TÜRKİYE CUMHURİYETİ
YILDIZ TEKNİK ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

TÜRKÇE METİNLERDE SİBER ZORBALIK TESPİTİ

20011076 — Beyza FINDIK
20011101 — Sevda KARAHAN

BILGISAYAR PROJESİ

Danışman
Prof.Dr. Banu Diri

Haziran, 2024

TEŞEKKÜR

Proje çalışmamız süresince değerli yardım ve yönlendirmelerinden ötürü proje danışmanımız Prof. Dr. Banu Diri'ye teşekkür ederiz. Süreç içerisinde bize yardımcı olup, proje geliştirme safhalarımızda değerli yorumlarını bizlerle paylaştığı için kendisine müteşekkiriz. Yaptığımız projemiz Bilgisayar Mühendisliği sürecimiz için değerli olup, bize katkı sağlamıştır. Bundan ötürü bize hem kendimizi geliştirme imkanı sunan, hem de gelişimimiz için destek veren başta Prof. Dr. Banu Diri olmak üzere, bölümümüzün tüm değerli öğretim üyelerine teşekkürü bir borç biliriz.

Beyza FINDIK
Sevda KARAHAN

İÇİNDEKİLER

KISALTMA LİSTESİ

BERT	Bidirectional Encoder Representations from Transformers
GB	GigaByte
GPU	Graphics Processing Unit
NLP	Natural Language Processing
RNN	Recurrent Neural Network
PTMs	Pre-trained Models
CNN	Convolutional Neural Network
SGD	Stochastic Gradient Descent
SVC	Support Vector Classifier
GRU	Gated Recurrent Unit
LSTM	Long Short-Term Memory
BiLSTM	Bidirectional Long Short-Term Memory
NaN	Not a Number
TF-IDF	Term Frequency-Inverse Document Frequency
RBF	Radial Basis Function
KNN	K-Nearest Neighbors
GloVe	Global Vectors for Word Representation

ŞEKİL LİSTESİ

ÖZET

TÜRKÇE METİNLERDE SİBER ZORBALIK TESPİTİ

Beyza FINDIK
Sevda KARAHAN

Bilgisayar Mühendisliği Bölümü
Bilgisayar Projesi

Danışman: Prof.Dr. Banu Diri

İnternet kullanımının yaygınlaşması ve sosyal medya platformlarının popülaritesinin artması siber zorbalık eylemlerinin hızla yayılmasına neden olmuştur. Dünya genelinde siber zorbalığa maruz kalan kişilerin sayısı her geçen gün artmakta ve bu da mağdurlar üzerinde büyük etkiler yaratmaktadır. Bu eylemin tespit edilmesi, yeni mağdurların ortaya çıkmaması ve mevcut mağdurların daha fazla bu eyleme maruz kalmaması açısından büyük önem taşımaktadır. Bu noktada yapay zeka alt disiplinleri olan makine öğrenmesi ve derin öğrenme ile siber zorbalık tespiti ön plana çıkmaktadır. Bu anlamda literatürde birçok çalışmanın gerçekleştirildiği görülmüş ancak Türkçe metinlerde yapılan çalışma sayısının arttırılması gerektiği tespit edilmiştir. Ayrıca bu alanda yapılan Türkçe çalışmalarda sadece makine öğrenmesi yöntemlerinin kullanılmasının yeterli olmadığı, derin öğrenme modellerinin de kullanılması gerektiği gözlemlenmiştir. Böylece iki yöntemin de kullanıldığı çalışmaların sayısının arttırılarak literatüre katkıda bulunulması hedeflenmektedir. Bu doğrultuda Kaggle adlı paylaşım sitesinden elde edilmiş 11.000'den fazla Türkçe tivitten oluşan bir sosyal medya paylaşımı veri seti üzerinde doğal dil işleme yöntemleri kullanılarak siber zorbalık tespiti gerçekleştirilmiştir. Çalışmada bu veri seti üzerinde öncelikle makine öğrenmesi modellerinden Logistic Regression, Support Vector Classifier (SVC), Stochastic Gradient Descent (SGD) Classifier, Multinomial Naive Bayes, KNeighbors Classifier, Random Forest Classifier, Decision Tree Classifier, Ada Boost Classifier, Bagging Classifier algoritmaları karşılaştırmalı olarak kullanılmıştır. Daha sonra derin öğrenme modelleri olarak CNN, RNN, LSTM, GRU, BiLSTM ve BERT farklı gömme yöntemleri ile kullanılmış ve karşılaştırmıştır.

Burada kullanılan gömme modelleri Keras Embedding, Glove ve FastText modelleridir. En nihayetinde derin öğrenme ve makine öğrenmesi modelleri birbiri ile kıyaslanarak siber zorbalık tespitinde en başarılı model tespit edilmiştir.

Anahtar Kelimeler: Doğal dil işleme, Derin öğrenme, Makine öğrenmesi, Siber zorbalık, Türkçe veri

ABSTRACT

DETECTION OF CYBERBULLYING IN TURKISH TEXTS

Beyza FINDIK
Sevda KARAHAN

Department of Computer Engineering
Computer Project

Advisor: Prof.Dr. Banu Diri

The widespread use of the Internet and the increasing popularity of social media platforms have led to the rapid spread of cyberbullying. The number of people who are exposed to cyberbullying worldwide is increasing day by day and this has a great impact on the victims. Detecting this act is of great importance in order to prevent the emergence of new victims and to prevent existing victims from being exposed to this act further. At this point, cyberbullying detection with machine learning and deep learning, which are sub-disciplines of artificial intelligence, comes to the fore. In this sense, it has been observed that many studies have been conducted in the literature, but it has been determined that the number of studies conducted in Turkish texts should be increased. In addition, it has been observed that it is not sufficient to use only machine learning methods in Turkish studies in this field, deep learning models should also be used. Thus, it is aimed to contribute to the literature by increasing the number of studies using both methods. In this direction, cyberbullying detection was performed using natural language processing methods on a social media sharing dataset consisting of more than 11,000 Turkish tweets obtained from the sharing site called Kaggle. Logistic Regression, Support Vector Classifier (SVC), Stochastic Gradient Descent (SGD) Classifier, Multinomial Naive Bayes, KNeighbors Classifier, Random Forest Classifier, Decision Tree Classifier, Ada Boost Classifier, Bagging Classifier algorithms were used comparatively on this dataset. Then CNN, RNN, LSTM, GRU, BiLSTM and BERT as deep learning models were used and compared with different embedding models. The embedding models used here are Keras Embedding, Glove and FastText models. Finally, deep learning and machine

learning models were compared with each other and the most successful model was determined in cyberbullying detection.

Keywords: Natural language processing, Deep learning, Machine learning, Cyberbullying, Turkish data

1
Giriş

Teknolojinin hızlı gelişimiyle birlikte, sosyal medya platformları insanların iletişimini kolaylaştırmış ve bilgi alışverişini arttırmıştır. Ancak, bu gelişmelerin yanında siber zorbalık gibi zararlı davranışların da yayılmasına zemin hazırlamıştır. Siber zorbalık, internet ve iletişim teknolojileri aracılığıyla gerçekleştirilen sürekli ve kasıtlı davranışlar olarak tanımlanmaktadır. Özellikle sosyal medya platformları, bu tür davranışların hızla yayılmasında etkili bir rol oynamaktadır. Herkesin kolaylıkla erişebildiği sosyal medya ortamları, denetim eksikliği ve görece anonimlik gibi faktörler nedeniyle dijital zorbalıkların artmasına katkı sağlamaktadır.

Siber zorbalık, mağdurlar üzerinde ciddi psikolojik etkilere neden olabilmektedir. Düşük öz saygı, artan korku, kendine zarar verme düşüncesi, öfke ve depresyon gibi olumsuz sonuçlar siber zorbalığın yaygınlaşmasıyla birlikte ortaya çıkan önemli sorunlardır. Bu zararlı etkilerin toplum üzerindeki geniş çaplı etkileri, siber zorbalığın tespiti konusundaki çalışmaların önemini artırmıştır.

Bu çalışmaların amacı, dijital ortamlarda gerçekleşen zararlı davranışları tanımlamak ve etkili önlemler geliştirmektir. Bu bağlamda, makine öğrenimi ve derin öğrenme gibi teknolojik yaklaşımların siber zorbalık tespitinde kullanımı önemli başarılar sağlamıştır. Bu çalışmada, Türkçe metinlerden oluşan bir veri seti üzerinde siber zorbalık tespiti gerçekleştirilmiştir. Bu doğrultuda öncelikle çeşitli makine öğrenimi algoritmalarının performansları karşılaştırılmıştır. Daha sonra farklı derin öğrenme modellerinin performansları karşılaştırılmıştır. Burada farklı gömme modellerinin derin öğrenme modellerinin başarısına etkisi de incellenmiştir. Son aşamada makine öğrenmesi modelleri ve derin öğrenme modelleri başarısı birbiri ile kıyaslanmış ve en başarılı model tespit edilmiştir.

Raporun geri kalanı ise şu şekilde organize edilmiştir; ikinci bölümde bu alanda önceden yapılan çalışmalar incelenmiş, üçüncü bölümde bu projenin fizibilite çalışmaları aktarılmıştır. dördüncü bölümde sistemin analizi, beşinci bölümde ise sistemin tasarımı üzerinde durulmuştur. Altıncı bölümde uygulamadan bahsedilmiştir.

Yedinci bölümde projedeki deneyesel sonuçlardan, sekizinci bölümde modellerin performans analizinden, son bölüm olan sonuç bölümünde ise projenin elde edilen sonuçlar özetlenmiştir ve yorumlanmıştır. Gelecek çalışmalarda projeye katkı sağlayabilecek önerilerde bulunulmuştur.

2
Ön İnceleme

Makaleler ve konu ile ilgili diger çeşitli materyaller üzerinde yapılan detaylı incelemeler sonucunda makine ögrenimi teknikleri kullanılarak ve karşılaştırılarak Türkçe metinler üzerinde siber zorbalık tespitinin hangi teknoloji ve yöntemlerle gerçekleştirileceği netleştirilmiştir.

Bu araştırmalar sonucunda makine öğrenmesi modeli olarak Logistic Regression, Support Vector Classifier (SVC), Stochastic Gradient Descent (SGD) Classifier, Multinomial Naive Bayes, KNeighbors Classifier, Random Forest Classifier, Decision Tree Classifier, Ada Boost Classifier, Bagging Classifier modellerinin karşılaştırmalı olarak kullanılmasının yararlı olacağına karar verilmiştir.

Bu modellerin kullanılmasının sebebi hem geniş yelpazede makine öğrenmesi modellerini deneyerek daha doğru bir kıyaslamada bulunabilmek iken hem de literatürde yapılan farklı farklı çalışmalarda bu modellerin kullanılmasıdır [1, 2]. Böylece burada elde edilen bulgular ile diğer çalışmalarda elde edilen bulgular kıyaslanabilecektir.

Çalışmada veri seti olarak ise Kaggle [3] adlı paylaşım sitesinden elde edilmiş 11.000'den fazla Türkçe tivitten oluşan bir sosyal medya paylaşımı veri seti kullanılmıştır. Veri seti, pozitif ve negatif içerikli paylaşımları içeren dengeli bir yapıya sahiptir (%55 pozitif, %45 negatif).

Bir sonraki aşamada derin öğrenme modelleri de Türkçe metinler üzerinde siber zorbalık tespitinde kullanılmıştır. Derin öğrenme yöntemlerinden ise CNN, RNN, GRU, LSTM, BLSTM, BERT modelleri kullanılmış ve karşılaştırılmıştır. [4, 5]. Bununla birlikte FastText ve Glove gömme modelleri derin öğrenme modelleri ile birlikte kullanılarak kıyaslanmıştır. Böylece gömme modellerinin derin öğrenme modellerinin performansına etkisi incelenmiş ve sadece makine öğrenmesi modelleri veya derin öğrenme modelleri kendi içinde kıyaslanmakla kalmamış, makine öğrenmesi ve derin öğrenme modelleri birbirleri ile de karşılaştırılmıştır. En nihayetinde en başarılı model

nihai model olarak seçilerek siber zorbalık tespitinde kullanılmıştır.

