Modüllerin içe aktarılması ve verilerin yüklenmesi

In [1]:
```
import numpy as mp
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sbn

veriler=pd.read_csv("C:\\Users\\Dell\\Desktop\\datathon2023\\train.csv")
testVerileri=pd.read_csv("C:\\Users\\Dell\\Desktop\\datathon2023\\test_x.csv")
```

Verilere genel bir bakış

In [2]:
```
print(veriler.isnull().sum())
print("----------------------------------")
veriler.info()
# verilerde her değişkende 5460 değer bulunuyor ve hiç kayıp görünmüyor.
```

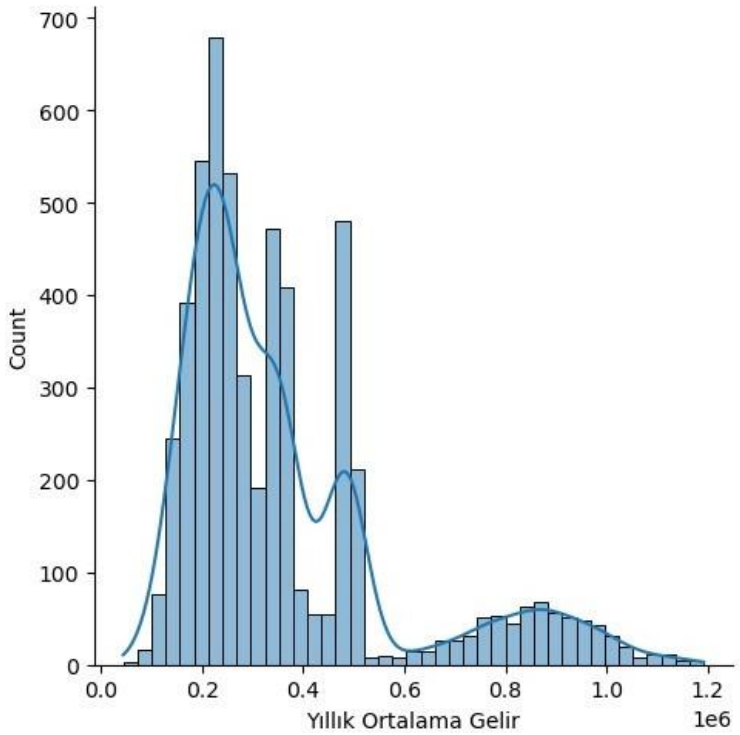
```
index                                          0
Cinsiyet                                       0
Yaş Grubu                                      0
Medeni Durum                                   0
Eğitim Düzeyi                                  0
İstihdam Durumu                                0
Yıllık Ortalama Gelir                          0
Yaşadığı Şehir                                 0
En Çok İlgilendiği Ürün Grubu                  0
Yıllık Ortalama Satın Alım Miktarı             0
Yıllık Ortalama Sipariş Verilen Ürün Adedi     0
Eğitime Devam Etme Durumu                      0
Öbek İsmi                                      0
Yıllık Ortalama Sepete Atılan Ürün Adedi       0
dtype: int64
----------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5460 entries, 0 to 5459
Data columns (total 14 columns):
 #    Column                                    Non-Null Count  Dtype
.....                                          ----------------
 0    index                                     5460 non-null   int64
 1    Cinsiyet                                  5460 non-null   object
 2    Yaş Grubu                                 5460 non-null   object
 3    Medeni Durum                              5460 non-null   object
 4    Eğitim Düzeyi                             5460 non-null   object
 5    İstihdam Durumu                           5460 non-null   object
 6    Yıllık Ortalama Gelir                     5460 non-null   float64
 7    Yaşadığı Şehir                            5460 non-null   object
 8    En Çok İlgilendiği Ürün Grubu             5460 non-null   object
 9    Yıllık Ortalama Satın Alım Miktarı        5460 non-null   float64
 10   Yıllık Ortalama Sipariş Verilen Ürün Adedi  5460 non-null   float64
 11   Eğitime Devam Etme Durumu                 5460 non-null   object
 12   Öbek İsmi                                 5460 non-null   object
 13   Yıllık Ortalama Sepete Atılan Ürün Adedi  5460 non-null   float64
dtypes: float64(4), int64(1), object(9)
memory usage: 597.3+ KB
```

In [3]:
```
print(testVerileri.isnull().sum())
print("----------------------------------")
testVerileri.info()
# verilerde her değişkende 2340 değer bulunuyor ve hiç kayıp görünmüyor.
```

```
index                                        0
Cinsiyet                                     0
Yaş Grubu                                    0
Medeni Durum                                 0
Eğitim Düzeyi                                0
İstihdam Durumu                              0
Yıllık Ortalama Gelir                        0
Yaşadığı Şehir                               0
En Çok İlgilendiği Ürün Grubu                0
Yıllık Ortalama Satın Alım Miktarı           0
Yıllık Ortalama Sipariş Verilen Ürün Adedi   0
Eğitime Devam Etme Durumu                    0
Yıllık Ortalama Sepete Atılan Ürün Adedi     0
dtype: int64
----------------------------------------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2340 entries, 0 to 2339
Data columns (total 13 columns):
 #   Column                                   Non-Null Count  Dtype
---  ------                                   --------------  -----
 0   index                                    2340 non-null   int64
 1   Cinsiyet                                 2340 non-null   object
 2   Yaş Grubu                                2340 non-null   object
 3   Medeni Durum                             2340 non-null   object
 4   Eğitim Düzeyi                            2340 non-null   object
 5   İstihdam Durumu                          2340 non-null   object
 6   Yıllık Ortalama Gelir                    2340 non-null   float64
 7   Yaşadığı Şehir                           2340 non-null   object
 8   En Çok İlgilendiği Ürün Grubu            2340 non-null   object
 9   Yıllık Ortalama Satın Alım Miktarı       2340 non-null   float64
 10  Yıllık Ortalama Sipariş Verilen Ürün Adedi  2340 non-null  float64
 11  Eğitime Devam Etme Durumu                2340 non-null   object
 12  Yıllık Ortalama Sepete Atılan Ürün Adedi 2340 non-null   float64
dtypes: float64(4), int64(1), object(8)
memory usage: 237.8+ KB
```

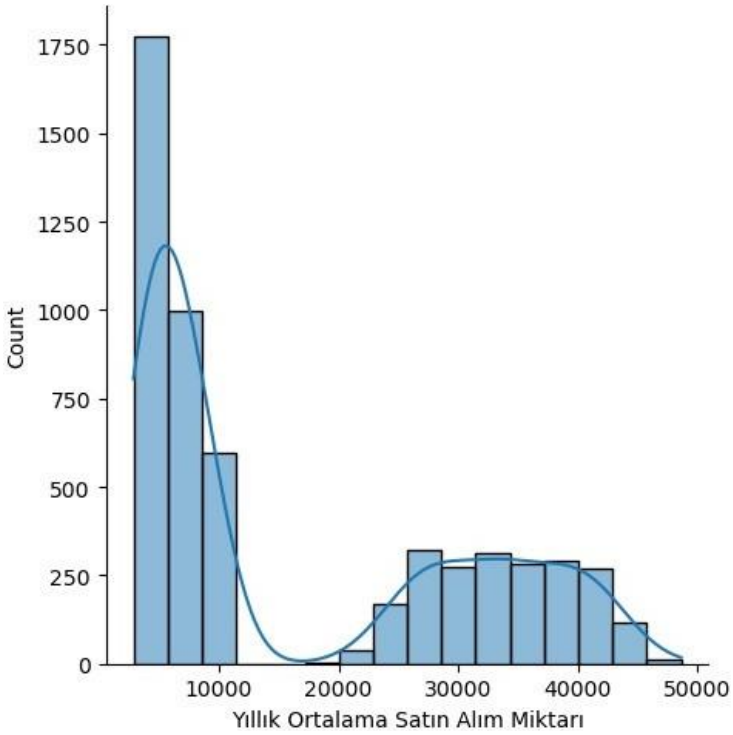
In [4]:
```
sbn.displot(data=veriler,x="Yıllık Ortalama Gelir", kde=True)
#veri normal dağılım göstermemekte ve aykırı değerler içermektedir.
```

Out[4]:
```
<seaborn.axisgrid.FacetGrid at 0x1732082e990>
```



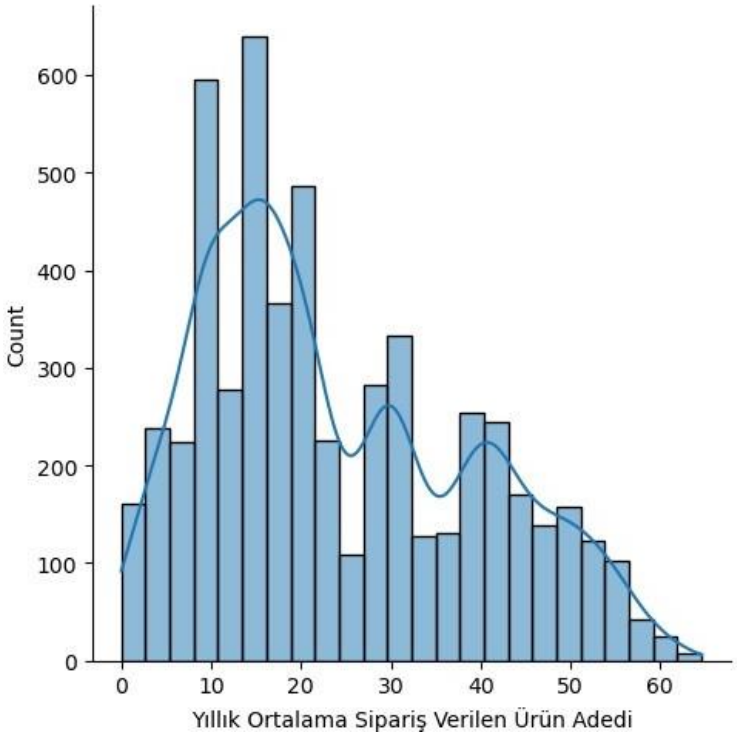
In [5]:
```
sbn.displot(data=veriler,x="Yıllık Ortalama Satın Alım Miktarı", kde=True)
#veri normal dağılım göstermemektedir.
```

Out[5]:
```
<seaborn.axisgrid.FacetGrid at 0x173208d9290>
```



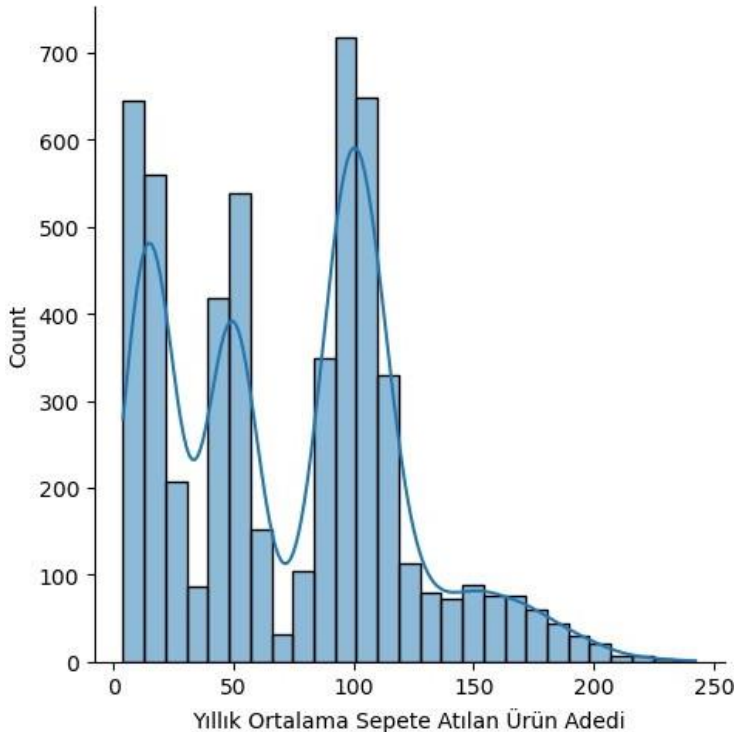
In [6]:
```
sbn.displot(data=veriler,x="Yıllık Ortalama Sipariş Verilen Ürün Adedi", kde=True)
#veri normal dağılım göstermemektedir.
```

