MAKİNE ÖĞRENMESİ

Ödev ve Proje Dosyası

Ayşenur YILDIZDAL

21410051027

2023-2024 Eğitim Öğretim Yılı

Öğrenci No:	21410051027
İsim soyisim:	Ayşenur YILDIZDAL
Ödev numarası:	101023K - 1
Tarih:	13.10.2023
Ödev sorusu :	Klasörde verilen m1 ve m2 txt dosyalarının içindeki metinlerin benzeşim oranını buldurunuz? Python dili (diğer dillerde kullanılabilir) kullanarak en az iki metin dosyasını karşılaştırınız ve benzeşim oranlarını bulunuz. Benzeşim oranını bulurken önce bu sözcükleri çıkarmadan bulunuz(benzeme oranı yüksek çıkacaktır), ardından bu sözcükleri çıkartarak benzeşim oranını bulunuz(benzeme oranı düşük çıkacaktır)

```
str1 = open('m1.txt', 'r').read()
str2 = open('m2.txt', 'r').read()

str1 = str1.replace(".","")
str1 = str1.replace(",","")
str1 = str1.replace("'","")

str2 = str2.replace(".","")
str2 = str2.replace(",","")
str2 = str2.replace("'","")


durma = ["ve","veya","ile","çünkü","birkaç","böyle","falan","herkes","hiçbiri",
        "gibi","hangi","kim","şu","şey","yada","zira","zaten","yine","neyse",
        "ama","ancak","asla","az","bazı","bazısı","belki","birçok","çok","çoğu",
        "daha","değil","diğer","elbette","hiç","ise","kendi","kime","niye","önce",
       "ötürü","rağmen","şunu","şunlar","tümü","veya","yoksa","zaten","zira"]
for i in durma:
   str1 = str1.replace(i,"")
   str2 = str2.replace(i, "")

l1 = list(str1.split(" "))
print(l1)
print(len(l1))

l2 = list(str2.split(" "))
print(l2)
print(len(l2))




s1 = set(l1)
print(s1)
print(len(s1))
s2 = set(l2)
print(s2)
print(len(s2))
```

```
st = set.union(s1,s2)
print(st)
print(len(st))

ts = len(s1)+len(s2)
print("İki metin için tekil sözcük sayısı = ", ts)
tss = len(st)
print("Birleşim sonrası tekil sözcük sayısı = ",tss)
fark = ts - tss
print("Fark = ", fark)
benzeme = (fark*100)/tss
print("Benzeme oranı = %",benzeme)
```

Öğrenci No:	21410051027
İsim soyisim:	Ayşenur YILDIZDAL
Ödev numarası:	101023K - 2
Tarih:	26.10.2023
Ödev sorusu :	Araştırma ödevi: Naive Bayes yöntemi hakkında araştırma yaparak, küçük bir özet anlatım yapınız. Kod ödevi girilen metnin Shannon Entropy değerini hesaplayarak bit şeklinde dönüştürünüz.

Naive Bayes, makine öğrenimi ve istatistiksel sınıflandırma problemleri için kullanılan bir algoritmadır. Temel olarak, Bayes teoremi esas alınarak geliştirilmiştir. Bu algoritma, özellikle metin sınıflandırma problemlerinde (spam filtreleme, duygu analizi, kategori tahmini vb.) yaygın olarak kullanılır.

Naive Bayes algoritması, "naive" (saf, basit) adını, her bir özelliğin sınıflandırmada birbirinden bağımsız olduğunu varsayan bir önyargıdan alır. Bu, gerçek dünyada genellikle doğru olmayan bir varsayımdır, ancak algoritmanın basit ve etkili olmasına katkıda bulunur.

Naive Bayes algoritması, verileri özellik vektörleri olarak temsil eder. Özellik vektörleri, sınıflandırmak istediğimiz nesnenin özelliklerini içerir. Örneğin, bir e-posta mesajını sınıflandırmak istiyorsak, özellik vektörümüz kelime frekansları olabilir.

```
import math

metin = "aysenur"

metin = metin.replace(" ","")

liste = []

for i in metin:
   liste.append(i)

print(liste)
k = set(liste)
print(k)

deste = {}

for i in k:
   adet = liste.count(i)
   oransal = adet/len(liste)
   deste.update({i:oransal})

print(deste)

shannon = 0

for i in deste:
```

```
    v = deste[i]
    shannon += v*math.log(v,2)

shannon *=-1;

print(shannon)

bit = math.ceil(shannon)
print("Bit sayısı: ",bit)

dk = list(k)

b = []
for i in range(int(math.pow(2,bit))):
    a = bin(i)[2:]
    b.append(a)

dk.sort()
print(dk)
print(b)

for i in range(len(b)):
    for j in range(bit-len(b[i])):
        b[i] = "0"+b[i]

print(b)

coded = ""
for i in liste:
    coded += b[dk.index(i)]+"-"

print(coded)
```

Öğrenci No:	21410051027
İsim soyisim:	Ayşenur YILDIZDAL
Ödev numarası:	101023K - 3
Tarih:	
Ödev sorusu :	Araştırma ödevi : k- En Yakın Komşuluk Algoritması (k- Nearest Neighbors Algorithm / k-NNA) yöntemi hakkında ön okuma yaparak anladıklarınızı ödev şablonuna yazınız. Kod ödevi : Bulduğunuz basit bir veri seti üzerinde Gaussian Naive Bayes üzerinde prediction yaptırınız. Ödevlerinizi google colab üzerinden rahatca yapabilirsiniz.

K-Nearest Neighbors (KNN), bir makine öğrenimi algoritmasıdır ve özellikle sınıflandırma ve regresyon problemleri için kullanılır. KNN, veri noktalarını uzayda bulunan komşularına dayanarak sınıflandırma veya tahmin yapar.

Algoritmanın temel prensibi, bir veri noktasının sınıfının, çevresindeki k-en yakın komşusunun sınıfına eşit olduğunu varsayar. "K" değeri, kaç komşunun dikkate alınacağını belirler. Örneğin, K=3 ise, bir veri noktasının sınıflandırılması için en yakın üç komşu kullanılır.

```
import numpy as np
from sklearn.navie_bayes import GaussianNB
# Özellikler
X= np.array([0,0,0,0],[1,1,0,1],
            [2,0,1,0],[0,0,2,0],
            [2,0,2,1],[1,1,1,1],
            [0,1,1,1],[0,0,1,0],
            [2,0,0,0],[1,1,2,0]);

Y= np.array([0,1,1,0,1,0,1,1,1,0]);

model= GaussianNB()
model.fit(X,Y)

print(model.predict(([0,0,1,2])))
```

Öğrenci No:	21410051027
İsim soyisim:	Ayşenur YILDIZDAL
Ödev numarası:	101023K - 4
Tarih:	
Ödev sorusu :	Bulduğunuz basit bir veri seti üzerinde KNN prediction yapınız.

```
import math
uyku_saati= [9,8,5,6,4,3,2,1,10,12]
calisma_saati=[5,4,3,2,16,12,11,10,8,6]
basari=["orta","başarılı","başarılı","orta","zayıf","orta","orta","baş
arılı","başarılı","başarılı"]

us=int(input("lütfen uyku saatinizi giriniz: "))
cs=int(input("lütfen çalışma saatinizi giriniz: "))
k=int(input("k değerini giriniz: "))

dictionary={}

for i in range(len(uyku_saati)):
  usdif = uyku_saati[i]-us
  csdif = calisma_saati[i]-cs
  length = math.sqrt(math.pow(usdif,2)+math.pow(csdif,2))

  dictionary.update({i:length})

print(dictionary)

distance = {k: v for k, v in sorted(dictionary.items(), key = lambda
item: item[1])}
print(distance)

for i in distance:
  print(uyku_saati[i]," ",calisma_saati[i]," ",basari[i],"
",distance[i])
distance_list = list(distance)


k_list=[]
for i in range(k):
  k_list.append(basari[distance_list[i]])
print(k_list)
```

