T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ZARARLI YAZILIM KAYNAKLI VERİ KAÇIRMA ATAKLARINA KARŞI DOKÜMAN SINIFLANDIRMA ALGORİTMASI GELİŞTİRME

YÜKSEK LİSANS TEZİ

Yahya KESENEK

Enstitü Anabilim Dalı	**:**	**BİLGİSAYAR VE BİLİŞİM MÜHENDİSLİĞİ**
Enstitü Bilim Dalı	**:**	**SİBER GÜVENLİK**
Tez Danışmanı	**:**	**Doç. Dr. İbrahim ÖZÇELİK**

Eylül 2019

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ZARARLI YAZILIM KAYNAKLI VERİ KAÇIRMA ATAKLARINA KARŞI DOKÜMAN SINIFLANDIRMA ALGORİTMASI GELİŞTİRME

YÜKSEK LİSANS TEZİ

Yahya KESENEK

Enstitü Anabilim Dalı	**BİLGİSAYAR MÜHDENDİSLİĞİ**
Enstitü Bilim Dalı	**BİLGİSAYAR VE BİLİŞİM**

Bu tez 09.09.2019 tarihinde aşağıdaki jüri tarafından oybirliği / oyçokluğu ile kabul edilmiştir.

Prof. Dr.	**Prof. Dr.**	**Doç. Dr.**
Cemil ÖZ	**Resul KARA**	**İbrahim ÖZÇELİK**
Jüri Başkanı	**Üye**	**Üye**

BEYAN

Tez içindeki tüm verilerin akademik kurallar çerçevesinde tarafımdan elde edildiğini, görsel ve yazılı tüm bilgi ve sonuçların akademik ve etik kurallara uygun şekilde sunulduğunu, kullanılan verilerde herhangi bir tahrifat yapılmadığını, başkalarının eserlerinden yararlanılması durumunda bilimsel normlara uygun olarak atıfta bulunulduğunu, tezde yer alan verilerin bu üniversite veya başka bir üniversitede herhangi bir tez çalışmasında kullanılmadığını beyan ederim.

Yahya KESENEK

…../…../2019

TEŞEKKÜR

Yaptığım çalışmada emeğini esirgemeyen başta tez danışmanım Doç. Dr. İbrahim ÖZÇELİK'e ve çalışmamız boyunca çalışmaya katkı sağlayan Emrah KAYA ve değerli çalışma arkadaşlarıma teşekkürü bir borç bilirim.

İÇİNDEKİLER

TEŞEKKÜR i
İÇİNDEKİLER ii
SİMGELER VE KISALTMALAR LİSTESİ v
ŞEKİLLER LİSTESİ vii
TABLOLAR LİSTESİ viii
ÖZET ix
SUMMARY x

BÖLÜM 1.
GİRİŞ 1
1.1. Literatür Çalışmaları 1
1.2. Tez Katkısı 5
1.3. Tez Organizasyonu 6

BÖLÜM 2.
ZARARLI YAZILIM VE MAKİNE ÖĞRENMESİNE AİT TEMEL KAVRAMLAR 7
2.1. Zararlı Yazılım ve Veri Kaçırma Atakları 7
2.1.1. Zararlı yazılım 7
2.1.2. Gelişmiş sürekli tehdit (APT) 7
2.2. Veri Kaçırma Atakları 8
2.2.1. Yapısal değişiklik saldırıları 9
2.2.1.1. Yer değiştirme (Transposition) saldırıları 9
2.2.1.2. Yapısal cümle değiştirime saldırıları 9
2.2.1.3. Şablon (Template) değiştirime saldırıları 9
2.2.1.4. Yerine koyma (Substitution) saldırıları 10

2.2.2. Dönüşüm (Transformation) saldırıları 10
2.2.2.1. Mapping veya hashing saldırıları..... 10
2.2.2.2. Şifreleme (Encryption) saldırıları 10
2.2.2.3. Rot (n) Şifreleme/Yerdeğiştirmeli şifreleme 11
2.2.3. Karartma (Obfuscation) saldırıları 11
2.2.3.1. Eş anlamlı (Synonym) kelime saldırıları 11
2.2.3.2. Çok anlamlı (Polysemy) kelime saldırıları 12
2.2.3.3. Kitap şifreleme (Book Cipher) saldırıları 12
2.3. Makine Öğrenmesi 12
2.3.1. Danışmanlı (Supervised) öğrenme 13
2.3.2. Danışmansız (Unsupervised) öğrenme..... 13
2.3.3. Takviyeli (Reinforcement) öğrenme 13
2.3.4. Makine öğrenmesinde kullanılan yöntem ve kavramlar 14
2.3.4.1. Sınıflama geçerliliği..... 14
2.3.4.2. Rasgele orman (Random forest) 15
2.3.4.3. Karar destek makineleri (Support Vector Machines) 17
2.3.4.4. Oylamalı sınıflandırıcı (Voting classifier)..... 22
2.3.4.5. Yapay sinir ağları (YSA) 23

BÖLÜM 3.
ZARARLI YAZILIM KAYNAKLI VERİ ATAKLARINA KARŞI DÖKÜMAN SINIFLANDIRMA ALGORİTMASININ GELİŞTİRİLMESİ 30
3.1. Giriş..... 30
3.2. DLP Veri Seti Oluşturma 31
3.2.1. Dyncorp..... 33
3.2.2. Mormon..... 33
3.2.3. Transcendental Meditation (TM) 33
3.2.4. DBpedia..... 34
3.3. Metin Çıkarımı 36
3.4. Metin Normalizasyonu ve Önişleme adımları 37
3.4.1. Noktalama işaretlerini kaldırma..... 39
3.4.2. Metnin küçük harfe çevrilmesi..... 39

3.4.3. Sayısal verilerin kaldırılması 39
3.4.4. Fonksiyonel kelimeler (Stop words) 39
3.5. Özellik Çıkarımı Ön İşlemleri 40
3.5.1. Token oluşturma (Tokenization) 40
3.5.2. Gövdeleme (Stemming) 41
3.5.3. Yazım Düzeltimi (Spell Correction) 43
3.5.4. Özellik çıkarımı 45
3.5.4.1. N-gram 46
3.5.4.2. K-skip-n-gram 47
3.5.4.3. Karakter N-gram 49
3.5.4.4. Gizli anlamsal analiz (Latent Semantic Analysis) 49
3.5.5. Özellik Seçimi - Terim ağırlıklandırma 50
3.5.6. Sınıflandırma 54
3.5.6.1. Rasgele orman (Random forest) 55
3.5.6.2. Karar destek makineleri (Support vector machines) 56
3.5.6.3. Çok katmanlı yapay sinir ağları (Multi layer perceptron) 57
3.5.6.4. Oylamalı sınıflandırıcı (Vote classifier) 58

BÖLÜM 4.
UYGULAMA VE PERFORMANS SONUÇLARI 60
4.1. Performans değerlendirme 60
4.1.1. F-skor (F-measure) 62
4.2. Veri Setlerine Göre Performans Değerlendirmesi 64
4.3 Doküman Tabanlı Sınıflandırmaya Göre Performans Değerlendirmesi 71

BÖLÜM 5.
SONUÇ VE ÖNERİLER 83

KAYNAKÇA 86
ÖZGEÇMİŞ 92

SİMGELER VE KISALTMALAR LİSTESİ

ABD	: Amerika Birleşik Devletleri
APT	: Advanced persistent thread
BY	: Atıf yapılabilir anlamındaki bir lisans türü
BOW	: Bag of word
CART	: Classifcation and regression tree
CC	: Creative commons
Char-gram	: Karakter gram
CNN	: Convolutional Neural Network
DDİ	: Doğal dil işleme
DLP	: Data loss prevention
G	: Gizli
SA	: Lisans devam anlamındaki lisans türü
SVM	: Support vektor machines
TF-IDF	: Term frequency-inverse document frequency
TM	: Transcendental Meditaion
KG	: Kurumsal Genel
KO	: Kurumsal Olmayan
LSA	: Latent semantic analysis
LSI	: Latent semantic index
L-BFGS	: Limited Memory Broyden–Fletcher–Goldfarb–Shanno
MLP	: Multi layer perceptron
NLP	: Natural language processsing
NLTK	: Natural language toolkit
IDF	: Inverse document frequency
RBF	: Radial basis function
Rf	: Random forest

SGD	: Schotastic gradient descent
SVD	: Singular value decomposition
T.C	: Türkiye Cumhuriyeti
TKA	: Tek katmanlı algılayıcı
YSA	: Yapay sinir ağları

ŞEKİLLER LİSTESİ

Şekil 2.1. İçerik veri kaçırma saldırı türleri 8
Şekil 2.2. Rot (13) şifreleme 11
Şekil 2.3. Rasgele Orman algoritması 17
Şekil 2.4. SVM çalışma yapısı 18
Şekil 2.5. Doğrusal olarak ayrılabilen veri kümesi için hiper-düzlem 19
Şekil 2.6. Doğrusal Ayrılamayan Veriler İçin Hiper Düzlem Belirleme 20
Şekil 2.7. Doğrusal olmayan veri setinin doğrusal olmayan haritalama yaklaşımı 21
Şekil 2.8. Oylamalı sınıflandırıcı genel çalışma şekli 23
Şekil 2.9. Yapay sinir hücre yapısı 24
Şekil 2.10. Tek katmanlı algılayıcı 26
Şekil 2.11. Basit Algılayıcı ağ modeli 26
Şekil 3.1. Metin sınıflandırma aşamaları 32
Şekil 3.2. Ön işleme aşamaları 38
Şekil 3.3. Özellik çıkarımı 46
Şekil 3.4. Karar destek makineleri sınıflama 56
Şekil 3.5. Svm önemli kernel çeşitleri 57
Şekil 3.6. Oylamalı sınıflandırma 59
Şekil 4.1: Test aşamaları 61
Şekil 4.2. TM test grafiği 68
Şekil 4.3. Mormon test grafiği 69
Şekil 4.4. Dyncorp test grafiği 69
Şekil 4.5. Kategori profilleri ile önerilen sistemin karşılaştırma grafiği 71
Şekil 4.6. Test 2 modeli ile CNN accuracy değerleri karşılaştırma grafiği 82

TABLOLAR LİSTESİ

Tablo 3.1. Veri Seti ve Etiket Tablosu 34
Tablo 3.2. Yöntem 1 aşamaları 35
Tablo 3.3. Yöntem 2 aşamaları 35
Tablo 3.4. Yöntem 3 aşamaları 35
Tablo 3.5. Yöntem 4 aşamaları 35
Tablo 3.6. Özellik çıkarım yöntemleri sonucu elde edilen özellik sayısı 53
Tablo 4.1. Hata Matrisi 61
Tablo 4.2. Veri setideki kullanılan eğitim ve test doküman sayısı 64
Tablo 4.3. Test verisine yapılan saldırı türleri. 65
Tablo 4.4. Yöntemlere ait test süreçleri 66
Tablo 4.5. TM test verisine yapılan saldırılara ait f-skor değerleri. 66
Tablo 4.6. Mormon test verisine yapılan saldırılara ait f-skor değerleri. 67
Tablo 4.7. Dyncorp test verisine yapılan saldırılara ait f-skor değerleri. 67
Tablo 4.8. Kategori profilleri yöntemi test sonuçları 70
Tablo 4.9. Veri setideki kullanılan eğitim ve test doküman sayısı 71
Tablo 4.10. Dokümana üzerine uygulanan saldırı vektörleri (genişletilmiş olarak) 72
Tablo 4.11. Yöntemlere göre test sonuçları 73
Tablo 4.12. Yöntem 2, Kategori profilleri ve SGD test sonuçları 76
Tablo 4.13. Tümleşik doküman üzerinde Test 2, CNN accuracy değerlerinin karşılaştırılması. 81

ÖZET

Anahtar kelimeler: Zararlı Yazılım Atakları, Döküman Sınıflama, Veri Kaçırma, Saldırı Altında Veri Sızıntısı Önleme, Gelişmiş Sürekli Tehdit

Kurumsal veya kurumsal olmayan değerli dokümanlar, erişim yetkisine sahip olmayan kişiler ya da uygulamalar tarafından ele geçirilerek kurum dışına çıkarılması veya sızdırılması günümüzde sıkça görülmektedir. Bu dokümanların sızdırılması ile kurum için değerli bilginin rakiplerinin eline geçmesi söz konusu olabilmektedir. Devlet kurumları için ise değerli bilginin ulusal bazda uygulanan politikaların değişmesine yol açabileceği gibi uluslar arasındaki ilişkilerin kopmasına da sebep olabilmektedir. Değerli bilginin kaçırılmasına yönelik yapılan Veri Sızıntısı Önleme (Data Leakage Protection – DLP) sistemleri genel olarak kural tabanlı, desen tabanlı ve istatistiksel yöntemler kullanmaktadır. Kural tabanlı ve desen tabanlı sistemler genel olarak eşleştirme algoritmaları kullanmaktadırlar. Eşleştirme algoritmalarının atlatılması dokümandaki küçük değişmelerle rahatlıkla yapılabilmektedir. İstatistiksel yöntemler, iyi olmalarına rağmen zararlı yazılım kaynaklı saldırılara karşı performansı düşebilmektedir. Bundan dolayı zararlı yazılımların kullandıkları sofistike yöntemlerin tespit edilerek, bu saldırılara karşı dayanıklı bir algoritmanın geliştirilmesi gerekmektedir.

Bu tezimizde, zararlı yazılımlar tarafından içeriklerin kaçırılması için yapılan saldırılar yeniden düzenlenerek, saldırı türlerine ait şema çıkarılmıştır. Bu şemadaki saldırı türlerinden, yapısal saldırılar ile karartma saldırılarına karşı bir çözüm önerisi sunulmuştur. Ayrıca ele alınan bu saldırıları gerçekleştirmeye yönelik bir yazılım geliştirilmiştir. Bu yazılım aracılığıyla dokümanlara saldırı yapılmış daha sonra geliştirilen yöntemin performansı ölçülmüştür. Geliştirilen algoritmada Doğal Dil İşleme (Natural Language Processing-NLP) yöntemleri, makine öğrenmesi ve yapay sinir ağları kullanılmıştır. Metin tabanlı sınıflandırma sistemlerinin sıkça kullandığı Doğal Dil İşleme algoritmaları özelliklerin çıkarılması aşamasında kullanılmıştır. Daha sonra sınıflandırma modelinde Karar Destek Makineleri (SVM), Rastgele Orman (Random Forest) ve Çok Katmanlı Sinir Ağları (Multi-Layer Perceptron) kullanılmıştır. Kullanılan bu sınıflandırma modellerinde karar mekanizması Oylamalı sınıflandırıcı (Vote Classifier) ile sağlanmıştır. Algoritmanın dayanıklılığı, veri sızıntısı önleme sistemleri ve metin sınıflandırma algoritmalarında kullanılan Kategori profilleri, SGD (Schotastic Gradient Descent) ve CNN (Convolutional Neural Network) yöntemleri ile karşılaştırılarak algoritmanın başarısı ölçülmüştür. Yapılan testlerde önerdiğimiz yöntemin sınıflama başarısına ait f1 skoru %99 olarak bulunmuştur.

DEVELOPING DOCUMENT CLASSIFICATION ALGORITHM AGAINST MALICIOUS DATA LEAKAGE ATTACKS

SUMMARY

Keywords: Malware attack, Document Classification,Data Leakage, Data Leakage Protection Against Malware Attacks, Advenced Persistent Attack

It is often seen that institutional or non-institutional valuable documents are seized by persons who are not authorized to access them and taken out or leaked. With the leakage of these documents, valuable information for the private enterprise may be passed into the hands of its opponents. Valuable information leakage can lead to changes in the policies applied on a national basis, as well as breaking the relations between nations. Data Leakage Protection (DLP) systems uses rule-based, pattern based and statistical methods. Rule based and pattern based systems generally use matching algorithms. Bypassing the matching algorithms can be easy done with small changes in the document. Although statistical methods are well but against of an attack it may decrease. Therefore it is necessary to identify the sophisticated methods used by malwares and develop an algorithm that is resistant to these attacks.

In this thesis, malware-based attacks are re-organized and attack types are shown in a schema. In this study, two type of attacks, which are structural attack and obfuscated attack are scoped. A software has been developed to carry out these attacks. With this software, the documents were attacked and then the performance of the developed method was measured. In the developed algorithm, Natural Language Processing (NLP) methods, machine learning and artificial neural networks were used. Natural language processing algorithms, which are commonly used by text-based classification systems, are used in the extraction of features. Later, Decision Support Machines (SVM), Random Forest and Multi-Layer Perceptron were used in the classification model. In these classification models, the decision mechanism is provided by Vote Classifier. The reliability of the algorithm was compared with several methods used in data leakage prevention systems and text classification algorithms and the success of the algorithm was measured.In the tests performed, the f1 score of the classification success of the proposed method was found to be 99%.

BÖLÜM 1. GİRİŞ

Veri sızıntısı özellikle ulusal ve kurumsal ağlarda büyük sorunlara yol açabilmektedir. Veri kaybının yaşandığı durumlarda hem ülke hem de kurumlar büyük yara alabilmektedir. Gizlilik arzeden bilgilerin kurum dışında veya içinde yetkisiz ellere geçmesi ülkenin menfaatine zarar vermektedir. Bazen verilerin sızdırılması uluslararası arenada imaj kaybına yol açabileceği gibi uluslararası ilişkilerin gerilmesine veya tamamen kopmasına da sebep olabilmektedir. Bu tür veri sızıntısını önlemeye yönelik olarak geliştirilen veri sızıntısı önleme (DLP) sistemleri günümüzde yaygın olarak kullanılmaktadır. Ücretli ve ücretsiz sürümlerinin bulunduğu bu programlara ek olarak güvenlik firmaları bu yapıları antivirus sistemlerine entegre ederek tümleşik bir yapı sunmaktadır.

Veri sızıntısı alanında yapılan çözümlerin ortak noktası, veriler üzerinde herhangi bir saldırının yapılmamış olması, yani verinin temiz olması, varsayımıdır. Geliştirilen çözümler saldırı vektörlerinin olmadığı durumlarda yüksek oranda başarım sağlıyorken yapılacak basit bir saldırıda bile başarım beklendiği gibi yüksek olmadığı görülebilmektedir. Bu durumda veri sızıntısı çözümlerinin yeniden ele alınarak geliştirilmesi gerekmektedir. Yapılan saldırılar altında içeriğin doğru sınıflandırılması için ortak veya tekil çözüm geliştirilmesi gerekmektedir.

1.1. Literatür Çalışmaları

Veri sızıntısı engellemek amacıyla ticari olan birçok firma veri sızıntısına yönelik çözümler üretmektedir. Bu ürünleri çoğu basit algoritmalar kullanmaktadır. Michael Hart ve ark. [37] veri sızıntısı ile ilgili yaptıkları çalışmada mevcut çözümlerin, bilginin açığa çıkarılması açısından tamamen gelişmiş olmadığını belirterek, aynı zamanda hassas verinin yakalanması açısından da eksikliklerin olduğunu

belirtmektedir. Radwan ve Yousef [38] veri sızıntısı ile ilgili yaptıkları literatür çalışmasında, mevcut veri sızıntısı çözümleri karartılmış veri kaçaklarını bulamadığını ifade etmektedir. Ayrıca hassas verinin diğer dosyalarla değiştirilmesi sonucu da veri kaçaklarının gerçekleşebileceğinden bahsetmektedir. Sultan Alneyadi ve akr. [39] veri sızıntısı ile ilgili yaptıkları çalışmada, veri sızıntısı önlemek için parmak izi (finger printing), kurallı ifadeler (regular expression) ve istatistiksel yöntemlerin kullanıldığını belirtermektedir. Parmak izi ve kurallı ifadeler anlamsal analizde başarısız olduğunu bu yüzden istatistiksel yöntemlerle bu başarısızlığın giderilebildiğini ifade etmektedir.

Sultan Alneyadi ve akr. [39] veri sızıntısı ile ilgili çalışmalarında eş anlamlı kelimeler (word synonyms), ekleme (addition), çıkarma (substraction) gibi veri değişikliklerine karşı k-skip-n-gram yönteminin başarılı olduğunu belirtmiştir. Bu yöntemle hassas veriler ile hassas olmayan verilerin k-skip-n-gramları çıkarılarak ayrı ayrı skip-gramlar oluşturulur ve elde edilen hassas olmayan k-skip-n-gramlar, hassas veri içerisindeki gereksiz n-gramların elimine edilmesi için kullanılır. Bu yöntemde verilerin indekslenmesi gerekmektedir. Bunun da ekstra alan gerektirdiği ve k-skip gram yerine ters metin frekansının (TD-IDF) kullanılmasının daha uygun olabileceği belirtilmektedir.

Tarique Mustafa, [4] mevcut veri sızıntısı ile ilgili yaptığı çalışmada mevcut veri sızıntısı çözümlerinin, zararlı yazılımların değiştirdiği veriyi yakalamadaki eksiklileri ortaya koymaktadır. Yavuz Canbay ve ark. [6] yaptığı çalışmada hassas veriler üzerinde yapılan modifikasyonları tespit etmek için bir model önermiştir. Bu modelde hassas veriler LSA yöntemiyle hassas olarak sınıflandırıldığında hassas veri içerip içermediği Smith Waterman ve Boyer Moore algoritmalarının ardışık uygulanmasıyla bulunabileceği belirtilmiştir. Burada ele alınan yöntemler önemli olmakla beraber LSA sınıflandırma aşamasında sızıntının olması durumunda çaresiz kalmaktadır. Burada dökümanın öncelikle doğru sınıflandırılması sağlanılmalıdır. Ayrıca hassas verinin içinde bulunduğu dökümanın tamamına yönelik yapılacak bir saldırıda yine bu yöntem zayıf kalmaktadır. Sınıflandırmada karşılaşılan bir diğer sorun, metnin içerisindeki hatalı yazılmış sözcüklerin olması durumudur. Bu durumda sözcüklerin

düzeltilmesi gerekir. Bruna Martins ve Mario J. Silva [40] yazım hatalarını düzeltme ile ilgili yaptıkları çalışmada yazım hatalarının oluştuğu durumları incelemiştir.

Bruna Martins ve Mario J. Silva'nın belirttiği yazım hatalarının dışında da çeşitli şekillerde yazım hataları olabilmektedir. Bu tür hatalarının bulunup düzeltilmesinin daha zor olduğu durumlar da vardır; örneğin, kelimenin doğru yazılması fakat anlamca bakıldığında orada olmaması gerektiği gibi. Farag Ahmed ve ark.[41] kelime düzeltme ile ilgili yaptıkları çalışmada, dilden bağımsız bir düzeltme modeli önermektedir. Bu çalışmada n-gram tabanlı bir düzeltme modelini önererek iyi sonuçlar elde ettiklerini ifade etmektedirler Yazım hatası düzeltmede bulanık arama işlemlerinde edit distance sıkça kullanılan etkili bir yazım düzeltme yöntemidir. Ayrıca doğal dil işleme yönteminde kullanılan kelime n-gramları kasti olarak yapılan saldırıların tespitinde yetersiz kalmaktadır. Bu durumlarda karakter n-gramlarla özellik çıkarım yöntemiyle dökümana ait bilgiler az da olsa bulunabilmektedir. Ayrıca kısa metinlerde sınıflamanın etkili olabilmesi için döküman hakkında daha fazla bilgiye ihtiyaç duyulmaktadır.

Skipgram'lar kelimenin anlamını çıkarmada vektör uzay modelinde kayda değer bir efektif sağlamaktadır. N-gramları çıkarmada skipgramlar farklı bir parametere olan k değerini kullanarak aradaki uzaklığa göre n-gramlar oluşturmaktadır [68]. Kısa metinlerde skipgramların kullanılması sınıflama için yeteri kadar bilginin sağlanmasını sağlamaktadır ayrıca karakter n-gramlar ile de kelime n-gramlarına ayrılamayan metinler hakkında daha fazla bilginin elde edilmesi sağlanabilmektedir.

Tf-idf, dokümanları vektör uzay modelinde tanımlayabilmemiz için kullanılan en önemli ağırlıklandırma metotlarından biridir. Tf-idf ağırlıklandırmasında her bir dokümandaki kelimelerin frekansı rol oynamaktadır. Böylece dokümanda daha fazla görülen kelimeler varsa o doküman için daha değerli olduğu anlaşılmaktadır. Ayrıca IDF tüm dokümanlarda seyrek görülen kelimeler ile ilgili bir ölçü vermektedir. Bu değer tüm eğitim dokümanlarından hesaplanmaktadır. Bu yüzden eğer bir kelime dokümanlarda sık geçiyorsa doküman için belirleyici olmadığı düşünülebilir [4]. Eğer kelime dokümanlarda çok sık geçmiyorsa o kelimenin o doküman için belirleyici

özelliği vardır denilebilir. Tfx-idf genel olarak sorgu vektörü ile eğitim dokümanı vektörü arasındaki benzerlik oranını bulmak için kullanılır. Tf-idf fonksiyonunun çeşitli versiyonları mevcuttur [6].

Ters belge frekansının temelindeki fikir, "bir terimin, veri seti içinde gözlendiği belge sayısı azaldıkça, gözlendiği belgeler açısından ayırt ediciliğinin artacağı" şeklindedir [70].

LSA yöntemi, kelimeler arasındaki anlamsal bağlantıyı bulmak için kullanılan bir yöntemdir. LSA, bir veri seti (corpus) kullanarak veriyi öğrenmektedir. Veriyi öğrenme aşamasında Kırpılmış SVD (Truncated SVD) kullanarak boyut azaltımı yapılmakta daha sonra uzaklık ölçümüne göre kelimeler arasındaki bağ bulunmaktadır [3]. LSA yöntemi özellikle karartılmış verinin üzerinde kullanılması ile kelimelerin anlamı belirginleştirilmektedir. Eş anlamlı ve çok anlamlı kelime saldırılarına karşı LSA özellik çıkarım yöntemi önem arzetmektedir. Özellik çıkarımı yapıldıktan sonra verilerin sınıflandırıcıya verilip eğitilmesi gerekmetedir. Sınıflandırma işlemi için bir çok yaklaşım kullanılmaktadır. Karar destek makineleri (SVM) istatistiksel bir öğrenme kuramıdır. Sınıflama ve regresyon için kullanılabilir [28]. Bu algoritma ile n özelliğe (feature) sahip bir veri kümesini bir düzlem (hyper-plane) ile ayırmaya çalışır.

Karar ağaçları birçok makine öğrenme yönteminde sıklıkla kullanılmaktadır. Öğrenme yöntemleriyle karşılaştırıldığında rasgele orman sınıflandırıcısı, özellikle hızlandırma yöntemine göre, eğitim aşamasında çok daha hızlıdır. Yeterliliği ve doğruluğu ile çok kullanışlı bir sınıflandırıcıdır. Hızlandırma yönteminin hesap yükü fazla olduğu için Torbalama yönteminden daha yavaştır. Fakat çoğu durumda bu yöntemden daha doğru sonuçlar verir. Hızlandırma yönteminin, çok yavaş ve gürültüye karşı duyarlı olması, tekrarlı eğitimin olabilmesi gibi dezavantajlarına karşın rasgele orman, hesapsal olarak hızlandırma sınıflandırıcısından çok daha basittir, gürültüye karşı duyarlı değildir.Karar Ağaçları, tüm değişkenler arasından en iyi dalı kullanarak her bir düğümü dallara ayırmak yerine, her bir düğümde rastgele olarak seçilen değişkenler arasından en iyisini kullanarak her bir düğümü dallara ayırır. Her bir veri seti orijinal veri setinden yer değiştirmeli olarak üretilir. Sonra rastgele özellik seçimi kullanılarak

ağaçlar geliştirilir. Geliştirilen ağaçlar budanmaz. Bu strateji Rasgele Ormanı'nın doğruluğunu eşsiz yapar. Rasgele Orman aynı zamanda çok hızlıdır, aşırı uyuma karşı dayanıklıdır ve ne kadar istenirse o kadar ağaçla çalışılır [50].

Perceptron, doğrusal olmayan çözümler üretemediği için hem mimari hem de eğitim algoritması açısından iyileştirilmiş Çok Katmanlı Algılayıcı (MLP) ağı önerilmiştir. Mimari açıdan doğrusal olmayan aktivasyon fonksiyonuna sahip birçok nöronun birbirine hiyerarşik olarak bağlandığı bir yapıya sahip olan MLP, Algılayıcı ve Adaline yöntemlerinin avantajları yanı sıra geri-yayılım adındaki öğrenme sistemini kullanmaktadır.

Topluluk yöntemleri (Ensemble Methods) genel olarak bir hesaplama yönteminin güvenirliğini geliştirmek için kullanılır. Bu yöntemler verilen öğrenme algoritmasını geliştirmek için çeşitli temel yaklaşımlar birleştirerek tahmini iyileştirmeyi amaçlar. Genel olarak iki topluluk metodu çokça kullanılmaktadır. Bunlardan ortalamalı yönteminde (averaging methods), bağımsız olarak geliştirilen temel yaklaşımlar uygulandıktan sonra bunların ortalaması alınır. Burada varyans değeri düştüğü için tekil olarak kullanılan yaklaşımdan daha iyi sonuç verir. Diğer yöntem hızlandırmalı yöntem (boosting methods), bu yaklaşımda temel hesaplayıcılar ardışıl olarak uygulanırken bir tane hesaplayıcı birleştirilmiş yaklaşımın bias (eğilim) değerini düşürmeye çalışır. Bu yöntem genel olarak zayıf model algoritmalar üzerinde kullanılarak daha güçlü algoritmalar elde etmede kullanılır. Oylamalı sınıflandırıcıları, değişik makine öğrenme algoritmalarının uygulanması sonucu en çok oy alan veya ortalama tahmin değerine (soft vote) bağlı olarak dökümanı sınıflandırır. Bu şekilde algoritmalardaki zayıflıkları azaltarak daha iyi sonuç üretmesini sağlanır [33].

1.2. Tez Katkısı

Veri sızıntısı önleme sistemleri, genel olarak dökümandaki hassas veriye yönelik saldırı tespitinde veya dökümandaki modifikasyonlarda yapılan değişikliğin kullanıcı hatasından kaynaklandığı varsayımı altında sınıflama yapmaktadır. Oysaki bu tür durumların dışında da veri sızıntısı olmaktadır. Verinin sızdırılmasını önlemek için

kullanılan kurallı ifadeler, eşleştirme ve parmak izi çalışmaları dökümana yapılacak saldırı durumunda yeteri derecede başarılı olamamaktadır. Ayrıca bu tür sistemler ele aldığımız saldırılara kapsamlı bir çözüm sunamamaktadır. Günümüzde zararlı yazılımlar bilgisayarda kalıp bilgisayarı bozmaktan çok kişinin bilgilerini çalan veya gerektiğinde gizlilik ihlali yapabilecek potansiyele sahip olabilen yazılımlar olarak üretilmektedir. Bu yazılımlar, gelişmiş yapıları sayesinde antivirüs çözümlerine takılmadan istediği işlemi rahatça yapabilmektedir. Bu durumlarda mevcut sistemin hassas veriyi güvenli tutması gerekmektedir. Fakat mevcut sistem tam olarak beklendiği gibi veriyi güvende tutamamaktadır. Tezimizde bu konudaki eksiklikleri irdeleyerek mevcut saldırıları gerçekleyen bir yazılım oluşturulmuştur. Bu yazılım aracılığıyla saldırılar gerçekleştirilmiş ve bu saldırılar altında dahi dokümanları daha iyi sınıflayabilen bir çözüm yöntemi önerilmiştir. Tezimizde, yaptığımız çalışmalar sonucu yapısal saldırı ve karartılmış saldırıların sonucunda oluşan dökümanları sınıflamak için yazım düzeltimi, kelime n-gram, karakter n-gram, k-skip-n-gramlar ve LSA yöntemlerinin özellik çıkarımı aşamasında sınıflandırıcı için etkili olduğu görülmüştür. Ayrıca eşleştirme yöntemlerinin zayıflığından dolayı, bunları kullanmak yerine makine öğrenmesi ve yapay sinir ağlarının kullanılması daha iyi sonuçlar verdiği görülmüştür.

1.3. Tez Organizasyonu

Bölüm 1'de yapılan çalışmaya ait çözümler incelenerek yapılacak çalışmanın amacı ve metotu, yapılan çalışmanın mevcut sisteme katkısı anlatılmıştır. Bölüm 2'de zararlı yazılımlar, gelişmiş sürekli tehdit (APT), makine öğrenmesi ve derin öğrenme yöntemlerine ait temel kavramlar anlatılmıştır. Bölüm 3'te kullanılan materyal ve metotlar ele alınmıştır. Bölüm 4'te ise önerilen sisteme ait uygulama ve performans sonuçları değerlendirilmiştir.

BÖLÜM 2. ZARARLI YAZILIM VE MAKİNE ÖĞRENMESİNE AİT TEMEL KAVRAMLAR

2.1. Zararlı Yazılım ve Veri Kaçırma Atakları

Zararlı yazılımlar, genel olarak bilgisayar sistemine zarar veren programlar olarak üretilmelerine rağmen son zamanlarda bu işlem yerine kullanıcıların verilerini kaçırmaya çalışan veya kritik sistemlerde ani müdahaleler yapabilmek için kullanılan birer araç haline getirilmiştir. Antivirüs gibi sistemler zararlı yazılımların çoğalmasını ve yayılmasını engellemeye yönelik çalışmalar yapmakta olup bunların yeni zararlı yazılımlara karşı çok da etkili olamadığı haber bültenlerinde sıkça geçmektedir.

2.1.1. Zararlı yazılım

Zararlı yazımlar kullanıcı tarafından izin verilmeyen işlemler gerçekleştiren kötü amaçlı programlardır [35]. Zararlı yazılımlar kullanıcı verisi silme, engelleme, kopyalama, değiştirme, çalma, bilgisayar ve bilgisayar ağlarının performansını düşürme gibi zaralı amaçlar için programlanmaktadırlar. Zararlı yazılımlar kullanıcı sistemlerine internet üzerinden bulaşabildiği gibi, harici diskler, USB'ler gibi harici medya üzerinden de bulaşabilir. Zararlı yazılım türleri arasında bilgisayar virüsleri, solucanlar, casus yazılımlar, reklam yazılımları, truva atları ve botnetler sayılabilir.

2.1.2. Gelişmiş sürekli tehdit (APT)

Kullanıcı veya kurum bilgisi toplama amacı ile yapılan hedef odaklı saldırılardır. Belirli bir hedefe veya kuruma yönelik olması, geleneksel güvenlik mekanizmalarını atlatabilmesi ve hedef kurumlarda uzun süreli olarak faaliyet göstermesi bu saldırıların tespitini zorlaştırmaktadır [4].

Özellikle APT saldırıları, veri kaçırma alanında mevcut sistemlerin çalışmasını zorlaştıran ve yakalanma olasılığını düşüren bir saldırı türüdür. Kurumdan veri kaçırmak için kullandığı çeşitli yöntemler sayesinde kolaylıkla verileri veya dökümanları kaçırabilmektedir.

2.2. Veri Kaçırma Atakları

Veri kaçırma saldırıları; dökümanı, izinsiz ve yetkisiz olduğu halde, elde etmeye yönelik yapılan zararlı faaliyetlerdir. Bu çalışmalar hakkında çok fazla çalışma yapılmamıştır. Burada esas olarak ele alabileceğimiz iki farklı çalışma bulunmaktadır. [4] Çalışmasında içerik tabanlı saldırı türleri ele alımıştır. Ayrıca [6] çalışmasında da kelime üzerinde yapılabilecek saldırılar ele alınmıştır. Kendi çalışmamızda [6] çalışmasındaki modifikasyon saldırıları, [4] çalışmasında ise yer değiştirme, yerine koyma, eş ve çok anlamlı saldırılar ele alınarak, bunlara yönelik bir çözüm önerisinde bulunulmuştur. Tezimizde, [4] ve [6] çalışmaları birleştirilerek içerik tabanlı yapılabilecek saldırı türleri (Şekil 2.1.)'de gösterilmiştir.

Şekil 2.1. İçerik veri kaçırma saldırı türleri

Bu tez içerisinde yapısal saldırılar ile karartma saldırıları türleri ile alakalı çalışmalar yapılmıştır. Bu amaçla içi dolu olarak gösterilen saldırı vektörleri gerçeklenmiş ve önerdiğimim çözümün performans bu saldırılar üzerinden ölçülmüştür.

2.2.1. Yapısal değişiklik saldırıları

Yapısal değişiklik saldırıları verilen dökümandaki cümlelerin veya kelimelerin tamamen değitirilmesi veya yeniden oluşturulması yöntemlerini kapsamaktadır. Bu yöntemlerle yapılan saldırılar rahatlıkla veri sızıntısına yol açabilmektedir. Bu yöntemlerin asıl amacı, özellikle eşleştirme algoritmalarını ve içerik tanıma sistemini atlatmaktır. Ayrıca yapısal değişiklik saldırıları, birden fazla saldırıyı aynı anda gerçekleştirerek daha etkili olması sağlanabilir.

2.2.1.1. Yer değiştirme (Transposition) saldırıları

Bu saldırı türünde verilen bir belgenin, cümlelerinin veya bölümlerinin sırasını bozacak şekilde hareket ettirmekle gerçekleştirilmektedir [4].

2.2.1.2. Yapısal cümle değiştirime saldırıları

Bir metindeki bir veya birden fazla cümlenin anlamı aynı kalacak şekilde, cümlenin yapısını değiştirerek dökümanın veya içeriğin farklı görünmesini sağlama yöntemine dayanır [4].

Bu yönteme bir örnek verirsek, “X şirketi satın alınacak” cümlesi yeniden tasarlanarak “Bir firma X şirketini alacak” şeklinde anlamı bozulmayacak şekilde oluşturulmasıdır.

2.2.1.3. Şablon (Template) değiştirime saldırıları

Bu tür saldırılar, kullanıcı tarafından girilmiş şablonları veya program tarafından oluşturulmuş hazır şablonları kapsamaktadır. Örneğin bir dosya türü, word dosyası olması, yasal terimleri içermesi ve bazı tanımlayıcı numaralar içermesi (id numarası) durumunda insan kaynakları tarafından yazıcıya gönderilebilir olarak tanımlarken, aksi durumda bloklanmasını sağlayacak bir şablon oluşturulursun [45]. Bu şablonu değiştirerek tüm durumlarda gönderilebilmesini sağlamak veya yazıcıya gönderilmesini engelleyerek sadece diğer kanallarla gönderilebilmeyi tanımlamak bu

saldırı türüne örnek verilebilir. Ayrıca kredi kartı veya T.C kimlik numaralarının formatını değiştirerek mevcut şablondan atlatılma işlemi sağlanabilir.

2.2.1.4. Yerine koyma (Substitution) saldırıları

Bu saldırı türü, bir dökümandaki bir kelimenin veya bir ifadenin, bir cümlenin herhangi bir parçasını veya dökümanın bir bölümünü ifade edecek şekilde yer değiştirmesidir. [4] Bu saldırı türüne örnek olarak "Ayşe tatile çıksın." cümlesindeki Ayşe isminin askerleri, tatile çıksın ifadesi ise saldırma eylemini temsil etmesi.

2.2.2. Dönüşüm (Transformation) saldırıları

Bu saldırı türünde dökümanlar bir dönüşüm fonksiyonu kullanılarak sayısal veya metinsel çıktılar elde edilmektedir. Bu yöntemler genelde en çok kullanılan veri kaçırma yöntemleri olup yakalanması veya verinin içeriğinin anlaşılması çok zordur.

2.2.2.1. Mapping veya hashing saldırıları

Haritalama (Mapping) F(Map) veya Hashing denilen H(C1..CK) fonksiyonu kullanarak verinin sözcüksel olmayan tekil bir seriye dönüştürülmesini kapsamaktadır [4]. Bu yöntemler genel olarak SHA1,SHA2 gibi hashing yöntemleri ile kullanıcı tanımlı haritalama (Mapping) fonksiyonlarını kapsamaktadır. Haritalama fonksiyonlarına bir örnek olarak,

map.text("county", "new jersey") # New Jersey counties
map.text("world", "ira") # iran and iraq
verilebilir [36].

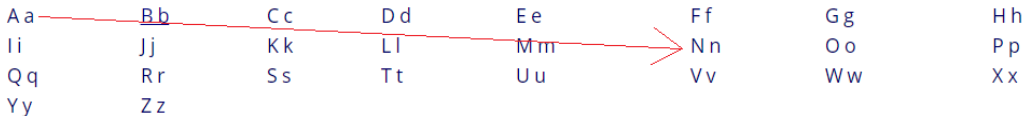
2.2.2.2. Şifreleme (Encryption) saldırıları

Şifreleme saldırıları, şifreleme algoritmaları kullanılarak verilen metnin şifrelenmesidir. Bu şifreleme yöntemlerinin kırılma olasılığı oldukça düşüktür. Çünkü

şifreleme anahtarı bilinmeden metnin çözülmesi için çok zaman almaktadır. Ayrıca basit şifreleme anahtarının seçiminde dahi, algoritmanın herhangi bir bilgiye sahip olmadığını varsayarsak, metnin çözülmesi tamamen şansa bağlıdır.

2.2.2.3. Rot (n) Şifreleme/Yerdeğiştirmeli şifreleme

Rot (n) şifreleme yönteminde verilen n sayısı kadar alfabedeki harf ileri veya geriye doğru kaydırılmaktadır. Bu aynı zamanda sezar şifreleme yöntemine de bezer. Örneğin n sayısını 13 aldığımızda a harfi n harfine denk gelmektedir (Şekil 2.2.). Bu şekilde basit bir şifreleme işlemi yapılabilir. Burada ele alınan şifreleme yöntemine ek olarak Sezar şifreleme de bu yöntem içerisinde değerlendirilebilir.



Şekil 2.2. Rot (13) şifreleme

2.2.3. Karartma (Obfuscation) saldırıları

Bu tür saldırılar kelimenin veya cümlenin anlamını gizlemeye yönelik saldırı türleridir. Burada asıl amaç sınıflama algoritmasını atlatmak için önemli veriyi önemsiz gibi göstermeye çalışmak veya veriyi gizlemektir.

2.2.3.1. Eş anlamlı (Synonym) kelime saldırıları

Verilen bir kelime yerine eş anlamlısını kullanarak veriyi kaçırma yöntemidir [4]. Bu yönteme örnek verirsek “Güçlü bir rakibi vardı” cümlesindeki “güçlü” kelimesinin eş anlamlısı olan “kuvvetli” kelimesi ile yer değiştirerek “Kuvvetli bir rakibi vardı” cümlesine dönüştürmek.

2.2.3.2. Çok anlamlı (Polysemy) kelime saldırıları

Bazı kelimelerin birden çok anlamı olabilmektedir. Bu kelimelerin hangi anlamda kullanıldığını açığa çıkarmak çoğu zaman zor olabilmektedir. Bu nedenle çok anlamlı kelimeler, anahtar kelimeler yerine kullanarak verinin/içeriğin karartması sağlanabilir. Bu şekilde yapılan saldırı ile verinin eşleştirme algoritmalarından kaçırılması rahatlıkla yapılabilmektedir [4].

2.2.3.3. Kitap şifreleme (Book Cipher) saldırıları

Belli bir kelime veya kelimler için şifre kullanmakla gerçekleştirilmektedir [4]. Örneğin, T.C kimlik numarasını kaçırmak için sayılar yerine onların kelime karşılığını kullanmak, bir harf yerine onun karşılığı olan sayının kelime karşılığını kullanmak.

Tezimizde yapısal ve karartma saldırılarına karşı bir çözüm geliştirilmiştir. Yapısal saldırılardan modifikasyon, yer değiştirme ve yerine koyma saldırıları ile karartma saldırılarından çok anlamlı ve eş anlamlı kelime saldırıları modellenerek test veri setlerimize uygulanmıştır. Daha sonra buradaki saldırılar geliştirdiğimiz algoritma tarafından teste tabi tutularak sonuçlar gözlemlenmiştir.

2.3. Makine Öğrenmesi

Makine Öğrenmesi, verilen bir problemi probleme ait ortamdan edinilen veriye göre modelleyen bilgisayar algoritmalarının genel adıdır. Yoğun çalışılan bir konu olduğu için önerilmiş birçok yaklaşım ve algoritma mevcuttur. Bu yaklaşımların bir kısmı tahmin (prediction) ve kestirim (estimation) bir kısmı da sınıflandırma (classification) yapabilme yeteneğine sahiptir.

Tahmin (prediction): Veriden öğrenen modellerde sistem çıkışının nicel olması durumunda kullanılan yöntemlerin ürettiği değerlerdir.

Sınıflandırma (classification): Giriş verisine ait çıkışların nitel olduğu durumlarda Kullanılan yöntemlerin her veri örneğinin hangi sınıfa ait olduğunu belirlemesidir. İstatistikte rasgele bir değişkenin bilinmeyen bir değerinin belirlenmesi için tahmin (prediction), bilinmeyen bir sabitin belirlenmesi içinse kestirim (estimation) kavramı kullanılır.

2.3.1. Danışmanlı (Supervised) öğrenme

Eğitim verileri üzerinden bir fonksiyon üreten bir makine öğrenmesi tekniğidir. Başka bir deyişle, bu öğrenme tekniğinde girdilerle (işaretlenmiş veri – labelled data) ile istenen çıktılar arasında eşleme yapan bir fonksiyon üretir. Eğitim verisi hem girdilerden hem çıktılardan oluşur. Fonksiyon, sınıflandırma (classification) veya eğri uydurma (regression) algoritmaları ile belirlenebilir [43].

Danışanlı öğrenmede herbir veriye ait çıktı önceden bellidir. Algoritma bu veriler üzerinde çalışarak özellikler çıkarmaktadır. Sonda da bu özellikleri kullanarak gelen yeni verinin sınıflanması sağlanmaktadır.

2.3.2. Danışmansız (Unsupervised) öğrenme

Bu yöntemde işaretlenmemiş (unlabelled) veri üzerinden bilinmeyen bir yapıyı tahmin etmek için bir algoritma kullanan makine öğrenmesi tekniğidir. Burada giriş verisinin hangi sınıfa ait olduğu belirsizdir. Bu yöntemde önceden belirlenmiş etiketler (labels) olmadığı için algoritma çeşitli yöntemlerle benzer veya aynı özelliğe sahip verileri bir kümeye yerleştirmektedir. Bu algoritma genel olarak kümeleme ve boyut kestirimi yöntemlerinde sıkça kullanılmaktadır.

2.3.3. Takviyeli (Reinforcement) öğrenme

Amaç odaklı bir yöntem olduğu için diğer iki öğrenme yöntemine göre biraz farklılıklar içermektedir. Aslında davranış psikolojisinden yola çıkan bir öğrenme yöntemidir [24].

Öğrenen etkene ajan (agent) denilmektedir. Ayrıca ajanlar çevreden bir geri bildirim almakta olup bunlara ödül (reward) denilmektedir. Burada asıl amaç ajanın çevreyle etkileşerek çevreden geri bildirim alıp ödülleri maksimuma çıkartarak hareket tarzını (optimum policy) bulmasıdır.

2.3.4. Makine öğrenmesinde kullanılan yöntem ve kavramlar

Sınıflandırma işleminin değerlendirilebilmesi için veri setleri eğitim ve test verilerine ayrılması gerekmektedir. Eğitim ve test veri setine ayrılan bu verilerden eğitim verisi modeli eğitmek, test verisi ise modelde görünmeyen, yeni verinin tahmin etmesi için kullanılmaktadır. Modelde test verilerine ait sınıflar ile sınıflandırıcının tahmin ettiği sonuçlar karşılaştırılarak modelin başarısı belirlenmektedir. Modelde eğitim ve test verilerinin uygun olarak parçalanıp uygulanması aşamasında, eğitim ve test verilerinin, sınıflama geçerliğini sağlaması gerekmektedir.

2.3.4.1. Sınıflama geçerliliği

Veriye dayalı olarak eğitim yapılmasının temel amacı eğitilen sistemin benzer konuda hiç bilinmeyen bir örneğe mantıklı bir cevap üretebilmesidir. Eldeki sınırlı veri kullanılarak sistemin hem eğitilmesi hem de başarısının tarafsız bir şekilde ölçülebilmesi gerekmektedir. Bunun için çapraz geçerlik (cross validation) adıyla anılan yöntemler kullanılmaktadır.

Bu temel prensibe dayanarak önerilmiş birkaç geçerlik yöntemi vardır. Ama hepsinde temel mantık aynıdır. Sistemin başarısını ölçebilmek için mevcut veri kümesi ikiye bölünür. Birisi eğitim için (train set) diğeri de sistemin hiç görmediği olası örnekleri temsilen (test set) kullanılır. Sistem, seçilen eğitim algoritması ile eğitim kümesini öğrenir. Eğitilen sistemin başarısı daha sonra test kümesi üzerinde hesaplanır [46].

Bu yöntemlerin dışında, dışarda tutma geçerlilik yöntemi de mevcuttur. Çalışmamızda saldırı vektörlerinin simule edilebilmesi için dışarda tutma yöntemi en uygun yöntem olarak belirlenmiştir.

2.3.4.1.1. Dışarda tutma

Veri setinin eğitim ve test olmak üzere 2 parçaya ayrıldığı yöntemdir. Test setinde kullanılan veri, eğitim setinin dışındaki verilerden oluşur. Eğitim veri seti ile öğrenme sağlanmakta, test veri seti ile de öğrenmenin ne kadar gerçekleştiği kontrol edilmekte, model performansı elde edilmektedir [47].

2.3.4.1.2. Aşırı eğitim (Overfitting)

Veri ile iterasyonal eğitim yapan modellerin hepsinde öğrenme sürecinin zamanı gelince durdurulması gerekir. Eğitim durdurulmazsa öğrenilmesi gereken veri içerisindeki tüm örnekler sistem tarafından ezberlenir ve bilinmeyen örnekleri tahmin yeteneği azalır. Aşırı eğitim (overfitting) denilen bu istenmeyen durum makine öğrenmesinin temel amacı olan genelleştirme kavramına zarar verir.

2.3.4.2. Rasgele orman (Random forest)

Rasgele orman veya rasgele karar ağaçları topluluk (ensemble learning) öğrenme yöntemleri sınıflandırma, regressyon ve diğer görevler için kullanılır. Rasgele karar ağaçları, karar ağaçlarının aşırı uyum davranışını çalışma zamanında (training time) düzeltmektedir. Algoritma ilk olarak Tin Kam Ho tarafından, rasgele altuzay örnekleme, Eugene Kleinberg tarafından öne sürülen "stokastik ayrımcılık" yaklaşımıyla ele alınan sınıflandırmayı çözmeyi amaçlıyordu [48].

Karar ağaçları birçok makine öğrenme yönteminde sıklıkla kullanılmaktadır. Öğrenme yöntemleriyle karşılaştırıldığında rasgele orman sınıflandırıcısı, özellikle hızlandırma yöntemine göre, eğitim aşamasında çok daha hızlıdır. Yeterliliği ve doğruluğu ile çok kullanışlı bir sınıflandırıcıdır [49]. Hızlandırma yönteminin hesap yükü fazla olduğu için Torbalama yönteminden daha yavaştır. Fakat çoğu durumda bu yöntemden daha doğru sonuçlar verir [50]. Hızlandırma yönteminin, çok yavaş ve gürültüye karşı duyarlı olması, tekrarlı eğitimin olabilmesi gibi dezavantajlarına karşın rasgele orman, hesapsal olarak hızlandırma sınıflandırıcısından çok daha basittir, gürültüye karşı

duyarlı değildir. Karar Ağaçları, tüm değişkenler arasından en iyi dalı kullanarak her bir düğümü dallara ayırmak yerine, her bir düğümde rastgele olarak seçilen değişkenler arasından en iyisini kullanarak her bir düğümü dallara ayırır. Her bir veri seti orijinal veri setinden yer değiştirmeli olarak üretilir. Sonra rastgele özellik seçimi kullanılarak ağaçlar geliştirilir. Geliştirilen ağaçlar budanmaz. Bu strateji Rasgele Ormanı'nın doğruluğunu eşsiz yapar. Rasgele Orman aynı zamanda çok hızlıdır, aşırı uyuma karşı dayanıklıdır ve ne kadar istenirse o kadar ağaçla çalışılır [50].

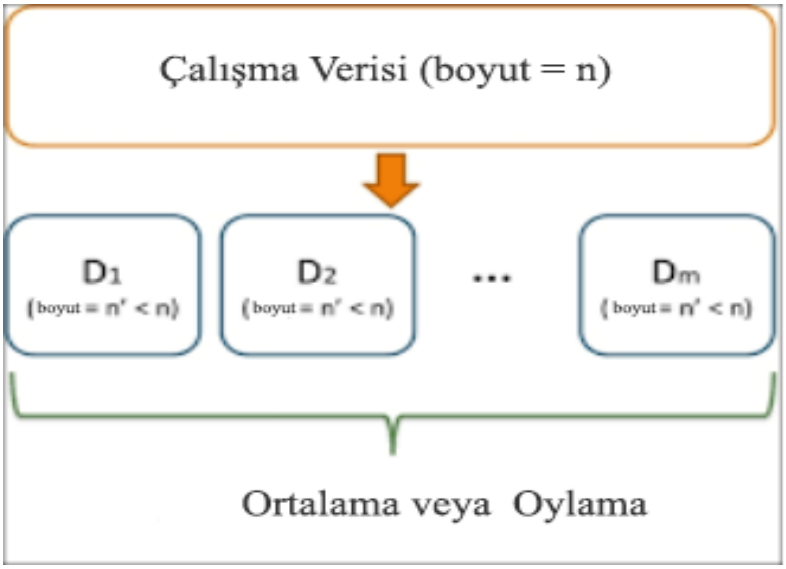
Rasgele Orman algoritmasını başlatmak için kullanıcı tarafından 2 parametre tanımlanmalıdır. Bu parametreler, en iyi bölünmeyi belirlemek için her bir düğümde kullanılan değişkenlerin sayısı (m) ve geliştirilecek ağaçların sayısı N'dir. Verilen eğitim verisinden budama olmadan ağaç geliştirilir. Her bir düğümde m değişkenleri tüm değişkenler arasından rastgele olarak seçilir ve bu değişkenler arasından en iyi dal belirlenir. Yeterli öngörü gücü ile yeterli miktarda düşük korelasyon sağlayan değişken sayısının seçimi son derece önemlidir [51]. Breiman'e göre toplam M adet değişken sayısının kare köküne eşit alınan m değişken sayısı genel olarak optimum sonuca en yakın sonucu verir [44]. Rasgele Orman ağaç üretmek için CART (Classifcation and Regression Tree) algoritmasını kullanır [50]. Her bir düğümde dallar CART algoritmasının kriterine (örn. GINI indeksi) göre oluşturulur. GINI indeksi sınıf homojenliğini ölçer ve (Denklem 2.1) ile ifade edilebilir.

$$\sum\sum_{j\neq i}(f(C_i,T)/|T|)(f(C_i,T)/|T|) \tag{2.1}$$

Burada T çalışma verisini C_i ise verilen dökümanın hangi sınıfa ait olduğunu $f(C_i,T)/|T|$ ifadesi de seçilen örneğin hangi C_i sınıfına ait olduğunun olasılığını ifade etmektedir.

Rasgele Orman sınıflandırıcı, zayıf sınıflandırma ve regressyon algoritmaların performansını güçlendirmek için birkaç teknik birleştirilerek uygulanabilir. Son zamanlarda en çok kullanılan teknikler torbalama (bagging), hızlandırma (boosting) ve rasgele altuzay (random subspace) örnekleme metodlarıdır. Tüm bu yöntemler çalışma verisini modifiye eder, sınıflandırıcı bu çalışma verisi üzerinde çalıştırılır ve

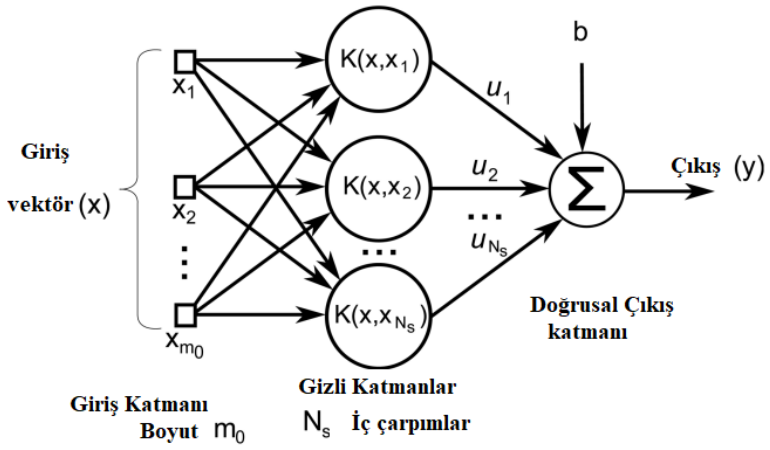
en son karar ağacı oluşturmak için basit veya ağırlıklandırılmış oylama usulune göre yapar (Şekil 2.3.).



Şekil 2.3. Rasgele Orman algoritması

2.3.4.3. Karar destek makineleri (Support Vector Machines)

Karar Destek Makineleri (SVM) istatistiksel öğrenme teorisine dayalı bir kontrollü sınıflandırma algoritmasıdır. SVM'nin sahip olduğu matematiksel algoritmalar başlangıçta iki sınıflı doğrusal verilerin sınıflandırılması problemi için tasarlanmış, daha sonra çok sınıflı ve doğrusal olmayan verilerin sınıflandırılması için genelleştirilmiştir. SVM'nin çalışma prensibi iki sınıfı birbirinden ayırabilen en uygun karar fonksiyonun tahmin edilmesi, başka bir ifadeyle iki sınıfı birbirinden en uygun şekilde ayırabilen hiper-düzlemin tanımlanması esasına dayanmaktadır [52] [53].



Şekil 2.4. SVM çalışma yapısı

Şekil 2.4.'te $K(x, x_i)$ çekirdek fonskiyonlarını ve u_i ise Langrange çarpanlarını göstermektedir. Çekirdek fonsksiyonları yardımıyla giridilerin iç çarpımları hesaplanlandıktan sonra Lagrange çarpanları çarpılarak toplanmaktadır.

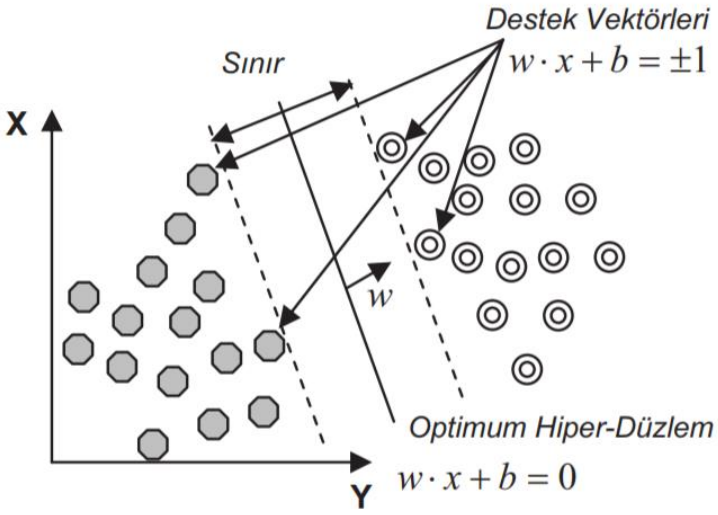
2.3.4.3.1. Doğrusal ayrılabilen veriler için SVM

SVM'de amaç, sınıfların birbirinden ayırt edilecek optimal bir ayırma hiper düzleminin oluşturulmasıdır. Bu aynı zamanda farklı sınıflara ait karar destek vektörleri arasındaki uzaklıkların en büyük yapılmasıdır. Doğrusal olarak iki sınıflı bir sınıflandırma probleminde SVM'in eğitimi için k sayıda örnekten oluşan bir eğitim verisinin $\{x_i, y_i\}$, $i = 1,...,k$ olmak üzere, optimum hiper-düzleme ait eşitsizlikler aşağıdaki gibi olmaktadır.

$w.x_i + b \geq 1$ her y=+1 için

$w.x_i + b \leq 1$ her y=-1 için

Burada $x \in R^N$ olup N-boyutlu bir uzayı, $y \in \{-1,+1\}$ ise sınıf etiketlerini, w ağırlık vektörlerini ve b eğilim değerini göstermektedir [54]. Optimum hiper-düzlemin belirlenebilmesi için bu düzlemin berlirlenmesi gerekir. (Şekil 2.5.) Bu hiper-düzlemleri kısace $w.x_i + b = \pm 1$ şeklinde gösterebiliriz.



Şekil 2.5. Doğrusal olarak ayrılabilen veri kümesi için hiper-düzlem

SVM'de optimum hiper-düzlemi bulmak için $\| w \|$ ifadesinin maksimum olması gerekmekte olup bu işlem de aşağıdaki sınırlı optimizasyon probleminin çözümünü gerektirir.

$$\min[\frac{1}{2} \| w \|^2] \tag{2.2}$$

Buna bağlı sınırlamalar,

$$y_i(w_i x_i + b) - 1 \geq 0 \text{ ve } y \in \{-1, +1\} \tag{2.3}$$

(Denklem 2.3) şeklinde ifade edilir [52]. Bu oprimizasyonun çözümü Lagrange deklemleri aracılığıyla aşağıdaki ifadeye dönüştürülebilir.

$$L(w, b, \alpha) = \tfrac{1}{2} \| w \| - \sum_{i=1}^{k} \alpha_i y_i (w.x_i + b) + \sum_{i=1}^{k} \alpha_i \tag{2.4}$$

(Denklem 2.4) eşitliğinde $\alpha_i \geq 0$ olmak üzere, her bir α_i Lagrange çarpanı olarak ifade edilir. L , w ağırlık vektörü ve b sabitini en küçükleyen ve negatif olmayan dual değişken α_i 'yi en büyükleyen (maksimize) bir fonksiyondur [55].

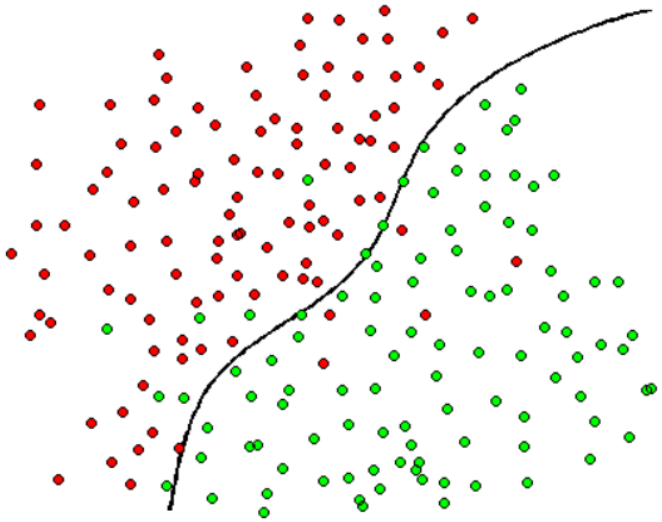
Son olarak doğrusal olarak iki sınıflı bir problem için karar fonksiyonu (Denklem 2.5) ile ifade edilebilir [53].

$$f(x) = sign(\sum_{i=1}^{k} \lambda_i y_i (x.x_i) + b) \tag{2.5}$$

2.3.4.3.2. Doğrusal ayrılamayan veriler için SVM

Birçok gerçek yaşam durumlarında veri setinin doğrusal bir fonksiyonla tam veya belirli bir hata ile ayrılamamaktadır. Doğal olarak sınıflama, ayırma eğrisinin tahmin edilmesiyle mümkün olabilmektedir. Bu işlem oldukça zor olabilmektedir.

Veri setinin doğrusal ayrılamama durumunda geometrik gösterimi Şekil 2.6.'da gösterilmiştir. Bu durumda k-boyutlu girdi vektörü x'in K-boyutlu özellik vektörü Ψ'ye dönüştürülmesi gerekmektedir [52]. Bu işlemi gerçekleştirebilmek için doğrusal olmayan haritalama yaklaşımından yararlanılır [56].



Şekil 2.6. Doğrusal Ayrılamayan Veriler İçin Hiper Düzlem Belirleme

Doğrusal olmayan haritalama, orijinal girdi uzayı x'in bir Hilbert uzayı olan daha yüksek boyutlu F özellik uzayına dönüştürülerek doğrusal ayrımının gerçekleştirilmesi için kullanılan bir yaklaşımdır [57]. "Hilbert uzayı" pozitif skaler çarpıma sahip ve

öğeleri fonksiyonlardan oluşan tam iç çarpım uzayları olarak ifade edilmektedir [58] (Şekil 2.6.).

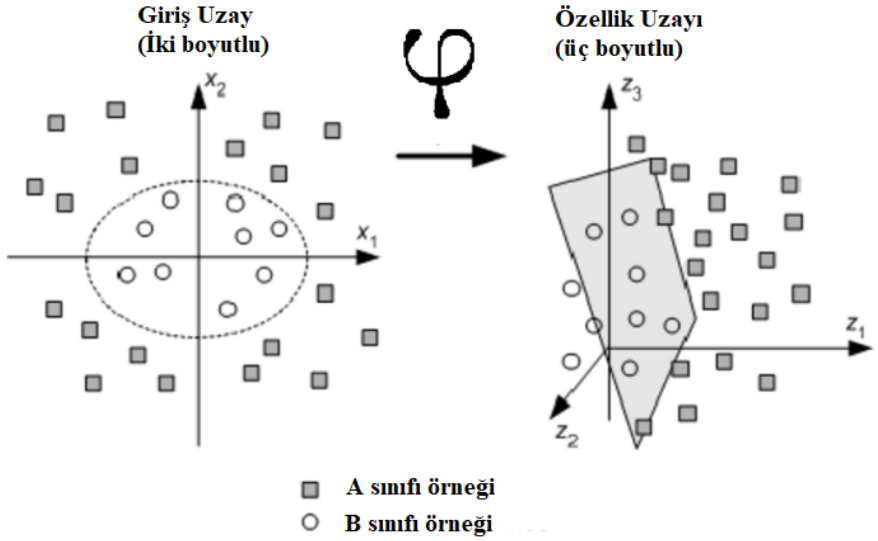
Doğrusal ayrılamayan veriler için elde edilen Ψ fonksiyonunun sınırının maksimum hale getirilmesi arasındaki denge pozitif değerler alan ve C ile gösterilen düzenleme parametresi tanımlamasıyla elde edilebilir [52]. Düzenleme parametresi ve yapay değişken kullanarak optimizasyon ifadesi (Denklem 2.6) gibi olur.

$$\min[\frac{1}{2}\| w\|^2 + C.\sum_{i=1}^{r}\psi_i] \tag{2.6}$$

Sınırlamalar ise (Denklem 2.7) ve (Denklem 2.8) gibidir.

$$y_i(w.\theta(x_i)+b)-1\geq 1-\psi_i \tag{2.7}$$

$$\psi_i \geq 0 \text{ ve } i=1,...,N \tag{2.8}$$



Şekil 2.7. Doğrusal olmayan veri setinin doğrusal olmayan haritalama yaklaşımı

Şekil 2.7.'de görüleceği üzere girdi uzayında doğrusal olarak ayrılamayan veri seti, özellik uzayında yüksek boyutlu bir uzayda görüntülenerek verilerin ayrımı yapılabilmekte ve sınıflar arasındaki hiper-düzlem belirlenebilmektedir.

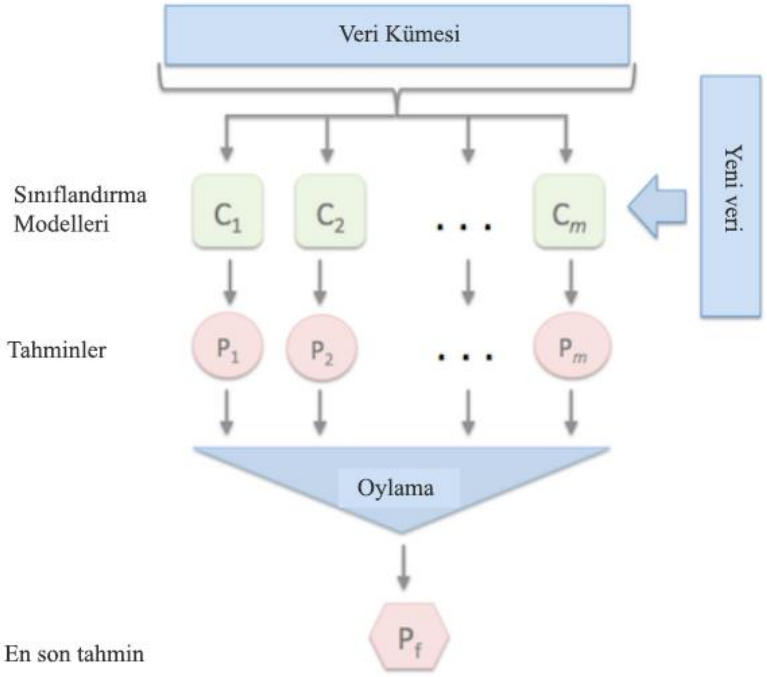
Karar destek makineleri (SVM) $K(x_i, x_j) = \psi(x).\psi(x_j)$ şeklinde bir kernek fonksiyonu yardımıyla doğrusal olmayan dönüşümler yapılabilmekte ve yüksek boyutta doğrusal olarak ayrım yapılabilmektedir. Bu şekilde kernek fonksiyonu kullanarak iki sınıflı bir problemin çözümü için karar kuralı (Denklem 2.9) ifadesi ile yapılabilmektedir [53].

$$f(x) = sign(\sum_i \alpha_i y_i \psi(x).\psi(x_i) + b) \tag{2.9}$$

Karar destek makineleri ile gerçekleştirilecek bir sınıflama için kullanılacak kernek fonksiyonu ve bu fonksiyona ait optimum parametrelerin belirlenmesi gerekir. Karar destek makineleri için birçok kernek fonksiyonu kullanılmaktadır. Bu kernel fonksiyonlarından en çok kullanılanlar ise radyal, polinom ve Pearson fonksiyonudur. Her kernel fonksiyonunun belli parametreleri olmakla beraber bazı parametrelerin manuel olarak atanması gerekmektedir. Kernek fonksiyonlarından radyal ve polinomiyal fonksiyonları algoritma çözümünde başarılı olmalarına rağmen polinomiyal çekirdek fonksiyonu ise hesaplama maliyeti açısından iyi olmadığı söylenebilir. Çünkü probleme ait çok boyutlu polinomu bulabilmek için çok fazla sayıda hesaplama yapması gerekmektedir. Ayrıca düzenleme parametresi (C) de uygulayıcı tarafından belirlenmesi gerekmektedir. Bu değerin seçimi performansı ciddi şekilde etkileyebilmektedir.

2.3.4.4. Oylamalı sınıflandırıcı (Voting classifier)

Oylamalı sınıflandırıcı meta-sınıflandırıcı olarak adlandırılmakta olup sınıflandırma işlemini yapmak için benzer sınıflandırıcıları veya farklı konsepteki sınıflandırıcıları birleştirerek çoklu oylamaya göre sınıflamaya karar vermeye yarar (Şekil 2.8.). Oylamalı sınıflandırıcı hard veya soft oylama olmak üzere iki farklı karar verme süreci bulunmaktadır. Hard oylamada, en çok oy alan sınıf, karar sınıf etiketi olarak seçilirken soft oylamada ise sınıf olasılıklarının ortalamaları baz alınarak karar sınıf etiketi olarak atanır.



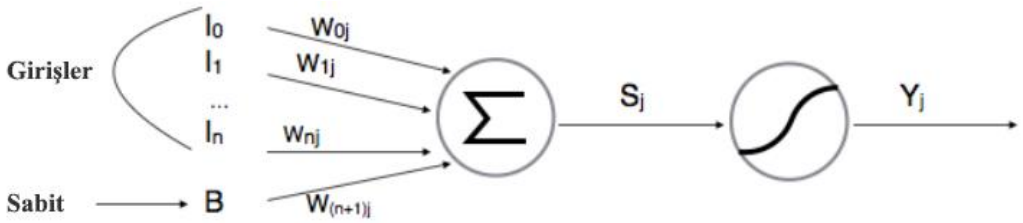
Şekil 2.8. Oylamalı sınıflandırıcı genel çalışma şekli

2.3.4.5. Yapay sinir ağları (YSA)

İnsan beynin nörofiziksel yapısından esinlenerek beynin matematiksel modeli çıkarılmaya çalışılmıştır. Beynin bütün davranışlarını tam olarak modelleyebilmek için fiziksel bileşenlerinin doğru olarak modellenmesi gerektiği düşüncesi ile çeşitli yapay hücre ve ağ modelleri geliştirilmiştir. Böylece Yapay Sinir Ağları denen yeni ve günümüz bilgisayarlarının algoritmik hesaplama yönteminden farklı bir bilim dalı ortaya çıkmıştır.

Genel anlamda YSA, beynin bir işlevi yerine getirme yöntemini modellemek için tasarlanan bir sistem olarak tanımlanabilir. YSA, yapay sinir hücrelerinin birbirleri ile çeşitli şekillerde bağlanmasından oluşur ve genellikle katmanlar halinde düzenlenir. Donanım olarak elektronik devrelerle veya bilgisayarlarda yazılım olarak gerçekleşebilir. Beynin bilgi işleme yöntemine uygun olarak YSA, bir öğrenme sürecinden sonra bilgiyi saklama ve genelleme yeteneğne sahip paralel dağılmış bir işlemcidir. Yapay bir sinir hücresi, biyolojik bir sinir hücresinin temel davranışlarından esinlenen matematiksel modeli ortaya koyan bir algoritma veya fiziksel araç olarak tanımlanabilir. Biyolojik sinir hücresinin tanımından hareket

ederek, yapay bir sinir hücresinin diğer sinir hücrelerinden aldığı sinyalleri bünyesinde topladığını ve toplam sinyal birikiminin belli bir eşiği aştığı anda, bu yapay sinir hücresinin kendi sinyalini bir başka sinir hücresine ilettiği söylenebilir (Şekil 2.9.).



Şekil 2.9. Yapay sinir hücre yapısı

YSA'nın görevi girdi değerine karşılık çıktı sinyali oluşturmak ve bu sinyali diğer hücrelere iletmektir. Her l_n ile y y_j arasındaki korelasyonu temsil eden w_{nj} ağırlıkları, yani yeni girdi örüntüsü ve çıktı siyaline göre tekrar ayarlanır. Bu ayarlama süreci öğrenme olarak adlandırılır. Öğrenmenin tamamlandığının belirlenebilmesi için, girdi örüntüntüleri w_{nj} ağırlıklarındaki değişim durağan olana dek sistemi beslemektedir. Durağanlık sağlandığında öğrenme işlemi tamamlanır.

2.3.4.5.1. Yapay sinir ağlarında öğrenme

Yapay sinir ağlarının en ayırt edici özelliklerinden birisi de öğrenme yeteneğine sahip olmasıdır. Öğrenme elde bulunan örnekler arasındaki yapının iyi bir davranış göstermesini sağlayabilecek olan bağlantı ağırlıklarının hesaplanması olarak tanımlanır. Yapay sinir ağları öğrenme esnasında elde ettiği bilgileri, sinir hücreleri arasındaki bağlantı ağırlıkları olarak saklar. Bu ağırlık değerleri yapay sinir ağlarının verileri başarılı bir şekilde işleyebilmesi için gerekli olan bilgileri içerir [59].

Yapay sinir ağlarının öğrenme prensibi çalışma verisinden yararlanmadır. Verilen çalışma verisinden yararlanarak girdi ve çıktı arasındaki ilişkiyi bulunur. Bu ilişkilerin bulunması için çok sayıda çalışma verisine ihtiyaç duyulmakta olup buradan çıkan ilişki kullanılarak yeni örneklere ait çıktılar belirlenmektedir. Yapay sinir ağları makine öğrenme yöntemlerinden gözetmenli, gözetmensiz veya takviyeli öğrenme yöntemlerinden biriyle öğrenme işlemini yapabilir.

2.3.4.5.2. Yapay sinir yapısı

Biyolojik sinir ağlarının sinir hücresi olduğu gibi yapay sinir ağlarının da yapay sinir hücreleri vardır. Bu elemanlar; girdiler, ağırlıklar, toplama ve aktivasyon fonksiyonudur.

Girdiler: Yapay sinir hücresine dışardan gelen bilgilerdir. Bunlar ağın öğrenmesini sağlar.
Ağırlıklar: Bir yapay hücreye gelen bilginin önemini ve hücre üzerindek etkisini gösterir. Ağırlıkların büyük veya küçük olması bilginin çok önemli veya önemsiz olduğunu göstermeyebilir.

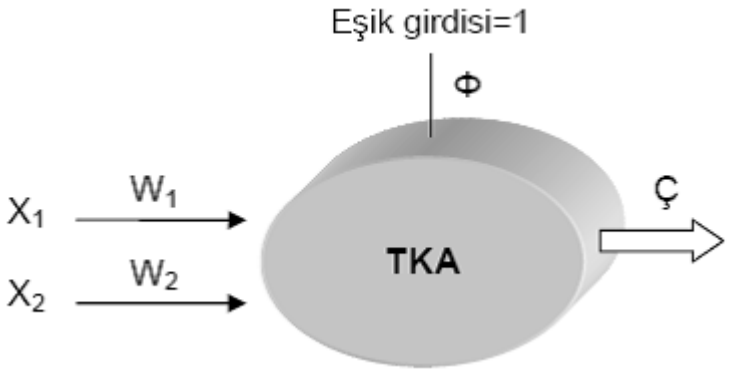
Toplama Fonksiyonu: Bu fonksiyon, bir hücreye gelen net girdiyi hesaplar. Bu işlemi yapabilmek için çeşitli fonksiyonlar kullanılmaktadır. En çok kullanılan toplama fonksiyonu ağırlıklı toplama fonksiyonu olup her girdi kendi ağırlığı ile çarpılarak toplanmaktadır. Bu şekilde ağa gelen net girdi bulunmuş olur.

- Yapay sinir ağlarının mimarisi

Yapay sinir ağlarının birçok mimarisi bulunmaktadır. Temel yapı olarak giriş değerlerine göre eğer arada gizli katmanlar da bulunuyorsa, buradaki gizli katmanlarda da öğrenme süreçlerine göre bir çıktı değeri üretmektedir. Ayrıca gizli katmanlarda ileri beslemeli (feedforward) olarak çalışmaktadır.

- Tek katmanlı algılayıcılar

Tek katmanlı yapay sinir ağları sadece girdi ve çıktı (Ç) katmanlarından oluşur. Çıktı üniteleri bütün girdi ünitelerine (X) bağlanmaktadır ve her bağlantının bir ağırlığı (W) vardır (Şekil 2.10).

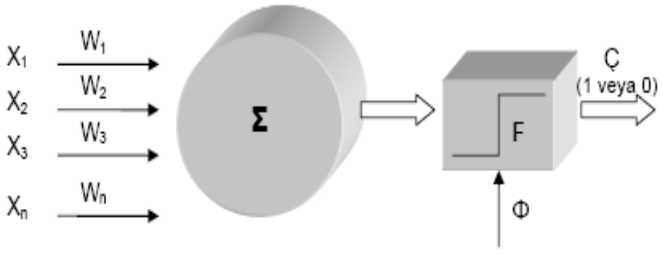


Şekil 2.10. Tek katmanlı algılayıcı

Tek katmanlı algılayıcılarda çıktı fonksiyonu doğrusal bir fonksiyondur ve 1 veya -1 değerlerini almaktadır. Eğer çıktı 1 ise birinci sınıfa -1 ise ikinci sınıfa kabul edilmektedir [60].

– Basit algılayıcı (Perceptron)

Basit algılayıcılar modeli 1950'li yılların sonlarında Rosenblatt tarafından ortaya atılmıştır. Esas olarak, basit algılayıcılar sınıflandırma amacıyla kullanılan, McCulloch ve Pitts tarafından geliştirilmiş daha karmaşık bir modeldir. Bir araya gelmiş sinir hücrelerinin miktarına bağlı olarak, basit algılayıcılar ile değişik sayıda sınıflandırma problemi çözebilir. Doğru bir sınıflandırma için sınıfların düzlemsel olarak ayrılması gerektiği gösterilmiştir [61]. Basit algılayıcılar bir sinir hücresinin birden fazla girdiyi alarak bir çıktı üretmesi prensibine dayanmaktadır. Ağın çıktısı bir veya sıfırdan oluşan mantıksal değerdir. Çıktının hesaplanmasında eşik değer fonksiyonu kullanılır. Ağın yapısı Şekil 2.11.'de verilmektedir.



Şekil 2.11. Basit Algılayıcı ağ modeli

– Çok katmanlı algılayıcı (Multilayer Perceptron)

Perceptron, doğrusal olmayan çözümler üretemediği için hem mimari hem de eğitim algoritması açısından iyileştirilmiş Çok Katmanlı Algılayıcı (MLP) ağı önerilmiştir. Mimari açıdan doğrusal olmayan aktivasyon fonksiyonuna sahip birçok nöronun birbirine hiyerarşik olarak bağlandığı bir yapıya sahip olan MLP, Algılayıcı ve Adaline yöntemlerinin avantajları yanı sıra geri-yayılım adındaki öğrenme sistemini kullanmaktadır ve genel olarak yapay sinir ağları ileri beslemeli ve geri beslemeli ağlar olarak ikiye ayrılmaktadır.

– Çok katmanlı algılayıcı algortiması

Çok Katmanlı Algılayıcı (MLP) öğrenme aşamasında Stokastik Gradient Decent (SGD), Adam veya L-BFGS yöntemlerini kullanmaktadır. SGD parametreleri bir hata fonksiyonu aracılığıyla güncellerken birkaç parametrenin manuel olarak ayarlanması gerekebilir [62].

$$w \leftarrow w - \eta(\alpha \frac{dR(w)}{dw} + \frac{dLoss}{dw}) \tag{2.10}$$

Denklem 2.10'de verilen ifadede η öğrenme oranı olup (learning rate) algoritmanın adım uzunluğunu arama uzayındaki parametreyi kontrol etmektedir. Buradaki Loss fonksiyonu ise ağdaki hata oranını hesaplayan fonksiyondur. Skotastik optimizasyon fonksiyonlarından olan, Adam da SGD gibi çalışmakla beraber, çok küçük bir zaman diliminde parametreleri otomatik olarak güncelleyebilir [62].

L-BFGS, Hessian matrisine yaklaşan, bir fonksiyonun ikinci dereceden kısmı türevini temsil eden bir çözücüdür. Ayrıca verilen Hessian matrisinin tersini de hesaplayarak parametreleri günceller. Diyelim ki n adet çalışma verisi ve m adet özellik verisi elde edilsin. Gizli katmanlar k olarak, her katmanda h sayıda nöron ve o adet de çıkış nöronunun olduğunu varsayalım. Bu durumda geri bildirimin algoritma karmaşıklığı $O(n.m.h^k.o.i)$ ile hesaplanır. Buradaki i algoritmanın çalışma adedini göstermektedir.

Burada eğer büyük algoritma karmaşıklığı değeri çıkıyorsa geri bildirim (backpropagation) değerinin düşürülmesi veya gizli katmanların azaltılması söz konusu olabilir [62].

Elimizde çalışma verilerine ait örnekler $(x_1, y_1),(x_2, y_2),...,(x_n, y_n)$, $x_i \in R^n$ ve $y \in \{0,1\}$ olmak üzere öğrenme fonksiyonu (Denklem 2.11) ifadesi ile tanımlanabilir.

$$f(x) = W_2 g(W_1^T x + b_1) + b_2 \quad W_1 \in R^m \text{ ve } W_2, b_1, b_2 \in R \tag{2.11}$$

(Denklem 2.11) deki W_1 ve W_2 girişlere ait sırasıyla giriş katmanı ve gizli katman ağırlıklarını göstermektedir. Aynı şekilde sırasıyla b_1 ve b_2 gizli katmana ve çıkış katmanına eklenen bias değerleridir. Buradak $g(.): R \to R$ ile tanımlı fonksiyonu aktivasyon fonksiyonu olup defakto olarak hiperbolik tanjant olarak (Denklem 2.12) ile tanımlanmaktadır.

$$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{2.12}$$

Eğer ikili sınıflandırma işlemi yapılacaksa f fonksiyonuna (2.13) de tanımlandığı gibi lojistik fonksiyonuna çevrilerek 0 ve 1 arasındaki değerleri alması sağlanmaktadır.

$$g(z) = \frac{1}{(1 + e^{-z})} \tag{2.13}$$

Eğer burada ikili sınıflamadan farklı olarak ikiden fazla sınıflı bir yapı kurulacaksa sınıf sayısına eşit bir çıktı üretebilmek için softmax denilen bir fonksiyon olarak tanımlanacaktır. Bu fonksiyon (Denklem 2.14) ifadesindeki gibi hesaplanmaktadır.

$$\text{softmax}(z)_i = \frac{\exp(z_i)}{\sum_{l=1}^{k} \exp(z_l)} \tag{2.14}$$

(Denklem 2.14) ifadesinde z_i, i. sınıfa ait softmax değeri ve k ise toplam sınıf sayısıdır. Sonuç olarak elde edilen vektör x örneğine ait, hangi sınıfa ait olabileceğini gösteren olasılık değerleridir. Çıkış ise en yüksek olasılığa ait sınıf olacaktır [62].

Ayrıca regressyon analizinde aktivasyon fonksiyonu birim fonksiyon olarak seçilecektir. MLP birçok farklı hata fonksiyonu problemin türüne göre kullanmaktadır. Hata fonksiyonu sınıflandırma işlemi için Cross-Entropy fonksiyonu olup ikili sınıflandırma için (Denklem 2.21) şeklinde tanımlanır.

$$Loss(\hat{y}, y, W) = -y \ln \hat{y} - (1-y)\ln(1-\hat{y}) + \alpha \| W \|_2^2 \tag{2.15}$$

(Denklem 2.15) ifadesideki, $\alpha \| W \|_2^2$ bir l2-regülasyon terimi olup komplex bir modeli dengeler. Ayrıca $a > 0$ bir hiperparametre olup hatanın boyutunun uzunluğunu kontrol altına alır.

BÖLÜM 3. ZARARLI YAZILIM KAYNAKLI VERİ ATAKLARINA KARŞI DÖKÜMAN SINIFLANDIRMA ALGORİTMASININ GELİŞTİRİLMESİ

3.1. Giriş

Zararlı yazılımlar, günümüzde sistemde kalıp ara ara veya bir komut geldiğinde çalışabilecek şekilde programlanmaktadır. Bu tür yazılımların amacı sistemde uzun süre kalıp kişinin mahremiyetini ihlal etmektir. Kurumsal ve uluslararası bağlamda ise bu tür yazılımların esas amacı gizli olan bilgiyi sızdırmaktır. Veri sızıntısını önlemeye yönelik yapılan çalışmalar zararlı yazılım ataklarına karşı son derece başarızdırlar. Bunun sebepleri arasında, ürünlerin bu tür saldırıları ihmal etmesinden veya güçlü bir algoritmanın geliştirilmemiş olmasından kaynaklanmakta olduğu ifade edilmektedir. Veri sızıntısı alanındaki bu açığı kapatmak amacıyla geliştirdiğimiz çalışmamızda, mevcut veri sızıntısı çözümleri incelenerek, zararlı yazılım ataklarına karşı daha dayanıklı bir algoritma önerilmektedir.

Doküman hakkında kelime n-gramlarıyla elde edemediğimiz durumlarda karakter-gramlar efektif olabilmektedir. Özellikle n-gramlara ayırma algoritmalarını atlatmaya yönelik yapılan saldırılarda karakter gramların kullanılması gerekmektedir. Ayrıca dökümana ait yeterli bilgi alınamayan kısa metinlerde skip-gram ve karakter gramların beraber kullanılması dökümana ait daha fazla bilgi elde edilmesi sağlanmaktadır.

LSA yöntemi, kelimeler arasındaki anlamsal bağlantıyı bulmak için kullanılan bir yöntemdir. LSA, bir veri seti (corpus) kullanarak veriyi öğrenmektedir. Veriyi öğrenme aşamasında Kırpılmış SVD (Truncated SVD) kullanarak boyut azaltımı yapılmakta daha sonra uzaklık ölçümüne göre kelimeler arasındaki bağ bulunmaktadır [3]. LSA yöntemi özellikle karartılımış verinin üzerinde kullanılması ile kelimelerin anlamı belirginleştirilmektedir. Eş anlamlı ve çok anlamlı kelime saldırılarına karşı LSA özellik çıkarım yöntemi önem arzetmektedir.

Özellik çıkarma aşamasından sonra doküman SVM, rasgele orman ve çok katmanlı algılayıcı sınıflandırma algoritmalarından oluşan oylamalı sınıflandırıcı tarafından eğitilmektedir. Eğitilen sınıflandırıcı aracılığıyla test dökümanlarının sınıflandırılması sağlanmaktadır. Ayrıca önerilen sistemde yazım düzeltiminin yapılması öngörülmüştür. Çünkü yapısal saldırılarda sınıflandırıcının sağlıklı karar verebilmesi için modifiye edilmiş kelimelerin düzeltimi yapılması gerekmektedir.

Önerilen algoritmanın eğitim aşamasında öncelikle metin üzerinde önişlemler yapılmış, daha sonrasında özellik çıkarımı sağlanmış, çıkarılan özellikler arasından özellik seçimi yapılarak sınıflandırma için daha az gürültülü bir özellik kümesi çıkarılmıştır. Seçilen özellikler sınıflandırıcılardan geçirilerek "Sınıflandırma Modeli" oluşmaktadır (Şekil 3.1.).

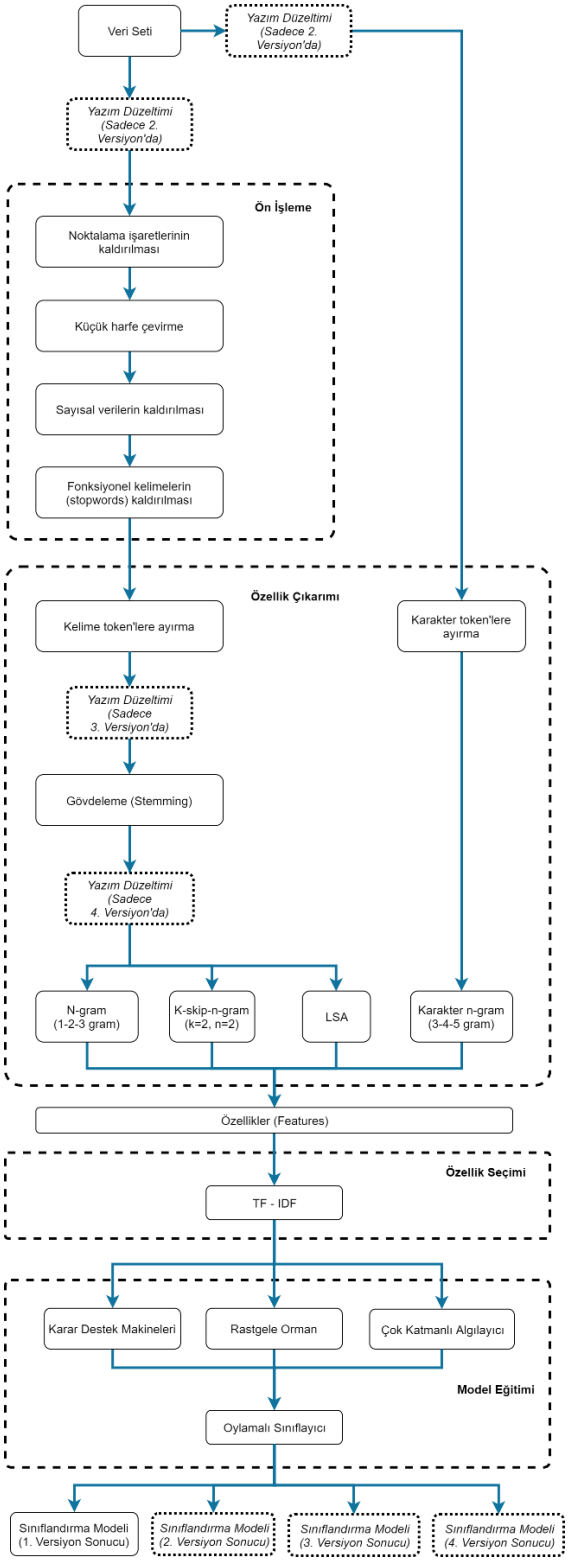
3.2. DLP Veri Seti Oluşturma

Michael Hart ve ark. [37] veri sızıntısı önleme (Data Loss Prevention-DLP) için metin sınıflandırma üzerine yaptıkları çalışmada bir DLP Corpora oluşturulmuştur. Çalışmamızın güvenirliğini test etmek amacıyla aynı veri kümeleri oluşturuldu.

Proje içerisinde üç adet doküman sınıf tanımlaması yapılmıştır:

1. Doküman Sınıf1 - Kurumsal Gizli – G (Private Enterprise): Kurumsal gizli (private) kategorisinde bulunan dökümanlar için kuruma ait politika dokümanları, yasal anlaşmalar, finansal kayıtlar, özel müşteri verileri ve kaynak kodlar, vb. bilgileri içerir. Bu çalışmada kurumlara ait Wikileaks sitesine sızdırılmış dökümanlar Kurumsal Gizli doküman olarak kabul edilmiştir.
2. Doküman Sınıf2 - Kurumsal Genel – KG (Public Enterprise): Kurumsal genel kategorisinde bulunan dökümanlar için kuruma ait web sayfaları, müşteriler ve diğer harici birimlere ait elektronik mailer, genel blog sayfaları, vb. bilgileri içerir. Bu çalışmada kuruma ait internet sitesindeki doküman ve sayfalar kullanıldı.
3. Doküman Sınıf 3 - Kurumsal Olmayan - KO (Non-Enterprise): Yukarıdaki sınıfta tanımı yapılmayan veri kümelerini içerir. Kurumsal olmayan veri kümeleri için

DBpedia 09/09/2015 tarihli 2. versiyonu kullanılmıştır [5]. Dbpedia tüm veri setleri için Kurumsal Olmayan olarak etiketlendirilmiştir (Tablo 3.1.).



Şekil 3.1. Metin sınıflandırma aşamaları

Ayrıca döküman bazlı bir çalışma yapabilmek ve yapılan çalışmada hassas verileri daha iyi tanımlayabilmek için elde ettiğimiz dökümanlardan birleştirilerek 4. bir veri seti daha oluşturmuştur. Bu veri seti tümleşik veri seti olarak tanımlanmıştır.

3.2.1. Dyncorp

Dyncorp, ABD'de merkezli uluslararası faaliyet gösteren bir güvenlik firmasıdır. Bu firmaya ait 23 adet döküman wikileaks internet sitesinde gizli olarak yayınlanmıştır. Bu dökümanlar pdf formatında olup sadece 5 tanesi metin içerikli olup diğer kalan dökümanlar ise askeri çizimlerin olduğu resim tabanlı dökümanlardır. Bu çalışmada 2 adet pdf dökümanı okunabilmiş ve hassas veri olarak kullanılmıştır. Ayrıca internet sitesinden gizli olmayan kurumsal (Enterprise) 198 adet pdf dosyası indirilerek kurumsal ama gizli olmayan dosya olarak etiketlenmiştir.

3.2.2. Mormon

Mormon, dini bir tarikat olup farklı ritüellere sahiptir. Mormon grubuna ait wikileaks'ten "Church Handbook of Instructions Book 1" adlı kitap ve bunlara ek olarak küçük bir kitap eklenerek hassas veriler oluşturulmuştur. Buradaki kitabın sınıflama aşamasında yeterli örnek oluşturması için 1000 karekterlik parçalara ayrılmıştır. Daha sonra internet sitesinde 3 adet pdf indirilerek bunlar da aynı şekilde 1000 karakterlik parçalara ayrılmıştır. Bu işlem sonunda 593 Kurumsal Gizli (Enterprise Private), 2541 adet Kurumsal Olmayan (None Enterprise) metin oluşturulmuştur.

3.2.3. Transcendental Meditation (TM)

Transcendental Meditation (TM), meditasyon ve teknikleriyle ilgilenen bir kuruluş olup bu tekniklerin insan hayatına olumlu etkilerini dile getirmektedir. Bu kuruma ait wikileaks internet sitesine sızdırılan 85 adet döküman Kurumsal Gizli (Enterprise Private) olarak etiketlendirilerek, web sitesinde ve blog sitesinde 120 adet internet sayfası da Kurumsal Olmayan (None Enterprise) olarak alınarak etiketlenmiştir.

3.2.4. DBpedia

DBpedia, Wikipedia projesindeki bilgilerden yapısal içerikler çıkartma amacıyla oluşturulmuştur. Bu yapısal içerikler internet ortamında ulaşılabilir hale getirilmektedir. DBpedia semantik olarak arada ilişki bulunan wikipedia kaynaklarına ve ilişkili olduğu diğer veritabanına ait linkle beraber sorgu yapma özelliği sağlar. 2007 yılında yayınlanan bu veritabanı, açık lisanslar (CC -BY-SA) altında diğer kullanıcıların kullanımına açılmıştır.

Wikipedia makaleleri çoğunlukla serbest metinlerden oluşmaktadır, fakat aynı zamanda yapısal bilgiler de gömülmüştür. Örneğin "infobox" tabloları, sınıflandırma bilgisi, resimler, harita bilgisi (koordinat) ve dış web sayfalarına bağlantılar sağlamaktadır. Bu şekilde yapısal bilgiler çıkarmak amacıyla sorgulanabilir bir forma sokulabilir [2]. DBpedia bu bilgileri kullanarak veritabanını yapılandırır. DBpeadia veritabanı bağlantısız (offline) versiyonu public olarak ulaşılabilir durumdadır [5]. DBpedia 09/09/2015 tarihli 2. versiyonunu kullandığımız projede Xiang Zhang ontolojik olarak sınıflandırmak amacıyla kullanmıştır. Bu projede birbirinden ayırt edilebilen 14 adet etiketlenmiş veri bulunmakta olup, "Company, Educational Institution, Artist, Athlete, OfficeHolder, Mean Of Transportation, Building Natural Place, Village, Animal, Plant, Album, Film, Written Work" kategorilerinden oluşmaktadır. Bu veri setinde eğitim amaçlı 560,000 ve test amaçlı 70,000 döküman yer almaktadır. Çalışmamızda bu veri kümesinden 2000 adet döküman alınmıştır. Tablo 3.1.'de öğrenme ve test aşamasında kullanılan döküman sayıları verilmiştir.

Tablo 3.1. Veri Seti ve Etiket Tablosu

Etiket	TM	Mormon	DynCorp	Dbpedia	Tümleşik
Kurumsal Gizli (G)	85	593	2	-	680
Kurumsal Genel (KG)	120	2541	198	-	2859
Kurumsal Olmayan (KO)	-	-	-	2000	2000

Yaptığımız çalışmada tüm veri setleri için 4 farklı yöntem uygulanmıştır. Bu yöntemler, yazım düzeltiminin yapılmadığı Yöntem 1 (Tablo 3.2.), yazım düzeltiminin döküman ön işleme aşamasından önce yapıldığı Yöntem 2 (Tablo 3.3.), yazım

düzeltiminin token çıkarımından sonra yapıldığı Yöntem 3 (Tablo 3.4.) ve son olarak gövdeleme işlemi bittikten sonra Yöntem 4 (Tablo 3.5.) olarak tanımlanmıştır.

Tablo 3.2. Yöntem 1 aşamaları

Veri seti	Eğitim veri seti 1			
Yazım düzeltimi	YOK			
NLP	NLP			YOK
Tokenlere ayırma	Kelime token			Karakter token
Yazım denetim	YOK			
Gövdeleme	Gövdeleme			YOK
Yazım denetimi	YOK			
Özellikler	N-gram	K-skip n-gram	LSA	Karakter n-gram

Tablo 3.3. Yöntem 2 aşamaları

Veri seti	Eğitim veri seti 2			
Yazım düzeltimi	VAR			
NLP	NLP			YOK
Tokenlere ayırma	Kelime token			Karakter token
Yazım denetim	YOK			
Gövdeleme	Gövdeleme			YOK
Yazım denetimi	YOK			
Özellikler	N-gram	K-skip n-gram	LSA	Karakter n-gram

Tablo 3.4. Yöntem 3 aşamaları

Veri seti	Eğitim veri seti 2			
Yazım düzeltimi	YOK			
NLP	NLP			YOK
Tokenlere ayırma	Kelime token			Karakter token
Yazım denetim	YOK			
Gövdeleme	Gövdeleme			YOK
Yazım denetimi	VAR			YOK
Özellikler	N-gram	K-skip n-gram	LSA	Karakter n-gram

Tablo 3.5. Yöntem 4 aşamaları

Veri seti	Eğitim veri seti 1			
Yazım düzeltimi	YOK			
NLP	NLP			YOK
Tokenlere ayırma	Kelime token			Karakter token
Yazım denetim	YOK			YOK
Gövdeleme	Gövdeleme			YOK
Yazım denetimi	VAR			YOK
Özellikler	N-gram	K-skip n-gram	LSA	Karakter n-gram

3.3. Metin Çıkarımı

Metin çıkarımı, eldeki dökümana bakarak anlamlı veri elde etme işlemidir. Veri birçok formatta bulunabilir. Yukarıda verilen DLP veri kümeleri pdf, csv ve html gibi farklı formata sahiptiler. PDF dökümanları textaract kütüphanesi ile, csv dosyaları ise python yazılım dilinin file modülü ile çıkartılmıştır. Web sitelerindeki linklerden yola çıkarak indirilen html sayfaları da Beautiful Soup kütüphanesi aracılığıyla html etiketlerinden ayrılarak metinler çıkarılmıştır. Burada kullandığımız kodlarımız dyncorp veriseti üzerinde şu şekilde uygulanmıştır.

```
import textract
from os import listdir
from os.path import isfile,join
import pickle
import csv
pathPrivate="data/dyncorp/private/"
pathPublic="data/dyncorp/public/"
pathCode="data/dyncorp/code/"
pathToPmormon="data/mormon/private/"
pathToPumormon="data/mormon/public/"
pathToNE="data/Dyncorp/notenterprise/train.csv"
from unidecode import unidecode
privateFiles=[f for f in listdir(pathPrivate) if isfile(join(pathPrivate,f))]
publicFiles=[f for f in listdir(pathPublic) if isfile(join(pathPublic,f))]
def loadDyncorpCorpus():
    X_train=[]
    y_train=[]
    X_vikipedia=[]
    y_vikipedia=[]
    for private in privateFiles:
        try:
            text=textract.process(pathPrivate+private)
            text=unidecode(text)
            X_train.append(text)
             y_train.append(0)
        except:pass
    for public in publicFiles:
        try:
            text=textract.process(pathPublic+public,encoding="utf-8")
            text=unidecode(text)
             X_train.append(text)
             y_train.append(1)
        except:pass
```

```
with open("data/dyncorp/notenterprise/train.csv") as csvFile:
    reader = csv.reader(csvFile, delimiter=",")
    for row in reader:
        if row[0] == "1":
            try:
                if len(y_vikipedia)< 2000:
                    txt=row[2]
                    txt=unidecode(txt)
                    y_vikipedia.append(2)
                    X_train.append(txt)
                    y_train.append(2)
            except:pass
return X_train,y_train
```

Elde edilen dökümanların daha sonra kullanabilmesi amacıyla "pickle" kütüphanesi kullanılarak dökümana ait metinler çıkarılıp pickle formatında saklanmıştır. Bu kısma ait kodlarımız aşağıdadır.

```
def saveDyncorpToPickle():
    x,y=loadDyncorpCorpus()
    filenameX="data/dyncorp/modelX.pkl"
    filenameY="data/dyncorp/modelY.pkl"
    pickle.dump(x,open(filenameX,"wb"))
    pickle.dump(y,open(filenameY,"wb"))
```

Benzer şekilde diğer veri setlerinden dökümana ait metinler çıkarılıp pickle formatında saklanmıştır.

3.4. Metin Normalizasyonu ve Önişleme adımları

Doğal dil işlemede, dökümana ait kelimeler kullanılmaktadır. Bu kelimelerin daha sonra kanonik bir forma getirilmesi sağlanılmaktadır. Bu kanonik forma metin normalizasyonu denilmektedir. Metin normalizasyon işleminde dökümana ait yapım eki almış kelimelerden bu ekler atılarak kelime sayısının azaltılması ve bu şekilde özellik matrisinin kestirimi sağlanması amaçlanmaktadır.

Ayrıca dökümandaki kelimelerin küçük harfe veya büyük harfe dönüştürülmesi, noktalama işaretlerinin atılması, sayısal verilerin kelimelere dönüştürülmesi, kısaltmaların genişletilmesi ve metnin kurallı hale getirilmesi gibi işlemler de metin

normalizasyonu içinde yer alabilir [67]. Bu yüzden metin normalizasyonu, sınıflandırma algoritmasının başarımını artıran önemli bir özelliktir. Model üzerinde doğal dil işleme yapıldığında elde edilen vektörler birer özellik vektörü olarak yorumlanır. Vektörler daha sonra belirtilen model çerçevesinde sayısal forma dönüştülmesi gerekmektedir. Oluşturulan bu sayısal form makine öğrenme algoritmalarının çalışabileceği bir yapıya sahip olmalıdır. Genel olarak algoritmalara uygun veri üretmek için sparse matrisleri oluşturulur. Bu matrisler aracılığıyla sınıflandırıcının hızlı işlem yaparak hızlı karar vermesi sağlanmaktadır.

Doğal dil işlemede (NLP) önişleme (preprocessing), dökümanlardan verilerin çıkarılması ve sınıflandırma için hazırlanma aşamalarını kapsar. Bu işlemler yapılırken dökümanı ifade edecek şekilde özelliklerin çıkarılması gerekir. Bu öğrenilen veriyi kullanarak yeni verilerin sınıflanmasının yapılması da gerekmektedir. Sınıflandırmanın başarısı önişleme aşamalarında elde edilen bilgilere bağlıdır. Doğal dil işlemede, önişleme aşamasında veri yapısına bağlı olarak kayıp verilerin (missing values) yerine yeni verilerin konulması veya sabit bir değerin girilmesi, metinler için ise fonksiyonel kelimlerin atılması ve dökümanıdaki kelimelerin kanonik forma getirilmesi (stemming) gibi işlemlerden oluşmaktadır. Ayrıca metinin harflerinin tamamının büyük veya küçük yapılması ve token oluşturma aşamaları da önişleme aşamalarında bulunabilir. Önişleme aşaması, metin çıkarımı yöntemlerini de içerisine alabilir (Şekil 3.2.).

Şekil 3.2. Ön işleme aşamaları

3.4.1. Noktalama işaretlerini kaldırma

Metin içerisindeki noktalama işaretleri metin hakkında fazla bilgi vermediği için kaldırılması uygun görülmektedir. Bu değerleri kaldırarak hem boyut azaltımı hem de karar vermede çok fazla bulunan bu karakterlerin sınıflandırmaya etkisi düşürülmektedir.

3.4.2. Metnin küçük harfe çevrilmesi

Metinlerde aynı kelimeler, büyük harfle başlamış veya tamamı büyük harfle yazılmış olabilir. Tüm metindeki harfleri küçük karaktere çevirerek buradaki farklılıkları azaltmış olmaktayız. Bu da boyut azaltımını az da olsa etkilemektedir.

3.4.3. Sayısal verilerin kaldırılması

Sayısal veriler döküman analizinde bir çok çalışmada alınmamaktadır. Bunun nedenlerinden birisi sayısal verilerin bir yargıyı bildirmemesidir. Kendi çalışmamızda da sayısal veriler metinden çıkarılmıştır.

3.4.4. Fonksiyonel kelimeler (Stop words)

Burada ele alınan fonksiyonel kelimeler cümlede herhangi bir yargı bildirmeyen veya genel olarak çokça kullanılan kelimelerdir. Bu kelimelerin atılması, bu kelimelerin zaten tekrar sayısının fazla olmasından dolayı, sınıflamaya olumsuz etki ettiği içindir.

Tawunrat Chalothorn ve Jeremy Ellman, Twitter üzerinde, kişi yönelimlerinin analizine yönelik yaptığı çalışmada python nltk kütüphanesini kullanmıştır. Bu çalışmada fonksiyonel kelimeleri ayrıştırmak için nltk kütüphanesindeki fonksiyonel kelimeler kullanılmıştır [10]. Python kütüphanesinde doğal dil işleme alanında kullanılan nltk ile ele alınan ve İngilizce için tanımlanan fonksiyonel kelimeler Tablo 3.9.'da gösterilmiştir. Hamid Poursepanj ve ark. [11] yaptıkları benzer bir çalışmada python nltk kütüphanesini ve fonksiyonel kelimleri kullanmıştır.

Tablo 3.9. Stop Words (Fonksiyonel Kelimler)

i,me,my,myself,we,our,ours,ourselves,you,you're,you've,you'll,you'd,your ,yours,yourself,yourselves,he,him,his,himself,she,she's,her,hers,herself,it,i t's,its,itself,they,them,their,theirs,themselves,what,which,who,whom,this,t hat,that'll,these,those,am,is,are,was,were,be,been,being,have,has,had,havi ng,do,does,did,doing,a,an,the,and,but,if,or,because,as,until,while,of,at,by, for,with,about,against,between,into,through,during,before,after,above,bel ow,to,from,up,down,in,out,on,off,over,under,again,further,then,once,here, there,when,where,why,how,all,any,both,each,few,more,most,other,some,s uch,no,nor,not,only,own,same,so,than,too,very,s,t,can,will,just,don,don't,s hould,should've,now,d,ll,m,o,re,ve,y,ain,aren,aren't,couldn,couldn't,didn,d idn't,doesn,doesn't,hadn,hadn't,hasn,hasn't,haven,haven't,isn,isn't,ma,migh tn,mightn't,mustn,mustn't,needn,needn't,shan,shan't,shouldn,shouldn't,was n,wasn't,weren,weren't,won,won't,wouldn,wouldn't

3.5. Özellik Çıkarımı Ön İşlemleri

Makine öğrenmesinin en önemli aşaması özellik çıkarım aşamasıdır. Makine öğrenmesinde kullanılan algoritmalar, yeteri kadar özellik bulduğunda sınıflandırma başarısı artmaktadır. Doğal dil işlemede (NLP) kullanılan özellik çıkarım yöntemleri genel olarak dökümandan metinlerin çıkarılması ve bu çıkarılan metinler üzerinde belli bir parçalama işlemi gerçeklertirmesine dayanır. Literatürde içerikten özellik çıkarım için birçok yöntem mevcuttur. Bu yöntemlerin birçoğunda dökümanın n-gram özellikleri kullanılmaktadır. Veri kaçaklarında, dökümanda yapılan kelime değişiklikleri, harf değişiklikleri ve dökümanı şifreleme gibi yöntemler sıkça kullanılmaktadır. Kelime ve harf değişikliklerinde dökümanın hangi sınıfa ait olduğu tespiti açısından n-gramların kullanılması daha sağlıklı görülmektedir. Ayrıca saldırı türleri açısından farklı özelliklerin eklenmesi ile algoritmanın daha sağlıklı karar verebileceği düşünülmektedir.

3.5.1. Token oluşturma (Tokenization)

Bir metinde, kullanılan kelimeler o dökümanı tanımlayan önemli bir özelliktir. Ayrıca dökümandaki kelimeler o dökümanın konusu, içeriği gibi bilgileri içeren ve dökümanları birbirinden ayırabilen bir ölçü olarak kullanılmaktadır. Döküman sınıflamada güçlü olduğu görülen n-gramlar, tokenler üzerinde çalışmaktadır. Bu nedenle tokenlerin çıkarımı önem arzetmektedir. Yapısal saldırılar kelimeler, paragraflar ve içeriğe yönelik olduğundan kelimeler burada ele alınması gereken

birincil hedeftir. Doğru sınıflama yapabilmek için kelime token özellikleri bize önemli bir ölçü sunmuştur. Ayrıca kelime tokenlerinin çıkarılamadığı veya yetersiz olduğu durumlar da vardır. Çünkü kelime token çıkarma algoritmaları noktalama işaretleri ve kelimeler arası boşluğa bakarak kelime tokenleri çıkarmaktadır. Zararlı yazılım tarafından bu açıklık kullanılarak metin token oluşturulamayacak duruma getirilebilir. Zararlı yazılımın kelime tokenlerine yaptığı bu saldırıda karakter tokenlerin çıkarılması metnin sınıflama aşamasındaki eşleşme oranını artırdığı görülmüştür.

Verilen bir metnin token'lerine ayrılmasında kullandığımız iki farklı yaklaşım bulunmaktadır. Bunlardan ilki metnin kelimelerine ayrılması kelime token, diğeri ise karakterlerine ayrılması yani karakter tokendir. Bu işlemlerden kelime tokenleri için nltk kütüphanesinin word_tokenize() metotunu kullandık.

```
def stem_tokens(tokens):
    d = [snowbal.stem(item.lower()) for item in tokens if item not in stops]
    return d
def nltktokenizer(text):
    t=text.lower()
    t=re.sub('\d+,','',t)
    return stem_tokens(word_tokenize(t))
```

Karaketer tokenler için ise Scikit-Learn kütüphanesindeki TfidfVectorizer sınıfı kullanılmıştır. Bu sınıf hem karakter hem de kelime gramları oluşturmak için kullanılan genel bir sınıftır. Buradaki analyzer parameteresini char olarak verildiğinde metni karakter tokenlerine ayırmaktadır.

```
character_model = TfidfVectorizer(analyzer="char", ngram_range=(3, 5), min_df=min_df)
```

3.5.2. Gövdeleme (Stemming)

Stemmer'lar veya gövde bulucu algoritmalar bir kelimenin kökünü veya kök ile beraber yapım ek almış halini bulmaya yarar. Doğal dil işlemede önemli bir özellik olarak görülen gövdeleme bilgi edinim sistemlerine çokça kullanılmaktadır.

Giridhar N S ve ark. Yaptıkları [21] çalışmada farklı gövdeleme yöntemlerini ele almışlar ve bu gövdeleme yöntemlerinin kullanılması ile döküman sınıflamada boyut azaltımı sağlanabileceğini belitmişlerdir. Çünkü gövdeleme işlemi ile aynı kelimenin farklı versiyonları tek bir çatı altına alarak tekilleştirilmiş olur. Ayrıca M. Ramya ve J.Alwin Pinakas [14] yaptıkları farklı özellik çıkarım yöntemlerinde gövdeleme yöntemi olarak Porter ve Lancaster algoritmalarını kullanmıştır. [21] ve [14] çalışmalarına göre en iyi gövdeleme yönteminin bulunması zor görünmektedir. Çünkü veri seti değiştiğinde sınıflandırıcının performansı değişebilmektedir.

Snowball farklı dillerdeki köklere ulaşma amacıyla geliştirilen büyük bir proje olup içerisinde farklı dillere ait gövdeleme (stemmer) algoritmalarına sahiptir.

Snowball projesinde İngilizce, Fransızca, İspanyolca, İtalyanca, Romanca, Almanca ve İskandinav ülke dillerine ait stemmer algoritmaları bulunmaktadır. Ayrıca diğer dillere de genişletmeye müsaittir [1].

Yukarıda ele alınan algoritmalar dışında çeşitli gövdeleme algoritmaları, istatistiksel algoritmalar ve karışık (mixed) algoritmalar da mevcuttur. Herbir algoritmaya ait avantaj ve dezavantajları bulunmaktadır.

Anjali G. Jivani [13] yaptığı karşılaştırmalı gövdeleme algoritmaları çalışmasında tüm gövdeleme yöntemlerini ele alarak bunların karşılaştırmalı zayıf ve güçlü yanlarını vermişlerdir. İlgili çalışmada tüm ihtiyaçlara cevap verebilecek bir gövdeleme yönteminin olmadığını belirtilmiş, farkı domainlerde farklı gövdeleme yöntemlerinin başarılı olabileceğini savunmuştur. Tezimizde Snowball aşağıdaki şekilde uygulanmıştır.

```
from nltk.stem import SnowballStemmer
snowbal=SnowballStemmer(language="english")
```

3.5.3. Yazım Düzeltimi (Spell Correction)

Dökümanlar her zaman doğru bir şekilde yazılmamış olabilir. Genel olarak insan kaynaklı yazım hatalarında fazla harf, eksik harf veya yanlış harf yazma gibi basit hatalar söz konusu olabilir. Yazılımsal ise maksatlı olarak çok farklı şekillerde yazım hataları ile karşılaşılabilmektedir. Tarique Mustafa [4] yaptığı çalışmada olabilecek içerik saldırılarını; yapısal, taşıma ve karartma olarak sınıflandırmış ve her bir saldırı için çözüm önerisinde bulunmuştur. Fakat bu önerilerin nasıl olabileceği ile ilgili açık bilgi vermemiştir. Doğal Dil İşleme (DDİ) yöntemleri kullanarak yapısal saldırılara karşı yapılabilecek önemli adımlardan birisi de yapısı bozulan dökümanı eski haline getirmektir. Yavuz canbay ve ark. [6] yaptıkları çalışmada, hassas verilere ait kelimelerden en çok tekrar edilenleri çıkararak, eldeki dökümanları LSI ile içeriklerine ayırmış ve daha sonra da benzerlik ölçüsüne göre hassas olduğu düşünülen dökümana sırasıyla Boyer Moore ve Smith Waterman algoritmasını uyguyanarak kelime ekleme, silme ve değiştirme durumlarında gelen dökümanın hassas olup olmadığını tespit etmeye çalışmıştır. Bu çalışmada önerilen yöntem, dökümandaki hassas veri dışında dökümanın tamamına yapılacak bir modifikasyon saldırısı durumunda algoritma savunmasız kalacaktır. Bu durumlarda yazım hatalarını düzelterek metni eski haline getirme daha iyi sonuç verebilmektedir. Casey whitelaw ve ark. [7] yaptıkları çalışmada web kullanarak n-gram bazlı bir hata düzeltme algoritması öne sürmüştür. Bu modelde veriler n-gram'larına ayrıldıktan sonra Levenshtein-Damerau uzaklık ölçüsü baz alınarak yazım düzeltme işlemi yapılmıştır. Fakat buradaki sistemde Yousef Bassil'in [8] yaptığı çalışmada belirtildiği gibi, her kelimeden sonra hangi kelimenin geleceğini hesaplama işlemi algoritma karmaşıklığını artıracaktır. Bu yüzden n-gramlı bir yöntemin tercih edilmesi algoritma karmaşıklığını artıracağı gibi performansın azalmasına da sebep olacaktır. Algoritma karmaşıklığının daha az olduğu Levenshtein-Demerau yönteminin daha hızlı bir algoritma ile kullanılması daha uygun olacaktır [9].