Out[6]: <seaborn.axisgrid.FacetGrid at 0x17324c8d010>



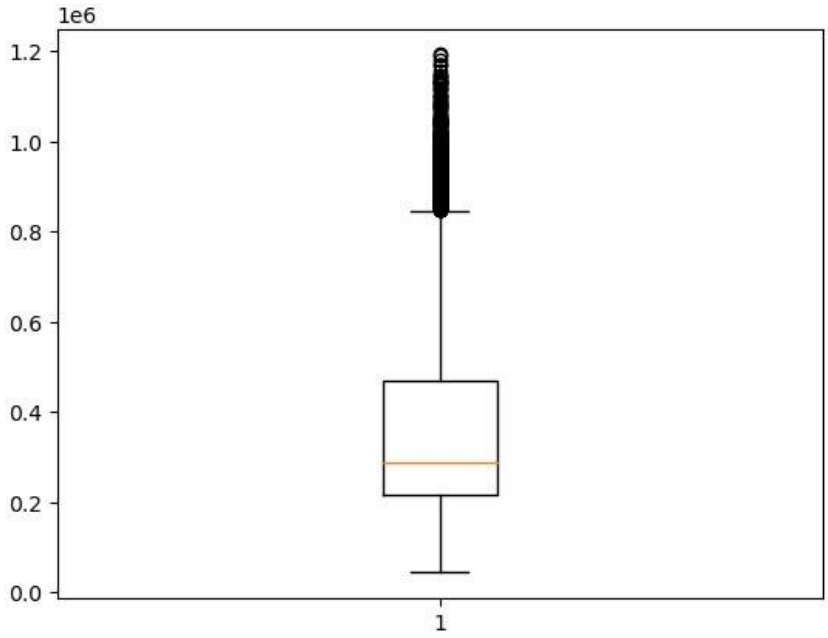
In [7]:
```
sbn.displot(data=veriler,x="Yıllık Ortalama Sepete Atılan Ürün Adedi", kde=True)
#veri normal dağılım göstermemektedir.
```

Out[7]: <seaborn.axisgrid.FacetGrid at 0x17324d5d750>



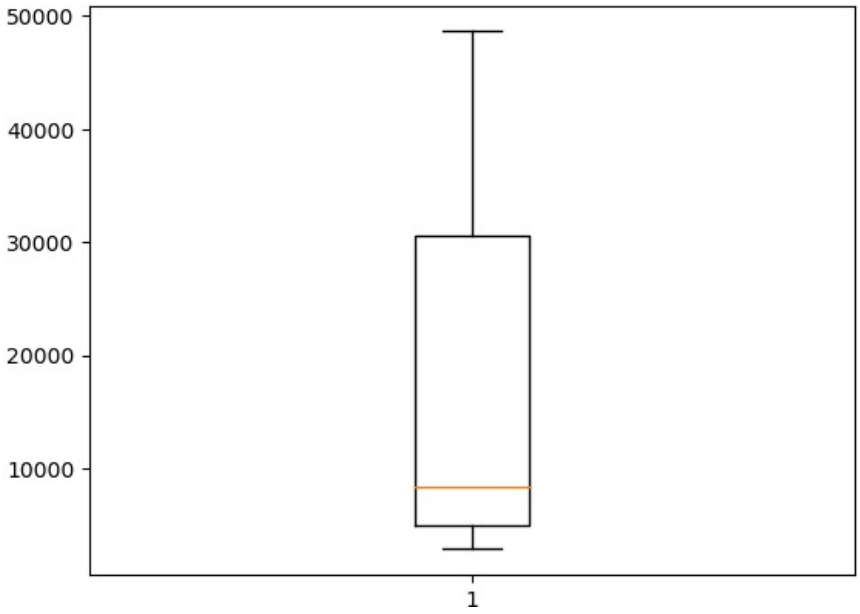
In [8]:
```
plt.boxplot(veriler["Yıllık Ortalama Gelir"])
# grafikte aykırı değer görülmektedir. grafiye göre eğik ve negatif bir durum söz konusudur.
```

Out[8]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x17325e23190>,
  <matplotlib.lines.Line2D at 0x17325e23f50>],
 'caps': [<matplotlib.lines.Line2D at 0x17325e34b90>,
  <matplotlib.lines.Line2D  at  0x17325e35610>],
 'boxes': [<matplotlib.lines.Line2D at 0x17324dc7490>],
 'medians': [<matplotlib.lines.Line2D at 0x17325e36110>],
 'fliers': [<matplotlib.lines.Line2D at 0x17325db6b50>],
 'means': []}
```



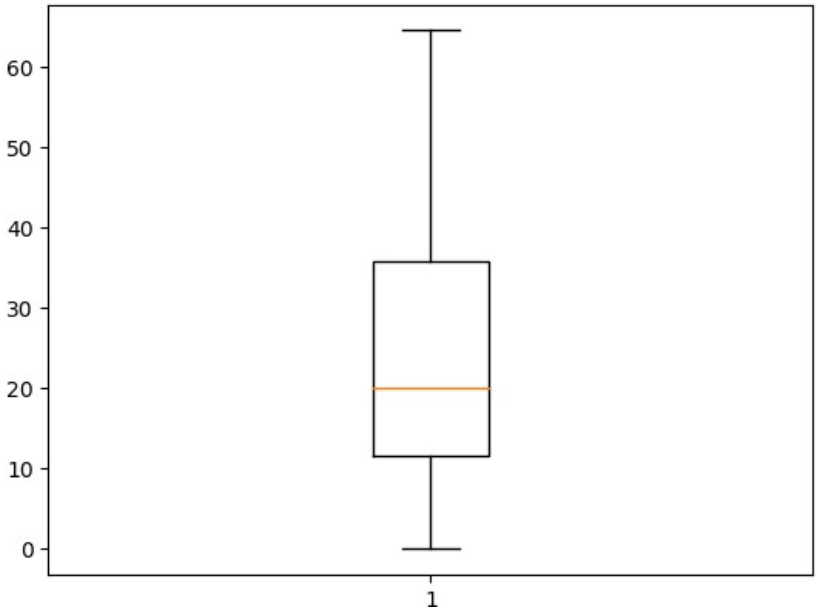
In [9]:
```
plt.boxplot(veriler["Yıllık Ortalama Satın Alım Miktarı"])
#eğik ve negatif bir durum söz konusudur. veriler üst limite daha yakındır.
```

Out[9]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x17325e99610>,
  <matplotlib.lines.Line2D at 0x17325e9a090>],
 'caps': [<matplotlib.lines.Line2D at 0x17325e9ac10>,
  <matplotlib.lines.Line2D  at  0x17325e9b6d0>],
 'boxes': [<matplotlib.lines.Line2D at 0x17325e98a10>],
 'medians': [<matplotlib.lines.Line2D at 0x17325ea4290>],
 'fliers': [<matplotlib.lines.Line2D at 0x17325ea4210>],
 'means': []}
```



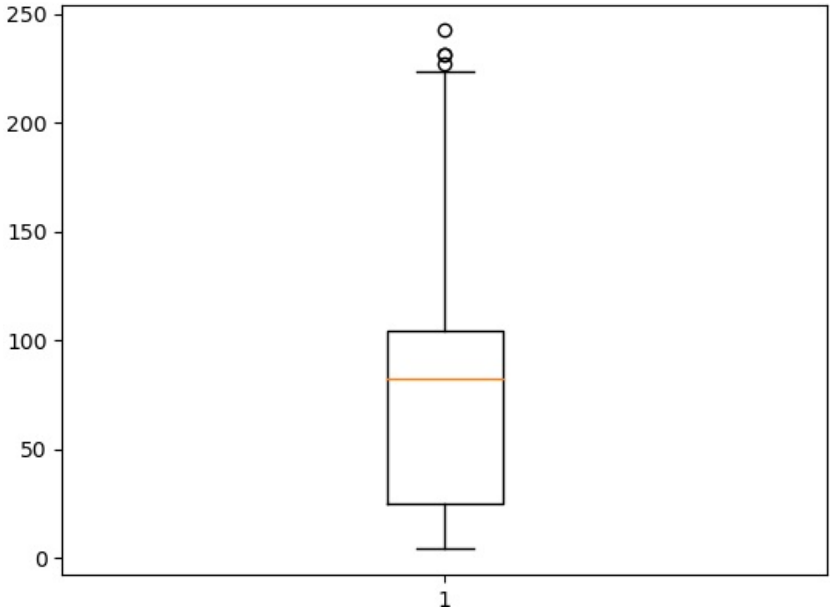
In [10]:
```
plt.boxplot(veriler["Yıllık Ortalama Sipariş Verilen Ürün Adedi"])
#eğik ve negatif bir durum söz konusudur. veriler üst limite daha yakındır.
```

Out[10]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x17325e43b10>,
  <matplotlib.lines.Line2D at 0x17325f09690>],
 'caps': [<matplotlib.lines.Line2D at 0x17325f0a1d0>,
  <matplotlib.lines.Line2D  at  0x17325f0ac90>],
 'boxes': [<matplotlib.lines.Line2D at 0x17325effd90>],
 'medians': [<matplotlib.lines.Line2D at 0x17325f0b750>],
 'fliers': [<matplotlib.lines.Line2D at 0x17325f10190>],
 'means': []}
```



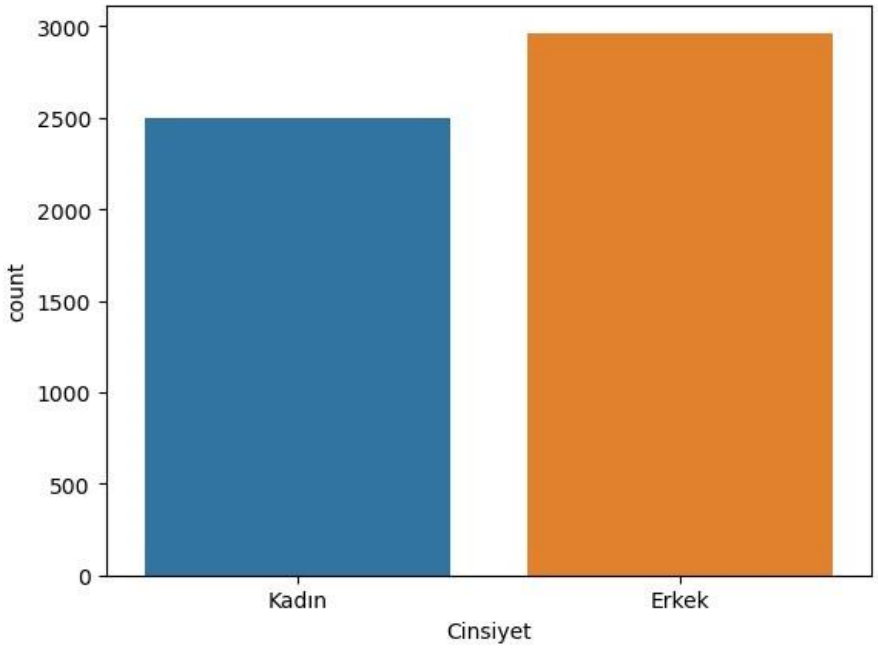
In [11]:
```
plt.boxplot(veriler["Yıllık Ortalama Sepete Atılan Ürün Adedi"])
# grafikte aykırı değer görülmektedir. #eğik ve pozitif bir durum söz konusudur. veriler alt limite daha yakınd
```

Out[11]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x17325ead9d0>,
  <matplotlib.lines.Line2D at 0x17325f76650>],
 'caps': [<matplotlib.lines.Line2D at 0x17325f77250>,
  <matplotlib.lines.Line2D  at  0x17325f77d90>],
 'boxes': [<matplotlib.lines.Line2D at 0x17325f74e50>],
 'medians': [<matplotlib.lines.Line2D at 0x17325f7c8d0>],
 'fliers': [<matplotlib.lines.Line2D at 0x17325f7d310>],
 'means': []}
```



