DOKUZ EYLUL UNIVERSITESI

FEN FAKULTESI BILGISAYAR BILIMLERI

ODEV: KUMELEME

OGRENCI ADI SOYADI: SERKAN YILDIRIM

OGRENCI NUMARASI: 2023280165

OGRETMEN ADI: PROF.DR. EFENDI NASIBOGLU

### **1. K-Means Kümeleme Algoritması**

**K-Means**, verileri belirli bir sayıda (k) kümeye ayıran ve her küme için bir merkez (centroid) hesaplayan bir kümeleme algoritmasıdır. Bu algoritma, gözlemleri kümelemek için **Öklid uzaklığını** kullanır ve küme merkezlerini (centroid) belirler.

**Tanım:**

* **K-Means**, veri noktalarını k adet kümeye ayırarak her bir gözlemi en yakın kümeye atamayı amaçlar. Her küme, bir **merkez** (centroid) etrafında toplanan veri noktalarından oluşur.
* Bu merkezler, algoritmanın her iterasyonunda güncellenir. Kümelerin merkezleri, küme içindeki tüm veri noktalarının ortalaması alınarak hesaplanır.

**Adımlar:**

1. **Başlangıç**: İlk olarak, k küme sayısı seçilir ve her küme için rastgele bir başlangıç merkezi seçilir.
2. **Atama Adımı**: Her bir veri noktası, en yakın merkeze atanır (Bu adımda genellikle **Öklid uzaklığı** kullanılır).
3. **Merkez Güncelleme**: Her küme için yeni bir **merkez** (centroid) hesaplanır. Yeni merkez, o kümeye ait tüm veri noktalarının ortalaması alınarak bulunur.
4. **Tekrar Etme**: Bu iki adım (atama ve merkez güncelleme) veri noktaları sabitlenene kadar tekrarlanır.

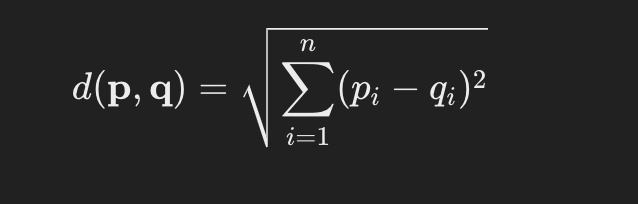
K-Means algoritması **denetimsiz öğrenme** yöntemlerinden biridir, yani veri üzerinde herhangi bir etiket (etiketli veri) kullanmadan çalışır. Kümeleme sonucunda, her bir veri noktası bir kümeye atanır ve kümelerin genel yapısı belirlenmiş olur.

**2. Öklid Uzaklığı (Euclidean Distance)**

**Öklid uzaklığı**, iki veri noktası arasındaki düz bir çizgi mesafesini hesaplamak için kullanılan en yaygın mesafe ölçüsüdür. K-Means algoritmasında, her bir veri noktasının hangi kümeye ait olduğunu belirlemek için bu mesafe kullanılır.

**Tanım:**

Öklid uzaklığı, bir noktadan diğerine olan mesafeyi hesaplamak için aşağıdaki formülle tanımlanır:



Burada:

* p ve q\mathbf{q}q iki veri noktasıdır.
* nnn, her veri noktasının boyutudur (örneğin, 2D veri için 2, 3D veri için 3).
* pip\_ipi​ ve qiq\_iqi​, sırasıyla p\mathbf{p}p ve q\mathbf{q}q veri noktalarının iii-inci koordinatındaki değerleridir.

**K-Means Algoritmasında Kullanımı:**

* K-Means algoritmasında, her veri noktası en yakın kümeye atanırken, **Öklid uzaklığı** kullanılır.
* Küme merkezleri ile her bir veri noktası arasındaki mesafeler hesaplanır ve veri noktası, en yakın küme merkezine atanır.
* Bu işlem, kümelerin merkezleri yeniden hesaplanana kadar devam eder.

**3. K-Means ve Öklid Uzaklığı Kullanımı**

K-Means algoritmasında **Öklid uzaklığı** temel bir bileşendir, çünkü algoritma her veri noktasının hangi kümeye ait olduğunu belirlerken bu mesafeyi kullanır. Kümeler, en yakın merkeze sahip olan veri noktalarından oluşur. Bu nedenle **Öklid uzaklığı**, algoritmanın başarısı ve doğruluğu için çok önemlidir.

