KÜMELEME ALGORİTMALARININ CPU VE GPU PERFORMANSLARININ ANALİZİ

T.C.
BURSA ULUDAĞ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

KÜMELEME ALGORİTMALARININ CPU VE GPU PERFORMANSLARININ ANALİZİ

Melek GÜLER MANAV
501731004

Doç. Dr. Metin BİLGİN
(Danışman)

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

BURSA – 2022

TEZ ONAYI

Melek GÜLER MANAV tarafından hazırlanan "KÜMELEME ALGORİTMALARININ CPU VE GPU PERFORMANSLARININ ANALİZİ" adlı tez çalışması aşağıdaki jüri tarafından oy birliği ile Bursa Uludağ Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Danışman : Doç. Dr. Metin BİLGİN

Başkan : Doç. Dr. Metin BİLGİN — İmza
000-000-000-000
Aaaaaaaaa Üniversitesi,
Aaaaaaaaaaaa Fakültesi,
Aaaaaaaa Aaaaaaaaaaa Anabilim Dalı

Üye : Aaaaa. Dr. Aaaaaaaa AAAAAAAA — İmza
000-000-000-000
Aaaaaaaaa Üniversitesi,
Aaaaaaaaaaaa Fakültesi,
Aaaaaaaa Aaaaaaaaaaa Anabilim Dalı

Üye : Aaaaa. Dr. Aaaaaaaa AAAAAAAA — İmza
000-000-000-000
Aaaaaaaaa Üniversitesi,
Aaaaaaaaaaaa Fakültesi,
Aaaaaaaa Aaaaaaaaaaa Anabilim Dalı

Üye : Aaaaa. Dr. Aaaaaaaa AAAAAAAA — İmza
000-000-000-000
Aaaaaaaaa Üniversitesi,
Aaaaaaaaaaaa Fakültesi,
Aaaaaaaa Aaaaaaaaaaa Anabilim Dalı

Üye : Aaaaa. Dr. Aaaaaaaa AAAAAAAA — İmza
000-000-000-000
Aaaaaaaaa Üniversitesi,
Aaaaaaaaaaaa Fakültesi,
Aaaaaaaa Aaaaaaaaaaa Anabilim Dalı

Yukarıdaki sonucu onaylarım

Prof. Dr. Hüseyin Aksel EREN
Enstitü Müdürü
../../….

B.U.Ü. Fen Bilimleri Enstitüsü tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- tez içindeki bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,
- görsel, işitsel ve yazılı tüm bilgi ve sonuçları bilimsel ahlak kurallarına uygun olarak sunduğumu,
- başkalarının eserlerinden yararlanılması durumunda ilgili eserlere bilimsel normlara uygun olarak atıfta bulunduğumu,
- atıfta bulunduğum eserlerin tümünü kaynak olarak gösterdiğimi,
- kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- ve bu tezin herhangi bir bölümünü bu üniversite veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı

beyan ederim.

…/…/……

Melek GÜLER MANAV

ÖZET

Yüksek Lisans Tezi

KÜMELEME ALGORİTMALARININ CPU VE GPU PERFORMANSLARININ ANALİZİ

Melek GÜLER MANAV

Bursa Uludağ Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Metin BİLGİN

Teknolojinin ilerlemesi, teknolojideki rekabeti her geçen gün artırmaktadır. Teknolojide ilerleme artarken müşterileri ve kullanıcıları memnun etmek güçleşmektedir. Çeşitli teknolojik aygıtlar sebebiyle üretilen veri miktarı artmakta bu da firmaların eldeki verileri analiz etmeleri için farklı metotlara yönelmelerine sebebiyet vermektedir. Günümüz dünyasında verilerin analiz edilmesi ve yorumlanması çok önemli olduğundan bu işlemi elle yapmak yerine makinelere yaptırma gereği ve ihtiyacı doğmuştur. Eldeki verilerin etiketlerinin bilinmediği durumlarda bunları analiz edebilmek adına kümeleme algoritmalarından yararlanılmaktadır. Kümeleme algoritmaları verileri gruplara ayırmaktadır ve bu sayede verilerin analiz edilmesi, yorumlanması kolay hale getirilmektedir.

Bu tez çalışmasında, mevcutta kullanılan beş farklı kümeleme algoritmasının CPU ve GPU üzerindeki performansları araştırılmış ve bunları tespit etmeye yönelik deneysel çalışmalar gerçekleştirilmiştir. Kümeleme algoritmalarının performanslarını ölçebilmek adına yapılan deneysel çalışmalarda e-postalardan oluşan Enron veri kümesi kullanılmıştır. Çalışmada kümeleme algoritmaları olarak; model bazlı Cobweb, yoğunluk bazlı Dbscan, grid bazlı Clique, bölümlemeli K-Means, hiyerarşik olarak ise Birch algoritmaları seçilmiştir. Deneysel çalışmalar için gerekli ortam Python dilinde Google Colab üzerinde gerçekleştirilmiştir. Deneysel çalışma sonuçları grafikler ve tablolar ile ifade edilerek analiz sonuçları sunulmuştur.

Anahtar Kelimeler: GPU, CPU, Makine Öğrenmesi, Kümeleme Algoritmaları

ABSTRACT

MSc Thesis

ANALYSIS OF CPU AND GPU PERFORMANCES OF CLUSTERING ALGORITHMS

Melek GÜLER MANAV

Bursa Uludağ University
Department of Computer Science

Supervisor: Assoc. Prof. Metin BİLGİN

The advancement of technology increases the competition in technology area day by day. As technology advances, it becomes more difficult to satisfy customers and users. The amount of data produced with the help of various technological devices is increasing, which causes companies to turn to different methods to analyze the data at hand. Since the analysis and interpretation of data is very important in today's world, the need and need to have this process done by machines has arisen instead of doing it manually. In cases where the labels of the available data are not known, clustering algorithms can be used to analyze them. With the help of clustering algorithms, the data can be grouped and made easier to analyze and interpret on this occasion.

In this thesis, the performances of five different clustering algorithms currently used on CPU and GPU were investigated and experimental studies were carried out to detect them. In order to measure the performance of clustering algorithms, a dataset consisting of e-mails that name is Enron was used in experimental studies. As clustering algorithms in the study; model based Cobweb, density based Dbscan, grid based Clique, segmented K-Means, hierarchical Birch were selected. The necessary environment for the experimental studies was carried out on Google Colab in Python language. Experimental study results are expressed with graphs and tables, and analysis results are presented.

Key words: GPU, CPU, Machine Learning, Classification Algorithms

ÖNSÖZ VE/VEYA TEŞEKKÜR

Beni tezim konusunda her konuda bilgilendiren desteğini esirgemeyen tez danışmanım Doç. Dr. Metin Bilgin'e teşekkürlerimi sunarım.

Akademik hayatım boyunca her zaman başarılarımda en büyük desteği olan canım annem ve babama teşekkürlerimi sunarım.

Üniversitede sınıf arkadaşım olan ve hiçbir zaman benden desteğini esirgemeyen canım eşim Gökhan'a teşekkürlerimi sunarım.

Zamanımın çoğunu alan fakat gülüşleriyle bana başarma enerjimi veren canım çocuklarım Zeynep'ime, Elif'ime ve Ahmet Emir'ime teşekkürlerimi sunarım.

Melek GÜLER MANAV
14/12/2022

İÇİNDEKİLER

Sayfa

KISALTMALAR DİZİNİ

Kısaltmalar	Açıklama
TCP	: Transmission Control Protocol
MWC	: Mobile World Congress
D2D	: Device to Device
DOM	: Document Object Model
AC	: Access Control List
SQLi	: SQL Injection
GPU	: Graphic processing unit (Grafik İşlem Birimi)
KVM	: Kernel Virtual Machine
ESX	: Elastic Search X
CPU	: Central Processing Unit (Merkezi İşlem Birimi)
SQL	: Structured Query Language (Yapısal Sorgulama Dili)
AU	: Activation Unit
DNS	: Domain Name Server
IIS	: Internet Information Services
IP	: Internet Protocol

ŞEKİLLER DİZİNİ

Sayfa

ÇİZELGELER DİZİNİ

Sayfa

1. GİRİŞ

Büyük firmalarda veri boyutu fazla olmaktadır, bu noktada verileri gruplandırmak yani kümelemek hız açısından en önemli adımlardan biri haline gelmektedir. Kullanıcı verilerini kümelemek kullanıcıların isteklerine hızlı cevap vermede en önemli adımlardandır. Daha iyi kümeleme kararları, kısa zamanda daha çok kullanıcıya hizmet edebilmeyi sağlayacağından daha fazla kullanıcı memnuniyeti sağlayacaktır. Fakat firmaların bir yandan da kendi performanslarını göz önünde bulundurmaları gerekmektedir.

Kümeleme yöntemleri için pek çok farklı algoritma yazılmış denenmiş ve başarılı sonuçlar alınmıştır. Örneğin bir galeri sahibi müşterilerinin profillerini, müşterilerin yaşları, gelir durumları dolayısıyla gider durumları ile otomobillere bakış açısı arasında bir fark olup olmadığını belirlemek istemiştir. Araştırmanın sonucunda otomobillerle en çok ilgilenen hedef kitlesini tespit edebilecek ve müşteri memnuniyetini sağlamaya yönelik hizmetlerini farklılaştıracaktır (Alhami 2009). Bu noktada kümeleme algoritmaları devreye girmektedir.

Bu çalışmada amaç, firmaların dikkate alması gereken en önemli ölçütlerin saptanmasında kümeleme analizi algoritmalarının uygun bir yöntem olarak kullanılabileceğini göstermektir. Çalışmada, Enron firmasındaki kullanıcıları yaşlarına, yıllık gelir ve yıllık giderlerine göre kümeleme yapılmaktadır. Bu işlemi yaparken kümeleme algoritmalarının beş yönteminden bir algoritma seçilip kullanılarak doğruluğun arttırılması amaçlanmış ve CPU ve GPU ortamlarında hız karşılaştırması yapılarak elde edilen sonuçlar sunulmuştur. Bu çalışma, literatürde sıkça duyulan müşteri segmentasyonu uygulamalarında bir yöntem olarak kullanılabilir. Ayrıca kullanıcıları daha iyi tanıma ve buna göre reklamlar sunma gibi alanlarda da kullanılabilir.

Çalışma, kümeleme algoritmalarının CPU ve GPU ortamlarında hız olarak karşılaştırıldığı ve yorumlandığı bölümlerle devam etmektedir. Çalışmanın bulguları performans açısından değerlendirilerek tez çalışması sona erdirilmiştir.

2. KURAMSAL TEMELLER ve KAYNAK ARAŞTIRMASI

Kümeleme veya kümeleme analizi, denetimsiz bir öğrenme biçimidir. Kullanıcıların davranış biçimleri veya yaşları, harcama alışkanlıkları, maaşları, harcama miktarları kümeleme yöntemiyle gruplandırılabilmektedir. Son zamanlarda müşteri segmentasyonu uygulamaları aktif yapılmaya başlanmıştır. Yöntem olarak da kümeleme algoritmaları kullanılmaktadır

Aralarından seçim yapabileceğiniz birçok kümeleme algoritması vardır ve tüm durumlar için tek bir en iyi kümeleme algoritması yoktur.

1970 yılının başlarında verilerde verimliliği artırmak için çeşitli çalışmalara başlanmıştır. 1990 yılından itibaren kümeleme algoritmaları hakkında çok farklı makaleler, uygulamalar ortaya konmaya başlanmıştır. Örneğin "Algorithms for Clustering Data" (Jain&Dubes, 1998) kümeleme algoritmaları hakkında yazılmış detaylı makaleler arasındadır (Akın 2008).

Sertbaş ve Çetinkaya CPU ve GPU mimarileri üzerinde uygulanan Derin Öğrenme Algoritmalarının deneysel performans analizini gerçekleştirmişlerdir. Çalışmada CPU'lar derin öğrenme algoritmalarında CPU'lardan daha hızlı ve başarılı sonuçlar vermiştir. Öğrenme aşamasındaki en önemli kriter veri seti büyüklüğü ve işlem süresi olarak kaydedilmiştir. Çalışma sonuçları CPU ve GPU sistemlerinin performans açısından önemli farklar ortaya koyarak gelecekte farklı alanlarda da kullanılması açısından önem taşımaktadır (Sertbaş, Çetinkaya 2021).

Öğüt ve Egemen tarafından Makine Öğrenmesi ve Derin Öğrenme algoritmalarında CPU ve GPU tabanlı hızlandırma tekniklerinin karşılaştırılması tez çalışması yapılmıştır. Çalışmada donanımsal farklılıklar çalışmaya eklenmiştir. Nvidia firmasının GPU Cuda Platformu kullanıldığında normal GPU ortam hızına göre 9 kat hız artışı sağlanmıştır. Bu çalışma GPU'ların CPU'lardan hızlı çalıştığının yanında, donanım değiştikçe alınan sonuçların değiştiğini öngörmüştür(Öğüt, Egemen 2019).

Shrivista ve meslektaşları ise Yapay Zeka ile kodladıkları yeni algoritmanın GPU'lardan daha iyi performans ortaya koyduğunu deneysel olarak açıklamışlardır. Araştırmaya göre, 200 milyon parametreli bir sinir ağını eğitirken NVIDIA GPU' da optimize edilmiş Tensorflow uygulaması denenmiştir. Birde sinir ağlarını eğitmek için Lsh Yapay Zeka algoritması denenmiştir. Algoritma performansı GPU hızını geçmiştir. 2.55 kat daha hızlı

çalışmıştır. Bu çalışma seçilen algoritmanın olumlu etkisini ortaya koymuştur.(Şekercioğlı 2021).

Kümeleme, eldeki verilerin hangileri daha yakın, hangileri daha benzerdir mantığına dayanmaktadır. Kümeleme de elde etiketsiz veriler vardır ve birbirine yakın özellikteki verileri aynı grupta toplamaktadır .Kümeleme, kümeleme sonuçlarının etiketli verilerle karşılaştırma şansı olmadığından denetimsiz öğrenme olarak kabul edilmiştir.

Kümeleme algoritmalarından olan K-Means algoritması, 1957 yılında Cox tarafından ortaya atılmıştır; 1967 yılında MacQueen tarafından K-Means adı verilmiştir. K-Means literatürde GPU ile performansının geliştirilmesi için üzerinde en fazla çalışılan kümeleme algoritmasıdır(MacQueen 1967).

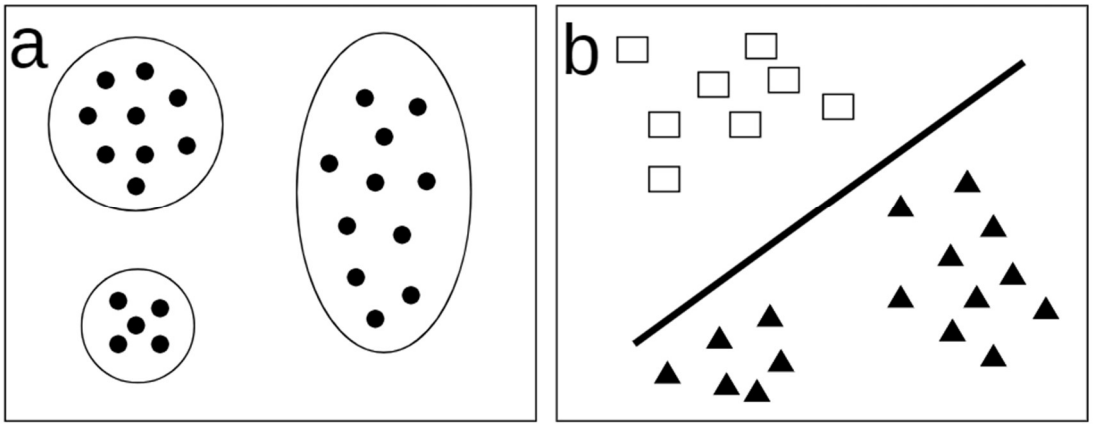
Hall & Hart kümeleme algortimasını GPU ile hızlandırmak için ilk deneysel çalışmayı yapmışlardır. Deneysel çalışmalarını yaptıkları zamanlarda GPU veri doku ünitelerinde saklanmakta idi. Bu durum aynı anda kısıtlı sayıda verinin hesaplanmasına izin veriyordu. Dolayısıyla veri ve boyut sayısında kısıtlamalar çok önemli bir sorundu. Yaptıkları çalışmalarda GPU ve CPU uygulamasına göre 1.5 ile 3 kat hız sağlayabilmiştir(Hall ,Hart 2009).

GPU bellek yapısından daha fazla yarar sağlamak isteyen Böhm ve arkadaşları çalışmalarında saklayıcılar da kullanmışlardır. Yazarların bu metodu GPU ortamlarının çok daha hızlı çalışmasını sağlamıştır.Bu çalışma aynı zamanda iyi bellek yapısına sahip olmayan ortamlarında saklayıcılar sayesinde ne kadar kazanç getirdiğini göstermiştir(Böhm 2009).

Espenshade ve arkadaşları çalışmalarında K-Means algoritmasının GPU ile hızlandırılmış bir uygulamasını Tesla mimarisinden yararlanarak sunmuşlardır. K-Means algoritmasında kümeleri bulanıklaştırmışlardır. Veri seti için en iyi k değerini bulmak için k-medoids algoritmasına entegre edilmiştir. GPU uygulaması, CPU versiyonuna göre 7 kata kadar hızlanma sağlamıştır(Espenshade ve arkadaşları 2009).

2.1. Makine Öğrenmesi

Makine öğrenmesi algoritmaları eldeki verileri kullanarak en iyi modeli oluşturmaya çalışırlar. Yeni veriler geldiğinde önceki verilerle oluşturulan modeli kullanarak yeni gelen verileri analiz etmeye çalışırlar. Eldeki verilerin işlenerek içerisinden anlamlı ve kullanışlı bilgilerin oluşturulmasına veri madenciliği denilmektedir (Çentik 2013). Elimizdeki verilerin sınıflarının ya da etiketlerinin verilmediği ya da bilinmediği durumlarda, veriler içerisinde birbirine benzer olanların gruplandırılarak oluşturduğu şekle kümeleme adı verilmektedir . (Amasyalı 2008). Her küme arasında maksimum sınıf içi benzerlik ve minimum diğer kümeler arası benzerliği vardır. Bu nedenle kümeleme tekniği, etiketlenmemiş veya gruplandırılmamış veriler için uygulanır. Kümeler arasındaki benzerlik metriğini artırmak için öznitelik tanımlanmalıdır. Küme özellikleri, bir kümeyi diğerinden ayıran profilleri tanımlamaktadır. İyi kümeleme tekniğinin performansı, gizlenen kalıpları tanımlayabilme ve maksimum sınıf içi benzerlik üretme ve diğer nesneler arasındaki sınıflar arası benzerliği azaltma yeteneği ile ölçülür. Hiyerarşik yöntemler, verilerin alt kümelerine ayrılmasıyla bir dendrogram oluşturur. Yoğunluğa dayalı yöntemler, veri alanında bulunan ve düşük yoğunluklu gürültü bölgeleriyle birbirinden ayrılan yoğun bölgeleri bulur (Şekil 2.1). Çalışmamızda mail veri seti olan Enron veri seti ile çalışılmıştır.



Şekil 2.1 : Veri madenciliği kümeleme gösterimi (Alsmadi, Alhami 2009)

2.2. Kümeleme Algoritmaları

Kümeleme etiketsiz veriler üzerinde denetimsiz olarak gruplama sağlar. Her küme arasında maksimum sınıf içi benzerlik ve minimum kümeler arası benzerliği vardır. Bu nedenle kümeleme tekniği, etiketlenmemiş veya gruplandırılmamış bir veri kümesi arasındaki içsel gruplandırmayı tanımlamak için uygulanır. Bu teknik, sınıflar bilinmediğinde ve sınıflandırmada kullanıldığı gibi sınıf etiketli örnekleri analiz etmediğinde kullanılabilir. Küme özellikleri, bir kümeyi diğerinden ayıran profilleri tanımlamak için analiz edilebilir. İyi kümeleme tekniğinin performansı, gizlenen kalıpları tanımlayabilme ve maksimum sınıf içi benzerlik üretme ve diğer nesneler arasındaki sınıflar arası benzerliği azaltma yeteneği ile ölçülür.

Kümelemede benimsenen teknikler hiyerarşik yöntemler, bölümleme yöntemleri, yoğunluğa dayalı yöntemler, Grid Bazlı yöntemler ve Model Bazlı yöntemler olmak üzere beş yönteme ayrılır (Çizelge 1.1). Bölümleme yöntemi, mesafe fonksiyonuna göre her bir kümeyi optimize edebilen kümeleri belirlemek için kullanılabilir. Hiyerarşik yöntemler, veri tabanının ayrışmasıyla bir dendrogram oluşturur. Yoğunluğa dayalı yöntemler, veri alanında bulunan ve düşük yoğunluklu gürültü bölgeleriyle birbirinden ayrılan yoğun bölgeleri bulur. Hiyerarşik yöntemler, üstte tek bir veya tümü dahil küme ve tek tek noktaların altından tek bir küme kümesi kullanarak iç içe geçmiş bir küme dizisi oluşturur. Yukarıdan aşağıya doğru oluşturulan hiyerarşi veya aşağıdan yukarıya şekilde ve aşırı uçlara kadar uzatılmaz. İstenilen sayıda küme elde edildiğinde, birleştirme veya bölme işlemi durdurulur. Bu yaklaşımda karşılaşılan ana dezavantaj, birleştirme veya bölme yapıldıktan sonra geri dönüşümlü olmamasıdır. Bu hiyerarşik tekniği kullanarak geliştirilen birçok algoritma vardır, bunlardan bazıları aşağıda listelenmiştir. Cure iki yeni fikir kullanan aşağıdan yukarıya doğru bir yöntemdir. İlk olarak, kümeler merkezcil içermeyen sabit sayıda iyi dağılmış nokta kullanılarak oluşturulur.

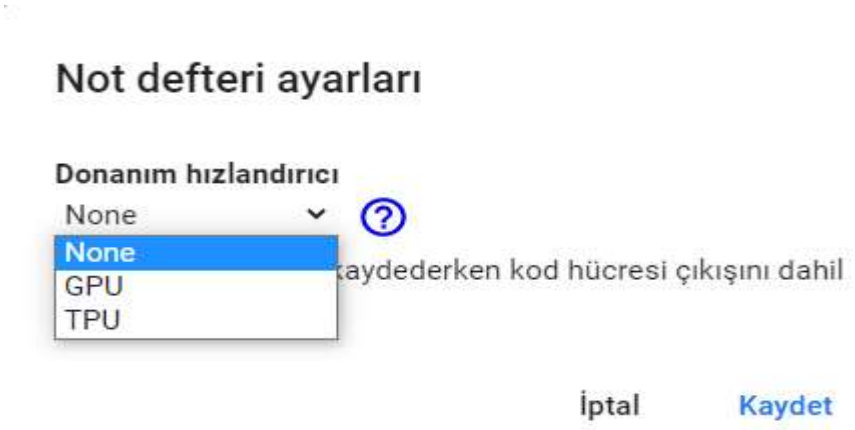
Çizelge 1.1: Kümeleme Algoritmaları (Anonim 2020d'den değiştirilerek alınmıştır)

Bölümlemeli	Hiyerarşik	Yoğunluk Bazlı	Grid Bazlı	Model Bazlı
1. K-Means	**1. Birch**	**1. Dbscan**	1.Wave-Cluster	1. Em
2. K-medoids	2. Cure	2. Optics	2. Sting	**2. Cobweb**
3. K-modes	3. Rock	3. Dbclasd	**3. Clique**	3. Classit
4. Pam	4. Chameleon	4. Denqlue	4. OptiGrid	4. SOMs
5. Clarans				
6. Clara				
7. Fcm				

2.3. Google Colapse Kullanımı

Google Colab (Google Colaboratory), derin öğrenme projeleri üzerinde çalışanlar için bulut tabanlı, ortak çalışmaya dayalı bir programlama ortamıdır. 2017 yılında Google tarafından piyasaya sürülmüştür. Bulut hesabından entegre olduktan sonra kod geliştirme ortamlarında çalışma yapılabilmektedir.

Google Colab ortamında CPU ve GPU ortamları arasında geçiş yapılabilerek istenilen ortamda çalışılabilmektedir. Colab ortamında Not defteri ayarlarında basit bir yöntem ile iki ortam arasında geçiş yapılabilmektedir (Şekil 2.2).



Şekil 2.2: Google Colapse GPU ve CPU ayarı

Colabda yapılan çalışmalar ve kodlamalar Google Drive'da saklanmaktadır. Kodlar hesaba yönelik olan sanal makinede çalıştırılmaktadır. Colab ortamında istenilen kütüphaneler

kolaylıkla indirilebilmektedir.

Colab ortamında çalışma zamanı istenilen zaman birimi cinsinden ekranın altında kod bloğu aracılığı ile listelenmektedir.

3. MATERYAL ve YÖNTEM

Bu bölümde, çalışmada kullanılan veri kümesi ve beş farklı kümeleme algoritması ile ilgili bilgiler sunulacaktır. Çalışmada kullanılan algoritmalarından ilki olan K-Means , veri madenciliği yönteminde en çok kullanılan algoritmadır. İkinci kullanılan algoritma olan Dbscan ise hızlı olması nedeniyle popüler hale gelmiştir. Üçüncü kullanılan algoritma, kümeleme için gerekli bilgiyi yakalama amacıyla bir ağaç oluşturan Birch algoritmasıdır. Kullanılan bir diğer algoritma yoğunluğa göre kümeleme yapan Clique algoritmasıdır. Ve son olarak bir karar ağacı mantığıyla çalışan Cobweb algoritmasıdır. Bu tezde, Python'da en iyi kümeleme algoritmalarının nasıl kullanılacağı, performansları, hangi durumda hangi algoritmanın daha hızlı ve kullanışlı olduğu tespit edilecektir. Birçok farklı kümeleme algoritması vardır ve tüm veri kümeleri için tek bir en iyi kümeleme algoritması yoktur.

3.1. Veri seti

Bu çalışmada Enron mail veri seti kullanılmıştır. Bu veri seti Enron şirketinde ait 150 kullanıcı bilgilerini içermektedir. 0.5 milyon mail barındırmaktadır. Şirket iflas ettikten sonra bilime yararlı olması açısından veri seti şeklinde veriler topluma sunulmuştur (Çizelge 3.1).

Çizelge 3.1. Enron Veri Seti Görüntüleme

Müşteri No	Cinsiyet	Yaş	Yıllık Gelir $-Sapma	Yıllık Gider$-Sapma
1	Kadın	24	22	23
2	Erkek	26	25	34
3	Erkek	29	34	45
4	Kadın	22	42	56
5	Kadın	31	12	23
6	Erkek	34	56	25
7	Kadın	23	34	37
8	Kadın	25	67	56
9	Kadın	27	56	67
10	Erkek	23	23	34

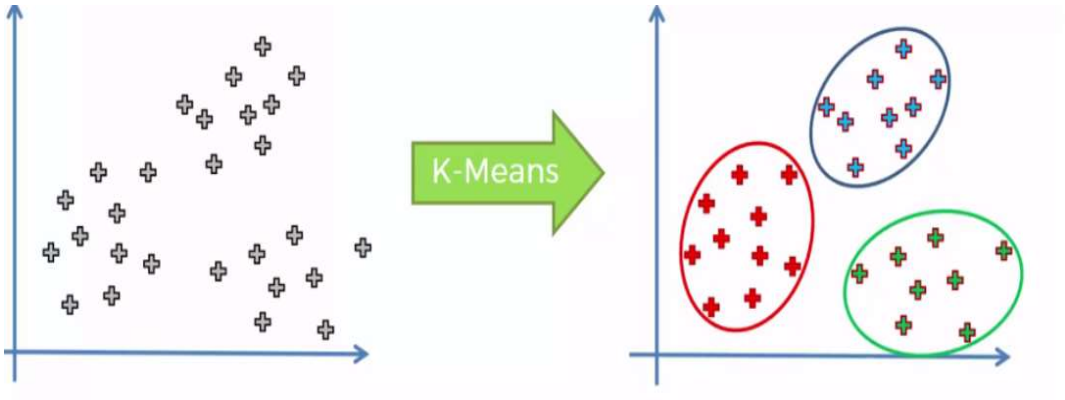
3.2. Kullanılan Kümeleme Algoritmaları

Çalışmada kullanılan kümeleme algoritmalarının detaylı açıklamaları ve Colab üzerinde çalıştırılmış ekran görüntüleri ve performans sonuçları paylaşılacaktır.