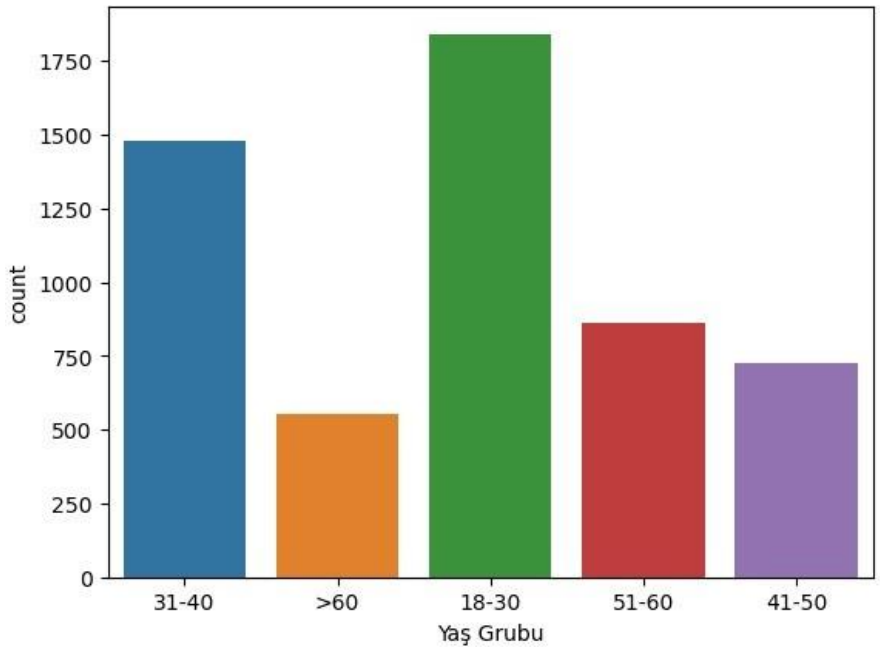
In [12]:
```
sbn.countplot(data=veriler,x="Cinsiyet")
#erkek sayısının kadın sayısına göre biraz daha fazladır.
```

Out[12]:
```
<Axes: xlabel='Cinsiyet', ylabel='count'>
```



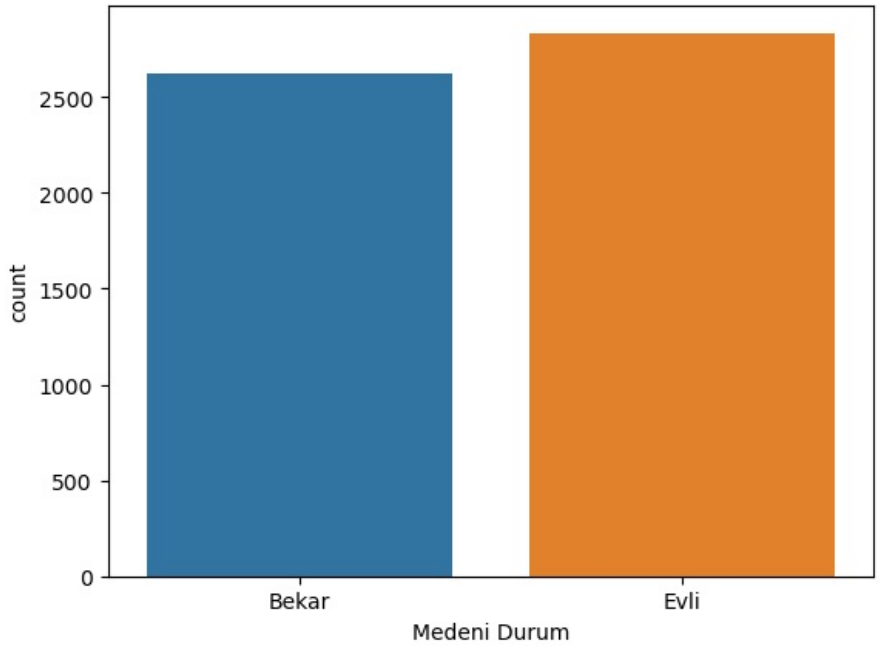
In [13]:
```
sbn.countplot(data=veriler,x="Yaş Grubu")
#yaş dağılımında genç kategoriden yaşlı kategoriye doğru bir azalış eğilimi göstermektedir.
```

Out[13]:
```
<Axes: xlabel='Yaş Grubu', ylabel='count'>
```



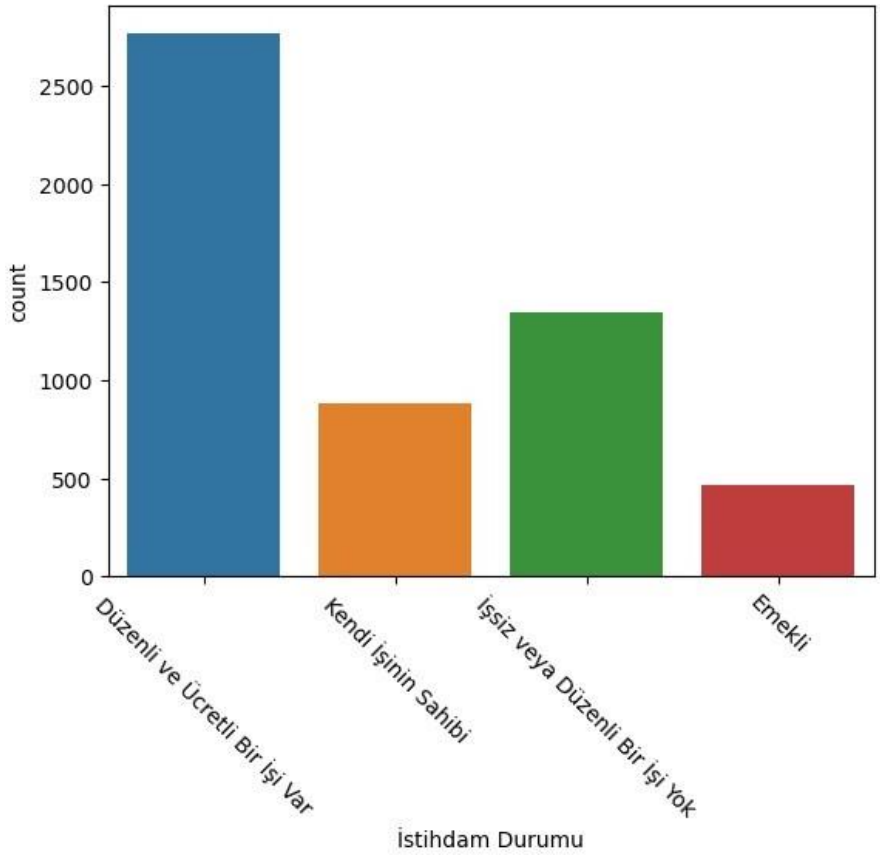
In [14]:
```
sbn.countplot(data=veriler,x="Medeni Durum")
#evli sayısı fazla olmasına rağmen dengeli bir dağılıma sahiptir.
```

Out[14]:
```
<Axes: xlabel='Medeni Durum', ylabel='count'>
```



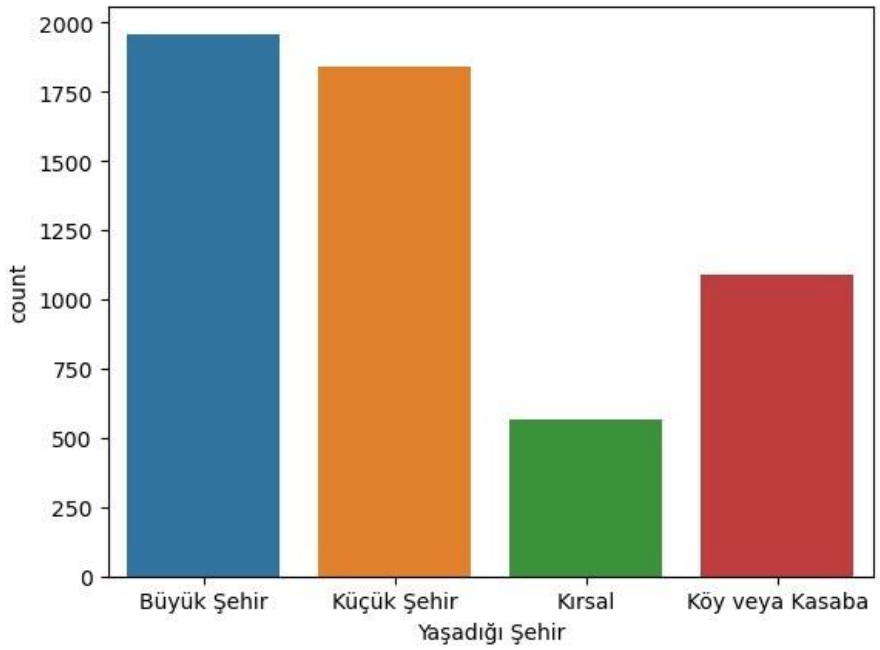
In [15]:
```
sbn.countplot(data=veriler,x="İstihdam Durumu")
plt.xticks(rotation=-45)
```

Out[15]:
```
(array([0, 1, 2, 3]),
 [Text(0, 0, 'Düzenli ve Ücretli Bir İşi Var'),
  Text(1, 0, 'Kendi İşinin Sahibi'),
  Text(2, 0, 'İşsiz veya Düzenli Bir İşi Yok'),
  Text(3, 0, 'Emekli')])
```



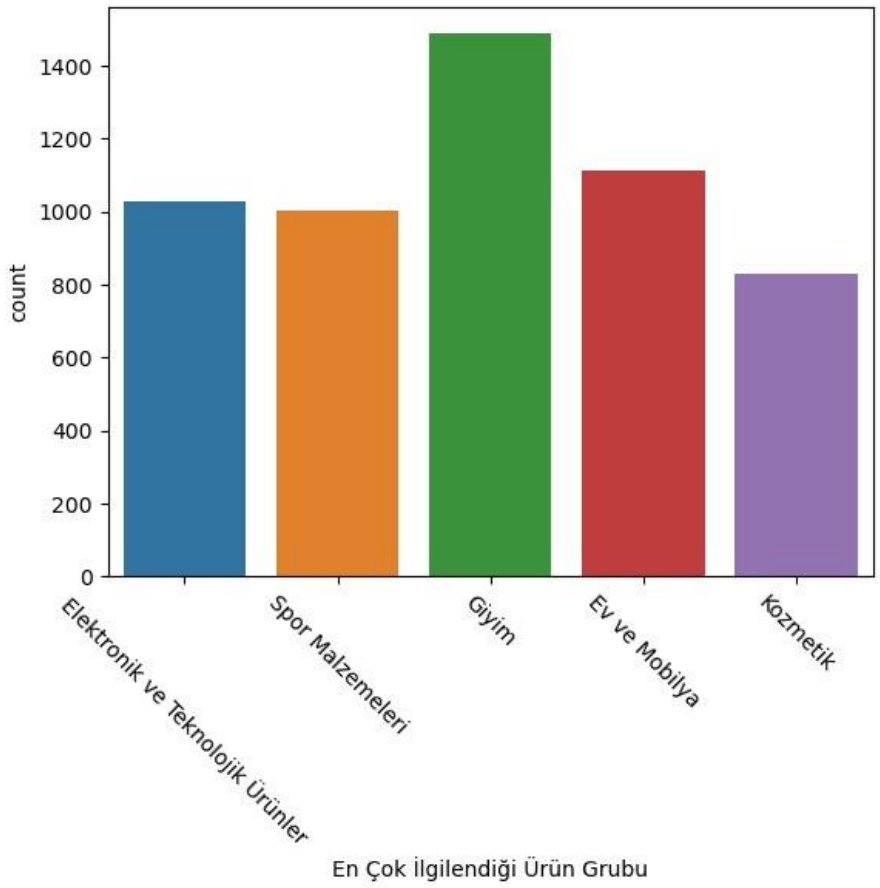
In [16]:
```
sbn.countplot(data=veriler,x="Yaşadığı Şehir")
```