Bu çalışmalarla birlikte, siber zorbalıkla mücadelede etkili çözümler geliştirmek için literatüre katkıda bulunulması hedeflenilmektedir. Sonuçların, çevrimiçi platformlarda güvenli bir iletişim ortamının sağlanmasına katkı sağlayacağı umulmaktadır.

3
Fizibilite

Bu bölümde proje için yapılan fizibilite araştırmasına yer verilmiştir. Teknik fizibilite, ekonomik fizibilite, zaman fizibilitesi ve yasal fizibilite üzerinde durulan ana başlıklardır.

3.1 Teknik Fizibilite

Teknik fizibilite yazılım fizibilitesi ve donanım fizibilitesi olmak üzere iki ayrı dala ayrılarak incelenmiştir.

3.1.1 Yazılım Fizibilitesi

Projenin yazılım fizibilitesi incelendiğinde, Google Colab [6] üzerinde Python kullanılarak gerçekleştirilmesi kararlaştırılmıştır. Bu platform, kullanıcı dostu arayüzü, farklı donanım hızlandırıcı seçenekleri ve ücretsiz bulut hesaplama kaynaklarıyla projenin geliştirilmesi için uygun bir ortam sağlamaktadır. Veri setleri doğrudan Kaggle platformundan alınmıştır ve bellekte işleneceği için ayrı bir veri tabanı yönetim sistemine ihtiyaç duyulmamıştır.

3.1.2 Donanım Fizibilitesi

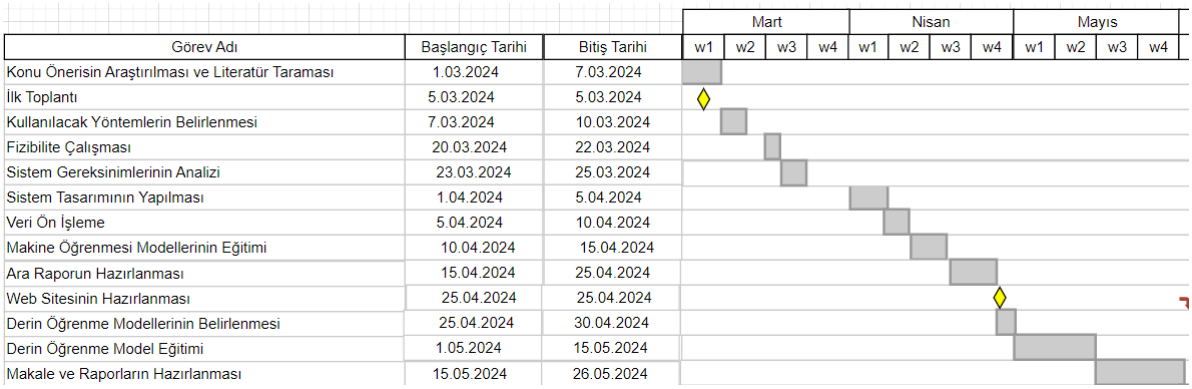
Modeller üzerinden yapılan çalışmalar Google Colaboratory üzerinden yapılmıştır. Ekip içi kod paylaşımı için GitHub platformu kullanılmıştır. Donanım anlamında özel bir fiziksel altyapıya ihtiyaç duyulmamış, bulut tabanlı hizmetler ve paylaşımlı geliştirme ortamları kullanılarak proje etkin bir şekilde yürütülmüştür.

3.2 Ekonomik Fizibilite

Bu proje herhangi bir gelir beklentisi olmadan hazırlanmıştır. Projedeki bilgisayar mühendisliği öğrencileri gönüllü olarak kendi bilgisayarları ile projeye katkı

sağlamıştır. Gerekli eğitimleri yapabilmek için ücretsiz versiyonu yeteli gelmemiş, 165 TL ile Colab Pro alınarak daha güçlü bir GPU'dan yararlanılmıştır.

3.3 Zaman Fizibilitesi



Görev Adı	Başlangıç Tarihi	Bitiş Tarihi
Konu Önerisin Araştırılması ve Literatür Taraması	1.03.2024	7.03.2024
İlk Toplantı	5.03.2024	5.03.2024
Kullanılacak Yöntemlerin Belirlenmesi	7.03.2024	10.03.2024
Fizibilite Çalışması	20.03.2024	22.03.2024
Sistem Gereksinimlerinin Analizi	23.03.2024	25.03.2024
Sistem Tasarımının Yapılması	1.04.2024	5.04.2024
Veri Ön İşleme	5.04.2024	10.04.2024
Makine Öğrenmesi Modellerinin Eğitimi	10.04.2024	15.04.2024
Ara Raporun Hazırlanması	15.04.2024	25.04.2024
Web Sitesinin Hazırlanması	25.04.2024	25.04.2024
Derin Öğrenme Modellerinin Belirlenmesi	25.04.2024	30.04.2024
Derin Öğrenme Model Eğitimi	1.05.2024	15.05.2024
Makale ve Raporların Hazırlanması	15.05.2024	26.05.2024

Şekil 3.1 Gantt Şeması

Projenin hazırlıkları 1 Mart 2024 tarihinde başlamış olup, belirli bir zaman çizelgesi dahilinde devam etmektedir. Gantt şeması, proje sürecinin planlanan aşamalarını ve zaman çizelgesi Şekil 3.1'de gösterilmektedir. Bu sayede proje, zaman yönetimi açısından etkili bir şekilde izlenmektedir.

3.4 Yasal Fizibilite

Projede kullanılan kaynaklar, kodlar ve veri seti yasal ve etik kurallara uygun olarak seçilmiştir. Kullanılan kodlar ve platformlar orijinal ve patentli olup, yasal haklara saygı gösterilmiştir. Ayrıca, kullanılan veri seti halka açık bir şekilde paylaşılmış olup, herhangi bir yasal kısıtlamaya tabi değildir. Projede yer alan tüm süreçler, yasal standartlara uygun olarak yürütülmüştür.

4
Sistem Analizi

Amaç: Bu sistem analizi çalışması, "Türkçe Metinlerde Siber Zorbalık Tespiti" için gerekli olan bileşenleri ve bu bileşenlerin işlevlerini tanımlamayı amaçlamaktadır.

Bileşenler:

1. Veri Seti: Hazır olarak 11.000 tivitten oluşan eğitim seti pozitif/negatif olarak etiketlenmiştir. Bu veri seti kullanılarak eğitim ve test yapılacaktır.
2. Ön İşleme: Veri setini eğitimden önce uygun hale getirilmesi içindir. Yöntem olarak kullanılan teknikler: boş verileri kaldırma, noktalama işaretleri ve stopwords kelimeleri temizleme, sık kullanılan kelimeleri kaldırma, kelimenin kökünü alma (lemmatization) ve emojileri kaldırma.
3. Makine Öğrenmesi: Makine öğrenimi bölümünde, farklı yapılarda çeşitli algoritmaların kullanılacağı belirtilmiştir. Bu algoritmalar arasında Logistic Regression, Support Vector Classifier (SVC), Stochastic Gradient Descent (SGD) Classifier, Multinomial Naive Bayes, KNeighbors Classifier, Random Forest Classifier, Decision Tree Classifier, Ada Boost Classifier, ve Bagging Classifier bulunmaktadır.
4. Derin Öğrenme: Derin öğrenme, karmaşık yapıdaki verilerin analizi ve özelliklerinin çıkarılması için kullanılan güçlü bir makine öğrenimi alt dalıdır. Bu kapsamda 3 farklı gömme yöntemi denenecek ve CNN, RNN, LSTM, BiLSTM, GRU ve BERT modelleri ile çalışma yapılacaktır.
5. Analiz: Analiz aşamasında, öncelikle çeşitli makine öğrenmesi ve derin öğrenme modellerinin performansları incelenecektir. Ayrıca, farklı gömme yöntemlerinin derin öğrenme modelleriyle çalıştırılacaktır. Bu modellerin, veri seti üzerinde eğitilerek elde edilen sonuçları değerlendirmek amacıyla farklı metriklerle ölçümleri yapılacaktır ve karşılaştırılacaktır.

İşlevler:

1. Veri Seti: Eğitilecek olan modelin doğruluğu için veri setinin doğru etiketlenmesi, yeterli miktarda başarılı sonuç verecek kadar çok olmalıdır.

2. Ön İşleme: Veri seti girdisi doğru formatta hazırlanılacak şekilde farklı teknikler kullanılarak sadeleştirilir. Bu işlem, modelin daha kolay öğrenmesine ve sonuç vermesini sağlar.

3. Makine öğrenmesi: Bu bileşen, farklı yapılarda çeşitli algoritmaların kullanılmasını içerir. Bu algoritmalar, eğitim veri seti üzerinde eğitilir ve ardından test veri seti üzerinde değerlendirilir.

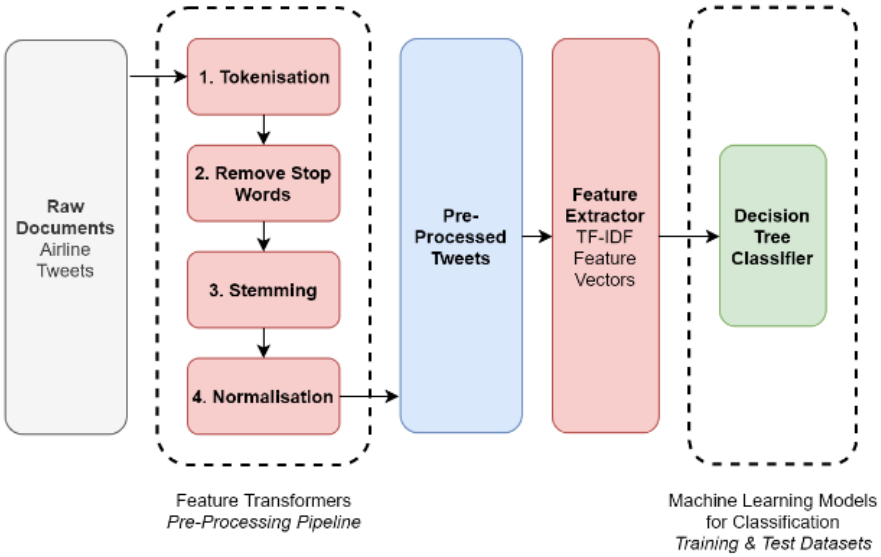
4. Derin öğrenme: Derin öğrenme modelleri, daha karmaşık yapıdaki verilere daha iyi uyum sağlayabilir ve daha yüksek doğruluk sağlayabilir. Bu nedenle, derin öğrenme modellerinin sisteme dahil edilmesi, siber zorbalık tespiti performansını artırabilir.

5. Analiz: Elde edilen sonuçlar, her bir modelin siber zorbalık tespiti üzerindeki etkinliğini ve doğruluğunu belirlemek için karşılaştırılacaktır. Bu karşılaştırma süreci, projenin amacına ulaşmak ve güvenli bir çevrimiçi iletişim ortamı sağlamak için en yüksek performanslı modelin belirlenmesine olanak sağlayacaktır.

5
Sistem Tasarımı

Sistem tasarımını yazılım tasarımı, veritabanı tasarımı ve girdi-çıktı tasarımı olmak üzere üç bölüme ayırarak incelemek mümkündür. Şekil 5.1, yazılım mimarisini göstermektedir.

5.1 Yazılım Tasarımı



Şekil 5.1 Yazılım Mimarisi

5.1.1 Ön İşleme

1. NaN olmayan değerlere sahip olan satırları seçme: İlk adım, veri setinizdeki eksik veya boş değerlere sahip satırları filtrelemektir. Bu yöntem, analizlerinizin doğruluğunu sağlar.

2. Özel karakterleri temizleme: Metin verilerindeki özel karakterler genellikle analizi karmaşıklaştırır veya yanıltıcı olabilir. Bu adımda, noktalama işaretleri,

parantezler, tırnak işaretleri gibi özel karakterleri temizleyerek metin verilerini basitleştirilir.

3. Stopword temizleme: Metin içinde sıkça kullanılan ancak analiz için önemsiz olan kelimeleri (stop kelimeler) temizlenir. Bu adım, analizin daha net ve odaklanmış olmasını sağlar. Örneğin, Türkçe için sıklıkla kullanılan "ve", "ama", "veya" gibi bağlaçlar, "bu", "bir", "şu" gibi belirsizlik ifade eden kelimeler temizlenebilir.