Öğrenci No:	21410051027
İsim soyisim:	Ayşenur YILDIZDAL
Ödev numarası:	101023K - 5
Tarih:	
Ödev sorusu :	Yapay Sinir Ağıyla kendi veri setini eğit.

```
X= df.iloc[:,0:4]
Y = df.select_dtypes(include=[object])

from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test= train_test_split(X,Y,test_size=0.30)

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)

X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

from sklearn.neural_network import MLPClassifier
mlp=MLPClassifier(hidden_layer_sizes=(10,10,10),max_iter=1000)
mlp.fit(X_train,Y_train.values.ravel())

tahmin= mlp.predict(X_test)
print(tahmin)
```

Öğrenci No:	21410051027
İsim soyisim:	Ayşenur YILDIZDAL
Ödev numarası:	101023K - 6
Tarih:	
Ödev sorusu :	Karar ağaçları kullanarak kendi veri setinizi eğitiniz.

```
import pandas as pd

path = "/content/data.csv"
df= pd.read_csv(path)
df.head()
X = df.iloc[:,0:4]
Y= df.select_dtypes(include=[object])

Y.head()
from sklearn import tree
clf= tree.DecisionTreeClassifier()
clf=clf.fit(X,Y)

clf.predict([[120,6000,30,1500]])
tree.plot_tree(clf)
```

Öğrenci No:	21410051027
İsim soyisim:	Ayşenur YILDIZDAL
Ödev numarası:	101023K - 7
Tarih:	
Ödev sorusu :	K-Means ile kendi veri setini eğit.

```
from sklearn.cluster import KMeans
import numpy as np
import matplotlib.pyplot as plt

X = np.array([[3, 5], [4, 2], [2, 3],
        [8, 9], [5, 6], [15, 2]])
kmeans = KMeans(n_clusters=2, random_state=0).fit(X)
kmeans.labels_

kmeans.predict([[0, 0], [12, 3]])

#kmeans.cluster_centers_

#fig, ax = plt.subplots()
#ax.plot(X,'bo')

#plt.show()
```

Öğrenci No:	21410051027
İsim soyisim:	Ayşenur YILDIZDAL
Ödev numarası:	101023K - 8
Tarih:	
Ödev sorusu :	SVM ile kendi veri setini eğit

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm

X = np.c_[
   (38.0, 15.0), #1
   (35.0, 12.0), #1
   (37.0, 20.0), #1
   (29.0, 9.0),  #0
   (30.0, 10.0), #1
   (23.0, 5.0),  #0
   (21.0, 4.0),  #0
   (31.0, 8.0),  #0
   (25.0, 6.0),  #0
   (40.0, 17.0), #1
   (32.0, 15.0), #1
   (36.0, 12.0), #1
   (28.0, 9.0),  #0
   (25.0, 9.0),  #0
   (24.0, 10.0), #0
   (34.0, 13.0), #1
].T
Y = [1,1,1,0,1,0,0,0,0,1,1,1,0,0,0,1]


fignum = 1

for kernel in ("linear", "poly", "rbf"):
   clf = svm.SVC(kernel=kernel, gamma=2)
   clf.fit(X, Y)


   plt.figure(fignum, figsize=(4, 3))
   plt.clf()

   plt.scatter(
      clf.support_vectors_[:, 0],
      clf.support_vectors_[:, 1],
      s=80,
      facecolors="none",
      zorder=10,
      edgecolors="k",
   )
   plt.scatter(X[:, 0], X[:, 1], c=Y, zorder=10, cmap=plt.cm.Paired, edgecolors="k")
```

```
    plt.axis("tight")
    x_min = 0
    x_max = 50
    y_min = 0
    y_max = 25

    XX, YY = np.mgrid[x_min:x_max:200j, y_min:y_max:200j]
    Z = clf.decision_function(np.c_[XX.ravel(), YY.ravel()])


    Z = Z.reshape(XX.shape)
    plt.figure(fignum, figsize=(4, 3))
    plt.pcolormesh(XX, YY, Z > 0, cmap=plt.cm.Paired)
    plt.contour(
        XX,
        YY,
        Z,
        colors=["blue", "k", "blue"],
        linestyles=["--", "-", "--"],
        levels=[-1, 0, 1],
    )

    plt.xlim(x_min, x_max)
    plt.ylim(y_min, y_max)