Out[16]:
```
<Axes: xlabel='Yaşadığı Şehir', ylabel='count'>
```



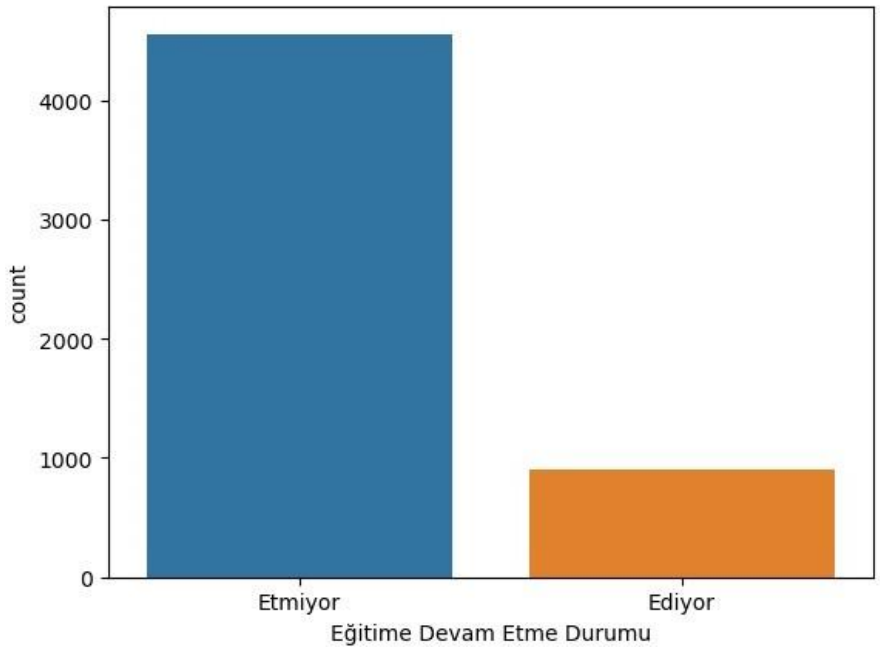
In [17]:
```
sbn.countplot(data=veriler,x="En Çok İlgilendiği Ürün Grubu")
plt.xticks(rotation=-45)
```

Out[17]:
```
(array([0, 1, 2, 3, 4]),
 [Text(0, 0, 'Elektronik ve Teknolojik Ürünler'),
  Text(1, 0, 'Spor Malzemeleri'),
  Text(2,  0,  'Giyim'),
  Text(3, 0, 'Ev ve Mobilya'),
  Text(4, 0, 'Kozmetik')])
```



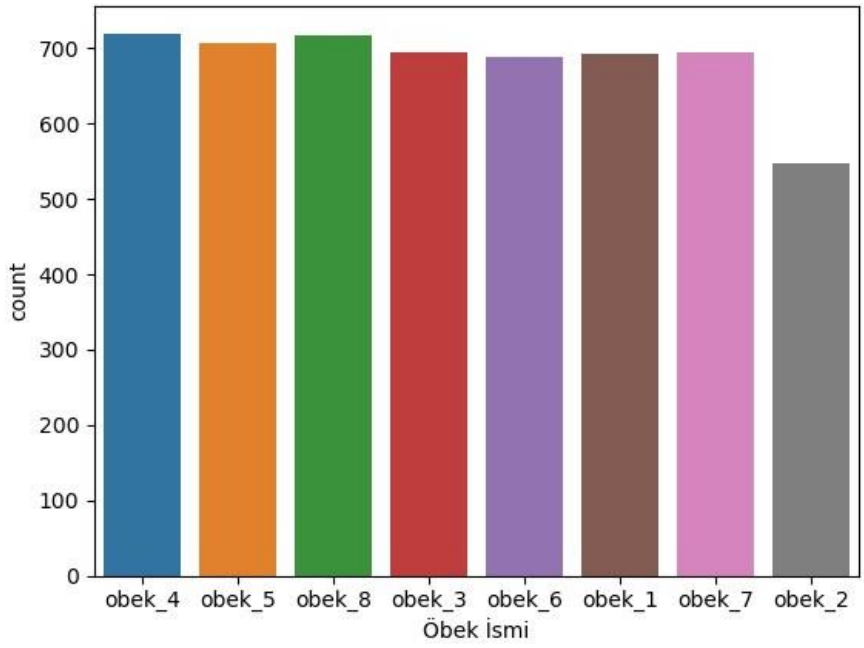
In [18]:
```
sbn.countplot(data=veriler,x="Eğitime Devam Etme Durumu")
```

Out[18]:
```
<Axes: xlabel='Eğitime Devam Etme Durumu', ylabel='count'>
```



In [19]:
```
sbn.countplot(data=veriler,x="Öbek İsmi"),
#endüşük kayıt sayısı obek 2 olarak görünmekte diğerleri dengeli bir dağılıma sahiptir.
```

Out[19]:
```
(<Axes: xlabel='Öbek İsmi', ylabel='count'>,)
```



In [20]:
```
veriler.describe()
```

Out[20]:

	index	Yıllık Ortalama Gelir	Yıllık Ortalama Satın Alım Miktarı	Yıllık Ortalama Sipariş Verilen Ürün Adedi	Yıllık Ortalama Sepete Atılan Ürün Adedi
count	5460.000000	5.460000e+03	5460.000000	5460.000000	5460.000000
mean	2729.500000	3.635711e+05	16616.612217	24.040884	73.445693
std	1576.310566	2.197144e+05	14099.171704	14.945655	47.214184
min	0.000000	4.392299e+04	2859.254000	0.000000	3.977559
25%	1364.750000	2.156934e+05	4931.859057	11.550502	25.009168
50%	2729.500000	2.869254e+05	8426.818967	20.095924	82.485579
75%	4094.250000	4.681882e+05	30579.244695	35.918161	104.473291
max	5459.000000	1.192437e+06	48605.594415	64.616196	242.308441

In [21]:
```
veriler.describe(include=['O'])
```

Out[21]:

	Cinsiyet	Yaş Grubu	Medeni Durum	Eğitim Düzeyi	İstihdam Durumu	Yaşadığı Şehir	En Çok İlgilendiği Ürün Grubu	Eğitime Devam Etme Durumu	Öbek İsmi
count	5460	5460	5460	5460	5460	5460	5460	5460	5460
unique	2	5	2	9	4	4	5	2	8
top	Erkek	18-30	Evli	Lise Mezunu	Düzenli ve Ucretli Bir İş Var	Büyük Şehir	Giyim	Etmiyor	obek_4
freq	2964	1841	2834	1388	2768	1959	1487	4554	720

In [22]:
```
testVerileri.describe()
```

Out[22]:

	index	Yıllık Ortalama Gelir	Yıllık Ortalama Satın Alım Miktarı	Yıllık Ortalama Sipariş Verilen Ürün Adedi	Yıllık Ortalama Sepete Atılan Ürün Adedi
count	2340.000000	2.340000e+03	2340.000000	2340.000000	2340.000000
mean	1169.500000	3.622524e+05	16674.303935	23.465787	72.078920
std	675.644137	2.232063e+05	14266.101056	14.559035	46.311708
min	0.000000	8.065675e+04	2870.657175	0.000000	5.283642
25%	584.750000	2.120081e+05	4807.074240	11.336283	24.383782
50%	1169.500000	2.836947e+05	8186.095300	20.094319	71.033886
75%	1754.250000	4.669887e+05	30747.419142	33.431156	104.202802
max	2339.000000	1.229399e+06	46380.542188	64.017477	237.768581

In [23]:
```
testVerileri.describe(include=['O'])
```

Out[23]:

	Cinsiyet	Yaş Grubu	Medeni Durum	Eğitim Düzeyi	İstihdam Durumu	Yaşadığı Şehir	En Çok İlgilendiği Ürün Grubu	Eğitime Devam Etme Durumu
count	2340	2340	2340	2340	2340	2340	2340	2340
unique	2	5	2	9	4	4	5	2
top	Erkek	18-30	Evli	Lise Mezunu	Düzenli ve Ücretli Bir İş Var	Büyük Şehir	Giyim	Etmiyor
freq	1250	710	1248	582	1193	854	636	1967

In [24]:
```
veriler.rename(columns={"Öbek İsmi":"ObekIsmi"},inplace=True)
obekListesi=["obek_1","obek_2","obek_3","obek_4","obek_5","obek_6","obek_7","obek_8"]
```

In [25]:
```
# sayısal verilerin okunabilirliğini arttırmak için her bir sutunu bir tablo haline dönüştürüyoruz.
def olustur(sutun):
    tablo1=veriler[sutun]
    tablo1=pd.concat([tablo1,veriler["ObekIsmi"]],axis=1)
    tablo2=pd.DataFrame(index=["Count","Sum","Mean","Std","Min","Max","Yüzde"],columns=["obek_1","obek_2","obek
                                                                                        "obek_4","obek_5",
                                                                                        "obek_7","obek_8"]
    for i in obekListesi:
        g=0
        liste=[]
        while g<len(tablo1):
            if tablo1["ObekIsmi"][g]==i:
                liste.append(tablo1[sutun][g])
            g=g+1
        liste=mp.array(liste)
        toplam=liste.sum()
        say=len(liste)
        ort=liste.mean()
        stds=liste.std()
        mind=liste.min()
        mak=liste.max()
        tablo2[i]["Sum"]=int(toplam)
        tablo2[i]["Mean"]=int(ort)
        tablo2[i]["Std"]=int(stds)
        tablo2[i]["Min"]=int(mind)
        tablo2[i]["Max"]=int(mak)
        tablo2[i]["Count"]=say
        tablo2[i]["Yüzde"]=say/len(tablo1)
    return tablo2
```

In [26]:
```
# kategorik verilerin okunabilirliğini arttırmak için her bir sutunu bir tablo haline dönüştürüyoruz.
sutunListesi=["Cinsiyet","Yaş Grubu","Medeni Durum","Eğitim Düzeyi","İstihdam Durumu","Yaşadığı Şehir",
              "En Çok İlgilendiği Ürün Grubu","Eğitime Devam Etme Durumu"]

cinsiyet=pd.DataFrame()
yasGrubu=pd.DataFrame()
medeniDurum=pd.DataFrame()
egitimDuzeyi=pd.DataFrame()
istihdamDurumu=pd.DataFrame()
yasadigiSehir=pd.DataFrame()
urunGrubu=pd.DataFrame()
mezunDurumu=pd.DataFrame()

for i in obekListesi:
    tablo=veriler[(veriler.ObekIsmi==i)]
    for k in sutunListesi:
        if k=="Cinsiyet":
            tablo2=pd.DataFrame(tablo[k].value_counts())
            tablo2.rename(columns={"count":i},inplace=True)
            cinsiyet=pd.concat([cinsiyet,tablo2],axis=1)
        if k=="Yaş Grubu":
            tablo2=pd.DataFrame(tablo[k].value_counts())
            tablo2.rename(columns={"count":i},inplace=True)
            yasGrubu=pd.concat([yasGrubu,tablo2],axis=1)
        if k=="Medeni Durum":
            tablo2=pd.DataFrame(tablo[k].value_counts())
            tablo2.rename(columns={"count":i},inplace=True)
            medeniDurum=pd.concat([medeniDurum,tablo2],axis=1)
        if k=="Eğitim Düzeyi":
            tablo2=pd.DataFrame(tablo[k].value_counts())
            tablo2.rename(columns={"count":i},inplace=True)
            egitimDuzeyi=pd.concat([egitimDuzeyi,tablo2],axis=1)
        if k=="İstihdam Durumu":
            tablo2=pd.DataFrame(tablo[k].value_counts())
            tablo2.rename(columns={"count":i},inplace=True)
            istihdamDurumu=pd.concat([istihdamDurumu,tablo2],axis=1)
        if k=="Yaşadığı Şehir":
            tablo2=pd.DataFrame(tablo[k].value_counts())
            tablo2.rename(columns={"count":i},inplace=True)
            yasadigiSehir=pd.concat([yasadigiSehir,tablo2],axis=1)
        if k=="En Çok İlgilendiği Ürün Grubu":
            tablo2=pd.DataFrame(tablo[k].value_counts())
```

```
            tablo2.rename(columns={"count":i},inplace=True)
            urunGrubu=pd.concat([urunGrubu,tablo2],axis=1)
        if k=="Eğitime Devam Etme Durumu":
            tablo2=pd.DataFrame(tablo[k].value_counts())
            tablo2.rename(columns={"count":i},inplace=True)
            mezunDurumu=pd.concat([mezunDurumu,tablo2],axis=1)
```

