BİTİRME PROJESİ

Haftalık Rapor – 27.05.2022

27 MAYIS 2022

KIRIKKALE ÜNİVERSİTESİ – BİLGİSAYAR MÜHENDİSLİĞİ İÖ

AHMET MUNGAN – 160255081

İÇİNDEKİLER

ÖZET

Metin madenciliğinde geliştirilen projenin sonuna gelinmiştir. Kullanılan makine öğrenmesi yöntemi ile özet çıkarımı yapılmıştır. Doğrulama yapılamamış olma nedeni, skorların sonuçları, ara yöntemlerin sonuca etkisi değerlendirilmiştir. K-means algoritmasında parametrelerin seçimleri, dirsek yöntemi ve birlikte kullanılan çarpıtma yöntemi ele alınmış ve uygulanıp sonuçları verilmiştir.

K – MEANS İLE KÜMELEME

Ortalamalara bağlı ayırt edicilik matrisini k-means ile kümeleme algoritmasına sokup özet çıkarımında bulunulacaktır. Burada k-means kullanımının nedeniyle ilgili dikkat edilmesi gereken birkaç husus vardır.

Birinci husus; veri seti gerçek hayat verisidir. Yani yorum yapanların tamamı bireysel kişiler ve kişiden bağımsız olarak selenium ile otomatik çekilen verilerdir. Dolayısıyla klasik metin özeti çalışmalarında referans ve makine özeti bulunmaktadır. Bu sayede metin özetine has olan rouge metrikleri hesaplanabilmektedir. Fakat bu çalışmada sabit bir veri seti olmamakla birlikte, referans özeti de mevcut değildir. Sadece çıkarımlar yapabilmek mümkündür. Bu durumda performans ölçütleri ise kümelemenin çıktıları olmalıdır.

İkinci husus; hızlı çalışması gereken bir algoritma olması gerekmektedir. Çünkü bu sistemi kullanacak kişilerin normal bir kullanıcı da olabileceği gibi, bir programcı/yazılımcı da olabilmektedir. Ya da aktif bir sisteme dahil edilecek framework olarak da düşünülebilir. Dolayısıyla hız önemli olduğu için ve karşılaştırılacak bir özet olmadığından, öğrenme sürecinin de uzun süreci göz önünde bulundurulduğunda kümeleme kullanılması avantajdır.

Üçüncü husus; özet konusu dil bazlı düşünüldüğünde belirli bir oranda dökümanın küçültülmesi gibi bir durum söz konusu değildir. Dolayısıyla özet sayısının çıktısı belirli bir büyüklükte olması kesin çizgilerle gerekmektedir denemez. Buradan hareketle doküman sayısının büyüklüğüne bağlı, ayırt edicilik skorlarına bağlı olarak çıkarımlarda bulunmak yararlı olabilir.

Dördüncü husus; En ayırt edici kümelerin merkez noktaları baz alınarak bir özet çıkarmak mümkündür. Bu özetler dökümanı tam anlamıyla temsil eder mi sorusunun cevabı ancak ve ancak referans özeti elde edilirse cevaplanır. Ayrıca ayırt edicilikleri ortalama olan (yani dökümanların ayırt edici özetleri) dökümanlar özet çıkarımı için kullanılabilir.

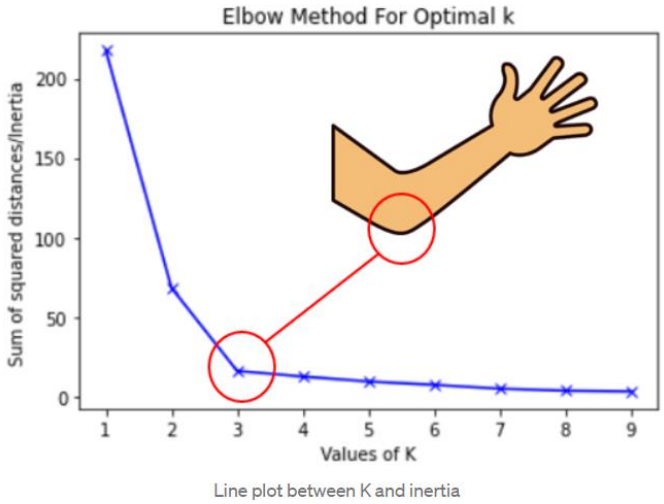
Bu dört hususlar dikkate alınarak en iyi kümeleme algoritması oluşturulmaya çalışılacaktır.

