION']
 ['ONION']]
```

Şekil 2.2 Crop type sayısallıştırma.

2.2.1 Soil Type, Region ve Weather Condition Değerlerini Sayısallaştırma

Soil Type, Region ve Weather Condition değerleri de kategoriktir. Bu değerler de sayısal değerlere ve array formatına dönüştürülüp veri setine eklenmiştir. Aşağıdaki şekillerde dönüştürme işlemi ve veri setinin son hali gösterilmiştir.

```
soil_encoder = OneHotEncoder()
soil_reshaped = np.array(df_categorical['SOIL TYPE']).reshape(-1, 1)
soil_values = soil_encoder.fit_transform(soil_reshaped)
region_encoder = OneHotEncoder()
region_reshaped = np.array(df_categorical['REGION']).reshape(-1, 1)
region_values = region_encoder.fit_transform(region_reshaped)
weather_encoder = OneHotEncoder()
weather_reshaped = np.array(df_categorical['WEATHER CONDITION']).reshape(-1, 1)
weather_values = weather_encoder.fit_transform(weather_reshaped)
```

```
crop_type = pd.DataFrame(crop_values.toarray(), columns=
                        ['BANANA', 'SOYABEAN', 'CABBAGE', 'POTATO' ,'RICE' ,'MELON' ,'MAIZE' ,'CITRUS',
 'BEAN', 'WHEAT', 'MUSTARD' ,'COTTON', 'SUGARCANE' ,'TOMATO', 'ONION'])
soil_type = pd.DataFrame(soil_values.toarray(), columns=['DRY', 'HUMID' ,'WET'])
region_type = pd.DataFrame(region_values.toarray(), columns=['DESERT', 'SEMI ARID', 'SEMI HUMID', 'HUMID'])
weather_type = pd.DataFrame(weather_values.toarray(), columns=['NORMAL', 'SUNNY', 'WINDY', 'RAINY'])
df_categorical_encoded = pd.concat([crop_type, soil_type, region_type, weather_type], axis=1)
print(df_categorical_encoded.shape)
df_categorical_encoded.head()
```

Şekil 2.3 Soil Type, Region ve Weather Condition Değerlerini Sayısallaştırma

Aşağıdaki Şekil 2.4'de veri setinin son hali verilmştir. Veri setimizin tamamına sayısal hale getirdik.(Kodlar kısmında daha detaylı gözükmektedir.)

ANANA	SOYABEAN	CABBAGE	POTATO	RICE	MELON	MAIZE	CITRUS	...	HUMID	WET	DESERT	SEMI ARID	SEMI HUMID	HUMID	NORMAL	SUNNY	WINDY	RAINY
1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0

Şekil 2.4 Veri setinin son hali.

2.3 Water Requirement (Su ihtiyacı) Sınıf Sayısının Düzenlenmesi

Su ihtiyaci gereksinimi bulmak istiyoruz ama bu değerler sayısal ve Şekil 2.5'de sadece bir kısmı görüldüğü üzere çok fazla sınıf mevcuttur.

```
print(df['WATER REQUIREMENT'].unique())
```

```
[8.750e+00 1.025e+01 9.650e+00 7.500e-01 9.850e+00 1.135e+01 1.075e+01
 8.500e-01 1.225e+01 1.165e+01 2.750e+00 1.175e+01 1.325e+01 1.265e+01
 3.750e+00 7.980e+00 9.400e+00 8.800e+00 2.500e-01 8.980e+00 1.040e+01
 9.700e+00 9.800e-01 9.980e+00 1.140e+01 1.080e+01 1.980e+00 1.090e+01
 1.240e+01 1.180e+01 2.700e+00 6.950e+00 8.450e+00 7.800e+00 1.000e-01
 7.950e+00 9.450e+00 5.000e-01 8.950e+00 1.045e+01 9.900e+00 9.500e-01
 1.900e+00 5.650e+00 7.150e+00 6.600e+00 8.100e+00 7.600e+00 3.000e-01
 7.680e+00 8.180e+00 8.600e+00 8.000e-01 8.650e+00 1.015e+01 6.500e-01
 6.695e+00 8.200e+00 7.700e+00 6.000e-01 7.695e+00 9.200e+00 8.700e+00
 1.020e+01 9.600e+00 7.000e-01 9.500e+00 1.100e+01 1.050e+01 1.500e+00
 5.920e+00 7.420e+00 6.900e+00 6.920e+00 8.420e+00 7.900e+00 7.920e+00
 9.420e+00 8.900e+00 8.920e+00 1.042e+01 9.200e-01 4.890e+00 6.390e+00
 5.800e+00 5.600e+00 7.100e+00 2.000e-01 6.500e+00 7.500e+00 9.000e+00
 8.500e+00 3.650e+00 5.150e+00 4.650e+00 6.150e+00 5.000e+00 6.000e+00
 6.650e+00 6.200e+00 7.650e+00 8.150e+00 9.150e+00 1.500e-01 4.370e+00
 5.870e+00 5.300e+00 6.800e+00 6.300e+00 7.300e+00 7.200e+00 3.345e+00
 4.850e+00 4.400e+00 5.900e+00 5.400e+00 5.540e+00 7.040e+00 6.520e+00
 8.020e+00 2.060e+00 3.560e+00 3.000e+00 4.500e+00 4.000e+00 4.100e+00
 5.100e+00 6.970e+00 8.470e+00 1.030e+01 9.800e+00 9.950e+00 1.145e+01
 1.095e+01 1.950e+00 9.480e+00 1.010e+01 1.130e+01 1.800e+00 1.245e+01
 2.800e+00 5.530e+00 7.030e+00 8.000e+00 8.550e+00 1.005e+01 5.500e-01
 4.510e+00 6.010e+00 5.510e+00 5.500e+00 7.000e+00 9.300e+00 5.330e+00
 6.830e+00 6.330e+00 7.400e+00 8.400e+00 8.340e+00 7.350e+00 3.400e-01
 4.710e+00 6.210e+00 5.710e+00 3.900e+00 4.900e+00 6.400e+00 6.700e+00
 4.000e-01 4.300e+00 3.800e+00 3.700e+00 5.200e+00 4.700e+00 6.100e+00
 8.300e+00 3.500e+00 4.600e+00 5.700e+00 2.600e+00 3.600e+00 1.640e+00
 3.140e+00 5.050e+00 6.550e+00 6.050e+00 4.200e+00 3.100e+00 2.300e+00
 3.300e+00 3.330e+00 4.800e+00 4.330e+00 2.830e+00 3.830e+00 4.830e+00
 5.830e+00 7.330e+00 2.160e+00 3.660e+00 3.160e+00 4.660e+00 4.160e+00
```

Şekil 2.5 Water Requirement (Su ihtiyacı) Sınıf Sayısı

Sınıf sayısını ayarlamak için su ihtiyacı değerlerine bakmamız ve uygun bir şekilde düzenlememiz gerekmektedir. Su ihtiyacının sayısal değerleri aşağıda verilmiştir.

	WATER REQUIREMENT
count	2880.000000
mean	6.463141
std	22.687385
min	0.100000
25%	2.700000
50%	5.860000
75%	8.000000
max	606.000000

Şekil 2.6 Su ihtiyacı sayısal değerleri.

Bu değerlere göre su ihtiyacının minmum değeri 0.1 maksimum değeri 606 ve ortalama değerinin ise 6.4633141 olduğu gözükmektedir. Bu değerler göz önünde bulundurularak su ihtiyacı sütünü kaldırılıp yerine katagori adında yeni bir sütün eklenmiş ve su ihtiyacı 1'den az, 1 ile 3 arasında, 3 ile 6 arasında, 6 ile 9 arasında, 9 ile 12 arasında ve 12'den çok olmak üzere 6 adet sınıf oluşturulmuştur.

```
su_ihtiyaci = df_new["WATER REQUIREMENT"].values
katagori = []
for num in su_ihtiyaci:
    if num < 1:
        katagori.append("1'den az")
    elif num > 1 and num < 3 :
        katagori.append("1 ile 3 arasi")
    elif num > 3 and num < 6 :
        katagori.append("3 ile 6 arasi")
    elif num > 6 and num < 9 :
        katagori.append("6 ile 9 arasi")
    elif num > 9 and num < 12 :
        katagori.append("9 ile 12 arasi")
    else:
        katagori.append("12'den cok")
katagori = pd.DataFrame(data=katagori, columns=["Katagori"])
data = pd.concat([df_new, katagori], axis=1)
data.drop(columns="WATER REQUIREMENT", axis=1, inplace=True)
```

Şekil 2.7 Sınıf oluşturma.

SOYABEAN	CABBAGE	POTATO	RICE	MELON	MAIZE	CITRUS	BEAN	...	WET	DESERT	SEMI ARID	SEMI HUMID	HUMID	NORMAL	SUNNY	WINDY	RAINY	Katagori
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	6 ile 9 arasi
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	9 ile 12 arasi
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	9 ile 12 arasi
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1'den az
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	9 ile 12 arasi
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	...	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1'den az
0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	...	1.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	3 ile 6 arasi
0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	...	1.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	6 ile 9 arasi
0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	...	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	3 ile 6 arasi
0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	...	1.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1'den az

Şekil 2.8 Veri setinin son hali.

Veri setinde bulunan kategorik değerler sayıllaştırıldı,su ihtiyacı düzenlendi ve 6 olan sütun sayısı 28'e çıkarıldı. Artık veri setimiz eğitim için hazır hale gelmiştir.

```
---   ------          --------------   -----
 0    TEMPERATURE     2880 non-null    int32
 1    BANANA          2880 non-null    float64
 2    SOYABEAN        2880 non-null    float64
 3    CABBAGE         2880 non-null    float64
 4    POTATO          2880 non-null    float64
 5    RICE            2880 non-null    float64
 6    MELON           2880 non-null    float64
 7    MAIZE           2880 non-null    float64
 8    CITRUS          2880 non-null    float64
 9    BEAN            2880 non-null    float64
 10   WHEAT           2880 non-null    float64
 11   MUSTARD         2880 non-null    float64
 12   COTTON          2880 non-null    float64
 13   SUGARCANE       2880 non-null    float64
 14   TOMATO          2880 non-null    float64
 15   ONION           2880 non-null    float64
 16   DRY             2880 non-null    float64
 17   HUMID           2880 non-null    float64
 18   WET             2880 non-null    float64
 19   DESERT          2880 non-null    float64
 20   SEMI ARID       2880 non-null    float64
 21   SEMI HUMID      2880 non-null    float64
 22   HUMID           2880 non-null    float64
 23   NORMAL          2880 non-null    float64
 24   SUNNY           2880 non-null    float64
 25   WINDY           2880 non-null    float64
 26   RAINY           2880 non-null    float64
 27   Katagori        2880 non-null    object
dtypes: float64(26), int32(1), object(1)
memory usage: 618.9+ KB
```

Şekil 2.9 Veri setinin eğitim için hazır hale getirilmesi.

3.EĞİTİM VE MODELLERİN PERFORMANSI

Veri setinin %20'si test için yüzde %80'i ise eğitim için ayrılmıştır. Aşağıdaki şekilde test ve eğitim verilerinin ayrılması gösterilmiştir.

```
from sklearn.model_selection import train_test_split

col_isimler = ['TEMPERATURE', 'BANANA', 'SOYABEAN', 'CABBAGE',
                    'POTATO','RICE','MELON','MAIZE','CITRUS','BEAN','WHEAT','MUSTARD','COTTON',
               'SUGARCANE','TOMATO','ONION','DRY','HUMID','WET','DESERT','SEMI ARID','SEMI HUMID','HUMID',
              'NORMAL','SUNNY','WINDY','RAINY' ]
tahmin_isimler = ['Katagori']

X = data[col_isimler].values

y = data[tahmin_isimler].values

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=2018)
```

Şekil 3.1 Veri setinin ayrılması.

Veri seti Naive Bayes, Yapay sinir ağları, Destek vektör makinaları ve karar ağaçları ile eğitilmiş ve test verisi üzerinde performansları karışıklılık matrisi ile gözlenmiştir.

3.1 Naive Bayes

Naive Bayes'in test veri üzerinde ki performansı %52 olarak gözlenmiştir. Aşağıdaki şekilde karışıklılık matrisi gözlenmektedir.

```
Naive Bayes için Karışıklık matrisi
[[ 41 110   1  10   0  24]
 [  4  40   0   8   0   0]
 [  0   0 113   1   5   7]
 [  2  16   0  22   0   7]
 [  0   1   6   1   0  11]
 [ 17  40   5   2   1  81]]

Sınıflandırma Raporu

                precision    recall  f1-score   support

 6 ile 9 arasi       0.64      0.22      0.33       186
9 ile 12 arasi       0.19      0.77      0.31        52
      1'den az       0.90      0.90      0.90       126
    12'den cok       0.50      0.47      0.48        47
 1 ile 3 arasi       0.00      0.00      0.00        19
 3 ile 6 arasi       0.62      0.55      0.59       146

      accuracy                           0.52       576
     macro avg       0.48      0.48      0.43       576
  weighted avg       0.62      0.52      0.52       576
```

Şekil 3.2 Naive Bayes karışıklılık matrisi.

3.2 Destek Vektör Makineleri

Destek Vektör Makineleri'nin test veri üzerinde ki performansı %83 olarak gözlenmiştir. Aşağıdaki şekilde karışıklılık matrisi gözlenmektedir.

```
Destek Vektör Makinaları için Karışıklık matrisi
[[164   5   1   3   0  13]
 [  4  39   0   9   0   0]
 [  1   0 118   0   1   6]
 [  8  10   0  23   1   5]
 [  0   0   6   0   1  12]
 [  5   0   5   0   0 136]]

Sınıflandırma Raporu

                precision    recall  f1-score   support

 6 ile 9 arasi       0.90      0.88      0.89       186
9 ile 12 arasi       0.72      0.75      0.74        52
      1'den az       0.91      0.94      0.92       126
    12'den cok       0.66      0.49      0.56        47
 1 ile 3 arasi       0.33      0.05      0.09        19
 3 ile 6 arasi       0.79      0.93      0.86       146

      accuracy                           0.84       576
     macro avg       0.72      0.67      0.68       576
  weighted avg       0.82      0.84      0.82       576
```

Şekil 3.3 Destek Vektör Makineleri karışıklılık matrisi.

3.3 Karar Ağaçları

Karar Ağaçları'nın test veri üzerinde ki performansı %77 olarak gözlenmiştir. Aşağıdaki şekilde karışıklılık matrisi gözlenmektedir.

```
Karar Ağaçları için Karışıklık matrisi
[[144  13   1   8   0  20]
 [  9  38   0   5   0   0]
 [  1   0 119   0   1   5]
 [  7  13   0  22   2   3]
 [  0   0   4   0  11   4]
 [ 20   0   4   2  11 109]]

Sınıflandırma Raporu

                precision    recall  f1-score   support

 6 ile 9 arasi       0.80      0.77      0.78       186
9 ile 12 arasi       0.59      0.73      0.66        52
      1'den az       0.93      0.94      0.94       126
    12'den cok       0.59      0.47      0.52        47
 1 ile 3 arasi       0.44      0.58      0.50        19
 3 ile 6 arasi       0.77      0.75      0.76       146

      accuracy                           0.77       576
     macro avg       0.69      0.71      0.69       576
  weighted avg       0.77      0.77      0.77       576
```

Şekil 3.4 Karar ağaçları karışıklılık matrisi.

3.4 Yapay Sinir Ağları

Yapay Sinir Ağları'nın test veri üzerinde ki performansı %89 olarak gözlenmiştir. Aşağıdaki şekilde karışıklılık matrisi gözlenmektedir.

```
Yapay Sinir Ağları için karışıklık matrisi
[[171   6   1   1   0   7]
 [  4  42   0   6   0   0]
 [  0   0 118   0   2   6]
 [  4   5   1  30   0   7]
 [  0   0   2   0  14   3]
 [  3   0   2   0   3 138]]

Sınıflandırma Raporu

                precision    recall  f1-score   support

 6 ile 9 arasi       0.94      0.92      0.93       186
9 ile 12 arasi       0.79      0.81      0.80        52
      1'den az       0.95      0.94      0.94       126
    12'den cok       0.81      0.64      0.71        47
 1 ile 3 arasi       0.74      0.74      0.74        19
 3 ile 6 arasi       0.86      0.95      0.90       146

      accuracy                           0.89       576
     macro avg       0.85      0.83      0.84       576
  weighted avg       0.89      0.89      0.89       576
```

Şekil 3.5 Yapay Sinir Ağları karışıklılık matrisi.

4. SONUÇLAR

Sonuç olarak veri seti üzerinde yapılan sınıflandırma çalışmasında en başarılı model yapay sinir ağları en başarısız model ise naive bayes olmuştur.

4.1 Karışıklılık Matrisi Hakkında Bilgi

Yapay Sinir Ağları'nın karışıklılık matrisini biraz daha açacak olursak:

```
Yapay Sinir Ağları için karışıklık matrisi
[[171   6   1   1   0   7]
 [  4  42   0   6   0   0]
 [  0   0 118   0   2   6]
 [  4   5   1  30   0   7]
 [  0   0   2   0  14   3]
 [  3   0   2   0   3 138]]

Sınıflandırma Raporu

                precision    recall  f1-score   support

 6 ile 9 arasi       0.94      0.92      0.93       186
9 ile 12 arasi       0.79      0.81      0.80        52
      1'den az       0.95      0.94      0.94       126
    12'den cok       0.81      0.64      0.71        47
 1 ile 3 arasi       0.74      0.74      0.74        19
 3 ile 6 arasi       0.86      0.95      0.90       146

      accuracy                           0.89       576
     macro avg       0.85      0.83      0.84       576
  weighted avg       0.89      0.89      0.89       576
```

2880 adet satırımız vardı ve yüzde %20(576) satırı test verisi olarak ayırmıştık. Bu 576 satırı da 6 sınıfa bölmüştük. Burada ilk sütün 6 ile 9 arasını, 2. Sütun 9 ile 12 arasını, 3. sütun 1'den az olan değerleri, 4. Sütun 12'den çok olan değerleri, 5. sütün 1 ile 3 arasını ve 6.sütun 3 ile 6 arasını temsil etmektedir. İlk satırı inceleyecek olursak 6 ile 9 arasında bulunan 186 adet değerin 171 adetini doğru tahmin ederek ilk satır için 176x100/186 işlemini yaptığımızda %94 başarı oranını yakalamış oluyoruz. Aynı işemler diğer satırlar için uygulanıp ortalama başarı oranı bulunuyor ve bu oran %89 değerine karşılık geliyor. Yapay sinir ağlarının en düşük başarı değeri ise %74 ile 1 ile 3 arasındaki değerlerinden gelmektedir. Bu satırda 19 adet değer bulunmakta ve bu değerlerin 14 tanesi doğru tahmin edilmiştir.

,

Aşağıdaki çizelgede modeller ve başarı oranları gösterilmiştir.

Modeller	Performans
Yapay Sinir Ağları	%89
Destek Vektör Makineleri	%83
Karar Ağaçları	%77
Naive Bayes	%52

Çizelge 4.1 Modellerin performansı.

5.KODLAR VE ÇIKTILARI

```
import pandas as pd
import pandas_datareader.data as pdr
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import accuracy_score
import numpy as np
import matplotlib as plt
import plotly.express as px
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn import metrics
plt.style.use('bmh')
import warnings
warnings.filterwarnings('ignore')
```

VERİ HAKKINDA BİLGİ EDİNELİM VE EKSİK VERİ OLUP OLMADIGINI KONTROL EDELİM

```
df = pd.read_csv('dataset.csv')
```

```
df
```

```
     CROP TYPE SOIL TYPE  REGION TEMPERATURE WEATHER CONDITION  \
0       BANANA       DRY  DESERT       10-20            NORMAL
1       BANANA       DRY  DESERT       10-20             SUNNY
2       BANANA       DRY  DESERT       10-20             WINDY
3       BANANA       DRY  DESERT       10-20             RAINY
4       BANANA       DRY  DESERT       20-30            NORMAL
...        ...       ...     ...         ...               ...
2875     ONION       WET   HUMID       30-40             RAINY
2876     ONION       WET   HUMID       40-50            NORMAL
2877     ONION       WET   HUMID       40-50             SUNNY
2878     ONION       WET   HUMID       40-50             WINDY
2879     ONION       WET   HUMID       40-50             RAINY

      WATER REQUIREMENT
0                 8.750
1                10.250
2                 9.650
3                 0.750
4                 9.850
...                 ...
2875              0.100
2876              4.625
2877              6.125
2878              5.625
2879              0.200
```

```
[2880 rows x 6 columns]
```

```
#Verimiz 2880 satır ve 6 sütündan oluşmaktadır.
df.columns
```

```
Index(['CROP TYPE', 'SOIL TYPE', 'REGION', 'TEMPERATURE', 'WEATHER CONDITION'
,
       'WATER REQUIREMENT'],
      dtype='object')
```

```
df.info() #Datasetimizde herhangi bir eksik veri yok.
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2880 entries, 0 to 2879
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   CROP TYPE          2880 non-null   object 
 1   SOIL TYPE          2880 non-null   object 
 2   REGION             2880 non-null   object 
 3   TEMPERATURE        2880 non-null   object 
 4   WEATHER CONDITION  2880 non-null   object 
 5   WATER REQUIREMENT  2880 non-null   float64
dtypes: float64(1), object(5)
memory usage: 135.1+ KB
```

```
df.describe()
```

```
       WATER REQUIREMENT
count        2880.000000
mean            6.463141
std            22.687385
min             0.100000
25%             2.700000
50%             5.860000
75%             8.000000
max           606.000000
```

```
df[df['WATER REQUIREMENT']==df['WATER REQUIREMENT'].max()]
```

```
     CROP TYPE SOIL TYPE     REGION TEMPERATURE WEATHER CONDITION  \
1114     MELON       WET  SEMI ARID       30-40             WINDY   
1306     MAIZE       WET  SEMI ARID       30-40             WINDY   
2074   MUSTARD       WET  SEMI ARID       30-40             WINDY   
2650    TOMATO       WET  SEMI ARID       30-40             WINDY   