In [27]:
```
#öbeklere göre cinsiyet dağılımı
cinsiyet
```

Out[27]:

Cinsiyet	obek_1	obek_2	obek_3	obek_4	obek_5	obek_6	obek_7	obek_8
Kadın	366	528	69	352	130	344	347	360
Erkek	326	19	626	368	576	344	348	357

In [28]:
```
#öbeklere göre yaş dağılımı
yasGrubu
```

Out[28]:

Yaş Grubu	obek_1	obek_2	obek_3	obek_4	obek_5	obek_6	obek_7	obek_8
18-30	174	134	134	292	251	20	139	697
41-50	169	109	130	5	40	41	231	2
31-40	159	106	148	412	350	33	259	10
51-60	134	99	153	9	37	390	33	6
>60	56	99	130	2	28	204	33	2

In [29]:
```
#öbeklere göre medeni durum dağılımı
medeniDurum
```

Out[29]:

Medeni Durum	obek_1	obek_2	obek_3	obek_4	obek_5	obek_6	obek_7	obek_8
Evli	404	424	562	164	309	460	430	81
Bekar	288	123	133	556	397	228	265	636

In [30]:
```
#öbeklere göre eğitim düzeyi dağılımı
egitimDuzeyi
```

Out[30]:

Eğitim Düzeyi	obek_1	obek_2	obek_3	obek_4	obek_5	obek_6	obek_7	obek_8
Eğitimsiz	142	94	64	16	5	26	26	8
Lise Mezunu	134	103	361	29	534	17	47	163
Ortaokul Mezunu	131	108	59	22	21	18	37	16
İlkokul Mezunu	123	121	60	24	19	14	34	22
Doktora Ötesi	42	30	4	46	20	35	30	2
Yüksekokul Mezunu	35	22	81	106	20	67	36	221
Üniversite Mezunu	30	27	38	280	29	414	218	221
Yüksek Lisans Mezunu	29	21	21	125	24	64	201	60
Doktora Mezunu	26	21	7	72	34	33	66	4

In [31]:
```
#öbeklere göre istihdam dağılımı
istihdamDurumu
```

Out[31]:

İstihdam Durumu	obek_1	obek_2	obek_3	obek_4	obek_5	obek_6	obek_7	obek_8
Düzenli ve Ücretli Bir İşi Var	357	97	281	363	428	247	576	419
İşsiz veya Düzenli Bir İşi Yok	205	417	124	252	119	7	5	220
Kendi İşinin Sahibi	127	29	225	101	156	87	78	76
Emekli	3	4	65	4	3	347	36	2

In [32]:
```
##öbeklere göre yaşadığı şehir dağılımı
yasadigiSehir
```

Out[32]:

Yaşadığı Şehir	obek_1	obek_2	obek_3	obek_4	obek_5	obek_6	obek_7	obek_8
Küçük Şehir	258	203	271	43	348	96	312	311
Köy veya Kasaba	171	118	146	27	224	330	35	41
Büyük Şehir	142	115	215	609	133	38	343	364
Kırsal	121	111	63	41	1	224	5	1

In [33]:
```
#öbeklere göre ürün grubu dağılımı
urunGrubu
```

Out[33]:

En Çok İlgilendiği Ürün Grubu	obek_1	obek_2	obek_3	obek_4	obek_5	obek_6	obek_7	obek_8
Giyim	435	221	136	156	112	129	142	156
Kozmetik	176	30	3	152	74	117	125	152
Ev ve Mobilya	49	211	149	131	48	185	189	152
Elektronik ve Teknolojik Ürünler	23	40	202	128	247	118	138	133
Spor Malzemeleri	9	45	205	153	225	139	101	124

In [34]:
```
#öbeklere göre mezuniyet dağılımı
mezunDurumu
```

Out[34]:

Eğitime Devam Etme Durumu	obek_1	obek_2	obek_3	obek_4	obek_5	obek_6	obek_7	obek_8
Etmiyor	651	526	663	654	658	679	627	96
Ediyor	41	21	32	66	48	9	68	621

In [35]:
```
#öbeklere göre gelir dağılımı
gelirTablosu=olustur("Yıllık Ortalama Gelir")
gelirTablosu.index.names=["Yıllık Ort. Gelir Tablosu"]
gelirTablosu
```

Out[35]:

Yıllık Ort. Gelir Tablosu	obek_1	obek_2	obek_3	obek_4	obek_5	obek_6	obek_7	obek_8
Count	692	547	695	720	706	688	695	717
Sum	158823582	94536647	227889870	588942371	156890169	330321029	244605233	183089482
Mean	229513	172827	327899	817975	222224	480117	351949	255354
Std	92665	65443	86768	189109	61378	55386	58257	67880
Min	43922	79099	45695	109362	137702	130268	106393	106170
Max	1146085	893623	757355	1192437	1044462	982577	963312	977308
Yüzde	0.12674	0.100183	0.127289	0.131868	0.129304	0.126007	0.127289	0.131319

In [36]:
```
#öbeklere göre sipariş verilen ürün miktar dağılımı
urunAdediTablosu=olustur("Yıllık Ortalama Sipariş Verilen Ürün Adedi")
urunAdediTablosu.index.names=["Y. Ort. Sipariş Ver. Ürün Adet Tablosu"]
urunAdediTablosu
```

Out[36]:

Y. Ort. Sipariş Ver. Ürün Adet Tablosu	obek_1	obek_2	obek_3	obek_4	obek_5	obek_6	obek_7	obek_8
Count	692	547	695	720	706	688	695	717
Sum	4058	11304	10570	28001	7552	20486	14094	35195
Mean	5	20	15	38	10	29	20	49
Std	6	6	2	6	4	3	3	8
Min	0	1	3	1	3	4	0	0
Max	58	51	50	58	52	54	53	64
Yüzde	0.12674	0.100183	0.127289	0.131868	0.129304	0.126007	0.127289	0.131319

In [37]:
```
#öbeklere göre satım alım miktar dağılımı
satinAlmaTablosu=olustur("Yıllık Ortalama Satın Alım Miktarı")
satinAlmaTablosu.index.names=["Yıllık Ort. Satın Alım Miktar Tablosu"]
satinAlmaTablosu
```

Out[37]:

	obek_1	obek_2	obek_3	obek_4	obek_5	obek_6	obek_7	obek_8
Yıllık Ort. Satın Alım Miktar Tablosu								
Count	692	547	695	720	706	688	695	717
Sum	4184659	2732833	5115684	23331606	3206867	27134908	18368359	6651783
Mean	6047	4996	7360	32405	4542	39440	26429	9277
Std	4333	5146	4443	6516	3689	6037	4739	2902
Min	3522	2859	3636	3522	3060	3693	3909	3655
Max	41875	43853	44351	46392	42128	48605	44282	42293
Yüzde	0.12674	0.100183	0.127289	0.131868	0.129304	0.126007	0.127289	0.131319

In [38]:
```
#öbeklere göre sepete atılan ürün miktar dağılımı
sepeteAtmaTablosu=olustur("Yıllık Ortalama Sepete Atılan Ürün Adedi")
sepeteAtmaTablosu.index.names=["Yıllık Ort. Sepete Atılan Ürün Miktar Tablosu"]
sepeteAtmaTablosu
```

Out[38]:

	obek_1	obek_2	obek_3	obek_4	obek_5	obek_6	obek_7	obek_8
Yıllık Ort. Sepete Atılan Ürün Miktar Tablosu								
Count	692	547	695	720	706	688	695	717
Sum	9175	27891	14992	70580	70291	34866	68235	104980
Mean	13	50	21	98	99	50	98	146
Std	17	15	11	17	15	13	17	34
Min	3	5	5	6	4	9	8	7
Max	187	172	132	155	199	194	188	242
Yüzde	0.12674	0.100183	0.127289	0.131868	0.129304	0.126007	0.127289	0.131319

In [39]:
```
# Makine öğrenmesinde ezber olmaması için id bilgilerini çıkartarak verileri yeni bir değişkene atıyoruz.
x=veriler.iloc[:,1:]
xt=testVerileri.iloc[:,1:]
```

Encoding İşlemi

```
# Öbek ismi sutunundaki metisel veriyi sayısal veriye dönüştürüyoruz.
i=0
while i<len(x):
    if x["ObekIsmi"][i]=="obek_1":
        x["ObekIsmi"][i]=0
    if x["ObekIsmi"][i]=="obek_2":
        x["ObekIsmi"][i]=1
    if x["ObekIsmi"][i]=="obek_3":
        x["ObekIsmi"][i]=2
    if x["ObekIsmi"][i]=="obek_4":
        x["ObekIsmi"][i]=3
    if x["ObekIsmi"][i]=="obek_5":
        x["ObekIsmi"][i]=4
    if x["ObekIsmi"][i]=="obek_6":
        x["ObekIsmi"][i]=5
    if x["ObekIsmi"][i]=="obek_7":
        x["ObekIsmi"][i]=6
    if x["ObekIsmi"][i]=="obek_8":
        x["ObekIsmi"][i]=7
    i=i+1
```

In [42]:
```
# x tablosunu genel bir bakış
x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5460 entries, 0 to 5459
Data columns (total 13 columns):
 #   Column                                    Non-Null Count  Dtype
---  ------                                    --------------  -----
 0   Cinsiyet                                  5460 non-null   object
 1   Yaş Grubu                                 5460 non-null   object
 2   Medeni Durum                              5460 non-null   object
 3   Eğitim Düzeyi                             5460 non-null   object
 4   İstihdam Durumu                           5460 non-null   object
 5   Yıllık Ortalama Gelir                     5460 non-null   float64
 6   Yaşadığı Şehir                            5460 non-null   object
 7   En Çok İlgilendiği Ürün Grubu             5460 non-null   object
 8   Yıllık Ortalama Satın Alım Miktarı        5460 non-null   float64
 9   Yıllık Ortalama Sipariş Verilen Ürün Adedi  5460 non-null   float64
 10  Eğitime Devam Etme Durumu                 5460 non-null   object
 11  ObekIsmi                                  5460 non-null   object
 12  Yıllık Ortalama Sepete Atılan Ürün Adedi  5460 non-null   float64
dtypes: float64(4), object(9)
memory usage: 554.7+ KB
```

In [42]:
```
# öbek ismi sutununu eğitimde hata vermemesi için object veri türünden integer veri türüne dönüştürüyoruz.
x["ObekIsmi"]=pd.to_numeric(x["ObekIsmi"], downcast='integer')
```

In [43]:
```
# verilerin label encoding ve one hot encoding işlemlerinin yapılması
x=pd.get_dummies(data=x,columns=["Cinsiyet","Yaş Grubu","Medeni Durum","Eğitim Düzeyi",
                                 "İstihdam Durumu","Yaşadığı Şehir",
                                 "En Çok İlgilendiği Ürün Grubu","Eğitime Devam Etme Durumu"
                                 ],dtype=float)
xt=pd.get_dummies(data=xt,columns=["Cinsiyet","Yaş Grubu","Medeni Durum","Eğitim Düzeyi",
                                 "İstihdam Durumu","Yaşadığı Şehir",
                                 "En Çok İlgilendiği Ürün Grubu","Eğitime Devam Etme Durumu"
                                 ],dtype=float)
```

