Customer Segmentation Project

Müşteri Segmentasyonu, denetimsiz öğrenmenin en önemli uygulamalarından biridir. Kümeleme teknikleri sayesinde **B2C** (İşletmeden Tüketiciye) şirketler, pazarlama açısından cinsiyet, yaş, ilgi alanları ve çeşitli harcama alışkanlıkları gibi farklı yönleriyle benzerlik gösteren müşteri segmentlerini belirleyebilir. Bu, potansiyel kullanıcı kitlesine yönelik hedefleme imkanı sağlar.

- Bu Veri Bilimi Projesi'nde, makine öğreniminin en temel uygulamalarından biri olan Müşteri Segmentasyonunu gerçekleştireceğiz. Bu projede, müşteri segmentasyonunu Python'da uygulayacağız. En iyi müşterinizi bulmanız gerektiğinde, müşteri segmentasyonu ideal bir metodolojidir.

Müşteri segmentasyonu projemize veriyi keşfederek başlayacağız. Verimiz, her bir müşteriyi tanımlayan aşağıdaki özellikleri içeren bir .csv dosyasından oluşmaktadır:

- **Müşteri Kimlikleri**
-**Yaş**
-**Cinsiyet**
-**Yıllık Gelir**.
-**Harcama Puanı**

In [ ]:
```
import pandas as pd                    # Pandas kütüphanesini içe aktarıyoruz, veri
import numpy as np                     # NumPy kütüphanesini içe aktarıyoruz, sayısa
import matplotlib.pyplot as plt        # Matplotlib kütüphanesini içe aktarıyoruz, gr
import seaborn as sns                  # Seaborn kütüphanesini içe aktarıyoruz, veri
import warnings                        # Warnings modülünü içe aktarıyoruz, uyarı mes

warnings.simplefilter("ignore", category=FutureWarning)   # Geleceğe yönelik uyarı
```

In [ ]:
```
customer = pd.read_csv('data/Mall_Customers.csv')   # 'data/Mall_Customers.csv' dos
df = customer.copy()                                # customer veri çerçevesini df
df.head()                                           # df veri çerçevesinin başlang
```

Out[ ]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

In [ ]:
```
df.describe().T # Veri çerçevesinin betimsel istatistiklerini görüntülüyoruz
```

Out[ ]:

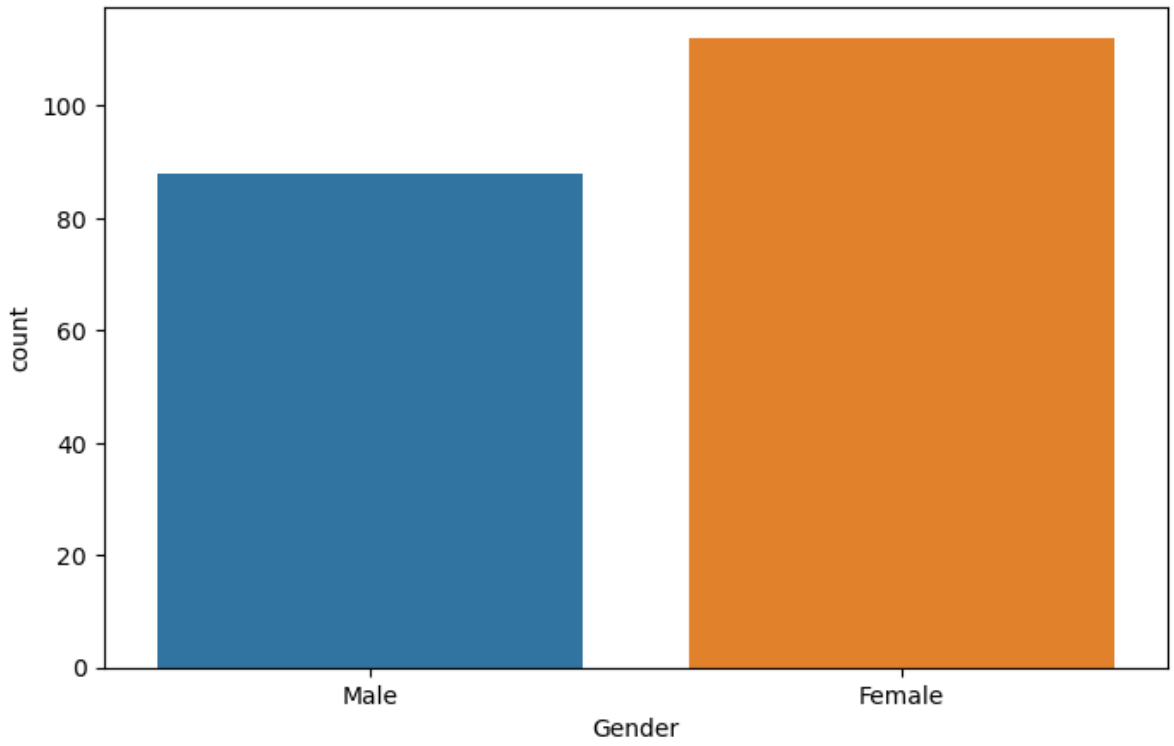
	count	mean	std	min	25%	50%	75%	max
CustomerID	200.0	100.50	57.879185	1.0	50.75	100.5	150.25	200.0
Age	200.0	38.85	13.969007	18.0	28.75	36.0	49.00	70.0
Annual Income (k$)	200.0	60.56	26.264721	15.0	41.50	61.5	78.00	137.0
Spending Score (1-100)	200.0	50.20	25.823522	1.0	34.75	50.0	73.00	99.0

Verilerden daha iyi anlaşılırşekilde elde etmek için daha fazla araştırmamız gerekiyor. Bunu yapmak için daha sonra veri setinin her bir özelliği üzerinde tek değişkenli bir analiz yapacağız.

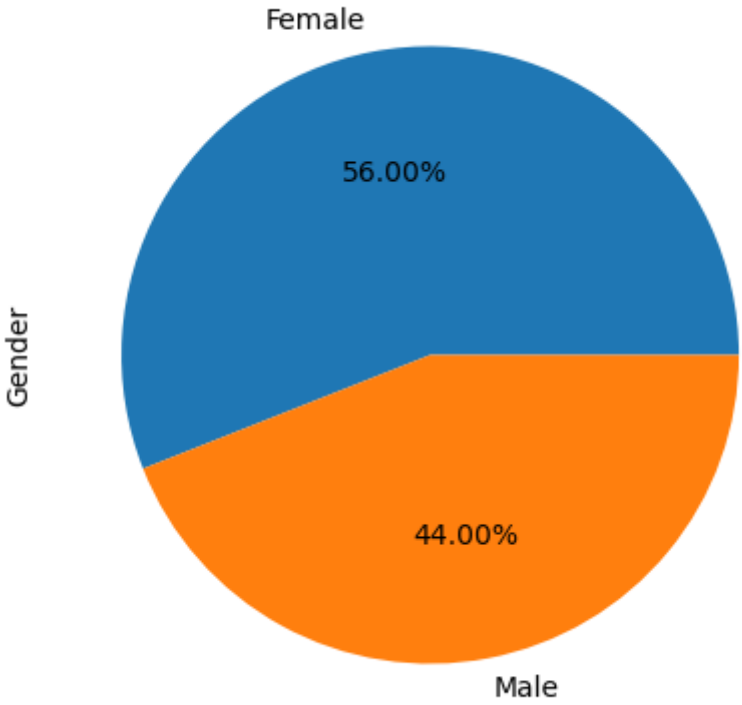
Cinsiyet :

- Burada, müşterilerimizin veri kümesindeki cinsiyet dağılımını göstermek için bir barplot ve bir piechart oluşturacağız.

In [ ]:
```
plt.figure(figsize=(8,5))           # Yeni bir grafik figürü oluşturuluyor ve boyutu
sns.countplot(x='Gender', data=df) # Seaborn kütüphanesinin countplot() fonksiyonu
plt.show()                          # Oluşturulan grafiği göstermek için kullanılıy
```



In [ ]:
```
plt.figure(figsize=(8,5))   # Yeni bir grafik figürü oluşturuluyor ve boyutu belir
df.Gender.value_counts().plot.pie(autopct='%.2f%%')
# 'Gender' sütunundaki değerleri sayarak bir pasta grafiği çiziliyor, autopct='%2.
plt.show()  # Oluşturulan grafiği göstermek için kullanılıyor
```



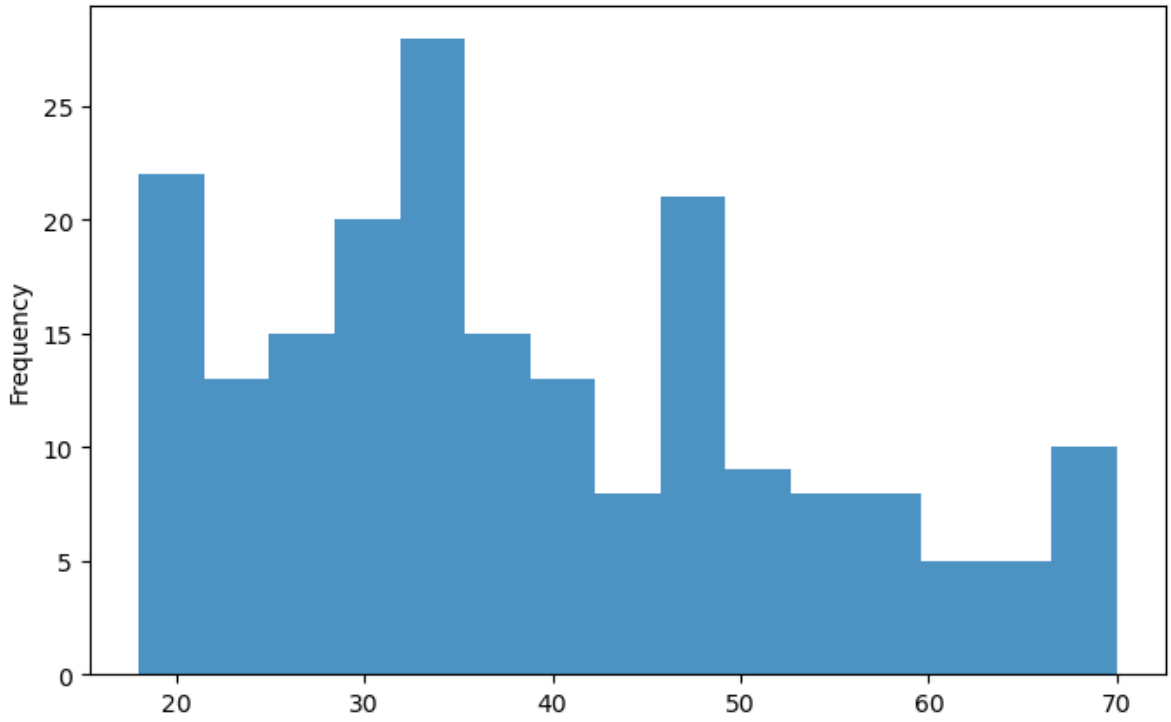
- Yukarıdaki grafikten, müşteri veri setimizde kadın yüzdesinin %56, erkek yüzdesinin ise %44 olduğu sonucuna vardık.

- **Yaş :**
  - Next, we analyse the age feature, it's a continuous variable, thus we will plot a histogram to view the distribution of customer ages. We will first proceed by taking summary of the this variable.

In [ ]:
```
df.Age.describe() # 'Age' sütununun betimsel istatistiklerini görüntülüyoruz
```

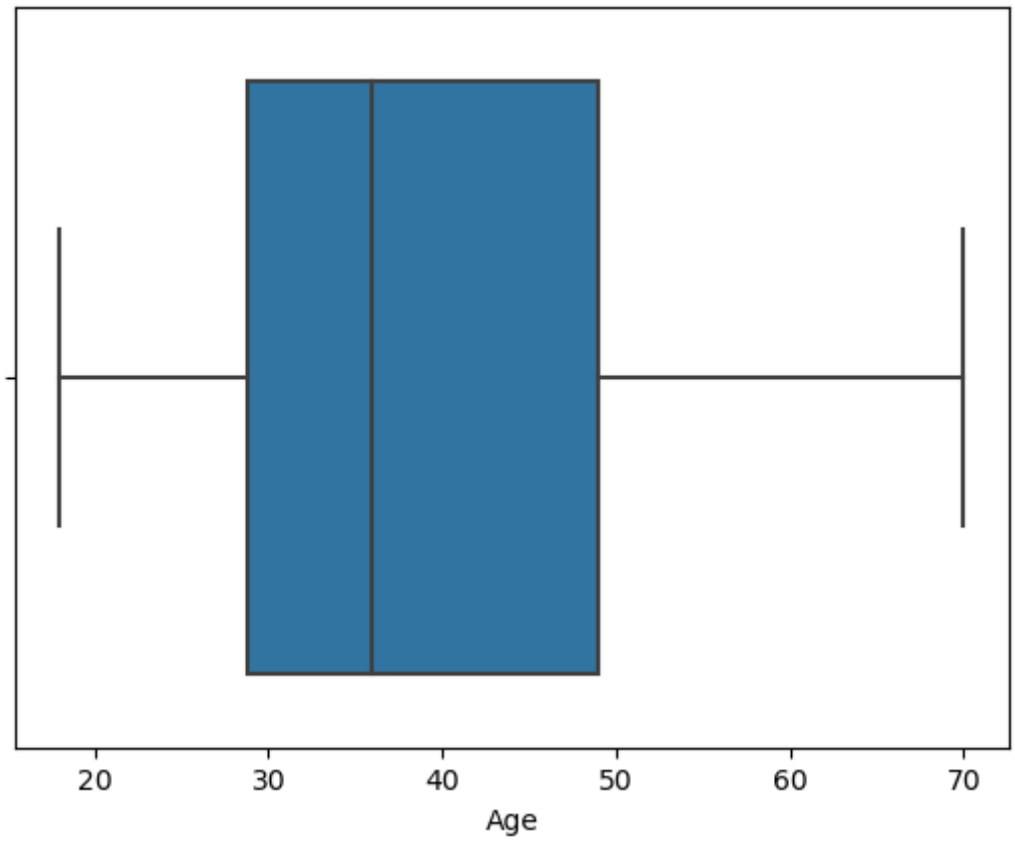
Out[ ]:
```
count    200.000000
mean      38.850000
std       13.969007
min       18.000000
25%       28.750000
50%       36.000000
75%       49.000000
max       70.000000
Name: Age, dtype: float64
```

In [ ]:
```
plt.figure(figsize=(8,5)) # Yeni bir grafik figürü oluşturuluyor ve boyutu belirle
df.Age.plot.hist(bins=15,alpha=.8) # 'Age' sütunundaki değerlerin histogram grafiğ
plt.show() # Oluşturulan grafiği göstermek için kullanılıyor
```



In [ ]:
```
sns.boxplot(x='Age',data=df) # Seaborn kütüphanesinin boxplot() fonksiyonuyla 'Age
```

Out[ ]:
```
<Axes: xlabel='Age'>
```



- From the above graphs, we can obviously conclude that most of the customers have an age between 30 and 35, also the minimum age of customers is 18, whereas the maximum age is 70.

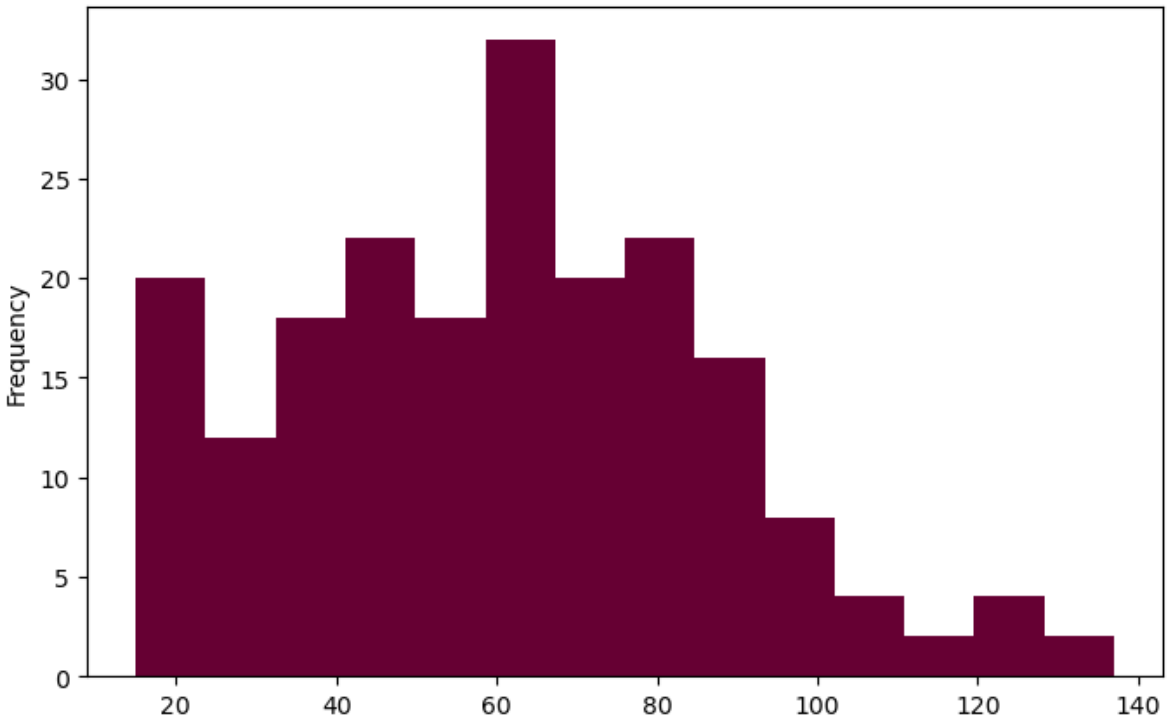
- **Yıllık gelir:**
  - Şimdi, bu özellikten içgörüler elde etmek için bir histogram ve yoğunluk çizimleri kullanarak yıllık gelir özelliğini keşfedeceğiz.

In [ ]:
```
df['Annual Income (k$)'].describe() # 'Annual Income (k$)' sütununun betimsel ista
```

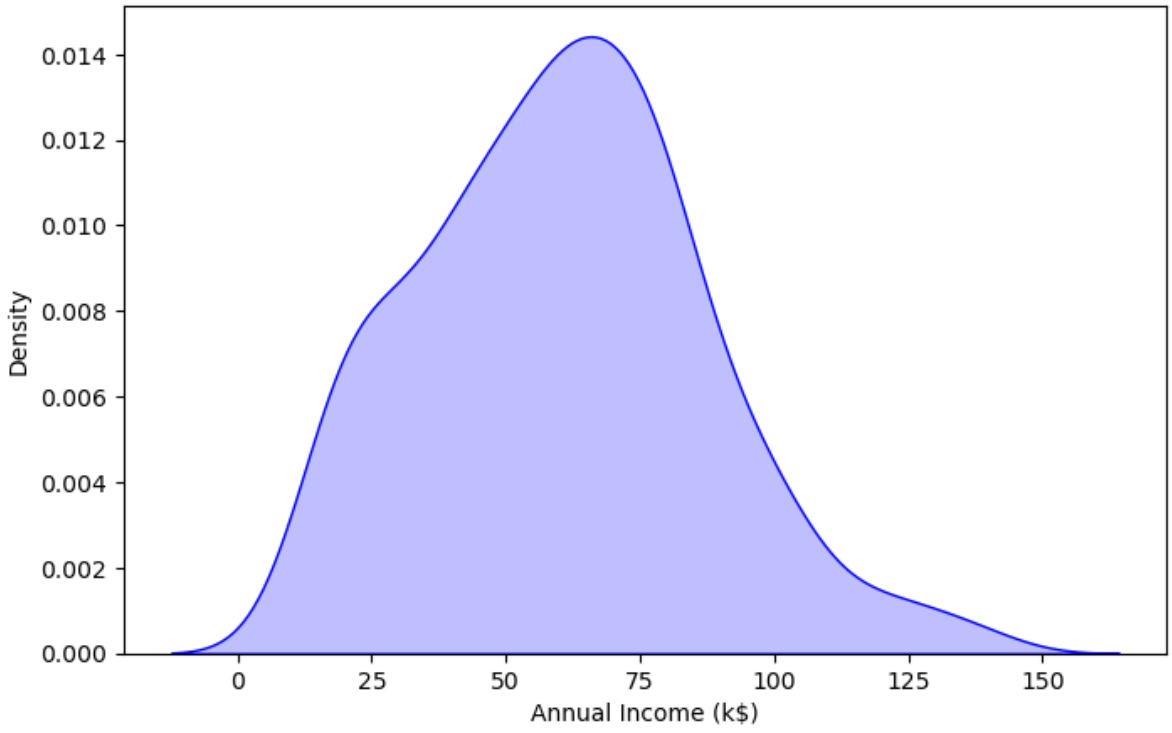
Out[ ]:
```
count    200.000000
mean      60.560000
std       26.264721
min       15.000000
25%       41.500000
50%       61.500000
75%       78.000000
max      137.000000
Name: Annual Income (k$), dtype: float64
```

- Veri setindeki müşterilerin yıllık gelirleri, 15.000ile137.000 arasında değişiyor. Minimum değer 15.000$vemaksimumdeğer$137.000 olarak belirlenmiş.
- Ortalama yıllık gelir, 60.560$ olarak hesaplanmış. Bu, veri setindeki müşterilerin genel yıllık gelir düzeyini temsil ediyor.
- Yıllık gelirlerin standart sapması, 26.264721 olarak hesaplanmış. Bu, yıllık gelirlerin dağılımının ne kadar değişken olduğunu gösteriyor.
- Çeyreklikler (25%, 50%, 75%) değerleri, yıllık gelirlerin dağılımını daha ayrıntılı olarak belirtir. Medyan (50%) değeri 61.500 $olarakhesaplanmış, yaniverisetindekimüşterilerinyarısının$61.500'dan daha düşük, yarısının ise daha yüksek yıllık gelire sahip olduğunu gösterir.
- 25% çeyreklik değeri 41.500, 75 olarak hesaplanmış. Bu, yıllık gelirin alt ve üst çeyreklerini belirtir ve genel bir dağılımın nasıl olduğunu gösterir.

In [ ]:
```
plt.figure(figsize=(8,5)) # Yeni bir grafik figürü oluşturuluyor ve boyutu belirle
df['Annual Income (k$)'].plot.hist(bins=14,color="#660033") # 'Annual Income (k$)'
plt.show()  # Oluşturulan grafiği göstermek için kullanılıyor
```



In [ ]:
```
plt.figure(figsize=(8,5)) # Yeni bir grafik figürü oluşturuluyor ve boyutu belirler
sns.kdeplot(df['Annual Income (k$)'], color="blue", shade=True)
# Seaborn kütüphanesinin kdeplot() fonksiyonuyla 'Annual Income (k$)' sütununa göre
plt.show()  # Oluşturulan grafiği göstermek için kullanılıyor
```



- Yukarıdaki grafiklerden müşterilerin minimum yıllık gelirinin 15, maksimum gelirinin ise 137 olduğunu açıkça görebiliriz. Histogram dağılımımızda ortalama geliri 70 olan kişiler en yüksek frekans sayısına sahiptir. Tüm müşterilerin ortalama geliri 60,56'dır. Yukarıda gösterdiğimiz Kernel Density Plot'ta yıllık gelirin normal bir dağılıma sahip olduğunu görüyoruz.

- **Harcama Puanı :**
  - Aynı şekilde harcama puanı özelliğini de inceleyeceğiz.

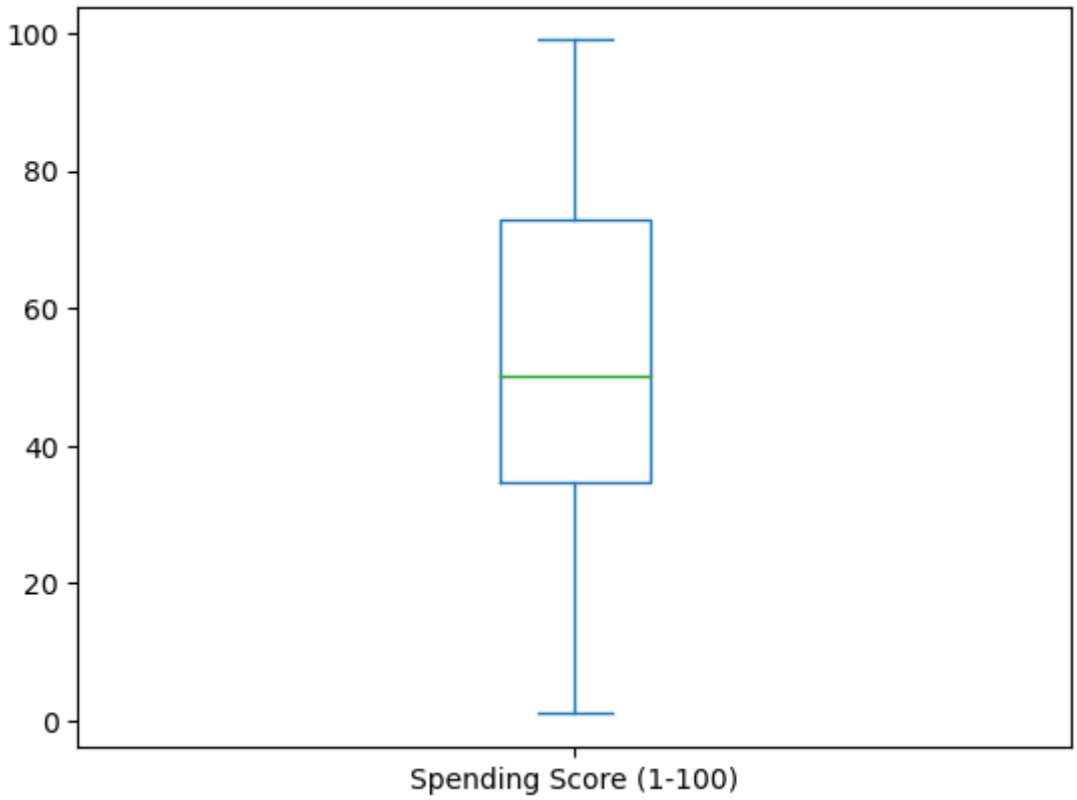
In [ ]:
```
df['Spending Score (1-100)'].describe() # 'Spending Score (1-100)' sütununun betims
```

Out[ ]:
```
count    200.000000
mean      50.200000
std       25.823522
min        1.000000
25%       34.750000
50%       50.000000
75%       73.000000
max       99.000000
Name: Spending Score (1-100), dtype: float64
```

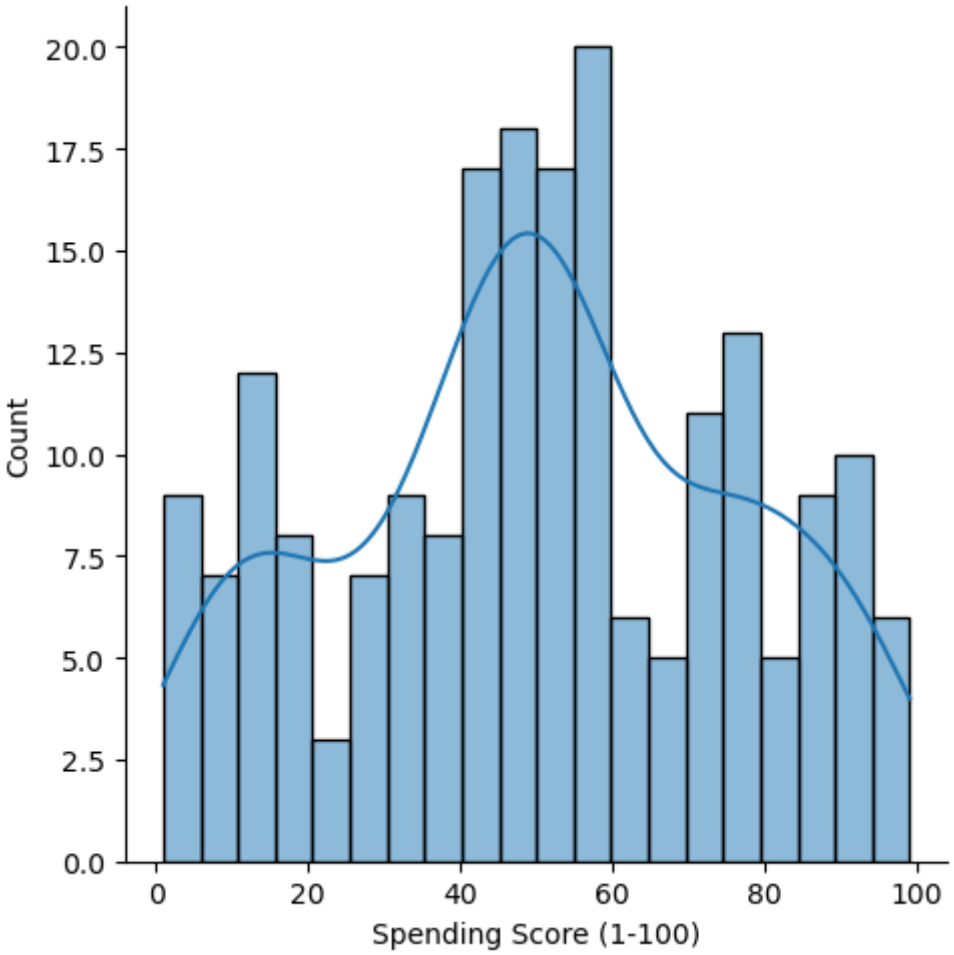
- Bu istatistiksel özet bilgileri, bir müşteri veri setindeki müşterilerin harcama skorlarına ilişkin bilgileri içeriyor.
- Veri setindeki müşterilerin harcama skorları, 1 ile 99 arasında değişiyor. Minimum değer 1, maksimum değer ise 99 olarak belirlenmiş.
- Ortalama harcama skoru, 50.200000 olarak hesaplanmış. Bu, müşterilerin genel harcama düzeyini temsil ediyor.

- Harcama skorlarının standart sapması, 25.823522 olarak hesaplanmış. Bu, harcama skorlarının ne kadar değişken olduğunu gösteriyor.
- Çeyreklikler (25%, 50%, 75%) değerleri, harcama skorlarının dağılımını daha ayrıntılı olarak belirtir. Medyan (50%) değeri 50.000 olarak hesaplanmış, yani veri setindeki müşterilerin yarısının 50'den daha düşük, yarısının ise daha yüksek harcama skoruna sahip olduğunu gösterir.
- 25% çeyreklik değeri 34.750, 75% çeyreklik değeri ise 73.000 olarak hesaplanmış. Bu, harcama skorunun alt ve üst çeyreklerini belirtir ve genel bir dağılımın nasıl olduğunu gösterir.

In [ ]:
```
df['Spending Score (1-100)'].plot.box()
plt.show()
```



In [ ]:
```
sns.displot(df['Spending Score (1-100)'], kde = True, bins = 20)
plt.show()
```



-Minimum harcama puanı 1, maksimum 99 ve ortalama 50,20'dir. Dağılım grafiğinden, harcama puanı 40 ile 50 arasında olan müşteri sınıfının tüm sınıflar arasında en yüksek frekansa sahip olduğu sonucuna varabiliriz.

2. K-means Sınıflandırması

-Veri setimizin her bir özelliğini keşfettikten sonra, şimdi müşterileri özelliklerine göre segmentlere ayırma zamanı. Bunu yapmak için k-means kümelemesini kullanacağız. Bu algoritma, merkezler olarak bilinen kümelerimiz için ilk merkezler olarak hizmet edecek veri kümesinden rastgele k nesne seçerek başlar. Daha sonra her adımda algoritma, her bir küme içindeki bireyler arasındaki mesafe olan iç mesafeyi en aza indirmeye ve kümeler arasındaki mesafe olan ara mesafeyi en üst düzeye çıkarmaya çalışır.

- **Optimal Küme Sayısının Belirlenmesi:**
  - Kümelerle çalışırken, kullanılacak küme sayısını belirtmemiz gerekir. Bu nedenle, en uygun küme sayısını bulmamız gerekiyor, bunun için Siluet yöntemini kullanacağız.

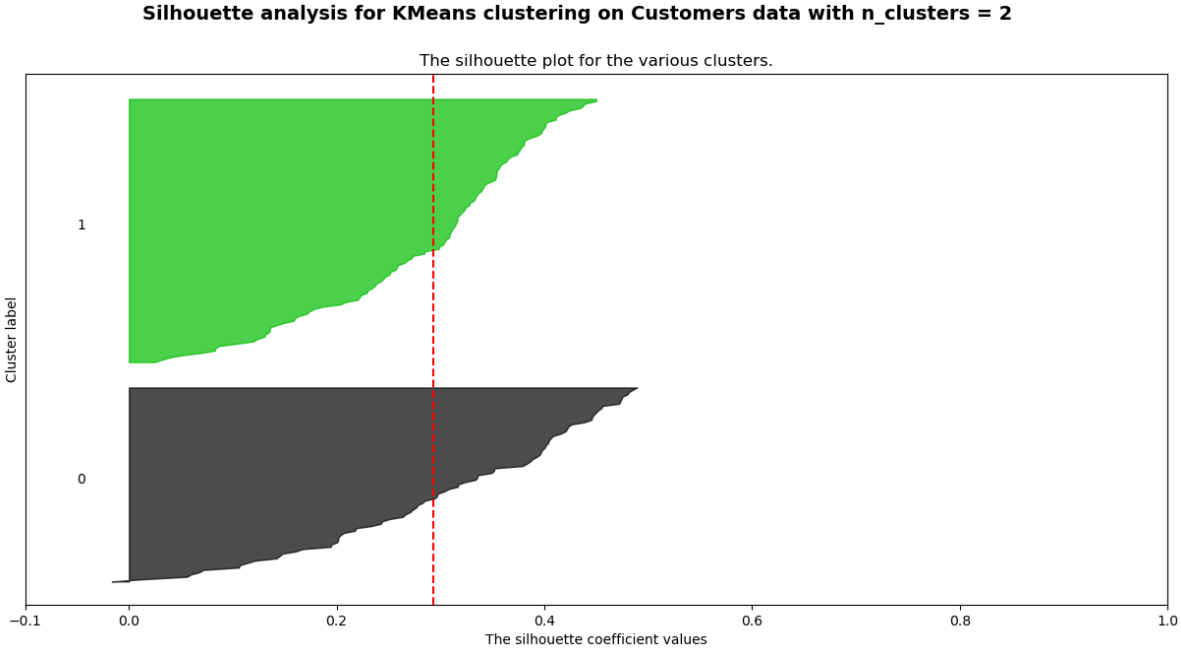
- **Ortalama Siluet Yöntemi:**
  -Ortalama siluet yöntemi yardımıyla kümeleme işlemimizin kalitesini ölçebiliriz. Bununla, veri nesnesinin küme içinde ne kadar iyi olduğunu belirleyebiliriz. Yüksek bir ortalama siluet genişliği elde edersek, iyi bir kümelemeye sahip olduğumuz anlamına gelir. Ortalama siluet yöntemi, farklı k değerleri için siluet gözlemlerinin ortalamasını hesaplar. Optimum k küme sayısıyla, k küme için önemli değerler üzerinden ortalama siluet maksimize edilebilir.

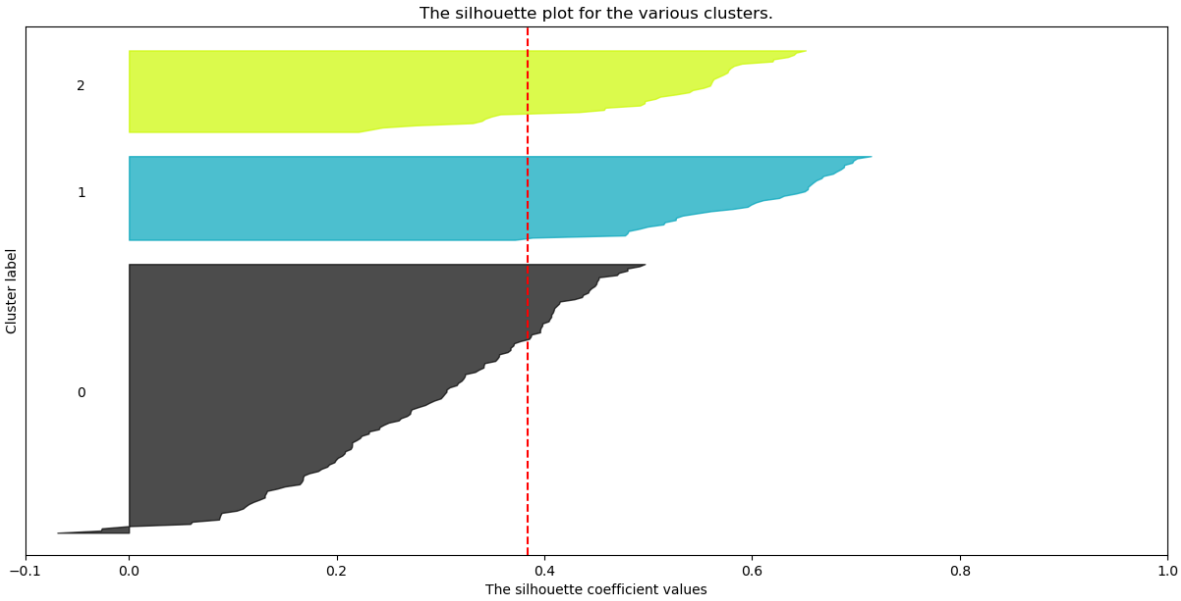
In [ ]:
```
df['Gender'] = df.Gender.map({'Male':1,'Female':0}) # cinse göre gruplandırıyoruz
```

In [ ]:
```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_samples, silhouette_score
import matplotlib.cm as cm

l=[]

# Veri setinizden sadece özellikleri seçerek X değişkenine atayın.
X = df.iloc[:,1:]

# 2'den 10'a kadar olan küme sayıları için döngü başlatın.
for n_clusters in range(2,11):
    # Subplot oluşturun ve boyutunu belirleyin.
    fig, ax1 = plt.subplots(1, 1)
    fig.set_size_inches(15, 7)

    ax1.set_xlim([-.1, 1])
    # Her bir küme arasında boşluk bırakmak için (n_clusters+1)*10 değerini kullan
    ax1.set_ylim([0, len(X) + (n_clusters + 1) * 10])

    # KMeans algoritmasını n_clusters değeri ve random_state=10 olarak başlatın.
    clusterer = KMeans(n_clusters=n_clusters, n_init=10, random_state=10)
    cluster_labels = clusterer.fit_predict(X)

    # Silhouette skoru, oluşan kümeleme için yoğunluk ve ayrımı gösterir.
    silhouette_avg = silhouette_score(X, cluster_labels)
    print("For n_clusters =", n_clusters,
          "The average silhouette_score is :", silhouette_avg)
    l.append(silhouette_avg)

    # Her bir örnek için silhouette skorlarını hesaplayın.
    sample_silhouette_values = silhouette_samples(X, cluster_labels)

    y_lower = 10
    for i in range(n_clusters):
        # i. küme için silhouette skorlarını toplayın ve sıralayın.
        ith_cluster_silhouette_values = \
            sample_silhouette_values[cluster_labels == i]

        ith_cluster_silhouette_values.sort()

        size_cluster_i = ith_cluster_silhouette_values.shape[0]
        y_upper = y_lower + size_cluster_i

        # Her bir silhouette grafiğini renklendirin.
        color = cm.nipy_spectral(float(i) / n_clusters)
        ax1.fill_betweenx(np.arange(y_lower, y_upper),
                          0, ith_cluster_silhouette_values,
                          facecolor=color, edgecolor=color, alpha=0.7)

        # Silhouette grafiğinin üzerine küme numaralarını ekleyin.
        ax1.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))

        # Bir sonraki plot için y_lower değerini güncelleyin.
        y_lower = y_upper + 10  # 10, 0 örneği için

    ax1.set_title("The silhouette plot for the various clusters.")
    ax1.set_xlabel("The silhouette coefficient values")
    ax1.set_ylabel("Cluster label")

    # Tüm değerlerin ortalama silhouette skoru için dikey çizgi çizin.
    ax1.axvline(x=silhouette_avg, color="red", linestyle="--")
```

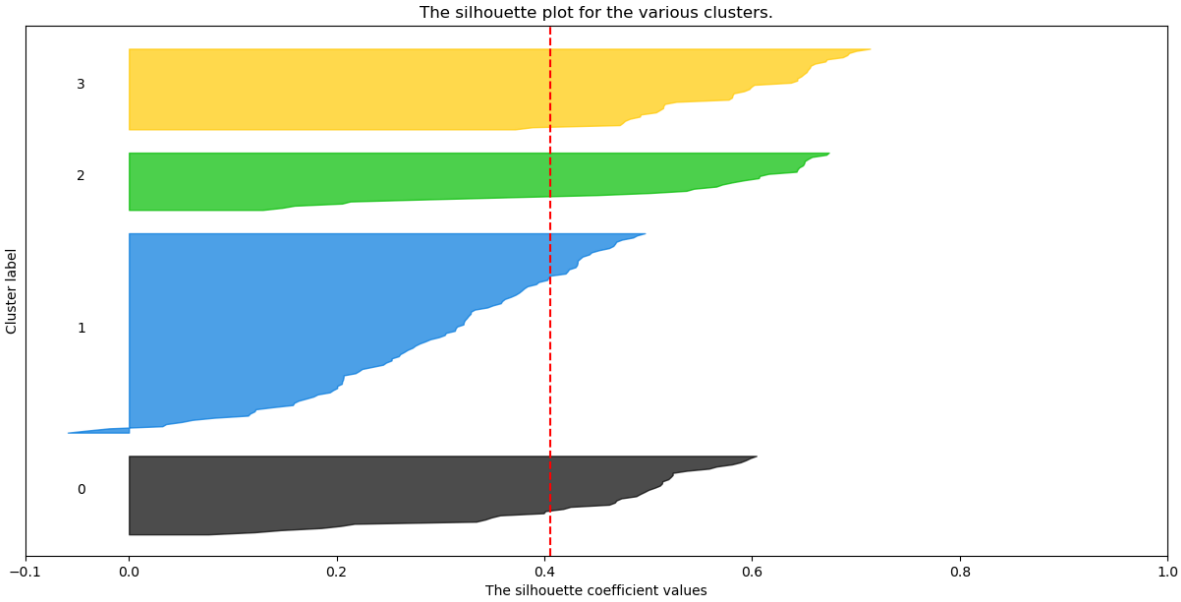
```
    ax1.set_yticks([])  # Y ekseni etiketlerini temizleyin
    ax1.set_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])

    plt.suptitle(("Silhouette analysis for KMeans clustering on Customers data "
                  "with n_clusters = %d" % n_clusters),
                 fontsize=14, fontweight='bold')
```

```
For n_clusters = 2 The average silhouette_score is : 0.29307334005502633
For n_clusters = 3 The average silhouette_score is : 0.383798873822341
For n_clusters = 4 The average silhouette_score is : 0.4052954330641215
For n_clusters = 5 The average silhouette_score is : 0.4440669204743008
For n_clusters = 6 The average silhouette_score is : 0.45205475380756527
For n_clusters = 7 The average silhouette_score is : 0.43949619264530887
For n_clusters = 8 The average silhouette_score is : 0.4349105351263195
For n_clusters = 9 The average silhouette_score is : 0.4080555594955236
For n_clusters = 10 The average silhouette_score is : 0.3828606213726962
```

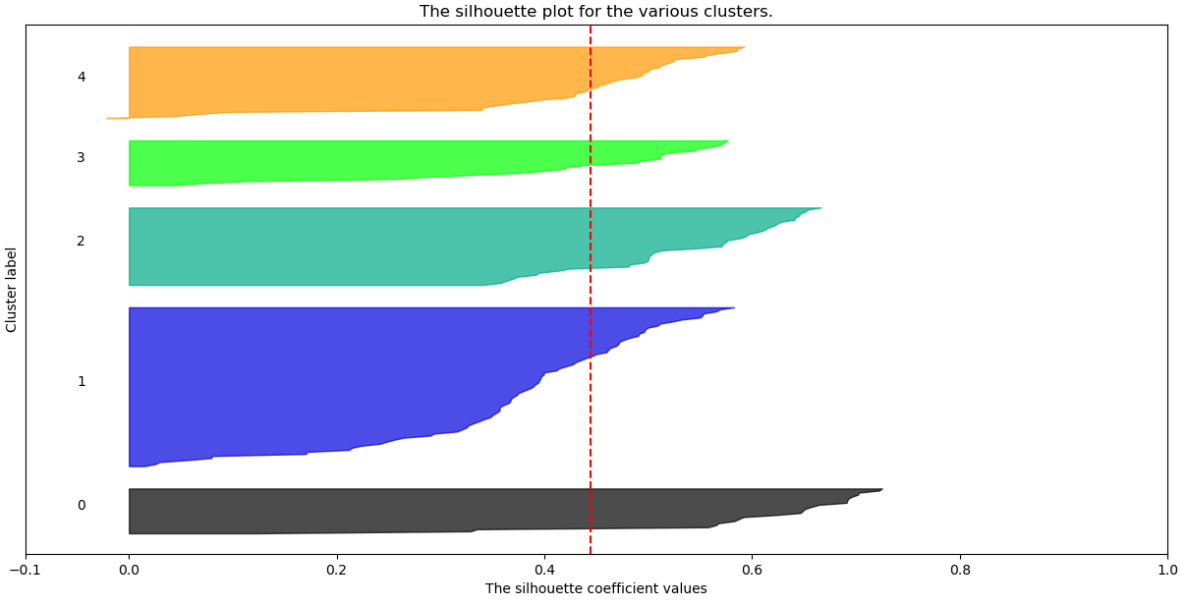




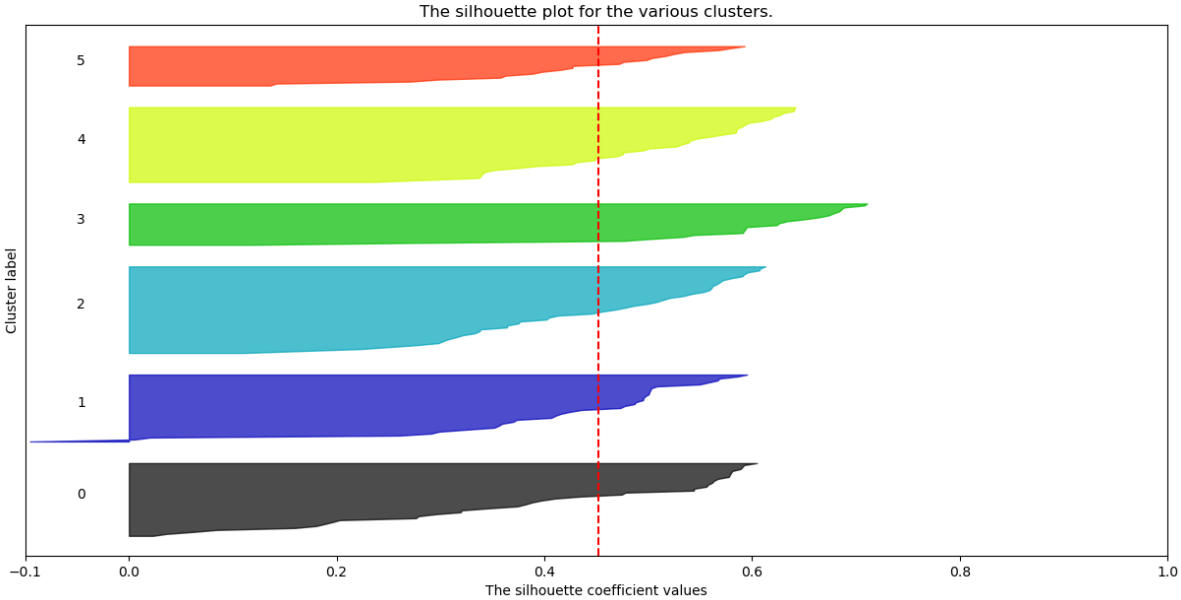
Silhouette analysis for KMeans clustering on Customers data with n_clusters = 4



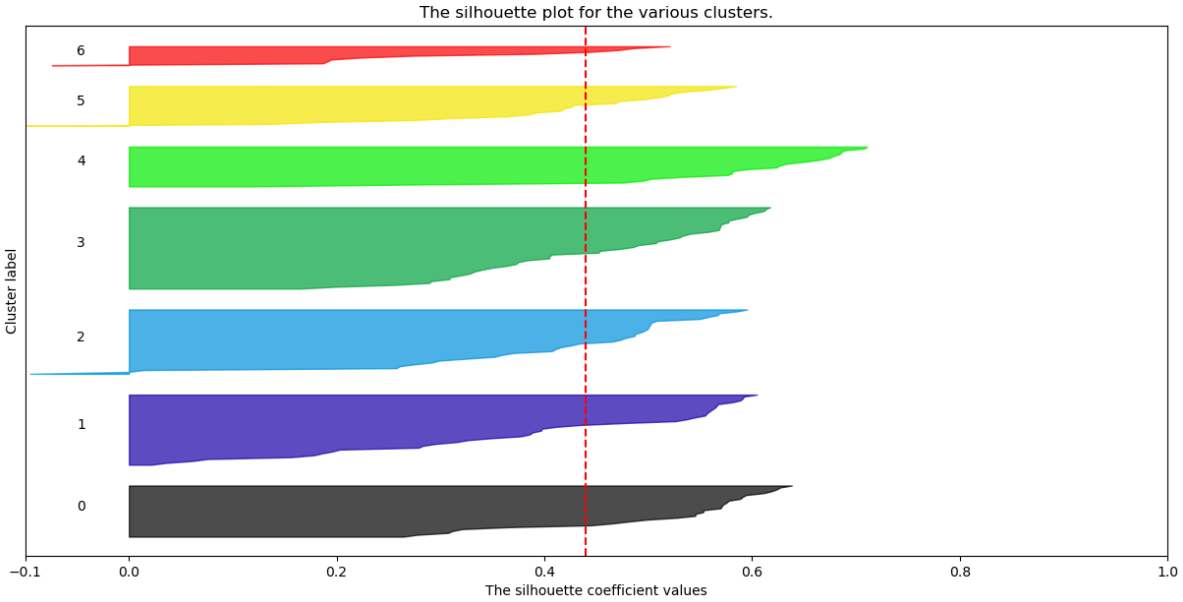
Silhouette analysis for KMeans clustering on Customers data with n_clusters = 5



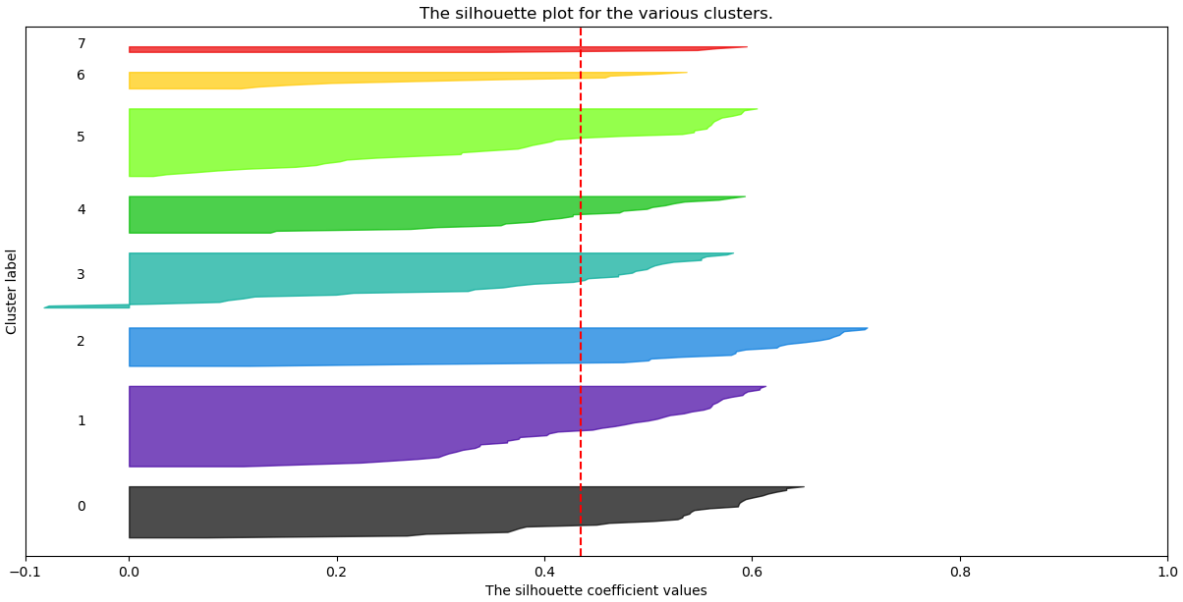
Silhouette analysis for KMeans clustering on Customers data with n_clusters = 6



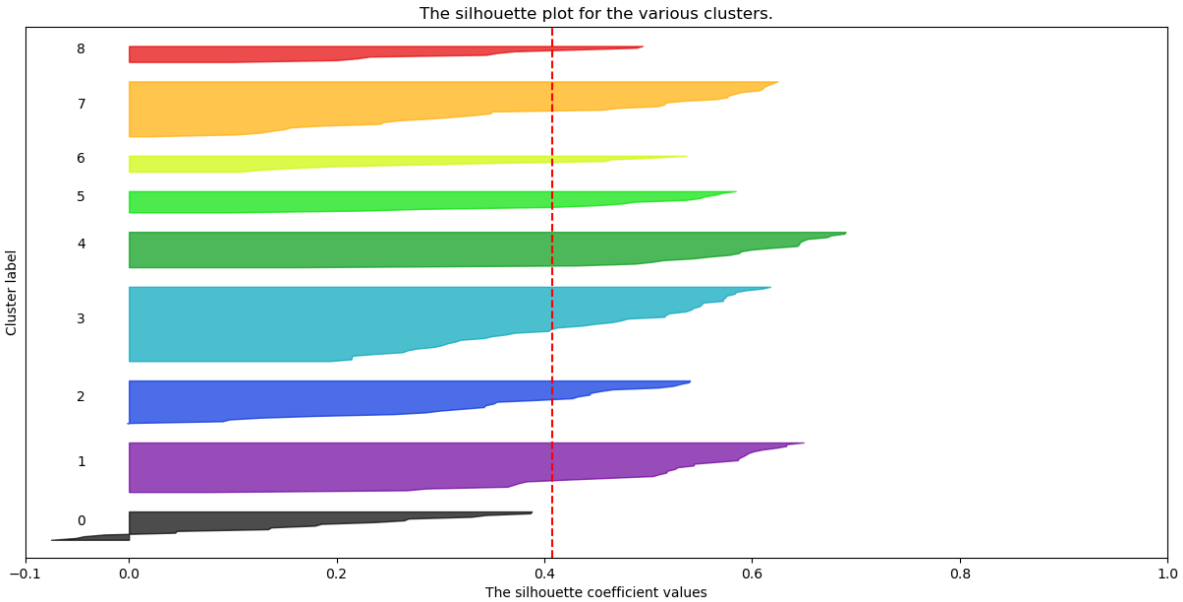
Silhouette analysis for KMeans clustering on Customers data with n_clusters = 7

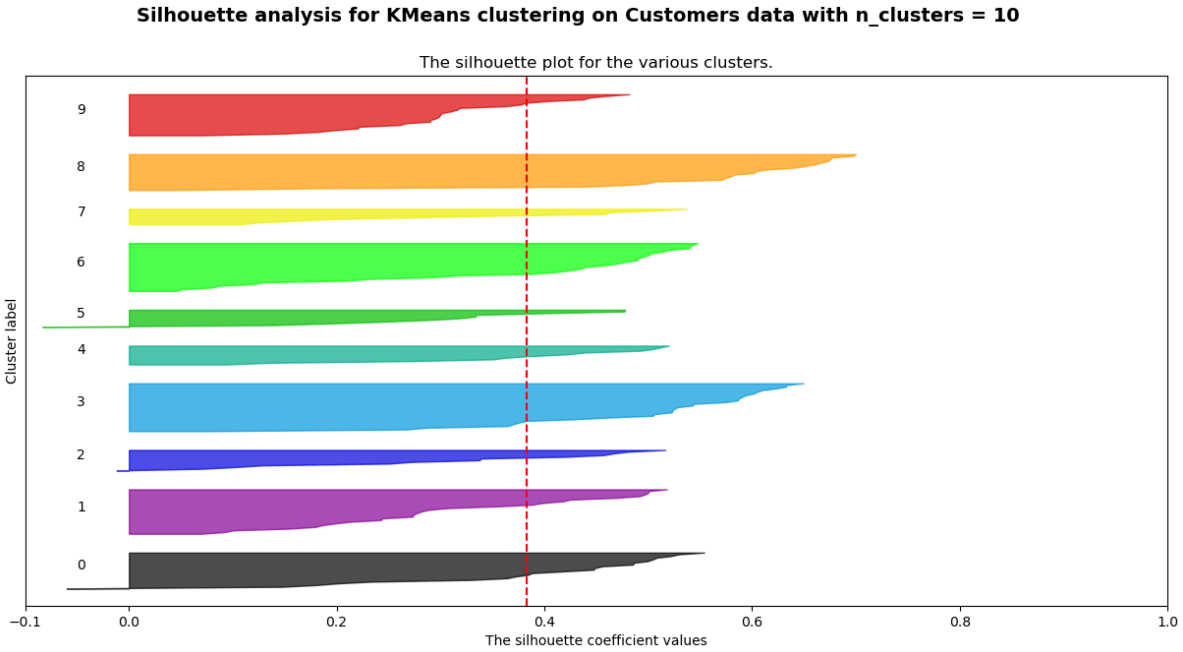


Silhouette analysis for KMeans clustering on Customers data with n_clusters = 8



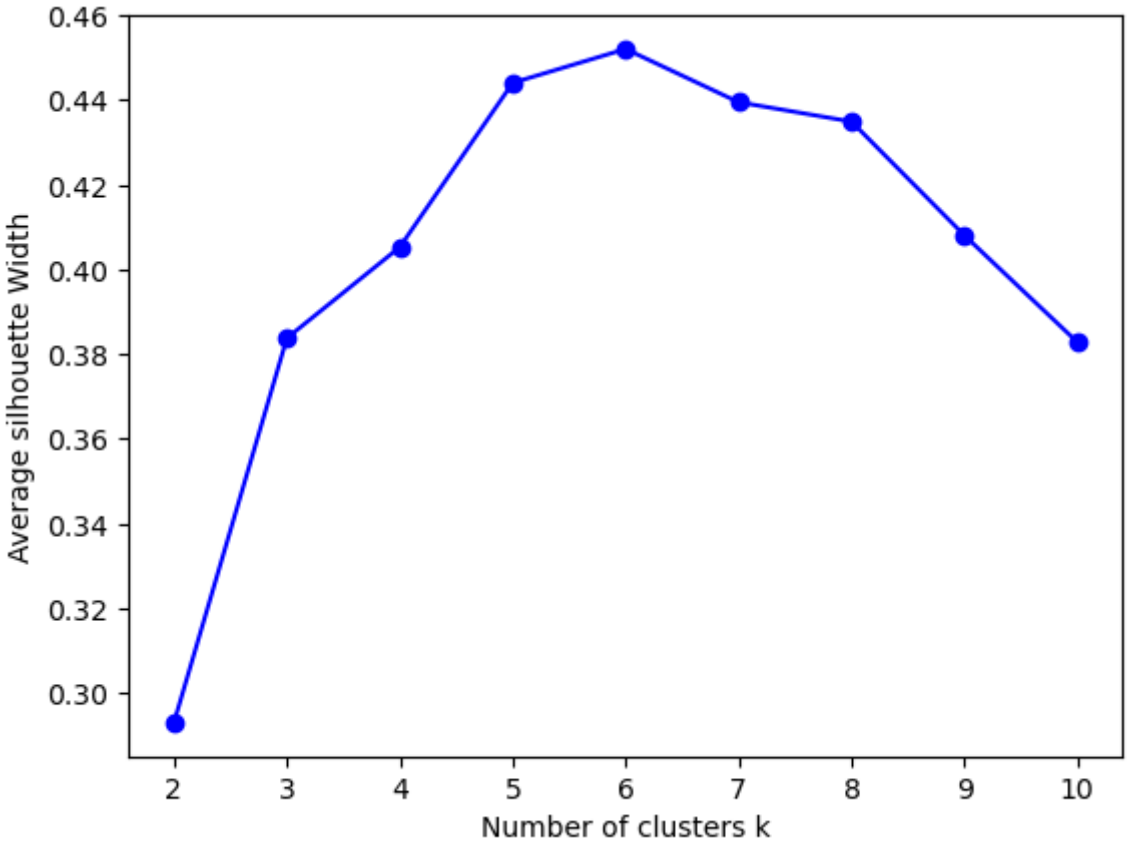
Silhouette analysis for KMeans clustering on Customers data with n_clusters = 9





In [ ]:

```
clusters = list(range(2, 11))
# clusters listesi, x ekseni için kullanılacak küme sayılarını içerir.

plt.plot(clusters, l, '-bo')
# Küme sayılarına karşı ortalama silhouette skorlarını grafiğe çizdirin.
# '-bo' ifadesi ile mavi renkte noktalı bir çizgi çizilir.

plt.xlabel("Number of clusters k")
# x ekseni etiketi, küme sayılarını temsil eder.

plt.ylabel("Average silhouette Width")
# y ekseni etiketi, ortalama silhouette skorlarını temsil eder.

plt.show()
# Grafiği göster.
```



- Yukarıdaki şekilden gördüğümüz gibi, müşteri segmentasyon görevi için optimal küme sayısı k, ortalama silhouette genişliği 0.45 olan 6'dır.

3. Visualizing the Clustering Results using the PCA with 2 components:

In [ ]:
```
from sklearn.decomposition import PCA
reduced = PCA(n_components=2).fit_transform(X)

kmeans = KMeans(init='k-means++', n_clusters=6, random_state=10)
kmeans.fit(reduced)
```

Out[ ]:
```
KMeans(n_clusters=6, random_state=10)
```

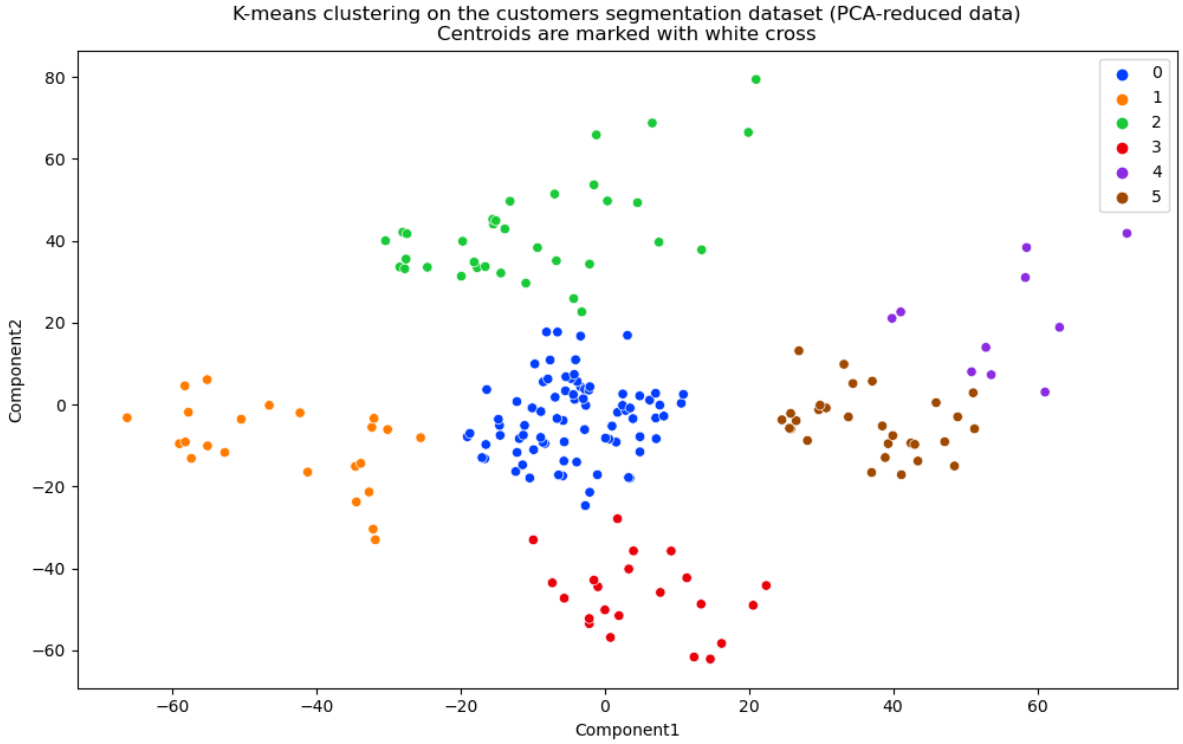
In [ ]:
```
pca_df = pd.DataFrame(reduced,columns=['Component1','Component2']) # PCA sonucu ol
pca_df['Segment'] = kmeans.labels_ # Oluşan kümeleri pca_df veri setine ekliyoruz
pca_df.head(10) # pca_df veri setinin ilk 10 gözlemini görüntülüyoruz
```

Out[ ]:

	Component1	Component2	Segment
0	-31.869945	-33.001252	1
1	0.764494	-56.842901	3
2	-57.408276	-13.124961	1
3	-2.168543	-53.478590	3
4	-32.174085	-30.388412	1
5	-2.176952	-52.227269	3
6	-59.065690	-9.543763	1
7	12.370862	-61.618021	3
8	-66.315769	-3.214232	1
9	-5.655562	-47.267222	3

In [ ]:

```
plt.figure(figsize=(12,7)) # Grafiğin boyutunu belirliyoruz
sns.scatterplot(x='Component1',y='Component2',data=pca_df,hue='Segment',palette='b
# PCA sonucu oluşan 2 bileşenli veri setini görselleştiriyoruz
plt.title('K-means clustering on the customers segmentation dataset (PCA-reduced da
          'Centroids are marked with white cross') # Grafiğin başlığını belirliyor
plt.legend() # Grafiğin sağ üst köşesindeki küme etiketlerini gösteriyoruz
plt.show() # Grafiği gösteriyoruz
```



- **Küme 0 ve 5** - Bu iki küme, yüksek yaş ve yüksek gelire sahip müşterilerden oluşur. Bu müşteriler, lüks ürün ve hizmetlere ilgi duyabilirler.
- **Küme 1** - Bu küme, düşük yaş ve düşük gelire sahip müşterileri temsil eder. Bu müşteriler, uygun fiyatlı ve kaliteli ürün ve hizmetleri tercih edebilirler.
- **Küme 2** - Bu küme, orta yaş ve orta gelire sahip müşterilerden oluşur. Bu müşteriler, hem fiyat hem de kalite açısından dengeli ürün ve hizmetleri arayabilirler.

- **Küme 3** - Bu küme, yüksek yaş ve orta gelire sahip müşterilerden oluşur. Bu müşteriler, güvenilir ve dayanıklı ürün ve hizmetleri önemseyebilirler.
- **Küme 4** - Bu kümede, düşük yaş ve orta gelire sahip müşteriler bulunur. Bu müşteriler, trend ve yenilikçi ürün ve hizmetlere ilgi gösterebilirler.

- Kümeleme yardımıyla, müşterilerin farklı özelliklerini daha iyi anlayarak onlara uygun pazarlama stratejileri geliştirebiliriz. Müşterilerin segmentasyonuyla birlikte, şirketler yaş, gelir, harcama alışkanlıkları vb. gibi birçok parametreye dayanan müşterilere hedeflenen ürün ve hizmetleri sunabilirler. Ayrıca, müşteri memnuniyeti, sadakat, geri bildirim gibi daha karmaşık desenler de daha iyi segmentasyon için dikkate alınabilir.

In [ ]: