



# Hacettepe Üniversitesi

**Mehmet Şenel**

**2210329077**

# İÇİNDEKİLER

---

İÇİNDEKİLER.....	1
ÖDEV HAKKINDA .....	3
BÖLÜM 1 .....	4
1.0 Bağımsız Veri .....	4
1.1 Hakkında .....	4
1.2 Verinin Yüklenmesi.....	4
1.3 Verinin Çıktısı .....	5
1.4 Verinin Temel Analizleri .....	5
1.5 Normallik Analizi .....	9
1.6 Normallik Grafikleri .....	10
1.7 Histogram Grafiği .....	11
1.8 Bar Grafiği .....	12
1.9 Çizgi Grafiği .....	13
2.0 Bağımsız Veri Nanparametrik Testler .....	15
2.1 Wilcoxon Sıra Sayı Testi.....	15
2.2 İşaret Testi .....	16
2.3 Mann-Whitney Testi .....	17
3.0 Kruskal-Wallis Testi .....	19
3.0 Bağımlı Veri.....	20
3.1 Veri Hakkında.....	20
3.2 Verinin Yüklenmesi .....	20
3.3 Verinin Çıktısı .....	21
3.4 Verinin Temel Analizleri .....	21
3.5 Normal Dağılım İncelemesi.....	26
3.6 Normallik Grafikleri .....	28
3.7 Histogram Grafiği .....	30
3.8 Line(Çizgi) Grafiği .....	31
3.0 Bağımlı Veri Nanparametrik Testler.....	33
3.1 Wilcoxon Sıra Sayı Testi.....	33

3.2 Friedman Testi .....	34
4.0 Eğilim Testi.....	36
4.1 Uygun Veri Tipi.....	36
4.2 Mann-Kendall Testi .....	36
BÖLÜM 2 .....	37
1.0 KÜTÜPHANELER HAKKINDA .....	37
2.0 PYTHON ÇIKTILARI(EK) .....	39
Kaynakça .....	50

# ÖDEV HAKKINDA

---

- ❖ Parametrik olmayan istatistiksel analizler ve bunların yorumlamaları gerçekleştirilecektir.
- ❖ İncelememiz üç aşamada gerçekleştirilecek olup;
  - ✓ Bağımlı özelliği taşıyan veriye ait analiz ve nanparametrik analizleri
  - ✓ Bağımsız özellik taşıyan bir veri grubunun analizi ve nanparametrik analizleri
  - ✓ 3. bir veri grubunun nanparametrik eğilim testinin gerçekleştirilmesi.
  
- ❖ Analiz ve incelemelerimizin tamamı Python aracılığı ile kodlama şeklinde gerçekleştirilecektir.
  - ✓ Python aracılığı verinin analizi
  - ✓ Grafik ve görselleştirme
  - ✓ Nanparametrik analizlerin gerçekleştirilmesi
  
- ❖ Bağımsız özellik taşıyan verimizin temel özellikleri, temel istatistik analizleri, normal dağılım incelenmesi, grafiksel incelemesi ve nanparametrik testlerin yapılması şeklinde devam edecektir.
  
- ❖ Bağımlı özellik taşıyan verimizin temel özellikleri, temel istatistik analizleri, normal dağılım incelenmesi, grafiksel incelemesi ve nanparametrik testlerin yapılması şeklinde devam edecektir.
  
- ❖ Ve son olarak 3. Bir veri grubumuzun nanparametrik eğilim testinin Python aracılığı ile gerçekleştirilip çıktısının yorumlanması ile son bulacaktır.

# BÖLÜM 1

---

## 1.0 Bağımsız Veri

### 1.1 Hakkında

- ❖ Verimiz Televizyon kanallarının tanıtım için ayırdıkları düşük, orta veya yüksek bütçelerini ifade etmekte olup, bu bütçelerin sosyal medya tanıtım bütçelerine bir katkısı varımdır bunun araştırılması ve incelenmesi sağlanacaktır.
- ❖ Verimiz 3 sütun ve 15 satırdan oluşmaktadır.
  - ✓ Sütunlar düşük, orta ve yüksek bütçeleri temsil etmektedir.
  - ✓ Satırlar ise verilerin aldığı değerleri temsil etmektedir.
- ❖ Verimiz kaggle üzerinden bir veri analizi içerisinde örneklem şeklinde çekilmiştir.
- ❖ Verimiz Excel formatından Python içerisine analiz için aktarılmıştır.
- ❖ Verimizin öncelikle olarak temel istatistik analizlerini;
  - ✓ Ortalama, mod, medyan vb. temel istatistik analizleri.
  - ✓ Basıklık ve çarpıklık katsayıları
  - ✓ Normallik testleri yapılacaktır.
- ❖ Ardından verinin uygunluğuna bağlı olacak şekilde grafikleri ve yorumlanması,
- ❖ Son olarak verimizin normallik durumuna bağlı olacak şekilde bazı nanparametrik testler uygulanacaktır.

### 1.2 Verinin Yüklmesi

- ❖ Verimizin analizi Python uygulaması üzerinde yapılacak olup verimizi excel formatı üzerinden kod sistemi ile uygulamaya aktarma işlemi gerçekleştirilir.

```
bagimsiz = pd.read_excel(r'C:\Users\pc\Desktop\nanpar veriler\75BB7C40.xlsx')
```

7] ✓ 0.0s

- ❖ Görsele de görüldüğü üzere pandas kütüphanesinden read\_excel kodu aracılığı dosyamızın bilgisardaki konumundan otomatik olarak uygulamaya aktarımını gerçekleştirmekteyiz.

### 1.3 Verinin Çıktısı

- ❖ Verimizin belirtilen kod aracılığı ile uygulamaya çekilmesinin ardından uygulama üzerinde oluşan çıktısı aşağıdaki gibidir.

...		low	medium	high
	0	2.57	1.82	6.55
	1	3.70	5.62	4.96
	2	1.80	2.15	3.89
	3	2.45	3.44	3.31
	4	3.21	6.36	4.32
	5	1.54	2.49	7.72
	6	3.07	4.18	3.51
	7	8.00	3.09	3.31
	8	2.40	2.12	4.92
	9	1.38	4.02	4.60
	10	2.57	1.75	3.61
	11	5.82	2.89	4.06
	12	3.88	9.07	3.79
	13	7.16	3.15	3.33
	14	1.20	2.36	5.52

❖ Veri hakkında kısmında bahsettiğimiz gibi görmekteyiz ki verimiz 3 sütun ve 15 satırdan oluşmaktadır.

❖ Verimize genel anlamda göz gezdirdiğimizde verimiz milyon dolar bazında oluşturulmuş ve virgülden sonraki iki hanesi yer alacak şekilde oluşturulmuştur.

❖ Verilerimiz 0-10 milyon dolar arasında şekillenmiş olup toplamda 45 adet gözlemimiz yer almaktadır.

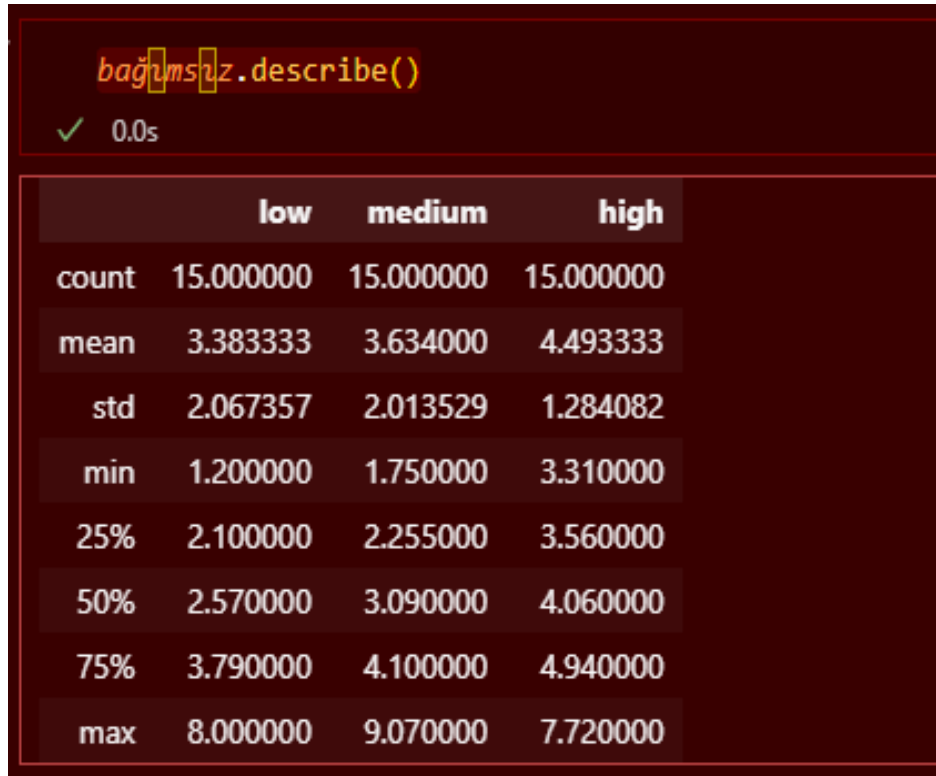
❖ Veri tipimiz birbirinden hem kategorik olarak hem de kendi içlerinde birbirinden bağımsızdır.

### 1.4 Verinin Temel Analizleri

- ❖ Bu bölümde yine Python uygulaması aracılığı ile verimize ait temel istatistiklerin analizini ve yorumlamasını gerçekleştirecek olup özellikle veri üzerinde uygulayacağımız normallik testi nanparametrik testlerin uygun olup olmayacağı hakkında bizlere bilgi verecektir.
- ❖ Analizlerin bazı aşamalarında bu analizlerin aynı zamanda grafiksel gösterimlerine de yer verilecektir.

### 1.4.1 Tanımlayıcı istatistikler

- ❖ Bu bölümde tanımlayıcı istatistikler adını verdiğimiz temel istatistiklerin Python üzerindeki çıktılarından incelemeler gerçekleştirilecektir.

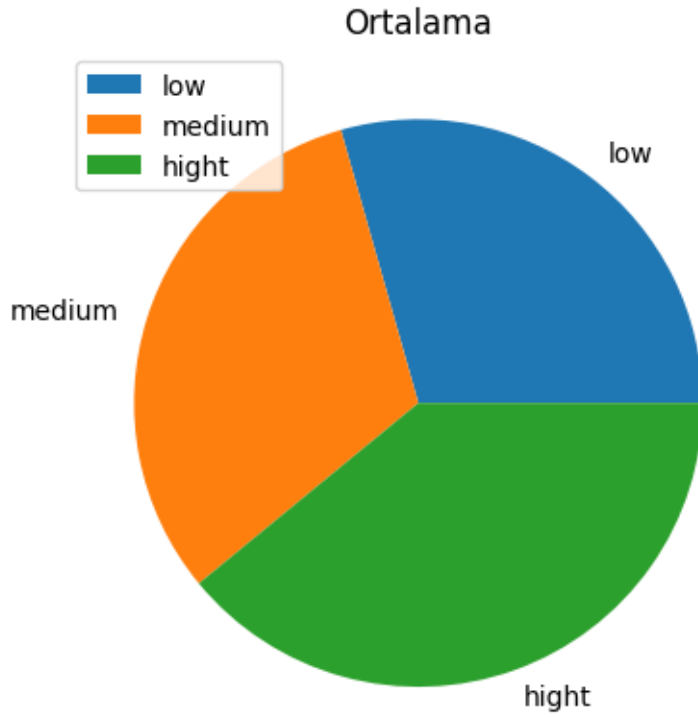


```
bagimsiz.describe()
```

✓ 0.0s

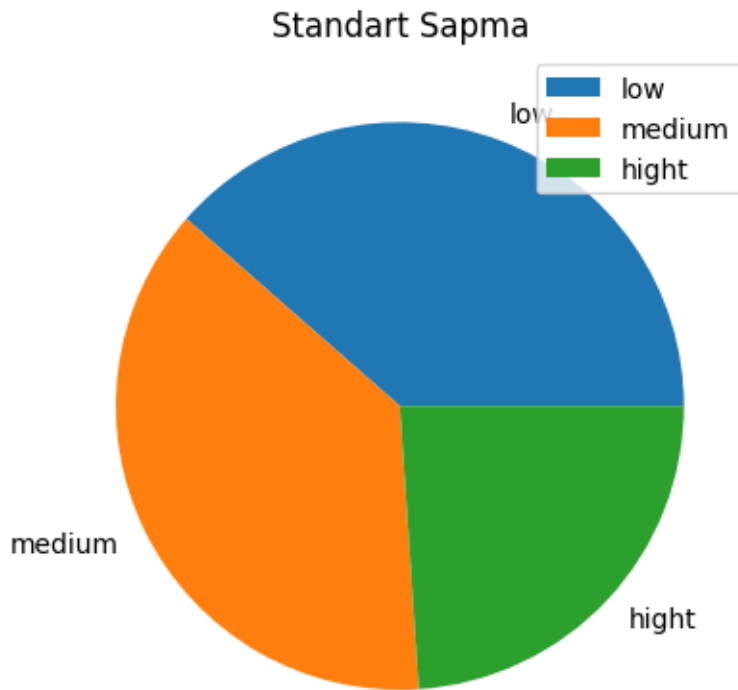
	low	medium	high
count	15.000000	15.000000	15.000000
mean	3.383333	3.634000	4.493333
std	2.067357	2.013529	1.284082
min	1.200000	1.750000	3.310000
25%	2.100000	2.255000	3.560000
50%	2.570000	3.090000	4.060000
75%	3.790000	4.100000	4.940000
max	8.000000	9.070000	7.720000

- ❖ Öncelikli olarak Python üzerinden “ . Describe () ” fonksiyonu aracılığı ile temel istatistiklere ulaşım sağlarız.
- ❖ Yukarıdaki görselde de görüldüğü üzere her üç değişkenimiz içinde ortalama, mod, medyan, standart sapma gibi analizler yer almaktadır.
- ❖ Verilen temel istatistiklere göz gezdirdiğimizde direkt olarak yüksek bütçe ayıran TV kanallarının ayırdıkları bütçelerin ortalamasının diğerlerine nazaran yüksek olmasına rağmen en düşük standart sapmaya sahip olduğu gözümüze çarpmaktadır.
- ❖ Bu durumu daha yakından incelediğimizde görüyoruz ki yüksek bütçe ayıran TV kanallarının diğer kanal bütçelerine nazaran dağılımının daha yüksek sayılarda ve daha yüksek olduğundan kaynaklandığını görmekteyiz.
- ❖ Bunun yanı sıra veri dağılımı olarak düşük ve orta bütçe harcaması yapan TV kanallarının birbirine dağılımsal olarak daha yakın olduğunu görmekteyiz.



❖ Örnek teşkil edebilmesi açısından grupların ortalamasının pasta grafiği üzerinde kapladığı alanlara yer verilmiştir.

❖ Görselden de görüleceği üzere pasta grafiği üzerinde verilerin kapladığı ortalamalar  $high > Medium > low$  şeklinde sıralanmaktadır.



❖ Dağılımın kafamızda şekillenebilmesi açısından grupların ortalamasının ardından özellikle standart sapmalarının dağılımını görmek adına standart sapmalara ait pasta grafiğine yer verilmiştir.

❖ Görüldüğü üzere standart sapmalarda ise durum ortalamanın tam tersi olmakla birlikte sıralama  $low > medium > high$  şeklinde yer almaktadır.

- ❖ İki grafiği karşılaştırdığımızda net bir şekilde görmekteyiz ki low ve medium verileri ortalamanın da standart sapmaya yakın olmasından anlaşılabileceği üzere dağılım olarak birbirine yakındır.
- ❖ High verisi ise verilerin dağılımından kaynaklı daha düzensiz bir veri türüdür.



### 1.4.2 Çarpıklık Katsayısı

- ❖ Normallik testinden hemen önce veriye ve verinin dağılımına dair fikir sahibi olabilmek için çarpıklık katsayısının incelemesini gerçekleştireceğiz.
- ❖ Çarpıklık katsayısı -1 ile +1 arasında değişiyor olmasını ve + değer alıyor ise sola çarpık, - değer alıyor ise dağılımın sağa çarpık olduğunu anlayabiliriz.
- ❖ Eğer değerimiz -1 ile +1 dışına çıkıyor ise normal dağılımdan uzak olduğunu anlayabiliriz.

```
skew(bagımsız, axis=0, bias=True)
✓ 0.0s
array([1.13240532, 1.49703124, 1.27016861])
```

- ❖ Yukarıda görüldüğü üzere çarpıklık katsayılarımız sırası ile 1'in üzerinde olmakla beraber veri dağılımımızın sola çarpık ve normallikten de uzak olduğunu söyleyebiliriz.

### 1.4.3 Basıklık Katsayısı

- ❖ Basıklık katsayısı da yine aynı şekilde 0'a ne kadar yakınsa normalliği o kadar sağlamakla beraber -1 ile +1 arasında değer almaktadır.
- ❖ Eğer basıklık değerimiz pozitif ise sivri olduğunu, negatif ise basık olduğunu ifade etmektedir.

```
kurtosis(bagımsız, axis=0, bias=True)
✓ 0.0s
array([0.15172462, 1.57407675, 0.80549145])
```

- ❖ Yukarıda görselde görüldüğü üzere basıklık değerinin hesabı için uygulama içerisinde kurtosis fonksiyonundan faydalanılmıştır.
- ❖ Oluşan sonuçları incelediğimizde tamamının pozitif olduğunu yani sivri bir dağılıma sahip olduğunu görmekteyiz.
- ❖ İlk veri grubumuz olan düşük bütçe ayıran TV kanallarının basıklık katsayısının diğerlerine nazaran 0'a daha yakın olduğunu görmekteyiz.

## 1.5 Normallik Analizi

- ❖ Verimizin dağılımı ve temel istatistikleri hakkında az çok bilgi sahibi olduktan sonra artık direkt olarak normallik testi üzerinden dağılımın normal olup olmadığının analizini gerçekleştireceğiz.
- ❖ Verimizin normal dağılıp dağılmadığını anlayabilmek adına  $n < 20$  olduğundan dolayı Shapiro-Wilk Testi ile verimizin normallliğini test edeceğiz.

### Hipotez:

$H_0$  = Veri setinin dağılımı normaldir.

$H_1$  = Veri setinin dağılımı normal değildir

### 1.5.1 Düşük Bütçeli

```
lw = baglmslz.low
bg1, bg2 = stats.shapiro(lw)
print("Test istatistiği : ", bg1)
print("p-value : ", bg2)
```

92] ✓ 0.0s

Test istatistiği : 0.8509675860404968  
p-value : 0.017922883853316307

- ❖ Kodumuz görüldüğü üzere stats kütüphanesi aracılığı ile ve shapiro fonksiyonu bile basit bir şekilde çalışmaktadır.
- ❖ Yukarıda görüldüğü üzere test istatistiği sonucu 0.85 , p-value değeri ise 0.01 çıkmaktadır.
- ❖ P-value < 0.05 olduğundan hipotezimiz rededilmektedir. Buda verimizin normal dağılmadığının ispatıdır.

### 1.5.2 Orta Bütçeli

```
md = baglmslz.medium
bg1, bg2 = stats.shapiro(md)
print("Test istatistiği : ", bg1)
print("p-value : ", bg2)
```

93] ✓ 0.0s

Test istatistiği : 0.8253953456878662  
p-value : 0.00793540757149458

- ❖ Kodumuz görüldüğü üzere stats kütüphanesi aracılığı ile ve shapiro fonksiyonu bile basit bir şekilde çalışmaktadır.
- ❖ Test istatistiği 0.82 , p-value değeri 0.007 çıkmaktadır.
- ❖ P-value < 0.05 olduğundan hipotezimiz reddedilmiştir. Verimiz normal dağılıma uygun değildir.

### 1.5.3 Yüksek Bütçeli

```
> hg = bagimsiz.high
bg1,bg2 = stats.shapiro(hg)
print("Test istatistiği : ",bg1)
print("p-value : ", bg2)

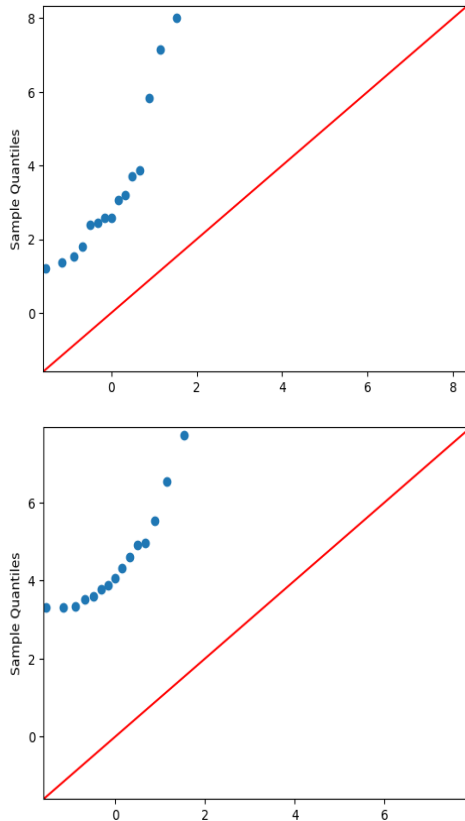
[8] ✓ 0.0s

.. Test istatistiği : 0.8495509028434753
p-value : 0.01711362786591053
```

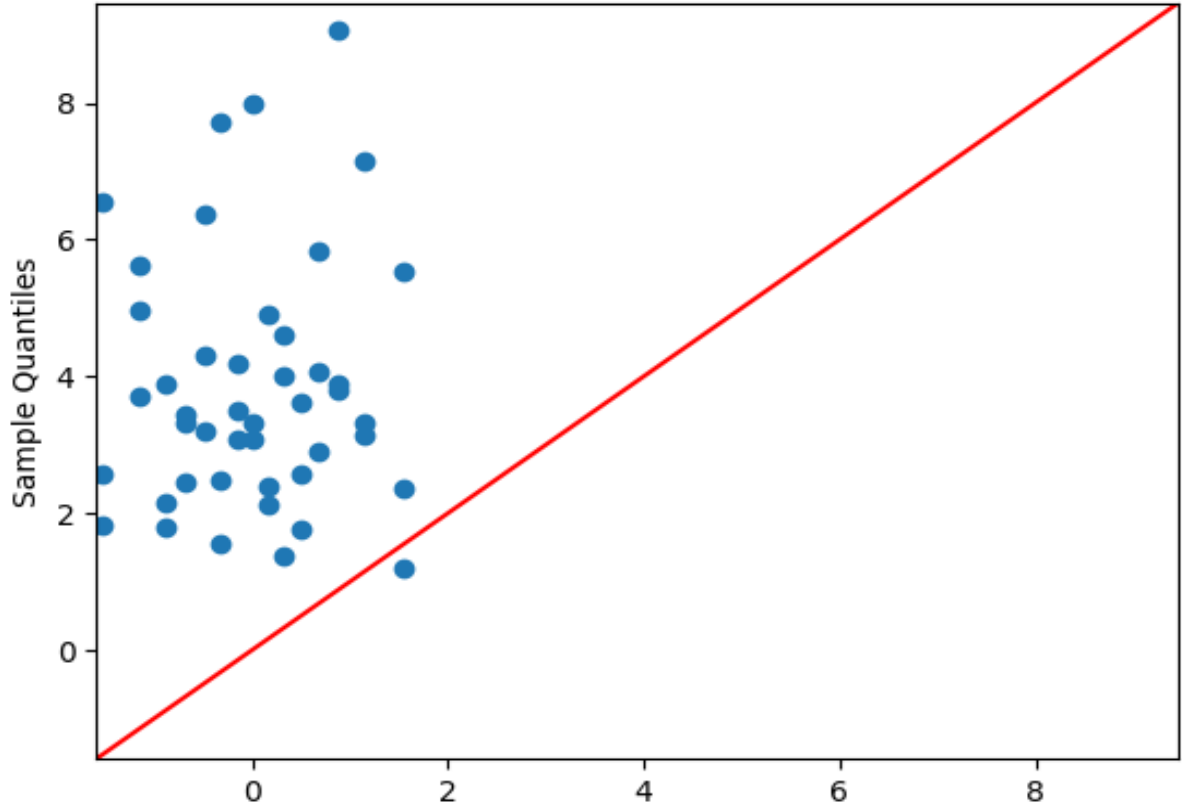
- ❖ Kodumuz görüldüğü üzere stats kütüphanesi aracılığı ile ve shapiro fonksiyonu bile basit bir şekilde çalışmaktadır.
- ❖ Test istatistiği 0.84 , p-value değeri 0.01 çıkmaktadır.
- ❖ P-value < 0.05 olduğundan hipotezimiz reddedilmiştir. Verimiz normal dağılıma uygun değildir.

### 1.6 Normallik Grafikleri

- ❖ Bir önceki madde de yer alan normallik testlerinin ardından durumu grafik üzerinde görmek adına ,tek tek ve bütün halinde incelemeler sağlayacağız.



- ❖ Sırasıyla low,medium ve high gruplarına ait Q-Q plot grafiğine yer verilmiştir.
- ❖ Q-Q plot grafiğinin temel incelenme koşulu verilerin yer alan çizginin etrafında dağılıyor olmasıdır. Bu durum dışında bir dağılım söz konusu ise verimizin normal dağılıma sahip olmadığını görmekte ve anlamaktayız.
- ❖ Her üç grubumuzda oldukça uzak dağılım göstermiş olup normallikten uzaktır.



- ❖ Yukarı da yer alan grafiğimiz tüm veri gruplarını tek bir grafik altında toplayacak şekilde oluşturulmuş bir Q-Q plot grafiğidir.
- ❖ Grafiğimiz içerisinde 3 veri grubumuzda yer almakta olup bir önceki grafikte tüm verilerimizi ayrı ayrı incelediğimizde her üçünde normallikten uzak olduğunu görmüştük.
- ❖ Her üç grubumuza barındıran bu grafiğimizi incelediğimizde yine aynı şekilde normallikten uzak olduğunu ve normal bir dağılıma sahip olmadığını söyleyebiliriz.

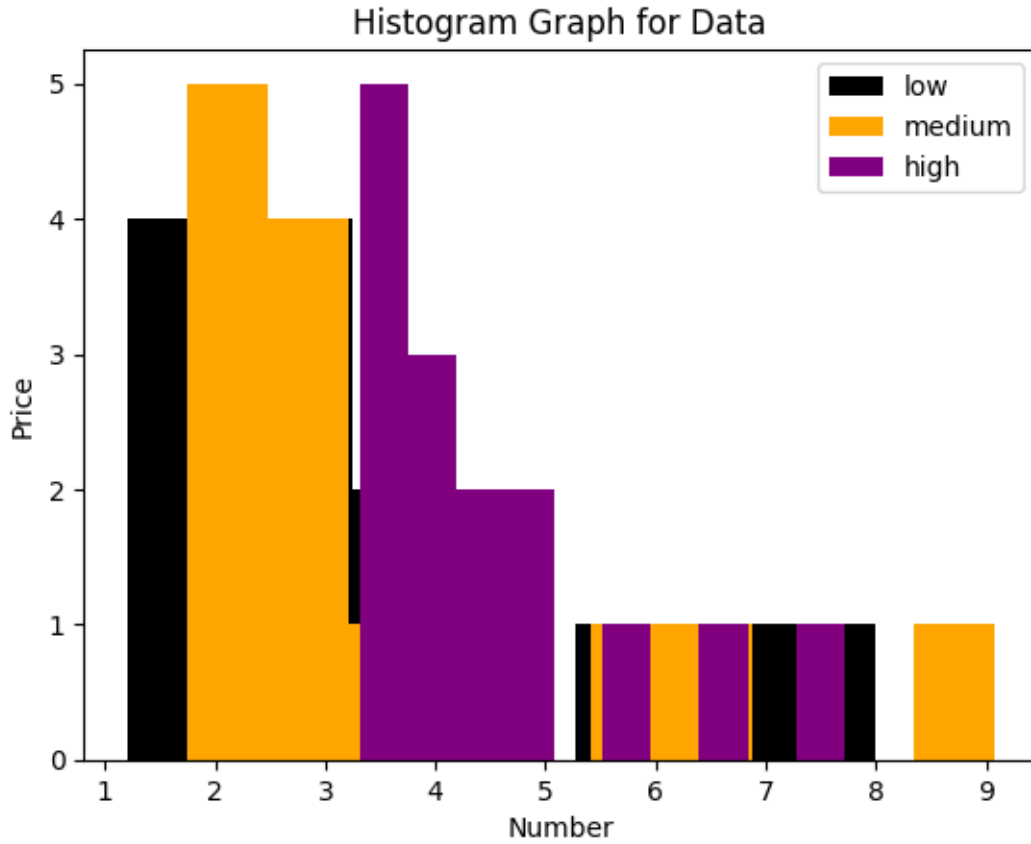
## 1.7 Histogram Grafiği

```
plt.hist(bagunsuz.low, color="black", label='low')
plt.hist(bagunsuz.medium, color="orange", label='medium')
plt.hist(bagunsuz.high, color="purple", label='high')
plt.legend()
plt.title('Histogram Graph for Data')
plt.xlabel('Number')
plt.ylabel('Price')
plt.show()
```

21 ✓ 0.2s

- ❖ Grafiğimizi matplotlib kütüphanesi olarak bilinen kütüphane yardımı ile hist fonksiyonu aracılığı ile Python da çizdirmiş bulunmaktayız.

- ❖ Grafiğimiz tüm veri gruplarının histogram grafiği üzerinde gösterimini baz alacak şekilde oluşturulmuştur.



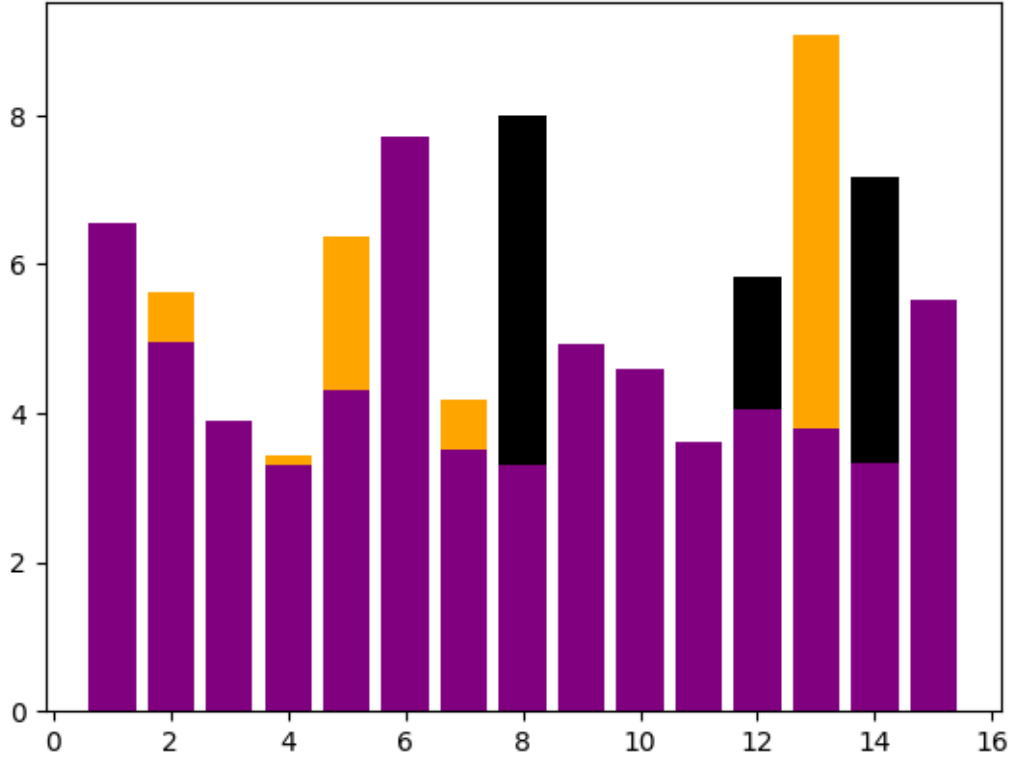
- ❖ Yukarıda görüldüğü üzere tüm veri grupları ve değerleri tek bir grafik içerisinde histogram olarak verilmiştir
- ❖ Grafiğimizi incelediğimizde grafiğimiz yaptığımız hesaplamalarla uyumlu olacak şekilde pozitif çarpık şekilde ve sivri bir yapıya sahiptir.
- ❖ Grafiğimiz pozitif çarpık ve sivri olması kaynaklı normal dağılımdan oldukça uzaktır.

## 1.8 Bar Grafiği

```
Number = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
plt.bar(Number,bagimsiz.low, color = "black")
plt.bar(Number,bagimsiz.medium, color = "orange")
plt.bar(Number,bagimsiz.high, color = "purple")
plt.show()
```

1 ✓ 0.1s

- ❖ Grafiğimizi matplotlib kütüphanesi olarak bilinen kütüphane yardımı ile bar fonksiyonu aracılığı ile Python da çizdirmiş bulunmaktayız



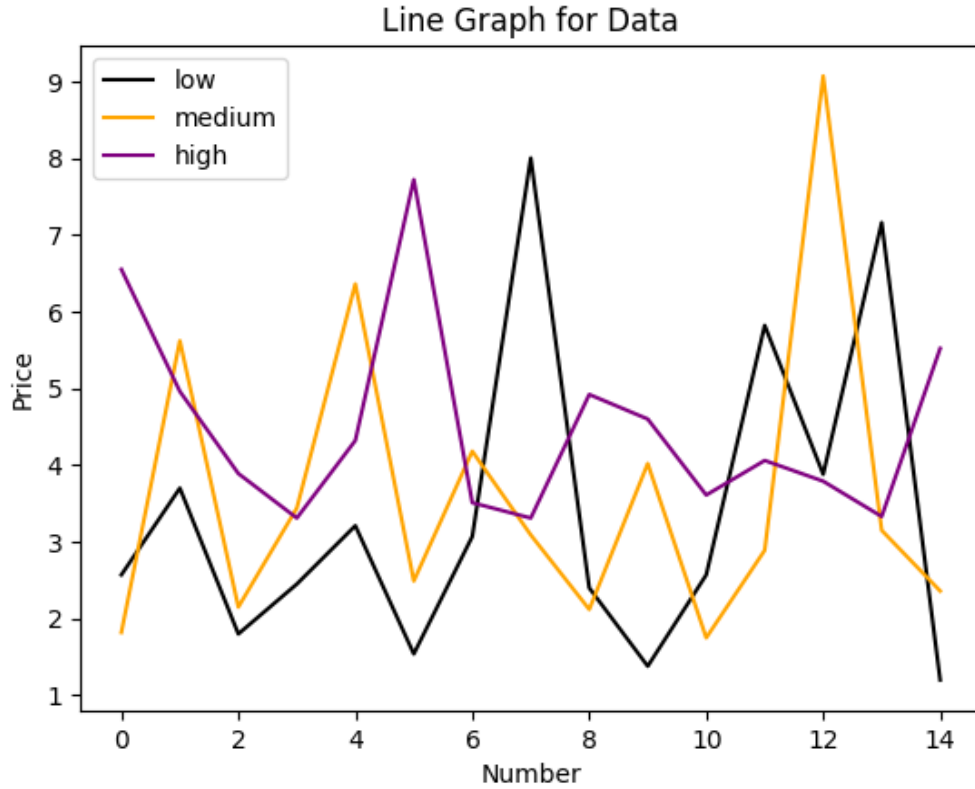
- ❖ Grafiğimiz tüm grupları içerisinde bulunduracak şekilde tasarlanmış ve oluşturulmuştur.
- ❖ Bu bar grafiği ile temel amacımız ve hedefimiz veri grubu içerisinde yer alan veri sayısının eşit olması nedeniyle ve aynı zamanda yapılan işin bir harcamayı temsil etmesi sebebiyle veri grupları içerisinde satır bazında bir karşılaştırma yapılmak istenmiştir.
- ❖ Görüldüğü üzere genel olarak en tepe değer verileri high adını verdiğimiz yüksek bütçe harcaması yapan TV kanallarına aittir.

## 1.9 Çizgi Grafiği

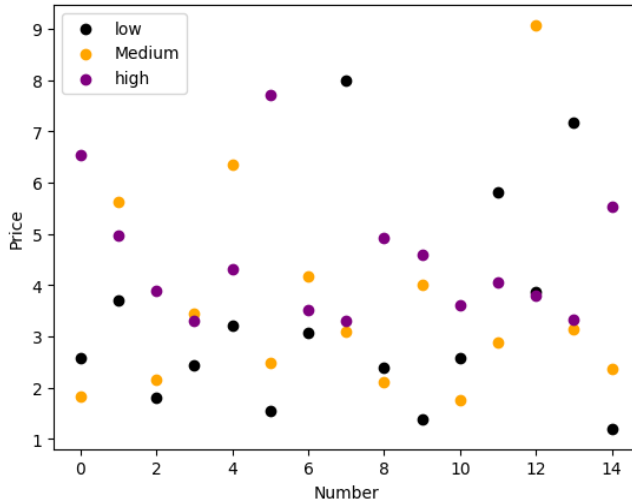
```
plt.plot(bağumsuz.low, color="black", label='low')
plt.plot(bağumsuz.medium, color="orange", label='medium')
plt.plot(bağumsuz.high, color="purple", label='high')
plt.legend()
plt.title('Line Graph for Data')
plt.xlabel('Number')
plt.ylabel('Price')
plt.show()
```

✓ 0.1s

- ❖ Grafiğimizi matplotlib kütüphanesi olarak bilinen kütüphane yardımı ile bar fonksiyonu aracılığı ile Python da çizdirmiş bulunmaktayız
- ❖ Çizgi grafiği aracılığı ile yer alan veri gruplarının değerlerinin dağılımlarını daha net bir şekilde görme fırsatı bulmuş olacağız.



- ❖ Yukarı da yer alan veri gruplarının çizgi grafikleri farklı renklerle temsil edilecek şekilde yer almaktadır.
- ❖ İlk bakış üzerinden yorum yapılmak istendiğinde her üç veri grubu içinde dalgalı ve iniş çıkışlı bir dağılım yer aldığını görmekteyiz.
- ❖ Bunun yanı sıra veri grupları üzerindeki yükseliş ve alçalışlar birbirini takip edecek şekilde kaydedilmiştir.
- ❖ Veri değerleri tepe ve taban değerlerine bakıldığında high veri grubu diğer veri gruplarına göre daha yüksek değerlere sahip olmakla beraber aynı zamanda değerler arası fark diğer gruplara göre daha azdır.
- ❖ Diğer veri grupları olan düşük ve orta bütçeli TV kanallarını incelediğimizde veri dağılımı ve almış olduğu değerler açısından daha dengesiz davranışları söz konusudur.



- ❖ Dağılımın daha da net ayırt edilebilmesi açısından nokta dağılım şeklinde grafikleştirme sağlanmıştır.
- ❖ Bu grafik aracılığı ile aynı zamanda sıralı değer bazında grupların aldıkları değerleri daha net görebilmekteyiz.
- ❖ X eksen verilerin satırlarını , y eksen ise aldıkları değerleri temsil etmektedir.

## 2.0 Bağımsız Veri Nanparametrik Testler

### 2.1 Wilcoxon Sıra Sayı Testi

- ❖ Yüksek bütçe harcaması yapan TV kanallarının harcadıkları miktarlarının meydanının 4.06 olduğu bilinmektedir.  
Mevcut bütçenin ortancasının 4.06'den az olup olmadığını Wilcoxon işaret sıra sayıları testi ile  $\alpha = 0.05$  yanılma düzeyinde test edeceğiz.

Hipotez:

$$H_0 = 4.06$$

$$H_1 < 4.06$$

HİPOTEZİMİZİN TESPİTİ GERÇEKLEŞİCEKTİR.

```
e=hg.median()
from scipy.stats import wilcoxon
wilcoxon(hg - e,
         alternative = 'less',
         method = 'approx',
         zero_method = 'zsplit',
         correction = True)
```

✓ 0.0s

WilcoxonResult(statistic=74.5, pvalue=0.8029247458745191)

- ❖ Görselden de anlaşıldığı üzere Python uygulamasının scipy kütüphanesinden wilcoxon fonksiyonu aracılığı ile sonuca ulaşmış bulunmaktayız.
- ❖ Gerekli analizlerin sonucunda wilcoxon test analiz sonucu : 74,5
- ❖ P-value değerimiz : 0.8 olarak yer almıştır.
- ❖ P-value > 0.05 olduğundan hipotezimiz reddedilemez.
- ❖ %95 güven düzeyinde TV kanallarının harcadıkları bütçelerin meydanlarının 4.06 dan küçük olmadığını söyleyebiliriz.



## 2.2 İşaret Testi

- ❖ Yüksek bütçe harcaması yapan TV kanallarının harcadıkları miktarlarının meydanının 4.06 olduğu bilinmektedir.  
Mevcut bütçenin ortancasının 4.06'den az olup olmadığını işaret sıra sayıları testi ile  $\alpha = 0.05$  yanılma düzeyinde test edeceğiz.

**Hipotez:**

$$H_0 = 4.06$$

$$H_1 < 4.06$$

HİPOTEZİMİZİN TESPİTİ GERÇEKLEŞİCEKTİR.

```
from scipy.stats import binomtest
nPlus = len(hg[hg > e])
nNeg = len(hg[hg < e])
n = nPlus + nNeg
binomtest(nPlus, n, p=0.5, alternative='less')
```

[45] ✓ 0.0s

... BinomTestResult(k=7, n=14, alternative='less', statistic=0.5, pvalue=0.604736328125)

- ❖ Python çıktısında gördüğümüz gibi işaret testi için scipy kütüphanesinden binomtest fonksiyonunu kullanarak hesaplamalarımızı gerçekleştirmiş bulunmaktayız
- ❖ Gerekli analizlerin sonucunda wilcoxon test analiz sonucu : 0.5
- ❖ P-value değerimiz : 0,60 olarak yer almıştır.
- ❖ P-value > 0.05 olduğundan hipotezimiz reddedilemez.
- ❖ Hipotezimiz reddedilemeyeceğinden dolayı veri grubumuzun medyanın 4.06 dan küçük olmadığını %95 güven düzeyinde söyleyebiliriz.

## 2.3 Mann-Whitney Testi

- ❖ Verilerimiz arasında konum yönünden fark olup olmadığını araştırmak adına nanparametrik testlerden birisi olan Mann – Whitney testinden faydalanacağız.
  - ❖ Sırasıyla Düşük-orta, Düşük-yüksek ve orta-yüksek veri gruplarına ait analizlerimizi %95 güven düzeyinde hesaplayacağız.
  - ❖ Hesaplamamalarımız için Python uygulamasının scipy kütüphanesinden stats maddesinin Mannwhitneyu fonksiyonundan faydalanılacaktır.
- 
- ❖ Her bir veri grubunun harcadıkları bütçe açısından birbiriyle karşılaştırıldığında
    1. Düşük – orta bütçeli kanallarda Düşük bütçenin daha etkili olduğu
    2. Düşük- yüksek bütçeli kanallarda yüksek bütçenin daha etkili olduğu
    3. Orta-yüksek bütçeli kanallarda Orta bütçeli kanalların daha etkili olduğuMann-Whitney testi ile söylenip söylenemeyeceğine ulaşacağız.

**Hipotez:**

$$H_0 = Q_A = Q_B$$

$$H_1 = Q_A < Q_B$$

HİPOTEZİMİZİN TESPİTİ GERÇEKLEŞİCEKTİR

```
group = [lw,md,hg]
groupname = ['düşük', 'orta', 'yüksek']
comb = (zip(group,groupname))
for grup1 , grup2 in combinations(comb , 2 ):
    mwd , mwp =stats.mannwhitneyu(grup1[0],grup2[0])
    print(f"{grup1[1]} ve {grup2[1]} :")
    print("istatistiksel değer :", mwd)
    print("p-value: \n", mwp)
```

✓ 0.0s

```
düşük ve orta :
istatistiksel değer : 100.0
p-value:
0.6186305224456992
düşük ve yüksek :
istatistiksel değer : 53.0
p-value:
0.014375111660888815
orta ve yüksek :
istatistiksel değer : 59.0
p-value:
0.027907646278761626
```

- ❖ Yukarıda belirtildiği ve görselde görüldüğü üzere uygulamamız üzerinden gerekli sonuç çıktıları elde edilmiş olup altında da p değerleri ve istatistik değerleri yer almaktadır.

### 2.3.1 Düşük- Orta

- ❖ Düşük – Orta bütçe harcaması yapan TV kanalları arasında yapmış olduğumuz Mann-Whitney konum testinin ;
- ❖ İstatistik değeri :100
- ❖ P-value : 0,61 olarak yer almıştır
- ❖ P-value > 0.05 olduğundan hipotezimiz reddedilemez.
- ❖ %95 güven düzeyinde Düşük düzeyde reklam harcaması yapan TV kanallarının orta düzeye göre daha etkili olduğunu söyleyebiliriz.

### 2.3.2 Düşük – Yüksek

- ❖ Düşük – Yüksek bütçe harcaması yapan TV kanalları arasında yapmış olduğumuz Mann-Whitney konum testinin ;
- ❖ İstatistik değeri :53
- ❖ P-value : 0,01 olarak yer almıştır
- ❖ P-value < 0.05 olduğundan hipotezimiz reddedilir.
- ❖ %95 güven düzeyinde Düşük düzeyde reklam harcaması yapan TV kanallarının yüksek düzeye göre daha etkili olmadığını ve arada bir fark olmadığını söyleyebiliriz.

### 2.3.3 Orta -Yüksek

- ❖ Orta – Yüksek bütçe harcaması yapan TV kanalları arasında yapmış olduğumuz Mann-Whitney konum testinin ;
- ❖ İstatistik değeri :59
- ❖ P-value : 0,02 olarak yer almıştır
- ❖ P-value < 0.05 olduğundan hipotezimiz reddedilir.
- ❖ %95 güven düzeyinde Orta düzeyde reklam harcaması yapan TV kanallarının yüksek düzeye göre daha etkili olmadığını ve arada bir fark olmadığını söyleyebiliriz.

### 3.0 Kruskal-Wallis Testi

- ❖ Gruplarımız arasında konum yönünden bir fark olup olmadığını anlamak adına , k örneklem konum testi olan Kruskal-Wallis testi gerçekleştirilecek olup eğer hipotezimiz doğru çıkmaz ise yine nanparametrik testlerden olan Post hoc testi ile analize devam edilecektir.

```
stats.kruskal(Lw,md,hg)
✓ 0.0s
KruskalResult(statistic=7.589019194448124, pvalue=0.022493934200113103)
```

- ❖ Görsel de de görüldüğü üzere ilgili kruskal testimizi Python aracılığı ile gerçekleştirilmiş olup p-value ve istatistik değerlerimiz yer almaktadır.
- ❖ İstatistik değeri : 7.5
- ❖ P-value : 0.02
- ❖ P-value< 0.05 olduğundan hipotezimiz reddedilir .
- ❖ Hipotezimiz reddedildiğinden ve farklılığın hangi gruptan geldiğini anlamak için detaylı incelemek adına Post hoc testlerine geçiş yapılır.

#### 3.0.1 Post hoc Testi

- ❖ Kruskal – Wallis testinin reddedilmesinden dolayı bir diğer nanparametrik testimiz olan Post hoc testlerinden dunn testi uygulanacaktır.
- ❖ Testimiz yine Python uygulaması aracılığı ile hesaplanacaktır.

```
sp.posthoc_dunn(group, p_adjust = 'bonferroni')
✓ 0.0s
```

	1	2	3
1	1.000000	1.000000	0.030347
2	1.000000	1.000000	0.096849
3	0.030347	0.096849	1.000000

- ❖ Yukarıda yer alan dunn testi sonucunda görmekteyiz ki gruplar arasındaki farklıların temel nedeni 3. Grup olan yüksek harcama bütçesine sahip TV kanallarından dolayı oluşmaktadır.

## 3.0 Bağımlı Veri

### 3.1 Veri Hakkında

- ❖ Verimiz TÜİK(Türkiye İstatistik Kurumu) aracılığı ile Excel formatında alınmıştır.
  - ❖ Verimiz 20 adet Ülkenin 2015, 2019 ve 2023 yıllarına ait ihracat miktarlarını milyar dolar bazında göstermektedir.
  - ❖ Verimiz 4 adet sütun : Ülkeler, 2015,2019 ve 2023 olmak üzere
  - ❖ 20 adet satırdan : Ülke isimleri ve veriler şeklinde oluşmaktadır.
- 
- ❖ Verimiz belirtildiği gibi TÜİK üzerinden alınmış olup incelememiş yıllık ülkelerin yapmış olduğu ihracat tutarlarının normal dağılılı sahibi olmadığını kanıtlayıp bazı nanparametrik testlerin yapılması şeklinde ilerleyecektir.
    - ✓ Temel istatistiklerin bulunması ve yorumlanması
    - ✓ Basıklık ve çarpıklık katsayıları
    - ✓ Normallik testleri
    - ✓ Görselleştirme ve grafik yorumlaması
    - ✓ Nanparametrik testler

Şeklinde sunumu ve analizi gerçekleşecektir.

### 3.2 Verinin Yüklmesi

- ❖ Verimiz Excel formatında olup yapacağımız tüm analiz ve incelemeler Python uygulaması üzerinden olacağı için verimizi önce uygulamaya aktarıyoruz.

```
bagiml = pd.read_excel(r'C:\Users\pc\Desktop\nanpar veriler\bagiml.xls')
```

1 ✓ 0.0s

- ❖ Verimiz uygulamaya pandas kütüphanesinin kısaltması olan pd den read\_excel fonksiyonu aracılığı ile aktarılmış olup aktarma esnasında görüldüğü gibi bilgisayarımızda ilgili veri dosyasının nerede olduğu net bir şekilde tanımlanmıştır.
- ❖ Veri dosyası bağımlı adında tanımlanmış olup bundan sonraki süreçte işlem yapılmak istendiğinde bu isim aracılığı ile ulaşım sağlanacaktır.

### 3.3 Verinin Çıktısı

- ❖ Bir önceki maddede belirtildiği gibi veriyi uygulamamıza çektikten sonra bağımlı başlığı altında çağırdığımızda aşağıdaki gibi bir görüntü ve çıktı elde etmekteyiz.

	Ülke	ikibinonbeş	ikibinondokuz	ikibinyirmiiç
0	Almanya	14.49	16.61	21.14
1	ABD	7.01	8.97	16.88
2	Irak	9.96	10.22	13.75
3	Birleşik Krallık	10.82	11.27	13.00
4	İtalya	7.15	9.75	12.38
5	İspanya	4.94	8.13	9.65
6	Fransa	6.10	7.94	9.53
7	Rusya Federasyonu	3.68	4.15	9.34
8	Hollanda	3.35	5.76	8.02
9	İsrail	2.80	4.46	7.03
10	Romanya	2.92	4.07	6.94
11	Polonya	2.42	3.44	5.17
12	BAE	4.93	3.62	5.25
13	Belçika	2.72	3.39	4.77
14	Bulgaristan	1.76	2.66	4.72
15	Mısır	3.24	3.50	4.55
16	Yunanistan	1.48	2.24	3.30
17	Çin	2.50	2.72	3.28
18	Fas	1.37	2.34	3.09
19	İran	4.11	2.73	3.06

❖ Görüldüğü üzere verimiz yanda yer aldığı gibi bir görünüşe sahip olmakla birlikte verimize tekrardan bir göz attığımızda 4 adet sütun ve 20 adet satırdan oluştuğunu görmekteyiz.

❖ Verimiz üzerinden yer alan ülke sütununu grafikleri dışında aktif olarak kullanmayacak olup bizler için önemli olan

sütunlar 2015, 2019 ve 2023 sütunlarına ait veriler olacaktır.

### 3.4 Verinin Temel Analizleri

- ❖ Bu bölümde yine Python uygulaması aracılığı ile verimize ait temel istatistiklerin analizini ve yorumlamasını gerçekleştirecek olup özellikle veri üzerinde uygulayacağımız normallik testi nanparametrik testlerin uygun olup olmayacağı hakkında bizlere bilgi verecektir.
- ❖ Analizlerin bazı aşamalarında bu analizlerin aynı zamanda grafiksel gösterimlerine de yer verilecektir.

### 3.4.1 Tanımlayıcı İstatistikler

- ❖ Verimiz üzerinde veri sayısı , mod, medyan , ortalama standart sapma gibi temel ve basit sonuçların yer aldığı ve yorumlandığı bir alandır.



```
bağımlı.describe()
```

✓ 0.0s

	ikibinonbeş	ikibinondokuz	ikibinyirmiüç
count	20.000000	20.000000	20.000000
mean	4.887500	5.898500	8.242500
std	3.478148	3.867054	5.016048
min	1.370000	2.240000	3.060000
25%	2.665000	3.225000	4.677500
50%	3.515000	4.110000	6.985000
75%	6.327500	8.340000	10.332500
max	14.490000	16.610000	21.140000

- ❖ Öncelikli olarak verimize ait bu temel istatistik değerlerine ulaşmak için veri adımız olan bağımlıdan “bağımlı.describe()” yapmamız gerekmektedir.
- ❖ Hemen ardından görüyoruz ki verimize ait temel istatistik değerleri karşımıza çıkmaktadır.
- ❖ Burada direk olarak gözümüze ülke isimlerinin olduğu sütunun yer almadığı çıkmaktadır. Bunun temel nedeni her birisi farklı bir ülke ismi olarak yer aldığından analize uygun bir veri tipi olmamasındandır.

- ❖ Verimizin temel değerlerine baktığımızda her bir veri grubunun 20adet gözlemden oluştuğunu, ortalamalar arasında her geçen yıl fark oluştuğunu ve buna bağlı olarak da her geçen yıl ülkelerin toplam bazda daha fazla ihracat yaptığını söyleyebiliriz.
- ❖ Bunların haricinde standart sapmaları veri gruplarına uygun şekilde bir dağılıma sahiptir.
- ❖ Max ve min değerlerin gözlemi üzerinde de ilk maddede belirttiğimiz yıllık bazda artış tespitini ispat etmiş oluyoruz.

### 3.4.2 Çarpıklık Katsayısı

- ❖ Normallik testinden hemen önce veriye ve verinin dağılımına dair fikir sahibi olabilmek için çarpıklık katsayısının incelemesini gerçekleştireceğiz.
- ❖ Çarpıklık katsayısı -1 ile +1 arasında değişiyor olmasını ve + değer alıyor ise sola çarpık, - değer alıyor ise dağılımın sağa çarpık olduğunu anlayabiliriz.
- ❖ Eğer değerimiz -1 ile +1 dışına çıkıyor ise normal dağılımdan uzak olduğunu anlayabiliriz.

```
skew(onbeş, axis=0, bias=True)
21] ✓ 0.0s
.. 1.3707563984153066

skew(ondokuz, axis=0, bias=True)
22] ✓ 0.0s
.. 1.2211877224149343

skew(yirmiüç, axis=0, bias=True)
23] ✓ 0.0s
.. 1.038413278382718
```

- ❖ Yukarıda görüldüğü üzere çarpıklık katsayılarımız sırası ile 1 'in üzerinde olmakla beraber veri dağılımımızın sola çarpık ve normallikten de uzak olduğunu söyleyebiliriz.

### 3.4.3 Basıklık Katsayısı

- ❖ Basıklık katsayısı da yine aynı şekilde 0'a ne kadar yakınsa normalliği o kadar sağlamakla beraber -1 ile +1 arasında değer almaktadır.



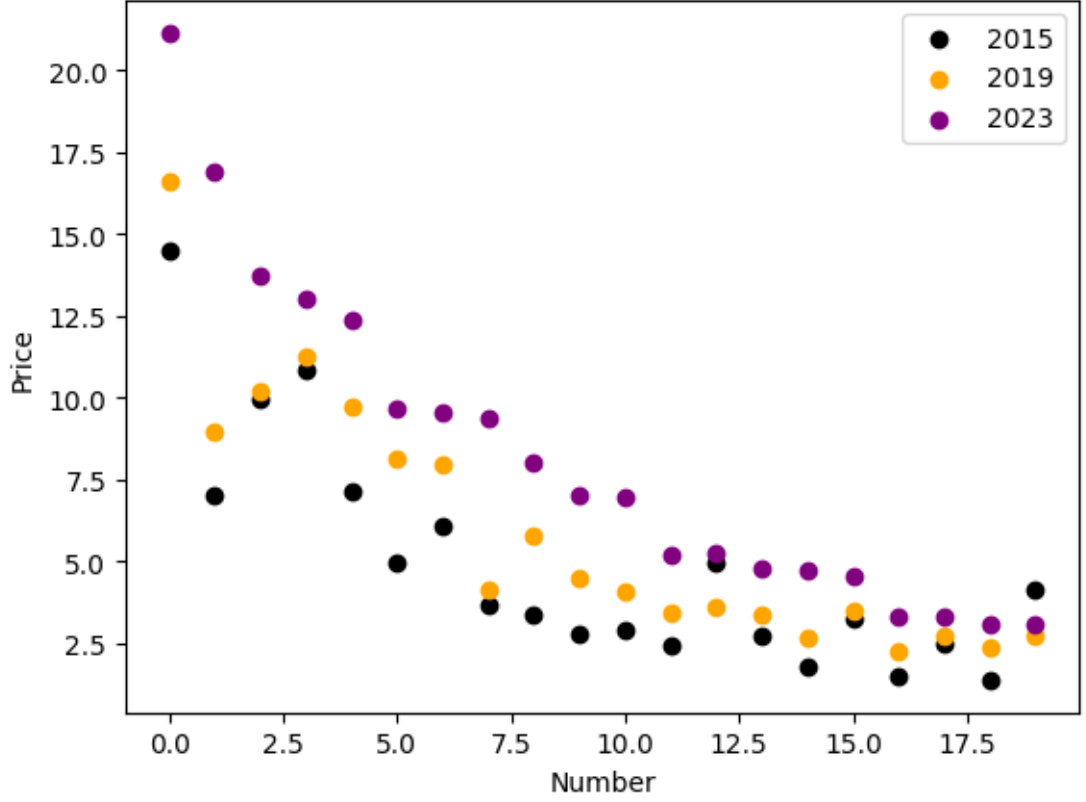
- ❖ Eğer basıklık değerimiz pozitif ise sivri olduğunu, negatif ise basık olduğunu ifade etmektedir.

```
[24] kurtosis(onbeş, axis=0 ,bias=True)
✓ 0.0s
... 1.1551513492531216

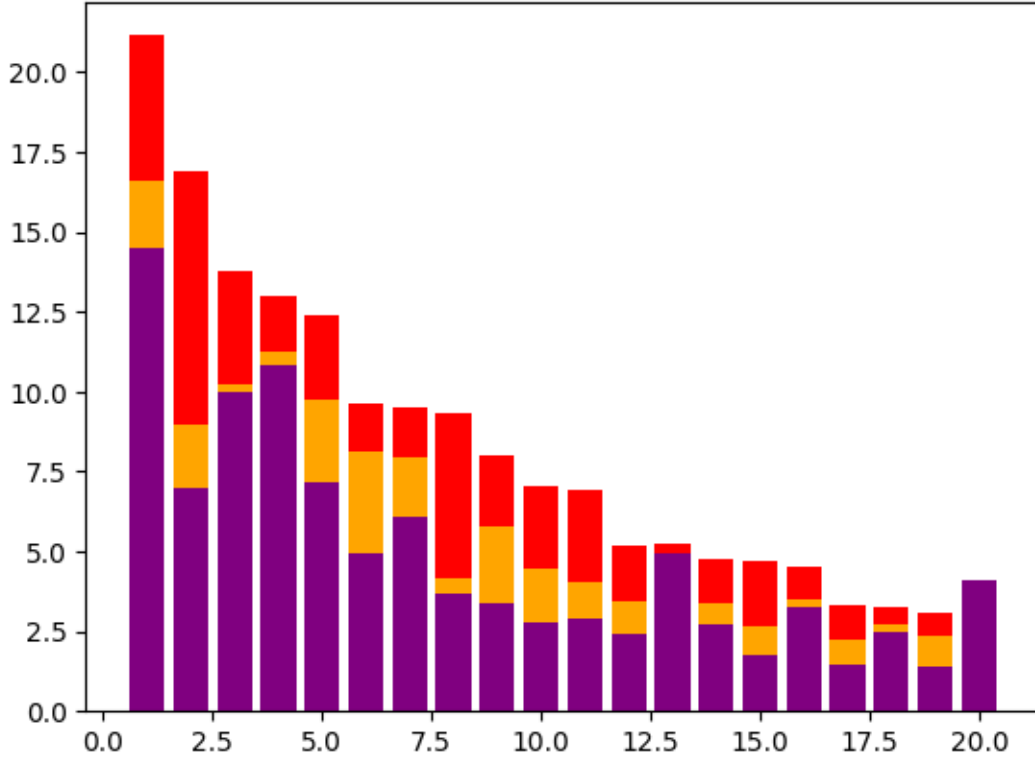
[25] kurtosis(ondokuz, axis=0 ,bias=True)
✓ 0.0s
... 0.8425518092387398

[26] kurtosis(yirmiüç, axis=0 ,bias=True)
✓ 0.0s
... 0.3482285826895639
```

- ❖ Yukarıda görselde görüldüğü üzere basıklık değerinin hesabı için uygulama içerisinde kurtosis fonksiyonundan faydalanılmıştır.
- ❖ Oluşan sonuçları incelediğimizde tamamının pozitif olduğunu yani sivri bir dağılıma sahip olduğunu görmekteyiz.
- ❖ 3. Veri grubumuz olan 2023 yılına ait çarpıklık katsayısı değerinin 0'a daha yakın olduğunu görmekteyiz, bu yüzden 2023 yılına ait ihracat tutarlarının verisi normallığe yakın olmakla beraber oda sivri bir dağılıma sahiptir.



- ❖ Verinin dağılımını daha net bir şekilde görebilmek adına yukarıda yer alan nokta grafiği üzerinde verilen aldıkları değerleri gösteren grafiklerimize yer verilmiştir.
- ❖ Grafikte de görüldüğü üzere 2023 ihracat verileri diğer yıllara nazaran daha yüksek değerler almakla beraber tepe değerlerini genel olarak 1. Dünya ülkeleri olarak adlandırdığımız ABD , Almanya , Çin gibi ülkeler , taban değerlerinde ise daha çok 3. Dünya ülkeleri olarak adlandırabileceğimiz ülkeler yer almaktadır.
- ❖ Bunun yanı sıra dağılımlarda da gördüğümüz gibi veri dağılımları genel olarak normallikten uzak ve daha sivri bir yapıya sahiptir.



- ❖ Verilerin dağılımları ayrıca yukarıda yer alan dağılım grafiğinde gösterilmiştir.
- ❖ Mor renk 2015, Turuncu renk 2019, Kırmızı renk ise 2023 verilerini temsil etmektedir.
- ❖ Bir önceki nokta grafiğinin dağılımsal olarak yükseltilerini göstermektedir.

### 3.5 Normal Dağılım İncelemesi

- ❖ Verimizin dağılımı ve temel istatistikleri hakkında az çok bilgi sahibi olduktan sonra artık direkt olarak normallik testi üzerinden dağılımın normal olup olmadığının analizini gerçekleştireceğiz.
- ❖ Verimizin normal dağılıp dağılmadığını anlayabilmek adına  $n < 20$  olduğundan dolayı Shapiro-Wilk Testi ile verimizin normalliğini test edeceğiz.

**Hipotez:**

$H_0$  = Veri setinin dağılımı normaldir.

$H_1$  = Veri setinin dağılımı normal değildir

```
bağmlı1 = [onbeş,ondokuz,yirmiüç]
bağ2 = pd.DataFrame(bağmlı1)
bağ2
```

✓ 0.0s

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
ikibinonbeş	14.49	7.01	9.96	10.82	7.15	4.94	6.10	3.68	3.35	2.80	2.92	2.42	4.93	2.72	1.76	3.24	1.48	2.50	1.37	4.11
ikibinondokuz	16.61	8.97	10.22	11.27	9.75	8.13	7.94	4.15	5.76	4.46	4.07	3.44	3.62	3.39	2.66	3.50	2.24	2.72	2.34	2.73
ikibinyirmiüç	21.14	16.88	13.75	13.00	12.38	9.65	9.53	9.34	8.02	7.03	6.94	5.17	5.25	4.77	4.72	4.55	3.30	3.28	3.09	3.06

- ❖ Normallik incelemesinden önce verimizi daha detaylı ve gerekli kısmın görülmesi için verimizin sadece 2015, 2019, 2023 yıllarına ait olan kısımlarını yeni bir veri kümesi haline getirmiş bulunmaktayız.

### 3.5.1 2015 Verisi

```
ob= stats.shapiro(onbeş)
ob
```

✓ 0.0s

ShapiroResult(statistic=0.8419543504714966, pvalue=0.003919442184269428)

- ❖ Kodumuz görüldüğü üzere stats kütüphanesi aracılığı ile ve shapiro fonksiyonu bile basit bir şekilde çalışmaktadır.
- ❖ Yukarıda görüldüğü üzere test istatistiği sonucu 0.84 , p-value değeri ise 0.03 çıkmaktadır.
- ❖ P-value < 0.05 olduğundan hipotezimiz rededilmektir.Buda verimizin normal dağılmadığının ispatıdır.

### 3.5.2 2019 Verisi

```
od = stats.shapiro(ondokuz)
od
```

✓ 0.0s

ShapiroResult(statistic=0.8360551595687866, pvalue=0.003140056738629937)

- ❖ Kodumuz görüldüğü üzere stats kütüphanesi aracılığı ile ve shapiro fonksiyonu bile basit bir şekilde çalışmaktadır.
- ❖ Yukarıda görüldüğü üzere test istatistiği sonucu 0.83 , p-value değeri ise 0.03 çıkmaktadır.
- ❖ P-value < 0.05 olduğundan hipotezimiz rededilmektir.Buda verimizin normal dağılmadığının ispatıdır.

### 3.5.3 2023 Verisi

```
od = stats.shapiro(ondokuz)
od
✓ 0.0s
ShapiroResult(statistic=0.8360551595687866, pvalue=0.003140056738629937)
```

- ❖ Kodumuz görüldüğü üzere stats kütüphanesi aracılığı ile ve shapiro fonksiyonu bile basit bir şekilde çalışmaktadır.
- ❖ Yukarıda görüldüğü üzere test istatistiği sonucu 0.83 , p-value değeri ise 0.03 çıkmaktadır.
- ❖ P-value < 0.05 olduğundan hipotezimiz rededilmektedir. Buda verimizin normal dağılmadığının ispatıdır.

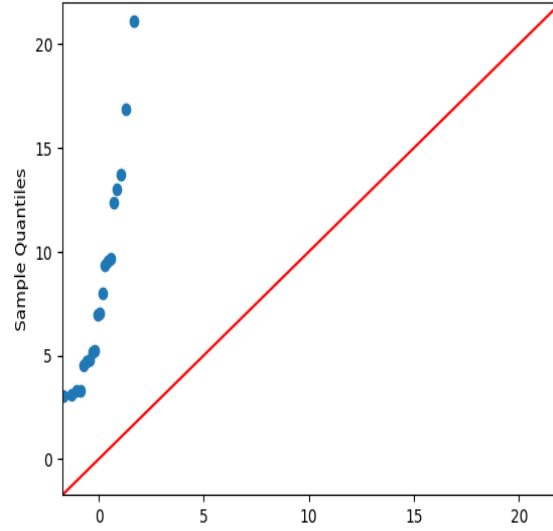
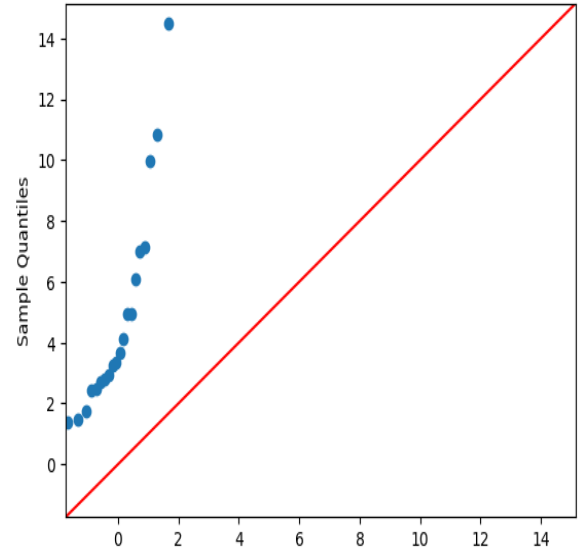
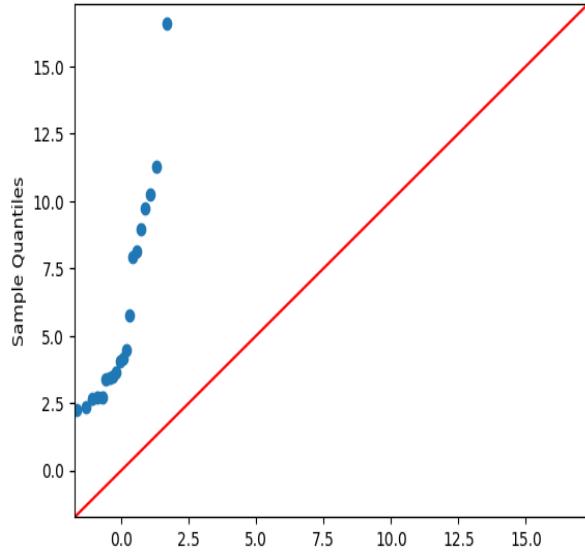
### 3.6 Normallik Grafikleri

- ❖ Bir önceki maddede görüldüğü üzere verilerimiz normal dağılıma uygun olmamakla beraber bunu daha net görebilmek üzere Q-Q plot üzerinde tek tek ve aynı zamanda toplu bir şekilde incelemelere devam edeceğiz.
- ❖ Q-Q plot yorumlanırken grafik üzerinde yer alan eğriye ve verinin bu eğrinin hangi alanında seyrettiğine göre yorumlama yapılmaktadır.
- ❖ Eğer veriler yer alan eğrinin etrafında ve yakınında yer alıyor ise verinin normal dağılım gösterdiğini , eğer veriler eğriden uzak gözlemleniyorsa normal dağılmadığını söyleyebiliriz.

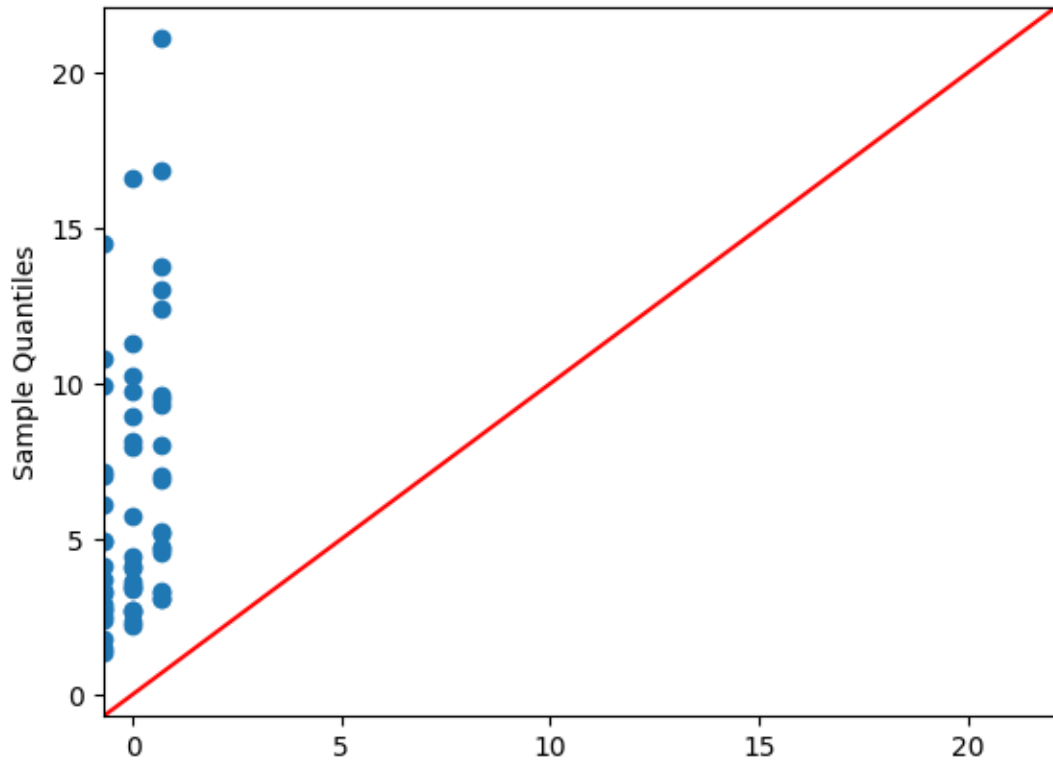
#### 3.6.1 Q-Q plot

```
sm.qqplot(bağ2, line='45')
plt.xlabel("")
plt.ylabel("Sample Quantiles")
plt.show()
✓ 0.1s
```

- ❖ Python uygulamasının matplotlib adını verdiğimiz plt kütüphanesi üzerinden qqplot fonksiyonu aracılığı ile yukarıda yer aldığı gibi grafiğimizi oluşturmaktayız.



- ❖ Yukarıda sırasıyla 2015,2019 ve 2023 yıllarına ait verilerin Q-Q plot grafikleri yer almaktadır
- ❖ Her bir grafiği sırasıyla incelediğimiz de bir önceki maddede de yer aldığı gibi grafik üzerine normal dağılım eğrisi ve verilerin nokta halinde gösterimleri yer almaktadır.
- ❖ Grafikleri incelediğimizde ise her bir veri görüldüğü üzere normal dağılım eğrisinden oldukça uzakta yer almış olup tüm verilerin normal dağılıma uymadığını görüntülemekteyiz.
- ❖ Verilerimiz normal dağılıma uygun olmadığından dolayı herhangi bir ekstra işlem gereksinimi duymaksızın tüm veriler için nanparametrik analiz ve nanparametrik testlerden faydalanarak analizlerimize devam edebiliriz.

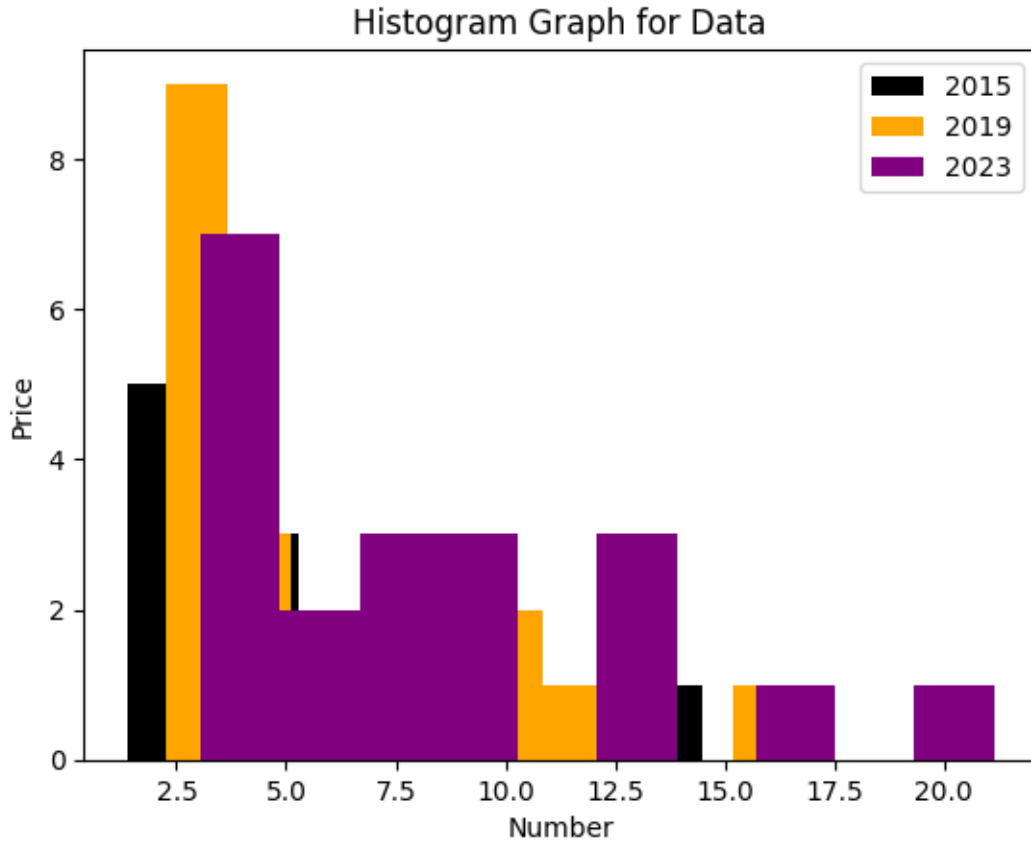


- ❖ Yukarı da yer alan grafiğimiz tüm veri gruplarını tek bir grafik altında toplayacak şekilde oluşturulmuş bir Q-Q plot grafiğidir.
- ❖ Grafiğimiz içerisinde 3 veri grubumuzda yer almakta olup bir önceki grafikte tüm verilerimizi ayrı ayrı incelediğimizde her üçünüde normallikten uzak olduğunu görmüştük.
- ❖ Her üç grubumuza barındıran bu grafiğimizi incelediğimizde yine aynı şekilde normallikten uzak olduğunu ve normal bir dağılıma sahip olmadığını söyleyebiliriz.

### 3.7 Histogram Grafiği

```
plt.hist(bagunsuz.low, color="black", label='low')
plt.hist(bagunsuz.medium, color="orange", label='medium')
plt.hist(bagunsuz.high, color="purple", label='high')
plt.legend()
plt.title('Histogram Graph for Data')
plt.xlabel('Number')
plt.ylabel('Price')
plt.show()
```

- ❖ Grafiğimizi matplotlib kütüphanesi olarak bilinen kütüphane yardımı ile hist fonksiyonu aracılığı ile Python da çizdirmiş bulunmaktayız.
- ❖ Grafiğimiz tüm veri gruplarının histogram grafiği üzerinde gösterimini baz alacak şekilde oluşturulmuştur.



- ❖ Yukarıda görüldüğü üzere tüm veri grupları ve değerleri tek bir grafik içerisinde histogram olarak verilmiştir
- ❖ Grafiğimizi incelediğimizde grafiğimiz yaptığımız hesaplamalarla uyumlu olacak şekilde pozitif çarpık şekilde ve sivri bir yapıya sahiptir.
- ❖ Grafiğimiz pozitif çarpık ve sivri olması kaynaklı normal dağılımdan oldukça uzaktır.

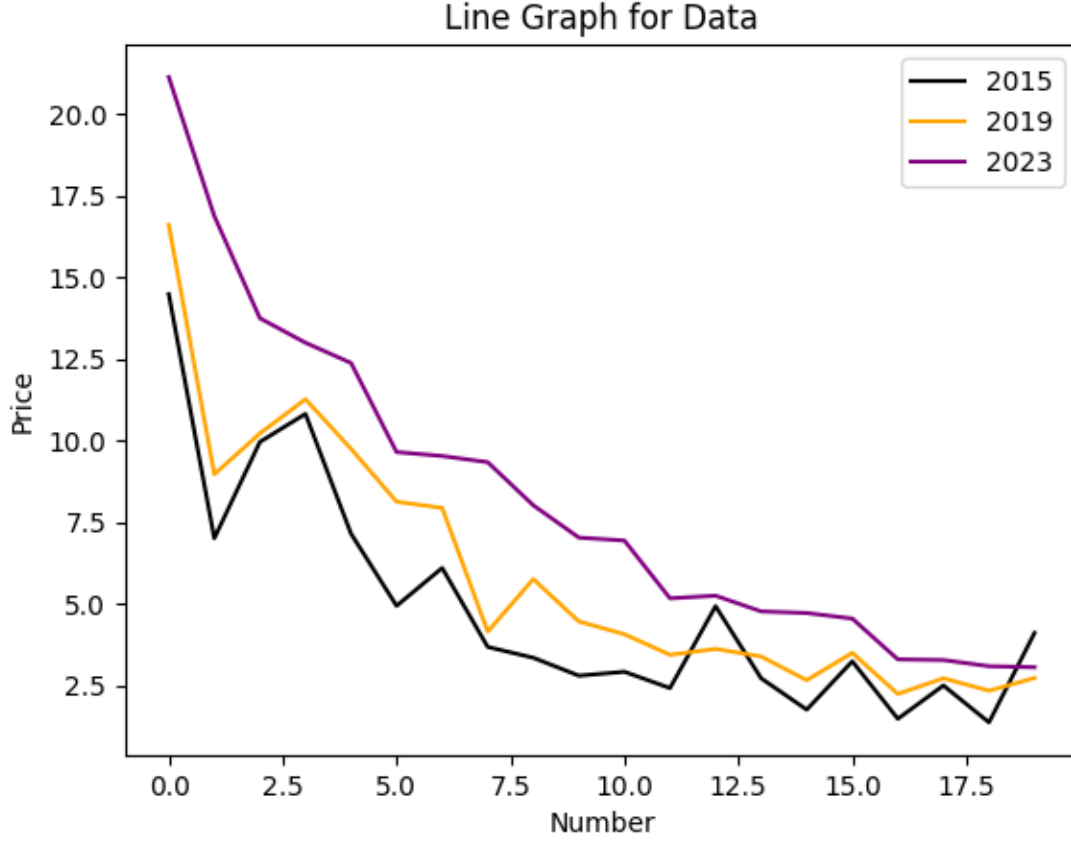
### 3.8 Line(Çizgi) Grafiği

```
plt.plot(onbeş, color = "black", label = '2015')
plt.plot(ondokuz, color = "orange", label = '2019')
plt.plot(yirmiüç, color = "purple", label = '2023')
plt.legend()
plt.title('Line Graph for Data')
plt.xlabel('Number')
plt.ylabel('Price')
plt.show()
```

✓ 0.1s

- ❖ Grafiğimizi matplotlib kütüphanesi olarak bilinen kütüphane yardımı ile bar fonksiyonu aracılığı ile Python da çizdirmiş bulunmaktayız
- ❖ Çizgi grafiği aracılığı ile yer alan veri gruplarının değerlerinin dağılımlarını daha net bir şekilde görme fırsatı bulmuş olacağız.





- ❖ Yukarı da yer alan grafiğimiz üzerinde öncelikli olarak belirli renklerin belirli yıllara ait verileri temsil ettiğini görmekteyiz.
- ❖ X eksini Ülkeleri , y ekseni ise bu yıllara göre verilerin almış olduğu değerleri temsil etmektedir.
- ❖ X ekseninde yer alan Ülkelerin uzunluğu kaynaklı x ekseninde isimler yerine bu ülkelere verilmiş olan değerler ve sıralamalara göre yer almaktadır.
- ❖ Grafiğimizi incelediğimizde genel olarak ülkelerin ihracatlarının her yıl artış gösterdiğini fakat istisnai olarak bazı ülkelerin de düşüş gösterdiğini net bir şekilde görmekteyiz.
- ❖ Veri değerleri tepe ve taban değerlerine bakıldığında 2023 veri grubu diğer veri gruplarına göre daha yüksek değerlere sahip olmakla beraber aynı zamanda değerler arası fark diğer gruplara göre daha azdır.

## 3.0 Bağımlı Veri Nanparametrik Testler

### 3.1 Wilcoxon Sıra Sayı Testi

- ❖ 2019 ve 2023 yıllarına ait ülkelerin yaptıkları ihracat tutarları milyon dolar cinsinden verimiz içerisinde yer almakta olup %95 güven düzeyinde yılların ihracat miktarının artışına etkisinin olup olmadığı Wilcoxon sıra sayı testi aracılığı ile test edilecektir.
- ❖ Testimiz Python uygulaması üzerinde gerçekleştirilecek olup elde edilen çıktı ve sonucuna göre yukarıda belirtilen etkinin var olup olmadığı sonucuna ulaşılabacaktır.

**Hipotez:**

$$H_0 = Q_A = Q_B$$

$$H_1 = Q_A < Q_B$$

HİPOTEZİMİZİN TESPİTİ GERÇEKLEŞİCEKTİR

```
> ##wilcoxon testi
stats.wilcoxon(ondokuz, yirmiüç,
alternative='greater',
method = 'approx',
zero_method = 'zsplit',
correction = True)
[53] ✓ 0.0s
... WilcoxonResult(statistic=0.0, pvalue=0.9999590202102997)
```

- ❖ Uygulama üzerinden scipy kütüphanesinde yer alan stats'a bağlı wilcoxon fonksiyonu aracılığı ile görüldüğü şekilde testimiz gerçekleştirilmiştir.
- ❖ Sonuca göre test istatistiği : 0
- ❖ P-value : 0.99 dır.
- ❖ P-value > 0.05 olduğundan hipotezimiz kabul edilmiştir.
- ❖ %95 güven düzeyinde yılların ülkelerin yapmış olduğu ihracat miktarlarını milyon dolar cinsinde etkili olduğunu söyleyebiliriz.

## 3.2 Friedman Testi

- ❖ Gruplarımız arasında konum yönünden bir fark olup olmadığını anlamak adına , k örneklem konum testi olan Friedman testi gerçekleştirilecek olup eğer hipotezimiz doğru çıkmaz ise yine nanparametrik testlerden olan Post hoc testi ile analize devam edilecektir.

```
stats.friedmanchisquare(onbeş,ondokuz,yirmiüç)
✓ 0.0s
FriedmanchisquareResult(statistic=32.69999999999999, pvalue=7.930219729907684e-08)
```

- ❖ Görselde de görüldüğü üzere gerekli olan testimiz Python uygulamasının stats kütüphanesinden friedmanchisquare fonksiyonu aracılığı ile gerçekleştirilmiştir.
- ❖ İstatistik değeri : 32.69
- ❖ P-value : 7.930e-08
- ❖ P-value <0.05 olduğundan hipotezimiz reddedilir.
- ❖ Hipotezimiz reddedilmesine neden olan veri grubunu öğrenebilmek adına post hoc testi ile analize devam edilir.

### 3.2.1 Post Hoc Testi

- ❖ Friedman testinin reddedilmesinden dolayı bir diğer nanparametrik testimiz olan Post hoc testlerinden dunn testi uygulanacaktır.
- ❖ Testimiz yine Python uygulaması aracılığı ile hesaplanacaktır.
- ❖ Verimize ait Post hoc hesapları dunn ve nemenyi testleri aracılığı ile uygulanmıştır.

```
sp.posthoc_dunn(data1, p_adjust = 'bonferroni')
✓ 0.1s
```

	1	2	3
1	1.000000	1.000000	0.026298
2	1.000000	1.000000	0.268768
3	0.026298	0.268768	1.000000

```
data1 = np.array(hog2)
sp.posthoc_nemenyi_friedman(hog2)
```

34] ✓ 13s

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	1.000000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.854289	0.763186	0.336284	0.899837	0.336284	0.102152	0.489674	0.034287	0.102152	0.021016	0.291376
1	0.900000	1.000000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.763186	0.900000	0.763186	0.440145	0.899837	0.212548	0.440145	0.149700	0.717635
2	0.900000	0.900000	1.000000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.626535	0.900000	0.626535	0.291376	0.763186	0.124999	0.291376	0.083958	0.580983
3	0.900000	0.900000	0.900000	1.000000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.580983	0.900000	0.580983	0.249612	0.717635	0.102152	0.249612	0.067895	0.535431
4	0.900000	0.900000	0.900000	0.900000	1.000000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.808736	0.900000	0.808736	0.489674	0.900000	0.249612	0.489674	0.178781	0.763186
5	0.900000	0.900000	0.900000	0.900000	0.900000	1.000000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.763186	0.900000	0.535431	0.763186	0.440145	0.900000
6	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	1.000000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.808736	0.900000	0.580983	0.808736	0.489674	0.900000
7	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	1.000000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.854289	0.900000
8	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	1.000000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.854289	0.900000
9	0.854289	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	1.000000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000
10	0.763186	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	1.000000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000
11	0.336284	0.763186	0.626535	0.580983	0.808736	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	1.000000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000
12	0.899837	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	1.000000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000
13	0.336284	0.763186	0.626535	0.580983	0.808736	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	1.000000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000
14	0.102152	0.440145	0.291376	0.249612	0.489674	0.763186	0.808736	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	1.000000	0.900000	0.900000	0.900000	0.900000	0.900000
15	0.489674	0.899837	0.763186	0.717635	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	1.000000	0.900000	0.900000	0.900000	0.900000
16	0.034287	0.212548	0.124999	0.102152	0.249612	0.535431	0.580983	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	1.000000	0.900000	0.900000	0.900000
17	0.102152	0.440145	0.291376	0.249612	0.489674	0.763186	0.808736	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	1.000000	0.900000	0.900000
18	0.021016	0.149700	0.083958	0.067895	0.178781	0.440145	0.489674	0.854289	0.854289	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	1.000000	0.900000
19	0.291376	0.717635	0.580983	0.535431	0.763186	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	1.000000

- ❖ Yukarıdaki çıktılarda da görüleceği üzere dunn testi ile direkt gruplar arası , nemenyi testi aracılığı ile de tüm veriler arasında bir karşılaştırma söz konusu olmuştur.
- ❖ Daha rahat inceleme sağlanması üzerinden görmekteyiz ki  $< 0.05$  olan grubumuz 3. Grup olan 2023 ihracat verileri olup gruplar arası farklılığın sebebidir.

## 4.0 Eğilim Testi

### 4.1 Uygun Veri Tipi

- ❖ Eğilim testi yapabilmek adına uygun görülmüş olan verimizi belirli tarihler de ki sıcaklık değerlerini veren bir veridir.

	gün	derece
0	2009-02-01	32.6
1	2009-02-02	24.4
2	2009-02-03	21.5
3	2009-02-04	21.7
4	2009-02-05	21.5
5	2009-02-06	23.5
6	2009-02-07	22.3
7	2009-02-08	28.3
8	2009-02-09	18.4
9	2009-02-10	14.9
10	2009-02-11	13.5
11	2009-02-12	16.1
12	2009-02-13	14.6
13	2009-02-14	12.4
14	2009-02-15	13.3
15	2009-02-16	17.2
16	2009-02-17	12.5
17	2009-02-18	18.0
18	2009-02-19	16.2
19	2009-02-20	18.7
20	2009-02-21	13.7
21	2009-02-22	15.5
22	2009-02-23	14.3
23	2009-02-24	12.9
24	2009-02-25	8.9
25	2009-02-26	15.0
26	2009-02-27	15.4
27	2009-02-28	16.0

- ❖ Verimiz yanda görüldüğü şekilde yer almaktadır.
- ❖ Veri grubumuz 2 sütun ve 28 satır olmak üzere toplamda 56 gözlemden oluşmaktadır.
- ❖ Sütunlarımızdan ilki görselde de görüldüğü üzere tarihileri belirtmektedir.
- ❖ Diğer sütunumuz ise belirtilen tarihte yer alan hava sıcaklık derecesine aittir.

### 4.2 Mann-Kendall Testi

```
import pymankendall as mk
test1 = np.array(test.derece)
mk.original_test(test1)
```

✓ 0.0s

Mann\_Kendall\_Test(trend='decreasing', h=True, p=0.0006768596985562425, z=-3.3987847243535483, Tau=-0.4576719576719577, s=-173.0, var\_s=2561.0, slope=-0.3909090909090909, intercept=21.327272727272728)

# BÖLÜM 2

## 1.0 KÜTÜPHANELER HAKKINDA

- ❖ Bu bölümümüzde analiz, görselleştirme ve nanparametrik analizlerin kodlanmasında yer alan kütüphaneleri kısaca tanıyacağız.

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import scipy
from scipy.stats import skew
from scipy.stats import kurtosis
from scipy import stats
import statsmodels.api as sm
from itertools import combinations
from scipy.stats import kruskal
import scikit_posthocs as sp
```

[155] ✓ 0.5s

- Numpy:** Temel bir Python kütüphanesi olup istatistiksel analiz ve matematik alanların da kullanım kolaylığı sağlamaktadır. **NumPy** (Numerical Python) bilimsel hesaplamaları hızlı bir şekilde yapmamızı sağlayan bir matematik kütüphanesidir. Numpy'ın temelini numpy dizileri oluşturur. Numpy dizileri Python listelerine benzer fakat hız ve işlevsellik açısından Python listelerinden daha kullanışlıdır. Ayrıca Python listelerinden farklı olarak Numpy dizileri homojen yapıda olmalıdır yani dizi içindeki tüm elemanlar aynı veri tipinden olmalıdır.
- Pandas:** pandas, veri işlemesi ve analizi için Python programlama dilinde yazılmış olan bir yazılım kütüphanesidir. Bu kütüphane temel olarak zaman etiketli serileri ve sayısal tabloları işlemek için bir veri yapısı oluşturur ve bu şekilde çeşitli işlemler bu veri yapısı üzerinde gerçekleştirilebilir olur. Yazılım ücretsizdir ve bir çeşit BSD ile lisansına sahiptir. Yazılım ismini bir ekonometri terimi olan veri panelinden almıştır. Bir veri paneli birçok zaman aralığı içinde farklı gözlemlerin işlenebildiği yapıyı tarif eder. Temel bir Python kütüphanesi olup yine aynı şekilde analiz ve matematik alanlarında kullanım kolaylığı sağlamaktadır.

- c. **Scipy:** pandas, veri işleme ve analizi için Python programlama dilinde yazılmış olan bir yazılım kütüphanesidir. Bu kütüphane temel olarak zaman etiketli serileri ve sayısal tabloları işlemek için bir veri yapısı oluşturur ve bu şekilde çeşitli işlemler bu veri yapısı üzerinde gerçekleştirilebilir olur. Yazılım ücretsizdir ve bir çeşit BSD ile lisansına sahiptir. Yazılım ismini bir ekonometri terimi olan veri panelinden almıştır. Bir veri paneli birçok zaman aralığı içinde farklı gözlemlerin işlenebildiği yapıyı tarif eder.
- d. **Matplotlib;** veri görselleştirmesinde kullandığımız temel python kütüphanesidir. 2 ve 3 boyutlu çizimler yapmamızı sağlar. Matplotlib genelde 2 boyutlu çizimlerde kullanılırken, 3 boyutlu çizimlerde başka kütüphanelerden yararlanır.
- e. **İtterTolls:** Döngüleri kullanırken her şey toz pembe ilerler fakat veri büyüdükçe döngüye sokmak biraz zorlaşmaya başlar. Bu tür durumlar için Python ile gelen core modüllerden biri olan itertools bize yardımcı oluyor. Hızlı çalışan, bilgisayarın kaynaklarını verimli kullanan fonksiyonlar barındırıyor.
- f. **Stats:** İstatistiksel modelleri tahmin etmek ve istatistiksel testler yapmamızı sağlayan bir kütüphanedir.

❖ Python uygulamasının temel çalışma mantığı kütüphaneler ve bu kütüphaneler içerisindeki fonksiyonların kullanımı biçiminde tasarlanmış olduğu için raporumuz içerisinde aynı işlemi yapan ve çıktısını veren fakat farklı kütüphaneler aracılığıyla yapılmış olma durumu söz konusudur. Bunun temel sebebi belirtilmiş olduğu gibi uygulamamız içerisinde çok fazla kütüphane bulunmakta olup bu kütüphaneler içerisinde temel bazı farklılıklara dayalı fonksiyonlar yer alabilmektedir. Bu yüzden özellikle görsel açıdan daha elverişli ve aynı işlevi yerine getiren farklı kütüphaneler ve fonksiyonlara rastlamak mümkündür.



## 2.0 PYTHON ÇIKTILARI(EK)

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import scipy
from scipy.stats import skew
from scipy.stats import kurtosis
from scipy import stats
import statsmodels.api as sm
from itertools import combinations
from scipy.stats import kruskal
import scikit_posthocs as sp

[1] ✓ 79s Python
```

```
bagimsiz = pd.read_excel(r'C:\Users\pc\Desktop\nanpar-veriler\75887C48.xlsx')
bagimsiz

[2] ✓ 1.5s Python
```

	low	medium	high
0	2.57	1.82	6.55
1	3.70	5.62	4.96
2	1.80	2.15	3.89
3	2.45	3.44	3.31
4	3.21	6.36	4.32
5	1.54	2.49	7.72
6	3.07	4.18	3.51
7	8.00	3.09	3.31
8	2.40	2.12	4.92
9	1.38	4.02	4.60
10	2.57	1.75	3.61
11	5.82	2.89	4.06
12	3.88	9.07	3.79
13	7.16	3.15	3.33
14	1.20	2.36	5.52

```
bagimsiz.describe()

[3] ✓ 0.0s Python
```

	low	medium	high
count	15.000000	15.000000	15.000000
mean	3.383333	3.634000	4.493333
std	2.067357	2.013529	1.284082
min	1.200000	1.750000	3.310000
25%	2.100000	2.255000	3.560000
50%	2.570000	3.090000	4.060000
75%	3.790000	4.100000	4.940000
max	8.000000	9.070000	7.720000

```
skew(bagimsiz, axis=0, bias=True)

[4] ✓ 0.0s Python
```

```
array([1.13248532, 1.49783124, 1.27816861])
```

```
kurtosis(bagimsiz, axis=0, bias=True)

[5] ✓ 0.0s Python
```

```
array([0.15172462, 1.57487675, 0.88549145])
```

```
lw = bagimsiz.low
t1,hg2 = stats.shapiro(lw)
print("Test-istatistiği :",hg1)
print("p-value :", hg2)

[6] ✓ 0.0s Python
```

```
Test-istatistiği : 0.858967586844968
p-value : 0.817922883853316387
```

```
md = bagimsiz.medium
t1,hg2 = stats.shapiro(md)
print("Test-istatistiği :",hg1)
print("p-value :", hg2)
```



```
... Test statistigi : 0.8253953456878662
p-value : 0.00793540757149458
```

```
hg = bogusuz.high
bg1,bg2 = stats.shapiro(hg)
print("Test statistigi : ",bg1)
print("p-value : ", bg2)
```

[8] ✓ 0.0s

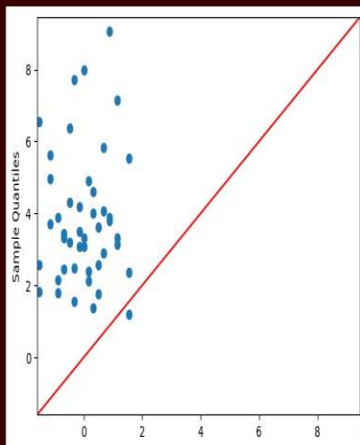
Python

```
... Test statistigi : 0.8495509020434753
p-value : 0.01711362786591053
```

```
sm.qqplot(bogusuz, line='45')
plt.title("")
plt.xlabel("")
plt.ylabel("Sample Quantiles")
plt.show()
```

[9] ✓ 0.2s

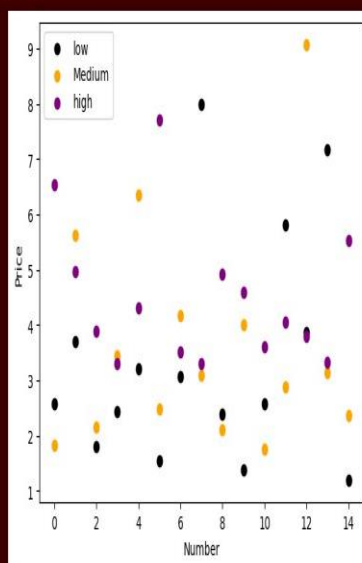
Python



```
plt.scatter(range(len(bogusuz.low)),bogusuz.low,c='black',label='low')
plt.scatter(range(len(bogusuz.medium)),bogusuz.medium,c='orange',label='Medium')
plt.scatter(range(len(bogusuz.high)),bogusuz.high,c='purple',label='high')
plt.xlabel('Number')
plt.ylabel('Price')
plt.legend()
plt.show()
```

[10] ✓ 0.2s

Python



```
Number = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
```

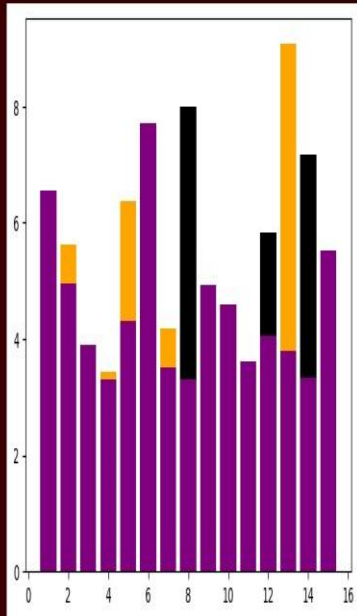
```

Number = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
plt.bar(Number,bogusize.low, color = "black")
plt.bar(Number,bogusize.medium, color = "orange")
plt.bar(Number,bogusize.high, color = "purple")
plt.show()

```

✓ 0.1s

Python



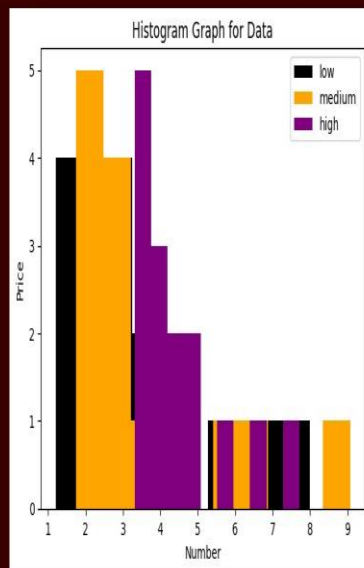
```

plt.hist(bogusize.low, color="black", label = 'low')
plt.hist(bogusize.medium,color="orange", label = 'medium')
plt.hist(bogusize.high,color="purple", label = 'high')
plt.legend()
plt.title('Histogram Graph for Data')
plt.xlabel('Number')
plt.ylabel('Price')
plt.show()

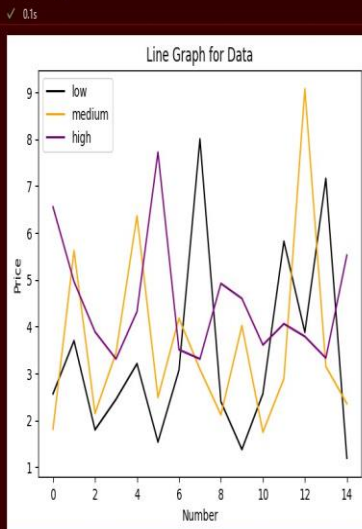
```

12) ✓ 0.1s

Python

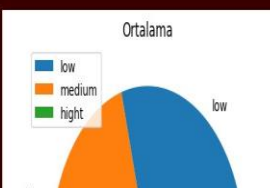
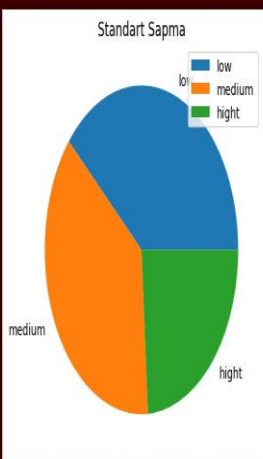


```
plt.plot(bogusuz_low,color="black",label="low")
plt.plot(bogusuz_medium,color="orange",label="medium")
plt.plot(bogusuz_high,color="purple",label="high")
plt.legend()
plt.title('Line Graph for Data')
plt.xlabel('Number')
plt.ylabel('Price')
plt.show()
```



```
y = [2.067337, 2.013525, 1.284882]
lb = ["low", "medium", "high"]
plt.title('Standart Sapma')
plt.pie(y, labels=lb)
plt.legend()
plt.show()

x = [3.383333, -3.634000, -4.483333]
lb = ["low", "medium", "high"]
plt.title('Ortalama')
plt.pie(x, labels=lb)
plt.legend()
plt.show()
```



```
e=hg.median()
from scipy.stats import wilcoxon
wilcoxon(hg - e,
         alternative = 'less',
         method = 'approx',
         zero_method = 'split',
         correction = True)
```

✓ 0.0s

Python

```
WilcoxonResult(statistic=74.5, pvalue=0.8029247458745191)
```

```
from scipy.stats import binomtest
nPlus = len(hg[hg > e])
nNeg = len(hg[hg < e])
n = nPlus + nNeg
binomtest(nPlus, n, p=0.5, alternative='less')
```

✓ 0.0s

Python

```
BinomTestResult(k=7, n=14, alternative='less', statistic=0.5, pvalue=0.604736328125)
```

[+ Code](#) [+ Markdown](#)

```
group = [Lw,md,hg]
groupname = ['düşük', 'orta', 'yüksek']
comb = (zip(group,groupname))
for grup1 , grup2 in combinations(comb , 2 ):
    md , mp =stats.mannwhitneyu(grup1[0],grup2[0])
    print(f'{grup1[1]} ve {grup2[1]} :")
    print("istatistiksel değer :", md)
    print("p-value: \n", mp)
```

✓ 0.0s

Python

```
düşük ve orta :
istatistiksel değer : 100.0
p-value:
0.6186385224456992
düşük ve yüksek :
istatistiksel değer : 53.0
p-value:
0.014375111660888815
orta ve yüksek :
istatistiksel değer : 59.0
p-value:
```

```
stats.kruskal(Lw,md,hg)
```

[17] ✓ 0.0s

Python

```
... KruskalResult(statistic=7.589019194448124, pvalue=0.022493934200113103)
```

```
sp.posthoc_dunn(group, p_adjust = 'bonferroni')
```

[14] ✓ 0.0s

Python

```
...
   1   2   3
1  1.000000  1.000000  0.030347
2  1.000000  1.000000  0.096849
3  0.030347  0.096849  1.000000
```

```
bagiml = pd.read_excel(r'C:\Users\pc\Desktop\anpar veriler\bagiml.xls')
```

	Ülke	İkbinonbeş	İkbinondokuz	İkibinyirmüç
0	Almanya	14.49	16.61	21.14
1	ABD	7.01	8.97	16.88
2	Irak	9.96	10.22	13.75
3	Birleşik Krallık	10.82	11.27	13.00
4	İtalya	7.15	9.75	12.38
5	İspanya	4.94	8.13	9.65
6	Fransa	6.10	7.94	9.53
7	Rusya Federasyonu	3.68	4.15	9.34
8	Hollanda	3.35	5.76	8.02
9	İsrail	2.80	4.46	7.03
10	Romanya	2.92	4.07	6.94
11	Polonya	2.42	3.44	5.17
12	BAE	4.93	3.62	5.25
13	Belçika	2.72	3.39	4.77
14	Bulgaristan	1.76	2.66	4.72
15	Mısır	3.24	3.50	4.55
16	Yunanistan	1.48	2.24	3.30
17	Çin	2.50	2.72	3.28
18	Fas	1.37	2.34	3.09
19	İran	4.11	2.73	3.06

```
bagiml.describe()
```