3.2.1. K-Means Algoritması

K-Means tüm metrik uzaylarda yaygın olarak kullanılan en popüler kümeleme algoritmasıdır. Başlangıçta k küme merkezinin seçimi rastgele yapılır; Daha sonra tüm noktaları en yakın merkez noktalarına yeniden atar ve yeni bir araya getirilen gruplar için merkez noktalarını yeniden hesaplar. Bu yinelemeli yer değiştirme, kriter fonksiyonu birbirine yakınsayana kadar gerçekleştirilir. Bazıları merkezlerden etkilendiğinden, K-ortalamaları aykırı değerlere ve gürültüye karşı oldukça hassastır. Bu algoritmanın ana avantajı, karesel hata fonksiyonu kullanıldığında bir amaç fonksiyonunu minimize etmesidir. K-Means algoritması aşağıdaki şekilde çalışmaktadır (Şekil 3.1).

- Öncelikle bir k küme sayısı seçilmelidir.
- K küme sayısı seçilmeden önce rastgele k küme oluşturulur ve küme merkezleri belirlenir.
- Her nokta en yakım küme merkezine atanır, yeni küme merkezleri tekrar hesaplanır.
- En uygun k değeri bulunana kadar önceki iki adım tekrarlanır.

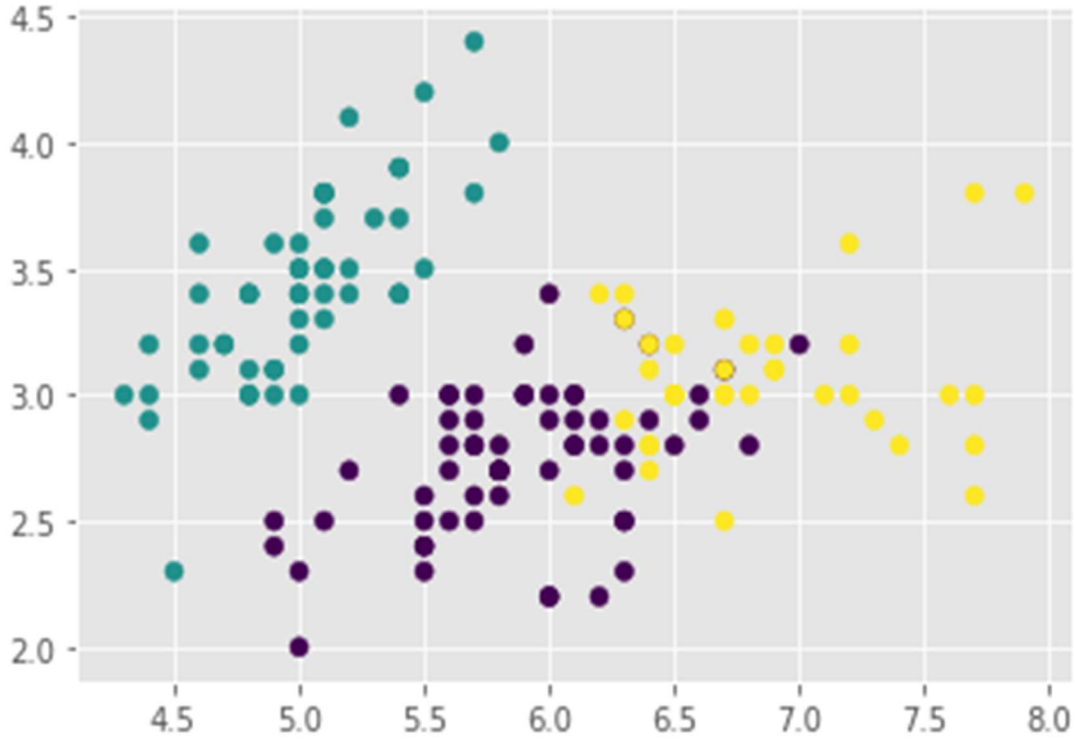


Şekil 3.1: Basit bir K-Means örneği (Maskell 2015)

K-Means algoritması avantajlara, dezavantajlara sahiptir (Çizelge 3.2).

Çizelge 3.2: K-Means avantaj ve dezavantajları (Menzell 2017'den değiştirilerek alınmıştır)

Avantajları	**Dezavantajları**
1. En yaygın kullanılan ve kolay uygulanan algoritmadır. 2. Hesaplamalı olarak daha hızlı yöntemlere sahiptir. 3. Ölçeklenebilirdir. 4. Düşük boyutlu veriler için daha hızlıdır. 5. Sıkı kümeler üretir. 6. Veri büyükse daha fazla alt küme bulunur. 7. Küme numarası belirtilir. 8. Yalnızca iyi şekillendirilmiş kümeler için çalışmaktadır. 9. Sabit sayıda küme yapılabilir. 10. K nin kaç olmasını tahmin etmek işlem gerektirir.	1. Üretilen kümelerin kalitesini karşılaştırmanın zorluğu. 2. K değerini tahmin etme zorluğu. 3. Farklı boyut ve yoğunlukta küresel olmayan veriler oluşabilir. 6. Aykırı değerleri ve gürültüyü tanımlayamıyor. 7. Merkez kavramına sahip verilerle sınırlıdır (merkez).



Şekil 3.2: K-Means ile oluşmuş kümeler

K-Means kümeleme algoritması en çok kullanılan ve en çok üzerinde çalışılmış kümeleme algoritmasıdır. K-Means algoritmasında algoritmaya verilen k değeri küme sayısıdır. Kümeleri belirlerken küm merkezlerininde belirlenmesi gerekmektedir. Küme merkezlerinin belirlenmesi bazen sorun olmaktadır.Çalışma içerisinde K-Means ile oluşmuş kümeler renklendirilmiştir (Şekil 3.2). Aşağıda K-Means kümeleme yönteminde algoritmaya verilen k değerinin en uygunu bulmak için kullanılan elbow yöntemi de anlatılmıştır.

K-Means algoritmasında bir k değeri seçilmesi gereklidir. K küme sayısı kaç olursa daha hızlı olur bunu bulmak için en iyi yöntem elbow yöntemidir. Bunun için tüm k değerleri için önce hesaplama yapılmaktadır. Bu sonuçlar grafiğe aktarıldığında grafikte dirsek şeklinde kırılma olan en düşük değer k değeri olarak seçilmektedir (Şekil 3.3).

Şekil 3.3: Elbow yöntemi (Guven 2019)

Yukarıdaki grafiğe baktığımızda 9 k değeri için yapılan hesaplamalarda k=3 olduktan sonra dirsek şekli oluşmuştur. Dolayısıyla k değeri 3 olarak seçilmiştir. K-Means algoritması kullanımı kolay doğruluk oranı yüksek bir algoritmadır fakat k değerini hesaplamak için zaman kaybı yaşanabilmektedir.

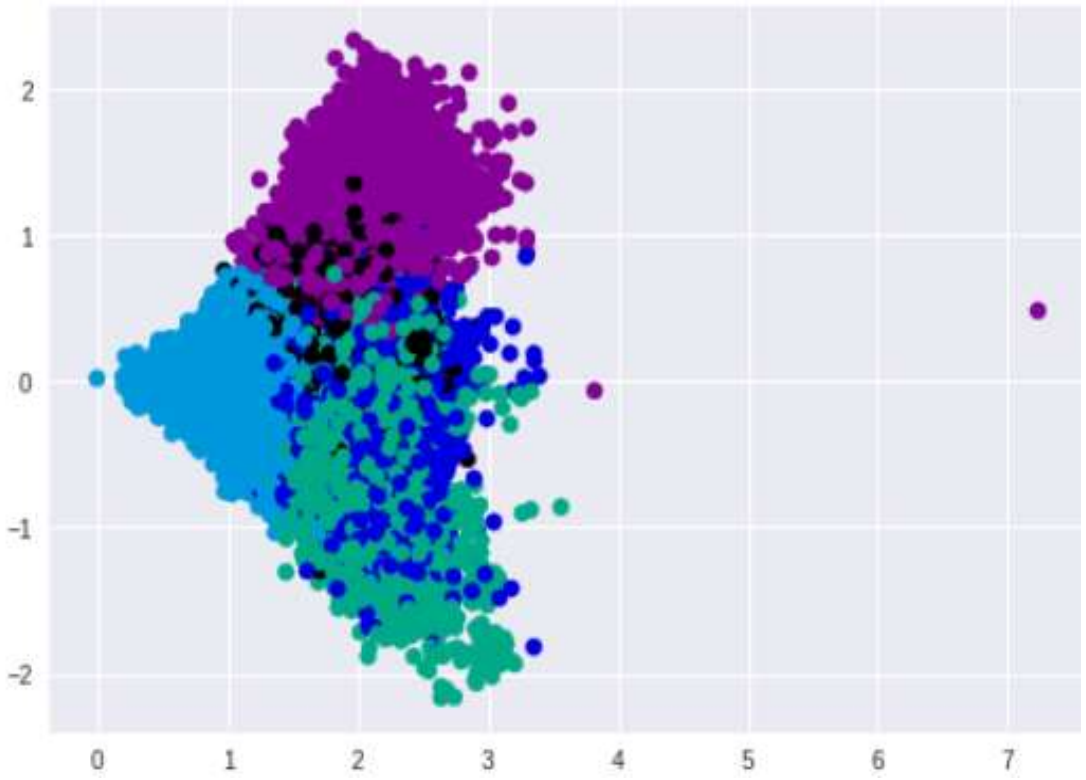
3.2.2. Dbscan Algoritması

Bir kümenin elemanları, üst üste binen komşuluğa sahip çekirdek nesneler kullanılarak oluşturulur ve diğer nesneler gürültü olarak kabul edilir. Bu algoritma, aykırı değerleri doğru bir şekilde kaldırması gerektiğinde ve giriş verilerinin sırası daha karmaşık olduğunda uygulanabilir. Bu algoritma, giriş parametrelerine karşı çok hassastır ve veri kümesinde bulunan yüksek boyutlu uzayları parçalara ayırır. Bu algoritmanın avantaj ve dezavantajları aşağıda tablolaştırılmıştır (Çizelge 3.3).

Çizelge 3.3: Dbscan Avantaj ve Dezavantajları (Alsmadi 2009'dan değiştirilerek alınmıştır.)

Avantajları	Dezavantajları
Dbscan, MinPts kullanarak rastgele şekillendirilmiş kümeleri bulabilir.	Dbscan, büyük veri ile iyi performans gösteremez. Çalışmamızda büyük veri ile çalışmadığımız için Dbscan en iyi performansı gösterecektir.

Bu çalışmada Dbscan ile kümelenmiş veriler renklendirilmiştir ve grafiksel gösterilmiştir (Şekil 3.4).

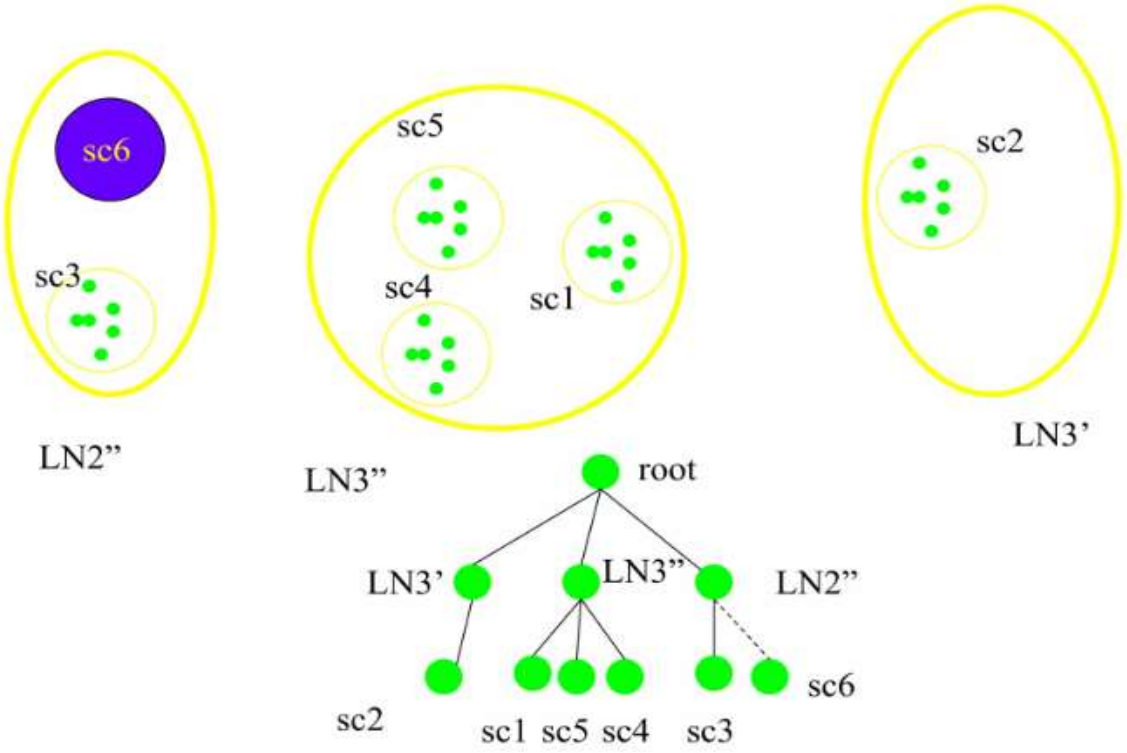


Şekil 3.4 : Dbscan ile kümelenmiş veriler

.