Örnek:

* Bir evin fiyatı ve metrekaresi gibi özelliklere göre verilerimiz var. Bu veriler, K-Means algoritması ile üç farklı kümeye ayrılabilir: Ucuz, Orta ve Pahalı.
* Her bir ev, en yakın kümeye atanır. Öklid uzaklığı, her evin **hangi kümeye ait olduğunun** belirlenmesinde kullanılır.

**4. K-Means ve Öklid Uzaklığı Algoritmasının Avantajları ve Dezavantajları**

**Avantajlar:**

* **Basit ve Etkili**: K-Means algoritması, genellikle hızlı ve verimli bir algoritmadır.
* **Hızlı Çalışma**: Büyük veri setlerinde bile etkili çalışabilir.
* **Uygulama Kolaylığı**: Çeşitli alanlarda (örneğin, pazarlama, biyoloji, finans) yaygın olarak kullanılır.

**Dezavantajlar:**

* **Küme Sayısının Önceden Belirlenmesi Gerekiyor**: k (küme sayısı) önceden belirlenmelidir. Yanlış küme sayısı seçimi, kötü sonuçlar verebilir.
* **Başlangıç Noktasına Bağımlılık**: Başlangıç küme merkezlerinin konumu algoritmanın sonucunu etkileyebilir. Bunun önüne geçmek için genellikle birkaç farklı başlangıç yapılır.
* **Yuvarlak Küme Yapıları**: K-Means, küme yapılarının yuvarlak olmasını varsayar, dolayısıyla karmaşık (asimetriği yüksek) yapılar kötü sonuçlar verebilir.

Kumeleme icin temizlenmis excel dosyasindaki verileri asagidaki python kodlari ile kumelendirdim.

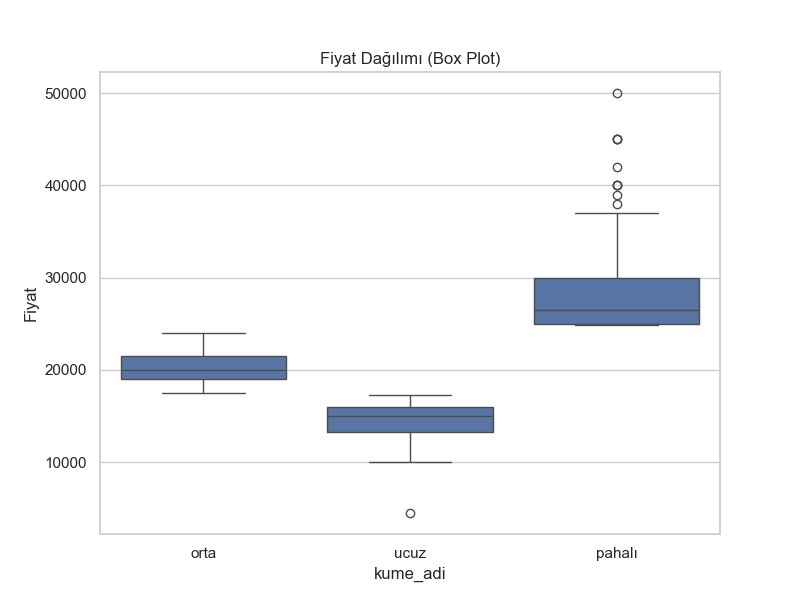
import pandas as pd  
from sklearn.preprocessing import StandardScaler  
from sklearn.cluster import KMeans  
  
# Excel dosyasını yükleyin  
df = pd.read\_excel('temiz\_buca\_kiralik\_daireler.xlsx') # kendi dosya yolunuzu ekleyin  
  
# Fiyat sütununu seçin ve normalleştirin  
scaler = StandardScaler()  
df['fiyat\_normalize'] = scaler.fit\_transform(df[['Fiyat']])  
  
# K-means algoritmasını kullanarak 3 küme oluşturun  
kmeans = KMeans(n\_clusters=3, random\_state=0)  
df['kume'] = kmeans.fit\_predict(df[['fiyat\_normalize']])  
  