      WATER REQUIREMENT
1114              606.0
1306              606.0
2074              606.0
2650              606.0
```

```
df[df['WATER REQUIREMENT']==df['WATER REQUIREMENT'].min()]
```

```
     CROP TYPE SOIL TYPE      REGION TEMPERATURE WEATHER CONDITION  \
35       BANANA       DRY  SEMI HUMID       10-20             RAINY
51       BANANA       DRY       HUMID       10-20             RAINY
83       BANANA     HUMID   SEMI ARID       10-20             RAINY
99       BANANA     HUMID  SEMI HUMID       10-20             RAINY
115      BANANA     HUMID       HUMID       10-20             RAINY
...         ...       ...         ...         ...               ...
2855      ONION       WET  SEMI HUMID       20-30            NORMAL
2859      ONION       WET  SEMI HUMID       30-40            NORMAL
2867      ONION       WET       HUMID       10-20            NORMAL
2871      ONION       WET       HUMID       20-30             RAINY
2875      ONION       WET       HUMID       30-40             RAINY

      WATER REQUIREMENT
35                  0.1
51                  0.1
83                  0.1
99                  0.1
115                 0.1
...                 ...
2855                0.1
2859                0.1
2867                0.1
2871                0.1
2875                0.1

[157 rows x 6 columns]
```

VERİ ÖNİŞLEME UYGULAMALARI

```
#VERİ SETİMİZDEKİ SICAKLIK DEGERLERİNİ ORTALAMA  DEGERLER İLE DEĞİŞTİRİP INT
TÜRÜNE DÖNÜŞTÜREBİLİRİZ.
df['TEMPERATURE'] = df['TEMPERATURE'].replace(['10-20'],'15')
df['TEMPERATURE'] = df['TEMPERATURE'].replace(['20-30'],'25')
df['TEMPERATURE'] = df['TEMPERATURE'].replace(['30-40'],'35')
df['TEMPERATURE'] = df['TEMPERATURE'].replace(['40-50'],'45')
df
```

```
     CROP TYPE SOIL TYPE  REGION TEMPERATURE WEATHER CONDITION  \
0        BANANA       DRY  DESERT          15            NORMAL
1        BANANA       DRY  DESERT          15             SUNNY
2        BANANA       DRY  DESERT          15             WINDY
3        BANANA       DRY  DESERT          15             RAINY
4        BANANA       DRY  DESERT          25            NORMAL
...         ...       ...     ...         ...               ...
2875      ONION       WET   HUMID          35             RAINY
2876      ONION       WET   HUMID          45            NORMAL
2877      ONION       WET   HUMID          45             SUNNY
2878      ONION       WET   HUMID          45             WINDY
```

```
2879     ONION       WET   HUMID          45             RAINY

      WATER REQUIREMENT
0                 8.750
1                10.250
2                 9.650
3                 0.750
4                 9.850
...                 ...
2875              0.100
2876              4.625
2877              6.125
2878              5.625
2879              0.200

[2880 rows x 6 columns]
```

```
df["TEMPERATURE"] = df.TEMPERATURE.astype(int)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2880 entries, 0 to 2879
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   CROP TYPE          2880 non-null   object
 1   SOIL TYPE          2880 non-null   object
 2   REGION             2880 non-null   object
 3   TEMPERATURE        2880 non-null   int32
 4   WEATHER CONDITION  2880 non-null   object
 5   WATER REQUIREMENT  2880 non-null   float64
dtypes: float64(1), int32(1), object(4)
memory usage: 123.9+ KB
```

```
df_numeric = df[['TEMPERATURE', 'WATER REQUIREMENT']]
df_categorical = df[['CROP TYPE', 'SOIL TYPE', 'REGION', 'WEATHER CONDITION']
]
```

```
df_numeric.head()
```

```
   TEMPERATURE  WATER REQUIREMENT
0           15               8.75
1           15              10.25
2           15               9.65
3           15               0.75
4           25               9.85
```

```
df_categorical.head()
```

```
  CROP TYPE SOIL TYPE  REGION WEATHER CONDITION
0    BANANA       DRY  DESERT            NORMAL
```

```
1    BANANA       DRY  DESERT             SUNNY
2    BANANA       DRY  DESERT             WINDY
3    BANANA       DRY  DESERT             RAINY
4    BANANA       DRY  DESERT            NORMAL
```

```
print(df['CROP TYPE'].unique())
print(df['SOIL TYPE'].unique())
print(df['REGION'].unique())
print(df['WEATHER CONDITION'].unique())
```

```
['BANANA' 'SOYABEAN' 'CABBAGE' 'POTATO' 'RICE' 'MELON' 'MAIZE' 'CITRUS'
 'BEAN' 'WHEAT' 'MUSTARD' 'COTTON' 'SUGARCANE' 'TOMATO' 'ONION']
['DRY' 'HUMID' 'WET']
['DESERT' 'SEMI ARID' 'SEMI HUMID' 'HUMID']
['NORMAL' 'SUNNY' 'WINDY' 'RAINY']
```

```
from sklearn.preprocessing import OneHotEncoder
import numpy as np
crop_encoder = OneHotEncoder()
crop_reshaped = np.array(df_categorical['CROP TYPE']).reshape(-1, 1)
crop_values = crop_encoder.fit_transform(crop_reshaped)
```

```
print(df_categorical['CROP TYPE'])
print()
print(crop_values.toarray())
print()
print(crop_encoder.inverse_transform(crop_values))
```

```
0       BANANA
1       BANANA
2       BANANA
3       BANANA
4       BANANA
         ...
2875     ONION
2876     ONION
2877     ONION
2878     ONION
2879     ONION
Name: CROP TYPE, Length: 2880, dtype: object

[[1. 0. 0. ... 0. 0. 0.]
 [1. 0. 0. ... 0. 0. 0.]
 [1. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]

[['BANANA']
 ['BANANA']
```

```
 ['BANANA']
 ...
 ['ONION']
 ['ONION']
 ['ONION']]

soil_encoder = OneHotEncoder()
soil_reshaped = np.array(df_categorical['SOIL TYPE']).reshape(-1, 1)
soil_values = soil_encoder.fit_transform(soil_reshaped)
region_encoder = OneHotEncoder()
region_reshaped = np.array(df_categorical['REGION']).reshape(-1, 1)
region_values = region_encoder.fit_transform(region_reshaped)
weather_encoder = OneHotEncoder()
weather_reshaped = np.array(df_categorical['WEATHER CONDITION']).reshape(-1,
1)
weather_values = weather_encoder.fit_transform(weather_reshaped)

crop_type = pd.DataFrame(crop_values.toarray(), columns=
                         ['BANANA', 'SOYABEAN', 'CABBAGE', 'POTATO' ,'RICE' ,'
MELON' ,'MAIZE' ,'CITRUS',
 'BEAN', 'WHEAT', 'MUSTARD' ,'COTTON', 'SUGARCANE' ,'TOMATO', 'ONION'])
soil_type = pd.DataFrame(soil_values.toarray(), columns=['DRY', 'HUMID' ,'WET
'])
region_type = pd.DataFrame(region_values.toarray(), columns=['DESERT', 'SEMI
ARID', 'SEMI HUMID', 'HUMID'])
weather_type = pd.DataFrame(weather_values.toarray(), columns=['NORMAL', 'SUN
NY', 'WINDY', 'RAINY'])
df_categorical_encoded = pd.concat([crop_type, soil_type, region_type, weathe
r_type], axis=1)
print(df_categorical_encoded.shape)
df_categorical_encoded.head()

(2880, 26)

   BANANA  SOYABEAN  CABBAGE  POTATO  RICE  MELON  MAIZE  CITRUS  BEAN  WHEAT
\
0     1.0       0.0      0.0     0.0   0.0    0.0    0.0     0.0   0.0    0.0
1     1.0       0.0      0.0     0.0   0.0    0.0    0.0     0.0   0.0    0.0
2     1.0       0.0      0.0     0.0   0.0    0.0    0.0     0.0   0.0    0.0
3     1.0       0.0      0.0     0.0   0.0    0.0    0.0     0.0   0.0    0.0
4     1.0       0.0      0.0     0.0   0.0    0.0    0.0     0.0   0.0    0.0

   ...  HUMID  WET  DESERT  SEMI ARID  SEMI HUMID  HUMID  NORMAL  SUNNY  \
0  ...    0.0  0.0     1.0        0.0         0.0    0.0     1.0    0.0
1  ...    0.0  0.0     1.0        0.0         0.0    0.0     0.0    0.0
2  ...    0.0  0.0     1.0        0.0         0.0    0.0     0.0    0.0
3  ...    0.0  0.0     1.0        0.0         0.0    0.0     0.0    1.0
4  ...    0.0  0.0     1.0        0.0         0.0    0.0     1.0    0.0

   WINDY  RAINY
0    0.0    0.0
```

```
1    1.0    0.0
2    0.0    1.0
3    0.0    0.0
4    0.0    0.0

[5 rows x 26 columns]
```

```
df_new = pd.concat([df_numeric, df_categorical_encoded], axis=1)
```

```
df_new
```

```
      TEMPERATURE  WATER REQUIREMENT  BANANA  SOYABEAN  CABBAGE  POTATO  RICE
\
0              15              8.750     1.0       0.0      0.0     0.0   0.0
1              15             10.250     1.0       0.0      0.0     0.0   0.0
2              15              9.650     1.0       0.0      0.0     0.0   0.0
3              15              0.750     1.0       0.0      0.0     0.0   0.0
4              25              9.850     1.0       0.0      0.0     0.0   0.0
...           ...                ...     ...       ...      ...     ...   ...
2875           35              0.100     0.0       0.0      0.0     0.0   0.0
2876           45              4.625     0.0       0.0      0.0     0.0   0.0
2877           45              6.125     0.0       0.0      0.0     0.0   0.0
2878           45              5.625     0.0       0.0      0.0     0.0   0.0
2879           45              0.200     0.0       0.0      0.0     0.0   0.0

      MELON  MAIZE  CITRUS  ...  HUMID  WET  DESERT  SEMI ARID  SEMI HUMID  \
0       0.0    0.0     0.0  ...    0.0  0.0     1.0        0.0         0.0
1       0.0    0.0     0.0  ...    0.0  0.0     1.0        0.0         0.0
2       0.0    0.0     0.0  ...    0.0  0.0     1.0        0.0         0.0
3       0.0    0.0     0.0  ...    0.0  0.0     1.0        0.0         0.0
4       0.0    0.0     0.0  ...    0.0  0.0     1.0        0.0         0.0
...     ...    ...     ...  ...    ...  ...     ...        ...         ...
2875    0.0    0.0     0.0  ...    0.0  1.0     0.0        1.0         0.0
2876    0.0    0.0     0.0  ...    0.0  1.0     0.0        1.0         0.0
2877    0.0    0.0     0.0  ...    0.0  1.0     0.0        1.0         0.0
2878    0.0    0.0     0.0  ...    0.0  1.0     0.0        1.0         0.0
2879    0.0    0.0     0.0  ...    0.0  1.0     0.0        1.0         0.0

      HUMID  NORMAL  SUNNY  WINDY  RAINY
0       0.0     1.0    0.0    0.0    0.0
1       0.0     0.0    0.0    1.0    0.0
2       0.0     0.0    0.0    0.0    1.0
3       0.0     0.0    1.0    0.0    0.0
4       0.0     1.0    0.0    0.0    0.0
...     ...     ...    ...    ...    ...
2875    0.0     0.0    1.0    0.0    0.0
2876    0.0     1.0    0.0    0.0    0.0
2877    0.0     0.0    0.0    1.0    0.0
2878    0.0     0.0    0.0    0.0    1.0
2879    0.0     0.0    1.0    0.0    0.0
```

```
[2880 rows x 28 columns]
```

```
print(df['WATER REQUIREMENT'].unique())
```

```
[8.750e+00 1.025e+01 9.650e+00 7.500e-01 9.850e+00 1.135e+01 1.075e+01
 8.500e-01 1.225e+01 1.165e+01 2.750e+00 1.175e+01 1.325e+01 1.265e+01
 3.750e+00 7.980e+00 9.400e+00 8.800e+00 2.500e-01 8.980e+00 1.040e+01
 9.700e+00 9.800e-01 9.980e+00 1.140e+01 1.080e+01 1.980e+00 1.090e+01
 1.240e+01 1.180e+01 2.700e+00 6.950e+00 8.450e+00 7.800e+00 1.000e-01
 7.950e+00 9.450e+00 5.000e-01 8.950e+00 1.045e+01 9.900e+00 9.500e-01
 1.900e+00 5.650e+00 7.150e+00 6.600e+00 8.100e+00 7.600e+00 3.000e-01
 7.680e+00 8.180e+00 8.600e+00 8.000e-01 8.650e+00 1.015e+01 6.500e-01
 6.695e+00 8.200e+00 7.700e+00 6.000e-01 7.695e+00 9.200e+00 8.700e+00
 1.020e+01 9.600e+00 7.000e-01 9.500e+00 1.100e+01 1.050e+01 1.500e+00
 5.920e+00 7.420e+00 6.900e+00 6.920e+00 8.420e+00 7.900e+00 7.920e+00
 9.420e+00 8.900e+00 8.920e+00 1.042e+01 9.200e-01 4.890e+00 6.390e+00
 5.800e+00 5.600e+00 7.100e+00 2.000e-01 6.500e+00 7.500e+00 9.000e+00
 8.500e+00 3.650e+00 5.150e+00 4.650e+00 6.150e+00 5.000e+00 6.000e+00
 6.650e+00 6.200e+00 7.650e+00 8.150e+00 9.150e+00 1.500e-01 4.370e+00
 5.870e+00 5.300e+00 6.800e+00 6.300e+00 7.300e+00 7.200e+00 3.345e+00
 4.850e+00 4.400e+00 5.900e+00 5.400e+00 5.540e+00 7.040e+00 6.520e+00
 8.020e+00 2.060e+00 3.560e+00 3.000e+00 4.500e+00 4.000e+00 4.100e+00
 5.100e+00 6.970e+00 8.470e+00 1.030e+01 9.800e+00 9.950e+00 1.145e+01
 1.095e+01 1.950e+00 9.480e+00 1.010e+01 1.130e+01 1.800e+00 1.245e+01
 2.800e+00 5.530e+00 7.030e+00 8.000e+00 8.550e+00 1.005e+01 5.500e-01
 4.510e+00 6.010e+00 5.510e+00 5.500e+00 7.000e+00 9.300e+00 5.330e+00
 6.830e+00 6.330e+00 7.400e+00 8.400e+00 8.340e+00 7.350e+00 3.400e-01
 4.710e+00 6.210e+00 5.710e+00 3.900e+00 4.900e+00 6.400e+00 6.700e+00
 4.000e-01 4.300e+00 3.800e+00 3.700e+00 5.200e+00 4.700e+00 6.100e+00
 8.300e+00 3.500e+00 4.600e+00 5.700e+00 2.600e+00 3.600e+00 1.640e+00
 3.140e+00 5.050e+00 6.550e+00 6.050e+00 4.200e+00 3.100e+00 2.300e+00
 3.300e+00 3.330e+00 4.800e+00 4.330e+00 2.830e+00 3.830e+00 4.830e+00
 5.830e+00 7.330e+00 2.160e+00 3.660e+00 3.160e+00 4.660e+00 4.160e+00
 5.660e+00 5.160e+00 6.660e+00 6.160e+00 1.330e+00 2.330e+00 1.000e+01
 1.070e+01 1.220e+01 1.170e+01 1.160e+01 1.310e+01 1.260e+01 2.900e+00
 6.750e+00 8.250e+00 7.750e+00 9.250e+00 9.750e+00 1.125e+01 1.750e+00
 5.450e+00 6.450e+00 9.100e+00 5.620e+00 7.120e+00 6.620e+00 8.120e+00
 7.620e+00 2.100e+00 5.370e+00 4.845e+00 4.345e+00 6.540e+00 7.520e+00
 3.060e+00 1.700e+01 1.850e+01 1.800e+01 1.950e+01 1.900e+01 2.050e+01
 2.000e+01 2.150e+01 2.100e+01 1.200e+01 1.550e+01 1.650e+01 1.750e+01
 1.350e+01 1.500e+01 1.450e+01 1.600e+01 1.250e+01 1.300e+01 1.400e+01
 1.150e+01 2.500e+00 1.000e+00 2.000e+00 6.120e+00 8.620e+00 9.620e+00
 9.120e+00 1.200e-01 1.062e+01 1.012e+01 1.120e+00 5.580e+00 7.080e+00
 6.580e+00 8.080e+00 7.580e+00 9.080e+00 8.580e+00 1.008e+01 9.580e+00
 5.800e-01 4.860e+00 6.360e+00 5.860e+00 7.360e+00 6.860e+00 8.360e+00
 7.860e+00 9.360e+00 8.860e+00 3.960e+00 5.460e+00 4.960e+00 6.460e+00
 5.960e+00 7.460e+00 6.960e+00 8.460e+00 7.960e+00 4.680e+00 6.180e+00
 5.680e+00 7.180e+00 6.680e+00 9.180e+00 8.680e+00 4.140e+00 5.640e+00
 5.140e+00 6.640e+00 6.140e+00 7.640e+00 7.140e+00 8.640e+00 8.140e+00
```

```
 3.420e+00 4.920e+00 4.420e+00 5.420e+00 6.420e+00 2.520e+00 4.020e+00
 3.520e+00 5.020e+00 4.520e+00 6.020e+00 5.520e+00 7.020e+00 4.560e+00
 4.060e+00 5.560e+00 5.060e+00 6.560e+00 6.060e+02 6.060e+00 7.560e+00
 7.060e+00 2.340e+00 3.840e+00 3.340e+00 4.840e+00 4.340e+00 5.840e+00
 5.340e+00 6.840e+00 6.340e+00 1.440e+00 2.440e+00 3.440e+00 4.440e+00
 1.054e+01 1.154e+01 2.540e+00 1.340e+01 1.360e+01 1.460e+01 1.060e+01
 1.600e+00 1.190e+01 2.400e+00 1.285e+01 1.235e+01 7.820e+00 9.320e+00
 8.820e+00 8.850e+00 1.035e+01 1.120e+01 1.700e+00 8.060e+00 9.560e+00
 1.100e+00 2.250e+00 1.110e+01 1.210e+01 1.200e+00 5.880e+00 7.380e+00
 6.880e+00 6.350e+00 7.850e+00 7.320e+00 8.320e+00 1.400e+00 6.250e+00
 7.250e+00 8.350e+00 9.840e+00 9.340e+00 6.040e+00 7.050e+00 8.560e+00
 8.050e+00 2.480e+00 3.980e+00 3.480e+00 4.980e+00 4.480e+00 5.980e+00
 5.480e+00 6.980e+00 6.480e+00 4.250e+00 5.750e+00 5.250e+00 4.125e+00
 5.625e+00 5.125e+00 6.625e+00 6.125e+00 7.625e+00 7.125e+00 8.625e+00
 8.125e+00 3.375e+00 4.875e+00 4.375e+00 5.875e+00 5.375e+00 6.875e+00
 6.375e+00 7.875e+00 7.375e+00 4.750e+00 3.250e+00 2.875e+00 3.875e+00
 2.375e+00 5.575e+00 6.575e+00 2.125e+00 3.625e+00 3.125e+00 4.625e+00
 1.625e+00 2.625e+00]
```

```
su_ihtiyaci = df_new["WATER REQUIREMENT"].values
katagori = []
for num in su_ihtiyaci:
    if num < 1:
        katagori.append("1'den az")
    elif num > 1 and num < 3 :
        katagori.append("1 ile 3 arasi")
    elif num > 3 and num < 6 :
        katagori.append("3 ile 6 arasi")
    elif num > 6 and num < 9 :
        katagori.append("6 ile 9 arasi")
    elif num > 9 and num < 12 :
        katagori.append("9 ile 12 arasi")
    else:
        katagori.append("12'den cok")
katagori = pd.DataFrame(data=katagori, columns=["Katagori"])
data = pd.concat([df_new, katagori], axis=1)
data.drop(columns="WATER REQUIREMENT", axis=1, inplace=True)
```

```
data
```

```
      TEMPERATURE  BANANA  SOYABEAN  CABBAGE  POTATO  RICE  MELON  MAIZE  \
0              15     1.0       0.0      0.0     0.0   0.0    0.0    0.0
1              15     1.0       0.0      0.0     0.0   0.0    0.0    0.0
2              15     1.0       0.0      0.0     0.0   0.0    0.0    0.0
3              15     1.0       0.0      0.0     0.0   0.0    0.0    0.0
4              25     1.0       0.0      0.0     0.0   0.0    0.0    0.0
...           ...     ...       ...      ...     ...   ...    ...    ...
2875           35     0.0       0.0      0.0     0.0   0.0    0.0    0.0
2876           45     0.0       0.0      0.0     0.0   0.0    0.0    0.0
2877           45     0.0       0.0      0.0     0.0   0.0    0.0    0.0
2878           45     0.0       0.0      0.0     0.0   0.0    0.0    0.0
```

```
2879          45     0.0       0.0      0.0     0.0   0.0    0.0    0.0

      CITRUS  BEAN  ...  WET  DESERT  SEMI ARID  SEMI HUMID  HUMID  NORMAL  \
0        0.0   0.0  ...  0.0     1.0        0.0         0.0    0.0     1.0
1        0.0   0.0  ...  0.0     1.0        0.0         0.0    0.0     0.0
2        0.0   0.0  ...  0.0     1.0        0.0         0.0    0.0     0.0
3        0.0   0.0  ...  0.0     1.0        0.0         0.0    0.0     0.0
4        0.0   0.0  ...  0.0     1.0        0.0         0.0    0.0     1.0
...      ...   ...  ...  ...     ...        ...         ...    ...     ...
2875     0.0   1.0  ...  1.0     0.0        1.0         0.0    0.0     0.0
2876     0.0   1.0  ...  1.0     0.0        1.0         0.0    0.0     1.0
2877     0.0   1.0  ...  1.0     0.0        1.0         0.0    0.0     0.0
2878     0.0   1.0  ...  1.0     0.0        1.0         0.0    0.0     0.0
2879     0.0   1.0  ...  1.0     0.0        1.0         0.0    0.0     0.0