3.2.3. Birch Algoritması

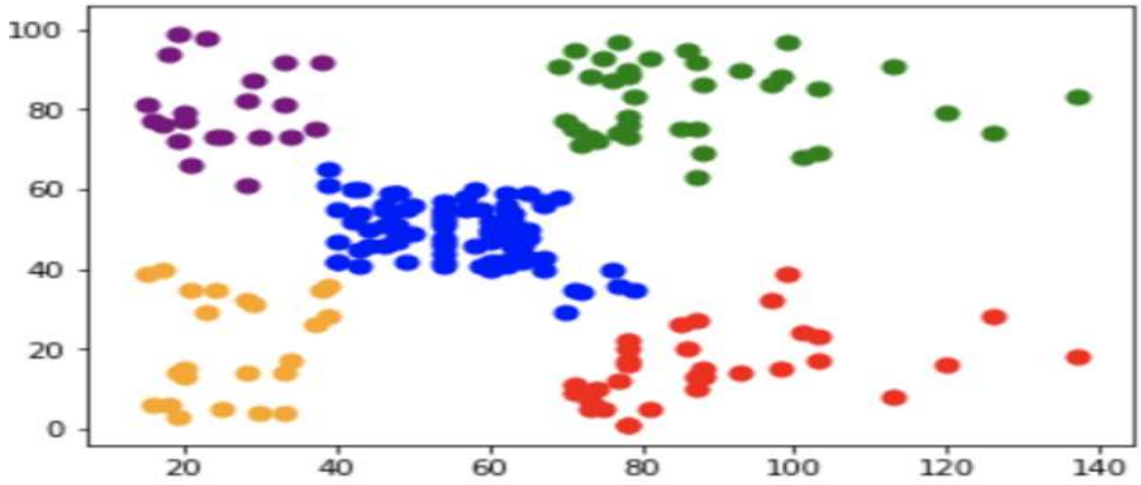
Hiyerarşik Kümeleme alt kümelere ayırma yöntemine dayanır. Örneğin bitkiler kümelere ayırıldığında önce yeşil yapraklılar sonra yeşil yapraklılar isimleri ile alt kümelere ayrılmaktadır. Bu şekilde dendogram grafikleri de oluşturabilmektedir. Avantajları kullanımı kolaydır bir k seçimi yapılmadığı için hızı da gayet iyidir. Alt kümelere ayırarak kümeleme yöntemini göstermektedir. Birch algoritmasının genel yapısı aşağıdaki örnekte şekilsel örneklenmiştir (Şekil 3.5).



Şekil 3.5. Birch Algoritması genel yapısı (Thamizharasi 2013’den değiştirilerek alınmıştır.)

Birch kümeleme algoritmasında, benzer nesneleri küme adı verilen gruplar halinde gruplandırır. İki tür hiyerarşik kümeleme algoritması vardır. Aşağıdan yukarıya yaklaşım: Birçok küçük kümeyle başlar ve daha büyük kümeler oluşturmak için bunları birleştirir. Yukarıdan aşağıya yaklaşım: Daha küçük kümelere bölmek yerine tek bir kümeyle başlar. Gruplama geçmişini görselleştirmek ve optimal küme sayısını bulmak için bir dendrogram kullanılabilir. Diğer kümelerin hiçbiriyle kesişmeyen en büyük dikey mesafeyi belirlenir. Her iki uçta da yatay bir çizgi çizilir. Optimal küme sayısı, yatay çizgiden geçen dikey çizgilerin sayısına eşittir. Aşağıdan yukarıya yaklaşım veya aglomeratif kümeleme olarak da bilinir. Bu kümeleme algoritması, küme sayısını önceden belirlememizi gerektirmez. Aşağıdan yukarıya algoritmalar, başlangıçta her veriyi tek bir küme olarak ele alır ve ardından tüm kümeler, tüm verileri içeren tek bir kümede birleştirilene kadar küme çiftlerini art arda kümeler. Ayrıca Birch algoritması her alt katmanı için hangi kümeye ait olduğunu bulmak için CF değeri hesaplar. CF değerini bulmak için dallanma katsayısı yani ağaçta en fazla kaç veri olacağını belirten değerdir. Eşik değeri ise her bir alt yaprakta çapın en fazla ne kadar olacağını gösteren değerdir. Bu hesaplamalarla Birch algoritması yerleşimlerini yani kümelemelerini yapmaktadır.

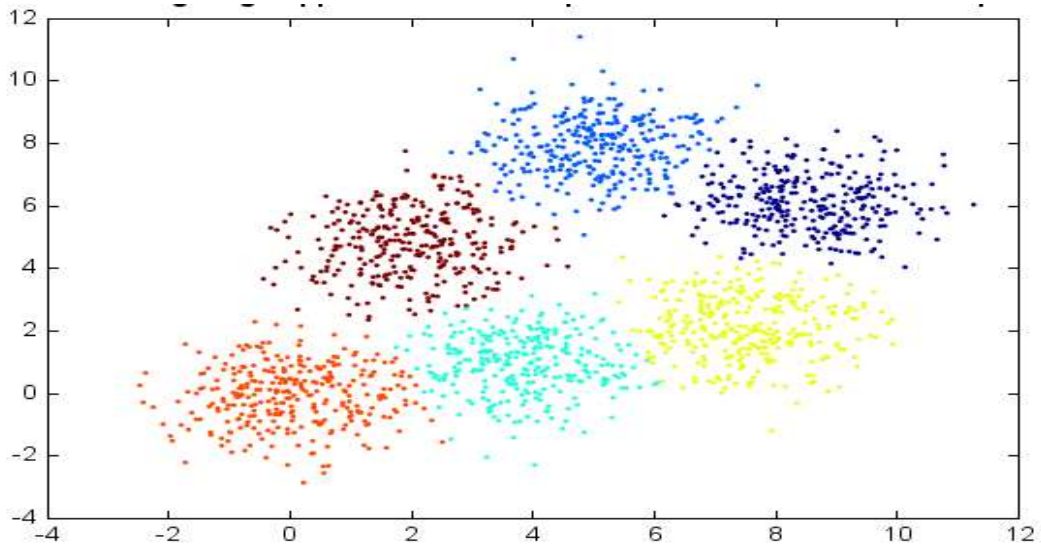
Bu çalışma esnasında veriler yaşa ve yıllık gelire göre Birch algoritması ile Colab ortamında kümelenerek grafiksel olarak gösterilmiştir (Şekil 3.6).



Şekil 3.6 : Birch kümeleme

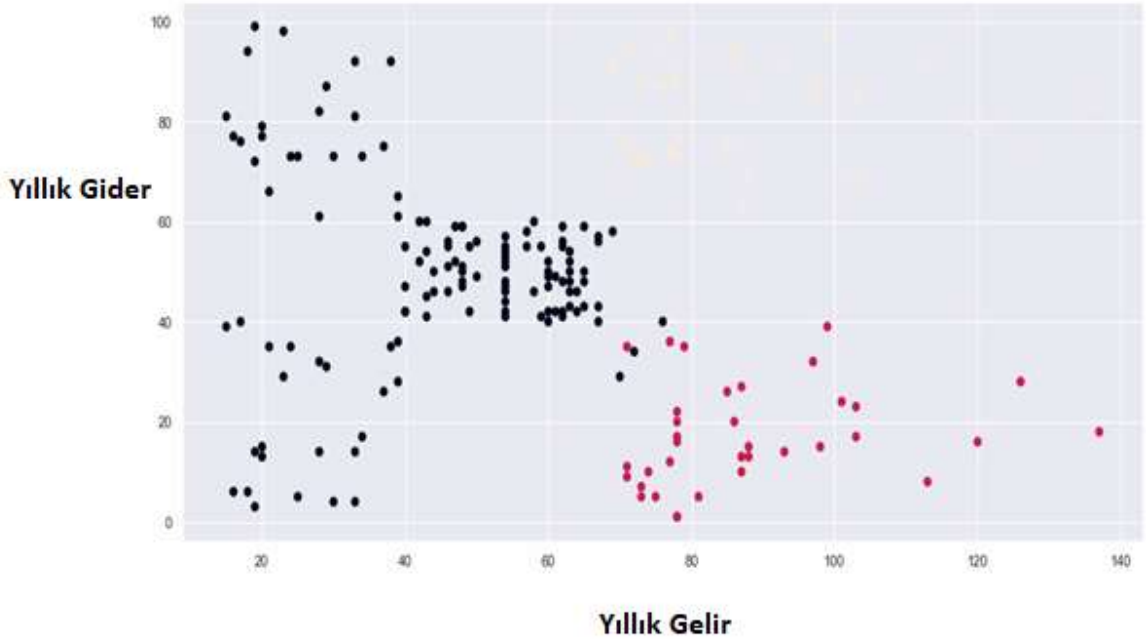
3.2.4.Clique Algoritması

Clique, sting ile birlikte örnekleme tekniklerini kullanmaktadır. Kümeleme işlemi, her düğümün bir k-medoid kümesi olan olası bir çözüm verdiği bir grafiğin aranmasıyla başlar. Bir medoid değiştirildikten sonra elde edilen sonuç, mevcut kümelemenin komşusu olarak bilinmektedir. Clique, bir düğüm seçerek ve onu yerel bir minimum arayan kullanıcı tanımlı sayıda komşuyla karşılaştırarak çalışmaktadır. Mükemmel bir komşu bulunamazsa işlem baştan tekrarlanır; aksi takdirde mevcut kümeleme yerel bir optimum olarak kabul edilmektedir. Yerel optimum belirlendiğinde, rastgele yeni bir düğüm seçilerek başlar ve yeniden yeni bir yerel optimum aranır.



Şekil 3.7: Clique ile kümelenmiş veriler (Fox 2017)

Clique algoritması ile Enron veri setindeki kullanıcılar yıllık gelir ve yıllık giderlerine göre kümelendiğinde oluşan grafik sonucu aşağıda verilmiştir (Şekil 3.8).



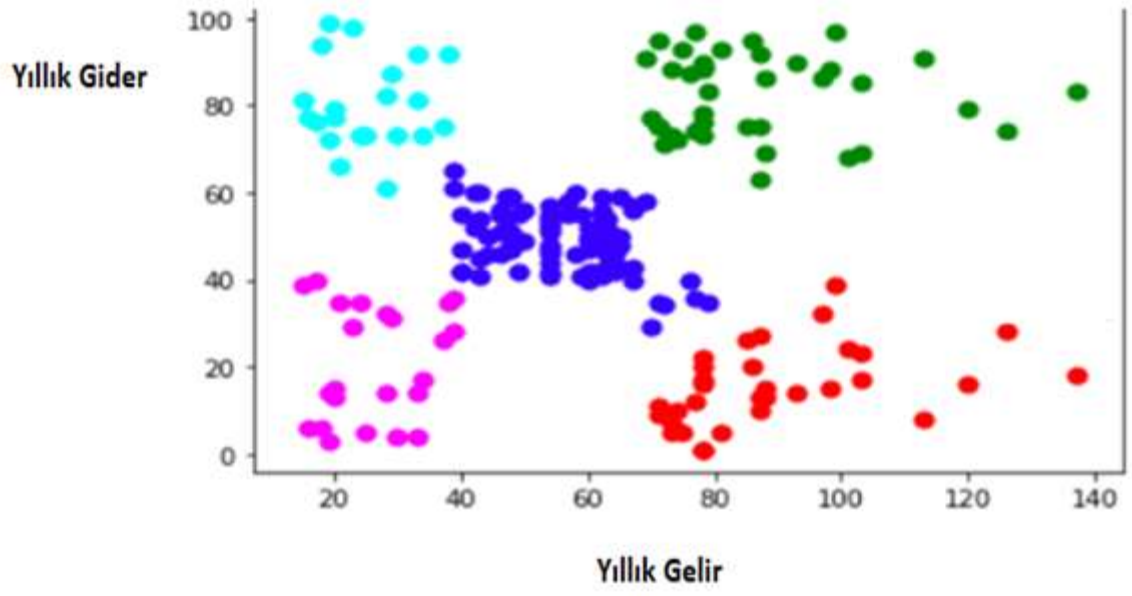
Şekil 3.8 : Clique ile yıllık gelir ve yıllık gidere göre kümeleme

Bir olgu hakkında daha derin bir anlayışa sahip olmak istediğinde, bunları gruplara ayırarak gruplandırma kümeleme yaklaşımlarından biridir. İlk olarak, mevcut medoidler olarak veri kümesindeki k nesneyi rastgele seçmektedir. Daha sonra rasgele bir mevcut orta boy x ve mevcut

orta boylardan biri olmayan bir nesne y seçmektedir. Seçilen x ve y sonucu Clique kümeleme algoritması gerçekleşmektedir. İstenilen kümeleme yapılmış olmaktadır.

3.2.5.Cobweb Algoritması

Cobweb, model bazlı kümelemede artımlı sistem kavramını kullanmaktadır. Her bir nodül, olasılık yaklaşımı kullanan bir sınıfı temsil eder ve nesne sınıflandırmasını ve nodüldeki her bir niteliğe karşılık gelen değeri özetler. Bu ağaç oluşumu, yeni nesnenin yanlış yerleştirilmiş niteliklerini ve değerlerini tanımlamaya yardımcı olmaktadır. Bu teknik, ağacın yapısını ve verimliliğini gözlemlemek için sınıf verimliliği olarak da adlandırılan sorgulayıcı tahmin ölçüsünü kullanmaktadır. Cobweb, kavramsal öğrenmenin popüler basit bir yöntemidir. Sınıflandırma ağacı şeklinde hiyerarşik bir kümeleme oluşturur. Her düğüm bir kavrama atıfta bulunur ve bu kavramın olasılıksal bir tanımını içerir. Cobweb algoritması ile Enron veri setindeki kullanıcılar yıllık gelir ve yıllık giderlerine göre kümelendiğinde oluşan grafik sonucu aşağıda verilmiştir (Şekil 3.9).