In [44]:
```
#verilerin encoding işleminden sonra sutunların veri tipi ve eğitim ve test veri setindeki sutun sıralamasının
# kontrolunun yapılması
x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5460 entries, 0 to 5459
Data columns (total 38 columns):
 #   Column                                                Non-Null Count  Dtype
---  ------                                                --------------  -----
 0   Yıllık Ortalama Gelir                                 5460 non-null   float64
 1   Yıllık Ortalama Satın Alım Miktarı                    5460 non-null   float64
 2   Yıllık Ortalama Sipariş Verilen Ürün Adedi            5460 non-null   float64
 3   ObekIsmi                                              5460 non-null   int8
 4   Yıllık Ortalama Sepete Atılan Ürün Adedi              5460 non-null   float64
 5   Cinsiyet_Erkek                                        5460 non-null   float64
 6   Cinsiyet_Kadın                                        5460 non-null   float64
 7   Yaş Grubu_18-30                                       5460 non-null   float64
 8   Yaş Grubu_31-40                                       5460 non-null   float64
 9   Yaş Grubu_41-50                                       5460 non-null   float64
 10  Yaş Grubu_51-60                                       5460 non-null   float64
 11  Yaş Grubu_>60                                         5460 non-null   float64
 12  Medeni Durum_Bekar                                    5460 non-null   float64
 13  Medeni Durum_Evli                                     5460 non-null   float64
 14  Eğitim Düzeyi_Doktora Mezunu                          5460 non-null   float64
 15  Eğitim Düzeyi_Doktora Ötesi                           5460 non-null   float64
 16  Eğitim Düzeyi_Eğitimsiz                               5460 non-null   float64
 17  Eğitim Düzeyi_Lise Mezunu                             5460 non-null   float64
 18  Eğitim Düzeyi_Ortaokul Mezunu                         5460 non-null   float64
 19  Eğitim Düzeyi_Yüksek Lisans Mezunu                    5460 non-null   float64
 20  Eğitim Düzeyi_Yüksekokul Mezunu                       5460 non-null   float64
 21  Eğitim Düzeyi_Üniversite Mezunu                       5460 non-null   float64
 22  Eğitim Düzeyi_İlkokul Mezunu                          5460 non-null   float64
 23  İstihdam Durumu_Düzenli ve Ücretli Bir İşi Var        5460 non-null   float64
 24  İstihdam Durumu_Emekli                                5460 non-null   float64
 25  İstihdam Durumu_Kendi İşinin Sahibi                   5460 non-null   float64
 26  İstihdam Durumu_İşsiz veya Düzenli Bir İşi Yok        5460 non-null   float64
 27  Yaşadığı Şehir_Büyük Şehir                            5460 non-null   float64
 28  Yaşadığı Şehir_Köy veya Kasaba                        5460 non-null   float64
 29  Yaşadığı Şehir_Küçük Şehir                            5460 non-null   float64
 30  Yaşadığı Şehir_Kırsal                                 5460 non-null   float64
 31  En Çok İlgilendiği Ürün Grubu_Elektronik ve Teknolojik Ürünler  5460 non-null   float64
 32  En Çok İlgilendiği Ürün Grubu_Ev ve Mobilya           5460 non-null   float64
 33  En Çok İlgilendiği Ürün Grubu_Giyim                   5460 non-null   float64
 34  En Çok İlgilendiği Ürün Grubu_Kozmetik                5460 non-null   float64
 35  En Çok İlgilendiği Ürün Grubu_Spor Malzemeleri        5460 non-null   float64
 36  Eğitime Devam Etme Durumu_Ediyor                      5460 non-null   float64
 37  Eğitime Devam Etme Durumu_Etmiyor                     5460 non-null   float64
dtypes: float64(37), int8(1)
memory usage: 1.5 MB
```

In [45]:
```
xt.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2340 entries, 0 to 2339
Data columns (total 37 columns):
 #   Column                                                  Non-Null Count  Dtype  
---  ------                                                  --------------  -----  
 0   Yıllık Ortalama Gelir                                   2340 non-null   float64
 1   Yıllık Ortalama Satın Alım Miktarı                      2340 non-null   float64
 2   Yıllık Ortalama Sipariş Verilen Ürün Adedi              2340 non-null   float64
 3   Yıllık Ortalama Sepete Atılan Ürün Adedi                2340 non-null   float64
 4   Cinsiyet_Erkek                                          2340 non-null   float64
 5   Cinsiyet_Kadın                                          2340 non-null   float64
 6   Yaş Grubu_18-30                                         2340 non-null   float64
 7   Yaş Grubu_31-40                                         2340 non-null   float64
 8   Yaş Grubu_41-50                                         2340 non-null   float64
 9   Yaş Grubu_51-60                                         2340 non-null   float64
 10  Yaş Grubu_>60                                           2340 non-null   float64
 11  Medeni Durum_Bekar                                      2340 non-null   float64
 12  Medeni Durum_Evli                                       2340 non-null   float64
 13  Eğitim Düzeyi_Doktora Mezunu                            2340 non-null   float64
 14  Eğitim Düzeyi_Doktora Ötesi                             2340 non-null   float64
 15  Eğitim Düzeyi_Eğitimsiz                                 2340 non-null   float64
 16  Eğitim Düzeyi_Lise Mezunu                               2340 non-null   float64
 17  Eğitim Düzeyi_Ortaokul Mezunu                           2340 non-null   float64
 18  Eğitim Düzeyi_Yüksek Lisans Mezunu                      2340 non-null   float64
 19  Eğitim Düzeyi_Yüksekokul Mezunu                         2340 non-null   float64
 20  Eğitim Düzeyi_Üniversite Mezunu                         2340 non-null   float64
 21  Eğitim Düzeyi_İlkokul Mezunu                            2340 non-null   float64
 22  İstihdam Durumu_Düzenli ve Ücretli Bir İşi Var          2340 non-null   float64
 23  İstihdam Durumu_Emekli                                  2340 non-null   float64
 24  İstihdam Durumu_Kendi İşinin Sahibi                     2340 non-null   float64
 25  İstihdam Durumu_İşsiz veya Düzenli Bir İşi Yok          2340 non-null   float64
 26  Yaşadığı Şehir_Büyük Şehir                              2340 non-null   float64
 27  Yaşadığı Şehir_Köy veya Kasaba                          2340 non-null   float64
 28  Yaşadığı Şehir_Küçük Şehir                              2340 non-null   float64
 29  Yaşadığı Şehir_Kırsal                                   2340 non-null   float64
 30  En Çok İlgilendiği Ürün Grubu_Elektronik ve Teknolojik Ürünler  2340 non-null   float64
 31  En Çok İlgilendiği Ürün Grubu_Ev ve Mobilya             2340 non-null   float64
 32  En Çok İlgilendiği Ürün Grubu_Giyim                     2340 non-null   float64
 33  En Çok İlgilendiği Ürün Grubu_Kozmetik                  2340 non-null   float64
 34  En Çok İlgilendiği Ürün Grubu_Spor Malzemeleri          2340 non-null   float64
 35  Eğitime Devam Etme Durumu_Ediyor                        2340 non-null   float64
 36  Eğitime Devam Etme Durumu_Etmiyor                       2340 non-null   float64
dtypes: float64(37)
memory usage: 676.5 KB
```

Verilerin Eğitim için Bölünmesi ve Ölçüklendirilmesi

In [46]:
```
y=x.iloc[:,3:4].values
x.drop("ObekIsmi",axis=1,inplace=True)
x=x.values
xt=xt.values
```

In [47]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

In [48]:
```
from sklearn.preprocessing import StandardScaler
#from sklearn.preprocessing import PowerTransformer
#from sklearn.preprocessing import RobustScaler
#from sklearn.preprocessing import MinMaxScaler
sc=StandardScaler()
x_train=sc.fit_transform(X_train)
x_test=sc.transform(X_test)
xt_test=sc.transform(xt)
```

In [49]:
```
#LDA
# Veri boyutlarını indirgemek ve sınıflar arasındaki mesafeyi maksimize etmek için tercih edildi. Veri normal d
# göstermediği için PCA tercih edilmedi.
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
lda=LDA(n_components=6)
x_train_lda=lda.fit_transform(x_train,Y_train)
x_test_lda=lda.transform(x_test)
predict_test_lda=lda.transform(xt_test)
```

```
C:\Users\Dell\anaconda3\Lib\site-packages\sklearn\utils\validation.py:1184: DataConversionWarning: A column-vec
tor y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
```

In [50]:
```
#Başarı ölçüm mekriklerinin içe aktarılması
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
```

Makine Öğrenmesi Algoritmalarının Oluşturulması ve Eğitilmesi

In [51]:
```
#Logistic Regression

rState=0
from sklearn.linear_model import LogisticRegression
classifier=LogisticRegression(random_state=rState)
classifier.fit(x_train, Y_train)

y_pred_lr=classifier.predict(x_test)
cm_lr=confusion_matrix(Y_test, y_pred_lr)
print(cm_lr)
acc_lr=accuracy_score(Y_test, y_pred_lr)
print(acc_lr)

print("--------LDA--------")
classifier_lda=LogisticRegression(random_state=rState)
classifier_lda.fit(x_train_lda, Y_train)

y_pred_lr_lda=classifier_lda.predict(x_test_lda)
cm_lr_lda=confusion_matrix(Y_test,y_pred_lr_lda)
print(cm_lr_lda)
acc_lr_lda=accuracy_score(Y_test, y_pred_lr_lda)
print(acc_lr_lda)
```

```
C:\Users\Dell\anaconda3\Lib\site-packages\sklearn\utils\validation.py:1184: DataConversionWarning: A column-vec
tor y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
[[120   1   4   3   0   0   0   1]
 [  1 105   0   0   0   2   3   1]
 [  0   2 118   0   1   3   2   0]
 [  2   1   1 140   2   1   1   2]
 [  2   2   0   0 134   0   1   1]
 [  2   1   0   0   2 138   1   0]
 [  1   1   2   1   2   0 146   0]
 [  0   1   0   1   2   1   0 133]]
0.9468864468864469
---------LDA---------
[[121   2   2   3   0   0   0   1]
 [  1 104   1   0   0   2   3   1]
 [  2   1 118   0   1   3   1   0]
 [  3   1   0 141   2   1   0   2]
 [  2   2   0   0 134   0   1   1]
 [  2   1   0   0   2 138   1   0]
 [  1   1   2   1   2   0 146   0]
 [  0   1   0   1   2   1   0 133]]
0.9478021978021978
C:\Users\Dell\anaconda3\Lib\site-packages\sklearn\utils\validation.py:1184: DataConversionWarning: A column-vec
tor y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
```

In [52]:
```
#KNN Classifier

n=30
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=n,metric="minkowski")
knn.fit(x_train, Y_train)

y_pred_knn=knn.predict(x_test)
cm_knn=confusion_matrix(Y_test, y_pred_knn)
print(cm_knn)
acc_knn=accuracy_score(Y_test, y_pred_knn)
print(acc_knn)

print("--------LDA--------")
knn_lda=KNeighborsClassifier(n_neighbors=n,metric="minkowski")
knn_lda.fit(x_train_lda, Y_train)

y_pred_knn_lda=knn_lda.predict(x_test_lda)
cm_knn_lda=confusion_matrix(Y_test,y_pred_knn_lda)
print(cm_knn_lda)
acc_knn_lda=accuracy_score(Y_test, y_pred_knn_lda)
print(acc_knn_lda)
```

```
C:\Users\Dell\anaconda3\Lib\site-packages\sklearn\neighbors\_classification.py:228: DataConversionWarning: A co
lumn-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for exampl
e using ravel().
  return self._fit(X, y)
C:\Users\Dell\anaconda3\Lib\site-packages\joblib\externals\loky\backend\context.py:110: UserWarning: Could not
find the number of physical cores for the following reason:
found 0 physical cores < 1
Returning the number of logical cores instead. You can silence this warning by setting LOKY_MAX_CPU_COUNT to th
e number of cores you want to use.
  warnings.warn(
  File "C:\Users\Dell\anaconda3\Lib\site-packages\joblib\externals\loky\backend\context.py", line 217, in _coun
t_physical_cores
    raise ValueError(
```

```
[[ 83  18  12   3   4   2   6   1]
 [  6  87   4   1   3   3   7   1]
 [ 12   4  80   0  23   3   4   0]
 [  2   1   0 136   3   1   3   4]
 [  5   7   1   0 118   0   5   4]
 [  4   2   1   1   3 124   9   0]
 [  8   2   3   7  16   3 110   4]
 [  0   1   0   4   1   1   0 131]]
0.7957875457875457
---------LDA---------
[[123   0   2   3   0   0   0   1]
 [  1 105   0   0   0   2   3   1]
 [  1   1 119   0   1   3   1   0]
 [  2   1   1 141   2   1   0   2]
 [  2   1   0   0 135   0   1   1]
 [  2   1   0   0   2 138   1   0]
 [  1   1   2   1   2   0 146   0]
 [  0   1   0   1   2   1   0 133]]
0.9523809523809523
```

```
C:\Users\Dell\anaconda3\Lib\site-packages\sklearn\neighbors\_classification.py:228: DataConversionWarning: A co
lumn-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for exampl
e using ravel().
  return self._fit(X, y)
```

