### **DEÜ Fen Fakültesi**

### **Bilgisayar Bilimleri Bölümü**

### **BİL 3013 Veri Madenciliğine Giriş**

### **Ödev** 2. Kümeleme

### **Kodların ve Veri setlerinin Kaggle linki:**

[**https://www.kaggle.com/code/ozgurd5/bil3013-data-mining-assignment-2-code**](https://www.kaggle.com/code/ozgurd5/bil3013-data-mining-assignment-2-code)

[**https://www.kaggle.com/datasets/ozgurd5/bil3013-data-mining-assignment-2-data**](https://www.kaggle.com/datasets/ozgurd5/bil3013-data-mining-assignment-2-data)

### **Grup 2**

### **Öğrenci: Özgür Dalbeler - 2022280084**

### **Öğretim Üyesi: Prof. Dr. Efendi NASİBOĞLU**

### **İzmir 2024**

### 1- Ödevin Tanımı

Herkes bir önceki ödevinde topladığı verilerdeki Fiyat (veya Kira bedeli) atributunu

kullanarak, piyasayı “Ucuz”, “Orta” ve “Pahalı” gibi 3 grupa (kümeye) ayırması gerekiyor.

Kümeleme için K-ortalamalar kümeleme algoritması ve Öklit uzaklığı kullanılmalıdır.

Sakai sistemine aşağıdaki dosyaların yüklenmesi gerekiyor:

a) Topladığınız verilerin temizlenmemiş halini içeren ham veriseti,

b) Temizlenmiş ve ön hazırlık işlemlerinden geçmiş veriseti,

c) Kümeleme sonuçlarının yansıtılması (kümeleme sonrası her verinin ait olduğu kümeyi

belirten numarası 0,1 veya 2 olarak temizlenmiş veri setine ayrı bir sütun şeklinde

eklensin)

d) Her bir kümedeki nesne sayısı, min, max, ortalama değeri, standart sapması,

histogramı gibi gösterge ve grafiklerini oluşturarak hazırlanmış ve piyasa analizini

içeren rapor dosyası,

e) Program kodlarını içeren dosya.

### 2- Kullanılan Yöntemler ve Teknolojiler

**Pandas:** Pandas, Python'da veri analizi ve manipülasyonu için kullanılan bir kütüphanedir. DataFrame, tablo yapısında verileri saklar ve CSV, XLSX, JSON gibi formatlardan veri okuma ve yazma imkanı sunar.

Kurulumu:  
pip install pandas

Kullanımı:  
import pandas as pd

df = pd.DataFrame(ilanlar)

**Openpyxl:** Pandas kütüphanesinin Excel dosyasına veri yazabilmesi için openpyxl kütüphanesinin yüklenmesi gerekir.

Kurulumu:  
pip install openpyxl

Kullanımı:

df.to\_excel('ilanlar.xlsx', index=False)

**Matplotlib:** Matplotlib, Python'da veri görselleştirme için kullanılan bir kütüphanedir. Grafikler, histogramlar, çubuk grafikler ve diğer görseller oluşturmak için sıkça kullanılır.

Kurulumu:

pip install matplotlib

Kullanımı:

import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]

y = [10, 20, 25, 30, 35]

plt.plot(x, y, marker='o')

plt.xlabel('X Ekseni')

plt.ylabel('Y Ekseni')

plt.title('Örnek Grafik')

plt.show()

### 3- Uygulama

Bu projede **hepsiemlak.com** sitesindeki Buca ilçesindeki kiralık dairelerin ilanlarının fiyatları k-means kümeleme yöntemi kullanılarak “Pahalı”, “Orta” ve “Ucuz” olmak üzere üç gruba kümelenmiştir.

Uygulamanın önemli adımları aşağıdaki gibidir:

**3.1- Modüller**

**# Pandas modülü**

**import pandas as pd**

**# Matplotlib modülü**

**import matplotlib.pyplot as plt**

**3.2- Ham veriyi okumak**

**# Excel dosyasını oku**

**df = pd.read\_excel("ham\_veri.xlsx")**

**3.3- Veriyi temizlemek**

Veri temizleme aşamasında duplikeler ve boş veriler kaldırılıp fiyat formatlanmış ve ardından veriler küçükten büyüğe sıralanmıştır.

**# Duplikeleri Link sütununa göre tespit et ve yazdır**

**duplikeler = df[df.duplicated(subset="Link", keep="first")]**

**print("Duplikeler (aynı linke sahip satırlar): ", len(duplikeler))**

**print(duplikeler[["Başlık", "Fiyat", "Link"]])**

**# Boş fiyatları tespit et ve yazdır**

**bos\_fiyatlar = df[df["Fiyat"].isna()]**

**print("\nBoş Fiyatı Olan Satırlar:", len(bos\_fiyatlar))**

**print(bos\_fiyatlar[["Başlık", "Fiyat", "Link"]])**

**# Duplikeleri kaldır**

**df = df.drop\_duplicates(subset="Link", keep="first")**

**# Fiyatı olmayan satırları kaldır**

**df = df.dropna(subset=["Fiyat"])**

**# Fiyatları düz sayı formatına çevir**

**# Değerler numpy değeri olarka dönüyor, bunu string'e çevir, ardından regex kullanarak sadece sayıları al ve integer'a çevir**

**# \D sayı olmayan karakterleri temsil eder**

**df["Fiyat"] = df["Fiyat"].astype(str).str.replace(r"\D", "", regex=True).astype(int)**

**# Fiyatları küçükten büyüğe sırala**

**df = df.sort\_values("Fiyat")**

**3.4- Outlier Tespiti**

Outlier tespiti için IQR yöntemi kullanılmıştır. Alt değer için katsayı 1.5 üst değer için ise 3. seçilmiştir.

# IQR yöntemi

# Çeyrek değerleri hesapla

Q1 = df["Fiyat"].quantile(0.25)

Q3 = df["Fiyat"].quantile(0.75)

IQR = Q3 - Q1

# Outlier sınırlarını belirle

lower\_bound = Q1 - 1.5 \* IQR

upper\_bound = Q3 + 3 \* IQR

# Outlierları seç

outliers = df[(df["Fiyat"] < lower\_bound) | (df["Fiyat"] > upper\_bound)]

print("\nOutlierlar (IQR yöntemi):")

print(outliers[["Fiyat"]])

# Outlierları kaldır

df = df[(df["Fiyat"] >= lower\_bound) & (df["Fiyat"] <= upper\_bound)]

**3.5- K-Means Kümeleme**

Hazır kütüphaneler aşırı kuvvetli ve çok parametreli olduğu için manuel k-means algoritması yazılmıştır.

k = 3

# Veri kümesindeki fiyat sütununu al.

# Değerler numpy değeri olarak dönüyor, bunu integer'a cast et, ardından listeye çevir

fiyatlar = df["Fiyat"].values.astype(int).tolist()

# K değeri kadar rastgele merkez seç

# Merkezlerin tipini float yap ve listeye çevir

merkezler = df.sample(n=k)["Fiyat"].values.astype(float).tolist()

# Önceki merkezleri tut, başlangıçta tüm değerleri 0 yap

önceki\_merkezler = [0] \* k

iterasyon\_sayacı = 0

# Merkezler değişene kadar döngüyü devam ettir

while True:

iterasyon\_sayacı += 1

print("iterasyon:", iteration\_counter)

# K değeri kadar küme oluştur

kümeler = []

for i in range(k):

kümeler.append([])

# Veri kümesindeki her fiyat için en yakın merkezi bul

for fiyat in fiyatlar:

en\_yakin\_merkez\_index = 0

en\_kucuk\_fark = abs(fiyat - merkezler[0]) # Öklit mesafesi

for i in range(1, k):

fark = abs(fiyat - merkezler[i]) # Öklit mesafesi

if fark < en\_kucuk\_fark:

en\_yakin\_merkez\_index = i

en\_kucuk\_fark = fark

kümeler[en\_yakin\_merkez\_index].append(fiyat)

# Her kümenin ortalamasını alarak yeni merkezleri hesapla

for i in range(k):

önceki\_merkezler[i] = merkezler[i]

merkezler[i] = sum(kümeler[i]) / len(kümeler[i])

# Merkezler değişmediyse döngüyü sonlandır

if önceki\_merkezler == merkezler:

break

**3.6- Grafikler**

Verileri görselleştirmek için sayıların değerlerinin gösterildiği sade bir grafik, scatter grafik ve box grafik kullanılmıştır.

# Grafik için plot oluştur

plt.figure(figsize=(10, 6), dpi=300)

# Grafik plot başlığını belirle

plt.title(f"Grafik")

# Index sütununu göster

plt.xlabel("Index")

# Fiyat sütununu göster

plt.ylabel("Fiyat")

# Grafik oluştur

plt.plot(range(len(fiyatlar)), fiyatlar, marker='o')

# Grafiği kaydet

plt.savefig("grafik.png", dpi=300, bbox\_inches='tight')

# Histogram için bir plot oluştur

plt.figure(figsize=(10, 6), dpi=300)

# Başlığı belirle

plt.title(f"Histogram")

# Fiyat sütununu göster

plt.xlabel("Fiyat")

# Histogram oluştur

plt.hist(fiyatlar, bins=100)

# Histogram'ı kaydet

plt.savefig("histogram.png", dpi=300, bbox\_inches='tight')

# Scatter için plot oluştur

plt.figure(figsize=(10, 6), dpi=300)

# Plot başlığını belirle

plt.title(f"K-Means Scatter Plot")

# Index sütununu göster

plt.xlabel("Index")

# Fiyat sütununu göster

plt.ylabel("Fiyat")

# Grid çiz

plt.grid(True)

# Veri kümesindeki index değerlerini x ekseninde göstermek için al

x\_values = df.index

# Veri kümesindeki fiyat değerlerini y ekseninde göstermek için al

y\_values = df["Fiyat"]

# Her küme için farklı renkler belirle

colors = ["red", "green", "blue"]

color\_values = []

for i in range(k):

color\_values.extend([colors[i]] \* len(kümeler[i]))

# Veri noktalarını çiz

plt.scatter(x\_values, y\_values, color=color\_values, s=2)

# Merkezleri çiz

plt.scatter(range(k), merkezler, color="black", s=100, marker="x")

# Scatter Plot'u kaydet

plt.savefig("scatter\_plot.png", dpi=300, bbox\_inches='tight')

# Box için bir plot oluştur

plt.figure(figsize=(10, 6), dpi=300)

# Başlığı belirle

plt.title(f"K-Means Box Plot")

# Index sütununu göster

plt.xlabel("Küme")

# Fiyat sütununu göster

plt.ylabel("Fiyat")

# Grid çiz

plt.grid(True)

# Kümeleri çiz

plt.boxplot(kümeler, patch\_artist=True, showmeans=True, showfliers=False)

# Box plot'u kaydet

plt.savefig("box\_plot.png", dpi=300, bbox\_inches='tight')

# Her bir küme için histogram oluştur

df["Küme"] = -1

for i in range(k):

df.loc[df["Fiyat"].isin(kümeler[i]), "Küme"] = i + 1

for i in range(k):

# Histogram için bir plot oluştur

plt.figure(figsize=(10, 6), dpi=300)

# Başlığı belirle

plt.title(f"Küme {i + 1} Histogram")

# Fiyat sütununu göster

plt.xlabel("Fiyat")

# Histogram oluştur

plt.hist(kümeler[i], bins=100)

# Histogram'ı kaydet

plt.savefig(f"küme\_{i + 1}\_histogram.png", dpi=300, bbox\_inches='tight')

**3.7- Veri Sayısı, Ortalama, Mod, Medyan, Standart Sapma, Min, Max**

**# Veri sayısı, ortalama, mod, medyan, standart sapma, minimum ve maksimum değerleri hesapla**

**veri\_sayisi = len(df)**

**print("\nVeri Sayısı:", veri\_sayisi)**

**ortalama = df["Fiyat"].mean()**

**print("Ortalama:", ortalama)**

**mod = df["Fiyat"].mode().values[0]**

**print("Mod:", mod)**

**medyan = df["Fiyat"].median()**

**print("Medyan:", medyan)**

**standart\_sapma = df["Fiyat"].std()**

**print("Standart Sapma:", standart\_sapma)**

**minimum = df["Fiyat"].min()**

**print("Minimum:", minimum)**

**maksimum = df["Fiyat"].max()**

**print("Maksimum:", maksimum)**

**# Her bir küme için bunları hesapla**

**for i in range(k):**

**print(f"\nKüme {i + 1} ({len(kümeler[i])} eleman):")**

**print("Ortalama:", sum(kümeler[i]) / len(kümeler[i]))**

**print("Mod:", pd.Series(kümeler[i]).mode().values[0])**

**print("Medyan:", pd.Series(kümeler[i]).median())**

**print("Standart Sapma:", pd.Series(kümeler[i]).std())**

**print("Minimum:", min(kümeler[i]))**

**print("Maksimum:", max(kümeler[i]))**

**3.8- Temizlenmiş Veriyi Kaydetmek**

**# Fiyatları küçükten büyüğe sırala, kümeleriyle birlikte yaz**

**df = df.sort\_values("Fiyat")**

**df["Küme"] = -1**

**for i in range(k):**

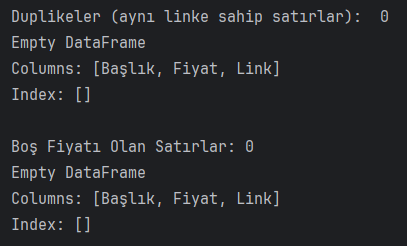
**df.loc[df["Fiyat"].isin(kümeler[i]), "Küme"] = i + 1**