	İkbinonbeş	İkbinondokuz	İkibinyirmüç
count	20.000000	20.000000	20.000000
mean	4.867500	5.898500	8.242500
std	3.478148	3.867054	5.016048
min	1.370000	2.240000	3.060000
25%	2.665000	3.225000	4.677500
50%	3.515000	4.110000	6.985000
75%	6.327500	8.340000	10.332500
max	14.490000	16.610000	21.140000

```
onbes = bagiml.İkbinonbeş  
ondokuz = bagiml.İkbinondokuz  
yirmüç = bagiml.İkibinyirmüç  
skew(onbes, axis=0, bias=True)
```

```
skew(ondokuz, axis=0, bias=True)
```

```
skew(yirmüç, axis=0, bias=True)
```

```
kurtosis(onbes, axis=0, bias=True)
```

```
kurtosis(ondokuz, axis=0, bias=True)
[26] ✓ 0.0s Python
... 0.8425518092387398

kurtosis(yirmiüç, axis=0, bias=True)
[27] ✓ 0.0s Python
... 0.3482285826895639

ob= stats.shapiro(onbes)
ob
[28] ✓ 0.0s Python
... ShapiroResult(statistic=0.8419543504714966, pvalue=0.003919442184269428)

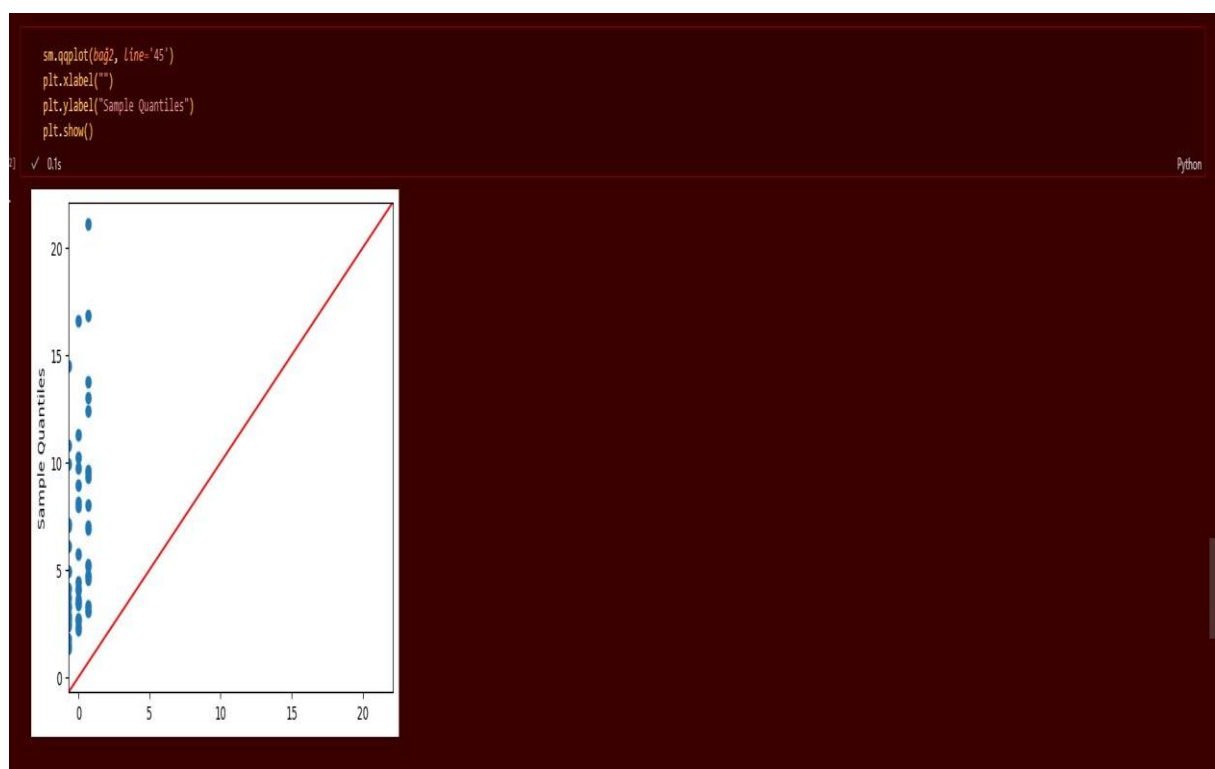
od = stats.shapiro(ondokuz)
od
[29] ✓ 0.0s Python
... ShapiroResult(statistic=0.8368551595687866, pvalue=0.003140856738629937)

yü = stats.shapiro(yirmiüç)
yü
[30] ✓ 0.0s Python
... ShapiroResult(statistic=0.8851362466812134, pvalue=0.0219122381787138)

bag1 = [onbes,ondokuz,yirmiüç]
bag2 = pd.DataFrame(bag1)
bag2
[31] ✓ 0.0s Python
...

```

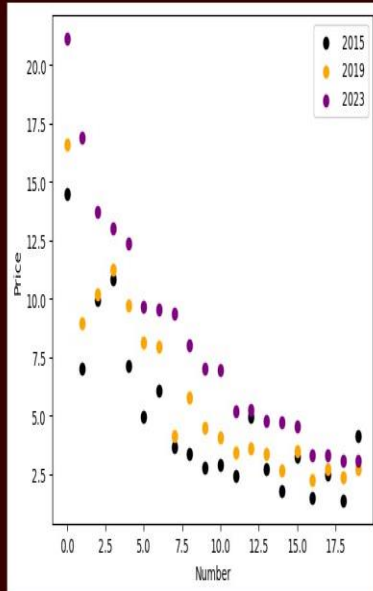
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
ikibinonbes	14.49	7.01	9.96	10.82	7.15	4.94	6.10	3.68	3.35	2.80	2.92	2.42	4.93	2.72	1.76	3.24	1.48	2.50	1.37	4.11
ikibinondokuz	16.61	8.97	10.22	11.27	9.75	8.13	7.94	4.15	5.76	4.46	4.07	3.44	3.62	3.39	2.66	3.50	2.24	2.72	2.34	2.73
ikibinyirmiüç	21.14	16.88	13.75	13.00	12.38	9.65	9.53	9.34	8.02	7.03	6.94	5.17	5.25	4.77	4.72	4.55	3.30	3.28	3.09	3.06



```
plt.scatter(range(len(bogunlu_ikibinbes)),bogunlu_ikibinbes,c= 'black', label = '2015')
plt.scatter(range(len(bogunlu_ikibinondokuz)),bogunlu_ikibinondokuz,c= 'orange', label = '2019')
plt.scatter(range(len(bogunlu_ikibinyirmiuc)),bogunlu_ikibinyirmiuc,c= 'purple', label = '2023')
plt.xlabel('Number')
plt.ylabel('Price')
plt.legend()
plt.show()
```

[39] ✓ 0.2s

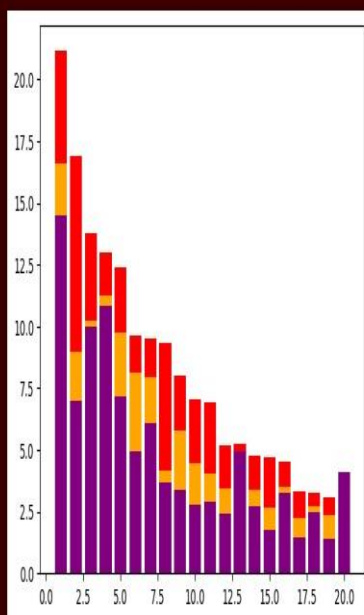
Python



```
Number = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]
plt.bar(Number,yirmiuc, color = "red")
plt.bar(Number,ondokuz, color = "orange")
plt.bar(Number,onbes, color = "purple")
plt.show()
```

[4] ✓ 0.1s

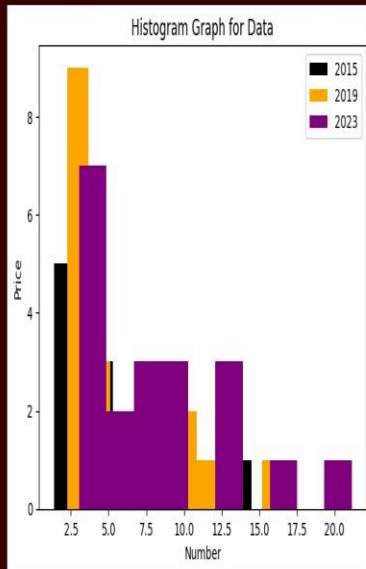
Python



```
plt.hist(onbes, color="black", label = '2015')
plt.hist(ondokuz, color="orange", label = '2019')
plt.hist(yirmiüç, color="purple", label = '2023')
plt.legend()
plt.title('Histogram Graph for Data')
plt.xlabel('Number')
plt.ylabel('Price')
plt.show()
```

✓ 0.1s

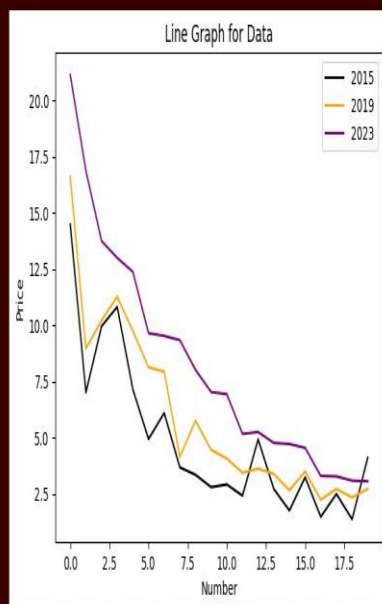
Python



```
plt.plot(onbes, color="black", label = '2015')
plt.plot(ondokuz, color="orange", label = '2019')
plt.plot(yirmiüç, color="purple", label = '2023')
plt.legend()
plt.title('Line Graph for Data')
plt.xlabel('Number')
plt.ylabel('Price')
plt.show()
```

✓ 0.1s

Python



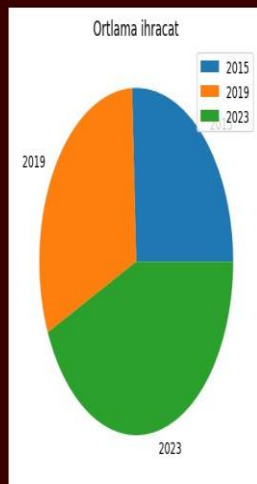


```

y = [4.887588, 5.898588, 8.242588]
lb = ["2015", "2019", "2023"]
plt.title("Ortlama ihracat")
plt.pie(y, labels =lb )
plt.legend()
plt.show()

x = [3.478148, 3.867854, 5.816848]
lb = ["2015", "2019", "2023"]
plt.title("Ortalama")
plt.pie(x, labels =lb )
plt.legend()
plt.show()

```



```

##wilcoxon testi
stats.wilcoxon(ondokuz, yirmiüç,
alternative='greater',
method = 'approx',
zero_method = 'zsplit',
correction = True)

```

✓ 0.0s

WilcoxonResult(statistic=0.0, pvalue=0.9999590282182997)

```
stats.friedmanchisquare(ondes,ondokuz,yirmiüç)
```

✓ 0.0s

FriedmanchisquareResult(statistic=32.69999999999999, pvalue=7.938219729987684e-80)

[+ Code](#) [+ Markdown](#)

```
data1 = np.array(hoj2)
sp.posthoc_nemenyi_friedman(hoj2)
```

✓ 0.4s

Python

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	1.000000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.854289	0.763186	0.336284	0.899837	0.336284	0.102152	0.489674	0.034287	0.102152	0.021016	0.291376
1	0.900000	1.000000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.763186	0.900000	0.763186	0.440145	0.899837	0.212548	0.440145	0.149700	0.717635
2	0.900000	0.900000	1.000000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.626535	0.900000	0.626535	0.291376	0.763186	0.124999	0.291376	0.083958	0.580983
3	0.900000	0.900000	0.900000	1.000000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.580983	0.900000	0.580983	0.249612	0.717635	0.102152	0.249612	0.067895	0.535431
4	0.900000	0.900000	0.900000	0.900000	1.000000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.808736	0.900000	0.808736	0.489674	0.900000	0.249612	0.489674	0.178781	0.763186
5	0.900000	0.900000	0.900000	0.900000	0.900000	1.000000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.763186	0.900000	0.535431	0.763186	0.440145	0.900000
6	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	1.000000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.808736	0.900000	0.580983	0.808736	0.489674	0.900000
7	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	1.000000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.854289	0.900000
8	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	1.000000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.854289	0.900000
9	0.854289	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	1.000000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000
10	0.763186	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	1.000000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000
11	0.336284	0.763186	0.626535	0.580983	0.808736	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	1.000000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000
12	0.899837	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	1.000000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000
13	0.336284	0.763186	0.626535	0.580983	0.808736	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	1.000000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000
14	0.102152	0.440145	0.291376	0.249612	0.489674	0.763186	0.808736	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	1.000000	0.900000	0.900000	0.900000	0.900000	0.900000
15	0.489674	0.899837	0.763186	0.717635	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	1.000000	0.900000	0.900000	0.900000	0.900000
16	0.034287	0.212548	0.124999	0.102152	0.249612	0.535431	0.580983	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	1.000000	0.900000	0.900000	0.900000
17	0.102152	0.440145	0.291376	0.249612	0.489674	0.763186	0.808736	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	1.000000	0.900000
18	0.021016	0.149700	0.083958	0.067895	0.178781	0.440145	0.489674	0.854289	0.854289	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	1.000000	0.900000
19	0.291376	0.717635	0.580983	0.535431	0.763186	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	1.000000

```
sp.posthoc_dunn(data1, p_adjust = 'bonferroni')
```

✓ 0.0s

Python

	1	2	3
1	1.000000	1.000000	0.026298
2	1.000000	1.000000	0.268768
3	0.026298	0.268768	1.000000

```
test = pd.read_excel(r"C:\Users\pc\Desktop\nanpar veriler\test.xlsx")
test
```

✓ 0.0s

Python

	gün	derece
0	2009-02-01	32.6
1	2009-02-02	24.4
2	2009-02-03	21.5
3	2009-02-04	21.7
4	2009-02-05	21.5
5	2009-02-06	23.5
6	2009-02-07	22.3
7	2009-02-08	28.3
8	2009-02-09	18.4
9	2009-02-10	14.9
10	2009-02-11	13.5
11	2009-02-12	16.1
12	2009-02-13	14.6
13	2009-02-14	12.4
14	2009-02-15	13.3
15	2009-02-16	17.2
16	2009-02-17	12.5
17	2009-02-18	18.0
18	2009-02-19	16.2
19	2009-02-20	18.7
20	2009-02-21	13.7
21	2009-02-22	15.5
22	2009-02-23	14.3
23	2009-02-24	12.9
24	2009-02-25	8.9
25	2009-02-26	15.0
26	2009-02-27	15.4
27	2009-02-28	16.0

```
import pymannkendall as mk
test1 = np.array(test.derece)
mk.original_test(test1)
```

✓ 0.0s Python

Mann\_Kendall\_Test(trend='decreasing', h=True, p=0.0006768596985562425, z=-3.3987047243535483, Tau=-0.4576719576719577, s=-173.0, var\_s=2561.0, slope=-0.3980909090909091, intercept=21.327272727272728)

+ Code + Markdown

## Kaynakça

- <https://data.tuik.gov.tr/Bulten/Index?p=Dis-Ticaret-Istatistikleri-Ekim-2023-49628#:~:text=Ekim%20ay%C4%B1nda%20ihracatta%20ilk%20s%C4%B1ray%C4%B1,dolar%20ile%20%C4%B0talya%20takip%20etti>. Ülkerin yıllara göre ihracat tutarlarını temsil eden verilerin alınması.
- <https://www.kaggle.com/datasets/yakhyojon/marketing-promotion> TV kanallarının reklam bütçelerine ait verilen alındığı alan.
- <https://medium.com/datarunner/matplotlibkutuphanesi-1-99087692102b> Python grafiklerinin çizdirilmesine dair destek alınan site.
- <https://miracozturk.com/python-kutuphaneleri-ve-ozellikleri/> Python kütüphanelerine ait bilgilerin alındığı alan.