In [53]:
```
#SVM

kernel="linear"
from sklearn.svm import SVC
svc=SVC(kernel=kernel)
svc.fit(x_train,Y_train)

y_pred_svc=svc.predict(x_test)
cm_svc=confusion_matrix(Y_test,y_pred_svc)
print(cm_svc)
acc_svc=accuracy_score(Y_test,y_pred_svc)
print(acc_svc)

print("---------LDA---------")
svc_lda=SVC(kernel=kernel)
svc_lda.fit(x_train_lda, Y_train)

y_pred_svc_lda=svc_lda.predict(x_test_lda)
cm_svc_lda=confusion_matrix(Y_test,y_pred_svc_lda)
print(cm_svc_lda)
acc_svc_lda=accuracy_score(Y_test, y_pred_svc_lda)
print(acc_svc_lda)
```

```
C:\Users\Dell\anaconda3\Lib\site-packages\sklearn\utils\validation.py:1184: DataConversionWarning: A column-vec
tor y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
[[123   1   1   3   0   0   0   1]
 [  1 105   0   0   0   2   3   1]
 [  0   0 121   0   1   3   1   0]
 [  2   1   1 140   2   1   1   2]
 [  2   2   0   0 134   0   1   1]
 [  2   1   0   0   2 138   1   0]
 [  1   1   2   1   2   0 146   0]
 [  0   1   0   1   2   1   0 133]]
0.9523809523809523
---------LDA---------
[[124   0   1   3   0   0   0   1]
 [  1 105   0   0   0   2   3   1]
 [  1   1 119   0   1   3   1   0]
 [  3   1   0 141   2   1   0   2]
 [  2   2   0   0 134   0   1   1]
 [  2   1   0   0   2 138   1   0]
 [  1   1   2   1   2   0 146   0]
 [  0   1   0   1   2   1   0 133]]
0.9523809523809523
```

```
C:\Users\Dell\anaconda3\Lib\site-packages\sklearn\utils\validation.py:1184: DataConversionWarning: A column-vec
tor y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
```

In [54]:
```
#Naive Bayes

from sklearn.naive_bayes import GaussianNB
gnb=GaussianNB()
gnb.fit(x_train,Y_train)

y_pred_nb=gnb.predict(x_test)
cm_nb=confusion_matrix(Y_test,y_pred_nb)
print(cm_nb)
acc_nb=accuracy_score(Y_test,y_pred_nb)
print(acc_nb)
```

```
print("---------LDA---------")
gnb_lda=GaussianNB()
gnb_lda.fit(x_train_lda, Y_train)

y_pred_gnb_lda=gnb_lda.predict(x_test_lda)
cm_gnb_lda=confusion_matrix(Y_test,y_pred_gnb_lda)
print(cm_gnb_lda)
acc_gnb_lda=accuracy_score(Y_test, y_pred_gnb_lda)
print(acc_gnb_lda)
```

```
[[117   1   7   3   0   0   0   1]
 [  3 101   0   1   2   2   2   1]
 [  3   2 115   0   1   3   2   0]
 [  3   1   0 141   2   1   0   2]
 [  4   6   0   0 124   0   2   4]
 [  1   1   1   0   2 136   3   0]
 [  2   0   2   3   3   1 142   0]
 [  0   1   0   1   2   1   0 133]]
0.923992673992674
---------LDA---------
[[121   1   3   3   0   0   0   1]
 [  1 103   2   0   0   2   3   1]
 [  6   0 115   0   1   3   1   0]
 [  3   1   0 141   2   1   0   2]
 [  2   1   0   0 135   0   1   1]
 [  2   1   0   0   2 138   1   0]
 [  1   1   2   1   2   0 146   0]
 [  0   1   0   1   2   1   0 133]]
0.945054945054945
```

```
C:\Users\Dell\anaconda3\Lib\site-packages\sklearn\utils\validation.py:1184: DataConversionWarning: A column-vec
tor y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
C:\Users\Dell\anaconda3\Lib\site-packages\sklearn\utils\validation.py:1184: DataConversionWarning: A column-vec
tor y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
```

In [55]:

```
#Decision Tree

n=90
crit="entropy"
rState=5
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier(criterion=crit,random_state=rState)
dtc.fit(x_train,Y_train)

y_pred_dtc=dtc.predict(x_test)
cm_dtc=confusion_matrix(Y_test,y_pred_dtc)
print(cm_dtc)
acc_dtc=accuracy_score(Y_test,y_pred_dtc)
print(acc_dtc)

print("---------LDA---------")
dtc_lda=DecisionTreeClassifier(criterion=crit,random_state=rState)
dtc_lda.fit(x_train_lda, Y_train)

y_pred_dtc_lda=dtc_lda.predict(x_test_lda)
cm_dtc_lda=confusion_matrix(Y_test,y_pred_dtc_lda)
print(cm_dtc_lda)
acc_dtc_lda=accuracy_score(Y_test, y_pred_dtc_lda)
print(acc_dtc_lda)
```

```
[[121   0   2   3   1   0   1   1]
 [  2 100   2   0   0   3   3   2]
 [  2   3 114   2   1   2   1   1]
 [  3   2   1 135   1   2   4   2]
 [  4   3   0   1 129   0   1   2]
 [  2   2   1   1   3 131   4   0]
 [  2   4   3   3   2   3 132   4]
 [  1   1   1   2   4   2   0 127]]
0.9056776556776557
---------LDA---------
[[118   0   4   3   0   1   0   3]
 [  1 103   0   1   0   3   3   1]
 [  3   0 110   2   4   5   2   0]
 [  2   2   0 136   2   2   3   3]
 [  3   2   0   1 130   0   1   3]
 [  2   4   3   3   2 129   1   0]
 [  3   2   4   3   2   1 136   2]
 [  1   2   1   1   3   2   0 128]]
0.9065934065934066
```

In [56]:

```
#Random Forest

n=72
crit="entropy"
rState=16
from sklearn.ensemble import RandomForestClassifier
```

```
rfc=RandomForestClassifier(n_estimators=n,criterion=crit,random_state=rState)
rfc.fit(x_train,Y_train)

y_pred_rfc=rfc.predict(x_test)
cm_rfc=confusion_matrix(Y_test,y_pred_rfc)
print(cm_rfc)
acc_rfc=accuracy_score(Y_test,y_pred_rfc)
print(acc_rfc)

print("---------LDA---------")
rfc_lda=RandomForestClassifier(n_estimators=n,criterion=crit,random_state=rState)
rfc_lda.fit(x_train_lda, Y_train)

y_pred_rfc_lda=dtc_lda.predict(x_test_lda)
cm_rfc_lda=confusion_matrix(Y_test,y_pred_rfc_lda)
print(cm_rfc_lda)
acc_rfc_lda=accuracy_score(Y_test, y_pred_rfc_lda)
print(acc_rfc_lda)
```

```
C:\Users\Dell\anaconda3\Lib\site-packages\sklearn\base.py:1151: DataConversionWarning: A column-vector y was pa
ssed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  return fit_method(estimator, *args, **kwargs)
[[124   0   1   3   0   0   0   1]
 [  1 105   0   0   0   2   3   1]
 [  0   0 121   0   1   3   1   0]
 [  2   1   1 141   2   1   0   2]
 [  2   1   0   0 135   0   1   1]
 [  2   1   0   0   2 138   1   0]
 [  1   1   2   1   2   0 146   0]
 [  0   1   0   1   2   1   0 133]]
0.9551282051282052
---------LDA---------
C:\Users\Dell\anaconda3\Lib\site-packages\sklearn\base.py:1151: DataConversionWarning: A column-vector y was pa
ssed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
  return fit_method(estimator, *args, **kwargs)
[[118   0   4   3   0   1   0   3]
 [  1 103   0   1   0   3   3   1]
 [  3   0 110   2   4   5   2   0]
 [  2   2   0 136   2   2   3   3]
 [  3   2   0   1 130   0   1   3]
 [  2   4   3   3   2 129   1   0]
 [  3   2   4   3   2   1 136   2]
 [  1   2   1   1   3   2   0 128]]
0.9065934065934066
```

In [57]:

```
#XGboost

n=100
maxDept=10
lRate=0.01
rState=5
from xgboost import XGBClassifier
xgb=XGBClassifier(n_estimators=n, max_depth=maxDept, learning_rate=lRate, random_state=rState)
xgb.fit(x_train,Y_train)

y_pred_xgb=xgb.predict(x_test)
cm_xgb=confusion_matrix(Y_test,y_pred_xgb)
print(cm_xgb)
acc_xgb=accuracy_score(Y_test,y_pred_xgb)
print(acc_xgb)

print("---------LDA---------")
xgb_lda=XGBClassifier(n_estimators=n, max_depth=maxDept, learning_rate=lRate, random_state=rState)
xgb_lda.fit(x_train_lda, Y_train)

y_pred_xgb_lda=xgb_lda.predict(x_test_lda)
cm_xgb_lda=confusion_matrix(Y_test,y_pred_xgb_lda)
print(cm_xgb_lda)
acc_xgb_lda=accuracy_score(Y_test, y_pred_xgb_lda)
print(acc_xgb_lda)
```

```
[[124   0   1   3   0   0   0   1]
 [  2 102   0   0   2   2   3   1]
 [  0   0 121   0   1   3   1   0]
 [  2   1   1 141   2   1   0   2]
 [  3   1   0   0 134   0   1   1]
 [  2   1   0   0   2 137   2   0]
 [  1   1   2   1   2   0 146   0]
 [  0   1   0   1   2   1   0 133]]
0.9505494505494505
---------LDA---------
[[124   0   1   3   0   0   0   1]
 [  1 105   0   0   0   2   3   1]
 [  0   0 120   1   1   3   1   0]
 [  2   1   1 139   2   2   1   2]
 [  2   1   0   0 135   0   1   1]
 [  2   1   0   0   2 138   1   0]
 [  1   1   2   2   2   0 145   0]
 [  0   1   0   1   2   1   0 133]]
0.9514652014652014
```