      SUNNY  WINDY  RAINY        Katagori
0       0.0    0.0    0.0   6 ile 9 arasi
1       0.0    1.0    0.0  9 ile 12 arasi
2       0.0    0.0    1.0  9 ile 12 arasi
3       1.0    0.0    0.0        1'den az
4       0.0    0.0    0.0  9 ile 12 arasi
...     ...    ...    ...             ...
2875    1.0    0.0    0.0        1'den az
2876    0.0    0.0    0.0   3 ile 6 arasi
2877    0.0    1.0    0.0   6 ile 9 arasi
2878    0.0    0.0    1.0   3 ile 6 arasi
2879    1.0    0.0    0.0        1'den az

[2880 rows x 28 columns]
```

```
print(data['Katagori'].unique())
```

```
['6 ile 9 arasi' '9 ile 12 arasi' "1'den az" "12'den cok" '1 ile 3 arasi'
 '3 ile 6 arasi']
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2880 entries, 0 to 2879
Data columns (total 28 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   TEMPERATURE  2880 non-null   int32
 1   BANANA       2880 non-null   float64
 2   SOYABEAN     2880 non-null   float64
 3   CABBAGE      2880 non-null   float64
 4   POTATO       2880 non-null   float64
 5   RICE         2880 non-null   float64
 6   MELON        2880 non-null   float64
 7   MAIZE        2880 non-null   float64
 8   CITRUS       2880 non-null   float64
```

```
 9   BEAN          2880 non-null   float64
 10  WHEAT         2880 non-null   float64
 11  MUSTARD       2880 non-null   float64
 12  COTTON        2880 non-null   float64
 13  SUGARCANE     2880 non-null   float64
 14  TOMATO        2880 non-null   float64
 15  ONION         2880 non-null   float64
 16  DRY           2880 non-null   float64
 17  HUMID         2880 non-null   float64
 18  WET           2880 non-null   float64
 19  DESERT        2880 non-null   float64
 20  SEMI ARID     2880 non-null   float64
 21  SEMI HUMID    2880 non-null   float64
 22  HUMID         2880 non-null   float64
 23  NORMAL        2880 non-null   float64
 24  SUNNY         2880 non-null   float64
 25  WINDY         2880 non-null   float64
 26  RAINY         2880 non-null   float64
 27  Katagori      2880 non-null   object
dtypes: float64(26), int32(1), object(1)
memory usage: 618.9+ KB

data.to_csv('sonhali4.csv')
```

Veri Setini Ayırma

```
from sklearn.model_selection import train_test_split

col_isimler = ['TEMPERATURE', 'BANANA', 'SOYABEAN', 'CABBAGE',
                    'POTATO','RICE','MELON','MAIZE','CITRUS','BEAN','WHEAT',
'MUSTARD','COTTON',
               'SUGARCANE','TOMATO','ONION','DRY','HUMID','WET','DESERT','SEMI
ARID','SEMI HUMID','HUMID',
              'NORMAL','SUNNY','WINDY','RAINY' ]
tahmin_isimler = ['Katagori']

X = data[col_isimler].values

y = data[tahmin_isimler].values

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=2018)
```

Model Oluşturma

Naive bayes

```
from sklearn.naive_bayes import GaussianNB


nb_model = GaussianNB()
```

```
nb_model.fit(X_train, y_train.ravel())
```

```
GaussianNB()
```

```
prediction_from_trained_data = nb_model.predict(X_train)
accuracy = metrics.accuracy_score(y_train, prediction_from_trained_data)

print ("Naive Bayes ile eğitim başarısı : {0:.4f}".format(accuracy))
```

```
Naive Bayes ile eğitim başarısı : 0.5638
```

```
nb_predict_test = nb_model.predict(X_test)
print(classification_report(y_test, nb_predict_test))
print("Naive Bayesin test verisi üzerinde'ki performansı %s" %
      accuracy_score(y_test, nb_predict_test))
```

```
                precision    recall  f1-score   support

 1 ile 3 arasi       0.00      0.00      0.00        19
      1'den az       0.90      0.90      0.90       126
    12'den cok       0.50      0.47      0.48        47
 3 ile 6 arasi       0.62      0.55      0.59       146
 6 ile 9 arasi       0.64      0.22      0.33       186
9 ile 12 arasi       0.19      0.77      0.31        52

      accuracy                           0.52       576
     macro avg       0.48      0.48      0.43       576
  weighted avg       0.62      0.52      0.52       576

Naive Bayesin test verisi üzerinde'ki performansı 0.515625
```

```
print ("Naive Bayes için Karışıklık matrisi ")


print ("{0}".format(metrics.confusion_matrix(y_test, nb_predict_test,

                                             labels=['6 ile 9 arasi',
 '9 ile 12 arasi',
 "1'den az",
 "12'den cok",
 '1 ile 3 arasi',
 '3 ile 6 arasi'])))

print ("")

print ("Sınıflandırma Raporu\n")


print ("{0}".format(metrics.classification_report(y_test, nb_predict_test, la
```

```
bels=['6 ile 9 arasi',
 '9 ile 12 arasi',
 "1'den az",
 "12'den cok",
 '1 ile 3 arasi',
 '3 ile 6 arasi'])))
```

```
Naive Bayes için Karışıklık matrisi
[[ 41 110   1  10   0  24]
 [  4  40   0   8   0   0]
 [  0   0 113   1   5   7]
 [  2  16   0  22   0   7]
 [  0   1   6   1   0  11]
 [ 17  40   5   2   1  81]]

Sınıflandırma Raporu

               precision    recall  f1-score   support

 6 ile 9 arasi       0.64      0.22      0.33       186
9 ile 12 arasi       0.19      0.77      0.31        52
      1'den az       0.90      0.90      0.90       126
    12'den cok       0.50      0.47      0.48        47
 1 ile 3 arasi       0.00      0.00      0.00        19
 3 ile 6 arasi       0.62      0.55      0.59       146

      accuracy                           0.52       576
     macro avg       0.48      0.48      0.43       576
  weighted avg       0.62      0.52      0.52       576
```

Destek Vektör Makineleri

```
from sklearn.svm import SVC
svm_model = SVC(kernel='linear', C=1, random_state=42)

svm_model.fit(X_train, y_train.ravel())

SVC(C=1, kernel='linear', random_state=42)

prediction_from_trained_data = svm_model.predict(X_train)
accuracy = metrics.accuracy_score(y_train, prediction_from_trained_data)

print ("Destek Vektör Makineleri ile eğitim başarısı : {0:.4f}".format(accura
cy))

Destek Vektör Makineleri ile eğitim başarısı : 0.8273

svm_predict_test = svm_model.predict(X_test)
print(classification_report(y_test, svm_predict_test))
```

```
print("Destek Vektör Makinalarının test verisi üzerinde performansı %s" %
      accuracy_score(y_test, svm_predict_test))
```

```
                precision    recall  f1-score   support

 1 ile 3 arasi       0.33      0.05      0.09        19
      1'den az       0.91      0.94      0.92       126
    12'den cok       0.66      0.49      0.56        47
 3 ile 6 arasi       0.79      0.93      0.86       146
 6 ile 9 arasi       0.90      0.88      0.89       186
9 ile 12 arasi       0.72      0.75      0.74        52

      accuracy                           0.84       576
     macro avg       0.72      0.67      0.68       576
  weighted avg       0.82      0.84      0.82       576

Destek Vektör Makinalarının test verisi üzerinde performansı 0.83506944444444
44
```