    plt.xticks(())
    plt.yticks(())
    fignum = fignum + 1
plt.show()
```

T.C
NECMETTİN ERBAKAN ÜNİVERSİTESİ
UYGULAMALI BİLİMLER FAKÜLTESİ
YÖNETİM BİLİŞİM SİSTEMLERİ BÖLÜMÜ
NORMAL ÖĞRETİM
MAKİNE ÖĞRENMESİ
FİNAL ÖDEVİ

ARAŞTIRMA KONUSU

DESTEK VEKTÖR SINIFLANDIRMASI İLE TELEFON FİYAT TAHMİNİ

ÖDEV SAHİBİ

AYŞENUR YILDIZDAL
21410051027

Destek Vektör Sınıflandırması İle Telefon Fiyat Sınıflandırması

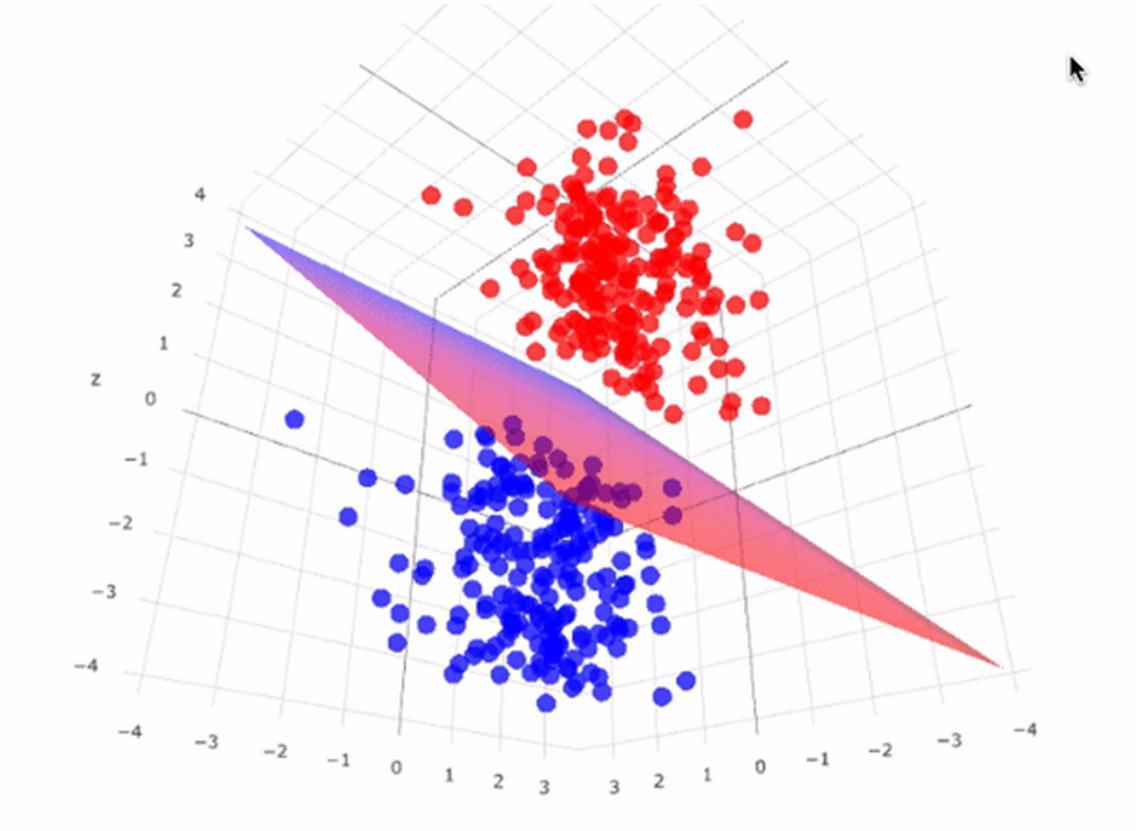
1. Support Vector Classification (Destek Vektör Sınıflandırma) Algoritması Nedir ?

"SVC" (Support Vector Classification), destek vektör makineleri (Support Vector Machines - SVM) ailesine ait bir sınıflandırma algoritmasıdır. SVM, özellikle sınıflandırma problemlerinde kullanılan güçlü ve esnek bir makine öğrenimi algoritmasıdır.

Support Vector Machine (SVM), genellikle "SVC" (Support Vector Classification) olarak adlandırılan, özellikle sınıflandırma problemlerinde kullanılan bir makine öğrenimi algoritmasıdır. SVM, özellikle doğrusal ve doğrusal olmayan sınıflandırma problemlerinde etkili bir şekilde çalışabilen bir algoritmadır.

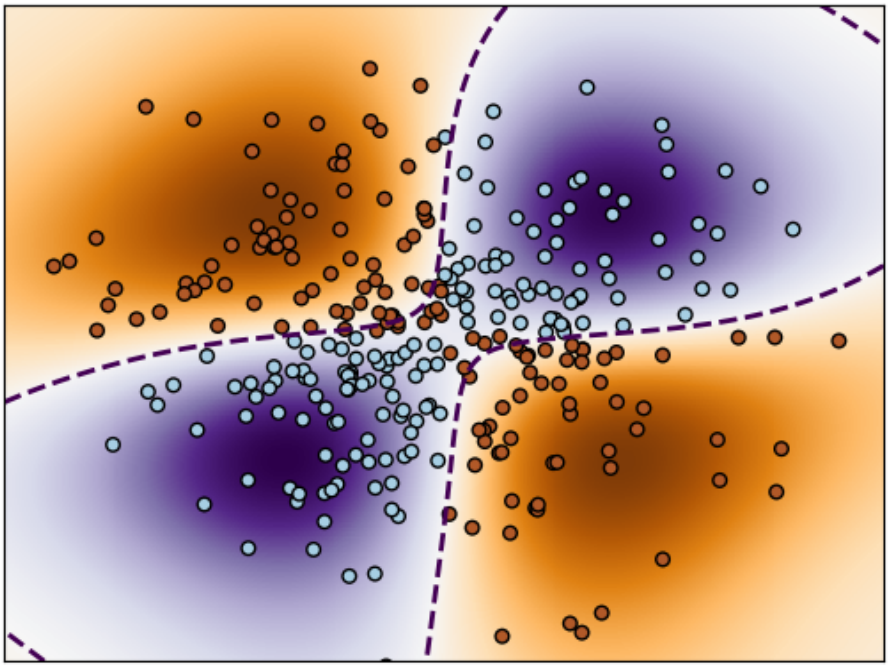
İşte SVM'nin temel prensipleri ve çalışma mantığı hakkında detaylı bir açıklama:

1. **Temel Prensip:**
    - SVM, sınıfları birbirinden ayıran optimal bir hiperdüzlemi (hyperplane) bulmaya odaklanır. İki sınıf arasındaki bu hiperdüzlem, veri noktalarını mümkün olduğunca iyi bölmek için seçilir.
2. **Sınıflandırma İçin Hiperdüzlem:**
    - İki sınıflı bir sınıflandırma problemi düşünelim. Hiperdüzlem, sınıfları birbirinden ayıran bir düzlemdir. Bu düzlem, iki sınıf arasındaki mesafeyi (marjin) maksimize etmeye çalışır. Bu marjin, sınıflar arasındaki güvenli bir bölgeyi temsil eder.



Sınıfları bölen hiperdüzlem

3. **Destek Vektörler:**
    - SVM, sınıfları bölen hiperdüzlemi belirlemek için destek vektörleri kullanır. Destek vektörler, hiperdüzleme olan uzaklıkları en küçük olan veri noktalarıdır. Bu noktalar, sınıflar arasındaki marjini maksimize etmek için kritik öneme sahiptir.
4. **Doğrusal ve Doğrusal Olmayan Sınıflandırma:**
    - SVM, doğrusal ve doğrusal olmayan sınıflandırma problemlerini çözebilir. Doğrusal SVM, veriyi doğrusal bir hiperdüzlemle bölmeye çalışırken, doğrusal olmayan SVM kernel fonksiyonları kullanarak veriyi daha yüksek boyutlu uzaylara taşıyarak sınıfları böler.
5. **Kernel Fonksiyonları:**
    - Kernel fonksiyonları, doğrusal olmayan sınıflandırmada kullanılır. Bu fonksiyonlar, veriyi orijinal özellik uzayından başka bir uzaya dönüştürür ve bu yeni uzayda daha kolay sınıflandırma yapılmasını sağlar.

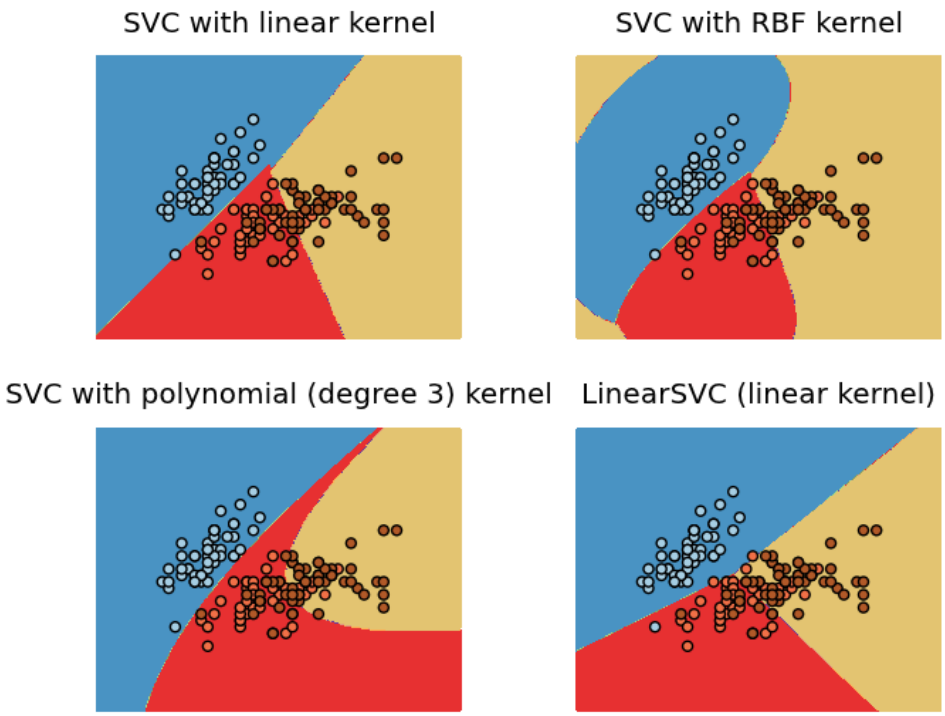


Doğrusal olmayan sınıflandırmada kernel fonksiyonlarının kullanılması

6. **Regülerleştirme Parametresi (C):**
    - SVM'nin bir regülerleştirme parametresi olan C, sınıflandırma hatası ve marjin arasındaki dengeyi kontrol eder. Küçük C değerleri, daha geniş bir marj sağlarken, büyük C değerleri sınıflandırma hatasını minimize etmeye odaklanır.
7. **Gamma (`gamma`):**
    - `gamma`, SVM'nin RBF (Radial Basis Function) çekirdeği kullanıldığında, özellik uzayındaki bir noktanın komşularının ne kadar etkili olacağını kontrol eden bir parametredir. Yüksek `gamma` değerleri, bir noktanın sınıflandırma üzerindeki etkisini daraltır ve modelin eğitim verisine aşırı uyum yapmasına neden olabilir.
    - Yüksek `gamma` değerleri, bir noktanın sadece yakın komşularını dikkate almasını sağlar ve bu da modelin daha karmaşık ve esnek olmasına neden olabilir. Düşük `gamma` değerleri ise bir noktanın daha geniş bir etki alanına sahip olmasına izin verir, bu da modelin daha düzenli ve genelleştirilebilir olmasına yardımcı olabilir.

8. **Degree (`degree`):**

- `degree`, SVM'nin polinom çekirdeği kullanıldığında, polinom derecesini belirleyen bir parametredir. Polinom çekirdeği, özellik uzayındaki veriler arasındaki ilişkiyi temsil etmek için bir polinom kullanır. Bu parametre, polinom çekirdeğinin kaçıncı dereceden olduğunu belirler.
- Yüksek `degree` değerleri, polinomun daha yüksek dereceli olduğu anlamına gelir. Daha yüksek dereceli polinomlar, daha karmaşık ve daha esnek sınıflandırma karar sınırları oluşturabilir. Ancak, çok yüksek dereceler, aşırı uyma (overfitting) riskini artırabilir. Genellikle, `degree` parametresi 1 ile 5 arasında bir değer alır.



İris veri seti üzerinde RBF , Linear ve Polynomial çekirdeklerin karşılaştırılması

SVM'nin avantajları arasında etkili sınıflandırma performansı, yüksek boyutlu veri setlerinde etkinlik ve marjin optimizasyonu yer almaktadır. Ancak, büyük veri setlerinde eğitim süresi uzun olabilir ve kernel seçimi önemli bir faktördür. SVM, özellikle orta ila

küçük boyutlu veri setleri ve orta ila yüksek boyutlu özellik uzayları için etkili bir tercih olabilir.

2. Telefon Fiyat Aralığı - Telefon Fiyat Aralığını Tahmin Etmek Neden Önemlidir ?

Telefon fiyat aralığını tahmin etmek, hem tüketiciler hem de endüstri oyuncuları için önemli bir konudur ve bir dizi avantaj ve etkiyi beraberinde getirir. Aşağıda telefon fiyatlarının tahmininin neden önemli olduğuna dair bazı temel noktalar listelenmiştir :

1. **Tüketiciler İçin Fiyat Değerlendirmesi:**
    - Tüketiciler, bir telefon alırken fiyatın, özelliklerle uygun bir denge oluşturup oluşturmadığını değerlendirirler. Doğru fiyat tahmini, tüketicilere bütçelerine uygun olan telefonları seçme konusunda yardımcı olabilir.
2. **Rekabet Stratejileri:**
    - Telefon üreticileri ve satıcıları, pazardaki rekabet avantajlarını sürdürmek ve hedef müşteri kitlesini çekmek için doğru fiyatlandırma stratejileri geliştirmelidir. Fiyat tahmini, doğru rekabet stratejilerinin oluşturulmasına yardımcı olabilir.
3. **Stok Yönetimi:**
    - Perakende satıcılar ve distribütörler, telefon stoklarını etkili bir şekilde yönetmek isterler. Doğru fiyat tahminleri, talebe daha iyi yanıt verebilmek için stok düzeylerini optimize etmelerine yardımcı olabilir.
4. **Pazar Analizi ve Segmentasyon:**
    - Fiyat aralığının doğru bir şekilde belirlenmesi, pazar analizi ve segmentasyon süreçlerine katkı sağlar. Farklı fiyat aralıklarındaki telefonlar, farklı tüketici segmentlerine hitap edebilir.
5. **Yatırım ve Karar Verme:**
    - Telefon üreticileri ve yatırımcılar, gelecekteki talep ve karlılık potansiyellerini değerlendirmek için fiyat tahminlerini kullanabilirler. Doğru fiyatlandırma stratejileri, yatırımların daha bilinçli bir şekilde yönetilmesine katkı sağlar.
6. **Tüketici Taleplerini Anlama:**

- Fiyat tahminleri, tüketicilerin belirli fiyat aralıklarındaki telefonlara olan taleplerini anlamak için kullanılabilir. Bu, yeni ürün geliştirmek veya mevcut ürünleri iyileştirmek için önemli bir bilgi kaynağıdır.

7. **Endüstri Gelişmelerini Öngörme:**
    - Fiyat tahminleri, telefon endüstrisindeki gelişmeleri öngörmeye yardımcı olabilir. Piyasadaki fiyat değişiklikleri, teknolojik yenilikler, rekabet ve tüketici talepleri gibi faktörlerle ilgili bilgiler, endüstri oyuncularına stratejik kararlar almalarında yardımcı olabilir.

Bu çalışma, telefon fiyat aralığını tahmin etmek amacıyla bir model oluşturmuştur. Veri seti, telefonların çeşitli özelliklerini içermektedir, bunlar arasında batarya gücü (battery_power), Bluetooth özelliği (blue), işlemci hızı (clock_speed), çift SIM kart desteği (dual_sim), ön kamera çözünürlüğü (fc), 4G bağlantısı (four_g), dahili bellek (int_memory), telefon ağırlığı (mobile_wt), çoklu işlemci çekirdek sayısı (n_cores), Primary Camera (pc), RAM kapasitesi (ram), konuşma süresi (talk_time), 3G bağlantısı (three_g), dokunmatik ekran özelliği (touch_screen), Wi-Fi bağlantısı (wifi), , ekran çözünürlüğü (px_heighxwidth) ve ekran boyutu (sc_hxw) yer almaktadır. Bu özellikler, endüstriyel alanda piyasaya sürülecek telefonların fiyatlarını etkileyen temel faktörleri temsil etmektedir. Geliştirilen model, bu özellikleri kullanarak fiyat aralığını tahmin etmeyi amaçlamaktadır. Bu tahminler, üreticilere pazar taleplerine uygun fiyatlandırma stratejileri oluşturma konusunda önemli bir yol gösterici sağlamaktadır.

3. Veri Setine istatiksel olarak bakış

Veri setindeki bazı özellikler :

1. **battery_power:** Telefonun batarya gücünü temsil eden sayısal bir değer. Bu değer, telefonun bataryasının kapasitesini belirtir.
2. **blue:** Bluetooth özelliğini ifade eden binominal (0 veya 1) bir değer. 1, Bluetooth'un etkin olduğunu, 0 ise etkin olmadığını gösterir.
3. **clock_speed:** Telefonun işlemci hızını temsil eden sayısal bir değer. Hız ne kadar yüksekse, işlemcinin daha hızlı çalıştığı anlamına gelir.

4. **dual_sim:** Çift SIM kart desteğini ifade eden binominal (0 veya 1) bir değer. 1, çift SIM kart desteğinin olduğunu, 0 ise olmadığını gösterir.
5. **fc:** Telefonun ön kamerasının çözünürlüğünü temsil eden sayısal bir değer. Megapiksel cinsinden ifade edilebilir.
6. **four_g:** 4G bağlantı özelliğini ifade eden binominal (0 veya 1) bir değer. 1, 4G bağlantısının etkin olduğunu, 0 ise etkin olmadığını gösterir.

Veri setine kısa bir bakış için pandas kütüphanesinden head() fonksiyonu kullanıldı :

```
data.head()
```

Out[2]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_h	sc_w	talk_time	thı
0	842	0	2.2	0	1	0	7	0.6	188	2	...	20	756	2549	9	7	19	
1	1021	1	0.5	1	0	1	53	0.7	136	3	...	905	1988	2631	17	3	7	
2	563	1	0.5	1	2	1	41	0.9	145	5	...	1263	1716	2603	11	2	9	
3	615	1	2.5	0	0	0	10	0.8	131	6	...	1216	1786	2769	16	8	11	
4	1821	1	1.2	0	13	1	44	0.6	141	2	...	1208	1212	1411	8	2	15	

5 rows × 21 columns

int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi	price_range
7	0.6	188	2	...	20	756	2549	9	7	19	0	0	1	1
53	0.7	136	3	...	905	1988	2631	17	3	7	1	1	0	2
41	0.9	145	5	...	1263	1716	2603	11	2	9	1	1	0	2
10	0.8	131	6	...	1216	1786	2769	16	8	11	1	0	0	2
44	0.6	141	2	...	1208	1212	1411	8	2	15	1	1	0	1

Veri seti hakkında betimsel analitik ve matematiksel birkaç bilgi vermesi için pandas kütüphanesi içinden “describe()” fonksiyonu kullanıldı. Burdaki amaç veri setini tanımak ve göz atmaktır.

- İşlemler Google Colab üzerinden yapılmıştır :