In [58]:
```
#XGboost RFClassifier

n=95
maxDept=7
lRate=0.001
#obj='binary:logistic'
r_s_xgbrfc=7
from xgboost import XGBRFClassifier
xgbrfc=XGBRFClassifier(n_estimators=n, max_depth=maxDept, learning_rate=lRate, random_state=r_s_xgbrfc)
xgbrfc.fit(x_train,Y_train)

y_pred_xgbrfc=xgbrfc.predict(x_test)
cm_xgbrfc=confusion_matrix(Y_test,y_pred_xgbrfc)
print(cm_xgb)
acc_xgbrfc=accuracy_score(Y_test,y_pred_xgbrfc)
print(acc_xgbrfc)

print("---------LDA---------")
xgbrfc_lda=XGBRFClassifier(n_estimators=n, max_depth=maxDept, learning_rate=lRate, random_state=r_s_xgbrfc)
xgbrfc_lda.fit(x_train_lda, Y_train)

y_pred_xgbrfc_lda=xgbrfc_lda.predict(x_test_lda)
cm_xgbrfc_lda=confusion_matrix(Y_test,y_pred_xgbrfc_lda)
print(cm_xgbrfc_lda)
acc_xgbrfc_lda=accuracy_score(Y_test, y_pred_xgbrfc_lda)
print(acc_xgbrfc_lda)
```

```
[[124   0   1   3   0   0   0   1]
 [  2 102   0   0   2   2   3   1]
 [  0   0 121   0   1   3   1   0]
 [  2   1   1 141   2   1   0   2]
 [  3   1   0   0 134   0   1   1]
 [  2   1   0   0   2 137   2   0]
 [  1   1   2   1   2   0 146   0]
 [  0   1   0   1   2   1   0 133]]
0.9542124542124543
---------LDA---------
[[124   0   1   3   0   0   0   1]
 [  1 105   0   0   0   2   3   1]
 [  0   0 121   0   1   3   1   0]
 [  2   1   1 141   2   1   0   2]
 [  2   1   0   0 135   0   1   1]
 [  2   1   0   0   2 138   1   0]
 [  1   1   2   1   2   0 146   0]
 [  0   1   0   1   2   1   0 133]]
0.9551282051282052
```

In [59]:
```
#Stochastic Gradient Descent

from sklearn.linear_model import SGDClassifier
sgdc=SGDClassifier(max_iter=45,loss="hinge")
sgdc.fit(x_train,Y_train)

y_pred_sgdc=sgdc.predict(x_test)
cm_sgdc=confusion_matrix(Y_test,y_pred_sgdc)
print(cm_sgdc)
acc_sgdc=accuracy_score(Y_test,y_pred_sgdc)
print(acc_sgdc)

print("---------LDA---------")
sgdc_lda=sgdc=SGDClassifier(max_iter=21,loss="hinge",penalty="l1")
sgdc_lda.fit(x_train_lda, Y_train)

y_pred_sgdc_lda=sgdc_lda.predict(x_test_lda)
cm_sgdc_lda=confusion_matrix(Y_test,y_pred_sgdc_lda)
print(cm_sgdc_lda)
acc_sgdc_lda=accuracy_score(Y_test, y_pred_sgdc_lda)
print(acc_sgdc_lda)
```

```
[[121   2   2   3   0   0   0   1]
 [  2 100   2   0   1   2   3   2]
 [  5   2 111   2   1   3   2   0]
 [  2   2   0 140   2   2   0   2]
 [  2   2   0   0 132   0   3   1]
 [  2   1   0   0   2 138   1   0]
 [  2   0   2   4   2   0 142   1]
 [  0   1   0   1   5   1   0 130]]
0.9285714285714286
---------LDA---------
[[110   1  14   3   0   0   0   1]
 [  1 103   1   0   1   2   3   1]
 [  3   0 117   1   1   3   1   0]
 [  3   1   0 141   2   1   0   2]
 [  2   2   0   0 134   0   1   1]
 [  1   1   1   0   2 138   1   0]
 [  1   1   2   1   2   0 146   0]
 [  0   1   0   1   2   1   0 133]]
0.9358974358974359
```

```
C:\Users\Dell\anaconda3\Lib\site-packages\sklearn\utils\validation.py:1184: DataConversionWarning: A column-vec
tor y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
C:\Users\Dell\anaconda3\Lib\site-packages\sklearn\linear_model\_stochastic_gradient.py:713:  ConvergenceWarning:
Maximum number of iteration reached before convergence. Consider increasing max_iter to improve the fit.
  warnings.warn(
C:\Users\Dell\anaconda3\Lib\site-packages\sklearn\utils\validation.py:1184: DataConversionWarning: A column-vec
tor y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
C:\Users\Dell\anaconda3\Lib\site-packages\sklearn\linear_model\_stochastic_gradient.py:713:  ConvergenceWarning:
Maximum number of iteration reached before convergence. Consider increasing max_iter to improve the fit.
  warnings.warn(
```

In [60]:
```python
#VotingClassifier

from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from xgboost import XGBRFClassifier
from xgboost import XGBClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.model_selection import cross_val_score

clf1=LogisticRegression(random_state=0)
clf2=RandomForestClassifier(n_estimators=78,criterion="entropy",random_state=14)
clf3=SVC(kernel="linear")
clf4=XGBRFClassifier(n_estimators=72, max_depth=9, learning_rate=0.01, random_state=42)
clf5=XGBClassifier(n_estimators=100, max_depth=10, learning_rate=0.01,  random_state=5)

vclf=VotingClassifier(estimators=[("lr",clf1),("rfc",clf2),("svc",clf3),("xgbrfc",clf4),("xgbc",clf5)],voting="

vclf.fit(x_train,Y_train)
y_pred_vclf=vclf.predict(x_test)
print(f"Vating Classifier Accuracy Score: {accuracy_score(Y_test, y_pred_vclf)}")
```

```
C:\Users\Dell\anaconda3\Lib\site-packages\sklearn\preprocessing\_label.py:97: DataConversionWarning: A column-v
ector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example usi
ng ravel().
  y = column_or_1d(y, warn=True)
C:\Users\Dell\anaconda3\Lib\site-packages\sklearn\preprocessing\_label.py:132: DataConversionWarning: A column-
vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example us
ing ravel().
  y = column_or_1d(y, dtype=self.classes_.dtype, warn=True)
Vating Classifier Accuracy Score: 0.9551282051282052
```

In [61]:
```python
#Sayısal veriye dönüştürdüğümüz Öbek İsmi sutununu tahmin dosyasını oluşturmak üzere tekrar metinsel ifadeye dö
def donustur(liste):
    kolon=testVerileri.iloc[:,0:1]
    kolon.rename(columns={"index":"id"},inplace=True)
    i=0
    liste=list(liste)
    while i<len(liste):
        if liste[i]==0:
            liste[i]="obek_1"
        if liste[i]==1:
            liste[i]="obek_2"
        if liste[i]==2:
            liste[i]="obek_3"
        if liste[i]==3:
            liste[i]="obek_4"
        if liste[i]==4:
            liste[i]="obek_5"
        if liste[i]==5:
            liste[i]="obek_6"
        if liste[i]==6:
            liste[i]="obek_7"
        if liste[i]==7:
```

```
            liste[i]="obek_8"
        i=i+1
    liste=pd.DataFrame(data=liste,columns=["Öbek İsmi"])
    kolon=pd.concat([kolon,liste],axis=1)
    return kolon
```

Tahminler

In [ ]:
```
# Logistic Regression Tahmin

# Parametreler

# test size oranı -> 0.2
# split random state -> 0
# Scaler -> Standart Scaler
# Lda -> Kullanıldı, n sayısı=7
# Random state -> 0
# Accuracy değeri -> 0,9478

predictList_lr_lda=classifier_lda.predict(predict_test_lda)
print(predictList_lr_lda)
sonuc=donustur(predictList_lr_lda)
print(sonuc)
sonuc.to_csv("C:\\Users\\Dell\\Desktop\\Lrc_lda predict list.csv",index=False)
```

In [ ]:
```
#  Knn  tahmin

# Parametreler

# test size oranı -> 0.2
# split random state -> 0
# Scaler -> Standart Scaler
# Lda -> Kullanıldı, n sayısı=7
# Knn n sayısı -> 30
# metrik -> minkowski
# Accuracy değeri -> 0,9551

predictList_knn_lda=knn_lda.predict(predict_test_lda)
print(predictList_knn_lda)
sonuc=donustur(predictList_knn_lda)
print(sonuc)
sonuc.to_csv("C:\\Users\\Dell\\Desktop\\Knn_lda predict list.csv",index=False)
```

In [ ]:
```
# Xgboost tahmin

# Parametreler

# test size oranı -> 0.2
# split random state -> 0
# Scaler -> Standart Scaler
# Lda -> Kullanılmadı
# Xgboost n sayısı -> 60
# Max dept -> 7
# Learnin rate -> 0.01
# Random state -> 5
# Accuracy değeri -> 0,9505

predictList_xgb=xgb.predict(xt_test)
print(predictList_xgb)
sonuc=donustur(predictList_xgb)
print(sonuc)
sonuc.to_csv("C:\\Users\\Dell\\Desktop\\XGBoost predict list.csv",index=False)
```

In [ ]:
```
# SVM tahmin

# test size oranı -> 0.2
# split random state -> 0
# Scaler -> Min Max Scaler
# Lda -> Kullanılmadı
# Kernel -> Linear
# Accuracy değeri -> 0,9523

predictList_svc=svc.predict(xt_test)
print(predictList_svc)
sonuc=donustur(predictList_svc)
print(sonuc)
sonuc.to_csv("C:\\Users\\Dell\\Desktop\\SVM predict list.csv",index=False)
```

In [ ]:
```
# Random Forest tahmin

# test size oranı -> 0.2
# split random state -> 0
# Scaler -> Min Max Scaler
# Lda -> Kullanılmadı
```

```
# Criterion -> Entropy
# RF n sayısı -> 78
# Random state -> 14
# Accuracy değeri -> 0,9551

predictList_rfc=rfc.predict(xt_test)
print(predictList_rfc)
sonuc=donustur(predictList_rfc)
print(sonuc)
sonuc.to_csv("C:\\Users\\Dell\\Desktop\\RFC predict list.csv",index=False)
```

In [ ]:
```
# XGBRFC tahmin

# Parametreler

# test size oranı -> 0.2
# split random state -> 0
# Scaler -> Min Max Scaler
# Lda -> Kullanıldı, n=6
# Xgbrfc n sayısı -> 72
# Max dept -> 9
# Learnin rate -> 0.01
# Random state -> 42
# Accuracy değeri -> 0,9551

predictList_xgbrfc_lda=xgbrfc_lda.predict(predict_test_lda)
print(predictList_xgbrfc_lda)
sonuc=donustur(predictList_xgbrfc_lda)
print(sonuc)
sonuc.to_csv("C:\\Users\\Dell\\Desktop\\XGBRFC predict list.csv",index=False)
```

In [ ]:
```
# Xgboost tahmin

# Parametreler

# test size oranı -> 0.2
# split random state -> 0
# Scaler -> Standart Scaler
# Lda -> Kullanıldı, n=6
# Xgboost n sayısı -> 95
# Max dept -> 7
# Learnin rate -> 0.001
# Random state -> 7
# Accuracy değeri -> 0,9542

predictList_xgb=xgb.predict(xt_test)
print(predictList_xgb)
sonuc=donustur(predictList_xgb)
print(sonuc)
sonuc.to_csv("C:\\Users\\Dell\\Desktop\\XGBoost 2 predict list.csv",index=False)
```

In [ ]:
```
# Naive Bayes

# Parametreler

# test size oranı -> 0.2
# split random state -> 0
# Scaler -> Standart Scaler
# Lda -> Kullanıldı, n=6
# Accuracy değeri -> 0,9450

predictList_gnb_lda=gnb_lda.predict(predict_test_lda)
print(predictList_gnb_lda)
sonuc=donustur(predictList_gnb_lda)
print(sonuc)
sonuc.to_csv("C:\\Users\\Dell\\Desktop\\Naive Bayes predict list.csv",index=False)
```

In [ ]:
```
#VotingClassifier tahmin

# Parametreler

# test size oranı -> 0.2
# split random state -> 0
# Scaler -> Standart Scaler
# Lda -> Kullanılmadı
# Logistic Regression -> random state=0
# RandomForestClassifier -> n=7, criterion = entropy, random state=14
# SVC -> kernel=linear
# XGBRFClassifier -> n=72, max_depth=9, learning rate=0.01, random state=42
# XGBClassifier -> n=100, max_depth=10, learning rate=0.01, random state=5
# VotingClassifier -> voting = hard
# Accuracy değeri -> 0,9551

predictList_vclf=vclf.predict(xt_test)
print(predictList_vclf)
sonuc=donustur(predictList_vclf)
print(sonuc)
```

```
sonuc.to_csv("C:\\Users\\Dell\\Desktop\\VotingClassifier predict list.csv",index=False)
```

In [ ]: