İSTANBUL
AYVANSARAY ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

VERİ MADENCİLİĞİ DERSİ

‘‘KREDİ KARTI VERİLERİ İÇEREN VERİ SETİNDE KÜMELEME ÇALIŞMASI’’

Doç. Dr. Metin ZONTUL

Hazırlayan:

Samet ÖÇSOY
18040101008

22.01.2022

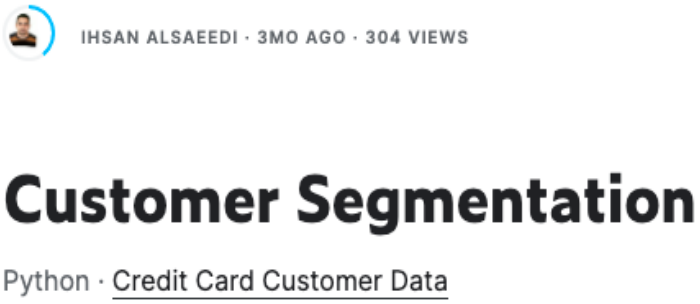
İçindekiler

1)Giriş

Doküman içeriğinde, makine öğrenmesinin çalışma alanlarından kümeleme analizi konusuna ilişkin bir uygulama bulunmaktadır.

Bu projedeki amaç; içerisinde kredi kartı verileri bulunan bir veri seti üzerinde çalışma yaparak, bu çalışma sonucunda bir model oluşturmak ve oluşturulan modeli kullanarak **birbirine yakın verileri farklı kümelere ayırmaktır.**

Projede kullanılan veri seti:



2)Veri Seti İçerisindeki Öznitelikler (Features) :

musteri_seri_no: Müşteri Seri Kimlik Numarası

kredikart_parola: Müşteri Kredi Kartı Şifresi

limit: Kredi Kartı Limit Değeri

kredikart_sayisi: Müşteri Sahip Olduğu Kredi Kart Sayısı

toplam_ziyaret: Müşteri Tarafından Yapılan Toplam Banka Ziyareti Sayısı

toplam_cevrimici_ziyaret: Banka Müşterisinin Toplam Çevrimiçi Ziyareti

toplam_arama: Müşteri Tarafından Bankaya Yapılan Toplam Çağrı

3)Uygulama Kodları

3.1)Kütüphaneler

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from yellowbrick.cluster import KElbowVisualizer
from sklearn.cluster import AgglomerativeClustering
```

Açıklama:

Numpy: Matris işlemleri için kullanılmaktadır.

Pandas: Veri ön işleme aşaması ve veri manipülasyonu için kullanılmaktadır.

Matplotlib: Görselleştirme için kullanılmaktadır.

Scikit-learn: Makine öğrenmesi aşaması için kullanılmaktadır.

3.2) Veri Setini Okuma

```
# veri seti okuma islemi
dataset = pd.read_csv('credit_card.csv')
print(dataset.to_string())
```

 # Çıktı:

Sl_No	Customer Key	Avg_Credit_Limit	Total_Credit_Cards	Total_visits_bank	Total_visits_online	Total_calls_made
1	87073.0	100000.0	2.0	1.0	1.0	0.0
2	38414.0	50000.0	3.0	0.0	10.0	9.0
3	17341.0	50000.0	7.0	1.0	3.0	4.0
4	40496.0	30000.0	5.0	1.0	1.0	4.0
5	47437.0	100000.0	6.0	0.0	12.0	3.0
6	58634.0	20000.0	3.0	0.0	1.0	8.0
7	48370.0	100000.0	5.0	0.0	11.0	2.0
8	37376.0	15000.0	3.0	0.0	1.0	1.0
9	82490.0	5000.0	2.0	0.0	2.0	2.0
10	44770.0	3000.0	4.0	0.0	1.0	7.0
11	52741.0	10000.0	4.0	0.0	5.0	5.0

Açıklama:

Veri setine ilk olarak bir göz atıldığında verilerin **etiketsiz** bir veri olduğunu gözlemleyebiliyoruz.

Makine öğrenmesi;
1 - Etiketli veriler için **Gözetimli Öğrenme,**
2 - Etiketsiz veriler için **Gözetimsiz Öğrenme**
olmak üzere ikiye ayrılmaktadır.

Not: Bunun dışında **Pekiştirmeli Öğrenme** tipi de mevcuttur. Bu problem etiketsiz bir verilerden oluştuğu için **Gözetimsiz Öğrenme** kapsamına girmektedir.

3.3)Veri Ön İşleme Aşaması

a) Kolonların İsimlerini Değiştirmek

```
# veri setindeki kolonlarin isimlerini degistirme islemi
print(dataset.columns)
new_columns = ["musteri_seri_no","kredikart_parola","limit",
"kredikart_sayisi,",

"toplam_ziyaret","toplam_cevrimici_ziyaret","toplam_arama"]
for i,c in zip(dataset.columns, new_columns):
 dataset.rename(columns = {i:c}, inplace = True)

print(dataset.columns)
print(dataset.to_string())
```

 # Çıktı:

```
# değişim öncesi kolon isimleri
Index(['Sl_No', 'Customer Key', 'Avg_Credit_Limit',
'Total_Credit_Cards',
       'Total_visits_bank', 'Total_visits_online', 'Total_calls_made'],
      dtype='object')


# değişim sonrası kolon isimleri
Index(['musteri_seri_no', 'kredikart_parola', 'limit',
'kredikart_sayisi,',
       'toplam_ziyaret', 'toplam_cevrimici_ziyaret', 'toplam_arama'],
      dtype='object')
```

b) Keşifsel Veri Analizi

b.1) Veri Seti Boyut Öğrenmek

```
# veri seti boyutunu ogrenmek
print("\n\nVeri seti boyut : ",dataset.shape)
print("\n\n")
```

 # Çıktı:

```
Veri seti boyut :  (685, 7)
```

Açıklama:

Pandas'ın shape fonksiyonu ile veri setine ilişkin boyut bilgilerini alıyoruz. Veri setimizde 685 satır ve 7 sütun var olduğunu öğreniyoruz.

---

b.2) Veri Seti Hakkında Bilgi Almak

```
# veri setine dair bilgi almak
print(dataset.info())
print("\n\n")
```

 # Çıktı:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 685 entries, 0 to 684
Data columns (total 7 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   musteri_seri_no           685 non-null    int64
 1   kredikart_parola          682 non-null    float64
 2   limit                     681 non-null    float64
 3   kredikart_sayisi,         683 non-null    float64
 4   toplam_ziyaret            684 non-null    float64
 5   toplam_cevrimici_ziyaret  680 non-null    float64
 6   toplam_arama              684 non-null    float64
dtypes: float64(6), int64(1)
memory usage: 37.6 KB
None
```

Açıklama: Veri setinde her verinin sayısal bir tipte olduğunu gözlemliyoruz. Hiçbir şekilde kategorik veri veri setimizde bulunmamaktadır.

```
# veri setine dair bilgi almak
print(dataset.describe().to_string())
```

 # Çıktı:

	musteri_seri_no	kredikart_parola	limit	kredikart_sayisi,	toplam_ziyaret	toplam_cevrimici_ziyaret	toplam_arama
count	685.00	682.00	681.00	683.00	684.00	680.00	684.00
mean	330.43	55084.54	33904.55	4.68	2.37	2.62	3.62
std	194.28	25673.29	37237.48	2.17	1.63	2.909	2.89
min	1.00	11265.00	3000.00	1.00	0.00	0.00	0.00
25%	163.00	33540.25	10000.00	3.00	1.00	1.00	1.00
50%	331.00	53874.50	18000.00	4.00	2.000	2.00	3.00
75%	498.00	77321.50	47000.00	6.00	4.00	4.00	6.000
max	669.00	99843.00	200000.00	10.00	5.00	15.00	10.00

Açıklama:

Kolonlarda verilerin minimum ve maksimum değerleri arasında bir sorun görünmüyor. Bu nedenle bu aşamada bir düzenlemeye ihtiyaç bulunmamaktadır.

Bu kısımda pandas'ın info() ve describe() fonksiyonları kullanılmıştır.

---

b.3) Veri Seti için Tekrarlayan Veri Kontrolü Yapmak

```
# veri setinde tekrarlayan verileri kontrol etmek
print("Veri setinden satirlarda tekrar eden veri olan satir sayisi :
",dataset.duplicated().sum())
print("\n\n")
```

 # Çıktı:

```
Veri setinden satirlarda tekrar eden veri olan satir sayisi :  16
```

Açıklama:

Bu aşamada veri setimizde 16 kolonun birbirini tekrarladığını farkediyoruz. **Burada tekrarlayan verilerin veri setinden silinmesi gerekmektedir.**

b.4) Veri Seti için Eksik Veri Kontrolü Yapmak

```
# veri setinde eksik olan verileri kontrol etmek
print("Veri setinde satirlarda eksik veri olan kayit sayisi :\n")
```

```
print(dataset.isnull().sum())
```

 # Çıktı:

```
Veri setinde satirlarda eksik veri olan kayit sayisi :

musteri_seri_no             0
kredikart_parola            3
limit                       4
kredikart_sayisi,           2
toplam_ziyaret              1
toplam_cevrimici_ziyaret    5
toplam_arama                1
dtype: int64
```

Açıklama:
Veri setimizde dataset.isnull().sum() fonksiyonu ile sütunlarda kaç adet eksik veri olduğunu buluyoruz.

Görüldüğü üzere sütunlarımızın yaklaşı tamamında eksik veri bulunmaktadır. Burada eksik olan verileri veri setinden **silinmeli ya da doldurulmalıdır.** Eksik verilerin sayıları çok az olduğu için burada biz **silme işlemini** tercih edeceğiz.

c) Keşifsel Veri Analizi Sonrası Veri Manipülasyonu

c.1) Tekrarlayan Verileri Silmek

```
# tekrarlayan veriler siliniyor.

print("Veri setinden satirlarda tekrar eden veri olan satir sayisi :
",dataset.duplicated().sum())
new_dataset = dataset.drop_duplicates()

# silinme sonrasi kontrol
print("Veri setinden satirlarda tekrar eden veri olan satir sayisi :
",new_dataset.duplicated().sum())
```

 # Çıktı:

```
# İşlem öncesi
Veri setinden satirlarda tekrar eden veri olan satir sayisi :  16

# İşlem Sonrası
Veri setinden satirlarda tekrar eden veri olan satir sayisi :  0
```

Açıklama:

Bu kısımda veri setimizdeki tekrar eden veriler için silme işlemini uyguladık.

c.2) Eksik olan Verileri Silmek

```
# eksik olan veriler siliniyor.

print("Veri setinde satirlarda eksik veri olan kayit sayisi :\n")
print(new_dataset.isnull().sum())

real_dataset = new_dataset.dropna()

print("\n\n")
print("Veri setinde satirlarda eksik veri olan kayit sayisi :\n")
print(real_dataset.isnull().sum())
```

 # Çıktı:

 # Silme İşlemi Öncesi

```
Veri setinde satirlarda eksik veri olan kayit sayisi :
musteri_seri_no             0
kredikart_parola            3
limit                       4
kredikart_sayisi,           2
toplam_ziyaret              1
toplam_cevrimici_ziyaret    5
toplam_arama                1
dtype: int64
```

 # Silme İşlemi Sonrası

```
Veri setinde satirlarda eksik veri olan kayit sayisi :
musteri_seri_no             0
kredikart_parola            0
limit                       0
kredikart_sayisi,           0
toplam_ziyaret              0
toplam_cevrimici_ziyaret    0
toplam_arama                0
dtype: int64
```

Açıklama:

Bu kısımda veri setimizdeki eksik veri teşkil eden veriler için silme işlemini uyguladık.

c.3) Modele Uygun Olmayan Verileri Silmek

```
# modele uymayan gereksiz veriler veri setinden siliniyor.

real_dataset.drop(columns = ['musteri_seri_no', 'kredikart_parola'],
inplace = True)
print(real_dataset.describe().to_string())
```

 # Çıktı:

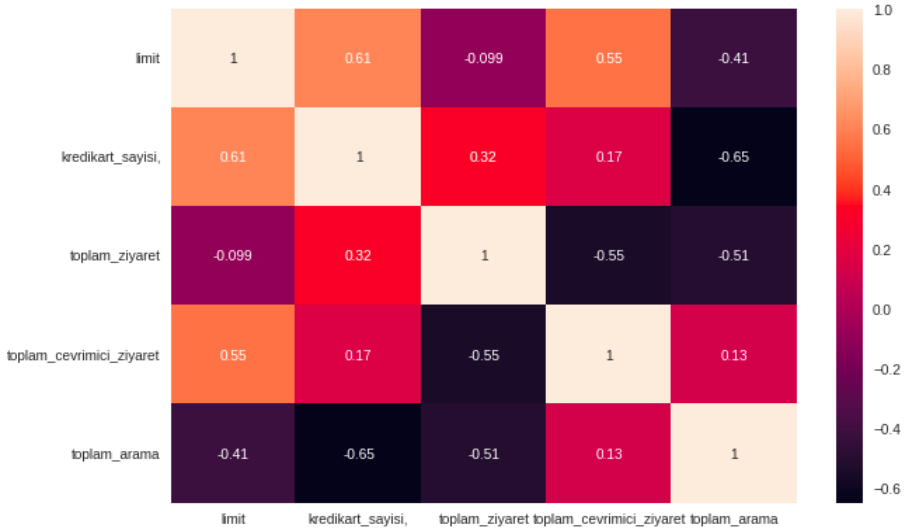
```
            limit     kredikart_sayisi,  toplam_ziyaret  toplam_cevrimici_ziyaret  toplam_arama
count      661.00          661.00            661.00              661.00              661.00
mean     34544.62            4.70              2.39                2.60                3.59
std      37604.68           2.16               1.63                2.93                2.86
min       3000.00            1.00              0.00                0.00                0.00
25%      10000.00            3.00              1.00                1.00                1.00
50%      18000.00            5.00              2.00                2.00                3.00
75%      48000.00            6.00              4.00                4.00                5.00
max     200000.00           10.00              5.00               15.00                10
```

---

d) Korelasyon Isı Haritası Oluşturmak

```
# korelasyon ısı haritası gösteriliyor
corrmat= real_dataset.corr()
plt.figure(figsize=(10,7))
sns.heatmap(corrmat,annot=True)
```

 # Çıktı:



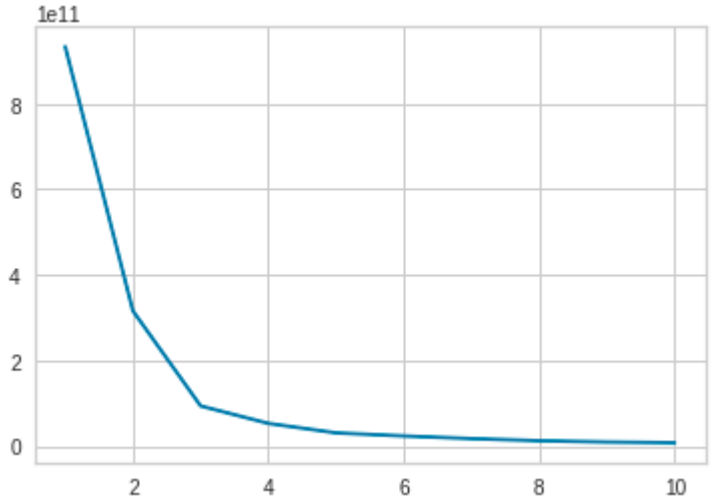
Açıklama:
Isı haritasında birbiri ile çok ilişkili kolonlar tespit edilmedi. Bu nedenle bu kısımda bir düzenlemeye ihtiyaç bulunmamaktadır.

3.4) Makine Öğrenmesi Aşaması

A)ELBOW YÖNTEMİ İLE K DEĞERİNİN HESAPLANMASI

```
# K-MEANS ALGORİTMASI İÇİN K DEĞERİNİ HESAPLIYORUZ

# Öncelikle k degerini buluyoruz.

from sklearn.cluster import KMeans

# elbow yontemi ile k degerini bulmaya calisiyoruz.
sonuclar = []
for i in range(1,11):
    kmeans = KMeans(n_clusters = i, init='k-means++', random_state= 123)
    kmeans.fit(real_dataset)
    sonuclar.append(kmeans.inertia_)  #inertia wcss değerleri

plt.plot(range(1,11),sonuclar)
plt.show()
```

 # Çıktı



Açıklama:

Elbow yöntemi ile ilk olarak K değerlerini hesapladık. Burada dirsek noktası x ekseninde 3 noktası olarak gözükmektedir. Bu nedenle k değerimiz 3 olarak alınmıştır.

B) K-MEANS ALGORİTMASI KULLANIMI

```
# GORSELLESTIRMEK ICIN VERI SETINI 2 KOLONA DUSURUYORUZ.

x1 = real_dataset.iloc[:, 1]
x2 = real_dataset.iloc[:, 3]

data = pd.concat([x1, x2], axis=1)

X = data.iloc[:, 0:2].values


##########

# ELBOW YONTEMI ILE BULDUGUMUZ K DEGERI ICIN ALGORITMAYI KULLANIYORUZ.

kmeans = KMeans(n_clusters = 3, init = 'k-means++', max_iter = 300,
n_init = 10, random_state = 0)

y_kmeans = kmeans.fit_predict(X)

plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 100, c =
'red', label = 'Küme 1')
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 100, c =
'blue', label = 'Küme 2')
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 100, c =
'green', label = 'Küme 3')

plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,
1], s = 300, c = 'orange', label = 'Küme Merkezleri')
plt.title('K-MEANS GRAFİK')
plt.xlabel('Kart Sayısı')
plt.ylabel('Çevrimiçi Ziyaret')
plt.legend()
plt.show()
```

C) HİYERARŞİK KÜMELEME ALGORİTMASI KULLANIMI

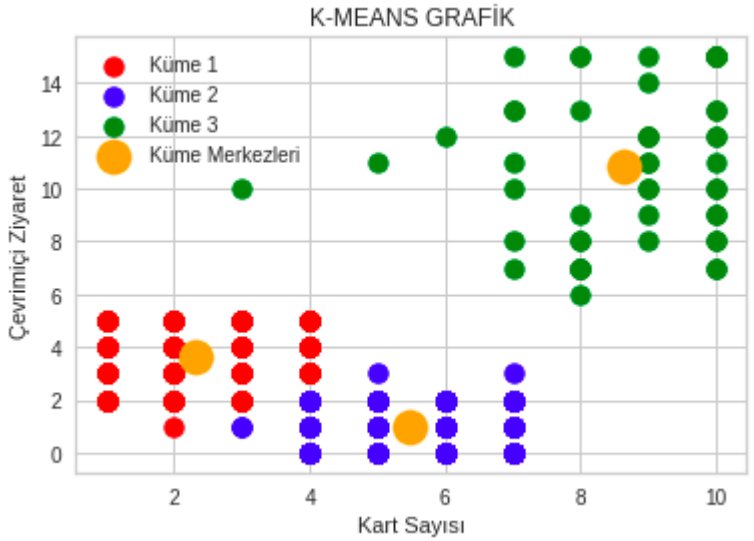
```
from sklearn.cluster import AgglomerativeClustering

ac = AgglomerativeClustering(n_clusters=3, affinity='euclidean',
linkage='ward')

y_predict = ac.fit_predict(X)

plt.scatter(X[y_predict==0,0],X[y_predict==0,1],s=100, c='red', label =
'Küme 1')
plt.scatter(X[y_predict==1,0],X[y_predict==1,1],s=100, c='blue', label
= 'Küme 2')
plt.scatter(X[y_predict==2,0],X[y_predict==2,1],s=100, c='green', label
= 'Küme 3')
plt.scatter(X[y_predict==3,0],X[y_predict==3,1],s=100, c='yellow')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,
1], s = 300, c = 'orange', label = 'Küme Merkezleri')
plt.title('HC GRAFİK')
plt.xlabel('Kart Sayısı')
plt.ylabel('Çevrimiçi Ziyaret')
plt.legend()

plt.show()
```

D) SOM ALGORİTMASI KULLANIMI

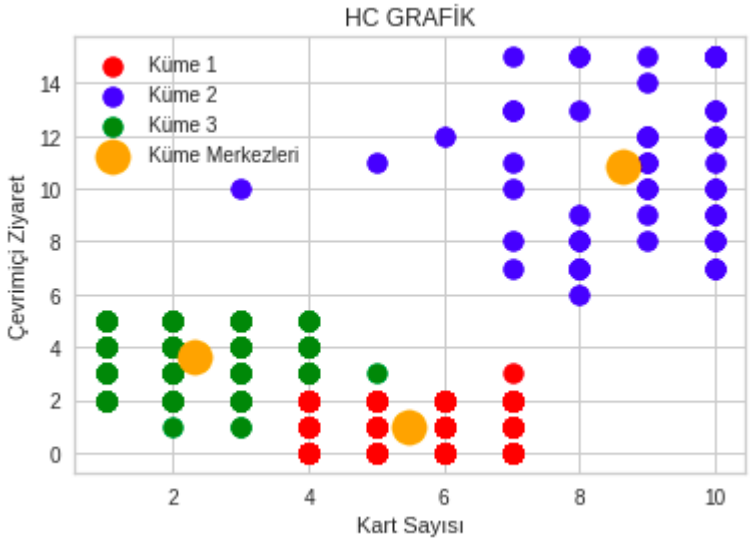
```
from matplotlib.colors import ListedColormap
from sklearn_som.som import SOM

som = SOM(m=3, n=1, dim=2, random_state=123)
som.fit(X)
predictions = som.predict(X)


x = X[:,0]
y = X[:,1]

colors = ['red', 'green', 'blue']
plt.scatter(x, y, c=predictions, cmap=ListedColormap(colors))
plt.title('SOM GRAFİK')
plt.xlabel('Kart Sayısı')
plt.ylabel('Çevrimiçi Ziyaret')
plt.legend()

plt.show()
```

4) SONUÇLAR

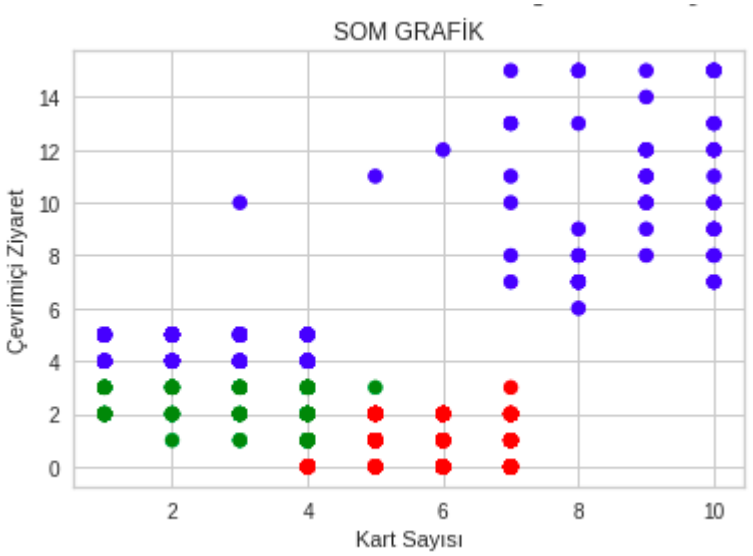
4.1) K-MEANS GRAFİĞİ



4.2) HİYERARŞİK KÜMELEME GRAFİĞİ



4.3)SOM GRAFİĞİ



Değerlendirme:

Verilerin küme merkezlerine olan uzaklıkları her bir grafik için incelendiğinde bu veri seti için hiyerarşik kümeleme çok daha iyi sonuç vermiştir.

Elbow yöntemi sayesinde k değerinin kaç olduğuna karar veriyoruz.

Burada 3 algoritma arasından som algoritmasının diğer iki algoritmaya kıyasla daha verimsiz olduğunu gözlemlenmektedir. Bu sonuçlar algoritmaların birbirlerine karşı üstünlük kurduğu anlamına gelmemektedir. Veri setlerine bağlı olarak sonuçlar değişmektedir.

Biz burada iyileştirme olarak Elbow yöntemini kullandık. Bu sayede k değerinin olabildiğince en iyi değerini bulmaya çalıştık.

5)Kaynakça

[1] Veri Seti

https://www.kaggle.com/ihsanalhasan82/customer-segmentation/data