4. Kelime Kökünü Alma (Lemmatization): Lemmatization, kelimelerin dil bilgisine uygun kök formlarını bulmayı sağlar. Örneğin, "çalışıyor", "çalıştı", "çalışmak" gibi farklı formlardaki kelimeleri "çalış" kelimesine dönüştürebiliriz. Bu, metin verilerindeki çeşitliliği azaltır ve analizi kolaylaştırır.

   Çok karşılaştırılan iki yöntem olan lemmatization ve stemming, metin verilerindeki kelimelerin köklerini bulmayı sağlayan iki farklı yöntemdir. Lemmatization, dil bilgisi kurallarını kullanarak kelimeleri dil bilgisine uygun kök formlarına dönüştürürken, stemming daha basit bir yaklaşımla kelimeleri köklerine indirger. Lemmatization genellikle daha doğru sonuçlar sağlar ancak daha karmaşıktır, stemming ise daha hızlı ve hafif bir işlemdir ancak bazı durumlarda doğrulukta azalma riski taşır.Şekil 5.2'de köklenme ve lemmatizasyonun farkını göstermektedir.

Stemming is a simple and practical approach that involves cutting off the ends of words with the intention of obtaining the correct root form.	Lemmatization aims to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the '*lemma*'.
Eg: automate, automatic, automation → automat	Eg: car, cars, cars', car's → car am, is, are → be

Şekil 5.2 Köklenme ve Lemmatizasyon

5. Emojileri Silme: Metin verilerindeki emojiler sıklıkla duygusal ifadeleri veya içeriği ifade etmek için kullanılır, ancak analizi karmaşıklaştırabilirler. Bu adımda, metin verilerinden emojileri temizleriz veya dönüştürürüz.

5.1.2 TF-IDF

TfidfVectorizer, metin verilerini vektör haline getirmek için kullanılan bir vektörleştirme aracıdır. TF-IDF (Term Frequency-Inverse Document Frequency) yöntemini kullanarak metin verilerini sayısal vektörlere dönüştürülür.[7]

TF-IDF, bir belgedeki bir kelimenin önemini ölçen bir istatistiksel ölçüdür. Bir kelimenin TF-IDF değeri, o kelimenin bir belgedeki sıklığını (TF) ve tüm belgelerdeki görünümünün nadirliğini (IDF) birleştirir. Böylece, TF-IDF, bir kelimenin bir belgede ne kadar önemli olduğunu ölçmek için kullanılır.

Bu özellikle belirli bir metinde sıkça geçen kelimelerin önemli olabileceği, ancak genel olarak dilde yaygın olan kelimelerin (örneğin, "ve", "ama", "veya" gibi bağlaçlar) daha az önemli olduğu durumlarda faydalıdır. Bu nedenle, TfidfVectorizer, metin sınıflandırma veya kümeleme gibi doğal dil işleme uygulamalarında sıklıkla kullanılır.

5.1.3 Makine Öğrenmesi Modelleri

```
models = [('LogisticRegression', LogisticRegression(solver='newton-cg', multi_class='multinomial')),
          ('SVC', SVC(kernel = "rbf")),
          ('SGDClassifier', SGDClassifier(tol=1e-3, penalty='l2')),
          ("MultinomialNB",MultinomialNB()),
          ("KNeighborsClassifier",KNeighborsClassifier()),
          ("RandomForestClassifier",RandomForestClassifier (random_state=16)),
          ("DecisionTreeClassifier",DecisionTreeClassifier(random_state=16)),
          ("AdaBoostClassifier",AdaBoostClassifier()),
          ("Bagging Classifier",BaggingClassifier())
          ]
```

Şekil 5.3 Makine Öğrenme Modelleri

Şekil 5.3'te kullanılan makine öğrenme modelleri gösterilmiştir.

1. Lojistik Regresyon: Doğrusal regresyonun bir türevidir ve sınıflandırma problemleri için kullanılan bir makine öğrenimi modelidir.

   Lojistik Regresyon, bir giriş özelliği setini alır ve bu özellikleri kullanarak bir nesnenin belirli bir sınıfa ait olup olmadığını tahmin etmek için bir çıktı sağlar. Sınıflandırma problemi ikili (binary) olduğunda, yani iki farklı sınıf arasında bir seçim yapılması gerektiğinde, Lojistik Regresyon kullanılır. Örneğin, bir e-postanın spam olup olmadığını belirlemek gibi.

   Sonuç olarak, bu modeli proje için kullandığımızda çok sınıflı sınıflandırma problemleri için Lojistik Regresyon modelini oluşturur ve bu modelin optimize edilmesi için belirli bir algoritmayı (Newton-Raphson) kullanır.

2. Destek Vektör Makinesi (SVM): Bu kod parçası, SVC (Support Vector Classifier) olarak bilinen Destek Vektör Makinesi (SVM) modelini oluşturur. SVM, sınıflandırma problemleri için güçlü ve yaygın olarak kullanılan bir makine öğrenimi modelidir.

   SVC, bir giriş özelliklerini alır ve bu özellikleri kullanarak verileri iki veya daha fazla sınıfa ayırmaya çalışır. SVC'nin temel amacı, farklı sınıflar arasındaki

ayrımı maksimize etmektir. Bunun için, veri noktalarını sınıflar arasındaki en iyi ayrıma hizmet eden hiperdüzlemler etrafında gruplamak için çalışır. SVC, bu hiperdüzlemleri oluştururken margin (kenar boşluğu) kavramını kullanır ve bu kenar boşluğunu maksimize etmeye çalışır [8].

Kod parçasında kullanılan kernel = "rbf" parametresi, SVC'nin RBF (Radial Basis Function) çekirdeğini kullanarak sınıflandırma yapacağını belirtir. RBF çekirdeği, genellikle non-lineer veri kümeleri için tercih edilir, çünkü verileri daha yüksek boyutlu bir uzaya dönüştürerek karmaşık sınıflandırma problemlerini çözmeye yardımcı olabilir. RBF çekirdeği, veri noktalarını çekirdeğin merkezine olan uzaklıklarına dayanarak bir uzayda dönüştürür. Bu, verileri lineer olarak ayrılamayan durumlarda bile etkili bir şekilde sınıflandırmaya yardımcı olabilir.

3. Skoloastik Gradyan Azalma (SGD): SGDClassifier, büyük veri kümeleri üzerinde etkili bir şekilde çalışabilen ve çeşitli makine öğrenimi problemlerini çözebilen bir algoritmadır.

   SGDClassifier, stokastik gradyan azalma algoritmasını kullanarak modeli eğitir. Bu algoritma, her bir eğitim örneği üzerindeki gradyanı kullanarak model parametrelerini günceller. Bu, tüm eğitim veri setini her bir eğitim iterasyonunda değil, rastgele seçilen örnekler üzerinde çalışarak yapılır. Bu, büyük veri setleri için daha hızlı eğitim sağlar.

   Kod parçasında kullanılan tol=1e-3 parametresi, algoritmanın durma kriterini belirtir. Bu değer, iterasyonlar arasındaki parametre değişikliğinin belirli bir eşiği aştığında eğitimi durdurur. penalty='l2' parametresi, L2 düzenlemesi kullanılarak modelin aşırı uyumu (overfitting) önler. L2 düzenlemesi, modelin karmaşıklığını kontrol etmek ve genelleştirme performansını artırmak için kullanılan bir regülarizasyon tekniğidir.

   SGDClassifier, genellikle büyük veri kümeleri ve yüksek boyutlu özellik uzayları için tercih edilir. Gradient Descent algoritmasının stokastik doğası, veri setinin boyutundan bağımsız olarak etkili bir şekilde çalışmasını sağlar. Bu nedenle, SGDClassifier, büyük ölçekli NLP uygulamaları gibi alanlarda sıklıkla kullanılır.

4. Multinomial Naive Bayes: Belirli bir öznitelik uzayında belirli bir sınıfa ait olma olasılığını hesaplamak için Bayes teoreminin temel alındığı bir olasılıksal sınıflandırma modelidir.

   Multinomial Naive Bayes, özelliklerin birden fazla nominal veya kategorik olabileceği durumlar için özellikle uygun bir modeldir. Örneğin, belirli bir metinde kelime sayılarını temsil eden özellikler üzerinde sınıflandırma yapmak için sıkça kullanılır.

Multinomial Naive Bayes modeli, her sınıf için özelliklerin olasılık dağılımını hesaplar. Ardından, yeni bir örnek verildiğinde, bu olasılık dağılımlarını kullanarak örneğin her sınıfa ait olma olasılığını tahmin eder ve en yüksek olasılığa sahip sınıfı tahmin eder.

Model, Bayes teoremi kullanılarak sınıf olasılıklarını hesaplar. Bu teorem, bir örneğin belirli bir sınıfa ait olma olasılığını, verilen özelliklere dayanarak hesaplamak için kullanılır. Çoklu sınıflandırma problemlerinde, sınıflar arasındaki olasılıkları karşılaştırarak en yüksek olasılığa sahip sınıfı seçer.

Sonuç olarak, Multinomial Naive Bayes sınıflandırıcısı, özellikle metin sınıflandırma gibi NLP uygulamalarında yaygın olarak kullanılan bir modeldir. Özellikle kategorik verilerle çalışırken ve sınıf etiketleri birden fazla olasılık dağılımı içeriyorsa etkilidir.

5. K-En Yakın Komşular(KNN): K-NN sınıflandırıcısı, bir veri noktasını sınıflandırmak için yakınındaki komşuların etiketlerini kullanır. Bir veri noktası, en yakın k komşusunun etiketlerine bakılarak sınıflandırılır. Burada, k bir hiperparametredir ve kullanıcı tarafından belirlenir. Verilen bir veri noktasının sınıfı, çoğunluk sınıfının oylamasıyla belirlenir. Örneğin, bir veri noktası için en yakın k komşusunun 3'ü sınıf A, 2'si sınıf B ise, o veri noktası sınıf A olarak sınıflandırılır.

   K-NN sınıflandırıcısı, veri noktalarının özellik uzayında yakınlıklarına dayanarak sınıflandırma yapar. Bu nedenle, model eğitimi sırasında herhangi bir öğrenme süreci gerçekleştirmez, sadece veri noktalarının konumlarını hatırlar. Bu, eğitim sürecinin oldukça hızlı olmasını sağlar.

   Ancak, K-NN sınıflandırıcısı, özellikle çok büyük veri setleri ve yüksek boyutlu özellik uzayları için maliyetli olabilir, çünkü tahmin yapmak için tüm veri noktalarının uzaklıklarının hesaplanması gerekir.

   Sonuç olarak, K-NN sınıflandırıcısı basit yapısı ve doğrudan uygulanabilirliği nedeniyle popülerdir, ancak büyük veri setlerinde ve yüksek boyutlu özellik uzaylarında bazı kısıtlamalara sahiptir.

6. Rasgtele Orman: Birçok karar ağacının bir araya gelmesiyle oluşan bir makine öğrenimi modelidir. Her bir karar ağacı, rastgele alt kümelerden oluşturulur ve en iyi ayrımı yapacak özellikler kullanılarak büyütülür.

   Model oluşturulurken 'andom-state=16' parametresi kullanılarak rastgelelik seviyesi belirtilir. Bu sayede model her çalıştırıldığında aynı sonuçları verir, tekrarlanabilirlik sağlanır.

   Rasgtele Orman sınıflandırıcısı, her bir karar ağacının tahminlerinin

ortalamasını alarak veya çoğunluk oylaması yaparak tahminlerde bulunur. Bu, her bir karar ağacının tek başına yaptığı tahminlerin toplu olarak değerlendirilmesiyle daha güvenilir tahminler elde edilmesini sağlar.

Rasgtele Orman, aşırı uyumu önlemek için karar ağaçlarının rastgele alt kümelerinin kullanılması sayesinde genellikle yüksek performans gösterir. Çeşitli makine öğrenimi problemlerinde başarılı sonuçlar elde etmek için yaygın olarak kullanılır.

7. Karar Ağacı: Veri setini özelliklerine göre sınıflandıran bir ağaç yapısıdır ve sınıflandırma ve regresyon problemleri için kullanılabilir.

   Karar Ağacı sınıflandırıcısı, veri setini sınıflandırmak için özelliklerin değerlerine göre basit bir "if-else" mantığı kullanır. Her düğümde, veri setinin bir özelliği belirli bir eşik değeriyle karşılaştırılır ve veri seti bu eşik değerine göre iki veya daha fazla alt kümeye ayrılır. Bu işlem, her düğümde tekrarlanarak bir ağaç yapısı oluşturulur.

   Modelin oluşturulması sırasında kullanılan 'random-state=16' parametresi, Karar Ağacı sınıflandırıcısını oluştururken rastgelelik seviyesini belirtir. Modelin tekrarlanabilir sonuçlar üretmesini sağlar.

   Karar Ağacı sınıflandırıcısı, veri setindeki ilişkileri ve desenleri anlamak için kullanışlıdır, çünkü sonuçları insan tarafından anlaşılabilir bir ağaç yapısıyla sunar. Ancak, tek bir karar ağacı genellikle aşırı uyuma (overfitting) eğilimlidir, bu nedenle Random Forest gibi birçok ağacın birleştirilmesi gerekebilir.

   Karar Ağacı sınıflandırıcısı, basitlik ve yorumlanabilirlik avantajları nedeniyle tercih edilir. Ancak, daha karmaşık veri kümelerinde ve yüksek boyutlu özellik uzaylarında bazı kısıtlamalara sahip olabilir.

8. AdaBoost : Zayıf sınıflandırıcıları bir araya getirerek güçlü bir sınıflandırıcı oluşturan bir ensemble yöntemidir. Zayıf sınıflandırıcılar, veri setindeki hatalı örnekleri daha fazla vurgulayarak ardışık olarak eğitilir. Bu, sonraki sınıflandırıcıların hatalı örnekleri daha iyi düzeltebilmesini sağlar.

   Modelin oluşturulması sırasında AdaBoostClassifier() parametresi, AdaBoost sınıflandırıcısını varsayılan parametrelerle oluşturur. AdaBoost sınıflandırıcısı, genellikle yüksek doğruluk ve genelleme performansı sağlar ve farklı veri kümeleri üzerinde geniş bir uygulama yelpazesine sahiptir.

   Ancak, AdaBoost, dengesiz ve gürültülü veri kümelerinde hassas olabilir. Ayrıca, çok karmaşık veri kümelerinde performansı azalabilir. Sonuç olarak, AdaBoost, etkili bir sınıflandırma çözümü sunar ve geniş bir uygulama alanına sahiptir.

9. Bagging: Birçok temel sınıflandırıcıyı (genellikle karar ağaçları) bir araya getirerek daha güçlü bir sınıflandırıcı oluşturan bir ensemble yöntemidir. Veri setinden rastgele örneklemler alınarak oluşturulan alt kümeler üzerinde her biri ayrı bir temel sınıflandırıcı eğitilir. Ardından, tahminlerin bir araya getirilmesiyle final tahmin yapılır.

   AdaBoost'tan farklı olarak, Bagging her bir temel sınıflandırıcının katkısını eşit olarak kabul eder, ağırlık ayarı yapmaz. AdaBoost ile Bagging'in ana farkları, örnek seçimi ve ağırlık ayarlaması gibi adımlarda görülür. AdaBoost, örneklerin ağırlıklarını ayarlayarak zor örneklerin daha fazla vurgulanmasını sağlarken, Bagging rastgele örnekleme yöntemi kullanarak farklı örnek alt kümeleri üzerinde eğitilir. AdaBoost, genellikle yüksek doğruluk sağlarken, Bagging daha kararlı sonuçlar verir ve genellikle daha iyi genelleme performansı sağlar.

5.1.4 Gömme Modelleri

Projede derin öğrenme modellerinde kullanılmak üzere önceden eğitilmiş 3 farklı gömme modeli kullanılmıştır.

1. Keras Embedding: Keras'ın yerleşik embedding katmanını kullanarak kelimeleri vektör olarak temsil eder.

2. GloVe: Bu proje Stanford tarafından açık kaynak kodlu olarak geliştirilmiştir. Kelimeleri vektörler olarak temsil eder ve bu vektörler arasındaki mesafeye bakarak anlamsal ilişkiler kurar. Bu model, farklı boyutlarda vektörler kullanılark eğitilir. Projede 100 boyutlu model kullanılmıştır.

3. FastText: Facebook tarafından geliştirilmiştir. Kelimenin yanı sıra kelime parçalarını birkaç harf bazlı "n-gram" halinde parçalayan bir yöntemdir. Bu nedenle, özellikle Türkçe gibi morfolojik olarak zengin dillerde daha iyi bir performans gösterebilir. Projede 300 boyutlu bir model kullanılmıştır.

5.1.5 Derin Öğrenme Modelleri

Türkçe metinlerde siber zorbalık tespiti için makine öğrenmesinden sonra derin öğrenme modellerinin kullanılmıştır.

1. Evrişimsel Sinir Ağları (CNN): Projede 5 katmanlı bir model kullanılmıştır. İlk katmanı olan gömme katmanı, verilen gömme matrisi kullanılarak girdi metinlerini gömme vektörlerine dönüştürür. 'trainable=False' parametresi

ile, gömme matrisi eğitilemez. İkinci katman, evrişim katmanıdır(Conv1D). Bu katmanda 128 filtre ve 5 birimlik kernel boyutu kullanılır. Aktivasyon fonksiyonu olarak da ReLu kullanılır. Bir sonraki katman MaxPooling katmanıdır. Burada, evrişim çıktılarının boyutunu azaltmak ve belirli özelliklerin vurgulanması içindir. Dördüncü katmanda düzleştirme katmanıdır. Elde edilen 2 boyutlu çıktı düzleştirilir ve son katman olan yoğun katman ikili bir sınıflandırma yapar. Aktivasyon fonksiyonu olarak sigmoid kullanılır, çünkü çıktının 0 ile 1 arasında olmasını sağlar.

2. Yinelemeli Sinir Ağları (RNN): RNN, sıralı verilerde işlevli bir modeldir. Kullanılan modelde 3 katman vardır: İlk olarak gömme katmanı, CNN ile aynıdır. İkinci olaran SimpleRNN katmanı vardır. SimpleRNN, en temel RNN türlerinden biridir. 128 gizli birim (hidden unit) kullanılır. Ayrıca, 'return_sequences=False' parametresi ile sadece son adımının çıktısını döndürür. Son katman yine yoğun katmandır. İkili sınıflandırma yapar ve aktivasyon olarak sigmoid kullanır.

3. Uzun Kısa Süreli Bellek (LSTM): LSTM, RNN' den türetilmiş bir modeldir. RNN' in uzun vadeli zorluklarına çözüm sunar. Giriş, çıkış ve unutma kapıları kullanır. Kullanılan LSTM modelinde RNN den farklı olarak sadece LSTM olarak değiştirilmiştir.

4. İki Yönlü Uzun Kısa Süreli Bellek (BiLSTM): LSTM hücrelerinin geliştirilmiş versiyonudur. Temel özelliği sıralı verileri ileri ve geri yönde işleyerek iki tabakayı bir araya getirir. Bu sayede daha kapsamlı bir bağlam sunabilir.

5. Kapı Özyinelemeli Geçitler (GRU): LSTM gibi bir RNN çeşididir. Uzun vadeli bağımlılıkları alırken LSTM' den daha basit bir yapı sunmaktadır. Temel olarak, GRU' da bir güncelleme kapısı ve bir sıfırlama kapısı bulunur; bu kapılar, mevcut girdi ile önceki gizli durum arasındaki ilişkiyi kontrol eder. GRU, daha az parametreye sahip olması ve daha hızlı öğrenme süreçleri sunar. Bu sebeple, daha küçük veri setleri üzerinde kullanılabilir.

6. Transformatörlerden Çift Yönlü Kodlayıcı Temsilleri (BERT): BERT, Google tarafından geliştirilmiştir. Doğal dil işleme görevlerinde iyi performans vermektedir. İki yönlü (bidirectional) olduğundan kelimelerin hem önceki hem de sonraki bağlamlarına bakar, böylece metinlerdeki anlamı daha iyi kavrar. Ayrıca önceden eğitilmiş (pre-trained) BERT modeli ince ayar (fine-tuning) yapılarak çeşitli görevlerde başarılı bir performans gösterebilir. Bu proje için dbmdz/bert-base-turkish-uncased önceden eğitilmiş modeli kullanılarak siber zorbalık tespiti yapılmıştır. Model, BERT tabanlı bir ön işleme katmanı, bir yoğun katman ve bir sigmoid aktivasyon katmanı içerir.

Tüm derin öğrenme modellerinde, 'adam' optimize edici kullanılarak binary_crossentropy kaybı minimize edilmeye çalışılır ve doğruluk (accuracy) performansları izlenir.

5.1.6 Model Eğitimi

Modeli eğitirken kullanılan cross-validation yöntemi olan K-fold çapraz doğrulama eğitmek ve değerlendirmek için tasarlanmıştır. K-fold çapraz doğrulama, veri setini rastgele alt kümeler (katmanlar) halinde böler ve her bir alt küme üzerinde modeli eğitip değerlendirerek genel model performansını değerlendirir. [9] K-Fold nesnesi, belirtilen sayıda (n-splits) alt kümeye bölen ve her bir alt küme için eğitim ve test dizinlerini oluşturan bir çapraz doğrulama yöntemidir. Bu sayede veri seti üzerinde tekrar tekrar eğitim ve test işlemleri gerçekleştirilir. Fonksiyonun işleyişi şu adımlardan oluşur:

1. Belirtilen sayıda K-fold'a (makine öğrenmesi için 15 fold, derin öğrenme için 5 fold) göre veri setini böler.
2. Her bir katman için eğitim ve test dizinlerini oluşturur.
3. Eğitim verilerini ve etiketlerini seçer, modeli eğitir.
4. Test verilerini kullanarak modelin tahminlerini yapar.
5. Gerçek etiketlerle yapılan tahminlerin doğruluğunu değerlendirir.
6. Her bir katman için doğruluk skorunu hesaplar ve bunları bir listeye kaydeder.
7. Tüm katmanlar işlendiğinde, doğruluk skorlarının ortalamasını hesaplar ve bunu bir değişkene atar.

Bu işlem, modelin genel performansını daha güvenilir bir şekilde değerlendirmeye yardımcı olur. Çünkü veri seti üzerindeki tüm örnekler hem eğitim hem de test için kullanılmış olur, bu da modelin genelleme yeteneğini daha iyi değerlendirmemizi sağlar.

5.2 Veritabanı Tasarımı

Bu proje kapsamında, Kaggle veri paylaşım platformundan elde edilen veri seti kullanılarak siber zorbalık tespiti için bir sistem geliştirilmektedir. Veri seti, 11.000'den fazla Türkçe tivitten oluşmaktadır ve pozitif/negatif olarak etiketlenmiştir. Bu veri

```python
def k_fold (vectorizer, model, data, name):

    pipeline = Pipeline([('vect', vectorizer),
                         ('chi', SelectKBest(chi2, k="all")),
                         ('clf', model)])
    kf = KFold(n_splits=15, shuffle=True)
    scores = []

    for train_index, test_index in kf.split(df):
        train_text = df.iloc[train_index]['emojisiz'].values.astype('U')
        train_y = df.iloc[train_index]['sınıf'].values.astype('U')

        test_text = df.iloc[test_index]['emojisiz'].values.astype('U')
        test_y = df.iloc[test_index]['sınıf'].values.astype('U')

        model = pipeline.fit(train_text, train_y)
        predictions = model.predict(test_text)
        with warnings.catch_warnings():
            warnings.filterwarnings(action='once')
            score = accuracy_score(test_y, predictions)
            print(score)
            scores.append(score)
            skor=str(sum(scores)/len(scores))


        return skor
```

Şekil 5.4 K-fold Cross-Validation

seti %55 pozitif, %45 negatif paylaşım içeren dengeli bir veri setidir. Bu veri setinin temizlenmesi ve işlenmesi için Google Colab üzerinde Python kullanılmıştır.

Veriler Kaggle platformundan çekilerek doğrudan bellekte işlenebildiği için veritabanı tasarımına ihtiyaç duyulmamıştır. Bu durumda, veritabanı yönetim sistemleriyle ilgili konfigürasyon veya kurulum gereksinimi ortadan kalkmaktadır.

Bu tasarım, projenin verimli bir şekilde gerçekleştirilmesini sağlamış ve siber zorbalık tespiti için gerekli makine öğrenmesi ve derin öğrenme modellerinin etkili bir şekilde geliştirilmesine imkan tanımıştır. Veritabanı gereksinimi olmadığı için, projenin karmaşıklığı azalamış ve veri işleme süreçleri daha hızlı bir şekilde gerçekleştirilebilmiştir.

5.3 Girdi - Çıktı Tasarımı

Projede öncelikle veri seti üzerinde makine öğrenmesi, sonrasında derin öğrenme modelleri denenmiştir. Burada farklı gömme modellerinin derin öğrenme model performanslarına etkisi de incelenmiştir. Tüm modeller arasından en iyi performansı

veren model seçilmiştir. Son aşamada, yapılan tüm yöntemlerin doğruluk metrikleri karşılaştırılmış ve en iyi sonuç veren modelle test edilip f1,recall,accuracy,precision değerleri projenin çıktısı olarak belirlenmiştir.

6
Uygulama

Bu bölümde çalışmada ele alınan problemi çözmek için geliştirilen uygulama anlatılmış, bu uygulamanın adımlarına detaylı bir şekilde yer verilmiştir.

6.1 Makine Öğrenmesi Modellerinin Eğitimi

Uygulama kısmında seçilen makine öğrenmesi modelleri bu kısımda detaylı bir şekilde incelenecektir.

6.1.1 Veri Ön İşleme

Şekil 6.1'de ve Şekil 6.2'de görüldüğü üzere veri seti, ilk olarak CSV dosyasından okunmuş ve bazı ön işlemlerden geçirilmiştir.

```
[ ]  from google.colab import drive
     drive.mount('/content/drive')

     Mounted at /content/drive

[ ]  import os
     os.chdir("drive/My Drive/AraProje-BanuDiri/Cyberbullying")
```

Şekil 6.1 Google Colab, Google Drive'a baglanma

```
[ ]  import numpy as np
     import pandas as pd
     df = pd.read_csv("tweetset.csv")
     df.head(1000)
```

Şekil 6.2 Veri Setinin Okunması

Öncelikle, veri setindeki boş değerler (NaN) temizlenmiş ve veri setinin yapısına uygun yeni bir CSV dosyası oluşturulmuştur. Bu işlem sırasında, veri setindeki paylaşımların

```
import numpy as np
import pandas as pd
df = pd.read_csv("tweetset.csv")
df.head(1000)
```

	Tip	Paylaşım	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5
0	Negatif,"Doğa ağzımıza sıçsa hakkı var",,,,	NaN	NaN	NaN	NaN	NaN
1	Pozitif,"Anne bir sanatçıdır, en güzel eseri d...	NaN	NaN	NaN	NaN	NaN
2	Negatif,"ibrahimin oğlunu koruyan Tanrı'ya da ...	NaN	NaN	NaN	NaN	NaN
3	Negatif,"Köpeğim suratına sıçsın senin namussu...	NaN	NaN	NaN	NaN	NaN
4	Negatif,"Ben söğüşledim, birazda sen söğüşle",,,,	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...
995	Negatif,"bunun gibi yollu kadınlara yol verece...	NaN	NaN	NaN	NaN	NaN
996	Negatif,"namussuz karı yoldan çıkarma milleti"...	NaN	NaN	NaN	NaN	NaN
997	Pozitif,"kimse kimseye namussuz diyemez",,,,	NaN	NaN	NaN	NaN	NaN
998	Negatif,"şerefsizlik senin kanına işlemiş",,,,	NaN	NaN	NaN	NaN	NaN
999	Negatif,"senin kanını akıtır sana içiririm",,,,	NaN	NaN	NaN	NaN	NaN

1000 rows × 6 columns

Şekil 6.3 Kaggle'dan indirilmiş CSV dosyası

türü (Tip) ve içeriği (Paylaşım) sütunlarına odaklanılmıştır. Veri setindeki boş değerler Şekil 6.3'te göstermektedir.

Kaggle'dan indirilmiş csv dosyasının bazı formatsal problemlere sahip olduğu gözlemlenebilmektedir, bazı düzenlemelere ihtiyaç vardır. Veri setinin düzenlenmiş hali Şekil 6.4'te yer almaktadır.

```
import pandas as pd
df = pd.read_csv("fixed_tweetset.csv")
df.head(1000)
```

	Tip	Paylaşım
0	Negatif	"Doğa ağzımıza sıçsa hakkı var"
1	Pozitif	"Anne bir sanatçıdır
2	Negatif	"ibrahimin oğlunu koruyan Tanrı'ya da ben soka...
3	Negatif	"Köpeğim suratına sıçsın senin namussuz karı"
4	Negatif	"Ben söğüşledim
...	...	...
995	Negatif	"namussuz karı yoldan çıkarma milleti"
996	Pozitif	"kimse kimseye namussuz diyemez"
997	Negatif	"şerefsizlik senin kanına işlemiş"
998	Negatif	"senin kanını akıtır sana içiririm"
999	Negatif	"kafanı keser kapının önüne asarım"

1000 rows × 2 columns

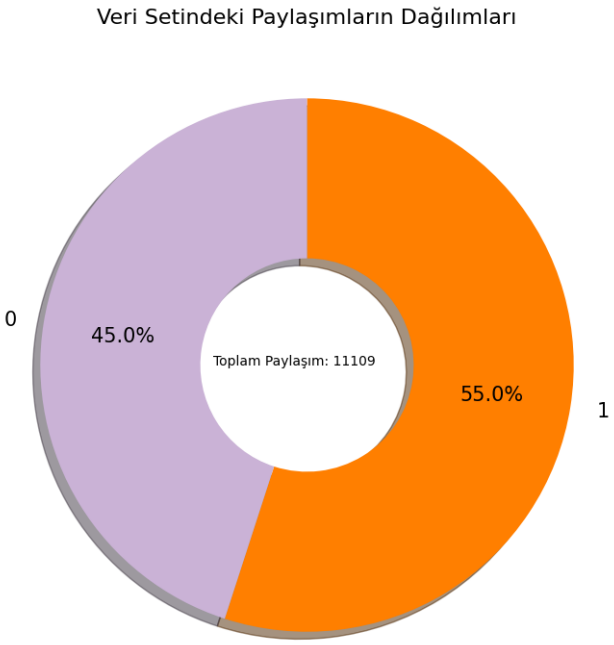
Şekil 6.4 Düzenlenmiş CSV dosyası

Metin verisi üzerinde çeşitli ön işlemler gerçekleştirilmiştir. Özel karakterler, gereksiz boşluklar ve emojiler temizlenmiş, ayrıca sık kullanılan kelimeler ve nadir kelimeler filtrelenmiştir. Ayrıca, metin verisi kelime köklerine indirgenmiş ve stop words (durak kelimeler) temizlenmiştir. Şekil 6.5'te metin ön işleme sonucu göstermektedir.

	Tip	Paylaşım	ozel_karaktersiz	stop_word	sık_kullanılan	kelime_kok	emojisiz	sınıf
0	Negatif	"doğa ağzımıza sıçsa hakkı var"	doğa ağzımıza sıçsa hakkı var	doğa ağzımıza sıçsa hakkı	doğa ağzımıza sıçsa hakkı	doğa ağzımıza sıçsa hakkı	doğa ağzımıza sıçsa hakkı	0
1	Pozitif	"anne bir sanatçıdır	anne bir sanatçıdır	anne sanatçıdır	anne sanatçıdır	anne sanatçıdır	anne sanatçıdır	1
2	Negatif	"ibrahimin oğlunu koruyan tanrı'ya da ben soka...	ibrahimin oğlunu koruyan tanrı'ya da ben sokayım	ibrahimin oğlunu koruyan tanrı'ya sokayım	ibrahimin oğlunu koruyan tanrı'ya sokayım	ibrahimin oğlunu koruyan tanrı'ya sokayım	ibrahimin oğlunu koruyan tanrı'ya sokayım	0
3	Negatif	"köpeğim suratına sıçsın senin namussuz karı"	köpeğim suratına sıçsın senin namussuz karı	köpeğim suratına sıçsın namussuz karı	köpeğim suratına sıçsın namussuz karı	köpeğim suratına sıçsın namussuz karı	köpeğim suratına sıçsın namussuz karı	0
4	Negatif	"ben söğüşledim	ben söğüşledim	söğüşledim	söğüşledim	söğüşledim	söğüşledim	0
5	Negatif	"şerefsizlik	şerefsizlik	şerefsizlik	şerefsizlik	şerefsizlik	şerefsizlik	0
6	Negatif	"kendisi de bilmiyordur çünkü beyinsiz"	kendisi de bilmiyordur çünkü beyinsiz	kendisi bilmiyordur beyinsiz	kendisi bilmiyordur beyinsiz	kendisi bilmiyordur beyinsiz	kendisi bilmiyordur beyinsiz	0
7	Negatif	"aşırı bıyıklı geleneksel keko adam	aşırı bıyıklı geleneksel keko adam	aşırı bıyıklı geleneksel keko adam	aşırı bıyıklı geleneksel keko adam	aşırı bıyıklı geleneksel keko adam	aşırı bıyıklı geleneksel keko adam	0
8	Pozitif	"en uzun yolculuklar bile	en uzun yolculuklar bile	uzun yolculuklar	uzun yolculuklar	uzun yolculuklar	uzun yolculuklar	1
9	Pozitif	"vallahi bu aralar istediğimiz tek şey huzur"	vallahi bu aralar istediğimiz tek şey huzur	vallahi aralar istediğimiz tek huzur	vallahi aralar istediğimiz tek huzur	vallahi aralar istediğimiz tek huzur	vallahi aralar istediğimiz tek huzur	1
10	Pozitif	"chp genel başkanı kemal kılıçdaroğlu'nun eşi ...	chp genel başkanı kemal kılıçdaroğlunun eşi se...	chp genel başkanı kemal kılıçdaroğlunun eşi se...	chp genel başkanı kemal kılıçdaroğlunun eşi se...	chp genel başkanı kemal kılıçdaroğlunun eşi se...	chp genel başkanı kemal kılıçdaroğlunun eşi se...	1

Şekil 6.5 Metin ön işleme

Ön işleme sonrası, veri setindeki metin verilerinin uzunlukları ve kelime sayıları histogram grafiği üzerinde görselleştirilmiştir. Şekil 6.6'da veri setindeki pozitif/negatif dağılımı gösterilmektedir.



Şekil 6.6 Veri Setindeki Paylaşımların Dağılımı (0: Negatif, 1: Pozitif)

Ayrıca, Şekil 6.7 ve Şekil 6.8'de metin verilerinden elde edilen word cloud (kelime bulutu) görselleriyle veri setinin içeriğine görsel bir bakış sunulmuştur.

Şekil 6.7 Negatif Metinler için WordCloud

Şekil 6.8 Pozitif Metinler için WordCloud

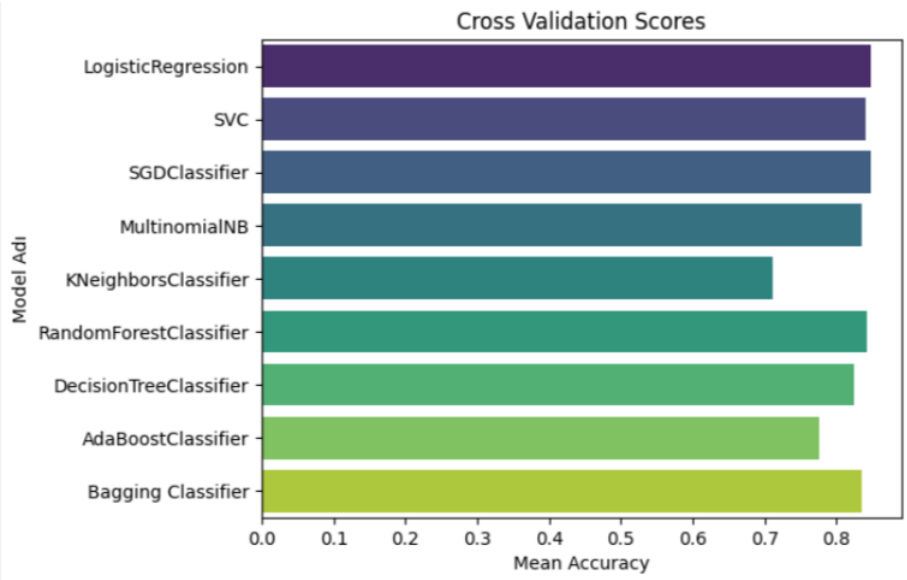
6.1.2 Model Eğitimi ve Değerlendirme

Farklı makine öğrenmesi algoritmaları kullanılarak model eğitimi gerçekleştirilmiştir. Eğitilen modeller arasında Logistic Regression, Support Vector Classifier (SVC), Stochastic Gradient Descent (SGD) Classifier, Multinomial Naive Bayes, K-Nearest Neighbors, Random Forest Classifier, Decision Tree Classifier, AdaBoost Classifier ve Bagging Classifier yer almaktadır.

Her bir model için k-fold çapraz doğrulama yöntemi kullanılarak doğruluk (accuracy) skorları hesaplanmıştır. Elde edilen sonuçlar Şekil 6.9'da sayısal olarak ve Şekil 6.10'da görsel olarak bar grafik üzerinde sunulmuştur.

	Model Adı	Başarım Oranı
0	LogisticRegression	0.847691213480687
1	SVC	0.8416628369259948
2	SGDClassifier	0.8487718082454926
3	MultinomialNB	0.8359934833619043
4	KNeighborsClassifier	0.7122174806385332
5	RandomForestClassifier	0.8419298245614034
6	DecisionTreeClassifier	0.8247345321029534
7	AdaBoostClassifier	0.7758587737535106
8	Bagging Classifier	0.8346373904268641

Şekil 6.9 Kullanılan Makine Öğrenme Modellerinin Başarım Oranları



Şekil 6.10 Modellerin Başarım Grafiği

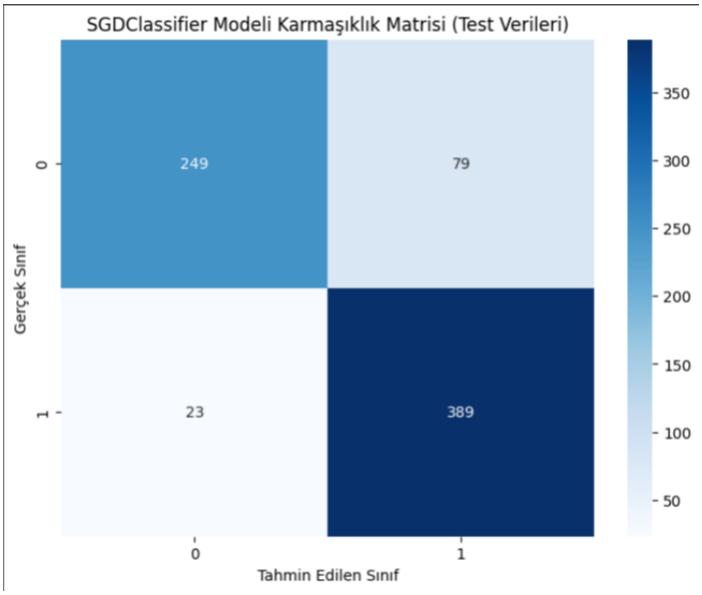
Ayrıca, en başarılı modelin belirlenmesi için başarımların karşılaştırılması yapılmış ve en yüksek başarıma sahip olan model vurgulanmıştır.Şekil 6.11'de en yüksek başarı oranı olarak SGDClassifier modeli gösterilmektedir.

SGD modelinin eğitilmesinde k-fold 15 parçaya bölünmüştür. Bu durumda test verisi de aynı şekile bölünür. Şekil 6.12'de modele ait bir test parçasından alınan karışıklık matrisi (Confusion Matrix) yer almaktadır.

```
def basarım (sonuc):
    max_basarım = max(sonuc,key = sonuc.get)
    return (max_basarım)
print("Yapmış olduğumuz 9 farklı makine öğrenmesi algoritması arasında en başarılı sonucu veren algoritma = {} ' dir".format(basarım(sonuc)))
```

```
Yapmış olduğumuz 9 farklı makine öğrenmesi algoritması arasında en başarılı sonucu veren algoritma = SGDClassifier ' dir
```

Şekil 6.11 En Başarılı Makine Öğrenmesi Modeli - SGDClassifier



Şekil 6.12 SGDClassifier Modelinin Karışıklık Matrisi

6.2 Derin Öğrenmesi Modellerinin Eğitimi

Seçilen derin öğrenme modelleri bu kısımda incelenecektir.

Makine öğrenmesi modelleri denenmeden önce veri seti ön işleme adımlarından geçirilip kullanılmıştı. Burada bu ön işlenmiş veri seti padding işlemi uygulanarak kullanılmıştır. Burada padding işleminin yapılmasının sebebi derin öğrenme modellerinin eğitim ve tahmin yaparken aynı uzunlukta girdilere ihtiyaç duymasıdır.

Şekil 6.13 padding işlemi yapılmadan önce verilerin boyutunu gösterirken, Şekil 6.14 "maxlen = 100" ile veri setindeki metin boyutunun yüz olarak sabitlendiğindeki veri boyutunu göstermektedir.

```
print("X_train shape:", train_x.shape)
print("y_train shape:", train_y.shape)
print("X_test shape:", test_x.shape)
print("y_test shape:", test_y.shape)
```

```
X_train shape: (8887,)
y_train shape: (8887,)
X_test shape: (2222,)
y_test shape: (2222,)
```

Şekil 6.13 Padding İşlemi Öncesi Veri Boyutu

```
# Sequence'leri pad etme
maxlen = 100  # Metinlerin aynı uzunlukta olmasını sağlamak için
train_padded = pad_sequences(train_sequences, maxlen=maxlen, padding='post')
test_padded = pad_sequences(test_sequences, maxlen=maxlen, padding='post')

print("train_padded shape:", train_padded.shape)
print("test_padded shape:", test_padded.shape)
```

```
train_padded shape: (8887, 100)
test_padded shape: (2222, 100)
```

Şekil 6.14 Padding İşlemi Sonrası Veri Boyutu

6.2.1 Modell Eğitimi ve Değerlendirme

Farklı gömme yöntemleri ve derin öğrenme modelleri kullanılarak eğitim gerçekleşmiştir. Gömme yöntemleri olarak FastText, GloVe, derin öğrenme modelleri olarak da CNN, RNN, LSTM, BiLSTM, GRU ve BERT yer almaktadır.

BERT haricinde her bir model için k-fold çaprazlama yöntemi ve farklı epoch değerleri kullanılarak doğruluk skorları hesaplanmıştır.

6.2.1.1 Keras Embedding Kullanılarak Derin Öğrenme Modellerinin Eğitimi

Keras embedding kullanıldığında derin öğrenmesi modellerinin eğitiminde kullanılan k-katlı çapraz doğrulama fonksiyonu Şekil 6.15'te sunulmuştur. Burada seçilen model 'n_splits' kata bölünür, her kat için model eğitilir ve değerlendirilir.

```
def k_fold_dl(build_fn, X, y, model_name, n_splits=5):
    kfold = KFold(n_splits=n_splits, shuffle=True, random_state=42)
    fold_no = 1
    accuracy_scores = []

    for train_index, val_index in kfold.split(X):
        print(f'Training fold {fold_no} for {model_name}')
        model = build_fn()

        X_train_fold = X[train_index]
        y_train_fold = y[train_index]
        X_val_fold = X[val_index]
        y_val_fold = y[val_index]

        history = model.fit(
            X_train_fold, y_train_fold,
            epochs=10,
            batch_size=32,
            validation_data=(X_val_fold, y_val_fold),
            verbose=0
        )

        scores = model.evaluate(X_val_fold, y_val_fold, verbose=0)
        print(f'Score for fold {fold_no}: {model.metrics_names[1]} of {scores[1]*100}%')
        accuracy_scores.append(scores[1])
        fold_no += 1

    return np.mean(accuracy_scores)
```

Şekil 6.15 K Katlı Çapraz Doğrulama Fonksiyonu

Sonrasında Keras embedding ile CNN, RNN, LSTM, BiLSTM, GRU ve BERT model mimarileri oluşturulmuştur. Şekil 6.16'da örnek olarak CNN model mimarisi verilmiştir.

```python
def build_cnn_model(input_length):
    model = Sequential([
        Embedding(input_dim=10000, output_dim=128, input_length=input_length),
        Conv1D(filters=128, kernel_size=5, activation='relu'),
        MaxPooling1D(pool_size=2),
        Flatten(),
        Dense(128, activation='relu'),
        Dense(1, activation='sigmoid')
    ])
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
    return model
```

Şekil 6.16 Keras Embedding - CNN Modeli

Keras embedding ve beş katlı çapraz doğrulama kullanılarak derin öğrenme modelleri eğitilmiştir. Şekil 6.17 örnek olarak CNN modelinin çıktısını göstermektedir.

```
Training fold 1 for CNN
Score for fold 1: accuracy of 84.36445593833923%
Training fold 2 for CNN
Score for fold 2: accuracy of 83.40832591056824%
Training fold 3 for CNN
Score for fold 3: accuracy of 84.97467637062073%
Training fold 4 for CNN
Score for fold 4: accuracy of 84.86212491989136%
Training fold 5 for CNN
Score for fold 5: accuracy of 86.55036687850952%
```

Şekil 6.17 Keras Embedding - CNN Model Performansı

Tüm modellerin bu şekilde eğitimi tamamlandıktan sonra gösterdikleri performanslar Şekil 6.18'de verilmiştir.

	Model Adı	Başarım Oranı
0	GRU	0.549003
1	RNN	0.551807
2	CNN	0.848320
3	LSTM	0.549003
4	BiLSTM	0.836166

Şekil 6.18 Keras Embedding - Derin Öğrenme Model Performansları

Bir sonraki aşamada Glove ve FastText gömme modelleri incelenecektir. Öncelikle modellerin vektörleri indirilip yüklenmiştir. Daha sonra embedding matrisleri oluşturularak bu matrislerle derin öğrenme model eğitimleri yapılmıştır. Böylece gömme modellerinin derin öğrenme model performansına etkisi incelenmiştir.

6.2.1.2 Glove ve FastText Modelleri Kullanılarak Derin Öğrenme Modellerinin Eğitimi

Beş katlı çapraz doğrulama ve üç epoch ile model eğitimleri belirtilen iki gömme modeli kullanılarak gerçekleştirilmiştir. Şekil 6.19 bu doğrultuda elde edilen çıktıları göstermektedir.

	Model Adı	Başarım Oranı
0	GRU_Glove	0.549007
1	RNN_Glove	0.517498
2	CNN_Glove	0.625410
3	LSTM_Glove	0.549002
4	BiLSTM_Glove	0.622147
5	GRU_FastText	0.549005
6	RNN_FastText	0.694598
7	CNN_FastText	0.858109
8	LSTM_FastText	0.549005
9	BiLSTM_FastText	0.863732

Şekil 6.19 Glove ve FastText - Derin Öğrenme Model Performansları

Sonuç olarak, her bir model için GloVe ve FastText embedding'leri kullanılarak ne kadar başarılı olduğu Şekil 6.19'da görülebilmektedir. Örneğin, CNN modeli GloVe embedding'lerini kullanarak %62.5 doğruluk elde ederken, FastText embedding'lerini kullanarak %85.8 doğruluk elde etmiştir. Bununla birlikte BiLSTM modeli FastText embedding'lerini kullanarak %86.4 doğruluk ile en yüksek doğruluk oranına sahiptir.

6.2.1.3 BERT Model Eğitimi

Siber zorbalık tespiti için kullanılan son derin öğrenme modeli ise BERT modelidir. Bu doğrultuda ilk olarak, veriler listeye dönüştürülmüş ve ardından BERT tokenizer kullanılarak token'lara ayrılıp tensor formatına dönüştürülmüştür. Model, GPU üzerinde eğitilmiş ve optimizer ile öğrenme planlayıcısı (scheduler) ayarlanmıştır. Eğitim tamamlandıktan sonra model, test verileri üzerinde değerlendirilmiştir. Şekil 6.20 üç epoch, Şekil 6.21 beş epoch değerleri için hesaplanan performans metriklerini (doğruluk, kesinlik, geri çağırma ve F1 skoru) göstermektedir.

```
Epoch 1/3, Loss: 0.31078528470808653
Epoch 2/3, Loss: 0.17894373042843967
Epoch 3/3, Loss: 0.1241051034881271
Test Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.86      0.84       992
           1       0.88      0.86      0.87      1230

    accuracy                           0.86      2222
   macro avg       0.86      0.86      0.86      2222
weighted avg       0.86      0.86      0.86      2222
```

Şekil 6.20 BERT 3 Epoch Sonucu Kayıp(Loss) Değerleri

```
Epoch 1/5, Loss: 0.3149868639306628
Epoch 2/5, Loss: 0.17888999917953136
Epoch 3/5, Loss: 0.11484281640339027
Epoch 4/5, Loss: 0.0710607649919262
Epoch 5/5, Loss: 0.049897515730892156
Test Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.87      0.85       992
           1       0.89      0.86      0.88      1230

    accuracy                           0.87      2222
   macro avg       0.86      0.87      0.86      2222
weighted avg       0.87      0.87      0.87      2222
```

Şekil 6.21 BERT 5 Epoch Sonucu Kayıp(Loss) Değerleri

7
Deneysel Sonuçlar

Türkçe metinlerde siber zorbalık tespiti için eğitilen modellerin uç noktalarının test edilmesi için veri setinin test kısmı kullanılmıştır. Şekil 7.1'de test veri setinin boyutları yer almaktadır. Burada özellikle tespit edilmesi zor olan ironi, sarkazm, çok anlamlılık ve alaycı dil gibi yöntemleri içeren bu veri seti, kullanılan modeller arasından en yüksek başarı oranı alınan SGDClassifier ve BERT ile test edilmiştir. Şekil 7.2 ve Şekil 7.3'te iki yöntemin de karışıklık matrisi gösterilmiştir.

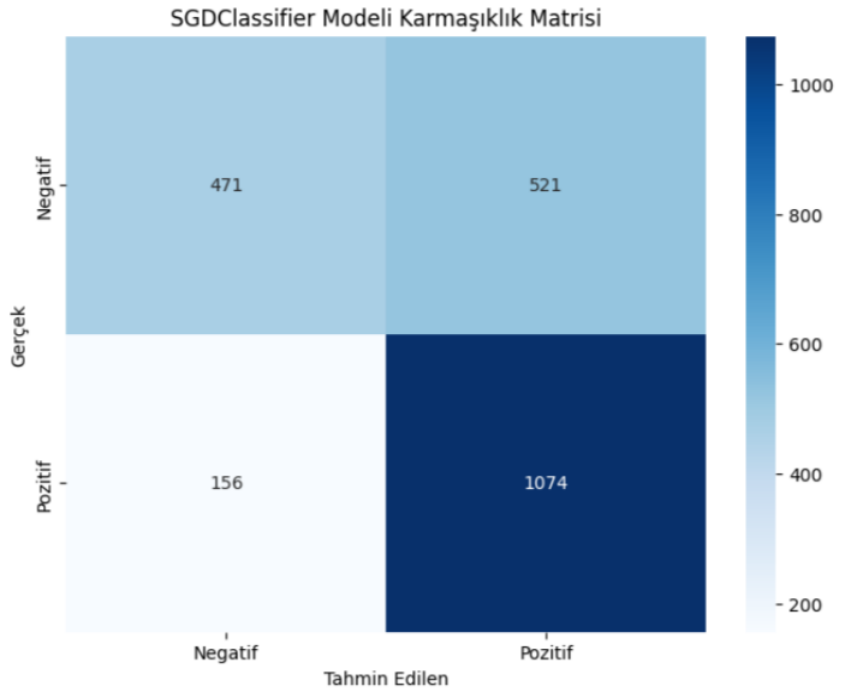
Sonuçlardan görüldüğü üzere BERT, SGDClassifier'dan daha doğru tahminler yapmıştır. Diğer taraftan başka etkenlere de bakılırsa BERT, SGDClassifier'a göre daha karmaşık yapıda olduğun için eğitimi uzun sürmüş ve daha fazla donanım ihtiyaçları olmuştur.

Sonuç olarak bu modeller kullanılırken projedeki gereksinimler iyi belirlenmeli ve ona göre model seçilmelidir.

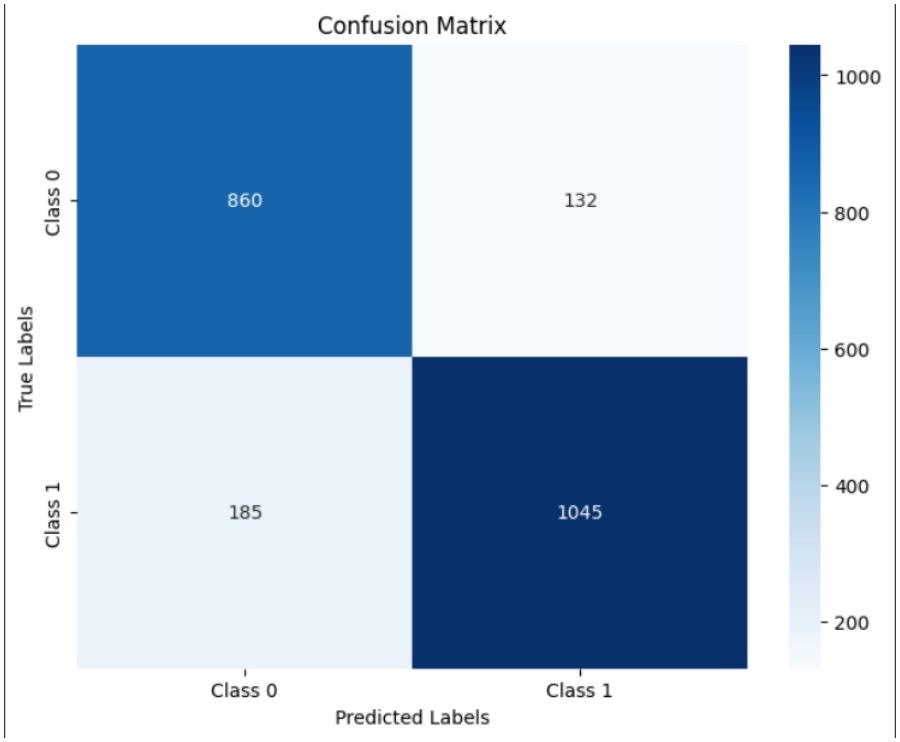
```
train_size = int(len(df)*0.8)
test_size = int(len(df)-train_size)

print("Eğitim Boyuyu=" ,train_size)
print("Test Boyuyu=" ,test_size)
```

```
Eğitim Boyuyu= 8887
Test Boyuyu= 2222
```

Şekil 7.1 Test Verisinin Boyutu



Şekil 7.2 SGDClassifier Modeli Test Karışıklık Matrisi



Şekil 7.3 BERT Test Karışıklık Matrisi

8
Performans Analizi

Projede kullanılan veri setine farklı makine ve derin öğrenme modelleriyle test veri seti ile aşağıdaki sonuçlar elde edilmiştir. Öncelikli amaç olarak burada farklı makine öğrenme teknikleri kullanılarak karşılaştırma yapılmış daha sonra derin öğrenme modellerinde farklı gömme vektörleri denemiştir. Bu gömme vektörleri ile her bir derin öğrenme modeliyle eğitim yapılmış ve çıkan sonuçlar ile farklı tekniklerde analiz yapılmaya olanak sağlamıştır.

Elde edile sonuçlar bu alanda daha önceden yapılan farklı çalışmaların sonuçları ile karşılaştırılmıştır. Bu çalışmalarda daha çok MultinomialNB ve Lojistik Regresyon makine öğrenme modelleriyle çalışılmıştır ve neredeyse projemizdeki sonuçlarla yaklaşık olarak aynı başarılar görülmüştür [10].

8.1 Modellerin Hesaplama Analizi

Kullanılan tüm modeller L4 GPU ile yapılmıştır. Makine öğrenme modelleri 15 katlı çaprazlama ile yaklaşık 15 dakika, derin öğrenme modelleri de 5 katlı çaprazlama ile 3 epoch, 32 batch size, 128 maksimum uzunluk ile yaklaşık 3 saat sürmüştür. Colab'da yaklaşık 96 işlem birimi ile yapılmıştır.

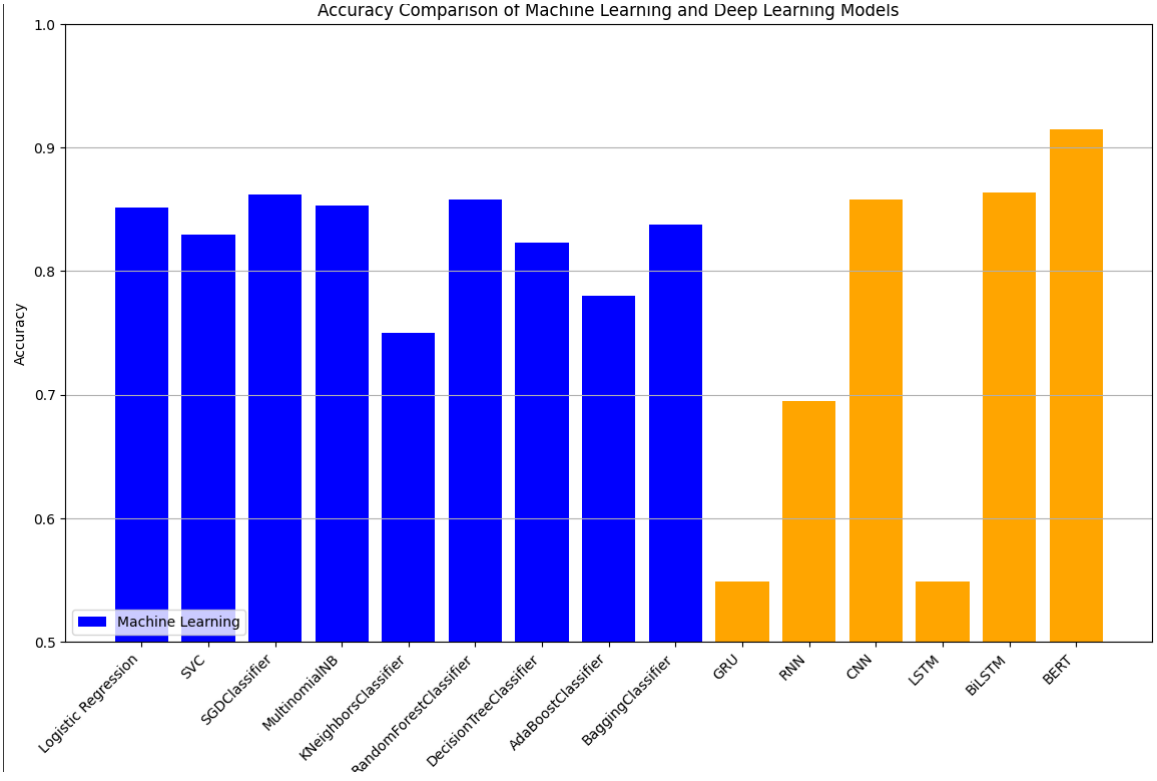
Şekil 8.1 ve 8.2'de kullanılan tüm yöntemlerin doğruluk değerlerinin karşılaştırması verilmiştir. Makine öğrenme modellerinde daha tutarlı ve benzer sonuçlar verirken derin öğrenme modellerinde fark oldukça fazladır. Kullanılan parametreler değiştirilerek, ince ayar yapılarak, düşük performans veren modellerin başarısı arttırılabilir. Bununla birlikte düşük performanslı modellerin veri seti ile uyumsuz olabileceği de göz önünde bulundurulmalıdır.

