T.C.
AYDIN ADNAN MENDERES ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
MATEMATİK ANABİLİM DALI
2019-YL-057

MERKEZ TABANLI METİN SINIFLANDIRMA

Orkun HARAÇVERMEZ

Tez Danışmanı:
Dr. Öğr. Üyesi Rıfat AŞLIYAN

AYDIN

T.C.
AYDIN ADNAN MENDERES ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ MÜDÜRLÜĞÜNE
AYDIN

Matematik Anabilim Dalı Yüksek Lisans Programı öğrencisi Orkun HARAÇVERMEZ tarafından hazırlanan "Merkez Tabanlı Metin Sınıflandırma" başlıklı tez, 9.07.2019 tarihinde yapılan savunma sonucunda aşağıda isimleri bulunan jüri üyelerince kabul edilmiştir.

	Unvanı, Adı Soyadı	Kurumu	İmzası
Başkan :	Dr. Öğr. Üyesi Rıfat AŞLIYAN	Aydın Adnan Menderes Üni. Fen Edeb. Fakültesi	……
Üye :	Dr. Öğr. Üyesi Korhan GÜNEL	Aydın Adnan Menderes Üni. Fen Edeb. Fakültesi	……
Üye :	Dr. Öğr. Üyesi Refet POLAT	Yaşar Üniversitesi Fen Edeb. Fakültesi	……

Jüri üyeleri tarafından kabul edilen bu Yüksek Lisans tezi, Enstitü Yönetim Kurulunun …… sayılı kararıyla …… tarihinde onaylanmıştır.

Prof. Dr. Gönül AYDIN
Enstitü Müdürü

T.C.
AYDIN ADNAN MENDERES ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ MÜDÜRLÜĞÜNE
AYDIN

Bu tezde sunulan tüm bilgi ve sonuçların, bilimsel yöntemlerle yürütülen gerçek deney ve gözlemler çerçevesinde tarafımdan elde edildiğini, çalışmada bana ait olmayan tüm veri, düşünce, sonuç ve bilgilere bilimsel etik kuralların gereği olarak eksiksiz şekilde uygun atıf yaptığımı ve kaynak göstererek belirttiğimi beyan ederim.

10/06/2019

Orkun HARAÇVERMEZ

ÖZET

MERKEZ TABANLI METİN SINIFLANDIRMA

Orkun HARAÇVERMEZ

Yüksek Lisans Tezi, Matematik Anabilim Dalı
Tez Danışmanı: Dr. Öğr. Üyesi Rıfat AŞLIYAN
2019, 71 sayfa

Son yıllarda bilişim alanındaki büyük ilerlemeden dolayı çok miktarda metin verilerinin işlenmesi ve sınıflandırılması ihtiyacı vardır. Sınıflandırılan verilere çok daha hızlı erişim mümkün olmaktadır. Metin sınıflandırma, dokümanların otomatik olarak önceden belirlenmiş metin kategorilerinin hangisine ait olduğunu belirleme işlemidir. Sınıflandırma yapmak için metin verileri eğitim ve test veri seti olmak üzere iki parçaya bölünür. Eğitim seti ile metotlar eğitilir. Test veri seti ile metotların sınıflandırma başarısı tespit edilir. Sınıflandırılacak dokümanlar ilk olarak ön işlemden geçirilir. Sonra her dokümanın öznitelikleri çıkarılarak doküman vektörleri oluşturulur. Bu vektörler metotlara girdi olarak alınır ve metotlar eğitilir. Böylece metoda bağlı eğitilmiş model elde edilir. Test aşamasında ise verilen bir dokümanın öznitelik vektörüne dönüştürüldükten sonra metot modeli ile hangi sınıfa ait olduğu belirlenir.

Bu tezde internet üzerinden elde edilen metin verilerini merkez tabanlı metotlardan Standart Merkez Tabanlı Metot, Yaklaştır-Uzaklaştır Metot, Geniş Mesafe Yaklaştır-Uzaklaştır Metot ve Çekim Modeli ile diğer önemli metotlardan K-En Yakın Komşu, Naive Bayes ve Destek Vektör Makinesi kullanarak uygulamalar yapılmıştır. Bahsi geçen yöntemlerin performansları, Doğruluk ve F1-Ölçüsü metrikleri kullanılarak karşılaştırılmıştır.

Anahtar Sözcükler: Metin Sınıflandırma, Merkez Tabanlı Metin Sınıflandırma, KEYK, Naive Bayes, DVM

ABSTRACT

CENTROID-BASED TEXT CLASSIFICATION

Orkun HARAÇVERMEZ

M.Sc. Thesis, Department of Mathematics
Supervisor: Rıfat AŞLIYAN Ph.D.
2019, 71 pages

In recent years, it is necessary to process and classify a lot of text textual data because of great advances in information technology. After text data is classified, it can be accessed very fast. Text classification is the process of automatically assigning the most suitable classes for documents. For classification the textual data is divided by two sets as training and testing sets. For classification, methods are trained with the training sets. With the test sets, we determine how the performance of a method is. The documents for classification are firstly preprocessed. After that, the features of each document are extracted and document vectors are constructed. These vectors are given as inputs for the classification methods and are trained with the methods. In this way, the models of each method are generated according to the training set. After converting the document to the feature vectors, in the testing operation, the document given as an input to categorize is decided which class it belongs to.

In this thesis, using the methods as Standard Centroid Based Method, Drag-Push Method, Large Margin Drag Push Method, Gravitational Model, K-Nearest Neighbor, Naive Bayes and Support Vector Machine we have developed the applications with the data sets obtained on the Internet. The performance of these methods are evaluated and compared with each other according to the Accuracy and F1-Score metrics.

Keywords: Text Classification, Centroid-Based Text Classification, K-NN, Naive Bayes, SVM

ÖNSÖZ

Tez çalışmamda planlanmasında, araştırılmasında, yürütülmesinde ve oluşumunda ilgi ve desteğini esirgemeyen, engin bilgi ve tecrübelerinden yararlandığım, yönlendirme ve bilgilendirmeleriyle çalışmamı bilimsel temeller ışığında şekillendiren sayın hocam Dr. Öğr. Üyesi Rıfat AŞLIYAN'a sonsuz teşekkürlerimi sunarım.

Orkun HARAÇVERMEZ

İÇİNDEKİLER

KISALTMALAR DİZİNİ

DVM	Destek Vektör Makinesi
SVM	Support Vector Machine
NBS	Naive Bayes Sınıflandırıcı
MNBS	Multinomial Naive Bayes Sınıflandırıcı
KNN	K-Nearest Neighbor
KEYK	K-En Yakın Komşu
TF	Term Frequency
TF-IDF	Term Frequency-Inverse Document Frequency
SMTM	Standart Merkez Tabanlı Metot
MTYUM	Merkez Tabanlı Yaklaştır-Uzaklaştır Metodu
MTGMYUM	Merkez Tabanlı Geniş Mesafe Yaklaştır-Uzaklaştır Metodu
MTÇM	Merkez Tabanlı Çekim Modeli
CBC	Centroid-Based Classifier
GM	Gravitational Model
OÖA	Olasılıksal Öğrenme Algoritması
TÖA	Toplu Öğrenme Algoritması

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

EKLER DİZİNİ

1. GİRİŞ

Günümüzde bilgisayar teknolojilerinin çok hızlı gelişmesinden dolayı metin verileri çok büyük ölçüde artmıştır. Bu verileri bireysel olarak kontrol etmek, incelemek veya sınıflandırmak artık mümkün değildir. Çünkü bunu yapmak için hem çok zaman gerekir, hem de bu işlem, insanlar için çok sıkıcı ve yorucu olmaktadır. Aynı zamanda belirli metinlerin sınıflandırılması kişiye göre değişebilir yani objektif olmayabilir. Bu sebeple metinlerin otomatik olarak sınıflandırılması gerekmektedir. Böylece çok daha hızlı, çok daha başarılı ve objektif bir sınıflandırma yapılmış olur.

Metin sınıflandırma (Salton, 1989; Cohen ve Hirsh, 1998; Joachims, 1998; McCallum ve Nigam, 1998; Yang ve Liu, 1999), metin dokümanların önceden belirlenmiş sınıflara otomatik olarak atanması işlemidir. Otomatik sınıflandırma, büyük miktardaki verilerden önemli bilgilere erişim hızlı olacağından oldukça faydalıdır. Metin sınıflandırmayı başarıyla gerçekleştirmek, veri setlerindeki büyük miktardaki örneklerden yani dokümanlardan, dokümanlardaki özniteliklerin çok fazla olmasından ve çok sayıda sınıfların olabilmesinden dolayı zor bir süreçtir.

Merkez tabanlı sınıflandırıcılar (Lertnattee ve Theeramunkong, 2006; Tan, 2007; Tan, 2008; Guan vd., 2009; Takçı ve Güngör, 2012; Dandan vd., 2014) metin sınıflandırma metotları arasında çok tercih edilen metotlardandır. Çünkü diğer metotlara göre çok verimlidir. Eğitim aşamasında algoritma karmaşıklığı doğrusaldır. Bu yüzden çok büyük boyuttaki metinleri sınıflandırmada çok önemli rol oynar. Merkez tabanlı sınıflandırıcılarda bir dokümanın eğitim setindeki sınıfların önceden belirlenmiş merkezlerine olan uzaklıkları belirlenerek bu dokümana en çok benzeyen sınıf belirlenir.

Günümüze kadar bir çok farklı metin sınıflandırma metotları geliştirilmiş ve uygulanmıştır (Baker ve McCallum, 1998; Joachims, 1998; Lam ve Ho, 1998; Yang ve Liu, 1999; Aşlıyan ve Günel, 2010; Kolyiğit vd., 2012). Günel, Aşlıyan ve Gör (2016), bir örneğin ait olduğu sınıfı belirlerken örneği temsil eden referans vektörünü dinamik olarak oluşturan hiper küreler üzerinde döndürerek kendisine en çok benzeyen sınıfın merkezine yakınlaşırken diğer sınıfların merkezlerinden uzaklaşmayı garantilemişlerdir. Dokümanların sınıflandırılmasında en çok kullanılan metotlar Merkez Tabanlı Sınıflandırma (Lertnattee ve Theeramunkong,

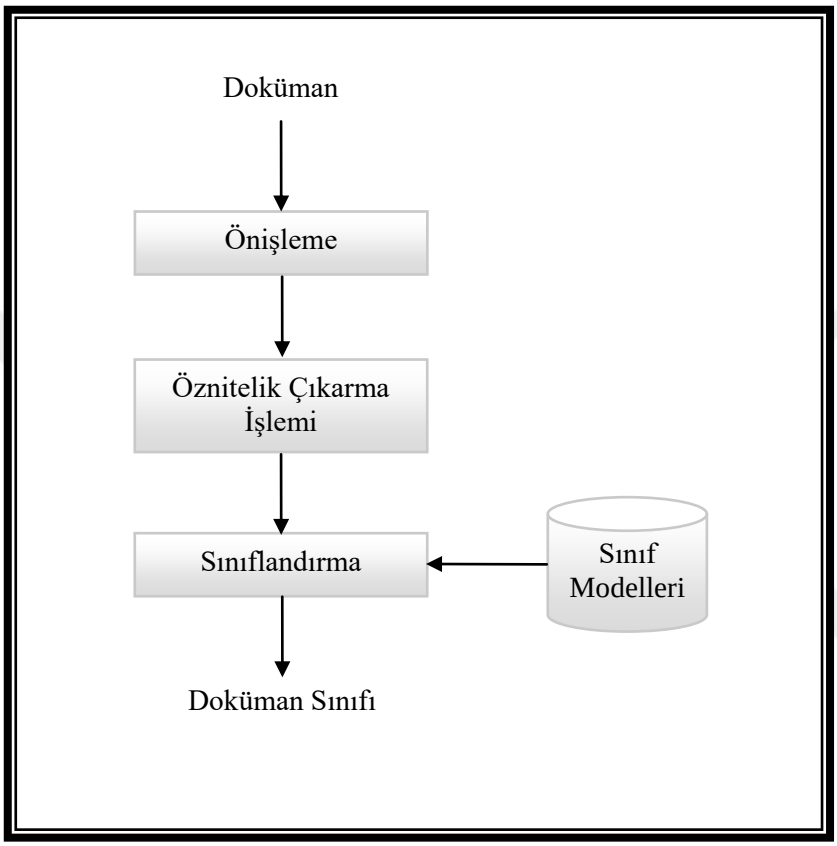
2006; Tan, 2007; Cataltepe ve Aygun, 2007; Tan, 2008; Guan vd., 2009; Takçı ve Güngör, 2012; Dandan vd., 2014; Liu vd., 20017), K-En Yakın Komşu (Yang ve Liu, 1999; Guo vd., 2006; Yu vd., 2016), Naive Bayes (McCallu ve Nigam, 1998; Kibriya vd., 2004), Destek Vektör Makinesi (Chen ve Hsieh, 2006; Zhang vd., 2006), C4.5 (Quinlan, 1993), Yapay Sinir Ağları (Chau vd., 2005; Mikolov vd., 2013; Agarwal ve Mittal, 2014), Karar Ağaçları (Lam ve Lee, 1999), Graf Tabanlı (Colace, 2014) ve Grup Sınıflandırıcı (Yu vd., 2015) metotlarıdır.

Şekil 1.1.'de görüldüğü gibi metin sınıflandırma üç aşamadan oluşmaktadır. Birincisi Önişleme, ikincisi Öznitelik Çıkarma ve son aşama da Sınıflandırma işlemidir.

Önişleme aşamasında girdi olarak alınan dokümanın, hangi dilde ise o dile ait harfleri alınmaktadır. Harfler dışındaki diğer karakterler atılmaktadır. Tüm harfler küçük harflere dönüştürülür. Sözcükler arasındaki fazla boşluklar silinerek her sözcük arasında sadece bir karakterlik boşluk bırakılması sağlanır.

Öznitelik Çıkarma'da ise önişlemden geçirilen dokümanlar sayısal verilere sahip vektörlere dönüştürülür. Burada her bir sözcüğün doküman içindeki frekansı (TF) (Doküman içinde kaç defa geçtiği) sonrasında sözcük-frekans-ters-doküman frekansı (TF-IDF) (Term-Frequency-Inverse-Document Frequency) değerleri hesaplanır. Doküman frekansı, bir sözcüğün bir sınıftaki kaç dokümanda geçtiği anlamına gelmektedir. Bu değerlere sahip vektörler en son olarak normalize edilirler.

En son aşamada vektörler şeklinde temsil edilen dokümanların, sınıf modelleri kullanılarak hangi sınıfa ait oldukları tespit edilir. Burada bir veya birkaç sınıflandırma metoduna ihtiyaç duyulur. Fakat ilkin bu sınıf modellerinin yine aynı sınıflandırma metodu ile eğitim aşamasındaki veri seti kullanılarak Önişleme ve Öznitelik Çıkarma işlemi yapılarak oluşturulması gereklidir.



Şekil 1.1. Metin sınıflandırma genel yapısı

Bu tezin geri kalan bölümleri şu şekilde organize edilmiştir. İkinci bölümünde merkez tabanlı sınıflandırma metotları, Destek Vektör Makinesi, Naive Bayes ve K-En Yakın Komşu metotları ayrıntılı bir şekilde açıklanmıştır. Üçüncü bölümde ise bu metotların başarıları performans ölçülerine göre verilmiştir ve karşılaştırılmıştır. Dördüncü yani sonuç bölümünde tüm sistemlerin başarıları özet olarak ifade edilmiştir.

2. MATERYAL VE YÖNTEM

Bu bölümde standart merkez tabanlı metin sınıflandırma, DrugPush (Yaklaştır-Uzaklaştır) (DP), Large Margin DragPush (Büyük Mesafe Yaklaştır-Uzaklaştır) (LMDP) ve Gravitational Model (Çekimsel Model) merkez tabanlı metotlar bahsedilecektir. Buna ek olarak K-En Yakın Komşu (KEYK), Naive Bayes ve Destek Vektör Makinesi (DVM) metotları açıklanacaktır. Bir sonraki bölümde bu metotlar deneysel sonuçlarla karşılaştırılmıştır.

2.1. Standart Merkez Tabanlı Metot (SMTM)

Merkez tabanlı metotların (Lertnattee ve Theeramunkong, 2006; Tan, 2007; Cataltepe ve Aygun, 2007; Tan, 2008; Guan vd., 2009; Takçı ve Güngör, 2012; Dandan vd., 2014; Liu vd. 20017) sınıflandırılmasında, veri setinde bulunan dokümanlar Vektör Uzayı Modeli (VUM) şeklinde temsil edilirler. Bu modelde her $\vec{d}$ dokümanı terim uzayındaki bir vektör olarak kabul edilir. C_n, n. sınıfı ifade etmektedir. $\overrightarrow{C_n^S}$ ve $\overrightarrow{C_n^N}$ sırasıyla n. sınıfa ait dokümanların toplamı ve dokümanların toplamının normalize edilmiş vektörlerini temsil etmektedir.

Denklem 2.1 ve 2.2'de görüldüğü gibi ilk olarak, her C_n sınıfına ait doküman vektörlerinin toplamı hesaplandıktan sonra $\overrightarrow{C_n^S}$ vektörü 2-norm'una bölünerek normalize edilir ve $\overrightarrow{C_n^N}$ olarak temsil edilir.

$$\overrightarrow{C_n^S} = \sum_{d \in C_n} \vec{d} \tag{2.1}$$

$$\overrightarrow{C_n^N} = \frac{\overrightarrow{C_n^S}}{\left\| \overrightarrow{C_n^S} \right\|_2} \tag{2.2}$$

Denklem 2.3'te belirtildiği üzere $\vec{d}$ vektörü ile C_n sınıfı arasındaki benzerlik, $\vec{d}$ vektörü ile $\overrightarrow{C_n^S}$ vektörünün noktasal çarpımı ile bulunur. Buradaki noktasal çarpım işlemi kosinüs benzerliğine denktir. Çünkü $\vec{d}$ doküman vektörü, TF-IDF denklemi (Denklem 2.5) kullanılarak ve sonrasında 2-norm'a göre normalize edilerek hesaplanmaktadır. Denklem 2.4'teki D, eğitim setindeki toplam doküman sayısıdır. m_t ise t terimini içeren doküman sayısıdır. $TF(t, \vec{d})$, $\vec{d}$ dokümanındaki t teriminin sıklığını temsil etmektedir. ε değeri de 0,01 olarak alınmıştır.

$$sim(\vec{d}, C_n) = \vec{d} \cdot \overrightarrow{C_n^S} \tag{2.3}$$

$$w_{TF-IDF}(t,\vec{d}) = TF(t,\vec{d}) \times \log(D/m_t + \varepsilon) \quad (2.4)$$

$$W^N_{TF-IDF}(t,\vec{d}) = \frac{TF(t,\vec{d})\times\log(D/m_t+\varepsilon)}{\sqrt{\sum_{t\in\vec{d}}[TF(t,\vec{d})\times\log(D/m_t+\varepsilon)]^2}} \quad (2.5)$$

$$C = \arg\max_{C_n}\left(\vec{d}\cdot\overrightarrow{C_n^N}\right) \quad (2.6)$$

Denklem 2.6'yı kullanarak $\vec{d}$ vektörüne en çok benzeyen C sınıfı hesaplanır.