# Küme merkezlerine göre "ucuz", "orta", "pahalı" olarak etiketleme  
# Önce küme merkezlerinin ortalamalarını alıyoruz ve küçükten büyüğe sıralıyoruz  
kume\_siralamasi = kmeans.cluster\_centers\_.flatten().argsort()  
kume\_etiketleri = {kume\_siralamasi[0]: 'ucuz', kume\_siralamasi[1]: 'orta', kume\_siralamasi[2]: 'pahalı'}  
df['kume\_adi'] = df['kume'].map(kume\_etiketleri)  
  
# Sonuçları yeni bir Excel dosyasına kaydedin  
df.to\_excel('evler\_kumelenmis.xlsx', index=False)

Analiz ve grafik icin ise bunlari kullanarak nesne sayisi, min, max, standart sapmasi ve histogram seklinde excelini asagidaki kodlarla olusturdum.

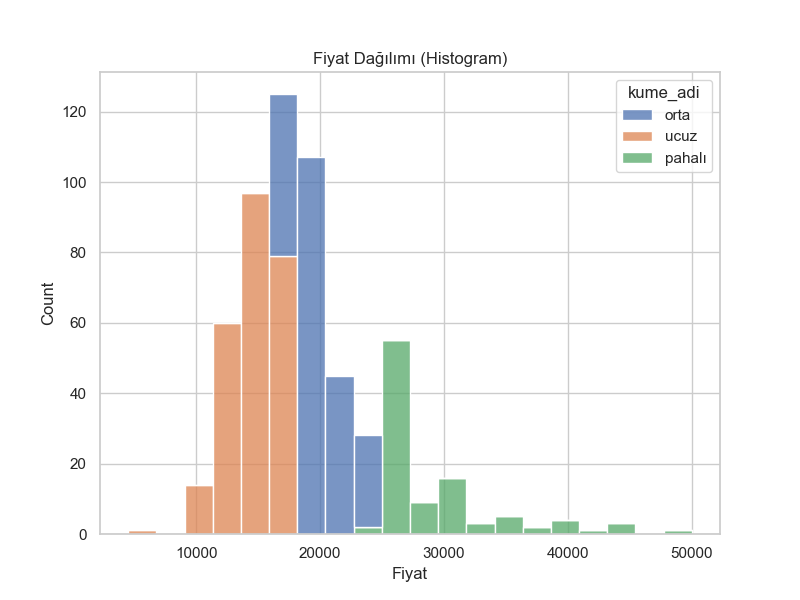
import pandas as pd  
import matplotlib.pyplot as plt  
import seaborn as sns  
  
# Excel dosyasını yükleyin (kume ve kume\_adi sütunları içeren dosya)  
df = pd.read\_excel('evler\_kumelenmis.xlsx') # dosya yolunu kendinize göre ayarlayın  
  
# Her bir kume\_adi için min, max, ortalama, standart sapma ve nesne sayısını hesaplayın  
istatistikler = df.groupby('kume\_adi')['Fiyat'].agg(['min', 'max', 'mean', 'std', 'count']).reset\_index()  
print(istatistikler)  
  
# Sonuçları Excel'e kaydetmek için  
istatistikler.to\_excel('kume\_istatistikleri.xlsx', index=False)  
  
# Grafik çizimi için görselleştirme ayarları  
sns.set(style="whitegrid")  
  
# Histogram çizimi - Kümelere göre fiyat dağılımı  
plt.figure(figsize=(8, 6))  
sns.histplot(data=df, x='Fiyat', hue='kume\_adi', multiple='stack', bins=20)  
plt.title('Fiyat Dağılımı (Histogram)')  
plt.savefig('fiyat\_histogram.png') # Grafik kaydedildi  
plt.clf() # Önceki grafiği temizle  
  
# Kutu grafiği (box plot) çizimi - Her kümenin fiyat dağılımı  
plt.figure(figsize=(8, 6))  
sns.boxplot(data=df, x='kume\_adi', y='Fiyat')  
plt.title('Fiyat Dağılımı (Box Plot)')  
plt.savefig('fiyat\_boxplot.png') # Grafik kaydedildi  
plt.clf() # Önceki grafiği temizle  
  
# Ortalama değer çubuğu grafiği - Kümelere göre ortalama fiyat  
plt.figure(figsize=(8, 6))  
sns.barplot(data=istatistikler, x='kume\_adi', y='mean')  
plt.title('Kümelere Göre Ortalama Fiyat')  
plt.savefig('ortalamalar.png') # Grafik kaydedildi  
plt.clf() # Önceki grafiği temizle  
  
# Standart sapma çubuğu grafiği - Kümelere göre standart sapma  
plt.figure(figsize=(8, 6))  
sns.barplot(data=istatistikler, x='kume\_adi', y='std')  
plt.title('Kümelere Göre Standart Sapma')  
plt.savefig('standart\_sapma.png') # Grafik kaydedildi  
plt.clf() # Önceki grafiği temizle

resimlerini tek tek kayit ettirdim.

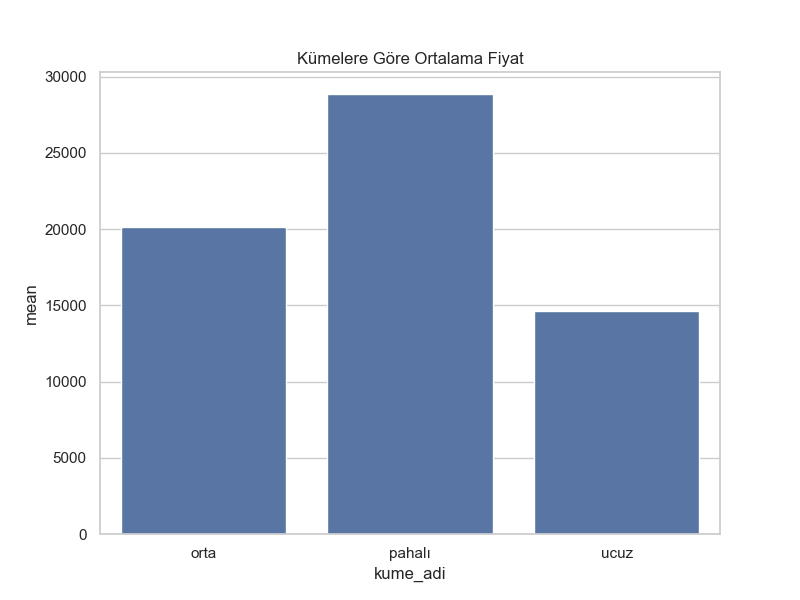
Fiyat kutu grafigi



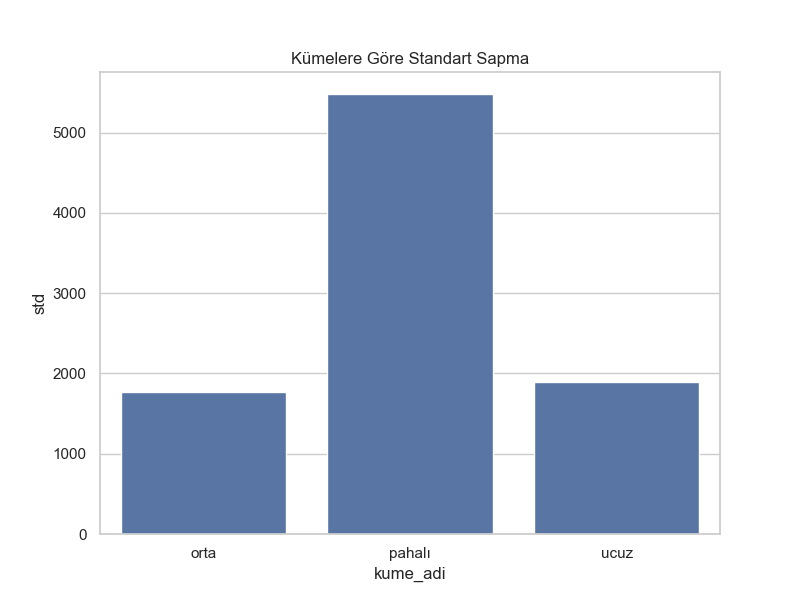
Fiyat histogram



Ortalamalar



Standart sapma



Kume analiz grafigi 4u birlikte