Şekil 3.9 : Cobweb ile kümelenmiş veriler

4. BULGULAR

Bu bölümde, kodlanan ve çalıştırılan beş farklı kümeleme algoritmasına ait bilgiler ve elde edilen sonuçlar sunulmuştur. Gerçekleştirilen karşılaştırmaların amacı hangi kümeleme algoritmasının performansının daha iyi olduğunu ve hangi durumlarda tercih edilmesi gerektiğini göstermektedir. Kümeleme algoritmaları çalışmaları grafiksel Sonuçları ve Performansları görüntülenecektir. Bilinen çeşitli kümeleme algoritmaları, uygulamaları, avantajları ve dezavantajları incelenmiştir. Ayrıca Enron mail veri seti kullanılarak bazı kümeleme algoritmalarının GPU ve CPU'da performans değerlendirmesine odaklanılmıştır (Çizelge 4.1, Çizelge 4.2).

Çizelge 4.1 : Enron veri seti verileri görüntüleme

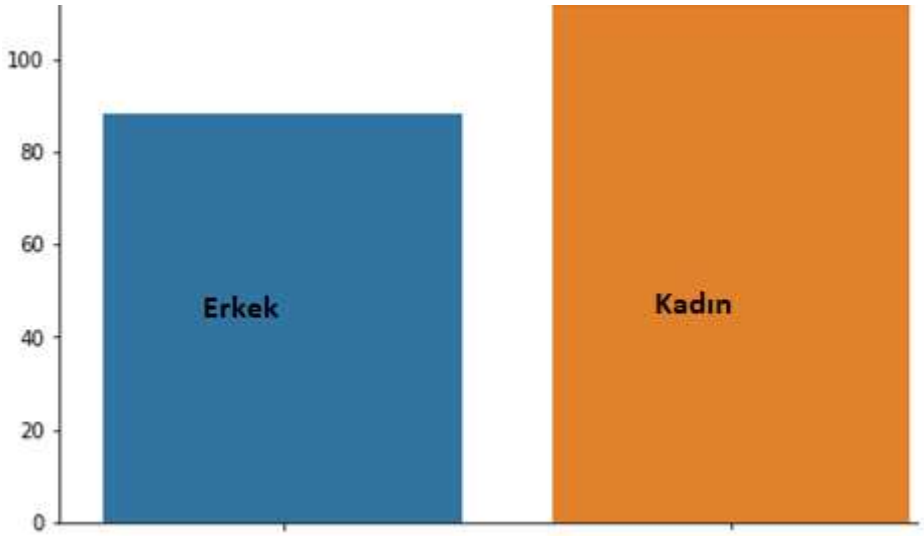
Müşteri No	**Cinsiyet**	**Yaş**	**Yıllık Gelir $-Sapma**	**Yıllık Gider$-Sapma**
1	Kadın	24	22	23
2	Erkek	26	25	34
3	Erkek	29	34	45
4	Kadın	22	42	56
5	Kadın	31	12	23
6	Erkek	34	56	25
7	Kadın	23	34	37
8	Kadın	25	67	56
9	Kadın	27	56	67
10	Erkek	23	23	34

Çizelge 4.2 : Enron veri seti verileri görüntüleme

Müşteri No	**Cinsiyet**	**Yaş**	**Yıllık Gelir $**	**Yıllık Gider$**
1	Kadın	24	15	39
2	Erkek	26	13	81
3	Kadın	29	23	98
4	Erkek	22	34	23
5	Kadın	31	45	44
6	Erkek	34	56	25
7	Kadın	23	34	37
8	Kadın	25	67	56
9	Kadın	27	56	67
10	Erkek	23	23	34

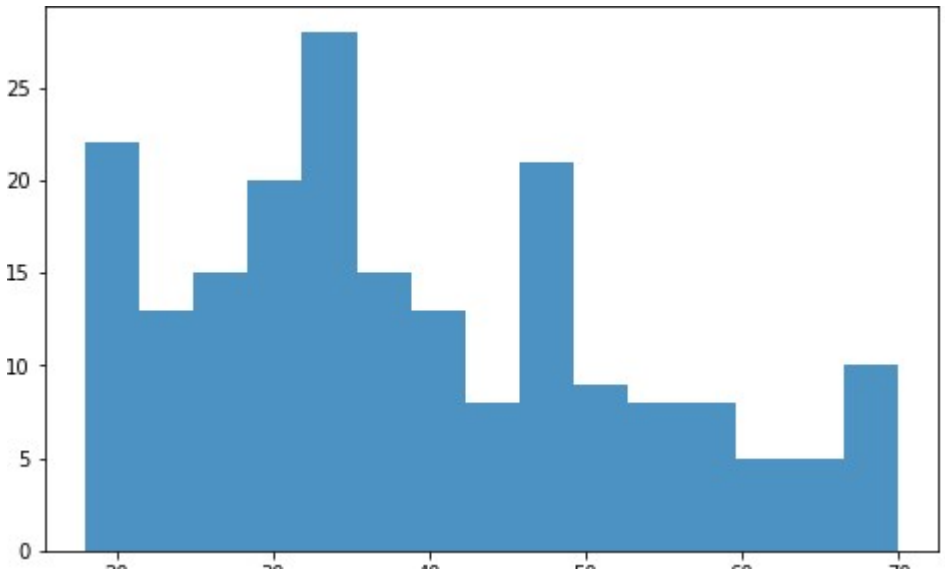
Cinsiyet:

Enron veri setindeki cinsiyet dağılımını göstermek için bir barplot ve bir pasta grafiği oluşturulmuştur (Şekil 4.1).



Şekil 4.1 : Enron veri seti cinsiyet verileri görüntüleme

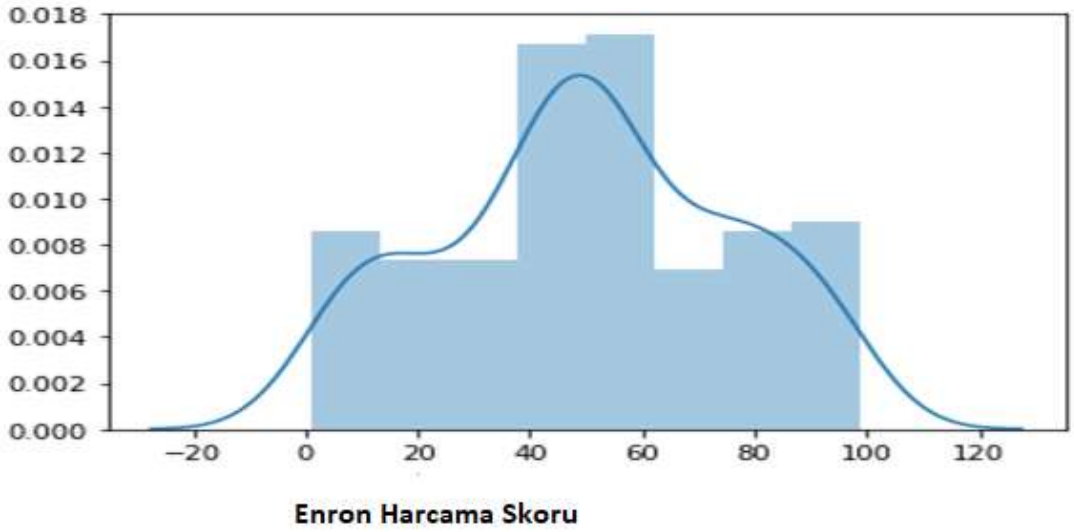
Daha sonra, yaşa göre analiz edileceği için bir histogram çizilmiştir. Öncelikle cinsiyet değişkenin özetini alarak ilerlenmiştir ve daha sonra yaş kriterine göre kümeleme gerçekleştirilmiştir.



Şekil 4.2 : Enron veri seti yaş verileri dağılımı

Yukarıdaki grafiklerden, mail kullanıcıların çoğunun 30 ila 35 yaşları arasında olduğu, ayrıca minimum kullanıcı yaşının 18, maksimum yaşın ise 70 olduğu sonucuna varılmıştır. Bir histogram ve bir yoğunluk grafiği kullanarak yıllık gelirleri görüntülenmiştir (Şekil 4.2).

Yukarıdaki grafiklerden, kullanıcıların minimum yıllık gelirinin 15, maksimum gelirin 137 olduğunu açıkça görülmektedir. Ortalama geliri 70 olan kişiler, histogram dağılımımızda en yüksek frekansa sahiptir. Tüm kullanıcıların ortalama geliri 60,56'dır. Enron kullanıcılarının yıllık gelirinin normal bir dağılıma sahip olduğu gözlemlenmektedir. (Şekil 4.2) Kullanıcıların yıllık gideri aynı şekilde yıllık harcama miktarları da kümelenmiştir.

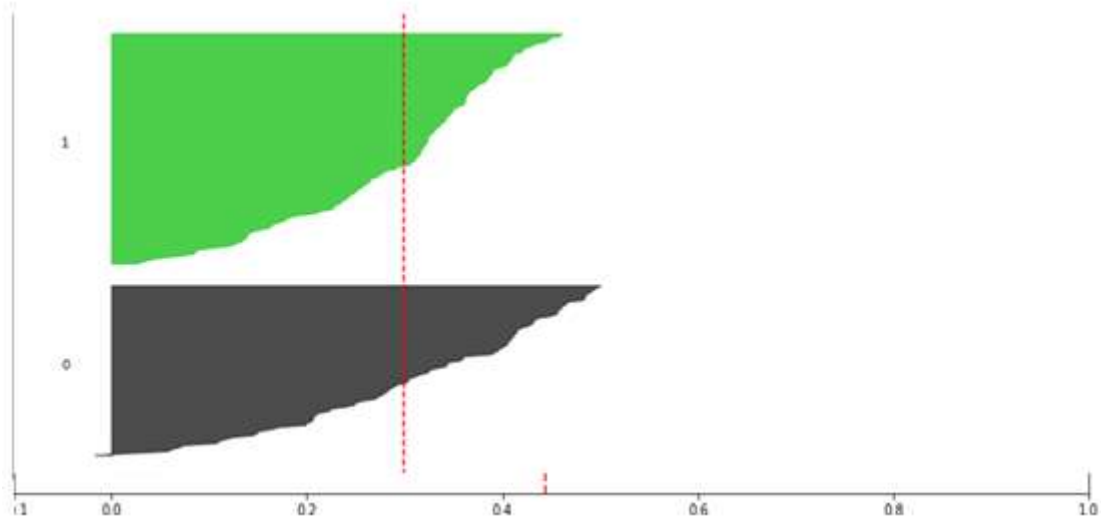


Şekil 4.3 : Enron veri seti yıllık gider harcama verileri grafiği

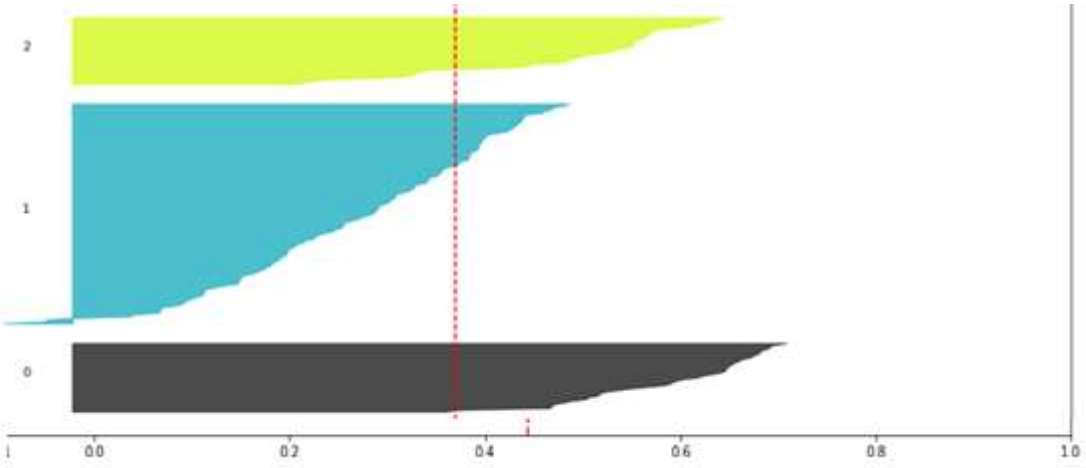
Minimum harcama puanı 1, maksimum harcama puanı 99 ve ortalama 50.20'dir. Dağılım grafiğinden, 40 ile 50 arasında bir harcama puanına sahip olan müşteri sınıfının tüm sınıflar arasında en yüksek frekansa sahip olduğu sonucuna varılmaktadır (Şekil 4.3) .
Veri setimizin her bir özelliğini keşfettikten sonra, kullanıcılar özelliklerine göre segmentlere ayrılmıştır. Öncelikle K-Means kümeleme algoritması ile çalışılmıştır. Bu algoritma, veri kümesinden, kümelerimiz için merkezler olarak bilinen ilk merkezler olarak hizmet edecek rastgele k nesneyi seçerek başlar. Daha sonra her adımda, algoritma her küme içindeki bireyler arasındaki mesafe olan iç mesafeyi en aza indirmeye ve kümeler arasındaki mesafe olan ara mesafeyi en üst düzeye çıkarmaya çalışmaktadır.

Optimum Küme Sayısının Belirlenmesi: K-Means algoritmasında ilk adım olan k sayısının belirlenmesi adımı için Siluet Yöntemi uygulanmıştır.

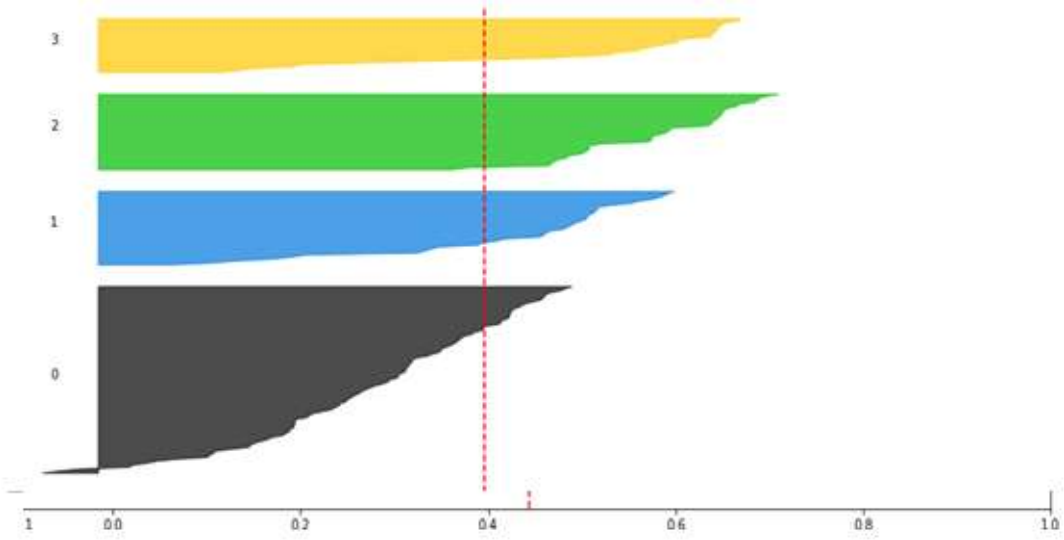
Ortalama Siluet Yöntemi: Ortalama siluet yöntemi yardımıyla kümeleme işleminin kalitesi ölçülebilmektedir. Bununla, veri nesnesinin küme içinde ne kadar iyi olduğu belirlenebilmektedir. Yüksek bir ortalama siluet genişliği elde edilirse, iyi bir kümeleme yapılmış olmaktadır. Ortalama siluet yöntemi, farklı k değerleri için siluet gözlemlerinin ortalamasını hesaplamaktadır. Optimal sayıda k küme ile, k küme için önemli değerler üzerinden ortalama siluet maksimuma çıkarılabilmektedir.



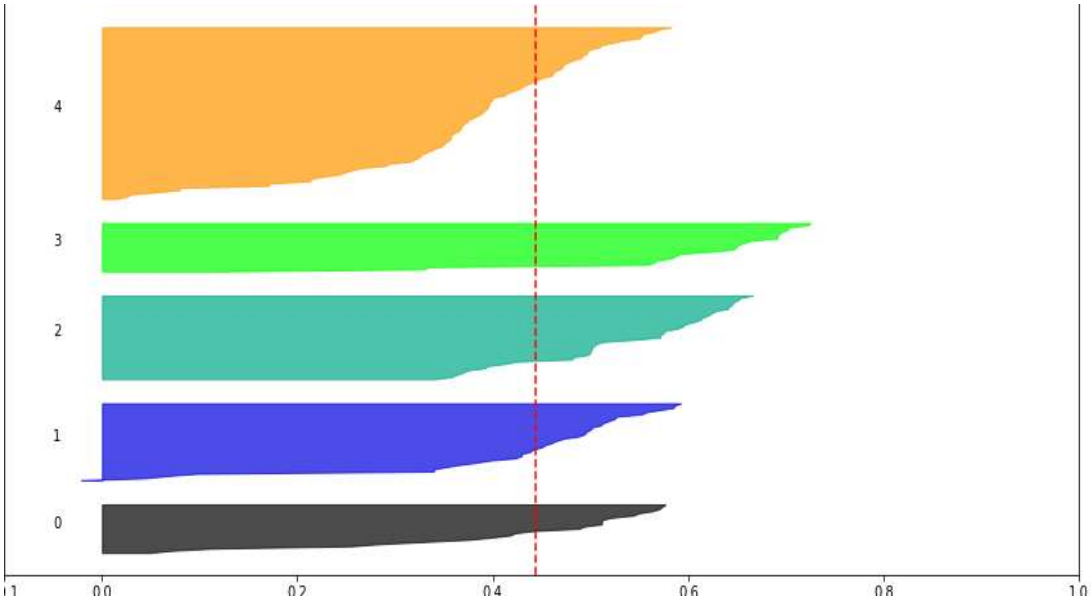
Şekil 4.4 : K=2 Değeri için siluet değeri hesaplama



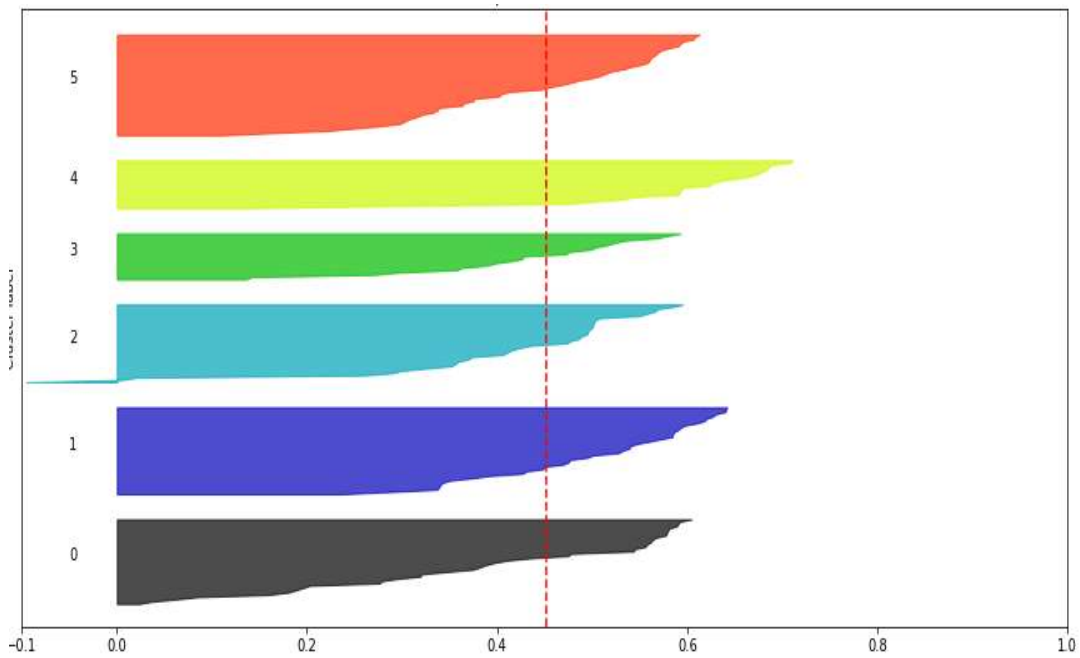
Şekil 4.5 : K=3 Değeri için siluet değeri hesaplama



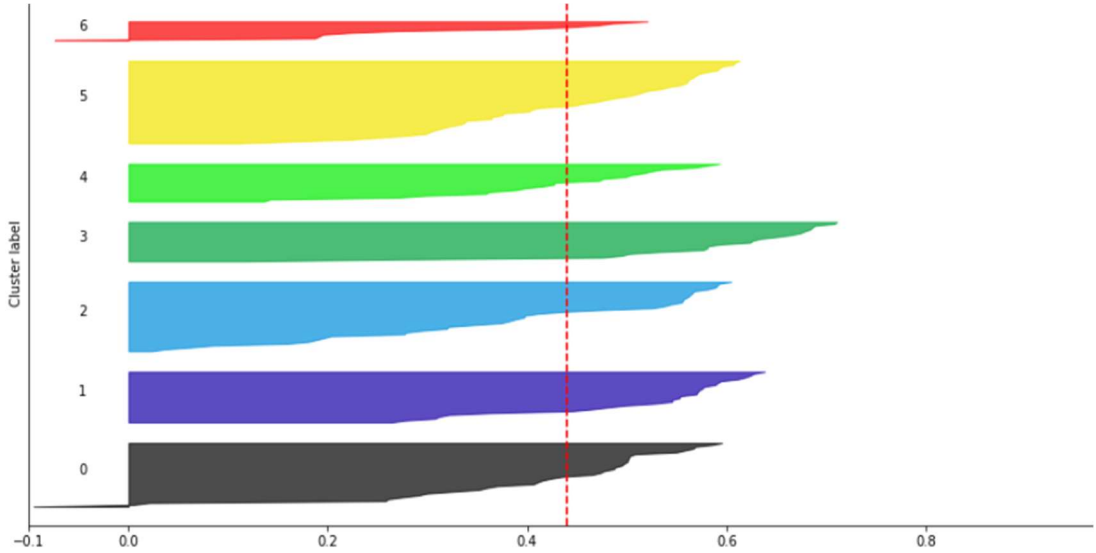
Şekil 4.6 : K=4 Değeri için siluet değeri hesaplama



Şekil 4.7 : K=5 Değeri için siluet değeri hesaplama

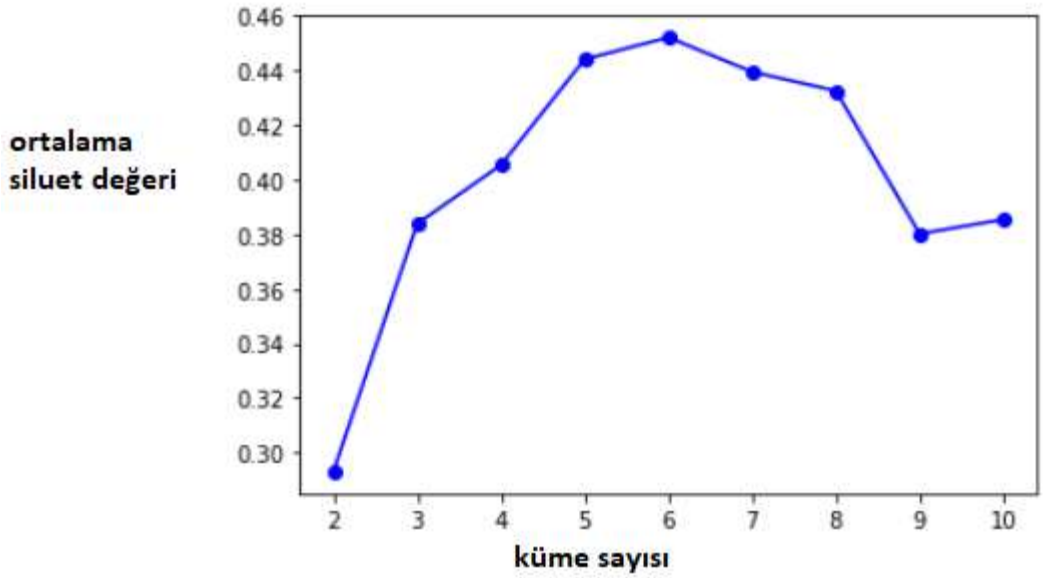


Şekil 4.8 : K=6 Değeri için siluet değeri hesaplama



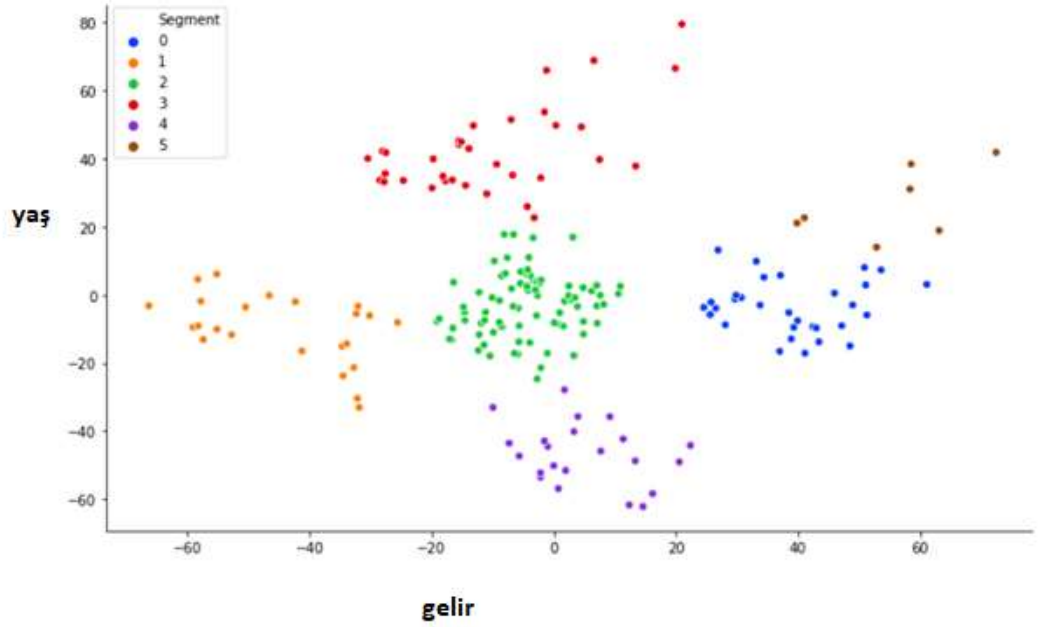
Şekil 4.9 : K=7 Değeri için siluet değeri hesaplama

K=10 değeri hesaplanmıştır.Algoritma verilerinin Şekil 4.4, Şekil 4.5, Sekil 4.6, Şekil 4.7, Şekil 4.8, Şekil 4.9 sonuçlarına göre ortalama siluet değeri grafiği çizilmiştir. Ortalama Siluet değeri bu şekilde görüntülenmiştir (Şekil 4.10). Siluet yöntemi ile en uygun K değerini belirlenmiştir.



Şekil 4.10 : Uygun K değeri hesaplama siluet yöntemi

Kullanıcılar için segmentasyon görevi için en uygun küme sayısı k, ortalama siluet genişliği 0,45 olan 6'dır. K=6 seçilmiştir ve K-Means kümeleme algoritması uygulanmıştır.



Şekil 4.11 : Gelir ve yaşa göre kullanıcıları kümeleme

K-Means algoritması kullanılarak Enron veri seti kullanıcıları gelir ve yaşa göre kümelenmiştir.(Şekil 4.11)

- Küme 0 ve 5 – Yüksek gelirli orta yaştaki kullanıcıları temsil etmektedir.
- Küme 1 – Yüksek yaş ve düşük gelir sahip müşterileri temsil etmektedir.
- Küme 2 – Orta yaş ve orta gelir puanına sahip müşterilerden oluşmaktadır.
- Küme 3 – Yüksek yaş ve orta yıllık gelir harcamasına sahip müşterilerden oluşmaktadır.
- Küme 4 – Geliri orta, yaşı küçük olan müşterilerden oluşmaktadır.