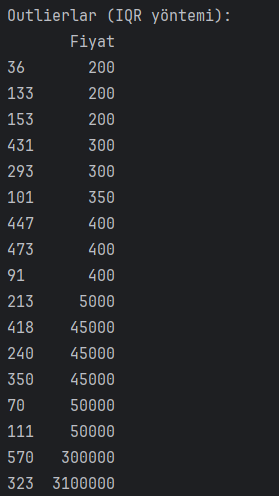
**# Excel dosyasına yaz**

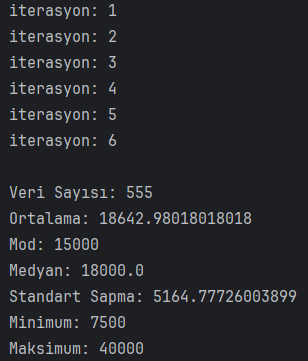
**print("\nExcel dosyası 'temizlenmiş\_veri.xlsx' olarak oluşturuldu.")**

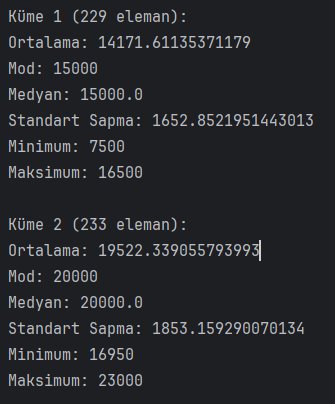
**df.reset\_index()[["Fiyat","Küme"]].to\_excel("temizlenmiş\_veri.xlsx", index=False)**