```
#Sayısal veriler hakkında istatiksel bilgiler
num_features = ['battery_power','clock_speed','int_memory','mobi

data[num_features].describe()
```

	battery_power	clock_speed	int_memory	mobile_wt	n_cores	ram
count	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000
mean	1238.518500	1.522250	32.046500	140.249000	4.520500	2124.213000
std	439.418206	0.816004	18.145715	35.399655	2.287837	1084.732044
min	501.000000	0.500000	2.000000	80.000000	1.000000	256.000000
25%	851.750000	0.700000	16.000000	109.000000	3.000000	1207.500000
50%	1226.000000	1.500000	32.000000	141.000000	4.000000	2146.500000
75%	1615.250000	2.200000	48.000000	170.000000	7.000000	3064.500000
max	1998.000000	3.000000	64.000000	200.000000	8.000000	3998.000000

1. **Count (Sayım):**
    - Her sütundaki non-null (boş olmayan) veri noktalarının sayısını ifade eder. Boş (null) değerler, sayıma dahil edilmez.
2. **Mean (Ortalama):**
    - Bir sütundaki sayısal değerlerin aritmetik ortalamasını ifade eder. Değerlerin toplamı, sayıya bölünerek hesaplanır.
3. **Std (Standart Sapma):**
    - Bir sütundaki değerlerin ne kadar yayıldığını ölçen bir istatistiksel ölçüdür. Daha yüksek bir standart sapma, değerlerin ortalamadan daha fazla sapma eğiliminde olduğunu gösterir.
4. **Min (Minimum):**
    - Bir sütundaki en küçük değeri ifade eder.
5. **25%, 50%, 75% (Çeyreklikler):**

- Veri sıralandığında, yüzde 25, yüzde 50 ve yüzde 75 noktalarındaki değerleri ifade eder. Yüzde 50'deki değer, median (ortanca) olarak da bilinir.

6. **Max (Maksimum):**
   - Bir sütundaki en büyük değeri ifade eder.

Veri setimiz temiz olduğu için herhangi bir aykırı değer bulunmuyor. Bundan emin olmak için özelliklerin bilgisi alındı. pandas kütüphanesindeki info() fonksiyonu ile belirli bilgiler edinilebilir.

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   battery_power  2000 non-null   int64  
 1   blue           2000 non-null   int64  
 2   clock_speed    2000 non-null   float64
 3   dual_sim       2000 non-null   int64  
 4   fc             2000 non-null   int64  
 5   four_g         2000 non-null   int64  
 6   int_memory     2000 non-null   int64  
 7   m_dep          2000 non-null   float64
 8   mobile_wt      2000 non-null   int64  
 9   n_cores        2000 non-null   int64  
 10  pc             2000 non-null   int64  
 11  px_height      2000 non-null   int64  
 12  px_width       2000 non-null   int64  
 13  ram            2000 non-null   int64  
 14  sc_h           2000 non-null   int64  
 15  sc_w           2000 non-null   int64  
 16  talk_time      2000 non-null   int64  
 17  three_g        2000 non-null   int64  
 18  touch_screen   2000 non-null   int64  
 19  wifi           2000 non-null   int64  
 20  price_range    2000 non-null   int64  
dtypes: float64(2), int64(19)
memory usage: 328.3 KB
```

Görüldüğü üzere 21 tane feature bulunuyor , bunları özellik mühendisliği ile özellik çıkarımı ve boyut azaltmaya gidilecektir.

```
null_sum=data.isnull().sum()
null_sum
```