Veri setinin algoritmalara göre çalışma zamanı hesaplanmıştır ve veri sayısı belirtilmiştir.(Çizelge 4.3)

Çizelge 4.3: Kümeleme algoritma deneysel sonuçlar

S.N	**Kümeleme Algoritmaları**	**Veri Adedi**	**Zaman (Dakika)**
1	Birch	8124	6.27
2	**Dbscan**	8124	1.1224
3	Cobweb (splits=87;merges= 90)	8124	7.92
4	K-Means clustering	8124	4.567
5	Clique	8124	6.75

Bu verilerin tamamı Google Colab üzerinde denenerek hesaplanmıştır.

Çizelge 4.4 : Kümeleme performans deneysel sonuçlar

S.No	**Kümeleme Algoritmaları**	**Accuracy Doğruluk**	**Recall Hassasiyet**	**Precision**	**F-Measure**
1	Birch	0.7623	0.5876	0.7654	0.664818
2	Clique	0.7815	0.5847	0.6872	0.63182
3	**Dbscan**	**0.9675**	**0.8287**	**0.9132**	**0.8689**
6	Cobweb	0.8765	0.7756	0.6156	0.814135
7	K-Means	0.9134	0.7862	0.7245	0.842377

Elde edilen sonuçlar, Dbscan algoritmasının diğer algoritmalarla karşılaştırıldığında tüm metrikler açısından iyi performans gösterdiğini göstermektedir. Enron kullanıcı veri seti üzerinde oluşturulan küme sayısı ve alınan süre bazında çeşitli kümeleme

algoritmaları arasında karşılaştırma yapılmıştır. Verilen veri kümesi için, Dbscan algoritmasının kümeleme gerçekleştirmesi daha az zaman alırken, Clique algoritması çok daha fazla zaman almıştır. Kümelenmiş örnekler durumunda, Dbsan algoritması daha fazla sayıda küme oluştururken, Cobweb küme algoritması daha az sayıda küme oluşturmuştur.

Çalışmada kullanılan diğer bir yöntem yukarıda kullandığımız beş kümeleme algoritmasını aynı şekilde GPU ortamında çalıştırmak ve sonuçları CPU ortamındaki sonuçlarla karşılaştırmaktadır. GPU ortamı için aynı şekilde Google Colapse kullanılmıştır. Deneysel çalışmalar esnasında kullanılan CPU'nun modeli Intel Xeon Gold 6126 dır. GPU modeli ise Tesla T4 k80' dir. Tesla T4 bir adet CPU çekirdeğine sahipken 2560 GPU çekirdeğine sahiptir. Bunun yanında GPU'lar , CPU'lara göre çok fazla sayıda ALU(Arithmetic Logic Unit) bulundururlar.

Çizelge 4.5: CPU ve GPU fiziksel özellikleri

	İşlemci Mimarisi	**CPU Clock**	**CPU Core**	**CUDA Core**	**GPU Memory**
CPU	Intel Xeon Gold 6126	2.3GHz	1	-	-
GPU	**Tesla T4 k80**	2.2GHz	1	2560	16 GB

Yapılan çalışmada Google Colab üzerinde GPU seçilerek tüm kümeleme algoritmaları tekrar çalıştırılmıştır ve matematiksel hesaplamalar yapılmıştır (Çizelge 4.5).

Bu çalışma boyunca amaç, seçilen kümeleme algoritmalarının kümeleme performanslarını yürütme sürelerine göre karşılaştırmaktır.

CPU ve GPU ortamlarında mail kümeleme algoritmalarının performanslarını kıyasladığımızda aynı veri seti ve veri adedi için performans farklılıkları oluşmaktadır. Fakat sonuç olarak her işlemde olduğu gibi GPU ‘ya sahip makinelerin kümeleme işlemini daha hızlı yaptığı kanısı ortaya çıkmaktadır. GPU'ların paralelleştirme kapasiteleri CPU’lardan daha yüksektir, çünkü GPU’lar Merkezi İşlem Birimlerinden (CPU'lar) çok daha fazla çekirdeğe sahiptir. Bu çalışmada CPU ve GPU arasında kıyaslama testleri yapılmıştır. Testlerde Tesla k80 GPU ve Intel Xeon Gold 6126 CPU kullanılmıştır. Testler için ihtiyaç duyulan donanım miktarının yüksek olması nedeniyle testlerde bulut üzerinde çalışan CPU ve GPU’lar kullanılmıştır. Testler sırasında bazı hiper parametreler ayarlanmış ve CPU ile GPU arasında performans değerleri karşılaştırılmıştır. Yapılan tüm testlerde GPU'nun CPU'dan daha hızlı çalıştığı gözlemlenmiştir. Bazı durumlarda CPU sunucusu ve CPU sunucusu üzerinde yapılan testlere göre GPU, CPU'dan 4-5 kat daha hızlıdır. Bu değerler, daha fazla özelliğe sahip bir GPU sunucusu kullanılarak daha da artırılabilir. Aşağıda CPU ve GPU ortamlarındaki test sonuçları birlikte tablosal gösterilmiştir (Çizelge 4.6).

Çizelge 4.6.: GPU ve CPU Kümeleme performans deneysel sonuçlar Karşılaştırma

S.N	**Kümeleme Algoritmaları**	**Veri Adedi**	**Kümeleme Zaman CPU (Dakika)**	**Kümeleme Zaman GPU (Dakika)**
1	Birch	8124	6.27	3.27
2	**Dbscan**	8124	1.1224	0.9
3	Cobweb (splits=87;merges=90)	8124	7.92	3.52
4	K-Means	8124	4.567	2.23
5	Clique	8124	6.75	3.34

5 . TARTIŞMA VE SONUÇ

Bu tezde beş kümeleme yöntemi, K-Means, Birch, Dbscan, Clique, Cobweb algoritmaları analiz edilmiştir. Ayrıca algoritmaların zayıf ve güçlü noktaları tartışılmıştır. Her bir algoritma Python kullanılarak kodlanmış ve sunum için okuyucunun farklılıkları görebilmesi için aynı örnek veriler kullanılmıştır. Kümeleme algoritması seçimi amaçlara bağlıdır, çünkü farklı kümeleme yöntemleri, farklı sonuçlara yol açmaktadır. Hiçbir yöntem üstün olarak kabul edilemez, ancak bazı yöntemler için daha uygundur denilebilmektedir. Çalışmada 8124 veri kümelenmeye çalışılmıştır. Kullanılan K-Means, Clique, Birch kümeleme, Dbscan, Cobweb algoritmalarıdr. İlk olarak K-Means algoritması ile Enron veriseti kullanıcıları yaş ve yıllık gelirlerine göre kümelenmiştir. Kümeleme esnasında küme içi benzerlikler maksimum, kümeler arası benzerlikler ise minimum olacak şekilde düzenlenmiştir. Beş algoritmada CPU ve GPU ortamlarında defalarca çalıştırılmıştır. Çizelge 2.10'u incelediğimizde en hızlı performansa sahip algoritma Dbscan algoritmasıdır, özellikle GPU ortamında çalıştırıldığında çok hızlı bir şekilde 0.9 dakikada kümeleme işlemini bitirmiştir. CPU ortamında çalıştırdığımızda 1.1224 dakika ile diğer algoritmalardan daha hızlı kümeleme yapmıştır. İkinci olarak K-Means algoritmamız gelmektedir. GPU ortamında 2.23 dakika, CPU ortamında 4.567 dakikada, 8124 verimizi yaş ve yıllık geliri baz alarak kümelemiştir. Üçüncü sırada Birch kümeleme algoritması GPU ortamında, CPU ortamına göre iki kat hızla çalışmıştır. Dördüncü sırada Clique algoritmamız ve beşinci sırada da Cobweb algoritmamız gelmektedir. Dbscan algoritmasının en hızlı çalışmış olmasının sebebi, algoritmanın komşuları ile olan mesafelerini hesaplarken önceden belirlenmiş eşik değerinden daha fazla olan alanları gruplandırarak kümeleme işlemini gerçekleştiriyor olmasıdır. Cobweb' in çalışmamızda yavaş kalmasının sebebi ise bu algoritmanın en iyi kümeyi bulana kadar karar verme aşamasında nesneyi geçici olarak bütün kümelere önce koyması, bütün kümeler için uygunluk değeri CU hesaplaması ve sonunda en büyük CU değerine sahip kümeye nesneyi yerleştirmesidir. Bu sebeple diğer algoritmalarımıza kıyasla yavaş kalmıştır. Ayrıca veri sayısının çok fazla olması durumunda küme sayısı çok fazla artmaktadır.

Sonuç olarak kümeleme yapılacağı zaman eldeki verinin ve problemin çok iyi analiz edilmesinin önemi ortaya çıkmıştır. Çünkü bütün algoritmalar farklı amaçlara daha iyi

hizmet etmektedir. Bu çalışmadaki beş algoritma performans hızına göre CPU ve GPU ortamlarında sıralanmıştır. CPU ve GPU ortamları kıyaslandığında GPU ortamında kümeleme yapmanın açık ara farkla önde olduğu gözlemlenmiştir. Fakat yapılan işin boyutuna GPU kullanmanın ekonomik boyutuna göre analizler yapılarak seçim yapılmalıdır. Büyük veriler için GPU ortamları kullanışlı olduğu için çok daha fazla tercih edilmektedir. GPU ortamında CPU ortamına göre ciddi hız farkı olduğu ve bizim veri setimize göre en hızlı algoritmanın Dbscan algoritması olduğu kanısına varılmıştır.

KAYNAKLAR

A. Z. Kamil, H. Şen, Ş. Kasalı, , Mayıs (2012). İç içe kümeleme , Bilgi Güvenliği Akademisi

Akın, K. Y. (2008).” Veri Madenciliğinde Kümeleme Algoritmaları ve Kümeleme Analizi ”. Doktora Tezi, Marmara Üniversitesi, Sosyal Bilimler Enstitüsü, İstanbul, 164s.

Abdalgader, K. (2017). Clustering Short Text using a CentroidBased Lexical Clustering Algorithm. IAENG International Journal of Computer Science, 44(4).

Akgündüz, G.Ö. 2019. Platon'un Timaios diyalogunda akıl-zorunluluk ilişkisi. *Beytulhikme: An International Journal of Philosophy*, 9(3): 625-650.

Anıl Guven (December 2019), K-Means Kümeleme Elbow Yöntemi Bilgi Güvenliği Akademisi, 2019

Bagirov, A.M., Ugon, J. and Webb, D., 2011. Fast modified global k-ortalama algorithm for incremental cluster SinceDirect,36(2), 451-461. Pattern Recognition,

Mesut, A. (2006). Veri Sıkıştırmada Yeni Yöntemler. Trakya Üniversitesi, Fen Bilimleri Enstitüsü, Doktora Tezi.

Brecheisen S, Kriegel HP, Pfeifle M. “Parallel density-based clustering of complex *objects. Pacific-Asia Conference on Knowledge Discovery and Data Mining”. 2006.*

Chang, D. and Xian, W., 2009. A genetic algorithm with gene rearrangement for k-ortalama clustering. Pattern Recognition, IEEE digital library, 42(7), 1210-1222.

Chen, J., Li, D. and Shen, H., 2009. A Fast k-ortalama Clustering Algorithm Based on Grid Data Reduction, IEEE digital library, 9980042(1095-323X), 1 – 6.

Çentik, Güven. Makine Öğrenmesi Yöntemlerinin Polisomnografik Verilere Uyarlanması, Yüksek Lisans Tez, Trakya Üniversitesi Fen Bilimleri Enstitüsü, 2013.

E. B. Brusco, The Emilion Model, , Nisan (2008). "Clustering Mail Systems", Cambridge Journal Economics.

Eswara Reddy, B., Viswanath , P. and Hitendra Sarma, T., 2012. A hybrid approach to speed-up the k-ortalama clustering method, Springer-Verlag , 4(2), 107-117.

Forsman, (2012). "Cluster and Cosuputional analysis", JIBS Literureture review, 25 (3), 229–236

Güncel SARIMAN (2014) Veri Madenciliğinde Kümeleme Teknikleri Üzerine Bir Çalışma: K-Means ve K-Medoids Kümeleme Algoritmalarının Karşılaştırılması . Doktora Tezi, Marmara Üniversitesi, Fen Bilimler Enstitüsü, İstanbul, 170s.

H.Önal, (2015). E-posta Başlıklarından Bilgi Toplama", Bilgi Güvenliği Akademisi

Durrani (2013) . Dbscan ve K-means kümeleme algoritmalarının bir HPC kümesi üzerinde paralelleştirilmesi. Yüksek Lisans Tez, Trakya Üniversitesi Fen Bilimleri Enstitüsü, 2015.

Karacan(2014), Bölümleyici kümeleme algoritmalarının farklı veri yoğunluklarında karşılaştırılması , Yüksek Lisans Tez, Balıkesir Üniversitesi Fen Bilimleri Enstitüsü, 2016.

I. Alsmadi, I. Alhami, (2009). Clustering and classification of email contents, Journal of King Saud University – Computer and Information Sciences, 27 (1), 46–57

Jain, A.K. Murty, M.N. Flynn, P.J. (1999). Data Clustering: A Review, ACM Computing Surveys. 3, 31.

Kaya (2018), Bulanık kümeleme analizinde bulanık kümeleme algoritmalarının karşılaştırılması, Yüksek Lisans Tez, Ege Üniversitesi Fen Bilimleri Enstitüsü, 2015.

K.Yıldız, Y.Çamurcu, B.Doğan (2010) . Veri Madenciliğinde Temel Bileşenler Analizi ve Negatifsiz Matris Çarpanlarına Ayırma Tekniklerinin Karşılaştırmalı Analizi, 22, 56-77. Erişim adresi: https://ab.org.tr/ab10/kitap/yildiz_camurcu_AB10.pdf

Maskell, (2015). Localied learning clustering aproach , Cambridge Journal Economics. Menzell, E. A. Fox, (2017). Cluster Life Cycle, Jena Economic Research papers

Kendall, M.G.(1966). Discrimination and Classification. In Multivariate Analysis (P.R. Krishnaiah, ed.), Academic Press, Inc., New York, pp. 165-185.