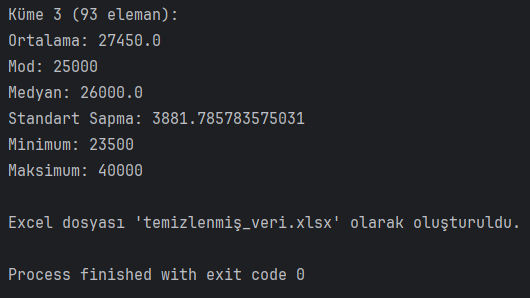
**3.9- Ekran Görüntüleri**

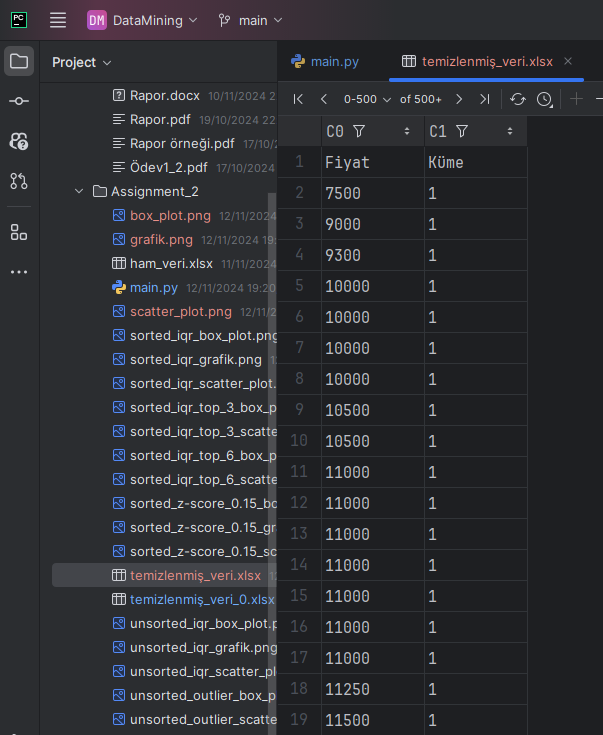
****

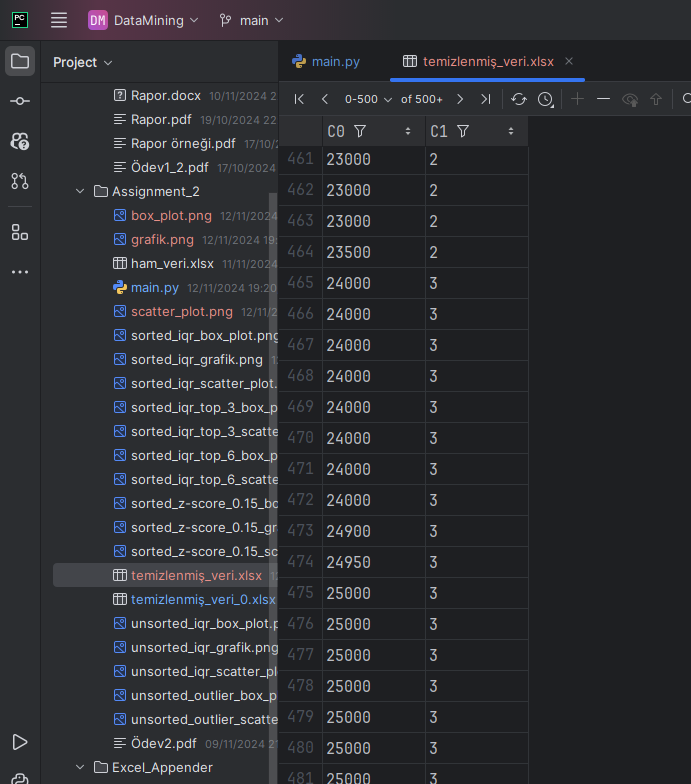
****

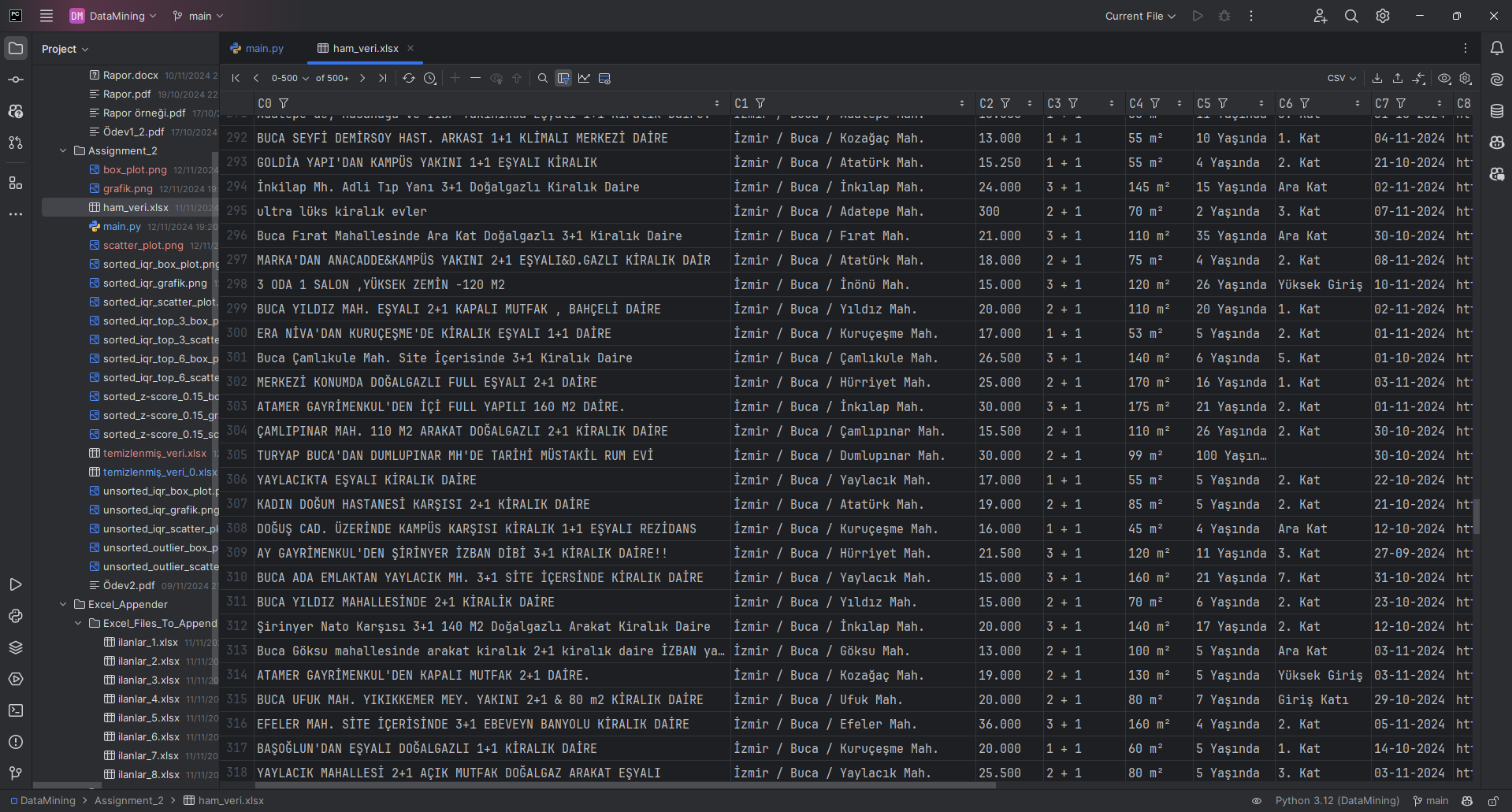
****

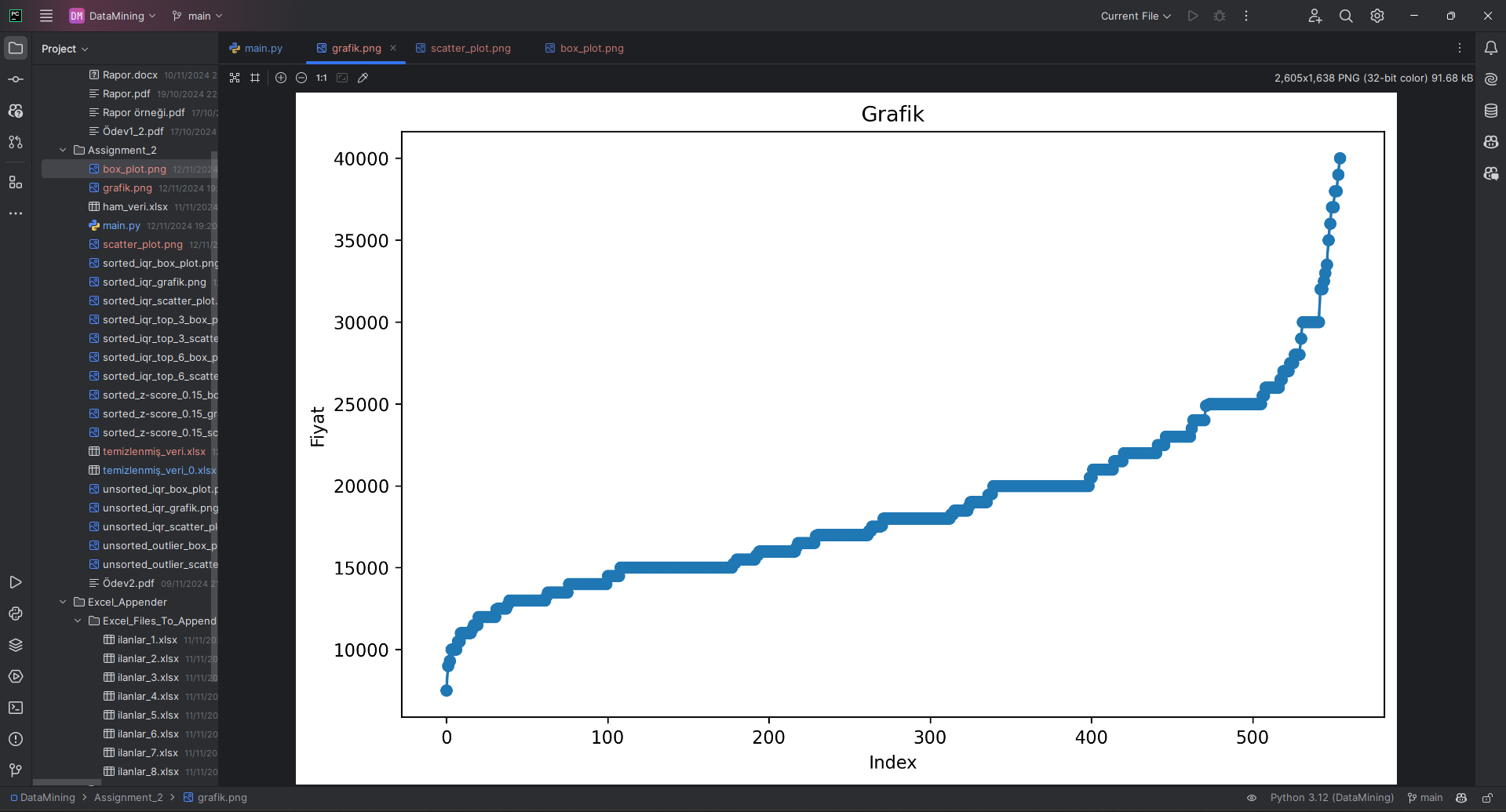
****

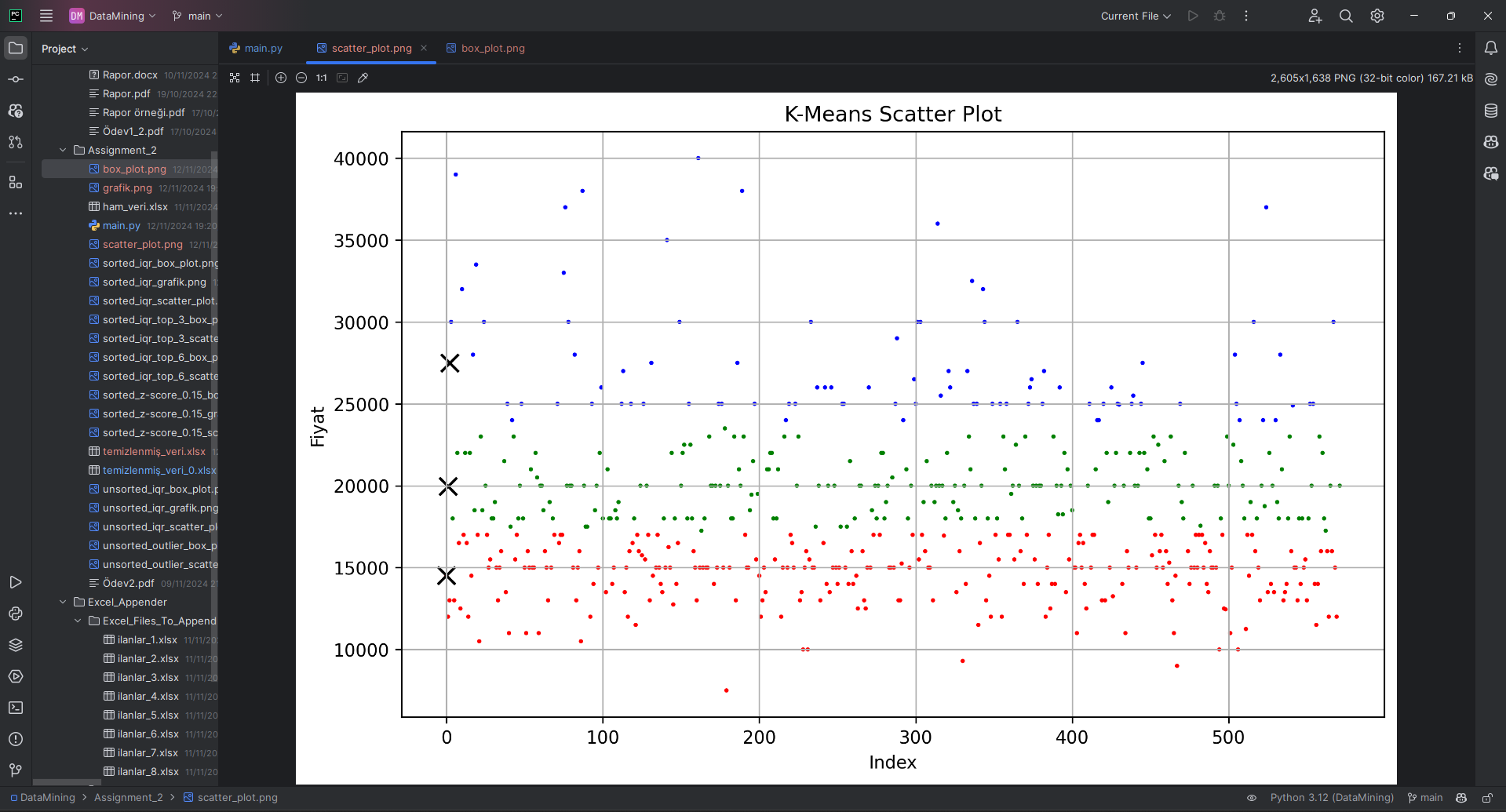
****

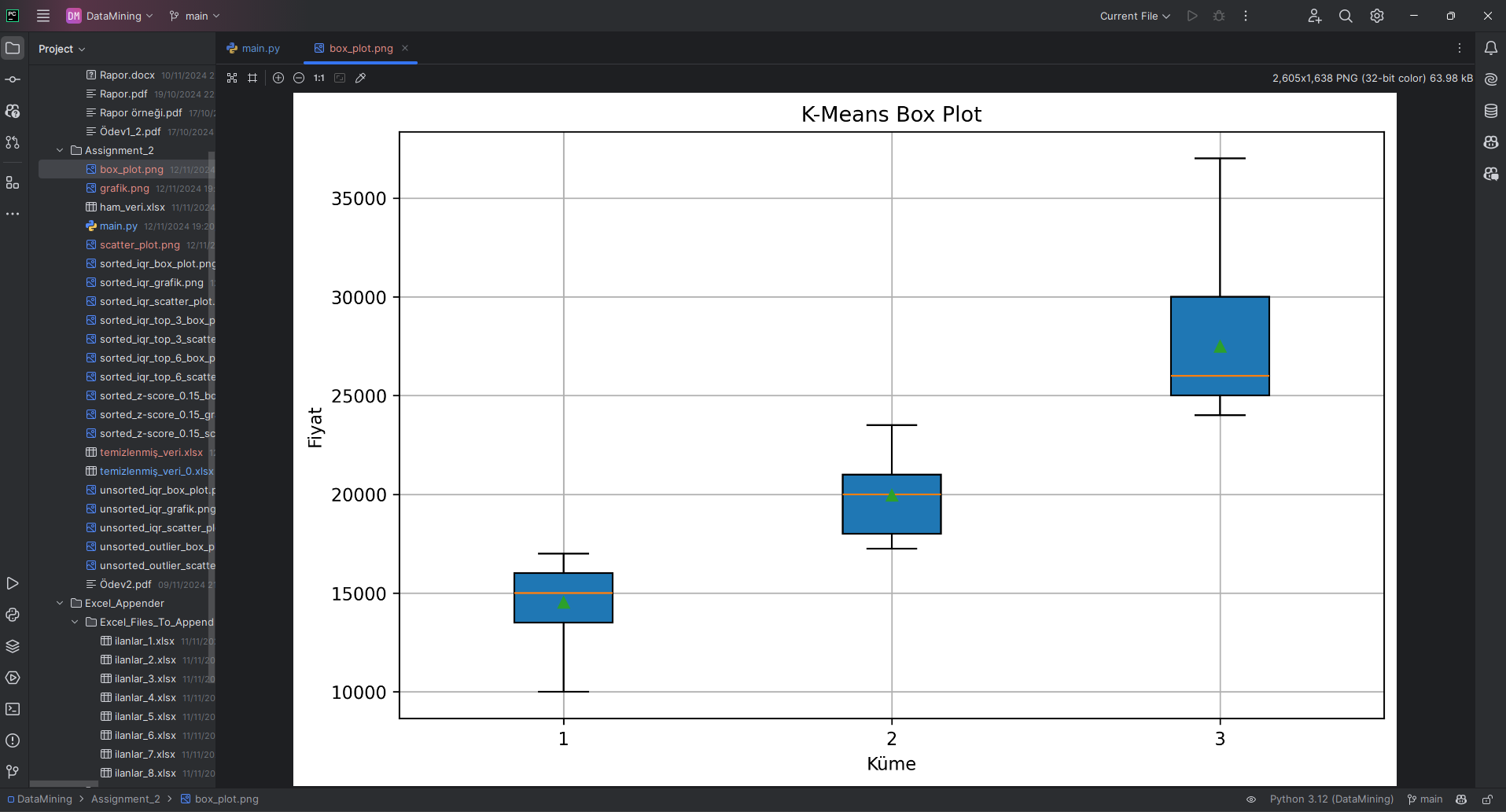
****

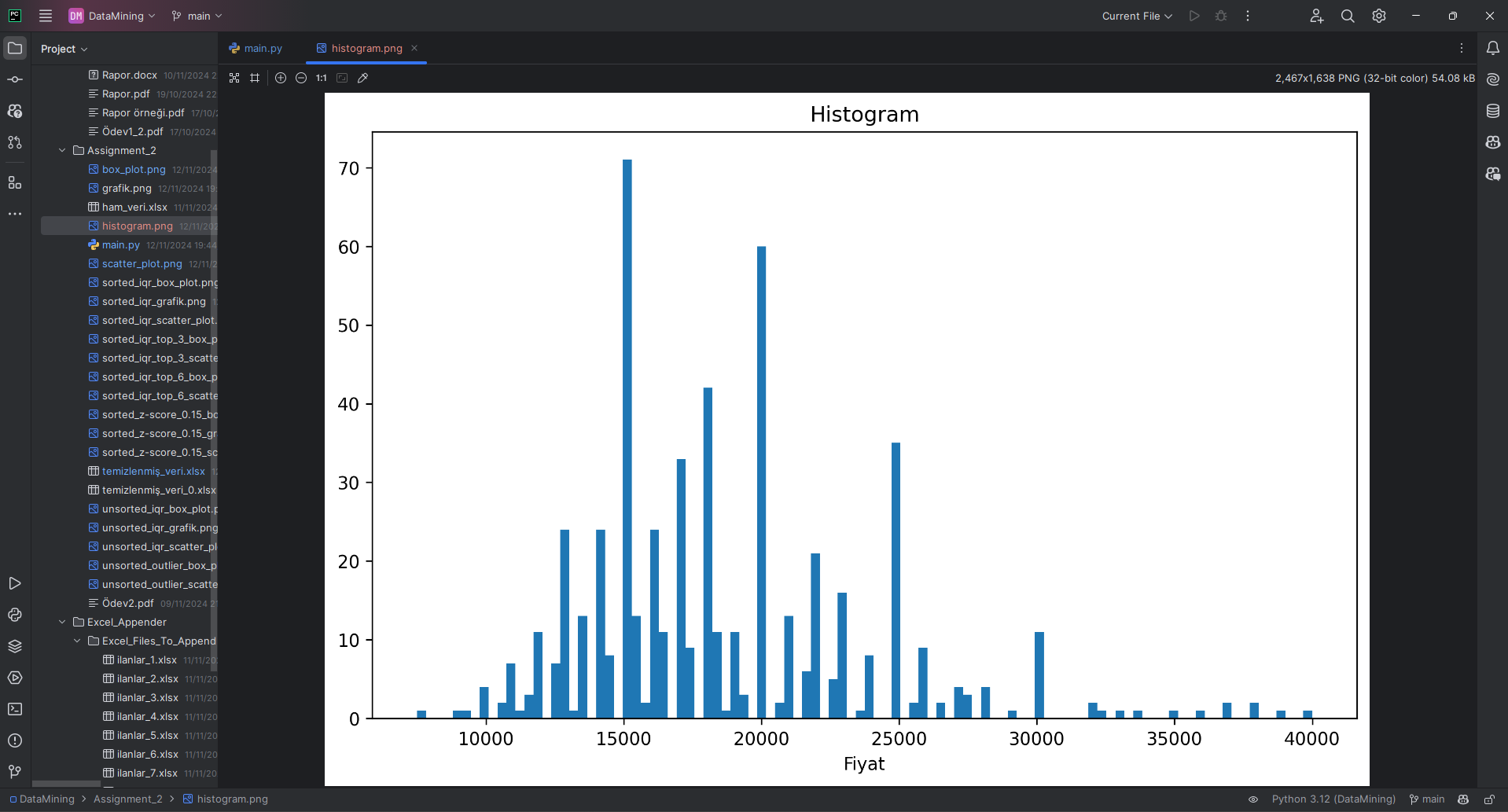
****

****

****

****

****

****

### 4- Değerlendirme

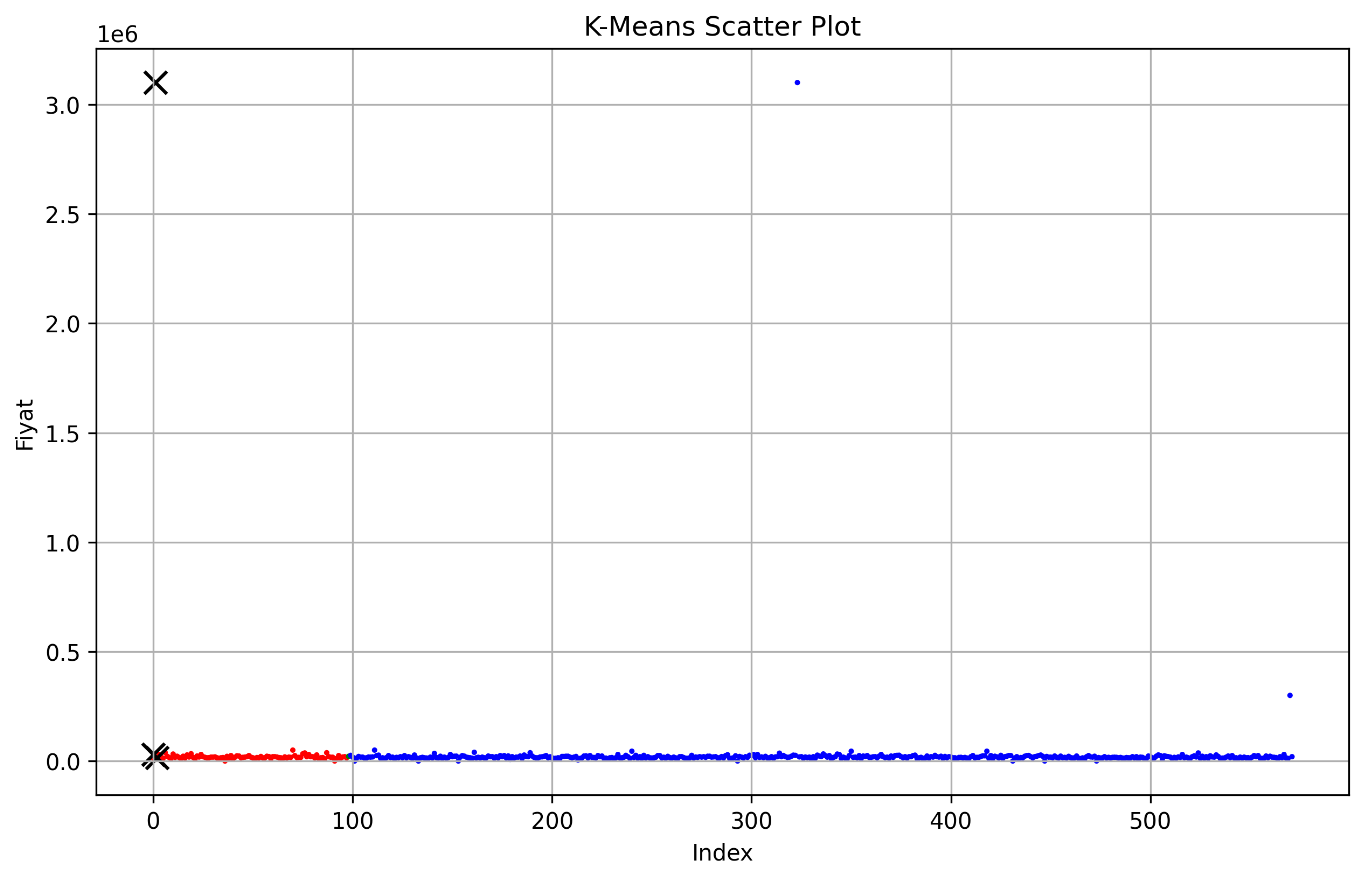
Projeyi yapmak için öncelikle elimde kayda değer miktarda veri olması gerekiyordu. Önceki ödevi az veri ile göndermiştim. Bu yüzden daha fazla veri çektim. Cloudflare doğrulaması her çıktığında modemimi açıp kapattım ve yeni bir ip adresine geçtim. Bu işlemi 8-9 defa yaptım. Elimde 573 ilan verisi mevcut olunca bıraktım.