2.2. Merkez Tabanlı Yaklaştır-Uzaklaştır (DragPush) Metodu (MTYUM)

Standart merkez sınıflandırma yapılırken, veri setlerine bağlı olarak merkezleri tespit edildiğinde hem test hem de eğitim setinin doğruluk başarı oranı düşebilmektedir. Bu problemi çözmek için en doğru yol eğitim setindeki örnekler ile merkezleri yeniden ayarlamaktır. Yaklaştır-Uzaklaştır (Liu vd., 2017) metodu merkezleri ayarlayarak sınıflandırma başarısını artırmayı amaçlar. Böylece hem eğitim hem de test setindeki doğruluk oranını iyileştirmek mümkün olur.

Bu yaklaşımda ilk olarak iterasyon sayısı (Maksimum İterasyon) ve öğrenme oranının (α) kaç olacağını belirlemek gerekir. Devamında her C_n sınıfı için $\overrightarrow{C_n^{S,0}}$ ve $\overrightarrow{C_n^{N,0}}$ (n. sınıfın toplam ve toplam normalize vektörleri) vektörleri hesaplanır. 0, ilk iterasyonu ifade etmektedir. $0 \rightarrow 1$ ise bir sonraki iterasyon anlamına gelmektedir. Her iterasyonda eğitim setindeki dokümanların her biri sınıflandırılır. Farz edelim ki *A* ve *B* olmak üzere iki sınıf olsun. *A* sınıfına ait bir $\vec{d}$ dokümanı, *B* sınıfı olarak yanlış sınıflandırılmış olsun. Denklem 2.7, 2.8, 2.9 ve 2.10'da görüldüğü üzere *A* sınıfının merkezi $\vec{d}$ dokümanına öğrenme oranı katsayısına göre yaklaştırılır. *B* sınıfının merkezi ise aynı oranda uzaklaştırılır. Burada $\vec{d}$ dökümanının her öznitelik değeri sıfırdan büyük olduğu değerde aşağıdaki işlemler gerçekleştirilir.

$$\overrightarrow{C_A^{S,0\rightarrow 1}} = \overrightarrow{C_A^{S,0}} + \alpha\times\vec{d} \quad (2.7)$$

$$\overrightarrow{C_A^{N,0\rightarrow 1}} = \frac{\overrightarrow{C_A^{S,0}}}{\left\|\overrightarrow{C_A^{S,0}}\right\|_2} \quad (2.8)$$

$$\overrightarrow{C_B^{S,0\rightarrow 1}} = \left[\overrightarrow{C_B^{S,0}} - \alpha\times\vec{d}\right]_+ \quad (2.9)$$

$$\overrightarrow{C_B^{N,0\rightarrow 1}} = \frac{\overrightarrow{C_B^{S,0}}}{\left\|\overrightarrow{C_B^{S,0}}\right\|_2} \tag{2.10}$$

Denklem 2.9'da $[x]_+$ ifadesi x sıfırın altında bir değere sahip olursa 0 değerini verir; 0'dan büyük değer içerirse o değeri çıktı olarak verir. Yani $[x]_+ = \max\{0, x\}$ olarak ifade edilir. Denklem 2.7 ve 2.8, $\vec{d}$ dokümanı yanlış sınıflandırıldığında yani A sınıfındaki $\vec{d}$ dokümanı B sınıfı olarak sınıflandırılırsa A sınıfına yaklaştırılacağını ifade etmektedir. Denklem 2.9 ve 2.10 ise yine $\vec{d}$ dokümanı yanlış sınıflandırıldığında yani A sınıfındaki $\vec{d}$ dokümanı B sınıfı olarak sınıflandırılırsa B sınıfından uzaklaştırılacağını göstermektedir.

2.2.1. Yaklaştır-Uzaklaştır Algoritması

1-Eğitim verilerini ve parametrelerini yükleyin

2-Her C_n sınıfı için $\overrightarrow{C_n^{S,0}}$ ve $\overrightarrow{C_n^{N,0}}$ vektörlerini hesaplayın

3-Döngü D1 n=1 : MaksimumİterasyonDöngü D1 n=1 : Maksimumİterasyon

4- Döngü D2 eğitim setindeki her $\vec{d}$ doküman için

5- Eğer A sınıfındaki $\vec{d}$ dokümanı B sınıfı olarak sınıflandırılırsa

6- A sınıfının merkezini $\vec{d}$ dokümanına yaklaştır

7- B sınıfının merkezini $\vec{d}$ dokümanından uzaklaştır

8- Eğer Yapısı Sonu

9- Döngü D2 Sonu

10-Döngü D1 Sonu

2.3. Merkez Tabanlı Geniş Mesafe Yaklaştır-Uzaklaştır (Large Margin DragPush) Metodu (MTGMYUM)

Geniş mesafe yaklaştır-uzaklaştır metodundaki mesafe düşüncesi, destek vektör makinesindeki (Vapnik, 1995) sınıflar arasındaki mesafeden esinlenerek ortaya

atılmıştır. Eğitim setindeki sınıfları ayıran düzlemin sınıflara olan mesafesinin en uygun seviyede olması amaçlanır.

Yaklaştır-uzaklaştır metodu sınıflandırma yaparken genelleştirme yapabilmektedir. Fakat genelleştirmeyi garantileyememektedir. Sadece eğitim setinin sınıflandırma hatasını amaç fonksiyonunun kriteri olarak kabul etmektedir. Bu sebeple yaklaştır-uzaklaştır metodunun başarısını artırmak için DVM'deki mesafe düşüncesi bu algoritmaya dahil edilmiştir.

Bu metot, her eğitim örneği için hipotez-mesafe ya da yaklaşık mesafenin hesaplanmasına ihtiyaç duyar. Hipotez-mesafe ile hipotez sınıf üzerinde bir uzaklık ölçüsü belirlenmektedir. Bir örneğin hipotez-mesafesi, kendi sınıfındaki ve diğer sınıflardaki en yakın örnekler arasındaki mesafe olarak ifade edilir. Denklem 2.11'de $\overrightarrow{x_p}$ örneğinin kendi sınıfındaki en yakın örneği $\overrightarrow{x_R}$ ve diğer sınıflara olan en yakın örnek x_M olmaktadır.

$$HM_p(\overrightarrow{x_p}) = \frac{1}{2}\left(\left\|\overrightarrow{x_p} - \overrightarrow{x_R}\right\| - \left\|\overrightarrow{x_p} - \overrightarrow{x_M}\right\|\right) \tag{2.11}$$

Denklem 2.11'e benzer şekilde Denklem 2.12'yi ifade edebiliriz. Bir $\overrightarrow{x_n}$ örneğinin C merkez sınıflandırıcısına göre hipotez mesafesi aşağıdaki gibi hesaplanır. Burada $\overrightarrow{C_R}$, $\overrightarrow{x_n}$ örneğinin kendi sınıfının merkezidir. $\overrightarrow{C_M}$ ise $\overrightarrow{x_n}$ örneğinin farklı sınıfların merkezlerin en yakın merkezi olmaktadır.

$$HM_C(\overrightarrow{x_n}) = \left(Sim\left(\overrightarrow{x_n}, \overrightarrow{C_R}\right) - Sim\left(\overrightarrow{x_n}, \overrightarrow{C_M}\right)\right) \tag{2.12}$$

Eğitim hatasının tanımına uygun olarak her örnek için genelleştirilmiş hata yani "MesafeHata" tanımlanır. Farz edelim ki bir $\vec{d}$ örneğinin hipotez-mesafesi sıfırdan daha büyükse ve 0,05 gibi çok küçük bir sabit sayıdan küçükse merkez sınıflandırıcısı $\vec{d}$ örneğine göre "MesafeHata" oluşturmuştur. Bu küçük sabit değer "MinimumMesafe" olarak adlandırılır. Bu durumda $\vec{d}$ örneği minimum mesafededir denir. Eğer $\vec{d}$ örneği minimum mesafede ise Denklem 2.7 ve 2.9, Denklem 2.13 ve 2.14'teki gibi olacak şekilde revize edilir.

$$\overrightarrow{C_A^{S,0\to 1}} = \overrightarrow{C_A^{S,0}} + MesafeAğırlık \times \vec{d} \tag{2.13}$$

$$\overrightarrow{C_B^{S,0\to 1}} = \left[\overrightarrow{C_B^{S,0}} - MesafeAğırlık \times \vec{d}\right]_+ \tag{2.14}$$

2.3.1. Geniş Mesafe Yaklaştır-Uzaklaştır Algoritması

1-Eğitim verilerini ve parametrelerini yükleyin

2-Her C_n sınıfı için $\overrightarrow{C_n^{S,0}}$ ve $\overrightarrow{C_n^{N,0}}$ vektörlerini hesaplayın

3-Döngü D1 n=1 : MaksimumİterasyonDöngü D2 eğitim setindeki her $\vec{d}$ doküman için

4- Her $\vec{d}$ dokümanı için C_R ve C_M'yi belirleyin

5- Eğer $HM(d_n) < 0$ ise

6- Denklem 2.7 ve 2.8'i yani yaklaştır işlemini uygulayın

7- Denklem 2.9 ve 2.10'u yani uzaklaştır işlemini uygulayın

8- Değilse Eğer $HM(d_n) < MinimumMesafe$ ise

9- Denklem 2.13 ve 2.8'i yani yaklaştır işlemini uygulayın

10- Denklem 2.14 ve 2.10'u yani uzaklaştır işlemini uygulayın

11- Eğer Yapısı Sonu

12- Döngü D2 Sonu

13- Döngü D1 Sonu

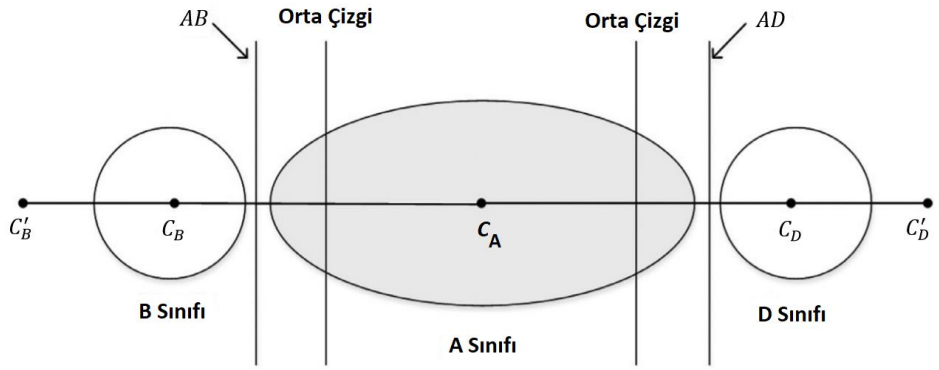
14-

2.4. Merkez Tabanlı Çekim Modeli (Gravitation Model) (MTÇM)

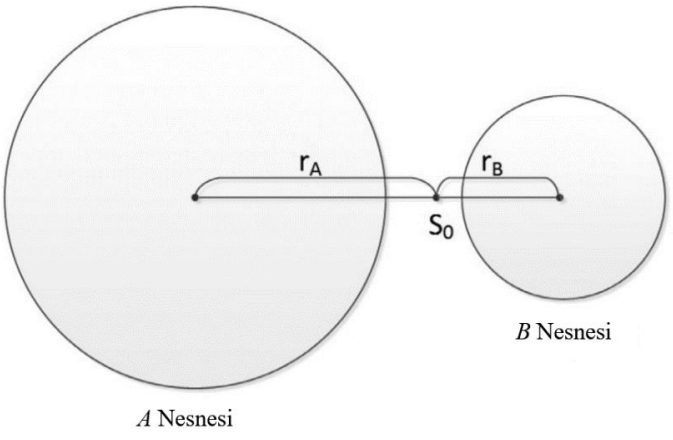
Çekim modeli (Gravitation Model-GM), merkez tabanlı sınıflandırıcıların (CBC) dengesiz sınıf veri setlerindeki eksikliğini gidermek için ortaya atılmış bir metottur. Bu model, CBC'den kaynaklanan eksikliği azaltmak için sınıflandırıcı hiper düzlemi daha iyi duruma getirmek için ayarlamalar yapar. MTÇM, daha önceki çalışmalardan oldukça farklıdır. Önceki modeller, model oluşturulurken terim ağırlıklarını ayarlayarak ya da eğitim aşamasında sınıf merkezlerinin pozisyonlarını değiştirerek sınıf merkezlerini bulur.

Çekim modeli, Newton'un evrensel çekim yasasından esinlenilerek tasarlanmıştır. Newton yasasına göre evrendeki her nesne birbirini bir kuvvetle çekmektedir. İki nesne arasındaki çekim kuvveti Denklem 2.15'te görüldüğü gibi hesaplanmaktadır. Burada G, yerçekimi sabiti; M, birinci nesnenin kütlesi; m ise ikinci nesnenin kütlesidir. r, iki nesne arasındaki uzaklığı ifade etmektedir. Bu denklem şunu ifade etmektedir: İki nesne arasındaki çekim kuvveti, iki nesnenin kütlelerinin çarpımı ile doğru orantılıdır; fakat onlar arasındaki uzaklığın karesi ile ters orantılıdır.

$$F = \frac{GMm}{r^2} \tag{2.15}$$



Şekil 2.1. B ve D sınıflarının iyileştirilmiş merkezleri



Şekil 2.2. A ve B arasındaki S_0 çekimsel denge noktası

Farz edelim ki M_A ve M_B sırasıyla A ve B nesnelerinin kütleleri olsun. A ve B nesnelerinin ortasında bulunan S nesnesinin kütlesi de m olsun. r_A ve r_B sırasıyla S nesnesinin A ve B nesnelerine olan uzaklıklarıdır. S nesnesi ile A ve B nesneleri arasındaki çekim kuvvetide sırasıyla F_A ve F_B dir. Şekil 2.2.'de görüldüğü gibi çekim yasasına göre A ve B nesnelerini birleştiren doğru üzerinde S_0 Lagrange noktası bulunmaktadır. Bu noktada F_A ve F_B çekim kuvvetleri birbirine eşittir. Eğer S nesnesi, S_0 noktasının solunda bulunursa B nesnesinin S nesnesine uyguladığı çekim kuvveti A nesnesinin S nesnesine uyguladığı çekim kuvvetinden küçük olur. Yani $F_A > F_B$ dir. Bu sebeple S nesnesi A nesnesine doğru hareket edecektir. Tersi olursa B nesnesine doğru hareket edecektir.

A ve B nesneleri, A ve B sınıfları gibi düşünülebilir. S nesnesi de vektör uzayındaki $\vec{d}$ örneği gibi kabul edilebilir. $\vec{d}$ örneğinin hangi sınıfa ait olacağı, A ve B sınıflarının çekim kuvvetleriyle hesaplanabilir. Eğer $\vec{d}$ örneği Lagrange S_0 noktasının solunda kalırsa $\vec{d}$ örneği $F_A > F_B$ olduğundan A sınıfı olarak sınıflandırılacaktır. Eğer $\vec{d}$ örneği Lagrange S_0 noktasının sağında kalırsa $\vec{d}$ örneği $F_B > F_A$ olduğundan B sınıfı olarak sınıflandırılacaktır. Çekim modelindeki sınıflandırma hiper düzlemi, merkez tabanlı sınıflandırıcıda olduğu gibi A ve B sınıflarının sınıflandırılacak $\vec{d}$ örneğine aynı benzerliği paylaşan hiper düzlemden farklıdır. Vektör uzayında iki sınıf arasındaki Lagrange hiper düzlemi, Denklem 2.16'daki gibidir.

$$\frac{M_A}{r_A^2} = \frac{M_B}{r_B^2} \tag{2.16}$$

Denklem 2.16'da r_A ve r_B sırasıyla $\vec{d}$ örneğin, A ve B sınıflarının merkezlerine olan uzaklıkları ifade etmektedir. $\frac{1}{r_A^2}$ ve $\frac{1}{r_B^2}$, $\vec{d}$ örneği ile A ve B sınıfları arasındaki benzerlik ölçüsü gibi kabul edilebilir. Böylece Denklem 2.16, tekrar yazılırsa Denklem 2.17 gibi olacaktır.

$$M_A * Sim(\vec{d}, \overrightarrow{C_A}) = M_B * Sim(\vec{d}, \overrightarrow{C_B}) \tag{2.17}$$

$Sim(\vec{d}, \overrightarrow{C_A})$ ve $Sim(\vec{d}, \overrightarrow{C_B})$ sırasıyla, $\vec{d}$ örneği ile A ve B sınıflarının merkezleri arasındaki benzerlik değerleridir. M_A ve M_B ise eğitim setinden öğrenilebilen kütle değişkenidir. Uygun kütle değişkenini bulmak asıl amaçtır. Burada, sınıflandırma hatasını en küçük seviyeye getirmek için en uygun hiper düzlemi bulmak amaçlanır. Test aşamasında ise çekim modeli, $\vec{d}$ örneği ile her sınıfın merkezi ile olan çekim kuvveti Denklem 2.18'de gösterildiği gibi hesaplanır.

$$F(\vec{d}, \overrightarrow{C_j}) = M_j * Sim(\vec{d}, \overrightarrow{C_j}) \quad (2.18)$$

M_j, $j \in [1, |C|]$ olacak şekilde j. sınıfın kütle değişkenini temsil etmektedir. $\vec{d}$ örneğinin sınıfı ise Denklem 2.19'de görüldüğü gibi en yüksek çekim kuvveti hangi sınıfa aitse o sınıf olacaktır.

$$Sınıf(\vec{d}) = \arg\max_j \ F(\vec{d}, \overrightarrow{C_j}) \quad (2.19)$$

Çekim modeli tarafından sunulan kütle değişkeni, eğitim safhasında sınıf dağılımını belirlemede kullanılır. Devamında da sınıfı bilinmeyen örneklerin sınıfının ne olduğunun bulunmasında uygulanır.

Şekil 2.1.'de görüldüğü gibi büyük sınıf yani C_A sınıfı için onun örnekleri vektör uzayındaki geniş dağılımından dolayı hatalı bir şekilde komşu sınıflara atanacaktır. Sistem değerlendirme ölçülerinden Geri Çağırma (Recall) ölçüsüne göre başarı düşmüş olacaktır. C_A ve C_B gibi küçük sınıflara göre ise komşu sınıfların örnekleri, küçük dağılım alanından dolayı bu sınıfa atanırlar. Yine bu durum, sistem değerlendirme ölçülerinden Duyarlık (Precision) ölçüsünü düşürecektir. Sınıfları dengesiz olan veri setlerinde standart merkez tabanlı metot yukarda bahsedilen sebeplerden dolayı sistemin performansı zayıf olacaktır. Fakat bu sisteme kütle değişkeni dahil edilirse problem kolay bir şekilde çözülmüş olacaktır. Bir merkezin içerdiği bölge çekim modelinde eşit olarak genişleyecektir.

Çekim modelinde sistemin eğitim veri setine tam uyması için her sınıfın kütle değişkenini belirlemek en önemli konudur. Çekim modelinde şu şekilde kütle değişkeninin eğitim veri setindeki verilerle öğrenme işlemi gerçekleşir. Hatalı sınıflandırılan örneklerin sayısı sıfıra ya da çok küçük bir değere düşene kadar her sınıfın kütle değişkeni hatalı sınıflandırılan örnekleri kullanarak güncelleştirilir. Kütle değişkeninin öğrenmesi ya da güncelleştirilmesi iki şekilde yapılmaktadır. Birincisi olasılıksal öğrenme, diğeri ise toplu-güncelleme öğrenmesidir. Her ikisinin de zaman karmaşıklığı doğrusaldır.

2.4.1 Olasılıksal Öğrenme Algoritması (OÖA)

Bu algoritmayla yanlış sınıflandırılan her örnek için iki kütle değişkeni güncelleştirilir. Örneğin, $\overrightarrow{C_n}$ sınıfında olan $\overrightarrow{d_n}$ dokümanı $\overrightarrow{C_j}$ sınıfına yanlış olarak sınıflandırılmış olsun. $\overrightarrow{d_n}$ örneğinin doğru sınıflandırılabilmesi için $\overrightarrow{d_n}$ üzerindeki $\overrightarrow{C_n}$ sınıfının çekim kuvvetinin artırılması gereklidir. Bununla birlikte $\overrightarrow{d_n}$ örneği ve

$\overrightarrow{C_n}$ sınıfı arasındaki çekim kuvveti de azaltılır. Bu sebeple M_n kütle değişkeni artırılmalıyken M_j kütle değişkeni ise azaltılmalıdır. Denklem 2.20 ve 2.21'de nasıl güncelleştirileceği gösterilmektedir. Buradaki ξ, güncelleştirme kuvvetini kontrol edebilmek için kullanılan ve kullanıcı tarafından belirlenen bir sabit değerdir.

$$M_n := M_n + \xi \tag{2.20}$$

$$M_j := M_j - \xi \tag{2.21}$$

2.4.1.1 Olasılıksal Öğrenme Algoritması

X: P sınıflı ve N örnekli bütün eğitim seti. $\overrightarrow{d_j^n}$, $\overrightarrow{C_j}$ sınıfındaki $n.$ örneği temsil etmektedir.

$\overrightarrow{c_j}$: C_j sınıfının merkezidir.

ξ: Kütle değişkeni güncelleştirmedeki adım uzunluğudur.

ε: Sınıflandırma hata oranının eşik değeridir.

MI: Maksimum iterasyon sayısıdır.

M_j: C_j sınıfının kütle değişkenidir. $1 \leq j \leq P$

1: **for** $j = 1$; $j \leq P$ **do**

2: $M_j = 1$;

3: end for

4: $R^{yeni} = 1$;

5: Repeat $a++$;

6: $hata = 0$;

7: $R^{eski} = R^{yeni}$;

8: **for** $n = 1$; $n \leq N$ **do**

9: $\tilde{\jmath} = \arg\max_{k=1,\dots,P} M_k * Sim(\overrightarrow{d_j^n}, \overrightarrow{c_k})$;

10: if $j \neq \tilde{j}$ then

11: $M_j = M_j + \xi$;

12: $M_{\tilde{j}} = M_{\tilde{j}} - \xi$;

13: $hata++$;

14: end if

15: end for

16: $R^{yeni} = hata/N$;

17: **Until** $\varepsilon > \frac{R^{eski} - R^{yeni}}{R^{eski}}$ ***or*** $a > MI$;

18: **Return** M_j $1 \leq j \leq P$;

Çekim Modeli eğitim sınıfındaki örnekleri birer birer sınıflandırmak için kullanılır. Yanlış sınıflandırılan örnek için iki iterasyon arasındaki sınıflandırma hatasının farkı belirli bir eşik değerin (ε) altında olana kadar ya da maksimum iterasyona kadar kütle çarpanları güncellenir. Yani bir sınıfın kapsadığı alan genişletilir ya da daraltılır.

2.4.2 Toplu Öğrenme Algoritması (TÖA)

Toplu öğrenme algoritması, olasılıksal öğrenme algoritmasından farklıdır. Olasılıksal öğrenme algoritması her örnek için adım adım kütle çarpanları güncellenir. Toplu öğrenme algoritmasında ise kütle çarpanları eğitim setindeki bütün örnekler sınıflandırıldıktan sonra güncelleme yapılır. C_j sınıfının M_j kütle çarpanı Denklem 2.22'de görüldüğü gibi güncellenir.

$$M_j := M_j + \left(\beta_{j1} - \beta_{j2}\right) * \xi \tag{2.22}$$

Denklem 2.22'de gösterilen β_{j1}, C_j sınıfı etiketli dokümanların sayısıdır. Fakat bu dokümanlar yanlış sınıflandırılarak başka sınıflara sınıflandırılmışlardır. β_{j2} ise C_j sınıfına yanlış sınıflandırılan dokümanların sayısıdır.

2.5. Naive Bayes Sınıflandırıcı (NBS)

Bir sınıflandırıcı, farklı nesneleri belli özelliklere göre ayırmak için kullanılan makine öğrenmesi modelidir.

Bir Naive Bayes sınıflandırıcısı (McCallu ve Nigam, 1998; Kibriya vd., 2004) sınıflandırma işi için kullanılan olasılıksal bir makine öğrenmesi modelidir. Sınıflandırıcının mantığı Bayes teoremine dayanır.

2.5.1 Bayes Teoremi

$$P(A|B) = \frac{P(B|A).P(A)}{P(B)} \tag{2.23}$$

Denklem 2.23'de görüldüğü gibi Bayes teoremini kullanarak, *B* olayı gerçekleştikten sonra *A*'nın gerçekleşme olasılığını bulabiliriz. Burada *B* kanıt ve *A* hipotezdir. Buradaki var olan öznitelikler birbirinden bağımsızdır. Yani birinin varlığı diğerlerini etkilemez. Bu yüzden naif (naive) adı verilir.

Daha iyi kavrayabilmek için bir örnek ele alalım. Futbol oynama problemine bir bakalım veri seti Çizelge 2.1.'de verildiği gibi olsun.

Çizelge 2.1. Hava durumuna göre futbol oynanabilirliği

	Hava Durumu	**Sıcaklık**	**Nem**	**Rüzgar**	**Futbol Oynanabilirlik**
1	Yağmurlu	Sıcak	Yüksek	Yok	Hayır
2	Yağmurlu	Sıcak	Yüksek	Var	Hayır
3	Açık	Sıcak	Yüksek	Yok	Evet
4	Güneşli	Ilıman	Yüksek	Yok	Evet
5	Güneşli	Serin	Normal	Yok	Evet
6	Güneşli	Serin	Normal	Var	Hayır
7	Açık	Serin	Normal	Var	Evet
8	Yağmurlu	Ilıman	Yüksek	Yok	Hayır
9	Yağmurlu	Serin	Normal	Yok	Evet
10	Güneşli	Ilıman	Normal	Var	Evet

O günün verilen özelliklerine göre, futbol oynamak için uygun bir gün olup olmadığının sınıflandırmasını yapacağız. Sütunlar öznitelikleri, satırlar ise öznitelik değerlerini göstermektedir. Veri setinin ilk satırını ele aldığımızda hava tahmininin yağmurlu, sıcaklığın ve nemin yüksek, havanın rüzgarlı olmamasından dolayı futbol oynamak için uygun olmadığını gözlemleriz. Burada iki tane kabulümüz var, birincisi yukarıda vurgulandığı gibi bu öznitelikler birbirlerinden bağımsızdır. Yani, havanın sıcak olması nemin yüksek olacağı anlamına gelmez. Diğer kabulümüz ise tüm bu özniteliklerin sonuca eşit oranda etki etmesidir. Yani o gün rüzgarlı olmasının futbol oynama/oynamama kararına diğerlerine göre daha fazla etkisi olamaz.

Bu örneğe göre, Bayes teoremi aşağıdaki gibi yeniden yazılabilir.

$$P(y|X) = \frac{P(X|y).P(y)}{P(X)} \tag{2.24}$$

y değişkeni verilen koşullara göre futbol oynamaya veya oynamamaya müsait olup olmadığını belirten sınıf değişkenidir. X değişkeni parametreleri gösterir.

$$X = (x_1, x_2, x_3, \dots, x_n) \tag{2.25}$$

Denklem 2.25'te $x_1, x_2, x_3, \dots, x_n$ X'in parametrelerini göstermektedir. Yani hava durumu, sıcaklık, nem ve rüzgar gibi öznitelikler. X'i yerine yazıp açarsak zincir kuralını kullanarak Denklem 2.26 elde edilir.

$$P(y|, x_2, x_3, \dots, x_n) = \frac{P(x_1|y)P(x_2|y)\dots P(x_n|y)P(y)}{P(x_1)P(x_2)\dots P(x_n)} \tag{2.26}$$

Şimdi veri setine bakarak ve denklemde yerine yazarak her biri için değerler elde edebiliriz. Veri setindeki tüm girdiler için payda değişmeden kalır. Bu yüzden Denklem 2.27'deki gibi payda kaldırılabilir.

$$P(y|, x_2, x_3, \dots, x_n) \approx P(y) \prod_{i=1}^{n} P(x_i|y) \tag{2.27}$$

Örneğe göre sınıfımızın iki çıktısı var; biri evet diğeri hayır. Yani iki sınıf vardır. Fakat çok sınıflı problemlerde olabilirdi. Bu sebepten dolayı buradaki y sınıfının maksimum olasılığını hesaplamak gerekir. Aşağıdaki denklemde görüldüğü gibi en yüksek olasılığa sahip sınıf bulunabilir.

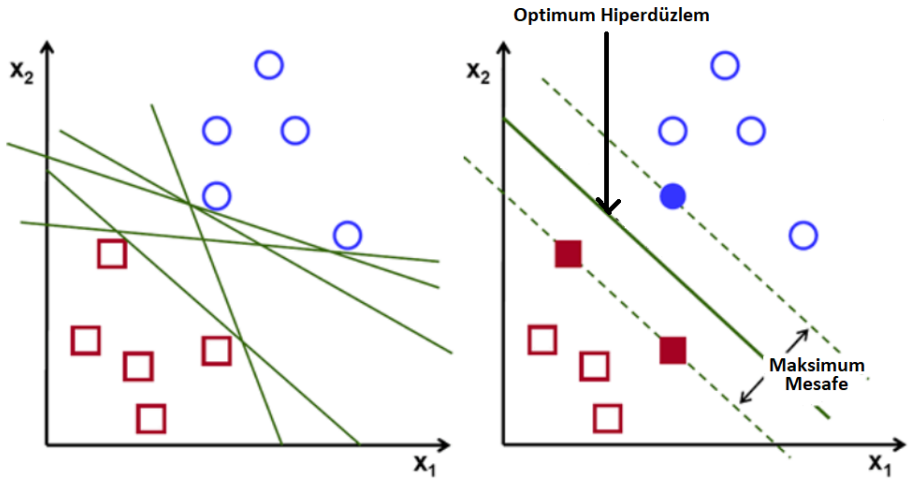
$$y = arg\, max_y\, P(y) \prod_{i=1}^{n} P(x_i|y) \tag{2.28}$$

2.6. Destek Vektör Makinesi (DVM)

Destek Vektör Makinesi, hem sınıflandırma hem de regresyon problemlerinde kullanılan yüksek doğruluk oranına sahip bir metottur (Chen ve Hsieh, 2006; Zhang vd., 2006). Fakat genelde sınıflandırma amacıyla kullanılmaktadır.

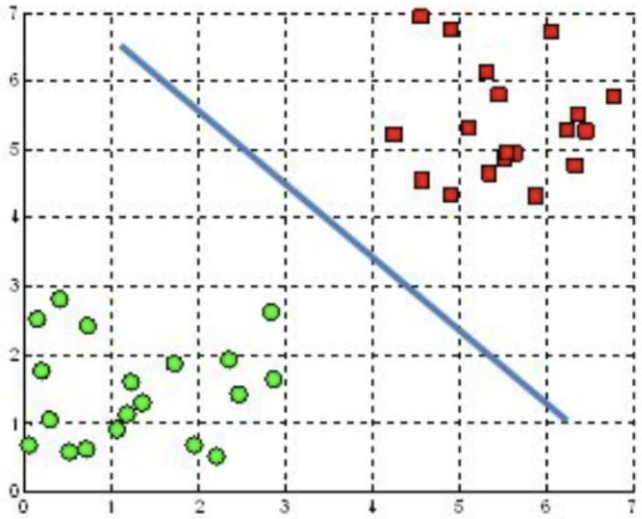
Destek vektörü makinesinin amacı, bir N-boyutlu uzaydaki örnekleri doğru bir şekilde sınıflandıran hiper düzlem bulmaktır.

Şekil 2.3.'te görüldüğü gibi örneklerden oluşan iki sınıfı vektör uzayında birbirinden ayırmak için, seçilebilecek birçok olası hiper düzlem vardır. Amaç buradaki hiper düzlemlerden en uygununu yani her iki sınıfa en uzak (maksimum mesafede) olan bir hiper düzlemi bulmaktır. Bu şekilde sınıflar daha iyi ayrılır ve bu sınırda olabilecek örnekler daha doğru bir şekilde sınıflandırılırlar.

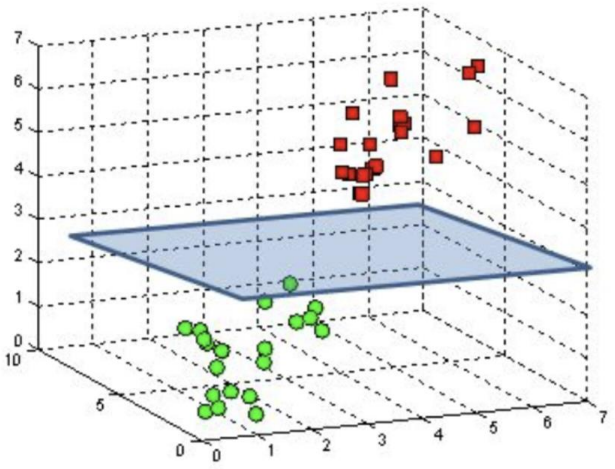


Şekil 2.3. Destek Vektör Makinesi hiper düzlemleri

2.6.1. Hiper Düzlemler ve Destek Vektörleri



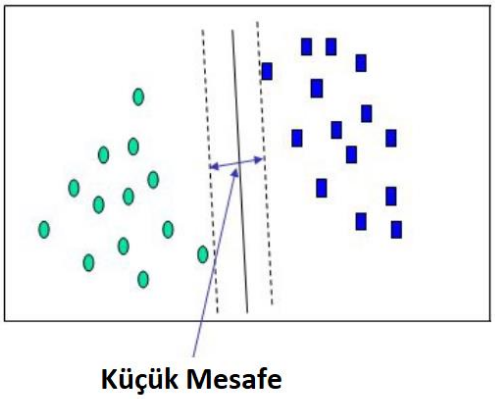
Şekil 2.4. $\mathbb{R}^2$'de hiper düzlem bir doğrudur



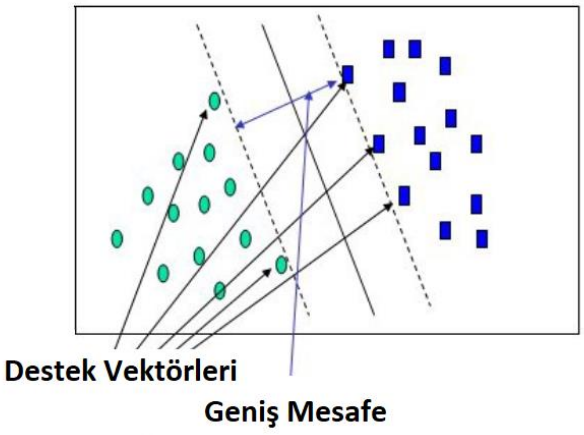
Şekil 2.5. $\mathbb{R}^3$'de hiper düzlem bir düzlemdir

Hiper düzlemler, örneklerin sınıflandırılmasına yardımcı olan karar verici sınırlardır. Hiper düzlemin diğer tarafına düşen örnekler farklı bir kategori olarak

sınıflandırılır. Ayrıca, hiper düzlemin boyutu, örneklerin özniteliklerinin sayısına bağlıdır. Şekil 2.4.'te gösterildiği gibi örneklerin özniteliklerinin sayısı 2 olduğunda, hiper düzlem sadece bir çizgi olacaktır. Fakat Şekil 2.5.'te görüldüğü gibi örneklerin özniteliklerinin sayısı 3 ise, hiper düzlem iki boyutlu bir düzlem haline gelir. Özniteliklerinin sayısı 3'ü geçtiğinde hiper düzlemleri tasavvur etmek biraz zorlaşmaktadır.



Şekil 2.6. Destek vektörleri (Küçük mesafe)



Şekil 2.7. Destek vektörleri (Geniş mesafe)

Şekil 2.6. ve 2.7.'de görüldüğü gibi destek vektörleri, hiper düzlemine daha yakın olan ve hiper düzlemin konumunu ve yönünü etkileyen örneklerdir. Bu destek vektörleri kullanılarak, sınıflandırıcının sınıflar arası mesafesi maksimize edilir. Destek vektörlerini kaldırmak, hiper düzlemin konumunu değiştirir. Bu örnekler, Destek Vektör Makinelerinin modellenmesinde çok önemli rol oynarlar.

Lojistik regresyonda, bir lineer fonksiyonun çıktısı alındıktan sonra bu değeri sigmoid fonksiyonundan geçirdiğimizde çıktıyı [0,1] aralığına dönüştürmüş oluruz. Eğer aldığımız değer bir eşik değerden (Örneğin 0,5 eşik değerinden) büyükse, ona etiket ya da sınıf olarak 1 atanır, aksi takdirde ona etiket olarak 0 atanır. DVM'de doğrusal fonksiyonun çıktısı alındıktan sonra eğer bu çıktı 1 ise, 1. sınıftan olduğunu; eğer çıktı -1 ise, 2. sınıftan olduğunu tespit ederiz.

DVM algoritmasında, örnekler ve hiper düzlem arasındaki mesafenin maksimize edilmesi istenir. Mesafeyi maksimize etmeye yardımcı olan hata fonksiyonu Denklem 2.29 ve 2.30'da gösterilmektedir.

$$c\big(\vec{x}, y, f(\vec{x})\big) = \begin{cases} 0, & if\ y * f(\vec{x}) \geq 1 \\ 1 - y * f(\vec{x}), & else \end{cases} \tag{2.29}$$

$$c\big(\vec{x}, y, f(\vec{x})\big) = (1 - y * f(\vec{x}))_+ \tag{2.30}$$

Sistemin tahmin ettiği değer ile gerçek değer aynı ise, hata 0 olacaktır. Aynı değilse, hata değeri hesaplanır. Hata fonksiyonuna bir düzenleyici parametre (λ) de eklenir. Bu düzenleyici parametrenin amacı, mesafe maksimizasyonunu ve kaybını dengelemektir. Düzenleyici parametresini ekledikten sonra, hata fonksiyonları Denklem 2.31'deki gibi görünür.

$$\min_w \lambda\|\vec{w}\|^2 + \sum_1^n (1 - y_i \langle \vec{x_i}, \vec{w} \rangle)_+ \tag{2.31}$$

Şimdi hata fonksiyonunu elde ettikten sonra, gradyanları bulmak için ağırlıklara göre kısmi türevler alıyoruz.

$$\frac{\delta}{\delta w_k} \lambda\|\vec{w}\|^2 = 2\lambda w_k \tag{2.32}$$

$$\frac{\delta}{\delta w_k} (1 - y_i \langle \vec{x_i}, \vec{w} \rangle)_+ = \begin{cases} 0, & if\ y_i \langle \vec{x_i}, \vec{w} \rangle \geq 1 \\ -y_i x_{ik} & else \end{cases} \tag{2.33}$$

Herhangi bir yanlış sınıflandırma olmadığında, yani modelimiz veri noktamızın sınıfını doğru bir şekilde tahmin ederse, sadece gradyanı normalleştirme parametresinden güncellememiz gerekir.

$$\vec{w} = \vec{w} - \alpha \cdot (2\lambda\vec{w}) \tag{2.34}$$

Yanlış bir sınıflandırma olduğunda, yani modelimiz veri noktamızın sınıfının öngörüsünde bir hata yaparsa, gradyan güncellemesi yapmak için normalizasyon parametresi ile birlikte hatayı da ekleriz.

$$\vec{w} = \vec{w} + \alpha \cdot (y_i \cdot \vec{x_i} - 2\lambda\vec{w}) \tag{2.35}$$

2.7. K-En Yakın Komşu Metodu (KEYK)

Uygulaması basit ve kolay bir algoritmadır. Öğretmenli öğrenme yaklaşımına sahip, sınıflandırma ve regresyon problemlerinde kullanılabilen bir algoritmadır. Öğretmenli öğrenme yaklaşımında sisteme hem girdi verilerini hem de çıktı sınıfını vererek öğrenme gerçekleştirilir. Öğretmenli öğrenme algoritmaları sınıflandırma ve regresyon problemlerini çözebilen algoritmalardır. Bir sınıflandırma problemi bir sistemin çıktısının ayrık (sürekli olmayan) değerine sahiptir.