N.Valarmathy, S.Krishnaveni (April,2019) . Performance Evaluation and Comparison of Clustering Algorithms used in Educational Data Mining, International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-7, Issue-6S5

Pallavi, G Sunila (2018). Karşılaştırmalı bir kümeleme analizi K-ortalama Algoritmaları, Hiyerarşik Kümeleme algoritmaları Enron, 22, 56-77. Erişim adresi: https://dergipark.org.tr/tr/download/article-file/751785

Revathi et.al. (2016). Performans Karşılaştırması Çeşitli Kümeleme Algoritmaları , 22, 56-77. Erişim adresi: https://dergipark.org.tr/en/download/article-file/1751876

Hacıoğlu H., K. (2016). Kümeleme Analizinde Kullanılan Bazı Benzerlik İndekslerinin Karşılaştırılması. Yüksek Lisans Tezi Gazi Üniversitesi, Fen Bilimleri Enstitüsü.,98.

Dinçer, Ş. E. (2006). Veri madenciliğinde K-means algoritması ve tıp alanında uygulanması (Master's thesis, Kocaeli Universitesi, Fen Bilimleri Enstitusu)

Sackson, M. (1990). The Use of Cluster Analysis for Business Game Performance Developments in Business, Simulation & Experiential Exercises, 17, pp. 150-154.

Sayılan (2019). Veri madenciliğinde bazı kümeleme algoritmalarının karşılaştırılması, , 22, 56-77. Erişim adresi: https://dergipark.org.tr/tr/download/article-file/193944

Talaván, (2006). A simple, configurable SMTP anti-spam filter: Greylists, Computers & Security, 25 (3), 229–236

T.Bilgin, Y. Çamurcu (2005). Dbscan, Optics ve K-Means Kümeleme Algoritmalarının Uygulamalı Karşılaştırılması, 10, 80-94. Erişim adresi: https://dergipark.org.tr/tr/download/article-file/384439

W. B. Frakes, R. Baeza-Yates (2017). Information Retrieval: Data Structures & Algorithms, Prentice-Hall, Inc., Upper Saddle River, NJ, USA

Rençber, A. (2019). Veri Madenciliğinde Kullanılan Kümeleme Algoritmalarının Karşılaştırılması Üzerine Bir İnceleme ,10, 80-94. Erişim adresi: https://dergipark.org.tr/tr/pub/dubited/issue/46290/551531

M.Bilgin (2017). Gerçek Veri Setlerinde Klasik Makine Öğrenmesi Yöntemlerinin Performans Analizi, 22, 56-77. Erişim adresi: https://ab.org.tr/ab17/bildiri/101.pdf

Silahtaroğlu, G. (2016). Veri madenciliği: Kavram ve algoritmaları. Papatya.

Dinçer, Ş. E. (2006). Veri madenciliğinde K-means algoritması ve tıp alanında uygulanması (Master's thesis, Kocaeli Universitesi, Fen Bilimleri Enstitusu)

Halkidi, M., Batistakis, Y., Vazirgiannis M.,(2001) . On Clustering Validation Techniques", Journal of Intelligent Information Systems, 17:2/3, 107–145, Kluwer Academic Publishers,

Hartigan JA. (1975). Clustering Algorithms, NewYork; John Willey&Sons , Erişim adresi: https://dergipark.org.tr/en/download/article-file/175187996

Çelik Ş.(1998). Classification of Provinces in Turkey according to Health Indıcators by Cluster Analysis, Doğuş Üniversitesi Dergisi 2013, 14(2): 175-94.

Peters M., Zaki M. J., CLICK (2002). Clustering Categorical Data Using K-partite Maximal Cliques, Survey of Clustering Data Mining Techniques.

Nilsson N. J (1996). Introduction to Machine Learning, An Early Draft of a Proposed Textbook, Stanford University, Department of Computer Science, Robotics Laboratory, CA 94305

Cobos, C.; Andrade, J.; Constain, W.; Mendoza, M.; León, E.: (2010) .Web Document Clustering Based on Global-Best Harmony Search, K-Means, Frequent Term Sets and Bayesian Information Criterion, IEEE Congress on Evolutionary Computation Barcelona, 1-8

Dhillon, I. S.; Guan, Y.; Kogan, J.: (2002) . Iterative Clustering of High Dimensional Text Data Augmented by Local Search, 2nd IEEE International Conference on Data Mining, 131-138.

EKLER

EK 1 K-Means Python Dilindeki Kodları

```
# Dividing based on K-Means Clusters
df['K-Means_Labels'] = clusters

# Short work emails. Internal work emails.
df_0 = df[df.K-Means_Labels == 0]
temp_0 = pd.DataFrame(pd.Series(' '.join(df_0['Cleaned_Ema
il']).split()).value_counts()[:200].sort_values())

# Fantasy Football
df_1 = df[df.K-Means_Labels == 1]
temp_1 = pd.DataFrame(pd.Series(' '.join(df_1['Cleaned_Ema
il']).split()).value_counts()[:200].sort_values())

# Talking about gas market. Maybe these are the business e
mails. COntract deal price.
df_2 = df[df.K-Means_Labels == 2]
temp_2 = pd.DataFrame(pd.Series(' '.join(df_2['Cleaned_Ema
il']).split()).value_counts()[:200].sort_values())

# Mostly spam offers services lots of images. This stayed
when size was increased also
df_3 = df[df.K-Means_Labels == 3]
temp_3 = pd.DataFrame(pd.Series(' '.join(df_3['Cleaned_Ema
il']).split()).value_counts()[:200].sort_values())

# Personal Emails
df_4 = df[df.K-Means_Labels == 4]
```

```
temp_4 = pd.DataFrame(pd.Series(' '.join(df_4['Cleaned_Ema
il']).split()).value_counts()[:200].sort_values())

#plt.bar(temp_4.index.values, pd.Series(' '.join(df_4['Cle
aned_Email']).split()).value_counts()[:20].sort_values(),
align='center', alpha=0.5)
#plt.xticks(rotation='vertical')

# Filtering K-Means Labels
index_label_0 = df.index[df['K-
Means_Labels'] == 0].tolist()
X0 = [X[x-1] for x in index_label_0[:-1]]

index_label_1 = df.index[df['K-
Means_Labels'] == 1].tolist()
X1 = [X[x-1] for x in index_label_1[:-1]]

index_label_2 = df.index[df['K-
Means_Labels'] == 2].tolist()
X2 = [X[x-1] for x in index_label_2[:-1]]

index_label_3 = df.index[df['K-
Means_Labels'] == 3].tolist()
X3 = [X[x-1] for x in index_label_3[:-1]]

index_label_4 = df.index[df['K-
Means_Labels'] == 4].tolist()
X4 = [X[x-1] for x in index_label_4[:-1]]

from sklearn.cluster import DBSCAN
min_samples_set = int(math.ceil(math.log(len(X3))))
db = DBSCAN(eps=1.83, min_samples=min_samples_set).fit(X3)
labels = db.labels_
```

```
svd = TruncatedSVD(n_components = 2)
reduced_data_0 = svd.fit_transform(X0)
print(np.unique(labels))
plt.scatter(reduced_data_0[:,0], reduced_data_0[:,1], c=[m
atplotlib.cm.spectral(float(i) /10) for i in labels])
```

EK 2 Dbscan Python Dilinde Oluşturulması

```
# Dividing based on DB Scan
df_0 =  df_0[:-1]
df_0['Dbscan_Labels'] = labels

df_0_0 = df_0[df_0.Dbscan_Labels == 0]
temp_0_0 = pd.DataFrame(pd.Series(' '.join(df_0_0['Cleaned
_Email']).split()).value_counts().sort_values())

df_0_1 = df_0[df_0.Dbscan_Labels == 1]
temp_0_1 = pd.DataFrame(pd.Series(' '.join(df_0_1['Cleaned
_Email']).split()).value_counts().sort_values())

df_0_2 = df_0[df_0.Dbscan_Labels == 2]
temp_0_2 = pd.DataFrame(pd.Series(' '.join(df_0_2['Cleaned
_Email']).split()).value_counts().sort_values())

df_0_3 = df_0[df_0.Dbscan_Labels == 3]
temp_0_3 = pd.DataFrame(pd.Series(' '.join(df_0_3['Cleaned
_Email']).split()).value_counts().sort_values())

df_0_4 = df_0[df_0.Dbscan_Labels == 4]
temp_0_4 = pd.DataFrame(pd.Series(' '.join(df_0_4['Cleaned
_Email']).split()).value_counts().sort_values())
# Determining optimal value of eps Label 1
```

```
from sklearn.neighbors import NearestNeighbors
import matplotlib.pyplot as plt
# Min_samples set to ln(number of points)
min_samples_set = int(math.ceil(math.log(len(X1))))
nbrs = NearestNeighbors(n_neighbors=min_samples_set).fit(X
1)
distances, indices = nbrs.kneighbors(X1)
distanceDec = sorted(distances[:,min_samples_set-
1], reverse=True)
plt.plot(list(range(1,len(X1) + 1)), distanceDec)
from sklearn.cluster import DBSCAN
min_samples_set = int(math.ceil(math.log(len(X1))))
db = DBSCAN(eps=1.3, min_samples=min_samples_set).fit(X1)
labels = db.labels_
svd = TruncatedSVD(n_components = 2)
reduced_data_1 = svd.fit_transform(X1)
print(np.unique(labels))
plt.scatter(reduced_data_1[:,0], reduced_data_1[:,1], c=[m
atplotlib.cm.spectral(float(i) /10) for i in labels])

# Dividing based on DB Scan
df_1 =  df_1[:-1]
df_1['Dbscan_Labels'] = labels

df_1_0 = df_1[df_1.Dbscan_Labels == 0]
temp_1_0 = pd.DataFrame(pd.Series(' '.join(df_1_0['Cleaned
_Email']).split()).value_counts()[:20].sort_values())

df_1_1 = df_1[df_1.Dbscan_Labels == 1]
temp_1_1 = pd.DataFrame(pd.Series(' '.join(df_1_1['Cleaned
_Email']).split()).value_counts()[:20].sort_values())

df_1_2 = df_1[df_1.Dbscan_Labels == 2]
```

```
temp_1_2 = pd.DataFrame(pd.Series(' '.join(df_1_2['Cleaned
_Email']).split()).value_counts()[:20].sort_values())

df_1_3 = df_1[df_1.Dbscan_Labels == 3]
temp_1_3 = pd.DataFrame(pd.Series(' '.join(df_1_3['Cleaned
_Email']).split()).value_counts()[:20].sort_values())
```

EK 2 .Dbscan Python Dilindeki Kodları

```
!pip install gensim
!pip install xlrd
!pip install openpyxl
import pandas as pd
import xlrd
import re
import io
import numpy as np
import math
from gensim.models.doc2vec import Doc2Vec, TaggedDocument
# Reading veri seti and extracting features and emails
data = pd.ExcelFile("Unique2.xlsx")
df = data.parse("Sheet 1")
df = df.dropna()
number_rows = 40000
df = df.iloc[list(range(0, number_rows)), :]



# Performing Dbscan
from sklearn.cluster import DBSCAN
from sklearn.decomposition import TruncatedSVD
import matplotlib
import matplotlib.pyplot as plt
min_samples_set = int(math.ceil(math.log(number_rows)))
db = DBSCAN(eps=1.85, min_samples= min_samples_set).fit(X)
labels = db.labels_
print(np.unique(labels))
svd = TruncatedSVD(n_components = 2)
```

```
reduced_data = svd.fit_transform(X)
plt.scatter(reduced_data[:,0], reduced_data[:,1], c = [matplotlib.
cm.spectral(float(i) /10) for i in labels])


# Dividing based on DB Scan
df_0 =  df_0[:-1]
df_0['Dbscan_Labels'] = labels

df_0_0 = df_0[df_0.Dbscan_Labels == 0]
temp_0_0 = pd.DataFrame(pd.Series(' '.join(df_0_0['Cleaned_Email']
).split()).value_counts().sort_values())

df_0_1 = df_0[df_0.Dbscan_Labels == 1]
temp_0_1 = pd.DataFrame(pd.Series(' '.join(df_0_1['Cleaned_Email']
).split()).value_counts().sort_values())

df_0_2 = df_0[df_0.Dbscan_Labels == 2]
temp_0_2 = pd.DataFrame(pd.Series(' '.join(df_0_2['Cleaned_Email']
).split()).value_counts().sort_values())

df_0_3 = df_0[df_0.Dbscan_Labels == 3]
temp_0_3 = pd.DataFrame(pd.Series(' '.join(df_0_3['Cleaned_Email']
).split()).value_counts().sort_values())

df_0_4 = df_0[df_0.Dbscan_Labels == 4]
temp_0_4 = pd.DataFrame(pd.Series(' '.join(df_0_4['Cleaned_Email']
).split()).value_counts().sort_values())
```

ÖZGEÇMİŞ

Adı Soyadı	: Melek GÜLER MANAV
Doğum Yeri ve Tarih	: BURSA 10.05.1985
Yabancı Dil	: İngilizce
İletişim (e-posta)	: melekguler16@gmail.com
Eğitim Durumu	
Lise	: Bursa Anadolu Lisesi
Lisans	: İstanbul Üniversitesi
Yüksek Lisans	: Uludağ Üniversitesi
Çalıştığı Kurum(lar)	:
	Durmazlar-Software Engineering Manager
	June 2021 - April 2022 (11 months)
	Siberson-New Business Development Manager
	July 2018 - July 2020 (2 years 1 month)
	FICOSA-ERP Consultant
	June 2017 - 2018 (1 year)
	Faurecia-IT Engineer
	January 2012 - June 2014 (2 years 6 months)
	Turkcell Technology-Erp Uzmanı
	January 2010 - February 2011 (1 year 2 months)
	Intertech /Denizbank -Yazılım Mühendisi
	November 2008 - January 2010 (1 year 3 months)