Elbow Yöntemi

K-means algoritmasında en büyük problemlerden biri k sayısının bulunmasıdır. Burada k sayısını optimizasyon algoritmaları ile deneyerek en iyi k sayısının bulunabileceği bir gerçektir. Fakat bu projenin bel kemiğini oluşturan ve k sayısının bulunması için bir yöntem kullanılmıştır.

Burada k sayısı için distortion score ile kıyaslamalı bir yöntem kullanılmıştır. Bu yöntemin faydasını anlamak için öncelikle elbow yönteminin ne olduğu ve nerede nasıl kullanıldığı bilinmelidir.

Elbow yöntemi veri setinin kaç kümeye bölüneceği için geliştirilen bir yöntemdir. Bu yöntemde her bir k sayısı yani küme sayısı veri seti üzerinde deneyerek en iyi skoru olan k sayısı seçilir. Her zaman k sayısı bu kadar kolay seçilememektedir. Bazen ise çok belirli bir şekilde grafikte dirsek görülmektedir ve aranan k sayısı bulunmaktadır.

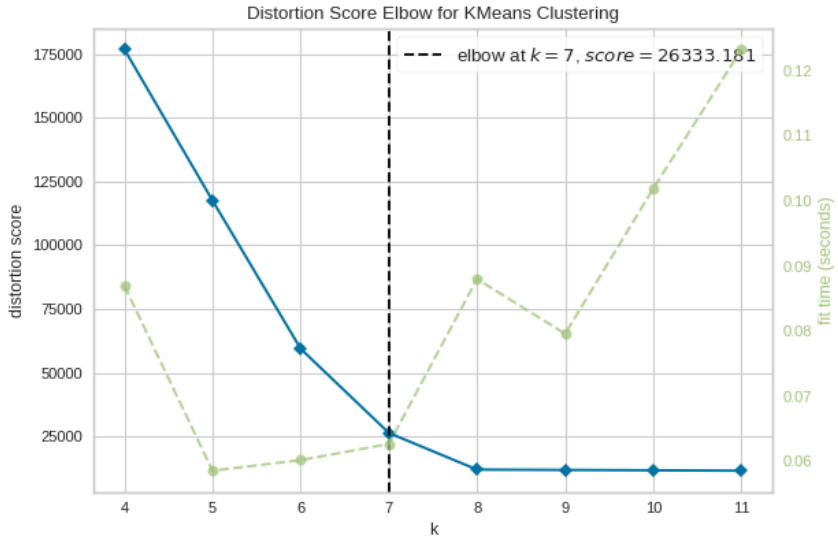


Line plot between K and inertia

Şekil 1

Şekil 1'de örnek veri seti üzerinde dirseği tespit etmek kolaydır. Burada küme sayısı yani k sayısı 3 olarak seçilebilir. Elbow yönteminde bazen şaşırtıcı durumlar oluşabilmektedir. Şekil 1'de k sayısı 2 olarak seçilirse performans ölçütlerinin daha yüksek sonuçlar vereceği kesindir. Çünkü ne kadar az sayıda küme olursa, kümeleme yapmak daha kolaydır. Fakat kesinlik gibi değerleri maalesef düşük olacaktır ve birkaç parametrenin yüksek gelmesiyle algoritmanın doğru çalıştığı düşünülüp yanılmalar yaşanabilir. Fakat her zaman yanılma olmaz, bazen k değeri için 2 sayısı gerçekten

vazgeçilmez olabilir. Burada kümelere bölünen veri setinin küme merkez noktalarının birbirine uzaklıkları önemlidir anlaşılacağı üzere.



Şekil 2

Şekil 2'de ise mavi gösterilen elbow yöntemiyle elde edilen fonksiyondur. Burada dirsek bulmak, herhangi bir ek yöntem kullanmadan bulmak zordur. Dolayısıyla elbow yöntemine ek olarak bir yöntem kullanılırsa, hem algoritma otomatik karar verir hem de en gerçekçi[1] sonuçlara ulaşılabilir.