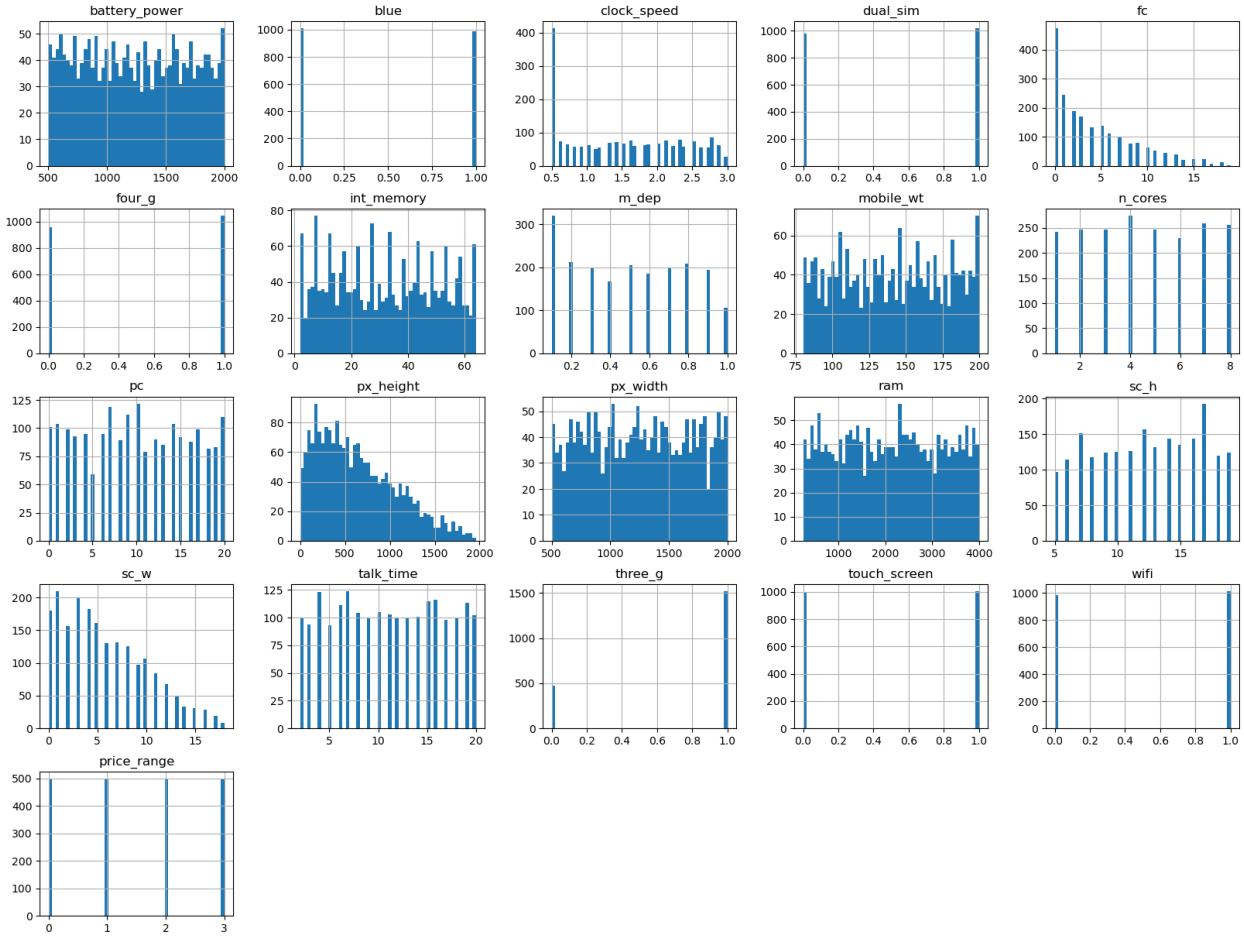
Görüldüğü üzere veri setimizde herhangi boş değer de bulunmuyor. Bu yüzden verileri temizlemeye ve doldurmaya ihtiyaç bulunmuyor fakat özellik azaltmaya gidilmesi gerekiyor.

4. Veri setini Hazırlama ve veri setinin analizi

Histogram:

> Histogram, bir veri setinin dağılımını gösteren bir grafik türüdür. Genellikle sayısal verilerin frekans dağılımını görselleştirmek için kullanılır. Histogramlar, belirli bir aralıktaki veri noktalarının yoğunluğunu ve frekansını gösteren çubuklu bir grafiktir.

Veri seti ile ilgili histogram grafikleri aşağıda verilmiştir:



Özellik azaltma yapılmamış veri setinin özelliklerinin histogram grafiği

Korelasyon:

Korelasyon, iki değişken arasındaki ilişkiyi ölçen bir istatistiksel terimdir. Korelasyon, bu iki değişken arasındaki doğrusal ilişkinin gücünü ve yönünü belirtir. İki değişken arasındaki ilişki pozitif, negatif veya sıfır olabilir.

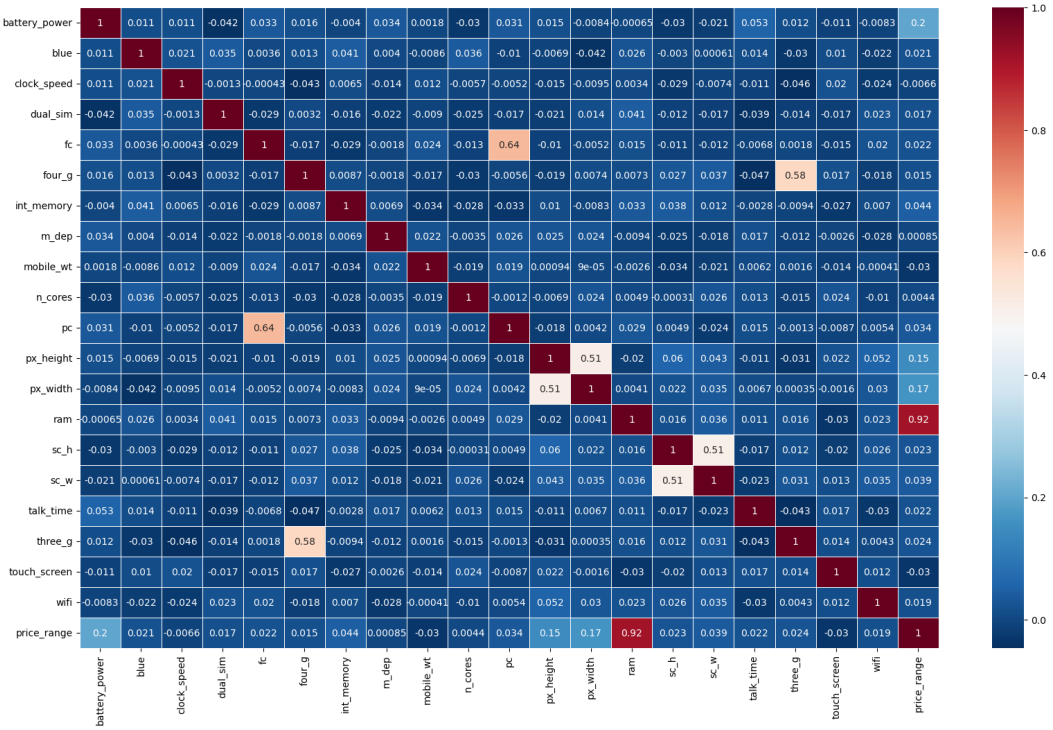
Korelasyon genellikle iki değişkenin birbirine ne kadar benzediğini veya birbirini nasıl etkilediğini anlamak amacıyla kullanılır. Korelasyon katsayısı, -1 ile 1 arasında bir değer alır:

1: Mükemmel pozitif korelasyon. Bir değişken artarken diğer değişken de artar.

0: Korelasyon yok (veya çok zayıf). Değişkenler arasında bir ilişki yoktur

1: Mükemmel negatif korelasyon. Bir değişken artarken diğer değişken azalır.

Görüldüğü üzere çok fazla ve ayrıntılı özellik var mesela m_dep kamera odaklanma özelliğinin fiyata pek bir etkisi olmaması gerekir. sc_h ve sc_w ekran genişliği ve yüksekliği olan iki özellik bir özelliğe indirgenebilir (ekran genişiği). Aynı şekilde px_width ve px_height te ekran çözünürlüğü olarak bir özelliğe indirgenebilir. Daha az özellik daha esnek ve daha az karmaşık bir model demektir. Aşağıda temizlenmeyen veri setinin korelasyon matrisi verilmiştir bunu inceleyerek bazı özellikler kaldıralacaktır.



Korelasyon matrisinden de görüldüğü üzere m_dep'in price_range ile neredeyse hiçbir alakası bulunmamakta bu yüzden bu özellik kaldırılıp boyut azaltma açısından birkaç özellik de birleştirilmiştir.

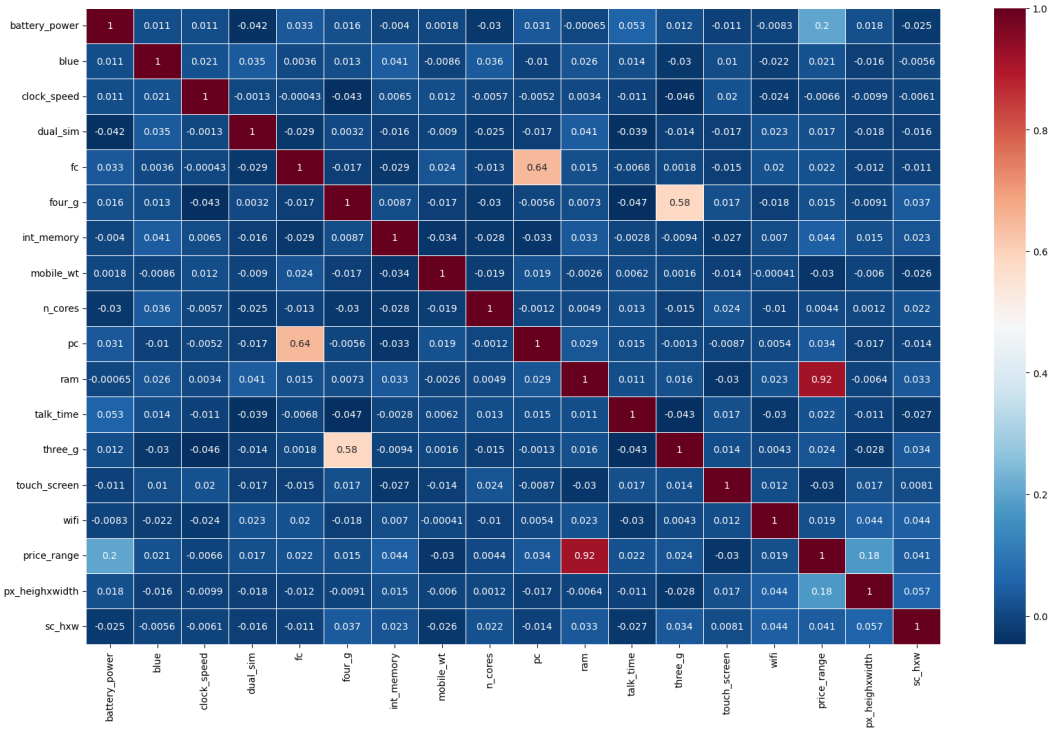
```
df = data.copy()

#Özellik mühendisliği
```

```
df['px_heighxwidth'] =df['px_height']*df['px_width']
df['sc_hxw'] = df['sc_h']*df['sc_w']
```

```
In [56]: df.shape
Out[56]: (2000, 18)
```

Şu an veri setinde toplamda 18 özellik bulunmaktadır. Aşağıda özellik mühendisliğnden sonra çizilmiş bir korelasyon matrisi bulunuyor.



price_range ram ile güçlü bir korelasyon ilişkisine sahip ayrıca birleştirdiğimiz özellikler orijinal özelliklere göre daha fazla korelasyon gösterdi.

5. Özellik ölçeklendirme ve Eğitim ve Test setlerini hazırlama

Özellik Ölçeklendirme ve Önemi:

Özellik ölçeklendirme, bir makine öğrenimi modeli oluşturulmadan önce veri setindeki özellik değerlerini belirli bir aralığa getirme veya standartlaştırma işlemidir. Bu işlem, veri setindeki özellik değerlerinin farklı ölçeklere sahip olması durumunda ortaya çıkan problemleri çözmek ve model performansını artırmak için kullanılır.

Neden Önemlidir:

1. **Model Performansı:** Özellikler arasındaki büyük değer farklılıkları, bazı algoritmaların modeli yanlış anlamasına veya yanıltmasına neden olabilir. Örneğin, bir özellik diğerine göre çok büyük değerler içeriyorsa, bu özellik diğerlerini domine edebilir. Ölçeklendirme, bu tür etkileşimleri düzeltir.
2. **Optimizasyon Hızı:** Birçok optimizasyon algoritması, özelliklerin aynı ölçeğe sahip olmasını tercih eder. Ölçeklendirme, optimizasyon algoritmalarının daha hızlı ve istikrarlı bir şekilde çalışmasına yardımcı olabilir.
3. **Model Karşılaştırmaları:** Farklı ölçeklerdeki özelliklerle oluşturulan modelleri karşılaştırmak zordur. Ölçeklendirme, modellerin adil bir şekilde karşılaştırılmasını sağlar.
4. **Hassasiyet ve Hızlı Yakınsama:** Ölçeklendirme, modelin daha hızlı ve daha hassas bir şekilde yakınsamasına katkıda bulunabilir. Bu, eğitim sürecini iyileştirir ve daha iyi sonuçlar elde edilmesini sağlar.

Özellikle sayısal değerler içeren veri setlerinde, özellik ölçeklendirme işlemi, modelin daha güvenilir ve etkili bir şekilde çalışmasını sağlar.

PCA (Principal Component Analysis - Temel Bileşen Analizi):

PCA, bir veri setindeki değişkenliği azaltmak ve veri setindeki temel yapıları (temel bileşenleri) belirlemek amacıyla kullanılan bir boyut indirgeme tekniğidir. Temel olarak, PCA, veri setindeki değişken sayısını azaltarak veri setindeki önemli bilgileri korumaya çalışır

```
X = df.drop('price_range', axis=1)  # Özellikler
y = df['price_range']  # Hedef

# Veriler eğitim ve test olarak bölme (80% eğitim, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_s

# Özelliklerin standardizasyonu
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# PCA uygulanması
pca = PCA(n_components=0.95)
X_train_pca = pca.fit_transform(X_train_scaled)
X_test_pca = pca.transform(X_test_scaled)
```

StandartScaler() nedir :

`StandardScaler`, veri setindeki sayısal özellikleri standartlaştırmak için kullanılan bir ölçeklendirme yöntemidir. Standartlaştırma işlemi, özellik değerlerini ortalaması 0, standart sapması 1 olacak şekilde dönüştürerek, özelliklerin aynı ölçeğe sahip olmasını sağlar.

6. Hiper Parametrizayon , Model Eğitimi ve Performans ölçekleri :

Hiperparametre Ayarlaması (Hiperparametre Optimizasyonu):

Hiperparametreler, bir makine öğrenimi modelinin yapılandırılmasını ve performansını etkileyen parametrelerdir. Bu parametreler, modelin öğrenme sürecini kontrol eder ve modelin genel başarısını belirleyen önemli faktörlerdir. Hiperparametre ayarlaması veya optimizasyonu, bu kritik parametrelerin en iyi değerlerini bulmayı amaçlar.

Çapraz Doğrulama (Cross-Validation):

Çapraz doğrulama, bir makine öğrenimi modelinin performansını değerlendirmek ve güvenilir sonuçlar elde etmek amacıyla kullanılan bir yöntemdir. Bu yöntem, modelin gerçek dünya verilerine daha iyi genelleme yapabilme yeteneğini değerlendirmek için kullanılır.

Grid Search Cross Validation (Izgara Arama Çapraz Doğrulama):

`GridSearchCV` (Grid Search Cross-Validation), makine öğrenimi modellerinde kullanılan hiperparametre optimizasyonu için bir yöntemdir. Bu yöntem, belirli bir hiperparametre uzayındaki farklı kombinasyonları sistematik olarak deneyerek, en iyi hiperparametre setini bulmayı amaçlar. Ayrıca, bu işlemi çapraz doğrulama (cross-validation) ile birleştirerek modelin genelleme yeteneğini de değerlendirir.

```
#Hiper Parametre Optimizasyonu ve model doğruluğu
param_grid = {
    'C': [0.1, 1, 10],
    'kernel': ['linear', 'rbf', 'poly'],
    'gamma': ['scale', 'auto'],
    'degree': [2, 3, 4]
}

svm_classifier = SVC(random_state=42)
grid_search = GridSearchCV(svm_classifier, param_grid, cv=5, sc
grid_search.fit(X_train_pca, y_train)


print("En iyi Parametreler:", grid_search.best_params_)

best_svm_classifier = grid_search.best_estimator_

cv_scores = cross_val_score(best_svm_classifier, X_train_pca, y_
print("Cross-Validation Skorları:", cv_scores)
print("Ortalama Cross-Validation Puanı:", cv_scores.mean())
```

```
y_pred = best_svm_classifier.predict(X_test_pca)
print("\nSVM Sınıflandırıcı (En İyi Parametreler) - Test Seti:")
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test,
```

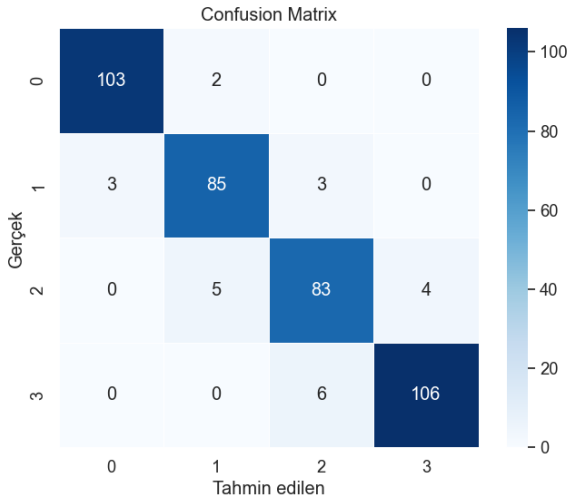
Bu kodun açıklaması ve nelerin kullanılıp ne amaca hizmet ettiği aşağıda belirtilmiştir :

1. **Parametre Uzayının Belirlenmesi:**
   - `param_grid` içinde belirtilen parametre uzayı, `C` (C değeri), `kernel` (kernel fonksiyonu), `gamma` (gamma değeri), ve `degree` (polinom derecesi) gibi SVM sınıflandırıcısının önemli hiperparametrelerini içerir. Bu parametreler, modelin performansını etkiler.
2. **SVM Sınıflandırıcı Modelinin Oluşturulması:**
   - `SVC` sınıflandırıcı modeli `random_state=42` ile oluşturulur.
3. **Grid Search Çapraz Doğrulama (Grid Search Cross-Validation):**
   - `GridSearchCV`, belirtilen parametre uzayındaki farklı kombinasyonları deneyerek en iyi performansı gösteren hiperparametreleri bulmaya çalışır. Bu işlem, `cv=5` ile belirtilen 5 katlı çapraz doğrulama (cross-validation) kullanılarak gerçekleştirilir.
4. **En İyi Parametrelerin Bulunması:**
   - Grid Search işlemi tamamlandıktan sonra, `best_params_` özelliği ile en iyi parametre seti görüntülenir.
5. **En İyi Modelin Seçilmesi:**
   - En iyi parametre seti kullanılarak `best_estimator_` ile en iyi model seçilir.
6. **Çapraz Doğrulama ile Performans Değerlendirmesi:**
   - En iyi model, çapraz doğrulama ile değerlendirilir. `cross_val_score` fonksiyonu, modelin belirtilen metrik (accuracy) üzerinde performansını değerlendirir.
7. **Test Setinde Performans Değerlendirmesi:**
   - En iyi model kullanılarak test seti üzerinde performans değerlendirmesi yapılır. `accuracy_score` fonksiyonu ile doğruluk oranı hesaplanır ve `classification_report`

ile detaylı sınıflandırma raporu oluşturulur.

Bu kod bloğu, SVM sınıflandırıcısının en iyi hiperparametre seti ile eğitilip değerlendirilmesini sağlar ve bu sayede modelin genelleme yeteneğini artırmayı amaçlar. Çapraz doğrulama, modelin stabilite ve güvenilirlik açısından değerlendirilmesinde önemli bir rol oynar.

Son olarak Performans olçüsü olarak confusion Matrix :



7. Yapılan işin yorumlanması :

Bu projede, çeşitli özelliklere sahip bir telefon veri seti üzerinde makine öğrenimi modellerini kullanarak telefon fiyat tahmini gerçekleştirdik. Veri setinde yer alan sayısal ve kategorik özellikleri keşfetmek ve analiz etmek amacıyla ön işleme aşamalarını gerçekleştirdik.

Veri setinin zenginliği içerisinde, telefonların fiyatlarını etkileyebilecek birçok faktörü ele aldık. Ardından, Random Forest, SVM gibi sınıflandırma modellerini kullanarak fiyat tahmini yapmaya odaklandık. Model performansını artırmak için Grid Search yöntemini kullanarak en iyi hiperparametreleri belirledik.

Boyut azaltma işlemi için Temel Bileşen Analizi (PCA) kullandık. Bu sayede, modelimizi daha az boyutlu bir veri seti üzerinde eğiterek hem hesaplama maliyetini düşürdük hem de modelin genelleme yeteneğini artırdık.

Performans değerlendirmesi için çapraz doğrulama (cross-validation) yöntemini ve Confusion Matrix'i kullandık. Elde ettiğimiz sonuçları daha anlaşılır hale getirmek adına seaborn kütüphanesini kullanarak görselleştirdik. Böylece, modelimizin doğruluğunu, hassasiyetini ve diğer performans metriklerini detaylı bir şekilde inceledik.

Sonuç olarak, makine öğrenimi modelleri kullanarak yapılan bu telefon fiyat tahmini projesi, veri bilimi süreçlerinin başından sonuna kadar detaylı bir analizi içermektedir. Bu çalışma, hem veri seti üzerindeki özelliklerin anlaşılması hem de modelin güvenilir bir şekilde eğitilip değerlendirilmesi açısından önemli bilgiler sağlamaktadır.