8.2 Gömme Matrislerinin Performans Analizi

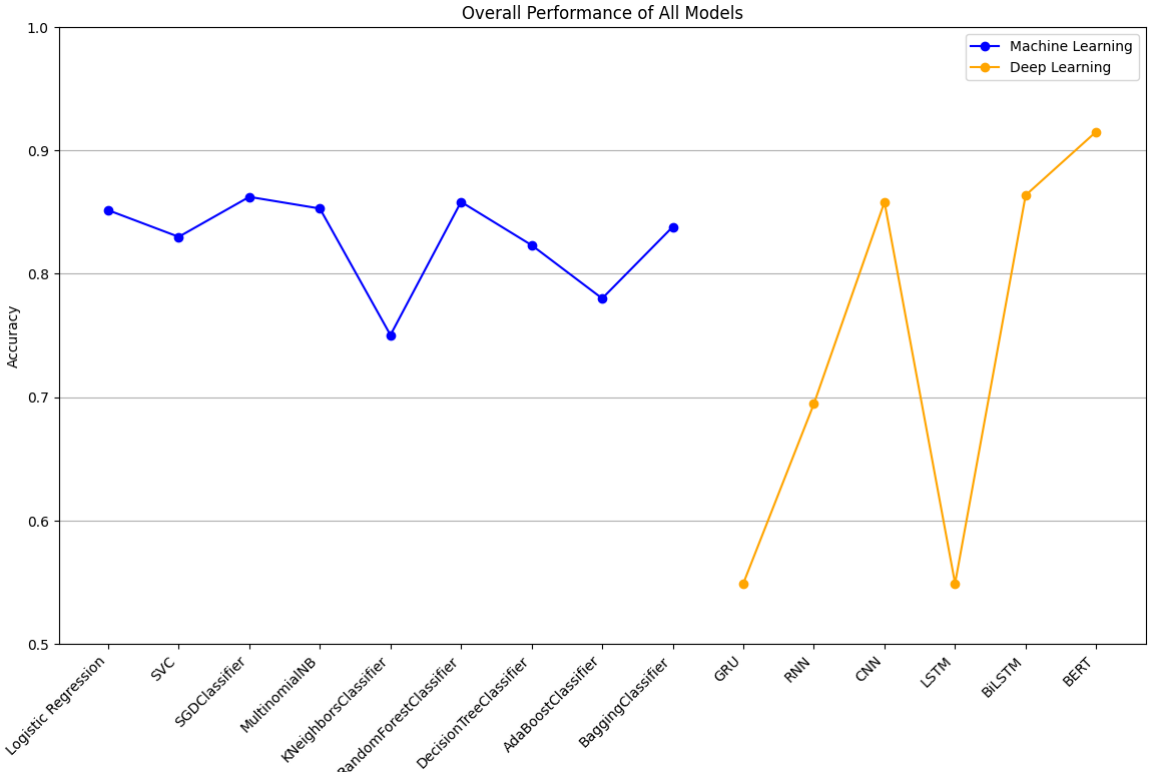
Kullanılan yöntemler önceden eğitilmiş gömme modelleridir. Glove modelinde 100 boyutlu bir vektörle çalışılmıştır. FastText'e kıyasla eğitim süresi daha kısa sürmüş olmasına rağmen genelinde FastText'den daha düşük bir başarı elde edilmiştir. Aynı zamanda gömme matrisleri ile keras da karşılaştırılmış ve FastText ile neredeyse aynı sonuçlar elde edilmiştir. Sonuç olarak derin öğrenme modellerinde kullanılan gömme matrislerin performansı Şekil 8.3'te gösterilmektedir.

8.3 BERT Modelinin Performansı

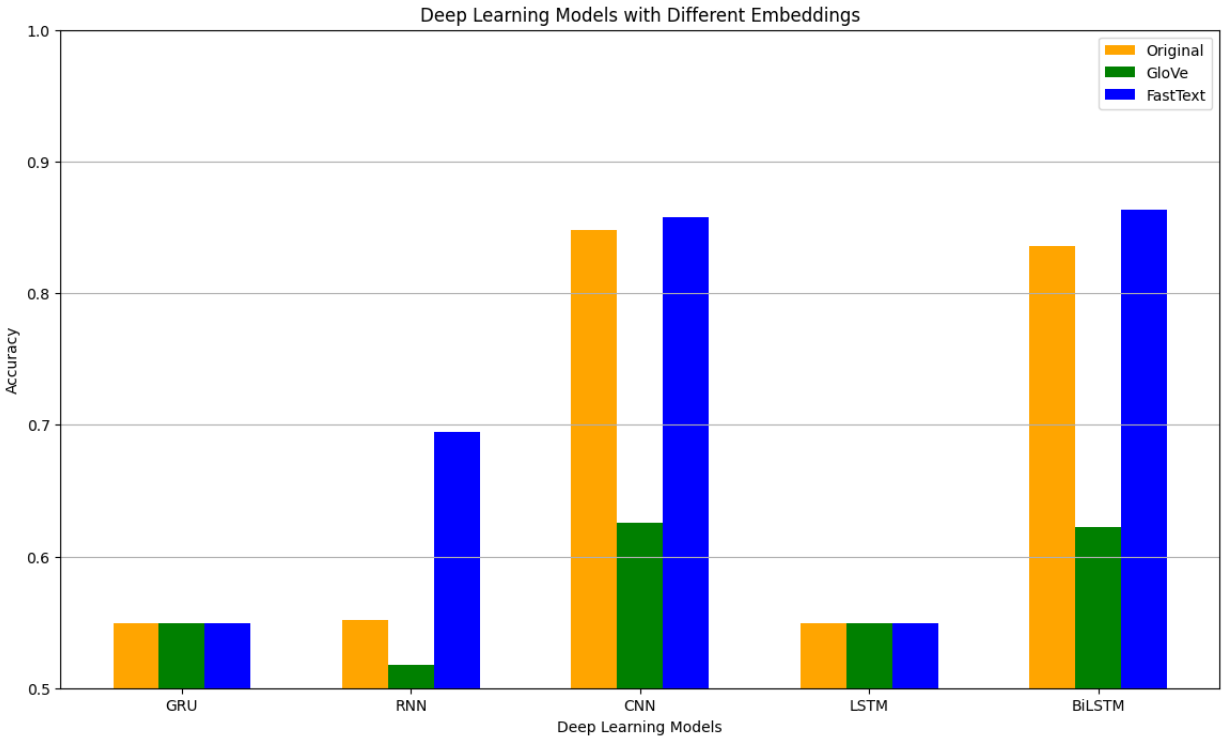
BERT modelinde fine-tune edilirken BertTokenizer ve BertForSequenceClassification modeli kullanılmıştır. Epoch değerleri olarak iki farklı değerde denenerek karşılaştırma yapılmıştır. Veri seti küçük-orta boyutta olduğu için overfitting olmaması adına 3 ve 5 epoch değerlerinde eğitim yapılmıştır. Model sonucundaki performans metrikleri Şekil 8.4 ve Şekil 8.5'te tablo olarak sunulmuştur.



Şekil 8.1 Makine Ve Derin Öğrenme Modellerinin Doğruluk Skoru Karşılaştırılması



Şekil 8.2 Tüm Modellerinin Doğruluk Skoru Karşılaştırılması



Şekil 8.3 Farklı Gömme Matrislerinin Deri Öğrenme Modelleri İle Performansı

Model Evaluation Metrics- Epoch 3

Metric	Value
Accuracy	0.8573357335733574
Precision	0.8583175658828065
Recall	0.8573357335733574
F1 Score	0.8575618032009215

Şekil 8.4 BERT 3 Epoch Sonucu Performans Metrikleri

Model Evaluation Metrics- Epoch 5

Metric	Value
Accuracy	0.8654365436543654
Precision	0.8665470063361048
Recall	0.8654365436543654
F1 Score	0.8656668467637658

Şekil 8.5 BERT 5 Epoch Sonucu Performans Metrikleri

9
Sonuç

Yapılan deneyler ve elde edilen bulgular makine ve derin öğrenme modelleri kullanılarak Türkçe metinlerde siber zorbalık tespit edilebileceğini göstermiştir.

Projede amaç farklı modeller arasındaki farklılıkları ortaya koyabilmek ve Türkçe'de yapılan çalışmalarda çeşitliliği arttırabilmektir. İlk aşamada veri seti ile ön işleme adımları yapılmıştır. Daha sonra belirlenen farklı yapılardaki makine öğrenme modelleriyle eğitim yapılmış ve sonuçlar izlenmiştir. İkinci aşama olarak derin öğrenme kısmına geçilmiş ve farklı gömme yöntemleri kullanılmıştır. Bu gömme matrisleriyle farklı derin öğrenme modeller eğitilmiş ve test edilmiştir. Son aşamada modeller kendi aralarında ve birbirleri ile kıyaslanarak en başarılı modeller tespit edilmiştir.

Elde edilen verilere göre makine öğrenmesi modelleri genel olarak benzer sonuçlarla %85 civarında bir başarı göstermiştir. Bununla birlikte farklı gömme modelleri ile derin öğrenme model çıktılarına bakıldığında Keras Embedding CNN, Keras Embedding BiLSTM, FastTexT CNN, FastTexT BiLSTM %85 civarında başarı göstermiştir. BERT'te ise yaklaşık olarak %95'in üstünde başarı elde edilmiştir.

Sonuç olarak başarılı derin öğrenme modelleri ve makine öğrenmesi modelleri benzer sonuçlarla %85 ve üzerinde bir başarı sağlamıştır. Gelecekteki çalışmalarda kullanılan parametreler değiştirilerek, ince ayar yapılarak, elde edilen performanslar daha da iyileştirlebilir. Ayrıca bu çalışmada denenmeyen modeller veya yöntemler denenerek çalışmanın kapsamı genişletilebilir ve başarısı arttırabilir.

Referanslar

[1] E. Yazğili and M. Baykara, "Türkçe metinlerde makine öğrenmesi yöntemleri ile siber zorbalık tespiti," *Gümüşhane Üniversitesi Fen Bilimleri Enstitüsü Dergisi*, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:247247091.

[2] O. Sevli and S. Sezgin, "Sosyal medya paylaşımlarında siber zorbalığın tespiti ve kategorizasyonuna yönelik makine öğrenmesine dayalı bir sınıflandırma (a machine learning-based classification on detection and categorization of cyberbullying in social media sharings)," Oct. 2022.

[3] Kaggle, *Kaggle*, https://www.kaggle.com, [Online; accessed 22-Sep-2023], 2023.

[4] M. YAKUT, Ç. ŞAHİN, and Y. Atay, "Machine and deep learning studies for cyberbullying detectionsiber zorbalık tespiti için makine öğrenmesi ve derin öğrenme çalışmaları," *Savunma Bilimleri Dergisi*, vol. 1, pp. 155–177, May 2023. DOI: 10.17134/khosbd.1087548.

[5] G. Nergiz and E. Avaroglu, "Türkçe sosyal medya yorumlarındaki siber zorbalığın derin öğrenme ile tespiti," *European Journal of Science and Technology*, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:245376508.

[6] Google, *Colaboratory faq*, https://research.google.com/colaboratory/faq.html, [Online; accessed 02-May-2023], 2023.

[7] A. Aizawa, "An information-theoretic perspective of tf-idf measures," *Information Processing & Management*, vol. 39, no. 1, pp. 45–65, 2003. DOI: 10.1016/S0306-4573(02)00021-3. [Online]. Available: https://doi.org/10.1016/S0306-4573(02)00021-3.

[8] J. Hani, M. Nashaat, M. Ahmed, Z. Emad, E. Amer, and A. Mohammed, "Social media cyberbullying detection using machine learning," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 10, no. 5, 2019. [Online]. Available: https://thesai.org/Publications/ViewPaper?Volume=10&Issue=5&Code=IJACSA&SerialNo=5.

[9] D. Anguita, L. Ghelardoni, A. Ghio, L. Oneto, and S. Ridella, "The 'k' in k-fold cross validation," in *ESANN 2012 Proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, Bruges, Belgium: i6doc.com, Apr. 2012, ISBN: 978-2-87419-049-0. [Online]. Available: http://www.i6doc.com/en/livre/?GCOI=28001100967420.

[10] F. Afşın, *Fork of doğal dil işleme metotları ile siber zorba*, https://www.kaggle.com/code/moneyshot495/fork-of-do-al-dil-leme-metotlar-ile-siber-zorba, [Online; accessed 08-June-2024], 2021.