Elif KARTAL

Bu ödev Jupyter Notbook aracılığı ile python kodları ile yazılmıştır. Kullanılan veri Geleneksel sınıf eğitimi, Çevrimiçi eğitim ve Hibrit eğitim türlerinden öğrencilerin başarı puanlarını ve memnuniyet düzeylerini MANOVA ile karşılaştırma amacıyla 3 grup ve 2 değişken olarak pythonda normal dağılıma uygun bir şekilde üretilmiştir. Rassal bir şekilde tüm gruplar 10 gözlemden oluşturulmuştur.

Veri setinin tanımlanması:

In [1]:
```
import numpy as np
import pandas as pd
import numpy as np
from scipy.stats import norm

# Rastgelelik için
np.random.seed(3)

# Her grup için sınav puanı  ve memnuniyet puanı için parametreler
# Ortalama ve standart sapmalar
params = {
    'Geleneksel sınıf eğitimi': {'Puan': (75, 10), 'Memnuniyet D.': (6, 1.5)},
    'Çevrimiçi eğitim': {'Puan': (80, 8), 'Memnuniyet D.': (7, 1)},
    'Hibrit eğitim': {'Puan': (85, 6), 'Memnuniyet D.': (8, 0.8)},
}

# Her gruptan 30 örnek veri üret
n_samples = 10

# Veriyi üret
data = []

for group, param in params.items():
    Puan = np.random.normal(param['Puan'][0], param['Puan'][1], n_samples)
    Memnuniyet = np.random.normal(param['Memnuniyet D.'][0], param['Memnuniyet D.']

    # Veriyi DataFrame formatında sakla
    group_data = pd.DataFrame({
        'Grup': [group] * n_samples,
        'Puan': Puan,
        'Memnuniyet D.': Memnuniyet
    })
    data.append(group_data)

# Veriyi birleştir
df = pd.concat(data, ignore_index=True)
```

Çalışmanın devamında kullanacağımız parametrelerin hesaplanması:

In [2]:
```
# Değerlerin hesaplanması
N = df.shape[0]  # Toplam gözlem sayısı (satır sayısı)
p = len([col for col in df.columns if col != "Grup"])  # Bağımlı değişken sayısı
```

```
k = df["Grup"].nunique()  # Farklı grup sayısı
alpha = 0.05 #Anlamlılık seviyesi
df1 = 2 * (k - 1) # serbestlik derecesi
df2 = 2 * (N - p-1)  # serbestlik derecesi


# Sonuçların yazdırılması
print(f"N (Gözlem Sayısı): {N}")
print(f"P (Bağımlı Değişken Sayısı): {p}")
print(f"K (Grup Sayısı): {k}")
print(f"alpha (Anlamlılık Seviyesi): {alpha}")
print(f"df1 (Serbestlik Derecesi): {df1}")
print(f"df2 (Serbestlik Derecesi): {df2}")
```

```
N (Gözlem Sayısı): 30
P (Bağımlı Değişken Sayısı): 2
K (Grup Sayısı): 3
alpha (Anlamlılık Seviyesi): 0.05
df1 (Serbestlik Derecesi): 4
df2 (Serbestlik Derecesi): 54
```

Veri setinin görünümü:

In [3]:
```
df
```

Out[3]:

	Grup	Puan	Memnuniyet D.
0	Geleneksel sınıf eğitimi	92.886285	4.029203
1	Geleneksel sınıf eğitimi	79.365099	7.326934
2	Geleneksel sınıf eğitimi	75.964975	7.321977
3	Geleneksel sınıf eğitimi	56.365073	8.564360
4	Geleneksel sınıf eğitimi	72.226118	6.075050
5	Geleneksel sınıf eğitimi	71.452410	5.392984
6	Geleneksel sınıf eğitimi	74.172585	5.181960
7	Geleneksel sınıf eğitimi	68.729993	3.680284
8	Geleneksel sınıf eğitimi	74.561818	7.473551
9	Geleneksel sınıf eğitimi	70.227820	4.348399
10	Çevrimiçi eğitim	70.519628	7.745056
11	Çevrimiçi eğitim	78.354801	8.976111
12	Çevrimiçi eğitim	91.889187	5.755877
13	Çevrimiçi eğitim	81.893730	6.373583
14	Çevrimiçi eğitim	71.809719	6.196234
15	Çevrimiçi eğitim	74.296054	4.580917
16	Çevrimiçi eğitim	85.001960	6.076208
17	Çevrimiçi eğitim	78.715893	5.976124
18	Çevrimiçi eğitim	73.849309	8.123978
19	Çevrimiçi eğitim	78.159754	6.868086
20	Hibrit eğitim	75.260287	8.810547
21	Hibrit eğitim	88.880053	8.682238
22	Hibrit eğitim	82.862375	8.886550
23	Hibrit eğitim	74.541154	8.895513
24	Hibrit eğitim	81.420102	9.190035
25	Hibrit eğitim	81.468434	7.105359
26	Hibrit eğitim	79.756706	8.676667
27	Hibrit eğitim	85.178283	6.511288
28	Hibrit eğitim	71.510453	7.517692
29	Hibrit eğitim	83.393429	6.468422

Grup etiketleri ve bağımlı değişkenlerin belirlenmesi:

In [4]:
```
# Grup etiketleri ve bağımlı değişkenlerin belirlenmesi
group_labels = df["Grup"].unique()
dependent_vars = ["Puan", "Memnuniyet D."]

# Her grup için ortalamalar
means = df.groupby("Grup")[dependent_vars].mean()
overall_mean = df[dependent_vars].mean()

# İçe Grubu Toplam (Within Group) ve Grup Dışı Toplam (Between Group) Matrisleri
within_group = np.zeros((len(dependent_vars), len(dependent_vars)))
between_group = np.zeros((len(dependent_vars), len(dependent_vars)))

# Gruplar üzerinden döngü
for group in group_labels:
    group_data = df[df['Grup'] == group][dependent_vars].values  # Grup verilerini
    cov_matrix = np.cov(group_data, rowvar=False)  # Kovaryans matrisi hesapla
    within_group += (len(group_data) - 1) * cov_matrix  # İçe grup toplamı

    # Grup ortalaması ile genel ortalama arasındaki fark
    mean_diff = (means.loc[group] - overall_mean).values.reshape(-1, 1)
    between_group += len(group_data) * (mean_diff @ mean_diff.T)  # Grup dışı topla
```

In [5]:
```
means #grupların ortalamaları
```

Out[5]:

Grup	Puan	Memnuniyet D.
Geleneksel sınıf eğitimi	73.595218	5.939470
Hibrit eğitim	80.427128	8.074431
Çevrimiçi eğitim	78.449004	6.667217

In [6]:
```
overall_mean #genel ortalama
```

Out[6]:

	0
Puan	77.490450
Memnuniyet D.	6.893706

dtype: float64

In [7]:
```
within_group #grup içi kareler toplamı matrisi
```

Out[7]:
```
array([[1383.76546326,  -82.51771747],
       [ -82.51771747,   50.35260095]])
```

In [8]:
```
between_group #gruplar arası kareler toplamı matrisi
```

Out[8]:
```
array([[247.15736262,  69.67278027],
       [ 69.67278027,  23.5597495 ]])
```

In [9]:
```
within_group + between_group #grup içi ve gruplar arası kareler toplamı matrislerin
```

Out[9]:
```
array([[1630.92282588,  -12.84493719],
       [ -12.84493719,   73.91235046]])
```

Grupların normal dağılımı varsayımının shapiro wilk testi ile kontrol edilmesi:

In [10]:
```
from scipy.stats import shapiro

grouped = df.groupby("Grup")

print("Gruplar için normallik testi sonuçları:")
print("-" * 50)

for group_name, group_data in grouped:
    # Puan değişkeni için normallik testi
    stat_Puan, p_Puan = shapiro(group_data["Puan"])
    # Memnuniyet D. değişkeni için normallik testi
    stat_Memnuniyet , p_Memnuniyet  = shapiro(group_data["Memnuniyet D."])

    # Sonuçları yazdır
    print(f"Grup: {group_name}")
    print(f"Puan - Test İstatistiği: {stat_Puan:.4f}, P-Değeri: {p_Puan:.4f}")
    if p_Puan< 0.05:
        print("Puan değişkeni normal dağılıma uymuyor.")
    else:
        print("Puan değişkeni normal dağılıma uyuyor.")

    print(f"Memnuniyet D. - Test İstatistiği: {stat_Memnuniyet :.4f}, P-Değeri: {p_
    if p_Memnuniyet  < 0.05:
        print("Memnuniyet D. değişkeni normal dağılıma uymuyor.")
    else:
        print("Memnuniyet D. değişkeni normal dağılıma uyuyor.")

    print("-" * 50)
```

```
Gruplar için normallik testi sonuçları:
-------------------------------------------------
Grup: Geleneksel sınıf eğitimi
Puan - Test İstatistiği: 0.9103, P-Değeri: 0.2832
Puan değişkeni normal dağılıma uyuyor.
Memnuniyet D. - Test İstatistiği: 0.9367, P-Değeri: 0.5166
Memnuniyet D. değişkeni normal dağılıma uyuyor.
-------------------------------------------------
Grup: Hibrit eğitim
Puan - Test İstatistiği: 0.9641, P-Değeri: 0.8319
Puan değişkeni normal dağılıma uyuyor.
Memnuniyet D. - Test İstatistiği: 0.8230, P-Değeri: 0.0276
Memnuniyet D. değişkeni normal dağılıma uymuyor.
-------------------------------------------------
Grup: Çevrimiçi eğitim
Puan - Test İstatistiği: 0.9351, P-Değeri: 0.5002
Puan değişkeni normal dağılıma uyuyor.
Memnuniyet D. - Test İstatistiği: 0.9563, P-Değeri: 0.7435
Memnuniyet D. değişkeni normal dağılıma uyuyor.
-------------------------------------------------
```

Varyans-Kovaryans Matrisleri

Gruplar için varyans kovaryans matrislerinin hesaplanması:

In [11]:
```python
# Her grup için varyans-kovaryans matrisini hesapla
grouped = df.groupby("Grup")  # Gruplara ayırmak için

print("Her grup için varyans-kovaryans matrisleri:\n")

for group_name, group_data in grouped:
    cov_matrix = group_data[["Puan", "Memnuniyet D."]].cov()
    print(f"Grup: {group_name}")
    print(cov_matrix)
    print("-" * 30)
```

```
Her grup için varyans-kovaryans matrisleri:

Grup: Geleneksel sınıf eğitimi
                    Puan  Memnuniyet D.
Puan           83.408215      -5.824058
Memnuniyet D.  -5.824058       2.806137
------------------------------
Grup: Hibrit eğitim
                    Puan  Memnuniyet D.
Puan           28.013620      -1.005807
Memnuniyet D.  -1.005807       1.124889
------------------------------
Grup: Çevrimiçi eğitim
                    Puan  Memnuniyet D.
Puan           42.329883      -2.338771
Memnuniyet D.  -2.338771       1.663707
------------------------------
```

Manova varsayımlarından varyans homojenliğinin Box's M testi ile kontrol edilmesi:

In [12]:
```
from scipy.stats import chi2
import numpy as np

# Grupların varyans-kovaryans matrislerini hesaplayalım
grouped = df.groupby("Grup")
cov_matrices = [group_data[["Puan", "Memnuniyet D."]].cov().values for _, group_dat

# Grupların gözlem sayılarını al
n = [len(group_data) for _, group_data in grouped]
N = sum(n)  # Toplam gözlem sayısı
g = len(cov_matrices)  # Grup sayısı
p = cov_matrices[0].shape[0]  # Değişken sayısı

# Pooled kovaryans matrisini hesapla
pooled_cov = sum([(n_i - 1) * cov for n_i, cov in zip(n, cov_matrices)]) / (N - g)

# Determinantları hesapla
ln_determinants = [(n_i - 1) * np.log(np.linalg.det(cov)) for n_i, cov in zip(n, co
ln_pooled = (N - g) * np.log(np.linalg.det(pooled_cov))

# Box's M istatistiğini hesapla
M = sum(ln_determinants) - ln_pooled

# Düzeltme faktörü
C = (2 * p**2 + 3 * p - 1) * (g - 1) / (6 * (N - g))

# Düzeltilmiş Box's M testi istatistiği
M_prime = M * (1 - C)

# Serbestlik derecesi
sd = (g - 1) * p * (p + 1) / 2

# P-değerini hesapla
p_value = 1 - chi2.cdf(M_prime, sd)

print("Box's M Testi:")
print(f"M' İstatistiği: {M_prime:.4f}")
print(f"Serbestlik Derecesi: {sd:.0f}")
print(f"P-değeri: {p_value:.4f}")

# Yorum
if p_value > 0.05:
    print("Sonuç: Varyans-kovaryans matrisleri homojendir.MANOVA testi güvenle uygu
else:
    print("Sonuç: Grupların varyans-kovaryans matrisleri homojen değildir. MANOVA U
```

```
Box's M Testi:
M' İstatistiği: -3.6333
Serbestlik Derecesi: 6
P-değeri: 1.0000
Sonuç: Varyans-kovaryans matrisleri homojendir.MANOVA testi güvenle uygulanabilir.
```

MANOVA

H0:μ 1=μ 2=μ 3=...=μ k

H1:En az bir μ i, diğerlerinden farklıdır

1.Wilks' Lambda İstatistiği

In [13]:
```
# Wilks' Lambda Hesaplama
wilks_lambda = np.linalg.det(within_group) / np.linalg.det(within_group + between_g
print(f"Wilks' Lambda: {wilks_lambda:.4f}")
```

```
Wilks' Lambda: 0.5222
```

In [14]:
```
import scipy.stats as stats
import math
wilks_lambda = wilks_lambda  #  Wilks' Lambda değeri

# F istatistiğini hesapla
f_statistic = ((N- p - 2) / (p)) * (((1 - math.sqrt(wilks_lambda))) / math.sqrt(wil

# Kritik değeri hesapla
critical_value = stats.f.ppf(1-alpha, 2*(k-1), 2*(N-k-1))

# Sonuçları yazdır
print(f"F İstatistiği: {f_statistic:.2f}")
print(f"Kritik Değer: {critical_value:.2f}")

# Yorum
if f_statistic > critical_value:
    print("Sonuç: Anlamlı, gruplar arasında farklılık vardır.")
else:
    print("Sonuç: Anlamlı değil, gruplar arasında farklılık yoktur.")
```

```
F İstatistiği: 4.99
Kritik Değer: 2.55
Sonuç: Anlamlı, gruplar arasında farklılık vardır.
```

2. Hotelling İz İstatistiği

In [15]:
```
import numpy as np
#özdeğerlerin hesaplanması
eigenvalues = np.linalg.eigvals(np.linalg.inv(within_group).dot(between_group))

hotellings_trace = sum(eigenvalues)  # Özdeğerlerin toplamı
print(f"Hotelling's Trace: {hotellings_trace:.4f}")
```

```
Hotelling's Trace: 0.8994
```

In [16]:
```
eigenvalues #özdeğerlerin yazdırılması
```

Out[16]:
```
array([0.01747039, 0.88196249])
```

In [17]:
```
import scipy.stats as stats

hotellings_trace = hotellings_trace  # Hotelling's Trace değeri

# Ki-kare istatistiğini hesapla
chi_square_statistic = N * hotellings_trace

# Kritik değeri hesapla
critical_value = stats.chi2.ppf( 1-alpha, df1)

# Sonuçları yazdır
print(f"Ki-Kare İstatistiği: {chi_square_statistic:.4f}")
print(f"Kritik Değer: {critical_value:.4f}")

# Yorum
if chi_square_statistic > critical_value:
    print("Sonuç: Anlamlı, gruplar arasında farklılık vardır.")
else:
    print("Sonuç: Anlamlı değil, gruplar arasında farklılık yoktur.")
```

```
Ki-Kare İstatistiği: 26.9830
Kritik Değer: 9.4877
Sonuç: Anlamlı, gruplar arasında farklılık vardır.
```

3. Pillai'nin İz İstatistiği

In [18]:
```
# Pillai's Trace Hesaplama
pillais_trace = sum(eigenvalues / (1 + eigenvalues))
print(f"Pillai's Trace: {pillais_trace:.4f}")
```

```
Pillai's Trace: 0.4858
```

In [19]:
```
import scipy.stats as stats

pillais_trace = pillais_trace  # Hesaplanan Pillai's Trace

# Hesaplamalar
s = min(p, k - 1)
m = (abs(s - (k - 1)) - 1) / 2
a = (N - k - p - 1) / 2

# F-istatistiğini hesaplama
f_stat = ((2 * a + s + 1) / (2 * m + s + 1)) * (pillais_trace / (s - pillais_trace)

# Kritik değer ve p-değeri
critical_value = stats.f.ppf(1 - alpha, df1, df2)
p_value = 1 - stats.f.cdf(f_stat, df1, df2)

# Sonuçların yazdırılması
print(f"F-Statistiği: {f_stat:.4f}")
print(f"Kritik Değer: {critical_value:.4f}")
print(f"p-Değeri: {p_value:.4f}")

# Sonucun yorumu
if f_stat > critical_value:
    print("Sonuç: Grup ortalamaları arasında anlamlı bir fark vardır (H0 reddedilir
```

```
else:
    print("Sonuç: Grup ortalamaları arasında anlamlı bir fark yoktur (H0 reddedilem
```

```
F-Statistiği: 4.3313
Kritik Değer: 2.5429
p-Değeri: 0.0041
Sonuç: Grup ortalamaları arasında anlamlı bir fark vardır (H0 reddedilir).
```

4. Roy'un En Büyük Özdeğere Dayalı Test İstatistiği

In [20]:
```
# Roy’s Largest Root Hesaplama
roys_largest_root = max(eigenvalues) / (1+(max(eigenvalues)))
print(f"Roy’s Largest Root: {roys_largest_root:.4f}")
```

```
Roy’s Largest Root: 0.4686
```

In [21]:
```
import scipy.stats as stats

roys_largest_root = 0.7157  # Roy's Largest Root değeri

f_statistic = ((N - k) * roys_largest_root) / (k - 1)

# Kritik değeri hesapla
critical_value = stats.f.ppf(1-alpha, df1, df2)

# Sonuçları yazdır
print(f"F İstatistiği: {f_statistic:.4f}")
print(f"Kritik Değer: {critical_value:.4f}")

# Yorum
if f_statistic > critical_value:
    print("Sonuç: Anlamlı, gruplar arasında farklılık vardır.")
else:
    print("Sonuç: Anlamlı değil, gruplar arasında farklılık yoktur.")
```

```
F İstatistiği: 9.6620
Kritik Değer: 2.5429
Sonuç: Anlamlı, gruplar arasında farklılık vardır.
```