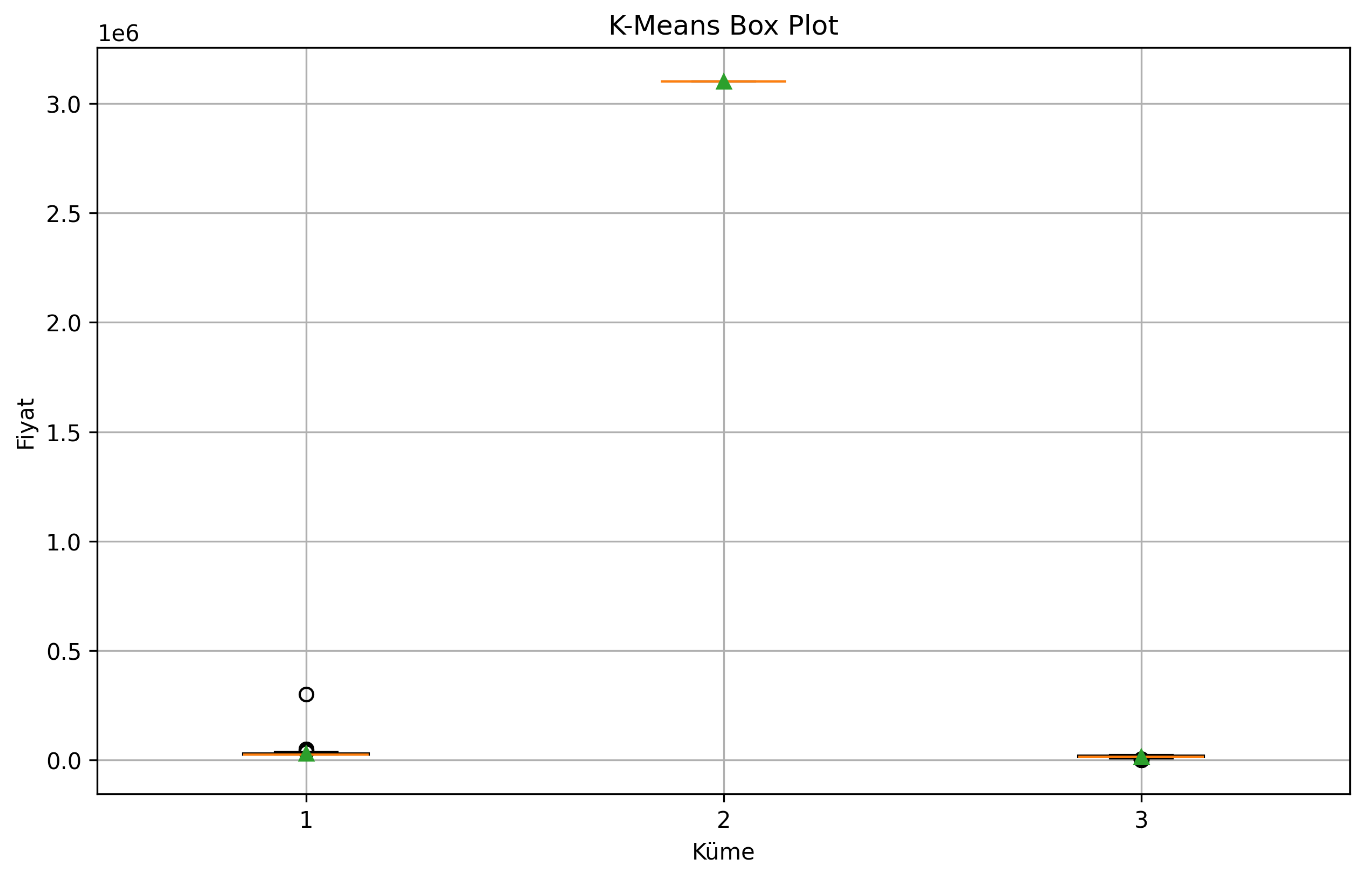
Ödevi yapmak için ilk önce yapay zekaya başvurdum ancak bu konuda fazlasıyla yetersizdi. Yeterli olduğu sohbetlerde ise ilerledikçe halüsinasyon görmeye başlayıp olmayan methodları ve attributeları kullanmaya başlıyordu. Kullandığım ide’deki copilot pek çok şeyi kolaylaştırmış olsa da o da çok iyi değildi. Bu yüzden kütüphane öğrenme sürecimi dökümantasyon, eğitim/tutorial siteleri ve forumlardan halletmeye çalıştım.

İlk olarak scikit-learn kütüphanesini kullandım ancak methodlar, parametreler ve genel olarak kullanımı hiç alışık olmadığım bir yoldaydı. Kendimi kaybolmuş ve seviyemin çok üstünde araçlar kullanıyormuş gibi hissettim. Bunun en büyük sebebi okul dışı zamanımın neredeyse tamamını oyun geliştirme, nesneye yönelik programlama ve c# ile geçirmem. Farklı kodlama tarzı, farklı programlama dili ve farklı konseptler ile uğraşıyordum. Bu yüzden slaytlarda yer alan k-means algoritmasını kendim yaptım ve dış kütüphanelere olabildiğince az bağlı olmaya çalıştım.

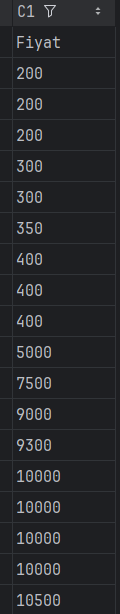
Pandas her ne kadar kolay kullanımı hedefleyen bir kütüphane olsa da veri tiplerinin belirsizliğinden doğan cast hataları, parametre hataları, operatörlerin geçersizliği gibi şeyler bana çok engel oldu. Python compile edilen bir dil olmadığı için bunların hepsi runtime’da belli oldu ve bu durum beni daha da yavaşlattı. Yine aynı sebepten ötürü kullandığım ide’nin yardımı çok az dokundu.

İlk çalışan verisyonumda verileri düzgün temizleyemediğimi fark ettim. Emlakçılar bazı ilanları satılık yerine kiralık olarak eklemiş ve bu ilanlar scrape yaparken veri setime girmiş. Kira bedeli 3 milyon olan bir ilan algortitmayı inanılmaz şekilde bozdu ve bu da böyle bir grafik ortaya çıkardı:

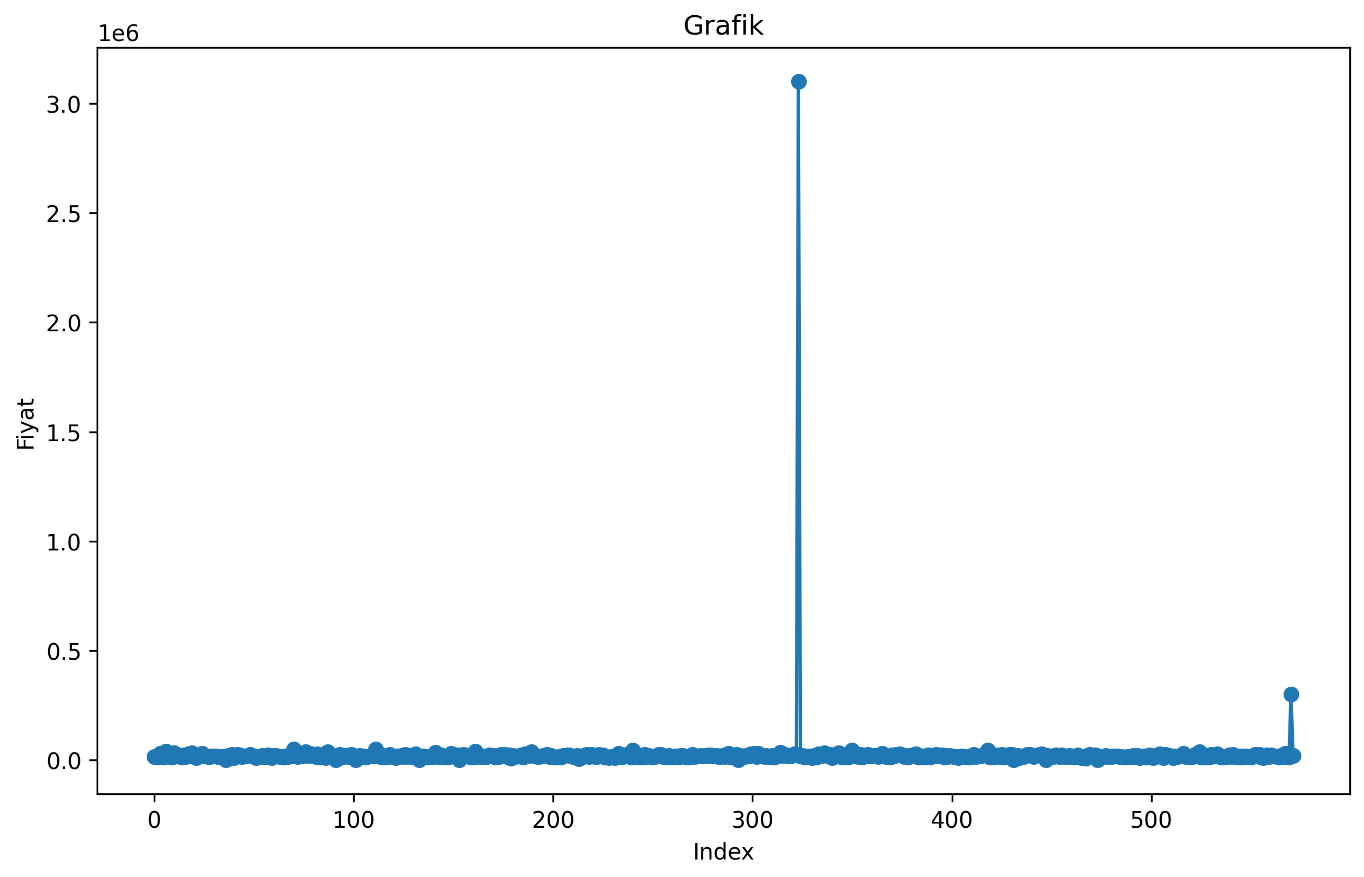




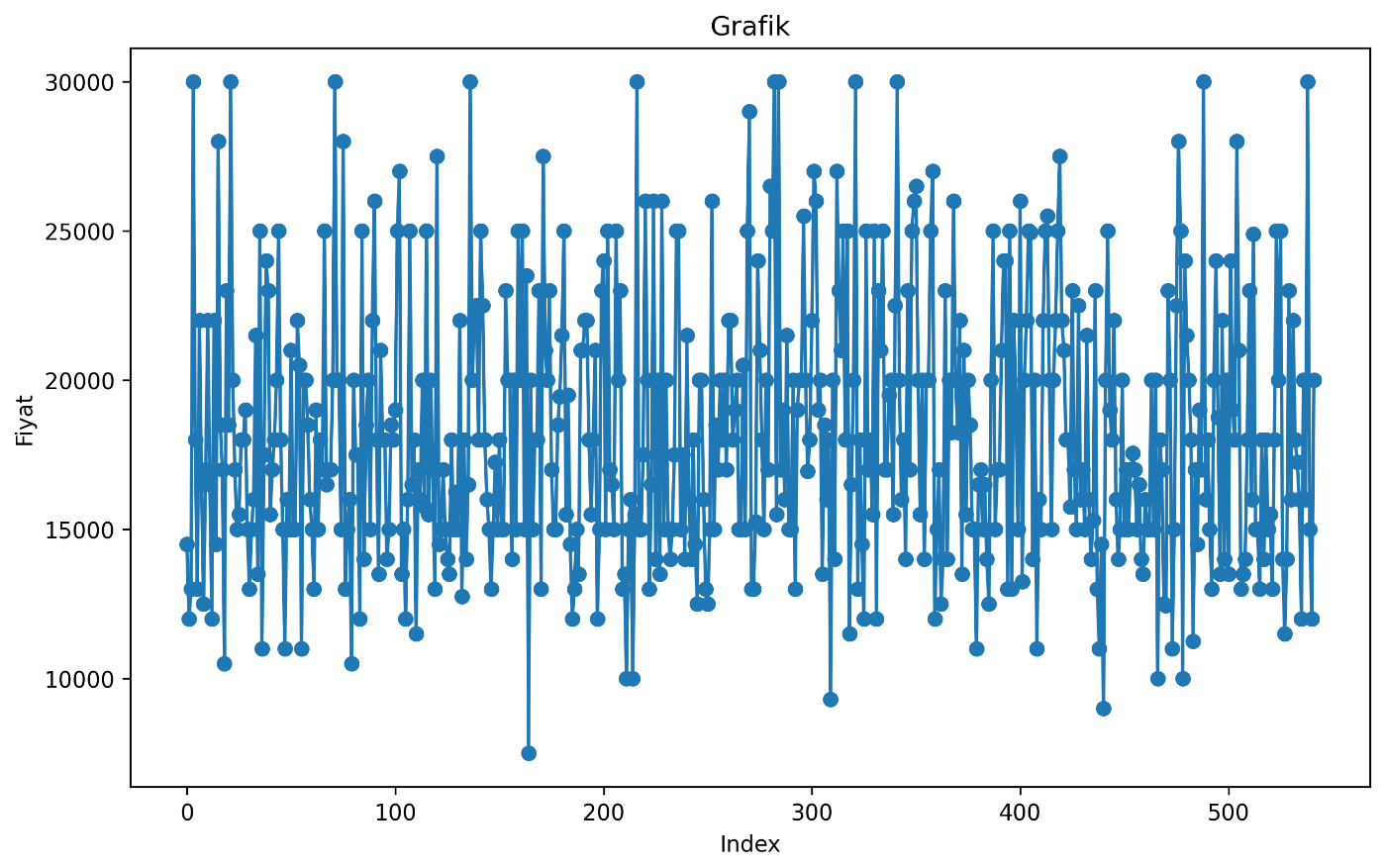
Daha sonra fiyat tablosuna biraz daha detaylı baktığımda kiralık daire ilanlarında günlük kiralık dairelerin de bulunduğunu ve bu ilanlarda fiyat olarak da günlük fiyat yazdığını fark ettim. Yani kira bedeli 200 TL, 300 TL olan pek çok ilan vardı. Bu gürültülerden kurtulmak gerekiyordu.

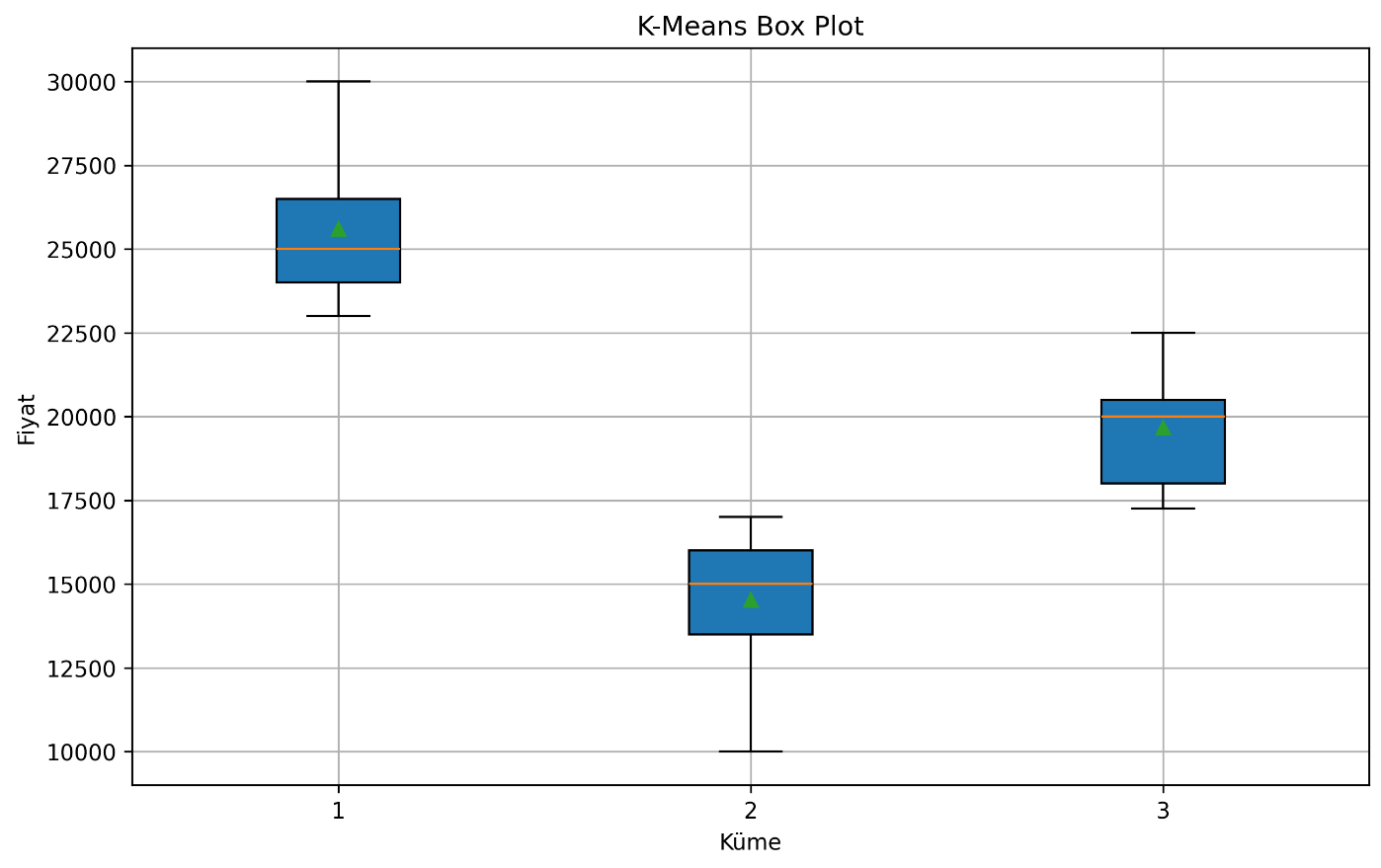


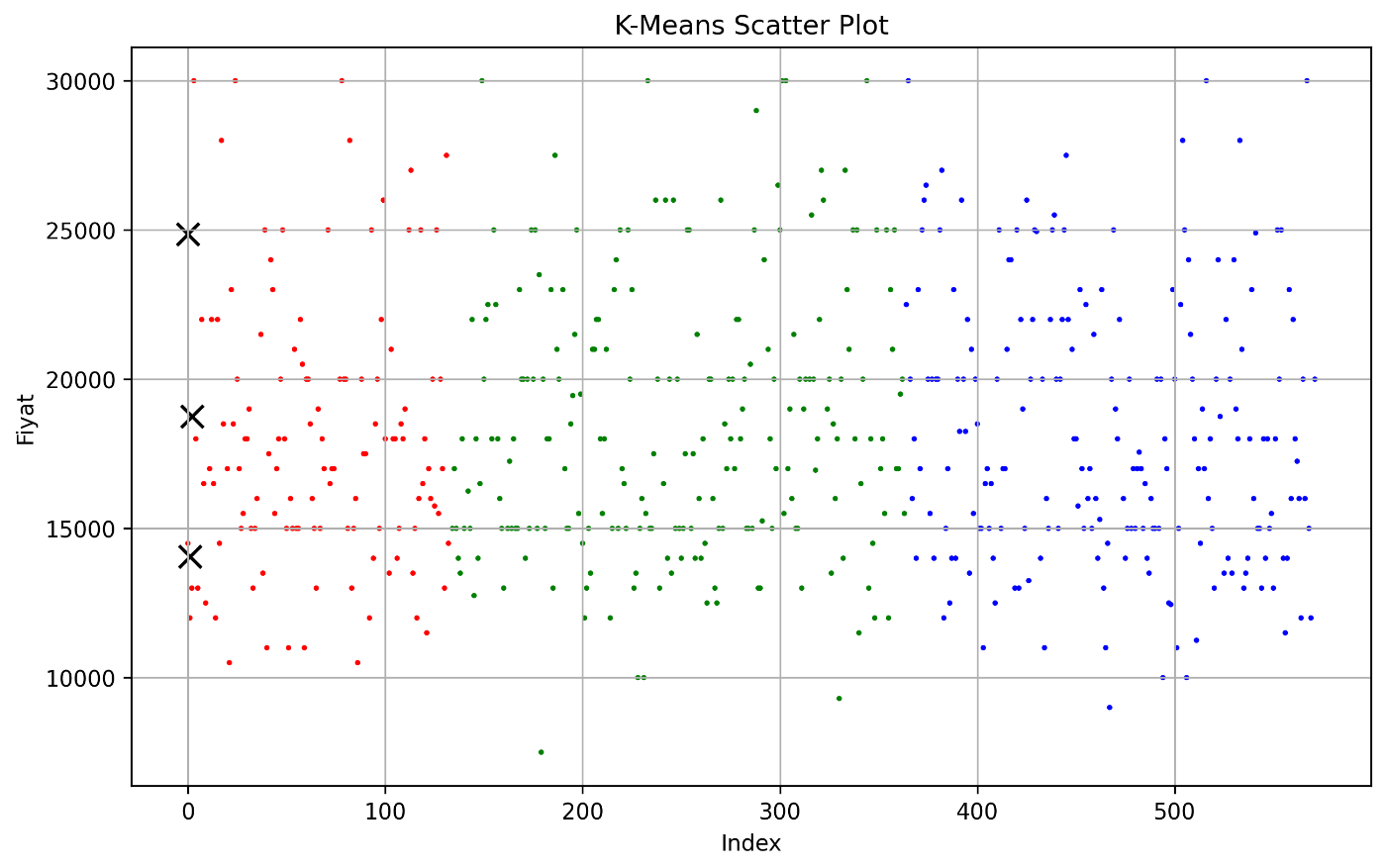
Gürültülerden kurtulmak için 2 yol kullanacaktım. Bunlardan birisi IQR yöntemi diğeri ise Z-Score yöntemi. IQR yöntemi eğilimli dağılım gösteren veriler, Z-Score ise normale yakın dağılım gösteren veriler için daha uygun. Verilerimin dağılımını gözlemlemek için grafik çizdim.



Bu grafik bana hiçbir bilgi vermedi. O yüzden iki yöntemi de denemeye karar verdim. Önce IQR ile başladım ve çarpanı genel olarak kullanılan 1.5 olarak belirledim. Verimi temizledim ve ardından bu grafikleri oluşturdum.