Elbow yönteminde yukarıda Şekil 2'de açık yeşil renkle gösterilen distortion (çarpıtma) fonksiyonunun hayat kurtarıcı olabilmektedir. Çarpıtma fonksiyonu ile elbow fonksiyonunun düzlemde kesiştiği nokta k noktası olarak seçilir. Çarpıtma fonksiyonu matematiksel formda kabaca şu şekilde ifade edilebilir:

$$f(x, y, z, \dots)_{distortion}$$

$$= f(x, y, z, \dots)_{elbow}{}^{-1} * (cluster_{euclidean.distance}(x, y)^2 +$$

$$cluster_{euclidean.distance}(x, z)^2 + cluster_{euclidean.distance}(y, z)^2 + \cdots)$$

şeklinde ifade edilir. Tümevarım ile ifade edilirse:

[1] En gerçekçi olmasının sebebi, en iyi sonuçlar olmadığını vurgulamaktır. En doğru sonuçlar için değil, en gerçekçi ve en iyi temsil eden kümelerin oluşması hedeflenmiştir.

$$f(x, y, z, \dots)_{distortion}$$

$$= f(x, y, z, \dots)_{elbow}{}^{-1} * \sum_{i=0}^{k\_max} \sum_{i=1}^{k\_max} cluster_{euclidan.distance}(i, i+1)$$

Burada bir diğer sorun ise çarpıtma fonksiyonunun elbow fonksiyonunu ondalıklı bir aralıkta kesmesidir. Eğer ondalıklı bir aralıkta keserse, 3 durum mevcuttur:

1. Yuvarlama yapılır. Örneğin kesme noktası 6,79 olsun. 7'ye daha yakın bir ondalık olduğu için k sayısı 7 seçilir ve performans ölçütleri hesaplanır.

2. Yuvarlama yapılamayacağı durumlarda örneğin 6,50 ve 6,49 gibi kesme noktaları olabilir. Bu durumda yuvarlama yerine integral ile iki fonksiyon arasında kalan noktaların alanları hesaplanır. Hangi alan daha küçükse (hangi tam sayı noktasına daha yakınsa) o nokta k sayısı olarak seçilir.

3. İki fonksiyon arasında integral ile hesaplanan alanların eşit çıkması durumu mevcuttur. Çok düşük bir ihtimal olsa da mümkündür. Çarpıtmanın yeterli düzeyde olmadığında (veri setinin çok iyi dağılmış olduğu durumlar) alanların eşit olduğu durumlar oluşabilmektedir. Burada genellikle küçük sayıya yuvarlama gibi çok net ve belirgin olmayan bir yöntem kullanılmaktadır. Dışarıdan hatalı gibi gözükse de küçük k sayısı her türlü iyi sonuçlar vereceği için herhangi bir hata yoktur denebilir.

Bu durumları göz önünde bulunduran 'yellowbrick' isimli alt kütüphane mevcuttur. Bu kütüphane 'sklearn' altında yer almaktadır. Aslında genellikle bu alt kütüphane elbow yöntemini görselleştirmek için kullanılsa da, altındaki bazı sınıfların çok işlevsel olduğu söylenebilir.

Ayrıca elbow yönteminde bir başka sorun ise k sayısının aralığını belirlemektir. Burada aralığı $[0 - length(document)]$ seçersek en iyi sonucun doküman sayısı kadar olduğu gözlemlenir. Teoride ve hesaplanan sonuçlarda çok iyi bir algoritma denebilir fakat gerçekçi düşünüldüğünde, her bir dökümanın bir kümesi varsa buradan hiçbir çıkarım yapılamayacağı çok açıktır. Burada k sayısı; doküman sayısından çok daha küçük ve boyutsal olarak düşünülmelidir. Bazı araştırmalarda k sayısının sınırı en az 2'den başlatıldığı gözlemlenmektedir. 2'nin seçilmesi de binary

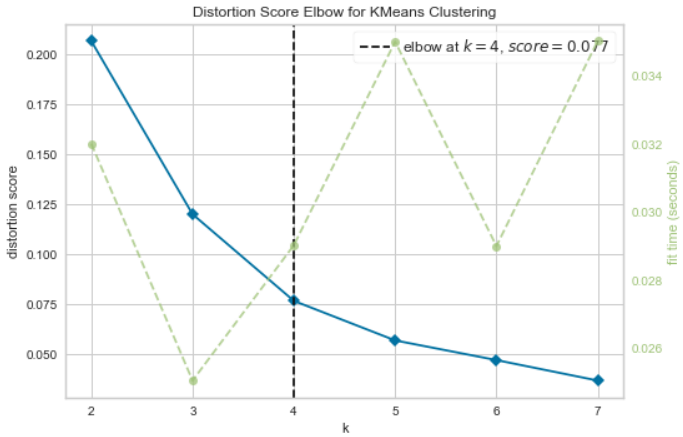
yöntemlere kaydıracağı için en az 2 olması faydalıdır. Üst değeri için ise tamamen veri setinin tanınmasıyla alakalıdır. Öncelikle matematiksel formunu görecek olursak:

$$k = [2, length(document)\%dimension]$$

Bu formda boyut olarak belirtilen tamamen veri setine bağımlıdır. Veri setimize bakıldığında 2 parametreli olduğu için boyuta 2 demek büyük bir hata olacaktır. Çünkü doküman sayılarına bakıldığında genellikle çok büyük sayılar göze çarpmamaktadır. Dolayısıyla iyi bir temsil olmayacaktır. Fakat boyut 3 olursa daha geniş bir aralık meydana gelmektedir. Bu da elbette performans metriklerinin skorunu düşürecektir fakat daha gerçekçi bir aralık seçilmiş olacaktır. Hatta bazı denemeler sonucu skorların da çok büyük değişkenlik göstermediği, ek olarak daha iyi kümeleme yapıldığı görülmüştür.

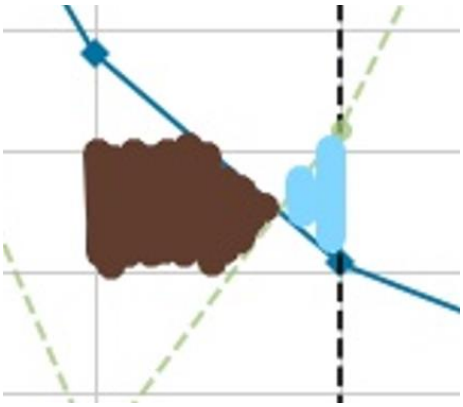
Code 1

```
>>> k_min = 2
>>> boyut = 3
>>> k_max = boyut*len(dokumanlar)//100
>>> elbow = agirlik.copy()
>>> kmeans_elbow = KMeans()
>>> visualizer = KElbowVisualizer(kmeans_elbow, k = (k_min, k_max))
>>> visualizer.fit(elbow)
>>> print("Elbow Yönteminden K Sayısı: ",visualizer.elbow_value_)
>>> print(k_min, k_max)
>>> visualizer.poof()
Elbow Yönteminden K Sayısı: 4
(2, 8)
```



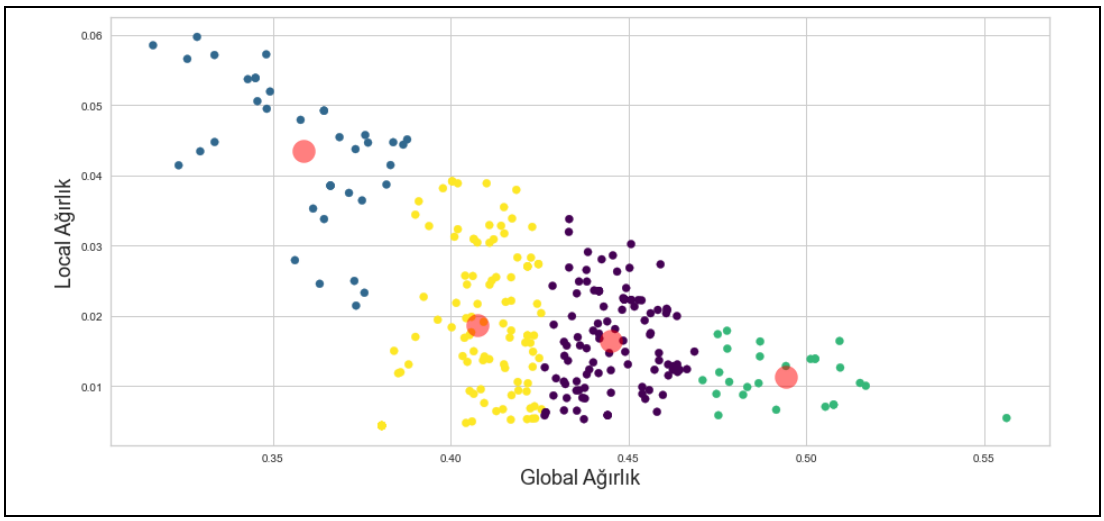

```
<AxesSubplot:title={'center':'Distortion Score Elbow for KMeans Clustering'}, xlabel='k', ylabel='distortion score'>
```

Code 1'de yukarıda daha evvel bahsedilen yöntemleri kullanan bir kod yazılmıştır. Code 1'de çıktı görselde fark edileceği üzere k sayısı ondalıklı bir aralıkta kesişmiştir. Fakat 4'e yakın kısmın alanı gözle bile daha küçük olduğu anlaşılmaktadır.



Şekil 3

Şekil 3'te Code 1'deki çıktının alanların boyanmış kısmından bahsedildiği unutulmamalıdır. Alanların büyüklüğü ve kesişimin [3,4] kapalı aralığında nereye yakın olduğu ortadadır.



Şekil 4

Şekil 4'te k-means algoritmasının çıktısı çizdirilmiştir. Bu çizimde kümelerin birbirine çok yakın olduğu görülmektedir. Dolayısıyla 4 küme ile ifade etmek iyi bir sonuç gibi gözükmektedir.

Ayrıca Şekil 4'te göze çarpan bir diğer nokta ise, ağırlıkların $f(x) = -kx$ fonksiyonunun üstüne ve fonksiyonun altında kalan alanda toplandığı görülmektedir. Bunun nedeni ise ayırt edicilikleri maksimum seviyeye gelse bile normalize edilmiş olan verilerin ne kadar dar ve istenen bir aralıkta olduğu gözükmektedir. Fakat bu durumun belli başlı sorunları mevcuttur. Kümeleme algoritmaları temelinde, yukarıda Şekil 4'te verilen koordinat sisteminden örnekle; (0.55, 0.06) noktasına da veri

beklemektedir. Buraya gelecek olan veriyi kümelemek çok zor olacağı için skorlar düşüktür. Eğer makineye daha farklı denetimli öğrenme yöntemleriyle hiçbir şekilde $f(x) = -kx$ fonksiyonunun üstüne veri çıkamayacağını öğretmek mantıklı olabilir. Bu sayede kümelemenin sonuçları gerçekten istenen düzeye gelebilir. Bu fonksiyonun üstüne çıkılamayacağının en büyük örneği ayırt edicilik skorlarının hiçbir zaman çok aykırı olmamasından kaynaklıdır. Çok aykırı ayırt edicilik olması mümkün değildir çünkü logaritma ile mutlaka muhattap olduğu için, dökümanların potansiyeli bu şekilde dar aralıkta olacağı kesindir. Bu kesinliğin matematiksel ispatı yanı sıra, 54 farklı restoran ile deneme sonucu fonksiyonun değişmediği gözlemlenmiştir.

Performans Metrikleri

Code 2

```
>>> silhouette_skor = silhouette_score(agirlik, kumeler, metric='euclide
an')
>>> silhouette_kesiti = silhouette_samples(agirlik, kumeler, metric='euc
lidean')
>>> dunn_endeksi = davies_bouldin_score(agirlik, kumeler) * 100
>>> print("Silüet Skoru: ", silhouette_skor)
>>> print("Dunn Endeksi: ", dunn_endeksi)
>>> print("Silüet Kesitleri: ", silhouette_kesiti)
Silüet Skoru:  0.4613917109886235
Dunn Endeksi:  66.24258429097678
Silüet Kesitleri:  [ 0.23925533  0.57706723  0.31419026  0.59808431  0.3
1419026  0.04282704
  0.42846597  0.42416993  0.70115213  0.50864293  0.59721062  0.50504143
  0.55316103  0.37290103  0.52536931  0.54653223  0.52981671  0.53733344
  0.42458542  0.50088919  0.68007955  0.44430341  0.41993773  0.31954686
  0.44371996  0.6038485   0.60653063  0.43145523  0.57162816  0.58929129
  0.62473399  0.63871772  0.60982325  0.3845447   0.55268588  0.43344483
  0.43984936  0.39651298 -0.0571371   0.58697681  0.6092878   0.45202547
  0.53862333  0.25433067  0.08558307  0.33321462  0.42797665  0.46436739
  0.29549478  0.5831709   0.25044952  0.43579849  0.69103741  0.39962679
  …]
```

Code 2'de görüldüğü üzere kümelemeye ait skorlar bulunmaktadır. Silüet skoru (-1, 1) açık aralığında değerler almaktadır. Eğer silüet skoru negatif bir değer çıkıyorsa bir noktada hata vardır demektir. Aynı şekilde 1'e ne kadar yakın olursa skor o kadar iyidir denebilir. Burada 0,46~ gibi bir değer alındığı görülmektedir. Daha iyi sonuçlar elbette alınabilir fakat k sayısını 2 seçerek gerçeklikten uzaklaşarak iyi gibi gözüken 0,78~ gibi yüksek skorlara ulaşılabilir. Ayrıca silüet skorunun düşük

olmasının sebeplerinden biri de $f(x) = -kx$ fonksiyonu ile ifade edildiğini öğretmememizden kaynaklıdır.

Dunn endeksi ise yine silüete benzer (0, 1) açık aralığında değer alan bir skordur. Bu skor merkezlerin birbirine olan uzaklıklarına dayandığı için ve uzaklıklar genellikle az olduğu için %66~ gibi bir değer alınmıştır.

Silüet kesitleri ise her bir kümeden alınan örneklerin karşılaştırmasıdır. Burada tüm küme elemanları dahil edilmektedir. Çok yakın ve hatalı kümeleme yapılan değerler negatif olarak ifade edilir. Aynı şekilde (-1, 1) açık aralığında her bir uzaklık değer almaktadır. Code 2'de silüet kesitlerinin çıktılarında göze çarpan birkaç değer mevcuttur. -0,57~ gibi negatif bir değer istenmeyen durumdur. Fakat genele vurulduğunda çok az kesitin sorunlu bir küme olduğunu göstermektedir. Dolayısıyla bu işin doğasında olan ve olağan bir sonuç denebilir.

UZAKLIKLARA GÖRE ÖZET ÇIKARIMI

Makine öğrenmesiyle tespit edilen kümelerin merkezlerine göre, kümeleri en iyi temsil eden yani küme merkezlerine en yakın olan dökümanlar özet olarak seçilecektir. Öncelikle her bir dökümanın sınıfı (ya da kümesi) belirlenmelidir. Aslında kümesi belirlidir fakat veri setini artık düzenlemek şarttır. Neticede elde edilen onca durumdan çıkarımla yapılan kümelemeyi bir şekilde depolamak gerekir. Veri setini depolamak yerine ayırt edicilik matrisine kümelerinin eklenmesiyle ve bu dökümanların kümelerine olan uzaklıklarının eklenmesiyle yeni bir matris elde edilir.

Dökümanların kümelerine olan uzaklık için türlü yöntemler olsa da iki boyutta geometriden bildiğimiz iki noktanın birbirine uzaklığı formülü kullanılmıştır. Kümelemeye uygun düzenlenince matematiksel formu: (k merkez noktalarıdır.)

$$geometric_{distance} = \sqrt{(k_x - document_x)^2 + (k_y - document_y)^2}$$

şeklinde ifade edilebilir.

Code 3

```
>>> agirlik["Class"] = kumeler
>>> uzaklik = []
>>> k = 0
>>> for i in agirlik.index:
>>>     uzaklik.append(math.sqrt(((agirlik["Lij"][i] - merkezler[kumeler
[k]][0])**2) + ((agirlik["Gi"][i] - merkezler[kumeler[k]][1])**2)))
>>>     k += 1
>>> agirlik["Uzaklik"] = uzaklik
>>> agirlik
```

	Lij	Gi	Class	Uzaklik
D0	0.017211	0.421588	3	0.014278
D1	0.019134	0.409237	3	0.001926
D2	0.004382	0.380573	3	0.030355
D3	0.015387	0.438197	0	0.006874
D4	0.004382	0.380573	3	0.030355
...	...	...	...	...
D272	0.045087	0.387748	1	0.029066
D273	0.027325	0.458976	0	0.017689
D274	0.029077	0.438558	0	0.014113
D275	0.004382	0.380573	3	0.030355
D276	0.033765	0.364382	1	0.011240

277 rows × 4 columns

Code 3'te görülen çıktılar ile matrise 2 farklı özellik eklenmiş ve bu şekilde kaydedilir. Eklenmesiyle algoritma maliyeti artmaz çünkü makine öğrenmesi yapılmış haldedir ve işlenmeye dahil edilmeyen kısımlardır. Sadece etiketleme ve merkez noktalarına olan uzaklıkları bulunup eklenmiştir.

Code 4

```
>>> ozet_dokumanlar = []
>>> ozetler = []
>>> for i in range(visualizer.elbow_value_):
>>>     ozet_dokumanlar.append([str(i), 500])
>>> for i in agirlik.index:
>>>     if ozet_dokumanlar[agirlik["Class"][i]][1] > agirlik["Uzaklik"][
i]:
>>>         ozet_dokumanlar[agirlik["Class"][i]][1] = agirlik["Uzaklik"]
[i]
>>>         ozet_dokumanlar[agirlik["Class"][i]][0] = i + "D"
>>>         ozet_dokumanlar[agirlik["Class"][i]][0] = ozet_dokumanlar[ag
irlik["Class"][i]][0][1:-1]
>>> aij = []
>>> indexler = []
>>> for i in agirlik.index:
>>>     aij.append(agirlik["Lij"][i] * agirlik["Gi"][i])
>>> aij.sort()
>>> for i in agirlik.index:
>>>     if agirlik["Lij"][i] * agirlik["Gi"][i] == aij[len(aij)//2]:
>>>         a = i + "D"
>>>         indexler.append(int(a[1:-1]))
>>>     if agirlik["Lij"][i] * agirlik["Gi"][i] == aij[len(aij)//2 + 1]:
>>>         a = i + "D"
>>>         indexler.append(int(a[1:-1]))
>>>     if agirlik["Lij"][i] * agirlik["Gi"][i] == aij[len(aij)//2 - 1]:
>>>         a = i + "D"
>>>         indexler.append(int(a[1:-1]))
>>> for i in range(len(ozet_dokumanlar)):
>>>     ozetler.append(orijinal_yorumlar[int(ozet_dokumanlar[i][0])][0])
>>> for i in range(len(indexler)):
>>>     ozetler.append(orijinal_yorumlar[indexler[i]])
>>> ozetler
['çok geç teslimat sağlandı soğuk ürün para zaiyati sadece',
 'biraz soğuk geldi sipariş onun dışında güzeldi ',
 'eski tadı yok ',
 'sipariş sıcak ve hızlı geldi.lezzetide 10 üzerinden 10 ustamin ellerin
e sağlık.',
 'soğuktu ve lezzetli degildi...',
 'çok soğuk ve kuru geldi hava soğuk anlıyorum ama bu kadarda soğuk gelm
esin ustad yeme şansımız yoktu çok kurumustu teşekkür ederim....',
 'çok güzeldi hamurun kalınlığına güzeldi çok ince olunca güzel olmuyor'
]
```

Code 4'te artık uzaklıkları ve öğrenmesi biten veri setinden özet çıkarma algoritmasıdır. Ele alınan restoranın son dönemlerde kötü yorumlar aldığı göz önüne

alınırsa Code 4'te özetlerin çıktısı da bu şekilde olduğu ve genellikle gelen yemeğin soğuk olmasını sistemin yakalamış olması aslında restoranın yavaş bir gönderim yaptığına dair bir özet çıkarılabilir. Bu özetlerin sayısı ve içeriğinden yola çıkarak restoran bilgilendirilebilir, restoran bu konularda kendini düzeltebilir veya iyi yanlarını perçinleyebilir.

Code 4'te fark edilen bir durum vardır. Code 4'teki çıktılarda durak kelimelerin de bulunduğu söylenebilir. Çünkü durak kelimeler de orijinal yorumlarda olduğu için bu kelimeler hiçbir şekilde makine öğrenmesine dahil edilmedi. Dahil edilmiş olsa neler olurdu diye deneyip sonuçların çok büyük oranda değişmediği gözlemlenmiştir. Fakat değişim olmaması algoritma maliyetini arttırdığı gerçeğini değiştirmemektedir. Algoritmanın maliyeti kelime bazında arttığı için ara matris hesaplamaları üssel boyutlarda arttığı gözlemlenmiştir. Dolayısıyla ön işleme süreçlerinin ne kadar önemli olduğu burada ortaya çıkmıştır.

EKLER

Bitirme Projesi 2’ye ait doküman, program kodu, haftalık rapor ve ek bilgilerin paylaşıldığı github linki için tıklayınız. (Güvenlidir.)