```
print ("Destek Vektör Makinaları için Karışıklık matrisi ")


print ("{0}".format(metrics.confusion_matrix(y_test, svm_predict_test,

                                             labels=['6 ile 9 arasi',
 '9 ile 12 arasi',
 "1'den az",
 "12'den cok",
 '1 ile 3 arasi',
 '3 ile 6 arasi'])))

print ("")

print ("Sınıflandırma Raporu\n")


print ("{0}".format(metrics.classification_report(y_test, svm_predict_test, l
abels=['6 ile 9 arasi',
 '9 ile 12 arasi',
 "1'den az",
 "12'den cok",
 '1 ile 3 arasi',
 '3 ile 6 arasi'])))
```

```
Destek Vektör Makinaları için Karışıklık matrisi
[[164   5   1   3   0  13]
 [  4  39   0   9   0   0]
 [  1   0 118   0   1   6]
 [  8  10   0  23   1   5]
 [  0   0   6   0   1  12]
```

```
 [  5   0   5   0   0 136]]

Sınıflandırma Raporu

               precision    recall  f1-score   support

 6 ile 9 arasi       0.90      0.88      0.89       186
9 ile 12 arasi       0.72      0.75      0.74        52
      1'den az       0.91      0.94      0.92       126
    12'den cok       0.66      0.49      0.56        47
 1 ile 3 arasi       0.33      0.05      0.09        19
 3 ile 6 arasi       0.79      0.93      0.86       146

      accuracy                           0.84       576
     macro avg       0.72      0.67      0.68       576
  weighted avg       0.82      0.84      0.82       576
```

Karar Ağaçları

```
from sklearn.tree import DecisionTreeClassifier
dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train, y_train.ravel())

DecisionTreeClassifier(random_state=42)

prediction_from_trained_data = dt_model.predict(X_train)
accuracy = metrics.accuracy_score(y_train, prediction_from_trained_data)

print ("Karar ağaçları ile eğitim başarısı : {0:.4f}".format(accuracy))

Karar ağaçları ile eğitim başarısı : 0.9987

dt_predict_test = dt_model.predict(X_test)

print(classification_report(y_test, dt_predict_test))
print("Karar Ağaçlarının test verisi üzerinde'ki performansı %s" %
      accuracy_score(y_test, dt_predict_test))

               precision    recall  f1-score   support

 1 ile 3 arasi       0.44      0.58      0.50        19
      1'den az       0.93      0.94      0.94       126
    12'den cok       0.59      0.47      0.52        47
 3 ile 6 arasi       0.77      0.75      0.76       146
 6 ile 9 arasi       0.80      0.77      0.78       186
9 ile 12 arasi       0.59      0.73      0.66        52

      accuracy                           0.77       576
     macro avg       0.69      0.71      0.69       576
  weighted avg       0.77      0.77      0.77       576
```

```
Karar Ağaçlarının test verisi üzerinde'ki performansı 0.7690972222222222

print ("Karar Ağaçları için Karışıklık matrisi ")


print ("{0}".format(metrics.confusion_matrix(y_test, dt_predict_test,

                                              labels=['6 ile 9 arasi',
 '9 ile 12 arasi',
 "1'den az",
 "12'den cok",
 '1 ile 3 arasi',
 '3 ile 6 arasi'])))

print ("")

print ("Sınıflandırma Raporu\n")


print ("{0}".format(metrics.classification_report(y_test, dt_predict_test, la
bels=['6 ile 9 arasi',
 '9 ile 12 arasi',
 "1'den az",
 "12'den cok",
 '1 ile 3 arasi',
 '3 ile 6 arasi'])))

Karar Ağaçları için Karışıklık matrisi
[[144  13   1   8   0  20]
 [  9  38   0   5   0   0]
 [  1   0 119   0   1   5]
 [  7  13   0  22   2   3]
 [  0   0   4   0  11   4]
 [ 20   0   4   2  11 109]]

Sınıflandırma Raporu

                precision    recall  f1-score   support

 6 ile 9 arasi       0.80      0.77      0.78       186
9 ile 12 arasi       0.59      0.73      0.66        52
      1'den az       0.93      0.94      0.94       126
    12'den cok       0.59      0.47      0.52        47
 1 ile 3 arasi       0.44      0.58      0.50        19
 3 ile 6 arasi       0.77      0.75      0.76       146

      accuracy                           0.77       576
     macro avg       0.69      0.71      0.69       576
```

```
  weighted avg       0.77      0.77      0.77       576
```

Yapay Sinir Ağları

```
ann_model = MLPClassifier(hidden_layer_sizes=(13,13,13),max_iter=500, random_
state=42)
ann_model.fit(X_train, y_train.ravel())
```

```
MLPClassifier(hidden_layer_sizes=(13, 13, 13), max_iter=500, random_state=42)
```

```
prediction_from_trained_data = ann_model.predict(X_train)
accuracy = metrics.accuracy_score(y_train, prediction_from_trained_data)

print ("Yapay Sinir Ağları ile eğitim başarısı : {0:.4f}".format(accuracy))
```

```
Yapay Sinir Ağları ile eğitim başarısı : 0.9210
```

```
ann_predict_test = ann_model.predict(X_test)

print(classification_report(y_test, ann_predict_test))
print("Yapay Sinir Ağlarının test verisi üzerinde'ki performansı %s" %
      accuracy_score(y_test, ann_predict_test))
```

```
                precision    recall  f1-score   support

 1 ile 3 arasi       0.74      0.74      0.74        19
      1'den az       0.95      0.94      0.94       126
    12'den cok       0.81      0.64      0.71        47
 3 ile 6 arasi       0.86      0.95      0.90       146
 6 ile 9 arasi       0.94      0.92      0.93       186
9 ile 12 arasi       0.79      0.81      0.80        52

      accuracy                           0.89       576
     macro avg       0.85      0.83      0.84       576
  weighted avg       0.89      0.89      0.89       576
```

```
Yapay Sinir Ağlarının test verisi üzerinde'ki performansı 0.890625
```

```
print ("Yapay Sinir Ağları için karışıklık matrisi")


print ("{0}".format(metrics.confusion_matrix(y_test, ann_predict_test,

                                             labels=['6 ile 9 arasi',
 '9 ile 12 arasi',
 "1'den az",
 "12'den cok",
 '1 ile 3 arasi',
 '3 ile 6 arasi'])))
```

```
print ("")

print ("Sınıflandırma Raporu\n")


print ("{0}".format(metrics.classification_report(y_test, ann_predict_test, l
abels=['6 ile 9 arasi',
 '9 ile 12 arasi',
 "1'den az",
 "12'den cok",
 '1 ile 3 arasi',
 '3 ile 6 arasi'])))
```

```
Yapay Sinir Ağları için karışıklık matrisi
[[171   6   1   1   0   7]
 [  4  42   0   6   0   0]
 [  0   0 118   0   2   6]
 [  4   5   1  30   0   7]
 [  0   0   2   0  14   3]
 [  3   0   2   0   3 138]]

Sınıflandırma Raporu

               precision    recall  f1-score   support

 6 ile 9 arasi       0.94      0.92      0.93       186
9 ile 12 arasi       0.79      0.81      0.80        52
      1'den az       0.95      0.94      0.94       126
    12'den cok       0.81      0.64      0.71        47
 1 ile 3 arasi       0.74      0.74      0.74        19
 3 ile 6 arasi       0.86      0.95      0.90       146

      accuracy                           0.89       576
     macro avg       0.85      0.83      0.84       576
  weighted avg       0.89      0.89      0.89       576
```