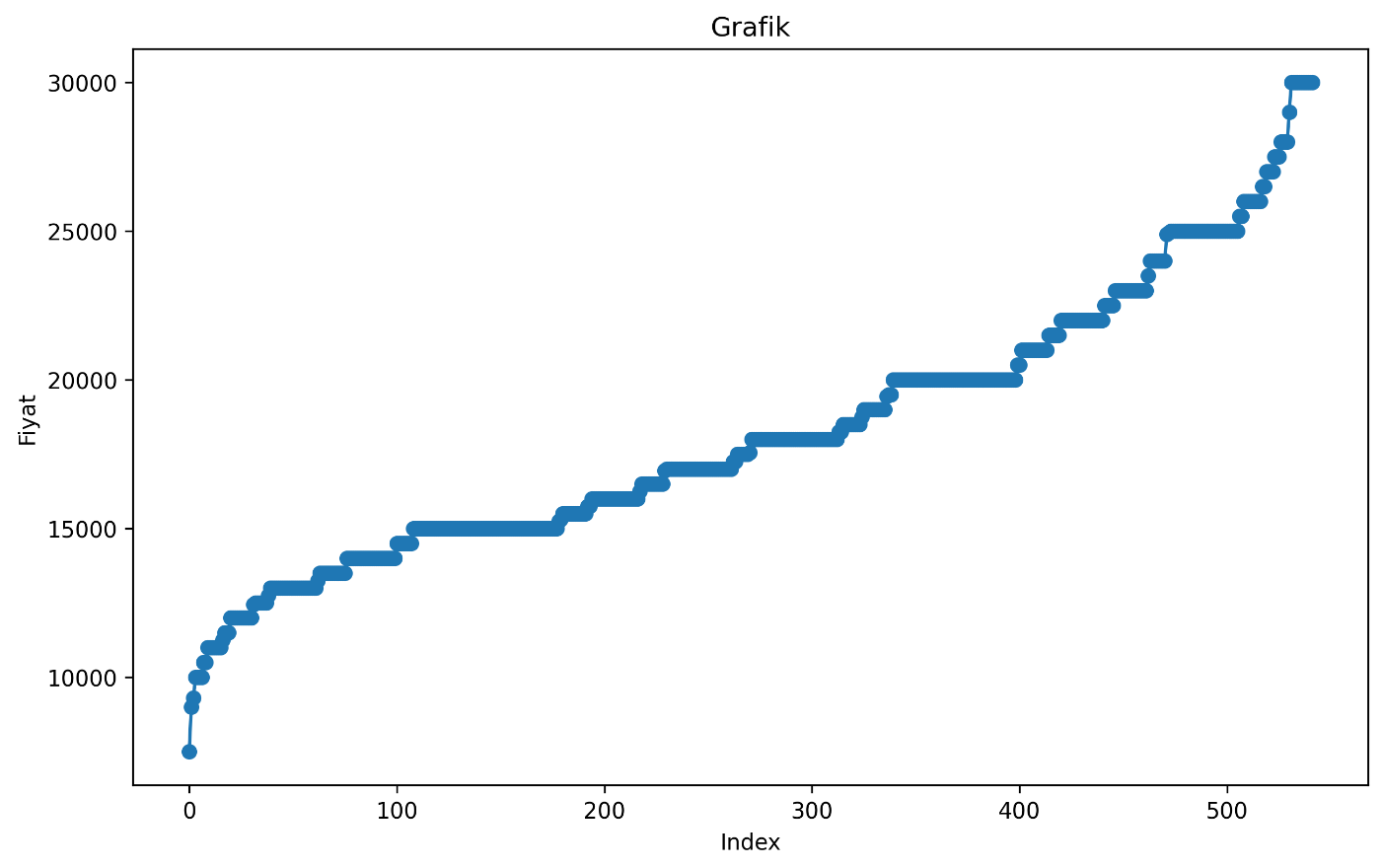


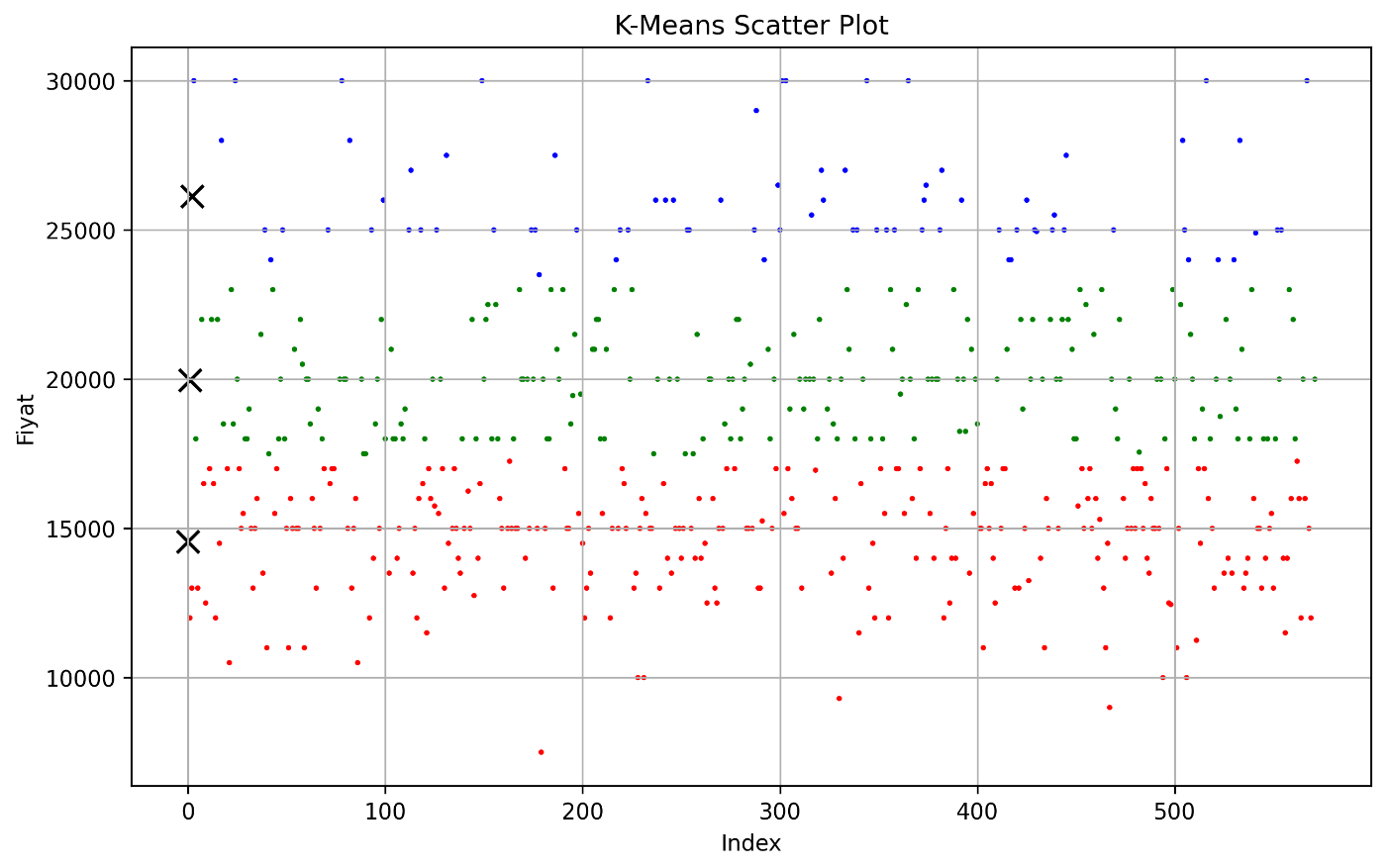


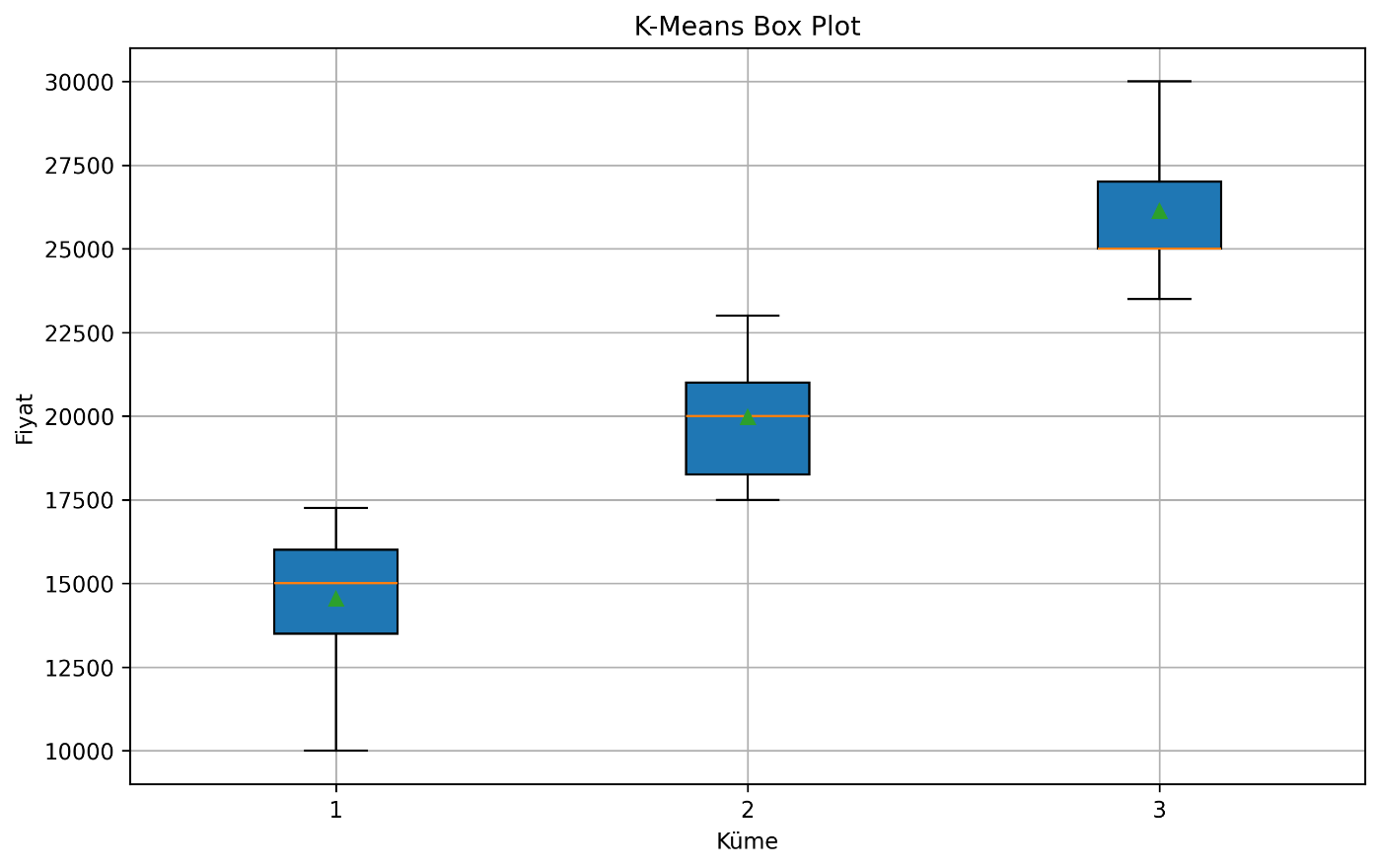


İlk grafiğin inişleri ve çıkışlarının x ekseninin zaman belirtmediği için anlamsız olduğunu fark ettim. Veriler sıralı olsaydı grafiğimiz çok daha düzgün görünürdü. Ayrıca scatter grafiğine bakınca fark ettim ki kümeleme algoritması indexlerin sıralı olmamasından ötürü kötü etkilenmiş.

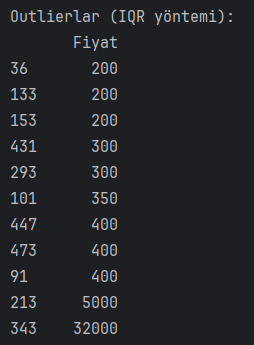
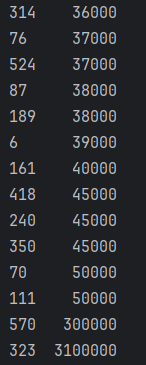
Verileri sıraladım ve tekrar grafik çıktıları aldım.



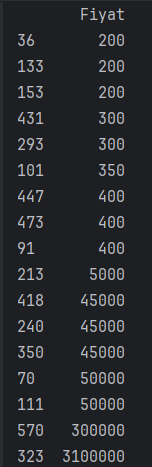


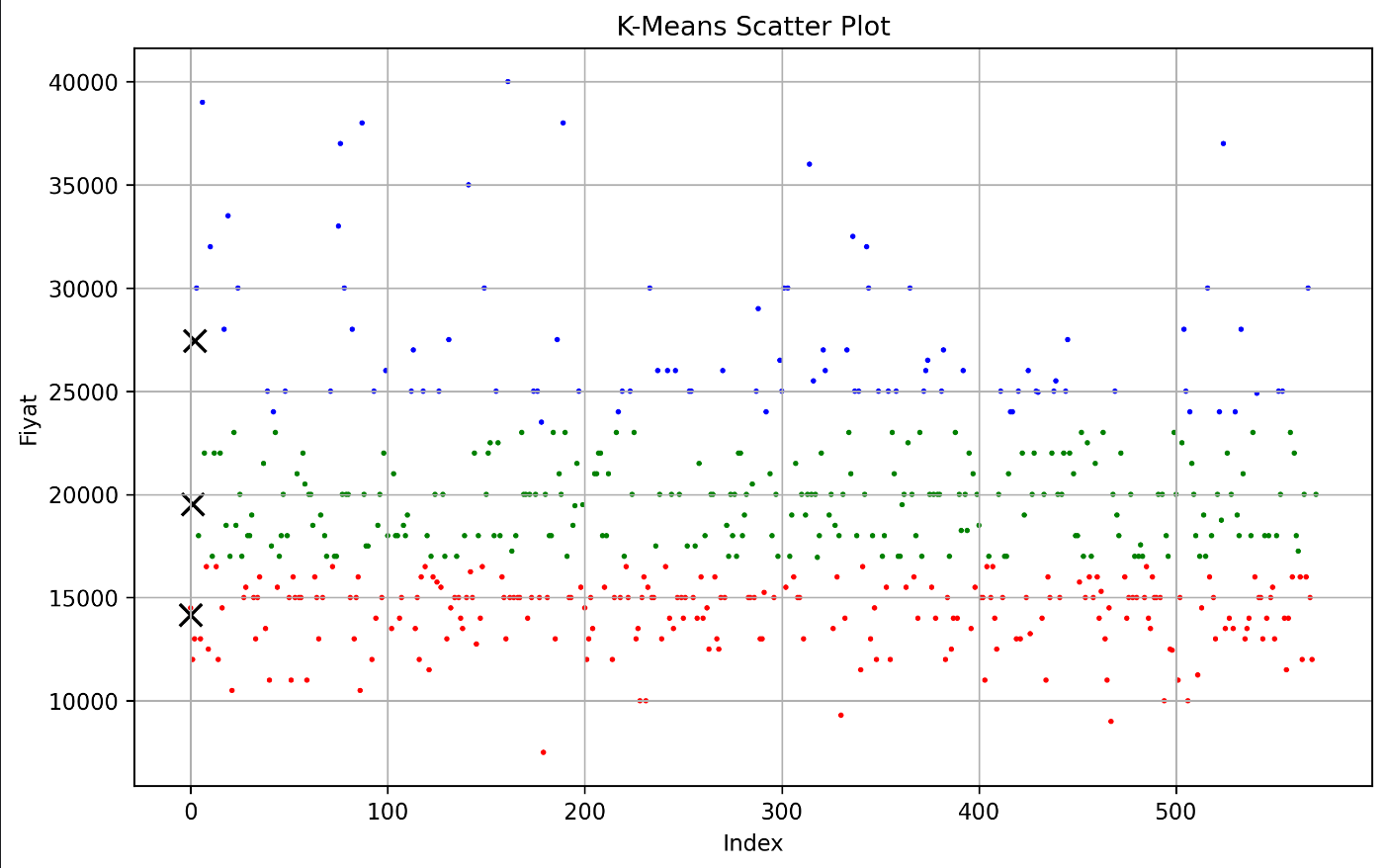


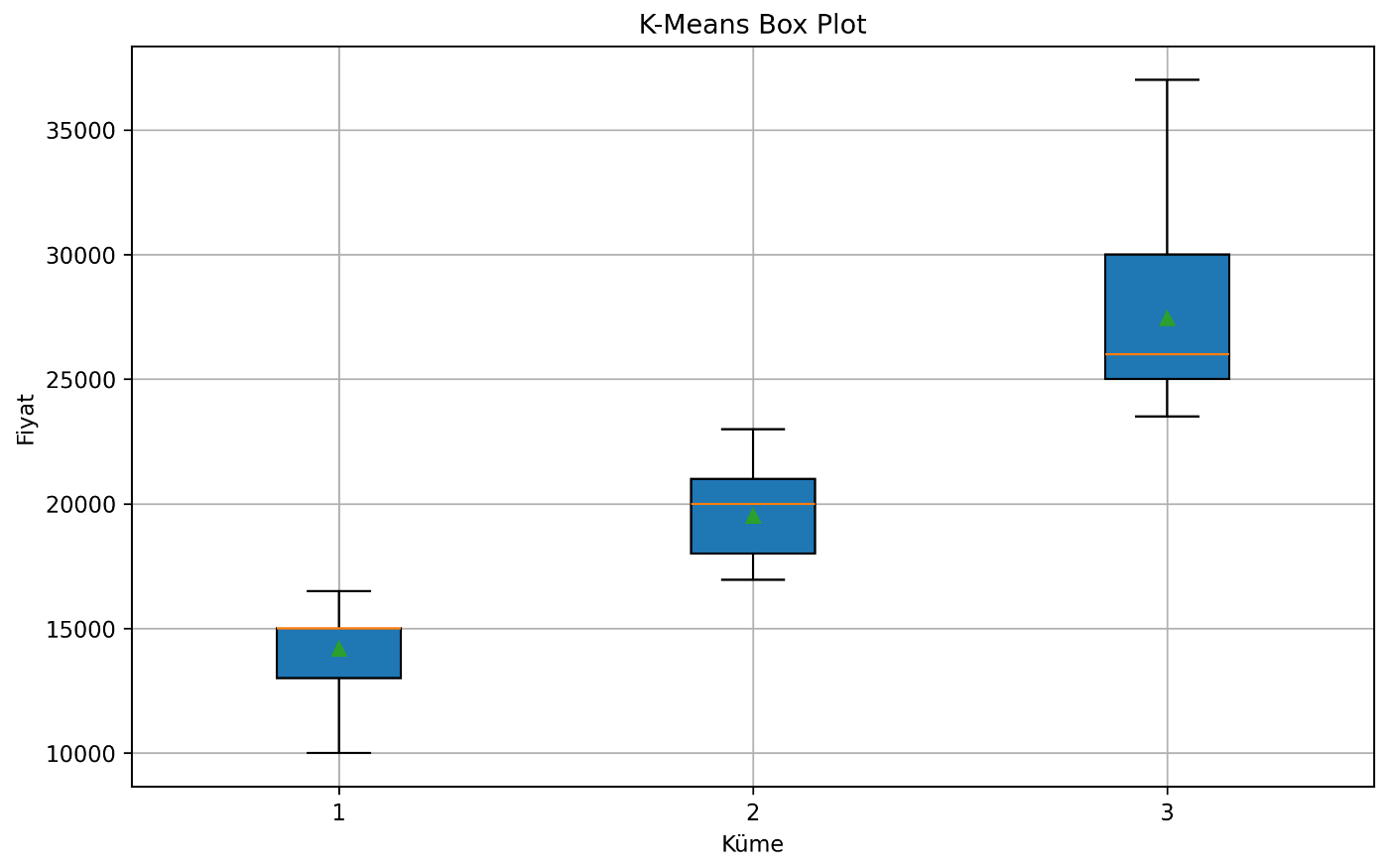
Outlierları grafiklerde belirtmem ekstrem değerlerden ötürü her şeyi inanılmaz küçültüyordu ve grafikleri bozuyordu. Bu yüzden outlierları grafikte göstermek yerine yazdırmaya karar verdim:

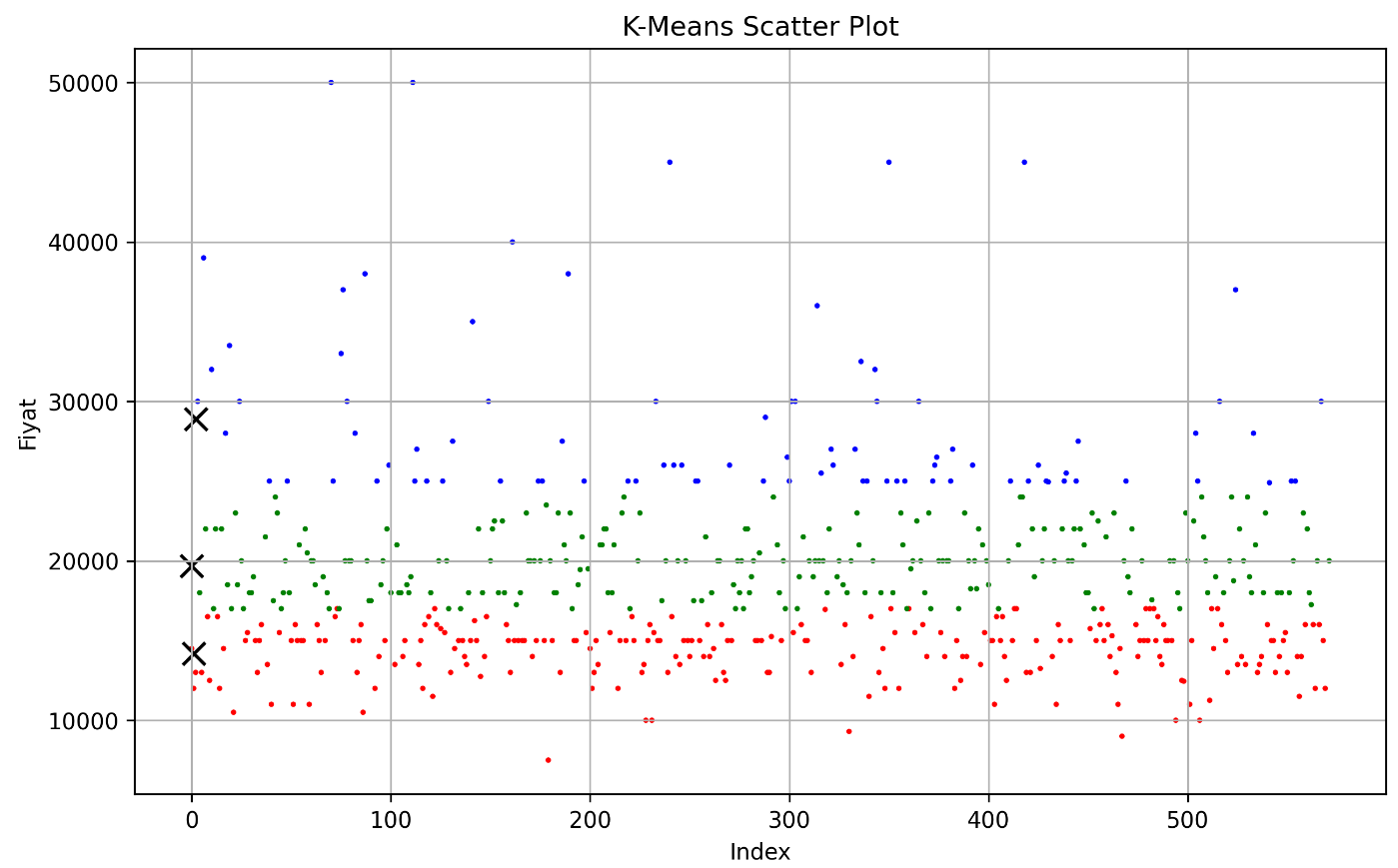
Soldaki sayılar indexleri sağdakiler ise fiyatı temsil ediyor. Buraya hepsini sığdıramadım ancak genel olarak şunu fark ettim: IQR yöntemi düşük değerleri iyi filtrelese de yüksek değerleri olması gerektiğinden daha fazla filtreliyordu. Günlük kiralık daireler ayrışmıştı ancak yüksek kiralı normal ilanlar da onlarla birlikte gitmişti. 32 bin ve 50 bin arasındaki değerler mantıklı değerler. Bu yüzden üst tabanın katsayısını 1.5’dan 3’e çıkardım. Outlierlar aşağıdaki gibi oldu ve yeni grafikleri oluşturdum.

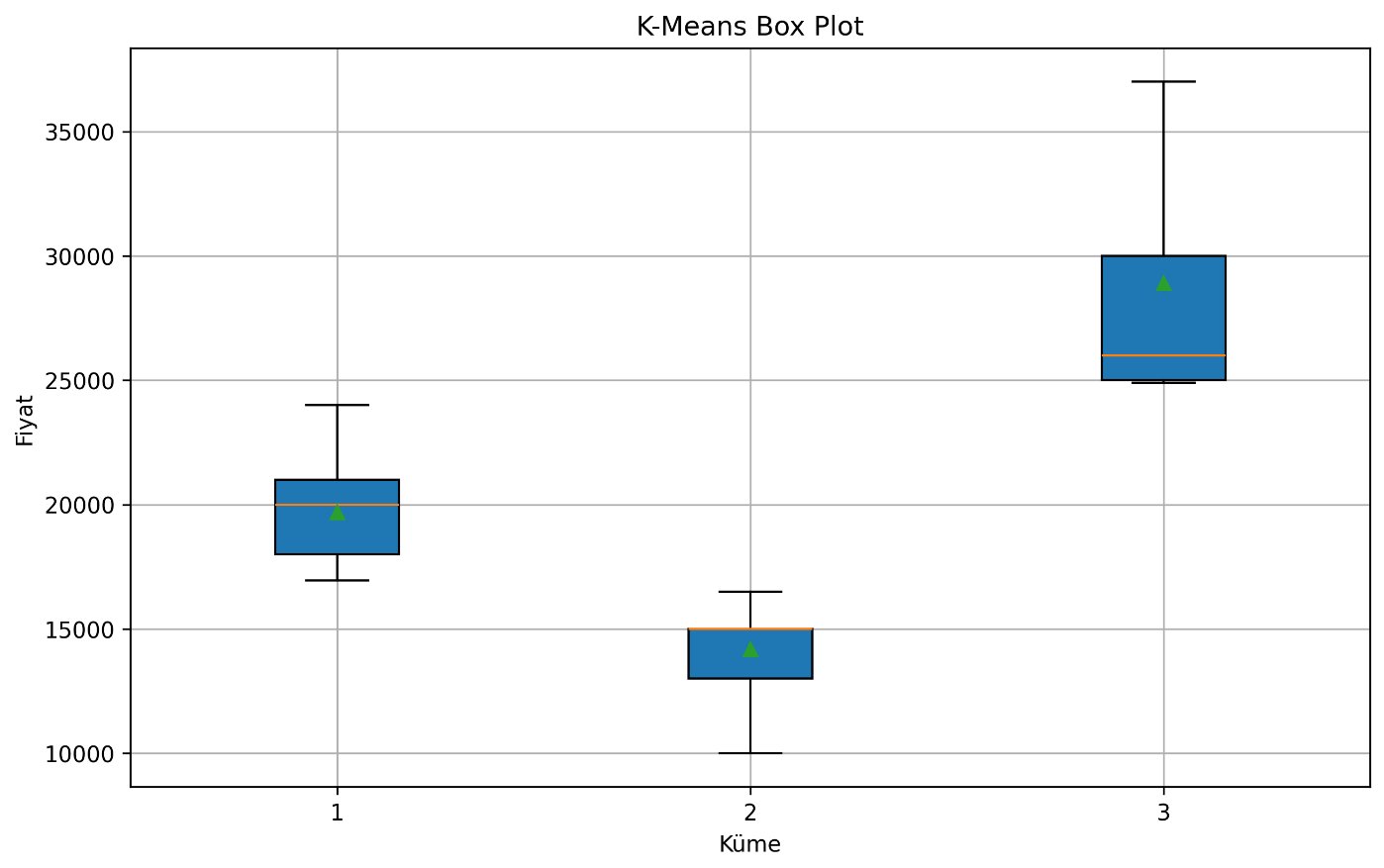






Outlier olarak seçilen 45 ve 50 bin liralık ilanları da veri setime katmak istedim ve bu yüzden üst tabanın katsayısını 6 yaparak bir daha grafikler oluşturdum.





Genel dağılımı biraz bozdukları için günün sonunda üst değer katsayısını 3 yapmaya karar verdim.