Örneğin, Çizelge 2.2.'de görüldüğü gibi lahmacun sevenler ve lahmacun sevmeyenlerin verileri ayrık ifadeler şeklinde verilmiştir. Burada iki değer vardır. Bunun dışında başka değer yoktur.

Çizelge 2.2. Lahmacun sevenler ve sevmeyenler veri seti örneği

Yaş	**Lahmacun Sevenler (1) ve Sevmeyenler (0)**
30	1
25	1
60	0
12	0
70	1
7	1

Çizelge 2.2.'de görüldüğü gibi yaşa göre lahmacun sevenler (1) ve sevmeyenler (0) listelenmektedir. Bu değerler, sınıflandırmaya bir örnek olabilir. Bu sınıflandırmaya göre belirli bir yaştaki kişinin lahmacun sevip sevmediği tahmin edilebilir.

Sınıflandırma problemlerinde sınıflar genellikle etiket değeri olarak 1, -1 ya da 0 gibi tamsayı değeri kullanılır. Bunlar üzerine matematiksel işlemler yapılmaz çünkü matematiksel işlemlerde anlamsız olur. Regresyon problemlerinde ise sistemin çıktısı bir reel sayıya eşit olur. Ayrık bir değere sahip değil (etiket)

Çizelge 2.3. İnsanların boyları ve ayak uzunlukları veri seti

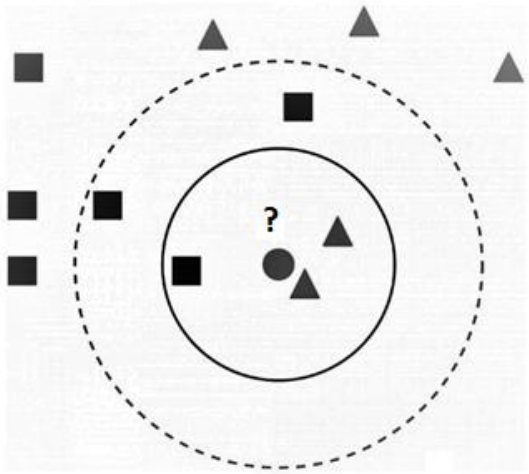
Boy (cm)	**Ayak Uzunluğu (cm)**
1,11	15
1,60	25
1,94	26
1,60	23
1,52	23,5
1,26	20
1,50	24

Örneğin, Çizelge 2.3.'de bir kişinin boyuna bağlı olarak ayak uzunluğu göstermektedir. Dolayısıyla bu çizelgeye göre herhangi bir boyda olan kişinin ayak uzunluğunun ne olacağı tahmin edilebilir. Regresyon problemi bu şekildedir.

Öğretmensiz öğrenme algoritmaları çıktı olmaksızın girdi verilerini kullanır. Yani sistemi eğitecek bir öğretmen yoktur. Bu yaklaşımda benzer veriler özelliklerine ve metriğe göre gruplandırmaktadır.

KEYK (Shakhnarovich vd., 2008) algoritması sınıflandırma metotlarındandır. KEYK, "Benzer veriler birbirine yakındır" mantığına dayanmaktadır. Benzerlikler, verilerin arasındaki uzaklıkların hesaplanmasıyla bulunur. En çok kullanılan metrik Öklid metriğidir.

Şekil 2.8.'de üçgen ve kare sınıfları olmak üzere iki sınıf bulunmaktadır. KEYK metoduna göre hangi sınıfta olduğu bilinmeyen yuvarlak örneği sınıflandırılmak istenirse $K = 3$ alındığında üçgen diye sınıflandırılacaktır. Çünkü yuvarlak örneğine en yakın üç örnekten biri kare, diğer ikisi üçgen sınıfındandır. Çoğunluk yaklaşımına göre üçgen sınıfında olacaktır. Fakat $K = 5$ alındığında ise kare olarak sınıflandırılacaktır. Bu durumda en yakın beş örnekten üçü kare sınıfında olduğundan çoğunluk yöntemine göre kare diye sınıflandırılıyor.



Şekil 2.8. KEYK örneği

2.7.1. KEYK Algoritması

1-Verilerin özelliği niteliğindedir.

2-Komşuluk için *K* değerini belirler.

3-Eğitim veri setindeki her örnek için;

3.1-Örnek veriyle bütün veriler arasındaki değerleri hesaplar.

3.2-Bütün veriler için uzaklıklar dizisini oluşturur.

4-Uzaklıklar dizisinin küçükten büyüğe doğru sırala

5-Sıralı diziden ilk *K* örneği al.

6-Seçilmiş *K* örneği etiketlerini (sınıflarını) al.

7-Eğer sınıflandırma yapılacaksa *K* etiketten en çok hangi sınıfa aitse o sınıfa ait olacaktır.

8-Eğer regresyon yapılacaksa bulunan *K* tane değerin ortalaması örneğimizin değeri olacaktır.

Bu metotta *K* seçimi çok önemlidir. *K*'yı doğru bir şekilde seçmek için farklı *K* değerleri KEYK algoritması birkaç kez çalışmamız gerekir. En düşük hata oranına sahip olan sistemdeki *K*'yı seçeriz.

2.7.2. KEYK Algoritmasının Avantajları

1-Basit ve uygulanması kolaydır.

2-Bir model oluşturmaya birçok parametre ayarlanmaya gerek duyulmaz.

3-Hem sınıflandırma hem de regresyon içinde kullanılabilir.

2.7.3. KEYK Algoritmasının Dezavantajları

1-Eğitim setindeki örnekler çok fazla olursa KEYK algoritmasının tahmin işlemi çok uzun zaman alır.

2.8. Terim-Frekansı Ters-Doküman-Frekansı (TF-IDF)

TF-IDF ağırlığı, sıklıkla yapay zekâ konularında kullanılmaktadır. Bu ağırlık, bir sözcüğün veri setindeki bir doküman için ne kadar önemli olduğunu tespit etmede kullanılan istatistiksel bir ölçüdür. Bu ağırlığın önemi bir sözcüğün kendi sınıfındaki dokümanlardaki frekansıyla doğru orantılı olarak artarken başka sınıflarda bulunan dokümanlardaki frekansıyla ters orantılıdır. Bu ağırlıklar web sayfalarının araştırılmasında kullanılan arama motorlarında da çok kullanılmaktadır. Verilen bir sorgunun, bir web sayfasının içeriği ile ne kadar ilgili olduğunu belirler. En basit sıralama fonksiyonlarından biri her bir sorgu teriminin TF-IDF ağırlığının toplanmasıyla hesaplanır. Bu ağırlıklar, metin özetleme ve sınıflandırma dâhil bir çok alanda etkisiz kelimelerin filtrelenmesinde başarıyla kullanılır.

TF-IDF ağırlığı iki terimden oluşmaktadır. Birincisi normalize edilmiş terim frekansını hesaplar. İkinci terim ise ters doküman frekansıdır. Yani veri setindeki dokümanların sayısının, sözcüğün göründüğü dokümanların sayısına bölünüp logaritması alınarak hesaplanır. Aşağıda s sözcüğünün TF ve IDF değerlerinin nasıl hesaplandığı verilmiştir.

TF(s) = (s sözcüğünün bir dokümandaki frekansı) / (Doküman içindeki toplam sözcük sayısı)

IDF(s) = $\log_e$(Toplam doküman sayısı / s sözcüğünü içeren dokümanların sayısı)

3. BULGULAR VE TARTIŞMA

Bu bölümde merkez tabanlı metotlarla birlikte KEYK, Naive Bayes ve Destek Vektör Makinesi sınıflandırma metotlarıyla yapılan uygulamaların deneysel sonuçları verilmiştir ve bu sonuçlar tartışılmıştır. Uygulamaların gerçekleştirildiği veri setleri de performans ölçüleri de bu bölümde anlatılmıştır.

3.1. Veri Setleri

Bu tezde yapılan uygulamalar Çizelge 3.1.'de görüldüğü gibi dört farklı veri seti üzerinde gerçekleştirilmiştir. Bütün veri setleri dengelidir. Veri seti1 (VS1), gazetelerin web sayfalarından elde edilmiş ve 6 sınıftan ("Eğitim", "Ekonomi", "Kültür-Sanat", "Otomobil", "Sağlık", "Spor") oluşmaktadır. Her bir sınıf 75 dokümandan oluşmaktadır. Diğer veri setleri ise Yıldız Teknik Üniversitesinin veri setlerinden elde edilmiştir. Veri seti2 (VS2), veri seti3 (VS3) ve veri seti4 (VS4) (Anonim, 2019) sırasıyla 7, 9 ve 11 sınıftan oluşmaktadır ve her bir sınıf içinde 600 doküman bulunmaktadır. Veri setlerinin içindeki sözcük bilgileri aşağıda gösterilmiştir. VS2, VS3 ve VS4 sırasıyla ("Ekonomi", "Kültür-Sanat", "Magazin", "Sağlık", "Siyaset", "Spor", "Teknoloji"), ("Dünya", "Ekonomi", "Kültür-Sanat", "Magazin", "Sağlık", "Siyaset", "Spor", "Teknoloji", "Yaşam") ve ("Dünya", "Ekonomi", "Genel", "Güncel", "Kültür-Sanat", "Magazin", "Sağlık", "Siyaset", "Spor", "Teknoloji", "Yaşam") sınıflarını içermektedir.

Çizelge 3.1. Uygulamalarda kullanılan veri setleri

Veri Setleri	Sınıf Sayısı	Toplam Doküman Sayısı	Farklı Sözcük Sayısı	Toplam Sözcük Sayısı
VS1	6	450	71884	497842
VS2	7	4200	111953	1037946
VS3	9	5400	125478	1247258
VS4	11	6600	141813	1527203

3.2. Performans Ölçüleri

Metin sınıflandırma uygulamalarının başarılarını değerlendirmek için Doğruluk (Accuracy) ve F1-Ölçüsü (Rijsbergen, 1979) kullanılmıştır. F1-Ölçüsü, aşağıdaki Denklem 3.2 ve 3.3'de görüldüğü gibi Geri Çağırma (Recall) ve Duyarlık (Precision) değerlendirme ölçüleri bir denklemde birleştirilerek hesaplanmaktadır.

$$\text{Doğruluk} = \frac{Doğru\ tahminlerin\ sayısı}{Bütün\ örneklerin\ sayısı} \tag{3.1}$$

$$Geri\ Çağırma = \frac{Doğru\ pozitif\ tahminlerin\ sayısı}{Pozitif\ örneklerin\ sayısı} \tag{3.2}$$

$$\text{Duyarlık} = \frac{Doğru\ pozitif\ tahminlerin\ sayısı}{Pozitif\ tahminlerin\ sayısı} \tag{3.3}$$

$$F1 - Ölçüsü = \frac{2 \times Geri\ Çağırma \times \text{Duyarlık}}{(Geri\ Çağırma + \text{Duyarlık})} \tag{3.4}$$

Denklem 3.4'te F1-Ölçüsünün hesaplanması gösterilmektedir. Farklı sınıflar için F1-Ölçüleri, F1-Ölçülerinin Makro ve Mikro ortalamaları alınarak genelleştirilir (Lewis vd., 1996).

Yapılan uygulamalarda sistemlerin performanslarını değerlendirmede daha doğru ve objektif sonuçlar elde edilmesi için K-Katman Çapraz-Doğrulama (K-Fold Cross-Validation) kullanılmıştır. Buradaki K değeri uygulamalarımızda $K = 5$ seçilmiştir. K-Katman Çapraz-Doğrulamada bütün veri seti 5 eşit parçaya bölünür ve ilk parça test veri seti için diğer dört parçası da eğitim veri seti için kullanılır ve sistemin başarısı ölçülür. Sonra parçalara bölünmüş veri setinin ikinci parçası test veri seti için ve diğer parçalar ise eğitim veri seti için kullanılır. Aynı şekilde sistemin tekrar başarısı ölçülür. Benzer şekilde 3. 4. ve 5. parçalar için de test veri seti olarak başarıları hesaplanır.

Beş farklı test veri setinin başarılarının ortalaması sistemin başarısı olarak kabul edilmektedir.

Çizelge 3.2. VS1 için Standart Merkez Tabanlı Metot sonuçları

		Öznitelik Boyutu					
		500	1000	5000	10000	20000	30000
Eğitim	Doğruluk	0,972	0,98	0,991	0,997	0,998	**0,999**
	F1-Ölçüsü-Mikro	0,972	0,98	0,991	0,997	0,998	**0,999**
	F1-Ölçüsü-Makro	0,972	0,98	0,991	0,997	0,998	**0,999**
Test	Doğruluk	0,944	0,962	**0,971**	0,969	0,967	0,967
	F1-Ölçüsü-Mikro	0,944	0,962	**0,971**	0,969	0,967	0,967
	F1-Ölçüsü-Makro	0,943	0,961	**0,97**	0,967	0,965	0,965
Toplam Süre (Saniye)		0,87	0,81	0,98	1,09	1,1	1,31

Çizelge 3.3. VS2 için Standart Merkez Tabanlı Metot sonuçları

		Öznitelik Boyutu					
		500	1000	5000	10000	20000	30000
Eğitim	Doğruluk	0,788	0,844	0,908	0,921	0,932	**0,937**
	F1-Ölçüsü-Mikro	0,788	0,844	0,908	0,921	0,932	**0,937**
	F1-Ölçüsü-Makro	0,788	0,844	0,909	0,921	0,932	**0,937**
Test	Doğruluk	0,759	0,815	0,869	0,877	**0,884**	**0,884**
	F1-Ölçüsü-Mikro	0,759	0,815	0,869	0,877	**0,884**	**0,884**
	F1-Ölçüsü-Makro	0,76	0,815	0,869	0,877	**0,884**	**0,884**
Toplam Süre (Saniye)		2,81	2,8	3,69	4,27	5,65	6,71

Çizelge 3.4. VS3 için Standart Merkez Tabanlı Metot sonuçları

		Öznitelik Boyutu					
		500	1000	5000	10000	20000	30000
Eğitim	Doğruluk	0,722	0,779	0,847	0,867	0,883	**0,891**
	F1-Ölçüsü-Mikro	0,722	0,779	0,847	0,867	0,883	**0,891**
	F1-Ölçüsü-Makro	0,721	0,778	0,846	0,866	0,882	**0,89**
Test	Doğruluk	0,697	0,744	0,8	0,808	0,814	**0,816**
	F1-Ölçüsü-Mikro	0,697	0,744	0,8	0,808	0,814	**0,816**
	F1-Ölçüsü-Makro	0,696	0,743	0,798	0,806	0,812	**0,814**
Toplam Süre (Saniye)		3,49	3,6	4,56	5,56	7,03	8,92

Çizelge 3.5. VS4 için Standart Merkez Tabanlı Metot sonuçları

		Öznitelik Boyutu					
		500	1000	5000	10000	20000	30000
Eğitim	Doğruluk	0,6	0,659	0,737	0,767	0,79	**0,802**
	F1-Ölçüsü-Mikro	0,6	0,659	0,737	0,767	0,79	**0,802**
	F1-Ölçüsü-Makro	0,59	0,651	0,729	0,759	0,783	**0,796**
Test	Doğruluk	0,559	0,61	0,665	0,678	0,683	**0,686**
	F1-Ölçüsü-Mikro	0,559	0,61	0,665	0,678	0,683	**0,686**
	F1-Ölçüsü-Makro	0,549	0,601	0,655	0,666	0,672	**0,674**
Toplam Süre (Saniye)		4,25	4,4	5,62	6,88	8,71	11,08

Çizelge 3.6. VS1 için Merkez Tabanlı Yaklaştır-Uzaklaştır Metodu sonuçları (Maksimum iterasyon=13, Öğrenme Katsayısı=0,2)

		Öznitelik Boyutu					
		500	1000	5000	10000	20000	30000
Eğitim	Doğruluk	**1**	**1**	**1**	**1**	**1**	**1**
	F1-Ölçüsü-Mikro	**1**	**1**	**1**	**1**	**1**	**1**
	F1-Ölçüsü-Makro	**1**	**1**	**1**	**1**	**1**	**1**
Test	Doğruluk	0,962	0,969	**0,984**	0,98	0,976	0,969
	F1-Ölçüsü-Mikro	0,962	0,969	**0,984**	0,98	0,976	0,969
	F1-Ölçüsü-Makro	0,961	0,968	**0,984**	0,979	0,974	0,968
Toplam Süre (Saniye)		0,92	0,94	1,22	1,48	1,73	2,04

Çizelge 3.7. VS2 için Merkez Tabanlı Yaklaştır-Uzaklaştır Metodu sonuçları (Maksimum iterasyon=8, Öğrenme Katsayısı=0,01)

		Öznitelik Boyutu					
		500	1000	5000	10000	20000	30000
Eğitim	Doğruluk	0,971	0,994	0,999	**0,999**	**0,999**	**0,999**
	F1-Ölçüsü-Mikro	0,971	0,994	0,999	**0,999**	**0,999**	**0,999**
	F1-Ölçüsü-Makro	0,971	0,994	0,998	**0,999**	**0,999**	**0,999**
Test	Doğruluk	0,775	0,847	0,893	0,897	**0,899**	**0,899**
	F1-Ölçüsü-Mikro	0,775	0,847	0,893	0,897	**0,899**	**0,899**
	F1-Ölçüsü-Makro	0,775	0,848	0,894	0,896	**0,899**	**0,899**
Toplam Süre (Saniye)		3,64	3,67	5,66	7,16	9,29	11,85

Çizelge 3.8. VS3 için Merkez Tabanlı Yaklaştır-Uzaklaştır Metodu sonuçları (Maksimum iterasyon=8, Öğrenme Katsayısı=0,01)

		Öznitelik Boyutu					
		500	1000	5000	10000	20000	30000
Eğitim	Doğruluk	0,928	0,992	0,998	0,998	**0,999**	**0,999**
	F1-Ölçüsü-Mikro	0,928	0,992	0,998	0,998	**0,999**	**0,999**
	F1-Ölçüsü-Makro	0,928	0,992	0,998	0,998	**0,999**	**0,999**
Test	Doğruluk	0,696	0,781	0,839	0,839	0,841	**0,842**
	F1-Ölçüsü-Mikro	0,696	0,781	0,839	0,839	0,841	**0,842**
	F1-Ölçüsü-Makro	0,695	0,78	0,839	0,838	0,84	**0,841**
Toplam Süre (Saniye)		5,71	5,85	8,12	10,4	13,55	17,92

Çizelge 3.9. VS4 için Merkez Tabanlı Yaklaştır-Uzaklaştır Metodu sonuçları (Maksimum iterasyon=8, Öğrenme Katsayısı=0,01)

		Öznitelik Boyutu					
		500	1000	5000	10000	20000	30000
Eğitim	Doğruluk	0,773	0,939	0,993	0,995	**0,996**	**0,996**
	F1-Ölçüsü-Mikro	0,773	0,939	0,993	0,995	**0,996**	**0,996**
	F1-Ölçüsü-Makro	0,772	0,938	0,993	0,995	**0,996**	**0,996**
Test	Doğruluk	0,529	0,622	0,706	0,716	0,722	**0,725**
	F1-Ölçüsü-Mikro	0,529	0,622	0,706	0,716	0,722	**0,725**
	F1-Ölçüsü-Makro	0,528	0,62	0,701	0,709	0,713	**0,715**
Toplam Süre (Saniye)		8,03	8,23	13,79	14,32	19,22	25,06

Çizelge 3.10. VS1 için Geniş-Mesafe Yaklaştır-Uzaklaştır Metodu sonuçları (Maksimum iterasyon=8, Öğrenme Katsayısı=0,2, MinMarjin=0,05, MarjinAğırlık=0,01)

		Öznitelik Boyutu					
		500	1000	5000	10000	20000	30000
Eğitim	Doğruluk	**1**	**1**	**1**	**1**	**1**	**1**
	F1-Ölçüsü-Mikro	**1**	**1**	**1**	**1**	**1**	**1**
	F1-Ölçüsü-Makro	**1**	**1**	**1**	**1**	**1**	**1**
Test	Doğruluk	0,962	0,967	**0,984**	0,98	0,976	0,969
	F1-Ölçüsü-Mikro	0,962	0,967	**0,984**	0,98	0,976	0,969
	F1-Ölçüsü-Makro	0,961	0,966	**0,984**	0,979	0,974	0,968
Toplam Süre (Saniye)		1,01	1,05	1,19	1,53	1,7	2,2

Çizelge 3.11. VS2 için Geniş-Mesafe Yaklaştır-Uzaklaştır Metodu sonuçları (Maksimum iterasyon=8, Öğrenme Katsayısı=0,01, MinMarjin=0,02, MarjinAğırlık=0,001)

		Öznitelik Boyutu					
		500	1000	5000	10000	20000	30000
Eğitim	Doğruluk	0,97	0,994	0,999	**0,999**	**0,999**	**0,999**
	F1-Ölçüsü-Mikro	0,97	0,994	0,999	**0,999**	**0,999**	**0,999**
	F1-Ölçüsü-Makro	0,97	0,994	0,998	**0,999**	**0,999**	**0,999**
Test	Doğruluk	0,775	0,847	0,893	0,897	**0,899**	**0,899**
	F1-Ölçüsü-Mikro	0,775	0,847	0,893	0,897	**0,899**	**0,899**
	F1-Ölçüsü-Makro	0,774	0,848	0,894	0,896	**0,899**	**0,899**
Toplam Süre (Saniye)		5,06	5,27	7,94	11	15,6	20,37

Çizelge 3.12. VS3 için Geniş-Mesafe Yaklaştır-Uzaklaştır Metodu sonuçları (Maksimum iterasyon=10, Öğrenme Katsayısı=0,01, Min Marjin=0,01, Marjin Ağırlık=0,001)

		Öznitelik Boyutu					
		500	1000	5000	10000	20000	30000
Eğitim	Doğruluk	0,94	0,995	**0,999**	**0,999**	**0,999**	**0,999**
	F1-Ölçüsü-Mikro	0,94	0,995	**0,999**	**0,999**	**0,999**	**0,999**
	F1-Ölçüsü-Makro	0,939	0,995	**0,999**	**0,999**	**0,999**	**0,999**
Test	Doğruluk	0,693	0,781	0,839	0,839	**0,842**	**0,842**
	F1-Ölçüsü-Mikro	0,693	0,781	0,839	0,839	**0,842**	**0,842**
	F1-Ölçüsü-Makro	0,692	0,78	0,839	0,838	**0,841**	**0,841**
Toplam Süre (Saniye)		7,25	7,61	11,56	15,7	21,71	29,32

Çizelge 3.13. VS4 için Geniş-Mesafe Yaklaştır-Uzaklaştır Metodu sonuçları (Maksimum iterasyon=10, Öğrenme Katsayısı=0,01, Min Marjin=0,01, Marjin Ağırlık=0,001)

		Öznitelik Boyutu					
		500	1000	5000	10000	20000	30000
Eğitim	Doğruluk	0,78	0,949	0,996	0,996	**0,997**	**0,997**
	F1-Ölçüsü-Mikro	0,78	0,949	0,996	0,996	**0,997**	**0,997**
	F1-Ölçüsü-Makro	0,778	0,95	0,996	0,996	**0,997**	**0,997**
Test	Doğruluk	0,523	0,616	0,706	0,717	0,722	**0,726**
	F1-Ölçüsü-Mikro	0,523	0,616	0,706	0,717	0,722	**0,726**
	F1-Ölçüsü-Makro	0,523	0,616	0,701	0,709	0,713	**0,716**
Toplam Süre (Saniye)		9,84	9,86	22,17	24,44	36,57	49,69

Çizelge 3.14. VS1 için Çekimsel Model Olasılıksal Öğrenme sonuçları (Maksimum iterasyon=100, Adım Gücü=0,01, Eşik Değer=0,0001)

		Öznitelik Boyutu					
		500	1000	5000	10000	20000	30000
Eğitim	Doğruluk	0,975	0,984	0,996	**0,999**	**0,999**	**0,999**
	F1-Ölçüsü-Mikro	0,975	0,984	0,996	**0,999**	**0,999**	**0,999**
	F1-Ölçüsü-Makro	0,975	0,984	0,996	**0,999**	**0,999**	**0,999**
Test	Doğruluk	0,967	0,969	**0,98**	0,978	0,971	0,969
	F1-Ölçüsü-Mikro	0,967	0,969	**0,98**	0,978	0,971	0,969
	F1-Ölçüsü-Makro	0,967	0,968	**0,979**	0,977	0,97	0,968
Toplam Süre (Saniye)		0,84	0,9	1,13	1,06	1,23	1,49

Çizelge 3.15. VS2 için Çekimsel Model Olasılıksal Öğrenme sonuçları (Maksimum iterasyon=100, Adım Gücü=0,001, Eşik Değer=0,00001)

		Öznitelik Boyutu					
		500	1000	5000	10000	20000	30000
Eğitim	Doğruluk	0,79	0,847	0,91	0,923	0,935	**0,939**
	F1-Ölçüsü-Mikro	0,79	0,847	0,91	0,923	0,935	**0,939**
	F1-Ölçüsü-Makro	0,79	0,847	0,91	0,923	0,935	**0,939**
Test	Doğruluk	0,763	0,819	0,872	0,879	**0,883**	**0,883**
	F1-Ölçüsü-Mikro	0,763	0,819	0,872	0,879	**0,883**	**0,883**
	F1-Ölçüsü-Makro	0,763	0,819	0,872	0,878	**0,883**	**0,883**
Toplam Süre (Saniye)		3,32	3,16	4,55	5,43	6,89	8,92

Çizelge 3.16. VS3 için Çekimsel Model Olasılıksal Öğrenme sonuçları (Maksimum iterasyon=100, Adım Gücü=0,001, Eşik Değer=0,000005)

		Öznitelik Boyutu					
		500	1000	5000	10000	20000	30000
Eğitim	Doğruluk	0,724	0,787	0,856	0,874	0,893	**0,901**
	F1-Ölçüsü-Mikro	0,724	0,787	0,856	0,874	0,893	**0,901**
	F1-Ölçüsü-Makro	0,724	0,787	0,856	0,874	0,892	**0,901**
Test	Doğruluk	0,694	0,754	0,805	0,812	**0,818**	**0,818**
	F1-Ölçüsü-Mikro	0,694	0,754	0,805	0,812	**0,818**	**0,818**
	F1-Ölçüsü-Makro	0,693	0,754	0,805	0,812	**0,818**	**0,818**
Toplam Süre (Saniye)		3,97	4,34	6,44	7,42	11,06	13

Çizelge 3.17. VS4 için Çekimsel Model Olasılıksal Öğrenme sonuçları (Maksimum iterasyon=100, Adım Gücü=0,001, Eşik Değer=0,000005)

		Öznitelik Boyutu					
		500	1000	5000	10000	20000	30000
Eğitim	Doğruluk	0,592	0,655	0,736	0,764	0,792	**0,804**
	F1-Ölçüsü-Mikro	0,592	0,655	0,736	0,764	0,792	**0,804**
	F1-Ölçüsü-Makro	0,591	0,654	0,735	0,763	0,791	**0,804**
Test	Doğruluk	0,547	0,601	0,649	0,661	0,668	**0,67**
	F1-Ölçüsü-Mikro	0,547	0,601	0,649	0,661	0,668	**0,67**
	F1-Ölçüsü-Makro	0,547	0,6	0,652	0,665	0,673	**0,674**
Toplam Süre (Saniye)		4,82	5,32	7,91	8,99	11,24	16,19

Çizelge 3.18. VS1 için Çekimsel Model Toplu Öğrenme sonuçları (Maksimum iterasyon=100, Adım Gücü=0,01, Eşik Değer=0,001)

		Öznitelik Boyutu					
		500	1000	5000	10000	20000	30000
Eğitim	Doğruluk	0,977	0,982	0,996	**0,999**	**0,999**	**0,999**
	F1-Ölçüsü-Mikro	0,977	0,982	0,996	**0,999**	**0,999**	**0,999**
	F1-Ölçüsü-Makro	0,977	0,982	0,996	**0,999**	**0,999**	**0,999**
Test	Doğruluk	0,962	0,969	**0,98**	0,978	0,971	0,969
	F1-Ölçüsü-Mikro	0,962	0,969	**0,98**	0,978	0,971	0,969
	F1-Ölçüsü-Makro	0,962	0,969	**0,979**	0,977	0,97	0,968
Toplam Süre (Saniye)		0,91	0,93	1,08	1,14	1,24	1,41

Çizelge 3.19. VS2 için Çekimsel Model Toplu Öğrenme sonuçları (Maksimum iterasyon=100, Adım Gücü=0,001, Eşik Değer=0,000001)

		Öznitelik Boyutu					
		500	1000	5000	10000	20000	30000
Eğitim	Doğruluk	0,781	0,848	0,91	0,923	0,934	**0,94**
	F1-Ölçüsü-Mikro	0,781	0,848	0,91	0,923	0,934	**0,94**
	F1-Ölçüsü-Makro	0,779	0,848	0,91	0,923	0,934	**0,94**
Test	Doğruluk	0,76	0,819	0,872	0,879	**0,884**	**0,884**
	F1-Ölçüsü-Mikro	0,76	0,819	0,872	0,879	**0,884**	**0,884**
	F1-Ölçüsü-Makro	0,758	0,819	0,872	0,879	**0,883**	**0,884**
Toplam Süre (Saniye)		3,08	3,31	4,36	5,77	7,09	9,16

Çizelge 3.20. VS3 için Çekimsel Model Toplu Öğrenme sonuçları (Maksimum iterasyon=100, Adım Gücü=0,001, Eşik Değer=0,000001)

		Öznitelik Boyutu					
		500	1000	5000	10000	20000	30000
Eğitim	Doğruluk	0,697	0,786	**0,91**	0,875	0,892	0,901
	F1-Ölçüsü-Mikro	0,697	0,786	**0,91**	0,875	0,892	0,901
	F1-Ölçüsü-Makro	0,699	0,786	**0,91**	0,874	0,892	0,901
Test	Doğruluk	0,668	0,749	**0,818**	0,812	0,816	**0,818**
	F1-Ölçüsü-Mikro	0,668	0,749	**0,818**	0,812	0,816	**0,818**
	F1-Ölçüsü-Makro	0,67	0,749	**0,818**	0,812	0,816	**0,818**
Toplam Süre (Saniye)		3,96	4,23	6,21	7,47	11,15	14,34

Çizelge 3.21. VS4 için Çekimsel Model Toplu Öğrenme sonuçları (Maksimum iterasyon=100, Adım Gücü=0,0001, Eşik Değer=0,000005)

		Öznitelik Boyutu					
		500	1000	5000	10000	20000	30000
Eğitim	Doğruluk	0,592	0,654	0,735	0,764	0,793	**0,806**
	F1-Ölçüsü-Mikro	0,592	0,654	0,735	0,764	0,793	**0,806**
	F1-Ölçüsü-Makro	0,59	0,653	0,733	0,761	0,79	**0,803**
Test	Doğruluk	0,55	0,601	0,657	0,67	**0,675**	**0,675**
	F1-Ölçüsü-Mikro	0,55	0,601	0,657	0,67	**0,675**	**0,675**
	F1-Ölçüsü-Makro	0,549	0,601	0,657	0,667	0,674	**0,675**
Toplam Süre (Saniye)		4,95	5,03	7,07	8,53	12,19	16,26

Çizelge 3.22. VS1 için Multinomial Naive Bayes Metodu sonuçları

		Öznitelik Boyutu					
		500	1000	5000	10000	20000	30000
Eğitim	Doğruluk	0,984	0,988	0,997	**0,999**	**0,999**	**0,999**
	F1-Ölçüsü-Mikro	0,984	0,988	0,997	**0,999**	**0,999**	**0,999**
	F1-Ölçüsü-Makro	0,984	0,987	0,997	**0,999**	**0,999**	**0,999**
Test	Doğruluk	0,964	0,967	**0,976**	0,971	0,969	0,967
	F1-Ölçüsü-Mikro	0,964	0,967	**0,976**	0,971	0,969	0,967
	F1-Ölçüsü-Makro	0,964	0,966	**0,975**	0,97	0,968	0,966
Toplam Süre (Saniye)		0,85	0,78	0,89	1	1,26	1,42

Çizelge 3.23. VS2 için Multinomial Naive Bayes Metodu sonuçları

		Öznitelik Boyutu					
		500	1000	5000	10000	20000	30000
Eğitim	Doğruluk	0,831	0,885	0,941	0,953	0,96	**0,964**
	F1-Ölçüsü-Mikro	0,831	0,885	0,941	0,953	0,96	**0,964**
	F1-Ölçüsü-Makro	0,831	0,885	0,941	0,953	0,96	**0,964**
Test	Doğruluk	0,789	0,842	0,891	0,896	**0,899**	0,897
	F1-Ölçüsü-Mikro	0,789	0,842	0,891	0,896	**0,899**	0,897
	F1-Ölçüsü-Makro	0,788	0,842	0,891	0,896	**0,899**	0,897
Toplam Süre (Saniye)		2,69	2,72	3,55	4,23	5,65	9

Çizelge 3.24. VS3 için Multinomial Naive Bayes Metodu sonuçları

		Öznitelik Boyutu					
		500	1000	5000	10000	20000	30000
Eğitim	Doğruluk	0,768	0,83	0,894	0,911	0,928	**0,935**
	F1-Ölçüsü-Mikro	0,768	0,83	0,894	0,911	0,928	**0,935**
	F1-Ölçüsü-Makro	0,767	0,828	0,893	0,91	0,927	**0,934**
Test	Doğruluk	0,725	0,774	0,831	0,834	**0,835**	0,833
	F1-Ölçüsü-Mikro	0,725	0,774	0,831	0,834	**0,835**	0,833
	F1-Ölçüsü-Makro	0,724	0,773	0,829	0,832	**0,833**	0,831
Toplam Süre (Saniye)		3,25	3,38	4,36	5,24	7,27	10,98

Çizelge 3.25. VS4 için Multinomial Naive Bayes Metodu sonuçları

		Öznitelik Boyutu					
		500	1000	5000	10000	20000	30000
Eğitim	Doğruluk	0,647	0,71	0,787	0,813	0,833	**0,848**
	F1-Ölçüsü-Mikro	0,647	0,71	0,787	0,813	0,833	**0,848**
	F1-Ölçüsü-Makro	0,626	0,687	0,765	0,796	0,82	**0,837**
Test	Doğruluk	0,592	0,646	0,693	0,705	0,705	**0,707**
	F1-Ölçüsü-Mikro	0,592	0,646	0,693	0,705	0,705	**0,707**
	F1-Ölçüsü-Makro	0,567	0,619	0,66	0,672	0,674	**0,678**
Toplam Süre (Saniye)		4,31	4,57	5,52	6,8	9,19	11,31

Çizelge 3.26. VS1 için Destek Vektör Makinesi sonuçları (Kernel=Lineer, Maksimum iterasyon=500)

		Öznitelik Boyutu					
		500	1000	5000	10000	20000	30000
Eğitim	Doğruluk	0,908	0,958	0,99	0,998	1	1
	F1-Ölçüsü-Mikro	0,908	0,958	0,99	0,998	1	1
	F1-Ölçüsü-Makro	0,908	0,957	0,99	0,998	1	1
Test	Doğruluk	0,856	0,911	0,955	0,973	0,976	**0,982**
	F1-Ölçüsü-Mikro	0,856	0,911	0,955	0,973	0,976	**0,982**
	F1-Ölçüsü-Makro	0,856	0,911	0,955	0,973	0,975	**0,982**
Toplam Süre (Saniye)		0,86	0,9	0,94	1,08	1,41	4,68

Çizelge 3.27. VS2 için Destek Vektör Makinesi sonuçları (Kernel=Lineer, Maksimum iterasyon=500)

		Öznitelik Boyutu					
		500	1000	5000	10000	20000	30000
Eğitim	Doğruluk	0,53	0,658	0,839	0,912	0,956	**0,988**
	F1-Ölçüsü-Mikro	0,53	0,658	0,839	0,912	0,956	**0,988**
	F1-Ölçüsü-Makro	0,527	0,658	0,839	0,912	0,956	**0,988**
Test	Doğruluk	0,49	0,593	0,748	0,802	0,854	**0,9**
	F1-Ölçüsü-Mikro	0,49	0,593	0,748	0,802	0,854	**0,9**
	F1-Ölçüsü-Makro	0,487	0,592	0,749	0,803	0,854	**0,9**
Toplam Süre (Saniye)		5,21	6,82	12,46	22,11	42,22	244,89

Çizelge 3.28. VS3 için Destek Vektör Makinesi sonuçları (Kernel=Lineer, Maksimum iterasyon=500)

		Öznitelik Boyutu					
		500	1000	5000	10000	20000	30000
Eğitim	Doğruluk	0,462	0,588	0,78	0,87	0,933	**0,98**
	F1-Ölçüsü-Mikro	0,462	0,588	0,78	0,87	0,933	**0,98**
	F1-Ölçüsü-Makro	0,458	0,587	0,78	0,87	0,933	**0,98**
Test	Doğruluk	0,407	0,512	0,661	0,735	0,786	**0,843**
	F1-Ölçüsü-Mikro	0,407	0,512	0,661	0,735	0,786	**0,843**
	F1-Ölçüsü-Makro	0,404	0,511	0,661	0,734	0,786	**0,843**
Toplam Süre (Saniye)		7,72	10,93	23,37	42,55	80,27	431,02

Çizelge 3.29. VS4 için Destek Vektör Makinesi sonuçları (Kernel=Lineer, Maksimum iterasyon=500)

		Öznitelik Boyutu					
		500	1000	5000	10000	20000	30000
Eğitim	Doğruluk	0,381	0,5	0,675	0,772	0,847	**0,935**
	F1-Ölçüsü-Mikro	0,381	0,5	0,675	0,772	0,847	**0,935**
	F1-Ölçüsü-Makro	0,368	0,489	0,665	0,765	0,842	**0,934**
Test	Doğruluk	0,331	0,42	0,546	0,606	0,661	**0,717**
	F1-Ölçüsü-Mikro	0,331	0,42	0,546	0,606	0,661	**0,717**
	F1-Ölçüsü-Makro	0,317	0,408	0,537	0,598	0,654	**0,711**
Toplam Süre (Saniye)		11,72	16,68	35,17	65,43	126,75	659,83

Çizelge 3.30. VS1 için K-NN Metodu sonuçları (K=3)

		Öznitelik Boyutu					
		50	100	250	500	1000	5000
Test	Doğruluk	0,784	0,862	0,92	0,935	0,958	**0,969**
	F1-Ölçüsü-Mikro	0,784	0,862	0,92	0,935	0,958	**0,969**
	F1-Ölçüsü-Makro	0,779	0,86	0,919	0,935	0,956	**0,968**
Toplam Süre (Saniye)		0,87	0,84	0,93	0,98	1,1	2,12

Çizelge 3.31. VS2 için K-NN Metodu sonuçları (K=3)

		Öznitelik Boyutu					
		50	100	250	500	1000	5000
Test	Doğruluk	0,411	0,484	0,519	0,684	0,749	**0,81**
	F1-Ölçüsü-Mikro	0,411	0,484	0,519	0,684	0,749	**0,81**
	F1-Ölçüsü-Makro	0,404	0,478	0,536	0,686	0,751	**0,812**
Toplam Süre (Saniye)		3,55	4,34	7,36	11,74	23,2	93,74

Çizelge 3.32. VS3 için K-NN Metodu sonuçları (K=3)

		Öznitelik Boyutu					
		50	100	250	500	1000	5000
Test	Doğruluk	0,347	0,413	0,454	0,611	0,667	**0,735**
	F1-Ölçüsü-Mikro	0,347	0,413	0,454	0,611	0,667	**0,735**
	F1-Ölçüsü-Makro	0,338	0,407	0,47	0,612	0,668	**0,734**
Toplam Süre (Saniye)		4,68	6,06	10,99	18,86	31,23	147,03

Çizelge 3.33. VS4 için K-NN Metodu sonuçları (K=3)

		Öznitelik Boyutu					
		50	100	250	500	1000	5000
Test	Doğruluk	0,284	0,341	0,392	0,457	0,502	**0,609**
	F1-Ölçüsü-Mikro	0,284	0,341	0,392	0,457	0,502	**0,609**
	F1-Ölçüsü-Makro	0,283	0,342	0,405	0,473	0,52	**0,607**
Toplam Süre (Saniye)		6,74	8,96	18,04	32,5	55,43	247,59

Çizelge 3.34. VS1 Eğitim veri seti için en başarılı metotlar ve değerleri

	Eğitim			
Metotlar	**Öznitelik Sayısı**	**Doğruluk**	**F1-Ölçüsü Mikro**	**F1-Ölçüsü Makro**
SMTM	30000	0,999	0,999	0,999
MTYUM	Hepsi	**1**	**1**	**1**
MTGMYUM	Hepsi	**1**	**1**	**1**
MTÇM-OÖ	10,20 ve 30 Bin	0,999	0,999	0,999
MTÇM-TÖ	10,20 ve 30 Bin	0,999	0,999	0,999
MNBM	10,20 ve 30 Bin	0,999	0,999	0,999
DVM	20 ve 30 bin	**1**	**1**	**1**
KEYK	-	-	-	-

Çizelge 3.35. VS1 Test veri seti için en başarılı metotlar ve değerleri

	Test			
Metotlar	**Öznitelik Sayısı**	**Doğruluk**	**F1-Ölçüsü Mikro**	**F1-Ölçüsü Makro**
SMTM	5000	0,971	0,971	0,97
MTYUM	**5000**	**0,984**	**0,984**	**0,984**
MTGMYUM	**5000**	**0,984**	**0,984**	**0,984**
MTÇM-OÖ	5000	0,98	0,98	0,979
MTÇM-TÖ	5000	0,98	0,98	0,979
MNBM	5000	0,976	0,976	0,975
DVM	30000	0,982	0,982	0,982
KEYK	5000	0,969	0,969	0,968

Çizelge 3.36. VS2 Eğitim veri seti için en başarılı metotlar ve değerleri

	Eğitim			
Metotlar	**Öznitelik Sayısı**	**Doğruluk**	**F1-Ölçüsü Mikro**	**F1-Ölçüsü Makro**
SMTM	30000	0,937	0,937	0,937
MTYUM	10,20,30B	**0,999**	**0,999**	**0,999**
MTGMYUM	10,20,30B	**0,999**	**0,999**	**0,999**
MTÇM-OÖ	30000	0,939	0,939	0,939
MTÇM-TÖ	30000	0,94	0,94	0,94
MNBM	30000	0,964	0,964	0,964
DVM	30000	0,988	0,988	0,988
KEYK	5000	-	-	-

Çizelge 3.37. VS2 Test veri seti için en başarılı metotlar ve değerleri

	Test			
Metotlar	**Öznitelik Sayısı**	**Doğruluk**	**F1-Ölçüsü Mikro**	**F1-Ölçüsü Makro**
SMTM	20,30B	0,884	0,884	0,884
MTYUM	20,30B	0,899	0,899	0,899
MTGMYUM	20,30B	0,899	0,899	0,899
MTÇM-OÖ	20,30B	0,883	0,883	0,883
MTÇM-TÖ	20,30B	0,884	0,884	0,883
MNBM	20000	0,899	0,899	0,899
DVM	30000	**0,9**	**0,9**	**0,9**
KEYK	5000	0,81	0,81	0,812

Çizelge 3.38. VS3 Eğitim veri seti için en başarılı metotlar ve değerleri

	Eğitim			
Metotlar	**Öznitelik Sayısı**	**Doğruluk**	**F1-Ölçüsü Mikro**	**F1-Ölçüsü Makro**
SMTM	30000	0,891	0,891	0,89
MTYUM	20,30B	**0,999**	**0,999**	**0,999**
MTGMYUM	5,10,20,30B	**0,999**	**0,999**	**0,999**
MTÇM-OÖ	30000	0,901	0,901	0,901
MTÇM-TÖ	5000	0,91	0,91	0,91
MNBM	30000	0,935	0,935	0,934
DVM	30000	0,98	0,98	0,98
KEYK	5000	-	-	-

Çizelge 3.39. VS3 Test veri seti için en başarılı metotlar ve değerleri

	Test			
Metotlar	**Öznitelik Sayısı**	**Doğruluk**	**F1-Ölçüsü Mikro**	**F1-Ölçüsü Makro**
SMTM	30000	0,816	0,816	0,814
MTYUM	30000	0,842	0,842	0,841
MTGMYUM	20,30B	0,842	0,842	0,841
MTÇM-OÖ	20,30B	0,818	0,818	0,818
MTÇM-TÖ	5,30B	0,818	0,818	0,818
MNBM	20000	0,835	0,835	0,833
DVM	30000	**0,843**	**0,843**	**0,843**
KEYK	5000	0,735	0,735	0,734

Çizelge 3.40. VS4 Eğitim veri seti için en başarılı metotlar ve değerleri

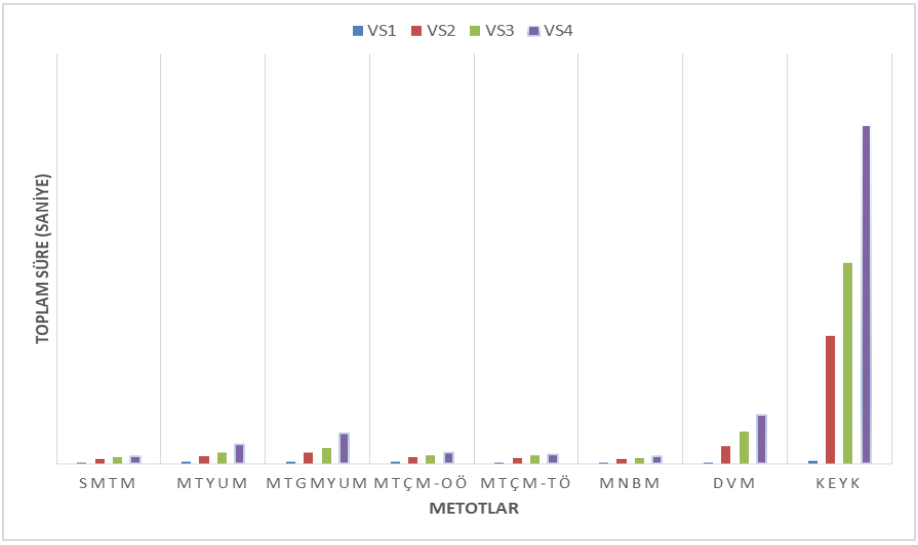
	Eğitim			
Metotlar	**Öznitelik Sayısı**	**Doğruluk**	**F1-Ölçüsü Mikro**	**F1-Ölçüsü Makro**
SMTM	30000	0,802	0,802	0,796
MTYUM	20,30B	0,996	0,996	0,996
MTGMYUM	20,30B	**0,997**	**0,997**	**0,997**
MTÇM-OÖ	30000	0,804	0,804	0,804
MTÇM-TÖ	30000	0,806	0,806	0,803
MNBM	30000	0,848	0,848	0,837
DVM	30000	0,935	0,935	0,934
KEYK	5000	-	-	-

Çizelge 3.41. VS4 Test veri seti için en başarılı metotlar ve değerleri

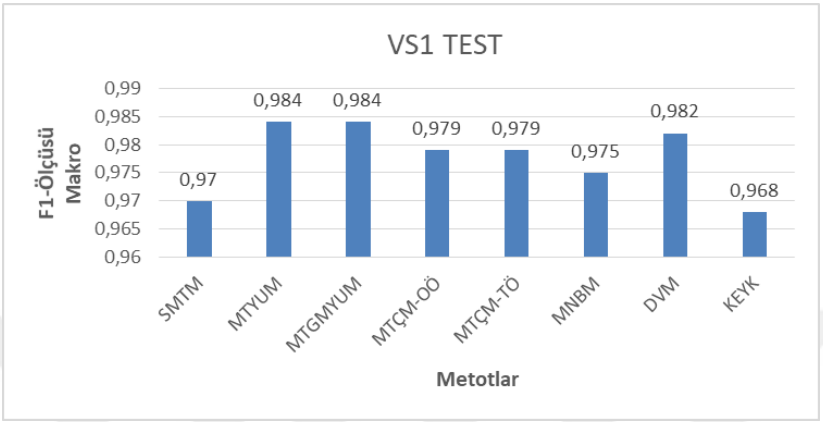
	Test			
Metotlar	**Öznitelik Sayısı**	**Doğruluk**	**F1-Ölçüsü Mikro**	**F1-Ölçüsü Makro**
SMTM	30000	0,686	0,686	0,674
MTYUM	30000	0,725	0,725	0,715
MTGMYUM	30000	**0,726**	**0,726**	**0,716**
MTÇM-OÖ	30000	0,67	0,67	0,674
MTÇM-TÖ	20,30B	0,675	0,675	0,675
MNBM	30000	0,707	0,707	0,678
DVM	30000	0,717	0,717	0,711
KEYK	5000	0,609	0,609	0,607

Çizelge 3.42. Metotların Eğitim ve Test Toplam Süreleri (Saniye)
(Öznitelik Sayısı: 5000)

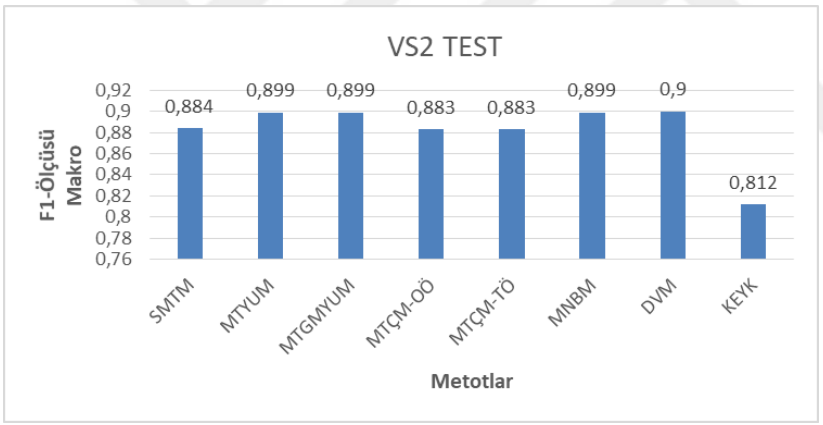
Metotlar	**VS1**	**VS2**	**VS3**	**VS4**
SMTM	0,98	3.69	4.56	5.62
MTYUM	1.22	5.66	8.12	13.79
MTGMYUM	1.19	7.94	11.56	22.17
MTÇM-OÖ	1.13	4.55	6.44	7.91
MTÇM-TÖ	1.08	4.36	6.21	7.07
MNBM	0,89	3.55	4.36	5.52
DVM	0,94	12.46	23.37	35.17
KEYK	2.12	93.74	147.03	247.59



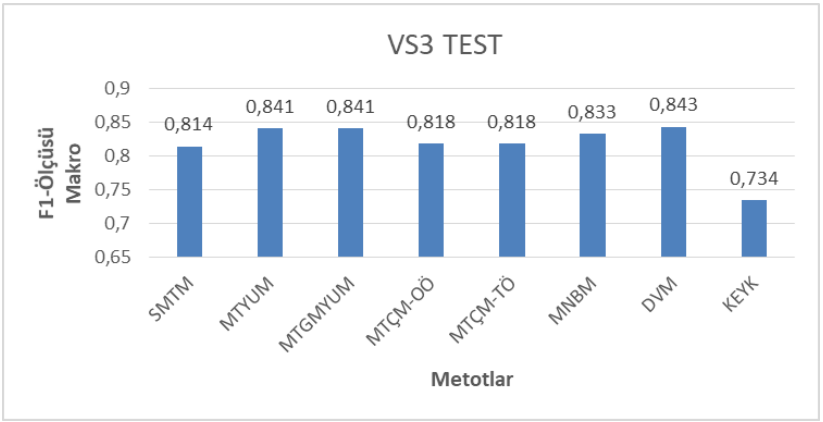
Şekil 3.1. Metotların toplam eğitim ve test süreleri



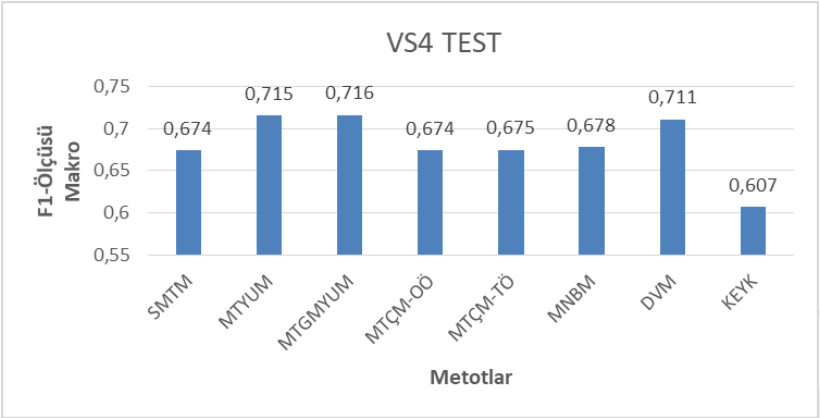
Şekil 3.2. VS1 için metotların F1-Ölçüsü Makro sonuçları



Şekil 3.3. VS2 için metotların F1-Ölçüsü Makro sonuçları



Şekil 3.4. VS3 için metotların F1-Ölçüsü Makro sonuçları



Şekil 3.5. VS4 için metotların F1-Ölçüsü Makro sonuçları

Veri setlerinin öznitelik boyutlarına göre başarılarını değerlendirdiğimizde VS1 için en yüksek başarı genelde 5000'de elde edilmiştir. Fakat diğer veri setlerinde ise öznitelik boyutu 30000 olduğunda en iyi başarıya ulaşılmıştır.

Çizelge 3.2.'den 3.33'e kadar olan bütün çizelgeler, hem eğitim hem de test veri setlerinin metotlara ve özniteliklere göre doğruluk, F1-Ölçüsü Mikro ve F1-Ölçüsü Makro değerleri içermek suretiyle verilmiştir.

Çizelge 3.34.'te görüldüğü üzere VS1'in eğitim veri setini değerlendirdiğimizde en başarılı metotlar MTYUM, MTGMYUM ve DVM'dir. Bu metotlar %100 başarılı bir şekilde eğitilebilmişlerdir. KEYK metodu eğitim aşaması olmadığından hiçbir değer içermemektedir. MTYUM ve MTGMYUM metotları bütün özniteliklerde %100 başarılı olmuştur. DVM ise 20000 ve 30000'de aynı başarıyı yakalamıştır. Çizelge 3.35'de VS1'in test veri seti başarı değerleri gösterilmektedir. Bu değerlere göre en başarılı test metotları MTYUM ve MTGMYUM olmuştur. Öznitelik sayısı 5000 olduğunda bütün ölçülerin %98,4'lük başarıyı yakaladıkları görülmektedir. VS1'in hem eğitim hem de test veri setinin en başarılı metotlarının MTYUM ve MTGMYUM olduğu söylenebilir. Metotların çoğu öznitelik sayısı 5000 olduğunda en yüksek başarıya eriştiler. Metotlar değerlendirilirken kullanılan Doğruluk, F1-Ölçüsü Mikro ve F1-Ölçüsü Markro değerleri arasında genelde çok büyük bir fark oluşmamıştır.

Çizelge 3.36.'de VS2'in eğitim veri seti değerlendirilmiştir. Bu çizelgeye göre en başarılı MTYUM ve MTGMYUM metotlarıdır ve başarıları %99,9 olacak şekilde eğitilebilmişlerdir. MTYUM ve MTGMYUM metotları en büyük başarıyı 10000,

20000 ve 30000 öznitelik sayılarında yakalamışlardır. Çizelge 3.37'de gösterildiği gibi VS2'nin test veri seti başarı değerlerine göre en başarılı test metot DVM'dir. Öznitelik sayısı 30000 olduğunda bütün ölçülerde %90'lık başarıyı elde edilmiştir. Metotların çoğu öznitelik sayısı genelde 30000 olduğunda en yüksek başarıya ulaştılar. MTYUM ve MTGMYUM'nin başarılarına bakıldığında DVM'nin değerine çok yakın olduğu görülür. Yani VS2 için en başarılı metotların DVM, MTYUM ve MTGMYUM olduğu söylenebilir.

Çizelge 3.38.'de VS3'in eğitim veri setinin başarı değerleri verilmiştir. Bu değerlere bakıldığında en başarılı metotların MTYUM ve MTGMYUM olduğu görülmektedir. MTYUM, öznitelik sayısı 20000 ve 30000 olduğunda MTGMYUM ise öznitelik sayısı 5000, 10000, 20000 ve 30000 olduğunda %99,9'luk başarıyı elde etmişlerdir. Çizelge 3.39'da gösterildiği gibi VS3'ün test veri seti başarı değerlerine göre en başarılı test metot DVM olmuştur. Öznitelik sayısı 30000 olduğunda bütün ölçülerde %84,4'lük başarıyı yakalamıştır. Metotların çoğu öznitelik sayısı genelde 30000 olduğunda en yüksek başarıya ulaştılar. VS2'de olduğu gibi MTYUM ve MTGMYUM'nin başarıları DVM'nin değerine oldukça yakındır. Bu veri setinde de en başarılı metotlar DVM, MTYUM ve MTGMYUM'dir.

Çizelge 3.40,'ta görüldüğü üzere VS4'ün eğitim veri setini değerlendirdiğimizde en başarılı metodun MTGMYUM olduğu görülmektedir. Bu metot %99,7'lik bir eğitim başarısı elde etmiştir. MTGMYUM metodu öznitelik sayısı 20000 ve 30000 olduğunda bu başarıya ulaşmıştır. Çizelge 3.41'de VS4'in test veri seti başarı değerleri gösterilmektedir. Bu değerlere göre en başarılı test metodu eğitimde oluğu gibi MTGMYUM olmuştur. Öznitelik sayısı 30000 olduğunda Doğruluk ve F1-Ölçüsü için Mikro %72,6 ve F1-Ölçüsü Makro için ise %71,6'lık başarı elde edilmiştir. VS4'ün hem eğitim hem de test veri setinin en başarılı metodun MTGMYUM olduğu görülmektedir. Metotların çoğu öznitelik sayısı 30000 olduğunda en yüksek başarıya eriştiler.

Bütün veri setlerine göre değerlendirildiğinde en başarılı metodun MTGMYUM olduğu söylenebilir. Sonraki başarılı metotlar MTYUM ve DVM olmaktadır. En başarısız metodun KEYK olduğu görülmektedir. En büyük başarılar genelde öznitelik sayısı 30000 olduğunda elde edildi. Sadece VS1 veri setinde öznitelik sayısı 5000 olduğunda en yüksek başarıya ulaşılmıştır. Test aşamasında boyut artıkça doğruluk ve F1-Ölçüsü değerleri artmaktadır.

Şekil 3.1.'de görüldüğü gibi toplam eğitim ve test süreleri bakımından en yavaş sonuç üreten metot K-En Yakın Komşu (KEYK) metodudur. KEYK metodunun eğitim süreci yoktur. Fakat test süreci çok yavaş olmaktadır. Çünkü bütün eğitim örnekleri ile örnek arasındaki uzaklıklar hesaplanmak zorundadır. Eğer eğitim setindeki örnekler çok fazla ise süreç yavaş olmaktadır. Sonraki yavaş metot DVM metodudur. DVM metodunun eğitim aşaması yani öğrenme süreci de uzun sürmektedir. DVM'nin test işlemi oldukça hızlıdır. Diğer metotlar ise hızlı bir şekilde sonuçlanmaktadır.

Şekil 3.2., 3.3., 3.4. ve 3.5.'de hangi veri setlerinde hangi metodun başarılı olduğu F1-Ölçüsü Makro değerlerine göre grafiksel olarak gösterilmektedir.

4. SONUÇ

Bu tez çalışmasında Merkez Tabanlı Sınıflandırma metotları, K-En Yakın Komşu, Destek Vektör Makinesi ve Naive Bayes sınıflandırma metotları ile ilgili uygulamalar geliştirilmiştir ve elde edilen sonuçlar karşılaştırılmıştır. Yapılan uygulamalar iki aşamadan oluşmaktadır. İlk olarak eğitim işleminden geçirilmiştir. Yani eğitim veri setindeki örneklerden metotlar öğretilerek modeller oluşturmuştur. Bu metotlardan sadece KEYK metodunda eğitim aşaması yoktur. Diğer metotların hepsinde eğitim gerçekleştirilmiştir. İkinci aşama ise test aşamasıdır. Bu aşamada, test veri setindeki örnekler metodun modeli kullanılarak uygulamanın başarısı ölçülür. Bu metotların başarısı dört farklı metin veri seti üzerinde uygulanmak suretiyle ölçülmüştür.

VS1'in eğitim veri setine göre en iyi öğrenen metotlar %100 başarıyla MTYUM, MTGMYUM ve DVM'dir. VS1'in test veri setine göre ise en başarılı (%98,4) test metotları MTYUM ve MTGMYUM olmuştur. VS2'in eğitim veri seti için en başarılı (%99,9) MTYUM ve MTGMYUM metotlarıdır. Fakat VS2'nin test veri setine göre ise en başarılı (%90) test metot DVM'dir. VS3'ün eğitim veri setine göre metotlar eğitildiğinde MTYUM ve MTGMYUM metotları en yüksek başarıyı %99,9 yakalamışlardır. Eğitim veri setinde ise DVM en çok başarılı olmuştur. Fakat MTYUM ve MTGMYUM metotlarının başarısı DVM'ye çok yakındır. VS4'ün hem eğitim hem de test veri setini değerlendirdiğimizde en başarılı MTGMYUM metodudur. Bu metot, %99,7 eğitim ve %72,6 test başarısı elde etmiştir.

Genel olarak değerlendirildiğinde en başarılı metodun MTGMYUM olduğu görülebilir. Sonrasında MTYUM ve DVM metotları gelmektedir. En başarısız metot ise KEYK'dir. Uygulamalardaki en yüksek başarılar öznitelik sayısı 30000 olduğunda elde edildi. Fakat VS1 veri setinde en yüksek başarı öznitelik sayısı 5000 olduğunda yakalanmıştır. Hız bakımından değerlendirildiğinde en yavaş metotlar KEYK ve DVM olmaktadır. Merkez tabanlı metotlar oldukça hızlıdır.

Sonraki çalışmalarda farklı metotlar ile metin veri setlerinde uygulamalar geliştirilecektir ve karşılaştırılacaktır. Merkez tabanlı metotlarda yeni yaklaşımlar bulunmaya çalışılacaktır.

KAYNAKLAR

Agarwal, B., Mittal, N. 2014. Text classification using machine learning methods-a survey. In **Proceedings of the Second International Conference on Soft Computing for Problem Solving (SocProS 2012),** December 28-30. 2012, Springer, pp. 701-709.

Anonim, 2019. **Kemik, Veri Kümeleri** [http://www.kemik.yildiz.edu.tr/?id=28] Erişim Tarihi: 03.05.2019.

Aşlıyan R., Günel K. 2010. Metin İçerikli Türkçe Dokümanların Sınıflandırılması. Akademik Bilişim 2010, Sözlü, Muğla, 10/02/2010.

Baker, L., McCallum, A. 1998. Distributional clustering of words for text classification. In **Proc. of the Fourth Int'l Conference on Knowledge Discovery and Data Mining**.

Cataltepe, Z., Aygun, E. 2007. An improvement of centroid-based classification algorithm for text classification. In **Proceedings of Data Engineering Workshop**, 2007 IEEE 23rd International Conference on, IEEE, pp. 952-956.

Chau, R.,Yeh, C., Smith, K.A. 2005. A neural network model for hierarchical multilingual text categorization. In **Proceedings of Advances in Neural Networks**, Springer, pp. 238-245.

Chen, R.C., Hsieh, C.H. 2006. Web page classification based on a support vector machine using a weighted vote schema. **Expert Syst. Appl.** 31 (2): 427-435.

Cohen, W.W., Hirsh, H. 1998. Joins that generalize: Text classification using WHIRL. In **Proc. of the Fourth Int'l Conference on Knowledge Discovery and Data Mining**.

Colace, F., De Santo, M., Greco, L., Napoletano, P. 2014. Text classification using a few labeled examples. **Comput. Human Behav.** 30: 689-697.

Dandan, W., Qingcai, C., Xiaolong, W. 2014. A framework of Centroid-Based methods for text categorization. **IEICE Trans. Inf. Syst.** 97(2): 245-254.

Guan, H., Zhou, Z., Guo, M., 2009. A Class-Feature-Centroid classifier for text categorization. In **Proceedings of the 18th International Conference on World Wide Web, ACM,** pp. 201-210.

Guo, G., Wang, H., Bell, D., Bi, Y., Greer, K. 2006. Using knn model for automatic text t categorization. **Soft. Comput.** 10 (5): 423-430.

Günel, K., Aşlıyan, R., Gör, İ. 2016. Geometrical Modification of Learning Vector Quantization Method for Solving Classification Problems. **Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü Dergisi.** 20(3): 414-420. Doi=10.19113/sdufbed.22419.

Joachims, T. 1998. Text categorization with support vector machines: Learning with many relevant features. In **Proc. of the European Conference on Machine Learning.**

Kibriya, A.M., Frank, E., Pfahringer, B., Holmes, G. 2004. Multinomial Naive Bayes for text categorization revisited. In **Proceedings of Advances in Artificial Intelligence**, Springer, pp. 488-499.

Kolyiğit Ö., Aşlıyan R., Günel K. 2012. Türkçe Dokümanlar İçin Yazar Tanıma, Akademik Bilişim 2012, Sözlü, Uşak, 3/02/2012.

Lam, W., Ho, C.Y. 1998. Using a generalized instance set for automatic text categorization. In **SIGIR-98**.

Lam, S.L., Lee, D.L. 1999. Feature reduction for neural network based text categorization. **Database Systems for Advanced Applications.** In **Proceedings of 6th International Conference on, IEEE,** pp. 195-202.

Lertnattee, V., Theeramunkong, T. 2006. Class normalization in centroid-based text categorization. **Inf. Sci. (Ny).** 176(12): 1712-1738.

Liu, C., Wang, W., Tu, G., Xiang, Y., Wang, S., Lv, F. 2017. A new Centroid-Based Classification model for text categorization. **Knowledge-Based Systems,** 136: 15-26.

Liu, C., Wang, W., Zhao, Q., Shen, X., Konan, M. 2017. A new feature selection method based on a validity index of feature subset. **Pattern Recognit. Lett.** 92: 1-8

McCallum, A., Nigam. K. 1998. A comparison of event models for naive bayes text classification. In **AAAI-98 Workshop on Learning for Text Categorization.**

Mikolov, T., Chen, K., Corrado, G., Dean, J. 2013. Efficient estimation of word representations in vector space. **arXiv Preprint arXiv:1301.3781**.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In

Proceedings of Advances in Neural Information Processing Systems, pp. 3111-3119.

Mikolov, T., Yih, W.T., Zweig, G. 2013. Linguistic regularities in continuous space word representations. In **Proceedings of Hlt-naacl, vol.13,** pp. 746-751.

Ross, J. 1993. Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo, CA.

Salton, G. 1989. Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer. Addison-Wesley.

Shakhnarovich, G., Darrell, T., Indyk, P. 2008. Nearest-neighbor methods in learning and vision. **IEEE Trans. Neural Networks**, 19(2): 377.

Takçı, H., Güngör, T. 2012. A high performance Centroid-Based Classification approach for language identification. **Pattern Recognit. Lett.**, 33(16): 2077-2084.

Tan, S. 2007. Large margin dragpushing strategy for centroid text categorization. **Expert Syst. Appl.** 33(1): 215-220.

Tan, S. 2008. An improved centroid classifier for text categorization. **Expert Syst. Appl.** 35(1): 279-285.

Vapnik, V.N. 1995. The Nature of Statistical Learning Theory. Springer-Verlag, New York.

Yang, Y., Liu, X. 1999. A re-examination of text categorization methods. In **SIGIR-99.**

Yu, Z., Li, L., Liu, J., Han, G. 2015. Hybrid adaptive classifier ensemble. **IEEE Trans. Cybern.** 45(2): 177-190.

Yu, Z., Chen, H., Liu, J., You, J., Leung, H., Han, G. 2016. Hybrid-nearest neighbor classifier. **IEEE Trans. Cybern.** 46 (6): 1263-1275.

Zhang, B.F., Su, J.S., Xu, X. 2006. A class-incremental learning method for multi-class support vector machines in text classification. In **Proceedings of Machine Learning and Cybernetics,** 2006 International Conference on, IEEE, pp. 2581-2585.

EKLER

Ek 1. Python Kaynak Kodları

Aşağıda Merkez Tabanlı, Yaklaştır-Uzaklaştır, Destek Vektör Makinesi, Naive Bayes ve K-En Yakın Komşu metotlarıyla geliştirilmiş uygulamaların Python kodları verilmiştir.

Ek 1.1. Python Merkez Tabanlı Metot Uygulaması

 # "MerkezTabanliMetotlar.py" diye kaydedilir.

```
import numpy as np
from scipy.spatial import distance
def prediction1(merkezler,testvektor,sinifsayisi):
    say=0
    uzakliklar=np.zeros(sinifsayisi, dtype=float)
    while(say<sinifsayisi):
        # uzaklıklar noktasal çarpıma göre hesaplanıyor
        # bu uzaklığa göre argmax kullanılır.
        uzakliklar[say] = np.dot(merkezler[say], testvektor)
        say=say+1
    return np.argmax(uzakliklar)
def predictionAll(merkezler,testmatris,sinifsayisi):
    say=0
    dosyasayisi=testmatris.shape[0]
    sonuclar=np.zeros(dosyasayisi,dtype=int)
    while(say<dosyasayisi):
        sonuclar[say]=prediction1(merkezler,testmatris[say],sinifsayisi)
        say=say+1
    return sonuclar
def merkezTabanliMetotNormalize(x,y,dokumansayisi,sinifsayisi):
    # Merkez tabanlı sınıflandırma metodu
    boyutsayisi=x.shape[1]
    sonuc=np.zeros((sinifsayisi,boyutsayisi), dtype=float)
```

```
    hersinifelemansayisi=np.zeros(sinifsayisi,dtype=int)
    say=0
    while (say<dokumansayisi): # Aynı sınıfta olanların toplamları bulunur
        sonuc[y[say]]=x[say]+sonuc[y[say]]
        hersinifelemansayisi[y[say]]=hersinifelemansayisi[y[say]]+1
        say=say+1
    say=0
    while (say<sinifsayisi):
        # Aynı sınıftakilerin toplamlarının normları bulunur
        # norm2 ye göre yani elemanların karelerinin toplamının karekökü
        sonuc[say]=sonuc[say]/np.linalg.norm(sonuc[say],ord=None)
        say=say+1
    return sonuc
    # Bulunan sonuç sınıfların merkez vektörleridir.
```

 # “CentroidUygulama.py” diye kaydedilir.

```
import os
import numpy as np
import sklearn.datasets as skd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn import metrics
from sklearn.metrics import accuracy_score
from sklearn.model_selection import KFold # import KFold
import MerkezTabanliMetotlar as M
import time
start_time = time.time()
def KlasorleriGetir(yol):
  klasorler = []
  # r=root, d=directories, f = files
  for r, d, f in os.walk(yol):
    for klasor in d:
```

```
    klasorler.append(klasor)
  return klasorler
yol = 'metinler1\\Dosyalar'
categories=KlasorleriGetir(yol)
news_train_test=skd.load_files(yol,categories=categories,encoding='Windows-1250')
count_vect =CountVectorizer()
x_train_test_tf=count_vect.fit_transform(news_train_test.data)
x_train_test_tf.shape
tfidf_transformer=TfidfTransformer()
x_train_test_tfidf=tfidf_transformer.fit_transform(x_train_test_tf)
x_train_test_tfidf.shape
x=x_train_test_tfidf.toarray() # Bütün dokümanlar satır vektör şeklinde
y=news_train_test.target # Dokümanlara karşılık gelen sınıflar (etiket) 0,1,2 şeklinde
# Kfoldcros validation
kf = KFold(n_splits=5)
kf.get_n_splits(x)
train_accuracy=[]
test_accuracy=[]
train_f1_score_micro=[]
test_f1_score_micro=[]
train_f1_score_macro=[]
test_f1_score_macro=[]
say=0
for train_index, test_index in kf.split(x):
  print()
  print("Fold:",say+1)
  x_train, x_test = x[train_index], x[test_index]
  y_train, y_test = y[train_index], y[test_index]
  dokumansayisi=y_train.shape[0]
  sinifsayisi=len(categories)
```

```
  # Metot eğitiliyor, yani merkezler bulunuyor
merkezler=M.merkezTabanliMetotNormalize(x_train,y_train,dokumansayisi,sinifs
ayisi)
    # Eğtitim setinde test ediliyor
  predicted=M.predictionAll(merkezler,x_train,sinifsayisi)
  sonuc=np.round(accuracy_score(y_train,predicted),3)
  print("Eğitim Doğruluk:",sonuc)
  train_accuracy.append(sonuc)
    sonuc=sonuc=np.round(metrics.f1_score(y_train,predicted,average='micro'),3)
  print("Eğitim F1_score_micro:",sonuc)
  train_f1_score_micro.append(sonuc)
    sonuc=sonuc=np.round(metrics.f1_score(y_train,predicted,average='macro'),3)
  print("Eğitim F1_score_macro:",sonuc)
  train_f1_score_macro.append(sonuc)
    # Test setinde test ediliyor
  predicted=M.predictionAll(merkezler,x_test,sinifsayisi)
  sonuc=sonuc=np.round(accuracy_score(y_test,predicted),3)
  print("Test Doğruluk:",sonuc)
  test_accuracy.append(sonuc)
    sonuc=sonuc=np.round(metrics.f1_score(y_test,predicted,average='micro'),3)
  print("Test F1_score_micro:",sonuc)
  test_f1_score_micro.append(sonuc)
    sonuc=sonuc=np.round(metrics.f1_score(y_test,predicted,average='macro'),3)
  print("Test F1_score_macro:",sonuc)
  test_f1_score_macro.append(sonuc)
  say+=1
  print()
print("Eğitim Set Ortalama Doğruluk:",np.round(np.mean(train_accuracy),3))
print("Eğitim Set Ortalama f1_score
micro:",np.round(np.mean(train_f1_score_micro),3))
print("Eğitim Set Ortalama f1_score
macro:",np.round(np.mean(train_f1_score_macro),3))
```

```
print()
print("Test Set Ortalama Doğruluk:",np.round(np.mean(test_accuracy),3))
print("Test Set Ortalama f1_score
micro:",np.round(np.mean(test_f1_score_micro),3))
print("Test Set Ortalama f1_score
macro:",np.round(np.mean(test_f1_score_macro),3))
elapsed_time = time.time() - start_time
print("Geçen zaman:",elapsed_time,"saniye")
print("Geçen zaman:",time.strftime("%H:%M:%S",
time.gmtime(elapsed_time)),"saniye")
```

Ek 1.2. Python Yaklaştır-Uzaklaştır Metodu Uygulaması

```
# "MerkezTabanliMetotlar.py"  diye kaydedilir.
import numpy as np
from scipy.spatial import distance
def prediction1(merkezler,testvektor,sinifsayisi):
    say=0
    uzakliklar=np.zeros(sinifsayisi, dtype=float)
    while(say<sinifsayisi):
        # uzaklıklar noktasal çarpıma göre hesaplanıyor
        # bu uzaklığa göre argmax kullanılır.
        uzakliklar[say] = np.dot(merkezler[say], testvektor)
        say=say+1
    return np.argmax(uzakliklar)
def predictionAll(merkezler,testmatris,sinifsayisi):
    say=0
    dosyasayisi=testmatris.shape[0]
    sonuclar=np.zeros(dosyasayisi,dtype=int)
    while(say<dosyasayisi):
        sonuclar[say]=prediction1(merkezler,testmatris[say],sinifsayisi)
        say=say+1
    return sonuclar
def merkezTabanliMetotNormalize(x,y,dokumansayisi,sinifsayisi):
```

```
    # Merkez tabanlı sınıflandırma metodu
    boyutsayisi=x.shape[1]
    sonuc=np.zeros((sinifsayisi,boyutsayisi), dtype=float)
    hersinifelemansayisi=np.zeros(sinifsayisi,dtype=int)
    say=0
    while (say<dokumansayisi): # Aynı sınıfta olanların toplamları bulunur
        sonuc[y[say]]=x[say]+sonuc[y[say]]
        hersinifelemansayisi[y[say]]=hersinifelemansayisi[y[say]]+1
        say=say+1
    say=0
    while (say<sinifsayisi):
        # Aynı sınıftakilerin toplamlarının normları bulunur
        # norm2 ye göre yani elemanların karelerinin toplamının karekökü
        sonuc[say]=sonuc[say]/np.linalg.norm(sonuc[say],ord=None)
        say=say+1
    return sonuc
    # Bulunan sonuç sınıfların merkez vektörleridir.
def merkezTabanliMetotNormalizeDragPush(x,y,merkezler,dokumansayisi,
sinifsayisi,maxiterasyon,ogrenmekatsayisi):
     iter=0
     while (iter<maxiterasyon):
         say=0
         while (say<dokumansayisi):
             siniftahmin=prediction1(merkezler,x[say],sinifsayisi)
             if siniftahmin!=y[say]: # yanlış sınıflandırılmış ise
                 # y[say] kendi sınıfı; siniftahmin yanlış sınıf olmaktadır
                 # Kendi sınıfına yaklaştırılıyor
                 merkezler[y[say]]=merkezler[y[say]]+ogrenmekatsayisi*x[say]
merkezler[y[say]]=merkezler[y[say]]/np.linalg.norm(merkezler[y[say]],ord=None)
                 # farklı sınıftan uzaklaştırılıyor
                 gecici=merkezler[siniftahmin]
                 gecici=gecici-ogrenmekatsayisi*x[say]
```

```
            gecici[gecici<0]=0; # sıfırdan küçük değerler 0 0ıur
            gecici=gecici/np.linalg.norm(gecici,ord=None)
            merkezler[siniftahmin]=gecici
            say=say+1
        iter=iter+1
    return merkezler
```

 # "YaklastirUzaklastir.py" diye kaydedilir

```
import os
import numpy as np
import sklearn.datasets as skd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn import metrics
from sklearn.metrics import accuracy_score
from sklearn.model_selection import KFold # import KFold
import MerkezTabanliMetotlar as M
import time
start_time = time.time()
def KlasorleriGetir(yol):
  klasorler = []
  # r=root, d=directories, f = files
  for r, d, f in os.walk(yol):
    for klasor in d:
      klasorler.append(klasor)
  return klasorler
yol = 'metinler1\\Dosyalar'
categories=KlasorleriGetir(yol)
news_train_test=skd.load_files(yol,categories=categories,encoding='Windows-
1250')
count_vect =CountVectorizer()
x_train_test_tf=count_vect.fit_transform(news_train_test.data)
x_train_test_tf.shape
```

```
tfidf_transformer=TfidfTransformer()
x_train_test_tfidf=tfidf_transformer.fit_transform(x_train_test_tf)
x_train_test_tfidf.shape
x=x_train_test_tfidf.toarray() # Bütün dokümanlar satır vektör şeklinde
y=news_train_test.target # Dokümanlara karşılık gelen sınıflar (etiket) 0,1,2 şeklinde
# Kfoldcros validation
kf = KFold(n_splits=5)
kf.get_n_splits(x)
train_accuracy=[]
test_accuracy=[]
train_f1_score_micro=[]
test_f1_score_micro=[]
train_f1_score_macro=[]
test_f1_score_macro=[]
maxiterasyon=8
ogrenmekatsayisi=0,2
say=0
for train_index, test_index in kf.split(x):
  print()
  print("Fold:",say+1)
  x_train, x_test = x[train_index], x[test_index]
  y_train, y_test = y[train_index], y[test_index]
  dokumansayisi=y_train.shape[0]
  sinifsayisi=len(categories)
  # Metot eğitiliyor, yani merkezler bulunuyor
merkezler=M.merkezTabanliMetotNormalize(x_train,y_train,dokumansayisi,sinifsayisi)
  # DragPush metodu uygulanıyor
merkezler=M.merkezTabanliMetotNormalizeDragPush(x_train,y_train,merkezler,dokumansayisi,sinifsayisi,maxiterasyon,ogrenmekatsayisi)
      # Eğtitim setinde test ediliyor
```

```
  predicted=M.predictionAll(merkezler,x_train,sinifsayisi)
  sonuc=np.round(accuracy_score(y_train,predicted),3)
  print("Eğitim Doğruluk:",sonuc)
  train_accuracy.append(sonuc)
    sonuc=sonuc=np.round(metrics.f1_score(y_train,predicted,average='micro'),3)
  print("Eğitim F1_score_micro:",sonuc)
  train_f1_score_micro.append(sonuc)
    sonuc=sonuc=np.round(metrics.f1_score(y_train,predicted,average='macro'),3)
  print("Eğitim F1_score_macro:",sonuc)
  train_f1_score_macro.append(sonuc)
    # Test setinde test ediliyor
  predicted=M.predictionAll(merkezler,x_test,sinifsayisi)
  sonuc=sonuc=np.round(accuracy_score(y_test,predicted),3)
  print("Test Doğruluk:",sonuc)
  test_accuracy.append(sonuc)
    sonuc=sonuc=np.round(metrics.f1_score(y_test,predicted,average='micro'),3)
  print("Test F1_score_micro:",sonuc)
  test_f1_score_micro.append(sonuc)
    sonuc=sonuc=np.round(metrics.f1_score(y_test,predicted,average='macro'),3)
  print("Test F1_score_macro:",sonuc)
  test_f1_score_macro.append(sonuc)
      say+=1
print()
print("Eğitim Set Ortalama Doğruluk:",np.round(np.mean(train_accuracy),3))
print("Eğitim Set Ortalama f1_score
micro:",np.round(np.mean(train_f1_score_micro),3))
print("Eğitim Set Ortalama f1_score
macro:",np.round(np.mean(train_f1_score_macro),3))
print()
print("Test Set Ortalama Doğruluk:",np.round(np.mean(test_accuracy),3))
print("Test Set Ortalama f1_score
micro:",np.round(np.mean(test_f1_score_micro),3))
```

```
print("Test Set Ortalama f1_score
macro:",np.round(np.mean(test_f1_score_macro),3))
elapsed_time = time.time() - start_time
print("Geçen zaman:",elapsed_time,"saniye")
print("Geçen zaman:",time.strftime("%H:%M:%S",
time.gmtime(elapsed_time)),"saniye")
```

Ek 1.3. Python DVM Uygulaması

```
import os
import numpy as np
from sklearn import svm
from sklearn.naive_bayes import GaussianNB, MultinomialNB, BernoulliNB
from sklearn.neighbors import KNeighborsClassifier
import sklearn.datasets as skd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn import metrics
from sklearn.metrics import accuracy_score
from sklearn.model_selection import KFold # import KFold
import MerkezTabanliMetotlar as M
import time
start_time = time.time()
def KlasorleriGetir(yol):
  klasorler = []
  # r=root, d=directories, f = files
  for r, d, f in os.walk(yol):
    for klasor in d:
      klasorler.append(klasor)
  return klasorler
yol = 'metinler1\\Dosyalar'
categories=KlasorleriGetir(yol)
news_train_test=skd.load_files(yol,categories=categories,encoding='Windows-
1250')
```

```
count_vect =CountVectorizer()
x_train_test_tf=count_vect.fit_transform(news_train_test.data)
x_train_test_tf.shape
tfidf_transformer=TfidfTransformer()
x_train_test_tfidf=tfidf_transformer.fit_transform(x_train_test_tf)
x_train_test_tfidf.shape
x=x_train_test_tfidf.toarray() # Bütün dokümanlar satır vektör şeklinde
y=news_train_test.target
# Kfoldcross validation
kf = KFold(n_splits=5)
kf.get_n_splits(x)
train_accuracy=[]
test_accuracy=[]
train_f1_score_micro=[]
test_f1_score_micro=[]
train_f1_score_macro=[]
test_f1_score_macro=[]
say=0
for train_index, test_index in kf.split(x):
  print()
  print("Fold:",say+1)
  x_train, x_test = x[train_index], x[test_index]
  y_train, y_test = y[train_index], y[test_index]
  dokumansayisi=y_train.shape[0]
  sinifsayisi=len(categories)
  # SVM modeli oluşturuluyor
  model = svm.SVC(kernel='linear', max_iter=500)
  model.fit(x_train,y_train)
  # Eğitim seti test ediliyor
  predicted=model.predict(x_train)
  sonuc=np.round(accuracy_score(y_train,predicted),3)
```

```
  print("Eğitim Doğruluk:",sonuc)
  train_accuracy.append(sonuc)
  sonuc=sonuc=np.round(metrics.f1_score(y_train,predicted,average='micro'),3)
  print("Eğitim F1_score_micro:",sonuc)
  train_f1_score_micro.append(sonuc)
  sonuc=sonuc=np.round(metrics.f1_score(y_train,predicted,average='macro'),3)
  print("Eğitim F1_score_macro:",sonuc)
  train_f1_score_macro.append(sonuc)
  # Test seti test ediliyor
  predicted=model.predict(x_test)
  sonuc=sonuc=np.round(accuracy_score(y_test,predicted),3)
  print("Test Doğruluk:",sonuc)
  test_accuracy.append(sonuc)
  sonuc=sonuc=np.round(metrics.f1_score(y_test,predicted,average='micro'),3)
  print("Test F1_score_micro:",sonuc)
  test_f1_score_micro.append(sonuc)
  sonuc=sonuc=np.round(metrics.f1_score(y_test,predicted,average='macro'),3)
  print("Test F1_score_macro:",sonuc)
  test_f1_score_macro.append(sonuc)
  say+=1
print()
print("Eğitim Set Ortalama Doğruluk:",np.round(np.mean(train_accuracy),3))
print("Eğitim Set Ortalama f1_score
micro:",np.round(np.mean(train_f1_score_micro),3))
print("Eğitim Set Ortalama f1_score
macro:",np.round(np.mean(train_f1_score_macro),3))
print()
print("Test Set Ortalama Doğruluk:",np.round(np.mean(test_accuracy),3))
print("Test Set Ortalama f1_score
micro:",np.round(np.mean(test_f1_score_micro),3))
print("Test Set Ortalama f1_score
macro:",np.round(np.mean(test_f1_score_macro),3))
elapsed_time = time.time() - start_time
```

```
print("Geçen zaman:",elapsed_time,"saniye")
print("Geçen zaman:",time.strftime("%H:%M:%S",
time.gmtime(elapsed_time)),"saniye")
```

Ek 1.4. Python Naive Bayes Uygulaması

```
import os
import numpy as np
from sklearn.naive_bayes import GaussianNB, MultinomialNB, BernoulliNB
from sklearn.neighbors import KNeighborsClassifier
import sklearn.datasets as skd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn import metrics
from sklearn.metrics import accuracy_score
from sklearn.model_selection import KFold # import KFold
import MerkezTabanliMetotlar as M
import time
start_time = time.time()
def KlasorleriGetir(yol):
  klasorler = []
  # r=root, d=directories, f = files
  for r, d, f in os.walk(yol):
    for klasor in d:
      klasorler.append(klasor)
  return klasorler
yol = 'metinler1\\Dosyalar'
categories=KlasorleriGetir(yol)
news_train_test=skd.load_files(yol,categories=categories,encoding='Windows-
1250')
count_vect =CountVectorizer()
x_train_test_tf=count_vect.fit_transform(news_train_test.data)
x_train_test_tf.shape
```

```
tfidf_transformer=TfidfTransformer()
x_train_test_tfidf=tfidf_transformer.fit_transform(x_train_test_tf)
x_train_test_tfidf.shape
x=x_train_test_tfidf.toarray() # Bütün dokümanlar satır vektör şeklinde
y=news_train_test.target # Dokümanlara karşılık gelen sınıflar
# Kfoldcros validation
kf = KFold(n_splits=5)
kf.get_n_splits(x)
train_accuracy=[]
test_accuracy=[]
train_f1_score_micro=[]
test_f1_score_micro=[]
train_f1_score_macro=[]
test_f1_score_macro=[]
say=0
for train_index, test_index in kf.split(x):
  print()
  print("Fold:",say+1)
  x_train, x_test = x[train_index], x[test_index]
  y_train, y_test = y[train_index], y[test_index]
  dokumansayisi=y_train.shape[0]
  sinifsayisi=len(categories)
  # Naive Bayes modeli oluşturuluyor
  nb = GaussianNB()
  nb.fit(x_train,y_train)
    # Eğitim seti test ediliyor
  predicted=nb.predict(x_train)
  sonuc=np.round(accuracy_score(y_train,predicted),3)
  print("Eğitim Doğruluk:",sonuc)
  train_accuracy.append(sonuc)
  sonuc=sonuc=np.round(metrics.f1_score(y_train,predicted,average='micro'),3)
```

```
  print("Eğitim F1_score_micro:",sonuc)
  train_f1_score_micro.append(sonuc)
  sonuc=sonuc=np.round(metrics.f1_score(y_train,predicted,average='macro'),3)
  print("Eğitim F1_score_macro:",sonuc)
  train_f1_score_macro.append(sonuc)
  # Test setinde test ediliyor
  predicted=nb.predict(x_test)
  sonuc=sonuc=np.round(accuracy_score(y_test,predicted),3)
  print("Test Doğruluk:",sonuc)
  test_accuracy.append(sonuc)
  sonuc=sonuc=np.round(metrics.f1_score(y_test,predicted,average='micro'),3)
  print("Test F1_score_micro:",sonuc)
  test_f1_score_micro.append(sonuc)
  sonuc=sonuc=np.round(metrics.f1_score(y_test,predicted,average='macro'),3)
  print("Test F1_score_macro:",sonuc)
  test_f1_score_macro.append(sonuc)
  say+=1
print()
print("Eğitim Set Ortalama Doğruluk:",np.round(np.mean(train_accuracy),3))
print("Eğitim Set Ortalama f1_score
micro:",np.round(np.mean(train_f1_score_micro),3))
print("Eğitim Set Ortalama f1_score
macro:",np.round(np.mean(train_f1_score_macro),3))
print()
print("Test Set Ortalama Doğruluk:",np.round(np.mean(test_accuracy),3))
print("Test Set Ortalama f1_score
micro:",np.round(np.mean(test_f1_score_micro),3))
print("Test Set Ortalama f1_score
macro:",np.round(np.mean(test_f1_score_macro),3))
elapsed_time = time.time() - start_time
print("Geçen zaman:",elapsed_time,"saniye")
print("Geçen zaman:",time.strftime("%H:%M:%S",
time.gmtime(elapsed_time)),"saniye")
```

Ek 1.5. Python K-EYK Uygulaması

```
import os
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
import sklearn.datasets as skd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn import metrics
from sklearn.metrics import accuracy_score
from sklearn.model_selection import KFold # import KFold
import MerkezTabanliMetotlar as M
import time
start_time = time.time()
def KlasorleriGetir(yol):
  klasorler = []
  # r=root, d=directories, f = files
  for r, d, f in os.walk(yol):
    for klasor in d:
      klasorler.append(klasor)
  return klasorler
yol = 'metinler1\\Dosyalar'
categories=KlasorleriGetir(yol)
news_train_test=skd.load_files(yol,categories=categories,encoding='Windows-
1250')
count_vect =CountVectorizer()
x_train_test_tf=count_vect.fit_transform(news_train_test.data)
x_train_test_tf.shape
tfidf_transformer=TfidfTransformer()
x_train_test_tfidf=tfidf_transformer.fit_transform(x_train_test_tf)
x_train_test_tfidf.shape
x=x_train_test_tfidf.toarray() # Bütün dokümanlar satır vektör şeklinde
```

```
y=news_train_test.target # Dokümanlara karşılık gelen sınıflar
# Kfoldcros validation
kf = KFold(n_splits=5)
kf.get_n_splits(x)
train_accuracy=[]
test_accuracy=[]
train_f1_score_micro=[]
test_f1_score_micro=[]
train_f1_score_macro=[]
test_f1_score_macro=[]
say=0
for train_index, test_index in kf.split(x):
  print()
  print("Fold:",say+1)
  x_train, x_test = x[train_index], x[test_index]
  y_train, y_test = y[train_index], y[test_index]
  dokumansayisi=y_train.shape[0]
  sinifsayisi=len(categories)
  # KNN modeli oluşturuluyor
  knn=KNeighborsClassifier(n_neighbors=3)
  knn.fit(x_train,y_train)
  # Test setinde test ediliyor
  predicted=knn.predict(x_test)
  sonuc=sonuc=np.round(accuracy_score(y_test,predicted),3)
  print("Test Doğruluk:",sonuc)
  test_accuracy.append(sonuc)
  sonuc=sonuc=np.round(metrics.f1_score(y_test,predicted,average='micro'),3)
  print("Test F1_score_micro:",sonuc)
  test_f1_score_micro.append(sonuc)
  sonuc=sonuc=np.round(metrics.f1_score(y_test,predicted,average='macro'),3)
  print("Test F1_score_macro:",sonuc)
```

```
  test_f1_score_macro.append(sonuc)
  say+=1
print()
print("Eğitim Set Ortalama Doğruluk:",np.round(np.mean(train_accuracy),3))
print("Eğitim Set Ortalama f1_score
micro:",np.round(np.mean(train_f1_score_micro),3))
print("Eğitim Set Ortalama f1_score
macro:",np.round(np.mean(train_f1_score_macro),3))
print()
print("Test Set Ortalama Doğruluk:",np.round(np.mean(test_accuracy),3))
print("Test Set Ortalama f1_score
micro:",np.round(np.mean(test_f1_score_micro),3))
print("Test Set Ortalama f1_score
macro:",np.round(np.mean(test_f1_score_macro),3))
elapsed_time = time.time() - start_time
print("Geçen zaman:",elapsed_time,"saniye")
print("Geçen zaman:",time.strftime("%H:%M:%S",
time.gmtime(elapsed_time)),"saniye")
```

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı : Orkun HARAÇVERMEZ
Doğum Yeri ve Tarihi : Aydın, 12.01.1985

EĞİTİM DURUMU

Lisans Öğrenimi : Matematik Bölümü, Fen Edebiyat Fakültesi, Dokuz Eylül Üniversitesi
Yüksek Lisans Öğrenimi : Adnan Menderes Üniversitesi, Matematik A.B.D.
Bildiği Yabancı Diller : İngilizce

İŞ DENEYİMİ

Çalıştığı Kurumlar ve Yıl : Çumra Mesleki ve Teknik Anadolu Lisesi, Konya, 2016-2019

İLETİŞİM

E-posta : orkun_haracvermez@yahoo.com
Tarih : 01.06.2019