Çalışmamızda yazım hatasını düzeltme amacıyla Symspell Compound kütüphanesi kullanılmıştır. Burada sözlük olarak ingilizce dilinde yazılmış 28765 kelime ve

kullanım sıklığı verilen Symspell Compound kütüphanesinin defakto sözlüğü kullanılmıştır.

```
from symspellpy.symspellpy import SymSpell,Verbosity
max_edit_distance_lookup = 3
max_edit_distance_dictionary = 3
prefix_length = 7
text=”thebrownfox”
corector = SymSpell(max_edit_distance_dictionary,prefix_length)
if not corector.create_dictionary('../dictionary/frequency_dictionary_en_82_765.txt'):
   print("dictionary not found")
   exit()
input_term = ("whereis th elove hehad dated forImuch of thepast who "
              "couqdn'tread in sixtgrade and ins pired him")
suggestions = sym_spell.lookup_compound(input_term,
                                        max_edit_distance_lookup)
for suggestion in suggestions:
      print("{}, {}, {}".format(suggestion.term, suggestion.distance,
                                suggestion.count))
```

Yukarıdaki yapıyı biraz daha aydınlatırsak, ilk gelen veri yanlış yazıma ait örnek olup ikinci sözlük yapısında olası doğru kelime listesi ve en sondaki parametre ise yazım yanlışına ait dökümandaki frekans değeridir. Bu aşamada sözlüğümüzü oluşturduktan sonra kelimeye ait olası tahminleri almak için aşağıdaki kodları çalıştırıyoruz.

“*whereas the love head dated for much of repast who couldn't read in six grade and inspired him, 10, 1*”

Symspell kütüphanesi yüksek oranda metni doğru tahmin edebilmektedir. Bu algoritmanın yazım düzeltme algoritmalarının zayıflığından kaynaklanan bazı sınırlamaları mevcuttur. Örneğin yanlış yazılmış kelimenin yerine konulabilecek kelimenin birden fazla alternatifi olabilmektedir. Bu alternatiflerin seçimi sırasında yapılcak hata ile metnin anlamı bozulabilmektedir. Bu sınırları kaldırmak tezimizin dışında kalmaktadır.

3.5.4. Özellik çıkarımı

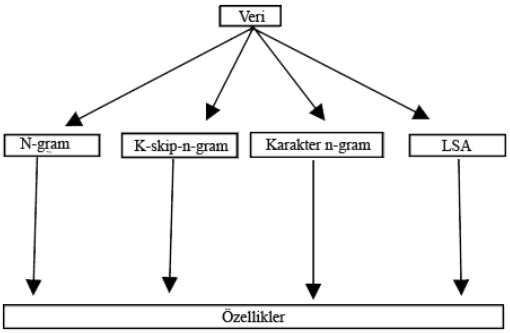
Döküman vektörlerini oluşturmak için metinsel verileri sayısal bir forma dönüştürmek gerekmektedir. Bu şekilde oluşturulan vektörler üzerinde makine öğrenmesi yöntemleri kullanılarak eğitilebilmektedir.

Oluşan vektör modelinde her boyut ayrı bir terime denk düşmektedir. Bir belgede bir terim bulunuyorsa, oluşan vektörde ilgili boyutun değeri sıfırdan farklıdır. Bir dökümandaki terimlerin ağırlıklandırılması için birçok yöntem ortaya atılmıştır. Vektörlerin terimleri olarak kelimeler, seçilmiş anahtar kelimeler ya da daha uzun kelime bileşke yapıları kullanılabileceği gibi; 2-gram, 3-gram ya da n-gram gibi hece yapıları da kullanılabilir. Belgeler, terimleri kelimelerden oluşan vektörler şeklinde gösterildiğinde bu yapıya kelime torbası (bag-of-words) denilmektedir [12].

Vineet John yaptığı [15] Yapay Sinir Ağları için metinden özellik çıkarımı ile ilgili çalışmasında genel olarak ele alınan yöntemleri incelemiştir. Bu çalışmada özellik çıkarımında performansın uygulamadan uygulamaya değiştiğinin altını çizerken farklı yöntemlerin kullanılması durumunda sonuçların da değişebileceğini ifade etmektedir.

Fatih Amasyalı ve ark. [16] yaptığı metin temsil biçimleriyle ilgili çalışmalarında farklı döküman temsil biçimlerinin farklı veri kümeleri üzerindeki etkilerini incelemişlerdir. Bu veri kümelerinde farklı temsil biçimlerinin farklı sonuçlar çıkardığı görülmektedir. Bu çalışmada 2 ve 3 harf gramların ikili ağırlıklandırma ile başarılı olduğu söylenebilir. Fakat bu çalışmada eksik noktalar bulunmaktadır. Bu noktalardan en önemlisi verinin temiz olduğu varsayımıdır. Fakat çoğu zaman veriler temiz değildir ve çeşitli saldırılar altında değişime uğramış olabilir. Bu durumda beklenen sonuçlar ile elde edilen sonuçlar arasında çok fark olmaktadır. Ayrıca algoritmada görülmemiş birçok yeni özellik ortaya çıktığı için örnekleme algoritmaları tarafından veriler yeniden oluşturulmakta olup özellik çıkarım yöntemlerine bağımlılık azalmaktadır.

Çalışmamızda önerilen özellik çıkarım yöntemi kelime torbası (BOW) üzerinde n-gram, skip-gram ve LSA yöntemlerinin uygulandığı bir algoritmadır (Şekil 3.3.). Veri kaçırma yöntemlerinin bir çoğu kelime bazlı olmaktadır. Bu durumda kelime torbası modelinin önemli derecede etkili olduğu görülmüştür.



Şekil 3.3. Özellik çıkarımı

3.5.4.1. N-gram

N-gram uzun bir metnin n-karakter parçalanması veya bölünmesidir. Literatürde dökümanda beraber geçen metinsel dökümanları da kapsamaktadır [17]. N-gramlara örnek olarak “bu döküman sınıflandırılacak” ifadesini ele alalım.

1-gram: “bu”,”döküman”,”sınıflandırılacak”
2-gram: “bu döküman”, “döküman sınıflandırılacak”, “sınıflandırılacak bu”
3-gram: “bu döküman sınıflandırılacak”

N-gramlar metin tabanlı sistemlerde çokça kullanılan bir yöntemdir. N-gram yönteminde 1-gram kullanılmasına unigram, 2-gram kullanılmasına bigram ve 3-gram kullanılmasına da trigram denilmektedir. Ayrıca n-gramları kelime grupları yerine hecelere de uygulayarak aynı şekilde karakter-gramlar da elde edilebilir.

Tez kapsamında n-gram uzunluğu 1, 2, ve 3 gramlar olarak kullanılmıştır. Ayrıca n-gram modellerine ters terim döküman frekansı (tf-idf) alınarak dökümanda çok geçen ve az geçen kelimelerin sınıflamaya etkisi eşitlenmeye çalışılmıştır.

Scikit-Learn kütüphanesiyle eğitim verisine ait 1 ile 3 arası n-gram özellikleri çıkarabilmek için aşağıdaki kodu yazılmıştır. Ayrıca "*analyser*" parametresi "char" olarak verilmesi durumunda verilen aralıkta karakter n-gramlar da çıkarabilmektedir.

```
from sklearn.feature_extraction.text import TfidfVectorizer
ngram=TfidfVectorizer(stop_words='english,"
   min_df=3,tokenizer=tokenizer,ngram_range=(1,3),analyzer="word")
ngram.fit_transform(X_t)
```

3.5.4.2. K-skip-n-gram

Kelime imzalama yöntemlerine temel teşkil eden k-skip-n-gram yöntemi, n-gram yönteminin genişletilmiş bir hali olan k-skip-n-gram, metin tespit ve sınıflandırma algoritmalarında kullanılan bir yöntemdir. N-gram yöntemi de eskiden bu yana metin sınıflandırmada kullanılmış bir yöntemdir. N-gram, daha uzun bir yazı dizisinden kesilmiş yan yana n tane karakterden ya da kelimeden oluşan bir dizidir. Bu dizilerin uzunluğuna göre uni-gram, bi-gram, tri-gram gibi terimlerle de bu yöntem ifade edilmiştir. Literatürdeki çalışmalarda hem karakter bazında hem de kelime bazında değerlendirmeler mevcuttur. DLP kapsamında anahtar kelimeler daha öne çıktığından n-gram kelime odaklı olarak değerlendirilecektir. Bu kesilmiş diziler oluşturulurken her seferinde bir kelime ilerlenerek aslında üst üste geçen birçok n-gram oluşturulmuş olur. Böylece bir kelimenin n yakınlığındaki bütün kelimeler dikkate alınmış ve farklı metinlerde farklı şekillerde yer alması muhtemel olan bu kelimelerin ne tür bir içerik içerisinde yer aldığı tahmin edilmeye çalışılmış olur.

Bu n-gramlar diğer metinlerde aranırken sadece bir kelimenin aranması sırasında yaşanabilecek yanlış alarm (false positive) durumunun önüne geçilmeye çalışılmakta ve orijinal metinde yer aldığı bağlamda aranmaktadır.

N-gramlar belli bir başarım sunsa da dilin yapısı gereği bir kelimenin hemen yanındaki kelimelerin bazen bağlam için gerekli bilgiyi taşımaması ve farklı cümleler içerisinde aranan kelime ile bir önceki bağlamdaki kelimeler arasına tamlayıcı nitelikte kelimelerin girmesi nedeniyle her durumda istenilen başarım sağlanamamaktadır. Bu sebeple bu yaklaşım, n kelimeli diziyi arada k tane kelimenin atlanarak oluşturulduğu

k-skip-n-gram yöntemi ile genişletilmiştir. Bu yeni yöntem daha başarılı sonuçlar sunmuştur. Ayrıca k değerinin değiştirilmesi ile ilgili kelimenin içinde bulunduğu bağlam daha esnek olarak ele alınabilmektedir. Bu da uzun ve tamlamaların çok olduğu cümlelerdeki başarımı arttırmaktadır.

Bu yöntemin bir diğer artısı ise daha basit karşılaştırmalar yapması ve doğrudan metin tabanlı değil imza tabanlı çalışması sebebiyle hızlı bir algoritma sunmasıdır. Bununla beraber, cümlenin yapısının değiştirilmesi, kelime ya da harf değiştirme/ekleme tarzı saldırılar bu algoritmalara karşı etkili olmakta ve veri sızıntısını engeleyebilmektedir. Bu sebeple tek başına kullanılması yerine anlamsal analiz ve doğal dil işleme yöntemleri ile beraber kullanılması güçlü bir algoritma elde etmek için gerekli görülmektedir.

Skipgram modeli için nltk ve Scikit-Learn kütüphanesi birlikte kullanılarak gerçeklenebilmektedir. Burada "*build_analyzer*" metodu ezerek (override) skipgram'lar daha sonra tf-idf dönüşümü yapılabilecek formata getirmek için uygun hale getrilmiştir. Aşağıdaki kodda yaptığımız dönüşüm görülmektedir.

```
k=2;n=2
from sklearn.feature_extraction.text import  TfidfVectorizer
from nltk import skipgrams
class Skipgram(TfidfVectorizer):
   def __init__(self,n,k,**kwargs):
      super(Skipgram,self).__init__(**kwargs)
      self.k=k
      self.n=n
   def build_analyzer(self):
      analyzer=super(Skipgram,self).build_analyzer()
      def skip(doc):
         j=skipgrams(doc,self.n,self.k)
         j=list(j)
         return (d for d in j)
      return lambda doc:skip(analyzer(doc))

vect = SkipGramVectorizer(tokenizer=tokenizer,min_df=3,k=k,n=n)
vect.fit(X_train)
```

Tezimizde 2-skip-2-gram kullandığımız için k ve n değerlerine 2 değerini atanmıştır. Bu yüzden her özellikten 2 adet alınmıştır. Bu işlemi yapabilmek için kayan pencere denilen değeri 4 olarak atanmıştır.

3.5.4.3. Karakter N-gram

Karakter n-gramlar döküman hakkında karakteristik bilgiler verebilmektedir. Karakter n-gramlar bir kelimenin hangi dile ait olduğu bulmakta kullanılmakla beraber, ayrıca birlikte geçen kelimeleri bularak tamlamaları bulmakta da kullanılabilmektedir. Ayrıca uygun seçilen karakter gramlarla dilin modellenmesi de sağlanabilmektedir. Bunun dışında özellikle çok yazım hatasının bulunduğu metinlerde karakter n-gramlar yanlış yazıma ait desenleri bulmakta güçlü bir yapı sağlar. Sparse matrislerinin oluşumu kelime n-gramlarına göre daha azdır. Ayrıca n değeri belirli aralıklarla seçilerek kelime n-gramlarına ulaşılabilmesi sağlanabilir. Karakter n-gramlar birçok problemin çözümünde kullanılmaktadır. Özellikle yazar tanıma, cinsiyet belirleme ve dil tanıma alanlarında güçlü bir özellik çıkarım yöntemi olarak kullanılmaktadır.

3.5.4.4. Gizli anlamsal analiz (Latent Semantic Analysis)

Latent Semantic Indexing (Açık anlamsal indeksleme) yöntemi, anahtar kelimeye benzer içerikler bulmaya yarayan bir doğal dil işleme (NLP) algoritmasıdır [22]. Buradaki amaç bir dökümanı en iyi ifade eden kelimeden yararlanarak, verilen anatar terimin ne ifade ettiğini bulmaya çalışmaktır. LSA algoritması, doküman ile terim arasındaki ilişkiyi açığa çıkarmak için terimlerin bulunduğu matristen Tekil Değer Ayrışımı (Sigular Value Decomposition-SVD) denilen yöntemi kullanarak matrisin boyutunu düşürerek, cosinüs uzaklığına (Cosinus Similarity) göre en yakın dökümanı bulur [23].

Uygulamamıza LSA entegre etmek için Scikit-Learn kütüphanesini kullanacağız. [42] referansına göre uygulamamıza ait kodlarımız aşağıdaki gibi olmaktadır.

```
tfidf_model2 = TfidfVectorizer(tokenizer=nltktokenizer, min_df=min_df)
```

```
character_model = TfidfVectorizer(analyzer="char", ngram_range=(3, 5),
min_df=min_df)
lsa_model = make_pipeline(tfidf_model2, TruncatedSVD(n_components=100))
```

Yukarıdaki kodlarımızda "*n_components*" parametresi ile çıkacak matris boyutu belirlenmektedir. SVD ile ayrışan matrisimizde dökümandaki en anlamlı terimlerden oluşan matris oluşturulmaktadır. Scikit-Learn referans dökümanına göre "*n_components=100"* alınması tavsiye edilmektedir. Fakat buradaki değerimiz toplam özellik vektöründen büyük olmaması önemlidir. Çünkü yapılan işlemle döküman daha az boyutlu bir uzayda değerlendirilmektedir. Ayrıca elde edilen dökümanı hangi algoritmaya göre çözüleceği "*algorithm*" parametresiyle belirlenebilmektedir. Burada "string" veya "randomized" olmak üzere 2 değer girilebilir. "*n_iter"* parametresi ise "randomized" seçildiğinde kaç iterasyon yapılacağını belirlemektedir. Diğer parametreler ise isteğe bağlı parametreler olup, "*random_state*" eğer RandomState nesnesi girilmişse bu nesne rastele sayı üretiminde kullanılır. "*None*"olarak seçilirse numpy kütüphanesindeki numpy.random fonksiyonu kullanılarak rastgele sayı üretilir. "*tol*" değeri ise çözümde kullanılan algoritmanın tolerans değerini belirler. Scikit-Learn kütüphanesi ardışıl yapılacak işlemler için Pipeline denilen bir kavramı kullanmaktadır.

```
vect=TfidfVectorizer(stop_words='english',min_df=3)
lsa=TruncatedSVD(n_components=10)
lsa.fit_transform(vect.fit_transform(X_train))
```

LS, anlamsal olarak birbirine yakın kelime saldırılarına ve çok anlamlı kelime saldırılarına karşı etkili olabileceği görülmüştür.

3.5.5. Özellik Seçimi - Terim ağırlıklandırma

Dokümanlardaki metinlerdeki kelimelerin sayısının (kelime frekansı) oluşturduğu matrise kelime torbası (Bag of word -BOW) adı verilmektedir. Çıkarılan kelime torbası modelinde (BOW) herbir terimi ağırlıklandırmak için farklı yöntemler bulunmaktadır [16]. Bu yöntemler genel olarak frekans ağırlıklı olmakla beraber ikili (binary) olarak da ağırlıklandırılmaktadır. İkili ağırlıklandırma verilen terim

dökümanda varsa 1 yoksa 0 değeri atanmaktadır. Frekans tabanlı ağırlıklandırmada ise dökümanda tekrar eden terimin kaç adet geçtiği hesaplanmaktadır.

Terim frekansı (Term Frequency) ise bir dökümanda geçen terimin, o dökümanda en çok geçen terimin frekansına bölünmesidir [69].

Bu modelde her terim bir sütun olarak düşünülmekte ve dökümanlar da birer satır olarak ele alınıp bir tablo şekli çıkarılmaktadır. Burada bazı dokümanlardaki bazı kelimelerin frekansının çok yüksek olması sınıflandırmada veya dokümanı tanımlamada sorunlara yer açmaktadır. Bu durumdan kurtulmak amacıyla TF-IDF denilen dönüşüm yapılmaktadır. D = {d1,d2,…,dn} gibi dokümanlardan oluşan bir küme, t bu dokümanlardaki bir terim olmak üzere ve tf(t, d) ifadesi t teriminin d dokümanındaki frekans değerini göstermek üzere,

$$idf(t) = \log \frac{|D|}{1+|\{d : t \in d\}|} \tag{3.1}$$

(Deklem 3.1) ile ters döküman frekansı hesaplanabilir. Burada formülü açıklarsak, |D| toplam döküman sayısı, burada n adet döküman olduğu için |D|=n, terimin geçtiği toplam doküman sayısına 1 ekleyerek bölüyoruz. Paydadaki değerin 0 (sıfır) olmaması için 1 değeri eklenmekte olup elde edilen sayının logaritması alınmaktadır. Bu değer bulunduktan sonra, TF-IDF değeri şu şekilde hesaplanır.

$$tf - idf(t) = tf(t,d).idf(t) \tag{3.2}$$

Buradaki değere biraz daha açıklık getirirsek, örneğin 50 adet döküman elimizde bulunsun. Bu dökümanda “private” kelimesi ele aldığımız dökümanda 2 defa geçmiş olsun. Bu ele aldığımız dökümandaki (d) en çok geçen kelime frekansını 40 olarak ele alırsak,

tf(private,d)=2/40=0,05

olarak bulunur. Daha sonra “private” kelimesini içeren toplam döküman sayısı örneğin 25 olsun. Bu durumda,

Idf(private)=log(50/(1+25))=0.28

olarak buluyoruz. Daha sonra da bu değerleri çarparak TF-IDF değerini buluyoruz.

tfidf(private,d)=0.05*0.28=0.014

Scikit-Learn kütüphanesinde TF-IDF dönüşümünde TfidfVectroizer sınıfı kullanılmaktadır. Bu sınıf TF-IDF matrisini oluşturduktan sonra kullanılmak amacıyla sparse formatına dönüştürmektedir. Ayrıca “*ngram_range*” parametresi sayesinde n gramlara göre değerleri oluşturmaktadır.

```
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
text="This is a simple text to be tokenized. This is another paragraph to be extracted"
tf=TfidfVectorizer(stop_words="english",ngram_range=(1,2))
tf.fit_transform([text])
print(tf.vocabulary_)
print(tf.get_feature_names())
```

Yukarıda n gram genişliği 1 ve 2 alınarak bir TF-IDF matrisi oluşturulmuştur. Bu matristen *tf.vocalulary_* ile kelime frekansları, *tf.get_feature_names()* ile de oluşturulan özellikler listelenmiştir. Bu kodun çıktısı sırasıyla aşağıdaki gibidir.

{u'simple text': 4, u'simple': 3, u'text': 5, u'tokenized paragraph': 8, u'tokenized': 7, u'extracted': 0, u'paragraph': 1, u'text tokenized': 6, u'paragraph extracted': 2}

[u'extracted', u'paragraph', u'paragraph extracted', u'simple', u'simple text', u'text', u'text tokenized', u'tokenized', u'tokenized paragraph']

Idf değerlerine ulaşabilmek için tf.idf_ komutunu kullanmak gerekiyor. Uygulamada karşılaşılan matrisler çok büyük boyutlu olabilir. Bu çok boyutlu matrislerin yaklaşık olarak %95-%99 u sıfır olan matrislerden oluşursa bu matrise seyrek matris (Sparse matrix) denilir. Bu matrisler ile çalışırken sıfırların depolanmasında ve sıfır sayısı ile dört işlem yapmaktan kaçılır. Bu şekilde çalışma zamanı ve karmaşıklık azaltılır. Ayrıca hafızada da az yer kaplar. Sparse matrislerini göstermek için 3 satırlı bir yapı kurulabilir. İlk satırda satır numarası, ikincisinde sütün numarası ve üçüncü satırda ise değeri olacak şekilde yeni bir matris tanımlanabilir. Bu çalışmada n-gramlara TF-IDF yöntemi uygulanmıştır ve terim frekansı 3'ten küçük olan terimler (n-gramlar ve LSA için kelimeler) elimine edilerek özellik sayısının azaltılması sağlanmıştır.

Özellik çıkarımı ve özellik seçimi ile alakalı yapılan çalışmalar sonrasında elde ettiğimiz özellik sayıları Tablo 3.6.'da verilmiştir.

Tablo 3.6. Özellik çıkarım yöntemleri sonucu elde edilen özellik sayısı

Veri seti	Eğitim	n-gram	k-skip-n-gram	Karakter gram	LSA
TM	1	6253	74509	7658	100
Dyncorp	1	8742	61112	10805	100
Mormon	1	21674	124900	34025	100
Tümleşik	1	31157	49525	152783	100
TM	2	6516	63883	7723	100
Dyncorp	2	8768	53402	10988	100
Mormon	2	21674	124900	34025	100
Tümleşik	2	30830	53229	101863	100
TM	3	6392	74509	7803	100
Dyncorp	3	8828	61112	11069	100
Mormon	3	22302	124900	38462	100
Tümleşik	3	31780	54364	152783	100
TM	4	6357	74509	7803	100
Dyncorp	4	8827	61112	11069	100
Mormon	4	22184	124900	38168	100
Tümleşik	4	31596	54153	152783	100

Tablo 3.6.'daki veri setleri ayrı ayrı olarak değerlendirilmelidir. Bu tablolardaki çıkarılan değerler o veri setine ait özelliklerden oluşmaktadır. Oluştuduğumuz "Tümleşik" veri seti, Dyncorp, Mormon, TM ve Dbpedia veri setlerinin birleştirilmesiyle oluşturulan yeni bir veri setidir. Bu setteki özelliklerin çıkarımı aşamasında ortak geçen özellikleri olmaktadır. Dolayısıyla burada ele aldığımız "Tümleşik" veri setindeki özelliklerin sayısı, diğer veri setlerinin toplamına eşit olmamıştır.

3.5.6. Sınıflandırma

Sınıflandırma algoritmaları verilen dökümanın hangi sınıfa dahil olduğunu bulmak için kullanılır. Sınıflandırma işlemi; makine öğrenmesi, yapay sinir ağları ve bulanık mantık gibi değişik bilim dallarında çözüm bulunabilen bir problemdir.

Kumar Ravi ve Vadlamani Ravi [25] yaptıkları duygu analizi (sentiment analysis) incelemesinde internetteki kullanıcı yorumlarının yönelimi ile ilgili yapılan birçok yöntemi incelemiştir. Ayrıca yapılan çalışmalar dışında uygulanabilecek yöntemleri belirtmiştir. Bu incelemede karar destek vektörleri, yapay sinir ağları ve sözlük bazlı yöntemler kullanıldığını belirterek, akıllı bazı yöntemlerin kullanılmadığını ifade etmiştir. Kullanılmayan bu yöntemlerden birkaçı, rastgele orman (random forest), evrimsel algoritmalar, ilişkili kural çıkarımı, bulanık kural tabanlı sitemler, kural çıkarıcı ve durumsal rastgele alan, konsept analizi, radyal tabanlı yapay sinir ağları fonksiyonu ve online öğrenme algoritmalarıdır. Kural tabanlı algoritmalar beraber kullanılan ortak kelimeleri bulmak için, evrimsel algoritmaları özellik çıkarımı için, hala etkili olarak kullanılmamasına rağmen, kullanılabileceğini belirtmiştir. Yine aynı çalışmada verilen cümlelerin belirsiz durumlarında bulanık mantık daha etkili olabileceğini belirtmiştir.

Metin analizi ile ilgili birçok yapılan çalışma mevcuttur. Fakat bu yöntemlerin detaylı bir karşılaştırılması yapılmamıştır. Metin analizine makine öğrenmesi yöntemlerinden; navie bayes, karar destek makineleri, rastgele orman, lojistik regresyon fonksiyonu gibi yöntemler çokça kullanılan yöntemlerdir. Bu yöntemlerin hassaslığını artırmak amacıyla hibrit yöntemler kullanılmaktadır.

Rachid Beghdad [34] yaptığı ağlardaki izinsiz girişler ile ilgili çalışmasında yapay sinir ağlarını karşılaştırarak atak türlerine göre farklı algoritmaların izinsiz girişlerini yakalamada farklı sonuçlar çıkardığını belirtmiştir. Diğer taraftan yapay sinir ağlarının birçok problemde başarılı olduğunu öne sürmüştür.

Hui Yang ve ark. [26], intihar notlarındaki duygu çıkarımı çalışmalarında durumsal rastgele alan (conditional random field), maksimmum entropy, karar destek makineleri ve naive bayes yöntemlerini kullanarak bunları karşılaştırmıştır. Karar destek makineleri diğer algoritmalara göre yüksek başarı elde etmiş fakat yanlış alarm sayısı da yine fazla çıkmıştır.

3.5.6.1. Rasgele orman (Random forest)

Rastgele orman (Random Forest), ağaç tipi sınıflandırıcılar topluluğu olarak tanımlanabilir. Rastgele orman, tüm değişkenler arasında en iyi dalı kullanarak her bir düğümü dallara ayırmak yerine, her bir düğümde rastgele olarak seçilen değişkenler arasında en iyisini kullanarak her bir düğümü dallara ayırır. Her bir veri seti orijinal veri setinden yer değiştirmeli olarak üretilir. Sonra rastgele özellik seçimi kullanılarak ağaçlar geliştirilir. Geliştirilen ağaçlar budanmaz [30].

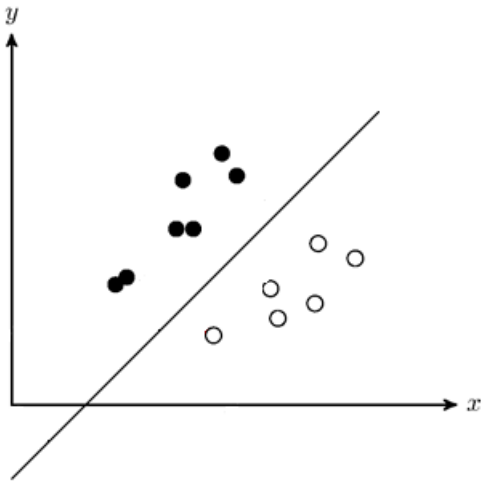
Scikit-Learn kütüphanesinde n_estimators parametresi ile ormanda kaçar adet ağaç bulunacağını belirliyoruz. Defakto değeri 100 olmaktadır. Ayrıca criterion değeri de ağaçların parçalanmasının bir ölçüsü olup gini veya entropy algoritması seçilebilir. max_features değeri parçalanma sırasında seçilecek maksimum özellik değerini belirler. max_depth ile de ağacın derinliğinin maksimum değeri belirlenir. random_state değeri ile de oluşturulacak rastgele sayı değeri belirlenir.

Scikit-Learn kütüphanesinde rasgele orman sınıflandırıcı topluluk (ensemble) sınıflandırıcı modülünde olup aşağıdaki şekilde kullanılır.

```
from sklearn.ensemble import RandomForestClassifier
Rf=RandomForestClassifier(n_estimators=100)
Rf.fit(X_train)
Rf.predict(X_test)
```

3.5.6.2. Karar destek makineleri (Support vector machines)

Karar destek makineleri (SVM) istatistiksel bir öğrenme kuramıdır. Sınıflama ve regresyon için kullanılabilir. [28] Bu algoritma ile n özelliğe (feature) sahip bir veri kümesini bir düzlem (hyper-plane) ile ayırmaya çalışır. Düzlemde verilen veri setlerini iki farklı grup olarak ayırdığımızı varsayarak grupları ikiye ayıran bir düzlem çizilebilir (Şekil 3.4.).

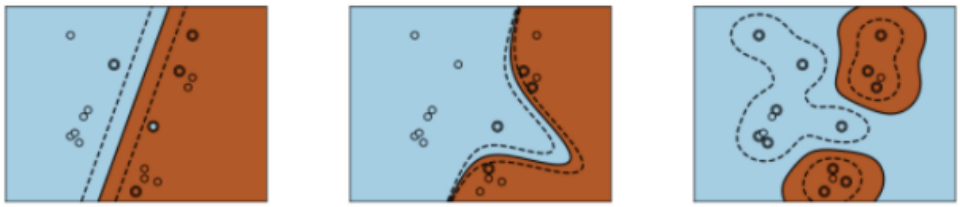


Şekil 3.4. Karar destek makineleri sınıflama

Eğer veri kümesi lineer değilse,lineer olmayan bir kernel fonksiyonu sayesinde özellik uzayından yüksek boyutlu bir uzaya bu fonksiyon uygulanarak birçok noktadan oluşan bir düzlem elde edilebilir [29].

Karar destek vektörleri birçok problemde kullanılan yaygın bir sınıflama algoritmasıdır. Bu algoritma kullanılarak doğrusal ve doğrusal olmayan problemler kolay bir şekilde çözülebilir. Scikit-Learn kütüphanesinde karar destek vektörleri sınıflama ve regressyon analizi için özelleştirilmiş modülleri mevcuttur. Bu modüller farklı problemlere göre oluşturulmuş olup önemli parametreleri ise C, gamma, kernel değerleridir. Buradaki kernel parametresi algoritmanın çalışma şeklini belirler. Burada poly, rbf, sigmoid, precomputed değerleri verilebilir. Eğer herhangi bir değer seçilmemişse rbf uygulanır. rbf ile yapay sinir ağı gibi davranır. C parametresi ise algoritmanın optimizasyonu ile ilgilidir. Bu parametre herbir eğitim kümesinde ne

kadarlık bir hata ile sınıflandırma yapılacağını belirler. Çok büyük C değeri yakın düzlemler seçerken çok küçük değerler ise biraz daha fazla açıklıkla düzlemleri seçer ve gamma değeri ile de eğitim düzeyinin etkisini etkiler. Küçük değerler daha uzak noktaların alınmasını, büyük değerler ise daha yakın noktaların alınmasını sağlar.



Şekil 3.5. Svm önemli kernel çeşitleri

Tezimizde, yine defakto parametreleri uyguladık. Bu pametreler kernel rbf, gama değeri auto, derece (degree) değerini 3, tolerans değerini ise 0,001 olarak aldık. Buradaki kernel fonksiyonu rbf ve gamma parametresini auto vererek aşağıdaki şekilde kodumuzu kullandık.

```
from sklearn import svm
clf = svm.SVC(kernel=kernel, gamma="auto")
```

3.5.6.3. Çok katmanlı yapay sinir ağları (Multi layer perceptron)

Yapay sinir ağları, insan beyninin çalışma şeklini örnek alarak geliştirilen bir dizi algoritmalardan oluşur. Bir yapay sinir ağı (Neural Netwok) biribirine bağlı nöronlardan oluşur. Her bir nöron bir girdi değeri için bir çıktı değeri üreten foksiyondan oluşur. Burada kullanılan fonksiyona aktivasyon (activation function) denilir [31].

Scikit-Learn kütüphanesinde çok katmanlı perceptron yapay sinir ağları için kullanılabilecek önemli parametreler hidden_layer_size gizli katmanda bulunabilecek nöron sayısını belirlemekte, activation değeri ise gizli katmandaki aktivasyon fonksiyonunu belirler. Burada kullanılabilecek fonsksiyonlar identity, logistic, tanh,

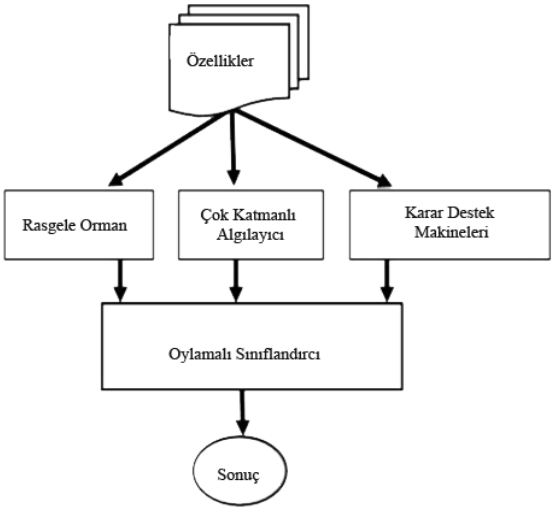
ve relu değeridir. Defakto olarak relu fonksiyonu seçilidir. Identity değeri aktivasyon kullanılmayacağını belirtir. Yapay sinir ağındaki ağırlıkları optimize etmek için bir çözüm fonksiyonuna ihtiyaç duyulmaktadır. Bu çözüm fonksiyonu solver parametresiyle belirlenmektedir. Solver değerine lbfgs, sgd, ve adam değerleri atanabilir. adam yüksek boyutlu veri kümeleri üzerinde iyi çalışmakta olduğu için bu çalışmada adam kullanılmıştır. Ayrıca yapay sinir ağlarında öğrenme düzeyi learning_rate parametresiyle belirlenir. Bu değer ağırlıkları güncellemek için kullanılır. learning_rate değeri constant, invscaling, adaptive parametrelerini alır. Burada hidden_layer_sizes, 60 ve diğer değerler defakto değerler seçilerek uygulanmıştır.

```
from sklearn.neural_network.multilayer_perceptron import MLPClassifier
mlp=MLPClassifier(hidden_layer_sizes=(60,))
```

3.5.6.4. Oylamalı sınıflandırıcı (Vote classifier)

Oylamalı sınıflandırıcıların arkasındaki temel felsefe, değişik makine öğrenme algoritmalarının uygulanması sonucu en çok oy alan veya ortalama tahmin değerine (soft vote) bağlı olarak dökümanı sınıflandırıyor olmasıdır. Bu şekilde algoritmalardaki zayıflıkları azaltarak daha iyi sonuç üretmesini sağlar [33].

Sriparna Saha ve Esif Akbal [32] yaptıkları çalışmada oylamalı sınıflandırıcıları isimli varlık tanıma (Named Entitiy Recognition) sisteminde kullanarak farklı konfigürasyonlarda etkilerini gözlemlemişlerdir. Çağatay Çatal ve Mehmet Nangir [27] duyarlılık analizi çalışmasında karar destek makineleri (bagginf), naive bayes ve karar testek makineleri (CVParameterSelection) algoritmalarına uyguladıktan sonra bu algoritmaların verdiği oya göre sınıflandırma işlemi yapmışlardır.



Şekil 3.6. Oylamalı sınıflandırma

Burada oylama sonucunda en çok oyu alan sınıfa göre sınıflandırma işlemi yapılmaktadır. İncelemelerimizin ışığında birden fazla algoritmanın aynı anda çalıştırılarak daha sağlıklı sonuçlar alınabileceği görülmektedir (Şekil 3.6.). Fakat oylama işleminde daha iyi bir karar vericinin olması daha efektif bir sonuç elde edilebilir. Dahası güçlü sınıflandırıcıların kullanılması özellikle kritik verilerin kaçırılmasına engel olabilir. Bizim öngördüğümüz çalışmada, yapay sinir ağları, rastgele orman ve karar destek makinelerinin kullanılması uygun görülmüştür.

Scikit-Learn kütüphanesinde voteclassifier modülü bu işlem için kullanılabilir. Burada kullanılabilecek önemli parametreler estimators, voting ve n_jobs'dir. Bu parametrelerden estimators kullanılacak algoritmalar olup bir liste şeklinde verilmesi gerekir. voting parametresi ile uygulanacak oylama türü verilir. Çalışmamızda kullandığımız örnek sınıflandırıcıya ait kodlar aşağıdaki gibidir.

```
from sklearn import svm
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier, VotingClassifier
clf = VotingClassifier(estimators=[
      ('svm',"svm.SVC(C=1, gamma='auto')),
      ('mlp',"MLPClassifier(hidden_layer_sizes=(60,))),
      ('rf',"RandomForestClassifier(n_estimators=100))   ])
```

BÖLÜM 4. UYGULAMA VE PERFORMANS SONUÇLARI

4.1. Performans değerlendirme

Bu bölümde bu tez çalışması içerisinde iki farklı amaçla yapılan performans değerlendirmesi kapsamı ve sonuçları anlatılacaktır. Birincisi TM, Mormon ve DynCorp veri setlerine göre performans değerlendirmesi, ikincisi ise ismi verilen veri setlerinin tamamının "Tümleşik" isimli tek bir veri seti haline getirilmesinden sonra veri setinden bağımsız doküman bazında performans değerlendirmesi. Her iki değerlendirmede de yapılan testlere ait akış diyagramı Şekil 4.1'de gösterilmiştir. Test dokümanlarına öncelikle saldırı gerçekleştirilmiştir. Saldırılar gerçekleştirilirken [1] çalışmasındaki yapısal ve karartma saldırıları esas alınarak bir saldırı aracı geliştirildir. Daha sonra [2] çalışması da eklenerek geliştirilen saldırı aracının kapsamı genişletildi. Daha sonraki aşamada her iki modelin performanlarını hesaplamak için yöntemler araştırıldı. Bu yöntemlerin çoğu, hata matrisi (confusion matrix) oluşturmaya dayanmaktadır [63]. Gerçek çıktı değeri ile model tarafından tahmin edilen değerler Tablo 4.1.'de gösterilmektedir. Tabloda üçlü sınıflandırma sonuçlarının değerlendirilmesi için kullanılan performans ölçütleri recall, precission, doğruluk ve hata oranı gibi ölçütlerdir.

Şekil 4.1. Test aşamaları

Tablo 4.1. Hata Matrisi

		Gerçek değer		
		Kurumsal Gizli	Kurumsal Genel	Kurumsal Olmayan
Tahmini değer	Kurumsal Gizil	A (TP)	B (FP)	C (FP)
	Kurumsal Genel	D (FN)	E (TP)	F (FP)
	Kurumsal Olmayan	G (FN)	H (FN)	I (TN)

Tablo 4.1.'de doğru pozitif (true positive), doğru olarak tahmin edilen dökümanları göstermektedir. Burada, -1 ve +1 ile etiketlenmiş sınıflamada verinin geçek değeri pozitif ve sınıflandırıcının değeri de pozitif olarak çıkardığı anlaşılmaktadır. Doğru negatif (true negatif) ise negatif sonuçlu bir verinin yine negatif olarak sınıflandığı anlaşılmaktadır. Yani sınıflandırıcı doğru sınıflandırma yapmıştır. Yanlış pozitif (false pozitif), sınıflamada pozitif olarak işaretlenen fakat gerçekte negatif olan veri sayısını göstermektedir. Bu aynı zamanda tip 1 hata olarak da bilinir. Yanlış negatif (false negative), gerçekte pozitif olan verinin negatif olarak sınıflandırıldığı verileri göstermektedir. Bu da tip 2 hata olarak adlandırılmaktadır.

Tablo 4.1.'de üçlü sınıflamanın doğruluğu (accuracy) ve hata oranı (error rate) değeri aşağıdaki gibi hesaplanmaktadır.

$$Doğruluk(acc) = \frac{A+E+I}{A4B+C+D+E+F+G+H+I} \quad (4.1)$$

$$Hata\ Oranı = 1 - acc \quad (4.2)$$

Doğruluk sistematik hataların bir ölçüsü olarak değerlendirilebilir. Ayrıca rasgele ve simetrik hataların farklı kobinasyonlarını açıklayan bir ölçü olup yüksek değerler yapılan sınıflandırıcının yüksek doğruluk oranına sahip olduğu anlamına gelmektedir.

Sınıflandırıcının pozitif sınıf etiketlerini tahmin etmedeki etkinliliği duyarlılık (true positive rate ya da recall) olarak adlandırılmaktadır. Duyarlılık, doğru sınıflandırılan pozitif örneklerin toplam pozitif örnek sayısına oranıdır.

$$recall = \frac{TP}{TP + FN} = \frac{A}{A + B + C} \quad (4.3)$$

Doğru sınıflandırılan pozitif örneklerin toplam pozitif tahmin edilen örneklere oranına kesinlik (precision); doğru sınıflandırılan negatif sınıf etiketine sahip örneklerin toplam negatif tahmin edilen örneklere oranına negatif öngörü denir.

$$prescission = \frac{TP}{TP+FP} = \frac{A}{A+G+L} \quad (4.4)$$

İkiden fazla sınıfın söz konusu olduğu çoklu-sınıf (multi class) sınıflandırma problemlerinde modelin performansının değerlendirilebilmesi için performans ölçüsü olarak ikili sınıflandırmada kullanılan yöntemlerin genelleştirilmiş biçimleri kullanılabilmektedir [65]. Öncelikle çoklu-sınıf sınıflandırıcı, birkaç tane ikili sınıflandırıcıya dönüştürülmekte, elde edilen her ikili sınıflandırıcı için performans ölçüsü hesaplanmakta ve son olarak hesaplanan performans ölçülerinin ortalamaları alınarak çoklu-sınıf performans ölçüleri elde edilmektedir [66].

4.1.1. F-skor (F-measure)

İstatistik biliminin bir skorlama kavramı olan ve literatürde, f1 skorlama, f-ölçümü olarak geçen kavram, bilgisayar biliminde özellikle veri çıkarımı (information extraction) ve veri getirimi (information retrieval) konularında kullanılmaktadır [64].

Kesinlik ve duyarlılık performans değerlendirme ölçülerinin harmonik ortalamasıdır.

$$f-score=\frac{2*precission*recall}{precission+recall} \tag{4.5}$$

Sınıflandırmada test verilerinin bir hassaslık ölçüsü olarak değerlendirilebilir. Ayrıca β>0, bir gerçel sayı olmak üzere genel formül aşağıdaki gibi tanımlanır.

$$F_{\beta}=(1+\beta^{2})*\frac{precision*recall}{(\beta^{2}*precision)+recall} \tag{4.6}$$

Buradaki β değeri 1 olaraka alındığında F_1-skor elde edilir. Ayrıca β değerinin seçimi kesinlik ve hassasiyet arasındaki dengeyi sağlar. Örneğin β=2 seçilmesiyle hassasiyeti iki katı etkili kılarken β=0.5 seçilmesi, kesinlik etkisini ik misline çıkarır.

Scikit-Learn kütüphanesinde F-skor modülü *metrics* içinde yer almaktadır. Parametere olarak y_true (doğru etiketler), y_pred (tahmin edilen etiketler) ve average değeri önemlidir. Average değeri recall ve precision değerlerinin ağırlıklandırılmış ortalamsını göstermektedir. F-skor değeri en iyi durum için 1 değerini gösterirken en kötü durum için ise 0 değerini göstermektedir. Ayrıca F_1-skor için precision ve recall değerleri kısmi katkası eşitttir. Burada average değeri "binary", "micro", "macro", "samples" ve "weighted" olarak atanabilir. Bu parametreler ikiden etiketli sınıflandırmada kullanılabilir. "binary" değeri sadece hedef sadece ikili (binary) ise kullanılabilir. "micro" değeri, toplam doğru pozitif, yanlış negatif ve yanlış pozitiflerin sayısına saymaya bağlı bir hesaplama yapmaktadır. "macro" değeri verilmişse, her etiket için metrik hesaplanarak, ağırlıklandırılmamış ortalamaları bulunur. Bu yöntem etiket dengesizliğini hesaba katmaz. "weighted" olarak seçilirse, her etiket için deskteklenen ağırlıklara bağlı olarak, ortlamaları bulur. Bu yöntem "macro" olarak seçilen parametreyi değiştirerek dengesiz etiketleri değiştirir. Aynı zamanda bu değer precision ve recall arasında olmayan bir F-skor ile sonuçlanabilir. Eğer average değeri "samples" olarak seçilirse, herbir örnek içi metrikleri hesaplayarak ortalamaları bulur. Bu yöntem sadece çok sınıflı sınıflandırma işleminde kullanılır [62]. Tezimizde kullandığımız F-skor değeri scikit-learn kütüphansesindeki ortalama ağılıkları alınarak elde edilmiştir. Burada örnek kod aşağıda görülmektedir.

```
from sklearn.metrics import f1_score
print(f1_score(y_test, predicted, average='weighted'))
```

4.2. Veri Setlerine Göre Performans Değerlendirmesi

Zararlı yazılım saldırılarına karşılık, dayanıklı bir döküman sınıflandırma algoritması geliştirmek amacıyla, geliştirilen modelin çapraz geçerleme (crossvalidation) işlemini yapabilmek için birini dışarda tut (holdout) yöntemi kullanılmıştır. Bu yöntemde veri, eğitim ve test veri seti olarak ikiye ayrılmıştır. Dökümanlar her bir veri setinin kendi içerisinde Gizli (G), Kurumsal Genel (KG) ve Kurumsal Olmayan (KO) olacak şekilde etiketlendirilmiş olup modelde, test verisi modelin performansını ölçmek için kullanılmıştır. Veri setinin %70 eğitim verisi seçilirken, diğer %30 ise test verisi olarak seçilmiştir. Daha sonra yaptığımız çalışmada %30 olarak seçilen kurumsal olmayan (KO) dökümanların fazla olmasından dolayı test sınıflamaya ait f-skorların yanıltıcı sonuç verildiği görülmüştür. Bundan dolayı veri setinde adet sayıları düşürülmüştür. Eğitim ve test aşamlarında kullanılan dokümanların veri setine bağlı olarak değerleri Tablo 4.2.'de verilmiştir.

Tablo 4.2. Veri setideki kullanılan eğitim ve test doküman sayısı

Veri seti	Eğitim (adet)			Test (adet)		
Doküman tipi	G	KG	KO	G	KG	KO
TM	68	88	1387	17	32	50
MORMON	424	1784	1385	169	169	169
DYNCORP	1	137	1402	1	61	61

Değerli verilerin ufak bir değişiklikle veya Gelişmiş Sürekli Tehdit (APT) aracılığıyla çok rahat bir şekilde kurumun dışına çıkarılabildiği mevcut çalışmalarda dile getirilmektedir. Değerli ve hassas olan verilerin güvende tutulması amacıyla kararlı ve saldırılar altında dahi iyi, içeriğin veya dokümanın türünün ne olduğu bilgisini bulabilecek, bir sissteme ihtiyaç duyulmaktadır. Bu amaçla mevcut saldırı türlerini inceleyerek yapısal ve karartma saldırılarını hedefleyecek bir algoritma oluşturulmuştur. Test verisinden iki tür veri seti oluşturulmuştur.

1. Üzerinde herhangi veri kaçırma atağı uygulanmamış test verisi. Aşağıdaki tablolarda Normal Veri olarak tanımlanmış satırları göstermektedir. Test işlemi, Tablo 3.6.'da verilen özelliklerin tamamına göre yapılan eğitim sonucundaki sınıfladırıcılarla yapılmıştır.
2. Üzerinde aşağıdaki Tablo 4.3.'de verilen Yapısal ve Karartma Saldırı türlerine göre veri kaçırma atakları uygulanmış test verisi. Aşağıdaki tablolarda Normal satırı haricinde olan diğer satırların tamamını tanımlamaktadır. Aşağıdaki tabloda verilen yapısal ve eş anlamlı veri kaçırma saldırıları (saldırı vektörleri), tüm test gruplarına yapılarak tüm saldırılara ait test verileri elde edilmiştir. Test işlemi, Tablo 3.6.'da verilen özelliklerin tamamına göre yapılan eğitim sonucundaki sınıfladırıcılarla yapılmıştır.

Tablo 4.3. Test verisine yapılan saldırı türleri

Saldırı türü		Yapılan saldırı	Pozisyon	Saldırı Kodu
Normal	Yok	Yok	Yok	s0
Yapısal saldırılar	Yer değiştirme	Paragrafın yerini değiştirme	Rasgele	s1
	Yerine koyma	Summarize ile metin özetini çıkarma	Tümü	s2
	Modifikasyon-kelimeyi değiştirme	Kelimeyi değiştirme	Tüm metinde kelimelerin sonuna harf eklendi	s3
		Kelimeyi değiştime	Tüm metinde kelimelerin başına rasgele harf eklendi	s4
		Kelimeyi değiştime	Tüm metinde kelimelerin ortasına rasgele harf eklendi	s5
		Kelimeyi değiştime	Tüm metindeki boşluklar kaldırıldı	s6
		Kelimeyi değiştime	Tüm metindeki boşluklar yerine rasgele harf konuldu	s7
		Kelimeyi değiştime	Tüm metindeki boşluklar yerine + sembolü konuldu	s8
		Kelimeyi değiştime	Tüm metindeki kelimeden rasgele bir harf çıkarıldı	s9
		Kelimeyi değiştime	Tüm metindeki E ve e harfi yerine 1 sayısı konuldu	s10
Karartma saldırıları	Eş anlamlı	Kelimeyi eş anlamlısı ile yer değiştirme	Tüm metin	s11

Tüm verilen test gruplarına, verilen saldırılar yapılarak, tüm saldırılara ait test verileri elde edildi. Daha sonra önerilen algoritma tüm veri setideki test gruplarına 4 farklı şekilde uygulandı. Test 1'de herhangi bir yazım düzeltme işlemi uygulanmadı. Test 2'de ise yazım düzeltme işlemi dökümana önişleme aşamasında uygulandı. Test 3'te ise yazım düzeltme işlemi metne ait tokenler çıkarıldıktan sonra uygulanıp gövdeleme işlemi yapıldı. Test 4'te ise döküman tokenlere ayrılıp gövdeleme işlemine girdikten sonra yazım düzeltimi yapıldı. Bu şekilde her aşamada yapılan yazım düzeltme işleminin etkisi gözlemlemlendi (Tablo 4.4.).

Tablo 4.4. Yöntemlere ait test süreçleri

	Yazım denetim	Önişlem	Özellik çıkarımı	
			Token	Gövdeleme
Yöntem 1	YOK	Uygulandı	Uygulandı	Uygulandı
Yöntem 2	Ön işlem öncesi	Uygulandı	Uygulandı	Uygulandı
Yöntem 3	Token sonrası	Uygulandı	Uygulandı	Uygulandı
Yöntem 4	Gövdeleme sonrası	Uygulandı	Uygulandı	Uygulandı

Tablo 4.5. TM test verisine yapılan saldırılara ait f-skor değerleri.

Saldırı Türü	Yöntem 1 f-skor	Yöntem 2 f-skor	Yöntem 3 f-skor	Yöntem 4 f-skor
Normal veri	0,908	0,898	0,91	0,898
Dökümandaki kelimelerin sonuna rasgele harf ekleme	0,58	0,821	0,83	0,769
Dökümandaki kelimelerin ortasına rasgele harf ekleme	0,339	0,833	0,64	0,648
Dökümandaki kelimelerin başına rasgele harf ekleme	0,382	0,854	0,83	0,795
Dökümandaki E veya e yerine 1 yazımı	0,819	0,898	0,87	0,807
Dökümandaki kelimelerden rasgele harf çıkarımı	0,401	0,865	0,78	0,651
Dökümandaki metinlerden boşlukları kaldırma	0,517	0,691	0,52	0,524
Dökümanlardaki boşluklar yerine bir harf ekleme	0,618	0,844	0,82	0,76
Dökümanlardaki boşluklar yerine + sembolü koyma	0,566	0,898	0,55	0,591
Metnin özetini çıkarma	0,558	0,531	0,52	0,471
Metindeki kelimeleri eş anlamları ile değiştirme	0,831	0,798	0,83	0,831
Metindeki paragrafların yerini değiştirme	0,908	0,898	0,91	0,908

Tablo 4.6. Mormon test verisine yapılan saldırılara ait f-skor değerleri.

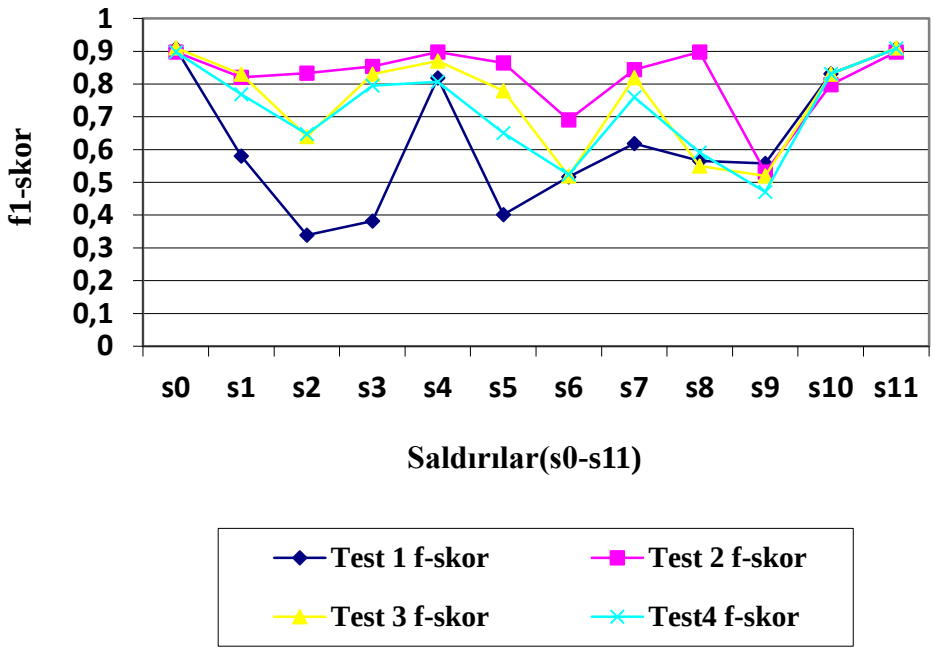
Saldırı Türü	Yöntem 1 f-skor	Yöntem 2 f-skor	Yöntem 3 f-skor	Yöntem 4 f-skor
Normal veri	0,99	0,982	0,99	0,992
Dökümandaki kelimelerin sonuna rasgele harf ekleme	0,97	0,99	0,99	0,98
Dökümandaki kelimelerin ortasına rasgele harf ekleme	0,726	0,992	0,99	0,99
Dökümandaki kelimelerin başına rasgele harf ekleme	0,95	0,99	0,98	0,99
Dökümandaki E veya e yerine 1 yazımı	0,95	0,986	0,93	0,954
Dökümandaki kelimelerden rasgele harf çıkarımı	0,942	0,964	0,99	0,974
Dökümandaki metinlerden boşlukları kaldırma	0,98	0,978	0,98	0,988
Dökümanlardaki boşluklar yerine bir harf ekleme	0,974	0,986	0,99	0,978
Dökümanlardaki boşluklar yerine + sembolü koyma	0,996	0,982	0,99	0,99
Metnin özetini çıkarma	0,984	0,978	0,99	0,992
Metindeki kelimeleri eş anlamları ile değiştirme	0,992	0,982	0,99	0,99
Metindeki paragrafların yerini değiştirme	0,992	0,984	0,99	0,992

Tablo 4.7. Dyncorp test verisine yapılan saldırılara ait f-skor değerleri.

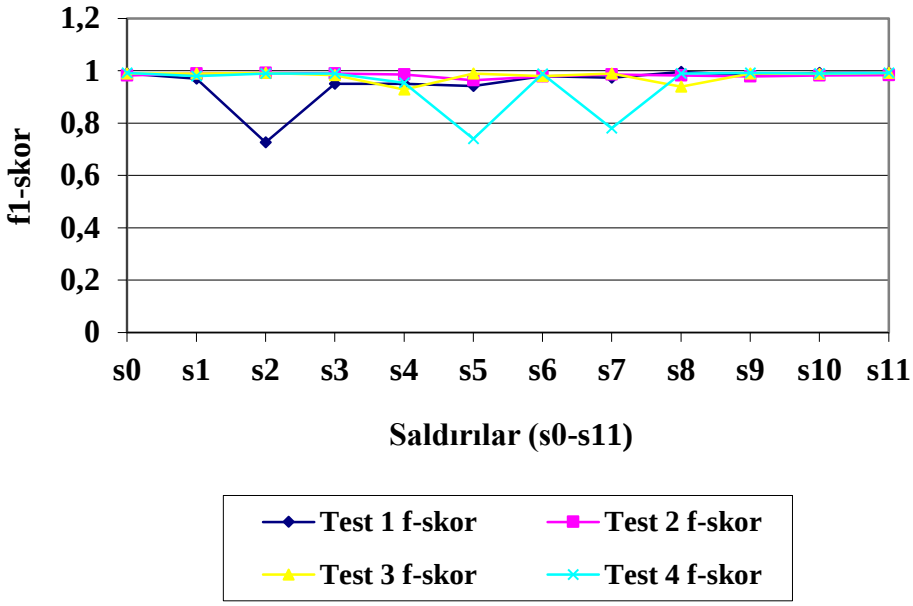
Saldırı Türü	Yöntem 1 f-skor	Yöntem 2 f-skor	Yöntem 3 f-skor	Yöntem 4 f-skor
Normal veri	0,988	0,988	0,99	0,988
Dökümandaki kelimelerin sonuna rasgele harf ekleme	0,33	0,91	0,88	0,38
Dökümandaki kelimelerin ortasına rasgele harf ekleme	0,33	0,947	0,5	0,397
Dökümandaki kelimelerin başına rasgele harf ekleme	0,33	0,947	0,63	0,329
Dökümandaki E veya e yerine 1 yazımı	0,955	0,98	0,97	0,964
Dökümandaki kelimelerden rasgele harf çıkarımı	0,33	0,939	0,66	0,364
Dökümandaki metinlerden boşlukları kaldırma	0,33	0,329	0,33	0,328
Dökümanlardaki boşluklar yerine bir harf ekleme	0,364	0,898	0,81	0,38
Dökümanlardaki boşluklar yerine + sembolü koyma	0,33	0,987	0,33	0,329
Metnin özetini çıkarma	0,429	0,346	0,44	0,429
Metindeki kelimeleri eş anlamları ile değiştirme	0,955	0,947	0,96	0,955
Metindeki paragrafların yerini değiştirme	0,988	0,988	0,99	0,98

Elde edilen sonuçlara göre modifikasyon saldırılarında yazım düzeltme işlemi başarımı artırmaktadır. Ayrıca yer değiştirme saldırılarının çok da etkisinin olmadığı görülmüştür. Yerine koyma saldırısında sistemin yakalama gücünün çok azaldığı görülmektedir.

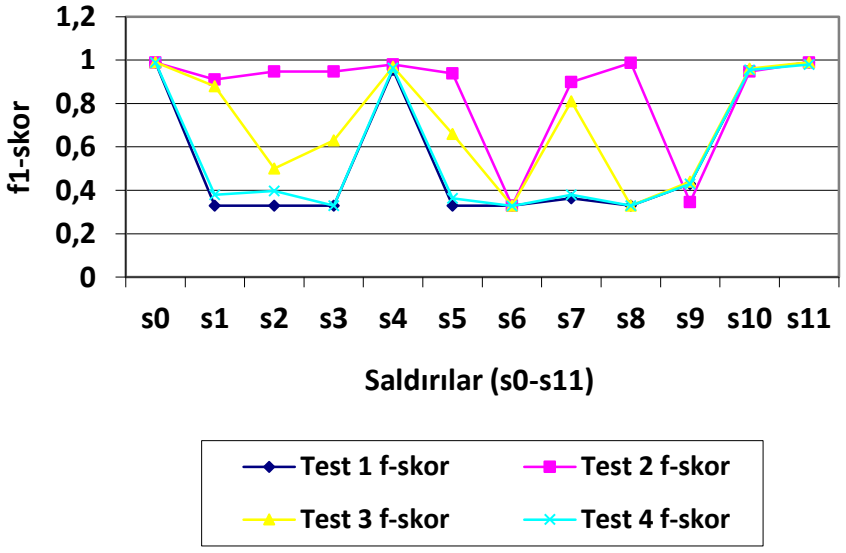
s0, s1, …, s11 saldırı vektörülerini göstemek üzere Şekil 4.1., Şekil 4.2. ve Şekil 4.3. grafikleri test sonuçlarını göstermektedir. Testlerde, Test 2 yönteminin her üç veri seti üzerinde kısmen daha başarılı olduğu görülmüştür.



Şekil 4.2. TM test grafiği



Şekil 4.3. Mormon test grafiği



Şekil 4.4. Dyncorp test grafiği

Canvar ve Trenkle [17] yaptıkları döküman sınıflandırma yönteminde kategoriler arası profil yöntemini geliştirmişlerdir. Kategoriler arası profil yönteminde eğitim dökümanına ait karakter n-gramlar 1 ile 5 arasında çıkarılmaktadır. Çıkarılan bu n-gramlar daha sonra terim frekansına göre büyükten küçüğe sıralandıktan sonra belirlenen sayıda ilk n-gram alınararak o dökümana ait kategori profili bulunmaktadır. Daha sonra yeni veriye aynı süreçler uygulanarak [17] de belirtildiği gibi kategoriler

arası mesafe ölçülmekte ve bu ölçü sonucunda bulunan en küçük dökümanın etiketi bu yeni dökümanın etiketi olarak atanmaktadır. Bu yöntemi [70] yaptığı veri sızıntısı önleme sisteminde uygulayıp çıkarılan karegori profillerinden en başarılı sonucun ilk 300 değerini alarak elde ettiğini belirtmektedir. Bu kullanılan yöntemi aynı şekilde ilk 300 değerini alarak, [71] ve [17] dökümanlarından geliştirdiğimiz yazılım aracılığıyla kendi veri setimizdeki Mormon veri setine uygulanmıştır. Elde ettğimiz f-skor değerleri Tablo 4.8.'de gösterilmiştir.

Tablo 4.8. Kategori profilleri yöntemi test sonuçları

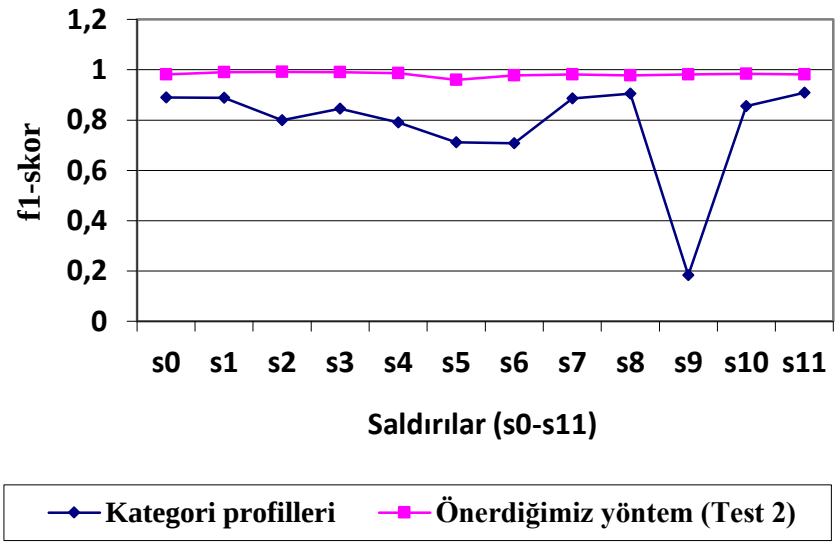
Saldırı Türü	Kategori profilleri f1-skor değerleri
Normal veri	0,89
Dökümandaki kelimelerin sonuna rasgele harf ekleme	0,889
Dökümandaki kelimelerin ortasına rasgele harf ekleme	0,80
Dökümandaki kelimelerin başına rasgele harf ekleme	0,845
Dökümandaki E veya e yerine 1 yazımı	0,79
Dökümandaki kelimelerden rasgele harf çıkarımı	0,71
Dökümandaki metinlerden boşlukları kaldırma	0,708
Dökümanlardaki boşluklar yerine bir harf ekleme	0,71
Dökümanlardaki boşluklar yerine + sembolü koyma	0,905
Metnin özetini çıkarma	0,184
Metindeki kelimeleri eş anlamları ile değiştirme	0,854
Metindeki paragrafların yerini değiştirme	0,909

Elde ettiğimiz sonuçları daha iyi değerlendirebilmek amacıyla Tablo 4.9.'da tek bir tablo haline getirdik.

Tablo 4.9. Önerilen sistem ile kategori profillerinin karşılaştırılması

Saldırı türü	Kategori profilleri	Önerdiğimiz yöntem (Test 2)
Normal veri	0,89	0,982
Dökümandaki kelimelerin sonuna rasgele harf ekleme	0,889	0,99
Dökümandaki kelimelerin ortasına rasgele harf ekleme	0,80	0,992
Dökümandaki kelimelerin başına rasgele harf ekleme	0,845	0,99
Dökümandaki E veya e yerine 1 yazımı	0,79	0,986
Dökümandaki kelimelerden rasgele harf çıkarımı	0,71	0,96
Dökümandaki metinlerden boşlukları kaldırma	0,708	0,978
Dökümandaki metinlerden boşluklar yerine bir harf koya	0,886	0,986
Dökümandaki boşluklar yerine + sembolü koyma	0,905	0,982
Dökümanın özetini çıkarma	0,184	0,978
Metindeki kelimeleri eş anlamları ile değiştirme	0,845	0,982
Metindeki paragrafların yerini değiştirme	0,909	0,984

Kategori profilleri yöntemi ile önerdiğimiz Test 2 yöntemi ile karşılaştırdığımızda Mormon veri seti üzerinde bizim yöntemimiz daha başarılı olduğu görülmüştür (Şekil 4.4.).



Şekil 4.5. Kategori profilleri ile önerilen sistemin karşılaştırma grafiği

4.3 Doküman Tabanlı Sınıflandırmaya Göre Performans Değerlendirmesi

Bir önceki bölümde sınıflandırma ve performans değerkendirmesi veri setlerine göre yapılmıştır. Bu bölümde ise veri setlerindeki Kurumsal Gizli (G), Kurumsal Genel (KG) ve Kurumsal Olmayan (KO) dokümanları birleştirilerek "Tümleşik" isimli bir veri seti oluşturulmuştur. Oluşturulan Tümleşik veri seti daha önceki çalışmada olduğu gibi %70 ve %30 oradanında sırasıyla eğitim ve test dokümanları olarak ayrılmıştır (Tablo 4.10.).

Tablo 4.9. Veri setideki kullanılan eğitim ve test doküman sayısı

Veri seti	Eğitim (adet)			Test (adet)		
Doküman tipi	G	KG	KO	G	KG	KO
TÜMLEŞİK	466	2023	1387	214	836	613

Kurumsal bir şirkette farklı etki alanlarına ait veri setleri olabilmektedir. Bu yüzden oluşturulan Tümleşik veri seti tamamen birbirinden ayırt edilebilecek durumda değildir. Bu da kurumsal firmalarda karşılaşılabilecek bir durumdur. Tümleşik veri seti bu anlamda gerçek bir durumu simüle etmek için kullanıldı. Doküman bazlı

sınıflandırma çalışmasında ayrıca bir önceki saldırı tiplerinden farklı olarak veri setine yeni saldırı vektörleri de eklenmiştir. Hem bir önceki bölüm saldırıları hem de bu bölüm içerisine dahil edilen saldırıların bütünü Tablo 4.10.'da verilmiştir. Tablo 4.10'da italik olarak verilen saldırılar bu veri seti için yeni eklenen saldırı vektörleridir.

Tablo 4.10. Dokümana üzerine uygulanan saldırı vektörleri (genişletilmiş olarak)

Saldırı Türü			Saldırı kodu
Normal veri	Saldırı yok	Saldırı yok	s0
Yapısal Saldırılar	Modifikasyon -Kelimeyi değiştirme	Dökümandaki kelimelerin sonuna harf ekleme	s1
		Tüm metindeki kelimelerin başına rasgele harf ekleme	s2
		Tüm metindeki kelimelerin ortasına ragele harf ekleme	s3
		Tüm metindeki boşlukları kaldırma	s4
		Tüm metindeki boşluklar yerine rasgele harf konulması	s5
		Tüm metindeki boşluklar yerine + konulması	s6
		Tüm metindeki kelimelerden rasgele bir harf çıkarılması	s7
		Tüm metindeki E ve e harfi yerine 1 sayısı konulması	s8
	Yer değiştirme	Paragrafların yerini değiştirme	s9
		Kelimelerin yer değiştirmesi	*s10*
	Yerine koyma	Summarizer ile özet	s11
		LSASumarizer ile özet	*s12*
		LEXMark ile özet	*s13*
		TEXRank ile özet	*s14*
		SUMBasic ile özet	*s15*
		KLSummarizer ile özet	*s16*
		ReductionSummarizer ile özet	*s17*
Karartma Saldırıları	Eş anlamlı kelime	Eş anlamlı kelime saldırıları	s18

Geliştirilen model "Tümleşik" veri setine uygulandı. Daha sonra doküman bazında çıkan metrik değerleri Tablo 4.11.'de gösterilmiştir.

Tablo 4.11. Yöntemlere göre test sonuçları

		Yöntem 1			**Yöntem 2**			**Yöntem 3**			**Yöntem 4**		
Saldırı Yok		G	KG	KO	G	KG	KO	G	KG	KO	G	KG	KO
	G	203	0	0	205	0	0	205	0	0	196	0	0
	KG	11	836	0	9	0	0	9	836	0	8	836	0
	KO	0	0	612	0	836	612	0	0	612	0	0	612
	F-skor	0.97	0.99	1.00	0.98	0.99	1.00	0.98	0.99	1.00	0.98	1.00	1.00
	F-skor(μ)	0.99			0.99			0.99			0.99		
Dokümandaki kelimelerin sonuna harf ekleme		G	KG	KO	G	KG	KO	G	KG	KO	G	KG	KO
	G	171	0	0	201	0	0	187	0	0	185	0	0
	KG	41	832	0	130	834	0	27	835	0	28	825	0
	KO	2	4	612	0	2	612	0	1	612	1	1	612
	F-skor	0.89	0.97	1.00	0.76	0.93	1.00	0.93	0.98	1.00	0.86	1.00	1.00
	F-skor(μ)	0.97			0.99			0.98			0.98		
Tüm metindeki kelimelerin başına rasgele harf ekleme		G	KG	KO	G	KG	KO	G	KG	KO	G	KG	KO
	G	167	0	0	203	0	0	181	1	0	177	0	0
	KG	39	832	0	11	836	0	31	834	0	34	833	0
	KO	8	4	612	0	0	612	2	1	612	3	3	612
	F-skor	0.88	0.97	0.99	0.97	0.99	1.00	0.91	0.98	1.00	0.91	0.98	1.00
	F-skor(μ)	0.97			0.99			0.98			0.98		
Tüm metindeki kelimelerin ortasına rastgele harf ekleme		G	KG	KO	G	KG	KO	G	KG	KO	G	KG	KO
	G	116	0	0	201	0	0	192	0	0	198	0	0
	KG	35	810	0	12	835	0	21	835	0	15	835	0
	KO	63	26	612	0	1	612	1	1	612	1	1	612
	F-skor	0.70	0.96	0.93	0.97	0.99	1.00	0.95	0.99	1.0	0.96	0.99	1.00
	F-skor(μ)	0.92			0.99			0.98			0.99		
Tüm metindeki boşlukları kaldırma		G	KG	KO	G	KG	KO	G	KG	KO	G	KG	KO
	G	187	0	0	185	1	0	184	0	0	182	0	0
	KG	16	812	0	20	805	0	18	799	0	20	806	0
	KO	11	24	612	9	30	612	12	37	612	12	30	612
	F-skor	0.93	0.98	0.97	0.97	0.99	1.00	0.92	0.97	0.96	0.92	0.97	0.97
	F-skor(μ)	0.97			0.99			0.96			0.96		
Tüm metindeki boşluklar yerine rasgele harf konulması		G	KG	KO	G	KG	KO	G	KG	KO	G	KG	KO
	G	184	0	0	197	0	0	194	0	0	198	0	0
	KG	16	812	0	17	835	0	20	834	0	15	834	0
	KO	11	24	612	0	1	612	0	2	612	1	2	612
	F-skor	0.92	0.98	0.99	0.96	0.99	1.00	0.95	0.99	1.00	0.96	0.99	1.00
	F-skor(μ)	0.98			0.99			0.99			0.99		
Tüm metindeki boşluklar yerine + konulması		G	KG	KO	G	KG	KO	G	KG	KO	G	KG	KO
	G	197	4	0	205	0	0	199	3	0	195	1	0
	KG	11	820	1	9	836	0	10	812	0	15	814	0
	KO	9	12	611	0	0	612	5	21	612	6	21	612
	F-skor	0.94	0.98	0.98	0.98	0.99	1.00	0.96	0.98	0.98	0.95	0.98	0.98
	F-skor(μ)	0.98			0.99			0.98			0.97		

Tablo 4.11. (Devamı)

Tüm metindeki kelimelerden rasgele bir harf çıkarılması		G	KG	KO	G	KG	KO	G	KG	KO	G	KG	KO
	G	158	0	0	191	0	0	176	0	0	174	0	0
	KG	48	832	0	23	835	1	37	834	1	39	834	0
	KO	8	4	612	0	1	611	1	2	611	1	2	612
	F-skor	0.85	0.97	0.99	0.94	0.99	1.00	0.90	0.98	1.00	0.90	0.98	1.00
	F-skor(μ)	0.96			0.99			0.97			0.97		
Tüm metindeki E ve e harfi yerine 1 sayısı konulması		G	KG	KO	G	KG	KO	G	KG	KO	G	KG	KO
	G	140	0	0	203	0	0	162	0	0	143	0	0
	KG	72	834	0	11	836	0	52	836	0	71	836	0
	KO	2	2	612	0	0	612	0	0	612	0	0	612
	F-skor	0.79	0.96	1.00	0.97	0.99	1.00	0.86	0.97	1.00	0.80	0.96	1.00
	F-skor(μ)	0.95			0.99			0.97			0.95		
Paragrafların yerini değiştirme		G	KG	KO	G	KG	KO	G	KG	KO	G	KG	KO
	G	203	0	0	205	0	0	205	0	0	206	0	0
	KG	11	835	0	9	836	0	9	836	0	8	836	0
	KO	0	1	612	0	0	612	0	0	612	0	0	612
	F-skor	0.97	0.99	1.00	0.98	0.99	1.00	0.98	0.99	1.00	0.98	1.00	1.00
	F-skor(μ)	0.99			0.99			0.99			0.99		
Kelimelerin yerini değiştirme		G	KG	KO	G	KG	KO	G	KG	KO	G	KG	KO
	G	202	0	0	202	0	0	205	0	0	205	0	0
	KG	12	835	0	12	836	1	9	836	0	9	836	0
	KO	0	1	612	0	0	611	0	0	612	0	0	612
	F-skor	0.97	0.99	1.00	0.97	0.99	1.00	0.98	0.99	1.00	0.98	0.99	1.00
	F-skor(μ)	0.99			0.99			0.99			0.99		
Summarizer ile özetleme		G	KG	KO	G	KG	KO	G	KG	KO	G	KG	KO
	G	194	4	0	199	6	0	197	6	0	196	7	0
	KG	11	820	1	8	818	1	8	817	1	9	816	1
	KO	9	12	612	7	12	611	9	13	611	9	13	611
	F-skor	0.94	0.98	0.98	0.95	0.98	0.98	0.94	0.98	0.98	0.94	0.98	0.98
	F-skor(μ)	0.98			0.98			0.98			0.98		
LSASummarize ile özetleme		G	KG	KO	G	KG	KO	G	KG	KO	G	KG	KO
	G	190	0	0	202	3	0	195	0	0	192	0	0
	KG	22	830	1	8	826	0	16	836	1	20	836	0
	KO	2	6	611	4	7	612	3	0	611	2	0	612
	F-skor	0.94	0.98	0.99	0.94	0.99	0.99	0.95	0.99	1.00	0.95	0.99	1.00
	F-skor(μ)	0.98			0.98			0.99			0.99		
LEXMark ile özetleme		G	KG	KO	G	KG	KO	G	KG	KO	G	KG	KO
	G	199	0	0	200	0	0	198	3	0	198	1	0
	KG	13	830	0	9	832	0	13	827	0	14	832	0
	KO	2	6	612	0	4	612	3	6	612	2	3	612
	F-skor	0.96	0.99	0.99	0.96	0.99	0.99	0.95	0.99	0.99	0.96	0.99	1.00
	F-skor(μ)	0.99			0.99			0.98			0.99		

Tablo 4.11. (Devamı)

		G	KG	KO	G	KG	KO	G	KG	KO	G	KG	KO
TEXMark ile özetleme	G	199	1	0	191	1	0	200	0	0	198	1	0
	KG	15	834	0	20	831	0	14	832	0	16	834	0
	KO	0	1	612	3	4	612	0	4	612	0	1	612
	F-skor	0.96	0.99	1.00	0.98	0.99	1.00	0.97	0.99	1.00	0.96	0.99	1.00
	F-skor(μ)	0.99			0.99			0.98			0.99		
		G	KG	KO	G	KG	KO	G	KG	KO	G	KG	KO
SUMBasic ile özetleme	G	207	1	0	206	0	0	208	1	0	206	0	0
	KG	5	827	0	4	826	0	5	826	0	5	828	0
	KO	2	0	612	4	10	612	1	9	612	3	829	612
	F-skor	0.98	1.00	1.00	0.98	0.99	0.99	0.98	0.99	0.99	0.98	0.99	0.99
	F-skor(μ)	0.99			0.99			0.99			0.99		
		G	KG	KO	G	KG	KO	G	KG	KO	G	KG	KO
KLSummarizer ile özetleme	G	195	0	0	194	1	0	190	2	0	192	1	0
	KG	17	830	1	15	820	1	21	828	1	20	829	1
	KO	2	6	611	5	15	611	3	6	611	2	6	611
	F-skor	0.95	0.99	0.99	0.95	0.98	0.98	0.94	0.98	0.99	0.94	0.98	0.99
	F-skor(μ)	0.98			0.98			0.99			0.98		
		G	KG	KO	G	KG	KO	G	KG	KO	G	KG	KO
ReductionSummarize ile özetleme	G	197	3	0	201	3	0	198	3	0	198	1	1
	KG	16	831	1	10	829	1	15	829	1	15	1	1
	KO	1	2	611	3	4	611	1	4	611	1	832	610
	F-skor	0.95	0.99	1.00	0.96	0.99	1.00	0.95	0.99	1.00	0.96	0.99	1.0
	F-skor(μ)	0.99			0.99			0.99			0,99		
		G	KG	KO	G	KG	KO	G	KG	KO	G	KG	KO
Eş anlamlı kelime	G	200	0	0	203	0	0	204	0	0	207	0	0
	KG	14	835	0	11	834	0	10	834	0	7	834	0
	KO	0	1	612	0	2	612	0	2	612	0	2	612
	F-skor	0.97	0.99	1.00	0.97	0.99	1.00	0.98	0.99	1.00	0.98	0.99	1.00
	F-skor(μ)	0.99			0.99			0.99			0.99		

Tümleşik veri seti sonuçlarına bakıldığında saldırılar karşısında kaç dokümanın doğru sınıflandırıldığı ve eğer yanlış sınıflandırılmışsa hangi sınıfa konulduğu Tablo 4.11.' de gösterilmiştir. Sınıf bazındaki f-skorları ayrı ayrı gösterilmiş olmakla beraber ortalama f-skorları da verilmiştir. Tabloda herbir test için, köşegendeki değerler doğru sınıflandırmayı göstermektedir. Bu değerler ve f-skora bakıldığında

Tablo 4.11.'de Yöntem 2'nin diğer yöntemlere göre daha başarılı olduğu kısmen söylenebilir. Yöntem 2'de yazım düzeltimi algoritması veri seti ön işleme aşamasından hemen önce yapılmaktadır. Bu bağlamda dokümanın daha belirginleştirilmesi adına yazım düzeltiminin sınıflandırmaya olumlu katkısı olduğu söylenebilir. Yapılan çalışmanın başarımını karşılaştırmak için daha önceki veri setleri bazında kullandığımız Kategori profilleri tümleşik veri setine uygulanmıştır. Ayrıca Veri sızıntısı önleme (DLP) sistemlerinde kullanılan N-gramların çeşitli saldırılar altında sınıflama başarısının düştüğü yapılan çalışmalarda belirtilmiştir. N-gramların etkisini ölçmek için [72] ile belirtilen N-gram bazlı çalışma ele alınmıştır. Bu çalışmada SGD (Stochastic Gradiend Descent) sınıflandırıcı ile 2-gram (Unigram) yönteminin başarılı olduğu belirtilmiştir. SGD sınıflandırıcı, Karar destek makineleri (SVM) ve Lojistik regresyon (LR) gibi dışbükey hata fonksiyonları altında dahi etkili ayırım yapabilen bir sınıflama yaklaşımıdır. Yüksek verinin çıkabileceği öngörülen modellerde kullanılmaktadır. SGD yönteminde gradient değerini hesaplamak yerine her iterasyonda rasgele seçilen örnekler üzerindeki gradient değerleri alınmaktadır. Bu şekile riski en aza indirmeye çalışmaktadır. Tezin bu aşamasında SGD yönteminin önerdiğimiz yöntemle karşılaştırılması uygun görülmüştür. Yapılan çalışmada Yöntem 2, Kategori profilleri ve SGD sınıflandırıcıya ait test sonuçları Tablo 4.12.'de verilmiştir.

Tablo 4.12. Yöntem 2, Kategori profilleri ve SGD test sonuçları

		Yöntem 2			**Kategori Profilleri**			**SGD**		
Saldırı Yok		G	KG	KO	G	KG	KO	G	G	KO
	G	205	0	0	149	20	0	206	0	0
	KG	9	0	0	64	810	1	4	833	0
	KO	0	836	612	1	6	611	4	3	612
	F-skor	0.98	0.99	1.00	0.78	0.95	0.99	0.98	1.00	0.99
	F-skor(μ)	0.99			0.94			0.99		
Dokümandaki kelimelerin sonuna harf ekleme		G	KG	KO	G	KG	KO	G	KG	KO
	G	201	0	0	145	0	2	1	3	0
	KG	130	834	0	9	810	8	1	39	1
	KO	0	2	612	60	26	602	212	794	611
	F-skor	0.76	0.93	1.00	0.80	0.97	0.93	0.01	0.09	0.55
	F-skor(μ)	0.99			0.93			0.25		

Tablo 4.12.(Devamı)

Tüm metindeki kelimelerin başına rasgele harf ekleme		G	KG	KO	G	KG	KO	G	KG	KO
	G	203	0	0	143	0	0	0	1	0
	KG	11	836	0	1	780	0	0	28	0
	KO	0	0	612	70	56	612	214	807	612
	F-skor	0.97	0.99	1.00	0.80	0.96	0.91	0.00	0.06	0.55
	F-skor(μ)	0.99			0.92			0.23		
Tüm metindeki kelimelerin ortasına rastgele harf ekleme		G	KG	KO	G	KG	KO	G	KG	KO
	G	201	0	0	129	10	0	0	10	0
	KG	12	835	0	10	783	1	1	28	2
	KO	0	1	612	75	43	611	213	808	610
	F-skor	0.97	0.99	1.00	0.73	0.96	0.91	0.00	0.06	0.50
	F-skor(μ)	0.99			0.91			0.234		
Tüm metindeki boşlukları kaldırma		G	KG	KO	G	KG	KO	G	KG	KO
	G	185	1	0	50	0	0	14	0	0
	KG	20	805	0	37	587	0	0	142	0
	KO	9	30	612	127	249	612	200	694	612
	F-skor	0.97	0.99	1.00	0.38	0.80	0.77	0.12	0.29	0.58
	F-skor(μ)	0.99			0.73			0.37		
Tüm metindeki boşluklar yerine rasgele harf konulması		G	KG	KO	G	KG	KO	G	KG	KO
	G	197	0	0	138	1	5	3	1	0
	KG	17	835	0	6	787	5	4	111	0
	KO	0	1	612	60	40	602	207	724	612
	F-skor	0.96	0.99	1.00	0.79	0.97	0.92	0.03	0.23	0.57
	F-skor(μ)	0.99			0.93			0.33		
Tüm metindeki boşluklar yerine + konulması		G	KG	KO	G	KG	KO	G	KG	KO
	G	205	0	0	147	18	0	0	0	0
	KG	9	836	0	60	810	1	0	20	0
	KO	0	0	612	5	8	611	214	816	612
	F-skor	0.98	0.99	1.00	0.78	0.95	0.99	0.38	0.20	0.22
	F-skor(μ)	0.99			0.94			0.22		

Tablo 4.12.(Devamı)

Tüm metindeki kelimelerden rasgele bir harf çıkarılması		G	KG	KO	G	KG	KO	G	KG	KO
	G	191	0	0	123	16	0	3	0	0
	KG	23	835	1	11	680	1	0	268	2
	KO	0	1	611	70	140	611	211	568	610
	F-skor	0.94	0.99	1.00	0.72	0.89	0.85	0.03	0.48	0.61
	F-skor(μ)	0.99			0.86			0.48		
Tüm metindeki E ve e harfi yerine 1 sayısı konulması		G	KG	KO	G	KG	KO	G	KG	KO
	G	203	0	0	107	1	0	58	0	0
	KG	11	836	0	2	788	1	25	760	2
	KO	0	0	612	95	47	611	131	76	610
	F-skor	0.97	0.99	1.00	0.69	0.97	0.90	0.43	0.94	0.85
	F-skor(μ)	0.99			0.91			0.84		
Paragrafların yerini değiştirme		G	KG	KO	G	KG	KO	G	KG	KO
	G	205	0	0	147	27	0	206	0	0
	KG	9	836	0	66	808	1	4	833	0
	KO	0	0	612	1	1	611	4	3	612
	F-skor	0.98	0.99	1.00	0.76	0.94	1.00	0.98	1.00	0.99
	F-skor(μ)	0.99			0.94			0.99		
Kelimelerin yerini değiştirme		G	KG	KO	G	KG	KO	G	KG	KO
	G	202	0	0	143	39	0	70	0	0
	KG	12	836	1	70	805	2	2	674	1
	KO	0	0	611	1	2	610	142	162	611
	F-skor	0.97	0.99	1.00	0.72	0.93	1.00	0.49	0.89	0.80
	F-skor(μ)	0.99			0.93			0.81		
Summarizer ile özetleme		G	KG	KO	G	KG	KO	G	KG	KO
	G	199	6	0	5	1	0	150	2	0
	KG	8	818	1	11	47	1	6	685	0
	KO	7	12	611	198	788	612	58	149	612
	F-skor	0.95	0.98	0.98	0.05	0.11	0.55	0.82	0.90	0.86
	F-skor(μ)	0.98			0.26			0.87		

Tablo 4.12.(Devamı)

LSASummarize ile özetleme		G	KG	KO	G	KG	KO	G	KG	KO
	G	202	3	0	9	9	0	166	0	0
	KG	8	826	0	19	300	1	5	785	0
	KO	4	7	612	186	527	611	43	51	612
	F-skor	0.94	0.99	0.99	0.08	0.52	0.63	0.87	0.97	0.93
	F-skor(μ)	0.98			0.50			0.94		
LEXMark ile özetleme		G	KG	KO	G	KG	KO	G	KG	KO
	G	200	0	0	12	13	0	161	1	0
	KG	9	832	0	12	165	1	6	733	0
	KO	0	4	612	190	658	611	47	102	612
	F-skor	0.96	0.99	0.99	0.10	0.33	0.59	0.86	0.93	0.89
	F-skor(μ)	0.99			0.39			0.91		
TEXMark ile özetleme		G	KG	KO	G	KG	KO	G	KG	KO
	G	191	1	0	18	28	0	185	0	0
	KG	20	831	0	28	300	1	6	773	0
	KO	3	4	612	168	508	611	23	63	612
	F-skor	0.98	0.99	1.00	0.14	0.52	0.64	0.93	0.96	0.93
	F-skor(μ)	0.99			0.51			0.95		
SUMBasic ile özetleme		G	KG	KO	G	KG	KO	G	KG	KO
	G	206	0	0	5	6	0	143	1	0
	KG	4	826	0	6	136	1	2	684	0
	KO	4	10	612	203	694	611	69	151	612
	F-skor	0.98	0.99	0.99	0.04	0.28	0.58	0.80	0.90	0.85
	F-skor(μ)	0.99			0.40			0.87		
KLSummarizer ile özetleme		G	KG	KO	G	KG	KO	G	KG	KO
	G	194	1	0	4	10	0	155	0	0
	KG	15	820	1	18	157	1	6	742	0
	KO	5	15	611	192	669	611	53	94	612
	F-skor	0.95	0.98	0.98	0.04	0.31	1.00	0.84	0.94	0.89
	F-skor(μ)	0.98			0.38			0.91		

Tablo 4.12.(Devamı)

ReductionSummarize ile özetleme		G	KG	KO	G	KG	KO	G	KG	KO
	G	201	3	0	13	18	0	173	1	0
	KG	10	829	1	27	269	1	5	723	0
	KO	3	4	611	174	549	611	36	112	612
	F-skor	0.96	0.99	1.00	0.11	0.47	0.63	0.89	0.92	0.89
	F-skor(μ)	0.99			0.48			0.91		
Eş anlamlı kelime		G	KG	KO	G	KG	KO	G	KG	KO
	G	203	0	0	130	0	0	193	1	0
	KG	11	834	0	2	776	1	3	817	1
	KO	0	2	612	82	60	611	18	18	612
	F-skor	0.97	0.99	1.00	0.76	0.96	0.90	0.95	0.99	0.97
	F-skor(μ)	0.99			0.91			0.98		

Kategori profilleri gizli dokümanlarda çok başarılı olamamıştır. 214 Kurumsal Gizli (G) dokümandan sadece 149 tanesini doğru tanıyabilmiştir. Önerilen modelde Yöntem 2'ye göre en kötü durumda 185 adet dokümanı doğru olarak tanınabilmiştir. Ayrıca f-skorlarına bakıldığında önerilen modelin değeri daha iyidir. Kategori profilleri modifikasyon saldırılarına karşı dayanıksız olduğu görülmüştür. Ayrıca yerine koyma saldırılarında özetleme algoritmasının çalışma şekline bağlı olarak normal sınıflama başarısını her zaman koruyamadığı görülmüştür. Yer değiştirme saldırılarında paragrafların yerinin değişiminde normal başarımını sağlamasına rağmen kelime yer değişiminde bu başarıyı sağlayamadığı görüldü. Karartma saldırılarında amaç dokümanı farklı biçimde göstermeye çalışmaktır. Bu nedenle verilen dokümandaki kelimeler ve harfler değişmektedir. Bu nedenle Kategori profilleri dokümanı sınıflayacak kadar yeterli bilgiye sahip olamaktadır. Oysaki önerdiğimiz yöntemle değişen bu durum LSA algoritması ile korunmaya çalışmaktadır. Bu yüzden modelimiz Karartma saldırılarında, Kategori profillerinden daha başarılı olmaktadır. Önerilen yöntemi, SGD yöntemi ile karşılaştırdığımızda saldırı olmadığı durumlarda ve yer değiştirme saldırılarından paragraf değiştirme saldırısında önerdiğimiz yöntemle neredeyse eşit başarım göstermiştir. Bu durum SGD sınıflandırıcıda, paragraf yer değiştirme saldırısında Unigramların yerlerinin değişmemesinden

kaynaklanmıştır. Çünkü paragraflar yer değişse bile ard arda gelen kelimelerin yeri değişmemiştir. Fakat kelimelerin yerlerinin değişmesi ile bu durumun bozulduğu görülmektedir. Fakat önerdiğimiz yöntemde saldırı yapılmayan model ile yer değiştirme saldırısından sonra elde edilen sonuçların korunduğu görülmüştür. Ayrıca Yerine koyma saldırılarında özetleme algoritmalarının çalışma sisteminin farklılıklarından dolayı yine başarının düştüğü görülmektedir. Özellikle modifikasyon saldırılarında SGD sınıflandırıcının çıkardığı özellikler sınıflandırma yapmasına yetmemektedir. Bu yüzden sınıflandırıcı hatalı sonuçlar üretmekte ve dolayısıyla sistemin başarısı düşmüştür. Fakat önerdiğimiz yöntemde kullandığımız yazım düzeltimi ve karakter n-gramlar sınıflandırıcı için yeteri kadar özellik sağladığından dolayı sınıflama başarısı azalsa da yine SGD sınıflandırıcıdan çok daha iyi sonuçlar çıkarmıştır.

Günümüzde derin öğrenme yöntemleri birçok alanda kullanılmaktadır. Başarısı yadsınamayan bu yöntemin DLP sistemlerinde doküman sınıflamada kullanılmasının gerçekleştirilen saldırılar altında başarısı için [73] çalışmasında ele alınmıştır. Bu çalışmada konvensiyonel derin öğrenme (CNN) yöntemi ile doküman sınıflandırma yapılmaktadır. CNN ileri beslemeli bir derin öğrenme yöntemidir. Genel olarak bilgisayar görmesi alanında çokça kullanılmasına rağmen son zamanlarda NLP alanında da kullanılmaktadır. Çalışma şekli olarak matrisler aracılığıyla desenler çıkararak, bulduğu desenlerden öğrenerek ilerlemektedir. [73]'teki çalışma Tümleşik veri setine uygulanarak hassaslık ölçüsü (accuracy) değeri bulunarak Yöntem 2 modeliyle karşılaştırıldı (Tablo 4.13.).

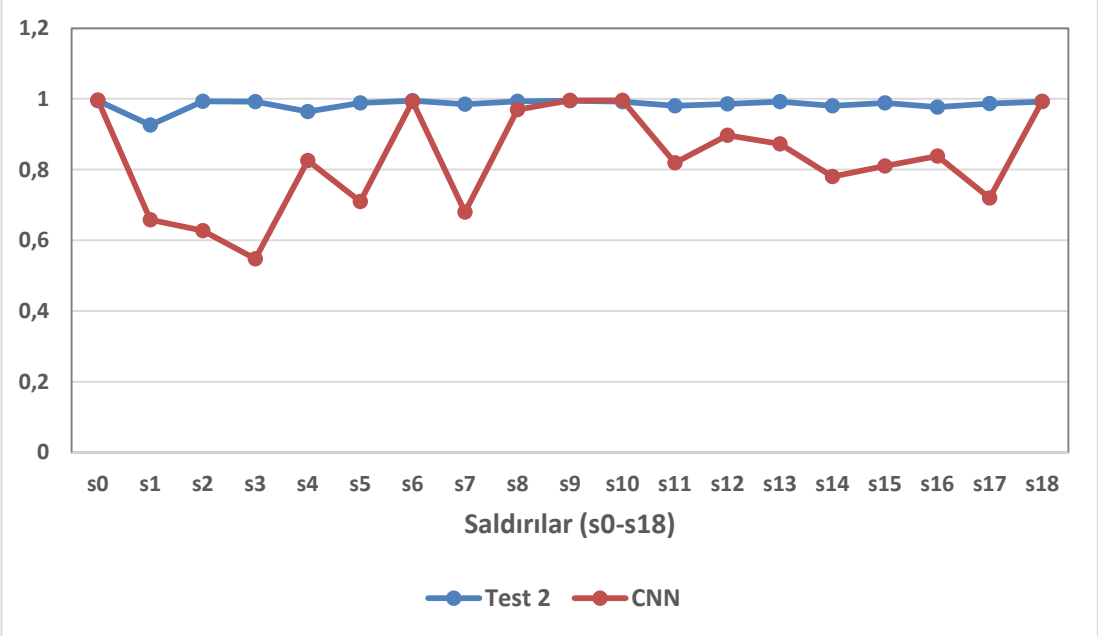
Tablo 4.13. Tümleşik doküman üzerinde Test 2, CNN accuracy değerlerinin karşılaştırılması.

Saldırı Türü			Test 2	CNN
Normal veri	Saldırı yok	Saldırı yok	0.995	0.997
Yapısal Saldırılar	Modifikaston -Kelimeyi değiştirme	Dökümandaki kelimelerin sonuna harf ekleme	0.926	0.658
		Tüm metindeki kelimelerin başına rasgele harf ekleme	0.993	0.627
		Tüm metindeki kelimelerin ortasına ragele harf ekleme	0.992	0.548
		Tüm metindeki boşlukları kaldırma	0.964	0.826
		Tüm metindeki boşluklar yerine rasgele harf konulması	0.989	0.71
		Tüm metindeki boşluklar yerine + konulması	0.995	0.993
		Tüm metindeki kelimelerden rasgele bir harf çıkarılması	0.985	0.68

Tablo 4.13.(Devamı)

		Tüm metindeki E ve e harfi yerine 1 sayısı konuldu	0.993	0.97
	Yer değiştirme	Cümlenin yerini değiştirme	0.995	0.996
		Kelimelerin yer değiştirmesi	0.992	0.996
	Yerine koyma	Summarizer ile özet	0.98	0.819
		LSASumarizer ile özet	0.986	0.897
		LEXMark ile özet	0.992	0.873
		TEXRank ile özet	0.98	0.78
		SUMBasic ile özet	0.989	0.81
		KLSummarizer ile özet	0.977	0.838
		ReductionSummarizer ile özet	0.987	0.72
Karartma Saldırıları	Eş anlamlı kelime	Eş anlamlı kelime saldırıları	0.992	0.993

Tablo 4.13.'te elde edilen sonuçlara ait grafik Şekil 4.6.'da gösterilmiştir.



Şekil 4.6. Test 2 modeli ile CNN accuracy değerleri karşılaştırma grafiği

CNN sınıflandırma başarısı yüksek olmasına rağmen önerdiğimiz modeldeki Yöntem 2'ye göre başarısı daha azdır. Bunun başlıca sebebi öğrenmenin saldırı çeşitlerine göre yapılmamasıdır. Bu da önerdiğimiz yöntemin saldırı altında dahi dokümanı doğru sınıflandırması açısından kayda değer eğitildiğini göstermektedir.

BÖLÜM 5. SONUÇ VE ÖNERİLER

Bu çalışmada zararlı yazılım kaynaklı saldırılarda dahi dokümanı doğru sınıflayacak bir algoritma çözümü sunulmuştur. Yapılan saldırı türleri yeniden kategorize edilerek saldırı türleri genişletilmiştir. Ayrıca saldırı türlerinden yapısal saldırılar (modifikasyon saldırıları, yerdeğiştirme ve yârine koyma) ile karartma saldırıları (eş anlamlı kelime) üzerinde tez sınırlandırılmıştır. Sınırlandırılan bu saldırı türlerinden birkaçına ait zararlı yazılım gerçeklenmiş ve bu saldırı yazılımı ile dokümanlara saldırı yapılarak test dokümanları oluşturulmuştur. Ayrıca saldırı yapılmayan bir test dokümanı da algoritmanın saldırı olmaması durumundaki başarısını ölçmek için kullanılmıştır. Istatistiksel ve doğal dil işleme yöntemlerinin kullanıldığı çalışmada özelliklerin çıkarılması kelime n-gramlar ile sağlanmaktadır Fakat ele alınan saldırılardan yapısal saldırı alt kategorisindeki modifikasyon saldırılarında, içeriğin değiştirilmesi söz konusu olduğunda dokümana ait kelime n-gramlar sağlıklı olarak çıkarılamamaktadır. Bu sebeple, bu çalışmada, N-gram kullanımının başarılı olabilmesi için n-gram'ların çıkarılma aşamasında içeriğe yönelik saldırılar dikkate alınarak önişlemler yapılmaktadır. Bu nedenle dokümandaki yazım hatalarının düzeltilmesi farklı aşamalarda uygulanarak sınıflamaya katkısı ölçülmüştür. Ölçümlerde yazım düzeltiminin modifikasyon saldırılarında etkili olduğu görülmüştür. Yazım düzeltimi uygulandığı halde hala N-gram çıkarımı yapılamıyorsa karakter n-gramlar kullanılması etkili olmaktadır.

Doküman özeti çıkarma şeklindeki yapısal saldırılarda ise içerik azaldığı için dokümanı sınıflandırmak için yeterli veri elde edilemeyebilmektedir. Ayrıca dokümandaki kelime veya cümlelerin yer değiştirilmesi ile n-gramların özellik vektörleri değişmekte ve sınıflandırıcı atlatılabilmektedir. Bu durumda k-skip-n-gramların kullanılması faydalı olmaktadır. Eş anlamlı kelimelerin kullanıldığı karartma saldırılarında ise kelimelerin tespitinde Gizli Anlam Analizi (Latent

Semantic Analysis – LSA) yöntemi kullanılması ile dokümana ait örtük bilginin açığa çıkarılması sağlanmaktadır. Bunun için çalışmamızda sunulan algoritmanın Özellik Çıkarımı aşamasında n-gram, skip-gram, LSA ve karakter-gram özellikleri çıkarılmakta ve sınıflandırıcıda bir arada kullanılmaktadır.

İçerik tabanlı saldırı vektörlerinin çeşitliliği, saldırı sonrasında oluşan doküman ile orijinal doküman arasındaki farkın çok belirgin olmaması gibi sebeplerden dolayı saldırının tespit edilmesi zor olmakla beraber, birden fazla saldırı metodunun uygulanması söz konusu olduğunda saldırı vektörünün tespiti daha da güçleşmektedir. Bu durumda tüm saldırı vektörleri için tekil çözümünün bulunması daha uygun olmaktadır. Sunulan çözümde bahsedilen yöntemler bir arada kullanılarak farklı saldırı vektörlerine karşı dayanıklı bir özellik çıkarımı sağlanmıştır. Bunun yanında farklı durumlarda sınıflandırma başarımını arttırmak için sınıflandırıcıların oylamalı olarak kullanılması tercih edilmiştir. Yazım düzeltimi yönteminin saldırılara karşı başarılı olduğu ortaya konmakla beraber, algoritmanın farklı aşamalarında uygulanmasının etkisini gösterebilmek amacıyla çalışmanın test bölümünde 4 farklı şekilde uygulanarak test edilmiştir. Yapılan testlerde Yöntem 2'nin daha başarılı olduğu görülmüştür. Tezin birinci aşamasında veri setlerine göre yapılan testlerde başarımın çok kararlı olmadığı görülmüştür. Çünkü bir veri setinde etkili olan algoritmamız aynı saldırı için diğer veri setinde beklenildiği gibi sonuç üretememiştir. Bu sonucun sebebi veri setlerindeki özelliklerin sınıflandırıcı için yeterli bilgi bulamaması olabilir. Ayrıca bir kategoriye ait yeterli doküman olduğu halde diğer kategori için yeterli doküman olmaması söz konusu olabilir. Örneğin Dyncorp veri setinde Gizli (G) kategorisinde sadece 1 doküman eğitim aşamasında kullanılmıştır. Bu duruma bağlı olarak bu testte istenilen sonuçların çıkmadığı görülmüştür. Ayrıca veri setlerine göre yaptığımız çalışmada DLP alanında kullanılan kategori profillerini Mormon veri setine uygulanarak Yöntem 2 ile performansı karşılaştırılmıştır. Bu karşılaştırmada Yöntem 2'nin daha başarılı olduğu görülmüştür. İkinci aşamada ise gerçek bir kurumda olabilecek farklı domainlere ait dokümanların kaçırılmasını simüle etmek için mevcut veri setleri olduğu gibi korunarak "Tümleşik" isimli yeni bir veri seti oluşturulmuştur. Bu şekilde, veri setlerinden doküman tabanlı bir sisteme geçiş yapılmıştır. Bu veri seti ile yaptığımız test sonuçlarında öğrenmenin daha iyi yapıldığı ve dolayısıyla

sonuçların istenen düzeyde çıktığı görülmüştür. Ayrıca kategori profilleri ve SGD sınıflandırıcı dokümanlar üzerinde çalıştırılarak herbir kategoriye ait metrikler bulunmuştur. Bu metriklerde yine önerilen modelin daha başarılı olduğu görüldü. Günümüzde çokça kullanılan derin öğrenme yaklaşımı kullanılarak elde edilen hassaslık değerleri Yöntem 2 ile karşılaştırılarak, önerilen modelin başarılı olduğu görüldü. Modelde herhangi bir saldırı olmadığında f-skor değeri %99.2 olarak bulunmuştur. Ayrıca saldırılara altında bu durum her zaman korunamamakla beraber f-skor değeri en küçük Yöntem 2'ye göre %97.7 olarak ölçülmüştür.

Modelde herhangi bir indirgeme yöntemi yapılmadığı gözönüne alındığında burada yapılacak özellik indirgeme yöntemi hem algoritmanın daha hızlı çalışmasını hem de daha iyi sınıflamasını sağlayabilir. Ayrıca kullanılan yöntemlerdeki çoğu parameterler standart olarak alınmıştır. Bu yüzden algoritmanın yüksek başarım sağlaması için özellik indirgeme ve seçilim algoritmalarının yanında bu parametrelerin uygun şekilde seçilmesi gibi çalışmalar da yapılabilir. Ayrıca içerik tabanlı saldırı vektörlerinin çeşitliliği, saldırı sonrasında oluşan doküman ile orijinal doküman arasındaki farkın çok belirgin olmaması gibi sebeplerden dolayı saldırının tespit edilmesi zor olmakla beraber, birden fazla saldırı metodunun uygulanması söz konusu olduğunda saldırı vektörünün tespiti daha da güçleşmektedir. Bu durumda tüm saldırı vektörleri için tekil çözümlerin bulunması daha uygun olabileceği düşünülmektedir. Eğer yapılan saldırının tespiti aşamasında başarılı olunabilirse ele aldığımız algoritma çözümlerinin tümünün uygulanması yerine ilgili algoritma veya metotun uygulanması hem hız hem de başarı açısından iyi olacaktır. Bu amaçla tezin sonraki aşamalarında yapılan saldırı tespiti ile sadece ilgili algoritmanın çalıştırılması gibi üzerinde çalışmalar da yapılabilir. Bunun yanında yukarıda ifade edildiği gibi algoritmayı hızlandırma adına sınıflandırıcıya ait parametre seçimi, özellik seçilim ve özellik indirgeme yöntemlerinin uygun olanının kullanımına ait çalışmalar gelecek çalışamalar kategorisinde sayılabilir.

KAYNAKÇA

[1] http://snowballstem.org/algorithms/, Erişim Tarihi: 24.11.2017.

[2] https://en.wikipedia.org/wiki/DBpedia, Erişim Tarihi: 10.05.2016.

[3] Altszyler, Edgar. ve Sigman, Mariano. ve Ribeiro, Sidarta. ve Slezak, Diego Fernández.," Comparative study of LSA vs Word2vec embeddings in small corpora: a case study in dreams database", 2016.

[4] Tariq Mustafa,Malicious Data Leak Prevention and Purposeful Evasion Attacks: An approach to Advanced Persistent Threat (APT) management, 2013.

[5] https://drive.google.com/drive/u/0/folders/ 0Bz8a_Dbh9Qhbfll6bVpmNUtUcFdjYmF 2SEpmZUZUcVNiMUw1TWN6RDV3a0JHT3kxLVhVR2M, Erişim Tarihi: 10.05.2016.

[6] Canbay, Yavuz. Ve Yazici, Hatice. ve Sagiroglu, Seref., "A Turkish language based data leakage prevention system", 2017.

[7] Whitelaw Casey Hutchinson Ben ,Chung Grace Y , Ellis Gerard, Using theWeb for Language Independent Spellchecking and Autocorrection, 2009.

[8] Bassil, Youssef., Parallel Spell-Checking Algorithm Based on Yahoo! N-Grams Dataset,2012.

[9] http://blog.faroo.com/2012/06/07/improved-edit-distance-based-spelling-correction/, Erişim Tarihi : 10.03.2018.

[10] Chalothorn, T. ve Ellman, J., TJP: Using Twitter to Analyze the Polarity of Contexts, 2013.

[11] Nakov, Preslav ve ark., System description for SemEval 2013 Task 2 Sentiment Analysis in Twitter, 2013.

[12] Biricik Göksel,Metin Sınıflama İçin Yeni Bir Özellik Çıkarım Yöntemi ,2011.

[13] Jivani, Anjali G., Comparative Study of Stemming Algorithms, 2011.

[14] M.Ramya , J.Alwin Pinakas, Different Type of Feature Selection for Text Classification, 2014.

[15] Vineet John,A Survey of Neural Network Techniques for Feature Extraction from Text,2017

[16] *Amasyalı, M. Fatih.* ve ark., Türkçe Metinlerin Sınıflandırılmasında Metin Temsil Yöntemlerinin Performans Karşılaştırılması

[17] Cavnar, William B Trenkle, John M Mi, Ann Arbor , N-Gram-Based Text Categorization,1994.

[18] Guthrie, D Allison, Ben Liu, Wei, A Closer Look at Skip-gram Modelling, 2006.

[19] Mikolov, Tomas Chen, Kai Corrado, Greg Dean, Jeffrey Efficient Estimation of Word Representations in Vector Space, 2013.

[20] Riezler, Stefan Goldberg, Yoav ,2016 Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning.

[21] Giridhar, N S. ve ark., A Prospective Study of Stemming Algorithms for Web Text Mining, 2011.

[22] https://en.wikipedia.org/wiki/Latent_semantic_analysis, Erişim Tarihi: 21.10.2017.

[23] http://www1.se.cuhk.edu.hk/~seem5680/lecture/LSI-Eg.pdf, Erişim Tarihi: 20.10.2017.

[24] https://medium.com/t%C3%BCrkiye/makine-%C3%B6%C4%9Frenmesi-nedir-20dee450b56e, Erişim Tarihi: 09.09.2018.

[25] Kumar, Ravi. ve ark., A survey on opinion mining and sentiment analysis: Tasks, approaches and applications, 2015.

[26] Hui Yang, Alistair Willis, Anne de Roeck, and Bashar Nuseibeh1, A hybrid model for automatic emotion recognition in suicide notes, 2012.

[27] Çatal, Çağatay. ve Nangir, Mehmet., A Sentiment Classification Model Based On Multiple Classifier, 2016.

[28] Manzoor, Muhammad Asif. ve Morgan, Yasser., "Realtime Support Vector Machine based Network Instrusion Detection system using Apache Storm", 2016.

[29] Yi, Zhehan. ve Etemadi Amir H.,"A Novel Detection Algorithm for Line-to-Line Faults in Photovoltaic (PV) Arrays Based on Support Vector Machine (SVM) ", 2016.

[30] http://www.academia.edu/25049513/Random_Forest_Rastgele_Orman_Algoritmas%C4%B1, Erişim Tarihi: 20.04.2017.

[31] Principles Of Artificial Neural Networks (3rd Edition) (Graupe, Daniel), 2013.

[32] Sriparna Saha , Asif Ekbal , Combining multiple classifiers using vote based classifier ensemble technique for named entity recognition ,2013.

[33] http://scikit-learn.org/stable/modules/ensemble.html#voting-classifier, Erişim Tarihi: 17.04..2017.

[34] Beghdad, Rechid., Critical study of neural networks in detecting intrusions, 2008.

[35] https://eezgiozgenn.wordpress.com/zararli-yazilim-nedir/, Erişim Tarihi: 01.04.2018.

[36] https://www.rdocumentation.org/packages/maps/versions/3.2.0/topics/map.text, Erişim Tarihi : 01.04.2018.

[37] Hart, Michael. ve ark, Text Classification for Data Loss Prevention, 2011.

[38] Radwan ve Yousef, Data Leakage/Loss Prevention Systems (DLP), 2014.

[39] Detecting data semantic: A data lekage prevention approach, Sultan Alneyadi ve ark. ,2015.

[40] Bruna Martins ve Mario J. Silva, Spelling Correction for Search Engine Queries, 2014.

[41] Farag Ahmed ve ark., Revised N-Gram based Automatic Spelling Correction Tool to Improve Retrieval Effectiveness.

[42] https://www.cs.csustan.edu/~mmartin/LDS/AHPCRC081910.pdf, Erişim Tarihi: 04.05.2019.

[43] https://www.e-adys.com/makine_ogrenmesi/hangisini-secmeliyim-supervised-ve-unsupervised-learning/, Erişim Tarihi:12.02.2016.

[44] Breiman L., (2002), Manual on setting up, using, and understanding random forests V3.1, http://oz.berkeley.edu/users/breiman/Using_random_forests_V3.1.pdf, Erişim Tarihi: 20.09.2011.

[45] http://docs.trendmicro.com/en-us/enterprise/officescan-110-sp1-server/using-data-loss-prev/data-loss-prevention1/predefined-dlp-templ.aspx, Erişim Tarihi: 13.01.2018.

[46] http://bmb.cu.edu.tr/uorhan/DersNotu/Ders01.pdf, Erişim Tarihi: 09.09.2018.

[47] https://showmethenotes.blogspot.com.tr/2017/10/mkina-ogrenmesi-5.html, erişim tarihi: 02.02.2017.

[48] Ho TK (1998). "The Random Subspace Method for Constructing Decision Forests" (PDF). *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 20 (8):832–844. doi:10.1109/34.709601.

[49] Damjan Krstajic,Ljubomir J Buturovic, David E Leahy, Simon Thomas,Cross-validation pitfalls when selecting and assessing regression and classification models, 2014

[50] Breiman, L. (2001), Random Forests, Machine Learning 45(1), 5-32.

[51] Horning N., (2010), RandomForests : An algorithm for image classifcation and generation of continuous felds data sets, International Conference on Geoinformatics for Spatial Infrastructure Development in Earth and Allied Sciences (GISIDEAS) 2010, Hanoi,Vietnam.

[52] Vapnik, V.N., 1995, The Nature of Statistical Learning Theory, 2. Baskı, Springer-Verlag, New York.

[53] Vapnik, V.N., 2000, The Nature of Statistical Learning Theory, 2. Bask谋 , Springer-Verlag, New York.

[54] Osuna, E.E., Freund, R., Girosi, F., 1997, Support Vector Machines: Training and Applications, A.I. Memo No. 1602, C.B.C.L. Paper No. 144, Massachusetts Institute of Technology and Artificial Intelligence Laboratory, Massachusetts.

[55] Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition, data mining and knowledge discovery. Kluwer Academic Publishers, 2 (2), 121-167.

[56] Busuttil, S. (2003). Support vector machines. In Proceedings of the Computer Science Annual Research Workshop, Villa Bighi, Kalkara, University of Malta.

[57] Suykens, J. A. K. (2002). Least squares support vector machines. River Edge, NJ : World Scientific, xiv, 294 s.

[58] Çakar, Ö. (2007). Fonksiyonel analize giriş I. A.Ü. Fen Fakültesi Döner Sermaye İşletmesi Yayınları, no:13, (Erwin KREYSZİG'den Uyarlama).

[59] Şen, Z. (2004).Yapay Sinir Ağları İlkeleri. İstanbul: Su Vakfı Yayınları.

[60] Öztemel, E. (2006) Yapay Sinir Ağları, 2. Baskı, İstanbul, Papatya Yayıncılık.

[61] Minsky M, Papert S 1969 Perceptrons. MIT Press, Cambridge, MA.

[62] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html, Erişim Tarihi: 24.04.2018.

[63] N Japkowicz, M Shah. 2011. Evaluating learning algorithms: a classification perspective Cambridge University Press.

[64] http://bilgisayarkavramlari.sadievrenseker.com/2010/09/30/f1-degerlendirme-f1-scoring/, Erişim Tarihi: 23.04.2018.

[65] M. P., 2002, Integrating Real Medical Studies into Teaching Biostatistics,Conference Proceedings, 2002, Greece.

[66] Kazemian, M., Moshiri, B., Palade, V. ve Nikbakht, H., 2010, Using classifier fusion techniques for protein secondary structure prediction, International Journal of Computational Intelligence in Bioinformatics and Systems Biology, 1 (4), 418–434.

[67] Hardeniya Nitin ve ark., Natural Language Processing: Python and NLTK. Packtpub, 2016.

[68] Kulmizev, Artur. ve Blankers, Bo. ve Bjerva, Johannes. ve Nissim, Malvina. ve Van Noord, Gertjan. ve Plank, Barbara. ve Wieling, Martijn., "The Power of Character N-grams in Native Language Identification", 2018.

[69] Menahem, Eitan Shabtai, Asaf Rokach, Lior Elovici, Yuval,2009 Improving malware detection by applying multi-inducer ensemble

[70] Murat, Topaloğlu., Özel Anlamlı Ifade Içeren Verilerde Sızıntı Önleme Için Bir Mimari Tasarım ve Gerçekleştirilmesi, 2012.

[71] http://blog.alejandronolla.com/2013/05/20/n-gram-based-text-categorization-categorizing-text-with-python/ Erişim Tarihi: 30.4.2019.

[72] Tripathy, Abinash ve ark. Classification of sentiment reviews using n-gram machine learning approach, 2016.

[73] https://medium.com/jatana/report-on-text-classification-using-cnn-rnn-han-f0e887214d5f, Erişim Tarihi: 6.04.2019.

ÖZGEÇMİŞ

Yahya KESENEK, 30.08.1983 de Şanlıurfa'da doğru. İlk, orta ve lise eğitimini Siverek'te tamamladı. 2006 yılında Kocaeli Üniversitesi İlköğretim Matematik Öğretmenliğinden mezun oldu. 2014 yılında Kütahya Dumlupınar Üniversitesi Bilgisayar Mühendisliği Bölümünü bitirdi. 2014 yılında Sakarya Üniversitesi, Siber Güvenlik Bölümüne girdi. 2016 yılında Kivy ile crossplatform uygulamalar geliştirme kitabını yayımladı.