Sıralı grafikte verimin dağılımının normal olmadığını fark ettiğim için uygulamama z-score eklemedim ancak yine de deneyip görmek istedim o yüzden buraya ekliyorum:

# Fiyat sütununu z-score'a çevir

df["Z-Score"] = (df["Fiyat"] - df["Fiyat"].mean()) / df["Fiyat"].std()

z\_score\_modifier = 0.15

# Z-Score değeri z\_score\_modifier'dan büyük veya eksi z\_score\_modifier'dan küçük olan satırları seç

outliers = df[(df["Z-Score"] > z\_score\_modifier) | (df["Z-Score"] < -z\_score\_modifier)]

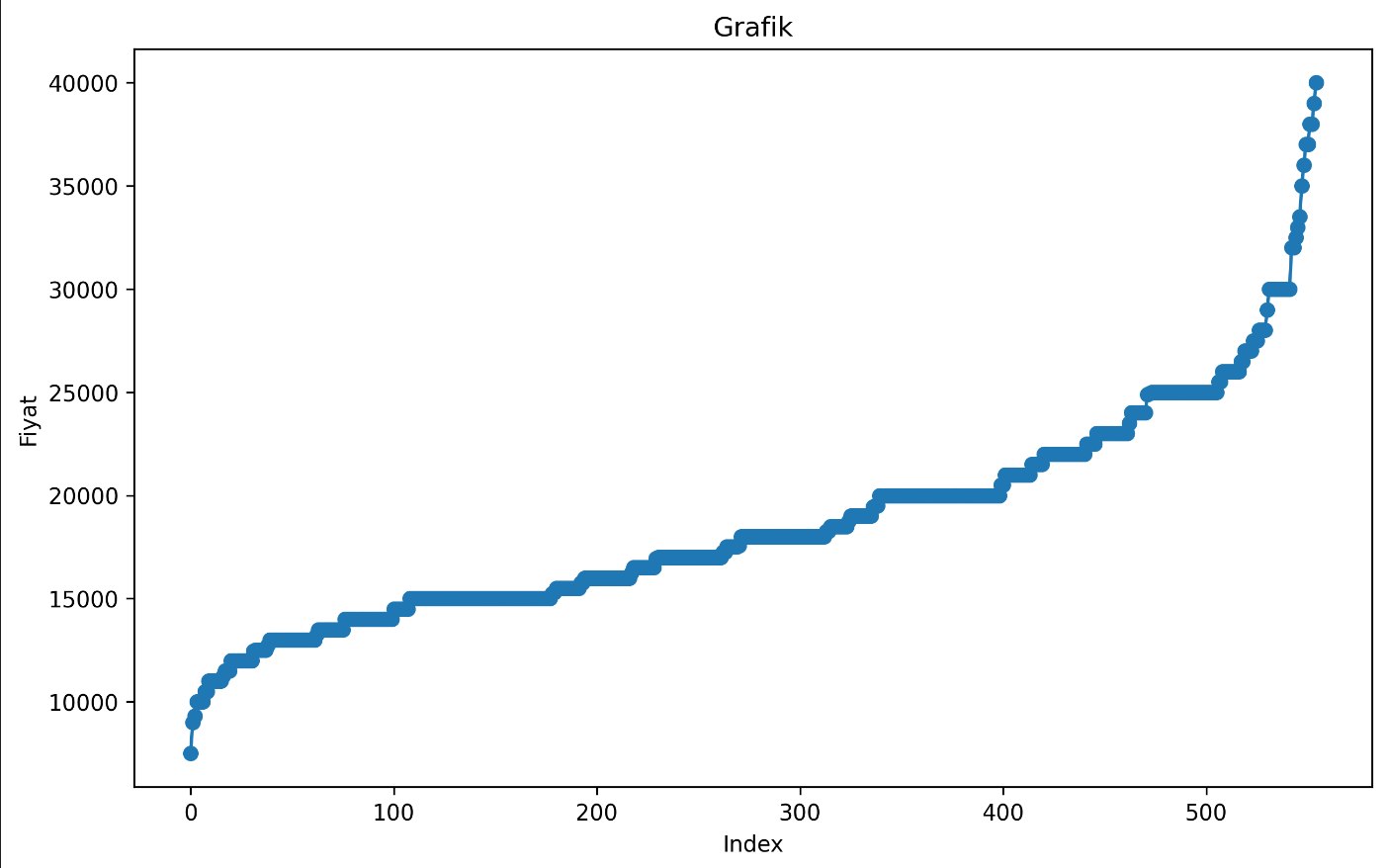
print("\nOutlierlar (Z-Score yöntemi):")

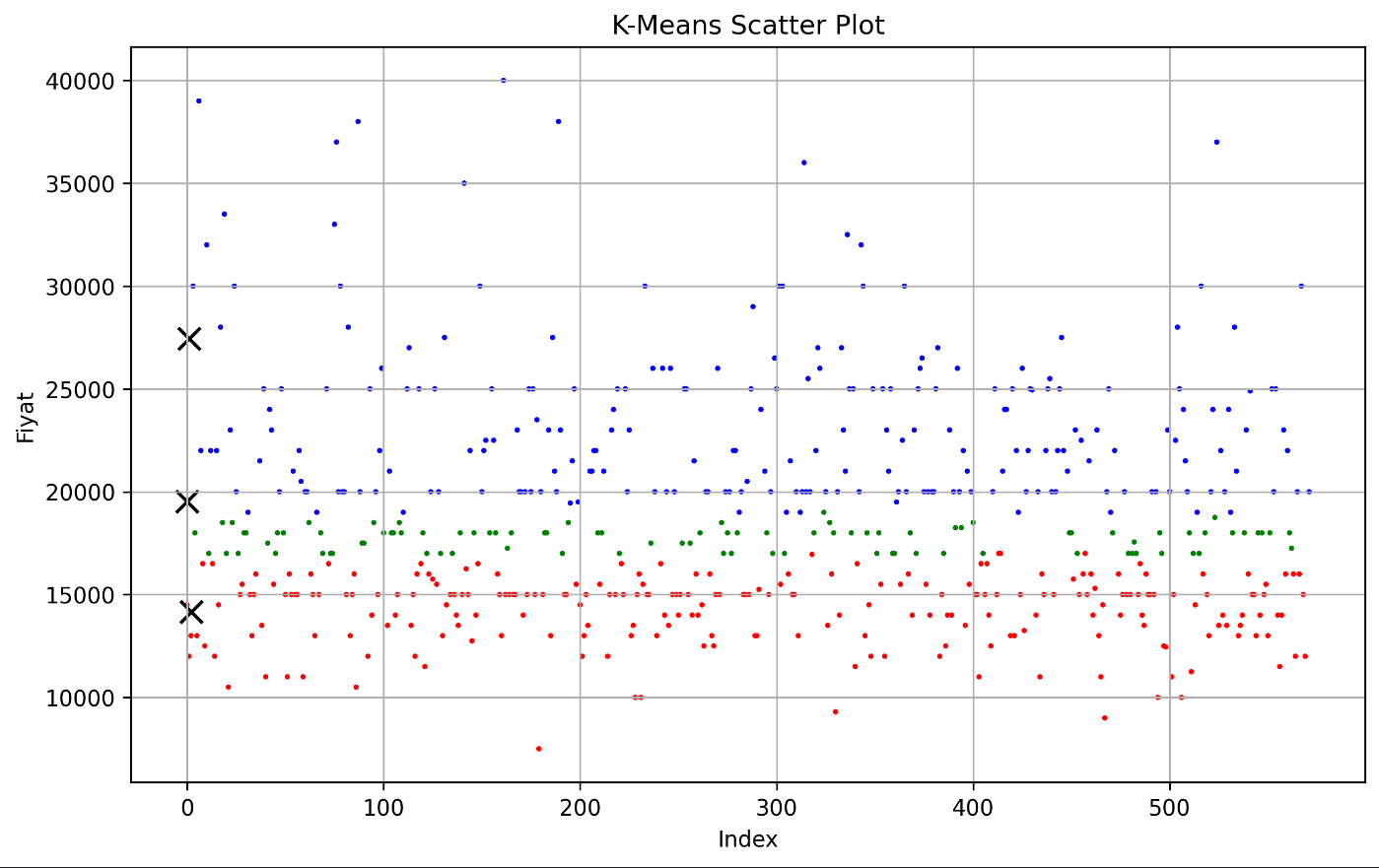
print(outliers[["Fiyat"]])

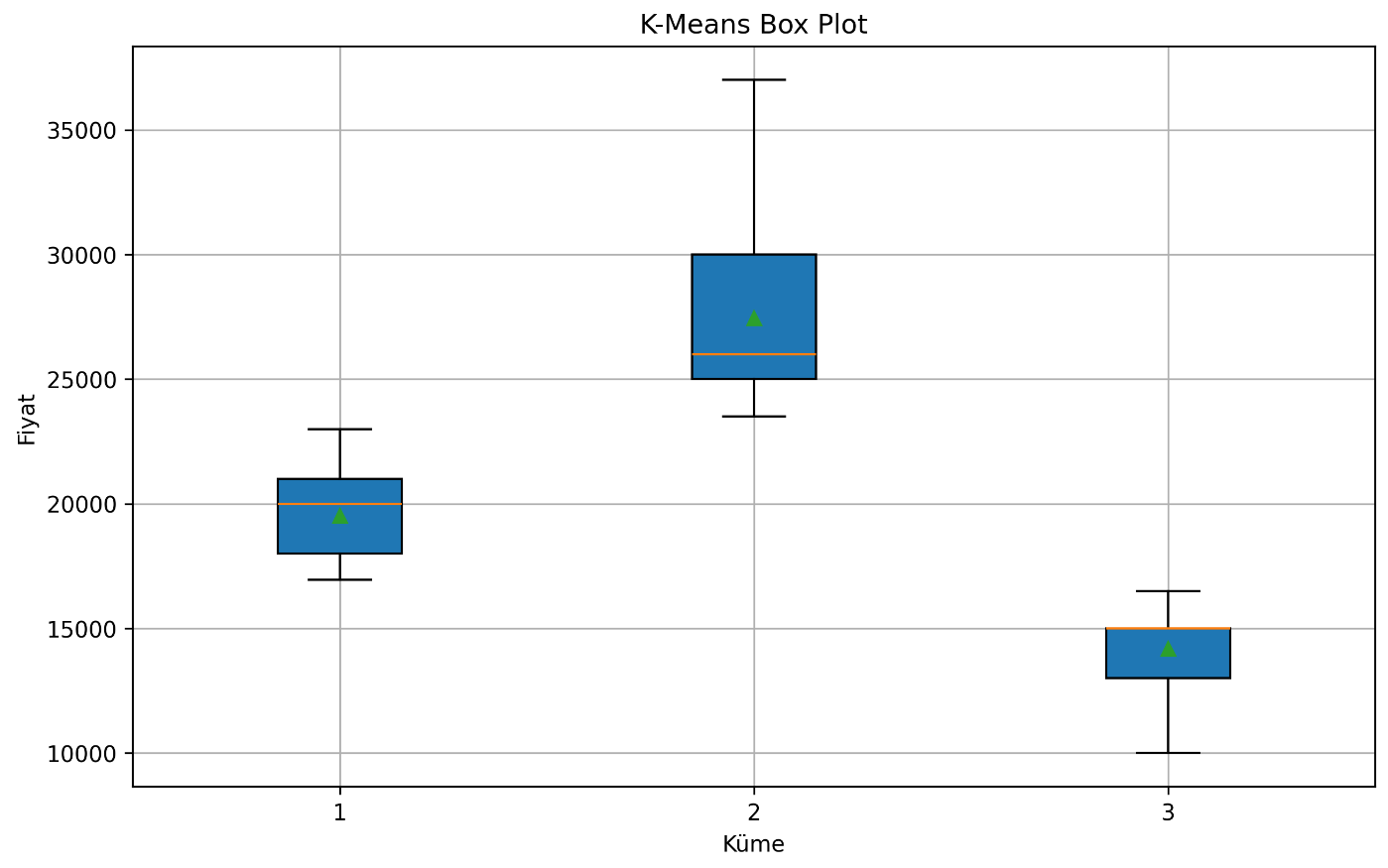
# Outlierları kaldır

df = df[(df["Z-Score"] <= z\_score\_modifier) & (df["Z-Score"] >= -z\_score\_modifier)]

z\_score\_modifier 0.15 olduğunda alt değer katsayısı 1.5 ve üst değer katsayısı 3 olan IQR yöntemi ile hemen hemen aynı değerler outlier olarak tespit edildi ve benzer grafikler elde ettim.







Farklı yol da aynı durumu ortaya çıkarınca uygun analizi yaptığımı düşünüp daha fazla deneme yapmayı bıraktım. Elbette random başlangıçtan ötürü her denememde kümeler biraz değişiyordu, bunu da birkaç deneme yapıp daha dengeli dağılım gözlemlediğim halinde durarak hallettim.

Benim için öğretici ve uğraşması keyifli bir ödevdi. İlk başta ham verileri elde etmek için modemimle defalarca uğraşmak biraz sinir bozucuydu ancak farklı yolları deneyip farklı grafikler yazdırmak, bunları değerlendirmek, parametrelerle uğraşıp yeni grafikler elde etmek güzeldi. Ayrıca k-means algoritmamı kendim yazdığım için de sevinçliyim. Kütüphaneler karmaşık syntax ve inanılmaz fazla sayıdaki parametreleri ile gözümü çok korkutmuştu.

### 5- Kaynaklar

Chat GPT

Ders Slaytları

<https://scikit-learn.org/1.5/modules/generated/sklearn.cluster.KMeans.html>

<https://www.w3schools.com/python/python_ml_k-means.asp>

<https://matplotlib.org/stable/index.html#matplotlib-release-documentation>

<https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.scatter.html>

<https://www.w3schools.com/python/matplotlib_scatter.asp>

<https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.boxplot.html>

<https://www.geeksforgeeks.org/box-plot-in-python-using-matplotlib/>

<https://regexr.com/>

<https://medium.com/analytics-vidhya/removing-outliers-understanding-how-and-what-behind-the-magic-18a78ab480ff>

<https://medium.com/@datasciencejourney100_83560/z-score-to-identify-and-remove-outliers-c17382a4a739>