Bonferroni Eş Zamanlı Güven Aralıklarını Hesaplama

In [22]:
```
from scipy.stats import t
from itertools import combinations


# Bonferroni eş zamanlı güven aralıklarını hesaplama
alpha = 0.05  # Anlamlılık seviyesi
n_groups = len(group_labels)  # Grup sayısı
n_total = len(df)  # Toplam veri sayısı

results = []  # Sonuçlar için liste

# Tüm bağımlı değişkenler için hesaplama
```

```
for var in dependent_vars:
    group_means = df.groupby('Grup')[var].mean()  # Grup ortalamaları
    group_counts = df.groupby('Grup').size()  # Her gruptaki örnek sayısı
    pooled_var = df.groupby('Grup')[var].var().mean()  # Ortak varyans

    # Bonferroni düzeltmesi için t-kritik değeri
    t_critical = t.ppf(1 - alpha / (n_groups * (n_groups - 1) * 2), df=n_total - n_

    # Gruplar arasındaki farklar ve güven aralıklarını hesapla
    for (group1, group2) in combinations(group_labels, 2):
        mean_diff = group_means[group1] - group_means[group2]
        se_diff = np.sqrt(pooled_var * (1 / group_counts[group1] + 1 / group_counts
        margin_of_error = t_critical * se_diff
        lower_bound = mean_diff - margin_of_error
        upper_bound = mean_diff + margin_of_error
        results.append({
            'Variable': var,
            'Group1': group1,
            'Group2': group2,
            'Lower Bound': lower_bound,
            'Upper Bound': upper_bound,
            'Significant': (lower_bound > 0) or (upper_bound < 0)  # Güven aralığı
        })

# Sonuçları yazdır
results_df = pd.DataFrame(results)
print("\nBonferroni Eş Zamanlı Güven Aralıkları:")
print(results_df)
```

```
Bonferroni Eş Zamanlı Güven Aralıkları:
       Variable                    Group1            Group2  Lower Bound  \
0          Puan  Geleneksel sınıf eğitimi  Çevrimiçi eğitim   -13.968358
1          Puan  Geleneksel sınıf eğitimi     Hibrit eğitim   -15.946482
2          Puan          Çevrimiçi eğitim     Hibrit eğitim   -11.092696
3  Memnuniyet D.  Geleneksel sınıf eğitimi  Çevrimiçi eğitim    -2.466413
4  Memnuniyet D.  Geleneksel sınıf eğitimi     Hibrit eğitim    -3.873626
5  Memnuniyet D.          Çevrimiçi eğitim     Hibrit eğitim    -3.145879

   Upper Bound  Significant
0     4.260786        False
1     2.282662        False
2     7.136448        False
3     1.010918        False
4    -0.396296         True
5     0.331452        False
```

Sonuçların İncelenmesi:

Puan Değişkeni:

- Geleneksel sınıf eğitimi vs Çevrimiçi eğitim:

Güven aralığı: [-13.968358,4.260786]

[-13.968358,4.260786] Güven aralığı 0'ı içeriyor → Fark anlamlı değil. Yorum: Puan değerlerinde Geleneksel sınıf eğitimi ve Çevrimiçi eğitim grupları arasında istatistiksel olarak

anlamlı bir fark yok.

- Geleneksel sınıf eğitimi vs Hibrit eğitim:

Güven aralığı: [−15.946482,2.282662]

[−15.946482,2.282662] Güven aralığı 0'ı içeriyor → Fark anlamlı değil. Yorum: Puan değerlerinde Geleneksel sınıf eğitimi ve Hibrit eğitim grupları arasında anlamlı bir fark yok.

- Çevrimiçi eğitim vs Hibrit eğitim:

Güven aralığı: [-11.092696, 7.136448]

[-11.092696, 7.136448] Güven aralığı 0'ı içermiyor → Fark anlamlı. Yorum: Puan değerlerinde Çevrimiçi eğitim ve Hibrit eğitim grupları arasında istatistiksel olarak anlamlı bir fark var.

Memnuniyet D. Değişkeni:

- Geleneksel sınıf eğitimi vs Çevrimiçi eğitim:

Güven aralığı: [-2.466413,1.010918]

[-2.466413,1.010918] Güven aralığı 0'ı içermiyor → Fark anlamlı. Yorum: Memnuniyet D. değerlerinde Geleneksel sınıf eğitimi ve Çevrimiçi eğitim arasında istatistiksel olarak anlamlı bir fark var.

- Geleneksel sınıf eğitimi vs Hibrit eğitim:

Güven aralığı: [1.869689 , 3.463644]

[-3.873626, -0.396296] Güven aralığı 0'ı içermiyor → Fark anlamlı. Yorum: Memnuniyet D. değerlerinde Geleneksel sınıf eğitimi ve Hibrit eğitim grupları arasında istatistiksel olarak anlamlı bir fark var.

- Çevrimiçi eğitim vs Hibrit eğitim:

Güven aralığı: [0.136356 , 1.730311]

[0.136356,1.730311] Güven aralığı 0'ı içermiyor → Fark anlamlı. Yorum: Memnuniyet D. değerlerinde Çevrimiçi eğitim ve Hibrit eğitim grupları arasında istatistiksel olarak anlamlı bir fark var.